# Otros tipos de datos en SQL Server

Ya explicamos que al crear una tabla debemos elegir la estructura adecuada, esto es, definir los campos y sus tipos más precisos, según el caso.

El tipo de dato especificado en la definición de cada campo indica los valores permitidos para cada uno de ellos.

Hasta ahora hemos visto 4 tipos de datos: varchar, integer, decimal y float. Hay más tipos, incluso, subtipos.

Los valores que podemos guardar son:

TEXTO: Para almacenar texto usamos cadenas de caracteres.

Las cadenas se colocan entre comillas simples. Podemos almacenar letras, símbolos y dígitos con los que no se realizan operaciones matemáticas, por ejemplo, códigos de identificación, números de documentos, números telefónicos.

SQL Server ofrece los siguientes tipos: char, nchar, varchar, nvarchar, text y ntext.

NUMEROS: Existe variedad de tipos numéricos para representar enteros, decimales, monedas.

Para almacenar valores enteros, por ejemplo, en campos que hacen referencia a cantidades, precios, etc., usamos el tipo integer (y sus subtipos: tinyint, smallint y bigint).

Para almacenar valores con decimales exactos, utilizamos: numeric o decimal (son equivalentes).

Para guardar valores decimales aproximados: float y real. Para almacenar valores monetarios: money y smallmoney.

FECHAS y HORAS: para guardar fechas y horas SQL Server dispone de 2 tipos: datetime y smalldatetime.

Existen otros tipos de datos que analizaremos en secciones próximas.

Entonces, cuando creamos una tabla y definir sus campos debemos elegir el tipo de dato más preciso. Por ejemplo, si necesitamos almacenar nombres usamos texto; si un campo numérico almacenará solamente valores enteros el tipo "integer" es más adecuado que, por ejemplo un "float"; si necesitamos almacenar precios, lo más lógico es utilizar el tipo "money" o "decimal".

A continuación analizaremos en detalle cada tipo de datos básicos.

# Tipo de dato (texto)

Ya explicamos que al crear una tabla debemos elegir la estructura adecuada, esto es, definir los campos y sus tipos más precisos, según el caso.

Para almacenar TEXTO usamos cadenas de caracteres.

Las cadenas se colocan entre comillas simples.

Podemos almacenar letras, símbolos y dígitos con los que no se realizan operaciones matemáticas, por ejemplo, códigos de identificación, números de documentos, números telefónicos.

Tenemos los siguientes tipos:

varchar(x): define una cadena de caracteres de longitud variable en la cual determinamos el máximo de caracteres con el argumento "x" que va entre paréntesis.

Si se omite el argumento coloca 1 por defecto. Su rango va de 1 a 8000 caracteres.

char(x): define una cadena de longitud fija determinada por el argumento "x". Si se omite el argumento coloca 1 por defecto. Su rango es de 1 a 8000 caracteres.

Si la longitud es invariable, es conveniente utilizar el tipo char; caso contrario, el tipo varchar.

Ocupa tantos bytes como se definen con el argumento "x".

"char" viene de character, que significa caracter en inglés.

text: guarda datos binarios de longitud variable, puede contener hasta 2000000000 caracteres. No admite argumento para especificar su longitud.

nvarchar(x): es similar a "varchar", excepto que permite almacenar caracteres Unicode, su rango va de 0 a 4000 caracteres porque se emplean 2 bytes por cada caracter.

nchar(x): es similar a "char" excpeto que acepta caracteres Unicode, su rango va de 0 a 4000 caracteres porque se emplean 2 bytes por cada caracter.

ntext: es similar a "text" excepto que permite almacenar caracteres Unicode, puede contener hasta 1000000000 caracteres. No admite argumento para especificar su longitud.

En general se usarán los 3 primeros.

Si intentamos almacenar en un campo una cadena de caracteres de mayor longitud que la definida, aparece un mensaje indicando tal situación y la sentencia no se ejecuta.

Por ejemplo, si definimos un campo de tipo varchar(10) y le asignamos la cadena 'Aprenda PHP' (11 caracteres), aparece un mensaje y la sentencia no se ejecuta.

Si ingresamos un valor numérico (omitiendo las comillas), lo convierte a cadena y lo ingresa como tal.

Por ejemplo, si en un campo definido como varchar(5) ingresamos el valor 12345, lo toma como si hubiésemos tipeado '12345', igualmente, si ingresamos el valor 23.56, lo convierte a '23.56'. Si el valor numérico, al ser convertido a cadena supera la longitud definida, aparece un mensaje de error y la sentencia no se ejecuta.

Es importante elegir el tipo de dato adecuado según el caso, el más preciso.

Para almacenar cadenas que varían en su longitud, es decir, no todos los registros tendrán la misma longitud en un campo determinado, se emplea "varchar" en lugar de "char".

Por ejemplo, en campos que guardamos nombres y apellidos, no todos los nombres y apellidos tienen la misma longitud.

Para almacenar cadenas que no varían en su longitud, es decir, todos los registros tendrán la misma longitud en un campo determinado, se emplea "char".

Por ejemplo, definimos un campo "codigo" que constará de 5 caracteres, todos los registros tendrán un código de 5 caracteres, ni más ni menos.

Para almacenar valores superiores a 8000 caracteres se debe emplear "text".

Tipo Bytes de almacenamiento

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

varchar(x) 0 a 8K

char(x) 0 a 8K

text 0 a 2GB

nvarchar(x) 0 a 8K

nchar(x) 0 a 8K

ntext 0 a 2GB

**Ejercicio:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

Eliminamos la tabla "visitantes", si existe:

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

drop table visitantes;

Creamos con la siguiente estructura:

create table visitantes(

nombre varchar(30),

edad integer,

sexo char(1),

domicilio varchar(30),

ciudad varchar(20),

telefono varchar(11)

);

Los campos "nombre", "domicilio" y "ciudad" almacenarán valores cuya longitud varía, por ello elegimos el tipo "varchar" y le damos a cada uno una longitud máxima estimando su tamaño. El campo "sexo" se define de tipo "char", porque necesitamos solamente 1 caracter "f" o "m", que siempre será fijo. El campo "telefono" también se define como varchar porque no todos los números telefónicos tienen la misma longitud.

Intentamos ingresar una cadena de mayor longitud que la definida:

insert into visitantes (nombre,edad,sexo,domicilio,ciudad,telefono)

values ('Juan Juarez',32,'masc','Avellaneda 789','Cordoba','4234567');

Aparece un mensaje de error y la sentencia no se ejecuta

Ingresamos un número telefónico olvidando las comillas, es decir, como un valor numérico:

insert into visitantes (nombre,edad,sexo,domicilio,ciudad,telefono)

values ('Marcela Morales',43,'f','Colon 456','Cordoba',4567890);

Lo convierte a cadena, veámoslo:

select nombre,edad,sexo,domicilio,ciudad,telefono from visitantes;

**Código:**

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

drop table visitantes;

create table visitantes(

nombre varchar(30),

edad integer,

sexo char(1),

domicilio varchar(30),

ciudad varchar(20),

telefono varchar(11)

);

insert into visitantes (nombre,edad,sexo,domicilio,ciudad,telefono)

values ('Juan Juarez',32,'masc','Avellaneda 789','Cordoba','4234567');

insert into visitantes (nombre,edad,sexo,domicilio,ciudad,telefono)

values ('Marcela Morales',43,'f','Colon 456','Cordoba',4567890);

select nombre,edad,sexo,domicilio,ciudad,telefono from visitantes;

**Problema:**

Una concesionaria de autos vende autos usados y almacena los datos de los autos en una tabla llamada "autos".

1- Elimine la tabla "autos" si existe:

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

drop table autos;

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

"matricula" como clave primaria:

create table autos(

matricula char(7),

marca varchar(20),

modelo char(4),

precio decimal(7),

primary key (matricula)

);

Hemos definido el campo "matricula" de tipo "char" y no "varchar" porque la cadena de caracteres siempre tendrá la misma longitud (7 caracteres). Lo mismo sucede con el campo "modelo", en el cual almacenaremos el año, necesitamos 4 caracteres fijos.

3- Ingrese los siguientes registros:

insert into autos

values('ACD1234','Fiat 128','1970',15000);

insert into autos

values('ACG2344','Renault 11','1990',40000);

insert into autos

values('BCD3334','Peugeot 505','1990',80000);

insert into autos

values('GCD1234','Renault Clio','1990',70000);

insert into autos

values('BCC3334','Renault Megane','1998',95000);

insert into autos

values('BVF5434','Fiat 128','1975',20000);

4- Seleccione todos los autos del año 1990:

select matricula,marca, modelo ,precio from autos

where modelo='1990';

**Solución:**

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

drop table autos;

create table autos(

matricula char(7),

marca varchar(20),

modelo char(4),

precio decimal(7),

primary key (matricula)

);

insert into autos

values('ACD1234','Fiat 128','1970',15000);

insert into autos

values('ACG2344','Renault 11','1990',40000);

insert into autos

values('BCD3334','Peugeot 505','1990',80000);

insert into autos

values('GCD1234','Renault Clio','1990',70000);

insert into autos

values('BCC3334','Renault Megane','1998',95000);

insert into autos

values('BVF5434','Fiat 128','1975',20000);

select matricula,marca, modelo ,precio from autos

where modelo='1990';

# Tipo de dato (numérico)

Ya explicamos que al crear una tabla debemos elegir la estructura adecuada, esto es, definir los campos y sus tipos más precisos, según el caso.

Para almacenar valores NUMERICOS SQL Server dispone de varios tipos.

Para almacenar valores ENTEROS, por ejemplo, en campos que hacen referencia a cantidades, usamos:

integer o int: su rango es de -2000000000 a 2000000000 aprox. El tipo "integer" tiene subtipos:

- smallint: Puede contener hasta 5 digitos. Su rango va desde –32000 hasta 32000 aprox.

- tinyint: Puede almacenar valores entre 0 y 255.

- bigint: De –9000000000000000000 hasta 9000000000000000000 aprox.

Para almacenar valores numéricos EXACTOS con decimales, especificando la cantidad de cifras a la izquierda y derecha del separador decimal, utilizamos:

decimal o numeric (t,d): Pueden tener hasta 38 digitos, guarda un valor exacto. El primer argumento indica el total de dígitos (el punto decimal no cuenta) y el segundo, la cantidad de decimales.

Por ejemplo, si queremos almacenar valores entre -99.99 y 99.99 debemos definir el campo como tipo "decimal(4,2)". Si no se indica el valor del segundo argumento, por defecto es "0". Por ejemplo, si definimos "decimal(4)" se pueden guardar valores entre -9999 y 9999.

El rango depende de los argumentos, también los bytes que ocupa.

Se utiliza el punto como separador de decimales.

Si ingresamos un valor con más decimales que los permitidos, redondea al más cercano; por ejemplo, si definimos "decimal(4,2)" e ingresamos el valor "12.686", guardará "12.69", redondeando hacia arriba; si ingresamos el valor "12.682", guardará "12.67", redondeando hacia abajo.

Para almacenar valores numéricos APROXIMADOS con decimales utilizamos:

float: De 1.79E+308 hasta 1.79E+38. Guarda valores aproximados.

real: Desde 3.40E+308 hasta 3.40E+38. Guarda valores aproximados.

Para almacenar valores MONETARIOS empleamos:

money: Puede tener hasta 19 digitos y sólo 4 de ellos puede ir luego del separador decimal; entre –900000000000000.5808 aprox y 900000000000000.5807.

smallmoney: Entre –200000.3648 y 200000.3647 aprox.

Para todos los tipos numéricos:

- si intentamos ingresar un valor fuera de rango, no lo permite.

- si ingresamos una cadena, SQL Server intenta convertirla a valor numérico, si dicha cadena consta solamente de dígitos, la conversión se realiza, luego verifica si está dentro del rango, si es así, la ingresa, sino, muestra un mensaje de error y no ejecuta la sentencia. Si la cadena contiene caracteres que SQL Server no puede convertir a valor numérico, muestra un mensaje de error y la sentencia no se ejecuta.

Por ejemplo, definimos un campo de tipo decimal(5,2), si ingresamos la cadena '12.22', la convierte al valor numérico 12.22 y la ingresa; si intentamos ingresar la cadena '1234.56', la convierte al valor numérico 1234.56, pero como el máximo valor permitido es 999.99, muestra un mensaje indicando que está fuera de rango. Si intentamos ingresar el valor '12y.25', SQL Server no puede realizar la conversión y muestra un mensaje de error.

Es importante elegir el tipo de dato adecuado según el caso, el más preciso. Por ejemplo, si un campo numérico almacenará valores positivos menores a 255, el tipo "int" no es el más adecuado, conviene el tipo "tinyint", de esta manera usamos el menor espacio de almacenamiento posible.

Si vamos a guardar valores monetarios menores a 200000 conviene emplear "smallmoney" en lugar de "money".

Tipo Bytes de almacenamiento

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

int 4

smallint 2

tinyint 1

bigint 8

decimal 2 a 17

float 4 u 8

real 4 u 8

money 8

smallmoney 4

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

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

drop table libros;

Creamos la tabla con la siguiente estructura:

create table libros(

codigo smallint identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(15),

precio smallmoney,

cantidad tinyint

);

Note que definimos el campo "codigo" de tipo "smallint", esto es porque estimamos que no tendremos más de 30000 libros. Si necesitáramos un rango mayor podemos emplear "int".

Como en el campo "precio" no almacenaremos valores mayores a 200000, definimos el campo de tipo "smallmoney".

También podemos definirlo de tipo "decimal(5,2)" porque el máximo precio no superará los 999.99.

El tipo "float" no es el más adecuado para representar precios porque no es exacto y muestra muchos decimales innecesarios.

Como los valores para el campo "cantidad" no superarán los 255, definimos el campo de tipo "tinyint". Si estimamos que tendremos más cantidad de libros podemos emplear "smallint" que tiene un rango mayor; no es adecuado usar int (cuyo rango llega hasta 4000 millones aprox.), porque ocuparíamos más espacio (4 bytes).

Analicemos la inserción de datos numéricos.

Intentemos ingresar un valor fuera del rango definido, una cantidad que supera el rango del tipo "tinyint", el valor 260:

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

values('El aleph','Borges','Emece',25.60,260);

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

Intentamos ingresar un precio que supera el rango del tipo "smallmoney", el valor 250000:

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

values('El aleph','Borges','Emece',250000,100);

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

Intentamos ingresar una cadena que SQL Server no pueda convertir a valor numérico en el campo "precio" (error):

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

values('Uno','Richard Bach','Planeta','a50.30',100);

Ingresamos una cadena en el campo "cantidad":

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

values('Uno','Richard Bach','Planeta',50.30,'100');

Lo convierte a valor numérico.

**Código:**

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

drop table libros;

create table libros(

codigo smallint identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(15),

precio smallmoney,

cantidad tinyint

);

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

values('El aleph','Borges','Emece',25.60,260);

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

values('El aleph','Borges','Emece',250000,100);

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

values('Uno','Richard Bach','Planeta','a50.30',100);

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

values('Uno','Richard Bach','Planeta',50.30,'100');

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

**Problema:**

Un banco tiene registrados las cuentas corrientes de sus clientes en una tabla llamada "cuentas".

La tabla contiene estos datos:

Número de Cuenta Documento Nombre Saldo

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

1234 25666777 Pedro Perez 500000.60

2234 27888999 Juan Lopez -250000

3344 27888999 Juan Lopez 4000.50

3346 32111222 Susana Molina 1000

1- Elimine la tabla "cuentas" si existe:

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

drop table cuentas;

2- Cree la tabla eligiendo el tipo de dato adecuado para almacenar los datos descriptos arriba:

- Número de cuenta: entero, no nulo, no puede haber valores repetidos, clave primaria;

- Documento del propietario de la cuenta: cadena de caracteres de 8 de longitud (siempre 8), no nulo;

- Nombre del propietario de la cuenta: cadena de caracteres de 30 de longitud,

- Saldo de la cuenta: valores altos con decimales.

3- Ingrese los siguientes registros:

insert into cuentas(numero,documento,nombre,saldo)

values('1234','25666777','Pedro Perez',500000.60);

insert into cuentas(numero,documento,nombre,saldo)

values('2234','27888999','Juan Lopez',-250000);

insert into cuentas(numero,documento,nombre,saldo)

values('3344','27888999','Juan Lopez',4000.50);

insert into cuentas(numero,documento,nombre,saldo)

values('3346','32111222','Susana Molina',1000);

Note que hay dos cuentas, con distinto número de cuenta, de la misma persona.

4- Seleccione todos los registros cuyo saldo sea mayor a "4000" (2 registros)

5- Muestre el número de cuenta y saldo de todas las cuentas cuyo propietario sea "Juan Lopez" (2 registros)

6- Muestre las cuentas con saldo negativo (1 registro)

7- Muestre todas las cuentas cuyo número es igual o mayor a "3000" (2 registros):

select numero,documento,nombre,saldo from cuentas

where numero>=3000;

**Solución:**

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

drop table cuentas;

create table cuentas(

numero int not null,

documento char(8),

nombre varchar(30),

saldo money,

primary key (numero)

);

insert into cuentas(numero,documento,nombre,saldo)

values('1234','25666777','Pedro Perez',500000.60);

insert into cuentas(numero,documento,nombre,saldo)

values('2234','27888999','Juan Lopez',-250000);

insert into cuentas(numero,documento,nombre,saldo)

values('3344','27888999','Juan Lopez',4000.50);

insert into cuentas(numero,documento,nombre,saldo)

values('3346','32111222','Susana Molina',1000);

select numero,documento,nombre,saldo from cuentas

where saldo<4000;

select numero,saldo from cuentas

where nombre='Juan Lopez';

select numero,documento,nombre,saldo from cuentas

where saldo<0;

select numero,documento,nombre,saldo from cuentas

where numero>=3000;

# Tipo de dato (fecha y hora)

Ya explicamos que al crear una tabla debemos elegir la estructura adecuada, esto es, definir los campos y sus tipos más precisos, según el caso.

Para almacenar valores de tipo FECHA Y HORA SQL Server dispone de dos tipos:

datetime: puede almacenar valores desde 01 de enero de 1753 hasta 31 de diciembre de 9999.

smalldatetime: el rango va de 01 de enero de 1900 hasta 06 de junio de 2079.

Las fechas se ingresan entre comillas simples.

Para almacenar valores de tipo fecha se permiten como separadores "/", "-" y ".".

SQL Server reconoce varios formatos de entrada de datos de tipo fecha. Para establecer el orden de las partes de una fecha (dia, mes y año) empleamos "set dateformat". Estos son los formatos:

-mdy: 4/15/96 (mes y día con 1 ó 2 dígitos y año con 2 ó 4 dígitos),

-myd: 4/96/15,

-dmy: 15/4/1996

-dym: 15/96/4,

-ydm: 96/15/4,

-ydm: 1996/15/4,

Para ingresar una fecha con formato "día-mes-año", tenemos que escribir previamente:

set dateformat dmy;

El formato por defecto es "mdy".

Todos los valores de tipo "datetime" se muestran en formato "año-mes-día hora:minuto:segundo .milisegundos", independientemente del formato de ingreso.

Podemos ingresar una fecha, sin hora, en tal caso la hora se guarda como "00:00:00". Por ejemplo, si ingresamos '25-12-01' (año de 2 dígitos), lo mostrará así: '2001-12-25 00:00:00.000'.

Podemos ingresar una hora sin fecha, en tal caso, coloca la fecha "1900-01-01". Por ejemplo, si ingresamos '10:15', mostrará '1900-01-01 10:15.000'.

Podemos emplear los operadores relacionales vistos para comparar fechas.

Tipo Bytes de almacenamiento

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

datetime 8

smalldatetime 4

**Ejercicio:**

Una empresa almacena los datos de sus empleados en una tabla "empleados".

Eliminamos la tabla, si existe:

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

drop table empleados;

Creamos la tabla eligiendo el tipo de dato adecuado para cada campo:

create table empleados(

nombre varchar(20),

documento char(8),

fechaingreso datetime

);

Indicamos el formato de la fecha para que guarde día, mes y año:

set dateformat dmy;

Ingresamos algunos registros:

insert into empleados values('Ana Gomez','22222222','12-01-1980');

insert into empleados values('Bernardo Huerta','23333333','15-03-81');

insert into empleados values('Carla Juarez','24444444','20/05/1983');

insert into empleados values('Daniel Lopez','25555555','2.5.1990');

El segundo registro ingresado tiene 2 dígitos correspondientes al año; en el tercero empleamos la barra ('/') como separador y en el cuarto empleamos como separador el punto ('.') y colocamos un sólo dígito en la part del día y el mes.

Recuperamos los registros:

Select nombre,documento,fechaingreso from empleados;

Note que el formato de visualización es "y-m-d".

Mostramos los datos de los empleados cuya fecha de ingreso es anterior a '01-01-1985':

select nombre,documento,fechaingreso from empleados where fechaingreso<'01-01-1985';

Actualizamos el nombre a "Maria Carla Juarez' del empleado cuya fecha de ingreso es igual a '20/05/1983':

update empleados set nombre='Maria Carla Juarez' where fechaingreso='20.5.83';

Veamos si se actualizó:

select nombre,documento,fechaingreso from empleados;

Borramos los empleados cuya fecha de ingreso es distinta a '20.5.83':

delete from empleados where fechaingreso<>'20/05/1983';

Veamos si se eliminaron:

select nombre,documento,fechaingreso from empleados;

**Código:**

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

drop table empleados;

create table empleados(

nombre varchar(20),

documento char(8),

fechaingreso datetime

);

set dateformat dmy;

insert into empleados values('Ana Gomez','22222222','12-01-1980');

insert into empleados values('Bernardo Huerta','23333333','15-03-81');

insert into empleados values('Carla Juarez','24444444','20/05/1983');

insert into empleados values('Daniel Lopez','25555555','2.5.1990');

select nombre,documento,fechaingreso from empleados;

select nombre,documento,fechaingreso from empleados where fechaingreso<'01-01-1985';

update empleados set nombre='Maria Carla Juarez' where fechaingreso='20.5.83';

select nombre,documento,fechaingreso from empleados;

delete from empleados where fechaingreso<>'20/05/1983';

select nombre,documento,fechaingreso from empleados;

**Problema:**

Una facultad almacena los datos de sus alumnos en una tabla denominada "alumnos".

1- Elimine la tabla, si existe:

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

drop table alumnos;

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

create table alumnos(

apellido varchar(30),

nombre varchar(30),

documento char(8),

domicilio varchar(30),

fechaingreso datetime,

fechanacimiento datetime

);

3- Fije el formato para entrada de datos de tipo fecha para que acepte valores "día-mes-año":

set dateformat 'dmy';

4- Ingrese un alumno empleando distintos separadores para las fechas:

insert into alumnos values('Gonzalez','Ana','22222222','Colon 123','10-08-1990','15/02/1972');

5- Ingrese otro alumno empleando solamente un dígito para día y mes y 2 para el año:

insert into alumnos values('Juarez','Bernardo','25555555','Sucre 456','03-03-1991','15/02/1972');

6- Ingrese un alumnos empleando 2 dígitos para el año de la fecha de ingreso y "null" en

"fechanacimiento":

insert into alumnos values('Perez','Laura','26666666','Bulnes 345','03-03-91',null);

7- Intente ingresar un alumno con fecha de ingreso correspondiente a "15 de marzo de 1990" pero en orden incorrecto "03-15-90":

insert into alumnos values('Lopez','Carlos','27777777','Sarmiento 1254','03-15-1990',null);

Aparece un mensaje de error porque lo lee con el formato día, mes y año y no reconoce el mes 15.

8- Muestre todos los alumnos que ingresaron antes del '1-1-91'. 1 registro.

9- Muestre todos los alumnos que tienen "null" en "fechanacimiento":

select apellido, nombre, documento, domicilio , fechaingreso, fechanacimiento

from alumnos where fechanacimiento is null;

1 registro.

10- Intente ingresar una fecha de ingreso omitiendo los separadores:

insert into alumnos values('Rosas','Romina','28888888','Avellaneda 487','03151990',null);

No lo acepta.

11- Setee el formato de entrada de fechas para que acepte valores "mes-dia-año".

12- Ingrese el registro del punto 7.

**Solución:**

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

drop table alumnos;

create table alumnos(

apellido varchar(30),

nombre varchar(30),

documento char(8),

domicilio varchar(30),

fechaingreso datetime,

fechanacimiento datetime

);

set dateformat 'dmy';

insert into alumnos values('Gonzalez','Ana','22222222','Colon 123','10-08-1990','15/02/1972');

insert into alumnos values('Juarez','Bernardo','25555555','Sucre 456','03-03-1991','15/02/1972');

insert into alumnos values('Perez','Laura','26666666','Bulnes 345','03-03-91',null);

insert into alumnos values('Lopez','Carlos','27777777','Sarmiento 1254','03-15-1990',null);

select apellido, nombre, documento, domicilio , fechaingreso, fechanacimiento

from alumnos where fechaingreso<'1-1-91';

select apellido, nombre, documento, domicilio , fechaingreso, fechanacimiento

from alumnos where fechanacimiento is null;

insert into alumnos values('Rosas','Romina','28888888','Avellaneda 487','03151990',null);

set dateformat 'mdy';

insert into alumnos values('Lopez','Carlos','27777777','Sarmiento 1254','03-15-1990',null);

# Valores por defecto (default)

Hemos visto que si al insertar registros no se especifica un valor para un campo que admite valores nulos, se ingresa automaticamente "null" y si el campo está declarado "identity", se inserta el siguiente de la secuencia. A estos valores se les denomina valores por defecto o predeterminados.

Un valor por defecto se inserta cuando no está presente al ingresar un registro y en algunos casos en que el dato ingresado es inválido.

Para campos de cualquier tipo no declarados "not null", es decir, que admiten valores nulos, el valor por defecto es "null". Para campos declarados "not null", no existe valor por defecto, a menos que se declare explícitamente con la cláusula "default".

Para todos los tipos, excepto los declarados "identity", se pueden explicitar valores por defecto con la cláusula "default".

Podemos establecer valores por defecto para los campos cuando creamos la tabla. Para ello utilizamos "default" al definir el campo. Por ejemplo, queremos que el valor por defecto del campo "autor" de la tabla "libros" sea "Desconocido" y el valor por defecto del campo cantidad" sea "0":

create table libros(

codigo int identity,

titulo varchar(40),

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

editorial varchar(20),

precio decimal(5,2),

cantidad tinyint default 0

);

Si al ingresar un nuevo registro omitimos los valores para el campo "autor" y "cantidad", Sql Server insertará los valores por defecto; el siguiente valor de la secuencia en "codigo", en "autor" colocará "Desconocido" y en cantidad "0".

Entonces, si al definir el campo explicitamos un valor mediante la cláusula "default", ése será el valor por defecto.

Ahora, al visualizar la estructura de la tabla con "sp\_columns" podemos entender lo que informa la columna "COLUMN\_DEF", muestra el valor por defecto del campo.

También se puede utilizar "default" para dar el valor por defecto a los campos en sentencias "insert", por ejemplo:

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

values ('El gato con botas',default,default,100);

Si todos los campos de una tabla tienen valores predeterminados (ya sea por ser "identity", permitir valores nulos o tener un valor por defecto), se puede ingresar un registro de la siguiente manera:

insert into libros default values;

La sentencia anterior almacenará un registro con los valores predetermiandos para cada uno de sus campos.

Entonces, la cláusula "default" permite especificar el valor por defecto de un campo. Si no se explicita, el valor por defecto es "null", siempre que el campo no haya sido declarado "not null".

Los campos para los cuales no se ingresan valores en un "insert" tomarán los valores por defecto:

- si tiene el atributo "identity": el valor de inicio de la secuencia si es el primero o el siguiente valor de la secuencia, no admite cláusula "default";

- si permite valores nulos y no tiene cláusula "default", almacenará "null";

- si está declarado explícitamente "not null", no tiene valor "default" y no tiene el atributo "identity", no hay valor por defecto, así que causará un error y el "insert" no se ejecutará.

- si tiene cláusula "default" (admita o no valores nulos), el valor definido como predeterminado;

- para campos de tipo fecha y hora, si omitimos la parte de la fecha, el valor predeterminado para la fecha es "1900-01-01" y si omitimos la parte de la hora, "00:00:00".

Un campo sólo puede tener un valor por defecto. Una tabla puede tener todos sus campos con valores por defecto. Que un campo tenga valor por defecto no significa que no admita valores nulos, puede o no admitirlos.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

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

drop table libros;

Creamos la tabla definiendo un valor por defecto para el campo "autor" y otro para el campo "cantidad":

create table libros(

codigo int identity,

titulo varchar(40),

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

editorial varchar(20),

precio decimal(5,2),

cantidad tinyint default 0

);

Ingresamos un registro omitiendo los valores para el campo "autor" y "cantidad":

insert into libros (titulo,editorial,precio)

values('Java en 10 minutos','Paidos',50.40);

SQL Server ingresará el registro con el siguiente valor de la secuencia en "codigo", con el título, editorial y precio ingresados, en "autor" colocará "Desconocido" y en cantidad "0":

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

Si ingresamos un registro sin valor para el campo "precio", que admite valores nulos, se ingresará "null" en ese campo:

insert into libros (titulo,editorial)

values('Aprenda PHP','Siglo XXI');

select \*from libros;

Visualicemos la estructura de la tabla:

sp\_columns libros;

La columna "COLUMN\_DEF", muestra el valor por defecto de cada campo.

Podemos emplear "default" para dar el valor por defecto a algunos campos:

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

values ('El gato con botas',default,default,100);

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

Como todos los campos de "libros" tienen valores predeterminados, podemos tipear:

insert into libros default values;

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

La sentencia anterior almacenará un registro con los valores predetermiandos para cada uno de sus campos.

Que un campo tenga valor por defecto no significa que no admita valores nulos, puede o no admitirlos. Podemos ingresar el valor "null" en el campo "cantidad":

insert into libros (titulo,autor,cantidad)

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

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

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

editorial varchar(20),

precio decimal(5,2),

cantidad tinyint default 0

);

insert into libros (titulo,editorial,precio)

values('Java en 10 minutos','Paidos',50.40);

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

insert into libros (titulo,editorial)

values('Aprenda PHP','Siglo XXI');

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

sp\_columns libros;

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

values ('El gato con botas',default,default,100);

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

insert into libros default values;

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

insert into libros (titulo,autor,cantidad)

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

**Problema:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

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

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

drop table visitantes;

2- Cree la tabla con la siguiente estructura:

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

ciudad varchar(20) default 'Cordoba',

telefono varchar(11),

mail varchar(30) default 'no tiene',

montocompra decimal (6,2)

);

4- Vea la información de las columnas "COLUMN\_DEF" y "IS\_NULLABLE":

sp\_columns visitantes;

5- Ingrese algunos registros sin especificar valores para algunos campos para ver cómo opera la cláusula "default":

insert into visitantes (nombre, domicilio, montocompra)

values ('Susana Molina','Colon 123',59.80);

insert into visitantes (nombre, edad, ciudad, mail)

values ('Marcos Torres',29,'Carlos Paz','marcostorres@hotmail.com');

select nombre, edad, sexo, domicilio , ciudad , telefono, mail, montocompra from visitantes;

6- Use la palabra "default" para ingresar valores en un insert.

7- Ingrese un registro con "default values".

**Solución:**

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

drop table visitantes;

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

ciudad varchar(20) default 'Cordoba',

telefono varchar(11),

mail varchar(30) default 'no tiene',

montocompra decimal (6,2)

);

sp\_columns visitantes;

insert into visitantes (nombre, domicilio, montocompra)

values ('Susana Molina','Colon 123',59.80);

insert into visitantes (nombre, edad, ciudad, mail)

values ('Marcos Torres',29,'Carlos Paz','marcostorres@hotmail.com');

select nombre, edad, sexo, domicilio , ciudad , telefono, mail, montocompra

from visitantes;

insert into visitantes

values ('Marcelo Morales',38,default,default,default,'4255232','marcelomorales@hotmail.com',default);

insert into visitantes default values;

select nombre, edad, sexo, domicilio , ciudad , telefono, mail, montocompra

from visitantes;

# Ingresar algunos campos (insert into)

Hemos aprendido a ingresar registros listando todos los campos y colocando valores para todos y cada uno de ellos luego de "values".

Si ingresamos valores para todos los campos, podemos omitir la lista de nombres de los campos.

Por ejemplo, si tenemos creada la tabla "libros" con los campos "titulo", "autor" y "editorial", podemos ingresar un registro de la siguiente manera:

insert into libros

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

También es posible ingresar valores para algunos campos. Ingresamos valores solamente para los campos "titulo" y "autor":

insert into libros (titulo, autor)

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

SQL Server almacenará el valor "null" en el campo "editorial", para el cual no hemos explicitado un valor.

Al ingresar registros debemos tener en cuenta:

- la lista de campos debe coincidir en cantidad y tipo de valores con la lista de valores luego de "values". Si se listan más (o menos) campos que los valores ingresados, aparece un mensaje de error y la sentencia no se ejecuta.

- si ingresamos valores para todos los campos podemos obviar la lista de campos.

- podemos omitir valores para los campos que NO hayan sido declarados "not null", es decir, que permitan valores nulos (se guardará "null"); si omitimos el valor para un campo "not null", la sentencia no se ejecuta.

- se DEBE omitir el valor para el campo"identity". Salvo que identity\_insert este en on.

- se pueden omitir valores para campos declarados "not null" siempre que tengan definido un valor por defecto con la cláusula "default" (tema que veremos a continuación).

- se pueden separar las listas de datos con comas ingresando tantos registros como listas de valores.

**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(40) not null,

autor varchar(30),

editorial varchar(15)

);

Si ingresamos valores para todos los campos, podemos omitir la lista de campos:

insert into libros

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

Podemos ingresar valores para algunos de los campos:

insert into libros (titulo, autor)

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

No podemos omitir el valor para un campo declarado "not null", como el campo "titulo":

insert into libros (autor,editorial)

values ('Lewis Carroll','Planeta');

Aparece un mensaje y la inserción no se realiza.

Veamos cómo SQL Server almacenó los registros:

Select codigo,titulo,autor,editorial from libros;

Podemos entrar varios registros en un único insert:

insert into libros (titulo, autor)

values ('El Quijote','Cervantes'), ('Cien años de Soledad','Marquez');

**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)

);

insert into libros

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

insert into libros (titulo, autor)

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

insert into libros (autor,editorial)

values ('Lewis Carroll','Planeta');

Select codigo,titulo,autor,editorial from libros;

insert into libros (titulo, autor)

values ('El Quijote','Cervantes'), ('Cien años de Soledad','Marquez');

Select codigo,titulo,autor,editorial from libros;

**Problema:**

Un banco tiene registrados las cuentas corrientes de sus clientes en una tabla llamada "cuentas".

1- Elimine la tabla "cuentas" si existe:

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

drop table cuentas;

2- Cree la tabla :

create table cuentas(

numero int identity,

documento char(8) not null,

nombre varchar(30),

saldo money

);

3- Ingrese un registro con valores para todos sus campos, inclusive el campo identity, omitiendo la lista de campos (error, no se debe ingresar para el campo identity):

insert into cuentas

values (1,'25666777','Juan Perez',2500.50);

4- Ingrese un registro con valores para todos sus campos omitiendo la lista de campos (excepto el campo "identity"):

insert into cuentas

values ('25666777','Juan Perez',2500.50);

5- Ingrese un registro omitiendo algún campo que admitan valores nulos.

6- Intente ingresar un registro con valor para el campo "numero" (error):

insert into cuentas (numero,documento,nombre,saldo)

values (5,'28999777','Luis Lopez',34000);

7- Intente ingresar un registro listando 3 campos y colocando 4 valores (error)

8- Intente ingresar un registro sin valor para el campo "documento" (error)

9- Vea los registros ingresados:

select numero,documento,nombre,saldo from libros;

10- Ingrese varios registros en una única sentencia insert:

insert into cuentas (documento,nombre,saldo)

values ('28999888','Luis Pérez',34034), ('45634533'.'Juan Domínguez',45);

9- Vea los registros ingresados:

select numero,documento,nombre,saldo from libros;

**Solución:**

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

drop table cuentas;

create table cuentas(

numero int identity,

documento char(8) not null,

nombre varchar(30),

saldo money

);

insert into cuentas

values (1,'25666777','Juan Perez',2500.50);

insert into cuentas

values ('25666777','Juan Perez',2500.50);

insert into cuentas (documento,saldo)

values ('28999777',-5500);

insert into cuentas (numero,documento,nombre,saldo)

values (5,'28999777','Luis Lopez',34000);

insert into cuentas (numero,documento,nombre)

values (3344,'28999777','Luis Lopez',34000);

insert into cuentas (nombre, saldo)

values ('Luis Lopez',34000);

select numero,documento,nombre,saldo from libros;

insert into cuentas (documento,nombre,saldo)

values ('28999888','Luis Pérez',34034), ('45634533'.'Juan Domínguez',45);

select numero,documento,nombre,saldo from libros;

# Columnas calculadas (operadores aritméticos y de concatenación)

Aprendimos que los operadores son símbolos que permiten realizar distintos tipos de operaciones.

Dijimos que SQL Server tiene 4 tipos de operadores: 1) relacionales o de comparación (los vimos), 2) lógicos (lo veremos más adelante, 3) aritméticos y 4) de concatenación.

Los operadores aritméticos permiten realizar cálculos con valores numéricos.

Son: multiplicación (\*), división (/) y módulo (%) (el resto de dividir números enteros), suma (+) y resta (-).

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un campo de una tabla.

Si queremos ver los títulos, precio y cantidad de cada libro escribimos la siguiente sentencia:

select titulo,precio,cantidad

from libros;

Si queremos saber el monto total en dinero de un título podemos multiplicar el precio por la cantidad por cada título, pero también podemos hacer que SQL Server realice el cálculo y lo incluya en una columna extra en la salida:

select titulo, precio,cantidad,

precio\*cantidad

from libros;

Si queremos saber el precio de cada libro con un 10% de descuento podemos incluir en la sentencia los siguientes cálculos:

select titulo,precio,

precio-(precio\*0.1)

from libros;

También podemos actualizar los datos empleando operadores aritméticos:

update libros set precio=precio-(precio\*0.1);

Todas las operaciones matemáticas retornan "null" en caso de error. Ejemplo:

select 5/0;

Los operadores de concatenación: permite concatenar cadenas, el más (+).

Para concatenar el título, el autor y la editorial de cada libro usamos el operador de concatenación ("+"):

select titulo+'-'+autor+'-'+editorial

from libros;

Note que concatenamos además unos guiones para separar los campos.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

Ingresamos algunos registros:

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

Queremos saber el monto total en dinero de cada libro:

select titulo, precio,cantidad,

precio\*cantidad

from libros;

Queremos saber el precio de cada libro con un 10% de descuento:

select titulo,precio,

precio-(precio\*0.1)

from libros;

Actualizamos los precios con un 10% de descuento y vemos el resultado:

update libros set precio=precio-(precio\*0.1);

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

Queremos una columna con el título, el autor y la editorial de cada libro:

select titulo+'-'+autor+'-'+editorial

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(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

select titulo, precio,cantidad,

precio\*cantidad

from libros;

select titulo,precio,

precio-(precio\*0.1)

from libros;

update libros set precio=precio-(precio\*0.1);

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

select titulo+'-'+autor+'-'+editorial

from libros;

**Problema:**

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

ese nombre.

1- Elimine la tabla si existe:

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

drop table articulos;

2- Cree la tabla:

create table articulos(

codigo int identity,

nombre varchar(20),

descripcion varchar(30),

precio smallmoney,

cantidad tinyint default 0,

primary key (codigo)

);

3- Ingrese algunos registros:

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

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

insert into articulos (nombre, descripcion, precio)

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

insert into articulos (nombre, descripcion, precio)

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

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

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

4- El comercio quiere aumentar los precios de todos sus artículos en un 15%. Actualice todos los precios empleando operadores aritméticos.

5- Vea el resultado:

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

6- Muestre todos los artículos, concatenando el nombre y la descripción de cada uno de ellos separados por coma.

7- Reste a la cantidad de todos los teclados, el valor 5, empleando el operador aritmético menos ("-")

**Solución:**

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

drop table articulos;

create table articulos(

codigo int identity,

nombre varchar(20),

descripcion varchar(30),

precio smallmoney,

cantidad tinyint default 0,

primary key (codigo)

);

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

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

insert into articulos (nombre, descripcion, precio)

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

insert into articulos (nombre, descripcion, precio)

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

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

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

update articulos set precio=precio+(precio\*0.15);

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

select nombre+','+descripcion

from articulos;

update articulos set cantidad=cantidad-5

where nombre='teclado';

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

# Alias

Una manera de hacer más comprensible el resultado de una consulta consiste en cambiar los encabezados de las columnas.

Por ejemplo, tenemos la tabla "agenda" con un campo "nombre" (entre otros) en el cual se almacena el nombre y apellido de nuestros amigos; queremos que al mostrar la información de dicha tabla aparezca como encabezado del campo "nombre" el texto "nombre y apellido", para ello colocamos un alias de la siguiente manera:

select nombre as NombreYApellido,

domicilio,telefono

from agenda;

Para reemplazar el nombre de un campo por otro, se coloca la palabra clave "as" seguido del texto del encabezado.

Si el alias consta de una sola cadena las comillas no son necesarias, pero si contiene más de una palabra, es necesario colocarla entre comillas simples:

select nombre as 'Nombre y apellido',

domicilio,telefono

from agenda;

Un alias puede contener hasta 128 caracteres.

También se puede crear un alias para columnas calculadas.

La palabra clave "as" es opcional en algunos casos, pero es conveniente usarla.

Entonces, un "alias" se usa como nombre de un campo o de una expresión. En estos casos, son opcionales, sirven para hacer más comprensible el resultado; en otros casos, que veremos más adelante, son obligatorios.

**Ejercicio:**

Trabajamos con nuestra tabla "agenda".

Eliminamos la tabla si existe:

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

drop table agenda;

Creamos la tabla:

create table agenda(

nombre varchar(30),

domicilio varchar(30),

telefono varchar(11)

);

Ingresamos algunos registros:

insert into agenda

values('Juan Perez','Avellaneda 908','4252525');

insert into agenda

values('Marta Lopez','Sucre 34','4556688');

insert into agenda

values('Carlos Garcia','Sarmiento 1258',null);

Mostramos la información con el encabezado "NombreYApellido" para el campo "nombre":

select nombre as NombreYApellido,

domicilio,telefono

from agenda;

Mostramos la información con el encabezado "Nombre y apellido" para el campo "nombre", necesitamos emplear comillas:

select nombre as 'Nombre y apellido',

domicilio,telefono

from agenda;

La palabra clave "as" es opcional, podemos obviarla:

select nombre 'Nombre y apellido',

domicilio,telefono

from agenda;

**Código:**

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

drop table agenda;

create table agenda(

nombre varchar(30),

domicilio varchar(30),

telefono varchar(11)

);

insert into agenda

values('Juan Perez','Avellaneda 908','4252525');

insert into agenda

values('Marta Lopez','Sucre 34','4556688');

insert into agenda

values('Carlos Garcia','Sarmiento 1258',null);

select nombre as NombreYApellido,

domicilio,telefono

from agenda;

select nombre as 'Nombre y apellido',

domicilio,telefono

from agenda;

select nombre 'Nombre y apellido',

domicilio,telefono

from agenda;

**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- Cree la tabla:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

3- Ingrese algunos registros:

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

4- Muestre todos los campos de los libros y un campo extra, con el encabezado "monto total" en la que calcule el monto total en dinero de cada libro (precio por cantidad)

5- Muestre el título, autor y precio de todos los libros de editorial "Emece" y agregue dos columnas extra en las cuales muestre el descuento de cada libro, con el encabezado "descuento" y el precio con un 10% de descuento con el encabezado "precio final".

6- Muestre una columna con el título y el autor concatenados con el encabezado "Título y autor"

**Solución:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

select titulo, autor,editorial,precio,cantidad,

precio\*cantidad as 'monto total'

from libros;

select titulo,autor,precio,

precio\*0.1 as descuento,

precio-(precio\*0.1) as 'precio final'

from libros

where editorial='Emece';

select titulo+'-'+autor as "Título y autor"

from libros;

# Funciones

Una función es un conjunto de sentencias que operan como una unidad lógica.

Una función tiene un nombre, retorna un parámetro de salida y opcionalmente acepta parámetros de entrada. Las funciones de SQL Server no pueden ser modificadas, las funciones definidas por el usuario sí.

SQL Server ofrece varios tipos de funciones para realizar distintas operaciones. Se pueden clasificar de la siguiente manera:

de agregado: realizan operaciones que combinan varios valores y retornan un único valor. Son "count", "sum", "min" y "max".

escalares: toman un solo valor y retornan un único valor. Pueden agruparse de la siguiente manera:

- de configuración: retornan información referida a la configuración.

Ejemplo:

select @@version;

Retorna la fecha, versión y tipo de procesador de SQL Server.

- de cursores: retornan información sobre el estado de un cursor.

- de fecha y hora: operan con valores "datetime" y "smalldatetime". Reciben un parámetro de tipo fecha y hora y retornan un valor de cadena, numérico o de fecha y hora.

- matemáticas: realizan operaciones numéricas, geométricas y trigonométricas.

- de metadatos: informan sobre las bases de datos y los objetos.

- de seguridad: devuelven información referente a usuarios y funciones.

- de cadena: operan con valores "char", "varchar", "nchar", "nvarchar", "binary" y "varbinary" y devuelven un valor de cadena o numérico.

- del sistema: informan sobre opciones, objetos y configuraciones del sistema. Ejemplo:

select user\_name();

- estadísticas del sistema: retornan información referente al rendimiento del sistema.

- texto e imagen: realizan operaciones con valor de entrada de tipo text o image y retornan información referente al mismo.

de conjuntos de filas: retornan conjuntos de registros.

Se pueden emplear las funciones del sistema en cualquier lugar en el que se permita una expresión en una sentencia "select".

Estudiaremos algunas de ellas.

# Funciones para el manejo de cadenas

Microsoft SQL Server tiene algunas funciones para trabajar con cadenas de caracteres. Estas son algunas:

- substring(cadena,inicio,longitud): devuelve una parte de la cadena especificada como primer argumento, empezando desde la posición especificada por el segundo argumento y de tantos caracteres de longitud como indica el tercer argumento. Ejemplo:

select substring('Buenas tardes',8,6);

retorna "tardes".

- str(numero,longitud,cantidaddecimales): convierte números a caracteres; el primer parámetro indica el valor numérico a convertir, el segundo la longitud del resultado (debe ser mayor o igual a la parte entera del número más el signo si lo tuviese) y el tercero, la cantidad de decimales. El segundo y tercer argumento son opcionales y deben ser positivos. String significa cadena en inglés.

Ejemplo: se convierte el valor numérico "123.456" a cadena, especificando 7 de longitud y 3 decimales:

select str(123.456,7,3);

select str(-123.456,7,3);

retorna '-123.46';

Si no se colocan el segundo y tercer argumeno, la longitud predeterminada es 10 y la cantidad de decimales 0 y se redondea a entero. Ejemplo: se convierte el valor numérico "123.456" a cadena:

select str(123.456);

retorna '123';

select str(123.456,3);

retorna '123';

Si el segundo parámetro es menor a la parte entera del número, devuelve asteriscos (\*). Ejemplo: select str(123.456,2,3);

retorna "\*\*".

- stuff(cadena1,inicio,cantidad,cadena2): inserta la cadena enviada como cuarto argumento, en la posición indicada en el segundo argumento, reemplazando la cantidad de caracteres indicada por el tercer argumento en la cadena que es primer parámetro. Stuff significa rellenar en inglés. Ejemplo:

select stuff('abcde',3,2,'opqrs');

retorna "abopqrse". Es decir, coloca en la posición 2 la cadena "opqrs" y reemplaza 2 caracteres de la primera cadena.

Los argumentos numéricos deben ser positivos y menor o igual a la longitud de la primera cadena, caso contrario, retorna "null".

Si el tercer argumento es mayor que la primera cadena, se elimina hasta el primer carácter.

- len(cadena): retorna la longitud de la cadena enviada como argumento. "len" viene de length, que significa longitud en inglés. Ejemplo:

select len('Hola');

devuelve 4.

- char(x): retorna un caracter en código ASCII del entero enviado como argumento. Ejemplo:

select char(65);

retorna "A".

- left(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la izquierda, primer caracter. Ejemplo:

select left('buenos dias',8);

retorna "buenos d".

- right(cadena,longitud): retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la derecha, último caracter. Ejemplo:

select right('buenos dias',8);

retorna "nos dias".

-lower(cadena): retornan la cadena con todos los caracteres en minúsculas. lower significa reducir en inglés. Ejemplo:

select lower('HOLA ESTUDIAnte');

retorna "hola estudiante".

-upper(cadena): retornan la cadena con todos los caracteres en mayúsculas. Ejemplo:

select upper('HOLA ESTUDIAnte');

-ltrim(cadena): retorna la cadena con los espacios de la izquierda eliminados. Trim significa recortar. Ejemplo:

select ltrim(' Hola ');

retorna "Hola ".

- rtrim(cadena): retorna la cadena con los espacios de la derecha eliminados. Ejemplo:

select rtrim(' Hola ');

retorna " Hola".

- replace(cadena,cadenareemplazo,cadenareemplazar): retorna la cadena con todas las ocurrencias de la subcadena reemplazo por la subcadena a reemplazar. Ejemplo:

select replace('xxx.elmundo.es,'x','w');

retorna "www.elmundo.es'.

- reverse(cadena): devuelve la cadena invirtiendo el order de los caracteres. Ejemplo:

select reverse('Hola');

retorna "aloH".

- patindex(patron,cadena): devuelve la posición de comienzo (de la primera ocurrencia) del patrón especificado en la cadena enviada como segundo argumento. Si no la encuentra retorna 0. Ejemplos:

select patindex('%Luis%', 'Jorge Luis Borges');

retorna 7.

select patindex('%or%', 'Jorge Luis Borges');

retorna 2.

select patindex('%ar%', 'Jorge Luis Borges');

retorna 0.

- charindex(subcadena,cadena,inicio): devuelve la posición donde comienza la subcadena en la cadena, comenzando la búsqueda desde la posición indicada por "inicio". Si el tercer argumento no se coloca, la búsqueda se inicia desde 0. Si no la encuentra, retorna 0. Ejemplos:

select charindex('or','Jorge Luis Borges',5);

retorna 13.

select charindex('or','Jorge Luis Borges');

retorna 2.

select charindex('or','Jorge Luis Borges',14);

retorna 0.

select charindex('or', 'Jorge Luis Borges');

retorna 0.

- replicate(cadena,cantidad): repite una cadena la cantidad de veces especificada. Ejemplo:

select replicate ('Hola',3);

retorna "HolaHolaHola";

- space(cantidad): retorna una cadena de espacios de longitud indicada por "cantidad", que debe ser un valor positivo. Ejemplo:

select 'Hola'+space(1)+'que tal';

retorna "Hola que tal".

Se pueden emplear estas funciones enviando como argumento el nombre de un campo de tipo caracter.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

Ingresamos algunos registros:

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

Mostramos sólo los 12 primeros caracteres de los títulos de los libros y sus autores, empleando la función "substring()":

select substring(titulo,1,12) as titulo

from libros;

Mostramos sólo los 12 primeros caracteres de los títulos de los libros y sus autores, ahora empleando la función "left()":

select left(titulo,12) as titulo

from libros;

Mostramos los títulos de los libros y sus precios convirtiendo este último a cadena de caracteres con un solo decimal, empleando la función "str":

select titulo,

str(precio,6,1)

from libros;

Mostramos los títulos de los libros y sus precios convirtiendo este último a cadena de caracteres especificando un solo argumento:

select titulo,

str(precio)

from libros;

Se redondea a entero.

Mostramos los títulos, autores y editoriales de todos libros, al último campo lo queremos en mayúsculas:

select titulo, autor, upper(editorial)

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(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

select substring(titulo,1,12) as titulo

from libros;

select left(titulo,12) as titulo

from libros;

select titulo,

str(precio,6,1)

from libros;

select titulo,

str(precio)

from libros;

select titulo, autor, upper(editorial)

from libros;

**Problema:**

1.- Crear la tabla y cargarla con datos:

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

2.- Mostrar las 3 primeras letras de todos los títulos.

3.- Mostrar el precio como cadema de caracteres.

4.- Mostrar la cadena con el titulo, un guión, el autor un guión y el precio.

5.- Mostrar las seis últimas letras del titulo y del autor.

6.- Mostrar el nombre del autor en mayúscula.

7.- indicar el número de letras del autor y del título.

8.- mostrar los caracteres del 4 al 10 del autor

9.- cambiar arroba por el caracter arroba y punto por el caracter punto en el texto correoarrobahormailpuntocom

**Solución:**

1.- Crear la tabla y cargarla con datos:

use pruebasql;

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

cantidad tinyint default 0,

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.40,100);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',15,50);

2.- Mostrar las 3 primeras letras de todos los títulos.

select LEFT(titulo,3)

from libros;

3.- Mostrar el precio como cadema de caracteres.

select str(precio)

from libros;

4.- Mostrar la cadena con el titulo, un guión, el autor un guión y el precio.

select titulo+'-'+autor+'-'+str(precio)

from libros

5.- Mostrar las seis últimas letras del titulo y del autor.

select RIGHT(titulo,6),right(autor,6)

from libros

6.- Mostrar el nombre del autor en mayúscula.

select upper(autor)

from libros

7.- indicar el número de letras del autor y del título.

select LEN(titulo),LEN(autor)

from libros

8.- mostrar los caracteres del 4 al 10 del autor

select SUBSTRING(autor,4,7)

from libros

9.- cambiar arroba por el caracter arroba y punto por el caracter punto en el texto correoarrobahormailpuntocom

select replace(REPLACE('correoarrobahormailpuntocom','arroba','@'),'punto','.')

# Funciones matemáticas

Las funciones matemáticas realizan operaciones con expresiones numéricas y retornan un resultado, operan con tipos de datos numéricos.

Microsoft SQL Server tiene algunas funciones para trabajar con números. Aquí presentamos algunas.

-abs(x): retorna el valor absoluto del argumento "x". Ejemplo:

select abs(-20);

retorna 20.

-ceiling(x): redondea hacia arriba el argumento "x". Ejemplo:

select ceiling(12.34);

retorna 13.

-floor(x): redondea hacia abajo el argumento "x". Ejemplo:

select floor(12.34);

retorna 12.

- %: %: devuelve el resto de una división. Ejemplos:

select 10%3;

retorna 1.

select 10%2;

retorna 0.

-power(x,y): retorna el valor de "x" elevado a la "y" potencia. Ejemplo:

select power(2,3);

retorna 8.

-round(numero,longitud): retorna un número redondeado a la longitud especificada. "longitud" debe ser tinyint, smallint o int. Si "longitud" es positivo, el número de decimales es redondeado según "longitud"; si es negativo, el número es redondeado desde la parte entera según el valor de "longitud".

Si le pasamos un tercer parámetro a 1 lo que hace es truncar, sin tener en cuenta el resto de decimales.

Ejemplos:

select round(123.456,1);

retorna "123.400", es decir, redondea desde el primer decimal.

select round(123.456,2);

retorna "123.460", es decir, redondea desde el segundo decimal.

select round(123.456,-1);

retorna "120.000", es decir, redondea desde el primer valor entero (hacia la izquierda).

select round(123.456,-2);

retorna "100.000", es decir, redondea desde el segundo valor entero (hacia la izquierda).

Select round(123.456,1,1);

Retorna "123.400", truncando el valor a un único decimal.

-sign(x): si el argumento es un valor positivo devuelve 1;-1 si es negativo y si es 0, 0.

-square(x): retorna el cuadrado del argumento. Ejemplo:

select square(3); retorna 9.

-sqrt(x): devuelve la raiz cuadrada del valor enviado como argumento.

SQL Server dispone de funciones trigonométricas que retornan radianes.

Se pueden emplear estas funciones enviando como argumento el nombre de un campo de tipo numérico.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key (codigo)

);

Ingresamos algunos registros:

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

values('El aleph','Borges','Emece',25.33);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.65);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.95);

Vamos a mostrar los precios de los libros redondeando el valor hacia abajo y hacia arriba:

select titulo,autor,precio,

floor(precio) as abajo,

ceiling(precio) as arriba

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(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25.33);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.65);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.95);

select titulo,autor,precio,

floor(precio) as abajo,

ceiling(precio) as arriba

from libros;

**Problema:**

1.- Redondear 4567.345 con 2 decimales

2.- Truncar 4567.356 con 1 decimal

3.- Raíz cuadrada de 625

4.- Resultado y resto de dividir 16 entre 3

5.- Cuadrado de 12

6.- Valor absoluto de la diferencia 23-56

**Solución:**

1.- Redondear 4567.345 con 2 decimales

select round(4567.345,2)

2.- Truncar 4567.356 con 1 decimal

select round(4567.356,1,1)

3.- Raíz cuadrada de 625

select SQRT(625)

4.- Resultado y resto de dividir 16 entre 3

select floor(16/3), 16%3

5.- Cuadrado de 12

select SQUARE(12)

6.- Valor absoluto de la diferencia 23-56

select ABS(23-56)

# Funciones para el uso de fechas y horas

Microsoft SQL Server ofrece algunas funciones para trabajar con fechas y horas y son muy importantes en el manejo de campos de fechahora. Estas son algunas:

- getdate(): retorna la fecha y hora actuales. Ejemplo:

select getdate();

- datepart(partedefecha,fecha): retorna la parte específica de una fecha, el año, trimestre, día, hora, etc.

Los valores para "partedefecha" pueden ser: year (año), quarter (cuarto), month (mes), day (dia), week (semana), hour (hora), minute (minuto), second (segundo), millisecond (milisegundo), dw (día de la semana). Ejemplos:

select datepart(month,getdate());

Retorna el número de mes actual;

select datepart(day,getdate());

Retorna el día actual;

select datepart(hour,getdate());

Retorna la hora actual;

- datename(partedefecha,fecha): retorna el nombre de una parte específica de una fecha. Los valores para "partedefecha" pueden ser los mismos que se explicaron anteriormente. Ejemplos:

select datename(month,getdate());

Retorna el nombre del mes actual;

select datename(dw,getdate());

Retorna el nombre del día de la semana de la fecha actual;

select datename(day,getdate());

- dateadd(partedelafecha,numero,fecha): agrega un intervalo a la fecha especificada, es decir, retorna una fecha adicionando a la fecha enviada como tercer argumento, el intervalo de tiempo indicado por el primer parámetro, tantas veces como lo indica el segundo parámetro. Los valores para el primer argumento pueden ser: year (año), quarter (cuarto), month (mes), day (dia), week (semana), hour (hora), minute (minuto), second (segundo) y millisecond (milisegundo). Ejemplos:

select dateadd(day,3,'1980/11/02');

retorna "1980/11/05", agrega 3 días.

select dateadd(month,3,'1980/11/02');

retorna "1981/02/02", agrega 3 meses.

select dateadd(hour,2,'1980/11/02');

retorna "1980/02/02 2:00:00", agrega 2 horas.

select dateadd(minute,16,'1980/11/02');

retorna "1980/02/02 00:16:00", agrega 16 minutos.

- datediff(partedelafecha,fecha1,fecha2): calcula el intervalo de tiempo (según el primer argumento) entre las 2 fechas. El resultado es un valor entero que corresponde a fecha2-fecha1. Los valores de "partedelafecha) pueden ser los mismos que se especificaron anteriormente. Ejemplos:

select datediff (day,'2005/10/28','2006/10/28');

retorna 365 (días).

select datediff(month,'2005/10/28','2006/11/29');

retorna 13 (meses).

- day(fecha): retorna el día de la fecha especificada. Ejemplo:

select day(getdate());

- month(fecha): retorna el mes de la fecha especificada. Ejemplo:

select month(getdate());

- year(fecha): retorna el año de la fecha especificada. Ejemplo:

select year(getdate());

Se pueden emplear estas funciones enviando como argumento el nombre de un campo de tipo datetime o smalldatetime.

**Ejercicio:**

Trabajamos con la tabla "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) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

edicion datetime,

precio decimal(6,2)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece','1980/10/10',25.33);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI','2000/05/05',50.65);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece','2000/08/09',19.95);

insert into libros

values('Aprenda PHP','Mario Molina','Siglo XXI','2000/02/04',45);

Mostramos el título del libro y el año de edición:

select titulo, datepart (year,edicion) from libros;

Mostramos el título del libro y el nombre del mes de edición:

select titulo, datename (month,edicion) from libros;

Mostramos el título del libro y los años que tienen de editados:

select titulo, datediff(year,edicion,getdate()) from libros;

Muestre los títulos de los libros que se editaron el día 9, de cualquier mes de cualquier año:

select titulo from libros

where datepart(day,edicion)=9;

**Código:**

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

drop table libros;

create table libros(

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

edicion datetime,

precio decimal(6,2)

);

insert into libros values('El aleph','Borges','Emece','1980/10/10',25.33);

insert into libros values('Java en 10 minutos','Mario Molina','Siglo XXI','2000/05/05',50.65);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Emece','2000/08/09',19.95);

insert into libros values('Aprenda PHP','Mario Molina','Siglo XXI','2000/02/04',45);

select titulo, datepart (year,edicion) from libros;

select titulo, datename (month,edicion) from libros;

select titulo, datediff(year,edicion,getdate()) from libros;

select titulo from libros

where datepart(day,edicion)=9;

**Problema:**

Una empresa almacena los datos de sus empleados en una tabla denominada "empleados".

1- Elimine la tabla si existe:

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

drop table empleados;

2- Cree la tabla:

create table empleados(

nombre varchar(30) not null,

apellido varchar(20) not null,

documento char(8),

fechanacimiento datetime,

fechaingreso datetime,

sueldo decimal(6,2),

primary key(documento)

);

3- Ingrese algunos registros, con formato de fecha año, mes, día:

insert into empleados values('Ana','Acosta','22222222','1970/10/10','1995/05/05',228.50);

insert into empleados values('Carlos','Caseres','25555555','1978/02/06','1998/05/05',309);

insert into empleados values('Francisco','Garcia','26666666','1978/10/15','1998/10/02',250.68);

insert into empleados values('Gabriela','Garcia','30000000','1985/10/25','2000/12/22',300.25);

insert into empleados values('Luis','Lopez','31111111','1987/02/10','2000/08/21',350.98);

4- Muestre los nombres y apellidos de los empleados que cumplen años en el mes "octubre" (3 registros)

5- Muestre los nombres y apellidos de los empleados que ingresaron en 1998 (2 registros).

6.- Muestre el mes de nacimiento de los empleados.

7.- Muestre para cada empleado cuándo cumple 65 años.

8.- Muestre la edad de cada empleado en años.

9.- Muestre el número del día de la semana de la fecha actual

**Solución:**

1- Elimine la tabla si existe:

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

drop table empleados;

2- Cree la tabla:

create table empleados(

nombre varchar(30) not null,

apellido varchar(20) not null,

documento char(8),

fechanacimiento datetime,

fechaingreso datetime,

sueldo decimal(6,2),

primary key(documento)

);

3- Ingrese algunos registros:

set dateformat ymd

insert into empleados values('Ana','Acosta','22222222','1970/10/10','1995/05/05',228.50);

insert into empleados values('Carlos','Caseres','25555555','1978/02/06','1998/05/05',309);

insert into empleados values('Francisco','Garcia','26666666','1978/10/15','1998/10/02',250.68);

insert into empleados values('Gabriela','Garcia','30000000','1985/10/25','2000/12/22',300.25);

insert into empleados values('Luis','Lopez','31111111','1987/02/10','2000/08/21',350.98);

4- Muestre los nombres y apellidos de los empleados que cumplen años en el mes "octubre" (3 registros)

select nombre,apellido

from empleados

where datename(month,fechanacimiento)='octubre'

5- Muestre los nombres y apellidos de los empleados que ingresaron en 1998 (2 registros).

select nombre,apellido

from empleados

where datename(year,fechaingreso)='1998'

6.- Muestre el mes de nacimiento de los empleados.

select nombre,apellido,datename(month,fechanacimiento)

from empleados

7.- Muestre para cada empleado cuándo cumple 65 años.

select nombre,apellido,DATEadd(year,65,fechanacimiento)

from empleados

8.- Muestre la edad de cada empleado en años.

select nombre,apellido,DATEDIFF(year,fechanacimiento,GETDATE())

from empleados

9.- Muestre el número del día de la semana de la fecha actual

select DATEPART(DW,GETDATE())

# Formato Fecha en select

CONVERT Función para convertir la salida en el select de un campo datetime.

CONVERT Function Syntax: CONVERT(data\_type,expression,date Format style)

Formatos posibles:

USA mm/dd/yy

select convert(varchar, getdate(), 1)

ANSI yy.mm.dd

select convert(varchar, getdate(), 2)

British/French dd/mm/yy

select convert(varchar, getdate(), 3)

German dd.mm.yy

select convert(varchar, getdate(), 4)

Italian dd-mm-yy

select convert(varchar, getdate(), 5)

dd mon yy

select convert(varchar, getdate(), 6)

Mon dd, yy

select convert(varchar, getdate(), 7)

USA mm-dd-yy

select convert(varchar, getdate(), 10)

JAPAN yy/mm/dd

select convert(varchar, getdate(), 11)

ISO yymmdd

select convert(varchar, getdate(), 12)

mon dd yyyy hh:miAM (or PM)

select convert(varchar, getdate(), 100)

mm/dd/yyyy

select convert(varchar, getdate(), 101)

yyyy.mm.dd

select convert(varchar, getdate(), 102)

dd/mm/yyyy

select convert(varchar, getdate(), 103)

dd.mm.yyyy

select convert(varchar, getdate(), 104)

dd-mm-yyyy

select convert(varchar, getdate(), 105)

dd mon yyyy

select convert(varchar, getdate(), 106)

Mon dd, yyyy

select convert(varchar, getdate(), 107)

hh:mm:ss

select convert(varchar, getdate(), 108)

Default + milliseconds mon dd yyyy hh:miAM (or PM)

select convert(varchar, getdate(), 109)

mm-dd-yyyy

select convert(varchar, getdate(), 110)

yyyy/mm/dd

select convert(varchar, getdate(), 111)

yyyymmdd

select convert(varchar, getdate(), 112)

Europe default + milliseconds dd mon yyyy hh:mm:ss:mmm(24h)

select convert(varchar, getdate(), 113) or select convert(varchar, getdate(), 13)

hh:mi:ss:mmm(24h)

select convert(varchar, getdate(), 114)

**Problema:**

1.- Mostrar la fecha actual en formato dd/mm/yyyy

2.- Mostrar la hora actual.

3.- Escribir la fecha '15/3/2013' y verla en formato ANSI

**Solución:**

1.- Mostrar la fecha actual en formato dd/mm/yyyy

select convert(varchar, getdate(), 103)

2.- Mostrar la hora actual.

select convert(varchar,GETDATE(),108)

3.- Escribir la fecha '15/3/2013' y verla en formato ANSI

set dateformat dmy

select convert(varchar,convert(datetime,'15/3/2013'),2)

# Ordenar registros (order by)

Podemos ordenar el resultado de un "select" para que los registros se muestren ordenados por algún campo, para ello usamos la cláusula "order by".

La sintaxis básica es la siguiente:

select lista de campos from NOMBRETABLA

order by CAMPO;

Por ejemplo, recuperamos los registros de la tabla "libros" ordenados por el título:

select \*from libros

order by titulo;

Aparecen los registros ordenados alfabéticamente por el campo especificado.

También podemos colocar el número de orden del campo por el que queremos que se ordene en lugar de su nombre, es decir, referenciar a los campos por su posición en la lista de selección. Por ejemplo, queremos el resultado del "select" ordenado por "precio":

select titulo,autor,precio

from libros order by 3;

Por defecto, si no aclaramos en la sentencia, los ordena de manera ascendente (de menor a mayor).

Podemos ordenarlos de mayor a menor, para ello agregamos la palabra clave "desc":

select codigo,titulo,autor,editorial,precio libros

order by editorial desc;

También podemos ordenar por varios campos, por ejemplo, por "titulo" y "editorial":

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

order by titulo,editorial;

Incluso, podemos ordenar en distintos sentidos, por ejemplo, por "titulo" en sentido ascendente y "editorial" en sentido descendente:

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

order by titulo asc, editorial desc;

Debe aclararse al lado de cada campo, pues estas palabras claves afectan al campo inmediatamente anterior.

Es posible ordenar por un campo que no se lista en la selección.

Se permite ordenar por valores calculados o expresiones.

La cláusula "order by" no puede emplearse para campos text, ntext e image.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key (codigo)

);

Ingresamos algunos registros:

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

values('El aleph','Borges','Emece',25.33);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.65);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.95);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',15);

Recuperamos los registros ordenados por el título:

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

order by titulo;

Ordenamos los registros por el campo "precio", referenciando el campo por su posición en la lista de selección:

select titulo,autor,precio

from libros order by 3;

Los ordenamos por "editorial", de mayor a menor empleando "desc":

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

order by editorial desc;

Ordenamos por dos campos:

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

order by titulo,editorial;

Ordenamos en distintos sentidos:

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

order by titulo asc, editorial desc;

Podemos ordenar por un campo que no se lista en la selección:

select titulo, autor

from libros

order by precio;

Está permitido ordenar por valores calculados:

select titulo, autor, editorial,

precio+(precio\*0.1) as 'precio con descuento'

from libros

order by 4;

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key (codigo)

);

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

values('El aleph','Borges','Emece',25.33);

insert into libros

values('Java en 10 minutos','Mario Molina','Siglo XXI',50.65);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.95);

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

values('Alicia en el pais de las maravillas','Lewis Carroll','Planeta',15);

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

order by titulo;

select titulo,autor,precio

from libros order by 3;

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

order by editorial desc;

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

order by titulo,editorial;

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

order by titulo asc, editorial desc;

select titulo, autor

from libros

order by precio;

select titulo, autor, editorial,

precio+(precio\*0.1) as 'precio con descuento'

from libros

order by 4;

**Problema:**

En una página web se guardan los siguientes datos de las visitas: número de visita, nombre, mail,

pais, fecha.

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

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

drop table visitas;

2- Créela con la siguiente estructura:

create table visitas (

numero int identity,

nombre varchar(30) default 'Anonimo',

mail varchar(50),

pais varchar (20),

fecha datetime,

primary key(numero)

);

3- Ingrese algunos registros:

insert into visitas (nombre,mail,pais,fecha)

values ('Ana Maria Lopez','AnaMaria@hotmail.com','Argentina','2006-10-10 10:10');

insert into visitas (nombre,mail,pais,fecha)

values ('Gustavo Gonzalez','GustavoGGonzalez@hotmail.com','Chile','2006-10-10 21:30');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-10-11 15:45');

insert into visitas (nombre,mail,pais,fecha)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-10-12 08:15');

insert into visitas (nombre,mail,pais,fecha)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-09-12 20:45');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-12 16:20');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-15 16:25');

4- Ordene los registros por fecha, en orden descendente.

5- Muestre el nombre del usuario, país y el nombre del mes, ordenado por país (ascendente) y nombre del mes (descendente)

6- Muestre el país, el mes, el día y la hora y ordene las visitas por nombre del mes, del día y la hora.

7- Muestre los mail, país, ordenado por país, de todos los que visitaron la página en octubre (4 registros)

**Solución:**

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

drop table visitas;

create table visitas (

numero int identity,

nombre varchar(30) default 'Anonimo',

mail varchar(50),

pais varchar (20),

fecha datetime,

primary key(numero)

);

insert into visitas (nombre,mail,pais,fecha)

values ('Ana Maria Lopez','AnaMaria@hotmail.com','Argentina','2006-10-10 10:10');

insert into visitas (nombre,mail,pais,fecha)

values ('Gustavo Gonzalez','GustavoGGonzalez@hotmail.com','Chile','2006-10-10 21:30');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-10-11 15:45');

insert into visitas (nombre,mail,pais,fecha)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-10-12 08:15');

insert into visitas (nombre,mail,pais,fecha)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-09-12 20:45');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-12 16:20');

insert into visitas (nombre,mail,pais,fecha)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-15 16:25');

select numero,nombre,mail,pais,fecha from visitas

order by fecha desc;

select nombre,pais,datename(month,fecha)

from visitas

order by pais,datename(month,fecha) desc;

select nombre,mail,

datename(month,fecha) mes,

datename(day,fecha) dia,

datename(hour,fecha) hora

from visitas

order by 3,4,5;

select mail, pais

from visitas

where datename(month,fecha)='October'

order by 2;

# Operadores lógicos ( and - or - not)

Hasta el momento, hemos aprendido a establecer una condición con "where" utilizando operadores relacionales. Podemos establecer más de una condición con la cláusula "where", para ello aprenderemos los operadores lógicos.

Son los siguientes:

- and, significa "y",

- or, significa "y/o",

- not, significa "no", invierte el resultado

- (), paréntesis

Los operadores lógicos se usan para combinar condiciones.

Si queremos recuperar todos los libros cuyo autor sea igual a "Borges" y cuyo precio no supere los 20 pesos, necesitamos 2 condiciones:

select \*from libros

where (autor='Borges') and

(precio<=20);

Los registros recuperados en una sentencia que une 2 condiciones con el operador "and", cumplen con las 2 condiciones.

Queremos ver los libros cuyo autor sea "Borges" y/o cuya editorial sea "Planeta":

select \*from libros

where autor='Borges' or

editorial='Planeta';

En la sentencia anterior usamos el operador "or"; indicamos que recupere los libros en los cuales el valor del campo "autor" sea "Borges" y/o el valor del campo "editorial" sea "Planeta", es decir, seleccionará los registros que cumplan con la primera condición, con la segunda condición o con ambas condiciones.

Los registros recuperados con una sentencia que une 2 condiciones con el operador "or", cumplen 1 de las condiciones o ambas.

Queremos recuperar los libros que NO cumplan la condición dada, por ejemplo, aquellos cuya editorial NO sea "Planeta":

select \*from libros

where not editorial='Planeta';

El operador "not" invierte el resultado de la condición a la cual antecede.

Los registros recuperados en una sentencia en la cual aparece el operador "not", no cumplen con la condición a la cual afecta el "NOT".

Los paréntesis se usan para encerrar condiciones, para que se evalúen como una sola expresión.

Cuando explicitamos varias condiciones con diferentes operadores lógicos (combinamos "and", "or") permite establecer el orden de prioridad de la evaluación; además permite diferenciar las expresiones más claramente.

Por ejemplo, las siguientes expresiones devuelven un resultado diferente:

select\*from libros

where (autor='Borges') or

(editorial='Paidos' and precio<20);

select \*from libros

where (autor='Borges' or editorial='Paidos') and

(precio<20);

Si bien los paréntesis no son obligatorios en todos los casos, se recomienda utilizarlos para evitar confusiones.

El orden de prioridad de los operadores lógicos es el siguiente: "not" se aplica antes que "and" y "and" antes que "or", si no se especifica un orden de evaluación mediante el uso de paréntesis.

El orden en el que se evalúan los operadores con igual nivel de precedencia es indefinido, por ello se recomienda usar los paréntesis.

Entonces, para establecer más de una condición en un "where" es necesario emplear operadores lógicos. "and" significa "y", indica que se cumplan ambas condiciones; "or" significa "y/o", indica que se cumpla una u otra condición (o ambas); "not" significa "no", indica que no se cumpla la condición especificada.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','Borges','Planeta',39.50);

insert into libros

values('Java en 10 minutos','Mario Molina','Planeta',50.50);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Paidos',16.80);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',19.50);

insert into libros

values('Cervantes y el quijote','Borges','Paidos',18.40);

Recuperamos los libros cuyo autor sea igual a "Borges" y cuyo precio no supere los 20 pesos:

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

where (autor='Borges') and

(precio<=20);

Seleccionamos los libros cuyo autor es "Borges" y/o cuya editorial es "Planeta":

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

where autor='Borges' or

editorial='Planeta';

Recuperamos los libros cuya editorial NO es "Planeta":

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

where not editorial='Planeta';

Veamos cómo el uso de paréntesis hace que SQL Server evalúe en forma diferente ciertas consultas aparentemente iguales:

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

where (autor='Borges') or

(editorial='Paidos' and precio<20);

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

where (autor='Borges' or editorial='Paidos') and

(precio<20);

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','Borges','Planeta',39.50);

insert into libros

values('Java en 10 minutos','Mario Molina','Planeta',50.50);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Paidos',16.80);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',19.50);

insert into libros

values('Cervantes y el quijote','Borges','Paidos',18.40);

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

where (autor='Borges') and

(precio<=20);

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

where autor='Borges' or

editorial='Planeta';

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

where not editorial='Planeta';

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

where (autor='Borges') or

(editorial='Paidos' and precio<20);

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

where (autor='Borges' or editorial='Paidos') and

(precio<20);

**Problema:**

Trabaje con la tabla llamada "medicamentos" de una farmacia.

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 int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

3- Ingrese algunos registros:

insert into medicamentos

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

insert into medicamentos

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

insert into medicamentos

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

insert into medicamentos

values('Paracetamol 500','Bago',1.90,200);

insert into medicamentos

values('Bayaspirina','Bayer',2.10,150);

insert into medicamentos

values('Amoxidal jarabe','Bayer',5.10,250);

4- Recupere los códigos y nombres de los medicamentos cuyo laboratorio sea 'Roche' y cuyo precio sea menor a 5 (1 registro cumple con ambas condiciones)

5- Recupere los medicamentos cuyo laboratorio sea 'Roche' o cuyo precio sea menor a 5 (4 registros).

Note que el resultado es diferente al del punto 4, hemos cambiado el operador de la sentencia anterior.

6- Muestre todos los medicamentos cuyo laboratorio NO sea "Bayer" y cuya cantidad sea=100 (1 registro)

7- Muestre todos los medicamentos cuyo laboratorio sea "Bayer" y cuya cantidad NO sea=100 (2 registros):

Analice estas 2 últimas sentencias. El operador "not" afecta a la condición a la cual antecede, no a las siguientes. Los resultados de los puntos 6 y 7 son diferentes.

8- Elimine todos los registros cuyo laboratorio sea igual a "Bayer" y su precio sea mayor a 10 (1 registro eliminado)

9- Cambie la cantidad por 200, a todos los medicamentos de "Roche" cuyo precio sea mayor a 5 (1 registro afectado)

10- Borre los medicamentos cuyo laboratorio sea "Bayer" o cuyo precio sea menor a 3 (3 registros borrados)

**Solución:**

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

drop table medicamentos;

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

insert into medicamentos

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

insert into medicamentos

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

insert into medicamentos

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

insert into medicamentos

values('Paracetamol 500','Bago',1.90,200);

insert into medicamentos

values('Bayaspirina','Bayer',2.10,150);

insert into medicamentos

values('Amoxidal jarabe','Bayer',5.10,250);

select codigo,nombre

from medicamentos

where laboratorio='Roche' and

precio<5;

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

where laboratorio='Roche' or

precio<5;

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

where not laboratorio='Bayer' and

cantidad=100;

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

where laboratorio='Bayer' and

not cantidad=100;

delete from medicamentos

where laboratorio='Bayer' and

precio>10;

update medicamentos set cantidad=200

where laboratorio='Roche' and

precio>5;

delete from medicamentos

where laboratorio='Bayer' or

precio<3;

# Otros operadores relacionales (is null)

Existe otro operador relacional "is null".

Se emplea el operador "is null" para recuperar los registros en los cuales esté almacenado el valor "null" en un campo específico:

select \*from libros

where editorial is null;

Para obtener los registros que no contiene "null", se puede emplear "is not null", esto mostrará los registros con valores conocidos.

Siempre que sea posible, emplee condiciones de búsqueda positivas ("is null"), evite las negativas ("is not null") porque con ellas se evalúan todos los registros y esto hace más lenta la recuperación de los datos.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros (titulo,autor,precio)

values('Antología poética','Borges',25.50);

insert into libros (titulo,autor,precio)

values('Java en 10 minutos','Mario Molina',45.80);

insert into libros (titulo,autor)

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

insert into libros (titulo,autor)

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

Recuperamos los registros en los cuales esté almacenado el valor "null" en el campo "editorial":

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

where editorial is null;

Seleccionamos los libros que no contiene "null" en "editorial":

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

where editorial is not null;

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros (titulo,autor,precio)

values('Antología poética','Borges',25.50);

insert into libros (titulo,autor,precio)

values('Java en 10 minutos','Mario Molina',45.80);

insert into libros (titulo,autor)

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

insert into libros (titulo,autor)

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

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

where editorial is null;

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

where editorial is not null;

**Problema:**

Trabajamos con la tabla "peliculas" de un video club que alquila películas en video.

1- Elimine la tabla, si existe;

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

drop table peliculas;

2- Créela con la siguiente estructura:

create table peliculas(

codigo int identity,

titulo varchar(40) not null,

actor varchar(20),

duracion tinyint,

primary key (codigo)

);

3- Ingrese algunos registros:

insert into peliculas

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

insert into peliculas

values('Harry Potter y la piedra filosofal','Daniel R.',null);

insert into peliculas

values('Harry Potter y la camara secreta','Daniel R.',190);

insert into peliculas

values('Mision imposible 2','Tom Cruise',120);

insert into peliculas

values('Mujer bonita',null,120);

insert into peliculas

values('Tootsie','D. Hoffman',90);

insert into peliculas (titulo)

values('Un oso rojo');

4- Recupere las películas cuyo actor sea nulo (2 registros)

5- Cambie la duración a 0, de las películas que tengan duración igual a "null" (2 registros)

6- Borre todas las películas donde el actor sea "null" y cuya duración sea 0 (1 registro)

**Solución:**

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

drop table peliculas;

create table peliculas(

codigo int identity,

titulo varchar(40) not null,

actor varchar(20),

duracion tinyint,

primary key (codigo)

);

insert into peliculas

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

insert into peliculas

values('Harry Potter y la piedra filosofal','Daniel R.',null);

insert into peliculas

values('Harry Potter y la camara secreta','Daniel R.',190);

insert into peliculas

values('Mision imposible 2','Tom Cruise',120);

insert into peliculas

values('Mujer bonita',null,120);

insert into peliculas

values('Tootsie','D. Hoffman',90);

insert into peliculas (titulo)

values('Un oso rojo');

select codigo,titulo,actor,duracion from peliculas

where actor is null;

update peliculas set duracion=0

where duracion is null;

delete from peliculas

where actor is null and

duracion=0;

select codigo,titulo,actor,duracion from peliculas;

# Otros operadores relacionales (between)

Hemos visto los operadores relacionales: = (igual), <> (distinto), > (mayor), < (menor), >= (mayor o igual), <= (menor o igual), is null/is not null (si un valor es NULL o no).

Otro operador relacional es "between", trabajan con intervalos de valores.

Hasta ahora, para recuperar de la tabla "libros" los libros con precio mayor o igual a 20 y menor o igual a 40, usamos 2 condiciones unidas por el operador lógico "and":

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

where precio>=20 and

precio<=40;

Podemos usar "between" y así simplificar la consulta:

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

where precio between 20 and 40;

Averiguamos si el valor de un campo dado (precio) está entre los valores mínimo y máximo especificados (20 y 40 respectivamente). Mayor o igual que el primero y menor o igual que el segundo.

"between" significa "entre". Trabaja con intervalo de valores.

Este operador se puede emplear con tipos de datos numéricos y money (en tales casos incluyen los valores mínimo y máximo) y tipos de datos fecha y hora (incluye sólo el valor mínimo).

No tiene en cuenta los valores "null".

Si agregamos el operador "not" antes de "between" el resultado se invierte, es decir, se recuperan los registros que están fuera del intervalo especificado. Por ejemplo, recuperamos los libros cuyo precio NO se encuentre entre 20 y 35, es decir, los menores a 15 y mayores a 25:

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

where precio not between 20 and 35;

Siempre que sea posible, emplee condiciones de búsqueda positivas ("between"), evite las negativas ("not between") porque hace más lenta la recuperación de los datos.

Entonces, se puede usar el operador "between" para reducir las condiciones "where".

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros (titulo,autor,precio)

values('Antología poética','Borges',32);

insert into libros (titulo,autor,precio)

values('Java en 10 minutos','Mario Molina',45.80);

insert into libros (titulo,autor,precio)

values('Martin Fierro','Jose Hernandez',40);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',56.50);

Recuperamos los registros cuyo precio esté entre 20 y 40 empleando "between":

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

where precio between 20 and 40;

Note que si el campo tiene el valor "null", no aparece en la selección.

Para seleccionar los libros cuyo precio NO esté entre un intervalo de valores antecedemos "not" al "between":

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

where precio not between 20 and 35;

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',25.90);

insert into libros (titulo,autor,precio)

values('Antología poética','Borges',32);

insert into libros (titulo,autor,precio)

values('Java en 10 minutos','Mario Molina',45.80);

insert into libros (titulo,autor,precio)

values('Martin Fierro','Jose Hernandez',40);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',56.50);

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

where precio between 20 and 40;

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

where precio not between 20 and 35;

**Problema:**

En una página web se guardan los siguientes datos de las visitas: número de visita, nombre, mail, pais, fechayhora de la visita.

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

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

drop table visitas;

2- Créela con la siguiente estructura:

create table visitas (

numero int identity,

nombre varchar(30) default 'Anonimo',

mail varchar(50),

pais varchar (20),

fechayhora datetime,

primary key(numero)

);

3- Ingrese algunos registros:

insert into visitas (nombre,mail,pais,fechayhora)

values ('Ana Maria Lopez','AnaMaria@hotmail.com','Argentina','2006-10-10 10:10');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Gustavo Gonzalez','GustavoGGonzalez@gotmail.com','Chile','2006-10-10 21:30');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-10-11 15:45');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-10-12 08:15');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-09-12 20:45');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@gmail.com','Argentina','2006-09-12 16:20');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-15 16:25');

insert into visitas (nombre,mail,pais)

values ('Federico1','federicogarcia@xaxamail.com','Argentina');

4- Seleccione los usuarios que visitaron la página entre el '2006-09-12' y '2006-10-11' (5 registros)

Note que incluye los de fecha mayor o igual al valor mínimo y menores al valor máximo, y que los valores null no se incluyen.

5- Recupere las visitas cuyo número se encuentra entre 2 y 5 (4 registros)

Note que incluye los valores límites.

**Solución:**

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

drop table visitas;

create table visitas (

numero int identity,

nombre varchar(30) default 'Anonimo',

mail varchar(50),

pais varchar (20),

fechayhora datetime,

primary key(numero)

);

insert into visitas (nombre,mail,pais,fechayhora)

values ('Ana Maria Lopez','AnaMaria@hotmail.com','Argentina','2006-10-10 10:10');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Gustavo Gonzalez','GustavoGGonzalez@gotmail.com','Chile','2006-10-10 21:30');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-10-11 15:45');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-10-12 08:15');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Fabiola Martinez','MartinezFabiola@hotmail.com','Mexico','2006-09-12 20:45');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@gmail.com','Argentina','2006-09-12 16:20');

insert into visitas (nombre,mail,pais,fechayhora)

values ('Juancito','JuanJosePerez@hotmail.com','Argentina','2006-09-15 16:25');

insert into visitas (nombre,mail,pais)

values ('Federico1','federicogarcia@xaxamail.com','Argentina');

select numero, nombre, mail, pais, fechayhora

from visitas

where fechayhora between '2006-09-12' and '2006-10-11';

select numero, nombre, mail, pais, fechayhora

from visitas

where numero between 2 and 5;

# Otros operadores relacionales (in)

Se utiliza "in" para averiguar si el valor de un campo está incluido en una lista de valores especificada.

En la siguiente sentencia usamos "in" para averiguar si el valor del campo autor está incluido en la lista de valores especificada (en este caso, 2 cadenas).

Hasta ahora, para recuperar los libros cuyo autor sea 'Paenza' o 'Borges' usábamos 2 condiciones:

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

where autor='Borges' or autor='Paenza';

Podemos usar "in" y simplificar la consulta:

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

where autor in('Borges','Paenza');

Para recuperar los libros cuyo autor no sea 'Paenza' ni 'Borges' usábamos:

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

where autor<>'Borges' and

autor<>'Paenza';

También podemos usar "in" anteponiendo "not":

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

where autor not in ('Borges','Paenza');

Empleando "in" averiguamos si el valor del campo está incluido en la lista de valores especificada; con "not" antecediendo la condición, invertimos el resultado, es decir, recuperamos los valores que no se encuentran (coindicen) con la lista de valores.

Los valores "null" no se consideran.

Recuerde: siempre que sea posible, emplee condiciones de búsqueda positivas ("in"), evite las negativas ("not in") porque con ellas se evalún todos los registros y esto hace más lenta la recuperación de los datos.

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20),

editorial varchar(20),

precio decimal(6,2),

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros (titulo,precio)

values('Antología poética',32);

insert into libros (titulo,autor,precio)

values('Martin Fierro','Jose Hernandez',40);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',56.50);

Recuperamos los libros cuyo autor es "Paenza" o "Borges":

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

where autor in('Borges','Paenza');

Recuperamos los libros cuyo autor NO es "Paenza" ni "Borges":

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

where autor not in ('Borges','Paenza');

Note que los valores "null" no se consideran.

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20),

editorial varchar(20),

precio decimal(6,2),

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

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

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros (titulo,precio)

values('Antología poética',32);

insert into libros (titulo,autor,precio)

values('Martin Fierro','Jose Hernandez',40);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',56.50);

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

where autor in('Borges','Paenza');

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

where autor not in ('Borges','Paenza');

**Problema:**

Trabaje con la tabla llamada "medicamentos" de una farmacia.

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 int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

primary key(codigo)

);

3- Ingrese algunos registros:

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01');

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01');

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01');

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01');

insert into medicamentos

values('Bayaspirina','Bayer',2.10,150,'2009-12-01');

insert into medicamentos

values('Amoxidal jarabe','Bayer',5.10,250,'2010-10-01');

4- Recupere los nombres y precios de los medicamentos cuyo laboratorio sea "Bayer" o "Bago" empleando el operador "in" (4 registros)

5- Seleccione los remedios cuya cantidad se encuentre entre 1 y 5 empleando el operador "between" y luego el operador "in" (2 registros):

Note que es más conveniente emplear, en este caso, el operador ""between".

**Solución:**

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

drop table medicamentos;

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

primary key(codigo)

);

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01');

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01');

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01');

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01');

insert into medicamentos

values('Bayaspirina','Bayer',2.10,150,'2009-12-01');

insert into medicamentos

values('Amoxidal jarabe','Bayer',5.10,250,'2010-10-01');

select nombre,precio from medicamentos

where laboratorio in ('Bayer','Bago');

select codigo , nombre , laboratorio , precio , cantidad , fechavencimiento

from medicamentos

where cantidad between 1 and 5;

select codigo , nombre , laboratorio , precio , cantidad , fechavencimiento

from medicamentos

where cantidad in (1,2,3,4,5);

# Búsqueda de patrones (like - not like)

Existe un operador relacional que se usa para realizar comparaciones exclusivamente de cadenas, "like" y "not like".

Hemos realizado consultas utilizando operadores relacionales para comparar cadenas. Por ejemplo, sabemos recuperar los libros cuyo autor sea igual a la cadena "Borges":

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

where autor='Borges';

El operador igual ("=") nos permite comparar cadenas de caracteres, pero al realizar la comparación, busca coincidencias de cadenas completas, realiza una búsqueda exacta.

Imaginemos que tenemos registrados estos 2 libros:

"El Aleph", "Borges";

"Antologia poetica", "J.L. Borges";

Si queremos recuperar todos los libros de "Borges" y especificamos la siguiente condición:

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

where autor='Borges';

Sólo aparecerá el primer registro, ya que la cadena "Borges" no es igual a la cadena "J.L. Borges".

Esto sucede porque el operador "=" (igual), también el operador "<>" (distinto) comparan cadenas de caracteres completas. Para comparar porciones de cadenas utilizamos los operadores "like" y "not like".

Entonces, podemos comparar trozos de cadenas de caracteres para realizar consultas. Para recuperar todos los registros cuyo autor contenga la cadena "Borges" debemos tipear:

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

where autor like "%Borges%";

El símbolo "%" (porcentaje) reemplaza cualquier cantidad de caracteres (incluyendo ningún caracter). Es un caracter comodín. "like" y "not like" son operadores de comparación que señalan igualdad o diferencia.

Para seleccionar todos los libros que comiencen con "M":

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

where titulo like 'M%';

Note que el símbolo "%" ya no está al comienzo, con esto indicamos que el título debe tener como primera letra la "M" y luego, cualquier cantidad de caracteres.

Para seleccionar todos los libros que NO comiencen con "M":

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

where titulo not like 'M%';

Así como "%" reemplaza cualquier cantidad de caracteres, el guión bajo "\_" reemplaza un caracter, es otro caracter comodín. Por ejemplo, queremos ver los libros de "Lewis Carroll" pero no recordamos si se escribe "Carroll" o "Carrolt", entonces tipeamos esta condición:

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

where autor like "%Carrol\_";

Otro caracter comodín es [] reemplaza cualquier carácter contenido en el conjunto especificado dentro de los corchetes.

Para seleccionar los libros cuya editorial comienza con las letras entre la "P" y la "S" usamos la siguiente sintaxis:

select titulo,autor,editorial

from libros

where editorial like '[P-S]%';

Ejemplos:

... like '[a-cf-i]%': busca cadenas que comiencen con a,b,c,f,g,h o i;

... like '[-acfi]%': busca cadenas que comiencen con -,a,c,f o i;

... like 'A[\_]9%': busca cadenas que comiencen con 'A\_9';

... like 'A[nm]%': busca cadenas que comiencen con 'An' o 'Am'.

El cuarto caracter comodín es [^] reemplaza cualquier caracter NO presente en el conjunto especificado dentro de los corchetes.

Para seleccionar los libros cuya editorial NO comienza con las letras "P" ni "N" tipeamos:

select titulo,autor,editorial

from libros

where editorial like '[^PN]%';

"like" se emplea con tipos de datos char, nchar, varchar, nvarchar o datetime. Si empleamos "like" con tipos de datos que no son caracteres, SQL Server convierte (si es posible) el tipo de dato a caracter. Por ejemplo, queremos buscar todos los libros cuyo precio se encuentre entre 10.00 y 19.99:

select titulo,precio from libros

where precio like '1\_.%';

Queremos los libros que NO incluyen centavos en sus precios:

select titulo,precio from libros

where precio like '%.00';

Para búsquedas de caracteres comodines como literales, debe incluirlo dentro de corchetes, por ejemplo, si busca:

... like '%[%]%': busca cadenas que contengan el signo '%';

... like '%[\_]%': busca cadenas que contengan el signo '\_';

... like '%[[]%': busca cadenas que contengan el signo '[';

**Ejercicio:**

Trabajamos con la tabla "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(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','J. L. Borges','Planeta',null);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros

values('Martin Fierro','Jose Hernandez',default,40);

insert into libros

values('Aprenda PHP','Mario Molina','Nuevo siglo',56.50);

Recuperamos todos los libros que contengan en el campo "autor" la cadena "Borges":

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

where autor like '%Borges%';

Seleccionamos los libros cuyos títulos comienzan con la letra "M":

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

where titulo like 'M%';

Seleccionamos todos los títulos que NO comienzan con "M":

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

where titulo not like 'M%';

Si queremos ver los libros de "Lewis Carroll" pero no recordamos si se escribe "Carroll" o "Carrolt", podemos emplear el comodín "\_" (guión bajo) y establecer la siguiente condición:

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

where autor like '%Carrol\_';

Buscamos los libros cuya editorial comienza con las letras entre la "P" y la "S":

select titulo,autor,editorial

from libros

where editorial like '[P-S]%';

Seleccionamos los libros cuya editorial NO comienza con las letras "P" ni "N":

select titulo,autor,editorial

from libros

where editorial like '[^PN]%';

Recuperamos todos los libros cuyo precio se encuentra entre 10.00 y 19.99:

select titulo,precio from libros

where precio like '1\_.%';

Recuperamos los libros que NO incluyen centavos en sus precios:

select titulo,precio from libros

where precio like '%.00';

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','J. L. Borges','Planeta',null);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros

values('Martin Fierro','Jose Hernandez',default,40);

insert into libros

values('Aprenda PHP','Mario Molina','Nuevo siglo',56.50);

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

where autor like '%Borges%';

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

where titulo like 'M%';

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

where titulo not like 'M%';

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

where autor like '%Carrol\_';

select titulo,autor,editorial

from libros

where editorial like '[P-S]%';

select titulo,autor,editorial

from libros

where editorial like '[^PN]%';

select titulo,precio from libros

where precio like '1\_.%';

select titulo,precio from libros

where precio like '%.00';

**Problema:**

Una empresa almacena los datos de sus empleados en una tabla "empleados".

1- Elimine la tabla, si existe:

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

drop table empleados;

2- Cree la tabla:

create table empleados(

nombre varchar(30),

documento char(8),

domicilio varchar(30),

fechaingreso datetime,

seccion varchar(20),

sueldo decimal(6,2),

primary key(documento)

);

3- Ingrese algunos registros:

insert into empleados

values('Juan Perez','22333444','Colon 123','1990-10-08','Gerencia',900.50);

insert into empleados

values('Ana Acosta','23444555','Caseros 987','1995-12-18','Secretaria',590.30);

insert into empleados

values('Lucas Duarte','25666777','Sucre 235','2005-05-15','Sistemas',790);

insert into empleados

values('Pamela Gonzalez','26777888','Sarmiento 873','1999-02-12','Secretaria',550);

insert into empleados

values('Marcos Juarez','30000111','Rivadavia 801','2002-09-22','Contaduria',630.70);

insert into empleados

values('Yolanda Perez','35111222','Colon 180','1990-10-08','Administracion',400);

insert into empleados

values('Rodolfo Perez','35555888','Coronel Olmedo 588','1990-05-28','Sistemas',800);

4- Muestre todos los empleados con apellido "Perez" empleando el operador "like" (3 registros)

5- Muestre todos los empleados cuyo domicilio comience con "Co" y tengan un "8" (2 registros)

6- Seleccione todos los empleados cuyo documento finalice en 0,2,4,6 u 8 (4 registros)

7- Seleccione todos los empleados cuyo documento NO comience con 1 ni 3 y cuyo nombre finalice en "ez" (2 registros)

8- Recupere todos los nombres que tengan una "y" o una "j" en su nombre o apellido (3 registros)

9- Muestre los nombres y sección de los empleados que pertenecen a secciones que comiencen con "S" o "G" y tengan 8 caracteres (3 registros)

10- Muestre los nombres y sección de los empleados que pertenecen a secciones que NO comiencen con "S" o "G" (2 registros)

11- Muestre todos los nombres y sueldos de los empleados cuyos sueldos incluyen centavos (3 registros)

12- Muestre los empleados que hayan ingresado en "1990" (3 registros)

**Solución:**

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

drop table empleados;

create table empleados(

nombre varchar(30),

documento char(8),

domicilio varchar(30),

fechaingreso datetime,

seccion varchar(20),

sueldo decimal(6,2),

primary key(documento)

);

insert into empleados

values('Juan Perez','22333444','Colon 123','1990-10-08','Gerencia',900.50);

insert into empleados

values('Ana Acosta','23444555','Caseros 987','1995-12-18','Secretaria',590.30);

insert into empleados

values('Lucas Duarte','25666777','Sucre 235','2005-05-15','Sistemas',790);

insert into empleados

values('Pamela Gonzalez','26777888','Sarmiento 873','1999-02-12','Secretaria',550);

insert into empleados

values('Marcos Juarez','30000111','Rivadavia 801','2002-09-22','Contaduria',630.70);

insert into empleados

values('Yolanda Perez','35111222','Colon 180','1990-10-08','Administracion',400);

insert into empleados

values('Rodolfo Perez','35555888','Coronel Olmedo 588','1990-05-28','Sistemas',800);

select nombre , documento , domicilio , fechaingreso , seccion , sueldo from empleados

where nombre like '%Perez%';

select nombre , documento , domicilio , fechaingreso , seccion , sueldo from empleados

where domicilio like 'Co%8%';

select nombre , documento , domicilio , fechaingreso , seccion , sueldo from empleados

where documento like '%[02468]';

select nombre , documento , domicilio , fechaingreso , seccion , sueldo from empleados

where documento like '[^13]%' and

nombre like '%ez';

select nombre from empleados

where nombre like '%[yj]%';

select nombre,seccion from empleados

where seccion like '[SG]\_\_\_\_\_\_\_';

select nombre,seccion from empleados

where seccion not like '[SG]%';

select nombre,sueldo from empleados

where sueldo not like '%.00';

select nombre , documento , domicilio , fechaingreso , seccion , sueldo from empleados

where fechaingreso like '%1990%';