28/03/00

Sistemas de datos

RDBMS: Sistema administración de Base de datos relacionables.

Relagional DateBase Management Sistem

Tiene programación, seguridad.

Oracle: es una red de RDBMS

Sybase

IBM DB/2

RDBMS Microsoft SQL Server

Guptta SQL

Informix

C.A. Jazmin

ANSI =(ISO)

Sequel

SQL (Standard query langer) Lenguaje standard común de consultas.

Herramientas (Programas que vienen con el sql server)

- SQL Enterprise Manager
- SQL Service Manager
- ISQL/W
- SQL Security Manager
- SQL Trace
- SQL Performance Monitor
- Microsoft Query (MS Query)
- SQL Server Web Assistant
- SQL Server Books on Line
- Microsoft ODBC
- SQL Server Driver
- SQL Distributed Management Objects
- SQL Setup.
- Manejador de servicios: prende o apaga servicios de NT
- Arma interface al motor de SQL: permite probar instrucciones al SQL Server.
- Permite manejar la seguridad (decide que usuario se puede conectar al SQL y cuales no).
- Muestra que hace el SQL Server, muestra las instrucciones que hace el SQL Server.
- Utilidad de configuración de clientes de SQL Server: especifica de que manera se tiene que conectar el

usuario al SQL Server.
- Monitor de performance del SQL Server: muestra actividad del SQL Server. Son indicadores de performance.
- Gráficamente indica que sentencias hay que darle al SQL.
- Permite mostrar datos del SQL Server en Internet (reporte HTML).
- Es la ayuda del SQL Server.

11−12. Forma de conectividad.

- Conjunto de funciones ya definidas (resumen para administrar SQL).
- Permite instalar − desinstalar programas del SQL Server.

Arquitectura del SQL Microsoft

Motor

Servicios prop. del

Sistema operativo

- Se encarga de administrar todo lo que es automatización. Ejecuta tareas programadas.
- Servicio motor de SQL Server. El motor en si mismo.

A−B Forman el C (lo que está detrás del corazón). La administración es complicada.

C. Facilitan la administración en Back End.

04/03/00

SQL se puede administrar por si mismo.

System Catalogo: es un conjunto de base de datos que son propios de SQL (del sistema), que utiliza el sistema para la administración.

Cada base de datos está orientada a una función (tarea) específica.

System Catálogo:

- Master database: (la base de datos maestra) Contiene un conjunto de elementos ej.: ctas. de usuarios. Guarda toda actividad interna del SQL Server.
- Model Database: es una base de datos modelo. Cada vez que se crea una base de datos copia la base modelo con el nombre que le damos.
- MSDB: Microsoft Database: contiene todos los elementos para interactuar con el Executive Service. La programación de tareas contiene datos dentro del MSDB.
- Temp Database: base de datos temporarios. Utilizada por SQL Server, para realizar trabajos temporales. No sirve para almacenar por largo tiempo.

Dentro del Master DataBase, Model DataBase, MSDB y Temp DataBase, existen tablas (tablas del sistema). Cada una almacena información específica.

Tablas: cada entidad es una tabla (un conjunto de tablas es una base de datos). Sirve para definir en forma lógica entidades de información. Se almacena en la Base de Datos.

Sys Users: (del sistema del usuario) guarda datos del usuario.

ERA: características que voy a almacenar de cada entidad.

Para que funcione hay que tener algo que permita almacenarlo.

Entidades: define que voy a almacenar. Ej.: cliente.

Relaciones: como la información de unos y otros se va relacionando.

Base de datos: conjunto de ERA que se almacenan en un dispositivo determinado.

Dispositivo: dentro del disco rígido tengo que tener una parte definida para el SQL Server.

Device (para almacenar datos): permite generar archivo físico que ocupa espacio para almacenar base de datos.

Todo se almacena dentro del dispositivo.

SQL

mediante el

Pedido

Almacena en un

Existen 2 tipos de Base de Datos: dentro del dispositivo solo pueden crearse Base de Datos.

- Data DataBase: contiene datos.
- Log DataBase: log (cuaderno de anotaciones del SQL) historial registro de anotaciones. Se utiliza para guardar información por un posible error, una posible pérdida.

Ventajas:

- Seguridad (deja base de datos = al log)

Por cada base de datos tiene que hacer un log.

Microsoft recomienda:

- Que los Device de Data y Log estén separados.
- El Device de Log tiene que se de ¼ a ½ de Data. Ej.: Data 25 o log 125 max.

Dentro de cada base de datos organiza los datos por páginas. Cada página ocupa 2k. Contiene información que yo le brindo en el SQL.

Para crear dispositivos (no base de datos)

Disk Init

Name =`Ejemplo_DV`, (nombre)

Physname=´C:\datos\EjeDV. Dat´, (fisical name. Nombre de archivo para S.O.)

Vdevno=15, (no es obligatorio, no puede repetirse, virtual device number, Nº disp.

Virtual).

Size=2048 (en página de 2k) x 2 = 4Mb (tamaño)

Disk Init

Name=`Ejemplo_log`,

Physname=`C:\log\el.dat`,

Vdevno=30,

Size=1024,

Después hay que apretar F5.

Borrar Device

SP_Drpdevice Ejemplo_log

Dar más o menos espacio al Device

Disk Recibe

Name=Ejemplo_Dv,

Size= 10000 (cantidad final de páginas de 2k, que deseo)

Crear Base de dato

Create Database Clase (nombre)

On (sobre) Ejemplo_Dv (el device de dato) =1024 (cant. de pág. De 2k que deseo tomar, si no

toma todo).

Log On Ejemplo_log=512

25−04−00

Tipos de datos soportados por Sql Server (que se pueden administrar)

	Tipo	Nomenclatura del Sql	
1.	BINARY	binary [(N)], var binary [(N)]	
2.	CHARACTER	char [(N)], var char [(N)]	
3.	DATE AND TIME	datetime, small date time	
4.	EXACT NUMERIC	decimal [(P[,S])], numeric [(P[,S])]	

5.	APPROXIMATE NUMERIC	float [(N)], real	Numéricos
6.	INTEGER	int, smallint, tin yent	
7.	MONETARY	money, small money	
8.	SPECIAL	bit, timestamp	Especiales
9.	TEXT AND IMAGE	text, image	

- *Binary*(binario): se guardan datos codificados a lenguaje máquina. Si específico el tamaño dentro del paréntesis solo guarda en ese tamaño.

var binary: se puede reducir al tamaño de la información guardada sin desperdiciar espacio, pero nunca va a aumentar su tamaño, siempre respeta el tope.

- *Character*(caracteres): no hay string, hay caracteres uno al lado del otro. Es como un vector, una cadena de caracteres).

El var char permite un máximo de 255 caracteres.

A igual que los binarios se diferencian por cual es el que permite reducir su tamaño según la información que se guarde. Analiza por fila no por tabla.

- *Date and Time:* Guarda datos de tipo fecha y hora: en Sql la fecha es el momento.

Devuelve hora/fecha/mes/año/hora/minutos/segundos.

La información se guarda en un mismo tipo de dato.

Puede guardar solamente la fecha o solamente la hora.

A diferencia del DATETIME el small date time guarda rango de fecha menores.

Numéricos:

Existen 3 categorías:

- *Exact numeric:* guarda el valor tal cual es. Es más lento porque tiene que ser exacto. Ocupa un espacio por bit.
    - Ambas clases son utilizados por igual

P (significa precisión): cantidad de dígitos en total que puede contener contando la parte entera y la parte decimal. Se pueden utilizar como máximo 38 números.

S (significa escala): se puede o no enumerar esa función cuando se utiliza por parte, decimal. Puede ser desde 0.

- *Approximate:* es más rápido, ocupa menos espacio y si se produce un desbordamiento lo redondea.
    - Real: número decimal real. Se ajusta de acuerdo al número
    - Float: no se utiliza porque no especifica parte de almacenamiento en bits.

- *Integer:*
    - Int: son los números que pertenecen a 32 bits.
    - Smalint: son los de 16 bits. (va desde −32.000 a +32.000)
    - Tyint: son los de 8 bits. (va desde 0 a 256),significa pequeño entero).
- *Monetary:*
    - *Money:* es un numeric más el signo $. Reconoce que se habla de dinero.
- Especiales:
    - *Special:*

Bit: 0 1 no puede almacenar más

Timestamp: tipo de dato que se genera en hexadecimal. Guarda el clock en el momento que se guarda la fila (no se puede ver ni operar, es único e irrepetible. Cuando se modifica la fila no se cambia).

- Text and image:

Texto: no tiene límites

Image: especial para guardar imágenes, pero no se utiliza con frecuencia.

Tipo de datos propios a la base

Se almacena dentro de la base de datos. Se llaman Use Defined Data Type UDDT (tipo de datos definidos por el usuario).

Ventajas:

- Claridad
- Uniformidad

Crear Tipo de Datos

Exec sp _ addtype (nombre del tipo de dato) código DJ, ´(tipo de dato)int´

go (separa comandos)

Exec sp _ addtype código postal, ´char[4]´

go

Las tablas están compuestas por columnas que dividen los datos

Crear Tablas

Use (nombre de la tabla) Ejemplo

go

(

(nombre de la columna)código (tipo de dato) código DJ (si acepta nulos o no)null o not null,

descripción char(100) not null,

fecha ejemplo datetime null

)

go

Buscar reglas de normalización para la creación de tablas(hay 4 formas 4FN)

Armar diseño de tabla para almacenar datos de clientes, mínimo tres entidades o tablas / relacionadas no más de 5.

Código C_TIPO_DOC

Nombre D_TIPO_DOC

Países Dirección Tipos_Documentos

C_PAIS Teléfono

D_PAIS N_Empresa

C_PAIS

C_TIPO_DOC

Nro_Documento

Clientes

Hay tres tipos de relaciones (de que manera se establece la relación)

- De 1 a 1
- De 1 a infinitos o muchos
- De muchos a muchos

Use Países

go

Create Table Países

(

C_PAIS smallint not null,

D_PAIS var char(20) not null

)

go

Use Tipos_Documentos

go

Create Table Tipos_Documentos

(

D_TIPO_DOC var char(5) not null,

C_TIPO_DOC smallint not null

)

go

Use Clientes

go

Create Table Clientes

(

go

(

Código smallint not null,

Nombre var char(50) not null,

Dirección var char(50) not null,

Teléfono tinyint null,

N_Empresa var char(30) null,

C_PAIS smallint not null,

C_TIPO_DOC smallint not null,

Nro_Documento tinyint not null, constraints pk_clientes Primary Key_clustered

)

go

02−05−00

Primary Key: clave primaria. Es una forma de mostrar ordenamiento. Identifica a las filas en forma única.

Constraints: son las relaciones. Son parte de la tabla. Se definen en el momento de creación de una tabla. Cada tipo está orientado hacia algún elemento de una tabla de datos. Los nombre son únicos, cada tabla tiene uno diferente.

Agregar clave primaria

Create table Países

C_Países smallint not null,

D_Países var char(50) null,

Constraints PK_Países Primary Key clustered/Non clustered

(C_País)

)

go

Es el nombre

Nombre con el que identifico el ordenamiento(que columna). Si agrego otro nombre de columna se separa con una coma.

Clustered: se ordena la fila de acuerdo al PK.

La ventaja es que la inserción es más rápida

La desventaja es que la búsqueda es más lenta.

1	2	3	4

Non clustered: se ordena a medida que se ingresa. Crea un índice que indica para cada posición el registro correspondiente.

3	1	4	Países PK
F2	F1	F3	Indice

04/05/00

ForeingKey: desde una tecla se verifica que la información dada sea correcta con otra tabla. Se utiliza para validación de datos y establecen relación. No se modifica el valor solo verifica.

Insert into Provincias

(

C_Prov,

D_Prov,

)values(

1,

Bs.As.

)go

Insert into Provincias

(

C_Prov

D_Prov,

)values(

2,

Córdoba

)go

Insert in to Clientes

C_Clientes,

D_Clientes,

C_Prov,

)values(

Javier,

5

)go

09/05/00

Create database `Ejemplo'

On(

Name=

Size=

)

log on(

)

Pedir datos:(consulta)

Select * (muestra todas las columnas)

From Clientes(desde y el nombre de la tabla)

Va a devolver:

Clientes	Nombre_Cliente	Domicilio
1	Javier	
2	Juan	
3	Pablo	

Select C_País, (hay que poner el nombre de las columnas en orden)

D_País

From países

Va a devolver:

C_País	D_País
1	Argentina
2	Chile
3	Brasil

Select código_País=C_País,

Nombre_País=D_País,

From países

Va a devolver:

Código_País	Nombre_País
1	Argentina
2	Chile
3	Brasil

Select Et1= Código:,

Código_Paises=C_País,

Et2= País:,

Nombre_País=D_País,

From paises

Et1	Código_País	Et2	Nombre_País
Código:	1	País:	Argentina
Código:	2	País:	Chile
Código:	3	País:	Brasil

Select Et1=Código:,

Código_Paises=C_País,

Et2=País:,

Nombre_Paises=D_País,

Total_Riesgo_País=N_Países + 2,

From países

	Function	Parameters
1.	ABS	(Numerics_expr)
2.	Acos, Asin, Atan	(Float_expr)
3.	Atn2	
4.	Cos, Sin	(Float_expr)
5.	Cto, Tan	
6.	Ceiling	(Numerics_expr)
7.	Deg Rees	(Numerics_expr)
8.	Exp	(Float_expr)
9.	Floor	(Numerics_expr)
10.	Log	(Float_expr)
11.	Logno	(Float_expr)
12.	Pi	( )
13.	Radians	(Numerics_expr)
14.	rand	([Seed])
15.	Round	(numeric_expr, length)
16.	Sign	(numeric_expr)

Function:

- Devuelve el valor absoluto. Cualquier número.
- Arco coseno, arco seno, arco tangente.
- Arco cotangente.

- Coseno, seno.
- Cotangente, tangente.
- Devuelve valore entero más cercano redondeando hacia arriba.
- Convierte radianes.
- Exponencial (parte).
- Es el valor más grande que se acerca al valor especificado (menor, igual).
- Logaritmo.
- Logaritmo en base 10.
- Devuelve constante.
- Transforma el valor a radianes.
- Genera valor aleatorio entre 0 y 1. Valores al azar.
- Redondea. Formatea valor decimal.
- Devuelve el signo. (positivo, negativo o si es 0).
- SQRT raíz cuadrada.

	Function	Parameters
1.	+	(Expression)
2.	ASCII	(Char_expr)
3.	Char	(Integer_expr)
4.	Lower	(Char_expr)
5.	LtRim	(Char_expr)
6.	Reserve	(Char_expr)
7.	Right	(Char_expr, Integer_expr)
8.	RtRim	(Char_expr)
9.	Space	(Integer_expr)
10.	Str	(folat_expr[,length[,decimal]])
11.	Sustring	(expressions, start, length)
12.	Upper	(Char_expr)

- Se utiliza para concatenar strings.
- Dado un carácter devuelve un valor ASCII.
- Dado un calor entero correspondiente a un ASCII devuelve un carácter.
- Pasa todo lo que sea string a minúscula.
- Borra los espacios libres de la izquierda.
- Invierto el string.
- Dado un string devuelve cantidad de caracteres contando de derecha a izquierda.
- Elimina espacios de la derecha.
- Dado un valor numérico devuelve un string de N espacios.
- Toma un número y lo devuelve como string.
- Permite dado un string tomar sólo una parte del mismo.
- Devuelve el string con todos sus caracteres en mayúsculas.

16/05/00

Filtros.

Filtros de datos de consulta es un modificador del select. Filtra filas no columnas.

Where: puede traer de 0 a N filas. Indica condición que se tiene que cumplir. Siempre devuelve un resultado aunque sean 0 filas.

Select * (muestra todas las columnas)

From Clientes

Where (C_Prov=1 and

C_Clie>1000) or

C_Clie<10

Select N_Factu,

C_Arti,

Precio = ABS(q_Arti) * P_Arti

From Detalle_Facturas

Where N_Facturas>500 and

N_Item = 1 and (ABS (q_arti)>2)

Dentro de la condición Where se puede evaluar con números =, >, <, >=,<= o el <>. Se pueden tener N condiciones y una condición con otra se relacionan con and, or. Se pueden agrupar entre (). Se pueden operar con columnas aunque no se muestren.

Between (entre): da los datos contenidos entre 2 valores.

Select *

From Clientes

Where F_Ingreso between ´3/1/2000´and ´5/31/2000´

Select C_Clien,

F_Factu,

Fecha_N = Convert (Datetime, Convert(Char(2), DatePart(mm, F_Venci)) +

/ + Convert(Char(2), DatePart(dd,F_Venci))+ / +

Convert(Char(4), DatePart(yyyy, Get Date())))

From facturas

Select *

From facturas

Where datepart (mm, F_Facturas) = datepart (mm, get date()) and

Datepart (yyyy, F_Facturas) = datepart (yyyy, get dat())

Like compara o busca un string carácter por carácter aunque estén en mayúsculas o en minúsculas.

El = compara carácter por carácter.

El % se utiliza como comodín, significa que no importa cual sea el contenido, se parece al *.

Select *

From Clientes

Where D_Clientes Like ´Javier de Jorge´

Where D_Clientes Like ´Javier%´

Where D_Clientes Like ´%Javier´

Where D_Clientes Like ´%Javier%´

18/05/00

Cómo se hace para tomar solo una parte de una fila.

Ej.: tomar solo los datos de un cliente cuyo código sea 1, 3, 5, 9, 11.

Select *

From Clientes

Where C_Provincia in (1,3,5,9,11)

En este caso el Where se utiliza como filtro en vez de tomar la fila toma solo la parte que emplea con una condición.

Select *

From Clientes

Where C_Provincias(1, 3, 5, 9,11) Evalúa cada código con esta lista.

Para pedir en tabla Provincias todos los códigos de provincias y que solo muestre los valores distintos que no se repitan. Para eso uso el Distinct.

Select Distinct C_Provincias Select distinct C_Pais, C_Provincias

From clientes From clientes

Todos estos filtros vistos hasta ahora son simples.

Para ordenar el resultado de un Select

Order By *y luego se coloca las columnas.*

Select * (Ordena todas las filas de la tabla clientes por el nombre).

Order By D_Cliente

Select *

From Clientes

Where C_Provincias (1, 5, 98, 115)

Order By D_Cliente

Si quiero ordenar por varias columnas se colocan los nombres de las columnas separados por comas.

Select *

From clientes

Where C_Provincias(1, 5, 98, 115)

Order By C_Provincias, D_Clientes

23/05/00

Select Distinct selección en fila cuyas columnas forman combinación, si la fila se repite no la muestra, la filtra. Devuelve distintas filas. Se utiliza por lo general para columnas numéricas.

Información sumarizada: cuántos clientes. Se utiliza para saber un valor.

Sumary Data: genera información que no es de una fila en particular, si no que es la suma de algunas de ellas.

Funciones.

- Avg: devuelve el promedio

Select avb(q_Cont) *Esta es una columna*

From Facturas_Items

Where C_Código_Prod = 1

- Count: es un contador.

Select Count(*) *Cuenta las filas sin importar el contenido*

From Clientes

Where F_Ingreso >= Convert(Date, 01/01 + Convert(Char(4), Datepart(yyyy, GetDate()))

- Max: devuelve el valor máximo.

Select Max(q_Cant)

From Facturas_Items

- Min: devuelve el menor valor.

Select Min(q_Cant)

From Facturas_Items

- Sum: devuelve la suma de alguna columna

Select Sum (q_Cant * I_Prod)

From Facturas_Items

Where N_Facturas = 800

Sumarización de datos.

Devuelve funciones:

Select C_Categoría, Count(*)

From Clientes,

Group By C_Categoría

Devuelve Categorías

Select C_Cate

From Clientes

Group By C_Cate

Select N_Carrera(*esta columna es la que muestra)*, Avg(E_Alumno)

From alumnos

Group By N_Carrera

Compute: puede generar el detalle y el total.

Select N_Carrera, E_Alumno

From Alumnos

Compute Avg (E_Alumno)

1 1 20

1 1 30

1 2 25

AVG

25

Devuelve todas las filas de todos los alumnos y al finalizar devuelve el promedio de edad.

30/05/00

	Clientes		**Países**	
	C_Clientes		C_País	
	D_Clientes		D_País	
	C_País			

Select A01.C_Clie,

A01.D_Clie,

T01.D_País,

From Clientes A01 *(alias)*

Left Outer Join Países T01

On A01.C_País=T01.C_País

C_Clie	D_Clie	D_País
1	María	Argentina
2	Pedro	<Null>
3	Juan	Francia

El left and right establece una relación que no se da físicamente solo la hace un esa consulta.

Cross Join es el tipo de Join que realiza mezcla de 2 tablas en busca de todas las posibilidades.

C_Clie	D_Clie	C_País	D_País
1	María	1	Argentina
2	Pedro	2	Alemania
		3	Uruguay

Select A01.C_Clie,

A01.D_Clie,

T01.D_País,

From Clientes A01

Cross Join Países T01

C_Clie	D_Clie	D_País
1	María	Argentina
1	María	Alemania
1	María	Uruguay
2	Pedro	Argentina
2	Pedro	Alemania
2	Pedro	Uruguay

Left Join es sobre la tabla que se realiza el from.

C_Clie
C_Clientes
N_Clientes
N_Grupo Fam

Clientes titulares que pertenezcan al grupo familiar.

Select A01.C_Clien,

A01.D_Clien,

A01B.C_Clien

From Clientes A01

Inner Join Clientes A01B

On A01.N_Grupo_Fam=A01B.C_Clie (cuando no tiene nada lo filtra)

Sub Quary: consulta de la consulta. Se puede tener dentro de un Select otro Select. Devuelve filas.

Clientes		Facturas_Header *(cabecera)*		Facturas_Items		Articulo
PK C_Clientes		PK N_Factu		PK N_Factu		PK C_Arti
D_Clientes		F_Factu		PK N_Items		D_Arti
		C_Tipo_Factu		C_arti		P_Uni
		F_Vto		Q_Arti		
		C_Clie		P_Arti		

Realizar una consulta que devuelva el número de la factura, la fecha de la factura, el nombre del cliente, código del artículo, nombre del artículo, cantidad y precio unitario.

Select f.n_factu,

f.f_factu,

cli.d_clie,

a.c_arti,

a.d_arti,

fi.g_arti,

fi.p_unit

from facturas_header f

inner join clientes cli

on f.c_clie = cli.c_clie

inner join facturas_items fi

on fi.n_factu = f.n_factu

inner join artículos a

on a.c_arti = fi.c_arti

Realizar una consulta que devuelva el total de la factura y el número de la factura.

Select f.n_factu,

Sum (f.q_arti * f.p_uni)

From facturas_items f

Where f.n_factu = 50

01/06/00

El left joinn cuando encuentra coincidencia muestra datos referenciados cuando no los encuentra muestra los datos de la izquierda.

Inner join filtra los datos cuando no hay en alguna de las dos columnas.

c_clie d_clie c_pais c_país d_país

1 María 1 1 Argentina

2 Verónica Null 2 Chile

3 Soledad 2 3 Brasil

Select A01.c_clie,

A01.d_clie,

T01.d_país

From clientes

Cross join Países T01(evalúa todas las posibilidades no importa si hay o no dato.

c_clie	d_clie	d_pais
1	María	Argentina
1	María	Chile
1	María	Brasil
2	Verónica	Argentina
2	Veronica	Chile
2	Veronica	Brasil
3	Soledad	Argentina
3	Soledad	Chile
3	Soledad	Brasil

El filtro que más restringe es el inner join. Es una relación estricta, si o si los valores tienen que coincidir.

Select A01.c_clie,

A01.d_clie,

T01.d_país

From clientes

Inner join Países T01

On A01.c_pais = T01.c_pais

c_clie	d_clie	d_pais
1	María	Argentina
3	Soledad	Chile

El left o right: muestran todas las filas de una tabla y solamente las filas de la otra tabla que son coincidentes.

La diferencia entre uno y otro es la tabla que se filtra y cual es la tabla que se muestra.

Select A01.c_clie,

A01.d_clie,

T01.d_país

From clientes A01

Left outer join países T01

On A01.c_pais = T01.c_pais

c_clie	d_clie	d_pais
1	María	Argentina
2	Verónica	Null
3	Soledad	Chile

Select A01.c_clie,

A01.d_clie,

T01.d_país

From clientes A01

Right outer join países T01

On A01.c_pais = T01.c_pais

c_clie	d_clie	d_pais
1	María	Argentina
3	Soledad	Chile
Null	Null	Brasil

Subquery: mientras se genera una consulta se realiza una subconsulta. Por cada fila del Select principal se evalúa el select interno.

Select A01.c_clie,

A01.d_clie

From clientes A01

Where c_pais in (select distict c_pais

From países)

13/06/00

Generación de resultado de consultas.

Primero se realiza el Create Table

Luego:

Insert in to #(almohadilla significa que es temporal. Cuando el cliente se desconecta se borra)

Select c_clie,

d_clie

from clientes

where c_clie >500

Borras filas en tablas.

Borrar todas las filas:

Delete *

From clientes

Cuando existan Constraint no borra las filas.

Delete

From clientes

Where c_clie notin(Select distinct c_clie

From fac_headers) Todos los clie. Que no tengan facturas.

Un delete es siempre de una sola tabla. No hay joins, pero puede haber subquery.

Actualización de datos.

Update Artículos (nombre de la tabla)

Set p_arti = ((p_arti * 10)/100)*100) + p_arti), (por cada columna valores que se le va a dar)

P_compra = ((p_compra * 5)/100) + p_compra,

D_arti = Clavos

Where c_linea = 1 (esto es para cuando se actualiza una fila cuando se nombra la PK, ej: where c_clie = 50)

Índice(para base de datos).

Es un ordenamiento lógico de los datos. Permiten especificar criterio. Posibilidad de ver de manera ordenada una serie de datos.

Actúa con una o más columnas de una tabla por tabla se pueden tener N.

Acelera los procesos de selección y actualización de datos.

Creación

Por cada clase de índice (valor distinto del índice) guarda una valor distinto.

Cuando se hace un Insert, Update o un Delete se actualiza.

No conviene utilizar los índices cuando:

- El índice donde haya poca variedad de valores.

- Cuando se hagan consultas muy poco.
- No son convenientes los índices alfanuméricos.

Si es conveniente utilizarlos cuando:

- Son columnas que se consultan diariamente.
- Columnas que son Constraints (las pk son índices automáticos)
- Y sobre valores numéricos o exactos.

Create Index id_clie *(nombre del índice)* on clientes *(nombre de la tabla)* (col1,col2,col3) *(nombre de las columnas)*

Programar.

Procedimientos almacenados (Stored procedures): un programa de sql contiene conjuntos de sentencias sql escritas en forma lógica. Es un elemento más de la base de datos. Es propio de cada base de datos (no de las tablas) y se guarda con cada una. Cada base de datos puede tener varios sp (puede hacer referencia a una o a varias tablas, pueden llamar (hacer referencia) u otro sp y pueden devolver datos) contienen código sql.

Crear SP.

No se puede modificar no contiene datos contienen operaciones.

Create Procedure consul_clie_fech (nombre no puede comenzar con las letras sp)

@c_clie int,

as Select c_clie, d_clie, f_ingre

from clientes

Order by f_ingre

From clientes

Where c_clie = @c_clie or @s_clie = 0 (devuelve toda la tabla)

Order by f_ingre

Devolución de datos.

Exec consul_clie_fech(nombre del procedimiento) 50

20/06/00

Declaración de variables dentro de un sp.

Declare @bandera int (debajo del as y tiene que estar antes de utilizarse).

Create Procedure pc_consulta

@n_socio_int

As

declare @bandera int

Select @bandera = select count(*)

from socios A01

where A01.n_grup_familiar = @n_socios

If @bandera <> 0 (condición)

Begin (Declara bloques de código)

Select A01.c_socio,

A01.d_socio,

Cantidad = @bandera

From socios A01 where A01.c_clie = @n_socios

End

Else

Begin

Select A01.c_Socio

A01.d_socio

From socios

Where A01.c_socio = @n_socio

End

Ejecución.

Pc_consulta 50 (este es un valor para @n_socio

Vistas lógicas.

Singnifican tablas virtuales. Son tablas que no existen como tablas, si no que son representadas a través de consultas. Para el cliente a la vista es una tabla más con ciertas restricciones.

Las vistas lógicas son más rápidas y no son parte de una tabla son parte de la Base de Datos.

Ventajas:

- Permite ocultar diseño de la base de datos.

- Permite reforzar la seguridad de la base de datos(no permite modificar).
- No le doy acceso a la tabla real, si no que le doy permiso a la vista lógica.
- Se puede seleccionar filas.
- Se puede hacer joins.
- Función de agregación (es todo lo que sea consulta).

Creación.

Create View Vista_clientes (nombre de la vista lógica)

As

Select c_socio,

d_socio

from socios

Llamada.

Select *

From vistas_clientes

Select A01.d_socio

From vistas_clientes A01

Where A01.d_socio >50

Create view vista_clientes

As

Select A01.c_socio,

A01.d_socio,

T01.d_provincia

From socios A01

Inner join Provincias T01

on T01.c_provincias = A01.c_provincia

Vistas lógicas.

Es más rápidas. No es parte de una tabla es parte de la base de datos.

En ella se pueden relacionar filas. No se recomienda actualizar las Vistas Lógicas. En ellas se pueden hacer joins y también se pueden hacer funciones de agregación (todo lo que sea consulta).

El Log Divice almacena las distintas operaciones que se van realizando. Permite programar antes de grabar físicamente una operación.

En la base de datos existen Tablas, Store Procedure, Vistas Lógicas, Default, Rolls, User Define data type, programas de mantenimientos de la base de datos y permisos de usuarios.

Trigger: todas las tablas los tienen. Se ejecutan antes de hacer la modificación física. No se ejecuta solo. Se tiene que ejecutar por una operación. Es solo de las tablas no pertenece a la base de datos.

Existen 3 tipos:

- For insert
- For delete
- For update

Cada tipo de trigger es una operación. Siempre se ejecutan.

Se utilizan en necesidades extremas (si no se puede hacer un S.P. o una Vista Lógica).

Hacen mas lento al sql. Cuando las cuentas no son uniformes (se necesita validación de datos antes de una modificación).

Creación.

Create trigger tg_aA01_in *(nombre del trigger)*

On clientes

For insert

As

Código sql

Ejemplo: si la fecha de ingreso del cliente corresponde al mes pasado a la columna calificación la incremento en 1 y si es de este mes necesitamos que quede igual.

Inserted set: se utiliza antes de hacer algo físicamente. Es el log que lo escribe y después lo hace físicamente.

Create trigger tg_A01_in

On clientes

For insert

As

Declare @f_ingreso datetime

Select @f_ingreso = select f_ingreso from inserted

If datediff (mm,@f_ingreso, get date())>1

Begin

Update insted

Set

C_calif=c_calif + 1

End if

22/06/00

Un trigger puede cancelar una operación (insert − delete − update)

Sql trabaja bajo el concepto de:

Transacción (comienza una operación y realiza un segunda hasta que se le da un Ok a toda la operación.

Por cada trigger el comienzo lo hace sql.

Raise error: devuelve a quien lo llamó

No significa que se haya terminado la transacción y puede hacer que devuelva un error como si fuera propio del sistemas como por ejemplo cuando nos olvidamos de poner una coma en el select en sql que nos dice que hay un error cerca de .....

Raise error (30016 *(nº de error),* 15 *(nivel de error), (mensaje de error)*)

Rollback transaction: deshago todo lo hecho hasta la transacción. Aborda transacción (volver atrás).

Ejemplo: teniendo la tabla clientes con títulos o solo viene un programa que quiere borrar un socio pero antes debe evaluar si es socio o si es grupo familiar, si es grupo familiar evalúa si es titular o si tiene alguien a cargo no se puede borrar; si es socio solo no hay ningún problema y se lo borra.

Create trigger tg_clie_del

On clientes

For delete

As

Declare @n_socio int

Declare @n_gf int

Declare @n_cont int

Select @n_socio = select n_socio from deleted

Select @n_gf = select n_grupo_fami from deleted

```
If @n_gf = @n_socio

Begin

Select @n_cant = (select count (*)

From socios

Where n_grupo_fami = @n_socio)

If @n_cant >1

Begin

Raise error (parámetros)

Rollback trans

End

End

Create trigger tg_clie

On movimientos

For update

As

Declare @c_clie int

Select @c_clie = (select c_clie from deleted)

If @c_clie <> (select c_clie from insertde)

Begin

Select @c_clie = (select c_clie from inserted)

End

If (select count (*)

From tabla_aux

Where c_clie = @c_clie)>0

Begin update tabla_aux

Set f_actb = get date()
```

Where c_clie = @c_clie

End

Else

Begin

Insert into tabla_aux

(c_clie, fecha_actu) value

(@c_clie, get dat())

end

27/06/00

El riseerror se puede ejecutar desde un store procedure

Indices.

Son objetos propios de la base de datos que trabajan para una tabla determinada. No es parte de la tabla.

Es un objeto que permite ordenar la tabla de una manera determinada, distinta a la forma que esta guardada.

Se crea el índice en cualquier momento (en la creación de la tabla o no) y es mantenido automáticamente por sql.

Es una estructura que muestra una manera distinta de ordenación pero los datos residen en el mismo lugar.

Ventajas:

- Permite ejecutar las consultas de una manera más rápida.
- Los joins tratan de realizar las acciones a través de los índices.
- No es necesario especificar que índice se tiene que utilizar.

Es conveniente la utilización de índices cuando:

- Se necesita una ordenación determinada para consultas que son utilizadas con frecuencia.
- Se usan filtros en varias consultas que son utilizadas con frecuencia.
- Se hagan consultas donde se hagan referencias a otras tablas(joins).

Se puede tener N índices por tablas.

Se puede utilizar para una sola columna o para un grupo, que no estén comprendidas en el PK porque el PK ya es de por si un índice.

No es conveniente la utilización de índices cuando:

- Hago consultas esporádicas. Esto significaría sobrecargar a sql.
- Los datos son uniformes entre las filas.

Existen distintos tipos. Si se hace un índice por dos o más columnas, cuando se hacen los filtros (Where) se tienen que hacer por las 2 columnas.

- Clustered: ordenamiento físico igual que lógico. Pueden existir solamente si las PK es non

Clustered (solamente puede existir un Clustered por columna).

- Non Clustered: ordenamiento lógico. Si no se especifica nada son todos non Clustered.
- Unique: cuyo valor o conjunto de valores no se puede repetir. Es un auxiliar del PK. Se puede tener la cantidad que se desee.
- Composite index: índice compuesto más de una columna.

Consultas en sql.

Create index ind_clientes *(nombre del índice)* on clientes *(columna,col)* Clustered *(se pone o no se pone nada. Si se pone unique puede ser Clustered).*

Go

Update_satistics *(estadística de la tabla)* clientes *(clientes)*

Esta estadística es conveniente realizarla cada 3 o 6 meses.

La integridad referencial de los datos.

Es al coherencia que tiene la base de datos sobre la información que contiene. Una base mal normalizada no tiene integridad de los datos.

Existen distintos niveles:

- Nivel de tabla: se puede asegurar de que siempre todas las tablas tengan una PK, si no existe la misma se tiene que generar una.
- Columna de tipo identity: se encarga de administrar a sql y genera para esa tabla valores únicos. Es una propiedad de la columna. Son valores numéricos. Se aplican a columnas de valores enteros. El incremento de este valor también es entero y lo especifica uno mismo. Una sola columna de este tipo por tabla (solamente administra uno). Es una solución para cuando no hay PK y no es obligatorio que la columna identity sea el PK.

Create table clientes

(c_numero int identity 1,1 *(desde que valor. Incremento)*)

go

No hay forma de generar o de regenera el valor. La única manera es hacer el Drop de la tabla. Por más que yo tenga el 25, después lo elimino y vuelvo a generar otro, automáticamente genera el 26.

Variables del sistema (@@*nombre*): son las que maneja el server.

@@identity: devuelve justo después de hacer el insert que número se genero. No se confunde en el tiempo en que tardo de hacer un insert del otro.

Ejemplo: devolver al usuario que nº. De cliente tiene.

Create table usuarios

(d_usua int identity (1,1),

d_user var char(50) not null,

d_pass var char (50) null

cosntraint PK_usuarios primary key Clustered (c_usua))

go

create procedure user_alta

@d_user var char (50),

@d_pass var char(50)

as

insert into usuarios

(D_user,d_pass)

values

(@d_user,@d_pass)

retturn @@identity *(retturn devuelve como el recordset)*

go

Contenido de columnas.

Reglas y default: seria dentro de la base de datos. Se puede aplicar a cualquier columna de la tabla.

Reglas: especifica que el contenido de esa columna cumpla con la condición.

Defautl: si no se especifica valora a columna asociada, le da un valor del default definido. (si se da enter si ingresar datos, le asigna el valor que tienen el default declarado).

29/06/00

Un default es un elemento que se encuentra dentro de la base de datos y que establece un valor. No depende de la tabla si no de la base de datos.

Creación de un default.

Create Dafault cero (es el nombre)

As 0 (valor default)

Go

Creación de una regla.

Create Rule Rv_sexo

As

@codigo = M or

@codigo = F

go

El @codigo es el valor que va a tomar la regla.

Normalmente los default se aplican a las columnas de las tablas. Tengo que crear un Store Procedure que es propio del sistema.

Sp_BindRule *(bindear la regla)* Clientes, codi_sex, ru_sexo *(en que tabla, columna y que regla)*

Para crear un default

SP_Bind Default Facturas_Items, Importe, cero

Cada vez que se ingrese un dato va a validar que se cumpla la regla. Se comprueba cada vez que carga un valor. Me permite mantener la integridad de los datos.

Cada vez que no se especifique un valor para esa columna me muestra el valor 0.

Los default no se pueden modificar. Lo primero que hay que hacer es borrarlo y vuelvo a crearlo. Para esto no tiene que estar bindeado a ninguna columna, para desbindearlo uso UnbindDefault.

Drop default cero

Sp_UnbindDefault facturas_Items, Importe, Cero

Los C*onstraint* son elementos de las tablas, que me permiten asegurar la integridad de las tablas a partir de unas propiedades.

- PK determina que no se repitan valores.
- FK asegura de que el valor que se esta ingresando existe en la tabla sobre la que se hace referencia.
- UNIQUE permite especificar que las columnas que aparecen en unique deben mantener un valor único sin llegar a ser la clave primaria.

Los constraint se agregan a la tabla, por lo cual puedo modificarlas usando:

Alter Table Usuarios

Add

Constraint U_Clave *(nombre de la columna)* Unique non Clustered

(d_password)

go

WithNocheak: lo deja como esta y a los demás lo modifica.

Cursores: mantiene el resultado en la memoria de sql y me permite recorrerla con el rule.

04/07/00

Para armar un cursor tengo que crear un Store Procedure.

Un cursor devuelve fila a fila.

Tengo que almacenar esas filasa en variables.

- Primer paso hay que declarar las variables.
- Luego guardar los datos en una tabla temporal, que no exista más cuando se deje de ejecutar el Store Procedure. No se necesita una identity porque ya hay clase unica.
- Declarar un cursor. Preparar un cursor para un select y especificar cual es. Todavía no se puede utilizar. Solo se lo declara.

Al abrir un cursor con la sentencia Open cur_Datos no significa que ya nos esta devolviendo datos, lo único que hace es resolver el select y guardarlo en memoria y a la vez le devuelve un puntero al objeto cur_datos.

La centencia Fetch toma lo que hay en el cursor y asigna los valores a las variables. Se recorre las filas siempre y cuando haya datos. Cuando no hay datos existe una variable que indica si hay o no hay datos. La misma se llama @@fetchstatus.

Ejemplo: si el código de actividad es del 1 al 10 hay que aplicar un 15% de descuento. Si el codigo de la cantidad supera los 10 habrá que aplicarle un 10%.

Siempre hay que guardar en forma temporal

Create Procedure Calculo

@mes int

As

Declare@n_socio int

Declare@c_acti int

Declare@q_acti numeric (5,2)

Create table#Proceso

(N_socio int,

```
C_acti int,
Q_acti numeric (5,2)
Constraint PK_Temp Primary Key Clustered (N_Socio, C_Acti))
Declare Cur_Datos cursor
For
Select n_socio,
c_acti,
q_acti,
from movimientos_socios
where datepart(m,f_acti) = @mes and datepart (yyyy,f_acti) = datepart (yyyy, getdate())
order by n_socios, c_acti
Open cur_datos
Fetch cur_datos
Into @n_socio, @c_acti, @q_acti
While @@fethcstatus = 0
Begin
If @c_acti <10
Begin
Select @q_acti = @q_acti − ((@q_acti * 15)/100)
End
If @c_acti <=100 and @c_acti and @c_acti >= 10
Begin
Select @q_acti = @q_acti − (@q_acti *10)/100)
End
If (Select (*) from #proceso where n_socio = @n_socio and c_acti = @c_acti)=0
Begin
```

Insert into #Proceso

(n_socio, c_acti, q_acti) values

(@n_socio, @c_acti, @q_acti)

end

else

Begin

Update #Proceso

Set q_acti = @q_acti + @q_acti

Where n_socio = @n_socio and

c_acti = @c_acti

end

Fetch cur_datos into @n_socio, @c_acti, @q_acti

End (proceso)

Select *

From #proceso

REPASO DE TRIGGERS

Un cliente de un video club puede llegar a extraviar el carnet de socio y por lo tanto hay que llevar una tabla donde se registren los distintos carnets del socio. (como si fuese un historia)

Create trigger tg_socios_inst

On socios

From insert

As @n_socio int

Select @n_socio = select n_socio from inserted

Insert into carnets

(n_socio, f_alta, d_estado) values (@n_socio, @f_alta, Activa

create trigger tg_A01_upd

on movimientos

for update

as

declare @c_clie form deleted)

if @c_clie<>(select c_clie form inserted

end

in select

Sistemas de datos − Página 2

C) Server/Back End (NT Service)

B) SQL Server Service (Engine)

- SQL Executive

Service

D) SQL Server Distributed Management Objects (DMO)

SQL Enterprise Manager.(Manejador de la empresa.)

Dispositivo

Archivo