Microsoft®
SQL Server®

GERARDO SOTO

# SQL
# Structured Query Language

Es un lenguaje de programación diseñado para almacenar, manipular y recuperar datos almacenados en bases de datos relacionales.

GERARDO SOTO

# Un poco de historia

- Los orígenes del SQL están ligados a los de las bases de datos relacionales.
- En 1970 Codd propone el modelo relacional y asociado a este; un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basado en estas ideas los laboratorios de IBM, definen el lenguaje SEQUEL (Structured English Query Language).
- Más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM.
- Fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.
- SEQUEL terminaría siendo el predecesor de SQL, siendo una versión evolucionada del primero.
- SQL pasa a ser el lenguaje por excelencia de los diversos SMBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI.
- Al año siguiente este estándar es también adoptado por la ISO.

# ¿QUE ES SQL SERVER?

Es un sistema para la gestión de base de datos basado en el sistema relacional

GERARDO SOTO

# CARACTERÍSTICAS DE SQL SERVER

- Soporta transacciones.
- Soporta procedimientos almacenados(Triggers).
- Incluye también un entorno gráfico.
- Permite trabajar en modo cliente - servidor.
- Permite administrar información de otros servidores de datos.

GERARDO SOTO

# DESVENTAJAS DE SQL SERVER

- Requiere de un sistema operativo basado en Windows por lo que no puede instalarse en uno como Linux.
- En versiones de 32 bits, SQL SERVER solo permite almacenar un máximo de 64 GB de memoria compartida y en sistemas de 64 bits, hasta 4 TB.

bigint

decimal

varchar

int

bit

timestamp

text

image

binary

xml

# Tipos de datos de SQL Server

real

char

money

tinyint

float

nvarchar

GERARDO SOTO

# TIPOS DE DATOS BÁSICOS

| | |
|---|---|
| Números exactos | Cadenas de caracteres Unicode |
| Númericos Aproximados | Cadenas Binarias |
| Fecha y Hora | Otros tipos de datos |
| Cadenas de caracteres | |

Para mayor detalle, click aquí

# TIPOS NÚMERICOS EXACTOS

- Este tipo de datos se usa para cantidades tanto enteras como decimales y de dinero que no requieren gran precisión.

| | |
|---|---|
| bigint | numeric |
| bit | smallint |
| decimal | smallmoney |
| int | tinyint |
| money | |

Para mayor detalle, click aquí

# TIPOS NÚMERICOS APROXIMADOS

- Este tipo de datos se usa para números que requieren mayor precisión que los exactos.

| | |
|---|---|
| float | real |

Para mayor detalle, click aquí

# TIPOS FECHA Y HORA

- Este tipo de datos son usados para referirse a fechas y hora con diferentes formatos, precisión y longitudes según se requiera.

| | |
|---|---|
| date | datetimeoffset |
| datetime2 | smalldatetime |
| datetime | time |

Para mayor detalle, click aquí

# TIPOS CADENA DE CARACTERES

- Este tipo de datos se emplean para almacenar cadenas de diferentes longitudes según cada tipo. Mientras que CHAR ofrece cadenas de longitud fija, podemos usar VARCHAR para limitar la longitud máxima.

| | |
|---|---|
| char | varchar |
| text | |

Para mayor detalle, click aquí

# TIPOS CADENA DE CARACTERES

- A diferencia de los anteriores donde cada carácter del tipo de dato VARCHAR se almacena en 8 bits (1 byte). Pero las cadenas NVARCHAR se almacenan en la base al estándar UTF-16 de 16 bits o dos bytes por carácter, eso quiere decir que con el tipo de dato NVARCHAR sólo podremos almacenar la mitad: 4000 caracteres.

| | |
|---|---|
| nchar | nvarchar |
| ntext | |

Para mayor detalle, click aquí

# TIPOS CADENA DE CARACTERES

- Con estos tipos de datos almacenamos cadenas de bytes de longitud fija o variable dada por el parametro, también tenemos la posibilidad de almacenar imágenes.

| | |
|---|---|
| binary | varbinary |
| image | |

Para mayor detalle, click aquí

# TIPOS CADENA DE CARACTERES

- Además de los tipos comunes, SQL Server nos da la posibilidad de almacenar dentro de las tablas otros tipos de datos más especificos como lo son las coordenadas geograficas, un sistema de coordenadas Euclidianas, entre otras.

| | |
|---|---|
| cursor | rowversion |
| hierarchyid | uniqueidentifier |
| sql_variant | xml |
| Spatial Geometry Types | Spatial Geography Types |
| table | |

Para mayor detalle, click aquí

# Comandos

Existen dos tipos decomandos SQL:

- DDL que permiten crear y definir nuevas bases de datos, campos e índices.

- DML que permiten generar consultas para ordenar, filtrar, extraer y actualizar datos de la base de datos.

- Para practicar vease el siguiente link

# DDL

- CREATE:
  Utilizado para crear nuevas tablas, campos e índices.
- DROP:
  Empleado para eliminar tablas e índices.
- ALTER:
  Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

# CREANDO BD EN SQL SERVER

```
CREATE DATABASE nombre_BD;
```

Por ejemplo:

```
CREATE DATABASE mi_Primera_BD;
```

GERARDO SOTO

# CREANDO ESQUEMAS EN SQL SERVER

```
CREATE SCHEMA nombre_Esquema;
```

Por ejemplo:

```
CREATE SCHEMA Accesos;
```

GERARDO SOTO

# CREANDO TABLAS

CREATE TABLE nombre_Tabla;

Por ejemplo:

CREATE TABLE Usuarios;

# CREANDO TABLAS EN SQL SERVER

La tabla debe ser definida con un nombre que la identifique y con el cual podremos acceder a ella.

```
CREATE TABLE Usuarios (
    nombre varchar(30),
    id_Usuario varchar(10)
);
```

Para nombres de tablas se puede utilizar cualquier carácter permitido para nombres de directorios donde el primero debe ser uno alfabético y no puede contener espacios.

# CREANDO TABLAS EN SQL SERVER

Para ver la estructura de una tabla usamos el procedimiento "sp_columns" junto al nombre de la tabla:

```
sp_columns usuarios;
```

```
...  COLUM_NAME   TYPE_NAME    LENGHT
     nombre       varchar      30
     clave        varchar      30
```

# ELIMINANDO BD EN SQL SERVER

```
DROP DATABASE nombre_BD;
```

Por ejemplo:

```
DROP DATABASE mi_Primera_BD;
```

O en versiones 2016 a actual,

```
DROP DATABASE IF EXISTS mi_Primera_BD;
```

# ELIMINANDO ESQUEMAS

```
DROP SCHEMA nombre_Esquema;
```

O en versiones 2016 a actual,

```
DROP SCHEMA IF EXISTS nombre_Esquema;
```

# ELIMINANDO TABLAS

```
DROP TABLE nombre_Tabla;
```

O en versiones 2016 a actual,

```
DROP TABLE IF EXISTS nombre_Tabla;
```

# ELIMINANDO TABLAS

Para eliminar una tabla dentro de la base de datos usamos “drop table” junto al nombre de la tabla a eliminar

```
DROP TABLE Usuarios;
```

si no existe..... aparece un mensaje de error

```
if object id(‘Usuarios') is not null
drop table usuarios;
```

Que nos indica que no es posible eliminar la tabla ya que contiene al menos una tupla.

# LLAVE PRIMARIA

Es un campo o varios que identifican una **única** tupla en una tabla.

```
CREATE TABLE nombre_Tabla
campo tipo,
.......
 PRIMARY KEY (nombre_Campo)
);
```

# LLAVE FORANEA

Es un campo o varios que hacen referencia a otra tabla. Es decir, nos sirve para expresar que ese conjunto de datos provienen de otra(s) tabla(s).

```
CREATE TABLE nombre_Tabla1
.......
campo1 tipo,
CONSTRAINT FK_nombre FOREIGN KEY (campo1)
REFERENCES Tabla2(CampoTabla2)
);
```

# COMANDO ALTER

- Esta instrucción modifica una base de datos o los archivos y grupos de archivos asociados con la base de datos.

- Agrega o elimina archivos y grupos de archivos de una base de datos, cambia los atributos de una base de datos o sus archivos y grupos de archivos, cambia la clasificación de la base de datos y establece las opciones de la base de datos.

- Los snapshots de la base de datos no se pueden modificar.

# ALTER EN SCHEMA

Para modificar un esquema:

```
ALTER SCHEMA nombre_Esquema TRANSFER Schema2.Tabla2;
```

Por ejemplo:

```
ALTER SCHEMA AsuntosEstudiantiles TRANSFER Académicos.Dirección;
```

Esto modifica el esquema AsuntosEstudiantiles transfiriendo Dirección del esquema Académicos.

# ALTER EN TABLE

La instrucción ALTER TABLE se usa para agregar, eliminar o modificar columnas en una tabla existente. También se usa para agregar y soltar varias restricciones en una tabla existente.

```
ALTER TABLE nombre_tabla
ADD nombre_columna tipo;
```

Por ejemplo:

```
ALTER TABLE Usuario ADD fecha_nacimiento;
```

Esto agrega una columna fecha_Nacimiento a la tabla Usuarios.

# ALTER EN LLAVE FORANEA

Para crear una restricción en una tabla ya creada.

```
ALTER TABLE nombre_Tabla1
ADD CONSTRAINT FK_nombre
FOREIGN KEY (campo1) REFERENCES Tabla2(CampoTabla2)
);
```

# ELIMINANDO LLAVES FORANEAS

```
ALTER TABLE nombre_Tabla1
DROP CONSTRAINT FK_campo;
```

GERARDO SOTO

# CAMPO CON ATRIBUTO IDENTITY

Un campo numérico puede tener un atributo extra “IDENTITY”. Los valores de un campo con este atributo genera valores secuenciales que se inician en 1 y se incrementan en 1 automáticamente.

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

# CAMPO CON ATRIBUTO IDENTITY

```
CREATE TABLE libros(
codigo IDENTITY,
titulo varchar(40) NOT NULL,
autor varchar(30),
editorial varchar(15),
precio float
);
```

Para ingresar registros se omite el campo definido como IDENTITY, por ej:

```
INSERT INTO libros (titulo,autor,editorial,precio)
values('El aleph','Borges','Emece',23);
```

# VALORES NULL

NULL significa dato desconocido o valor inexistente, pero no es lo mismo que un valor 0, una cadena vacía o una cadena literal.

# TRUNCATE TABLE

TRUNCATE TABLE libros;

La sentencia “TRUNCATE TABLE“vacía la tabla (elimina todos los registros) y conserva la estructura de la tabla.

La diferencia con “DROP TABLE” es que esta sentencia borra la tabla, “TRUNCATE TABLE” la vacía.

La diferencia con “DELETE” es la velocidad, es más rápido “TRUNCATE TABLE” que “DELETE” (se nota cuando la cantidad de registros es muy grande) ya que éste borra los registros uno a uno.

# DML

- SELECT:
  Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
- INSERT:
  Utilizado para cargar datos en la base de datos en una única operación.
- UPDATE:
  Utilizado para modificar los valores de los campos y registros especificados.
- DELETE:
  Utilizado para eliminar registros de una tabla de una base de datos.

GERARDO SOTO

# INSERTAR REGISTROS (INSERT)

```
INSERT INTO NOMBRETABLA (NOMBRECAMPO1, ..., NOMBRECAMPOn)
VALUES (VALORCAMPO1, ..., VALORCAMPOn);
```

Para agregar un registro a la tabla escribimos:

```
INSERT INTO usuarios (nombre, clave) VALUES ('Mariano' , 'Payaso');
```

# INSERTAR/RECUPERAR REGISTROS (SELECT)

Para ver los registros de una tabla usamos ”SELECT”

```
SELECT * from usuarios
```

El comando select recupera los registros de una tabla y con el * se indica que muestre todos los campos de la misma

GERARDO SOTO

# RECUPERAR REGISTROS (WHERE)

```
SELECT nombreCampo1, ..., nombre_CampoN
FROM nombre_Tabla
WHERE condición;
```

```
SELECT nombre, clave
FROM usuarios
WHERE nombre = 'Marcelo';
```

# OPERADORES RELACIONALES

- = igual
- <> distinto
- > mayor
- < menor
- >= mayor o igual
- <= menor o igual

# BORRAR REGISTROS EN SQL SERVER (DELETE)

Para eliminar los registros de una tabla usamos el comando “DELETE”:

```
DELETE FROM usuarios;
```

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

```
DELETE FROM usuarios
WHERE nombre='Marcelo';
```

# ACTUALIZAR REGISTROS EN SQL SERVER (UPDATE)

```
UPDATE usuarios SET clave='realmadrid';
UPDATE usuarios SET clave='boca'
WHERE nombre='FedericoLopez';
UPDATE usuarios SET nombre='MarceloDuarte', clave='Marce'
WHERE nombre='Marcelo';
```

# COMENTARIOS

Los comentarios son textos que no se ejecutan y para ello se emplea dos guiones al comienzo de la línea.

```
SELECT * FROM libros -- mostramos los registros de los libros.
```

para varias líneas de comentarios usamos /* al inicio y */ al final.

```
SELECT titulo, autor
/* mostramos titulos y nombres
de los autores */
FROM libros
```

# POLITICAS DE MANTENIMIENTO

Puede ocurrir que, para una determinada clave foránea, la política adecuada en caso de borrado sea diferente de la adecuada en caso de modificación.

Por ejemplo, puede ser necesario aplicar la restricción en caso de borrado y la actualización en cascada en caso de modificación.

El diseñador puede elegir para cada llave foránea qué política se aplicará en caso de borrado de la clave primaria referenciada.

# POLITICAS DE MANTENIMIENTO

Las principales políticas son:

NO ACTION:
Motor de base de datos genera un error y se revierte la acción de eliminación o actualización de la fila de la tabla primaria.

CASCADE:
Si esa fila se actualiza o elimina en la tabla primaria, las filas correspondientes se actualizan o eliminan en la tabla de referencia.CASCADE no se puede especificar si una columna **timestamp** es parte de una clave externa o de la clave con referencia. ON DELETE CASCADE no se puede especificar en una tabla que tenga un desencadenador INSTEAD OF DELETE. No se puede especificar ON UPDATE CASCADE para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

# POLITICAS DE MANTENIMIENTO

SET NULL:
Cuando se actualiza o elimina la fila correspondiente en la tabla primaria, todos los valores que componen la clave externa se establecen en NULL. Para que esta restricción se ejecute, las columnas de clave externa deben aceptar valores NULL. No se puede especificar para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

SET DEFAULT:
Todos los valores que forman la clave externa se establecen en los valores predeterminados si se actualiza o elimina la fila correspondiente de la tabla primaria. Para que esta restricción se ejecute, todas las columnas de clave externa deben tener definiciones predeterminadas. Si una columna acepta valores NULL y no se ha establecido un valor predeterminado explícito, NULL se convierte en el valor predeterminado explícito de dicha columna. No se puede especificar para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

# POLITICAS DE MANTENIMIENTO

ON DELETE CASCADE:
Especifica que dato hijo es borrado cuando el dato padre es borrado.

ON UPDATE:
Opcional, Especifica que hacer con el dato hijo cuando el dato padre es actualizado. Se tienen las opciones CASCADE, SET NULL, SET DEFAULT y NO ACTION.

CASCADE, SET NULL, SET DEFAULT y NO ACTION se pueden combinar en las tablas con relaciones referenciales entre sí. Si el Motor de base de datos detecta NO ACTION, detiene y revierte las acciones CASCADE, SET NULL y SET DEFAULT relacionadas.

Cuando una instrucción DELETE hace que se combinen las acciones CASCADE, SET NULL, SET DEFAULT y NO ACTION, todas las acciones CASCADE, SET NULL y SET DEFAULT se aplican antes de que el Motor de base de datos compruebe la existencia de NO ACTION.

# POLITICAS DE MANTENIMIENTO

El SQL nos ofrece la posibilidad de especificar, al definir una clave foránea, qué política queremos seguir.

```
CREATE TABLE nombre_tabla
    (definición_columna
    , definición_columna…
     , restricciones_tabla
     );
```

Donde una de las restricciones de tabla era la definición de claves foráneas, que tiene el siguiente formato:

```
FOREIGN KEY llave_foranea REFERENCES tabla [(llave_primaria)]
[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
```