Escuela de Sistemas y Tecnologías

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2013 – Materia:

Diseño e Implementación de BdeD

TEMA: Transact SQL

Agenda

- Introducción
- Características
- Tipos de Datos
- Lenguaje de Definición de Datos
- Lenguaje de Manipulación de Datos
- Procedimientos Almacenados
- Elementos de Transact SQL
- Transacciones

Introducción (1)

- **SQL** (*Structured Query Language*) es el lenguaje de consultas de base de datos declarativo más usado por los DBMS y estandarizado por ANSI/ISO.
- **Lenguaje declarativo:** indica qué se quiere obtener o qué se va a hacer, pero no cómo se tiene que hacer.
- **Opera sobre un conjunto de registros:** Sus operaciones se basan en la teoría de conjuntos, por lo cual cada comando opera sobre un conjunto de registros.

Introducción (2)

- **Se divide en tres sub lenguajes:**
  - **LDD** (Lenguaje de Definición de Datos): permite definir los objetos de la base de datos
  - **LMD** (Lenguaje de Manipulación de Datos): permite administrar los datos mantenidos en la base de datos
  - **LCD** (Lenguaje de Control de Datos): permiten definir los permiso sobre los objetos de la base de datos

Introducción (3)

- **Base de Datos:** Conjunto de datos que tienen una temática en común y están relacionados. Permite mantener un conjunto de tablas y objetos relacionados a estas tablas.
- **Objetos de Base de Datos:** Los objetos de la base de datos son diferentes elementos que nos permiten operar sus datos. Por ejemplo: Tablas, Indices, Vistas, Usuarios, etc.

Introducción (4)

- **Tabla:** Conjunto de datos que tienen una misma estructura. Está formada por un conjunto de campos y mantiene un conjunto de registros. Una tabla es similar a una relación en el Modelo Relacional.
- **Campos:** Determinan la estructura de la tabla. Los campos son similares a los atributos del MR.
- **Registro:** Cada registro representa un elemento de la vida real. Los registros son similares a las tuplas del MR.
- **Dato:** Es el valor asociado a un campo de un registro.

Características

- Un script es un conjunto de comandos que permiten la automatización de tareas.
- SQL no diferencia mayúsculas de minúsculas.
- El palabra *GO* no es un comando, sino una orden que indica a las utilidades de SQL Server el final de un lote de instrucciones T-SQL.
- Para colocar comentarios se utiliza
  - **--**
  - **/* */**

Tipos de datos

- Los tipos de datos más comunes son:
  - **Booleanos**: bit
  - **Enteros**: tinyint, int, samllint, bigint
  - **Reales** : Decimal, Money, Float
  - **Cadena de Texto básicas**: Char, VarChar(max)
  - **Cadena de Texto Unicode**: NChar, NVarChar(max)
  - **Fecha - Hora**: DateTime
  - **Otros**: Image, VarBinary(max)

LDD - Base de Datos

```
CREATE DATABASE  NomLogicoBD
```

```
CREATE DATABASE  NomLogicoBD
ON (
     NAME = NombreBD ,
     FILENAME='Unidad:\Nombre.mdf'
   )
```

```
DROP DATABASE  NomLogicoBD
```


LDD - Tablas (1)

```
CREATE   TABLE    NomTabla
(
      NomCampo   tipoDato
                 [Null / Not Null]
                 [Primary Key]
                 [Identity (X,X)]
                 [Foreign Key References  NomTabla(campo)]
                 [UNIQUE ],
      NomCampo2 …….. ,
       PrimaryKey (NomCampo, NomCampo2)
)
```


LDD - Tablas (2)

- Primero se tiene que poner en uso la base de datos en la cual se agregará el objeto, con el comando

  ```
  USE   NomLogicoBD
  ```

- Por cada campo de la tabla se deberá determinar el conjunto de restricciones que se le aplican :
  - **Tipo de dato:** es la primera restricción; es obligatoria.
  - **Not Null**: no acepta valores nulos. Sino se coloca nada, se asume que si los acepta (Null).
  - **Primary Key**: indica que el campo es PK.
  - **Identity(x,x)**: campo auto-numérico.
  - **Foreign Key**: vinculación con campo de otra tabla.
  - **UNIQUE**: no permite valores repetidos.

LDD - Tablas (3)

```
ALTER TABLE   NomTabla ADD nomCampo tipoDato
```

```
ALTER TABLE   NomTabla DROP NomCampo
```

```
DROP TABLE    NomTabla
```

LMD - Introducción

- Los comando *INSERT*, *UPDATE* y *DELETE* pueden genera errores por las restricciones impuestas por los comandos LDD.
- Los errores más comunes en los comandos *INSERT* y *UPDATE*:
  - Tipos de datos que no correspondan
  - Restricción NOT NULL
  - Colocar valores duplicados para campos únicos o clave primaria
  - Valor de clave foránea que no tenga un valor relacionado en el campo referenciado.
- Los errores más comunes en el comando *DELETE*:
  - Intentar eliminar una clave primaria que esta relacionada con una clave foránea

LMD – Agregar Registros

- Cuando se hace una inserción de registros en una tabla, deben coincidir los tipos de datos y la cantidad de valores con los campos indicados.
- Si no se coloca un campo en dicha sentencia, automáticamente se le asigna el valor **null**.
- Los campos auto-numéricos no pueden colocarse en este tipo de sentencias.

```
INSERT NomTabla  VALUES (v1,v2,…,vN)

INSERT NomTabla (Campo1,…, CampoN) VALUES (v1,...,vN)

INSERT NomTabla (CampoA,…,CampoN) SELECT v1,…,vN
```


LMD – Actualizar Registros

- Sin una condición, se actualizan ***todos*** los registros de la tabla

```
UPDATE   nomTabla
SET      nomCampo = valor

-------------------------------------------------------------------------------

UPDATE   nomTabla
SET      nomCampo = valor,  nomCampo = nomCampo * Valor

-------------------------------------------------------------------------------

UPDATE   nomTabla
SET      nomCampo = valor
WHERE    (condicion_filtro)
```


LMD – Eliminar Registros

- Sin una condición, se eliminan ***todos*** los registros de la tabla

```
DELETE   FROM  nomTabla

-------------------------------------------------------------

DELETE   FROM   nomTabla
WHERE           (condicion_filtro)
```

LMD – Consultar Registros (1)

```
SELECT   [ Distinct ] [ Top nº ]
         [ Tabla.Campo ]  [ Campo ]  [ Tabla.* ]  [ * ]

FROM     [ tabla1  join  tabla2
                 on   tabla1.Campo =  tabla2.Campo]

WHERE  [ not  / and  /  or ]
       [ CampoString  like   'Patron']
       [ Campo  between  valor1  and  valor2 ]
       [ Campo  is null ]
       [ Campo  in  (valor1, valor2....)  ]

GROUP BY  Campo  [,  Campo  ]

HAVING    ( condición  filtro  para grupo )

ORDER BY  Campo  [DESC / ASC]
```


LMD – Consultar Registros (2)

- Clausula ***SELECT***
  - Selecciona los campos cuyos valores se desplegarán en el resultado
  - Permite realizar operaciones que incluyan valores de campos
  - Modificador **Distinct**: determina que no se devuelvan valores duplicados para un campo
  - Modificador **TOP**: determina la cantidad de registros del resultado que se desplegarán
  - * - se recuperan todos los campo de una tabla, o de todas las tablas
- Clausula FROM
  - determina de cuales tablas se obtendrán los registros de la consulta

LMD – Consultar Registros (3)

- Clausula ***WHERE***
  - Especifica los filtros que se aplicaran a todos los registros de las tablas, que forman parte de la consulta
  - Permite realizar operaciones que incluyan valores de campos
  - Patrón ***LIKE***:
    - ***%*** indica de 0 a N lugares sin importar valor
    - _ significa un lugar obligatorio
    - ***[A-Z]*** rango valores para lugar obligatorio
    - ***^[A-M]*** negativa valores para lugar obligatorio
    - ***A*** valor fijo y obligatorio

LMD – Consultar Registros (4)

- Clausula ***GROUP BY***
  - Permite generar grupos con los registros
  - Luego de creado el grupo, el único valor accesible, es el valor que creo el grupo
  - Se pueden aplicar funciones agregadas a dichos grupos
  - ***Funciones Agregadas***:
    - ***Count***: cuenta cantidad de registros
    - ***Sum***: suma los valores de un campo en un grupo de registros
    - ***Avg***: promedia los valores de un campo en un grupo de registros
    - ***Min / Max***: determina el valor mínimo / máximo de un campo en un grupo de registros

LMD – Consultar Registros (5)

- Clausula ***HAVING***
  - Especifica los filtros que se aplicaran a cada grupo de registros, generados con la clausula ***Group by*** de la consulta
- Clausula ***ORDER BY***
  - Permite ordenar los resultados de la consulta en forma ascendente (por defecto) o descendente, en función de uno o varios campos.

LMD – Consultar Registros (6)

- Sub Consultas
  - El resultado de una consulta, se lo utiliza en otra consulta. Operadores para unir consultas:
    - **>= *all*** hallar máximo de un conjunto
    - **<= *all*** hallar mínimo de un conjunto
    - ***In*** permite hallar la intersección de dos conjuntos de valores
    - ***Not In*** realiza la resta del contenido del 1º conjunto de valores, menos el contenido del 2º conjunto de valores
  - ***Union*** unir el resultado de dos consultas (generar un único conjunto de registros)
  - ***Intersect*** obtener los registros que pertenezcan a los conjuntos de registros de dos consultas

Procedimientos Almacenados (1)

- Conjunto de sentencias SQL con nombre, previamente compiladas y almacenadas en la base de datos.
- Acepta parámetros de entrada/salida y permite devolver valores enteros (*return*).
- No permiten que los usuarios accedan directamente a las tablas, ocultando la estructura de la base de datos. Además, permiten que los usuarios tengan permisos para ejecutarlos, pero no acceso a las tablas.
- Cuando se generó el procedimiento, ya se realizó el plan de ejecución (una sola vez).
- Los usuarios realizan tareas complejas mediante el envío de una única sentencia: reduce el tráfico entre el servidor y el cliente.

Procedimientos Almacenados (2)

- Los pasos para procesar un SP son:
  - Creación:
    - Se valida la sintaxis de las instrucciones.
    - No se valida la existencia de los objetos o campos involucrados.
  - Primera Ejecución:
    - Analiza la existencia de los objetos y campos involucrados.
    - Se genera y guarda un plan de ejecución para optimizar la ejecución de las sentencias.

Procedimientos Almacenados (3)

```
CREATE PROC   NomSP   @NomPar   tipoDato [output]   As
BEGIN
        -- Cuerpo del SP
END
go

-------------------------------------------------------------------------------

EXEC   NomSP   ValorParam1, ……., ValorParamN
```

Procedimientos Almacenados (4)

- Parámetros de Salida:
  - Los SP pueden devolver información modificando en la ejecución los valores de un parámetro de salida (OUTPUT) .
- Sentencia **Return**:
  - Sólo devuelve valores enteros.
  - Interrumpe incondicionalmente la ejecución del SP.

Elementos T-SQL (1)

- *Variables*: Sirven para retener un valor individual de un tipo de dato específico. Comienzan siempre con el carácter @
  - Definición de variables:

    DECLARE @NomVar tipoDato
  - Asignación de valores:

    SET @NomVar = valor

@@ROWCOUNT
Cantidad de filas afectadas por la última instrucción.

@@IDENTITY
Último valor de auto numérico generado en la BD actual.

@@ERROR
Número de error de la última sentencia ejecutada. Valor 0 significa que no se produjeron errores.

Elementos T-SQL (2)

- ***Funciones del Servidor***

GETDATE()
Devuelve la fecha y hora del servidor

IDENT_CURRENT (NomTabla)
Devuelve el último valor auto numerado de la tabla que recibe como parámetro.

DATEDIFF (código, fechaInicio, fechaFinal)
Devuelve la diferencia en días , meses o años entre dos fechas (yy - mm - dd)

DATEADD (código, fecha, valor)
Devuelve una nueva fecha, habiendo sumado el valor correspondiente en días , meses o años (yy - mm - dd)

Elementos T-SQL (3)

- Sentencia IF:

```
IF (expresion)
   Begin
         -- sentencias
   End
ELSE
   Begin
         -- SENTENCIAS
   End
```

Transacciones (1)

- Una transacción es una secuencia de operaciones realizadas como una unidad única de trabajo, donde intervienen sentencias que modifican datos en una o más tablas de la base de datos.
- Si una transacción tiene éxito, ***todas*** las modificaciones de los datos realizadas durante la transacción se confirman y convierten en modificaciones permanentes.
- Si durante el transcurso de una transacción, alguna de las sentencias encuentra errores, se cancelan ***todas*** las modificaciones llevadas a cabo en la transacción.

Transacciones (2)

- **Atomicidad**: debe ser una única unidad de trabajo. Se deben realizar todas las modificaciones o ninguna.
- **Coherencia**: Cuando finaliza una transacción debe dejar todos los datos en un estado coherente.
- **Aislamiento**: Las modificaciones realizadas por transacciones simultaneas se deben aislar de modificaciones llevadas a cabo por otras transacciones. Una transacción reconoce los datos en el estado que estaban antes de que otra transacción simultanea los modificara o después de que la segunda transacción haya concluido. No reconoce un estado intermedio.
- **Durabilidad**: Una vez concluida una transacción sus efectos son permanentes en el sistema.

Transacciones (3)

- Las sentencias que se utilizan para programar una transacción son las siguientes:
  - **BEGIN TRANSACTION**: Marca el punto de inicio de una transacción.
  - **COMMIT TRANSACTION**: Marca el final correcto de una transacción. Hace que todas las sentencias de modificación efectuadas sobre los datos desde el comienzo de la transacción, se confirmen y sea permanentes en la BD.
  - **ROLLBACK TRANSACTION**: Marca un final incorrecto de una transacción. Aborta todos los cambios efectuados desde el comienzo de la transacción.


BIOS
FIN
Transact SQL