Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Bases de Datos

Guía de Trabajo Nro. 2

Tablas - SQL Data Manipulation Language

Msc. Lic. Hugo Minni
2014

Creación de tablas

- Creamos una tabla con la sentencia ANSI SQL `CREATE TABLE`:

```
CREATE TABLE cliente
  (
   codCli     int           NOT NULL,
   ape        varchar(30)   NOT NULL,
   nom        varchar(30)   NOT NULL,
   dir        varchar(40)   NOT NULL,
   codPost    char(9)       NULL
  )
```

- Cada fila posee la *definición de una columna* de la tabla:

  - El **nombre de la columna** puede poseer hasta 30 caracteres y no puede contener ni espacios ni signos de puntuación.

  - El **tipo de dato** puede ser cualquiera de los tipos que soporte el RDBMS.

  - El **status de null** indica como se comportará la columna frente a los valores null. Una cláusula `NULL` indica que la columna admitirá valores nulos. Un cláusula `NOT NULL` indica que la columna no admitirá valores nulos.

    En T-SQL se puede especificar una tercer alternativa para el status de null: `IDENTITY`. Especificamos `INDENTITY` cuando queremos que SQL Server proporcione para la columna un único valor ascendente cada vez que se inserte una nueva fila. A este tipo de columnas se les denomina **autoincrementales**.

Columnas autoincrementales

- Son las columnas que definimos con la cláusula `IDENTITY`[1].

- Solo las columnas de tipo `integer`, `smallint`, `tinyint`, `decimal` o `numeric` pueden ser establecidas como autoincrementales.

[1] Las columnas con status de identity no pueden utilizarse para proporcionar números de fila absolutos. Es decir, si una columna identity posee el valor 130 esto no garantiza que se trate de la fila 130 de la tabla. Esto se debe a que SQL Server reserva los valores identity antes de confirmar el éxito o fracaso de una operación `INSERT`. O sea, un valor identity asignado para un `INSERT` que ha fracasado no vuelve a ser utilizado.

- La cláusula `IDENTITY` posee dos parámetros opcionales: `SEED` y `STEP`.
  `SEED` es el valor inicial que recibirá la primer fila insertada. Su valor por omisión es `1`. `STEP` es el valor que tendrá el incremento entre filas consecutivas. Su valor por omisión también es `1`.
  En el siguiente ejemplo la tabla `proveed` es definida con un identificador de proveedor con status de `IDENTITY`:

```
CREATE TABLE proveed
  (
   codProv    int           IDENTITY(1,1),
   razonSoc   varchar(30)   NOT NULL,
   dir        varchar(30)   NOT NULL,
  )
```

Valores por omisión

- Podemos establecer valores por omisión para una columna a través de la constraint `DEFAULT`. La constraint `DEFAULT` especifica un valor que la nueva fila deberá asumir cuando no se indique un valor explícito en la *sentencia DML*[2] *de inserción*:

```
CREATE TABLE cliente
  (
   codCli     int           NOT NULL,
   ape        varchar(30)   NOT NULL,
   nom        varchar(30)   NOT NULL,
   dir        varchar(40)   NOT NULL,
   codPost    char(9)       NULL DEFAULT 3000
  )
```

1. A continuación definiremos un pequeño esquema de tablas para realizar ejercicios de manipulación de datos. En el *Analizador de consultas SQL* seleccione la base de datos pubs y ejecute las siguientes sentencias SQL:

```
CREATE TABLE cliente
  (
   codCli     int           NOT NULL,
   ape        varchar(30)   NOT NULL,
   nom        varchar(30)   NOT NULL,
   dir        varchar(40)   NOT NULL,
   codPost    char(9)       NULL DEFAULT 3000
  )
```

[2] Data Manipulation Language.

```
CREATE TABLE productos
  (
   codProd    int           NOT NULL,
   descr      varchar(30)   NOT NULL,
   precUnit   money         NOT NULL,
   stock      smallint      NOT NULL,
  )


CREATE TABLE pedidos
  (
   numPed     int           NOT NULL,
   fechPed    datetime      NOT NULL,
   codCli     int           NOT NULL
  )



CREATE TABLE detalle
  (
   codDetalle       int           NOT NULL,
   numPed           int           NOT NULL,
   codProd          int           NOT NULL,
   cant             int           NOT NULL,
   precioTot        float         NULL
  )



CREATE TABLE proveed
  (
   codProv    int           IDENTITY(1,1),
   razonSoc   varchar(30)   NOT NULL,
   dir        varchar(30)   NOT NULL
  )
```

Inserción de filas

- Insertamos filas en una tabla a través de la sentencia ANSI SQL `INSERT`. `INSERT` posee la siguiente sintaxis:

```
INSERT [INTO] <tabla>
   [ (<columna1>, <columna2> [, <columnan> ...] ) ]
   VALUES ( <dato1> [, <dato2>...] )
```

- Si vamos a proporcionar datos para todas las columnas podemos omitir la lista de las mismas:

```
INSERT [INTO] <tabla>
   VALUES ( <dato1> [, <dato2>...] )
```

- Los datos de tipo `char` o `varchar` se especifican entre comillas simples. Los valores de tipo `float` se especifican con un punto decimal. (Por ejemplo: `243.2`). Las fechas se especifican con el formato `mm/dd/aaaa` entre comillas simples (Por ejemplo: `'11/30/2007'`).

2. Inserte en la tabla `cliente` el siguiente lote de datos: `1, 'LOPEZ', 'JOSE MARIA', 'Gral. Paz 3124'`. Permita que el código postal asuma el valor por omisión previsto. Verifique los datos insertados.

3. Inserte en la tabla cliente el siguiente lote de datos: `2, 'GERVASOLI', 'MAURO', 'San Luis 472'` pero evite que la fila asuma el valor por omisión para el código postal. Verifique los datos insertados.

4. Inserte en la tabla `proveed` dos proveedores: `'FLUKE INGENIERIA', 'RUTA 9 Km. 80'` y `'PVD PATCHES', 'Pinar de Rocha 1154'`. Verifique los datos insertados.

5. Defina una tabla de ventas (`Ventas`) que contenga:
-Un código de venta de tipo entero (`codVent`) autoincremental.
-La fecha de carga de la venta (`fechaVent`) no nulo con la fecha actual como valor por omisión.
-El nombre del usuario de la base de datos que cargó la venta (`usuarioDB`) no nulo con el *database user actual* como valor por omisión.
-La cantidad vendida (`cant`) que admita nulos.

6. Inserte dos ventas de `$100` y `$200` respectivamente. No proporcione ninguna información adicional. Verifique los datos insertados.

Otras formas de inserción de filas

- Podemos crear una nueva tabla e insertar a la vez filas de una existente usando la sentencia `SELECT` con la cláusula `INTO`[3]:

```
SELECT <lista de columnas>
   INTO <tabla-nueva>
   FROM <tabla-existente>
   WHERE <condicion>
```

- Una variante de la sentencia `INSERT` permite que insertemos en una tabla los datos de salida de una sentencia `SELECT`. La tabla sobre la que vamos a insertar debe existir previamente:

```
INSERT <tabla-destino>
   SELECT *
      FROM <tabla-origen>
      WHERE <condicion>
```

7. Cree una tabla llamada `clistafe` a partir de los datos de la tabla `cliente` para el código postal `3000`. Verifique los datos de la nueva tabla.

8. Inserte en la tabla `clistafe` la totalidad de las filas de la tabla `cliente`. Verifique los datos insertados.

[3] `SELECT INTO` es una operación que no se registra en el *transaction log*.

Modificación de datos

- Modificamos los datos de una o varias filas a través de la sentencia ANSI SQL `UPDATE`:

```
UPDATE <tabla>
   SET <col> = <nuevo valor-o-expres> [,<col> = <nuevo-valor-o-expres>...]
   WHERE <condición>
```

Si omitimos la cláusula `WHERE`, todas las filas de la tabla resultan modificadas. Los cambios son irreversibles.

9. En la tabla `cliente`, modifique el dato de domicilio. Para todas las columnas que incluyan el texto `'1'` reemplace el mismo por `'The Crystal Method 168'`.

Update a valores por omisión

T-SQL permite que, en una sentencia `UPDATE` establezcamos el valor de una columna al valor por omisión especificado en su definición (constraint `DEFAULT`):

```
UPDATE <tabla>
   SET <columna> = DEFAULT
```

10. Establezca el valor de la columna `CodPost` de la tabla `cliente` a su valor por omisión para todas las filas de la misma.

Eliminación de filas

- Eliminamos una o varias filas a través de la sentencia ANSI SQL `DELETE`:

```
DELETE <tabla>
   WHERE <condicion>
```

Si omitimos la cláusula `WHERE`, todas las filas de la tabla resultan eliminadas.

11. Elimine todos las filas de la tabla diStaFe cuyo código postal sea nulo.

Truncar tablas

Podemos eliminar todas las filas de una tabla conservando su estructura a través de la sentencia:

```
TRUNCATE TABLE <tabla>
```

`TRUNCATE TABLE` inicializa los valores de las columnas con status de `IDENTITY` y no se registra en el *transaction log*[4].

Eliminar tablas

Eliminamos una tabla a través del comando ANSI SQL `DROP TABLE`. Por ejemplo, para eliminar la tabla `cliente`:

```
DROP TABLE cliente
```

Debemos tener en cuenta que si eliminamos una tabla, la única manera de recuperar sus datos es a partir de una copia de backup que hayamos realizado previamente.

[4] Podemos obtener un resultado similar disparando una sentencia `DELETE` sin cláusula `WHERE`, pero esta modalidad no inicializa los valores de las columnas con status de `IDENTITY` y registra la operación en el *transaction log*.

Tablas temporales

- Las tablas temporales son tablas regulares como las que hemos visto, pero existen por un lapso de tiempo determinado. Se crean como cualquier otra tabla, pero anteponiendo a su nombre el símbolo #:

```
CREATE TABLE #clienteTemp
  (
   codCli     int           NOT NULL,
   ape        varchar(30)   NOT NULL,
   nom        varchar(30)   NOT NULL,
   dir        varchar(40)   NOT NULL,
   codPost    char(9)       NULL
  )


SELECT * FROM #clienteTemp
```

- Estas tablas se denominan **tablas temporales locales**. Existen mientras existe la conexión cliente/servidor que las creó y sólo ésta conexión tiene permisos para accederlas. Son creadas implícitamente en la base de datos del sistema `tempdb`, y no es posible establecer permisos sobre las mismas.

- SQL Server también permite crear **tablas temporales globales**, que también existen mientras existe la conexión cliente/servidor que las creó pero -a diferencia de las locales- son accesibles por todos los usuarios del sistema. Se crean igual que una tabla temporal local pero anteponiendo a su nombre dos símbolos #.

12. Cree una tabla temporal local llamada `Tempi` con un par de columnas: `Codcli int NOT NULL` y `Ape varchar(30) NOT NULL`.

13. Utilice `SELECT...INTO` para crear una copia temporal de los autores del estado de California (`CA`) de la tabla `authors`, pero solo con `apellido`, `nombre` `domicilio` y `ciudad`. Verifique los datos en la tabla temporal creada.