Unidad 10 – Vistas, tablas temporales

- Vistas
  - Creación
  - Actualizaciones
    - Columnas NOT NULL
    - Restricciones
    - Cálculos y agregados
    - Filas fantasma
  - Modificar una vista
- Tablas temporales
  - Locales
  - Globales
  - Variables tipo tabla
  - Diferencias con las vistas
- Funciones inline

Vistas

- Una vista es una tabla virtual
  - Procede de una instrucción SELECT
  - No ocupa espacio
  - Permite filtrar los datos y definir reglas de acceso
  - No puede definirse una ordenación
  - Simplifican las consultas y evitan errores

- Creación

```
CREATE VIEW V_Concursantes AS
  SELECT di, tdi, tipo_artista, nombre, apellidos, rep_di,
  nombre_artistico, modalidad, fecha_nacimiento, posicion FROM
  BF_Artistas JOIN BF_Concursantes
  ON di=art_di
```

¿Qué se puede hacer con vistas?

- Hacer SELECT y crear otras vistas.
- Definir desencadenadores (TRIGGERS) del tipo INSTEAD OF
- Insertar, actualizar y borrar filas
  - La actualización tiene ciertas limitaciones

¿Qué no se puede?

- No se puede actualizar si se define con TOP, UNION, GROUP BY o DISTINCT
- En vistas sobre más de una tabla sólo se permiten INSERT y UPDATE si afectan sólo a una de ellas. No se permite DELETE
- Para insertar filas deben estar en la vista todas las columnas NOT NULL sin DEFAULT
- Deben respetarse todas las restricciones
- No pueden añadirse ni modificarse columnas que sean el resultado de un cálculo
- Sí pueden insertarse filas que no estén en la vista (WITH CHECK OPTION)

Modificar una vista

```
ALTER VIEW V_Bailaores AS
   SELECT ID, NombreArtistico, Telefono FROM BF_Artistas
     WHERE tipo_artista LIKE 'Bailaor%'
```

- Es muy parecida a la creación
- Cuidado con cambiar el orden de las columnas (problemas con los permisos)
- Debe existir previamente

Tablas temporales

- Se definen igual que una tabla normal
  - ```
    CREATE TABLE #Guitarristas (
    di Varchar (9) Not Null)
    ```
- Suelen crearse a partir de los resultados de una sentencia SELECT
  - ```
    SELECT di INTO #Cantaores
    From BF_Artistas
    Where tipo_artista ='Cantaor'
    ```
- Sí ocupan espacio
- No están sincronizadas con los datos de origen
- Se crean en una base de datos del sistema llamada *tempdb,* por lo que no pueden repetirse los nombres (usar prefijos BD)

Tablas locales y globales

- Locales
  - Su nombre comienza por simple #
  - Sólo son accesibles por la conexión que las crea
  - Desaparecen cuando se cierra la conexión o se borran con DROP TABLE
  - No se recomienda su uso
- Globales
  - Su nombre comienza por ##
  - Son accesibles por cualquier conexión.
  - Desaparecen cuando se para el servidor o se borran con DROP TABLE
  - Se usan como forma de compartir datos entre distintos procesos, aunque ha de tenerse mucho cuidado, como cualquier objeto global

Variables tipo tabla

- Deben usarse en lugar de las tablas temporales locales
  - ```
    DECLARE @Bailaores AS TABLE (
    di Varchar(9) Not Null)
    ```
- No pueden definirse restricciones FK sobre tablas temporales ni variables tipo tabla ni referirse a ellas en FK definidas en otra tabla
    ```

Limitaciones de variables tipo Tabla

- Una columna no puede definirse de tipo TABLE
- No pueden pasarse como parámetros
- No pueden crearse con SELECT INTO
- No pueden definirse índices
- Su tiempo de vida está circunscrito al lote donde se declaran (de GO a GO)

Limitaciones de variables tipo Tabla

- No forman parte de las transacciones (no registradas).
- Las variables se almacenan en memoria RAM
- Su uso abusivo puede saturar un servidor (multiusuario)

Funciones inline

Devuelven un tipo de datos **TABLE**
La cláusula **RETURN** contiene una sola instrucción **SELECT** entre paréntesis
El conjunto de resultados de la instrucción **SELECT** forma la tabla que devuelve la función
Pueden utilizarse para obtener la funcionalidad de vistas parametrizadas

Funciones inline

CREATE FUNCTION *NombreFunción* (*lista parámetros*)

RETURNS TABLE AS

RETURN (SELECT …)