Explotación y administración de Base de datos

Juan Carlos Otaegui
jotaegui@unlam.edu.ar

SQL – DDL

- Lenguaje de definición de datos.
  - Modifica la estructura de los objetos de base de datos.
- Las cuatro operaciones más importantes:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

SCHEMA

- Similar a los directorios-catalogos en los sistemas de archivos.
- Al nivel superior de la jerarquía se la denomina Catalogo.

  Ej. catalogo5.esquema-banco.cuenta
- Se pueden configurar esquemas por defecto en la conexión para no indicarlo en cada sentencia.
- También evita los conflictos por nombres de objetos.

CREATE

- Nombre que identifique el contenido.
- No puede contener palabras reservadas.
- Tiene un límite de cantidad de caracteres: Ejemplo: Oracle 30 caracteres.
- En el caso de las vistas/tablas, las columnas también deben ser auto descriptivos.
- Se pueden indicar restricciones (Constraint).
- Se pueden dar indicaciones respecto a donde (recurso físico) se aloja.

CREATE TABLE

```
--Disk-Based CREATE TABLE Syntax
CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ] table_name
    [ AS FileTable ]
    ( { <column_definition> | <computed_column_definition>
        | <column_set_definition> | [ <table_constraint> ]
| [ <table_index> ] [ ,...n ] } )
    [ ON { partition_scheme_name ( partition_column_name ) | filegroup
        | "default" } ]
    [ { TEXTIMAGE_ON { filegroup | "default" } ]
    [ FILESTREAM_ON { partition_scheme_name | filegroup
        | "default" } ]
    [ WITH ( <table_option> [ ,...n ] ) ]
[ ; ]

<column_definition> ::=
column_name <data_type>
    [ FILESTREAM ]
    [ COLLATE collation_name ]
    [ SPARSE ]
    [ NULL | NOT NULL ]
    [
        [ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
      | [ IDENTITY [ ( seed,increment ) ] [ NOT FOR REPLICATION ]
    ]
```

CREATE INDEX

- Mejora la performance de las operaciones.
  - Se guarda elemento y algún indicador de posición.
  - Se realiza sobre una o más columnas.
- Requiere espacio de disco extra pero es mucho menor que el del total de la tabla.

CREATE INDEX

```
-- SQL Server Syntax

CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WHERE <filter_predicate> ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
         | filegroup_name
         | default
         }
    ]
```

Ejemplos:

```
create index I_libros_autoreditorial
on libros(autor,editorial);
```

CREATE VIEW

- Las vistas son tablas virtuales.
- Son creadas a partir de una consulta SELECT SQL.
- La consulta es guardada en la METADATA.
- Existen variantes intermedias como las vistas materializadas.
- Los permisos para acceder a una vista son similares a los que se utilizan con las tablas

CREATE VIEW

```
CREATE VIEW CLIENTE_VTA  AS
SELECT C.NOMBRE, SUM(V.MONTO) MONTO
FROM CLIENTE C, VENTA V
WHERE C.CLIENTE = V.CLIENTE
GROUP BY C.NOMBRE;


CREATE VIEW CLIENTE_MKT  AS
SELECT *
FROM CLIENTE C
WHERE DPTO = ‘MARKETING’;
```

ALTER

- Se utiliza para modificar objetos existentes.
- Muchos objetos o características de objetos no pueden ser modificadas. Ej. Tablas particionadas en Teradata.
- Ej. SQL Server no permite modificar:
  - Campos de tipo text, image, ntext y timestamp.
  - Un campo que es usado en un campo calculado

ALTER TABLE

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name . ] table_name
{
    ALTER COLUMN column_name
    {
        [ type_schema_name. ] type_name
            [ (
                {
                    precision [ , scale ]
                  | max
                  | xml_schema_collection
                }
            ) ]
        [ COLLATE collation_name ]
        [ NULL | NOT NULL ] [ SPARSE ]
      | {ADD | DROP }
        { ROWGUIDCOL | PERSISTED | NOT FOR REPLICATION | SPARSE }
    }
        | [ WITH { CHECK | NOCHECK } ]

    | ADD
    {
        <column_definition>
      | <computed_column_definition>
      | <table_constraint>
      | <column_set_definition>
    } [ ,...n ]

    | DROP
     {
         [ CONSTRAINT ]
         {
              constraint_name
              [ WITH
               ( <drop_clustered_constraint_option> [ ,...n ] )
              ]
          } [ ,...n ]
          | COLUMN
          {
              column_name
          } [ ,...n ]
     } [ ,...n ]
    | [ WITH { CHECK | NOCHECK } ] { CHECK | NOCHECK } CONSTRAINT
        { ALL | constraint_name [ ,...n ] }

    | { ENABLE | DISABLE } TRIGGER
        { ALL | trigger_name [ ,...n ] }

    | { ENABLE | DISABLE } CHANGE_TRACKING
        [ WITH ( TRACK_COLUMNS_UPDATED = { ON | OFF } ) ]
```

```
Alter table NombreTabla
Alter column CAMPO
NuevaDefinicion;

alter table libros
  alter column titulo
varchar(40) not null;
```

ALTER INDEX

- En SQL Server Reorganizar un índice puede ser útil cuando existe fragmentación.

```
-- SQL Server Syntax

ALTER INDEX { index_name | ALL }
    ON <object>
    { REBUILD
        [ PARTITION = ALL ]
        [ WITH ( <rebuild_index_option> [ ,...n ] ) ]
        | [ PARTITION = partition_number
            [ WITH ( <single_partition_rebuild_index_option> ) [ ,...n ] ]
          ]
    | DISABLE
    | REORGANIZE
        [ PARTITION = partition_number ]
        [ WITH ( LOB_COMPACTION = { ON | OFF } ) ]
    | SET ( <set_index_option> [ ,...n ] )
    }
[ ; ]

<object> ::=
{
    [ database_name. [ schema_name ] . | schema_name. ]
    table_or_view_name
}
```

```
ALTER INDEX IX_Employee_Org ON
HumanResources.Employee REORGANIZE ;
```

```
ALTER INDEX ALL ON Production.Product REBUILD
WITH (FILLFACTOR = 80, SORT_IN_TEMPDB = ON,
STATISTICS_NORECOMPUTE = ON);
```

ALTER VIEW / REPLACE

```
REPLACE VIEW CLIENTE_VTA  AS
SELECT C.NOMBRE, SUM(V.MONTO) MONTO
FROM CLIENTE C, VENTA V
WHERE C.CLIENTE = V.CLIENTE
AND DPTO <> 'MARKETING'
GROUP BY C.NOMBRE;


REPLACE VIEW CLIENTE_MKT  AS
SELECT NOMBRE, APELLIDO, DIRECCION
FROM CLIENTE C
WHERE DPTO = 'MARKETING';
```

DROP

- Se utiliza para eliminar objetos.
- La diferencia entre TRUNCATE y DROP es que TRUNCATE elimina todo rastro del contenido de la tabla.

- Ejemplos:
- DROP TABLE NOMBRETABLA;
- DROP VIEW NOMBREVISTA;
- DROP INDEX NOMBREINDICE;

TRIGGERS

- Se asocian directamente a tablas.
- Se trata de un procedimiento almacenado que se ejecuta cuando a la tabla asociada se le realiza una actualización con las clausulas:
  - UPDATE
  - DELETE
  - INSERT

Ejemplo TRIGGERS

```
create trigger Ajustar_Stock
  on Ventas_Diarias
  for delete
 as
   update stock_productos set stock=
stock_productos.stock+deleted.cantidad
   from stock_productos
   join deleted
   on
deleted.id_producto=stock_productos.id_p
roducto;
```

Ejemplo TRIGGERS

```
create trigger Verificar_Stock
  on Ventas_Diarias
  for insert
  as
   declare @stock int
   select @stock= stock from stock_productos
                  join inserted
                  on inserted.id_producto=stock_productos.id_producto

  if (@stock>=(select cantidad from inserted))
    update stock_productos set stock=stock-inserted.cantidad
     from stock_productos
     join inserted
     on inserted.id_producto=stock_productos.id_producto


  else
  begin raiserror ('Hay menos libros en stock de los solicitados para la
venta', 16, 1)
    rollback transaction
  end
```

Ejercicio

- Generar un script SQL:
  - Cree tablas
  - Cree al menos 1 índice para cada tabla
  - Cree al menos 1 vista para cada tabla
  - Cree un Trigger para update, delete e insert sobre una de las tablas creadas en los puntos anteriores.