TEMA 4. LENGUAJE DE CONSULTA ESTRUCTURADO (SQL).

1. QUÉ ES SQL
2. CÓMO SE USA SQL
3. PARA QUÉ SIRVE SQL
4. TIPOS DE SENTENCIAS
    - 4.1. LENGUAJE DE MANIPULACIÓN DE DATOS (LMD)
    - 4.2. LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

4.2 LENGUAJE DE DEFINICIÓN DATOS (LDD)

Las sentencias **CREATE y DROP** permiten definir nuevos objetos o destruir los existentes.

La sentencia **ALTER** permite modificar la estructura de los objetos creados.

TIPOS DE DATOS

SQL estándar maneja los siguientes tipos de datos:

Datos Numéricos

- INTEGER. Entero binario de palabra completa.
- SMALLINT. Entero binario de media palabra.
- DECIMAL(p,q). Número decimal de p dígitos y signo con q dígitos a la derecha del punto decimal.
- FLOAT(p). Número de punto flotante con precisión de p.

Datos de Cadena

- CHARACTER(n). Cadena de longitud fija con n caracteres de 8 bits
- VARCHAR(n). Cadena de longitud variable con hasta n caracteres de 8 bits

Datos de Fecha y Hora

- DATE. Fecha
- TIME. Hora
- DATETIME . Marca de tiempo. Combinación de fecha y hora.

(Ver en el enlace Manual SQL w3schools: SQL Data Types, y en enlace Manual de Referencia de Transact-SQL: Tipos de datos)

SENTENCIA CREATE TABLE

Permite crear una tabla. Formato:

CREATE TABLE tabla (col1 tipo_de_datos [NOT NULL] [RESTRICCIONES],.)

Si no ponemos NOT NULL col1 admite nulos, es decir col1 por defecto admite nulos a menos que pongamos NOT NULL.

- Uso de IDENTITY **:** Indica que la nueva columna es una columna de identidad. Cuando se agrega una nueva fila a la tabla, el Database Engine (Motor de base de datos) proporciona un valor incremental único para la columna. Las columnas de identidad se utilizan normalmente con las restricciones PRIMARY KEY como identificadores de fila únicos de la tabla. La propiedad IDENTITY se puede asignar a las columnas tinyint, smallint, int, bigint, decimal(p,0) o numeric(p,0). **Sólo se puede crear una columna de identidad para cada tabla**. Las **restricciones DEFAULT no se pueden utilizar en las columnas de identidad**. Deben especificarse el valor de inicialización (seed) y el incremento (increment), o ninguno de estoS valores. Si no se especifica ninguno, el valor predeterminado es (1,1).

seed

Es el valor que se utiliza para la primera fila cargada en la tabla.

increment

Es el valor incremental que se agrega al valor de identidad de la fila cargada anterior.

Tipos de datos numéricos exactos que utilizan datos enteros:

Tipo de datos	Intervalo	Almacenamiento
bigint	De -2^63 (-9.223.372.036.854.775.808) a 2^63-1 (9.223.372.036.854.775.807)	8 bytes
int	De -2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647)	4 bytes
smallint	De -2^15 (-32.768) a 2^15-1 (32.767)	2 bytes
tinyint	De 0 a 255	1 byte

- # RESTRICCIÓN PRIMARY KEY

EmployeeID int PRIMARY KEY CLUSTERED

Los siguientes ejemplos de restricción de PRIMARY KEY se encuentran en "EjemplosRestricciónPrimaryKey.SQL". El archivo pertenece a EJEMPLOS_LDD.

```
--EJEMPLOS DE CREACIÓN DE TABLAS.
--RESTRICCIÓN PRIMARY KEY y uso de IDENTITY.

CREATE DATABASE pruebas1;
GO
USE pruebas1;

--Tabla sin PRIMARY KEY. No se suele hacer.
CREATE TABLE tabla1
(c1 INT,
 c2 CHAR(2)
 );

/*Tabla con creación de la PRIMARY KEY en la definición del propio
campo.*/
CREATE TABLE tabla2
(c1 INT PRIMARY KEY,
 c2 CHAR(2)
 );
```

```sql
--Tabla con IDENTITY.
CREATE TABLE tabla3
(c1 INT PRIMARY KEY,
 c2 INT IDENTITY(100,1)
 );

/*Añadir datos a un campo IDENTITY:
  No se pone ni el nombre de la columna, ni el valor.
*/
INSERT INTO tabla3 (c1)
VALUES (1),(2);

/*Si introduces los valores en todos los campos  (excepto
en el campo identity), puedes no poner nombre a las columnas*/
INSERT INTO tabla3
VALUES (3),(4);

--Esto da error.
INSERT INTO tabla3 (c1,c2)
VALUES (5,100),(6,101);

SELECT * FROM tabla3;

----Tabla con IDENTITY en la PRIMARY KEY.
CREATE TABLE tabla4
(c1 INT IDENTITY PRIMARY KEY,
 c2 CHAR(1) NOT NULL
 );

INSERT INTO tabla4 (c2)
VALUES ('a'),('b');

INSERT INTO tabla4
VALUES ('c'),('d');

SELECT * FROM tabla4;

/*El inconveniente de una columna IDENTITY es que si borramos
la fila, por ejemplo las filas 2 y 3, los valores del campo
IDENTITY no se reasignan.
*/
DELETE FROM tabla4
WHERE c1>1 AND c1<4;

SELECT * FROM tabla4;
```

```
INSERT INTO tabla4
VALUES ('e'),('f');

SELECT * FROM tabla4;

/*ACTIVAR Y DESACTIVAR IDENTITY para solucionar el problema
anterior.*/

--DESACTIVAR IDENTITY para poder insertar valores en dicho campo.
SET IDENTITY_INSERT tabla4 ON;

SELECT * FROM tabla4;

INSERT INTO tabla4 (c1,c2)
VALUES (2,'b'),(3,'c');

--ACTIVAR IDENTITY para que se autoincremente.
SET IDENTITY_INSERT tabla4 OFF;

/*Tabla con creación de la PRIMARY KEY después de declarar
todos los campos.*/
CREATE TABLE tabla5
(c1 INT,
 c2 INT,
 c3 VARCHAR(10),
 PRIMARY KEY (c1)
 );

INSERT INTO tabla5
VALUES (1,10,'aa'),(2,20,'bb');

SELECT * FROM tabla5;

/*Cuando la PRIMARY KEY es compuesta, ES OBLIGATORIO
  hacer la declaración de esta después de declarar
  todas las columnas.*/
  CREATE TABLE tabla6
  (c1 INT,
   c2 INT,
   c3 VARCHAR(10),
   PRIMARY KEY (c1,c2)
   );
```

```
INSERT INTO tabla6
VALUES (1,1,'aa'), (1,2,'bb');

SELECT * FROM tabla6;

--RESTRICCIÓN PRIMARY KEY eligiendo el nombre.
--Los nombres para las restricciones SON ÚNICOS en la base de
datos.

CREATE TABLE tabla7
(c1 INT CONSTRAINT res1 PRIMARY KEY,
 c2 INT,
 c3 VARCHAR(10)
 );

CREATE TABLE tabla8
(c1 INT,
 c2 INT,
 c3 VARCHAR(10),
 CONSTRAINT res2 PRIMARY KEY (c1)
 );

CREATE TABLE tabla9
(c1 INT,
 c2 INT,
 c3 VARCHAR(10),
 CONSTRAINT res3 PRIMARY KEY (c1,c2)
 );
```

∘ RESTRICCIÓN FOREIGN KEY

Cuando definimos en el propio campo:

SalesPersonID INT NULL REFERENCES SalesPerson(SalesPersonID)

O bien,

SalesPersonID INT NULL FOREIGN KEY REFERENCES
SalesPerson(SalesPersonID)

Cuando ya hemos terminado de definir todos los campos:

FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID)

Poniendo nombre a la restricción:

CONSTRAINT FK_SpecialOfferProduct_SalesOrderDetail
FOREIGN KEY (ProductID, SpecialOfferID)
REFERENCES SpecialOfferProduct (ProductID, SpecialOfferID)

Antes de ver los ejemplos debemos saber que cuando tenemos dos tablas relacionadas, se le llama TABLA OBJETIVO a la tabla que contiene la PRIMARY KEY a la que hace referencia la FOREIGN KEY. A la tabla que contiene la FOREIGN KEY se le llama TABLA REFERENCIAL.

Los siguientes ejemplos de restricción de FOREIGN KEY se encuentran en "EjemplosRestricciónForeignKey.SQL". El archivo pertenece a EJEMPLOS_LDD.

```
--EJEMPLOS DE CREACIÓN DE TABLAS.
--RESTRICCIÓN FOREIGN KEY.

CREATE DATABASE pruebas2;
GO
USE pruebas2;

--Creación de la TABLA OBJETIVO.
CREATE TABLE tabla_objetivo
(c1 INT PRIMARY KEY,
 c2 INT
 );

--Creación de varias TABLAS REFERENCIALES

--RESTRICCIÓN FOREIGN KEY en la declaración del propio campo.
CREATE TABLE tabla_referencial1
(c1 INT PRIMARY KEY,
 c2 INT REFERENCES tabla_objetivo(c1)
 );
```

```
CREATE TABLE tabla_referencial2
(c1 int PRIMARY KEY,
 c2 int FOREIGN KEY REFERENCES tabla_objetivo(c1)
 );

--RESTRICCIÓN FOREIGN KEY después de declarar todos los campos.
CREATE TABLE tabla_referencial3
(c1 INT,
 c2 INT,
 c3 INT,
 c4 CHAR(1),
 PRIMARY KEY (c1),
 FOREIGN KEY (c2) REFERENCES tabla_objetivo(c1)
 );

--RESTRICCIÓN FOREIGN KEY eligiendo el nombre.
/*Los nombres para las restricciones SON ÚNICOS en la base de
datos.*/

CREATE TABLE tabla_referencial4
(c1 INT PRIMARY KEY,
 c2 INT CONSTRAINT res1 REFERENCES tabla_objetivo(c1)
 );

CREATE TABLE tabla_referencial5
(c1 INT PRIMARY KEY,
 c2 INT CONSTRAINT res2 FOREIGN KEY REFERENCES tabla_objetivo(c1)
/*MySQL no permite la declaración así*/
 );

 CREATE TABLE tabla_referencial6
(c1 INT,
 c2 INT,
 CONSTRAINT res4 PRIMARY KEY (c1),
 CONSTRAINT res5 FOREIGN KEY (c2) REFERENCES tabla_objetivo(c1)
 );
```

```
/*Ver en el diagrama de la base de datos la FOREIGN KEY:
  nombre y reglas por defecto (NO ACTION). */

/*En las reglas de eliminación (ON DELETE) y actualización (ON
UPDATE)
podemos poner:
CASCADE | NO ACTION | SET NULL |SET DEFAULT

-SET DEFAULT es igual SET NULL si la clave foránea admite nulos y
no tiene establecido un valor predeterminado. En caso de tener un
valor por defecto se pone este.
*/
```

Para mantener la INTEGRIDAD de la base de datos se debe especificar por cada FOREIGN KEY que se defina en una tabla: la regla del borrado (ON DELETE) y la regla de la modificación (ON UPDATE).

REGLA DEL BORRADO (ON DELETE): Indica en la definición de la FOREIGN KEY qué debe ocurrir en la tabla referencial cuando hay un intento de borrar una fila en la tabla objetivo cuyo valor de PRIMARY KEY existe como valor de FOREIGN KEY en la tabla referencial.

Los posibles valores son: NO ACTION | CASCADE | SET NULL | SET DEFAULT.

Valor por defecto: NO ACTION.

REGLA DE LA MODIFICACIÓN (ON UPDATE): Indica en la definición de la FOREIGN KEY qué debe ocurrir en la tabla referencial cuando hay un intento de modificar un valor de PRIMARY KEY en la tabla objetivo que existe como valor de FOREIGN KEY en la tabla referencial.

Los posibles valores son: NO ACTION | CASCADE | SET NULL | SET DEFAULT.

Valor por defecto: NO ACTION.

Los siguientes ejemplos de restricción de FOREIGN KEY especificando las reglas de integridad para la modificación y la eliminación se encuentran en “Ejemplos ON DELETE y ON UPDATE.SQL”. El archivo pertenece a EJEMPLOS_LDD.

```
/*EJEMPLOS DE RESTRICCIÓN FOREIGN KEY ESPECIFICANDO LAS REGLAS
DE INTEGRIDAD PARA LA MODIFICACIÓN Y LA ELIMINACIÓN*/

CREATE DATABASE prueba3;
GO
USE prueba3;
```

```
/*A continuación, vemos diferentes ejemplos en los que ON DELETE y
ON UPDATE toman el mismo valor. Pero, estos valores pueden ser
diferentes para ajustarse a lo que pida la especificación.*/

/*Ejemplos con:
ON DELETE NO ACTION
ON UPDATE NO ACTION
*/

/*ON DELETE NO ACTION
Cuando hay un intento de borrar una fila de la TABLA
departamento y el valor de PK existe como FK en la tabla
empleado: NO SE PERMITE EL borrado. Así funciona por defecto,
si no se pone la cláusula ON DELETE.*/

/*ON UPDATE NO ACTION
Cuando hay un intento de modificar el valor de la PK
de la TABLA departamento y el valor de PK existe como FK
en la tabla empleado: NO SE PERMITE la modificación.
Así funciona por defecto si no se pone la cláusula ON UPDATE.*/

CREATE TABLE departamento
(
  CodDep INT NOT NULL,
  NomDep VARCHAR(20) NOT NULL,
  PRIMARY KEY (CodDep)
);

CREATE TABLE empleado
(
  CodEmp INT NOT NULL,
  NomEmp VARCHAR(60) NOT NULL,
  CodDep INT ,
  PRIMARY KEY (CodEmp),
  FOREIGN KEY (CodDep) REFERENCES departamento(CodDep)
                                  ON DELETE NO ACTION
                                  ON UPDATE NO ACTION
);

INSERT INTO departamento
VALUES (1,'DEP1'),(2,'DEP2'),(3,'DEP3');
```

```sql
INSERT INTO empleado
VALUES (1,'EMP1',1), (2,'EMP2',1), (3,'EMP3',1),
       (4,'EMP4',2), (5,'EMP5',2), (6,'EMP6',NULL);

SELECT * FROM departamento;
SELECT * FROM empleado;

DELETE FROM departamento
WHERE CodDep=1;

BEGIN TRANSACTION;

DELETE FROM departamento
WHERE CodDep=3;

ROLLBACK TRANSACTION;

UPDATE departamento
SET CodDep=CodDep*100
WHERE CodDep=1;

BEGIN TRANSACTION;

UPDATE departamento
SET CodDep=CodDep*100
WHERE CodDep=3;

ROLLBACK TRANSACTION;

DROP TABLE empleado, departamento;
```

```
/*Ejemplos con:
ON DELETE CASCADE
ON UPDATE CASCADE
*/

/*ON DELETE CASCADE
Cuando hay un intento de borrar una fila de la TABLA
departamento y el valor de PK existe como FK en la tabla
empleado: La fila SE BORRA de la TABLA departamento y
SE BORRAN todas las filas de la TABLA empleado que tengan
el mismo valor.*/

/*ON UPDATE CASCADE
Cuando hay un intento de modificar el valor de la PK
de la TABLA departamento y el valor de PK existe como FK
en la tabla empleado: La PK se modifica en la
TABLA departamento y se modifica el valor de la FK en
todas las filas de la TABLA empleado que tengan
el mismo valor.*/

CREATE TABLE departamento
(
  CodDep INT NOT NULL,
  NomDep VARCHAR(20) NOT NULL,
  PRIMARY KEY (CodDep)
);

CREATE TABLE empleado
(
  CodEmp INT NOT NULL,
  NomEmp VARCHAR(60) NOT NULL,
  CodDep INT ,
  PRIMARY KEY (CodEmp),
  FOREIGN KEY (CodDep) REFERENCES departamento(CodDep)
                                  ON DELETE CASCADE
                                  ON UPDATE CASCADE
);


INSERT INTO departamento
VALUES (1,'DEP1'),(2,'DEP2'),(3,'DEP3')

INSERT INTO empleado
VALUES  (1,'EMP1',1), (2,'EMP2',1),(3,'EMP3',1),
        (4,'EMP4',2), (5,'EMP5',2), (6,'EMP6',NULL);
```

```
SELECT * FROM departamento;
SELECT * FROM empleado;

BEGIN TRANSACTION;

DELETE FROM departamento
WHERE CodDep=1;

ROLLBACK TRANSACTION;

BEGIN TRANSACTION;

DELETE FROM departamento
WHERE CodDep=3;

ROLLBACK TRANSACTION;

BEGIN TRANSACTION;

UPDATE departamento
SET CodDep=CodDep*100
WHERE CodDep=1;

ROLLBACK TRANSACTION;

DROP TABLE empleado, departamento;
```

```
/*Ejemplos con:
ON DELETE SET NULL
ON UPDATE SET NULL
IMPORTANTE: El campo de la FK debe permitir nulos.
*/

/*ON DELETE SET NULL
Cuando hay un intento de borrar una fila de la TABLA
departamento y el valor de PK existe como FK en la tabla
empleado: La fila SE BORRA de la TABLA departamento y
en la TABLA empleado SE PONEN A NULL los valores de la
FK que coincidan con la PK de la fila que hemos borrado*/

/*ON UPDATE SET NULL
Cuando hay un intento de modificar el valor de la PK
de la TABLA departamento y el valor de PK existe como FK
en la tabla empleado: La PK se modifica en la
TABLA departamento y SE PONEN A NULL los valores de la
FK que coincidan con la PK de la fila que hemos borrado
os valores de la FK */

CREATE TABLE departamento
(
  CodDep INT NOT NULL,
  NomDep VARCHAR(20) NOT NULL,
  PRIMARY KEY (CodDep)
);

CREATE TABLE empleado
(
  CodEmp INT NOT NULL,
  NomEmp VARCHAR(60) NOT NULL,
  CodDep INT,
  PRIMARY KEY (CodEmp),
  FOREIGN KEY (CodDep) REFERENCES departamento(CodDep)
                               ON DELETE SET NULL
                               ON UPDATE SET NULL
);

INSERT INTO departamento
VALUES (1,'DEP1'),(2,'DEP2'),(3,'DEP3')

INSERT INTO empleado
VALUES  (1,'EMP1',1), (2,'EMP2',1), (3,'EMP3',1),
       (4,'EMP4',2), (5,'EMP5',2),    (6,'EMP6',NULL);
```

```sql
SELECT * FROM departamento;
SELECT * FROM empleado;

BEGIN TRANSACTION;

DELETE FROM departamento
WHERE CodDep=1;

ROLLBACK TRANSACTION;

BEGIN TRANSACTION;

UPDATE departamento
SET CodDep=CodDep*100
WHERE CodDep=1;

ROLLBACK TRANSACTION;

DROP TABLE empleado, departamento;

/*Ejemplos con:
  ON DELETE SET DEFAULT
  ON UPDATE SET DEFAULT
*/
/*Igual que SET NULL, pero la FK toma el valor indicado
por defecto en la creación de la TABLA REFERENCIAL, en vez de
tomar el valor NULL.*/

CREATE TABLE departamento
(
  CodDep INT NOT NULL,
  NomDep VARCHAR(20) NOT NULL,
  PRIMARY KEY (CodDep)
);
CREATE TABLE empleado
(
  CodEmp INT NOT NULL,
  NomEmp VARCHAR(60) NOT NULL,
  CodDep INT  DEFAULT 0,
  PRIMARY KEY (CodEmp),
  FOREIGN KEY (CodDep) REFERENCES departamento(CodDep)
                                  ON DELETE SET DEFAULT
                                  ON UPDATE SET DEFAULT
);
```

```
INSERT INTO departamento
VALUES (0,'Sin departamento'),(1,'DEP1'),(2,'DEP2'),(3,'DEP3')

INSERT INTO empleado
VALUES (1,'EMP1',1), (2,'EMP2',1),(3,'EMP3',1),
       (4,'EMP4',2),(5,'EMP5',2),(6,'EMP6',2),
       (7,'EMP7', DEFAULT);

SELECT * FROM departamento;
SELECT * FROM empleado;

BEGIN TRANSACTION;

DELETE FROM departamento
WHERE CodDep=1;

ROLLBACK TRANSACTION;

BEGIN TRANSACTION;

UPDATE departamento
SET CodDep=CodDep*100
WHERE CodDep=1;

ROLLBACK TRANSACTION;
```

Para terminar con la restricción de FOREIGN KEY veamos un ejemplo de FOREIGN KEY compuesta y del comportamiento de la base de datos cuando se tiene una entidad débil con dependencia en identificación y cuando se tiene una entidad débil con dependencia en existencia.

Los siguientes ejemplos de restricción de FOREIGN KEY se encuentran en “Ejemplo FK Compuesta.SQL”. El archivo pertenece a EJEMPLOS_LDD.

```
CREATE DATABASE coches;
GO
USE coches;
```

```sql
--Ejemplos de FOREIGN KEY compuesta.
--Ejemplos de entidad débil con dependencia en IDENTIFICACIÓN.
CREATE TABLE marca
(
  CodMar INT NOT NULL,
  NomMar VARCHAR(50) NOT NULL,
  PRIMARY KEY (CodMar)
);

CREATE TABLE modelo
(
  CodMar INT NOT NULL,
  CodMod INT NOT NULL,
  NomMod VARCHAR(100) NOT NULL,
  PRIMARY KEY (CodMar, CodMod),
  FOREIGN KEY (CodMar) REFERENCES marca(CodMar)
                               ON DELETE CASCADE
                               ON UPDATE NO ACTION
);

CREATE TABLE especificacion
(
  CodEsp INT NOT NULL,
  Caballos INT NOT NULL,
  CodMar INT NOT NULL,
  CodMod INT NOT NULL,
  PRIMARY KEY (CodEsp),
  FOREIGN KEY (CodMar, CodMod) REFERENCES modelo(CodMar, CodMod)
                                          ON DELETE CASCADE
                                          ON UPDATE NO ACTION
);

/*Siguiendo las normas de un entidad débil con dependencia
en IDENTIFICACIÓN insertamos datos en las tablas marca,
modelo y especificacion*/

INSERT INTO marca VALUES
(1, 'FORD'),
(2, 'NISSAN');

INSERT INTO modelo VALUES
(1, 1, 'FOCUS'),
(1, 2, 'PUMA'),
(2, 1, 'QASHQAI'),
(2, 2, 'JUKE');
```

```sql
INSERT INTO especificacion VALUES
(1, 2000, 1, 2),
(2, 3500, 1, 2),
(3, 2250, 2, 2);

BEGIN TRANSACTION;

DELETE FROM marca
WHERE NomMar LIKE 'FORD';


SELECT * FROM marca;
SELECT * FROM modelo;
SELECT * FROM especificacion;

ROLLBACK TRANSACTION;

--Ejemplos de entidad débil con dependencia en EXISTENCIA

CREATE TABLE marca
(
  CodMar INT NOT NULL,
  NomMar VARCHAR(50) NOT NULL,
  PRIMARY KEY (CodMar)
);

CREATE TABLE modelo
(
  CodMod INT NOT NULL,
  CodMar INT NOT NULL,
  NomMod VARCHAR(100) NOT NULL,
  PRIMARY KEY (CodMod),
  FOREIGN KEY (CodMar) REFERENCES marca(CodMar)
                               ON DELETE CASCADE
                               ON UPDATE NO ACTION
);
```

```sql
CREATE TABLE especificacion
(
  CodEsp INT NOT NULL,
  Caballos INT NOT NULL,
  CodMod INT NOT NULL,
  PRIMARY KEY (CodEsp),
  FOREIGN KEY (CodMod) REFERENCES modelo(CodMod)
                               ON DELETE CASCADE
                               ON UPDATE NO ACTION
);

/*Siguiendo las normas de un entidad débil con dependencia
en EXISTENCIA insertamos datos en las tablas marca,
modelo y especificación*/

INSERT INTO marca VALUES
(1, 'FORD'),
(2, 'NISSAN');

INSERT INTO modelo VALUES
(1, 1, 'FOCUS'),
(2, 1, 'PUMA'),
(3, 2, 'QASHQAI'),
(4, 2, 'JUKE');

INSERT INTO especificacion VALUES
(1, 2000, 2),
(2, 3500, 2),
(3, 2250, 4);

BEGIN TRANSACTION;

DELETE FROM marca
WHERE NomMar LIKE 'FORD';


SELECT * FROM marca;
SELECT * FROM modelo;
SELECT * FROM especificacion;

ROLLBACK TRANSACTION;
```

∘ RESTRICCIÓN DEFAULT

Los valores predeterminados suministran un valor cuando no se especifica ninguno. Por ejemplo, la base de datos **AdventureWorks** puede incluir una tabla de búsqueda con los distintos trabajos que los empleados pueden realizar en la compañía. En la columna que describe cada trabajo, el valor predeterminado de cadena de caracteres puede suministrar una descripción si no se ha escrito una descripción de forma explícita.

DEFAULT 'New Position - title not formalized yet'

Además de constantes, las definiciones de DEFAULT pueden incluir funciones.

La restricción DEFAULT no se puede aplicar a aquellos campos que tengan la propiedad `IDENTITY`.

Los siguientes ejemplos de restricción de DEFAULT se encuentran en "EjemplosRestricciónDefault.SQL". El archivo pertenece a EJEMPLOS_LDD.

```
--EJEMPLOS DE CREACIÓN DE TABLAS.
--RESTRICCIÓN DEFAULT

CREATE DATABASE prueba4;
GO
USE prueba4;

--CREACIÓN DE UN TIPO DE DATOS.
/*Utilizamos el procedimiento almacenado sp_addtype. Una vez creado
lo podemos ver en Programmability/Types*/
sp_addtype nombres, 'VARCHAR(30)','NOT NULL';

--Una vez creada la tabla ver el apartado Constraints.
CREATE TABLE tabla1
(c1 INT PRIMARY KEY,
 c2 nombres DEFAULT 'Málaga'
 );

--Añadimos datos utilizando la palabra reservada DEFAULT.
 INSERT INTO tabla1 (c1,c2)
 VALUES (1, DEFAULT);

 INSERT INTO tabla1
 VALUES (2, DEFAULT);

 INSERT INTO tabla1 (c1)
 VALUES (3);

SELECT * FROM tabla1;
```

```sql
--El campo c2 toma por defecto el valor 0.
CREATE TABLE tabla2
(c1 INT PRIMARY KEY,
 c2 INT DEFAULT 0
 );

/*
 -Observa que al no introducir valor en c2, toma el valor por
  defecto. Esto ocurre tanto si el campo admite nulos, como si no.
 -Si el campo admite nulos y queremos almacenar NULL, se tiene que
  escribir explícitamente.*/
 INSERT INTO tabla2 (c1)
 VALUES (1);

 INSERT INTO tabla2 (c1,c2)
 VALUES (2,DEFAULT),(3,NULL);

 SELECT * FROM tabla2;

--El campo c2 toma por defecto la fecha-hora actual.
CREATE TABLE tabla3
(c1 INT PRIMARY KEY,
 c2 DATETIME DEFAULT CURRENT_TIMESTAMP
 );

INSERT INTO tabla3
VALUES(1,DEFAULT);

SELECT * FROM tabla3;

--El campo c2 toma por defecto la fecha-hora actual.
CREATE TABLE tabla4
(c1 INT IDENTITY PRIMARY KEY,
 c2 DATETIME DEFAULT GETDATE()
 );

INSERT INTO tabla4
VALUES(DEFAULT);

SELECT * FROM tabla4;
```

```
/*
 -RESTRICCIÓN DEFAULT eligiendo el nombre.
 -Los nombres para las restricciones SON ÚNICOS en la base de
  datos.
 -Las restricciones de PRIMARY KEY, FOREIGN KEY y UNIQUE se pueden
  definirse como restricciones de tabla después de declarar todas
  las columnas, pero las restricciones DEFAULT sólo pueden
  declararse en la definición de la columna.
 -El valor por defecto también puede ir entre paréntesis.
*/
CREATE TABLE tabla5
(c1 INT PRIMARY KEY,
 c2 INT CONSTRAINT res1 DEFAULT (0)
 );
```

∘ RESTRICCIÓN UNIQUE

Las restricciones UNIQUE se utilizan para exigir la unicidad en las columnas de claves no principales. En el siguiente ejemplo se impone la restricción de que la columna Name de la tabla Product debe ser única.

Name nvarchar(100) NOT NULL UNIQUE NONCLUSTERED

CLUSTERED | NONCLUSTERED

Indica que se ha creado un índice agrupado o no agrupado para la restricción PRIMARY KEY o UNIQUE. De forma predeterminada, el valor de las restricciones PRIMARY KEY es CLUSTERED, y el de las restricciones UNIQUE es NONCLUSTERED.

En una instrucción CREATE TABLE, se puede especificar CLUSTERED tan sólo para una restricción. Si especifica CLUSTERED para una restricción UNIQUE y especifica también una restricción PRIMARY KEY, el valor predeterminado de PRIMARY KEY es NONCLUSTERED.

Los índices agrupados (CLUSTERED) ordenan y almacenan las filas de los datos de la tabla de acuerdo con los valores de la clave del índice. Sólo puede haber un índice clúster por cada tabla, porque las filas de datos sólo pueden estar ordenadas de una forma.

Un índice no agrupado (NONCLUSTERED) contiene los valores de clave del índice y localizadores de fila que apuntan a la ubicación de almacenamiento de los datos de la tabla.

Los siguientes ejemplos de restricción de UNIQUE se encuentran en "EjemplosRestricciónUnique.SQL". El archivo pertenece a EJEMPLOS_LDD.

```
--EJEMPLOS DE CREACIÓN DE TABLAS.
--RESTRICCIÓN UNIQUE

/*
-Crea un índice sobre el campo.
-Se suelen crear sobre las claves alternativas, puesto
 que normalmente se harán búsquedas con frecuencia sobre ellas.
-Un campo con restricción UNIQUE puede admitir valores nulos,
 pero no valores duplicados.
*/

USE prueba4;

--Una vez creada la tabla ver los apartados Keys e Indexes.
CREATE TABLE tabla6
(c1 INT IDENTITY PRIMARY KEY,
 c2 nombres,
 c3 INT UNIQUE
 );

INSERT INTO tabla6 (c2,c3)
VALUES ('Juan',10),('Lola',21),('Ana',NULL);

SELECT * FROM tabla6;

--RESTRICCIÓN UNIQUE eligiendo el nombre.
--Los nombres para las restricciones SON ÚNICOS en la base de
datos.
CREATE TABLE tabla7
(c1 int PRIMARY KEY,
 c2 nombres,
 c3 INT CONSTRAINT res2 UNIQUE
 );

CREATE TABLE tabla8
(c1 INT PRIMARY KEY,
 c2 nombres,
 c3 INT,
 CONSTRAINT res3 UNIQUE(c3)
 );
```

```
/*
Observa cómo se ordenan las filas de la siguiente tabla
por el valor de la PK en orden ascendente.
Esto es debido a que cuando se ha creado la tabla, se han
creado dos índices automáticamente: uno para la PK de tipo
CLUSTERED y otro para el campo de la restricción UNIQUE de
tipo NONCLUSTERED
*/
INSERT INTO tabla8
VALUES (1,'Pepe',10),(3,'Ana',20);

INSERT INTO tabla8
VALUES (2,'Rosa',15);

SELECT * FROM tabla8;

/*Creación de una tabla con la restricción UNIQUE especificando
que es CLUSTERED.
En este caso, el índice de la PK de crea NONCLUSTERED, y por tanto,
las filas de la tabla se ordenan por el campo que tiene la
restricción UNIQUE.
*/
CREATE TABLE tabla9
(c1 INT PRIMARY KEY,
 c2 nombres,
 DNI CHAR(9) UNIQUE CLUSTERED
 );

INSERT INTO tabla9
VALUES (1,'Pepe','12345678B'),
       (2,'Ana','12345678A');

SELECT * FROM tabla9;
```

∘ RESTRICCIÓN CHECK

Las restricciones CHECK no se pueden definir en las columnas **text**, **ntext** o **image**. CHECK obliga a que los valores introducidos en el campo cumplan el predicado que se especifica en dicha restricción.

En el siguiente ejemplo se muestra una restricción para los valores escritos en la columna CreditRating de la tabla Vendedor. La restricción no tiene nombre.

CHECK (CreditRating >= 1 and CreditRating <= 5)

En este ejemplo se especifica que **los valores** se deben incluir **en una lista** específica **o seguir un patrón dado**.

CHECK (emp_id IN ('1389', '0736', '0877', '1622', '1756') OR emp_id

LIKE '99[0-9][0-9]')

En este ejemplo se muestra una restricción con nombre con una restricción de patrón en los datos de caracteres escritos en la columna de la tabla.

CONSTRAINT CK_emp_id CHECK (emp_id LIKE

'[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]'

OR emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]')

Los siguientes ejemplos de restricción de CHECK se encuentran en “EjemplosRestricciónCheck.SQL”. El archivo pertenece a EJEMPLOS_LDD.

```
--EJEMPLOS DE CREACIÓN DE TABLAS.
--RESTRICCIÓN CHECK

/*Observa que la forma que tiene la restricción CHECK es:
-CHECK(PREDICADO).
-Donde PREDICADO será cualquiera de los estudiados en
 la sentencia SELECT del LMD.
-Los únicos tipos de predicados que NO se permiten son los
 que incluyen un subselect.*/

USE prueba4;
```

```
/*
-Una vez creada la tabla ver el apartado Constraints.
-Observa que en el apartado de Constraints de solo
 aparecen las restricciones de DEFAULT y CHECK.
*/
CREATE TABLE tabla10
(c1 INT PRIMARY KEY,
 c2 INT CHECK (c2 >= 16)
 );

--Esta inserción incumple la restricción CHECK y dará error
INSERT INTO tabla10
VALUES (1,10);

CREATE TABLE tabla11
(c1 INT PRIMARY KEY,
 c2 INT CHECK (c2 >=16 AND c2<65)
 );

CREATE TABLE tabla12
(c1 INT PRIMARY KEY,
 c2 INT CHECK (c2 IN ('1389', '0736', '0877', '1622', '1756')
              OR c2 LIKE '99[0-9][0-9]')
);

CREATE TABLE tabla13
(c1 INT PRIMARY KEY,
 c2 INT CHECK (c2 BETWEEN 0 AND 1000)
);

CREATE TABLE tabla14
(c1 INT PRIMARY KEY,
 c2 CHAR(9) CHECK (c2 LIKE
                 '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]')
);

CREATE TABLE tabla15
(c1 INT PRIMARY KEY,
 c2 CHAR(9) NOT NULL UNIQUE CHECK (c2 LIKE
            '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]')
);
```

```
--RESTRICCIÓN CHECK eligiendo el nombre.
/*Los nombres para las restricciones SON ÚNICOS en la base de
datos.*/
CREATE TABLE tabla16
(c1 INT PRIMARY KEY,
 c2 INT CONSTRAINT res4 CHECK (c2 >=16)
);

CREATE TABLE tabla17
(c1 INT PRIMARY KEY,
 c2 INT,
 CONSTRAINT res5 CHECK (c2 >=16)
 );
```

SENTENCIA DROP TABLE

Permite eliminar una tabla o más tablas

DROP TABLE tabla1 [, …n];