Administración Bases de Datos

Semana 4

Docente: Lic. Norberto A. Orlando

Modelo Cliente/Servidor

Durante las clases de este curso, nuestra aplicación cliente será el Managment Studio de SQL Server, y la aplicación servidora será el motor de base de datos SQL Server

Modelo Cliente/Servidor

Características de un cliente

● Son quienes inician las solicitudes o peticiones y tienen un papel activo en la comunicación.

● Esperan y reciben las respuestas del servidor.

● Por lo general, pueden conectarse a varios servidores a la vez según la lógica de negocio que necesitan resolver.

● Normalmente interactúan directamente con los usuarios finales mediante una interfaz gráfica de usuario. También pueden ser servicios o programas sin interacción con el usuario.

Modelo Cliente/Servidor

Características de un servidor

● Al iniciarse esperan a que lleguen las solicitudes de los clientes para poder atenderlas y procesarlas, desempeñando un papel pasivo en la comunicación.

● Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.

● Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).

● No es frecuente que interactúen directamente con los usuarios finales.

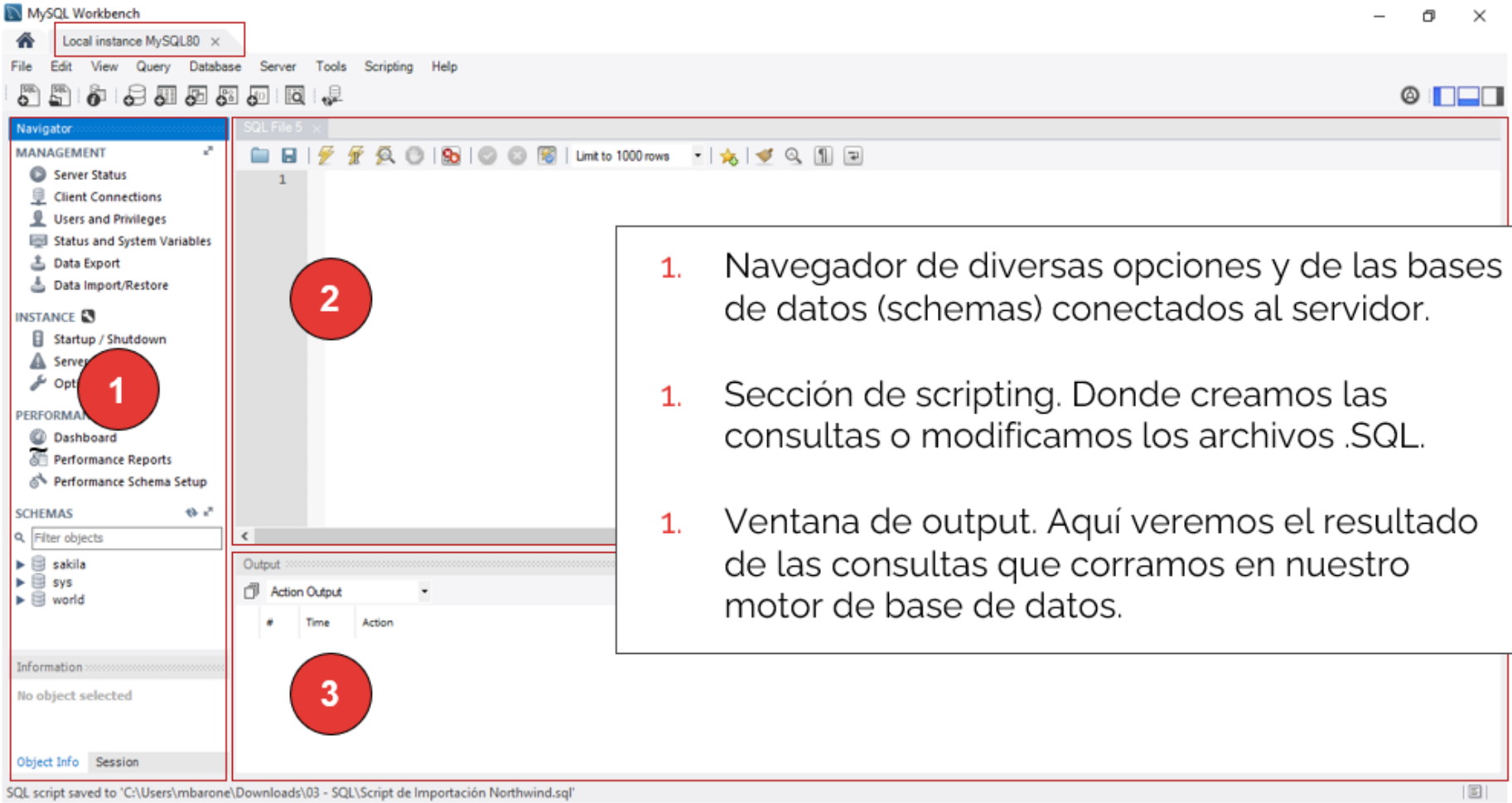
Modelo Cliente/Servidor

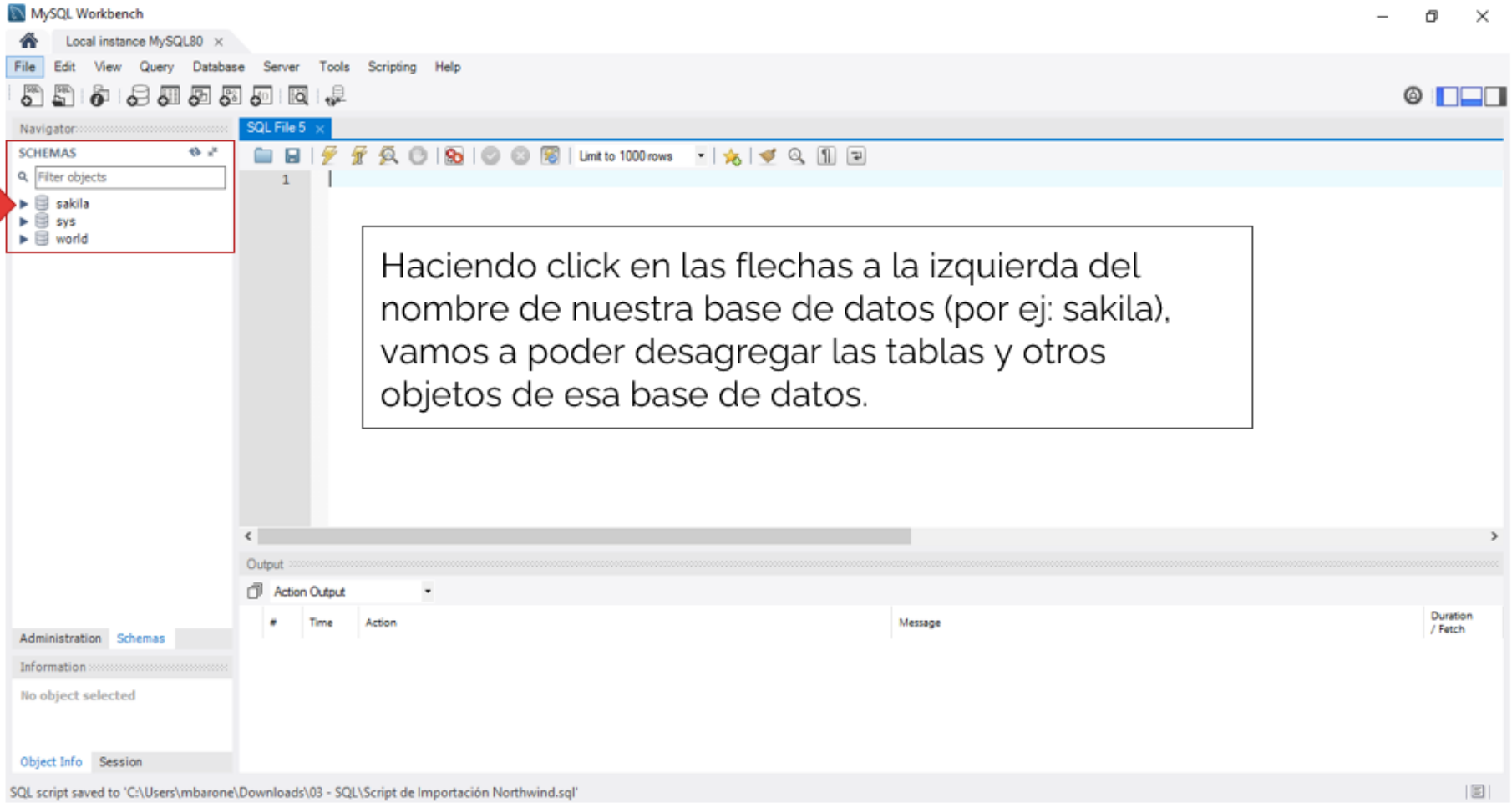
Ventajas

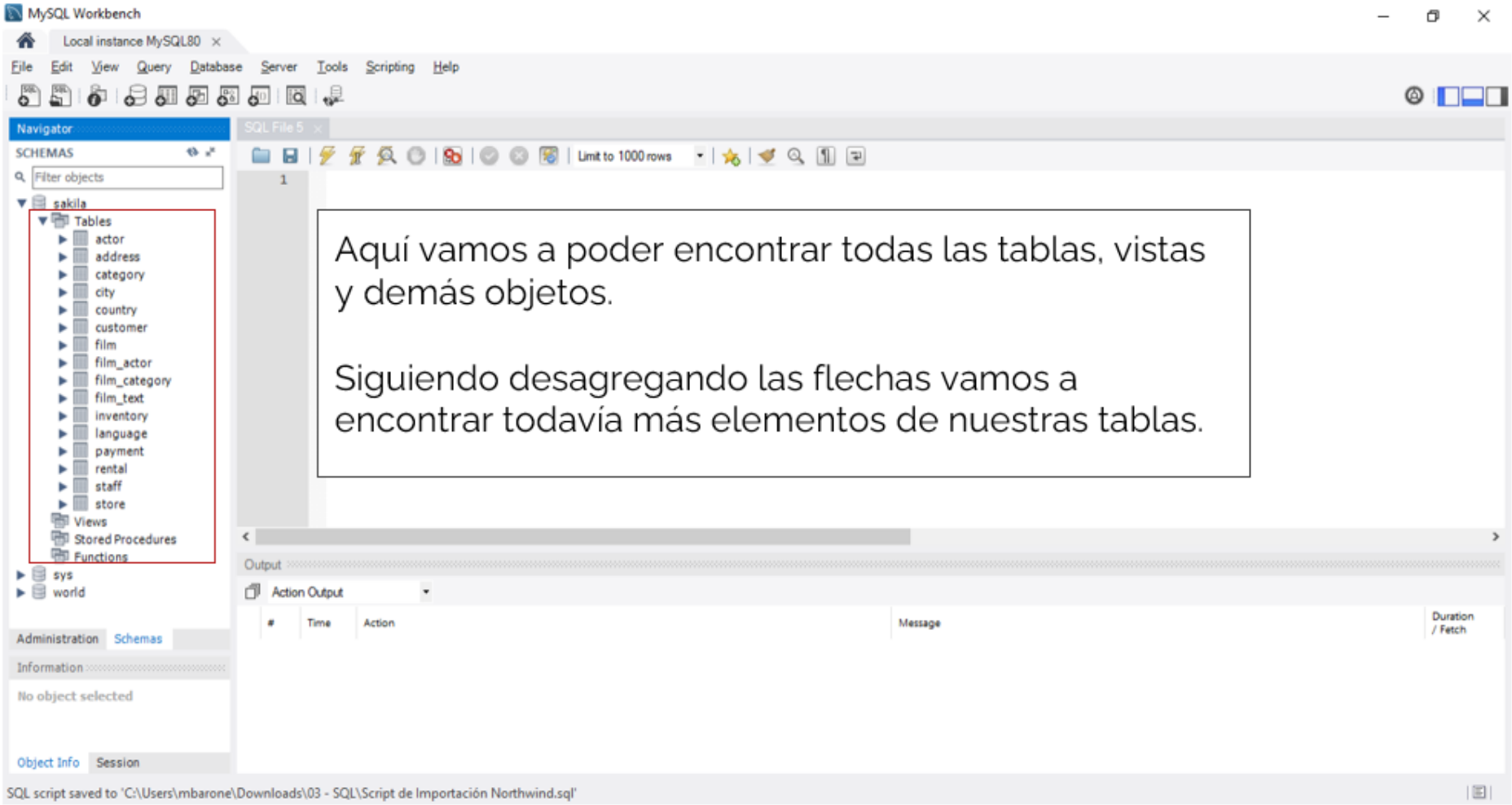
● **Escalabilidad** : Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden agregar nuevos nodos a la red (clientes y/o servidores).

● M**antenimiento** : Al estar distribuidas las funciones y responsabilidades entre varias computadoras independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por el cambio. Hoy día es frecuente también tener servidores en la nube

● **Seguridad:** SQL Server permite administrar permisos a TODO. Permisos a nivel de servidor, a nivel de base de datos, seguridad en tablas, permitir o no lectura de datos, escritura de datos, ejecución de procedimientos almacenados de la base de datos,etc..







Creación de una base de datos

Para crear una base de datos (por ejemplo, con el nombre ***ComercioIT***), la **instrucción SQL** a ejecutar será:

```
CREATE DATABASE ComercioIT;
```

ALTER o DROP DATABASE

- **Para cambiar las opciones de las bases de datos utilice ALTER DATABASE**
- **Para eliminar una base de datos utilice la instrucción DROP DATABASE { nombre_database}.**

[code]

```
USE master;
GO
        ALTER DATABASE AdventureWorks SET READ_ONLY
```

[/code]

[code]

```
USE master;
GO
         ALTER DATABASE AdventureWorks SET READ_WRITE
```

[/code]

Tablas

- Son el **objeto principal de una base de datos**, ya que en ellas se **almacenarán** los datos.
- Son objetos compuestos por una estructura que almacena datos relacionados acerca de algún **objeto/entidad** en general.
- Las tablas tienen un **nombre** y debe ser único en toda la base datos.
- Están compuestas por **registros (filas) y campos (columnas)**.
- Los registros y campos pueden estar en **diferentes órdenes**.
- Una base de datos puede contener **varias tablas**.

Código	Nombre	Precio	Stock
1	iPod	299	200
2	iPhone	399	300
3	iPad	499	250
4	MacBook Pro	1199	150

Ejemplo de tabla de 4 campos y 4 registros.

Tablas

CREATE TABLE

La sintaxis básica y simplificada para crear una tabla es la siguiente:

```
CREATE TABLE NombreTabla(
        NombreCampo1 TipoDeDato,
        ...
        NombreCampoN TipoDeDato
)
```

[code]

```
CREATE TABLE usuarios (
  nombre varchar(30),
  clave varchar(10)
                    )
```

[/code]

Ejemplo:

```
CREATE TABLE Productos(
idProducto INT(11) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(50) NOT NULL,
Precio DOUBLE,
Marca VARCHAR(30) NOT NULL,
Categoria VARCHAR(30) NOT NULL,
Stock INT(6) NOT NULL,
Disponible BOOLEAN DEFAULT false
);
```

ALTER TABLE

Este comando permite **modificar la estructura de una tabla**. Se utiliza para agregar o eliminar columnas, modificar el tipo de datos de una columna existente o renombrar una columna o una tabla.

En el ejemplo debajo, se **agrega una nueva columna** con el nombre *Observaciones*, a una tabla con el nombre *articulos*:

```
ALTER TABLE articulos ADD COLUMN Observaciones VARCHAR(50) NULL;
```

Agregar columnas a una tabla

En el siguiente ejemplo, se **agrega una nueva columna** con el nombre *Primera*, a una tabla con el nombre *clientes*. Y se la **ubica al comienzo de la tabla**:

```
ALTER TABLE clientes ADD COLUMN Primera VARCHAR(50) NULL FIRST;
```

Cambiar el nombre de una columna

En el siguiente ejemplo, se **modifica el nombre de la columna *observaciones* por *comentarios***, en la tabla *articulos*:

```
ALTER TABLE articulos
CHANGE observaciones comentarios VARCHAR(40) NULL;
```

Eliminar una columna

En el siguiente ejemplo, se **elimina la columna *comentarios*** de la tabla *articulos*:

```
ALTER TABLE articulos DROP COLUMN comentarios;
```

Y en el siguiente ejemplo, se **eliminan las columnas *Primera* y *Siguiente*** de la tabla *clientes*, en una sola sentencia:

```
ALTER TABLE clientes DROP COLUMN Primera, DROP COLUMN Siguiente;
```

Tablas

Eliminar Tablas

Se utiliza la sentencia **DROP TABLE:**

```
DROP TABLE  NombreDeLaTabla;
```

Super simple y **PELIGROSO!** Este tipo de sentencias no son reversibles, hay que tener mucho cuidado cuando se está redactando en no cometer errores al introducir el nombre de la tabla.

En entornos productivos suele limitarse este tipo de sentencias a ciertos usuarios administradores únicamente.

Eliminar una tabla

Para eliminar una tabla (por ejemplo, con el nombre *Productos*) de una base de datos, la **instrucción SQL** a ejecutar será:

```
DROP TABLE Productos;

o

DROP TABLE IF EXISTS Productos;
```

Nota: la cláusula ***IF EXISTS*** devuelve una advertencia en caso de que no exista la tabla a eliminar.

Tipos de Dato de las Columnas de la Tabla

- El tipo de dato especifica el tipo de información que puede guardar un campo.
- Cada columna, variable local, expresión y parámetro tiene un tipo de dato relacionado.

Categorías de tipos de dato

Categoría	Tipo de Dato	
Numéricos	Enteros	Int, bigint, smallint, tinyint
	Exactos	decimal, numeric
	Aproximados	float, real
	Monetarios	money, smallmoney
Fecha y Hora	datetime, smalldatetime	
Caracter	No-Unicode	char, varchar, text
	Unicode	nchar, nvarchar, ntext
Binarios	binary, varbinary, image	
Especiales	timestamp, uniqueidentifier, xml, table, bit	

Integridad de Datos

Se entiende por integridad de datos, a las restricciones, controles y validaciones que diseñamos para proteger la información almacenada en la base de datos y que la misma quede libre de incoherencias según nuestro criterio y la lógica de negocio de los datos que se modelan.

Restricciones

Propiedades para asegurar la integridad de los datos:

- Tipos de dato
- Definiciones NULL y NOT NULL
- Valores por omisión para campos con definiciones DEFAULT
- Propiedades IDENTITY O AUTO_INCREMENT
- Reglas de validación con la propiedad CHECK
- Desencadenadores (triggers)
- Indices y claves primarias/foráneas.

Integridad de campos

La Integridad de campos también llamada de dominio es la validación de las entradas en una determinada columna:

- Definiciones NULL y NOT NULL

Características:

- Si no se indica en la creación del campo, por omisión siempre aceptan nulos.
- La nulabilidad de una columna determina si las filas en la tabla pueden contener valores nulos para esa columna.
- Un valor nulo no es lo mismo que un cero
- Un valor nulo no es lo mismo que un blanco
- Un valor nulo no es lo mismo que una cadena de caracteres de longitud cero.

```
CREATE TABLE CLIENTES
(
        id              INT NOT NULL,
        Nombre          VARCHAR(50) NOT NULL,
        Nombre2         VARCHAR(50) NULL,
        Apellido VARCHAR(50) NOT NULL
        )
```

Integridad de campos

- DEFAULT: Valores por omisión → Los valores por omisión indican que valor será guardado en una columna si no se especifica un valor para la columna cuando se inserta una fila. Cada columna en una tabla puede contener una sola definición DEFAULT.

Características:

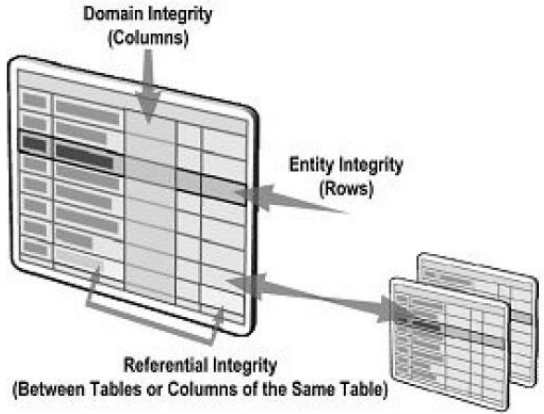
- Va asociado a la columna y no al tipo de dato.
- Solo un DEFAULT puede ser asignado a una columna.
- No puede ser asignado a una columna de tipo IDENTITY.
- Permiten utilizar funciones del sistema.

```
CREATE TABLE Production.Location
(
        LocationID      SMALLINT IDENTITY(1,1) NOT NULL,
        Name            dbo.Name NOT NULL,
        CostRate        SMALLMONEY NOT NULL
                        CONSTRAINT DF_Location_CostRate
                        DEFAULT ((0.00)),
        Availability    DECIMAL(8, 2) NOT NULL
                        CONSTRAINT DF_Location_Availability
                        DEFAULT ((0.00)),
        ModifiedDate    DATETIME NOT NULL
                        CONSTRAINT DF_Location_ModifiedDate
                        DEFAULT (GETDATE())
)
```

Integridad de Filas

Integridad de Filas

- La Integridad de filas define una fila como una única instancia de una entidad para una tabla en particular.
- La integridad de filas asegura la integridad de la columna de identificación o la clave primaria de una tabla (a través de índices, restricciones UNIQUE, restricciones PRIMARY KEY, o propiedades IDENTITY).
- La Integridad referencial preserva las relaciones definidas entre tablas, cuando se ingresan, modifican o borran registros.



Integridad de Filas

Integridad Referencial

- Se implementa definiendo **Claves Primarias** y **Claves Foráneas**.

La integridad referencial intenta preservar los datos de los siguientes problemas que pueden suceder entre una tabla padre y una tabla hijo:

- Agregar registros a una tabla relacionada (tabla hijo) si no hay registros asociados en la correspondiente tabla primaria (tabla padre).
- Cambiar valores en la tabla primaria (tabla padre) que resulten en registros huérfanos en las tablas relacionadas (tablas hijo).
- Borrar registros desde una tabla primaria (tabla padre) si existen registros relacionados en la tabla dependiente (tabla hijo).

Integridad de Filas

Clave Primaria (PRIMARY KEY)

Se llama clave primaria a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla.

Características:

- Se pueden crear restricciones PRIMARY KEY cuando se crea la tabla o agregarla a una tabla ya existente
- No debe existir otra restricción PRIMARY KEY para esa tabla
- Se puede modificar o eliminar una restricción PRIMARY KEY después que ha sido creada
- Cuando una restricción PRIMARY KEY se agrega a una columna (o columnas) existentes en un tabla, SQL Server controla los datos ya almacenados en las columnas para asegurar que se cumple la restricción, no existan valores nulos ni valores duplicados

```
CREATE TABLE nombre_de_la_tabla (
    id_col  INT,
    col2    varchar(20),
    ...
    CONSTRAINT pk_tabla PRIMARY KEY(id_col),
    ... )
```

```
CREATE TABLE nombre_de_la_tabla (
    id_col  INT  PRIMARY KEY,
    col2    varchar(20),
    ...
 )
```

Integridad de Filas

Clave Foránea (FOREIGN KEY)

Una clave foránea es aquella columna que existiendo como dependiente en una tabla, es a su vez clave primaria en otra. tabla.

Características

- Una tabla puede tener varias restricciones FOREIGN KEY.
- Una restricción FOREIGN KEY no puede modificarse, debe eliminarse y volverse a crear.
- No se puede eliminar una tabla referenciada en una restricción FOREIGN KEY, aparece un mensaje de error.
- Para ver información acerca de esta restricción podemos ejecutar el procedimiento almacenado sp_helpconstraint.
- La integridad referencial se activa en cuanto creamos una clave foránea y a partir de ese momento se comprueba cada vez que se modifiquen datos que puedan alterarla.

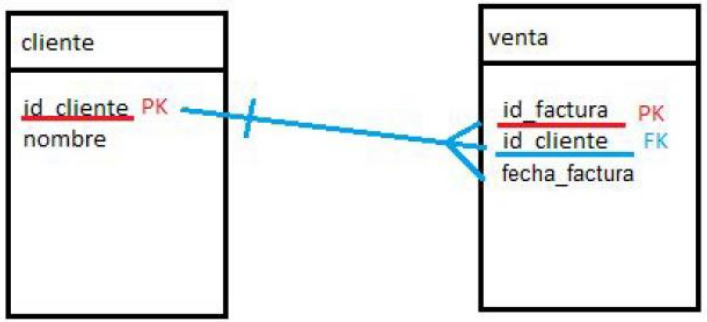
Integridad de Filas

Clave Foránea (FOREIGN KEY)

Sintaxis

La siguiente es la sintaxis parcial general para agregar una restricción "foreign key":

```
alter table NOMBRETABLA1
  add constraint NOMBRERESTRICCION
  foreign key (CAMPOCLAVEFORANEA)
  references NOMBRETABLA2 (CAMPOCLAVEPRIMARIA);
```



Por ejemplo, para agregar una restricción "foreign key" al campo "codigoeditorial" de la tabla “libros":

```
ALTER TABLE libros
ADD CONSTRAINT FK_libros_codigoeditorial
FOREIGN KEY (codigoeditorial) REFERENCES editoriales(codigo);
```

Integridad de Filas

Agregar y eliminar restricciones

En el siguiente ejemplo, se **elimina la restricción *Primary Key*** de la tabla *Articulos*:

```
ALTER TABLE Articulos DROP Primary Key;
```

En el siguiente ejemplo, se **agrega la restricción *Primary Key*** al campo *ArticuloID*, en la tabla *Articulos*:

```
ALTER TABLE Articulos ADD Primary Key(ArticuloID);
```

Integridad de Filas

En el siguiente ejemplo, se **elimina la restricción *FOREIGN KEY* del campo *fk_articulo***, de la tabla *facturas*:

```
ALTER TABLE facturas DROP FOREIGN KEY fk_articulo;
```

Y en el ejemplo siguiente, se **agrega la restricción *FOREIGN KEY* al campo *fk_articulo*** de la tabla *facturas*:

```
ALTER TABLE facturas ADD CONSTRAINT fk_articulo
FOREIGN KEY(ArticuloID) REFERENCES Articulos(ArticuloID);
```