# **El lenguaje SQL**

Una vez que se han determinado los requerimientos de información, y funciones que una aplicación debe realizar, se realiza un modelo lógico que permite abstraer la estructura de dicha información y las interrelaciones entre los datos. Posteriormente, ***este*** ***modelo*** ***lógico*** ***es*** ***implementado*** usando un SGBD ***y*** ***pasa*** ***a*** ***llamarse*** ***modelo*** ***físico***.

Para codificar la BD, restricciones de acceso y operaciones sobre los datos se utiliza el lenguaje ***SQL*** cuyas siglas significan ***Structured*** ***Query*** ***Language*** (Lenguaje de Consulta Estructurado).

SQL (pronunciado S-Q-L) es un lenguaje específico del dominio utilizado en programación diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. (RDBMS), o para el procesamiento de flujos en un sistema de gestión de flujos de datos (flujos continuos de información) relacionales (RDSMS). Es particularmente útil en el manejo de datos estructurados, es decir, datos que incorporan relaciones entre entidades y variables.

SQL fue uno de los primeros lenguajes comerciales para el modelo relacional de Edgar Frank Codd como se describió en su artículo de investigación de 1970 “*El modelo relacional de datos para grandes bancos de datos compartidos*”. A pesar de no adherirse totalmente al modelo relacional descrito por Codd, pasó a ser el lenguaje de base de datos más usado.

Inicialmente IBM rechazó implementar el modelo relacional de Codd quien decidió presentarlo a los clientes de IBM quienes a su vez presionaron a la compañía quien desarrollo un lenguaje (no relacional) llamado SEQUEL que sirvió de base para Oracle DB lanzado en 179.

El primer RDBMS de IBM fue SQL/DS lanzado en 1981, sin embargo, SQL pasó a ser el estándar del Instituto Nacional Estadounidense de Estándares (ANSI) en 1986 y de la Organización Internacional de Normalización (ISO) en 1987. Desde entonces, el estándar ha sido revisado para incluir más características. A pesar de la existencia de ambos estándares, la mayoría de los códigos SQL no son completamente portables entre sistemas de bases de datos diferentes sin otros ajustes debido a que los fabricantes no se adhieren completamente al estándar e implementan características y extensiones propias.

Las versiones del estándar SQL van desde SQL-86 hasta SQL:2019 siendo las más extendidas SQL-99 y SQL:2003

SQL se basa en la ***Teoría*** ***Matemática*** ***del*** ***Álgebra*** ***Relacional***, conjunto de operaciones que describen paso a paso cómo computar una respuesta sobre las relaciones, tal y como éstas son definidas en el modelo relacional.

El lenguaje SQL consta de varios elementos:

|  |  |
| --- | --- |
| CREATE | Para crear objetos en la base de datos. |
| ALTER | Modifica la estructura de la base de datos. |
| DROP | Borra objetos de la base de datos. |
| TRUNCATE | Se quitan todos los registros de una tabla, incluidos todos los espacios asignados para los registros. La tabla es reiniciada |

* ***Lenguaje*** ***de*** ***definición*** ***de*** ***datos*** ***(DDL)***: proporciona órdenes para crear y modificar objetos de base de datos (BD, tablas, vistas, procedimientos, índices, etc.).

CREATE TABLE employees (

id INTEGER PRIMARY KEY,

first\_name VARCHAR(50) not null,

last\_name VARCHAR(75) not null,

mid\_name VARCHAR(50) not null,

dateofbirth DATE not null

);

|  |  |
| --- | --- |
| SELECT | Esta sentencia se utiliza para realizar consultas sobre los datos |
| INSERT | Con esta instrucción podemos insertar los valores en una base de datos |
| UPDATE | Sirve para modificar los valores de uno o varios registros. |
| DELETE | Se utiliza para eliminar las filas de una tabla. |

* ***Lenguaje de Manipulación de Datos (DML)***: proporciona órdenes para insertar, suprimir y modificar registros o filas de las tablas. También contempla la realización de consultas sobre la BD.

INSERT INTO employees (first\_name, last\_name, fname) VALUES ('John', 'Capita', 'xcapit00');

* ***Lenguaje*** ***de*** ***Control*** ***de*** ***Datos*** ***(DCL)***: permite establecer derechos de acceso de los usuarios sobre los distintos objetos de la base de datos. Lo forman las instrucciones GRANT y REVOKE.

Grant SELECT,INSERT,UPDATE,DELETE on Employee To User1

* ***Lenguaje*** ***de*** ***Control*** ***de*** ***Transacciones*** ***(DCL)***: gestiona las transacciones de BD. En SQL Server y otros RDBMS por defecto se gestionan de forma automática.

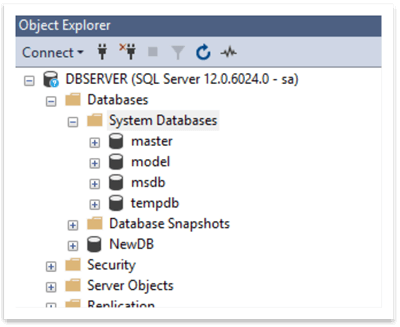
|  |  |
| --- | --- |
| COMMIT | Confirma una transacción |
| ROLLBACK | Revierte una transacción en caso de que ocurra algún error (no se puede deshacer si ha sido confirmada) |
| SET TRANSACTION | Especifica las características de la transacción. |

# **2.9 Creación de Bases de Datos con SQL**

La base de datos es una colección de objetos como tablas, vistas, procedimientos almacenados, funciones, triggers, etc. que guardan información e instrucciones para su consulta y modificación. En MS SQL Server, hay dos tipos de bases de datos disponibles:

* **Bases de datos del sistema:**

Las bases de datos del sistema se crean automáticamente cuando instalamos MS SQL Server. A continuación se muestra una lista de bases de datos del sistema −

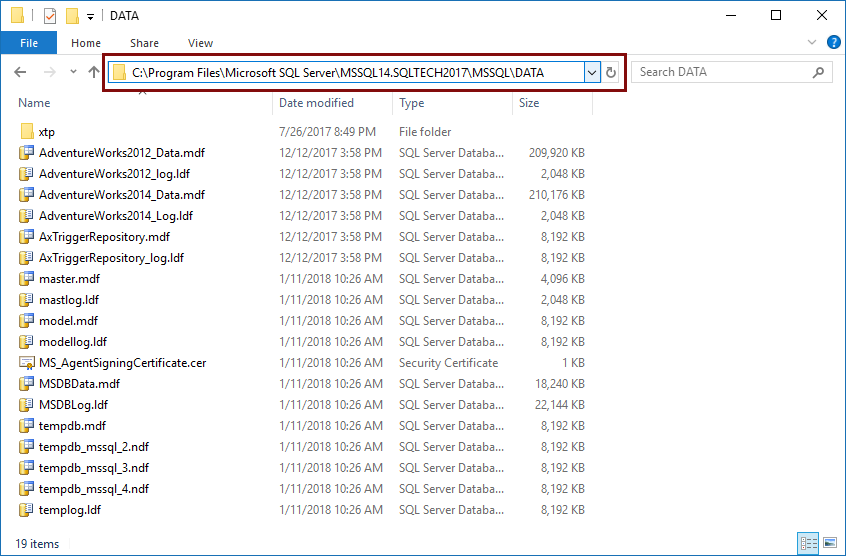


* + Master: La base de datos maestra registra toda la información de nivel de sistema para un sistema SQL Server. Esto incluye metadatos de toda la instancia, como cuentas de inicio de sesión, puntos de conexión, servidores vinculados y opciones de configuración del sistema
  + Model: se usa como plantilla cada vez que creas una base de datos. Por ejemplo, si se necesita que exista una tabla en particular en cada base de datos creada en una instancia de SQL Server, debe de estar en Model.
  + MSDB: El Agente SQL Server utiliza la base de datos msdb para programar alertas y trabajos,
  + Tempdb: La base de datos tempdb es un recurso global disponible para objetos temporales creados por los procesos internos que ejecutan SQL Server y los objetos temporales creados por usuarios o aplicaciones. se vuelve a crear cada vez que se reinicia SQL Server
  + Resource: Es una BD oculta de solo lectura Complementa la base de datos maestra porque SQL Server ahora se basa en ella. Contiene todos los objetos del sistema incluidos con SQL Server 2005 y versiones posteriores y permite realizar actualizaciones más rápidas del RDBMS
* **Bases de datos de usuarios:** Las bases de datos de usuario son creadas por usuarios (administradores, desarrolladores y evaluadores que tienen acceso para crear bases de datos).

## **2.9.1 Creación de Bases de Datos con T-SQL**

La instrucción CREATE DATABASE se utiliza para crear una nueva base de datos SQL.

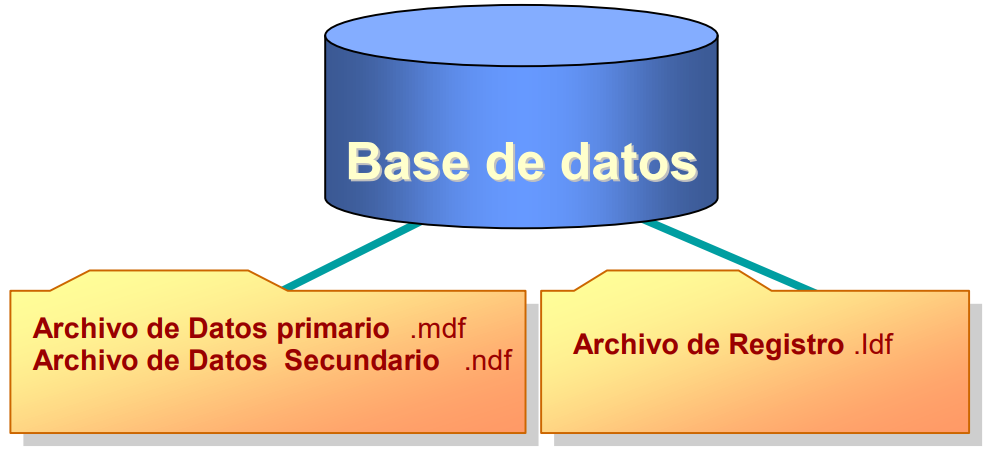
CREATE DATABASE *databasename*;

Por defecto, SQL Server guarda los archivos de la base de datos en su carpeta de instalación, específicamente en la carpeta DATA:

Si usamos la forma simple de la instrucción CREATE, sin ningún otro parámetro, el archivo principal de la base de datos toma el tamaño del archivo principal de la base de datos MODEL. El registro de transacciones toma el tamaño del archivo del registro de transacciones de la base de datos MODEL. Como no se ha especificado el parámetro MAXSIZE, los archivos pueden crecer hasta llenar todo el espacio disponible en el disco

**NOTA:** Puede usar el comando sp\_ helpDB para presentar información acerca de la base de datos especificada o de todas las bases de datos.

Sintaxis: SP\_HELPDB [Nombre de la base de datos]

Generalmente se crean tres (03) tipos de archivos para almacenar una base de datos.

* **Archivos principales:** Estos archivos contienen la información de inicio para la base de datos. Este archivo se utiliza también para almacenar datos. Cada base de datos tiene un único archivo principal. Tienen extensión **.mdf**.
* **Archivos secundarios:** Estos archivos contienen todos los datos que no caben en el archivo de datos principal. No es necesario que las bases de datos tengan archivos de datos secundarios si el archivo principal es lo suficientemente grande. Tienen extensión **.ndf**.
* **Registro de transacciones:** Estos archivos contienen la información de registro que se utiliza para recuperar la base de datos. Debe haber al menos un archivo de registro de transacciones. Su tamaño mínimo es 512 KB y su extensión es **.ldf**.

Es posible cambiar la ubicación de la BD y especificar el tamaño por defecto y máximo:

CREATE DATABASE Departamentos

ON

(

NAME = Departamentos\_Data,

FILENAME = 'D:\Data\Departamentos\_Data.mdf',

SIZE = 4,

MAXSIZE = 10,

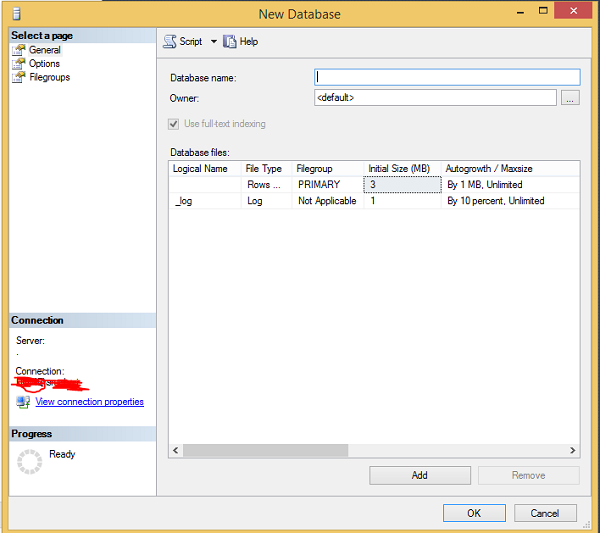
FILEGROWTH = 1

)

GO

## **2.9.2 Creación de Bases de Datos usando Management Studio**

Conéctese a la instancia de SQL Server y haga clic con el botón secundario en la carpeta de bases de datos. Haga clic en la nueva base de datos y aparecerá la siguiente pantalla:



Introduzca el campo nombre de la base de datos con el nombre de la base de datos (por ejemplo, para crear una base de datos con el nombre 'Testdb') y haga clic en Aceptar

# **2.10 Copias y Restauración de Bases de Datos**

**Para realizar un respaldo de la BD:**

Usando la interfaz gráfica de SQL Server Management Studio:

1. Inicie SQL Server Management Studio (SSMS) y conéctese a la instancia de SQL Server.
2. Expanda el nodo **Bases de datos** del **Explorador de objetos**.
3. Haga clic con el botón derecho en la base de datos, mantenga el puntero sobre **Tareas** y seleccione **Hacer copia de seguridad…**
4. En **Destino**, confirme que la ruta de acceso de la copia de seguridad es correcta. Si necesita cambiarla, seleccione **Quitar** para quitar la ruta de acceso existente y, luego, **Agregar** para escribir una nueva ruta de acceso. Puede usar el botón de puntos suspensivos para desplazarse a un archivo específico.
5. Seleccione **Aceptar** para realizar una copia de seguridad de la base de datos.

Usando T-SQL:

Un backup completo:

BACKUP DATABASE databasename

TO DISK = 'filepath';

Por ejemplo:

Creemos esta BD:

USE [master];

GO

CREATE DATABASE [SQLTestDB];

GO

USE [SQLTestDB];

GO

CREATE TABLE SQLTest (

ID INT NOT NULL PRIMARY KEY,

c1 VARCHAR(100) NOT NULL,

dt1 DATETIME NOT NULL DEFAULT GETDATE()

);

GO

USE [SQLTestDB]

GO

INSERT INTO SQLTest (ID, c1) VALUES (1, 'test1');

INSERT INTO SQLTest (ID, c1) VALUES (2, 'test2');

INSERT INTO SQLTest (ID, c1) VALUES (3, 'test3');

INSERT INTO SQLTest (ID, c1) VALUES (4, 'test4');

INSERT INTO SQLTest (ID, c1) VALUES (5, 'test5');

GO

SELECT \* FROM SQLTest;

GO

El backup se haría:

USE [master];

GO

BACKUP DATABASE [SQLTestDB]

TO DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\SQLTestDB.bak'

WITH NOFORMAT, NOINIT,

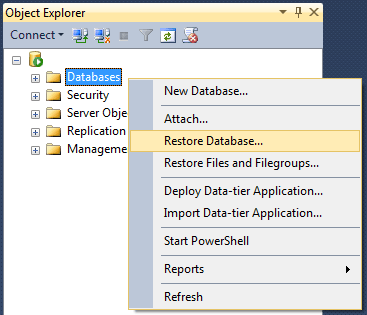
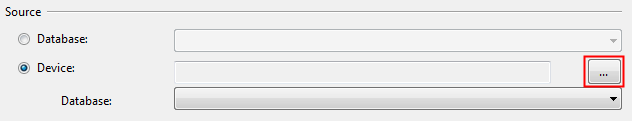
NAME = N'SQLTestDB-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10;

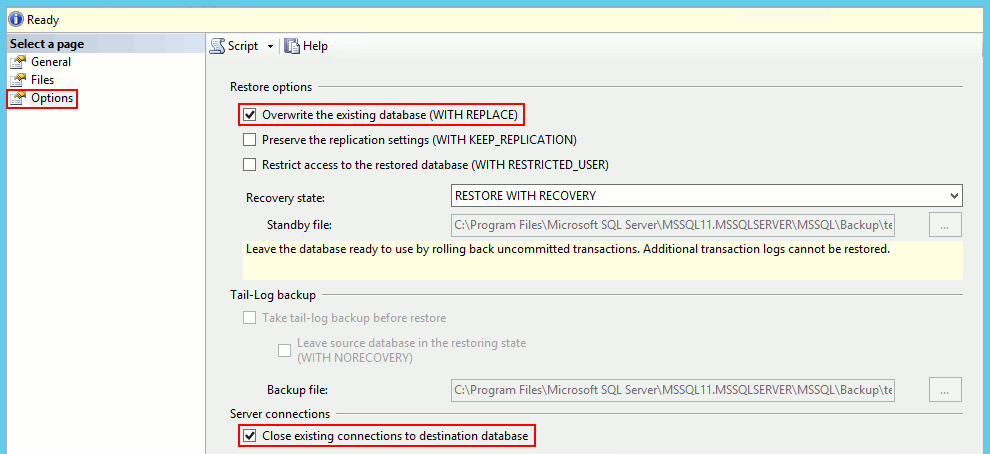
GO

* TO <backup\_device> [ ,...n ]: Especifica el dispositivo de copia de seguridad físico o lógico que se va a utilizar para la operación de copia de seguridad (DISK, TAPE, URL)
* WITH indica las opciones de copia de seguridad:
  + { NOFORMAT | FORMAT }: Dar formato o no a cualquier volumen de un conjunto de medios.
  + {NOINIT | INIT}: Controla si la operación de copia de seguridad anexa o sobrescribe los conjuntos de copias de seguridad existentes en el medio. El valor predeterminado es anexar al conjunto de copias de seguridad más reciente en el medio (NOINIT).
  + {NAME}: Especifica el nombre del conjunto de copia de seguridad. Los nombres pueden tener un máximo de 128 caracteres. Si no se especifica NAME, está en blanco.
  + { NOSKIP | SKIP }: Ignorar o no la fecha y la hora de expiración de los conjuntos de copia de seguridad en el medio antes de sobrescribirlos
  + { REWIND | NOREWIND }: Estas opciones solo se utilizan para dispositivos de cinta. Se omitirán si se utiliza otro tipo de dispositivo
  + { UNLOAD | NOUNLOAD }: especifica que la cinta se rebobina y descarga automáticamente al terminar la copia de seguridad o permanece cargada en la unidad de cinta
  + STATS: Muestra un mensaje de progreso cada cierto porcentaje

**Para restaurar una BD:**

Puede hacerse desde SSMS:

1. Inicie la sesión en el sistema en el que desea restaurar la base de datos.
2. Abra Microsoft SQL Server Management Studio.
3. En la barra de navegación, pulse con el botón derecho **Bases de datos** y pulse **Restaurar base de datos**
4. En la sección Origen, seleccione **Dispositivo** y pulse el botón de tres puntos.
5. En la ventana emergente que se abre, pulse **Añadir** y busque el archivo de copia de seguridad. Pulse **Aceptar**.
6. En el menú de navegación izquierdo, pulse **Opciones**.
7. En el panel derecho, seleccione **Sobrescribir base de datos existente (CON REEMPLAZAR)** y **Cerrar conexiones existentes con base de datos de destino.**



1. Pulse **Aceptar**.

Usando T-SQL

USE [master];

GO

RESTORE DATABASE [SQLTestDB]

FROM DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Backup\SQLTestDB.bak' WITH FILE = 1, NOUNLOAD, STATS = 5;

GO

# **2.11 Separación y Adjuntado de archivo de Bases de Datos**

Los archivos de datos y de registro de transacciones de una base de datos se pueden desasociar y, a continuación, volver a adjuntarse a la misma instancia de SQL Server. Desasociar y adjuntar una base de datos es útil si desea cambiar la base de datos a una instancia diferente de SQL Server en el mismo equipo o mover la base de datos.

Separación usando T-SQL

1. Conéctese con el Motor de base de datos.
2. En la barra Estándar, seleccione Nueva consulta.
3. Copie y pegue el ejemplo siguiente en la ventana de consulta y seleccione Ejecutar. En este ejemplo se separa la base de datos AdventureWorks con la opción skipchecks establecida en true.

EXEC sp\_detach\_db 'AdventureWorks2012', 'true';

Adjuntado usando T-SQL

CREATE DATABASE MyAdventureWorks

ON (FILENAME = 'C:\MySQLServer\AdventureWorks\_Data.mdf'),

(FILENAME = 'C:\MySQLServer\AdventureWorks\_Log.ldf')

FOR ATTACH;

# **2.12 Tipos de Datos**

El tipo de datos de una columna define qué valor puede contener la columna: entero, carácter, dinero, fecha y hora, binario, etc.

Cada columna de una tabla de base de datos debe tener un nombre y un tipo de datos.

Un desarrollador de SQL debe decidir qué tipo de datos se almacenarán dentro de cada columna al crear una tabla. El tipo de datos es una guía para que SQL comprenda qué tipo de datos se esperan dentro de cada columna, y también identifica cómo SQL interactuará con los datos almacenados.

Nota: Los tipos de datos pueden tener diferentes nombres en diferentes bases de datos. E incluso si el nombre es el mismo, ¡el tamaño y otros detalles pueden ser diferentes!

En SQL Server, tenemos para los tipos string:

| Tipo de dato | Descripción | Tamaño máximo | Almacenamiento |
| --- | --- | --- | --- |
| char(n) | Cadena de caracteres de ancho fijo | 8.000 caracteres | Ancho definido |
| varchar(n) | Cadena de caracteres de ancho variable | 8.000 caracteres | 2 bytes + número de caracteres |
| varchar(máx.) | Cadena de caracteres de ancho variable | 1.073.741.824 caracteres | 2 bytes + número de caracteres |
| text | Cadena de caracteres de ancho variable | 2 GB de datos de texto | 4 bytes + número de caracteres |
| nchar | Cadena Unicode de ancho fijo | 4.000 caracteres | Ancho definido x 2 |
| nvarchar | Cadena Unicode de ancho variable | 4.000 caracteres |  |
| nvarchar(máx.) | Cadena Unicode de ancho variable | 536.870.912 caracteres |  |
| ntext | Cadena Unicode de ancho variable | 2 GB de datos de texto |  |
| binary(n) | Cadena binaria de ancho fijo | 8.000 bytes |  |
| varbinary | Cadena binaria de ancho variable | 8.000 bytes |  |
| varbinary (máx.) | Cadena binaria de ancho variable | 2 GB |  |
| image | Cadena binaria de ancho variable | 2 GB |  |

Para los tipos numéricos, tenemos:

|  |  |  |
| --- | --- | --- |
| **Tipo de datos** | **Descripción** | **Almacenamiento** |
| bit | Entero que puede ser 0, 1 o NULL |  |
| tinyint | Permite números enteros de 0 a 255 | 1 byte |
| smallint | Permite números enteros entre -32,768 y 32,767 | 2 bytes |
| int | Permite números enteros entre -2,147,483,648 y 2,147,483,647 | 4 bytes |
| bigint | Permite números enteros entre -9,223,372,036,854,775,808 y 9,223,372,036,854,775,807 | 8 bytes |
| decimal (p, s) | Números de escala y precisión fijos.  Permite números de -10 ^ 38 +1 a 10 ^ 38 -1.  El parámetro p indica el número total máximo de dígitos que se pueden almacenar (tanto a la izquierda como a la derecha del punto decimal). p debe ser un valor de 1 a 38. El valor predeterminado es 18.  El parámetro s indica la cantidad máxima de dígitos almacenados a la derecha del punto decimal. s debe ser un valor de 0 a p. El valor predeterminado es 0 | 5-17 bytes |
| numeric (p, s) | Números de escala y precisión fijos.  Permite números de -10 ^ 38 +1 a 10 ^ 38 -1.  El parámetro p indica el número total máximo de dígitos que se pueden almacenar (tanto a la izquierda como a la derecha del punto decimal). p debe ser un valor de 1 a 38. El valor predeterminado es 18.  El parámetro s indica la cantidad máxima de dígitos almacenados a la derecha del punto decimal. s debe ser un valor de 0 a p. El valor predeterminado es 0 | 5-17 bytes |
| smallmoney | Datos monetarios de -214,748.3648 a 214,748.3647 | 4 bytes |
| money | Datos monetarios de -922,337,203,685,477.5808 a 922,337,203,685,477.5807 | 8 bytes |
| float (n) | Datos del número de precisión flotante desde -1.79E + 308 a 1.79E + 308.  El parámetro n indica si el campo debe contener 4 u 8 bytes. float (24) contiene un campo de 4 bytes y float (53) contiene un campo de 8 bytes. El valor predeterminado de n es 53. | 4 u 8 bytes |
| real | Datos numéricos de precisión flotante desde -3.40E + 38 a 3.40E + 38 | 4 bytes |

Para los tipos fecha:

|  |  |  |
| --- | --- | --- |
| **Tipo de datos** | **Descripción** | **Almacenamiento** |
| datetime | Del 1 de enero de 1753 al 31 de diciembre de 1999, con una precisión de 3,33 milisegundos | 8 bytes |
| datetime2 | Desde el 1 de enero de 0001 hasta el 31 de diciembre de 1999, con una precisión de 100 nanosegundos | 6-8 bytes |
| smalldatetime | Del 1 de enero de 1900 al 6 de junio de 2079 con una precisión de 1 minuto | 4 bytes |
| date | Almacenar una fecha solamente. Del 1 de enero de 0001 al 31 de diciembre de 9999 | 3 bytes |
| time | Almacenar un tiempo solo con una precisión de 100 nanosegundos | 3-5 bytes |
| datetimeoffset | Lo mismo que datetime2 con la adición de un desplazamiento de zona horaria | 8-10 bytes |
| timestamp | Almacena un número único que se actualiza cada vez que se crea o modifica una fila. El valor de la marca de tiempo se basa en un reloj interno y no corresponde a tiempo real. Cada tabla puede tener una sola variable de marca de tiempo |  |

# **2.13 Estado de Bases de Datos**

Una base de datos siempre se encuentra en un estado específico. Esos estados pueden ser ONLINE, OFFLINE, SUSPECT, etc. Se puede comprobar el estado de la base de datos, mediante alguna sentencia T-SQL, y también mediante el entorno gráfico (SSMS).

Con T-SQL utilizaremos la vista de sistema llamada sys.databases:

use pruebas

go

select name, state\_desc from sys.databases where name = 'pruebas'

|  |  |
| --- | --- |
| **Estado** | **Definición** |
| **ONLINE** | La base de datos está disponible para su acceso. El grupo de archivos principal está en línea. |
| **OFFLINE** | La base de datos no está disponible. Una base de datos pasa a estar sin conexión por la acción explicita del usuario y permanece sin conexión hasta que el usuario toma otra acción. Por ejemplo, la base de datos puede dejarse sin conexión para mover un archivo a un nuevo disco. La base de datos se vuelve a poner en línea una vez completado el traslado |
| **RESTORING** | Uno o varios archivos del grupo de archivos principal se está restaurando, o uno o varios archivos secundarios se están restaurando sin conexión. La base de datos no está disponible |
| **RECOVERING** | Se está recuperando la base de datos. El proceso de recuperación es un estado transitorio, la base de datos se pone automáticamente en línea si la recuperación tiene éxito. Si la recuperación no tiene éxito pasará a ser sospechosa. La base de datos no está disponible |
| **RECOVERY PENDING** | SQL Server ha encontrado un error relacionado con un recurso durante la recuperación. La base de datos no está dañada, pero pueden faltar archivos o bien limitaciones de recursos del sistema pueden estar impidiendo que se inicie. La base de datos no está disponible. Se necesita una acción adicional por parte del usuario para resolver el error y permitir que se complete el proceso de recuperación. |
| **SUSPECT** | Como mínimo un grupo de archivos principal es sospechoso y puede estar dañado. La base de datos no se puede recuperar durante el inicio de SQL Server. La base de datos no está disponible. Se requiere una acción adicional por parte del usuario para resolver el problema. |
| **EMERGENCY** | El usuario ha cambiado la base de datos y ha establecido el estado en EMERGENCY. La base de datos está en modo de usuario único y se puede reparar o restaurar. La base de datos está marcada como READ\_ONLY, el registro está deshabilitado y el acceso está limitado a miembros del rol **sysadmin.**EMERGENCY se utiliza principalmente para solucionar problemas. Por ejemplo, una base de datos marcada como sospechosa se puede establecer en estado EMERGENCY. Esto puede permitir al administrador del sistema acceso de solo lectura a la base de datos. Solo los miembros del rol **sysadmin** pueden establecer una base de datos en status EMERGENCY |

|  |  |
| --- | --- |
| ONLINE |  |
| OFFLINE |  |
| EMERGENCY |  |
| SUSPECT |  |
| RESTORING |  |

# **2.14 Integridad de Datos con SQL**

La integridad de los datos es el mantenimiento y la garantía de la precisión y consistencia de los datos durante todo su ciclo de vida y es un aspecto crítico para el diseño, implementación y uso de cualquier sistema que almacene, procese o recupere datos

## **2.14.1 Integridad de Dominio**

Se conoce como el dominio de un atributo al conjunto de valores aceptables para dicho atributo. La integridad de dominio establece qué condiciones deben cumplir los valores a insertar en una columna.

La integridad de dominio se define mediante reglas de validación, valores predeterminados, conjunto de valores permitidos en la columna (llave foránea), tipo y formato de los datos.

## **2.14.2 Integridad de Entidad**

Una tabla almacena los datos de cada una de las ocurrencias de una entidad. La entidad (o tabla) requiere que todas sus filas sean únicas. Esto se garantiza definiendo para cada fila de la entidad un identificador único (llave primaria).

## **2.14.3 Integridad Referencial**

La integridad referencial garantiza que la relación entre la llave primaria (en la tabla referenciada) y la llave foránea (en la tabla de referencia) siempre se mantiene. Una fila en una tabla referenciada no puede anularse, ni cambiar su valor de la llave primaria, si una llave foránea se refiere a la fila

**Sintaxis de SQL y operadores:**

La sintaxis del lenguaje SQL es un conjunto único de reglas y directrices, que no distingue entre mayúsculas y minúsculas para escribir sus instrucciones más no los valores de los registros que dependen del juego de caracteres que use el motor de BD. Su sintaxis está definida y mantenida por las normas ISO y ANSI.

Los siguientes son algunos de los puntos más importantes sobre la sintaxis SQL que deben recordarse:

* Puede escribir las palabras clave de SQL tanto en mayúsculas como en minúsculas, pero escribir las palabras clave sql en mayúsculas mejora la legibilidad de la consulta SQL.
* Las instrucciones SQL o la sintaxis dependen de las líneas de texto. Podemos colocar una sola instrucción SQL en una o varias líneas de texto.
* Puede realizar la mayor parte de la acción en una base de datos con instrucciones SQL.
* La sintaxis SQL depende del álgebra relacional y del cálculo relacional de la tupla.

Cada instrucción SQL comienza con cualquiera de las palabras clave SQL y termina con el punto y coma (;). El punto y coma se utiliza en el SQL para separar las múltiples instrucciones.

**Operadores de SQL:**

Los administradores y usuarios de base de datos utilizan consultas SQL para manipular y acceder a los datos de las tablas y vistas de la base de datos. La manipulación y recuperación de los datos se realizan con la ayuda de palabras y caracteres reservados, que se utilizan para realizar operaciones aritméticas, operaciones lógicas, operaciones de comparación, operaciones compuestas, etc.

En la siguiente tabla, los operadores en la parte superior tienen una prioridad alta, y los operadores que aparecen en la parte inferior tienen una prioridad baja.

|  |  |
| --- | --- |
| **Símbolos de operador de SQL** | **Operadores** |
| \*\* | Operador de exponenciación |
| +, - | Operador de identidad, operador de negación |
| \*, / | Operador de multiplicación, Operador de división |
| +, -, || | Operador de suma (más), operador de resta (menos), operador de concatenación de cadenas |
| =, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN | Operadores de comparación |
| NOT | Operador de negación lógica |
| && ó AND | Operador de conjunción |
| OR | Operador de inclusión |

Algunos ejemplos:

* El operador **EQUAL** (=). es muy utilizado en consultas SQL. El operador equal en SQL solo muestra los datos que coinciden con el valor especificado en la consulta.

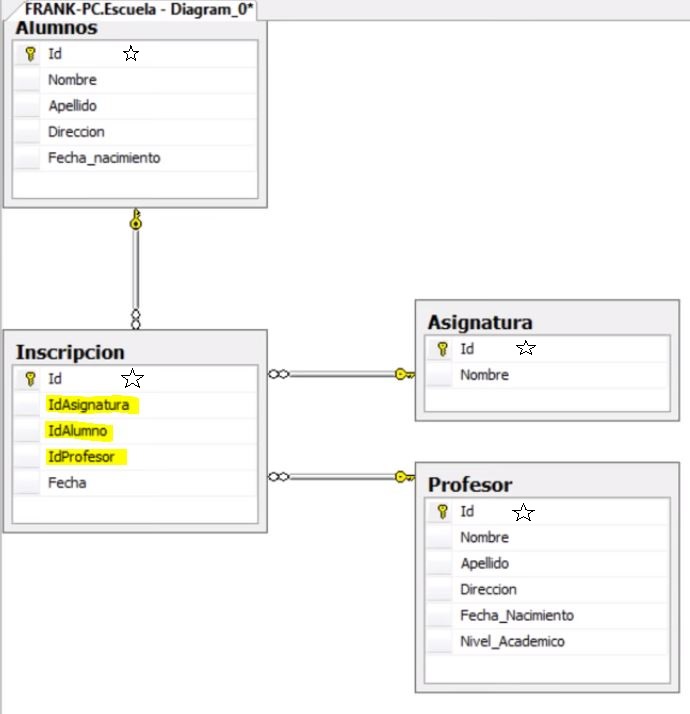
SELECT \* FROM Employee\_details WHERE Emp\_Salary = 30000;

* El operador **BETWEEN** en SQL muestra el registro dentro del rango mencionado en la consulta SQL. Este operador funciona con los números, caracteres y operandos de fecha/hora.
* El operador ALL en SQL compara el valor especificado con todos los valores de una columna de la subconsulta en la base de datos SQL.Este ejemplo consta de una tabla Employee\_details, que tiene tres columnas Emp\_Id, Emp\_Name, Emp\_Salary y Emp\_City.

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp Id** | **Emp Name** | **Emp Salary** | **Emp City** |
| 201 | Abhay | 25000 | Gurgaon |
| 202 | Ankit | 45000 | Delhi |
| 203 | Bheem | 30000 | Jaipur |
| 204 | Ram | 29000 | Mumbai |
| 205 | Sumit | 40000 | Calcuta |

Si queremos acceder al id de empleado y nombres de empleado de aquellos empleados de la tabla cuyos salarios son mayores que el salario de los empleados que viven en la ciudad de Jaipur, entonces tenemos que escribir la siguiente consulta en SQL.

SELECCIONE Emp\_Id, Emp\_Name DESDE Employee\_details DONDE Emp\_Salary > TODO ( SELECCIONE Emp\_Salary DESDE Employee\_details DONDE Emp\_City = Jaipur)e

Ejemplo de creación de BD

CREATE DATABASE Escuela

go

use Escuela

go

/\* Creando tabla Alumno, asignatura y profesor \*/

Create table Alumnos(

Id char(8) primary key,

Nombre varchar(20) not null,

Apellido varchar(20) not null,

Direccion varchar(50),

Fecha\_nacimiento char(8)

);

Create table Asignatura(

Id char(8) primary key,

Nombre varchar(20) not null

);

Create table Profesor(

Id char(8) primary key,

Nombre varchar(20) not null,

Apellido varchar(20) not null,

Direccion varchar(50),

Fecha\_nacimiento char(8),

Nivel\_Academico varchar (20)

);

/\* Creando tabla Inscripción con campos para llaves foraneas \*/

Create table Inscripcion(

Id char(8) primary key,

IdAsignatura char(8) not null,

IdAlumno char(8) not null,

IdProfesor char(8) not null,

Fecha char(8),

CONSTRAINT fk\_Asignatura FOREIGN KEY (IdAsignatura) REFERENCES Asignatura (Id),

CONSTRAINT fk\_Alumno FOREIGN KEY (IdAlumno) REFERENCES Alumnos (Id),

CONSTRAINT fk\_Profesor FOREIGN KEY (IdProfesor) REFERENCES Profesor (Id)

);

# **2.15 Modificación del Esquema de Base de Datos con SQL**

El termino ALTER TABLE se utiliza para agregar, eliminar o modificar columnas de una tabla existente.

Ejemplos:

Agregar el campo trabajo a una tabla Customers:

* ALTER TABLE Customers ADD Trabajo nvarchar(15);

Eliminar la columna de Fax de la table Customers:

* ALTER TABLE Customers DROP COLUMN Fax;

Cambiar el tipo de dato de la columna Fax de la tabla Customers a int:

* ALTER TABLE Customers ALTER COLUMN Fax int;

# **2.17 Gestión de Registros (Insert-Update-Delete)**

**Para agregar datos a una tabla:**

Es *necesario* hacer uso de la instrucción **Insert** **into**, que es uno de los comandos más usado del código Sql. Para insertar los registros lo podemos hacer de uno en uno, o podemos agregar varios registros a través de una misma instrucción.

Usando múltiples inserciones:

INSERT INTO Alumnos (Id, Nombre, Apellido, Direccion, Fecha\_nacimiento)

VALUES ('0101', 'Franklin', 'Garcia', 'avenida 01', '12/01/80')

INSERT INTO Alumnos VALUES ('0102', 'Juan', 'Hernandez', 'avenida 01', '12/01/80')

INSERT INTO Alumnos (Id, Nombre, Apellido) VALUES ('0103', 'Juan', 'Perez')

Usando un solo insert:

INSERT INTO Alumnos (Id, Nombre, Apellido, Direccion, Fecha\_nacimiento)

VALUES

('0104', 'Franklin2', 'Garcia', 'avenida 01', '12/01/80'),

('0105', 'Franklin3', 'Garcia', 'avenida 01', '12/01/80')

Siguiendo el ejemplo del diagrama mostrado antes hay que insertar registros en las tablas Asignatura y profesor:

INSERT INTO Asignatura (Id, Nombre) VALUES ('BD01', 'Base de Datos 1')

INSERT INTO Profesor VALUES ('PF01', 'Antonio','Perez', 'avenida 01', '12/01/80','Licenciado')

Se procede a insertar los datos en la tabla inscripción:

INSERT INTO Inscripcion VALUES ('INS01', 'BD01','0101', 'PF01', '12/01/80')

**Para eliminar registros de una tabla:**

La instrucción SQL **DELETE** se utiliza para eliminar filas de una tabla. Por lo general, la instrucción DELETE quita uno o más registros de una tabla. Si no se especifica una condición **WHERE**, eliminará todos los registros de la tabla sin reiniciarla. Cuando se especifica esta condición elimina todos aquellos registros donde se cumpla.

DELETE FROM table\_name [WHERE condition];

|  |
| --- |
| Advertencia |
| Utiliza siempre la instrucción DELETE con la condición WHERE para no perder todos los registros de la tabla |

Por ejemplo, para remover un registro de la tabla alumnos del ejemplo anterior, tendríamos:

DELETE FROM ALUMNOS WHERE ID=<Id\_Code>

**Para actualizar o modificar registros de una tabla:**

La instrucción SQL **UPDATE** se utiliza para cambiar los datos de los registros en las tablas. Es recomendable usar una condición para elegir las filas a actualizar. Para especificar la condición, utilizamos la cláusula **WHERE**.

La sintaxis es como sigue:

UPDATE table\_name SET [column\_name1= value1,... column\_nameN = valueN] [WHERE condicion]

Por ejemplo, para actualizar un registro de la tabla alumnos del ejemplo anterior, tendríamos:

UPDATE ALUMNOS SET APELLIDO='Gutiérrez' WHERE ID=<Id\_Code>

Y para actualizar múltiples campos:

UPDATE ALUMNOS SET NOMBRE='Juan', APELLIDO='Gutiérrez' WHERE ID=<Id\_Code>

|  |
| --- |
| Advertencia |
| Tenga cuidado al actualizar los registros. Si omite la cláusula WHERE, ¡TODOS los registros se actualizarán! |

También es posible crear instrucciones más complejas para actualizar o modificar registros mediante **subconsultas** usando la condición **EXISTS**. Por ejemplo:

UPDATE suppliers

SET supplier\_name = (SELECT customers.name

FROM customers

WHERE customers.customer\_id = suppliers.supplier\_id)

WHERE EXISTS (SELECT customers.name

FROM customers

WHERE customers.customer\_id = suppliers.supplier\_id);

O bien esta:

DELETE FROM contacts

WHERE EXISTS (SELECT \*

FROM employees

WHERE employees.last\_name = contacts.last\_name);

**Para actualizar o modificar la estructura de una tabla:**

Es muy posible que después de crear una tabla, a medida que comience a usarla, descubra que ha olvidado mencionar cualquier columna o restricción o ha especificado un nombre incorrecto para la columna.

En tal situación, puede utilizar la instrucción ALTER TABLE para modificar o cambiar una tabla existente agregando, cambiando o eliminando una columna de la tabla.

* ALTER TABLE table\_name ADD column\_name data\_type constraints;
  + ALTER TABLE Cars ADD Car\_Color Varchar(50);
* ALTER TABLE table\_name DROP Column column\_name ;
  + ALTER TABLE Cars DROP COLUMN Car\_Color ;
* ALTER TABLE table\_name RENAME COLUMN old\_name to new\_name;
  + ALTER TABLE Cars RENAME COLUMN Car\_Color to Colors;
* ALTER TABLE table\_name

ADD CONSTRAINT FK\_table1\_table2 FOREIGN KEY (column\_table2)

REFERENCES Table1(id);

**Para eliminar la BD:**

Use master;

Go

Drop database db\_name;

Go

Referencias

<https://www.tutorialrepublic.com/sql-tutorial/>

<https://www.javatpoint.com/sql-tutorial>

<https://www.w3schools.com/sql/default.asp>

<https://www.techonthenet.com/sql_server/index.php>

<https://www.sqlservertutorial.net/sql-server-basics/>

<https://codigosql.top/sql-server/>