

Componente formativo

**Trabajo con datos**

**Breve descripción:**

Este componente formativo tiene, como objetivo, la comprensión del uso de los sistemas de bases de datos y el modelo de diseño establecido para el almacenamiento y resguardo de la información, suministrada a través del *software*.

**Área ocupacional:**

Ciencias naturales, aplicadas y relacionadas

**Noviembre 2022**

Tabla de contenido

[Introducción 3](#_Toc129335944)

[1. Gestores de base de datos y formatos de archivos 4](#_Toc129335945)

[2. Lenguajes de sistemas administradores de base de datos (DBMS) y lenguaje estructurado de consulta (SQL) 10](#_Toc129335946)

[3. Tipos de datos 15](#_Toc129335947)

[4. Lenguajes de definición de datos DDL 20](#_Toc129335948)

[5. Restricciones de integridad 25](#_Toc129335949)

[6. Lenguajes de manipulación de datos DML 29](#_Toc129335950)

[7. Construcción de base de datos NoSQL 34](#_Toc129335951)

[Síntesis 37](#_Toc129335952)

[Material complementario 38](#_Toc129335953)

[Glosario 39](#_Toc129335954)

[Referencias bibliográficas 41](#_Toc129335955)

[Créditos 42](#_Toc129335956)

Introducción

Apreciado aprendiz, bienvenido a este componente formativo donde abordaremos la temática relacionada con los sistemas de almacenamiento como son las bases de datos, las cuales permiten que un sistema de información pueda resguardar los datos y la información, desde un entorno gráfico.

En el siguiente video conocerá, de forma general, la temática que se estudiará a lo largo del componente formativo:

1. Introducción – Trabajo con datos



[**Enlace de reproducción del video**](https://www.youtube.com/watch?v=MJUopc_lgn0)

|  |
| --- |
| **Síntesis del video: Introducción – Trabajo con datos** |
| Apreciado aprendiz, bienvenido a esta nueva experiencia de aprendizaje y conocimiento donde abordaremos varias temáticas como:  a. Manipulación de los datos en una base de datos.  b. Lenguaje DDL.  c. Lenguaje SQL.  d. Integridad referencial.  e. Manejo de los diferentes tipos de datos que puede manejar un motor de base de datos  f. Restricciones de la integridad referencial.  g. Consultas de información teniendo en cuenta la necesidad funcional del usuario que manipula el sistema de información. |

# Gestores de base de datos y formatos de archivos

Para iniciar, aprenderemos que los gestores de bases de datos son aplicativos que permiten realizar las tareas de administración y mantenimiento de datos, de manera sencilla y eficiente. Se caracterizan por tener interfaces amigables y funciones establecidas para que el proceso de manutención de las mismas, sea sencillo. Actualmente, en el mercado, existen muchos tipos de gestores de bases de datos lo que implica que, en la mayoría de las ocasiones, se tenga que debatir cuál es el mejor para implementar en el ámbito empresarial; lo que muchas veces no se toma en cuenta, es el tipo de motor de bases de datos, que se desea operar y la infraestructura que se utilizará.

Por este motivo, debemos ubicarlos en diferentes categorías, las cuales abordaremos a continuación:

1. **Relacionales:** Este tipo de gestores, trabajan con las bases de datos más utilizadas actualmente y que también son denominadas como bases de datos tradicionales. Se basan en una estructura de conexiones lógicas llamadas relaciones, las cuales tienen como objetivo, conectar las diferentes entidades o tablas de la base de datos, lo que permite tener un control sobre la forma como se distribuyen los datos, mediante un concepto llamado integridad referencial, que establece unas reglas de conexión y restricción de los datos conectados. Estas relaciones son: relación uno-muchos, relación uno-uno.

El gestor de bases de datos relacional, debe permitir la creación del modelo relación o también llamado MER, para visualizar las diferentes relaciones que puede contener y la restricción que tiene cada una de ellas.

1. **No relacionales:** Los gestores de bases de datos no relacionales, trabajan bajo el mismo esquema y filosofía de las bases de datos no relacionales; tienen una peculiar manera de conectar los datos y la información que se encuentra almacenada y mediante líneas de comando, es decir, no tienen una restricción mediante configuración, sino que la realiza el programador a través de códigos, o sea, las consultas SQL y lenguaje DML son los encargados de establecer dicha lógica de conexión entre las diferentes entidades. A lo largo de los tiempos se ha establecido que estas bases de datos tienen algunas ventajas, tales como la independencia de los datos, ayudan mucho al tema de realización de reportes, le dan mayor versatilidad y creatividad al programador o administrador de las mismas. Por otro lado, es un poco complicado su mantenimiento, al no estar relacionados los datos por configuración o integridad referencial, al realizar actualizaciones o cambios masivos; toca realizarlo manualmente, lo que puede hacer que su mantenimiento sea más costoso y tome más tiempo.
2. **Orientado a objetos:** Es un sistema de gestión de base de datos mediante el cual representamos la información en forma de objetos que son utilizados en programación orientada a objetos.

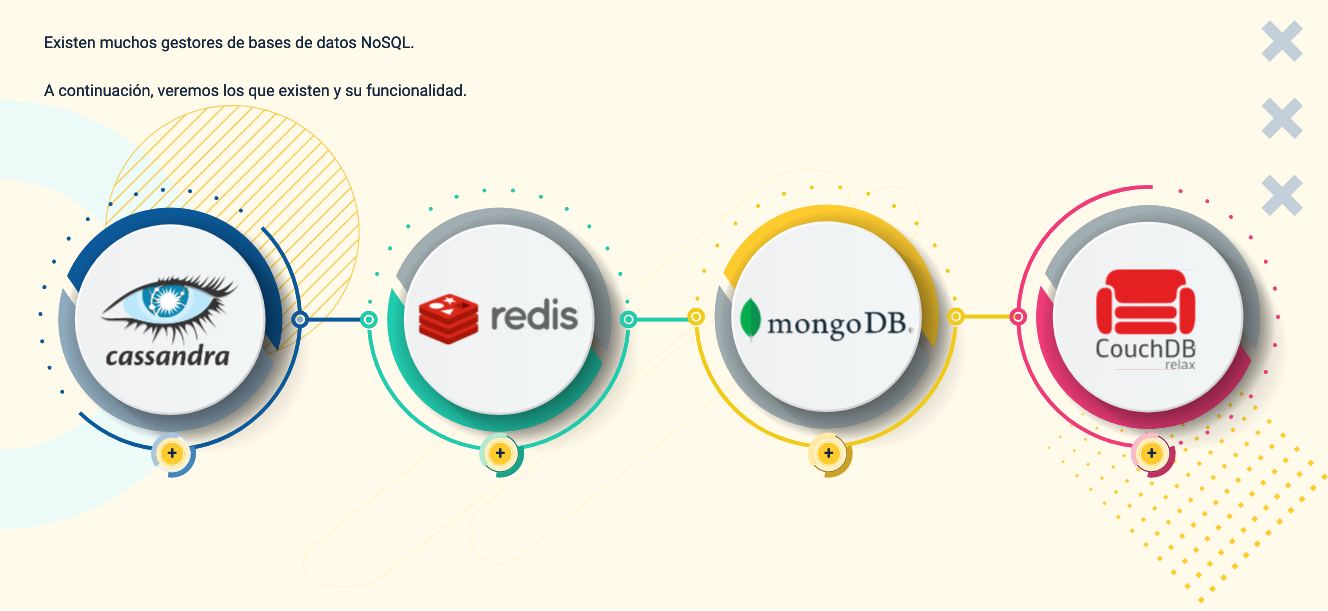
La investigación y el desarrollo de la base de datos orientada a objetos fue impulsada en gran medida por su alto rendimiento en almacenamiento de datos y los requerimientos de acceso que tenían entornos de soporte de diseño tales como el CAD (diseño asistido por computadora). La base de datos orientada a objetos ha encontrado su nicho de aplicación en áreas tales como la ingeniería, las bases de datos espaciales, las telecomunicaciones y en áreas científicas como la física de alta energía y la biología molecular.

Otro conjunto de bases de datos orientadas a objetos se centra en el uso integrado en dispositivos, paquetes de software y sistemas en tiempo real; es decir, permiten la creación de datos y su expansión depende de lo que requiera el sistema. Una manera de ver este tipo de sistemas de almacenamiento es el caso de los sistemas de inteligencia artificial, los cuales pueden generar nuevas instancias y conexiones de acuerdo a la información que van adquiriendo, haciendo de ellos un sistema de almacenamiento sin límites.

**Gestión de la base de datos**

Existen muchos gestores de bases de datos NoSQL.

A continuación, veremos los gestores que existen y su funcionalidad:



1. **CASSANDRA:** Base de datos creada por Apache del tipo clave-valor. Dispone de un lenguaje propio para realizar consultas CQL (Cassandra Query Language). Cassandra es una aplicación Java, por lo que puede correr en cualquier plataforma que cuente con la JVM. Es multiplataforma.
2. **REDIS:** Base de datos creada por Salvatore Sanfilippo y Pieter Noordhuis y está apoyada por VMWare. Se trata de una base de datos del tipo clave-valor. Se puede imaginar como un array gigante en memoria para almacenar datos que pueden ser cadenas, hashes, conjuntos de datos o listas. Tiene la ventaja de que sus operaciones son atómicas y persistentes. Sin embargo, Redis no permite realizar consultas, solo se pueden insertar y obtener datos, además de las operaciones comunes sobre conjuntos (diferencia, unión e inserción). Creado en ANSI C, por lo tanto, es compatible y funciona sin problemas en sistemas Unix, Linux y sus derivados, Solaris, OS/X; sin embargo, no existe soporte oficial para plataformas Windows
3. **MongoDB:** Base de datos creada por MongoDB Inc. (anteriormente 10gen), del tipo orientada a documentos, de esquema libre, es decir, que cada entrada puede tener un esquema de datos diferente que nada tenga que ver con el resto de registros almacenados. Es bastante rápida a la hora de ejecutar sus operaciones, ya que maneja datos binarios. En poco tiempo, MongoDB se ha convertido en una de las bases de datos NoSQL favoritas por los desarrolladores. Está escrita en lenguaje C++. Para el almacenamiento de la información, utiliza un sistema propio de documento conocido con el nombre BSON, que es una evolución del formato JSON pero con la peculiaridad de que puede almacenar datos representados de forma binaria. Está disponible para los sistemas operativos Windows, Linux, OS/X y Solaris.
4. **CouchDB:** Sistema creado por Apache y escrito en el lenguaje Erlang que funciona en la mayoría de sistemas POSIX (multiplataforma), incluyendo GNU/Linux y [[macOS|OS/X], además de sistemas Windows. Como características más importantes cabe destacar el uso de RESTful HTTP API como interfaz y JavaScript como principal lenguaje de interacción. Para el almacenamiento de los datos se utilizan archivos JSON. Permite la creación de vistas, que son el mecanismo que permite la combinación de documentos para retornar valores de varios documentos, es decir, CouchDB permite la realización de las operaciones JOIN típicas de SQL.

Recordemos que un SGBD o Sistema de Gestión de Bases de Datos, permite que se puedan realizar las tareas de administración de las bases de datos, de una manera mucho más cómoda y eficiente. Estos poseen interfaces de administración que ayudan en los procesos de manejo de los sistemas de almacenamiento de bases de datos, al igual que el mantenimiento de los datos y las copias de seguridad. Es importante tener en cuenta que, dependiendo de las necesidades o requerimientos del sistema, así será la selección y uso del sistema de gestión de bases de datos.

**Formatos de archivos**

Por otra parte, cuando nos referimos a formatos de archivos estamos hablando de una estructura específica que manejan para realizar las operaciones de almacenamiento y forma de acceso a la información contenida, es decir, cada formato de archivo tiene una estructura propia para realizar el almacenamiento de la información dentro de los mismos. Actualmente existen muchos formatos y tipos de archivos, los cuales, de acuerdo con la necesidad de acceso de la información o requerimientos de la aplicación, permiten guardar información que puede ser posteriormente consultada por el usuario que manipula la aplicación y que requiere utilizar dicha información.

Es importante tener en cuenta que cuando hablamos de almacenamiento en archivos nos referimos al almacenamiento en disco, lo que quiere decir que es información permanente en disco duro.

Conozcamos algunos de estos formatos:

1. **JavaScript Object Notation (JSON):** es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript; es comúnmente utilizado para transmitir datos en aplicaciones web. Estos archivos también son utilizados para el intercambio de información en aplicaciones web, porque brindan una estructura robusta al momento de organizar la información para, después, poder ser accedida por el usuario a través de la aplicación; otra característica particular es la cantidad de datos que pueden almacenar sin necesidad de sacrificar grandes volúmenes de espacio en disco duro, es decir, son muy livianos tanto en su ejecución, como en su peso y tamaño.
2. **Los archivos con extensión TXT:** son archivos que se consideran planos, ya que no contienen ningún tipo de formato específico para el almacenamiento de la información; a pesar de esto son utilizados para el almacenamiento de información, ya que ocupan muy poco espacio de almacenamiento en disco. Otra de sus ventajas es la compatibilidad que tienen con prácticamente cualquier sistema operativo, es decir, con los editores de texto que tienen los principales sistemas operativos y tipos de letras de estos mismos; por lo tanto pueden ser editados en prácticamente cualquier sistema. Otra de las particularidades que tienen estos tipos de archivos es que se adaptan a cualquier tipo de estructura, es decir, no llevan un orden específico de tabulación en el documento, pero sí debemos destacar la ventaja de que son muy livianos y tienen compatibilidad con todos los sistemas operativos y aplicaciones del mundo.
3. **XML:** Es un lenguaje de marcado similar a HTML. Significa *Extensible Markup Language* (Lenguaje de Marcado Extensible) y es una especificación de W3C como lenguaje de marcado de propósito general. Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet; como bien sabemos, es muy utilizado en la estructura de archivos para páginas web y maquetación de las mismas, que acompañado con el lenguaje de etiquetado HTML, permiten generar la estructura de las páginas.

Ampliemos la información sobre las bases de datos y, para hacerlo, lo invitamos a ver el siguiente video:

1. Bases de datos



[**Enlace de reproducción del video**](https://www.youtube.com/watch?v=fkCvmC6Savc)

|  |
| --- |
| **Síntesis del video: Bases de datos** |
| Las bases de datos NoSQL han tomado un gran papel e importancia en los sistemas de almacenamiento, su velocidad, rapidez y la integración con los diferentes lenguajes de programación, las convierten en una gran alternativa.  Es importante destacar que en una base de datos NoSQL no existen consultas SQL convencionales, sino que existen métodos propios para realizar las diferentes operaciones que se deben realizar en la base de datos.  La implementación de sistemas de almacenamiento con la capacidad de poder manejar la información de forma rápida, sencilla y segura, es necesario; estos procesos de almacenamiento y manejo de información se basan en tres pilares fundamentales, los cuales son: integridad, disponibilidad y confidencialidad. |

# Lenguajes de sistemas administradores de base de datos (DBMS) y lenguaje estructurado de consulta (SQL)

Continuemos hablando sobre los lenguajes de sistemas de administradores de bases de datos, los cuales son una serie de comandos que permiten realizar las operaciones requeridas en la base de datos. Estas operaciones van desde la creación de la base de datos hasta la manipulación de los datos que se encuentran almacenados; estos lenguajes se basan en estándares establecidos de manera general para la mayoría de los motores de bases de datos que se encuentran en la actualidad, lo que quiere decir que pasar de un motor de bases de datos a otro no implica mayor esfuerzo para el programador o administrador de la base de datos.

A continuación, veremos los diferentes lenguajes que se encuentran disponibles para estos procesos:

1. **DDL:** El lenguaje DDL, cuyas siglas traducidas significan **Lenguaje de definición de datos**, es utilizado para establecer la estructura de la base de datos; con este lenguaje creamos nuestra base de datos y se asignan las diferentes entidades o tablas que esta base de datos poseerá, lo cual hace dependiente todo el proceso de las bases de datos, ya que sin la creación de la base de datos no existe nada, ni relaciones o información. Dentro de este lenguaje encontramos varios comandos, tales como:

* ***Alter Procedure*:** recopilar un procedimiento almacenado.
* ***Create Table*:** crear una tabla.
* ***Create Index*:** crear un índice.
* ***Drop*:** eliminar objetos de una base de datos.
* ***Drop Index*:** eliminar un índice.
* ***Grant*:** conceder privilegios o papeles, roles, a un usuario o a otro rol.
* ***Truncate*:** elimina todos los registros de una tabla.
* ***Revoke*:** retirar los privilegios de un usuario o rol de la base de datos.
* ***Alter Table*:** añadir o redefinir una columna, modificar la asignación de almacenamiento.
* ***Analyze*:** recoge estadísticas de rendimiento sobre los objetos de la BD para utilizarlas en el optimizador basado en costes.

1. **DML**: El lenguaje DML, cuyas siglas significan **Lenguaje de manipulación de datos**, nos permite realizar la gestión de la información que se encuentra dentro de la base de datos; dichos procesos involucran desde la inserción de información, su actualización y eliminación de la misma. Este lenguaje proporciona todas las instrucciones para que el usuario a través del aplicativo pueda realizar las labores de manipulación de la información. Algo importante a destacar de este lenguaje es que muchas de sus operaciones no son reversibles, como por ejemplo la eliminación, ya que una vez se realiza este proceso no se puede recuperar la información. A continuación, veremos algunos comandos utilizados en este lenguaje:

* ***Insert*:** añadir filas de datos a una tabla.
* ***Delete*:** eliminar filas de datos de una tabla.
* ***Update***: modificar los datos de una tabla.
* ***Select***: recuperar datos de una tabla

1. **DCL**: Un Lenguaje de Control de Datos es un lenguaje proporcionado por el sistema de gestión de base de datos, que incluye una serie de comandos SQL que permiten al administrador controlar el acceso a los datos contenidos en la base de datos. Algunos ejemplos de comandos incluidos en el DCL son:

* ***GRANT*:** permite asignar permisos sobre el objeto de la base de datos.
* ***REVOKE*:** para quitar permisos.
* ***COMMIT*:** finaliza la transacción y realiza los cambios hechos durante la transacción.
* ***ROLLBACK*:** rechaza la transacción y no aplica cambios, volviendo al estado antes de iniciarse la transacción.

1. **VDL**: Lenguaje de Definición de Vistas. Es utilizado por el DBMS para especificar las vistas del usuario y sus correspondencias con el esquema conceptual. En las Bases de Datos Relacionales, el SQL representa una combinación de los anteriores, es decir, este lenguaje permite agrupar lo mejor de los antes mencionados dando como resultado un proceso de ilustración de información para el usuario; muchos de estos se pueden utilizan en la muestra de reportes y vistas al usuario final. Este lenguaje también es muy empleado para realizar las diferentes consultas de datos que se generan dentro de un sistema de bases de datos y luego traducirlas en información de utilidad para el usuario final, adicional a ello cuenta en la mayoría de gestores de bases de datos con interfaces muy amigables para su generación.

**Lenguaje estructurado de consulta (SQL)**

Ahora, el lenguaje de consulta estructurado es un estándar de codificación para bases de datos que permiten, mediante una serie de instrucciones precisas, ubicar los datos y la información contenidos dentro de una base de datos. Es importante tener en cuenta que este lenguaje se crea como un estándar para la mayoría de motores de bases de datos que se manejan hoy en día, como SQLServer, Oracle, MySQL, entre otros. Una de las características más notable de este lenguaje es la sintaxis que maneja, porque se estructura como preguntas que se hacen como si fuese una dirección para encontrar una vivienda. En la figura se puede observar un ejemplo.

1. Sentencia SQL

La imagen muestra el siguiente código:
SELECT COUNT(codigo) as 'Cantidad de platos', tipoplato as 'Tipo de plato'
FROM comidas
GROUP BY tipoplato
HAVING COUNT (codigo) >= 1

Es importante tener presente que el lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado, estos elementos agrupados permiten que sea posible la elaboración de consultas que permiten recuperar la información que se encuentra almacenada dentro de la base de datos. Es importante tener en cuenta que esta información que se recupera es solo de lectura, es decir, permite realizar la visualización de la información contenida dentro de la base de datos pero no se puede, a través de un comando SQL, realizar la actualización o inserción de nueva información.

Dentro de las cláusulas encontramos SELECT, FROM, WHERE, AS, BETWEEN; dentro de los operadores encontramos =, >, <, >=, <=; dentro de las funciones de agregado encontramos SUM, COUNT, HAVING, entre otras. Es importante tener en cuenta que la estructuración del lenguaje SQL debe ser con un orden establecido dentro de la secuencia de comandos para que puedan generar el resultado correcto y no error de información.

Ampliemos la información sobre algunos comandos y su respectiva descripción:

1. ***SELECT***: Palabra clave que indica que la sentencia de SQL que queremos ejecutar es una consulta. Selecciona tanto los campos que se enumeran como todos los registros que cumplan con la condición de la parte WHERE. Cuando los atributos se toman de distintas tablas en la parte FROM, también realiza la reunión (join). Por eso se dice que es un lenguaje ortogonal.
2. ***FROM***: Indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o reunión join. En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.
3. ***WHERE***: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR además de los relacionales y otros.
4. ***GROUP BY*:** Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.
5. ***ORDER BY*:** Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones. Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros —y no a registros individuales—, permite una alta productividad en codificación y la orientación a objetos.

De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros. SQL también tiene las siguientes características:

1. **Lenguaje de definición de datos:** El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
2. **Lenguaje interactivo de manipulación de datos:** El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
3. **Integridad:** El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.

Definición de vistas: el LDD incluye comandos para definir las vistas.

Control de transacciones: SQL tiene comandos para especificar el comienzo y el final de una transacción.

1. **SQL incorporado y dinámico:** Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, PHP, COBOL, Pascal y Fortran.
2. **Autorización:** El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

En los datos básicos de SQL, encontramos los siguientes tipos de datos:

* ***Varchar***: recibe cadena de palabras compuestas de letras, números y caracteres especiales.
* ***Int***: es el principal tipo de datos de valores enteros de SQL Server. Con números enteros con o sin signo.
* ***Date***: una fecha de calendario que contiene el año (de cuatro cifras), el mes y el día.
* ***Time***: la hora del día en horas minutos segundos (el valor predeterminado es 0).

# Tipos de datos

Los tipos de datos son, de alguna manera, restricciones que se establecen en los datos que se almacenan en una base de datos y están estrechamente relacionados con la información que se desea almacenar y manipular dentro de la misma. Es importante tener en cuenta que, por ejemplo, existen algunas restricciones, precisamente, con los tipos de datos; un ejemplo de esto son los datos de tipo *int* que son datos que solo reciben valores numéricos, es decir, si un usuario trata de almacenar, por ejemplo, una letra o palabra, automáticamente la base de datos nos informará de una violación de tipos de datos.

Es decir, estamos incumpliendo esa restricción de solo insertar datos números en ese tipo de datos, pero en el caso, por ejemplo, de los tipos de datos *varchar*, estos permiten almacenar datos tales como letras, números y símbolos, pero al momento de tratar de realizar alguna operación con ellos, mandará error; por ejemplo, tenemos un dato de tipo *varchar* y guardamos el valor 5, él permite almacenarlo sin problemas, pero si tratamos de hacer una operación matemática con este campo, no podremos hacerla, a menos que realicemos una conversión.

A continuación, veremos un contexto más detallado sobre esta temática:

1. **Datos de caracteres:** El tipo más común de datos está en formato de carácter (información que se almacena como cadenas de caracteres y pueden ser manipulados solo en cadenas). Algunos ejemplos de esta información son los nombres de clientes, direcciones, teléfonos, etc. Dos cadenas de caracteres se pueden ubicar juntas (concatenado), una subcadena puede ser seleccionada de una cadena más larga y una cadena puede ser sustituida por otra.
2. **Datos numéricos:** Otro tipo de datos muy común es el numérico (datos que se almacenan como un número). Por ejemplo, pueden ser almacenados números decimales (por ejemplo, 10.5, 2.34567, 23456.7) o números enteros (por ejemplo, 1, 2, 248). Cuando los datos se almacenan como un número pueden ser utilizados en operaciones numéricas como sumar, restar y hacer cuadraturas.

Tenga en cuenta que si los datos no van a ser utilizados para operaciones numéricas es mejor almacenarla como una cadena de caracteres, porque el programador va a utilizarla como una cadena de caracteres y no requerirá conversión. Por ejemplo, probablemente no se quiera añadir los dígitos en los números telefónicos de los usuarios, por lo que los números de teléfono deberían estar almacenados como cadenas de caracteres.

1. **Datos de fecha y hora:** Un tercer tipo común de datos son los de fecha y hora. Los datos almacenados como fecha pueden ser desplegados en una variedad de formatos de fecha. También se puede utilizar para determinar la longitud de tiempo entre dos fechas o períodos o entre una fecha y hora específica y una fecha o tiempo arbitrario.
2. **Datos de enumeración:** Algunas veces los datos pueden tener solo un número limitado de valores. Por ejemplo, los únicos valores posibles de una columna pueden ser sí o no.

De acuerdo con las necesidades del usuario, los datos se pueden categorizar con diferentes tipos, los cuales generan ciertas restricciones al momento de usarlos.

A continuación, abordaremos algunas de ellas:

1. **Tipo número:** Si se desea almacenar un dato de tipo número para realizar operaciones matemáticas con él, es indispensable primero que todo que este campo posea la propiedad *INT*, la cual identifica a este campo como numérico y tiene la restricción de solo almacenar datos de tipo número; en caso de que el usuario intente almacenar otro tipo de dato, el motor de la base de datos informará sobre el error cometido.
2. **Tipo fecha:** Si se desea almacenar un dato de tipo fecha, se debe utilizar en su declaración la palabra reservada ***DATE***, la cual le indica a la base de datos que este dato se refiere a una fecha. Ahora bien, en algunos motores de bases de datos afecta mucho la configuración que tengamos de la fecha en la computadora, por ejemplo: 09/09/2022 indicando día/mes/año, pero algunos motores de bases de datos manejan otra configuración, por ejemplo; 2022/12/25 indicando año/mes/día; por ello es importante tener en cuenta este tipo de aclaraciones.
3. **Tipo cadena:** Cuando se refiere a cadena está hablando de cadena de caracteres, este es uno de los tipos de datos ***VARCHAR*** o ***CHAR***, los cuales son los más utilizados, ya que permiten una versatilidad al momento de almacenar información, es decir, permiten almacenar cualquier tipo de datos incluyendo números, letras, símbolos, caracteres, entre otros. Pero su restricción está en que a pesar de que almacena cualquier tipo de datos, al momento de usarlos se tiene que realizar la conversión al dato requerido para esa operación; ejemplo: si se tiene un campo de tipo ***VARCHAR*** y se tiene almacenado el valor 4, quiere decir que si se desea hacer una operación matemática con dicho campo, se debe convertir al tipo de dato correspondiente, es decir, ***INT***, de lo contrario no se podrá realizar dicha operación. Esta restricción aplica para los demás valores que puedan ser almacenados en el mismo.
4. **Enumerable o autonumérico:** Este tipo de datos se utiliza cuando se desea que un valor contenido dentro de un campo de la base de datos se genere de manera automática; esto es posible gracias a un algoritmo que permite en la tabla de almacenamiento de la base de datos configurar que un campo genere de manera automática y ordenada una secuencia de números.
5. **Tipo float:** Este es un tipo de datos que tiene como restricción recibir valores numéricos que tengan puntos decimales, por ejemplo: 0.5, 25.6, etc. Este tipo de datos se utiliza mucho cuando se realizan operaciones matemáticas de multiplicación de porcentajes o divisiones, ya que tienden a tener una probabilidad de generar dicho tipo de dato.

**Valores por defecto (Default)**

La restricción DEFAULT se usa para proporcionar un valor predeterminado para una columna.

El valor predeterminado se agregará a todos los registros nuevos SI no se especifica ningún otro valor.

El siguiente SQL establece un valor DEFAULT para la columna «Ciudad» cuando se crea la tabla «Personas».

1. Comando default

La imagen muestra el siguiente código:
CREATE TABLE Personas (
 ID int NOT NULL,
 Apellidos varchar (255) NOT NULL,
 Nombre varchar (255),
 Edad int,
 Ciudad varchar(255) DEFAULT ‘Medellín’
);


La restricción *DEFAULT* también se puede usar para insertar valores del sistema mediante el uso de funciones como *GETDATE* *().*

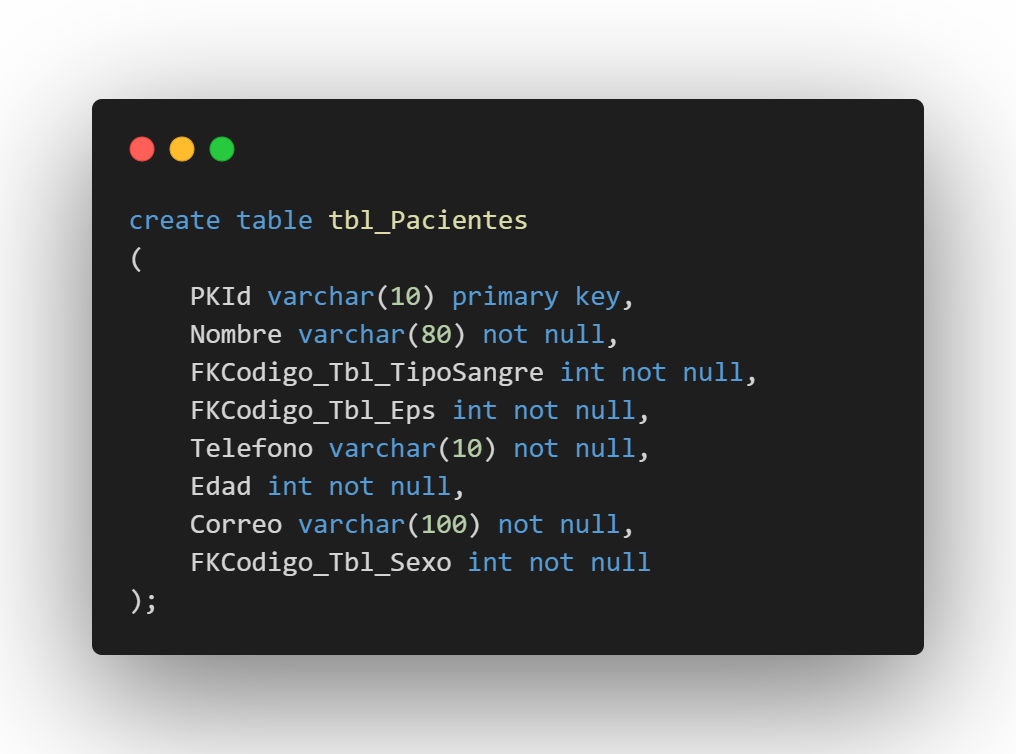
1. Asignación de fecha comando default

La imagen muestra el siguiente código:
CREATE TABLE Orders (
 ID int NOT NULL,
 NumeroPedido int NOT NULL,
 FechaPedido date DEFAULT GETDATE ()
);


***Null y Not null***

Por defecto, las columnas pueden contener valores NULL. Se usa una restricción NOT NULL en SQL para evitar insertar valores NULL en la columna especificada, considerándolo entonces como un valor no aceptado para esa columna. Esto significa que se debe proporcionar un valor válido SQL NOT NULL a esa columna en las instrucciones INSERT o UPDATE, porque la columna siempre contendrá datos; un ejemplo de esto, lo observamos en la figura.

1. Uso de cláusula Not null

****

Como se puede observar en la figura anterior, el uso de la cláusula NOT NULL es para generar una restricción e impedir que se reciban valores nulos o vacíos dentro de los campos seleccionados; por ejemplo, en la tabla **tbl**\_**Pacientes** que se observa en el ejemplo, el campo de nombres está como NOT NULL, porque para la base de datos no tendría ningún sentido registrar a un paciente sin nombre y de allí nace la necesidad del uso de esta cláusula. Para el caso que se desee poder recibir valores nulos, simplemente quitamos esta restricción; es decir, retiramos las palabras reservadas NOT NULL y automáticamente la base de datos permitirá almacenar valores NULL dentro de la misma; un ejemplo de ello es cuando se desea registrar a una persona y se solicita primer nombre y segundo nombre, no todas las personas tienen segundo nombre, por lo tanto se le puede dejar la posibilidad que estos sean opcionales y no afecten el almacenamiento de los datos. Observe la figura:

1. Uso de cláusula null

La imagen refleja el uso de la cláusula null por medio del siguiente codigo: 

create table tbl_Pacientes
(
    PKId varchar (10) primary key,
     Primer _nombre varchar (80) not null,
Segundo_nombre varchar (80),
Contacto varchar (10) not null
); 


Como se observa en la figura, el campo señalado segundo nombre, no tiene restricción porque no es obligatorio que todas las personas tengan segundo nombre; en este caso, al retirar la cláusula la base de datos entiende que es un valor no obligatorio en la inserción de los datos.

# Lenguajes de definición de datos DDL

El lenguaje DDL, cuyas siglas significan Lenguaje de Definición de Datos, tiene la misión de entregar todas las instrucciones y comandos que permiten la creación de la estructura de la base de datos y las restricciones y relaciones que se pueden dar dentro de la misma. Es importante tener en cuenta que este lenguaje es universal para la mayoría de las bases de datos que se encuentran en la actualidad y esto quiere decir que es un estándar que se utiliza para este propósito. A continuación, veremos sus instrucciones y para qué se utiliza cada una de ellas:

***Create***

La instrucción CREATE tiene como propósito realizar la creación de la estructura principal de la base de datos y es la que permite realizar el inicio de las sentencias DDL. Sin la creación de la base de datos, sería imposible el agregado de los demás componentes que esta tiene; ahora bien, la inicialización de la base de datos es importante en estos sentidos, debido a que para todos los motores de bases de datos, está vigente y tiene el mismo propósito de la creación de la base de datos. Observe el ejemplo:

1. Comando *CREATE*

**La imagen muestra el siguiente código:
Create database dbs_RegistroPasajeros
Go
Create table tbl_Pasajeros
(
 PKIdentificacion varchar(11) primary key,
Nombres varchar(50) not null, 
Contacto varchar(10) not null,
Correo varchar(60) not null
)**

Se puede visualizar en la figura anterior la utilización del comando CREATE, el cual se encuentra acompañado de otras instrucciones que se utilizan para la creación de la base de datos, que en este caso se llama dbs\_RegistroPasajeros, la cual luego se le adiciona una tabla llamada tbl\_Pasajeros.

***Alter***

La instrucción **ALTER** se utiliza cuando se desea cambiar el tipo de dato, nombre o relación de un componente de la base de datos; en este caso particular, se podría decir que este comando tiene varios propósitos que están acompañados, de acuerdo con la necesidad de cambio que se requiera dentro de la base de datos, y de la empresa que requiera dichos cambios; aunque en el proceso de la creación de la base de datos se realizan varias verificaciones y en la mayoría de los casos los cambios que se realizan no están relacionados con problemas de creación o asignación de tipos de datos y/o relaciones, sino que se tratan de actualizaciones en la base de datos, donde un dato requiere ser recibido de alguna manera o formato específico, entonces se requiere realizar ciertas modificaciones. Observemos en la tabla donde aplica la instrucción **ALTER**.

1. Sentencia ALTER

En la imagen se muestran algunos comandos o instrucciones con la sentencia ALTER como por ejemplo agregar nueva columna Edad en la tabla:
alter table tbl_Pasajeros
add Edad varchar(3)

En la tabla anterior se visualizan los ejemplos que se pueden aplicar para la instrucción ALTER, la cual permite modificar componentes de la estructura de la base de datos, donde se pueden quitar campos, agregar campos y modificar los valores de tipo de datos que esta contiene, aunque existe un propósito que está vinculado con las relaciones de las entidades de la base de datos y se refiere a realizar la conexión de las mismas. Observemos un ejemplo.

1. Instrucción ALTER en relación de tablas

En la imagen se describe cómo relacionar dos tablas y se señala el siguiente código:
Alter table tbl_Pasajeros
Add constraint FKCodigo_tbl_Sexo
Foreing key (FKCodigo_TBL_Sexo) references tbl_Sexo

En la figura anterior se puede observar el uso del comando ***ALTER*** para realizar la relación de la tabla tbl\_Sexo con la tabla tbl\_Pasajeros. Se aplica el comando ***ALTER*** a la tabla tbl\_Pasajeros debido a que esta es la que debe realizar el cambio de permitir la relación con la tabla tbl\_Sexo y de esta manera realizamos una modificación en la tabla tbl\_Pasajeros para recibir la relación; es otra de las funcionalidades que tiene la instrucción ***ALTER***.

***Drop***

La instrucción **DROP** se utiliza cuando se desea quitar o remover un componente de la base de datos; esto no solo se aplica para tablas y campos, sino que también se puede aplicar para las mismas bases de datos, cuando se requiere de este proceso. Observemos algunos ejemplos:

1. Utilización de la instrucción DROP

La imagen nos muestra ejemplos de uso de la instrucción DROP, ejemplo:
Eliminar la tabla tbl_Calificaciones:
drop table tbl_Calificaciones

En la tabla anterior se encuentran las situaciones en las que se aplica la instrucción ***DROP***; esta es utilizada en caso de que se requiera remover un componente de la base de datos, es importante tener en cuenta que este procedimiento se debe realizar con suma precaución debido a las implicaciones que puede generar tanto en la estructura de la base de datos como en su correcto funcionamiento.

***Truncate* para tablas**

La instrucción ***TRUNCATE*** tiene una de las complicaciones más severas que se pueden aplicar en una base de datos, por eso requiere de un adecuado uso, ya que esta instrucción realiza una afectación directa a los datos almacenados, es decir, cuando se utiliza la instrucción ***TRUNCATE*** no solo se eliminan los datos que se encuentran almacenados en la tabla de la base de datos, sino que este comando reinicia la configuración de campos con numeración. Es decir, si se tiene una tabla con un campo o columna autonumérica y su conteo va por el 450 y se aplica el comando ***TRUNCATE***, este automáticamente borra todos los datos y reinicia en 1 el valor del campo autonumérico; por eso es importante tener en cuenta al momento de realizar su aplicación, porque puede tener algunas implicaciones adicionales al momento de usarse. A continuación, veremos un ejemplo.

1. Uso de comando TRUNCATE

En la imagen se observa el código:
use dbs_Calificaciones
go
truncate table tbl_Asignaturas

En la figura se aprecia el uso de la instrucción TRUNCATE en la tabla llamada tbl\_Asignaturas, lo que borraría todos los datos contenidos y adicionalmente el código de la asignatura, que es autonumérico, iniciaría en 1 nuevamente. Esto se llama reiniciar el contador.

**Índices**

Ahora, un índice es una estructura de datos definida sobre una columna de tabla (o varias) y permite localizar de forma rápida las filas de la tabla en base a su contenido en la columna indexada, además de permitir recuperar las filas de la tabla ordenadas por esa misma columna.

Conozcamos cuáles son estos índices:

1. **Índice simple y compuesto:** Un índice simple está definido sobre una sola columna de la tabla, mientras que un índice compuesto está formado por varias columnas de la misma tabla (tabla sobre la cual está definido el índice). Cuando se define un índice sobre una columna, los registros que se recuperen utilizando el índice aparecerán ordenados por el campo indexado. Si se define un índice compuesto por las columnas col1 y col2, las filas que se recuperen utilizando dicho índice aparecerán ordenadas por los valores de col1 y todas las filas que tengan el mismo valor de col1 se ordenarán a su vez por los valores contenidos en col2, función igual que la cláusula *ORDER* *BY*. Por ejemplo, si definimos un índice compuesto basado en las columnas (provincia, localidad), las filas que se recuperen utilizando este índice aparecerán ordenadas por provincia y dentro de la misma provincia por localidad.
2. **Índice agrupado y no agrupado:** El término índice agrupado no se debe confundir con índice compuesto, el significado es totalmente diferente. Un índice agrupado (*CLUSTERED*) es un índice en el que el orden lógico de los valores de clave determina el orden físico de las filas correspondientes de la tabla. El nivel inferior u hoja de un índice agrupado contiene las filas de datos en sí de la tabla. Una tabla o vista permite un solo índice agrupado al mismo tiempo. Los índices no agrupados existentes en las tablas se vuelven a generar al crear un índice agrupado, por lo que es conveniente crear el índice agrupado antes de crear los índices no agrupados. Un índice no agrupado especifica la ordenación lógica de la tabla. Con un índice no agrupado, el orden físico de las filas de datos es independiente del orden indexado.
3. **Índice único:** Índice único es aquel en el que no se permite que dos filas tengan el mismo valor en la columna de clave del índice. Es decir, que no permite valores duplicados.

# Restricciones de integridad

Las restricciones de integridad que existen en una base de datos se refieren a la protección y mantenimiento de los datos, es decir, cuando hablamos de integridad nos referimos a la capacidad del sistema de mantener los datos de manera coherente y permanente sin que sufran cambios inesperados y que estos cambios que se den estén sujetos a políticas o necesidades del sistema mismo. Un ejemplo de ello sería un cajero automático que debe realizar operaciones todos los días y que estas operaciones deben realizarse de manera precisa y eficiente.

Adicional a ello, cuando nos referimos a eficiente es un tema relacionado con la integridad, es decir, los descuentos o movimientos que realice deben estar sujetos a programación establecida y no deben cambiar a menos que una política interna de la organización o el programador realice dicho cambio.

**Integridad de unicidad**

La regla de integridad de unicidad está relacionada con la definición de clave primaria. Concretamente, establece que toda clave primaria que se elija para una relación, no debe tener valores repetidos, como se observa a continuación:

1. Tabla despachos

|  |  |  |
| --- | --- | --- |
| Edificio | Número | Superficie |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

En esta relación, dado que la clave primaria está formada por edificio y número, no hay ningún despacho que repita tanto edificio como número de otro despacho. Sin embargo, sí se repiten valores de edificio (por ejemplo, Marina); y también se repiten valores de número (120). A pesar de ello, el edificio y el número no se repiten nunca al mismo tiempo.

La regla de integridad de unicidad de la clave primaria establece que si el conjunto de atributos CP es la clave primaria de una relación R, entonces la extensión de R no puede tener en ningún momento dos tuplas con la misma combinación de valores para los atributos de CP.

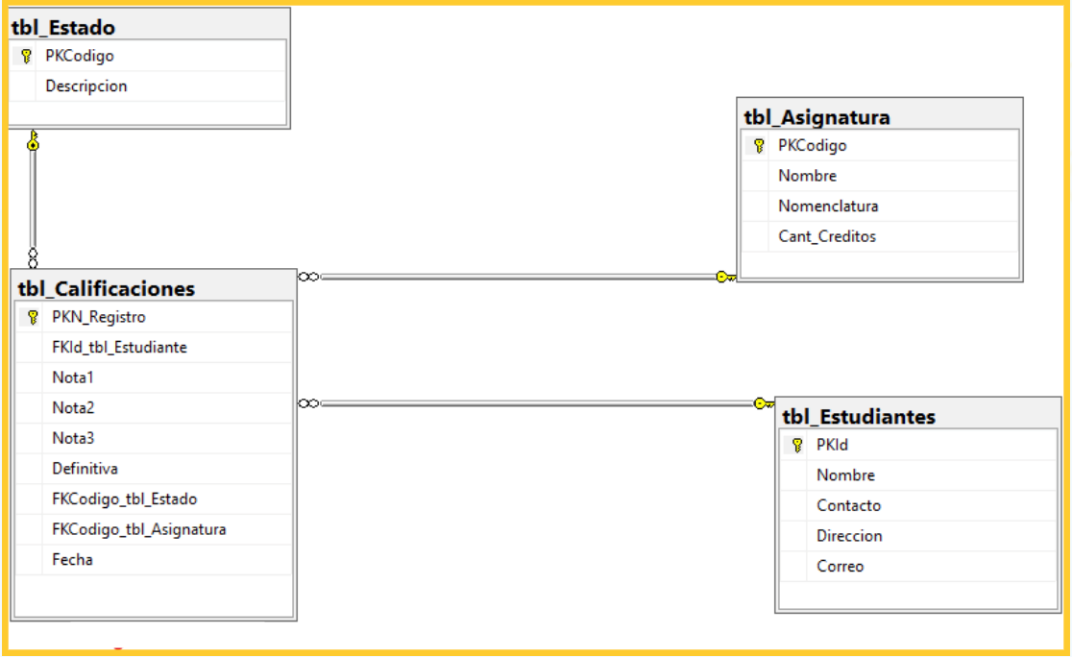
Un SGBD relacional deberá garantizar el cumplimiento de esta regla de integridad en todas las inserciones, así como en todas las modificaciones que afecten a atributos que pertenecen a la clave primaria de la relación.

En esta relación no se debería poder insertar la tupla <Diagonal, 120, 30>, ni modificar la tupla <Marina, 122, 15 >, de modo que pasará a ser <Marina, 120, 15>.

**Integridad referencial**

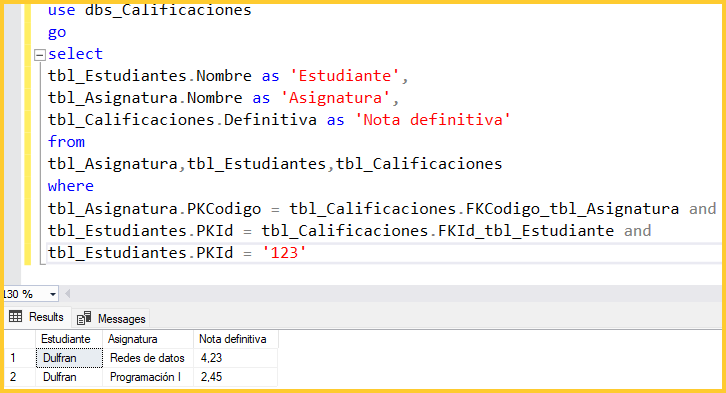
La función de integridad referencial comprueba que cada valor de datos de clave foránea, sea de columna única o de varias columnas (concatenado), pueda acceder a la clave primaria de una tabla relacionada, esto se refiere a la capacidad de la base de datos de poder crear conexiones para la realización de las consultas que la misma ejecuta. Cuando se realiza una consulta SQL regularmente, la información tiene diferentes tablas de donde se saca la información y para que esta información pueda ser accedida se requiere que las llaves tanto primarias como foráneas sean iguales para que se pueda traer el dato requerido. A continuación, veamos un ejemplo.

1. Diagrama entidad-relación



Como se puede observar en la figura anterior, el diagrama entidad - relación indica que existe una restricción de integridad referencia, en este caso la llave PKCodigo de la tabla tbl\_Asignatura está relacionada con la llave foránea de la tabla tbl\_Calificaciones; por lo tanto en la consulta se debe crear esta comparación para que pueda asociar las asignaturas con las calificaciones que esta recibe. Adicional a ello vemos que la tabla tbl\_Estudiantes está relacionada con la tabla tbl\_Calificaciones por medio de la llave foránea, esto quiere decir que un estudiante puede recibir varias calificaciones de diferentes asignaturas; esto es lo que se conoce como integridad referencial. A continuación, veremos la consulta SQL utilizando la integridad referencial para recuperar los datos y relacionarlos entre sí.

**Figura 10.** Integridad referencial bajo código



Como se puede observar, en la figura anterior se encuentra aplicada la integridad referencial comparando las llaves primarias con las foráneas para encontrar coincidencia en los datos; en caso de que esta consecuencia sea positiva, la consulta traerá el resultado esperado realizando las diferentes combinaciones que se requieran para realizar dicho proceso.

**Integridad de dominio**

Se refiere a la corrección de la información contenida en la BD. A cada atributo está asociado un dominio de valores posibles. Los límites de dominio son la forma más elemental de restricciones de integridad. Son fáciles de probar por el sistema siempre que se introducen nuevos datos a la BD.

Algunas características aplicadas son:

1. A cada atributo está asociado un dominio de valores posibles.
2. Los límites de dominio son la forma más elemental de restricciones de integridad.
3. Son fáciles de probar por el sistema siempre que se introducen nuevos datos a la base de datos.

Un ejemplo de ello sería examinando una base de datos que almacena información de proveedores y partes, entonces tendríamos lo siguiente:

1. El peso de una parte no puede ser negativo.

En la base de datos proveedores tendríamos lo siguiente:

1. Los números de una sucursal no pueden ser negativos.
2. Las ciudades de sucursales, deben provenir de una cierta lista.

La integridad de dominio es la validez de las restricciones que debe cumplir una determinada columna de la tabla, lo cual incluye:

1. **Datos requeridos:** Establece que una columna tenga un valor *NOT NULL*. Se define efectuando la declaración de una columna es *NOT NULL* cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia *CREATE TABLE.*
2. **Chequeo de validez:** Cuando se crea una tabla cada columna tiene un tipo de datos y el DBMS asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

Se pueden observar los diferentes tipos de integridades y la afectación que pueden tener dentro de la base de datos; es importante tener en cuenta que esta afectación va desde los registros de una columna o de una fila hasta las relaciones que se establecen en las entidades o tablas de la base de datos, por eso es importante al momento de realizar el diagrama entidad-relación tener en cuenta que cada llave a relacionar debe tener el mismo tipo de datos y longitud para evitar pérdida de datos o problemas de relacionamiento de entidades.

# Lenguajes de manipulación de datos DML

Hablemos ahora sobre el lenguaje DML orientado a bases de datos, el cual tiene los comandos e instrucciones para realizar la manipulación de la información que se encuentra dentro de la base de datos; es importante tener en cuenta que uno de los pilares de los sistemas de almacenamiento es que la información debe mantener su integridad y disponibilidad para los usuarios que la requieran, por eso es importante tener en cuenta que una mala manipulación de estos comandos puede traer repercusiones que pueden costarle la integridad a la información. Otro de los aspectos a tener en cuenta es que cuando se realicen procedimientos como son el *UPDATE* o *DELETE* se debe tener la precaución de verificar que se está afectando la fila e información porque una vez realizado el proceso no existe vuelta atrás.

A continuación, explicaremos estos comandos y su utilización:

1. ***Insert***: El comando ***INSERT*** se utiliza cuando se requiere realizar el registro de una nueva fila en la base de datos: este comando es el que permite alimentar la base de datos y consta de algunas consideraciones en su sintaxis para tener en cuenta al momento de utilizarlo. Otro aspecto importante es que la sentencia ***INSERT*** puede llegar a entrar en conflicto con las llaves de conexión de la base de datos, un ejemplo de ello son las llaves primarias que se colocan en las tablas para evitar que un registro duplique su información. Al momento de realizar la inserción se debe verificar que este valor no sea duplicado, pues generará un error en la base de datos, ya que las llaves primarias no permiten duplicidad de datos. A continuación, veremos una corta explicación sobre esto.

**Figura 11.** Comando *Insert*

En la imagen se observa el siguiente código:
use dbs_Calificaciones
go
insert into tbl_Asignatura
(PKCodigo, Nombre, Nomenclatura)
values('ING6','Algoritmos I', 'ALG1'

En la imagen se puede observar la sintaxis que aplica la sentencia ***INSERT***: primero se colocan las instrucciones *insert* *into* que le indican al motor de la base de datos que es una instrucción que colocará un nuevo registro dentro de la base de datos; luego se indica la tabla donde se insertarán los datos seguido de los campos o columnas que posee la tabla en el orden específico de inserción; luego la instrucción *values* indicando los valores que serán insertados en el orden específico que requiere la tabla.

1. ***Update***: La instrucción ***UPDATE*** permite realizar la actualización de la información que previamente ha sido registrada en la tabla de la base de datos. Esto es muy útil en el sentido de que permite que se puedan realizar cambios en la información almacenada luego de realizar el registro de la misma, es importante tener en cuenta que al momento de realizar la actualización se debe verificar que la fila que se está actualizando sea la que requiere el cambio; esto lo podemos especificar con la instrucción *where* donde se coloca la llave primaria para realizar esta operación. A continuación, se ilustra la manera como se utiliza este comando.

**Figura 11.** Comando *Update*

En la imagen se observa el código:
use dbs_Calificaciones
go
update tbl_Asignatura
set Nombre= 'Algoritmos y programación avanzada'
where
PKCodigo='ING6'


Como se observa en la imagen, la sintaxis que sigue este comando es la siguiente: primero se indica el comando ***UPDATE*** para que el motor de la base de datos comprenda que la operación a realizar es de actualización de datos; luego se indica la tabla que contiene la información a actualizar seguido de la instrucción ***SET***, la cual permite colocar el campo o en algunos casos los campos que recibirán el nuevo dato; luego se procede a colocar la condición con la cláusula ***WHERE***, donde regularmente se utiliza la llave primaria, ya que esta no contiene dato repetido y se utiliza como referente para la actualización.

1. ***Delete***: El comando ***DELETE*** se utiliza en los casos en que se desea realizar la eliminación de una fila dentro de una tabla de la base de datos. Es importante tener en *cuenta* que este comando le aplica las mismas características del comando ***UPDATE***, donde se debe especificar la fila que se debe eliminar utilizando la sentencia ***WHERE*** con la llave primaria de esa fila. Esto se hace con el fin de prevenir que se eliminen todos los registros de la base de datos; en el caso de no especificar dicha fila el motor de la base de datos interpretará que se hará una eliminación de todas las filas que este posee. A continuación, veremos un ejemplo sobre esta apreciación.

**Figura 13.** Comando *Delete*

En la imagen se observa el código:
use dbs_Calificaciones
go
delete from tbl_Asignatura where PKCodigo='ING6'

Como se puede apreciar en la imagen, se realiza la ejecución del comando ***DELETE***, el cual tiene la siguiente sintaxis: primero se procede a colocar el comando ***DELETE*** para indicarle al motor de la base de datos que se ejecutará una sentencia de eliminación de fila; luego se procede a colocar la instrucción ***FROM***, la cual indica el origen de los datos que se van a eliminar seguido del nombre de la tabla de donde provienen esos datos; adicional a ello se debe utilizar la sentencia ***WHERE*** como habíamos comentado anteriormente, que se utiliza también en la sentencia ***UPDATE*** para especificar mediante la llave primaria la fila a la que corresponde el dato que se desea eliminar; esto con el fin de que no se eliminen todos los registros de la tabla, sino el que pertenece a la fila que se desea afectar.

1. ***Merge***: La sentencia ***MERGE*** actualiza un destino (una tabla o vista, o las tablas o vistas subyacentes de una selección completa) utilizando datos de una fuente (resultado de una referencia de tabla).

Las filas del destino que coinciden con la fuente se pueden suprimir o actualizar, según se especifique, y las filas que no existen en el destino se pueden insertar. Si se actualiza, suprime o inserta una fila en una vista, se actualiza, suprime o inserta la fila en las tablas en las que se basa la vista.

1. ***Select***: La sentencia ***SELECT*** es una de las sentencias que le dan un grado de importancia a los sistemas de bases de datos, ya que esta se crea como un estándar de consultas para la recuperación y generación de vistas de la información almacenada en las tablas de la base de datos; dicho de otra manera, esta sentencia es la que hace posible poder visualizar y consultar la información que se encuentra almacenada en una base de datos. Es importante tener en cuenta que este comando tiene diferentes maneras de expresar la forma en la que los datos son devueltos al usuario y que también están conectados con las necesidades y requerimientos de la aplicación de software que se esté desarrollando. Este tiene 3 tipos de consultas a realizar, las cuales son: consultas generales, consultas específicas y consultas personalizadas.

**Funciones DML**

Las funciones **DML** se establecen para realizar operaciones requeridas dentro de las bases de datos. Estas operaciones van desde simples procesos de cálculos de operaciones hasta realizar complejos procesos para el mantenimiento de los datos en la base de datos. Aunque todos estos procesos se llevan a cabo de acuerdo a necesidades técnicas que el usuario de la aplicación requiera, dichos procesos pueden estar relacionados con procesos que afectan distintas partes del sistema, por ello se hace necesario en algunas ocasiones establecer un orden específico de ejecución de los mismos para garantizar un correcto funcionamiento en el sistema mismo.

Para ampliar la información, le invitamos a leer el PDF.  [Anexo 1. Funciones DML](Funciones_DML.pdf)

**Operadores DML**

Los operadores DML permiten acompañar las labores de manipulación de los datos en una base de datos, dentro de estos se encuentran varias instrucciones que hacen uso de dichos operadores para cambiar, comparar o realizar operaciones de tipo matemático para entregar al usuario los resultados que este requiere en los procesos a desarrollar. Aunque los operadores pueden combinarse para realizar operaciones que llegan a ser de un nivel complejo, se debe entender que en una operación lo importante es la aplicación lógica de las operaciones y su integración con los valores de entrada y salida para mostrar al usuario resultados correctos. A continuación, veremos los diferentes operadores utilizados en DML.

Para ampliar la información, lo invitamos a leer el PDF. [Anexo 2. Operadores DML.](Operadores_DML.pdf)

**Consultas combinadas DML**

Como hemos estudiado anteriormente, el lenguaje DML permite realizar la manipulación de los datos que se encuentran en las tablas de acuerdo con instrucciones específicas y con un propósito definido, pero en este caso abordaremos algunas de ellas que son mucho más precisas y permiten llegar al nivel de profundidad y comprensión mucho más compleja. Para comprender mejor este contexto, le invitamos a estudiar algunas de ellas en el PDF. [Anexo 3. Consultas combinadas DML.](Consultas_combinadas_DML.pdf)

**Sentencias de agregación**

Las sentencias de agregación permiten realizar cálculos específicos dentro de la base de datos, en otras palabras, son funciones que ya se encuentran programadas en la base de datos y que solo hace falta llamarlas para aplicarlas en ella. Es importante que tengamos en cuenta que estas funciones no afectan los datos almacenados en las tablas de la base de datos, sino que utilizan los datos en modo de lectura para poder mostrar al usuario el resultado esperado.

Le invitamos a conocer estas funciones en el documento Sentencias de agregación. [Anexo 4. Sentencias de agregación.](Sentencias_de_agregacion.pdf)

# Construcción de base de datos NoSQL

Las bases de datos NoSQL nacen como una alternativa a las bases de datos tradicionales, estas tienen las características de ser muy rápidas y de adaptarse prácticamente a cualquier lenguaje de programación, aunque son muchas las ventajas que estas ofrecen, su mayor característica es el poco espacio que ocupan dentro de los sistemas de almacenamiento. Ahora bien, cada base de datos NoSQL permite realizar operaciones tal y como las realizaría una base de datos convencional, solo que el trabajo que esta realiza se hace de manera plana utilizando archivos con extensiones JSON o XML, lo cual permite su fácil adaptación y velocidad. A continuación, veremos varias características de las mismas.

**Operaciones básicas**

Las operaciones que se pueden realizar con una base de datos NoSQL son: insertar nuevos datos, buscar un dato previamente registrado, operaciones matemáticas según requiera el cliente y generación de búsquedas con patrones específicos. Esto se refiere a que puede realizar consultas complejas teniendo como resultado un poco tiempo de espera, por eso es que son muy utilizadas en aplicaciones que tienen grandes volúmenes de información y al ser de poco peso no se requiere de una infraestructura poderosa para realizar su ejecución.

Las operaciones básicas, se conforman por diferentes prototipos que reflejan su naturaleza, los cuales veremos a continuación:

1. **Bases de datos clave – valor:** Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una llave única, lo que permite la recuperación de la información de forma muy rápida; información que habitualmente está almacenada como un objeto binario (BLOB). Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras. Algunos ejemplos de este tipo son *Cassandra*, *BigTable* o *HBase*.
2. **Bases de datos documentales:** Este tipo almacena la información como un documento, generalmente utilizando para ello una estructura simple como JSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. Algunos ejemplos de este tipo son MongoDB o CouchDB.

Podemos decir que las bases de datos NoSQL tienen muchas ventajas sobre las bases de datos tradicionales, aunque es importante aclarar que dependiendo del requerimiento que se desee elaborar así será la elección a tomar teniendo en cuenta muchas características de estabilidad, crecimiento y capacidad de almacenamiento. Dentro de estas bases de datos se pueden realizar las mismas operaciones de inserción, búsqueda, actualización y eliminación que se hacen en una base de datos convencional teniendo en cuenta que esta lo maneja a nivel de archivos y no de registros.

**Operaciones CRUD**

Las operaciones de CRUD se refiere a las 4 operaciones principales que se realizan en una base de datos las cuales son: crear nuevos registros, actualizar registros, eliminar registros y consultarlos. Basados en esas 4 operaciones realizaremos una exploración de cada una de ellas para entrar en este contexto.

Son muchas las operaciones que se pueden hacer en MongoDB. Algunas de las más esenciales son insertar un nuevo documento (*create*), obtener un documento (*read*), actualizar un documento existente (*update*) y eliminar un documento (*delete*).

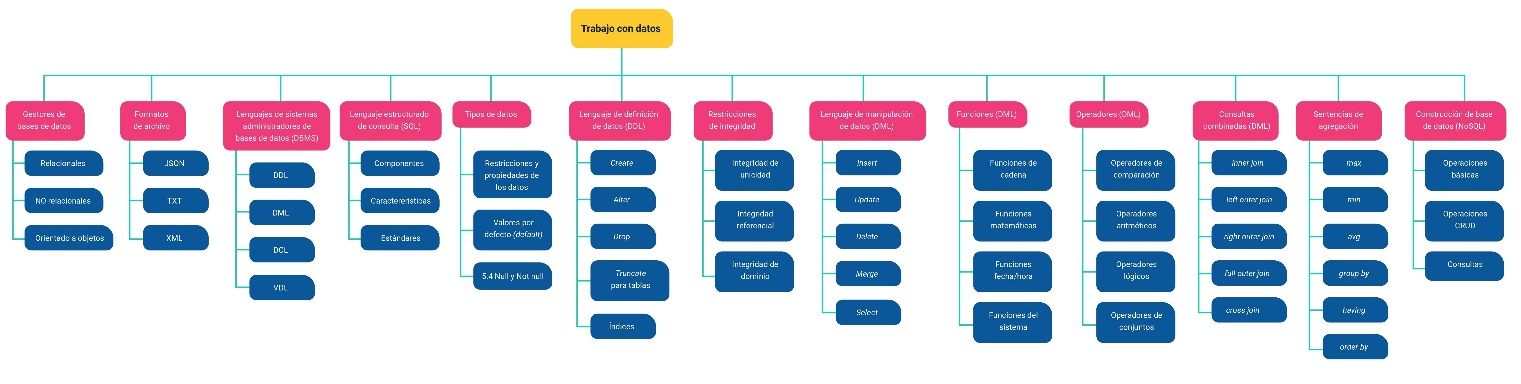
**Consultas**

Las consultas son muy importantes en los sistemas de almacenamiento, sobre todo para realizar procesos de manipulación y procesamiento de la misma. En el caso de las consultas, en las bases de datos NoSQL se realizan de una manera mucho más precisa que en las tradicionales, es decir, se debe indicar el documento a acceder y qué información de cada uno de ellos se va a sustraer. A continuación, veremos una serie de instrucciones para realizar este proceso.

Recuerde explorar los demás recursos que se encuentran disponibles en este componente formativo; para ello, diríjase al menú principal donde encontrará la síntesis, una actividad didáctica para reforzar los conceptos estudiados, material complementario, entre otros.

Síntesis

El siguiente mapa integra los criterios y especificidades de los conocimientos expuestos en el presente componente formativo:



[Anexo. Síntesis](sintesis.pdf)

Material complementario

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material | Enlace del Recurso o  Archivo del documento material |
| Gestores de base de datos y formatos de archivos | EDteam. (2020). *¿Qué son las BASES DE DATOS y cómo funcionan?* (video). YouTube. | Video | <https://www.youtube.com/watch?v=knVwokXITGI> |
| Lenguajes de sistemas administradores de base de datos(DBMS) y lenguaje estructurado de consulta (SQL) | Bases de Datos. (2020). *01 lenguaje de Consulta Estructurado (SQL) – Conceptos básicos* (video). YouTube. | Video | <https://www.youtube.com/watch?v=HlRhFDY1Bmg> |
| Lenguajes de definición de datos DDL | Robles, J. (2021). *Lenguaje de Definición de Datos (DDL)* (video). YouTube. | Video | <https://www.youtube.com/watch?v=7KQbw8v8jN8> |

Glosario

**Bases de datos documentales:** almacena la información como un documento, generalmente utilizando para ello una estructura simple como JSON o XML y donde se utiliza una clave única para cada registro.

**DATE**: es un tipo de datos que permite almacenar valores de tipo fecha y se puede combinar con comandos como GETDATE() para que almacene de manera automática la fecha que tiene asignada el servidor.

**DDL**: lenguaje de definición de datos el cual permite crear la estructura de la base de datos y sus componentes como tablas y campos, adicional a ello permite la creación de las diferentes relaciones de las entidades de la base de datos.

**DML**: lenguaje de manipulación de datos que permite realizar las operaciones para el mantenimiento de los datos de las tablas, las cuales se llaman CRUD, que significa: Create, Read, Update, Delete.

**Integridad de unicidad:** la regla de integridad de unicidad está relacionada con la definición de clave primaria. Concretamente, establece que toda clave primaria que se elija para una relación no debe tener valores repetidos.

**Integridad referencial:** se refiere al tipo de restricción que se establece en las bases de datos para evitar que la información sea afectada y no sea de utilidad posteriormente.

**JSON:** son archivos de intercambio de información que se usan a menudo en los sitios web, tienen una estructura específica y su tamaño y peso en disco son muy pequeños.

**NoSQL**: son un tipo de bases de datos que permiten almacenar un gran volumen de información y trabajan con archivos planos, lo que hace que su peso y velocidad sean muy favorables.

**SQL**: lenguaje de consulta estructurado, el cual permite mediante una serie de instrucciones específicas, recuperar la información almacenada dentro de las tablas de la base de datos.

**VARCHAR**: es un tipo de dato que se utiliza en las bases de datos para tener la posibilidad de almacenar cualquier tipo de dato sea carácter, número, símbolo entre otros.

Referencias bibliográficas

Carrasquel, S., Coronado, D., Rodríguez, R. & Tineo, L. (2018). Gestión De Datos Difusos: Atributos Tipo 2 Y Tipo 3 En Bases De Datos Relacionales. *Publicaciones En Ciencias Y Tecnología*, 12(2), p. 83-95. <https://dialnet.unirioja.es/descarga/articulo/6641662.pdf>

Przygodzki-Lionet, N., Mignol, M. & Cadet, F. (2019). *Puntos de vista de los estudiantes sobre la aceptabilidad de bases de datos del gobierno para la identificación de personas: un estudio piloto*.<https://repository.javeriana.edu.co/handle/10554/42321>

R. Domínguez Mónaco. "Reseña De Libro: Hazte Gurú De Base De Datos. SQL Diseño Y Normalización. Por L. Ponte Cordero." Publicaciones En Ciencias Y Tecnologia 12.1 (2018): 43-46. Web. <https://revistas.uclave.org/index.php/pcyt/article/view/836/847>

Créditos

Elaborado por Innovative Education

