

Bases de datos: SQL y NoSQL

**Breve descripción:**

Este componente aborda la gestión de bases de datos SQL y NoSQL, destacando sus características y aplicaciones. Incluye la configuración de entornos de trabajo con MySQL y MySQL Workbench, diseño y administración de bases de datos, y exploración de bases NoSQL como MongoDB. Los ejercicios prácticos refuerzan el aprendizaje mediante estudios de caso reales, como sistemas de facturación y gestión académica.

**Julio 2024**

Tabla de contenido

[Introducción 5](#_Toc172780080)

[1. Establecer un entorno de trabajo 6](#_Toc172780081)

[1.1. XML Extensible Markup Language 6](#_Toc172780082)

[1.2. Sistema de diseño y gestión de bases de datos 7](#_Toc172780083)

[1.3. Sistemas basados en MySQL 9](#_Toc172780084)

[2. Ejercicios de diseño e implementación 10](#_Toc172780085)

[2.1. Análisis y diseño sistema de facturación 10](#_Toc172780086)

[Análisis y diseño 11](#_Toc172780087)

[2.2. Identificación de sentencias DDL 20](#_Toc172780088)

[2.3. Generar la base de datos 26](#_Toc172780089)

[2.4. Operaciones de mantenimiento y actualización 27](#_Toc172780090)

[2.5. Actualización por interfaz de línea de comandos 29](#_Toc172780091)

[Diseño conceptual 32](#_Toc172780092)

[Diseño lógico 35](#_Toc172780093)

[Diseño físico 36](#_Toc172780094)

[3. Administración de bases de datos 38](#_Toc172780095)

[3.1. Copia de seguridad de base de datos con MySQL Workbench 39](#_Toc172780096)

[3.2. Restaurar una copia de seguridad con MySQL Workbench 40](#_Toc172780097)

[4. Gestores de contenido CMS 41](#_Toc172780098)

[4.1. Definición de NoSQL 41](#_Toc172780099)

[4.2. Cómo diferenciar NoSQL de SQL 42](#_Toc172780100)

[4.3. Tipos de bases de datos NoSQL 44](#_Toc172780101)

[Bases de datos basadas en clave-valor 44](#_Toc172780102)

[Bases de datos orientadas a documentos 47](#_Toc172780103)

[En base de datos relacional 49](#_Toc172780104)

[Bases de datos no relacionales documentales 51](#_Toc172780105)

[Bases de datos orientada a grafos 55](#_Toc172780106)

[4.4. Sistemas de Gestión de Bases de Datos (SGBD) NoSQL 55](#_Toc172780107)

[5. NoSQL con MongoDB 58](#_Toc172780108)

[5.1. Conceptos básicos 58](#_Toc172780109)

[5.2. Consola interactiva 59](#_Toc172780110)

[5.3. Crear primer documento 63](#_Toc172780111)

[Primera colección 64](#_Toc172780112)

[6. Manipulación de datos con MongoDB 66](#_Toc172780113)

[Crear documentos 66](#_Toc172780114)

[Leer y consultar colecciones 68](#_Toc172780115)

[Actualizar documentos 70](#_Toc172780116)

[Borrar documentos 71](#_Toc172780117)

[Síntesis 72](#_Toc172780118)

[Material complementario 73](#_Toc172780119)

[Glosario 74](#_Toc172780120)

[Referencias bibliográficas 76](#_Toc172780121)

[Créditos 78](#_Toc172780122)

Introducción

En el mundo actual, la gestión eficiente de datos es fundamental para el éxito de cualquier organización. Este componente está diseñado para proporcionar una comprensión integral de las bases de datos SQL y NoSQL, destacando sus características, diferencias y aplicaciones prácticas.

Las bases de datos SQL, como MySQL, se caracterizan por su estructura relacional y el uso del lenguaje SQL para la gestión de datos. MySQL es uno de los sistemas de gestión de bases de datos más populares, conocido por su robustez, flexibilidad y facilidad de uso. Este componente cubre desde la configuración del entorno de trabajo hasta el diseño y administración de bases de datos utilizando MySQL y MySQL Workbench. Los usuarios aprenderán a diseñar modelos de datos, ejecutar consultas SQL, y realizar tareas administrativas esenciales para la gestión de bases de datos.

Por otro lado, las bases de datos NoSQL surgieron para abordar las limitaciones de escalabilidad y flexibilidad de las bases de datos relacionales tradicionales. Estas bases de datos no utilizan SQL como su principal lenguaje de consultas y se adaptan mejor a grandes volúmenes de datos y aplicaciones en tiempo real. Este componente explora los diferentes tipos de bases de datos NoSQL, como las basadas en clave-valor, orientadas a documentos y de grafos, destacando sus ventajas y casos de uso.

A lo largo del componente, los usuarios tendrán la oportunidad de aplicar los conceptos aprendidos mediante ejercicios prácticos y estudios de caso, como el diseño de un sistema de facturación y la gestión de matrículas en un instituto de formación técnica. Esto permitirá consolidar los conocimientos y desarrollar habilidades prácticas en el manejo de bases de datos SQL y NoSQL.

# Establecer un entorno de trabajo

Es esencial conocer cómo obtener, instalar, configurar y probar las herramientas necesarias para realizar las operaciones de diseño. Existen muchas tecnologías para diseñar e implementar bases de datos; en este caso, se utilizará MySQL como sistema gestor de bases de datos y MySQL Workbench como herramienta de diseño. Es importante que, a medida que se presentan los contenidos, realice de forma local y en su computadora el trabajo con los ejercicios y ejemplos propuestos.

## XML Extensible Markup Language

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual. Esto significa que cuenta con una licencia pública general y/o una licencia comercial por parte de Oracle Corporation, y según medios de comunicación, es considerada la base de datos de código abierto más popular del mundo. Tener licencia dual implica que se puede usar la licencia de uso público sin incurrir en gastos; no obstante, se puede adquirir una licencia de pago que incluye paquetes que robustecen el producto y servicios de soporte técnico especializado.

#### Proceso de descarga de MySQL Server Community y MySQL Workbench

Lo invitamos a consultar el PDF “**Proceso de descarga de MySQL Server Community y MySQL Workbench**", el cual se encuentra en la carpeta Anexos.

#### Proceso de instalación de MySQL Server Community

Lo invitamos a consultar el PDF “**Proceso de instalación de MySQL Server Community**", el cual se encuentra en la carpeta Anexos.

## Sistema de diseño y gestión de bases de datos

Es una herramienta visual unificada para arquitectos, desarrolladores y administradores de bases de datos. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, administración de usuarios, respaldo y muchas otras características y, lo más importante es que es multiplataforma, lo que indica que puede descargarse para sistemas operativos Windows, Linux y Mac.

Algunas de las características más importantes de Workbench son las siguientes:

1. **Diseño**

MySQL Workbench permite a un administrador de bases de datos, desarrollador o arquitecto de datos diseñar, modelar, generar y administrar bases de datos visualmente. Incluye todo lo que un modelador de datos necesita para crear modelos ER complejos, ingeniería directa e inversa, y también ofrece características clave para realizar tareas difíciles de gestión de cambios y documentación, que normalmente, requieren mucho tiempo y esfuerzo.

En Workbench se pueden crear nuevos modelos, lo que equivale a crear una nueva base de datos e ir agregando los objetos requeridos como tablas, atributos, tipos de datos, relaciones, entre otras.

1. **Desarrollar**

MySQL Workbench ofrece herramientas visuales para crear, ejecutar y optimizar consultas SQL. El editor de SQL proporciona resaltado de sintaxis de color, autocompletado, reutilización de fragmentos de código SQL e historial de ejecución de SQL. El panel de conexiones de base de datos permite a los desarrolladores administrar fácilmente las conexiones de base de datos estándar, incluido MySQL Fabric. El explorador de objetos proporciona acceso instantáneo al esquema y los objetos de la base de datos.

1. **Administrar**

MySQL Workbench proporciona una consola visual para administrar fácilmente los entornos MySQL y obtener una mejor visibilidad de las bases de datos. Los desarrolladores y administradores de bases de datos pueden utilizar las herramientas visuales para configurar servidores, administrar usuarios, realizar copias de seguridad y recuperación, inspeccionar datos de auditoría y ver el estado de la base de datos.

1. **Panel de rendimiento visual**

MySQL Workbench proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los administradores de bases de datos pueden ver rápidamente los indicadores clave de rendimiento mediante el panel de rendimiento. Los informes de rendimiento proporcionan una fácil identificación y acceso a puntos de acceso de E / S (entrada y salida), declaraciones SQL de alto costo y más. Además, con un clic, los desarrolladores pueden ver dónde optimizar su consulta con el plan visual explain mejorado y fácil de usar.

1. **Migración de bases de datos**

MySQL Workbench proporciona una solución completa y sencilla de usar para migrar Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL y otras tablas, objetos y datos RDBMS a MySQL. Los desarrolladores y administradores de bases de datos pueden convertir rápida y fácilmente las aplicaciones existentes para que se ejecuten en MySQL, tanto en Windows como en otras plataformas. La migración también admite la migración de versiones anteriores de MySQL a las últimas versiones.

## Sistemas basados en MySQL

El diseño de un cuestionario está directamente relacionado con el tipo de información que se desea obtener, ya sea información exploratoria o para probar una hipótesis. Son instrumentos de investigación que consisten en conjunto de preguntas u otro tipo de indicadores que son diseñados específicamente para recolectar información del usuario encuestado. Esta técnica incluye preguntas cerradas y abiertas, que al final se traducen como herramientas de índole cualitativas o cuantitativas.

A continuación, encontrará el PDF Manual de instalación de MariaDB en Windows. En él solo se instala el SGBD, por lo que debe instalar por aparte MySQL Workbench, para que administre las bases de datos de MariaDB.

#### Manual de instalación de MariaDB en Windows

Lo invitamos a consultar el PDF “**Manual de instalación de MariaDB en Windows**”, el cual se encuentra en la carpeta Anexos.

# Ejercicios de diseño e implementación

El aprendizaje basado en problemas es una buena estrategia a seguir en este tipo de actividades, debido a que es el análisis de un problema el que da origen a un modelo de base de datos. Por lo tanto, se propondrán dos problemas a considerar, y en torno a ellos se van reforzando los conocimientos adquiridos a la vez que se va profundizando en algunos conceptos. Para lograrlo, se toma como problema los casos comunes como lo son el análisis y diseño de un sistema de facturación en inventarios y un sistema para gestionar las matrículas de estudiantes en un instituto de formación técnica.

## Análisis y diseño sistema de facturación

#### Analicemos el primer caso, llamado Sistema de facturación.

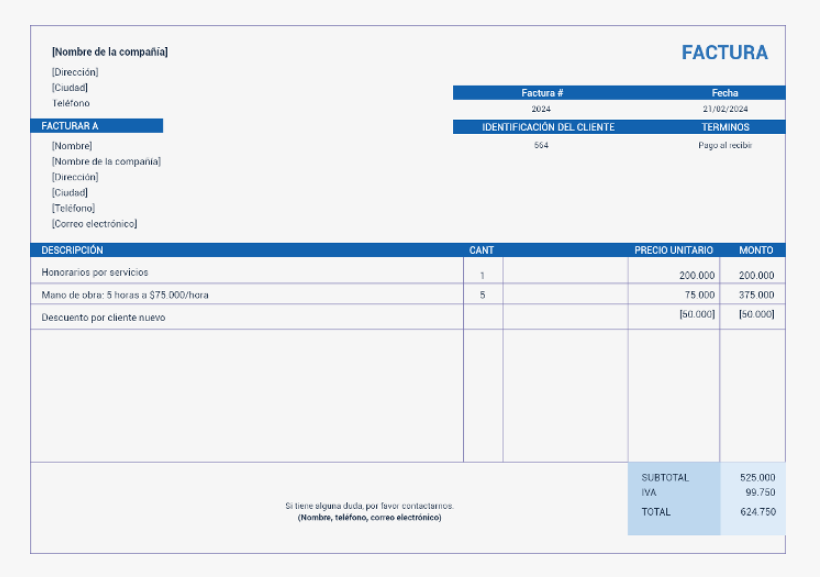
Suponga que lo contratan para diseñar una base de datos de una droguería, el objetivo principal es llevar las existencias de cada producto, a la vez que se va facturando lo que se vende, también interesa la información de los clientes, porque a futuro se quiere hacer campañas publicitarias de email marketing, según los productos que cada cliente consume.

1. Artículo más vendido.
2. Artículos que más utilidades dejan por unidad.
3. Empresa o persona que más factura (cantidad de factura).
4. Empresa o persona que más factura (volumen de ventas en pesos).
5. Productos sin existencia.

También fijó los productos que se organizan por categoría, porque le interesa saber el precio de compra y de venta, para conocer qué descuento hacer a una factura.

El cliente ofrece un modelo sobre cómo debe ser la factura, como se presenta en la siguiente figura.

1. Modelo de factura



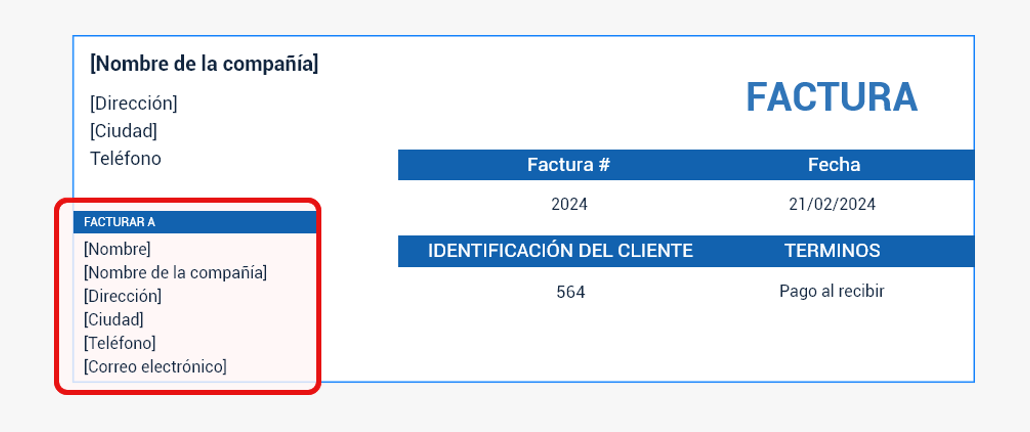
### Análisis y diseño

Una buena forma de empezar el análisis es con el formato de la factura; se inicia revisando la cabecera de la factura.

Un primer impulso puede hacer pensar que se debe crear una entidad para la compañía o empresa y pensar en definir una entidad o empresa con atributos “nombre\_compañia”, “dirección”, “ciudad”, etc.; sin embargo, considere lo siguiente:

El cliente le ha pedido que modele un sistema de facturación para su droguería, es decir, para una droguería, no para un conjunto de droguerías. Es diferente, si solicitaran “se necesita un sistema de facturación que administre el inventario y facturación de múltiples droguerías”, a estos problemas se les llama multitenant (multiinquilino), pero este no es el caso. Por lo tanto, no es la entidad droguería parte del modelo, porque no se está modelando la facturación de múltiples droguerías.

1. Factura con información resaltada

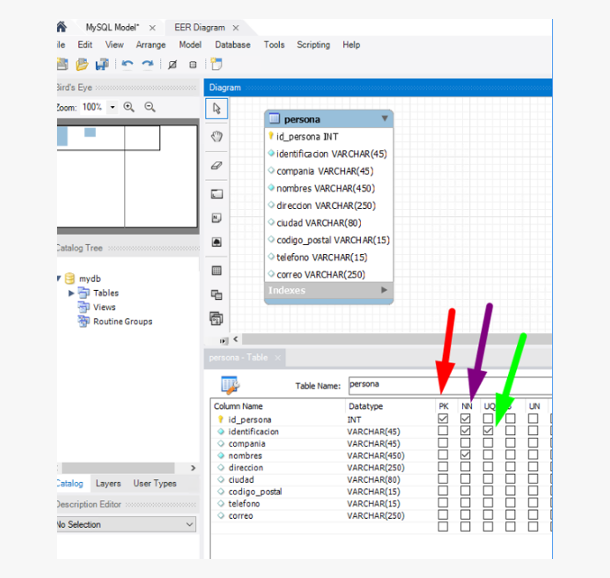


Tal parece que los datos más relevantes de los clientes están presentados en la parte resaltada de la imagen, lo que hace pensar que se requiere una entidad, cliente, usuario o persona y, que la persona puede tener relacionada una empresa o tal vez una persona hace la compra para una empresa. Lo anterior se puede modelar con una entidad, la entidad persona que relaciona los datos (nombre, compañía, dirección, ciudad, teléfono, correo electrónico).

Una forma simplificada sería creando una entidad persona y para poder identificar cada fila usar una llave primaria (id\_persona), pero también es importante usar un identificador único de persona como lo es el documento de identificación (porque las personas nunca tienen el mismo número), de igual forma, debe tener en cuenta que no puede existir una persona si al menos no se define el número de identificación y el nombre.

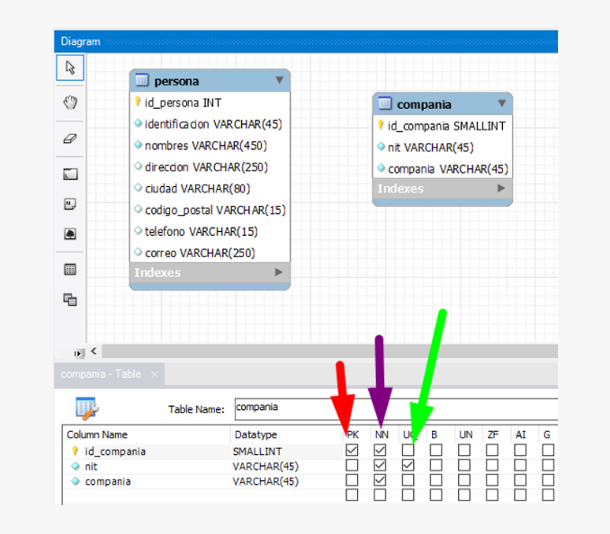
Lo anterior se puede identificar en la siguiente imagen (flecha roja para definir la llave primaria, morada para definir los datos que deben ser registrados obligatoriamente, y la verde una columna cuyo valor no puede repetirse en toda la tabla).

1. Identificación



Tal parece que los datos más relevantes de los clientes están presentados en la parte resaltada de la imagen, lo que hace pensar que se requiere una entidad, cliente, usuario o persona y, que la persona puede tener relacionada una empresa o tal vez una persona hace la compra para una empresa.

1. Creación de tabla con las compañías



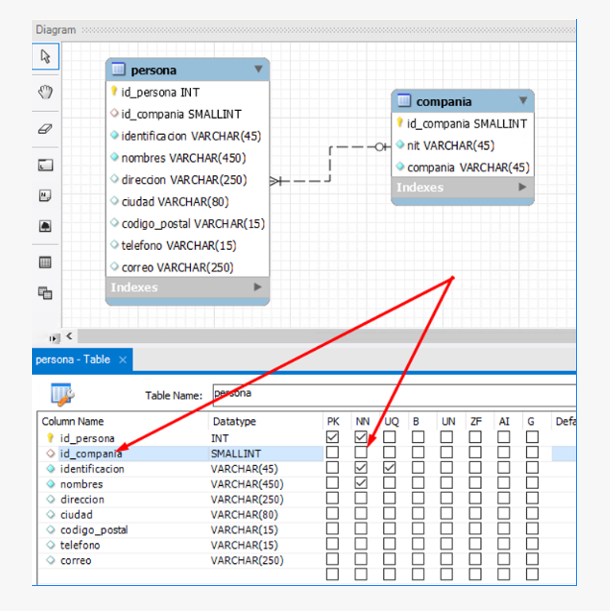
Como hay menos compañías que personas, se ha tomado id\_compania como SMALLINT. Esta decisión permite cumplir con dos requerimientos no funcionales:

* Ahorro de espacio de almacenamiento.
* Ahorro en el tiempo computacional de las consultas.

Nota: es importante comprender cómo la decisión de tipo de dato apunta a cumplir con los requerimientos no funcionales.

Es posible que piense que debe haber una relación entre la persona y la compañía de la siguiente forma:

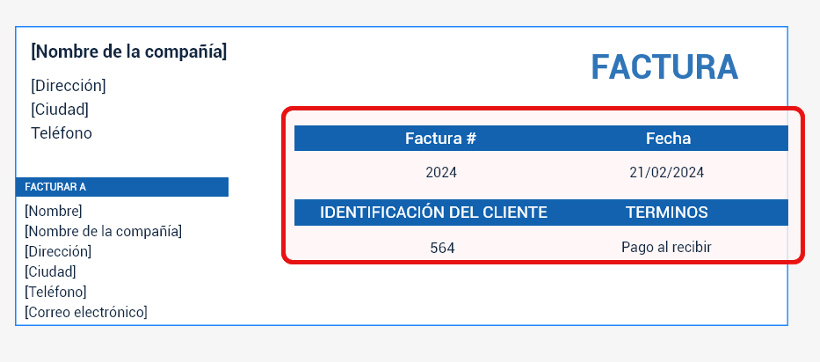
1. Relación entre la persona y la compañía



Al hacerlo de esta forma, una persona puede pertenecer a una empresa (solo a una), pero esa misma persona podría comprar en la farmacia unas veces a nombre propio y otras a nombre de la empresa, y este modelo no permitiría saber cada caso. Otro problema es que esa persona un día compra a nombre de una empresa y otro día puede comprar a nombre de otra empresa (si cambia de empresa) y al actualizar la empresa de la persona, se estarían actualizando las facturas pasadas, lo cual sería un error.

De esta forma, la manera correcta sería relacionar la persona y la empresa con la factura.

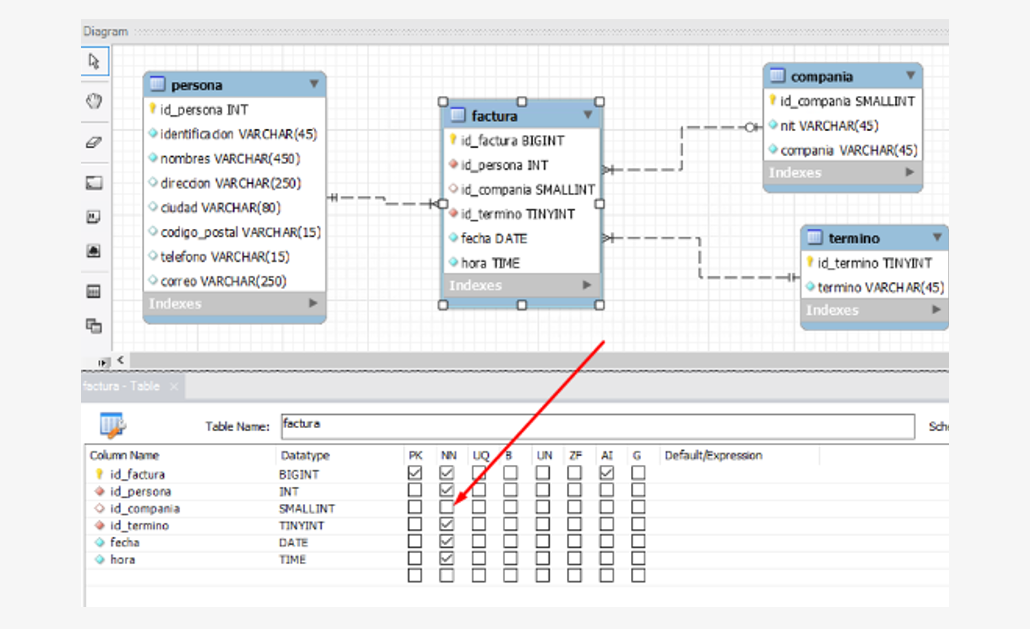
1. Información de la factura



Ahora, se presentan los datos de la factura y sus datos, como número de factura (que debe ser único y secuencial), fecha, la identificación del cliente (código interno id\_persona), y términos de pago.

Como se puede identificar en la siguiente imagen, se creó la tabla término de pago, que almacenará los medios de pago (crédito, contado, contra entrega, etc.), note también que se usó el tipo de datos TINYTINT que solo almacena números de 0-126 y se hace una relación de 1 a muchos (un término puede tener relacionadas muchas facturas, o lo que es igual muchas facturas tienen un mismo término de pago).

1. Creación de la tabla término de pagos



Note que una factura necesariamente debe tener relacionada la referencia de una persona, el término, la fecha y la hora, pero la compañía (id\_compania) no es obligatoria. Esto es porque hay facturas que las compran las personas naturales y otras que las compran las empresas o compañías, pero en un caso o en el otro, siempre habrá una persona que hace la compra, es por esto que el id\_persona es obligatoria.

La factura tiene un tipo de dato BIGINT, porque se espera que pueda facturar muchos al ser minorista, note también la columna AI (Auto Incrementable), esto hace que el gestor de base datos le asigne automáticamente y secuencialmente un número a cada fila para la columna id\_factura.

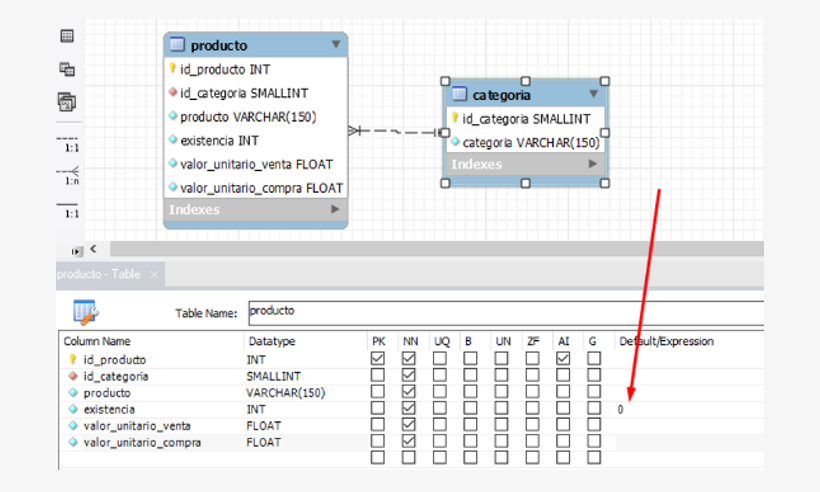
Otro segmento de la factura son los ítems o productos que la componen:

1. Ítems que componen la factura

| Descripción | Cantidad | Precio Unitario | Monto |
| --- | --- | --- | --- |
| Horarios por servicios | 1 | 200,00 | 200,00 |
| Mano de obra: 5 horas a 75€/hr | 5 | 75,00 | 375,00 |
| Descuento por nuevo cliente | - | (50,00) | (50,00) |

La tabla sugiere que los productos tienen una descripción, un precio unitario y como mencionó el cliente, un precio de compra:

1. Apartados que tienen los productos



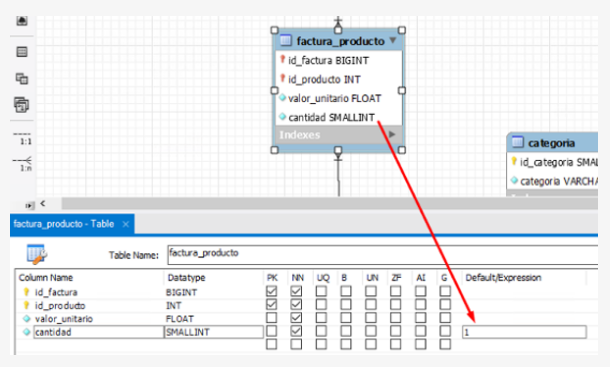
Una categoría puede tener varios productos o varios productos pertenecen a una categoría, esto es una relación de 1 a muchos, que queda definida en la imagen anterior. Note los tipos de datos definidos y también que la columna existencia no puede ser null, además de eso, si el usuario no pone un dato o trata de poner dato null en la existencia, entonces el sistema gestor de base de datos le pondrá “0” a esa fila en esa columna. La imagen sugiere que los productos tienen una descripción, un precio unitario y como mencionó el cliente, un precio de compra:

Ahora, resta relacionar los productos con la factura, y una factura puede relacionar varios productos, y un mismo producto puede estar en muchas facturas, por lo tanto, es una relación de muchos a muchos (n:m), esto hace que se genere una entidad intermedia que se llama producto\_factura.

La usabilidad es una medida de calidad que busca medir la eficacia, eficiencia y satisfacción en la experiencia del usuario con respecto a un producto, lo ideal es que este tipo de interacción entre usuario/producto sea lo más fácil y simple posible.

La entidad factura\_producto es una relación de muchos a muchos y agrega los datos necesarios para saber en cuánto y cuántos productos se venden por factura. También en la siguiente imagen se presenta cómo son obligatorias todas las columnas de este modelo y que la cantidad por defecto tiene valor de 1, debido a que si se intenta guardar un valor nulo, no debe ser posible y en lugar de insertar nulo insertará 1, porque no tiene sentido agregar un producto a una factura y no vender al menos un producto.

1. Columnas del modelo

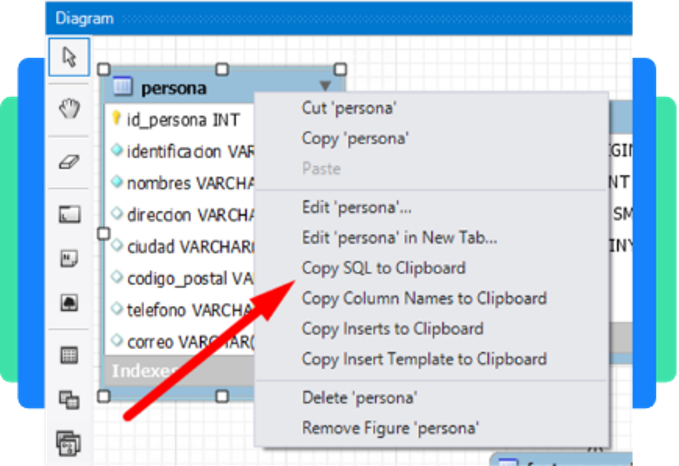


## Identificación de sentencias DDL

Suponga que lo contratan para diseñar una base de datos de una droguería, el objetivo principal es llevar las existencias de cada producto, a la vez que se va facturando lo que se vende, también interesa la información de los clientes, porque a futuro se quiere hacer campañas publicitarias de email marketing, según los productos que cada cliente consume.

MySQL Workbench ofrece la interfaz gráfica con la que se pueden generar estos códigos. Se mostrará cómo se hace con algunos ejemplos, siga los siguientes pasos:

1. **Haga clic derecho a la entidad persona y luego a “Copy to Clipboard” (copiar SQL en portapapeles).**
2. Copy to Clipboard



1. **Después se debe pegar (Ctrl+V) el contenido en un editor de texto y el resultado será:**

REATE TABLE IF NOT EXISTS `mydb`.`persona` (

`id\_persona` INT NOT NULL,

`identificacion` VARCHAR(45) NOT NULL,

`nombres` VARCHAR(450) NOT NULL,

`direccion` VARCHAR(250) NULL,

`ciudad` VARCHAR(80) NULL,

`codigo\_postal` VARCHAR(15) NULL,

`telefono` VARCHAR(15) NULL,

`correo` VARCHAR(250) NULL,

PRIMARY KEY (`id\_persona`),

UNIQUE INDEX `identificacion\_UNIQUE` (`identificacion` ASC) VISIBLE)

ENGINE = InnoDB;

Como puede apreciar, es posible obtener la definición de la base de datos en lenguaje SQL (DDL), si no se tuviera MySQL Workbench, se tendría que elaborar cada una de las sentencias.

Pruebe sus conocimientos a través del siguiente ejercicio, en el que identifique en la sintaxis:

* Cómo en SQL define una llave primaria.
* Cómo en SQL se define un índice único.

Es importante que se familiarice con estas sentencias, porque las operaciones de mantenimiento de bases de datos como adición o sustracción de columnas o tablas, son más fáciles cuando se hacen a través de la ejecución de scripts en lenguaje SQL.

**Nota**: como no se ha especificado el nombre de la base de datos MySQL Workbench pone por defecto `mydb` a la base de datos.

A continuación, se presenta una entidad más compleja, como lo es la entidad factura:

CREATE TABLE IF NOT EXISTS `mydb`.`factura` (

`id\_factura`BIGINT NOT NULL AUTO\_INCREMENT,

`id\_persona` INT NOT NULL,

`id\_compania`SMALLINT NULL,

`id\_termino`TINYINT NOT NULL,

`fecha`DATE NOT NULL,

`hora`TIME NOT NULL,

PRIMARY KEY(`id\_factura`),

INDEX`fk\_factura\_persona\_idx` (`id\_persona`ASC)VISIBLE,

INDEX`fk\_factura\_compania1\_idx` (`id\_compania`ASC)VISIBLE,

INDEX`fk\_factura\_termino1\_idx` (`id\_termino`ASC)VISIBLE,

CONSTRAINT`fk\_factura\_persona`

FOREIGN KEY(`id\_persona`)

REFERENCES`mydb`.`persona` (`id\_persona`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT`fk\_factura\_compania1`

FOREIGN KEY(`id\_compania`)

REFERENCES`mydb`.`compania` (`id\_compania`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT`fk\_factura\_termino1`

FOREIGN KEY(`id\_termino`)

REFERENCES`mydb`.`termino` (`id\_termino`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

Observe cómo las llaves foráneas les define un índice:

INDEX `fk\_factura\_persona\_idx` (`id\_persona`ASC)VISIBLE,

Con esta sintaxis, lo que hace MySQL Workbench es decirle al gestor de base de datos MySQL que esta columna id\_persona debe guardarse en estructura de árbol binario (que se llame fk\_factura\_persona\_idx) y no de manera secuencial. Este tipo de estructura de datos en árbol mejora la velocidad de las operaciones de consulta y búsqueda.

**Nota**: usar MySQL Workbench ayuda a definir la estructura de almacenamiento de los datos para que cumpla con el requerimiento no funcional de rendimiento en las consultas.

También se crean índices para las columnas id\_compania e id\_termino.

CONSTRAINT`fk\_factura\_persona`

FOREIGN KEY(`id\_persona`)

REFERENCES`mydb`.`persona` (`id\_persona`)

ON DELETE NO ACTION ACTION,]

A continuación, se presenta línea a línea, recordando que la columna id\_persona es una llave foránea de la tabla persona:

1. Restricciones

| Nombre restricción | Descripción |
| --- | --- |
| CONSTRAINT `fk\_factura\_persona` | Indica al SGDB que debe crear una restricción (CONSTRAINT) que se llama fk\_factura\_persona |
| FOREIGN KEY`(`id\_persona`) | Indica la columna que es llave foránea, es decir, que id\_persona es una referencia de una fila de otra tabla. |
| REFERENCES `mydb`.`persona` (`id\_persona`) | La fila de la otra tabla de la que es referencia de la base de datos mydb, tabla persona columna id\_persona |
| ON DELETE NO ACTION | Qué acción debe hacer el SGDB si borran la fila de la tabla persona con los datos de la tabla factura que estén relacionados, acá dice que no le permite borrar personas que ya tienen facturas. |
| ON UPDATE NO ACTION, | Qué acción debe hacer el SGDB si actualiza la columna id\_persona una fila de la tabla persona con los datos de la tabla factura que estén relacionados, acá dice que no le permita actualizar la columna id\_persona de la tabla persona que ya tiene facturas. |

No es de preocuparse si no es muy claro hasta la decisión del SQL de llaves foráneas, dado que a lo largo de la práctica y la revisión del material complementario, podría complementar los diferentes elementos y conceptos.

## Generar la base de datos

A continuación, se presentará cómo obtener el modelo físico de una base de datos en formato natural de las bases de datos (esto es SQL). Con el siguiente procedimiento se obtendrá un modelo de datos en sentencias DDL, en un archivo con extensión .sql, estos documentos son los códigos fuente de las bases de datos, generalmente en el proceso de desarrollo de software son versionados como cualquier archivo de código fuente.

De acuerdo con lo anterior, una forma de generar la base de datos es la siguiente:

1. Ponga un nombre a la base de datos dando clic derecho en editar Schema.
2. Seleccione Ingeniería hacia adelante (Forward Ingingger).
3. Entre los datos de root base de datos.
4. Seleccione las opciones de la base de datos y se deja por defecto.
5. Luego, pide la contraseña de usuario root que se puso en el proceso de instalación de MySQL.
6. Después muestra un resumen de lo que creará.
7. Finalmente, le da en Next y el sistema generará el DDL de la base datos.

Se guarda un archivo “Save to file” y se copia al portapapeles “Copy to clipboard”. Se sugiere antes de dar Next y continuar el proceso, primero guardar como un archivo y ponerle este nombre, luego de ponerlo le da Next para que ejecute ese script en el SGDB instalado.

El archivo generado guardado en formato de texto con extensión .sql es lo que se llama el script DDL de la base de datos, es el principal producto de un modelo de datos, es el entregable de un proyecto y por lo que materialmente se puede pasar una factura al cliente.

## Operaciones de mantenimiento y actualización

Por lo general, la base datos se diseña y se despliega en servidores en la nube, hosting o en instancias de almacenamiento, sistemas operativos o de plataforma que no tienen interfaz gráfica, es por eso que es importante el archivo de texto DDL con extensión .sql, la base de datos hasta ahora generada presenta una inexactitud que se ha puesto para ejemplificar una operación de actualización o mantenimiento.

Suponga que el cliente ha contratado un servidor en la nube para la base de datos y al darle subir el archivo de extensión .sql ha generado la base de datos, pero, se ha dado cuenta de que su modelo está mal o incompleto y lleva varios meses trabajando la base de datos, hay muchos datos, por lo que no se puede volver a empezar con una base de datos vacía, sino que se debe ajustar la base de datos actual (base en producción) sin borrar o alterar los datos que la base ya tiene.

Además, junto a su equipo de trabajo ha detectado que no modelaron el pie de la factura:

#### Pie de la factura

Subtotal: 525,00

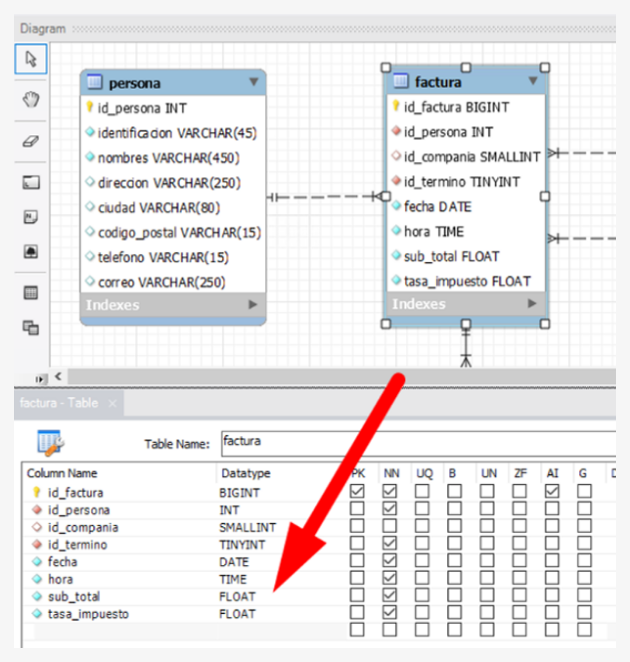
Tasa de impuesto: 4,250%

Impuesto: 22,31

TOTAL : 547,31 €

Este ajuste consiste en agregar en la tabla factura, el valor del subtotal y la tasa de impuesto a aplicar; con estos datos se puede calcular el total:

1. Información a agregar en la tabla factura



Ahora, se necesitan sincronizar estos cambios (los del diagrama) con la base de datos generada. De acuerdo con lo anterior, puede revisar los siguientes pasos:

1. Hacer clic en Database > Synchronize Model.
2. Seleccionar la conexión local.
3. Seleccionar los datos en la parte de generación de script adicionales.
4. Ingresar la contraseña para conectarse.
5. Analizar las bases de datos presentes y las características del servidor.
6. Seleccionar la base de datos con la que se va a sincronizar.
7. Analizar las diferencias.
8. Al ubicarse en cada una de las entidades o tablas, ignorar la actualización si es necesario.
9. Si se desea actualizar el modelo (diagrama) para que quede conforme a la base de datos, hacer clic en el botón “Update Model”.
10. Para actualizar la base de datos, hacer clic en Next.

Al final del proceso se debe tener el DDL donde se genera la base datos y el DDL donde se actualiza la tabla factura.

## Actualización por interfaz de línea de comandos

Muchas veces no es el computador local el que interesa actualizar, sino más bien, un servidor en la nube. Algunas plataformas tienen un entorno gráfico que permite aplicar el script con unos cuantos clics desde una interfaz web, pero la mayoría ofrece una interfaz de consola o terminal, de manera que los recursos computacionales no se desperdician en interfaces gráficas.

#### Verifique los siguientes archivos:

Lo invitamos a consultar los siguientes archivos: “**DDL\_drogueria**“, “**DDL\_update**” y “**Modelo Facturación Droguería**”, los cuales se encuentran en la carpeta Anexos.

A continuación, se presenta cómo aplicar el script desde una terminal. Esta base de datos se encuentra diseñada para cumplir el requerimiento funcional de administración de facturas de una droguería:

* **Paso 1**

Dar clic en el menú inicio, buscar MySQL, una terminal o interface de línea de comandados (Command Line Interface, CLI).

* **Paso 2**

A continuación, pedirá la contraseña de root.

* **Paso 3**

Luego, se debe seleccionar la base de datos con la sentencia SQL: use farmacia.

A continuación, se describe un problema del cual se pretende obtener un modelo de datos que permita gestionar la información, a través de un sistema gestor de base de datos, como se realizó en el caso anterior, donde se fue empleando una estrategia empírica, luego de conocer los conceptos de bases de datos.

Se presentará un problema más sencillo con el ánimo de concentrar esfuerzos en el análisis desde un enfoque metodológico, que se puede basar en tres fases:

1. **Diseño conceptual**

Encargado de entender el problema y la naturaleza de los datos (qué son y qué representan en el mundo real). Las actividades a desarrollar son:

* Identificar entidades.
* Identificar atributos.
* Identificar las relaciones.
* Identificar las cardinalidades de las relaciones.
* Diagrama entidad relación.

1. **Diseño lógico**

Se encarga de analizar la estructura de los datos y que se cumpla con los requisitos del negocio, modelo de negocio o problema. Las actividades a desarrollar son:

* Mapear diagrama entidad relación a un modelo relacional.
* Análisis de restricciones de integridad.
* Validación de requisitos funcionales.

1. **Diseño físico**

Se preocupa por cómo se almacenarán los datos. Las actividades a desarrollar son

* Definición de tipos de datos.
* Analizar las restricciones de llaves foráneas.
* Definición de índices.

Estudiemos ahora el caso 2, gestión de matrículas.

Se necesita un sistema que permita controlar las matrículas de un instituto de formación técnica profesional; donde interesa la información pertinente de los estudiantes como son los datos personales y la edad; y de los profesores parte de los datos personales como la titulación profesional o técnica y el año de graduación.

Cada programa técnico se desarrolla por semestres y un semestre tiene varias materias, cada semestre un estudiante puede matricular máximo 7 materias, un profesor puede orientar varias materias y la misma materia puede ser dictada por varios profesores.

Un estudiante puede estudiar varios programas, aunque no a la vez, y el estudiante se inscribe a un programa y cada año puede cursar 2 semestres (de enero a junio período 1, y de julio a diciembre período 2), si registra por semestre una matrícula con las materias a cursar. Algunos programas duran 5, 6 o 7 semestres.

### Diseño conceptual

Para la elaboración del modelo conceptual es fundamental identificar las entidades, luego los atributos de cada una de ellas, luego las relaciones entre entidades y, si se desea, se elabora un diagrama de entidad relación como resultado del diseño conceptual:

1. **Lista de entidades**

* Persona (estudiantes y profesores): existen personas estudiantes y personas profesores.
* Programa: representa la carrera técnica.
* Inscripción: de un estudiante a un programa.
* Materia.
* Matrícula: dura un semestre y puede contener máximo 7 materias.

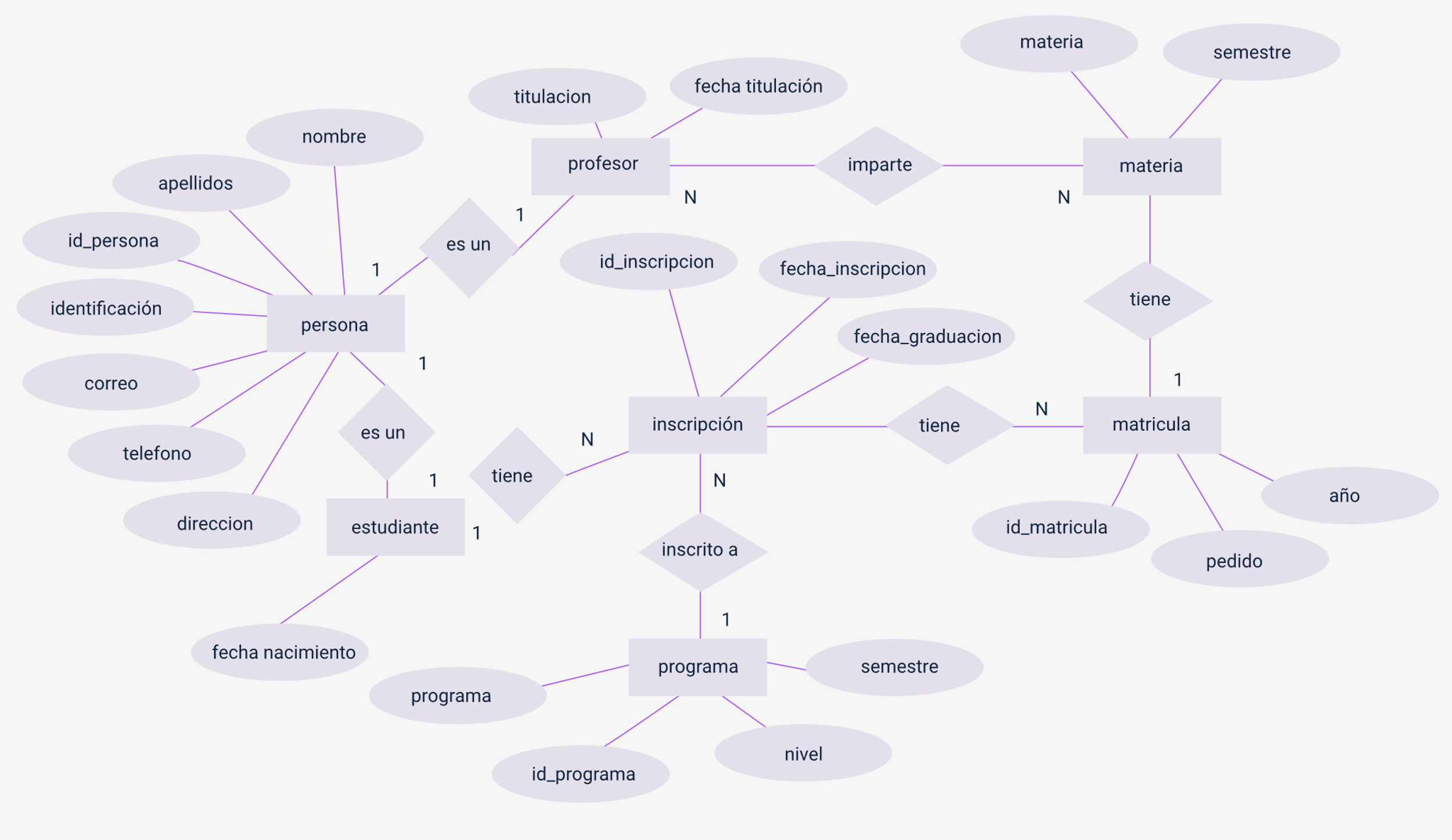
1. **Lista de atributos**

* Estudiante (nombre, apellidos, identificación, fecha\_nacimiento, correo, teléfono, dirección).
* Profesor (nombre, apellidos, identificación, correo, teléfono, dirección, titulación, fecha de titulación).
* Programa (nombre del programa, nivel, número de semestres).
* Inscripción (fecha, fecha de graduación).
* Materia (nombre de la materia y semestre).
* Matrícula (año, período).

1. **Relaciones y cardinalidad**

* El estudiante puede tener varias inscripciones, pero una inscripción por un solo estudiante.
* Un profesor puede dictar varias materias y una materia ser dictada por varios profesores.
* Una inscripción puede ser la relación entre un único estudiante y un único programa.
* Una inscripción puede tener varias matrículas, pero una matrícula corresponde a una única inscripción.
* Una matrícula se relaciona con varios estudiantes y un estudiante con varias matrículas.
* Una matrícula puede tener más de una materia, y la misma materia puede estar relacionada con varias matrículas.

1. Diagrama entidad relación



Actualmente, algunos diseñadores de bases de datos no diseñan el modelo entidad relación, por lo general directamente diseñan el modelo lógico o diagrama relacional, en algunos casos se emplea el modelo que se denomina modelo conceptual, que es equivalente al modelo entidad relación, pero difiere en la notación del diagrama al hacerse más parecido al modelo relacional. Por este mismo motivo es poco usado, porque supone hacer un nuevo paso o transformación del modelo conceptual a un modelo lógico, siendo este último el que en realidad puede convertirse en una base de datos.

### Diseño lógico

Una vez se tiene el modelo conceptual, se mapea el diagrama entidad relación o también llamado modelo conceptual a un modelo lógico, esto mediante las reglas de multiplicidad entre entidades (de uno a uno, de uno a muchos y de muchos a muchos).

Puede verificar algunos modelos, pero es recomendable que lo cree de manera que ejercite, lo hasta ahora visto.

Lo invitamos a consultar el archivo “**Modelos.mwb**”, el cual se encuentra en la carpeta Anexos.

Ahora, analicemos los siguientes pasos:

1. **Análisis de restricciones de integridad**

La entidad matrícula tiene relación con una inscripción, y esta a su vez con un estudiante y con el programa al que pertenece, también podría haberse realizado la matrícula con el programa y el estudiante, pero convenientemente, al relacionar matrícula con inscripción con menos datos, se está relacionando el programa y el estudiante de forma indirecta. Las demás relaciones responden al proceso de análisis de mapeo, por lo que no se entrará en detalles

1. **Validación de requisitos funcionales**

Para conocer si se cumplen los requisitos funcionales de lógica es importante prestar atención a los siguientes puntos:

* Revise por cada atributo si es conveniente que sea Not Null (Not null o NN), o si no nulo representa algo conceptualmente en el proceso.
* Si un atributo o columna es Not Null, revise si es importante definir un valor por defecto según la lógica del problema.

### Diseño físico

Finalmente, se debe revisar cuidadosamente la naturaleza de los datos antes de generar el archivo, conocer los tipos de datos que soporta el SGDB y estar seguro de que están bien definidos (si pueden no ser nulos, la longitud máxima y mínima, etc.):

1. **Definición de tipos de datos**

Revise:

* Que el valor (entero y flotante) corresponda a los valores que en un momento dado puede tener, por ejemplo, mire que las materias id\_materia es tipo SMALLINT (lo que permitiría 32767 materias) mientras que en personas id\_persona es tipo INT (lo que permitiría 2147483647 personas, es decir, el más del 30% de la población mundial).
* Preste especial atención en las cadenas de texto, tamaño máximo y mínimo en longitud.

1. **Analizar las restricciones**

Considere:

Este es el punto estratégico del modelo, revisar la no nulidad o nulidad de las llaves foráneas es de vital importancia, porque ello determina en gran porcentaje si su modelo cumple los requerimientos funcionales del sistema, por ejemplo, una inscripción debe tener relacionado un id\_estudiante y un id\_programa, porque la lógica de una inscripción es que un estudiante se matricula a un programa de formación, por lo tanto, ni id\_estudiante, ni id\_programa deben ser Null, por lo tanto, son atributo con restricción Not Null (NN).

1. **Definición de índices**

Valide:

Cómo se usa MySQL Workbench, esta hace por el diseñador muchas cosas, una de ellas es identificar los índices; sin embargo, si en algún momento decide hacer la base de datos en esta herramienta, no cree índices en los siguientes casos:

* Por cada llave foránea se crea un índice.
* Si va a hacer consultas por rango de fechas, a las columnas por las que va a filtrar por rango de fechas, créeles un índice.
* No es necesario crear índices para las llaves primarias, por defecto MySQL lo hace.

Si algún día usa otro motor de base de datos como Oracle, Postgres o SQL Server, verifique manualmente la creación de índices, porque según la herramienta que esté usando, puede ayudarle a crearlos para las llaves foráneas o no.

# Administración de bases de datos

Las funciones de un administrador de bases de datos (Data Base Administrator, DBA), es la gestión general de la base de datos.

Debe conocer las reglas de la tecnología particular, que en este caso sería MySQL, pero también está entre sus responsabilidades:

* **Validar o revisar el modelado y diseño de las bases de datos**

Debe concentrar su interés en los requerimientos de rendimiento y administración de índices.

* **Procesos de auditoría**

Identificar qué usuarios tienen acceso a qué tablas o recursos a insertar, actualizar o eliminar datos, y cuándo.

* **Resguardo y recuperación de datos**

Esto es hacer copias de seguridad periódicas de los datos, mantenerlos a salvo de la destrucción accidental o intencional. Además, diseñar, implementar y probar un plan de recuperación, considerando que pueden presentarse problemas, y validar los datos que se pueden restaurar rápidamente (que las copias generadas sean útiles).

* **Planificación de capacidad**

Gestionar el volumen de datos y diseñar los planes apropiados para administrarlos, esto ya que los datos crecen rápidamente y se debe llevar un registro de la taza de este crecimiento.

* **Administración de cambios**

A menudo es la responsabilidad del DBA realizar el análisis de impacto antes de generar los cambios dentro de un SGDB.

De las anteriores actividades, se puede decir que al validar o revisar el modelado y diseño, ya se cuenta con la competencia para hacerlo, al igual que la administración de cambios, cuando se vio operaciones de mantenimiento y actualización por el MySQL Workbench o por la interfaz línea de comandos.

## Copia de seguridad de base de datos con MySQL Workbench

Durante todo el ciclo de vida del software (desde que se construye hasta que está en operación), se necesitan operaciones de copias y restablecimiento de las bases de datos. Es por esto que se explicará una de las formas para obtener una copia de la base de datos, tanto en estructura como de los datos:

* Antes de cualquier cosa, se debe hacer una conexión a la base de datos si no existe dando clic en la casa y luego en el icono más “+”.
* Posteriormente, en test connection y luego en él se escribe la contraseña del root.
* Se le da doble clic a la conexión creada.

Luego, se da clic en la pestaña, administración y Data Export.

Se debe seleccionar:

* La base de datos a respaldar.
* Si se va a restaurar la estructura (metadatos) o solo los datos, o solo la estructura (se selecciona estructura y datos).
* La copia en un solo archivo y la ruta.
* Se crea la base de datos si no existe.
* Finalmente, en la carpeta se generará el backup.
* El proceso termina diciendo que la copia ha terminado.

## Restaurar una copia de seguridad con MySQL Workbench

El proceso complementario de restablecer una copia de base de datos es fundamental para la instalación de un sistema o el restablecimiento de la operación, cuando el sistema falla o se quiere volver a una versión anterior de la base de datos. Es por esto que se presentará un método sencillo para restablecer la base de datos a partir de un backup en formato SQL.

Siga los siguientes pasos luego de conectar a la base de datos con MySQL Workbench como se presenta a continuación.

1. Ingrese a la pestaña de administración.
2. Seleccione Data Import/Restore.
3. Marque la opción Import from Self-Contained File.
4. Busque el archivo generado antes.
5. Seleccione la base de datos a restaurar.
6. Haga clic en el botón Star Import.

Hasta este punto, puede tener la seguridad que está en capacidad de respaldar y restablecer la base de datos, por lo menos en un entorno de desarrollo y producción para bases pequeñas. Este proceso le puede servir para bases de datos grandes en las que debe usar la interfaz de comando “línea” con el comando “source”. Esto debido a que si la copia de seguridad es del tamaño de gigas, MySQL Workbench no podrá abrirlo y puede bloquearse, mientras que por la interfaz de línea de comando, el tamaño de la copia puede ser tan grande como se desee.

# Gestores de contenido CMS

NoSQL (“no SQL”) comprende una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas, como tablas; normalmente, no soportan operaciones JOIN ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y, habitualmente, escalan bien horizontalmente, es decir, pueden aumentar la estructura de manera sencilla. Los sistemas NoSQL se denominan, a veces, “no solo SQL” para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

## Definición de NoSQL

La web 2.0 (donde quienes suministran el contenido son los usuarios) trajo consigo grandes necesidades de datos e infraestructura tecnológica, principalmente para empresas como Google, Amazon y Facebook. Estas empresas tuvieron que buscar soluciones propias a estos problemas, enfocándose en la gran cantidad de almacenamiento y el crecimiento de sistemas web con acceso a información en tiempo real. Surgió la necesidad de proporcionar información gestionada y procesada desde grandes volúmenes de datos. Dado que las bases de datos relacionales no respondían de forma adecuada a esta necesidad, especialmente en tiempos de alta demanda, nacen las bases de datos NoSQL.

* **Escalabilidad**

Es la capacidad que tiene un sistema de ser configurado en software o hardware, para mejorar la respuesta a las exigencias o nuevas demandas de capacidad de procesamiento o almacenamiento (conservando las mismas funcionalidades). Actualmente, se usa mucho el término ‘escalabilidad vertical’, para referirse a este tipo de escalabilidad.

* **Escalabilidad horizontal**

Consiste en potenciar el rendimiento del sistema desde un aspecto de mejora global, a diferencia de aumentar la potencia de una única parte del mismo. Este tipo de escalabilidad se basa en la modularidad de su funcionalidad (Junta de Andalucía, s. f.), es decir, que las funcionalidades se puedan extender con la misma facilidad con la que mejoran el rendimiento y capacidad.

* **Escalabilidad vertical y horizontal**

Las bases de datos no relacionales están diseñadas para ofrecer sistemas con altas capacidades de escalabilidad vertical y horizontal.

## Cómo diferenciar NoSQL de SQL

Al utilizar el término NoSQL no se hace referencia a una base de datos específica o a un tipo particular de base de datos, sino que define un conjunto de tipos de bases de datos que son diferentes a las bases de datos convencionales (relacionales). La principal diferencia entre las bases de datos NoSQL y las bases de datos relacionales tradicionales es que las NoSQL emplean una forma de almacenamiento no estructurado o estructuras diferentes a tablas. Simplificando la estructura de almacenamiento, las NoSQL pueden concentrarse en procesar grandes volúmenes de datos.

A continuación, se presenta una comparación rápida de las funciones de NoSQL con las bases de datos relacionales.

1. Comparación de las funciones de NoSQL

| Característica | Base de datos NoSQL | Precio Unitario |
| --- | --- | --- |
| Desempeño | Alto | Bajo |
| Fiabilidad | Pobre | Buena |
| Disponibilidad | Buena | Buena |
| Consistencia | Pobre | Buena |
| Almacenamiento de datos | Optimizado para muchos datos | Mediano o grande |
| Estabilidad | Alta | Alta (pero más costosa) |

A continuación, se presentan algunas ventajas y desventajas que tiene cada uno de estos tipos de bases de datos:

* **Ventajas**

Contrario a las bases de datos relacionales, las NoSQL están basadas en key-value pairs (clave valor).

Las NoSQL tienen diferentes tipos de almacenamiento, como el almacén de columnas, documentos, claves, valor de gráficos (capas), objetos, en formato XML y otros modos de tipos de datos. Las bases de datos

NoSQL de código abierto tienen una implementación rentable económicamente, al no requerir licencias ni hardware de precio alto

Trabajando con bases de datos NoSQL, la expansión de funcionalidades es más fácil y económica que usando bases de datos relacionales.

* **Desventajas**

Casi todas las bases NoSQL no admiten funciones de fiabilidad o integridad de los datos, mientras que las bases de datos relacionales, sí. Las bases NoSQL no soportan las funcionalidades o restricciones de consistencia en los datos. Para garantizar la integridad, es responsabilidad de los desarrolladores implementar con código propio, lo que agrega más complejidad al sistema.

La falta de integridad referencial en los datos hace que no sean fiables para realizar transacciones seguras, como en los sistemas bancarios, donde la integridad de los datos es primordial. Otras formas de complejidad encontradas en la mayoría de las bases de datos NoSQL incluyen la incompatibilidad con consultas SQL; se necesita un lenguaje de consulta manual, haciendo los procesos mucho más lentos y complejos.

## Tipos de bases de datos NoSQL

Dentro de las bases de datos no relacionales, se encuentran distintos modelos o tipos de implementaciones. Este grupo de bases de datos no relacionales NoSQL desempeña funciones para diferentes ambientes y con necesidades variadas. Existen algunos tipos más comunes que otros, ya que están orientadas a documentos por permitir mejor escalabilidad horizontal (Macarrón, 2021)

### Bases de datos basadas en clave-valor

Las bases de datos basadas en clave-valor son un tipo de base de datos NoSQL sencillo, que sirve para insertar y consultar datos. El concepto de ‘clave-valor’ significa que los datos se almacenan como una colección de pares clave-valor; la clave identifica de forma única la colección y el valor contiene los datos, los cuales se ordenan según la clave.

#### Ejemplo

Supóngase que se quieren almacenar datos de una persona, como su cédula, nombres, apellidos, correo y varios números de teléfono, y que, por ejemplo, necesita guardar los datos para las siguientes personas:

Cédula: 1029283762

Nombre: JUAN CARLOS

Apellidos: MARTÍNEZ

Correo: juancho@nerss.et

Teléfono: 3150817273

Teléfono: 3168272372

Cédula: 98765322

Nombre: DANIELA

Apellidos: VELÉZ LEMOS

Correo: dnala@nerss.et

Se toma la cédula como una especie de llave para identificar a una persona, es decir, la cédula es el identificador único para cada persona y no hay dos personas con la misma cédula.

#### Más posibilidades

En la misma base de datos, se necesita guardar datos de carros, siendo la placa lo que permite identificar al vehículo; entonces, los datos a guardar serían:

1. Placa: MSR-13
2. Modelo: 2015
3. Marca: MAZDA
4. Tipo: SEDAN

La base de datos sería:

1. Ejemplo 2 de base de datos

| Clave | Valor |
| --- | --- |
| persona/cédula | 1029283762 |
| persona/1029283762/nombres | JUAN CARLOS |
| persona/1029283762/apellidos | MARTÍNEZ |
| persona/1029283762/correo | juancho@nerss.et |
| persona/1029283762/teléfono | 3150817273 |
| persona/1029283762/teléfono | 3150817273 |
| carro/placa | MSR-13 |
| carro/MSR-13/modelo | 2015 |
| carro/MSR-13/marca | MAZDA |
| carro/MSR-13/tipo | SENAN |
| persona/cédula | 98765322 |
| persona/98765322/nombres | DANIELA |
| persona/98765322/apellidos | VELEZ LEMOS |
| persona/98765322/correo | dnala@nerss.et |

Estas bases de datos son óptimas para consultas en ambientes de respuesta rápida y para aplicaciones en tiempo real, como sistemas de telefonía IP o sistemas de chat.

### Bases de datos orientadas a documentos

Los SGBD con estas características son denominados bases de datos documentales y funcionan en el marco de la definición de un “documento”. La mayoría de estos SGBD coinciden en cómo se almacena la información, empleando un formato estándar. Los formatos estándares más comunes o populares son JSON, BSON y XML. Este último tipo de formato es uno de los más utilizados en la actualidad.

#### Formato XML

XML, sigla en inglés de eXtensible Markup Language, traducido como “Lenguaje de Marcado Extensible” o “Lenguaje de Marcas Extensible”, es un metalenguaje que permite definir lenguajes de marcas. Desarrollado por el World Wide Web Consortium (W3C), se utiliza para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

#### Modelar información

Suponga que se desea modelar la información contenida en un mensaje de correo electrónico.

<Edit\_Mensaje>

<Mensaje>

<Remitente>

<Nombre>Nombre del remitente</Nombre>

<Mail>Correo del remitente</Mail>

</Remitente>

<Destinatario>

<Nombre>Nombre del destinatario</Nombre>

<Mail>Correo del destinatario</Mail>

</Destinatario>

<Texto>

<Asunto>

Este es mi documento con una estructura muy sencilla no contiene atributos ni entidades…

</Asunto>

<Parrafo>

Este es mi documento con una estructura muy sencilla no contiene atributos ni entidades...

</Parrafo>

</Texto>

</Mensaje>

</Edit\_Mensaje>

### En base de datos relacional

Si se quisiera modelar esto en una base de datos relacional, seguramente se definiría una tabla para personas (con atributos nombre y correo) y una tabla para los correos, con dos relaciones a la tabla persona: una para el remitente y otra para el destinatario.

1. Persona: (nombres, correo)
2. Correo: (asuntos y texto)

Sin embargo, en XML, en lugar de tener filas en las tablas, cada correo es un archivo con la estructura dada en el ejemplo.

#### Formato JSON

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

1. Tipos de datos en JSON

| Datos | Definición |
| --- | --- |
| Cadenas de texto | Al ser una secuencia de ninguno o más caracteres, se ponen entre doble comilla, como en lenguaje JavaScript. Ejemplo: “Hola Mundo”. |
| Números | Se permiten números negativos y positivos, que también pueden contener parte no entera separada por puntos. Ejemplo: 654.321. |
| Arreglos | Representan una lista con ninguno o más valores, los cuales pueden ser todos del mismo o de diferente tipo. Los valores separados por comas y el vector, entre llaves ([]). Ejemplo: [“Luis”, “Carlos”, “Julian”, 99]. |
| Booleanos | Representan valores booleanos y pueden tener dos valores: true y false. |
| Valores nulos | Representan el valor nulo (null).  Son colecciones no ordenadas de pares de la forma <nombre>:<valor>, donde el nombre debe ser separado por comas y puestos entre corchetes ({}). |

### Bases de datos no relacionales documentales

Las bases de datos no relacionales documentales utilizan el concepto de guardar la información con el modelo clave/valor. Por esta razón, los documentos más comunes son en formato JSON. La diferencia radica en que un documento se almacena en una clave. Esta definición puede parecer abstracta, pero se explica mejor con un ejemplo:

{

“nombres”: “JUAN CARLOS”,

“apellidos”: “MARTINEZ”,

“fecha\_nacimiento”: “1995-05-15”,

“edad”: 65,

“saldo”: 9876.134,

“hobbies”: [“música”, “futbol”, “leer”],

“correo”: “juancho@nerss.et”,

“pedidos”: [],

“acreedores”: null

}

Note cómo “nombres” es la clave y “JUAN CARLOS” es el valor. De la misma forma, “edad” es la clave y 65 el valor.

**Ejemplo**

Suponga que necesita modelar un menú de una aplicación como se presenta a continuación:

1. Ejemplo



#### Representación en formato BSON

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

**Xml**

<menu id="file" value="Archivo">

<popup>

<menuitem value="Nuevo" onclick="CreateNewDoc()" />

<menuitem value="Abrir" onclick="OpenDoc()" />

<menuitem value="Cerrar" onclick="CloseDoc()" />

<menuitem value="Guardar" onclick="SaveDoc()" />

</popup>

</menu>

**json**

{

“menu”: {

“id”: “file”,

“value”: “Archivo”,

“popup”: {

“menuitems”: [

{

“value”: “Nuevo”,

“onclick”: “CreateNewDoc()”

},

{

“value”: “Abrir”,

“onclick”: “OpenDoc()”

},

{

“value”: “Cerrar”,

“onclick”: “CloseDoc()”

},

{

“value”: “Guardar”,

“onclick”: “SaveDoc()”

}

]

}

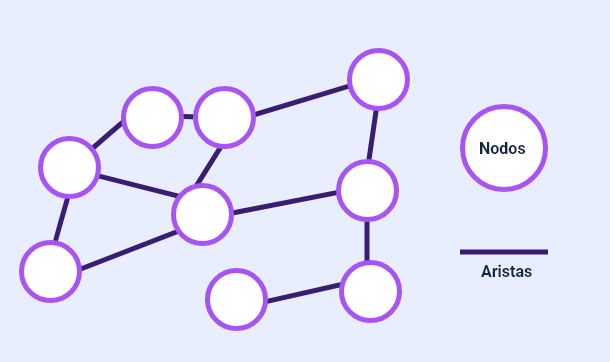
}

}

### Bases de datos orientada a grafos

Este tipo de base de datos es muy diferente a los anteriores; maneja la información de una manera particular, empleando grafos y teoría de grafos. Cada nodo solo debe contener una sola columna, por lo tanto, se deben normalizar completamente las bases de datos. Solo aplica para grafos de tipo binario; cada nodo solo se puede relacionar con otros dos nodos como máximo. Las ventajas de este tipo de bases de datos se enfocan en la integridad de los datos y la optimización de consultas.

1. Grafos



## Sistemas de Gestión de Bases de Datos (SGBD) NoSQL

Como se sabe, la base de datos solo sirve si existe un sistema que la gestione, un SGBD. En NoSQL, hay varios SGBD y la mayoría son de código libre o de libre distribución. A continuación, se presentan algunos de los más populares, con sus respectivas características. Es necesario tener en cuenta que, no obstante, pueden existir otros SGBD menos populares pero que pueden ser una buena opción para trabajar con Big Data.

1. Sistemas de gestión de bases de datos

| Tipo | Logotipo | Nombre | Lanzamiento | Licencia |
| --- | --- | --- | --- | --- |
| Documental | Imagen del logotipo de MongoDB | MongoDB | 2008 | Apache License 2.0 |
| Documental | Imagen del logotipo de CounchDB | CouchDB | 2005 | Apache License 2.0 |
| Documental | Imagen del logotipo de RavenDB | Raven DB | 2010 | GNU AGPL v3.0 |
| Clave/Valor | Imagen del logotipo de Apache Cassandra | Apache Cassandra | 2010 | Apache License 2.0 |
| Clave/Valor | Imagen del logotipo de Riak | Riak | 2009 | Apache License 2.0 |
| Grafos | Imagen del logotipo de Redis | Redis | 2009 | Licencia: BSD |
| Grafos | Imagen del logotipo de Neo4j | Neo4j | 2009 | GNU AGPL v3.0 |
| Grafos | Imagen del logotipo de Dex | Dex | 2008 | Comercial |
| Grafos | Imagen del logotipo de Sones DB | Sones GraphDB | 2012 | GNU AGPL v3.0 y comercial |

# NoSQL con MongoDB

Como se mencionó antes, MongoDB es un sistema de bases de datos no relacionales y multiplataforma, es decir, se puede instalar en cualquier sistema operativo para servidor (Windows, Linux, BSD, MAC). El tipo de base de datos empleado es el documental, pero también se pueden administrar bases de datos clave/valor. El tipo de licencia de software libre es, específicamente, GNU AGPL v3.0. MongoDB usa el formato BSON (JSON compilado) para guardar la información.

## Conceptos básicos

Los siguientes conceptos son clave para la asimilación de todos los elementos de este componente formativo. Para la implementación de bases de datos, conviene tener claridad acerca de lo que es un documento, una colección y a qué se hace referencia cuando se habla de creación y actualización de bases de datos, instalación de licencias, entre otros.

* **JSON – JavaScript Object Notation**

Como la información gestionada usa el formato JSON para representar objetos según la especificación del documento RFC 46274, y como JSON es independiente del lenguaje, se ha popularizado mucho el uso del lenguaje JavaScript. JSON es usado por muchos sistemas para intercambiar datos, por lo sencillo que es, si se compara con XML.

* **Documento**

Un documento es un conjunto de datos estructurados, pero esta estructura no es estática, es decir, puede variar de un documento a otro. La estructura básica contiene pares clave/valor y se usa BSON (JSON Binario) como formato para almacenar los documentos dentro del SGBD. Un documento es como el equivalente a una fila dentro de una tabla de un modelo relacional.

* **Colección**

En este contexto, cuando se habla de colección, se está haciendo referencia a un conjunto de documentos. Se puede asemejar a una tabla en las bases de datos relacionales.

## Consola interactiva

MongoDB tiene, al igual que todas las bases de datos, una interfaz de línea de comando (CLI), desde la cual se puede acceder a un API de las funcionalidades. Podemos acceder a su consola interactiva y realizar nuestras primeras interacciones con MongoDB.

* **Ingreso**

Para ingresar a su consola interactiva, hay que dirigirse a la carpeta donde quedó instalado MongoDB. (C:\Program Files\MongoDB\Server\4.4\bin). Se debe dar clic en el ejecutable que se llama mongo.exe.

* **Instalación exitosa**

Si la vista es como se presenta a continuación, es porque la instalación fue exitosa.

MongoDB shell version v4.4.6

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { “id” : UUID(“e805b48a-4059-47e6-ae1f-89cb7bcb19c2”) }

MongoDB server version: 4.4.6

Welcome to the MongoDB shell.

For interactive help, type “help”.

For more comprehensive documentation, see

https://docs.mongodb.com/

Questions? Try the MongoDB Developer Community Forums

https://community.mongodb.com/

—

The server generated these startup warnings when booting:

2021-06-21T21:35:16.749-05:00: Access control is not enabled for the database. Read and write access brto data and

configuration is unrestricted

—

Enable MongoDB’s free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

—

>

* **Prueba**

Para probar que está funcionando bien, se escribe el comando help.

> help

db.help() help on db methods

db.mycoll.help() help on collection methods

sh.help() sharding helpers

rs.help() replica set helpers

help admin administrative help

help connect connecting to a db help

help keys key shortcuts

help misc misc things to know

help mr mapreduce

show dbs show database names

show collections show collections in current database

show users show users in current database

show profile show most recent system.profile entries with time >= 1ms

show logs show the accessible logger names

show log [name] prints out the last segment of log in memory, 'global' is default

use <db\_name> set current database

db.mycoll.find() list objects in collection mycoll

db.mycoll.find( { a : 1 } ) list objects in mycoll where a == 1

it result of the last line evaluated; use to further iterate

DBQuery.shellBatchSize = x set default number of items to display on shell

exit quit the mongo shell

* **Seleccionar base de datos**

Al igual que en bases de datos relacionales, se debe usar un comando para seleccionar una base de datos. Antes de eso, es necesario mirar qué bases de datos existen con el siguiente comando: show dbs.

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

* **Comando de selección**

El comando para seleccionar una base de datos es use <nombre base de datos>.

> use local;

switched to db local

* **Crear base de datos**

El comando use se usa también para crear una nueva base de datos. El comando sería use pruebas.

< use pruebas;

switched to db pruebas

## Crear primer documento

Se creará una base de datos y una colección, y eso se hará almacenando un documento usando el objeto db. Sin embargo, primero se debe ejecutar el comando use. Un documento puede tener, en teoría, un máximo de hasta 16MB de información.

Aprovechando el primer objeto visto, JSON, para almacenarlo en MongoDB y consultarlo, se debe tener en cuenta, dentro de la consola interactiva, usar variables para crear o modificar documentos; de esta manera, se pueden evitar accidentes con una mala manipulación directa de la base de datos.

Ejemplo de uso:

1. **Ejecutar el comando use para seleccionar o crear una base de datos:**

use miBaseDeDatos

1. **Crear una colección y almacenar un documento usando una variable:**

var miDocumento = {

nombre: “Juan”,

apellidos: “Pérez”,

edad: 30,

correo: “juan.perez@example.com”

};

db.miColeccion.insert(miDocumento);

Este método ayuda a mantener el control sobre los datos y evita errores al manipular directamente la base de datos.

### Primera colección

Es así como se ha creado la primera colección de nombre colección personas y nuestro primer documento. Para confirmar esto, se puede ejecutar el comando show collections.

< show collections

coleccion\_personas

>

#### Consulta de documentos:

{

“\_id”: ObjectId(“60d154ca1e0ed70729ced13d”),

“nombres”: “JUAN CARLOS”,

“apellidos”: “MARTINEZ”,

“fecha\_nacimiento”: “1995-05-15”,

“edad”: 65,

“saldo”: 987.432,

“hobbies”: [“musica”, “futbol”, “leer”],

“correo”: “juancho@nerss.et”,

“pedidos”: [],

“acreedores”: null

}

# Manipulación de datos con MongoDB

Las bases de datos deben poder manipular la información de varias maneras, ya que los procesos en los sistemas informáticos necesitan de esas operaciones básicas: crear datos, consultarlos, actualizarlos y, en algún punto, eliminarlos (Create, Read, Update, and Delete - CRUD). Por tanto, queda establecido que almacenarla o contenerla no es su mera y única función básica.

### Crear documentos

Es la primera de cuatro funciones elementales a la hora de utilizar bases de datos. Crear documentos permite insertar unidades de información; en el caso de MongoDB, a través de documentos en colecciones de datos.

1. **Crear un documento: Para poder crear un documento, MongoDB posee los métodos .insert(), .save() y, de una manera especial, .update()**

documento = {

“titulo”: “Titulo Libro”,

“Autores”: [“Peter”, “Johnathan”]

}

{

“titulo”: “Titulo Libro”,

“Autores”: [“Peter”, “Johnathan”]

}

db.libros.insert(documento)

WriteResult({ “nInserted”: 1 })

1. **Insertar varios documentos simultáneamente: Se pueden insertar varios documentos a la vez.**

documentos = [{

“titulo”: “Otro Libro”,

“Autores”: [“Peter”, “Yolima”]

}, {

“titulo”: “Nuevo Libro”,

“Autores”: [“Zulema”]

}]

[{

“titulo”: “Otro Libro”,

“Autores”: [“Peter”, “Yolima”]

}, {

“titulo”: “Nuevo Libro”,

“Autores”: [“Zulema”]

}]

db.libros.insert(documentos) = [{

BulkWriteResult({

“writeErrors”: [],

“writeConcernErrors”: [],

“nInserted”: 2,

“nUpserted”: 0,

“nMatched”: 0,

“nModified”: 0,

“nRemoved”: 0,

“upserted”: []

})

}]

1. **Actualizar un documento: Este método tiene todas las funciones de .insert(), pero además permite actualizar un documento si ya existe el \_id de dicho documento. En ese caso, .insert() mostraría una excepción.**

### Leer y consultar colecciones

Para leer y consultar la información, la forma básica consiste en usar el método find() de la colección, tal como se mencionó anteriormente en este componente formativo.

1. **Leer y consultar colecciones**

> db.libros.find()

{ “\_id” : ObjectId(“60d159461e0ed70729ced13e”), “titulo” : “Titulo Libro”, “Autores” : [ “Peter”, “Johnathan” ] }

{ “\_id” : ObjectId(“60d159df1e0ed70729ced13f”), “titulo” : “Otro Libro”, “Autores” : [ “Peter”, “Yolima” ] }

{ “\_id” : ObjectId(“60d159df1e0ed70729ced140”), “titulo” : “Nuevo Libro”, “Autores” : [ “Zulema” ] }

1. **Función pretty(): otra forma de mejor visualización es aplicar la función pretty().**

> db.libros.find().pretty()

{

“\_id”: ObjectId(“60d159461e0ed70729ced13e”),

“titulo”: “Titulo Libro”,

“Autores”: [“Peter”, “Johnathan”]

},

{

“\_id”: ObjectId(“60d159df1e0ed70729ced13f”),

“titulo”: “Otro Libro”,

“Autores”: [“Peter”, “Yolima”]

},

{

“\_id”: ObjectId(“60d159df1e0ed70729ced140”),

“titulo”: “Nuevo Libro”,

“Autores”: [“Zulema”]

}

### Actualizar documentos

Es momento de conocer un ejemplo de cómo se actualiza la información en la base de datos, conociendo el contenido del atributo \_id del documento a actualizar. Preste atención a la siguiente estructura:

db.collection.update(

{ consulta - criterios },

{ documento modificado },

{ upsert: true | false, multi : true | false }

)

La opción upsert permite agregar un documento si no existe, siempre y cuando esta opción esté activada.

> db.libros.update({ “\_id”: ObjectId(“60d159df1e0ed70729ced13f”) }, { “titulo”: “Titulo Modificado” })

WriteResult({ “nMatched”: 1, “nUpserted”: 0, “nModified”: 1 })

>

### Borrar documentos

El método remove() elimina uno o más documentos de una colección. Recibe parámetros para realizar una eliminación selectiva; si no se le pasa ningún parámetro, elimina todos los documentos de la colección (Graterol, 2014). Esto nos lleva a concluir que las bases de datos no relacionales son más sencillas de administrar, al no tener que hacer uso de estructuras complejas e interrelacionadas.

> db.libros.remove({ “\_id”: ObjectId(“60d159df1e0ed70729ced13f”) })

WriteResult({ “nRemoved”: 1 })

La función drop() elimina toda una colección y es la más recomendable a la hora de realizar esta tarea, ya que utiliza menos recursos que remove() (Yohan D, 2014).

< db.libros.drop();

true

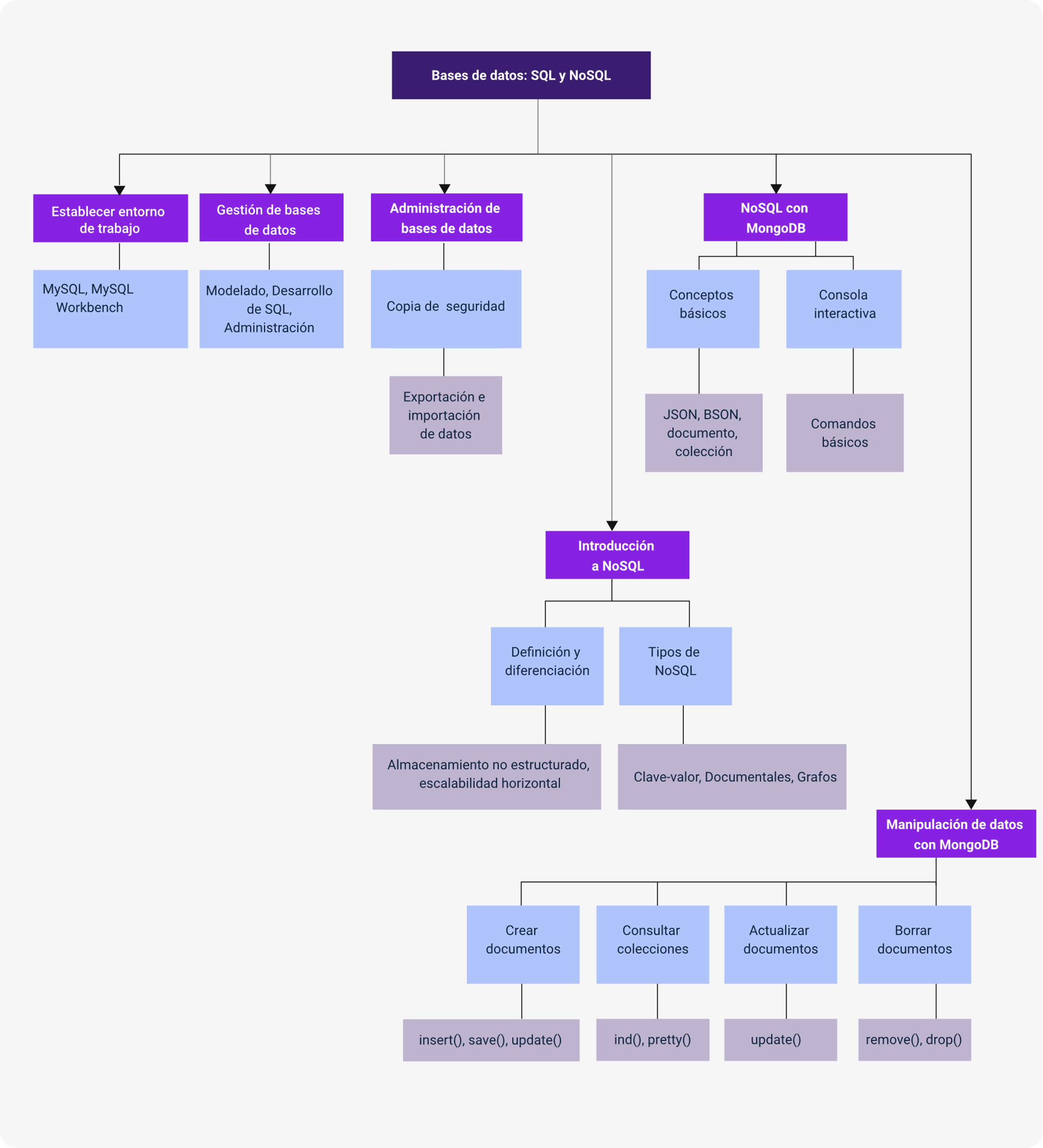
< show collections

coleccion\_personas

<

Síntesis

A continuación, se muestra un mapa conceptual con los elementos más importantes desarrollados en este componente.



Material complementario

| Tema | Referencia | Tipo de material | Enlace del recurso |
| --- | --- | --- | --- |
| Establecer un entorno de trabajo | Bustos, H. (2019). Ejemplo introductorio DDL+DML (MySQL y Workbench) [video]. YouTube. | Video | <https://www.youtube.com/watch?v=xwfzw9paFwo> |
| NoSQL con MongoDB | Fazt. (2019, 22 agosto). MongoDB Curso, Introducción Práctica a NoSQL [Vídeo]. YouTube. | Vídeo | <https://www.youtube.com/watch?v=lWMemPN9t6Q> |
| NoSQL con MongoDB | Ávila, J. (2020, 15 junio). Bases de Datos NoSQL [Vídeo]. YouTube. | Vídeo | <https://www.youtube.com/watch?v=caS51djbuk4> |

Glosario

**Base de datos**: conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

**Colección**: es un compendio de documentos que comparten una relación de estructura similar, no necesariamente igual para todos los objetos.

**DBA**: Data Base Administrator es el rol que diseña un ingeniero responsable de las bases de datos en una organización.

**Diccionario de datos**: conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

**Documento**: en bases de datos no relacionales, un documento es un objeto en formato JSON que es almacenado en binario en el motor NoSQL (BSON).

**Escalabilidad**: es la capacidad que tiene un sistema de ser configurado en software o hardware para mejorar la respuesta a las exigencias o nuevas demandas de capacidad de procesamiento o almacenamiento (conservando las mismas funcionalidades).

**Metadatos**: conjunto unitario de instrucciones que permite a una computadora realizar funciones diversas como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos y el manejo de bancos de datos.

**Modelo conceptual**: enfocado en prestar las entidades, los atributos y las relaciones entre dualidades con su multiplicidad.

**Modelo de datos**: representación de la base de datos a través de un diagrama.

**NoSQL**: base de datos que no usa el estándar SQL.

Referencias bibliográficas

Altarade, M. (2017, 20 abril). Bases de datos NoSQ: Guía definitiva. Pandorafms. <https://pandorafms.com/blog/es/bases-de-datos-nosql>

Graterol, Y. (2014). Mongo DB en español. GITHUB. <https://github.com/yograterol/ebook-mongodb-basico>

Junta de Andalucía. (s. f.). Conceptos sobre la escalabilidad. Marco de Desarrollo de la Junta de Andalucía. <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/220>

Macarrón, P. (2021, 8 marzo). Tipos de bases de datos no relacionales. Consultoría Certia. <https://www.certia.net/tipos-de-bases-de-datos-no-relacionales/>

Microsoft. (2021, 19 mayo). Descripción de normalización de la base de datos. Microsoft Docs. <https://docs.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description>

RAE. (s. f.). base | Diccionario de la lengua española. «Diccionario de la lengua española» - Edición del Tricentenario. <https://dle.rae.es/base#CiiosqO>

Resnick, M. (2008, 12 noviembre). Sembrando las semillas para una sociedad más creativa. Eduteka. <http://eduteka.icesi.edu.co/articulos/ScratchResnickCreatividad>

Sudarshan, H. (2002). Fundamentos de bases de datos. McGraw-Hill.

Vertex42. (s.f.). Modelo de facturas gratis.

Wikipedia. (2021). MariaDB. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=MariaDB&oldid=135391766>

Wikipedia. (2021). MySQL. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/MySQL>

Wikipedia. (2021, 10 junio). Extensible Markup Language. <https://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&oldid=136233446>

Wikipedia. (2021a, enero 17). NoSQL. <https://es.wikipedia.org/w/index.php?title=NoSQL&oldid=132486305>

Créditos

| Nombre | Cargo | Centro de Formación y Regional |
| --- | --- | --- |
| Milady Tatiana Villamil Castellanos | Responsable del Ecosistema | Dirección General |
| Olga Constanza Bermúdez Jaimes | Responsable de Línea de Producción | Centro de Servicios de Salud - Regional Antioquia |
| Henry Eduardo Bastidas Paruma | Experto Temático | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Peter Emerson Pinchao | Experto Temático | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Paola Alexandra Moya Peralta | Evaluadora Instruccional | Centro de Servicios de Salud - Regional Antioquia |
| Carlos Julián Ramírez Benítez | Diseñador de Contenidos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Edgar Mauricio Cortés García | Desarrollador Fullstack | Centro de Servicios de Salud - Regional Antioquia |
| Edgar Mauricio Cortés García | Actividad Didáctica | Centro de Servicios de Salud - Regional Antioquia |
| Luis Gabriel Urueta Álvarez | Validador de Recursos Educativos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Margarita Marcela Medrano Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |
| Daniel Ricardo Mutis Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |