**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | ADSO |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501096. Desarrollar la solución de *software* de acuerdo con el diseño y metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-02. Construir la base de datos para el *software* a partir del modelo de datos. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 013 |
| NOMBRE DEL COMPONENTE FORMATIVO | 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. |
| PALABRAS CLAVE | Bases de datos, SQL, NoSQL, MySQL, MongoDB |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

1. Establecer un entorno de trabajo

1.1. Sistema gestor de bases de datos MySQL

1.2. Sistema de diseño y gestión de bases de datos

1.3. Sistemas basados en MySQL

2. Ejercicios de diseño e implementación

2.1 Análisis y diseño sistema de facturación

2.2 Identificación de sentencias DDL

2.3 Generar la base de datos

2.4 Operaciones de mantenimiento y actualización

2.5 Actualización por interfaz de línea de comandos

3. Administración de bases de datos

3.1 Copia de seguridad de base de datos con MySQL Workbench

3.2 Restaurar una copia de seguridad con MySQL Workbench

4. Introducción a NoSQL

4.1 Definición de NoSQL

4.2 Cómo diferenciar NoSQL de SQL

4.3 Tipos de bases de datos NoSQL

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

5. NoSQL con MongoDB

5.1 Conceptos básicos

5.2 Consola interactiva

5.3 Crear primer documento

6. Manipulación de datos con MongoDB

1. **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.

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.

1. **DESARROLLO DE CONTENIDOS:**
2. **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 *M*ySQL 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.

* 1. **Sistema gestor de bases de datos *MySQL***

*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.

|  |  |
| --- | --- |
| A yellow and white file folder with a green circle and a black background  Description automatically generated | **Proceso de descarga de MySQL Server Community y MySQL Workbench**  Lo invitamos a visitar el siguiente PDF, para conocer el proceso de descarga de MySQL Server *Community* y MySQL *Workbench*. |
| A yellow and white file folder with a green circle and a black background  Description automatically generated | **Proceso de instalación de MySQL**  Lo invitamos a visitar el siguiente PDF, para conocer el proceso de instalación de MySQL. |

**1.2 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:

|  |
| --- |
| Acordeón  CF013\_1.2\_Sistema de diseño y gestión de bases de datos |

**1.3 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.

A person sitting on a bed using a computer

Description automatically generated

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.

|  |  |
| --- | --- |
| A yellow and white file folder with a green circle and a black background  Description automatically generated | **Manual de instalación de MariaDB en Windows**  Lo invitamos a visitar el siguiente PDF, para conocer el Manual de instalación de MariaDB en Windows. |

1. **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. | A person sitting at a computer  Description automatically generated |

**2.1 Análisis y diseño sistema de facturación**

Analicemos el primer caso, llamado Sistema de facturación.

A computer screen with a yellow circle

Description automatically generated

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.

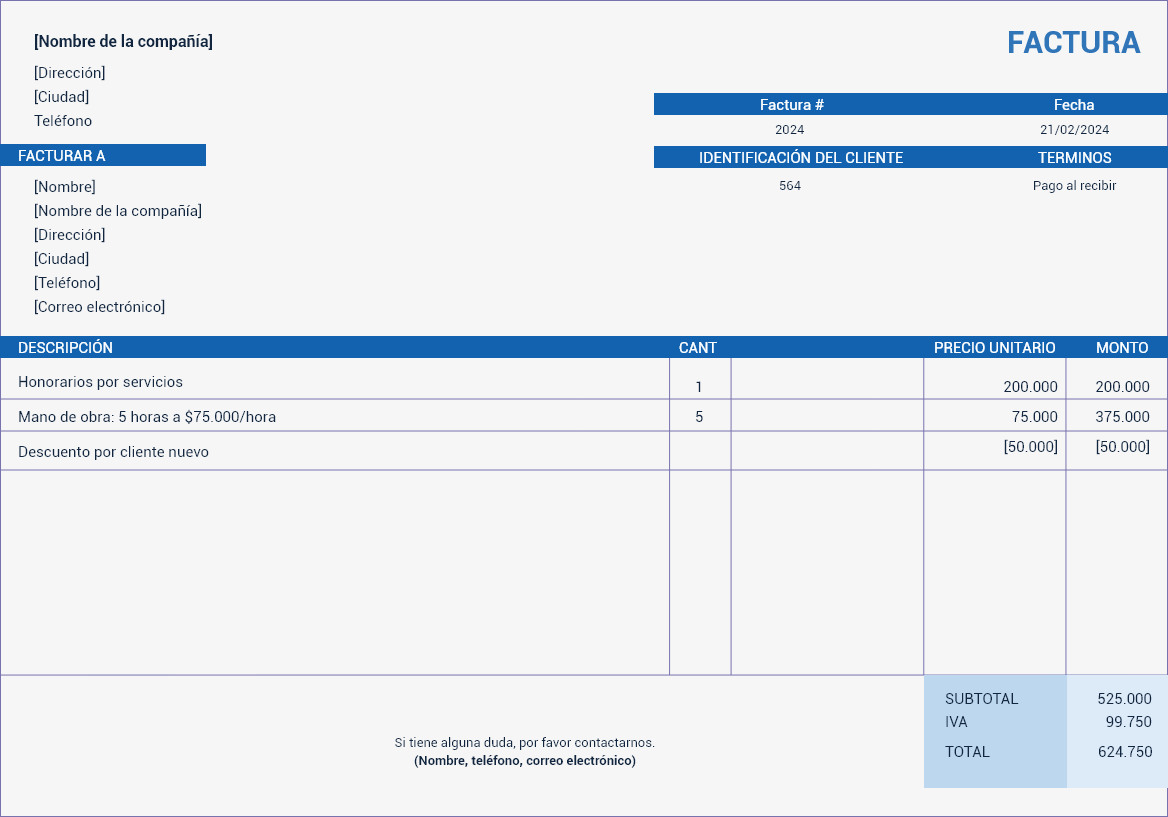
1. A person sitting at a desk with a computer

   Description automatically generatedArtículo 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.

**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. | A person sitting at a computer  Description automatically generated |

**Figura 2.** Factura con información resaltada



A calculator with multiple squares

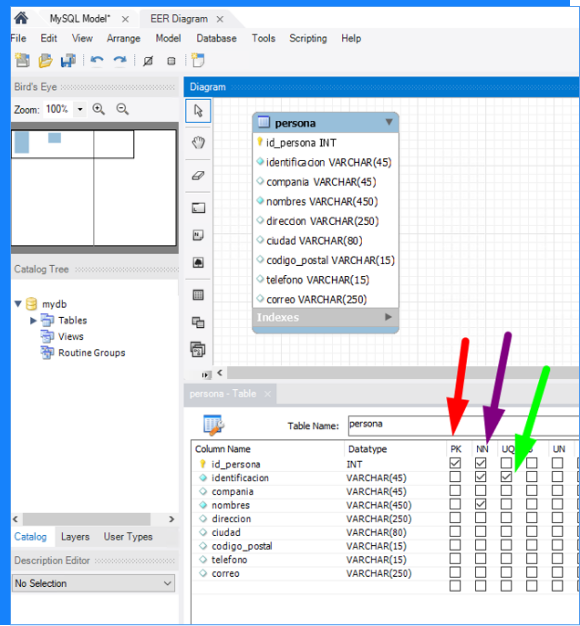
Description automatically generated

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. | A person sitting at a desk writing on a paper  Description automatically generated |

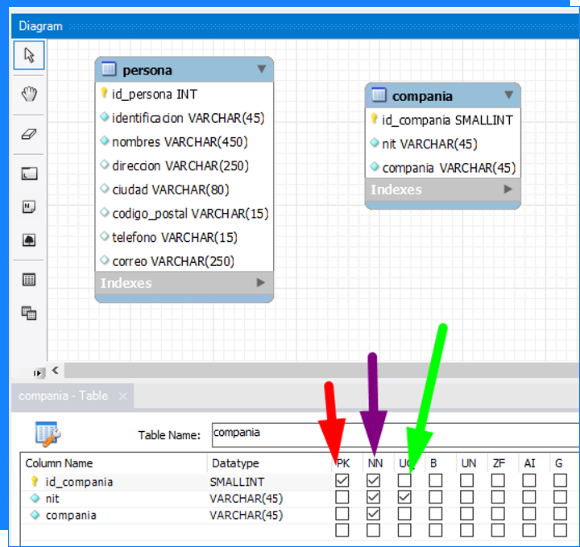
Lo anterior se puede identificar en la siguiente imagen (flecha roja para definir la llave primaria, morada para definir los datos que deben ser registrados obligatoriamente, y la verde una columna cuyo valor no puede repetirse en toda la tabla).

**Figura 3.** 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.

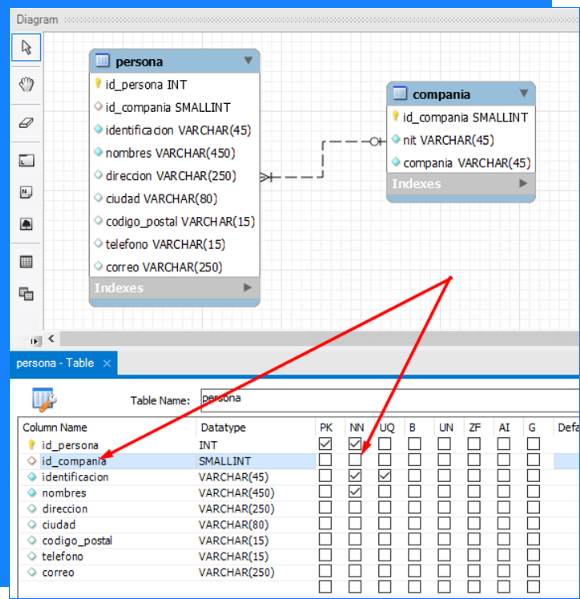
**Figura 4.** Creación de tabla con las compañías



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

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

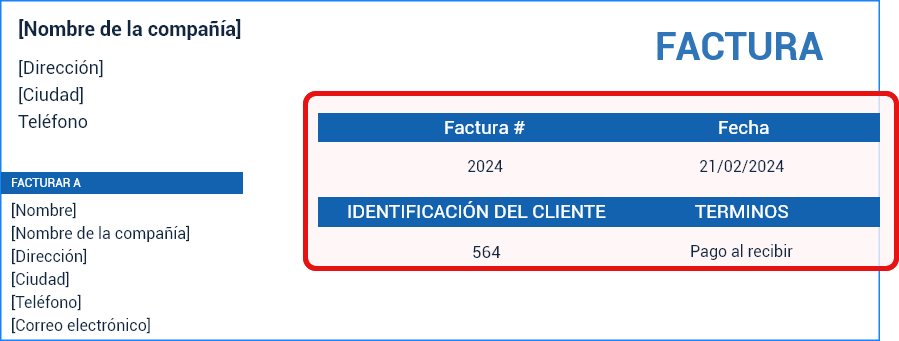
**Figura 5.** Relación entre la persona y la compañía



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.

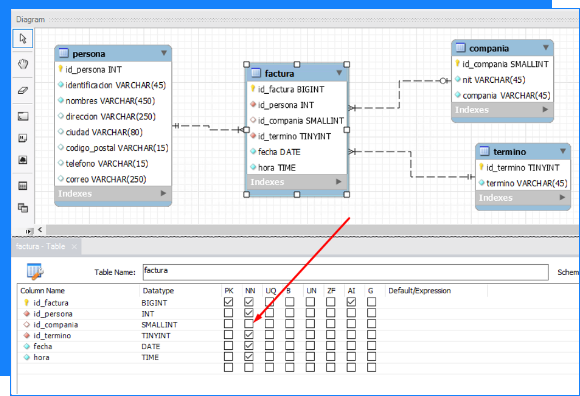
**Figura 6.** Información de la factura

****

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). | A person typing on a computer  Description automatically generated |

**Figura 7.** 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. | A person holding a paper and a computer  Description automatically generated |

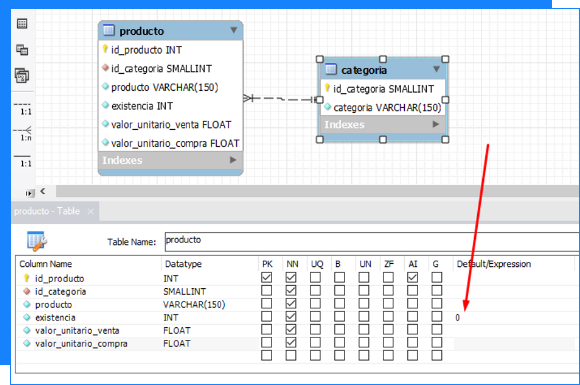
Otro segmento de la factura son los ítems o productos que la componen:

**Tabla 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:

**Figura 8.** 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:

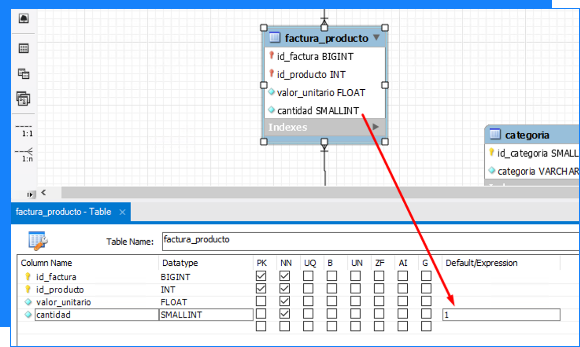
A computer screen with a yellow and blue circle and a yellow circle

Description automatically generated

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.

**Figura 9.** Columnas del modelo



**2.2 Identificación de sentencias DDL**

A person sitting at a computer

Description automatically generatedSuponga 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, s*egún 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).

**Figura 10.** *Copy to Clipboard*

A screenshot of a computer

Description automatically generated

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

|  |
| --- |
| **CREATE 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:**

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.

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.

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:

**Tabla 2.** 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.

A cartoon of a person holding a computer

Description automatically generated**2.3 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:

|  |
| --- |
| Pasos  CF013\_2.3\_Generar la base de datos |

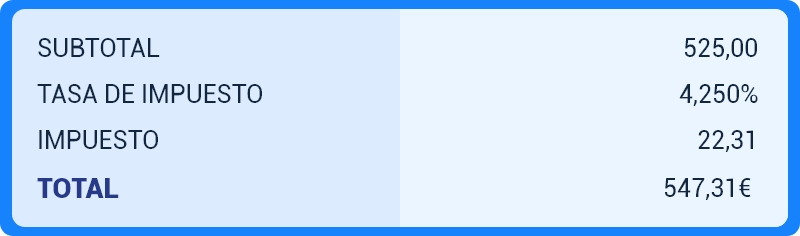
Se guarda un archivo **“Save to file”** y se copia al portapapeles **“Copy to clipboard”**. Se sugiere antes de dar *Next* y continuar el proceso, primero guardar como un archivo y ponerle este nombre, luego de ponerlo le da *Next* para que ejecute ese *script* en el SGDB instalado.

**2.4 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.

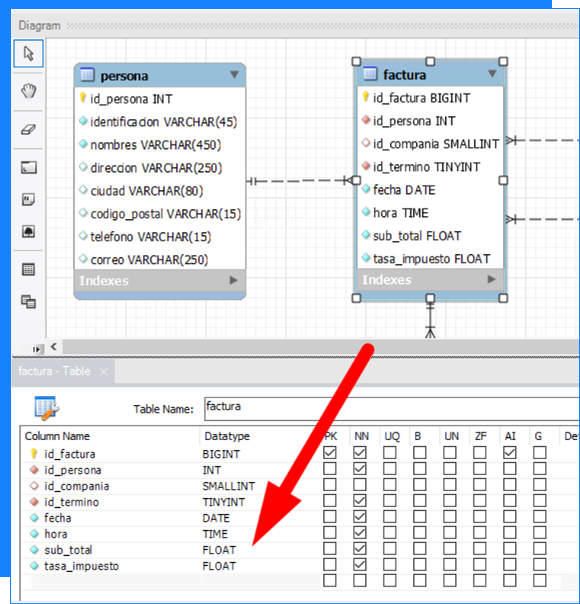
Además, junto a su equipo de trabajo ha detectado que no modelaron el pie de la factura:

**Figura 11.** Pie de la factura



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:

**Figura 12.** 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. A person sitting at a desk with a computer

   Description automatically generatedSeleccionar 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.

A person pointing at a computer screen

Description automatically generated**2.5 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:**

|  |  |  |
| --- | --- | --- |
| A blue and green check mark on a paper  Description automatically generated  Anexo. DDL\_drogueria.sql | A blue and green check mark on a paper  Description automatically generated  Anexo. DDL\_update.sql | A blue and green check mark on a paper  Description automatically generated  Anexo. ModeloFacturaciónDrogueria.mwb |

**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:

|  |
| --- |
| Pasos  CF013\_2.5\_Actualización por interfaz de línea de comandos (1) |

A computer with a magnifying glass

Description automatically generatedA 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:

|  |
| --- |
| Slide  CF013\_2.5\_Actualización por interfaz de línea de comandos (2) |

**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. | A person sitting at a desk with two computer screens  Description automatically generated |

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:**

|  |
| --- |
| Acordeón  CF013\_2.5\_Actualización por interfaz de línea de comandos (3) |

**Figura 13.** Diagrama entidad relación

A diagram of a diagram

Description automatically generated with medium confidence

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.

|  |  |
| --- | --- |
| A yellow and white file folder with a green circle and a black background  Description automatically generated | **Modelos.mwb**  Lo invitamos a visitar el siguiente, archivo Modelos.mwb |

**Ahora, analicemos los siguientes pasos:**

|  |
| --- |
| Slides  CF013\_2.5\_Actualización por interfaz de línea de comandos (4) |

**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.):

|  |
| --- |
| Acordeón  CF013\_2.5\_Actualización por interfaz de línea de comandos (5) |

1. **Administración de bases de datos**

A yellow circle with a black background and a blue server

Description automatically generatedLas 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:

|  |
| --- |
| Pestañas  CF013\_3\_ Administración de bases de datos |

|  |  |
| --- | --- |
| A person sitting in a chair with a computer  Description automatically generated | 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. |

**3.1 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:

* **A person standing at a desk with a computer

  Description automatically generated**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.

**3.2 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.

Hands holding a folder icon

Description automatically generatedSiga 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.

1. **Introducción a NoSQL**

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 sistema*s NoSQL se deno*minan, a veces, “no solo *SQL*” para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

* 1. **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.

|  |
| --- |
| Slide  CF013\_4.1\_Definición de NoSQL |

* 1. **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.

**Tabla 3.** Comparación de las funciones de NoSQL

| Característica | Base de datos NoSQL | Bases de datos relacionales |
| --- | --- | --- |
| 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:

**Figura 14**. Ventajas y desventajas bases de datos

* 1. **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](mailto:juancho@nerss.et)  Teléfono: 3150817273  Teléfono: 3168272372 | Cédula: 98765322  Nombre: DANIELA  Apellidos: VELÉZ LEMOS  Correo: [dnala@nerss.et](mailto: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:

**Tabla 4.** 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](mailto: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](mailto: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.

**Tabla 5.**  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). |
| Objetos | 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:

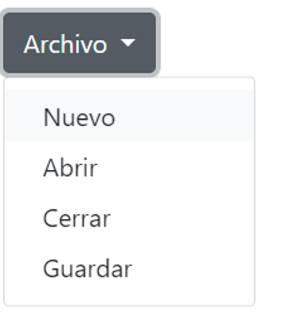
|  |
| --- |
| {  "nombres": "JUAN CARLOS",  "apellidos": "MARTINEZ",  "fecha\_nacimiento": "1995-05-15",  "edad": 65,  "saldo": 9876.134,  "hobbies": ["música", "futbol", "leer"],  "correo": "juancho@nerss.et",  "pedidos": [],  "acreedores": null  } |

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:

**Figura 15**. Ejemplo



**Representación en formato BSON**

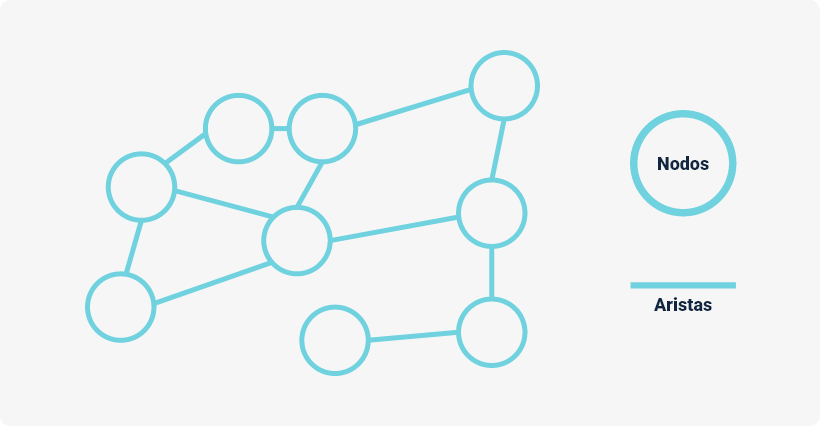
La representación de JSON que se presenta en esta tabla es una representación en formato texto, pero también se puede representar en formato binario (unos y ceros). A esta otra representación equivalente en binario se le llama BSON. Es una cadena de unos y ceros sin nada que la caracterice de más, excepto que al convertirla en cadenas de texto se presenta como se observó en el ejemplo anterior.

|  |  |
| --- | --- |
| **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()"  }  ]  }  }  } |

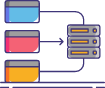
**Bases de datos orientada a grafos**

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.

**Figura 16.** Grafos

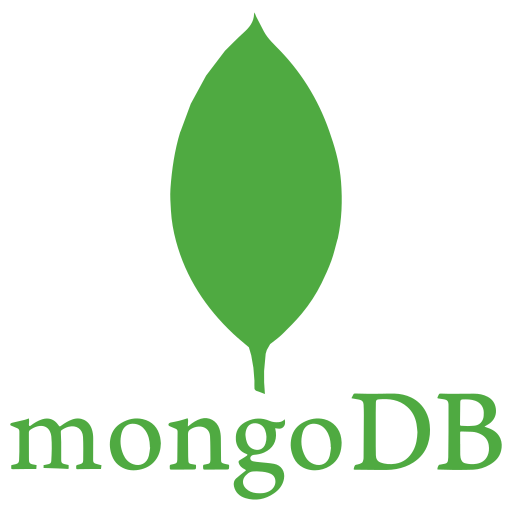


**4.4. Sistemas de Gestión de Bases de Datos (SGBD) NoSQL**

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*.

**Tabla 6.** Sistemas de gestión de bases de datos

| Tipo | Logotipo | Nombre | Lanzamiento | Licencia |
| --- | --- | --- | --- | --- |
| Documental | Miles de bases de datos MongoDB comprometidos y llevan a cabo para el  rescate – Naked Security | MongoDB | 2008 | GNU AGPL v3.0 |
| Documental |  | CouchDB | 2005 | Apache License 2.0 |
| Documental | RavenDB logo small | Raven DB | 2010 | GNU AGPL v3.0 |
| Clave/Valor | Apache Cassandra - Wikipedia, la enciclopedia libre | Apache Cassandra | 2008 | Apache License 2.0 |
| Clave/Valor | riak – Riak | Riak | 2009 | Apache License 2.0 |
| Clave/Valor | REDIS: qué, cómo, ejemplos - yosoy.dev | Redis | 2009 | Licencia: BSD |
| Grafos | Primeros pasos con Neo4j | Neo4j | 2009 | GNU AGPL v3.0 |
| Grafos |  | Dex | 2008 | Comercial |
| Grafos | Sones GraphDB - Wikipedia | Sones GraphDB | 2012 | GNU AGPL v3.0 y comercial |



**5. 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.

**5.1. 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.

|  |
| --- |
| Tarjetas  CF013\_5.1\_Conceptos básicos (1) |

**5.2. 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.

|  |
| --- |
| Acordeón  CF013\_5.2\_Consola interactiva |

**5.3. 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  } |

**6. 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**  **>** |

1. **SÍNTESIS**

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo.

A diagram of a company

Description automatically generated

1. **ACTIVIDADES DIDÁCTICAS (Se debe incorporar mínimo 1, máximo 2)**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Creación y administración de bases de datos SQL y NoSQL |
| Objetivo de la actividad | Aprender a configurar, diseñar y administrar bases de datos utilizando MySQL y MongoDB, aplicando conceptos de diseño conceptual, lógico y físico. |
| Tipo de actividad sugerida | *Relacionar conceptos* |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | *CF013\_Actividad didactica* |

1. **MATERIAL COMPLEMENTARIO:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| Establecer un entorno de trabajo | Bustos, H. (2019). Ejemplo introductorio DDL+DML (MySQL y Workbench) [video]. YouTube. https://www.youtube.com/watch?v=xwfzw9paFwo | 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> |

1. **GLOSARIO:**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| 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. |

1. **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>

<https://www.vertex42.com/es/excel-factura.html>

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>

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia  *(Para el SENA indicar Regional y Centro de Formación)* | Fecha |
| Autor (es) | Henry Eduardo Bastidas Paruma | Experto Temático | Regional Cauca - Centro de Teleinformática y Producción Industrial | Noviembre 2022 |
| Peter Emerson Pinchao Solís | Experto Temático | Regional Tolima - Centro agropecuario La Granja | Noviembre 2022 |
| Paola Alexandra Moya | Evaluadora instruccional | Regional Antioquia - Centro de Servicios de Salud | Junio 2024 |
|  | Olga Constanza Bermúdez Jaimes | Responsable Línea de Producción Antioquia | Regional Antioquia - Centro de Servicios de Salud | Junio 2024 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| Autor (es) |  |  |  |  |  |