**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Aseguramiento de la calidad de *software* |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA |  | RESULTADOS DE APRENDIZAJE |  |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 08 |
| NOMBRE DEL COMPONENTE FORMATIVO | Modelo entidad relación |
| BREVE DESCRIPCIÓN | Una vez conocidos los principios del diseño de bases de datos relacionales ,se hace importante fortalecer las habilidades de análisis, debido a que diseñar un modelo de datos no solamente es una tarea que intenta cumplir un conjunto de requisitos funcionales de un sistema informático, sino también el requerimiento de rendimiento, escalabilidad y flexibilidad. |
| PALABRAS CLAVE | Modelo conceptual, modelo relacional, modelo físico, base de datos, diseño. |

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

1. **TABLA DE CONTENIDOS:**

**Introducción**

1. **Establecer un entorno de trabajo**
   1. Sistema gestor de bases de datos MySQL
   2. Sistema de diseño y gestión de bases de datos
   3. Sistemas basados en MySQL
2. **Ejercicios de diseño e implementación**
   1. Análisis y diseño sistema de facturación
   2. Identificación de sentencias DDL
   3. Generar la base de datos
   4. Operaciones de mantenimiento y actualización
   5. Actualización por interfaz de línea de comandos
   6. Análisis y diseño gestión de matrículas
      1. Diseño conceptual
      2. Diseño lógico
      3. Diseño físico
3. **Administración de base de datos**
   1. Copia de seguridad de base de datos con MySQL *Workbench*
   2. Restaurar una copia de seguridad con MySQL *Workbench*
4. **INTRODUCCIÓN**



Bienvenido al presente componente formativo.

Una vez definidos los conceptos básicos de las bases de datos, se está listo para afrontar su análisis y diseño, el cual consiste en tomar los datos más importantes que están implicados en un problema o qué datos son necesarios para llevar el control por un sistema informático a implementar.

Para extraer los datos, se debe hacer un análisis detallado del problema, para poder descartar los que no son susceptibles a la administración por el sistema a desarrollar, luego de tener extraídos los datos esenciales, se puede empezar a construir el modelo adecuado, por lo que se necesitará una herramienta de diseño de base datos y a través de ella expresar todos los aspectos técnicos de la solución a implementar.

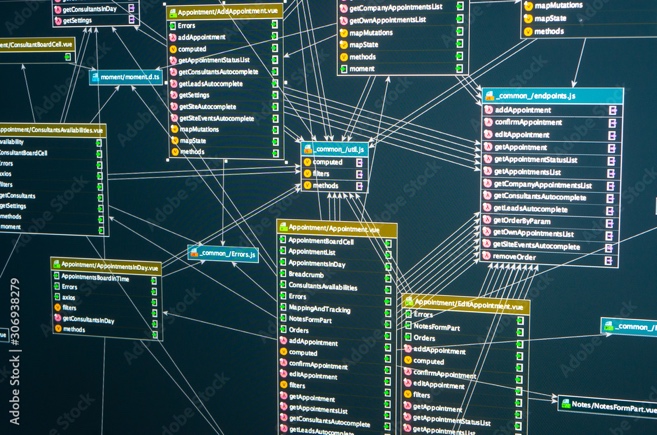
Este componente brinda una serie de ejercicios prácticos de análisis y diseño, a la vez que profundiza en los temas de modelado.

Le deseamos muchos éxitos en este proceso de aprendizaje.

1. **DESARROLLO DE CONTENIDOS:**
2. **Establecer un entorno de trabajo**

Es importante saber 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, utilizaremos MySQL como sistema gestor de bases de datos y MySQL *Workbench* como herramienta de diseño, es importante que en la medida que se van presentando los contenidos, realice de forma local y en su computador 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, lo que significa que es una licencia pública general y/o licencia comercial por Oracle Corporation y, que según medios de comunicación, es considerada como la base de datos de código abierto más popular del mundo. Tener licencia dual significa que se puede usar la licencia de uso público sin incurrir en gastos; sin embargo, se puede adquirir una licencia de pago que incluye paquetes que robustecen el producto y servicios de soporte técnico especializado.

Lo invitamos a visitar los siguientes PDF, para conocer el proceso de descarga e instalación de MySQL *Server Community.*

PDF Proceso de descarga de MySQL *Server Community* y MySQL *Workbench*

PDF Proceso de instalación de MySQL

* 1. **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:

|  |
| --- |
| CF8\_1\_2\_Caracteristicas |

* 1. **Sistemas basados en MySQL**

Existen otros sistemas que son completamente compatibles 100 % con MySQL, es decir, que toda base de datos que es gestionada por MySQL puede ser gestionada en otro motor de sistema de gestión de bases de datos, un ejemplo de ello es MariaDB.

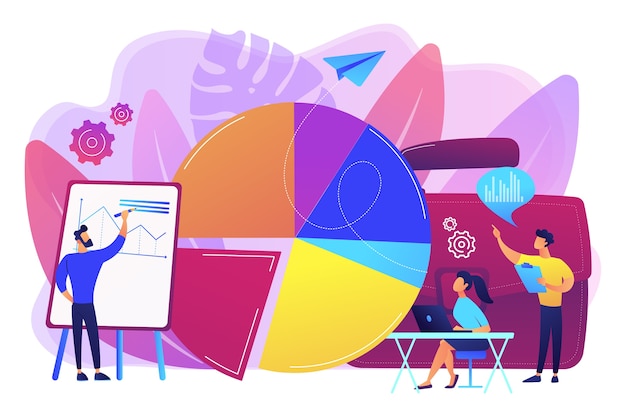


MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (*General Public License*). Es desarrollado por Michael (Monty) Widenius —fundador de MySQL. Este sistema es el más usado en entornos Linux en lugar del MySQL.

Existen alternativas para el sistema de gestión de base de datos (SGBD) y herramientas para la gestión, administración y diseño.

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.

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.

* 1. **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.

En una entrevista, el cliente dijo que necesitaba algunos reportes como:

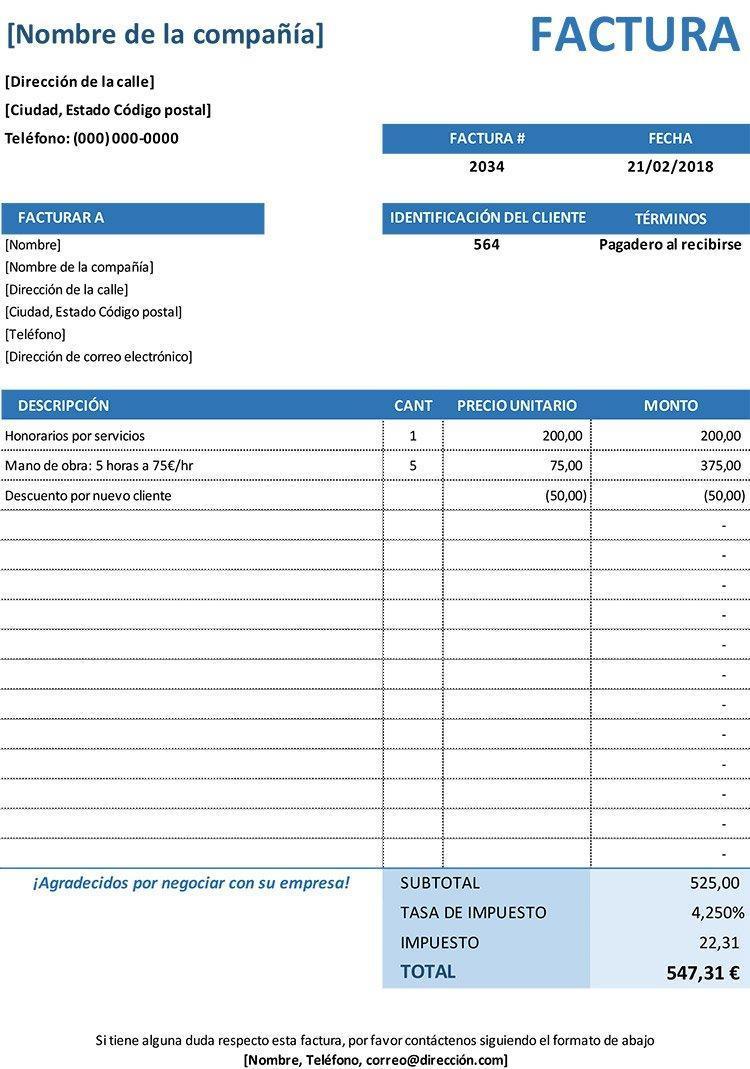
* Artículo más vendido.
* Artículos que más utilidades dejan por unidad.
* Empresa o persona que más factura (cantidad de factura).
* Empresa o persona que más factura (volumen de ventas en pesos).
* Productos sin existencia.

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 muestra en la siguiente figura.

**Figura 1**

*Modelo de factura*



**Análisis y diseño**

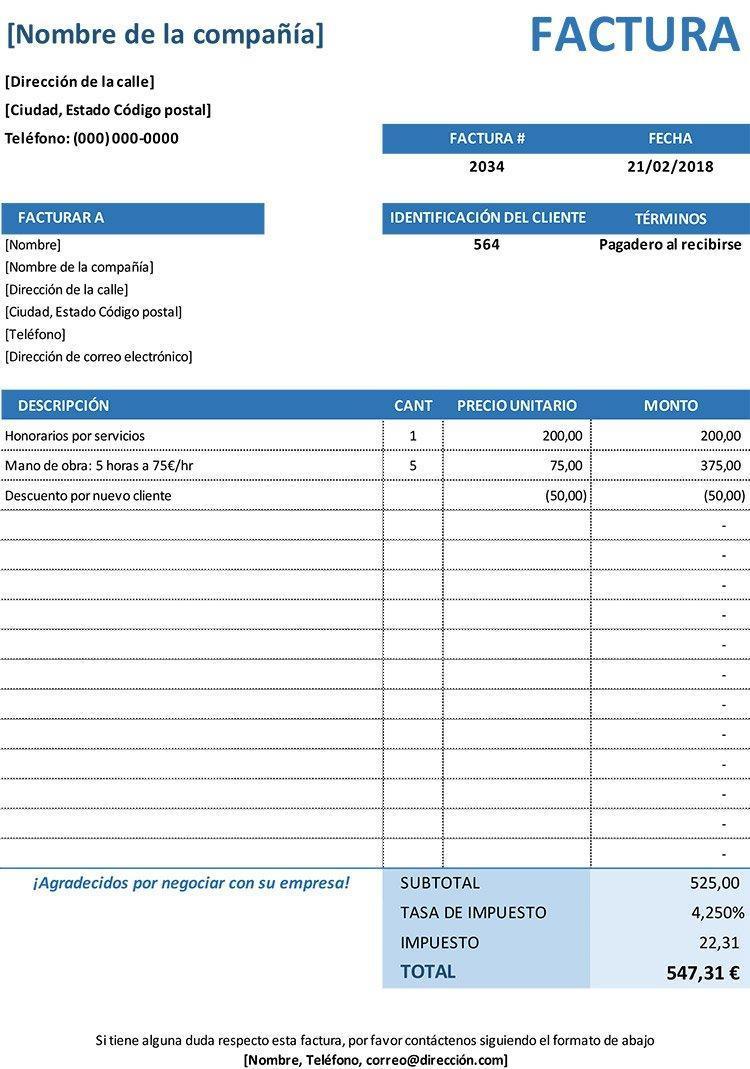
Una buena forma de empezar el análisis es con el formato de la factura; se inicia revisando la cabecera de la factura.

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”, “direccion”, “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.

**Figura 2**

*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.

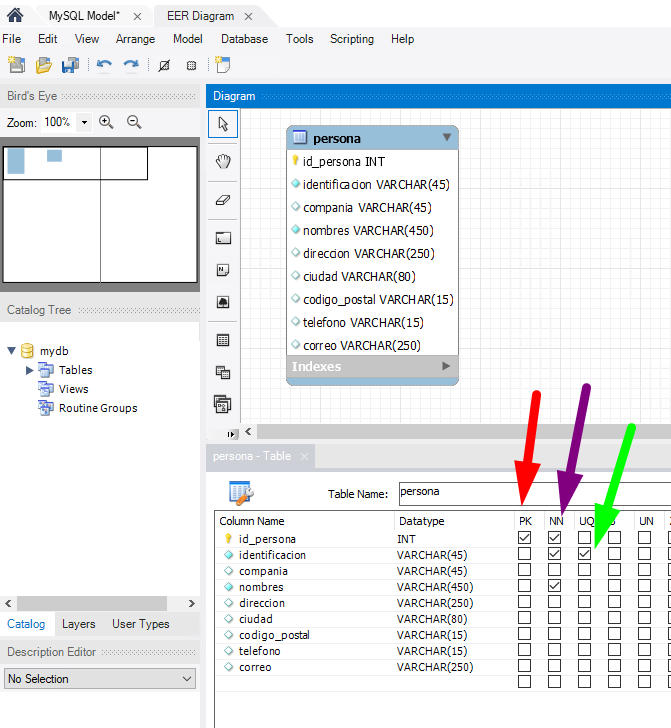
Lo anterior se puede modelar con una entidad, la entidad persona que relaciona los datos (nombre, compañía, dirección, ciudad, teléfono, correo electrónico).

Una forma simplificada sería creando una entidad persona y para poder identificar cada fila usar una llave primara (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 ver 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**

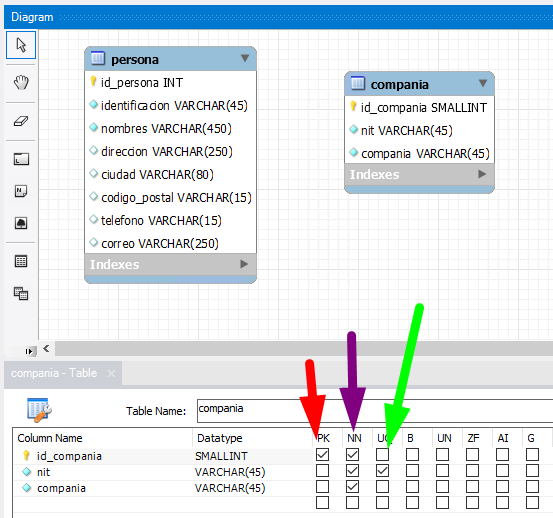
*Identificación*



Pero como el cliente necesita un reporte de las empresas que más le compran, es necesario que la compañía sea modelada por aparte, de esta forma se eliminaría la columna compañía de la tabla persona y se crea una tabla con las compañías. En la llave primaria (rojo) no se puede definir una compañía sin tener su NIT (Número de Identificación Tributaria) flecha morada y, finalmente marcar la unicidad para el NIT, debido a que puede haber empresas con el mismo nombre, pero no con la misma identificación tributaria NIT.

**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:

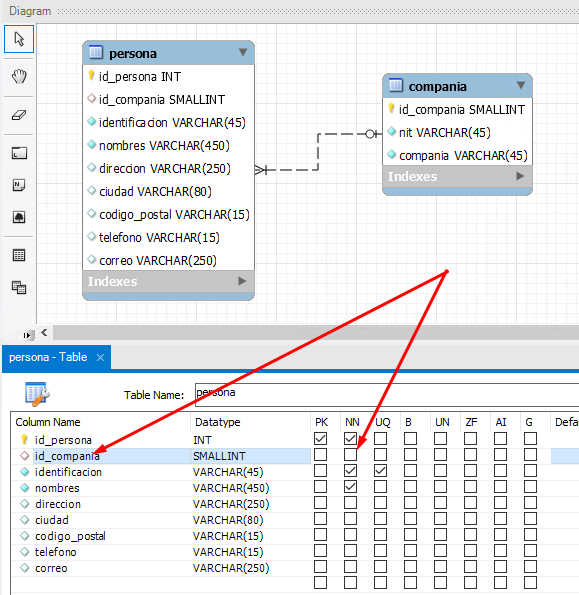
1. Ahorro de espacio de almacenamiento.
2. Ahorro en el tiempo computacional de las consultas.

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:

**Figura 5**

*Relación entre la persona y la compañía*

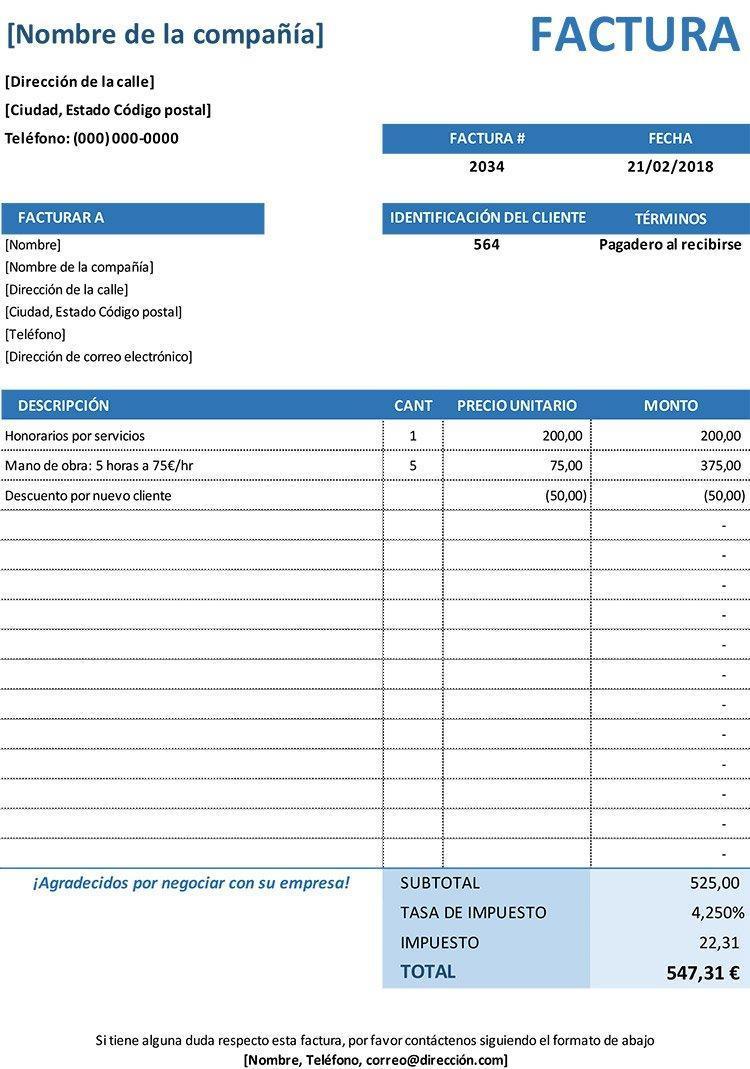


Al hacerlo de esta forma, una persona puede pertenecer a una empresa (solo a una), pero esa misma persona podría comprar en la farmacia unas veces a nombre propio y otras a nombre de la empresa, y este modelo no permitiría saber cada caso. Otro problema es que esa persona un día compra a nombre de una 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*

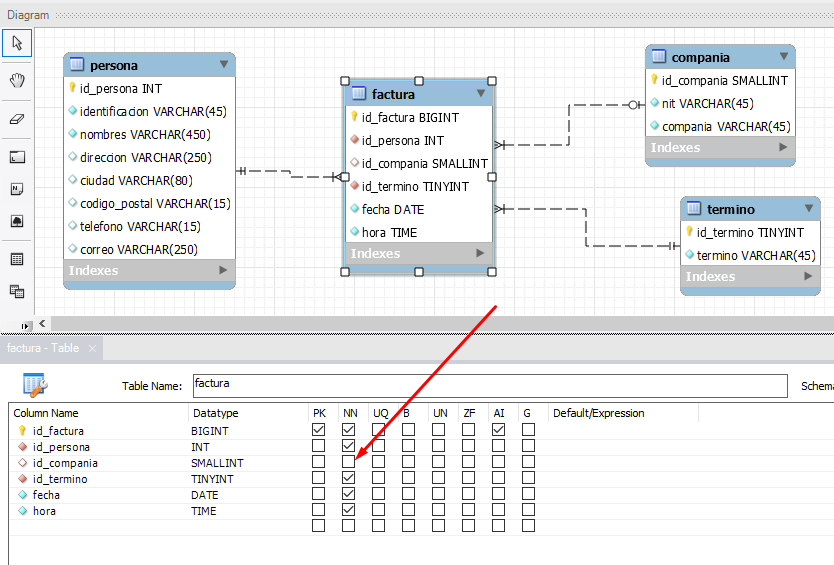


Ahora, se presentan los datos de la factura y sus datos, como número de factura (que debe ser único y secuencial), fecha, la identificación del cliente (código interno id\_persona), y términos de pago.

Como se puede ver 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).

**Figura 7**

*Creación de la tabla término de pagos*



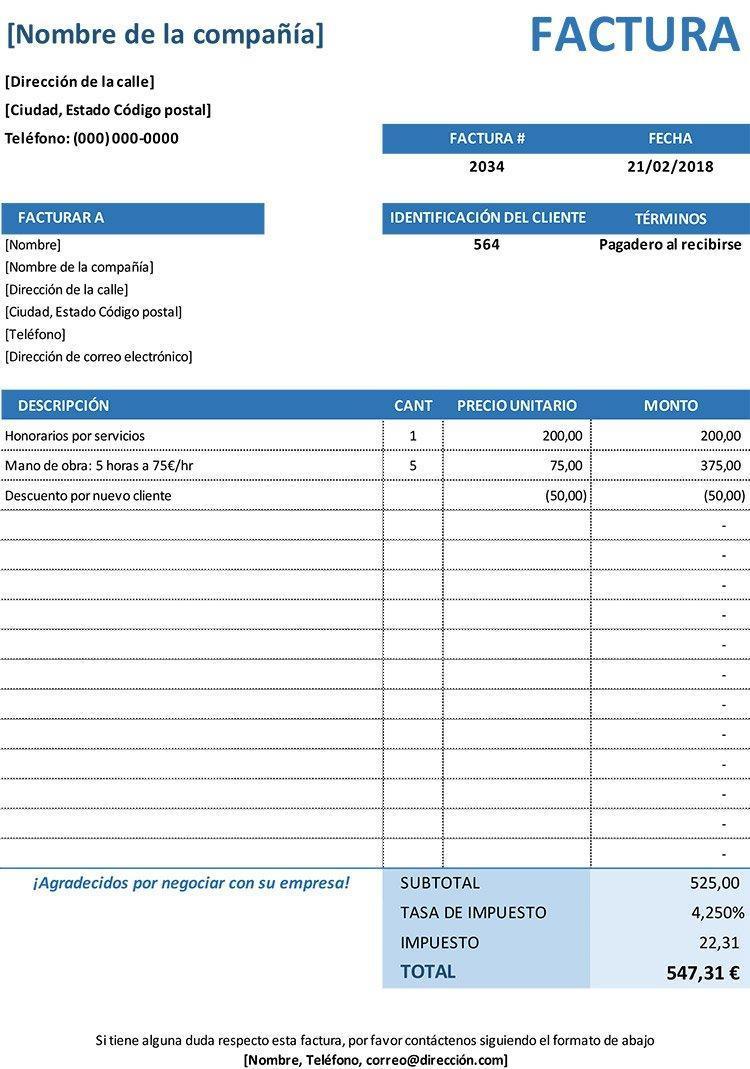
Note que una factura necesariamente debe tener relacionada la referencia de una persona, el término, la fecha y la hora, pero la compañía (id\_compania) no es obligatoria. Esto es porque hay facturas que las compran las personas naturales y otras que las compran las empresas o compañías, pero en un caso 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 (**A**uto **I**ncrementable), 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:

**Figura 8**

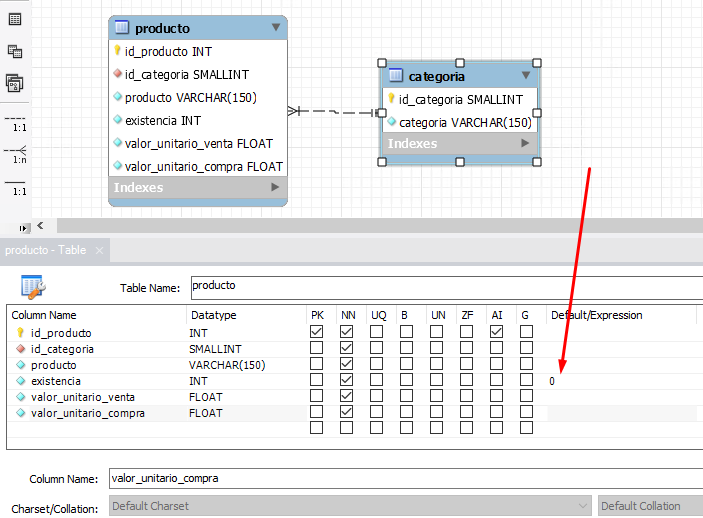
*Ítems que componen la factura*



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

**Figura 9**

*Apartados que tienen los productos*



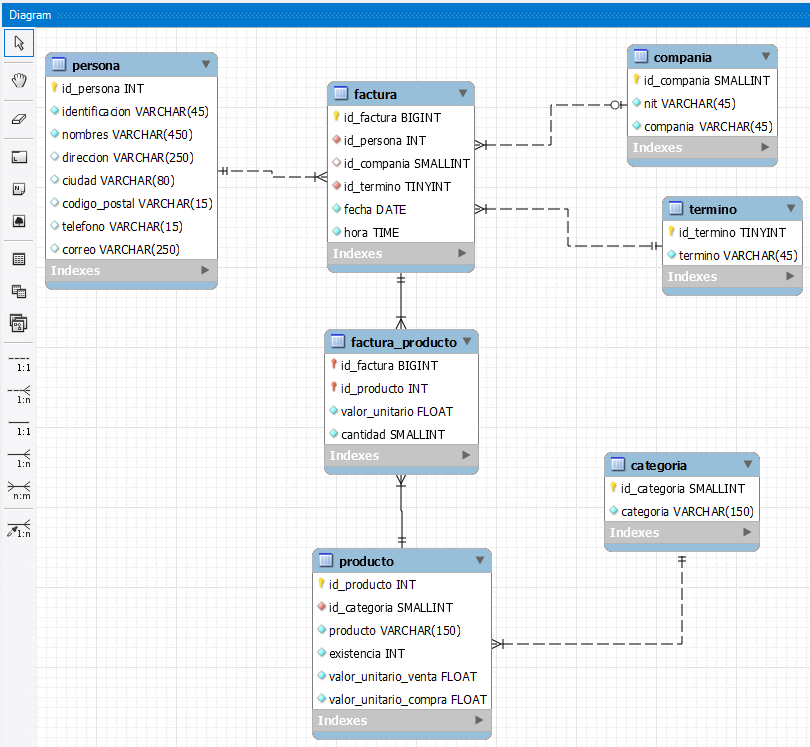
Una categoría puede tener varios productos o varios productos pertenecen a una categoría, esto es una relación de 1 a muchos, que queda definida en la imagen anterior. Note los tipos de datos definidos y también que la columna existencia no puede ser *null*, además de eso, si el usuario no pone un dato 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.

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.

Cada una de estas filas representa un producto que está relacionado con una factura, la cantidad vendida y el valor unitario al que fue vendido (porque el cliente puede tener descuento, según su entrevista).

**Figura 10**

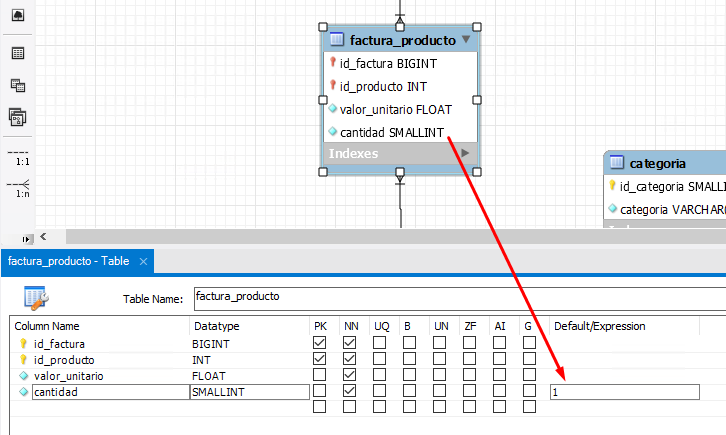
*Filas*



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 muestra 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 11**

*Columnas del modelo*



* 1. **Identificación de sentencias DDL**

Como se mencionó antes, el SGDB (MySQL), reconoce sentencias SQL (*Standar Query Languaje*, Lenguaje Estándar de Consultas), y un subconjunto de esta lengua son las sentencias DDL (*Data Definition Language*, Lenguaje de Definición de Datos).

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:

|  |
| --- |
| CF8\_2\_1\_Pasos |

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.

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

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 MySQL *WorKbench* crea unas restricciones que permiten mantener la integridad de los datos.

**CONSTRAINT** `fk\_factura\_persona`

**FOREIGN KEY** (`id\_persona`)

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

**ON DELETE NO ACTION**

**ON UPDATE NO ACTION,**

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

**Tabla 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.

* 1. **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 uno 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:

|  |
| --- |
| CF8\_2\_3\_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 *scrip*t 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.

* 1. **Operaciones de mantenimiento y actualización**

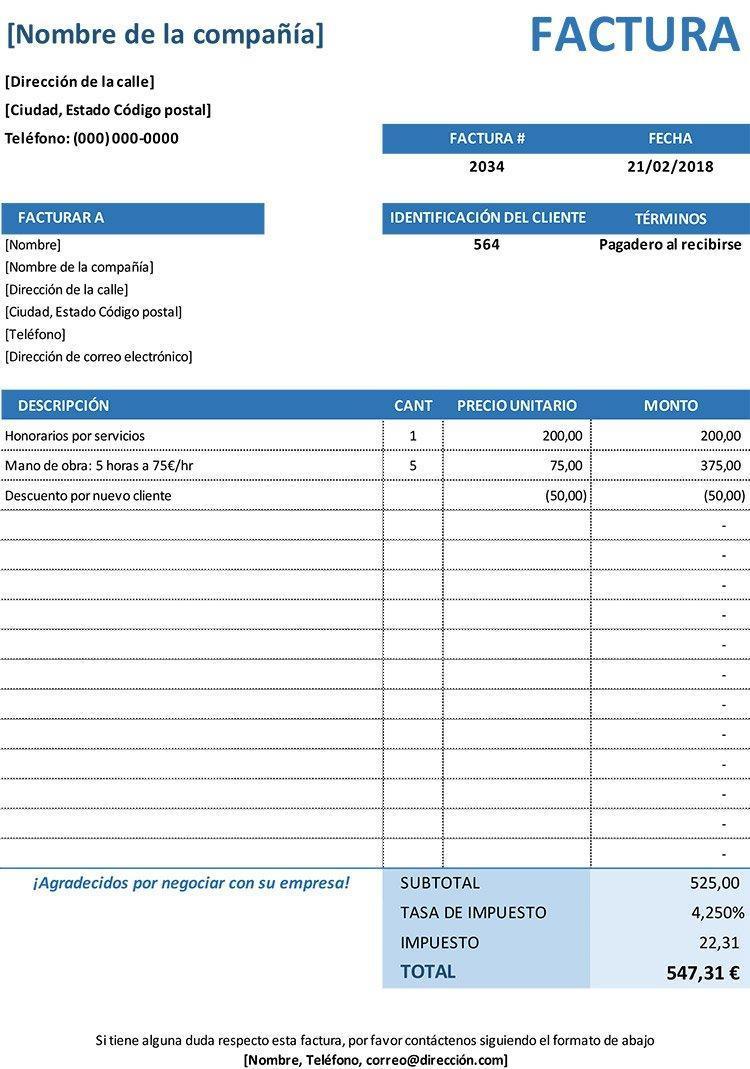
Por lo general, la base datos se diseña y se despliega en servidores en la nube, *hosting* o en instancias de almacenamiento, sistemas operativos o de plataforma que no tienen interfaz gráfica, es por eso que es importante el archivo de texto DDL con extensión .sql, la base de datos hasta ahora generada 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 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:

**Figura 13**

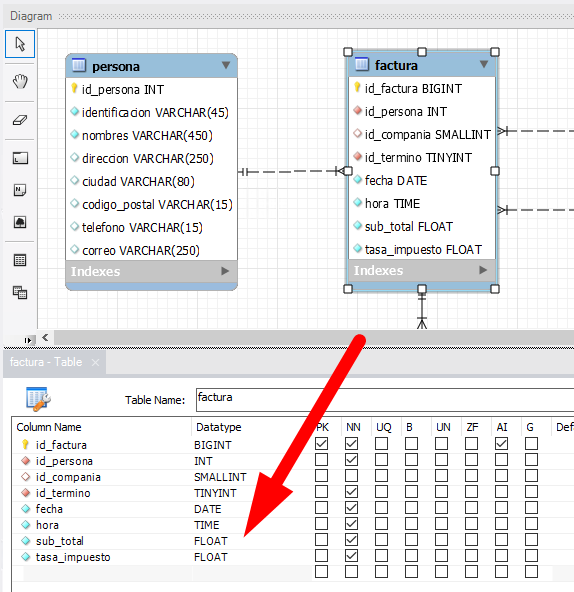
*Pie de la factura*



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 14**

*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. Esto se logra dando clic en Database > Synchronize Model.
2. Luego, seleccione la conexión local.
3. Luego, seleccione los datos en la parte de generación de *script* adicionales.
4. Después de pedir la contraseña se conecta; analice las bases de datos presentes y características del servidor.
5. Seleccione la base de datos con la que se va a sincronizar.
6. Analice las diferencias.
7. Al ubicarse en cada una de las entidades o tablas, se puede ignorar la actualización.
8. En lugar de actualizar la base de datos, si lo que se quiere es actualizar el modelo (diagrama) para que quede conforme a la base de datos, se cliquea en el botón “Update Model”. Pero este no es el caso de interés porque se necesita actualizar la base de datos, entonces se da 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.

* 1. **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:

DDL\_drogueria.sql

DDL\_update.sql

ModeloFacturaciónDrogueria.mwb

A continuación, se muestra 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:

|  |
| --- |
| CF8\_2\_5\_Script |

* 1. **Análisis y diseño gestión de matrículas**

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:

|  |
| --- |
| CF8\_2\_6\_Fases |

Estudiemos ahora el caso 2, gestión de matrículas.



Se necesita un sistema que permita controlar la 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.

* + 1. **Diseño conceptual**

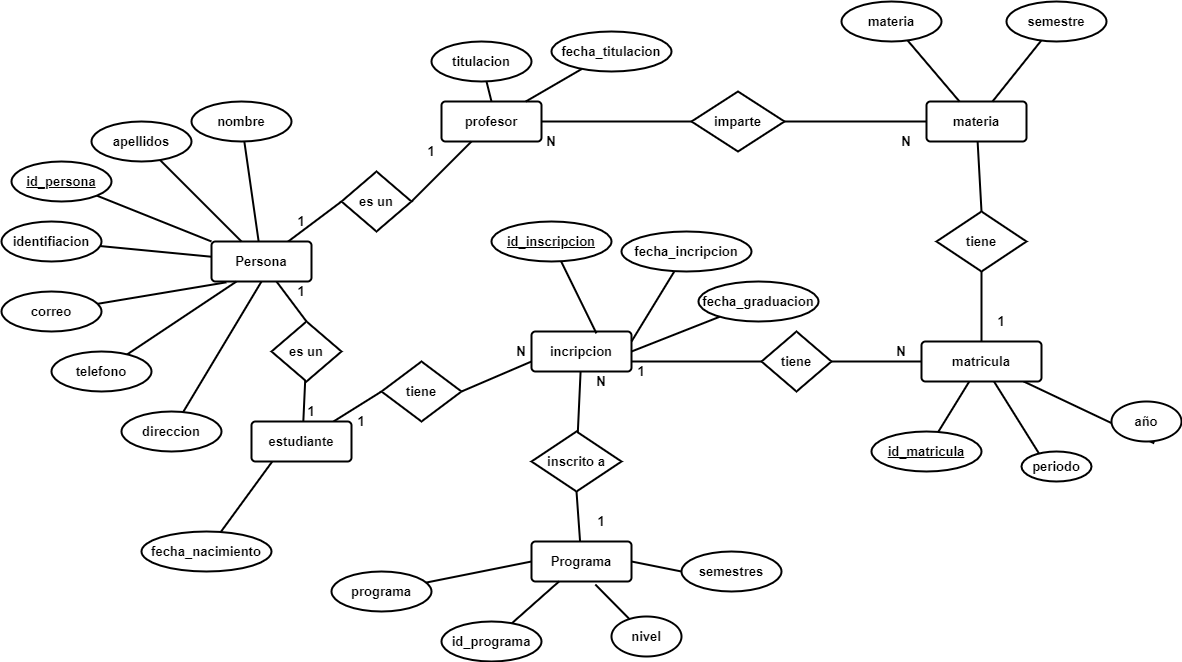
Para la elaboración del modelo conceptual es fundamental identificar las entidades, luego los atributos de cada una de ellas, luego las relaciones entre entidades y, si se desea, se elabora un diagrama de entidad relación como resultado del diseño conceptual:

|  |
| --- |
| CF8\_2\_6\_1\_Diseno\_conceptual |

Cada vez es menos común hacer un diagrama de entidad relación, pero si sirve a su propósito de entender mejor el problema puede elaborarlo.

**Figura 15**

*Diagrama entidad relación*



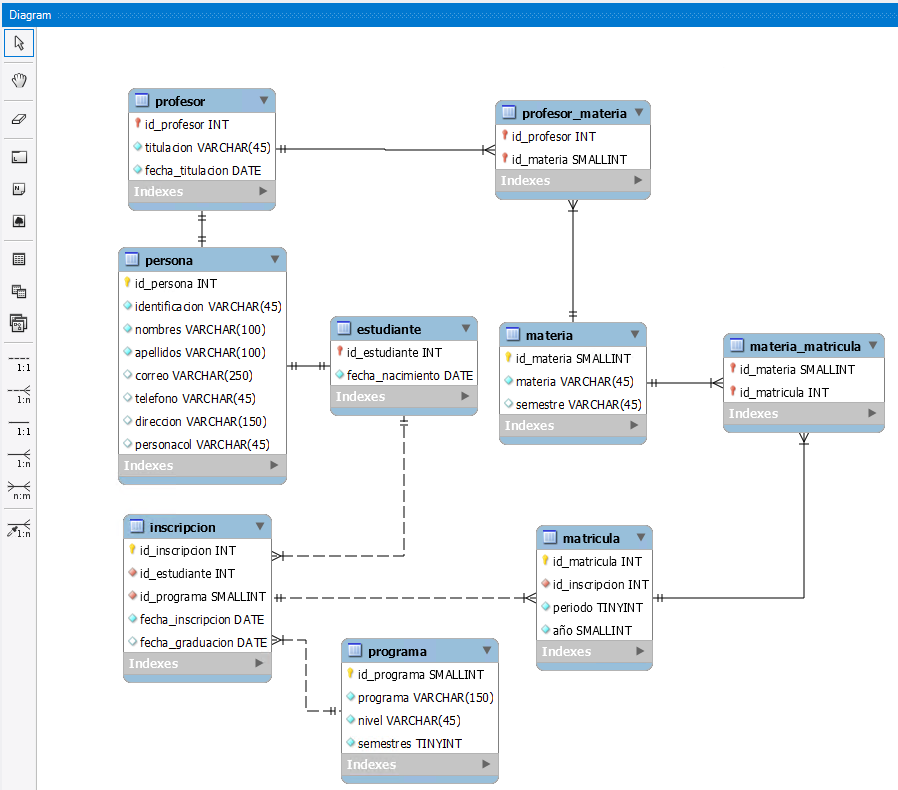
Actualmente, algunos diseñadores de bases de datos no diseñan el modelo entidad relación, por lo general directamente diseñan el modelo lógico o diagrama relacional, en algunos casos se emplea el modelo que se denomina modelo conceptual, que es equivalente al modelo entidad relación, pero difiere en 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.

* + 1. **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).

**Figura 16**

*Mapear diagrama entidad relación a un modelo relacional*



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

Descarque el archivo Modelos.mwb

Ahora, analicemos los siguientes pasos:

|  |
| --- |
| CF8\_2\_6\_2\_Diseno\_logico |

* + 1. **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.):

|  |
| --- |
| CF8\_2\_6\_3\_Diseno\_fisico |

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.

1. **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:

|  |
| --- |
| CF8\_3\_Responsabilidades |

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.

A continuación, se verá una aproximación sobre cómo hacer y restaurar una copia de la base de datos.

* 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:

1. 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 “+”.
2. Posteriormente, en *test connection* y luego en él se escribe la contraseña del *root.*
3. Se le da doble clic a la conexión creada.
4. Luego, se da clic en la pestaña administración y *Data Export.*
5. 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.*

1. El proceso termina diciendo que la copia ha terminado.

A continuación, se verá cómo se restablece la copia que se ha generado, usando la herramienta MySQL *Workbench.*

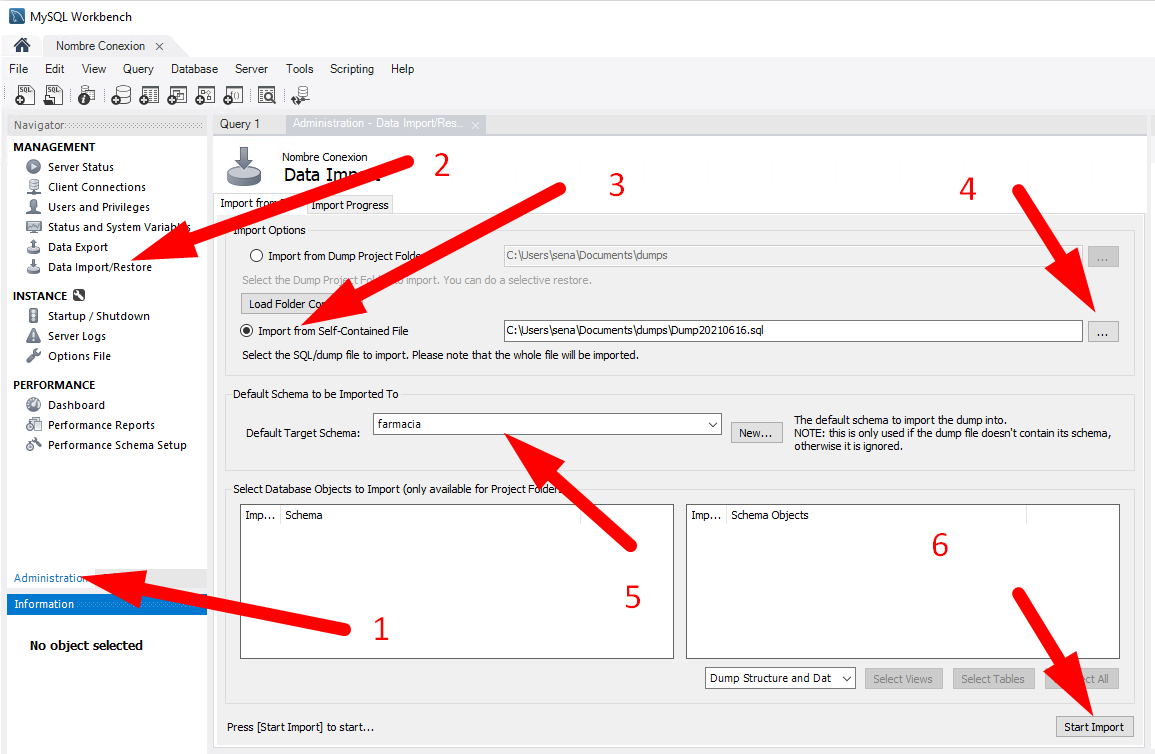
* 1. **Restaurar una copia de seguridad con MySQL *Workbench***

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 muestra en la imagen:

**Figura 17**

*Pasos*

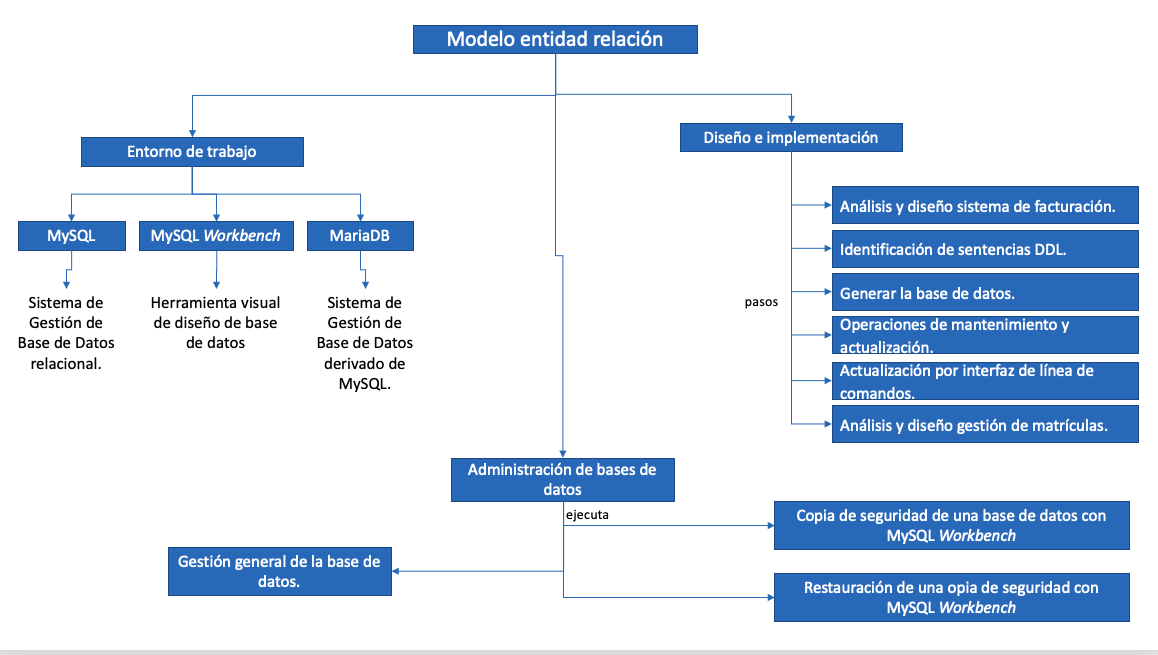


1. Ingrese a la pestaña de administración.
2. Seleccione *Data Import/Restore.*
3. Marque la opción *Import from Self-Contained File*.
4. Busque el archivo generado antes.
5. Seleccione la base de datos a restaurar.
6. Haga clic en el botón *Star Import.*

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. **SÍNTESIS**

Lo invitamos a revisar el siguiente mapa conceptual, en el cual podrá hacer un breve recorrido sobre las temáticas abordadas.



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

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Fases en el análisis y diseño |
| Objetivo de la actividad | Identificar las actividades correspondientes a cada fase en el análisis y diseño. |
| Tipo de actividad sugerida | Relacionar términos |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | CF8\_Actividad\_didactica.docx |

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. | Video | <https://www.youtube.com/watch?v=xwfzw9paFwo> |

1. **GLOSARIO:**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| Base datos: | conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. |
| DBA: | data base administrador 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. |
| 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. |
| *Multitenant:* | tenencia múltiple o multiinquilino hace referencia a aplicaciones que se diseñan para soportar el mismo sistema para distintos usuarios o empresas, a pesar de estar en la misma base de datos y usar el mismo código fuente el sistema sabe distinguir entre los datos de un usuario o empresa de los otros, sin confundirlos. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Vertex42. (s.f.). *Modelo de* *facturas gratis*. <https://www.vertex42.com/es/excel-factura.html>

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>

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 | Junio de 2021 |
| Peter Emerson Pinchao Solís | Experto temático | Regional Cauca - Centro de Teleinformática y Producción Industrial | Junio de 2021 |
| Ana Catalina Córdoba Sus | Evaluadora instruccional | Regional Antioquia - Centro de Servicios de Salud | Febrero 2024 |
| Olga Constanza Bermúdez Jaimes | Responsable Línea de Producción Antioquia | Regional Antioquia - Centro de Servicios de Salud | Febrero 2024 |

1. **CONTROL DE CAMBIOS**

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

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