INSTITUTO SUPERIOR TECNOLÓGICO

JAPÓN

Amor al conocimiento

GUÍA METODOLÓGICA

BASE DE DATOS

DESARROLLO DE SOFTWARE

COMPILADOR: MGS. DIANA MONCAYO

2019

INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN
GUIA DE APRENDIZAJE

1. IDENTIFICACIÓN DE

Nombre de la Asignatura: Base de Datos	Componentes del Aprendizaje	COGNOSCITIVOS

Resultado del Aprendizaje:

Conoce el significado de las Base de datos y puede comprender como se aplica en las diversas herramientas

Conocer los nuevos métodos para organizar y simplificar la información a través de herramientas de BDD.

Entiende y comprende el significado de las BDD (Bases de Datos) y como aplicarlas en la carrera de Desarrollo de Software.

Organiza sus contenidos en las siguientes unidades de aprendizaje: Consultas, procedimientos almacenados, modelos de base de datos.

Comprende y desarrolla en el estudiante la capacidad de implementar una base de datos empleando adecuadamente los fundamentos de normalización, el modelo entidad-relación y el diseño lógico

Utiliza las técnicas, las habilidades y las herramientas modernas necesarias para la práctica de la ingeniería.

Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

OBJETIVOS:

- Conocer los fundamentos de un Sistema de Administración de Base de Datos.
- Categorizar los modelos de datos basados en los tipos de conceptos que proporcionan para
- Describir la estructura de la base de datos, es decir, datos conceptuales y representación de modelo de datos.
- Describir los principios básicos del modelo de datos relacional.
- Definir la terminología fundamental usada en el modelo de los datos relacional.
- Conocer la manipulación de datos haciendo uso del lenguaje SQL.
- Conocer los fundamentos de un Sistema de Administración de Base de Datos.
- Definir la necesidad de Escalabilidad, Portabilidad en Sistemas de Administración de Base de Datos.
- Conocer Arquitectura de Business Intelligence para la Toma de Decisiones.

COMPETENCIAS

- Conoce y elabora un diseño (conceptual y lógico) apropiado de los archivos, así como la interpelación de estos, que permita almacenar información clave y estratégica.
- Conoce como diseñar, interpretar y analizar base de datos relacionales basados en los requerimientos de información de una organización, optimizando el acceso a los datos de las
  mismas bajo criterios de normalización y algebra relacional.

- Conocer las herramientas, espacios y recursos del Aula Virtual.
- Adquirir habilidades de trabajo en equipo en un entorno virtual de aprendizaje.
- Obtener habilidades de análisis, tratamiento, interpretación, elaboración y estructuración de la información digital.
- Monitorizar y facilitar analíticamente las actividades relacionadas con los conocimientos.

INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN
GUIA DE APRENDIZAJE

Docente de Implementación: Msc. Diana Moncayo				
		Duración: 40 horas		
Unidades	**Competencia**	**Resultados de Aprendizaje**	**Actividades**	**Tiempo de Ejecución**
Unidad I Introducción, Modelos de Información y Sistemas de Base de Datos	a. Historia y Motivación para Sistemas de Información, Dato, Información y Base de Datos b. Componentes de un sistema de base datos, arquitectura e independencia de información	Entiende y comprende los fundamentos para el uso de base de datos.	Práctica en Clase Diapositivas Videos Instalación deaplicaciones	**5 horas**
UNIDAD II Modelamiento de Información; El Modelo Relacional y el Leguaje de Consulta Estructurado (SQL) básico	Desarrollan diseños de modelo de base de datos	Aplican conocimientos iniciales y ponen en práctica modelados de base de datos.	Practica en clase , se analiza una empresa y identifica los datos con los que se van a trabajar	**5 horas**

UNIDAD III Sentencias DDL: CREATE, DROP, ALTER	Comprende la creación de tablas. Creación de relaciones entre tablas. Implementa la integridad de datos a nivel de tabla, dominio y referencia. Normalización de base de datos	Diferencia y aplica sentencias de DDL y DML con criterio en la implementación y manipulación de la base de datos en la solución de un problema de la organización. Diferencia y aplica procedimientos y triggers.	Ejercicios en Clase, con ejemplos de casos reales. Debate estudiantil Prácticas en clase	**8 horas**
UNIDAD IV Sentencias CRUD: SELECT, INSERT, DELETE, UPDATE	Creación de consultas, actualización de registros, registro de información y eliminación	Conoce como consultar la información relacionada. Puede insertar información considerando las relaciones e identidades de los datos.	Ejercicios varios con verificación de resultados. Prácticas en clase Evaluaciones Explosión de casos de estudio	**12 horas**

UNIDAD V Programación Avanzada con el Lenguaje de Consulta Estructurado (SQL) Vistas Procedimientos almacenados FUNCIONES ODBC	a. El lenguaje SQL. Lenguaje de Definición de Datos (DDL) y Lenguaje de Manipulación de Datos (DML), tipos de datos, tablas, esquemas, restricciones, reglas de integridad en SQL b. Vistas, procedimientos almacenados, store procedures.	Analizan y sintetizan sobre la necesidad de información en las organizaciones. Exponen con claridad y discuten el tema de investigación en casos prácticos individuales	Practica calificada 1 Cardinalidad, dependencia y clasificación de una relación, algebra y cálculo relacional. Operadores relacionales.	**10 horas**

2. CONOCIMIENTOS PREVIOS Y RELACIONADOS

Co-requisitos

1.-Identificar los conceptos básicos de la importancia de las Bases de datos y como se aplican en la práctica empresarial.

2.-Realizar practicas en clase que permitan conocer como dar identidad a los datos y como se obtiene información veraz.

3. Conocer los modelos de diseño de base de datos, para que el estudiante pueda visualizar de manera amplia las tablas de datos con las que interactúa

4. El estudiante debe tener instalado en su equipo de trabajo, modelador de base de datos y una base de datos para realizar trabajos y prácticas con sentencias de consulta, inserción , edición y eliminación.

3. UNIDADES TEÓRICAS

- **Desarrollo de las Unidades de Aprendizaje (contenidos)**

A. Base Teórica

UNIDAD I : Introducción, Modelos de Información y Sistemas de Base de Datos

TEMA 1: Introducción a las bases de datos relacionales.

Una base de datos (BD), o mejor dicho, un sistema gestor de bases de datos (SGBD), es un software que gestiona una o más bases de datos y nos permite explotar los datos almacenados en ellas de forma relativamente simple mediante SQL.

Esta es una definición muy simplificada, pero para que el aprendizaje sea distendido lo supondremos así, de ese modo podemos centrarnos en aprender cómo y con qué propósito accedemos a los datos, dejando para el final como creamos, alimentamos o modificamos la BD.

Algunos ejemplos de SGBD son: Oracle, MySQL, MS SQL Server

¿Qué es un dato?

Un dato nos permite describir un objeto. Dicho objeto podemos llamarlo entidad, por ejemplo una casa en la que viven personas. La casa es la entidad y la cantidad de personas que viven en la casa son un dato, que en este caso es numérico.

- Identificación y evaluación de productos de gestores de bases de datos del mercado.
- Definición de necesidades y requerimientos.

TEMA 2 : Estructura mínima de almacenamiento:

Tabla:

Objeto de almacenamiento perteneciente a una BD. Es una estructura en forma de cuadrante donde se almacenan registros o filas de datos. Cada tabla tiene un nombre único en la BD.

Registro:

Cada una de las filas de una tabla, este compuesto por campos o atributos.

Campo:

Cada uno de los “cajoncitos” de un registro donde se guardan los datos. Cada campo tiene un nombre único para la tabla de la cual forma parte, además es de un tipo (naturaleza) determinado, por tanto, no podemos guardar limones en el cajón de las naranjas, en términos informáticos y a modo de ejemplo, no encontraremos un dato alfanumérico (letras y números) en un campo diseñado para guardar datos numéricos. Dedicaremos una lección a los tipos de datos más adelante.

TEMA 3 : Tipos de base de datos.

Hay bases de datos relacionales, como MySQL, SQL Server y Oracle. Como su nombre lo indica utilizan el modelo relacional y siempre es mejor usarlas cuando los datos son consistentes y ya tienes algo planificado.

También existen las no relacionales, como MongoDB y Redis, conocidas como NO-SQL (Not Only SQL). Estas son más flexibles en cuanto a consistencia de datos y se han convertido en una opción que intenta solucionar algunas limitaciones que tiene el modelo relacional.

Existen diferentes clasificaciones de las bases de datos, atendiendo a características puntuales:

Según su variabilidad. Conforme a los procesos de recuperación y preservación de los datos, podemos hablar de:

- Bases de datos estáticas. Típicas de la inteligencia empresarial y otras áreas de análisis histórico, son bases de datos de sólo lectura, de las cuales se puede extraer información, pero no modificar la ya existente.
- Bases de datos dinámicas. Aparte de las operaciones básicas de consulta, estas bases de datos manejan procesos de actualización, reorganización, añadidura y borrado de información.

Según su contenido.

De acuerdo a la naturaleza de la información contenida, pueden ser:

- Bibliográficas. Contienen diverso material de lectura (libros, revistas, etc.) ordenado a partir de información clave como son los datos del autor, del editor, del año de aparición, del área temática o del título del libro, entre otras muchas posibilidades.

- De texto completo. Se manejan con textos históricos o documentales, cuya preservación debe ser a todo nivel y se consideran fuentes primarias.

- Directorios. Listados enormes de datos personalizados o de direcciones de correo electrónico, números telefónicos, etc. Las empresas de servicios manejan enormes directorios clientelares, por ejemplo.

- Especializadas. Bases de datos de información hiperespecializada o técnica, pensadas a partir de las necesidades puntuales de un público determinado que consume dicha información.

TEMA 4: Introducción- Modelo Entidad – Relación

El objetivo principal en el diseño de bases de datos es crear modelos de bases de datos completos normalizados, no redundantes, conceptuales, lógicos y físicos totalmente integrados. La fase de ejecución se incluye estructuras de almacenamiento, carga de datos, entre otros.

Modelo Entidad – Relación (MER)

El modelo de datos entidad – relación (E-R) está basado en una percepción del mundo real consistente en objetos básicos: a) Entidades b) Relaciones

Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de una empresa que representa la estructura completa.

Entidades, Atributos, Relaciones, Claves

Diagrama Entidad – Relación

Agregación

Reducción de los diagramas E – R a tablas

TEMA 5: Entidades y Relaciones

El modelo de datos más extendido es el denominado ENTIDAD/RELACIÓN (E/R) En el modelo E/R se parte de una situación real a partir de la cual se definen entidades y relaciones entre dichas entidades:

Entidad.- Objeto del mundo real sobre el que queremos almacenar información (Ej: una persona). Las entidades están compuestas de atributos que son los datos que definen el objeto (para la entidad persona serían DNI, nombre, apellidos, dirección,...). De entre los atributos habrá uno o un conjunto de ellos que no se repite; a este atributo o conjunto de atributos se le llama clave de la entidad, (para la entidad persona una clave seria DNI). En toda entidad

siempre hay al menos una clave que en el peor de los casos estará formada por todos los atributos de la tabla. Ya que pueden haber varias claves y necesitamos elegir una, lo haremos atendiendo a estas normas:

- Que sea única.
- Que se tenga pleno conocimiento de ella.- ¿Por qué en las empresas se asigna a cada cliente un número de cliente?.
- Que sea mínima, ya que será muy utilizada por el gestor de base de datos.

Relación.- Asociación entre entidades, sin existencia propia en el mundo real que estamos modelando, pero necesaria para reflejar las interacciones existentes entre entidades. Las relaciones pueden ser de tres tipos:

Relaciones 1-1.- Las entidades que intervienen en la relación se asocian una a una (Ej: la entidad HOMBRE, la entidad MUJER y entre ellos la relación MATRIMONIO).

Relaciones 1-n.- Una ocurrencia de una entidad está asociada con muchas (n) de otra (Ej: la entidad EMPERSA, la entidad TRABAJADOR y entre ellos la relación TRABAJAR-EN).

Relaciones n-n.-Cada ocurrencia, en cualquiera de las dos entidades de la relación, puede estar asociada con muchas (n) de la otra y viceversa (Ej: la entidad ALUMNO, la entidad EMPRESA y entre ellos la relación MATRÍCULA).

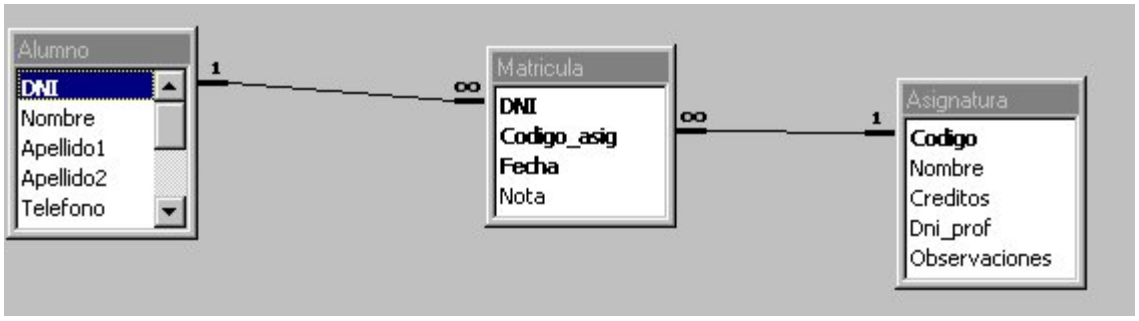


Figura 1: Ejemplo relación de Tablas

TEMA 3: WorkBench Modelador de Base de Datos

Este software nos facilita la creación del Diagrama Entidad Relación y exportarlo a Mysql.

Este programa está disponible en internet en la siguiente dirección web.

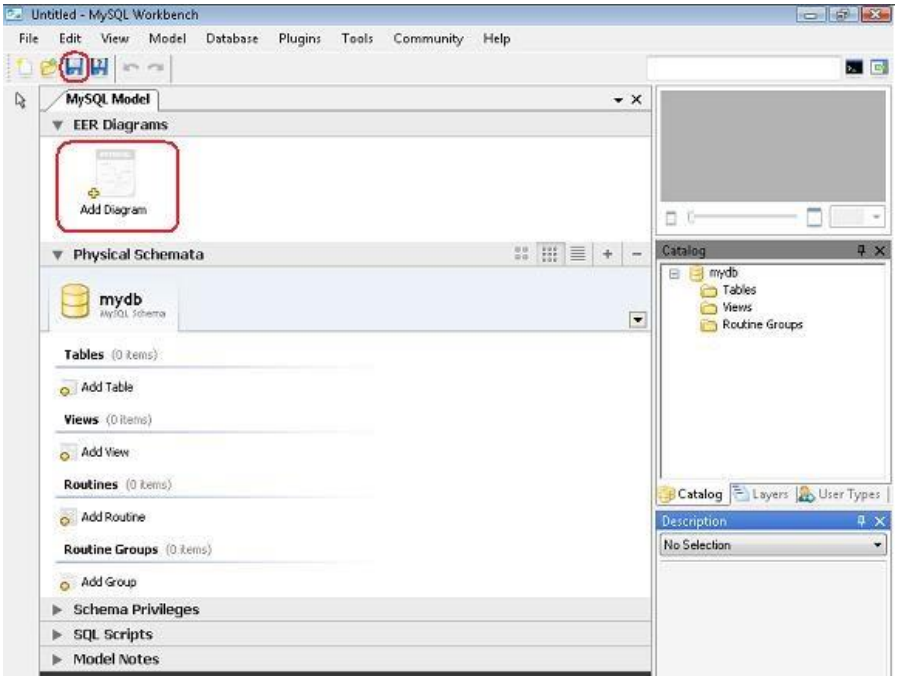


Figura 2: Iniciar creación modelo ER

Para comenzar a trabajar, debemos hacer doble clic en **Add Diagram** y lo primero que hacemos es guardar el archivo haciendo clic en el diskete. Luego de hacer doble clic nos mostrara la siguiente imagen.

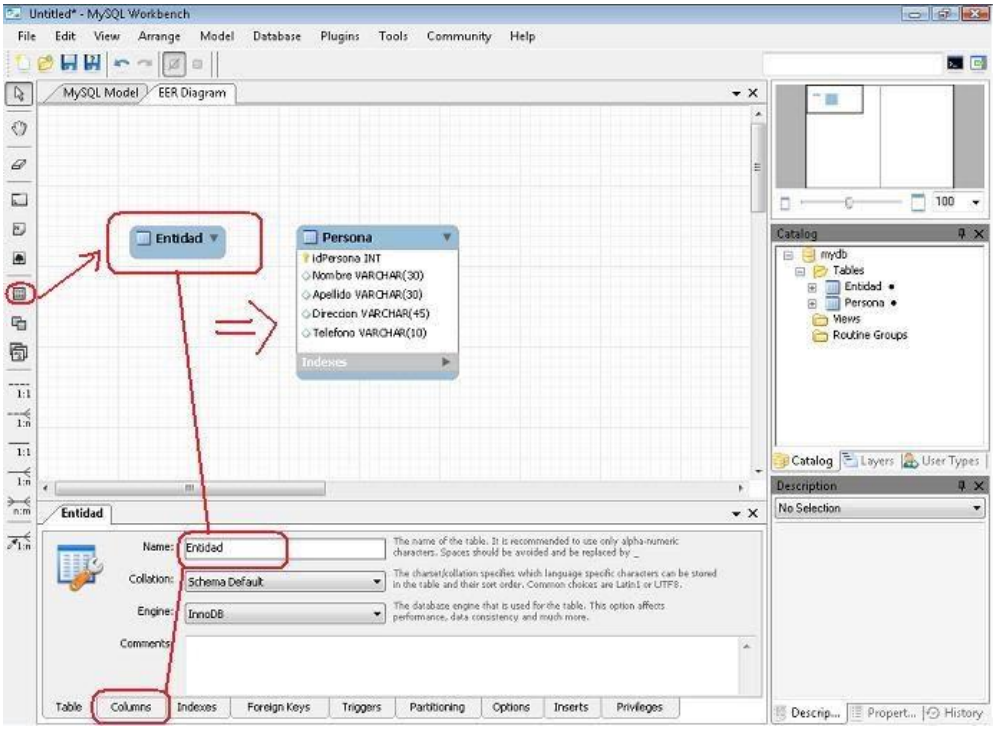


Figura 3: Crear una tabla ER

Al lado izquierdo nos muestra la barra de herramientas entre ellas tenemos crear una tabla, para ello debemos hacer clic en el icono que se encuentra marcado con un circulo al lado izquierdo de la imagen, luego hacemos clic sobre el área de trabajo y nos muestra un rectángulo azul con la palabra tabla el cual debemos hacer doble clic sobre él y editar el nombre y además lograr ingresar los atributos correspondientes a esta tabla, quedando como se aprecia en la segunda tabla. Para el ingreso de los atributos debemos hacer clic en la pestaña de abajo que dice columns como se muestra a continuación:

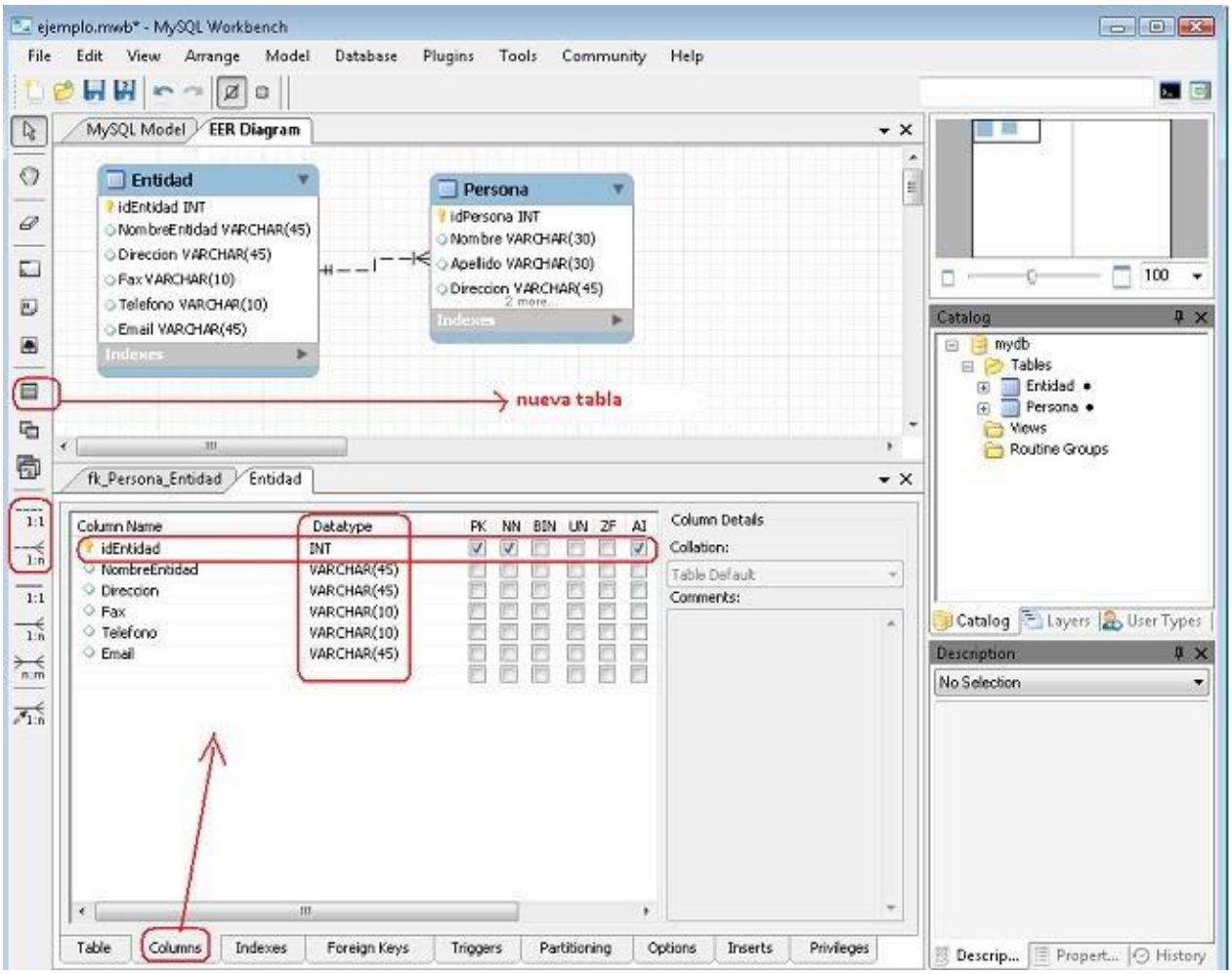


Figura 4: Crear atributos tabla ER

En este caso hemos ingresado algunos atributos a la tabla Entidad y la hemos relacionado con la tabla persona la relación es (1:n), aquí podemos seleccionar el tipo de datos (datatype) con sus respectivos tamaños correspondiente a cada atributo y también darle las características exclusivas de la llave primaria (primary key). Hemos seleccionado otras áreas del programa, por ejemplo donde se crean las tablas, donde están los iconos de relación, los cuales se encuentran como (1:1), (1:n), etc, y también hemos seleccionado la pestaña de columns para que pudiéramos ingresar los atributos de la tabla.

Para exportar este diagrama y lo podamos pasa a mysql debemos realizar la siguiente operación:

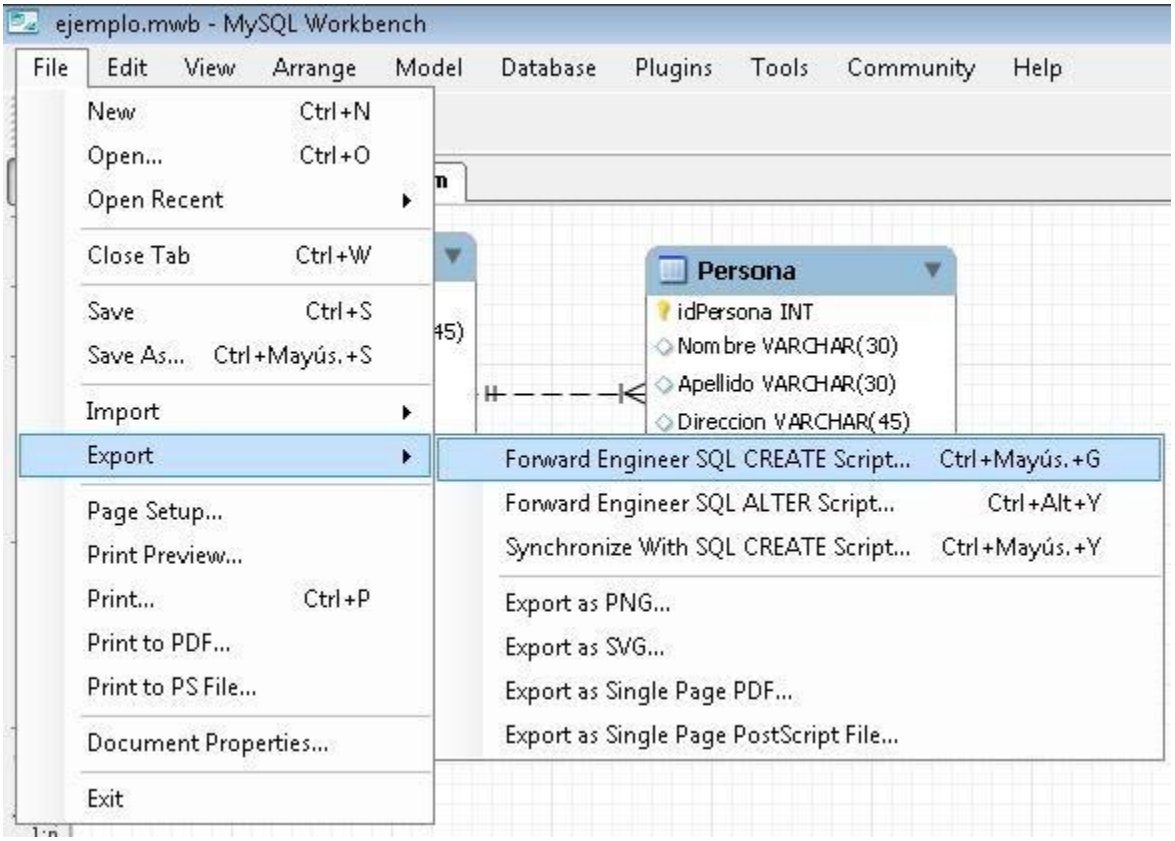


Figura 5: Exportar Script Modelo ER

Nos dirigimos a File luego Exportar y clic en Forward Engineer SQL CREATE Script, y nos muestra la siguiente configuración.

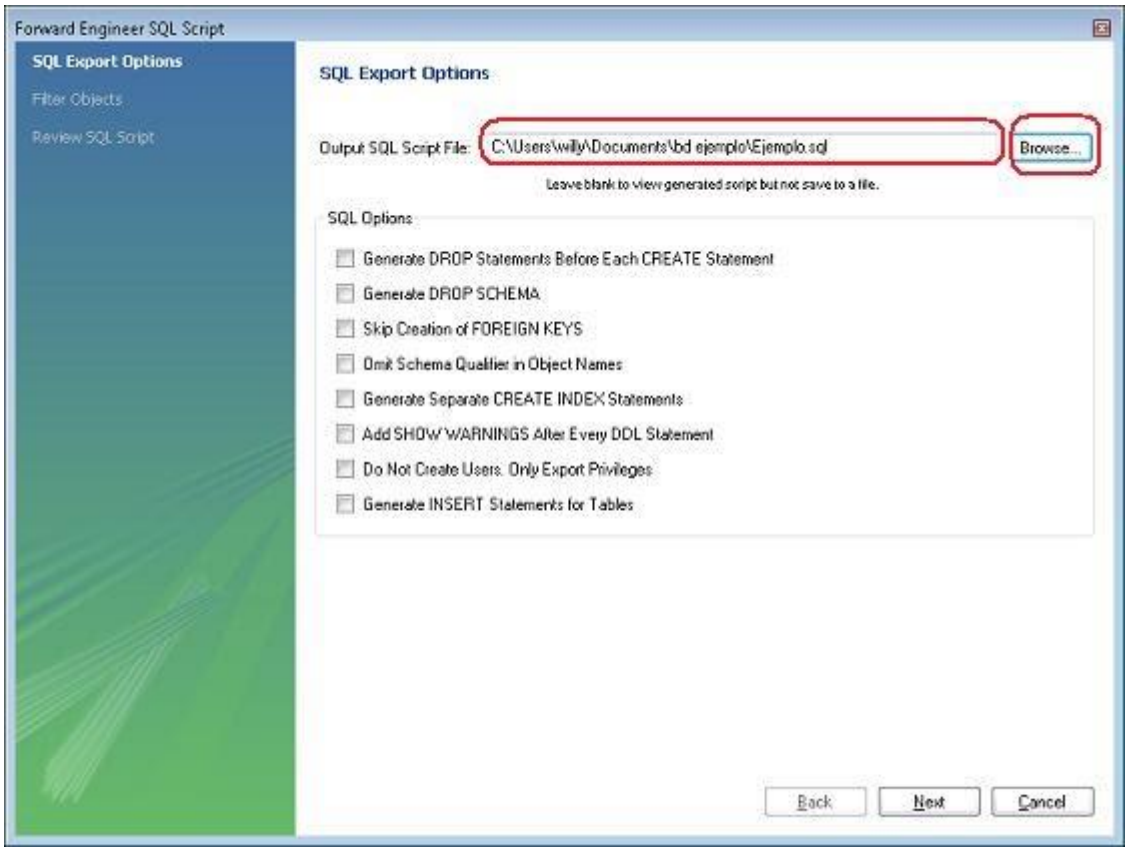


Figura 6: Exportar Script Modelo ER

En esta pantalla debemos hacer clcic en Browse y escoger la ubicacion donde deseamos guardar el archivo sql que más tarde necesitaremos para el procesos en phpmysql. Luego de esto clic en NEXT dos veces y nos muestra la siguiente pantalla.

Para cambiar el nombre de la base de datos, realizamos la siguiente operación desde Notepad.

1. Cargamos el Notepad y abrimos el archivo que exportamos en este caso se llama ejemplo.sql.

2. Nos dirigimos al menú Buscar y hacemos clic en reemplazar. Como lo muestra la imagen.

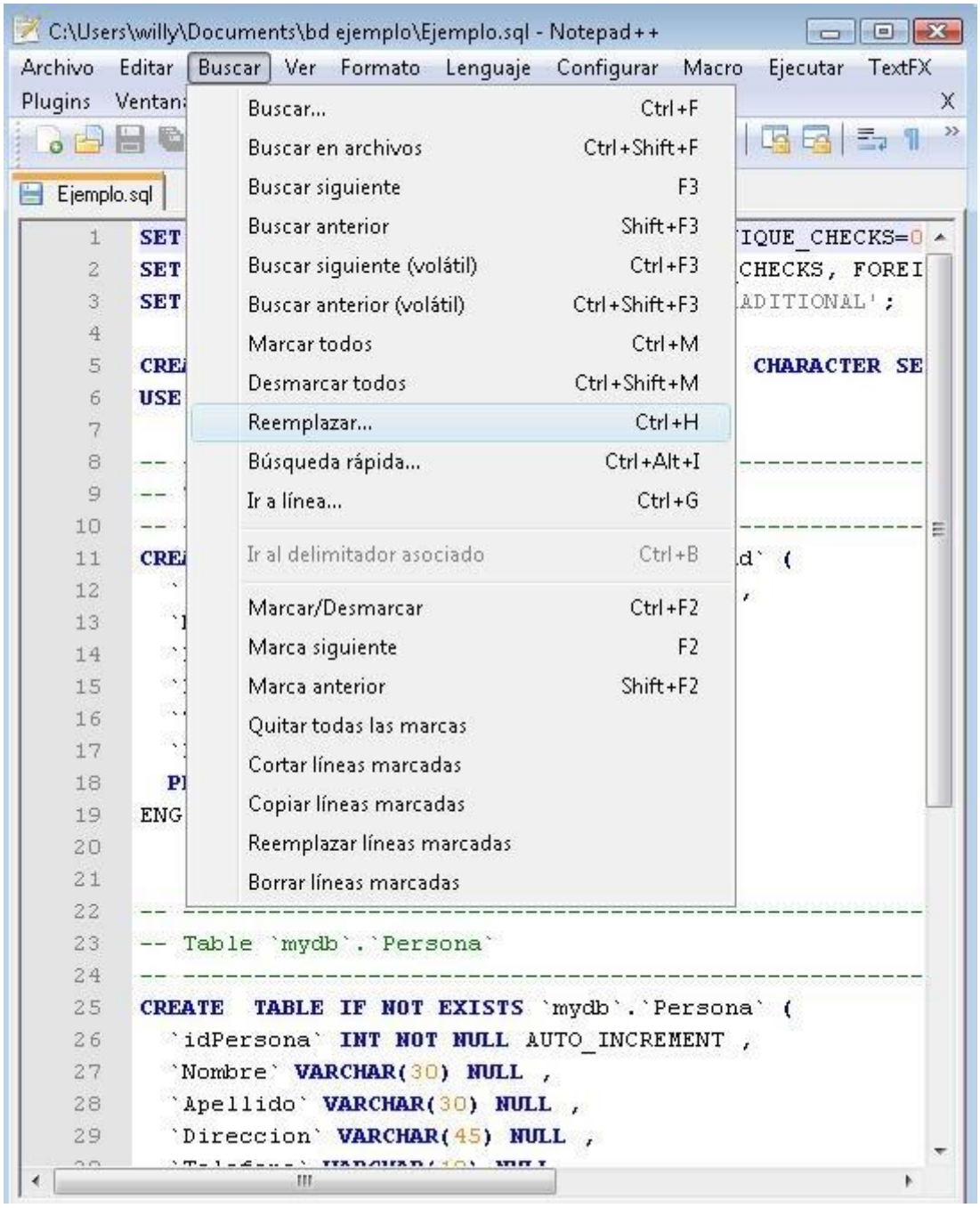


Figura 7: Reemplazar nombre de la base de datos

Nos muestra la siguiente imagen, donde debemos colocar la frase a buscar y la palabra a reemplazar, luego reemplazar todo, aceptar y cerrar, eso es todo. Como lo muestra la imagen.

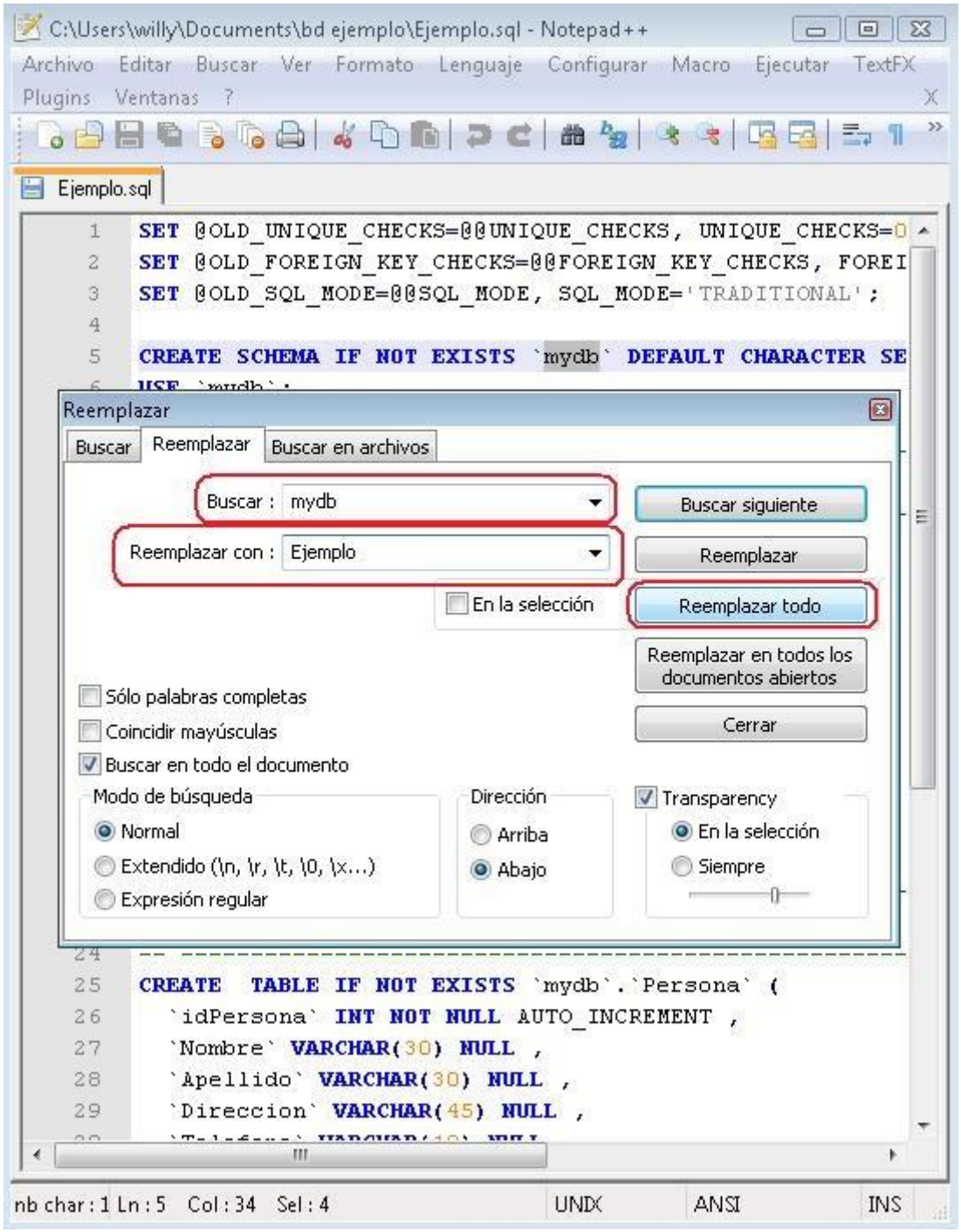


Figura 8: Reemplazar nombre de la base de datos

Luego cuando ya tenemos todo podemos cargar el Xamp y ejecutamos phpmyadmin. Creamos una base de datos llamada Ejemplo y nos dirigimos al menú importar como lo muestra la imagen.

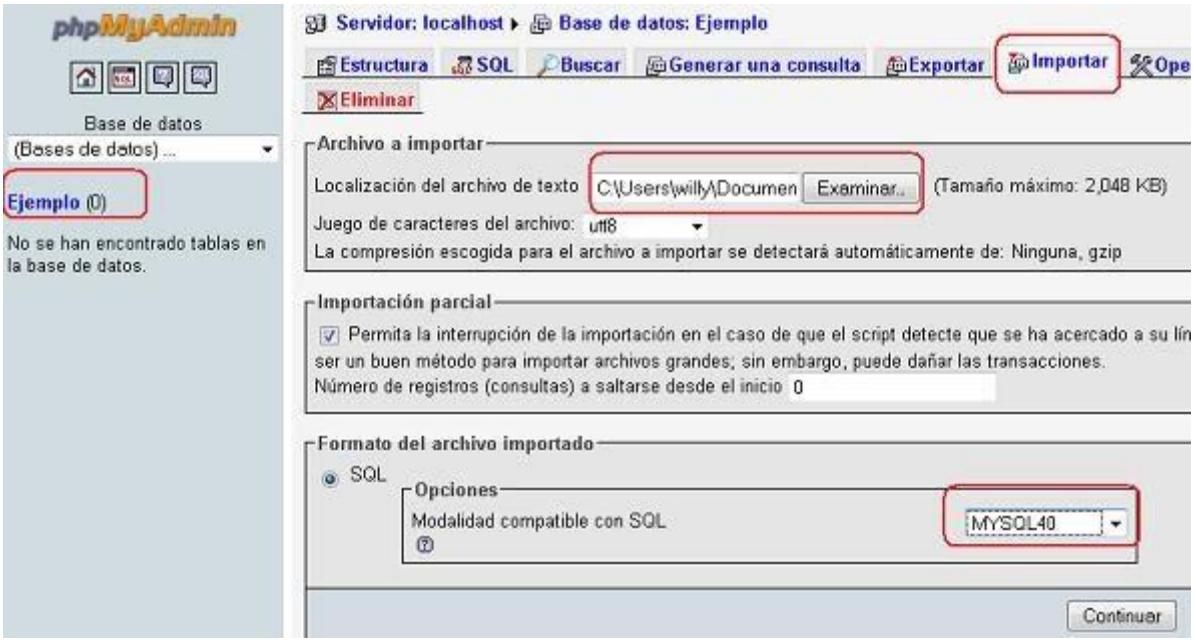


Figura 9: Creando BDD desde phpmyadmin

Al lado izquierdo nos muestra la base de datos ya creada, (para mayor referencia de cómo crear una base de datos dirigirse al menú de la pagina en la sección de wampserver) en este caso Ejemplo (0), al hacer clic en importar nos muestra esta imagen, debemos seleccionar el archivo a importar, para ello seleccionamos examinar y buscamos el archivo y en la parte de abajo. Es bueno tener activa la modalidad de compatibilidad en MYSQL40 que aunque en muchos casos no es necesario no está de más colocarla y luego en continuar y listo, la base de datos ha sido importada con sus tablas.

Al final obtenemos el siguiente resultado.

Figura 10: Creando BDD desde phpmyadmin

UNIDAD II: Modelamiento de Información; El Modelo Relacional y el Leguaje de Consulta Estructurado (SQL) básico

TEMA 1: Sistema Gestor de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener una base de datos, y proporciona acceso controlado a la misma.

En general, un SGBD proporciona los siguientes servicios:

- Permite la **definición de la base de datos** mediante el lenguaje de definición de datos (**DDL – Data Description Language**). Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos.
- Permite la **inserción, actualización, eliminación y consulta de datos** mediante el lenguaje de manejo o manipulación de datos (**DML - Data Manipulation Language**).
- Proporciona un acceso controlado a la base de datos mediante:
    - Un sistema de seguridad, de modo que los usuarios no autorizados no puedan acceder a la base de datos, mediante el lenguaje de control de datos (**DCL - Data Control Language**);
    - Un sistema de integridad que mantiene la integridad y la consistencia de los datos;
    - Un sistema de control de concurrencia que permite el acceso compartido a la base de datos;
    - Un sistema de control de recuperación que restablece la base de datos después de que se produzca un fallo del hardware o del software;
    - **Un diccionario de datos o catálogo** accesible por el usuario que contiene la descripción de los datos de la base de datos.

La principal herramienta de un SGBD es la interfaz de programación con el usuario. Esta interfaz consiste en un lenguaje muy sencillo mediante el cual el usuario interactua con el servidor. Este lenguaje

comúnmente se denomina **SQL, Structure Query Language**, está estandarizado por la ISO 1, es decir, todas las BD que soporten SQL deben tener la misma sintaxis a la hora de aplicar el lenguaje.

Tema 2: Sistema Modelo relacional

En este tipo de sistemas los datos se centralizan en una base de datos común a todas las aplicaciones. Estos serán los sistemas que estudiaremos en este curso.

Sus ventajas son las siguientes:

- Menor redundancia. No hace falta tanta repetición de datos. Aunque, sólo los buenos diseños de datos tienen poca redundancia.
- Menor espacio de almacenamiento. Gracias a una mejor estructuración de los datos.
- Acceso a los datos más eficiente. La organización de los datos produce un resultado más óptimo en rendimiento.
- Datos más documentados. Gracias a los metadatos que permiten describir la información de la base de datos.
- Independencia de los datos y los programas y procesos. Esto permite modificar los datos sin modificar el código de las aplicaciones.
- Integridad de los datos. Mayor dificultad de perder los datos o de realizar incoherencias con ellos.
- Mayor seguridad en los datos. Al limitar el acceso a ciertos usuarios.
- Como contrapartida encontramos los siguientes inconvenientes:

- Instalación costosa. El control y administración de bases de datos requiere de un software y hardware potente.
- Requiere personal cualificado. Debido a la dificultad de manejo de este tipo de sistemas.
- Implantación larga y difícil. Debido a los puntos anteriores. La adaptación del personal es mucho más complicada y lleva bastante tiempo.

- En la siguiente figura se muestra un sistema de información basado en bases de datos. La información está relacionada y no es redundante.

El modelo relacional de bases de datos se rige por algunas normas sencillas:

Todos los datos se representan en forma de tablas (también llamadas “relaciones”, ver nota anterior). Incluso los resultados de consultar otras tablas. La tabla es además la unidad de almacenamiento principal.

Las tablas están compuestas por filas (o registros) y columnas (o campos) que almacenan cada uno de los registros (la información sobre una entidad concreta, considerados una unidad).

Las filas y las columnas, en principio, carecen de orden a la hora de ser almacenadas. Aunque en la implementación del diseño físico de cada SGBD esto no suele ser así. Por ejemplo, en SQL Server si añadimos una clave de tipo "Clustered" a una tabla haremos que los datos se ordenen físicamente por el campo correspondiente.

El orden de las columnas lo determina cada consulta (que se realizan usando SQL).

Cada tabla debe poseer una clave primaria, esto es, un identificador único de cada registro compuesto por una o más columnas.

Para establecer una relación entre dos tablas es necesario incluir, en forma de columna, en una de ellas la clave primaria de la otra. A esta columna se le llama clave externa. Ambos conceptos de clave son extremadamente importantes en el diseño de bases de datos.

Basándose en estos principios se diseñan las diferentes bases de datos relacionales, definiendo un diseño conceptual y un diseño lógico, que luego se implementa en el diseño físico usando para ello el gestor de bases de datos de nuestra elección (por ejemplo SQL Server).

Por ejemplo, consideremos la conocida base de datos Northwind de Microsoft.

Esta base de datos representa un sistema sencillo de gestión de pedidos para una empresa ficticia. Existen conceptos que hay que manejar como: proveedores, empleados, clientes, empresas de transporte, regiones geográficas, y por supuesto pedidos y productos.

El diseño conceptual de la base de datos para manejar toda esta información se puede ver en la siguiente figura, denominada diagrama Entidad/Relación o simplemente diagrama E-R:

Northwind_EF
(pulsa para aumentar)

Como vemos existen tablas para representar cada una de estas entidades del mundo real: Proveedores (Suppliers), Productos, Categorías de productos, Empleados, Clientes, Transportistas (Shippers), y Pedidos (Orders) con sus correspondientes líneas de detalle (Order Details).

Además están relacionadas entre ellas de modo que, por ejemplo, un producto pertenece a una determinada categoría (se relacionan por el campo CategoryID) y un proveedor (SupplierID), y lo mismo con las demás tablas.

Cada tabla posee una serie de campos que representan valores que queremos almacenar para cada entidad. Por ejemplo, un producto posee los siguientes atributos que se traducen en los campos correspondientes para almacenar su información: Nombre (ProductName), Proveedor (SupplierID, que identifica al proveedor), Categoría a la que pertenece (CategoryID), Cantidad de producto por cada unidad a la venta (QuantityPerUnit), Precio unitario (UnitPrice), Unidades que quedan en stock (UnitsInStock), Unidades de ese producto que están actualmente en pedidos (UnitsOnOrder), qué cantidad debe haber para que se vuelva a solicitar más producto al proveedor (ReorderLevel) y si está descatalogado o no (Discontinued).

Los campos marcados con "PK" indican aquellos que son claves primarias, es decir, que identifican de manera única a cada entidad. Por ejemplo, ProductID es el identificador único del producto, que será por regla general un número entero que se va incrementando cada vez que introducimos un nuevo producto (1, 2, 3, etc..).

Los campos marcados como "FK" son claves foráneas o claves externas. Indican campos que van a almacenar claves primarias de otras tablas de modo que se puedan relacionar con la tabla actual. Por ejemplo, en la tabla de productos el campo CategoryID está marcado como "FK" porque en él se guardará el identificador único de la categoría asociada al producto actual. En otras palabras: ese campo almacenará el valor de la clave primaria (PK) de la tabla de categorías que identifica a la categoría en la que está ese producto.

Los campos marcados con indicadores que empiezan por "I" (ej: "I1") se refieren a índices. Los índices generan información adicional para facilitar la localización más rápida de registros basándose en esos campos. Por ejemplo, en la tabla de empleados (Employees) existe un índice "I1" del que forman parte los campos Nombre y Apellidos (en negrita además porque serán también valores únicos) y que indica que se va a facilitar la locación de los clientes mediante esos datos. También tiene otro índice "I2" en el campo del código postal para localizar más rápidamente a todos los clientes de una determinada zona.

Los campos marcados con indicadores que empiezan con "U" (por ejemplo U1) se refieren a campo que deben ser únicos. Por ejemplo, en la tabla de categorías el nombre de ésta (CategoryName) debe ser único, es decir, no puede haber -lógicamente- dos categorías con el mismo nombre.

Como vemos, un diseño conceptual no es más que una representación formal y acotada de entidades que existen en el mundo real, así como de sus restricciones, y que están relacionadas con el dominio del problema que queremos resolver.

TEMA 2: Modelo lógico

Una vez tenemos claro el modelo E-R debemos traducirlo a un modelo lógico directamente en el propio sistema gestor de bases de datos (Oracle, MySQL, SQL Server...). Si hemos utilizado alguna herramienta profesional para crear el diagrama E-R, seguramente podremos generar automáticamente las instrucciones necesarias para crear la base de datos.

La mayoría de los generadores de diagramas E-R (por ejemplo Microsoft Visio) ofrecen la capacidad de exportar el modelo directamente a los SGBD más populares.

Entonces, todo este modelo conceptual se traduce en un modelo lógico que trasladaremos a la base de datos concreta que estemos utilizando y que generalmente será muy parecido. Por ejemplo, este es el mismo modelo anterior, mostrado ya como tablas en un diagrama de SQL Server:

UNIDAD III: Sentencias DDL: CREATE, DROP, ALTER

TEMA 1: Introducción

Lenguaje de definición de datos

Un lenguaje de definición de datos (Data Definition Language, DDL por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

La definición de la estructura de la base de datos incluye tanto la creación inicial de los diferentes objetos que formarán la base de datos, como el mantenimiento de esa estructura. Las sentencias del DDL utilizan unos verbos que se repiten para los distintos objetos. Por ejemplo para crear un objeto nuevo el verbo será CREATE y a continuación el tipo de objeto a crear. CREATE DATABASE es la sentencia para crear una base de datos, CREATE TABLE nos permite crear una nueva tabla, CREATE INDEX crear un nuevo índice… Para eliminar un objeto utilizaremos el verbo DROP (DROP TABLE, DROP INDEX…) y para modificar algo de la definición de un objeto ya creado utilizamos el verbo ALTER (ALTER TABLE, ALTER INDEX…).

Las principales funcionalidades de SQL como lenguaje de definición (DDL) son la creación, modificación y borrado de las tablas que componen la base de datos, así como de los índices,

vistas, sinónimos, permisos, etc. que pudieran definirse sobre las mismas. Este documento introduce los comandos para el trabajo básico con tablas.

- CREATE TABLE: Crear una tabla
- SHOW TABLES: mostrar tablas
- DROP TABLE<nombre de tabla>: Borrar tabla
- DESCRIBE <nombre de tabla> Mostrar estructura de una tabla

Instrucciones DDL

Es el lenguaje que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos.

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, procedimientos almacenados e índices
DROP	Empleado para eliminar tablas, procedimientos almacenados e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos
TRUNCATE	Se utiliza para borrar toda la tabla y volverla a crear

TEMA 2: Sintaxis para sentencias DDL

Instrucción CREATE

Creación Schema en MySQL:

CREATE SCHEMA `Programas – Desarrollo Humano Integral.'

Creación de tabla Tipos_afiliacion. Sus capos son Codigo_Afiliacion y Descripcion, la clave primeria corresponde al campo Codigo_Afiliacion:

CREATE TABLE 'Tipos_Afiliacion' (

`Codigo Afiliacion` DECIMAL(10,0) NOT NULL,

`Descripcion` VARCHAR(45) NOT NULL,

CREATE TABLE `Programas` (

`Codigo_Programa` DECIMAL(10,0) NOT NULL,

`Descripcion` VARCHAR(45) NULL,

`Codigo_Facilitador` DECIM+++AL(10,0) NULL,

`Disponible_Desde` DATE NULL,

`Disponible_Hasta` DATE NULL,

PRIMARY KEY (`Codigo_Programa`),

CONSTRAINT `Codigo_Facilitador`

FOREIGN KEY (`Codigo_Facilitador`)

REFERENCES `Facilitadores` (`Codigo`)

A continuación se muestra la sintaxis para la sentencias DDL su respectivo Ejemplos.

DDL:

Sentencia Create Table: Este comandó sirve para crear una nueva base de datos, tabla o procedimiento de almacenamiento. Consiste en crear un objeto dentro del Sistema que dependiendo del programador puede soportar la creación de tablas.

Ejemplo de la Sentencia CREATE TABLE:

- '' CREATE [TEMPORARY] TABLE [nombre_tabla] ([definicion_columna]) [parámetro_tabla]. ''

Drop: Este comandó permite eliminar un objeto de la base de dato.

Ejemplo de la Sentencia DROP:

- '' DROP objeto_eliminar;
- DROP TABLE myTable;
- DROP SEQUENCE mySequence;
- DROP INDEX myIndex; ''

ALTER: Este comandó permite agregar, borrar o modificar columnas en una tabla existente

Ejemplo Sentencia ALTER

- -Para agregar:
- '' ALTER TABLE nombre_tabla; ADD column_name tipo_datos; ''
- -Para borrar:
- '' ALTER TABLE nombre_tabla;DROP COLUMN nombre_columna; ''

Truncate: Borra todo el contenido de la tabla, solo la data que contaba.

Ejemplo Sentencia Truncate

- '' TRUNCATE TABLE Nombre_Tabla; ''

UNIDAD VI : Sentencias CRUD: SELECT, INSERT, DELETE, UPDATE

TEMA 1: Operaciones básicas de manipulación de datos en SQL

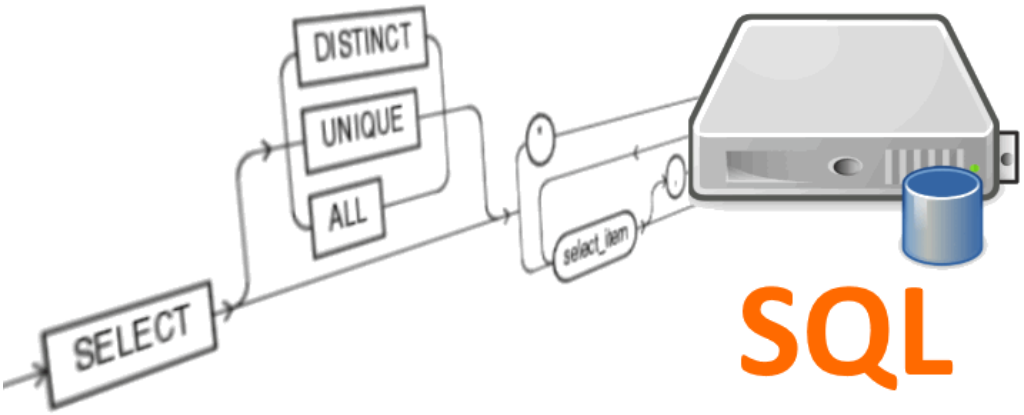


Figura 11: Representación Sentencias CRUD

Como hemos visto, las instrucciones DML (Data Manipulation Language – Lenguaje de Manipulación de Datos) trabajan sobre los datos almacenados en nuestro SGBD, permitiendo consultarlos o modificarlos.

En general a las operaciones básicas de manipulación de datos que podemos realizar con SQL se les denomina operaciones CRUD (de Create, Read, Update and Delete, o sea, Crear, Leer, Actualizar y Borrar, sería CLAB en español, pero no se usa). Lo verás utilizado de esta manera en muchos sitios, así que apréndete ese acrónimo.

Hay cuatro instrucciones para realizar estas tareas:

- INSERT: Inserta filas en una tabla. Se corresponde con la “C” de CRUD.
- SELECT: muestra información sobre los datos almacenados en la base de datos. Dicha información puede pertenecer a una o varias tablas. Es la “R”.
- UPDATE: Actualiza información de una tabla. Es, obviamente, la “U”.
- DELETE: Borra filas de una tabla. Se corresponde con la “D”.

Consulta de datos

Ahora nos vamos a centrar en la “R” de CRUD, es decir, en cómo recuperar la información que nos interesa de dentro de una base de datos, usando para ello el lenguaje de consulta o SQL. Ya nos preocuparemos luego de cómo llegamos a introducir los datos primeramente.

Para realizar consultas sobre las tablas de las bases de datos disponemos de la instrucción SELECT. Con ella podemos consultar una o varias tablas. Es sin duda el comando más versátil del lenguaje SQL.

Existen muchas cláusulas asociadas a la sentencia SELECT (GROUP BY, ORDER, HAVING, UNION). También es una de las instrucciones en la que con más frecuencia los motores de bases de datos incorporan cláusulas adicionales al estándar, que es el que veremos aquí.

Vamos a empezar viendo las consultas simples, basadas en una sola tabla. Veremos cómo obtener filas y columnas de una tabla en el orden en que nos haga falta.

El resultado de una consulta SELECT nos devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene filas/registros, con una serie de campos/columnas. Igual que cualquier tabla de la base de datos. Sin embargo esta tabla está en memoria mientras la utilicemos, y luego se descarta. Cada vez que ejecutamos la consulta se vuelve a calcular el resultado.

La sintaxis básica de una consulta SELECT es la siguiente (los valores opcionales van entre corchetes):

```
SELECT [ ALL / DISTINC ] [ * ] / [ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ]
```

A continuación analizaremos cada una de las partes de la consulta para entenderla mejor.

SELECT

Permite seleccionar las columnas que se van a mostrar y en el orden en que lo van a hacer. Simplemente es la instrucción que la base de datos interpreta como que vamos a solicitar información.

ALL / DISTINCT

ALL es el valor predeterminado, especifica que el conjunto de resultados puede incluir filas duplicadas. Por regla general nunca se utiliza.

DISTINCT

Especifica que el conjunto de resultados sólo puede incluir filas únicas. Es decir, si al realizar una consulta hay registros exactamente iguales que aparecen más de una vez, éstos se eliminan. Muy útil en muchas ocasiones.

Nombres de campos

Se debe especificar una lista de nombres de campos de la tabla que nos interesan y que por tanto queremos devolver. Normalmente habrá más de uno, en cuyo caso separamos cada nombre de los demás mediante comas.

Se puede anteponer el nombre de la tabla al nombre de las columnas, utilizando el formato Tabla.Columna. Además de nombres de columnas, en esta lista se pueden poner constantes, expresiones aritméticas, y funciones, para obtener campos calculados de manera dinámica.

Si queremos que nos devuelva todos los campos de la tabla utilizamos el comodín “*” (asterisco).

Los nombres indicados deben coincidir exactamente con los nombre de los campos de la tabla, pero si queremos que en nuestra tabla lógica de resultados tengan un nombre diferente podemos utilizar:

AS

Permite renombrar columnas si lo utilizamos en la cláusula SELECT, o renombrar tablas si lo utilizamos en la cláusula FROM. Es opcional. Con ello podremos crear diversos alias de columnas y tablas. Enseguida veremos un ejemplo.

FROM

Esta cláusula permite indicar las tablas o vistas de las cuales vamos a obtener la información. De momento veremos ejemplos para obtener información de una sola tabla.

Como se ha indicado anteriormente, también se pueden renombrar las tablas usando la instrucción “AS”.

WHERE

Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

Condiciones

Son expresiones lógicas a comprobar para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

> (Mayor)

>= (Mayor o igual)

< (Menor)

<= (Menor o igual)

= (Igual)

<> o != (Distinto)

IS [NOT] NULL (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)

Se dice que una columna de una fila es NULL si está completamente vacía. Hay que tener en cuenta que si se ha introducido cualquier dato, incluso en un campo alfanumérico si se introduce una cadena en blanco o un cero en un campo numérico, deja de ser NULL.

LIKE: para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: "%" y "_". Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda. Por ejemplo:

- El nombre empieza por A: Nombre LIKE 'A%'
- El nombre acaba por A: Nombre LIKE '%A'
- El nombre contiene la letra A: Nombre LIKE '%A%'
- El nombre empieza por A y después contiene un solo carácter cualquiera: Nombre LIKE 'A_'
- El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: Nombre LIKE 'A_E%'

BETWEEN: para un intervalo de valores. Por ejemplo:

Clientes entre el 30 y el 100: CodCliente BETWEEN 30 AND 100

Clientes nacidos entre 1970 y 1979: FechaNac BETWEEN '19700101' AND '19791231'

IN( ): para especificar una relación de valores concretos. Por ejemplo: Ventas de los Clientes 10, 15, 30 y 75: CodCliente IN(10, 15, 30, 75)

Por supuesto es posible combinar varias condiciones simples de los operadores anteriores utilizando los operadores lógicos OR, AND y NOT, así como el uso de paréntesis para controlar la prioridad de los operadores (como en matemáticas). Por ejemplo: … (Cliente =

100 AND Provincia = 30) OR Ventas > 1000 … que sería para los clientes de las provincias 100 y 30 o cualquier cliente cuyas ventas superen 1000.

ORDER BY

Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

ASC / DESC

ASC es el valor predeterminado, especifica que la columna indicad en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica DESC se ordenará de forma descendente (de mayor a menor).

Por ejemplo, para ordenar los resultados de forma ascendente por ciudad, y los que sean de la misma ciudad de forma descendente por nombre, utilizaríamos esta cláusula de ordenación:

ORDER BY Ciudad, Nombre DESC …

Como a la columna Ciudad no le hemos puesto ASC o DESC se usará para la misma el valor predeterminado (que es ASC)

OJO: Aunque al principio si aún no se está habituado, pueda dar la impresión de que se ordena por ambas columnas en orden descendente. Si es eso lo que queremos deberemos escribir … ORDER BY Ciudad DESC, Nombre DESC …

Algunos ejemplos

Para terminar este repaso a las consultas simples practicarlas un poco, veamos algunos ejemplos con la base de datos Northwind en SQL Server:

- Mostrar todos los datos de los Clientes de nuestra empresa:

SELECT * FROM Customers

- Mostrar apellido, ciudad y región (LastName, city, region) de los empleados de USA (nótese el uso de AS para darle el nombre en español a los campos devueltos):

SELECT E.LastName AS Apellido, City AS Ciudad, Region
FROM Employees AS E
WHERE Country = 'USA'

- Mostrar los clientes que no sabemos a qué región pertenecen (o sea, que no tienen asociada ninguna región) :

SELECT * FROM Customers WHERE Region IS NULL

- Mostrar las distintas regiones de las que tenemos algún cliente, accediendo sólo a la tabla de clientes:

SELECT DISTINCT Region FROM Customers WHERE Region IS NOT NULL

- Mostrar los clientes que pertenecen a las regiones CA, MT o WA, ordenados por región ascendentemente y por nombre descendentemente.

CODE SELECT * FROM Customers WHERE Region IN('CA', 'MT', 'WA')
ORDER BY Region, CompanyName DESC

- Mostrar los clientes cuyo nombre empieza por la letra "W":

SELECT * FROM Customers WHERE CompanyName LIKE 'W%'

- Mostrar los empleados cuyo código está entre el 2 y el 9:

SELECT * FROM Employees WHERE EmployeeID BETWEEN 2 AND 9

- Mostrar los clientes cuya dirección contenga "ki":

SELECT * FROM Customers WHERE Address LIKE '%ki%'

UNIDAD V: Programación Avanzada con el Lenguaje de Consulta Estructurado (SQL)

Vistas, Procedimientos almacenados y funciones.

TEMA 1: Vistas

Una vista es una tabla virtual que se genera a partir de una consulta de selección. Dicho de otro modo. Escribimos una consulta de selección (sobre una o más tablas) para leer los datos, y almacenamos el resultado en una vista. Si usas PHPMyAdmin, la vista te aparecerá como un elemento más de la base de datos en curso. Al pulsar sobre ella, verás el resultado de la consulta de selección, como si fuera una tabla más, aunque con ciertas limitaciones (no puedes editar o añadir registros, por ejemplo) porque lo que estás viendo no es una tabla, sino el resultado de una consulta.

La ventaja que tienes es que si actualizas las tablas originales (en el proceso en curso, o desde cualquier otro punto de la aplicación), la vista queda, automáticamente, actualizada. Por lo tanto, ya no tienes que repetir la consulta de selección para recuperar los nuevos datos de la tabla. Sólo tienes que mirar en la vista, que tien todos los datos de la consulta original actualizados en tiempo real.

CREAR Y USAR LAS VISTAS

La forma de crear una vista sobre una consulta es, genéricamente, la siguiente:

CREATE VIEW IF NOT EXISTS nombre_de_la_vista AS consulta_de_seleccion;

En el caso de la consulta anterior, esto podría quedar así:

```
CREATE VIEW IF NOT EXISTS usuarios_con_provincia AS
SELECT
usuarios.id,
usuarios.login,
usuarios.premium,
provincias.provincia
FROM usuarios
INNER JOIN provincias
```

```
ON provincias.id = usuarios.id_provincia
ORDER BY usuarios.id;
```

Ahora, para recuperar los resultados actualizados en tiempo real sólo tenemos que recuperar la vista, así:

```
SELECT * FROM nombre_de_la_vista;
```

Esto presenta tres grandes ventajas:

Esta última consulta se ejecuta más rápido.

Cómo hemos dicho, está actualizada si se han modificado las tablas originales.

Es mucho más simple y corta de escribir que la consulta original. Por lo tanto, si tenemos que hacer la consulta de lectura en varios puntos de nuestra aplicación, estamos obteniendo un código más limpio y optimizado.

Sintaxis de ALTER VIEW

```
ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
  VIEW nombre_vista [(columnas)]
       AS sentencia_select
      [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Esta sentencia modifica la definición de una vista existente. La sintaxis es semejante a la empleada en CREATE VIEW. Consulte Sección 21.2, “Sintaxis de CREATE VIEW”. Se requiere que posea los permisos CREATE VIEW y DELETE para la vista, y algún privilegio en cada columna seleccionada por la sentencia SELECT.

TEMA 2: Procedimientos almacenados

Un procedimiento almacenado o también conocido como Stored Procedure, es un conjunto de instrucciones en lenguaje SQL que realizan una tarea determinada, por ejemplo podríamos crear un procedimiento almacenado para sumar 2 números.

Por lo general se usa un procedimiento almacenado cuando la tarea es muy pesada y hacerla desde la aplicación sería muy costoso.

Ventajas

La ventaja principal es que se libera cualquier tarea pesada que podría tener nuestra aplicación, aunque se debe tener en cuenta y hacerlo en circunstancias necesarias ya que crear de forma excesiva procedimientos almacenados, puede ser una mala práctica ya que se puede tener a futuro problemas en cuanto a la organización del código entre otras cosas.

La sentencia para crear un procedimiento almacenado es de la siguiente forma:

```
CREATE PROCEDURE `nombre` ([parameter[,...]])
BEGIN
    //instrucciones
END
```

Los parámetros indican la entrada y salida de datos que podríamos usar para el procedimiento y la salida de datos que devolvería el procedimiento almacenado.

En el siguiente ejemplo creamos un procedimiento almacenado para contar el número de productos por código:

```
CREATE PROCEDURE `contar_productos` (IN codigo varchar(5),
  OUT nproductos INT)
BEGIN
SELECT
  COUNT(*)
INTO
  nproductos
FROM
  productos
WHERE
  cprincipal=codigo;
END
```

Para saber si tu procedimiento sirve, deberías llamarlo de la siguiente manera:

```
1 CALL contar_productos('001', @cantidad);
2 select  @cantidad;
```

Parámetros

Como has visto, los parámetros se definen separados por una coma. Los parámetros de los procedimientos almacenados de MySQL pueden ser de tres tipos:

IN: Es el tipo de parámetro que se usa por defecto. La aplicación o código que invoque al procedimiento tendrá que pasar un argumento para este parámetro. El procedimiento trabajará con una copia de su valor, teniendo el parámetro su valor original al terminar la ejecución del procedimiento.

OUT: El valor de este parámetros pude ser cambiado en el procedimiento, y además su valor modificado será enviado de vuelta al código o programa que invoca el procedimiento.

INOUT: Es una mezcla de los dos conceptos anteriores. La aplicación o código que invoca al procedimiento puede pasarle un valor a éste, devolviendo el valor modificado al terminar la ejecución. En caso de resultarte confuso, echa un ojo al ejemplo que verás más adelante.

Procedimiento almacenado con parámetros IN

Vamos a obtener los productos que tienen un determinado estado. Para crear el procedimiento, ejecuta estas sentencias SQL:

```
DELIMITER $$
CREATE PROCEDURE obtenerProductosPorEstado(IN nombre_estado VARCHAR(255))
BEGIN
    SELECT *
    FROM productos
    WHERE estado = nombre_estado;
END$$
DELIMITER
```

El nombre del estado está contenido en el parámetro nombre_estado que hemos definido como IN. Suponiendo que quieras obtener los productos con estado disponible, tendrías que invocar al procedimiento de este modo:

```
CALL obtenerProductosPorEstado('disponible')
```

Este será el resultado:

```
+---------+--------------+------------+---------+
| id      | nombre       | estado     | precio  |
+---------+--------------+------------+---------+
| 1       | Producto A   | disponible | 8.00    |
| 2       | Producto B   | disponible | 1.50    |
+---------+--------------+------------+---------+
```

Procedimiento almacenado con parámetros OUT

Vamos a obtener el número de productos según su estado. Para crear el procedimiento, ejecuta estas sentencias SQL:

```
DELIMITER $$
CREATE PROCEDURE contarProductosPorEstado(
    IN nombre_estado VARCHAR(25),
    OUT numero INT)
BEGIN
    SELECT count (id)
    INTO numero
    FROM productos
    WHERE estado = nombre_estado;
END$$
DELIMITER
```

Al igual que antes, pasamos el estado como nombre_estado, definido como IN. También definimos número como parámetro OUT. Suponiendo que quieras obtener el número de productos con estado disponible, debes llamar al procedimiento de este modo:

```
CALL contarProductosPorEstado('disponible', @numero);
SELECT @numero AS disponibles;
```

Este será el resultado:

```
+-------------+
| disponibles |
+-------------+
| 2           |
+-------------+
```

Para obtener el número de productos agotados, debemos invocar al procedimiento de este modo:

```
CALL contarProductosPorEstado('agotado', @numero);
SELECT @numero AS agotados;
```

Este será el resultado:

```
+----------+
| agotados |
+----------+
| 1        |
+----------+
```

Eliminar un procedimiento almacenado

Para borrar un procedimiento almacenado haciendo uso de la sentencia DROP PROCEDURE. Por ejemplo, si quieres eliminar el procedimiento venderProducto del ejemplo anterior, tendrás que ejecutar esta sentencia:

```
DROP PROCEDURE venderProducto;
```

Tema 3 : Funciones

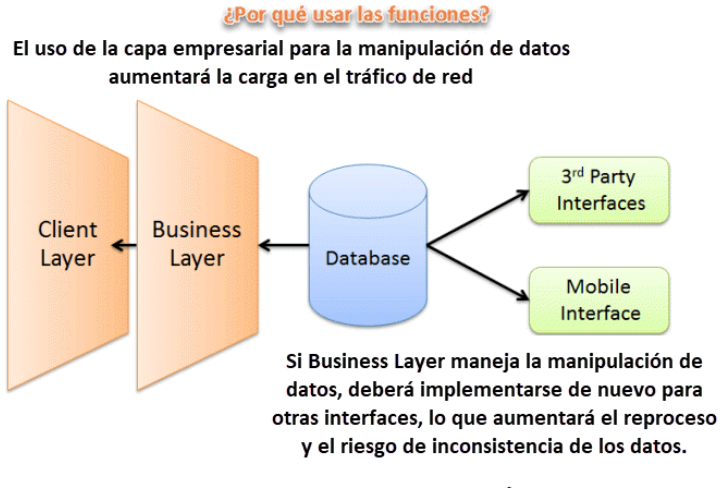


Figura 12: Uso de funciones

Es una rutina creada para tomar unos parámetros, procesarlos y retornar en una salida.

MySQL tiene muchas funciones que podemos usar en procedimientos almacenados y consultas, pero en ocasiones se puede crear funciones para hacer sentencias más especializadas.

Vamos a ver cómo crear funciones en MySQL:

```
DELIMITER //
CREATE FUCNTION holaMundo() RETURNS VARCHAR(20)
BEGIN
   RETURN 'HolaMundo';
END
//
```

Para comprobar que funciona tecleamos lo siguiente en la consola de MySQL :

```
Select holaMundo();
```

Ejemplo:

Realizamos un "select" sobre "preciocomercial" que muestre la placa, motor y calcule el iva asl 12%

```
DELIMITER |
CREATE FUNCTION f_calculoiva(avalor float, aiva float)
RETURNS decimal(9,2)
BEGIN
   DECLARE calculaiva DECIMAL(9,2);
    SET calculaiva =  avalor+((avalor * aiva )/100);
    RETURN calculaiva;
end |
```

```
select veh_codigo,veh_motor,veh_precio_comercial,
f_calculoiva(veh_precio_comercial,12) from d_vehiculo
```

TEMA 4: ODBC

Open Database Connectivity (ODBC) es una API estandarizada que permite la conexión a los servidores de base de datos SQL.

A continuación se detallan los pasos para cubrir una instalación y configuración de base de datos .

1. Descargar el instalador conector ODBC de 32 o 64 bits
2. Instalación de programa

Figura 13: Instalacion conector ODBC

Una vez instalado ya podemos crear las conexiones ODBC, os pondré las rutas para cada uno de los sistemas operativos.

Windows 7 32 bits

Para configurar la conexión ODBC tenemos que ir a la carpeta “C:\Windows\System32” y ejecutar el archivo “odbcad32.exe”

Windows 7 64 bits

Para configurar la conexión ODBC tenemos que ir a la carpeta “C:\Windows\SysWOW64” y ejecutar el archivo “odbcad32.exe”

Para configurar la conexión ODBC tenemos que ir a la carpeta “C:\Windows\SysWOW64” y ejecutar el archivo “odbcad32.exe”

Agregaremos un nuevo ODBC y seleccionaremos el que acabamos de instalar de “MySQL ODBC 5.2 ANSI Driver”

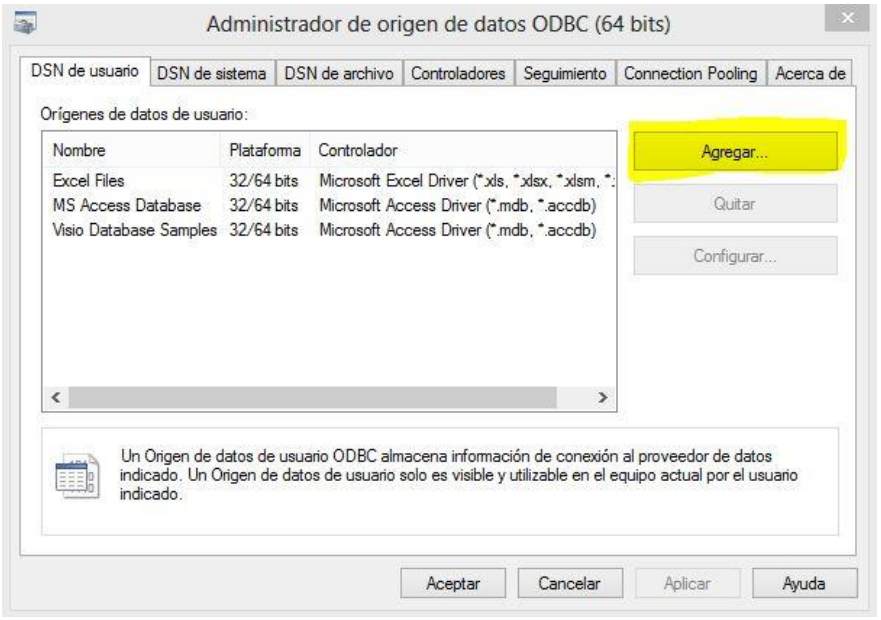


Figura 14: Instalacion conector ODBC

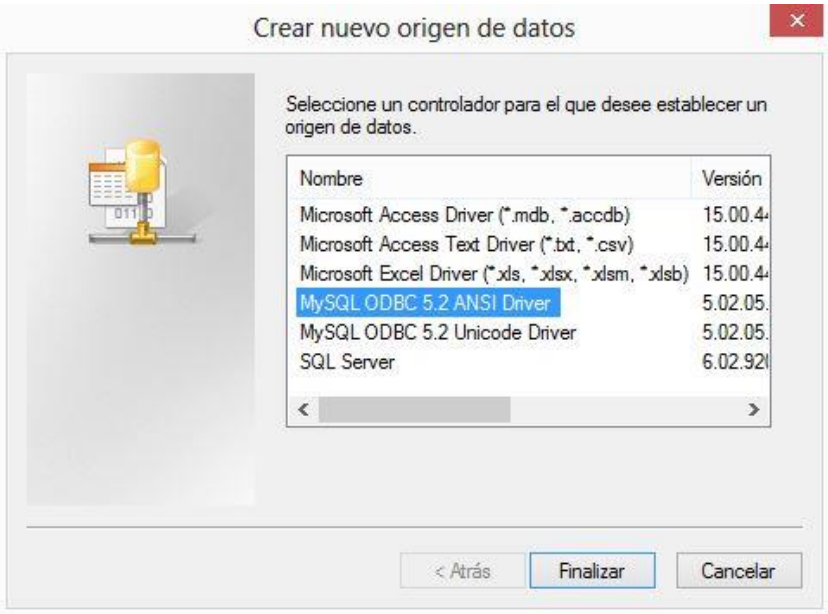


Figura 15: Instalacion conector ODBC

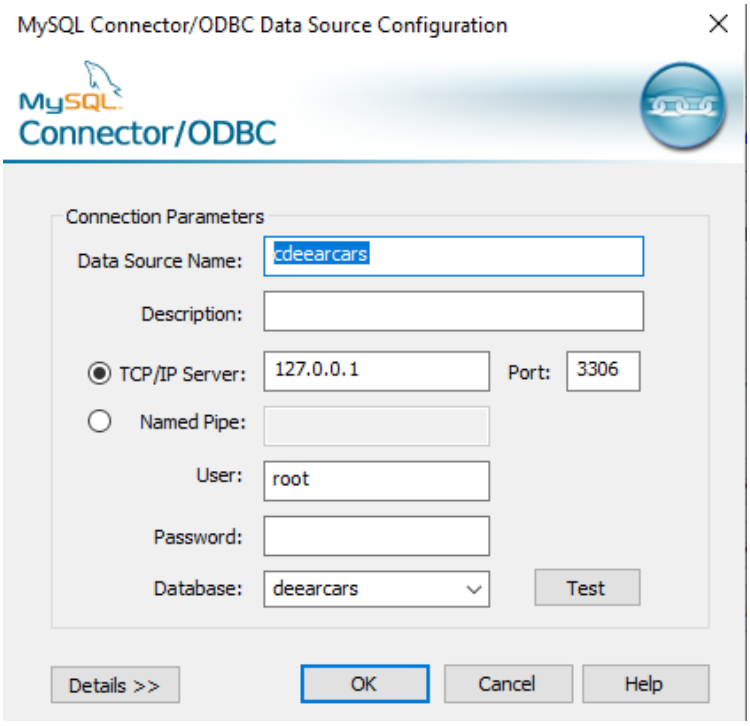


Figura 16: parametros de conexion

Para terminar y comprobar que el conector funciona correctamente crearemos un vinculo de tablas entre el MySQL y el Access 2013 a través del conector.

Para ello abrimos una nueva BBDD en Access y en “Datos Externos” seleccionamos Base de datos ODBC”.

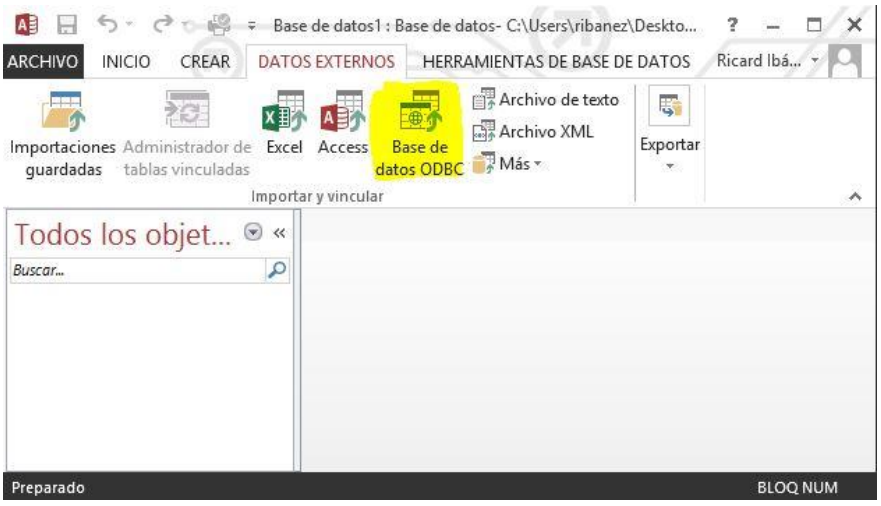
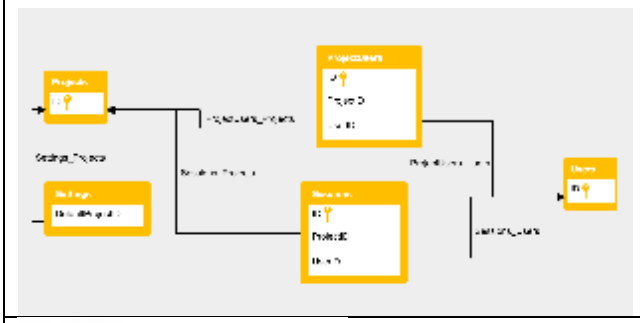


Figura 17: Conexión de aplicaciones conon base de datos

B. Base de Consulta

TÍTULO	AUTOR	EDICIÓN	AÑO	IDIOMA	EDITORIAL
Base de Datos	López Montalbán, Iván	1	2013	Español	México Alfaomega
Gestión de base de datos con SQL MySQL y Access Curso práctico	Orbegozo Arana, Borja	1	2013	Español	México Alfaomega
Database processing: fundamentals, design, and implementation	Kroenke, David M.	15	2018	Español	New Jersey Pearson Education
Database: design, application development, and administration	Mannino, Michael	6	2015	Ingles	Chicago Business Press.
Fundamentos de bases de datos	Silberschatz, Abraham	1	2014	Español	Madrid McGraw-Hill
PHP y MySQL: tecnología para el desarrollo de aplicaciones Web	Cobo, Ángel	1	2005	Español	Barcelona Díaz de Santos S.A
Domine PHP y MySQL	López Quijado, José	2	201	Ingles	México Alfaomega

C. Base práctica con ilustraciones

TRABAJOS A PRESENTAR	DETALLE
	**2.- Creación de un diagrama de base de datos** Se debe crear un modelo entidad relación ,utilizando el programa MYSQL Workbench.
	3- Presentación del Proyecto Impreso Debe contener carátula, encabezado y pie de página , Índice, inicio a normas APA, Aquí se describe a detalle los pasos para realizar el proyecto , se explican sentencias usadas

4. ESTRATEGIAS DE APRENDIZAJE

ESTRATEGIA DE APRENDIZAJE 1: Análisis y Planeación
Motivación : • Presentación de los objetivos del Curso. Experiencia vivencial motivadora. • Orientar al estudiante en el desarrollo de cada unidad del curso. • Diálogo profesor-alumnos sobre los contenido del tema a tratar. • Organiza de mejor manera a los grupos de trabajo. • Orientar a los estudiantes en el desarrollo del proyecto final de la asignatura **Materia:** • Discusión sobre las lecturas, artículos y videos. • Exposición del tema y ejemplos prácticos. • Mostrar los atributos del Explorador de Objetos del Gestor de Datos. Creación de una Base

de Datos mediante el modelador WorkBench usando el lenguaje de Definición de Datos del SQL.

- Laboratorio: Resolución de ejercicios utilizando el modelador de datos
- Realizar prácticas en clase guiadas, ejercicios de sentencias SQL.
- Ejercicios sobre operaciones de mantenimiento de base de datos: INSERT,UPDATE,DELETE
- Crear debates de participación.
- . Metodología activa. Participación activa de los alumnos en grupos de trabajo y de forma individual.
- Diseñar soluciones a problemas empleando el diseño concurrente.
- Crear y operar una base de datos, incluyendo un esquema de seguridad de acceso y roles
- Capacitación constante de Procesos de Backup y Restore

Ambiente(s) requerido:

Aula amplia con buena iluminación, y laboratorios.

Material (es) requerido:

- ✓ Aula de clase
- ✓ Aulas virtuales
- ✓ Bibliotecas, páginas web
- ✓ Videos a fines al tema impartido
- ✓ Proyector

Computador

Docente:

Con conocimiento de la materia.

5. ACTIVIDADES

- Controles de lectura
- Exposiciones
- Presentación del Trabajo final
- CD con contenido del Proyecto
- Habilidad y esfuerzo en el proyecto entregado

6. EVIDENCIAS Y EVALUACIÓN

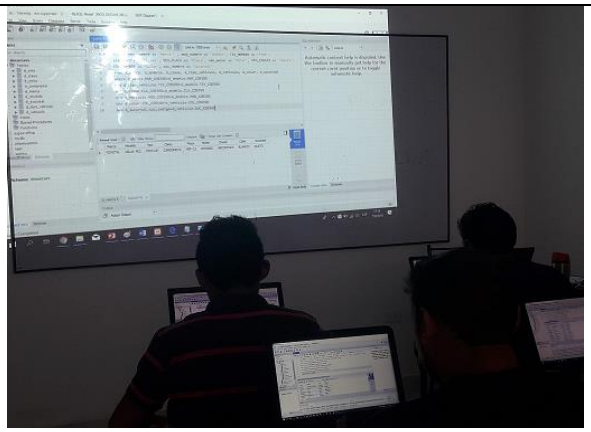
Tipo de Evidencia	Descripción (de la evidencia)
De conocimiento:	Creación de un proyecto aplicando conocimientos del diseño de base de datos, aplicando sentencias DLL y CRUD. Como resultado comprende que basado en las relaciones se persiguen mejores resultados en consulta de datos.
Desempeño:	Trabajo grupal presentación del trabajo sobre la creación de un modelo entidad relación , los estudiantes defienden su proyecto en una exposición , justificando su realización.
De Producto:	✓ Desarrollo de un proyecto innovador, que debe ser promocionado a los demás estudiantes , generando interés significativo y positivo. Intervención mediante una práctica de los estudiantes en consola usando PHPmyadmin. ✓ Exposiciones orales sobre los temas de investigación individuales asignados a los estudiantes, sobre hojas de cálculo
De Innovación	Se revisará la participación investigativa por parte del alumno en cuanto refiere a la innovación y desempeño al proyecto entregado el cual debe contener la mejor solución de un modelador de base de datos.
Criterios de Evaluación (Mínimo 5 Actividades por asignatura)	Se valida la presentación de proyectos Se valida desempeño y justificación de las sentencias ejecutadas Se valida desarrollo de proyecto en Word, investigación de sentencias. Se valida preguntas y resolución de sentencias de forma individual.

ANEXO 1 EVIDENCIAS DE APRENDIZAJE

Msc. Diana Moncayo	**Alexis Benavides**	**Milton Altamirano**
Elaborado por: **(Docente)**	**Revisado Por:** **(Coordinador)**	**Reportado Por:** **(Vicerrector)**

ANEXO 1

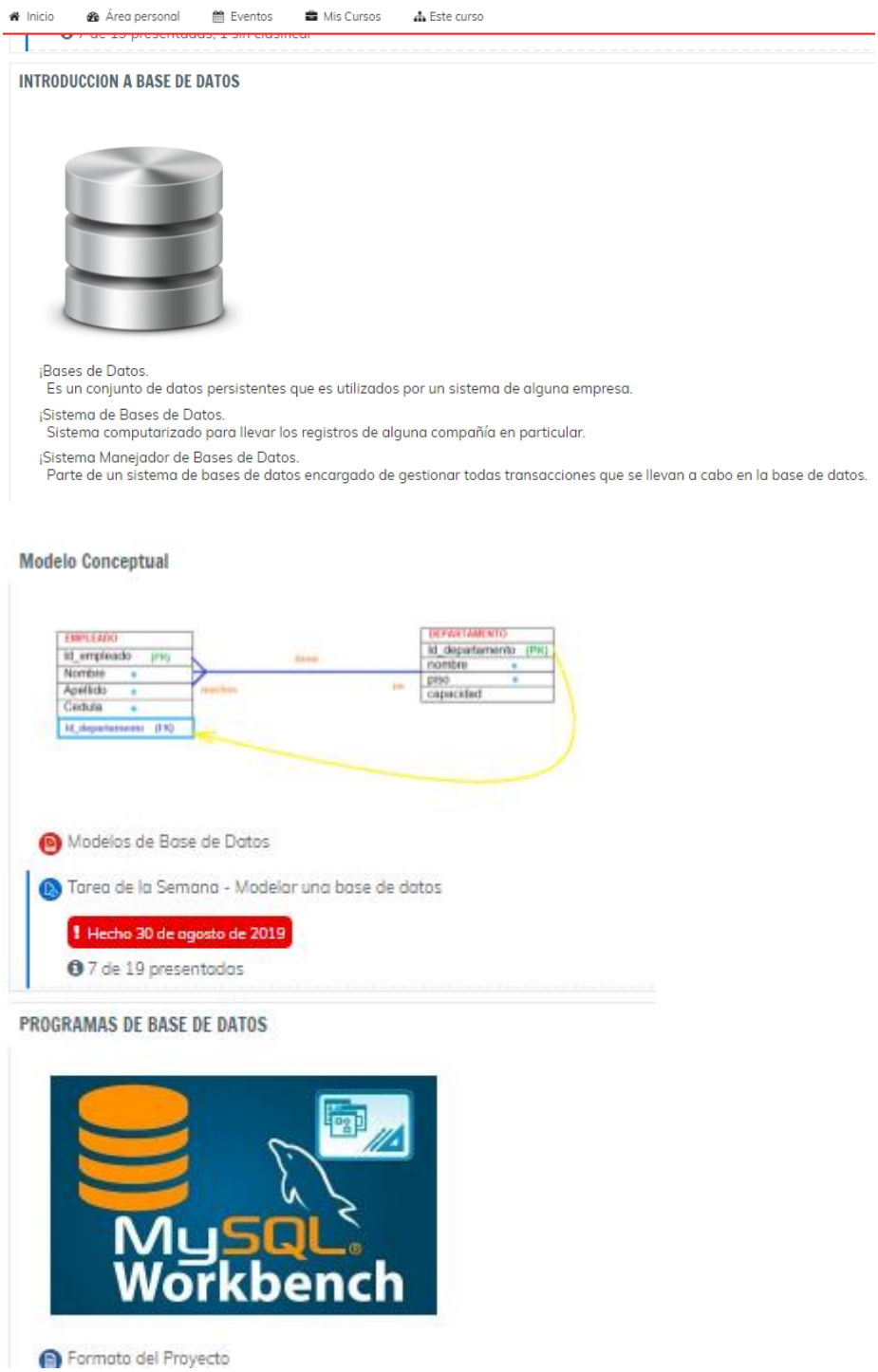
Prácticas en clase



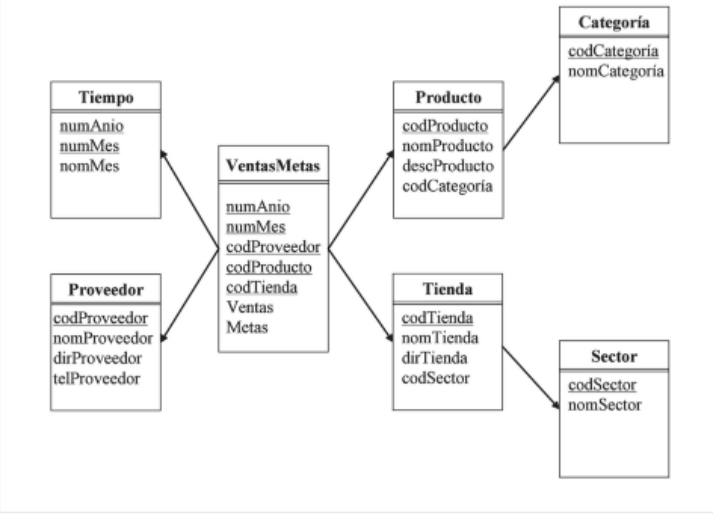
ANEXO 1

Uso de la plataforma

UNIDADES



Modelo Físico



CREATE , ALTER Y DROP TABLE Y DATABASE

Trabajos

SENTENCIAS SQL CRUD

Trabajo Semanal 4

Hecho 13 de septiembre de 2019

6 de 19 presentadas

Consultas SQl

Trabajo Semanal 5 Practica de Sentencias CRUD

Hecho 20 de septiembre de 2019

6 de 19 presentadas

PROCEDIMIENTOS ALMACENADOS

"Procedimiento"

```
DELIMITER //
    CREATE PROCEDURE ejercicio1(IN equipo VARCHAR(25), OUT ganancias INT)
        BEGIN
            SELECT SUM(m.premio)
            INTO ganancias
            FROM maillot m, ciclista c, lleva l
            WHERE m.codigo=l.codigo AND c.dorsal=l.dorsal AND c.nomequipo=equipo;
        END; //
DELIMITER;
```

"Llamada"

```
CALL ejercicio1("Banesto", @ganancias);
```

"Consulta"

```
SELECT @ganancias;
```

Corrección de la Prueba

Evaluaciones

Inicio | Área personal | Eventos | Mis Cursos | Este curso

horas

		LUIS XAVIER ANDRADE RAMOS	xavierblesth@hotmail.com	Enviado para calificar Calificado	Calificación 9,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 00:00	MOD_BIBLIOTECA.pdf MOD_DEEARCARS PAIS.pdf TRABAJO SEMANAL2 (1).pdf	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:30
		ALEJANDRO DAVID FLORES BARROS	adfbflores@gmail.com	Enviado para calificar Calificado	Calificación 4,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 00:10	TRABAJO 2 BIBLIOTECA.doc	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:31
		ALEX DARWIN GREFA YUMBO	elisebas_02@hotmail.com	Enviado para calificar Calificado	Calificación 8,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 12:56	DEBER BASE DATO2.doc	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:33
		WILSON RICARDO MUÑOZ QUISILEMA	wmunoz717@gmail.com	Enviado para calificar Calificado	Calificación 7,50 / 10,00	Editar	viernes, 30 de agosto de 2019, 18:55	BIBLIOTECA.mwb.pdf DECARDS.pdf	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:34
		ANTHONY ANDERSON TITUAÑA SIMBAÑA	Anthony_extr@hotmail.com	Enviado para calificar Calificado	Calificación 7,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 19:48	MOD_BLIBLIOTECA.pdf TRABAJO SEMANAL2.doc	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:38
		PABLO ESTEBAN CEVALLOS ORDOÑEZ	pblostebn_rachis@hotmail.com	Enviado para calificar Calificado	Calificación 7,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 20:29	DEBER2.pdf	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:39
		RICARDO XAVIER NOROÑA NIETO	rxnorona81@hotmail.com	Enviado para calificar Calificado	Calificación 9,00 / 10,00	Editar	viernes, 30 de agosto de 2019, 20:44	TRABAJO SEMANAL2.pdf	Comentarios (0)	jueves, 12 de septiembre de 2019, 12:40

INSTITUTO SUPERIOR TECNOLÓGICO

JAPÓN

Amor al conocimiento

Guía metodológica de base de datos
Carrera de desarrollo de software
Mgs. Diana Moncayo
2019

Coordinación editorial general:
Mgs. Milton Altamirano Pazmiño
Ing. Alexis Benavides Vinueza
Mgs. Lucía Begnini Dominguez

Diagramación: Sebastián Gallardo Ramírez
Corrección de estilo: Mgs. Lucía Begnini Dominguez
Diseño: Sebastián Gallardo Ramírez
Imprenta: JKIMPRIMA

Instituto Superior Tecnológico Japón
AMOR AL CONOCIMIENTO

ISBN: 978-9942-811-84-4