

Instituto Tecnológico de Villahermosa
Ing. Sistemas Computacionales
6to Semestre



Taller de Base de Datos

Reporte Unidad 1, 2 y 3

Reynaldo Bernard de Dios de la Cruz



Miércoles, 24 de
Febrero de 2016

Taller de Base de Datos

Contenido

Introducción	3
1 - Instalación y configuración del sistema gestor de bases de datos en distintas plataformas	4
Marco teórico	4
¿Qué es un sistema de gestión de base de datos (SGBD)?	4
Funciones de un SGBD	4
Características de un SGBD	4
Marco práctico	5
1.1 Requerimientos del SGBD	5
1.2 Instalación del SGBD	5
1.3 Configuración del SGB	7
Extra: Instalación y configuración de MySQLWorkbench	14
2 - Lenguaje de definición de datos(DDL)	20
Marco teórico	20
¿Qué es el DDL?	20
Marco práctico	20
2.1 Creación del esquema de la base de datos.....	20
2.2 Actualización, modificación y eliminación del esquema de la base de datos.....	21
Extra: Explicación del script SQL generado por MySQL Workbench	26
3 - Lenguaje de manipulación de datos(DML)	29
Marco teórico	29
¿Qué es DML?	29
Sentencias DML	29
Marco práctico	30
3.1 Inserción, eliminación y modificación de registros.....	30
3.2 Consultas de registros	31
Conclusión	37
Anexo	38
Bibliografía.....	38

Taller de Base de Datos

Introducción

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. A continuación te presentamos una guía que te explicará el concepto y características de las bases de datos. Entre las principales características de los sistemas de base de datos podemos mencionar: Independencia lógica y física de los datos, redundancia mínima, acceso concurrente por parte de múltiples usuarios, integridad de los datos, consultas complejas optimizadas, seguridad de acceso y auditoría, respaldo y recuperación, acceso a través de lenguajes de programación estándar.

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

En este trabajo se presentará una breve serie de pasos para lograr desde crear una base de datos, hasta ingresar datos, todo esto con ayuda de herramientas gráficas y rápidas que permitirán a la persona lograr un desarrollo eficiente y rápido de el proyecto.

Se encuentra dividido en tres temas:

- Tema 1 - Instalación y configuración del sistema gestor de bases de datos en distintas plataformas
- Tema 2 - Lenguaje de definición de datos(DDL)
- Tema 3 - Lenguaje de manipulación de datos(DM)

Cada uno de los temas mencionados anteriormente contará con su parte teórica y práctica, en este caso el llevado a cabo durante el curso de Taller de Base de Datos y el SGBD a utilizar será MySQL de Oracle.

Taller de Base de Datos

1 - Instalación y configuración del sistema gestor de bases de datos en distintas plataformas

Marco teórico

¿Qué es un sistema de gestión de base de datos (SGBD)?

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permiten definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

Funciones de un SGBD

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Características de un SGBD

Abstracción de la información. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Taller de Base de Datos

Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Marco práctico

1.1 Requerimientos del SGBD

- 512 Mb de memoria RAM
- 1024 Mb maquina virtual
- 1 GB de espacio de disco duro
- Sistema operativo: Windows, Linux y Unix
- Arquitectura del sistema 32/64 bit
- Protocolo de red TCP/IP

1.2 Instalación del SGBD

Instalaremos MySQL sobre un equipo con Windows. La instalación en otras plataformas es igual de sencilla, pero no será tratada en este momento.

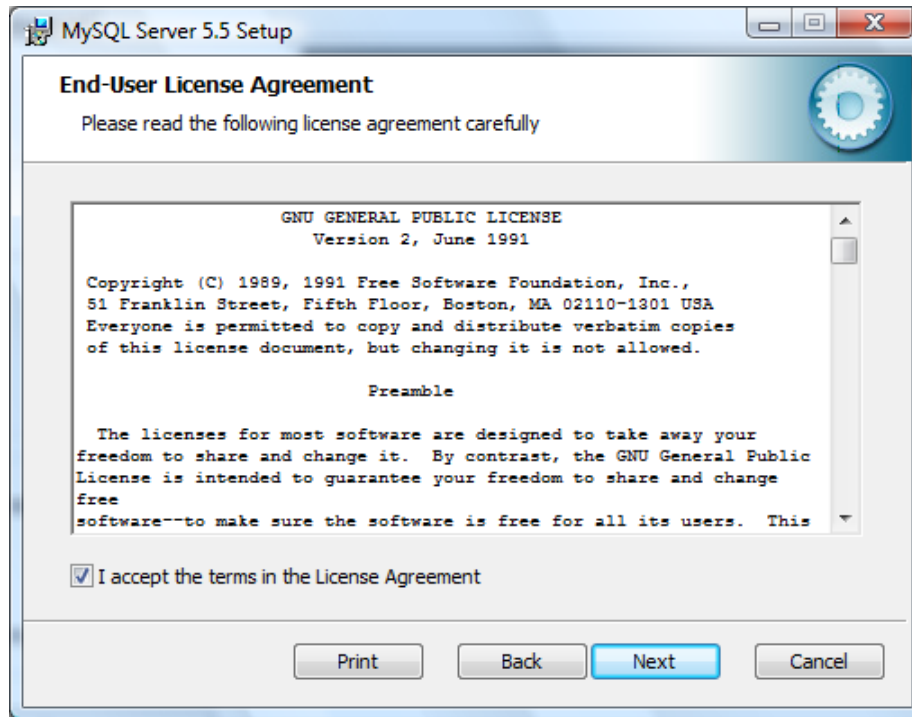
Para descargarte MySQL debes dirigirte a la sección de descargas de la [página oficial](#) y elegir MySQL Community Server, que es la versión gratuita del producto. Selecciona Windows como plataforma y elige el instalador MSI que mejor se adapte a tu sistema operativo (32 o 64 bits).

El proceso de instalación es muy simple y prácticamente no requiere intervención por parte del usuario.

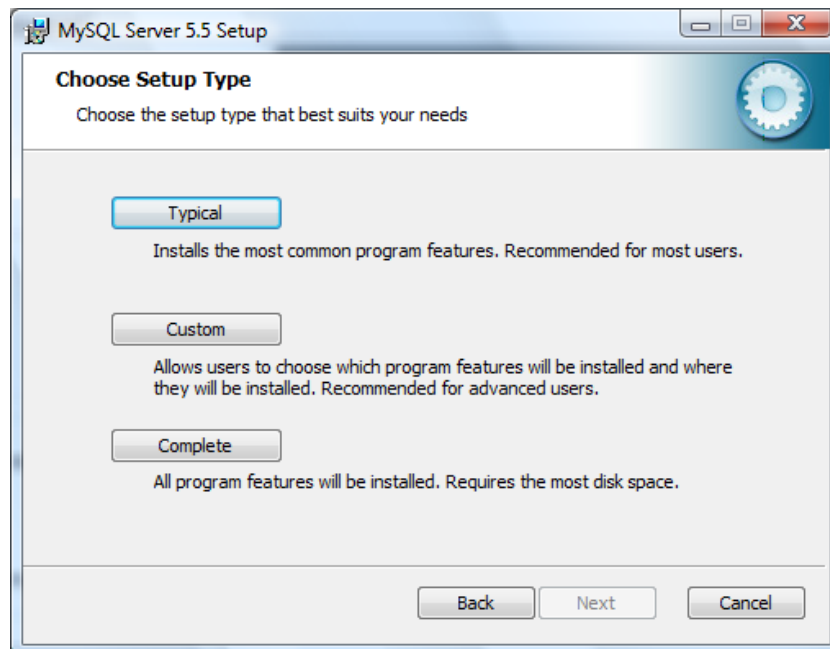


Taller de Base de Datos

Comienza el proceso; sólo nos llevará un par de minutos...

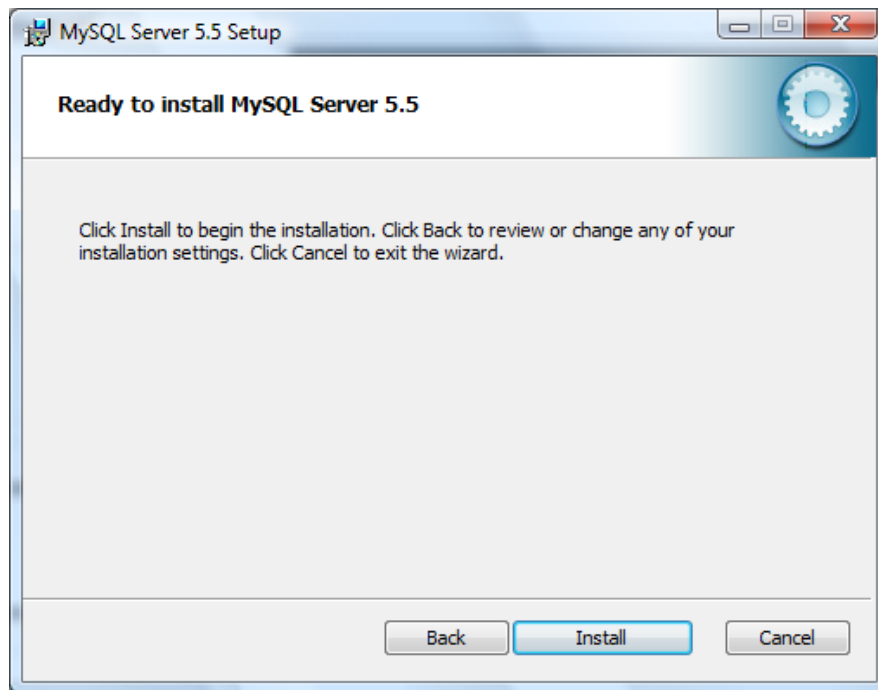


Cada vez que se muestra la pantalla de la GNU GPL nos da entender que el proyecto es de calidad. No sólo por las condiciones y el precio: es además, una garantía de profesionalidad.



Estadísticamente, la instalación típica será la que mejor se adapte a las necesidades.

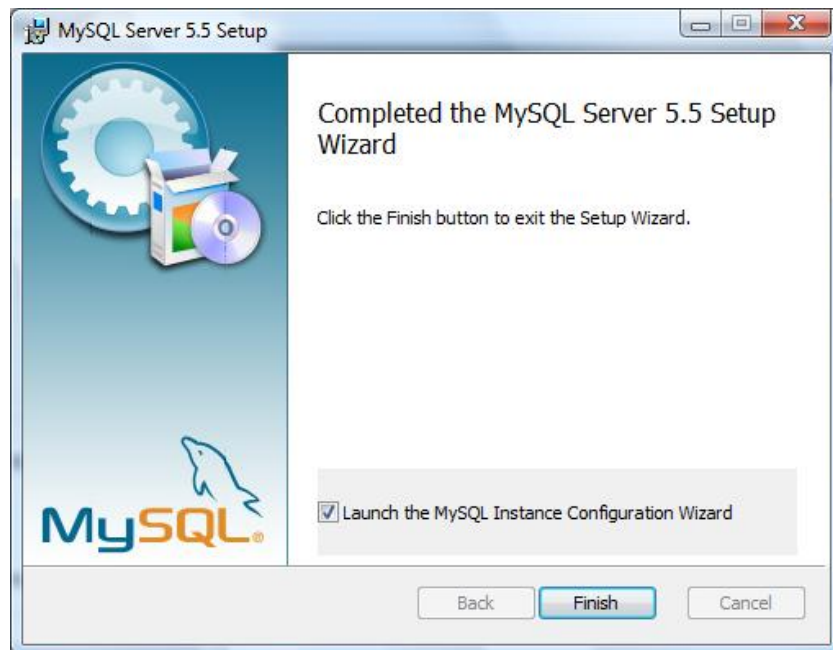
Taller de Base de Datos



Clic en el botón Install y hay que esperar a que termine de instalarse.

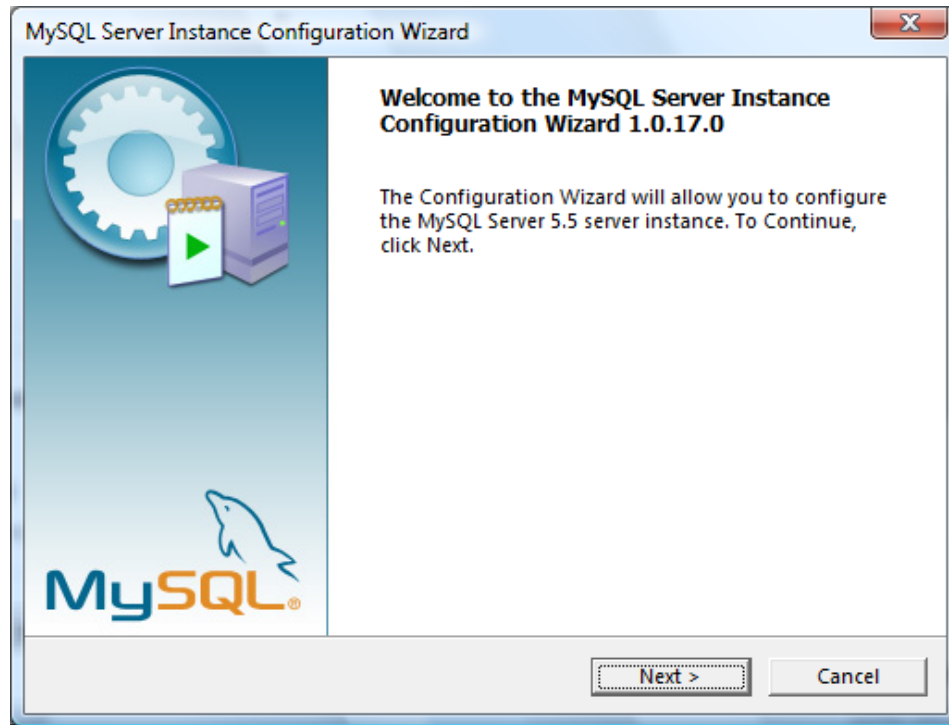
1.3 Configuración del SGB

Una vez instalado MySQL, la siguiente fase es la configuración del servidor en sí mismo. Hay que asegurarse de que la marca Launch the MySQL Instance Configuration Wizard esté activa.

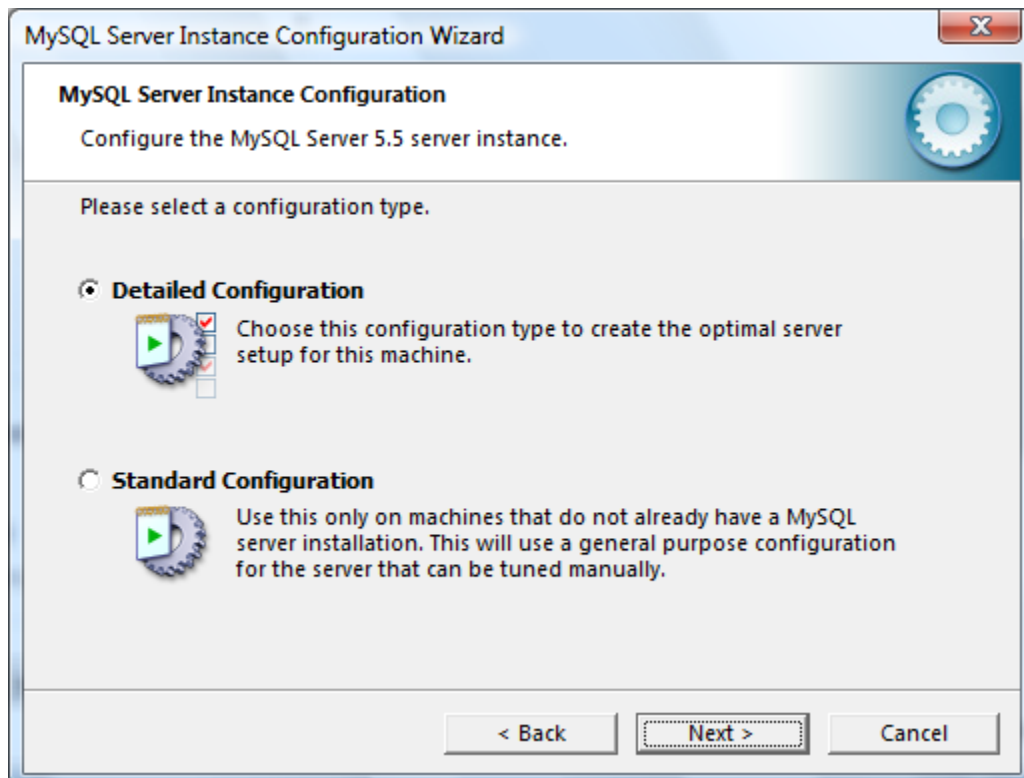


Taller de Base de Datos

Al hacer clic en Finish se abrirá una nueva ventana que permitirá configurar el SGBD. Hacer clic en Next.

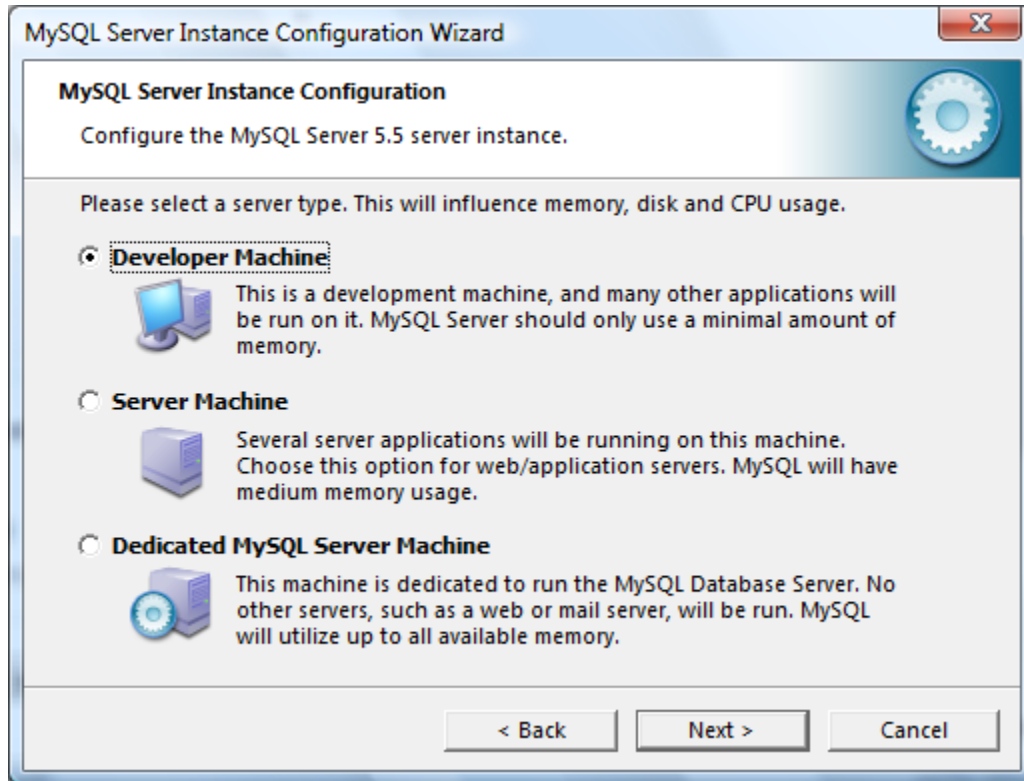


Optamos por Detailed Configuration, de modo que se optimice la configuración del servidor MySQL.

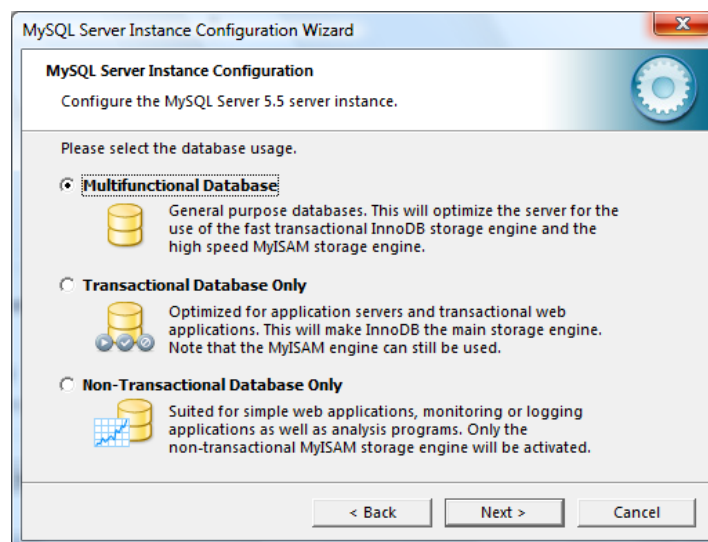


Taller de Base de Datos

Ha llegado un momento crucial. Dependiendo del uso que vayamos a darle a nuestro servidor deberemos elegir una opción u otra, cada una con sus propios requerimientos de memoria. Puede que te guste la opción Developer Machine, para desarrolladores, la más apta para un uso de propósito general y la que menos recursos consume. Si vas a compartir servicios en esta máquina, probablemente Server Machine sea tu elección o, si vas a dedicarla exclusivamente como servidor SQL, puedes optar por Dedicated MySQL Server Machine, pues no te importará asignar la totalidad de los recursos a esta función.

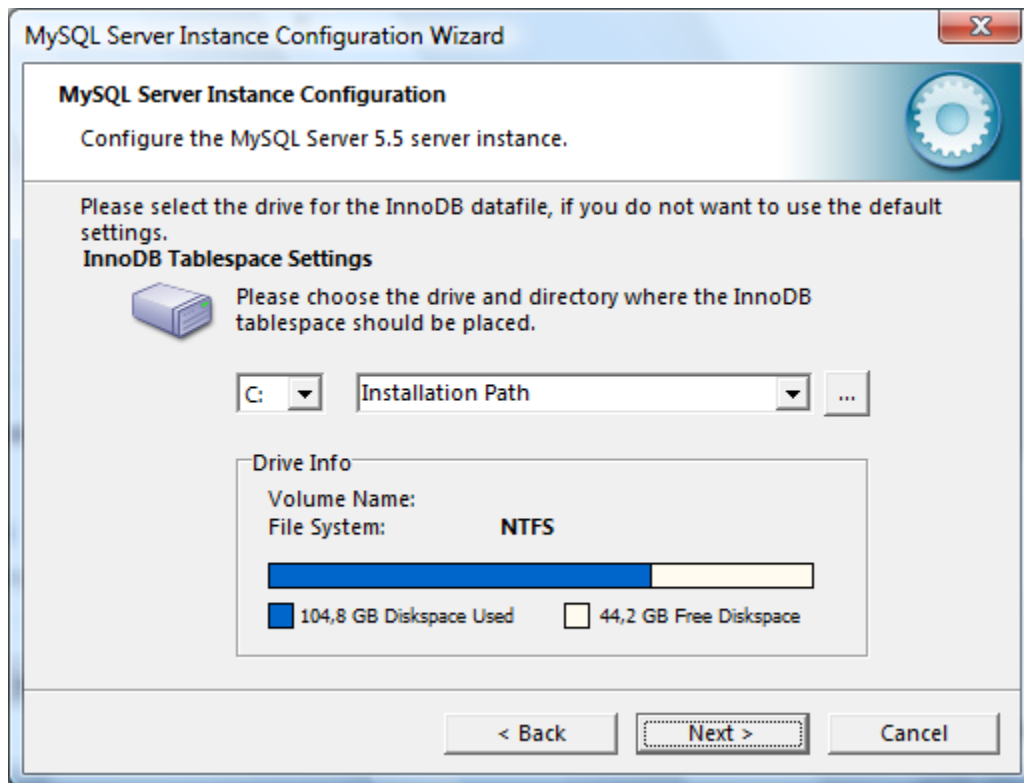


De nuevo, para un uso de propósito general, te recomiendo la opción por defecto, Multifunctional Database.

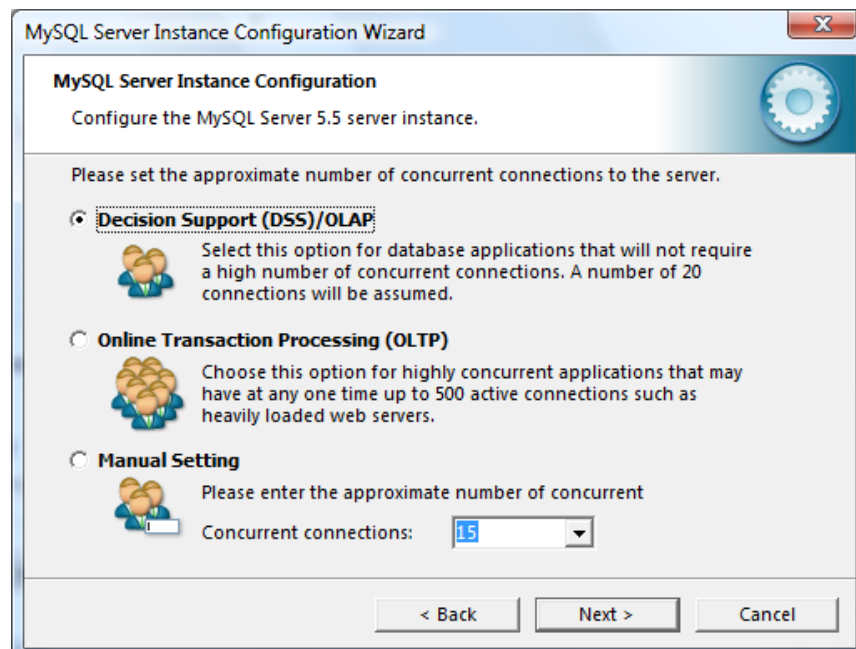


Taller de Base de Datos

InnoDB es el motor subyacente que dota de toda la potencia y seguridad a MySQL. Su funcionamiento requiere de unas tablas e índices cuya ubicación puedes configurar. Sin causas de fuerza mayor, acepta la opción por defecto.

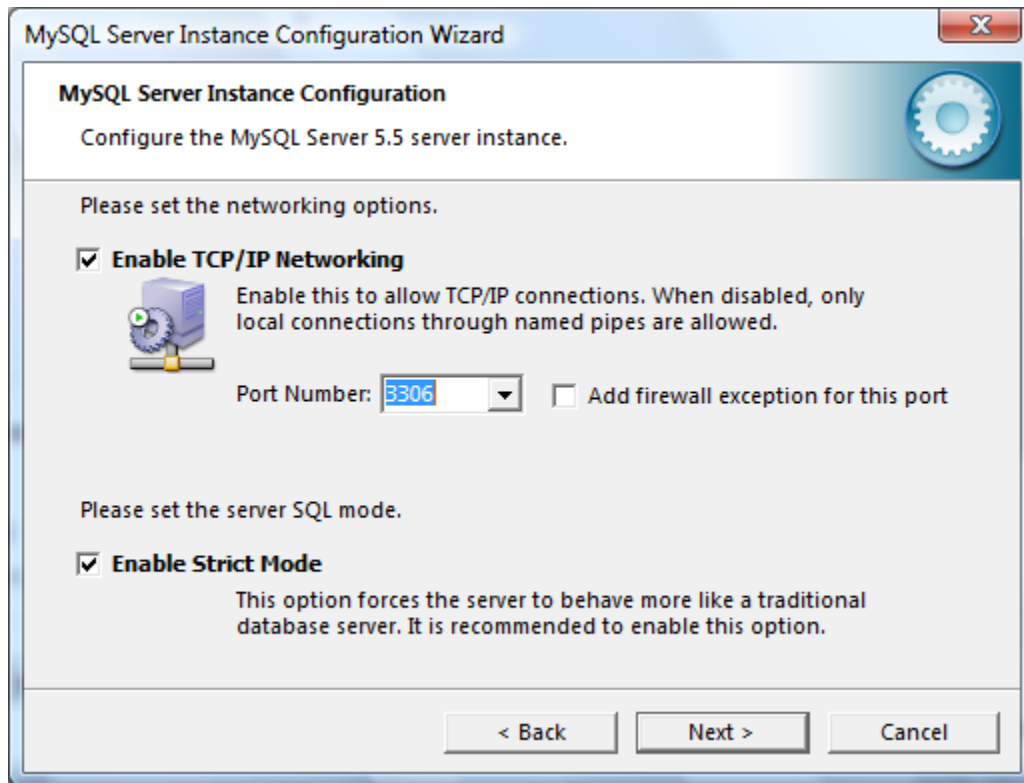


Esta pantalla nos permite optimizar el funcionamiento del servidor en previsión del número de usos concurrentes. La opción por defecto, Decision Support (DSS) / OLAP será probablemente la que más te convenga.

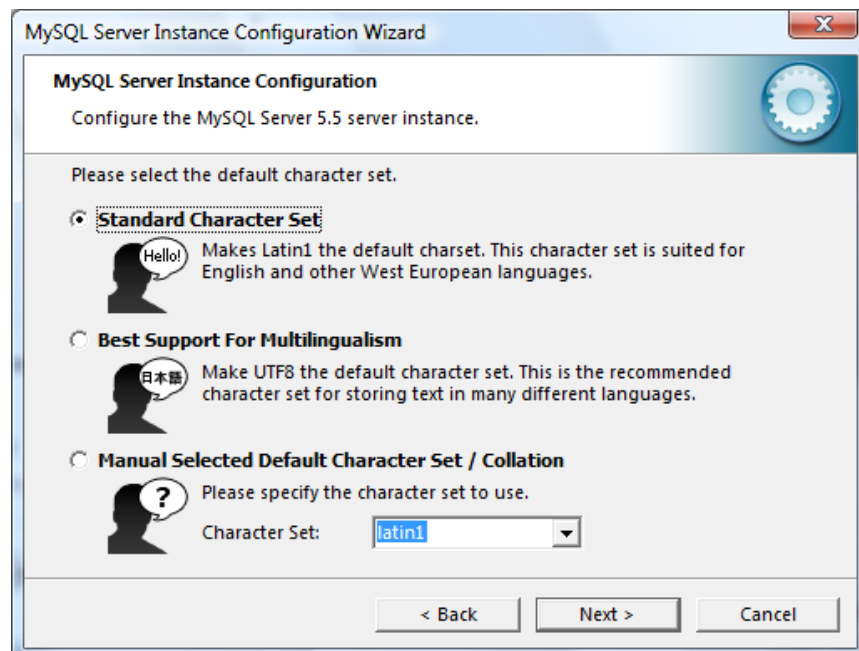


Taller de Base de Datos

Deja ambas opciones marcadas, tal como vienen por defecto. Es la más adecuada para un uso de propósito general o de aprendizaje, tanto si eres desarrollador como no. Aceptar conexiones TCP te permitirá conectarte al servidor desde otras máquinas (o desde la misma simulando un acceso web típico).



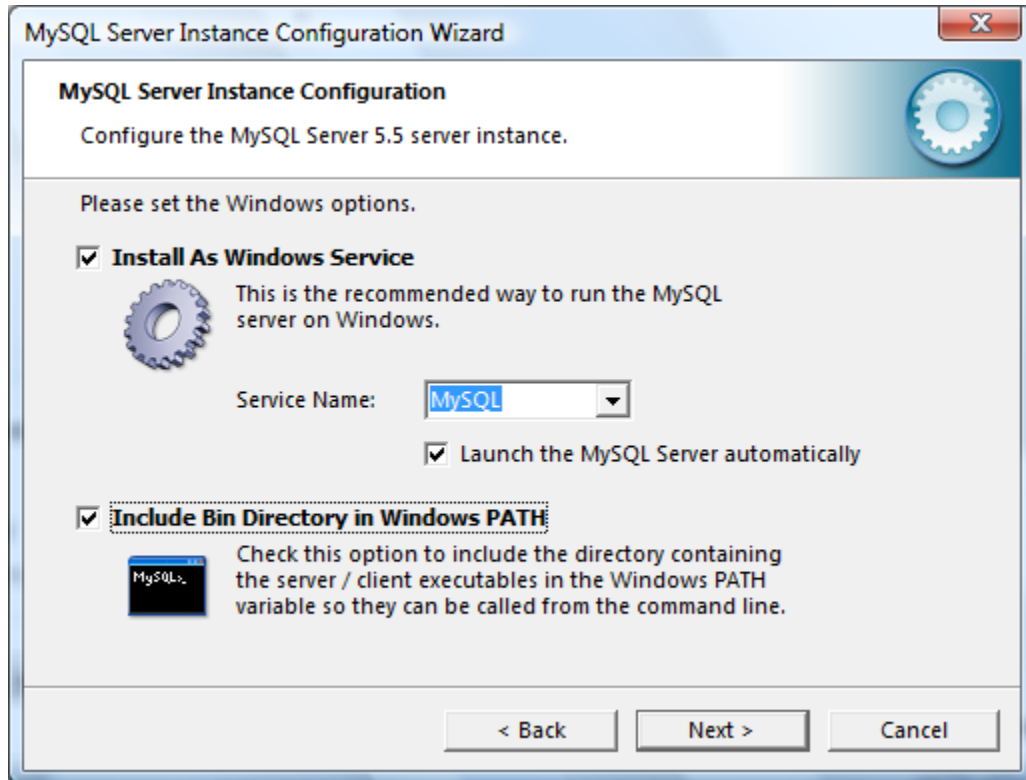
Hora de decidir qué codificación de caracteres emplearás. Salvo que quieras trabajar con Unicode porque necesites soporte multilinguaje, probablemente Latin1 te sirva (opción por defecto).



Taller de Base de Datos

Instalamos MySQL como un servicio de Windows (la opción más limpia) y lo marcamos para que el motor de la base de datos arranque por defecto y esté siempre a nuestra disposición. La alternativa es hacer esto manualmente.

Además, hay que asegurarse de marcar que los ejecutables estén en la variable PATH, para poder invocar a MySQL desde cualquier lugar en la línea de comandos



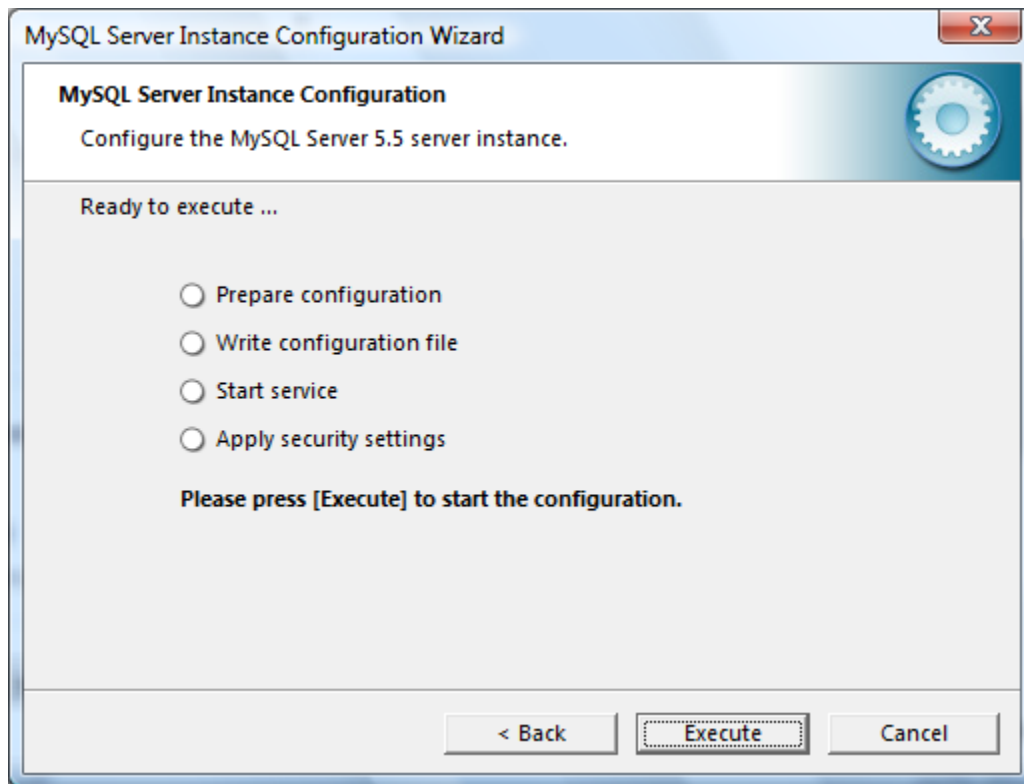
Pon una contraseña al usuario root. Esto siempre es lo más seguro.

Si lo deseas, puedes indicar que el usuario root pueda acceder desde una máquina diferente a esta, aunque debo advertirte de que eso tal vez no sea una buena práctica de seguridad.

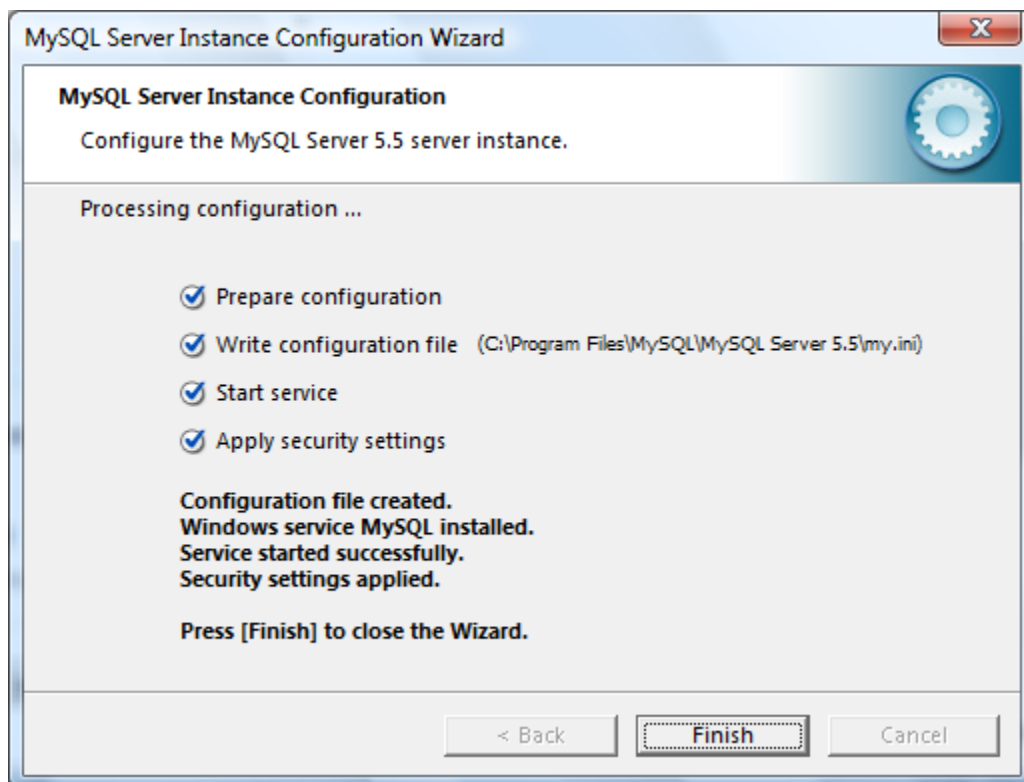


Taller de Base de Datos

Última etapa: listos para generar el fichero de configuración y arrancar el servicio.



Se acabó... ¿Te esperabas algo más?



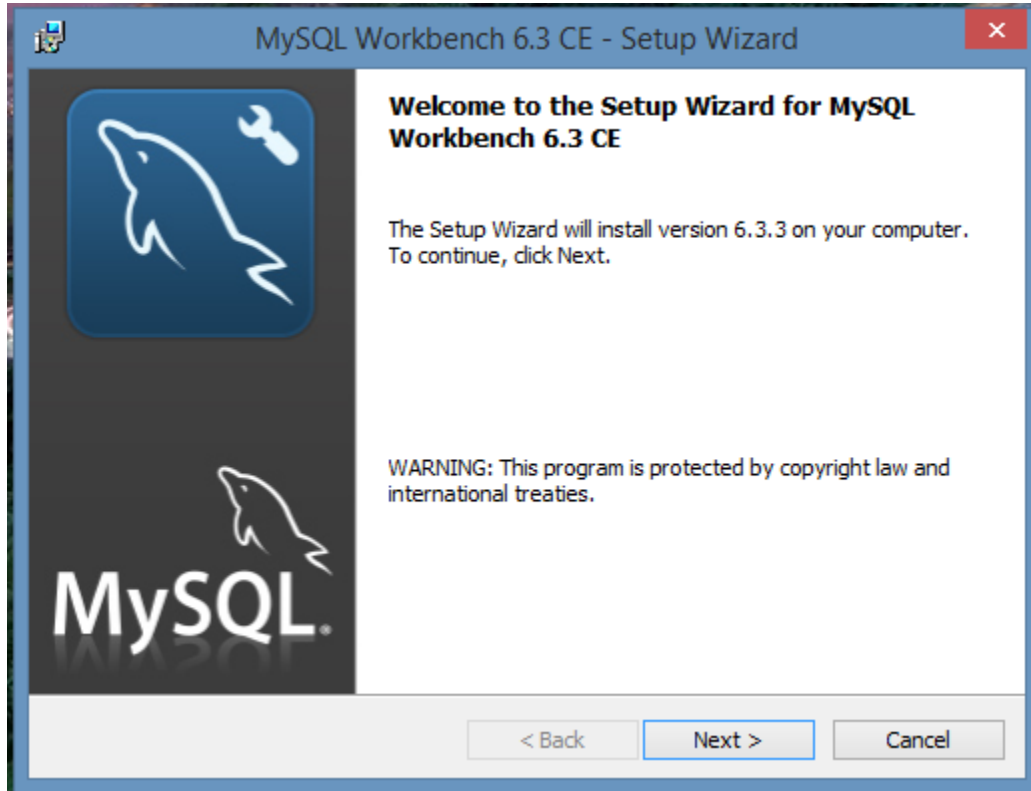
Taller de Base de Datos

Extra: Instalación y configuración de MySQLWorkbench

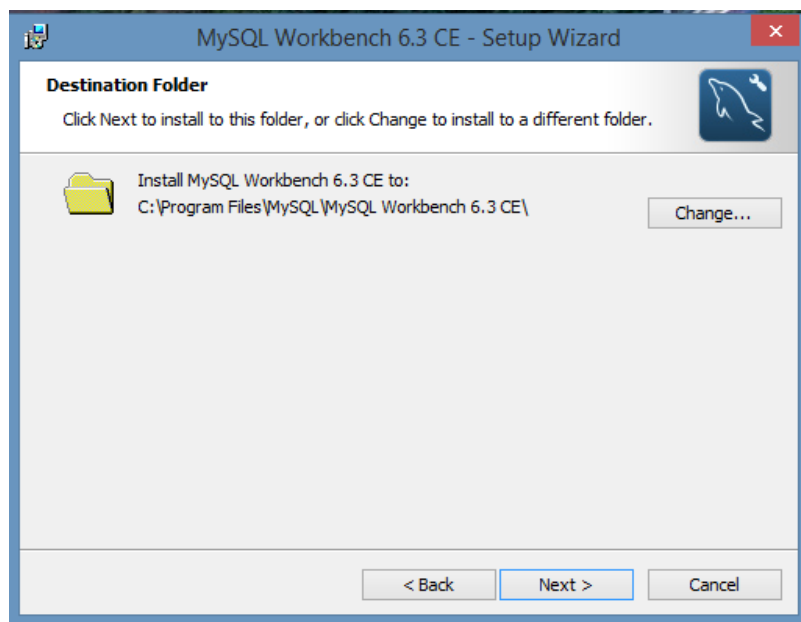
Ya se instaló MySQL y se configuró una instancia para su uso.

Ahora instalaremos la interfaz grafica para MySQL que usaremos para manejar bases de datos en MySQL recuerda que debes tener Windows, Windows XP SP3 como mínimo para poder instalar Workbench en nuestra computadora.

Primero que nada ejecutamos el archivo de instalación de Mysql WorkBench que descargamos desde la [página oficial de workbench](#). Clic en Next.

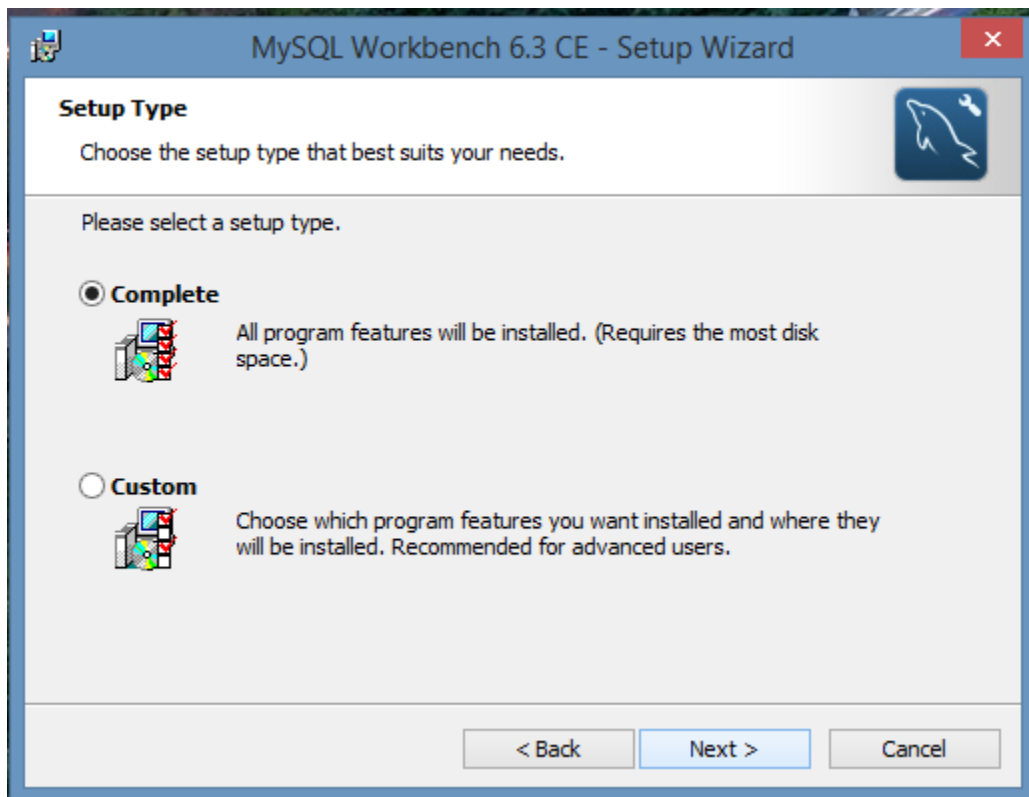


Seleccionar la ruta en donde se instalará MySQL Workbench (clic en change) y luego clic en Next

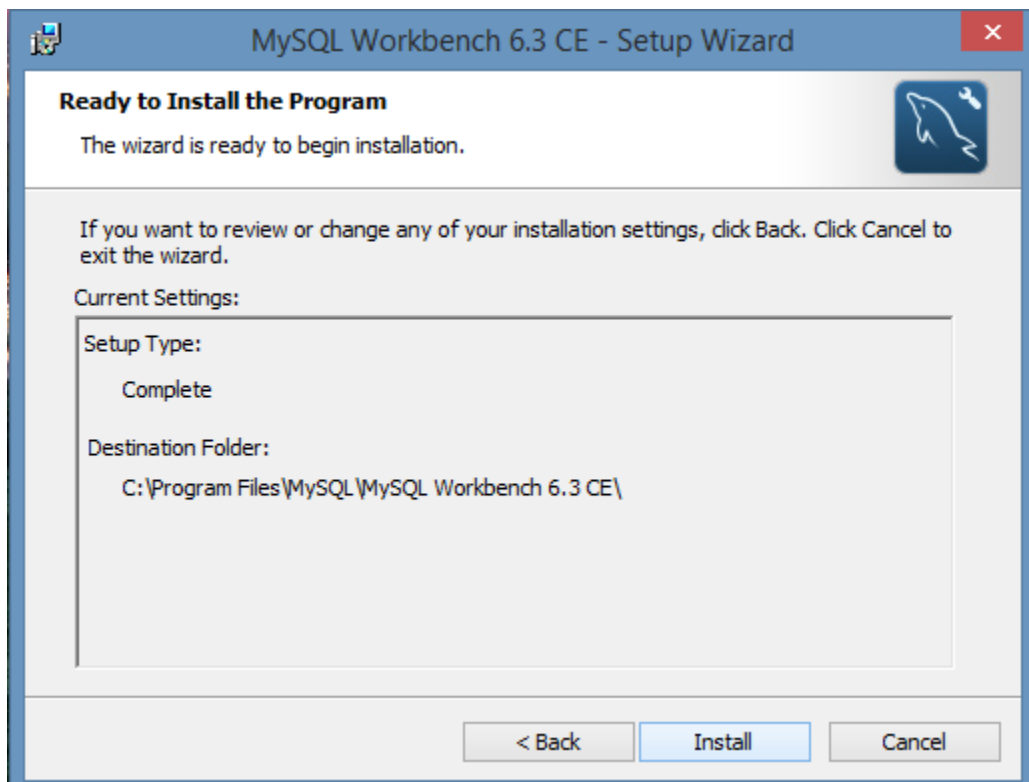


Taller de Base de Datos

Seleccionar instalación completa, es decir, la opción Complete y posteriormente clic en Next.

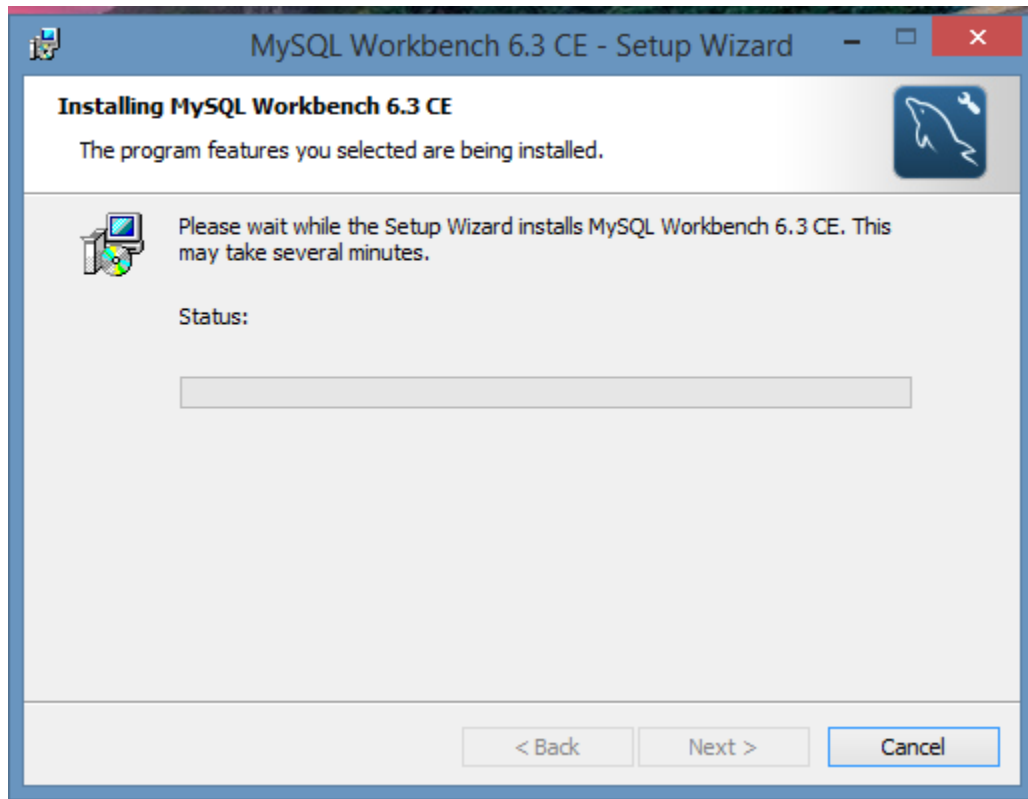


Aparecerá entonces un resumen de la instalación. Clic en Install para iniciar la instalación.

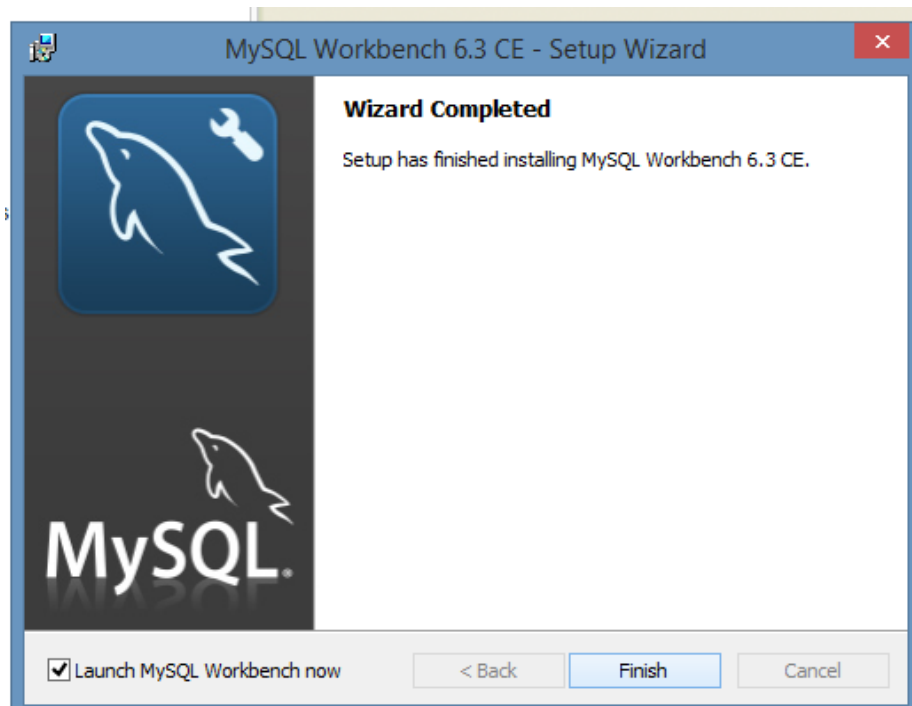


Taller de Base de Datos

Hay que esperar hasta que se instale el programa, puede realizar cualquier otra actividad mientras.

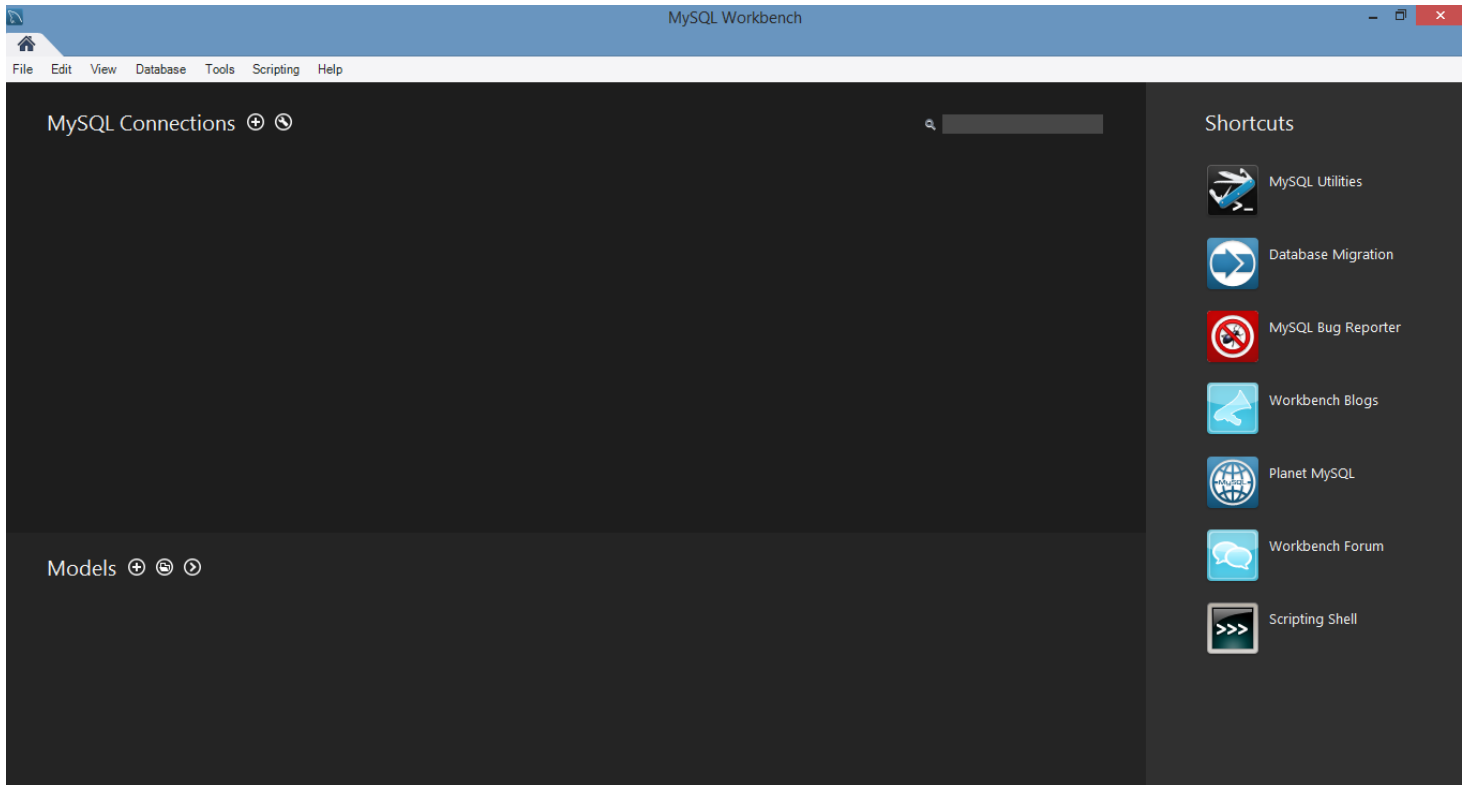


En caso de que el programa solicite permisos para poder instalarse, permítale hacerlo. Usualmente sale una ventana con las opciones Si y No, elija S y espere a que el programa termine de instalarse. Al terminar presione el botón Finish y se abrir el programa.

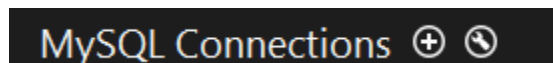


Taller de Base de Datos

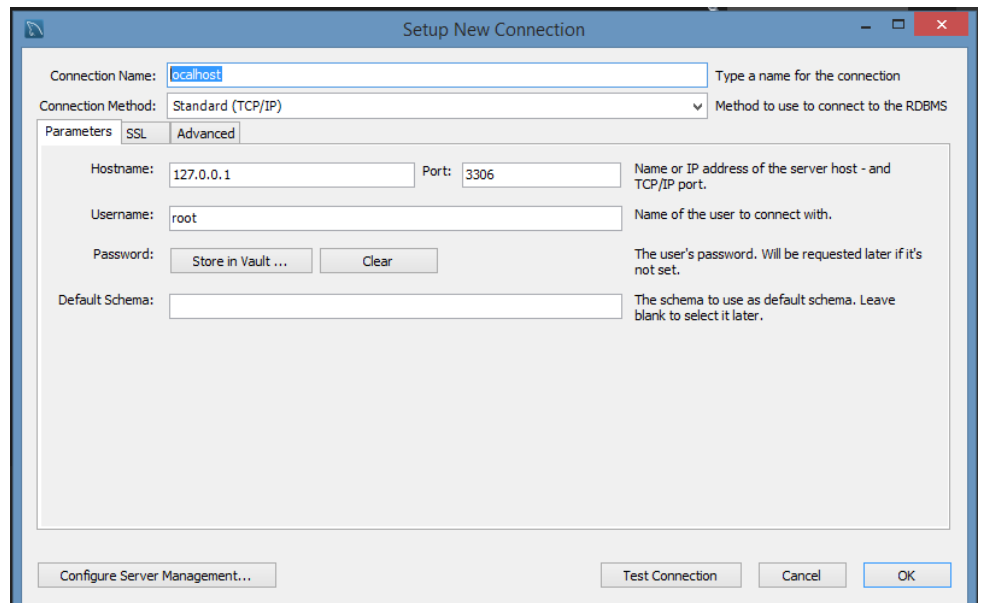
Se verá a continuación la siguiente pantalla.



Posteriormente crearemos la conexión con MySQL para poder trabajar, para ello hay que dar clic en el icono de más (+) en MySQL Connections.



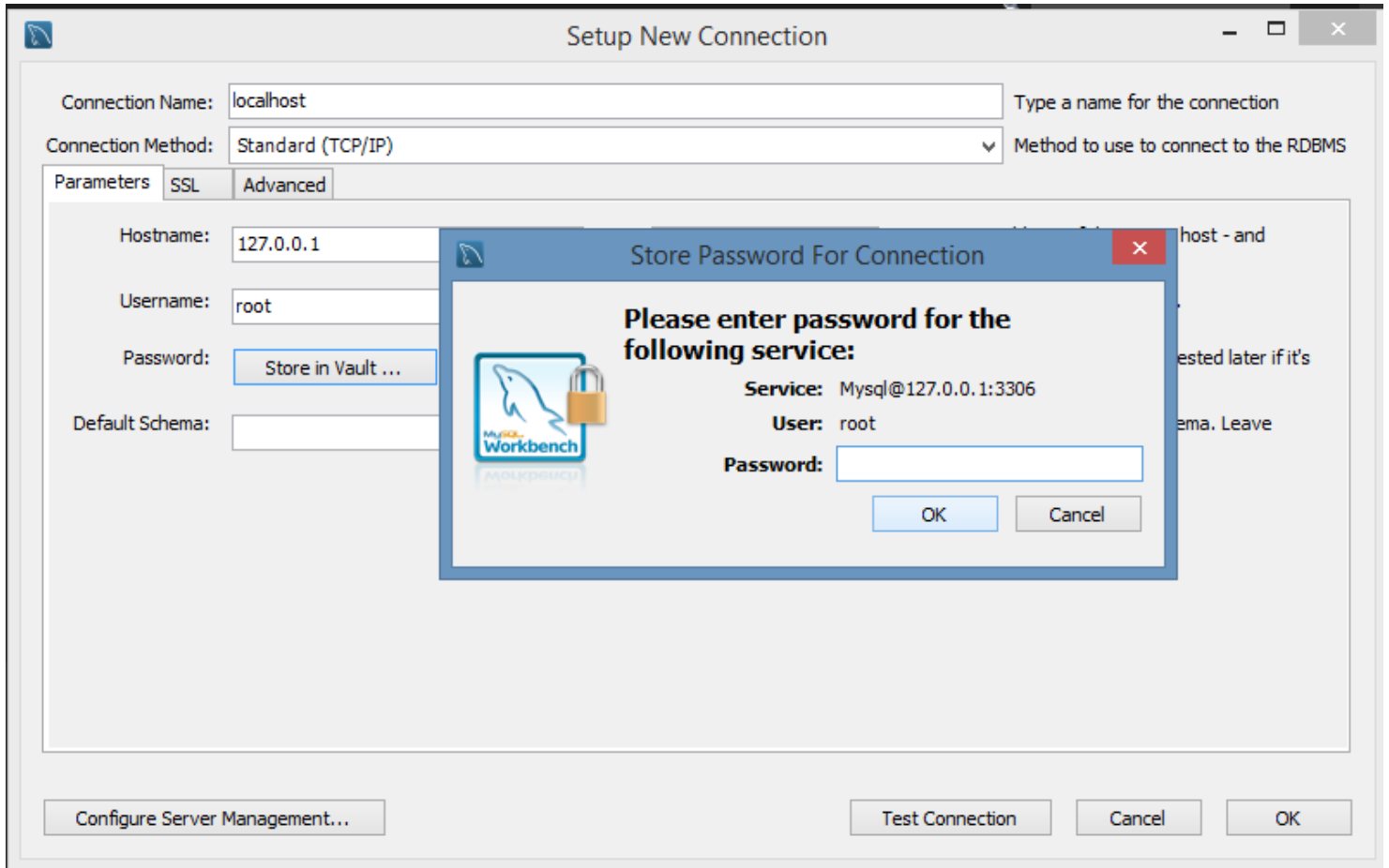
Se abrirá una nueva ventana y escribiremos "localhost" en el nombre de conexión (Connection Name), aunque puede llamarse como uno quiera, para este ejemplo se pone así ya que en realidad solo que conecta en la máquina local. Solo hay que cambiar el usuario y poner su respectiva contraseña.



Taller de Base de Datos

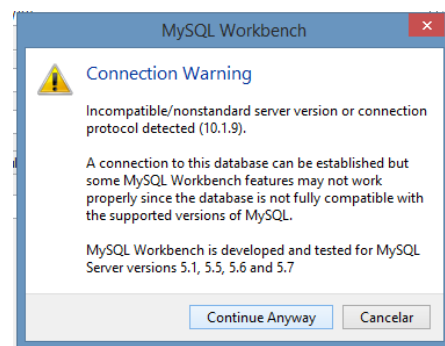
El usuario por defecto es *root*, de hecho hay que recordar que durante la instalación de MySQL Server se creó el usuario *root* y se le asignó contraseña.

Para establecer la contraseña de la conexión pasaremos a dar clic en el botón Store in Vault y saldrá una nueva ventana, donde hay que poner la contraseña que se ha asignado anteriormente en la instalación de MySQL Server.

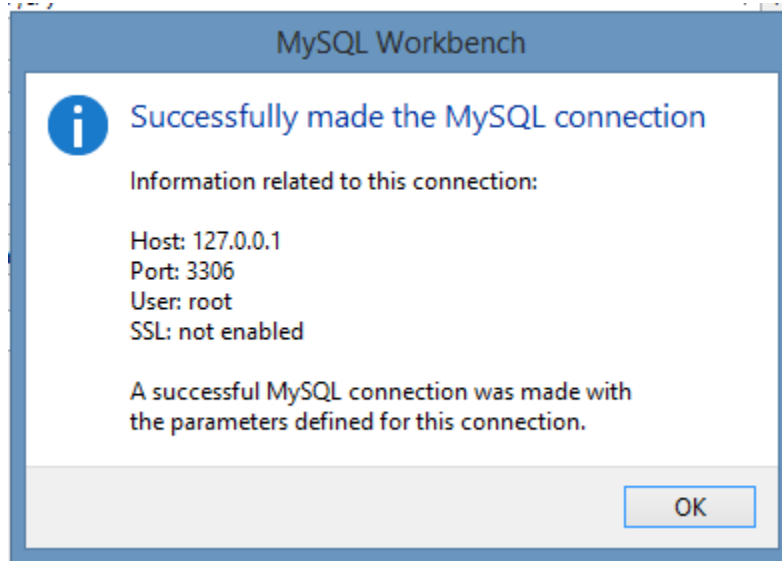


Una vez ingresada la contraseña hay que dar clic en el botón OK. Para probar la conexión daremos clic en el botón Test Connection y si el usuario y contraseña están correctos permitirá la conexión de forma exitosa. Hay que tener en mente las diferentes versiones de MySQL Server y MySQL Workbench, ya que algunos no son compatibles y si ese es el caso, algunas funciones de MySQL Workbench no podrían funcionar bien. Si ese es el caso aparecerá un mensaje de advertencia.

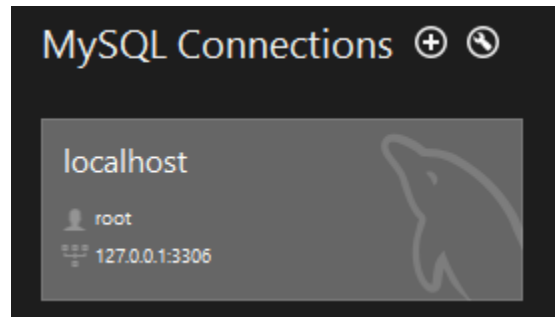
Aún así se puede usar MySQL Workbench pero, otra vez, algunas opciones no estarán disponibles. Eso no significa que el usuario y la contraseña sean incorrectos. Si el mensaje aparece solo daremos clic en el botón Continue Anyway (para ignorar la advertencia) y veremos el siguiente mensaje.



Taller de Base de Datos



Daremos clic en OK para cerrar la ventana y posteriormente en OK de la ventana de configuraciones de conexiones, entonces inmediatamente aparecerá la conexión en MySQL Workbench de la siguiente manera.



Listo, ahora tenemos lo necesario para trabajar.

Taller de Base de Datos

2 - Lenguaje de definición de datos(DDL)

Marco teórico

¿Qué es el DDL?

El DDL (Data Definition Language, o Data Description Language según autores), es la parte del SQL dedicada a la definición de la base de datos, consta de sentencias para definir la estructura de la base de datos, permite definir gran parte del nivel interno de la base de datos. Por este motivo estas sentencias serán utilizadas normalmente por el administrador de la base de datos.

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...).

Los objetos que veremos en este tema son:

- Bases de datos
- Tablas
- Vistas
- Índices

Como ya hemos comentado, las sentencias DDL están más orientadas al administrador de la base de datos, es el que más las va a utilizar, el programador tiene que conocer cuestiones relativas a la estructura interna de una base de datos, pero no tiene que ser experto en ello por lo que el estudio del tema se centrará en las sentencias y sobre todo en las cláusulas que pensamos pueden ser útiles a un programador y no entraremos en mucho detalle en cuanto a la estructura física de la base de datos y en la administración de la misma.

Marco práctico

2.1 Creación del esquema de la base de datos

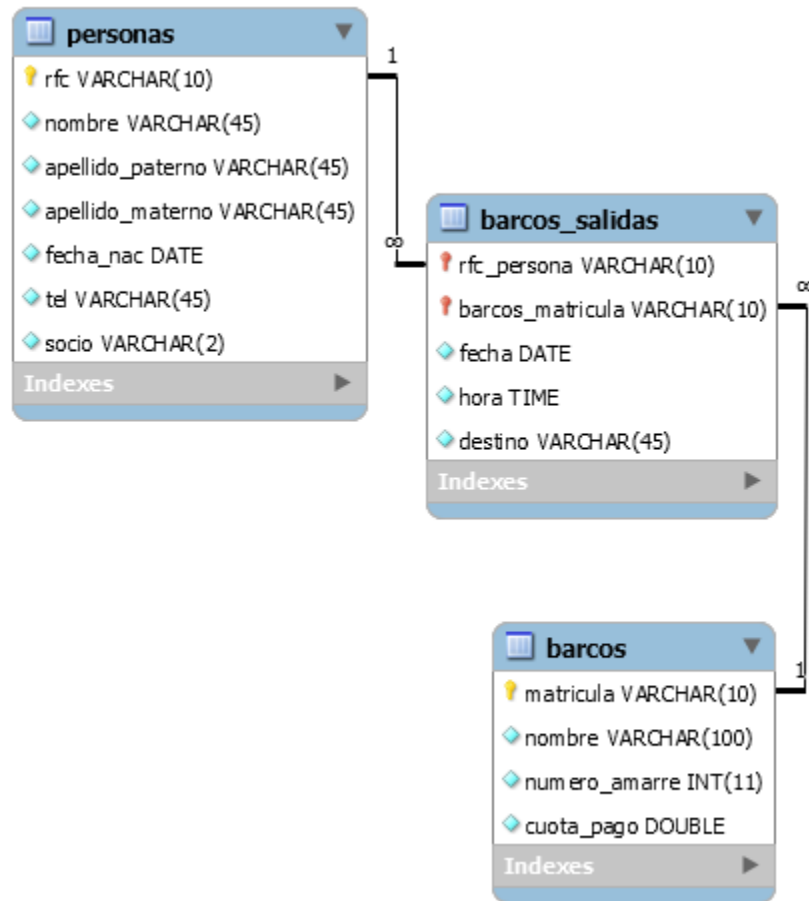
Empezaremos con presentar el problema planteado y la posible solución se dará en breve.

Problema:

Se quiere diseñar una base de datos relacional para gestionar los datos de los socios de un club náutico. De cada socio se guardan los datos personales y los datos del barco o barcos que posee: número de matrícula, nombre, número del amarre y cuota que paga por el mismo. Además, se quiere mantener información sobre las salidas realizadas por cada barco, como la fecha y hora de salida, el destino y los datos personales del patrón, que no tiene porque ser el propietario del barco, ni es necesario que sea socio del club.

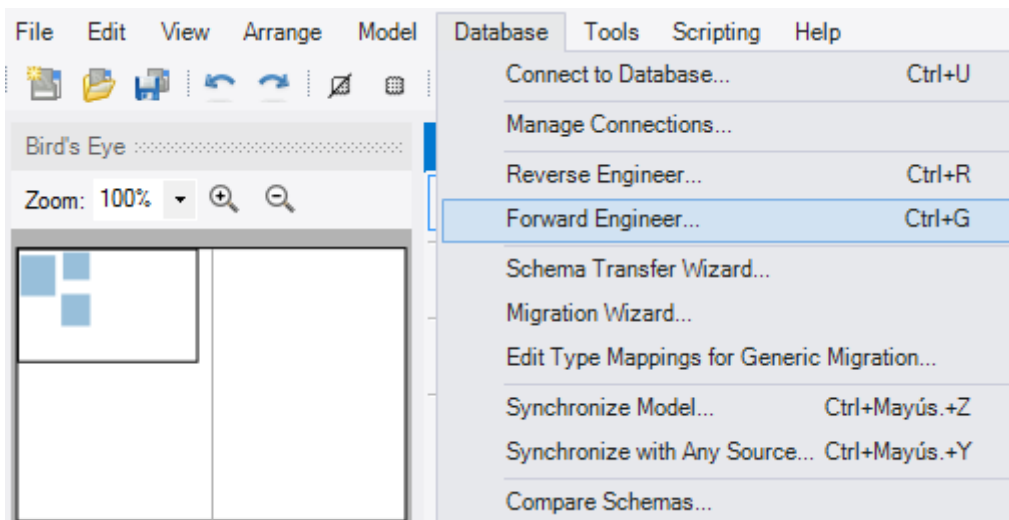
Taller de Base de Datos

Para resolver este problema pasaremos creando un MER (Modelo Entidad - Relación)



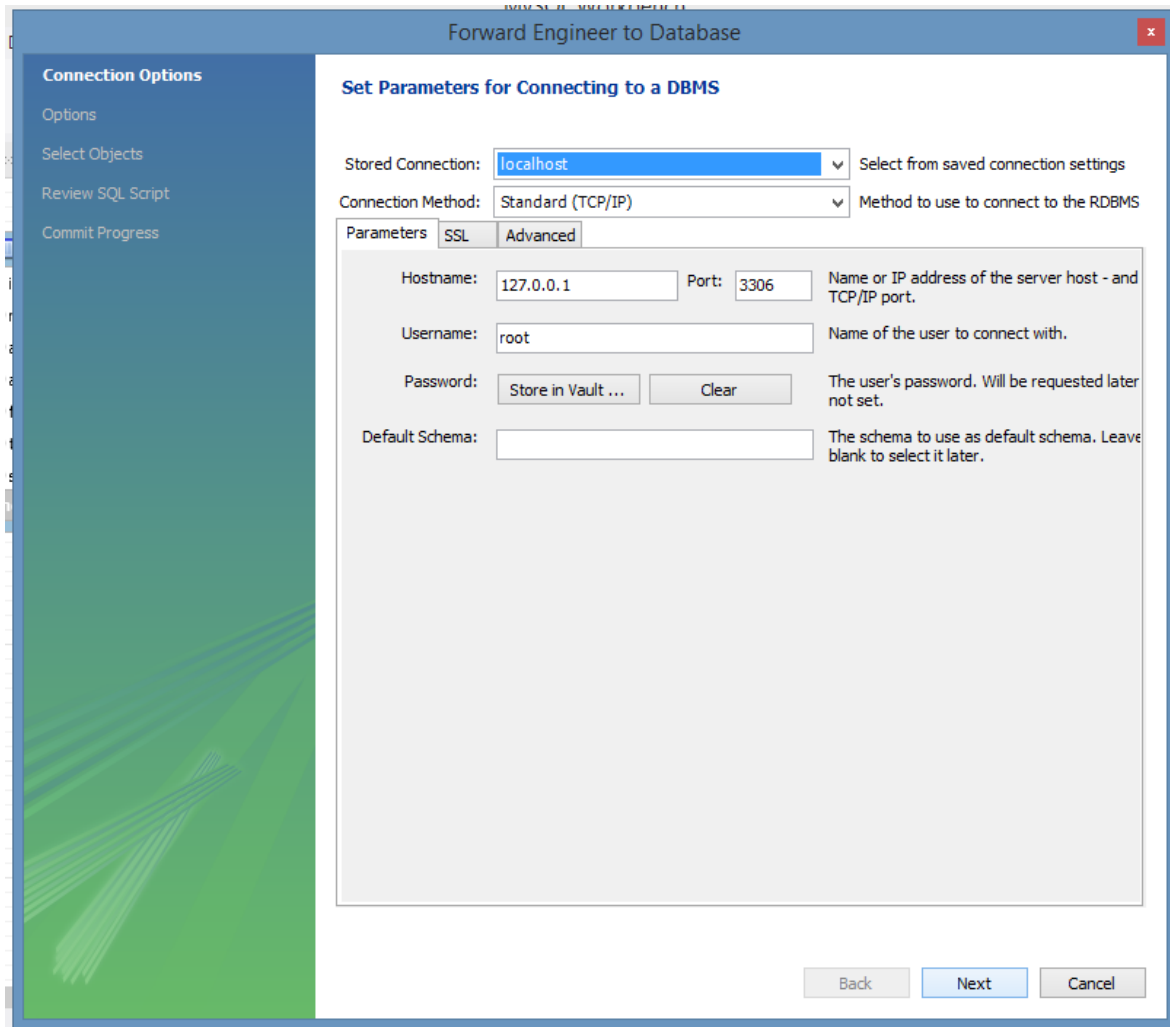
2.2 Actualización, modificación y eliminación del esquema de la base de datos.

Usaremos la opción de ingeniería directa (Forward engineering) que MySQL Workbench nos ofrece para crear el DDL (Lenguaje de Definición de Datos), para ellos nos ubicamos en el menú Database > Forward engineering o podemos simplemente presionar Control + G.



Taller de Base de Datos

Aparecerá entonces el siguiente cuadro de diálogo y seleccionaremos la conexión que se creó en el apartado [Extra: Instalación de MySQLWorkbench](#), en este caso, localhost corresponde al nombre de la conexión. Clic en Next.



Seleccionaremos las siguientes opciones y lo demás lo dejamos sin seleccionar. Clic en Next

- Drop objects before each CREATE object
- Generate DROP SCHEMA
- Include model attached scripts

Code Generation

- ☒ DROP objects before each CREATE object
- ☒ Generate DROP SCHEMA
- ☐ Omit schema qualifier in object names
- ☒ Generate USE statements
- ☐ Add SHOW WARNINGS after every DDL statement
- ☒ Include model attached scripts






Taller de Base de Datos

En la siguiente ventana dejaremos las opciones por defecto. Luego, clic en Next.

Forward Engineer to Database

Select Objects to Forward Engineer

To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.

	<input checked="" type="checkbox"/> Export MySQL Table Objects	3 Total Objects, 3 Selected	Show Filter
	<input type="checkbox"/> Export MySQL View Objects	0 Total Objects, 0 Selected	Show Filter
	<input type="checkbox"/> Export MySQL Routine Objects	0 Total Objects, 0 Selected	Show Filter
	<input type="checkbox"/> Export MySQL Trigger Objects	0 Total Objects, 0 Selected	Show Filter
	<input type="checkbox"/> Export User Objects	0 Total Objects, 0 Selected	Show Filter

Back Next Cancel

A continuación se mostrará el SQL que se utilizará para crear las tablas y las relaciones previamente diseñadas en el Modelo Entidad-Relación (MER). Puede ser guardado en un archivo de texto plano o ser copiado en el portapapeles. Clic en Next

Taller de Base de Datos

```
1  -- MySQL Workbench Forward Engineering
2
3  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_
5  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INV
6
7  -----
8  -- Schema nautica
9  -----
10 DROP SCHEMA IF EXISTS `nautica` ;
11
12 -----
13 -- Schema nautica
14 -----
15 CREATE SCHEMA IF NOT EXISTS `nautica` DEFAULT CHARACTER SET utf8 C
16 USE `nautica` ;
17
18 -----
19 -- Table `nautica`.`personas`
20 -----
21 DROP TABLE IF EXISTS `nautica`.`personas` ;
22
23 CREATE TABLE IF NOT EXISTS `nautica`.`personas` (
24   `id_persona` INT NOT NULL AUTO_INCREMENT,
25   `nombre` VARCHAR(45) NOT NULL,
26   `apellido_paterno` VARCHAR(45) NOT NULL,
27   `apellido_materno` VARCHAR(45) NOT NULL,
28   `fecha_nac` DATE NOT NULL,
29   `tel` VARCHAR(45) NOT NULL,
30   `socio` VARCHAR(2) NOT NULL,
31   PRIMARY KEY (`id_persona`))
32 ENGINE = InnoDB;
33
34
35
```

Save to File... Copy to Clipboard

Si todo está correcto, se ejecutará el SQL y creará la base de datos con todo lo diseñado previamente. Clic en Close para finalizar.

Forward Engineering Progress

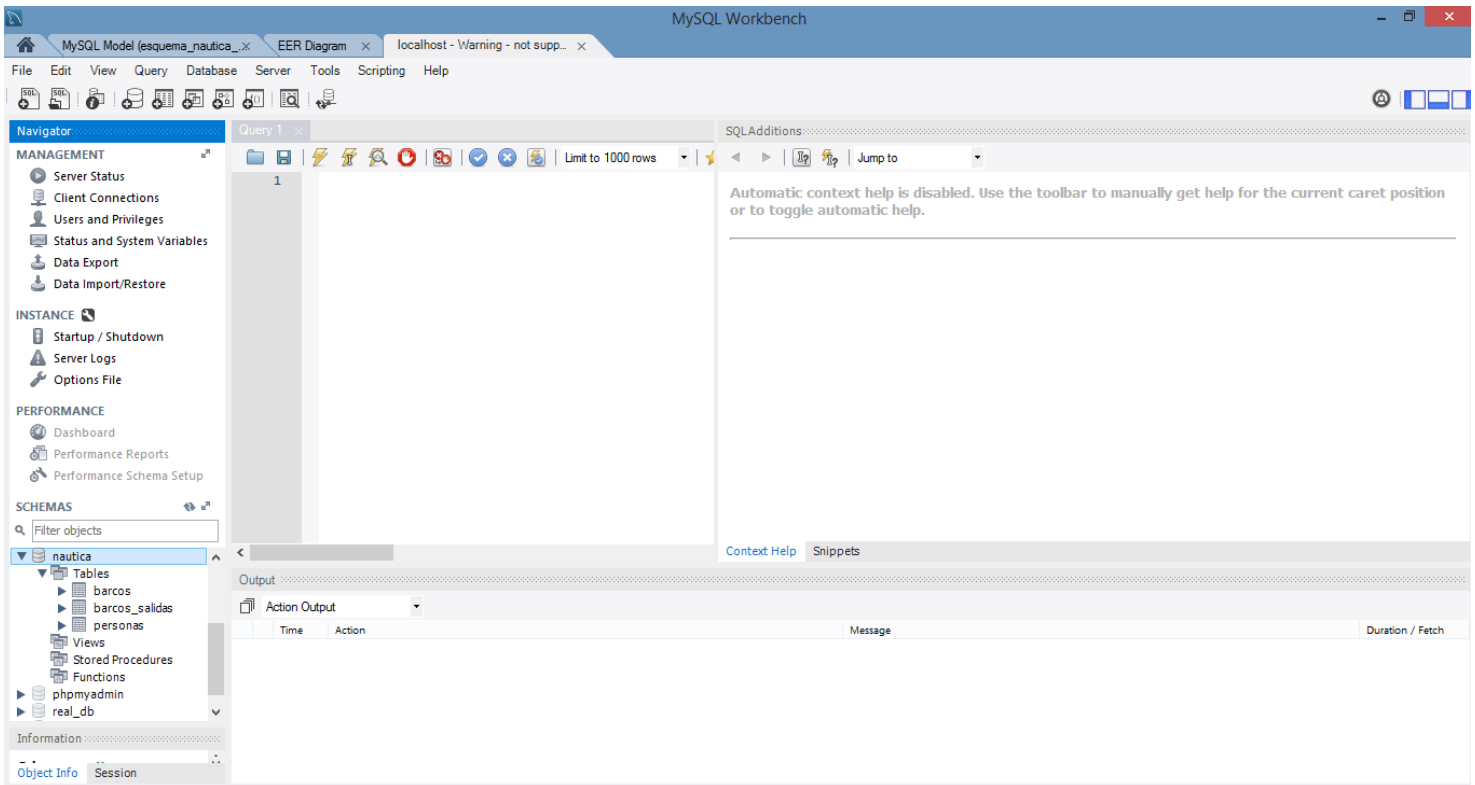
The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

- ☒ Connect to DBMS
- ☒ Execute Forward Engineered Script
- ☒ Read Back Changes Made by Server
- ☒ Save Synchronization State

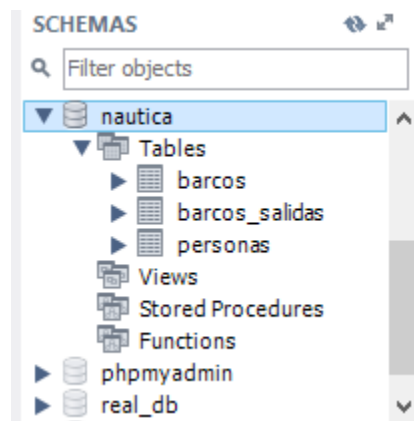
Forward Engineer Finished Successfully

Taller de Base de Datos

Abriremos la conexión para ver si efectivamente se ha creado, para ello damos clic en la pestaña que tiene por icono una casita en la parte superior izquierda. Luego, doble clic en el nombre de la conexión y se mostrará una ventana de esta forma.



En la parte izquierda inferior existe un apartado llamado SCHEMAS (Esquemas en inglés) donde podremos ver las base de datos que existen. Si nos fijamos bien podremos ver la base de datos.



Taller de Base de Datos

Extra: Explicación del script SQL generado por MySQL Workbench

```
DROP SCHEMA IF EXISTS `nautica` ;
```

Si el esquema o base de datos nautica existe, entonces se elimina por completo.

```
CREATE SCHEMA IF NOT EXISTS `nautica` DEFAULT CHARACTER SET utf8 ;
```

Crea la base de datos nautica si no existe, por defecto una el codificado de caracteres UTF-8 (muy usado).

```
USE `nautica` ;
```

Selecciona la base de datos nautica para poder ejecutar consultas a la misma.

```
DROP TABLE IF EXISTS `nautica`.`personas` ;
```

Si la tabla personas existe, entonces se elimina por completo.

```
CREATE TABLE IF NOT EXISTS `nautica`.`personas` (  
  `rfc` VARCHAR(10) NOT NULL,  
  `nombre` VARCHAR(45) NOT NULL,  
  `apellido_paterno` VARCHAR(45) NOT NULL,  
  `apellido_materno` VARCHAR(45) NOT NULL,  
  `fecha_nac` DATE NOT NULL,  
  `tel` VARCHAR(45) NOT NULL,  
  `socio` VARCHAR(2) NOT NULL,  
  PRIMARY KEY (`rfc`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

Crea la tabla personas si no existe, dentro de los paréntesis se define el nombre del campo, tipo, si es nulo o no y si es autoincrementable. Por tanto los campos a crear son:

- **rfc** *Tipo de dato varchar con tamaño de 10, no nulo.*
- **nombre** *Tipo de dato varchar con tamaño de 45, no nulo.*
- **apellido_paterno** *Tipo de dato varchar con tamaño de 45, no nulo*
- **apellido_materno** *Tipo de dato varchar con tamaño de 45, no nulo*
- **fecha_nac** *Tipo de dato date, no nulo*
- **tel** *Tipo de dato varchar con tamaño de 45, no nulo*
- **socio** *Tipo de dato varchar con tamaño de 2, no nulo*

La llave primaria de esta tabla es el atributo **rfc** y el motor a utilizar es InnoDB

```
DROP TABLE IF EXISTS `nautica`.`barcos` ;
```

Si la tabla barcos existe, entonces se elimina por completo.

Taller de Base de Datos

```
CREATE TABLE IF NOT EXISTS `nautica`.`barcos` (  
  `matricula` VARCHAR(10) NOT NULL,  
  `nombre` VARCHAR(100) NOT NULL,  
  `numero_amarre` INT(11) NOT NULL,  
  `cuota_pago` DOUBLE NOT NULL,  
  PRIMARY KEY (`matricula`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

Crea la tabla barcos si no existe, dentro de los paréntesis se define el nombre del campo, tipo, si es nulo o no y si es autoincrementable. Por tanto los campos a crear son:

- **matricula** *Tipo de dato varchar con tamaño de 10, no nulo.*
- **nombre** *Tipo de dato varchar con tamaño de 100, no nulo.*
- **numero_amarre** *Tipo de dato entero, no nulo*
- **cuota_pago** *Tipo de dato double, no nulo*

La llave primaria de esta tabla es el atributo **matricula** y el motor a utilizar es InnoDB.

```
DROP TABLE IF EXISTS `nautica`.`barcos_salidas` ;
```

Si la tabla barcos_salidas existe, entonces se elimina por completo.

```
CREATE TABLE IF NOT EXISTS `nautica`.`barcos_salidas` (  
  `rfc_persona` VARCHAR(10) NOT NULL,  
  `barcos_matricula` VARCHAR(10) NOT NULL,  
  `fecha` DATE NOT NULL,  
  `hora` TIME NOT NULL,  
  `destino` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`rfc_persona`, `barcos_matricula`),  
  INDEX `fk_personas_has_barcos_barcos1_idx` (`barcos_matricula` ASC),  
  INDEX `fk_personas_has_barcos_personas1_idx` (`rfc_persona` ASC),  
  CONSTRAINT `fk_personas_has_barcos_barcos1`  
    FOREIGN KEY (`barcos_matricula`)  
    REFERENCES `nautica`.`barcos` (`matricula`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
  CONSTRAINT `fk_personas_has_barcos_personas1`  
    FOREIGN KEY (`rfc_persona`)  
    REFERENCES `nautica`.`personas` (`rfc`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

Crea la tabla barcos_salida si no existe, dentro de los paréntesis se define el nombre del campo, tipo, si es nulo o no y si es autoincrementable. Por tanto los campos a crear son:

- **rfc_persona** *Tipo de dato varchar con tamaño de 10, no nulo.*
- **barcos_matricula** *Tipo de dato varchar con tamaño de 10, no nulo.*
- **fecha** *Tipo de dato date, no nulo*
- **hora** *Tipo de dato time, no nulo*

Taller de Base de Datos

- **destino**

Tipo de dato varchar con tamaño de 45, no nulo.

La llave primaria de esta tabla es la combinación de los atributos **rfc_persona** y **barcos_matricula**.

Los índices en MySQL permiten localizar y devolver registros de una forma sencilla y rápida.

Crea el índice **fk_personas_has_barcos_barcos1_idx** usando el atributo **barcos_matricula** de forma ascendente.

Crea el índice **fk_personas_has_barcos_personas1_idx** usando el atributo **personas_id_persona** de forma ascendente.

A continuación crea las restricciones (constraint) para las llaves foráneas, la referencia y la forma de actualización si es que el campo es alterado.

El motor que usará esta tabla es InnoDB.

Taller de Base de Datos

3 - Lenguaje de manipulación de datos(DML)

Marco teórico

¿Qué es DML?

Lenguaje de manipulación de datos (DML: Data Manipulation Language): Lenguaje artificial de cierta complejidad que permite el manejo y procesamiento del contenido de la base de datos. En la práctica puede consistir en un subconjunto de instrucciones de otro lenguaje informático. Las aplicaciones que trabajan sobre la base de datos se programan en un lenguaje de programación (C, Cobol, ...) insertando en el código fuente sentencias del DML. Al utilizar un DML se deben especificar los datos que serán afectados por las sentencias del lenguaje. Un DML puede tener o no procedimientos, según sea necesario especificar además cómo deben obtenerse esos datos. Los DML con procedimientos tienen sentencias de control de flujo como bucles o condicionales. Los DML sin procedimientos son conocidos también como declarativos.

Lenguaje de Manipulación de Datos (Data Manipulation Language, DML) es un lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos contenidos en las Bases de Datos del Sistema Gestor de Bases de Datos.

El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional.

Está conformado por las sentencias que nos permiten Ingresar, Actualizar y Borrar datos a una base de datos, de manera que podamos Manipular cada uno de ellos.

Sentencias DML

1.- **INSERT**: Por medio de esta sentencia podemos ingresar los datos en nuestras tablas. La sintaxis de esta sentencia es la siguiente:

```
INSERT INTO nombre_tabla (campo1,campo2) VALUES ('valor1','valor2');
```

Fácil no?? si deseamos insertar todos los datos dentro de nuestra tabla no es necesario llenar todos los campos sino que se ejecuta la sentencia de la siguiente manera:

```
INSERT INTO nombre_tabla VALUES('valor1','valor2');
```

2.-**REPLACE**: Como su nombre lo indica sirve para reemplazar, la sintaxis es igual que la de un INSERT, la diferencia es que si existe una clave bien sea primaria o única esta reemplazara el valor del registro quedando olvidado el anterior, si no existen claves dentro de la tabla el uso de esta sentencia no es útil ya que se comportaría como un INSERT.

3.-**UPDATE**: Por medio de esta sentencia podemos actualizar nuestros registros que tenemos dentro de nuestra tabla, la sintaxis es la siguiente:

```
UPDATE TABLE nombre_tabla SET campo1 = 'valor',campo2 = 'valor2' WHERE campo1 = 'valor';
```

Si nos fijamos en la sintaxis del UPDATE es simple, sencillamente indicamos la tabla que deseamos actualizar, cuales son los valores y luego establecemos una condición que indica cual registro actualizar, ojo hay que estar muy atentos al momento de realizar un UPDATE ya que de no

Taller de Base de Datos

establecer esta condición que indica cual registro actualizar se actualizaran todos los registros de nuestra tabla con los valores establecidos.

4.- DELETE: Este tipo de sentencia nos sirve para borrar registros de una tabla, su sintaxis es la siguiente:

```
DELETE FROM nombre_tabla WHERE campo = 'valor';
```

La sentencia del DELETE borra el campo especificado en la condición, al igual que como se indico con el UPDATE hay que tener en cuenta que si no se indica una condición se borrarán todos los registros de la tabla.

5.- TRUNCATE: Nos sirve para borrar todos los campos de una tabla, tiene algunas diferencias básicas con el DELETE, una de ellas que esta sentencia reinicia los campos autoincrementables, la sentencia TRUNCATE borra todos los campos de la tabla, sin establecerse ninguna condición, la sintaxis es la siguiente:

```
TRUNCATE TABLE nombre_tabla;
```

Como se puede ver en este tipo de sentencia no existe condición ya que es para el borrado completo de una tabla, yo normalmente la uso para reiniciar tablas cuando implemento sistemas.

Marco práctico

3.1 Inserción, eliminación y modificación de registros

Una vez creada la base de datos y las tablas correspondientes, en tiempo de ir poblándola con registros. Para ello usaremos la sentencia INSERT INTO, por ejemplo:

```
INSERT INTO `personas`  
  (`rfc`, `nombre`, `apellido_paterno`, `apellido_materno`, `fecha_nac`, `tel`, `socio`)  
VALUES ('AIAD552513', 'Elena', 'Alirio', 'Cordero', '1998-02-23', '9987653879', 'si');
```

Para la eliminación puede usarse la sentencia DELETE, por ejemplo:

```
DELETE * FROM personas WHERE nombre = 'Elena';
```

Esto borrará todos los registros que existan, o mejor dicho, a todas las personas que posean el nombre de Elena.

Para modificar un dato usamos la sentencia UPDATE, por ejemplo:

```
UPDATE `personas` SET `apellido_materno`='Ramirez' WHERE (`rfc`='AIAD552513');
```

En este caso se cambiará el apellido materno de la personas que posea el **rfc** igual a **AIAD552513**. Como este campo es una llave primaria solo afectará a una sola persona.

Taller de Base de Datos

3.2 Consultas de registros

3.2.1 Recuperación de datos

Para poder ver los datos se usa la sentencia SELECT, en este ejemplo se obtienen todos los registros de la tabla barcos.

```
SELECT * FROM barcos;
```

Al ejecutar la consulta el resultado sería algo parecido a esto.

Message	Result1	Profile	Status
matricula	nombre	numero_amarre	cuota_pago
BA1054	Emilio	1478	2474.74
BA1084	Paola	987	7172.32
BA1102	Valeria	1298	3329.98
BA1109	America	165	6250.3
BA1117	Isabela	701	2224.57
BA1121	Maximiliano	772	4251.52
BA1168	Elena	1877	5172.4
BA1171	Linda	1440	3616.8
BA1200	Jaime	844	4676.2
BA1211	Alexia	1776	4519.07
BA1232	Mateo	1867	4074.95
BA1242	Lucian	439	4077.79
BA1258	Israel	313	7885.85

+ - ✓ ✕ ↺ ⌂

```
SELECT * FROM barcos;
```

3.2.2 Restricción y ordenación de datos

La siguiente consulta muestra como se obtiene el registro que posea el numero de amarre igual a 701, es decir, solamente aquel o aquellos registros que pertenezcan a esa restricción, se muestra la consulta a continuación.

```
SELECT * FROM barcos WHERE numero_amarre = 701;
```

El resultado sería el siguiente.

Message	Result1	Profile	Status
matricula	nombre	numero_amarre	cuota_pago
BA1117	Isabela	701	2224.57

+ - ✓ ✕ ↺ ⌂

```
SELECT * FROM barcos WHERE numero_amarre = 701;
```

Taller de Base de Datos

Si quisiéramos saber el nombre, apellido paterno, apellido materno y si es socio de la náutica podríamos usar algún dato relevante, por ejemplo su RFC, ya que es único y no se repite, además lo destaca de los demás, la consulta sería la siguiente.

```
SELECT nombre,apellido_materno, apellido_paterno, socio
FROM personas
WHERE rfc = "BBMG039362";
```

El resultado a mostrar sería lo siguiente.

Message	Result1	Profile	Status								
	<table><thead><tr><th>nombre</th><th>apellido_materno</th><th>apellido_paterno</th><th>socio</th></tr></thead><tbody><tr><td>▶ Alfredo</td><td>Sixto</td><td>Alvarez</td><td>si</td></tr></tbody></table>	nombre	apellido_materno	apellido_paterno	socio	▶ Alfredo	Sixto	Alvarez	si		
nombre	apellido_materno	apellido_paterno	socio								
▶ Alfredo	Sixto	Alvarez	si								
SELECT nombre,apellido_materno, apellido_paterno, socio FROM personas WHERE rfc = "BBMG039362";											

Si quisiéramos saber que barcos pagan una cuota mayor a 1000 unidades monetarias, usaríamos la siguiente consulta.

```
SELECT * FROM barcos
WHERE cuota_pago >= 1000;
```

Lo que se mostrará sería lo siguiente.

Message	Result1	Profile	Status																																																				
	<table><thead><tr><th>matricula</th><th>nombre</th><th>numero_amarre</th><th>cuota_pago</th></tr></thead><tbody><tr><td>▶ BA1054</td><td>Emilio</td><td>1478</td><td>2474.74</td></tr><tr><td>BA1084</td><td>Paola</td><td>987</td><td>7172.32</td></tr><tr><td>BA1102</td><td>Valeria</td><td>1298</td><td>3329.98</td></tr><tr><td>BA1109</td><td>America</td><td>165</td><td>6250.3</td></tr><tr><td>BA1117</td><td>Isabela</td><td>701</td><td>2224.57</td></tr><tr><td>BA1121</td><td>Maximiliano</td><td>772</td><td>4251.52</td></tr><tr><td>BA1168</td><td>Elena</td><td>1877</td><td>5172.4</td></tr><tr><td>BA1171</td><td>Linda</td><td>1440</td><td>3616.8</td></tr><tr><td>BA1200</td><td>Jaime</td><td>844</td><td>4676.2</td></tr><tr><td>BA1211</td><td>Alexia</td><td>1776</td><td>4519.07</td></tr><tr><td>BA1232</td><td>Mateo</td><td>1867</td><td>4074.95</td></tr><tr><td>BA1242</td><td>Lucian</td><td>439</td><td>4077.79</td></tr></tbody></table>	matricula	nombre	numero_amarre	cuota_pago	▶ BA1054	Emilio	1478	2474.74	BA1084	Paola	987	7172.32	BA1102	Valeria	1298	3329.98	BA1109	America	165	6250.3	BA1117	Isabela	701	2224.57	BA1121	Maximiliano	772	4251.52	BA1168	Elena	1877	5172.4	BA1171	Linda	1440	3616.8	BA1200	Jaime	844	4676.2	BA1211	Alexia	1776	4519.07	BA1232	Mateo	1867	4074.95	BA1242	Lucian	439	4077.79		
matricula	nombre	numero_amarre	cuota_pago																																																				
▶ BA1054	Emilio	1478	2474.74																																																				
BA1084	Paola	987	7172.32																																																				
BA1102	Valeria	1298	3329.98																																																				
BA1109	America	165	6250.3																																																				
BA1117	Isabela	701	2224.57																																																				
BA1121	Maximiliano	772	4251.52																																																				
BA1168	Elena	1877	5172.4																																																				
BA1171	Linda	1440	3616.8																																																				
BA1200	Jaime	844	4676.2																																																				
BA1211	Alexia	1776	4519.07																																																				
BA1232	Mateo	1867	4074.95																																																				
BA1242	Lucian	439	4077.79																																																				
+ - ✓ ✕ ↺ ⌂ SELECT * FROM barcos WHERE cuota_pago >= 1000;																																																							

Taller de Base de Datos

De la misma forma podemos consultar aquellos barcos que tengan un número de amarre superior o igual a 1500, además ordenarlos de manera ascendente según el número de amarre. La consulta sería la siguiente.

```
SELECT * FROM barcos
WHERE numero_amarre >= 1500
ORDER BY numero_amarre;
```

El resultado que se mostrará puede verse de la siguiente manera.

Message	Result1	Profile	Status	
	matricula	nombre	numero_amarre	cuota_pago
▶	BA1211	Alexia	1776	4519.07
	BA1739	Pedro	1782	915.24
	BA1607	Fernanda	1783	882.44
	BA1566	Marco	1790	5693.04
	BA1590	Fernanda	1858	2525.93
	BA1232	Mateo	1867	4074.95
	BA1168	Elena	1877	5172.4
	BA1902	Guillermo	1884	2736.6
	BA1451	Nicolas	1941	2820.72

```
SELECT * FROM barcos
WHERE numero_amarre >= 1500
ORDER BY numero_amarre;
```

En el siguiente ejemplo se muestra la consulta para visualizar todas las salidas de barcos con fecha superior al 8 de Febrero del año 2004.

```
SELECT * FROM barcos_salidas
WHERE fecha >= 2004-02-08
ORDER BY fecha;
```

La consulta arrojaría lo siguiente.

Message	Result1	Profile	Status		
	rfc_persona	barcos_matricula	fecha	hora	destino
▶	DIU695472	BA1345	1995-06-16	16:20:43	Huimanguillo
	MJLH264839	BA1054	2000-08-13	08:30:51	Centro
	CKLJ610123	BA1902	2002-08-11	04:06:48	Huimanguillo
	AIAD552513	BA1232	2004-02-08	13:21:05	Emiliano Zapata
	EMBJ557200	BA1663	2004-09-01	22:14:57	Balancan
	LFAI034723	BA1451	2004-11-22	23:30:27	Nacajuca
	AIAD552513	BA1242	2005-02-11	16:29:21	Nacajuca
	DFCH950877	BA1200	2006-02-07	03:04:48	Macuspana
	EDBF467297	BA1607	2006-06-26	05:48:05	Huimanguillo
	FLMH862656	BA1856	2008-10-26	14:44:49	Nacajuca

+

-

✓

✕

↺

🔍

```
SELECT * FROM barcos_salidas WHERE fecha >= 2004-02-08 ORDER BY fecha;
```

Taller de Base de Datos

3.2.3 Informes de datos agregados mediante funciones de grupo

La cláusula de grupo nos permite juntar los elementos que tengan en común una columna, es decir, para este ejemplo tenemos una relación de muchos a muchos con personas y barcos en la tabla `barcos_salidas`, pero... ¿Cómo podemos saber cuántos barcos tiene cada persona?. Para saber eso ejecutamos el siguiente ejemplo, el cual nos devolverá la cantidad de barcos que posee cada persona.

```
SELECT rfc_persona, COUNT(*) as cantidad_barcos  
FROM barcos_salidas  
GROUP BY barcos_matricula  
ORDER BY cantidad_barcos;
```

El resultado sería el siguiente.

Message	Result1	Profile	Status
	rfc_persona	cantidad_barcos	
▶	DFCH950877	1	
	MAIM669494	1	
	LFAI034723	1	
	EDBF467297	1	
	FLMH862656	1	
	LMLE981803	1	
	MJLH264839	1	
	EMBJ557200	1	
	EDBF467297	2	
	EACJ483884	2	
	AIAD552513	2	
	CEIE997246	3	
	AIAD552513	5	

+

-

✓

✕

↺

🔍

```
SELECT rfc_persona, COUNT(*) as cantidad_barcos  
FROM barcos_salidas  
GROUP BY barcos_matricula ORDER BY cantidad_barcos;
```

3.2.4 Visualización de datos de varias tablas

Podemos también seleccionar varios campos de varias tablas y con ello crear una nueva, aunque solo se refleja en una vista. De esta manera podemos acceder a ver varios valores con una consulta. Supongamos que queremos saber el nombre de la persona que usó un barco y con ello la hora y destino. No tenemos una tabla que tenga el nombre de la persona pero podemos mandar esa consulta, por ejemplo.

```
SELECT nombre, hora, destino  
FROM personas, barcos_salidas  
WHERE rfc = rfc_persona;
```

Taller de Base de Datos

El resultado mostrado por la consulta sería la que se muestra a continuación.

Message	Result1	Profile	Status
	nombre	hora	destino
	Elena	13:21:05	Emiliano Zapata
	Elena	16:29:21	Nacajuca
	Sandra	22:45:34	Macuspana
	Jairo	04:06:48	Huimanguillo
	Emiliano	03:04:48	Macuspana
	Emiliano	11:28:53	Cardenas
	Israel	16:20:43	Huimanguillo
	Elsa	24:44:10	Macuspana
	Ruben	07:03:57	Teapa
	Ruben	16:39:41	Huimanguillo
	Ruben	05:48:05	Huimanguillo
►	Luis	22:18:12	Nacajuca
	Luis	14:14:29	Huimanguillo

+

-

✓

✕

↺

🔍

```
SELECT nombre, hora, destino
FROM personas, barcos_salidas
WHERE rfc = rfc_persona;
```

3.2.5 Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE o dentro de otra subconsulta. Supongamos que queremos saber el nombre de la persona, si es o no socio de la náutica y el teléfono de la persona del barco con matrícula BA1610, entonces la consulta sería la siguiente.

```
SELECT nombre, socio, tel
FROM personas
WHERE rfc = (
SELECT rfc_persona
FROM barcos_salidas
WHERE barcos_matricula = "BA1610");
```

La vista sería la siguiente.

Message	Result1	Profile	Status
	nombre	socio	tel
►	Luis	no	9968384791

```
SELECT nombre, socio, tel FROM personas
WHERE rfc = ( SELECT rfc_persona FROM barcos_salidas WHERE barcos_matricula = "BA1610");
```

Taller de Base de Datos

3.2.6 Operadores set

La cláusula SET indica las columnas a modificar y los valores que deben tomar. Supongamos que un barco con número inicial de amarre equivalente a 987 desea ser modificado a número de amarre 1000, la consulta sería la siguiente.

```
UPDATE barcos  
SET numero_amarre = 1000  
WHERE numero_amarre = 987;
```

Esta consulta no regresa nada, es más bien una instrucción que se ejecuta para alterar el valor o mejor dicho, reemplazar el valor anterior por uno actual.

Taller de Base de Datos

Conclusión

Cuando la base de datos contiene muchos registros es casi imposible buscarlos y analizarlos de manera manual, para ello existen herramientas gráficas que permiten obtener los valores deseados o simplemente hacer las consultas sin necesidad de comprometer la información, eso en cuanto a las consultas de obtención de información.

Por otro lado existen consultas que permiten alterar un registro o dicho de manera técnica, una tupla. Es decir, la fila que corresponde a un registro en la base de datos. Para llevar a cabo una consulta es necesario saber las tablas y las respectivas columnas que se van a utilizar.

La ventaja de usar una herramienta de desarrollo como MySQL Workbench es la enorme facilidad para poder realizar consultas y visualizarlas, además de crear la base de datos con el Modelo Entidad-Relación o al aplicar ingeniería inversa a una base de datos, por lo cual se puede obtener el Modelo Entidad-Relación de esa respectiva base de datos.

No necesariamente se deben mostrar todas las columnas de las tablas, es por ello que se aplican restricciones, es decir, una condición para que se filtre la consulta y devuelva específicamente lo que uno está buscando, con ayuda de WHERE es realmente sencillo de entender y además de consultar.

Se recomienda llevar una práctica constante en las consultas ya que son muy usadas en desarrollo web o cualquier otro sistema que necesite de un filtrado de información.

Taller de Base de Datos

Anexo

Link de repositorio el GIT del script escrito en python para generar los datos, la base de datos en un archivo txt y además el reporte en formato PDF.

https://github.com/Reynald0/taller_bd

Bibliografía

Aula Cic. (2012). El DDL, Lenguaje de Definición de Datos (I). Febrero 18, 2016, de Aula Clic Sitio web: http://www.aulaclic.es/sqlserver/t_8_1.htm

MySQL. (2016). MySQL 5.7 Reference Manual. Febrero 15, 2016, de MySQL Sitio web: <http://dev.mysql.com/doc/refman/5.7/en/http://www.desarrolloweb.com/articulos/intro-indices-mysql.html>

Conchoi. (2004). Integridad referencial en MySQL . Febrero 15, 2006, de Programacion.net Sitio web: http://programacion.net/articulo/integridad_referencial_en_mysql_263/4

Russvell Oblitas Valenzuela. (2007). Habilitando InnoDB en MySQL. Febrero 15, 2016, de desarrolloweb.com Sitio web: <http://www.desarrolloweb.com/articulos/habilitando-innobd-en-mysql.html>

Wikipedia. (2015). Lenguaje de manipulación de datos. Febrero 25, 2016, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Lenguaje_de_manipulaci%C3%B3n_de_datos#Clasificaci.C3.B3n_de_l os_DML

MySQL con clase. (2005). UPDATE. Febrero 15, 2016, de MySQL con clase Sitio web: <http://mysql.conclase.net/curso/?sqlsen=update>

Edu4Java. (2008). DDL y DML. Febrero 15, 2016, de Edu4java Sitio web: <http://www.edu4java.com/es/sql/sql4.html>