

Instituto Tecnológico de Villahermosa
Ing. Sistemas Computacionales
6to Semestre



Taller de Base de Datos

Reporte Unidad 6 y 7

Reynaldo Bernard de Dios de la Cruz



Domingo, 22 de Mayo
de 2016

Taller de Base de Datos

Contenido

Introducción	3
6 - SQL procedural	4
Marco teórico	4
6.1 Procedimientos almacenados	4
6.2 Disparadores	5
Marco práctico	7
7 - Conectividad de Bases de Datos	9
Marco teórico	9
7.1 ODBC	9
7.2 ADO.NET	10
7.3 JDBC	10
Marco práctico	10
7.4 Conectividad desde un lenguaje huésped	10
Conclusión.....	15
Anexo	15
Bibliografía.....	16

Taller de Base de Datos

Introducción

Los Triggers o Disparadores son objetos que se asocian con tablas y se almacenan en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

La utilidad principal de un trigger es mejorar la administración de la base de datos, ya que no requieren que un usuario los ejecute. Por lo tanto, son empleados para implementar las REGLAS DE NEGOCIO (tipo especial de integridad) de una base de datos. Una Regla de Negocio es cualquier restricción, requerimiento, necesidad o actividad especial que debe ser verificada al momento de intentar agregar, borrar o actualizar la información de una base de datos. Un trigger puede prevenir errores en los datos, modificar valores de una vista, sincronizar tablas, entre otros.

Open DataBase Connectivity (ODBC) es un estándar de acceso a las bases de datos. El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos.

Java Database Connectivity, más conocida por sus siglas JDBC, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Taller de Base de Datos

6 - SQL procedural

Marco teórico

6.1 Procedimientos almacenados

Un procedimiento es un conjunto de instrucciones que se guardan en el servidor para un posterior uso, ya que se ejecutarán frecuentemente. En MySQL se nombran con la cláusula PROCEDURE.

A diferencia de las funciones, los procedimientos son rutinas que no retornan en ningún tipo de valor. Simplemente se llaman desde el cliente con un comando y las instrucciones dentro del procedimiento se ejecutarán.

Ventajas de usar procedimientos en MySQL

- **Seguridad:** Los procedimientos ocultan el nombre de las tablas a usuarios que no tengan los privilegios para manipular datos. Simplemente llaman los procedimientos sin conocer la estructura de la base de datos.
- **Estándares de código:** En un equipo de desarrollo usar el mismo procedimiento permite crear sinergia en las fases de construcción. Si cada programador crea su propio procedimiento para realizar la misma tarea, entonces podrían existir problemas de integridad y pérdida de tiempo
- **Velocidad:** Es mucho más fácil ejecutar un programa ya definido mediante ciertos parámetros, que reescribir de nuevo las instrucciones.

Crear un procedimiento en MySQL

La creación de un procedimiento se inicia con las cláusulas **CREATE PROCEDURE**. Luego definimos un nombre y los parámetros que necesita para funcionar adecuadamente. Veamos su sintaxis:

```
CREATE PROCEDURE nombre ([parámetro1,parámetro2,...])  
[Atributos de la rutina]  
BEGIN  
    instrucciones  
END
```

Un procedimiento puede tener uno o más parámetros o también no tener ninguno. Puede carecer de atributos o puede poseer varios. Y como ves, el cuerpo del procedimiento es un bloque de instrucciones definido.

Ejemplo de un procedimiento con un parámetro in

En el siguiente ejemplo desarrollemos un procedimiento para el siguiente requerimiento:

Imprima los números del 1 hasta n, donde n esta dado por el usuario.

Usaremos un procedimiento para capturar el numero n del usuario. Incorporaremos una variable contadora que comience en 1 y un WHILE para el incremento e impresión. Veamos:

```
DELIMITER //  
  
CREATE PROCEDURE numeros_1_hasta_n (IN n INT)  
BEGIN  
    DECLARE contador INT DEFAULT 1;  
    WHILE contador<=n DO
```

Taller de Base de Datos

```
SELECT contador;  
SET contador = contador + 1 ;  
END WHILE;  
END//  
  
DELIMITER ;
```

La sentencia DELIMITER cambia el carácter de terminación ';' por cualquier otro carácter, en este caso elegimos '/'. Se hace con el fin de que MySQL no termine el procedimiento al encontrar el primer punto y coma. Al final restablecemos el valor original del carácter de escape.

¿Cómo ejecuto un procedimiento ya almacenado?

Usaremos el comando **CALL** enseguida del nombre del procedimiento y si tiene parámetros, entonces se ingresan sus parámetros. Ahora veamos cómo llamar al anterior procedimiento:

```
CALL numeros_1_hasta_n(5)
```

6.2 Disparadores

Un **trigger** o **disparador** en MySQL es un programa almacenado(stored program), creado para ejecutarse automáticamente cuando ocurra un evento en nuestra base de datos. Dichos eventos son generados por los comandos INSERT, UPDATE y DELETE, los cuales hacen parte del DML(Data Modeling Language) de SQL.

Esto significa que invocaremos nuestros Triggers para ejecutar un bloque de instrucciones que proteja, restrinja o preparen la información de nuestras tablas, al momento de manipular nuestra información. Para crear triggers en MySQL necesitas los privilegios SUPER Y TRIGGER.

Crear un trigger en MySQL

Usaremos una sintaxis similar a la creación de Procedimientos en MySQL. Observemos:

```
CREATE [DEFINER={usuario|CURRENT_USER}]  
TRIGGER nombre_del_trigger {BEFORE|AFTER} {UPDATE|INSERT|DELETE}  
ON nombre_de_la_tabla  
FOR EACH ROW  
<bloque_de_instrucciones>
```

Obviamente la sentencia CREATE es conocidísima para crear nuevos objetos en la base de datos. Eso ya lo tienes claro. Enfoquemos nuestra atención en las otras partes de la definición:

- **DEFINER={usuario|CURRENT_USER}**: Indica al gestor de bases de datos qué usuario tiene privilegios en su cuenta, para la invocación de los triggers cuando surjan los eventos DML. Por defecto esta característica tiene el valor CURRENT_USER que hace referencia al usuario actual que está creando el Trigger.
- **nombre_del_trigger**: Indica el nombre de nuestro trigger. Existe una nomenclatura muy práctica para nombrar un trigger, la cual nos da mejor legibilidad en la administración de la base de datos. Primero ponemos el nombre de tabla, luego especificamos con la inicial de la operación DML y seguido usamos la inicial del momento de ejecución(AFTER o BEFORE). Por ejemplo:

Taller de Base de Datos

clientes_BI_TRIGGER

- **BEFORE|AFTER:** Especifica si el Trigger se ejecuta antes o después del evento DML.
- **UPDATE|INSERT|DELETE:** Aquí eliges que sentencia usarás para que se ejecute el Trigger.
- **ON nombre_de_la_tabla:** En esta sección estableces el nombre de la tabla asociada.
- **FOR EACH ROW:** Establece que el Trigger se ejecute por cada fila en la tabla asociada.
- **<bloque_de_instrucciones>:** Define el bloque de sentencias que el Trigger ejecutará al ser invocado.

Identificadores NEW y OLD en triggers

Si queremos relacionar el trigger con columnas específicas de una tabla debemos usar los identificadores OLD y NEW.

OLD indica el valor antiguo de la columna y NEW el valor nuevo que pudiese tomar. Por ejemplo: OLD.idproducto ó NEW.idproducto.

Si usamos la sentencia UPDATE podremos referirnos a un valor OLD y NEW, ya que modificaremos registros existentes por nuevos valores. En cambio si usamos INSERT solo usaremos NEW, ya que su naturaleza es únicamente de insertar nuevos valores a las columnas. Y si usamos DELETE usaremos OLD debido a que borraremos valores que existen con anterioridad. Es decir:

Sentencia	Valor(es)
UPDATE	OLD y NEW
INSERT	NEW
DELETE	OLD

Triggers BEFORE y AFTER

Estas cláusulas indican si el Trigger se ejecuta antes o después del evento DML. Hay ciertos eventos que no son compatibles con estas sentencias.

Por ejemplo, si tuvieras un Trigger AFTER que se ejecuta en una sentencia UPDATE, sería ilógico editar valores nuevos NEW, sabiendo que el evento ya ocurrió. Igual sucedería con la sentencia INSERT, el Trigger tampoco podría referenciar valores NEW, ya que los valores que en algún momento fueron NEW, han pasado a ser OLD.

Ver la información de un trigger en MySQL

Usa el comando **SHOW CREATE TRIGGER** y rápidamente estarás viéndolas especificaciones de tu Trigger creado. Observa el siguiente ejemplo:

```
SHOW CREATE TRIGGER futbolista_ai_trigger;
```

También puedes ver los Triggers que hay en tu base de datos con:

```
SHOW TRIGGERS;
```

Eliminar un trigger en MySQL

DROP, como ya sabes usamos este comando para eliminar casi cualquier cosa en nuestra base de datos:

Taller de Base de Datos

```
DROP TRIGGER [IF EXISTS] nombre_trigger
```

Recuerda que podemos adicionar la condicion IF EXISTS para indica que si el Trigger ya existe, entonces que lo borre.

¿Qué utilidades tienen los triggers?

Con los Triggers podemos implementar varios casos de uso que mantengan la integridad de la base de datos, como Validar información, Calcular atributos derivados, Seguimientos de movimientos en la base de datos, etc.

Cuando surja una necesidad en donde veas que necesitas que se ejecute una acción implícitamente(sin que la ejecutes manualmente) sobre los registros de una tabla, entonces puedes considerar el uso de un Trigger.

Marco práctico

En este ejemplo se muestra la creación de diez disparadores para la base de datos NAUTICA y además la creación de una nueva tabla donde se almacenaran los movimientos que haga el usuario.

Creación de tabla para triggers

Esta tabla se encarga de almacenar el usuario, fecha, acción y el nombre de la tabla donde se hizo el movimiento, para ello vamos a ocupar todos los eventos disponibles, es decir, INSERT, DELETE y UPDATE en cada tabla, para este caso la tabla barcos, personas y barcos_salidas.

```
CREATE TABLE `movimientos` (  
  `usuario` varchar(100) NOT NULL,  
  `fecha` datetime NOT NULL,  
  `accion` varchar(1) NOT NULL,  
  `tabla` varchar(30) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

1 Trigger BEFORE INSERT - barcos

```
CREATE TRIGGER insertar_barcos AFTER INSERT ON barcos  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'I', 'barcos');
```

2 Trigger BEFORE DELETE - barcos

```
CREATE TRIGGER eliminar_barcos AFTER INSERT ON barcos  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'E', 'barcos');
```

3 Trigger BEFORE UPDATE - barcos

```
CREATE TRIGGER actualizar_barcos AFTER UPDATE ON barcos
```

Taller de Base de Datos

```
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'A', 'barcos');
```

4 Trigger BEFORE INSERT - personas

```
CREATE TRIGGER insertar_personas AFTER INSERT ON personas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'I', personas);
```

5 Trigger BEFORE DELETE - personas

```
CREATE TRIGGER eliminar_personas AFTER DELETE ON personas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'E', personas);
```

6 Trigger BEFORE UPDATE - personas

```
CREATE TRIGGER actualizar_personas AFTER UPDATE ON personas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'A', personas);
```

7 Trigger BEFORE INSERT - barcos_salidas

```
CREATE TRIGGER insertar_barcos_salidas AFTER INSERT ON barcos_salidas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'I', barcos_salidas);
```

8 Trigger BEFORE DELETE - barcos_salidas

```
CREATE TRIGGER eliminar_barcos_salidas AFTER DELETE ON barcos_salidas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'E', barcos_salidas);
```

9 Trigger BEFORE UPDATE - barcos_salidas

```
CREATE TRIGGER actualizar_barcos_salidas AFTER UPDATE ON barcos_salidas  
FOR EACH ROW  
INSERT INTO movimientos VALUES(user(), now(), 'A', barcos_salidas);
```


Taller de Base de Datos

10 Trigger AFTER INSERT barcos

El siguiente trigger inserta en una tabla llamada total el valor de la suma de todas las cuotas de pago de los barcos y se activa cuando se inserta un nuevo barco.

Tabla:

```
CREATE TABLE `total` (  
  `total_cuota_pago` double NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Trigger:

```
DELIMITER //  
BEGIN  
DECLARE total_pago DOUBLE;  
SELECT SUM(cuota_pago) INTO total_pago FROM barcos;  
INSERT INTO total VALUES(total_pago);  
END//  
  
DELIMITER ;
```

Ahora solo falta hacer unos movimientos y consultarlos de la tabla movimientos para ver si se estan activando, para ello se consulta la tabla movimientos.

```
mysql> SELECT * FROM movimientos;
```

usuario	fecha	accion	tabla
root@localhost	2016-05-16 17:23:56	I	barcos
root@localhost	2016-05-16 17:24:02	E	barcos
root@localhost	2016-05-16 17:48:28	I	barcos
root@localhost	2016-05-22 11:39:36	I	barcos
root@localhost	2016-05-22 11:40:32	I	barcos
root@localhost	2016-05-22 11:42:29	I	barcos
root@localhost	2016-05-22 11:43:47	I	barcos
root@localhost	2016-05-22 11:44:52	E	barcos
root@localhost	2016-05-22 11:44:54	I	barcos

9 rows in set (0.00 sec)

7 - Conectividad de Bases de Datos

Marco teórico

7.1 ODBC

El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos. ODBC logra esto al insertar una capa intermedia (CLI) denominada nivel de Interfaz de Cliente SQL, entre la aplicación y el DBMS.

El propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda. Para que esto funcione tanto la aplicación como el DBMS deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el DBMS debe ser capaz de responder a ellos.

Taller de Base de Datos

El software funciona de dos modos, con un software manejador en el cliente, o una filosofía cliente-servidor. En el primer modo, el driver interpreta las conexiones y llamadas SQL y las traduce desde el API ODBC hacia el DBMS. En el segundo modo para conectarse a la base de datos se crea una DSN dentro del ODBC que define los parámetros, ruta y características de la conexión según los datos que solicite el creador o fabricante.

7.2 ADO.NET

ADO.NET es un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos. Es una parte de la biblioteca de clases base que están incluidas en el Microsoft .NET Framework. Es comúnmente usado por los programadores para acceder y para modificar los datos almacenados en un Sistema Gestor de Bases de Datos Relacionales, aunque también puede ser usado para acceder a datos en fuentes no relacionales. ADO.NET es a veces considerado como una evolución de la tecnología ActiveX Data Objects (ADO), pero fue cambiado tan extensivamente que puede ser concebido como un producto enteramente nuevo.

7.3 JDBC

Java Database Connectivity (JDBC) es un derivado inspirado en el mismo, una interfaz de programación de aplicaciones que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

Marco práctico

7.4 Conectividad desde un lenguaje huésped

Para este ejemplo se tiene aún la base de datos NAUTICA y para no hacer tan pesado el ejercicio solamente se requiere mostrar los registros que la tabla barcos posee. Sus respectivos atributos son:

matricula	nombre	numero_amarre	cuota_pago
-----------	--------	---------------	------------

Para ello se usará el conector JDBC y ODBC, en el caso de ODBC se usará un puente entre JDBC y ODBC para permitir la conexión.

A continuación se muestra un resumen de las líneas necesitadas en Java para realizar la conexión.

IMPORTANTE:

Para ambos casos es necesario tener el controlador instalado.

JDBC

Para crear la conexión usando JDBC es necesario saber el host, usuario y contraseña de la base de datos, para ello se crean las siguientes variables.

```
private static String URL = "jdbc:mysql://localhost/nautica";
```

En la URL hay que dejar el inicio de esa forma, es decir, **jdbc:mysql:** ya que le indica a Java que conector usará. Después de los dos puntos y seguido de doble slash (//) se especifica el HOST seguidamente un slash y el nombre de la base de datos a la cual se conectará.

Taller de Base de Datos

```
private static final String USUARIO = "root";  
private static final String CONTRASENA = "12345";
```

Se declara también la conexión

```
private Connection conexion = null;
```

Ahora, para crear la conexión simplemente hay que poner la siguiente línea.

```
conexion = DriverManager.getConnection(URL,USUARIO,CONTRASENA);
```

Así de simple, si no ocurre ningún error entonces la conexión a la base de datos fue exitosa.

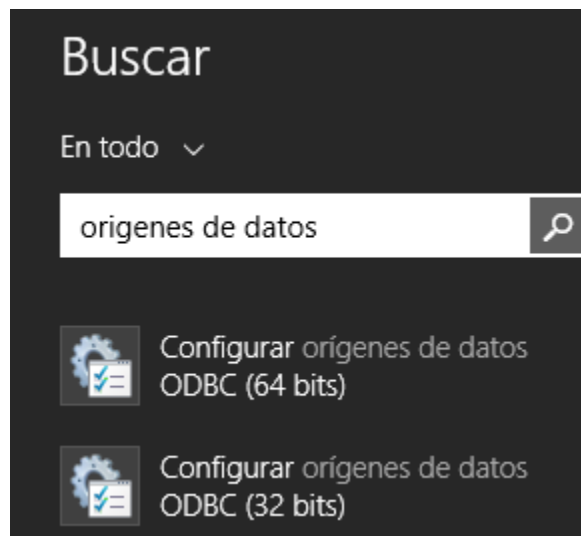
ODBC

Para crear la conexión usando ODBC es necesario el nombre del ODBC y este es dado a través del sistema operativo los pasos para crear un Orígenes de Datos (ODBC) son los siguientes (en Windows 8):

IMPORTANTE:

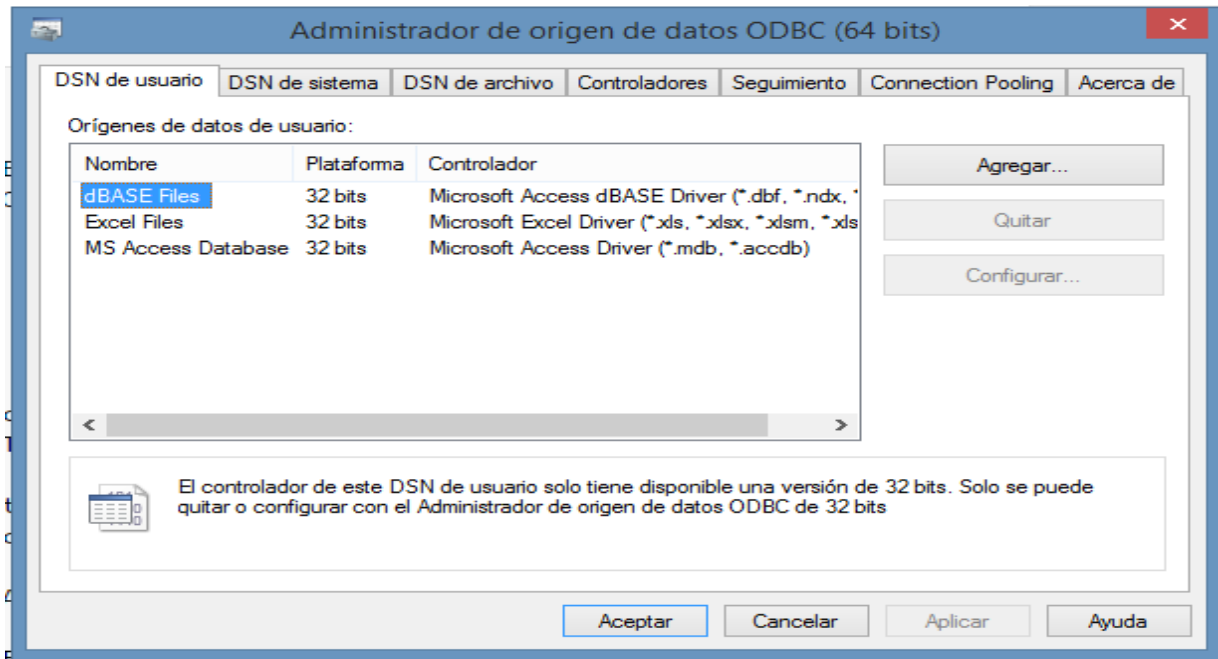
En Java 8 ya no puedes hacer puente JDBC - ODBC, tienes que usar la versión 7 o menor ya que la librería necesaria para conectarse usando el driver ODBC deja de tener soporte a partir de la versión 8 de Java.

1. Presionar la tecla Windows.
2. Escribir orígenes de datos y abrir.

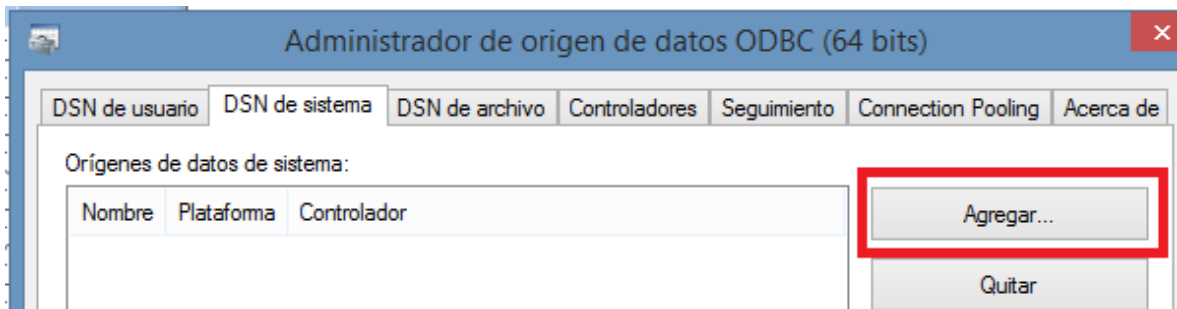


3. Aparecerá una ventana como la siguiente.

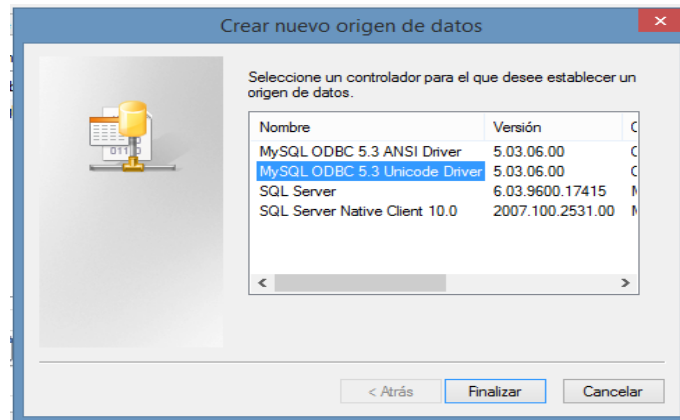
Taller de Base de Datos



4. Clic en la pestaña DSN de sistema y a continuación en el botón agregar

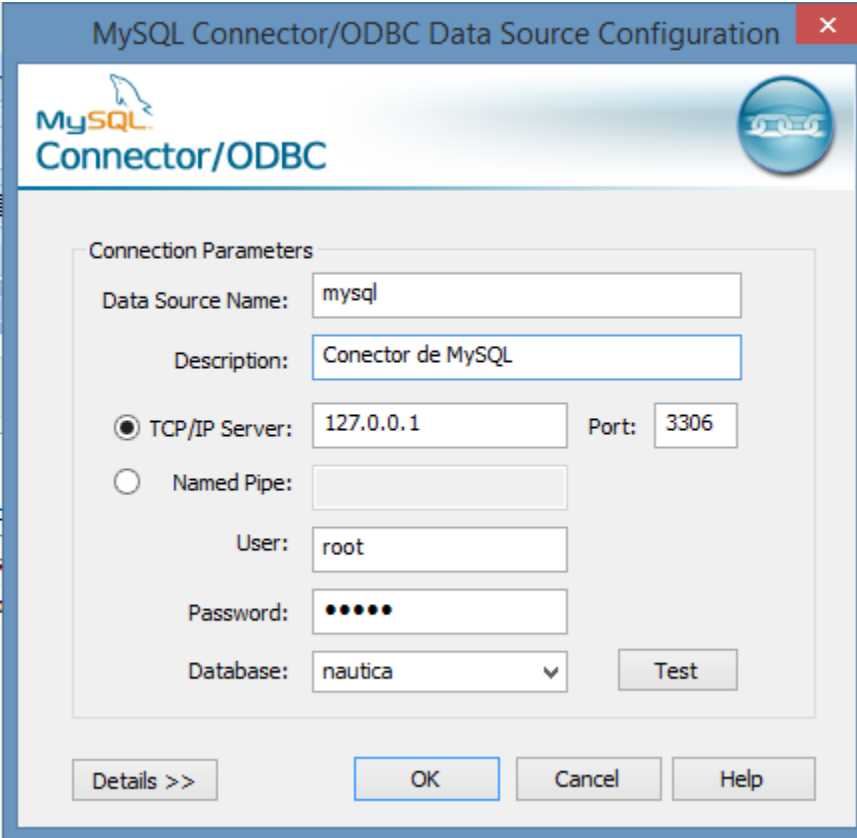


5. Seleccionar la opción de MySQL ODBC, aquí se está eligiendo para que SGBD se va a usar. En este ejemplo MySQL. Clic en Finalizar



Taller de Base de Datos

6. Se abrirá un formulario y es simplemente rellenarlo con los datos.



The screenshot shows the 'MySQL Connector/ODBC Data Source Configuration' dialog box. It has a title bar with a close button. The main area is titled 'MySQL Connector/ODBC'. Below this, there's a section for 'Connection Parameters'. The 'Data Source Name' is 'mysql'. The 'Description' is 'Conector de MySQL'. The 'TCP/IP Server' is selected, with the IP address '127.0.0.1' and 'Port' '3306'. The 'Named Pipe' option is unselected. The 'User' is 'root'. The 'Password' is masked with dots. The 'Database' is 'nautica'. There is a 'Test' button. At the bottom, there are buttons for 'Details >>', 'OK', 'Cancel', and 'Help'.

MySQL Connector/ODBC Data Source Configuration

MySQL Connector/ODBC

Connection Parameters

Data Source Name: mysql

Description: Conector de MySQL

☒ TCP/IP Server: 127.0.0.1 Port: 3306

☐ Named Pipe:

User: root

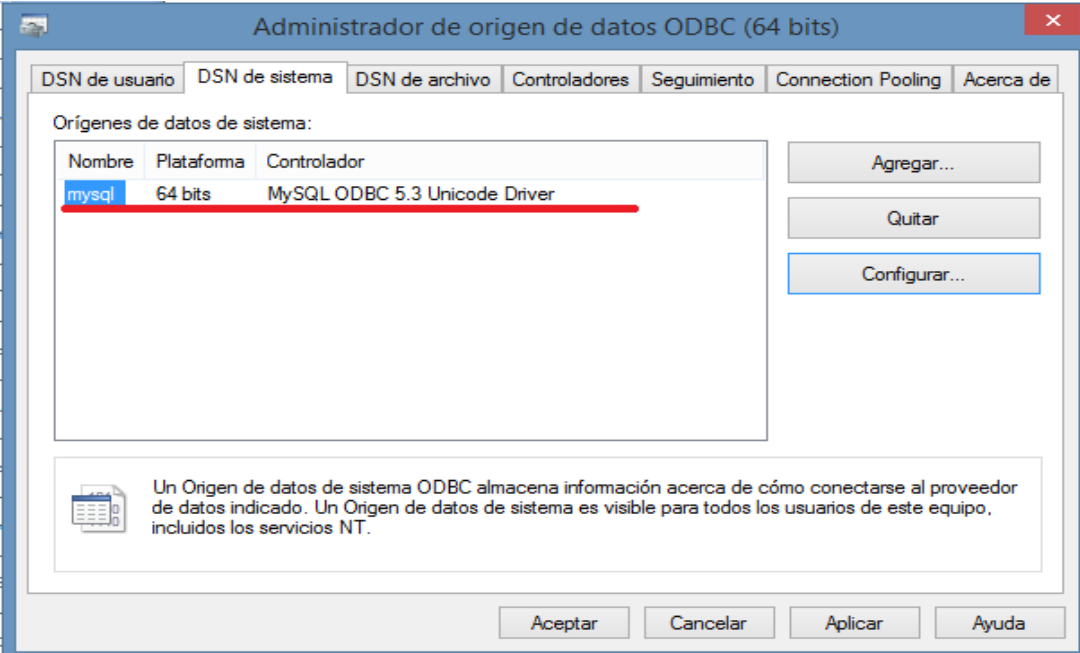
Password: •••••

Database: nautica

Test

Details >> OK Cancel Help

7. Se puede probar la conexión apretando el botón "Test". Una vez finalizado todo el formulario basta con dar clic en OK y deberá aparecer de la siguiente manera.



The screenshot shows the 'Administrador de origen de datos ODBC (64 bits)' window. It has a title bar with a close button. The 'DSN de sistema' tab is selected. The 'Orígenes de datos de sistema:' section shows a table with columns 'Nombre', 'Plataforma', and 'Controlador'. The first row is 'mysql', '64 bits', and 'MySQL ODBC 5.3 Unicode Driver'. To the right of the table are buttons 'Agregar...', 'Quitar', and 'Configurar...'. At the bottom, there is a text box with a document icon and a description of system DSNs. At the very bottom are buttons 'Aceptar', 'Cancelar', 'Aplicar', and 'Ayuda'.

Administrador de origen de datos ODBC (64 bits)

DSN de usuario DSN de sistema DSN de archivo Controladores Seguimiento Connection Pooling Acerca de

Orígenes de datos de sistema:

Nombre	Plataforma	Controlador
mysql	64 bits	MySQL ODBC 5.3 Unicode Driver

Agregar...
Quitar
Configurar...

Un Origen de datos de sistema ODBC almacena información acerca de cómo conectarse al proveedor de datos indicado. Un Origen de datos de sistema es visible para todos los usuarios de este equipo, incluidos los servicios NT.

Aceptar Cancelar Aplicar Ayuda

Taller de Base de Datos

Listo!!!! Eso es todo para crear un ODBC de MySQL.

Las líneas necesarias en Java para poder conectarse son las siguientes:

Se crea una variable URL donde **jdbc:odbc:** es necesario para indicarle a Java que se creará un puente JDBC - ODBC, seguidamente el nombre del origen de datos. (Mirar el paso 6 descrito anteriormente)

```
private static String URL = "jdbc:odbc:mysql";
```

Se declara también la conexión

```
private Connection conexion = null;
```

Ahora, para hacer la conexión basta con poner lo siguiente:

```
conexion = DriverManager.getConnection(URL);
```

Explicación

A continuación se deja un video ilustrativo para entender un poco más el tema.



<https://youtu.be/E46sqilwyCg>

Taller de Base de Datos

Conclusión

Un trigger es un disparador, como un proceso automático integrado en la BD que se activa cuando se realiza una operación concreta a la que queremos controlar.

Un trigger te puede ayudar para realizar operaciones automáticas cada vez que se haga una inserción o modificación en una tabla de la base de datos. Por ejemplo, tienes una tabla con los accesos que se hacen a un museo con un campo que dice si es un acceso para entrar o para salir, pues puedes crear un trigger que cada vez que se inserte en esta tabla te modifique un contador de otra tabla para saber cuánta gente hay dentro en cada momento.

El trigger o disparador no es más que una pequeña instrucción o rutina que se dispara o ejecuta ante alguna operación (insert, update, o delete) en algún momento (before, after) sobre una tabla. ¿Qué rutina? La que se indique en el cuerpo de dicho disparador. Esta rutina contendrá, obviamente, instrucciones SQL a ejecutar. Si bien existe un estándar, cada motor añade sus propias cláusulas y tiene su propio manejo de triggers.

Es decir que cada vez que se inserte, modifique, o elimine algún elemento en una tabla, se ejecutarán las rutinas empleadas. Esto nos lleva a que hay 6 tipos de triggers: after insert, before insert, after update, before update, after delete, before delete. Dependiendo de las necesidades, y la rutina se debe optar por uno o otro contexto.

Naturalmente, los triggers son opcionales. Pero es muy común utilizarlo, sobre todo en bases de datos dedicadas a la gestión como ventas y facturación y controles de inventarios.

Conectar tus aplicaciones Java a un servidor de bases de datos es imprescindible para la funcionalidad de una solución de Software. Para ello se necesita de una interfaz que proporcione las clases necesarias para gestionar una conexión.

Una API es un conjunto de interfaces relacionadas que permiten realizar una conexión de un servidor de bases de datos a una aplicación Java. Normalmente se le denomina JDBC (Java Database Connectivity). La idea es usar un Driver que gestione el acceso a un servidor Mysql, y así abrir la conexión con la intención de ejecutar comandos en la 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

Sencilla página web que da un resumen del trabajo realizado y su seguimiento.

http://reynald0.github.io/taller_bd/

URL del proyecto JAVA para realizar la conexión con JDBC y ODBC

<https://svn.code.sf.net/p/tap-itvh/code/trunk/Nautica7/>

Taller de Base de Datos

Bibliografía

Javier Revelo. (12 de Junio de 2014). Crear procedimientos almacenados en MySQL. 18/05/2016, de Hermosa Programación Sitio web: <http://www.hermosaprogramacion.com/2014/06/mysql-procedure-show-create/>

GUEBS. (Desconocido). Capítulo 19. Procedimientos almacenados y funciones. 18/05/2016, de Manuales GUEBS Sitio web: <http://manuales.guebs.com/mysql-5.0/stored-procedures.html>

GUEBS. (Desconocido). Capítulo 20. Disparadores (triggers). 18/05/2016, de Manuales GUEBS Sitio web: cna1-v5.blogspot.mx/2014/03/ccna-1-capitulo-2-v50-exam-respuestas.html

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

James Revelo. (28 Junio de 2014). Crear Triggers en MySQL. 20/05/2016, de Hermosa Programación Sitio web: <http://www.hermosaprogramacion.com/2014/06/mysql-trigger/>

James Revelo. (9 de Julio de 2014). ¿Cómo conectar MySQL con Java?. 14/05/2016, de Hermosa Programación Sitio web: <http://www.hermosaprogramacion.com/2014/07/mysql-java-conectar-como/>