Seminario 1 - Acceso a bases de datos

Driver ODBC

Para gestionar la información que hay almacenada en una base de datos desde un programa que estamos implementando, necesitamos acceder desde dicho programa a la BD.

Una forma de hacerlo es a través de un driver; un componente software que permite la comunicación entre nuestra aplicación y el SGBD de nuestra base de datos. Nosotros hemos instalado el driver ODBC, desarrollado por SQL Access Group. ODBC permite acceder a cualquier dato desde cualquier aplicación, sin importar el SGBD que almacene los datos. Introduce una capa entre la aplicación y el SGBD cuyo propósito es el de traducir las consultas de datos de las aplicación en comandos que el SGBD entienda. Para que esto funcione, la aplicación debe ser capaz de producir comandos ODBC y el SGBD debe ser capaz de responder a ellos. Para ello, hemos importado en nuestra aplicación la librería pyodbc, dado que la hemos implementado en lenguaje Python.

Conexión a la base de datos

Para conectarnos a la base de datos que hemos creado con SQL_Developer, desde la aplicación, en primer lugar, debemos de crear variables para las credenciales de conexión:

> SERVER = "Dirección del servidor"
>
> PORT = "Nombre del puerto a través del cual se va a realizar la comunicación"
>
> DATABASE = "Nombre de la base de datos"
>
> USERNAME = "Nombre de usuario de la cuenta de SLQ_Developer"
>
> PASSWORD = "Constraseña de la cuenta de SLQ_Developer"

A continuación se crea una variable de cadena de conexión mediante la interpolación de cadenas:

> connectionString = f'DRIVER={{Oracle in instantclient_21_11}};SERVER={SERVER};DATABASE={DATABASE};UID={USERNAME};PWD={PASSWORD}'

donde DRIVER es el nombre del driver que hemos descargado e instalado para poder acceder a la base de datos.

Para conectarse finalmente a la base de datos, se usa la función pyodbc.connect:

> conn = pyodbc.connect(connectionString).

Para cerrar la conexión, simplemente, se hace

> conn.close(), donde conn es una conexión.

Clase Controlador

Se ha decidido desarrollar la aplicación siguiendo un diseño arquitectónico llamado Modelo-Vista-Controlador, que se ha utilizado con objeto de organizar el codigo de la aplicación. Actúa como intermediario entre el Modelo (parte de la aplicación que se encarga de interactuar internamente con la base de datos) y la Vista (parte de la aplicación que se encarga de la interfaz gráfica que los usuarios ven y con la que interactúan). El controlador captura las acciones del usuario en la interfaz, para procesar esas acciones y comunicase con el modelo para realizar las operaciones necesarias en la base de datos. Luego, actualiza l Vista para mostrar los resultadoa al usuario.

En esta clase se han implementado toda la funcionalidad (todos los métodos) de la aplicación; todas las acciones que se realizan de forma interna al pulsar los botones de los menus que se han implementado.

Así, al crear una instancia de tipo controlador en el programa principal (main), se crea la conexión con la base de datos, se crear un cursor (estructura de control utilizada para el recorrido de los registros del resultado de una consulta a una base de datos. Se utiliza para el procesamiento individual de las filas devueltas por el sistema gestor de base de datos para una consulta), se crean los

menus ("crear" como variables) e instancias de clases creadas por nosotros que permiten capturar a través de la interfaz los datos que introuce el usuario de un pedido y los detalles de un pedido, y permite mostrar el contenido de la base de datos.

> **Nota:** Para la implementación de la interfaz de usuario, los menús, y todas las operaciones que implican interacción con el usuario, se ha utilizado la librería de python ***Tkinter***.

Implementación de los Menús

Los menús se han implementado como dos clases distintas, con un constructor, al que se le habrá de pasar como argumento una instancia de la clase controlador, para poder darle funcionalidad a los botones.

Se ha utilizado la librería Tkinter de python, como ya se ha indicado.

El menú se presenta en pantalla al llamar a la función mostrar. Dicha función crea una ventana (con el constructor de tkinter) con unas dimensiones concretas y a la que se le añaden botones (tk.Button) con determinadas características y a los que se le da una funcionalidad cuando son pulsados (command=funcion, donde funcion es accedida por medio del controlador que se le ha pasado como argumento al menu).

Para que la ventana con los botones aparezca en pantalla, se llama a la función mainloop() de tk.

Descripción de las funciones del controlador

añadir_Pedido:

Captura de la interfaz los datos introducidos por el usuario. Para ello, llama a la función capturaPedido(), de la clase capturaPedido, que crea una ventana con tres Labels (un objeto tk.Label es un widget utilizado para mostrar texto o imágenes en una interfaz gráfica), empaquetados (se usa el método pack() que empaqueta el Label en la ventana, lo que lo hace visible en la interfaz de usuario, posicionándolo según los parámetros que se le pasen al pack). Con la función tk.Entry, se crea un cuadro de inserción para el dato a recibir por parte del usuario y guarda lo que este escriba en una variable concreta. Una vez el usuario ha terminado de insertar los datos, el botón añadir llama a la función button_clicked que guarda

en un vector los datos introducidos por el usuario y se cierra la ventana.

Una vez guardados los datos introducidos por el usuario en un vector correspondiente, se hace uso del cursor para insertar la tupla de datos en la tabla de Pedido de nuestra base de datos. Para ello se ha seguido una estructura try...except para manejar excepciones en caso de que se produzca algún error de tipo pyodbc.Error. En caso de que haya algún error en la inserción, se vuelve al menú 1. Si no hay ninguna excepción y la inserción en la base de datos se ha realizado correctamente, se ejecuta el bloque else. En este bloque se realiza una operación de "SAVEPOINT" en la base de datos; se crea un punto de guardado en la transacción de la base de datos que se llama pedido (el savepoint se llama "pedido"). De esta forma, si se hiciera un ROLLBACK to pedido, se eliminarían todas las operaciones posteriores a la inserción de pedido en la tabla PEDIDO. Finalmente, cuando la inserción se ha hecho correctamente, se muestra el segundo menú.

añadirDetallePedido

En este método se capturan los detalles de un pedido concreto: código del producto y cantidad, de la misma forma que en el método pedido. Se realiza la inserción del tupla de la misma forma que antes. No obstante se comprueba que la cantidad del producto en stock sea superior a la del pedido (se ha obtenido por consulta la cantidad del producto Cproducto y se ha recuperado la fila de Cproducto con fetchone() (método del objeto de cursor que se utiliza para recuperar una sola fila de resultados de la consulta más reciente que se ejecutó utilizando ese cursor)). Si no hay ningún error, entonces se resta a la cantidad de producto que hay en stock, la cantidad de producto del pedido, y se muestra el contenido de la base de datos. Finalmente, se vuelve al menú 2.

consultarDatosBD

Este método muestra una ventana con las tres tablas que tenemos en nuestra base de datos y el contenido de las mismas. Para ello, en primer lugar, capturamos los datos de las tablas: si queremos capturar los datos de la tabla STOCK, con el cursor ejecutamos la orden SQL que nos permite consultar el contenido de la tabla STOCK. A continuación hacemos fetchall() que lo que hace es recuperar todas las filas de la consulta más reciente que se ejecutó

utilizando el cursor y las guarda en la variable stock. Finalmente, modificamos la lista stock, convirtiendo todos los valores obtenidos en cadenas de texto. Para cada tupla t de stock se itera sobre los elementos item dentro de esa tupla, cada uno de los cuales se convierte en una cadena de texto (str(item)). Se crea así una nueva tupla con los elementos convertidos y se agrega a la lista resultante.

Hacemos este proceso para las tres tablas.

Una vez guardado el contenido de las tablas en listas de string, se llama al método que se encargará de graficar el contenido de dichas variables: pintaConsultas. Dicho método utiliza el widget de la biblioteca tkinter, ttk.Treeview, que permite crear tablas con encabezados. Para ello, se definen las columnas (tree.column), se crean los encabezado (tree.heading) y se van insertando (con tree.insert) por filas las tuplas de la lista que corresponda (stock, pedido, detallespedido).

confirmarCambios

Se cierra el menu2, se hace un commit y se muevle a mostrar el menu1.

eliminarDetalles

Se hace un ROLLBACK TO pedido, el savepoint que se había establecido al insertar los datos de alta del pedido (solo queremos eliminar los detalles del pedido (que es lo que se hace después de dar de alta el pedido)). A continuación se muestra la base de datos.

eliminarDetallesYPedido

Se hace un ROLLBACK (eliminándose todos los datos de pedido y sus detalles). Se cierra el menu2, se comprueba, consultado la base de datos y s abre el menu1.

borradoCreacion

Consiste en una serie de instrucciones SQL que ejecuta el cursor que consisten en eliminación de tablas (DROP), creación de tablas (CREATE TABLE...), inserción de tuplas en las tablas (INSERT INTO ...), y guardado de cambios (COMMIT).

*__cerrarConexion__

Se cierra el cursor y la conexión con close().

Programa principal (main)

En el programa principal, simplemente habrá que crear una instancia de tipo controlador, que se encargará de mostrar el menu1 (a través del menu1, en contreto, tras dar de alta un pedido, se muestra el menú 2) y de cerrar la conexión con la base de datos (crear la conexión lo hace en su constructor).