PRÁCTICA 1

BASES DE DATOS

Curso 2022/2023

CREACIÓN Y CONSULTAS A UNA BASE DE DATOS USANDO JAVA

1. Objetivos Generales

- Crear una base de datos en base a un diagrama E-R e insertar el contenido mediante la creación de un programa inicial de carga de datos.
- Acceder a una Base de Datos y realizar consultas SQL mediante un programa implementado en código Java.
- Manejar el conector JDBC (Java DataBase Conectivity), librería de clases para el acceso a un SGBD y para la realización de consultas SQL desde Java.

2. Objetivos específicos

El alumno será capaz de:

- Realizar conexiones a un SGBD desde un programa Java.
- Realizar consultas simples y complejas SQL manejando las clases adecuadas del conector JDBC.
- Tratamiento de los resultados de las consultas SQL dentro del código Java.
- Manejo de las excepciones en la gestión de consultas.
- Implementación de un sistema basado en el concepto de transacción usando código Java.

3. Práctica a realizar

Se desea realizar una aplicación Java que gestione la información de una base de datos que almacena información sobre una cadena de panaderías. En concreto, se almacena información sobre los locales de las panaderías, los empleados que han trabajado en cada local a lo largo del tiempo y los productos que se venden en cada local, con los ingredientes de los mismos y el modo de cocinado de cada ingrediente en cada producto.

La Figura 1 muestra el diagrama E-R que modela la base de datos mencionada.

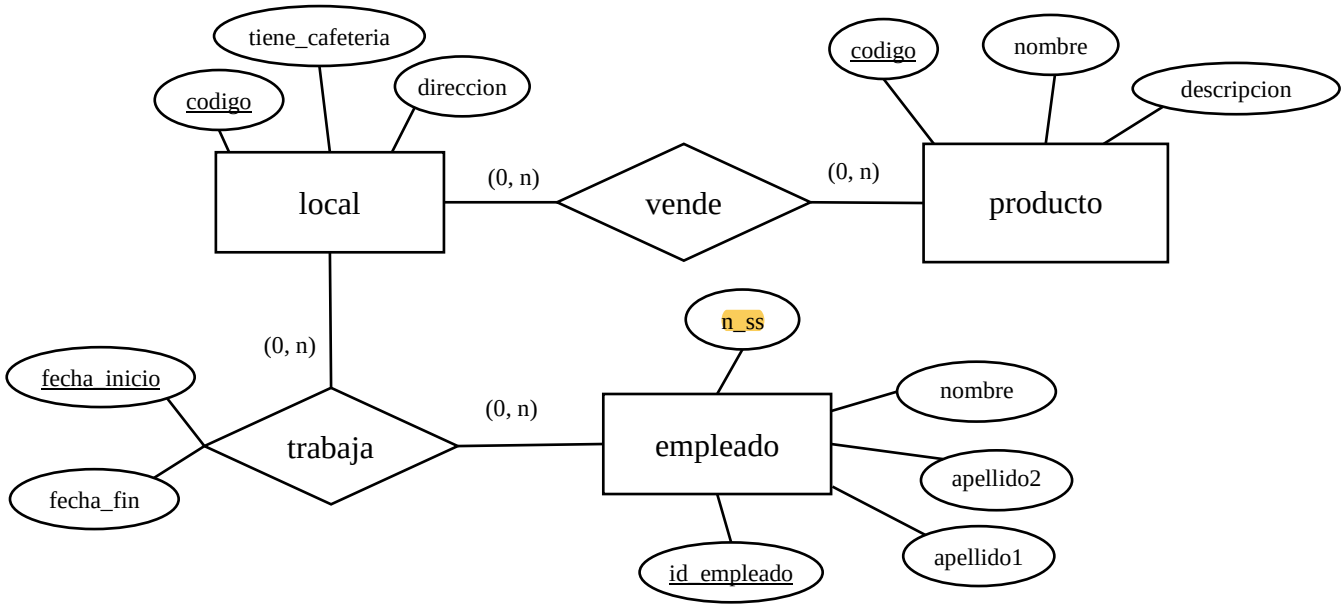

Figura 1. Diagrama entidad relación de la base de datos "Panaderias".

Junto a este enunciado, encontrarás los siguientes elementos:

- Clases Java: clases del paquete Java panaderías. Todas las clases serán de este paquete y no se debe modificar el paquete en ninguna de ellas. Estas clases contendrán la definición de algunos atributos necesarios (podría ser necesario añadir más), constantes y las cabeceras de los métodos que se deben implementar.

  Importante: queda prohibido modificar la definición de cualquier método ya existente en el fichero. Todos ellos deben hacer un correcto tratamiento de las excepciones. Si se necesita crear algún método adicional, éste debe ser privado.

- Ficheros CSV: ficheros CSV que contienen datos de ejemplo para cargar en cada una de las tablas que se generan sin que se produzcan errores. En las pruebas que se realicen en la corrección los ficheros CSV podrían ser cambiados y que éstos contengan errores de integridad de los datos (PK duplicadas, FK que hagan referencia a PK que no existan). Puede suponerse que la primera línea siempre contiene las cabeceras, que el orden de los campos siempre será el dado en estos ficheros de ejemplo y cada campo se separa del siguiente mediante un punto y coma (;).

Con todo ello, se deben implementar un sistema Object-Relation Mapping (ORM), de forma que exista una clase por tabla y que cada objeto de la clase se corresponda con una fila de la tabla. Cada vez que se modifique el valor de un atributo de un objeto, éste debe ser actualizado en su fila correspondiente y cuando se solicite el valor de un atributo de un objeto, éste debe ser actualizado desde la base de datos.

En todos los casos se prohíbe modificar las cabeceras de los métodos que se proporcionan y, especialmente, deben tratarse las excepciones en los mismos, quedando prohibido lanzaras hacia arriba.

Para realizar esta implementación, se pide realizar las siguientes actividades en cada sesión práctica:

Primera sesión práctica

En esta primera sesión se desarrollará una clase que provea una interfaz simplificada para la interacción con la base de datos por parte de los objetos que representen cada una de las filas. Para ello se proporciona el fichero DBConnection.java, en el que se definen:

- Tres constantes que se usarán como valores “centinela” para representar valores nulos en algunas situaciones.
- Cuatro variables de clase para almacenar información relativa a la conexión.
- Un constructor para almacenar los parámetros de la conexión.
- Siete métodos para implementar la funcionalidad.

En concreto en esta sesión, se pide implementar el constructor y los métodos de acuerdo con la siguiente descripción:

1. `DBConnection`: constructor de la clase que debe inicializar los valores de las variables “user”, “pass” y “url” (ésta última debe ser algo del tipo “jdbc:mysql:// server:3306/database”). La implementación del constructor no tiene una puntuación como tal, pero es condición necesaria para seguir trabajando.

2. `connect()` [1 punto]: este método debe implementar la apertura de la conexión con la base de datos. El método debe devolver *true* si la conexión está disponible para su uso tras llamarlo (aunque ya lo estuviera previamente) y *false* si la conexión no está disponible. Se debe llamar a este método cuando se vayan a ejecutar operaciones contra la base de datos en otras partes del código. Nunca deben abrirse varias conexiones.
3. `close()` [1 punto]: este método debe implementar el cierre de la conexión con la base de datos. El método debe devolver *true* si se ejecuta sin errores y *false* si ocurre alguna excepción. Sólo debería ser llamado desde el programa principal para terminar la conexión.
4. `update(String sql)` [1.5 puntos]: este método debe ejecutar el comando SQL (que modifica el contenido de la base de datos y que no tiene parámetros) pasado como parámetro. Debe retornar -1 en caso de que se produzca alguna excepción y el número de filas afectadas en caso contrario.
5. `update(String sql, ArrayList<Object> a)` [2 puntos]: este método debe ejecutar el comando SQL (que modifica el contenido de la base de datos y que tiene parámetros) pasado como parámetro. Debe retornar -1 en caso de que se produzca alguna excepción y el número de filas afectadas en caso contrario.

   Los parámetros se pasan en un ArrayList de Object, debiendo llamar al método set correspondiente de la clase PreparedStatement para cada uno de ellos y hacerles un casting adecuado. Es posible obtener un String con el nombre de la clase a la que un objeto `element` pertenece con `element.getClass().getName()`. En el caso de tipos primitivos y String, el nombre de la clase retornada es "java.lang.Integer", "java.lang.Float", "java.lang.String", etc.

   Los valores nulos son siempre pasados como instancias de su tipo o clase correspondiente, haciendo uso de los valores centinela definidos (nunca como null) y debe hacer uso del siguiente método para establecerlos:

   ```
   PreparedStatement.setNull(int index, int sqlType);
   ```

   Como puede verse, el tipo es necesario en la llamada a setNull, por lo que este es el motivo del uso de los valores centinela. Los tipos pueden especificarse como constantes de `java.sql.Types`: `Types.VARCHAR`, `Types.INTEGER`, etc.
6. `query(String sql)` [1.5 puntos]: este método debe ejecutar la consulta SQL (que no modifica el contenido de la base de datos y que no tiene parámetros) pasada como parámetro. Debe retornar null en caso de que se produzca alguna excepción y el ResultSet correspondiente en caso contrario.
7. `query(String sql, ArrayList<Object> a)` [2 puntos]: este método debe ejecutar la consulta SQL (que no modifica el contenido de la base de datos y que tiene parámetros) pasada como parámetro. Debe retornar null en caso de que se produzca alguna excepción y el ResultSet correspondiente en caso contrario. Debe considerarse las mismas indicaciones dadas en el apartado 5.
8. `tableExists(String tableName)` [1.5 puntos]: este método debe retornar true si la tabla cuyo nombre se pasa como parámetro existe en la base de datos y falso en caso contrario. Puede ser de utilidad el comando SQL "`SHOW TABLES`", que retorna las tablas existentes en la base de datos como si fuera una consulta SELECT.

Segunda sesión práctica

En esta segunda sesión se desarrollarán las clases que representan a las tablas provenientes de entidades en el diagrama Entidad-Relación: Empleado, Local y Producto. Para ello, se proporcionan los esqueletos de dichas clases, así como el código de una clase abstracta de la que todas heredan y que no hay que modificar: DBTable. En esta clase abstracta se definen dos atributos: un objeto de la clase DBConnection desarrollada en la primera sesión y un booleano que indica si el objeto debe sincronizarse con la base de datos o no.

En cada una de las clases se pide implementar los siguientes métodos, siempre haciendo uso de las funcionalidades de BDConnection (no se permite crear nuevas conexiones, Statements o PreparedStatements, aunque sí puede ser necesario definir ResultSet para recoger datos):

1. `createTable()` [1 punto]: ejecuta el comando para crear la tabla correspondiente. Retorna true si el comando se ejecuta sin errores y false en caso contrario (intentar crear una tabla ya existente debe considerarse un error y, por tanto, retornar false).
2. `insertEntry()` [1 punto]: ejecuta el INSERT el elemento actual en la base de datos. Recordatorio: si alguno de los datos es nulo, debe pasarse el valor centinela correspondiente. Retorna true si el elemento se ha insertado y false en caso contrario.
3. `updateEntry()` [1 punto]: ejecuta el UPDATE del elemento actual en la base de datos. Deben actualizarse en la base de datos todos los campos que no formen parte de la clave primaria, usando los que forman parte de ésta para filtrar el elemento a actualizar. Retorna true si el elemento se ha actualizado y false en caso contrario.
4. `deleteEntry()` [1 punto]: ejecuta el DELETE del elemento actual en la base de datos. Debe filtrarse por los valores de la clave primaria para localizar el elemento a borrar. Retorna true si el elemento se ha borrado y false en caso contrario.
5. `getEntryChanges()` [1 punto]: ejecuta el SELECT para recuperar los datos almacenados en la base de datos sobre el elemento actual. Deben actualizarse en el objeto todos los atributos que no formen parte de la clave primaria, usando los que forman parte de ésta para filtrar el elemento a buscar en la base de datos. No retorna nada.
6. Constructores de la clase [1.5 puntos]: por cada clase deben implementarse dos constructores. Uno de ellos inicializa solamente los valores de la clave primaria (poniendo a nulo el resto) y el otro todos los atributos del objeto. Siempre que se cree un elemento, si la sincronización con la base de datos está activada, debe crearse la tabla correspondiente si no existe previamente e insertarse el elemento. De no poder insertar el elemento, deben ponerse a nulo también los elementos que forman parte de la clave primaria y desactivar la sincronización con la base de datos.
7. Getters de los atributos [1.25 puntos]: deben implementarse los getters de todos los atributos, actualizando los valores de los todos los atributos del objeto previamente desde la base de datos si la sincronización está activada.
8. Setters de los atributos [1.25 puntos]: deben implementarse los setters de todos los atributos que no formen parte de la clase primaria. Si la sincronización está activada, antes de realizar la modificación, deben actualizarse los valores de todos los atributos del objeto desde la base de datos y, tras la modificación de acuerdo al parámetro proporcionado, actualizar el contenido de la base de datos.
9. `destroy()` [1 punto]: borra el elemento de la base de datos si la sincronización está activada, pone a nulo (o valores centinela para tipos básicos) todos los atributos del objeto y desactiva la sincronización. No retorna nada.

Tercera sesión práctica

En esta sesión se pide implementar las clases que mapean con las tablas generadas por las relaciones (Trabaja y Vende), con los mismos pesos que los expuestos en la segunda sesión.

Por otra parte, se pide implementar los métodos estáticos de la clase DataManager, que sirve para cargar los datos que existen en la base de datos en un ArrayList de elementos de la clase que mapea con la tabla y para leer datos de un fichero CSV, cargándolos tanto en un ArrayList como en la tabla que mapea con cada clase. En concreto se pide implementar los siguientes métodos:

1. `getEmpleadosFromDB(DBConnection conn, boolean sync)` [1 punto]: el método debe obtener todos los datos de la tabla "empleado", creando un objeto de la clase Empleado por cada fila y añadiendo cada uno de ellos a un ArrayList, que es lo que el método retorna. Si la tabla existe pero está vacía, se debe retornar un ArrayList vacío, mientras que si se produce alguna excepción se debe retornar null.
   Dado que se va a intentar crear objetivos de la clase Empleado con valores de clave primaria que ya existen en filas de la tabla correspondiente (justamente es lo que se pretende), es necesario crear los objetos sin sincronización con la base de datos en primera instancia y activar su sincronización tras la creación con el método `setSync(true)`.
2. `getEmpleadosFromCSV(String filename, DBConnection conn, boolean sync)` [1 punto]: el método irá leyendo cada línea del CSV y creando objetos de la clase Empleado con dichos datos, con la sincronización indicada en el parámetro `sync` (si está activada, se irán insertando las filas correspondientes). En el fichero CSV siempre habrá una línea de cabecera con los nombres de las columnas, pero dichas columnas aparecen siempre según el mismo orden, tal y como se define en los ficheros de ejemplo proporcionados, por lo que no es necesario controlar este aspecto. Las columnas se separan unas de otras mediante un punto y coma (;). Sí podrían aparecer columnas con datos nulos (vacíos). El método debe retornar el ArrayList con las instancias de Empleado leídas (y, en su caso, insertadas en la base de datos). Si se produce una excepción debe retornarse null.
3. `getLocalesFromDB(DBConnection conn, boolean sync)` [1 punto]: análogo al punto 1, pero sobre la clase Local.
4. `getLocalesFromCSV(String filename, DBConnection conn, boolean sync)` [1 punto]: análogo al punto 2, pero sobre la clase Local.
5. `getProductosFromDB(DBConnection conn, boolean sync)` [1 punto]: análogo al punto 1, pero sobre la clase Producto.
6. `getProductosFromCSV(String filename, DBConnection conn, boolean sync)` [1 punto]: análogo al punto 2, pero sobre la clase Producto.
7. `getTrabajaFromDB(DBConnection conn, boolean sync)` [1 punto]: análogo al punto 1, pero sobre la clase Trabaja.
8. `getTrabajaFromCSV(String filename, DBConnection conn, boolean sync)` [1 punto]: análogo al punto 2, pero sobre la clase Trabaja.
9. `getVendeFromDB(DBConnection conn, boolean sync)` [1 punto]: análogo al punto 1, pero sobre la clase Vende.
10. `getVendeFromCSV(String filename, DBConnection conn, boolean sync)` [1 punto]: análogo al punto 2, pero sobre la clase Vende.

Cuarta sesión práctica

Esta sesión será de recuperación para los estudiantes que no hubieran acabado los métodos de las sesiones anteriores.

El programa debe conectarse cuando se vaya a realizar una consulta/operación contra la BD por primera vez (no al arrancar el programa) y no debe desconectarse hasta que el programa finalice.

El mapeo de las clases con las sesiones prácticas es el siguiente:

- Clase 23 marzo: desarrollo de la primera sesión práctica.
- Clase 27 marzo: corrección primera sesión práctica + desarrollo segunda sesión práctica.
- Clase 30 marzo: corrección segunda sesión práctica + desarrollo tercera sesión práctica.
- Clase 13 abril: corrección tercera sesión práctica + repesca sesiones 1 y 2.

4. Evaluación y otras indicaciones

Es importante que se cumplan los requisitos establecidos en la práctica, incluyendo:

- El método connect() debe ser el encargado de: cargar el driver y realizar la conexión. Es obligatorio que se implemente esta funcionalidad dentro de dicho método. Parámetros:
    - Servidor: localhost:3306
    - Usuario: panaderia_user
    - Password: panaderia_pass
    - Nombre base de datos: panaderías

- Todos los identificadores deben almacenarse como tipo INT, la columna "tiene_cafeteria" de la tabla "local" debe ser también un INT que tome los valores 0 (falso) o 1 (verdadero), las fechas deben almacenarse como tipo DATE y el resto de campos (de texto) se almacenarán como VARCHAR(100).

- Que sean los métodos ya dados los que, al menos, se encarguen de proporcionar el resultado final. El estudiante puede generar otros métodos adicionales si lo considera relevante, pero el método asignado a cada ejercicio debe devolver el resultado. No se pueden alterar los nombres de los ficheros, paquetes y clases que se proporcionan.

- Cada clase de evaluará de 0 a 10 puntos mediante un corrector automático. Esto quiere decir que los métodos deben funcionar perfectamente y todos los formatos deben ser acordes al presente guion. A la hora de la corrección se usarán tanto los datos especificados aquí, como otros datos para la comprobación de la corrección de las soluciones.

- Cada clase pedida se evaluará en la sesión siguiente a la que se propone, según el calendario expuesto anteriormente. Si un método pasa la prueba a la primera (en la primera ronda de corrección de la clase), obtendrá el 100% de su calificación, si la pasa a la segunda (segunda ronda de corrección de la clase) obtendrá el 75% de la misma y si

la pasa a la tercera (en la repesca) obtendrá el 60% de la misma. Si no la pasa en esos tres intentos, la calificación en el método será de 0 puntos. Nótese que por la distribución de clases la tercera entrega práctica solo da opción a las posibilidades del 100% o 75%.

- El peso de cada clase en la nota final de la práctica es el siguiente:
  - DBConnection: 20%
  - Empleado, Local, Producto, Trabaja: 15% cada una
  - Vende: 10%
  - DataManager: 20%

- Esta corrección automática dará una nota preliminar de la práctica, que puede ser inferior si se dan los supuestos que se enumeran a continuación, y que se verificarán una vez sea entregada la práctica en la fecha propuesta.
  - **Limpieza y claridad del código** que, además, debe gestionar correctamente los errores, estar optimización, contener comentarios, etc.: no cumplir con estos aspectos puede suponer la deducción de hasta 1.5 puntos.
  - **Fichero entregado con nombre incorrecto o entrega de ficheros adicionales:** -0.5 puntos.
  - **Conexión con un usuario que no sea el dado:** -2 puntos (si es con root), -1 punto (si es con otra combinación de usuario/contraseña inválida).
  - **Realización de más de una conexión/realización de conexión en momento inoportuno:** -1 punto.
  - **No usarse los cierres de las estructuras necesarias:** -0.5 puntos.
  - **No usar PreparedStatement cuando haya parámetro o usarlo incorrectamente (usando concatenaciones):** -3 puntos.
  - **Creación de tablas sin claves foráneas:** -2 puntos.
  - **No identificar correctamente los tipos de errores:** -0.5 puntos.
  - **Lanzamiento de las excepciones hacia arriba:** para esto es necesario modificar las cabeceras de los métodos y supone la calificación automática con un 0.
  - **Usar clases creadas por el grupo y que no sean inner.** El código no funcionará, lo que supone una calificación automática con un 0.
  - Otros supuestos: Los profesores pueden aplicar otras penalizaciones en caso de encontrar errores diferentes a los aquí descritos. Quedará a discreción del profesorado la penalización a aplicar en cada caso.

- La calificación final de la práctica no superará en ningún caso los 10 puntos.

- Si un grupo no asiste a las sesiones prácticas podrá entregar la práctica completa hasta el 16 de abril a las 23:59, pero en ese caso sólo se permitirá un intento en el corrector automático y se aplicará un factor de penalización.

- Dada la naturaleza de la práctica, la asistencia a las sesiones es obligatoria por todos los integrantes del grupo, salvo en la última sesión. Las faltas no justificadas implicarán un 0 en la nota de las clases a desarrollar en esa sesión. En el caso de faltas justificadas, la nota final (individual del miembro del grupo que falta) se multiplicará por los siguientes factores correctores:
  - 1 falta justificada: sin penalización
  - 2 faltas justificadas: factor de 0.5
  - 3 faltas justificadas: factor de 0.0

- Los profesores se reservan el derecho de citar a cualquier grupo a defender la práctica si lo considerara necesario.

5. Entrega de la Práctica y resolución de dudas

El grupo de prácticas deberá **entregar mediante la plataforma Moodle en una tarea habilitada para ello un único fichero comprimido (*.zip, *.rar) cuyo nombre sea (Grupo_N_JDBC.rar) (donde N es el número de grupo correspondiente) que contenga:**

1. El paquete *panaderias* con la implementación de las clases solicitadas.
2. Opcionalmente un documento en PDF que contenga comentarios acerca de los problemas surgidos al hacer la práctica o cualquier otro comentario que los alumnos estimen oportuno.

La práctica debe ser entregada por **sólo uno de los miembros del grupo** (preferiblemente el representante).

La fecha límite para la entrega de esta práctica es el viernes **16 de abril de 2023 a las 23:55**.

6. Resolución de dudas

Las dudas deben plantearse en primer a instancia a través del **foro** habilitado para dicho fin en Moodle. Si fuera necesario concertar una tutoría, debe enviarse un correo electrónico al profesor de tu grupo de prácticas.