PRÁCTICA 2

BASES DE DATOS

Curso 2020/2021

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 series de distintas plataformas de streaming y las valoraciones y comentarios de los usuarios. En concreto, la base de datos donde almacena información sobre las series (con su identificador, título, sinopsis, fecha de estreno e idioma), sus géneros (con su identificador y descripción), sus temporadas (con el número de temporada respecto a la serie, el número de capítulos, el título y la fecha de estreno), los capítulos (con el número de orden del capítulo dentro de la temporada, la fecha de estreno, título y duración) y los usuarios (con su identificador, fotografía y nombre y apellidos).

Las series se componen de temporadas y las mismas de capítulos. Además, una serie puede tener varios géneros. Un usuario puede realizar comentarios en una fecha determinada sobre una serie (pueden ser varios a lo largo del tiempo) y también puede dar una puntuación a un capítulo (también podría dar varias puntuaciones a lo largo del tiempo).

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

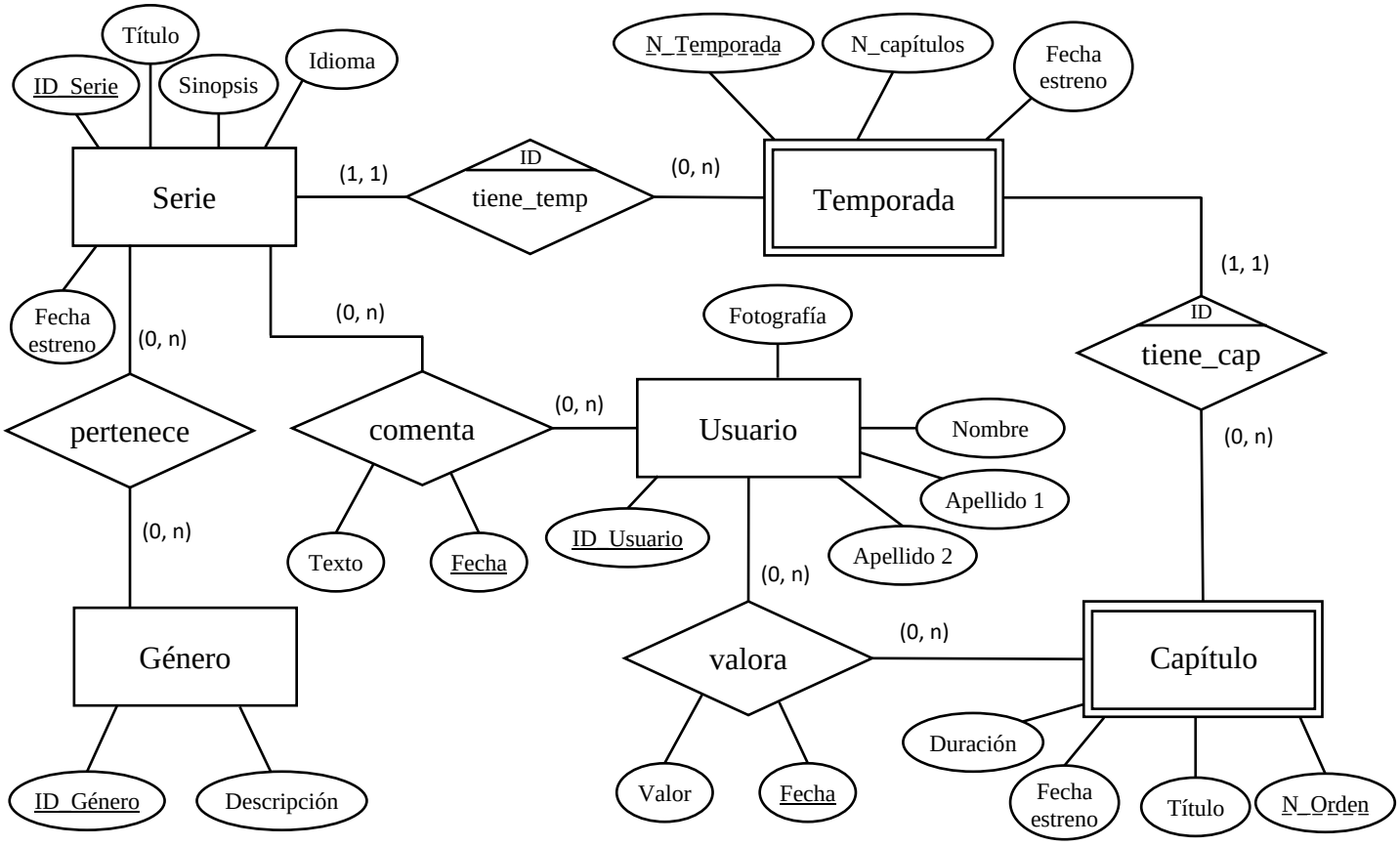

Figura 1. Diagrama entidad relación de la base de datos "Series".

Junto a este enunciado encontrarás cinco elementos:

- *series.sql*: incluye la definición del esquema de la base de datos (y su borrado si ya existe para resetearlo, ¡usar con precaución!), así como la creación de varias tablas y la inserción de datos en ellas. Este script debe importarse o ejecutarse desde el Workbench antes de comenzar a desarrollar la aplicación.
- *series*: paquete de Java que incluye dos clases. La clase *SeriesDatabase* contiene el esqueleto de los métodos a implementar y es el único fichero que debe modificarse. El estudiante debe entregar como resultado de esta práctica únicamente el fichero .java que implementa esta clase, completando con el código todo lo que se pide. Por otra parte, el paquete contiene una clase *Main*, que implementa un menú para probar la funcionalidad de la aplicación.

  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.
- *capitulos.csv* y *valoraciones.csv*: ficheros CSV que incluyen una lista de capítulos y de valoraciones respectivamente. Pueden servir de ejemplo para ser insertados sin que se produzcan errores suponiendo que se ha reseteado previamente la base de datos con el script *series.sql*. La primera línea incluye los nombres de las columnas y cada campo se

separa mediante un punto y coma (;). Son los ficheros cargados al llamar a los métodos desde la clase Main.

- *HomerSimpson.jpg*: fichero JPEG con una foto de ejemplo para el método *setFoto*. Es el fichero usado al llamar a dicho método desde la clase Main.

Con todo ello, se deben implementar los siguientes métodos en Java (además de añadir el código que fuera necesario en cualquier otra parte del fichero):

1. `openConnection` [0.25 puntos]: este método debe implementar la apertura de la conexión con la base de datos. El método devuelve *true* si abre correctamente la conexión y *false* si ninguna conexión es abierta (bien porque no se pueda o porque ya estuviera abierta). Se debe llamar a este método cuando se vaya a realizar la conexión en otras partes del código. Nunca debe abrir varias conexiones.
2. `closeConnection` [0.25 puntos]: este método debe implementar el cierre de la conexión con la base de datos. El método devuelve *true* si se ejecuta sin errores y *false* si ocurre alguna excepción. No es necesario llamar a este método en el código, ya está llamado donde corresponde en la clase *Main*.
3. `createTableCapitulo` [0.5 puntos]: este método debe implementar la creación de la tabla *capitulo*. El método devuelve *false* si la tabla no se ha podido crear (por algún tipo de error o porque ésta ya existía) y *true* si la tabla se ha creado correctamente.

   Prestar atención a cualquier restricción del enunciado. La tabla debe llamarse exactamente *capitulo*. Se deben nombrar las claves foráneas (si las hubiera) usando exactamente el mismo nombre que la primaria a la que referencian.

   El resto de atributos deben llamarse exactamente de la siguiente manera y deben tener los siguientes tipos:

   - n_orden → entero
   - titulo → cadena de longitud variable de hasta 100 caracteres
   - duracion → entero
   - fecha_estreno → fecha
4. `createTableValora` [0.5 puntos]: este método debe implementar la creación de la tabla *valora*. El método devuelve *false* si la tabla no se ha podido crear (por algún tipo de error o porque ésta ya existía) y *true* si la tabla se ha creado correctamente.

   Prestar atención a cualquier restricción del enunciado. La tabla debe llamarse exactamente *valora*. Se deben nombrar las claves foráneas (si las hubiera) usando exactamente el mismo nombre que la primaria a la que referencian.

   El resto de atributos deben llamarse exactamente de la siguiente manera y deben tener los siguientes tipos:

   - fecha → fecha
   - valor → entero
5. `loadCapitulos` [1 punto]: este método debe insertar en la base de datos los capítulos de las temporadas contenidos en el fichero que se pasa como parámetro. El método irá leyendo cada línea del CSV e insertando cada una de ellas (la primera línea contiene los nombres de las columnas de la tabla capitulo y los campos se separan mediante punto y

coma). Obligatoriamente, debe ejecutarse la creación y carga de todos los datos como si fuera una única transacción, de tal forma que cualquier fallo intermedio de lugar a deshacer por completo los cambios anteriores. El método debe retornar la cantidad de elementos insertados en la tabla.

6. `loadValoraciones` [1 punto]: este método debe insertar en la base de datos las valoraciones que cada usuario realiza a cada capítulo y que se encuentran en el fichero *valoraciones.csv*. El método irá leyendo cada línea del CSV e insertando cada una de ellas (la primera línea contiene los nombres de las columnas de la tabla valoraciones y los campos se separan mediante punto y coma). Cada inserción debe ser tratada como una transacción separada del resto. El método debe retornar la cantidad de elementos insertados en la tabla.
7. `catalogo` [1.5 punto]: este método debe consultar en la base de datos la lista de todas las series y la cantidad de episodios de cada temporada (consultada del campo correspondiente en la tabla *temporada*). El método debe retornar un String con dicha información estructurada de la siguiente forma:

   ```
   {nombre_serie_1:[v1,v2,v3],nombre_serie_2:[v1,v2]}
   ```

   Dicho string no debe contener ninguna comilla ni espacios. Entre cada par de corchetes habrá tantos valores como temporadas tenga la serie, indicando cada valor el número de episodios de las temporadas, ordenadas de la que tenga un número de temporada más bajo al más alto (N_Temporada). El orden de las series será creciente según su ID_Serie.

   Si no hay series almacenadas, debe retornarse "`{}`" (sin las comillas). Si una serie no tiene temporadas registradas, para dicha serie se tendrá "`…,nombre_serie:[],…`" (sin las comillas). Si se produjera alguna excepción, el método debe retornar *null*.
8. `noHanComentado` [1.5 puntos]: este método debe retornar una lista con los nombres y apellidos de los usuarios que no han comentado ninguna serie. El método debe retornar un String con dicha información estructurada de la siguiente forma:

   ```
   [nombre_1 apellido1_1 apellido2_1, nombre_2 apellido1_2 apellido2_2, ...]
   ```

   Dicho string no debe contener ninguna comilla, pero debe contener espacios después de un nombre, un primer apellido y una coma (y sólo en esas posiciones). Los elementos deben estar ordenados alfabéticamente por el primer apellido, en caso de empate, por el segundo y, en caso de nuevo empate, por el nombre.

   Si no hay usuarios registrados o todos han realizado algún comentario, debe retornarse "`[]`" (sin las comillas). Si se produjera alguna excepción, el método debe retornar *null*.
9. `mediaGenero` [1.5 puntos]: este método debe consultar en la base de datos la lista de todos los capítulos que pertenezcan a una serie en la que la descripción de al menos uno de sus géneros coincida con el String del parámetro del método. El método debe retornar la media de las valoraciones asociadas a dichos capítulos. Si la descripción del género no existe en la base de datos debe retornarse -1.0. Si se produce alguna excepción, debe retornar -2.0. Si el género existe, pero no hay capítulos para ese género se retornará 0.0.
10. `duracionMedia` [1.5 puntos]: este método debe consultar en la base de datos la lista de todos los capítulos cuya serie esté en un determinado idioma (pasado como parámetro) y que no hayan recibido ninguna valoración por parte de los usuarios. El método debe retornar la duración media de los capítulos que cumplan dichos criterios. Si no hay

capítulos que cumplan con las condiciones, debe retornarse -1.0. Si se produce alguna excepción, debe retornar -2.0.

11. `setFoto` [1 puntos]: el método debe añadir la foto contenida en el fichero cuyo nombre se pasa como parámetro al usuario cuyo primer apellido es "Cabeza" en caso de que no tenga ya una foto. El método debe devolver *true* si inserta la foto y *false* en caso contrario (si, por ejemplo, el usuario no existe, si hay más de un usuario cuyo primer apellido es "Cabeza" o si el usuario ya tiene una foto).

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 (de esto último se encarga ya la clase Main).

4. Consideraciones sobre el paquete Java *series*

El paquete de Java *series* contiene los métodos necesarios para ser ejecutado, ya que la clase *Main* contiene el método *main*, que ejecuta el menú del programa. El menú está preparado para que se ejecuten las diferentes operaciones que se pueden escoger (de la 0 a la 7) y que, tras ello, vuelva a mostrarse el menú. Sólo se sale del menú (y del programa) escogiendo la opción 0, que es la de salir. Está prohibido modificar esta clase.

La clase *SeriesDatabase* contiene las cabeceras de los métodos que se pide implementar, con lo que el estudiante simplemente debe rellenar el código en dichos métodos. Está prohibido modificar la definición de dichos métodos, pero pueden definirse tantos métodos privados y clases inner nuevas como se deseen. El fichero que contiene esta clase es lo único que se debe entregar.

5. Evaluación y otras indicaciones

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

- El método openConnection() 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: series_user
    - Password: series_pass
    - Nombre base de datos: series
- 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.
- El fichero de la clase Java a entregar debe llamarse obligatoriamente "SeriesDatabase.java", tal y como se entrega. Es decir, debe usarse en realidad ese fichero, no se debe crear uno nuevo. No debe cambiarse tampoco el nombre del paquete.

Solo puede entregarse dicho fichero .java. En caso de necesitar crear clases extra para el procesamiento de datos (aunque no se considera necesario), deben hacerse como clases inner[1].

- La práctica se evaluará de 0 a 10 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. Si un método no pasa la prueba automática, se calificará con 0 puntos. Esta corrección automática dará una nota preliminar de la práctica, que puede ser inferior si se dan los supuestos que se enumeran a continuación.

    - **Limpieza y claridad del código** que, además, debe gestionar correctamente de 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.

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

[1] https://www.tutorialspoint.com/java/java_innerclasses.htm

6. Entrega de la Práctica

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 (JDBC.rar o JDBC.zip) que contenga:**

1. Clase SeriesDatabase.java con la implementación solicitada (no cambiar el nombre).
2. Fichero .sql con las consultas que se han desarrollado para los métodos de Java.
3. 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**.

La fecha límite para la entrega de esta práctica es el viernes **28 de mayo de 2021 a las 23:55**.

7. 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 Antonio Jesús Díaz Honrubia (antoniojesus.diaz@upm.es).