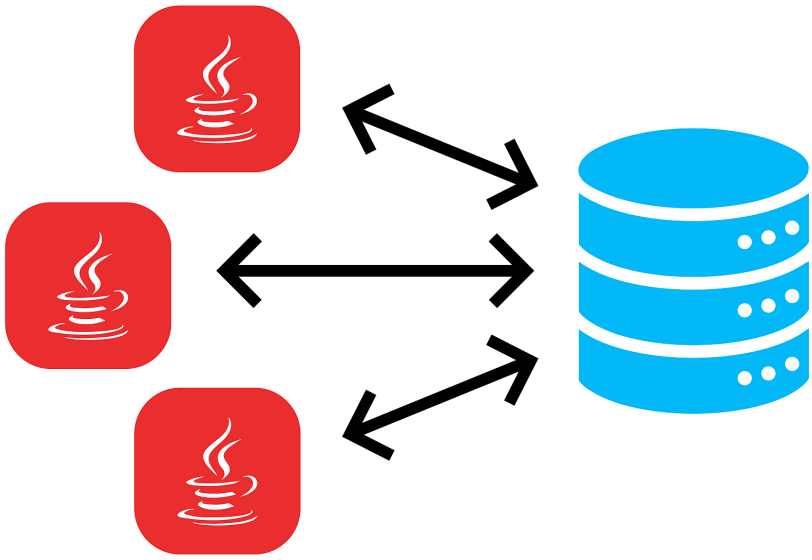
Ejercicios de la UD10



1. Ejercicios

1. Crear base de datos

   Para la realización de los ejercicios deberás crear una base de datos en tu SGBD (MySql) de nombre `pr2425_tuNombre` (en mi caso `pr2425_DavidMartinez`) .

   Para ello deberás de crear un usuario de nombre `tuNombre` con contraseña `1234` y con una base de datos con el mismo nombre:

```
<div style="text-align: center;"><img src="../../img/ud09/phpmyadmin01.png"
alt="phpmyadmin" style="zoom:100%;" /></div>
<div style="text-align: center;"><img src="../../img/ud09/phpmyadmin02.png"
alt="phpmyadmin" style="zoom:100%;" /></div>
<br/><hr>
Descarga el fichero sql [**tablas.sql**](./tablas.sql){:target="_blank"} e
insértalas en tu base de datos. <br />
Para ello, ve a la pestaña de phpMyAdmin `Importa`, selecciona el fichero
descargado anteriormente y ejecútalo.
```

   ???+ question "Siguiendo el ejemplo anterior..."

```
Después de esto, crea un ejercicio de nombre `ListarProveedores.java` en el que se
liste todos los proveedores.
```

2. Crea una aplicación que nos permita gestionar la base de datos network.

   Debe tener un menú desde el que se puedan gestionar (Create, Read, Update, Delete) usuarios, posts y comentarios.

2. Actividades

2.1. _01_GestionEmpleados

Tenemos nuestra base de datos pr_tuNombre que almacena información sobre empleados. La tabla empleados tiene las siguientes columnas:

- id: Identificador único del empleado (entero).
- nombre: Nombre del empleado (cadena de texto).
- salario: Salario del empleado (decimal).

Es escribir un programa Java _01_GestionEmpleados que realice las siguientes operaciones utilizando diferentes tipos de resultado y opciones de concurrencia:

- listarEmpleados (Connection conn): Mostrar en la consola todos los empleados y sus salarios.
- actualizarSalarios (Connection conn): Incrementar el salario de todos los empleados en un 10%.
- eliminarEmpleados (Connection conn): Eliminar todos los empleados cuyo salario sea menor que 3000€.

Importante Consejo: En el main ejecuta por este orden:

- 1º listarEmpleados
- 2º actualizarEmpleados
- 3º listarEmpleados
- 4º eliminarEmpleados
- 5º listarEmpleados

Ampliación Para probar...

Puedes implementar cada operación utilizando un tipo de resultado y opción de concurrencia diferente para familiarizarte con su uso.

No olvides manejar las excepciones SQLException adecuadamente.

Por ejemplo, podrías probarlas siguientes operaciones:

- Lista todos los empleados junto con sus salarios utilizando un ResultSet de tipo TYPE_SCROLL_SENSITIVE y opción de concurrencia CONCUR_READ_ONLY.
- Actualiza los salarios de todos los empleados incrementándolos en un 10% utilizando un ResultSet de tipo TYPE_FORWARD_ONLY y opción de concurrencia CONCUR_UPDATABLE.
- Elimina todos los empleados cuyo salario sea menor a 3000€ utilizando un Statement estándar sin necesidad de un ResultSet.

Importante No olvides:

1. Manejar las excepciones SQLException adecuadamente.
2. Ajustar la cadena de conexión a tu base de datos y reemplazar "usuario" y "contraseña" con las credenciales adecuadas.

2.2. _02_GestionProductos

Supongamos que tienes una base de datos que almacena información sobre productos. La tabla productos tiene las siguientes columnas:

- id: Identificador único del producto (entero).
- nombre: Nombre del producto (cadena de texto).
- precio: Precio del producto (decimal).

Tu tarea es escribir un programa Java _02_GestionProductos que realice las siguientes operaciones utilizando los métodos proporcionados:

1. `mostrarProductosPorPagina()`: Mostrar una página de productos cada vez que el usuario lo solicite. Cada página debe contener 5 productos. Implementa las funciones para mover el cursor a la primera página, página siguiente, página anterior, última página y una página específica utilizando el método absolute(int row).
2. `buscarProductoPorNombre(String nombre)`: Permitir al usuario buscar un producto por su nombre. Utiliza el método relative(int registros) para desplazar el cursor hacia adelante o hacia atrás según la coincidencia del nombre.

2.3. _03_GestionAlumnos

Supongamos que tienes una base de datos que almacena información sobre alumnos. La tabla alumnos tiene las siguientes columnas:

- id: Identificador único del alumno (entero).
- nombre: Nombre del alumno (cadena de texto).
- edad: Edad del alumno (entero).

Tu tarea es escribir un programa Java _03_GestionAlumnos que realice las siguientes operaciones utilizando los métodos proporcionados:

- **Mostrar la información del alumno más joven y más viejo**: Utiliza los métodos first() y last() para mover el cursor a la primera y última fila respectivamente y obtener la información del alumno más joven y más viejo.
- **Desplazarse por los alumnos en orden inverso de edad**: Muestra la información de los alumnos en orden inverso de edad. Utiliza el método previous() para desplazarte hacia atrás a través de los registros.

2.4. _04_GestionLibros

Supongamos que tienes una base de datos que almacena información sobre libros. La tabla libros tiene las siguientes columnas:

- id: Identificador único del libro (entero).
- titulo: Título del libro (cadena de texto).
- autor: Nombre del autor del libro (cadena de texto).
- anio_publicacion: Año de publicación del libro (entero).

Tu tarea es escribir un programa Java _04_GestionLibros que realice las siguientes operaciones utilizando los métodos proporcionados:

- `mostrarLibrosPorDecada(Connection con, int decada)`: Permite al usuario ingresar una década y mostrar todos los libros publicados en esa década.

> **Importante** Sugerencia en este método:
>
> Puedes realizar el método mostrarLibrosPorDecada() de dos formas:
>
> 1. Utiliza el método `createStatemnt()` para crear el Resulset con el atributo ResultSet.TYPE_SCROLL_INSENSITIVE. Utiliza dentro los métodos afterLast() y previous() para mover el cursor al final y luego retroceder, así puedes comenzar desde la última fila.
> 2. Utiliza el método `preparedStatement(sql)` con una consulta en la que se listen los libros comprendidos en una década y ordenados de forma descendente por el anio_publiacion.

`buscarLibroPorAutor(Connection con, String autor)`: Permite al usuario ingresar el nombre de un autor y muestra todos los libros escritos por ese autor.

> **Importante** Sugerencia en este método:
>
> Puedes realizar el método `buscarLibroPorAutor()` de dos formas:
>
> 1. Utiliza el método createStatemnt() para crear el Resulset con el atributo ResultSet.TYPE_SCROLL_INSENSITIVE. Utiliza dentro el método relative(int registros) para desplazarte a través de los registros según las coincidencias del autor.
> 2. Utiliza el método preparedStatement(sql) con una consulta en la que se listen los libros que contengan la cadena autor dentro del campo autor.

2.5. _05_GestionVentas

Supongamos que tienes una base de datos que almacena información sobre ventas. La tabla ventas tiene las siguientes columnas:

- id: Identificador único de la venta (entero).
- producto: Nombre del producto vendido (cadena de texto).
- cantidad: Cantidad de productos vendidos (entero).
- total: Total de la venta (decimal).

Tu tarea es escribir un programa Java _05_GestionVentas que realice las siguientes operaciones utilizando los métodos proporcionados:

- Calcular el total de ventas: Utiliza el método next() para recorrer todas las ventas y sumar los totales para obtener el total general de ventas.
- Buscar ventas por producto: Permite al usuario ingresar el nombre de un producto y muestra todas las ventas asociadas a ese producto. Utiliza el método relative(int registros) para desplazarte a través de los registros según las coincidencias del producto.

2.6. _06_GestionPedidos

Supongamos que tienes una base de datos que almacena información sobre pedidos. La tabla pedidos tiene las siguientes columnas:

- id: Identificador único del pedido (entero).
- cliente: Nombre del cliente que realizó el pedido (cadena de texto).
- producto: Nombre del producto pedido (cadena de texto).
- cantidad: Cantidad del producto solicitada en el pedido (entero).
- fecha: Fecha en que se realizó el pedido (fecha).

Tu tarea es escribir un programa Java _06_GestionPedidos que realice las siguientes operaciones utilizando los métodos proporcionados:

- Listar pedidos por cliente: Permite al usuario ingresar el nombre de un cliente y mostrar todos los pedidos realizados por ese cliente. Utiliza el método relative(int registros) para desplazarte a través de los registros según las coincidencias del cliente.
- Buscar pedidos por fecha: Permite al usuario ingresar una fecha y mostrar todos los pedidos realizados en esa fecha. Utiliza el método afterLast() y previous() para mover el cursor al final y luego retroceder, así puedes comenzar desde la última fila.

2.7. _07_GestionEmpleados (continuación)

Continuando con el ejercicio de gestión de empleados, copia el programa GestionEmpleados, cambia el nombre a _07_gestionEmpleados y agrega algunas funcionalidades adicionales:

- Mostrar información del empleado por ID: Permite al usuario ingresar el ID de un empleado y muestra toda la información relacionada con ese empleado. Utiliza el método absolute(int row) para posicionarte en el registro del empleado especificado.
- Buscar empleados por salario: Permite al usuario ingresar un rango de salarios y mostrar todos los empleados cuyo salario esté dentro de ese rango. Utiliza el método next() para recorrer todas las filas y filtrar los empleados según el criterio de salario.

2.8. _08_GestionEstudiantes

Supongamos que tienes una base de datos que almacena información sobre estudiantes. La tabla estudiantes tiene las siguientes columnas:

- id: Identificador único del estudiante (entero).
  nombre: Nombre del estudiante (cadena de texto).
  edad: Edad del estudiante (entero).
  promedio: Promedio de calificaciones del estudiante (decimal).

Tu tarea es escribir un programa Java _08_GestionEstudiantes que realice las siguientes operaciones utilizando los métodos proporcionados:

- Mostrar la posición actual del estudiante: Muestra la posición del estudiante actual en el conjunto de resultados. Utiliza el método getRow() para obtener el número de registro actual.
- Validar la posición del cursor: Verifica si el cursor está antes del primer registro, en el primer registro, en el último registro o después del último registro. Utiliza los métodos isBeforeFirst(), isFirst(), isLast() e isAfterLast() para realizar estas verificaciones.

2.9. _09_GestionProductos (continuación)

Continuando con el ejercicio de gestión de productos del segundo ejercicio, copia el programa GestionProductos, cambia el nombre a _09_gestionProductos y y agrega algunas funcionalidades adicionales:

- Mostrar el número total de productos: Muestra el número total de productos en la base de datos. Utiliza el método getRow() para obtener el número de registro actual y last() para mover el cursor a la última fila.
- Verificar si hay productos disponibles: Verifica si hay algún producto disponible en la base de datos. Utiliza los métodos isBeforeFirst() e isAfterLast() para determinar si el cursor está antes del primer registro o después del último registro, respectivamente.

2.10. _10_GestionEmpleados (continuación)

Continuando con el ejercicio de gestión de empleados del séptimo ejercicio, copia el programa _10_GestionEmpleados y agrega algunas funcionalidades adicionales:

- Verificar si hay empleados en la base de datos: Verifica si hay algún empleado registrado en la base de datos. Utiliza los métodos isBeforeFirst() e isAfterLast() para determinar si el cursor está antes del primer registro o después del último registro, respectivamente.
- Mostrar el primer empleado: Muestra la información del primer empleado en la base de datos. Utiliza el método first() para mover el cursor al primer registro y luego muestra la información del empleado.

2.11. _11_GestionClientes

Imagina que tienes una base de datos que almacena información sobre clientes. La tabla clientes tiene las siguientes columnas:

- id: Identificador único del cliente (entero).
- nombre: Nombre del cliente (cadena de texto).
- correo: Correo electrónico del cliente (cadena de texto).
- telefono: Número de teléfono del cliente (cadena de texto).

Tu tarea es escribir un programa Java _11_GestionClientes que realice las siguientes operaciones utilizando los métodos proporcionados:

- Mostrar la posición actual del cliente: Muestra la posición actual del cliente en el conjunto de resultados. Utiliza el método getRow() para obtener el número de registro actual.
- Mostrar información del último cliente: Muestra la información del último cliente en la base de datos. Utiliza el método last() para mover el cursor al último registro y luego muestra la información del cliente.

2.12. Paquete redes

Crea una aplicación que nos permita gestionar la base de datos redes.sql

Debe tener un menú desde el que se puedan gestionar (Create, Read, Update, Delete) usuarios, posts y comentarios.

En el apartado DAO de esta unidad ya tenemos implementado el DAO para usuarios y la clase Main.java.

3. Fuentes de información

- Wikipedia
- Programación (Grado Superior) - Juan Carlos Moreno Pérez (Ed. Ra-ma)
- Apuntes IES Henri Matisse (Javi García Jimenez?)
- Apuntes AulaCampus
- Apuntes José Luis Comesaña
- Apuntes IOC Programació bàsica (Joan Arnedo Moreno)
- Apuntes IOC Programació Orientada a Objectes (Joan Arnedo Moreno)
- https://arturoblasco.github.io/pr