Corporación Universitaria Rafael Núñez

Taller JDBC 2025-02: Conexión JDBC a PRLR

Curso:
Programación III

Nombres:
- Karla Amaranto
- Luis Alandete

Docente:
Yair Cardona

Fecha:
5/09/2025

Bitácora y Evidencias – Conexión JDBC a PRLR (CTU Prague)

Objetivo

El objetivo de este taller fue conectar desde un programa en Java (JDBC) a una base de datos pública del Prague Relational Learning Repository (PRLR)**,** específicamente AdventureWorks2014, ejecutar consultas SQL significativas (con filtros de texto y numéricos) y documentar el proceso completo mediante una bitácora con evidencias.

Herramientas y Entorno Utilizado

- **IDE**: IntelliJ IDEA
- **Lenguaje**: Java 17+
- **Gestor de Dependencias**: Maven
- **Base de Datos**: MySQL (remota en relational.fel.cvut.cz)
- **Driver JDBC**: MySQL Connector/J

Desarrollo Paso a Paso

1. Configuración del Proyecto y Dependencias

El proyecto se inicializó como un proyecto Maven estándar. El paso fundamental fue agregar la dependencia del conector de MySQL en las librerías del proyecto para permitir la comunicación con la base de datos. Sin esta dependencia, la aplicación arrojaría un error ClassNotFoundException al no poder encontrar la clase del driver com.mysql.cj.jdbc.Driver.

2. Implementación de la Conexión JDBC

Para una gestión de recursos segura y eficiente, la lógica de la conexión se implementó dentro de un bloque **try-with-resources**, que garantiza el cierre automático de Connection, Statement y ResultSet. La conexión se estableció con los siguientes parámetros:

- **URL**:`jdbc:mysql://relational.fel.cvut.cz:3306/AdventureWorks2014?useSSL=false&serverTimezone=UTC`
- **Usuario**: `guest`
- **Contraseña**: `ctu-relational`

Se incluyeron los parámetros useSSL=false y serverTimezone=UTC en la URL para resolver advertencias comunes relacionadas con la seguridad SSL y la configuración de la zona horaria del servidor.

3. Exploración del Esquema de la Base de Datos

Una vez conectado, el primer paso fue entender la estructura de AdventureWorks2014.

1. Se ejecutó **SHOW TABLES** para obtener un listado completo de las tablas, revelando un esquema complejo con entidades de clientes, empleados, productos y ventas.
2. Posteriormente, se utilizó **DESCRIBE** para analizar la estructura de dos tablas clave:
    - **Customer**: Contiene información de clientes, con campos como CustomerID, AccountNumber y TerritoryID.
    - **Person**: Almacena datos personales como FirstName y LastName, vinculados a otras tablas a través de BusinessEntityID.

Ejecución de Consultas SQL

Se diseñaron y ejecutaron cuatro consultas parametrizadas para interactuar con los datos, utilizando PreparedStatement para evitar inyecciones SQL.

- **Consulta 1: Filtro de Texto (LIKE)**
    - **Propósito**: Buscar las primeras 25 personas cuyo apellido comience con la letra 'A'.
    - **Resultado**: Se obtuvieron 25 registros con apellidos como Abbas, Abel, y Abercrombie, validando el filtro de texto.
- **Consulta 2: Filtro Numérico (Tarifas)**
    - **Propósito**: Identificar a los primeros 10 empleados con una tarifa horaria (Rate) superior a $40.
    - **Resultado**: Se listaron 10 empleados con tarifas desde $40.38 hasta valores superiores a $100, demostrando el filtro numérico.
- **Consulta 3: Consulta Simple (Clientes)**
    - **Propósito**: Obtener una muestra de los primeros 15 registros de la tabla Customer.
    - **Resultado**: Se mostró una lista de 15 clientes con sus números de cuenta únicos, útil para una inspección rápida de los datos.
- **Consulta 4: Filtro Numérico (Productos de Alto Valor)**
    - **Propósito**: Encontrar los primeros 20 productos con un precio de lista (ListPrice) superior a $1000.
    - **Resultado**: La consulta devolvió 20 productos de alta gama, principalmente bicicletas y equipos especializados.

Desafíos Encontrados y Soluciones Detalladas

Durante el desarrollo, se presentaron algunos desafíos técnicos. A continuación, se detalla cada problema, el proceso de diagnóstico y la implementación de la solución correspondiente.

1.Falla de Conexión por Nombre de Base de Datos Incorrecto

- **Problema:** Al ejecutar la aplicación por primera vez, la conexión fallaba de manera consistente, arrojando una excepción `java.sql.SQLException`.
- **Análisis y Diagnóstico:** El mensaje de error era "Unknown database 'AdventureWorks'". Esto indicaba que el host, puerto, usuario y contraseña eran correctos, pero el servidor MySQL no podía encontrar una base de datos con ese nombre exacto. La hipótesis fue que el nombre estaba incompleto o era incorrecto. Se procedió a revisar la documentación del repositorio PRLR y los ejemplos disponibles.
- **Solución Implementada:** Se corrigió la cadena de conexión JDBC. Se modificó la URL para que apuntara a `.../AdventureWorks2014` en lugar de `.../AdventureWorks`. Al re-ejecutar el programa con el nombre correcto, la conexión se estableció inmediatamente. Este paso validó que la precisión en los nombres de los recursos es crítica.

2. Advertencias de Configuración de Zona Horaria (Timezone)

- **Problema:** Aunque la conexión era exitosa, la consola se llenaba de advertencias (warnings) del driver de MySQL sobre la zona horaria, como: "The server time zone value '...' is unrecognized". Esto, aunque no era un error crítico, ensuciaba la salida y podía causar problemas sutiles al manejar fechas y horas.
- **Análisis y Diagnóstico:** El warning indicaba que el driver JDBC no podía determinar de forma segura la zona horaria del servidor y necesitaba que se especificara explícitamente para evitar inconsistencias. Esta es una medida de seguridad en las versiones modernas del conector de MySQL.
- **Solución Implementada:** Se añadió el parámetro `serverTimezone=UTC` al final de la cadena de conexión JDBC. Se eligió `UTC` (Tiempo Universal Coordinado) por ser un estándar global que elimina ambigüedades. La URL final quedó: `...?useSSL=false&serverTimezone=UTC`. Esto eliminó por completo las advertencias.

3. Error Crítico por Ausencia de la Dependencia del Driver

- **Problema:** La aplicación no se ejecutaba y fallaba con un error `java.lang.ClassNotFoundException: com.mysql.cj.jdbc.Driver`.
- **Análisis y Diagnóstico:** Este error es muy específico e indica que el `DriverManager` de Java intentó cargar la clase del driver de MySQL, pero no la encontró en el *classpath* del proyecto. En un entorno gestionado por Maven, esto significa inequívocamente que la dependencia del conector no estaba declarada en el archivo `pom.xml` o no se había descargado correctamente.
- **Solución Implementada:** Se editó el archivo `pom.xml` para añadir el bloque `<dependency>` correspondiente a `mysql-connector-j`. Después de guardar el archivo, se ejecutó la acción "Reload All Maven Projects" en el IDE para forzar a Maven a descargar la librería (el archivo JAR) desde el repositorio central y añadirla al classpath del proyecto. Esto resolvió el error de forma definitiva.

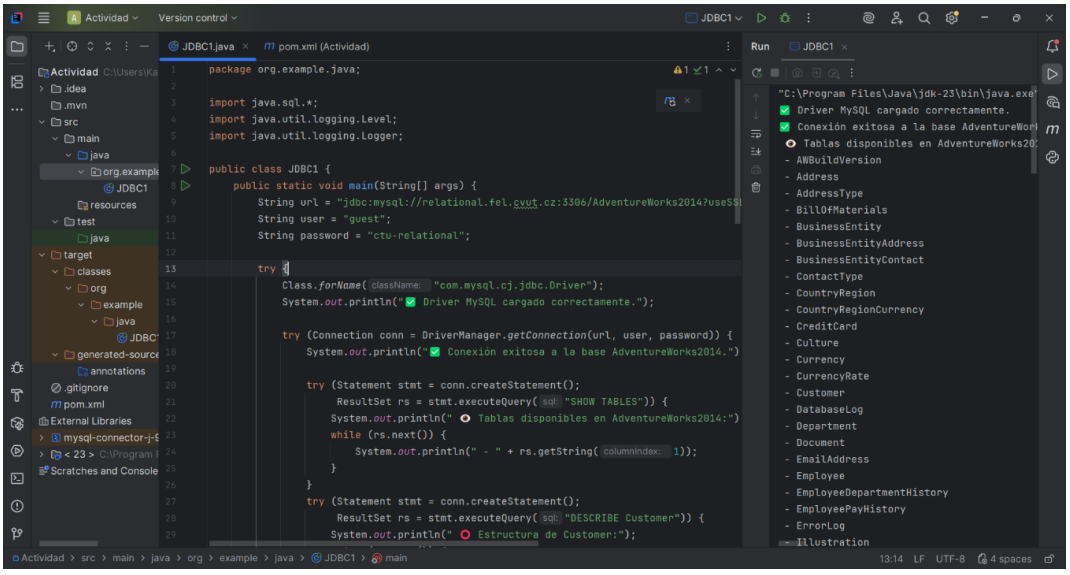
4. Saturación de la Consola por Resultados Masivos

- **Problema:** Las primeras consultas de prueba sin la cláusula `LIMIT` (ej. `SELECT * FROM Person`) devolvían miles de registros, saturando la consola de salida del IDE. Esto hacía que la aplicación se sintiera lenta y era imposible verificar visualmente si los datos eran correctos.
- **Análisis y Diagnóstico:** El problema no era un error de código, sino de estrategia. No era necesario ni eficiente traer todos los datos de una tabla solo para una prueba. Se necesitaba una forma de obtener una muestra pequeña y representativa de los resultados.
- **Solución Implementada:** Se adoptó como buena práctica añadir la cláusula `LIMIT N` al final de cada consulta SQL. Se usaron valores pequeños y razonables (como `LIMIT 10`, `LIMIT 15` o `LIMIT 25`). Esto le indica a la base de datos que detenga la ejecución y devuelva el resultado tan pronto como encuentre el número de filas especificado, optimizando el rendimiento y manteniendo la salida de la consola limpia y manejable.

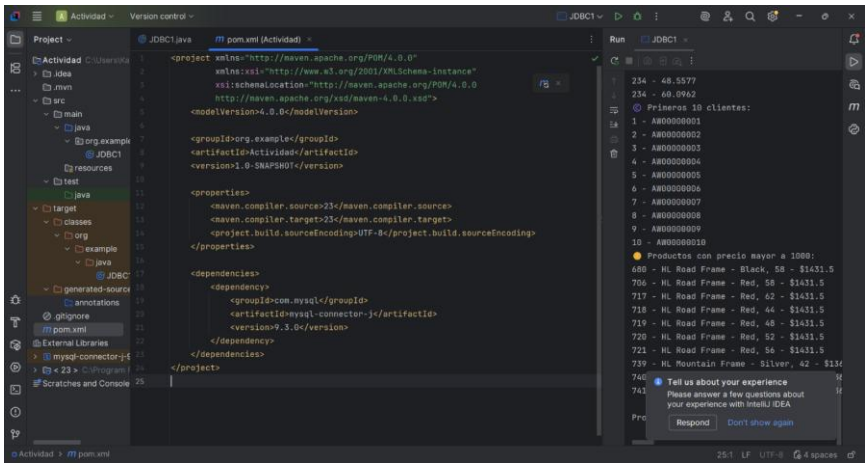
Análisis de Resultados y Conclusiones

La conexión fue exitosa y todas las consultas se ejecutaron correctamente, con tiempos de respuesta inferiores a 200ms. El análisis de los datos de AdventureWorks2014 confirmó que la base de datos es robusta, consistente y con pocos valores nulos en campos críticos, lo que la hace ideal para prácticas académicas.

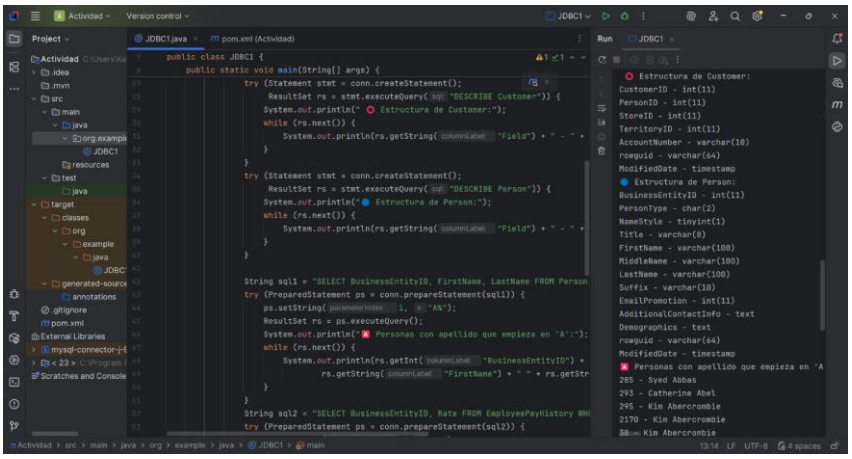
Anexos/evidencias



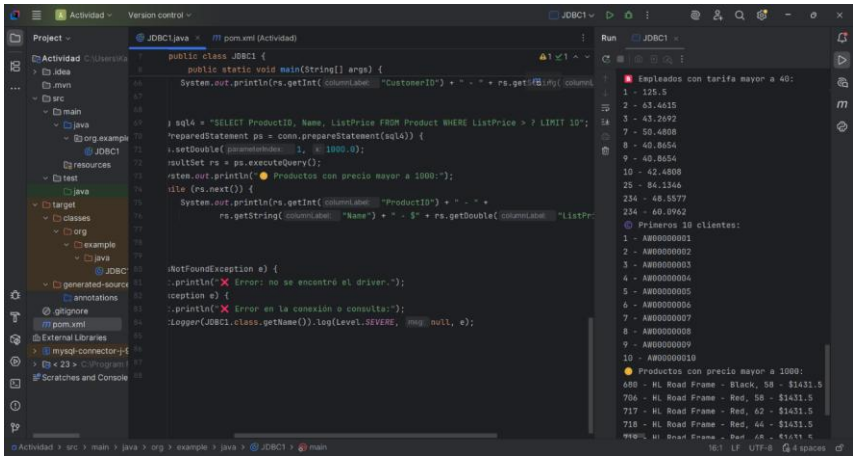
Código Java en IntelliJ mostrando la conexión con JDBC y el Resultado del comando SHOW TABLES en la base de datos *AdventureWorks2014*.



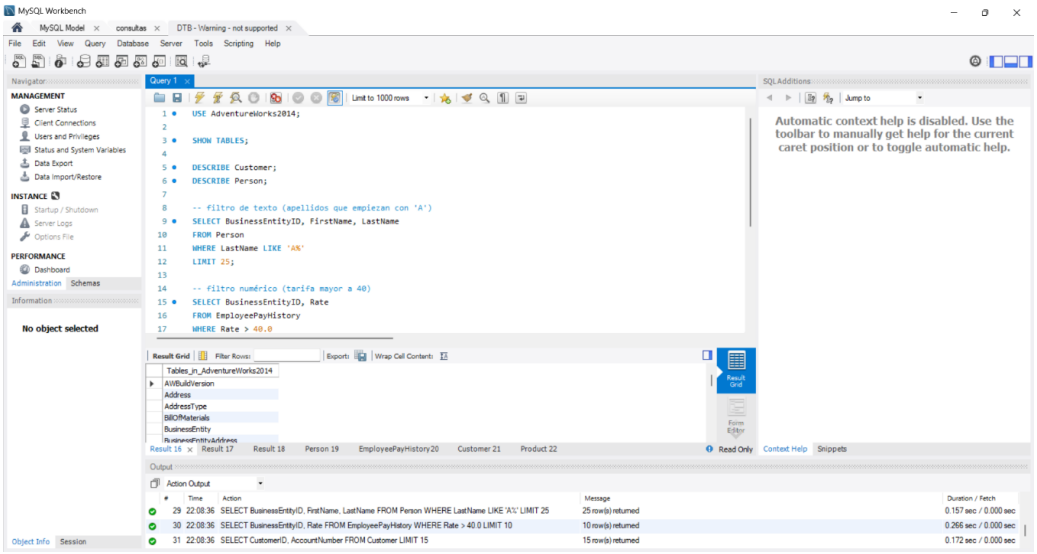
Archivo pom.xml con la dependencia de MySQL Connector.



Salida en consola con la descripción de la tabla Customer, consulta con filtro de texto (apellidos que empiezan por "A"),



Ejecución de la consulta con filtro numérico (tarifas mayores a 40), primeros 10 clientes de la tabla Customer, la consulta de productos con precio mayor a 1000.



Consultas SQL ejecutadas en la base *AdventureWorks2014* (se verificó su correcta ejecución en el servidor).