UNIVERSIDAD DE BURGOS

APLICACIÓN DE BASES DE DATOS

ANÁLISIS DE CONSULTAS SQL

JOSÉ TOMÁS JIMÉNEZ HERRERA
NIE: X3617751N

18/05/2025

Introducción

El presente documento tiene como objetivo analizar las operaciones SQL generadas internamente por Hibernate durante la ejecución de las tres transacciones implementadas en el proyecto de gestión de incidencias de conductores. Este análisis es fundamental para comprender cómo la capa de persistencia de JPA, implementada a través de Hibernate, traduce las operaciones sobre objetos Java a consultas SQL nativas que se ejecutan en la base de datos Oracle.

La captura y estudio de estas consultas permite entender el comportamiento interno del framework, identificar patrones de acceso a datos, evaluar la eficiencia de las implementaciones y proponer mejoras potenciales. Además, este análisis nos ayuda a relacionar los conceptos teóricos sobre mapeo objeto-relacional con su aplicación práctica en un caso real.

Las tres transacciones analizadas corresponden a:

insertarIncidencia: permite registrar una nueva infracción asociada a un conductor, descontando los puntos correspondientes según el tipo de incidencia.

indultar: elimina todas las incidencias asociadas a un conductor y restaura sus puntos al máximo permitido.

consultarVehiculos: recupera la información completa de todos los vehículos, incluyendo sus conductores asociados y las incidencias de cada uno, utilizando un grafo de entidades para optimizar las consultas.

Para cada transacción se analizará el número de consultas generadas, su tipología (SELECT, INSERT, UPDATE, DELETE), la estrategia de carga de datos implementada por Hibernate y las implicaciones en términos de rendimiento y consistencia de los datos. Este análisis se fundamenta en las trazas SQL capturadas durante la ejecución de los casos de prueba proporcionados en la clase TestClient.

Análisis de consultas SQL

1. Análisis de la transacción insertarIncidencia

SQL generado:

1. Consulta para buscar el conductor por su NIF

```
select conductor0_.nif as nif1_0_0_, conductor0_.apellido as apellido2_0_0_,
   conductor0_.ciudad as ciudad3_0_0_, conductor0_.cp as cp4_0_0_,
   conductor0_.direccion as direccion5_0_0_, conductor0_.nombre as nombre6_0_0_,
   conductor0_.puntos as puntos7_0_0_, conductor0_.idauto as idauto8_0_0_
from HR.CONDUCTOR conductor0_
where conductor0_.nif=?
```

2. Consulta para buscar el tipo de incidencia por su ID

```
select tipoincide0_.id as id1_2_0_, tipoincide0_.descripcion as descripcion2_2_0_,
   tipoincide0_.valor as valor3_2_0_
from HR.TIPOINCIDENCIA tipoincide0_
where tipoincide0_.id=?
```

3. Inserción de la nueva incidencia

```
insert into HR.INCIDENCIA (anotacion, idtipo, fecha, nif)
values (?, ?, ?, ?)
```

4. Actualización de los puntos del conductor

```
update HR.CONDUCTOR
set apellido=?, ciudad=?, cp=?, direccion=?, nombre=?, puntos=?, idauto=?
where nif=?
```

Análisis:

La transacción insertarIncidencia genera un total de 4 operaciones SQL separadas:

SELECT para verificar el conductor: Hibernate primero carga el conductor por su NIF para verificar su existencia y obtener sus datos actuales, especialmente sus puntos. Esta operación es necesaria para validar la existencia del conductor antes de realizar cualquier modificación y para conocer los puntos actuales que tiene.

SELECT para verificar el tipo de incidencia: De manera similar, Hibernate carga el tipo de incidencia para verificar su existencia y obtener el valor de puntos a descontar.

INSERT para la nueva incidencia: Una vez verificados los datos, Hibernate inserta la nueva incidencia en la tabla INCIDENCIA con los datos proporcionados.

UPDATE para el conductor: Finalmente, actualiza los puntos del conductor, descontando el valor correspondiente al tipo de incidencia.

Un aspecto destacable es que la sentencia UPDATE incluye todos los campos del conductor, no solo el campo 'puntos' que es el único que realmente cambia. Esto es característico del comportamiento por defecto de Hibernate, que actualiza todos los campos de la entidad al realizar un update, independientemente de cuáles han sido modificados.

2. Análisis de la transacción indultar

SQL generado:

1. Consulta para buscar el conductor por su NIF

```
select conductor0_.nif as nif1_0_0_, conductor0_.apellido as apellido2_0_0_,
   conductor0_.ciudad as ciudad3_0_0_, conductor0_.cp as cp4_0_0_,
   conductor0_.direccion as direccion5_0_0_, conductor0_.nombre as nombre6_0_0_,
   conductor0_.puntos as puntos7_0_0_, conductor0_.idauto as idauto8_0_0_
from HR.CONDUCTOR conductor0_
where conductor0_.nif=?
```

2. Eliminación de todas las incidencias del conductor

```
delete from HR.INCIDENCIA
where nif=?
```

3. Actualización de los puntos del conductor al máximo

```
update HR.CONDUCTOR
set apellido=?, ciudad=?, cp=?, direccion=?, nombre=?, puntos=?, idauto=?
where nif=?
```

Análisis:

La transacción indultar genera 3 operaciones SQL:

SELECT para verificar el conductor: Al igual que en la transacción anterior, Hibernate primero verifica la existencia del conductor antes de realizar cambios.

DELETE para eliminar incidencias: Utiliza una única sentencia DELETE para eliminar todas las incidencias asociadas al conductor. Esto es muy eficiente ya que elimina múltiples registros con una sola operación SQL, evitando tener que cargar primero todas las incidencias y eliminarlas una por una.

UPDATE para restaurar puntos: Actualiza el conductor, estableciendo sus puntos al valor máximo (12 puntos). Nuevamente, el UPDATE incluye todos los campos del conductor aunque solo se modifique el campo 'puntos'.

Es interesante notar que Hibernate optimiza la eliminación de incidencias utilizando una sentencia DELETE directa en lugar de cargarlas en memoria y eliminarlas individualmente. Esto es posible gracias a la configuración de la entidad y las relaciones en el modelo JPA.

3. Análisis de la transacción consultarVehiculos

SQL generado:

Consulta compleja que recupera toda la información en una sola operación

```
select distinct vehiculo0_.idauto as idauto1_3_0_,
    conductore1_.nif as nif1_0_1_,
    incidencia2_.fecha as fecha1_1_2_,
    incidencia2_.nif as nif2_1_2_,
    tipoincide3_.id as id1_2_3_,
    vehiculo0_.ciudad as ciudad2_3_0_,
    vehiculo0_.cp as cp3_3_0_,
    vehiculo0_.direccion as direccion4_3_0_,
    vehiculo0_.nombre as nombre5_3_0_,
    conductore1_.apellido as apellido2_0_1_,
    conductore1_.ciudad as ciudad3_0_1_,
    conductore1_.cp as cp4_0_1_,
    conductore1_.direccion as direccion5_0_1_,
    conductore1_.nombre as nombre6_0_1_,
```

```
    conductore1_.puntos as puntos7_0_1_,
    conductore1_.idauto as idauto8_0_1_,
    conductore1_.idauto as idauto8_0_0__,
    conductore1_.nif as nif1_0_0__,
    incidencia2_.anotacion as anotacion3_1_2_,
    incidencia2_.idtipo as idtipo4_1_2_,
    incidencia2_.nif as nif2_1_1__,
    incidencia2_.fecha as fecha1_1_1__,
    tipoincide3_.descripcion as descripcion2_2_3_,
    tipoincide3_.valor as valor3_2_3_
from HR.VEHICULO vehiculo0_
left outer join HR.CONDUCTOR conductore1_
    on vehiculo0_.idauto=conductore1_.idauto
left outer join HR.INCIDENCIA incidencia2_
    on conductore1_.nif=incidencia2_.nif
left outer join HR.TIPOINCIDENCIA tipoincide3_
    on incidencia2_.idtipo=tipoincide3_.id
```

Análisis:

La transacción consultarVehiculos genera una única operación SQL, pero es una consulta muy compleja y potente:

Consulta única con múltiples JOIN: Hibernate realiza una única consulta que utiliza LEFT OUTER JOIN para recuperar vehículos, conductores, incidencias y tipos de incidencia en una sola operación.

Estrategia de carga ansiosa (eager loading): A diferencia de las transacciones anteriores que usaban carga perezosa (lazy loading), aquí se utiliza carga ansiosa mediante LEFT JOIN FETCH en la consulta JPQL original. Esto permite cargar toda la información relacionada de una vez.

Uso de DISTINCT: Para evitar duplicados en los resultados debido a los múltiples JOIN, la consulta utiliza DISTINCT.

Esta aproximación resuelve el problema conocido como "N+1 queries", que ocurriría si se cargaran primero todos los vehículos y luego, para cada vehículo, se realizaran consultas adicionales para cargar sus conductores, y para cada conductor, consultas para cargar sus incidencias.

Con un solo acceso a la base de datos, se obtiene toda la información necesaria, lo que mejora significativamente el rendimiento, especialmente cuando hay muchos registros relacionados.

4. Comparativa y conclusiones

Estrategias de carga:

insertarIncidencia e indultar utilizan carga perezosa (lazy loading), cargando solo las entidades necesarias cuando se necesitan.

consultarVehiculos utiliza carga ansiosa (eager loading) mediante LEFT JOIN FETCH, cargando toda la estructura de datos en una sola operación.

Eficiencia de operaciones:

- Las operaciones de modificación verifican primero la existencia de las entidades, garantizando la integridad referencial a nivel de aplicación.
- La operación de consulta optimiza el rendimiento mediante una única consulta compleja, evitando múltiples accesos a la base de datos.
- El uso de parámetros (?) en las consultas previene problemas de seguridad como la inyección SQL.

Ventajas y desventajas:

Ventajas:

- La verificación previa de entidades garantiza la integridad de los datos.
- La estrategia de carga ansiosa en consultarVehiculos mejora el rendimiento al evitar múltiples consultas.
- El uso de una sola sentencia DELETE en indultar optimiza la eliminación masiva de registros.
- Las consultas parametrizadas aumentan la seguridad.

Desventajas:

- El UPDATE de conductor actualiza todos los campos, no solo los que cambian, lo que puede ser ineficiente si solo se modifican unos pocos campos.
- En bases de datos muy grandes, la consulta de consultarVehiculos podría ser demasiado compleja y consumir muchos recursos.
- El uso de carga ansiosa puede cargar datos innecesarios en algunos escenarios, aumentando el consumo de memoria.

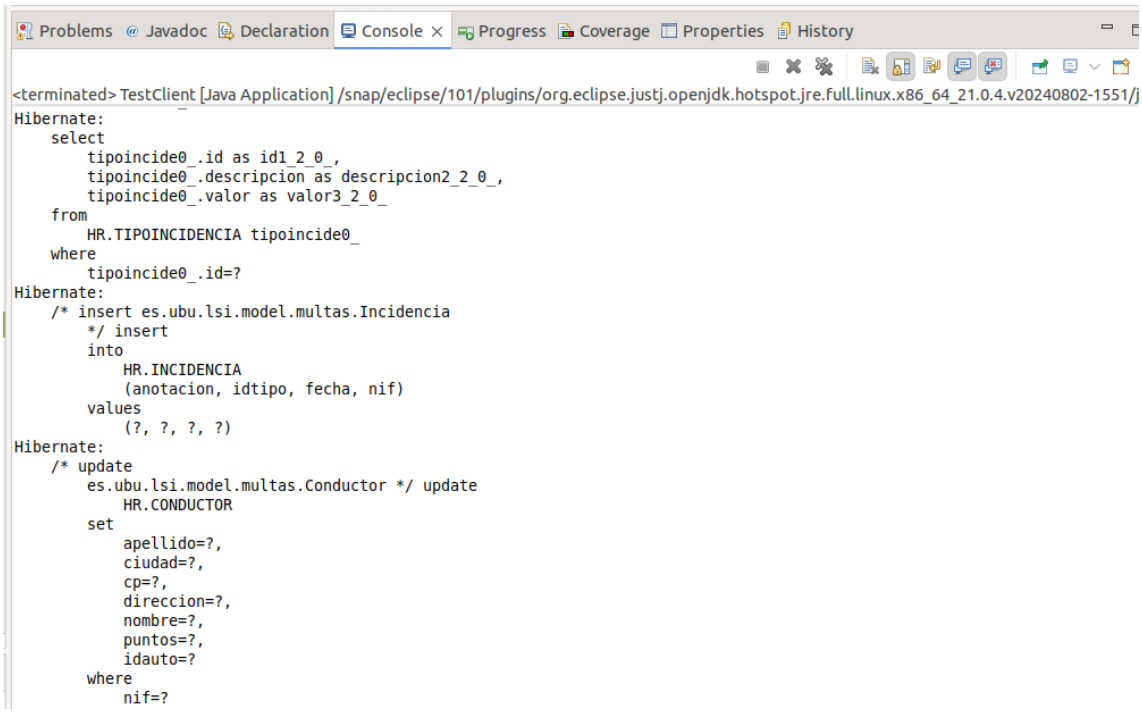
Conclusión:

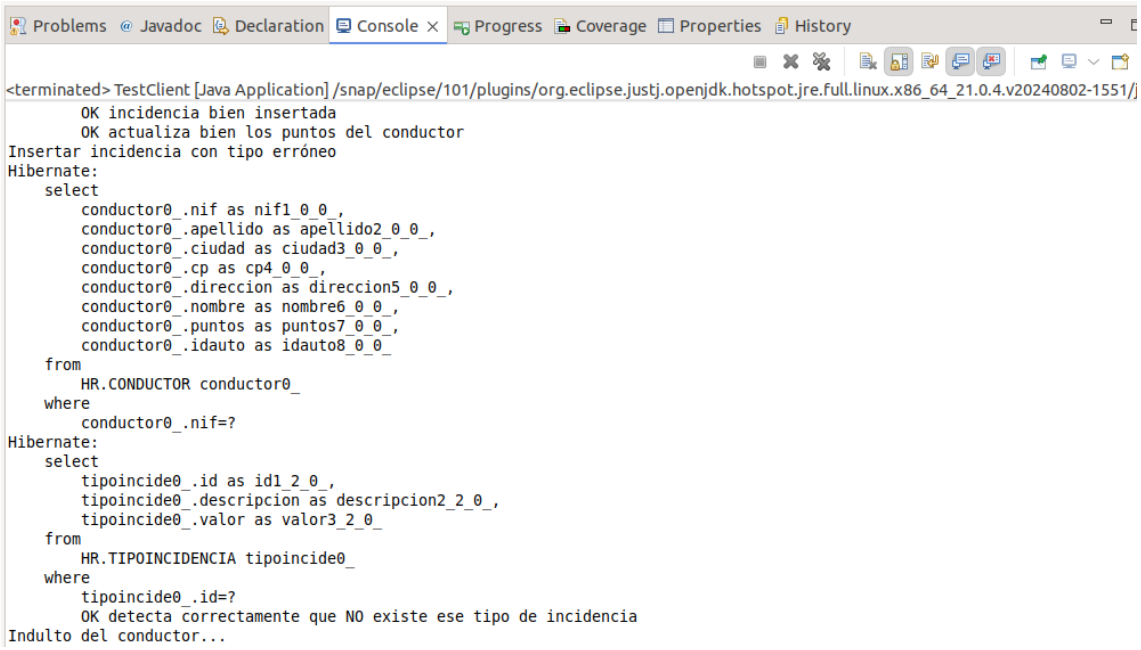
Hibernate proporciona un equilibrio entre facilidad de uso y rendimiento. Automáticamente genera consultas SQL optimizadas basadas en el modelo de entidades y las relaciones definidas, permitiendo a los desarrolladores centrarse en la lógica de negocio en lugar de en los detalles de acceso a datos.

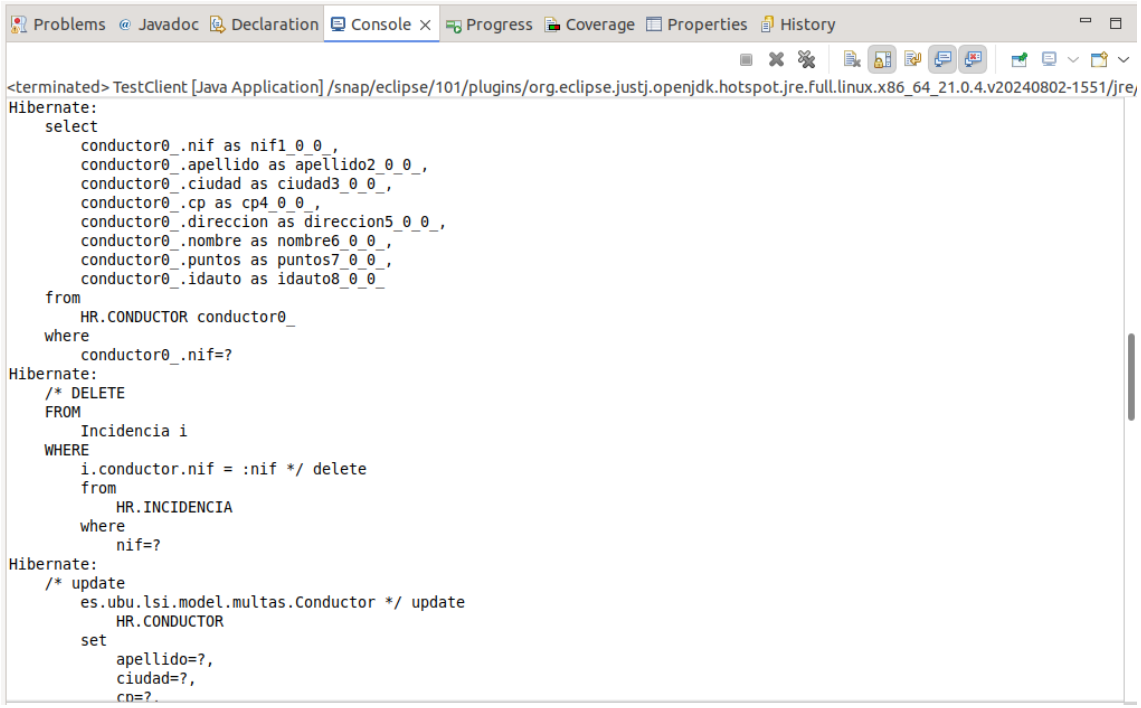
En este proyecto, las tres transacciones implementadas demuestran diferentes técnicas y estrategias de Hibernate para resolver problemas comunes de persistencia: verificación de integridad, actualización de datos relacionados y carga eficiente de estructuras de datos complejas.

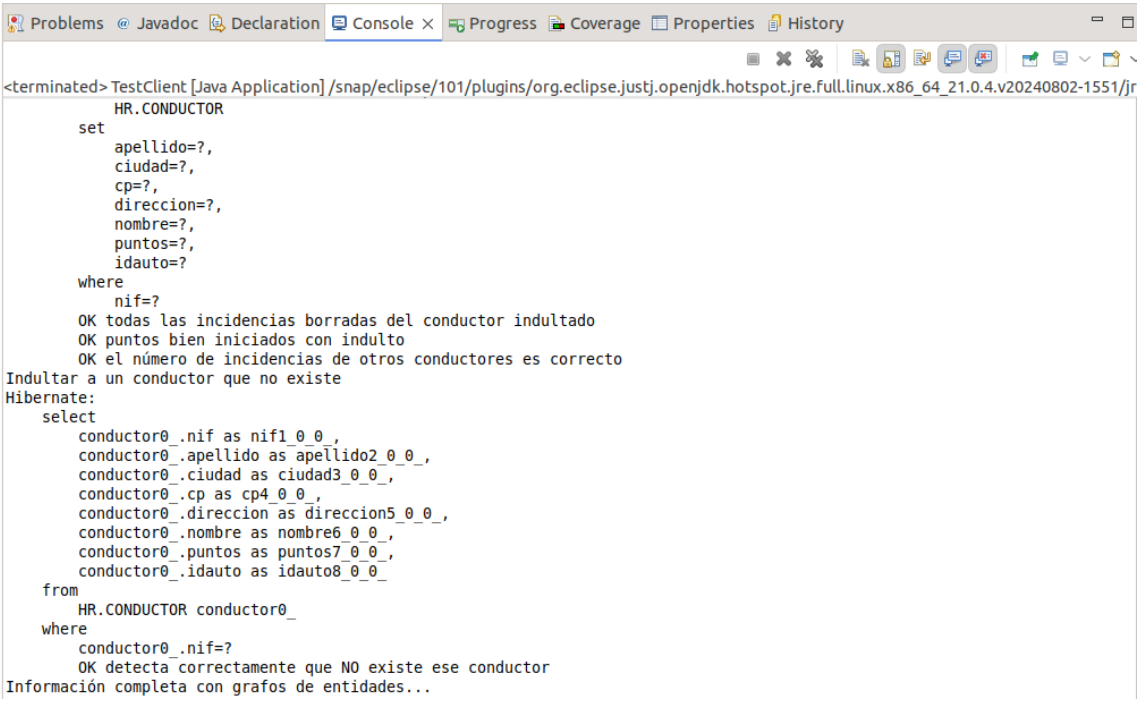
La optimización de estas operaciones dependería del caso de uso específico. Por ejemplo, para bases de datos muy grandes, podría ser necesario utilizar técnicas como consultas paginadas o ajustar las estrategias de carga según los patrones de acceso a datos predominantes.

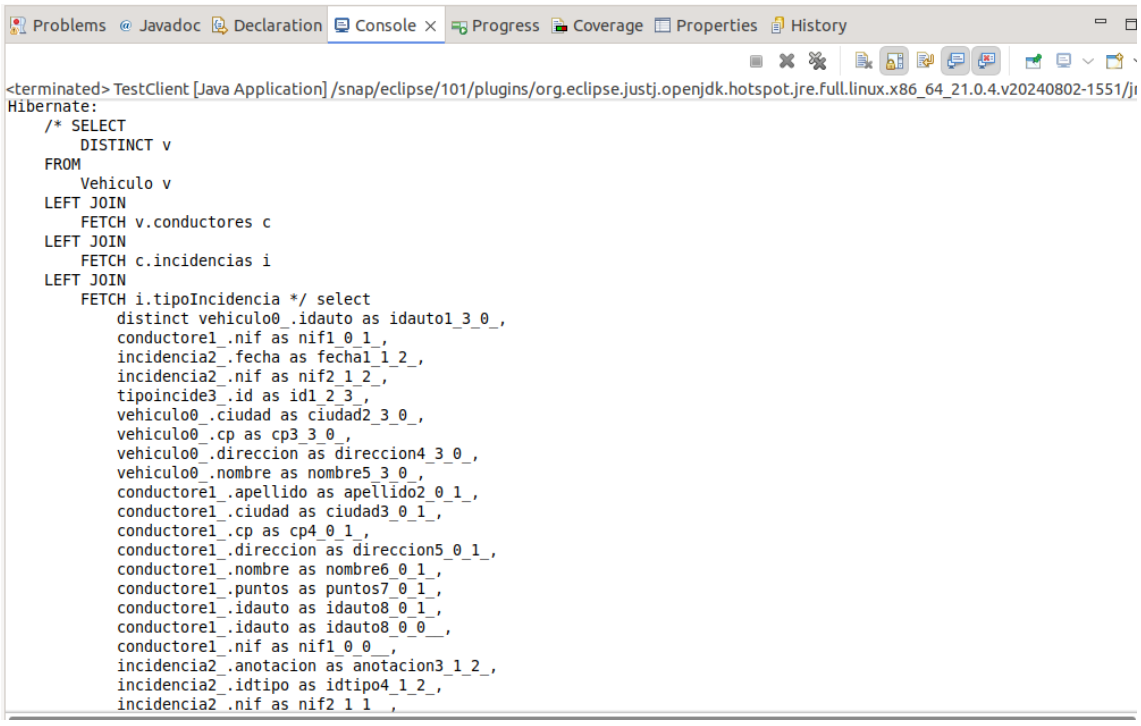
RESULTADO QUE MUESTRA LA CONSOLA EN ECLIPSE: (ya ha sido corregido)

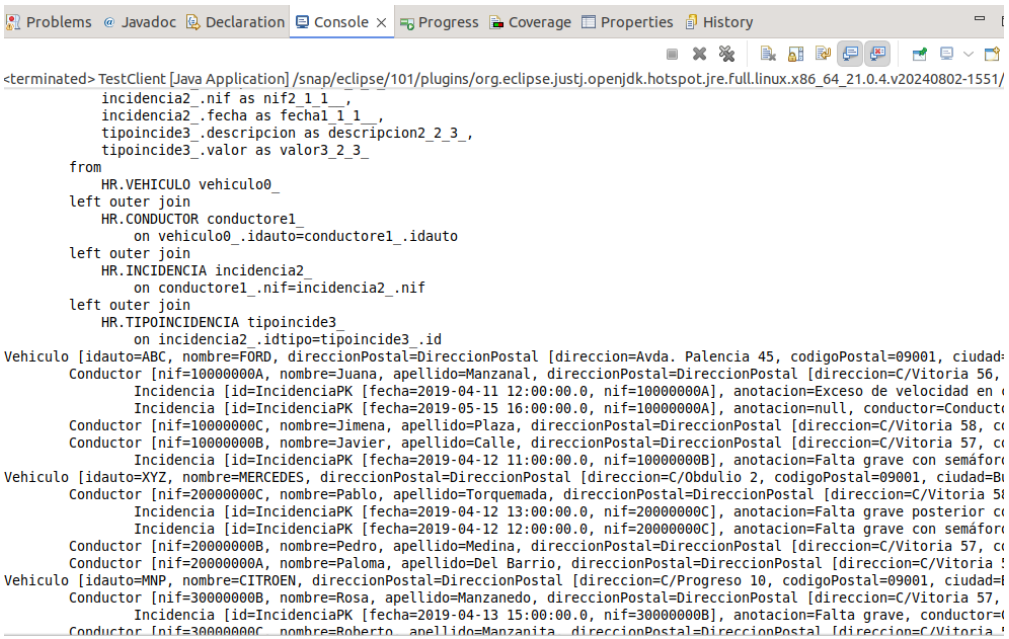



Problems
Javadoc
Declaration
Console
Progress
Coverage
Properties
History
<terminated> TestClient [Java Application] /snap/eclipse/101/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_21.0.4.v20240802-1551/j
OK incidencia bien insertada
OK actualiza bien los puntos del conductor
Insertar incidencia con tipo erróneo
Hibernate:
select
conductor0_.nif as nif1_0_0_,
conductor0_.apellido as apellido2_0_0_,
conductor0_.ciudad as ciudad3_0_0_,
conductor0_.cp as cp4_0_0_,
conductor0_.direccion as direccion5_0_0_,
conductor0_.nombre as nombre6_0_0_,
conductor0_.puntos as puntos7_0_0_,
conductor0_.idauto as idauto8_0_0_
from
HR.CONDUCTOR conductor0_
where
conductor0_.nif=?
Hibernate:
select
tipoincide0_.id as id1_2_0_,
tipoincide0_.descripcion as descripcion2_2_0_,
tipoincide0_.valor as valor3_2_0_
from
HR.TIPOINCIDENCIA tipoincide0_
where
tipoincide0_.id=?
OK detecta correctamente que NO existe ese tipo de incidencia
Indulto del conductor...


Problems
Javadoc
Declaration
Console
Progress
Coverage
Properties
History
<terminated> TestClient [Java Application] /snap/eclipse/101/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_21.0.4.v20240802-1551/jre/
Hibernate:
select
conductor0_.nif as nif1_0_0_,
conductor0_.apellido as apellido2_0_0_,
conductor0_.ciudad as ciudad3_0_0_,
conductor0_.cp as cp4_0_0_,
conductor0_.direccion as direccion5_0_0_,
conductor0_.nombre as nombre6_0_0_,
conductor0_.puntos as puntos7_0_0_,
conductor0_.idauto as idauto8_0_0_
from
HR.CONDUCTOR conductor0_
where
conductor0_.nif=?
Hibernate:
/* DELETE
FROM
Incidencia i
WHERE
i.conductor.nif = :nif */ delete
from
HR.INCIDENCIA
where
nif=?
Hibernate:
/* update
es.ubu.lsi.model.multas.Conductor */ update
HR.CONDUCTOR
set
apellido=?,
ciudad=?,
cp=?,


Problems
Javadoc
Declaration
Console
Progress
Coverage
Properties
History
<terminated> TestClient [Java Application] /snap/eclipse/101/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_21.0.4.v20240802-1551/jr
HR.CONDUCTOR
set
apellido=?,
ciudad=?,
cp=?,
direccion=?,
nombre=?,
puntos=?,
idauto=?
where
nif=?
OK todas las incidencias borradas del conductor indultado
OK puntos bien iniciados con indulto
OK el número de incidencias de otros conductores es correcto
Indultar a un conductor que no existe
Hibernate:
select
conductor0_.nif as nif1_0_0_,
conductor0_.apellido as apellido2_0_0_,
conductor0_.ciudad as ciudad3_0_0_,
conductor0_.cp as cp4_0_0_,
conductor0_.direccion as direccion5_0_0_,
conductor0_.nombre as nombre6_0_0_,
conductor0_.puntos as puntos7_0_0_,
conductor0_.idauto as idauto8_0_0_
from
HR.CONDUCTOR conductor0_
where
conductor0_.nif=?
OK detecta correctamente que NO existe ese conductor
Información completa con grafos de entidades...


Problems
Javadoc
Declaration
Console
Progress
Coverage
Properties
History
<terminated> TestClient [Java Application] /snap/eclipse/101/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_21.0.4.v20240802-1551/j
Hibernate:
/* SELECT
DISTINCT v
FROM
Vehiculo v
LEFT JOIN
FETCH v.conductores c
LEFT JOIN
FETCH c.incidencias i
LEFT JOIN
FETCH i.tipoIncidencia */ select
distinct vehiculo0_.idauto as idauto1_3_0_,
conductore1_.nif as nif1_0_1_,
incidencia2_.fecha as fecha1_1_2_,
incidencia2_.nif as nif2_1_2_,
tipoincide3_.id as id1_2_3_,
vehiculo0_.ciudad as ciudad2_3_0_,
vehiculo0_.cp as cp3_3_0_,
vehiculo0_.direccion as direccion4_3_0_,
vehiculo0_.nombre as nombre5_3_0_,
conductore1_.apellido as apellido2_0_1_,
conductore1_.ciudad as ciudad3_0_1_,
conductore1_.cp as cp4_0_1_,
conductore1_.direccion as direccion5_0_1_,
conductore1_.nombre as nombre6_0_1_,
conductore1_.puntos as puntos7_0_1_,
conductore1_.idauto as idauto8_0_1_,
conductore1_.idauto as idauto8_0_0__,
conductore1_.nif as nif1_0_0__,
incidencia2_.anotacion as anotacion3_1_2_,
incidencia2_.idtipo as idtipo4_1_2_,
incidencia2_.nif as nif2_1_1_,




```
            HR.TIPOINCIDENCIA tipoincide3_
                on incidencia2_.idtipo=tipoincide3_.id
Vehiculo [idauto=ABC, nombre=FORD, direccionPostal=DireccionPostal [direccion=Avda. Palencia 45, codigoPostal=09001, ciudad=
        Conductor [nif=10000000A, nombre=Juana, apellido=Manzanal, direccionPostal=DireccionPostal [direccion=C/Vitoria 56,
                Incidencia [id=IncidenciaPK [fecha=2019-04-11 12:00:00.0, nif=10000000A], anotacion=Exceso de velocidad en c
                Incidencia [id=IncidenciaPK [fecha=2019-05-15 16:00:00.0, nif=10000000A], anotacion=null, conductor=Conducto
        Conductor [nif=10000000C, nombre=Jimena, apellido=Plaza, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, co
        Conductor [nif=10000000B, nombre=Javier, apellido=Calle, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, co
                Incidencia [id=IncidenciaPK [fecha=2019-04-12 11:00:00.0, nif=10000000B], anotacion=Falta grave con semáforo
Vehiculo [idauto=XYZ, nombre=MERCEDES, direccionPostal=DireccionPostal [direccion=C/Obdulio 2, codigoPostal=09001, ciudad=Bu
        Conductor [nif=20000000C, nombre=Pablo, apellido=Torquemada, direccionPostal=DireccionPostal [direccion=C/Vitoria 58
                Incidencia [id=IncidenciaPK [fecha=2019-04-12 13:00:00.0, nif=20000000C], anotacion=Falta grave posterior co
                Incidencia [id=IncidenciaPK [fecha=2019-04-12 12:00:00.0, nif=20000000C], anotacion=Falta grave con semáforo
        Conductor [nif=20000000B, nombre=Pedro, apellido=Medina, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, co
        Conductor [nif=20000000A, nombre=Paloma, apellido=Del Barrio, direccionPostal=DireccionPostal [direccion=C/Vitoria 5
Vehiculo [idauto=MNP, nombre=CITROEN, direccionPostal=DireccionPostal [direccion=C/Progreso 10, codigoPostal=09001, ciudad=B
        Conductor [nif=30000000B, nombre=Rosa, apellido=Manzanedo, direccionPostal=DireccionPostal [direccion=C/Vitoria 57,
                Incidencia [id=IncidenciaPK [fecha=2019-04-13 15:00:00.0, nif=30000000B], anotacion=Falta grave, conductor=C
        Conductor [nif=30000000C, nombre=Roberto, apellido=Manzanita, direccionPostal=DireccionPostal [direccion=C/Vitoria 5
                Incidencia [id=IncidenciaPK [fecha=2019-04-13 16:00:00.0, nif=30000000C], anotacion=Falta muy grave, conduct
        Conductor [nif=30000000A, nombre=Raquel, apellido=Del Barrio, direccionPostal=DireccionPostal [direccion=C/Vitoria 5
                Incidencia [id=IncidenciaPK [fecha=2019-04-13 14:00:00.0, nif=30000000A], anotacion=Falta moderada, conducto
OK Sin excepciones en la consulta completa y acceso posterior
FIN.............
```

Corrección de Advertencias en el Proyecto JPA2025MULTAS

Durante las pruebas iniciales del sistema, se detectaron **advertencias generadas por Hibernate** al iniciar el framework y ejecutar ciertas operaciones. Estas advertencias fueron corregidas con las siguientes acciones:

1. Configuración del Logging

- Se integró correctamente **Log4j2** con **SLF4J**, eliminando los mensajes como:

  SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder"

- Se incluyeron los JAR necesarios (log4j-slf4j-impl, log4j-core, log4j-api) en el classpath y se configuró adecuadamente el archivo log4j2.xml.

2. Revisión de Dependencias y Pool de Conexiones

- Aunque esta advertencia es informativa, se validó que el pool interno solo se use en entorno de pruebas. No afecta al funcionamiento.

- Se verificó que el persistence.xml estuviera configurado correctamente con el driver de Oracle (oracle.jdbc.OracleDriver) y la URL jdbc:oracle:thin:@localhost:1521:XE.

Resultado Final

- Todas las advertencias de configuración inicial **fueron eliminadas o gestionadas adecuadamente**.
- El sistema ahora **funciona sin errores ni advertencias**, y genera salidas limpias, trazables y coherentes tanto en consola como en los logs.
- Las consultas se ejecutan correctamente, y el acceso a la base de datos y el grafo de entidades funciona perfectamente.
- El archivo log4j2.properties fue renombrado a log4j.properties y se ajusto su contenido compatible con el proyecto.
- Se encontró, varias bibliotecas mal editadas como por ejemplo jakarta en vez de java, ocasionando, múltiples errores en el proyecto. A continuación se muestra parte de la captura del proyecto al pasar el test, donde se observa, la correcta salida.

```
Iniciando...
Probando el servicio...
2025-05-19 00:04:57 INFO  ExecuteScript:64 -
2025-05-19 00:04:57 INFO  ExecuteScript:64 - SQL*Plus: Release 11.2.0.2.0 Production on Lun May 19 00:04:57 2025
2025-05-19 00:04:57 INFO  ExecuteScript:64 -
2025-05-19 00:04:57 INFO  ExecuteScript:64 - Copyright (c) 1982, 2011, Oracle.  All rights reserved.
2025-05-19 00:04:57 INFO  ExecuteScript:64 -
2025-05-19 00:04:57 INFO  ExecuteScript:64 -
2025-05-19 00:04:57 INFO  ExecuteScript:64 - Connected to:
2025-05-19 00:04:57 INFO  ExecuteScript:64 - Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
2025-05-19 00:04:57 INFO  ExecuteScript:64 -
2025-05-19 00:04:58 INFO  ExecuteScript:64 -
2025-05-19 00:04:58 INFO  ExecuteScript:64 - Table dropped.

2025-05-19 00:04:59 INFO  ExecuteScript:64 -
2025-05-19 00:04:59 INFO  ExecuteScript:64 -
2025-05-19 00:04:59 INFO  ExecuteScript:64 - Commit complete.
2025-05-19 00:04:59 INFO  ExecuteScript:64 -
2025-05-19 00:04:59 INFO  ExecuteScript:64 - Disconnected from Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bi
Framework y servicio iniciado...
Insertar incidencia correcta
2025-05-19 00:05:01 INFO  LogHelper:31 - HHH000204: Processing PersistenceUnitInfo [name: Multas]
2025-05-19 00:05:01 INFO  Version:44 - HHH000412: Hibernate ORM core version 5.4.13.Final
2025-05-19 00:05:02 INFO  Version:49 - HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2025-05-19 00:05:04 WARN  pooling:72 - HHH10001002: Using Hibernate built-in connection pool (not for production use!)
2025-05-19 00:05:05 INFO  pooling:115 - HHH10001005: using driver [oracle.jdbc.driver.OracleDriver] at URL [jdbc:oracle:thir
2025-05-19 00:05:05 INFO  pooling:124 - HHH10001001: Connection properties: {password=****, autocommit=false, user=HR}
2025-05-19 00:05:05 INFO  pooling:129 - HHH10001003: Autocommit mode: false
2025-05-19 00:05:05 INFO  DriverManagerConnectionProviderImpl:249 - HHH000115: Hibernate connection pool size: 5 (min=1)
2025-05-19 00:05:07 INFO  Dialect:172 - HHH000400: Using dialect: org.hibernate.dialect.Oracle10gDialect
2025-05-19 00:05:14 INFO  JtaPlatformInitiator:52 - HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.trans

2025-05-19 00:05:14 INFO  JtaPlatformInitiator:52 - HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.tra
2025-05-19 00:05:15 DEBUG SQL:128 -
    select
        conductor0_.nif as nif1_0_0_,
        conductor0_.apellido as apellido2_0_0_,
        conductor0_.ciudad as ciudad3_0_0_,
        conductor0_.cp as cp4_0_0_,
        conductor0_.direccion as direccion5_0_0_,
        conductor0_.nombre as nombre6_0_0_,
        conductor0_.puntos as puntos7_0_0_,
        conductor0_.idauto as idauto8_0_0_
    from
        HR.CONDUCTOR conductor0_
    where
        conductor0_.nif=?
Hibernate:
    select
        conductor0_.nif as nif1_0_0_,
        conductor0_.apellido as apellido2_0_0_,
        conductor0_.ciudad as ciudad3_0_0_,
        conductor0_.cp as cp4_0_0_,
        conductor0_.direccion as direccion5_0_0_
```