Preguntas de Respositorios de Información

> Buenas me llamo ***, y como tú, también fui un estudiante de RI en Uniovi (curso 2017-2018).
>
> Si te ha llegado este pdf es que mi objetivo inicial sigue vivo: busco fomentar los buenos apuntes y que se compartan entre estudiantes, por lo que he hecho este pdf con todas las preguntas que he encontrado de esta asignatura. Si encuentras nuevas que no aparezcan, siéntete libre de crear un nuevo pdf y unirlo al final de este para así ayudar a los estudiantes de los futuros cursos (pagina para unir pdfs: https://www.ilovepdf.com/es/unir_pdf).
>
> Las preguntas no siguen un orden de temas por lo que encontraras cuestiones de los 5 bloques que yo tuve que cursar: NoSQL con Neo4J y Cypher, Elastic Search, XQuery, JDBC y JPA.
>
> Este pdf fue creado con el lenguaje Markdown y luego pasado a pdf gracias a esta página web: http://www.mdtr2pdf.com/index_en.html

1. Describe los motivos por los que se utilizan pool de conexiones.
    - Aumenta la eficiencia si se abren y cierran muchas aplicaciones.
    - Puede ser genérico siendo válido con diferentes drivers.
2. Enumera los distintos tipos de rowset.
    - JdbcRowSet
    - CachedRowSet
    - JoinRowSet
    - FilteredRowSet
    - WebRowSet
3. Según la arquitectura vista en clase SL_TS_TDG enumera las distintas capas que la componen y explica la finalidad de cada una.
    - **Service Layer:** Define los límites de la aplicación con una capa de servicios que establece el conjunto de operaciones disponible.

Desacopla presentación de negocio y además ayuda en el control de transacciones y en la seguridad.

- **Transaction Script:** Es un patrón para organizar la lógica de negocio o dominio. Utiliza procedimientos de forma que cada uno se encarga de procesar una petición desde presentación. Están separados de presentación y persistencia, por eso están en la capa de negocio. Una clase por uso.
- **Table Data GateWay:** Gestiona por completo el acceso a los datos(persistencia) de una entidad. Hace de pasarela. Puede ser Row o Table. Hay una clase por cada entidad o tabla. “AveriasGatewayImpl”. En las clases de negocio (TS) “AddMechanic” etc. se invoca a las clases de persistencia mediante la factoría “PersistenceFactory” que nos dará la Gateway que necesitamos

4. Enumera y explica los distintos niveles de aislamiento.

    - **Read uncommited:** Permite los tres tipos de violaciones de aislamiento. MySQL
    - **Read Committed:** evita lectura sucia. Lo tiene Oracle y PostgreSQL y MySQL
    - **Repeatable read:** Evita lectura sucia y no repetible . Lo tiene MySQL
    - **Serializable:** Evita Todas las violaciones de aislamiento. Esta en Oracle y PostgreSQL y MySQL

5. ¿En qué estado están los objetos devueltos por una consulta JQL? ¿y el que devuelve el método find?

    - En estado manager.
    - Tambien en estado manager.

6. ¿Cuales son los diferentes estados de una consulta JQL?

    - **New:** el objeto está en memoria pues acaba de ser creado o eliminado de la base de datos.
    - **Manager:** el objeto está presente tanto en memoria como en la base de datos. Las modificaciones que reciba el objeto en memoria se verán reflejadas también en la base ya que están asociados.

- **Detached:** el objeto existe en memoria y en la base de datos, pero estos no están asociados por lo que no podemos acceder a otras clases que estén interactuando con él pues necesitaríamos cargarlas desde la base, pero no tenemos acceso a ella.

7. Explica brevemente el teorema CAP.

    - El teorema CAP o teorema Brewer en NoSql, dice que en sistemas distribuidos es imposible garantizar a la vez: consistencia, disponibilidad y tolerancia a particiones.
    - AP: garantizan disponibilidad y tolerancia a particiones, pero no la consistencia, al menos de forma total.
    - CP: garantizan consistencia y tolerancia a particiones. Para lograr la consistencia y replicar los datos a través de los nodos, sacrifican la disponibilidad.
    - CA: garantizan consistencia y disponibilidad, pero tienen problemas con la tolerancia a particiones. Este problema lo suelen gestionar replicando los datos.

8. Define Entidad y Value Type. ¿Qué es mutable e inmutable en cada uno de ellos? ¿Sobre qué atributos se debe deben definir los métodos hashCode() y equals() en cada uno de ellos?

    - Entidades (Entities): Una entidad representa un concepto del dominio. Se puede asociar a otras entidades. Tiene un ciclo de vida independiente
    - Tipos valor (value type): Value Types representas información adicional, no conceptos principales. Atributos de una entidad. Su ciclo de vida depende de la entidad. No puede tener referencias entrantes
    - Los métodos se deben definir sobre la clave natural de las Entidades y sobre todos los atributos en los valueType.

9. ¿Por qué escribimos métodos para el mantenimiento de las asociaciones addXxxx(…) y removeXxxx(…) en vez de hacer las asignaciones directamente?

    - Porque nos permiten mantener el encapsulamiento y la integridad de las referencias cruzadas.

10. ¿Cómo se implementa en Java una clase asociativa UML?
    - Se implementan con una clase en la que haya un atributo y un gettercorrespondientes a cada una de las clases que relaciona. El constructor, que tendrá como parámetros objetos de las entidades, y el método unlink, sin parámetros y void, se encargarán de mantener la asociación. Finalmente en las clases correspondientes a cada identidad se crean dos getters:
11. ¿Qué parámetros mínimos debe recibir el constructor de una Entidad?
    - Todos aquellos que formen la entidad, es decir lo que en BBDD llamaríamos clave primaria.
12. ¿Qué parámetros debe recibir el constructor de un ValueType?
    - Todos ya que, al ser inmutables y no tener setters, no podrán modificar ningun atributo fuera de su constructor.
13. ¿Qué métodos no debe tener la implementación de un ValueType?
    - Los setters ya que son inmutables y no se rompe la encapsulación al devolverlos en los getters.
14. ¿Por qué los ValueTypes deben ser inmutables?
    - Porque representan un valor. Por ejemplo, una moneda de 2€ no importa si es ésta o aquella, sólo importa que es de 2€.
15. ¿Qué atributos no deben cambiar nunca en una Entidad?
    - El atributo (o combinación) que permite distinguir a esa entidad de las demás, es decir, los que representarian a la clave primaria en una tabla.
16. En la implementación de una entidad ¿A qué llamabamos atributos naturales?
    - A los que tienen significado en el contexto de uso, es decir, las que el usuario entiende y maneja en el mundo real. Ejemplos: DNI, Nº de la SS, etc.

17. ¿Para qué es útil redefinir el método toString() de las clases del modelo?
    - Es útil para depuración. Es útil ya que al redefinir este método se visualiza de una manera más clara el estado en que está el objeto en un momento dado
18. Describe las diferencias entre los estados “Transient”, “Persistent” y “Detached”.
    - **Transient:** Un objeto recién creado que no ha sido enlazado con el gestor de persistencia (sólo existe en memoria de la JVM)
    - **Persistent:** Un objeto enlazado con la sesión. Todos los cambios que se le hagan serán persistentes
    - **Detached:** Un objeto persistente que sigue en memoria después de que termina la sesión: existe en java y en la BDD.
19. ¿Cómo resuelve un mapeador JPA la herencia?
    - *Hay 3 estrategias diferentes:*
    - Tabla única en la que persisten todas las clases involucradas en la herencia (SINGLE_TABLE).
    - Tabla por cada clase no abstracta (TABLE_PER_CLASS).
    - Tabla por cada clase involucrada en la herencia, independientemente de sus características (JOINED).
20. ¿Cómo resuelve la diferencia de granularidad?
    - Implementando como una clase un conjunto de atributos. Y dejando como atributo ina instancia de dicha clase que los engloba.
21. ¿Cómo resuelve la detección de la identidad?
    - Se mapean con la etiqueta @Id la clase identidad
22. ¿Cómo resuelve la cardinalidad MANY to MANY de las asociaciones?
    - Se crea una clase intermedia que contendrá colecciones de cada una de las entidades relacionadas, y se utiliza la anotación @ManyToMany.

23. ¿Cómo resuelve la navegación?
    - **Unidireccional:**
    - Supongamos que la flecha de la relación va así -->. En la entidad de el extremo sin flecha instanciaremos un atributo de la otra entidad (Si es de muchos pondriamos un set). Y en el extremo con la flecha no haría falta poner nada.
    - **Bidirecional:**
    - Sería igual que la unidirecional, con la diferencia de que en ambos casos si deberíamos instanciar un atributo haciendo referencia a la otra entidad.
24. ¿Qué estrategias usan los mapeadores O/R para recrear en memoria una zona del grafo?
    - **Eager loading** (precarga): Se carga un objeto y sus asociados. Puede cargar más objetos de la cuenta y llenar la memoria. Riesgo de producto cartesiano.
    - **Lazy loading** (bajo demanda): Se carga al necesitarlo. Puede genera demasiadas SELECT * FROM. Tiene el problema de las n +1 consultas.
25. Para acelerar el acceso al grafo los mapeadores implementan una caché. ¿Qué características tiene esa caché?
    - Optimiza el rendimiento al reducir el trasiego con la BBDD
    - Permite hacer optimizaciones: Write-behind delayed, Batch load/ update.
26. En este orden: se crea una instancia del mapeador, se abre transacción, se recupera un objeto con el método find() y se invoca a unos cuántos métodos del objeto recuperado que le producen modificaciones. ¿Qué hay que hacer después para actualizar la base de datos con los cambios?
    - NADA. Ya está persistente.

27. Describe brevemente las clases (o interfaces) principales que ofrece el API JPA: cuáles son, qué misión tienen, y cómo se interrelacionan.

    - **EntityManagerFactory**: Esta es una clase de fábrica de EntityManager. Crea y gestiona múltiples instancias EntityManager.
    - **EntityManager**: Es una interfaz, que gestiona la persistencia de objetos. Funciona como una factoría para instancias de la clase Query.
    - **Entity**: Las entidades son los objetos de persistencia, tiendas como registros en la base de datos.
    - **EntityTransaction**: Tiene una relación de uno a uno con EntityManager. Para cada EntityManager, se mantienen operaciones por la clase EntityTransaction.
    - **Persistence**: Esta clase contiene métodos estáticos para obtener instancias de EntityManagerFactory.
    - **Query**: Esta interfaz es implementada por cada proveedor JPA para obtener objetos relacionales que cumplan los criterios.
    - *Cómo se interrelacionan.*
    - La relación entre EntityManagerFactory EntityManager es de uno a varios. Se trata de una clase de fábrica para instancias de EntityManager.
    - La relación entre método EntityManager y EntityTransaction es uno a uno. Para cada operación de EntityManager, hay una instancia de EntityTransaction.
    - La relación entre EntityManager y Query es de uno a varios. Muchas consultas se pueden ejecutar mediante una instancia EntityManager.
    - La relación entre Entity y EntityManager es uno de muchos. Una instancia de EntityManager puede administrar varias Entities.

28. ¿Se puede navegar por el grafo una vez cerrado el contexto de persistencia? Matiza la respuesta.

    - Navegar por zonas aún no cargadas del grafo es sólo posible con el contexto de persistencia abierto. Una vez cerrado sólo se puede navegar lo que ya está en memoria (detached)

29. ¿A que puede ser debido que me salte una LazyInitializationException?
    - A intentar cargar un objeto en memoria habiéndose cerrado antes la sesión y, por tanto, no pudiendo acceder a la base de datos
30. ¿De qué formas se puede añadir información de mapeo en JPA?
    - *Existen varias formas:*
    - Si la persistencia es de tipo JDK lo tiene por defecto, el mapeador ya sabe cómo hacerlo
    - Para tablas con herencia: En la clase raíz añadir @DiscriminatorColumn y en la clase hija añadir @DiscriminatorValue.
    - Mapeo de clases asociativas.
    - Mapeo básico de Set poniendo @ElementCollection
    - Mapeo básico de List poniendo @ElementCollection. También podemos poner @OrdenColumn(name = "...") para determinar el orden de aparición.
    - Mapeo básico de Map poniendo @ElementCollection
31. Añadiendo anotaciones de mapeo, ¿cómo se vinculan los dos extremos de una asociación bidireccional? ¿qué pasa si no se hace?
    - En uno a muchos poniendo sobre la colección del extremo uno que guarda del extremo muchos @OneToMany(mappedBy= "nombreDelObjetoEnClaseDelLadoDeMuchos").
    - En muchos a muchos se pone en uno de los extremos @ManyToMany (mappedBy="nombreDeLaCloeccionLaOtraClaseMuchos"). En la otra simlpemente ponemos @ManyToMany.
    - Si no se indica mappedBy se interpreta como dos asociaciones unidireccionales separadas.
32. En JPA, con configuración de mapeo por defecto ¿los extremos MANY de una asociación se cargan de forma agresiva o bajo demanda?, ¿y los extremos ONE?
    - Los MANY bajo demanda y los ONE de forma agresiva.

33. JPA, Hibernate, Eclipse Link. Explica qué relación hay entre esos términos.
    - ava Persistence Api es una especificación, Hibernate y EclipseLink son implementaciones.
34. Tenemos una aplicación desarrollada con mapeador JPA Hibernate funcionando contra una base de datos MySQL. Por alguna razón hay que cambiar de servidor de base de datos, ahora usaremos una Oracle. ¿Cómo lo resolvemos?
    - Modificando los datos de conexión a la base de datos antigua por los de la nueva.
35. ¿De qué formas se pueden especificar y asignar los parámetros de una consulta?
    - Enlace nominal y enlace posicional.
36. Enumera los patrones que aparecen en la realización del CarWorkshop con mapeador.
    - Modelo-vista-controlador (MVC)
    - Data Access Object (DAO)
    - Transaction Script
    - Facade
    - Command
    - Factory Method
37. Estoy escribiendo un comando en el que me están saliendo muchas líneas de código, ¿quées muy probable que estéhaciendo mal?
    - Estás metiendo demasiada lógica o persistencia en el comando en vez de delegarlo en el modelo.
38. Imagina que el CarWorkshop que se ha desarrollado pide login y password a cada usuario que lo usa. Una vez verificada la identidad del usuario su login queda accesible en una variable global de hilo (un ThreadLocal). Ahora nos piden que modifiquemos el programa de forma que guarde en un fichero de log una línea cada vez que un

usuario realiza una acción (que guarde una auditoria). ¿En qué clase añadirías esa funcionalidad de forma que sólo se escriba una vez?

    - En las clases que implementen Command, por ejemplo, AddContrato (implements Command), o en el propio CommandExecutor.

---

1. Suponiendo configuración de mapeo por defecto, ¿funcionará el código mostrado a continuación? En caso negativo, explica el por qué.

Capa de presentación

```
AdminService as = ServiceFactory.adminServices();
Cliente c = as.findClienteById(...);
for (Vehiculo v: c.getVehiculos()){
Printer.print(v);
}
…
```

Capa de servicio

```
class AdminServicesImpl{
…
Cliente findClienteById(Long id){
EntityManager em = getEntityManagerAndOpenTrx();
try{
return em.find(Cliente.class, id);
} finally {
closeTrxEntityManager();
}
}
```

- El código no funcionaria. Esto se debe a que buscamos el objeto en si en la capa de servicio. Como lo buscamos en la base de datos y este estará asociado a esta, el objeto se encontrará en estado detached pero, cuando el método findClienteById(...) termina volvemos a la capa de presentación y el objeto pasa a estado detached perdiendo así su asociación con la base. Por lo que al realizar el bucle for daría un error pues no podemos acceder a los vehículos asociados al cliente.

- ¿Qué devuelve esta expresión?

```
let $i := (1, 2, 3, 4)
return {$i}
```

- 1 2 3 4

9.¿Qué devuelve esta expresión?

```
let $i in (1, 2, 3, 4)
for $j in $i
return {($i, $j)}
```

- 1 2 3 4 1
- 1 2 3 4 2
- 1 2 3 4 3
- 1 2 3 4 4

- Con el grafo de películas, indica brevemente que devuelve esta consulta Cypher:

```
match (p:Person)-[:ACTED_IN1…3]-(m:Movie)<-[ACTED_IN]-
(a:Person)
where p.name='Kevin Bacon'
return m.title, a.name, count()
```

- Creo que devuelve el título, de las tres primeras películas como mucho, en las que participara "Kevin Bacon", así junto con el nombre de los actores que trabajaron junto a él y el número de estos.
- Con el grafo anterior de películas, escribir una consulta en Cypher que devuelva las películas que tienen al menos dos directores, directores que a su vez han actuado en la película.

```
Match (p1:Person)-[:DIRECTED]-(m1:Movie)<-[:Directed]-(p2:Person)
(p3:Person)-[:ACTED_IN]-(m2:Movie)<-[: ACTED_IN]-(p4:Person)
where p1= p3 AND p2= p4 AND m1= m2
return m1.title
```

1. ¿Qué fallos encuentras en estas consultas en JQL sobre CarWorkshop?

```
select * from TMecanicos m
select m from Mecanicos m
```

```
select m from mecanicos m
select * from TMecanicos
```

- ¿Correcta? Selecciona todos los mecánicos.
- ¿Incorrecta? Nombre de tabla incorrecto
- ¿Incorrecta? Nombre de tabla incorrecto.
- ¿Incorrecta? No tiene alias (m).
- Sobre CarWorkshop ¿es correcta esta consulta? En caso negativo explica por qué no.

```
select a.vehiculo.cliente
from Averia a
where a.factura.fecha = ‘12/12/2014’
```

- Si es correcta.

```
select f.averias.vehiculo.cliente
from Factura f
where f.fecha = ‘12/12/2014’
```

- No es correcta porque el final del camino no puede ser multivaluado, es decir, no se puede pasar a través de asociaciones oneToMany.