EXAMEN

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`.

- Conectados: JdbcRowSet.
  Mantiene una conexión con el SGBD permanentemente (los cambios en el RowSet se reflejan en la tabla)
- Desconectados: CachedRowSet, WebRowSet, JoinRowSet, FilteredRowSet.
  Se conecta para recibir los datos (del SGBD).
  Se desconecta y permite trabajar.
  Se conecta para enviar los datos (al SGBD)
  Verifica los conflictos que se puedan haber producido y los resuelve.
  Ligero, serializadle y puede transmitir por la red a distintos dispositivos, aplicaciones, etc

3. Según la arquitectura vista en clase SL_TS_TDG enumera las distintas capas que la componen y explica la finalidad de cada una.

- Capa de presentación: gestiona las interacciones con el usuario.
- Capa de servicio/negocio: gestiona toda la lógica.
- Capa de persistencia: gestiona la persistencia.

4. Enumera y explica los distintos niveles de aislamiento.

El nivel de aislamiento se modifica con la función **setTransactionIsolation(tipoAislamiento)**

- **Connection.TRANSACTION_READ_UNCOMMITTED**
  Pueden recuperar datos modificados pero no confirmados por otras transacciones. En este nivel se pueden producir todos los efectos secundarios de simultaneidad (lecturas sucias, lecturas no repetibles y lecturas fantasma)

- **Connection.TRANSACTION_READ_COMMITTED**
  Permite que entre dos lecturas de un mismo registro en una transacción A, otra transacción B pueda modificar dicho registro, obteniéndose diferentes resultados de la misma lectura. Evita las lecturas sucias

- **Connection.TRANSACTION_REPEATABLE_READ**
  Evita que entre dos lecturas de un mismo registro en una transacción A, otra transacción B pueda modificar dicho registro, con el efecto de que en la segunda lectura de la transacción A se obtuviera un dato diferente.

- **Connection.TRANSACTION_SERIALIZABLE**
  Garantiza que una transacción recuperará exactamente los mismos datos cada vez que repita una operación de lectura. Previene las lecturas sucias, lecturas no repetibles y lecturas fantasma

5. Según la arquitectura vista en clase SL_TS_TDG dibuja el diagrama de secuencia de la operación "añadir un contrato".

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

Los objetos solo pueden estar en 3 estados:

- New: el objeto esta 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 el pues necesitaríamos cargarlas desde la base pero no tenemos acceso a ella.

En ambos casos, el objeto se encuentra en estado manager.

7. 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.

8. Sobre CarWorkShop, ¿es correcta esta consulta JQL? En caso negativo explica por qué no.

```
select
f.averias.vehiculo.cliente
                from Facturas f
                where f.fecha = '12/12/2014'
```

9. 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?

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

11. Añadiendo anotaciones de mapeo, ¿cómo se vinculan los dos extremos de una asociación bidireccional? ¿Qué pasa si no se hace?

12. ¿Qué devuelve esta expresión?

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

<sal> 1 2 3 4 </sal>

13. ¿Qué devuelve esta expresión?

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

<sal> 1 2 3 4 </sal>
<sal> 1 2 3 4 </sal>
<sal> 1 2 3 4 </sal>
<sal> 1 2 3 4 </sal>

Para las preguntas 14, 15 y 16 considérese el siguiente grafo.

---

14. Listar un título de cada libro junto con el número de autores.

15. listar por orden alfabético los nombres de todos los autores y editoriales.

16. mostrar los libros cuyo precio no supera la media.

17. ¿Además de las bases de datos en grafos, que otros tipos de bases de datos se incluyen habitualmente dentro del grupo de sistemas NoSQL?

18. Resumen brevemente al menos dos inconvenientes que se señalan habitualmente como típicas de las bases de datos NoSQL?

19. ¿Qué alternativas existen, para realizar la distribución de datos de una base de datos en un cluster de servidores?

20. ¿Qué es la consistencia eventual?

---

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

```
match (p:Person)-[:ACTED_IN*1…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.

22. 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
```

23. 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.

24. Describe brevemente que son y para qué sirven los Quorum de lectura y escritura en un cluster de servidores.

Preguntas interesantes

Implementar modelo en Java

1-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): Un entidad representa la existencia de "algo" en el dominio (de la realidad) que tiene identidad propia. Sus propiedades pueden cambiar a lo largo del tiempo pero sigue siendo "ella". Puede estar asociada con otras entidades Su ciclo de vida es independiente de otras entidades. Debe tener una identidad (lo que en BDD llamaríamos clave primaria).

Mutables serían todos aquellos atributos que se pudieran modificar en un futuro y rompieran el encapsulamiento.

HashCode y equals redefinido SÓLO sobre los atributos que determinan identidad.

Tipos valor (value type): Representan un valor, no tienen identidad. Ej: Una moneda de 2€ no importa si es esta o aquella, sólo importa que es de 2€. Su valor es inalterable. Se suelen presentar como atributos de una entidad Su ciclo de vida depende enteramente de la entidad que las posee.

Son atributos y son inmutables, no se cambian sus valores (no tienen setters).

HashCode y equals redefinido sobre TODOS los atributos.

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

(Modelos de dominio pag 15)
Utilizamos métodos para el mantenimiento de las asociaciones addXxxx y remove Xxx
ya que estas generarían código repetitivo y eliminan la encapsulación, además de esta manera mantenemos también las referencias cruzadas
(No se si va aquí pero los tipo valor no rompen la encapsulación al devolverlos en los getters y setters)

3-¿Cómo se implementa en Java una clase asociativa UML?

Implementación de modelos de dominio en Java - Diapositiva 23
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 (clases que relaciona xD), y el método unlink, sin parámetros y void, se encargarán de mantener la asociación (en el primero se hace el add y en el segundo el remove).
Finalmente en las clases correspondientes a cada identidad se crean dos getters: get público que será accesible por el resto de la aplicación; _get de paquete: será accesible solo por la clase asociativa.

4-¿Qué parámetros mínimos debe recibir el constructor de una Entidad?

Todos aquellos que formen la entidad, es decir lo que en BDD llamaríamos clave primaria.

5-¿Qué parámetros debe recibir el constructor de un ValueType?

No está en las diapos, pero en Modelos de dominio diapo 21 que no pueden tener setters al ser inmutables, por tanto supongo que deben recibir como parámetros toda la información o valor que vaya a tener.

6-¿Qué métodos no debe tener la implementación de un ValueType?

Implementación de modelos de dominio en Java - Diapositiva 21

Setters, ya que son inmutables y no se rompe la encapsulación al devolverlos en los getters.

7-¿Por qué los ValueTypes deben ser inmutables?

Para así no romper el encapsulamiento.

8-¿Qué atributos no deben cambiar nunca en una Entidad?

Modelos de dominoo 29

El atributo (o combinación) que permite distinguir a esa entidad de las demás (un objeto de otro, como la clave primaria en las tablas). Tambien debe incluir todos los tipos valor (no tiene setter, nunca cambian)

Para una factura seria vehiculo, mecanico (valuesType) y fecha que no cambiará.

9-En la implementación de una entidad ¿A qué llamabamos atributos naturales?

Aquellos que pertenecen a la misma.

10-¿Para qué es útil redefinir el método toString() de las clases del modelo?

Es útil para depuración.

11-Describe las diferencias entre los estados “Transient”, “Persistent” y “Detached”.

estion de objetos persistentes diapo 3

- 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.

12-Un mapeador de objetos a relacional (O/R) debe resolver las diferencias entre los dos paradigmas.

A− ¿Cómo resuelve un mapeador JPA la herencia?

Mapeo de clases- Diapositiva 57

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).

B− ¿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.

C− ¿Cómo resuelve la detección de la identidad?

Se mapean con la etiqueta @Id la clase identidad

Un EntityManager(gestor de persistencia), que es la sesion (El ciclo de vida tiene lugar en la memoria de la JVM siendo una cache de primer nivel) Garantiza la identidad java y la identidad en BDD haciendo que no haya varios objetos en sesión representando la misma fila.

Gestion de objetos...5

La identidad se garantiza porque una fila de la BBDD sólo se carga una vez y es representada por un único objeto java por contexto de persistencia

Si existe otro objeto persistente con misma identidad BDD. Copiar detached en persistente. Si existe en BD cargar y actualizar datos con los del detached Si no esta en BDD es objeto nuevo, se hace persistente

D− ¿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.

E− ¿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 de muchos pondriamos un set). Y en el extremo con la flecha no haría flaa instanciar nada.

Bidirecional:

Sería igual que la unidirecional, con al diferencia de que en ambos casos si deberíamos instanciar un atributo haciendo referencia a la otra entidad.

- ¿Cómo resuelve la concurrencia?

13-En un programa concurrente ¿cuántas copias parciales del grafo habrá en memoria?

¿¿¿¿¿¿?????

14-¿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. El problema de las n+1 consultas.

15-Para acelerar el acceso al grafo los mapeadores implementan una caché. ¿Qué características tiene esa caché?

es una caché de primer nivel(sesion) es un gestor de persistencia Garantiza la identidad java y la identidad en BDD. Se lleva a cabo una unidad de trabajo. Al final de la unidad de trabajo se sincroniza con la BBDD

16-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?

Un commit. ¿?¿?¿?¿?¿?

Marcos dice: NADA!? Ya está persistente.

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

- **Hibernate:** herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java. Facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.
- EclipseLink, TopLink, CocoBase, OpenJPA, Kodo, DataNucleus, Amber.

18-¿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)

19-¿A que puede ser debido que me salte una LazyInitializationException?

http://cursohibernate.es/doku.php?id=unidades:06_objetos_validaciones:01_trabajando_objetos#lazyinitializationexception
A intentar cargar un objeto en memoria habiéndose cerrado antes la sesión y, por tanto, no pudiendo acceder a la base de datos

20-¿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 como 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

21-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 al otra simlpemente ponemos @ManyToMany.

Si no se indica mappedBy se interpreta como dos asociaciones unidireccionales separadas.

22-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?

Consultas en JPA - Diapositiva 38
Los MANY bajo demanda y los ONE de forma agresiva.

23-Un alumno se matrícula en varias asignaturas. En una asignatura se matricularán muchos alumnos. Para cada asignatura matriculada se obtiene una nota final. Dibuja el

modelo UML. Escribe el código java mínimo para representar esa situación y las anotaciones de mapeo necesarias.

24-JPA, Hibernate, Eclipse Link. Explica qué relación hay entre esos términos.

Java Persistence Api es una especificación, Hibernate y EclipseLink son implementaciones.

25-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.

26- ¿De qué formas se pueden especificar y asignar los parámetros de una consulta?

Enlace nominal y enlace posicional.

27-¿Cómo se llama ese estilo de configuración de una Query a base de llamar a métodos que siempre devuelven el mismo objeto?

28- ¿En qué estado están los objetos devueltos por una consulta JQL?, ¿y el que devuelve el método find?

Los objetos solo pueden estar en 3 estados:

- **New:** el objeto esta 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 el pues necesitaríamos cargarlas desde la base pero no tenemos acceso a ella.

En ambos casos, el objeto se encuentra en estado manager.

29-¿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
```

1. Correcta? Selecciona todos los mecánicos.
2. Incorrecta? Nombre de tabla incorrecto
3. Incorrecta? Nombre de tabla incorrecto.
4. Incorrecta? No tiene alias (m).

30-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'
```

31-Sobre CarWorkshop ¿es correcta esta consulta? En caso negativo explica por qué no.

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

32-Enumera los patrones que aparecen en la realización del CarWorkshop con mapeador.

Command Executor, Fachada y Factoria.

33- Estoy escribiendo un comando en el que me están saliendo muchas líneas de código, ¿qué es muy probable que esté haciendo mal?

Estás metiendo demasiada lógica o persistencia en el comando en vez de delegarlo en el modelo. (OO)

34-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.

35-. 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.

En el código de un comando: Tengo un objeto "Vehiculo" obtenido con un find() y necesito todas sus averías ¿cómo me hago con ellas? Comenta las posibilidades y señala la más sencilla. En el código de una clase Action: Tengo un objeto "Vehiculo" obtenido con una llamada a la capa de lógica y necesito todas sus averías ¿cómo me hago con ellas? Comenta las posibilidades y señala la más sencilla.