Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Ingeniería en Sistemas de Información

AÑO 2016

Materia: Diseño de Sistemas

Código de Materia: 082028

Curso: K-3054

Docentes:

Mur, Pablo
Oliva, Miguel
Procopio, Demian
Rico Mendoza, René
Sosa, Ezequiel
Valido, Leandro

Trabajo Práctico: *Persistencia*

Tipo: *Grupal*

GRUPO Nº 11	
NOMBRE Y APELLIDO	**LEGAJO Nº**
Salomone, Cecilia	112280-0
Jorge, Martinez Rodriguez	140831-8
Francisco Enzo,DiGiorgio	149744-3
Walter Hernan, Aguilar	127938-5
Javier, Salvatella	111320-3

Fecha prevista de entrega: 11/ 10/ 2016

Fecha real de entrega: 11/ 10/ 2016

Calificación..... **Firma**.....

Versión <1.0>

Historia de revisión

Fecha	Descripción	Autor	Versión
11/10/2016	-	-	1.0

Tabla de Contenidos

1 Enunciado

Entrega Persistencia

Partiendo de la implementación o especificación existente, ud. debe diseñar el mapeo entre el modelo de objetos de dominio (puntos de interés, búsquedas con su criterio, etc.) y un esquema relacional.

- Para cada uno de los siguientes ejemplos de impedance mismatch debe elegir un caso concreto y explicar qué decisión de diseño tomó
  - implementación relacional de la herencia en objetos
  - manejo de identidad
  - relaciones
    - uno a muchos
    - muchos a muchos
    - recursivas hacia la misma entidad
  - Presten atención a este punto sobre todo los que trabajan en .NET, ya que de elegir el Entity Framework muchas de estas decisiones están preestablecidas. Aunque las tablas y la forma de persistir se haya generado automáticamente deben explicar en un documento cada uno de estos items.
- Implementar la solución utilizando un framework ORM. Se recomienda Hibernate1 para Java, de manera tal que los ayudantes puedan dar soporte. **La base de datos a utilizar es Mysql2 y su uso es obligatorio** . Los que realicen su TP con .NET coordinan estos temas con su ayudante.

● Explicar

○ ¿Qué cambios tuvieron que realizar en el diagrama de clases ?

○ ¿Qué agregaciones/desnormalizaciones fueron realizadas?

○ ¿Qué objetos NO fueron persistidos?

● Se recomienda el uso de https://www.mysql.com/products/workbench/ para verificar el comportamiento del orm y hacer ingenieria inversa de las tablas.

Alcance

● Entidades y requerimientos de las entregas 1,2 y 3

● NO hace falta persistir los procesos ni el contenido de la cache, pero si que origenes de datos externos se utilizan para consultar.

Casos de prueba Mínimos

● Todos los tests realizados hasta el momento deben seguir funcionando sin depender de la base de datos!!!

● Hacer estas pruebas en archivos separados, no mezclarlas con las que tienen hasta el momento.

● El código de negocio NO debe hacer ninguna referencia a Session o SessionFactory, pero
sí pueden crear nuevos objetos que las contengan.

● El esquema básico para una prueba es:

○ Inicializar la base de datos (en principio recomendamos usar hbm2ddl3 en *create* pero a medida que comprueben que están guardando correctamente, por una cuestión de rendimiento, pasen a *validate*). Esto deberían hacerlo en setUp o setUpClass según mejor les quede.

○ En el setUp abrir una Session

○ Crear / Traer de la base uno o varios objetos (al principio conviene chequear que lo que trajeron es lo que esperaban)

○ Realizar alguna operación y guardar los objetos

○ Cerrar la Session (al principio, en este punto conviene mirar qué tablas/filas fueron modificadas en la base de datos)

○ Abrir una nueva Session y seleccionar los objetos que se modificaron por el proceso.

○ Verificar que el estado de el/los objetos consultados sea el esperado

- Ejemplo
    - setUp: Abrir una Session y guardar un escenario(conjunto de objetos) en la DB
    - Realizar una consulta por palabra clave de locales y ver que aparecen x resultados
    - Quitar la palabra clave de uno de los PDIs y guardarlo.
    - Cerrar la Session, volverla a abrir y hacer la consulta
    - Verificar que haya x 1 resultados
- Hay un ejemplo sencillo de esto con 2 entidades (Casa y Habitación) subido al web campus. Cualquier duda mandenla al foro.

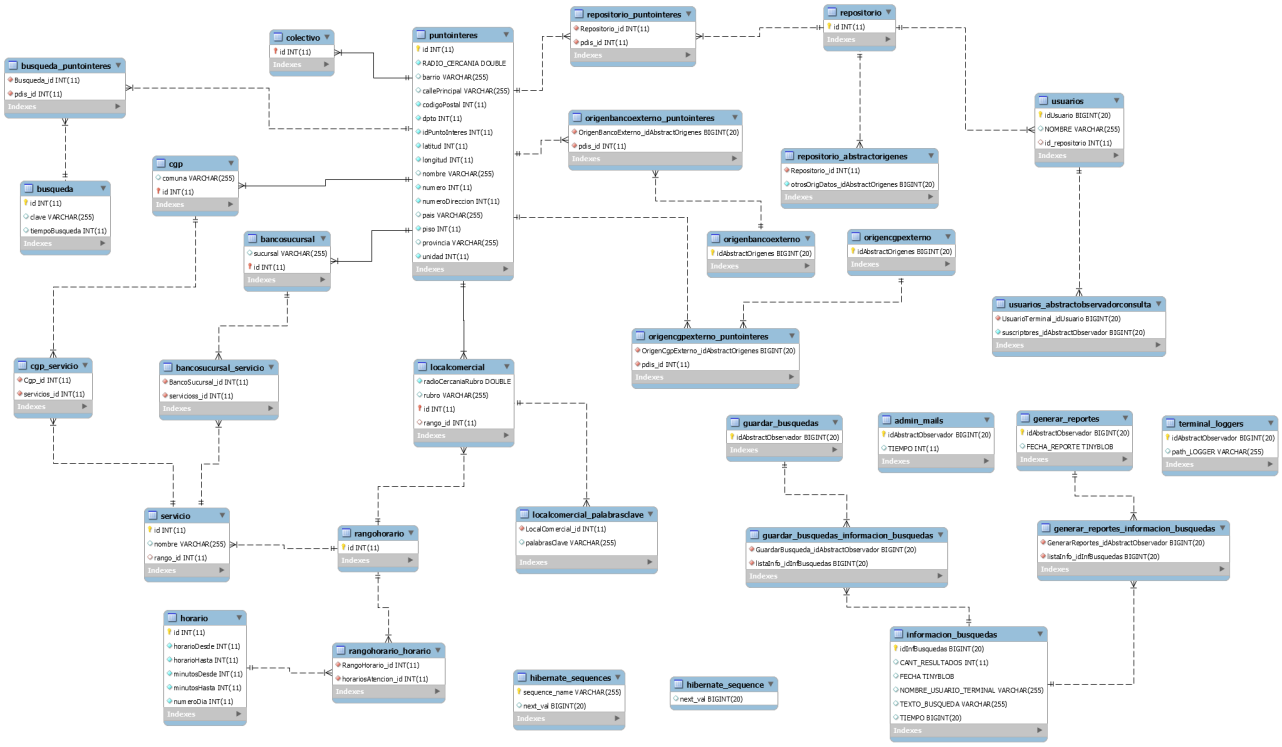
1) Hacer un ABM prueba por clase/objeto a guardar. Conviene hacer esto antes que nada

2) A un Mapa/Repositorio sin orígenes de datos externos, realizar una consulta de PDIs. Luego agregarle un origen de datos (ej: Bancos), guardar, volver a cargar el repositorio y realizar la misma consulta (debería modificarse la lista a retornar)

3) Probar el efecto de cada uno de los procesos de la entrega 4

2 Desarrollo

1. Diagrama Entidad Relación



2. Impedance mismatch

a. <u>Implementacion relacional de la herencia de objetos:</u>
Dada la estructura de herencia: Clase Abstracta PuntoInteres con sus clases concretas hijas Cgp, LocalComercial, BancoSucursal y Colectivo, se decidió implementar en Hibernate la estrategia JOINED (@Inheritance(strategy=InheritanceType.JOINED)).
Si bien es más costoso el INSERT dado que afecta a 2 tablas, pensando en un mapa el cual tendrá probablemente millones de POI y recibió infinidad de consultas, posiblemente concurrentes, es mejor distribuir los registros en varias tablas, que dejar en una sola tabla a todos juntos, lo que daría como resultado una única tabla muy pesada. De este modo esta estrategia prepara el sistema con mayor performance para las búsquedas y/o consultas a realizar en la BD.
Por otra parte no se elige la estrategia clase concreta, dado que hay muchos atributos compartidos entre las clases hijas, los cuales figuran en la clase padre PuntoInteres, evitando así la duplicidad de información.

b. <u>Manejo de identidad</u>:
Para la identidad se optó por la estrategia de generación de id AUTO (@GeneratedValue(strategy=GenerationType.AUTO)). De este modo se genera un id numérico en forma automática cada vez que se crea un nuevo registro en la BD.

c. <u>Relaciones</u>:

i. Uno a muchos:
   Un ejemplo de implementación de esta relación ocurre en el mapeo dado entre la clase Repositorio y PuntoInteres.
   Se utilizo el tipo de relacion OneToMany (@OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)) en la clase Repositorio haciendo referencia a la lista de objetos PuntoInteres que contiene.
   Otro ejemplo de implementacion de esta relacion ocurre en el mapeo dado entre las clases Cgp y Servicio. Dado que un Cgp puede tener muchos servicios.

ii. Muchos a muchos:
   No utilizadas al momento.

iii. Recursivas:
   No utilizadas al momento.

3. Implementar la solución utilizando un framework ORM

Se utiliza Hibernate versión 5.2.2.Final.

4. Explicar

a. ¿Qué cambios tuvieron que realizar en el diagrama de clases?

   En el caso de la persistencia de las acciones y de los origenes de datos, al tratarse de listas de interfaces, no se puede hacer la persistencia normalmente pues Hibernate no lo soporta. Entonces la solución que se buscó fue crear una clase abstracta de la que heredan todas las clases que implementan la interfaz. Mediante @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) .De esta forma se puede persistir las clases que implementan la interfaz.
   En la clase loggerTerminal se agregó un nuevo atributo para guardar el path del log.

b. ¿Qué agregaciones/desnormalizaciones fueron realizadas?
   En relacion a la estructura de los POIs, se debe crear una clase Horario, la cual debe estar relacionada desde la clase RangoHorario, dado que la misma contiene un objeto del tipo Triplet para manejar los horarios, y no se encontró el recurso desde Hibernate para interpretar dicha estructura.

c. ¿Qué objetos NO fueron persistidos?
   En relacion a la estructura de los POIs: Polygon, Point.
   En relación a las acciones: el log, el mock del mail, la cantidad de logs