**UNIVERSIDAD DEL QUINDÍO**
**FACULTAD DE INGENIERÍA**
**PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**

| **Información general** | |
|---|---|
| **Fecha :** | 2015-02-18 |
| **Duración estimada en minutos:** | 240 |
| **Docente:** | Christian Andrés Candela |
| **Guía no.** | 05 |
| **Nombre de la guía:** | Entidades – Relaciones Entre Entidades |

**Información de la Guía**

**Objetivos:** Estudiar el uso de las entidades, las relaciones entre estas y su aplicación en el modelamiento de datos.

**Conceptos Básicos:** Manejo de Eclipse, Java, Bases de Datos, JDBC, XML, GlassFish

**Contextualización Teórica**: Uno de los aspectos más importantes es la definición de relaciones entre entidades. Las relaciones pueden ser bidireccionales o unidireccionales, es decir, se puede establecer la relación solo en una de las entidades (unidireccional), o se puede establecer la relación en ambas entidades a través de atributos permitiendo así la navegación en ambos sentidos (bidireccional).

En toda relación existe una entidad que hace las veces de propietaria de la otra (owner), este aspecto de las relaciones define como se propagan las modificaciones. En las relaciones bidireccionales el lado opuesto al propietario es conocido como inverso, y debe referenciar al propietario de la relación usando el elemento mappedBy en la anotación de la relación.

Existen 4 tipos de relaciones:

- Uno a uno: Para la creación de las relaciones uno a uno se debe marcar el o los atributos de la relación con la anotación javax.persistence.OneToOne. Este tipo de relación se establece entre entidades donde la relación entre la una y la otra es exclusiva. En este tipo de relación la entidad propietaria es quien referencia la otra. Ejemplo un país tiene un presidente. Existe una relación uno a uno entre país y presidente. Para este caso, la entidad propietaria es el país.

```
@Entity public class Pais implements Serializable{
        …
        @OneToOne
        private Presidente presidente;
}

@Entity public class Presidente extends Persona{
        …
        @OneToOne(mappedBy=”presidente”)
        private Pais país;
}
```

- Uno a muchos: Para la creación de la relaciones uno a muchos se debe marcar el o los atributos de la relación con la anotación javax.persistence.OneToMany. Este tipo de relación se establece entre entidades donde una de las entidades se relaciona con varias instancias de la otra entidad. Para esta relación la entidad propietaria al igual que en la anterior es quien referencia a la otra, es decir, quien esta del lado de los muchos. Ejemplo un país tiene muchos departamentos. En este ejemplo existe una relación uno a muchos entre país y los departamentos. Para este caso, la entidad propietaria es el departamento.

```
@Entity public class Pais implements Serializable{
        …
        @OneToMany(mappedBy=”pais”)
        private List<Departamento> departamentos;
}

@Entity public class Departamento implements Serializable{
        …
        @ManyToOne
        private Pais país;
}
```

- Muchos a uno: Para la creación de las relaciones muchos a uno se debe marcar el o los atributos de la relación con la anotación javax.persistence.ManyToOne. Este tipo de relación se establece entre entidades cuando varias instancias de una de las entidades están relacionadas con una instancia de la otra entidad. Para esta relación la entidad propietaria al igual que en la anterior es quien referencia a la otra, es decir, quien esta del lado de los muchos. Por ejemplo varios automóviles pueden ser de la misma marca. Para este caso la entidad propietaria es el Automovil.

```
@Entity public class Marca implements Serializable{
        …
        @OneToMany(mappedBy=”marca”)
        private List<Automovil> automoviles;
}

@Entity public class Automovil implements Serializable{
        …
        @ManyToOne
        private Marca marca;
}
```

- Muchos a muchos: Para la creación de las relaciones muchos a muchos se debe marcar el o los atributos de la relación con la anotación javax.persistence.ManyToMany. Este tipo de relación se establece entre entidades cuando una instancia de una de las entidades esta relacionada con muchas instancias de la otra, y a su vez, una instancia de la otra entidad esta relacionada con muchas instancias de la otra. En este tipo de entidad es particularmente difícil identificar cual de las entidades es la propietaria de la relación, ya que cualquiera de las entidades puede referenciar a la otra. Por ejemplo un estudiante puede estar registrado en muchas materias, y a su vez en una materia pueden estar registrados muchos estudiantes. Para este caso podemos decir que desde un punto de vista lógico la materia referencia al estudiante, por lo tanto, es la materia la entidad que hace las veces de propietaria de la relación.

```
@Entity public class Materia implements Serializable{
        …
        @ManyToMany
        private List<Estudiante> estudiantes;
}
@Entity public class Estudiante implements Serializable{
        …
        @ManyToMany(mappedBy=”estudiantes”)
        private List<Materia> materias;
}
```

Además de los 4 tipos de relaciones enunciados, existe otro tipo de relación entre entidades el cual es conocido como herencia. Este tipo de relación es muy diferente a los anteriores, y se crea a través del lenguaje por medio del extends. Por ejemplo un presidente es una persona, por lo tanto existe una relación de herencia entre el presidente y la persona.

Al momento de crear la relación se puede hacer uso de 3 diferentes estrategias que si bien no representan una diferencia entre las entidades si origina cambios significativos en como se generan las tablas en la base de datos.

- Una Tabla: En este tipo de estrategia se crea en la base una única tabla sin importar cuantas entidades extiendan de la entidad principal.

```
@Entity
@Inheritance
@DiscriminatorValue("Mamifero")  //OPCIONAL
@DiscriminatorColumn(name="tipo" ) //OPCIONAL
public class Mamifero {
        …
}

@Entity
@DiscriminatorValue("Gato")  //OPCIONAL
public class Gato extends Mamifero{
        …
}
```

- Una tabla por clase concreta: En este tipo de estrategia se crea una tabla por cada entidad presente en la relación salvo en los casos en los que la entidad principal sea marcada como superclase (@MappedSuperclass), en cuyo caso no existirá como tabla. Los atributos de la entidad principal se duplicaran en las tablas de sus subtipos.

```
@Entity
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS )
@MappedSuperclass
public class Mamifero {
        …
}
```

```
@Entity
public class Gato extends Mamifero{
        …
}
```

- Relación por medio de joins: En este tipo de estrategia se crea una tabla por cada una de las entidades que participan de la relación, pero no se duplican los atributos de la entidad principal, en su lugar se crea una llave foránea en las tablas generadas por las entidades hijas.

```
@Entity
@Inheritance(strategy=InheritanceType.JOINED )
public class Mamifero {
        …
}

@Entity
@DiscriminatorValue("Presidente")  //OPCIONAL
public class Gato extends Mamifero{
        …
}
```

**Precauciones y Recomendaciones:** Recuerde verificar que el servidor de aplicaciones soporte el motor de base de datos que usará, de igual forma debe verificar que eclipse este asiendo uso del JDK y no del JRE y recuerde adicionar al workspace el servidor de aplicaciones Glassfish antes de crear cualquier proyecto. También puede ser importante verificar que los puestos usados por Glassfish no estén ocupados (Para ello puede hacer uso del comando ***netstat -npl*** o ***netstat -a***)

**Artefactos:** Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse Luna), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish y el motor de base de datos Mysql.

**Evaluación o Resultado:** Se espera que el alumno logre modelar entidades lógicas de una aplicación y por medio de ellas se cree tablas en la base de datos que las representen de forma adecuada.

## Procedimiento

1. Para empezar será necesario crear una base de datos en mysql.
2. Adicione al workspace el servidor de aplicaciones GlassFish.
3. En eclipse cree un Proyecto de tipo Maven Proyect con el arquetipo proyecto-archetype.
4. Configure la conexión a su base de datos. Si ya posee una puede omitir este paso.
5. Abra el archivo persistence.xml, y configurelo para que las tablas sean creadas.

6. Para continuar con esta guía se hará uso de las clases elaboradas al modelar el sistema propuesto como trabajo final. Se ira identificando y trabajando con las relaciones existentes entre dichas clases.

7. **Relación de herencia**

8. Identifique un par de clases cuya relación sea de herencia, la herencia es un tipo de relación uno a uno entre las clases. En la clase madre o clase principal adicione la anotación @Entity, identifique cual es su llave y marque con el atributo @Id. Tras estos pasos deberá seleccionar el nombre de la clase y en la pestaña JPA Detalis busque las propiedades relacionadas a la "inheritance" y en la propiedad estrategia seleccione seleccione ***Joined.***

9. En la clase hija o secundaria adicione la anotación @Entity. Configurela adecuadamente para que funcione bajo la estrategia seleccionada en el punto anterior. Con esto deberá haber establecido la relación uno a uno de herencia entre ambas clases.

10. Para probar los resultados genere las tablas correspondientes a las entidades creadas. Para ello despliegue su proyecto (corra el proyecto en el servidor de aplicaciones) y verifique que las tablas se hayan creado correctamente.

11. Identifique dentro de sus clases todas aquellas en las que aplique este tipo de relación y configúrelas de forma adecuada.

12. **Relación uno a uno**

13. Identifique dos clases entre las que haya una relación uno a uno. Marque ambas clases como entidades (@Entity) y de igual forma identifique su llave y marquelas usando la etiqueta @Id.

14. Ahora en la entidad propietaria de la relación seleccione el atributo que representa a la otra entidad y en la pestaña JPA Details de clic en el vinculo que dirá click here y seleccione la relación uno a uno. Repita este proceso en la otra entidad perteneciente a la relación.

15. Ahora en la clase secundaria (inverso), seleccione el atributo que representa la otra clase. En la propiedad **Join Strategy** cambie la estrategia a **Mapped by**. Ahora seleccione en el combobox el atributo que representa dicha clase en la otra.

16. Para probar los resultados genere las tablas correspondientes a las entidades creadas. Verifique que las tablas se hayan creado correctamente.

17. **Relación uno a muchos – muchos a uno**

18. Ahora identifique dos clases entre las que haya una relación uno a muchos. En la entidad propietaria debería existir un atributo que representa la instancia de la entidad con la que esta relacionado. De igual forma en la entidad secundaria de la relación debería existir un atributo de tipo lista que contenga las instancias de la otra entidad con las cuales esta relacionada. Marque ambas clases como entidades (@Entity) y de igual forma identifique su llave y marquelas usando la etiqueta @Id.

19. Tome la entidad propietaria de la relación y seleccione el atributo que representa la clase inversa o

secundaria en la relación. En la pestaña JPA Details de clic sobre el vinculo (click here). Seleccione la relación correspondiente (muchos a uno).

20. Ahora en la entidad secundaria seleccione el atributo lista que lo asocia a la otra entidad y en la pestaña JPA Details de clic en el vinculo que dirá click here y seleccione la relación uno a muchos. En la propiedad **Join Strategy** cambie la estrategia a **Mapped by** y seleccione en el combobox el atributo que representa en la instancia de la entidad en la entidad propietaria de la relación.

21. Para probar los resultados genere las tablas correspondientes a las entidades creadas. Verifique que las tablas se hayan creado correctamente.

22. Identifique dentro de sus clases todas aquellas en las que aplique este tipo de relación y configurelas de forma adecuada.

23. **Relación muchos a muchos**

24. Identifique dos clases cuya relación sea de muchos a muchos. Marque ambas clases como entidades (@Entity).

25. Marque las llaves de las clases seleccionadas usando la etiqueta @Id.

26. Determine cual de las entidades pertenecientes a la relación es la propietaria de la misma. En ella seleccione el atributo lista que representa a la otra entidad y en la pestaña JPA Details seleccione el tipo de relación que representa el atributo. En este caso muchos a muchos.

27. Tome la otra entidad y seleccione el atributo lista que representa a la otra clase. En la pestaña JPA Details seleccione el tipo de relación que representa el atributo. En este caso muchos a muchos. En la Propiedad **Joining Strategy** cambie la estrategia a **Mapped By** y en el combobox de atributos que quedará a su disposición seleccione el atributo de la otra entidad que representa la relación con la entidad seleccionada.

28. Para probar los resultados genere las tablas correspondientes a las entidades creadas. Verifique que las tablas se hayan creado correctamente.

29. Identifique dentro de sus entidades todas aquellas en las que aplique este tipo de relación y configúrelas de forma adecuada.