PERSISTENCIA DE OBJETOS EN BASE DE DATOS RELACIONALES

FRANCISCO LEÓN NAJERA
CÓDIGO: 20092295009
CEDULA: 80087371

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS BELTRÁN
FACULTAD DE INGENIERÍA
MAESTRIA EN CIENCIAS DE LA INFORMACION Y LAS TELECOMNICACIONES
BOGOTÁ
2009

1. INTRODUCCION

Recientemente la mayoría de los proyectos de desarrollo de software usan avanzados entornos de programación Orientados a Objetos tales como Java o C# para construir el software de las aplicaciones y usan motores de base de datos relacionales para almacenar los datos.

No significa que no existan otras tecnologías diferentes a la tecnología de base de datos, ejemplos son los archivos planos, documentos XML etc.

De todas maneras, existen algunos retos a la hora de implementar directamente la serialización de un modelo estructural de Clases de la aplicación a la estructura de Tablas Relacionales en un Motor de Base de datos, debido a los diferentes enfoques de modelado que cada entorno tiene. Es diferente del caso de la serialización con otras tecnologías similares a XML, mediante las cuales existe una correspondencia directa entre el modelo estructural UML y el lenguaje de serialización Orientado a Objetos.

2. OBJETIVOS

2.1 Definir estratégias de como implementar Objetos Persistentes mediante Bases de Datos Relacionales.

2.2 Determinar como se establece la correspondencia entre el modelo dinamico de la estructura de clases en una aplicación, y la persistencia de las instancias de estas clases en registros de tablas de bases de datos relacionales.

2.3 Consolidar estratégias para implementar las relaciones estructurales del modelo dinámico de clases en una aplicación en forma de un modelo Relacional de Base de Datos.

3. MARCO TEORICO

3.1 Persistencia de Objetos.

La Persistencia (o serialización) consiste en un proceso de codificación de un Objeto (programación orientada a objetos) en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON, entre otros.

La serie de bytes o el formato pueden ser usados para crear un nuevo objeto que es idéntico en todo al original, incluido su estado interno (por tanto, el nuevo objeto es un clon del original). La serialización es un mecanismo ampliamente usado para transportar objetos a

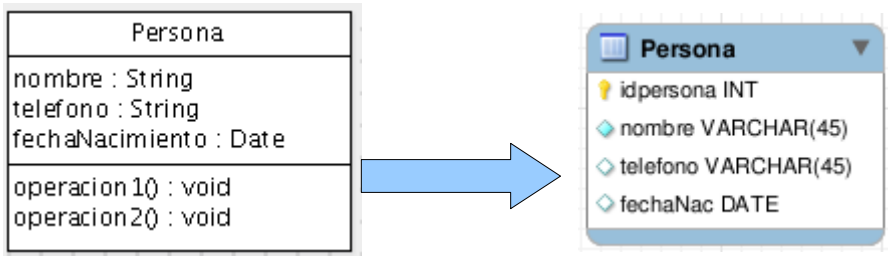
través de una red, para hacer persistente un objeto en un archivo o base de datos, o para distribuir objetos idénticos a varias aplicaciones o localizaciones.

Los mismos objetos pueden tener métodos para serializarse en algun medio fisico, el cual implementa la forma de codificar cada atributo, ya sea simple o compuesto, en su representación de archivo.

Un compendio de estratégias basicas para mapear (o corresponder) clases de objetos persistentes en una base de datos relacional, ha sido definido en el libro de Técnicas Agiles para Bases de Datos de John Wiley.[1]

3.2 Conversión de Clases con Atributos simples a Tablas.

Según Wiley [1] Las clases que tienen unicamente atributos escalares (es decir, no tiene atributos compuestos) , pueden representarse en Entidades con los mismos campos correspondientes:

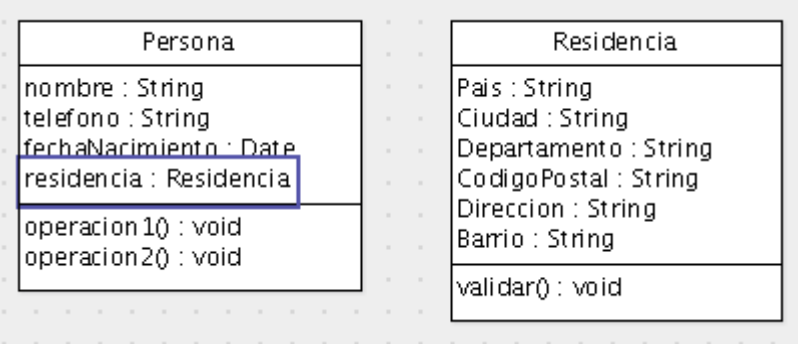


Pasos para implementar la persistencia de una clase con atributos escalares:

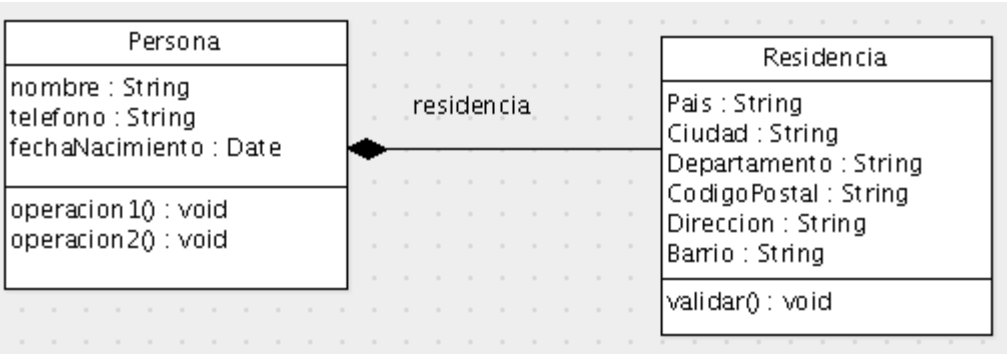
- Se seleccionan los atributos que requieran persistencia. Variables de estado dinamico, banderas, o valores temporales o de cache no requieren almacenarse.
- Se crea la tabla con el nombre de la clase.
- Se inseran las columnas, una para cada atributo.
- Si la clase no define una tupla de atributos que distingan a los objetos unos de otros, se agrega una columna extra con un identificador simple. Un consecutivo.

3.3 Implementación de las relaciones estructurales del modelo de Clases al modelo relacional de las tablas.

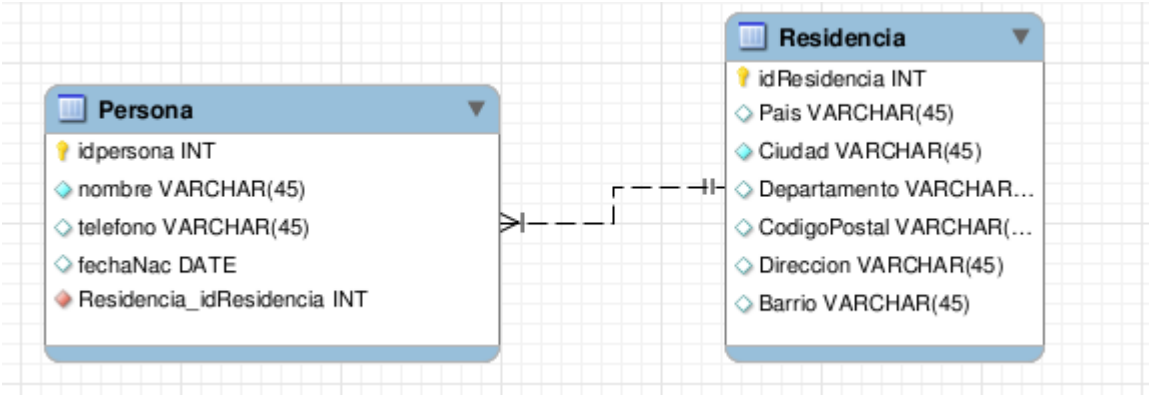
Las clases pueden definir atributos compuestos, osea aquellos que no son de un tipo escalar básico sino son referencias a otros objetos. Por ejemplo el atributo referencia de la clase **Persona** no es un valor escalar sino una referencia a un objeto **Residencia:**



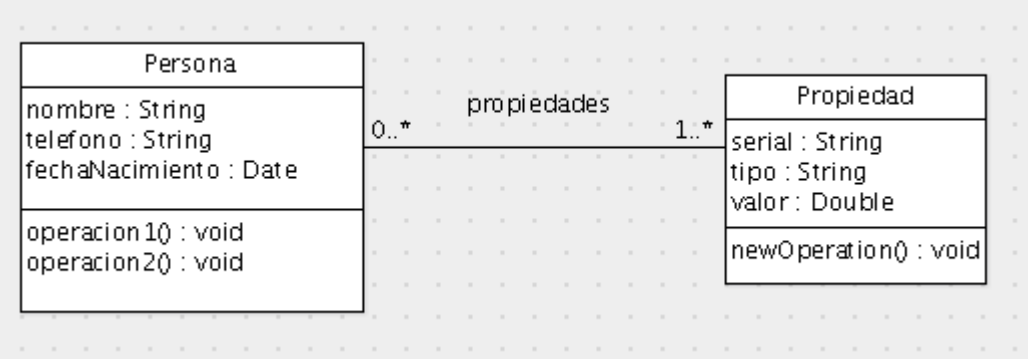
Entonces, el primer paso es abstraer el atributo compuesto como una relación de composición y/o de agregación:



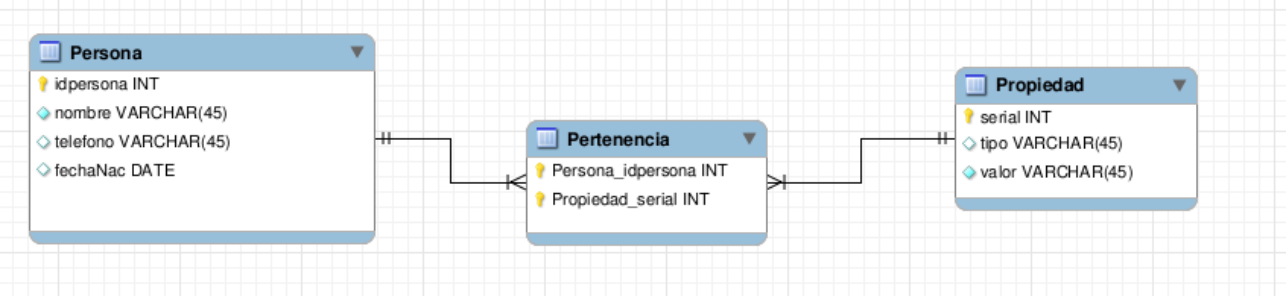
Y entonces en el modelo de Base de Datos Relacional se define una relación **no indentificadora** (relacion debil no nula) en el modelo de tablas:



Otro caso de asociación es el de uno a muchos, por ejemplo, una persona puede tener muchas propiedades, y una propiedad puede tener varios dueños.

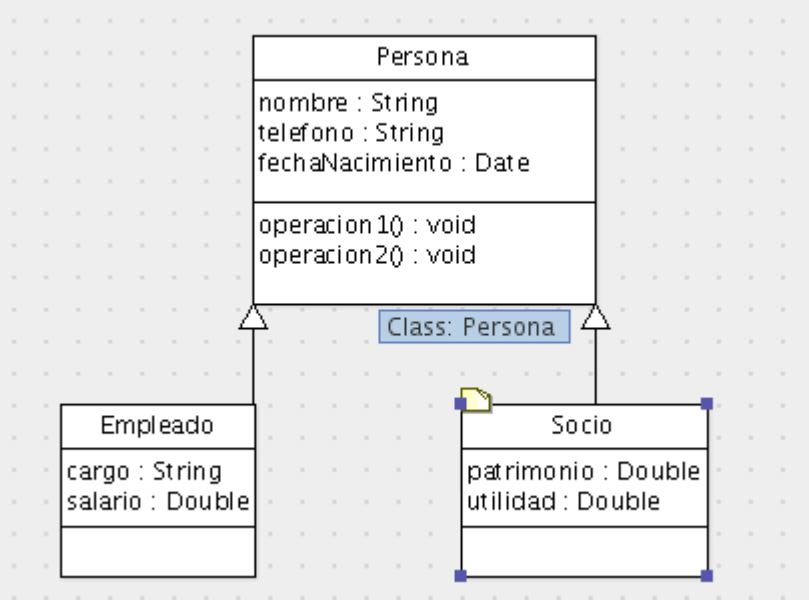


En este caso se tiene que modelar la relación con una tabla que represente la relación entre propiedades y dueños:



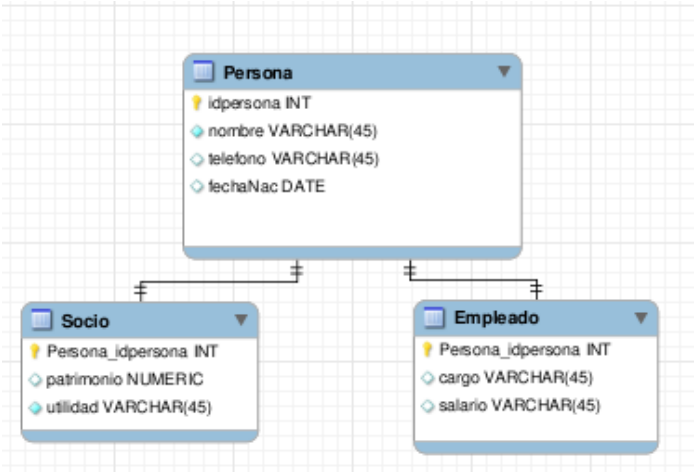
La tabla de relación llamada **Pertenencia** es una entidad hija que mantiene relaciones fuertes identificadoras con Persona y **Propiedad.**

3.4 Implementación de la herencia de Clases a Tablas.

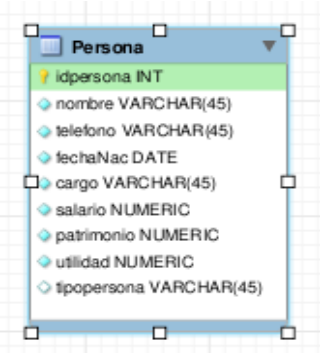


Según Wiley [1], para este caso hay dos alternativas: Una tabla maestra que contenga todos los campos de las clases derivadas y un campo de tipo, o implementar la herencia mediante **composición**.

Herencia por composición



Herencia por tabla monolítica



4. CONCLUSIONES

- Las clases pueden convertirse directamente a tablas siempre y cuando tengan atributos escalares simples.
- Los atributos compuestos deben convertirse en una tabla relacionada con la clase padre.
- Las relaciones de muchos o muchos se implementan con una tabla que relacione las dos entindades
- La herencia se puede implementar mediante la composición.

5. BIBLIOGRAFIA

– [1] Wiley, John. Agile Database Techniques, disponible en http://www.agiledata.org/essays/mappingObjects.html.
– [2] Miller, Jeremy. “Patterns In Practice, Persistence Pattern”, MSDN Library, disponible en http://msdn.microsoft.com/en-us/magazine/dd569757.aspx.