Capa de Persistencia - Fundamentos

Pablo Sánchez

Dpto. Ingeniería Informática y Electrónica
Universidad de Cantabria
Santander (Cantabria, España)
p.sanchez@unican.es

Advertencia

Todo el material contenido en este documento no constituye en modo alguno una obra de referencia o apuntes oficiales mediante el cual se puedan preparar las pruebas evaluables necesarias para superar la asignatura.

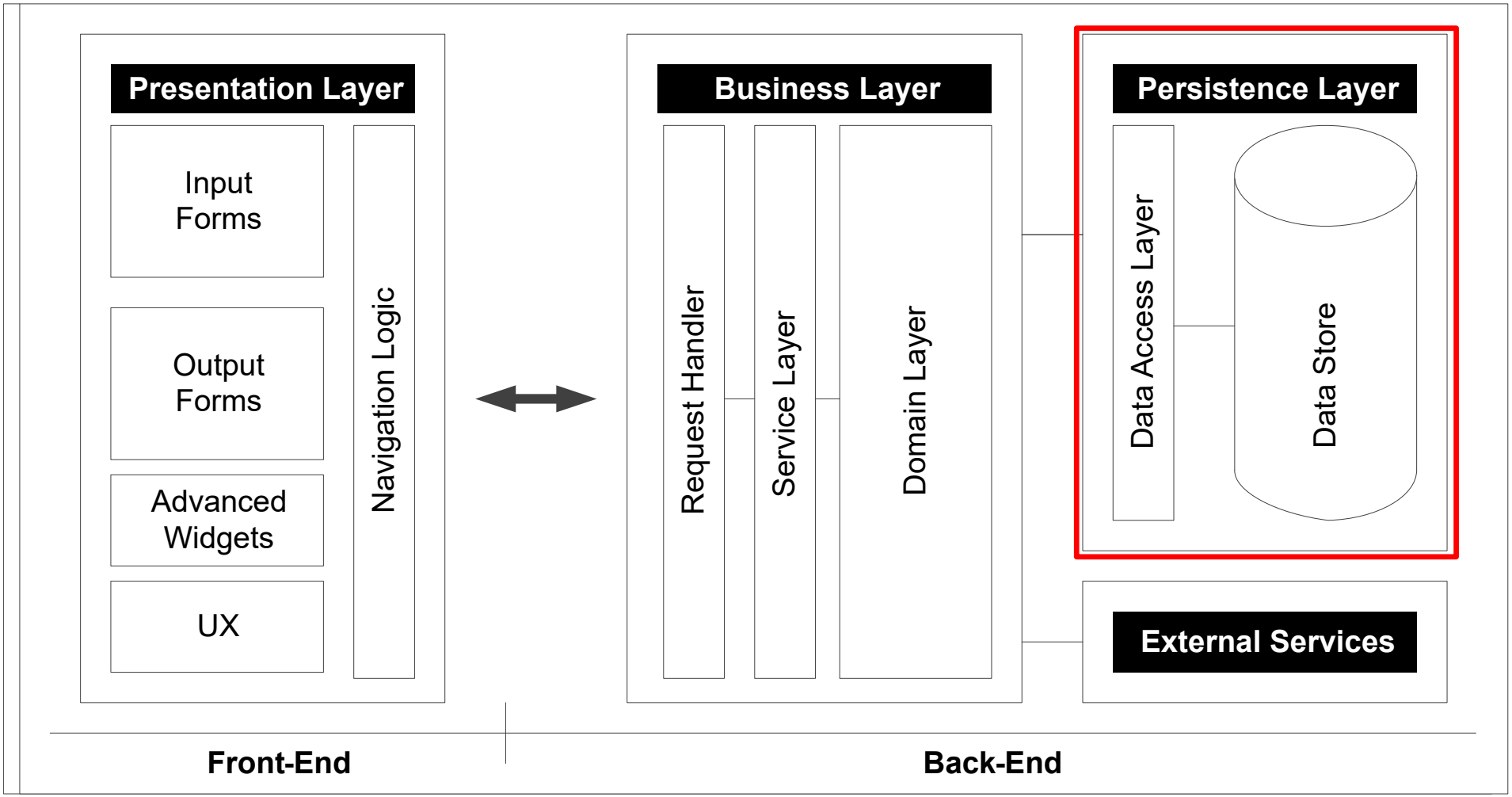
Este documento contiene exclusivamente una serie de diapositivas cuyo objetivo es servir de complemento visual a las actividades realizadas en el aula para la transmisión del contenido sobre el cual versarán las mencionadas pruebas evaluables.

Dicho de forma más clara, estas transparencias no son apuntes y su objetivo no es servir para que el alumno pueda preparar la asignatura.

Índice

Capa de Persistencia

Capa de Persistencia



Responsabilidades de la Capa de Persistencia

1. Almacenar los datos de manera no volátil.
2. Recuperar datos del almacén persistente.
3. Asegurar la disponibilidad de los datos.
4. Controlar la integridad de los datos.
5. Asegurar un acceso eficiente a los datos.

Objetivos del Tema

1. Comprender en profundidad cuáles son las responsabilidades de la capa de persistencia.
2. Comprender el objetivo de los puentes objeto-(relacional).
3. Comprender los principios de los patrones de persistencia estructurales.
4. Comprender el funcionamiento de los patrones de acceso a datos.
5. Ser capaz de utilizar *JPA* para generar esquemas relacionales.
6. Ser capaz de utilizar repositorios *Spring* para acceder a datos.

Bibliografía

Fowler, M. (2002).
Patterns of Enterprise Application Architecture.
Addison-Wesley.

Esposito, D. y Saltarello, A. (2014).
Microsoft .NET - Architecting Applications for the Enterprise. 2ª Ed..
Microsoft Press

Bauer, C., King. G. y Gregory G. (2015).
Java Persistence with Hibernate. 2ª Ed.
Manning

Índice

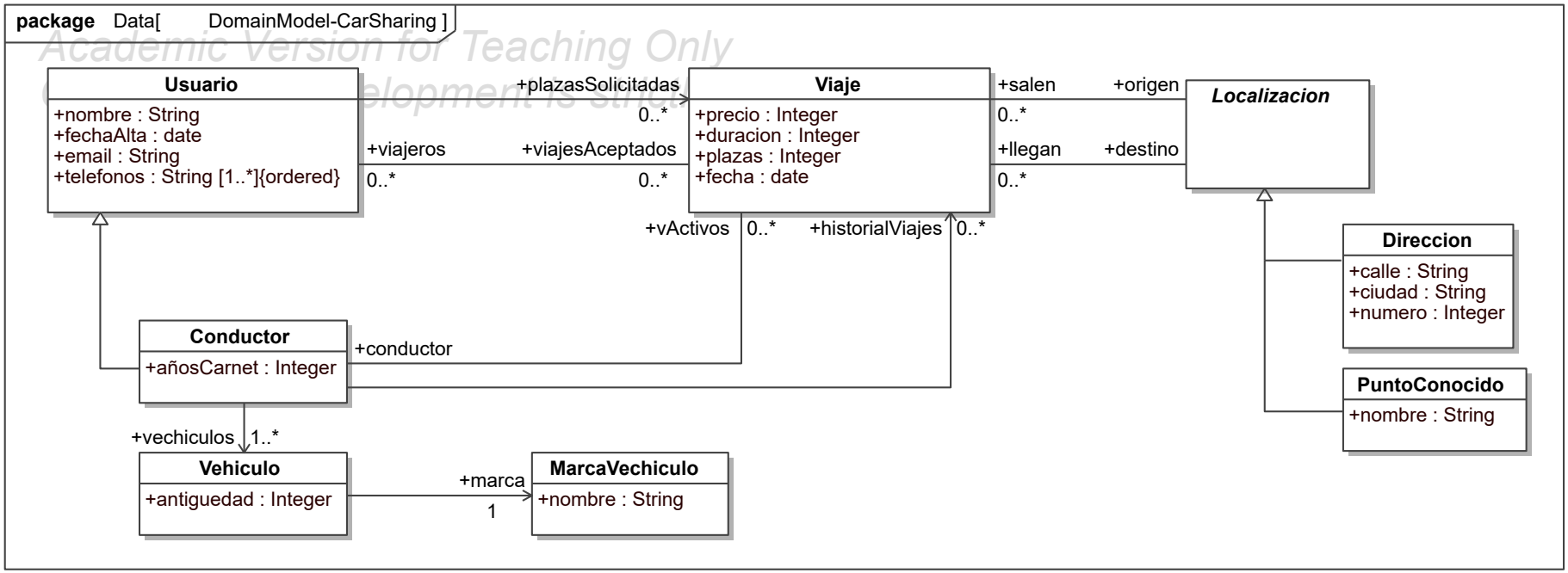
Índice

Impedancia Objeto - (Relacional)

Impedancia Objetual

La *impedancia objetual* se refiere al desacoplamiento que puede existir en los conceptos del paradigma orientado a objetos y el paradigma utilizado por el almacén de persistencia.

Impedancia Objeto - Relacional

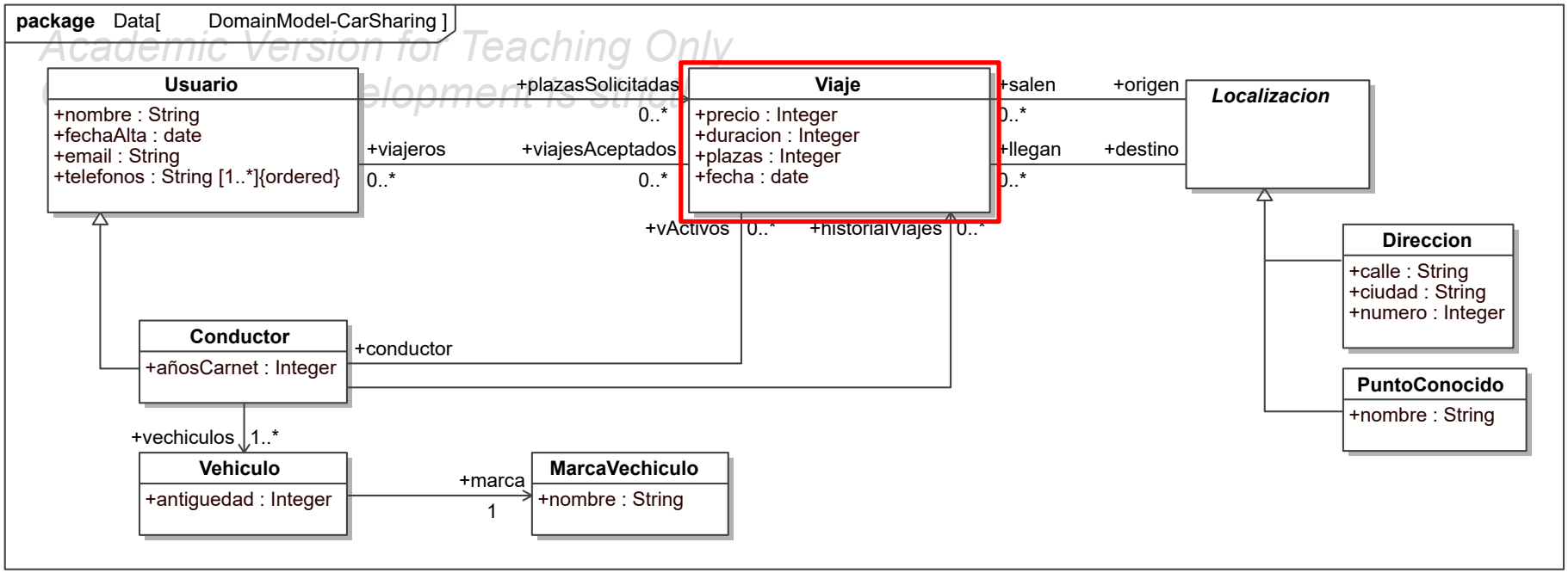


Impedancia Objeto - Relacional

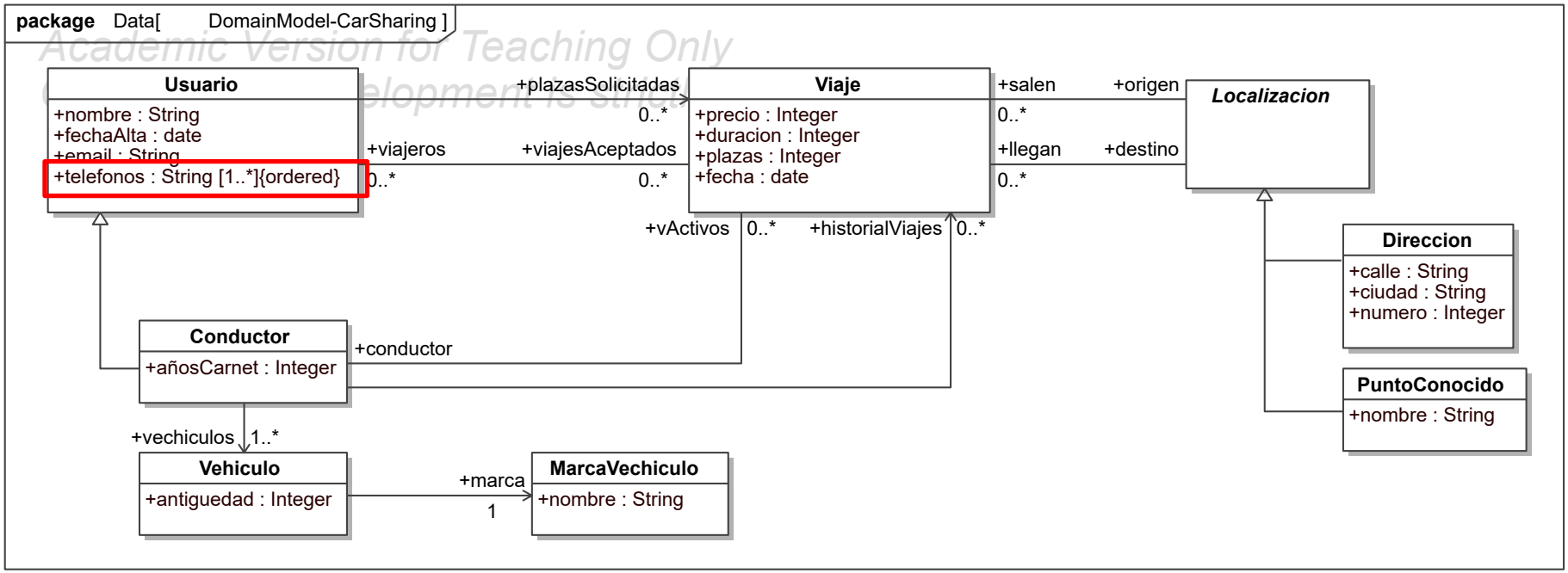
Viaje

precio	duracion	plazas	fecha	

Impedancia OR: Claves Primarias



Impedancia OR: Atributos Multivaluados

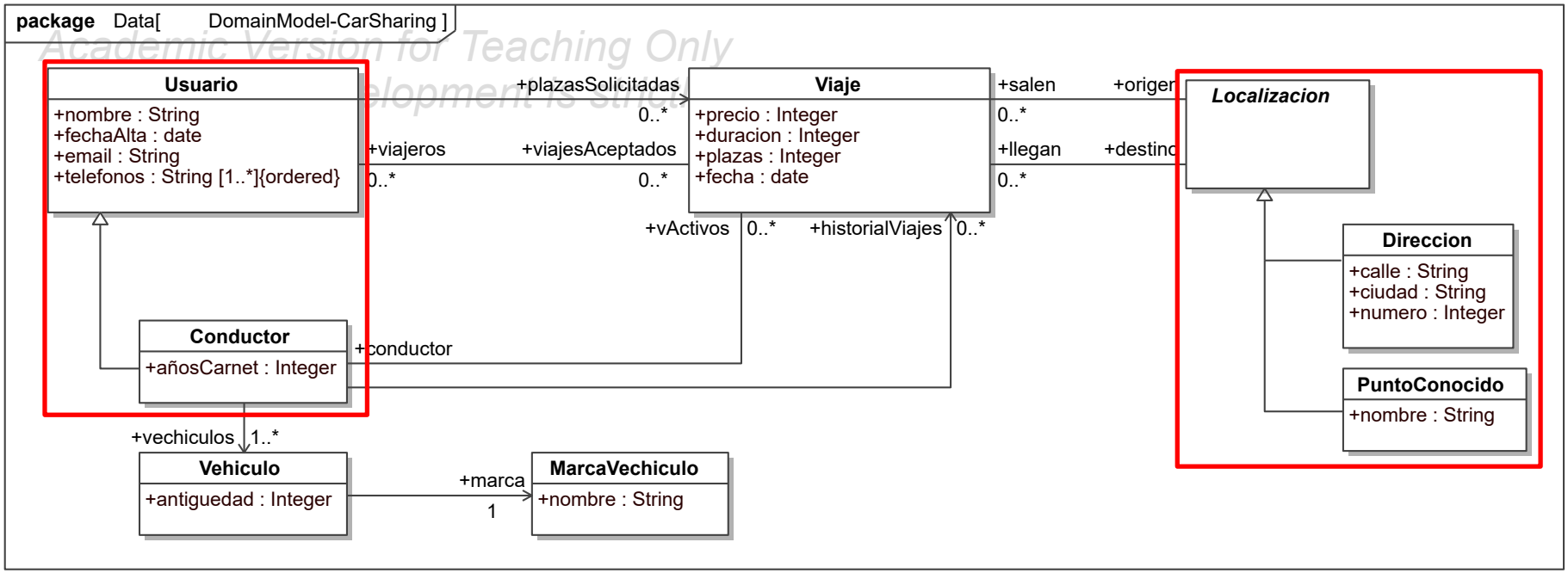


Impedancia OR: Atributos Multivaluados

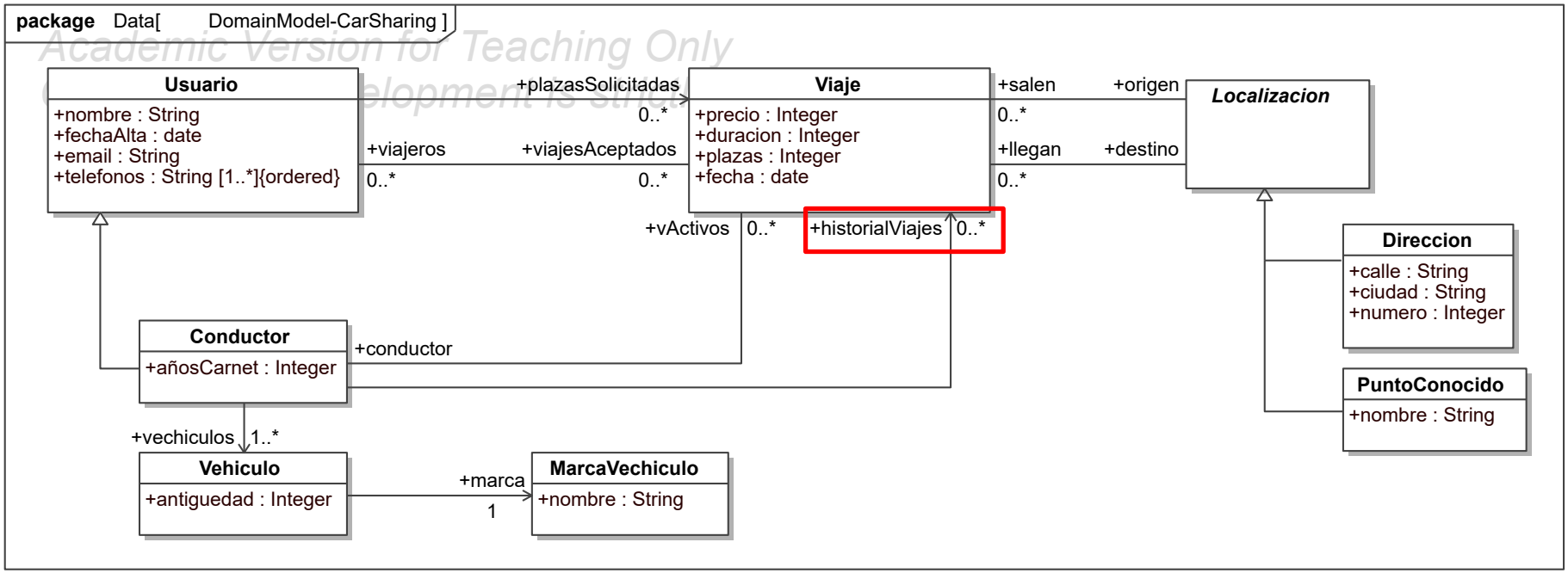
Usuario

nombre	fechaAlta	email	telefono00	telefono02	telefono03	...

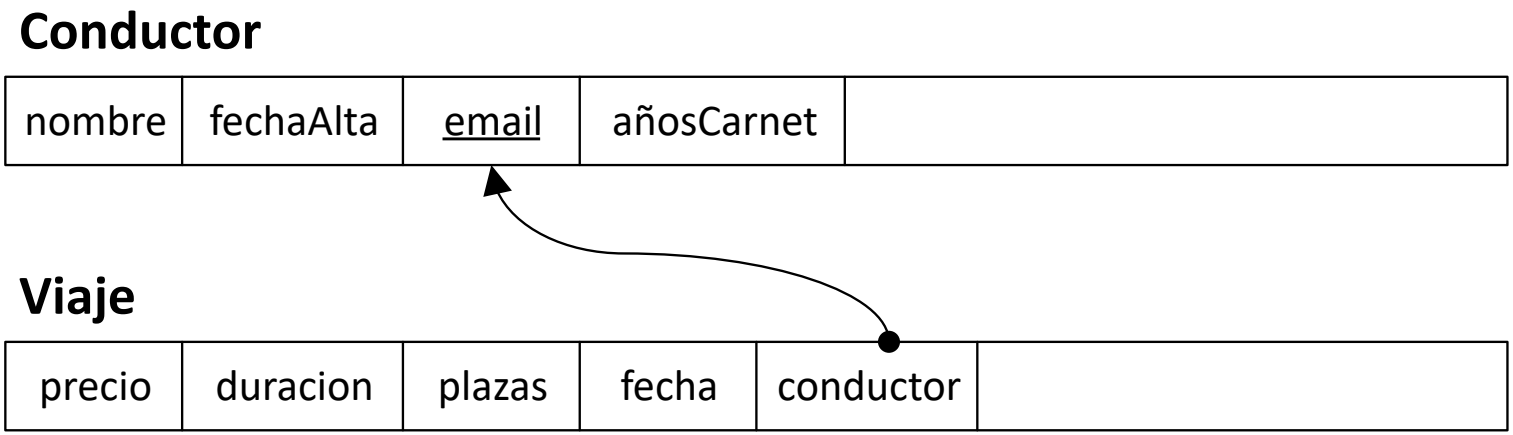
Impedancia OR: Herencia



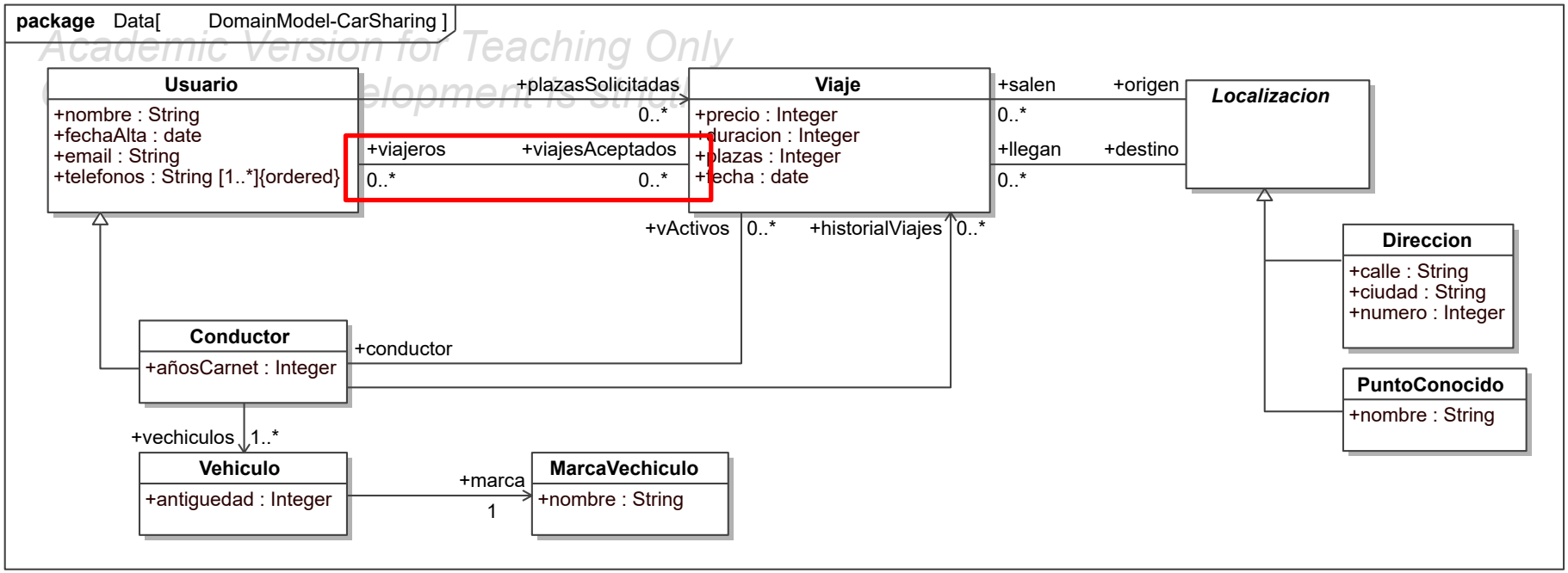
Impedancia OR: Navegabilidad Asociaciones



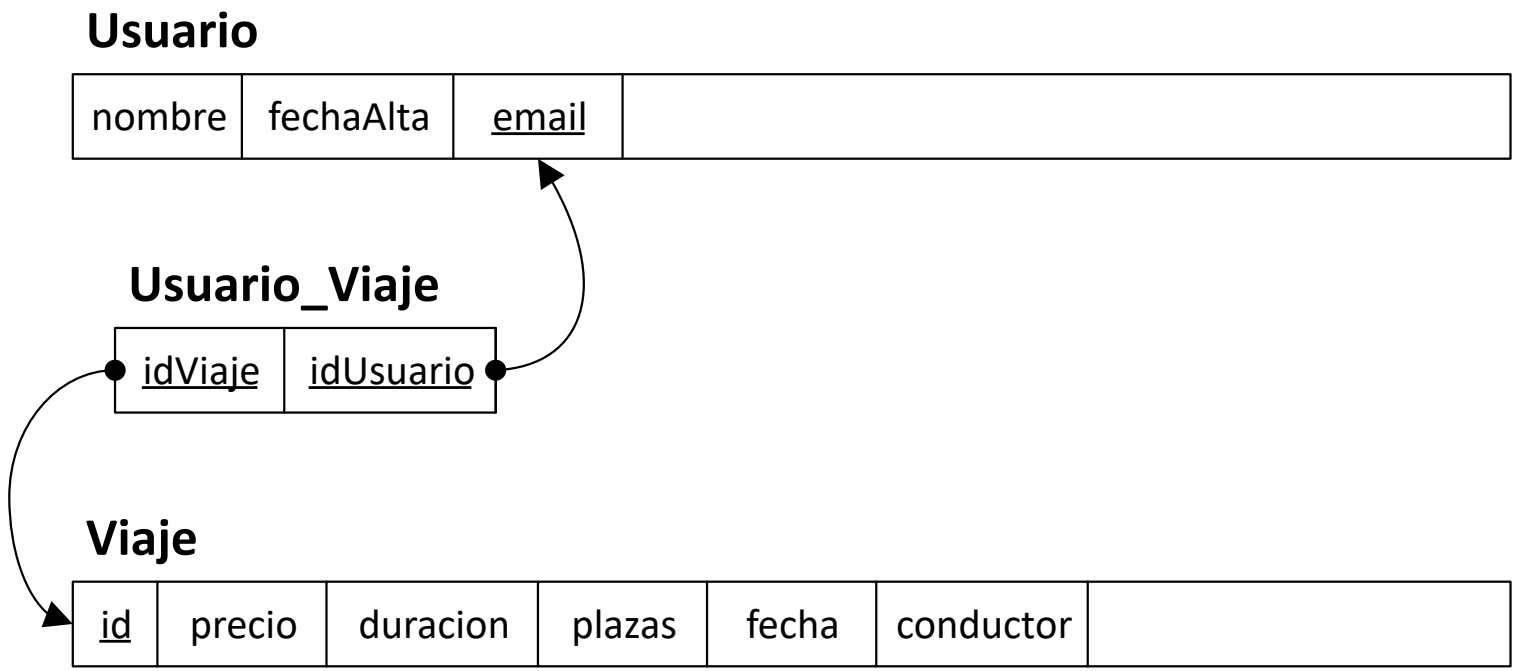
Impedancia OR: Navegabilidad Asociaciones



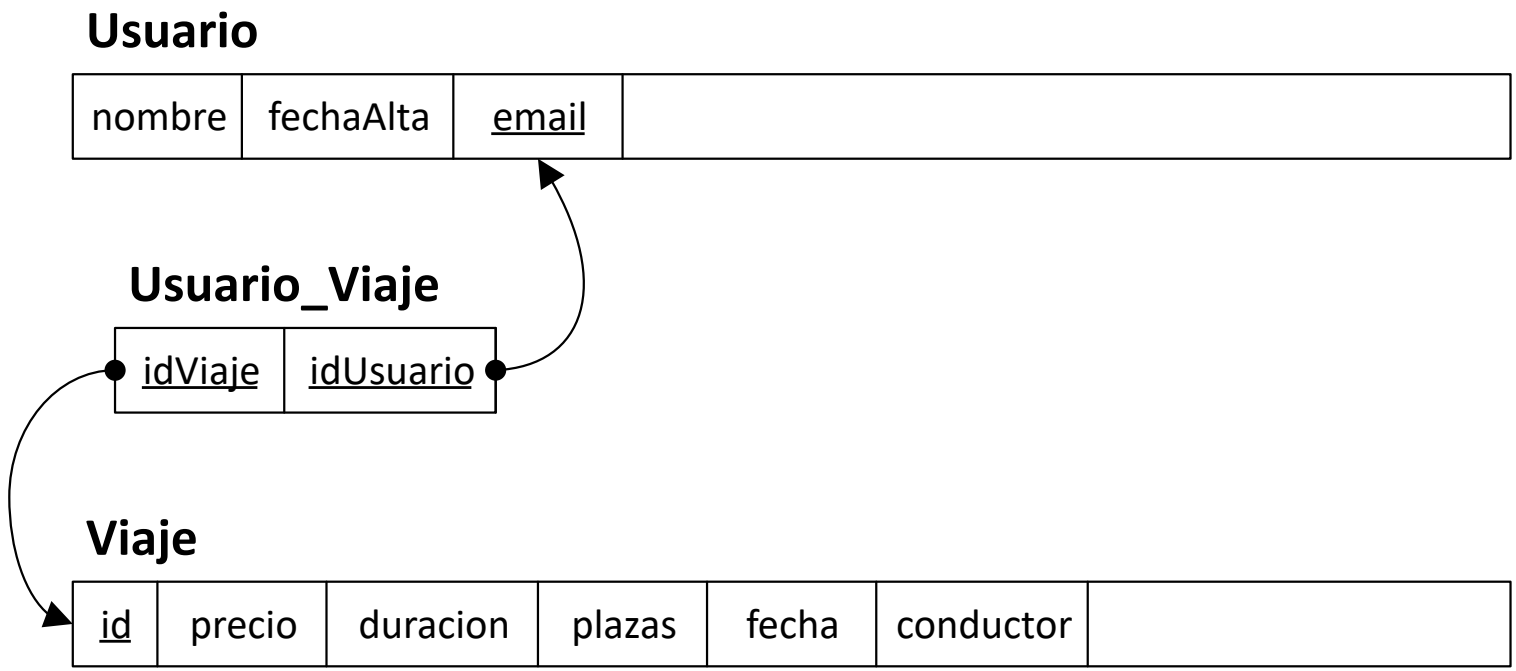
Impedancia OR: Asociaciones Muchos a Muchos



Impedancia OR: Asociaciones Muchos a Muchos

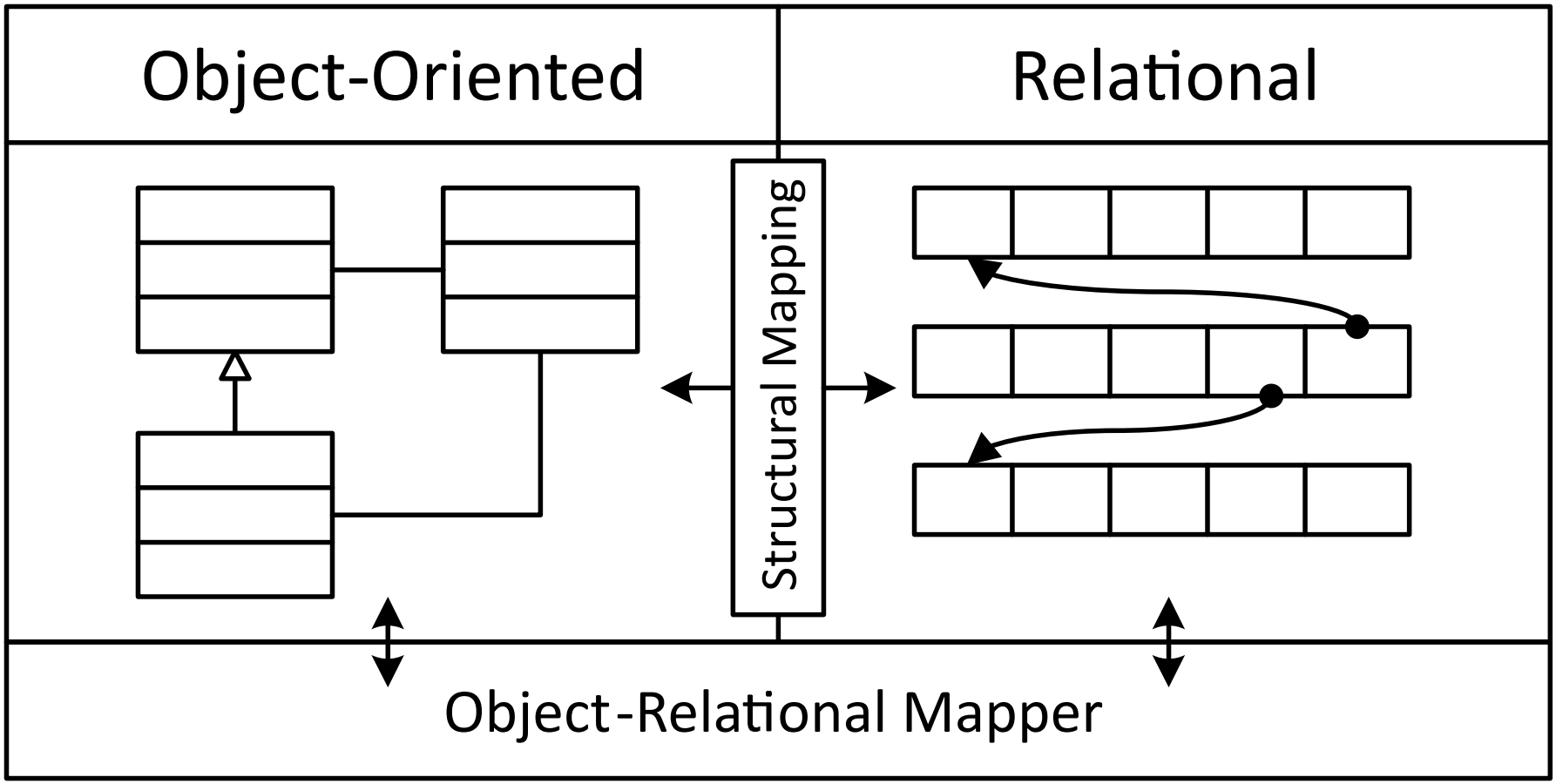


Impedancia OR: Granularidad



Índice

Puentes de Persistencia de Objetos



Índice

Índice

Class to Table

Problema

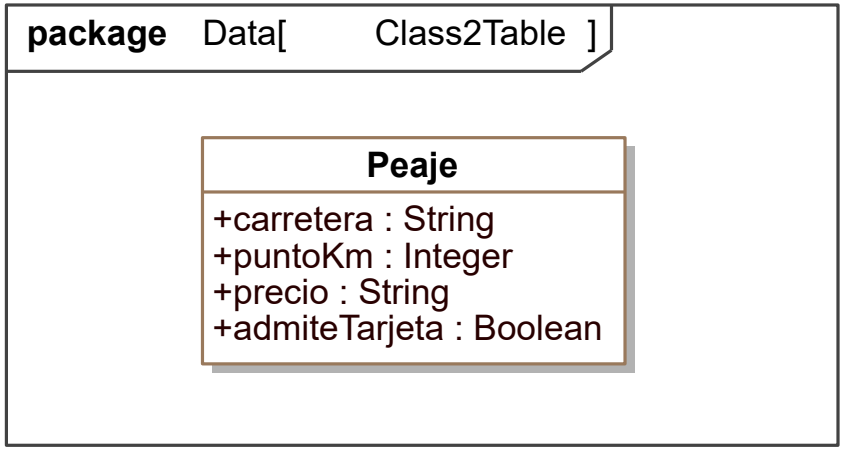
1. ¿Cómo transformo una clase a relacional?

Solución

Si la clase es una *entidad*, no está afectada por herencia, y no tiene atributos multivaluados ni asociaciones con otras clases:

1. Crear una tabla con el mismo nombre de la clase.
2. Crear una columna en dicha tabla por cada atributo de la clase.
3. Asignar como tipo de la columna el tipo que corresponda a cada atributo.

Class to Table



Peaje

carretera	puntoKm	precio	admiteTarjeta

Índice

Identity Field

Problema

1. ¿Cómo consigo que cada objeto de una clase tenga asociada una clave primaria que pueda utilizar para almacenarlo en una tabla de una base de datos relacional?
2. ¿Cómo consigo mantener la correspondencia entre cada objeto de una clase y su correspondiente representación relacional cuando los objetos no tienen clave natural?

Solución

1. Incorporar un nuevo atributo, representando una clave artificial, para aquellos objetos que no poseen una clave natural, o cuya clave natural se considere inadecuada para el modelo relacional.
2. Para las claves artificiales, elegir una estrategia de generación.

Generación de Claves Artificales

1. Columna autoincrementada
   - Clave no disponible hasta finalizar la transacción.
   - Impide *escrituras en batch*
2. Sequence
   - No disponible en todos los gestores.
3. GUID.
   - Demasiado grandes y complejos.
4. Generada por la aplicación (ORM).

Índice

Foreign Key Mapping

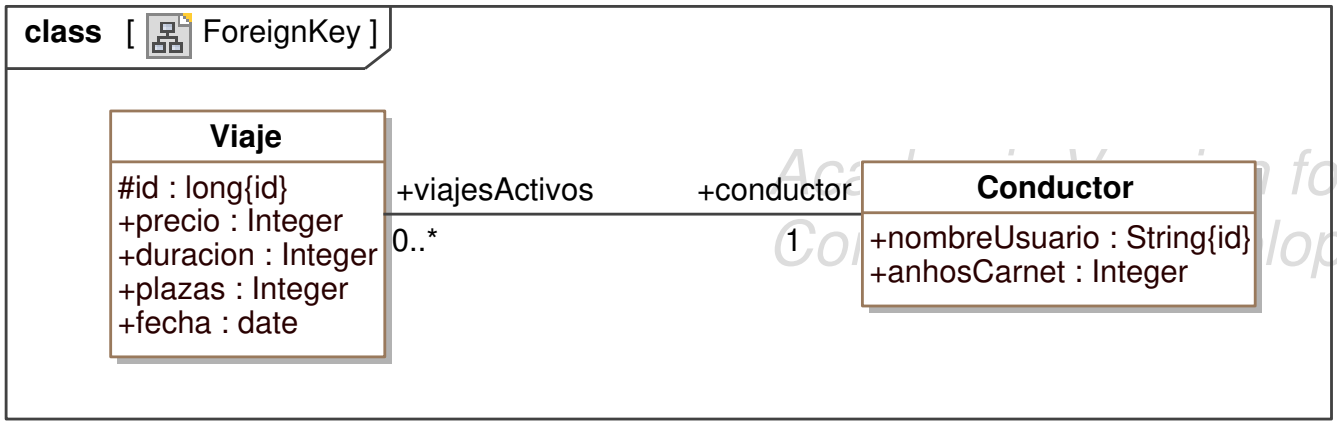
Problema

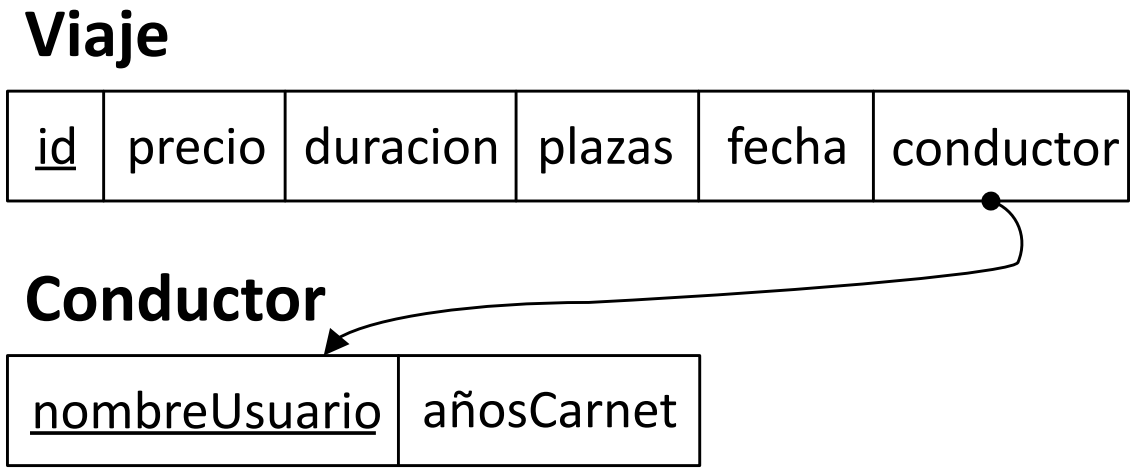
¿Cómo transformo una asociación entre dos clases A y B al modelo relacional?

Solución

Si una clase A posee un extremo de asociación referenciando una clase B con multiplicidad máxima uno, puedo añadir en la tabla correspondiente a la clase A una clave externa a la tabla de la clase B.

Foreign Key Mapping





Índice

Association Table Mapping

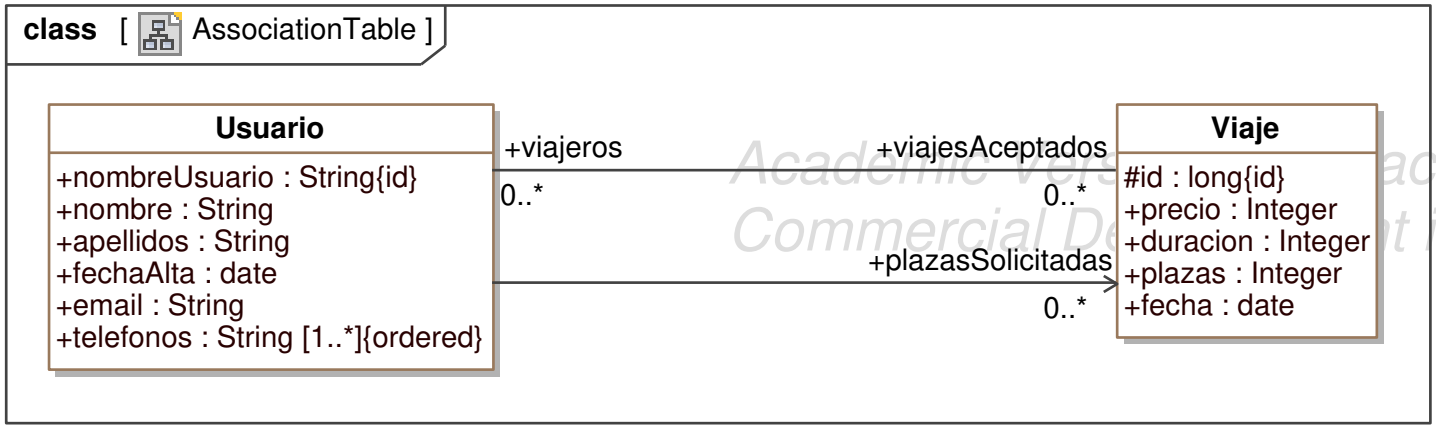
Problema

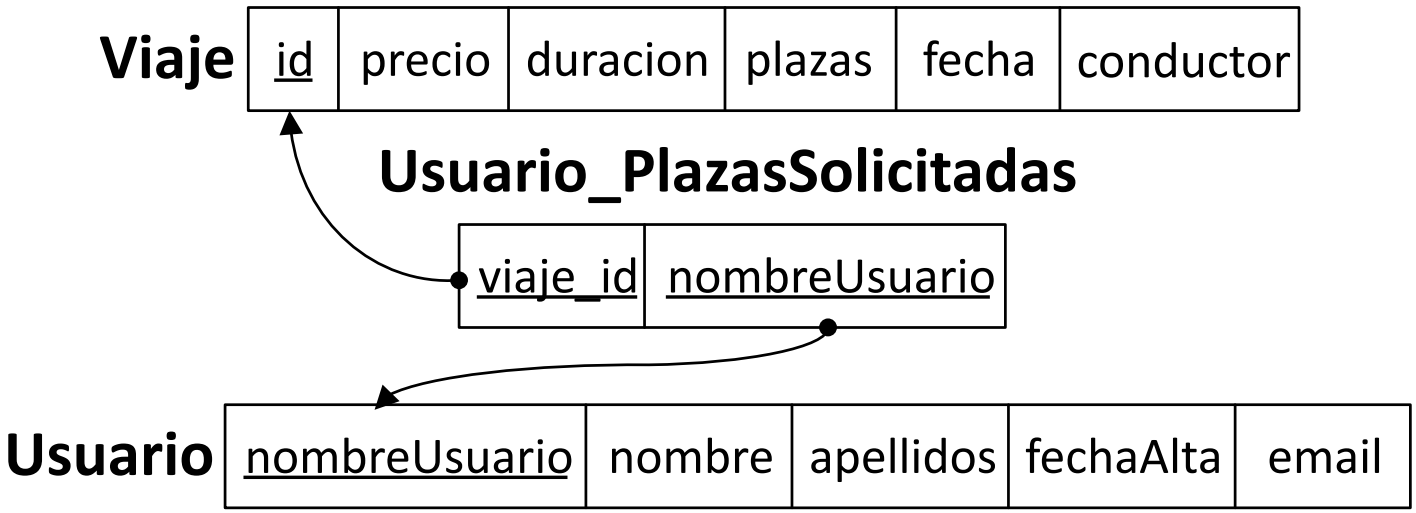
¿Cómo transformo una asociación entre clases al modelo relacional?

Solución

1. Si ambos extremos de asociación tiene multiplicidad superior a uno, crear una tabla intermedia A_B que almacene la relación entre ambas clases.
2. La tabla intermedia almacenará como datos las claves primarias asociadas a las clases A y B.
3. Cada una de estas claves primarias almacenadas será una clave externa a su correspondiente tabla.
4. La clave primaria de la tabla intermedia será la unión de las claves primarias de las tablas relacionadas.

Association Table Mapping





Índice

Embedded Value

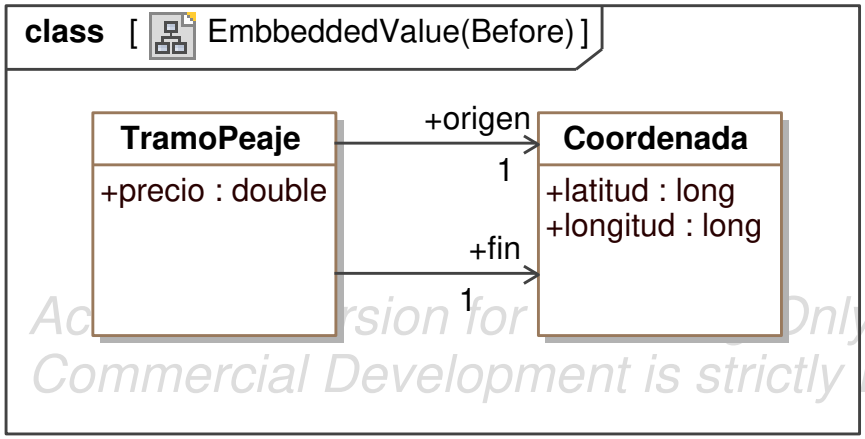
Problema

¿Cómo mapeo asociaciones con *value objects* de manera eficiente?

Solución

Si una clase C tiene una asociación de multiplicidad máxima 1 con un *value object* V, puedo simplemente añadir los campos de V a la clase C y luego mapear C como una clase simple.

Embedded Value



TramoPeaje

precio	origen_latitud	origen_longitud	fin_laetud	fin_longitud

Índice

Serialised LOB

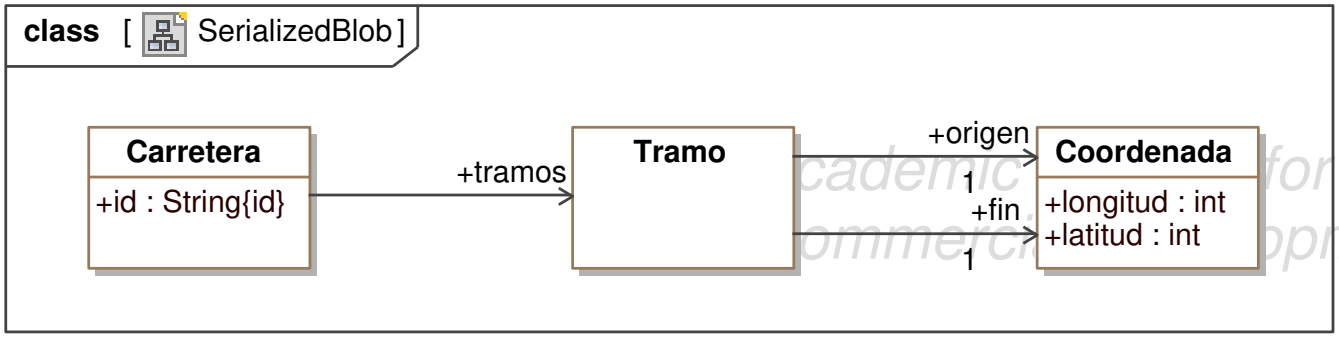
Problema

¿Cómo mapeo asociaciones con *value objects* y/o *entities* internas a un *aggregate* de manera eficiente?

Solución

Guardar todo un grafo de objetos en una única columna de tipo *LOB*.

Serialised LOB



Carretera

id	tramos

Índice

Single Table Inheritance

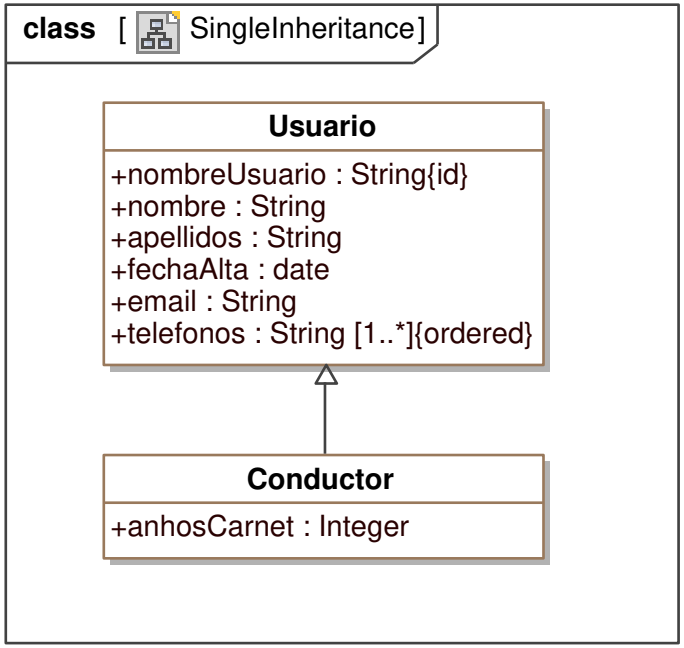
Problema

1. ¿Cómo transformo una jerarquía de herencia en un esquema relacional?

Solución

1. Comprimir la jerarquía en una sola clase, incorporando los atributos de las clases hijas a la raíz de la jerarquía.
2. Añadir un atributo que indique de qué tipo concreto es cada instancia de la nueva clase resultante.
3. Transforma la clase resultante a una tabla.
4. Cuando un objeto de dicha jerarquía se almacena dentro de la tabla resultante, los atributos que no correspondan a dicha instance simplemente se ignoran.

Single Table Inheritance



Usuario

nombreUsuario	nombre	apellidos	fechaAlta	email	anhosCarnet	tipoUsuario

Índice

Concrete Table Inheritance

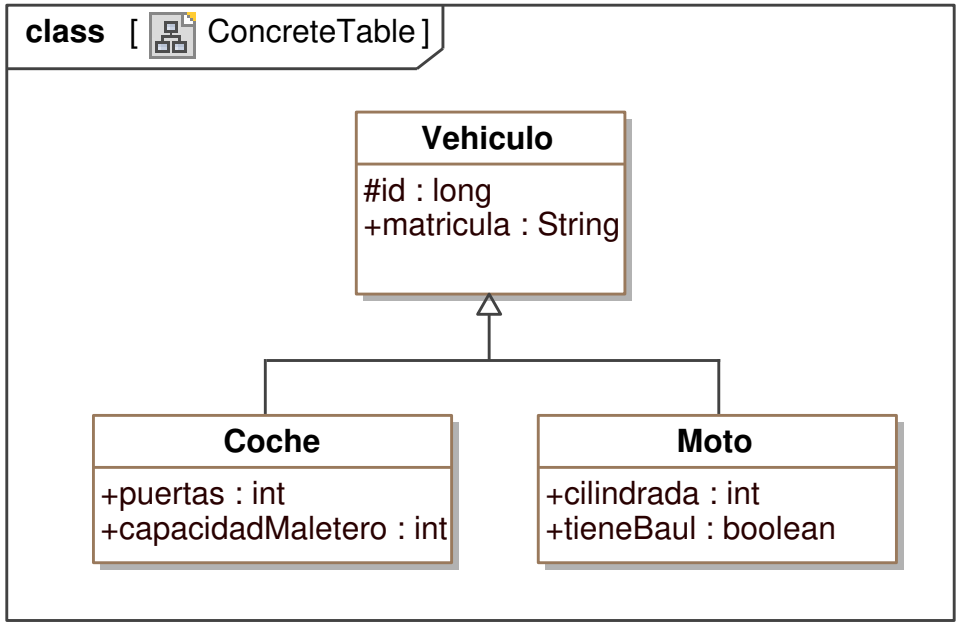
Problema

1. ¿Cómo transformo una jerarquía de herencia en un esquema relacional?

Solución

1. Hacer descender los atributos de la clases abstractas a las clases concretas.
2. Transformar cada clase resultante a una tabla.

Concrete Table Inheritance



Coche

id	matricula	puertas	capacidadMaletero

Moto

id	matricula	cilindrada	tieneBaul

Índice

Class Table Inheritance

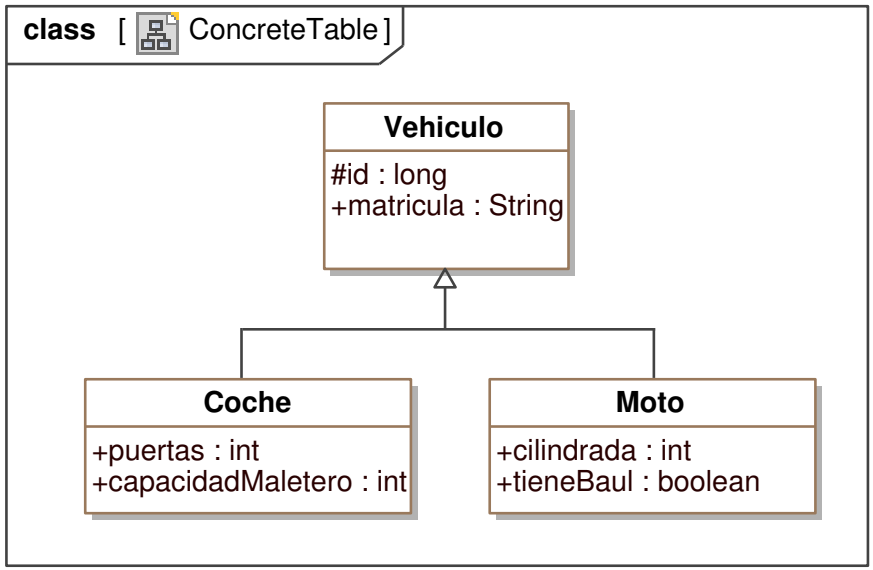
Problema

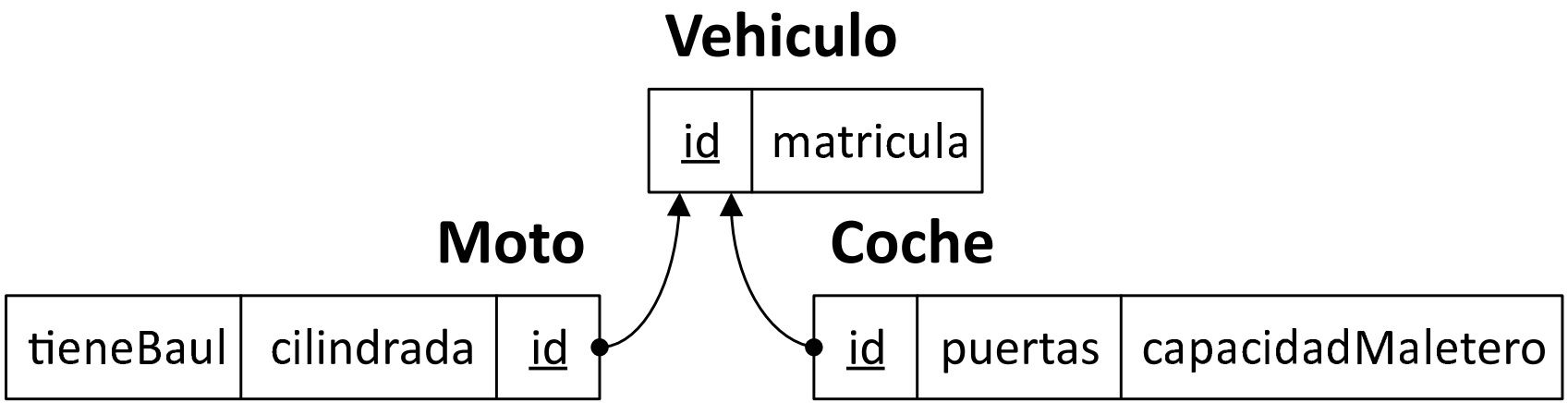
1. ¿Cómo transformo una jerarquía de herencia en un esquema relacional?

Solución

1. Transformar cada clase en la jerarquía a una tabla.
2. Mantener las relaciones de herencia mediante relaciones de clave externa.

Class Table Inheritance





Índice

Estrategias de Transformación de Herencias

	Single	Concrete	Class
Gestión Valores Nulos	✗	✓	✓
Consultas Polimórficas	✓	✗	✗
Consultas Subclases	✗	✓	✓
Uso de *Joins*	✓	✓	✗
Contención BBDD	✗	✓	✓
Evolución Superclase	✓	✗	✓
Clave Primaria Dispersa	✗	✓	✗

¿Cuándo Aplicar cada Estrategia?

1. Single Table Inheritance
   - Jerarquías de herencia donde las subclases básicamente redefinen comportamientos y añaden pocos datos nuevos.
   - La mayoría de las asociaciones entre clases y búsquedas se realizan a nivel de superclase.
2. Concrete Table Inheritance
   - Jerarquías de herencia donde las superclases son muy abstractas, no tienen apenas correspondencia a nivel de dominio y básicamente abstraen datos y comportamientos comunes.
   - La mayoría de las asociaciones entre clases y búsquedas se realizan a nivel de subclase.
3. Class Table Inheritance
   - No se dan ninguna de las condiciones anteriores con claridad.

Índice

Índice

Data Mappers/Data Access Objects

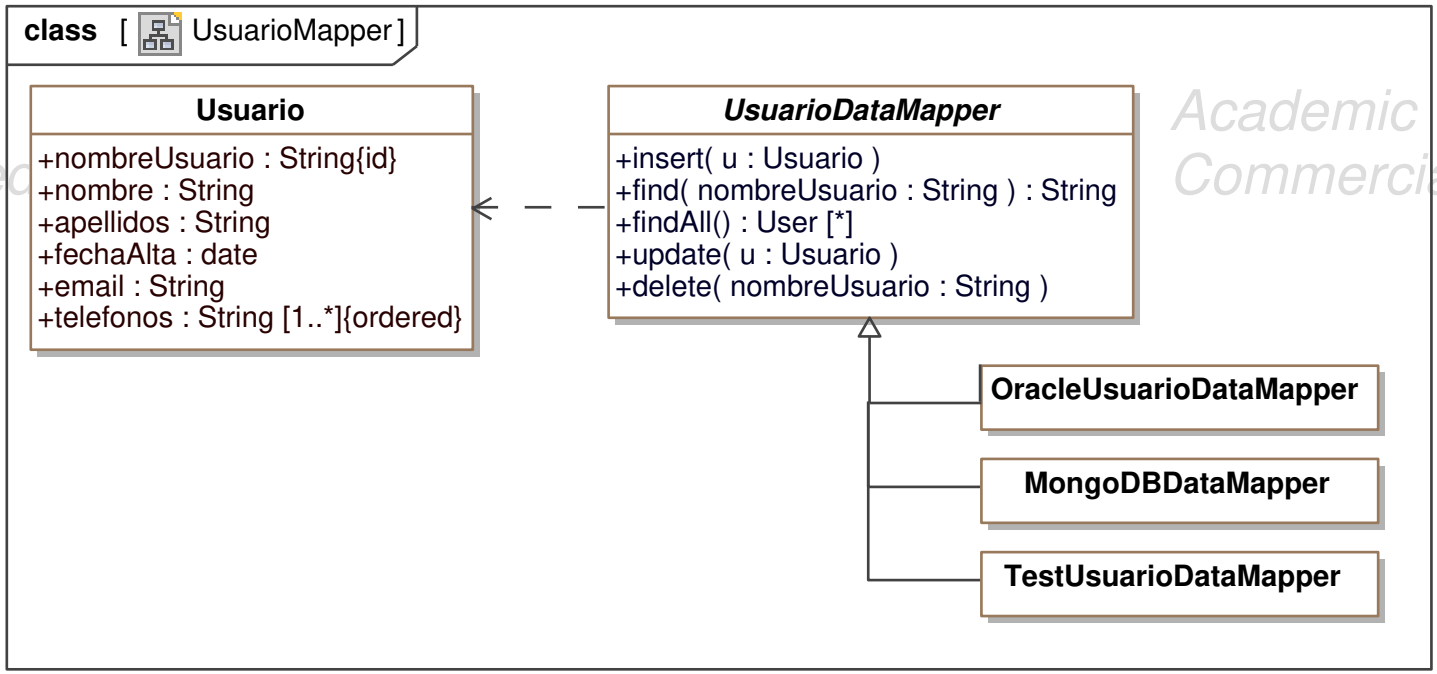
Problema

¿Cómo almacenar, recuperar, actualizar y eliminar objetos del almacén persistente manteniendo al modelo de dominio independiente de su forma de almacenamiento?

Solución

Por cada clase C del modelo de dominio, crear una clase C_Mapper que se encargue de gestionar la correspondiente transformación.

Data Mappers/Data Access Objects



Índice

Metadata Mapping

Problema

¿Cómo saber cómo se ha realizado la transformación objeto-relacional para poder así implementar *Data Mappers* genéricos?

Solución

Especificar mediante algún tipo de mecanismo adecuado la correspondencia entre elementos del modelo de dominio y el esquema relacional asociado.

Índice

Identity Map

Problema

¿Cómo puedo evitar cargar múltiples copias de un mismo objeto?

Solución

1. Crear un sitio donde almacenar los objetos que se carguen desde el almacén de persistencia.
2. Antes de cargar cualquier objeto, comprobar si está ya cargado en el almacén persistente.

Índice

Lazy Load

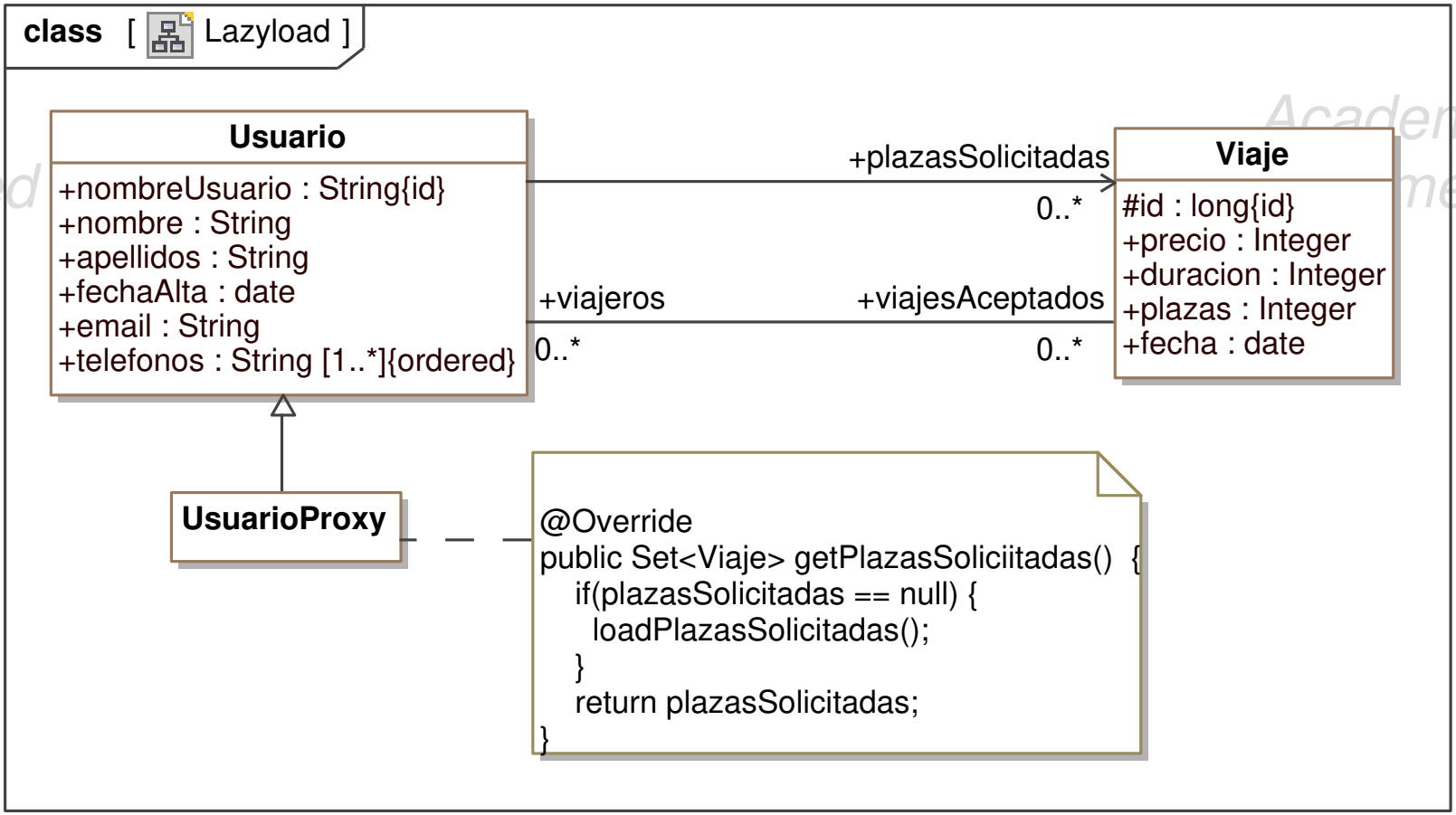
Problema

¿Cómo evito tener que cargar todos los objetos asociados a un objeto O cuando cargo O desde el almacén persistente?

Solución

Aplicar el patrón *proxy* de manera que los objetos referenciados por el objeto O sólo se carguen bajo demanda.

Lazy Load


class [Lazyload]
Usuario
+nombreUsuario : String{id}
+nombre : String
+apellidos : String
+fechaAlta : date
+email : String
+telefonos : String [1..*]{ordered}
+plazasSolicitadas
0..*
Viaje
#id : long{id}
+precio : Integer
+duracion : Integer
+plazas : Integer
+fecha : date
+viajeros
0..*
+viajesAceptados
0..*
UsuarioProxy
@Override
public Set<Viaje> getPlazasSoliciitadas() {
if(plazasSolicitadas == null) {
loadPlazasSolicitadas();
}
return plazasSolicitadas;
}

Índice

Query Object

Problema

¿Cómo realizar búsquedas de objetos arbitrariamente complejas sobre colecciones de datos largas aprovechando las facilidades proporcionadas por el almacén persistente?

Solución

Representar las consultas como objetos independientes del almacén persistente y pasar estos objetos a los repositorios de acceso a datos.

Índice

¿Qué tengo que saber de todo ésto?

1. Entender el problema del desacoplamiento objeto-(relacional).
2. Comprender el objetivo y estructura de un puente-objeto relacional.
3. Ser capaz de hacer una transformación de un modelo de dominio en un esquema relacional.
4. Comprender cómo funcionan los patrones que permiten independizar un modelo de dominio de su forma de persistencia.