Patrones de Aplicaciones de Arquitectura Enterprise

FI.UBA

75.10 - Técnicas de Diseño

Aplicaciones Enterprise

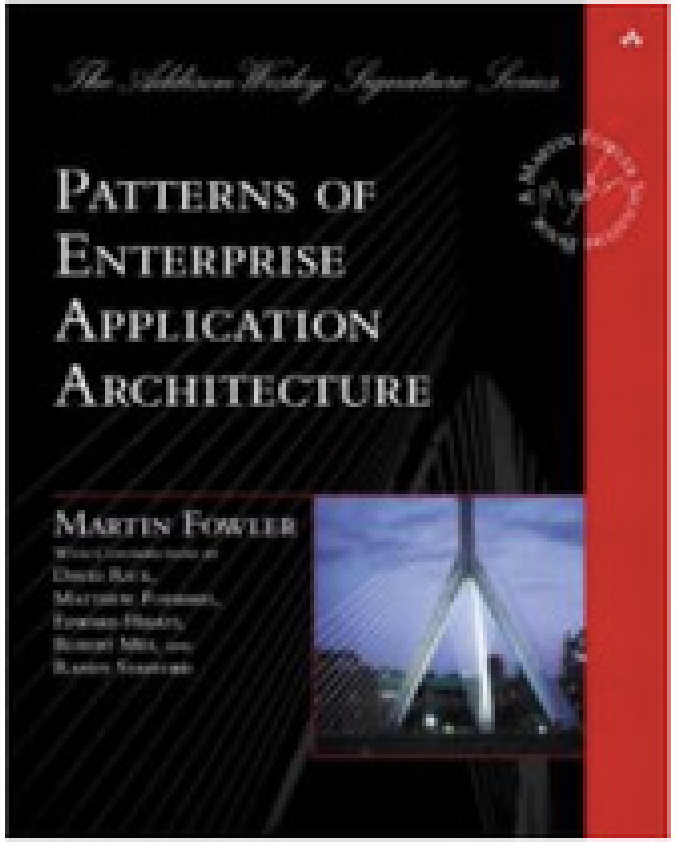
Una aplicación enterprise es un sistema fuertemente orientado a un negocio determinado, que debe cumplir ciertos requerimientos no funcionales. En general, las aplicaciones enterprise:

- **persisten datos de forma masiva**
- **suelen ser multiusuarios (varios usuarios pueden acceder de forma concurrente)**
- **cuentan con muchísimas interfaces de usuario**
- **se integran con otros sistemas**
- **presentan disonancia conceptual (usuarios con vistas contrapuestas)**
- **tienen lógica de negocio**

Decisiones a tomar al definir la arquitectura de este tipo de aplicaciones

- Como codificar las reglas de negocio
- Como representar las entidades de negocio
- Como persistir su estado
- Como garantizar la coherencia de datos
- Como manejar la distribución de la aplicación

Catálogo de Patrones



Patterns of Enterprise Application Architecture (2002)

Martin Fowler
www.martinfowler.com

- **Domain Logic Patterns:** Transaction Script (110), Domain Model (116), Table Module(125), Service Layer (133).

- **Data Source Architectural Patterns:** Table Data Gateway (144), Row Data Gateway(152), Active Record (160), Data Mapper (165).

- **Object-Relational Behavioral Patterns:** Unit of Work (184), Identity Map (195), Lazy Load (200)

- **Object-Relational Structural Patterns:** Identity Field (216), Foreign Key Mapping(236), Association Table Mapping (248), Dependent Mapping (262), Embedded Value(268), Serialized LOB (272), Single Table Inheritance (278), Class Table Inheritance(285), Concrete Table Inheritance (293), Inheritance Mappers (302).
- **Object-Relational Metadata Mapping Patterns:** Metadata Mapping (306), Query Object (316), Repository (322).
- **Web Presentation Patterns:** Model View Controller (330), Page Controller (333),Front Controller (344), Template View (350), Transform View (361), Two-Step View(365), Application Controller (379).
- **Distribution Patterns:** Remote Facade (388), Data Transfer Object (401)
- **Offline Concurrency Patterns:** Optimistic Offline Lock (416), Pessimistic Offline Lock(426), Coarse Grained Lock (438), Implicit Lock (449).
- **Session State Patterns:** Client Session State (456), Server Session State (458),Database Session State (462).
- **Base Patterns:** Gateway (466), Mapper (473), Layer Supertype (475), Separated Interface (476), Registry (480), Value Object (486), Money (488), Special Case (496),Plugin (499), Service Stub (504), Record Set (508)

Domain Logic Patterns

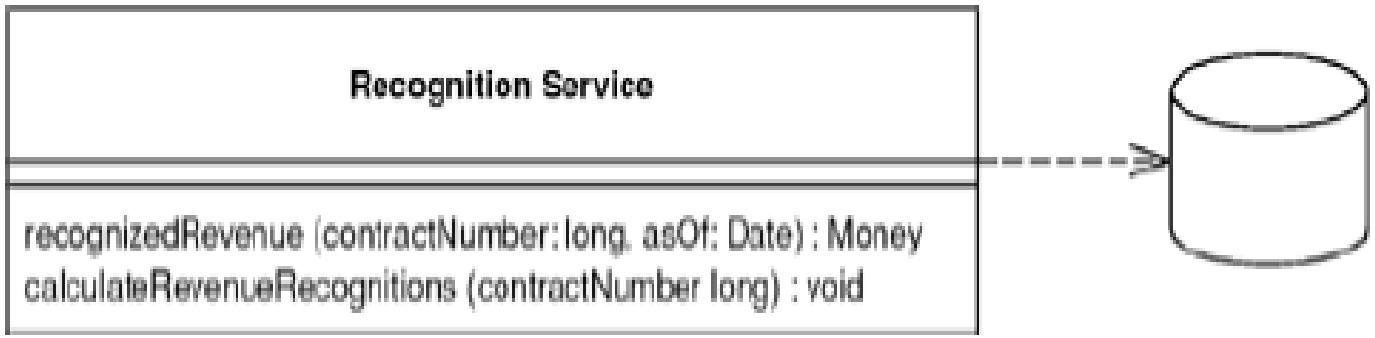
Transaction Script

La organización fundamental es un solo procedimiento para cada acción que el usuario requiere, toda la lógica requerida esta en el procedimiento.

Es un modelo simple de entender pero se puede caer en repetición de código y de lógica

Transaction Script

Organizes business logic by procedures where each procedure handles a single request from the presentati



Domain Model

La organización esta dado por modelar el dominio , donde toda la lógica de negocio , cálculos esten en el mismo. La lógica no esta toda en un solo lugar sino que se distribuye en responsabilidades a cada objeto del dominio.

Table Module

La organización esta dada por clases también como el Domain Model, pero hay solo una para manejar todas las instancias.

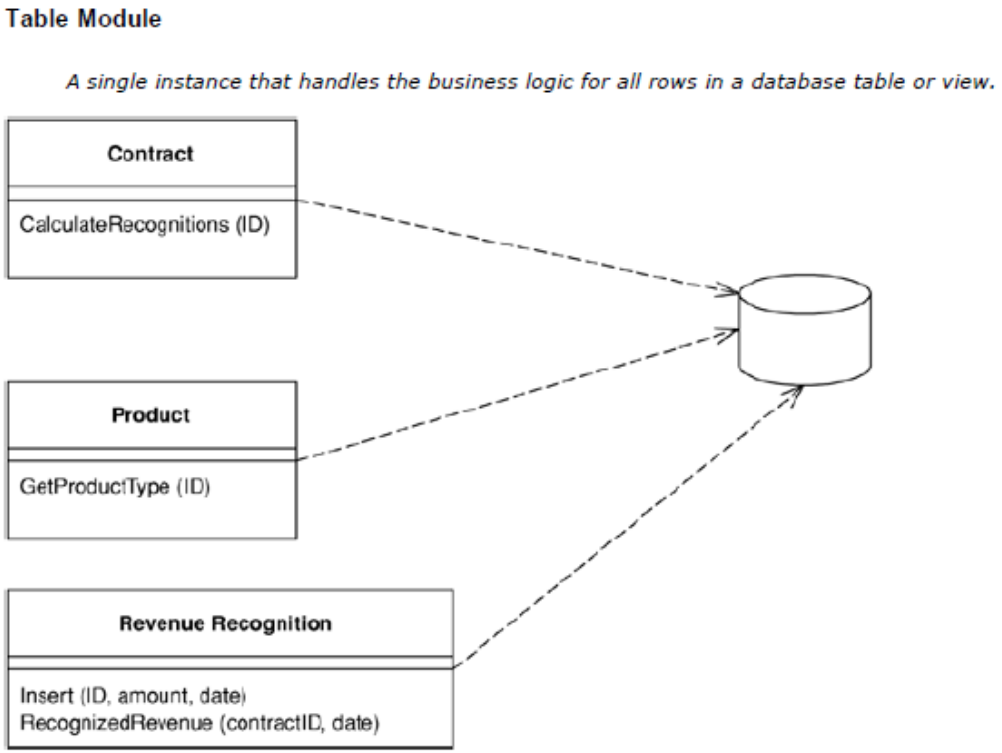
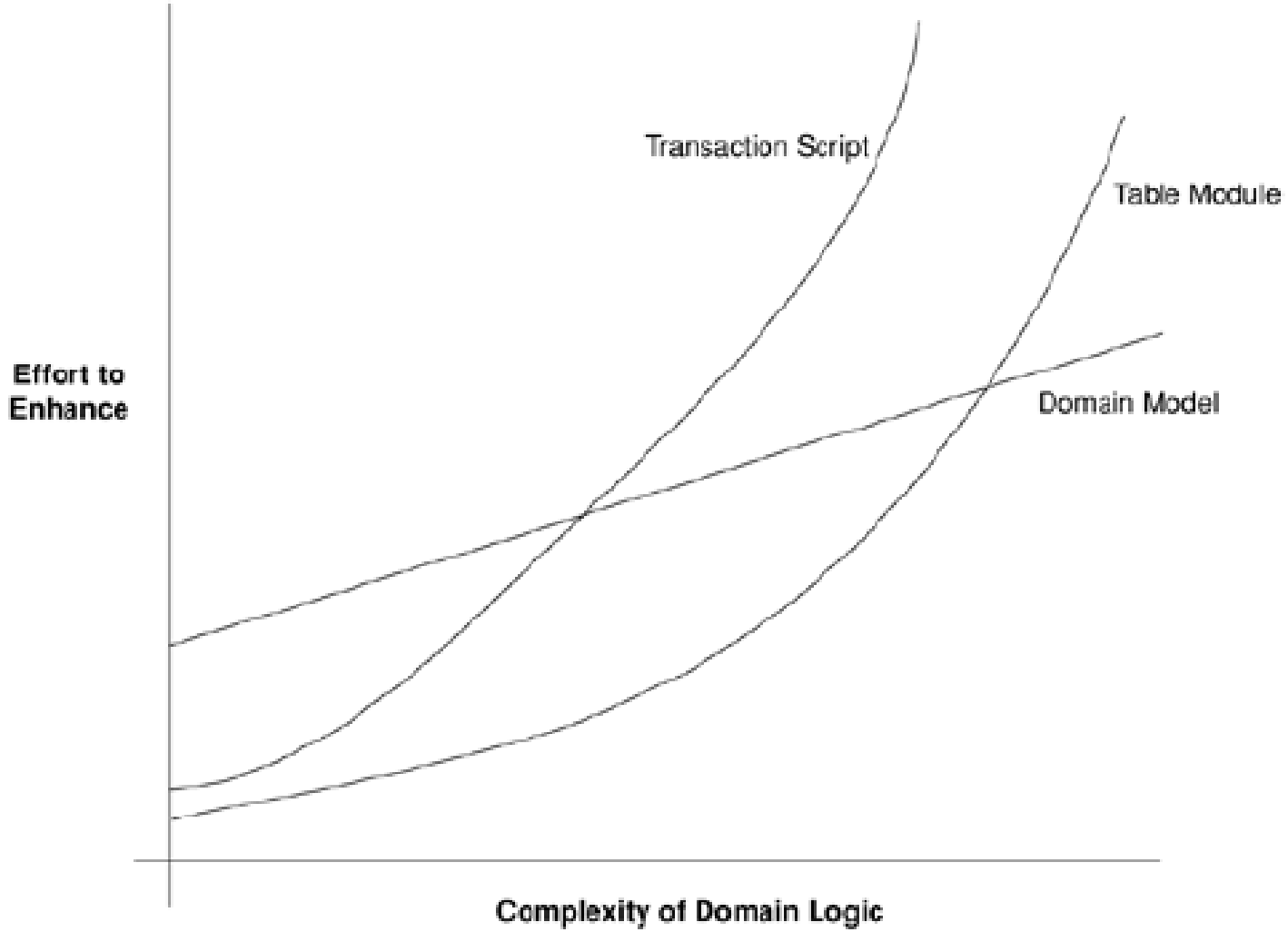


Figure 2.4. A sense of the relationships between complexity and effort for different domain logic styles.



Service Layer

Actúa como una API para la aplicación. Brinda un único punto de acceso o fachada al dominio.

Ademas es un buen lugar para usar cosas como transacciones, seguridad, logs, etc.

Data Source Architectural Patterns

Row Data Gateway

Propone tener una instancia del gateway por cada fila que retorna una consulta

Figure 3.1. *A* **Row Data Gateway (152)** *has one instance per row returned by a query.*

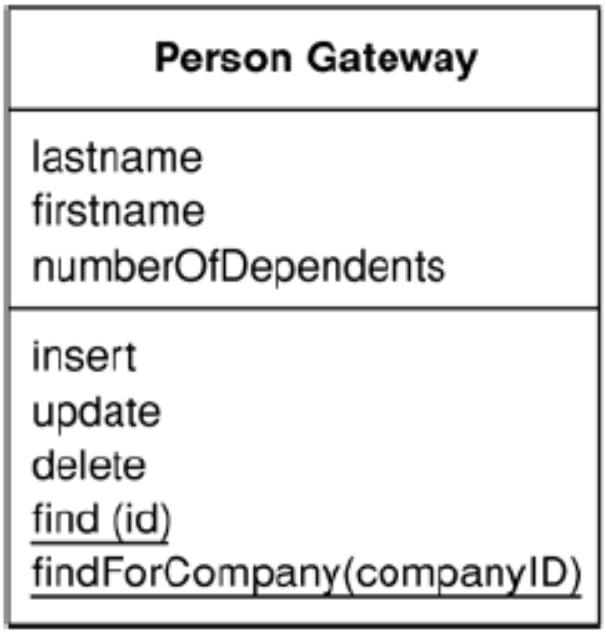
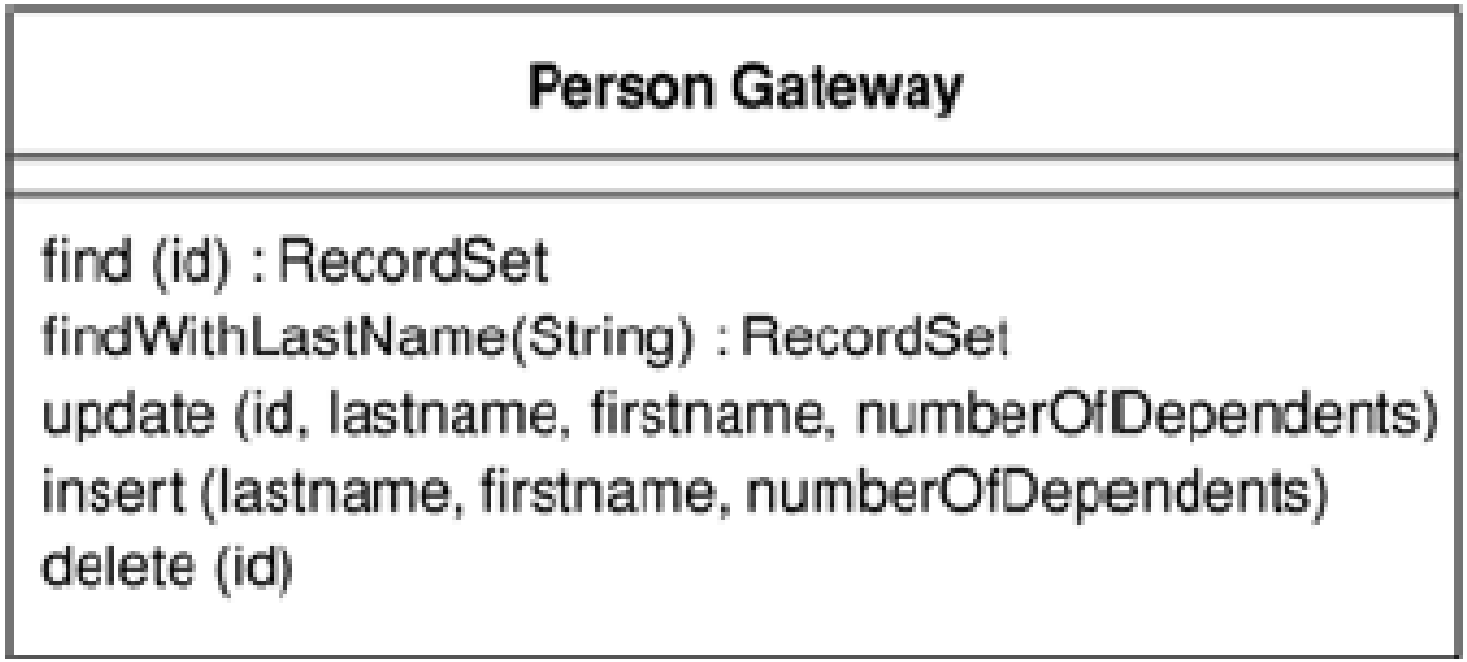


Table Data Gateway

Provee metodos de consulta a una base de dartos y retornan un recordset

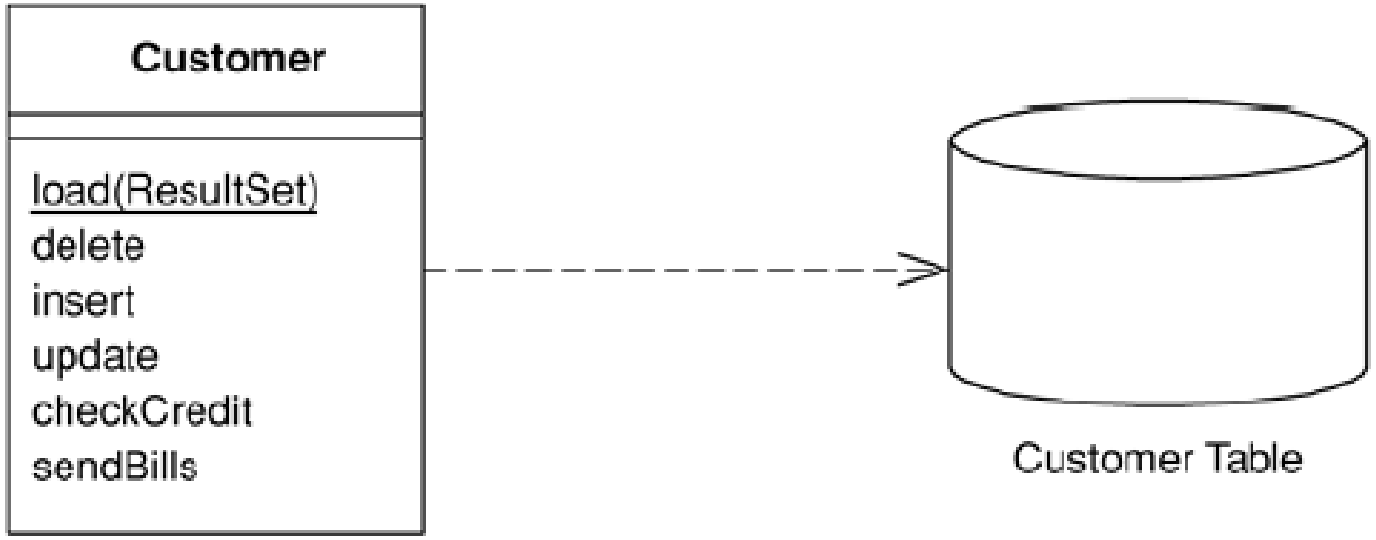
Figure 3.2. *A* **Table Data Gateway (144)** *has one instance per table.*



Active Record

En aplicaciones de Domain Model simples con una clase de dominio por tabla, Active Record propone que cada objeto sea el responsable de cargarse y guardarse en la base de datos

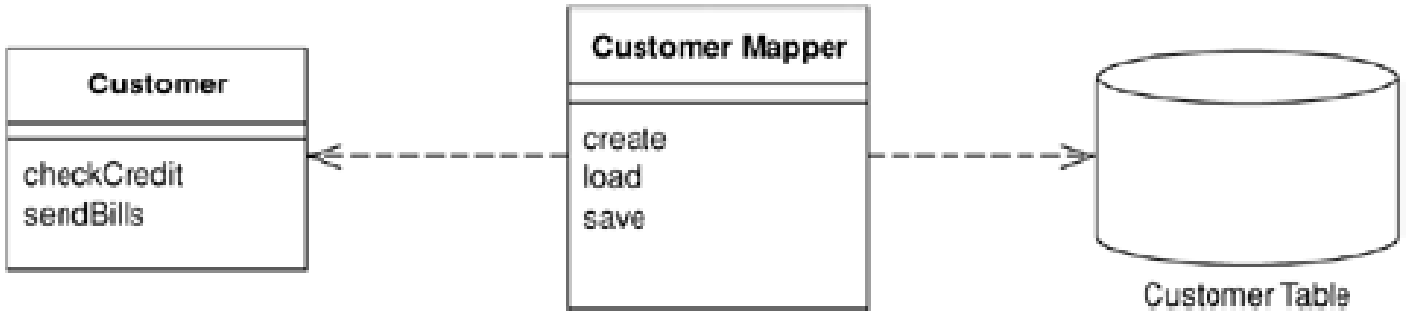
Figure 3.3. ***In the Active Record (160) a customer domain object knows how to interact with database tables.***



Data Mapper

Propone separar el Domain Model de la base de datos, poniendo una indirección responsable del mapeo entre objetos de dominio y tablas de la base de datos

Figure 3.4. *A* **Data Mapper (165)** *insulates the domain objects and the database from each other.*



Object-Relational Behavioral Patterns

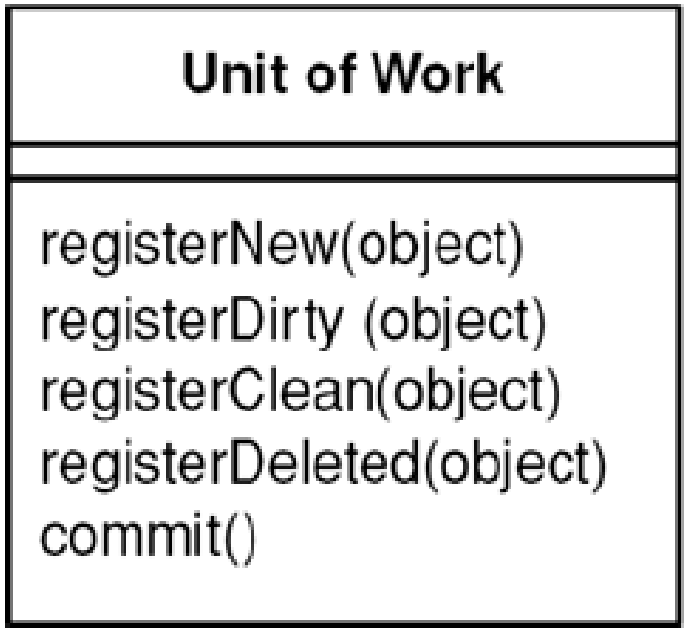
Unit of Work

Mantiene el seguimiento de todo lo que se hace durante la transacción de negocio que afecta a la base de datos.

Al final el realiza todos los cambios contra la base de datos.

Unit of Work

Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.

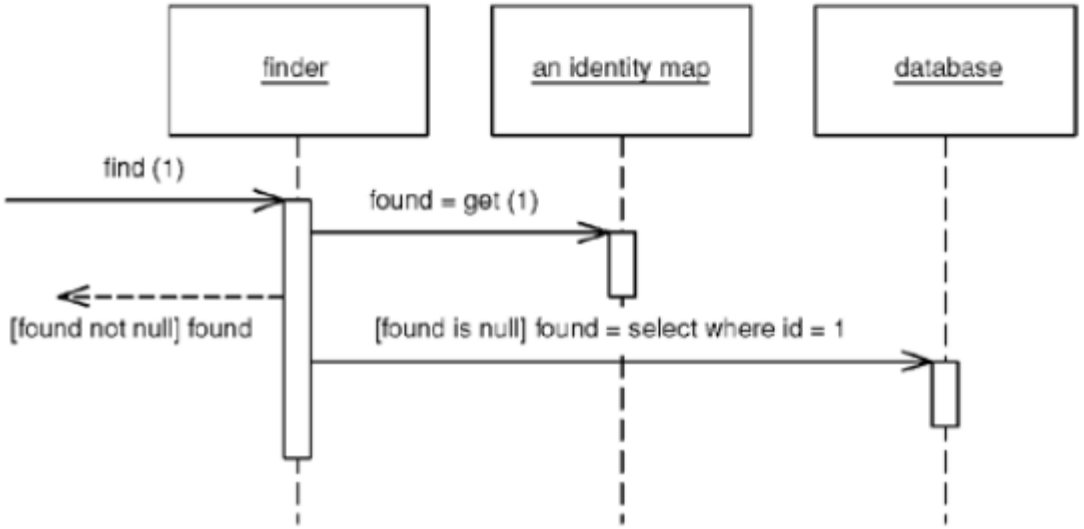


Identity Map

Se asegura de que cada objeto de dominio de carga una única vez en cada transacción de negocio.

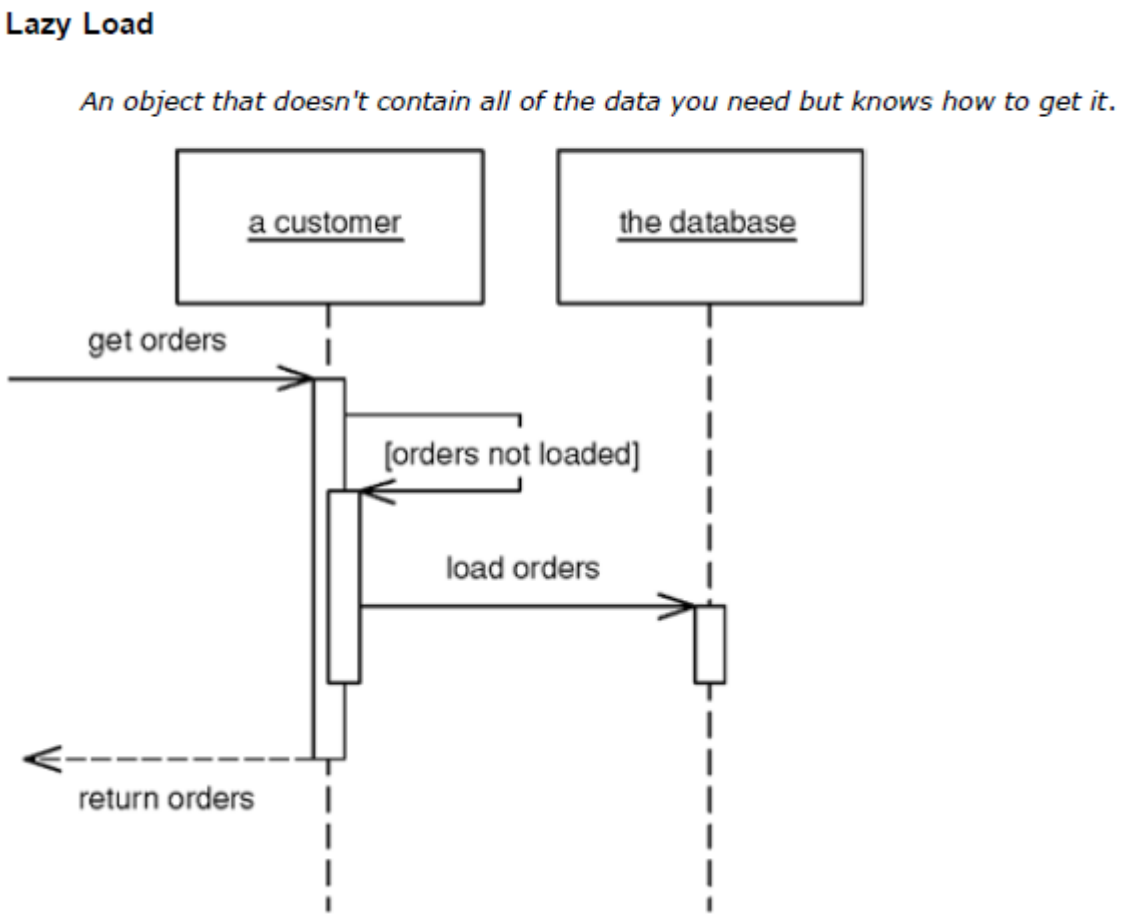
Identity Map

Ensures that each object gets loaded only once by keeping every loaded object in a map. Looks up objects using the map when referring to them.



Lazy Load

Interrumpe la carga de objetos relacionados a un objeto de dominio, permitiendo cargarlo en forma transparente en cuanto el mismo sea requerido.



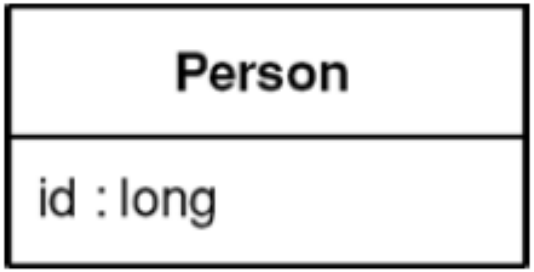
Object-Relational Structural Patterns

Identity Field

Permite identificar un objeto de dominio persistido en una base de datos.

Identity Field

Saves a database ID field in an object to maintain identity between an in-memory object and a database r



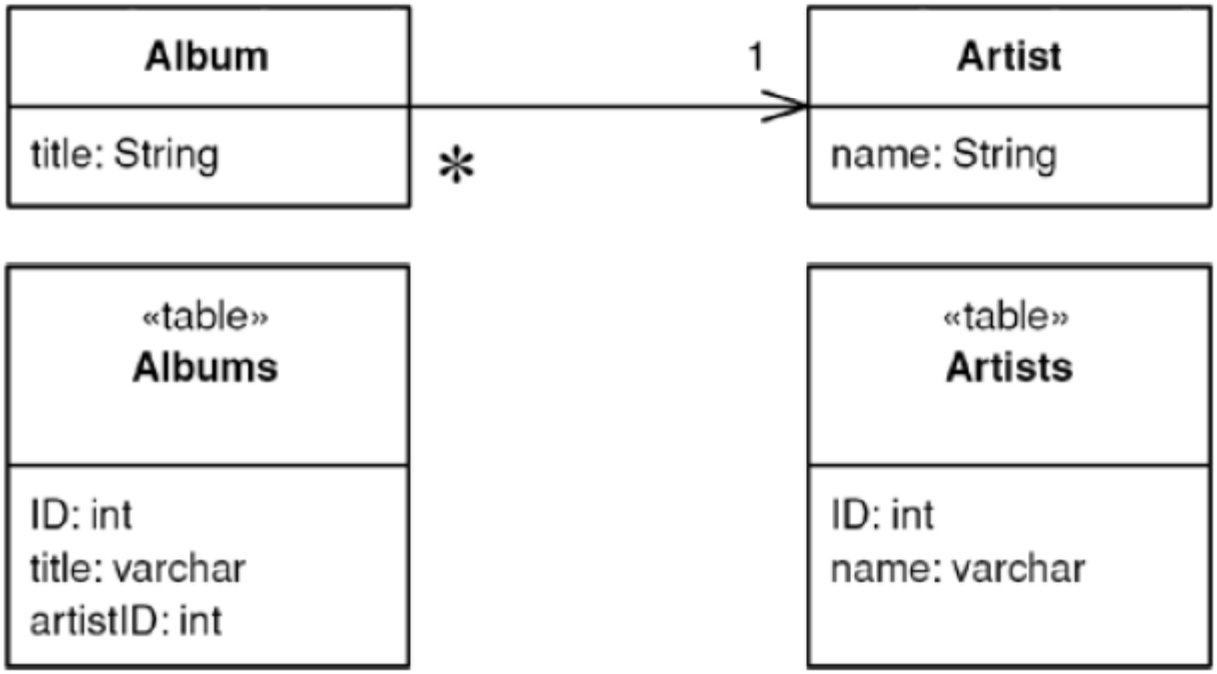
Object-Relational Structural Patterns

Foreign Key Mapping

Permite identificar las relaciones o agregaciones de un objeto de dominio persistido en una base de datos.

Foreign Key Mapping

Maps an association between objects to a foreign key reference between tables.



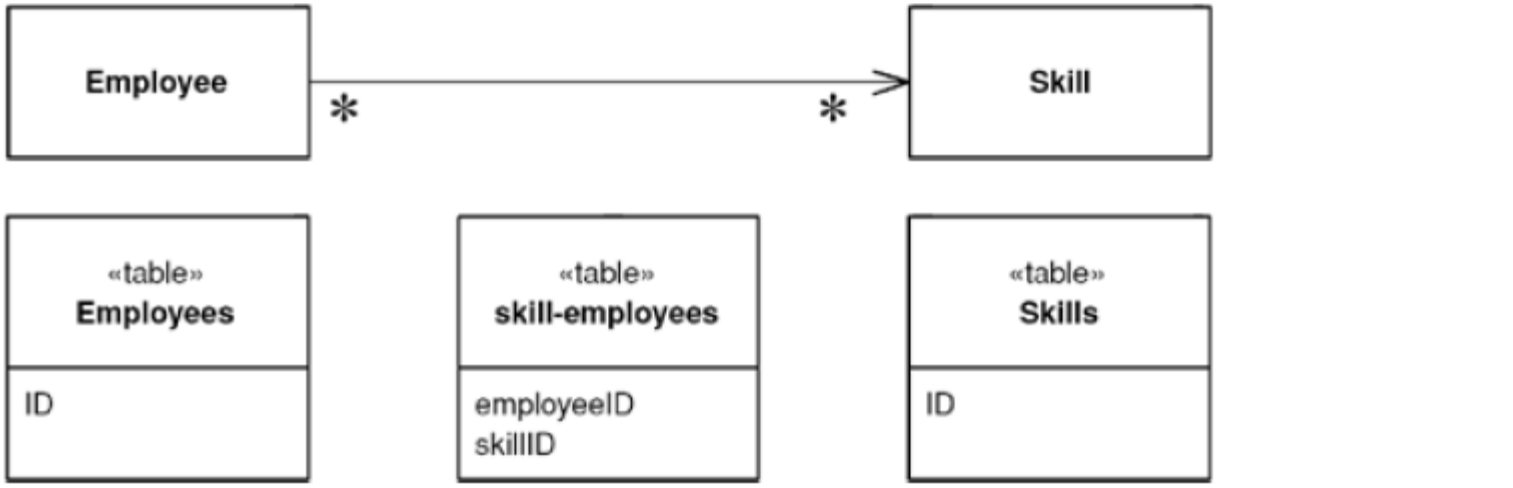
Object-Relational Structural Patterns

Association Table Mapping

Permite identificar las asociaciones muchos a muchos de objeto de dominio persistidos en una base de datos.

Association Table Mapping

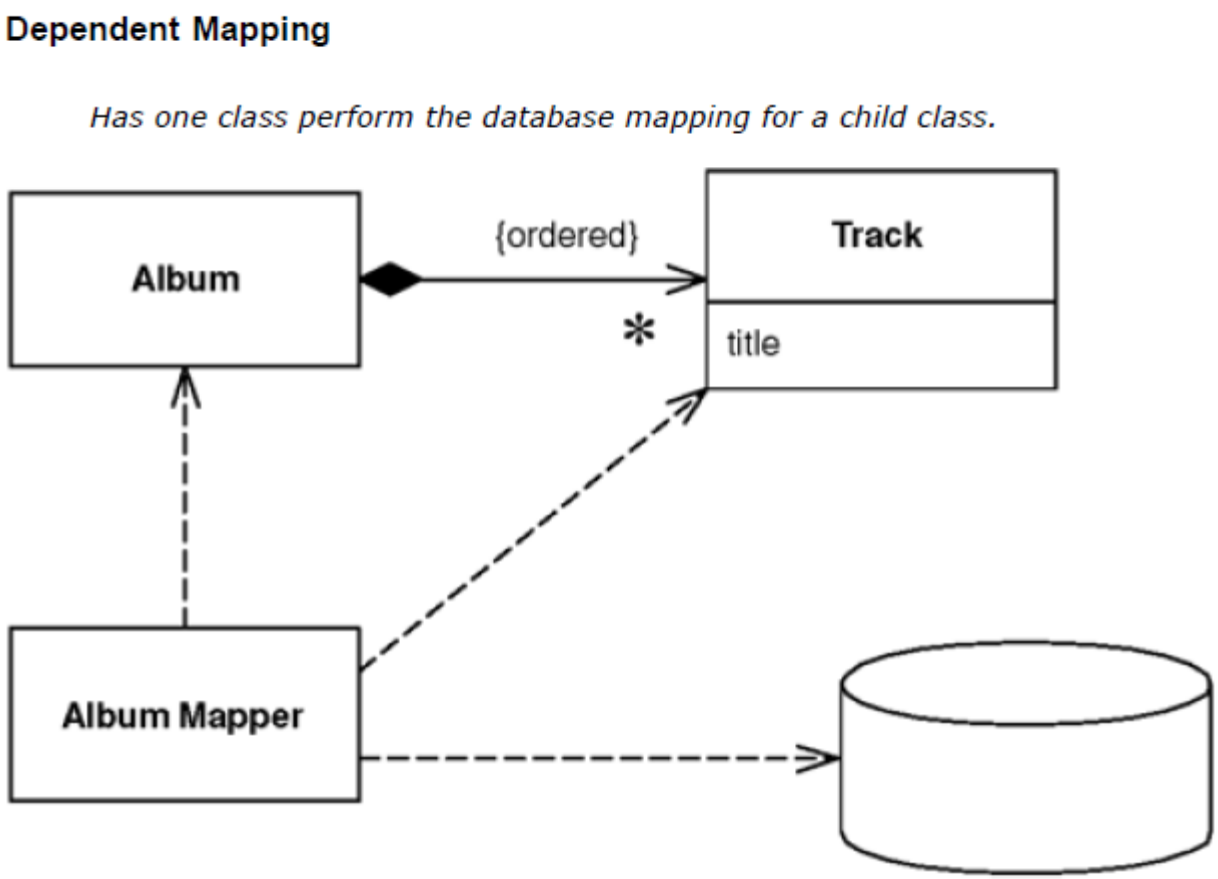
Saves an association as a table with foreign keys to the tables that are linked by the association.



Object-Relational Structural Patterns

Dependent Mapping

La clase dependiente (en la composición) delega la persistencia a otra clase, la owner.



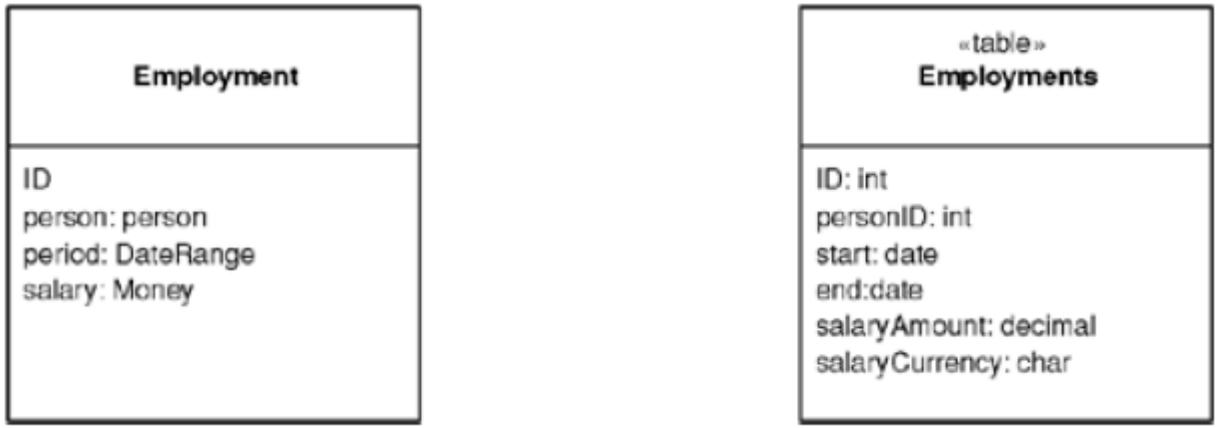
Object-Relational Structural Patterns

Embedded Value

Mapea los valores de un objeto a campos de un registro del objeto owner.

Embedded Value

Maps an object into several fields of another object's table.



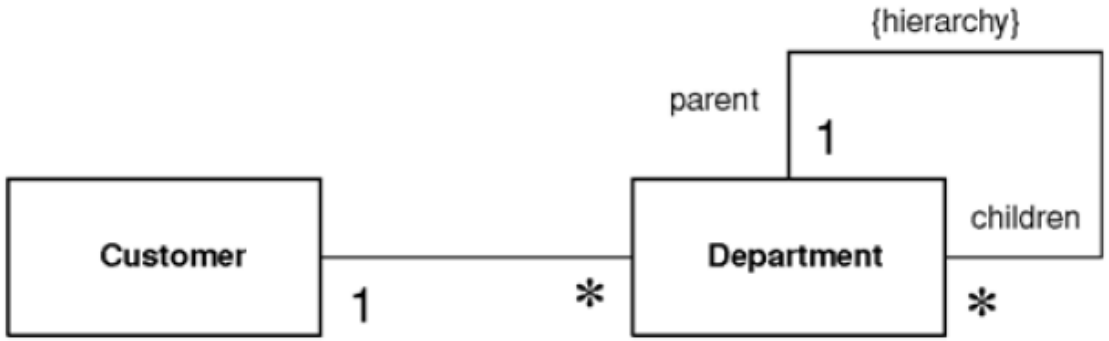
Object-Relational Structural Patterns

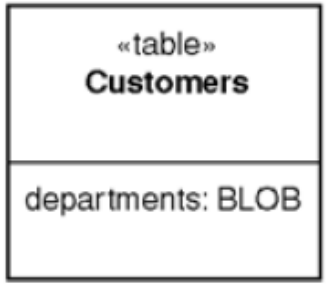
Serialized LOB

Propone no almacenar ciertos objetos en registros de la base de datos, sino serializar en un tipo de dato BLOB su serialización.

Serialized LOB

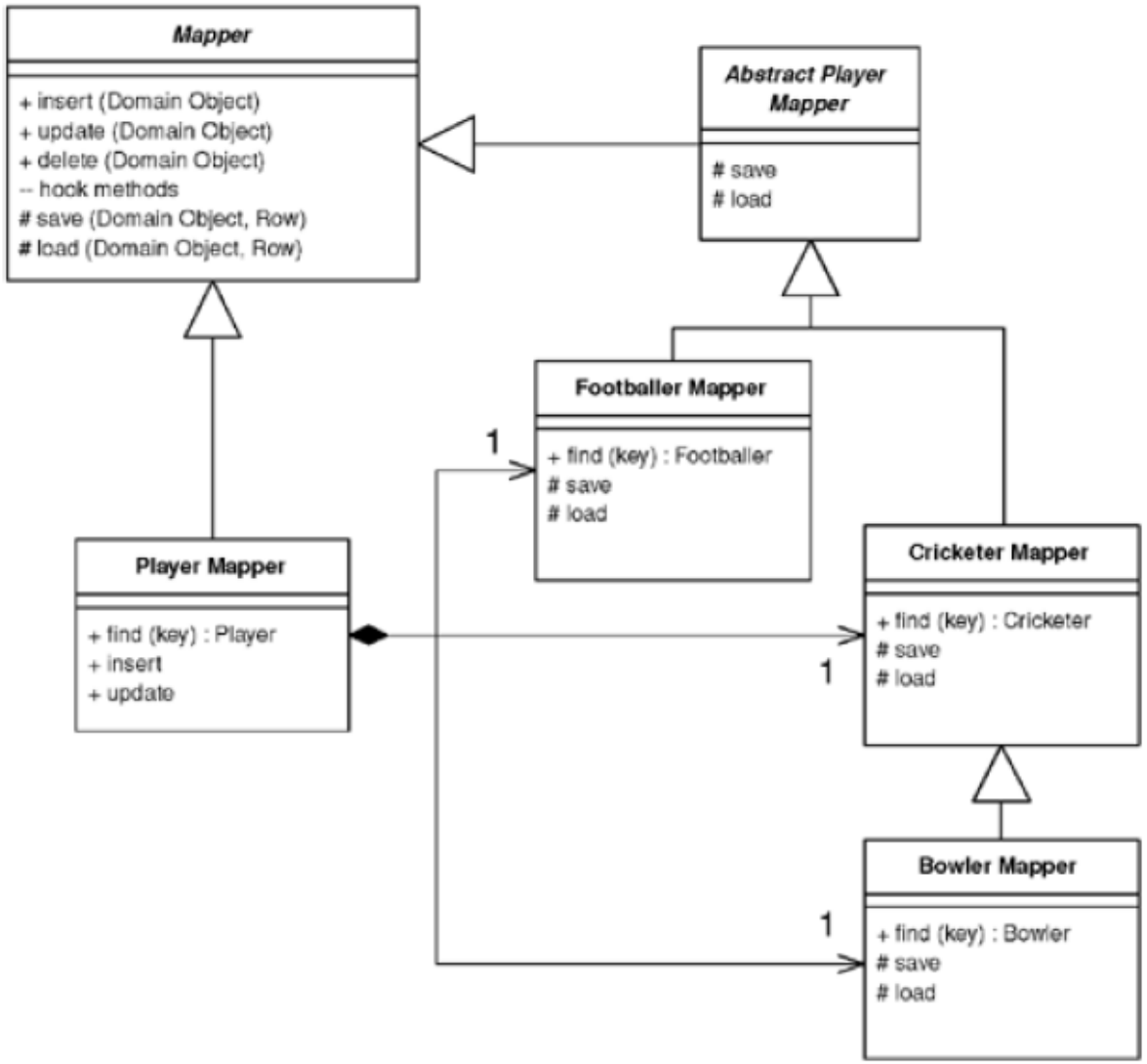
Saves a graph of objects by serializing them into a single large object (LOB), which it stores in a database field.





Object-Relational Structural Patterns

Figure 12.8. *The generic class diagram of* Inheritance Mappers (302).

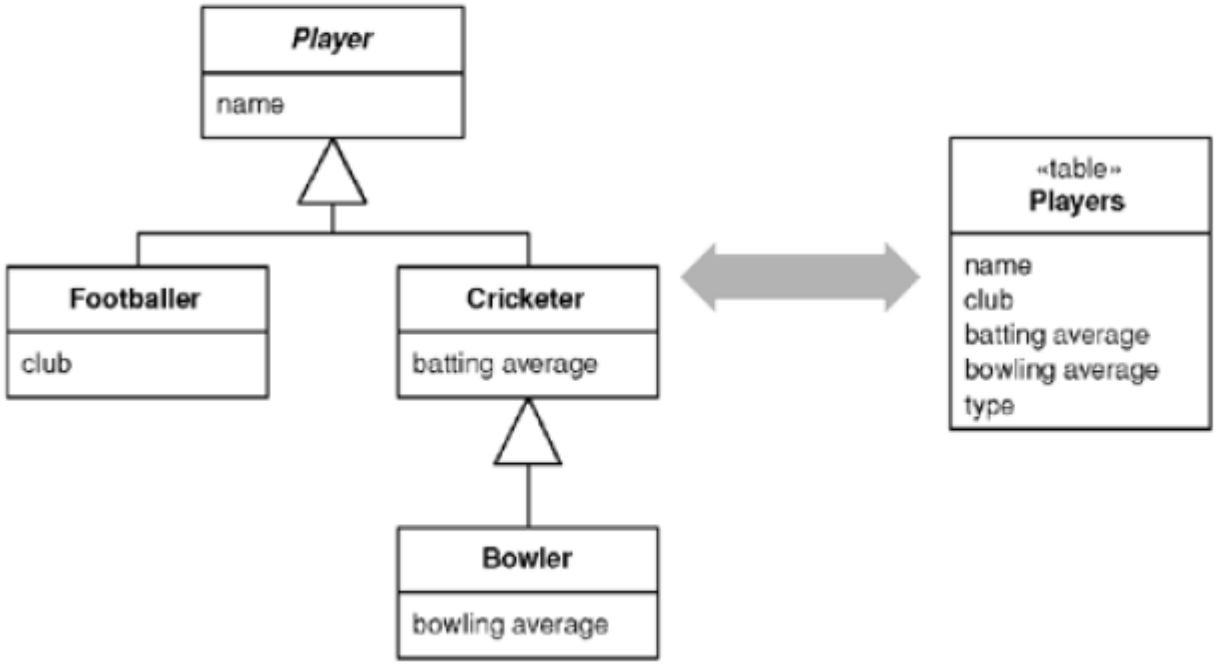


Object-Relational Structural Patterns

Single Table Inheritance

Single Table Inheritance

Represents an inheritance hierarchy of classes as a single table that has columns for all the fields of the various classes.

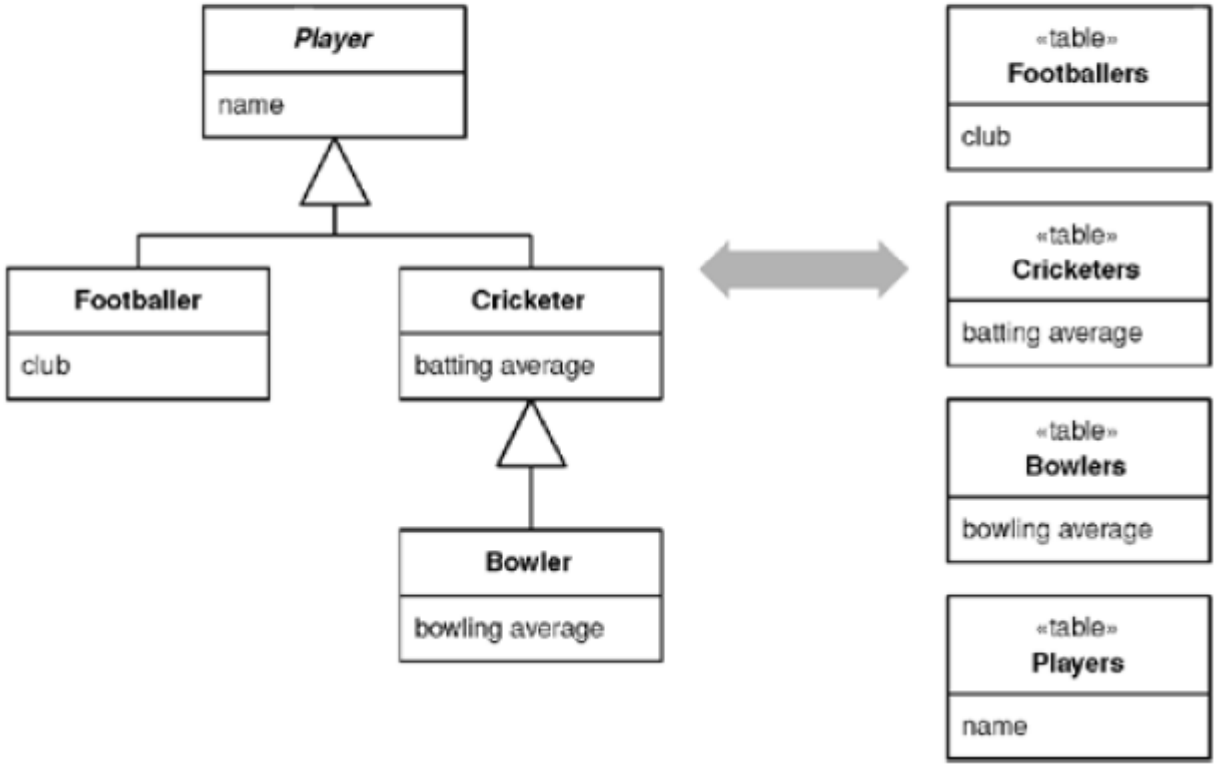


Object-Relational Structural Patterns

Class Table Inheritance

Class Table Inheritance

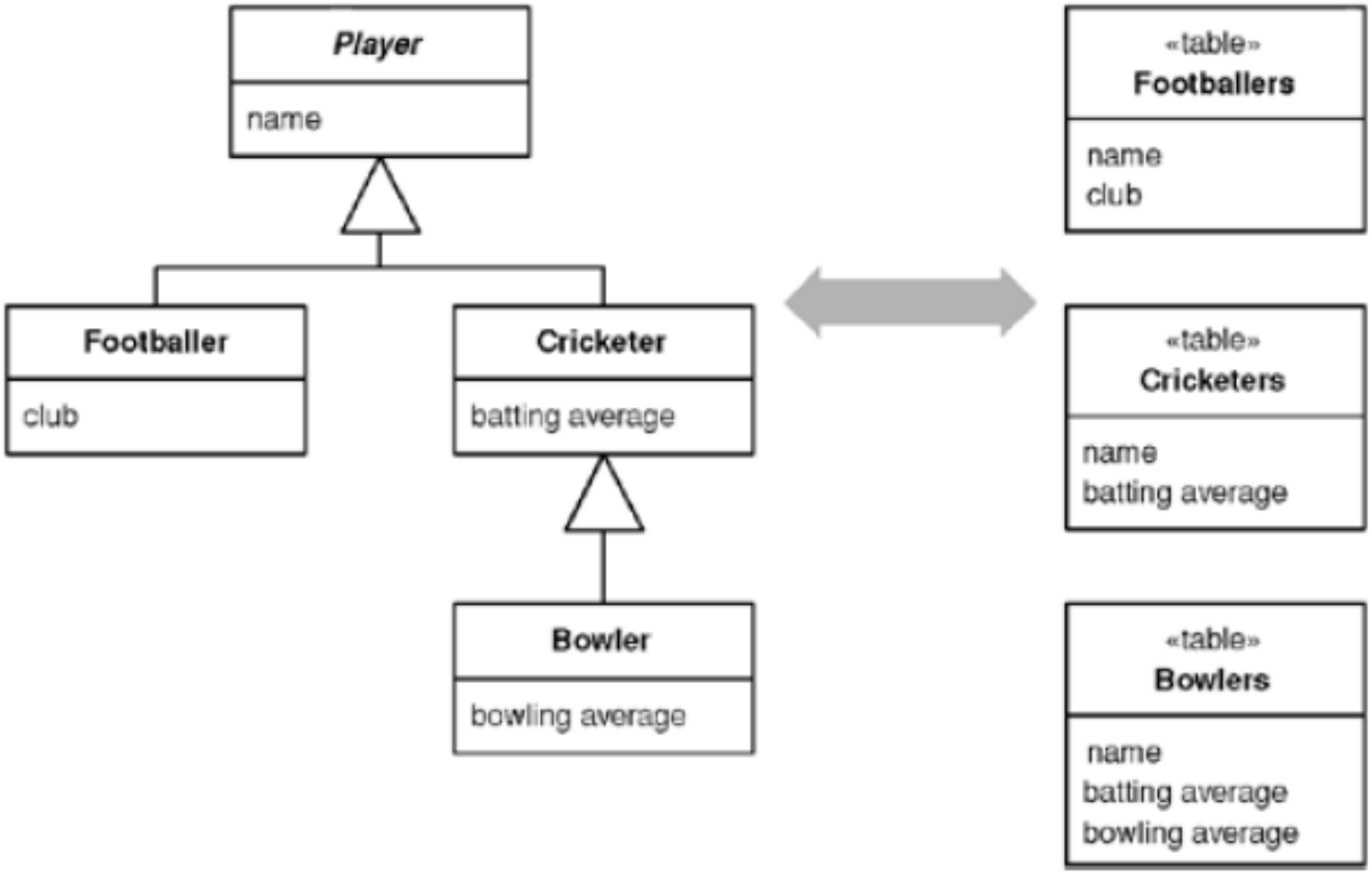
Represents an inheritance hierarchy of classes with one table for each class.



Object-Relational Structural Patterns

Concrete Table Inheritance

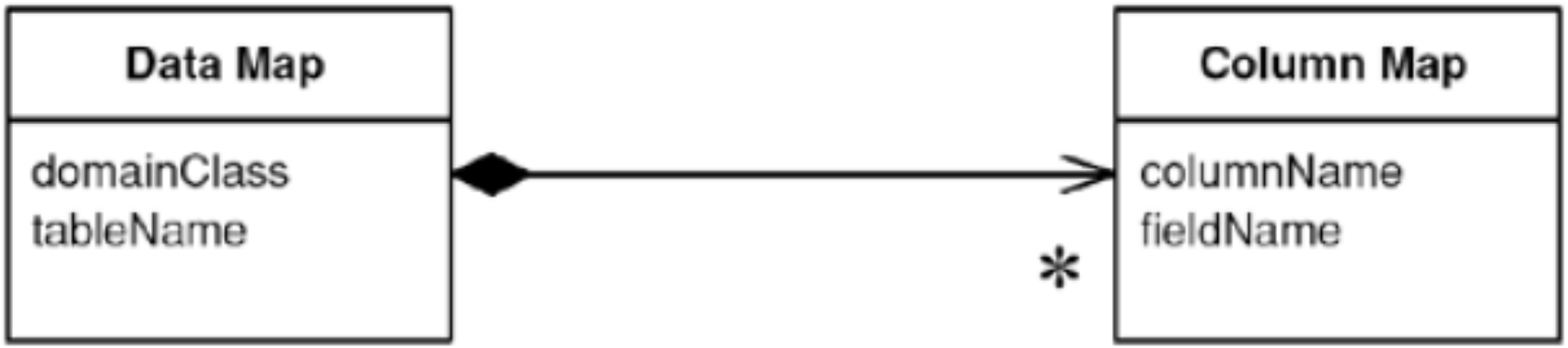
Represents an inheritance hierarchy of classes with one table per concrete class in the hierarchy.



Object-Relational Metadata Mapping Patterns

Metadata Mapping

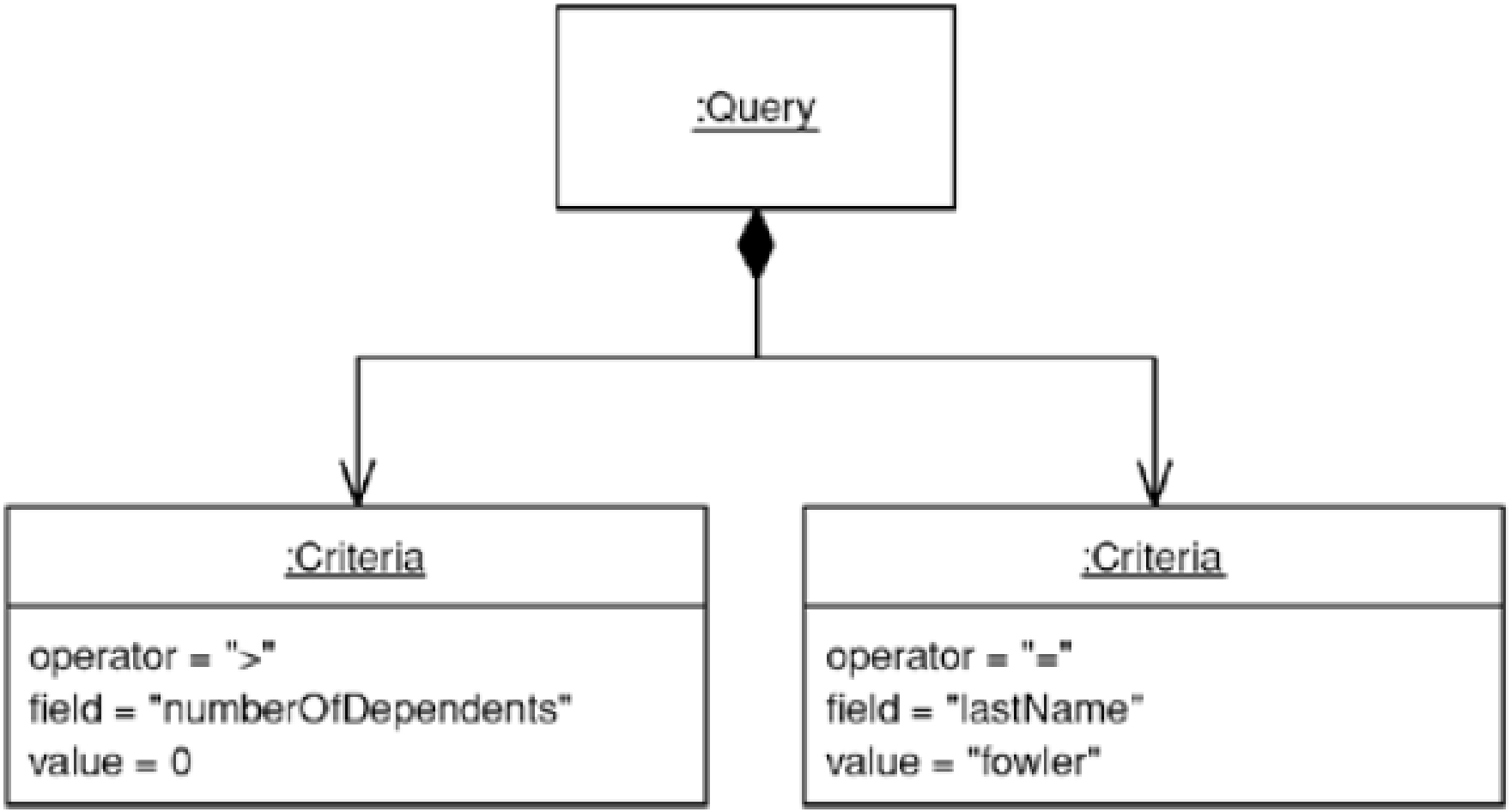
Holds details of object-relational mapping in metadata.



Object-Relational Metadata Mapping Patterns

Query Object

An object that represents a database query.

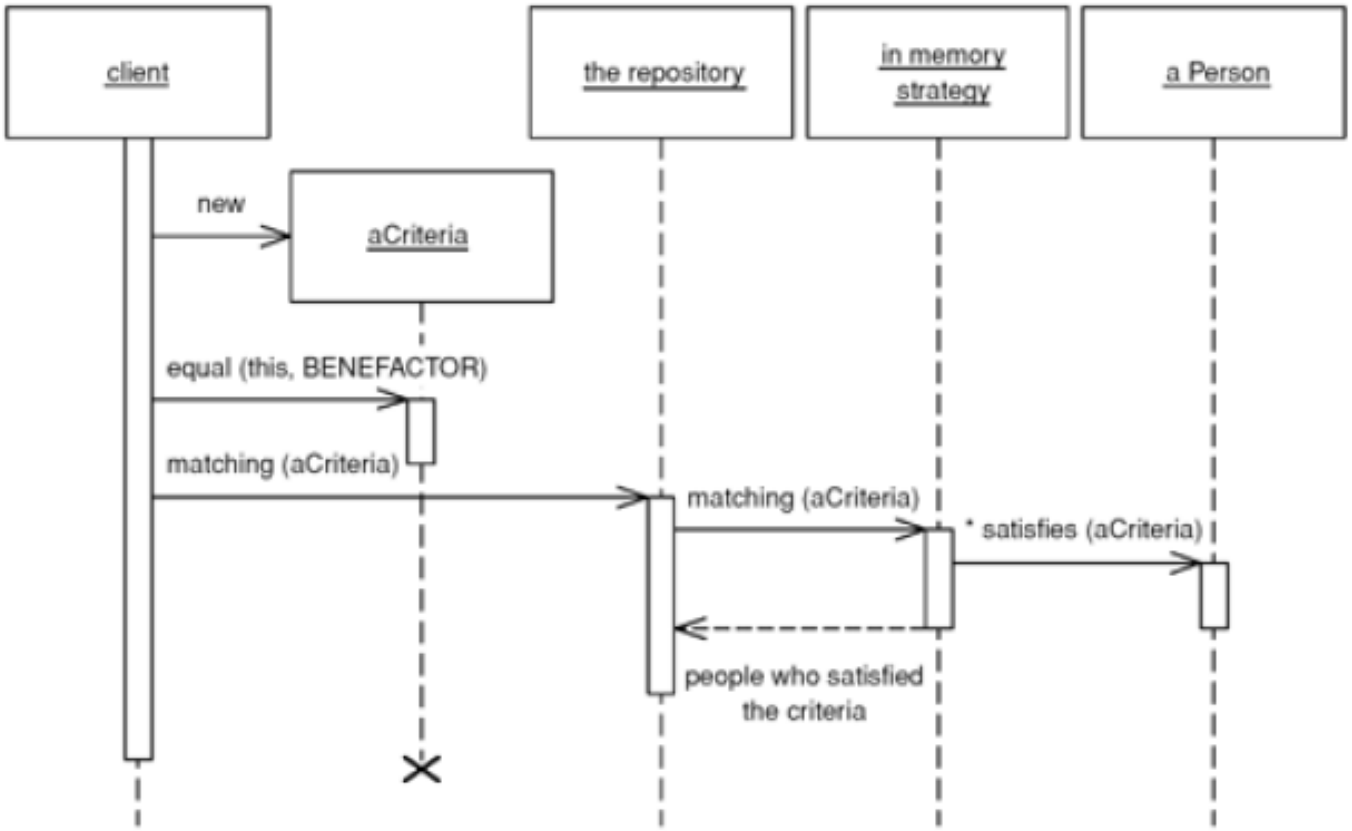


Object-Relational Metadata Mapping Patterns

Repository

by Edward Hieatt and Rob Mee

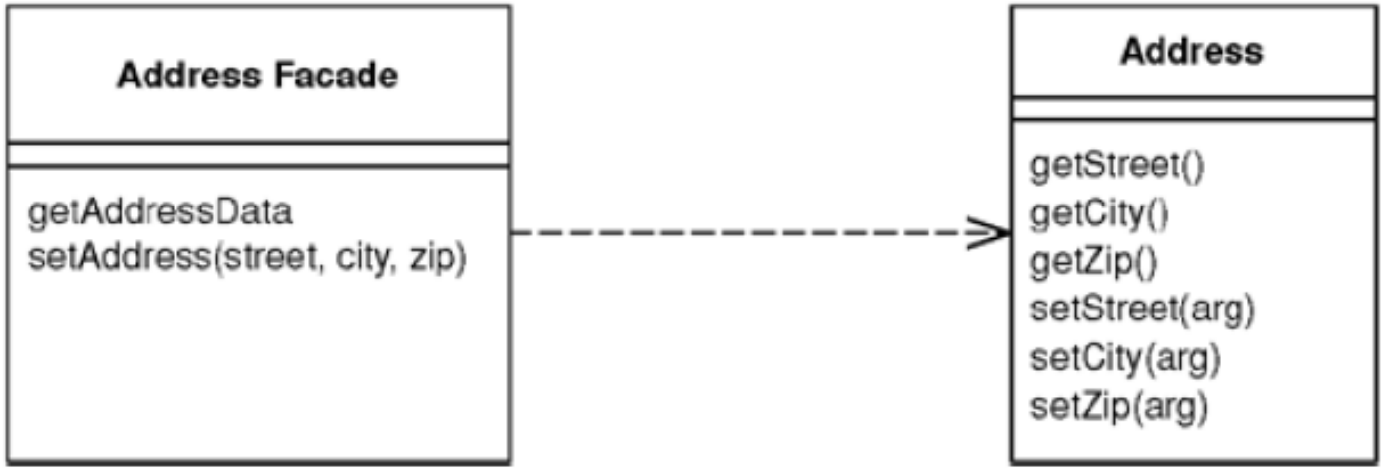
Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.



Distribution Patterns

Remote Facade

Provides a coarse-grained facade on fine-grained objects to improve efficiency over a network.



Distribution Patterns

Data Transfer Object

An object that carries data between processes in order to reduce the number of method calls.

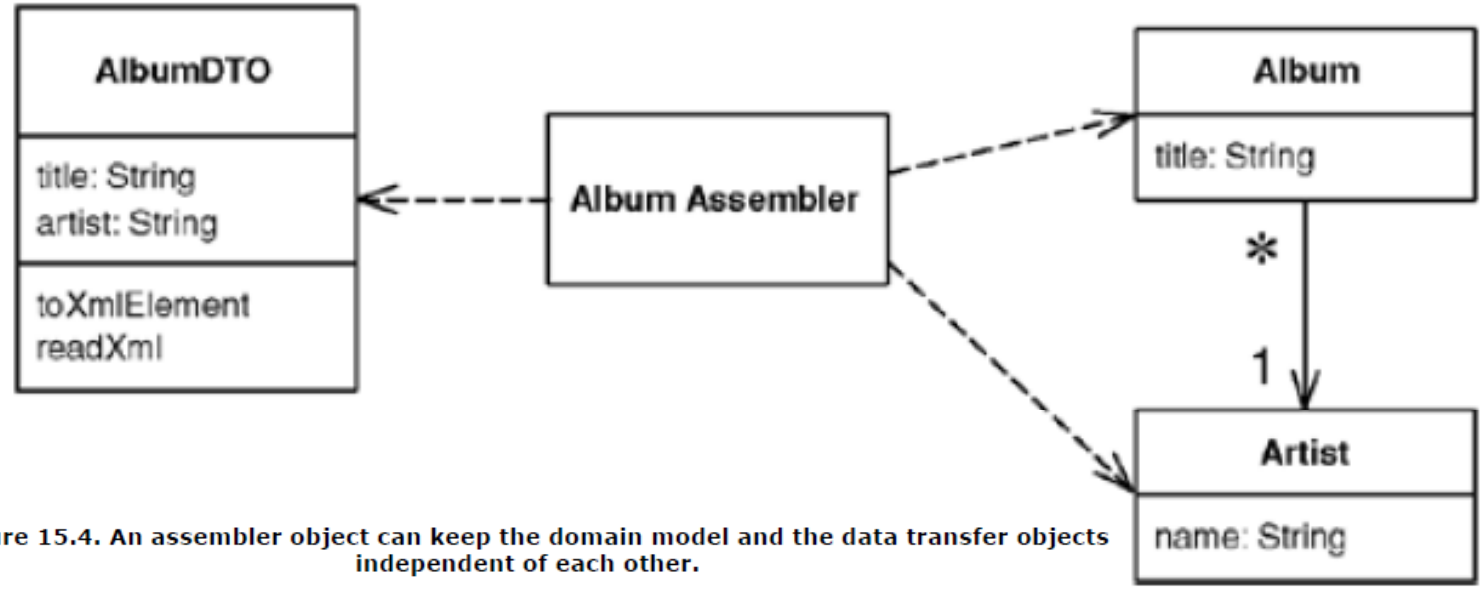
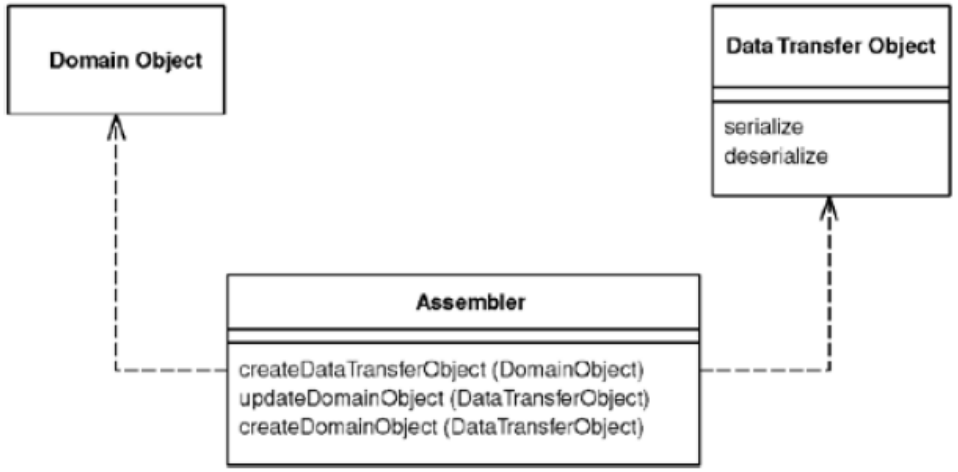


Figure 15.4. An assembler object can keep the domain model and the data transfer objects independent of each other.

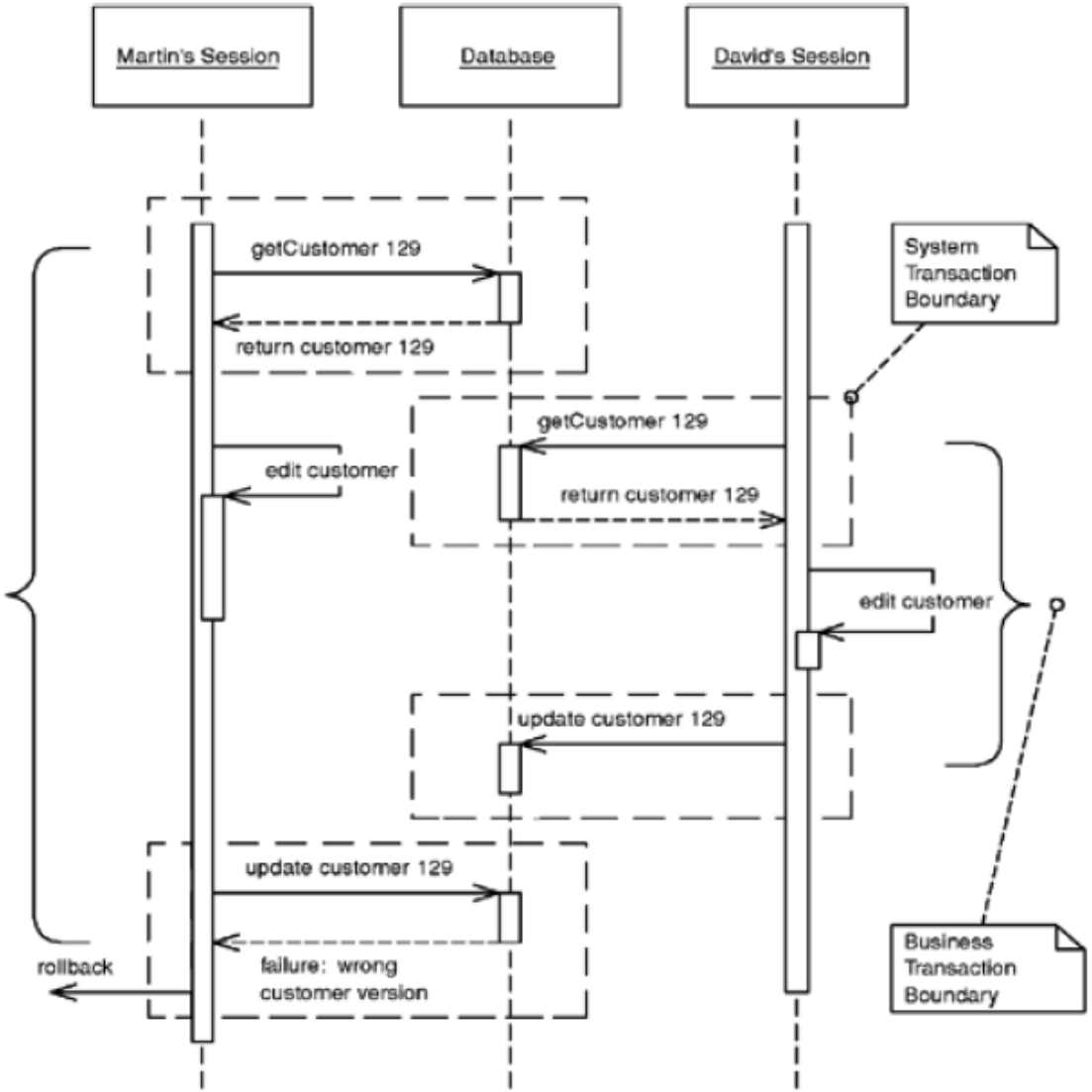


Offline Concurrency Patterns

Optimistic Offline Lock

by David Rice

Prevents conflicts between concurrent business transactions by detecting a conflict and rolling back the transaction.

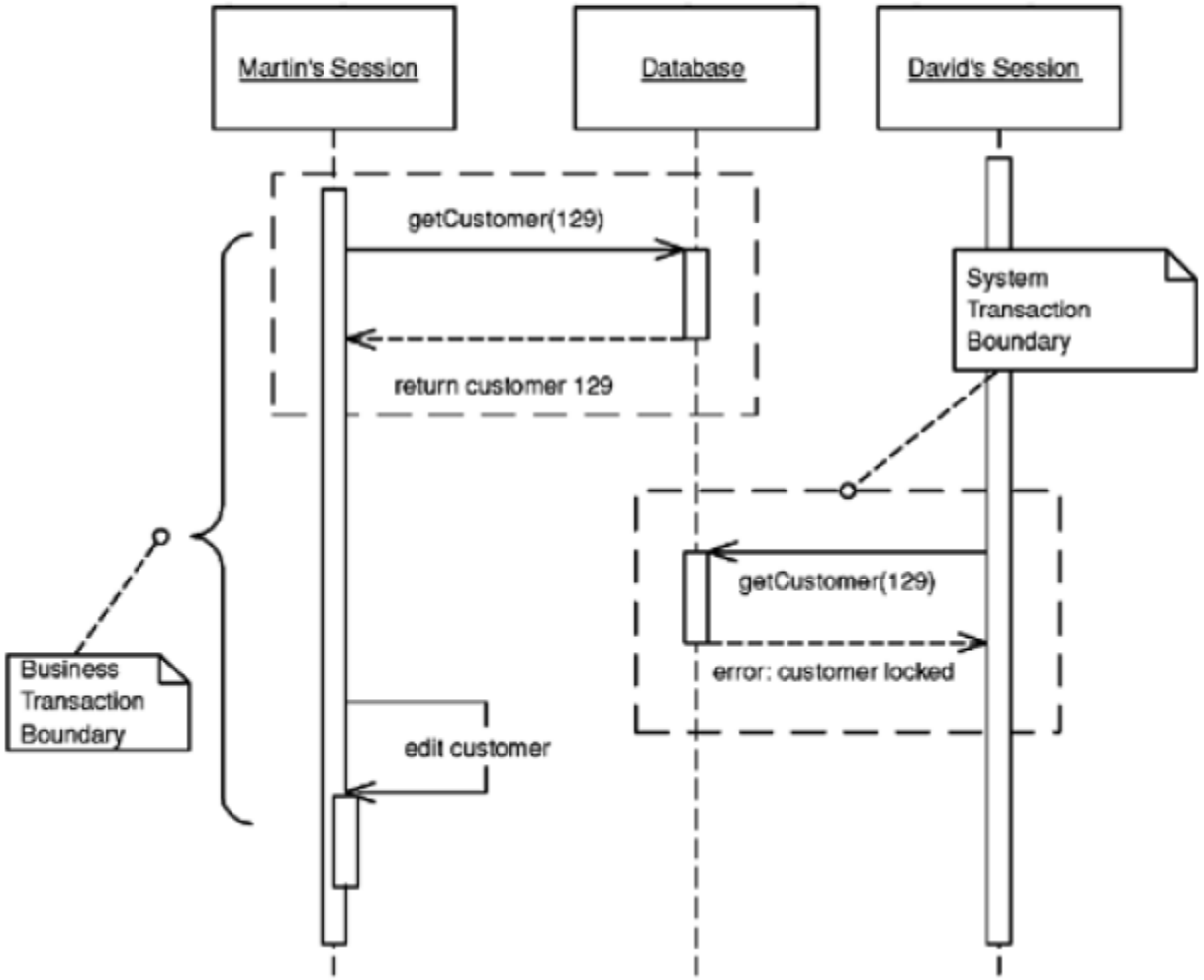


Offline Concurrency Patterns

Pessimistic Offline Lock

by David Rice

Prevents conflicts between concurrent business transactions by allowing only one business transaction at a time to access data.

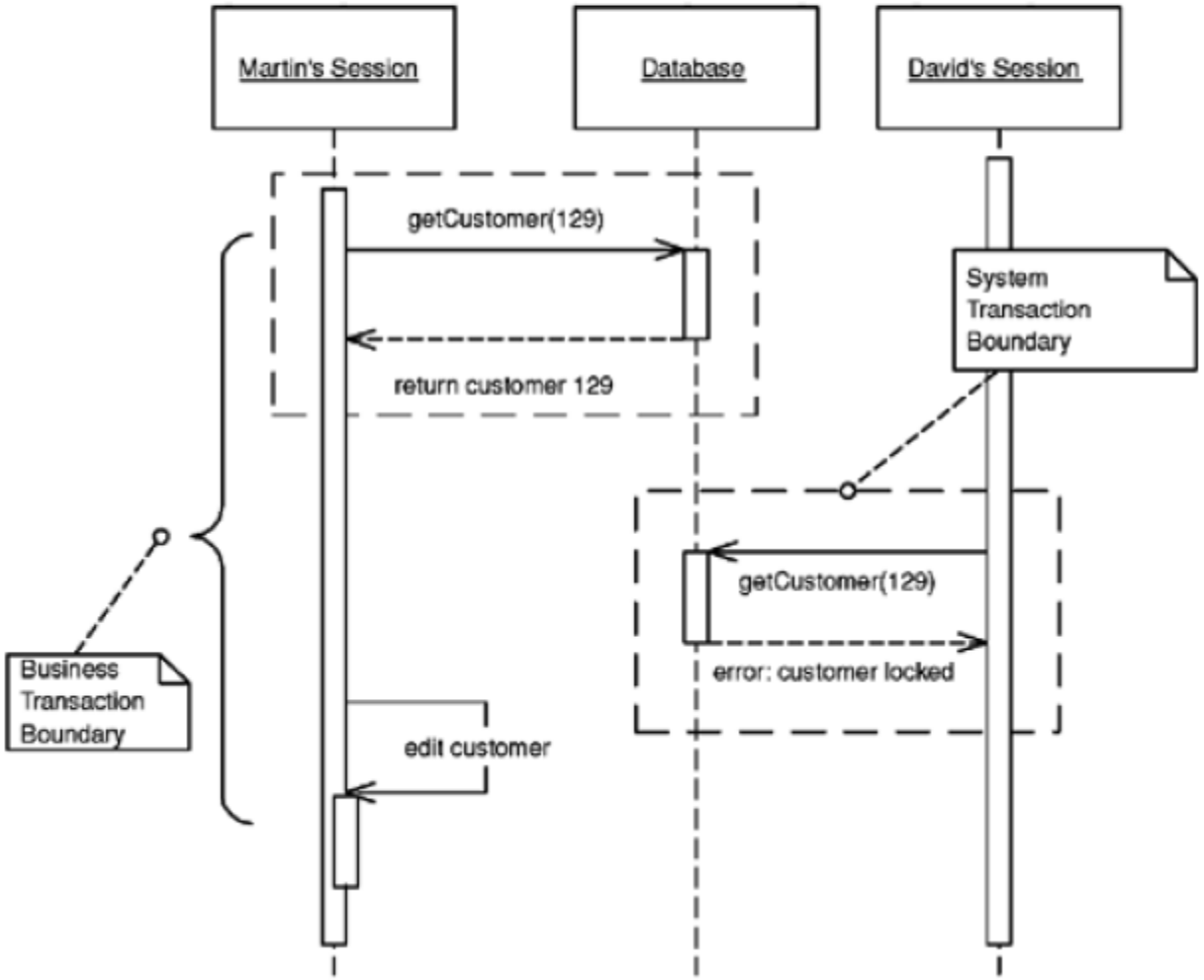


Offline Concurrency Patterns

Pessimistic Offline Lock

by David Rice

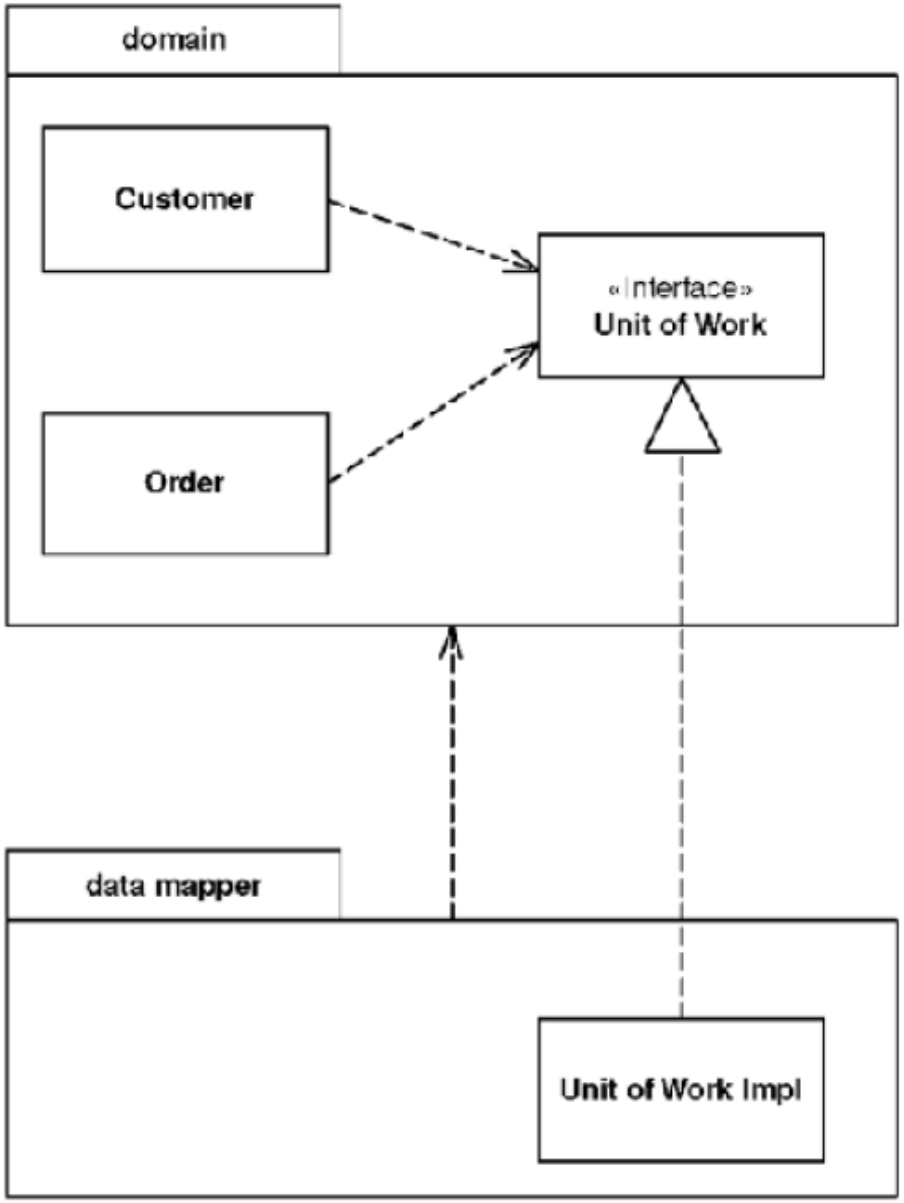
Prevents conflicts between concurrent business transactions by allowing only one business transaction at a time to access data.



Base Patterns

Separated Interface

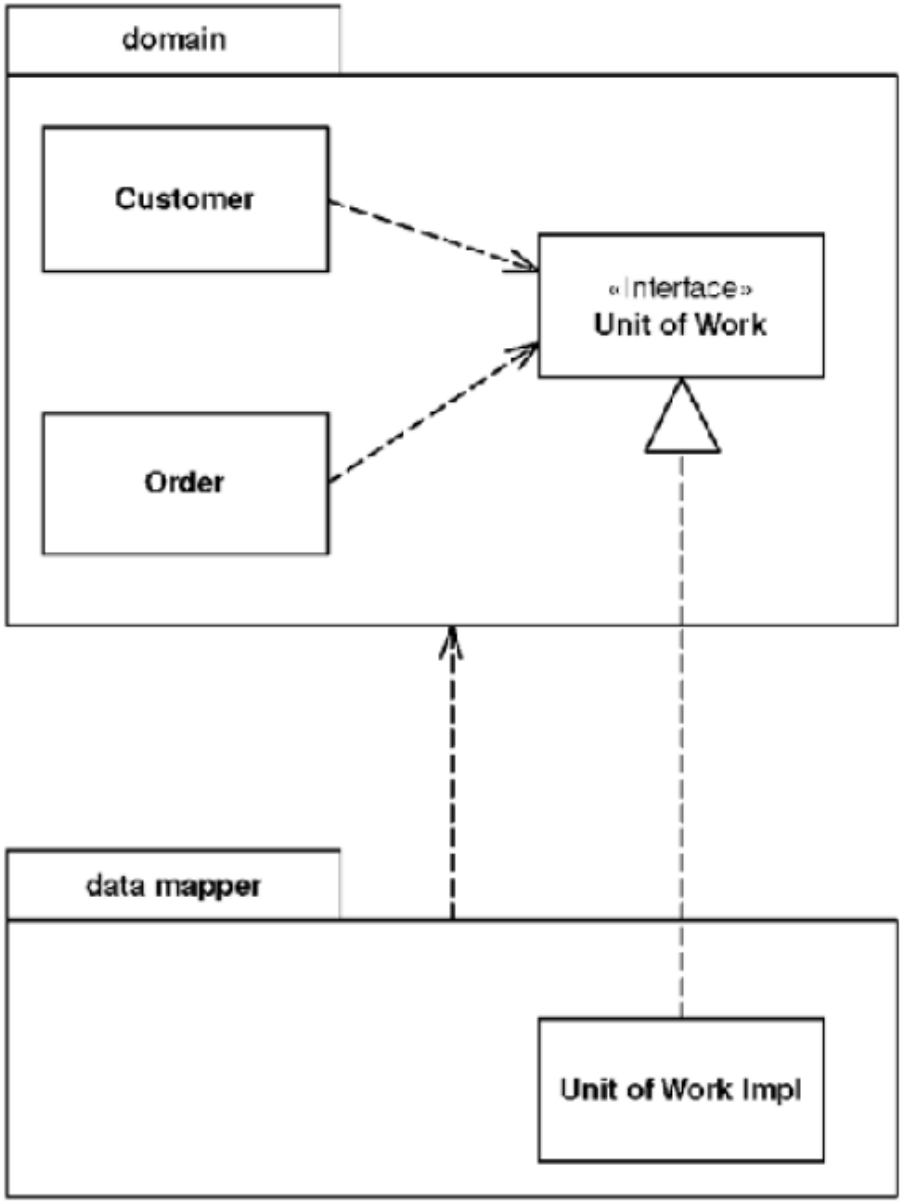
Defines an interface in a separate package from its implementation.



Base Patterns

Separated Interface

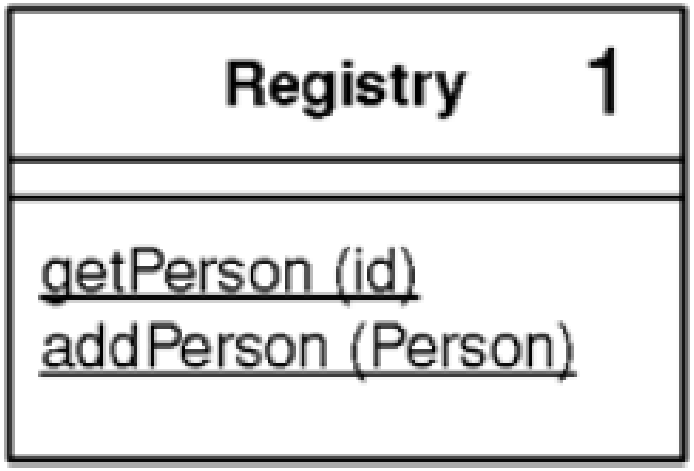
Defines an interface in a separate package from its implementation.



Base Patterns

Registry

A well-known object that other objects can use to find common objects and services.



Bibliografía

- Clean Code (Robert C. Martin)
- Code Complete, Steve McConnel (obligatorio, clásico, nivel inicial)
- Implementation Patterns, Kent Beck (nivel intermedio)
- Refactoring, Kent Beck (nivel intermedio, clásico)