Mapping to Relational Databases

Architectural Patterns

One class per database table

This is a **Database Gateway**

Row Data Gateway is when you have a class for each row returned for a query.

Record Set a generic data structure of tables and rows that mimics tabular databases.

Tabular Data Gateway allows you to query the DB and returns a Record Set

When using **Domain Model** there will be one domain class per table.

Active Record is when you have an object per table that contains moderately complex business logic.

Another way of thinking about Active Record is starting with Row Data Gateway and adding business logic.

Data Mapper is responsible for mapping data between the database and domain layers.

The Behavioral Problem

Unit of Work introduces a commit style flow to saving data to the DB. It keeps track of all the objects read and updated and stores them all in one place. A programmer would then tell the Unit of Work to commit the changes to the DB.

Without an explicitly defined **Unit of Work** the **Domain Layer** acts as the controller deciding when to read and write to the DB.

A Unit of Work results from factoring the database mapping controller behavior (the read/write logic) into it's own object.

An **Identity Map** keeps a reference to each row loaded into memory preventing duplicate loading/updating. If data is already loaded into memory the Identity Map will return a reference to it ensuring updates are properly coordinated.

Identity Map doubles as a DB cache.

Identity Map's main purpose is to maintain correctness not boost performance so a cache may still be required.

Lazy Loading objects is when there's placeholder reference on an object instead of the object itself. Only if you try to follow the reference does the data load.

Reading in Data

Prefer single queries that return more than enough data but be careful to not lock up too many rows while doing so.

Structural Mapping Patterns

These are useful when dealing with compositional hierarchies.

Identity Field - A field that references the id to a foreign object in a separate table.

Associate Table Mapping - A separate table to handle many-to-many relationships in relational databases. Typically id-to-id rows.

Serialized LOB - Serialized Large Object

BLOB - Binary Large Object

CLOB - Character Large Object

Inheritance (pg. 45)

These are useful when dealing with class hiearchy linked by inheritance.

Each tradeoff is between the duplication of data structure and speed of access.

Single Table Inheritance - One table for all the classes in a class hiearchy. This is when you make a mapping object that maps all of the seperate values onto one class. Caution, whenever a super class is changed you must to alter the tables and the mapping code.

Concrete Table Inheritance - One table for each concrete class. Avoids joins but is brittle to changes. The top level class's data is duplicated onto each concrete class table allowing querying without joins.

Class Table Inheritance - One table per class in the hierachy. This is the simplist relationship between the classes and the tables but it takes multiple joins to load a single object which usually reduces performance.

With any particual task, start off by thinking about the *Domain Model* first and inegrate each piece in the database as you go.

If the schema already exists determine if it is simple or complex domain logic.

If it's simple build Row Data Gateway or Table Data Gateway clasic taht mimic the database and layer the domain logic on top of it.

If it's more complex gradually build up a Domain Model and include Data mappers to persist the data to the existing database.

Double Mapping

When dealing with *data with many differences* it's wise to have multiple mapping layers, one for each data source.

Sometimes similar data needs to be pulled from more than one source. Multiple databases may hold similar data but with small differences.

When dealing with *very similar data* with little differences between the tables is to have two step mapping layers.

The first mapping layer converts data from the in-memory schema to a logical data store schema. The logical data store schema is designed to maximize the similarities in the data source formats.

The second step maps the logical data store schea to the physical data store schema. This is step contains all the differnces.

The mapping from the logical data store (in memory) to the physical data store should be used as a **Gateway** and should use any of the mapping techniques to map from the application logic to the logical data store.

Using Metadata