Object Relational Mappers

CMPU4023 - Enterprise Application Development

Relational Model

- The dominant database of choice in the enterprise is the traditional relational database model or RDBMS
- The relational model derives from mathematical set theory and provides a very powerful and robust means of storing and analysing data
- The big trade-off is that real world data must be modeled as *entities* and *relationships* through sets of tuples of attributes but which allows a powerful dynamic querying capability through SQL
- But another trade-off is that relational model is not how most application software abstracts its view of data so there is often a mismatch between the application and the database

Logical Data Model

- The LDM is the software application's abstraction of the persistent storage system, i.e. a database
- The model presents a programmer view of the state and behaviour of the underlying stored entities and relationships
- In most cases, the model will be implemented using higher order language features such as classes, objects and interfaces
- A key question is how the application models can interoperate with the database relations, how data is read and written, how entity relationships are captured and how ACID properties are maintained

Relations or Objects?

- There is a choice between embracing the relational model or eschewing it in favour of an object-oriented approach or adopting a hybrid, more middle-ground approach
- Adopting a relational approach would see the intermingling of SQL and data-as-rows into the application logic - programming relationally with CRUD operations as if in an SQL environment
- An object approach would see the adoption of an object-relational mapper which would hide the details of the SQL and relational modeling to a large and expose objects and methods to implement the CRUD operations

Object-Relational Mappers

- An ORM is a library which exposes the database models and operations on those models in a more natural way from the perspective of the middleware language. In simple terms:

Relational	**OOP**
Table	Class
Row	Instance
Attribute	Member Field
SQL	Class or Instance Methods

Schema Mapping

- There are two broad approaches taken in the ORM world to schema mapping
- In either case, the idea is to have one side of the mapping be the single source of truth and the other side be automatically mapped and synchronised

1. **Derive the relational model from OOP code or configuration** - the programmer defines the schema on the OOP side and the ORM generates and executes the required DDL.
2. **Derive the OOP code from the table** - the ORM uses database introspection to query an already-defined database schema to build the models on the OOP side

ORM Query Formation

- Database operations are expressed in OOP syntax which automatically generates the appropriate SQL for execution in the database
- The results are automatically marshalled back into objects for processing by the programmer. For example, in SequalizeJS:

```
Post.findAll({
  where: {
    authorId: 12,
    status: 'active'
  },
  count: 100,
  offset: 50
});
```

```
SELECT * FROM posts
WHERE author_id = 10
AND status = 'active'
OFFSET 50
LIMIT 100;
```

ORM - The Arguments For

- Single language to develop middleware and data access logic which is cleaner to express than intermingling SQL syntax
- Cleaner abstraction of data tier, insulating the middleware from the vagaries of particular database details
- More portable if database provider needs to be swapped out
- ORM can generate better SQL code than the middleware programmer
- Isolates the middleware developers from having to deal with the database maintainers, particular concern in a heavily demarcated enterprise environment

ORM - The Arguments Against

- The ORM is a false friend, representing another complex component to learn and use when a perfect good one (relational SQL) already exists
- ORMs can never give access to the power of the relational model or the advanced query capabilities of SQL such as window functions, joins and aggregations, constraints or partitioning
- In practice, hand-written SQL by an expert is likely to be better than anything an ORM can generate
- Transactional integrity is harder to guarantee outside the database

ORM is an Anti-Pattern

- **Inadequate abstraction** - in practice an ORM does not actually fully abstract the relational model. In any moderately complex application you have to know the SQL you want the ORM to generate only to have to learn how to get the ORM to do it
- **Incorrect abstraction** - Assuming that the relational model is the correct model for the data, then mapping them to objects misses the point that SQL is about querying and answering questions about the data not about in-memory representations of the answers as objects
- **Inefficient** - In order to do so much processing in middleware, the data has to be continually fetched from the database into memory

Alternative to ORMs

- Make the service models about wrapping the database query and updates and not about replacing the relational model
- Embrace SQL inside the models to carry out the CRUD operations allowing whatever results structure is appropriate for the API
- Treat schema definition and DDL as a separate problem outside of the service
- Don’t be fooled by so-called separation-of-concerns arguments if using a relational model - business logic is very much the concern of the database
- If a dumb objects store is what you need, then don’t use a relational database in the first place

Summary

- An ORM is a database API which tries to abstract the relational model and SQL and as OOP classes, objects and interfaces
- The programmer can, in many cases, deal exclusively with the storage tier in the same language as the middleware
- But an ORM can be a false friend introducing yet another complex layer that doesn’t even properly abstract the relational model
- ORMs can be less performant due to inefficient query generation, unnecessary data fetches and increased object creation and garbage collection overhead
- If you are using the relational model, embrace SQL!

Object Relational Mappers

CMPU4023 - Enterprise Application Development