Term paper
ORM in DBMS

Name: Bhavin Raichura
Roll no: 19111019
Subject: DBMS
Branch: Biomedical Engineering

Abstract

One of the challenges of using object-oriented programming (OOP) languages and databases is the complexity of aligning the programming code with database structures. Object-relational mapping (ORM) is a technique that creates a layer between the language and the database, helping programmers work with data without the OOP paradigm.

The necessity to learn and code in structured query language (SQL) in order to link their application to a SQL database is a problem for OOP developers. Data-access code can be written by developers who are familiar with SQL. Because the developer must extract the data items from the code strings, this raw SQL coding might take a long time.To provide extra information about the data, SQL query builders provide a layer of abstraction to the SQL code. Developers, on the other hand, must be able to read and write SQL.

In this term paper I will provide an overview of ORMs, and compare them with SQL tools using an example of an database based application using python.

This term paper will highlight the following aspects:

1. Introduction: ORM
2. How ORM Works
3. Common Languages Between ORMs and OOPs
4. ORM vs Raw SQL
5. OOPs Concepts
6. Some Basic SQL Operations in ORM using SQLAlchemy
7. Features
8. Pros of ORM
9. Cons of ORM
10. Conclusion
11. Reference

1 Introduction: ORM

An object-relational mapper provides an object-oriented layer between relational databases and object-oriented programming languages(like C++, C, Python, JAVA) without having to write SQL queries. It standardizes interfaces reducing boilerplate and speeding development time.

Object-oriented programming includes many states and codes in a format that is complex to understand and interpret. ORMs translate this data and create a structured map to help developers understand the underlying database structure. The mapping explains how objects are related to different tables. ORMs use this information to convert data between tables and generate the SQL code for a relational database to insert, update, create and delete data in response to changes the application makes to the data object. Once written, the ORM mapping will manage the application's data needs and you will not need to write any more low-level code.

2 How ORM Works

It operates on the objects. So the whole methodology followed by ORMs is dependent on the object-oriented paradigm. ORMs generate objects which map to tables in the database virtually. It create a model of the object-oriented program with a high-level of abstraction. In other words, it makes a level of logic without the underlying details of the code. Mapping describes the relationship between an object and the data without knowing how the data is structured. The model can then be used to connect the application with the SQL code needed to manage data activities. This "plumbing" type of code does not have to be rewritten, saving the developer a tremendous amount of time.

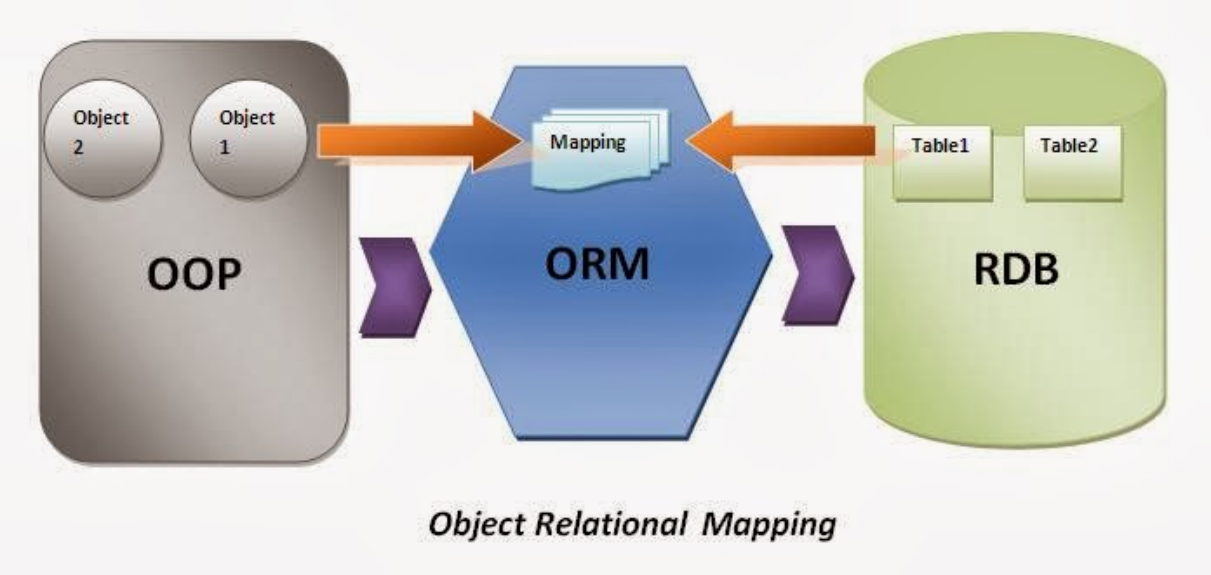

Figure 1:Working of ORM

3 Common Languages Between ORMs and OOPs

- Hibernate: JAVA
- Django ORM: Django
- SQL Alchemy: Flask
- Dapper ORM: C#
- JOOQ ORM: JAVA
- Doctrine: PHP

4 ORM vs Raw SQL

Developers can use raw SQL code to write a direct interface between the application and the database. Most relational databases support SQL to build data interfaces and applications. It's stable, and since SQL has been used since the 1970s, it's well documented and supported. Programmers maintain a lot of control over their data interface with SQL. It requires a lot of work, but it is more flexible and detailed than an ORM abstraction.

Object Relational Mapping

```python
from django.db import models

# Create table using OOP
class STUDENT(models.Model):
    Rollno = models.IntegerField( max_length=50)
    Name = models.CharField(max_length=200)
    Class = models.IntegerField(max_length=12)
    Section = models.CharField(max_length=50)

# Insert data into table using ORM
create_user= STUDENT(Rollno=1001,Name="rahul",Class=12,Section='B')
create_user.save()

# fetch data from database using ORM
STUDENT.objects.filter(Rollno=1001)
```

Raw SQL Queries

```python
import mysql.connector as connector
# connect to database
mydb = connector.connect(host="localhost",
        database="HOSPITAL_BED_AVAILIBILITY")
db_cur = mydb.cursor()
# Create table using raw sql
db_cur.execute("""create table STUDENT(
        rollno INT PRIMARY KEY,
        name TEXT,
        class INT,
        section CHAR)""")
# Insert data into database
db_cur.execute("""insert into STUDENT values(1001,"krisha",12,'b');""")
mydb.commit()
# Fetch data from database
db_cur.execute("select * from BEDS;")
datas = db_cur.fetchall()
```

Figure 2: ORM vs Raw SQL

4.1 Native Querying with SQL

Using raw SQL also has its drawbacks. For instance, the developer is responsible for the safety and security of the database code. SQL injection is a problem where user input can affect the data state causing issues with the application and data integrity. ORMs sanitizes the code, making it easier to avoid these problems.

4.2 SQL Query Builders

Query builders add a layer of abstraction over the raw SQL without masking all of the underlying details. The builders formalize querying patterns and add methods to or functions that add escape items for easier application integration. They add a templating layer to help developers understand the database structure within the same coding application. Template builders still require developers to understand the database structure, requiring them to know SQL.

5 OOPs Concept

Object oriented programming is a type of programming which uses objects and classes its functioning. The object oriented programming is based on real world entities like inheritance, polymorphism, data hiding, etc. It aims at binding together data and function work on these data sets into a single entity to restrict their usage.

Some basic concepts of object oriented programming are

- CLASS
- OBJECTS
- ENCAPSULATION
- POLYMORPHISM
- INHERITANCE
- ABSTRACTION

5.1 Class

A class is a data-type that has its own members i.e. data members and member functions. It is the blueprint for an object in object oriented programming language. It is the basic building block of object oriented programming in c++. The members of a class are accessed in programming language by creating an instance of the class.

Some important properties of class are

- Class is a user-defined data-type.
- A class contains members like data members and member functions.
- Data members are variables of the class.

- Member functions are the methods that are used to manipulate data members.
- Data members define the properties of the class whereas the member functions define the behaviour of the class.

A class can have multiple objects which have properties and behaviour that in common for all of them.

5.2 Objects

An object is an instance of a class. It is an entity with characteristics and behaviour that are used in the object oriented programming. An object is the entity that is created to allocate memory. A class when defined does not have memory chunk itself which will be allocated as soon as objects are created.

5.3 Encapsulation

In object oriented programming, encapsulation is the concept of wrapping together of data and information in a single unit. A formale defination of encapsulation would be: encapsulation is binding togather the data and related function that can manipulate the data.

5.4 Polymorphism

The name defines polymorphism is multiple forms. which means polymorphism is the ability of object oriented programming to do some work using multiple forms. The behaviour of the method is dependent on the type or the situation in which the method is called.

5.5 Operator overloading

In operator overloading and operator can have multiple behaviour in different instances of usage.

5.6 Function overloading

Functions with the same name that can do multiple types based on some condition.

5.7 Inheritance

It is the capability of a class to inherit or derive properties or characteristics other class. it is very important and object oriented program as it allows reusability i.e. using a method defined in another class by using inheritance. The class that derives properties from other class is known as child class or subclass and the class from which the properties are inherited is base class or parent class.

5.8 Abstraction

Data abstraction or Data Hiding is the concept of hiding data and showing only relevant data to the final user. It is also an important part object oriented programing.

6 Some Basic SQL Operations in ORM using SQLAlchemy

In general we would need to write raw SQL queries, pass them to the database engine and parse the returned results as a normal array of records.

In this section I am using python's one of the most common library for ORM is **SQLAlchemy**. SQLAlchemy provides a nice "Pythonic" way of interacting with databases. So rather than dealing with the differences between specific dialects of traditional SQL such as MySQL or PostgreSQL or Oracle, you can leverage the Pythonic framework of SQLAlchemy to streamline your workflow and more efficiently query your data.

6.1 SQLAlchemy

SQLAlchemy is a library that facilitates the communication between Python programs and databases. Most of the times, this library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements.

6.2 Connecting to a database

To start interacting with the database we first we need to establish a connection. Object of MetaData class from SQLAlchemy Metadata is a collection of Table objects and their associated schema constructs. It holds a collection of Table objects as well as an optional binding to an Engine or Connection.

```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, String
engine = create_engine('sqlite:///college.db', echo = True)
meta = MetaData()
```

Figure 3: Connection with database

6.3 Create Table

We are creating a SQLite database college.db with a students table in it. The create_all() function uses the engine object to create all the defined table objects and stores the information in metadata.

ORM

```
students = Table(
   'students', meta,
   Column('id', Integer, primary_key = True),
   Column('name', String),
   Column('lastname', String),
)
meta.create_all(engine)
```

Raw SQL

```
CREATE TABLE students (
   id INTEGER NOT NULL,
   name VARCHAR,
   lastname VARCHAR,
   PRIMARY KEY (id)
)
```

Figure 4: Create table (ORM vs SQL)

6.4 Insert Data

SQL expressions are constructed using corresponding methods relative to target table object. For example, the INSERT statement is created by executing insert() method as follows:

ORM

```
ins = students.insert()
ins = students.insert().values(name = 'Ravi', lastname = 'Kapoor')
conn = engine.connect()
result = conn.execute(ins)
```

Raw SQL

```
INSERT INTO
    students (name, lastname)
VALUES ('Ravi', 'Kapoor');
```

Figure: Insert data into database

6.5 Print Data

The select() method of table object enables us to construct SELECT expression. Using this we can print data from table.

ORM

```
s = students.select()
conn = engine.connect()
result = conn.execute(s)

for row in result:
```

Raw SQL

```
SELECT id, name, lastname
FROM students'
```

Output

```
(1, 'Ravi', 'Kapoor')
(2, 'Rajiv', 'Khanna')
(3, 'Komal', 'Bhandari')
(4, 'Abdul', 'Sattar')
(5, 'Priya', 'Rajhans')
```

Figure 6: Print data from database

6.6 Filter Data using 'Where'

As similar to the SQL in ORM we have where() method to filter rows. so we can use it like this:

Figure 7:Print data from database with condition

7 Features

- It makes the application independent of the database management system being used in the backend, and so you can write a generic query. In case of migrating to another database, it becomes fairly a good deal to have ORM implemented in the project.
- Hassles of coders are reduced to learn SQL syntaxes separately for whichever database being used to support the application. Coders can shift their focus on optimizing the code and improving performance rather than dealing with connectivity issues.
- All small or big changes can be carried out via ORM, so there are no such restrictions when we deal with data. For example, JDBC comes with a lot of restrictions on extracting a result-set, process it and then commit it back to the database. This is not the case with ORMs. Even a single cell in the database can be retrieved, changed and saved back.
- The connection becomes robust, secure as there will be less intervention in code. It will handle all the necessary configurations required to map application programming language with the database's query language since there will be lesser intervention promoting secure application as a whole.
- There is a fairly large deal of ORMs present in the market as per the application language used. One can choose easily as per business requirements.
- There is an attached disadvantage in using ORM as well. That is when the database is in legacy file systems and disarranged. It becomes a task to arrange a whole lot of data and then map this with ORM. It is thereby suggested to use ORM when the back end is fairly managed.

8 Pros of ORM

ORM tools are popular with OOP developers because they minimize the amount of SQL knowledge required to connect a database to an application. ORMs also automatically generate the SQL code, allowing you to focus on generating business logic. There are four significant benefits of using object-related mappers to manage the interface between applications and databases.

8.1 Productivity

Writing data-access code is time-consuming and doesn't add a lot of value to the application's functionality. It's essentially the plumbing of the code. Using a tool like an ORM that automatically generates the data-access code saves tremendous development time that does not add value to the application. In some cases, the ORM can write 100 percent of the data-access code for the application. The ORM can also help you keep track of database changes making it easier to debug and change the application in the future.

8.2 Application Design

A well-written ORM will implement design patterns to force you to use best practices for application design. If you use an ORM to manage the data interface, you do not need to create the perfect database schema in advance. You will be able to change the existing interface easily. Separating the database table from the programming code also allows you to switch out data for different applications.

8.3 Code Reuse

One way to reuse data is to create a class library to generate a separate dynamic-link library (DLL). You can create a new application without needing to duplicate the data-access code.

8.4 Reduced Testing

Since the code generated by the ORM is well-tested, you do not need to spend as much time testing the data-access code. Instead, you can focus on testing the business logic and code.

9 Cons of ORM

ORMs are an excellent tool for many applications, but some developers found several drawbacks in using ORMs for data-access applications. The issues seem to correlate with the complexity of the application. With simple applications, having a high-level of abstraction helps the development process. But when the applications are complex, abstraction covers up many details needed to address data-related issues.

9.1 Performance

A common complaint among OOP developers is the extra code generated by the ORM. The added code slows application performance and makes it harder to maintain. A well-designed ORM should be able to create high-quality code without affecting application speed.

9.2 Need to Know SQL

High-level abstractions do not always generate the best SQL code, and developers cannot rely on the ORM 100 percent of the time. You still need to know SQL as well as the syntax generated by the ORM.

9.3 Poor Mapping

ORMs can sometimes create an incorrect mapping between data tables and objects. These problems can cause application problems and be difficult to recognize. ORMs also encourage one-to-one mapping even though it is rare that business applications have many one-to-one relationships.

10 Conclusion

Writing SQL code to attach a relational database to an object-oriented application can be a time-consuming activity that generates little value to the business application. Developers can write raw SQL code or use SQL query builders to improve the process, but both methods still require in-depth database knowledge and the ability to code in SQL. ORMs enhance productivity by creating highly-abstract data models and automatically generating SQL code. These tools also make it easier to separate the database from the programming logic giving developers more flexibility. But ORMs have their detractors. Common complaints include reduced performance, extra coding, and poor mapping depending on the ORM quality.

11 Reference

1. https://www.altexsoft.com/blog/object-relational-mapping/
2. http://www.diva-portal.org/smash/get/diva2:1014983/FULLTEXT02
3. https://www.educba.com/what-is-orm/