Object-Relational Mapping (ORM)

Agenda

1 Persistence

2 Object-Relational Mismatch

3 Object-Relational Mapping

4 ORM Framework

5 Advantages of ORM

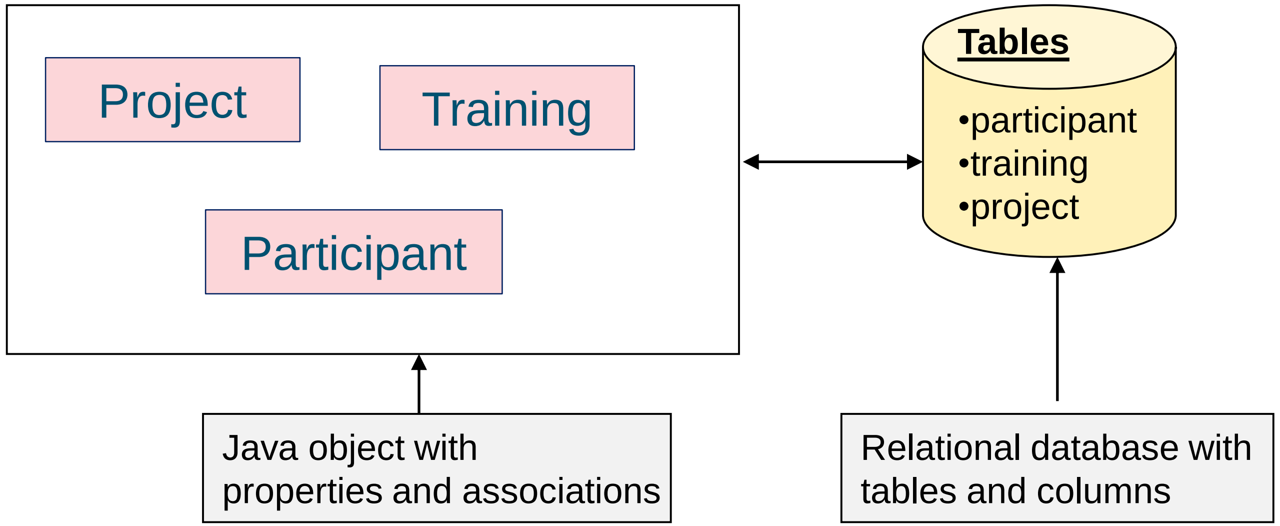
6 Summary

Persistence

- Persistence is the availability of the object/data even after the process that created it, is terminated
- Persistence can be achieved through:
  - **Data Base Management System (DBMS)**
    - Data is stored in the database and can be retrieved efficiently using Structured Query Language (SQL)
    - DBMS provides features like Concurrency, Data Sharing, Data Integrity, and Data Security
  - **Serialization**
    - It is a mechanism for storing objects to disk in such a way (as files) that they can be retrieved back again later

Object / Relational Mismatch

- Many object-oriented applications may have to implement both object model and relational model of some business entities
- When working with such applications, there are mismatch between object model and relational database


Project
Training
Participant
Tables
•participant
•training
•project
Java object with properties and associations
Relational database with tables and columns

Object / Relational Mismatch

Relational database with tables and columns

```
Create Table Training(
TrainingId Number(5) PrimaryKey,
TrainingName varchar2(15),
Rating number(3)
)
Create Table Project(
ProjId Number(5) PrimaryKey,
ProjName varchar2(15),
duration number(9,2)
)
Create table Participant (
Pid number(5) Primary Key,
Name varchar2(20),
Address varchar2(20),
trainingId number(5) references
Training(TrainingId),
projId number(5) references Project(ProjId) )
```

Java object with properties and associations

```
public class Training {
private int trainingId;
private String trainingName;
private int completionRating;
}
public class Project {
private int projectId;
private String projectName;
private int duration;
}
public class Participant {
private int pid;
private String name;
private String address;
private Set<Training> trainings;
private Set<Project> projects;
}
```

Object / Relational Mismatch

- Problems faced when a relational model is used to store objects:
  - Problem of granularity
  - Problem of Identity
  - Problems related to associations
  - Problem of subtypes
  - Problem of data navigation

Problem of granularity

- Let us take a table called USER1 with the given structure

```
Create table USER1 (
USERNAME1 VARCHAR(15) NOT NULL PRIMARY KEY,
NAME1 VARCHAR(50) NOT NULL,
ADDRESS1 VARCHAR(100)                    )
```

- For implementation we need to create a class to represent the USER1 table
- Here *Address* attribute is represented as a single field in reality it needs more information like city, state, pin etc.
- How are we going to represent it? As *separate class or as fields in same class*?
- In database as an *individual attributes or as a separate table*?
- This is *problem of granularity*

Problem of Identity

- Object identity is not same as database identity.
- *Object identity* can be established by checking for the *references of 2 objects* or using equals method for checking for equality.
- *Database identity* is established by having a *primary key* which identifies each record distinctly.
- And in databases we can have tables without primary key as databases support tables without primary key.

Problems related to associations and Subtypes

- In object model associations are represented as unidirectional
- For Bi-directional it needs to be defined twice
- In Relational model associations are relationship established with associating a foreign key along with the primary key of another table
- The concept of inheritance and polymorphism is not directly part of relational database

Problems of data navigation

- In object model information access is through *method calls*
- accessing data from database is not same, there will be more trips to the database for a single query.
- In relational model *data* can be *accessed in large chunks*.

ORM

- There are mismatch between the object model and relational database
- Object relational mapping (ORM) *is a technique of mapping the data representation from an object model to a relational data model*
- There are alternatives for performing object relational mapping like
  - Java Data Base Connectivity (JDBC)
  - Serialization
  - Enterprise Java Beans (EJB)
  - Object Oriented Data Base Management System (OODBMS)
- Best Approach are through ORM Frameworks

ORM Frameworks

- There are several ORM Frameworks available
- Some top frameworks are TopLink, castor, Hibernate, Spring DAO etc.
- *Hibernate Framework provides APIs for performing basic CRUD* (Create Read Update Delete) operations
- Framework operates *on objects of persistent classes* that are mapped to the Tables in the Database
- Mappings are done through metadata for classes (POJO) to relational database tables

Advantages of ORM Framework

- Productivity
- Maintainability
- Performance
- Vendor Independence

Summary

In this module you were able to

- Define Persistence
- Identify the problems with Object/Relational mismatch
- Describe Alternatives for persistence management
- Define Object / Relational Mapping (ORM) in detail

Thank you