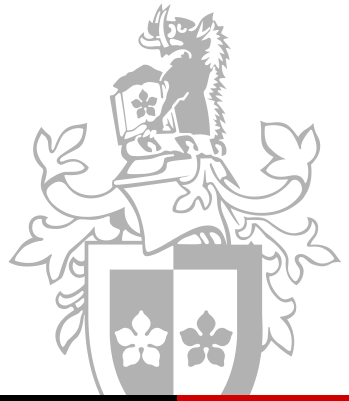
COS30041 Create Secure and Scalable Software

Lecture 03a Object-Relational Mapping (ORM)



SWIN BUR NE

SWINBURNE UNIVERSITY OF TECHNOLOGY

Learning Objectives

- After studying the lecture material, you will be able to
  - Understand and describe ideas about Object Relational Mapping (ORM)
  - Discuss issues related to ORM

Pre-requisites

- Some concepts in Object Oriented Programming
- Some ideas about Relational Database

Outline

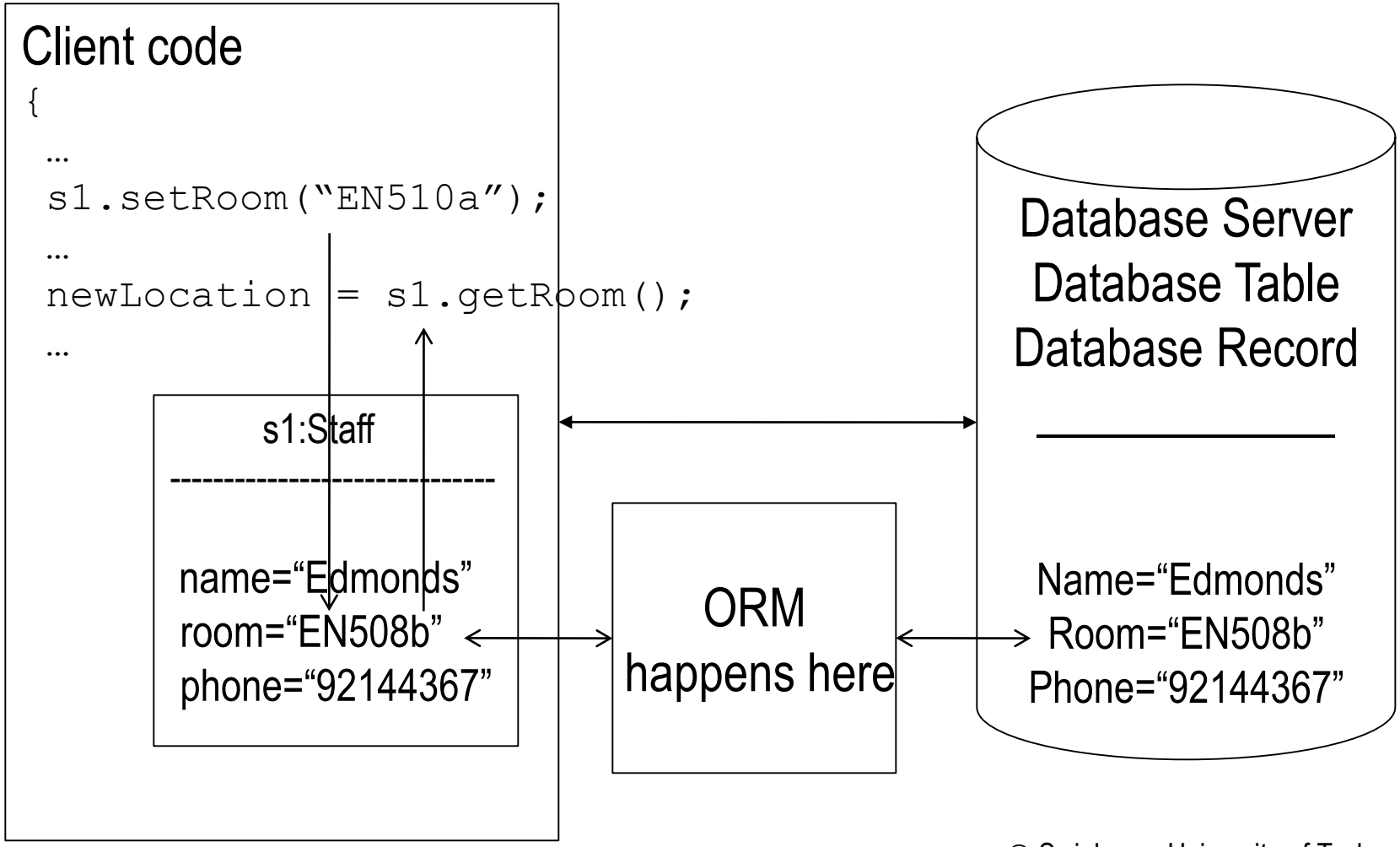
- Object-relational Mapping, ORM
- Object-relational Impedance Mismatch

Some Background

- Recently
  - We have JDBC programming paradigm that can access different databases via embedded SQL
- Problem: Writing SQL statements is too much of a trouble. Can we make it simpler?
  - Code – normally written in OO languages
  - Database – normally is relational

Object-relational Mapping, ORM

- A mapping between objects (in OOP) and database records



Entity Class [Java]

- The class that maps to a database table via the ORM
- A lightweight persistence domain object
  - It represents a table in a relational database
  - Each **entity instance** corresponds to a row in the table
- Example
  - BankAccount
  - Customer

Why Entity Object / Class?

- Is handy to treat data as objects
- Can associate simple methods with objects
- Can store the data in memory for performance

- Need an overview of how it works

- An entity class ←→ entity manager [ORM occurs here] ←→ database server

Issues in ORM

- Some ORM tools do not perform well during bulk deletion of data
  - Better do it using stored procedures – but not portable
  - Probably JDBC programs will do a better job
- Heavy reliance on ORM software contributes to a major factor in producing poor design
  - Experience show

Object-Relational Impedance Mismatch

Objects in OOP

- Encapsulation
  - Private representation of objects
- Accessibility
  - private, public, protected
- Interface, Inheritance and Polymorphism

Records in DB

- No Encapsulation
  - Public uses of data
- Accessibility
  - Data value is for all
- Not supported (?...)
  - Some ORM technologies do provide some supports but there are still issues around

Current approach: Only map relational tables to associations found during Object-Oriented Analysis and Design

ORM – Resolving Inheritance

- A table per concrete class
  - Each concrete class has to store all attributes from its superclass(es)
  - Problem: Tables not normalized
- A table per subclass
  - Each subclass only stores its own attributes
  - Problem: Join operation is often required

Others are possible … [but your investigation is needed]

Other possible solutions?

- Use ORM to manage the objects and transparently store them in DB
  - The approach used in Java EE 5 or later
- Use an Object Oriented DB to store objects (no mapping)
  - Problem: Need to abandon the existing RDBMS!

References

- ORM
  - en.wikipedia.org/wiki/Object-relational_mapping
- Object-relational Impedance Mismatch
  - en.wikipedia.org/wiki/Object-relational_impedance_mismatch
- Object Oriented Databases
  - en.wikipedia.org/wiki/Object_database