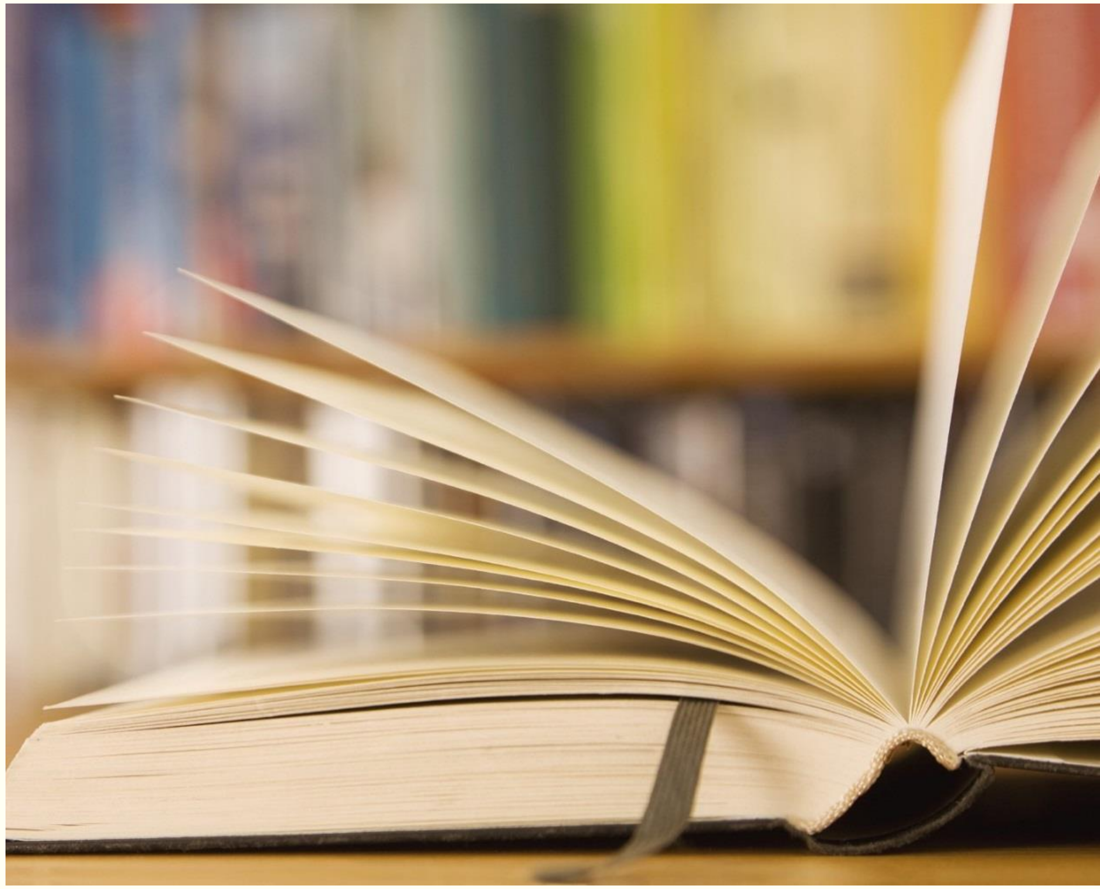
ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD

Version 2.1 ©2021



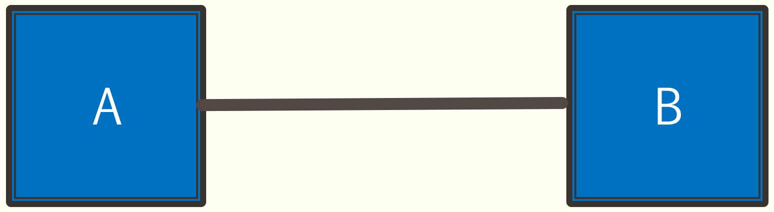
ENTITIES AND RELATIONSHIPS

OO Programming Properties

- Encapsulation
- Delegation
- Propagation
- To achieve delegation and propagation we need relations between objects.
- To establish data structures, we need relations.
- To achieve modular design and real-world simulations we need relations.
- How do we represent relations in DB?
- How do we represent relations in OO?

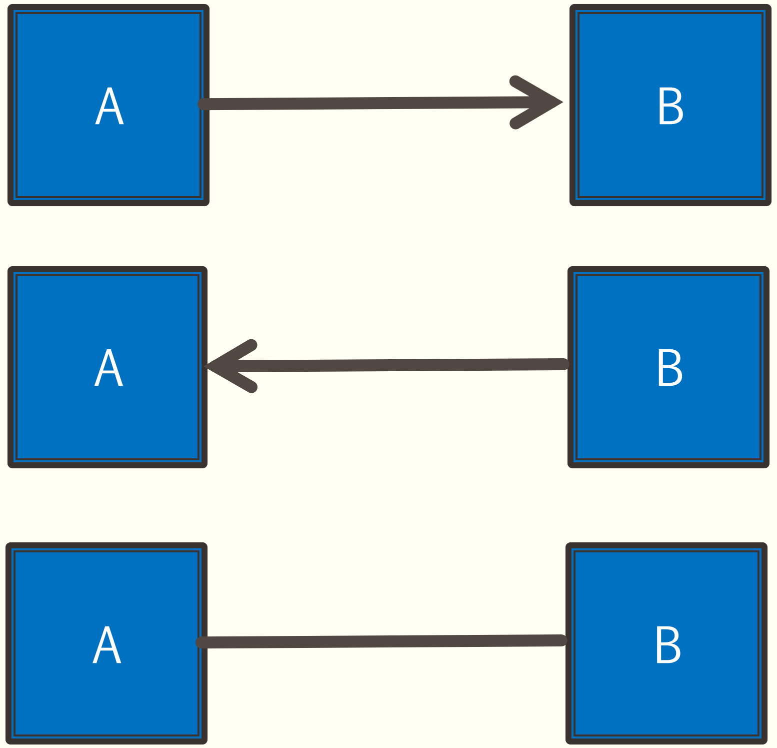
Relationships

- How do we represent this relationship in OO?
- How do we represent this relationship in RDBMS?
  - Can we have multiple ways to represent it?
- @OneToOne
- @JoinColumn
- @JoinTable



Directionality

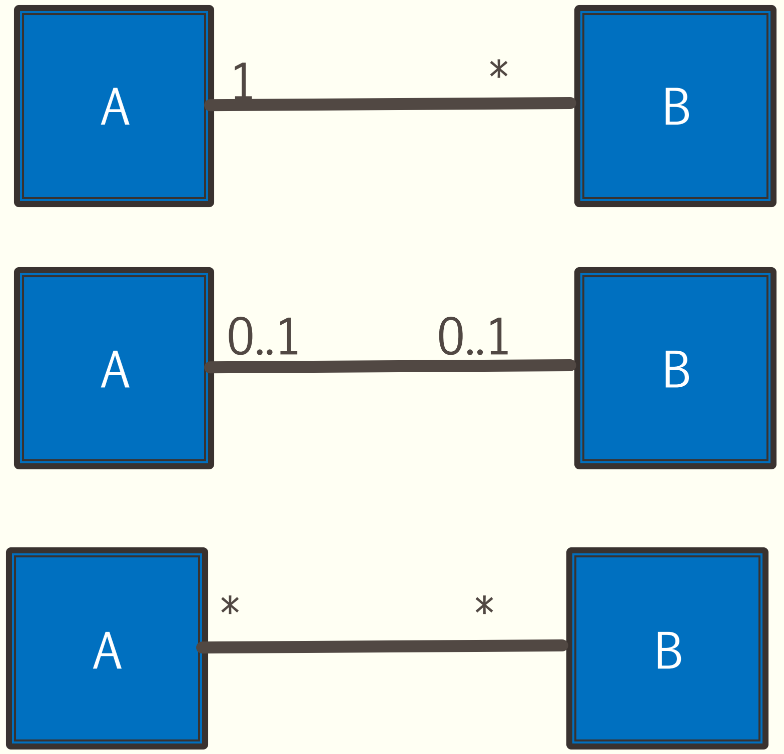
- How are these different?
- In OO?
- In RDBMS?
- mappedBy



Cardinality & Ordinality

- @OneToMany
- @ManyToOne
- @ManyToMany

- What is the default for each one?



Fetching

- What are your options when fetching objects in relationships?
- What is the advantage of each?
- What is the default?
- @OneToOne(fetch=FetchType.EAGER)
- @OneToOne(fetch=FetchType.LAZY)

Cascading

- By default EnityManager operates on the passed entity (persist, remove, find).
- Relations may result in the need to apply the operation on related entities.
- @OneToMany(cascade= CascadeType.PERSIST)
- Types of cascading
  - PERSIST
  - REFRESH
  - REMOVE
  - MERGE
  - DETACH
- Cascades are unidirectional.

- Default em cascading
- Persistent Unit

```
<entity-mappings>
  <persistence-unit-metadata>
    <persistence-unit-defaults>
      <cascade-persist/>
    </persistence-unit-defaults>
  </persistence-unit-metadata>
  ...
<entity-mappings>
```

Extra

- Association not typed.
  - TargetEntity=Student.class
- What if the Entity Table is not normalized?
  - Example, Student table has student address information.
  - @Embedded private Address address;
  - @Embeddable @Access(AccessType.FIELD) public class Address { … }
  - @Embedded
    - @AttributeOverrides({@AttributeOverride(name="zip", column=@Column(name="zipCode"})

MORE ON ENTITIES

Possible Attribute Access

- Field
- Method
- Mixed
  - @Access(AccessType.FIELD)
  - @Access(AccessType.PROPERTY)
- Anything else?
- Attributes not to persist
  - @Transient
  - transient private int age;
- Get attribute later
  - @Basic(fetch=FetchType.LAZY)

Attribute Types

- Enumeration
    - Do nothing
    - @Enumerated(EnumType.STRING)
- Temporal
    - @Temporal(TemporalType.DATE)
        - Three Types
        - java.sql.Date
        - java.sql.Time
        - java.sql.Timestamp
        - java.util.Date
        - Java.util.Calendar

```
@Entity
Student {

    Private int id;
    private String name;
    private float gpa;
    @Embeded
    private Address address;

}

@Embedable
Address {

    private int bldNum;
    private int roomNum;

}
```

- Student Table:

id	INT PK
name	VARCHAR(30)
gpa	FLOAT
bldNum	INT
roomNum	INT

Main Point

- We can represent the OO relationships between entities in JPA. JPA provides a lot of flexibility when mapping OO relationships to DB. This is helpful since we may be working with legacy or denormalized DB systems, but we do not wish to compromise our OO design.
- There are no compromises in nature.

INHERITANCE