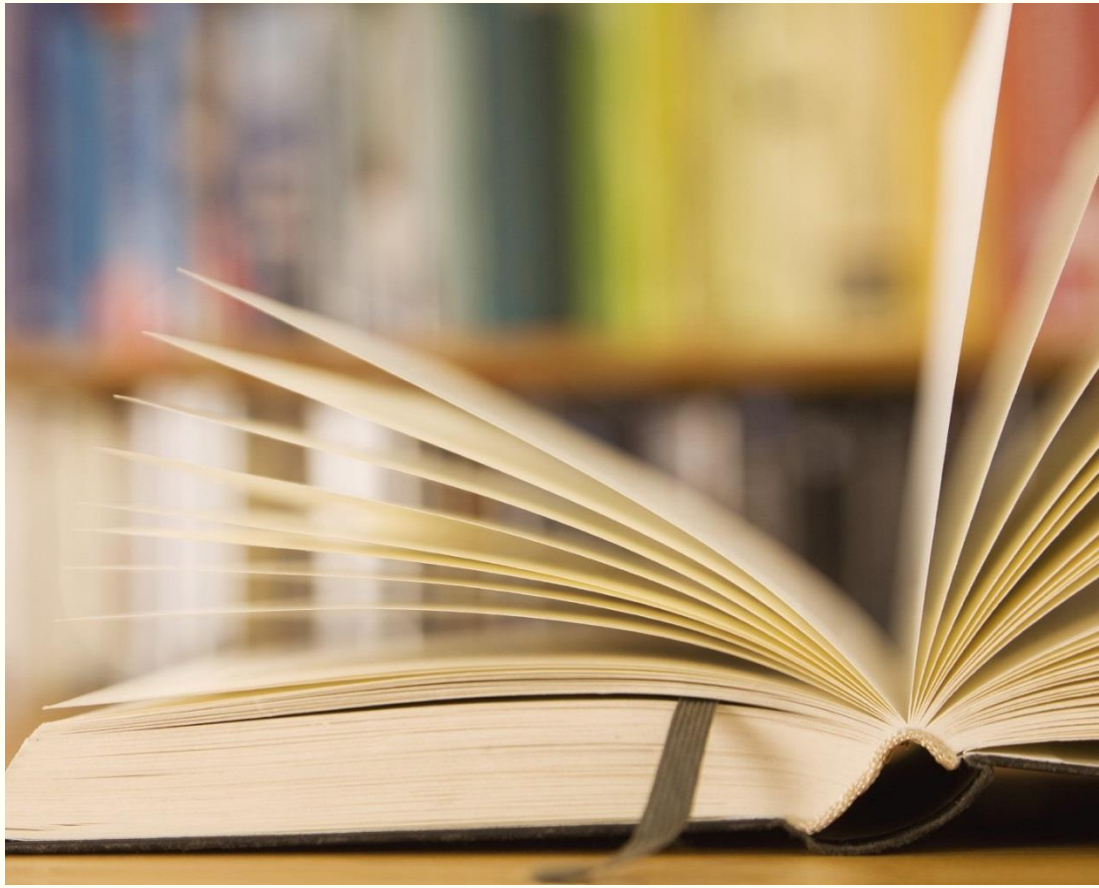
ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD



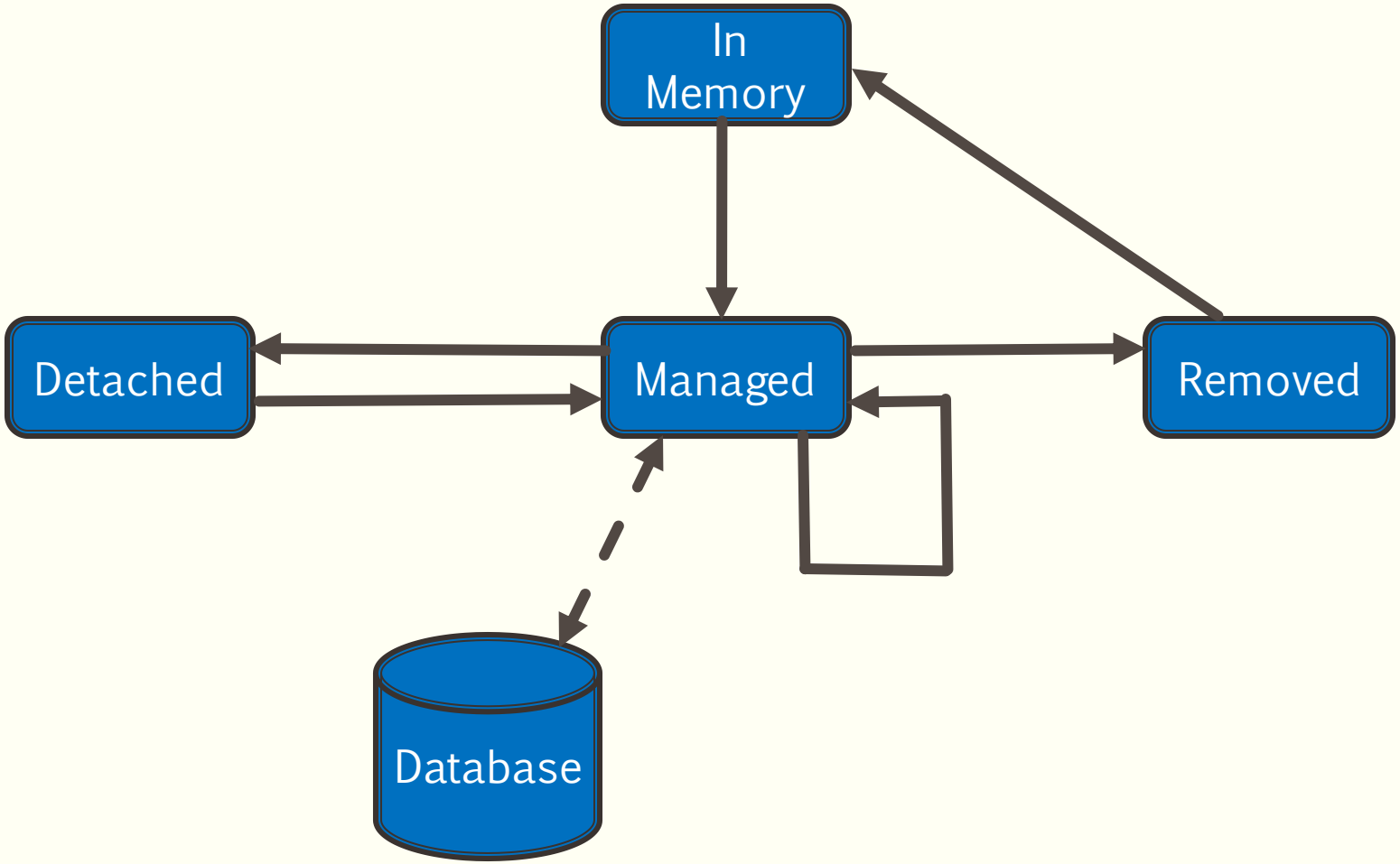
LESSON 05 ENTITY LIFECYCLE AND CALLBACKS

Problems When Using EM & Entity

- Eager loading results in unnecessary memory consumption.
- Lazy loading generates new types of problems.
  - DB connection needs to be available when loading in the future.
- IoC is great but it also results in loss of visibility. What if we need some level of visibility?
  - Perform some task right before deleting an entity.
  - Perform some task right after persisting an entity.

ENTITY LIFECYCLE

Entity Lifecycle



Entity State in EntityManager

- In Memory
  - POJO, not an Entity yet (or no longer an entity).
- Managed
  - Changes are reflected on DB.
  - Entity in persistence context.
- Detached
  - Changes not reflected on DB.
  - Cannot perform lazy operations.
- Removed
  - Entity no longer in DB.

Detachment Situations

- Committed transaction.
  - All entities in the TX are detached.
- Persistence context closes.
  - All entities in context detach.
- EntityManager clear() is invoked.
  - Clears the entities from the context.
- EntityManager detach() is invoked.
  - Entity passed to the method is detached.
- Transaction rollback.
  - All entities in TX are detached.
- Entity serialized.
  - The serialized form of the entity is detached.

The Lazy Loading Problem

- Detached entity with lazy loading.
  - Behavior not defined in Specification.
- In some cases, it cannot be resolved (Serializable). Check if entity is loaded using isLoaded() in Utility class (not covered).

MERGING

Merge

- EntityManager.merge(detachedEntity);
- Takes detached entity as input.
- Returns reference to the entity in persistence context.
- If the merge is called within a TX changes will be reflected in the context.
- Entity "student" may be a new one or already exists in the context.
- If a Student entity exists with the ID in context, it get updated.

- Will not save the change to GPA.

```
public void fixStudent(Student std) {
    em.merge(std);
    std.setGpa(3.3f);
}
```

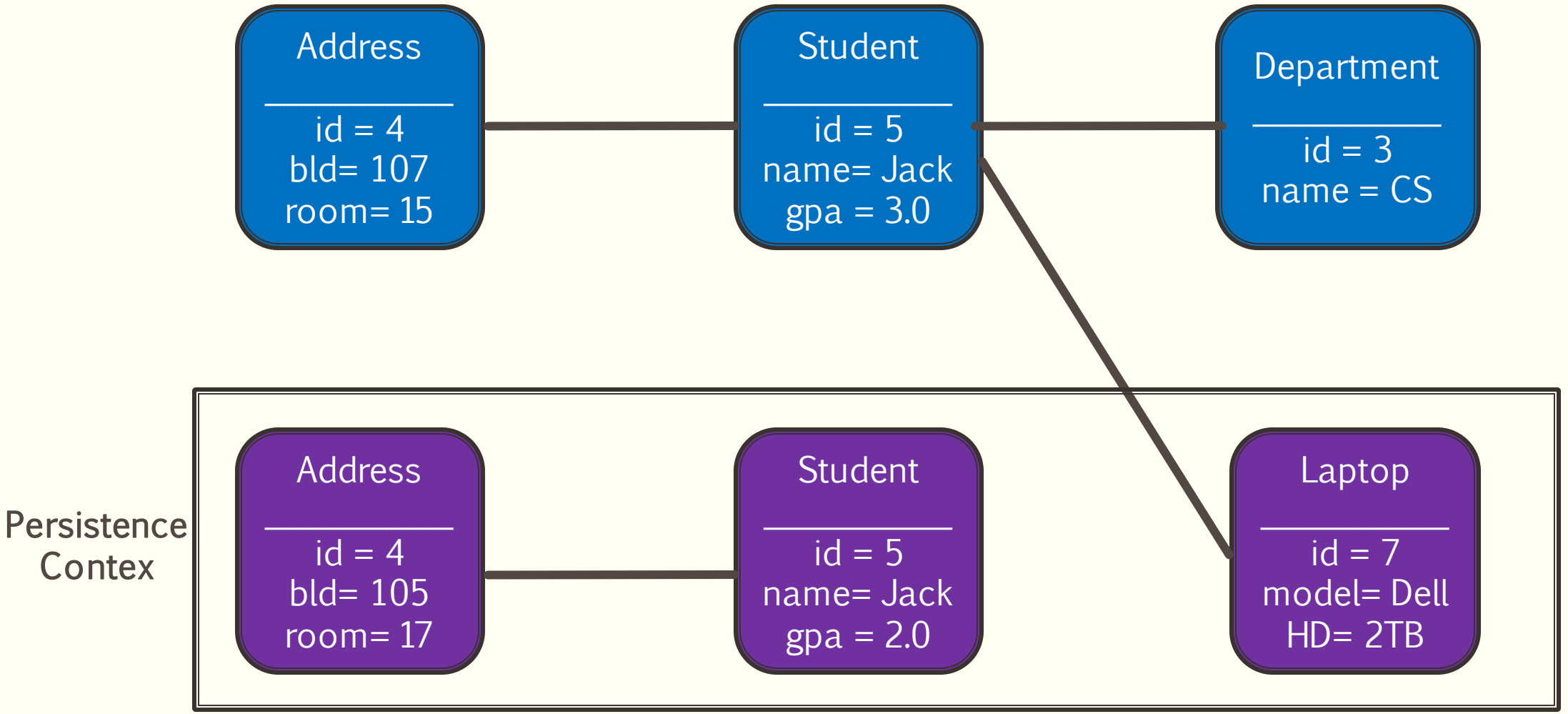
- Will save the change to GPA.

```
public void fixStudent(Student std) {
    Student student= em.merge(std);
    studet.setGpa(3.3f);
}
```

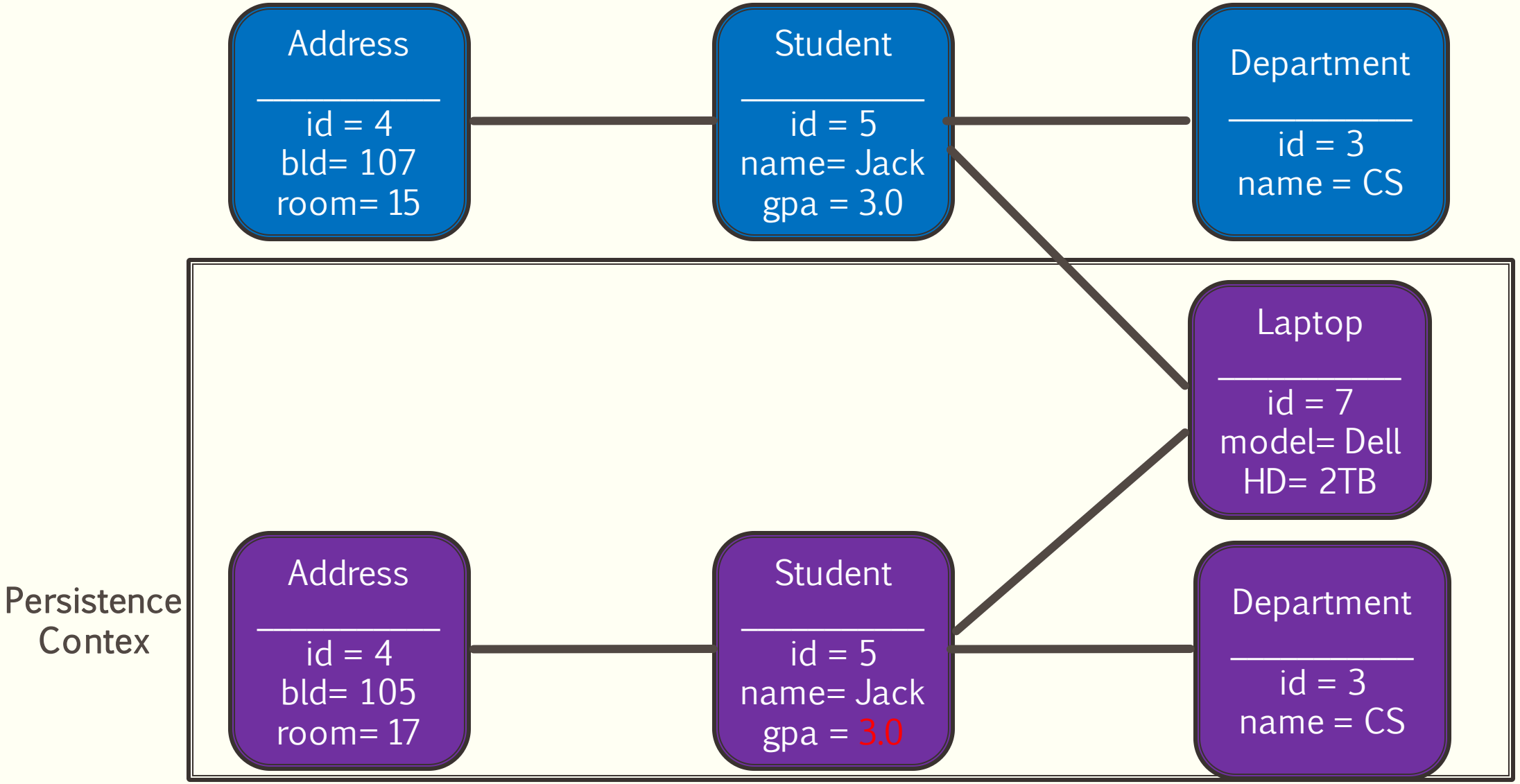
More on Merge()

- When invoking merge on a new instance it acts like persist.
- The behavior of related entities without an ID in a merge is not defined.
  - Either merge them as well or throw an exception.
  - Recommended cascading for MERGE to have a defined behavior.

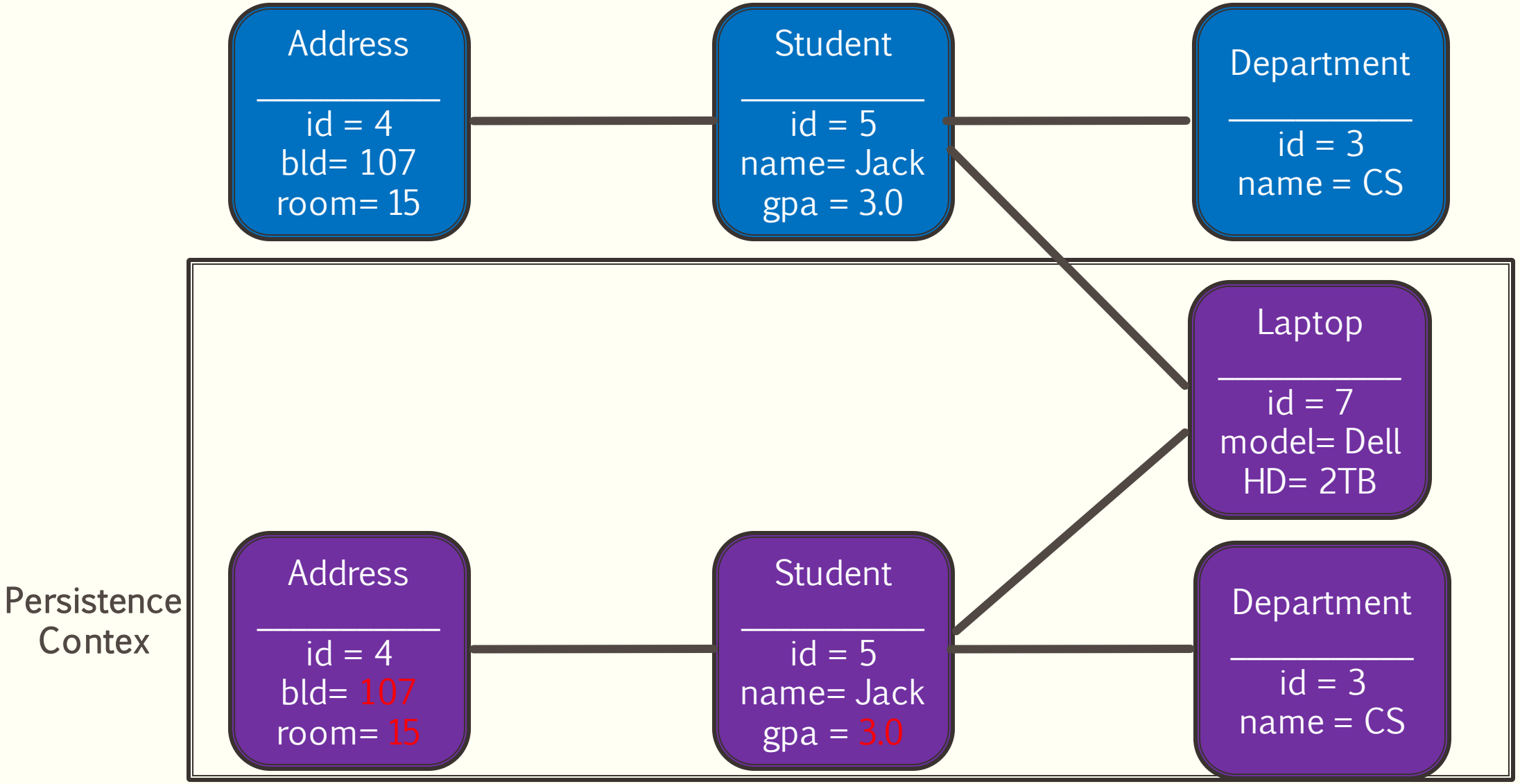
Example


Address
id = 4
bld= 107
room= 15
Student
id = 5
name= Jack
gpa = 3.0
Department
id = 3
name = CS
Persistence Contex
Address
id = 4
bld= 105
room= 17
Student
id = 5
name= Jack
gpa = 2.0
Laptop
id = 7
model= Dell
HD= 2TB

Example – merge(student) - Default No Cascade



Example – merge(student) - With Cascade



Main Point

- The entity goes through different states with respect to persistence context. When an entity is detached changes on it are not reflected. Only managed entities are kept in synch with database. Detached entities may be attached back using merge. When merging it is best practice to merge the root of the relationship graph.