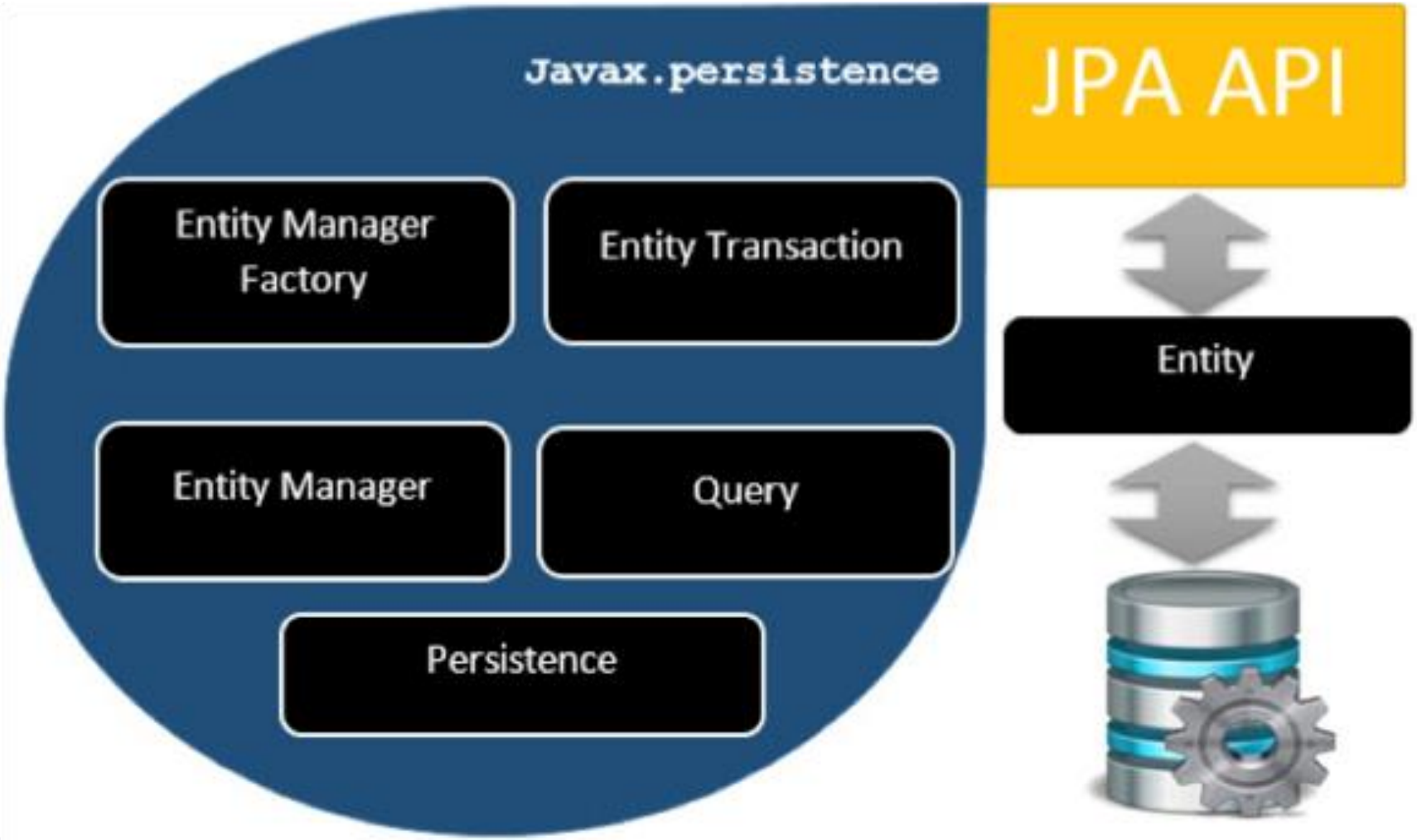
JPA – werking

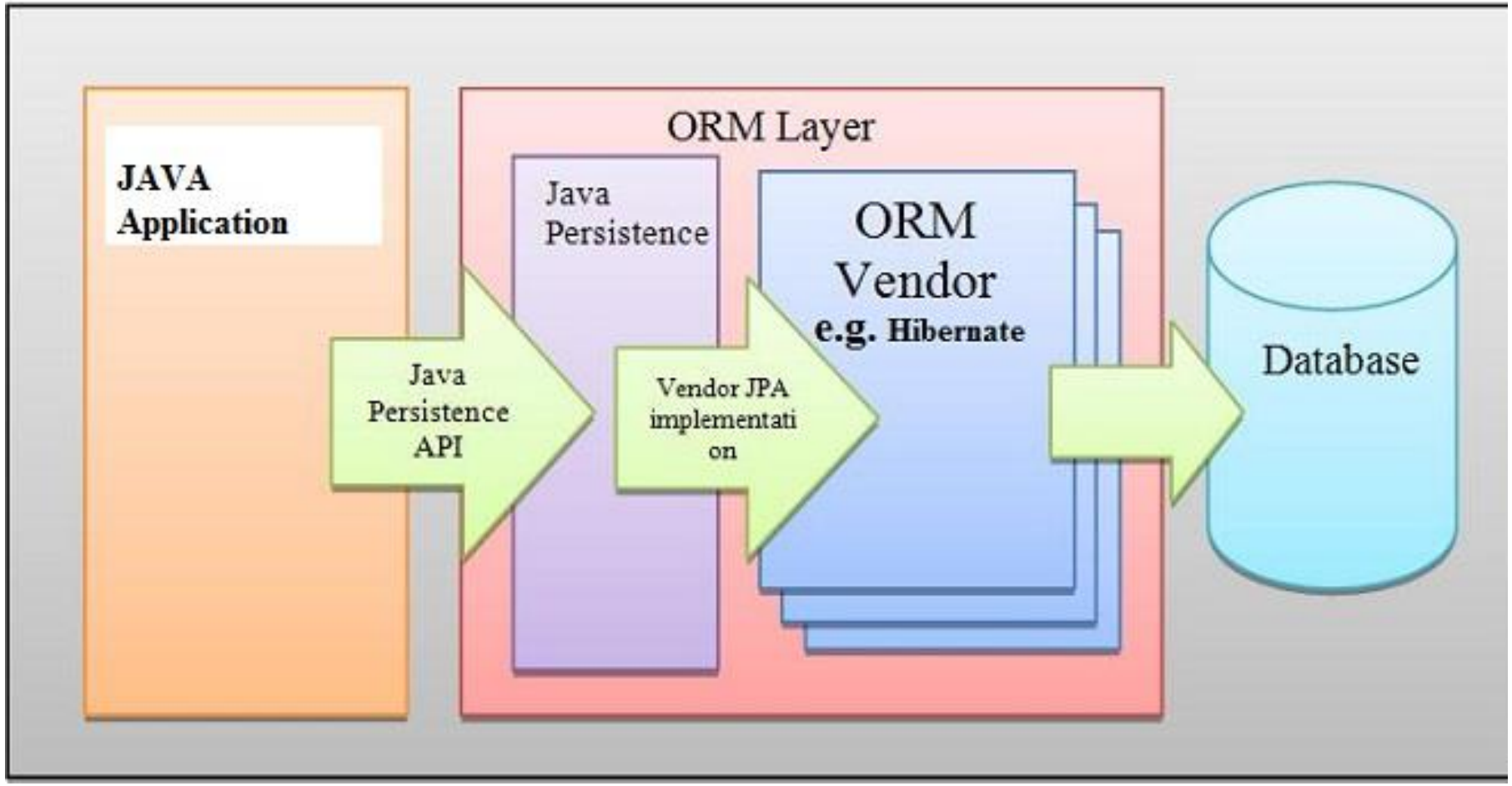
Veerle Ongenae

Configuratie



https://www.tutorialspoint.com/jpa/jpa_architecture.htm

JPA



bron: http://thecafetechno.com/tutorials/hibernate/hibernate-and-java-persistence-apijpa/

EntityManager

- Beheert entities in een “persistence context”
- Persistence context
  - Container

```
@PersistenceContext
EntityManager em;
```

  - Application

```
@PersistenceUnit
EntityManagerFactory emf;
```

```
EntityManager em = emf.createEntityManager();
```

EntityManager

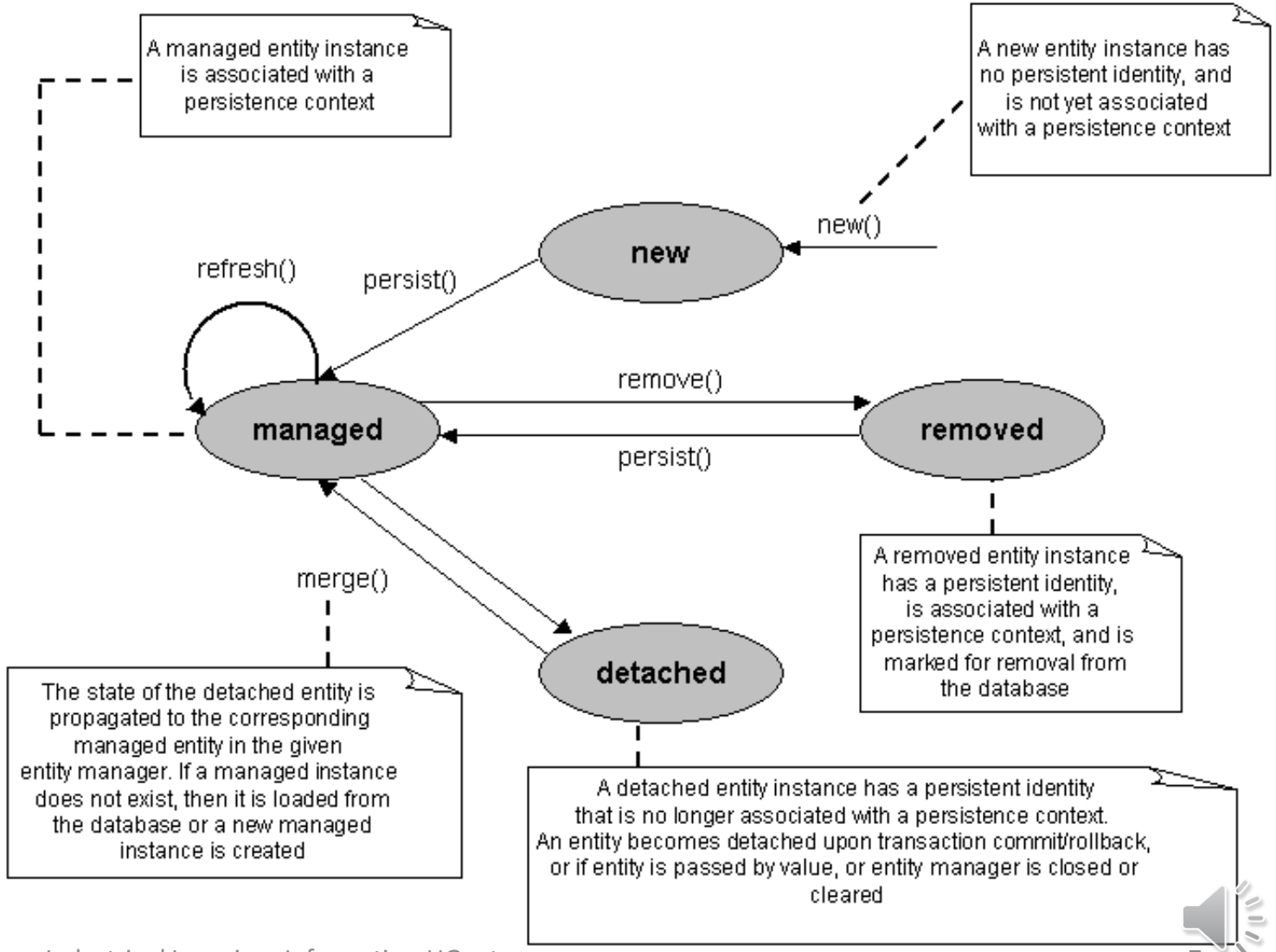
```
String[] clubs = {"EIKENLO", … , "LANDEGEM BC FV"};

EntityManagerFactory entityManagerFactory =
     Persistence.createEntityManagerFactory("BadmintonJPAPU");
EntityManager entityManager = entityManagerFactory.createEntityManager();
entityManager.getTransaction().begin();
for (String club : clubs) {
     Sportclub sportclub = new Sportclub();
     sportclub.setNaam(club);
     entityManager.persist(sportclub);
}
entityManager.getTransaction().commit();
entityManager.close();
```

Werken met objecten

- Toestanden van een object
- Objecten persistent maken
- Objecten opvragen
- Objecten verwijderen
- Objecten wijzigen

Levensloop object



bron: http://java.boot.by/scbcd5-guide/ch06.html

Werken met objecten

- Toestanden van een object
- Objecten persistent maken
- Objecten opvragen
- Objecten verwijderen
- Objecten wijzigen

Objecten persistent maken

- Methode `persist` van `EntityManager`
- Object toevoegen aan databank
  - Persistent maken
  - Indien transactie uitgevoerd
- Indien unieke id niet gegenereerd
  - Toekennen voor persistent maken

```
entityManager.persist(sportclub);
```

- Aangemaakt met new
- Niet geassocieerd met persistence context
- Niet beheerd door een entitymanager
- Unieke id niet ingevuld

- Beheerd door een entitymanager
- Wordt bewaard in databank (na uitvoeren transactie)
- Identifier ingevuld
- Bewaard in persistence context
- Veranderingen → databank (door Hibernate, …)

Cascading style - persist

- Persistent maken object
  - Wat met geassocieerde objecten?
- Bv.

```
@Entity
public class Person {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int id;
  private String name;
  @OneToMany(mappedBy = "person", cascade = CascadeType.PERSIST)
  private List<Address> addresses;
}
```

```
@Entity
public class Address {
 @Id
 @GeneratedValue(strategy=…)
 private int id;
 private String street;
 private int houseNumber;
 private String city;
 private int zipCode;
 @ManyToOne
 private Person person;
}
```

Werken met objecten

- Toestanden van een object
- Objecten persistent maken
- Objecten opvragen
- Objecten verwijderen
- Objecten wijzigen

Objecten opvragen

- Op basis van identifier
- Zonder identifier
  - Java Persistence Query Language
  - SQL

Opvragen op basis van identifier

```
public Sportclub getSportclub(Long id) {
    EntityManager em = emf.createEntityManager();
    Sportclub sportclubOpId = em.find(Sportclub.class, id);
    em.close();
    return sportclubOpId;
}
```

- Methode `find` van `EntityManager`

```
T find(Class<T> entityClass, Object primaryKey)
```

  - `null` indien onbestaand persistent object

- Meermaals opvragen object met zelfde identifier → verschillende referenties naar zelfde object (in cache)

Lazy fetching

UNIVERSITEIT GENT

- Opvragen object
  - Geassocieerde objecten niet mee opgevraagd
  - Opgevraagd indien nodig
    - ➢ Enkel als sessie nog open is
- Onmiddellijk mee opvragen

```
@ManyToOne(fetch=FetchType.EAGER)
```

```
@Entity
public class Address {
 @Id
 @GeneratedValue(strategy=…)
 private int id;
 private String street;
 private int houseNumber;
 private String city;
 private int zipCode;
 @ManyToOne(fetch = FetchType.LAZY)
 private Person person;
}
```


Voorbeeld lazy

```
public List<Lid> getLeden(Long id) {
    EntityManager em = emf.createEntityManager();
    Sportclub sportclubOpId = em.find(Sportclub.class, id);

    // lazy, ophalen en kopiëren
    List<Lid> leden = new ArrayList<>(sportclubOpId.getLeden());
    em.close();
    return leden;
}
```

Werken met objecten

- Toestanden van een object
- Objecten persistent maken
- Objecten opvragen
- Objecten verwijderen
- Objecten wijzigen

Objecten verwijderen

```
EntityManager em = … ;
Employee employee = em.find(Employee.class, 1);
em.getTransaction().begin();
em.remove(employee);
em.getTransaction().commit();
```

- Methode `remove` van `EntityManager`
- Ook voor geassocieerde eniteiten?
  - Instellen met `cascade=CascadeType.REMOVE` op associatie (@)

- Beheerd door een entitymanager
- Wordt bewaard in databank (na uitvoeren transactie)
- Identifier ingevuld
- Bewaard in persistence context
- Veranderingen → databank (door Hibernate, …)

- Gemarkeerd om verwijderd te worden
- Pas verwijderd bij uitvoeren transactie

Werken met objecten

- Toestanden van een object
- Objecten persistent maken
- Objecten opvragen
- Objecten verwijderen
- Objecten wijzigen

Objecten wijzigen

UNIVERSITEIT GENT

- Persistente objecten
  - Aangemaakt via `persist`
  - Opgehaald via `find` of een zoekopdracht
- Wijzigen tijdens een transactie
  - Methode `flush`
    - Wijzigingen doorvoeren naar databank
    - Wordt ook impliciet opgeroepen bij `commit` van de transactie

```
EntityManager em = … ;
Employee employee = em.find(Employee.class, 1);
em.getTransaction().begin();
employee.setNickname("Joe the Plumber");
em.getTransaction().commit();
```


Merge

UNIVERSITEIT GENT

- Methode `merge` van `EntityManager`
  - Het meegegeven object in de huidige persistentiecontext kopiëren
  - Resultaat = object uit persistentiecontext
- Enkel nodig bij doorgeven objecten tussen contexten

```
EntityManager em = createEntityManager();
Employee detached = em.find(Employee.class, id);
em.close();
...
em = createEntityManager();
em.getTransaction().begin();
Employee managed = em.merge(detached);
em.getTransaction().commit();
```


Objecten wijzigen

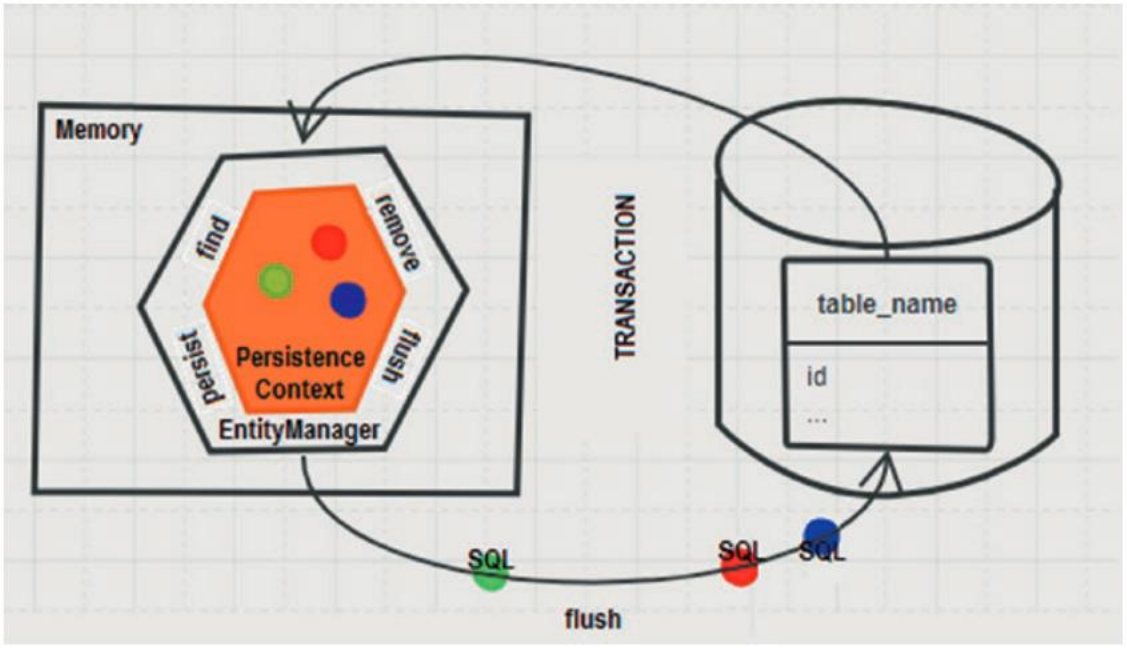
- Detached objecten
  - Objecten aangemaakt door andere/niet meer bestaande EntityManager
  - Object wijzigen
  - Wijzigingen bewaren met nieuwe EntityManager
    - ➢ Methode `merge`
      - Na methode is het object persistent
      - Eventueel wijzigingen kopiëren in een bestaand object met zelfde Id
- Geassocieerde entiteiten ook?
  - Instellen op associatie: `cascade=CascadeType.MERGE`

- Persistent object
- Niet gekoppeld aan persistence-context omdat entitymanager gesloten is
- Niet beheerd door een entitymanager

- Beheerd door een entitymanager
- Wordt bewaard in databank (na uitvoeren transactie)
- Unieke id ingevuld
- Bewaard in persistence context
- Veranderingen → databank (door Hibernate, …)

```
public void addThesissen(List<Thesis> thesissen) {
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    for (Thesis thesis : thesissen) {
        Student student = em.merge(thesis.getStudent());
        thesis.setStudent(student);
        student.setThesis(thesis);
        em.persist(thesis);
    }
    em.getTransaction().commit();
    em.close();
}
```

Persistence context

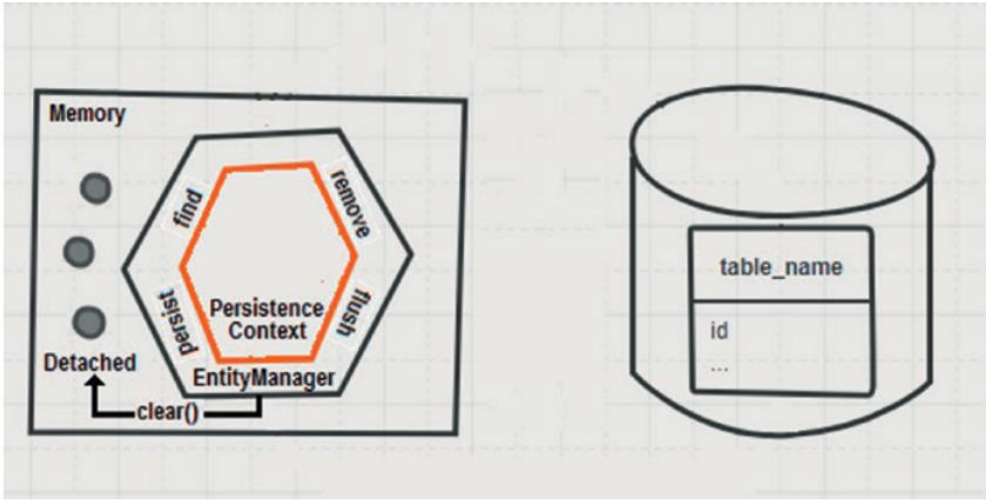


Bron: https://link.springer.com/content/pdf/bbm:978-1-4842-5626-8/1.pdf

- Deel van
  - JPA-EntityManager
- Nuttig voor
  - Automatisch checken dirty objecten
  - Dirty objecten
    - ➢ Gewijzigd
    - ➢ Wijzigingen nog niet doorgevoerd op database
    - ➢ Doorvoeren wijzigingen zo lang mogelijk uitgesteld, bv. commit transactie
  - First level cache
    - ➢ Entiteiten cachen
    - ➢ Meermaals opvragen → referentie naar zelfde object
- Bestaat zolang de EntityManager geopend is

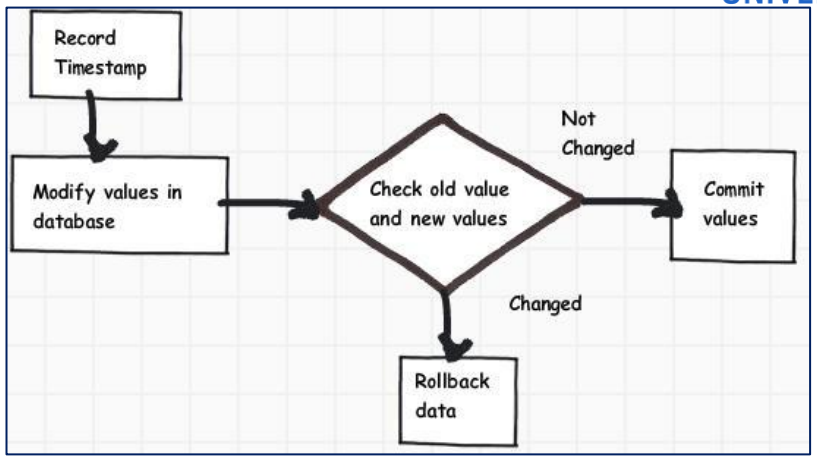
Object verwijderen uit persistence context

- EntityManager sluiten
- Methode `detach` van EntityManager
  - Verwijderd uit session cache
    - Aanpassingen niet meer doorgevoerd in databank
    - Toestand = detached
  - Kan nog gebruikt worden in programma
  - Ook op geassocieerde objecten?
    - `cascade="CascadeType.DETACH"` (annotaties)
  - Alle persistenten objecten verwijderen uit session cache (= detached maken)
    - Methode `clear`
    - Nog niet “geflushte” opdrachten worden niet uitgevoerd op de database

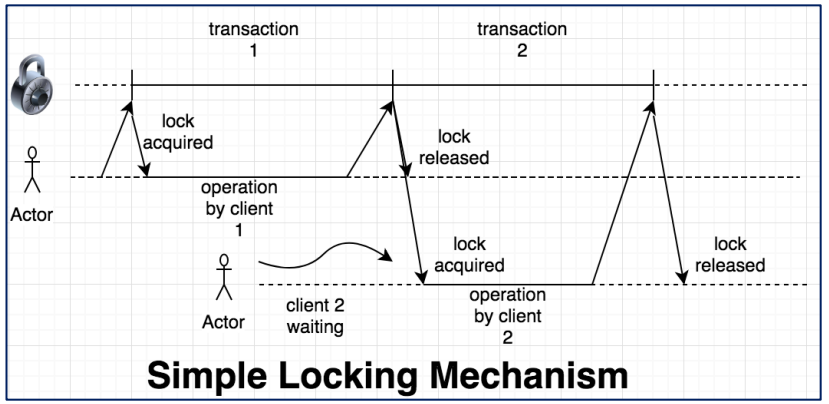


Entity Locking/Concurrency

- Gelijktijdige toegang?
  - Dataintegriteit bewaren?
- Standaard
  - Optimistic locking
    - ➢ Voor aanpassen kijken of data niet veranderd is a.d.h.v. versiekolom
- Pessimistic locking
  - Transactie legt lock op data zolang de transactie loopt



Bron: https://www.c-sharpcorner.com/UploadFile/shivprasadk/3-ways-of-doing-optimistic-locking-in-net/



Bron: https://tech.urbancompany.com/pessimistic-locking-for-a-distributed-system-part-1-cc85c755c357