Chương 6: JPA

Version 2.x: Java Persistence API

Version 3.x: Jakarta Persistence API

Trường Đại học Công nghệ Sài gòn
Khoa Công nghệ Thông tin

TỔNG QUAN

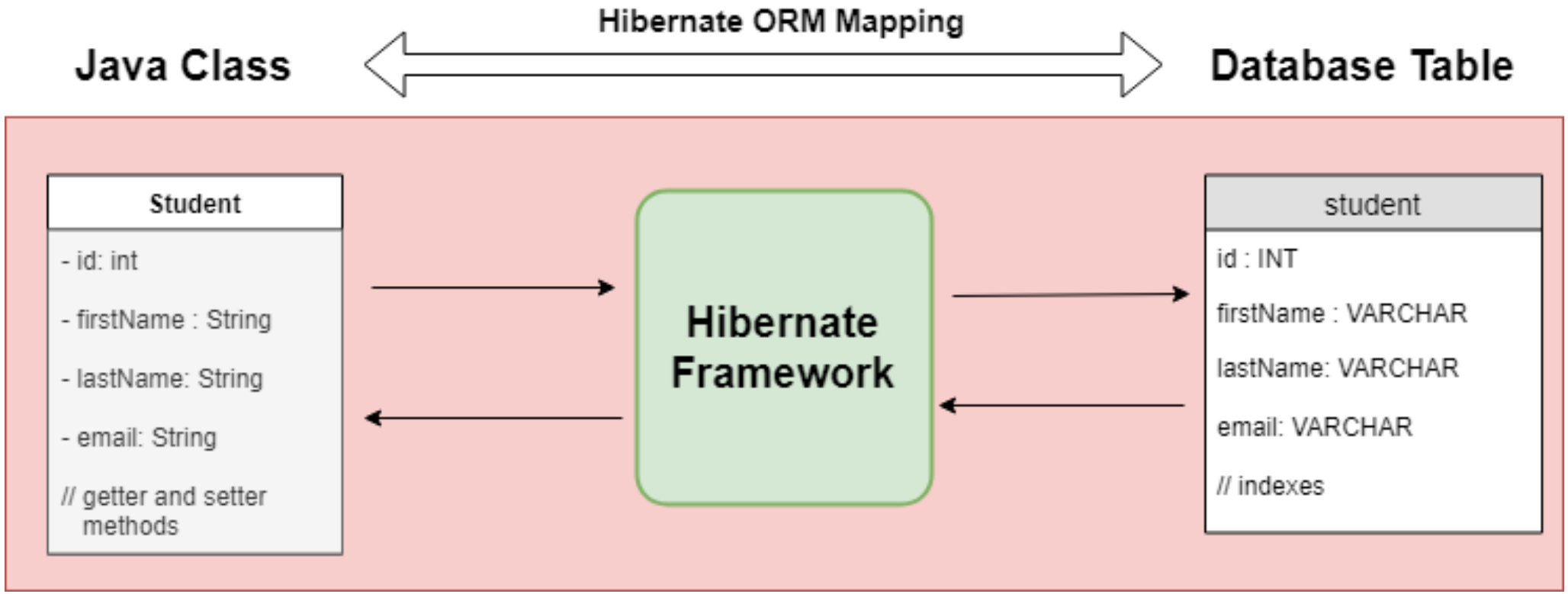
- Phần trước chúng ta đã học:
  - JDBC
  - Data Access Object (DAO) và Data transfer Object (DTO)
- Trong JDBC chúng ta đã “hard code” SQL trong app
- Sử dụng Data source/Connection
- Sử dụng DAO/DTO
- Nhưng điều này chỉ “hide” hiện thực từ tầng business logic và tầng gui, vẫn phải hiện thực DAO dùng JDBC

Các vấn đề không được giải quyết

- Chúng ta phải hiểu rất chi tiết (connection, statement, resulkset…)
- Các mối quan hệ, join…
- Object<-> database không khớp
- J2EE giúp giải quyết vấn đề trên với EJB (Entity Enterprise JavaBean)
- Hoặc đơn giản hơn khi dùng J2SE + bao gồm:
  - Các công cụ ORM (Object Relation Mapping): JDO, Ecliplink, Hibernate, Open JPA, TopLink….

ORM (Object-relational mapping)

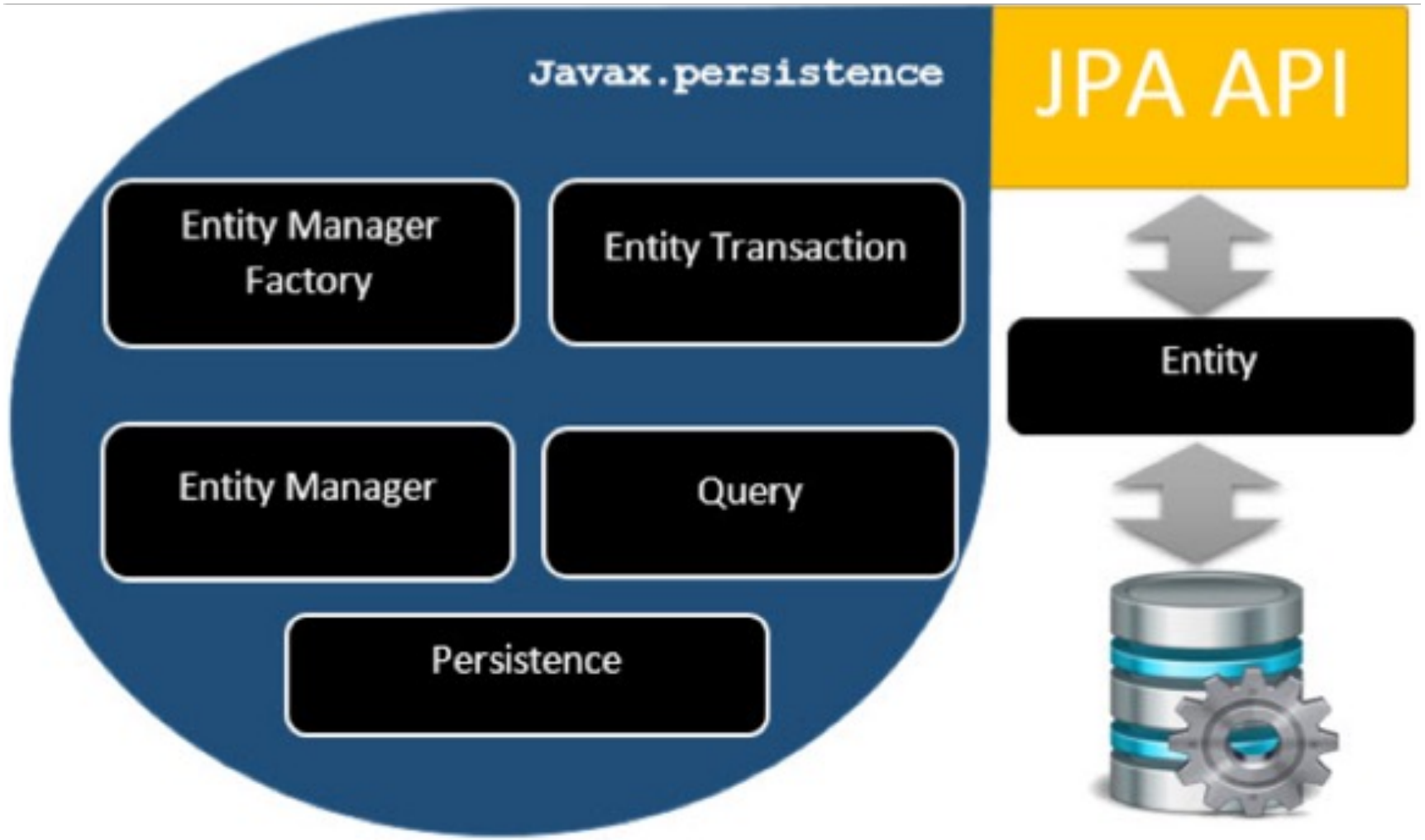
- Là kỹ thuật lập trình để ánh xạ các đối tượng ứng dụng vào các bảng cơ sở dữ liệu quan hệ.



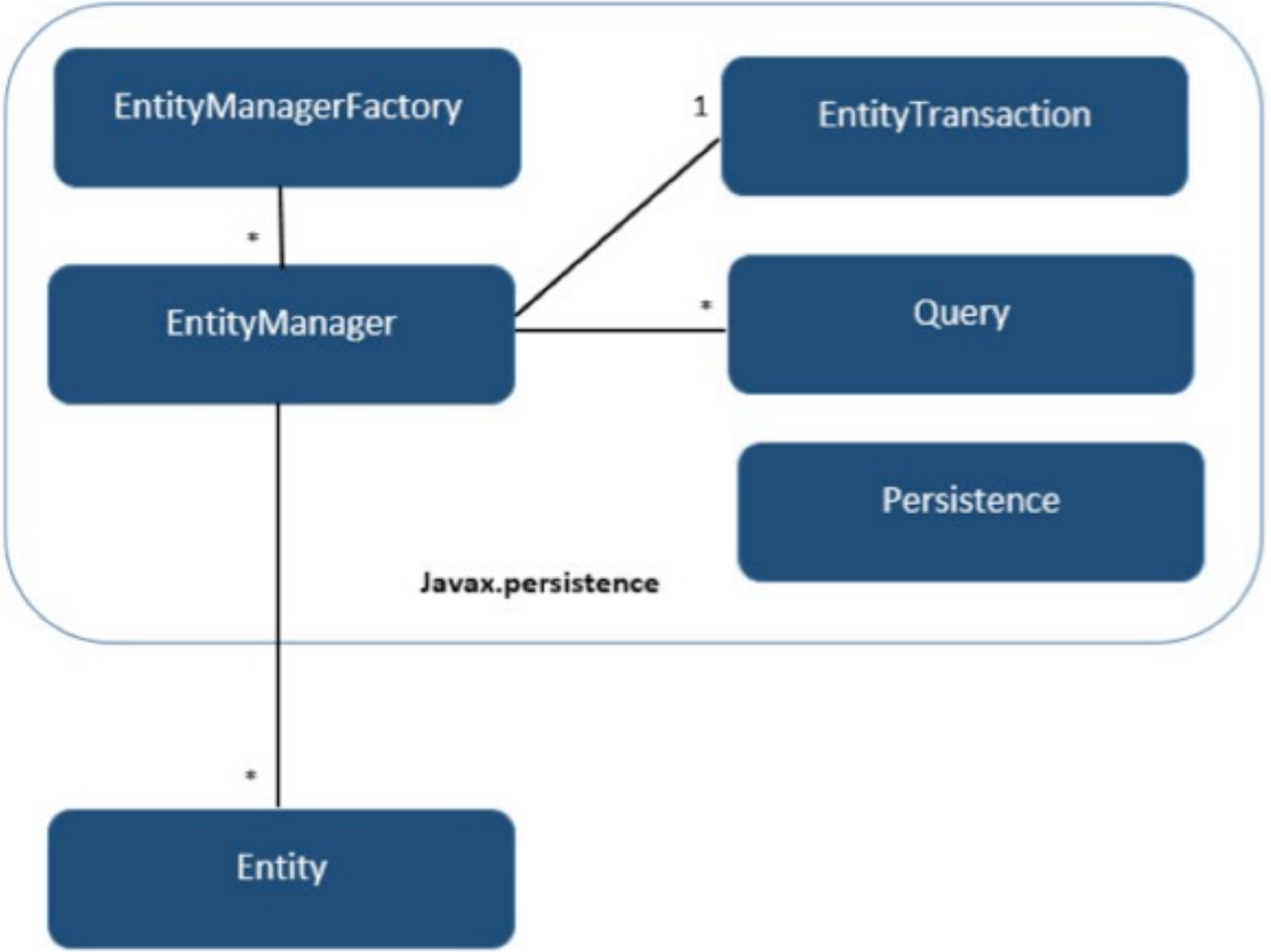
JPA

- Là một đặc tả cho việc ánh xạ quan hệ đối tượng để quản lý dữ liệu quan hệ trong các ứng dụng Java.
- Cung cấp một nền tảng làm việc trực tiếp với các đối tượng thay vì sử dụng các câu lệnh SQL.
- Chỉ xác định các thông số kỹ thuật, không cung cấp việc triển khai.
- Việc triển khai JPA được cung cấp bởi các nhà cung cấp O/R như **Hibernate**, EclipseLink và Apache OpenJPA.

Kiến trúc JPA



Mối liên hệ các lớp JPA



Entity class

- Entity là một POJO (Plain old Java Object)
- Class mô tả một bảng trong CSDL.
- Các biến thể hiện ánh xạ một dòng trong CSDL.
- Yêu cầu:
  - ✓ Phải có phương thức khởi dựng không đối số.
  - ✓ Được gắn annotation @Entity (javax.persistence.entity)
  - ✓ Class không được khai báo final.
  - ○ Nên hiện thực giao diện Serializable.
  - ○ Các field nên khai báo private.

Persistent Fields and Properties

- Trạng thái của entity có thể được truy xuất qua các field hoặc thuộc tính.
  1. Persistent fields:
  2. Persistent properties (Getter/Setter):

- Các kiểu dữ liệu được hỗ trợ:

  Kiểu cơ sở.

  String, Enum.

  Entity hoặc tập Entity.

  …

- Tất cả Fields/Properties không kết hợp khai báo @Transient đều được lưu trữ.

Primary key

- Mỗi entity phải có khai báo định danh đối tượng (persistent identifier)

```
@Entity
public class Employee {
    @Id
    private int id;
    private String name;
    private Date age;
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }. . . }
```

Identifier Generation

- Định danh có thể được tạo trong database bằng khai báo @GeneratedValue
- Có 4 dạng:
  - AUTO, **IDENTITY**, SEQUENCE, TABLE

```
@Id @GeneratedValue(strategy=AUTO)
private int id;
```

Khai báo không dùng mặc định

- Mặc định tên table được ánh xạ tới tên class
- Tên cột được ánh xạ tới tên field
- Trong trường hợp không trùng ta phải khai báo
  - @Table
  - @Column
- Ví dụ:

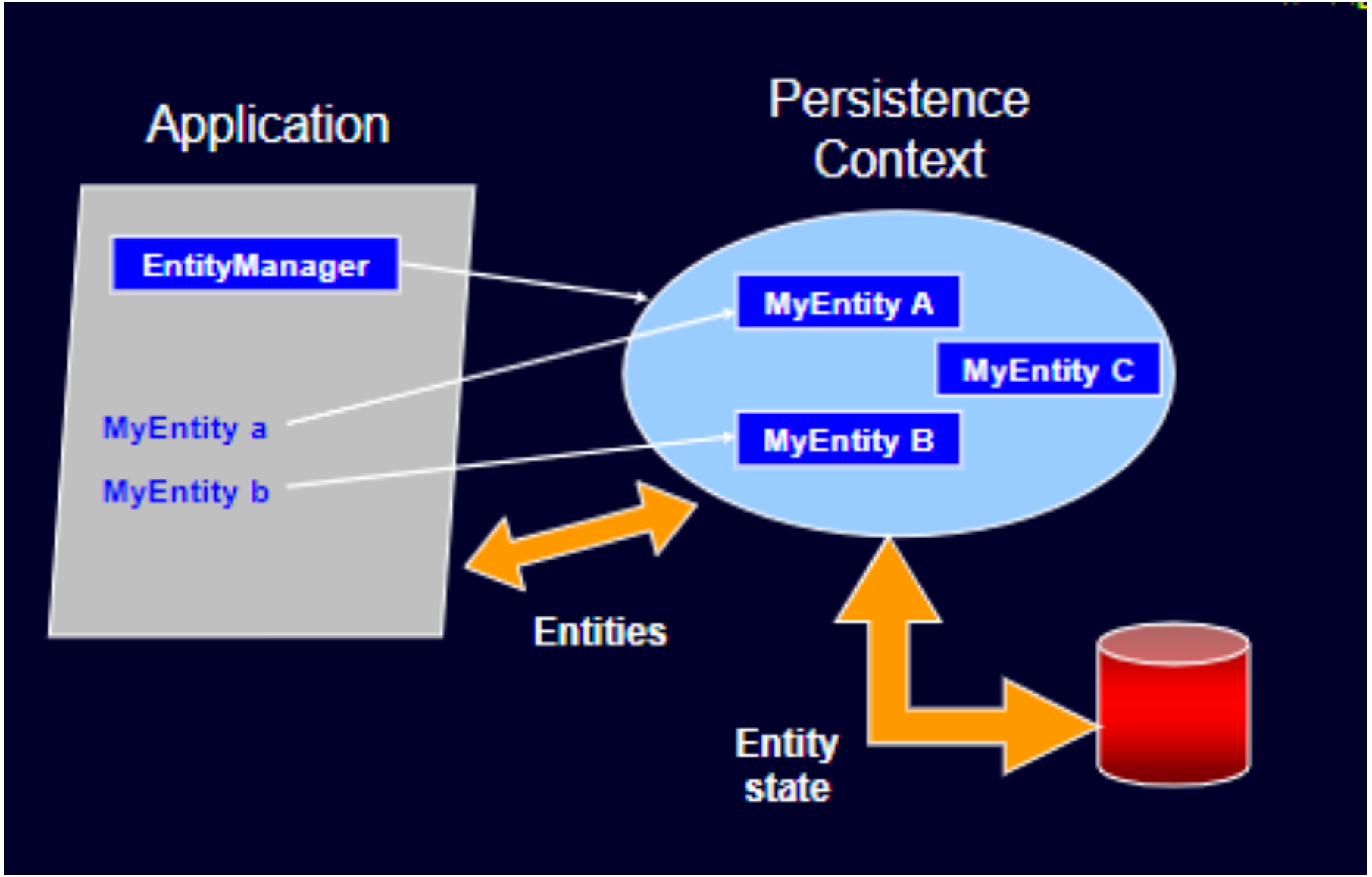
```
@Entity
@Table(name = "FULLTIME_EMPLOYEE")
public class Employee{ ... }
```

```
@Id @Column(name = "EMPLOYEE_ID", nullable = false)
private String id;

@Column(name = "FULL_NAME" nullable = true, length = 100)
private String name;
```

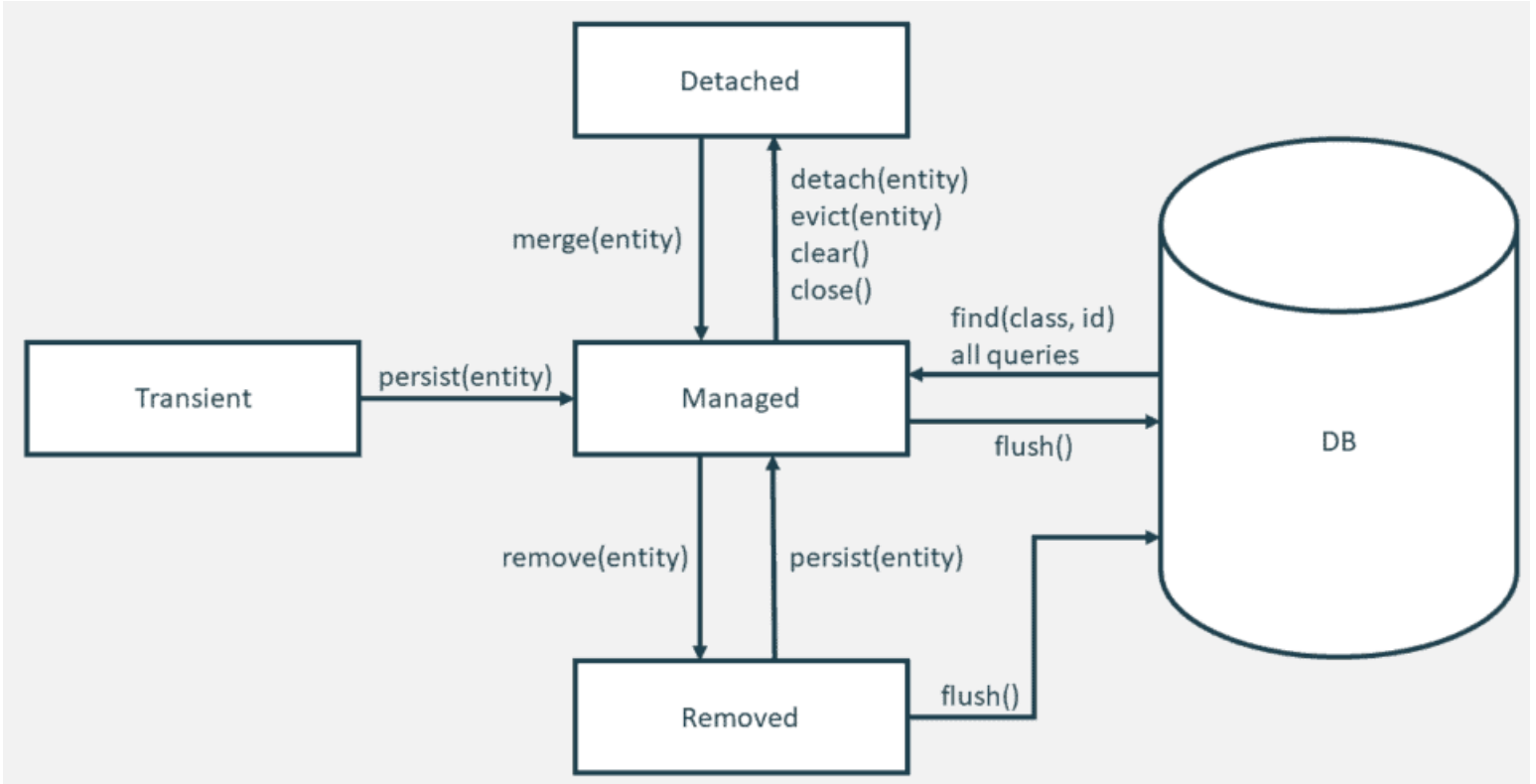
QUẢN LÝ ENTITY

- Các Entity được quản lý bởi đối tượng entity manager
- Mỗi thể hiện EntityManager được kết hợp với một "persistence context"
- Một "persistence context" xác định phạm vi theo đó các entity cụ thể được create, persist, và remove


Application
Persistence Context
EntityManager
MyEntity a
MyEntity b
MyEntity A
MyEntity C
MyEntity B
Entities
Entity state

EntityManager

- EntityManager sử dụng quản lý trạng thái và chu kỳ sống của các entity trong "persistence context"
- Các trạng thái của Entity:
  - 1.New
  - 2.Managed
  - 3.Detached
  - 4.Removed



Tạo EntityManager

```
public class PersistenceProgram {
  public static void main(String[] args)
  {
    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("SomePUnit");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    // Perform finds, execute queries,
…
    // update entities, etc.
    em.getTransaction().commit();
    em.close();
    emf.close();
  }
}
```

Transaction

- JPA transactions có thể quản lý bởi:
  - Application User
  - Framework (.. Spring)
  - Java EE container
- Transactions có thể điều khiển theo 2 cách:
  - Java Transaction API(JTA)
    - container-managed entity manager
  - EntityTransactionAPI (tx.begin(), tx.commit(), …)
    - Application-managed entity manager

Một số tác vụ trên đối tượng Entity

- persist()
- remove()
- refresh()
- merge()
- find()
- createQuery()
- createNamedQuery()
- createNativeQuery()
- contains()
- flush()

Persistence Unit

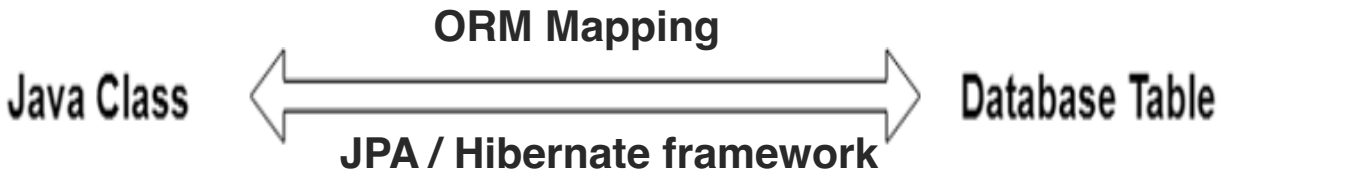
- “Persistence unit” định nghĩa một tập các class entity được quản lý bởi đối tượng trong ứng dụng.
- Mỗi “persistence unit” có thể có provider và database driver khác nhau.
- “Persistence unit” định nghĩa trong tập tin cấu hình “persistence.xml” đặt trong thư mục META-INF

persistence.xml

```
<persistence>
        <!-- Define persistence unit -->
        <persistence-unit name="qlsv-jpa">
                <!-- Define mapping classes (1) -->
                <class>domain.SinhVien</class>

                <properties>
                <!-- database connection -->
                <property name="javax.persistence.jdbc.url" value="url" />
                <property name="javax.persistence.jdbc.user" value="user" />
                <property name="javax.persistence.jdbc.password" value="pass" />
                 <!- Packages to scan entity (2) -->
                <property name="packagesToScan" value="dto"/>
                </properties>
        </persistence-unit>
</persistence>
```

Casestudy:



SinhVien
- mssv: int
- hoten: String
- dtb: float

SinhVien <<Table>>
mssv: int (primary key, auto)
hoten: varchar(50)
dtb: float