FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH JPA NÂNG CAO

J4 PROGRAMMING – SERVLET/JSP & JPA

www.poly.edu.vn

NỘI DUNG

- Phần 1: Thực thể kết hợp và ràng buộc
  - Phân tích CSDL mẫu
  - Ánh xạ ràng buộc 1-N và N-1
  - Ánh xạ ràng buộc duy nhất
  - Lập trình truy xuất dữ liệu dựa trên thực thể kết hợp
- Phần 2: JPQL, SQL, NamedQuery và Entity LifeCycle
  - JPQL
  - SQL - @NativeQuery
  - @NamedQuery, @NamedNativeQuery, @NamedStoredProcedureQuery
  - Entity LifeCycle

1 Entity kết hợp và ràng buộc

CASE STUDY

- CSDL PolyOE được xây dựng để quản lý video giải trí và cho phép người sử dụng chọn lựa các video yêu thích của mình.
  - Videos lưu tất cả video
  - Users lưu tất cả người sử dụng
  - Favorites lưu các video được yêu thích của người sử dụng

Videos

Column Name	Data Type	Identity
Id	char(11)	☐
Title	nvarchar(50)	☐
Poster	nvarchar(50)	☐
Description	nvarchar(MAX)	☐
Active	bit	☐
Views	int	☐
		☐

Favorites

Column Name	Data Type	Identity
Id	bigint	☑
VideoId	char(11)	☐
UserId	nvarchar(20)	☐
LikeDate	date	☐
		☐

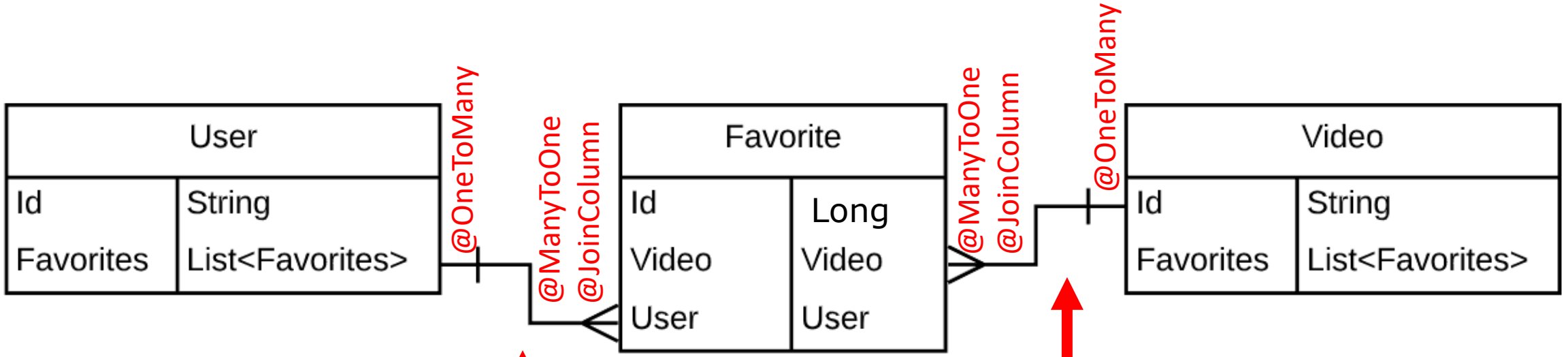
Users

Column Name	Data Type	Identity
Id	nvarchar(20)	☐
Password	nvarchar(50)	☐
Fullname	nvarchar(50)	☐
Email	nvarchar(50)	☐
Admin	bit	☐
		☐

@ANNOTATION ÁNH XẠ RÀNG BUỘC

- Một số ràng buộc được yêu cầu trong CSDL PolyOE
  - Favorites
    - Id: **tự tăng**
    - (VideoId, UserId): **duy nhất**
  - Ràng buộc khóa ngoại
    - Videos <-> Favorites **(1-N)**
    - Users <-> Favorites **(1-N)**
- Giải quyết yêu cầu bằng cách sử dụng các @Annotation
  - @Table**(uniqueConstraints)**: mô tả ràng buộc duy nhất
  - **@GeneratedValue**: mô tả field tự tăng
  - **@Temporal**: mô tả kiểu thời gian
  - **@ManyToOne @JoinColumn**: mô tả khóa ngoại (1-N)
  - **@OneToMany**: mô tả kết hợp 1-N

@ManyToOne & @OneToMany



Videos

	Column Name	Data Type	Identity
🔑	Id	char(11)	☐
	Title	nvarchar(50)	☐
	Poster	nvarchar(50)	☐
	Description	nvarchar(MAX)	☐
	Active	bit	☐
	Views	int	☐
			☐

Favorites

	Column Name	Data Type	Identity
🔑	Id	bigint	☑
	VideoId	char(11)	☐
	UserId	nvarchar(20)	☐
	LikeDate	date	☐
			☐

Users

	Column Name	Data Type	Identity
🔑	Id	nvarchar(20)	☐
	Password	nvarchar(50)	☐
	Fullname	nvarchar(50)	☐
	Email	nvarchar(50)	☐
	Admin	bit	☐
			☐

@ANNOTATION ÁNH XẠ RÀNG BUỘC

```
@Entity
@Table(name = "Favorites", uniqueConstraints={
      @UniqueConstraint(columnNames = {"VideoId" , "UserId"})
})
Favorite
```

```
@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
id: Integer
```

```
@ManyToOne @JoinColumn(name="VideoId")
video: Video
```

```
@ManyToOne @JoinColumn(name="UserId")
user: User
```

```
@Temporal(TemporalType.DATE)
likeDate: Date
```

```
@Entity @Table(name="Videos")
Video
```

```
@OneToMany(mappedBy="video")
favorites: List<Favorite>
```

```
...
```

```
@Entity @Table(name="Users")
User
```

```
@OneToMany(mappedBy="user")
favorites: List<Favorite>
```

```
...
```

- Thay khóa ngoại = @ManyToOne
- Ngược với @ManyToOne sẽ là @OneToMany

CÁC @ANNOTATION ÁNH XẠ

- Ràng buộc duy nhất
  - uniqueConstraints={@UniqueConstraint(columnNames={"VideoId" , "UserId"})}
- Ràng buộc tự tăng
  - @GeneratedValue(strategy = GenerationType.IDENTITY)
- Ràng buộc khóa ngoại (Kết hợp N-1)
  - @ManyToOne
  - @JoinColumn(name="VideoId")
- Kết hợp 1-N
  - @OneToMany(mappedBy="video")
- Thuộc tính thời gian (java.util.Date)
  - @Temporal(TemporalType.DATE)
- *Có thể bỏ @Column và @Table nếu*
  - *Entity Class và Table cùng tên*
  - *Field và Column cùng tên*

CÁC LỚP THỰC THỂ

```java
@Entity @Table(name = "Users")
public class User {
    @Id
    String id;
    String password;
    String fullname;
    String email;
    Boolean admin = false;
    @OneToMany(mappedBy = "user")
    List<Favorite> favorites;
```

```java
@Entity @Table(name = "Videos")
public class Video {
    @Id
    String id;
    String title;
    String poster;
    String description;
    Integer views = 0;
    Boolean active = true;
    @OneToMany(mappedBy = "video")
    List<Favorite> favorites;
```

favorites

user

user

favorites

```java
@Entity
@Table(name = "Favorites", uniqueConstraints={
        @UniqueConstraint(columnNames = {"VideoId" , "UserId"})
})
public class Favorite {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    @ManyToOne  @JoinColumn(name = "UserId")
    User user;
    @ManyToOne  @JoinColumn(name = "VideoId")
    Video video;
    @Temporal(TemporalType.DATE)
    Date likeDate = new Date();
```

- *Bổ sung thêm getters/setters và các constructors phù hợp*


DEMO

TỔNG HỢP @ANNOTATION ÁNH XẠ

Annotation	Parameter	Mô tả
@Entity		Entity Class
@Table(name, uniqueConstraints)	Name: tên bảng uniqueConstraints: duy nhất	Bảng
@Column(name)	Name:tên cột	Cột
@Id		Primary Key
@GeneratedValue (strategy)	Strategy: Cách sinh mã	Tự tăng
@Temporal(type)	Type: Kiểu thời gian	Date/Time/Timestamp
@ManyToOne		Foreign Key
@JoinColumn(name)	Name: tên cột khóa ngoại	
@OneToMany(mappedBy)	mappedBy: tên thực thể kết hợp @ManyToOne	Kết hợp theo khóa ngoại

TRUY VẤN THỰC THỂ KẾT HỢP

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("PolyOE");
EntityManager em = factory.createEntityManager();

// Truy vấn các thực thể kết hợp 1-N
List<Favorite> likes = em.find(User.class, "TeoNV").getFavorites();
                       ------------------
                            User
// Truy vấn thực thể kết hợp (N-1)
User user = em.find(Favorite.class, 1000).getUser();
                  ---------------------
                        Favorite
em.close();
```

- Ưu điểm của thực thể kết hợp là tự động JOIN
  - OneToMany: có User có thể suy ra các Favorite của User đó
  - ManyToOne: Có Favorite có thể suy ra đầy đủ thông tin của User kết hợp với Favorite đó

TRUY VẤN LIÊN QUAN THỰC THỂ KẾT HỢP

```
// Truy vấn các video được yêu thích bởi User có id là TeoNV
String jpql = "SELECT o.video FROM Favorite o WHERE o.user.id=:uid";
TypedQuery<Video> query = em.createQuery(jpql, Video.class);
query.setParameter("uid", "TeoNV");
List<Video> videos = query.getResultList();

// Truy vấn các video được yêu thích có tựa chứa "tình yêu"
String jpql = "SELECT o.video FROM Favorite o WHERE o.video.title LIKE :keyword";
TypedQuery<Video> query = em.createQuery(jpql, Video.class);
query.setParameter("keyword", "%tình yêu%");
List<Video> videos = query.getResultList();
```

- Sử dụng DISTINCT để loại các video xuất hiện nhiều lần

TRUY VẤN LIÊN QUAN THỰC THỂ KẾT HỢP

```
// Truy vấn các video không được yêu thích
String jpql = "SELECT o FROM Video o WHERE o.favorites IS EMPTY";
TypedQuery<Video> query = em.createQuery(jpql, Video.class);
List<Video> videos = query.getResultList();

// Truy vấn số lượt yêu thích của video có mã là "N5nFWe0Wr_4"
String jpql = "SELECT count(o) FROM Favorite o WHERE o.video.id=?0";
TypedQuery<Long> query = em.createQuery(jpql, Long.class);
query.setParameter(0, "N5nFWe0Wr_4");
Long count = query.getSingleResult();
```

- *o.favorites IS EMPTY: tập hợp rỗng*
- *count() luôn luôn trả về Long*


DEMO

TRUY VẤN LIÊN QUAN THỰC THỂ KẾT HỢP

```
// Thống kê số lượt, thời gian truy xuất của các video
String jpql = "SELECT " +
              " o.video.title, count(o.id), min(o.likeDate), max(o.likeDate) " +
              " FROM Favorite o " +
              " GROUP BY o.video.title";
TypedQuery<Object[]> query = em.createQuery(jpql, Object[].class);
List<Object[]> list = query.getResultList();
```

Dựa vào mệnh đề SELECT dễ dàng suy ra Object[] gồm 4 phần tử và có kiểu dữ liệu là {String, Long, Date, Date}

TRUY VẤN LIÊN QUAN THỰC THỂ KẾT HỢP

```
@Entity
public class Report {
        @Id
        Serializable group;
        Long likes;
        Date newest;
        Date oldest;
        public Report(Serializable group, Long likes, Date newest, Date oldest) {
                this.group = group;
                this.likes = likes;
                this.newest = newest;
                this.oldest = oldest;
        }
        getters/setters
}
```

```
String jpql = "SELECT new Report(o.video.title, count(o), "
            + " max(o.likeDate), min(o.likeDate)) "
            + " FROM Favorite o "
            + " GROUP BY o.video.title ";
TypedQuery<Report> query = em.createQuery(jpql, Report.class);
List<Report> list = query.getResultList();
```


DEMO

② JPQL, SQL, Stored Proc

JPQL LÀ GÌ?

- JPQL là ngôn ngữ truy vấn đối tượng có cú pháp tương tự SQL, sử dụng
  - **Entity** thay vì ~~Table~~
  - **Property** thay vì ~~Column~~
- Ví dụ:

SELECT o.video FROM Favorite o WHERE o.id=2020

- **Favorite** hiểu là **Entity Class**
- o.**video** hiểu là o.**getVideo()**
- o.**id** hiểu là o.**getId()**

```
SELECT [DISTINCT] <Properties>
FROM <Entity Class>
[WHERE <Condition>]
[GROUP BY <Expression>]
[ORDER BY <Expression> DESC|ASC]
```

TOÁN TỬ TRONG JPQL

- JPQL sử dụng các toán tử thông thường
  - Số học: +, -, *, /
  - So sánh: >, >=, <, <=, !=
  - Quan hệ: AND, OR, NOT
- Toán tử đặc biệt
  - [NOT] IN
  - [NOT] BETWEEN
  - IS [NOT] NULL
  - [NOT] LIKE
  - IS [NOT] EMPTY

- Ví dụ:

```
SELECT o FROM User o WHERE o.email LIKE '%@fpt.edu.vn'

SELECT o FROM Favorite o WHERE year(o.likeDate) BETWEEN 2010 AND 2020 ORDER BY o.likeDate DESC

SELECT o FROM Video o WHERE o.description IS NOT NULL

SELECT o FROM Favorite o WHERE o.user.id IN ('TeoNV', 'NghiemN')

SELECT count(o) FROM Video o WHERE o.favorites IS EMPTY
```

JPQL HÀM

- Tổng hợp số liệu
  - Sum(), Count(), Min(), Max(), Avg(), Size()
- Xử lý chuỗi
  - Length(), Trim(), Lower(), Upper(), Substring(), Concat(), Locate()
- Xử lý thời gian
  - Year(), Month(), Day(), Hour(), Minute(), Second(), Current_Date(), Current_Time(), Current_Timestamp()
- Các hàm khác
  - Sqrt(), str(), cast(), ceil(), floor()

```
SELECT month(o.likeDate), count(o),
max(o.likeDate) FROM Favorite o
GROUP BY month(o.likeDate)

SELECT upper(o.fullname),
lower(o.email) FROM User o  WHERE
locate(o.email, '@fpt.edu.vn') >= 0

SELECT o FROM Favorite o WHERE
(year(current_date())-
year(o.likeDate)) > 5

SELECT count(o)/8.0 FROM Video o
```

TRUY VẤN VỚI JPQL

```
String jpql = "SELECT... FROM..."
TypedQuery<X> query = em.createQuery(jpql, X.class);
List<X> list = query.getResultList();
X value = query.getSingleResult();
```

- Căn cứ vào kiểu dữ liệu của mệnh đề SELECT để xác định **X**
- Căn cứ vào toàn bộ câu lệnh JPQL để xác định số lượng phần tử kết quả, từ đó sử dụng phương thức nào để truy vấn dữ liệu
  - ***getResultList***()
  - ***getSingleResult***()

XÁC ĐỊNH KIỂU VÀ CHỌN PHƯƠNG THỨC TRUY VẤN

- SELECT o FROM Favorite o
  - => TypedQuery<**Favorite**>.**getResultList**()
- SELECT o.admin FROM User o
  - => TypedQuery<**Boolean**>.**getResultList**()
- SELECT o.fullname, o.admin FROM User o
  - => TypedQuery<**Object[]**>.**getResultList**()
- SELECT min(o.views) FROM Video o
  - => TypedQuery<**Long**>.**getSingleResult**()
- SELECT o FROM User o WHERE o.id='TeoNV'
  - => TypedQuery<**User**>.**getSingleResult**()
- SELECT o.user FROM Favorite o WHERE o.id=2020
  - => TypedQuery<**User**>.**getSingleResult**()

@NamedQuery

```
@NamedQueries({
      @NamedQuery(name="User.findAll", query="SELECT o FROM User o"),
      @NamedQuery(name="User.findByEmail",
            query="SELECT o FROM User o WHERE o.email LIKE :email"),
      @NamedQuery(name = "User.findByRole",
            query="SELECT o FROM User o WHERE o.admin=:role"),
})
@Entity
@Table(name = "Users")
public class User {...}
```

Khai báo jpql trên từng Entity và **đặt tên** cho chúng

Sử dụng các jpql **đã khai báo** để truy vấn

```
TypedQuery<User> query = em.createNamedQuery("User.findAll", User.class);
List<User> list = query.getResultList();

TypedQuery<User> query = em.createNamedQuery("User.findByRole", User.class);
query.setParameter("role", true);
List<User> list = query.getResultList();

TypedQuery<User> query = em.createNamedQuery("User.findByEmail", User.class);
query.setParameter("email", "nghiemn@fpt.edu.vn");
User entity = query.getSingleResult();
```

SQL - TRUY VẤN ĐẶC THÙ

```
String sql = "SELECT * FROM Users WHERE email LIKE ?";
Query query = em.createNativeQuery(sql, User.class);
query.setParameter(1, "%@gmail.com");
List<User> list = query.getResultList();
```

- JPA cho phép truy vấn dữ liệu với câu lệnh ***SQL đặc thù*** của từng hệ quản trị CSDL.
- Ưu điểm:
  - Tận dụng được sức mạnh của hệ quản trị của CSDL
- Hạn chế:
  - Thay đổi hệ quản trị CSDL thì phải viết lại SQL

@NamedNativeQuery

```
@NamedNativeQueries({
        @NamedNativeQuery(
                name = " Users .findByEmail",
                query = "SELECT * FROM Users WHERE email LIKE ?",
                resultClass = User.class
        )
})
@Entity
@Table(name = "Users")
public class User {...}
```

```
Query query = em.createNamedQuery("Users.findByEmail");
query.setParameter(1, "%@gmail.com");
List<User> list = query.getResultList();
```

XÉT THỦ TỤC LƯU

```
CREATE PROCEDURE sp_FindByEmail(@Email NVARCHAR(50))
AS
BEGIN
        SELECT * FROM Users WHERE email LIKE @Email
END
```

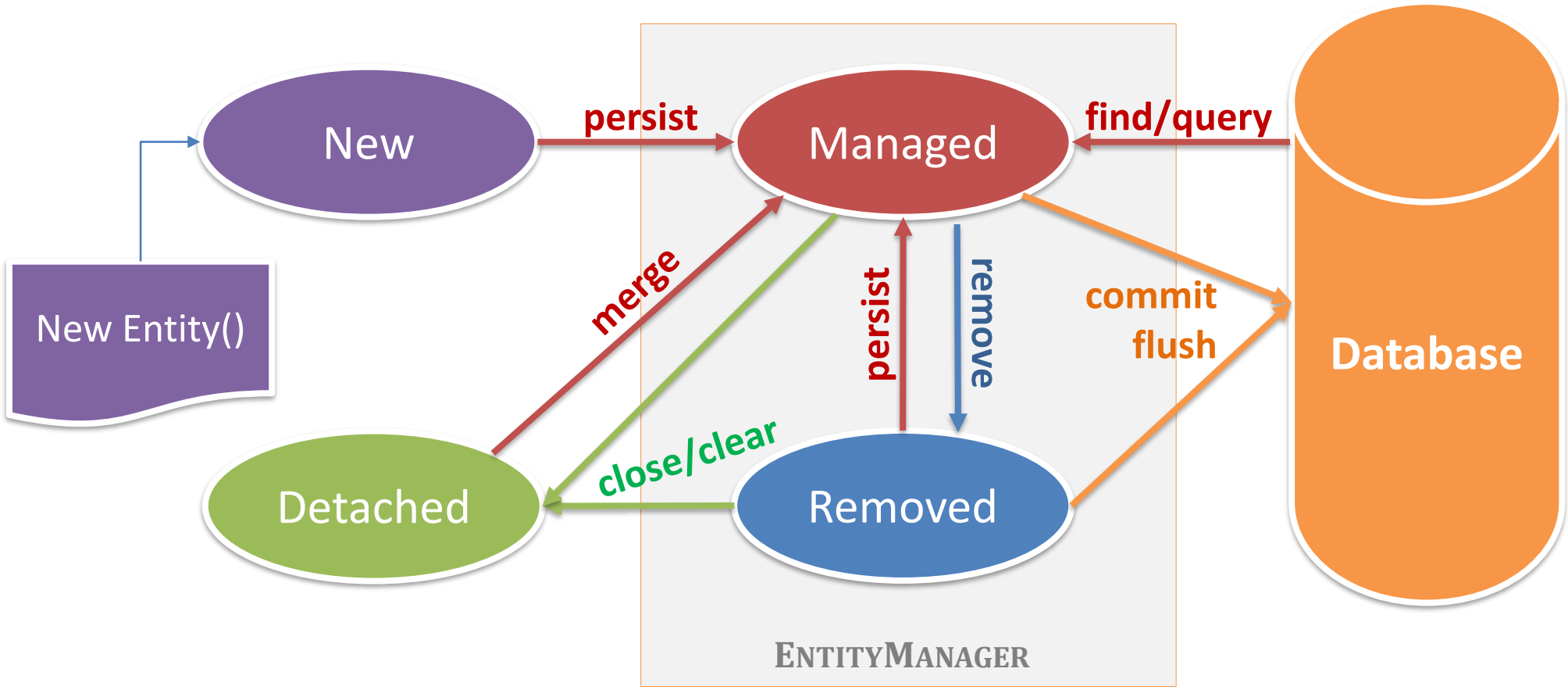
Gọi thủ tục lưu trong JPA như thế nào?

@NamedStoredProcedureQuery

```
@NamedStoredProcedureQueries({
        @NamedStoredProcedureQuery(
                name="User.spFindByEmail",
                procedureName = "sp_FindByEmail",
                resultClasses = {User.class},
                parameters = @StoredProcedureParameter(name = "email", type = String.class)
        )
})
@Entity
@Table(name = "Users")
public class User {...}
```

```
StoredProcedureQuery query =
        em.createNamedStoredProcedureQuery("User.spFindByEmail");
query.setParameter("email", "%@gmail.com");
List<User> list = query.getResultList();
```

JPA Entity Object Life Cycle



SUMMARY

- ☑ Ánh xạ thực thể kết hợp
  - ☑ @OneToMany(mappedBy)
  - ☑ @ManyToOne @JoinColumn
- ☑ Ràng buộc duy nhất
- ☑ Truy vấn các thực thể kết hợp
- ☑ Tìm hiểu sâu hơn về JPQL
- ☑ Truy vấn đặc thù (sử dụng SQL)
- ☑ @NamedQuery, @NamedNativeQuery
- ☑ @NamedStoredProcedureQuery
- ☑ Vòng đời của entity


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn