ORM là kỹ thuật lập trình cho phép tương tác với CSDL quan hệ bằng cách sử dụng đối tượng. Thay vì sử dụng câu lệnh sql ta có thể sử dụng các đối tượng và phương thức để truy vấn sql.

ORM giúp tăng năng xuất vì không phải viết các đoạn mã sql dài dòng phức tạp.

Giảm độ phức tạp : tự động xử lý việc kết nối

Dễ bảo trì : không cần phải thay đổi quá nhiều khi có thay đổi về cấu trúc. Vì chỉ cần thay đổi cấu trúc của đối tượng.

Đơn giản việc chuyển đối CSDL : chỉ cần thay đổi thư viện và cấu hình database

Cơ hoạt động :

* Mỗi bảng được ánh xạ với 1 lớp, và mỗi 1 thuộc tính được ánh xạ tới 1 cột trong db từ đó dễ dàng mapping dữ liệu truy vấn.
* CRUD Operations: Tự động sinh ra câu lệnh sql
* Lazy Loading: Trong truy vấn có mối quan hệ 1-n, ORM sẽ không join dữ liệu ngay lập tức mà chỉ khi nào lấy dữ liệu của field mới thực hiện join lấy dữ liệu.

Ngày ra còn có EAGER sẽ truy vấn join ngay khi lấy dữ liệu.

* Hỗ trợ caching :

First level cache : tự động bật, được quản lý trong session.

Second level cache : bật thủ công, các session khác nhau cũng có thể lấy được cache.

Trong JPA nếu muốn sử dụng First level thì hàm gọi phương thức truy vấn phải thêm annotation @Transactional

Hiệu xuất:

JDBC hiệu xuất cao hơn với trường hợp thao tác với dữ liệu lớn và có thể quản lý dữ liệu join dễ dàng.

JDBC dễ dàng tuỳ chỉnh lấy dữ liệu cột cần thiết.

ORM chỉ có thế mạnh về việc tối ưu thời gian phát triển.

JPA là triển khai ORM vì nó có cơ chế ánh xạ tương tự ORM nhưng nó kết hợp giữa nhiều ORM khác nhau: Hibernate, eclipseLink, OpenJPA

OneToMany:

Một dữ liệu ở bảng hiện tại được liên kết với nhiều dữ liệu ở bảng khác.

Cần khai báo map với class nào

Many to One:

Nhiều bản ghi ở bảng hiện tại map với 1 bản ghi ở bảng khác

Cần khai báo tên cột cần map

ManyToMany:

Một bản ghi ở bảng hiện tại map với nhiều bản ghi ở bảng khác và cũng có thể nhiều bản ghi ở bảng hiện tại được map bởi 1 bản ghi ở bảng khác.

ORM tự động sinh ra bảng trung gian

CascadeType: các thao tác trên thực thể cha ảnh hưởng tới các dữ liệu con.

Mặc định khi thêm không có CascadeType

Convert:

Sử dụng thư viện:

Sử dụng mapstruct giúp tăng hiệu xuất map: tạo ra lớp interface khai báo đối tượng cần convert, những lưu ý khi map bằng annotation: @mapping

Thư viện sẽ hỗ trợ triển khai interface đó ra 1 class.

Sử dụng Java Reflection:

Sử dụng khi không muốn thêm thư viện

Khó khăn khi thao tác với nhiều đối tượng.

Giảm hiệu xuất.

Native query: Cho phép viết câu lệnh sql trong ORM.

* Có thể định nghĩa trực tiếp trong repository sử dụng EntityManager
* Định nghĩa trong Entity:

Sử dụng annotation NameNativeQuery

* Truyền tham số

Truyền tham số qua đối tượng Query.

Những vị trí trong sql cần truyền tham số có cấu trúc :<name>

* Dữ liệu nhận được từ native query là list

Muốn quản lý commit hay rollback cần khởi tạo EntityTransaction

Specification: cho phép điều chỉnh điều kiện 1 cách linh hoạt thông qua Specification

Page: Giúp xác định kích thước dữ liệu và sắp xếp, vị trí cắt dữ liệu.

Điểm yếu của jdbc là khi thay đổi nghiệp vụ là phải thay lại hết câu sql.

Điểm yếu jpa: mất tg gen query

Có thể bị stackoverfolow: vòng lặp vô tận.

Mắc query n+1 nếu sử dụng fetchType: lazy

Muốn update hay delete phải select trước

Hấng dữ liệu bằng interface

Sử dụng dto để hấng

* Sử dụng cho native query

Gtbt:

Tên:

Lĩnh vực, vị trí ứng tuyển

Số năm kinh nghiệm

Project gần nhất

Các nghiệp vụ xử lý

Công nghệ sử dụng trong project

Cậu hỏi:

Trả lời lý thuyết, công dụng, cách sử dụng, khi nào sử dụng