1. Eager and Lazy
2. Fetch Mode
3. N+1 Problem
4. Schema Migration

Git ✓
Spring Boot ✓
APIs ✓
DB ✓

Category {
    - List <Product> products;
    - id
    - name
}
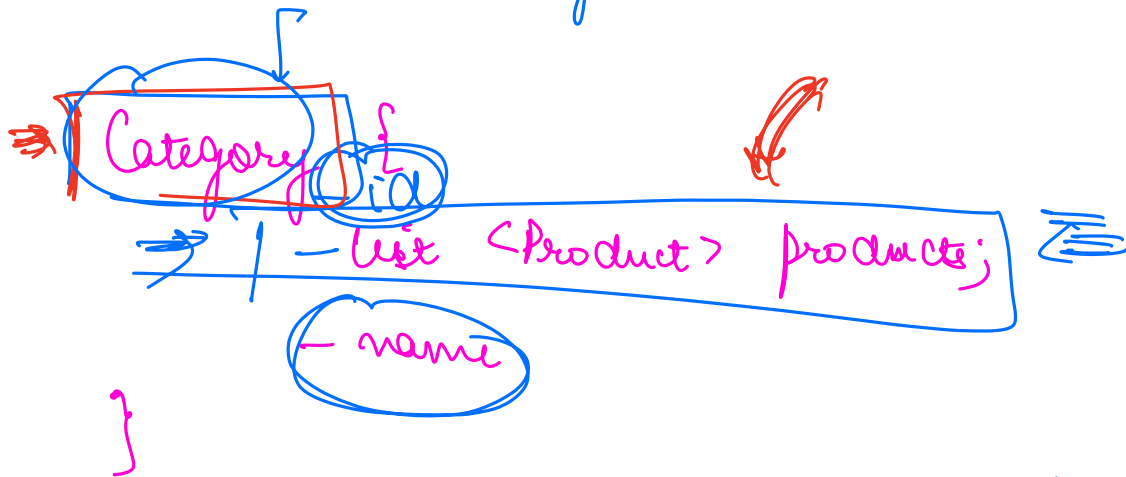
CategoryRepository . fetchById (1)
→ 1000 products

Category

Select ^
from categories c
join products p
on c.id = p.c_id

Cons :

① Queries will become slow.

② Use more memory

Pro:

① Single DB call

---

2 types of fetches: (Attributes)

① lazy fetch → Only fetched when it is accessed

② Eager Fetch :
  ↳ will be fetched auto alongwith the obj of Parent.

@Getter

Class (Category) {

eager → int id; eager

String name;

User creator;

lazy { List <Product> → products;

}

Category C = Category Repository. fetchById (1)

→ [C. get Products ()] ⟸ { Select * from products where c.id =1 }

Fetch products only if needed.

Pro

① you will only fetch an obj if it is needed.

→ Faster Query

→ lesser Mem usage

Cons

① System may become slow unexpectedly.

Eager

```
Select *
from category c
join product p
on p. cat-id = c.id
```

Lazy

```
Select *
① from category c
where c.id = _
```

```
Select *
from product
where cat-id = ?
```

Category. get Products ()

Category 1
→ P1
→ P2
→ P3

Category 2
→ P4

→ P4

Category 3 → P6
→ P7

List<UUID> category IDs
↓

Get Product Titles (List <Categories>)
→ { c1, c2, c3 }

List < String > titles [ ]

① Get these categories

    find All By Id In (CategoryIDs)   (N)

② for each of these categories.

    get their products

    put the title in a list

① Get category from IDs   (H)   ⇒ ①
for each category : (N)   ⇒ ②

    get Products for that category.

FetchType → (when) will an attr be
fetched

eager → immediately

lazy → when needed

fetch Mode → (How) will an attr be fetched
→ SELECT
→ JOIN
→ SUBQUERY

@ Query ("   =   ")