Agenda.

→ Query Methods
  → Declared Queries
  → HQL
  → SQL.

→ Representing Cardinalities.
  → Mapped By
  → Cascading

→ Fetch Types { Eager, Lazy }

→ Schema Versioning.

JPA Repository

findById(--) =>

```
select * from products
where title like '%iphone%'
```

https://docs.spring.io/spring-data/jpa/reference/repositories/query-methods-details.html

Declared Queries.

⇒ No need to write SQL queries on our own.

⇒ Just write the method name & ORM will convert that method name into corresponding query

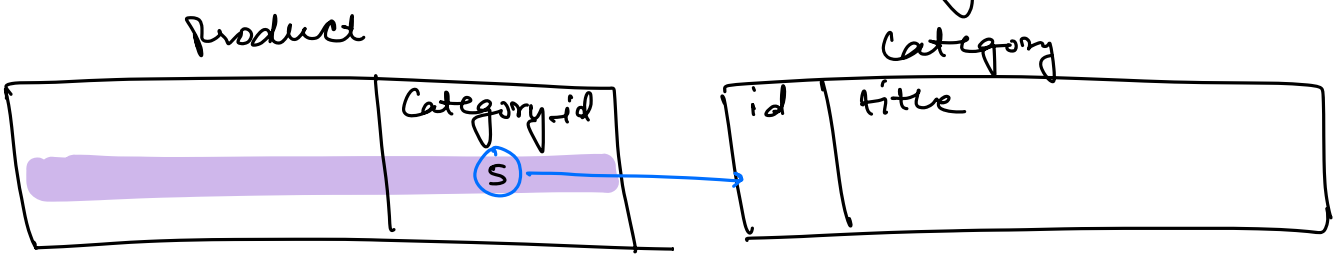
All select queries, like method, delete, update will be converted by JPA in corresponding queries.

You just have to write the method name and ORM will convert these into your corresponding queries

Fetch Type.

```
Class Product {
    id
    title
    desc
    =
    Category ↓ EAGER.
}
```

Product p = ProductRepo.findById(10)



Left Join

Class Category {

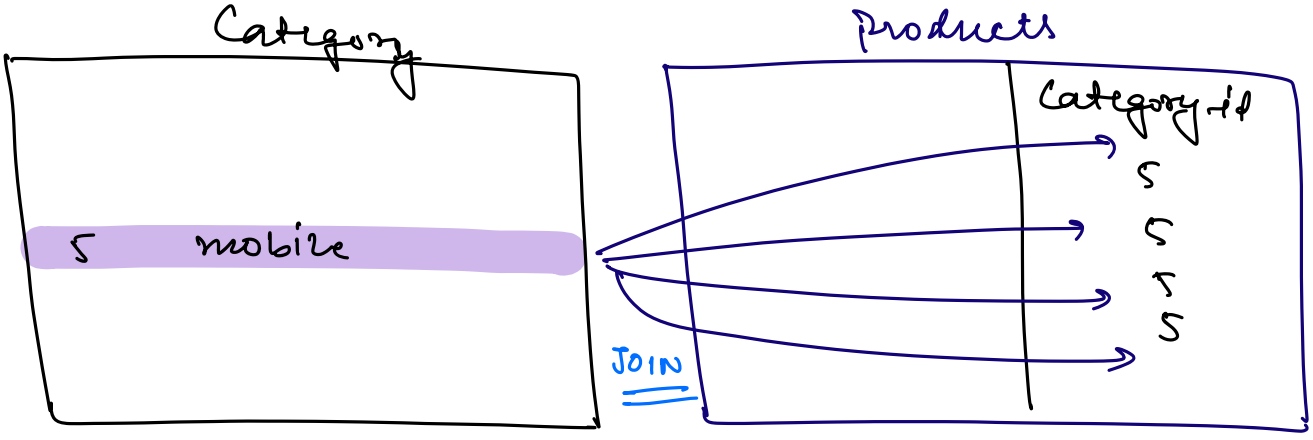
id

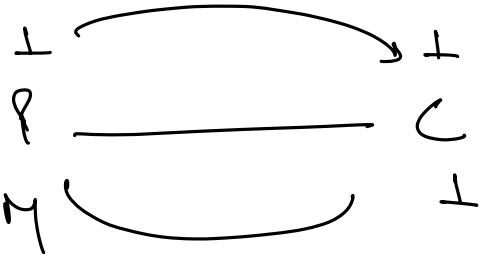
title

List<Product> products → lazy

}

Category c = categoryRepository.findById(5)





When a class has an attr of another class then to fetch the details of other attrs, JOIN operation needs to be performed.

⇒ Fetch Type
- Eager
- Lazy

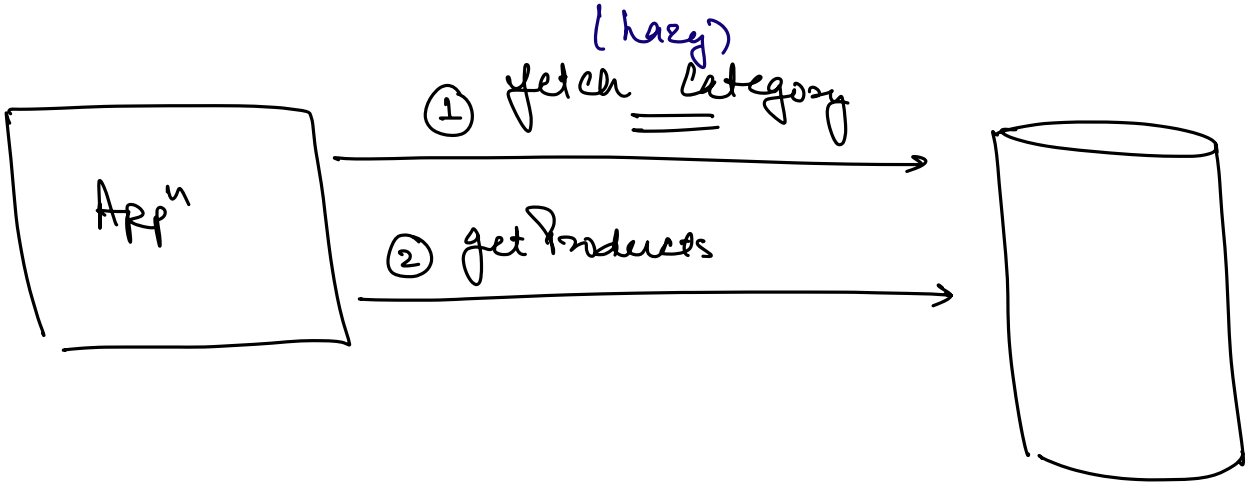
Eager : Fetch the details of inner object along with outer object.

Lazy : Don't fetch the details of inner object along with the outer object.

Only fetch when required.

```
Category c = CategoryRepository.findById(10)
```

```
List<Product> products = c.getProducts()
```



⇒ By default, all the attrs are fetched EAGERLY except collections.

⇒

```
A {
    B b; ⇒ EAGER.
}
```

⇒ JOIN.

```
X {
    List<Y> ⇒ LAZY
}
```

HQL & SQL

Hibernate Query Language.

SQL + OOPS.

=> We don't have complete control over the declard query.

=> Performance.

=> If we have usecase of writing custom queries, we can write using HQL | SQL.