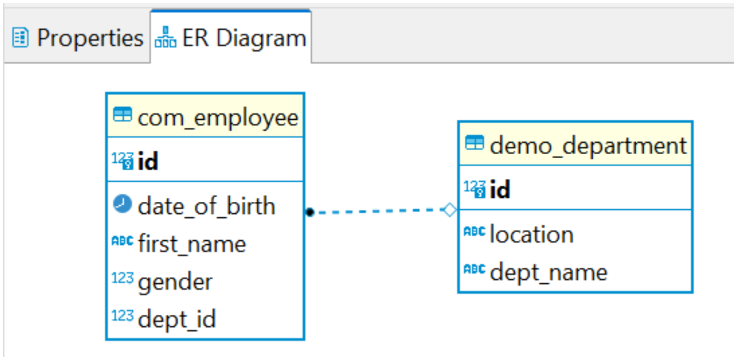
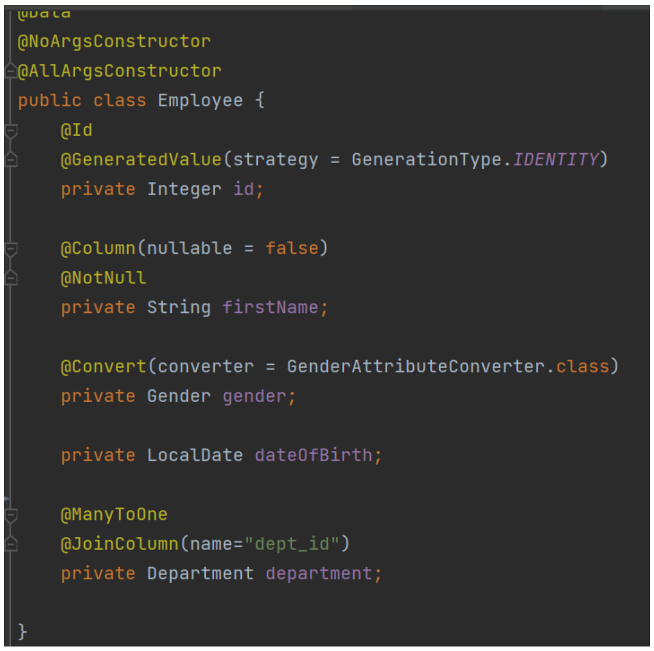
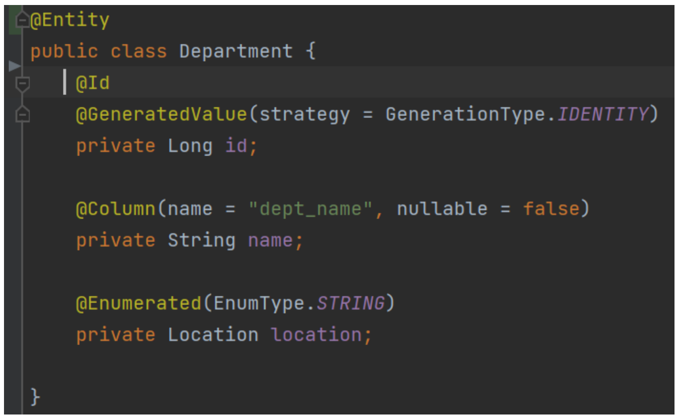
N+1 Problem





```java
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    @NotNull
    private String firstName;

    @Convert(converter = GenderAttributeConverter.class)
    private Gender gender;

    private LocalDate dateOfBirth;

    @ManyToOne
    @JoinColumn(name="dept_id")
    private Department department;

}
```

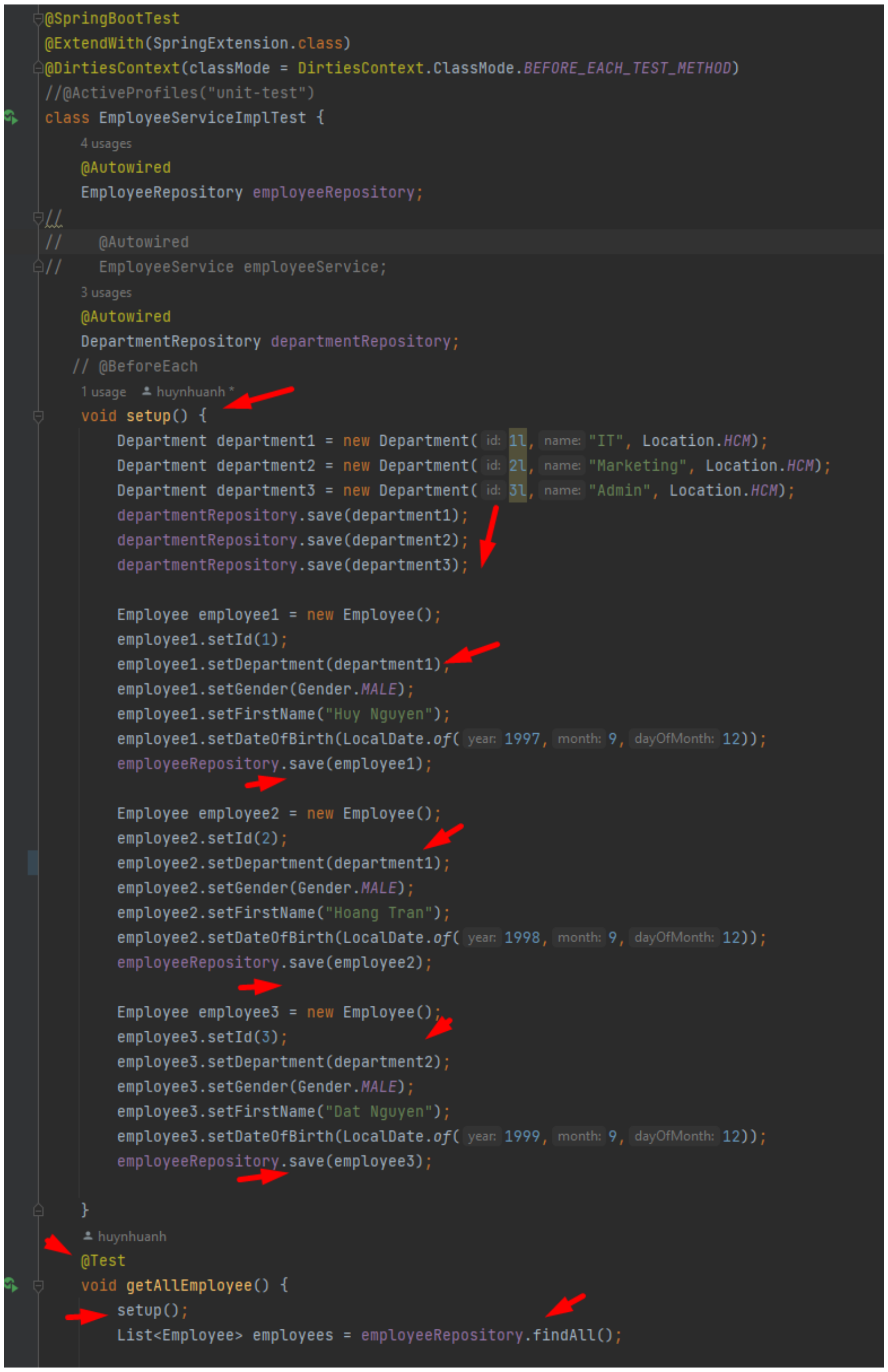


```java
@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "dept_name", nullable = false)
    private String name;

    @Enumerated(EnumType.STRING)
    private Location location;

}
```

Write Unit test for EmployeeService. Connect to H2 Database and create a setup method to:

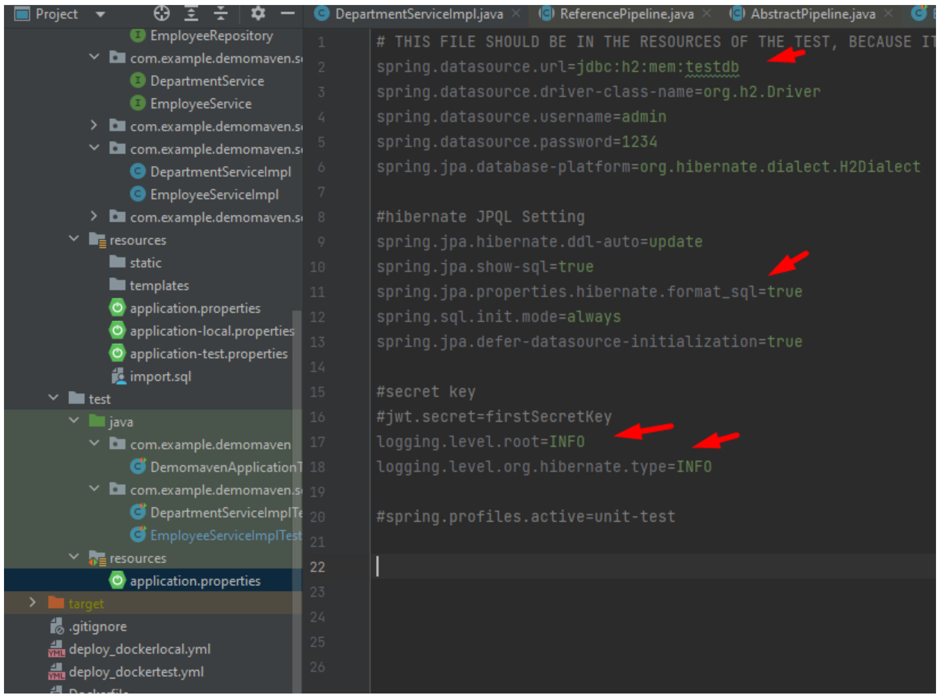
- Insert 3 Departments: IT, Marketing & Admin to Department table
- Insert 3 Employees: 2 first Employees in IT Department and 3rd belongs to Marketing

As following picture:



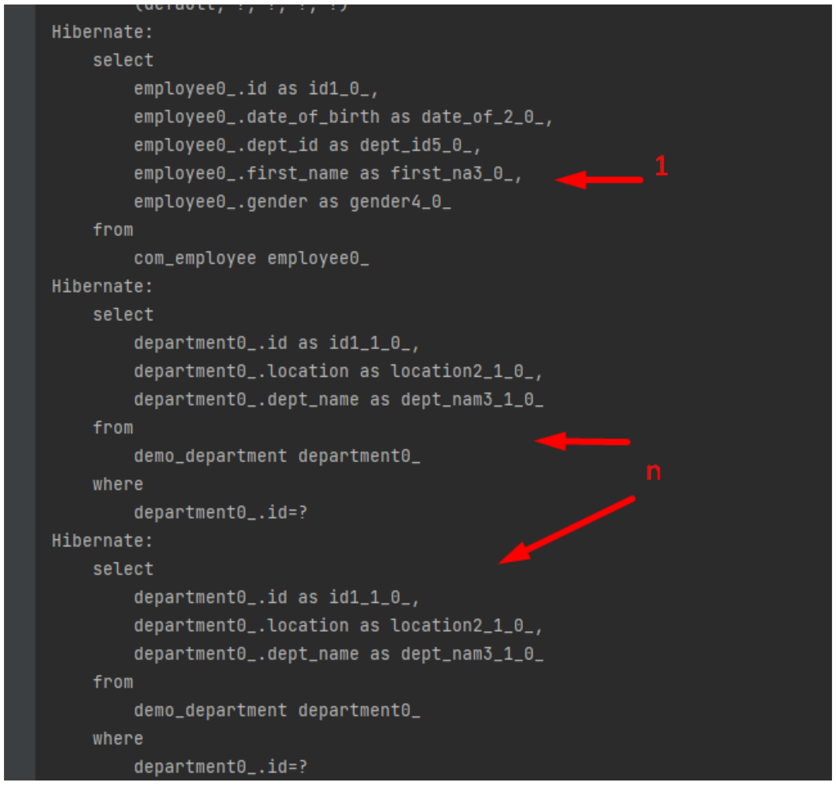
```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@DirtiesContext(classMode = DirtiesContext.ClassMode.BEFORE_EACH_TEST_METHOD)
//@ActiveProfiles("unit-test")
class EmployeeServiceImplTest {
    4 usages
    @Autowired
    EmployeeRepository employeeRepository;
//
//    @Autowired
//    EmployeeService employeeService;
    3 usages
    @Autowired
    DepartmentRepository departmentRepository;
   // @BeforeEach
    1 usage  huynhuanh *
    void setup() {
        Department department1 = new Department( id: 1l, name: "IT", Location.HCM);
        Department department2 = new Department( id: 2l, name: "Marketing", Location.HCM);
        Department department3 = new Department( id: 3l, name: "Admin", Location.HCM);
        departmentRepository.save(department1);
        departmentRepository.save(department2);
        departmentRepository.save(department3);

        Employee employee1 = new Employee();
        employee1.setId(1);
        employee1.setDepartment(department1);
        employee1.setGender(Gender.MALE);
        employee1.setFirstName("Huy Nguyen");
        employee1.setDateOfBirth(LocalDate.of( year: 1997, month: 9, dayOfMonth: 12));
        employeeRepository.save(employee1);

        Employee employee2 = new Employee();
        employee2.setId(2);
        employee2.setDepartment(department1);
        employee2.setGender(Gender.MALE);
        employee2.setFirstName("Hoang Tran");
        employee2.setDateOfBirth(LocalDate.of( year: 1998, month: 9, dayOfMonth: 12));
        employeeRepository.save(employee2);

        Employee employee3 = new Employee();
        employee3.setId(3);
        employee3.setDepartment(department2);
        employee3.setGender(Gender.MALE);
        employee3.setFirstName("Dat Nguyen");
        employee3.setDateOfBirth(LocalDate.of( year: 1999, month: 9, dayOfMonth: 12));
        employeeRepository.save(employee3);

    }
    huynhuanh
    @Test
    void getAllEmployee() {
        setup();
        List<Employee> employees = employeeRepository.findAll();
```

Remember to set log level to Info in properties file in Test Resource;



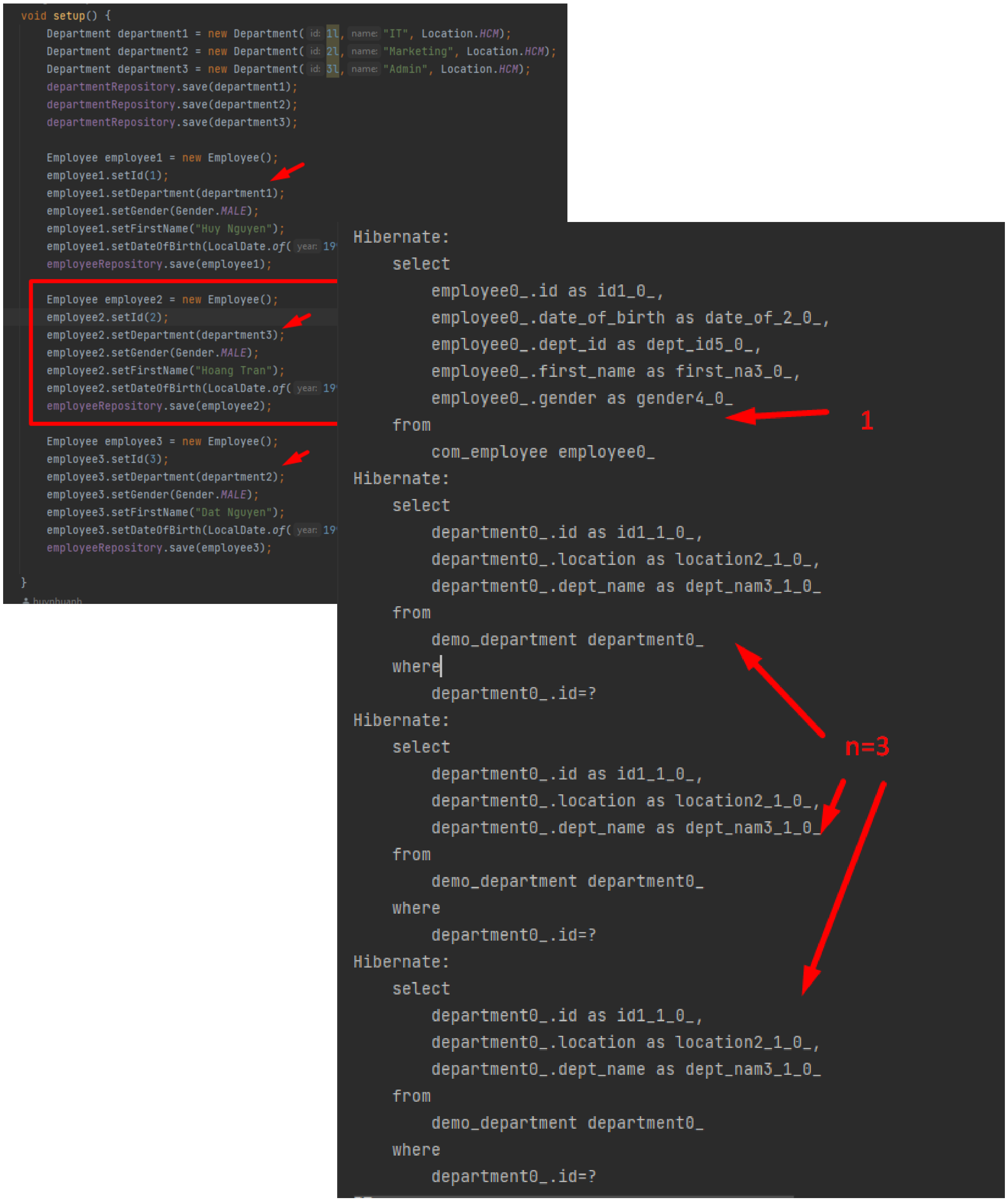
```
# THIS FILE SHOULD BE IN THE RESOURCES OF THE TEST, BECAUSE IT
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=admin
spring.datasource.password=1234
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

#hibernate JPQL Setting
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.sql.init.mode=always
spring.jpa.defer-datasource-initialization=true

#secret key
#jwt.secret=firstSecretKey
logging.level.root=INFO
logging.level.org.hibernate.type=INFO

#spring.profiles.active=unit-test
```

Run the test and SQL show like this: n = 2 (because we set 3 employees in **2 department** (IT and Marketing)



```
Hibernate:
    select
        employee0_.id as id1_0_,
        employee0_.date_of_birth as date_of_2_0_,
        employee0_.dept_id as dept_id5_0_,
        employee0_.first_name as first_na3_0_,
        employee0_.gender as gender4_0_
    from
        com_employee employee0_
Hibernate:
    select
        department0_.id as id1_1_0_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_
    from
        demo_department department0_
    where
        department0_.id=?
Hibernate:
    select
        department0_.id as id1_1_0_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_
    from
        demo_department department0_
    where
        department0_.id=?
```

If I set Employee2 belong to Department “Admin”, n will be **3 as follows:**



```
void setup() {
    Department department1 = new Department( id: 1l, name: "IT", Location.HCM);
    Department department2 = new Department( id: 2l, name: "Marketing", Location.HCM);
    Department department3 = new Department( id: 3l, name: "Admin", Location.HCM);
    departmentRepository.save(department1);
    departmentRepository.save(department2);
    departmentRepository.save(department3);

    Employee employee1 = new Employee();
    employee1.setId(1);
    employee1.setDepartment(department1);
    employee1.setGender(Gender.MALE);
    employee1.setFirstName("Huy Nguyen");
    employee1.setDateOfBirth(LocalDate.of( year: 19
    employeeRepository.save(employee1);

    Employee employee2 = new Employee();
    employee2.setId(2);
    employee2.setDepartment(department3);
    employee2.setGender(Gender.MALE);
    employee2.setFirstName("Hoang Tran");
    employee2.setDateOfBirth(LocalDate.of( year: 19
    employeeRepository.save(employee2);

    Employee employee3 = new Employee();
    employee3.setId(3);
    employee3.setDepartment(department2);
    employee3.setGender(Gender.MALE);
    employee3.setFirstName("Dat Nguyen");
    employee3.setDateOfBirth(LocalDate.of( year: 19
    employeeRepository.save(employee3);

}
```

```
Hibernate:
    select
        employee0_.id as id1_0_,
        employee0_.date_of_birth as date_of_2_0_,
        employee0_.dept_id as dept_id5_0_,
        employee0_.first_name as first_na3_0_,
        employee0_.gender as gender4_0_
    from
        com_employee employee0_
Hibernate:
    select
        department0_.id as id1_1_0_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_
    from
        demo_department department0_
    where
        department0_.id=?
Hibernate:
    select
        department0_.id as id1_1_0_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_
    from
        demo_department department0_
    where
        department0_.id=?
Hibernate:
    select
        department0_.id as id1_1_0_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_
    from
        demo_department department0_
    where
        department0_.id=?
```

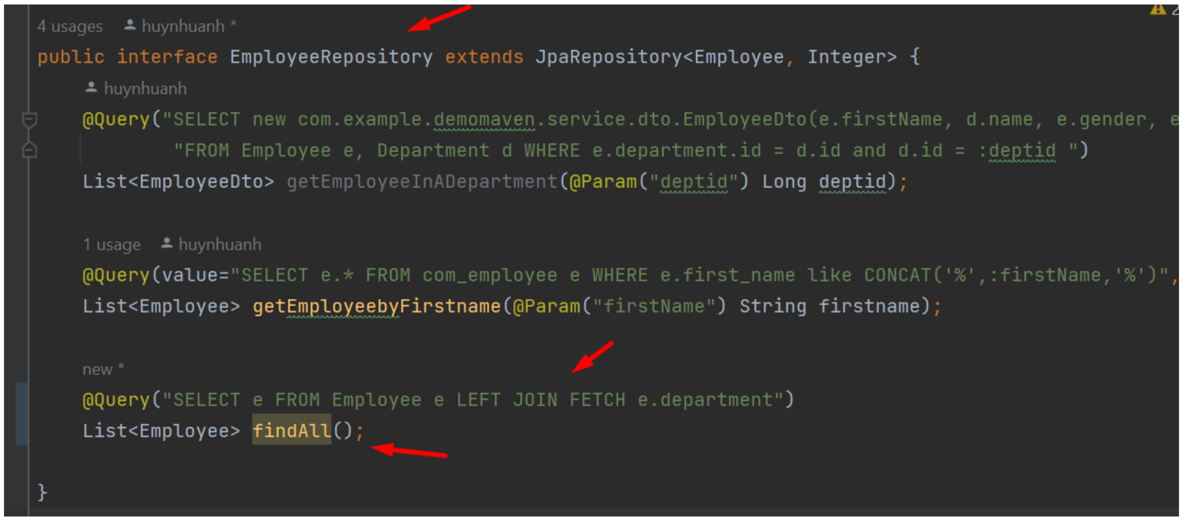
How to solve this Problem?

https://thorben-janssen.com/5-ways-to-initialize-lazy-relations-and-when-to-use-them/

1. Using **lazy load** not **eager fetch mode**; but we have problem with getDepartment() ➔ return null
2. Fetch join in JPQL or Criteria API
3. **Named Entity Graph** or **Dynamic Entity Graph** (> JPA 2.1 New features)

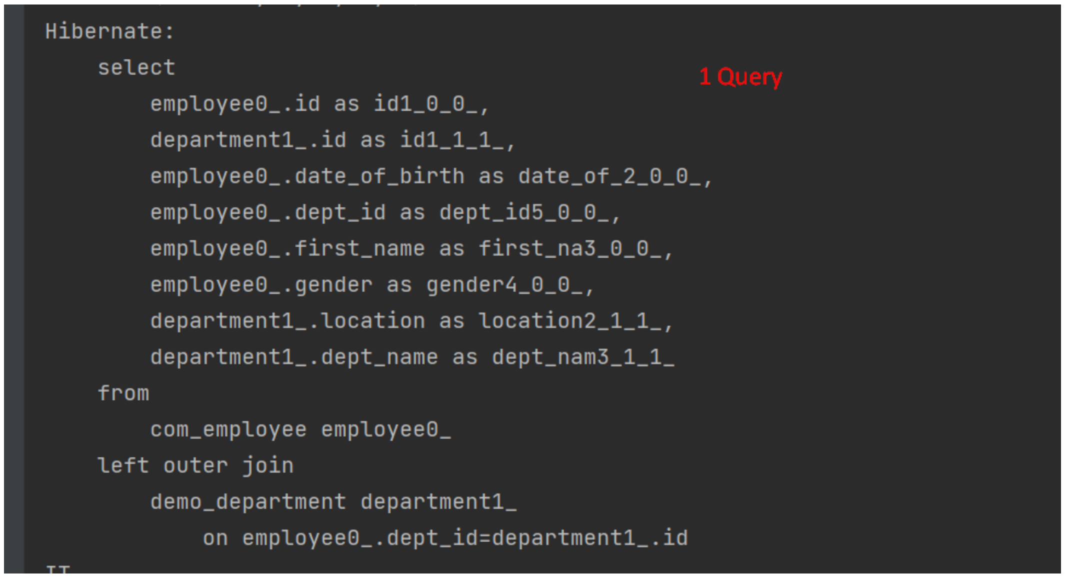
Part 1: demo using Fetch join in JPQL:

Step 1: rewrite the findAll function in EmployeeRepositorie:



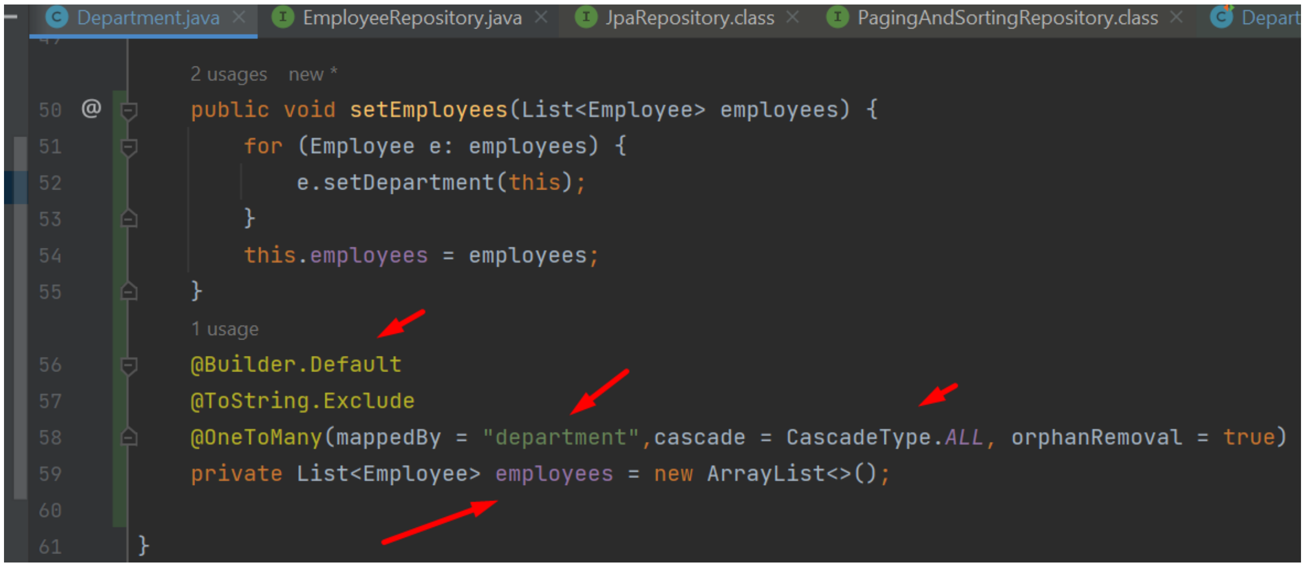
```
4 usages   huynhuanh *
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
    huynhuanh
    @Query("SELECT new com.example.demomaven.service.dto.EmployeeDto(e.firstName, d.name, e.gender, e
            "FROM Employee e, Department d WHERE e.department.id = d.id and d.id = :deptid ")
    List<EmployeeDto> getEmployeeInADepartment(@Param("deptid") Long deptid);

    1 usage   huynhuanh
    @Query(value="SELECT e.* FROM com_employee e WHERE e.first_name like CONCAT('%',:firstName,'%')",
    List<Employee> getEmployeebyFirstname(@Param("firstName") String firstname);

    new *
    @Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department")
    List<Employee> findAll();

}
```

Step 2: Run the unit test again to see the generated sql:



```
Hibernate:
    select
        employee0_.id as id1_0_0_,
        department1_.id as id1_1_1_,
        employee0_.date_of_birth as date_of_2_0_0_,
        employee0_.dept_id as dept_id5_0_0_,
        employee0_.first_name as first_na3_0_0_,
        employee0_.gender as gender4_0_0_,
        department1_.location as location2_1_1_,
        department1_.dept_name as dept_nam3_1_1_
    from
        com_employee employee0_
    left outer join
        demo_department department1_
            on employee0_.dept_id=department1_.id
```

Part 2: demo using "Named Entity Graph"

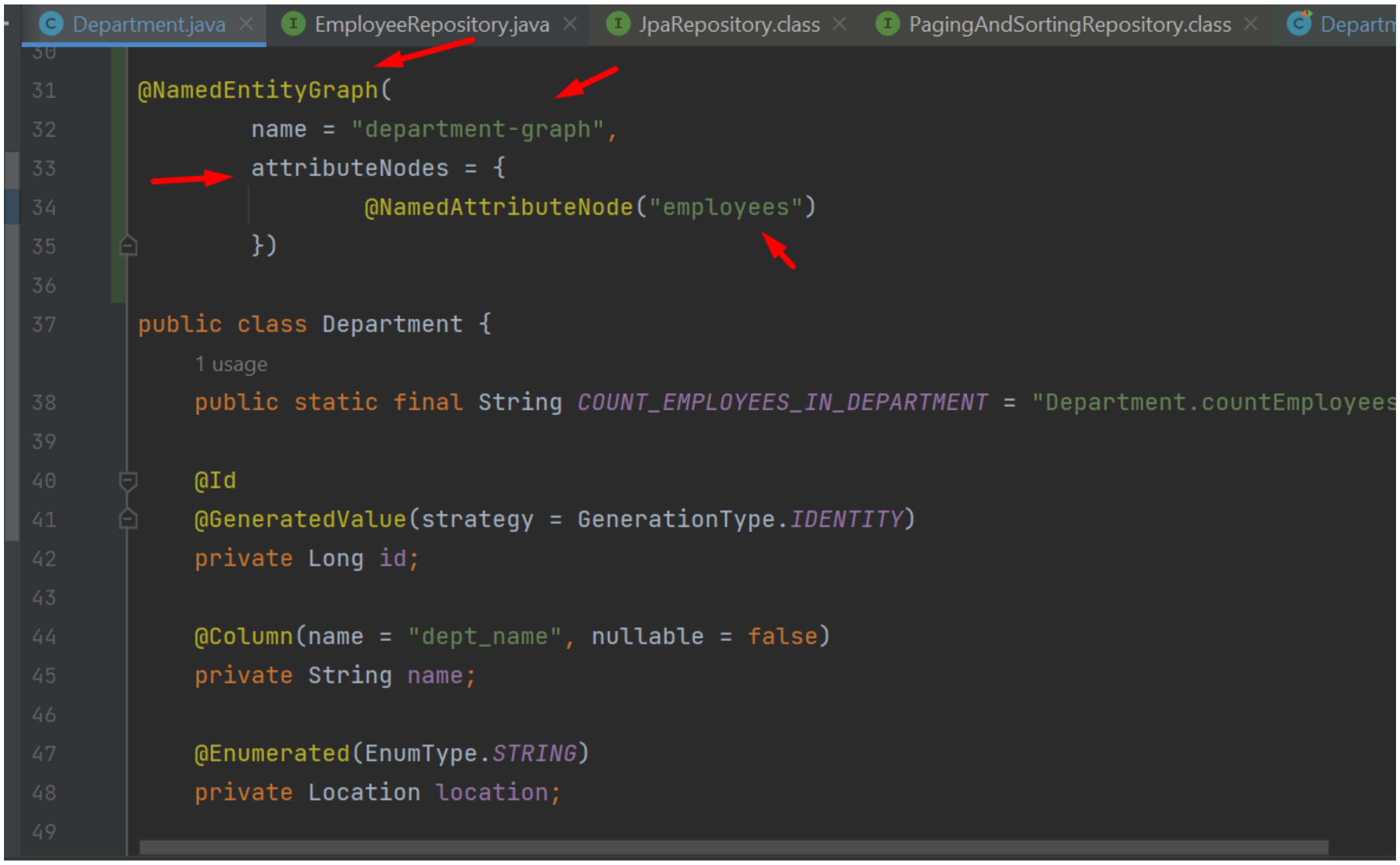


Department.java × EmployeeRepository.java × JpaRepository.class × PagingAndSortingRepository.class × Depart

```
    2 usages   new *
    public void setEmployees(List<Employee> employees) {
        for (Employee e: employees) {
            e.setDepartment(this);
        }
        this.employees = employees;
    }
    1 usage
    @Builder.Default
    @ToString.Exclude
    @OneToMany(mappedBy = "department",cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Employee> employees = new ArrayList<>();

}
```

In Department entity, we add bidirectional mapping with @OneToMany to employees.

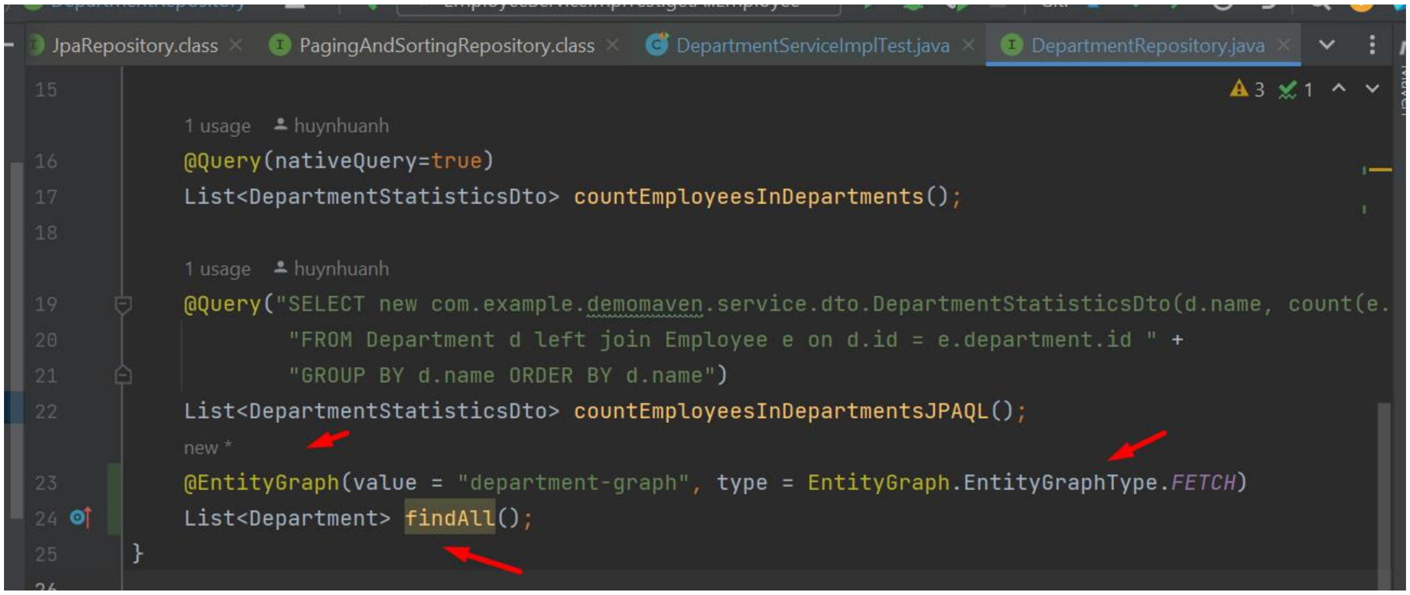
Then we declare Named Entity Graph and its attribute Nodes as follow:



Department.java × EmployeeRepository.java × JpaRepository.class × PagingAndSortingRepository.class × Departm

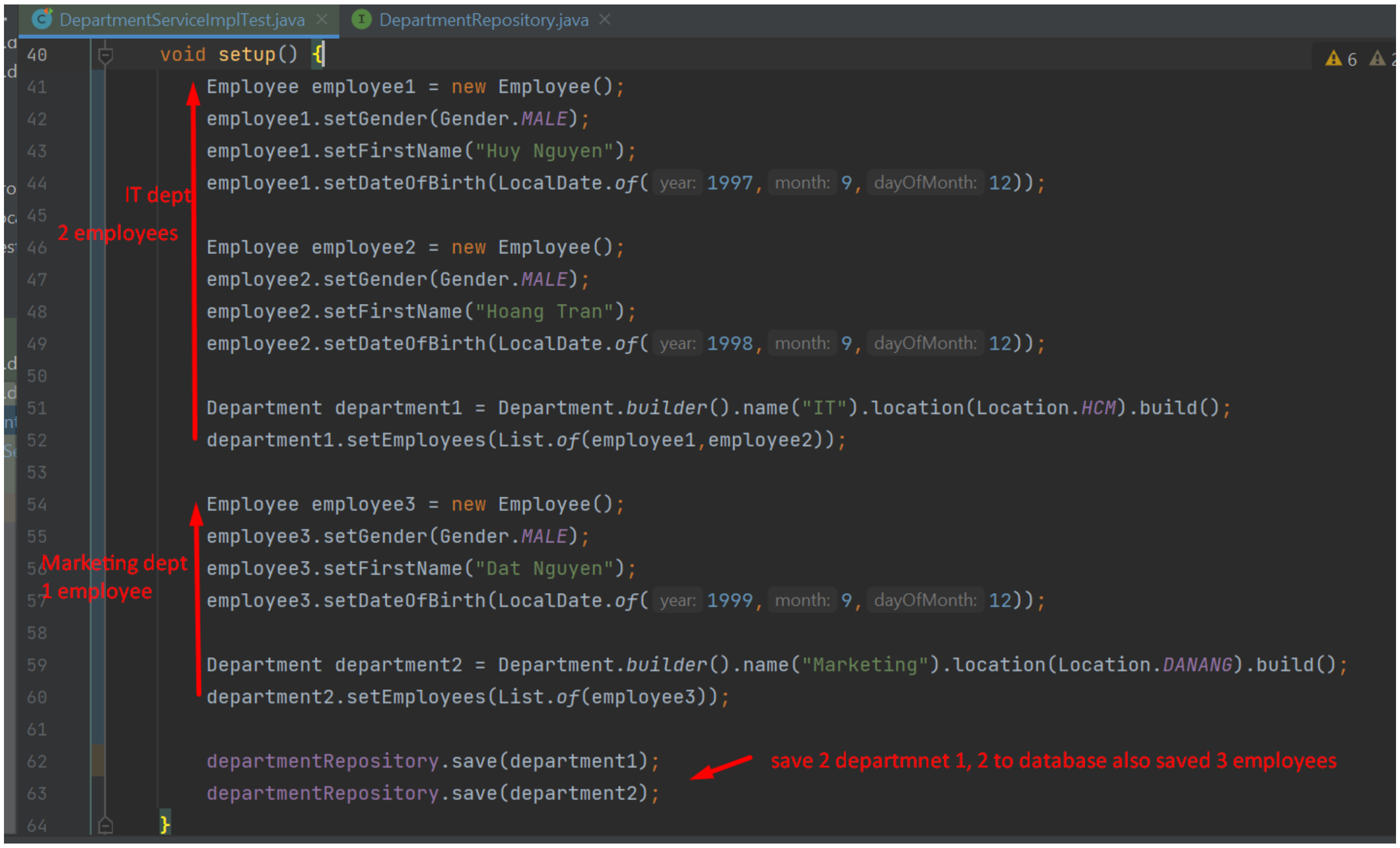
```
@NamedEntityGraph(
        name = "department-graph",
        attributeNodes = {
                @NamedAttributeNode("employees")
        })

public class Department {
    1 usage
    public static final String COUNT_EMPLOYEES_IN_DEPARTMENT = "Department.countEmployees

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "dept_name", nullable = false)
    private String name;

    @Enumerated(EnumType.STRING)
    private Location location;

```

In the Department Repositories, we override findAll method as follows:


JpaRepository.class
PagingAndSortingRepository.class
DepartmentServiceImplTest.java
DepartmentRepository.java

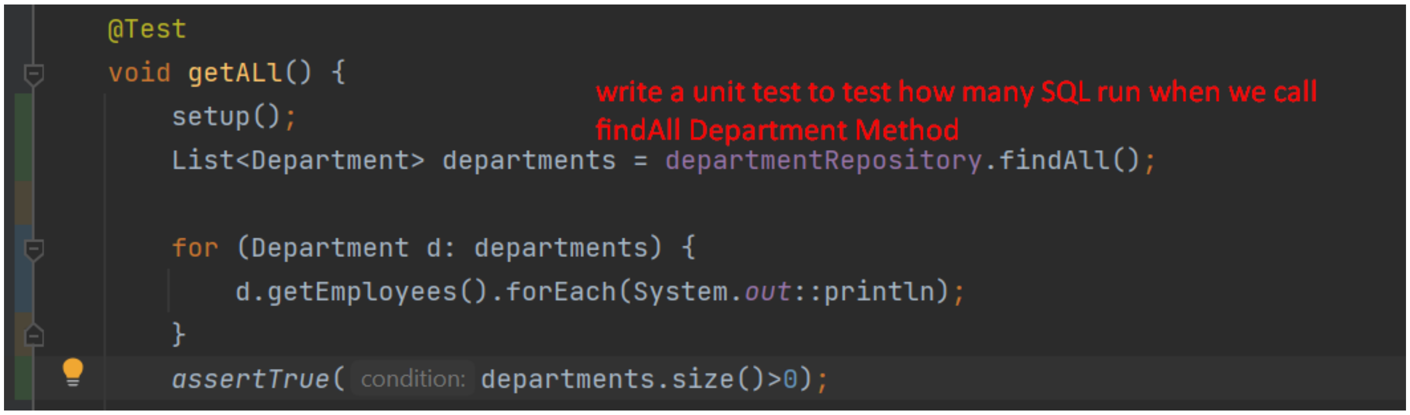
```
    1 usage  huynhuanh
    @Query(nativeQuery=true)
    List<DepartmentStatisticsDto> countEmployeesInDepartments();

    1 usage  huynhuanh
    @Query("SELECT new com.example.demomaven.service.dto.DepartmentStatisticsDto(d.name, count(e.
            "FROM Department d left join Employee e on d.id = e.department.id " +
            "GROUP BY d.name ORDER BY d.name")
    List<DepartmentStatisticsDto> countEmployeesInDepartmentsJPAQL();
    new *
    @EntityGraph(value = "department-graph", type = EntityGraph.EntityGraphType.FETCH)
    List<Department> findAll();
}
```

Setup method will create 2 departments as well as employees in them.



```
    void setup() {
        Employee employee1 = new Employee();
        employee1.setGender(Gender.MALE);
        employee1.setFirstName("Huy Nguyen");
        employee1.setDateOfBirth(LocalDate.of( year: 1997, month: 9, dayOfMonth: 12));

        Employee employee2 = new Employee();
        employee2.setGender(Gender.MALE);
        employee2.setFirstName("Hoang Tran");
        employee2.setDateOfBirth(LocalDate.of( year: 1998, month: 9, dayOfMonth: 12));

        Department department1 = Department.builder().name("IT").location(Location.HCM).build();
        department1.setEmployees(List.of(employee1,employee2));

        Employee employee3 = new Employee();
        employee3.setGender(Gender.MALE);
        employee3.setFirstName("Dat Nguyen");
        employee3.setDateOfBirth(LocalDate.of( year: 1999, month: 9, dayOfMonth: 12));

        Department department2 = Department.builder().name("Marketing").location(Location.DANANG).build();
        department2.setEmployees(List.of(employee3));

        departmentRepository.save(department1);
        departmentRepository.save(department2);
    }
```

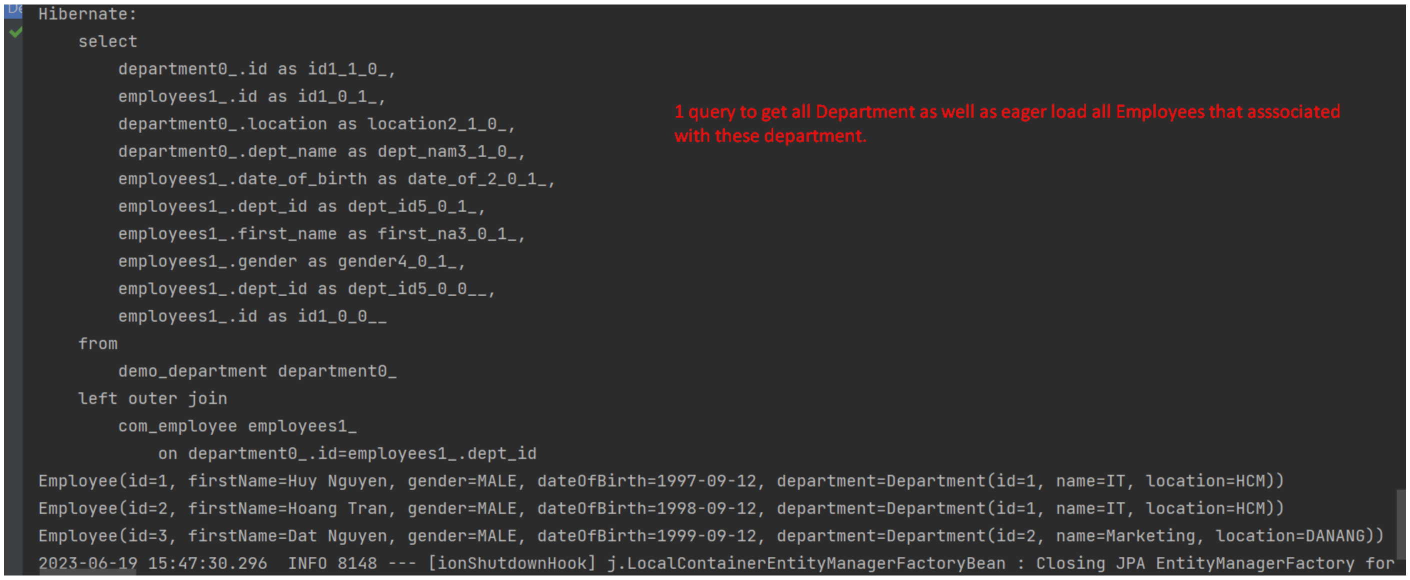
save 2 departmnet 1, 2 to database also saved 3 employees



```java
@Test
void getALL() {
    setup();
    List<Department> departments = departmentRepository.findAll();

    for (Department d: departments) {
        d.getEmployees().forEach(System.out::println);
    }
    assertTrue( condition: departments.size()>0);
```

write a unit test to test how many SQL run when we call findAll Department Method

And console log will show that only 1 query run as following picture:



```
Hibernate:
    select
        department0_.id as id1_1_0_,
        employees1_.id as id1_0_1_,
        department0_.location as location2_1_0_,
        department0_.dept_name as dept_nam3_1_0_,
        employees1_.date_of_birth as date_of_2_0_1_,
        employees1_.dept_id as dept_id5_0_1_,
        employees1_.first_name as first_na3_0_1_,
        employees1_.gender as gender4_0_1_,
        employees1_.dept_id as dept_id5_0_0__,
        employees1_.id as id1_0_0__
    from
        demo_department department0_
    left outer join
        com_employee employees1_
            on department0_.id=employees1_.dept_id
Employee(id=1, firstName=Huy Nguyen, gender=MALE, dateOfBirth=1997-09-12, department=Department(id=1, name=IT, location=HCM))
Employee(id=2, firstName=Hoang Tran, gender=MALE, dateOfBirth=1998-09-12, department=Department(id=1, name=IT, location=HCM))
Employee(id=3, firstName=Dat Nguyen, gender=MALE, dateOfBirth=1999-09-12, department=Department(id=2, name=Marketing, location=DANANG))
2023-06-19 15:47:30.296  INFO 8148 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for
```

1 query to get all Department as well as eager load all Employees that asssociated with these department.

=================Thank You =================