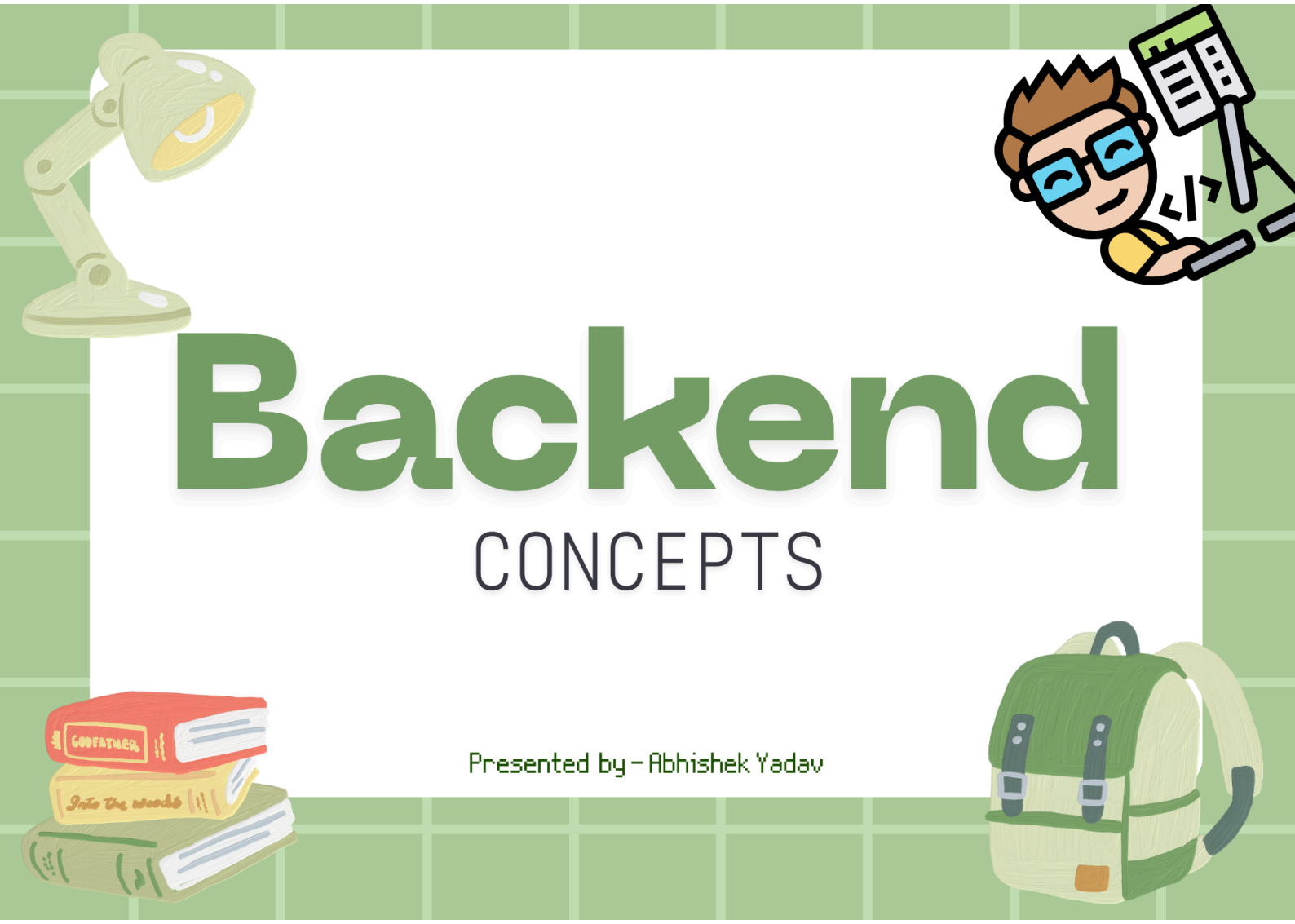

N+1 Problem

I had a confusion that n+1 is only caused by lazy loading but when I dig deep. this is what I found.

✅ **N+1 Problem**: Primarily caused by **lazy loading**

But **can also** happen with **eager loading**, **depending on how the data is accessed**.

🔍 What is the N+1 Problem?

When you fetch a **list of parent entities**, and for each parent, a **separate query** is fired to fetch its associated child entities — you end up with:

- 1 query to fetch all parents (N)

- +1 query **per parent** to fetch its children
  ➡️ So total = **N+1 queries**

🟡 Lazy Loading Causes N+1 Problem (Most Common)

List<Department> departments = departmentRepo.findAll(); // 1 query

for (Department dept : departments) {

  System.out.println(dept.getEmployees().size()); // triggers 1 query per dept

}

Explanation: The getEmployees() call lazily fetches employees for each department individually — causing **N additional queries**.

🔴 Eager Loading **Can Also** Cause N+1 Problem

If you set associations as EAGER by default, like:

@OneToMany(fetch = FetchType.EAGER)

private List<Employee> employees;

And you do:

List<Department> departments = departmentRepo.findAll(); // seems like 1 query

☠️ Hibernate will **still trigger N+1 queries** *unless you use a JOIN FETCH* — because EAGER doesn't always mean joined in **1 SQL** query — it can **load eagerly via multiple queries** (using batch fetching or subselects depending on config).

✅ How to Avoid N+1?

- Use JOIN FETCH (JPQL)
- Use @EntityGraph (Spring Data JPA)
- Use @BatchSize (Hibernate batching)
- Avoid default EAGER on collections

✅ Conclusion

Loading Type	Can cause N+1?	Why?
Lazy	✅ Most common	Data is loaded on access inside loops
Eager	✅ Sometimes	If no join is used, it loads eagerly but via separate queries

✅ N+1 is not just about *when* data is loaded, but also *how many queries* are fired — and that can happen with both LAZY and EAGER.