7/27 ~ 7/31 4주차 - 1

궁금한것 정리하기.

자바 14부터 새로 생긴것

- 런타임에서 객체 타입을 확인하는 instanceof 연산자가 조금 강화되었다.

- 새로운 패키징 도구가 도입되었다
- 이 인큐베이팅 기능은 자바 애플리케이션에서 플랫폼 별 패키지를 작성하기 위한 새로운 도구인 `jpackage`를 제공한다이는 리눅스(linux)의 `deb` 또는 `rpm` 파일, macOS의 `pkg` 또는 `dmg` 파일 그리고 Windows의 `msi` 또는 `exe` 파일을 의미한다.

Helpful NullPointerExceptions

`NullPointerException`이 발생하면 코드 라인 넘버를 보고 어디가 문제인지 유추해야한다. 라인 넘버만 나오기 때문에, 정확한 이유를 유추해야 하는 불편함이 있었는데, 이 부분이 조금 강화되었다.

```
private void runJEP358Test() {
    // 이런 메서드를 호출하는데, `b`의 리턴이 `null` 이면?
    a().b().c();
}
```

Java

위 메서드에서 만일 `b` 메서드의 리턴이 `null`이면 어떻게 될까? 기존 버전의 자바에서는 아래와 같은 메시지가 출력됐을 것이다.

```
# 이전 버전에서는 아래와 같이 출력되었다. 대충 이런 모습...
Exception in thread "main" java.lang.NullPointerException
    at MadPlay.runJEP358Test(MadPlay.java:43)
    at MadPlay.main(MadPlay.java:117)
```

Bash

그런데, 자바 14버전부터는 실행 옵션에 `-XX:+ShowCodeDetailsInExceptionMessages` 넣어주면, 아래와 같이 `NPE` 메시지가 바뀐다.

`Exception in thread "main" java.lang.NullPointerException: Cannot invoke "MadPlay.c()" because the return value of "MadPlay.b()" is null at MadPlay.runJEP358Test(MadPlay.java:43) at MadPlay.main(MadPlay.java:1317)`

a. 가능한 짧은 시간의 쿼리 index를 사용할 것

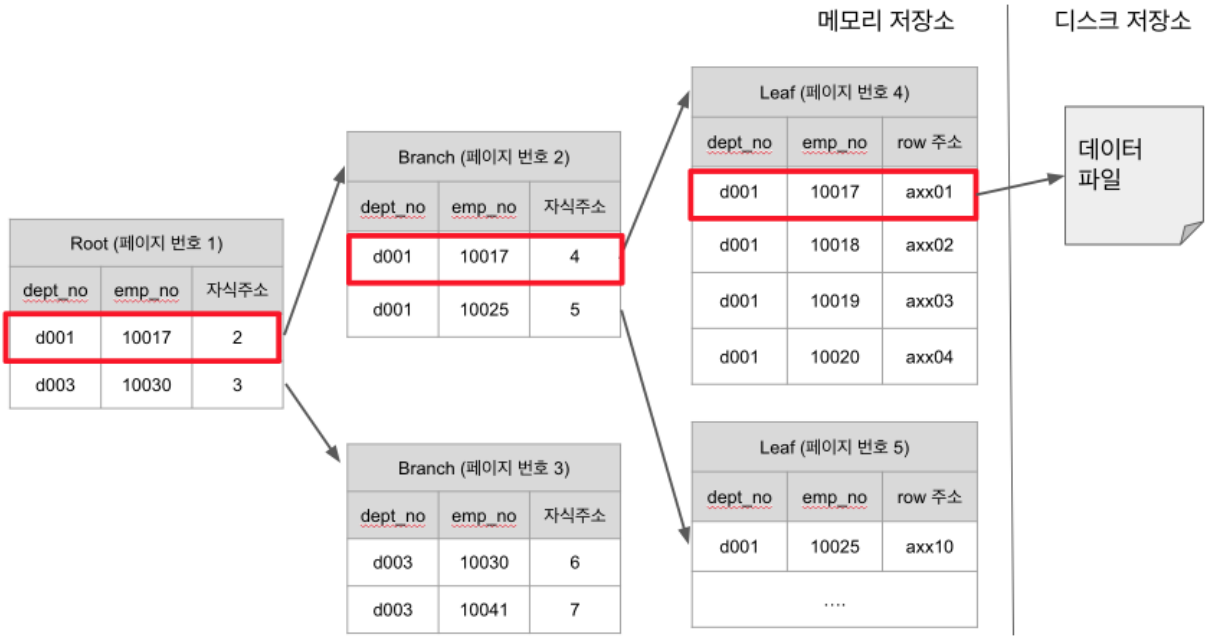
b. 데이터를 불러올 경우 역시 적은 호출만으로 가능할 것

인덱스에 대한 궁금증

> 인덱스 == 정렬 이다.

- 인덱스는 결국 지정한 컬럼들을 기준으로 메모리 영역에 일종의 **목차**를 생성하는 것이다.
- insert, update, delete (Command)의 성능을 희생하고 대신 **select (Query)의 성능을 향상**시킨다.
- 여기서 주의할 것은 update, delete 행위가 느린것이지, **update, delete를 하기 위해 해당 데이터를 조회하는것은 인덱스가 있으면 빠르게 조회**가 됩니다.

- 인덱스가 없는 컬럼을 조건으로 update, delete를 하게 되면 굉장히 느려 많은 양의 데이터를 삭제 해야하는 상황에선 인덱스로 지정된 컬럼을 기준으로 진행하는것을 추천한다.



(B-Tree 인덱스 구조)

인덱스 탐색은 Root → Branch → Leaf → 디스크 저장소 순으로 진행된다.

인덱스 컬럼 기준

인덱스에 사용되는 컬럼은 카디널리티(Cardinality)가 가장 높은 것을 잡아야 한다.

> 카디널리티(Cardinality)란 해당 컬럼의 중복된 수치를 나타낸다.

> 예를 들어 성별, 학년 등은 카디널리티가 낮다.

> 반대로 주민등록번호, 계좌번호 등은 카디널리티가 높다.

인덱스로 최대한 효율을 뽑아내려면, 해당 인덱스로 많은 부분을 걸러내야 한다.

만약 성별을 인덱스로 잡는다면, 남/녀 중 하나를 선택하기 때문에 인덱스를 통해 50%밖에 걸러내지 못한다.

하지만 주민등록번호나 계좌번호 같은 경우엔 **인덱스를 통해 데이터의 대부분을 걸러내기 때문에** 빠르게 검색이 가능하다.