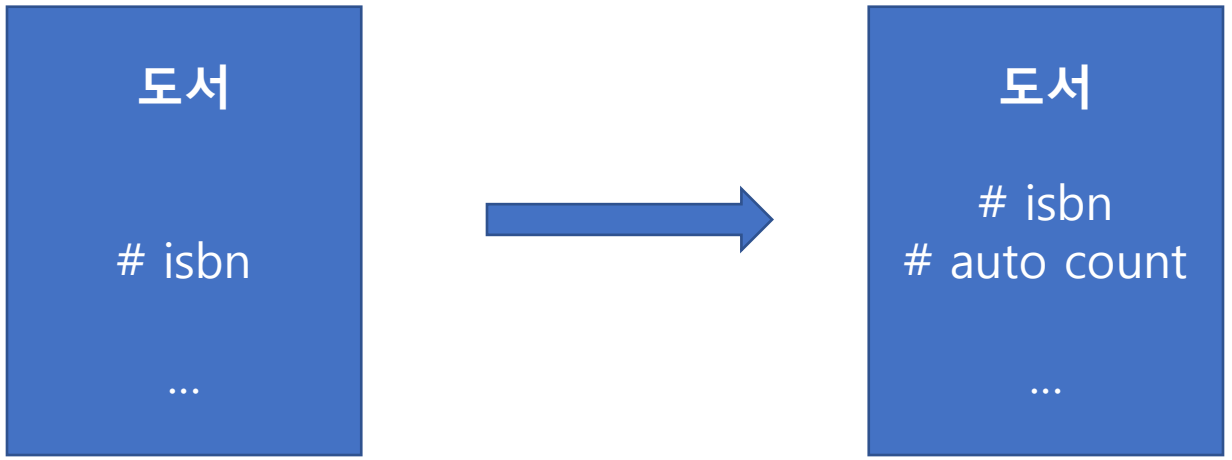
추가 과제 피드백 정리

1. 정규화와 역정규화

1) 1차 정규화를 한다면...

* '도서'엔터티에서 완전 같은 책이지만 ISBN이 다른 케이스를 생각할 것 – PK가 구분 되는 값이 아님.

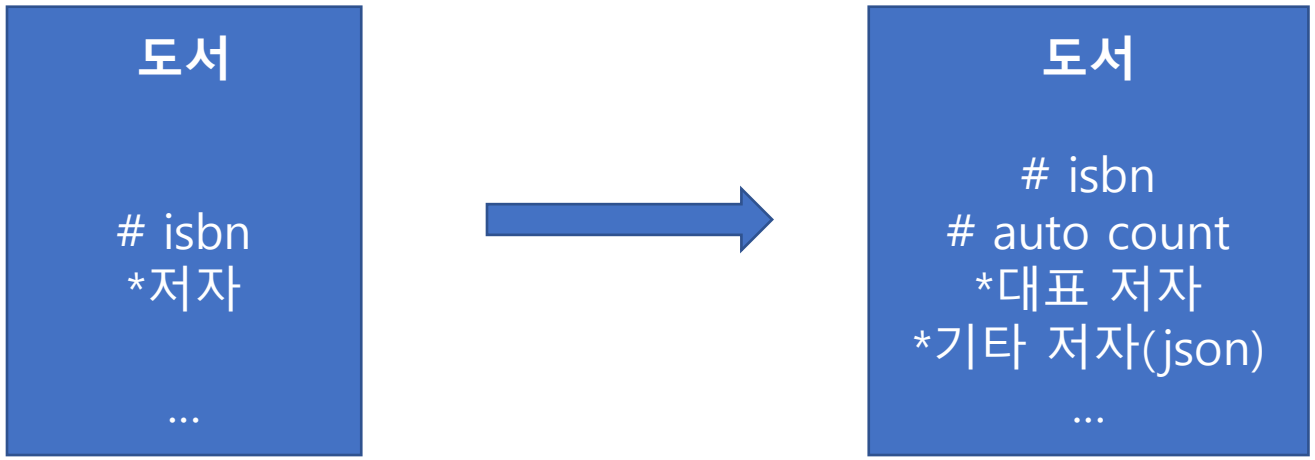
* 해결 : [ISBN과 순번]을 PK로 만들기.



1. 정규화와 역정규화

2) 2차 정규화를 한다면...

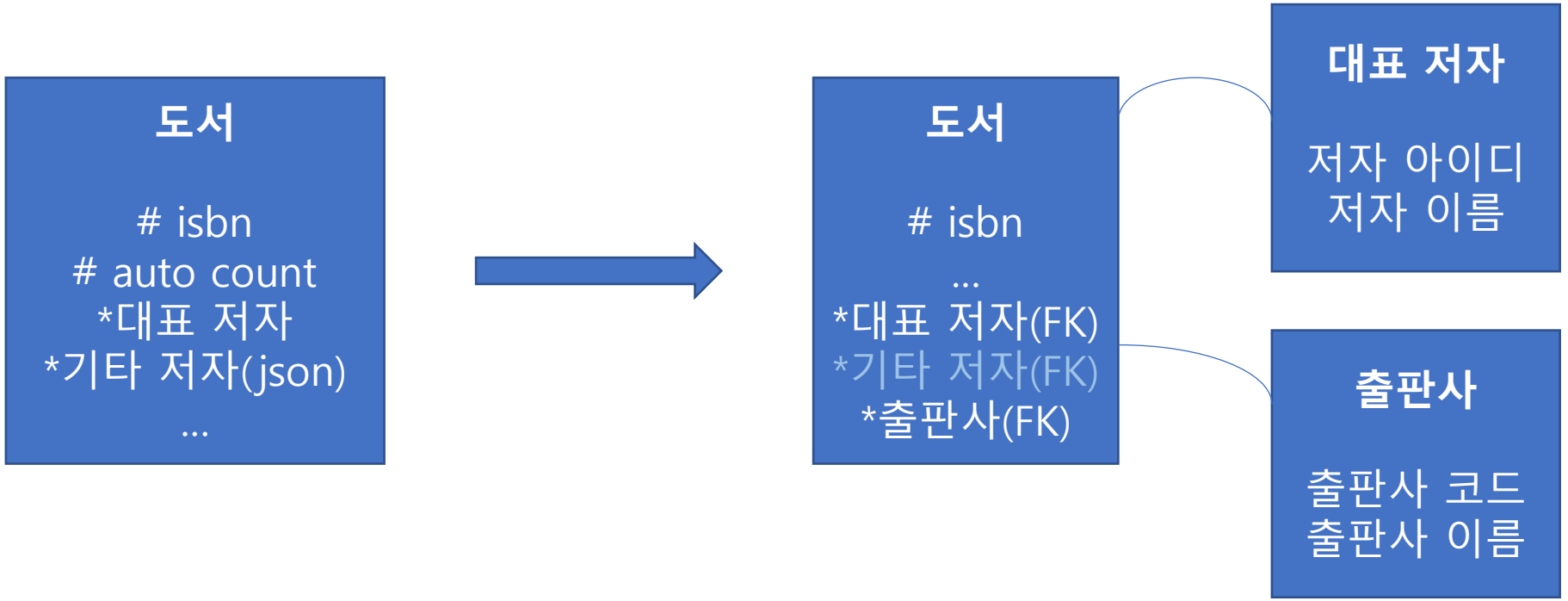
- 2차 정규화 : 쉽게 생각해서 PK에 대해서만 각각의 column들이 값을 1개씩 가질지 생각하기
- 한 책에 대하여, 저자의 이름은 여러 개가 있을 수 있음
  -> '대표 저자'(필수)와 나머지 저자를 '기타저자'(선택)(json파일 : 여러 명)를 나누기



1. 정규화와 역정규화

3) 3차 정규화를 한다면...

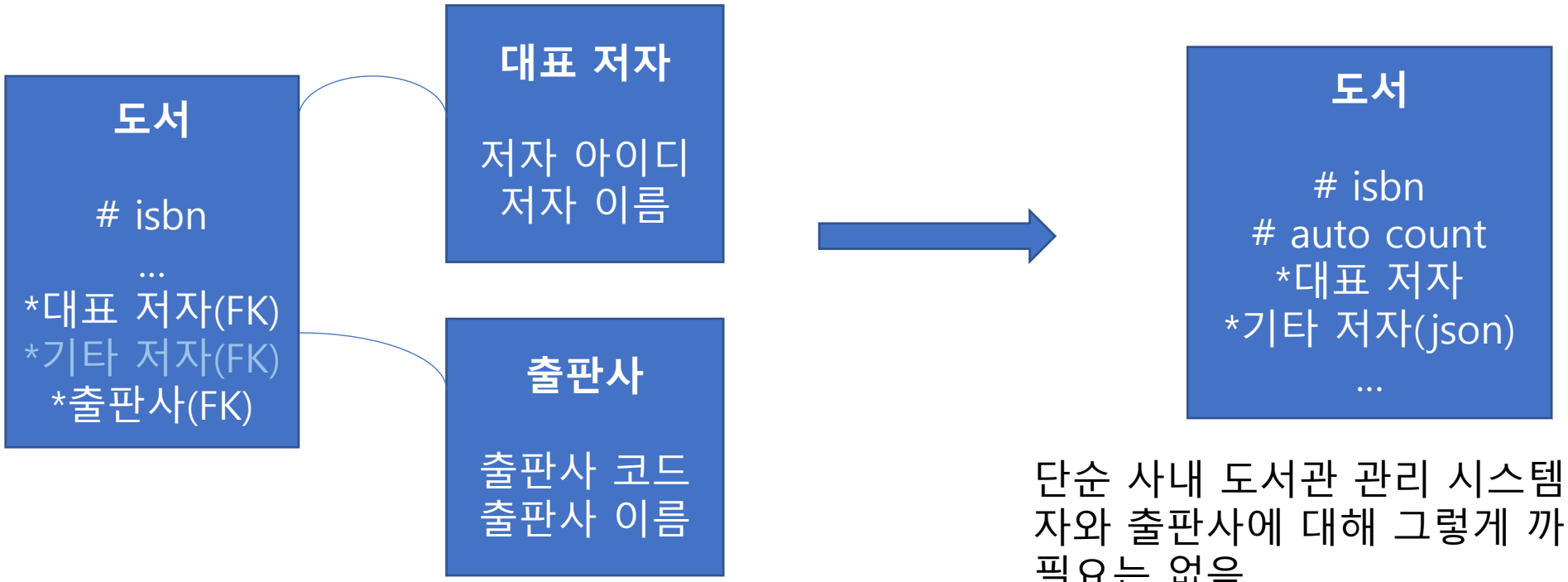
- 모든 column들끼리 서로 1개의 값만을 가지는 지 확인하기
- 대표 저자와 출판사는 1:1일까? No – 한 저자가 쓴 책이 여러 출판사가 있을 수 있음
  -> '대표저자' 및 '출판사'를 하나의 엔티티로 만들고, '도서'에 외래키로 있기



1. 정규화와 역정규화

But) 역정규화??

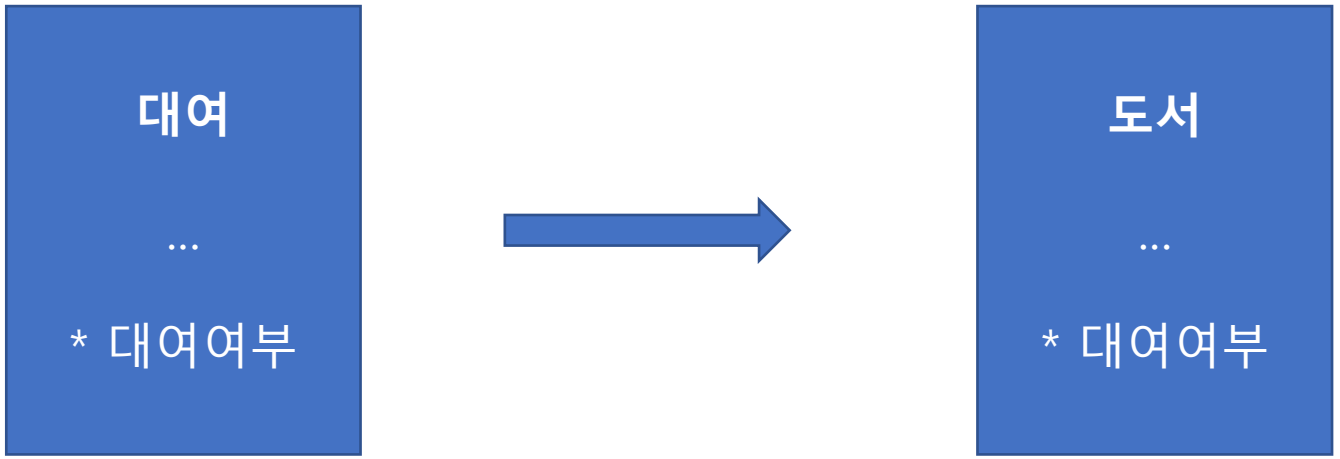
- 정규화 되어 있는 것을 일부러 정규화 되지 않은 상태로 만드는 것
- 정규화는 이론이고, 실제로는 그렇게 까지 할 필요가 없는 경우가 대다수이다.
- 역정규화는 DA의 꽃 : 기계가 할 수 있는 영역이 아님



단순 사내 도서관 관리 시스템이고, 저자와 출판사에 대해 그렇게 까지 나눌 필요는 없음

1. 정규화와 역정규화

또다른 역정규화



정규화에 따르자면, 대여와 관련있는 대여여부는 대여 엔터티에 있는 것이 맞겠지만, 실제로는 대여 엔터티는 대여를 한 후 생기는 것이므로, 대여여부는 항상 true가 되고, 의미가 없어진다. 또한 다른 것과 연산을 하려고 해도 너무나 어려워짐

대여 엔터티의 대여여부를 도서 엔터티의 대여여부로 옮기기
대여 여부를 해당 도서가 대여가 되었는지 확인을 하는 것이 사용자 관점에서 더 옳다.

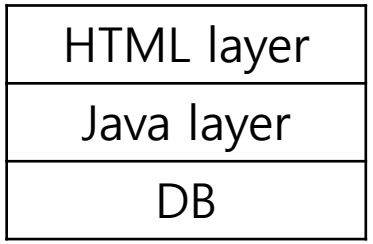
2. trigger에 대하여

실제 db설계 전에 DB와 개발소스를 분류한 것이 조금 더 현업과 가깝다.

즉, trigger는 아예 안써도 무방한 것임.
이 시스템은 사람이 많은 시스템은 아니지만, 만약 사람이 많다고 생각한다면, 내가 설계한 대로 trigger가 있을 때, 비싼 DB에 오랫동안 머무르는 시간이 많아지고, 전체 프로그램에 대한 속도 저하가 온다.
따라서, 사용자의 연체 여부 및 빌린 책 수는 DB내의 trigger로 하기 보다 Java등의 프로그램에서 다루는 것이 더 나음.

MVCC이론) 처리하는 layer를 여러 개 두는 이론

- 약 3년전에 유행하던 이론 (cf : 10년전 – 객체지향 유행, 현재 – 컨테이너,클라우드 유행)
- 한 곳에서 다 처리하는 것이 아니라 여러 layer를 두고 처리한다는 것
- 다만, DB의 경우, layer로 보기보다 그냥 DB로 보는 것이 좋음 – 복잡한 데이터 처리는 비용이 적고, 수정도 쉬운 더 윗단(Java등)에서 관리하는 것이 훨씬 좋음.



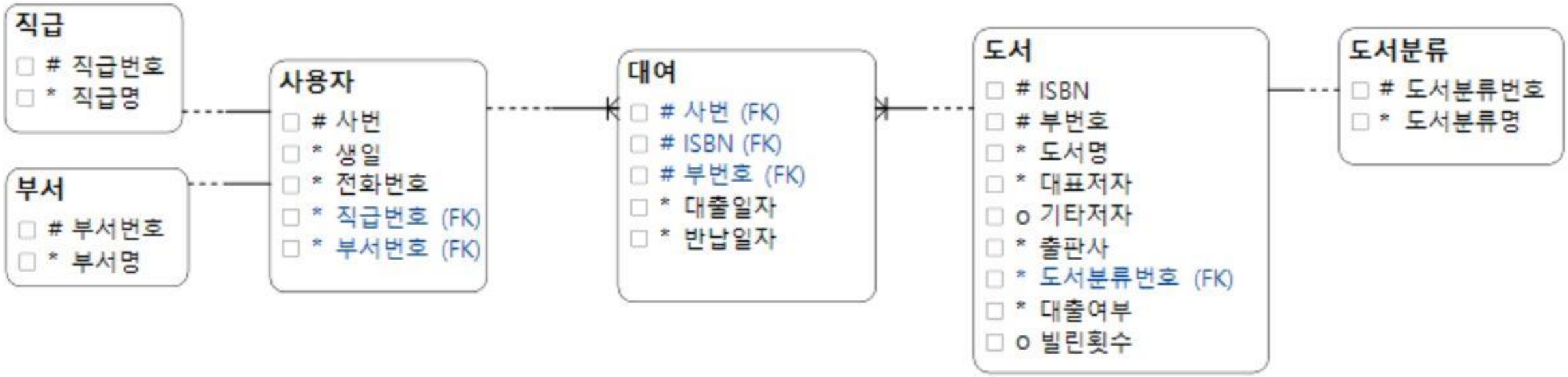
2. trigger에 대하여

2번째 trigger를 사용하면 좋지 않은 점은 실제 일하는 것을 시뮬레이션하는 것과 관련있다.

실제 일을 할 때에는 여러 부서가 한다. 어떠한 것이 바뀔 때, 가장 안쪽 단이라고 할 수 있는 DB까지 바꾸기에는 부서간 충돌도 많을 것이며, 매번 귀찮을 것이다. 따라서, 변경 사항이 있다면 바로바로 수정할 수 있는 윗단에 개발 소스로 남겨 놓는 것이 좋다.

피드백 적용 DB설계

Trigger있는 부분 – 모두 지워버림(개발 소스에서 처리할 것)
도서부분에 PK키를 늘리고, 저자의 경우 대표 저자와 기타 저자로 나눔 하지만 제3정규화까지 가지 않음



피드백 적용 DB설계

Trigger있는 부분 – 모두 지워버림(개발 소스에서 처리할 것)
도서부분에 PK키를 늘리고, 저자의 경우 대표 저자와 기타 저자로 나눔 하지만 제3정규화까지 가지 않음