데이터베이스 - 정규화

주차	22주차
스터디 일자	@2024/03/29

데이터베이스 정규화란?

정규화는 이상현상이 있는 릴레이션을 분해하여 이상현상을 없애는 과정이다.

이상현상이 존재하는 릴레이션을 분해하여 여러 개의 릴레이션을 생성하게 된다.

이를 단계별로 구분하여 정규형이 높아질수록 이상현상은 줄어들게 된다.

릴레이션이란?

릴레이션이란 **관계형 데이터베이스에서 정보를 구분하여 저장하는 기본 단위이다.**

결국, 릴레이션은 **DB 테이블이다.**

이상현상이란?

💡 이상현상(Anomaly)의 종류

`삭제 이상` : 데이터 삭제 시 의도와는 상관없이 다른 정보까지 연쇄적으로 삭제되는 현상

`삽입 이상` : 데이터 삽입 시 의도와는 상관없이 원하지 않는 값들도 함께 삽입되는 현상

`수정 이상` : 데이터 수정 시 의도와는 상관없이 데이터의 일부만 수정되어 일어나는 데이터 불일치 현상

- 해당 테이블은 정규화를 수행하지 않은 것

- 학과 테이블과 학생 테이블을 하나로 합친 테이블

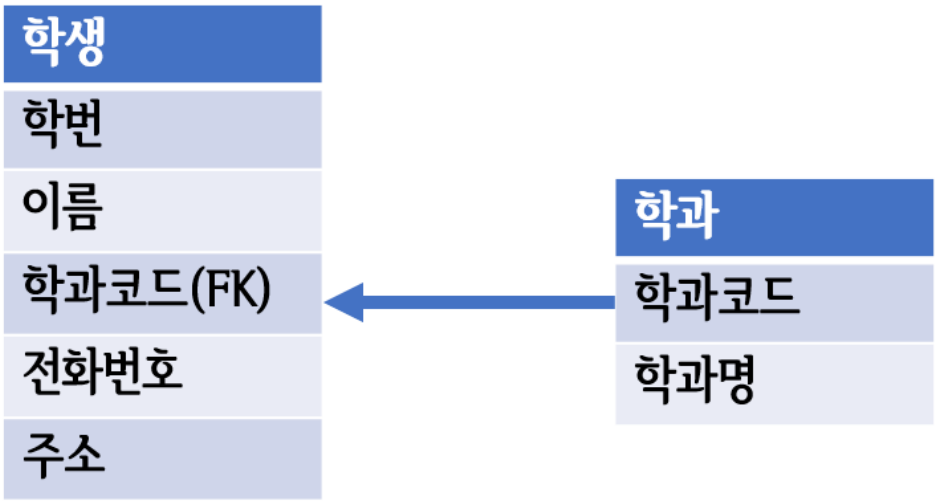
- 만약 위의 테이블에서 **새로운 학생이 한명 추가될 때 필요한 학과코드가 없다면** 임의의 값을 넣어줘야 한다.

  ⇒ 즉, 불필요한 정보가 함께 추가되는 것이다.

  반대로 '컴퓨터공학과'라는 학과를 추가할 때는 학생이 없기 때문에 임의의 값으로 학생의 정보들을 입력해야 한다.

  이러한 문제를 **이상현상(Anomlay)**이라고 부른다~!

이러한 현상을 해결하기 위해서는 `정규화`를 수행해서 테이블을 관리해주어야 한다.



- 이렇게 테이블이 분해되면 학생과 학과 테이블간에 불필요한 데이터를 입력해도 되지 않으므로

  중복 데이터가 제거되고 필요한 경우, 학과코드로 **조인(Join)**을 수행하여 하나의 합집합으로 만들어 사용할 수도 있다.

정규화의 장점, 단점

장점

- 데이터베이스 변경 시 이상현상을 제거할 수 있다.
- 정규화된 데이터베이스 구조에서는 새로운 데이터 형의 추가로 인한 확장 시, 그 구조를 변경하지 않아도 되거나 일부만 변경해도 된다.
- 데이터베이스와 연동된 응용 프로그램에 최소한의 영향만을 미치게 되어 응용프로그램의 생명을 연장시킨다.

단점

- 릴레이션의 분해로 인해 릴레이션 간의 JOIN 연산이 많아진다.

  → 만약 조인이 많이 발생하여 성능 저하가 나타나면 **반정규화**를 적용할 수도 있다.
- 질의에 대한 응답 시간이 느려질 수도 있다.
  - 데이터의 중복 속성을 제거하고 결정자에 의해 동일한 의미의 일반 속성이 하나의 테이블로 집약되므로 한 테이블의 데이터 용량이 최소화되는 효과가 있다.

    ⇒ 따라서 데이터를 처리할 때 속도가 빨라질 수도 있고 느려질 수도 있다.

단계별 정규화

제1 정규형 `1NF`

아래와 같은 규칙들을 만족하면 **제1 정규형을 만족하는 상태**이다.

1️⃣ 1. 각 컬럼이 **하나의 속성만**을 가져야 한다.

2. 하나의 컬럼은
같은 종류나 **타입**의 값을 가져야 한다.

3. 각 컬럼이
유일한 이름을 가져야 한다.

4. 컬럼의
순서가 상관없어야 한다.

제1 정규화 예시

학생번호	이름	과목
101	아이유	운영체제, DB
102	한지민	자바
103	한효주	C, C++

제1 정규화가 필요한 테이블

1. 각 컬럼이 하나의 값(속성)만을 가져야 한다. → ❌ `하나의 칼럼(과목)에 두 개의 값을 가짐`
2. 하나의 컬럼은 같은 종류나 타입의 값을 가져야 한다. → ✅
3. 각 컬럼이 유일한 이름을 가져야 한다. → ✅
4. 컬럼의 순서가 상관없어야 한다. → ✅

1번 규칙을 불만족하기 때문에, 아래와 같이 분해해주어야 한다.

학생번호	이름	과목
101	아이유	운영체제
101	아이유	DB
102	한지민	자바
103	한효주	C
103	한효주	C++

1NF 만족 상태

제2 정규형 2NF

아래와 같은 규칙들을 만족하면 **제2 정규형을 만족하는 상태**이다.

2️⃣

1. **제1 정규형**을 만족해야 한다.

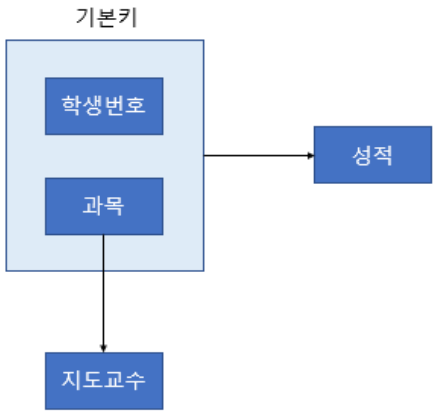
2. 모든 컬럼이
부분적 종속이 없어야 한다.
즉,
모든 컬럼이 **완전 함수 종속**을 만족해야 한다.

부분적 종속이란 기본키 중에 특정 컬럼에만 종속되는 것이다.

완전 함수 종속이란 기본키의 부분집합이 결정자가 되어선 안된다는 것이다.

제2 정규화 예시

학생번호	과목	지도교수	성적
101	운영체제	김운체	100
101	DB	조디비	60
102	자바	박자바	70
103	C	김씨	80
103	C++	이씨플	90



제2 정규화가 필요한 테이블

- 위 테이블의 기본키는 `학생번호`, `과목` 으로 복합키이다.
- 성적의 특정 값을 알기 위해서는 학생번호 + 과목이 있어야 한다.
- 하지만, 특정 과목의 지도교수는 과목명만 알면 **지도교수가 누구인지 알 수 있다.**

  ⇒ 지도 교수 컬럼이 `학생번호`, `과목` 에 종속되지 않고 `과목` 에만 종속되는 **부분적 종속**이다!

따라서, 제2 정규형을 만족하기 위해 아래와 같이 테이블을 분해해야 한다.

학생번호	과목	성적
101	운영체제	100
101	DB	60
102	자바	70
103	C	80
103	C++	90

과목	지도교수
운영체제	김운체
DB	조디비
자바	박자바
C	김씨
C++	이씨플

2NF 만족 상태

제3 정규형 3NF

아래와 같은 규칙들을 만족하면 **제3 정규형을 만족하는 상태**이다.

3

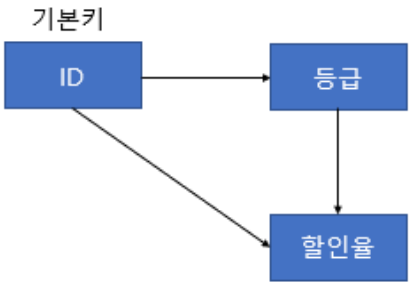
1. **제2 정규형**을 만족해야 한다.

2. 기본키를 제외한 속성들 간의
이행 종속성이 없어야 한다.

이행 종속성이란 `A → B`, `B → C` 일 때 `A → C` 가 성립하면 **이행 종속**이라고 한다.

제3 정규화 예시

ID	등급	할인율
101	Vip	40%
102	Gold	20%
103	Bronze	10%



제3 정규화가 필요한 테이블

- 해당 테이블에서 `ID(A)` 를 알면 `등급(B)` 을 알 수 있다.

  → `등급(B)` 을 알면 `할인율(C)` 을 알 수 있다.

  → 따라서 `ID(A)` 를 알면 `할인율(C)` 을 알 수 있다.

⇒ 따라서 이행 종속성이 존재하므로 제3 정규형을 만족하지 않는다.

제3 정규형을 만족하기 위해서는 아래와 같이 테이블을 분해해야 한다.

ID	등급
101	Vip
102	Gold
103	Bronze

등급	할인율
Vip	40%
Gold	20%
Bronze	10%

기본키

3NF 만족 상태

BCNF `Boyce-Codd Normal Form`

BCNF는 제3 정규형을 좀 더 강화한 버전으로 아래와 같은 규칙을 만족해야 한다.

3

1. 제3 정규형을 만족해야 한다.

2. 모든 결정자가
후보키 집합에 속해야 한다.

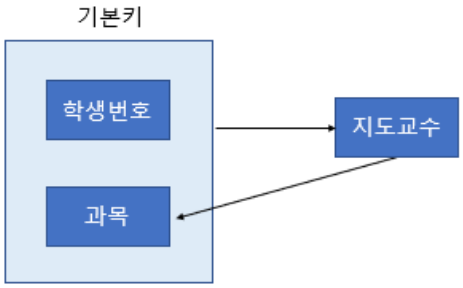
모든 결정자가 후보키 집합에 속해야 한다.

⇒ 후보키 집합에 없는 컬럼이 **결정자가 되어서는 안 된다는 뜻**이다.

후보키는 기본키가 될 수 있는 후보들 (`유일성과 최소성을 만족한 컬럼들`)

BCNF 예시

학생번호	과목	지도교수
101	자바	김자바
101	C++	박플플
102	자바	오자바
103	C#	조씨샵
104	자바	김자바



BCNF가 필요한 테이블

- `학생 번호`, `과목` 이 기본키로, 지도교수를 알 수 있다.
- 같은 과목을 다른 교수가 가르칠 수도 있어서 `과목` → `지도교수` 종속은 성립하지 않는다.
- 하지만 지도교수가 어떤 과목을 가르치는지는 알 수 있어서 `지도교수` → `과목` 종속이 성립한다.

이처럼 **후보키 집합이 아닌 칼럼**이 결정자가 되어버린 상황을 BCNF를 만족하지 않는다고 한다.

(위 테이블은 제3 정규형까지는 만족하는 테이블)

BCNF를 만족하기 위해서는 아래와 같이 분해하면 된다.

학생번호	지도교수
101	김자바
101	박플플
102	오자바
103	조씨샵
104	김자바

지도교수	과목
김자바	자바
박플플	C++
오자바	자바
조씨샵	C#

기본키

BCNF 만족 상태

☝️ BCNF 이상의 정규형(4, 5 정규형) 도 있지만, 보통 정규화는 BCNF 까지만 하는 경우가 많다고 한다.

그 이상 정규화를 하면 정규화의 단점이 나타날 수도 있기 때문이다!

출처:

https://code-lab1.tistory.com/48

https://coding-factory.tistory.com/872