정규화

데이터베이스 정규화란?

- 불필요한 데이터를 없애고 삽입/갱신/삭제 시 발생할 수 있는 각종 이상현상들을 방지하기 위한 데이터베이스 설계를 재구성하는 기술

장/단점

- 장점
  - 데이터 베이스 변경 시 이상 현상을 제거할 수 있다.
  - 정규화된 데이터베이스 구조에서는 새로운 데이터 형의 추가로 인한 확장 시, 그 구조를 변경하지 않아도 되거나 일부만 변경해도 된다.
  - 데이터베이스와 연동된 응용프로그램에 최소한의 영향만을 미치게 되어 응용 프로그램의 생명을 연장시킨다.
- 단점
  - 릴레이션의 분해로 인해 릴레이션 간의 join 연산이 많아진다.
  - 질의에 대한 응답시간이 느려질 수도 있다.
  - 만약 조인이 많이 발생하여 성능저하가 나타나면 반정규화를 적용할 수도 있다.

정규화를 하지 않았을때의 문제점 (이상현상)

S_id	S_Name	S_address	Subject
201	강동옥	California	Math
202	김태현	Nevada	English
203	박준영	New Jersey	Physics
204	성아영	Utah	Physics
205	양지훈	Oregon	Math
206	임경훈	Nevada	Physics
207	정인수	Texas	English
208	조성현	Ohio	Math
202	김태현	Nevada	Math

1. **Update** : 김태현의 address가 변경되었을 때, 여러줄의 데이터 갱신. 이로인해 데이터의 불일치가 발생
2. **Insert** : 강동옥이 아무 수업을 듣지 않을때, Subject컬럼에 Null 값이 들어감
3. **Delete** : 박준영이 과목 수강을 취소한다면 박준영의 레코드는 아예 테이블에서 삭제

→ 위와 같이 정규화가 제대로 되어있지 않으면 Update / Insert / Delete시 다양한 문제점이 발생. 이를 테이블의 구성을 논리적으로 변경하여 해결하고자 하는것이 정규화

정규화의 법칙

- 단계별로 1차 정규화, 2차 정규화, 3차 정규화, BCNF, 4차 정규화, 5차 정규화가 존재

1차 정규화

- 각 칼럼이 하나의 값만 가져야한다.
- 하나의 칼럼은 같은 종류나 타입을 가져야한다.
- 각 칼럼이 유일한 이름을 가져야한다.
- 칼럼의 순서가 상관없어야한다.
  - 아래의 표의 경우 원자성을 만족하지 않는다.

S_id	S_Name	S_address	Subject
201	강동옥	California	Math
202	김태현	Nevada	English,Math
203	박준영	New Jersey	Physics
204	성아영	Utah	Physics
205	양지훈	Oregon	Math
206	임경훈	Nevada	Physics
207	정인수	Texas	English
208	조성현	Ohio	Math

  - 1차 정규화를 위해 한개의 row를 증가. 1차 정규화 만족

S_id	S_Name	S_address	Subject
201	강동옥	California	Math
202	김태현	Nevada	English
203	박준영	New Jersey	Physics
204	성아영	Utah	Physics
205	양지훈	Oregon	Math
206	임경훈	Nevada	Physics
207	정인수	Texas	English
208	조성현	Ohio	Math
202	김태현	Nevada	Math

2차 정규화

- 1차 정규화 + 테이블의 모든 칼럼이 완전 함수적 종속을 만족

▼ *완전 함수적 종속 만족?*

기본 키 중 특정 칼럼에만 종속된 칼럼(부분적 종속)이 없어야 함.

S_id	Subject	S_Name	S_Score
201	Math	강동옥	3.5
202	English	김태현	4.5
203	Physics	박준영	2.5
204	Physics	성아영	3.0
205	Math	양지훈	4.0
206	Physics	임경훈	4.5
207	English	정인수	2.0
208	Math	조성현	3.0
202	Math	김태현	3.5

이 테이블에서 기본키 → (S_id, Subject)

이 기본키는 성적을 결정하고 있다. 그러나 여기서 S_Name 컬럼은 기본키의 부분집한인 S_id에 의해 결정될 수 있다. 즉, 기본키의 부분키인 S_id가 결정자이기 때문에 위의 테이블의 경우 다음과 같이 분해하여 별도의 테이블로 관리하여 제 2 정규형을 만족 시킨다.

학생

S_id	S_Name
201	강동옥
202	김태현
203	박준영
204	성아영
205	양지훈
206	임경훈
207	정인수
208	조성현

수강

S_id	Subject	S_Score
201	Math	3.5
202	English	4.5
203	Physics	2.5
204	Physics	3.0
205	Math	4.0
206	Physics	4.5
207	English	2.0
208	Math	3.0
202	Math	3.5

3차 정규화

- 제 2 정규화 + 이행적 종속을 없애도록 테이블을 분해하는 것. 기본키를 제외한 속성들 간의 이행적 함수 종속이 없는 것. 즉, 기본키 이외의 다른 칼럼이 그 외 다른 칼럼을 결정할 수 없는 것

▼ *이행적 종속?*

A → B, B→C가 성립할 때 A→C 가 성립되는 것을 의미

계절학기

S_id	Subject	Cost
201	Math	30000
202	English	40000
206	Physics	50000
207	English	40000
208	Math	30000

S_id → Subject 결정, Subject → Cost 결정

따라서 이 테이블은 (S_id, Subject) 와 (Subject, Cost) 테이블로 분해해야한다.

이행적 종속을 제거하는 이유?

예를들어, S_id 208 학생이 수강을 바꾼다고 하자. 그렇다면 그 강좌에 맞게 cost를 변경해주어야한다. 이 번거로움을 해결하기 위해 제 3 정규화를 하는 것이다.

계절 수강

S_id	Subject
201	Math
202	English
206	Physics
207	English
208	Math

수강료

Subject	Cost
Math	30000
English	40000
Physics	50000
English	40000
Math	30000

BCNF 정규화

제 3 정규화를 진행한 테이블에 대해 모든 결정자가 후보키가 되도록 테이블을 분해.

모든 결정자가 후보키 집합에 속해야한다. 즉, 후보키 집합에 없는 칼럼이 결정자가 되어서는 안된다는 뜻.

▼ *후보키?*

릴레이션을 구성하는 속성들 중 튜플을 유일하게 식별 할 수 있는 속성들의 부분집합

모든 릴레이션은 반드시 하나 이상의 후보키를 가져야한다.

ex) 만약 학생 릴레이션에 학번, 주민등록번호, 이름, 주소가 있다면 후보키는 학번과 주민등록번호가 된다. 즉 기본키가 될 수 있는 키들을 후보키라고 한다.

S_id	Subject	professor
201	Math	Bob

S_id	Subject	professor
202	English	Emily
206	Physics	Ryan
207	English	Emily
208	Math	Brian

(S_id, Subject)가 기본키로 Professor를 알 수 있다. 하지만 같은 과목을 다른 교수가 가르칠수도 있기에, Subject → Professor 종속 성립 X(3차 정규형 만족). 하지만 Professor가 어떤 과목을 가르치는지 알 수 있으므로 Professor → Subject 가 성립. 이처럼 후보키 집합이 아닌 칼럼이 결정자가 되어버린 상황을 BCNF를 만족 하지 않는다고 한다.

BCNF를 만족하기 위해선 아래와 같이 분해하면 된다.

S_id	Subject
201	Math
202	English
206	Physics
207	English
208	Math

Subject	professor
Math	Bob
Math	Brian
English	Emily
Physics	Ryan

reference)

- https://code-lab1.tistory.com/48
- https://3months.tistory.com/193