함수적 종속과 정규화

함수적 종속과 정규화

함수적 종속

- 논리적 설계 단계에서 데이터 중복 문제를 해결하기 위한 수당

  → 테이블을 분해함으로써 달성 가능

- = 무결성 제약의 한 종류
- 테이블 내 필드 간의 관계성을 표현

  → 데이터 중복의 발생 여부를 파악하는데 사용

  - 필드 X의 값이 동일한 임의의 레코드에 대해 필드 Y의 값도 동일하다면,
  - X→Y : Y는 X에 함수적 종속된다 (X는 결정자, Y는 종속자)

    = X는 Y를 함수적으로 결정한다.

- 함수 종속의 관계
  - 다대일 : stu_id → dept_name
  - 일대일 : stu_id → resident_id
    resident_id → stu_id
- 필드 집합 간의 함수 종속
  - 결정자와 종속자가 두 개 이상의 필드로 구성
  - (stu_id, dept_name) → office
  - stu_id → (name, dept_name)

    : 학번이 같은 두 개의 동일한 학생이 있다면, 그 둘은 이름도 같고 학과 이름도 같다~

함수적 종속의 특징

- 포함 규칙
  - X ⊃ Y이면 X → Y가 항상 성립 : X → X / (X, Y) → X
- 분해 규칙
  - X → (Y, Z)이면 X → Y, X → Z가 각각 항상 성립한다.

주의) 반대는 성립하지 않음

  - (A, B) → C가 성립한다고 A → B, B → C가 만족되지 않음

- 합성 규칙
  - X → Y, X → Z가 성립하면 X → (Y, Z) 도 성립한다.
- 이행 규칙
  - X → Y, Y → Z가 성립하면 X → Z도 성립한다.
- 키와 함수적 종속의 관계
  - 테이블의 모든 필드는 키에 함수적으로 종속된다
  - 테이블의 필드 X가 나머지 필드들을 함수적으로 결정하면 필드 X는 수퍼키이다.
  - 테이블 R에서 필드 K가 수퍼키이면, K→R이 성립함

함수적 종속의 유지 방법

- 테이블에 데이터를 삽입할 때 함수적 종속이 계속 유지되어야 함
- RDBMS에서의 함수적 종속 유지 방법 (관계형 데이터베이스 관리 시스템(RDBMS)은 **관계형 데이터베이스를 만들고 업데이트하고 관리하는 데 사용하는 프로그램**입니다.)
  - 기본키의 경우 함수적 종속이 계속 유지될 수 있음

    → 결정자가 기본키인 함수적 종속들은 자동적으로 만족됨
  - 기본키가 아닌 필드에 대해서는 함수적 종속 유지가 어려움

    → 예) dept_name → office를 항상 만족하기 위한 적합한 방법은 없음
- 해결 방법
  - 데이터베이스를 설계할 때 이러한 문제가 발생하지 않도록 테이블들을 정의 →정규화(normalization)

정규화

불필요한 데이터의 중복을 피하기 위해 스키마를 분해하는 과정 → 테이블 분해

→ 함수적 종속이 중요한 역할을 함

- 데이터 중복의 문제점

  : 데이터가 중복 저장하는 경우에는 데이터 변경에 의해 원하지 않던 결과가 발생할 수 있음

  → 이상 현상

        - 삽입 이상
            - 데이터를 삽입할 수 없거나 원치 않는 데이터를 삽입
        - 삭제 이상
            - 삭제되지 말아야 할 정보까지 함께 삭제되는 현상
        - 수정 이상
            - 중복된 정보의 일부만 수정하여 정보의 불일치가 발생하는 현상
- 해결 방안
    - 테이블을 분해한다 → 이상 현상이 더 이상 발생하지 않음
- 정규화의 효과
    - 테이블에 대한 삽입, 삭제, 수정 등의 연산으로 인해 발생할 수 있는 이상 현상을 방지하는 수단을 제공
    - 데이터 중복으로 발생하는 문제 해결

      → 단, 함수적 종속이 유지 될 수 있도록 데이터베이스 설계해야 함

- 정규형이란? 각 단계별 정규화 과정을 통해 분해된 테이블들
- 정규형의 종류
    - 1차~ 5차 정규형까지 있다~ + 특수인 보이스-코드 정규형도 있다~
    - + 4,5차 정규형은 고려하지 않음~

1차 정규형

테이블 R에 속한 모든 도메인이 원자값(atomic value)만으로 구성되어 있다면 R은 1차 정규형임

- 문제점 → 부분 종속 : 키가 아닌 필드 A가 키의 일부인 필드 X에 종속되는 경우 →중복

2차 정규형

- 부분 종속의 제거 → 부분 종속에 해당하는 결정자와 종속자를 별도의 테이블로 분리

테이블 R에서 키가 아닌 모든 필드가 키에 함수적으로 종속되며, 키의 부분집합이 결정자가 되는 부분 종속이 존재하지 않으면 R은 2차 정규형이다

- **테이블 분해 조건**
    - **종속성 보존 분해** : 분해되기 전의 함수적 종속들이 분해 후에도 유지되어야 한다.

+) 데이터의 종속성은 프로그램의 구조가 데이터의 구조에 영향을 받는 것을 의미

즉, 데이터의 구조가 프로그램의 데이터 저장 방식을 결정하고 반대로 프로그램의 데이터 저장 방식에 따라 데이터의 저장 방식이 바뀌는 것을 말합니다. 데이터의 종속성 때문에 데이터의 구조가 변경되면 프로그램까지 같이 바뀌는 비용이 들기 때문에 프로그램 개발과 유지 보수가 어려워집니다.

- **무손실 조인 분해** : 분해를 하는데 분해를 할 때 손실이 없어야 한다.

  +규칙

- 문제점 → 기본키의 이행 종속
- 해결 방안 → 이행 종속에 참여한 필드들을 다른 테이블로 분해
  - 종속성 보존과 무손실 분해가 모두 만족됨

3차 정규형

테이블 R이 2차 정규형이면서 키에 속하지 않은 모든 필드가 기본키에 이행 종속되지 않는다면 R은 3차 정규형이다 ⇒ 부분종속도 없고, 이행종속도 없는 경우

- 문제점 → 키에 포함되는 필드 집합 A와 키에 포함되지 않은 필드 집합 X에 대하여
  - X → A라는 함수적 종속이 존재할 경우 데이터 중복 발생

보이스-코드 정규형

테이블 R에 존재하는 모든 함수적 종속에서 결정자가 후보키이면 R은 보이스-코드 정규형이다.

- 문제점 → 함수적 종속이 보존 X

관계형 데이터베이스 설계의 원칙

3차 정규형

- 장점: 무손실 조인 분해, 종속성 보존 / 단점: 데이터 중복으로 인한 이상 현상

보이스-코드 정규형

- 장점: 이상 현상 방지 / 단점: 종속성 보존이 안됨

무엇을 선택할 것인가?
- 이상 현상 방지보다 종속성 보존이 더 중요함
- 함수적 종속성 보존이 가능하다면 보이스-코드 정규형 선택
- 함수적 종속성이 보존되지 않는다면 3차 정규형 선택

정규형을 감안한 논리적 설계 과정

1. ER스키마를 테이블 스키마로 변환
2. 변환된 각 테이블에서 함수적 종속을 정의
3. 각 테이블이 보이스-코드 정규형인지를 검증하고, 아니라면 보이스-코드 정규형으로 분해
4. 분해 결과 함수적 종속성이 보존되지 않는다면, 제 3정규형으로만 분해

역정규화

정규화된 데이터베이스에서 **성능을 개선**하기 위해 사용되는 전략

- 정규화의 단점으로 언급되었던 것이 '비용(Cost)'이다.

역정규화는 데이터베이스의 비용을 최소화하기 위해 중복을 허용하며 Entity를 다시 통합하거나 분할하여 정규화 과정을 통해 도출된 DB 구조를 재조정하는 과정임