RDBMS 설계방법

Date @July 15, 2021

DB 설계 목적

- 프로젝트, 명세서 등의 정보 요구사항에 대한 정확한 이해
- 분석자, 개발자, 사용자 간의 원활한 의사소통 수단
- 데이터 중심의 분석 방법
- 현행 시스템만이 아닌 신규 시스템 개발의 기초 제공

- 설계를 대충하면 기능 한 개 추가 될 때마다 DB와 관련된 이미 개발된 프로그램도 함께 뜯어고쳐야 하는 경우가 발생한다.

개념적 설계

- 데이터 베이스에 대한 사용자의 요구사항을 수집하고 분석해서 아래와 같이 요구사항(기능) 명세서를 작성

> 🚀 **요구사항 명세 샘플 - 항공상 DB**
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야한다.
> 회원의 신용카드 정보는 여러개를 저장할 수 있다. 신용카드 번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한명은 좌석을 하나만 예약할 수 있고, 한 좌석은 회원 한명만 예약할 수 있다.

- 작성한 요구사항 명세서에서 데이터베이스를 구성하는 필요한 개체, 속성, 개체 간의 관계를 추출하여 ERD를 생성
  1. 개체(Entity)와 속성(Attribute)을 추출 : 대부분 명사
  2. 개체 간의 관계(Relationship)을 추출 : 대부분 동사

1. 관계에 속한 속성이 있을 수 있다. 개체간의 관계는 무조건 동사
2. 1:1, 1:N, N:1
3. 필수적인 참여, 선택적인 참여

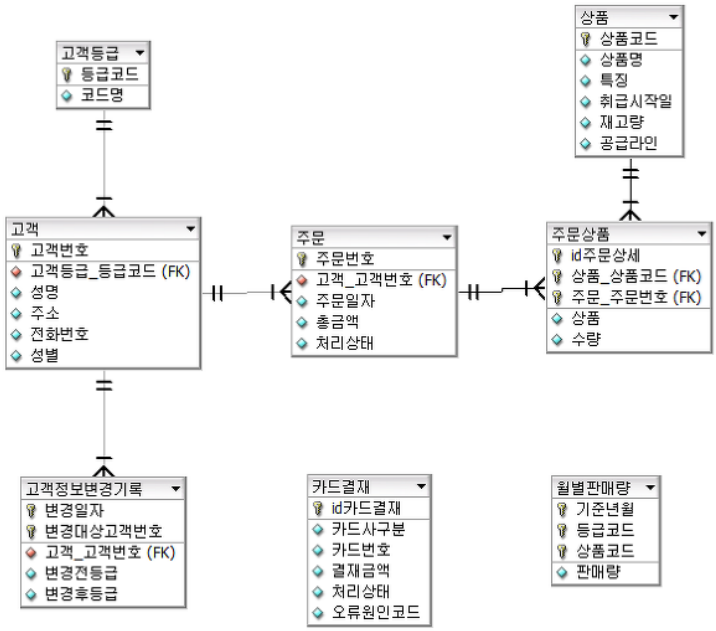
- 요구사항에서 개체(Entity)는 대부분의 명사, 속성(Attribute)과 구별하여 추출한다.

> 🚀 **요구사항 명세 샘플 - 항공상 DB**
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야한다.
> 회원의 신용카드 정보는 여러개를 저장할 수 있다. 신용카드 번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한명은 좌석을 하나만 예약할 수 있고, 한 좌석은 회원 한명만 예약할 수 있다.

- 개체 간의 관계(Relationship)는 여러가지로 분류해서 정의된다.

> 🚀 **요구사항 명세 샘플 - 항공상 DB**
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야한다.
> 회원의 신용카드 정보는 여러개를 저장할 수 있다. 신용카드 번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한명은 좌석을 하나만 예약할 수 있고, 한 좌석은 회원 한명만 예약할 수 있다.

- ERD 생성



논리적 설계

- 모든 개체를 릴레이션(Table)로 변환
- N:M 관계는 릴레이션(Table)으로 변환
- 1:N 관계는 외래키(FK)로 변환
- 1:1 관계는 외래키(FK)로 변환
- 다중 값 속성은 독립 릴레이션(Table)로 변환

물리적 스키마 및 구현

- ERD를 실제 테이블로 생성한다 (Workbench 같은 DB Tool이나 SQL 스크립트 사용으로도 가능해야 함)

정규화 vs 반정규화

: 정규화된 엔티티 타입, 속성, 관계를 시스템의 성능 향상, 개발과 운영의 단순화를 위해 모델을 통합하는 프로세스

정규화 모델
이상적인 논리모델은 모든 엔티

반정규화 모델

티타입, 속성, 관계가 반드시 한 개만 존재하며 따라서 입력, 수정, 삭제도 한군데에서만 발생하므로 데이터 값이 변질되거나 이질화 될 가능성이 없다. 반면 여러 테이블을 생성되어야 하므로 SQL 작성이 용이하지 않고 과다한 테이블 조인이 발생하여 성능이 저하될 가능성이 높다.

반대로 반정규화를 하면 여러 개의 테이블이 단순해지므로 SQL 작성이 용이하고 성능이 향상될 가능성이 많다. 그러나 같은 데이터가 여러 테이블에 걸쳐 존재하므로 무결성이 깨질 우려가 있다.

⇒ 둘 중 무조건 하나가 좋은 것은 아니다! 적절하게 잘 조합하는게 최선! 개념 잘 잡기!

테이블 반정규화 방법

- 1:1 관계의 테이블 병합
- 1:N 관계의 테이블 병합
- 수퍼/서브 타입 테이블 병합
- 수직 분할(집중화된 일부 컬럼을 분리)
- 수평 분할(행으로 구분하여 구간별 분리)
- 테이블 추가(중복테이블, 통계 테이블, 이력테이블, 부분 테이블)

컬럼 반정규화

- 중복 컬럼 추가(자주 조회하는 컬림이 있는 경우)
- 파생 컬럼 추가(미리 계산한 값)
- PK에 의한 컬럼 추가
- 응용시스템 오작동을 위한 컬럼 추가(이전데이터 임시보관_

관계 반정규화

- 중복 관계 추가(이미 A테이블에서 C테이블의 정보를 읽을 수 있는 관계가 있음에도 관계를 중복하여 조회(Read) 경로를 단축

Quiz

- DB 설계의 목적이 아닌것은? (4)
  1. 요구사항에 대한 정확한 이해
  2. 시스템 개발의 기초 제공
  3. 데이터 중심의 분석방법
  4. 개발시간의 단축
- 정규화된 엔티티타입, 속성, 관계를 시스템의 성능향상, 개발과 운영의 단순화를 위해 모델을 통합하는 프로세스를 무엇이라고 하나? (3)
  1. 정규화
  2. 최적화
  3. 반정규화
  4. 단순화
- DB 설계를 **대충**하면 기능 한개 추가될 때마다 DB 관련된 이미 개발된 프로그램도 뜯어 고쳐야하는 경우가 생긴다.
- DB 구성하는 개체(Entity)와 속성(Attribute)을 추출할 때 대부분 **명사**로 선별한다.
- DB 설계할 때 반정규화 방법이 가장 좋다 **X**

  ⇒ 무조건이라는 건 없다!