

DB설계

DB설계의 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 위한 요구사항 분석

- 데이터베이스에 대한 사용자의 요구사항을 수집하고 분석해서 요구사항(기능) 명세서를 작성

개념적 설계

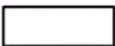


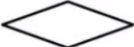


- 작성한 요구사항 명세서에서 데이터 베이스를 구성하는데 필요한 개체, 속성, 개체간의 관계를 추출하여 ERD를 생성
 - 개체(Entity)와 속성(Attribute)을 추출한다.
 - 대부분 명사로 선별한다
 - 개체 간의 관계(Relationship)를 추출한다.
 - 대부분 동사로 선별한다
 - 관계에 속한 속성도 있을 수 있다.(1:1, 1:N, N:M)

1. 개체(Entity)와 속성(Attribute)을 추출

- 요구사항에서 개체는 대부분 명사로 이루어져 있지만, 속성과 구별하여 추출한다.

2. 개념 설계 기반으로 ERD 생성

- 개체 간의 관계는 여러가지로 분류해서 정의 된다.
 - ERD 표준 기준

기 호	의 미
	개체
	속성
	기본키
	관계
	개체 타입과 속성을 연결
	개체간의 관계 타입

논리적 설계

- 모든 개체는 릴레이션(Table)으로 변환
 - ER다이어그램의 각 개체를 릴레이션으로 변환
 - 개체 → 테이블
 - 속성 → 테이블의 속성
- N:M관계는 릴레이션(Table)으로 변환
 - 관계 → 릴레이션 이름
 - 관계 속성 → 릴레이션 속성
- 1:N 관계는 외래키로 표현
 - 일반적으로 1:N관계에서 1측 개체의 기본키를 N측 릴레이션에 포함시키고 외래키(FK)로 지정
- 1:1관계는 외래키로 표현
 - 일반적 1:1관계는 외래키(FK)를 서로 주고 받는다
- 다중 값 속성은 독립 릴레이션(Table)으로 변환
 - 릴레이션에서는 다중 값 속성을 가질 수 없으므로 다중 값 속성은 별도의 릴레이션으로 생성해야 함

물리적 스키마 및 구현

- ERD를 실제 테이블로 생성한다
- 현업에서는 SQL 스크립트를 주로 사용함

반정규화

반정규화란?

- 정규화 권 엔티티 타입, 속성, 관계를 시스템의 성능향상, 개발과 운영의 단순화를 위해 모델을 통합하는 프로세스
 - 〈정규화 모델〉

이상적인 논리모델은 모든 엔티티타입, 속성, 관계가 반드시 한 개만 존재하며 따라서 입력, 수정, 삭제도 한군데에서만 발생하므로 데이터 값이 변질되거나 이질화 될 가능성이 없다. 반면 여러 테이블이 생성되어야 하므로 **SQL작성이 용이하지 않고 과다한 테이블 조인이 발생하여 성능이 저하될 가능성이 높다.**
 - 〈반정규화 모델〉
 - 반대로 반정규화를 하면 여러 개의 테이블이 단순해지므로 SQL작성이 용이하고 성능이 향상될 가능성이 많다. **그러나 같은 데이터가 여러 테이블에 걸쳐 존재하므로 무결성이 깨질 우려가 있다.**

테이블 반정규화 방법

- 1:1관계의 테이블 병합
- 1:N관계의 테이블 병합
- 슈퍼/서브 타입 테이블 병합
- 수직분할(집중화된 일부 컬럼을 분리)
- 수평분할(행으로 구분하여 구간별 분리)
- 테이블 추가(중복테이블, 통계 테이블, 이력테이블, 부분테이블)

대표적 반정규화 - 컬럼 반정규화

- 중복 컬럼 추가(자주 조회하는 컬럼이 있는 경우)
- 파생 컬럼 추가(미리 계산한 값)
- PK에 의한 컬럼 추가
- 응용 시스템 오작동을 위한 컬럼 추가(이전 데이터 임시 보관)

대표적 반정규화 - 관계 반정규화

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

그렇다면, 정규화가 좋은가 반정규화가 좋은가?

→ 정답은 정해져 있지 않고 그때 그때 다르다. 개념을 잘 알고 적절하게 사용해야한다

면접

Q. 반정규화에 대해 간단히 설명해 보세요

A : 데이터 베이스의 **성능 향상**을 위하여, 데이터 중복을 허용하고 조인을 줄이는 데이터 베이스 성능 향상 방법입니다.