영어권 학술장 한국학 연구성과의 디지털 구현: 온톨로지 및 데이터베이스 설계를 중심으로

Presenter : Jae-hong, Joo
2025. 7. 2

목차

- 데이터베이스를 설계하는 6가지 과정
  - 1단계: 요구사항 분석 (Requirements Analysis)
  - 2단계: 개념적 설계 (Conceptual Design)
  - 3단계: 논리적 설계 (Logical Design)
  - 4단계: 물리적 설계 (Physical Design)
  - 5단계: 구현 (Implementation)
  - 6단계: 테스트 및 유지보수 (Testing & Maintenance)

1단계: 요구사항 분석 (Requirements Analysis)

- 데이터베이스가 왜 필요한지, 무엇을 위해 사용할 것인지를 제시하는 단계
- 데이터베이스의 **목적**과 **범위**를 명확하게 정의해야함

- 어떤 종류의 데이터를 다뤄야하는지?
- 데이터베이스로 어떤 기능들이 구현되어야 하는지?
- 데이터베이스를 누가 사용할 것인지?
- 어떤 정보를 저장하고 조회, 수정, 삭제하는지?

- 데이터의 성격에 따라, 목적에 따라 필요한 작업들이 달라짐
  (데이터 수집, 데이터 전처리, 툴 사용, 프레임워크 설계 등)

1단계: 요구사항 분석 (Requirements Analysis)

- 1단계에서 가장 중요한 것은 **'모든 참여자가 동의하는 명확한 정의'**
- 요구사항, 상호작용, 데이터 정보 등 모든 과정에서 모호함이 존재하면 안 됨

- 요구사항 정의서 (Requirements Definition Document, RDD)
  - DB를 설계함으로써 달성하고자 하는 프로젝트의 최종 목표 + 기대 효과
  - 시스템이 반드시 수행해야 하는 기능적 요구사항 (예: 사용자는 이메일과 비밀번호로 로그인)
    Usecase Diagram으로 사용자와 시스템 간 상호작용을 시각화하여 보완 가능
  - 성능, 보안, 가용성 등 품질과 관련된 비기능적 요구사항 (예: 검색 결과는 1초 내로 응답해야함)
  - 데이터의 종류, 출처, 보존 기간, 접근 권한 등의 데이터 요구사항

- 데이터 사전 (Data Dictionary) / 용어 정의서 (Glossary)
  - 프로젝트에서 사용되는 모든 데이터 용어를 명확하게 정의해야함
    (예: '연구주제'는 '해당 연구에서 명시한 keywords 목록'을 의미함)

1단계: 요구사항 분석 (Requirements Analysis)

- ‘검색이 빨랐으면 좋겠다’ 와 같은 모호하고 추상적인 요구사항
  - 빠르다의 기준은? 데이터의 양에 따라 달라지는 속도는?
  - ‘사용자가 검색을 했을 때 모든 결과는 2초 내로 조회되어야 함’ → 명확함

- ‘이런 것도 조회가 되면 좋지 않을까?’ ‘이런 기능도 넣어볼까요?’
  - DB에 요구하는 기능은 명확한 우선순위를 가져야 함
    (반드시 있어야 하는 기능 vs 있으면 좋은 기능)
  - 프로젝트가 중요하게 생각하는 기능을 우선적으로 설계 (속도? 안정성? 보안?)
  - 체계적인 절차를 거치지 않고 변동사항이 생기면 적용이 어려움
  - 개발자가 생각하는 요구사항이 기획자나 사용자와 다를 수 있으니 주의

- ‘이게 아닌데요?’
  - 서로가 생각하는 결과가 다르다면, 많은 시간과 비용이 낭비되기 쉬움
  - 모든 결정사항이 문서화 + 해당 내용은 모두가 동의하는 결과물이어야 함

1단계: 요구사항 분석 (Requirements Analysis)

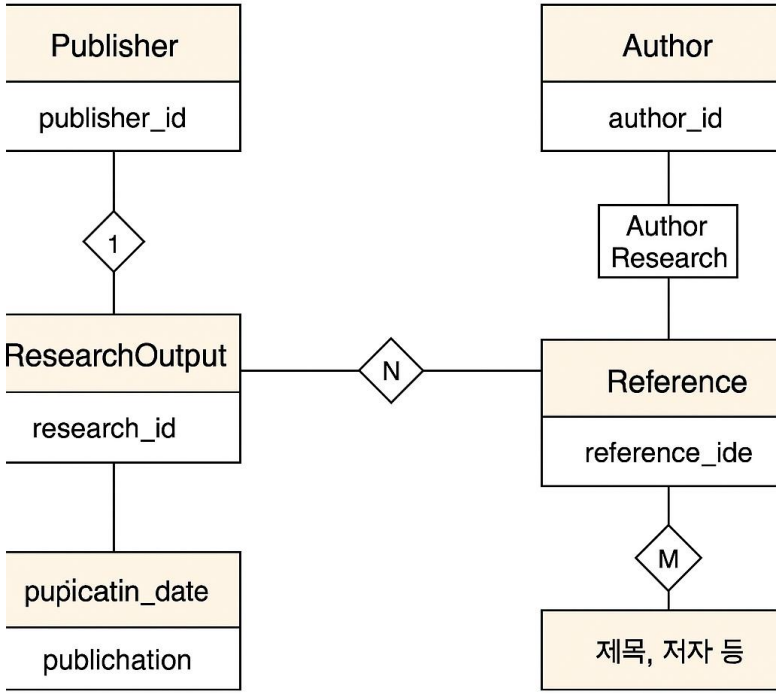
- 영어권 학술장 한국학 연구성과의 디지털 구현
  - 영어권 학술장을 어떻게 정의할 것인가? → 예: 특정 학술지들을 선정
  - 한국학 연구성과를 어떻게 정의할 것인가? → 예: 특정 키워드가 포함되는지를 확인
  - 한국학의 종류는? → 예: 문학, 사학, 철학, 예술, 과학, 교육
  - 어떤 데이터를 저장할 것인가? → 예: 논문명, 저자, 시기, 출판사명, 참고문헌 등
  - DB를 어떤 목적으로 사용할 것인가? → 예: 웹페이지에 연동하여 열람이 가능한 서비스 제공
  - 제공되어야 하는 기능은? → 예: 조회, 추가, 삭제 등

→ 명확한 기준을 기반으로 진행해야하며, 초기 과정이 중요한만큼 시간이 오래 걸림
시간적 분석, 주제별 분석, 연구자 및 기관 분석, 관계 및 영향력 분석 등 목적 제시 필요

2단계: 개념적 설계 (Conceptual Design)

- 요구사항 분석을 바탕으로 데이터베이스의 뼈대를 그리는 단계
- 핵심 개체(Entity)와 그들 간의 관계(Relationship)를 표시한 다이어그램

- 핵심 개체 식별
  - 연구 성과
  - 연구자
  - 소속기관
  - 연구분야
  - 출판시기



- 관계 정의
  - 한 명의 연구자 → 여러 성과 (1:N 관계)
  - 저자 → 연구 성과 (N:M 관계)

  → 관계를 잘못 설계한 경우 고치는 과정에서 DB 구조 전체를 뒤엎는 경우도 발생

3단계: 논리적 설계 (Logical Design)

- 개념 모델을 관계형 데이터베이스 이론에 맞춰서 구체화하는 단계
- 테이블 구조와 같은 논리적인 구조를 생성 (3단계의 핵심은 **정규화**)
- 기본 키(PK)
  - 고유하게 식별할 수 없는 데이터는 사용할 수가 없음
  - PK는 절대 변하지 않는 값을 사용하는 것이 원칙
    (예: 이메일로 PK를 설정하는 경우, 이메일 변경 시 이를 참조하던 모든 FK를 변경해야함)

- 외래 키(FK)
  - 다른 테이블의 기본 키를 참조하여 테이블 간의 관계를 연결하는데 사용
  - 데이터의 중복 최소화 및 무결성 보장을 위한 정규화를 위해 확인 필요

- 일관된 명명 규칙 필요
  - 저자 → Author, Researcher 등 서로 다른 이름으로 정의되면 안 됨
  - 25년 7월 2일 → 20250702, 250702 등 하나의 규칙으로 통일할 필요 있음

3단계: 논리적 설계 (Logical Design)

- 연구성과
  - 고유 ID(PK), 제목, 유형(논문/단행본), 출판시기, DOI, 저자(FK), 학술지(FK) 등
- 연구자
  - 고유 ID(PK), 이름, 직위, 소속기관 등
- 학술지
  - 고유 ID(PK), 저널명, 출판기관 등

Institutions			
bigint	institution_id	PK	소속 기관 고유 ID
varchar(255)	name		기관명
varchar(100)	country		국가

Publications			
bigint	publication_id	PK	연구 성과 고유 ID
varchar(500)	title		제목
enum	pub_type		성과 유형
date	publication_date		발행일
text	abstract		초록
varchar(100)	doi		DOI
bigint	journal_id	FK	학술지 ID

개체나 속성 등의 불필요 또는 중복 내용 최종 확인 (초기 틀을 완성한 이후 데이터 확장 → 샘플링 활용)
(예: 1차년도 DB 설계 후 데이터 입력 → 2차년도 DB 추가 변경시 1차년도의 데이터 추가 작업 필요)

4단계: 물리적 설계 (Physical Design)

- 논리적 설계를 바탕으로 실제 DB 관리 시스템에 맞게 명세서를 작성하는 단계
- DB 성능, 저장 공간, 보안 등을 고려하여 **특정 DBMS에 최적화된 형태로 변환**

- 데이터 성격에 맞는 타입, 인덱스, 제약 조건 설정
  (값이 비어있는지의 여부, 값이 고유한지의 여부 등)
  - NOT NULL : 해당 컬럼에 반드시 값이 존재
    (연구성과의 저자는 반드시 값이 존재해야함 → NOT NULL의 제약으로 무결성 보장)
  - UNIQUE : 해당 컬럼의 모든 값이 서로 다른 고유값
    예: 연구성과의 DOI의 경우 고유한 값이여야 함 → 서로 다른 논문이 같은 DOI라면?
    연구성과의 Keywords도 고유한 값이여야 함 → A 논문의 korea와 B 논문의 korea가 다르다면?

- 데이터에 적합한 타입 설정 → 효율적인 DB의 설계 (메모리, 디스크 공간 낭비)
- 인덱스의 전략적 사용 → 효율적인 DB의 사용 (읽기/쓰기 성능 감소)
- 적절한 제약 조건 설정 → 데이터 중복 방지 및 신뢰도 보장

5단계: 구현 (Implementation)

- 물리적 설계 명세서를 기반으로 SQL을 이용하여 DB 및 테이블을 만드는 단계
- DDL(Data Definition Language)를 이용하여 DB 관리
- 스크립트로 관리로 로그 기록 가능 (Human Error를 막을 수 있어 안정성 확보)

- 목적에 따라 적합한 툴 적용 가능
  (그래프 DB인 Neo4j에 적용)
  - 노드와 관계로 구분
  - 각 노드의 고유 ID 관리
  - 명확한 관계의 종류와 속성
  - 일관된 데이터 유지

→ 온톨로지 설계 내용이 적용되었는지 확인

```sql
CREATE TABLE Researchers (
  researcher_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  birth_year INT,
  affiliation_id INT,
  FOREIGN KEY (affilication_id) REFERENCES Affiliation(affiliation_id)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

CREATE TABLE Affiliation (
  affiliation_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL
);
```

6단계: 테스트 및 유지보수 (Testing & Maintenance)

- 생성한 데이터베이스가 잘 동작하는지 테스트하고 지속적으로 관리하는 단계
- 설계한 DB에 테스트 데이터를 입력한 후 여러 기능에 오류가 없는지 확인

- 성능 모니터링 및 튜닝
  - 쿼리 조회 결과의 정확도와 속도, 분석 결과에 문제가 없는지를 확인
  - 인덱스의 활용으로 성능을 개선하고, 데이터의 품질을 관리하여 데이터를 꾸준히 표준화
- 데이터 백업 및 복구
  - 주기적인 데이터의 백업 진행 (해당 데이터로 복구까지 가능한지를 테스트)
  - 새로운 연구 성과의 추가가 가능할 수 있도록 업데이트 파이프라인 구축
- 요구사항 변경에 따른 스키마 수정
  - 참고문헌이 한국학 연구성과 DB에 포함되는 경우와 아닌 경우 어떻게 처리할지?
  - 요구사항이 추후에 변경되기 쉬운 데이터라면 이를 고려하여 유연한 스키마를 설계
  - 데이터의 성격과 목적에 따라 ERD가 달라져 DB 작업이 달라지므로 신중히 설계할 것

Thank You
Q & A