CP2. 데이터 모델과 질의 언어

Part 1. 데이터 시스템의 기초
2. 데이터 모델과 질의 언어
개요
관계형 모델과 문서 모델
데이터를 위한 질의 언어
그래프형 데이터 모델

Part 1. 데이터 시스템의 기초

2. 데이터 모델과 질의 언어

▼ 개요

- 데이터 모델은 아마도 소프트웨어 개발에서 제일 중요한 부분일 것이다.
  왜냐면 데이터 모델은 소프트웨어가 어떻게 작성됐는지 뿐만 아니라 해결하려는
  문제를 어떻게 생각해야 하는지에 대해서도 지대한 영향을 미치기 때문이다.
- 대부분의 애플리케이션은 하나의 데이터 모델을 다른 데이터 모델 위에 계층을 둬서 만든다. 각 계층의 핵심적인 문제는 다음 하위 계층 관점에서 **데이터 모델을 표현**하는 방법이다.
  - 애플리케이션 개발자는 현실(사람, 조직, 상품 등)을 보고 객체나 데이터 구조, 그리고 이러한 데이터 구조를 다른 API를 모델링한다. (주로 애플리케이션에 특화됨)
  - 데이터 구조를 저장할 때는 JSON 이나 XML 문서, 관계형 데이터베이스 테이블이나 그래프 모델 같은 범용 데이터 모델로 표현한다.
  - 데이터베이스 소프트웨어를 개발하는 엔지니어는 JSON/XML/관계형/그래프 데이터를 메모리나 디스크 또는 네트워크 상의 바이트 단위로 표현하는 방법을 결정한다. 이 표현은 다양한 방법으로 데이터를 질의, 탐색, 조작, 처리할 수 있게 한다.
  - 더 낮은 사준에서 하드웨어 엔지니어는 전류, 빛의 파동, 자기장 등의 관점에서 바이트를 표현하는 방법을 알아냈다.
- 복잡한 애플리케이션에서는 여러 API를 기반으로 만든 API처럼 중간 단계를 더 둘 수 있지만 기본 개념은 여전히 동일하다. 각 계층은 명확한 데이터 모델을 제공해 하위 계층의 복잡성을 숨긴다. 추상화는 다른 그룹의 사람들(DBA, 애플리케이션 개발자 등)이 효율적으로 함께 일 할 수 있게 한다.
- 다양한 유형의 데이터 모델이 있고 각 데이터 모델은 사용 방법에 대한 가정을 나타낸다. 어떤 종류의 사용법은 쉽고 어떤 동작은 지원하지 않는다. 어떤 연산은 빠르고, 어떤건 느리다.어떤 데이터 변환은 자연스럽지만 다른 어떤 데이터는 부자연스럽다.

- 하나의 데이터 모델만을 완전히 익히는 데도 많은 노력이 필요하다. 데이터 모델을 하나만 사용하고 내부 동작에 대한 걱정이 없을지라도 소프트웨어 작성은 그자체로 충분히 어렵다. 그러나 **데이터 모델은 그 위에서 소프트웨어가 할 수 있는 일과 할 수 없는 일에 지대한 영향을 주므로 애플리케이션에 적합한 데이터 모델을 선택하는 작업이 상당히 중요하다.**

▼ 관계형 모델과 문서 모델

- 개요 & 정의
  - 가장 잘 알려진 데이터 모델은 관계형 모델을 기반으로 한 SQL 이다.
    데이터는
    (SQL - table) 관계로 구성되고 각 관계는 순서 없는 튜플 **(SQL - row)** 모음이다.
  - 관계형 데이터베이스 관리 시스템(RDBMS)과 SQL은 정규화된 구조로 데이터를 저장하고 질의할 필요가 있는 사람들 대부분이 선택하는 도구가 되었다.
  - 관계형 데이터베이스의 근원은 **비즈니스 데이터 처리**에 있다. 이 사용 사례는 보통 **트랜잭션**과 **일괄 처리**로 오늘날의 관점에서는 일상적으로 수행되는 일이다.
  - 수년 동안 데이터 저장과 질의를 위해 많은 접근 방식이 경쟁했다. 하지만 이긴건 SQL
    (70, 80 초반 - 네트워크 모델과 계층 모델)
    (80후반, 90초반 - 객체 데이터 베이스)
    (00 초반 - XML 데이터베이스)
  - 컴퓨터가 훨씬 더 강력해지고 네트워크화됨에 따라 컴퓨터늘 점점 더 다양한 목적으로 사용하기 시작했다. (사용사례가 비즈니스 데이터 처리를 넘어 폭 넓은 다양한 사용 사례에도 보편화됨)
- NoSQL (Not Only SQL) 등장
  - NoSQL 데이터베이스를 채택하게 된 원동력
    - 대규모 데이터셋이나 매우 높은 쓰기 처리량 달성을 관계형 데이터베이스보다 쉽게 할 수있는 뛰어난 확장성 필요
    - 상용 데이터베이스 제품보다 무료 오픈소스 소프트웨어에 대한 선호도 확산
    - 관계형 모델에서 지원하지 않는 특수 질의 동작
    - 관계형 스키마의 제한에 대한 불만과 더욱 동적이고 표현력이 풍부한 데이터 모델에 대한 바람
  - 관계형 데이터베이스가 폭넓은 다양함을 가진 비관계형 데이터 스토어와 함께 사용될 것이다. 이런 개념을 **다중 저장소 지속성** 이라 한다.
- 객체 관계형 불일치
  - 대부분의 애플리케이션은 객체지향 프로그래밍 언어로 개발한다. 이는 SQL 데이터 모델을 향한 공통된 비판을 불러온다. 데이터를 관계형 테이블에 저장하려면 애플리케이션 코드와 데이터 베이스 모델 객체 사이에 거추장스러운 전환 계층이 필요하다. 이런 모델 사이의 분리를 **임피던스 불일치**라 부른다.

    - ORM (객체 관계형 매핑) 프레임워크로 양을 줄이지만, 두 모델 간의 차이를 완벽히 숨길 수 없다.
    - 1 대 N 관계를 처리하는 다양한 방법
        - SQL 방식, 테이블 여러개와 Join
        - SQL 에 구조화된 데이터 타입과 XML, JSON 지원으로 단일 로우에 다중 값 저장, 문서 내 질의와 색인 가능해짐.
        - N 이 될 수 있는 정보를 JSON, XML 문서로 부호화해 애플리케이션 단에서 해당 구조와 내용을 해석하게 하는 방식.
    - JSON 표현은 다중 테이블 스키마보다 더 나은 **지역성**을 갖는다. 관계형 테이블 이 여러개였다면 난잡한 다중 조인을 수행해야한다.
- 1 대 N과 N 대 M 관계
    - ID, Key를 사용하는 이유
        - ID 사용시 사람에게 의미 있는 정보는 한 곳에만 저장하고 그것을 참조하는 모든 것을 ID를 사용한다. (DB 내에서만 의미 있음)
        - 텍스트를 직접 저장한다면 그것을 사용하는 모든 레코드에서 사람을 의미하는 정보를 중복해서 저장하게 된다.
        - ID 자체는 아무런 의미가 없음으로 변경할 필요가 없다. 즉 식별 정보를 변경해도 ID는 동일하게 유지할 수 있다. 하지만 의미를 갖는 경우 미래에는 언젠간 변경해야 할 수도 있다. 정보가 중복되어 있으면 쓰기 오버헤드와 불일치 위험이 있다. 이런 중복을 제거하는 일이 **데이터베이스의 정규화** 핵심 개념이다. (중복된 데이터를 정규화하려면 다대일 관계가 필요)
    - 1 대 N 관계 - ex) 계정, 캐릭터
        - 계정 기준으로 여러개의 캐릭터를 둘 수 있다.
        - 캐릭터 기준으로 계정은 하나의 계정이다.
    - N 대 M 관계 - ex) 강의, 학생
        - 학생 기준으로 강의를 여러개 들을 수 있다.
        - 강의 기준으로 여러 학생을 수강 할 수 있다.

▼ 데이터를 위한 질의 언어

▼ 그래프형 데이터 모델