[데이터 모델링의 이해]

<1 장. 데이터 모델링>

1. 데이터 모델링의 이해

(1) 데이터 모델링의 중요성 및 유의점

1) 중복 : 같은 시간 같은 데이터 제공
2) 비유연성 : 데이터 모델이 수시로 변경되면 안됨
3) 비일관성 : 데이터 간 상호 연관관계에 대해 명확히 정의

(2) 데이터 모델링의 특징

1) 추상화 : 현실세계를 간략히 표현
2) 단순화 : 누구나 쉽게 이해할 수 있게 표현
3) 명확성 : 명확하게 의미가 해석되어야 함, 한 가지 의미를 가져야 함.

(3) 데이터 모델링 단계

1) 개념적 모델링
- 추상화 수준 가장 높음
- 핵심 엔터티 도출
- ERD 작성
- 전사적 관점

2) 논리적 모델링
- 특정 DB 모델에 종속됨
- 세부속성, 식별자, 관계 등을 정확히 표현
- 정규화 수행
- 재사용성 높음
- 논리 모델링의 외래키는 물리 모델에서 반드시 구현되지는 않음

3) 물리적 모델링
- 추상화 수준 가장 낮음 (가장 구체적)
- 실제 DB 구축시 참고되는 모델
- 성능, 보안, 가용성 등을 고려

(4) 데이터 모델링 관점

1) 데이터 관점
- 비지니스 프로세스에서 사용되는 데이터를 의미

2) 프로세스 관점
- 비지니스 프로세스에서 수행하는 작업을 의미

3) 데이터와 프로세스 관점
- 프로세스와 데이터 간의 관계를 의미

(5) 데이터 모델링을 위한 ERD

1) 작업순서
엔터티 그림 -> 엔터티 배치 -> 엔터티 관계 설정 ->
관계명 서술 -> 관계 참여도 표현 -> 관계 필수여부 표현

(6) 좋은 데이터 모델의 요소
1) 완전성
2) 중복배제
3) 업무규칙
4) 데이터 재사용
5) 의사소통
6) 통합성

(7) 데이터 모델링의 3 요소
1) 어떤 것 (Things)
2) 성격 (Attributes)
3) 관계 (Relationships)

2. 3 층 스키마

(1) 3 층 스키마
- 사용자, 설계자, 개발자가 DB 를 보는 관점에 따라 DB 를 기술하고
이들 간의 관계를 정의한 ANSI 표준.
- DB 의 독립성을 확보하기 위한 방법.
- 3 단계 계층으로 분리해서 독립성을 확보하는 방법으로 각 계층을
뷰(View)라고도 함.

※ 3 층 스키마의 독립성
-> 논리적 독립성 : 개념 스키마가 변경되도 외부 스키마가 영향 X
-> 물리적 독립성 : 내부 스키마가 변경되도 개념 스키마가 영향 X

(2) 3 층 스키마 구조

1) 외부 스키마
- 사용자 관점
- 여러 개의 외부 스키마 존재
- 응용 프로그램이 접근하는 DB

2) 개념 스키마
- 설계자 관점
- 사용자 전체 집단의 DB 구조
- 통합 DB 구조

3) 내부 스키마
- 개발자 관점

- DB 의 물리적 저장 구조

3. 엔터티(Entity)

(1) 엔터티

- 업무에서 관리해야 하는 데이터 집합, 저장되고 관리되어야 하는 데이터.
- 개념, 사건, 장소 등의 명사

(2) 엔터티 특징

1) 식별자

- 유일한 식별자가 있어야 함

2) 인스턴스 집합

- 2 개 이상의 인스턴스가 있어야 함

3) 속성

- 반드시 속성을 가지고 있음

4) 관계

- 다른 엔터티와 최소한 1 개 이상 관계가 있어야 함

5) 업무

- 업무에서 관리되어야 하는 집합

(3) 엔터티 종류

1) 유/무형에 따른 분류

- 유형 엔터티 : 물리적 형태, 지속적으로 사용
- 개념 엔터티 : 개념적 정보
- 사건 엔터티 : 비지니스 프로세스를 실행하면서 생성

2) 발생시점에 따른 분류

- 기본 엔터티 : 키 엔터티, 독립적으로 생성
- 중심 엔터티 : 기본 엔터티로부터 발생, 행위 엔터티를 생성
- 행위 엔터티 : 2 개 이상의 부모 엔터티로부터 발생, 자주 바뀌거나 양이 증가

4. 속성(Attribute)

(1) 속성

- 엔터티가 가지는 항목
- 더 이상 분리되지 않는 최소의 데이터 단위

(2) 특징

- 1 개의 속성은 1 개의 속성값을 가짐
- 주식별자에게 함수적으로 종속됨

- 1 개의 엔터티는 2 개 이상의 인스턴스 집합
- 1 개의 엔터티는 2 개 이상의 속성을 가짐

(3) 종류

1) 분해 여부에 따라
- 단일 속성 : 하나의 의미로 구성
- 복합 속성 : 여러 개의 의미가 있는 것 ex) 주소
- 다중값 속성 : 속성에 여러 개의 값을 가질 수 있는 것 ex) 상품 리스트

2) 특성에 따라
- 기본 속성
  -> 비지니스 프로세스에서 도출되는 본래의 속성

- 설계 속성
  -> 데이터 모델링 과정에서 발생
  -> 유일한 값을 부여 ex) 상품코드

- 파생 속성
  -> 다른 속성에 의해 만들어지는 속성 ex) 합계, 평균 등

※ 도메인
- 속성이 가질 수 있는 값의 범위
- 제약사항 지정

5. 관계(Relationship)

(1) 관계
- 엔터티 간의 관련성을 의미, 존재 관계와 행위 관계로 분류.

(2) 관계의 종류

1) 존재 관계
- 엔터티 간의 상태를 의미

2) 행위 관계
- 엔터티 간에 어떤 행위가 있는 것.

(3) 관계 차수(Cardinality)

1) 관계 차수
- 두 개의 엔터티 간에 관계에 참여하는 수

2) 관계 차수의 종류
- 1 : 1
- 1 : M
- M : N

3) 관계 체크사항
- 2 개의 엔터티 사이에 관심있는 연관 규칙 있는지?
- 2 개의 엔터티 사이에 정보의 조합 발생 하는지?
- 업무기술서 장표에 관계연결에 대한 규칙 서술 하는지?
- 업무기술서 장표에 관계연결을 가능케 하는 동사 있는지?

(4) 식별 관계와 비식별 관계

1) 식별 관계
- 강한 개체(실선 표기)
- 독립적으로 존재
- 기본키를 공유

2) 비식별 관계
- 강한 개체의 기본키를 다른 엔터티의 기본키가 아닌 일반 칼럼으로 관계를 가짐

6. 엔터티 식별자(Entity Identifier)

(1) 주식별자(기본키, PK)

1) 최소성
2) 대표성
3) 유일성
4) 불변성

※ 키의 종류
- 기본키(PK) : 후보키 중 엔터티를 대표할 수 있는 키
- 후보키 : 유일성 O, 최소성 O
- 슈퍼키 : 유일성 O, 최소성 X
- 대체키 : 여러 개의 후보키 중 기본키를 선정하고 남은 키
- 외래키(FK) : 하나 혹은 다수의 다른 테이블의 기본 키 필드를 가리키는 것으로
참조 무결성을 확인하기 위해 사용

(2) 비식별자(Non-Identifying Relationship)
- 부모 엔터티로부터 속성을 받았지만 자식 엔터티의 주식별자로 사용하지 않고
일반적인 속성으로만 사용하는 것

(3) 식별자의 종류

1) 대표성 여부
- 주식별자
-> 유일성 O, 최소성 O, 엔터티를 대표하는 식별자
-> 다른 엔터티와 참조 관계로 연결될 수 있음

- 보조 식별자
  -> 유일성 O, 최소성 O, 대표성은 만족 X

2) 자체 생성 여부
- 내부 식별자
  -> 엔터티 내부에서 스스로 생성
- 외부 식별자
  -> 다른 엔터티의 관계로 인하여 만들어지는 식별자

3) 속성의 수
- 단일 식별자
  -> 하나의 속성으로 구성
- 복합 식별자
  -> 두 개 이상의 속성으로 구성

4) 대체 여부
- 본질 식별자
  -> 비지니스 프로세스에서 만들어지는 식별자
- 인조 식별자
  -> 인위적으로 만들어지는 식별자
  -> 후보 식별자 중 주식별자로 선정할 것이 없거나 주식별자가 너무 많은 칼럼으로 되어 있는 경우 사용

<2 장. 데이터 모델과 성능>

1. 정규화

(1) 정규화(Normalization)
- 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위한 방법
- 데이터 중복 제거, 데이터 모델의 독립성 확보
- 실질적으로 제 3 정규화까지만 수행
- 엔터티의 의미 해석이 명확해짐
- 일반적으로 테이블의 수가 증가
- 정규화 수행 후 반정규화를 수행

(2) 함수적 종속성

1) 제 1 정규화
- 속성의 원자성 확보
- 기본키(PK) 설정
- 함수적 종속성을 근거

2) 제 2 정규화

- 기본키가 2 개 이상의 속성으로 이루어진 경우, 부분 함수 종속성 제거
- 기본키가 하나의 칼럼으로 이루어지면 제 2 정규화는 생략

3) 제 3 정규화
- 기본키(주식별자)를 제외한 칼럼 간에 종속성 제거
- 이행 함수 종속성 제거

4) BCNF(Boyce-Code Normal Form)
- 기본키를 제외하고 후보키가 있는 경우, 후보키가 기본키를 종속시키면 분해함
- 복수의 후보키가 있고, 후보키들이 복합 속성이어야 하며, 서로 중첩되어야 함

2. 정규화의 성능

(1) 정규화의 문제점
- 테이블을 분해해서 데이터 중복을 제거하기 때문에 데이터 모델의 유연석을 높임
- 데이터 조회시 조인을 유발하기 때문에 CPU 와 메모리를 많이 사용
- 조인으로 인해 성능이 저하되는 문제를 반정규화로 해결

3. 반정규화(De-Normalization)

(1) 반정규화
- DB 의 성능 향상을 위해 데이터 중복을 허용하고 조인을 줄이는 DB 성능 향상 방법

※ 일반적으로
- 정규화시 입력/수정/삭제 성능 향상
- 반정규화시 조인 성능 향상

(2) 반정규화를 수행하는 경우
1) 수행 속도가 느려지는 경우
2) 다량의 범위를 자주 처리해야 하는 경우
3) 특정 범위의 데이터만 자주 처리하는 경우
4) 요약/집계 정보가 자주 요구되는 경우

※ 반정규화 절차
- 대상 조사 및 검토 : 데이터 처리 바무이, 통계성 등을 확인해서 대상을 조사
- 다른 방법 검토
  -> 반정규화 수행 전 다른 방법이 있는지 검토
  -> ex) 클러스터링, 뷰, 인덱스 튜닝, 응용 프로그램, 파티션 등을 검토
- 반정규화 수행 : 테이블, 속성, 관계 등을 반정규화 함

※ 클러스터링(Clustering)

- 클러스터링 인덱스는 인덱스 정보를 저장할 때 물리적으로 정렬해서 저장하는 방법
- 조회시 인접 블록을 연속적으로 읽기 때문에 성능이 향상

(3) 반정규화 기법

1) 계산된 칼럼 추가

2) 테이블 수직분할 : 하나의 테이블을 두 개 이상의 테이블로 분할

3) 테이블 수평분할 : 하나의 테이블에 있는 값을 기준으로 테이블을 분할

※ 파티션(Partition) 기법

파티션을 사용하면 논리적으로는 하나의 테이블이지만 여러 개의 데이터 파일에 분산되어서 저장

- Range Partition : 범위를 기준
- List Partition : 특정 값을 지정
- Hash Partition : 해시 함수를 적용
- Composite Partition : 범위와 해시를 복합적으로 사용

-> 데이터 조회시 범위가 줄어 성능이 향상
-> 데이터가 분할되어 있으므로 I/O 성능이 향상
-> 각 파티션을 독립적으로 백업 및 복구 가능

4) 테이블 병합

- 1 : 1 관계의 테이블을 하나의 테이블로 병합해서 성능을 향상
- 1 : N 관계의 테이블을 병합하여 성능을 향상(데이터 중복이 발생)
- 슈퍼 타입과 서브 타입 관계가 발생하면 테이블을 통합하여 성능을 향상

※ 슈퍼 타입 및 서브 타입 변환 방법

- OneToOne Type
  -> 슈퍼 타입과 서브 타입을 개별 테이블로 도출
  -> 테이블의 수가 많아 조인이 많이 발생하고 관리가 어려움
- Plus type
  -> 슈퍼 타입과 서브 타입 테이블로 도출
  -> 조인이 발생하고 관리가 어려움
- Single Type
  -> 슈퍼 타입과 서브 타입을 하나의 테이블로 도출
  -> 조인 성능이 좋고 관리가 편리, 입출력 성능은 나쁨

4. 분산 데이터베이스

(1) 분산 데이터베이스

- 여러 곳으로 분산되어 있는 DB 를 하나의 가상 시스템으로 사용할 수 있도록 한 DB

- 논리적으로 동일한 시스템에 속하지만, 컴퓨터 네트워크를 통해 물리적으로 분산되어
있는 데이터 집합

※ 분산 데이터베이스의 투명성 종류

- 분할 투명성(단편화)
  -> 하나의 논리적 Relation 이 여러 단편으로 분할되어 각 사본이 여러 Site 에 저장
- 위치 투명성
  -> 사용하려는 데이터의 저장 장소 명시 불필요 위치정보가 , 시스템 카탈로그에 유지
- 지역사상 투명성
  -> 지역 DBMS 와 물리적 DB 사이의 Mapping 보장
- 중복 투명성
  -> DB 객체가 여러 Site 에 중복되어 있는지 알 필요 없는 성질
- 장애 투명성
  -> 구성요소의 장애와 무관한 트랜잭션의 원자성 유지
- 병행 투명성
  -> 다수 트랜잭션 동시 수행시 결과의 일관성 유지
  -> TimeStamp, 분산 2 단계 Locking 이용

(2) 분산 데이터베이스 설계 방식

1) 상향식 설계 방식
   - 지역 스키마 작성 후 향후 전역 스키마를 작성하여 분산 데이터베이스를 구축

2) 하향식 설계 방식
   - 전역 스키마 작성 후 해당 지역 사상 스키마를 작성하여 분산 데이터베이스를 구축

(3) 분산 데이터베이스 장점과 단점

1) 장점
   - 신뢰성, 가용성 높음
   - 빠른 응답 속도
   - 시스템 용량 확장이 쉬움
   - 통신 비용 절감

2) 단점
   - 관리와 통제가 어려움
   - 데이터 무결성 관리가 어려움
   - 설계가 복잡
   - 처리 비용 증가