210728 데이터모델링_2

생성일	@2021년 7월 28일 오전 9:23
성함	나한주
속성	
수업 유형	이론
수정일	@2021년 7월 29일 오전 1:03
작성자	㊊ 현동빈

3교시. 작업절차 및 기본기법

Chapter3. 속성분석

▼ 속성 분석을 위한 질문

- 세분화 → 잘게 찢어졌는가?
- 일관적, 배타적
- 같은 어커런스에 대해 복수의 값?
- 복수 어커런스가 같은 값이면 업무의 의미 있는지

- 도출 가능 → 모든 어커런스에 적용 가능?

▼ 질문

- 분리가능 → 최대한 분리
- 일관성 & 배타적 → 배타적이지 않을 때, 속성을 분리
- 속성이 같은 어커런스에 대해 복수의 값을 갖는가? → 변별력 하락, 만약 그렇다면 별도의 엔티티로 도출
- 복수 어커런스가 같은 이 속성에 대해 같은 값일 때 그 값들의 의미가 있는지 ? → 만약 그렇다면 별도의 엔티티로 분리
- 도출 가능한 속성이라면 그 값이 모든 어커런스에 적용 가능? → 도출 가능한 속성은 제외

Chapter4. 정규화

▼ 정의

- 효율적인 관리를 위한 데이터구조를 도출하기 위하여, 속성간의 관계를 분석하여 별도 엔티티 도출 여부를 결정해 나가는 절차에 관련된 이론

▼ 효과

- 기본 원칙 : 하나의 테이블에는 중복된 데이터가 없도록

  - 데이터 중복에 따른 정합성 오류 발생가능성 제거

  - 성능 개선(단, 전체 시스템을 고려해야 함)

▼ 규칙

- 제 1 정규화 → 제 2 정규화 → 제 3 정규화
- 제 1 정규화 : 반복 또는 복수 값을 갖는 속성의 제거 (중복 데이터 제거)
- 제 2 정규화 : 기본키에 종속 되지 않는 속성 제거(기본키의 일부에 종속되는 속성의 제거)
- 제 3 정규화 : 기본키가 아닌 속성에 종속적인 속성의 제거

▼ 제 1 정규화

- 반복그룹의 속성을 별도의 엔티티로 분리

- 작업절차

  - 반복하는 속성 그룹으로 새로운 엔티티 작성

  - 새로운 엔티티 키(복합 키 - Concatenate Key) 및 관계 정의

▼ 제 2 정규화

- 기본키에 종속되지 않는 모든 속성을 분리하여 새로운 엔티티 구성
- 복합키(Concatenated Key)가 있는 엔티티에 적용됨
- 작업절차

  - 속성 중에 Key의 일부에만 종속하는 속성 분리

  - 새로운 엔티티 정의

  - 새로운 엔티티 키 및 관계 정의

▼ 제 3 정규화

- 엔티티의 속성 중에 기본 키가 아닌 속성에 종속하는 모든 속성을 분리하여 새로운 엔티티 생성
- 작업절차

  - 속성 중 Key가 아닌 속성에 종속하는 속성을 분리

  - 새로운 엔티티 정의

  - 새로운 엔티티 키 및 관계 정의

5교시. 프로젝트 실무적용

Chapter1. 논리/물리 모델 구분하기

▼ 개념모델

- 데이터 모델의 첫 단계
- 고객이 필요로 하는 요구사항을 수집, 분석해 데이터 모델의 전체적인 모양 결정
- 엔티티와 관계 위주로 모델링이 이루어짐 (세부적인 속성이나 정확한 식별자는 이후 논리 모델링 단계에서 정해짐)
- 목표 : 프로젝트의 비즈니스적인 내용에 대해 고객과 모델러가 합의를 이루는 것

▼ 논리모델

- 개념 모델을 기반으로 논리적인 구성요소를 빠짐없이 정의하는 단계

- 비즈니스에 필요한 모든 내용이 명확히 정의되어야만 논리 모델이 완료(모든 엔티티, 관계, 속성, 식별자)
- 정의된 모든 내용들은 정규화라는 단계를 통해 정제된 상태여야 함
- 정규화 완료 → 전체 모델 내에 중복된 내용이 없음
- 데이터 모델의 엔티티와 서브타입은 논리적인 집합

▼ 물리모델

- 구현을 위한 물리적인 구성과 실제 구현을 위한 설계
- 논리적 모델을 특정 데이터베이스로 설계함으로 생성된, 데이터를 저장할 수 있는 물리적인 스키마

Chapter2. 업무 분석하기

▼ 업무데이터 분석 과정

- 업무의 이해 → 업무 데이터의 이해 → 데이터 구조 모델링 → DB 생성

Chapter3. 또 다시 Entity

▼ 범주화

- 유사한 것들을 일정하게 묶는 프로세스

▼ 추상화

- 문제 영역에서 가장 핵심적인 특성만을 추리는 과정

▼ 데이터모델링?

- 엔티티 : 업무 수행에 필요한 데이터를 성격이 유사한 것끼리 모아놓은 집합
- 모델링 : 정보를 담는 최소 단위인 속성의 종속성을 분석하여 유사한 것끼리 모으고 독립적인 것은 분리하는 과정
- 이러한 과정에서 속성의 엔티티 찾기 가능
- 어떤 개체가 속하는 범주를 규명하여 개체 집합을 분리하고 묶는 수준을 고민하는 과정

▼ Entity모델링의 고충

- 데이터 집합을 정의하기 쉽지 않음
- 데이터의 본질을 볼 줄 알아야 함
- 추상화 수준의 결정

- 하위 트랜잭션 데이터로 상위 논리집합 발견

Chapter4. 메타데이터의 관리

▼ 데이터 표준화

- 표준화 : 이음동의어와 동음이의어 관리
- 표준용어 : 표준 단어들의 조합 (복합어의 허용 제한 필요)
- 메타데이터 관리시스템

▼ 코드

a. 코드란?

- 속성에 사용되는 값을 간단하고 알기 쉽게 나타낼 수 있도록 약속한 또 다른 값
- 짧은 설명이나 텍스트를 더 짧은 상징으로 표현한 것

b. 코드를 사용하는 이유

- 데이터가 일관적으로 유지됨
- 데이터를 구분하기 위해서 사용
- 저장 공간을 줄여주는 부수적인 효과를 얻을 수 있음

c. 코드를 정의할 때 유의점

- 하나의 코드에 여러의미가 포함되지 않도록
- 성질이 다르면 다른 코드로 부여

d. 속성 코드와 식별자 코드

- 코드 속성은 코드값이(ex.01) 사용된 속성 의미
- 식별자 코드는 엔티티의 주 식별자로 속성 이름이 "~코드"로 끝난 속성, 코드 속성과 식별자 코드는 성격이 다름
- 식별자 코드의 속성명에만 "~번호" 등으로 변경해도 문제 없음

e. 코드 엔티티의 주의점

- 실체 엔티티는 처음에는 속성이 적어 코드 엔티티처럼 보이지만 코드 엔티티가 아니며, 추후 속성이 추가될 가능성이 크므로 공통 코드 엔티티에 포함시키지 말고 별도의 엔티티로 도출하는 것이 좋음
- 인스턴스가 매우 많은 코드성 데이터도 공통 코드 엔티티에 포함시키지 않는 것이 좋음(메모리 효율 등을 위해서)

- 일반 코드(속성 코드)가 식별자 코드가 될 경우, 공통 코드 엔티티에서 관리하던 인스턴스는 삭제해야함
- 경합을 줄이기 위해 공통 코드 엔티티에서 별도의 엔티티로 도출하는 경우가 있음 (다만, 별다른 이유없이 공통 코드 엔티티에서 관리해도될 인스턴스를 별도의 엔티티에서 관리할 경우 저장공간 낭비 발생)
- 식별자의 경우 부서번호, 지점번호, 대리점번호 등과 같이 번호를 사용하고, 일반 코드(속성 코드)의 경우 "~구분코드", "~유형코드", "~종류코드"와 같이 사용하는 것을 권장
- 참고

  - 구분코드 : 성질이나 특징이 다른 코드명이 고정적일 때 사용 ex. 남녀구분코드, 매수매도구분코드

  - 유형코드 : 코드명을 성질이나 특징이 공통적인 것끼리 묶을 때 사용 ex. 거래유형코드

  - 종류코드 : 고정적이지 않고 지속적으로 늘어날 수 있는 코드명을 나열할 때 사용 ex. 서비스종류코드
- 참고 : http://www.gurubee.net/lecture/3632

6. 개별코드와 통합코드

- 개별코드 : 유형만 관리, 일반화 가능
- 통합코드 : 실체 관리
- 데이터를 통합 관리함 → 등록/참조/분배 등 관리상의 장점
- 코드 사용빈도가 하나의 테이블에 응집 - 메모리 성능상의 이점 기대
- 어플리케이션 아키텍쳐를 고려한 선택 필요

Chapter5. DBMS

- 관계형 DBMS

Chapter6. 데이터의 품질관리

Chapter7. 데이터모델링 도구