**데이터 모델링 개념**

**데이터 모델링이란?**

- 현실 세계에 대해 우리가 관심있어하는 대상을 데이터베이스화 하기 위한 개념적 도구

**잘된 데이터 설계**

- 데이터 중복 배제, 일관성 확보, 프로그램 유연성 확보

**모델링 절차**

- 개념적 데이터 모델링 -> 논리적 데이터 모델링 -> 물리적 데이터모델링

**개념적 데이터 모델링**

- 주제 영역을 정의하고 주제 영역내의 핵심 엔터티를 도출하여 그들간의 관계를 정의하는 것

\*주제영역, 핵심엔티티, 관계

- 주제 영역이란 부서의 분류하는 것과 유사. (최상위 개념적 데이터 모델링)

- 엔터티란 데이터를 담는 그릇이며 물리 모델에서 테이블로 전환. 핵심 엔터티란 엔터티 분류 중 키 엔터티, 메인 엔터티, 액션 엔터티 중 액션 엔터티를 제외한 나머지 부분.

|  |  |  |
| --- | --- | --- |
| 개념 데이터 모델 | 논리 데이터 모델 | 물리 데이터 모델 |
| 주제영역 정의  엔터티 후보 식별  핵심 엔텉 정의  관계 정의 | 속성 정의  엔터티 상세화  식별자 확정  정규화  M;M 관계 해소  참조무결성 정의  이력관리 정의 | 논리–물리 데이터 모델 변환  반정규화 |

**데이터 표준화**

- 구성 요소 : 표준 단어, 표준 용어, 표준 도메인

- 표준 단어/용어 : 속성명과 컬럼명의 표준화

- 표준 도메인 : 데이터 타입과 길의 및 값을 표준화하여 일관된 data 생성 유도

**전사 아키텍처**

- IT는 건축과 달리 가시성이 안좋으므로 전사 아키텍처는 현재의 모습을 가시화하여 관리를 가능하게 하는 것이다.

- 전사아키텍처 정책 수립, 전사아키텍처 정보 수립, 전사아키텍처 관리 정책 수립이 있다.

**데이터 모델링**

**데이터 모델링 탄생 배경**

1. 정보시스템의 핵심인 데이터를 어떻게 중복 없이 관리 할 것인가?

2. 정확한 정보 시스템 구축을 위해서 설계/개발 보다는 정확한 업무의 업무파악 (데이터에 대한 정확한 분석) 이 선결되어야 한다는 결론

3. 현실 세계를 좀 더 잘 표현할 수 있는 ERD으로 발전

**모델의 정의**

- 현실 세계에 대해 우리가 관심있어 하는 대상을 데이터베이스화 하기 위한 개념적 도구

**모델링의 정의**

- 기업 업무에 대한 종합적인 이해를 바탕으로

- 데이터에 존재하는 업무 규칙에 대해 참 또는 거짓을 판별할 수 있는 사실을 정의

- 어떻게, 누가 접근하는지에 대한 전산화와는 별개의 관점

- 현재 업무를 파악하여 문제점을 인식하고

- 개선사항을 도출하여 미래에 적합한 설계를 이끌어 내기 위해

- 인간이 해야할 대부분의 결정을 내리는 단계까지를 모두 포함하는 것.

**데이터 모델이 제공하는 것**

- 시스템을 현재 또는 원하는 모습으로 가시화하도록 해준다.

- 시스템의 구조와 행동을 명세화할 수 있게 한다.

- 시스템을 구축하는 틀을 제공

- 우리가 결정한 것을 문서화한다.

- 다양한 영역에 집중하기 위해 다른 영역의 세부사항은 숨기는 다양한 관점을 제공

- 특정 목표에 따라 다양한 상세 수준을 제공한다.

**데이터 모델의 필요성**

고품질의 데이터 모델 : 시스템의 안정성, 유연성, 성능에 지대한 영향.

-> 고품질의 데이터 모델링은 시스템 개발에 있어서 가장 핵심적인 과정

프로세스 중심적인 시스템 구축 방법

- 고품질의 데이터 모델을 설계하기 위한 단계에서 정보 공유의 문제점 다수 발생

- 데이터 무결성에 좋지 않은 영향 -> 즉, 데이터 품질에도 악영향

-> 시스템이 효과적이고 효율적으로 운영되기 위해서는 기능별 시스템을 가로지르는 시스템 간의 상호작용과 데이터 통합이 중요하다.

데이터 모델(설계)이 중요한 이유

- 파급효과, 간결한 표현, 데이터 품질

어플리케이션과 데이터의 통합

- 어플리케이션 코딩 차원의 통합에 너무도 많은 비용 초래

- 데이터를 기반으로 한 통합은 효과적이고 저비용이며 성공적으로 완수하기 위한 필요조건

**개발자들의 시스템 이해**

- 개발자들이 개발할 시스템과 데이터를 좀 더 확실하게 이해하기 위해 데이터의 모형화 필요

- 사용자 관점 데이터: 사용자가 원하는 논리적 개념(View), 시스템이 어떻게 그것을 제공하는 지의 물리적 개념을 명확하게 표현 가능

- 물리적 표현 또는 사용에 관계없는 데이터 그 자체의 본질

- 어플리케이션 간 데이터 사용

**데이터 모델링 시 주의점**

- 중복배제, 유연성, 일관성이 있어야 한다!

**데이터 모델링 단계**

- 개념 데이터 모델링 : 주제영역을 정의하고 주제영역 내의 핵심 엔터티를 도출하고 그들 간의 관계를 정의하고 이를 표현하기 위해 ERD 작성

- 논리 데이터 모델링 : 상세 속성 정의, 엔터티 상세화, 이력관리 결정

- 물리 데이터 모델링 : 목표하는 DBMS 특성 및 구현 환경 등을 감안한 스키마를 일정한 기준과 규칙에 의해 도출. ex) 데이터 사용량 분석. 인덱스 정의. 반정규화 수행

**모델링 기본 원칙**

- 커뮤니케이션 원칙 : 요구사항이 정의한 그대로 쓸 것

- 모델링 상세화 원칙 : 데이터의 본질과 잠재적 사용을 이해할 수 있을만큼 상세화해야함.

- 논리적 표현 원칙 : 물리적 제약조건 없이 비즈니스를 그대로 반영.

-> 논리적 설계와 물리적 설계를 구별하지 못하면 잘못된 방향으로 진행하게 됨

좋은 데이터모델의 요소

- 완전성, 중복 배제, 비즈니스 룰, 데이터 재사용, 안정성 및 확장성, 간결함, 의사소통, 통합성

**데이터 모델링 기법 이해**

**데이터 모델 목적**

- 데이터 베이스 설계에 대한 계획 또는 청사진

- 시스템의 데이터 구조에 대한 형상을 이해하고 원활한 소통을 도모하게 함

- 다르게 인식하고 있는 뷰들을 하나로 통합할 수 있는 단일화된 설계안을 만들 수 있음

- 서로 다른 뷰를 충족시킬 수 있는 데이터 처리와 제약조건 등의 요구사항을 정의하기 위해

**개체 – 관계 모델 기법**

- 요즘은 서브타입이 추가된 확장형을 그냥 ERD라 함

- ERD를 사용하면 뷰를 하나로 통합할 수 있으며 서로 다른 뷰를 충족시킬 수 있는 데이터 처리와 제약조건 등의 요구사항을 정의 할 수 있다.

**개체 – 관계 모델 구성 요소**

- 엔터티, 속성, 식별자, 관계, 카디널리티, 존재종속, 서브타입

**객체지향**

- 인스턴스 = Row, Record, 튜플. / 엔터티 = Relation, 테이블. / 식별자 = PK = Key

- 객체지향 장점은 재사용 코드와 같은 개념이 실제로 가능한 환경, 모든 비즈니스 규칙이 표현될 수 있는 유일한 환경을 제공하며, 프로세스와 데이터 모델링을 함께 운영.

**\*서브타입 관련 문구**

- 개괄적으로 봤을 경우 서브타입은 배타적 혹은 포괄적 (IE도 마찬가지).

- 그러나 바커 표기법은 서브타입의 중복을 허락하지 않는 상호 배타적.