# 1 데이터 모델링의 이해

## 데이터 모델의 이해

1. 데이터 모델의 이해

정의 : 일정한 표기법에 의해 표현해 놓은 모형

- 추상화(모형화), 단순화., 명확화
  - 추상화
    - 모형화 또는, 가설적 이라고 표현할 수 있음
    - **일정한 양식**인 **표기법**에 의해 표기
  - 단순화
    - 제약된 표기법
    - 쉽게 이해할 수 있도록 하는 개념
  - 명확화
    - 애매모호함을 제거
    - 정확하게 현상을 기술

** 정보시스템 구축에서 모델링 활용

- **계획, 분석, 설계** 시 **분석 및 설계**에 사용
- **구축, 운영** 단계에서는 **변경 및 관리**의 목적으로 이용

모델링의 세 가지 관점 : **모델링 = 데이터 관점(What) + 프로세스 관점(How)**, 더불어, 상호 관계적 의미 규명을 통해 **상관관점** 또한 정의 내릴 수 있다

- 데이터 관점이란, 업무가 **어떤 데이터와** 관련이 있는 지 더불어, **데이터 간의 관계**는 무엇인지에 대해서 모델링 하는 방법.
- 프로세스 관점이란, 업무가 **실제하고 있는 일**은 무엇인지 더불어,. **무엇을 해야 하는지**를 모델링하는 방법

2. 데이터 모델의 기본개념의 이해

데이터 모델의 정의 :

- **정보시스템** 구축을 위한, **데이터 관점**의 업무 **분석 기법**
- **데이터(What)에 대해 약속된 표기법에 의해 표현하는 과정**
- **데이터베이스를 구축하기 위한 분석/설계의 과정**

데이터 모델이 제공하는 기능

- 시스템을 **가시화**에 도움
- 시스템의 구조와 행동을 **명세화**
- 시스템을 구축하는 **구조화된 틀**을 제공
- 시스템을 구축하는 과정에서 결정된 것을 **문서화**
- 다양한 영역에 집중하기 위해 다른 영역의 세부 사항은 숨기는 **다양한 관점을 제공**
- **상세 수준의 표현방법을 제공**

3. 데이터 모델링의 중요성 및 유의점
   - 파급효과(Leverage)
     병행, 통합테스트를 수행 중, 만약 모델의 변경이 불가피한 경우에 대하여
     **데이터 모델의 형태에 따라서 그 영향 정도는 차이가 있지만, 데이터 구조의 변경으로 인한 일련은 변경작업은 전체 시스템 구축 프로젝트에서 큰 위험요소**
   - 복잡한 정보 요구사항의 간결한 표현(Conciseness : 간결)
     **데이터 모델 : 구축할 시스템의 정보 요구사항과 한계를 가장 명확하고 간결하게 표현할 수 있는 도구**
   - 데이터 품질(Data Quality)

데이터 모델링을 할 때 유의점
- 중복 ( Duplication ) : 데이터베이스가 여러 장소에 같은 정보를 저장하는 것을 유의
- 비유연성 ( Inflexibility ) : 사소한 업무변화에 대해, 데이터 모델이 수시로 변경된다면 유지보수의 어려움이 가중될 수 있음
  - **해결책 : 데이터의 정의**를 데이터의 사용 **사용 프로세스**와 *** 분리
- 비일관성 ( inconsistency ) : 유의할 것은 데이터의 중복이 없더라도 비일관성은 발생할 수 있음
  - **해결책 : 데이터 모델링 시 데이터와 데이터간 상호 연관 관계에 대한 명확한 정의는 사전에 예방가능**

**4.** 데이터 모델링의 3 단계 진행
   - 시간에 따라 진행되는 과정
   - 추상화 수준에 따라 **개념적, 논리적, 물리적 데이터 모델**로 정리

**시간에 따른 진행과정 특징정리**
** 개체와 데이터베이스는 상호 교류한다.
** 개체의 개념적 모델링을 통해 개념적 구조가 성립
** 개념적 구조의 논리적 모델링을 통해 논리적 구조가 성립
** 개념적 구조와 논리적 구조는 상호 교류한다.
** 논리적 구조의 물리적 모델링을 통해 데이터베이스가 성립

**각 데이터 모델링에 대한 특징정리**

**** 개념적 데이터 모델링 : 추상화 수준이 높고, 업무중심적, 포괄적이 수준의 모델링 진행**
**EA(Enterprise Architecture)수립, 전사적 데이터 모델링에 많이 이용된다.**

전사적 데이터 모델링 : 데이터 모델링 과정이 전 조직에 걸쳐 이루어 진다면, 전사적 데이터모델링 이라고한다.

**** 논리적 데이터 모델링 : (시스템으로 구축하고자 하는 업무에 대해)_ Key, 속성, 관계 등을 정확하게 표현, 재사용성이 높음**

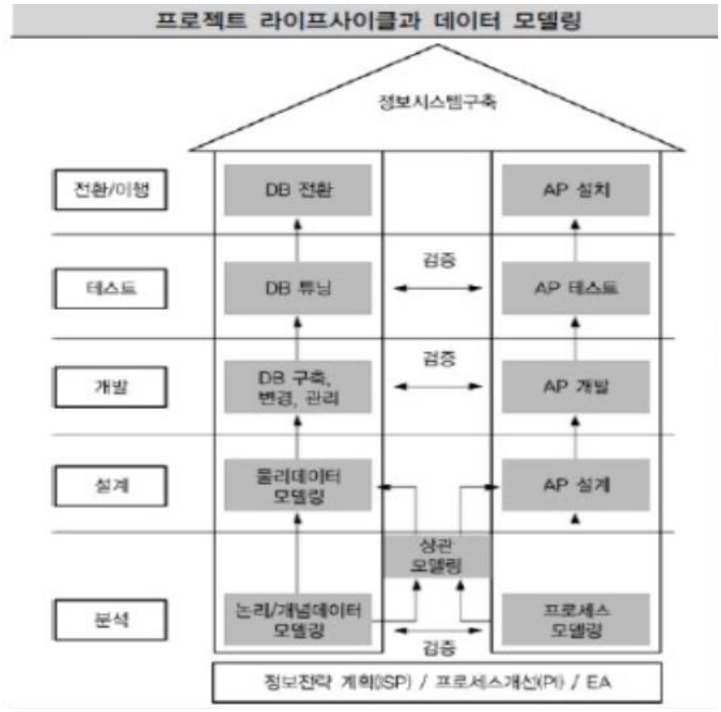
**** 물리적 데이터 모델링 : 실제로 데이터베이스에 이식할 수 있도록 성능, 저장, 등 물리적인 성격을 고려하여 설계**

**5.** 프로젝트 생명주기( Life Cycle )에 대해서 데이터 모델링

Water-fall 기반 : 데이터 모델링이 분석, 설계단계로 구분되어 명확하게 정의

- 정보공학이나 구조적 방법론
  1) **분석단계**에서, **논리적인 데이터 모델링**을 수행
  2) **설계단계**에서 **하드웨어와 성능을 고려**한 **물리적인 데이터 모델링**을 수행
- 나선형모델
  1) 업무크기에 따라, **논리적 데이터 모델**과 **물리적 데이터 모델**이 분석
  2) **통상, 분석단계에서 논리적인 데이터 모델링이 더 많이 수행**됨
  3) ** **데이터** 측과 **어플리케이션** 측으로 구분되어 프로젝트가 진행되면서 각각에 도출된 사항은 **상호 검증을 지속적으로 수행**하면서 단계별 완성도를 높이게 된다.

일반적으로는
개념, 분석단계 > 개념적 데이터 모델링
분석단계 > 논리적 데이터 모델링
설계단계 > 물리적 데이터 모델링이 수행
단, 현실 프로젝트에서는 개념적 데이터 모델이 생략된 개념/논리 데이터 모델링이 분석단계 때 대부분 수행된다.



단, 객체지향 개념은 데이터 모델링과 프로세스 모델링을 구분하지 않고 일체형으로 진행
(대표적인 예가 데이터(속성)와 프로세스(Method)가 같이 있는 클래스(Class))
∵ 데이터와 프로세스를 한꺼번에 바라보면서 모델링 전개

6. 데이터 모델링에서 데이터 독립성의 이해
   - 데이터 독립성의 필요성
     **유지보수, 중복성, 복잡도, 요구사항 대응 저하** 등의 이유로 인해, 독립성은 필요

**독립성의 효과**
** 각 View 의 독립성을 유지, 계층별 View 에 영향을 주시 않고 변경이 가능
** 단계별 Schema 에 따라, 정의어(DDL), 조작어(DML)가 다름을 제공
**ANSI 표준모델 : 구조, 독립성, 사상(Mapping)_** 이와 같이 데이터독립성을 3 단계로 표현, 이해할 수 있다.

*** 데이터베이스 3 단계 구성
ANSI/SPARC 3 단계 구성의 데이터 독립성 모델 : 외부단계, 개념적단계, 내부적단계로 **서로 간섭되지 않는 모델을 제시**
- 내부-개념 사이 물리적 데이터독립성
- 개념-외부 사이 논리적 데이터독립성

**다. 데이터독립성 요소**

[표 Ⅰ-1-2] 데이터독립성 구성요소

| 항목 | 내용 | 비고 |
|---|---|---|
| 외부스키마 (External Schema) | - View 단계 여러 개의 사용자 관점으로 구성, 즉 개개 사용자 단계로서 개개 사용자가 보는 개인적 DB 스키마<br>- DB의 개개 사용자나 응용프로그래머가 접근하는 DB 정의 | 사용자 관점 접근하는 특성에 따른 스키마 구성 |
| 개념스키마 (Conceptual) Schema | - 개념단계 하나의 개념적 스키마로 구성 모든 사용자 관점을 통합한 조직 전체의 DB를 기술하는 것<br>- 모든 응용시스템들이나 사용자들이 필요로 하는 데이터를 통합한 조직 전체의 DB를 기술한 것으로 DB에 저장되는 데이터와 그들간의 관계를 표현하는 스키마 | 통합관점 |
| 내부스키마 (Internal Schema) | - 내부단계, 내부 스키마로 구성, DB가 물리적으로 저장된 형식<br>- 물리적 장치에서 데이터가 실제적으로 저장되는 방법을 표현하는 스키마 | 물리적 저장구조 |

데이터 베이스의 스키마 구조 : 3 단계로 구분하며, 각각은 상호 독립적인 의미와 고유한 기능을 소유함
++ 데이터 모델링은 통합관점의 뷰를 가지고 있는 개념 스키마를 만들어가는 과정

두 영역의 데이터 독립성

**라. 두 영역의 데이터독립성**

논리적인 독립성 &물리적인 독립성 : 3단계로 개념의 각 영역에 대한 독립성을 지정하는 용어

[표 Ⅰ-1-3] 논리적, 물리적 데이터독립성

| 독립성 | 내용 | 특징 |
|---|---|---|
| 논리적 독립성 | - 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원하는 것<br>- 논리적 구조가 변경되어도 응용 프로그램에 영향 없음 | - 사용자 특성에 맞는 변경가능<br>- 통합 구조 변경가능 |
| 물리적 독립성 | - 내부스키마가 변경되어도 외부/개념 스키마는 영향을 받지 않도록 지원하는 것<br>- 저장장치의 구조변경은 응용프로그램과 개념스키마에 영향 없음 | - 물리적 구조 영향 없이 개념구조 변경가능<br>- 개념구조 영향 없이 물리적인 구조 변경가능 |

논리적인 데이터 독립성 : 외부의 변경에도 개념스키마가 변하지 않는 특징

사상( Mapping ) : 상호 독립적인 개념을 연결시켜주는 다리

크게 – 외부적/개념적 사상과, - 개념적/내부적 사상 도출

[표 Ⅰ-1-4] 사상(Mapping)

| 사상 | 내용 | 예 |
|---|---|---|
| 외부적/개념적 사상 (논리적 사상) | - 외부적 뷰와 개념적 뷰의 상호 관련성을 정의함 | 사용자가 접근하는 형식에 따라 다른 타입의 필드를 가질 수 있음. 개념적 뷰의 필드 타입은 변화가 없음 |
| 개념적/내부적 사상 (물리적 사상) | - 개념적 뷰와 저장된 데이터베이스의 상호관련성 정의 | 만약 저장된 데이터베이스 구조가 바뀐다면 개념적/내부적 사상이 바뀌어야 함. 그래야 개념적 스키마가 그대로 남아있게 됨 |

논리적 사상 : 외부 화면이나 사용자에게 인터페이스하기 위한 스키마 구조는 전체가 통합된 개념적 스키마와 연결된다는 것

물리적 사상 : 통합된 개념적 스키마 구조와 물리적으로 저장된 구조의 물리적인 테이블스페이스와 연결되는 구조

7. 데이터 모델링의 중요한 3 가지 개념 : Things < Attributes < Relationships
   - 단수와 복수(집합)의 명명

[표 Ⅰ-1-5] 용어의 구분정의

| 개념 | 복수/집합개념 타입/클래스 | 개별/단수개념 어커런스/인스턴스 |
|---|---|---|
| 어떤 것 (Thing) | 엔터티 타입(Entity Type) | 엔터티(Entity) |
| | 엔터티(Entity) | 인스턴스(Instance), 어커런스(Occurrence) |
| 어떤 것간의 연관 (Association between Things) | 관계(Relationship) | 페어링(Pairing) |
| 어떤 것의 성격 (Characteristic of a Thing) | 속성(Attribute) | 속성값(Attribute Value) |

– 어떤 것 : 실제 실무 현장에서는 복수/집합개념도 엔터티로 짧게 명명
엔터티를 집합개념으로 사용하는 경우, 개별요소에 대해서는 인스턴스/어커런스를 단수의 개념으로 사용
관계(Relationship) : 일반적으로 단수든 복수든 대부분 관계라고 표현

8. 데이터 모델링의 이해관계자

**이해관계자의 데이터 모델링 중요성 인식**

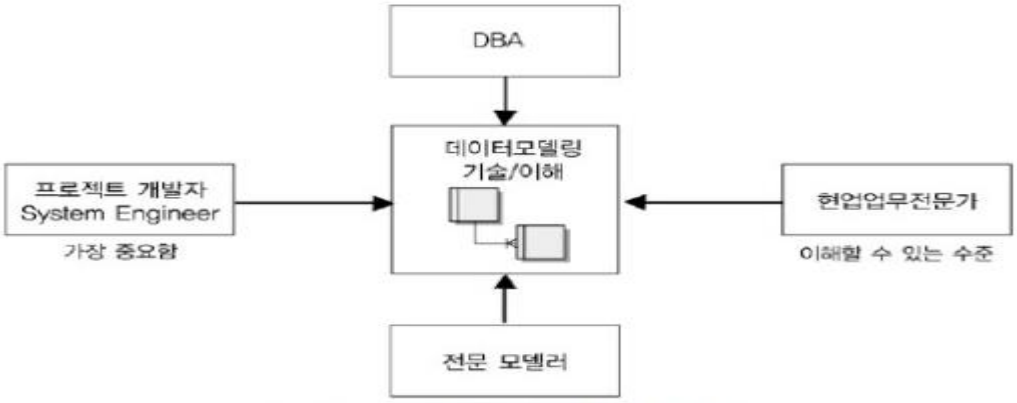
- 실질적으로 DBA 보다 업무시스템을 개발하는 응용시스템 개발자가 데이터 모델링도 같이하는 경우가 많음

◦ 대부분의 기업에 있어서 정보시스템의 데이터베이스 구조는 사용자에게 숨겨진 형태로 구축되어 왔다.
→ 정보의 고립화


◦ 프로그램은 6가지 유형의 데이터베이스 유지절차 – C. Finkelstein
◦ Programmer is the Navigator in the sea of data. – Bachmann

[그림 Ⅰ-1-7] 자료 처리의 중심은 데이터

- 데이터 모델링의 이해관계자
  : 서로가 프로젝트 수행 중에 의사소통을 잘 할 수 있고 업무를 잘못 해석하여 잘못된 시스템을 구축하는 위험(Risk)을 줄일 수 있음. 업무를 가장 잘 알고 있는 사람이 가장 훌륭한 모델러가 될 수 있음



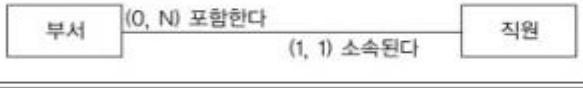
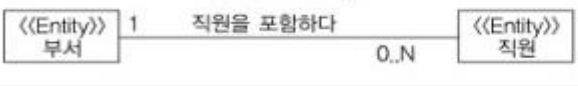

[그림 Ⅰ-1-8] 데이터 모델 이해관계자

1. 데이터 모델의 표기법인 ERD 의 이해

- 데이터 모델 표기법

1976년 피터첸(Peter Chen)이 Entity-relationship model(E-R Model)이라는 표기법

[표 Ⅰ-1-6] 표기법

| 표기법 | 설명 |
|---|---|
| Chen<br>  | - 대학교재에서 가장 많이 이용하는 표기법<br>- 실무적으로 사용안함 |
| IDEF1X<br>  | - 마름모와 원을 이용한 표기법으로 실무현장에서는 소수 활용<br>- ERWin |
| IE/Crow's Foot<br>  | - 까마귀발 모양의 표기법으로 가장 많이 사용함<br>- ERWin, ERStudio |
| Min-Max/ISO<br>  | - 기수성을 좀 더 정교하게 표현한 방법으로 많이 활용안됨 |
| UML<br>  | - 스테레오타입을 이용하여 엔터티 표현<br>- UML로 표현하여 데이터 모델링 할 때 사용<br>- Rational Rose |
| Case*Method/Barker<br>  | - Crow's Foot을 적용하면서 관계 표기법 등 일부 다름(Barker's Notation)<br>- DA# |

좋은 데이터 모델의 요소

가. 완전성(Completeness)

나. 중복배제(Non-Redundancy)

다. 업무규칙(Business Rules)

라. 데이터 재사용(Data Reusability)

기업이 관리하고자 하는 데이터를 합리적으로 균형이 있으면서도 단순하게 분류하는 것

마. 의사소통(Communication)

바. 통합성(Integration)

완.중.규.재.의.통

# 데이터 모델링의 이해
## 엔터티

1. 엔터티의 개념

: 데이터베이스 내에서 변별 가능한 객체 (C.J Date 1986 )

: 정보를 저장할 수 있는 어떤 것 (James Marth 1989)

- 업무에 필요하고 유용한 정보를 저장하고 관리하기 위한 **집합적인 것(Thing)**으로 설명할 수 있다.
- 대상들 간에 동질성을 지닌 인스턴스들
- 그들이 행하는 행위의 집합으로 정의할 수 있다.

- 엔터티는 그 집합에 속하는 개체들의 특성을 설명할 수 있는 속성을 갖는다. (Attribute)_ Ex) 엔터티-학생—학번, 이름, 이수학번 등....의 **속성들로 특정지어질 수 있다**.
- 엔터티(복수 Thing), 인스턴스(단수 Thing) 둘 모두가 공유할 수 있는 속성(Attribute)이 있고, 둘 중 일부에만 해당하는 개별속성도 있다.
- 엔터티는 인스턴스의 집합 개념이며, 반대개념 역시 성립한다..
- 눈에 보이는(Tangible)한 것만 엔터티로 생각해서는 안된다.

**** 참고 : '오브젝트 모델링'에는 클래스(Class)와 오브젝트(Object) 개념이 있으며, 클래스는 단지 오브젝트를 포함하는 빈 오브젝트이다.**

2. 엔터티의 특징

: **반드시** 해당 **업무에서 필요하고 관리하고자 하는 정보**이어야 한다

: **유일한 식별자에 의해 식별이 가능해야한다.**

: 영속적으로 존재하는 인스턴스의 집합(두개 이상)

: 엔터티는 업무 **프로세스에 의해 이용되어야 한다**. ***

: 반드시 **속성**이 있어야한다.

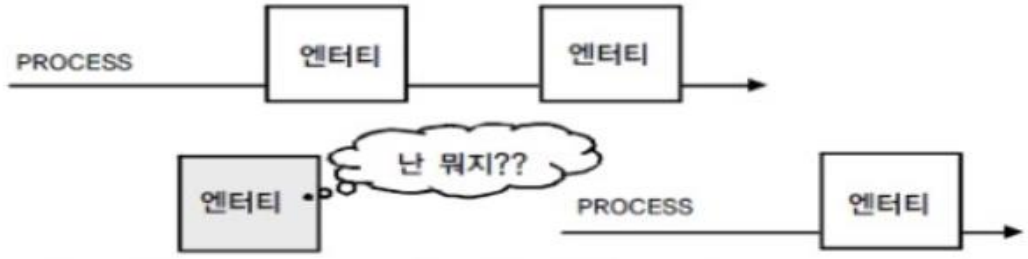
: 최소 한 개 이상의 엔터티 끼리의 관계가 있어야 한다.

- 업무에서 필요로 하는 정보일 것
- 식별이 가능해야 할 것
  : 식별자(일련번호)를 부여하여 유일하게 만들 수도 있지만, **업무적으로 의미를 가지는 인스턴스**가 **식별자에 의해 한 개씩만 존재**하는지 검증 필요
  : 유일한 식별자는 그 엔터티의 인스턴스만의 고유한 이름이다.

이름은 동명이인 때문에 유일한 식별자가 될 수 없고, 사원번호는 회사에 입사한 사람에게 고유하게 부여된 번호이므로 유일한 식별자가 될 수 있음

- 인스턴스의 집합
  : **영속적**으로 존재하는 인스턴스의 집합이 되어야 함
  ** 만약, 인스턴스가 한 개 밖에 없는 회사, 병원 엔터티라면 집합이 아니므로 엔터티 성립이 안된다.
- 업무 프로세스에 의해 이용
  : 업무에서 반드시 필요하다고 생각하여 엔터티로 선정하였더라도, **업무프로세스에 의해** 전혀 이용되지 않는다면 **분석정확도, 엔터티선정, 업무프로세스도출** 적절하지 않을 수 있음

이러한 경우는 데이터 모델링을 할 때 미처 발견하지 못하다가 프로세스 모델링을 하면서 데이터 모델과 검증을 하거나, 상관 모델링을 할 때 엔터티와 단위프로세스를 교차 점검하면서 문제점이 도출된다.




업무 프로세스에 의해 이용되지 않는 엔터티는 그 업무의 엔터티가 아님
[그림 Ⅰ-1-20] 엔터티 특성-프로세스 존재

- 속성을 포함
  : 엔터티에는 반드시 속성(Attribute)이 포함되어야 함
  **엔터티의 이름만 가지고 있거나** 또는, **주식별자만 존재하고 일반속성은 전혀 없는 경우**도 마찬가지로 적절한 엔터티라고 할 수 없다.

단, 예외적으로 **관계엔터티(Associative Entity)의 경우는 주식별자 속성만 가지고 있어도 엔터티로 인정한다.**

- **관계**의 존재
  : 기본적으로 엔터티가 도출되었다는 것은 해당 업무내에서 업무적인 연관성(**존재적 연관성, 행위적 연관성**)을 가지고 다른 엔터티와의 연관의 의미를 가지고 있음을 나타낸다.

데이터 모델링을 하면서**관계를 생략하여 포현해야 하는 경우도 있는데,
통계성, 코드성, 시스템 처리사 내부 필요에 의한 엔터티 도출과 같은 경우이다.

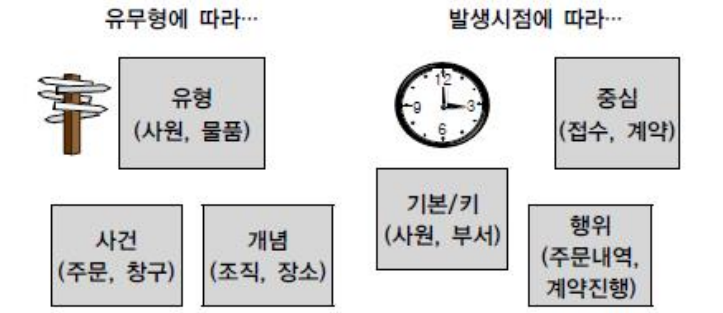
** 통계를 위한 엔터티는 경우는, 통계업무(Read Only)만을 위해 별도로 엔터티를 다시 정의
** 코드를 위한 엔터티의 경우는, 너무 많은 엔터티와 엔터티간의 관계 설정으로 모델의 읽기 효율성(Readability)이 저하 더불어, 물리적으로 테이블과 프로그램 구현 이후에도 **외부키에 의한 참조무결성을 체크하기 위한 규칙을 데이터베이스 기능에 맡기지 않는 경우가 대부분**이기 때문에 논리적으로나 물리적으로 관계를 설정할 이유가 없다.

** 시스템 처리시 내부 필요에 의한 엔터티(예를 들어, 트랜잭션 로그 테이블 등)의 경우 트랜잭션이 업무적으로 연관된 테이블과 관계 설정이 필요하지만 이 역시 업무적인 필요가 아니고 시스템 내부적인 필요에 의해 생성된 엔터티이므로 관계를 생략하게 된다.

3. 엔터티의 분류
   자신의 성격에 의해 실제 **유형**에 따라 Or 업무를 구성하는 모습에 따라 구분이 되는 **발생시점**에 의해 분류해 볼 수 있다.

- **유무형에 따른 분류**
  - **유형엔티티(Tangible Entity)**
  - **개념엔티티(Concepture Entity) :** 물리적인 형태는 존재하지 않고, 관리해야 할 개념적 정보로 구성 _ 조직, 보험상품
  - **사건엔티티(Event Entity) :** 업무를 수행함에 따라 발생되는 엔터티로서 비교적 발생량이 많으며 각종 통계자료에 이용될 수 있다. 주문, 청구, 미납 등이 이에 해당된다.
- **발생시점에 따른 분류**
  - **기본/키 엔터티 :** 독립적으로 생성이 가능, 자신은 타 엔터티의 부모의 역할을 하게 됨, 다른 엔터티로부터 주식별자를 상속받지 않고 자신의 고유한 주식별자를 가지게 된다. (사원, 부서, 고객, 상품, 자재 등)
  - **중심엔터티 : 기본엔터티로부터 발생**되고 그 업무에 있어서 중심적인 역할을 한다. **데이터의 양이 많이 발생**되고 다른 엔터티와의 관계를 통해 **많은 행위엔터티를 생성한다**. (계약, 사고, 예금원장, 청구, 주문, 매출 등)
  - **행위엔터티 : 두 개 이상의 부모엔터티로부터 발생**되고 자주 내용이 바뀌거나 데이터량이 증가된다.

**분석초기 단계에서는 잘 나타나지 않으며** 상세 설계단계나 프로세스와 상관모델링을 진행하면서 도출될 수 있다. (주문목록, 사원변경이력 등)




엔터티를 도출할 때 일정한 그룹에 의해 그룹화하면 도출작업에 효율적임

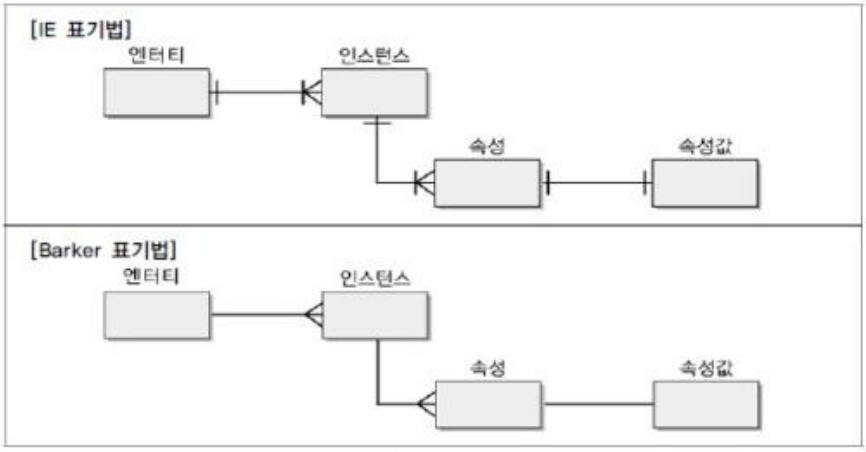
[그림 Ⅰ-1-23] 엔터티 분류

4. 엔터티의 명명
   - 첫 번째는 가능하면 현업업무에서 사용하는 용어를 사용한다.
   - **두 번째는 가능하면 약어를 사용하지 않는다.**
   - 세 번째는 단수명사를 사용한다.
   - 네 번째는 모든 엔터티에서 **유일하게 이름이 부여되어야 한다.**
   - 다섯 번째는 엔터티 생성의미대로 이름을 부여한다.

**데이터 모델링의 이해**
**속성**

1. 속정(Attribute)의 개념
   - 업무에서 필요로 한다.
   - 의미상 더 이상 분리되지 않음
   - **엔터티를 설명하고 인스턴스의 구성요소가 된다.**

2. 엔터티, 인스턴스, 속성, 속성값에 대한 내용 및 표기법
   - **관계**
     - 1 개 엔터티, 두 개 이상의 인스턴스
     - 1 개 엔터티, 두개 이상의 속성
     - 1 개의 속성 1 개의 속성값




[그림 Ⅰ-1-25] 엔터티-속성의 관계

예시 : 예를 들어 사원이라는 엔터티에는 홍길동이라는 사람(엔터티)이 있을 수 있다.

홍길동이라는 사람의 이름은 홍길동이고 주소는 서울시 강서구이며 생년월일 1967 년 12 월 31 일이다. 여기에 이름, 주소, 생년월일과 같은 각각의 값을 대표하는 이름들을 속성이라 하고 홍길동, 서울시 강서구, 1967 년 12 월 31 일과 같이 각각의 이름에 대한 구체적인 값을 속성 값(VALUE)이라고 한다.

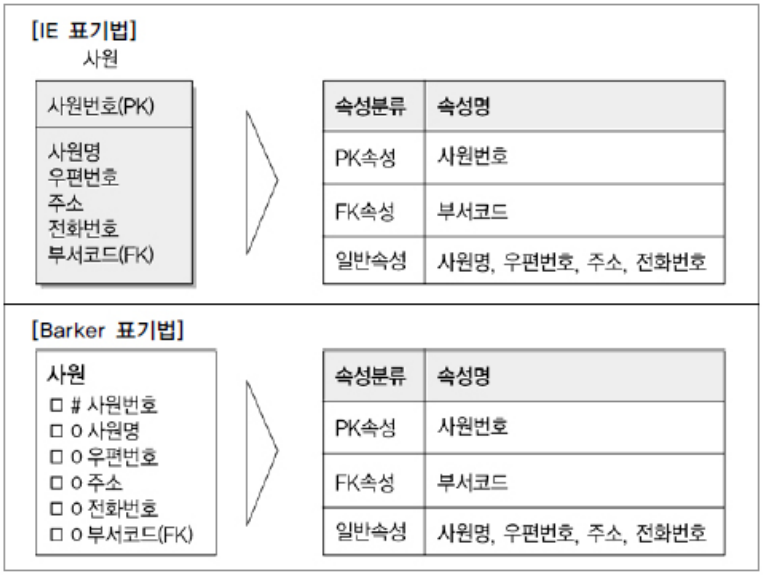
3. 속성의 특징
   - 주식별자에 함수적 종속성을 가져야 한다.
   - 일반적으로 한 개의 속성에는 한 개의 값만을 가지며, **1 개의 속성에 여러 개의 값이 있는 다중값일 경우 별도의 엔터티를 이용하여 분리한다.**

4. 속성의 분류
   - 속성의 **특성에 따른 분류**
     - **기본속성**은,
       - 가장 일반적이고 많은 비중을 차지
       - 제외속성 : **코드성, 엔터티 식별을 위한 일련번호, 다른 속성을 계산 또는 영향을 받아 생성된 속성은 제외**

주의해야 할 것은 업무로부터 분석한 속성이라도 이미 업무상 코드로 정의한 속성이 많다는 것이다. 이러한 경우도 속성의 값이 원래 속성을 나타내지 못하므로 기본속성이 되지 않는다.

- **설계속성**은,
  - **데이터 모델링 목적**
  - 업무를 **규칙화**하기 위해 속성을 새로 만들거나 변형하여 정의하는 속성 (위 주의사항과 구분하여 이해할 것)
  - 대개 **코드성** 속성은 **업무상** 필요에 의한 설계속성, 일련번호와 같은 단일속성은 **식별자 부여**를 목적으로, **모델상**에서 새로 정의하는 설계속성이다.
- **파생속성**은,
  - 다른 속성에 영향을 받아 발생, 보통 계산된 값들이 이에 해당
  - ***** 프로세스 설계 시 데이터의 정합성을 유지하기 위해 유의해야 할 점이 많으며 가급적 파생속성을 적게 정의하는 것이 좋음**

- 엔터티 구성방식에 따른 분류,
  - 엔터티 식별 속성을 PK(Primary Key)속성이라고 하고, 다른 엔터티와의 관계에서 포함된 속성을 **FK(Foreign Key)속성** 엔터티에 포함되어 있고 PK, FK 에 포함되지 않은 속성을 **일반속성**이라 한다.




[그림 Ⅰ-1-28] 속성의 분류

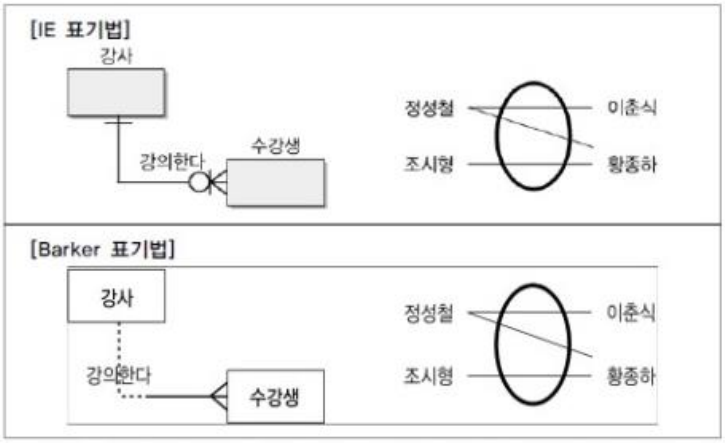
5. 도메인

각 속성은 가질 수 있는 값의 범위가 있는데 이를 그 속성의 도메인(Domain)이라 한다.
예를 들면 학생이라는 엔터티가 있을 때 학점이라는 속성의 도메인은 0.0 에서 4.0 사이의 실수 값이며 주소라는 속성은 길이가 20 자리 이내인 문자열로 정의할 수 있다. 여기서 물론 각 속성은 도메인 이외의 값을 갖지 못한다. 따라서 도메인을 좀더 이해하기 쉽게 정리하면, 엔터티 내에서 속성에 대한 데이터타입과 크기 그리고 제약사항을 지정하는 것이라 할 수 있다.

**데이터 모델링의 이해**
**관계**

1. 관계의 정의
   - 관계의 정의
     - **인스턴스** 사이의 논리적인 연관성이 부여된 상태
   - 관계의 패어링
     - 관계는 엔터티 안에서의 집합이다.
       - 엔터티 안의 인스턴스가 개별적으로 관계를 가짐 == 패어링
       - 두 개의 엔터티 사이에 2 개 이상의 관계가 형성도리 수 있음
       - 각각의 엔터티의 인스턴스들은 자신이 관련된 인스턴스들과 관계의 Occurance 로 참여하는 형태를 **관계 패어링(Relationship Paring)이라 한다**.

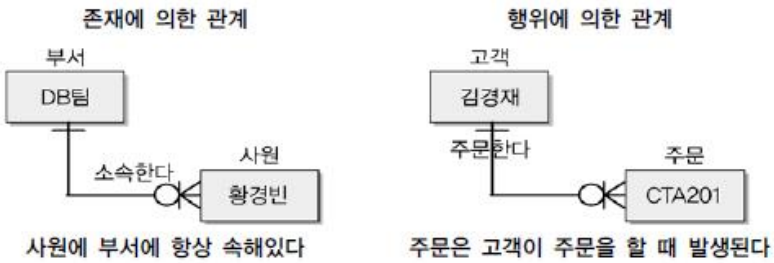
엔터티는 인스턴스의 집합을 논리적으로 표현하였다면 관계는 관계 패어링의 집합을 논리적으로 표현한 것이다.




인스턴스 각각은 자신의 연관성을 가지고 있을 수 있음. 이것을 집합하여 '강의'라는 관계 도출

[그림 Ⅰ-1-31] 관계의 패어링

2. 관계의 분류




[그림 Ⅰ-1-32] 관계의 분류

   - 관계의 분류는 두 가지로 생각해 볼 수 있는데, [그림 1 – 1 - 32]의 첫 번째의 경우와 같이, 단순히 **존재의 형태에 의해 관계가 형성되어 있는 것**
   - 그리고, 두 번째 경우와 같이 **행위에 의한 관계** 두 가지를 고려해 볼 수 있다.
3. 관계의 표기법
   - 3 가지 개념과 표기법
     - 관계명(Membership) : 관계의 이름
     - 관계차수(Cardinality) : 1:1, 1:M, M:M
     - 관계선택사양(Optionality) : 필수관계, 선택관계

- 관계명(Membership)

  : 각각의 관계는 두 개의 관계명, 즉 2 개의 관점으로 표현될 수 있다.

  : 참여자의 관점에 따라 **관계이름이 능동적(Active)이거나 수동적(Passive)으로 명명된다**. (** 애매한 동사 피하기, 현재형으로 표기)

- 관계차수(Degree, Caardinality)

  : 1:1, 1:M, M:M (1-One, M-Many)

  : 1:M_ 반대의 방향은 단지 하나만의 관계를 가지고 있다.

  : M:M_

  - 반대의 방향도 동일하게 관계에 참여하는 각각의 엔터티는 관계를 맺는 다른 엔터티의 엔터티에 대해 하나 또는 그 이상의 수와 관계를 가지고 있다.
  - 이후에 두 개의 주식별자를 상속받은 관계엔터티를 이용하여 **3 개의 엔터티로 구분하여 표현한다.**

- 관계선택사항(Optionality)

  : 필수참여관계(Mandatory), 선택참여관계(Optional) 두 개로 나뉨

  : 참여하는 엔터티가 항상 참여하는지 아니면 참여할 수도 있는지를 나타내는 방법이 필수(Mandatory Membership)와 선택참여(Optional Membership)이다.

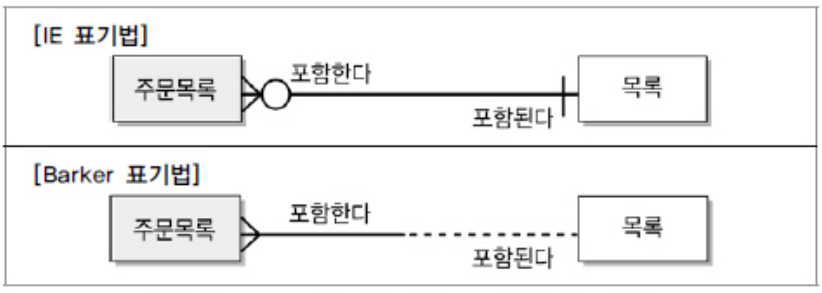
  **Example )**

  - 주문서 – 주문목록 : 필수참여
  - 주문목록 – 주문 : 선택참여

** 선택참여된 항목은 물리속성에서 Foreign Key 로 연결될 경우 Null 을 허용할 수 있는 항목이 된다.

** 만약 선택참여로 지정해야 할 관계를 필수참여로 잘못 지정하면 애플리케이션에서 데이터가 발생할 때 반드시 한 개의 트랜잭션으로 제어해야 하는 제약사항이 발생한다.

선택참여관계는 ERD에서 관계를 나타내는 선에서 선택참여하는 엔터티 쪽을 원으로 표시한다. 필수참여는 아무런 표시를 하지 않는다.
만약 관계가 표시된 양쪽 엔터티에 모두 선택참여가 표시된다면, 즉 0:0(Zero to Zero)의 관계가 된다면 그 관계는 잘못될 확률이 많으므로 관계설정이 잘못되었는지를 검토해 보아야 한다.

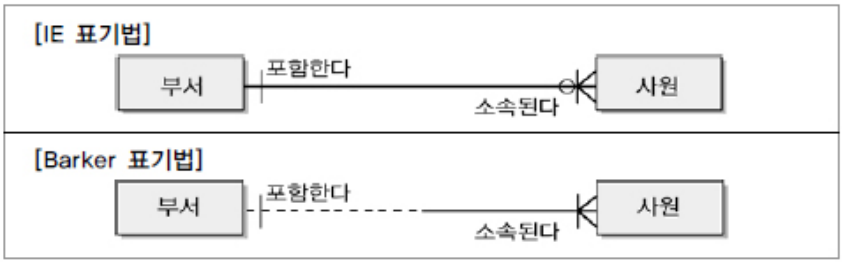



하나의 주문목록에는 한 개의 목록을 항상 포함하고
한 목록은 여러 개의 주문 목록에 의해 포함될 수 있다.

[그림 Ⅰ-1-38] 관계선택참여

관계차수 ERD 중 1:M 관계표기방법과 구분하여 이해할 것

2) **1:M(ONE TO MANY) 관계를 표시하는 방법**




**한 명의 사원은 한 부서에 소속되고 한 부서에는 여러 사원을 포함한다**

**[그림 Ⅰ-1-35] 관계차수(1:M)**

4. 관계의 정의 및 읽는 방법

**가. 관계 체크사항**

두 개의 엔터티 사이에서 관계를 정의할 때 다음 사항을 체크해 보도록 한다.

**두 개의 엔터티 사이에 관심있는 연관규칙이 존재하는가?**
**두 개의 엔터티 사이에 정보의 조합이 발생되는가?**
**업무기술서, 장표에 관계연결에 대한 규칙이 서술되어 있는가?**
**업무기술서, 장표에 관계연결을 가능하게 하는 동사(Verb)가 있는가?**

**나. 관계 읽기**

데이터 모델을 읽는 방법은 먼저 관계에 참여하는 기준 엔터티를 하나 또는 각(Each)으로 읽고 대상 엔터티의 개수(하나, 하나 이상)를 읽고 관계선택사양과 관계명을 읽도록 한다.

기준(Source) 엔터티를 한 개(One) 또는 각(Each)으로 읽는다.
대상(Target) 엔터티의 관계참여도 즉 개수(하나, 하나 이상)를 읽는다.
관계선택사양과 관계명을 읽는다.

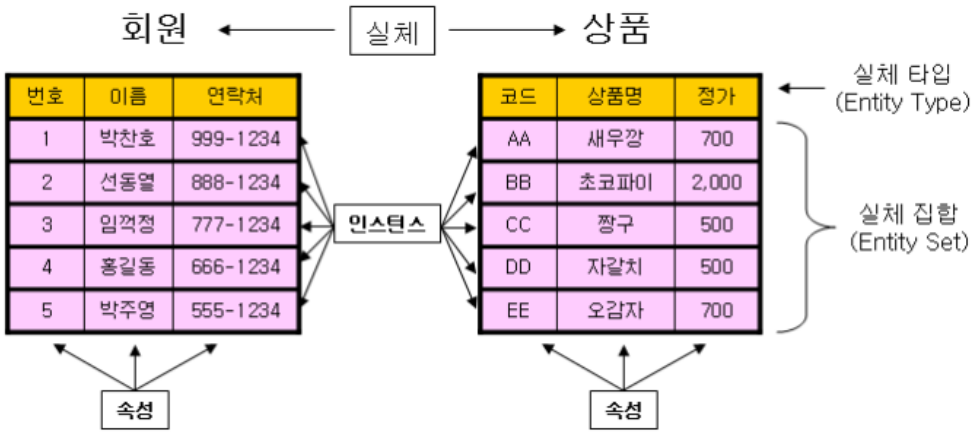
(수정) - 교재에는 이렇게 되어 있습니다.
- 각각의 사원은 한 부서에 항상 속한다.
- 각 부서에는 여러사원이 때때로 소속된다.


**[그림 Ⅰ-1-39] 관계의 읽는 방법**

**데이터 모델링의 이해**

**식별자**

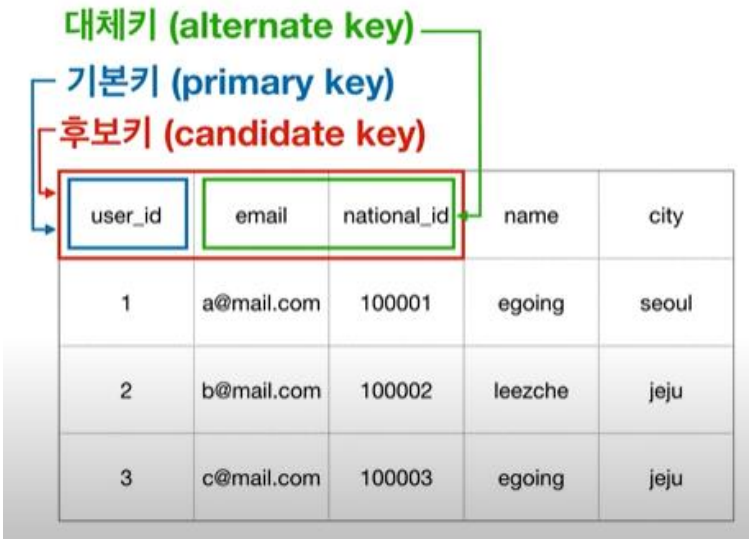
**Preview) 엔터티, 속성, 인스턴스의 구분**




1. 식별자 개념

   - 엔터티를 **대표**할 수 있다.
   - 엔터티는 반드시 **하나의 유일**한 식별자가 존재한다.
   - 엔터티를 **구분**짓는 논리적인 이름

**식별자(Identifier) : 대표속성이라고 볼 수 있음 – 이후 PK 가 될 것임**



중복키 (composite key)

| emp_no<br>직원번호 | dept_no<br>부서번호 | from_date<br>부서배정일 |
|---|---|---|
| 1 | 1 | 2010 |
| 2 | 1 | 2011 |
| 1 | 2 | 2013 |

2. 식별자 특징

- 주식별자에 의해 엔터티내에 모든 인스턴스들이 **유일하게 구분**되어야 한다.
- 주식별자를 구성하는 **속성의 수**는 **유일성을 만족하는 최소의 수**가 되어야 한다.
- 지정된 주식별자의 값은 자주 변하지 않는 것이어야 한다.
- 주식별자가 지정이 되면 반드시 값이 들어와야 한다.

[표 Ⅰ-1-7] 주식별자의 특징

| 특징 | 내용 | 비고 |
|---|---|---|
| 유일성 | 주식별자에 의해 엔터티내에 모든 인스턴스들을 유일하게 구분함 | 예) 사원번호가 주식별자가 모든 직원들에 대해 개인별로 고유하게 부여됨 |
| 최소성 | 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 함 | 예) 사원번호만으로도 고유한 구조인데 사원분류코드+사원번호로 식별자가 구성될 경우 부절한 주식별자 구조임 |
| 불변성 | 주식별자가 한 번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함 | 예) 사원번호의 값이 변한다는 의미는 이전기록이 말소되고 새로운 기록이 발생되는 개념임 |
| 존재성 | 주식별자가 지정되면 반드시 데이터 값이 존재 (Null 안됨) | 예) 사원번호 없는 회사직원은 있을 수 없음 |

3. 실별자 분류 및 표기법

- 식별자 분류

[표 Ⅰ-1-8] 식별자의 분류체계

| 분류 | 식별자 | 설명 |
|---|---|---|
| 대표성 여부 | 주식별자 | 엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이며, 타 엔터티와 참조관계를 연결할 수 있는 식별자 |
| | 보조식별자 | 엔터티 내에서 각 어커런스를 구분할 수 있는 구분자이나 대표성을 가지지 못해 참조관계 연결을 못함 |
| 스스로 생성여부 | 내부식별자 | 엔터티 내부에서 스스로 만들어지는 식별자 |
| | 외부식별자 | 타 엔터티와의 관계를 통해 타 엔터티로부터 받아오는 식별자 |
| 속성의 수 | 단일식별자 | 하나의 속성으로 구성된 식별자 |
| | 복합식별자 | 둘 이상의 속성으로 구성된 식별자 |
| 대체 여부 | 본질식별자 | 업무에 의해 만들어지는 식별자 |
| | 인조식별자 | 업무적으로 만들어지지는 않지만 원조식별자가 복잡한 구성을 가지고 있기 때문에 인위적으로 만든 식별자 |

글

| 아이디 | 제목 | 내용 | 저자 아이디 |
|---|---|---|---|
| 1 | 제목 1 | 내용 1 | 1 |
| 2 | 제목 2 | 내용 2 | 1 |
| 3 | 제목 3 | 내용 3 | 1 |

저자

| 아이디 | 이름 | 소개 |
|---|---|---|
| 1 | 이름 1 | 소개 1 |
| 2 | 이름 2 | 소개 2 |
| 3 | 이름 3 | 소개 3 |

댓글

| 아이디 | 제목 | 내용 | 작성일 | 저자 아이디 |
|---|---|---|---|---|
| 1 | 제목 1 | 내용 1 | 작성일 1 | 1 |
| 2 | 제목 2 | 내용 2 | 작성일 2 | 1 |
| 3 | 제목 3 | 내용 3 | 작성일 3 | 2 |

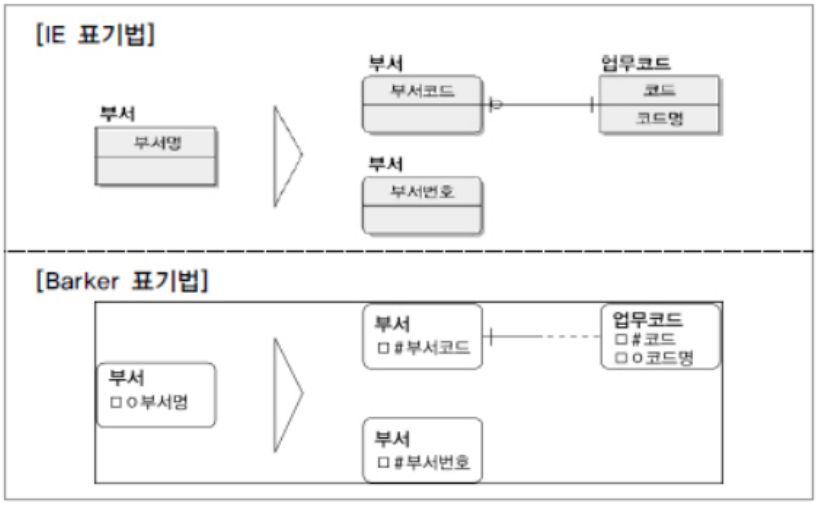
SELECT 댓글.내용, 댓글.작성일, 저자.이름, 저자.소개 FROM 댓글 LEFT JOIN 저자 ON 댓글.저자 아이디 = 저자.아이디

| 댓글 내용 | 댓글 작성일 | 저자 | 저자 소개 |
|---|---|---|---|
| 댓글 1 내용 | 댓글 1 작성일 | 저자 1 이름 | 저자 1 자기소개 |
| 댓글 2 내용 | 댓글 2 작성일 | 저자 1 이름 | 저자 1 자기소개 |
| 댓글 3 내용 | 댓글 3 작성일 | 저자 1 이름 | 저자 1 자기소개 |

외래 키(FK)가 저자의 주식별자와 연결되어(PK) JOIN

4. 주식별자 도출기준

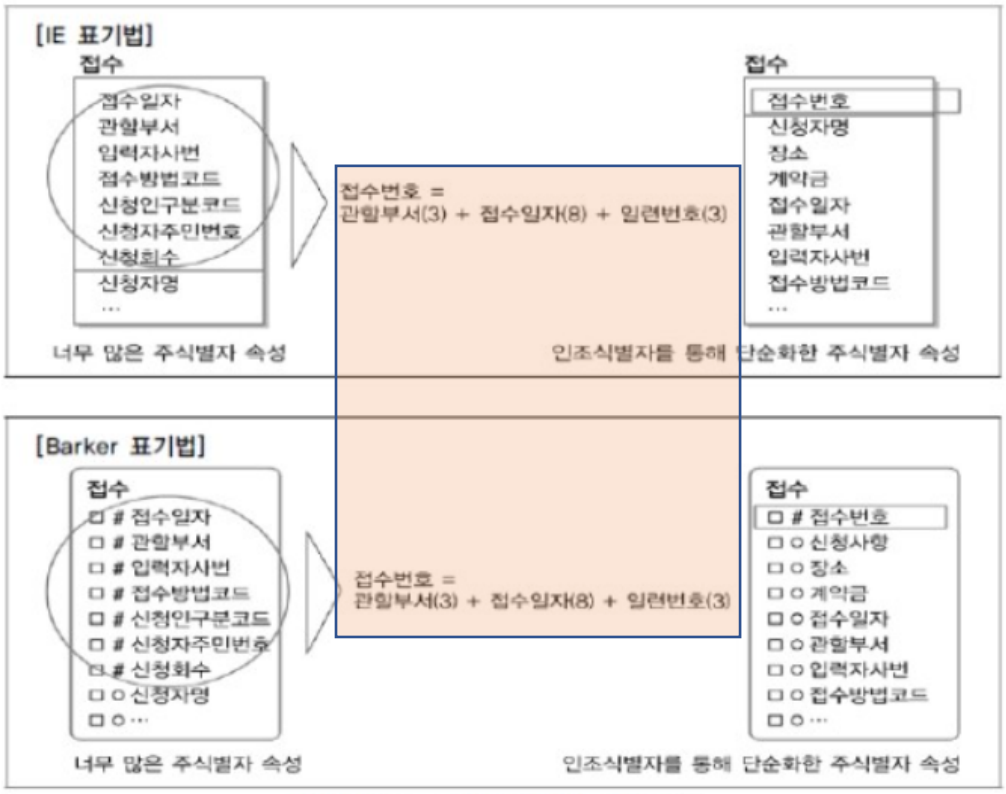
- 해당 업무에서 자주 이용되는 속성을 지정
- **이름으로 기술되는 것들은 피한다(예_ 명칭, 내역)**
- **군분자가 존재하지 않을 경우 새로운 식별자를 생성한다.**




[그림 Ⅰ-1-44] 주식별자-명칭/내역

**복합으로 주식별자로 구성할 경우 너무 많은 속성이 포함되지 않도록 한다.**

주식별자의 개수가 많을 경우 새로운 인조식별자(Artificial Identifier)를 생성한다.




[그림 Ⅰ-1-45] 주식별자-복합속성

5. **식별자관계**와 **비식별자관계**에 따른 식별자

- 식별자관계와 비식별자 관계의 결정
  - **관계와 속성을 정의**하고 **주식별자를 정의**하면 **논리적인 관계**에 의해 자연스럽게 **외부식별자**가 도출되지만 중요하게 **고려**해야 할 사항이 있다. 엔터티에 주식별자가 지정되고 엔터티간 관계를 연결하면 **부모쪽의 주식별자를 자식엔터티의 속성으로 내려 보낸다. 이 때 자식엔터티에서 부모엔터티로부터 받은 외부식별자를 자신의 주식별자로 이용할 것인지 또는 부모와 연결이 되는 속성으로서만 이용할 것인지를 결정해야 한다.**
- 식별자관계
  - 자식엔터티의 주식별자로 부모의 주식별자가 상속되는 경우를 말한다.
  - Null 값이 오면 않된다. (**부모쪽 주식별자로부터**)
  - 1:1 관계 : 부모로부터 받은 속성을 자식엔터티가 모두 사용하고 그것만으로 주식별자로 사용할 경우
  - 1:M 관계 : 부모로부터 받은 속성, 다른 부모엔터티에서 받은 속성, 스스로 가진 속성으로 주식별자가 구성된 경우
- 비식별자관계(Non-Identifying-Relationship)
  - 부모엔터티로부터 속성을 받았지만 자식엔터티의 주식별자로 사용하지 않고 일반적인 속성으로 사용하는 경우.
    - Example)

- 부모  없이 자식엔터티가 독자적으로 생성될 수 있는 경우
- 각각의 엔터티가 별도의 관계를 가질 때
- 엔터티별 Life Cycle 을 다르게 관리할 경우
- 자식엔터티에서 별도의 주식별자를 생성하는 것이 더 유리할 경우

▪ 식별자 관계로만 설정할 경우의 문제점
: 주식별자 속성이 지속적으로 증가 및 이로인한 복잡성과 오류가능성을 유발

▪ 비식별자 관계로만 설정할 경우의 문제점
: **속성이 자식 엔터티로 상속되지 않는다**
: 속성이 상속되지 않으니, 부모엔터티까지 조인되는 현상 즉, 불필요한 조인이 발생되고 SQL 구문도 길어져서 성능이 저하됨.

따라서, 일정한 규칙을 가지고 데이터 모델링을 하는 기술이 필요

- **비식별자관계 설정 고려사항 :**
  - **과계가 약할 때**
  - **자식테이블 독립 PK 가 필요하다고 판단될 때**
  - **SQL 복잡도 증가로 개발생산성이 저하된다면, PK 속성 단순화를 위해, 비식별자 관계 설정을 고려한다.**

**[표 Ⅰ-1-10] 식별자와 비식별자관계 비교**

| 항목 | 식별자관계 | 비식별자관계 |
|---|---|---|
| 목적 | 강한 연결관계 표현 | 약한 연결관계 표현 |
| 자식 주식별자 영향 | 자식 주식별자의 구성에 포함됨 | 자식 일반 속성에 포함됨 |
| 표기법 | 실선 표현 | 점선 표현 |
| 연결 고려사항 | – 반드시 부모엔터티 종속<br>– 자식 주식별자구성에 부모 주식별자포함 필요<br>– 상속받은 주식별자속성을 타 엔터티에 이전 필요 | – 약한 종속관계<br>– 자식 주식별자구성을 독립적으로 구성<br>– 자식 주식별자구성에 부모 주식별자 부분 필요<br>– 상속받은 주식별자속성을 타 엔터티에 차단 필요<br>– 부모쪽의 관계참여가 선택관계 |