

# 데이터 모델링

데이터베이스 구성과 처리에 있어서 가장 핵심적인 요소로 건물의 설계도와 같이 전체 데이터베이스를 구성, 결정하는데 사용합니다. 데이터모델링은 DBMS 사용에 있어가장 중요한 부분이지만 쉽게 간과하기도 합니다.

데이터 모델링은 기업 내 산재한 데이터들의 체계적인 수집과 분류로, 비즈니스 정보의 명료하고 정확한 분석 및 설계를 통해 전사적 아키텍처를 고려한 데이터 설계와 최적의 데이터 구조 설계하는 과정입니다.

서비스의 전체적인 개발에 있어서 데이터베이스 관련 부분으로 놓고 보자면 (업무파악 후) 개념설계, 논리설계, 물리설계 (그리고 DBMS구축)으로 구분해볼수 있습니다.



데이터 모델링 vs 데이터 아키텍처 (개발자, DBA VS DBA)

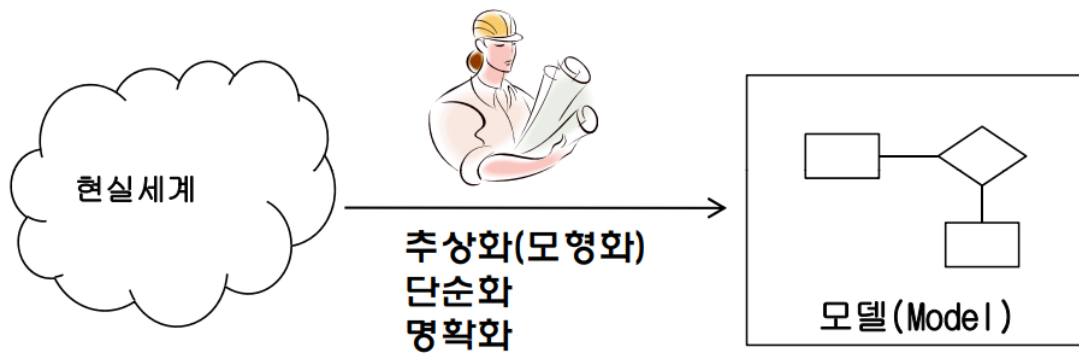
## ▼ 활용 도구

- [draw.io](https://app.diagrams.net) (<https://app.diagrams.net>)
- sqldeveloper(보기 > Data Modeler)
- ermaster (이클립스 플러그인, <http://ermaster.sourceforge.net>) \* 21c 호환성 참고
- erwin(<https://www.erwin.com>)
- erdCLOUD(<https://www.erdcloud.com>)

## 1. 데이터 모델링 개요

현실 세계의 데이터를 컴퓨터 세계의 데이터베이스로 옮기는 과정으로 데이터베이스 핵심 설계 과정이라고 할 수 있습니다.

데이터 모델링은 프로젝트 초기에 하는 중요한 작업 중 하나로 복잡한 파악한 업무에서 요구사항을 단순, 명확화 하고 간결하게 표현하며 현실 문제등을 해결하는데 가장 중요한 도구로 사용할 수 있습니다.



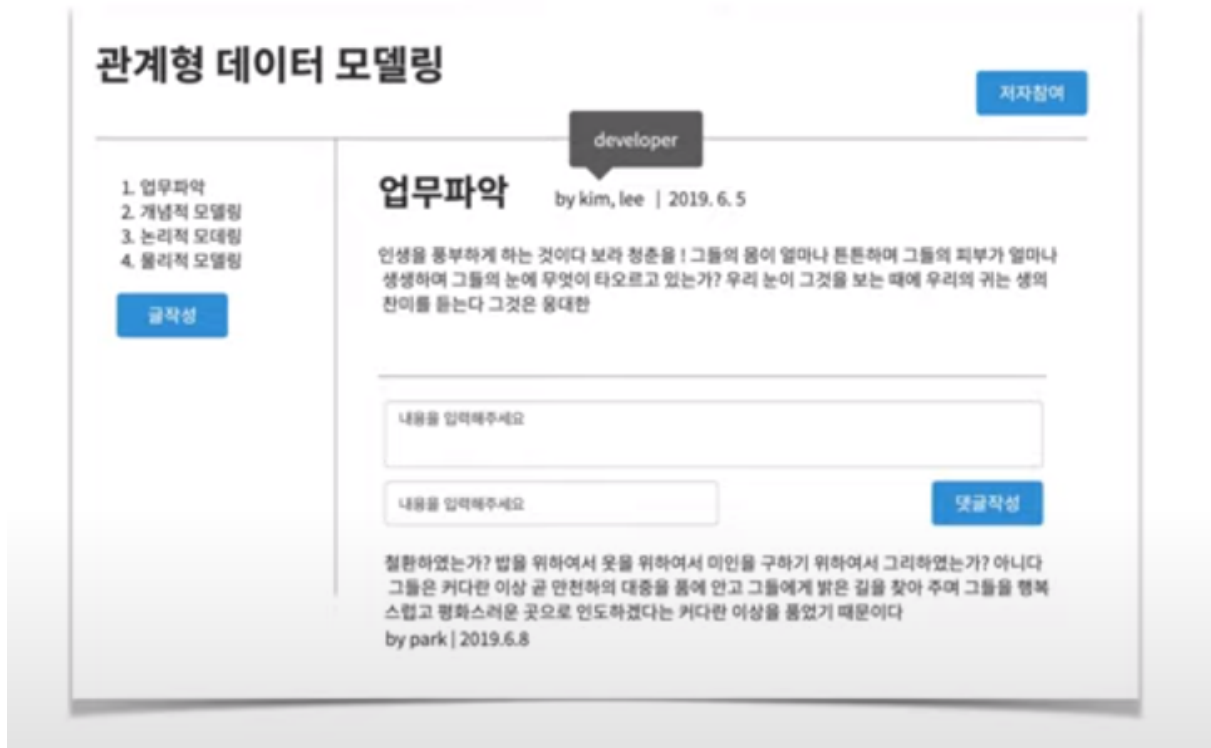
<출처 - 한국데이터베이스진흥원>



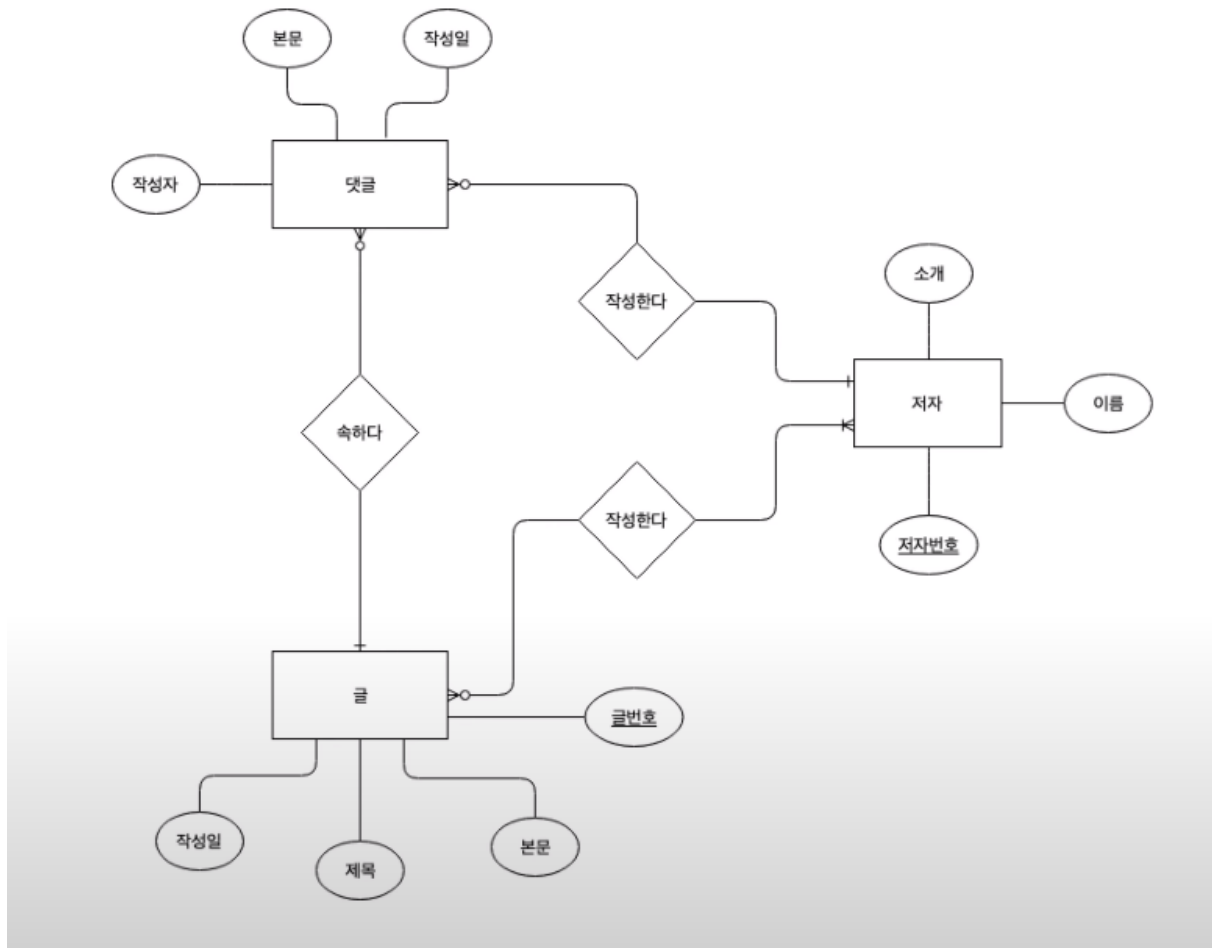
데이터 모델링은 데이터베이스를 구축하기 위한 분석, 설계의 과정으로 해당 업무에 어떤 데이터가 존재하고 어떤 업무가 필요하는지를 파악한 뒤 이를 바탕으로 시스템을 구축하기 위한 분석 방법중 하나입니다.

## 2. 데이터 모델링 종류

- 개념적 데이터 모델 : 현실 세계를 개념적으로 모델링하여 데이터베이스 개념 구조로 표현(개체-관계 모델, Peter Chen / IE, Crow's Foot / Barker, ... 표기법이 있음 )
- 엔터티의 범위, 필수 속성 정의, 식별자(PK) 지정, 각 엔터티들의 관계 파악이 핵심



< 업무파악 - 요구사항 (기획서) by 생활코딩 >



< ERD : 정보, 그룹, 관계를 파악하게 해줌 >

- 논리적 데이터 모델 : 개념적 구조를 모델링 하여 데이터베이스의 논리적 구조로 표현 (관계모델)

속성(에트리뷰트),  
열(column) 이라고도 함  
(차수=4)

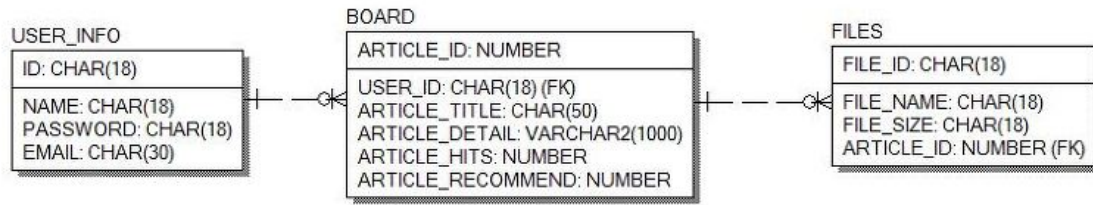
| 도서번호 | 도서이름    | 출판사   | 가격    |
|------|---------|-------|-------|
| 1    | 축구의 역사  | 굿스포츠  | 7000  |
| 2    | 축구아는 여자 | 나무수   | 13000 |
| 3    | 축구의 이해  | 대한미디어 | 22000 |
| 4    | 골프 바이블  | 대한미디어 | 35000 |
| 5    | 피겨 교본   | 굿스포츠  | 8000  |

← 스키마(내포) Schema

← 인스턴스(외연) Instance

투플(tuple),  
행(row) 이라고도 함  
(카디널리티=5)

- 물리적 데이터 모델 : 실제 데이터베이스에 이식할 수 있는 성능, 저장공간등을 고려하여 설계

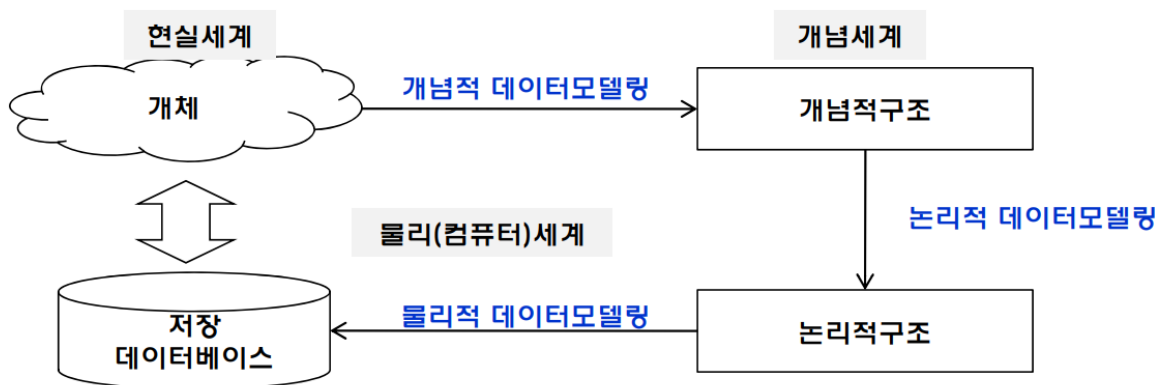


< 엔터티 다이어그램 >



대부분의 데이터 모델링 도구는 개념모델링과 논리모델링을 구분하지 않고 하나로 (논리모델링) 관리합니다.

### 3. 데이터 모델링 3단계



| 구분    | ERD | 설명  |
|-------|-----|---|
| 개념 모델 |     | <p>핵심 엔터티를 도출하고 그들간의 관계를 정의하여 전체 데이터 모델의 골격을 생성한다.</p> <ul style="list-style-type: none"> <li>- 향후 논리데이터 모델링 단계에서 보다 상세하게 정의될 데이터 모델의 기본 틀을 정의한다.</li> <li>- 업무의 구성 요소 중 업무기능과 대응되는 데이터에 대한 이해를 높인다.</li> <li>- 업무 영역에서 필요로 하는 주요 엔터티를 찾아내고, 주요 엔터티와 다른 엔터티들 간의 관계를 파악한다.</li> <li>- 원활한 의사 소통을 통하여 데이터를 명확하게 이해하는 과정을 지원한다.</li> </ul> |
| 논리 모델 |     | <p>개념 데이터 모델에서 정의한 핵심 엔터티와 관계를 바탕으로 모든 엔터티를 도출, 상세화 하고 속성을 정의하여 식별자를 확정 하고, 모든 관계를 도출 한다.</p> <ul style="list-style-type: none"> <li>- 정규화 작업을 통해 중복 속성을 제거하여 데이터의 중복 및 불일치를 최소화 한다.</li> <li>- 데이터의 통합 및 분리를 구체화 한다.</li> <li>- 시스템 설계자와 업무담당자 간의 명확한 의사소통의 도구 역할을 한다.</li> </ul>  |
| 물리 모델 |     | <p>논리모델에서 정의한 비즈니스 중심의 모델을 DBMS의 특성을 고려하여 데이터 베이스의 저장구조(물리 데이터 모델)로 변환하여 데이터 모델을 최종적으로 완성한다.</p> <ul style="list-style-type: none"> <li>- 트랜잭션 패턴, 대용량 특성을 고려하여 으로 테이블 통합/분리를 결정 한다.</li> <li>- PK구성 및 인덱스 특성을 감안한 물리 설계를 한다.</li> <li>- 성능 향상을 위하여 역정규화를 한다.</li> <li>- 테이블 용량, 보존 기간에 따른 파티션 설계를 한다.</li> </ul>                       |

<개념-논리 모델의 추상화 >

#### 4. 프로젝트 라이프사이클 단계에서의 모델링

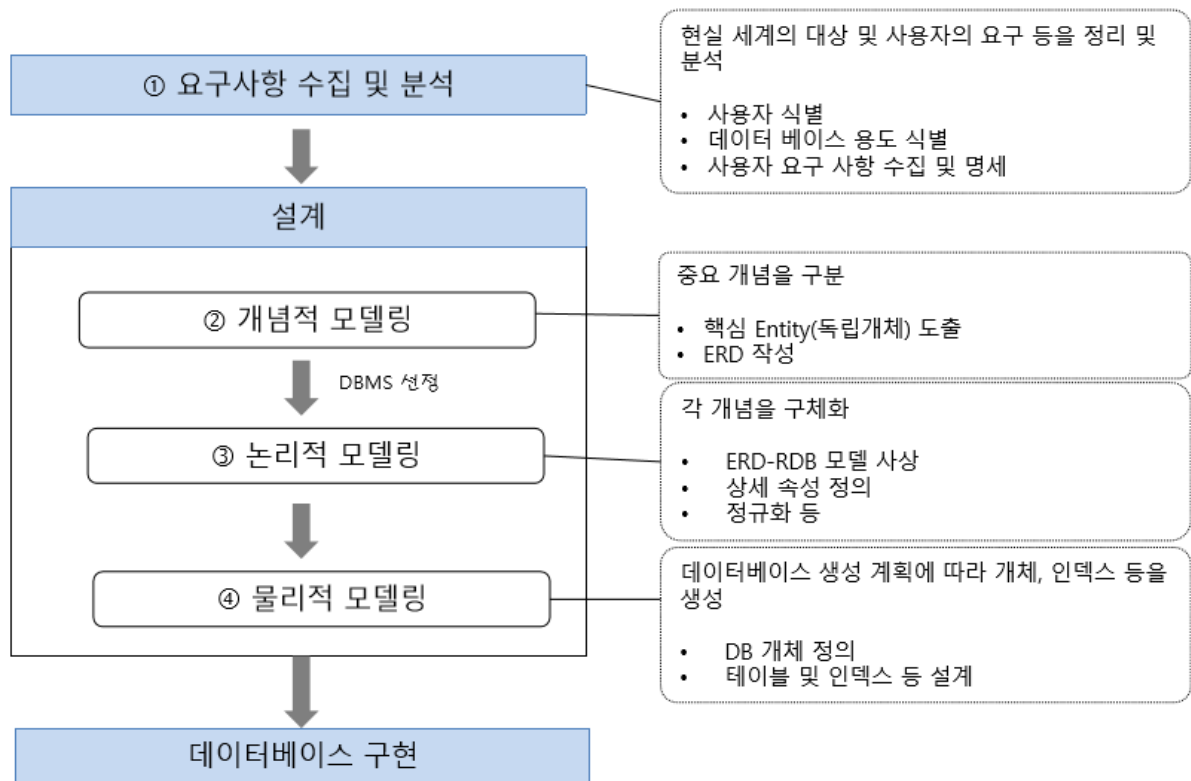
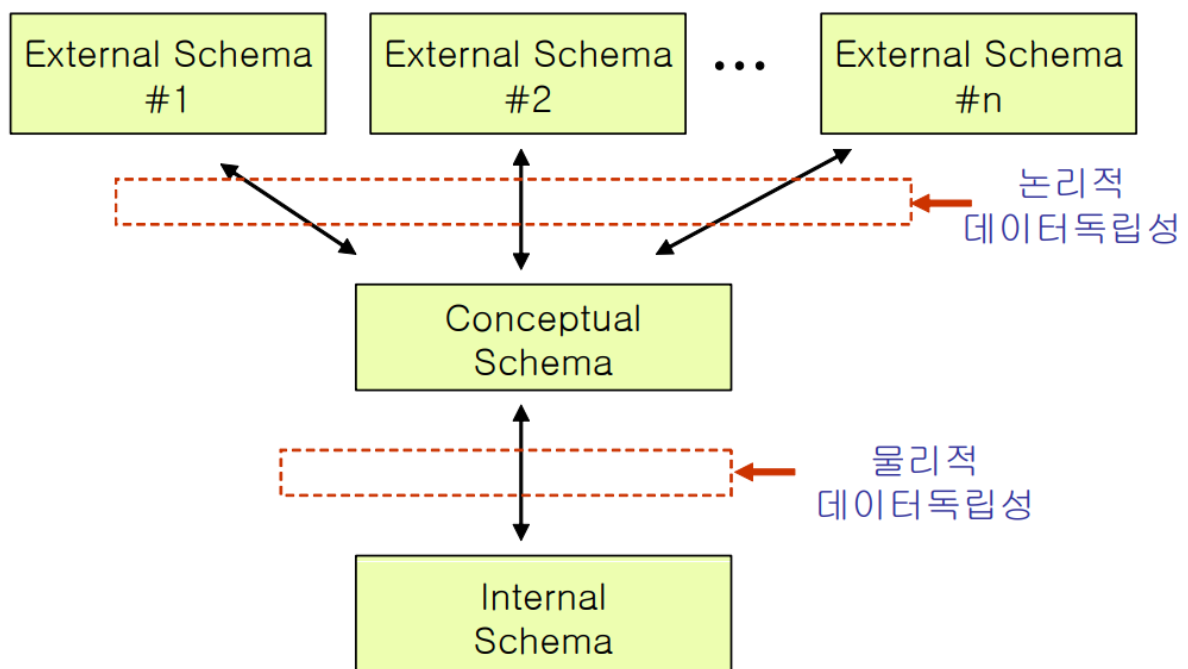


그림 6-4 데이터 모델링 과정

< 출처 - JM-log Tistory >

## 5. 데이터베이스 구조

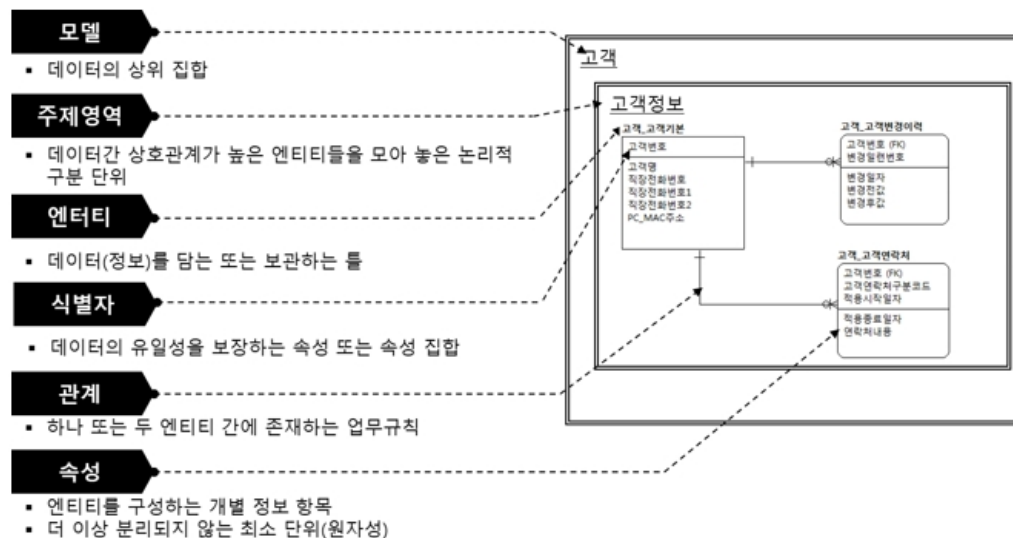


## 6. 데이터 모델링 세가지

- 업무가 관여하는 어떤 것(Thing)
- 업무가 관여하는 어떤 것 간의 관계(Relationships)
- 어떤 것이 가지는 성격(Attributes)

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

< 개념적 모델링 - 세가지 관점 >



< 데이터 모델의 요소 - 한국데이터베이스 진흥원 >

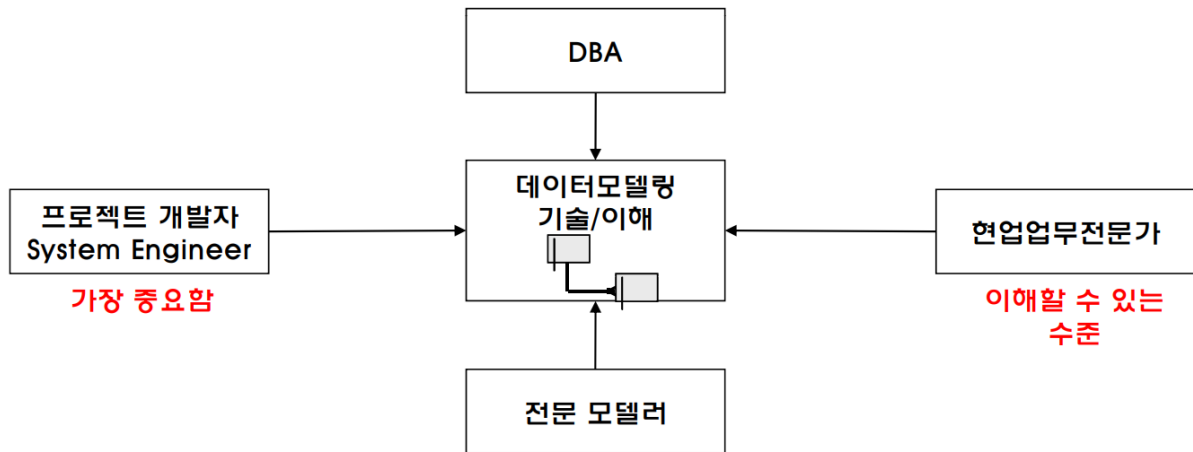
## 7. 데이터 모델링 이해관계자



- 정보시스템을 구축하는 모든 사람(프로젝트 참여 모든 IT 작업자)
- 전공자나 IT관련 작업자가 아닌 사람

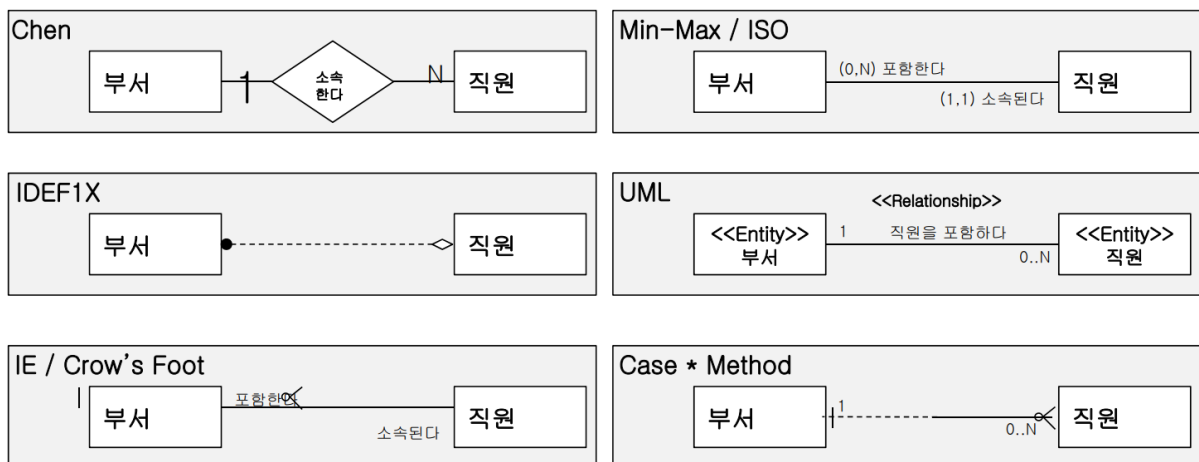


실제 DBA가 현장에 많이 있지도 않고, 그렇다고 개발자가 DBA 업무를 모두 수행할 수도 없습니다. 상황에 따라 DBA가 모델링을 겸할수 있으나 그것도 대부분 초급개발자의 업무영역이 아닙니다.



<출처 - 한국데이터베이스 진흥원>

## 8. 데이터모델링 표기법



## 9. ERD 작업순서

ERD(Entity Relationship Diagram)는 각 업무분석에서 도출된 엔터티와 엔터티간의 관계를 이해하기 쉽게 도식화된 다이어그램으로 표시하는 방법으로서 실제 프로젝트에서는 도식화된 그림 정도로만 생각하지 않고 해당 업무에서 데이터의 흐름과 프로세스와의 연관성을 이야기하는 데 가장 중요한 표기법이며 산출물이 됩니다.

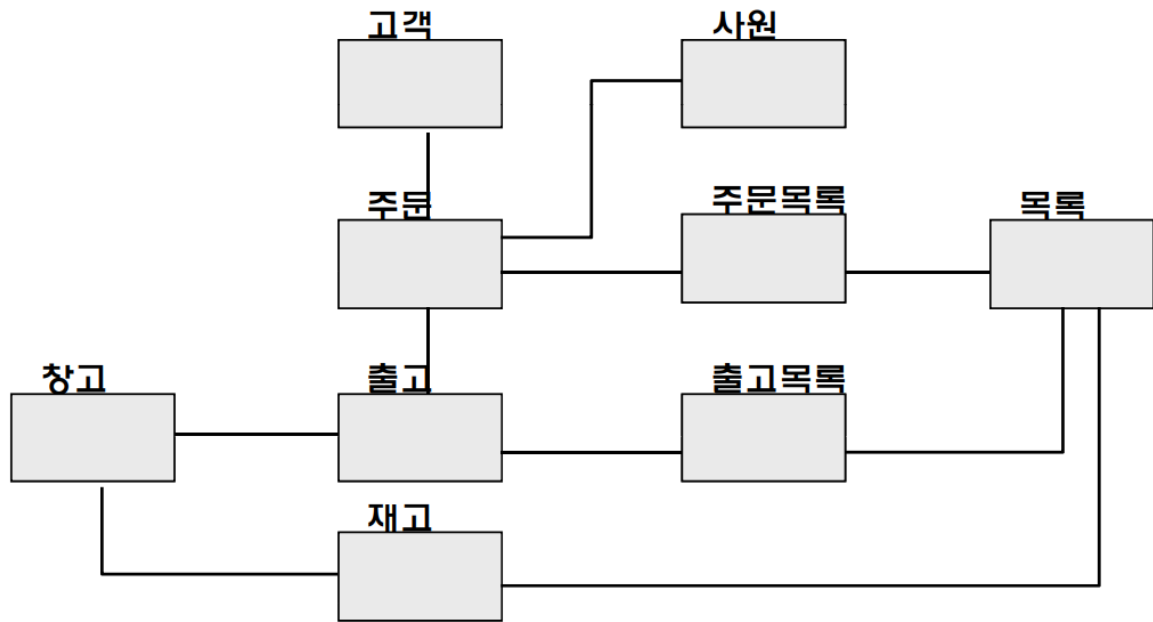
- 1) 엔터티를 그린다 (=직사각형, 단수명사 형태로 작성)
- 2) 엔터티를 적절하게 배치합니다 (=수평, 수직)
- 3) 엔터티간 관계를 설정합니다 (=각 엔터티 사이에 다이아몬드)
- 4) 관계명을 기술합니다 (=동사 형태)
- 5) 관계의 참여도를 기술합니다 (= 필수 mandatory, 선택 optional)
- 6) 관계 필수여부를 기술합니다 (= 1:1, 1:N, M:N)

※ 논리 모델링단계만 존재하는 M:N은 관계해소를 통해 1:N으로 만들어주는것이 좋습니다.

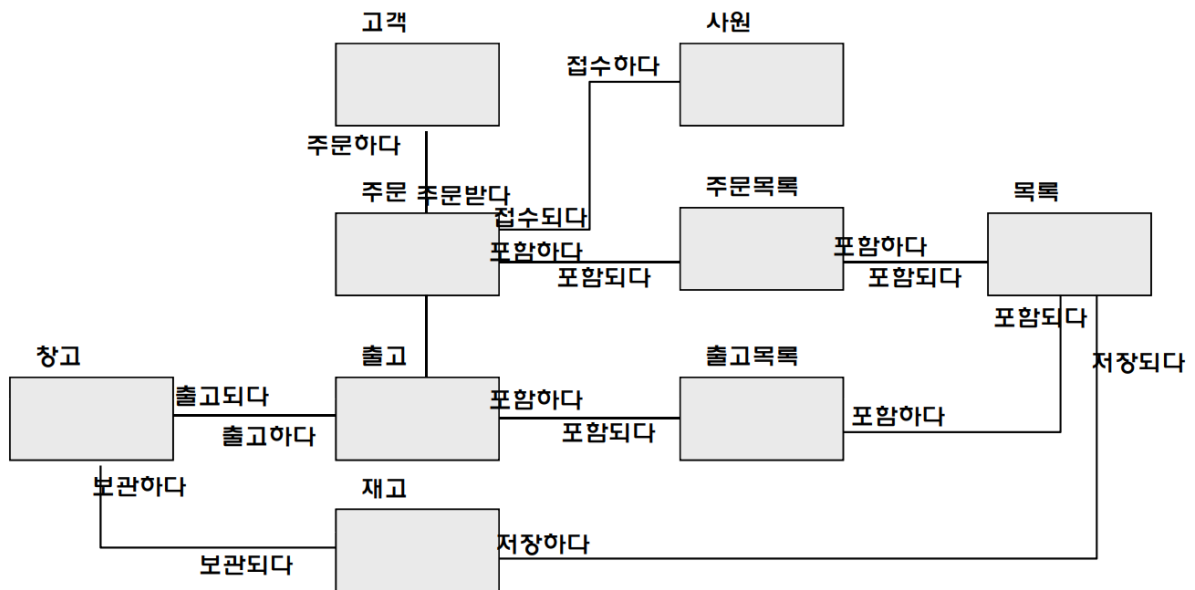
※ 요리-식재료, 상품-주문



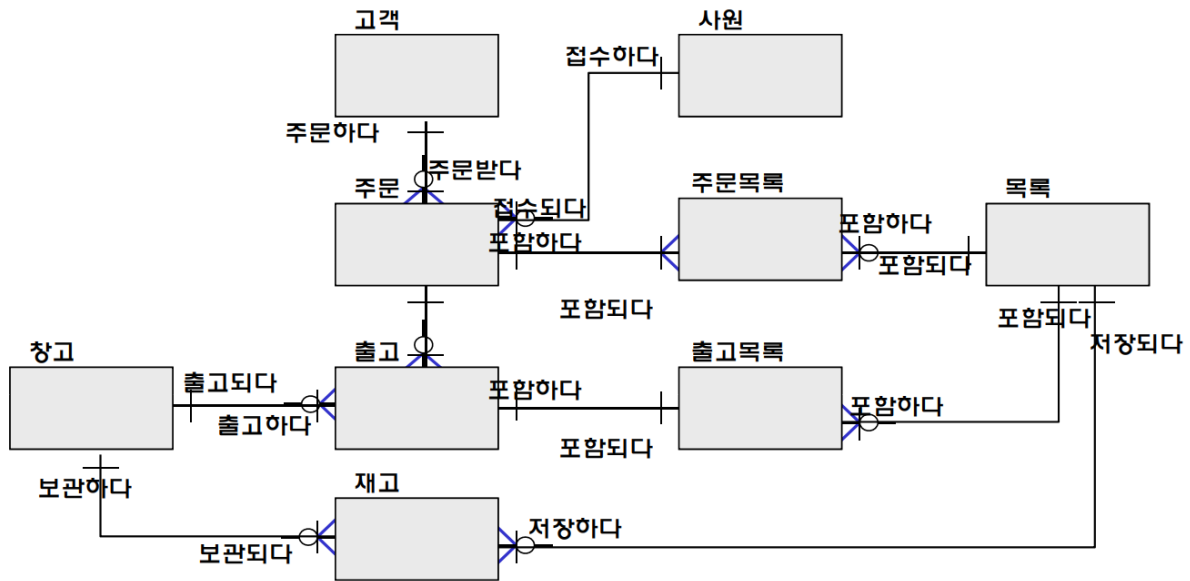
< 엔터티 배치 >



< 관계의 연결 >



< 관계의 표시 >



< 관계의 차수와 선택성 표시 >



좋은 데이터 모델이란? 업무흐름이 ERD만 봐도 쉽게 파악되는 모델이 좋은 데이터 모델이라 할 수 있습니다.

| 중요성                       | 설명  |
|---------------------------|---|
| 완전성(Completeness)         | 업무에서 필요로 하는 모든 데이터가 데이터 모델에 정의됨   |
| 중복배제(Non-Redundancy)      | 하나의 데이터베이스 내에 동일한 사실은 반드시 한 번만 기록하여야 함  |
| 업무규칙(Business Rules)      | 무규칙(Business Rules)을 데이터 모델에 표현하고 이를 해당 데이터 모델을 활용하는 모든 사용자가 공유                     |
| 데이터 재사용(Data Reusability) | 데이터의 통합성과 독립성에 대해서 충분히 고려   |
| 의사소통(Communication)       | 정보시스템을 운용, 관리하는 많은 관련자들이 설계자가 정의한 업무 규칙들을 동일한 의미로 받아들이고 정보시스템을 활용할 수 있게 하는 역할을 하게 됨 |
| 통합성(Integration)          | 가장 바람직한 데이터 구조의 형태는 동일한 데이터는 조직의 전체에서 한번만 정의되고 이를 여러 다른 영역에서 참조, 활용하는 것             |

## 10. 엔터티 개요

- 우리말로 실체, 객체라고 번역하기도 하는데 실무적으로 엔터티라는 외래어를 많이 사용합니다. 현실 세계에 존재하는 객체인 사원, 고객, 상품등은 키 엔터티라고 합니다.

- 엔터티는 업무에 필요하고 유용한 정보를 저장하고 관리하기 위한 집합적인 것(thing)을 의미합니다.
- 키 엔터티 간의 작용(=관계,Relationship)에 따라 만들어지는 엔터티를 액션 엔터티라고 합니다.



과목



강사

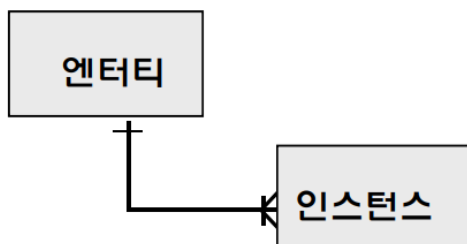


사건

※ 엔터티는 사람, 장소, 물건, 사건, 개념 등의 명사에 해당합니다.

- 엔터티는 업무상 관리가 필요한 관심사
- 엔터티는 저장이 되기 위한 어떤 것(Thing)

### 엔터티-인스턴스 ERD



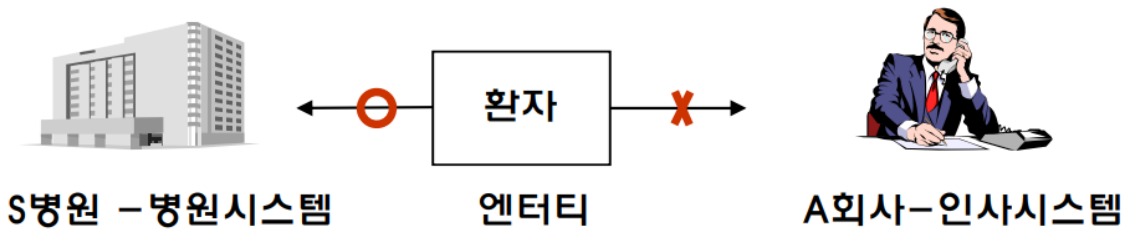
엔터티는 인스턴스의 집합

### 엔티티-인스턴스의 예

| 엔터티 | 인스턴스     |
|-----|----------|
| 과 목 | 수학       |
|     | 영어       |
| 강 사 | 이춘식      |
|     | 조시형      |
| 사 건 | 2010-001 |
|     | 2010-002 |

### 10-1. 엔터티 특징

- 반드시 해당 업무에서 필요하고 관리하고자 하는 정보이어야 합니다.



- 유일한 식별자에 의해 식별이 가능해야 합니다.

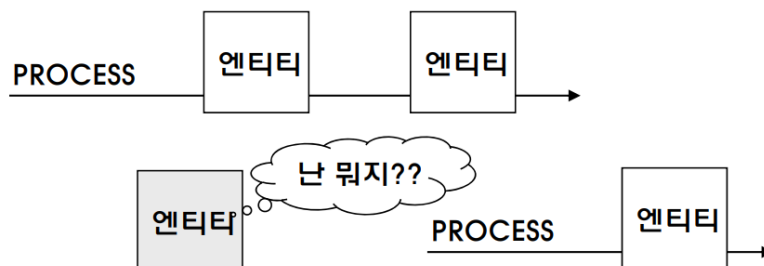


모두다 동일한 이름, 속성, 관계??

- 영속적으로 존재하는 인스턴스의 집합이어야 합니다.('한 개'가 아니라 '두 개 이상')

|    |   |     |        |
|----|---|-----|--------|
| 회사 | ⇒ | 엔터티 | 인스턴스   |
|    |   | 회사  | LG CNS |
| 병원 | ⇒ | 엔터티 | 인스턴스   |
|    |   | 병원  | 삼성의료원  |

- 엔터티는 업무 프로세스에 의해 이용되어야 합니다.



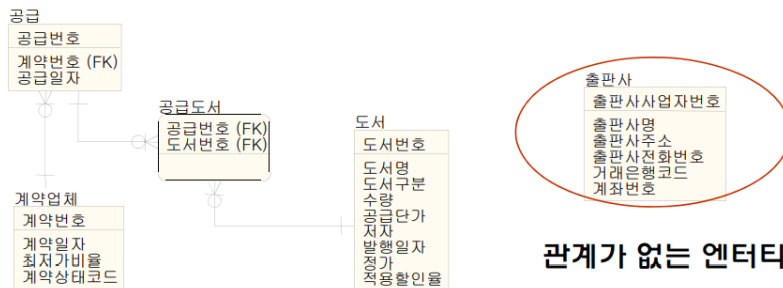
- 엔터티는 반드시 속성이 있어야 합니다.

|      |
|------|
| 태풍   |
| 태풍이름 |
| 발생지역 |
| 중속   |

|      |
|------|
| 날씨   |
| 날씨이름 |
|      |

엔티티?

- 엔티티는 다른 엔티티와 최소 한 개 이상의 관계가 있어야 합니다.



## 10-2. 엔티티 이름 규칙

- 가능하면 현업업무에서 사용하는 용어를 사용합니다.
- 가능하면 약어를 사용하지 않습니다.
- 단수명사 또는 복합 명사형태를 사용합니다.
- 모든 엔티티에서 유일하게 이름이 부여되어야 합니다.
- 엔티티 생성 의미대로 이름을 부여합니다.

## 11. 속성의 개요

- 사전적인 의미로는 사물(事物)의 성질, 특징 또는 본질적인 성질, 그것이 없다면 실체를 생각할 수 없는 것으로 정의
- 업무에서 필요로 하는 인스턴스로 관리하고자 하는 의미상 더 이상 분리되지 않는 최소의 데이터 단위(=원자성)
- 범위를 한정하는 한정자 + 값을 표현하는 명사를 구분합니다. (번호 vs 글번호, 주소 vs 집주소)
- 의미상 더 이상 분리되지 않습니다.
- 엔티티를 설명하고 인스턴스의 구성요소가 됩니다.

## 엔터티



강사

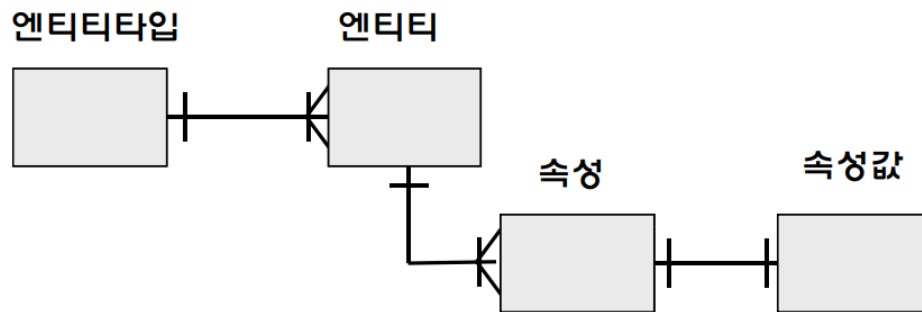
엔터티는 속성들에 의해 설명된다.

## 속성

- 이름
- 주소
- 생년월일
- 계약일자
- 전문분야

### 11-1. 속성 - 엔터티, 인스턴스, 속성, 속성값의 관계

- 한 개의 엔터티는 두 개 이상의 인스턴스의 집합이어야 합니다.
- 한 개의 엔터티는 두 개 이상의 속성을 갖습니다.
- 한 개의 속성은 한 개의 속성값을 갖습니다.



### 11-2. 속성 표기법

속성의 표기법은 엔터티 내에 이름을 포함하여 표현합니다.

| 과목           | 강사         | 사건           |
|--------------|------------|--------------|
| 과목이름         | 강사이름       | 사건번호         |
| 교재이름<br>생성일자 | 주소<br>생년월일 | 발생장소<br>발생일시 |

### 11-3. 속성의 특징

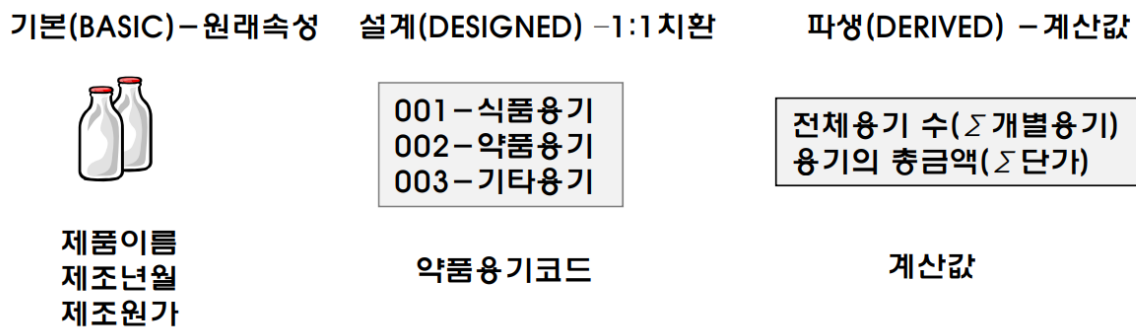
- 엔터티와 마찬가지로 반드시 해당 업무에서 필요하고 관리하고자 하는 정보이어야 합니다. (예, 강사의 교재이름)



- 정규화 이론에 근간하여 정해진 주식별자에 함수적 종속성을 가져야 합니다.
- 하나의 속성에는 한 개의 값만을 가진다. 하나의 속성에 여러 개의 값이 있는 다중값일 경우 별도의 엔터티를 이용하여 분리합니다.

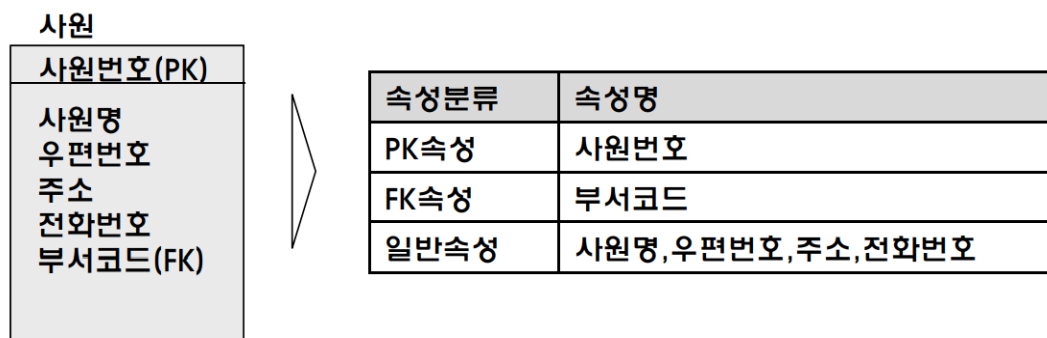
#### 11-4. 속성의 분류(1)

업무 분석을 통해 구분한 기본속성, 원래 업무상 존재하지 않지만 설계하면서 도출하는 설계 속성 또는 다른 속성으로부터 계산이나 변형을 통해 파생된 속성으로 분류합니다.



#### 11-4. 속성의 분류(2)

엔터티 구성 방식에 따라 분류할 수도 있습니다.



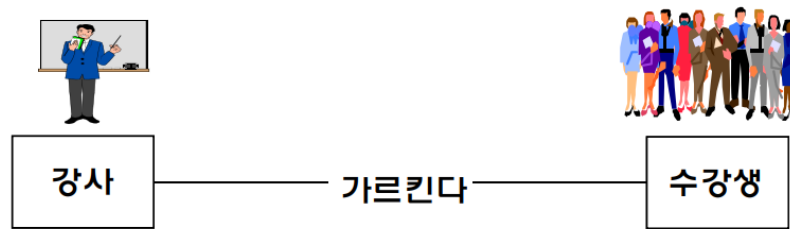
#### 11-5. 속성의 이름규칙

- 해당업무에서 사용하는 이름을 부여 하도록 합니다.
- 서술식 속성명은 사용하지 않도록 합니다.
- 약어사용은 가급적 제한합니다.

- 전체 데이터모델에서 유일성 확보하는 것이 좋습니다.

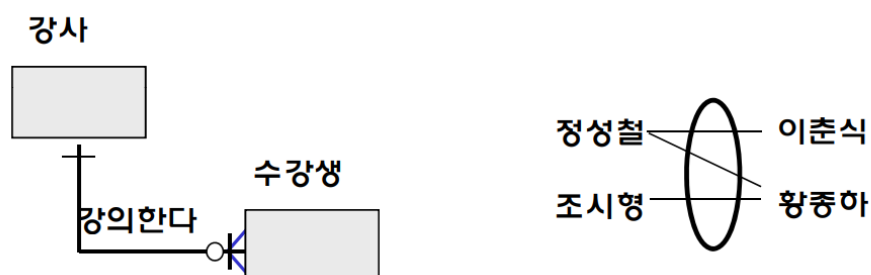
## 12. 관계의 정의

- 사전적으로 정의하면 상호 연관성이 있는 상태
- 엔터티의 인스턴스 사이의 논리적인 연관성으로서 존재의 형태로서나 행위로서 서로에게 연관성이 부여된 상태



### 12-1. 관계의 페어링

- 각각의 엔터티의 인스턴스들은 자신이 관련된 인스턴스들과 관계의 어커런스로 참여하는 형태
- 인스턴스 각각은 자신의 연관성을 가지고 있을 수 있음. 이것을 집합하여 '강의'라는 관계도출함.



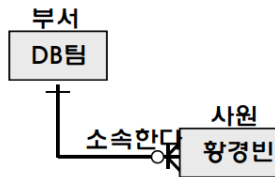
### 12-2. 관계의 분류

관계가 존재에 의한 관계와 행위에 의한 관계로 구분될 수 있는 것은 관계를 연결함에 있어 어떤 목

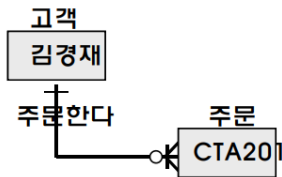
적으로 연결되었느냐에 따라 분류하기 때문입니다.

- 필수(mandatory)
- 선택(option)

존재에 의한 관계

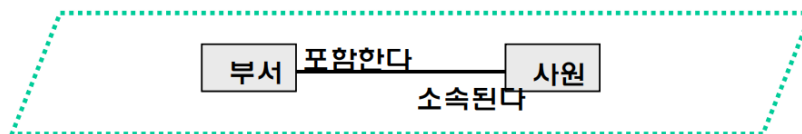


행위에 의한 관계



### 12-3. 관계명 정의 규칙

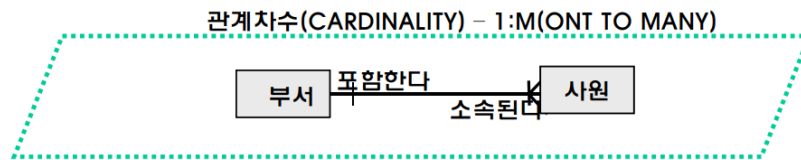
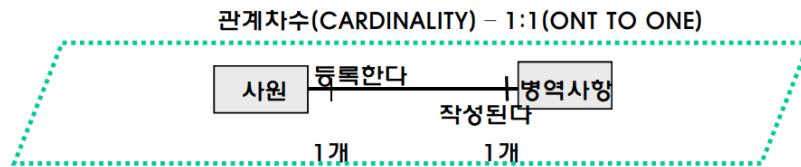
- 엔터티에서 관계가 시작되는 편을 관계시작점(The Beginning)이라고 부르고 받는 편을 관계끝점(The End)이라고 부릅니다.
- 관계 시작점과 끝점 모두 관계이름을 가져야 하며 참여자의 관점에 따라 관계이름이 능동적(Active)이거나 수동적(Passive)으로 명명됩니다.
- 구체적이지 않는 애매한 동사를 피합니다. (예. 관련이 있다, ~이다, ~한다)
- 현재형으로 표현합니다. (예. 수강신청을 한다, 예매를 한다)



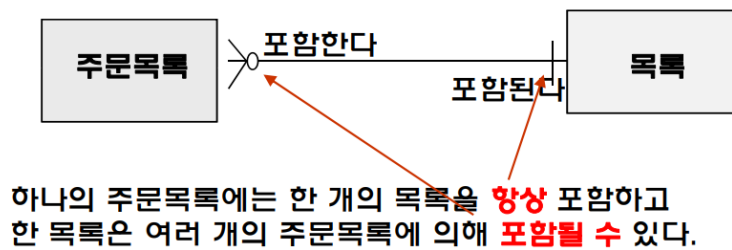
### 12-3. 관계의 차수(Degree)

- 두 개의 엔터티간 관계에서 참여자의 수를 표현하는 것을 관계차수(Cardinality)라고 합니다.
- 가장 일반적인 관계차수 표현방법은 1:M, 1:1, M:N 이며, 가장 중요하게 고려해야 할 사항은 한 개의 관계가 존재하느냐 아니면 두 개 이상의 멤버십이 존재하는지를 파악하는 것이 중요합니다.

- 한 개가 참여하는 경우는 실선을 그대로 유지하고 다수가 참여한 경우는(Many) 까마귀 발과 같은 모양으로 그려줍니다.

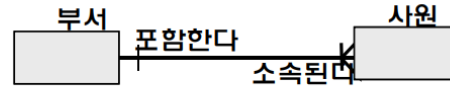


- 선택참여관계는 ERD에서 관계를 나타내는 선에서 선택참여하는 엔터티 쪽을 원으로 표시하고, 필수참여는 아무런 표시를 하지 않습니다.



#### 12-4. 관계 읽기

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

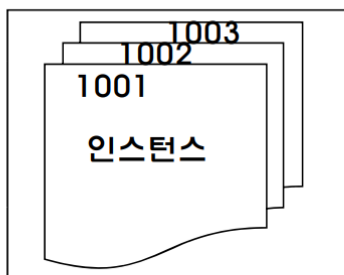


| 각각의 / 하나의 | 기준엔터티 (Source) | 관계 차수 | 관련엔터티 (Target) | 선택사항 필수/ 선택 | 관계명  |
|-----------|----------------|-------|----------------|-------------|------|
| 각각의       | 사원은            | 한     | 부서에            | 때때로         | 속한다  |
| 각         | 부서에는           | 여러    | 사원이            | 항상          | 소속된다 |

### 13. 식별자(Identifier)

엔터티는 인스턴스들의 집합입니다. 여러 개의 집합체를 담고 있는 하나의 통에서 각각을 구분할 수 있는 논리적인 이름이 있어야 하는데, 이 구분자를 식별자(Identifier)라고 합니다.

※ 하나의 엔터티에서 개별 레코드를 식별할 수 있는 속성의 조합을 의미함.



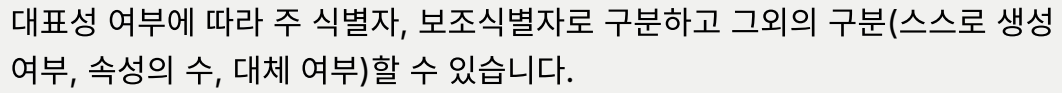
주문 엔터티



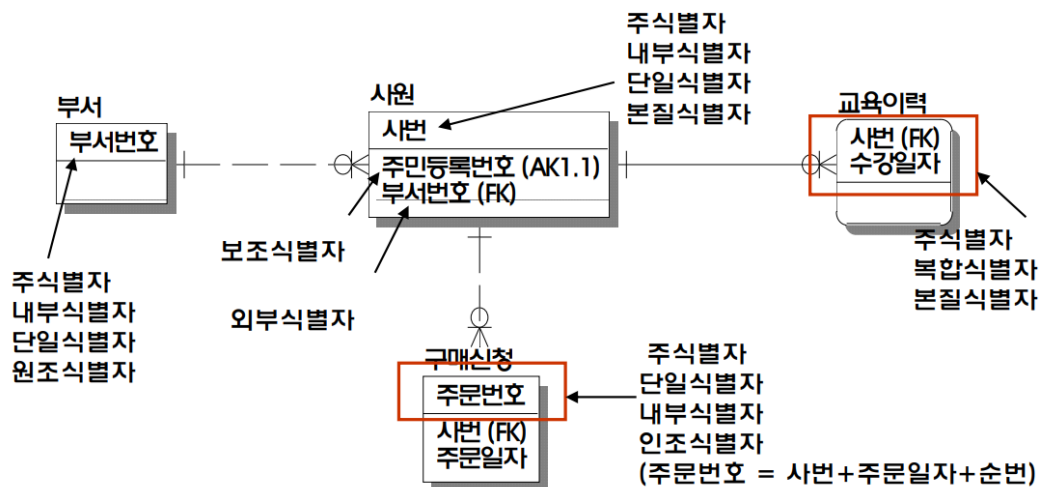
식별자는 엔터티내에서 인스턴스들을 구분할 수 있는 구분자이다.

#### 13-1. 식별자 특징

- 주식별자에 의해 엔터티내에 모든 인스턴스들이 유일하게 구분되어야 합니다.
- 주식별자를 구성하는 속성의 수는 유일성(UNIQUE)을 만족하는 최소의 수가 되어야 합니다.
- 지정된 주식별자의 값은 자주 변하지 않는 것이어야 합니다.
- 주식별자가 지정이 되면 반드시 값이 들어와야(NOT NULL) 합니다.

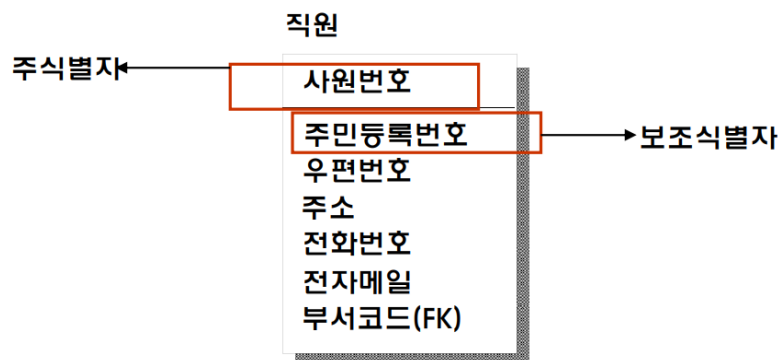


### 13-2. 식별자 표기법



### 13-3. 식별자 도출기준

- 해당 업무에서 자주 이용되는 속성을 주식별자로 지정합니다.



- 명칭, 내역 등과 같이 이름으로 기술되는 것들은 가능하면 주식별자로 지정하지 않습니다.





엔터티 사이 관계유형은 업무특징, 자식엔터티의 주식별자구성, SQL 전략에 의해 결정합니다.

### 13-5. 식별자 관계

- 자식엔터티의 주식별자로 부모의 주식별자가 상속이 되는 경우를 식별자 관계 (Identifying Relationship)라고 지칭합니다.
- 부모로부터 받은 식별자를 자식엔터티의 주식별자로 이용하는 경우는 Null값이 오면 안 되므로 반드시 부모엔터티가 생성되어야 자기 자신의 엔터티가 생성되는 경우입니다.
- 두 엔터티간의 관계에 있어서 부모 엔터티가 자식 엔터티를 만들어 내는데 필수적인 역할을 하고 있다면 식별 관계라고 할 수 있습니다.



< 외부식별자의 주식별자 역할 >

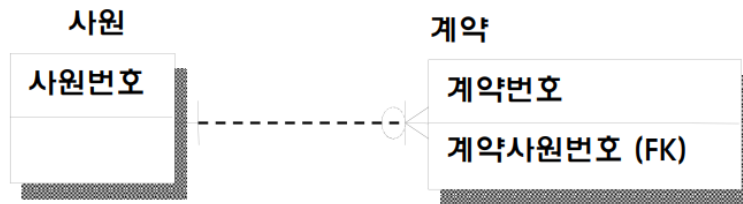
### 13-6. 비식별자 관계

부모엔터티로부터 속성을 받았지만 자식엔터티의 주식별자로 사용하지 않고 일반적인 속성으로만 사용하는 경우를 비식별자 관계(Non-Identifying Relationship)라고 하며 다음의 네가지 경우에 비식별자 관계에 의한 외부속성을 생성합니다.

- 자식엔터티에서 받은 속성이 반드시 필수가 아니어도 무방하기 때문에 부모 없는 자식이 생성될 수 있는 경우
- 엔터티별로 데이터의 생명주기(Life Cycle)를 다르게 관리할 경우
- 여러 개의 엔터티가 하나의 엔터티로 통합되어 표현되었는데 각각의 엔터티가 별도의 관계를 가질 때

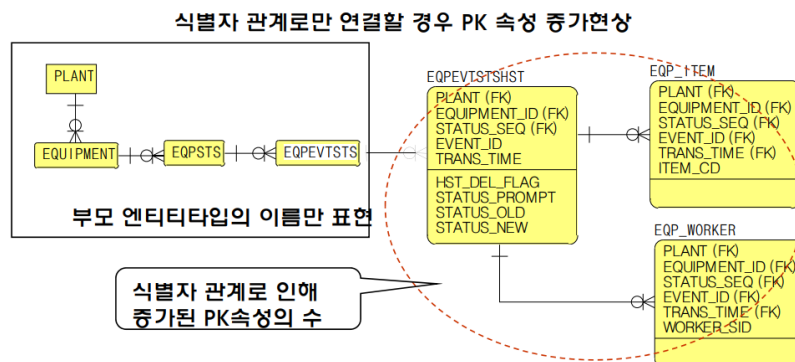


- 자식엔터티에 주식별자로 사용하여도 되지만 자식엔터티에서 별도의 주식별자를 생성하는 것이 더 유리하다고 판단될 때



식별 관계로만 설정할 경우의 문제점은 어떤것이 있을까요?

#### ▼ 문제점



세 개의 테이블에서 정보를 가져오는 SQL구문을 만들면 어떻게 될까요?

원 부모엔터티 : 1개

2대 부모엔터티 : 2개 이상 = 원부모 1개 + 추가 1개 이상 +

3대 부모엔터티 : 3개 이상 원부모 1개 + 2대 1개 + 추가 1개 이상

68

3대 부모엔터티 : 3개 이상 = 원부모 1개 + 2대 1개 + 추가 1개 이상

3대 부모엔터티 : 3개 이상 = 원부모 1개 + 2대 1개 + 3대 1개 + 추가 1개 이상

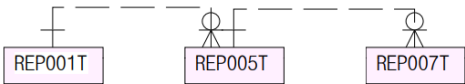
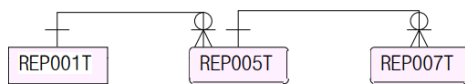
4대 부모엔터티 : 4개 이상 = 원부모 1개 + 2대 1개 + 3대 1개 + 4 1개 + 추가 1개 이상

...

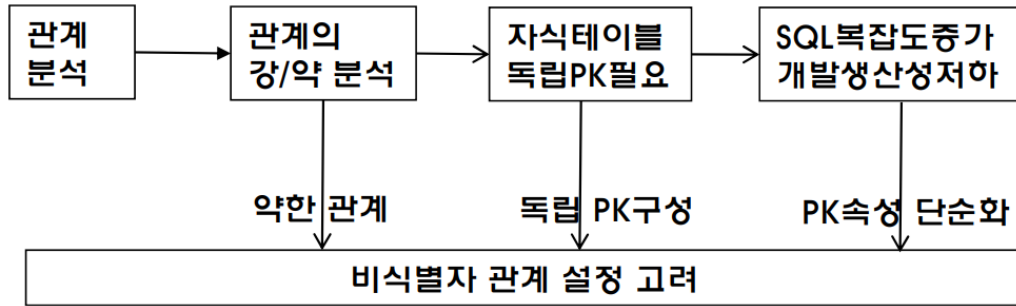
고작 3개 정도의 엔터티를 조인했을 뿐인데 SQL구문의 WHERE절이 매우 길어진 사실을 확인할 수 있습니다. 실제로 프로젝트에서는 개발자가 개발할 때 당연히 데이터 모델

을 참조하면서 엔터티와 관계를 이용하여 개발해야 하는데 생성된 엔터티 스키마 정보만을 보고 개발하는 경우가 많습니다.

```
SELECT A.EVENT_ID, A.TRANS_TIME, A.HST_DEL_FLAG, A.STATUS_
A.STATUS_OLD, A.STATUS_NEW
FROM EQPEVTSTSHST A, EQP_ITEM B, EQP_WORKER C
WHERE A PLANT= B PLANT WHERE A.PLANT= B.PLANT
AND A.EQUIPMENT_ID= B.EQUIPMENT_ID
AND A.STATUS_SEQ= B.STATUS_SEQ
AND A.EVENT_ID= B.EVENT_ID
AND A.TRANS_TIME= B.TRANS_TIME
AND B ITEM CD 'A001' AND B.ITEM_CD= 'A001'
AND A.PLANT = C.PLANT
AND A.EQUIPMENT_ID= C.EQUIPMENT_ID
AND A.STATUS_SEQ= C.STATUS_SEQ
AND A.EVENT_ID= C.EVENT_ID
AND A.TRANS_TIME= C.TRANS_TIME
AND C.WORKER_SID= 'A012008001';
```

| 비식별자관계  | 식별자 관계   |
|---|--|
|    |  |
| SELECT A.기본가중치, A.조정가중치<br>FROM REP007T A, REP005T B, REP001T C<br>WHERE A.분야번호 = B.분야번호<br>AND B.점검번호 = C.점검번호<br>AND C.점검번호 = '105' | SELECT A.기본가중치, A.조정가중치<br>FROM REP007T A<br>WHERE A.점검번호 = '105'                    |
| 테이블 REP007T에서 점검번호 = '301' 인 데이터를 조회  |  |

<식별자와 비식별자관계에 따른 SQL 비교>



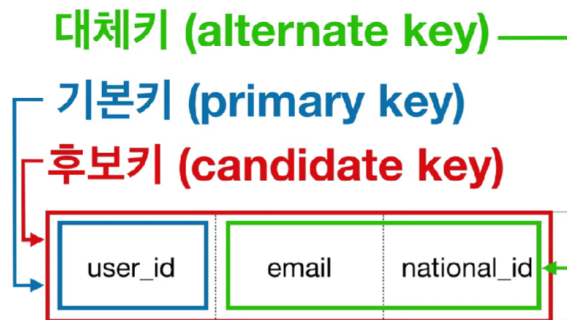
<비식별자 관계 설정 고려사항>

| 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    | <ul style="list-style-type: none"> <li>- 반드시 부모엔터티 종속</li> <li>- 자식 주식별자구성에 부모 주식별자 포함 필요</li> <li>- 상속받은 주식별자속성을 타 엔터티에 이전 필요</li> </ul> | <ul style="list-style-type: none"> <li>- 약한 종속관계</li> <li>- 자식 주식별자구성을 독립적으로 구성</li> <li>- 자식 주식별자구성에 부모 주식별자 부분 필요</li> <li>- 상속받은 주식별자속성을 타 엔터티에 차단 필요</li> <li>- 부모쪽의 관계참여가 선택관계</li> </ul> |

<식별자와 비식별자관계 비교>

프로젝트에서 데이터 모델링을 위한 충분한 시간을 할당하는 일이 거의 없고, 급박하게 진행되는 프로젝트에서 짧은 시간 내에 ERD를 작성하다 보면 대부분 비식별 관계(=점선)으로 표기할때가 있지만, 비 식별 관계는 엔터티 간의 중요한 영향 관계를 보여주지 못하므로 가 급적 식별,비식별 관계는 잊어버리고 표기하는 것이 좋겠습니다.

#### ▼ 식별관계 부연설명



| user_id | email      | national_id | name    | city  |
|---------|------------|-------------|---------|-------|
| 1       | a@mail.com | 100001      | egoing  | seoul |
| 2       | b@mail.com | 100002      | leezche | jeju  |
| 3       | c@mail.com | 100003      | egoing  | jeju  |

< 출처 - 오픈튜토리얼스 >

### 중복키 (composite key)

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

< 출처 - 생활코딩 >

## 13-7. 정리

- 엔터티(□)는 논리적 모델링시 테이블(Table)로 변환됩니다. 각 속성을 그룹화한 개념입니다.

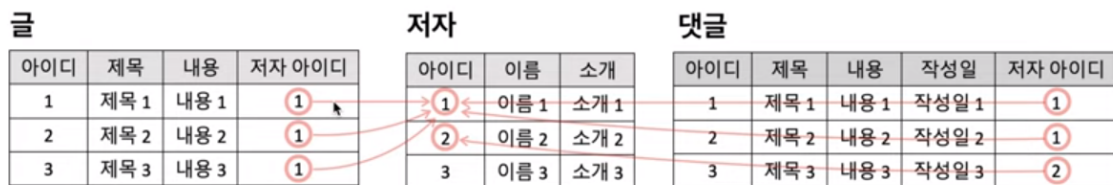
ex>글, 저자, 댓글 ...

- **속성(○)**은 논리적 모델링시 열(Column)으로 변환됩니다. 각각의 엔터티에 소속됩니다.

ex> 글 번호, 글 제목, 저자, 작성일 ..

저자번호, 저자 이름, 저자 연락처, 저자 소개 ...

- **관계(◇)**는 PK나 FK로 변환됩니다.



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 |

## ▼ 데이터 모델링 실습

[데이터모델링.zip](#)

< 압축을 해제하고, draw.io에서 xml파일을 불러들여서 확인 하세요 >

[bookstore.drawio](#)