



배워두면 좋은 SQL



SQL Developer



자격증 준비하면서 익히자!

기본 배경 지식부터  
자격증 준비까지 썩~다



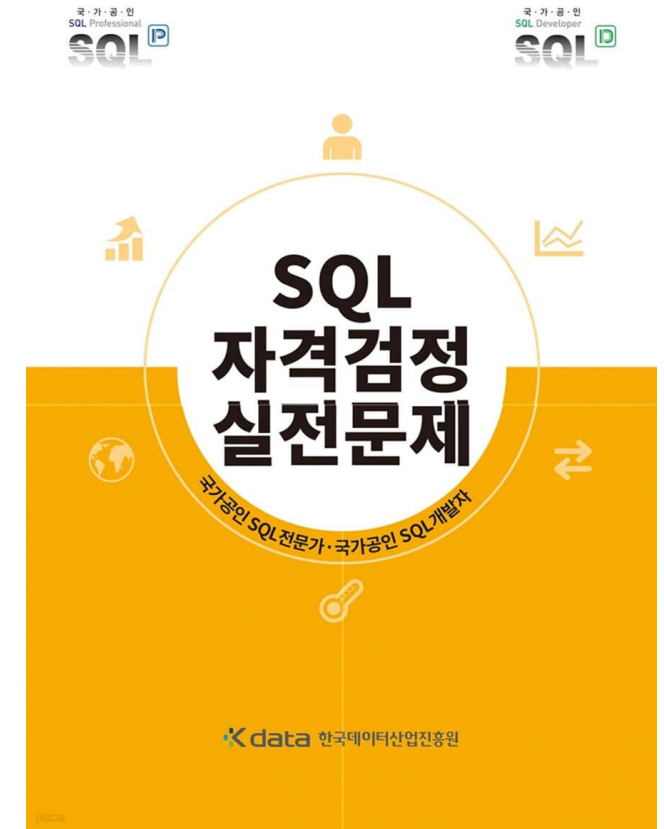
# 오늘 다룰 내용은 시험 범위 중 ...

## ■ 과목 I 데이터 모델링의 이해 (10 문항)

- 제1장 데이터 모델링의 이해 ◀ 여기에 해당하는 부분!
- 제2장 데이터 모델과 SQL

## ■ 과목 II SQL 기본과 활용 (40 문항)

- 제1장 SQL 기본
- 제2장 SQL 활용
- 제3장 SQL 최적화 기본 원리



# 모델링(Modeling)이란?

✓ 모델링이란? → 우리 현실세계를 모방/표현하는 것

## ▪ (저의) 머신러닝/딥러닝 모델링

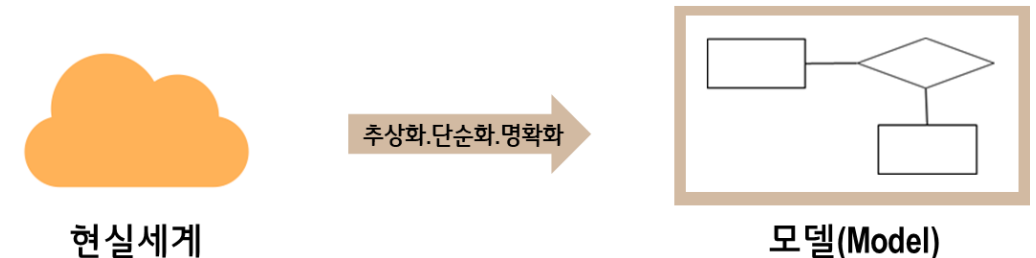
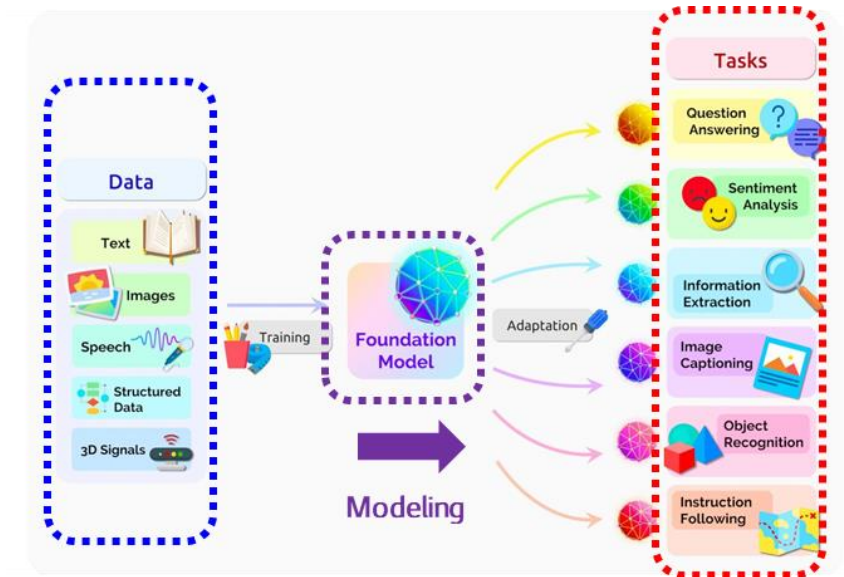
- 현실의 문제(TASK)를 데이터(DATA)를 통해 학습하고 해결하려고 하는 것을 의미함

## ▪ 데이터 모델링 정의

- 정보시스템을 구축하기 위한 데이터 관점의 업무 분석 기법
- 현실세계의 데이터(What)에 대한 약속된 표기법에 의해 표현하는 과정
- 데이터베이스를 구축하기 위한 분석/설계의 과정

## ▪ 데이터 모델링의 특징

- 추상화 (Abstraction): 현실세계를 일정한 형식으로 표현해야 함(추상화)
- 단순화 (Simplification): 이러한 일정한 형식은 단순해야 함(단순화)
- 명확화 (Clarity): 이러한 일정한 형식은 명확해야 함(명확화)



\* 데이터 모델링 3요소: 엔티티, 속성, 관계

# 모델링 관점 (3가지)

## ✓ 모델링 관점

여러 관점을 모두 고려하는 것은 시스템을 다양한 측면에서 이해하고, 더욱 정확하고 효율적인 시스템 설계를 가능하게 합니다.  
데이터만을 고려하거나 프로세스만을 고려하는 것보다 더 종합적인 시야를 제공하기 때문에 모델링 시 상당히 중요한 접근법입니다.

### ■ 데이터 관점 (Data View)

- 엔티티, 속성, 엔티티 간의 관계를 중심으로 데이터의 구조를 정의합니다.
- 데이터의 저장, 검색, 유지 관리 방법에 집중합니다.
- 예: 학생과 선생님 엔티티, 그리고 이들 사이의 수강 관계를 정의합니다.

### ■ 프로세스 관점 (Process View)

- 시스템이 어떤 기능을 수행하는지, 데이터가 시스템 내에서 어떻게 흐르는지에 집중합니다.
- 작업의 순서, 조건, 규칙을 포함하여 프로세스의 흐름을 정의합니다.
- 예: 성적을 입력하는 과정에서 선생님이 시스템을 통해 학생 식별자, 과목 코드, 성적을 입력하고, 시스템이 이를 처리하여 성적 데이터를 기록하는 프로세스입니다.

### ■ 데이터와 프로세스 관점 (Data & Process View)

- 데이터 구조와 프로세스의 동작이 상호작용하며 시스템을 구성하는 방식에 집중합니다.
- 데이터가 프로세스에 의해 어떻게 사용되는지, 그리고 그 결과가 시스템 전체에 어떻게 영향을 미치는지를 종합적으로 고려합니다.
- 예: 성적 입력 프로세스를 통해 학생과 과목 데이터가 어떻게 연동되어 성적 데이터를 생성하는지, 이 데이터가 시스템 내 다른 프로세스 (예: 성적 분석)에 어떻게 사용되는지를 통합적으로 보여줍니다.

# 데이터 모델링의 시 유의사항

## ✓ **파급효과(Leverage)**를 고려하자!

- 데이터 구조의 변경으로 인한 일련의 변경작업은 전체 시스템 구축 프로젝트에서 큰 위험요소가 된다.

## ✓ 복잡한 정보 요구사항을 **간결하게 표현(Conciseness)**하자!

- 데이터 모델은 구축할 시스템의 정보 요구사항과 한계를 가장 명확하고 간결하게 표현할 수 있는 도구이다.

## ✓ 좋은 **데이터 품질(Data Quality)**을 확보하자!

- 데이터베이스에 담겨 있는 데이터는 기업의 중요한 자산이다.
- 보관하고 있는 자산의 품질이 좋지 않다면, 가치 역시 떨어진다.





# 어떤 요소들이 데이터 품질을 위협할까? : 데이터 품질 저해요소

## ✓ 중복(Duplication)

- 같은 정보가 데이터베이스 내 여러 번 또는 여러 위치에 반복해서 저장되는 경우를 의미합니다.
  - 이로 인해 데이터 관리의 복잡성이 증가하고, 공간의 낭비 및 데이터 동기화 문제가 발생할 수 있습니다.
  - Ex. 한 학교 시스템에서 학생의 주소 정보가 학생 프로파일과 도서 대출 기록에 별도로 저장되어 있어, 학생이 이사할 경우 두 곳 모두 업데이트해야 하는 번거로움이 있습니다.

## ✓ 비-유연성(Inflexibility)

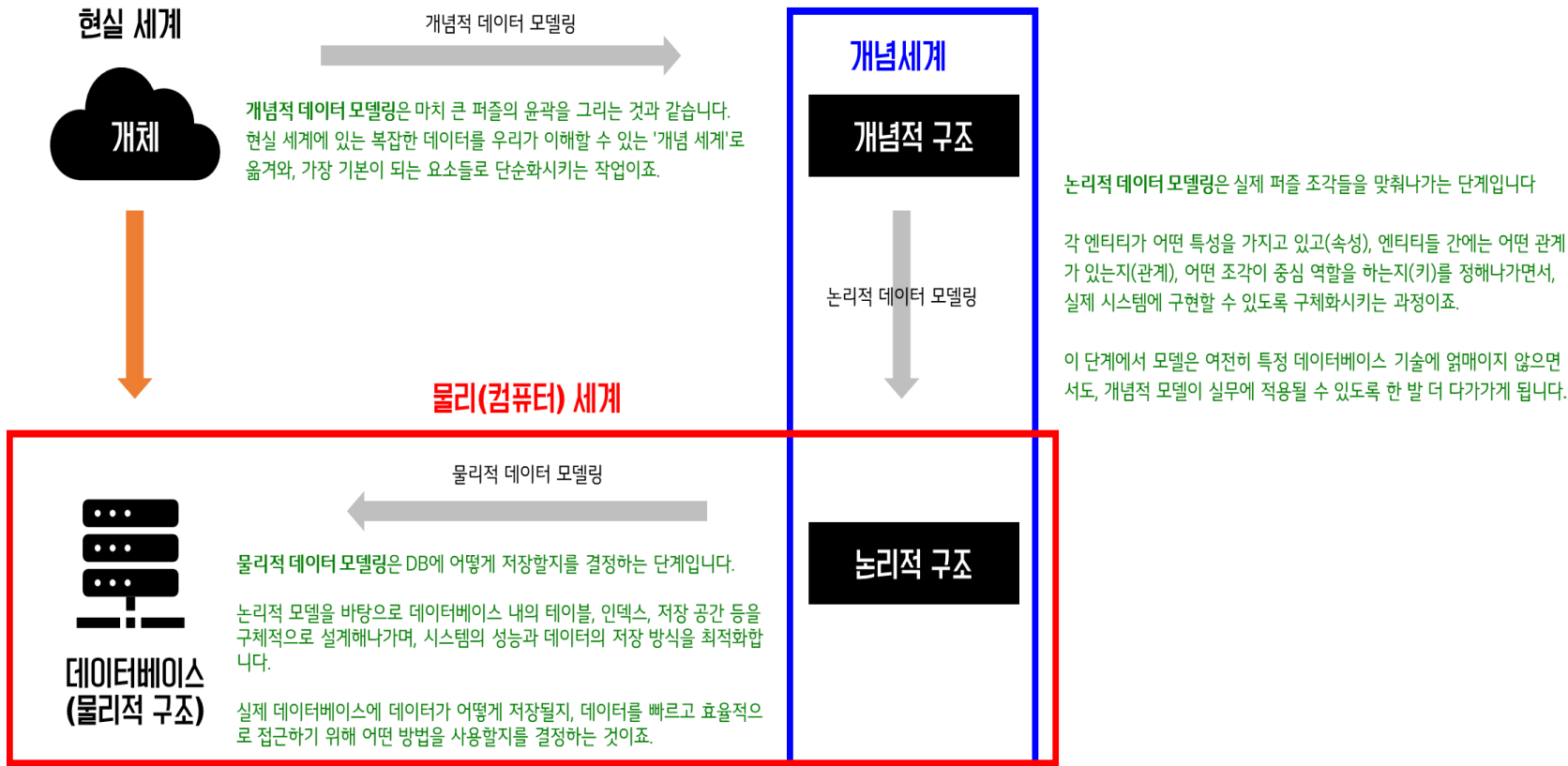
- 시스템이 데이터 구조나 프로세스의 변화에 쉽게 적응하지 못하고, 작은 변경에도 큰 수정이 필요한 상태를 말합니다.
  - 이는 시스템의 확장성과 유지보수성을 떨어뜨릴 수 있습니다.
  - Ex. 회사의 직원 관리 시스템에서 직원의 역할을 나타내는 단일 열을 사용하고 있을 때, 한 직원이 다중 역할을 수행하게 되면 새로운 데이터 구조로의 이행이나 추가적인 프로세스 설계가 필요할 수 있습니다.

## ✓ 비-일관성(Inconsistency)

- 데이터가 여러 소스로부터 수집될 때 각 소스 간 정보/저장 형식의 불일치 등의 원인으로 사용자에게 혼란을 주는 상태입니다.
  - 이는 데이터의 신뢰성과 정확성을 저하시킬 수 있습니다.
  - Ex. 마케팅 부서와 영업 부서에서 각각 고객의 연락처를 관리하는데, 같은 고객의 연락처가 한 부서에서는 최신으로 갱신되어 있는 반면 다른 부서에서는 구버전으로 남아 있어 고객에게 서로 다른 정보로 연락하는 일이 발생할 수 있습니다.

# 데이터 모델링 3단계

✓ 현실세계에서 데이터베이스까지 만들어지는 과정: 개념적 데이터 모델 → 논리적 데이터 모델 → 물리적 데이터 모델



데이터 모델링	내용
개념적 데이터 모델링	추상화 수준이 높고 업무 중심적이고 포괄적인 수준의 모델링 진행. 전사적 데이터 모델링, EA(Enterprise Architecture) 수립 시 많이 사용됨
논리적 데이터 모델링	시스템으로 구축하고자 하는 업무에 대해 Key, 속성, 관계 등을 정확하게 표현, 재사용성이 높음 (모델링된 구조가 한 시스템 또는 애플리케이션에서 한정되지 않음)
물리적 데이터 모델링	데이터베이스에 이식할 수 있도록 성능, 저장 등 물리적인 성격을 고려하여 설계

# 스키마란?

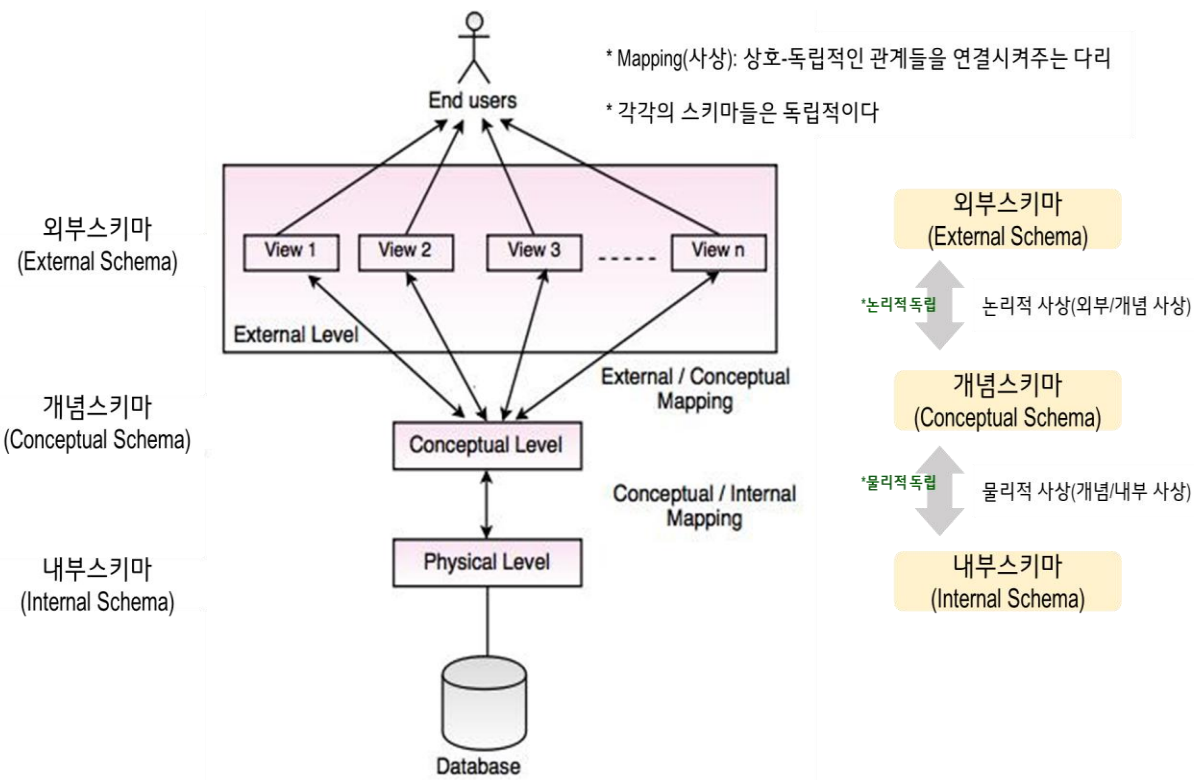
- ✓ 스키마는 데이터베이스의 구조, 데이터의 조직, 관계 등을 기술하는 메타데이터의 집합입니다. 데이터베이스의 설계도라고 볼 수 있으며, 데이터베이스를 이해하고 사용하는 데 필수적인 정보를 제공합니다. (= DB의 설계도)
- ✓ ANSI/SPARC의 3단계 구성(스키마)의 데이터 독립성 모델은 외부단계(스키마), 개념적 단계(스키마), 내부적 단계(스키마)로 구성된 서로 간섭되지 않는 전체적인 아키텍처를 기술합니다.

항목	내용	비고
외부스키마 (External Schema)	<ul style="list-style-type: none"><li>- View 단계, 여러 개의 사용자 관점으로 구성</li><li>- 각각의 개별 유저들이 바라보는 개인적 DB 스키마</li></ul>	사용자 관점
개념스키마 (Conceptual Schema)	<ul style="list-style-type: none"><li>- 개념 단계, 모든 사용자 관점을 통합한 조직 전체의 DB를 통합하여 기술하는 것</li><li>- 모든 응용시스템들이나 사용자들이 필요로 하는 데이터를 통합한 조직 전체의 DB를 기술한 것, 데이터와 그들 간의 관계를 표현하는 스키마</li></ul>	통합 관점
내부스키마 (Internal Schema)	<ul style="list-style-type: none"><li>- 내부 단계, 내부 스키마로 구성, DB가 물리적으로 저장된 형식</li><li>- 물리적으로 데이터가 실제로 저장되는 방법을 표현하는 스키마</li></ul>	물리적 저장구조



# 스키마와 데이터 독립성

- ✓ 데이터 독립성: 하위 수준의 데이터 조직 변화가 상위 수준의 스키마에 영향을 주지 않는 성질을 말합니다.
- ✓ 사상(Mapping): 한 뷰(view)에서 다른 뷰로 데이터가 어떻게 저장되고 처리될지를 연결 짓는 규칙이나 절차를 정의합니다.
  - 이것은 데이터베이스가 여러 레벨 간에 일관성을 유지하고, 한 레벨에서의 변경이 다른 레벨에 불필요한 영향을 미치지 않도록 합니다.



독립성	내용
논리적 독립성	<ul style="list-style-type: none"><li>* 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원하는 것</li><li>* 논리적 구조가 변경되어도 응용 프로그램에 영향 없음</li></ul>
물리적 독립성	<ul style="list-style-type: none"><li>* 내부스키마가 변경되어도 외부/개념 스키마는 영향을 받지 않도록 지원하는 것</li><li>* 저장장치의 구조변경은 응용 프로그램과 개념스키마에 영향 없음</li></ul>

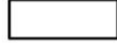


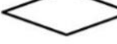

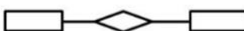
사상	내용
외부적/개념적 사상 (논리적 사상)	<ul style="list-style-type: none"><li>- 외부적 뷰(스키마)와 개념적 뷰(스키마)의 <b>상호 관련성<sup>1)</sup></b>을 정의함</li></ul> <p>Ex) 외부 스키마에서 제공되는 '최근 주문 목록'이라는 뷰는 실제로 '주문' 테이블의 '주문 날짜'에 따라 필터링된 결과를 보여주는 개념적 스키마에 매핑됩니다.</p>
개념적/내부적 사상 (물리적 사상)	<ul style="list-style-type: none"><li>- 개념적 뷰와 저장된 데이터베이스의 <b>상호 관련성<sup>1)</sup></b>을 정의함</li></ul> <p>Ex) '주문' 테이블에 있는 데이터가 어떤 파일 시스템에 저장될 것인지, 어떤 인덱스 구조를 사용할 것인지 등을 정의하는 것입니다.</p>

1) 상호 관련성: 각각의 스키마가 어떻게 서로 연결되어 있는지, 즉 어떻게 데이터의 한 형태가 다른 형태로 변환되는지를 정의하는 것을 말합니다.

# 데이터 모델링의 3요소와 ERD

✓ 데이터 모델링의 핵심 3 요소는 아래와 같습니다:

- 1) 업무가 관여하는 어떤 것(Things) => Entity(엔티티)
- 2) 어떤 것이 가지는 성격(Attributes) => Attributes(속성)
- 3) 업무가 관여하는 어떤 것 간의 관계(Relationships) => 관계(Relationships)

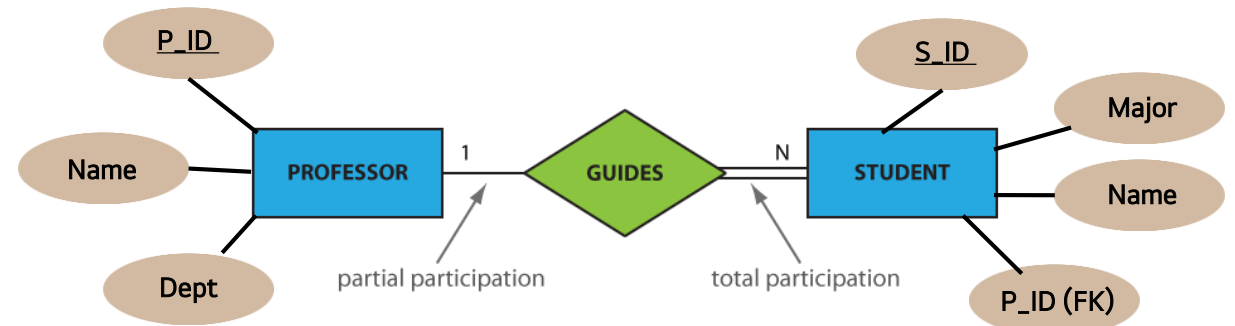
기 호	의 미
	개체 (테이블)
	속성
	기본키
	관계 (PK, FK)
	개체 타입과 속성을 연결
	개체간의 관계 타입

✓ ERD (Entity Relationship Diagram)

- ERD는 데이터 모델링 과정에서 생겨나는 엔티티, 속성, 그리고 관계라는 데이터 모델링 핵심 3 요소들을 시각적으로 나타내는 도구입니다.
- 이를 통해 복잡한 데이터 구조와 정보의 흐름을 한눈에 파악할 수 있습니다.

✓ ERD 기본 기호 (Peter Chen 표기법)

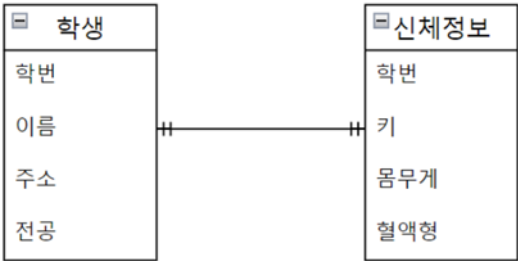
- 직사각형: 엔티티(Entity, 객체(테이블))를 나타냅니다.
- 타원: 엔티티의 속성(Attribute)을 나타냅니다.
- 다이아몬드: 엔티티 간의 관계(Relationship)를 나타냅니다.
- 선: 엔티티와 관계, 속성을 연결합니다.
- 선 위의 마커: 관계의 **차원(Cardinality)**과 **참여도(Optionality)**



# 관계의 차원 (Cardinality)

✓ 관계가 존재하는 두 entity 사이에 한 entity에서 다른 entity 몇개의 개체와 대응되는지 선으로 제약조건을 표기하게 됩니다.  
대표적인 관계로는 일대일(1:1), 일대다(1:N), 다대다(M:N) 등이 있습니다.

## ✓ 일대일(1:1)



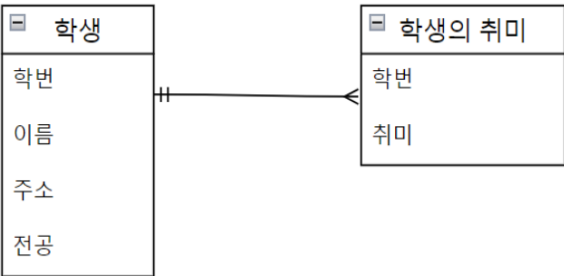
학생 테이블

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	컴퓨터
21004	박한나	부산	수학

신체정보 테이블

학번	키	몸무게	혈액형
21001	175	70	A
21002	169	65	B
21003	180	60	O
21004	170	85	B

## ✓ 일대다(1:N)



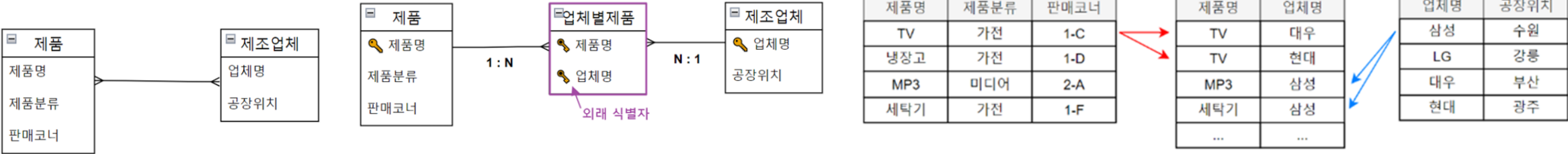
학생 테이블

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	컴퓨터
21004	박한나	부산	수학

학생의 취미 테이블

학번	취미
21002	낚시
21002	등산
21003	낚시
21004	여행

## ✓ 다대다(M:N)

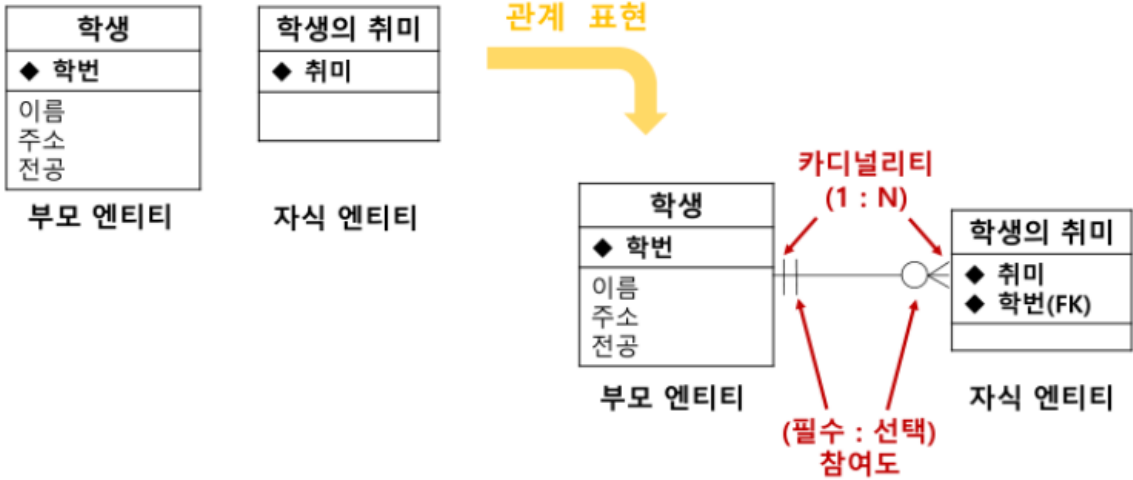


# 관계의 참여도 (Participation)

- ✓ 관계의 참여도(Participation)는 특정 엔터티가 해당 관계에 반드시 참여해야 하는지 여부를 나타냅니다.
- ✓ 관계는 필수(Mandatory) 참여와 선택(Optional) 참여로 구분됩니다.

관계의 참여도

Symbol	Meaning
	One-필수
	Many-필수
	One-선택
	Many-선택



## ※ 정리

관계의 차원 (Cardinality)과 관계의 참여도 (Participation)를 정리해보면 아래와 같이 정리할 수 있습니다:

관계	선택적 참여	필수적 참여
1:1 관계	-----	=====
1:n 관계	-----<	=====<
n:m 관계	>-----<	>=====<

# ERD 주요 표기 방법의 차이

## ✓ 기호 및 표기 방법의 차이

### ■ 1. Chen 표기법

- 1976년에 Peter Chen에 의해 제안된 이 표기법은 ERD의 가장 초기이자 가장 기본적인 표현 방식입니다. Chen 표기법에서는 엔티티를 사각형으로, 관계를 마름모로, 속성을 타원 또는 원으로 나타냅니다. 속성 중에서도 주요 키(Key)는 밑줄로 표시하여 엔티티 내에서 고유하게 식별할 수 있는 속성임을 나타냅니다.
- 관계는 관련된 엔티티 사이를 선으로 연결하고, 선 위나 마름모 안에 관계의 이름을 기술합니다. 다중성(카디널리티)은 관계 선 근처에 1:1, 1:N, M:N 등으로 표현하여 두 엔티티 간의 관계 유형을 명확하게 합니다.

### ■ 2. I/E 표기법 (Information Engineering 표기법)

- I/E 표기법은 James Martin에 의해 개발되었으며, 정보 공학에서 사용되는 방법입니다. 이 방법은 Chen 표기법과 유사한 구성 요소를 가지고 있지만, 엔티티 간의 관계를 나타낼 때 더 세분화된 규칙을 적용합니다.
- 특히 다중성 표현에 있어 더 명확하며, 엔티티 간의 최소 및 최대 관계를 괄호를 사용하여 표시합니다 (예: (0,1), (1,1), (0,N)). 이 표기법은 보다 복잡한 데이터 구조를 명확히 설명할 수 있는 장점이 있습니다.

### ■ 3. Barker 표기법

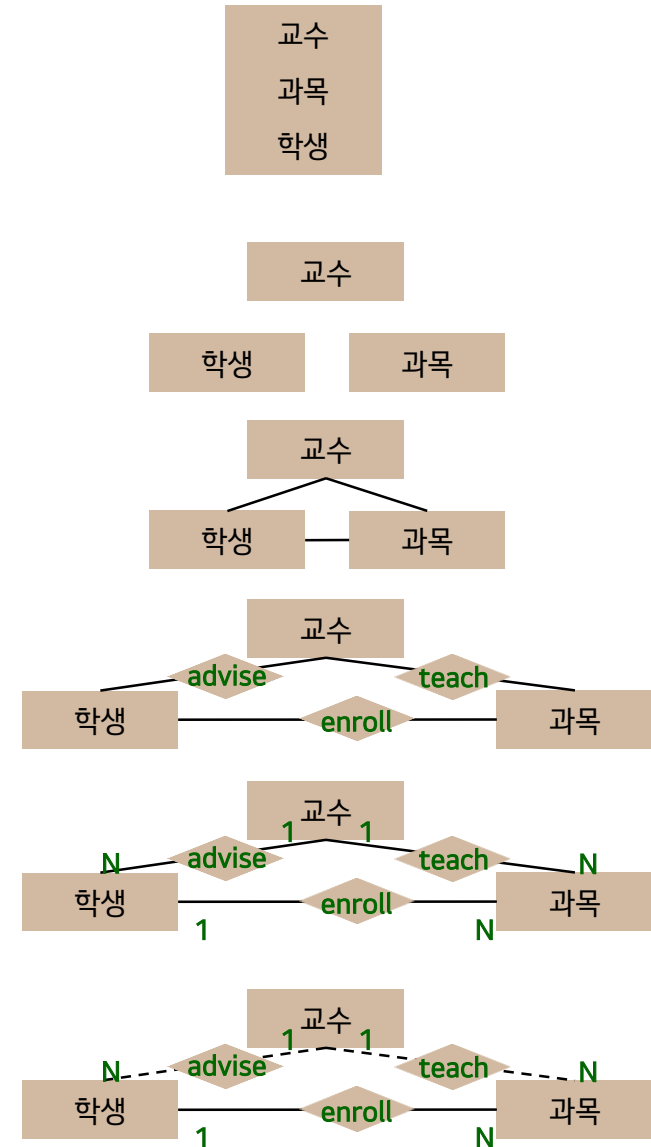
- Barker 표기법은 Richard Barker와 그의 동료들에 의해 개발된 방식으로, Oracle의 CASE 도구에 사용되었습니다. 이 방법은 실용성과 명확성에 중점을 두고 있으며, 엔티티를 사각형으로, 속성을 사각형 내부에 목록 형식으로 나열합니다.
- 관계는 선으로 표현되되, 다중성을 꺾쇠(<, >) 또는 동그라미(O)와 선(I)의 조합으로 나타내며, 관계의 방향성이 강조됩니다. Barker 표기법은 특히 엔터프라이즈 수준의 데이터 모델링에 적합하며, 이해하기 쉽고 직관적인 표현 방식입니다.

# ERD는 그림 어떻게 작성할까요? : ERD 작성 순서

\*예시로 임의로 만든 것임으로 관계가 실제 상황과 다를 수 있습니다.

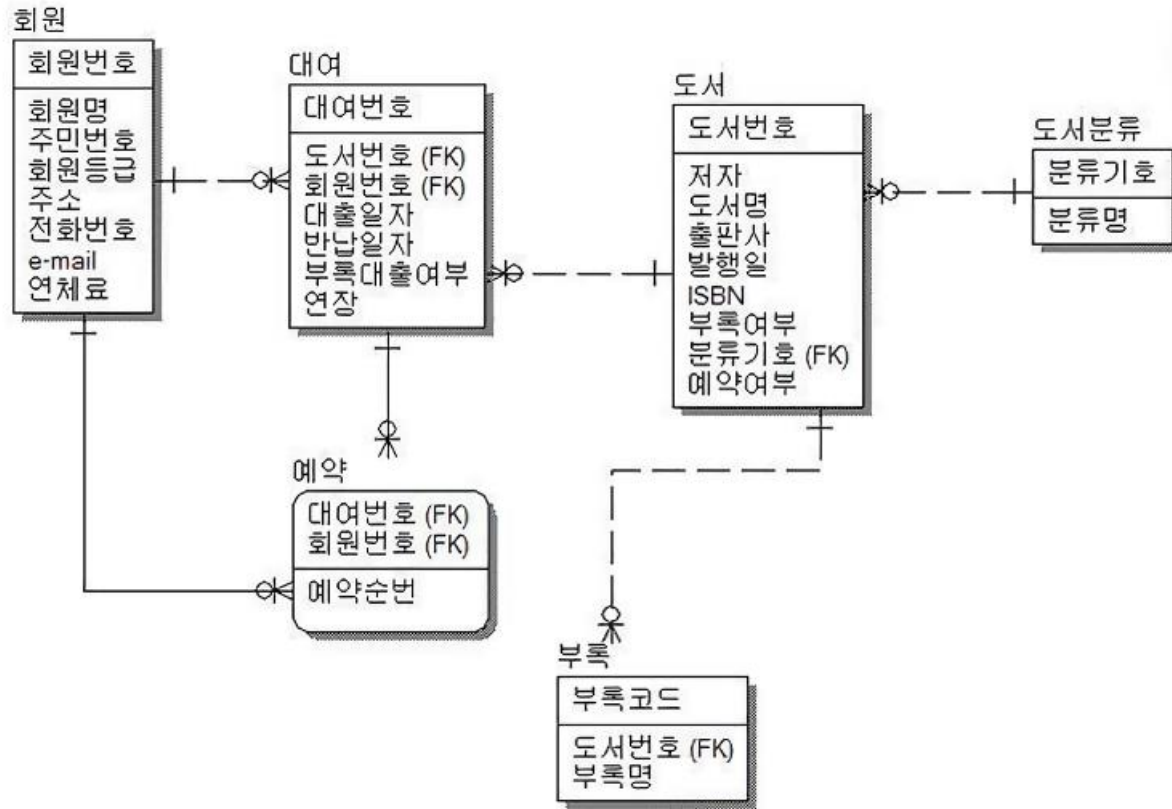


- 엔터티란 데이터베이스에서 저장하고자 하는 실제 세계의 객체(테이블)를 말합니다.
  - 예를 들어, 학생, 교수, 과목 등이 엔터티가 될 수 있습니다.
- 사각형 박스 안에 엔터티의 이름을 기술합니다.
- 관계를 고려하여 엔터티들을 적절히 배치합니다.
- 관련 있는 엔터티들은 가까이 놓고, 관련이 없는 엔터티들은 멀리 배치합니다.
- 이해하기 쉽고 가독성이 좋도록 배치합니다.
- 엔터티들 사이의 관계를 파악하고 연결선으로 표시합니다.
  - 예를 들어, 학생과 과목 간에는 해당 학생이 특정 과목을 '수강한다'라는 관계가 존재합니다.
- 엔터티 간 관계를 나타내는 적절한 이름을 연결선 위에 기술합니다.
  - 예를 들어, 학생과 수강내역 사이의 관계명은 '수강'이 될 수 있습니다.
- 한 엔터티의 인스턴스가 관계에 몇 번 참여할 수 있는지를 표시합니다.
  - 예를 들어, 한 학생은 여러 과목을 수강할 수 있으므로 '1:N' 참여도를 갖습니다.
- 관계에 반드시 참여해야 하는지 선택적인지를 표시합니다.
- 필수 관계는 실선, 선택 관계는 점선으로 표현합니다.
  - 예를 들어, 한 과목은 반드시 한 명 이상의 학생이 수강해야 하므로 필수 관계입니다.





## 예제: 도서관 시스템



### [회원 ↔ 대여]

- 회원번호PK가 대여 테이블에서 FK로 일반속성으로 쓰이고 있다. (점선)
- 회원은 대여를 여러개 할 수 있다. (1:N)
- 아예 대여하지 않은 회원이 있을 수 있다. (1:N[선택])
- 대여를 할땐 반드시 회원 정보가 필수로 존재해야한다. (1[필수]:N[선택])

### [도서 ↔ 대여]

- 도서번호PK가 대여 테이블에서 FK로 일반속성으로 쓰이고 있다. (점선)
- 도서가 과거에 여러번 대여된 기록이 있을 수 있으니. (1:N)
- 아예 대여하지 않은 도서가 있을 수 있다. (1:N[선택])
- 대여를 할땐 반드시 도서 정보가 필수로 존재해야한다. (1[필수]:N[선택])

# 엔티티(Entity)

## ✓ 엔티티(Entity)란?

- 실체, 객체의 어떠한 개념을 가진 명사에 해당합니다.
- 업무상 관리가 필요한 관심사, 저장되기 위한 것들을 말합니다.
  - 인스턴스의 집합으로서, 그 집합에 속하는 개체들의 특성을 설명할 수 있는 속성(Attribute)와 그들이 행하는 행위(Operation)의 집합임

## ✓ 엔티티의 특징

- 반드시 해당 업무에 필요하며, 관리하고자 하는 정보입니다. (업무에 쓰이는 정보여야 함)
- 유일한 식별자에 의해 식별이 가능해야 합니다.
- 영속적으로 존재하는 인스턴스의 집합입니다. (2개 이상을 가짐)
- 반드시 속성을 가집니다. (관계 엔티티의 경우에는 주 식별자만 가져도 인정한다)
- 다른 엔티티와 최소 한 개 이상의 관계를 가져야 합니다.

## ✓ 엔티티 명명(命名)법

- 업무에서 사용하는 용어여야 합니다.
- 약어를 사용하지 않아야 합니다.
- 단수명사를 사용해야 합니다.
- 고유성을 가져야 합니다.
- 생성 의미대로 이름을 부여해야 합니다.

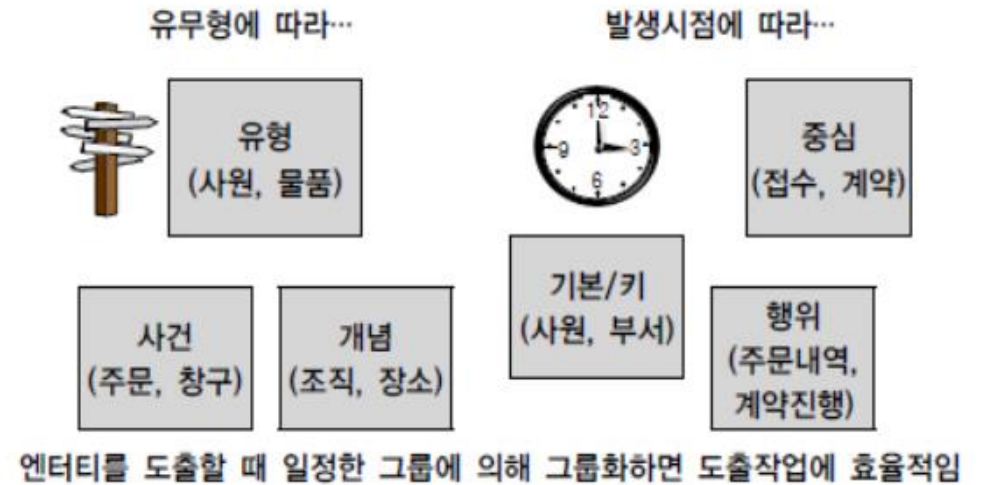
## ✓ 엔티티의 분류

### ■ 유무형의 따른 분류

- **유형** : 물리적인 형태. 안정적, 지속적임. (Ex. 상품, 회원)
- **개념** : 개념적 정보, 물리적인 형태가 없음. (Ex. 조직, 보험 상품)
- **사건** : 업무를 수행함에 따라 발생하는 것. (Ex. 주문, 청구 등)

### ■ 발생 시점에 따른 분류

- **기본** : 업무에 원래 존재하는 정보, 관계에 의해 생성되지 않고 독자적으로 생성함.  
타 엔티티의 부모역할, 고유한 주식별자를 보유. (Ex. 상품, 사원 등)
- **중심** : 기본 엔티티로부터 발생, 업무에 있어서 중심 역할을 수행함.  
다른 엔티티와의 관계를 통해 많은 행위 엔티티를 생성. (Ex. 주문, 매출 등)
- **행위** : 두 개 이상의 부모 엔티티로부터 발생함  
자주 내용이 바뀌거나 증가. (Ex. 주문 목록, 응모 이력 등)



# 속성 (Attribute)

## ✓ 속성(Attribute)이란?

- 데이터베이스에서 엔터티 또는 객체의 특성을 나타내는 항목임.
  - 각 속성은 그 자체로 완전한 의미를 가지며, 요구하는 정보를 나타내기 위해 더 이상 의미있게 분리할 수 없는 최소한의 단위임
    - 분리할 수 없는 최소 단위? → '이름' 속성은 그 학생을 식별하는데 필요한 기본적이고 중요한 정보로, '정의석', '홍길동'처럼 구체적인 값을 가질 수 있습니다.  
그러나 이를 분리하게 더 작은 의미로 쪼개어 버린다면 더 이상 이름이라는 속성의 의미를 갖지 못하게 됩니다.
- 속성값(Attribute Value)
  - 특정 속성에 할당된 실제 데이터임. 예를 들어, '고객' 엔터티의 '이름' 속성에는 '홍길동', '김철수' 등의 속성값이 들어갈 수 있음.

## ✓ 속성의 특징

- 엔터티와 마찬가지로 반드시 해당 업무에서 필요하고 관리하고자 하는 정보이어야 함.
- (정규화 이론에 근간하여) 정해진 주식별자(PK)에 함수적 종속성을 가져야 함.
- 하나의 속성에는 **한 개의 값(속성값)** 만을 가짐. 하나의 속성에 여러 개의 값이 있는 다중값일 경우 별도의 엔터티를 이용하여 분리해야 함.

## ✓ 속성 명명(命名)법

- 해당 업무에서 사용하는 이름을 부여해야 함
- 서술식 속성명은 사용하지 않아야 함
- 약어 사용은 가급적 제한해야 함
- 전체 데이터모델에서 유일성을 확보해야 함

# 속성 (Attribute)의 분류

## ✓ (데이터 모델 내에서 어떤 역할을 수행하는지에 따른) 속성의 분류

- 기본 속성 (Basic Attribute)
  - 정의: 업무 분석 과정에서 직접적으로 관찰하거나 수집할 수 있는 속성입니다. 이 속성들은 엔티티에 관련된 가장 기본적이고 중요한 정보를 나타냅니다.
  - 예시: '직원' 엔티티에 대한 기본 속성은 '이름', '주소', '전화번호' 등이 될 수 있습니다.
- 설계 속성 (Designed Attribute)
  - 정의: 데이터 모델을 설계하는 과정에서 식별을 용이하게 하기 위해 인위적으로 도입한 속성입니다. 종종 고유 식별자나 키로 사용됩니다.
  - 예시: '직원' 엔티티에 대한 설계 속성은 '직원ID'가 될 수 있으며, 이는 각 직원을 구별하기 위해 부여된 고유번호입니다.
- 파생 속성 (Derived Attribute)
  - 정의: 다른 속성의 값으로부터 계산되거나 변형되어 얻어지는 속성입니다. 이는 보통 저장되지 않고 필요할 때 계산되어 사용됩니다.
  - 예시: '직원' 엔티티가 '생년월일'을 속성으로 가지고 있다면, '나이'는 파생 속성이 될 수 있습니다. '나이'는 현재 날짜와 '생년월일'을 기준으로 계산됩니다.

## ✓ (엔티티 구성 방식에 따른) 속성의 분류

- 기본 키 (Primary Key, PK) 속성
  - 정의: 엔티티 내의 각 인스턴스를 유일하게 식별할 수 있는 속성입니다. 기본 키는 중복될 수 없고, Null 값을 가질 수 없습니다.
  - 예시: '사용자' 엔티티의 경우, 각 사용자를 식별할 수 있는 '사용자ID'가 기본 키가 될 수 있습니다. 이 '사용자ID'는 모든 사용자에게 대해 고유해야 합니다.
- 외래 키 (Foreign Key, FK) 속성
  - 정의: 다른 엔티티의 기본 키를 참조하여 두 엔티티 간의 관계를 설정하는 속성입니다. 외래 키는 다른 엔티티의 기본 키와 동일한 값을 가질 수 있습니다.
  - 예시: '주문' 엔티티가 '사용자' 엔티티와 관계를 맺었을 때, '주문' 엔티티 내의 'ID'는 외래 키가 됩니다. 'ID'는 '사용자' 엔티티의 기본 키를 참조하여, 식별하는 데 사용됩니다.
- 일반 속성 (Non-Key Attribute)
  - 정의: 기본 키나 외래 키에 해당하지 않으면서 엔티티 정보를 제공하는 속성입니다. 일반 속성은 엔티티의 특성을 나타내며, 엔티티에 대한 추가적인 정보를 포함합니다.
  - 예시: '직원' 엔티티의 경우, '이름', '번호'와 같은 속성들이 일반 속성에 해당합니다. 이는 직원의 고유 정보를 나타내지만, 식별 또는 관계 설정 시에 사용하지 않습니다.

# 관계 (Relationship)

## ✓ 관계란?

- 상호 연관성. A엔티티와 B엔티티 사이의 논리적인 연관성으로서 존재의 형태, 행위로서 서로에게 연관성이 부여된 상태.
- 관계 페어링의 집합을 논리적으로 표현한 것.

## ✓ 관계의 분류

- 어떤 목적(존재, 행위)으로 연결되었는지에 따라 관계를 분류할 수 있음:
  - 존재에 의한 관계 (존재관계): 사원은 부서에 항상 속해 있다 (존재)
  - 행위에 의한 관계 (행위관계): 회원이 제품을 장바구니에 추가했다 (행위)

## ✓ 관계의 표기법 (엔티티간의 관계를 표기하기 위해 작성하는 항목)

- 관계 명(Membership): 관계의 이름, 관계에 참여하는 형태 지칭.
- 관계 차수(Cardinality): 1:1, 1:M, M:N
- 관계 선택사양(Optionality): 필수관계, 선택관계





감사합니다