DDL

관계형 데이터베이스

- 관계형 데이터베이스(RDBMS – Relation Data Base Management System)
  - ✓ 데이터베이스를 테이블의 집합으로 설명하는 데이터베이스 관리 시스템
  - ✓ 1970년 Codd에 의해 개발
  - ✓ 상용 관계형 데이터베이스 관리 시스템으로는 Oracle, IBM의 DB2, 마이크로소프트의 MS-SQL Server, Sybase를 포함한 SAP의 HANA DB, TeraData Database, Tibero 등이 있음
  - ✓ 오픈 소스 DBMS는 MySQL, PostgreSQL, SQLite 등이 있으며 MySQL이 Oracle에 인수된 이후 가장 많이 사용되는 MySQL의 포크는 MariaDB

관계형 데이터베이스

- 관계형 데이터베이스를 구성하는 용어
  - ✓ 릴레이션(Relation): 정보 저장의 기본 형태가 2차원 구조의 테이블
  - ✓ 속성(attribute): 테이블의 각 열을 의미하는 것으로 Column, Field 라고도 함
  - ✓ 도메인(Domain): 속성이 가질 수 있는 값들의 집합
  - ✓ 튜플(Tuple): 테이블이 한 행을 구성하는 속성들의 집합으로 Record, Row 라고도 함
  - ✓ 카디널리티(Cardinality): 서로 다른 테이블 사이에 대응되는 수

관계형 데이터베이스

- ❖ 관계형 데이터베이스를 구성하는 용어
  - ✓ 릴레이션(Relation)- 테이블 이라고도 부르며 NoSQL에서는 Collection이라고 함
    - 정의
      - ◆ 릴레이션은 행(row)과 열(column)로 구성되는 2차원 구조
      - ◆ 행은 하나의 개체를 나타내고 열은 개체의 속성 의미
      - ◆ 관계형 모델에서는 행은 튜플(tuple), 열은 속성(attribute)
    - 특징
      - ◆ 하나의 릴레이션에 있는 튜플들은 모두 상이(distinct)
      - ◆ 하나의 릴레이션에서 튜플들의 순서와 속성들의 순서는 없음
      - ◆ 하나의 릴레이션에서 같은 이름을 가진 속성들이 있을 수 없음
      - ◆ 각 속성이 가질 수 있는 값들의 범위(도메인, domain)를 벗어나는 값을 가진 튜플들이 존재할 수 없음
      - ◆ 행과 열이 교차되는 곳은 원자값(atomic value)으로만 표현 – 원자값은 논리적으로 더 이상 쪼개질 수 없는 값으로서 여러 개의 값을 갖는 속성을 직접 표현하는 것이 불가능함을 의미

관계형 데이터베이스

- ❖ 관계형 데이터베이스를 구성하는 용어
  - ✓ 키(key)
    - 릴레이션을 구성하는 각 튜플들을 데이터 값들에 의해 유일하게 식별할 수 있는 속성 또는 속성의 집합
      - ◆ 학번
      - ◆ {학번, 이름}, {학번, 학년}, {학번, 이름, 학과}
    - 후보키(Candidate Key)
      - ◆ 릴레이션의 한 속성 집합(K)이 릴레이션이 전체 속성 집합 A의 부분 집합이면서 유일성(uniqueness)과 최소성(minimality)을 만족하는 속성 또는 속성의 집합(K)
      - ◆ 학생 테이블에서의 학번

학생

학번	이름	학년	학과
100	홍길동	4	컴퓨터
200	이하늘	3	전기
300	임꺽정	1	컴퓨터
400	송호준	4	컴퓨터
500	박현진	2	음악

관계형 데이터베이스

- 관계형 데이터베이스를 구성하는 용어
  - ✓ 키(key)
    - 기본키(Primary Key)
      - 후보키 중에서 선정한 키로 릴레이션의 튜플들을 유일하게 식별할 수 있는 키
      - 하나의 테이블에 하나의 기본키만 설정 가능
    - 대체키(Alternate Key)
      - 기본키를 제외한 나머지 후보키(Candidate Key)
    - 외래키(Foreign Key)
      - 다른 테이블의 행을 식별할 수 있는 속성으로 다른 테이블에서는 기본키 이거나 유일한 값을 갖는 속성이어야 함
      - 설정 방법
        - 1:1 관계: 양쪽의 기본키를 서로 다른 테이블의 외래키로 추가
        - 1:M 관계: 1쪽의 기본키를 M쪽의 외래키로 추가
        - N:M 관계: 양쪽의 기본키를 가지는 별도의 테이블을 생성해서 외래키로 설정

관계형 데이터베이스

- 릴레이션의 연결: 공통된 의미를 갖는 속성을 공유하여 관계형 데이터베이스 내의 릴레이션 들을 연결 – 외래키를 설정

수강과목

학번	이름	학점	학과	과목번호
100	홍길동	4	컴퓨터	C123
200	이하늘	3	전기	C234
300	임꺽정	1	컴퓨터	C236
400	송호준	4	컴퓨터	C156
500	박현진	2	음악	C234

외래 키

과목

과목번호	과목이름	학점	학과	과목이름
C123	컴퓨터개론	3	컴퓨터	임사랑
C234	프로그램밍	3	전기	정파란
C236	자료구조	3	컴퓨터	최하늘
C156	파일처리	3	컴퓨터	최정상
C198	데이터베이스	3	컴퓨터	김홍두

관계형 데이터베이스

- ❖ 도메인(Domain)과 속성(Attribute)
  - ✓ 속성 값(Attribute Value)
    - ● 개개의 속성이 가지는 데이터 값
    - ● 관계형 모델에서 이러한 데이터 값들은 더 이상 분해할 수 없는 원자 값만 허용
  - ✓ 도메인 (Domain)
    - ● 하나의 속성이 가질 수 있는 값 들의 집합

관계형 데이터베이스

- ❖ 무결성 제약 조건
  - ✓ 개체 무결성(Entity Inetgrity)
    - 기본키는 null이거나 중복될 수 없음
  - ✓ 참조 무결성(Referential Intigrity)
    - 릴레이션은 참조할 수 없는 외래키의 값을 가져서는 안됨
    - 다른 테이블에 존재하는 값이거나 null 이어야 함
  - ✓ 도메인 무결성
    - 속성은 도메인에 설정된 값만을 가져야 함

모델링

- 데이터 모델
  - ✓ 물리적 모델
    - 데이터가 어떻게 컴퓨터에 저장되는 지에 관심
    - 레코드의 형식, 레코드 순서, 접근 경로 와 같은 정보
  - ✓ 개념적 모델
    - 사용자가 데이터를 어떻게 인식하는 지에 관심
    - 사용자에게 시스템이 데이터를 인식하는 것은 중요하지 않음
    - E-R 모델이 대표적인 모델
  - ✓ 논리적 모델
    - 컴퓨터와 사용자 둘 다를 고려하는 데이터 모델
    - 사용자가 이해할 수 있게 하면서 실제로 저장 될 수 있는 구조
    - 계층, 네트워크, 관계, 객체지향 데이터 모델들이 여기에 속함

모델링

- ❖ 데이터베이스 추상화 와 데이터 독립성
  - ✓ 추상화
    - 자동차를 운전하면서 운전자는 운전 방법만 알면 되지 엔진이 어떻게 작동하는지, 피스톤이나 크랭크축의 움직임과 같이 자세하고 세부적인 내용은 알 필요가 없는데 이처럼 알 필요가 없는 복잡한 부분을 은폐하는 것이 추상화(Abstraction)
    - 데이터베이스에서 최종 사용자는 자신이 원하는 일부분의 데이터에만 관심이 있기 때문에 데이터베이스의 복잡한 내부 구조는 은폐하는 추상화가 필요
    - 데이터베이스 관리 시스템의 주된 역할은 사용자에게 데이터에 대한 추상적인 뷰(View)를 제공하는 것으로 데이터가 어떻게 저장되고 유지되는가에 대한 정보는 숨겨도 무방

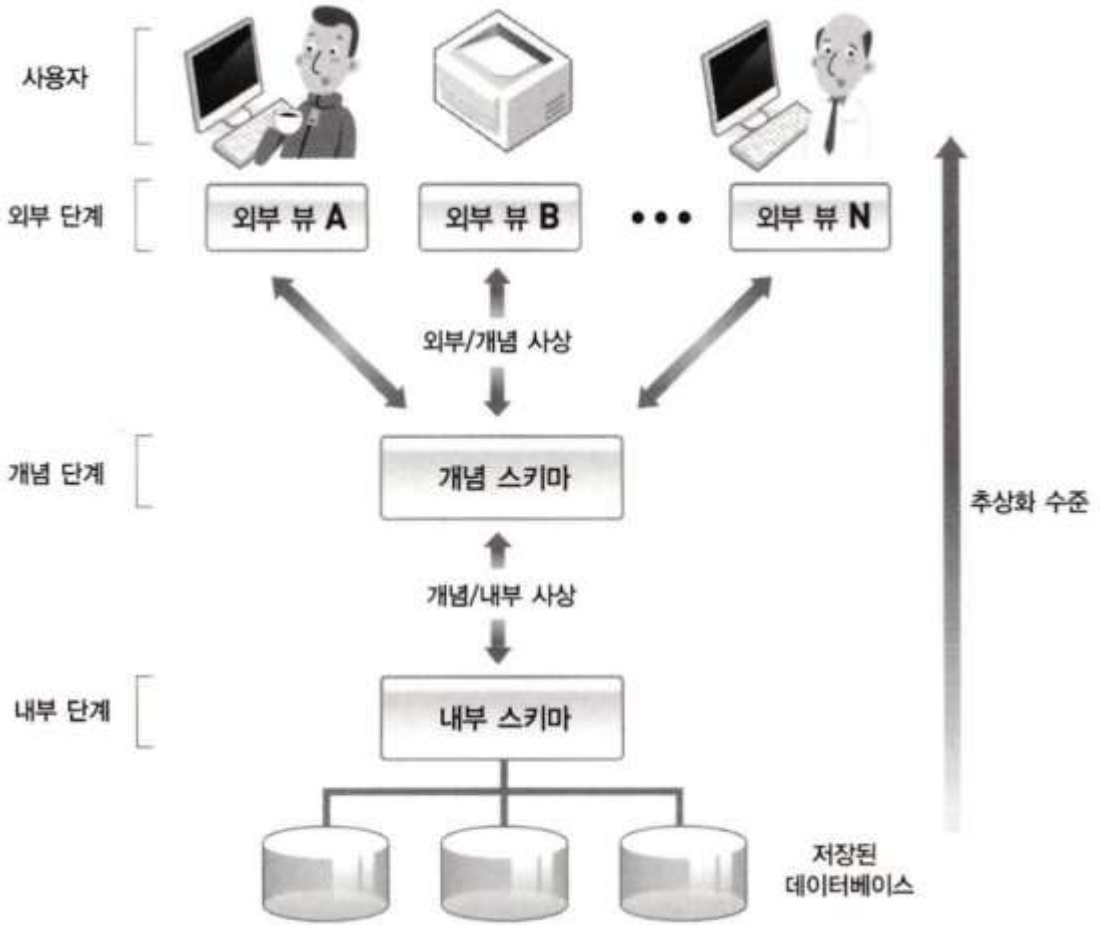
모델링

- ❖ 데이터베이스 추상화 와 데이터 독립성
  - ✓ 3계층 스키마: 사용자의 응용 프로그램과 물리적 데이터베이스를 분리시키는 것이 목적
    - Internal Schema(내부 스키마): 데이터베이스의 물리적인 저장 구조를 기술하고 물리적 데이터 모델을 사용하며 데이터베이스에 대한 접근 경로를 기술
    - Conceptual Schema(개념 스키마)
      - ◆ 모든 단계의 사용자를 위해서 전체 데이터베이스의 구조를 기술
      - ◆ 단순히 스키마라고도 하며 물리적인 저장 구조의 세부 사항은 생략하고 엔티티, 관계, 제약조건, 무결성 규칙 등을 기술하는 데 중점을 두는데 ER 다이어그램 같은 고수준 데이터 모델 또는 구현 데이터 모델로 표현
    - External Schema(외부 스키마)
      - ◆ 사용자나 응용 프로그램 개발자가 관심있어 하는 데이터베이스 부분을 기술
      - ◆ 사용자에게는 이 부분만 노출하고 나머지 부분은 감추어 짐

모델링

❖ 데이터베이스 추상화 와 데이터 독립성

✓ 3계층 스키마: 사용자의 응용 프로그램과 물리적 데이터베이스를 분리시키는 것이 목적



모델링

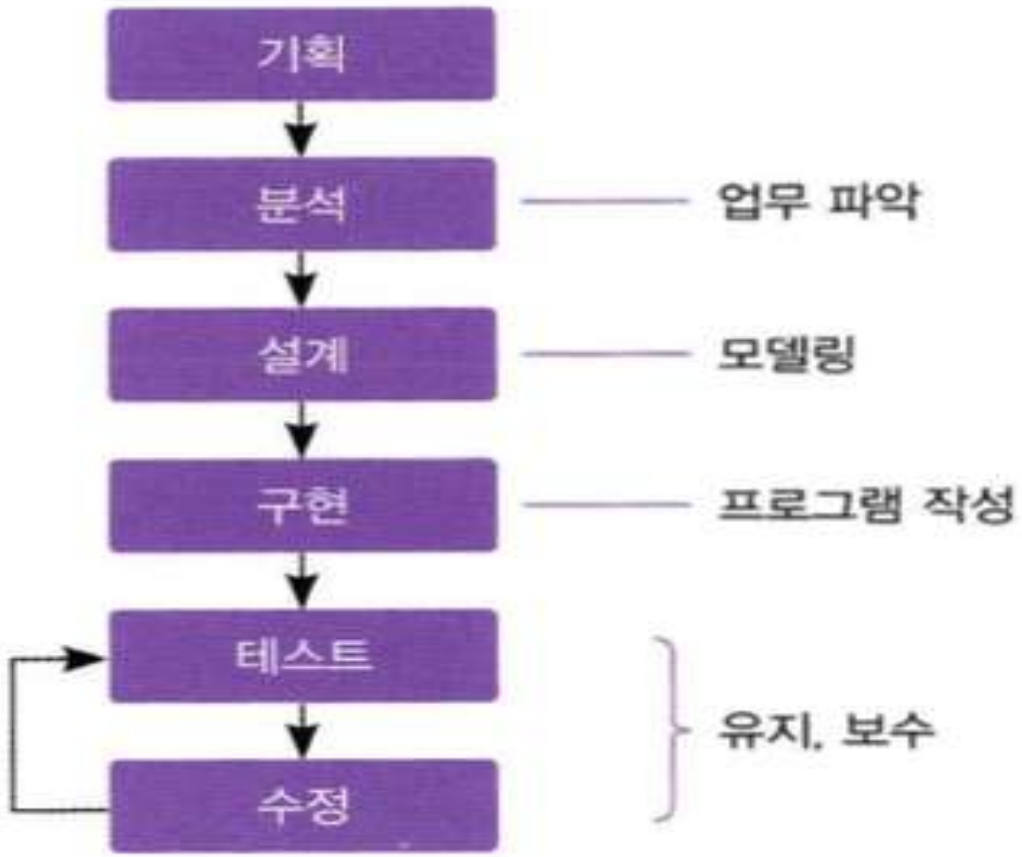
❖ 데이터베이스 추상화 와 데이터 독립성

✓ 데이터 독립성

- 특정 스키마를 변경할 때 상위 단계의 스키마에 영향을 미치지 않는 성질
- 고수준의 데이터 독립성을 논리적 데이터 독립성이라고 하고 저 수준의 데이터 독립성은 물리적 데이터 독립성이라고 함
- 데이터베이스의 구조를 변경하더라도 사용자의 사용 방법이나 응용 프로그램의 구조를 변경할 필요가 없도록 하는 것
- 논리적 데이터 독립성은 외부 스키마나 응용 프로그램을 변경하지 않고 개념 스키마를 변경할 수 있는 능력으로 외부 스키마에 영향을 미치지 않으면서 개념 스키마에 적용되는 제약조건을 변경할 수 있도록 해서 논리적 구조가 변경되어도 응용 프로그램은 영향을 받지 않게 하는 것으로 데이터베이스의 데이터 타입이나 데이터 항목을 추가하거나 삭제할 때 개념 스키마가 변경됨
- 물리적 데이터 독립성은 개념 스키마를 변경하지 않고 내부 스키마를 변경할 수 있는 능력으로 내부 스키마가 변경되어도 외부 스키마와 개념 스키마는 영향을 받지 않는 것으로 검색 성능을 향상시키려고 접근 구조를 추가하거나 파일들이 재 구성될 때 내부 스키마가 변경됨

모델링

❖ 데이터베이스 생명주기



모델링

❖ 데이터베이스 생명주기

- ✓ 기획: 어떤 프로그램을 누가 어떻게 작성할지 계획을 세우며 요구 기능, 예산, 개발 기간을 결정하는 과정으로 정치적인 면이 많아 개발 과정에 속한다고 보기는 애매하며 개발자가 기획에 참여하는 경우는 별로 없음
- ✓ 요구 사항 수집 및 분석: 요구 사항 수집 및 분석의 단계에서는 사용자들의 요구 사항을 듣고 분석하여 데이터베이스 구축의 범위를 정하는 과정
- ✓ 설계: 분석된 요구사항을 기초로 주요 개념과 업무 프로세스 등을 식별하고(개념적 설계) 사용하는 DBMS의 종류에 맞게 변환(논리적 설계)한 후 데이터베이스 스키마를 도출(물리적 설계)하는데 설계의 단계에서 개념적 모델링을 하여 ER 다이어그램을 도출하고 이를 이용하여 관계 스키마 모델을 도출하고 이를 물리적 모델링하여 관계 스키마를 도출

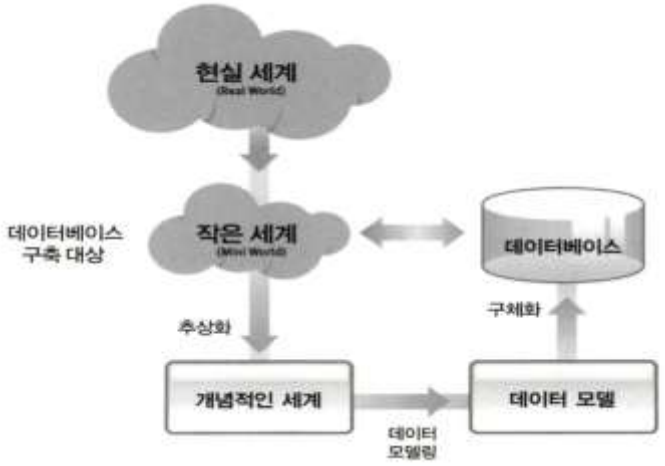
- ✓ 구현: 설계 단계에서 생성한 스키마를 실제 DBMS에 적용하여 테이블 및 관련 객체(뷰 or 인덱스)를 생성
- ✓ 운영: 구현된 데이터베이스를 기반으로 소프트웨어를 구축하여 서비스를 제공
- ✓ 감시 및 개선: 데이터베이스 자체의 문제점을 파악하여 개선

모델링

❖ 모델링의 이해

✓ 모델링의 정의: 복잡한 현실 세계를 일정한 표기법에 의해 표현하는 것

- Webster 사전
  - ◆ 가설적 일정 양식에 맞춘 표현
  - ◆ 어떤 것에 대한 예비 표현으로 그로부터 최종 대상이 구축되도록 하는 계획으로서 기여하는 것
- 복잡한 현실 세계 를 단순화시켜 표현하는 것
- 모델이란 사물 또는 사건에 관한 양상(Aspect)이나 관점(Perspective)을 연관된 사람이나 그룹을 위하여 명확하게 하는 것
- 모델이란 현실 세계의 추상화된 반영



모델링

- ❖ 모델링의 이해
  - ✓ 모델링의 3가지 관점
    - ● 데이터 관점
      - ◆ 업무가 어떤 데이터와 관련이 있는지?
      - ◆ 데이터간의 관계는 무엇인지?
      - ◆ 비지니스 프로세스에서 사용하는 데이터 – Data
      - ◆ What
      - ◆ 정적 분석과 구조 분석
    - ● 프로세스 관점
      - ◆ 업무(비지니스 프로세스에서 수행하는 작업)가 실제하고 있는 일이 무엇인지?
      - ◆ 무엇을 모델링해야 하는지?
      - ◆ How
      - ◆ 시나리오 분석, 도메인 분석, 동적 분석
    - ● 데이터 와 프로세스의 상관 관점
      - ◆ 업무가 처리하는 일의 방법에 따라 데이터는 어떻게 영향을 받고 있는지?
      - ◆ Interaction(상호작용)
      - ◆ CRUD

모델링

❖ 모델링의 이해

✓ 모델링의 중요성

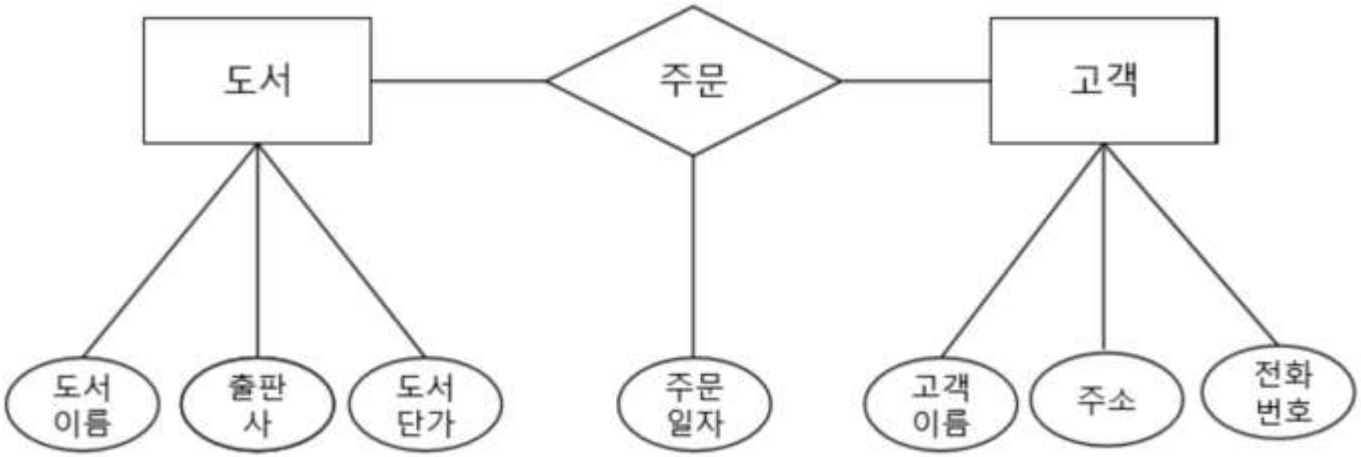
- 건설 현장에서 볼 수 있는 조감도는 완공 후의 모습을 하늘에서 새가 보듯이 한눈에 알아볼 수 있게 해주는데 조감도처럼 데이터 모델링은 최종적으로 구축할 데이터베이스 시스템 전체에 대해서 약속된 표기법으로 표현하게 해줌
- 데이터 모델링은 시스템을 설계 및 구축하는 데 사용되는 것은 물론이고 데이터베이스 시스템을 구축하지 않더라도 해당 업무 내용을 분석하고 이해하는 데도 유용하게 사용할 수 있게 해주며 전체 시스템의 내용을 약속된 표기법으로 그리고 쉽게 알아보도록 표현하였기에 데이터베이스 관리자(DBA)부터 최종 사용자까지 다양한 계층의 데이터베이스 사용자 간에 의사소통을 쉽게 해 줌
- 데이터베이스가 설계되고 생성되는 과정에서 상세한 내용보다는 전체적인 개념을 파악해서 작업을 할 수 있으므로 전체적인 조화와 균형을 유지할 수 있게 해주는데 엔티티들이 추출될 때 적절하게 동일한 수준으로 추출되었는지 중복되어 설계하지는 않았는지 빠뜨린 내용은 없는지 일관성은 있는지 등을 쉽게 파악할 수 있게 해줌
- 데이터베이스의 구조에 따라 응용 프로그램의 설계가 달라질 수도 있고 결함을 발견하지 못한 채로 시스템이 운영된다면 다음에 데이터베이스 자체의 구조는 물론 연결된 응용 프로그램까지 전부 수정해야 하는 상황이 발생할 수 있는데 이러한 상황을 방지하고자 데이터와 응용 프로그램이 서로 독립성을 유지하고 계층으로 구조화되어 있지만 데이터베이스의 구조 변경이 응용 프로그램에게 100% 영향을 미치지 않는다고는 할 수 없으므로 초기 데이터베이스 모델링은 체계적이고 정교하게 이루어져야 함

모델링

- ER Diagram(Model)
  - ✓ ER Diagram 이란 Entity-Relationship Model을 표현하는 것으로 현실 세계의 요구 사항(Requirements)들로 부터 Database를 설계하는 과정에서 활용
  - ✓ 개념을 모델링하는 것으로 개체(entity)와 속성(attribute), 관계성(relationship)을 표현

모델링

- ER Diagram(Model)
  - ✓ Peter Cheng's Notation



모델링

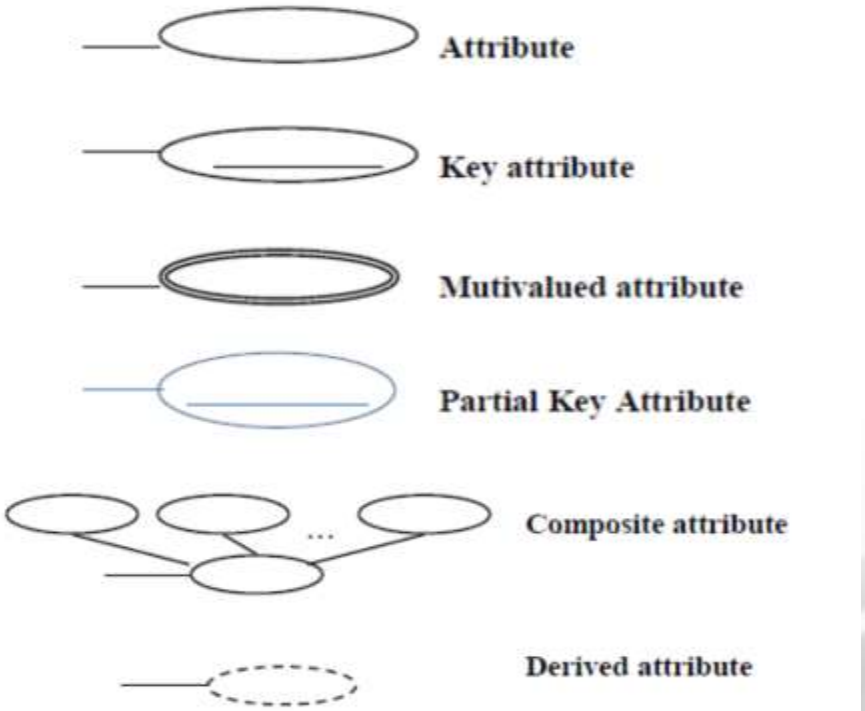
- ER Diagram(Model)
  - ✓ Peter Cheng's Notation
    - Entities

(Regular, Strong) entity type

Weak entity type

모델링

- ER Diagram(Model)
  - ✓ Peter Cheng's Notation
    - Attributes



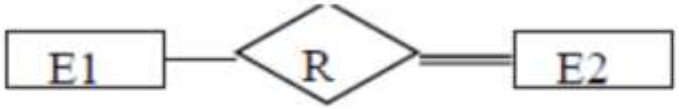
모델링

- ER Diagram(Model)
  - ✓ Peter Cheng's Notation
    - Relation

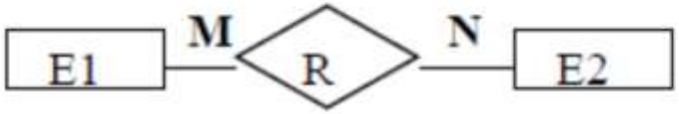
기호	의미
	• 비식별자 관계(non-identifying relationship) : 강한 개체 타입 • 부모 개체의 키가 일반 속성으로 포함되는 관계
	• 식별자 관계(identifying relationship) : 약한 개체 타입 • 부모 개체의 키가 주식별자로 포함되는 관계
	• 일대다(1:N)의 관계 : N 쪽에 새발을 표시
	• 0(선택 참여), 최소 참여가 0일 경우
	• 1(필수 참여), 최소 참여가 1일 경우

모델링

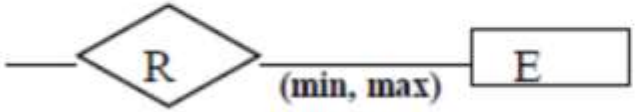
- ER Diagram(Model)
  - ✓ Peter Cheng's Notation



Partial participation, Total participation



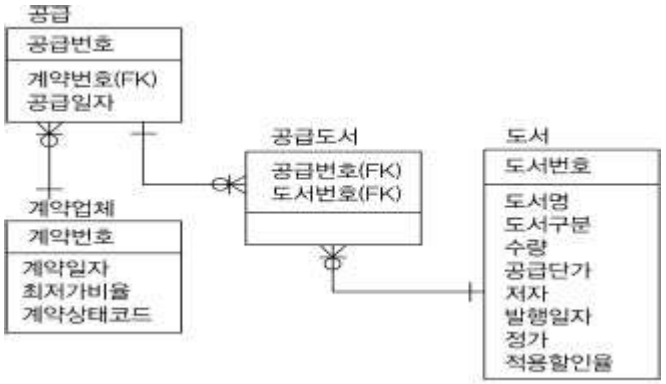
Cardinality

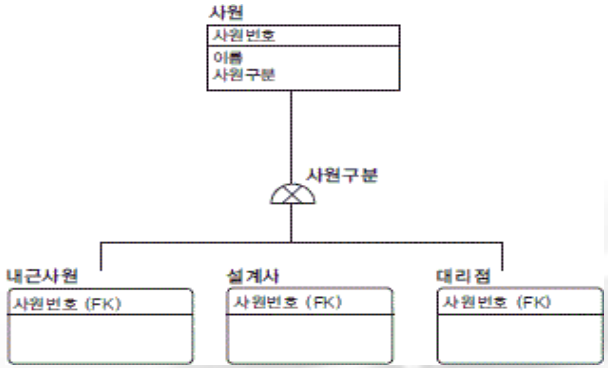


Structural constraint (min, max) on Participation of E in R

모델링

- ER Diagram
  - ✓ 정보 공학(IE – 까마귀 발)표기법
    - 제임스 마틴(James Martin)이 주창한 Information Engineering에서 사용하는 Data 모델 표기법
    - 까마귀발 모양과 같은 형태의 관계 표현 때문에 Crow's Foot Model로 불리기도 함
    - 가장 널리 사용되는 표기법의 하나이지만 서브 타입과 같은 개체 집합의 상세 표현에 있어서 공간을 많이 차지하는 단점이 있음





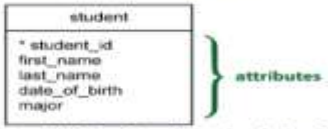
모델링

- ER Diagram
  - ✓ 정보 공학(IE – 까마귀 발)표기법

Entities

An entity is represented by a rectangle, with its name on the top. The name is singular (entity) rather than plural (entities).

Attributes – describe the characteristics of a particular entity instance. The attribute(s) that uniquely distinguishes an instance of the entity is the identifier. Usually, this type of attribute is marked with an asterisk.

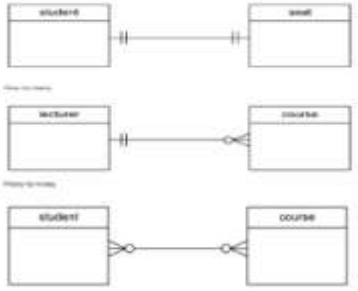


Relationships

They are presented as a straight line. Usually, each relationship has a name, expressed as a verb, written on the relationship line.

Cardinality of the Relationship in ERD of Crow's Foot Notation

Symbol	Meaning
	Mandatory & One (One and Only One)
	Optional & One (Zero or one)
	Mandatory & Many (One or many)
	Optional & Many (Zero or many)

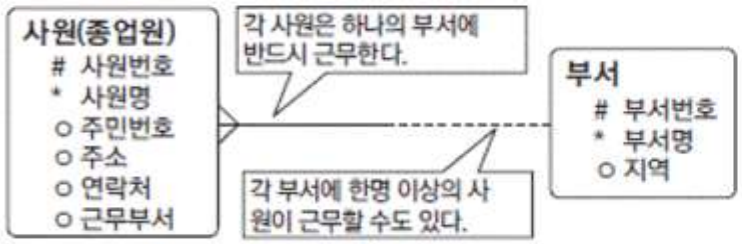


min
max

Zero
One
Many

모델링

- ER Diagram
  - ✓ Barker's Notation
    - 1986년, 영국 컨설팅 회사 CACI에 근무하던 Richard Barker, Lan Palmer, Harry Ellis 등에 의해 처음 개발
    - Original Chen style 에서 사용되던 Crow's Foot style을 변형하여 개발
    - 오라클로 이직한 리차드 바커에 의해 오라클에서 Case Method로 채택하여 사용되고 Oracle CASE modeling tools에 적용
    - 오늘날 널리 사용되는 표기법의 하나로 개체 집합과 그 관계의 표현에 있어서 직관적 이해성이 뛰어나고 구체적이고 상세한 표현성 및 공간 활용의 장점 등으로 인해 사용자가 지속적으로 증가



모델링

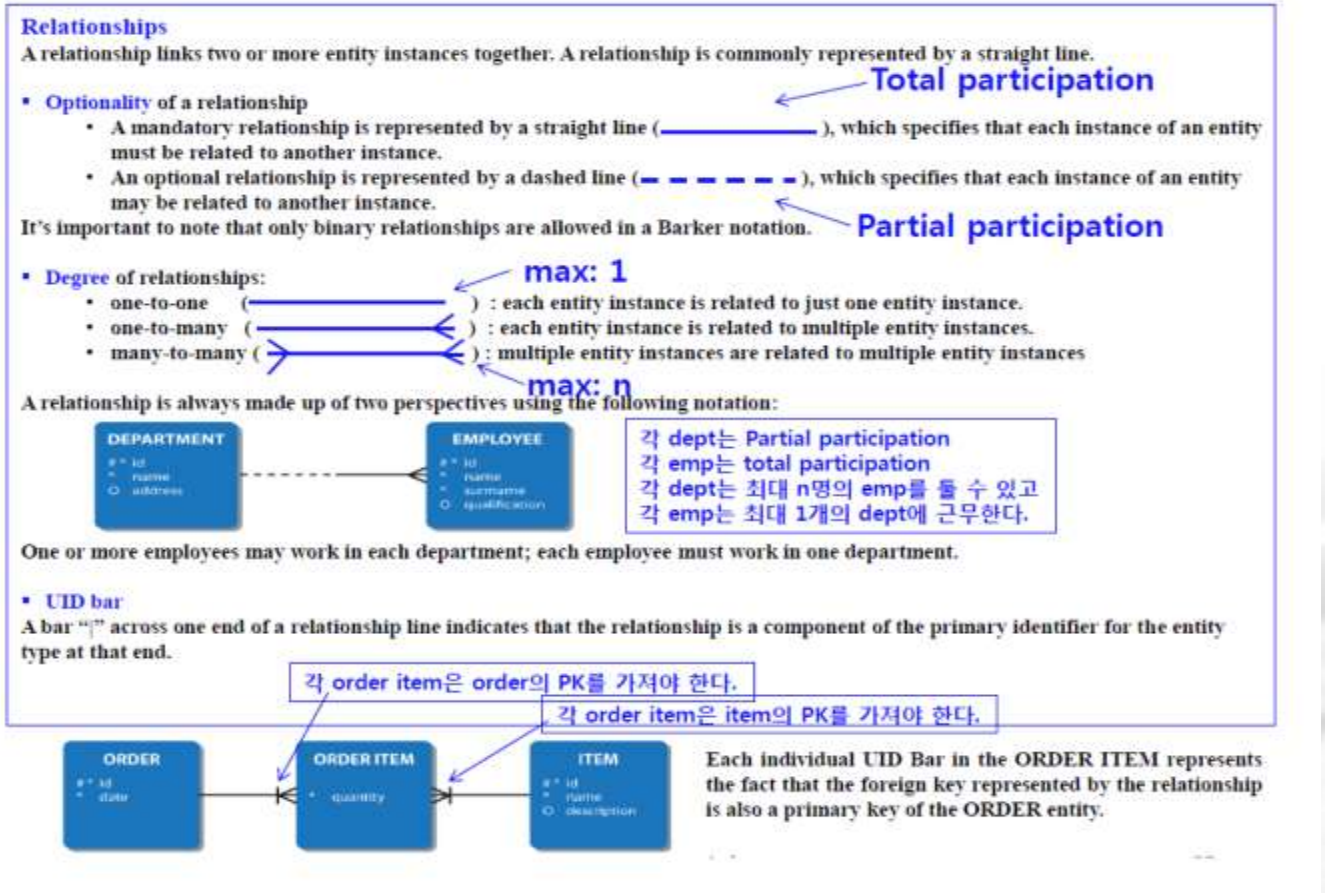
- ER Diagram
  - ✓ Barker's Notation

Attributes – describe the characteristics of a particular entity instance. An attribute can be of three types:

- **Unique Identifier – uniquely identifies an entity instance**
- **Mandatory – its value cannot be null**
- **Optional – its value can be null**

모델링

- ER Diagram
  - ✓ Barker's Notation



모델링

- ER Diagram
  - ✓ ERD 작업 순서
    - Entity를 도출하고 그림
    - Entity를 적절하게 배치
    - Entity간 관계를 설정: 연결
    - 관계 이름을 기술
    - 관계의 참여도를 기술
    - 관계의 필수 여부를 기술
  - ✓ Entity 배치
    - 좌에서 우로, 위에서 아래로
    - 가장 중요한 Entity(고객, 주문)을 좌측 상단에 배치
    - 업무 흐름의 중심이 되는 Entity(출고, 주문 목록, 출고 목록)를 중앙에 배치
    - 중심 Entity와 관계 있는 Entity(창고, 고객, 사원, 재고)를 주위에 배치

모델링

- ER Diagram
  - ✓ ERD 관계의 연결
    - 서로 관련있는 Entity 간의 관계를 설정
    - 초기에는 모두 PK로 속성이 상속되는 식별자 관계를 설정
    - 중복 관계, Cycle 관계 등을 유의
  - ✓ ERD 관계 이름의 표시
    - 관계 이름은 현재형을 사용
    - 지나치게 포괄적인 용어(이다, 가진다 등)는 사용하지 않도록
    - 실무에서는 생략해도 무방 – 관계 이름이 없어도 ERD의 흐름을 알 수 있음
  - ✓ ERD 관계 관계 차수와 선택성 표시

관계	선택성	IE 표기법	Barker 표기법
1 : 1	필수		
1 : 1	선택		
1 : n	필수		
1 : n	선택		

모델링

- Entity
  - ✓ 실체, 객체 – 정보 시스템의 구조 분석 결과
  - ✓ Entity
    - 변별할 수 있는 사물 – Peter Chen (1976)
    - 데이터베이스 내에서 변별 가능한 객체 – C.J Date (1986)
    - 정보를 저장할 수 있는 어떤 것 – James Martin (1989)
    - 정보가 저장될 수 있는 사람, 장소, 물건, 사건 그리고 개념 등 – Thomas Bruce (1992)
    - Entity는 데이터의 집합
    - Entity는 사람, 장소, 물건, 사건, 개념 등의 명사에 해당
    - Entity는 업무 상 관리가 필요한 관심사에 해당
    - Entity는 저장이 되기 위한 어떤 것
  - ✓ Instance
    - Entity(객체)가 실제 구현된 예가 Instance(사례, 경우)
    - Entity는 Instance의 집합

모델링

- ❖ Entity
  - ✓ 특징
    - 업무에서 필요로 하는 정보: 반드시 해당 업무에서 필요하고 관리하고자 하는 정보

병원시스템	Entity	인사시스템
O	환자	X
X	토익점수	O

    - 식별이 가능해야 함: 유일한 식별자에 의해 식별이 가능해야 함
    - Instance 의 집합
      - ◆ 영속적으로 존재하는 Instance의 집합
      - ◆ 2개 이상의 Instance의 집합
      - ◆ 1개의 Instance로 이루어진 집합은 Entity가 아님
    - 업무 프로세스에 의해 이용: 업무 프로세스가 반드시 그 Entity를 이용
    - 속성을 포함
      - ◆ Entity 에는 반드시 속성(Attributes)이 포함되어야 함
      - ◆ 식별자만 존재하고 일반 속성이 전혀 없는 객체는 Entity가 될 수 없음
      - ◆ 단 관계 Entity의 경우엔 주 식별자 속성만으로도 Entity로 인정

모델링

- ❖ Entity
  - ✓ 특징
    - ● 관계의 존재
      - ◆ Entity는 다른 Entity와 최소 한 개 이상의 관계가 존재하여야 함
      - ◆ 데이터 모델링에서 관계를 생략하여 표현하는 경우
        - ▪ 시스템 처리시 내부적으로 필요한 Entity: 로그 테이블
        - ▪ 통계를 위한 데이터: 통계 만을 위한 Read Only Table
        - ▪ 코드성 Entity
          - • 너무 많은 Entity 들과의 관계로 데이터 모델이 복잡해 짐
          - • 일반적으로 코드 테이블에 FK를 설정하지 않는 경우가 대부분

모델링

- Entity
  - ✓ 명명
    - 가능하면 현업 업무에서 사용하는 용어를 사용
    - 가능하면 약어를 사용하지 않음
    - 가능하면 단수 명사를 사용
    - 유일하게 이름을 부여
    - Entity 생성 의미대로 이름을 부여

모델링

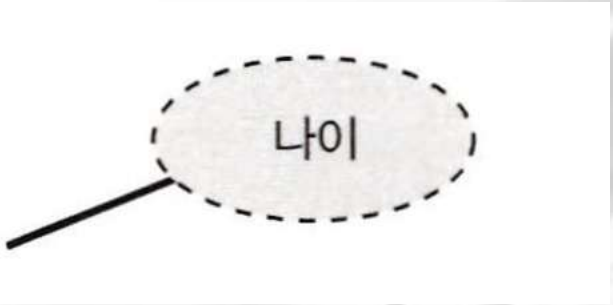
- ❖ 속성(Attribute)
  - ✓ 사전적 의미
    - 사물의 성질, 특징, 또는 본질적인 성질, 그것이 없다면 실체를 생각할 수 없는 것
  - ✓ 데이터 모델링 관점에서 속성(Attribute)
    - 업무에서 필요로 하는 인스턴스로 관리하고자 하는 의미상 더이상 분리되지 않는 최소의 데이터 단위
    - 업무 상 관리하기 위한 최소의 의미 단위
    - Entity가 가지는 항목으로 속성은 Entity를 설명
    - 속성은 인스턴스의 구성 요소
    - 업무에서 관리되는 정보로 하나의 값만 가져야 하고 주식별자(기본키)에 함수적으로 종속됨
  - ✓ 예시
    - 생년월일은 그 자체로 의미가 있으므로 속성이라 할 수 있음
    - 생년, 생월, 생일 로 분리가 가능하지만 이는 하나의 속성을 관리 목적으로 분리한 것일 뿐 각각을 속성이라 할 수는 없음
    - 이름과 주소는 각각 의미있는 속성이지만 이름주소로 묶는다면 하나의 속성이 두가지 의미를 가지므로 기본 속성이라 할 수 없음

모델링

- ❖ 속성(Attribute)
  - ✓ 속성의 분해 여부에 따른 분류
    - 단일 속성: 하나의 의미로 구성된 것으로 아이디나 이름
    - 복합 속성: 여러 개의 의미가 있는 것으로 주소 속성이 있음
    - 다중 값 속성: 여러 개의 값을 가질 수 있는 것으로 리스트 형태 들인데 이 속성은 다른 Entity로 독립시켜야 함

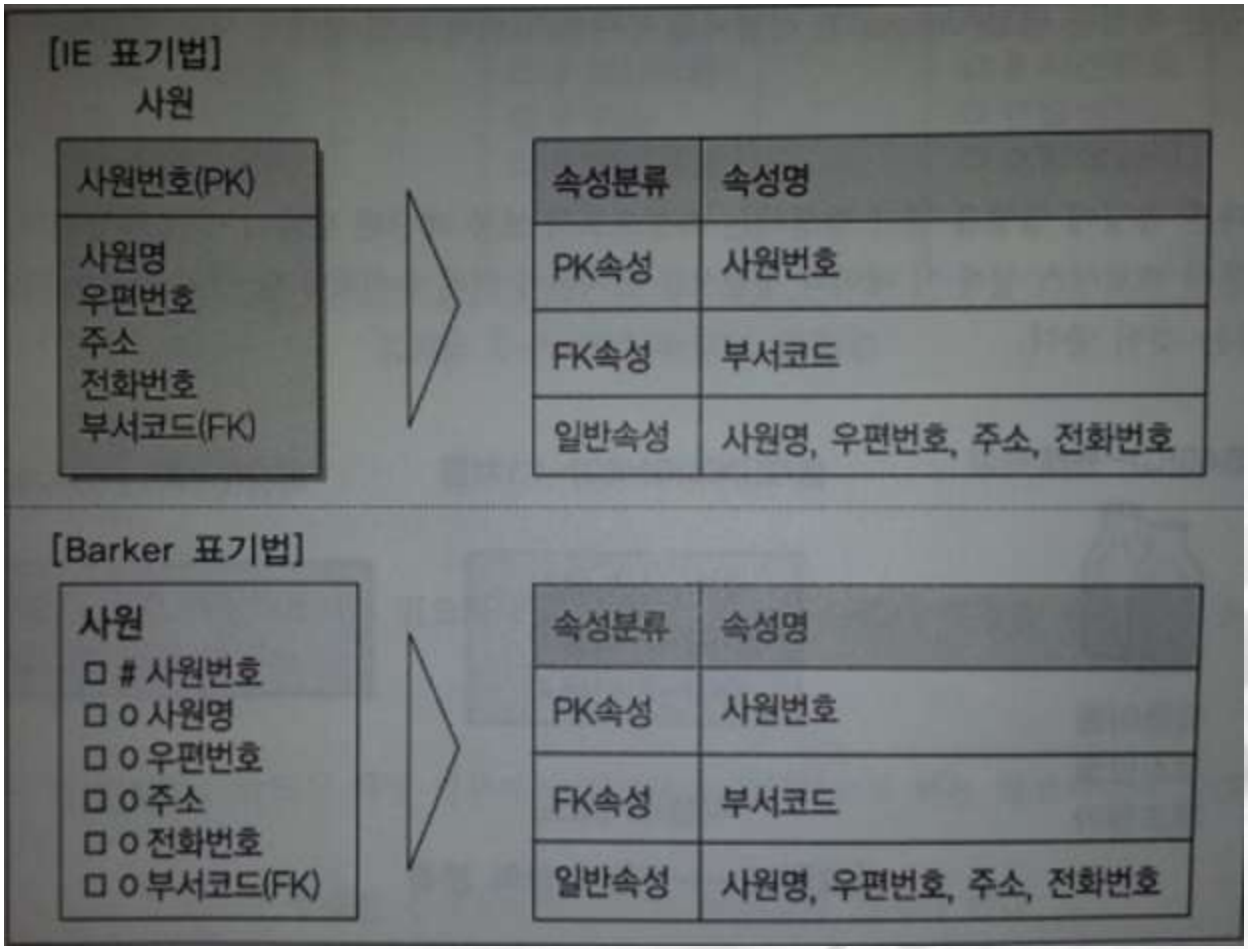
모델링

- ❖ 속성(Attribute)
  - ✓ 속성의 특성에 따른 분류
    - 기본 속성: 업무로부터 추출한 모든 속성
    - 설계 속성: 코드성 데이터, Entity 식별용 일련번호
    - 파생 속성: 다른 속성에 영향을 받아 발생하는(Derived) 속성, 계산된 값, 합계, 재고, 잔액
      - ◆ 파생 속성은 그 속성이 가지고 있는 계산 방법에 대해 반드시 어떤 Entity의 어떤 속성에 의해 영향을 받는지 정의가 되어야 함
      - ◆ 타 속성에 의해 지속적으로 영향을 받아 자신의 값이 변하는 성질을 가지고 있는 속성
      - ◆ 파생 속성은 꼭 필요한 경우에만 정의하여 업무 로직이 속성 내부로 스며들지 못하도록 주의해야 함
      - ◆ 파생 속성을 정의한 경우라면 그 값의 정합성을 유지할 수 있도록 해야 함
      - ◆ 통계 관련 Entity, 배치 작업 수행 관련



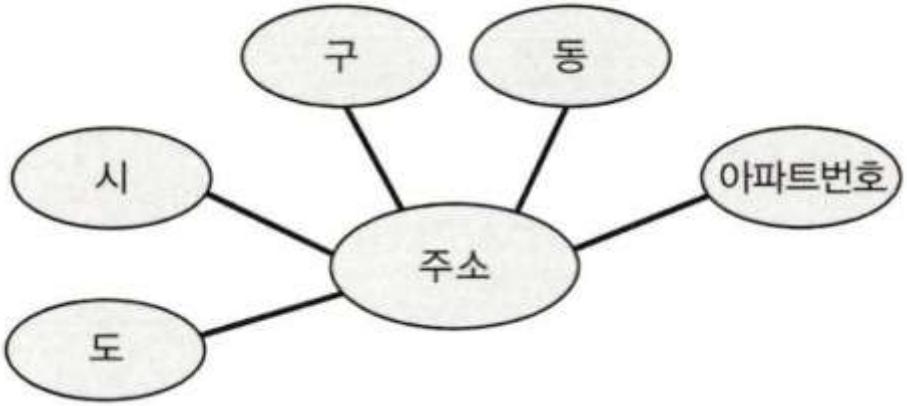
모델링

- 속성(Attribute)
  - ✓ Entity 구성 방식에 따른 분류 – 단순(Simple) 속성 과 키(Key) 속성



모델링

- ❖ 속성(Attribute)
  - ✓ 기타 분류
    - ● 복합(Composite) 속성: 2개 이상 속성으로 구성

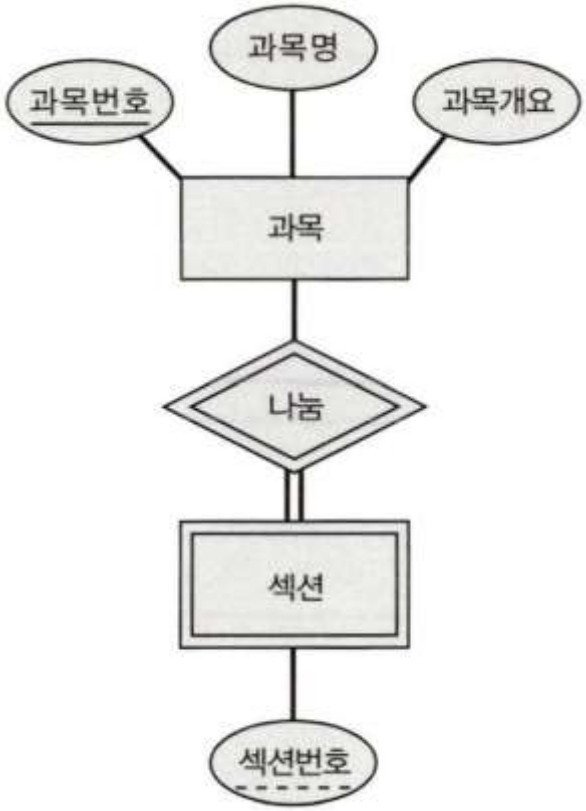


모델링

- ❖ 속성(Attribute)
  - ✓ 기타 분류
    - ● 다치(Multi Value) 속성: 2개 이상의 값을 가질 수 있는 속성

모델링

- ❖ 속성(Attribute)
  - ✓ 기타 분류
    - ● 부분 키(Partial Key) 속성: 다른 속성과 함께 하면 Key가 될 수 있는 속성

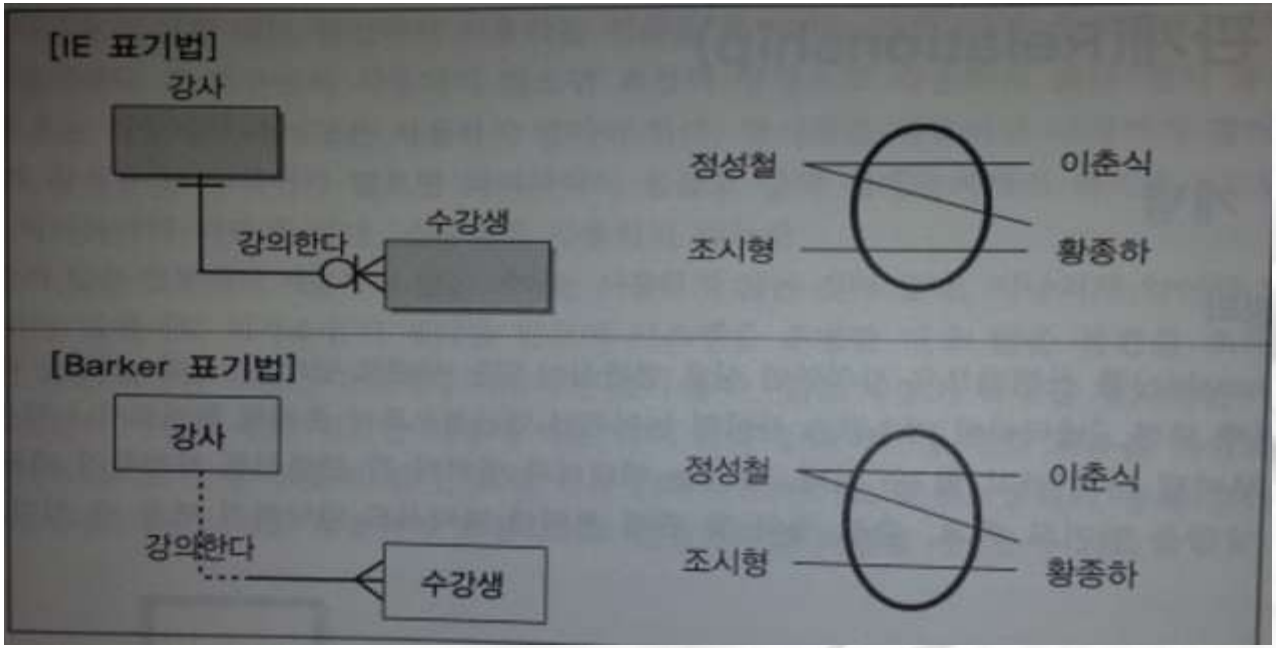


모델링

- ❖ 속성(Attribute)
  - ✓ 도메인
    - ● 속성이 가질 수 있는 값의 범위
      - ◆ 학점은 0.0~4.0 사이의 실수
      - ◆ 주소는 20자리 문자열
  - ✓ 속성 이름 부여 원칙
    - ● 해당 업무에서 사용하는 이름을 부여
    - ● 서술식 속성 명은 사용하지 않음
    - ● 약어 사용은 가급적 제한
    - ● 전체 데이터 모델에서 유일성 확보하는 것이 좋음

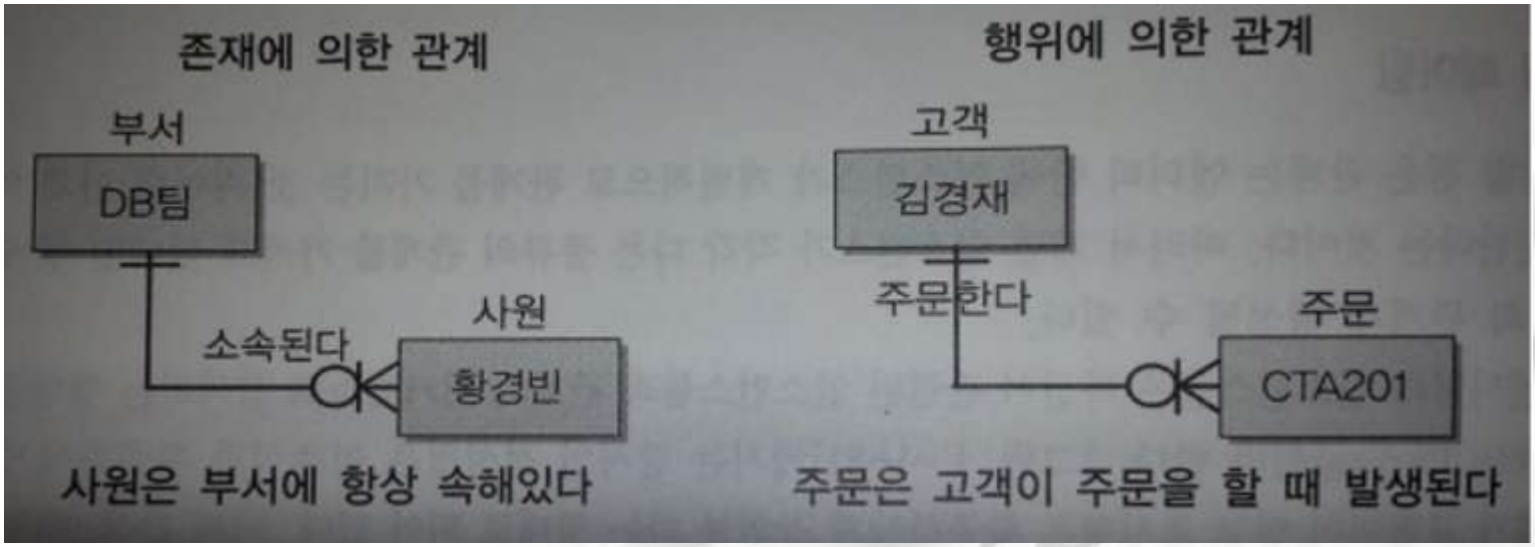
모델링

- 관계
  - ✓ 관계의 정의: 인스턴스 사이의 논리적 연관성으로서 존재 또는 행위로서 서로에게 연관성이 부여된 상태
  - ✓ 관계의 페어링(Relationship Paring)
    - Relationship은 Entity 안의 Instance가 개별적으로 관계를 가지는 것(Paring)이고 이것의 집합을 관계로 표현한다는 것
    - 개별 인스턴스가 각각 다른 종류의 관계를 가지고 있다면 두 Entity 사이에 2개 이상의 관계가 형성 될 수 있음



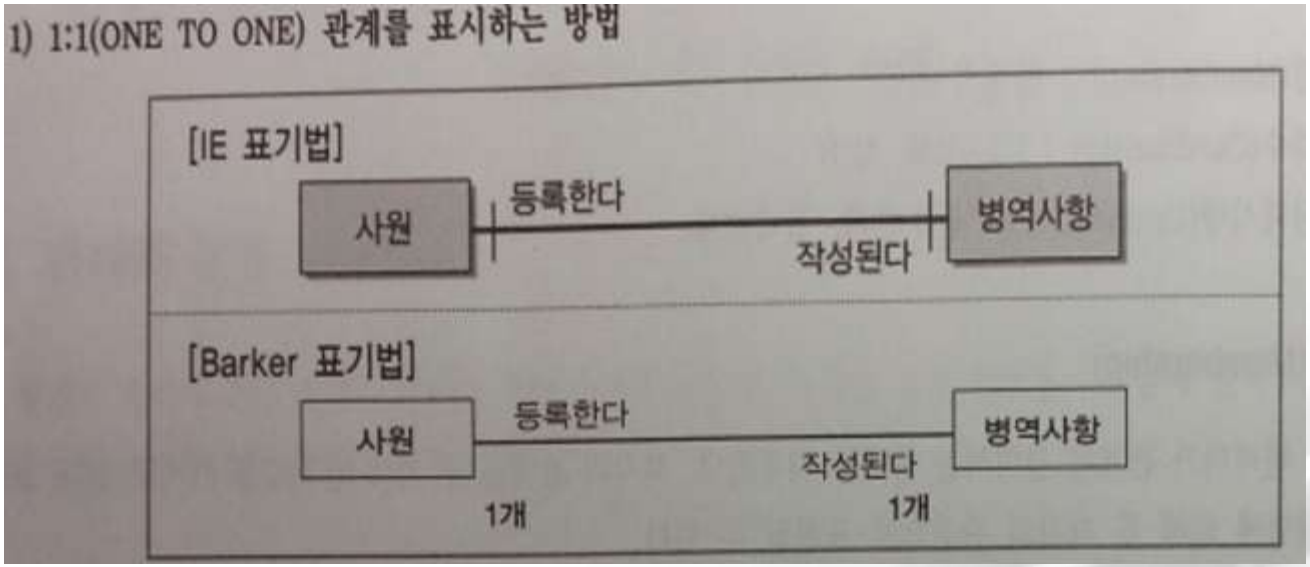
모델링

- ❖ 관계
  - ✓ 분류
    - ● 존재 관계
      - ◆ Entity 간의 상태를 의미
      - ◆ 학과와 학생처럼 학생이 아무런 행위를 하지 않아도 특정 학과에 소속되는 경우
    - ● 행위 관계
      - ◆ Entity 간의 행위가 있는 것
      - ◆ 특정 행위를 해야만 만들어지는 관계



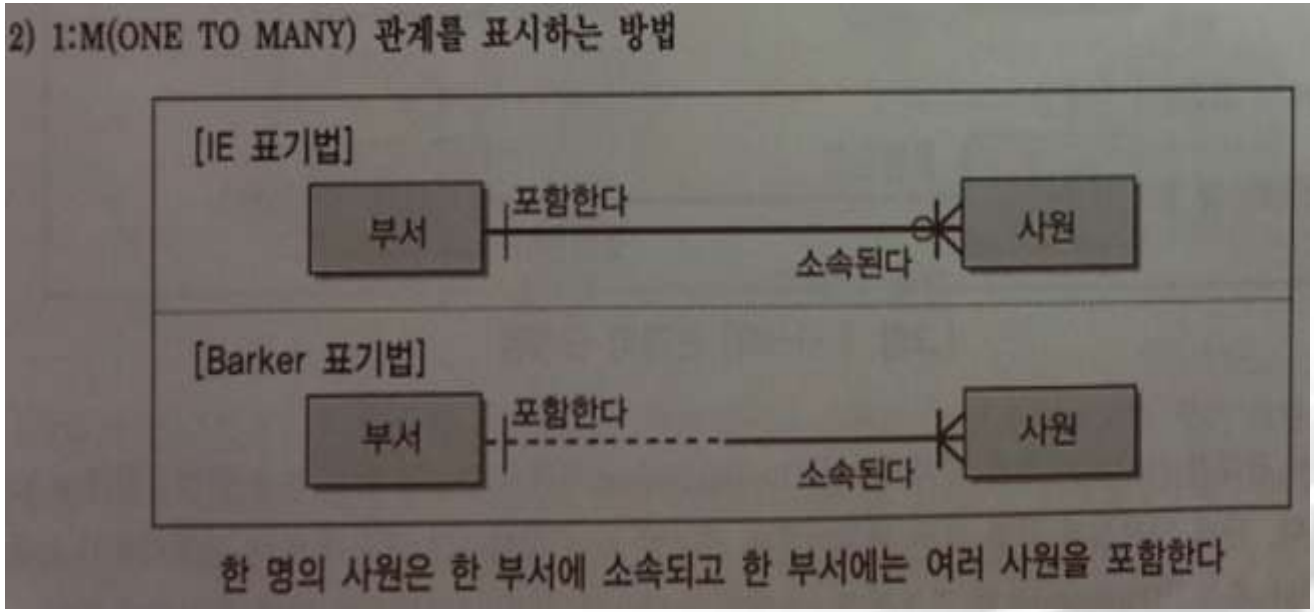
모델링

- ❖ 관계
  - ✓ 관계 표현
    - ● 1:1 관계
      - ◆ 필수적 관계와 선택적 관계로 구분할 수 있음



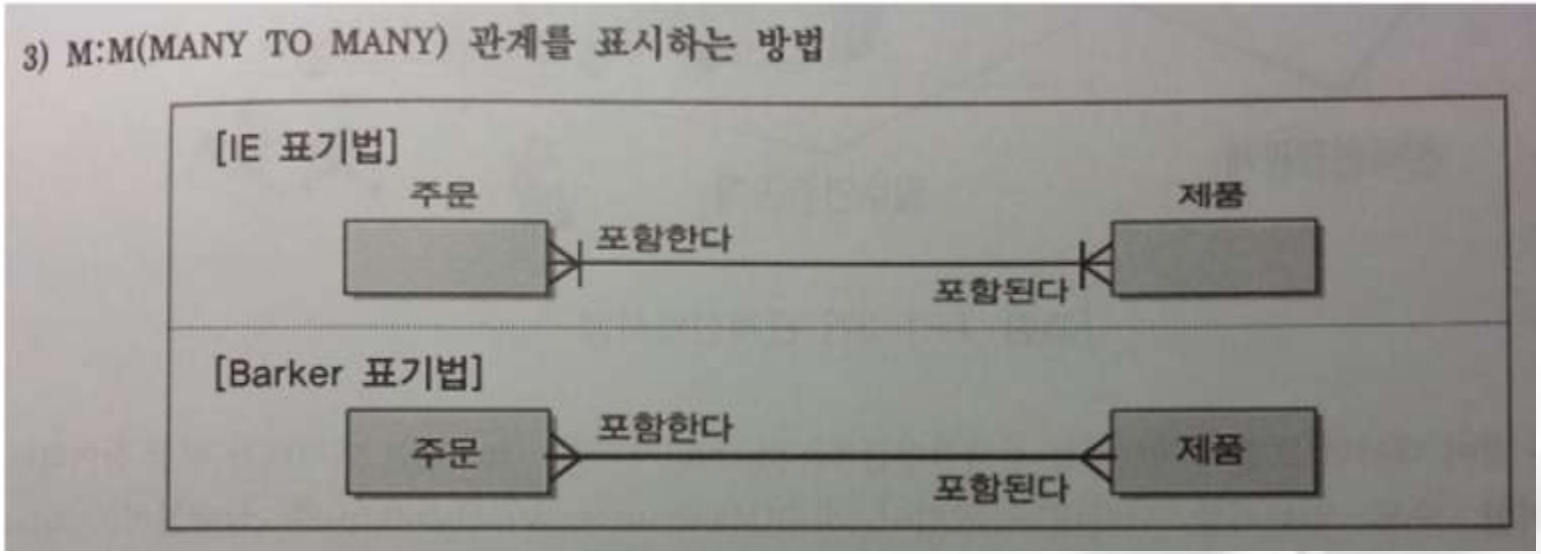
모델링

- ❖ 관계
  - ✓ 관계 표현
    - ● 1:N 관계



모델링

- ❖ 관계
  - ✓ 관계 표현
    - ● M:N 관계

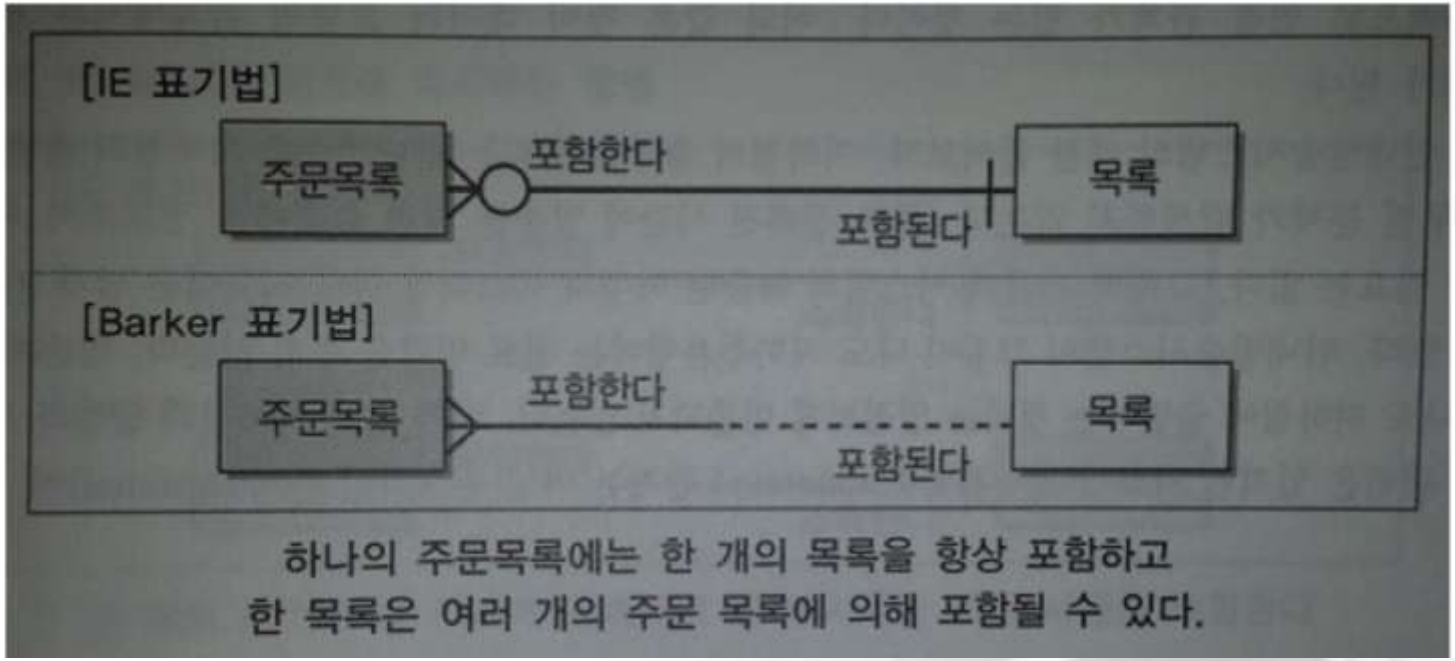


모델링

❖ 관계

✓ 관계 표현

● 관계 선택 사양(Optionality): 필수 관계, 선택 관계

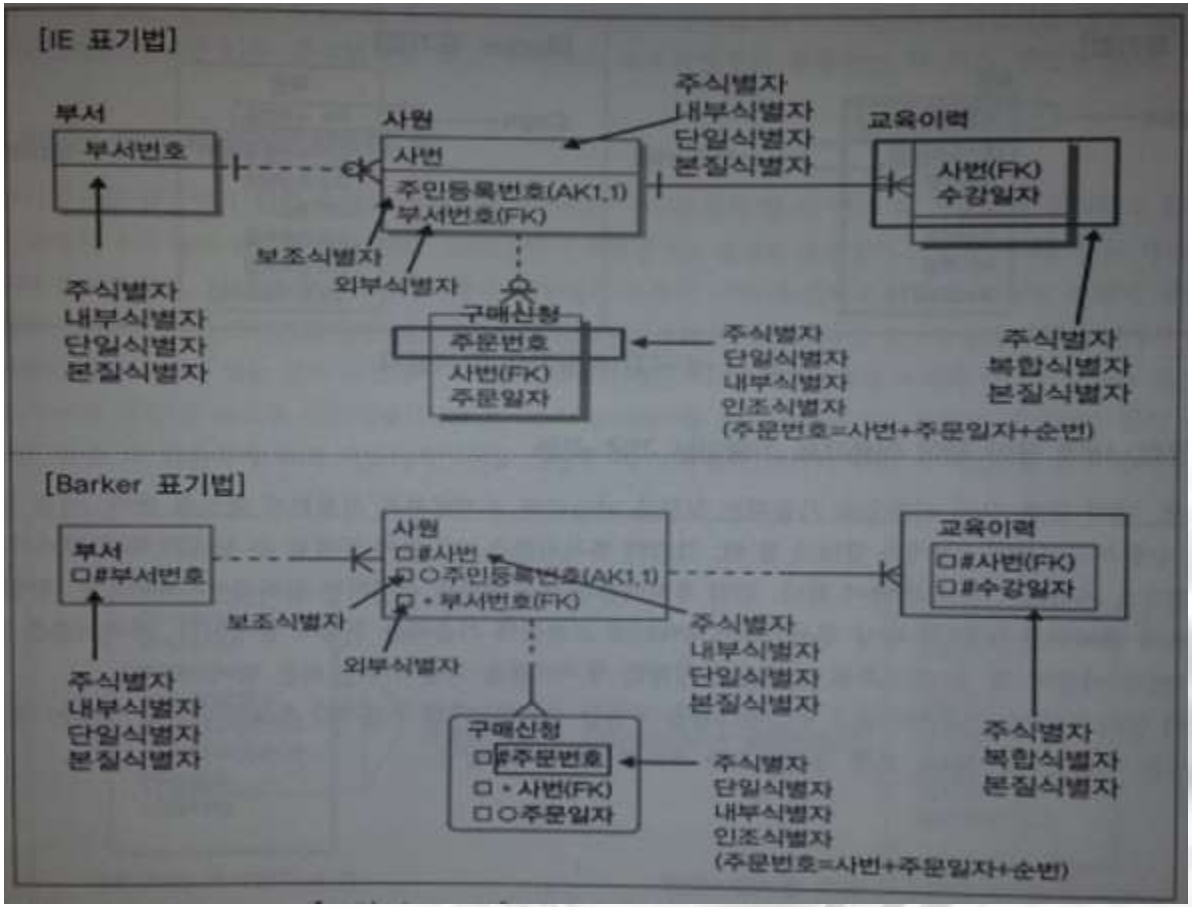


모델링

- Identifier
  - ✓ Entity 내에서 Instance들을 구분할 수 있는 구분자
  - ✓ 식별자의 특징
    - 유일성: 주 식별자에 의해 Entity 내 모든 Instance 들을 유일하게 구분함 – 사원번호를 주식별자로 설정하면 사원 Entity 내의 모든 직원들에 대해 개인별로 사원번호를 부여함
    - 최소성: 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야 함 – 사원 번호만으로 식별할 수 있다면 사원 분류 코드 + 사원 번호 는 주식별자로 부적절함
    - 불변성: 주식별자가 한 번 특정 Entity 에 지정되면 그 값 은 변하지 말아야 함 – 사원 번호의 값이 변한다는 의미는 이 기록이 말소되고 새로운 기록이 발생되는 개념임
    - 존재성: 주식별자가 지정되면 반드시 데이터 값이 존재(Null 안됨) – 사원 번호 없는 회사직원은 있을 수 없음.

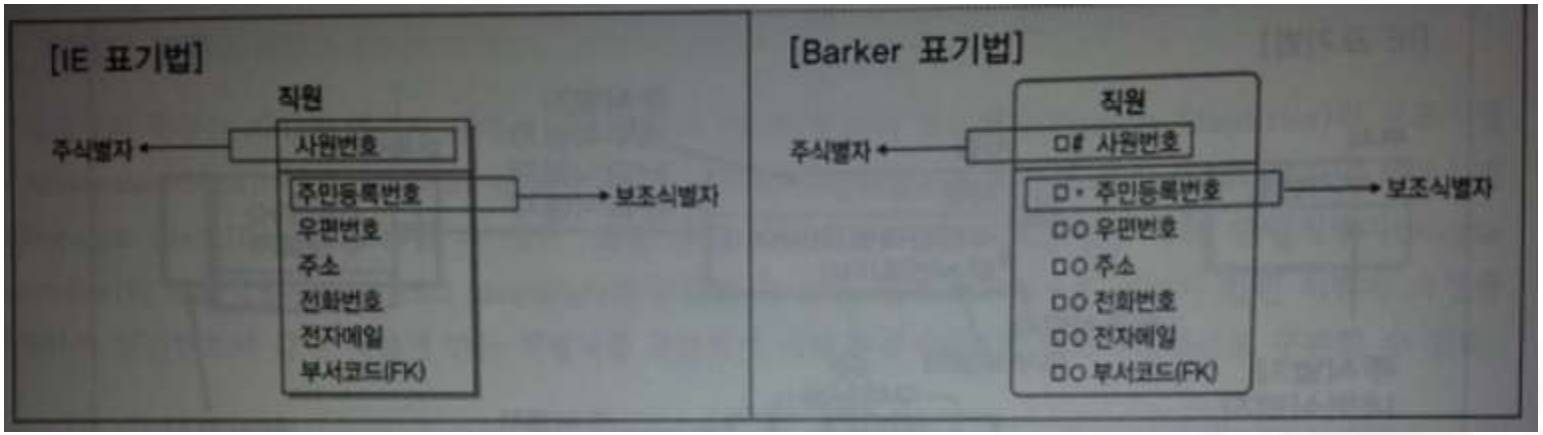
모델링

- Identifier
  - ✓ 식별자 표기법



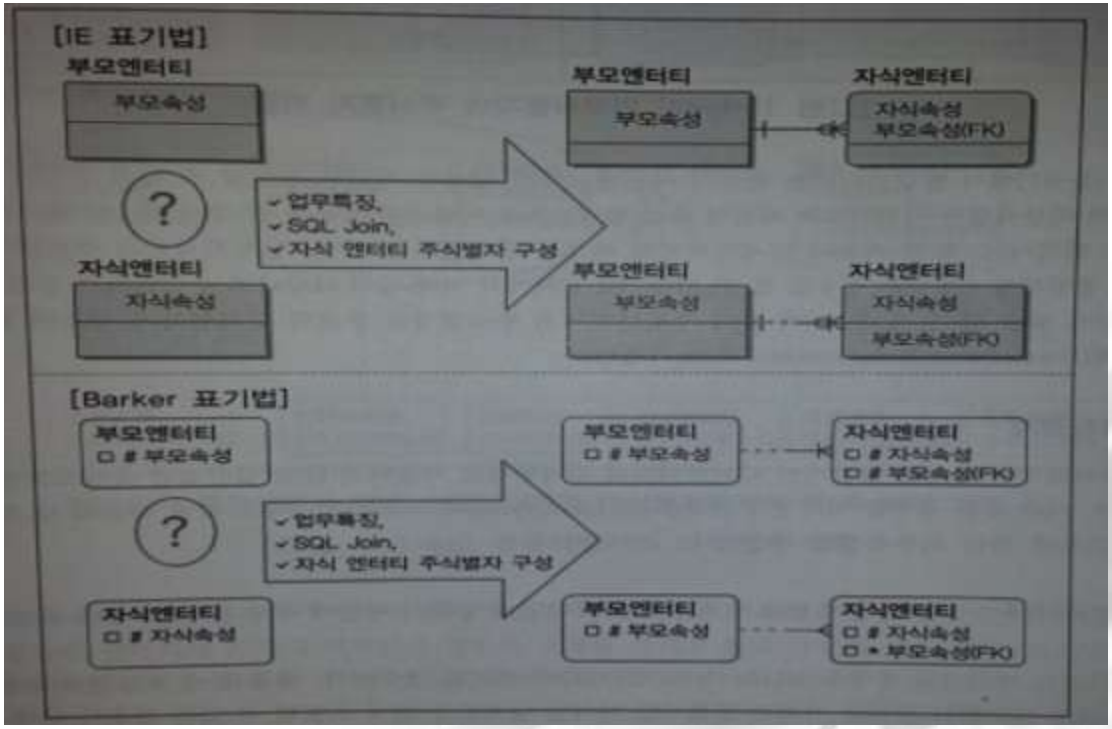
모델링

- ❖ Identifier
  - ✓ 주 식별자 도출 기준
    - ● 해당 업무에서 자주 이용되는 속성을 주식별자로 지정하도록 함



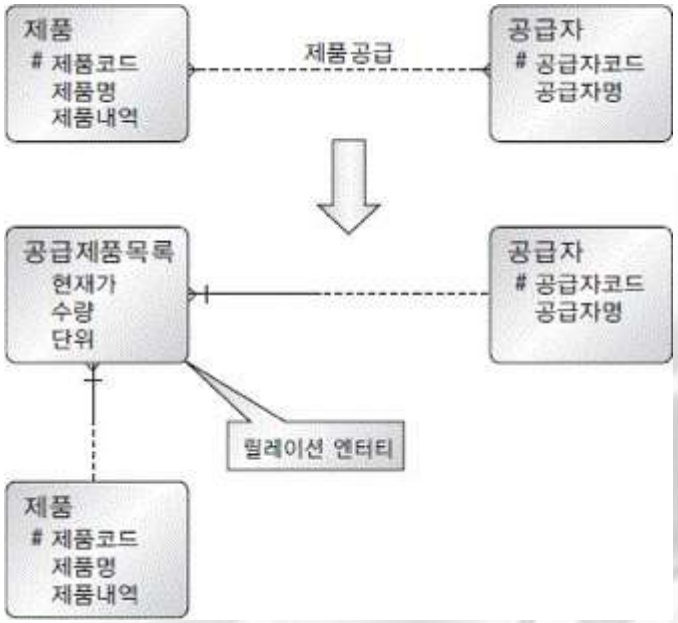
모델링

- ❖ Identifier
  - ✓ 식별자 관계와 비식별자 관계의 결정
    - 부모와 자식간의 관계에 의해 외부 식별자가 생성
    - 부모로부터 받은 외부 식별자를 자신의 주식별자로 이용할 것인지? --> 식별자 관계
    - 부모와 연결이 되는 속성으로만 이용할 것인지? --> 비식별자 관계로 할 것인지 결정



모델링

- M:M 관계
  - ✓ M:M 관계는 논리 데이터 모델링 과정에서 많이 나타나며 최종적으로 완성된 데이터 모델에는 존재할 수 없는 형태라고 할 수 있음
  - ✓ 제품 Entity 와 공급자 Entity 간에는 M:M 관계가 존재하는데 이런 경우는 1:M 관계 2개로 분할해야 함
  - ✓ 이 관계가 해소된 결과로 공급제품목록이라는 새로운 Entity가 만들어지고 또한 각각의 두 Entity 즉 제품, 공급자와 1:M 의 관계를 부모로서 가지게 됨

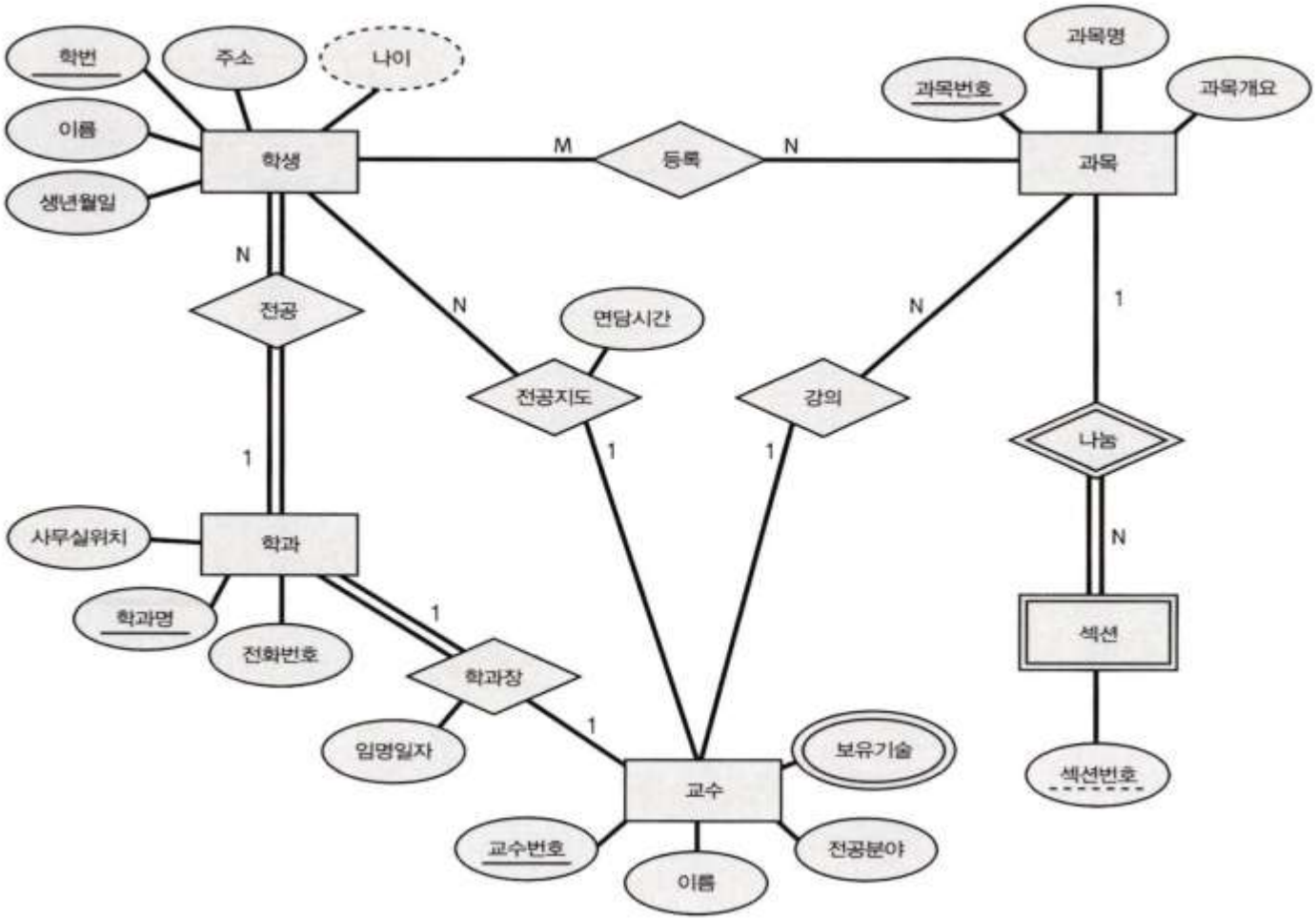


모델링

- E-R Diagram Sample
  - ✓ 분석 내용
    - 모든 학생은 고유한 학번을 갖고 특정 학과에 소속되며 이름, 주소, 생년월일, 나이도 관리
    - 학과는 학과명, 학과 사무실 위치, 전화번호 등을 관리하고 학교 내에서 같은 이름의 학과는 없음
    - 학생은 수강할 과목을 등록하는데 과목에는 과목번호, 과목명, 과목개요 등이 있음
    - 과목은 여러 섹션으로 나누어질 수 있는데 섹션에는 고유한 섹션번호가 있으며 모든 과목이 섹션으로 나누어지는 것이 아니므로 섹션은 과목이 없으면 존재할 필요가 없고 또한 다른 과목의 섹션은 같은 섹션번호를 가질 수 있음
    - 교수는 교수번호로 식별할 수 있고 교수이름, 전공분야, 보유기술 등을 관리하며 교수는 여러 개의 보유기술을 가질 수 있음
    - 교수는 과목을 강의하고 학생에 대해 전공지도를 하는데 일부는 학과의 학과장이 되고 학과마다 학과장은 한 명씩 있음

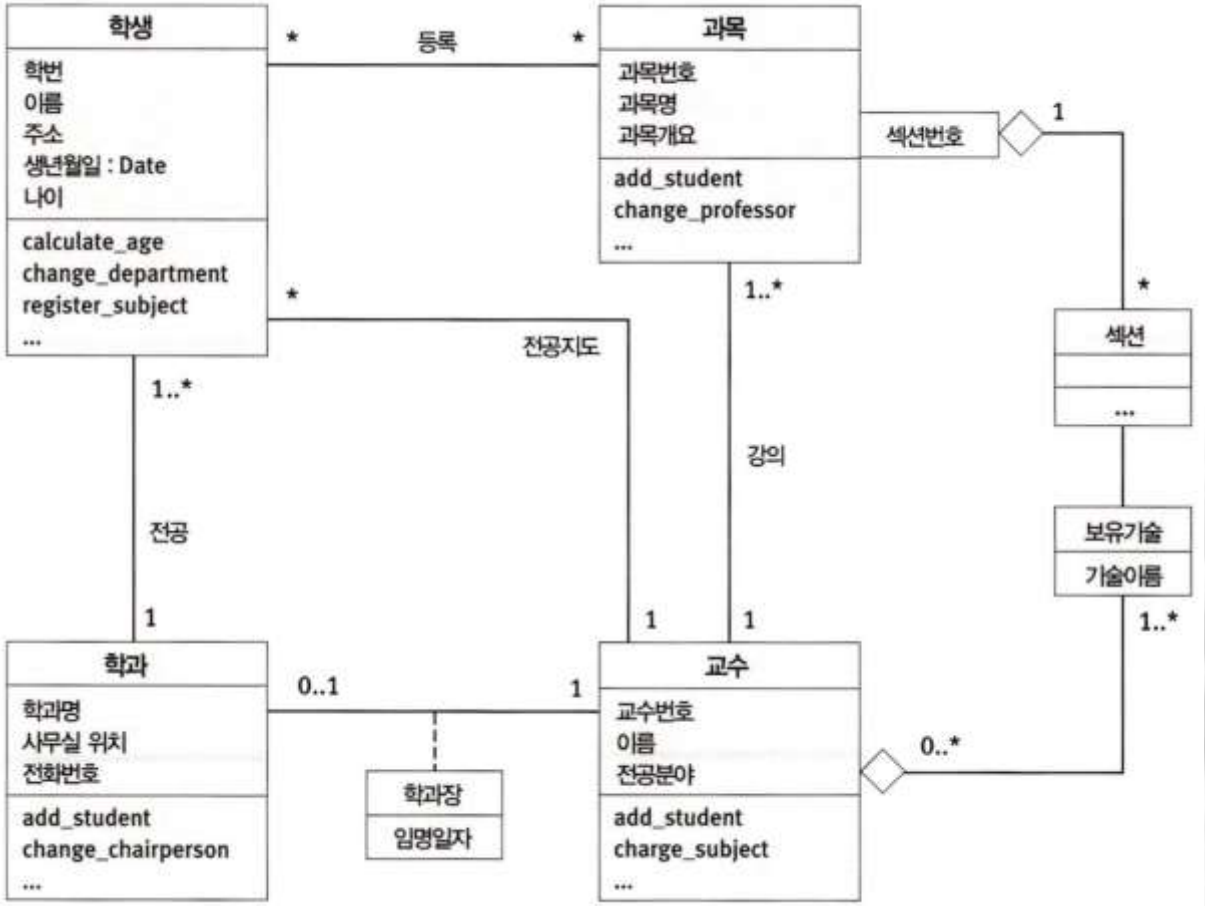
모델링

- E-R Diagram Sample
  - ✓ E-R Diagram



모델링

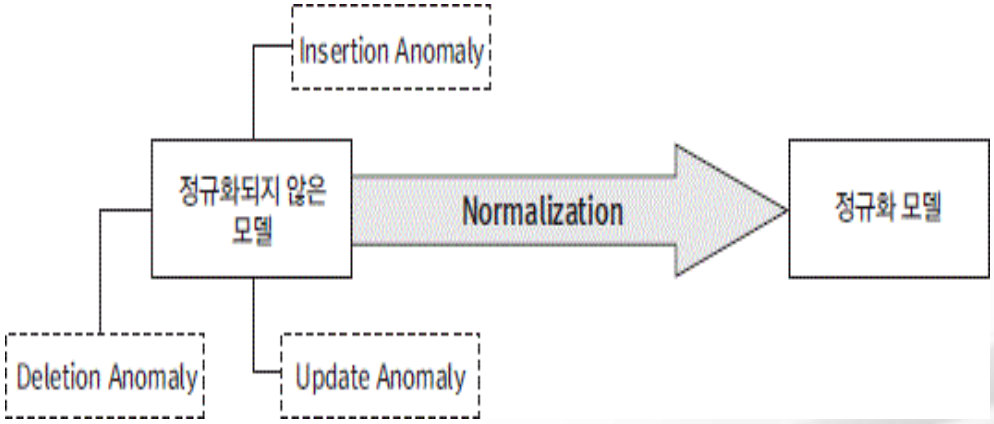
- E-R Diagram Sample
  - ✓ UML



모델링

❖ 데이터의 중복으로 인한 이상 현상

- ✓ 변경 이상(Modification Anomaly)
- ✓ 삽입 이상(Insertion Anomaly)
- ✓ 삭제 이상(Deletion Anomaly)



모델링

❖ 이상 현상(Anomaly)

✓ 삽입 이상: 릴레이션에 새 데이터를 삽입하기 위해 원치 않는 불필요한 데이터도 함께 삽입해야 하는 문제

고객아이디	이벤트번호	당첨여부	고객이름	등급
apple	E001	Y	정소화	gold
apple	E005	N	정소화	gold
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
orange	E004	N	김용욱	silver
melon	NULL	NULL	성원용	gold

모델링

❖ 이상 현상(Anomaly)

✓ 삭제 이상: 릴레이션에서 데이터를 삭제하면 꼭 필요한 데이터까지 함께 삭제하여 데이터가 손실되는 연쇄 삭제 현상

고객아이디	이벤트번호	당첨여부	고객이름	등급
apple	E001	Y	정소화	gold
apple	E005	N	정소화	gold
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
~~orange~~	~~E004~~	~~N~~	~~김용욱~~	~~silver~~

← 데이터 손실 발생!

모델링

❖ 이상 현상(Anomaly)

✓ 갱신 이상: 릴레이션의 중복된 데이터들 중 일부만 수정하여 데이터가 불일치하게 되는 모순이 발생하는 것

고객아이디	이벤트번호	당첨여부	고객이름	등급
apple	E001	Y	정소화	vip
apple	E005	N	정소화	vip
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
orange	E004	N	김용욱	silver

← 데이터 불일치 발생!

모델링

- 함수적 종속(Functional Dependency)
  - ✓ 함수적 종속이란 어떤 릴레이션 R이 있을 때 속성 X 와 Y가 존재하는 경우 X의 값을 알면 Y의 값을 바로 식별할 수 있고 X의 값에 Y의 값이 달라질 때 Y는 X에 함수적 종속되었다고 하고 X를 결정자 그리고 Y를 종속자라고 함
  - ✓ 기호로 표현하면 X→Y
  - ✓ 함수적 종속 관계의 종류
    - 완전 함수적 종속
    - 부분 함수적 종속
    - 이행적 함수 종속

모델링

❖ 함수적 종속(Functional Dependency)

- ✓ 학번을 알면 이름, 나이, 성별 속성을 식별할 수 있으며 학번이 다르면 그에 따른 값도 다름
- ✓ 이름, 나이, 성별 속성은 학번에 함수적 종속 관계
- ✓ 전공 속성 또한 전공코드에 함수적 종속 관계
- ✓ 학번→이름, 학번→나이, 학번→성별 형태로 표현

학번	이름	나이	성별	전공코드	전공명
110011	박지현	26	여성	AAA1	국문학과
110011	박지현	26	여성	C0B7	컴퓨터공학과
131001	김민석	25	남성	C0A5	전기전자공학과
120006	홍현희	25	여성	B1027	무용과
150705	한태민	23	남성	C0A5	전기전가공학과
171024	설화영	22	여성	B01K2	공예과

모델링

❖ 함수적 종속(Functional Dependency)

✓ 완전 함수적 종속(Full Functional Dependency)

- 기본키를 구성하는 모든 속성이 포함된 기본키의 집합에 종속된 경우로 기본키가 하나의 속성으로 구성된 경우는 모든 속성이 기본키에 대하여 무조건 완전 함수적 종속
- 이 릴레이션에서는 기본키가 회원번호 속성으로 구성되어 있는데 여기서 이름, 나이, 거주지역 속성은 기본키인 회원번호를 알아야 식별 가능하기 때문에 이름, 나이, 거주지역은 회원번호에 완전 함수 종속

회원번호	이름	나이	거주지역
A001	송민지	17	서울
A002	박아람	15	부산
A003	이예은	16	대전

모델링

❖ 함수적 종속(Functional Dependency)

✓ 완전 함수적 종속(Full Functional Dependency)

- 릴레이션의 기본키는 고객ID와 상품코드 속성으로 구성되어 있는데 수량 속성은 기본키를 구성하는 고객ID, 상품코드 속성을 모두 알아야 식별할 수 있기 때문에 수량은 기본키에 대하여 완전 함수 종속

고객ID	상품코드	주문상품	수량	가격
AAAA01	T001	티셔츠	2	12000
AAAA01	B110	청바지	1	11000
AAAA02	B110	청바지	2	22000
AAAA03	T091	와이셔츠	1	15000
AAAA03	O100	원피스	1	19000

모델링

❖ 함수적 종속(Functional Dependency)

✓ 부분 함수적 종속(Partial Functional Dependency)

- 기본키가 여러 속성으로 구성되어 있을 경우 기본키를 구성하는 속성 중 일부에 종속되는 경우
- 기본키가 고객ID 와 상품코드 속성으로 구성된 릴리이션에서 주문상품은 기본키 중 상품코드만 알아도 식별할 수 있는데 이 경우 주문상품 속성은 기본키에 부분 함수 종속된 관계

고객ID	제품코드	주문상품	수량	가격
AAAA01	T001	티셔츠	2	12000
AAAA01	B110	청바지	1	11000
AAAA02	B110	청바지	2	22000
AAAA03	T091	와이셔츠	1	15000
AAAA03	O100	원피스	1	19000

모델링

❖ 함수적 종속(Functional Dependency)

✓ 이행적 함수 종속(Transitive Functional Dependency): 릴레이션에서 X, Y, Z라는 3 개의 속성이 있을 때 X→Y, Y→Z 이란 종속 관계가 있을 경우, X→Z가 성립될 때 이행적 함수 종속이라고 하는데 X를 알면 Y를 알고 그를 통해 Z를 알 수 있는 경우

● 회원번호를 알면 생년월일을 알 수 있고 생년월일을 알면 나이를 알 수 있으면 회원번호를 알면 나이를 알 수 있는데 이러한 관계가 이행적 함수 종속

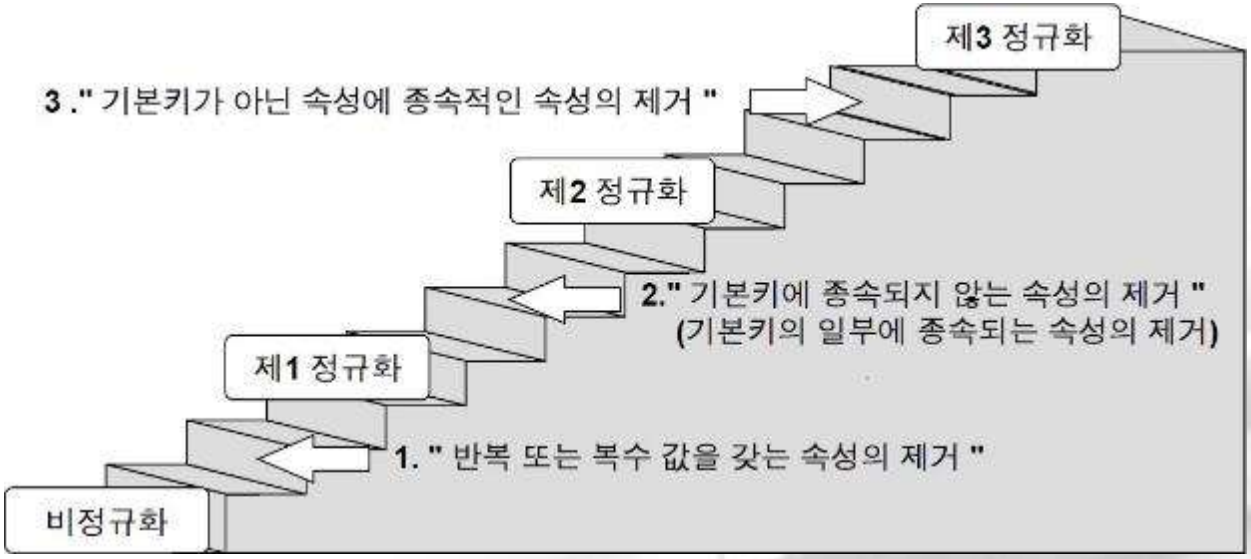
회원번호	생년월일	나이	주소
A001	1995.1.3	26	군포 산본
A002	1996.1.16	25	강남 청담
A003	1997.2.11	24	오클랜드
A004	1997.3.27	23	부리람

모델링

- 정규화(Normalization)
  - ✓ 논리 데이터 모델을 일관성이 있고 안정성 있는 자료 구조로 만드는 단계
  - ✓ 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위한 방법으로 데이터를 분해하는 과정
  - ✓ 데이터 모델의 독립성을 확보하기 위한 방법
  - ✓ 정규화를 수행하면 비지니스의 변경에 유연하게 대처해서 데이터 모델의 변경을 최소할 할 수 있음
  - ✓ 정규화의 장점
    - 중복값이 줄어듬
    - NULL 값이 줄어듬
    - 복잡한 코드로 데이터 모델을 보완할 필요가 없음
    - 새로운 요구 사항의 발견 과정을 도움
    - 업무 규칙의 정밀한 포착을 보증
    - 데이터 구조의 안정성을 최대화

모델링

- 정규화
  - ✓ 정규화는 1차 정규화부터 BCNF(Boyce-Codd Normal Form)을 포함한 5차 정규화까지로 구성
  - ✓ 일반적으로 중복이 최소로 발생하는 제 3 정규화 까지 진행



모델링

- 정규화
  - ✓ 제 1 정규형(1NF, First Normal Form): 한 릴레이션을 구성하는 모든 도메인이 원자값으로 구성
  - ✓ 제 2 정규형(2NF, Second Normal Form): 제 1 정규형을 만족하면서 릴레이션에 존재하는 부분 함수적 종속을 제거하여 모든 속성이 기본키에 완전 함수 종속이 되도록 만들어진 정규형
  - ✓ 제 3 정규형(3NF, Third Normal Form): 제 2 정규형을 만족하면서 릴레이션을 구성하는 속성들 간의 이행적 종속관계를 분해하여 속성들이 비이행적 함수 종속관계를 만족하도록 만들어진 정규형
  - ✓ 보이스-코드(BCNF, Boyce-Codd Normal Form): 제 3 정규형을 만족하면서 릴레이션의 모든 결정자가 후보키가 되도록 하는 정규형
  - ✓ 제 4 정규형(4NF, Fourth Normal Form): BCNF를 만족하면서 릴레이션에서 다치 종속 관계를 제거한 정규형
  - ✓ 제 5 정규형(5NF, Fifth Normal Form): 후보키를 통하지 않은 조인 종속(Join Dependency)을 제거한 정규형

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

다가속성(Multivalued Attributes)

- 같은 종류의 값을 여러 개 가지는 속성을 다가 속성이라 함

#고객ID	고객이름	주민등록번호	전화번호
Blues	홍길동	123456-7890123	123-4567, 234-5678, 345-6789
Pupils	김길동	234567-8901234	456-7890, 567-8901

- 위 그림의 모델은 하나의 속성에 여러 전화번호를 관리하고 있으므로 속성은 반드시 하나의 값을 가져야한다는 1정규형에 어긋남
- 위와 같이 다가 속성이 존재하면 새로운 릴레이션이 필요함
- 위와 같은 릴레이션을 1정규화하면 아래와 같음

[고객]

#고객ID	고객이름	주민등록번호
Blues	홍길동	123456-7890123
Pupils	김길동	234567-8901234

[고객전화번호]

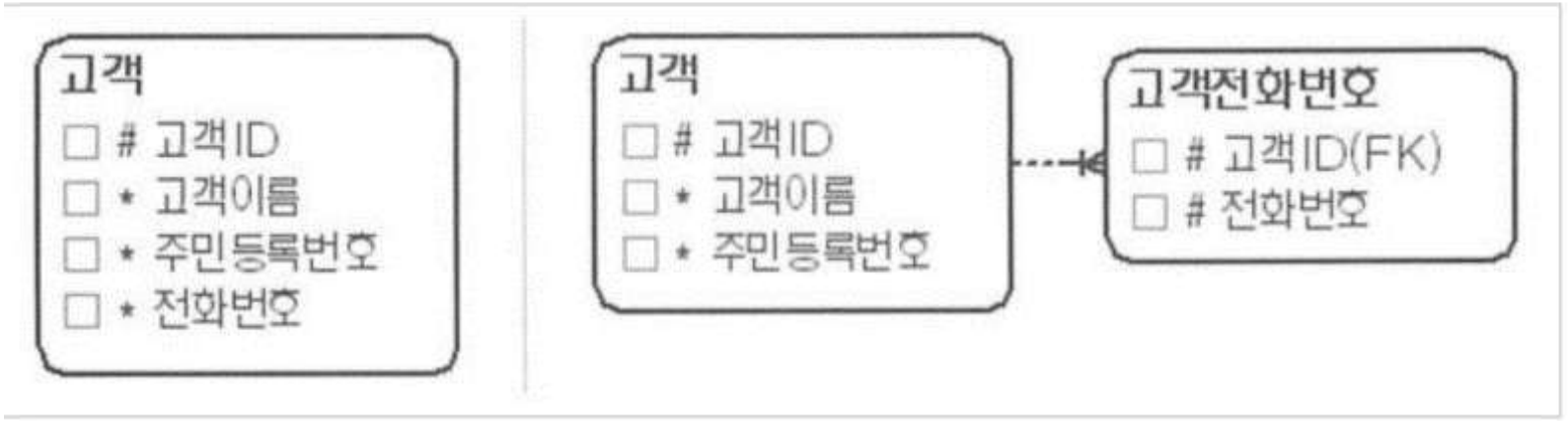
#고객ID	#전화번호
Blues	123-4567
Blues	234-5678
Blues	345-6789
Pupils	456-7890
Pupils	567-8901

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

• 이를 모델로 표현하면 아래의 오른쪽 모델과 같음



모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

- 위와 같이 하나의 속성에 여러 값을 가지는 사례는 흔치 않고 보통은 아래와 같은 릴레이션으로 관리함

#고객ID	고객이름	주민등록번호	#전화번호
Blues	홍길동	123456-7890123	123-4567
Blues	홍길동	123456-7890123	234-5678
Blues	홍길동	123456-7890123	345-6789
Pupils	김길동	234567-8901234	456-7890
Pupils	김길동	234567-8901234	567-8901

모델링

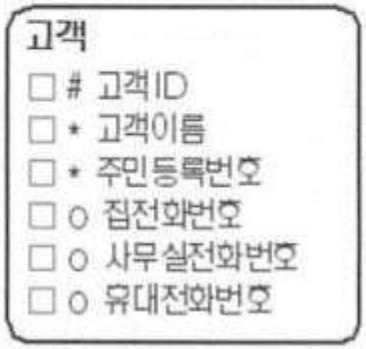
❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

- 위 릴레이션은 물리적으로는 모든 속성이 하나의 값만을 가지고 있지만, 논리적으로 하나의 값을 가지고 있다고 볼 수 없음
  고객이름과 주민등록번호 속성은 중복되었고, 고객ID에 종속돼서 2정규형을 만족하지 못함
- 하지만 성능을 위해 비정규화된 아래와 같은 모델을 사용할 수 있음(반복 속성이 고정적일 경우)
- 이 경우 속성에 의미를 명확히 부여해야함 전화번호1, 전화번호2, 전화번호3 과 같이 사용하는 것은 바람직하지 않음

[고객]

#고객ID	고객이름	주민등록번호	집전화번호	사무실전화번호	휴대전화번호
Blues	홍길동	123456-7890123	123-4567	234-5678	345-6789
Pupils	김길동	234567-8901234	456-7890	567-8901	{null}



- 위와 같은 비정규형은 null 데이터가 많이 발생할 수 있고 인덱스가 많이 필요하며 'or' 조회가 많이 발생할 수 있는 단점이 존재

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

복합속성(Composite Attributes)

- 하나의 속성이 여러 개의 속성으로 분리될 수 있을 때 이를 복합 속성이라 함
- 복합 속성은 속성의 관리 수준에 따라 달라질 수 있음

#고객ID	고객이름	집전화번호	주소
Blues	홍길동	123-4567	서울시 은평구 증산동 0-1
Pupils	김길동	456-7890	서울시 구로구 온수동 0-2

- 주소 속성은 시,구,동,번지 데이터를 따로 관리하게 되면 복합 속성이 됨
- 복합 속성처럼 보인다 하여 무조건 분해하면 안 됨
- 위와 같은 주소 속성의 경우 한꺼번에 입력하고 조회하는데 분해해서 입력하고 각각을 합치는 것은 비효율적
- 위의 표에서 시,구,동,번지가 각각 유의미한 속성으로 통계등에 활용된다면 분해하는 것이 좋음

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

반복속성

- 한 릴레이션에서 반복 형태의 속성이 있어서는 안 됨

#주문번호	고객ID	주문일자	상품코드1	수량1	상품코드2	수량2	상품코드3	수량3
1234	blues	2027-10-10	P0001	2	A0001	1	{null}	{null}
1235	pupils	2028-01-01	B0002	1	C0002	1	B0001	2
1236	blues	2029-06-30	P0001	1	{null}	{null}	{null}	{null}

- 아래와 같이 정규화 하여야함

[주문]

#주문번호	고객ID	주문일자
2027	blues	2027-10-10
2028	pupils	2028-01-01
2029	blues	2029-06-30

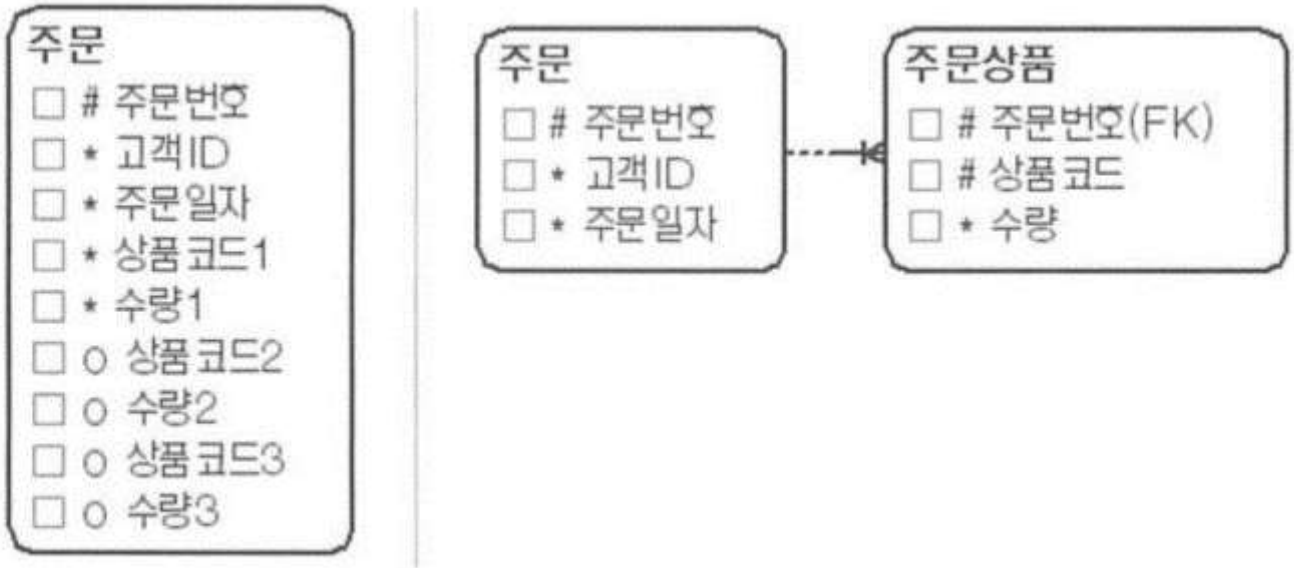
[주문상품]

#주문번호	#상품코드	수량
1234	P0001	2
1234	A0001	1
1235	B0002	1
1235	C0002	1
1235	B0001	2
1236	P0001	1

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)



- 다만 위와 같은 모델에서 상품이 3개 이상은 절대로 주문 불가하다면 비정규화된 모델이 유리할 수 있음

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)

#주문번호	#상품코드	고객ID	주문일자	수량
1234	P0001	blues	2027-10-10	2
1234	A0001	blues	2027-10-10	1
1235	B0002	pupils	2028-01-01	1
1235	C0002	pupils	2028-01-01	1
1235	B0001	pupils	2028-01-01	2

- 고객 ID와 주문일 속성이 인스턴스 별로 중복되며 상품코드와 수량은 다가 속성의 성격을 가짐
- 다음 그림과 같이 표현할 수 있고 이를 중첩 릴레이션(Nested Relation 또는 Relation-Valued Attribute)라고 함(하나의 인스턴스 내부에 다시 인스턴스가 존재하는 형태)

모델링

❖ 정규화

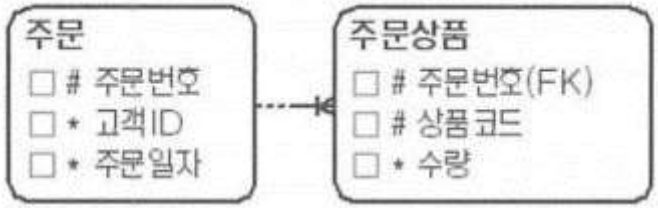
✓ 제 1 정규형(1NF, First Normal Form)

#주문번호	고객ID	주문일자	#상품코드	수량
1234	blues	2027-10-10	P0001	2
			A0001	1
1235	pupils	2028-01-01	B0002	1
			C0002	1
			B0001	2

모델링

❖ 정규화

✓ 제 1 정규형(1NF, First Normal Form)



[주문]

#주문번호	고객ID	주문일자
1234	blues	2027-10-10
1235	pupils	2028-01-01

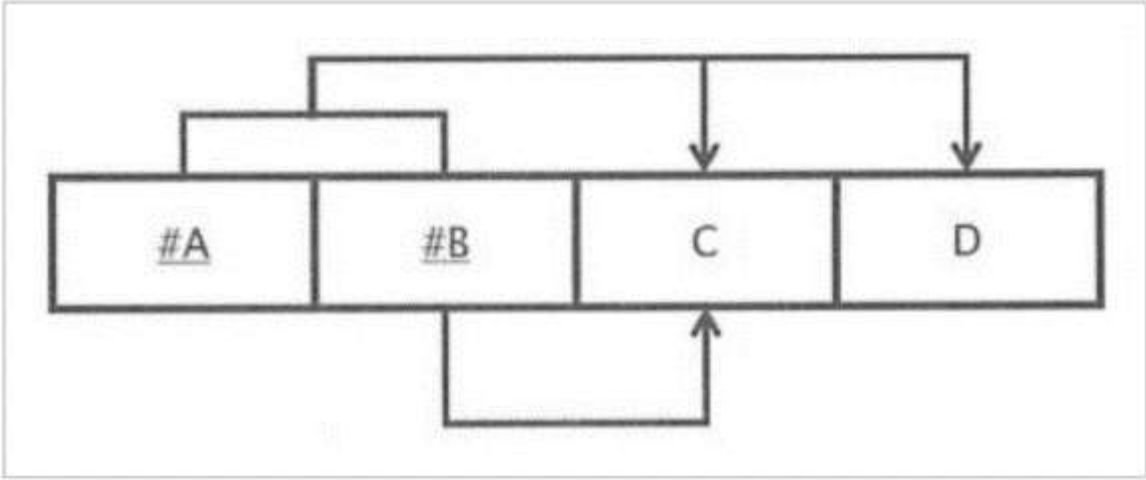
[주문상품]

#주문번호	#상품코드	수량
1234	P0001	2
1234	A0001	1
1235	B0002	1
1235	C0002	1
1235	B0001	2

모델링

❖ 정규화

✓ 제 2 정규형(2NF): 모든 속성은 기본키에 완전 함수 종속 – 기본키가 2개 이상의 복합 속성으로 구성된 경우만 수행



- C 속성은 주 식별자의 일부분인 B 속성에만 종속되어 부분 함수 종속이므로 B 속성을 주 식별자로 하는 릴레이션을 추가해 두 개의 릴레이션으로 분리해야 함

모델링

❖ 정규화

✓ 제 2 정규형(2NF)

[주문상품]

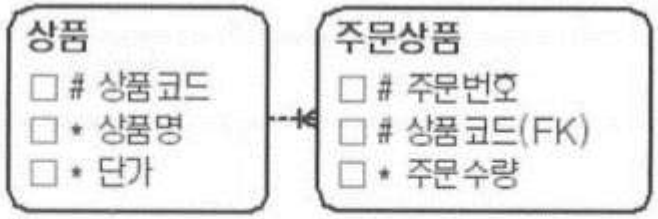
#주문번호	#상품코드	상품명	단가	주문수량
1234	P0001	오라클 아키텍처	10000	2
1234	A0001	데이터 모델링이란	15000	1
1235	B0002	DW 구축	12000	1
1235	C0002	지식 경영 시대	5000	1
1235	A0001	데이터 모델링이란	15000	2

- 상품명과 단가는 주 식별자(주문번호,상품코드)에 종속되지 않고 상품코드에만 종속되므로 2정규화 대상

모델링

❖ 정규화

✓ 제 2 정규형(2NF)



[상품]

#상품코드	상품명	단가
P0001	오라클 아키텍처	10000
A0001	데이터 모델링이란	15000
B0002	DW 구축	12000
C0002	지식 경영 시대	5000

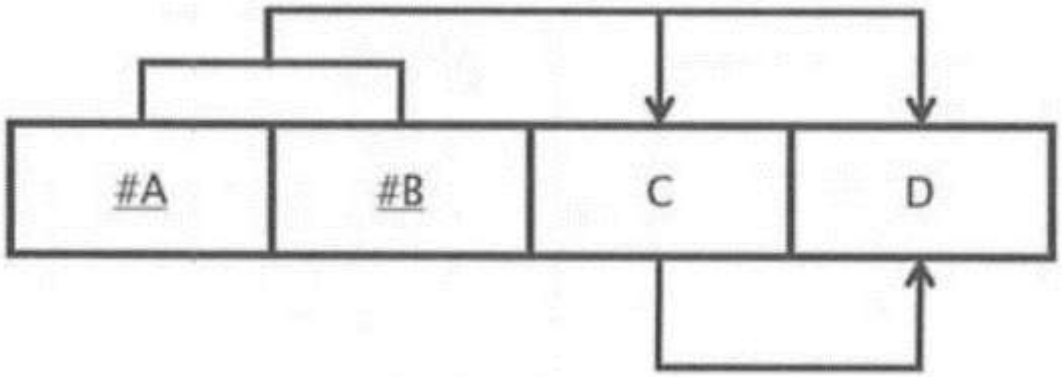
[주문상품]

#주문번호	#상품코드	주문수량
1234	P0001	2
1234	A0001	1
1235	B0002	1
1235	C0002	1
1235	A0001	2

모델링

❖ 정규화

✓ 제 3 정규형(3NF): 이행적 함수 종속 제거



모델링

❖ 정규화

✓ 제 3 정규형(3NF): 이행적 함수 종속 제거

[주문]

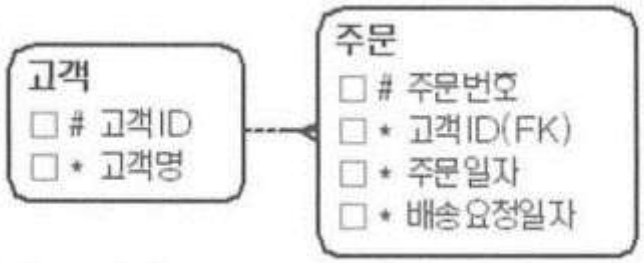
#주문번호	고객ID	고객명	주문일자	배송요청일자
1234	blues	홍길동	2027-10-10	2027-10-13
1235	pupils	김길동	2028-01-01	2028-01-03
1236	blues	홍길동	2027-12-25	2027-12-26

- 고객 ID는 고객명의 결정자임

모델링

❖ 정규화

✓ 제 3 정규형(3NF): 이행적 함수 종속 제거



[고객]

#고객ID	고객명
blues	홍길동
pupils	김길동

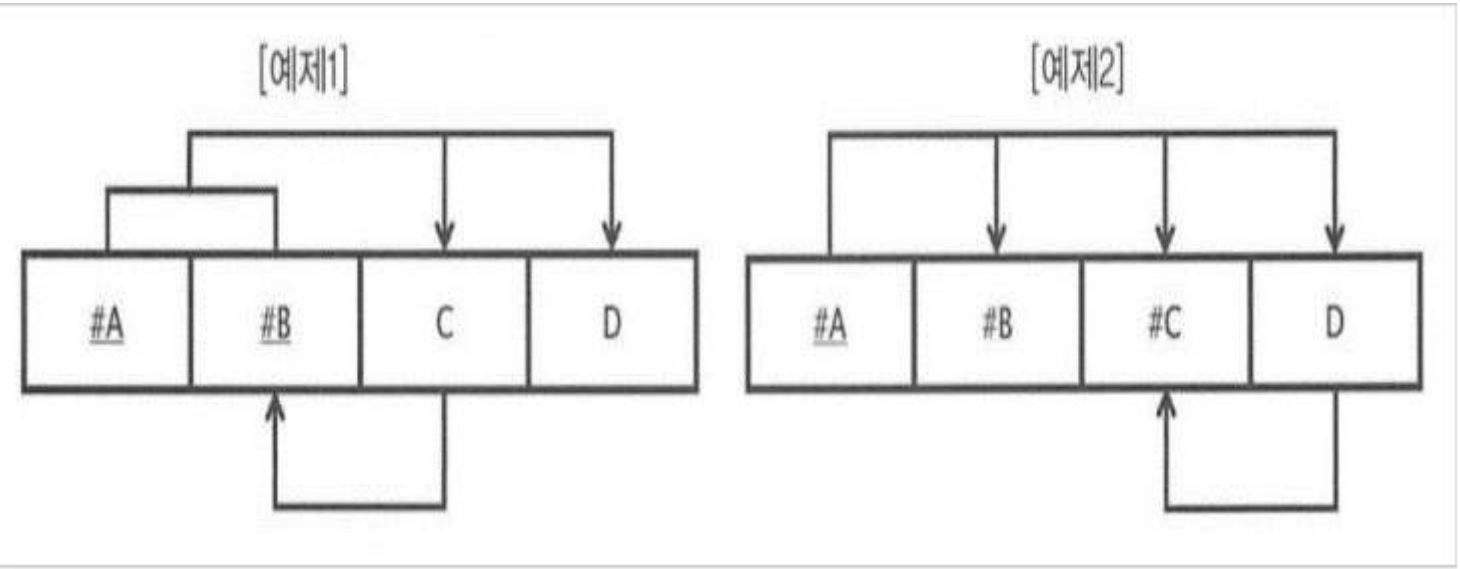
[주문]

#주문번호	고객ID	주문일자	배송요청일자
1234	blues	2027-10-10	2027-10-13
1235	pupils	2028-01-01	2028-01-03
1236	blues	2027-12-25	2027-12-26

모델링

❖ 정규화

✓ Boyes-Codd 정규형(BCNF – 강한 3 정규형): 모든 결정자가 후보키이어야 함



- 예제 1에서 C 가 B의 값을 결정함
- 예제 2에서 A가 B, C, D의 값을 결정함
- 후보키 아닌 속성이 다른 속성을 결정함

모델링

❖ 정규화

✓ Boyes-Codd 정규형(BCNF)

[수강과목]

#학생번호	#과목명	교수번호	학점
S124	수학	P987	A
S124	물리학	P654	B
S124	전자기학	P321	B
S568	물리학	P654	C
S568	철학	P135	A

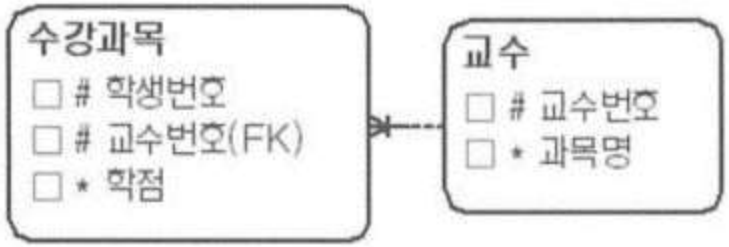
모델링

- ❖ 정규화
  - ✓ Boyes-Codd 정규형(BCNF)
    - ● 학생의 학점을 관리하는 릴레이션
      - ◆ 학생은 여러 과목 수강 가능
      - ◆ 교수는 한 과목만 강의
      - ◆ 여러 교수가 동일한 과목 강의 불가
      - ◆ FD1: (학생번호,과목명) -> (교수번호,학점)
      - ◆ FD2: 교수번호->과목명
    - ● 종속자인 과목명 속성이 주 식별자에 포함돼 있으므로 BCNF에 어긋남

모델링

❖ 정규화

✓ Boyes-Codd 정규형(BCNF)



[교수]

#교수번호	과목명
P987	수학
P654	물리학
P321	전자기학
P135	철학

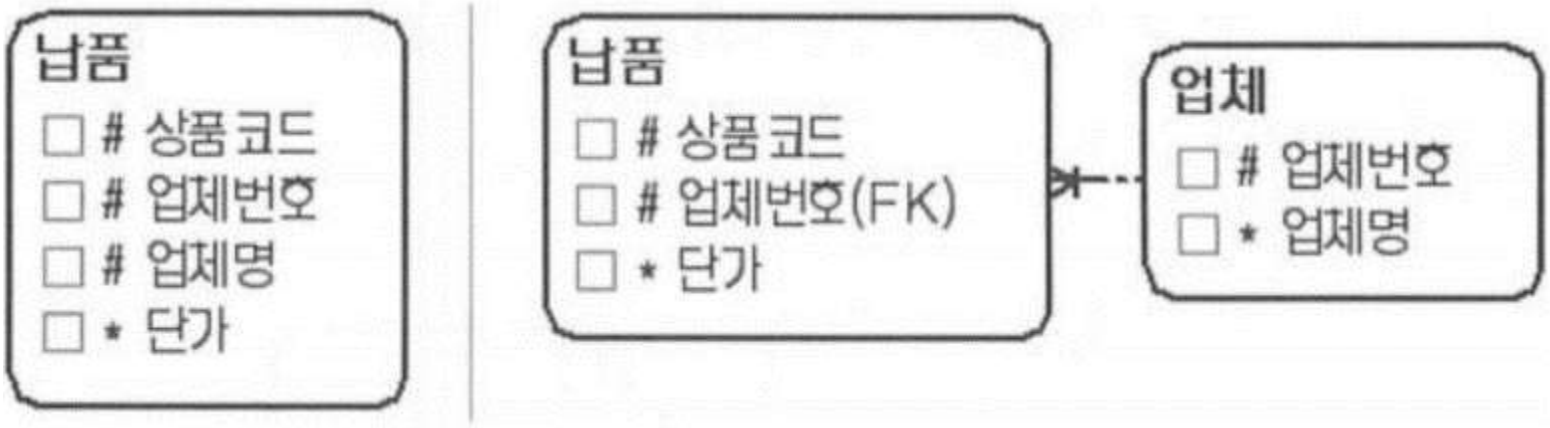
[수강과목]

#학생번호	#교수번호	학점
S124	P987	A
S124	P654	B
S124	P321	B
S568	P654	C
S568	P135	A

모델링

❖ 정규화

✓ Boyes-Codd 정규형(BCNF)



- 업체번호 속성이 업체명 속성을 종속하므로 오른쪽 모델과 같이 정규화해야 함
- 상품코드와 업체번호만으로도 주 식별자가 될 수 있는데 업체명까지 넣음으로써 슈퍼 식별자가됨

모델링

- 정규화
  - ✓ 제 4 정규형(4NF)
    - 다가 종속(MVD:Multivalued Dependency) 개념이 기반이 되는 정규형
    - 다가 종속이란 한 릴레이션에 다가 속성이 두 개 이상 존재할 때 발생
    - 하나의 다가 속성의 모든 값이 다른 다가 속성의 모든 값 마다 중복되는 문제점이 발생할 수 있는데 이를 다가 종속이라 함
    - 속성 A의 하나의 값이 속성 B의 여러 값을 결정하면 A->->B로 표시하며 'A가 B를 다가 결정(Multidetermine)한다' 또는 'B가 A에 다가종속(Multidependent)됐다' 라고 함
    - 4 정규형은 이런 다가 종속을 제거하는 것

모델링

❖ 정규화

✓ 제 4 정규형(4NF)

[릴레이션1]

사원	기술	언어
홍길동	모델링 튜닝	영어 한국어
김길동	자바 C++ 닷넷	한국어 일어

[릴레이션2]

#사원	#기술	#언어
홍길동	모델링	영어
홍길동	모델링	한국어
홍길동	튜닝	영어
홍길동	튜닝	한국어
김길동	자바	한국어
김길동	자바	일어
김길동	C++	한국어
김길동	C++	일어
김길동	닷넷	한국어
김길동	닷넷	일어

모델링

- ❖ 정규화
  - ✓ 제 4 정규형(4NF)
    - 사원은 여러 기술을 가지고 있으며 구사할 수 있는 언어도 여러 개임
    - 기술과 언어는 아무 연관이 없음
    - 이렇게 되면 많은 중복 데이터가 발생함(아노말리 발생 가능)
    - 중복 데이터를 줄이기 위해 그림처럼 수정하면 모델링과 영어가 연관성을 가지는 것처럼 보이게 됨

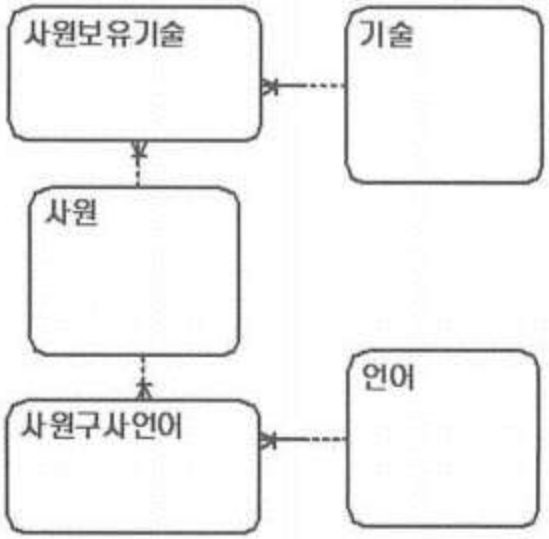
모델링

❖ 정규화

✓ 제 4 정규형(4NF)

#사원	#기술
홍길동	모델링
홍길동	튜닝
김길동	자바
김길동	C++
김길동	닷넷

#사원	#언어
홍길동	영어
홍길동	한국어
김길동	한국어
김길동	일어



모델링

- 정규화
  - ✓ 제 5 정규형(5NF)
    - 조인 종속(Join Dependency) 개념을 기반으로 수행
    - 조인 종속을 없앤 것이 5정규형
    - 무손실 조인 비 부가 조인: 만약 하나의 릴레이션을 여러 개의 릴레이션으로 분해(Decomposition)한 후 공통(식별자) 속성으로 조인하여 데이터 손실없이 원래의 릴레이션으로 복원할 수 있으면 이를 무손실 조인(Lossless join)이라 함
    - 조인한 결과에 원래 릴레이션에 없는 데이터(가짜 튜플)가 존재하지 않으면 이를 비 부가적 조인(NonADDitive join)이라 함
    - 필요한 데이터가 사라지지 않는 무손실 분해가 되고 필요없는 데이터가 생기지 않는 비 부가적 분해가 된 릴레이션이 5정규형임

모델링

❖ 정규화

✓ 제 5 정규형(5NF)

#사원	#기술	#언어
홍길동	모델링	영어
홍길동	파워포인트	중국어
홍길동	모델링	프랑스어
김길동	파워포인트	일어
김길동	모델링	러시아어

모델링

❖ 정규화

✓ 제 5 정규형(5NF)

#사원	#기술
홍길동	모델링
홍길동	파워포인트
김길동	파워포인트
김길동	모델링

#기술	#언어
모델링	영어
파워포인트	중국어
모델링	프랑스어
파워포인트	일어
모델링	러시아어

#사원	#언어
홍길동	영어
홍길동	중국어
홍길동	프랑스어
김길동	일어
김길동	러시아어

모델링

❖ 정규화

✓ 제 5 정규형(5NF)

- 세 개의 릴레이션 중에서 어느 두 개의 릴레이션만 조인해서는 원 데이터를 만들 수 없고 반드시 세 개의 릴레이션을 조인해야 원하는 요건을 얻을 수 있음
- 5 정규형은 조인 종속이 존재하지 않아야 하므로 3개의 릴레이션으로 분해하면 더 이상 분해할 수 없는 5 정규형이 됨
- 5 정규형은 분할하고(Project) 합치는(Join) 개념 때문에 PJ 정규형(Project-Join Normal Form)이라고도 함
- 5 정규형은 더는 분해할 수 없는 릴레이션이므로 가장 이상적인 정규형이지만 하나의 릴레이션에서 데이터를 관리해도 아노말리 현상이 발생하지 않아 조인 종속이 발생하는 상태로 릴레이션을 관리하는 것이 현실적임
- 4 정규형은 5 정규형과 유사하지만 속성 간의 연관 관계가 있을 때 5 정규형이 필요

모델링

- ❖ 정규화
  - ✓ 비 정규형과 정규형의 특징
    - 비정규형(Non-Normal Form)
      - ◆ 업무 요건의 변경에 매우 취약 – 모델의 확장성이 좋지 않음
      - ◆ 인덱스 수가 증가하고 특정 요건 조회 시 SQL이 복잡해 짐
      - ◆ Entity의 속성이 추가될 가능성이 없을 때 사용 가능
      - ◆ 속성 레벨로 관리되는 데이터의 자식 Entity를 가질 수 없음(여러 데이터가 하나의 인스턴스에 묶여 있어 개별로 관리해야 할 하위 데이터가 있으면 관리 불가능)
    - 정규형(Normal Form)
      - ◆ 업무 요건 변경이 유연. 확장성 좋음
      - ◆ 인덱스 수 감소, 특정 요건 조회 시 SQL 단순해 짐(복잡해질 수도 있음)
      - ◆ Entity의 속성이 추가될 가능성이 존재할 때 사용
      - ◆ 로우 레벨로 관리되는 데이터의 자식 Entity를 가질 수 있음

모델링

- ❖ 정규화의 문제점 과 해결 방안
  - ✓ 정규화의 문제점
    - 빈번한 Join 연산의 증가 : 시스템 성능 저하
    - 부자연스러운 DB Semantic 초래
    - 조회/검색 위주의 응용 시스템에 부적합
  - ✓ 문제점 해결 방안 및 업계 동향
    - 정규화 완료 후 업무특성과 성능 향상을 위해 De-normalization 수행으로 유연성 확보
    - 업무 특성 별 정규화 수준
      - ◆ 온라인 처리: 소규모 정규화
      - ◆ 단순 조회 용 데이터: 비정규화
      - ◆ 배치 처리: 비정규화
    - 응답 시간이 요구되는 온라인 실시간 시스템에는 제한적인 정규화
    - 기억 장치가 대용량화 되고 저렴해지면서 정규화의 중요성이 저하되고 성능이 우선시 됨

모델링

❖ 반정규화

✓ 반정규화

- 반정규화는 정규화된 Entity, 속성, 관계를 시스템의 성능 향상 및 개발과 운영의 단순화를 위해 중복,통합,분리 등을 수행하는 데이터 모델링 기법
- 디스크 I/O량이 많아서 조회 시 성능이 저하되거나 테이블끼리의 경로가 너무 멀어 조인으로 인한 성능저하가 예상되거나 컬럼을 계산하여 읽을 때 성능이 저하될 것이 예상되는 경우 반정규화를 수행
- 업무적으로 조회에 대한 처리 성능이 중요하다고 판단될 때 부분적으로 반정규화를 고려하는데 반정규화를 수행하게 되면 모델의 유연성은 떨어지게 됨
- 설계 단계에서 반정규화를 적용하게 되며 반정규화 미수행시에는 다음과 같은 현상이 발생될 수 있음
  - ◆ 성능이 저하된 데이터베이스가 생성될 수 있음
  - ◆ 구축 단계나 시험 단계에서 반정규화를 적용할 때 수정에 따른 노력 비용이 많이 소모

모델링

- ❖ 반정규화
  - ✓ 반정규화 적용 방법
    - 반정규화에 대한 필요성이 결정되면 컬럼의 반정규화 뿐만 아니라, 테이블의 반정규화, 관계의 반정규화를 종합적으로 고려하여 적용
    - 반정규화는 막연하게 중복을 유도하는 것만을 수행하기 보다는 성능을 향상시킬 수 있는 다른 방법을 고려하고 그 이후에 반정규화를 적용
    - 반정규화의 대상을 조사
      - ◆ 자주 사용되는 테이블에 액세스하는 프로세스의 수가 가장 많고 항상 일정한 범위만을 조회하는 경우에 반정규화를 검토
      - ◆ 테이블에 대량데이터가 있고 대량의 범위를 자주 처리하는 경우 성능을 보장할 수 없는 경우에 반정규화를 검토
      - ◆ 통계성 프로세스에 의해 통계정보를 필요로 할 때 별도의 통계 테이블(반정규화)를 생성
      - ◆ 테이블에 지나치게 조인을 많이 하게 되어 데이터를 조회하는 것이 기술적으로 어려울 경우 반정규화를 검토

모델링

- ❖ 반정규화
  - ✓ 반정규화 적용 방법
    - ● 반정규화의 대상에 대해 다른 방법으로 처리할 수 있는지 검토
      - ◆ 여러 테이블을 조인하여 데이터를 조회하는 것이 기술적으로 어려운 경우 Vlew를 검토하는데 조회 성능을 향상 시키지는 않지만 SQL작성의 미숙함으로 인하여 생기는 성능 저하를 예방할 수 있음
      - ◆ 대량의 데이터 처리나 부분 처리에 의해 성능이 저하되는 경우 클러스터링을 적용하거나 인덱스 조정을 통해 성능을 향상 시킬 수 있음
      - ◆ 대량의 데이터는 PK의 성격에 따라 파티셔닝 기법을 적용하여 성능 저하를 방
      - ◆ 어플리케이션에서 로직을 구현하는 방법을 변경하므로써 성능을 향상 시킬수 있음
    - ● 반정규화 적용
      - ◆ 반정규화 대상으로는 테이블, 속성, 관계에 대해 적용할 수 있으며, 중복을 통한 방법만이 반정규화가 아니고, 테이블,속성,관계를 추가/분할/제거할 수도 있다.

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 추가
      - ◆ 중복 테이블 추가: 다른 업무이거나 서버가 다른 경우 동일한 테이블구조를 중복하여 원격 조인을 제거하여 성능을 향상
      - ◆ 통계 테이블 추가: SUM,AVG 등을 미리 수행하여 계산해 둠으로써 조회시 성능을 향상
      - ◆ 이력 테이블 추가: 이력 테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력 테이블에 존재시키는 방법
      - ◆ 부분 테이블 추가: 하나의 테이블을 전체 칼럼 중 자주 이용하는 집중화된 컬럼이 있을 경우, 디스크I/O를 줄이기 위해 해당 컬럼들을 모아놓은 별도의 반정규화된 테이블을 생성

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 병합
      - ◆ 1:1 관계 테이블 병합 – 1:1 관계를 통합하여 성능 향상
      - ◆ 1:M 관계 테이블 병합 – 1:M 관계를 통합하여 성능 향상
      - ◆ 슈퍼/서브타입 테이블 병합 – 슈퍼/서브 관계를 통합하여 성능 향상
        - ▪ Super type: 여러 Sub Type이 공통으로 가지는 내용을 소유하고 있는 타입
        - ▪ Sub Type: 자신만의 가져야 하는 내용을 소유하고 있는 타입

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 분할 – 파티셔닝
      - ◆ 장점
        - ▪ 관리적 측면: partition 단위 백업, 추가, 삭제, 변경
          - • 전체 데이터를 손실할 가능성이 줄어들어 데이터 가용성이 향상
          - • partition별로 백업 및 복구가 가능
          - • partition 단위로 I/O 분산이 가능하여 UPDATE 성능이 향상
        - ▪ 성능적 측면: partition 단위 조회 및 DML수행
          - • 데이터 전체 검색 시 필요한 부분만 탐색해 성능이 증가
          - • Full Scan에서 데이터 Access의 범위를 줄여 성능 향상
          - • 필요한 데이터만 빠르게 조회할 수 있기 때문에 쿼리 자체가 가벼워짐
      - ◆ 단점
        - ▪ TABLE간 JOIN에 대한 비용이 증가
        - ▪ TABLE과 index를 별도로 파티셔닝할 수 없음
        - ▪ TABLE과 index를 같이 파티셔닝해야 하

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 대량 데이터발생에 따른 테이블 분할
      - ◆ 대량 데이터가 발생하는 테이블의 문제점
        - 설계가 잘 되어 있는 데이터 모델이라도 대량의 데이터가 하나의 테이블에 집약되어 있고 하나의 하드웨어 공간에 저장되어 있으면 성능 저하를 피하기 어려움
        - 인덱스도 또한 트리가 커지고 깊이가 깊어져 조회 성능에 영향을 미침
        - 입력/수정/삭제의 트랜잭션인 경우도 인덱스의 특성상 일의 양이 증가하여 더 많은 성능저하를 유발
        - 컬럼이 많아지게 되면 물리적인 디스크의 여러 블록에 걸쳐 데이터가 저장되게 되며 로우 길이가 너무 길어서 로우 체이닝과 로우 마이그레이션이 많아지게 되어 성능이 저하
      - ◆ 한 테이블에 많은 수의 칼럼을 가지고 있는 경우
        - 200개의 컬럼을 가진 도서 정보 테이블이 있다고 가정하고, 하나의 로우 길이가 10K이고 블록이 2K단위로 쪼개져 있으면 로우는 대략 5개의 블록에 걸쳐 저장
        - 여러 블록에 컬럼이 흩어져 있다면 디스크 I/O가 많이 발생
        - 트랜잭션 발생시 어떤 컬럼에 대해 집중적으로 발생되는지 분석하여 테이블 분할을 하면 디스크 I/O가 감소하여 성능을 개선할 수 있음

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 분할 – 파티셔닝
      - ◆ 종류
        - ▪ 수직 분할 – 컬럼 단위의 테이블을 디스크 I/O를 분산처리하기 위해 테이블을 1:1로 분리하여 성능 향상(트랜잭션의 처리되는 유형 파악이 선행)
        - ▪ 수평 분할(Shading) – 로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크I/O 및 데이터 접근의 효율성을 높여 성능을 향상하기 위해 로우 단위로 테이블을 쪼갬(관계가 없음)
        - ▪ 수직/수평 분할 절차
          - 데이터 모델링을 완성
          - 데이터베이스 용량을 산정
          - 대량 데이터가 처리되는 테이블에 대해서 트랜잭션 처리 패턴을 분석
          - 컬럼 단위로 집중화된 처리가 발생하는지, 로우 단위로 집중화된 처리가 발생되는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토
        - ▪ 컬럼 수가 너무 많은 경우는 테이블을 1:1형태로 분리할 수 있는지 검증하고 컬럼 수가 적지만 데이터 용량이 많아 성능저하가 예상되는 경우 테이블에 대해 파티셔닝 전략을 고려

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 분할 - 파티셔닝
      - ◆ 분할 기준
        - ▪ 범위 분할 (range partitioning): 분할 키 값이 범위 내에 있는지 여부로 구분하는데 우편 번호를 분할 키로 수평 분할하는 경우
        - ▪ 목록 분할 (list partitioning): 값 목록에 파티션을 할당 분할 키 값을 그 목록에 비추어 파티션을 선택하는 경우로 Country 라는 컬럼의 값이 Iceland , Norway , Sweden , Finland , Denmark 중 하나에 있는 행을 빼낼 때 북유럽 국가 파티션을 구축 할 수 있음
        - ▪ 해시 분할 (hash partitioning): 해시 함수의 값에 따라 파티션에 포함할지 여부를 결정하는 것으로 4개의 파티션으로 분할하는 경우 해시 함수는 0-3의 정수를 리턴
        - ▪ 합성 분할 (composite partitioning): 여러 기술을 결합하는 것을 의미하며, 예를 들면 먼저 범위 분할하고, 다음에 해시 분할 같은 것을 추가하는 방법으로 컨시스턴트 해시법은 해시 분할 및 목록 분할의 합성으로 간주 될 수 있고 키 공간을 해시 축소함으로써 일람할 수 있도록 함

모델링

- ❖ 반정규화
  - ✓ 테이블 반정규화
    - ● 테이블 분할 – 오라클의 경우 LIST PARTITION, RANGE PARTITION, HASH PARTITION, COMPOSITE PARTITION 등이 가능
      - ◆ RANGE PARTITION
        - ▪ 요금 정보처럼 항상 월 단위로 데이터를 처리하는 특성을 가진 경우 PK인 요금 일자의 년+월을 이용하여 12개의 파티션 테이블을 만들어서 성능 개선
        - ▪ 대상 테이블의 컬럼의 값이 날짜 또는 숫자 값으로 분리가 가능하고 각 영역별로 트랜잭션이 분리된다면 적용
        - ▪ 데이터 보관 주기에 따라 테이블에 데이터를 쉽게 지우는 것이 가능하므로 (파티션 테이블을 DROP) 테이블 관리가 용이
      - ◆ LIST PARTITION
        - ▪ 지점, 사업소, 사업장, 핵심적인 코드 값 등으로 PK가 구성되어 있는 테이블이라면, 값 각각에 의해 파티셔닝이 되는 LIST PATITION을 적용
        - ▪ 특정 값에 따라 분리 저장할 수는 있으나 RANGE PARTITON과 같이 데이터 보관 주기에 따라 쉽게 삭제하는 기능은 제공될 수 없음
      - ◆ HASH PARTITION 적용
        - ▪ HASH 조건에 따라 해시알고리즘이 적용되어 테이블이 분리되므로 설계자는 데이터가 어떤 테이블에 어떻게 들어갔는지 알 수 없으며 보관주기에 따라 쉽게 삭제하는 기능은 제공 될 수 없음

모델링

- ❖ 반정규화
  - ✓ 컬럼 반정규화
    - 중복 컬럼 추가: 조인시 성능저하를 예방하기 위해,중복된 컬럼을 위치시킴
    - 파생 컬럼 추가: 트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방하기 위해, 미리 계산하여 컬럼에 보관
    - 이력 테이블 컬럼 추가: 대량의 이력 데이터 처리시 불특정 일 조회나 최근 값을 조회할때 나타날 수 있는 성능저하를 예방하기 위해 기능성 컬럼(최근 값여부, 시작 일자,종료일자)을 추가함
    - PK에 의한 컬럼 추가: 복합 의미를 갖는 PK를 단일 속성으로 구성했을 때 발생되며, PK 안에 데이터가 존재하지만 성능 향상을 위해 일반속성으로 포함하는 방법
    - 응용 시스템 오작동을 위한 컬럼 추가: 업무적으로는 의미가 없으나, 데이터 처리시 오류로 인해 원래값으로 복구하길 원하는 경우 이전 데이터를 임시적으로 중복 보관하는 방법
  - ✓ 관계 반정규화
    - 중복 관계 추가: 여러 경로를 거쳐 조인이 가능하지만 성능 저하를 예방하기 위해 추가적인 관계를 맺는 방법

모델링

- ❖ 데이터베이스 구조와 성능
  - ✓ 인덱스 특성을 고려한 PK/FK 데이터베이스 성능 향상
    - ● PK/FK 칼럼 순서와 성능
      - ◆ 인덱스는 데이터를 접근할 때 경로를 제공하는 성능 측면에도 중요한 의미를 가지고 있기 때문에 설계 단계 말에 컬럼의 순서를 조정할 필요가 있음
      - ◆ PK가 복합 식별자인 경우 앞쪽에 위치한 속성이 가급적 '=' 이거나 최소 범위 'BETWEEN' '<>'가 들어와야 인덱스를 이용할 수 있음
      - ◆ FK라 하더라도 인덱스를 생성하도록 하고 인덱스 컬럼의 순서도 조회 조건을 고려하여 접근이 가장 효율적인 순서대로 생성
      - ◆ 성능 저하 이유
        - ▪ 인덱스는 PK의 순서대로 생성되는데 테이블에 접근하는 트랜잭션의 특징에 효율적이지 않은 인덱스로 생성되면 인덱스 범위를 넓게 사용하거나 Full Scan이 발생
        - ▪ PK에 없는 컬럼으로 테이블의 데이터를 자주 조회하는 경우
        - ▪ PK속성이 A+B와 B+A의 형태로도 빈번하게 조회되는 경우 더 자주 이용되는 형태로 PK순서를 구성하고 순서를 바꾼 인덱스를 추가로 생성하는 것이 필요

모델링

- 데이터베이스 구조와 성능
  - ✓ 물리적인 테이블에 FK제약이 걸려있지 않을 경우 인덱스 미생성으로 성능 저하
    - 두 테이블 사이에 FK참조 무결성 관계가 걸려있지 않더라도 데이터 모델 관계에 의해 상속받은 FK속성들은 조인 조건으로 이용하는 경우가 많으므로 FK인덱스를 생성하는 것을 기본정책으로 하는 것이 좋음
    - FK인덱스 생성을 기본정책으로 하되 향후 트랜잭션에 의해 거의 활용되지 않았을 때만 지우는 것이 적절한 방법

모델링

- ❖ 성능 데이터 모델링
  - ✓ 고려 사항: 성능 데이터 모델은 다음과 같은 프로세스로 진행하는 것이 데이터 모델링 단계에서 성능을 충분히 고려할 수 있는 방안
    - 데이터 모델링을 할 때 정규화를 정확하게 수행
    - 데이터베이스 용량 산정을 수행
    - 데이터베이스에 발생되는 트랜잭션의 유형을 파악
    - 용량과 트랜잭션 유형에 따라 반 정규화를 수행
    - 이력 모델의 조정, PK/FK조정, 슈퍼타입/서브타입 조정 등을 수행
    - 성능관점에서 데이터 모델을 검증: 데이터 모델을 검토할 때는 일반적인 데이터 모델 규칙만 검증하지 말고, 충분하게 성능이 고려 되었는지도 체크리스트에 포함하여 검증

모델링

❖ 성능 데이터 모델링

✓ 데이터베이스 성능 향상을 목적으로 설계 단계의 데이터 모델링 때 부터 정규화, 반 정규화, 테이블 통합, 테이블 분할, 조인 구조, PK, FK등 여러가지 성능과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것

✓ 성능 데이터 모델링 수행 시점

- 사전에 할 수록 비용이 들지 않음
- 분석/설계단계에서 성능을 고려한 데이터 모델링을 수행하면 나중에 성능 저하 때문에 발생하는 재업무(Rework)비용을 최소화할 수 있음
- 비즈니스 처리에 핵심적인 트랜잭션이 있다면, 프로젝트 초기에 운영 환경에 대비한 테스트환경을 구축하고 트랜잭션을 발생시켜 실제 성능테스트를 해보아야 함
- 데이터 모델의 구조도 변경하면, 가장 적절한 구조인지를 검토하여 디자인하는 전략 요구

분산 데이터베이스

- 분산 데이터베이스
  - ✓ 여러 곳으로 분산되어 있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스
  - ✓ 논리적으로 동일한 시스템에 속하나 네트워크를 통해 물리적으로 분산되어 있는 데이터들의 모임
  - ✓ 데이터베이스를 연결하는 빠른 네트워크환경을 이용하여 데이터베이스를 여러 지역 및 노드로 위치시켜 사용성/성능을 극대화시킨 데이터베이스
  - ✓ 분산 데이터베이스의 투명성(Transparency)
    - 분할 투명성(단편화): 하나의 논리적 Relation이 여러 단편으로 분할되어 각 단편의 사본이 여러 사이트에 저장
    - 위치 투명성: 사용하려는 데이터의 저장 장소를 명시가 필요하지 않음. 위치정보는 System Caltalog에 유지되어야 함
    - 지역 사상 투명성: 지역DBMS와 물리적 DB사이의 Mapping 보장 – 각 지역 시스템 이름과 무관한 이름 사용가능
    - 중복 투명성: DB객체가 여러 사이트에 중복되어 있는지 알 필요가 없는 성질
    - 장애 투명성: 구성요소(DBMS, Computer)의 장애에 무관한 트랜잭션의 원자성 유지
    - 병행 투명성: 다수 트랜잭션 동시 수행 시 결과의 일관성 유지, TimeStamp, 분산 2단계 Locking을 이용하여 구현

분산 데이터베이스

- 분산 데이터베이스 설계 방식
  - ✓ 상향식 설계 방식
    - 지역 스키마 작성 후 향후 전역 스키마를 작성하여 분산 데이터베이스를 구축
    - 지역별로 데이터베이스를 구축한 후에 전역 스키마로 통합하는 것
  - ✓ 하향식 설계 방식
    - 전역 스키마 작성 후 해당 지역 사상 스키마를 작성하여 분산 데이터베이스를 구축
    - 기업 전체의 전사 데이터 모델을 수렴하여 전역 스키마를 생성하고 그 다음 각 지역별로 지역 스키마를 생성하여 분산 데이터베이스를 구축하는 것

분산 데이터베이스

❖ 분산 데이터베이스의 적용 방법 및 장단점

✓ 장점

- 지역 자치성, 점증적 시스템 용량 확장
- 신뢰성과 가용성
- 효용성과 융통성
- 빠른 응답속도와 통신비용 절감
- 데이터의 가용성과 신뢰성 증가
- 시스템 규모의 적절한 조절
- 각 지역 사용자의 요구 수용 증대

✓ 단점

- 소프트웨어 개발 비용
- 오류의 잠재성 증대
- 처리비용의 증대
- 설계,관리의 복잡성과 비용
- 불규칙한 응답 시간
- 통제의 어려움
- 데이터 무결성에 대한 위협

분산 데이터베이스

- 분산 데이터베이스의 적용 기법
  - ✓ 분산 종류: 테이블 위치 분산, 테이블 분할 분산, 테이블 복제 분산, 테이블 요약 분석 전략이 있음
  - ✓ 가장 많이 사용하는 방식은 테이블 복제 분산 방식이며 성능이 저하되는 많은 데이터베이스에서 가장 유용하게 적용할 수 있는 기술적인 방법
  - ✓ 설계하는 방법은 일단 통합 모델링을 하고 각 테이블 별로 업무적 특징에 따라 지역 또는 서버 별로 테이블을 분산 또는 복제 배치하는 형태로 설정
  - ✓ 테이블 위치 분산
    - 테이블의 구조는 변하지 않으며 다른 데이터베이스에 중복되어 생성 되지도 않음
    - 설계된 테이블의 위치를 각각 다르게 위치시키는 것

분산 데이터베이스

- ❖ 분산 데이터베이스의 적용 기법
  - ✓ 테이블 분할(Fragmentation) 분산
    - 단순히 테이블의 위치만 다른 곳에 두는 것이 아니라 각각의 테이블을 쪼개어 분산하는 방법
    - 수평 분할(Horizontal Fragmentation)
      - ◆ 테이블을 특정 컬럼의 값을 기준으로 Row를 분리
      - ◆ 컬럼은 분리되지 않으며, 데이터를 한군데 집합시켜 놓아도 PK에 의해 중복발생이 일어나지 않음.
      - ◆ 각 지사 별로 사용하는 Row가 다를 때 이용
      - ◆ 각 지사에 존재하는 테이블에 대해 통합처리하는 경우 JOIN이 발생하여 성능 저하가 예상되므로 통합 처리 프로세스가 많지 않은 경우에만 이용
    - 수직 분할(Vertical Fragmentaion)
      - ◆ 특정 컬럼 값을 기준으로 컬럼을 분리
      - ◆ 컬럼을 기준으로 분할하였으므로 각각의 테이블에는 동일한 PK구조와 값을 가지고 있어야 함
      - ◆ 제품의 재고량은 각 지사별로 관리하고 단가는 본사에서 관리하는 경우
      - ◆ 실제 프로젝트에는 이런 환경을 구성하는 사례는 드뭄

분산 데이터베이스

- 분산 데이터베이스의 적용 기법
  - ✓ 테이블 복제(Replication) 분산
    - 동일한 테이블을 다른 지역이나 서버에 동시에 생성하여 관리하는 유형
    - 부분 복제(Segment Replication)
      - 테이블의 일부 내용만 다른 지역이나 서버에 위치시키는 방법
      - 본사 데이터베이스에는 테이블의 전체 내용이 들어가고 각 지사 데이터베이스에는 각 지사별로 관계된 데이터만 들어가는 경우
      - 보통 지사에서 데이터가 먼저 발생되고 본사에는 지사 데이터를 이용하여 통합하여 발생
      - 다른 지역간 데이터를 복제하는데는 많은 시간이 소요되므로 야간 배치 작업을 통해 수행되며 본사 지사 양쪽 데이터를 수정하여 전송하는 경우 정합성을 일치시키는 것이 어렵기 때문에 한쪽(지사)에서 데이터 수정을 하고 본사로 복제
    - 광역 복제(Broadcast Replication)
      - 테이블의 내용을 각 지역이나 서버에 위치시키는 방법
      - 본사와 지사 모두 동일한 정보를 가지고 있어 데이터 처리에 특별한 제약을 받지 않음
      - 본사에서 데이터가 입력,수정,삭제가 되어 지사에서 이용하는 형태
      - 부분 복제와 마찬가지로 데이터를 복제하는데 많은 시간이 소요되고 많은 부하가 발생되므로 배치를 통해 복제가 되도록 해야 함

분산 데이터베이스

- ❖ 분산 데이터베이스의 적용 기법
  - ✓ 테이블 요약(Summarization) 분산
    - ● 테이블 요약 분산은 지역간 또는 서버간에 데이터가 비슷하지만 서로 다른 유형으로 존재하는 경우 수행
    - ● 분석 요약
      - ◆ 동일한 테이블 구조를 가지고 있으면서 분산되어 있는 동일한 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식
      - ◆ 각 지사 별로 존재하는 요약 정보를 본사에서 통합하여 다시 전체에 대해서 요약 정보를 산출하는 분산 방법
    - ● 통합 요약
      - ◆ 분산되어 있는 다른 내용의 데이터를 이용하여 통합된 데이터를 산출하는 방식
      - ◆ 각 지사 별로 존재하는 다른 내용의 정보를 본사에 통합하여 다시 전체에 대해서 요약 정보를 산출하는 분산 방법

분산 데이터베이스

- 데이터 웨어하우스(Data Warehouse)
  - ✓ 많고 다양한 형태의 오퍼레이셔날 데이터베이스(Operational Database)가 운용되고 시간이 지날수록 그 데이터베이스의 크기가 커지게 되는데 데이터베이스 위에서 의사 결정(decision making) 등을 위해 사용되는 다양한 종류의 응용프로그램들은 그 질의(query) 수행이 원만하게 이루어지기 위하여 오퍼레이셔날 데이터베이스 위에 새로운 형태의 통합된 데이터 저장소(repository)가 필요했는데 이것이 데이터웨어하우스가 등장한 배경
  - ✓ 데이터웨어하우스는 조직이 통상적으로 운용하는 트랜잭션 프로세싱을 위한 일반적이고 다양한 종류의 데이터의 저장소가 아니며 의사 결정에 필요한 특정 주제(subject)의 데이터만을 가지고 그 외의 데이터는 포함하지 않음
  - ✓ 전사적으로 구축된 데이터웨어하우스로부터 특정 주제, 부서 중심으로 구축된 소규모 단일 주제의 데이터웨어하우스가 데이터 마트

분산 데이터베이스

- 데이터 마이닝
  - ✓ 데이터 마이닝은 대용량의 데이터에서 필요한 지식(Knowledge)을 얻고자 하는 과정 이라 간단히 정의할 수 있는데 데이터베이스에서 분야에서는 지식 발견(KDD : Knowledge Discovery in Databases)이라고도 불리며 그 응용 분야에 따라 비즈니스 인텔리전스, 지식 추출, 정보 분석 등의 용어로도 불리는데 여기서 발견하고자하는 지식은 뻔한 사실이 아니고(Non-Trivial) 데이터에 직접적으로 나타나지 않고 내포되었으며(Implicit) 기존에 발견되지 않은 잠재적으로 유용한 지식을 의미
  - ✓ 데이터 마이닝은 잠재적으로 폭 넓은 응용 분야를 가지고 있는데 고객 구매 성향 분석, 타겟 마케팅, 크로스-마켓팅 등의 마켓팅 분석 및 관리 분야, 위험 분석 및 관리 분야, 사기 상행위 발견 및 예측 분야, 텍스트 나 웹 등에서의 정보 발견, DNA 데이터 분석 및 의료정보학, 기타 스포츠, 과학 등을 망라하는데 데이터 마이닝 기술이 발전하면서 그 응용의 폭이 점점 더 커지고 있는 추세

테이블 관련 명령

❖ 테이블 생성 구문

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]테이블명(
    컬럼명1 타입 [CONSTRAINT 제약조건이름] 컬럼제약조건,
    컬럼명2 타입,
    컬럼명3 타입,
    ..........
     [CONSTRAINT 제약조건이름] 테이블 제약 조건)ENGINE=엔진명;
```

테이블 관련 명령

❖ MySQL에서 지원하는 데이터 형식의 종류

✓ 숫자 데이터 형식

데이터 형식	바이트 수	숫자 범위	설명
BIT(N)	N/8		1~64bit를 표현. b'0000' 형식으로 표현
TINYINT	1	-128~127	정수
★SMALLINT	2	-32,768~32,767	정수
MEDIUMINT	3	-8,388,608~8,388,607	정수
★INT INTEGER	4	약 -21억~+21억	정수
★BIGINT	8	약 -900경~+900경	정수
★FLOAT	4	-3.40E+38~-1.17E-38	소수점 아래 7자리까지 표현
★DOUBLE REAL	8	-1.22E-308~1.79E+308	소수점 아래 15자리까지 표현
★DECIMAL(m, [d]) NUMERIC(m, [d])	5~17	$-10^{38}+1$~$+10^{38}-1$	전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자형 예) decimal(5, 2)은 전체 자릿수를 5자리로 하되, 그 중 소수점 이하를 2자리로 하겠다는 의미

- 실수 저장 시 dec(M, D), fixed(M, D) 자료형도 사용 가능

테이블 관련 명령

❖ MySQL에서 지원하는 데이터 형식의 종류

✓ 문자 데이터 형식

데이터 형식		바이트 수	설명
★CHAR(n)		1~255	고정길이 문자형. n을 1부터 255까지 지정. character의 약자 그냥 CHAR만 쓰면 CHAR(1)과 동일
★VARCHAR(n)		1~65535	가변길이 문자형. n을 사용하면 1부터 65535까지 지정. Variable character의 약자
BINARY(n)		1~255	고정길이의 이진 데이터 값
VARBINARY(n)		1~255	가변길이의 이진 데이터 값
TEXT 형식	TINYTEXT	1~255	255 크기의 TEXT 데이터 값
	TEXT	1~65535	N 크기의 TEXT 데이터 값
	MEDIUMTEXT	1~16777215	16777215 크기의 TEXT 데이터 값
	★LONGTEXT	1~4294967295	최대 4GB 크기의 TEXT 데이터 값
BLOB 형식	TINYBLOB	1~255	255 크기의 BLOB 데이터 값
	BLOB	1~65535	N 크기의 BLOB 데이터 값
	MEDIUMBLOB	1~16777215	16777215 크기의 BLOB 데이터 값
	★LONGBLOB	1~4294967295	최대 4GB 크기의 BLOB 데이터 값
ENUM(값들…)		1 또는 2	최대 65535개의 열거형 데이터 값
SET(값들…)		1, 2, 3, 4, 8	최대 64개의 서로 다른 데이터 값

테이블 관련 명령

❖ MySQL에서 지원하는 데이터 형식의 종류

✓ 날짜 데이터 형식

DATE	1000-01-01 ~ 9999-12-31	YYYY-MM-DD
DATETIME	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS
TIMESTAMP	1970-01-01 ~ 2037년 임의 시간(1970-01-01 00:00:00 를 0으로 해서 1초단위로 표기)	
TIME	-838:59:59 ~ 838:59:59	
YEAR	901~2155	

✓ BOOL: true 또는 false

✓ JSON: 5.7.8 버전 이후에서 제공

✓ GEOMETRY: 공간 데이터 형식

테이블 관련 명령

- ❖ 테이블 생성 시 옵션
  - ✓ ENGINE
    - MyISAM
      - ◆ 조회에 유리
      - ◆ KEY INDEX를 지원하여 KEY로 정의된 칼럼의 값을 조회 조건으로 넣으면 INDEX로 빠르게 검색하는 기능을 소유
    - InnoDB
      - ◆ 트랜잭션 처리에 유리하지만 KEY를 이용한 검색에는 불리
  - ✓ AUTO_INCREMENT=초기값 을 추가하면 시퀀스의 초기 값을 설정할 수 있음
    - ALTER TABLE [테이블명] AUTO_INCREMENT=[시작하려는 값] 을 이용해서 초기값을 재설정 할 수 있음

테이블 관련 명령

- 테이블 생성 시 옵션
  - ✓ CHARACTER SET: 문자를 어떻게 표현할 지 여부
  - ✓ DEFAULT CHARSET 인코딩 방식 을 이용해서 기본적인 인코딩 방식 설정
    - UTF8: 한 글자를 저장할 때 1~3바이트를 사용하는데 웹의 표준이지만 가변 길이를 3바이트로 제한하기 때문에 4바이트가 필요한 일부 현대적인 문자(이모지)를 표현하지 못함
    - UTF8mb4 (Most Bytes 4): 한 글자당 1~4바이트를 사용하는데 UTF-8의 하위 호환성을 유지하면서 4바이트 영역의 문자까지 모두 수용할 수 있도록 확장된 버전으로 이모지를 표현할 수 있음

```
mysql> show character set;

+----------+---------------------------------+---------------------+
| Charset  | Description                     | Default collation   |
+----------+---------------------------------+---------------------+
| armscii8 | ARMSCII-8 Armenian              | armscii8_general_ci |
| ascii    | US ASCII                        | ascii_general_ci    |
| ujis     | EUC-JP Japanese                 | ujis_japanese_ci    |
| utf16    | UTF-16 Unicode                  | utf16_general_ci    |
| utf16le  | UTF-16LE Unicode                | utf16le_general_ci  |
| utf32    | UTF-32 Unicode                  | utf32_general_ci    |
| utf8     | UTF-8 Unicode                   | utf8_general_ci     |
| utf8mb4  | UTF-8 Unicode                   | utf8mb4_0900_ai_ci  |
+----------+---------------------------------+---------------------+
```

테이블 관련 명령

- 테이블 생성 시 옵션
  - ✓ Collation
    - 문자를 어떻게 정렬할 지 여부
      - 대소문자 구분 여부('A' 와 'a')
      - 악센트('O' 와 'Ò')
      - Locale-specific rules (e.g. 'A' vs. 'Å' vs. 'AA' in Danish and Norwegian)
      - Numeric characters (e.g. '2' vs. ' ii ')
      - Punctuation (e.g. 'blackbird' vs. 'black-bird')

```
mysql>  SHOW COLLATION;

+------------------------------+-----------+-----+---------+-----------+
| Collation                    | Charset   | Id  | Default | Compiled  |
+------------------------------+-----------+-----+---------+-----------+
| armscii8_bin                 | armscii8  |  64 |         | Yes       |
| armscii8_general_ci          | armscii8  |  32 | Yes     | Yes       |
| ascii_bin                    | ascii     |  65 |         | Yes       |
| ascii_general_ci             | ascii     |  11 | Yes     | Yes       |
| utf8mb4_0900_ai_ci           | utf8mb4   | 255 | Yes     | Yes       |
| utf8mb4_0900_as_ci           | utf8mb4   | 305 |         | Yes       |
| utf8mb4_0900_as_cs           | utf8mb4   | 278 |         | Yes       |
| utf8mb4_0900_bin             | utf8mb4   | 309 |         | Yes       |
| utf8mb4_bin                  | utf8mb4   |  46 |         | Yes       |
| utf8mb4_croatian_ci          | utf8mb4   | 245 |         | Yes       |
| utf8mb4_cs_0900_ai_ci        | utf8mb4   | 266 |         | Yes       |
| utf8mb4_cs_0900_as_cs        | utf8mb4   | 289 |         | Yes       |
```

테이블 관련 명령

❖ 연락처 테이블(contact) 만들기

```
CREATE TABLE contact (
num integer auto_increment primary key,
name varchar(20),
address char(100),
tel varchar(20),
email char(100),
birthday date
) ENGINE=MyISAM auto_increment=1 DEFAULT CHARSET=utf8;
```

테이블 관련 명령

❖ 테이블 수정

✓ 컬럼 추가

ALTER TABLE 테이블명 ADD 새로운컬럼명 컬럼타입 [first 또는 after 컬럼명];

- contact 테이블에 age 컬럼을 정수형으로 추가

```
ALTER TABLE contact ADD age int;

desc contact;
```

테이블 관련 명령

❖ 테이블 수정

✓ 컬럼 삭제

ALTER TABLE 테이블명 DROP 삭제할 컬럼명1, 삭제할 컬럼명2;

- 컬럼 삭제 – 제약조건이 설정된 경우 제약조건을 먼저 삭제
- contact 테이블에서 age 컬럼 삭제

```
ALTER TABLE contact DROP age;
DESC contact;
```

테이블 관련 명령

- ❖ 테이블 수정
  - ✓ 컬럼 변경
    - 기존 컬럼의 자료형의 이름과 자료형 변경: ALTER TABLE 테이블명 change 이전 컬럼명 새로운 컬럼명 컬럼 타입;
    - 기존 컬럼의 자료형 변경: ALTER TABLE 테이블명 modify 컬럼명 컬럼 타입;
    - NOT NULL 제약조건 수정도 modify 이용
    - contact 테이블 컬럼 중 tel char(20)을 phone int로 변경

      ALTER TABLE contact change tel phone int;

      desc contact

테이블 관련 명령

- ❖ 테이블 수정
  - ✓ 컬럼 순서 조정
    - ● 첫번째 위치로 이동

      ALTER TABLE 테이블명 MODIFY COLUMN 컬럼명 자료형 FIRST;
    - ● 특정 컬럼 뒤로 이동

      ALTER TABLE 테이블명 MODIFY COLUMN 컬럼명 자료형 AFTER 다른컬럼;

테이블 관련 명령

- 테이블 수정
  - ✓ 테이블 이름 수정

    ALTER TABLE 이전 테이블명 rename 새로운 테이블명;

테이블 관련 명령

- ❖ 테이블 삭제
  - ✓ 기본 형식
    - DROP TABLE 테이블명;
    - contact 테이블 삭제

      DROP TABLE contact;

      SHOW TABLES;

테이블 관련 명령

- ❖ 테이블의 모든 데이터 삭제
  - ✓ 기본 형식
    - TRUNCATE TABLE 테이블명;

테이블 관련 명령

- 테이블 압축
  - ✓ CREATE TABLE 다음에 ROW_FORMAT=COMPRESSED 추가하면 테이블을 압축해서 생성
  - ✓ 용량은 작아지는데 작업 시간은 길어짐

테이블 관련 명령

❖ 주석 설정

COMMENT ON TABLE 원본테이블 IS '주석';

제약조건

- 데이터 무결성(Integrity)
  - ✓ 데이터의 정확성과 일관성을 유지하고 보증하는 것
  - ✓ 적용 범위
    - 도메인 무결성(Domain Integrity): 컬럼 하나에 저장되는 원자적인 값을 점검하며 타입 지정, 널 허용 여부, 값의 범위 나 종류 체크, 기본값 등의 제약이 있음
    - 엔티티 무결성(Entity Integrity): 레코드 끼리 중복 값을 가지지 않도록 하기 위하여 유일한 식별자를 관리하는 것으로 기본키 와 유니크 제약이 있음
    - 참조 무결성(Referential Integrity): 테이블 간의 관계를 구성하는 키가 항상 유효하도록 관리하며 외래키 제약으로 관리하는데 하나의 테이블이 참조하는 정보가 다른 테이블에 반드시 존재해야 함

제약조건

- NULL 허용
  - ✓ NULL은 아무것도 입력되어 있지 않은 것이며 알 수 없거나 결정되지 않은 특수한 상태를 의미
  - ✓ 필드의 NULL 허용 속성은 NULL 상태가 존재하는지를 지정
  - ✓ 기본은 NULL을 허용하는 것이며 테이블을 만들 때 NOT NULL을 컬럼에 설정하면 NULL을 대입할 없음

제약조건

- NULL 허용
  - ✓ 테이블 생성 및 확인

```
CREATE TABLE tNullable
(
    name CHAR(10) NOT NULL,
    age INT
);

INSERT INTO tNullable (name, age) VALUES ('흥부', 36);
INSERT INTO tNullable (name) VALUES ('놀부');
INSERT INTO tNullable (age) VALUES (44); -- 에러
```

제약조건

- 체크
  - ✓ 체크 제약은 필드의 값 종류를 제한
  - ✓ 컬럼 선언문에 CHECK 키워드와 함께 컬럼 값으로 가능한 값을 조건문으로 지정

제약조건

❖ 체크

```
CREATE TABLE tCheckTest(
        gender CHAR(3) NULL CHECK(gender = '남' OR gender = '여'),
        grade INT NULL CHECK (grade >= 1 AND grade <= 3),
        origin CHAR(3) NULL CHECK(origin IN ('동','서','남','북')),
        name CHAR(10) NULL CHECK(name LIKE '김%')
);

INSERT INTO tCheckTest (gender) VALUES ('여');
INSERT INTO tCheckTest (grade) VALUES (1);
INSERT INTO tCheckTest (origin) VALUES ('동');
INSERT INTO tCheckTest (name) VALUES ('김좌진');

INSERT INTO tCheckTest (gender) VALUES ('노');
INSERT INTO tCheckTest (grade) VALUES (0);
INSERT INTO tCheckTest (origin) VALUES ('중');
INSERT INTO tCheckTest (name) VALUES ('청산리');
```

제약조건

- ❖ 체크
  - ✓ 연습문제
    - ● 직원 테이블의 각 필드에 제약 조건을 설정하여 부서는 영업부, 총무부, 인사과 중 하나만 성별은 남 아니면 여 로 제한하고 월급은 0 보다 크다는 조건을 설정

제약조건

- UNIQUE
  - ✓ 필드의 중복값을 방지하여 모든 필드가 고유한 값을 가지도록 강제
  - ✓ 기본키 와 의 차이점
    - 기본키는 NULL을 허용하지 않지만 유니크는 NULL을 허용
    - UNIQUE와 NOT NULL을 동시에 지정하면 기본키 와 유사해지지만 기본키는 테 이블 당 하나만 지정할 수 있지만 유니크는 개수에 상관없이 얼마든지 지정 가능

제약조건

- UNIQUE

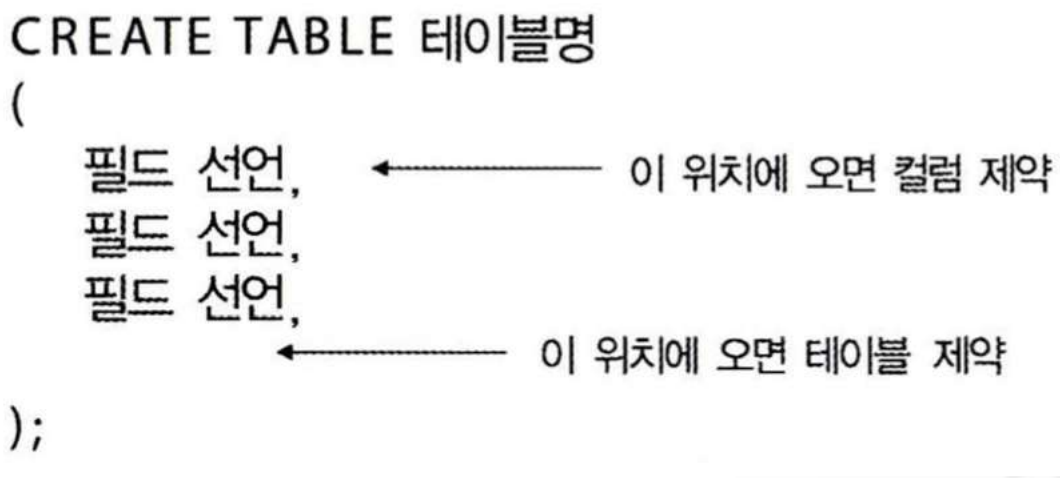
```
CREATE TABLE tCityUnique
(
        name CHAR(10) PRIMARY KEY,
        area INT NULL ,
        popu INT NULL,
        metro CHAR(1) NOT NULL,
        region CHAR(6) NOT NULL,
        CONSTRAINT Unique_tCity_area_popu UNIQUE(area, popu)
);
```

제약조건

- Primary Key
  - ✓ Key
    - 식별자 – Key
      - 키는 값이 꼭 있어야 하며 구분을 위한 고유 값을 가져야 함
      - NULL 이거나 중복되면 안됨
    - 후보키 – Candidate Key
      - 슈퍼 키 중 더 이상 줄일 수 없는(irreducible) 형태를 가진 키
    - 기본키 – Primary Key
      - 후보키 중에서 선정한 키
      - 고려 사항
        - 대표성: 레코드를 상징하는 값이어야 함
        - 자주 참조하는 속성: 기본키에는 기본적으로 인덱스가 생성되어 검색 효율이 좋아야 함
        - 가급적 짧은 속성: 테이블간의 연결 고리가 되므로 비교 속도가 빨라야 함

제약조건

- ❖ Primary Key
  - ✓ 기본키 지정
    - ● 컬럼 제약 과 테이블 제약 두 가지 방법



- ◆ 컬럼 기본키 제약: [CONSTRAINT 이름] PRIMARY KEY
- ◆ 테이블 기본키 제약: [CONSTRAINT 이름] PRIMARY KEY(대상필드)

제약조건

- Primary Key
  - ✓ 기본키 지정
    - PRIMARY KEY 제약은 NOT NULL 속성을 겸하기 때문에 NOT NULL을 붙일 필요 없음
    - 제약조건 이름은 나중에 편집이나 삭제 등을 편리하게 하기 위해서 사용
    - 2개 이상의 컬럼의 조합으로 기본키를 설정할 수 있는데 이 경우에는 테이블 제약조건으로 설정해야 합
    - 기본키 설정

```
CREATE TABLE tCityPK
(
        name CHAR(10),
        area INT NULL ,
        popu INT NULL ,
        metro CHAR(1) NOT NULL,
        region CHAR(6) NOT NULL,
        CONSTRAINT PK_tCity_name PRIMARY KEY(name)
);
```

제약조건

- Primary Key
  - ✓ 기본키 지정

```
CREATE TABLE tCityCompoKey(
        name CHAR(10) PRIMARY KEY,
        region CHAR(6) PRIMARY KEY,
        area INT NULL ,
        popu INT NULL ,
        metro CHAR(1) NOT NULL
);

CREATE TABLE tCityCompoKey(
        name CHAR(10) NOT NULL,
        region CHAR(6) NOT NULL,
        area INT NULL ,
        popu INT NULL ,
        metro CHAR(1) NOT NULL,
        CONSTRAINT PK_tCity_name_region PRIMARY KEY (name, region)
);
```

제약조건

- Primary Key
  - ✓ 기본키 지정

```
INSERT INTO tCityCompoKey VALUES ('광주', '전라', 123, 456, 'y');
INSERT INTO tCityCompoKey VALUES ('광주', '경기', 123, 456, 'n');
```

제약조건

- Primary Key
  - ✓ 연습문제
    - 이름과 부서와 성별을 복합키로 지정하여 새로운 직원 테이블 tStaffCompoKey를 생성

제약조건

- 참조 무결성
  - ✓ 2개의 테이블을 생성하고 샘플 데이터 생성

```
CREATE TABLE tEmployee(
name CHAR(10) PRIMARY KEY,
salary INT NOT NULL,
addr VARCHAR(30) NOT NULL
);

INSERT INTO tEmployee VALUES ('아이린', 650, '대구시');
INSERT INTO tEmployee VALUES ('슬기', 480, '안산시');
INSERT INTO tEmployee VALUES ('웬디', 625, '서울시');
```

제약조건

- 참조 무결성
  - ✓ 2개의 테이블을 생성하고 샘플 데이터 생성

```
CREATE TABLE tProject
(
projectID INT PRIMARY KEY,
employee CHAR(10) NOT NULL,
project VARCHAR(30) NOT NULL,
cost INT
);

INSERT INTO tProject VALUES (1, '아이린', '홍콩 수출건', 800);
INSERT INTO tProject VALUES (2, '아이린', 'TV 광고건', 3400);
INSERT INTO tProject VALUES (3, '아이린', '매출분석건', 200);
INSERT INTO tProject VALUES (4, '슬기', '경영 혁신안 작성', 120);
INSERT INTO tProject VALUES (5, '슬기', '대리점 계획', 85);
INSERT INTO tProject VALUES (6, '웬디', '노조 협상건', 24);
```

제약조건

- 참조 무결성
  - ✓ 외래키를 설정하지 않은 상태에서의 작업

```
INSERT INTO tProject VALUES (7, '조이', '원자재 매입', 900);

DELETE FROM tEmployee WHERE name = '아이린';
```

제약조건

- 참조 무결성
  - ✓ 외래키 설정

```
DROP TABLE tProject;
DROP TABLE tEmployee;

CREATE TABLE tEmployee
(
name CHAR(10) PRIMARY KEY,
salary INT NOT NULL,
addr VARCHAR(30) NOT NULL
);

INSERT INTO tEmployee VALUES ('아이린', 650, '대구시');
INSERT INTO tEmployee VALUES ('슬기', 480, '안산시');
INSERT INTO tEmployee VALUES ('웬디', 625, '서울시');
```

제약조건

- 참조 무결성
  - ✓ 외래키 설정

```
CREATE TABLE tProject(
projectID INT PRIMARY KEY,
employee CHAR(10) NOT NULL,
project VARCHAR(30) NOT NULL,
cost INT,
CONSTRAINT FK_emp FOREIGN KEY(employee) REFERENCES tEmployee(name)
);
```

제약조건

- 참조 무결성
  - ✓ 외래키 설정

```
INSERT INTO tProject VALUES (1, '아이린', '홍콩 수출건', 800);
INSERT INTO tProject VALUES (2, '아이린', 'TV 광고건', 3400);
INSERT INTO tProject VALUES (3, '아이린', '매출분석건', 200);
INSERT INTO tProject VALUES (4, '슬기', '경영 혁신안 작성', 120);
INSERT INTO tProject VALUES (5, '슬기', '대리점 계획', 85);
INSERT INTO tProject VALUES (6, '웬디', '노조 협상건', 24);
```

제약조건

- 참조 무결성
  - ✓ 외래키를 설정한 동일한 작업 수행 – 에러 발생

```
INSERT INTO tProject VALUES (7, '조이', '원자재 매입', 900);

DELETE FROM tEmployee WHERE name = '아이린';
```

제약조건

- 참조 무결성
  - ✓ 외래키를 설정한 동일한 작업 수행

```
INSERT INTO tEmployee VALUES ('조이', 330, '제주');
INSERT INTO tProject VALUES (7, '조이', '원자재 매입', 900);

DELETE FROM tProject WHERE employee = '아이린';
DELETE FROM tEmployee WHERE name = '아이린';
```

제약조건

- 참조 무결성
  - ✓ 외래키로 참조되는 테이블 삭제 불가

```
DROP TABLE tEmployee;
```

제약조건

- ❖ 참조 무결성
  - ✓ 참조 무결성 옵션 – 외래키 설정 뒤에 작성
    - ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
    - ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

제약조건

- 참조 무결성
  - ✓ 참조 무결성 옵션 – 테이블 수정

```
DROP TABLE tProject;
DROP TABLE tEmployee;

CREATE TABLE tEmployee
(
 name CHAR(10) PRIMARY KEY,
 salary INT NOT NULL,
 addr VARCHAR(30) NOT NULL
);

INSERT INTO tEmployee VALUES ('아이린', 650, '대구시');
INSERT INTO tEmployee VALUES ('슬기', 480, '안산시');
INSERT INTO tEmployee VALUES ('웬디', 625, '서울시');
```

제약조건

- 참조 무결성
  - ✓ 참조 무결성 옵션 – 테이블 수정

```
CREATE TABLE tProject(
 projectID INT PRIMARY KEY,
 employee CHAR(10) NOT NULL,
 project VARCHAR(30) NOT NULL,
 cost INT,
 CONSTRAINT FK_emp FOREIGN KEY(employee) REFERENCES tEmployee(name)
 ON DELETE CASCADE ON UPDATE CASCADE
);

INSERT INTO tProject VALUES (1, '아이린', '홍콩 수출건', 800);
INSERT INTO tProject VALUES (2, '아이린', 'TV 광고건', 3400);
INSERT INTO tProject VALUES (3, '아이린', '매출분석건', 200);
INSERT INTO tProject VALUES (4, '슬기', '경영 혁신안 작성', 120);
INSERT INTO tProject VALUES (5, '슬기', '대리점 계획', 85);
INSERT INTO tProject VALUES (6, '웬디', '노조 협상건', 24);
```

제약조건

- 참조 무결성
  - ✓ 참조 무결성 옵션

```
DELETE FROM tEmployee WHERE name = '아이린';

SELECT * FROM tEmployee;

SELECT * FROM tProject;

UPDATE tEmployee
SET name = '조이', salary=330, addr='제주'
WHERE Name = '웬디';

SELECT * FROM tEmployee;

SELECT * FROM tProject;
```

제약조건

- 제약조건 수정
  - ✓ 제약 조건 확인

    SELECT *

    FROM information_schema.table_constraints;
  - ✓ 제약 조건 수정

    ALTER TABLE 테이블이름 MODIFY 컬럼이름 자료형 제약조건;
  - ✓ 제약 조건 추가

    ALTER TABLE 테이블이름 ADD 제약조건(컬럼이름)
  - ✓ 제약 조건 삭제

    ALTER TABLE 테이블이름 DROP CONSTRAINT 제약조건이름

제약조건

- 기본값
  - ✓ 기본값은 필드 값을 지정하지 않을 때 자동으로 입력할 값
  - ✓ 수치형은 0이 적당하고 문자열은 비워 두거나 N/A 등을 많이 사용
  - ✓ DEFAULT 키워드와 함께 디폴트 값을 지정
  - ✓ DEFAULT 값을 입력하고자 할 때는 컬럼을 제외하고 삽입하거나 DEFAULT 로 설정

제약조건

❖ 기본값

```
CREATE TABLE tCityDefault(
        name CHAR(10) PRIMARY KEY,
        area INT NULL ,
        popu INT NULL ,
        metro CHAR(1) DEFAULT 'n' NOT NULL,
        region CHAR(6) NOT NULL
);

INSERT INTO tCityDefault (name, area, popu, region) VALUES ('진주', 712, 34, '경상');
INSERT INTO tCityDefault (name, area, popu, metro, region) VALUES ('인천', 1063, 295, 'y',
'경기');
```

제약조건

❖ 기본값

```
INSERT INTO tCityDefault VALUES ('강릉', 1111, 22, '강원');                -- 에러
INSERT INTO tCityDefault VALUES ('강릉', 1111, 22, DEFAULT, '강원'); -- 정상 실행

UPDATE tCityDefault SET metro = DEFAULT WHERE name = '인천'

SELECT *
FROM tCityDefault;
```

제약조건

- ❖ 기본값
  - ✓ 연습문제
    - ● 직원 테이블의 각 필드에 기본값을 적용하여 tStaffDefault 테이블을 생성하는데 부서는 영업부, 직급은 수습, 초봉은 280, 성취도는 1.0의 기본 값을 적용

제약조건

- AUTO_INCREMENT
  - ✓ 필드 선언문에 AUTO_INCREMENT라고 선언하면 자동 증가하는 일련번호가 매겨짐
  - ✓ 테이블에 일련번호 설정 및 확인

```
CREATE TABLE tSale
(
        saleno INT AUTO_INCREMENT PRIMARY KEY,
        customer NCHAR(10),
        product NCHAR(30)
);

INSERT INTO tSale (customer, product)
VALUES ('단군', '지팡이');

INSERT INTO tSale (customer, product)
VALUES ('고주몽', '고등어');

SELECT *
FROM tSale;
```

제약조건

- AUTO_INCREMENT
  - ✓ 삭제 후 추가

```
DELETE FROM tSale
WHERE saleno = 2;

INSERT INTO tSale (customer, product)
VALUES ('박혁거세', '계란');

SELECT *
FROM tSale;

INSERT INTO tSale (saleno, customer, product)
VALUES (2, '고주몽', '고등어');
```

제약조건

- AUTO_INCREMENT
  - ✓ 일련번호 수정

```
ALTER TABLE tSale AUTO_INCREMENT = 100;

INSERT INTO tSale (customer, product)
VALUES ('왕건', '너구리');

UPDATE tSale
SET product = '짜파게티'
WHERE saleno = LAST_INSERT_ID();

SELECT *
FROM tSale;
```

데이터베이스 모델링 실습

모델링 실습 1

❖ 고객이 방문한 내역 기록

고객 방문 기록

고객 이름	출생연도	주소	연락처	구매한 물건	단가(천 원)	수량
이승기	1987	서울	011-111-1111			
김범수	1979	경남	011-222-2222	운동화	30	2
김범수	1979	경남	011-222-2222	노트북	1000	1
김경호	1971	전남	019-333-3333			
조용필	1950	경기	011-444-4444	모니터	200	1
바비킴	1973	서울	010-000-0000	모니터	200	5
윤종신	1969	경남	안 남김			
김범수	1979	경남	011-222-2222	청바지	50	3
임재범	1963	서울	016-666-6666			
바비킴	1973	서울	010-000-0000	메모리	80	10
성시경	1979	경남	안 남김	책	15	5
은지원	1978	경북	011-888-8888	책	15	2
임재범	1963	서울	016-666-6666			
은지원	1978	경북	011-888-8888	청바지	50	1
바비킴	1973	서울	010-000-0000	운동화	30	2
은지원	1978	경북	011-888-8888			
은지원	1978	경북	011-888-8888	책	15	1
바비킴	1973	서울	010-000-0000	운동화	30	2
조관우	1965	경기	018-999-9999			

모델링 실습 1

❖ 기록된 내용에서 물건을 구매한 적이 없는 고객을 위쪽으로 정렬

고객 이름	출생연도	주소	연락처	구매한 물건	단가(천 원)	수량
이승기	1987	서울	011-111-1111			
김경호	1971	전남	019-333-3333			
윤종신	1969	경남	안 남김			
임재범	1963	서울	016-666-6666			
임재범	1963	서울	016-666-6666			
은지원	1978	경북	011-888-8888			
조관우	1965	경기	018-999-9999			
김범수	1979	경남	011-222-2222	운동화	30	2
김범수	1979	경남	011-222-2222	노트북	1000	1
조용필	1950	경기	011-444-4444	모니터	200	1
바비킴	1973	서울	010-000-0000	모니터	200	5
김범수	1979	경남	011-222-2222	청바지	50	3
바비킴	1973	서울	010-000-0000	메모리	80	10
성시경	1979	경남	안 남김	책	15	5
은지원	1978	경북	011-888-8888	책	15	2
은지원	1978	경북	011-888-8888	청바지	50	1
바비킴	1973	서울	010-000-0000	운동화	30	2
은지원	1978	경북	011-888-8888	책	15	1
바비킴	1973	서울	010-000-0000	운동화	30	2

모델링 실습 1

❖ L자형 테이블을 빈칸이 있는 곳과 없는 곳으로 분리

고객 테이블

고객 이름	출생연도	주소	연락처
이승기	1987	서울	011-111-1111
김경호	1971	전남	019-333-3333
윤종신	1969	경남	안 남김
임재범	1963	서울	016-666-6666
임재범	1963	서울	016-666-6666
은지원	1978	경북	011-888-8888
조관우	1965	경기	018-999-9999
김범수	1979	경남	011-222-2222
김범수	1979	경남	011-222-2222
조용필	1950	경기	011-444-4444
바비킴	1973	서울	010-000-0000
김범수	1979	경남	011-222-2222
바비킴	1973	서울	010-000-0000
성시경	1979	경남	안 남김
은지원	1978	경북	011-888-8888
은지원	1978	경북	011-888-8888
바비킴	1973	서울	010-000-0000
은지원	1978	경북	011-888-8888
바비킴	1973	서울	010-000-0000

구매 테이블

구매한 물건	단가(천 원)	수량
운동화	30	2
노트북	1000	1
모니터	200	1
모니터	200	5
청바지	50	3
메모리	80	10
책	15	5
책	15	2
청바지	50	1
운동화	30	2
책	15	1
운동화	30	2

모델링 실습 1

❖ 고객 테이블의 중복을 제거 – 기본키를 설정

고객 테이블

고객 아름	출생연도	주소	연락처
이승기	1987	서울	011-111-1111
김경호	1971	전남	019-333-3333
윤종신	1969	경남	안 남김
임재범	1963	서울	016-666-6666
은지원	1978	경북	011-888-8888
조관우	1965	경기	018-999-9999
김범수	1979	경남	011-222-2222
조용필	1950	경기	011-444-4444
바비킴	1973	서울	010-000-0000
성시경	1979	경남	안 남김

모델링 실습 1

❖ 구매 테이블만 보니 누가 구매한 것인지를 알 수가 없으므로 구매 테이블의 앞에 회원을 구분할 수 있는 회원의 기본 키로 설정된 이름을 추가

구매 테이블

고객 이름	구매한 물건	단가(천 원)	수량
김범수	운동화	30	2
김범수	노트북	1000	1
조용필	모니터	200	1
바비킴	모니터	200	5
김범수	청바지	50	3
바비킴	메모리	80	10
성시경	책	15	5
은지원	책	15	2
은지원	청바지	50	1
바비킴	운동화	30	2
은지원	책	15	1
바비킴	운동화	30	2

모델링 실습 1

❖ 관계 설정

고객 이름	출생연도	주소	연락처
이승기	1987	서울	011-111-1111
김경호	1971	전남	019-333-3333
윤종신	1969	경남	안 남김
임재범	1963	서울	016-666-6666
은지원	1978	경북	011-888-8888
조관우	1965	경기	018-999-9999
김범수	1979	경남	011-222-2222
조용필	1950	경기	011-444-4444
바비킴	1973	서울	010-000-0000
성시경	1979	경남	안 남김

고객 이름	구매한 물건	단가(천 원)	수량
김범수	운동화	30	2
김범수	노트북	1000	1
조용필	모니터	200	1
바비킴	모니터	200	5
김범수	청바지	50	3
바비킴	메모리	80	10
성시경	책	15	5
은지원	책	15	2
은지원	청바지	50	1
바비킴	운동화	30	2
은지원	책	15	1
바비킴	운동화	30	2

모델링 실습 1

❖ 구현

테이블 이름	열 이름	데이터 형식	Null 허용	기타
고객 테이블 (userTBL)	고객 이름(userName)	문자(최대 3글자)	X	PK
	출생연도(birthYear)	숫자(정수)	X	
	주소(addr)	문자(최대 2글자)	X	
	연락처(mobile)	문자(최대 12글자)	O	
구매 테이블 (buyTBL)	고객 이름(userName)	문자(최대 3글자)	X	FK
	구매한 물건(prodName)	문자(최대 3글자)	X	
	단가(price)	숫자(정수)	X	
	수량(amount)	숫자(정수)	X	

모델링 실습 2

- 쇼핑몰 운영 규칙
  - ✓ 처음 가입시 준회원으로 등록하며 매출이 10000원을 넘을 경우 정회원이 되고 50000원이 되면 우수회원으로 등록
  - ✓ 정회원 이상은 배송비가 무료이며 우수 회원은 10% 추가 할인
  - ✓ 간편한 결제를 위해 예치금을 맡기고 예치금으로만 쇼핑
  - ✓ 현금이나 신용카드로는 구매할 수 없음
  - ✓ 가입 즉시 축하금 1000원을 예치금으로 지급하고 예치금 10000원 적립 시 12000원 입금으로 처리
  - ✓ 상품은 몇 가지 유형으로 나누고 유형에 따라 할인율, 배송비, 반품여부 가 다름
  - ✓ 유형은 고정이며 편집 기능은 제공하지 않음
  - ✓ 미성년자(19세 미만)에게는 성인용품을 판매하지 않음

모델링 실습 2

- Entity 추출
  - ✓ 모델링의 첫 단계는 업무 분석 결과를 바탕으로 Entity를 추출하는 것
  - ✓ 쇼핑을 하려면 사는 사람이 있어야 하고 팔 물건이 있어야 하니 회원과 상품 두 명사가 기본적인 Entity
  - ✓ 회원은 상품을 구입하는 식으로 관계를 맺으므로 이 관계를 주문 Entity로 정의
  - ✓ 주문이 없으면 회원과 상품이 관계를 맺을 수 없으므로 주문은 쇼핑몰의 주요 Entity
  - ✓ 예치금과 고객의 나이. 주문 수량 등은 Entity의 속성일 뿐 Entity는 아님
  - ✓ Entity 추출 결과 회원, 상품, 주문이라는 3개의 Entity가 필요함을 파악

모델링 실습 2

❖ 회원 테이블

✓ 회원의 속성을 나열해 보고 이 중 쇼핑몰에 어떤 정보가 필요한지 생각

✓ 회원은 사람이므로 다음과 같은 속성을 가지는데 이 중 업무에 사용할 만 한 속성만 추려내야 함

이름, 전화번호, 주소, 이메일, 나이, 키, 몸무게, 고향, 시력, 혈액형, 좋아하는 연예인,

- 이름은 기본 정보라 당연히 필요하고 배송을 위한 주소가 있어야 함
- 배송 안내를 위해 이메일이나 전화번호가 필요
- 업무 규칙에 미성년자 어쩌고 하는 조항이 있어 나이는 필요하지만 키나 몸무게, 고향 등은 쇼핑몰 업무와는 하등의 상관이 없어 제외

✓ 여기까지 회원 자신이 가진 고유한 속성을 선정하였는데 이 외에 업무에 필요한 인위적인 속성을 추가하는데 업무 규칙에 회원의 등급을 관리하는 항목이 있어 각 회원은 등급을 가져야 하고 예치금은 회원별로 관리되므로 회원의 속성으로 표현하는 것이 논리상 합당

✓ 다음은 식별을 위한 기본키를 선정하는데 이름은 동명이인 문제로 기본키로 잘 쓰지 않으며 전화 번호나 주소는 한 집에 사는 두 회원(예를 들어 고시원이나 하숙생)이 따로 가입할 수 있어 검색 대상일 수는 있어도 기본키로는 부적합한데 이럴 때는 별도의 ID를 쓰는 것이 보편적

✓ 여기까지 모델링한 결과를 문서로 정리하여 구현 단계에서 실제 테이블로 생성

모델링 실습 2

❖ 회원 테이블

```
-- 회원 테이블
CREATE TABLE tMember
(
    member VARCHAR(20) PRIMARY KEY,          -- 아이디
    age INT NOT NULL,                        -- 나이
    email VARCHAR(30) NOT NULL,              -- 이메일
    addr VARCHAR(50) NOT NULL,               -- 주소
    money INT DEFAULT 1000 NOT NULL,         -- 예치금
    grade INT DEFAULT 1 NOT NULL,            -- 고객등급. 1=준회원, 2=정회원, 3=우수
     회원
    remark VARCHAR(100) NULL                 -- 메모 사항
);

-- 회원 데이터
INSERT INTO tMember VALUES ('춘향',16,'1004@naver.com','전남 남원시',20000, 2, '');
INSERT INTO tMember VALUES ('이도령',18,'wolf@gmail.com','서울 신사동',150000, 3, '');
INSERT INTO tMember VALUES ('향단',25,'candy@daum.net','전남 남원시',5000, 2, '');
INSERT INTO tMember VALUES ('방자',28,'devlin@ssang.co.kr','서울 개포동',1000, 1, '요주
    의 고객');
```

모델링 실습 2

- 상품 테이블
  - ✓ 상품명이 필요하고 제조사, 정가 등의 기본적인 정보도 있어야 함
  - ✓ 재고 정보는 상품 자체의 속성은 아니지만 업무상 필요
  - ✓ 잘 팔리는 제품은 재고를 넉넉히 확보해 놔야 하며 인기 없는 상품은 한 두개 만 있어도 됨
  - ✓ 개별 상품 별로 레코드를 정의할 필요는 없고 같은 상품은 레코드 하나로 정의하고 재고 수만 기록
  - ✓ 고객 입장에서는 남은 재고 중 어떤 상품을 구입 하나 동일
  - ✓ 상품의 또 다른 속성으로 할인율과 배송비, 반품 여부 등이 있음
  - ✓ 식품은 포장비가 많이 들어 배송비가 비싸며 할인 여력이 없고 부패 위험이 있어 반품을 받지 않음
  - ✓ 전자 제품은 마진이 높아 할인을 왕창 해 줘도 되고 반품도 받아 줌
  - ✓ 이런 규칙은 개별 상품마다 다른 것이 아니라 유형 별로 결정
  - ✓ 기본키인 상품에 종속적이지 않고 일반 필드인 유형에 종속적

모델링 실습 2

❖ 상품 테이블

✓ 1차 모델링

상품	제조사	재고	정가	유형	할인율	배송비	반품여부
노트북	샘성	3	820000	가전	20	2500	y
모니터	알지	1	450000	가전	20	2500	y
사과	문경농원	24	16000	식품	0	3000	n
대추	보은농원	19	15000	식품	0	3000	n
전자담배	T&G	4	7000	성인	5	1000	n
청바지	방방	99	21000	패션	10	2000	y

모델링 실습 2

- 상품 테이블
  - ✓ 1차 모델링한 위 테이블은 기본키가 아닌 일반 필드끼리 독립적이어야 한다는 제 3 정규화 규칙을 위반
  - ✓ 할인율, 배송비, 반품 여부가 계속 중복되어 노트북과 모니터, 사과와 대추의 뒷부분이 같음
  - ✓ 정규화하여 상품과 유형 테이블로 분리
  - ✓ 상품 Entity에 직접 종속되지 않는 할인율, 배송비, 반품 여부는 제외
  - ✓ 유형과 관련된 정보는 별도의 유형 테이블로 분리
  - ✓ 상품명이 중복되는 경우도 가끔 있어 ID 필드가 있으면 좋지만 논리를 간단히 하기 위해 상품의 이름인 item을 PK로 지정
  - ✓ 담배 회사는 T&G가 맞지만 오라클이 &를 특수 기호로 쓰고 있어 TNG로 작성
  - ✓ 상품의 분류는 category 필드로 지정하며 유형별 상세 정보를 구할 수 있는 외래키

모델링 실습 2

❖ 상품 테이블

상품	제조사	재고	정가	유형
노트북	샘성	3	820000	가전
모니터	알지	1	450000	가전
사과	문경농원	24	16000	식품
대추	보은농원	19	15000	식품
전자담배	T&G	4	7000	성인
청바지	방방	99	21000	패션

유형	할인율	배송비	반품여부
가전	20	2500	y
식품	0	3000	n
성인	5	1000	n
패션	10	2000	y

모델링 실습 2

❖ 상품 테이블

```
-- 상품 분류 테이블
CREATE TABLE tCategory
(
   category VARCHAR(10) PRIMARY KEY,          -- 분류명
   discount INT NOT NULL,                     -- 할인율
   delivery INT NOT NULL,                     -- 배송비
   takeback CHAR(1)                  -- 반품 가능성
);

-- 분류 데이터
INSERT INTO tCategory (category, discount, delivery, takeback) VALUES ('식품', 0, 3000,
    'n');
INSERT INTO tCategory (category, discount, delivery, takeback) VALUES ('패션', 10, 2000,
    'y');
INSERT INTO tCategory (category, discount, delivery, takeback) VALUES ('가전', 20, 2500,
    'y');
INSERT INTO tCategory (category, discount, delivery, takeback) VALUES ('성인', 5, 1000,
    'n');
```

모델링 실습 2

❖ 상품 테이블

```
-- 상품 테이블
CREATE TABLE tItem
(
    item VARCHAR(20) PRIMARY KEY,                -- 상품명
    company VARCHAR(20) NULL,                    -- 제조사
    num INT NOT NULL,                            -- 재고
    price INT NOT NULL,                  -- 정가
    category VARCHAR(10) NOT NULL,               -- 분류
    CONSTRAINT item_fk FOREIGN KEY(category) REFERENCES tCategory(category)
);
```

모델링 실습 2

❖ 상품 테이블

```
-- 상품 데이터
INSERT INTO tItem (item,company,num,price,category) VALUES ('노트북', '샘성', 3, 820000,
    '가전');
INSERT INTO tItem (item,company,num,price,category) VALUES ('청바지', '방방', 80, 32000,
    '패션');
INSERT INTO tItem (item,company,num,price,category) VALUES ('사과', '문경농원', 24,
    16000, '식품');
INSERT INTO tItem (item,company,num,price,category) VALUES ('대추', '보은농원', 19,
    15000, '식품');
INSERT INTO tItem (item,company,num,price,category) VALUES ('전자담배', 'TNG', 4,
    70000, '성인');
INSERT INTO tItem (item,company,num,price,category) VALUES ('마우스', '논리텍', 3,
    90000, '가전');
```

모델링 실습 2

- 주문 테이블
  - ✓ 주문은 회원이 상품을 구입하는 동작인데 이 둘은 전형적인 다:다 관계
  - ✓ 다대다 관계 이므로 상품, 회원 두 테이블만으로는 주문 관계를 표현 할 수 없으며 중간에 두 Entity의 관계를 1:다로 표현하는 주문 Entity가 필요
  - ✓ 주문은 누가 언제 어떤 상품을 구입했는지 모두 포함해야 하며 제대로 배송했는지 상태를 기록하는 속성도 있어야 하고 메모 사항도 남길 수 있어야 함

모델링 실습 2

❖ 주문 테이블

```
CREATE TABLE tOrder
(
  orderID INT AUTO_INCREMENT PRIMARY KEY,  -- 주문 번호
  member VARCHAR(20) NOT NULL REFERENCES tMember(member),          -- 주문
    자
  item VARCHAR(20) NOT NULL REFERENCES tItem(item),                -- 상품
  orderDate DATE DEFAULT SYSDATE() NOT NULL,          -- 주문 날자
  num INT NOT NULL,                                   -- 개수
  status INT DEFAULT 1 NOT NULL,                      -- 1:주문, 2:배송중, 3:배송완료,
    4:반품
  remark VARCHAR(1000) NULL                           -- 메모 사항
);
```

모델링 실습 2

❖ 주문 테이블

```
-- 주문 데이터
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('춘향','청바지','2019-12-3',3,2);
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('향단','대추','2019-12-4',10,1);
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('방자','전자담배','2019-12-2',4,1);
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('향단','사과','2019-12-5',5,2);
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('이도령','노트북','2019-12-5',2,1);
INSERT INTO tOrder (member,item,orderDate,num,status) VALUES ('방자','노트북','2019-12-1',1,3);
```