



Part.1_Conceptual Data Modeling

Introduction to Database Systems

Data Modeling (Conceptual)

국민대학교

김 남 규

데이터 모델링이란?

데이터베이스를 설계하는 일련의 과정

크게 컨셉철모델링 로직컬모델링 두 단계로 구분하여 수행된다.

그중에서 이번 시간에는 컨셉철모델링에 대해 학습

로직컬 모델링은 다음 강좌에서 다룰 예정

Database Applications

● 모든 서비스에 DB가 필요한가?

- (1) 입금, 출금, 이체 등 은행 거래
- (2) 호텔 객실의 예약
- (3) 신호등의 램프 제어
- (4) 온라인 쇼핑몰에서의 물품 구매
- (5) 전자식 개폐장치의 비밀번호 관리

● DB 시스템의 특성

- 최초 적재>Loading) → 이벤트 발생에 따른 잦은 변경(Interaction)
- 대용량의 데이터를 다룸
 - 사용자들이 원하는 순간 데이터에 접근하기 위해서는?
→ 대용량의 데이터가 체계적으로 조직화되어 있어야 함

데이터모델링을 본격적으로 다루기전에 용어에 대해 알아보자

빅데이터는 알아도 빅데이터의 조상격인 데이터베이스에 대해서는 들어보지 못한 경우 대다수

데이터베이스란 말그래도 데이터들의 저장소를 나타냄

데이터베이스의 DB를 따서 DB라고 간단히 줄여서 사용

데이터베이스를 사용하지 않는 시스템을 알고있고 사용하는 시스템을 알고있다면 비교하면 편하지만

지금 거의 모든 시스템에 컴퓨터시스템에 전기가 들어가있는 모든 시스템에 DB가 들어가 있다고해도 과언이 아님

DB가 어떤 역할을 하고 어떤 서비스에 들어가야하는지를 알기가 역설적으로 더 어려움

→ DB는 데이터가 많을 때 효율적으로 관리하기 위해 필요한 것

ex) * 모든 서비스에 DB가 필요한가? (1,2,4)

(1),(2) 데이터베이스 필요 : 기록을 남겨야하기때문에 데이터 필요

(3) 신호등에도 데이터가 있을까? 빨간색, 좌회전, 초록색, 주황색 이런 색깔들이 있다. 방금 전에 무슨 색이었는지 알아야 상태를 기억하고 있어야 다음 상태를 결정할 수 있기 때문에

상황에 대한 데이터가 들어가지 않을까 한다.

하지만 사용자가 있는 것이 아니고 반도체 칩 회로 안에 알고리즘으로 저장이 되어 있어서 상태를 변하게끔 프로그래밍이 되어 있는 것이지 수많은 방대한 데이터가 필요한 것이 아니다. 그래서 회로로 해결을 하는 것 DB가 필요하지 않다.

(4) 당연히 데이터베이스 필요. 지불내역 기록이 남아있어야 된다. 과거 구매내역도 볼 수 있어야 하기 때문

(5) 전자식 개폐장치 : 디지털 도어락 → 비밀번호를 눌렀을 때 찐 비밀번호를 제외한 나머지는 열리지 않아야 하기 때문에 비밀번호를 기억하고 있어야 해서 데이터를 기록하고 있어야 한다. 하지만 데이터가 필요하다고 해서 기억하고 저장해야할게 필요하다고 해서 모든 시스템에 DB가 필요한 것은 아니다. 숫자를 관리하면 되는 거고, 한 사람이 접속하고 네 자리 여섯 자리 코드를 집어넣으면 되기 때문에 방대한 데이터를 효율적으로 관리한다는 DB의 목적과는 다름 따라서 DB 필요 X .

DB는 많은 데이터를 다루고 있을 때만 필요하게 된다.

많은 사용자들이 동시에 접속하는 상황을 가정하게 된다. 즉, 대용량의 데이터에 대해서 많은 사용자들이 동시에 사용하는 경우에 우리가 데이터베이스를 사용하게 된다.

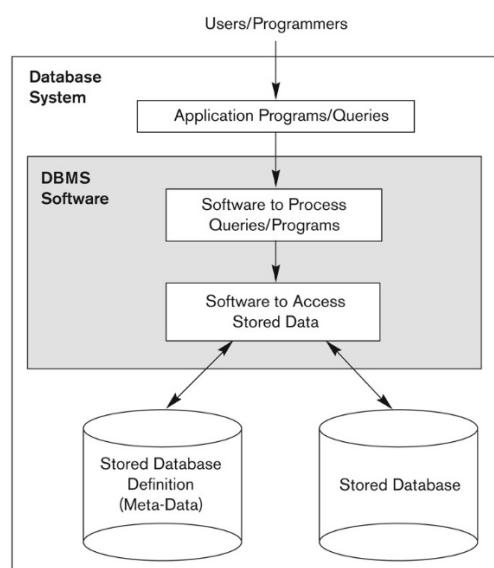
Database System

- Database

- 데이터 및 데이터간 관계의 집합

- DBMS (Database Management Systems)

- 사용자가 Database에 접근할 수 있도록 지원해주는 프로그램의 집합



Query 를 이용해서 데이터베이스에 접근하게 된다.

Example of a Database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

STUDENT

Name	Student_number
Smith	17
Brown	8

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

- 질의(Query) 예:

- “Database” Course의 Section을 하나라도 수강한 학생을 찾으시오.
- Section “135”, Student “8”의 Grade를 “B”로 수정하시오.

- Brown

쿼리 : 삽입 삭제 갱신 조회

Schemas versus Instances

● 데이터베이스 스키마 (Schema)

- 데이터베이스 구조, 데이터 타입, 그리고 제약조건에 대한 명세
- 데이터베이스 설계 단계에서 명시되며, 자주 변경되지 않음

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

데이터모델링에 들어가기 위해서 즉 데이터베이스 설계를 하기 위해서는 스키마라는 개념과 인스턴스라는 개념을 이해해야 한다.

데이터베이스 스키마라는 것은 데이터베이스 구조, 타입, 제약조건에 대한 명세이다.

데이터베이스 설계 단계에서 명시되며, 자주 변경되지 않는다.

Schemas versus Instances

● 데이터베이스 인스턴스 (Instance)

- 특정 시점에 데이터베이스에 실제로 저장되어 있는 데이터
= Database instance = Occurrence = Snapshot

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

인스턴스 : 특정 시점에 데이터베이스에 실제로 저장되어 있는 데이터

값들이 계속 쌓이게 되고 혹은, 교수이름이 변경되면 수정이 일어날 수 있음

빨간색 부분 : 실제 데이터가 담기는 부분 ⇒ 인스턴스

테이블에 인스턴스 빼고 뭐가 남나 → 회색 부분 : 컬럼명들 : 스키마라고 할 수 있음

데이터들은 추가되고 삭제되고 업데이트 되고 하니까 자주 변경이 된다.

자주 변경되기 때문에 어제와 오늘이 다르고 오늘과 내일이 다르기 때문에 한 시점의 사진으로 찍힌 것과 같다라고 해서 스냅샷이라고도 부르고, DB는 원래 수학자들 특히 집합을 하시는 분들이 고안을 해낸 아이디어라고 볼 수 있음

집합하는 사람들은 주로 Occurrence 라고도 함

엔지니어, 일반적으로 DB를 배우는 사람은 인스턴스라는 이름을 많이 사용함

인스턴스라는 것은 실제 값이 담기는 부분

스키마라는 것은 값이 담기는 부분을 제외한 껍데기 (이지만 중요)

Schemas versus Instances

● 데이터베이스 스키마 (Schema)

- 데이터베이스 구조, 데이터 타입, 그리고 제약조건에 대한 명세
- 데이터베이스 설계 단계에서 명시되며, 자주 변경되지 않음

STUDENT				
Name	Student_number	Class	Major	

COURSE				
Course_name	Course_number	Credit_hours	Department	

PREREQUISITE	
Course_number	Prerequisite_number

SECTION				
Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT		
Student_number	Section_identifier	Grade

각 테이블에 어떤 이름들이 있어야하는지 명시해줌

SQL (Structured Query Language)

● DML / DDL

- 데이터 정의어 (DDL: Data Definition Language)
 - 스키마를 기술하기 위해 사용되며, 주로 DB 설계자가 사용함
- 데이터 조작어 (DML: Data Manipulation Language)
 - 데이터의 조회(Retrieval), 삽입(Insertion), 삭제(Deletion), 갱신(Update)에 사용됨
- cf) DCL(Data Control Language), TCL(Transaction Control Language)

● 독립 실행형 / 내장형

- 독립 실행형
 - SQL 인터페이스를 이용하여 SQL 쿼리를 직접 DBMS에 입력
- 내장형
 - C, C++, Java 등의 프로그래밍 언어에 내장됨
 - Host language + Data sublanguage 로 구성됨

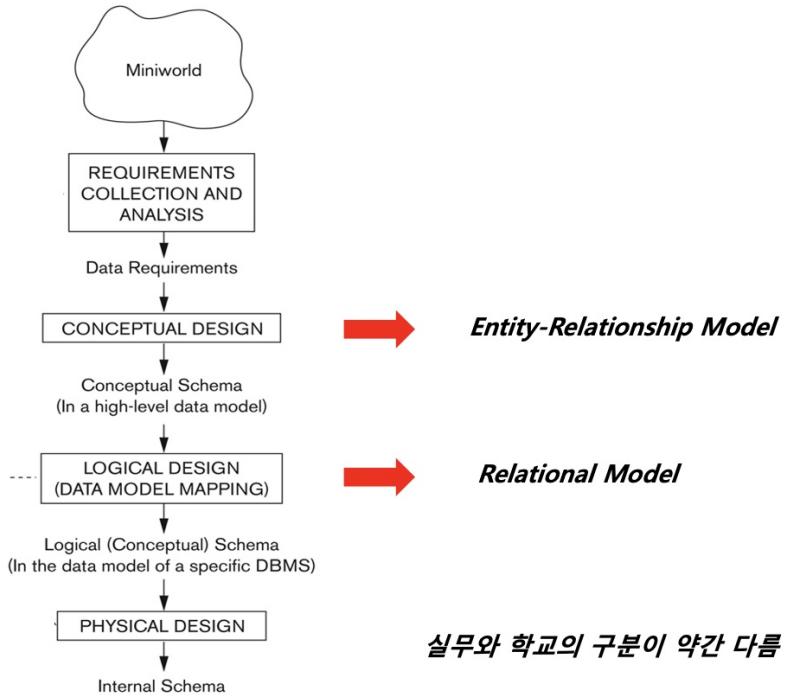
스키마를 기술한다는 것 : 테이블을 만든다든지, 컬럼을 만든다든지, 관계를 맺어준다는 이런 것들 ⇒ **DDL**

데이터를 집어넣고, 삭제하고, 혹은 데이터를 변경하는 값을 바꾸는 것들이니까 인스턴스에 대한 내용들 ⇒ **DML**

사용자 권한을 관리한다든가 트랜잭션을 관리한다든가 등등 ⇒ **DCL**, **TCL**

스키마에 대한 명령어, 인스턴스에 대한 명령어, 사용자에 대한 명령어, 트랜잭션에 대한 명령어가 구분되어 있음

Overview of Database Design Process



데이터베이스 설계는 크게 4단계로 나뉨

1. 요구사항 분석
2. 개념적 설계
3. 논리적 설계
4. 물리적 설계

Miniworld : 데이터베이스로 구축하고자하는 관심대상을 의미

1. 요구사항분석 : 업무기술서를 기술하게 됨 - ex) 학생은 학번을 가지고 나이와 성별을 명시한다, 학생은 과목을 수강한다, 하나의 과목은 여러개의 분반으로 개설 될 수 있다.
2. 개념적 설계 : 그 업무기술서의 내용을 서로 이해하기 좋게 오해의 소지가 없게 개념화를 하게 됨 - ex) 학생이 과목을 수강한다고 하면 도식화해서 표현할 수 있는 것

가장 많이 쓰이는 도식화 방법, 개념적 설계에 가장 많이 쓰이는 모델

⇒ Entity-Relationship Model : 객체관계 모형/모델

3. 논리적 설계 : 컴퓨터 안에 집어 넣어야 데이터베이스 구조가 만들어질 것이다. 그래서 컴퓨터가 알아들을 수 있는 구조로 바꾸는 과정

논리적 설계 중에 가장 많이 설계되는 모델 테이블 형태로 나타내는 모델

⇒ **Relational Model** : 관계형 모델

그러면 테이블 형태로 만들어진다.

4. 물리적 설계 : 관계형모델로 나타낸 것을 실제로 구현하는 과정

- 실무와 학교의 구분이 약간 다름 :
- Relationship ⇒ 관계 / Relation ⇒ 관계

DB에서 두 단어는 완전히 다른 개념

Relationship 이라는 것은 ER모델에서의 마름모에 해당하는 부분,

DB에서 관계라고 할때 Relationship 이라고 함

Relation 은 테이블을 의미한다 : 테이블로 나타낸 데이터베이스다 해서 ⇒ Relational Database

ER Model Concepts

● 개체 (Entity)

- 실세계에 존재하는 의미있는 하나의 정보 단위
- 물리적 객체 뿐 아니라 개념적 객체도 포함
 - (학생, 자동차, 강의실, …) / (프로젝트, 직업, 교과목, …)

● 관계 (Relationship)

- 개체들 사이의 연관성
 - [학생]과 [교과목] 사이의 [수강] 관계



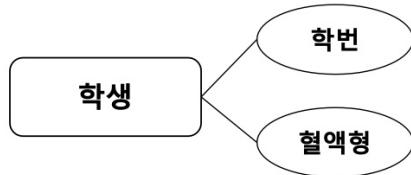
개체와 관계를 구분하는 것이 상당히 어려움

ER 다이어그램을 그릴때는 개체로 할지 관계로 할지 고민하게 되지만 테이블로 나중에 가게되면 릴레이션쉽 모델로 가게되면 개체로 할지 관계로 할지 애매했던 것들이 릴레이셔널 디비로 가게 되면 똑같아짐 따라서 크게 고민하지 않아도 된다.

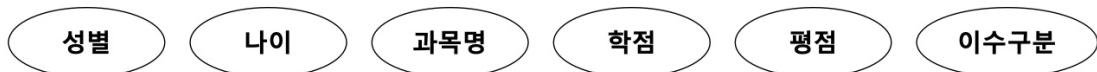
ER Model Concepts

- 속성 (Attribute)

- 개체 또는 관계의 본질적 성질



- 다음 개체 및 관계에서 주어진 속성의 주인 (Owner)는?



ER 모델을 구성하는 세가지 요소 - 개체, 관계, 속성

- 학생 : 성별, 나이

- 교과목 : 과목명, 학점

평점 : 학생의 고유한 본질적 성질이 아님, 교과목의 고유 본질적 속성도 아님 ⇒ 관계의 본질적 성질로 따져야함

: 수강의 속성이다.

이수구분 : 과목의 특징이 아니라 누가 뭘 들었느냐에서 나오는 특징

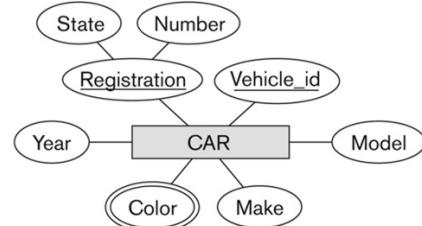
Types of Attributes

● Single-valued vs. Multivalued

- 나이 vs. 취미

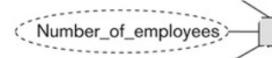
● Simple vs. Composite

- Simple Attribute - 더 이상 쪼개지지 않는 원자값을 갖는 속성
 - 나이, 학번, …
- Composite Attribute - 몇 개의 요소로 분해될 수 있는 속성
 - 주소 → 시, 군, 구, 번지, …



● Stored vs. Derived

- Derived Attribute - 저장된 다른 데이터로부터 유도 가능한 속성
 - 각 과목의 성적 → 총점, 주민등록번호 → 나이, …



- Single-valued Attributes : 항상 하나의 값만을 갖는 속성
- Multivalued Attributes : 여러 개의 값을 갖는 속성
- Simple Attributes :
- Composite Attributes : 둘 이상으로 여러가지로 나뉘어도 여전히 의미를 갖는 속성

⇒ 어떤 속성을 과연 쪼개도 되는지 안되는지는 설계자가 판단을 해야한다. ex) 어떤 속성을 안쪼갰더니 왜 안쪼갰어 ? ⇒ 우리는 쪼개서 쓰는 일이 없고 하나로 하는게 편해 ⇒ 쪼갤 필요가 없이 simple 로 있으면 됨

우리는 나눠서 사용하는 경우 많아서 나눠서 저장하는게 좋아 ⇒ simple simple 로 나눠서 사용

Entity Types and Key Attributes

● 키 속성 (Key Attributes)

- 어떤 개체에 대해서 항상 유일한 값을 갖는 속성 (또는 속성들의 집합)
 - 학생의 학번, 책의 ISBN, 자동차의 차량번호, …
- 특정 Snapshot이 아닌, 해당 개체의 모든 가능한 Snapshot의 집합을 고려하여 파악되어야 함
 - 다음의 SSN, 이름, 혈액형 중 키 속성을 찾으시오.



SSN: 740721-1111111

이름: 흥길동

혈액형: A



SSN: 820424-2222222

이름: 유관순

혈액형: A



SSN: 100201-3333333

이름: 강감찬

혈액형: A

- 키 속성 : 주민번호

Entity Types and Key Attributes

- 키 속성 (Key Attributes)

- 다음에서 키 속성을 찾으시오.



팀명: ManU
선수명: Park
등번호: 13
포지션: FW



팀명 : Chelsea
선수명 : Cech
등번호: 1
포지션 : GK



팀명 : Arsenal
선수명 : Park
등번호: 9
포지션 : SS



팀명 : ManU
선수명 : Berbatov
등번호: 9
포지션 : FW

- 키속성 : 팀명과 등번호

Entity Types and Key Attributes

- 키 속성 (Key Attributes)

- 복합 키 (Composite Key)
 - Composite Attribute가 키 속성이 되는 경우
 - 복합 키는 최소성을 가져야 함
 - (팀명, 등번호) vs. (팀명, 등번호, 선수명)
- 각 개체는 하나 이상의 키를 가질 수 있음
- 어떤 개체는 키를 갖지 않을 수도 있음
 - 약성 개체 (Weak Entity)

- 키 속성 특징 : 유일성, 최소성

- conceptional : Identifier
- logical : primary key

Identifier 을 가지고 primary key 을 만들게 된다.

Example COMPANY Database

● 요구사항 분석

- 회사는 여러 개의 부서로 구성되어 있다.
- 각 부서는 부서명, 부서번호, 부서 내 직원 수, 그리고 그 부서를 관리하는 직원(관리자)의 정보를 갖고 있다.
- 직원은 최대 하나의 부서를 관리할 수 있다. 한편 해당 직원이 부서 관리자로 근무를 시작한 정보도 저장해야 한다.
- 각 부서는 여러 지역에 위치하고 있을 수 있으며, 각 부서는 여러 프로젝트를 동시에 관리할 수 있다. 각 프로젝트는 고유의 프로젝트명, 프로젝트번호를 가지며, 하나의 프로젝트는 하나의 지역에서만 진행된다.
- 직원에 대해 직원명, 주민번호, 주소(시/도, 상세주소), 급여, 성별, 생년월일의 정보를 저장한다.
- 각 직원은 하나의 부서에 소속되어 있으며, 여러 프로젝트에 참여할 수 있다.
- 한편 각 직원이 참여한 프로젝트에 대해 해당 직원의 주당근무시간 정보도 관리해야 한다.
- 각 직원에 대해 해당 직원의 직속상사(감독자)에 대한 정보를 관리한다.
- 각 직원은 여러 명의 부양가족을 가질 수 있다. 부양가족에 각각에 대해 부양가족명, 성별, 생년월일, 그리고 해당 직원과의 관계에 대한 정보를 관리한다.

• Entity : 부서, 직원, 프로젝트명, 부양가족

→ 회사는 엔티티가 될 수 없음 : 여러 개의 회사를 비교하는 것이 아니고 한 개의 회사이기 때문

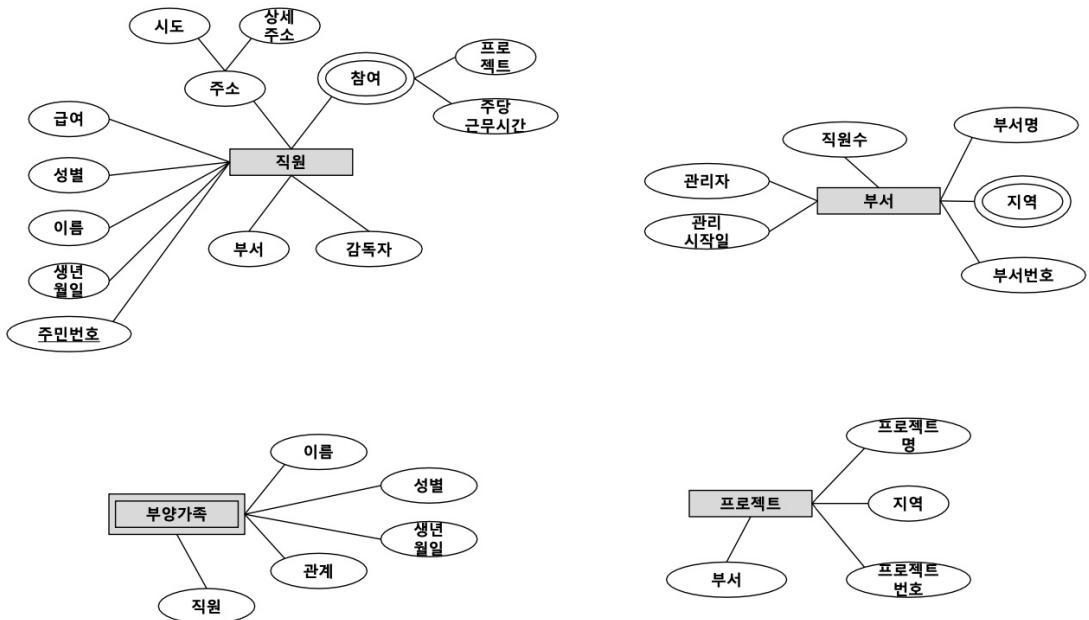
Entity ⇒ Relationship ⇒ Attribute

Initial Schema for COMPANYDatabase

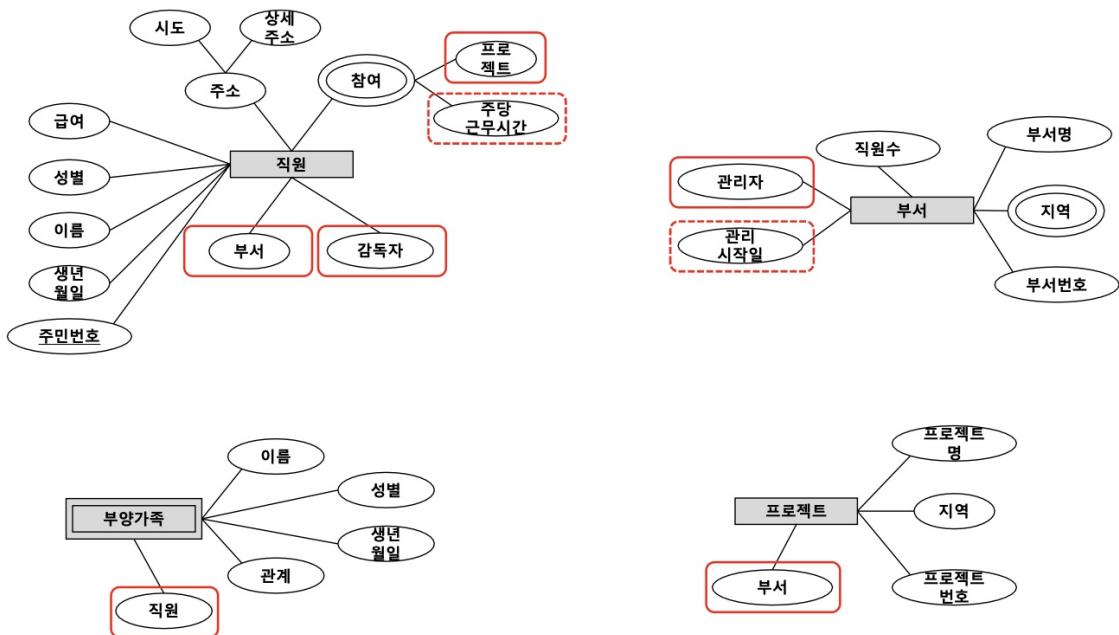
● 요구사항 정리

- Entity 부서:
 - 부서명, 부서번호, 관리자, 관리시작일, 직원수, 지역
- Entity 프로젝트:
 - 프로젝트명, 프로젝트번호, 지역, 부서
- Entity 직원:
 - 이름, 주민번호, 성별, 주소, 급여, 생년월일, 부서, 감독자, (프로젝트, 주당근무시간)
- Entity 부양가족:
 - 직원, 이름, 성별, 생년월일, 관계

Initial Schema for COMPANYDatabase



Initial Schema for COMPANYDatabase



Relationships

● 관계(Relationship) 설정

- 한 개체의 속성이 다른 개체를 참조할 때 관계가 형성됨
- [직원]의 [프로젝트] 속성이 [프로젝트] 개체를 참조함
- [부서]의 [관리자] 속성은 [직원] 개체를 참조함
- ...

● 관계의 차수(Degree)

- 관계에 참여하는 개체의 수
- Binary, Ternary, Unary, ...

● 관계의 대응수(Cardinality)

- 해당 개체가 해당 관계에서 참여할 수 있는 관계 인스턴스의 최대 수
- 1:1, 1:N (or N:1), M:N

Relationship

- 대응수에 따른 관계의 분류

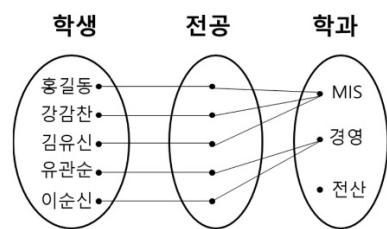
- 일대다 (1:N) 관계



- 일대일 (1:1) 관계

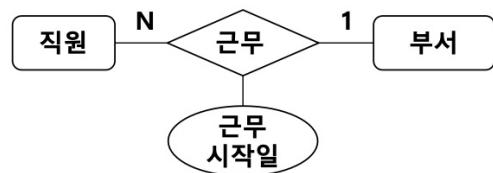


- 다대다 (M:N) 관계



Relationship

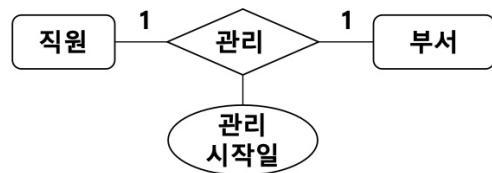
- 대응수에 따른 관계의 분류



- 일대다 (1:N) 관계
 - 위의 [근무] 관계의 [근무시작일] 속성이 이동한다면...
 - [근무시작일]은 어떤 개체에 속할 수 있는가?

Relationship

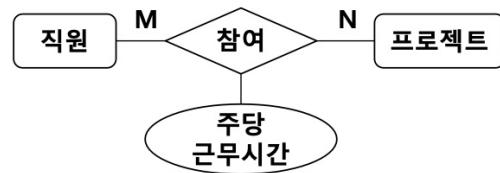
- 대응수에 따른 관계의 분류



- 일대일 (1:1) 관계
 - 위의 [관리] 관계의 [관리시작일] 속성이 이동한다면…
 - [관리시작일]은 어떤 개체에 속할 수 있는가?

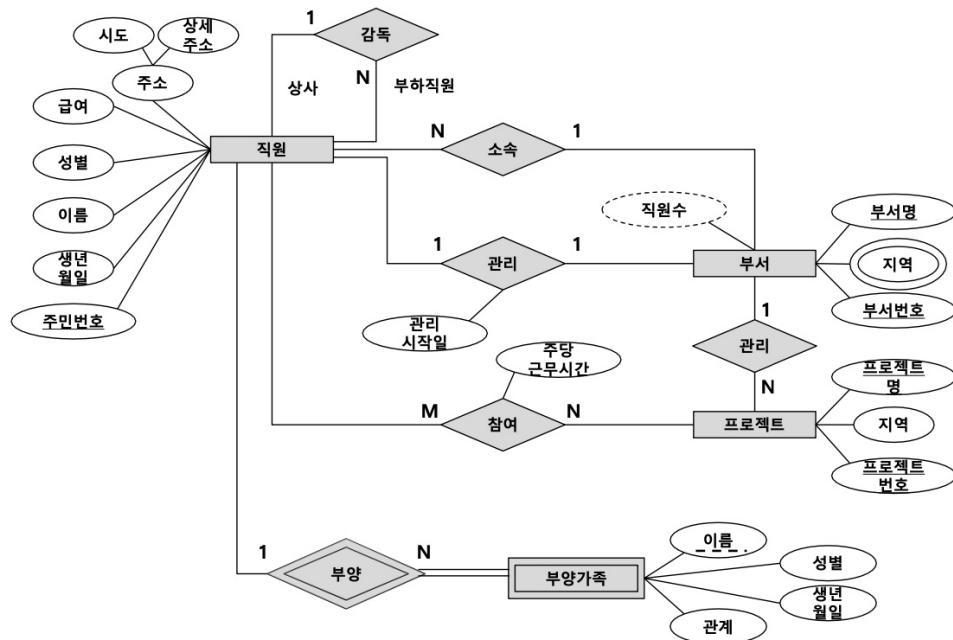
Relationship

- 대응수에 따른 관계의 분류



- 다대다 (M:N) 관계
 - 위의 [참여] 관계의 [주당근무시간] 속성이 이동한다면…
 - [주당근무시간]은 어떤 개체에 속할 수 있는가?

Refined Schema for COMPANYDatabase



Practical Approach for ER Modeling

● 주요 개체 도출

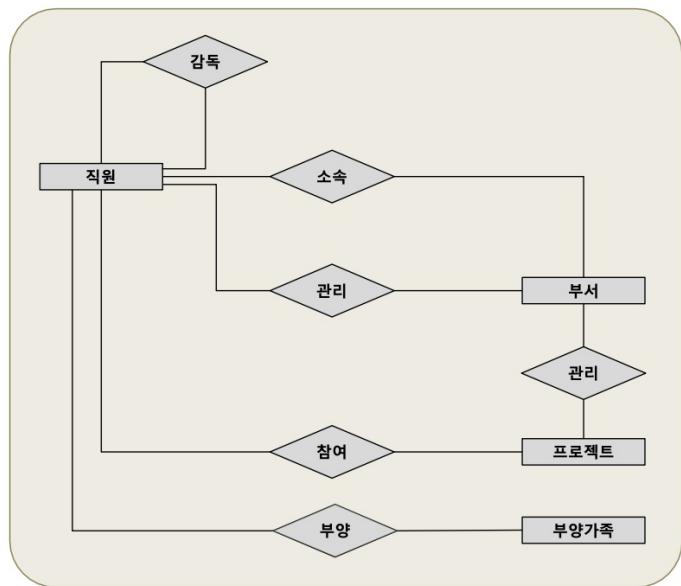
- 직원, 부서, 프로젝트, 부양가족

● 개체 간 관계 도출

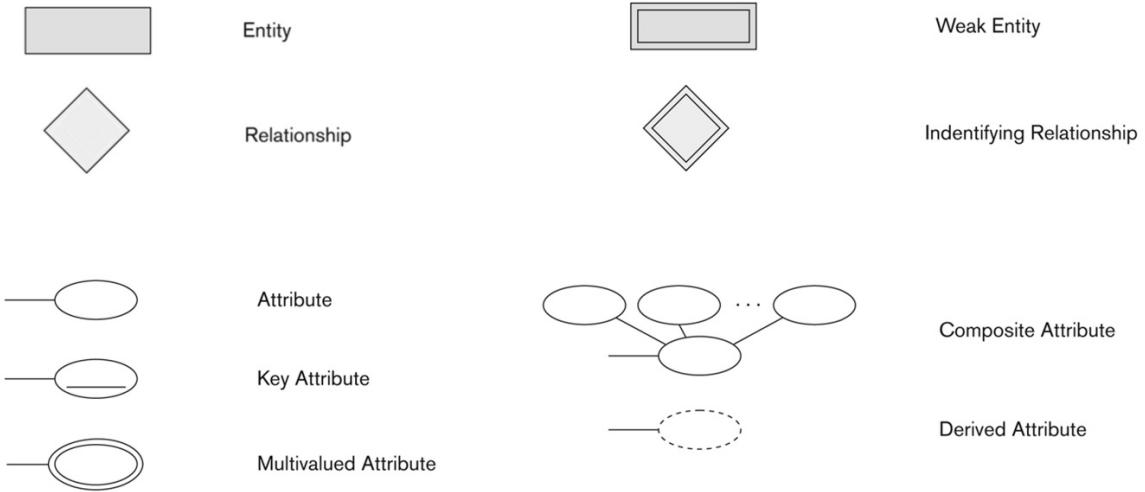
- 관계 도출 →
- 대응수 도출

● 개체 및 관계의 속성 도출

- 키 속성 도출
 - 약성 개체 확인
- 일반 속성 도출



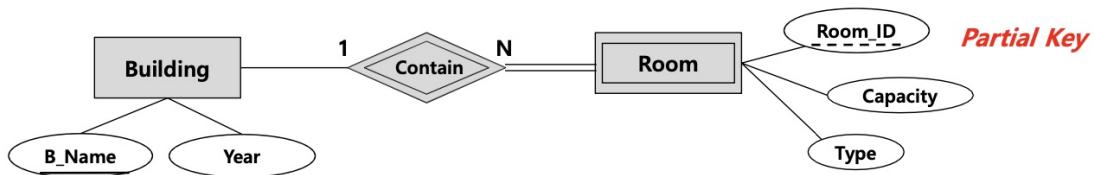
Notation for ER diagrams



Weak Entity Types

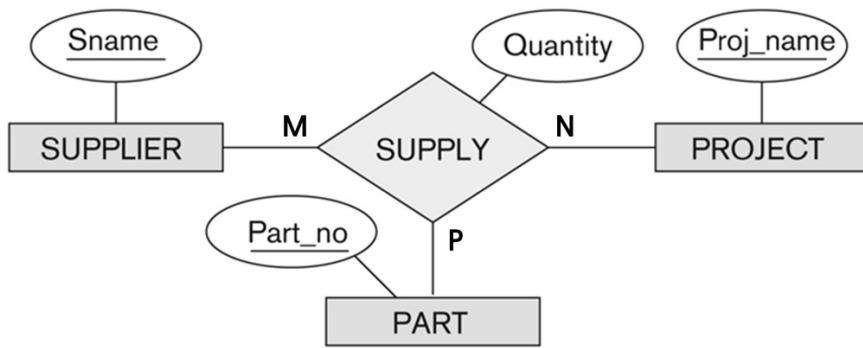
- 약성 개체 (Weak Entity) (\leftrightarrow Strong Entity)

- 키 속성을 갖고 있지 않은 개체 → 부분키 (Partial Key)만을 가짐
- 약성 개체는 개체를 식별할 수 있는 다른 개체와 식별 관계 (Identifying Relationship)으로 맺어져야 함
- 약성 개체의 식별자는 다음 속성의 조합으로 구성됨
 - 약성 개체의 부분키 속성+ 식별 개체(Identifying Entity)의 키 속성
 - ex) B_Name + Room_ID → 경영관 413호



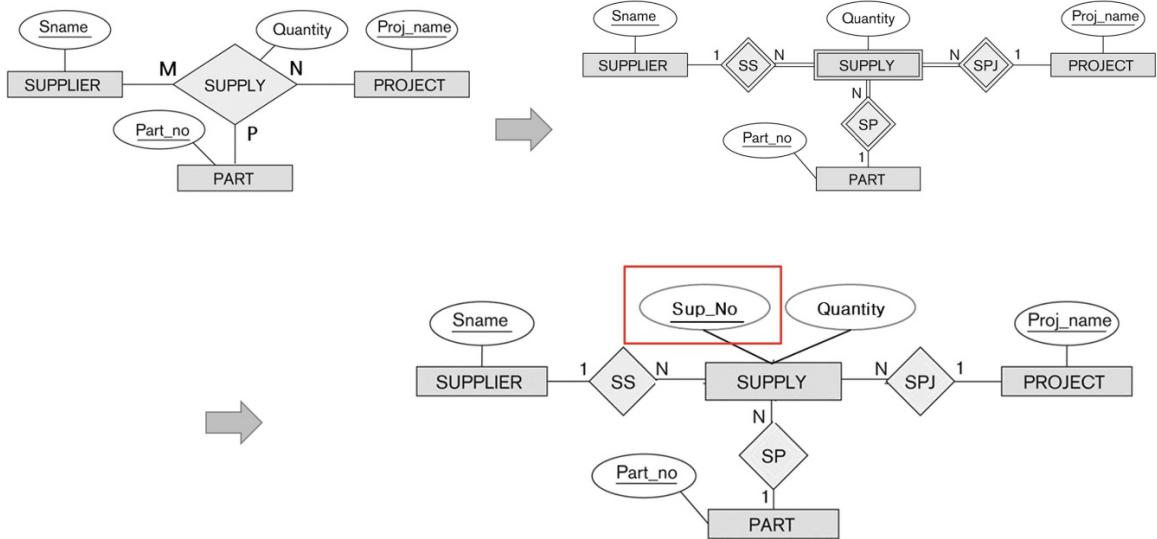
N-ary relationships ($n > 2$)

- Ternary Relationship

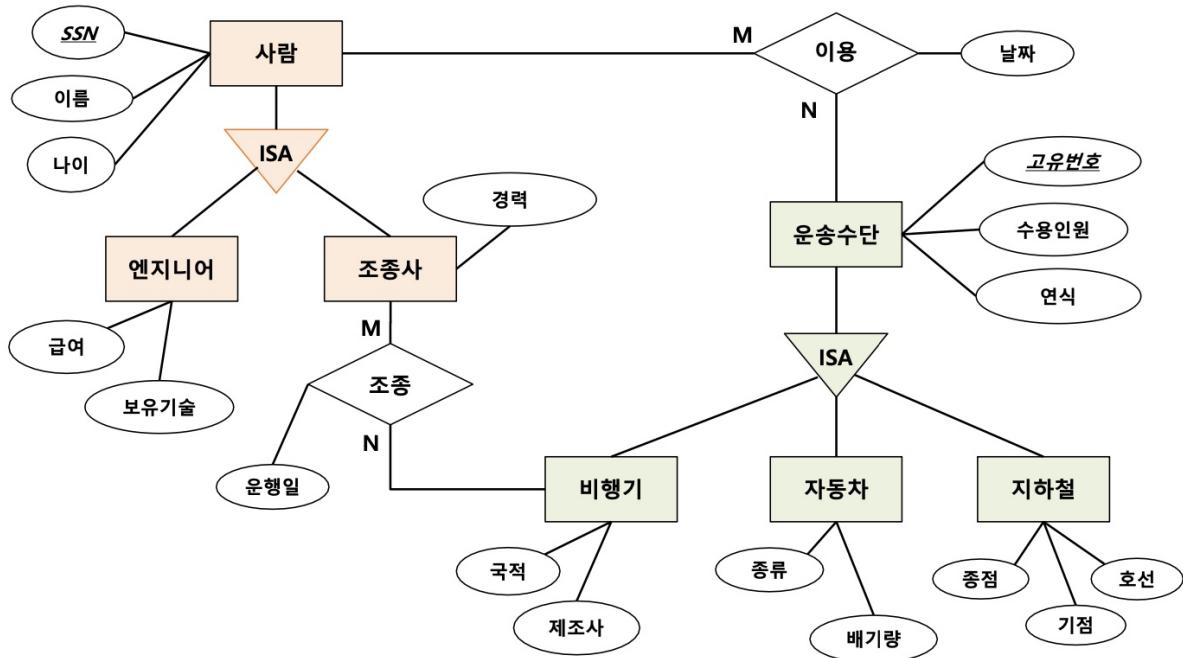


N-ary relationships ($n > 2$)

- Ternary Relationship → Binary Relationships



Generalization - ERD



Modeling Example (HW #1)

- 다음 요구사항에 부합하는 ERD를 작성하시오.

- 이 대학은 각 학생의 이름, 학번, 주민번호, 주소, 성별, 학년, 전공, 부전공에 대한 정보를 관리한다. 시스템 사용자는 학생의 주소를 조회하기 위해 City, State, Zip을 사용한다. 학생의 이름은 First Name과 Last Name으로 구분된다. 각 학생은 고유한 주민번호와 학번을 갖는다.
- 각 학과는 학과명, 코드, 사무실번호, 전화번호, 소속단과대학의 정보를 갖는다. 각 학과는 유일한 학과명과 코드를 갖는다.
- 각 과목은 과목명, 과목소개, 과목번호, 학점, 대상학년, 개설학과의 정보를 갖는다. 각 과목은 유일한 과목번호를 갖는다.
- 각 분반은 담당교수, 개설연도, 개설학기, 해당과목, 분반번호의 정보를 갖는다. 동일 연도 및 학기에 동일 교과목의 분반이 여러 개 개설될 수 있으며, 이들 분반은 분반번호에 의해 식별된다.
- 학생은 수강한 분반에서 평점을 부여받는다.
- 담당교수는 이름, 최종학위, 연구분야, 교번의 정보를 갖는다. 담당교수는 유일한 교번을 가지며, 한 교수가 하나 이상의 분반을 담당할 수 있다.

Solution (HW #1)

