

DBMS

3일차

강사 : 김근형

◆ 데이터 모델링

- 어떠한 정보 시스템을 구축할 때 내부에 필요한 데이터를 설계하는 과정
- 주어진 개념으로부터 논리적인 데이터 모델을 구성하는 작업
 - 데이터베이스의 골격을 이해하고 그 이해를 바탕으로 모델의 기능과 성능적인 측면에서 효율적인 작성이 필요
 - 물리적인 데이터베이스 모델로 환원하여 고객의 요구에 따라 특정 정보 시스템의 데이터 베이스에 반영하는 작업도 포함
- 데이터베이스 설계에서 첫 번째 단계에 해당

◆ 목적

- 업무 정보를 구성하는 기초가 되는 정보들을 일정한 표기법에 의해 표현함으로써 정보 시스템 구축의 대상이 되는 업무 내용을 정확하게 분석하는 것
- 분석된 모델을 가지고 실제 데이터베이스를 생성하여 개발 및 데이터 관리에 사용하기 위한 것

◆ 중요성

- 해당 업무 내용을 분석하고 이해하는 데도 유용하게 사용
- 데이터베이스 사용자 간에 의사소통을 쉽게 해준다
- 전체적인 조화와 균형을 유지할 수 있게 해준다
- 일관성은 있는지 등을 쉽게 파악할 수 있게 해준다

◆ 데이터 모델의 종류

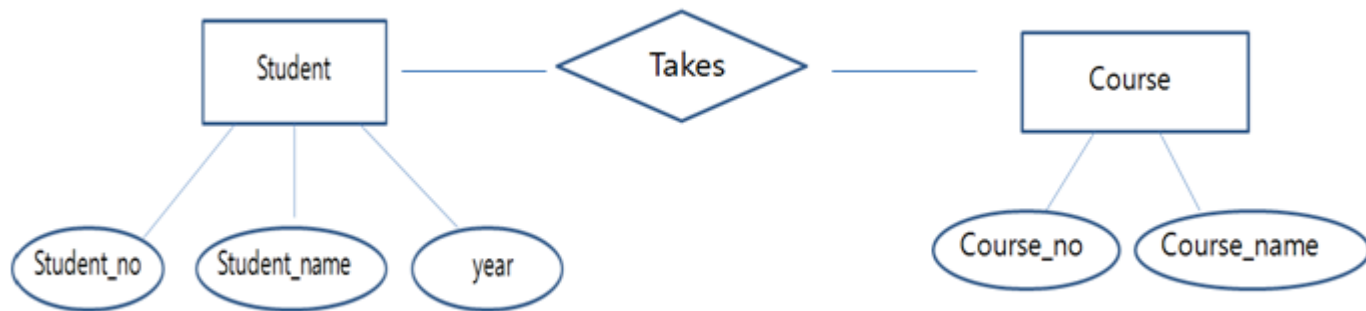
- 개념적 데이터 모델
- 논리적 데이터 모델
- 물리적 데이터 모델

◆ 개념적 데이터 모델

- 고수준의 데이터 모델
- 전체 시스템에 대한 개념적인 정보를 나타내는 데 사용
- 시스템 구상 단계에서 대략적인 스케치를 한다고 생각하면 좋다
- 논리 모델을 만들기 위한 기초 작업
- 엔티티-관계(E-R) 모델을 구성한다

◆ 엔티티-관계(E-R) 모델

- 개념적 데이터 모델
- 엔티티, 애트리뷰트, 관계를 이용해서 실세계를 개념적으로 표현
- 구성 요소
 - 엔티티(개체) : 모델의 관리 대상, 실체가 있는 것이나 개념, DB의 테이블, 사각형 표시
 - 애트리뷰트(속성) : 엔티티의 구성 요소, 타원 표시
 - 관계 : 엔티티 간의 관계(1:1, 1:N, N:M), 관계형 DB로 매핑(사상), 마름모



◆ ER 모델 → 엔티티

- 실제로 존재하는 대상
- ER 모델에서 가장 기본이 되며, 고유하게 식별이 되어야 한다
- EX) 엔티티의 예
 - 사람 : 학생, 교수, 사원 등....
 - 장소 : 학교, 강의실 등...
 - 사물 : 교과서, 생산 제품 등...
 - 사건 : 강의, 면담 등...
 - 개념 : 강좌, 과목, 프로젝트 등.....

◆ ER 모델 → 엔티티 타입

- 엔티티가 모여서 하나의 집단을 이룬 형태
- ER 다이어그램에서 엔티티 타입은 사각형으로 표현
- 그냥 엔티티라고 생각해도 좋다
- 분류
 - 강한 엔티티 타입 : 보통의 엔티티 타입
 - 약한 엔티티 타입 : 자신의 키(기본 키) 애트리뷰트가 없는 엔티티 타입, 다른 엔티티에 종속되어서 해당 엔티티가 없다면 존재하지 않는 종속성을 가진다

◆ ER 모델 → 애트리뷰트

- 엔티티 또는 관계가 갖는 성질이나 특성
- 엔티티는 반드시 하나 이상의 키 애트리뷰트를 갖고 있어서 나머지 애트리뷰트를 유일하게 정의할 수 있다
- 쉽게 말하자면 보통의 엔티티들은 자신의 키(기본 키) 애트리뷰트를 가지고 있다는 뜻이다
- 애트리뷰트의 분류
 - 단순 애트리뷰트
 - 키 애트리뷰트
 - 복합 애트리뷰트
 - 다치 애트리뷰트
 - 유도된 애트리뷰트
 - 부분 키 애트리뷰트

◆ ER 모델 → 단순 애트리뷰트

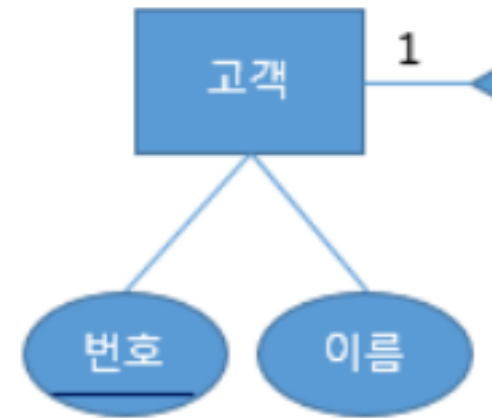
- 보통의 애트리뷰트



◆ ER 모델 → 키 애트리뷰트

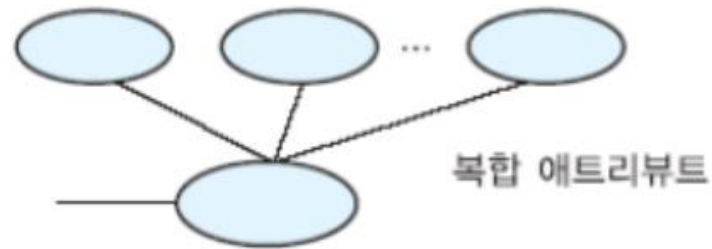
- 엔티티들을 식별할 수 있는 유일한 제약조건을 갖는 애트리뷰트
- 보통 ER 다이어그램에서는 기본키나 후보키를 키 애트리뷰트라고 하고 모두 밑줄을 그어서 표시한다

- EX
 - 학생의 학번
 - 사원의 사원번호
 - 사람의 주민등록번호



◆ ER 모델 → 복합 애트리뷰트

- 두 개 이상의 애트리뷰트
- 쉽게 말하자면 애트리뷰트가 또 다른 애트리뷰트로 나뉘지는 것이다



◆ ER 모델 → 다치 애트리뷰트

- 애트리뷰트 하나에 여러 값이 들어갈 수 있는 애트리뷰트
- 예를 들면 강사 엔티티의 보유 기술 애트리뷰트는 강사가 보유한 다양한 기술이 들어갈 수 있다
- 두 선으로 타원을 그려 표시



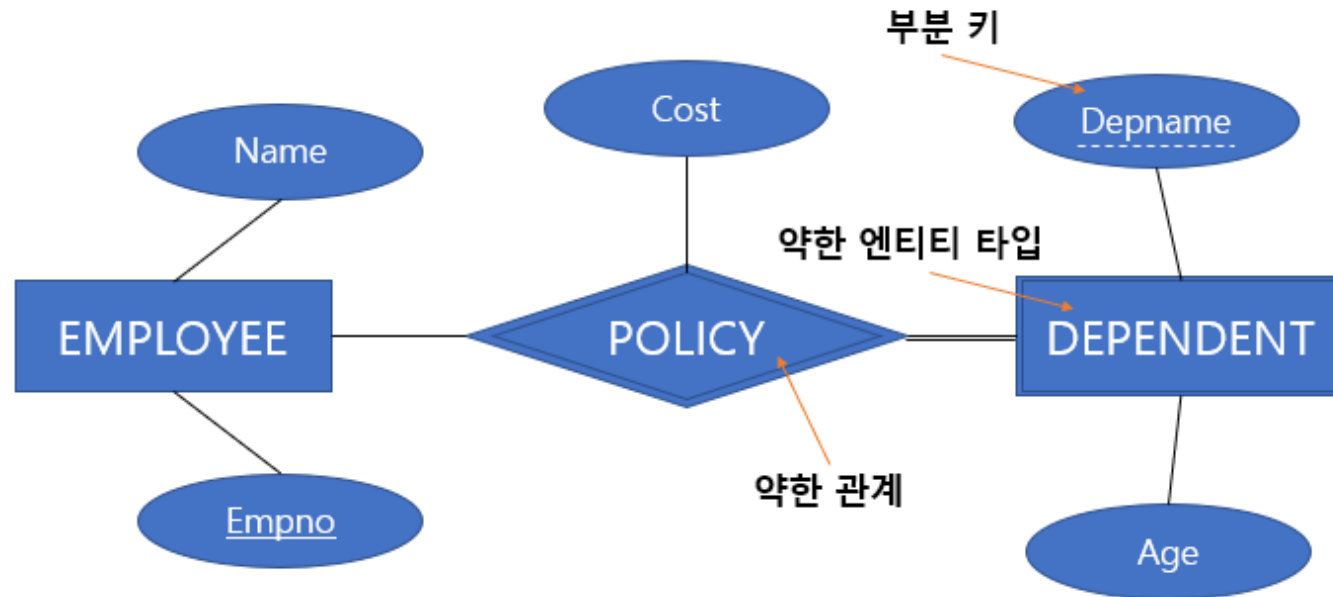
◆ ER 모델 → 유도된 애트리뷰트

- 실제 값이 저장되어 있는 것이 아니라 저장된 값에서 계산해 얻은 값을 사용하는 애트리뷰트
- 예시로 나이는 실제 나이를 저장하지 않고 생년월일(저장된 값)과 오늘 날짜로부터 계산해서 얻는다



◆ ER 모델 → 부분 키 애트리뷰트

- 키와 비슷하지만 완벽하게 키라고는 할 수 없고 약한 엔티티에서만 사용
- 보통 잘 사용하지 않지만 점선으로 밑줄을 그어서 표현



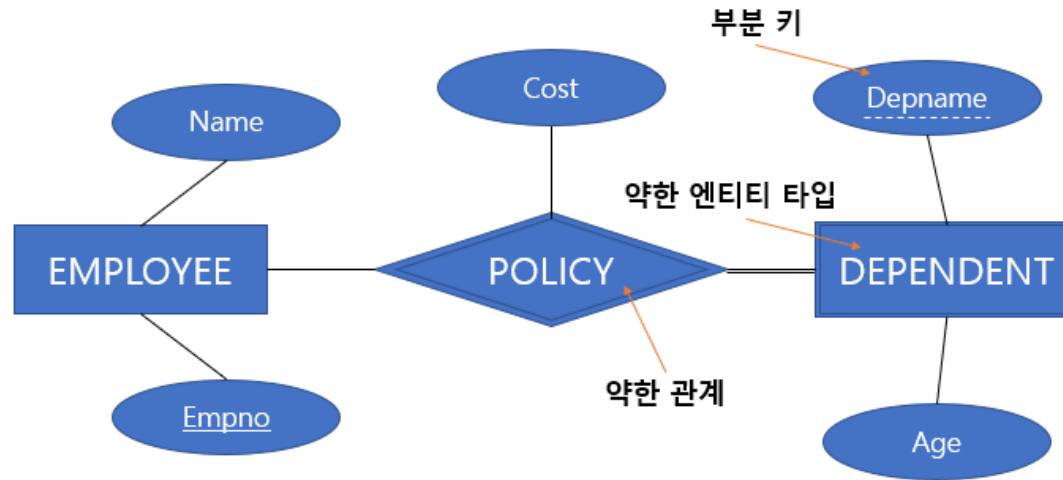
◆ ER 모델 → 관계

- 엔티티 타입 간의 관계를 표현할 때 사용
- 엔티티 간에 존재하는 수학적 관계를 말한다
- ER 다이어그램에서 마름모를 사용하여 표현
- 일대일(1:1), 일대다(1:N), 다대다(N:M) 관계가 있음



◆ ER 모델 → 식별 관계

- 약한 엔티티를 식별할 때 사용하는 관계 타입
- 강한 엔티티와 약한 엔티티의 연결 고리
- 그림에서 보이는 약한 관계가 식별 관계이다



◆ ER 모델 → 카디널리티 비율

- 두 엔티티 사이의 관계에서 참여자의 수를 표현하는 것을 카디널리티 비율이라고 한다
- 비율의 종류
 - 1:1
 - 1:N
 - N:M

◆ ER 모델 → 카디널리티 비율 → 일대일 관계(1:1)

- 하나의 엔티티에 대하여 하나의 엔티티만이 관계를 맺는 경우
- 예를 들면 고유한 한 사람마다 고유한 하나의 주민번호를 가지기 때문에 이 관계는 1:1 관계 입니다



◆ ER 모델 → 카디널리티 비율 → 일대다 관계(1:N)

- 하나의 엔티티에 대하여 여러 개의 엔티티가 관계를 맺는 경우
- 예시를 들면 하나의 공급자가 여러 물건들을 공급한다고 한다면 이 관계는 1:N 관계입니다



◆ ER 모델 → 카디널리티 비율 → 일대다 관계(1:N)

- 여러 개의 엔티티에 대하여 여러 개의 엔티티가 관계를 맺는 경우
- 예시를 들면 각 사원은 여러 대의 PC를 사용할 수 있고, 각 PC 는 여러 명의 사원이 사용한다고 한다면 이 관계는 M:N입니다.

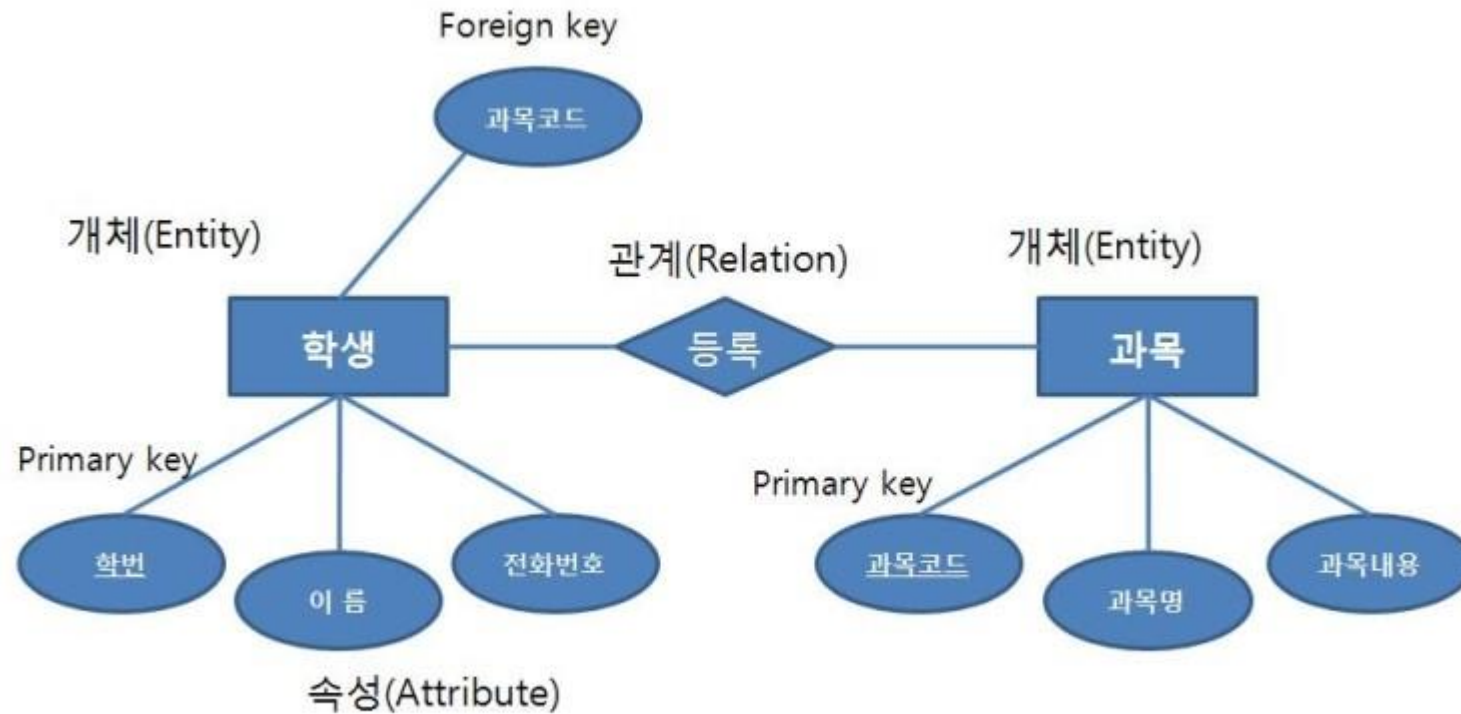


◆ ER 모델 → 전체 참여와 부분 참여

- 모든 부서마다 반드시 1명의 관리자가 있어야 하기 때문에, 부서(department) 엔티티는 관리(manage) 관계에 전체 참여 → 두 줄 표시
- 반면, 일부 직원들만 부서의 관리자가 될 수 있기 때문에, 직원(employee) 엔티티는 관리(manage) 관계에 부분 참여 → 한 줄 표시










◆ ER 다이어그램 예시



◆ ER 다이어그램을 그려보자

- <https://www.drawio.com/> 에 접속 후 다운로드

 draw.io-21.5.1-windows-installer.exe	99.3 MB	4 days ago
 draw.io-21.5.1-windows-installer.exe.blockmap	106 KB	4 days ago
 draw.io-21.5.1-windows-no-installer.exe	99.1 MB	4 days ago
 draw.io-21.5.1.msi	131 MB	4 days ago
 draw.io-arm64-21.5.1.dmg	147 MB	4 days ago
 draw.io-arm64-21.5.1.dmg.blockmap	159 KB	4 days ago
 draw.io-arm64-21.5.1.zip	142 MB	4 days ago

◆ ER 다이어그램을 그리기 연습문제

- 고객을 관리하기 위해 고객번호, 고객이름, 주소, 직업, 나이를 저장한다
- 계좌를 관리하기 위해 계좌번호, 유형, 잔액을 저장한다
- 고객 한 명이 여러 계좌를 소유할 수 있다
- 계좌 하나는 한 명의 고객만 소유할 수 있다

◆ ER 다이어그램을 그리기 연습문제2

- 항공사에서 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야 한다
- 신용카드 정보는 여러 개를 저장할 수 있는데, 세부적으로는 카드번호, 유효기간을 저장할 수 있다 → 복합 애트리뷰트
- 항공사에서는 보유한 비행기에 대해 비행기번호, 출발날짜, 출발시간 정보를 저장하고 있다
- 항공사에서는 좌석에 대해 좌석번호, 등급 정보를 저장하고 있다
- 회원은 좌석을 예약하는데, 회원 한 명은 좌석을 하나만 예약할 수 있고, 한 좌석은 회원 한 명만 예약할 수 있다
- 비행기에는 좌석에 존재하는데, 비행기 하나에는 좌석이 여러 개 존재할 수 있고 한 좌석은 반드시 하나의 비행기에만 존재해야 한다. 그리고 좌석은 비행기가 없으면 의미가 없다

◆ 논리적 데이터 모델

- 표현 데이터 모델 또는 구현 데이터 모델로도 불린다
- 구축할 시스템의 이미지
- 종류
 - 관계형 데이터 모델
 - 계층형 데이터 모델
 - 네트워크형 데이터 모델
 - 객체지향형 데이터 모델

◆ 논리적 데이터 모델 → 관계형 데이터 모델

- 테이블 형태로 나타내는 데이터 모델
- 간단하여 이해하기가 쉬우며, 데이터의 저장 위치와 접근 방법을 DBMS가 결정하기 때문에 사용자는 필요한 데이터만 명시하면 된다

속성(Attribute)

학생 Relation

학번	이름	학년	전공
20163456	홍길동	4	컴퓨터공학
20171234	김철수	3	물리학
20193543	이순신	3	수학

스키마(Schema)

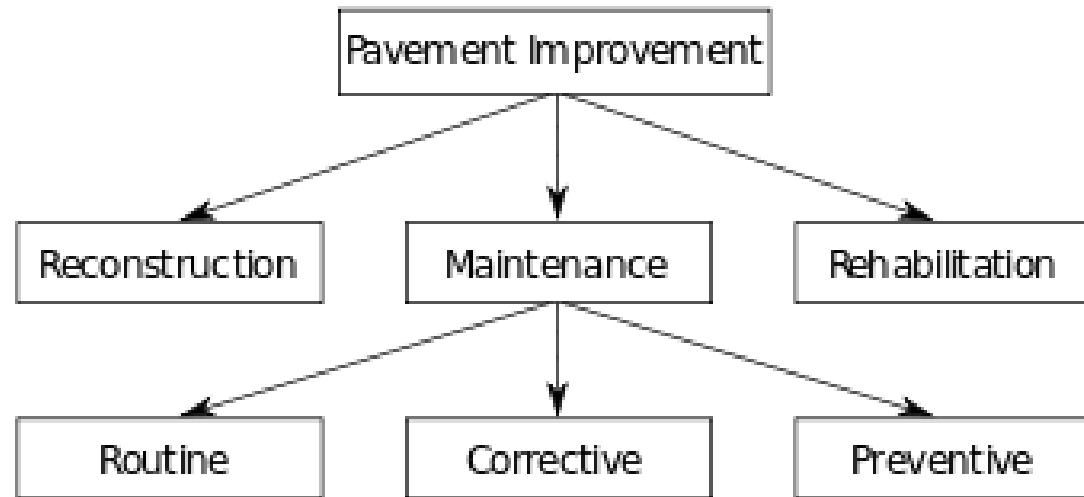
튜플(Tuple)

The diagram illustrates a '학생 Relation' (Student Relation) table. A red box highlights the '학년' (Year) column, with a red arrow pointing to it from the label '속성(Attribute)'. A green arrow points to the '전공' (Major) column from the label '스키마(Schema)'. A blue arrow points to the second row (20171234, 김철수, 3, 물리학) from the label '튜플(Tuple)'. The table has four columns: '학번' (Student ID), '이름' (Name), '학년' (Year), and '전공' (Major). It contains three rows of data.

◆ 논리적 데이터 모델 → 계층형 데이터 모델

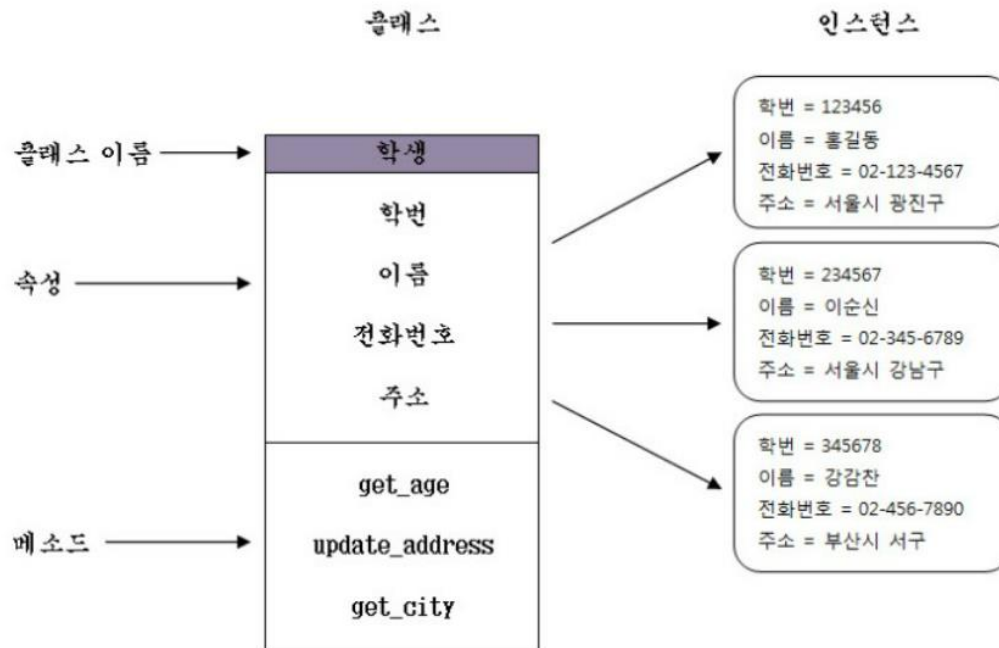
- 계층형에서 데이터는 레코드와 링크로 구성된 트리 형태로 나타낸다
- 링크로 연결된 레코드의 집합은 부모-자식 관계를 표현

Hierarchical Model



◆ 논리적 데이터 모델 → 객체지향형 데이터 모델

- 데이터와 메소드를 하나의 객체로 다루기 때문에 이해하기 쉽고, 유지와 변경이 용이
- 객체 지향 프로그래밍의 패러다임을 기반으로 함



◆ 논리적 데이터 모델 → 모델 장단점 비교

종류	장점	단점
관계형 데이터 모델	<ul style="list-style-type: none">유지보수가 쉬움사용자 편의성업무 변화에 대한 적응력 우수	<ul style="list-style-type: none">메모리 용량이 많이 필요다대다(N:M) 관계의 표현이 어려움정보 추출에 많은 시간이 필요
계층형 데이터 모델	<ul style="list-style-type: none">일대다(1:N) 관계의 표현이 쉬움데이터 처리가 빠름	<ul style="list-style-type: none">숙련된 전산 요원 필요다대다(N:M) 관계의 표현이 어려움업무 변화에 대한 적응력이 부족
네트워크형 데이터 모델	<ul style="list-style-type: none">다대다(N:M) 관계의 표현이 쉬움데이터 접근이 쉬움	<ul style="list-style-type: none">데이터의 종속성운용이 불편함
객체지향형 데이터 모델	<ul style="list-style-type: none">재사용성이 쉬움유지보수가 쉬움업무 변화에 대한 적응력이 우수	<ul style="list-style-type: none">개념이 미정립됨설계가 어려움

◆ 물리적 데이터 모델

- 컴퓨터 내부에서 실제로 데이터들이 어떻게 저장되는 가를 표현
- 레코드가 실제로 저장되는 형식, 특정 레코드를 효율적으로 탐색하는 구조인 접근 경로 등을 기술한다