4주차 ER Model RDBMS

2022. 3. 24. 13:21
https://blog.naver.com/lsm980515/222681648587

Database Design Process

1. Requirements Analysis

- What is going to be stored?
- How is it going to be used?
- What are we going to do with the data?
- who should access the data?

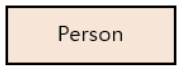
2. Conceptual Design

- A high-level description of the database
- Sufficiently precise that technical people can understand it
- But, not so preciese that non-technical people can't participate

3. Logical, Physical, Security etc..

- Logical Databse Design
- Physical Database Design
- Security Design

Entities and Entity Sets



entity set

Entity는 Entity set안에 있는 특정한 object이다.

그래서 이러한 entity가 모여서 entity set이 된다.

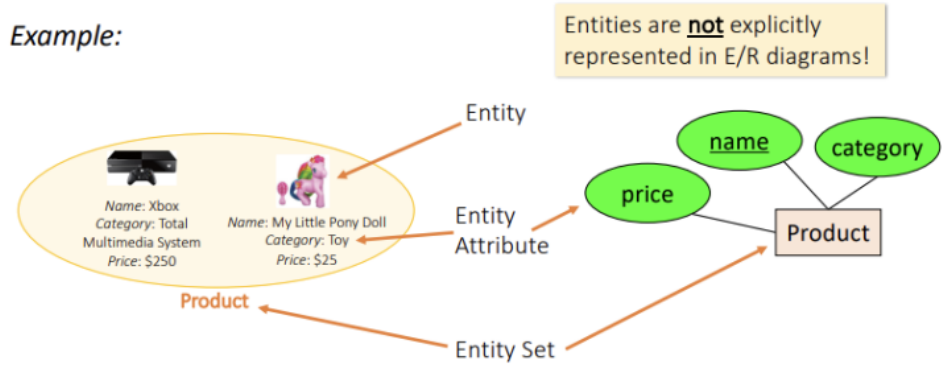
예로 특정한 제품은 하나의 entity이고, DB 안의 모든 제품들의 모음은 하나의 entity set이 된다.

따라서 entity는 tuple이고, 이런 entity의 모음이 entity set이 된다.

위와 같은 **entity set은 여러 attribute를 가진다.**

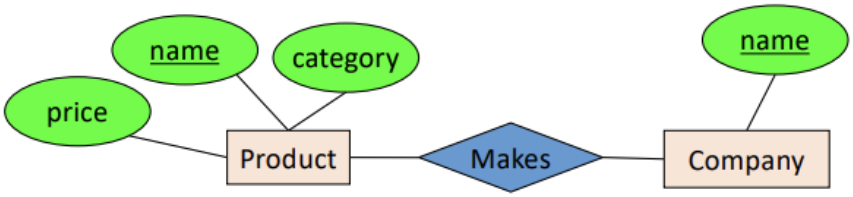
그리고 이러한 entity set은 고유하게 식별되는 attribute 집합을 가지고 있어야되는데 없으면 weak entity set이 될수 있다. <<<아래에 추가 설명

E/R모델에서는 tuple은 표기하지 않음



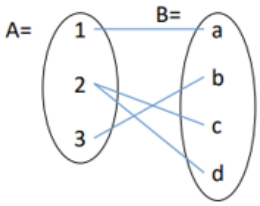
The R in E/R : Relationships

객체-관계 모델링(Entity-Relationship Modelling)이라고 하고
ERM 프로세스의 산출물을 가리켜 **개체-관계 다이어그램(Entity-Relationship Diagram)**이라 함.

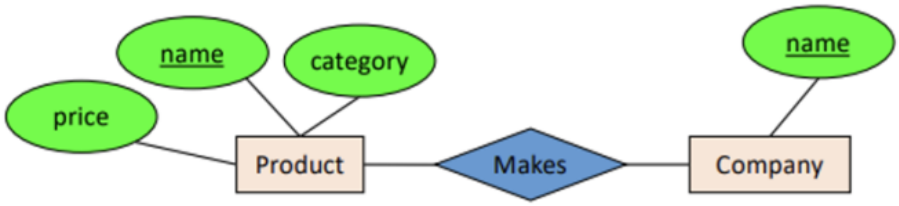


위 그림은
두개의 entity set(entity라고 부름)이 있다. 바로 product와 company이고
각 attribute는 product(price, name, catergory), company(name)이고
두 entity는 makes라는 관계(relationship)로 연결되어있다.
여기서 relationship은 두 entity를 연결해주는 또 다른 table이다.
이 ER diagram은 회사와 제품의 관계를 makes로 나타내고 있다.

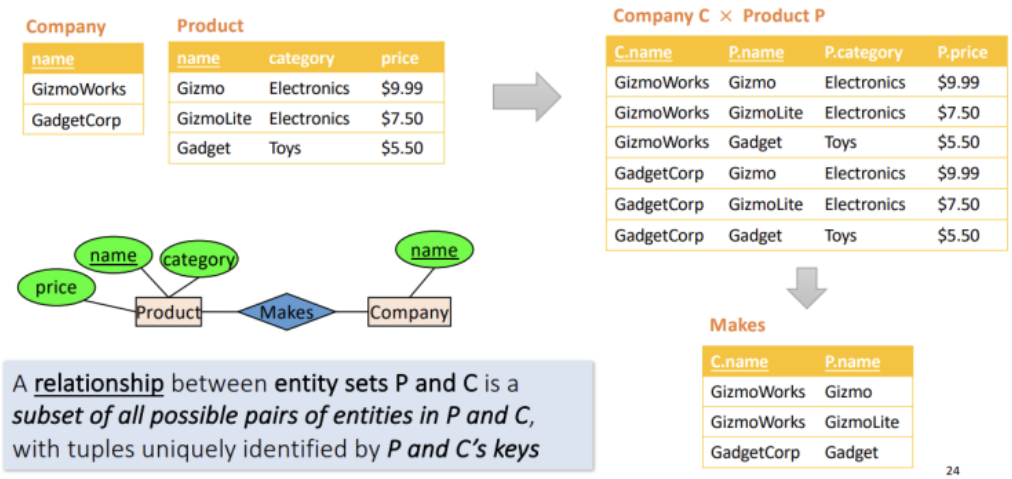
What is a Relationship?



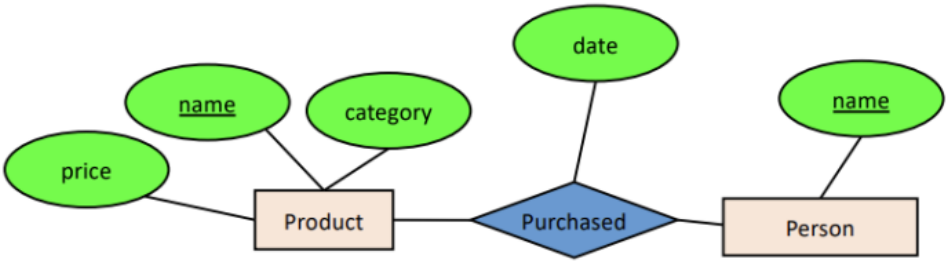
두 집합의 Cross-product에서 모든 원소를 가지는 것을 relationship이라고 하지 않고
Cross-product한 집합을 P라고 하고 relationship을 R이라고 하면,
$R = \{r \mid r \in P \text{ and } R \subset P\}$
즉, **R은 AxB의 subset이다.**



여기서 makes라는 relationship은 Product와 Company로 만들 수 있는 모든 쌍(PxC)의 부분집합 이다.
그리고 makes relationship은 각 개체(entity)의 primary key로 유일하게 구분될 수 있다.



그림에서 PxC의 결과만 봤을때 하나의 attribute로는 유일성을 충족하지 못하기 때문에 이 조건을 충족하기 위해 두개의 attribute를 합쳐서 하나의 primary key를 형성한 것을 알 수 있다.
Company와 Product 조합에서 하나의 관계(relationship)만 유일하다.
왜냐하면 두 entity의 primary key로 relationship을 구성하기 때문에 유일하다.



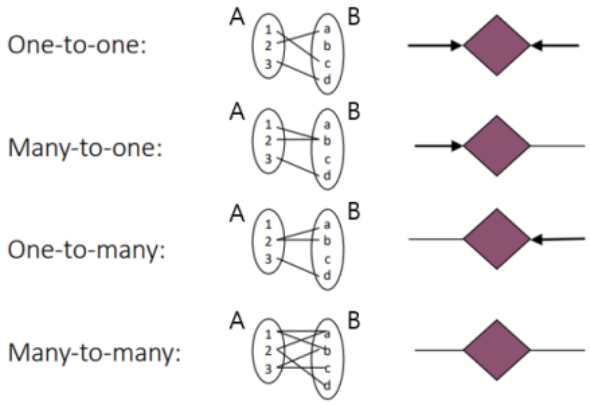
이 relationship은 유일하지만 한 사람이 특정한 날에 하나의 제품들만 살 수 있다.
왜냐하면 사람의 name과 제품의 name으로 유일한 relationship을 만들었는데 동일 날짜에 한사람이 똑같은 제품을 두개 산다면, 해당 table의 tuple이 똑같은 값을 갖게 된다.

Multiplicity of E/R Relationships

ER(entity-relationship) 데이터 모델은 데이터를 객체들과 그들관은 관계에 의하여 묘사하는 것

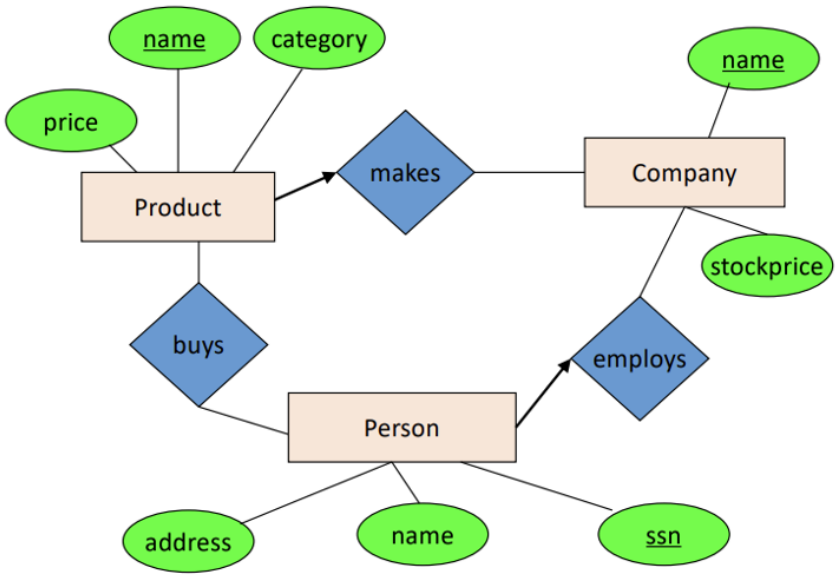
표기방법

기호	의미
	개체(entity)
	관계(Relationship)
	속성(attribute)
	기본키(Primary key)
	개체간의 관계(연관성)
	개체와 속성을 연결
	ISA관계(상속관계)



One-to-one(1 : 1 관계) : 집합 A의 각 원소가 집합B의 원소 1개와 대응
Many-to-one(N : 1 관계) : 집합 A의 각 원소는 집합 B의 원소 한 개와 대응할 수 있고, 집합 B의 각 원소는 집합A의 원소 여러 개와 대응 가능하다.
One-to-many:(1 : N 관계) : 집합 A의 각 원소는 집합 B의 원소 여러 개와 대응할 수 있고, 집합 B의 각 원소는 집합 A의 원소 한 개와 대응할 수 있다.
Many-to-many(M : N관계) : 집합 A의 원소는 개체 집합 B의 원소 여러개와 대응할 수 있고, 집합 B의 각 원소는 집합 A의 원소 여러 개와 대응할 수 있다.

※tip : one-to-one을 제외하고 many가 있는 쪽에서 화살표가 출발하는 방향임.

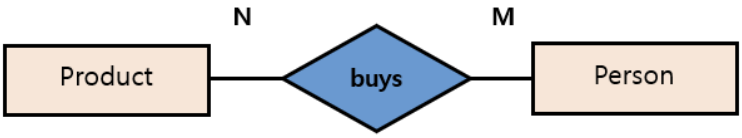


위의 예시로 이것이 의미하는 것을 알아보면
Person(entity)에는 [address, name, ssn](attribute)이 있고
Product(entity)에는 [price, name, category](attribute)이 있고
Company(entity)에는 [name, stockprice](attribute)이 있고

Person과 Company는 employs로 불리는 relationship으로 연결이 되어있고
many to one 관계이다.

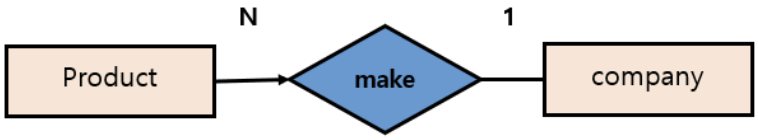
한명의 사람은 하나의 회사에 소속 되고, 하나의 회사는 여러 사람들이 있을 수 있다.

Product와 Person은 buys로 불리는 relationship으로 연결이 되어있고
many to many 관계이다.



1개의 제품은 여러사람들에게 구매되어질 수 있고, 한명의 사람은 여러 제품을 살수 있다.

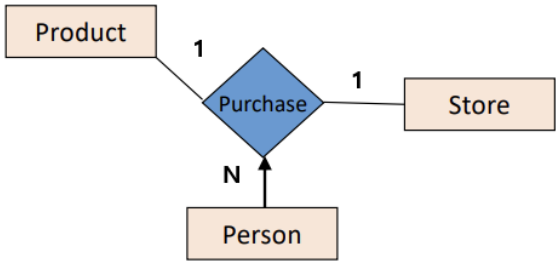
Product와 Company는 makes로 불리는 relationship으로 연결이 되어있고
many to one 관계이다.



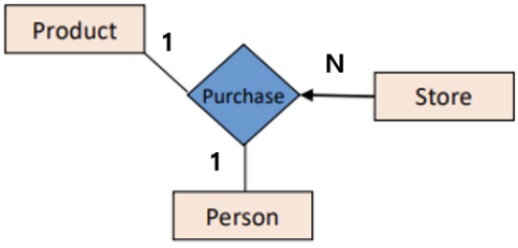
한개의 제품은 한 회사에서 만들어 지고, 한개의 회사는 여러개의 제품을 만든다.

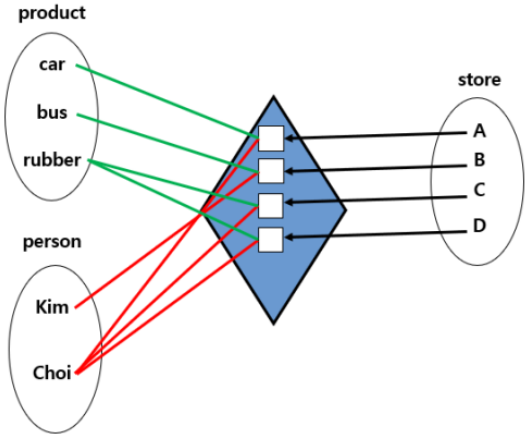
Multi-way Relationships

Q: What does the arrow mean ?



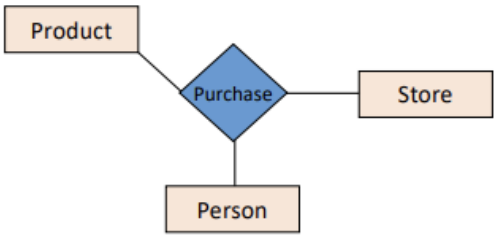
ans : 여러 사람들이 하나의 물건을 하나의 상점에서 구매할 수 있다.





ans : 매장은 여러개가 있고, 특정한 매장에서 한 사람이 한 제품을 구매할 수 있다.
그림처럼 각 관계를 보면 한 사람이 한 매장에서 한 개의 제품을 구매할 수 있음을 알 수 있다.

Q: How do we say that every person shops in at most one store ?

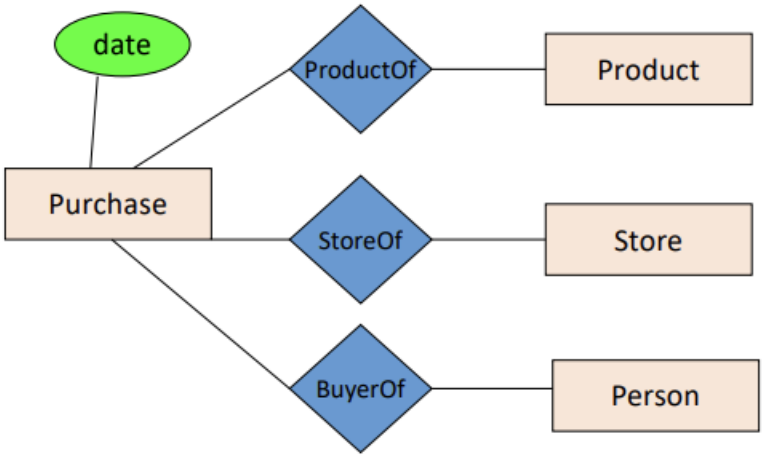


A: Cannot. This is the best approximation.
(Why only approximation ?)

위 문제에서 적어도 한개의 매장이라는 부분 때문에, 지금과 같은 관계로는 정확하게 표현하지 못한다.

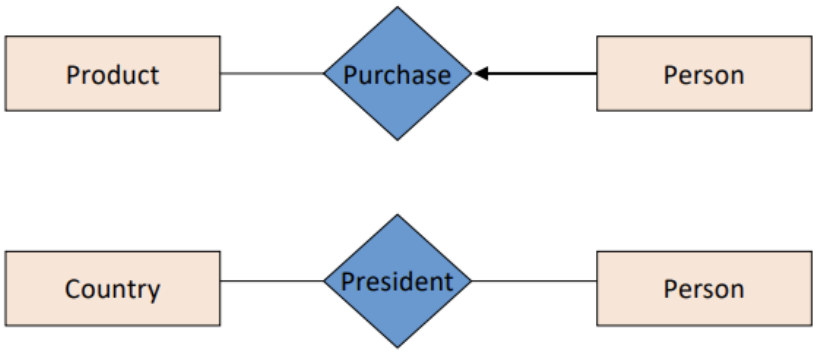
따라서 이런 방법은 문제의 해석에 따라서 복잡해 지기 때문에 이런 방식으로는 잘 표현하지 않음.
하지만 위와 같이 표현해야 될 때가 있는데 그 경우는 세 개의 entity가 합치해야 되는 경우에 사용한다.
ex) 3개의 entity가 모여 unique한 관계를 나타낼 수 있을때? <- 개인적인 의견 확실하지 않음.

또 다른 방식으로 아래의 binary 방식으로 relationships은 표현한다.



이 방식은 Entity를 하나더 만들고 두 개의 entity가 하나의 relationship을 가지고 있다.
누가, 어느 매장에서, 어떤 제품을 구매했는지 알고 싶다면 이 모든 관계를 다 통과해야지 얻을 수 있다.
이 방법의 장점은 자유도가 좀 더 높아져 유연한 표현이 가능해짐.

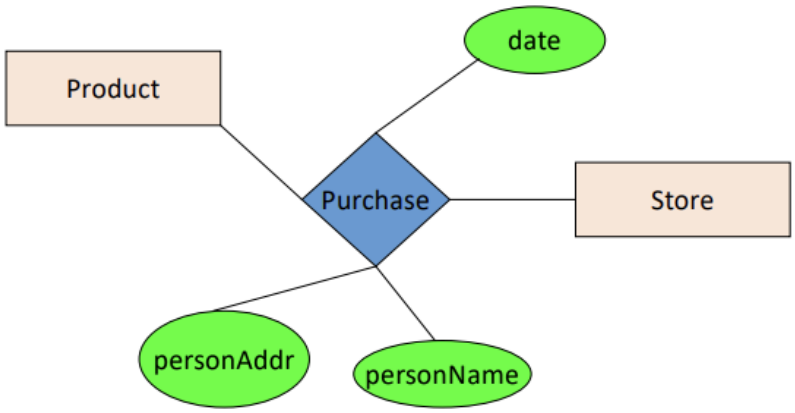
Design Principles



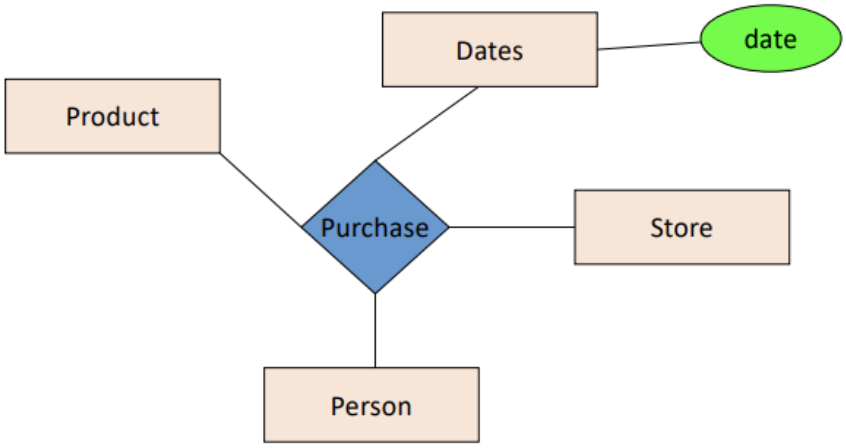
이 관계를 하나씩 해석해보자

먼저 처번째 purchase라는 relationship으로 이뤄진 관계를 해석해보면
person은 하나의 제품만을 구매할수 있고, 하나의 제품은 여러 사람들에게 구매되어질 수 있다.

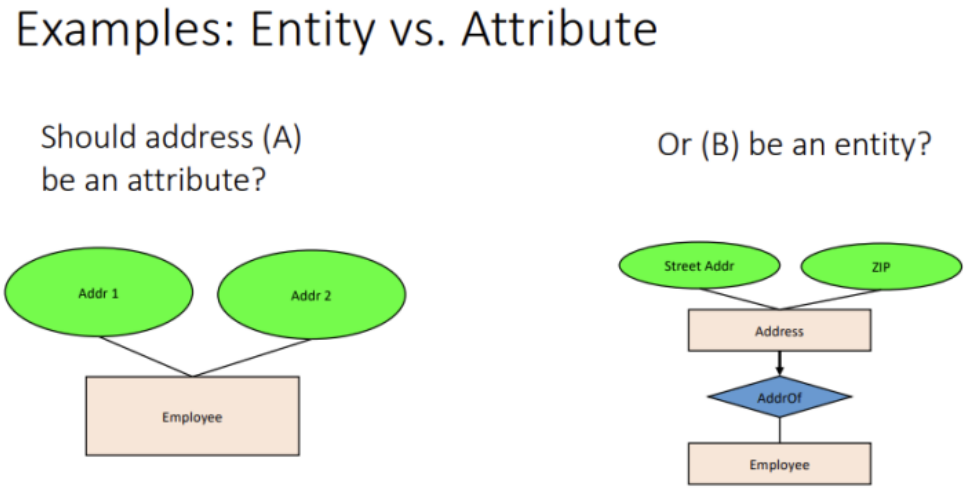
두번째 President라는 relationship으로 이뤄진 관계를 해석해보면
한사람이 여러 나라의 대통령이될 수 있고, 한 나라의 대통령이 여려명일 수 있다.



이 자료에서 잘못된 점은
purchase라는 relationship에 있는 attribute 때문이다.
attribute의 존재는 문제가 되지 않지만, 사람과 연관된 attribute가 두 개나 있기 때문에 문제가 된다.
이런 경우에는 attribute로 사람에 대한 속성을 나타내는 것 보다는 person이라는 entity를 하나더 만들어서 relationship을 형성하는 것이 더 이돕다.

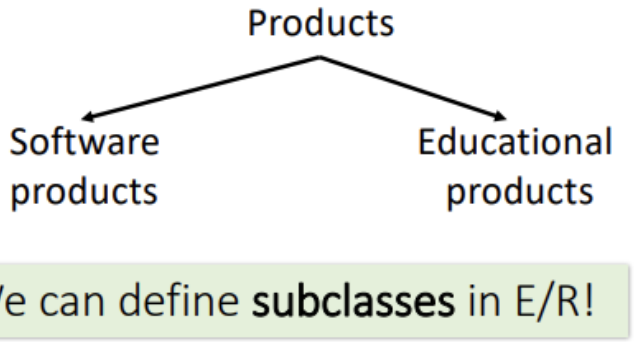


하나의 attribute를 가진 entity를 만드는 것을 안좋다.
그러므로 date라는 entity를 지우고 purchase의 attribute로 date를 붙여야 된다.

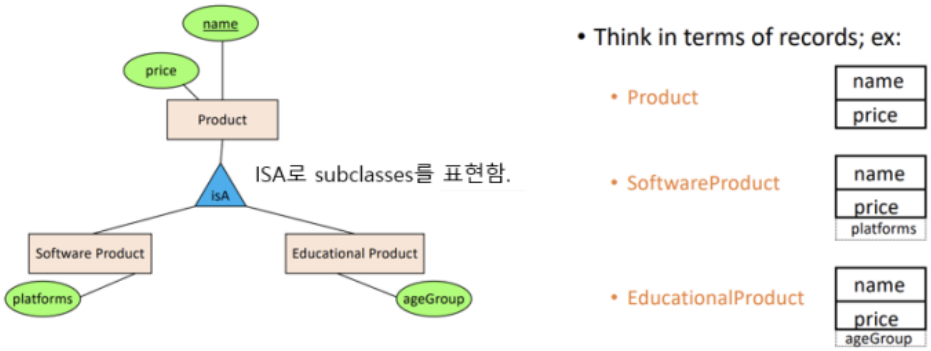


위 그림에서 A그림은 employee entity가 attribute들로 add1,add2을 가지는데 이 표현은 근로자가 두 개의 주소를 가지고 있다는 전제에서 시작하기 때문에, 주소가 한 개인사람 또는 3개 이상인 사람이 있기 때문에 별로 좋은 표현 방법이 아니다.
또 주소에 대한 여러 속성들을 추가해야될 때 이 표현은 추가하기 어려운 방법이다.
반면에 그림 B를 해석해보면 한 개의 주소는 한 사람이 가지고 있고, 한 사람은 여러개의 주소를 가지고 있을 수 있다.
즉 이 표현방법은 주소가 한 개인 사람과 여러개인 사람을 모두 포함할 수 있으므로 자유도와 유연성이 더 높은 표현 방법이다.

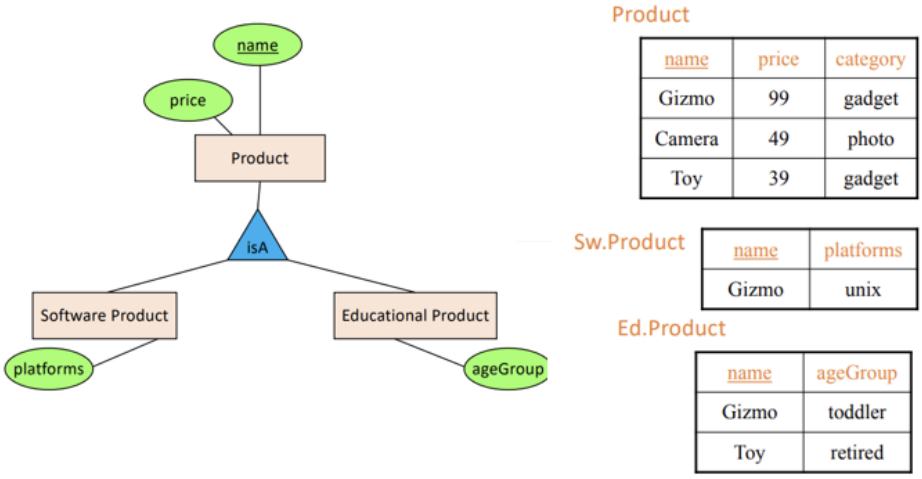
Modeling Subclasses



위와 같이 products라는 entity가 있는데 이 entitty를 좀 더 세분화 하고 싶을때, 여기서는 software 제품과 교육용 제품으로 세분화 하고 싶어하며, 이 때 subclasses를 정의할 수 있다.



ISA를 통해 subclasses은 표현 가능하면
ISA로 표현된 subclasses은 두 값 중 하나로만 표현이 가능하다.
즉 소프트웨어제품에 대한 이름과 가격 플랫폼을 표현하거나
교육제품에 대한 이름과 가격 나이대를 표현하거나 둘 중에 하나이다.

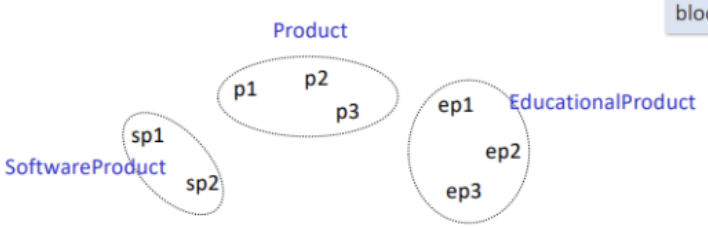


이 관계에서 software product와 educational product entity는 unique하지 못하기 때문에 유일하게 구분하기 위해서는 상위클래스에 있는 primary key를 참조해서 table을 구성 해야된다.
이런 subclass관계는 union이랑 비슷하다.

Difference between OO and E/R inheritance

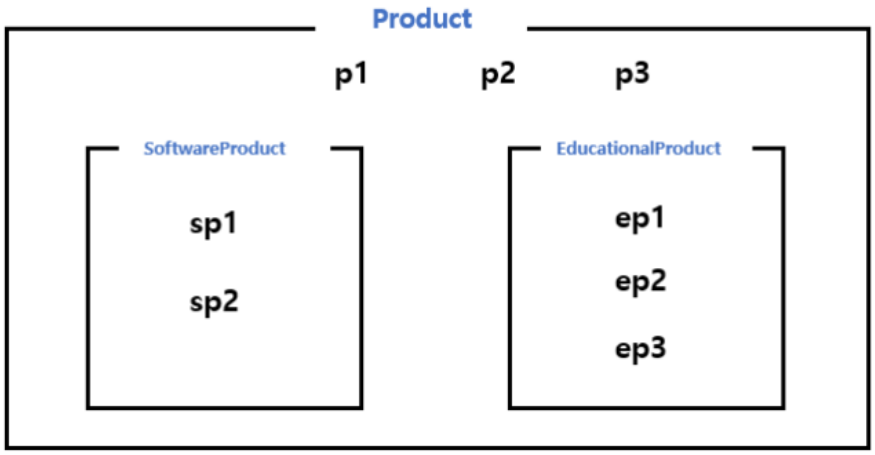
• OO: Classes are disjoint (same for Java, C++)

OO = **Object Oriented**. E.g. classes as fundamental building block, etc...



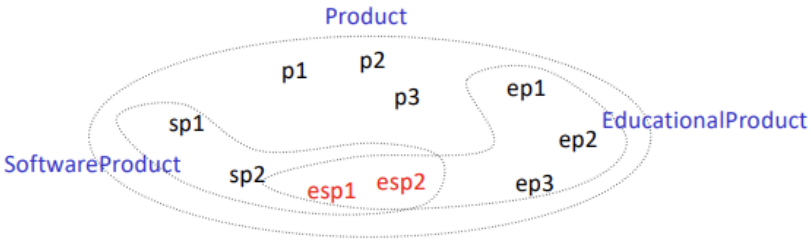
그림과 같이 자바나 C++에서는 각각의 entity가 독립적으로 존재한다.

• E/R: entity sets overlap



하지만 E/R 에서는 Product라는 집합안에 부분집합으로 software, educationalproduct가 들어 있다.

We have three entity sets, but four different kinds of objects



No need for multiple inheritance in E/R

이 그림에서는 세 개의 entity set이 4개의 다든 객체를 나타낸다.

esp1, esp2는 softwareProduct이면서 EducationalProduct이다.

즉, 두 개의 subclass가 같은 개체를 포함 하는 것인데, 이 여부를 결정하는 것을 중첩 제약조건이라고 한다.

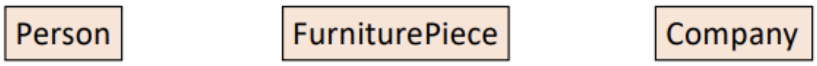
따라서 중첩 제약조건을 허용하면 중첩이 되어도 되며, 따로 언급이 없으면, 개체집합들은 중첩하지 않는다고 간주함

IsA Review

- If we declare ***A IsA B*** then every **A** is a **B**
- We use IsA to
  - Add descriptive attributes to a subclass
  - To identify entities that participate in a relationship
- **No need for multiple inheritance**

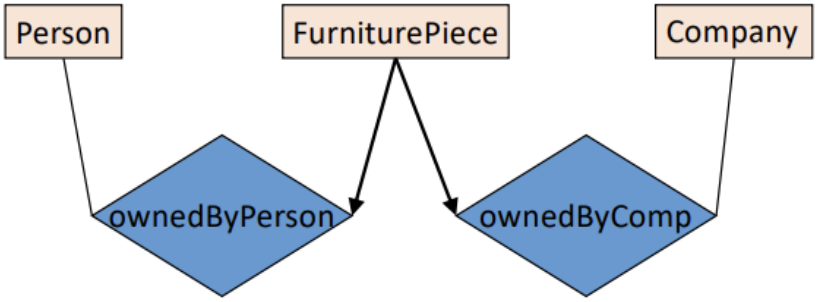
- IsA는 A는 B의 일부분이라는 상속 관계를 나타냄 (A⊂B)
- IsA를 사용하는 이유는 subclass에 attribute를 묘사하기 위해서다.
- 또 하나의 relationship에 참여하는 entity들은 식별하기 위해서 사용한다.

Modeling UnionTypes With Subclasses

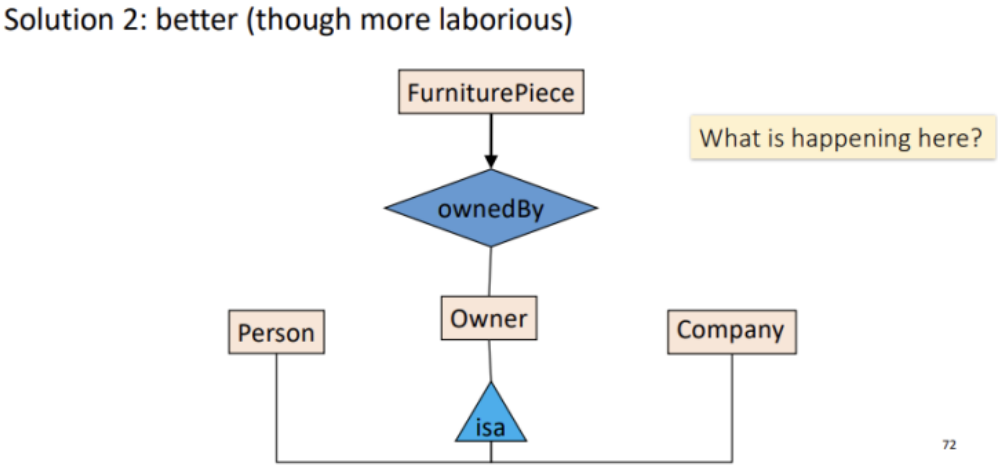


Q : Suppose each piece of furniture is owned either by a person, or by a company.
각각의 가구가 개인 소유이거나 회사 소유라고 가정해 보자.

Solution 1. Acceptable, but imperfect (What's wrong ?)



이렇게 표현해서 생기는 문제는
사람과 회사가 같은 가구를 가지는 경우가 발생할 수 있다.



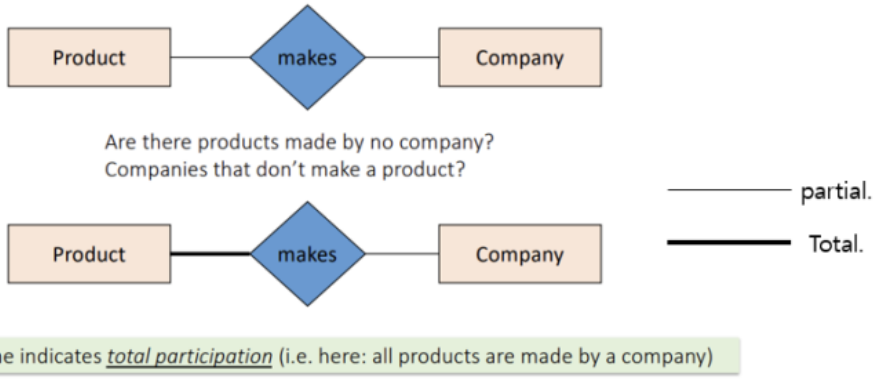
이렇게 부모자식간의 상속관계로 표현하면
이때는 owner라는 entity를 통해 가구의 주인 명단을 다 가지고 있고, subclass를 해당 소유주가 회사인지 개인인지 확인 가능하다.
또 다중상속이 가능하기 때문에 회사와 개인이 똑같은 물건의 주인일 수 도 있다.

Constraints in E/R Diagrams

- Keys: Implicit constraints on uniqueness of entities
  - *Ex: An SSN uniquely identifies a person*
- Single-value constraints:
  - *Ex: a person can have only one father*
- Referential integrity constraints: Referenced entities must exist
  - *Ex: if you work for a company, it must exist in the database*
- Other constraints:
  - *Ex: peoples' ages are between 0 and 150*

E/R 다이어그램의 부분에서 제약조건은 찾는 방법은 아래와 같다.
- Key : entity에 밑줄을 그어서 표현함.
- Single-value constraints : 유일한 값을 가지게 만드는 제약조건 (1:n, 1:1 ...)
- Referential integrity constraints : 참조 무결성 제약조건이다.
 이 말은 primary key와 foreign key의 값이 같아야 된다는 뜻이다.
여기서 foreign key에 있는 값이 primary key에 없으면 안된다.
이런문제는 삽입과 삭제시 발생하는데 삽입할때는 primary key에 해당 값이 존재하는지 확인해야하며
primary key에서 삭제할 때에는 foreign key에 해당 값이 없는지 확인하고 삭제해야 한다.

Participation Constraints : Partial vs. Total

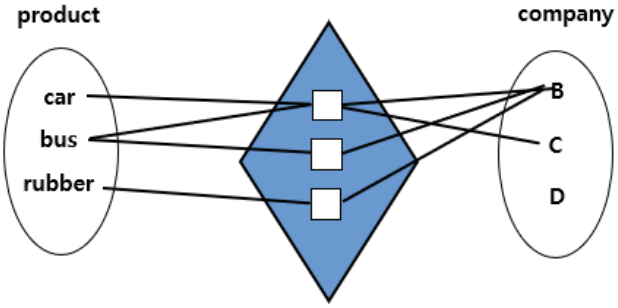


참여 제약조건(Participation Constrainst) 이라고도 한다.

관계집합(relationship)에 entityset의 참여는 전체적(total)이라고 하고, 전체적이 아닌 참여도를 부분적(partial)이라고 한다.

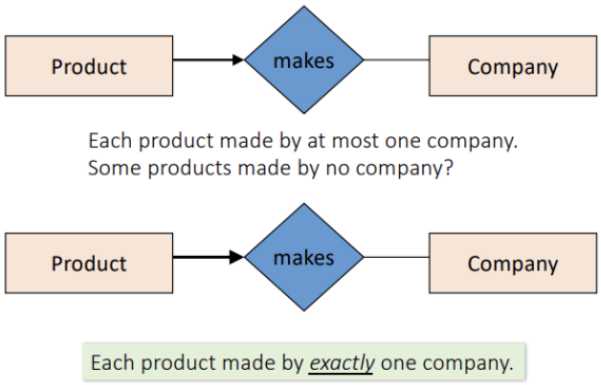
굵은 선이 의미하는 것은 total , 얇은 선이 의미하는 것은 partial 이다.

여기서 굵은 선이 있으면 한 제품은 꼭 만든회사가 있어야 된다라는 의미가 된다. (따라서 product 전체가 이 관계에 참여)



하지만 지금과 같은 경우는 makes라는 관계에서 모든 회사가 하나의 제품을 만드는 것은 아니므로 company의 참여는 부분적(partial) 이다.

Referential Integrity Constraints



여기서 위의 그림은 문제가 있다 왜냐하면 제품은 공장 등에서 생산이 되는것인데 얇은선으로 표기했기 때문에 제품의 일부는 제조한 회사가 누구인지 알 수 없다는 것을 나타내는데 이 관계는 올바르지 않다.

반대로 아래의 굵은 선은 모든 제품은 특정한 한 회사에서 만들어졌다라는 것을 의미하고, 또 특정한 한 회사는 제품을 여러개 만들 수 도 있고 안만들수도 있다는 것을 의미한다.

Weak Entity Sets

약개체는 굵은 실선으로 표현된다 .

Week Entity Set(약개체)은 그것의 attribute 일부와 identifying owner(식별 소유자)에 해당하는 entity set의 primary key를 결합하여야만 유일하게 식별할 수 있다.

약개체의 Key는 단독으로 primary key가 될 수 없고, 유일하게 식별되기 위해 소유자 개체집합의 primary key와 함께 조합되어 사용되어짐.

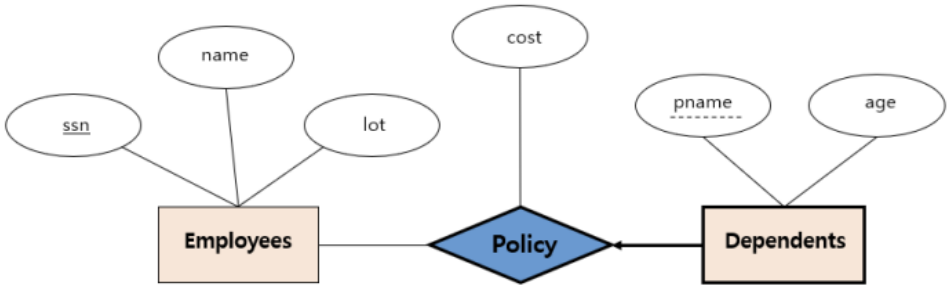
강개체에 대하여 하나의 약개체를 유일하게 식별하는 약개체집합의 attribute의 부분집합은 약개체 집합의 부분키(partial key)라고 부른다.

Partial key는 파단선으로 밑줄이 그어져 있다.

약개체의 attribute 중 강개체의 특정 개체에 대해서 유일한 값을 갖는 attribute들의 집합

또 weak entity set은 다음 cosntraints 만족해야 됨.

- 소유자 개체집합(강개체)과 약개체집합은 1:n(일-대-다) 관계집합으로 참여해야 함.
- 하나의 소유자 개체는 여러 약개체와 연관되지만, 각각의 약개체는 오직 하나의 소유자를 가짐.
- 이러한 관계집합은 해당 약개체집합의 식별 관계집합(identifying relationship set)이라고 함.
- 약개체집합은 식별 관계집합에 전체적으로 참여하여야 한다. (즉 굵은 선(total participation)!!)



위 그림을 보면

Dependents entity의 전체참여를 나타내기 위해 굵은 선으로 연결하였고,

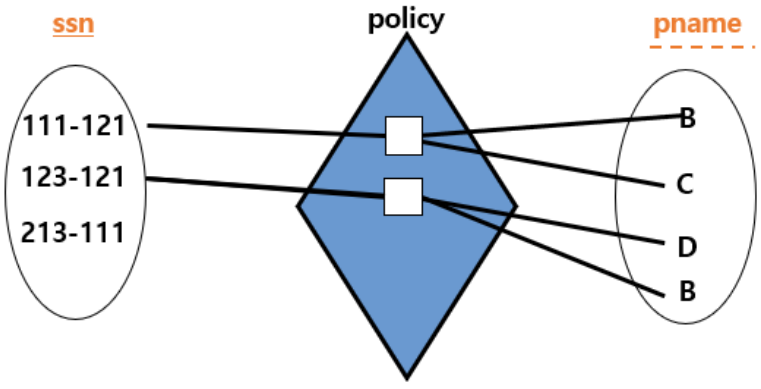
Policy로 가는 방향으로 화살표가 되어있기 때문에 한명의 피부양자는 한명의 직원에 의해 계약되고, 한명의 직원은 여러명의 피부양자가 있을 수도 있고, 없을 수 도 있다.

Dependents가 약개체집합이며 Policy가 그것의 식별 관계집합이라는 사실을 강조하기 위해, 둘 다 굴고 진한 선으로 그린다.

유일하게 식별하기 위해 emloyees의 primary key인 ssn과 Dependents의 partial key를 조합하여 유일함을 표현한다.

Dependents의 부분키가 pname이라는 것을 알 수 있도록 파선으로 밑줄을 긋고,

이 의미는 두 피부양자(dependents)가 같은 pname값을 가질 수 없다는 것을 의미. <- 이부분은 사실 확인이 필요해 보임



제 생각은 중복값을 가질 수 있으나 이 중복값이 primary key로 인해 유일성을 만족해야 된다는 것 같다.

E/R Summary

- E/R diagrams are a visual syntax that allows technical and non-technical people to talk
  - For conceptual design
- Basic constructs: **entity**, **relationship**, and **attributes**
- A good design is faithful to the constraints of the application, but not overzealous

참고 사이트 입니다. 두 사이트 이번 내용을 다 잘정리해준 사이트 입니다.

https://blog.naver.com/ksw6889/222253804618

DB 데이터베이스 설계(관계의 대응수, 일대일, 일대다, 다대다, …
관계의 대응수 - 관계집합에서 각 개체들이 참여할 수 있는 대응의 개수 대…
blog.naver.com

https://dad-rock.tistory.com/373

[Database] ER Model | ER 모델
ER Model ER 모델 - 실세계 조직체에 대한 데이터…
dad-rock.tistory.com

Prof. Lee Seungjin님이 제공해준 Data Processing Systems의 자료를 활용하여 제작하였습니다.
또 Database Mangement System 책을 참고했습니다.