The E/R Model

Prof. Hyuk-Yoon Kwon
https://sites.google.com/view/seoultech-bigdata

Today's Lecture

1. **E/R Basics: Entities & Relations**
2. **E/R Design considerations**
3. **Advanced E/R Concepts**

Most parts are based on slides used in Stanford (http://web.stanford.edu/class/cs145)

Database Design Process

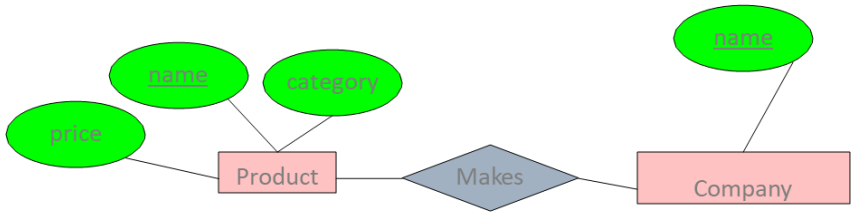
1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

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?

Technical and non-technical people are involved

E/R Model & Diagrams used



2. Conceptual Design

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

This is where E/R fits in.

3. More

- Logical Database Design
- Physical Database Design
- Security Design

Database design: Why do we need it?

- Agree on structure of the database
  before deciding on a particular implementation

Consider issues such as:

- What entities to model
- How entities are related
- What constraints exist in the domain
- How to achieve good designs

Several formalisms exist

- We discuss one flavor of E/R diagrams

E/R is a *visual syntax* for DB design which is ***precise enough*** for technical points, but ***abstracted enough*** for non-technical people

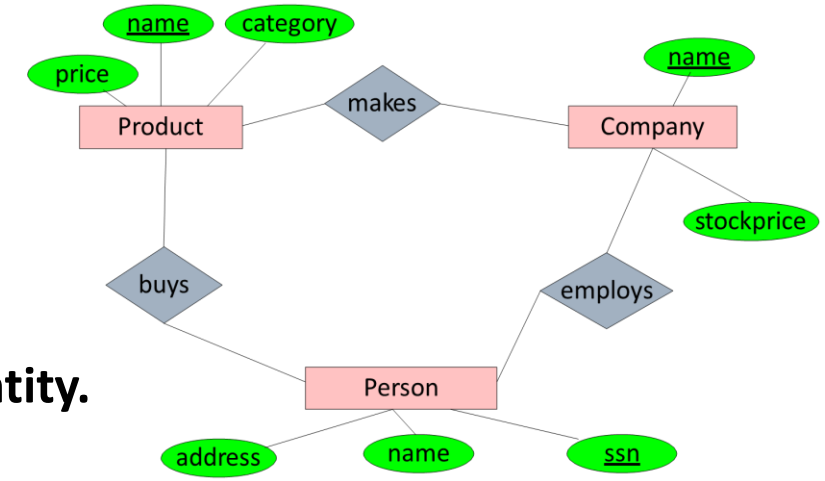
Entities and Entity Sets

Entities & entity sets are the primitive unit of the E/R model

- Entities are the individual objects, which are members of entity sets
  - Ex: A specific person or product
- Entity sets are the *classes* or *types* of objects in our model
  - *Entity sets represent the sets of all possible entities*
  - Ex: Person, Product
  - *These are what is shown in E/R diagrams - as rectangles*

A relationship is between two entities

The R in E/R: Relationships :

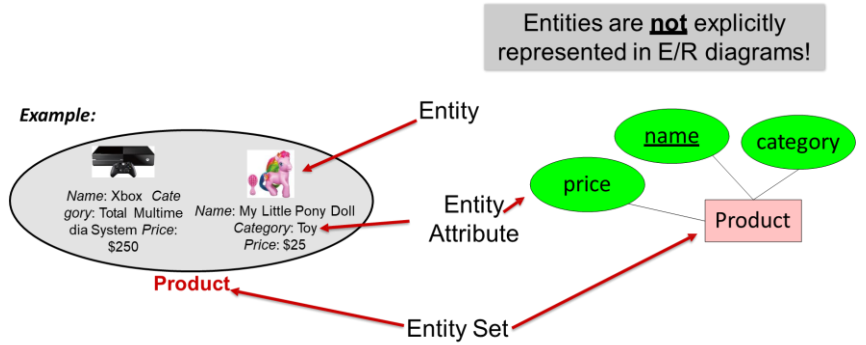


A *key* is a minimal set of attributes that uniquely identifies an entity.

{name, category} is **not** a key (it is not *minimal*).

The E/R model forces us to designate a single **primary** key, though there may be multiple candidate keys

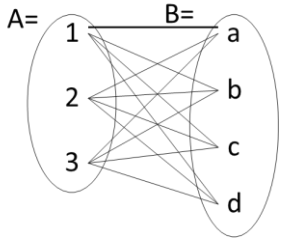
Note: no formal way to specify *multiple* keys in E/R diagrams…

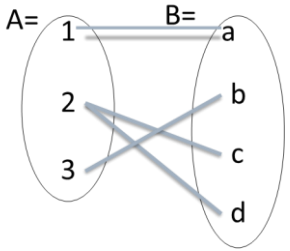


Makes relationship : a *subset* of Product × Company

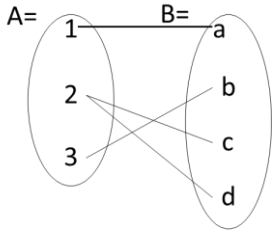
What is a Relationship?

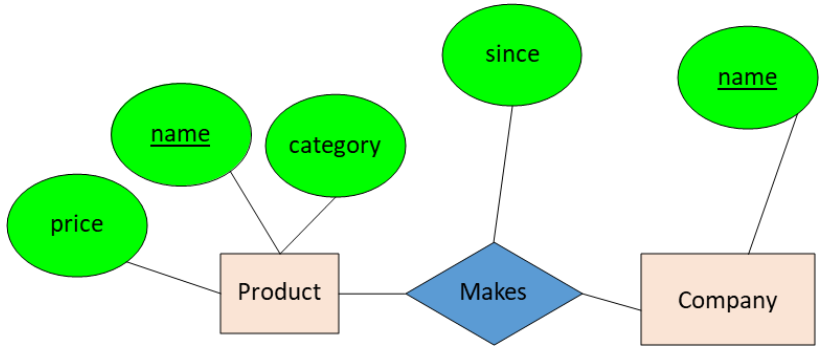
- ***A mathematical definition:***





  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d}*
  - A x B (the ***cross-product***) is the set of all pairs (a,b)
    - $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$
  - **We define a relationship to be a subset of A x B**
    - *R = {(1,a), (2,c), (2,d), (3,b)}*





A **relationship** between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C***, with tuples uniquely identified by ***P and C's keys***

- **There can only be one relationship for every unique combination of entities**
- **This also means that the relationship is uniquely determined by the keys of its entities**
- ***Example: the "key" for Makes (to right) is {Product.name, Company.name}***

What is a Relationship?

Company

name
GizmoWorks
GadgetCorp

Product

name	category	price
Gizmo	Electronics	$9.99
GizmoLite	Electronics	$7.50
Gadget	Toys	$5.50

Company C × Product P

C.name	P.name	P.category	P.price
GizmoWorks	Gizmo	Electronics	$9.99
GizmoWorks	GizmoLite	Electronics	$7.50
GizmoWorks	Gadget	Toys	$5.50
GadgetCorp	Gizmo	Electronics	$9.99
GadgetCorp	GizmoLite	Electronics	$7.50
GadgetCorp	Gadget	Toys	$5.50

Makes

C.name	P.name
GizmoWorks	Gizmo
GizmoWorks	GizmoLite
GadgetCorp	Gadget

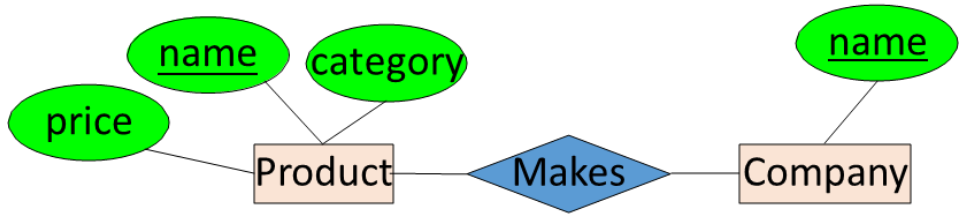
A relationship between entity sets P and C is a *subset of all possible pairs of entities in P and C,* with tuples uniquely identified by *P and C's keys*

Relationships may have attributes as well.

For example: "since" records when company started making a product

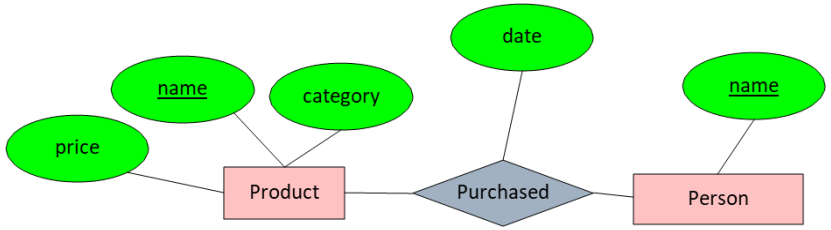
Note #1: "*since*" is implicitly unique per pair here! Why?

Note #2: Why not "how long"?



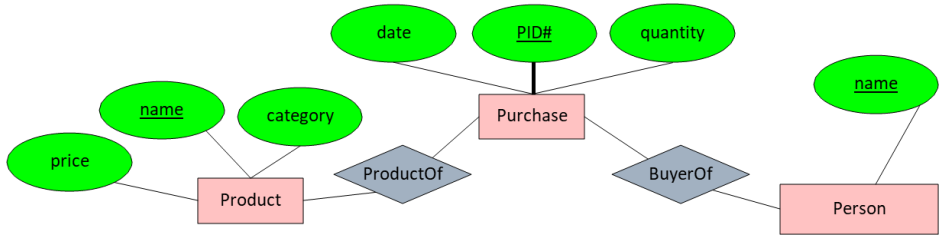
Decision: Relationship vs. Entity?

- **Q: What does this say?**



- **A: A person can only buy a specific product once (on one date)**

Modeling something as a relationship makes it unique;
what if not appropriate?

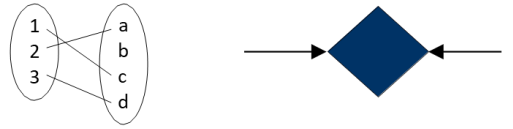


- ***Now we can have multiple purchases per product, person pair!***

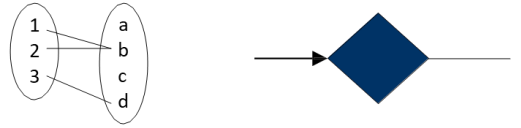
We can always use **a new entity** instead of a relationship
For example, to permit multiple instances of each entity combination!

Multiplicity of E/R Relationships

One-to-one:



Many-to-one:



One-to-many:

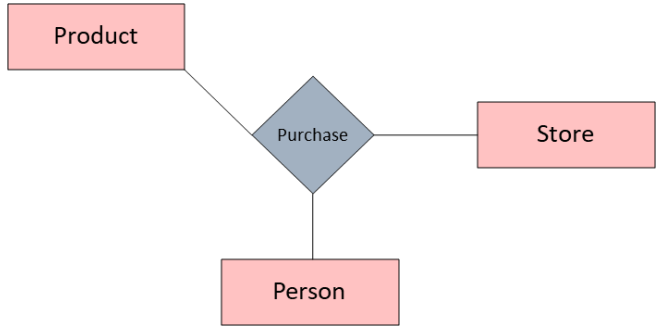
Many-to-many:

X -> Y : **function mapping from X to Y exists**
(recall the definition of a function)

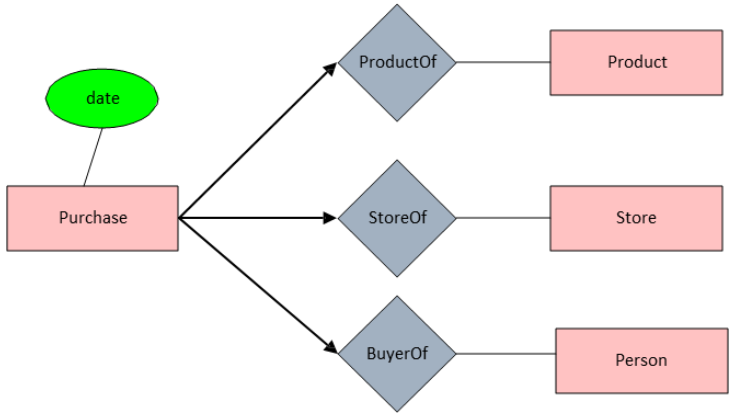
Decision: Multi-way or New Entity + Binary?

Converting Multi-way Relationships to Binary

(A) Multi-way Relationship



(B) Entity + Binary



- (A) is useful when a relationship really is between multiple entities
  - *Ex: A three-party legal contract*
- *Covered earlier:* (B) is useful if we want to have multiple instances of the "relationship" per entity combination
- (B) is also useful when we want to add details (constraints or attributes) to the relationship
  - "A person who shops in only one store"
  - "How long a person has been shopping at a store"

1) 다중 관계 : 여러 Entity 간의 복잡한 관계 표현

2) Entity + Binary : 다양한 제약 조건 & 속성 추가

⇒ Data Modeling 목적 & 관계 복잡도 & 필요에 따라 선택

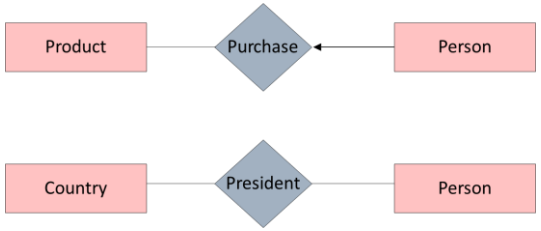
Multiple purchases per (product, store, person) combo possible here!

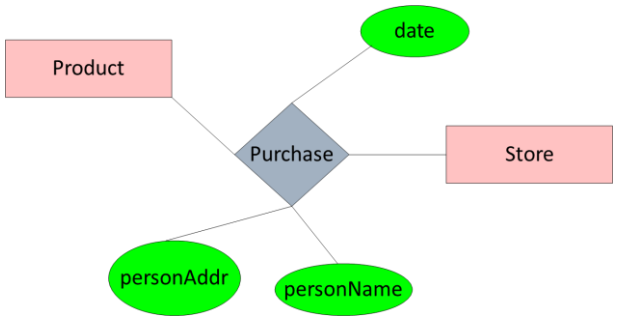
We can add more-fine- grained constraints here!

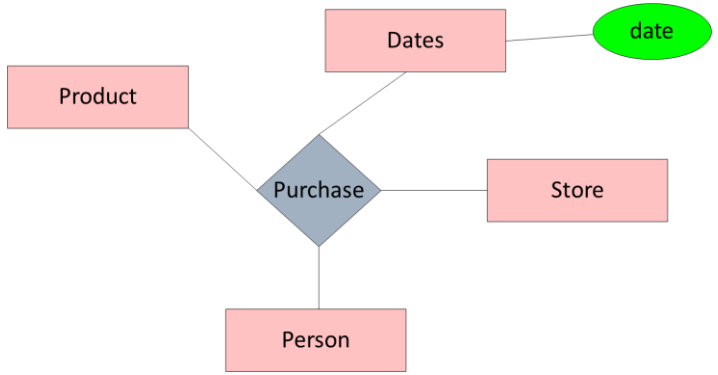
Should we use a single **multi-way relationship** or a ***new entity with binary relations?***

Design Principles

What's wrong with these examples?



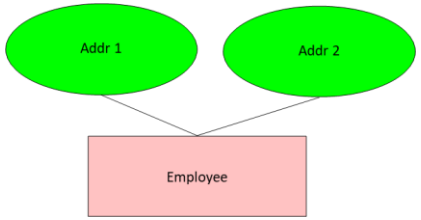




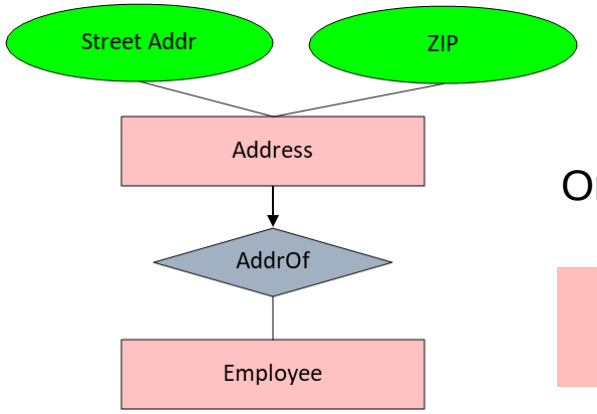
Examples: Entity vs. Attribute

How do we handle employees with multiple addresses here?

How do we handle addresses where internal structure of the address (e.g. zip code, state) is useful?



Should address (A) be an attribute?

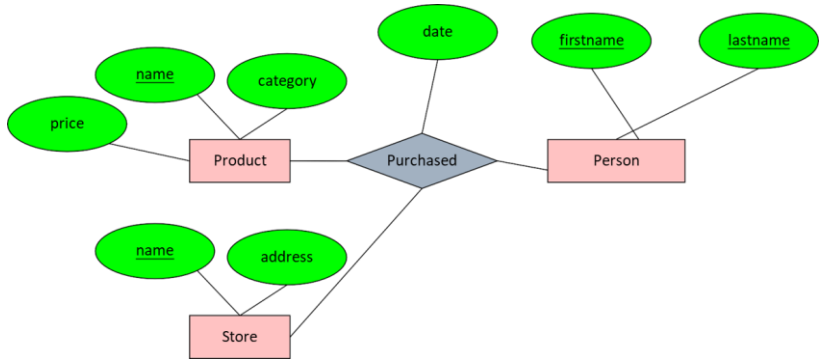


Or (B) be an entity?

In general, when we want to record several values, we choose new entity

From E/R Diagram to Relational Schema

Key concept : Both ***Entity sets*** and ***Relationships*** become relations (tables in RDBMS)



How do we represent this as a relational schema?

```
CREATE TABLE Product(
  name     CHAR(50) PRIMARY KEY,
  price    DOUBLE,
  category VARCHAR(30)
)
```

Product

Name	Price	Category
Gizmo1	99.99	Camera
Gizmo2	19.99	Edible

```
CREATE TABLE Purchased(
  name CHAR(50),
  firstname CHAR(50),
  lastname  CHAR(50),
  date      DATE,
  PRIMARY KEY (name, firstname, lastname),
  FOREIGN KEY (name)
          REFERENCES Product,
  FOREIGN KEY (firstname, lastname)
          REFERENCES Person
)
```

Purchased

Pname	Firstname	Lastname	Date
Gizmo1	Bob	Alice	01/01/15
Gizmo2	Alice	Bob	01/03/15
Gizmo1	Joe	Smith	01/05/15

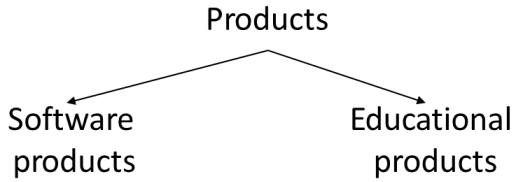
- An entity set becomes a relation (multiset of tuples / table)
  - Each tuple = one entity
  - Each tuple is composed of the entity's attributes, and has the same primary key

- A relation between entity sets $A_1, \ldots, A_N$ *also* becomes a multiset of tuples / a table
  - Each row/tuple is one relation, i.e. one unique combination of entities $(a_1,\ldots,a_N)$
  - Each row/tuple is
    - composed of the **union of the entity sets' keys**
    - has the entities' primary keys as foreign keys
    - has the union of the entity sets' keys as primary key

Modeling Subclasses

- Some objects in a class may be special, i.e. worthy of their own class
  - Define a new class?
    - *But what if we want to maintain connection to current class?*
    - Better: define a *subclass*

We can define **subclasses** in E/R!



- Product

name	price	category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

name
price

- SoftwareProduct

name	platforms
Gizmo	unix

name
price
platforms

- EducationalProduct

name	ageGroup
Gizmo	toddler
Toy	retired

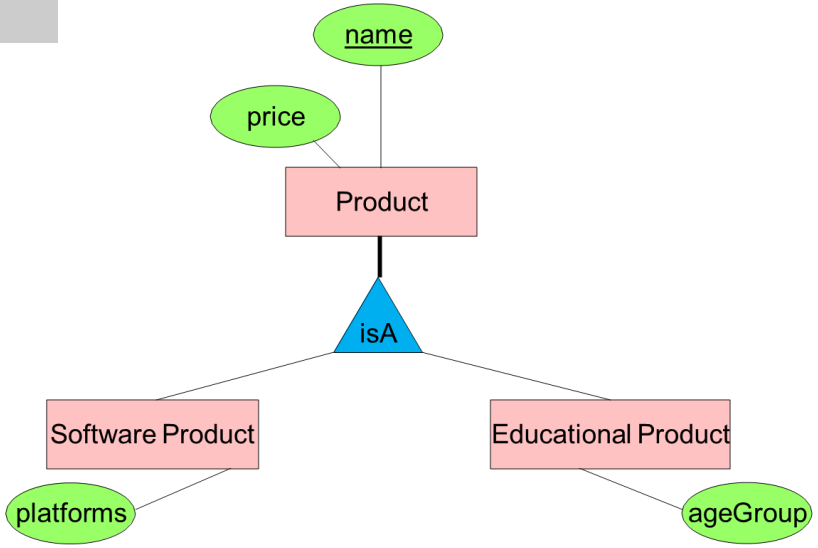
name
price
ageGroup

Child subclasses contain all the attributes of *all* of their parent classes **plus** the new attributes shown attached to them in the E/R diagram

Sub-entity

: Super-entity의 특성 상속받음 + 추가적인 특성 가질 수 있음

* A IsA B : B = parent , A = Child

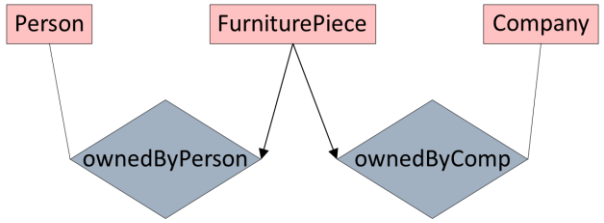


- **If we declare *A IsA B* then every A is a B**
- **We use IsA to Add descriptive attributes to a subclass**

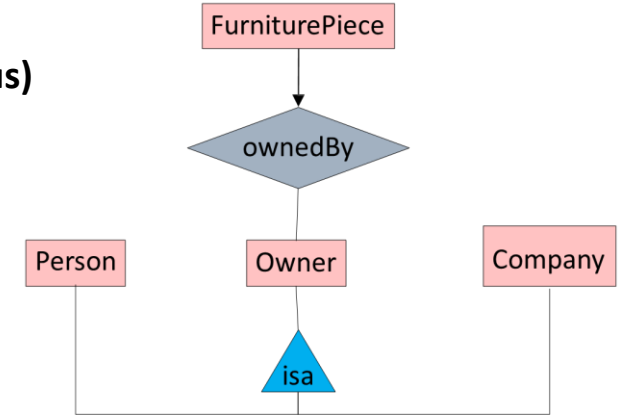
Modeling UnionTypes With Subclasses

Say: each piece of furniture is owned either by a person, or by a company

<u>Solution 1.</u> Acceptable, but imperfect (What's wrong ?)



Solution 2: better (though more laborious)



Constraints in E/R Diagrams

- Participation Constraints: Partial v. Total

참여 제약 조건 : E-R 연결에서 특정 entity가 관계에 얼마나 참여해야 하는가

1) Partial Participation

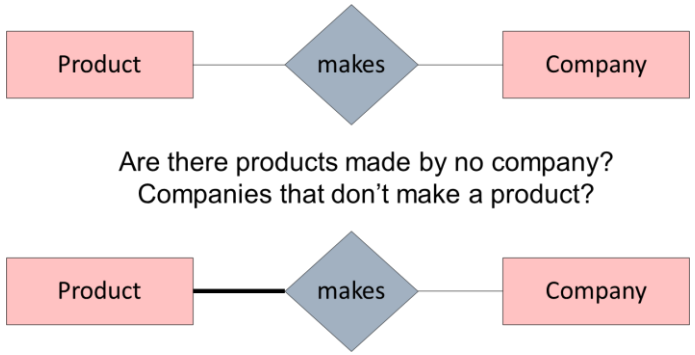
: 관계의 한 쪽 entity가 다른 쪽 entity와 관계에 참여하지 않을 수 있음

ex) Project에 아직 배정되지 않은 직원 있을 수 있음

2) Total Participation

: 두 Entity 모두 반드시 관계에 참여해야 함

ex) 모든 직원은 반드시 한 부서에 속함 & 모든 부서는 1인 이상의 사원 필요

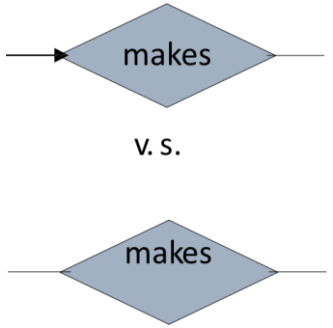


Bold line indicates *<u>total participation</u>* (i.e. here: all products are made by a company)

Constraints in E/R Diagrams

- **Finding constraints is part of the E/R modeling process. Commonly used constraints are:**
  - 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*

Key 제약 조건 : Entity Set의 1 이상 속성 값이 집합 내 모든 Entity에서 고유해야 함

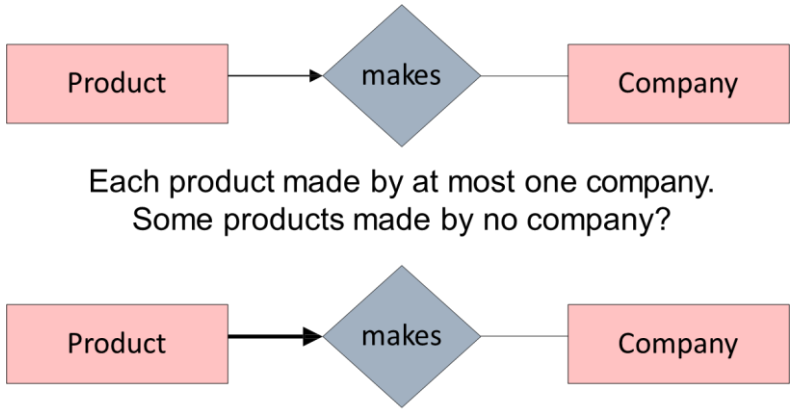


  - Other constraints:
    - *Ex: peoples' ages are between 0 and 150*

단일 값 제약 조건 : Entity가 특정 속성에 대해 1개 값만 가질 수 있음
-> 주로 일대일 관계에서 사용됨

  - Referential integrity constraints: Referenced entities must exist
    - *Ex: if you work for a company, it must exist in the database*

참조 무결성 제약 조건 : 참조된 Entity가 존재해야 함
-> Foreign Key로 참조되는 table의 무결성 보호
= 참조하는 table에서 data가 변경 / 삭제될 때 참조하는 다른 table에서도 data 일관성 유지
1) Foreign Key 값은 참조하는 table의 primary key 값과 일치
2) 참조하는 table primary key value 변경 / 삭제 시,
참조하는 모든 table에서 해당 data 수정/ 삭제 필요
3) 참조되는 table에서 삭제되는 행에 대한 참조 있는 경우 참조하는 행도 삭제해야 함

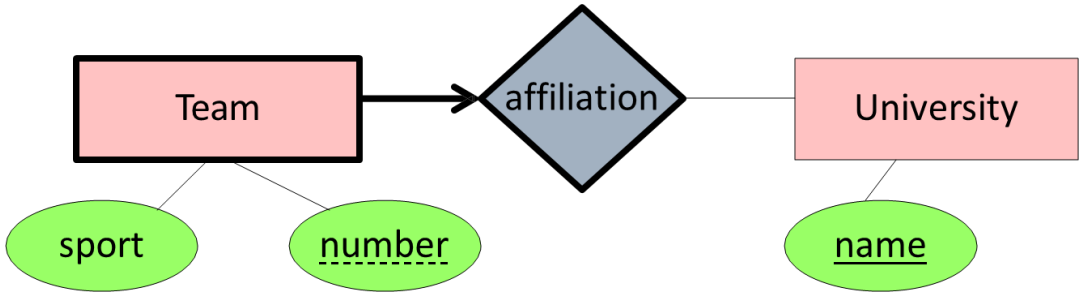


Each product made by *exactly* one company.

Recall FOREIGN KEYs!

Weak Entity Sets

Weak Entity Set
: 다른 Entity의 기본 key에 의존해야 하는 Entity Set
= Weak Entity Set의 기본 key는 다른 Entity의 기본 key에 부착되는 partial key
-> 기본 키를 형성하는 Entity = 소유자 Entity
-> 소유자 Entity의 기본키를 Foreign Key로 사용
-> Weak Entity는 소유자 Entity 존재 시에만 존재 가능



- number is a ***partial key***. (denote with dashed underline).
- University is called the ***identifying owner***.
- Participation in affiliation must be total. Why?

Entity sets are ***weak*** when their key comes from other classes to which they are related.

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**