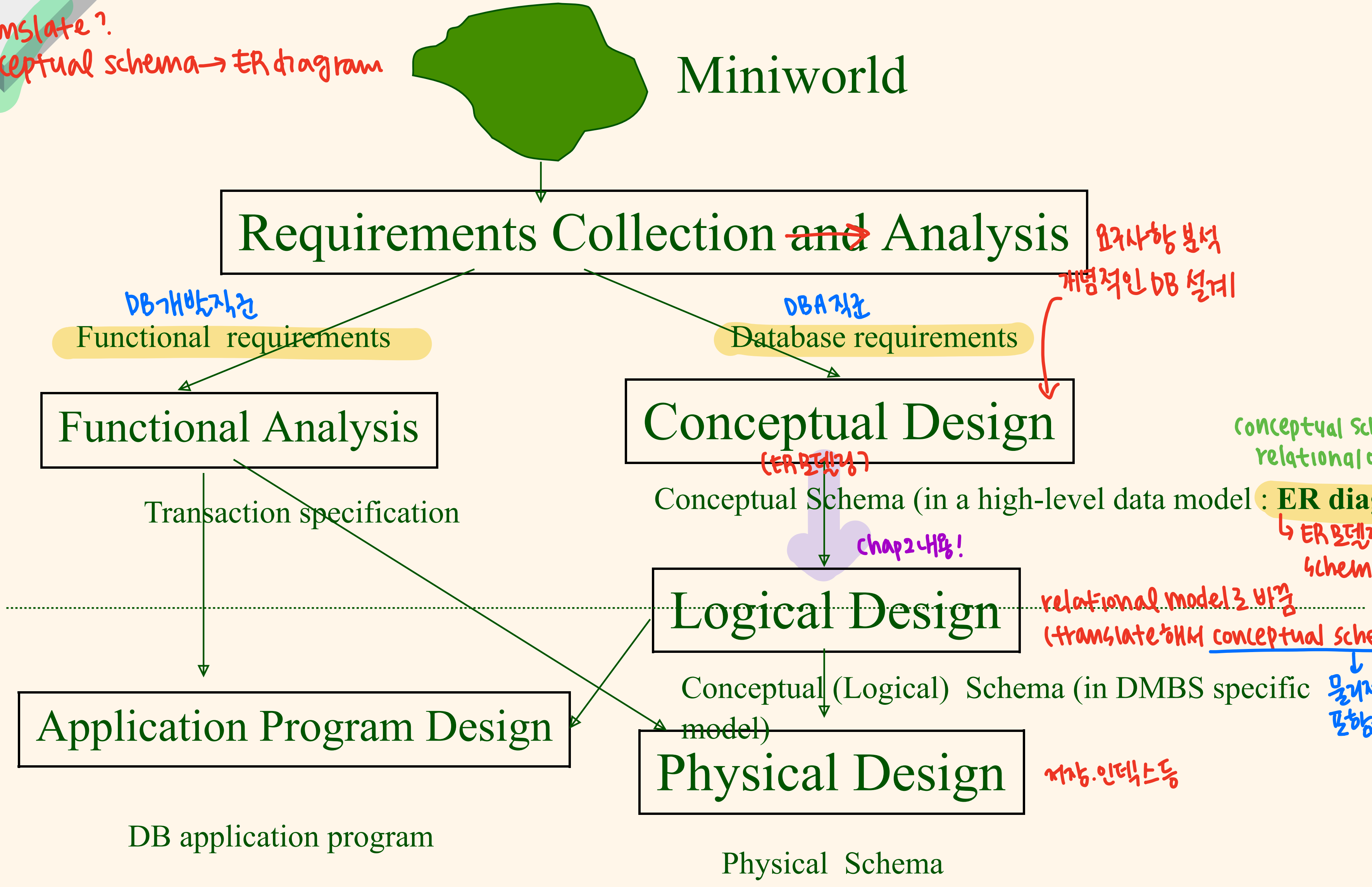
The Entity-Relationship Model ER model

Chapter 2

관계형

Overview of Database Design

translate?
conceptual schema → ER diagram



Miniworld

Requirements Collection ~~and~~ Analysis

요구사항 분석

개념적인 DB 설계

DB개발자관점

Functional requirements

DBA관점

Database requirements

Functional Analysis

Conceptual Design

(ER모델링)

Transaction specification

Conceptual Schema (in a high-level data model : **ER diagram**)

conceptual schema 자체 relational이 아닌건아님

ER모델링으로 만든 schema

chap2내용!

Logical Design

relational model로 바꿈
(translate해서 conceptual schema로)

Conceptual (Logical) Schema (in DMBS specific model)

물리적인요소를 포함하지 않음

Application Program Design

Physical Design

저장.인덱스등

DB application program

Physical Schema

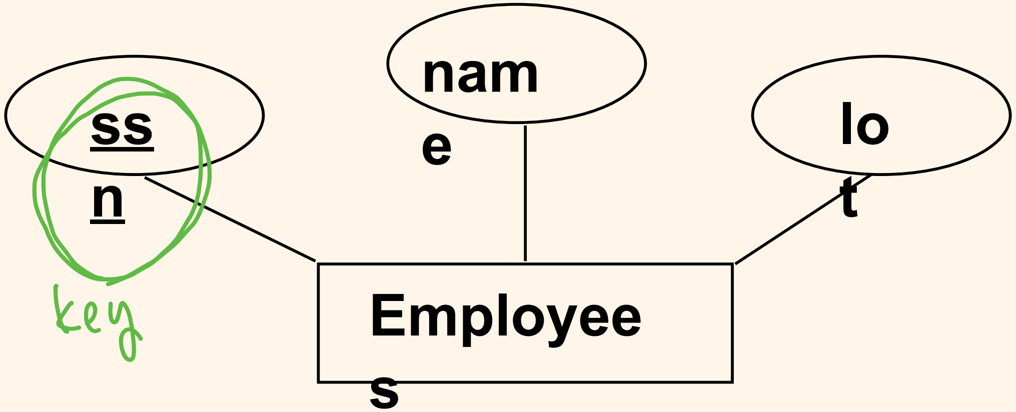
✓ ✓ 스페셜?.

Overview of Database Design

<u>*Conceptual design*</u>: *(ER Model is used at this stage.)*

- What are the *entities* set and *relationships* set in the enterprise? conceptual schema라서 구조를 그려야함.
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hold?
- A database `schema' in the ER Model can and should be represented pictorially (*ER diagrams*).
- Can map an ER diagram into a relational schema.

ER Model Basics (Entity)

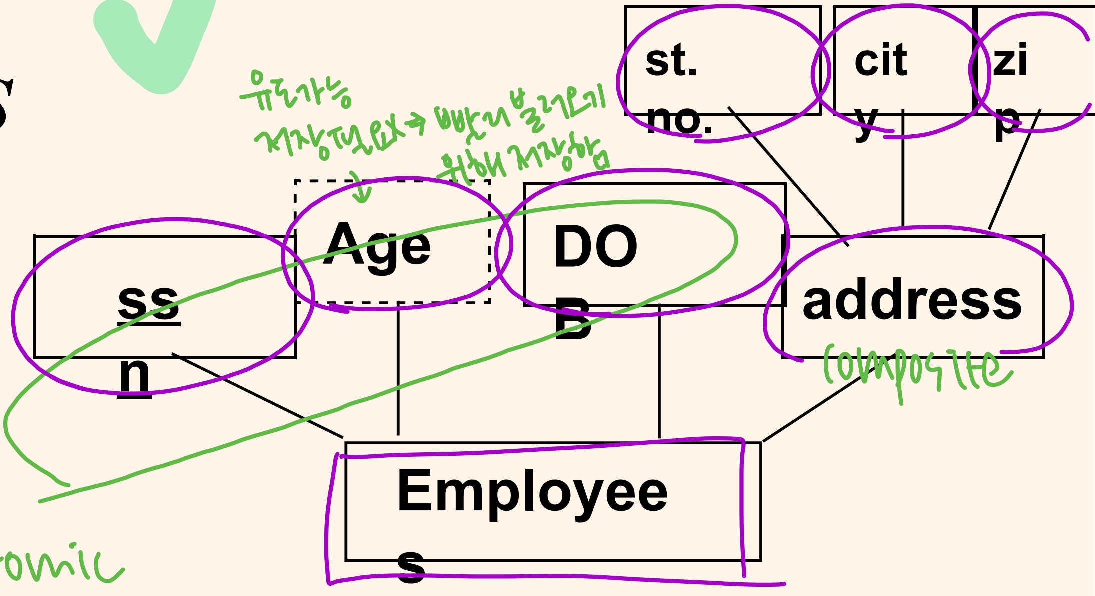


- *Entity*: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a *entity identifier (key)*. → unique
  - Each attribute has a *domain*.
    ↳ data type

entity — object
property method
attribute

ER Model Basics (Entity Cont'd)

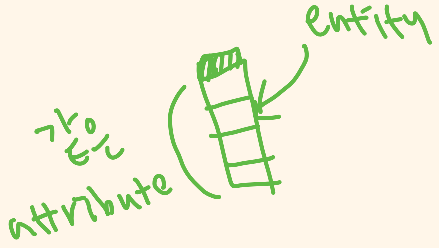
2검오류 →



atomic

- *Attribute*: Property of an entity.
- *Type*:
  - *Atomic vs Composite*
  - *Single-valued vs Multi-valued*
  - *Stored vs Derived*
  - Relational model allows only *atomic* and *single-valued*.

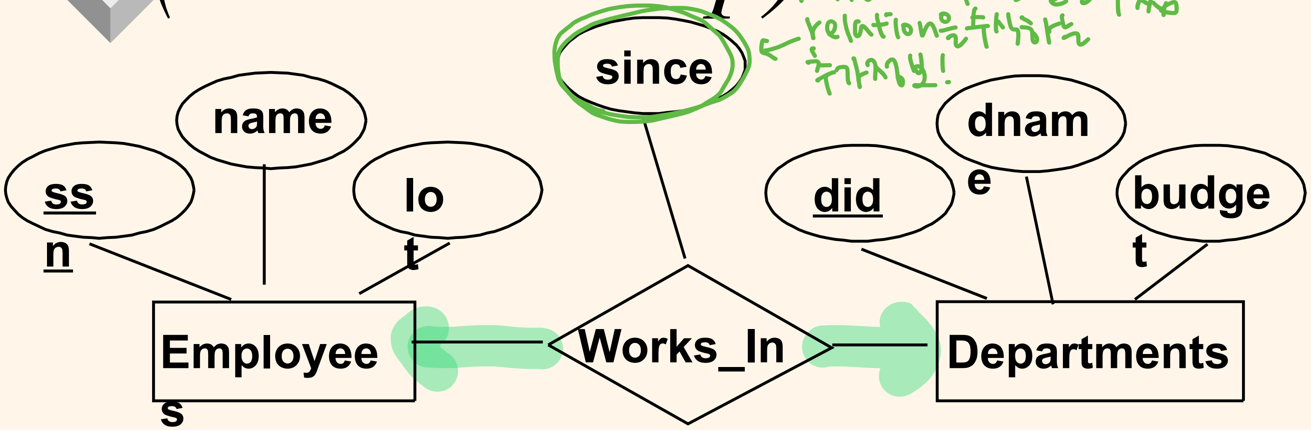
relational model
table 형태
atomic, single-valued only



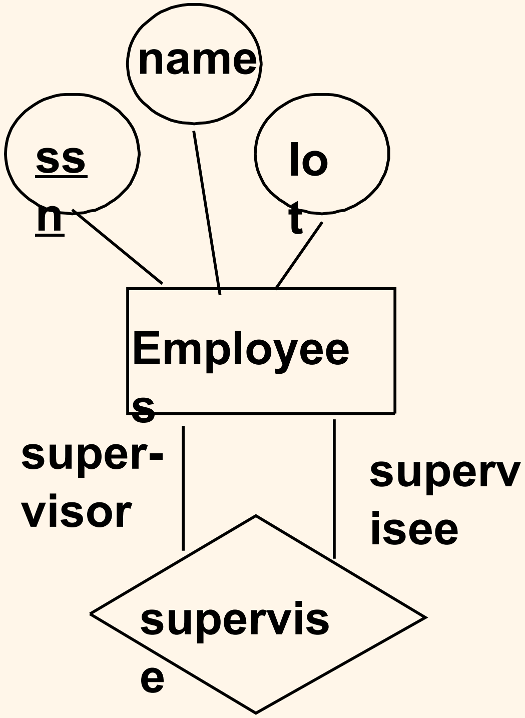
ER Model Basics (Relationship)

① ◇ relationship set
multi 속성을 가진다?
↳ relation이 2개여서?

② since가 Employees에 붙으면?



relationship에도 붙을수있음
relation을 수식하는
추가정보!



employee가 또다른 employee를

Relationship: Association among two or more entities.

E.g., Tom works in Pharmacy department.

Relationship Set: Collection of similar relationships.

An n-ary relationship set R relates n entity sets E1 ... En;

Same entity set could participate in different relationship sets, or in different "roles" in same set.

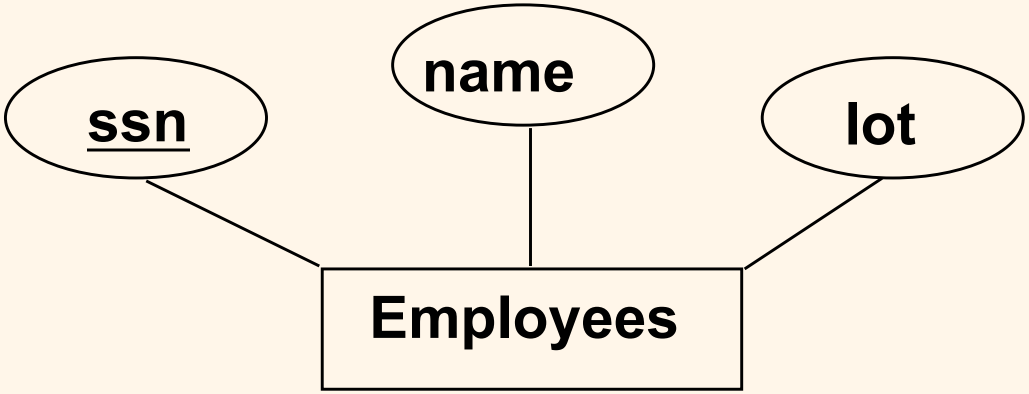
unary
binary
trinary
quainary
⋮

constraint 필요
몇명까지관리할지?

Overview of Database Design

- *Conceptual design*: (*ER Model is used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database `schema' in the ER Model can and should be represented pictorially (*ER diagrams*). 설계도
  - Can map an ER diagram into a relational schema.

ER Model Basics (Entity)



□ *Entity*: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.

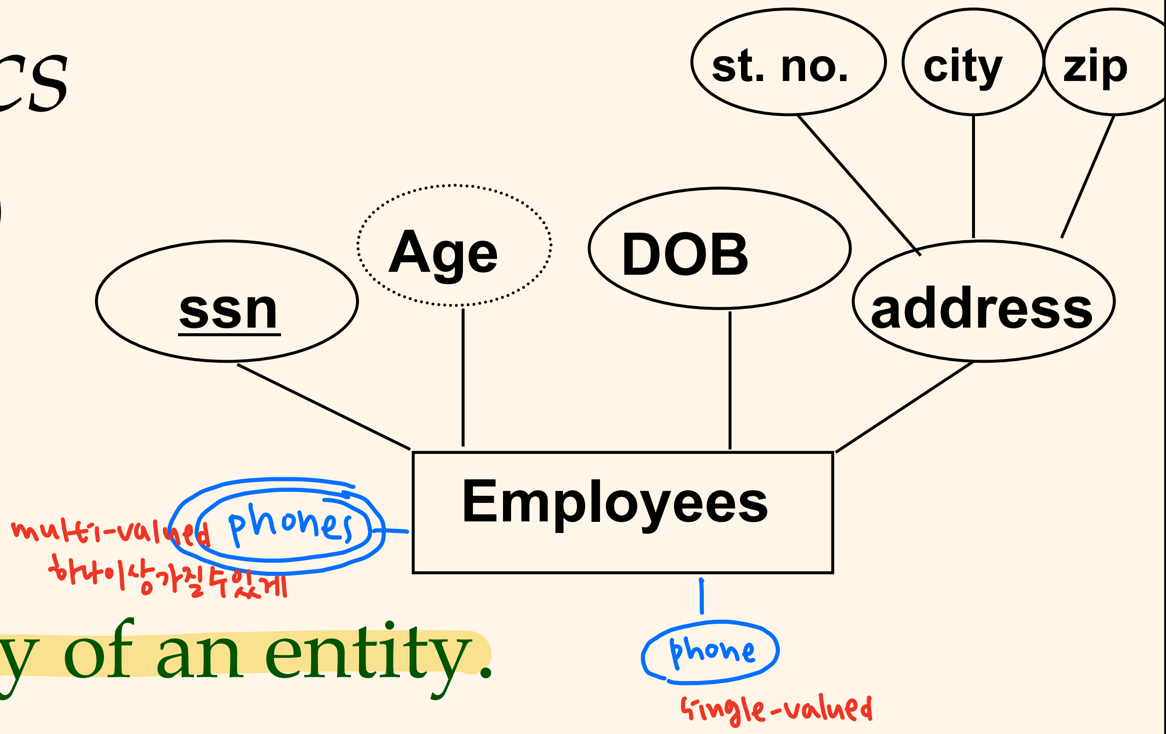
□ *Entity Set*: A collection of similar entities. E.g., all employees.

대부분 다 set을 의미하지만 생략함

- All entities in an entity set have the same set of attributes.
- Each entity set has a *entity identifier (key)*.
- Each attribute has a *domain*.

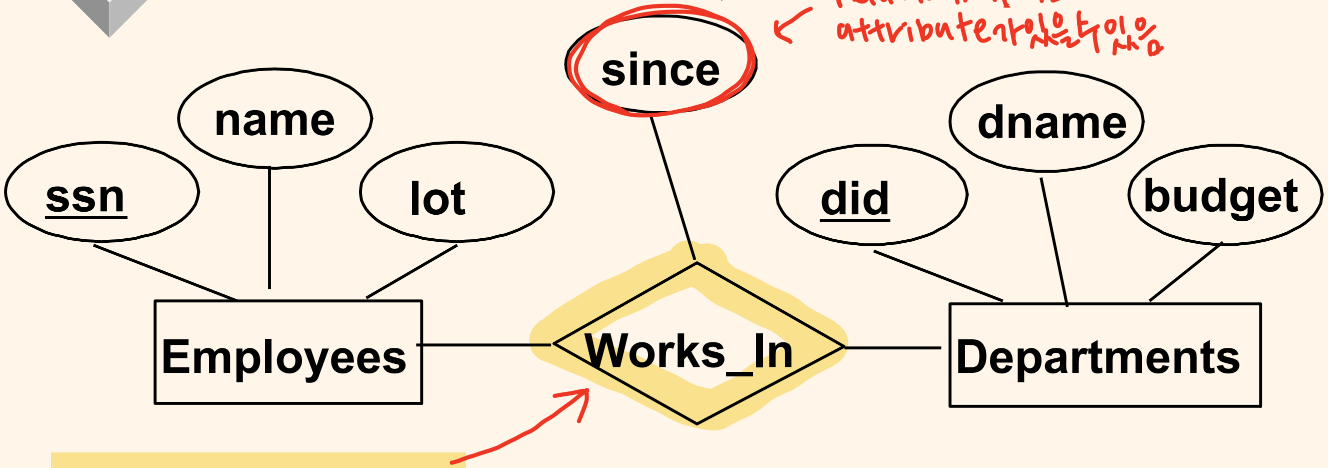
↳ data type

ER Model Basics (Entity Cont'd)

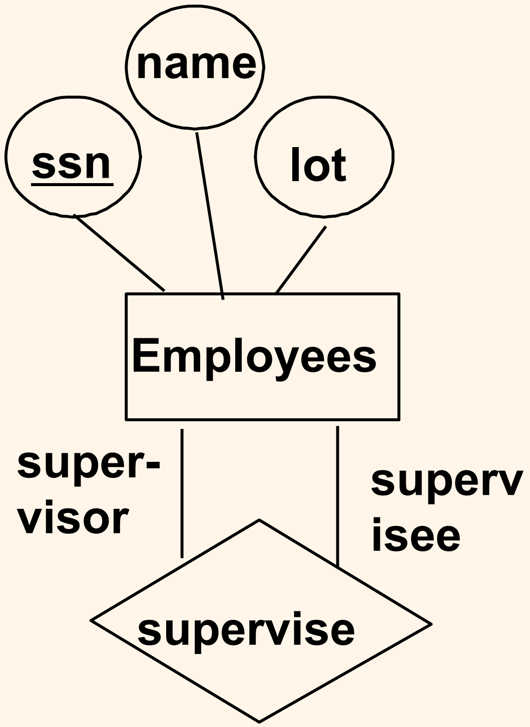


- *Attribute*: Property of an entity.
- *Type*:
  - *Atomic vs Composite*
  - *Single-valued vs Multi-valued*
  - 꼭 *Stored vs Derived* → 저장안할수있음! 유도가능
  - Relational model allows only *atomic* and *single-valued.*

ER Model Basics (Relationship)



relationship에도 attribute가 있을수있음



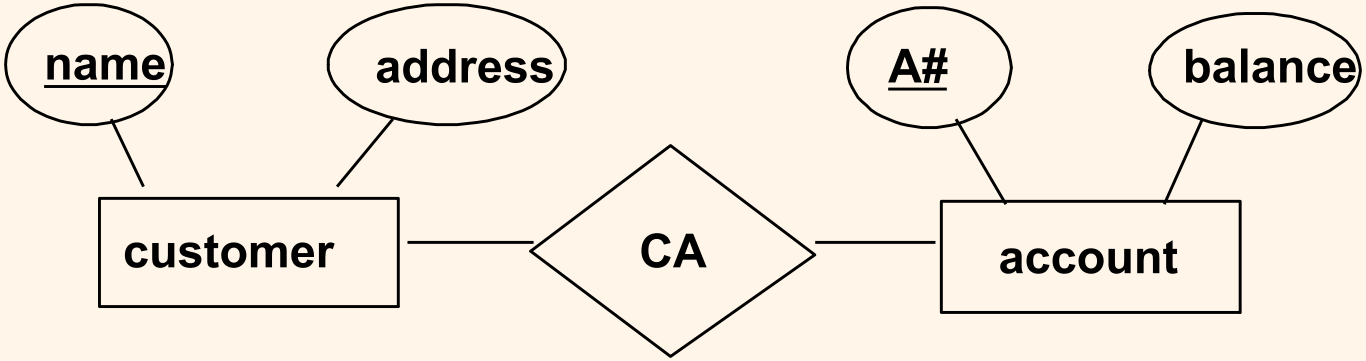
- *Relationship*: Association among two or more entities. E.g., Tom works in Pharmacy department.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En;

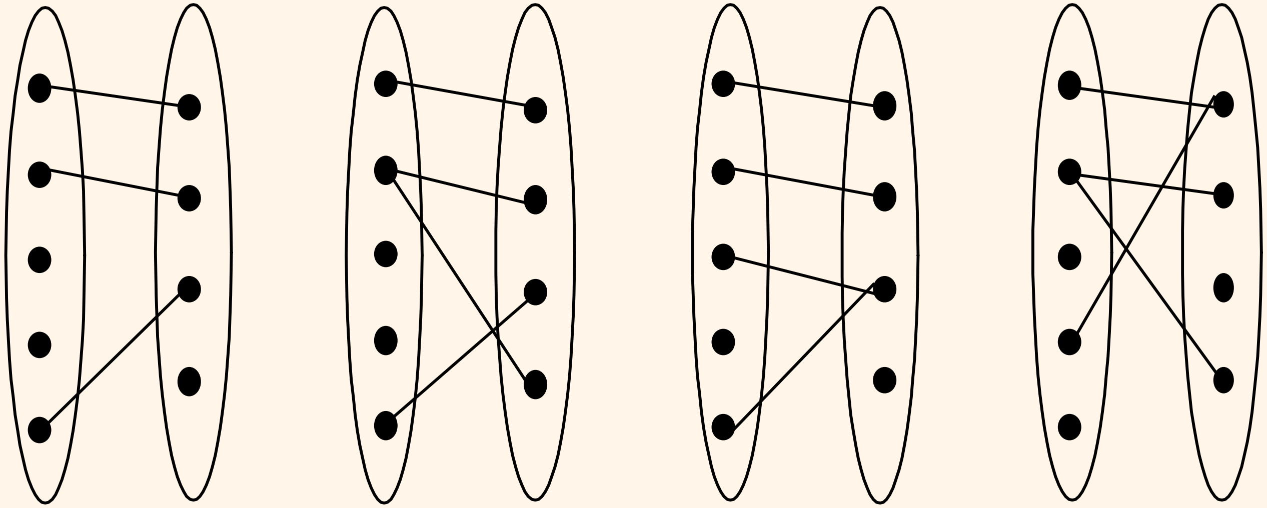
    n차

    - Same entity set could participate in different relationship sets, or in different "roles" in same set.

Mapping Constraints (Cardinality Ratio)

: binary





1-to-1

[✓ 1-to Many]

Many-to-1

Many-to-Many

Participation Constraints

* relationships에서 관심있는것
① mapping ratio (constraint)
:몇대 몇으로 묶이는지?
② participation constraint < total / partial
: 나도 참여해야돼?

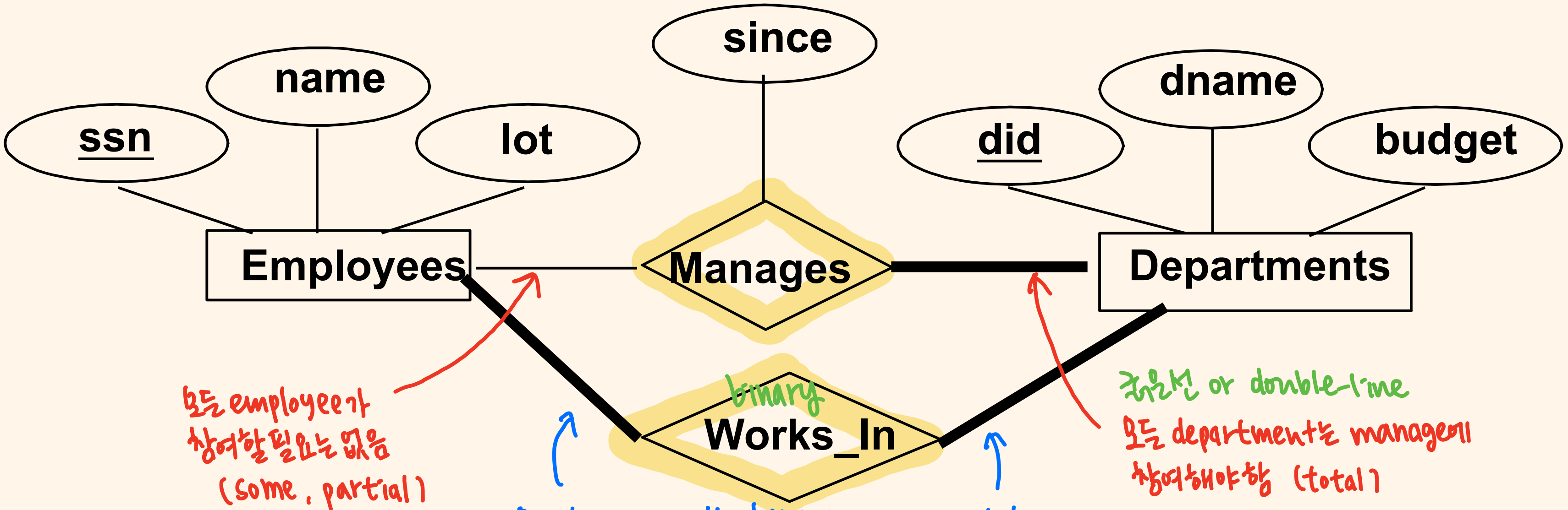
binary relationship – ex. X와 Y의 관계

X
Y
x1
x2
y1
y2

participate한 요소 : x1

ex) total은 치역 == 공역

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Total participation is represented in thick (double) line.
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



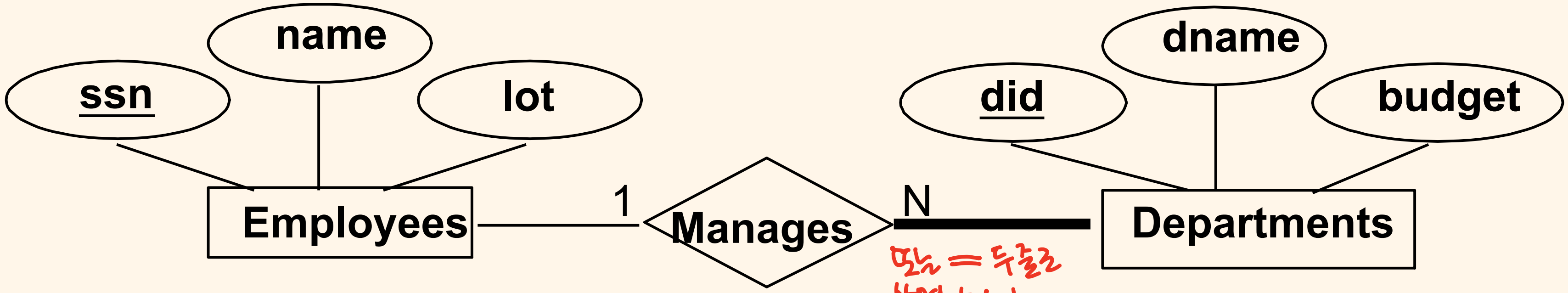
모든 employee가 참여할 필요는 없음 (some, partial) not total

굵은선 or double-line
모든 department는 manage에 참여해야함 (total)

모든 department는 최소1명의 employee가 있음
모든 employee는 최소1개의 department가 있음

⇒ 모든 department에는 manager가 있어야함

Exercise of Mapping Constraints and Participation Constraints

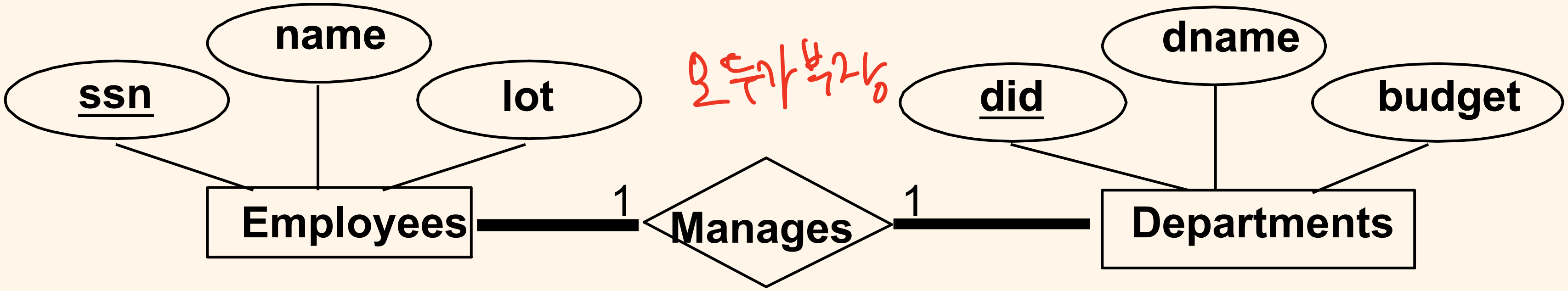


또는 = 두줄로 쓰면 total

- An employee can be the manager of multiple departments while some employees are not manager. Each department must have exactly one manager.

1:N이므로!

Exercise of Mapping Constraints and Participation Constraints (Cont'd)



오두가복가능

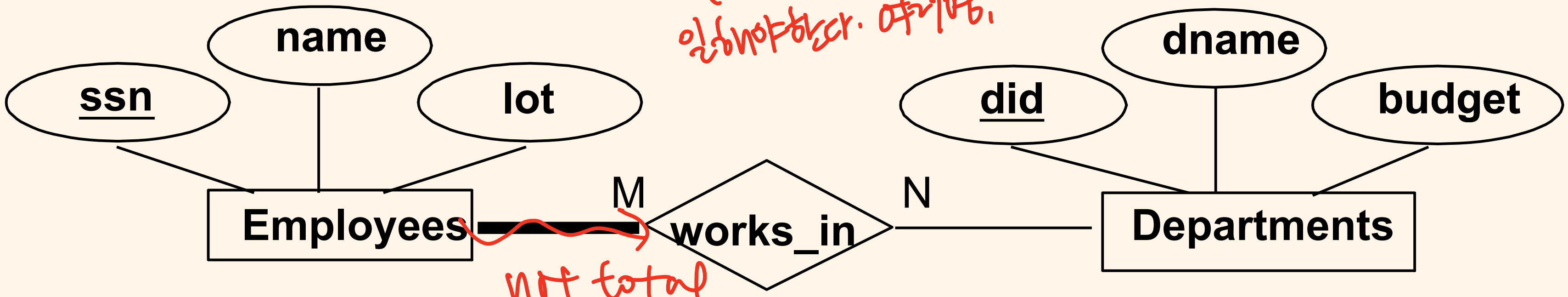
- Every employee must manage exactly one department, while each department must also have exactly one manager.

일대일대응

Exercise of Mapping Constraints and Participation Constraints (Cont'd)

0명인부서가능
모든직원은속한부서가있어야함

모든직원은
어느 1부서이상에
일해야한다. 여러명.



not total

- Every employee must work in at least one or more than one departments. Each department may have multiple employees while there may be some departments which do not have any employee to work in.

V

key를 이용해야만 구별할 수 있는 객체

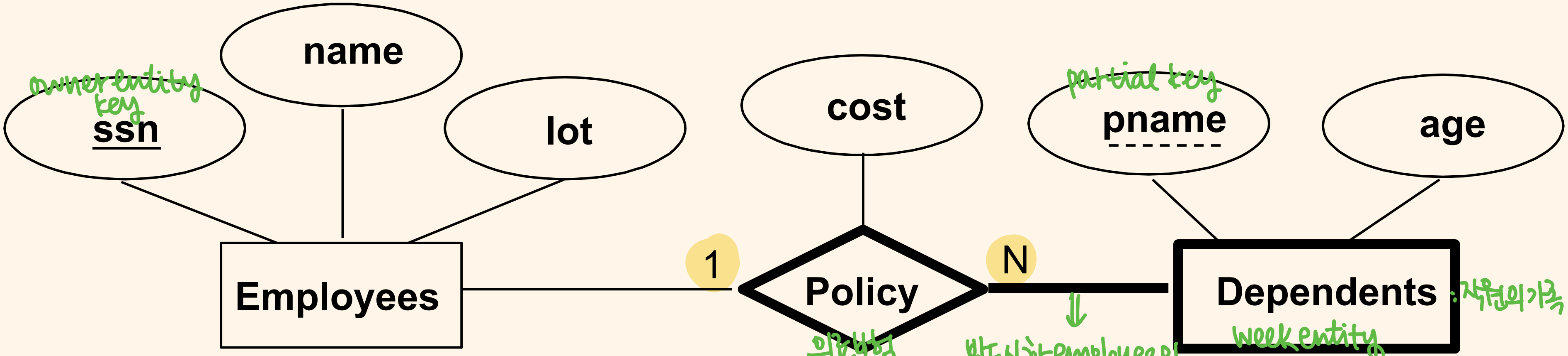
Weak Entities → object

↔ entity (strong x!!)

disitinguishable하지 않는 entity
무언가에 딸려있음 → owner key 파악 + partial key와 합성해서 식별,
key 없음. partial key만

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Key of weak entity : key of owner entity + partial key
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.
  - Removing owner entity results in removing its all weak entities : *existence dependency*.

A의 가족 B
C의 가족 B) 동명이인의 경우?
↳ 내 이름 B와 가족 이름을 같이 가짐



owner entity key

partial key

: 직원의 가족

week entity

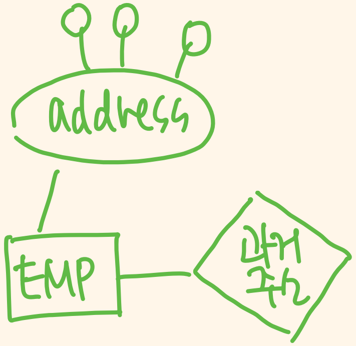
의료보험 ↙ 관계(O) 객체(X)

⇓ 반드시 한 employee의 가족으로 연결돼야 함
↳ total로 표시!!

모든걸다담을수없음

Conceptual Design Using the ER Model

- **Design choices:** 바라보는시각마다다름! subjective
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.



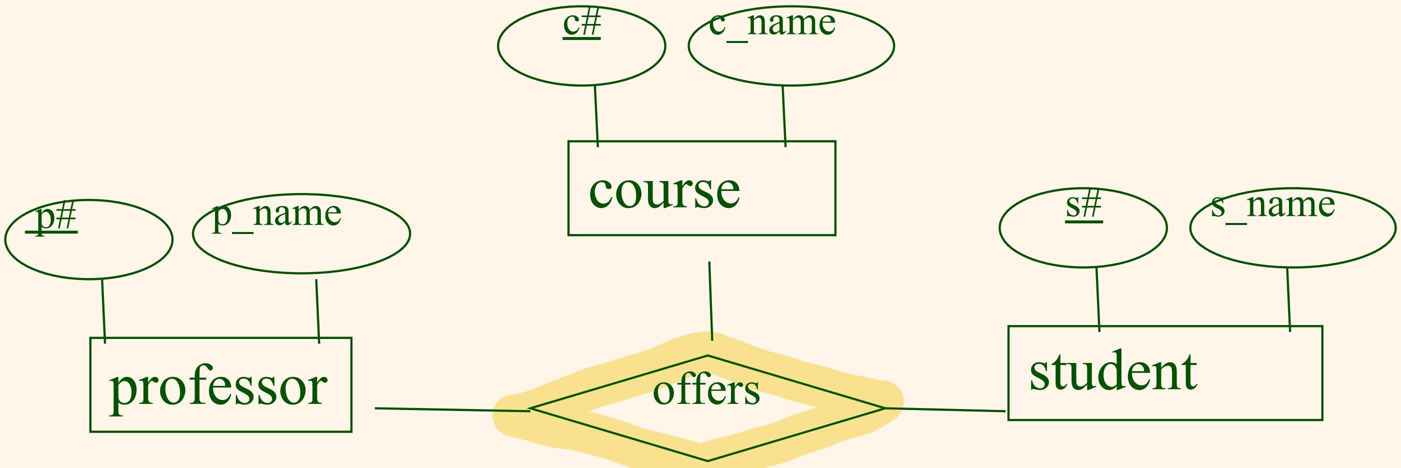
객체일수도. 관계일수도 → 디자이너마음!

Entity vs. Attribute

property
: entity를 묘사하는 data

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
    - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* can be modeled as an entity (since attribute values are atomic).

Ternary Relationships



3개의 entity 참여
→ n-ary relationship

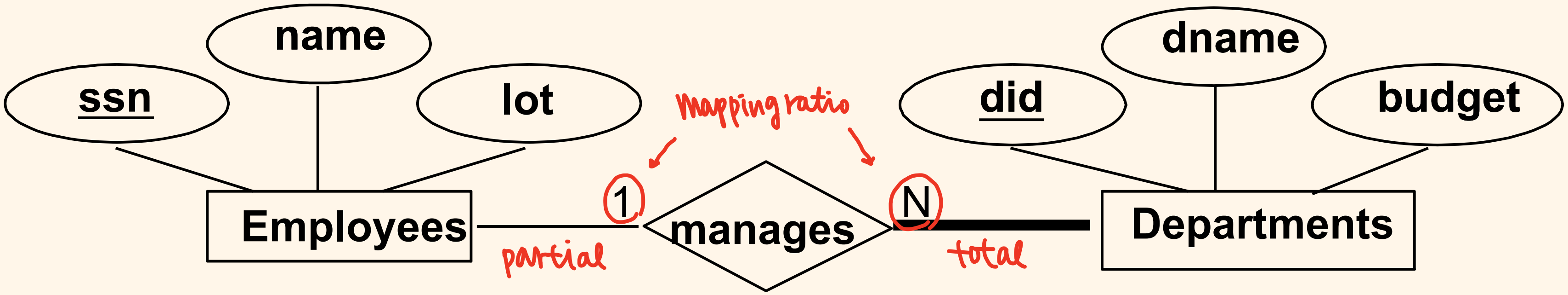
- An example of n-ary relationship
- Types of cardinality:
  - 1:1:1, 1:1:N, 1:M:N, M:N:P

M:N은 1:N과 N:1 모두를 만족
1:M:N은 1:1:N과 1:M:1 모두 만족

→ M:N:P가 동일한 pair(M:N)에 대해 다수의 P를 가지는 경우

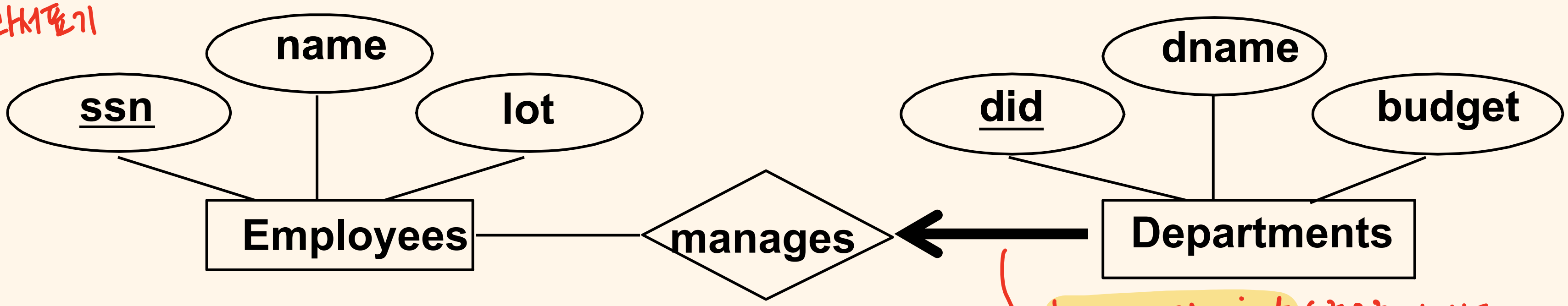
9/21

Different Notations



□ Ramakrishnan

✓ 교과서표기

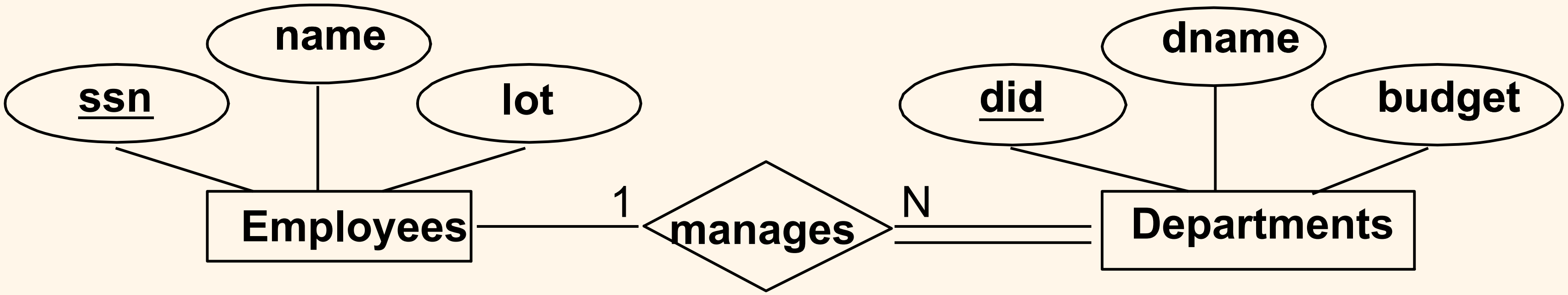


key constraint (굵은줄+화살표)

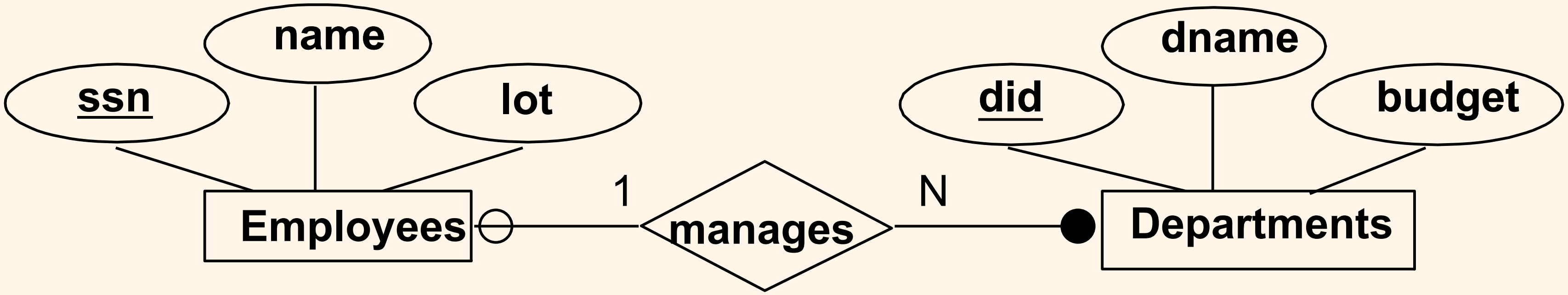
: department는 반드시 1번만 참여
(모든 department는 반드시 manager가 있고 한명만 가질 수 있다)
⇒ department 측면에서만 설명,
manager가 여러 department를 가질 수 있는지 여부는 알 수 없음

□ Ramakrishnan

Different Notations (Cont'd)



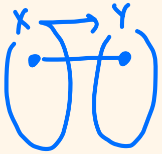
- Elmasri and Navathe



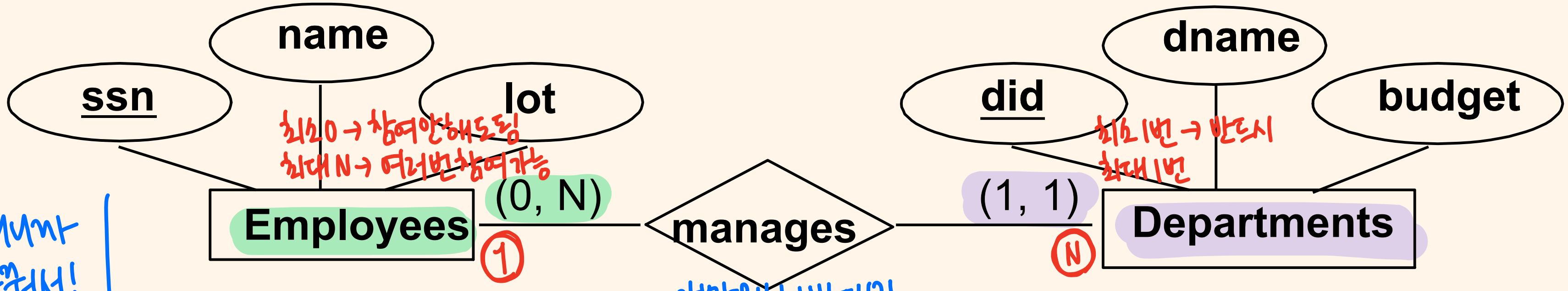
- Chen

범위를 정하고 싶은 경우
ex) 최소 1과목 ~ 7과목 수강신청

Different Notations (Cont'd)



서로가 있어야
relation 성립



최소 0 → 참여 안해도 됨
최대 N → 여러번 참여 가능

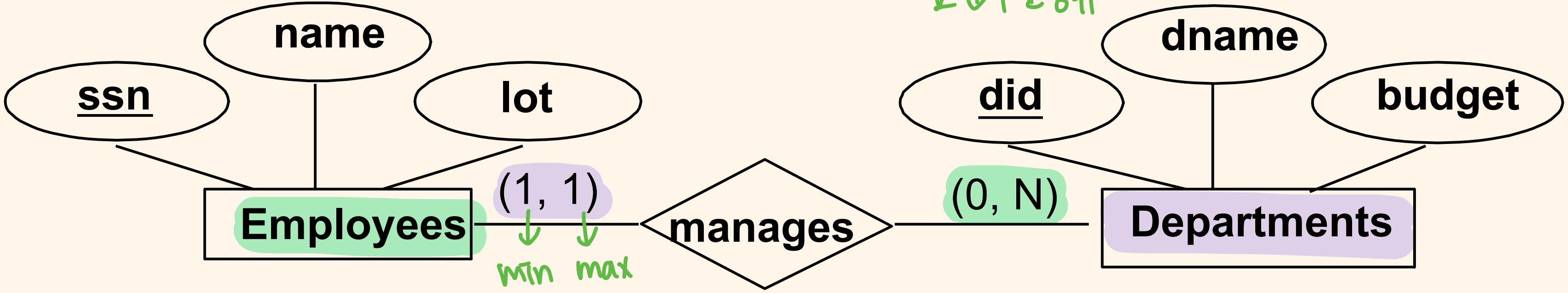
최소 1번 → 반드시
최대 1번

헷갈리니까
위치 바꿔서!
원래 쓰던대로
쓰자~

얘만 위치가 반대?!

□ Min-Max Notation with *Look Here* Notation

표현의 한 종류



□ Min-Max Notation with *Look Across* Notation

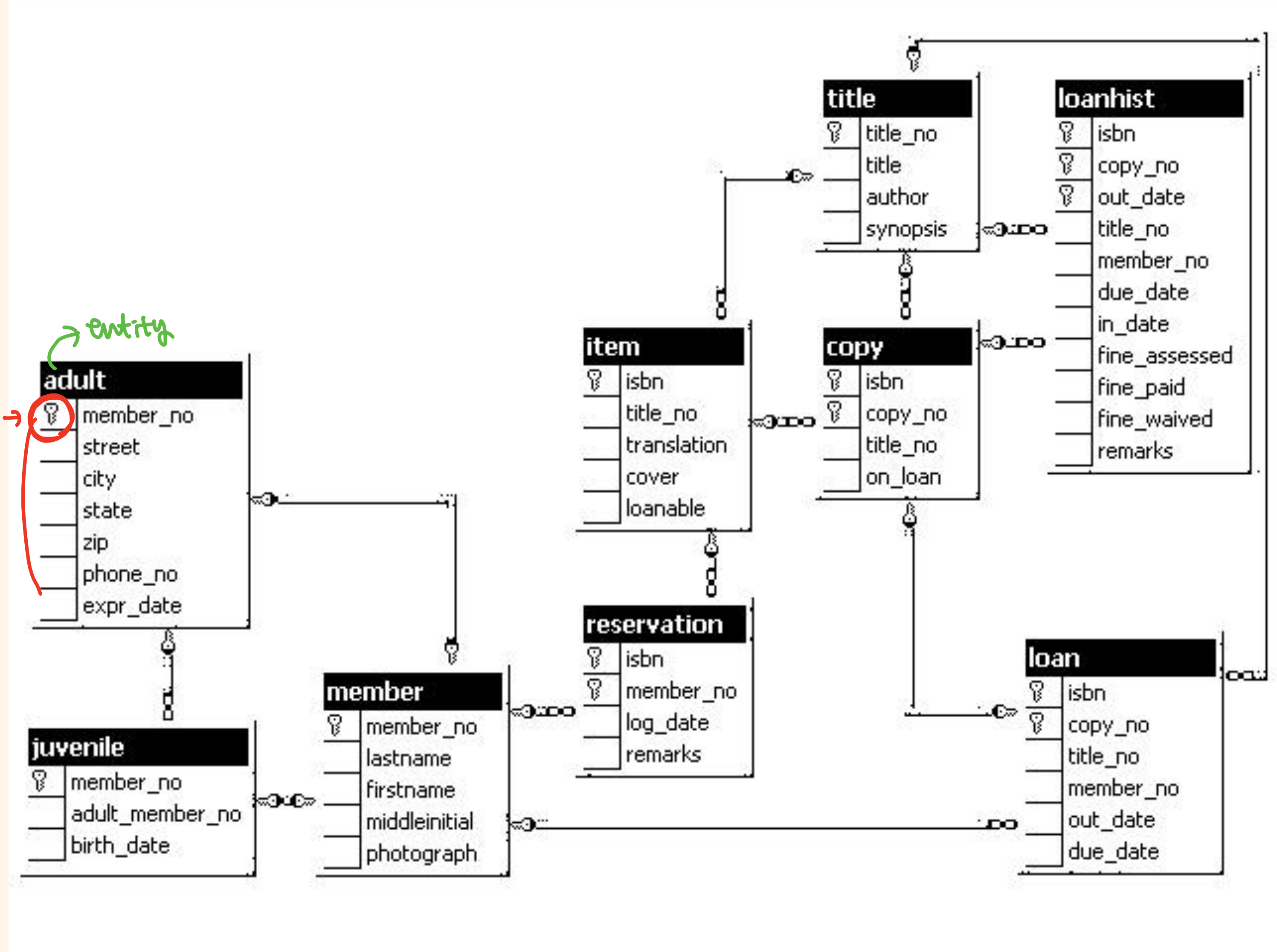
반대로

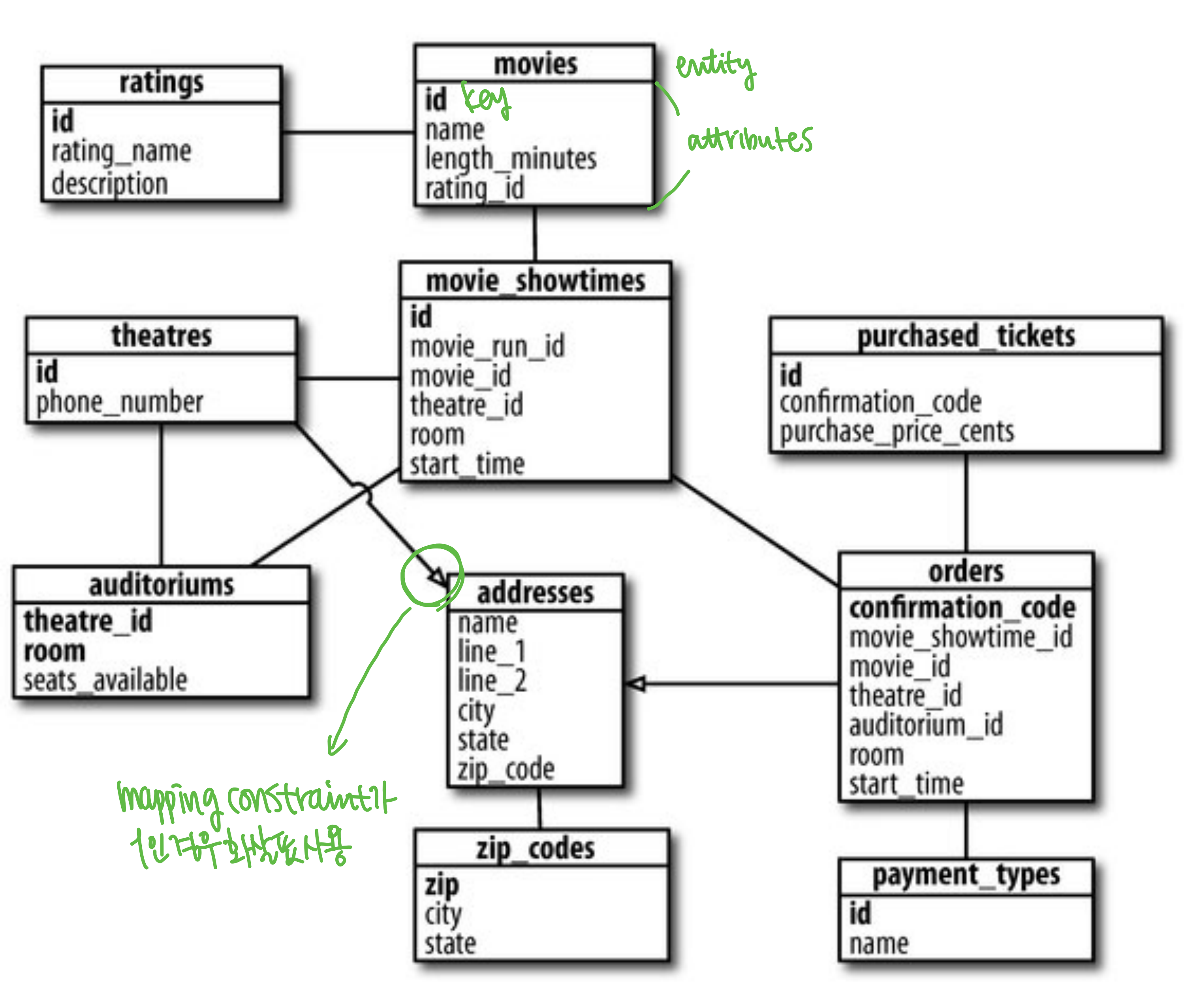
look-here인지 look-across인지
말해주지 않으면 그림만 보고는 알 수 없음!
통일해서 쓰면 됨

⇒ 결국은 모두 다이어그램은 같은
그림을 그린 것!

(object)
entity, relationship 모두 table이 될수있음 (모두 attribute를 가질수 있으니까)

⇒ table만 있는 아래 그림으로는 뭐가 entity고 relation인지 알수 없다?!





ratings
id
rating_name
description

movies
id
name
length_minutes
rating_id

movie_showtimes
id
movie_run_id
movie_id
theatre_id
room
start_time

theatres
id
phone_number

purchased_tickets
id
confirmation_code
purchase_price_cents

auditoriums
theatre_id
room
seats_available

addresses
name
line_1
line_2
city
state
zip_code

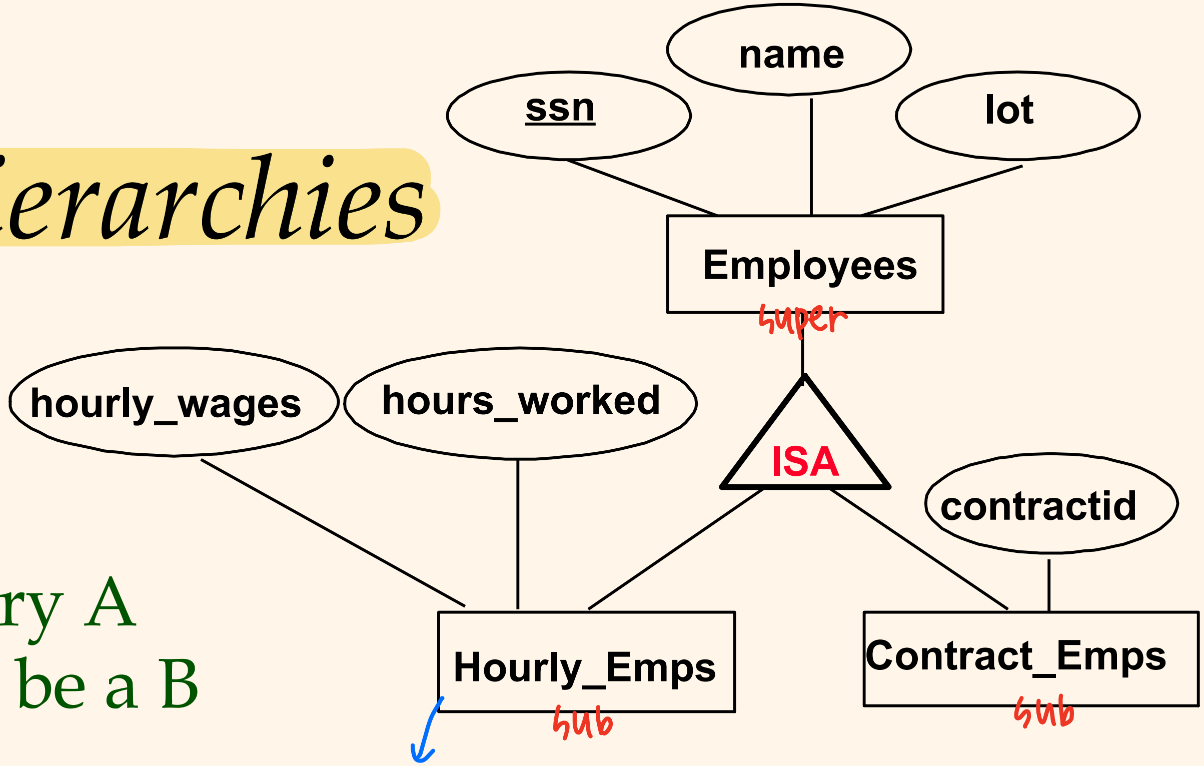
orders
confirmation_code
movie_showtime_id
movie_id
theatre_id
auditorium_id
room
start_time

zip_codes
zip
city
state

payment_types
id
name

mapping constraint가
1인경우화살표사용

ISA (`is a') Hierarchies



Hourly_Emps는 Employees의 것을 상속받음

B
(super)

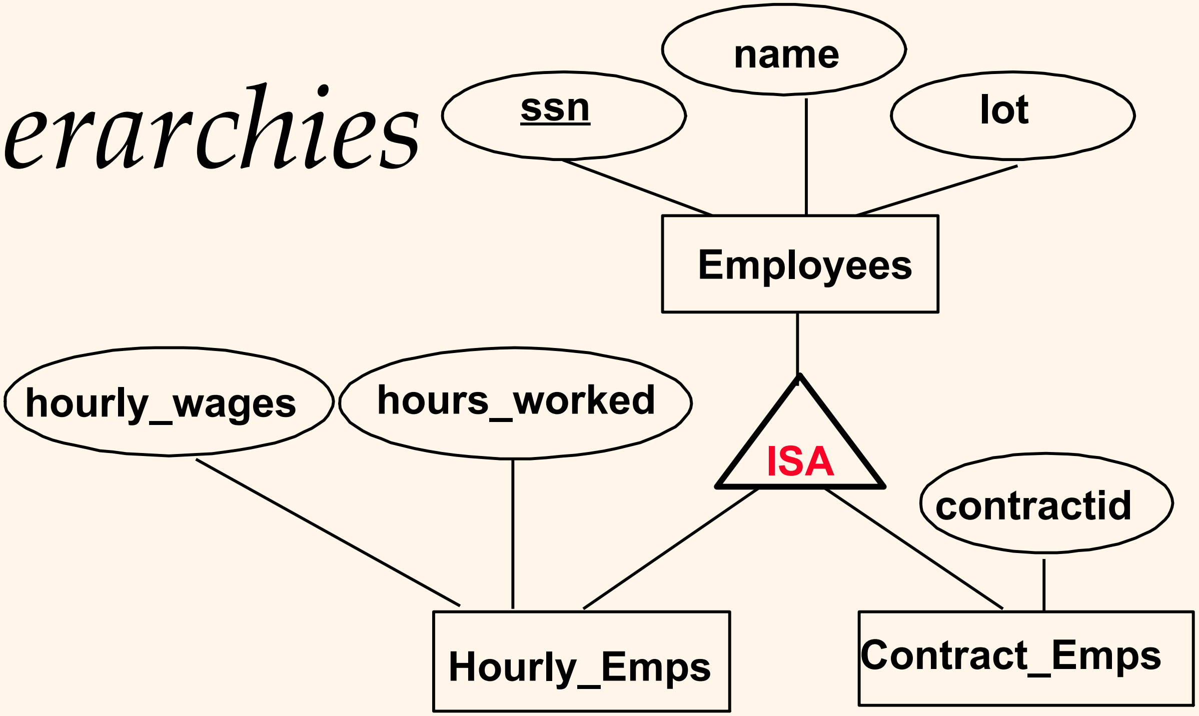
generalization ↑↓ specialization

A
(sub)

↳ A is a B (O)
B is a A (X)

- As in C++, or other PLs, attributes are inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity
  - A is a subclass of B. B is a superclass of A.
  - Also called generalization/specialization hierarchy
    - Generalization: bottom up
    - Specialization: top-down
  - Reasons for using ISA:
    - To add descriptive attributes specific to a subclass.
    - To identify entities that participate in a relationship.

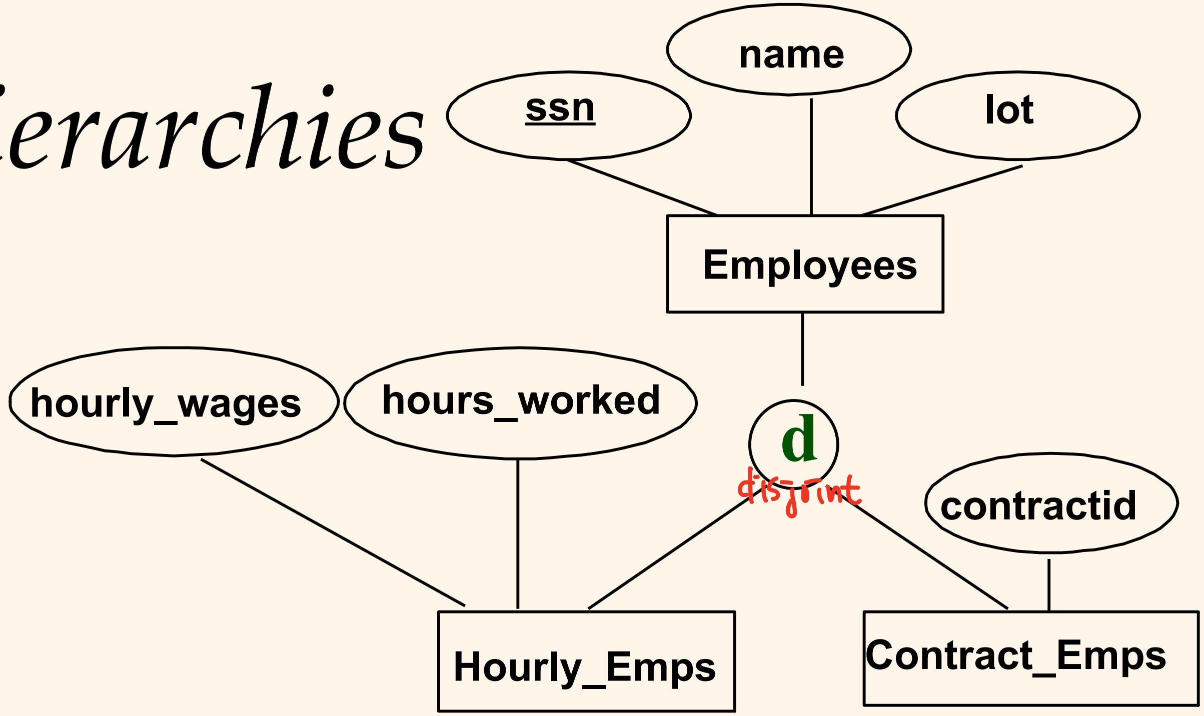
ISA (`is a') Hierarchies (Cont'd)



- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)

둘중하나는꼭

ISA (`is a') Hierarchies (Cont'd)



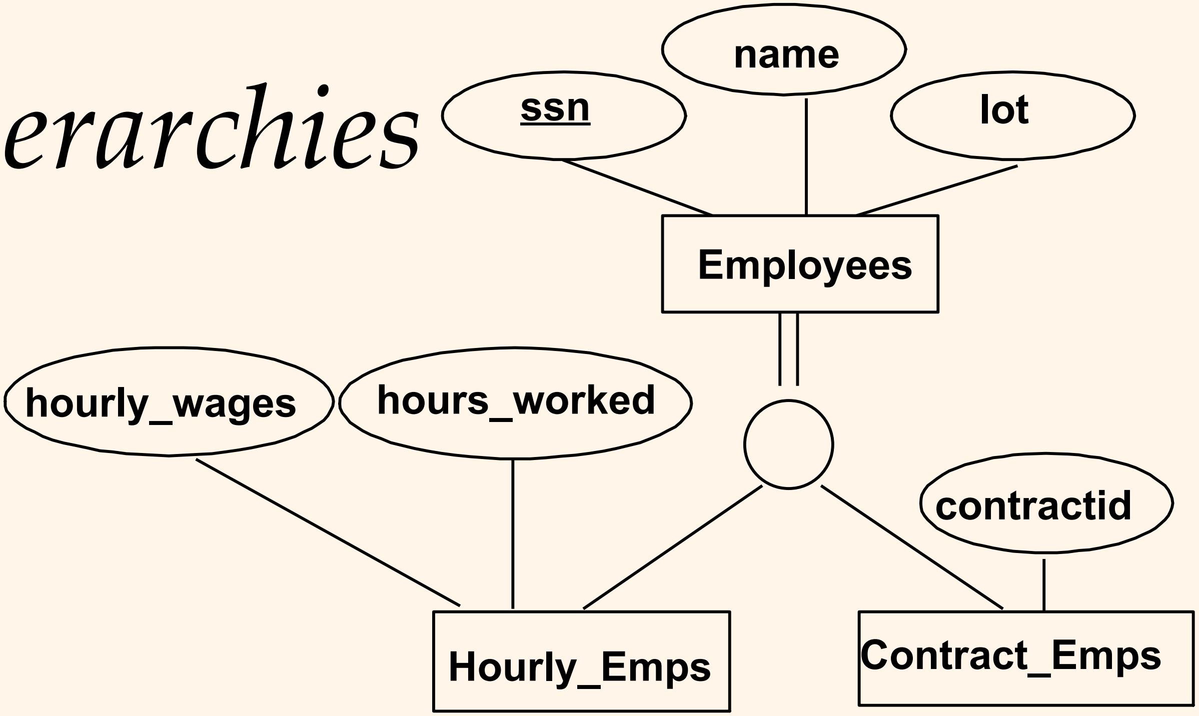
subclass간의 관계

partition

- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
  - If allowed, then **o**verlap, circled **o.**
  - If disalloed, then **d**isjoint, circled **d**.

동시불가능

ISA (`is a') Hierarchies (Cont'd)



계층간의

- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
  - If yes, then **thick (or double) line.** → Cover(O) : 둘중하나는 꼭
  - If no, then **think (or single) line**. → Cover(X)

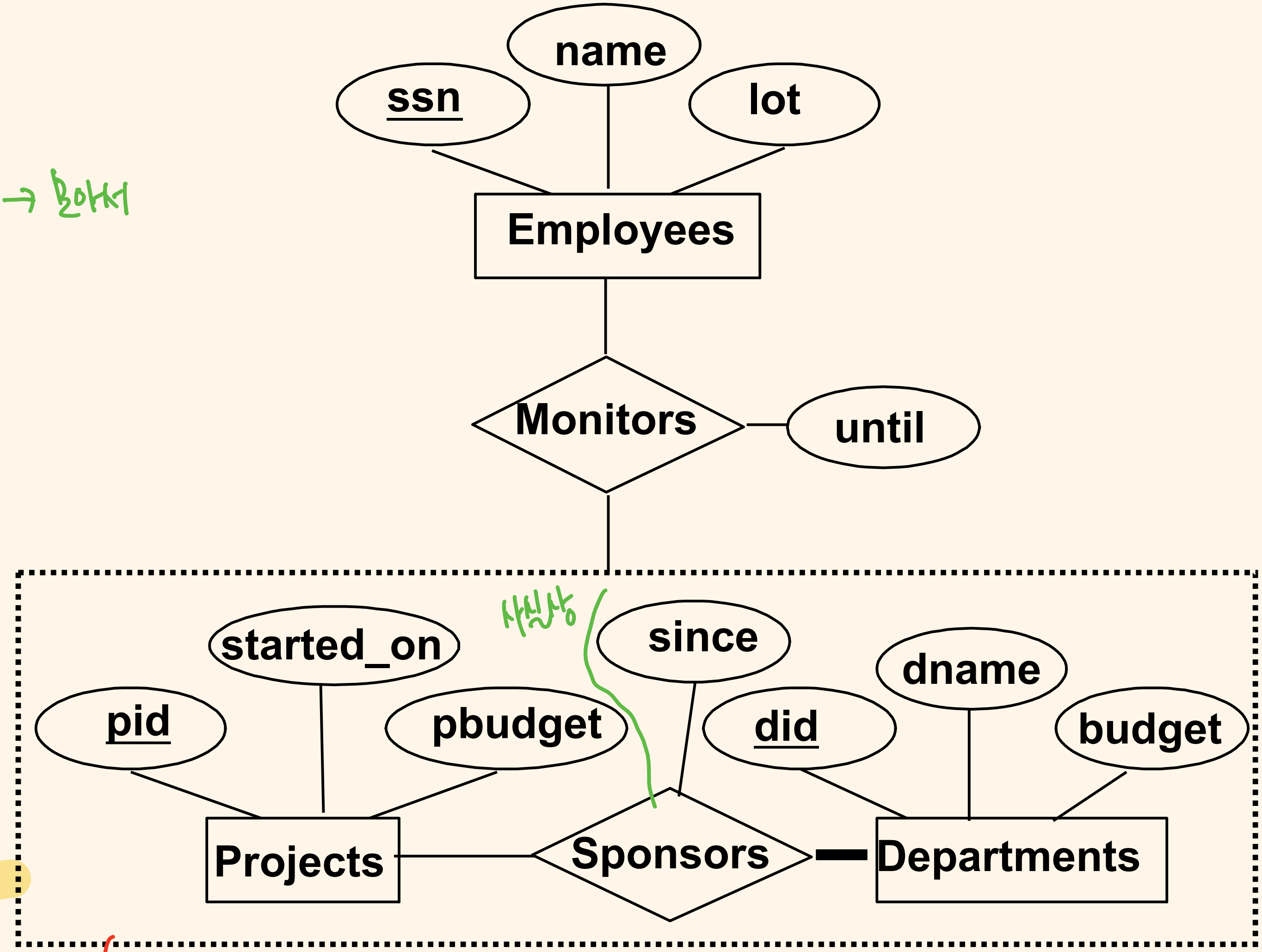
* class 나누기

hierachy
aggregation – ex) 자동차 – 타이어. 엔진, ...

Aggregation → 묶어서

- Used when we have to model a relationship involving (entitity sets and) a *relationship set*.
  - *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.

relation은 entity간의것인데
relationship라도 묶고싶다면?



aggregation된 부분
: 하나의 entity처럼 생각하기

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis,*
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model.

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *mapping cardinality constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further.