Chapter 3

Data Modeling Using the Entity-Relationship Model

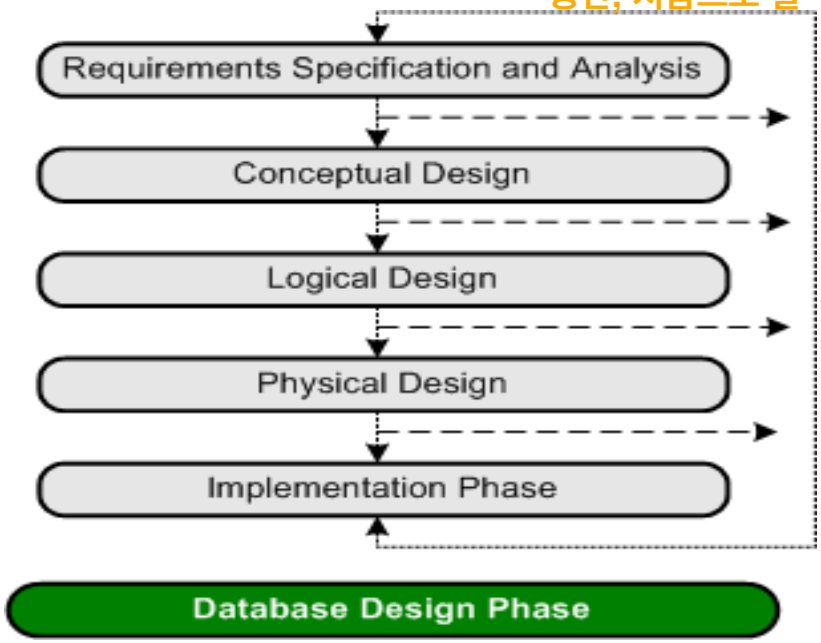
Table of Contents

Data Modeling Using the ER Model

개념 설계 - 구현하고자 하는 실세계를 모델링 함

- **Database design and implementation phases:**
  - Requirements specification and analysis
    - University administration
  - Conceptual design
    - ER diagram
  - Logical design
    - STUDENT , COURSE, …
  - Physical design
  - Implementation phase

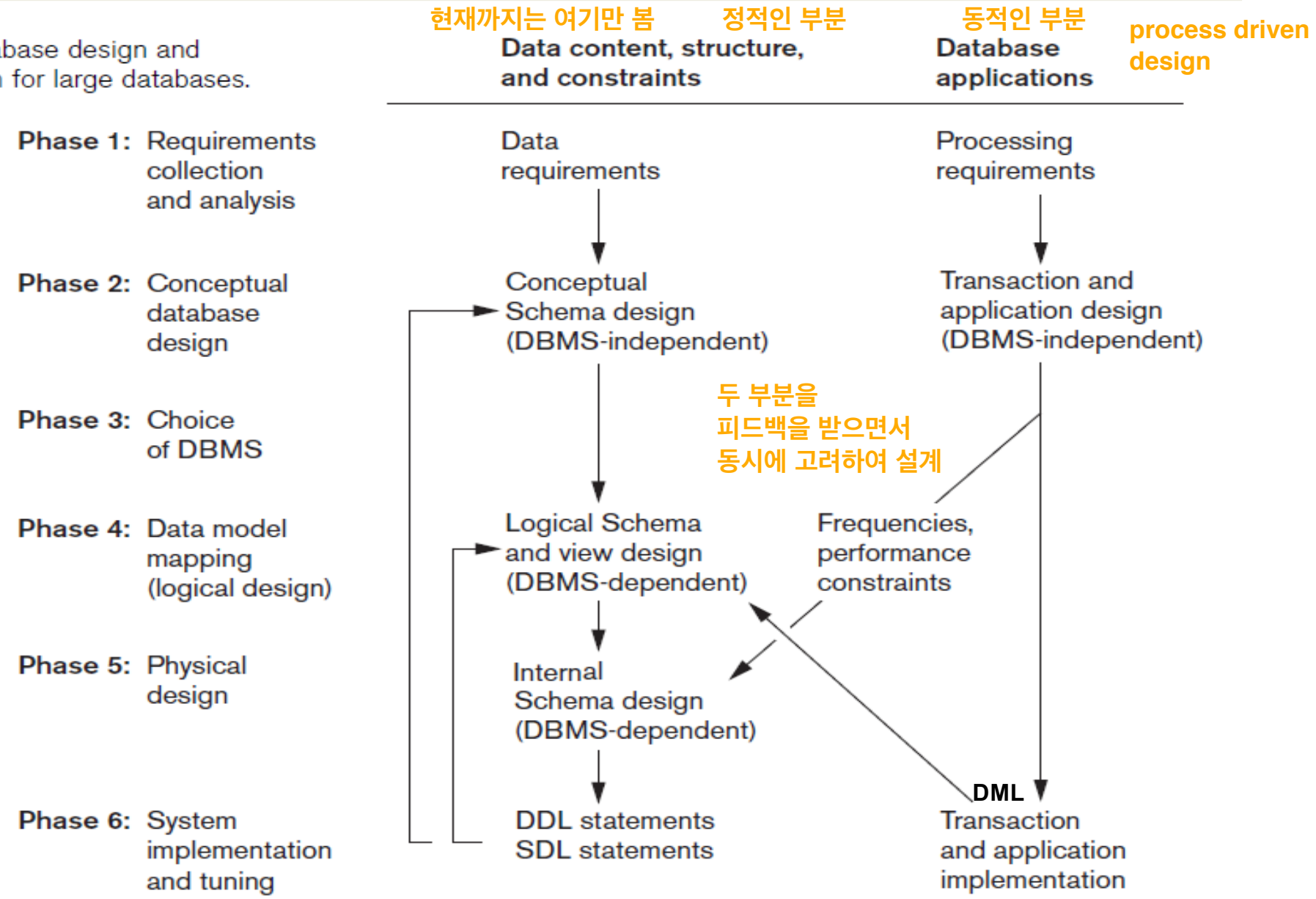
화살표 하나만 있는게 아님
중간, 처음으로 갈 수도 있음



- **In this chapter, we focus on the conceptual design by ER model**
  - ER model is a popular high-level conceptual data model
  - Entity is an object that exists and is distinguishable from other objects 개체(네모)
  - Attribute is a property of an entity 속성(동그라미)
  - Relationship is an association among several entities 관계(다이아 몬드)
  - ER diagram is a diagrammatic notation associated with the ER model

3.1 Using High-Level Conceptual Data Models for DB Design

Figure 10.1
Phases of database design and implementation for large databases.

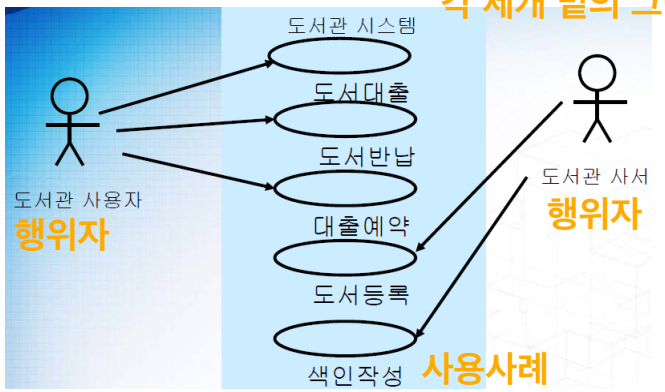


3.1 Using High-Level Conceptual Data Models for DB Design

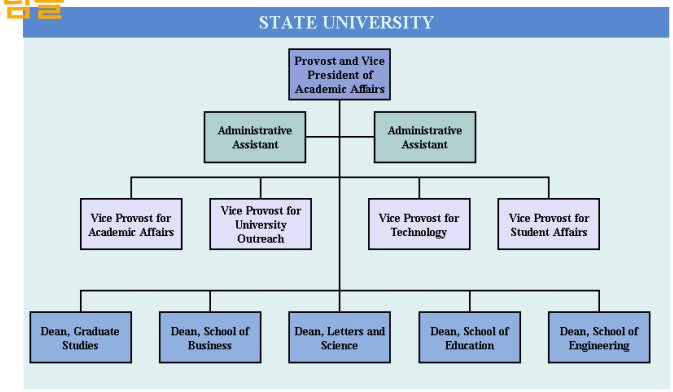
- **Phase 1: Requirements collection and analysis** 경영정보학과
  - **Activities:** DB designers and application programmers interview prospective DB users to understand and document their data and operations requirements. These users' requirements should be specified in as detailed and completed form as possible. 인터뷰, 회사 문서 통해서 분석 -> 결과를 문서화 등 다이아드램으로 만듬
  - **Requirements specification techniques:** Analysis document for user requirements, use case diagram, class diagram, data flow diagram, glossary of terms, etc UML

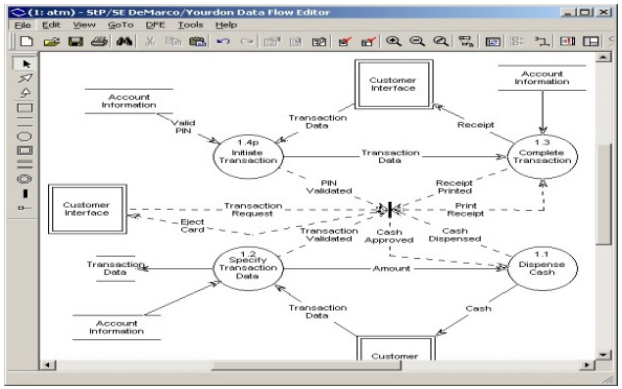
도서관 디비

각 세개 밑의 그림들



사용사례





- **Phase 2: Conceptual database design**
  - **Conceptual schema design:** Examine the data requirements resulting from Phase 1 and produce a conceptual DB schema such as an ER diagram
  - **Transaction and application design:** Examines the DB application analyzed in Phase 1 and produces high level specifications - identify input/output and functional behavior of DB applications or transactions
    - Using a notation that is independent of any specific DBMS

3.1 Using High-Level Conceptual Data Models for DB Design

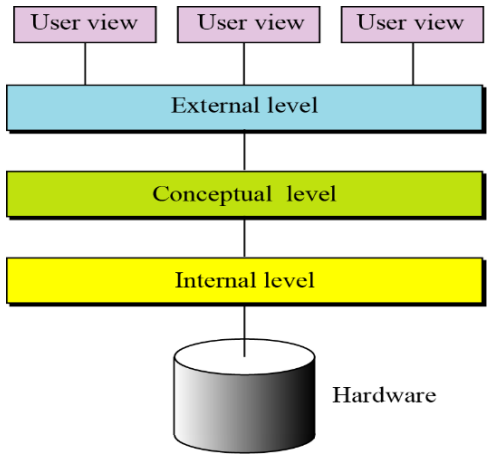
- **Phase 3: Choice of a DBMS**
  - Governed by a number of factors such as technical, economic, and political matters

- **Phase 4: Logical database design**
  - The conceptual schema from Phase 2 into the data model of the chosen DBMS 크게 두 가지 작업 필요
    ① Map the conceptual design into a set of relations by mapping rules
    ② Evaluate the relations for goodness by the normalization theory to achieve higher normal forms
  - Many automated CASE design tools can generate DDL from a conceptual schema design
  - In terms of the ANSI/SPARC three-level DBMS architecture, the result of this phase is a conceptual and an external schema for specific applications

3장 다음
er -> 테이블 매핑

케이스 툴을 이용해 er다이아드램을 만들고 곧바로 테이블 생성 해줄수도 있음

초록까지 이부분에 해당
노란색부터 밑은 피지컬에 해당



3.1 Using High-Level Conceptual Data Models for DB Design

- **Phase 5: Physical database design**
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified
  - Criteria used to guide choice of physical database design options:
    - Response time, Space utilization, Transaction throughput 이런것들 고려하여 설계
  - This phase corresponds to design the internal schema in the terminology of the three-level DBMS architecture
- **Phase 6: Database system implementation and Tuning**
  - This is typically the responsibility of the DBA and carried out in conjunction with the DB designers
    - Compose DDL, convert data from earlier systems
    - DDL/SDL of the selected DBMS are compiled to create the DB schemas and DB files
    - The DB can then be loaded with the data
  - DB programs with DML are implemented by application programmers
  - DB tuning continues as long as performance problems are discovered, and while the requirements keep changing

퍼포먼스 문제가 발생하면
꾸준히 튜닝하거나 프로그램
바꾸는 작업들이 필요하다

3.2 A Sample Database Application

- **Notation for ER diagram**
  - Rectangle represents entity which is an abstract object of some sort
  - Diamond represents relationship which is an association among several entities
  - Oval which is attached by a straight line to its entity or relationship represents an attribute
  - Underline indicates primary key attributes

1단계 - 사용자 요구 분석 결과물

- **This ER diagram shows the information about the company where some employees work on the projects and some employees manage the projects**

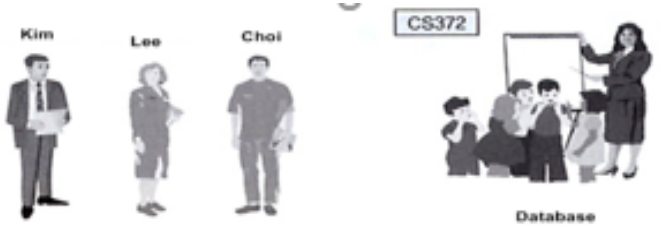
릴레이션십에 엔티티가 붙을 수 있음

봤을 때 뭐가 엔티티인지, 릴레이션십을 확인해야 함
빨간색 - 엔티티, 릴레이션 십 - 파란색

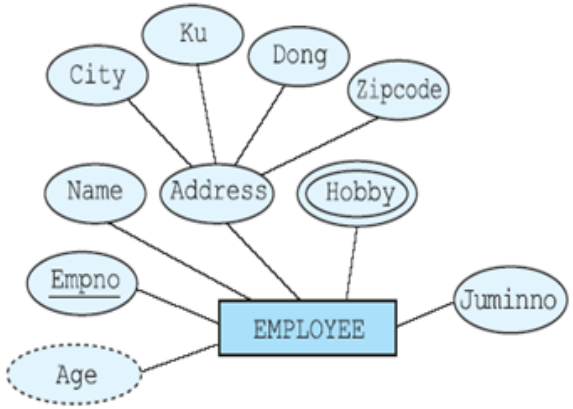
Entities, attributes and relationships in an E-R diagram

3.3.1 Entities and Attributes

- **Entity**
  - A thing in real world with independent existence
  - An object with a physical or a conceptual existence
  - General rule: If a concept has significant properties and/or describes classes of objects with an autonomous existence (A noun from the interview documents)



- **Attributes**
  - Particular properties that describe entity
  - General rule: If a concept has a simple structure and has no relevant properties associated with it (A noun or an adjective from the interview documents)
  - Types of attributes: 속성은 simple, composite 두 개로 나누어질 수 있음
    - Simple vs. composite attribute (Address)
    - Single-valued vs. multivalued attribute (Hobby)
    - Stored vs. derived attribute (Age)
      - Can be computed from other attributes
    - Complex attributes
      - Composite and multivalued attributes can be nested arbitrarily
      - {Address(City, Ku, Dong, Zipcode)}



3.3.2 Entity Types, Entity Sets, Keys, and Value Sets

- **Entity type**
  - Schema or intension for a set of entities that share the same structure
  - Each entity type in the DB is described by its name and attributes
  - An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name
- **Entity set**
  - Extension of the entity type
  - The collection of all entities of a particular entity type in the DB at any point in time
  - Entity set is usually referred to using the same name as the entity type

employee 이렇게 같이 사용함

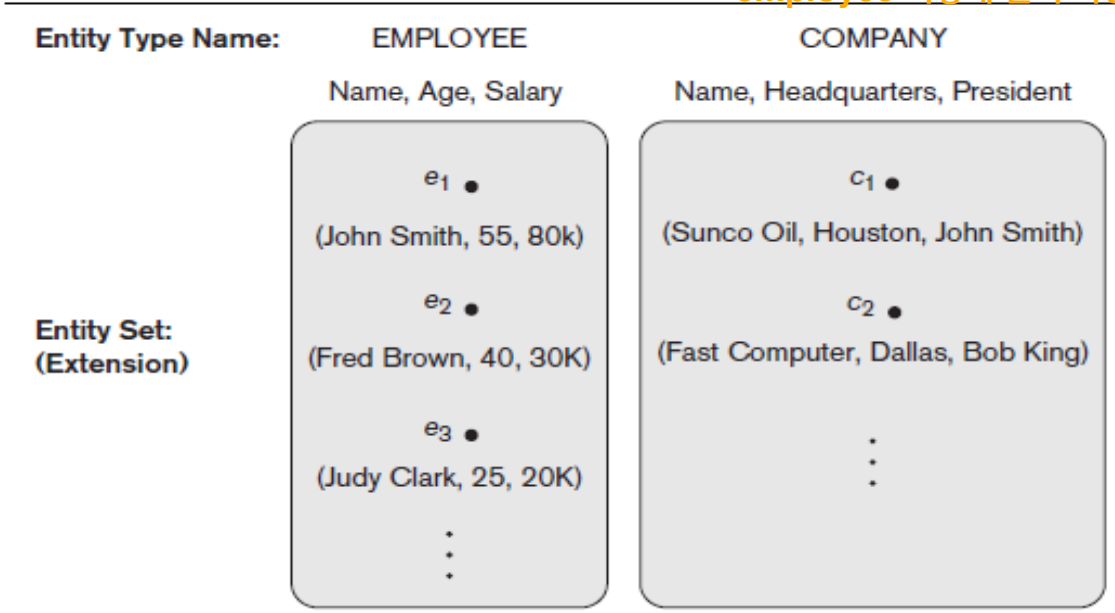


Figure 7.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

3.3.2 Entity Types, Entity Sets, Keys, and Value Sets

- **Key attribute**
  - Attributes whose values are distinct for each individual entity in entity type
  - Composite key: Registration attribute is an example of a composite key formed from two simple component attributes, State and Number
  - In ER diagrammatic notation, each key attribute has its name underlined inside the oval

unique 하게 구분함

- **Some entity types have more than one key attribute**
  - Each of the Registration and Vehicle_id attributes of the entity type CAR is a key in its own right
  - An entity type may also have no key, in which case it is called a weak entity type

퀴즈

- **Value sets (Domain of values) of attributes**
  - Specifies set of values that may be assigned to that attribute for each individual entity (Value sets are not displayed in ER diagrams)

(b)

CAR

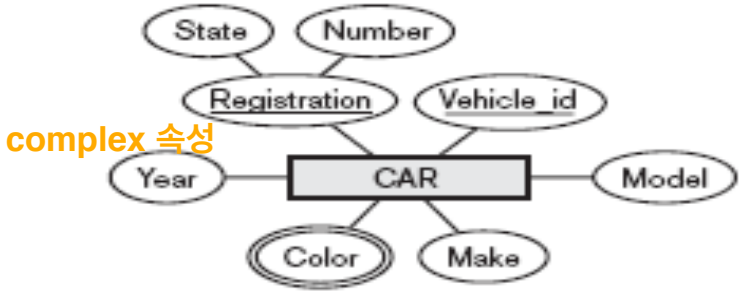
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

두개로 구분 가능

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

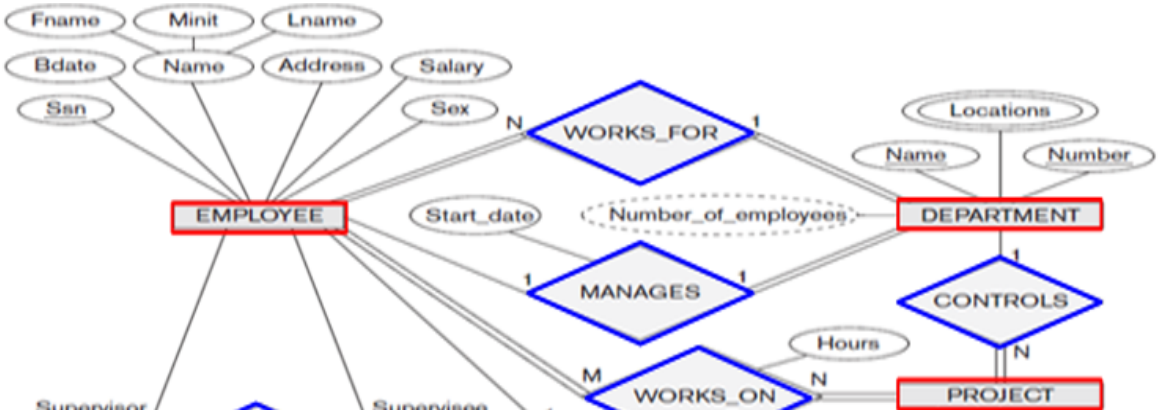
CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})



complex 속성

3.4.1 Relationship Types, Sets, and Instances

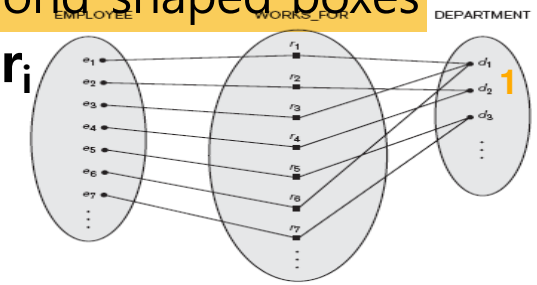
- An employee has a name, Ssn, address, salary, sex, and birth date. An employee is assigned to one department, but may work on several projects. ... Each department has a name and a number, the several locations, and a particular employee who manages the department.



- **Relationship**
  - When an attribute of one entity type refers to another entity type
  - Represent references as relationships not attributes
  - General rule: If the requirements contain a concept that provides a logical link between two (or more) entities (A verb from the interview documents)
- **Relationship type R among n entity types E_1, ... ,E_n**
  - Defines a set of associations among entities from those entity types
  - In ER diagrams, relationship types are displayed as diamond-shaped boxes
- **Relationship set R is a set of relationship instances r_i**
  - Each r_i associates n individual entities (e_1, ..., e_n) &
  - Each entity e_j in r_i is a member of entity set E_j

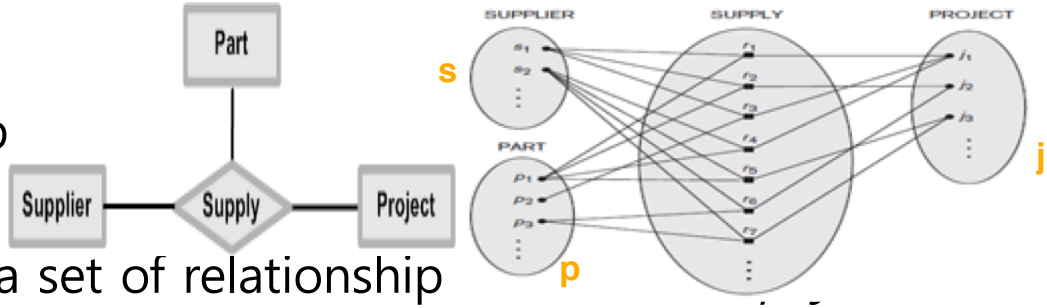
er다이아그램에서는 type만 고려

type, set
두 개로 구분



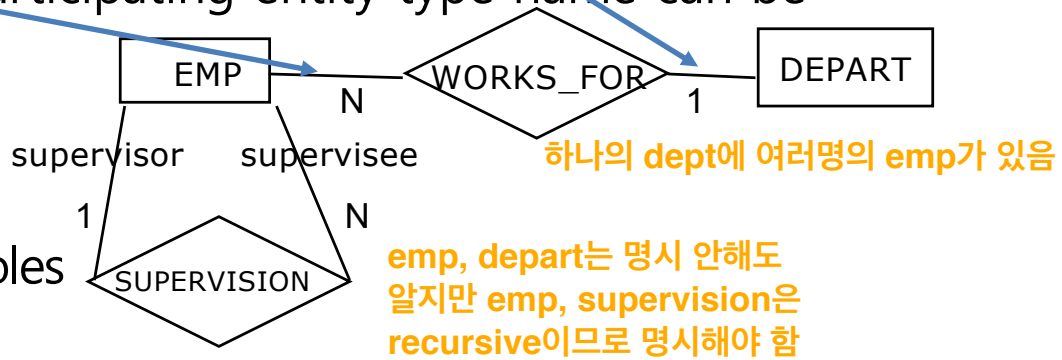
3.4.2 Relationship Degree, Role Names and Recursive Relationship

- **Degree of a relationship type**
  - Number of participating entity typ
  - Binary, ternary(SUPPLY), etc
  - The relationship set of SUPPLY, is a set of relationship *s* is a SUPPLIER, who is currently supplying a PART *p*, to a PROJECT *j*



- **Role names** 엔티티에 역할이 있음 emp : 일함 / dept : 고용
  - Each entity type that participates in a relationship type plays a particular role in the relationship
  - The role name signifies role that a participating entity plays in each relationship instance, and helps to explain what the relationship means
  - For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of worker and DEPARTMENT plays the role of employer
  - Role names are not necessary in relationship types where all the participating entity types are distinct, since each participating entity type name can be used as the role name

- **Recursive relationships** 이거는 명확하지 않으면 명시할 필요가 있음
  - Same entity type participates more than once in a relationship type in different roles
  - Must specify a role name



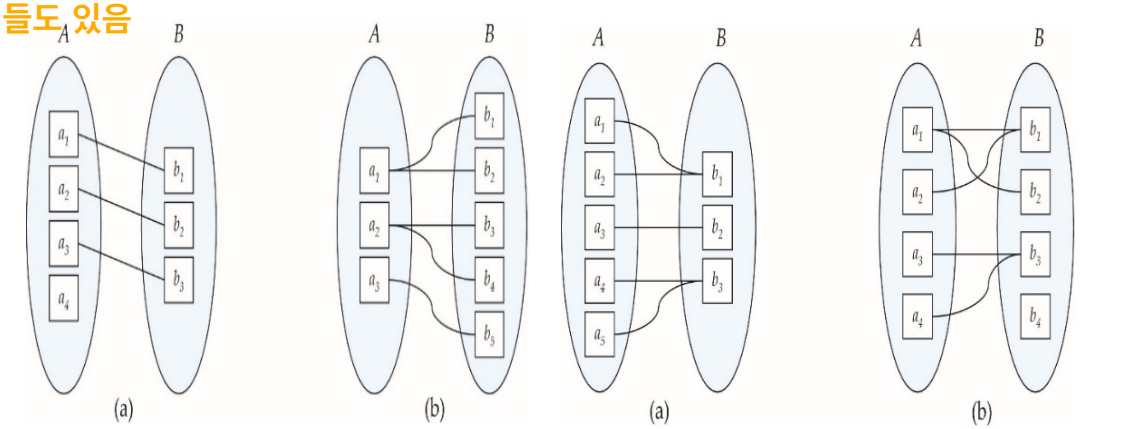
3.4.3 Constraints on Binary Relationship Types

- **Relationship types usually have <u>certain constraints</u> that limit the possible combinations of entities that may participate in the corresponding relationship set**
  - Cardinality ratio **1:1, 1:N, M:N**
  - participation **total, partial**

부분인지 전체인지 구분

- **Cardinality ratio for a binary relationship**
  - Specifies maximum number of relationship instances that entity can participate in.
  - The possible cardinality ratios for binary relationship types are 1:1(MANAGES), 1:N(WORKS_FOR, CONTROLS), M:N(WORKS_ON)

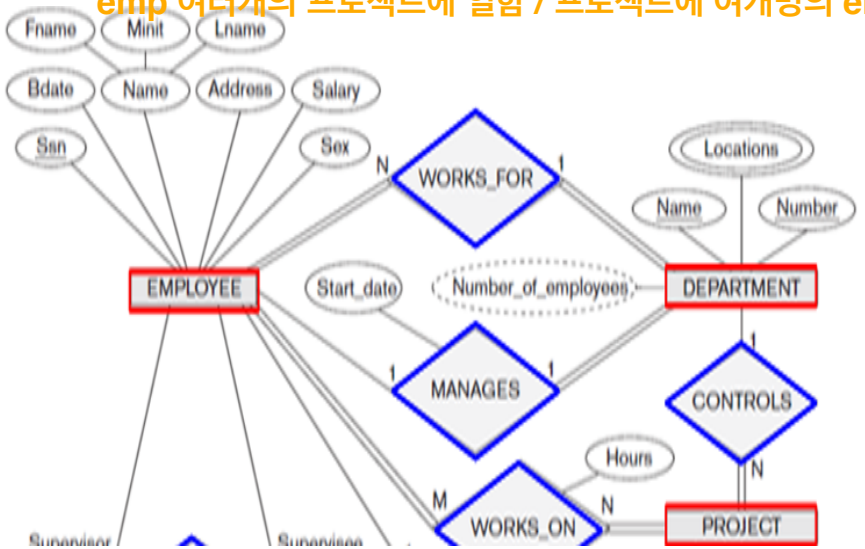
비어있을수도
- emp-man
없는애들도 있음



One to one　One to many　Many to one　Many to many

부부관계, 지도관계 등

emp 여러개의 프로젝트에 일함 / 프로젝트에 여개명의 emp 속함



- Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

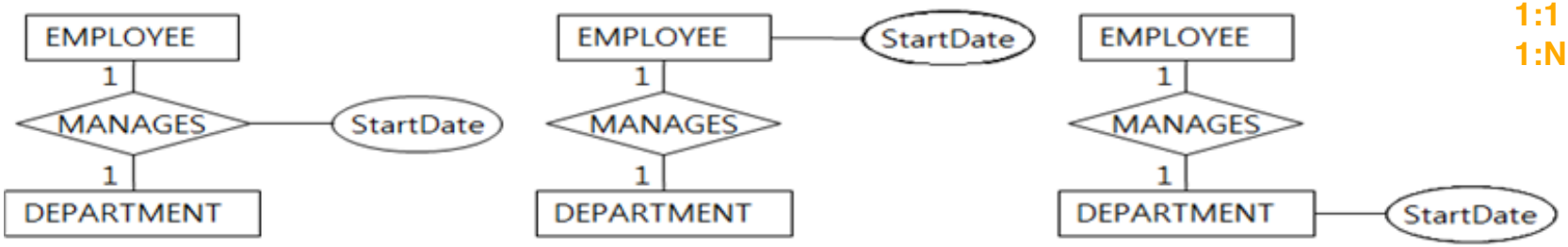
3.4.3 Constraints on Binary Relationship Types

- **Participation constraint and existence dependencies**
  - Specifies whether existence of entity depends on its being related to another entity via the relationship type
  - Types: total participation (double line) and partial participation (single line)
  - Total participation is also called existence dependency

- **Ex: MANAGES relationship type**
  - We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial
  - If a company policy states that every DEPARTMENT must have a manager, then the participation of DEPARTMENT in the MANAGES relationship type is total

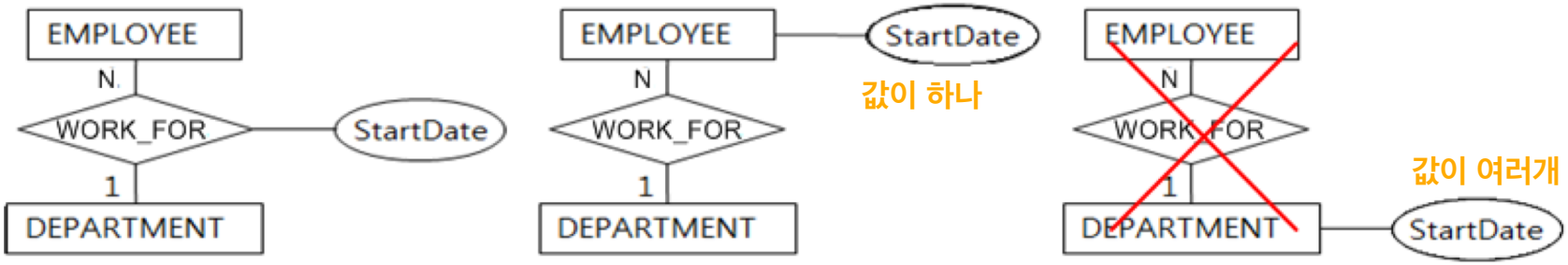
3.4.4 Attributes of Relationship Types

- **Relation types can also have attributes, similar to those of entity types**
- **One example is to include the date on which a manager started managing a department via an attribute StartDate for the MANAGES relationship type**
- **Attributes of 1:1 or 1:N relationship types can be migrated to one entity type**
  - **For 1:1 relationship type:** Relationship attribute can be migrated to either entity type



1:1 아무대나 가능
1:N N:M 아무대나 ㄴ

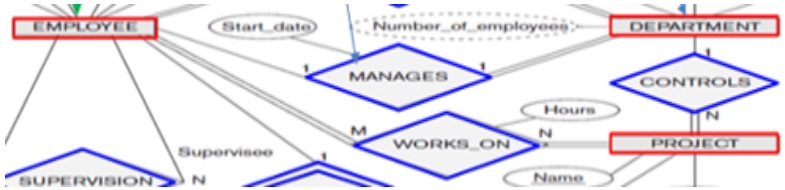
  - **For 1:N relationship type:** Relationship attribute can be migrated only to entity type on N-side of relationship



값이 하나

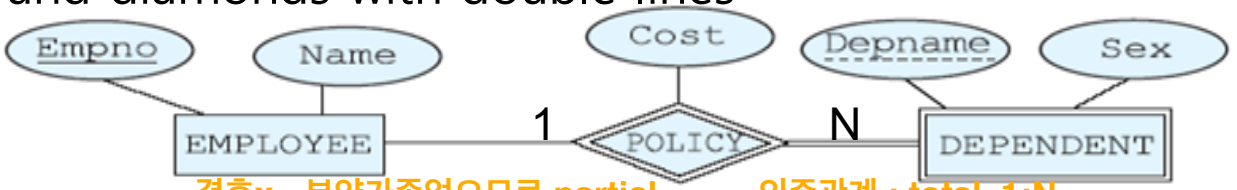
값이 여러개

- **Attributes of M:N relationship types must be specified as relationship attributes**
  - Hours attribute of the M:N relationship WORKS_ON cannot be migrated to either entity type



3.5 Weak Entity Types

- **Weak entity type** (⇔ **Strong entity type**) 구분되어 부를때는 스트롱이라 부름
  - An entity whose existence depends on the presence of another entity
  - An entity type that does not have a primary key 프라이머리 키가 없는 엔티티 + 다른 엔티티에 의존
- **The existence of a weak entity type depends on the existence of a identifying entity type (**or **owner entity type)**
  - It must relate to the identifying entity type via a total (existence dependency) and one-to-many relationship type from the identifying to the weak entity type
  - Both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines



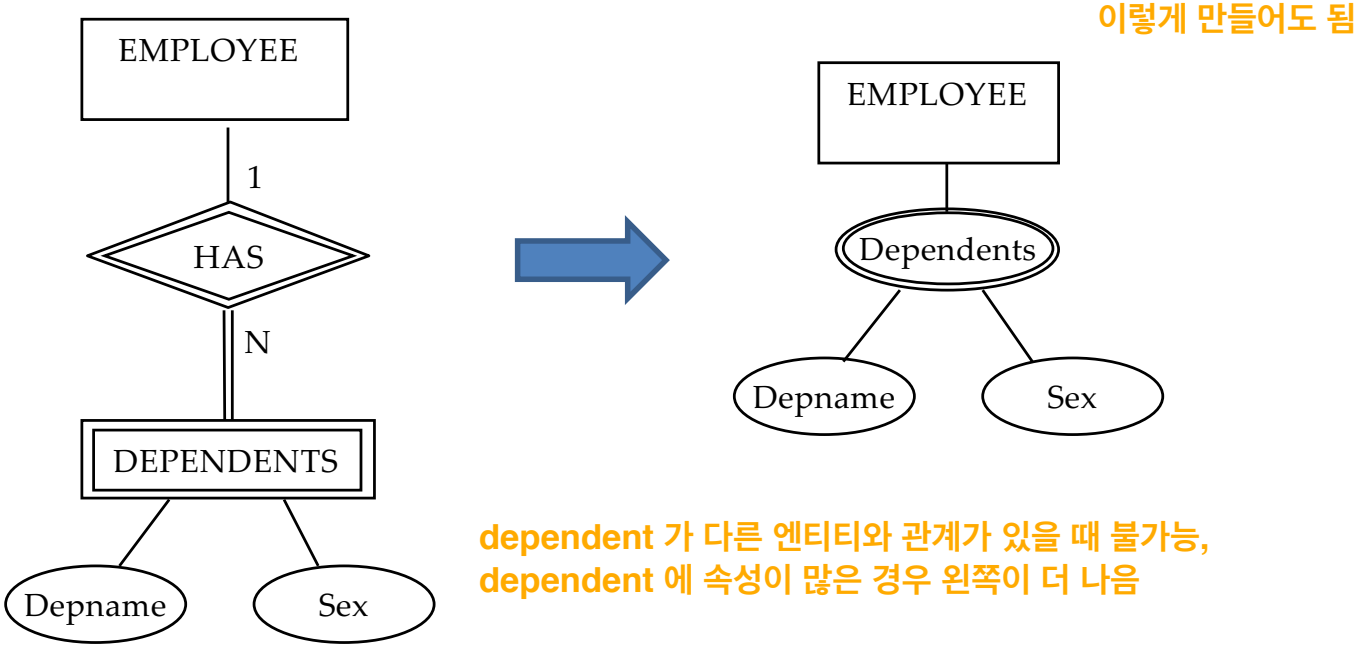
결혼x - 부양가족없으므로 partial 의존관계 : total, 1:N

- **The discriminator (or partial key) of a weak entity type is the set of attributes that can uniquely identify weak entities that are related to the same owner entity**
  - In our example, since no two dependents of the same employee have the same name, the attribute "Depname" of DEPENDENT is a discriminator
  - We underline the discriminator of a weak entity type with a dashed line
  - The primary key of a weak entity type is formed by the primary key of its identifying entity type plus the weak entity type's discriminator
    - Primary key for dependent: (Empno, Depname)

하나의 emp는 depname 구별 가능(심청이) 전체를 구분할 때는 empno와 dempname 합쳐서 관리

3.5 Weak Entity Types

- **Weak entity types can sometimes be represented as complex (multivalued & composite) attributes.**
  - We could specify a multivalued attribute Dependents for EMPLOYEE, which is a composite attribute with component attributes <Depname and Sex>



- **The choice of which representation to use is made by the DB designer**
  - One criterion that may be used is to choose the weak entity type representation if it has many attributes.
  - If the weak entity participates independently in relationship types other than its identifying relationship type, then it should not be modeled as a complex attribute

3.6 ER Design for the COMPANY Database

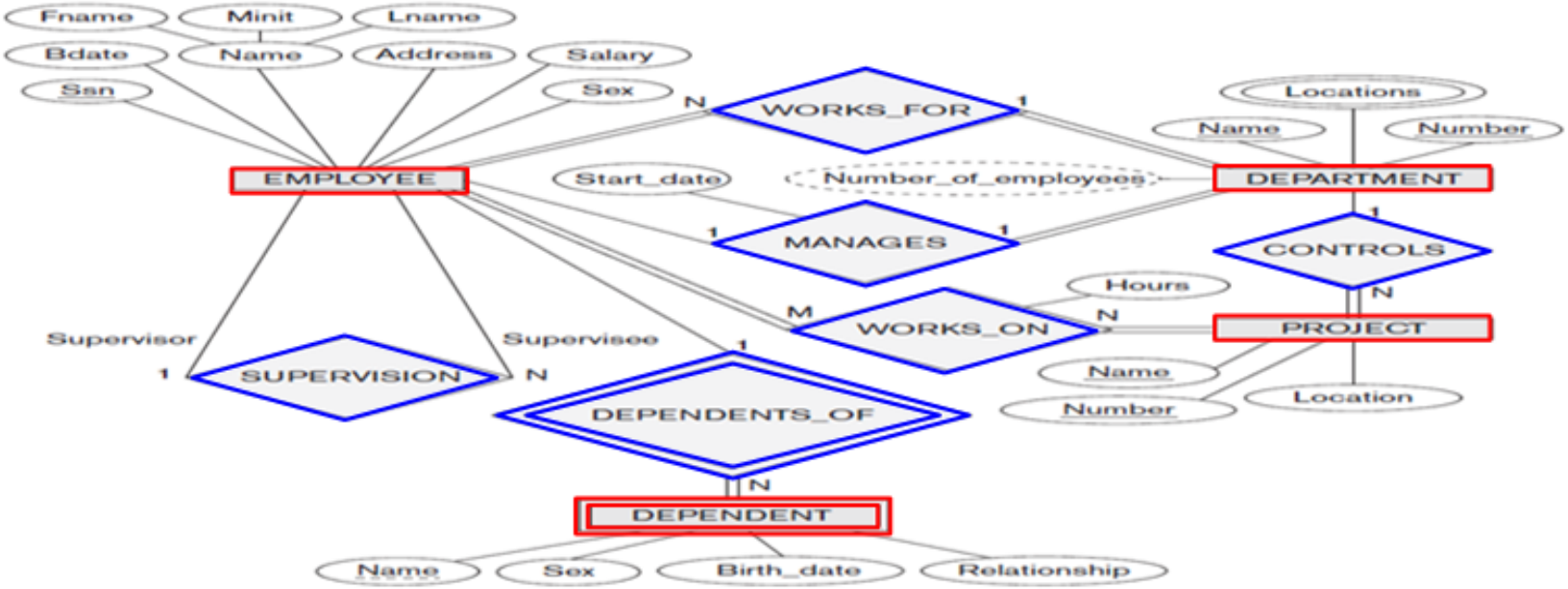
- **Database design process** top down 방식 (일반적임) , botton-up 방식도 존재함
  - Give a description by collecting an application's requirements
  - Distinguish the entity types related to the application (Noun)
    1. Distinguish a noun from a statement of work, a set of accounts, and the interview documents
    2. Remove ill-defined or broad scaled concepts 너무 큰 개념 등 삭제 (Company를 엔티티로 정했을 경우)
    3. Check whether there are omitted entity types
  - Distinguish the relationship types related to the application (Verb)
  - Decide the relationship types among 1:1, 1:N, and M:N
  - Draw an ER schema diagram for application
  - Distinguish the attributes needed for entity types and relation types (Noun or adjective)
  - Distinguish the primary key for entity types
  - Examine whether the ER schema diagram coincides with the application's requirements
  - Map the ER schema diagram to the corresponding data model used in a DBMS

3.6 ER Design for the COMPANY Database

실제 교재는 일반적인 방식과 조금 다름
두 단계를 거침

- **Data requirements:** The COMPANY database keeps track of a company's employees, departments, and projects
  - Each department has a name and a number, the several locations, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department.
  - An employee has a name, Ssn, address, salary, sex, and birth date. An employee is assigned to one department, but may work on several projects. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee
  - We keep track of the dependents of each employee for insurance purposes. We keep each dependent's name, sex, birth date, and relationship to the employee
  - A department controls a number of projects, each of which has a name and a number, and a location

1. mansge 를 처음에 속성으로
2. 관계로 변경

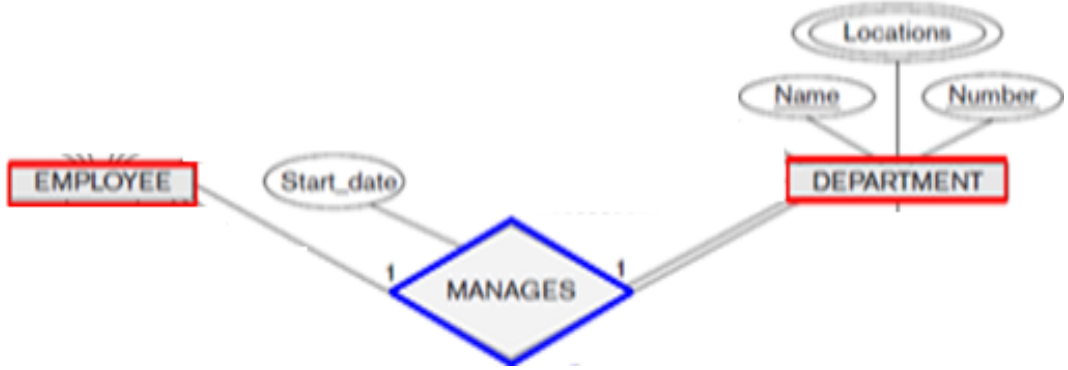


3.6 ER Design for the COMPANY Database

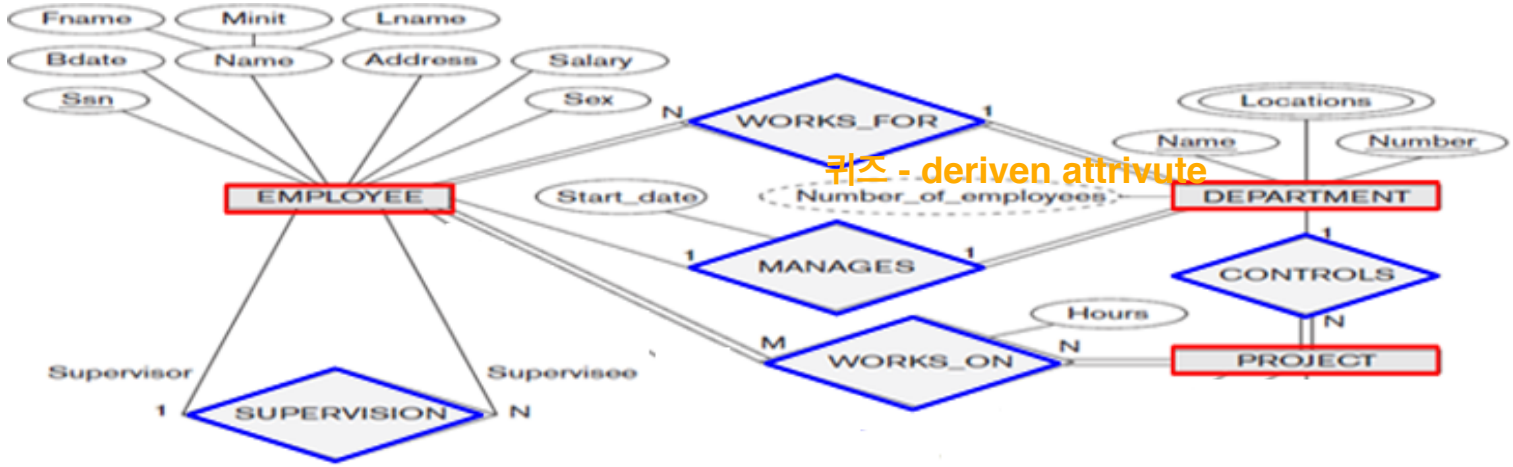
전체를 한번에 만드는 방법은 힘드므로 합치는 방식으로

- Each department has a name and a number, the several locations, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department.

카디널리티나 포셔널 등 문서에 제대로 나와있지 않지만 문맥상 기입할 수도 있음



- An employee has a name, Ssn, address, salary, sex, and birth date. An employee is assigned to one department, but may work on several projects. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee

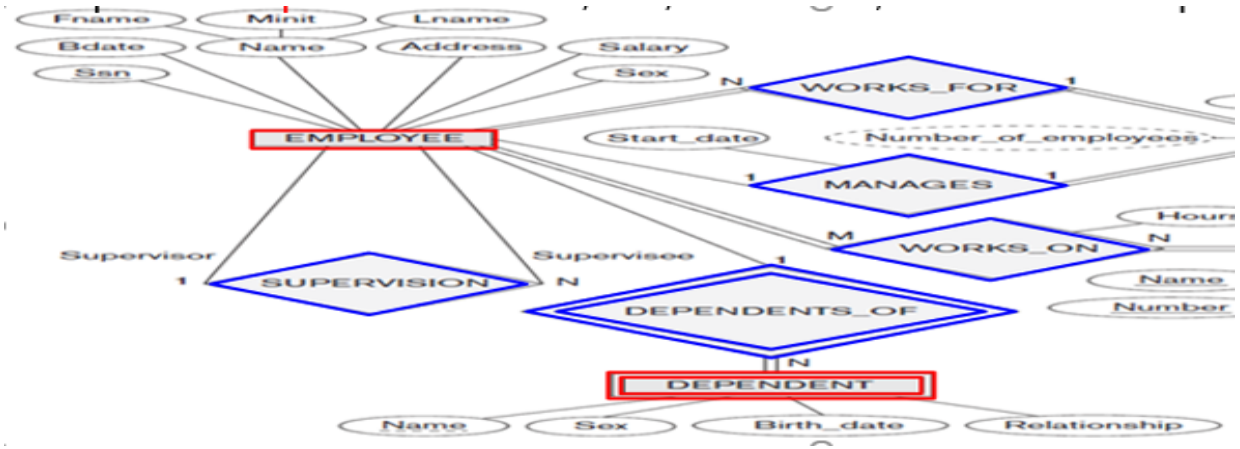


퀴즈 - 리컬시브 릴레이션, 롤네임 명시해야 함

3.6 ER Design for the COMPANY Database

- A department controls a number of projects, each of which has a name and a number, and a location

- We keep track of the dependents of each employee for insurance purposes. We keep each dependent's name, sex, birth date, and relationship to the employee



퀴즈 - 맵핑 하고 정규화과정 필요

Example of ER Design

- Draw an ER diagram about a university's registration office. The office manages the information about a professor, a student, and a course. A professor can teach several courses and supervises the several students. A grade is recorded about each student-course pair.

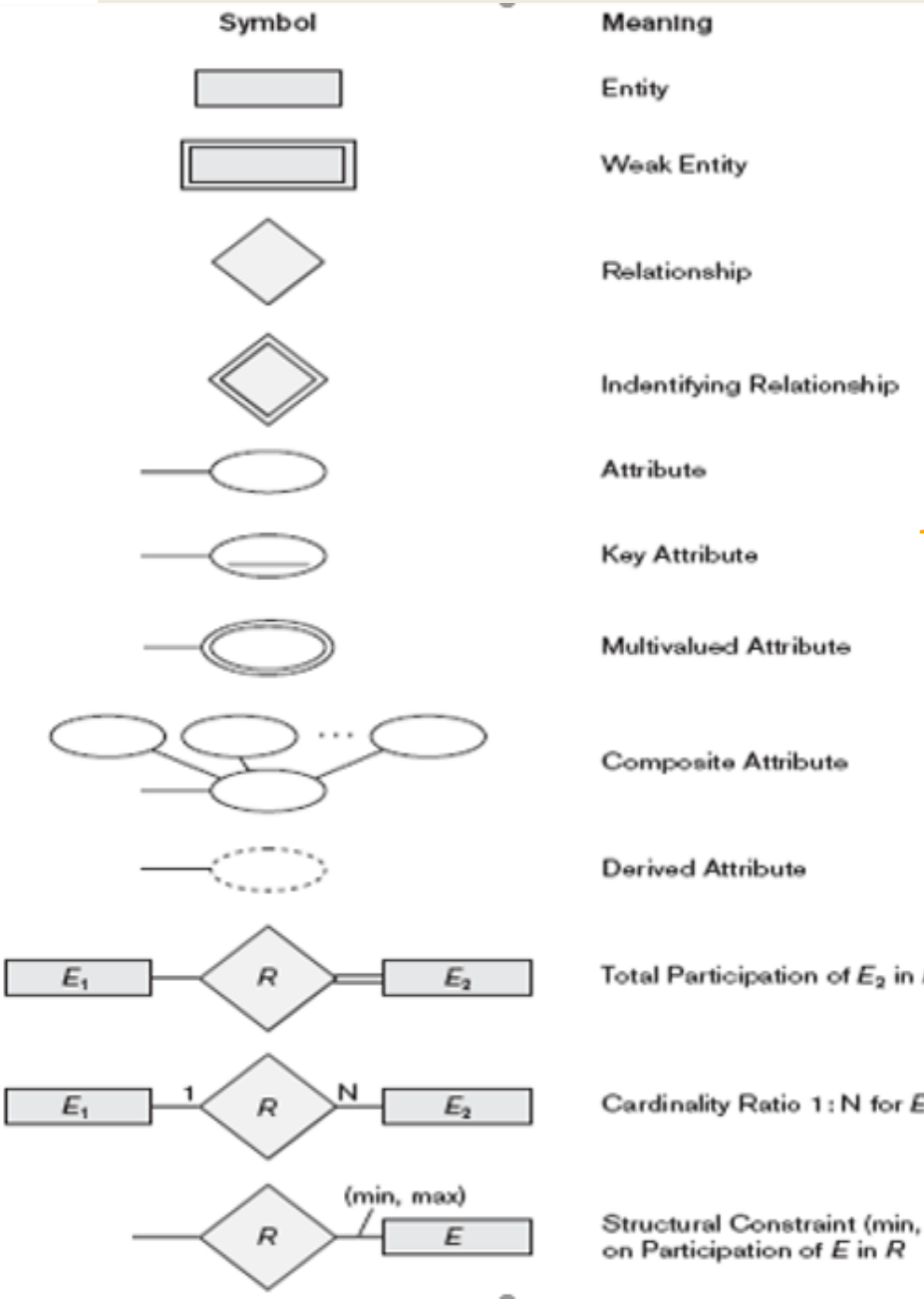
사진

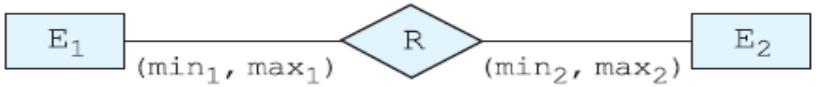
- Draw an ER diagram about the hospital. There are a series of patients and doctors in the hospital. Several test records are associated with a patient.

사진

시험에 나온다면 이렇게 추상적으로는 안나옴
그러나 명확한 것은 알아서 써야함
필요하면 답안지에 ~이러한 가정을 했다고 쓸 수 있음

3.7.1 Summary of Notation for ER Diagrams





[그림 5.16] 카디날리티의 최소값과 최대값

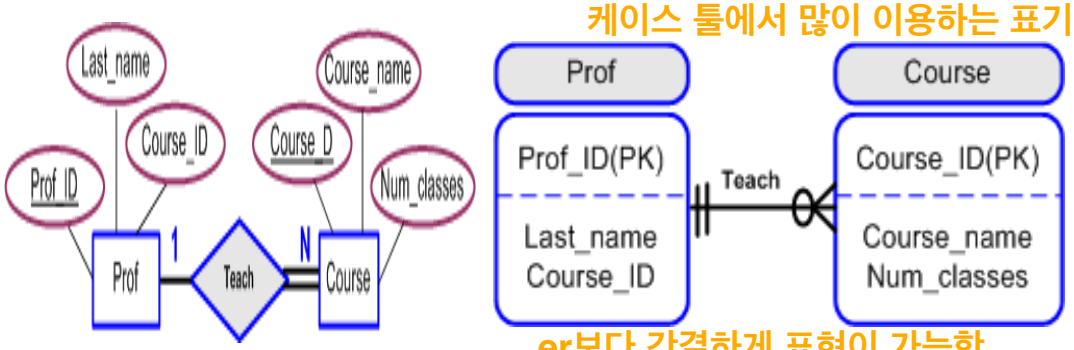
〈표 5.3〉 카디날리티들의 몇 가지 유형

관계	(min1, max1)	(min2, max2)	그래픽 표기
1 : 1	(0, 1)	(0, 1)	1 — 1
1 : N	(0, *)	(0, 1)	1 — N
M : N	(0, *)	(0, *)	M — N

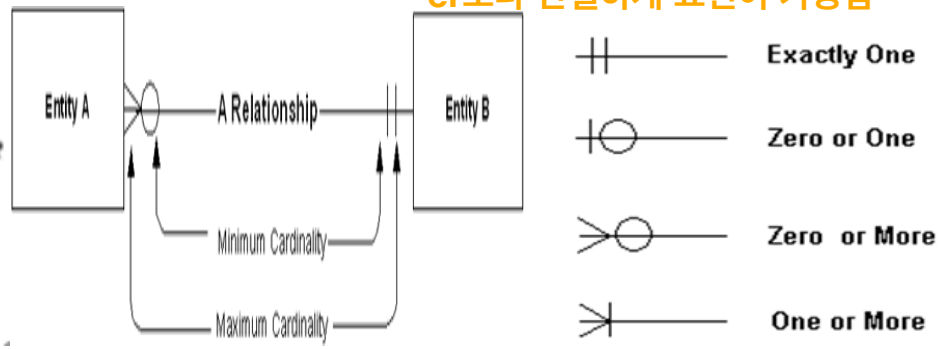
1:1 1:N 등이 아니라 이렇게 명확히 할 수도 있음, 예시 교재 figure7.16?

Crow's foot notation

케이스 툴에서 많이 이용하는 표기법



er보다 간결하게 표현이 가능함

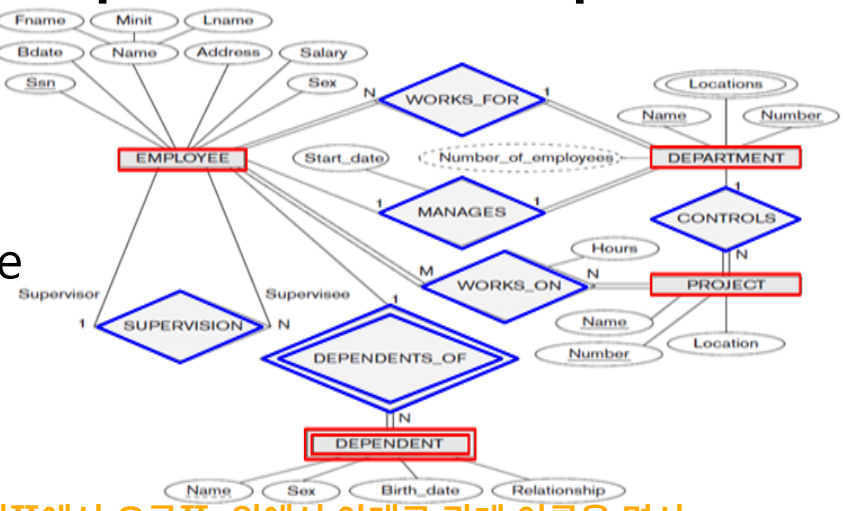


3.7.2 Proper Naming of Schema Constructs

- **Naming conventions**
  - Choose names that convey the meanings attached to different constructs in the schema 의미있는 이름을 지어라
  - In our ER diagrams, we use the convention that entity type and relationship type names are uppercase letters, attribute names have their initial letter capitalized, and role names are lowercase letters 이렇게 대소문자 구분하는 것이 좋다
- **As a general practice, given a narrative description of the DB requirements**

전에 이야기 했던 팁

  - Nouns give rise to entity type names
  - Verbs indicate names of relationship types
  - Attribute names generally arise from additional nouns or adjectives that describe the nouns corresponding to entity types



- **Choose binary relationship names to make ER diagram readable from left to right and from top to bottom**

왼쪽에서 오른쪽, 위에서 아래로 관계 이름을 명시

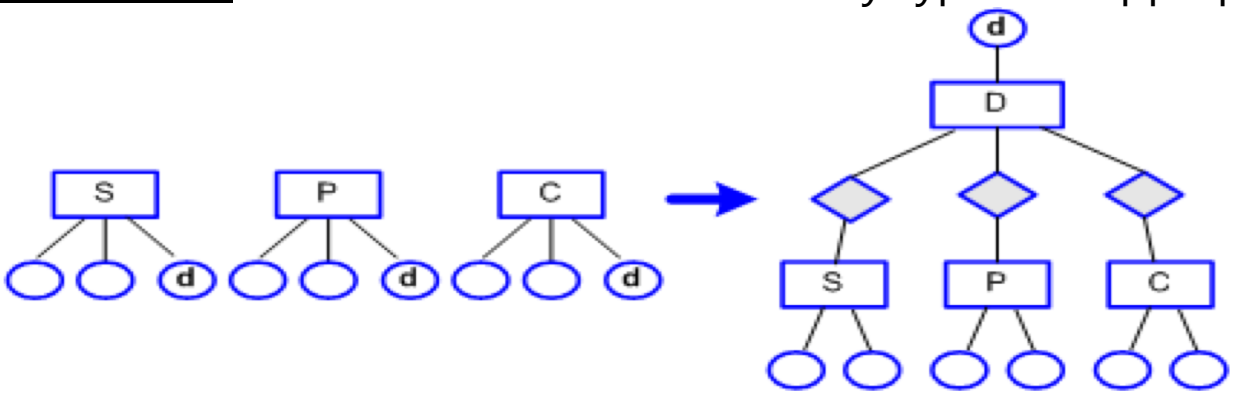
  - We have one exception to the convention in the Fig.—the DEPENDENTS_OF relationship type, which reads from bottom to top
  - To change this to read from top to bottom, we could rename the relationship type to HAS_DEPENDENTS

has dependent가 더 바람직

3.7.3 Design Choices for ER Design

- **It is occasionally difficult to decide whether a particular concept in the miniworld should be modeled as an entity type, an attribute, or a relationship type**
- **In general, the schema design process should be considered an iterative refinement process until the most suitable design is reached**
- **Some of the refinements that are often used include the following:**
  - **A concept may be first modeled as an attribute**
    - Refined into a relationship if the attribute is a reference to another entity type
  - **Attribute that exists in several entity types may be elevated to an independent entity type**
    - Suppose that a UNIVERSITY DB has STUDENT, PROF, and COURSE entity types and each has an attribute Department in the initial design;
    - Then a designer may choose to create an entity type DEPARTMENT with an attribute Dname and relate it to the three entity types via appropriate relationships

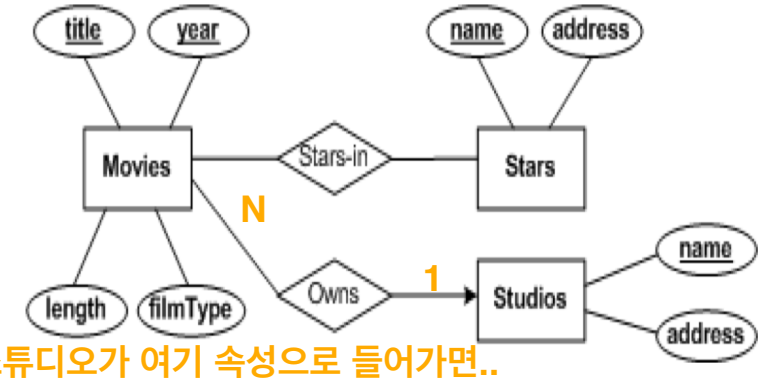
한 번 했다고 끝나는게 아니라 계속 개선 작업을 통해서 재정의 과정을 거쳐야 함



3.7.3 Design Choices for ER Design

- **Some other refinements that are often used include the following:**
  - Were we wise to make Studios an entity type? Should we instead have made the name and address of the Studios be attributes of Movies and eliminated the studio entity type?
  - 엔티티 Studios 배제하고 Movies 속성으로 스튜디오 이름과 스튜디오 주소를 사용한다면?
    - 각 영화마다 반복적으로 스튜디오 주소를 반복한다.
    - 특정 스튜디오가 아직 소유하고 있는 영화가 없다면 스튜디오 정보는 저장할 길이 없다.
  - 반면 스튜디오 이름만 사용한다면 Movies 속성으로 스튜디오 이름 사용 가능

따라서 독립적으로 구성하는게 바람직



만약 스튜디오가 여기 속성으로 들어가면..

서로 다른 정보가 테이블로 매핑되서 생기는 문제

EMP_DEPT (Redundancy)

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

DEPT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

정규화 과정

EMP

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

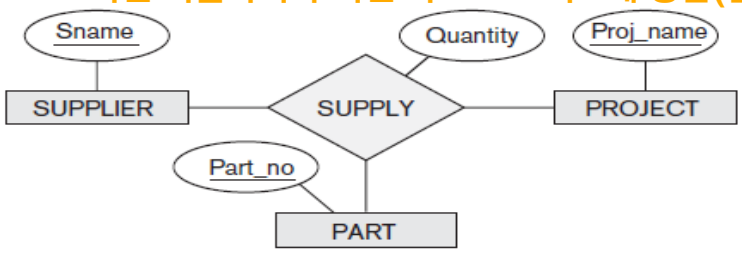
  - Should we model a price as an attribute or an entity set of a relationship type?
    - 왼쪽 그림은 공급자는 동일한 상품을 수량에 따라 서로 다른 가격으로 공급할 수 없다.
    - 반면 오른쪽 그림 같이 가격을 엔티티 타입으로 모델링하면 이 문제를 해결할 수 있다.

오른쪽이 더 많은 의미를 내포하고 있음

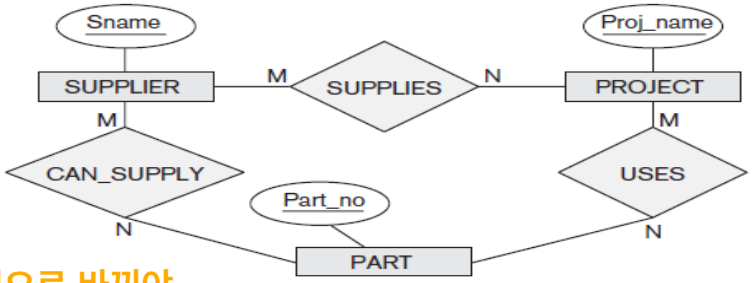
3.9 Relationship Types of Degree Higher than Two

- **Degree of a relationship type**
  - Number of participating entity types.
  - Binary, ternary, etc.
  - In general, an n-ary relationship is not equivalent to n binary relationships.
    - (s, p) & (j, p) & $(s, j) \neq > (s, j, p)$

어떤 서플라이가 어떤 파트 프로젝트에 공급(밀접)



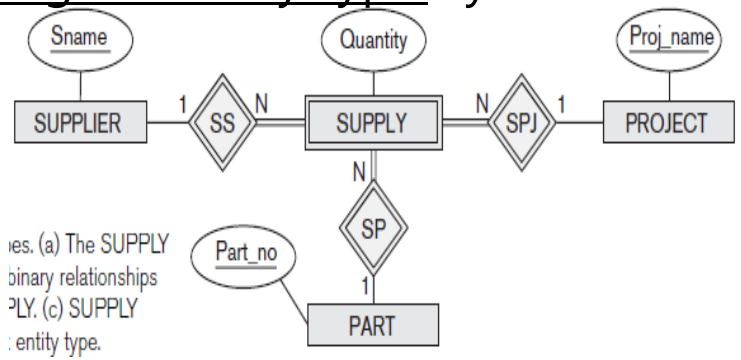
항상 두 가지가 같은 의미가 아님 따라서 아래처럼 만듬



- **Some DB design tools permit only binary relationships** 툴이 바이너리 관계만 지원한다면 이런식으로 바꿔야
  - A ternary relationship can be represented as a weak entity type with no partial key and three identifying relationships.

weak는 종속되는 것이기 때문에

    - Hence, an entity in the weak entity type SUPPLY is identified by the combination of its three owner entities from SUPPLIER, PART, and PROJECT.
  - A ternary relationship can be represented as a regular entity type by introducing an artificial or surrogate key.
    - In this example, a key attribute Supply_id could be used for the supply entity type, converting it into a regular entity type.



퀴즈 - 서플라이 키? 세개를 합쳐서

Summary

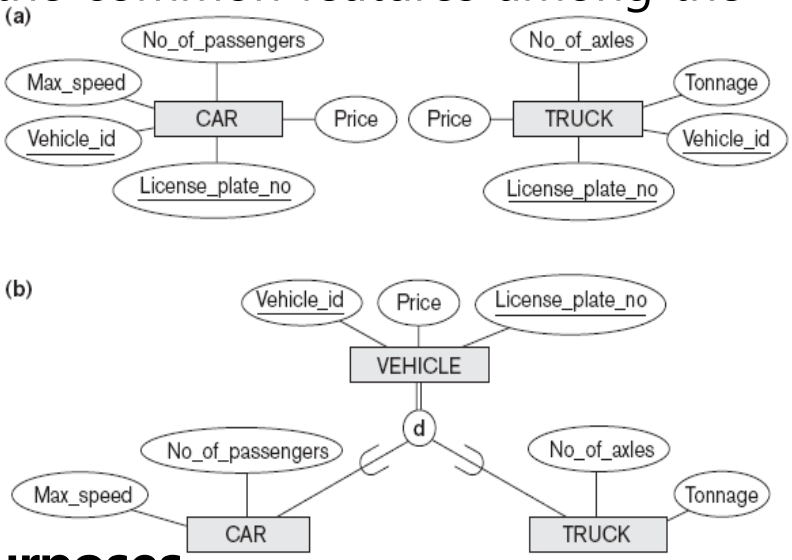
- **Entity-Relationship (ER) model**
  - Popular high-level conceptual data model where a real world can be represented by entities, attributes, and relationships with some constraints
- **EER model (Ch4) improves its representational capabilities with additional concepts such as generalization, specialization, and aggregation**

이 두 개 객체지향(밑의 사진)

  - Generalization is the process of defining a generalized entity type from the given entity types. For this, we can identify the common features among the given entity types and generalize them into a single superclass of which the original entity types are special subclasses

b->a / a->b

  - Specialization process can be viewed as being functionally the inverse of the generalization process



- **ER model is usually used for educational purposes**
  - Crow's foot notation, Barker's notation, Information Engineering notation, etc
  - CASE tool: Oracle designer, ER-Studio, ERwin, Power designer, UML, etc

인테테 안에 속성이 들어감 직선만 들어가서 간결하게 표현 가능