Data Modeling Using the Entity-Relationship Model

406.426 Design & Analysis of Database Systems

Jonghun Park

jonghun@snu.ac.kr

Dept. of Industrial Engineering

Seoul National University

outline

- overview of database design process
- example database application (COMPANY)
- ER model concepts
  - entities and attributes
  - entity types, value sets, and key attributes
  - relationships and relationship types
  - weak entity types
  - roles and attributes in relationship types
- ER diagrams: notation
- ER diagram for COMPANY schema
- alternative notations – UML class diagrams, others

database design process

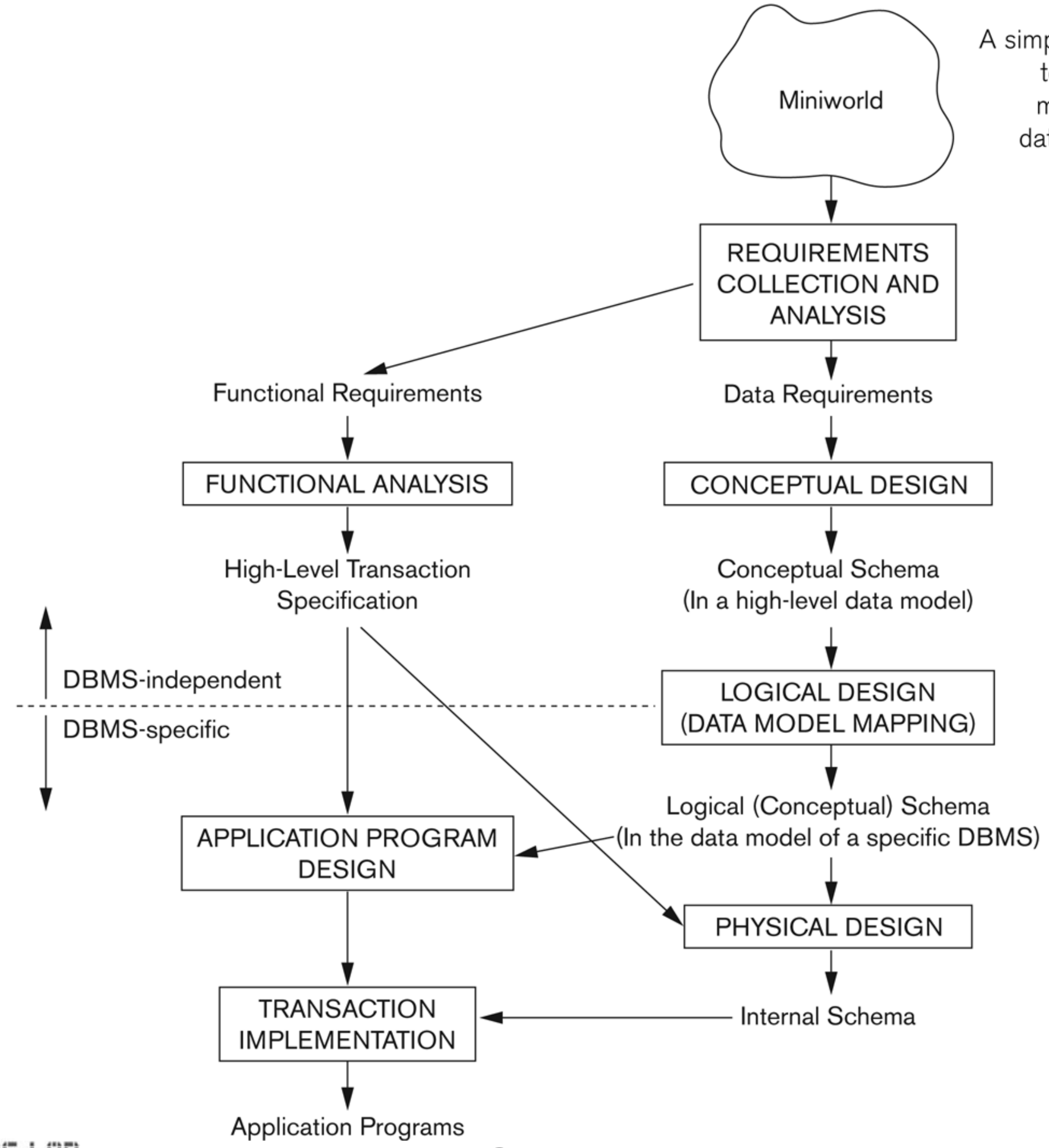

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

an example DB application

- data **requirements** of COMPANY database
  - The company is organized into **DEPARTMENT**s. Each department has a **name**, **number** and an employee who **manages** the department. We keep track of the **start date** of the department manager. A department may have **several locations**
  - Each department controls a number of **PROJECT**s. Each project has a unique **name**, a unique **number** and is located at a **single location**.
  - We store each **EMPLOYEE**'s **SSN**, **address**, **salary**, **sex**, and **birthdate**. Each employee works for **one department** but may work on **several projects**. We keep track of the number of **hours** per week that an employee currently works on each project. We also keep track of the direct **supervisor** of each employee.
  - Each employee may have a number of **DEPENDENT**s. For each dependent, we keep track of their **name**, **sex**, **birthdate**, and **relationship** to employee.

ER model

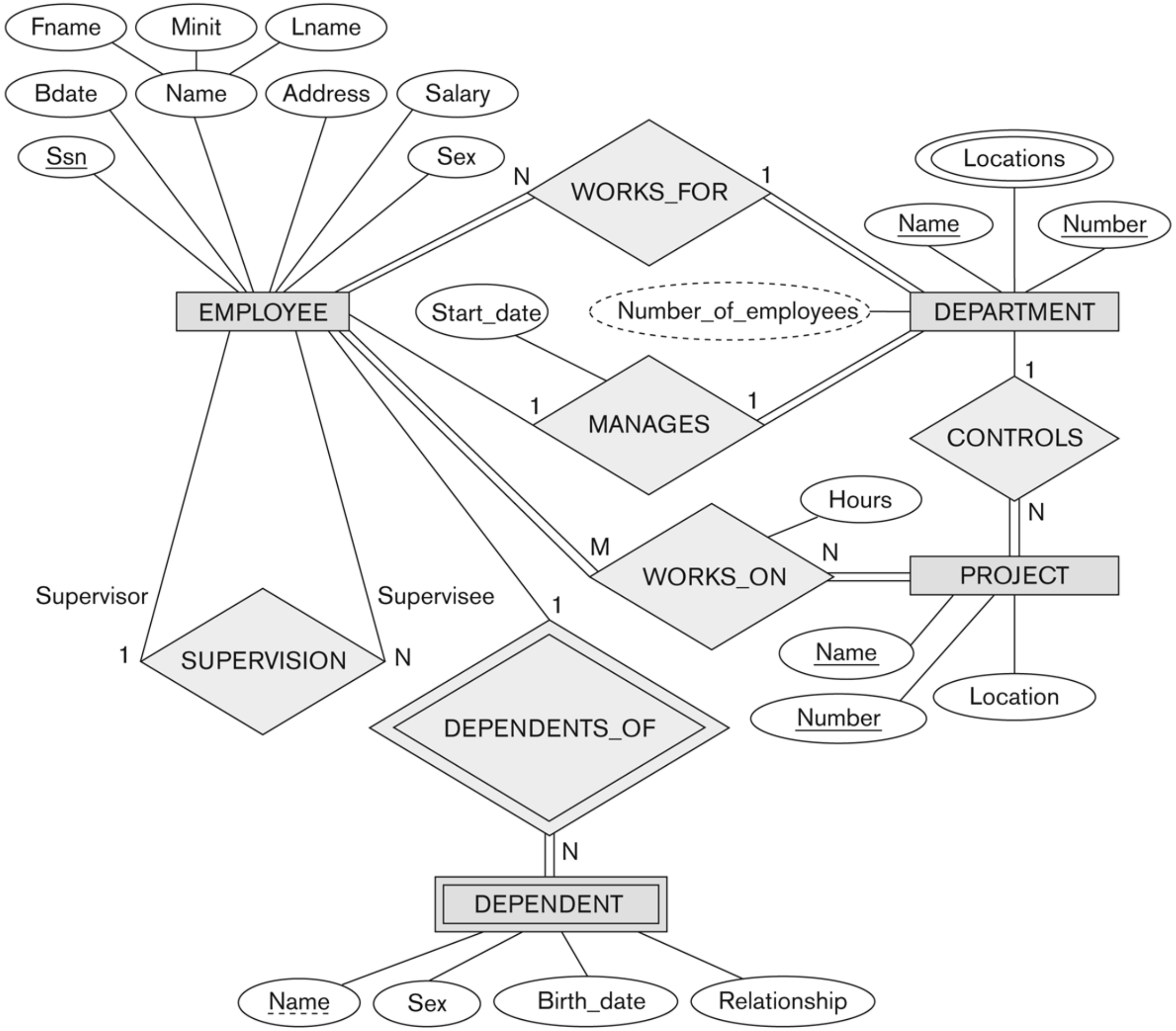
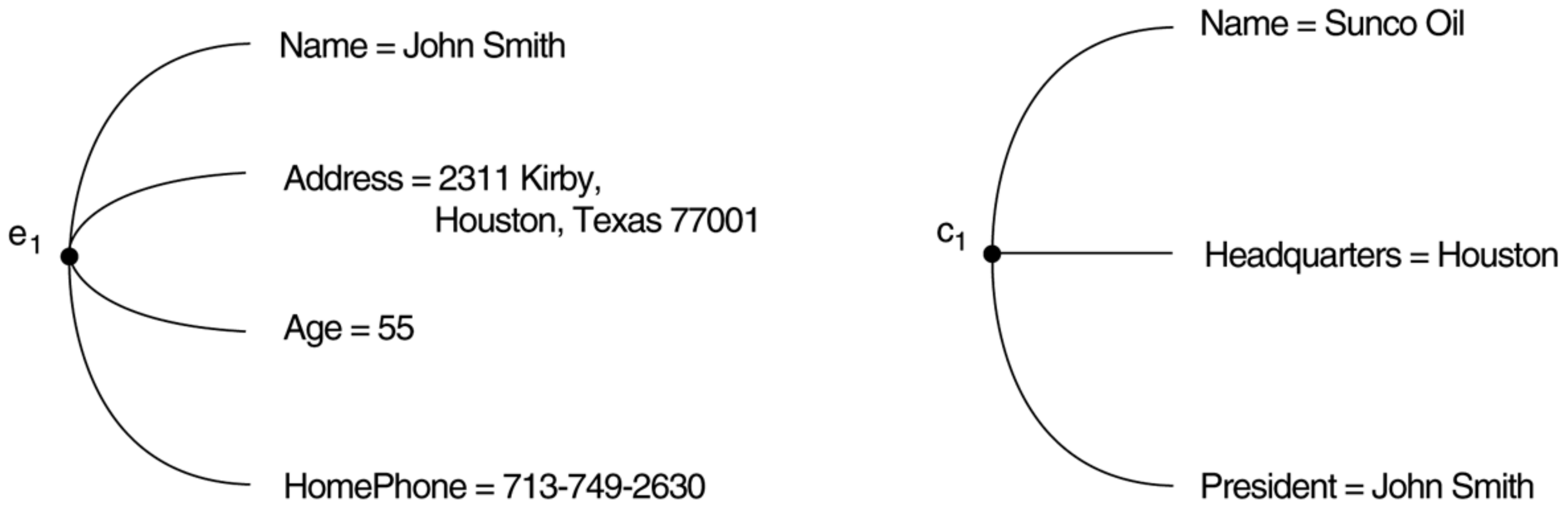


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

- ER mode describes data as **entities**, **relationships**, and **attributes**

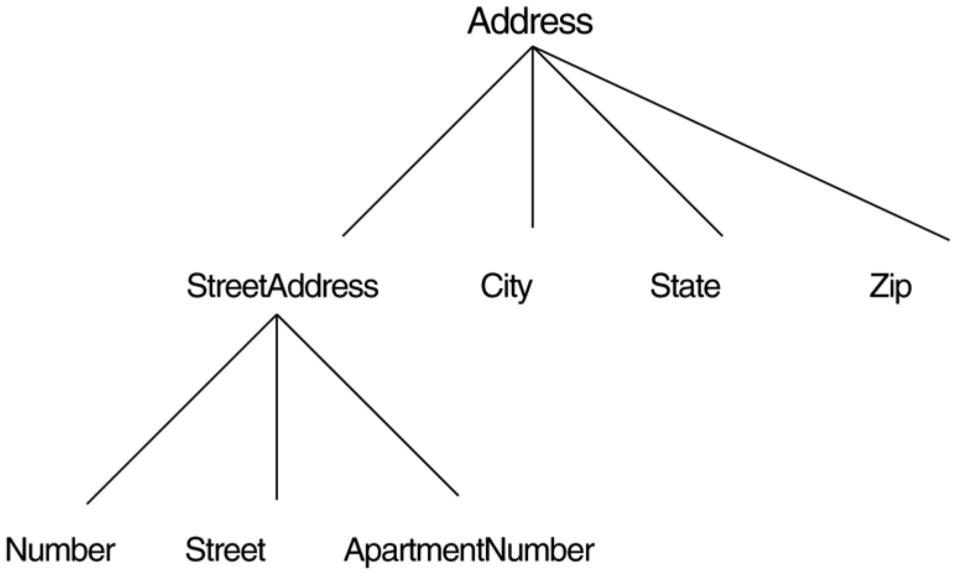
entities and attributes

- **entity**: a “thing” in the real world with an independent existence
- **attributes**: particular **properties** that describe an entity



types of attributes

- **composite** vs. **simple** attributes
  - composite attributes can be divided into **smaller subparts**, which represent more basic attributes with independent meanings (e.g., address attribute)
  - attributes that are not divisible are called **simple** or **atomic** attributes
  - the value of a composite attribute is the **concatenation** of the values of its constituent simple attributes



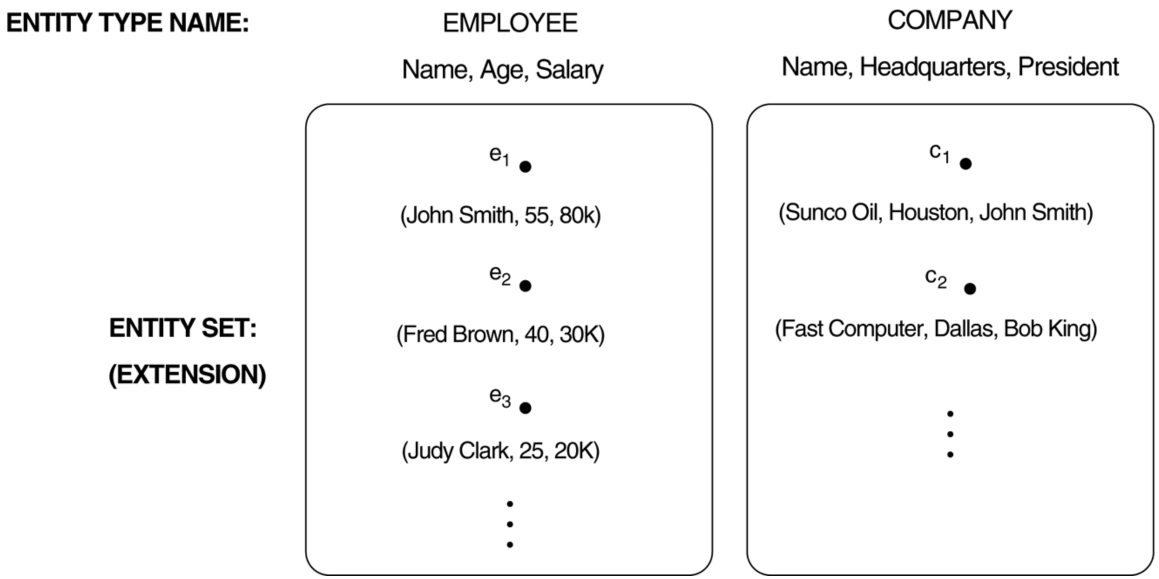
types of attributes

- **single-valued** vs. **multi-valued**
  - example: Age vs. CollegeDegrees
- **stored** vs. **derived**
  - Example: Age <- BirthDate and CurrentDate
- **null** values
  - unknown (missing, not known), not applicable
- **complex** attributes
  - (): composite attribute
  - {}: multivalued attribute
  - example: a person can have more than one residence and each residence can have multiple phones

{AddressPhone({Phone(AreaCode,PhoneNumber)},
Address(StreetAddress(Number,Street,ApartmentNumber),
City,State,Zip)) }

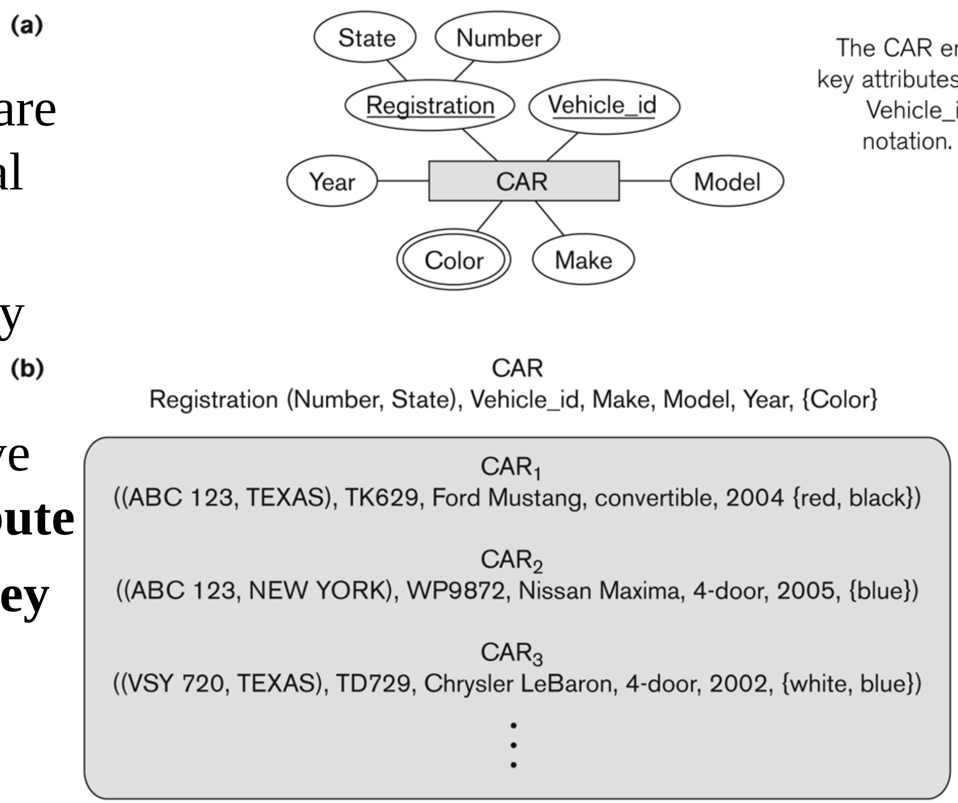
entity types and entity sets

- entity type
  - a collection of **entities** that have the **same attributes**
  - described by its **name** and **attributes**
  - describes the **schema** (or intension) for a set of entities
- entity set
  - the **collection of all entities** of a particular entity type in the database at any point
  - describes the **extension** of a entity type



key attributes

- **key attribute**
  - an attribute whose values are **distinct** for each individual entity in the entity set
  - used to **identify** each entity **uniquely**
  - some entity types may have **more than one key attribute**
  - an entity type **without a key attribute** is called a **weak entity type**
- **composite** attribute
  - a set of attributes of which the combination of the attribute values are unique



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

CAR$_1$
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

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})

⋮

Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

value sets (domains) of attributes

- **value set**: the set of values that may be assigned to an attribute for each individual entity
- an attribute A of entity type E whose value set is V can be defined as a **function** from E to the power set $P(V)$ of V:
  - $\boldsymbol{A: E \rightarrow P(V)}$
- let $A(e)$ be the value of attribute A for entity e
  - $A(e)$ is a **singleton set** for each entity e in E for single-valued attributes
  - no restriction on multivalued attributes
- for a composite attribute A, the value set V is the **Cartesian product** of $P(V_1)$, $P(V_2)$, …, $P(V_n)$, where V_1, V_2, …, V_n are the value sets of the simple component attributes that form A:
  - $\boldsymbol{V = P(V_1) \times P(V_2) \times \ldots \times P(V_n)}$

initial conceptual design of the COMPANY DB

- based on the requirements, we can identify **4 entity types**
- what are the key attributes?

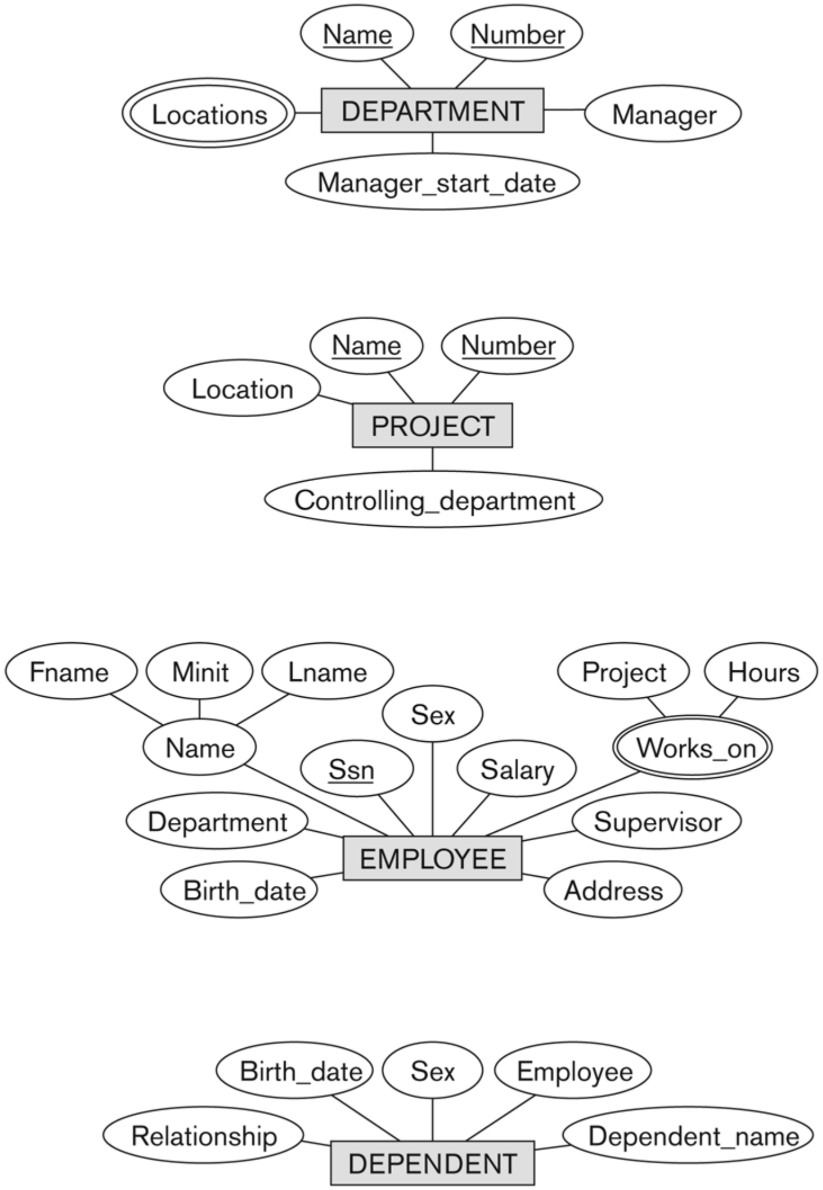


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

relationship types, sets, and instances

- whenever an **attribute** of one entity type **refers to** another **entity type**, some relationship exists
  - e.g., the attribute Manager of DEPARTMENT
  - in the ER model, these references should **not be represented as attributes but as relationships**
- in the initial design of entity types, **relationships are typically captured in the form of attributes**

relationship types, sets, and instances

- a **relationship type** R among n entity types, E_1, E_2, …, E_n defines a set of **associations** (or a relationship set) among entities from these entity types
- the **relationship set** R is a set of **relationship instances** r_i, where each r_i **associates** n individual entities (e_1, e_2, …, e_n), and each entity e_j in r_i is a member of entity type E_j, 1 <= j <= n
- hence, a **relationship type** is a **subset of the Cartesian product** $E_1 \times E_2 \times \ldots \times E_n$
- each of the entity types E_1, E_2, …, E_n is said to **participate in** the relationship type R
- each of the individual entities e_1, e_2, …, e_n is said to **participate in** the relationship instance $r_i = (e_1, e_2, \ldots, e_n)$

example

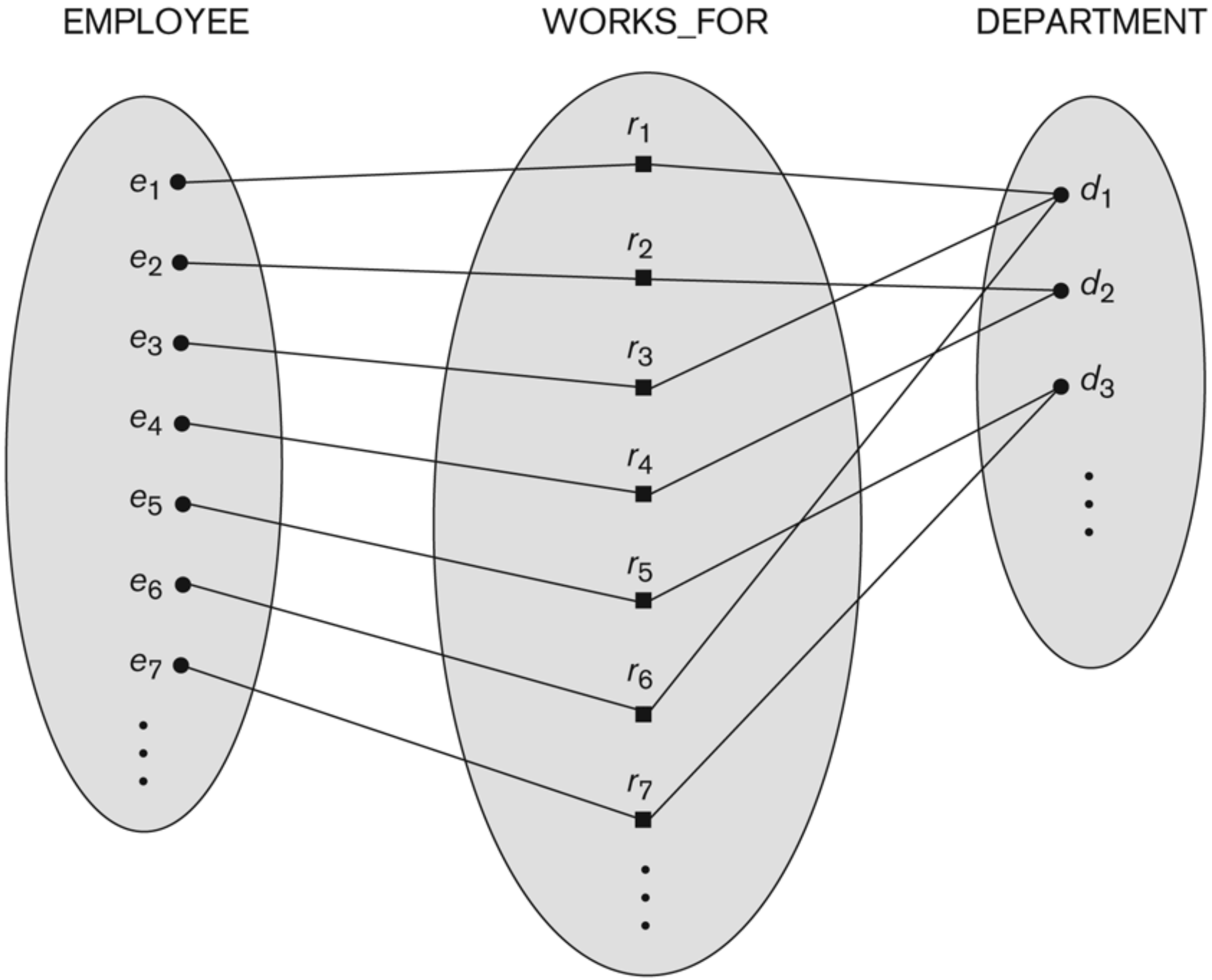
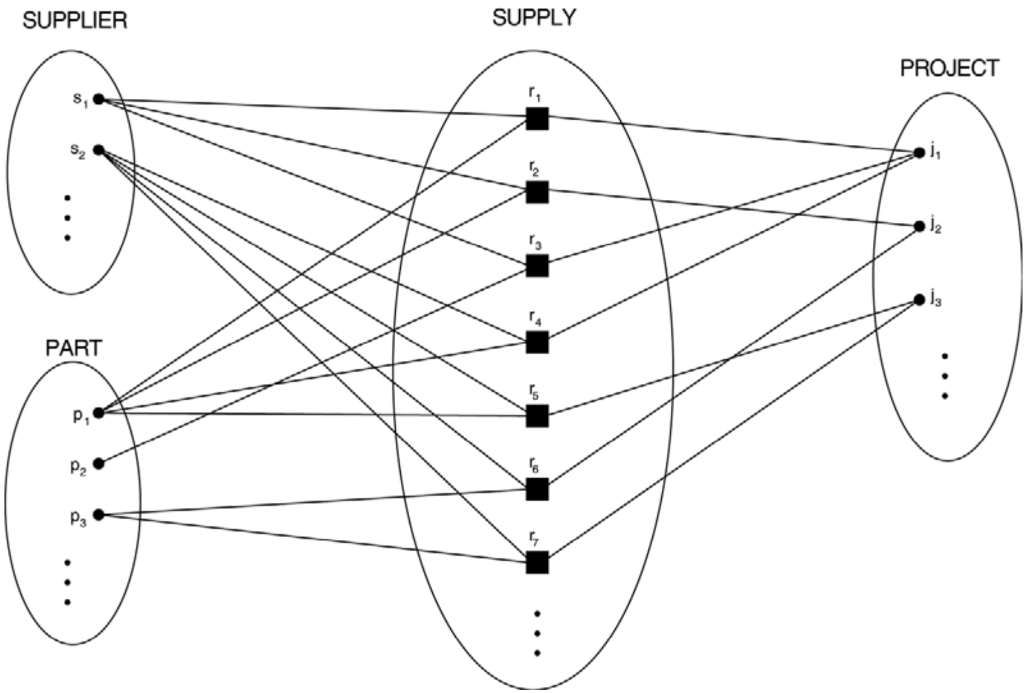


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

relationship degree and relationships as attributes

- **degree** of a relationship: **# of participating entity types**
  - binary, ternary, …
- **relationships as attributes**
  - in case of binary relationships
  - example: WORKS_FOR relationship type -> as a Department attribute of EMPLOYEE or Employees attribute of DEPARTMENT



role names and recursive relationships

- each entity type that participates in a relationship type plays particular **role in the relationship**
  - example: employee and department roles in WORKS_FOR relationship
- the same entity type may participate **more than once in a relationship type in different roles** -> recursive relationships
  - example: 1 for supervisor role, 2 for supervisee role

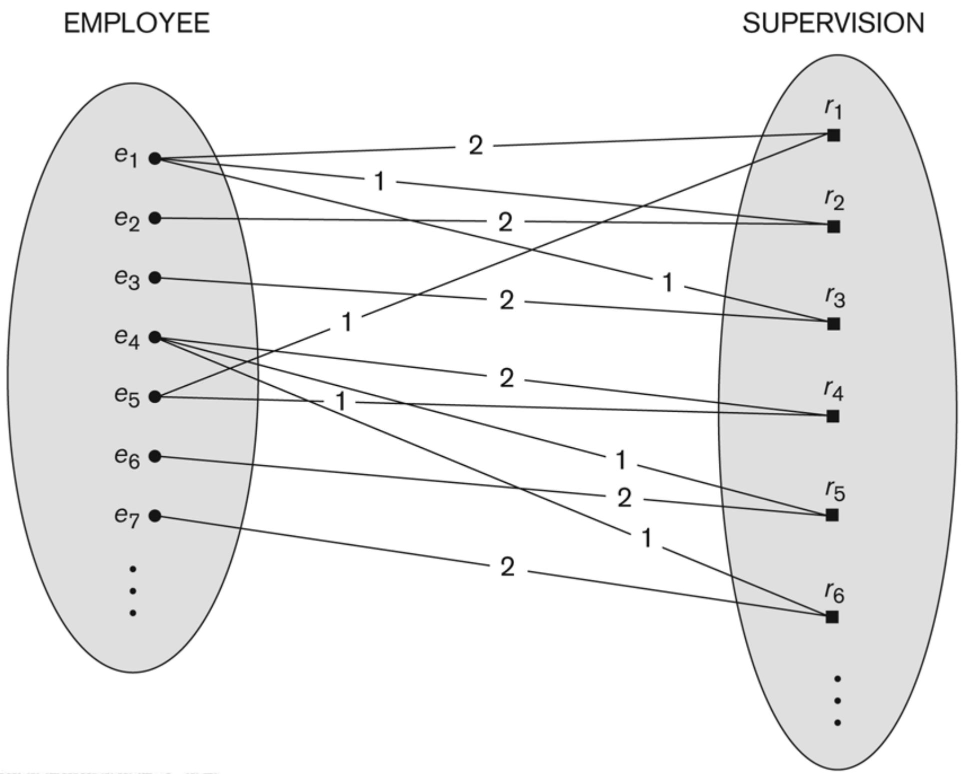
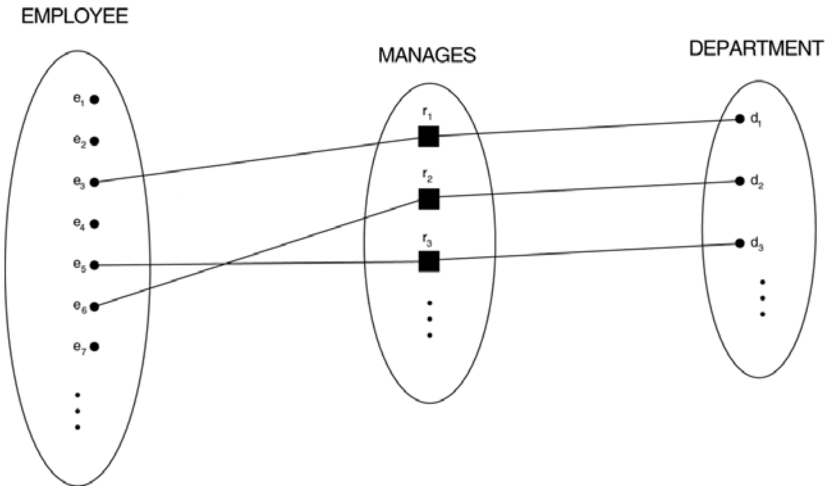
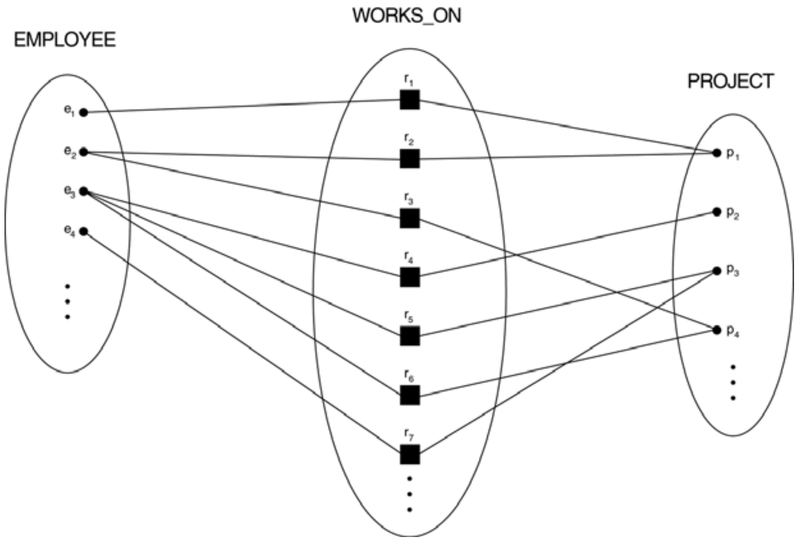


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

relationship constraints (structural constraints)

- relationship types usually have certain constraints that **limit the possible combinations of entities** that may participate in the corresponding relationship set
- **cardinality ratio** for binary relationships
  - specifies the **maximum number** of **relationship instances** that an entity can participate in
  - example: DEPARTMENT:EMPLOYEE = 1:N
  - can be 1:1, 1:N, N:1, M:N





relationship constraints

- participation constraints
  - specifies whether the existence of an entity depends on its being related to another entity via the relationship type
  - specifies the **minimum number of relationship instances** that each entity can participate in
  - “**total**” (or existence dependency) if every entity in a entity set must be related to an entity in some other entity set via some relationship
    - example: the participation of EMPLOYEE in WORKS_FOR is total if every employee must work for a department
  - “**partial**”, otherwise
    - example: the participation of EMPLOYEE in the MANAGES relationship type

attributes of relationship types

- relationship types can also have attributes
  - example: an attribute Hours for the WORKS_ON relationship type
- attributes of 1:1 relationship types can be **migrated to either one** of the participating entity types
  - example: StartDate attribute the MANAGES relationship can be an attribute of either EMPLOYEE or DEPARTMENT
- for an 1:N relationship type, a relationship attribute can be **migrated only to the entity type on the N-side** of the relationship
  - example: if the WORKS_FOR relationship type has an attribute StartDate, the attribute should be included as an attribute of EMPLOYEE
- for M:N relationship types, some attributes may be determined by the combination of participating entities in a relationship instance, **not by any single entity**
  - such attributes **must be specified as relationship attributes**
  - example: Hours attribute of the M:N relationship WORKS_ON

weak entity types

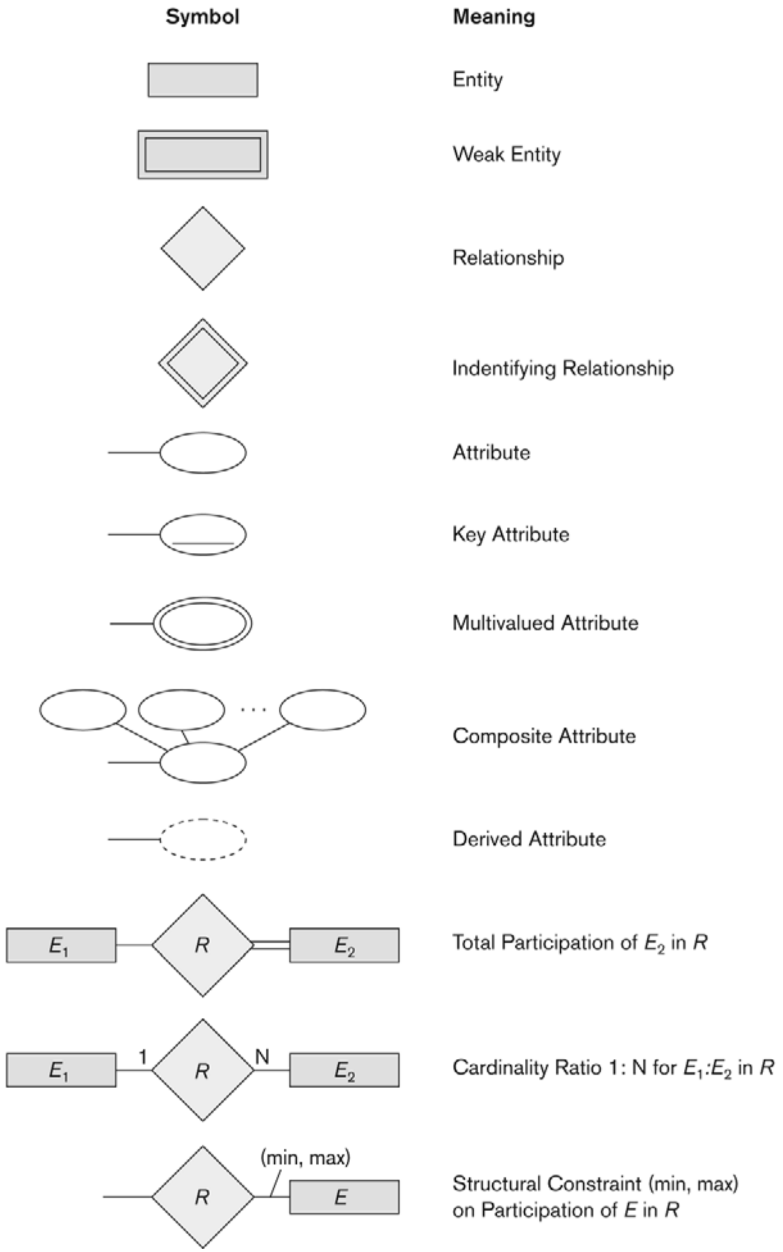
- **weak entity type**: entity type that **does not have key attributes** of its own
  - cf: strong entity type
- entities belonging to a weak entity type are **identified by being related to specific entities** from another entity type (called **identifying entity type**) in combination with one of their attribute values
- **identifying relationship**: the relationship type that relates a weak entity to its identifying entity type
- a weak entity type **always has a total participation constraint w.r.t. its identifying relationship**
- but, **not every existence dependency results in a weak entity type**
  - example: DRIVER_LICENSE: has a existence dependency on PERSON entity, but it is a strong entity
- a weak entity type normally has a **partial key**, which is the **set of attributes that can uniquely identify weak entities** that are related to the same owner entity
  - example: the attribute Name of DEPENDENT entity set
- weak entity types can sometimes be alternatively represented as **complex attributes**
  - example: a multivalued attribute Dependents for EMPOYEE

refining the ER design for the COMPANY DB

- refinement through changing the attributes that represent relationships into relationship types
- should have the **least possible redundancy**
- identified relationship types
  - MANAGES (EMPLOYEE: DEPARTMENT)
    - 1:1, EMPLOYEE: partial participation, DEPARTMENT: total participation
  - WORKS_FOR (DEPARTMENT:EMPLOYEE)
    - 1:N, both are total participation
  - CONTROLS (DEPARTMENT, PROJECT)
    - 1:N, PROJECT: total, DEPARTMENT: partial
  - SUPERVISION (EMPLOYEE:EMPLOYEE)
    - 1:N, both partial, supervisor role:supervisee role
  - WORKS_ON (EMPLOYEE:PROJECT)
    - M:N, attribute: Hours, both total
  - DEPENDENTS_OF (EMPLOYEE:DEPENDENT)
    - 1:N, identifying relationship for DEPENDENT, EMPLOYEE: partial, DEPENDENT: total

conventions for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.



Entity

Weak Entity

Relationship

Indentifying Relationship

Attribute

Key Attribute

Multivalued Attribute

Composite Attribute

Derived Attribute

Total Participation of E_2 in R

Cardinality Ratio 1: N for E_1:E_2 in R

Structural Constraint (min, max) on Participation of E in R

proper naming of schema constructs

- the choice of names for entity types, attributes, relationship types, and roles is not always straightforward
- **nouns** appearing in the narrative tend to give rise to **entity type names**
- **verbs** tend to indicate names of **relationship types**
- **attribute** names generally arise from **additional nouns** that describe the nouns corresponding to entity types

design choices for ER conceptual 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 -> the schema design process should be considered an **iterative refinement process**
- a concept may be first modeled as **an attribute and then refined into a relationship** because it is determined that the attribute is a reference to another entity type
- an **attribute** that exists in several entity types may be **promoted to an independent entity type**
- an **entity type** that is related to only one other entity type may be **demoted to an attribute** of the other entity type

alternative notations for ER diagrams

- association of a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where 0 ≤ min ≤ max and max ≥ 1
- the numbers mean that for **each** entity *e* in *E*, *e* must participate in **at least min** and **at most max relationship instances** in *R* at any point in time
- therefore,
  - min = 0 -> partial participation
  - min > 0 -> total participation

example

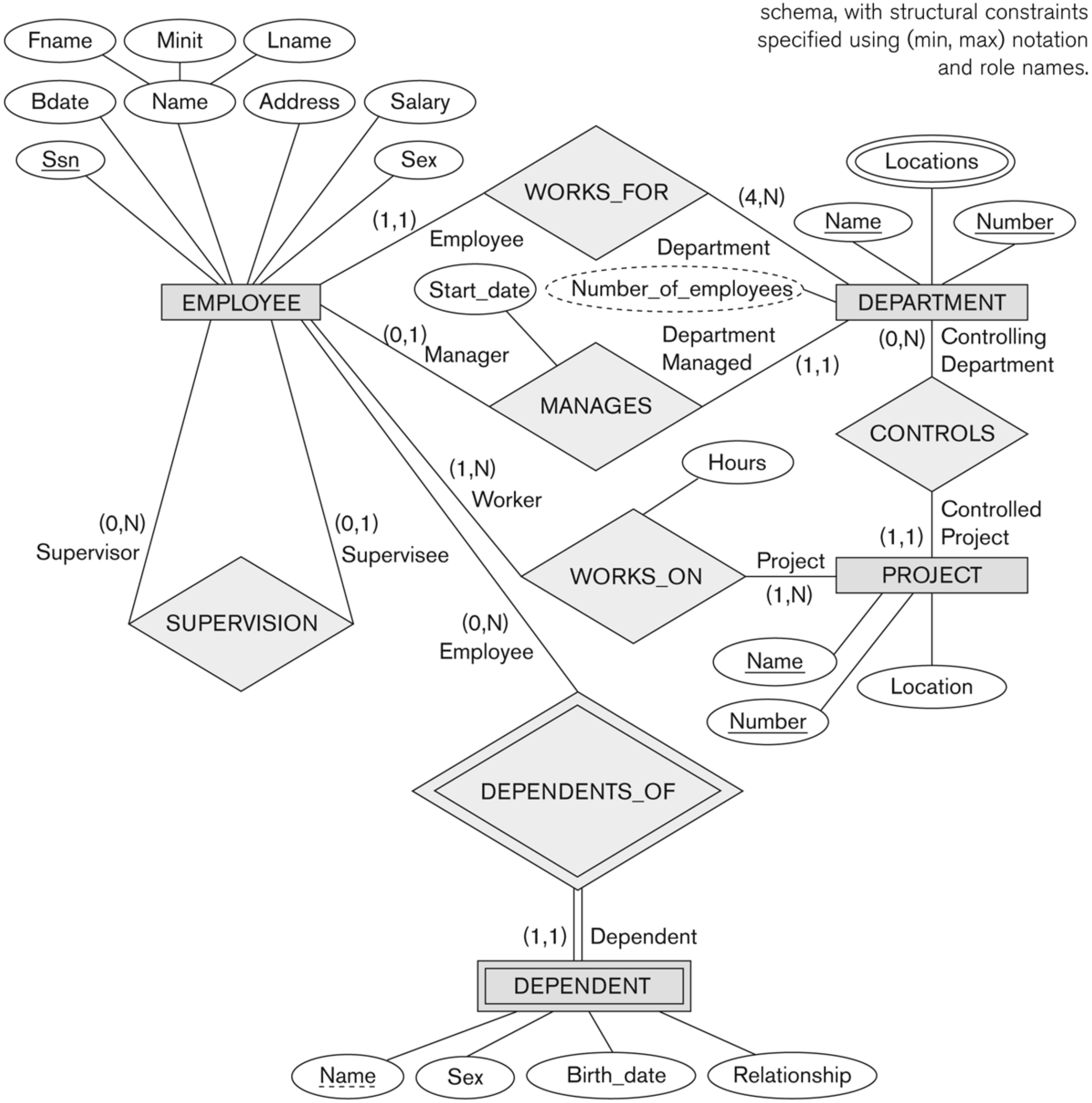
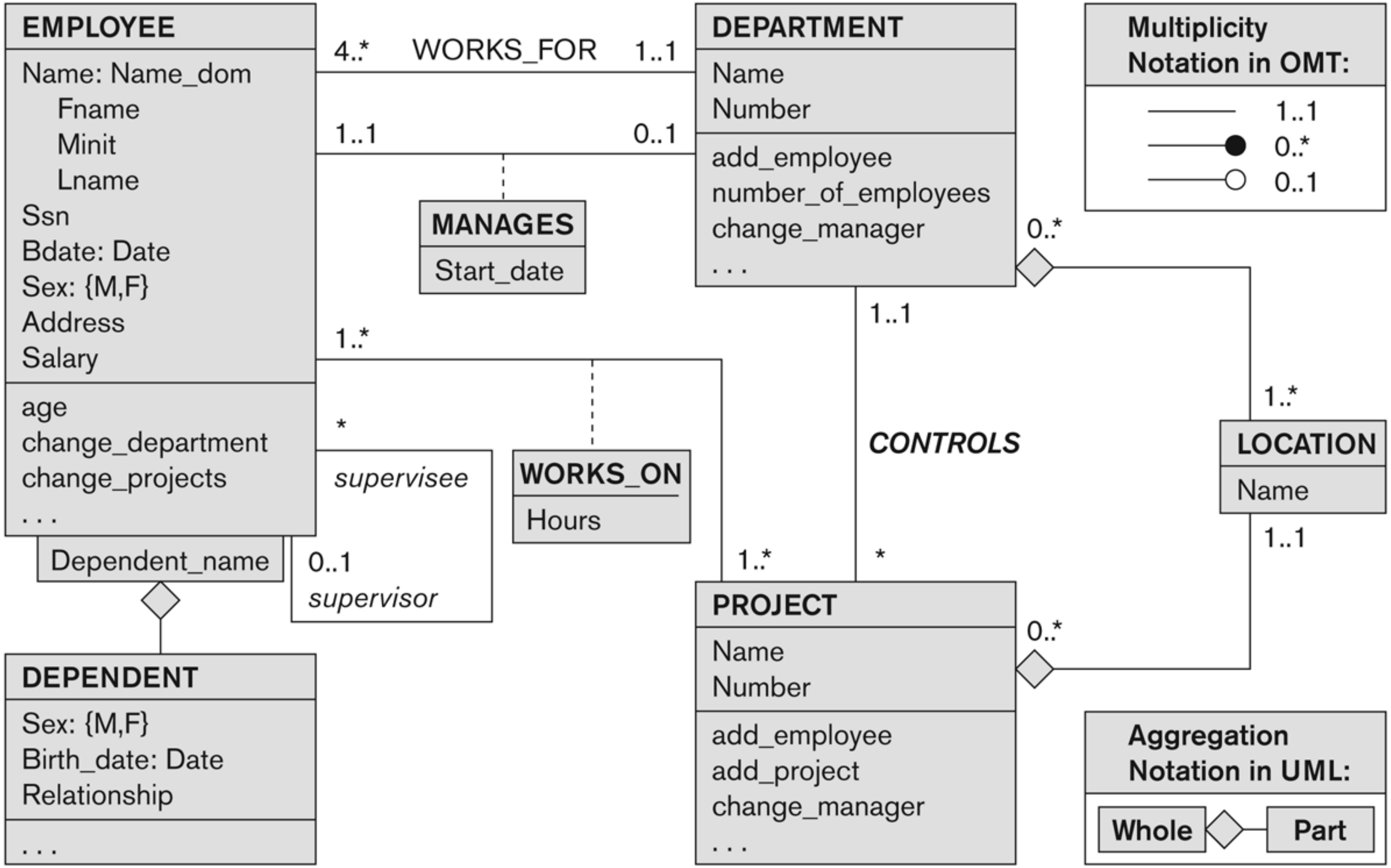


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

another powerful notation: UML

Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.



relationship types of degree higher than 2

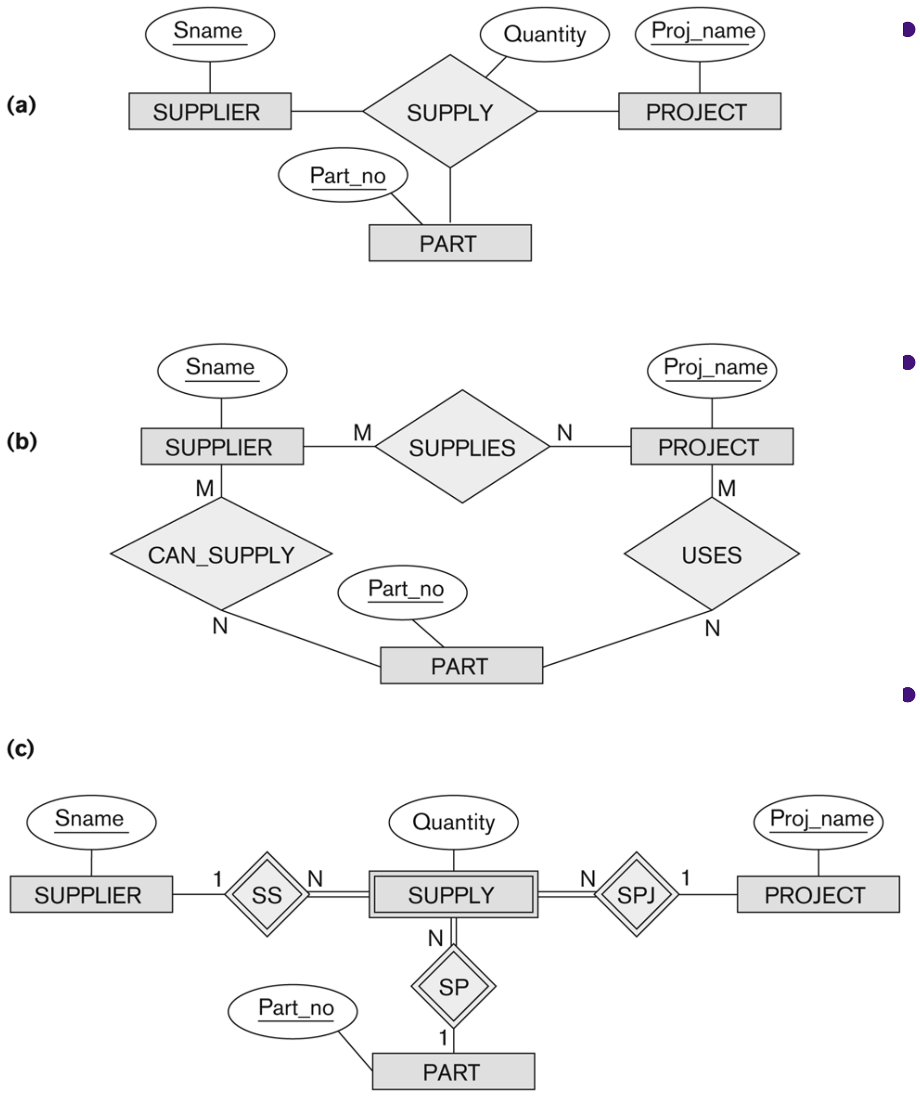


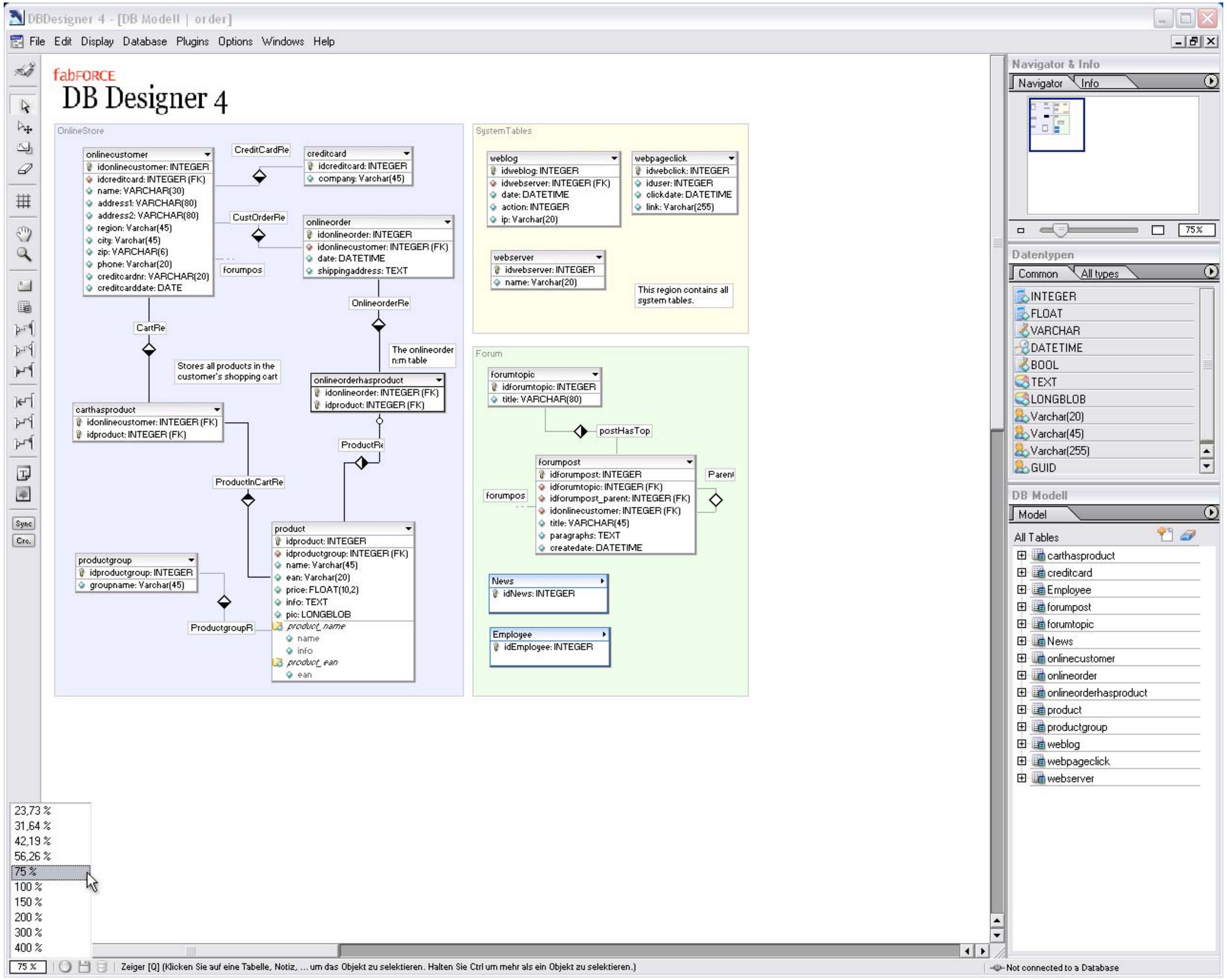
Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

- the relationship set of SUPPLY is a set of relationship instances (*s*, *j*, *p*), where *s* is a SUPPLIER who is supplying a PART *p* to a PROJECT *j*
- the existence of 3 relationship instances (*s*, *p*), (*j*, *p*), and (*s*, *j*) does not necessarily imply that an instance (*s*, *j*, *p*) exists in the ternary relationship SUPPLY
- a ternary relationship such as SUPPLY can be represented as **a weak entity type, with no partial key** and with three identifying relationships

some popular data modeling tools

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Developer 2000/Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (CA)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++

DBDesigner 4



Stores all products in the customer's shopping cart

The onlineorder n:m table

This region contains all system tables.

Zeiger [Q] (Klicken Sie auf eine Tabelle, Notiz, ... um das Objekt zu selektieren. Halten Sie Ctrl um mehr als ein Objekt zu selektieren.)

Not connected to a Database