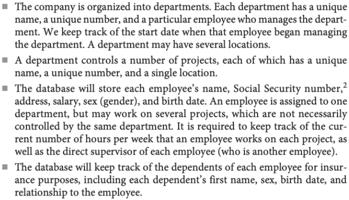
COS221
L04 - Entity-Relationship Conceptual Modelling (Part 2)

(Chapter 7 in Edition 6 and Chapter 3 in Edition 7)

Linda Marshall

27 February 2023

Quick recap - From L03



- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number,[2] address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

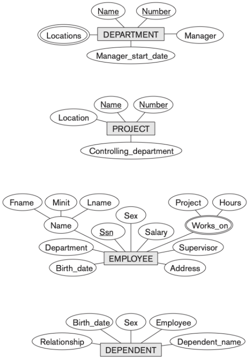


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Identify the following: Key attribute, Composite attribute, Derived attribute, Multivalued attribute, Atomic attribute, Entity type and a Weak entity type

Relationship Types, Sets and Instance

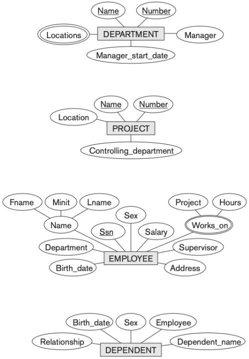


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

A *relationship* is when one *entity type* refers to another *entity type*. For example `DEPARTMENT` refers to an `EMPLOYEE` through the `Manager` attribute.

This kind of association should not be modelled as attributes, but using the *relationship* notation.

Relationship Types, Sets and Instance - Formalised

- A *relationship type*, R, among n entity types $E_1, E_2, E_3, ..., E_n$ defines a set of associations between these entity types – also referred to as *relationship sets*.
- The *relationship set*, R, is a set of *relationship instances*, r_i, where each r_i has n individual entities associated with it, $(e_1, e_2, e_3, ..., e_n)$. Each e_j in r_i is a member of E_j.
- A relationship is a subset of the Cartesian product of $E_1 \times E_2 \times E_3 \times ... \times E_n$. Each E_k participates in the relationship type, R, and each e_k participates in a relationship instance $r_i = (e_1, e_2, e_3, ..., e_n)$.

Note: As with entities, a relationship type and a relationship set are referred to by the same name.

Relationship Types, Sets and Instance - Informally

- Each r_i in R is an association of entities.
- The association includes one entity from each participating entity type.
- Each entity type in a relationship plays a role in the relationship and has a role associated with it.

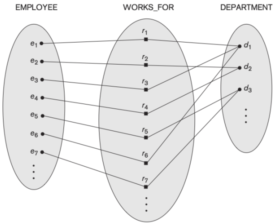


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

WORKS_FOR associates each employee with a department
In the WORKS_FOR relationship, the employee plays a *employee/worker* role and the department a *department/employer* role.

Relationship Types, Sets and Instance - Recursive

- Role names are not always necessary.
- In some instances when it is not clear what the role is, such as in a *recursive relationships*, the role name may be significant.

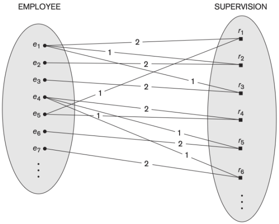


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Relationship Types, Sets and Instance - Degree

- The *degree* of a relationship is the number of participating entity types in the relationship.
- The WORKS_FOR relationship is an example of a *binary* relationship.
- Higher dimension relationships are possible, for example *ternary*, etc.

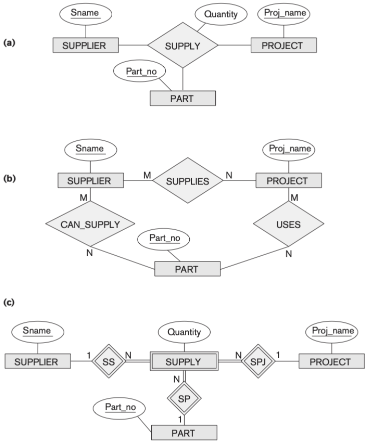


Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Relationship Types, Sets and Instance - Constraints

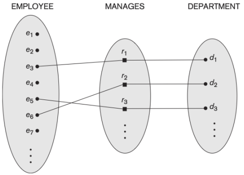
- Relationships may have constraints, referred to as *structural constraints*
- The two (2) main types of structural constraints are:
  - Cardinality ratios
  - Participation

Relationship Types, Sets and Instance - Constraints

Cardinality ratios

- The maximum number of relationship instances that an entity may participate in.
- Possible cardinality ratios are: 1:1, 1:N, N:1, M:N

Figure 3.12
A 1:1 relationship,
MANAGES.



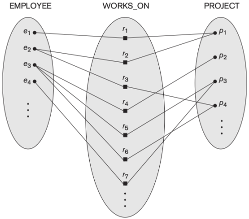
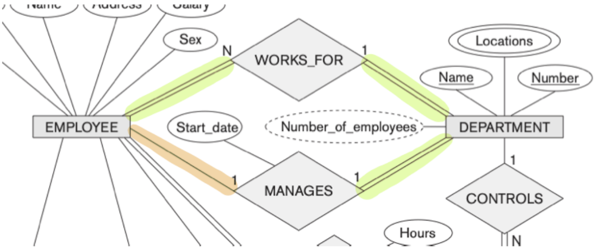


Figure 3.13
An M:N relationship,
WORKS_ON.

Relationship Types, Sets and Instance - Constraints

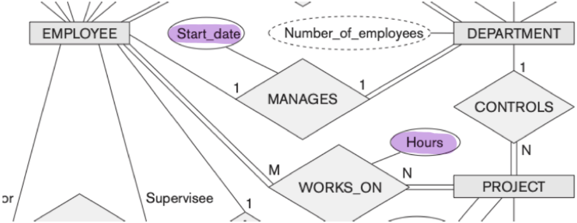
Participation

- Specifies whether an entity depends on being related.
- Participation constraints can be:
  - *Total* - every entity in one set must be related to another set via a relationship. For example, WORKS_FOR. Drawn as a double line on the ER-diagram.
  - *Partial* - some of the entities in one set are related to the entities in the other set. For example, MANAGES. Drawn as a single line on the ER-diagram.



Attributes of relationship types

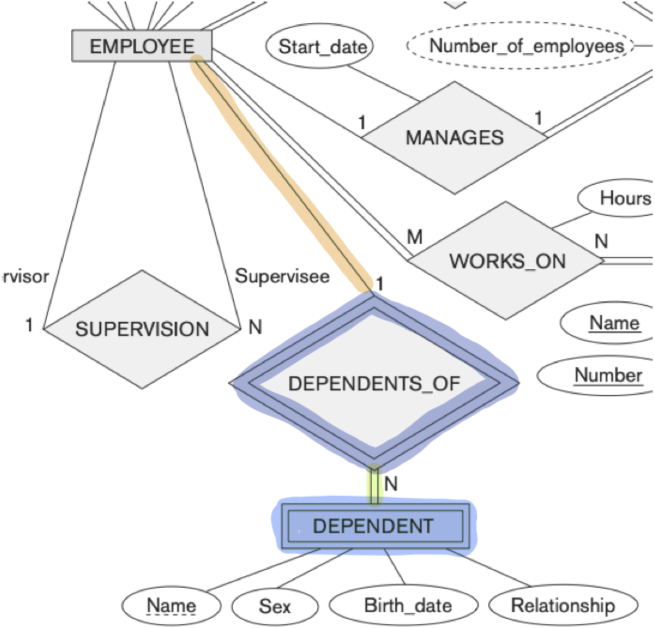
- Similar to entity types having relationships, relationship type may also have attributes.
- For example:
  - In the WORKS_ON relationship, the hours an employee works is modelled on the relationship.
  - Rather than storing the start date that an employee began managing a department in the DEPARTMENT or EMPLOYEE entity types, the attribute is captured as part of the MANAGES relationship type.



Weak entity types

- Entity types that do not have keys of their own are referred to as *weak entity types*. *Strong entity types* can therefore be seen as having keys.
- Entities of a weak entity type are related to specific entities of another entity type - referred to as the *identifying or owner entity type*.
- The *identifying relationship* (drawn with double lines) is the relationship between the owner of the weak entity type and the weak entity type. The owner of a weak entity type may itself be a weak entity type.
- A weak entity type has a *total participation constraint* between it and the owner entity. The weak entity type cannot exist without the owner.
- A weak entity type has a *partial key*. It can uniquely identify the weak entity related to the same owner entity.
- Weak entity types and their relationships are identified in the ER-diagrams as rectangles and diamonds with double boarders.

Weak entity types



Representation in an ER-diagram - Know your model

Chen's conceptual notation

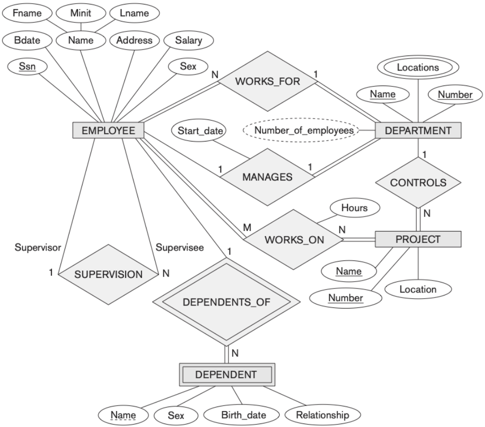
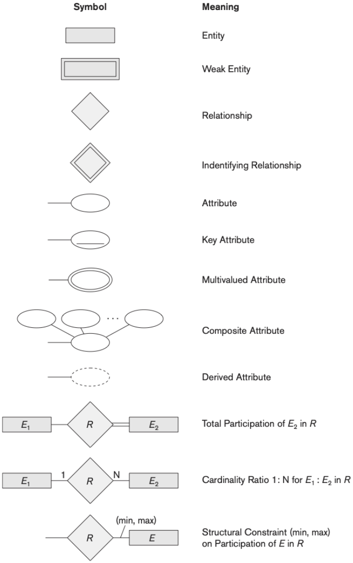


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.

ER-diagram - Notation summary



Symbol	Meaning
	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

Figure 3.14
Summary of the notation for ER diagrams.

Alternative notations

(min,max) notation

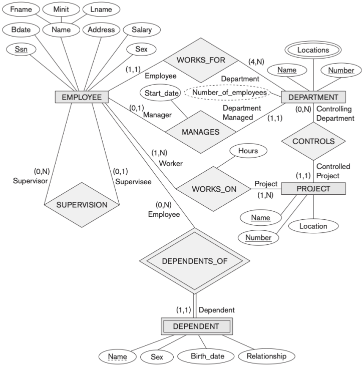


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

UML notation

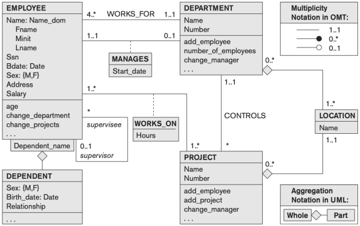


Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.

Alternative notations

Crow's foot notation

FIGURE 4.10 A WEAK ENTITY IN AN ERD

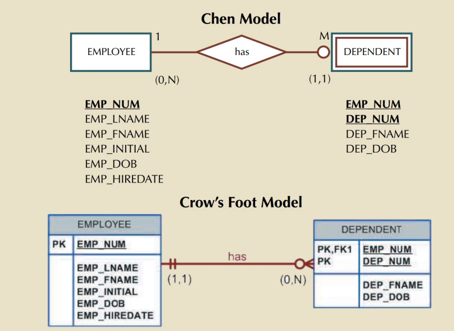
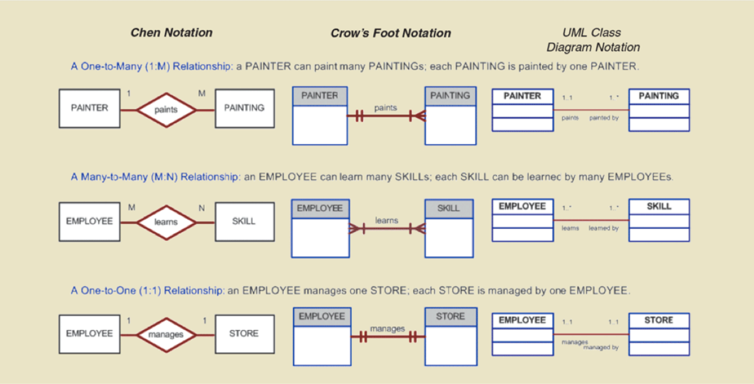


Image taken from: `https://frankfu.click/database/database-system-design-management/chapter4-entity-relationship-modeling/`

`https://vertabelo.com/blog/crow-s-foot-notation/`

`https://www.codeproject.com/Articles/878359/Data-Modelling-using-ERD-with-Crow-Foot-Notation`

Notations



A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.

A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.

A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.

Image taken from: `http://www.csc.villanova.edu/~enwafor/database/lectures/lecture_3.pdf`