Database Systems

Entity Relationship Model

Prof. Dr. Agnès Voisard Muhammed-Ugur Karagülle

Institute of Computer Science, Databases and Information Systems Group

Fraunhofer FOKUS

2025

Notes

Notes

Outline

1 **Requirements collection and analysis**
Database designers *interview* prospective database users to understand their *data requirements*. In parallel, *specify the known functional requirements* of the application, i.e., user-defined operations (or transactions) applied to a database, including retrieval and updates.
Techniques such as *data flow diagrams*.

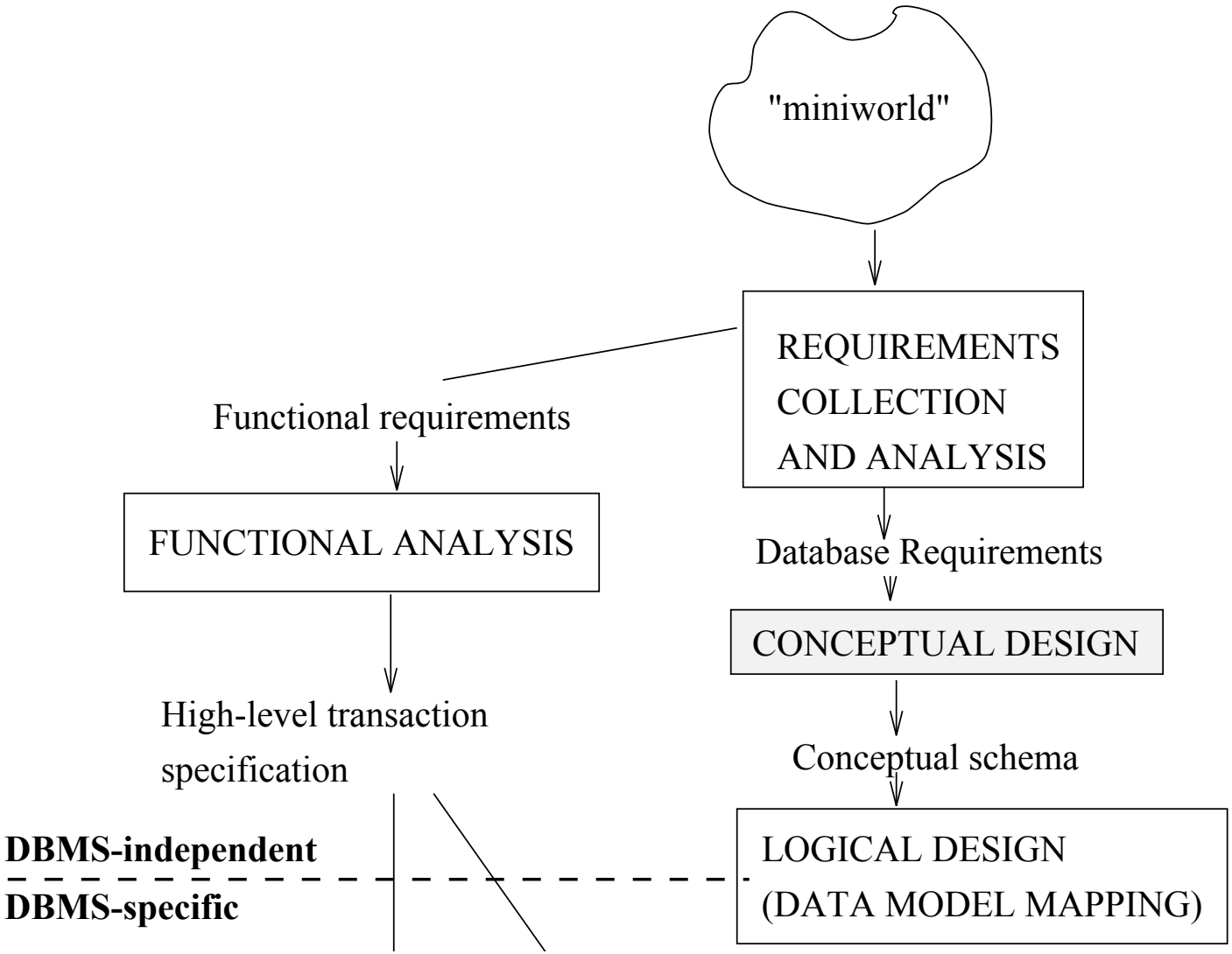
2 **Conceptual database design**
Create a *conceptual schema* using a high-level conceptual model. Description of the data requirements of the users: Detailed descriptions of data types, relationships, constraints.

3 **Specify high-level transactions**
Using basic data model operations. Transactions correspond to user-defined operations (of the functional analysis).

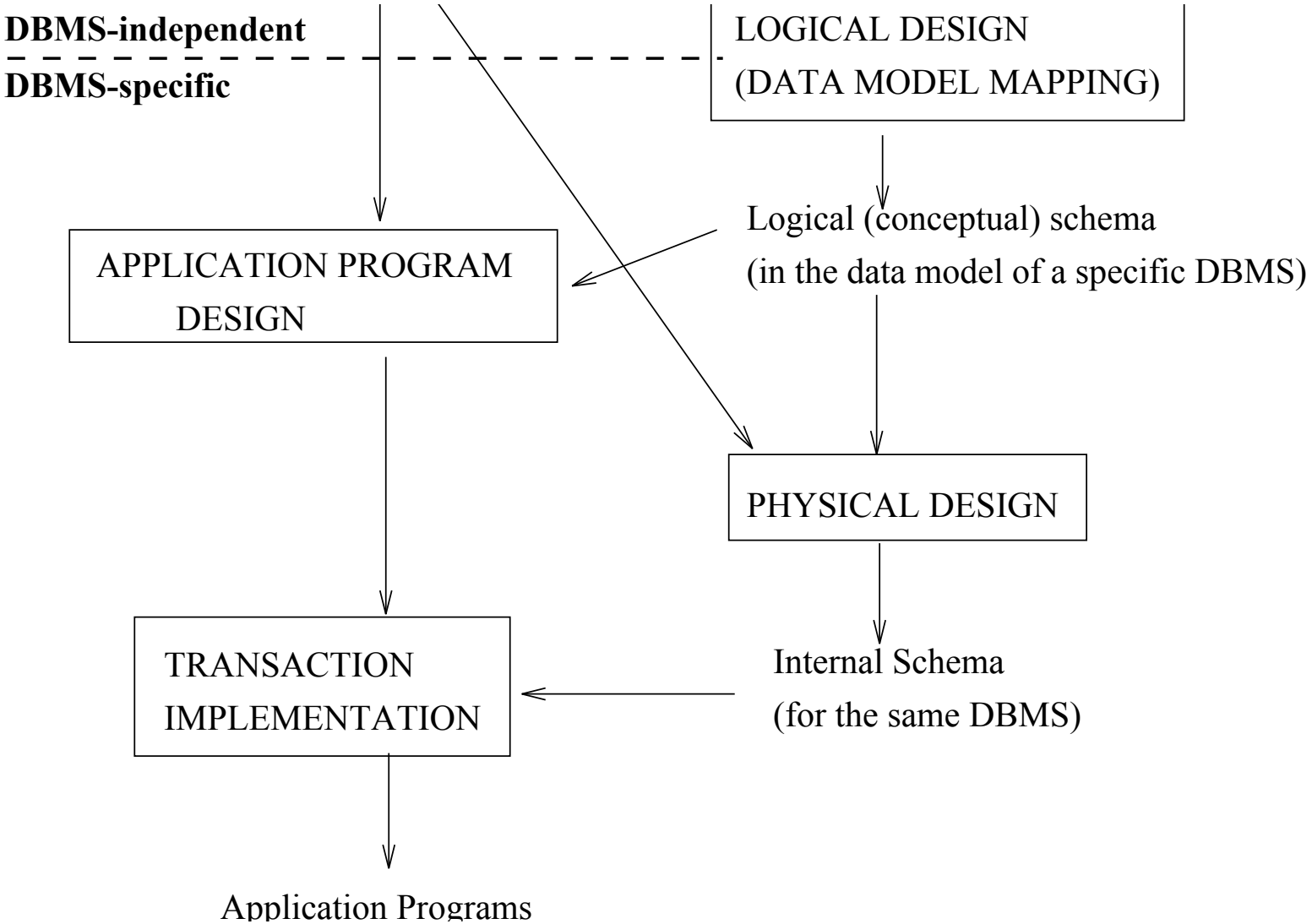
4 **Implementation of the DB using a DBMS**
Conceptual schema transformed from the high-level data model into the implementation data model.
Logical database design or **data mapping**.

5 **Physical design phase**
Internal storage structures and **file organizations** for the database are specified.
In parallel: Application programs designed and implemented as database transactions.

Phases of Database Design (cont'd)



Phases of Database Design (cont'd)



The Entity Relationship (ER) Model

- A precise modeling language with well-defined concepts
- Introduced in 1976 by C.C. Chen
- First step before implementation
- Active research field from the end of the 70's - early 80's
- Many variations and extensions
- Basis for "semantic data model"

Basic Concepts of the ER Model

- Basic object: **Entity**
  Distinguishable “thing” in the real world.
  E.g., car, person, house but also company, work, or loan.
- Properties of each entity: **Attributes**
  E.g., employees described by their name, address, salary, or position.
- A particular entity has a **value** for each of its attributes.

Example:

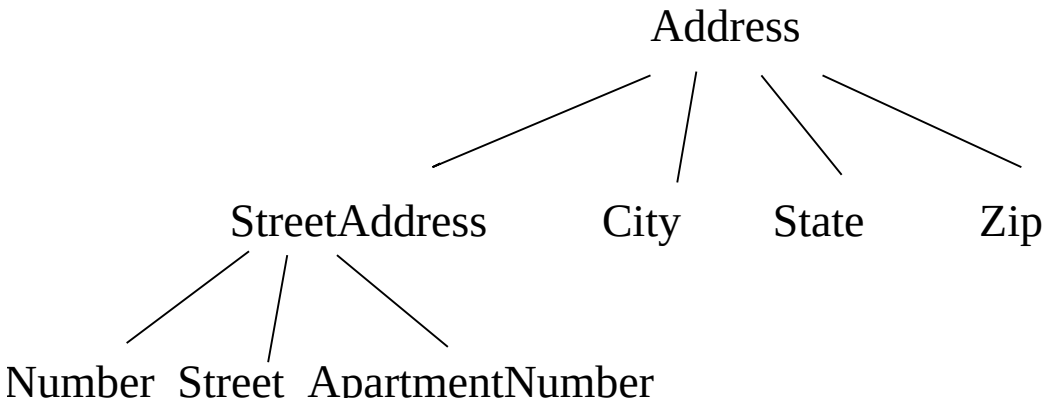
- Entity e1 (EMPLOYEE):
  Name = John Doe
  Address = 2208 Hyde St., San Francisco, CA 94109
  Age = 28
  HomePhone = 415 440 22 08

- Entity c1 (COMPANY):
  Name = TransCom
  Headquarters = Berkeley
  President = John Doe

Types of Attributes

1 Simple/composite

- **Simple** or atomic attributes
  Not divisible
- **Composite** attributes
  Can be divided into smaller subparts. Can form a hierarchy.



The whole address can be used as a single attribute
if there is no need to access the zip code, etc.

2 Single-value/multivalued

- **Single-valued** attribute
  A single value for a particular entity.

Example:

Age is a single-valued attribute of person.

Types of Attributes (cont'd)

- **Multivalued** attributes
  Set of values for the same entity.

Example:

CollegeDegrees attribute of person. A person can have 0, 1, or n degrees: **Different persons have different** *number of values* **for CollegeDegrees**.

- Attributes can have lower/upper bounds on the number of values for an individual entity.

Example:

Colors attribute of a car may have between 1 and 5 values (if a car has at most 5 colors).

Types of Attributes (cont'd)

3 Stored/derived

- **Stored** attribute

Example:

Birthdate

- **Derived** attribute
  Determined by other attributes.

Example:

Age derivable from the Birthdate attribute (and the current date).

Types of Attributes (cont'd)

Example:

NumberOfEmployees of a company obtained by counting all the employees of the company.

Attribute values derived from *related entities*.

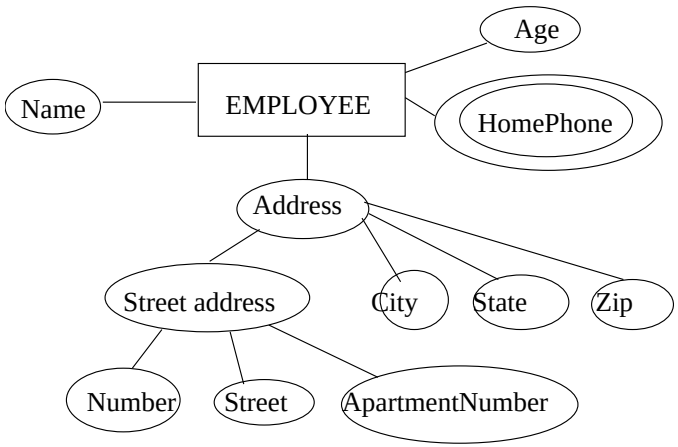
4 **Null values** Particular entity does not have a given attribute value. Ex.: ApartmentNumber not applicable for a house. Special value called null is created. Similar to unknown values.

Entity Types

- Collection of entities that have the same attributes
- Each entity has its own values
- Entity type describes the **schema** or **intension** for a set of entities.
- Individual entities of a particular entity type grouped into a **collection** or **entity set** (also called the **extension** of the entity type).

Emphasis on schemas rather than instances

- Entity type: Rectangular box enclosing the entity type name
- Attributes names: In ovals; attached to their entity types by straight lines
- Composite attributes: Attached to their components by straight lines
- Multivalued attributes: In double ovals



Key Attributes of an Entity Type

- Usually an entity has an attribute whose values are distinct for each individual entity: **Key attribute**
- **Values of key attribute used to identify each entity uniquely**

Example:

Name attribute: Key for the COMPANY entity type.
For a person: Usually a SocialSecurityNumber.

- Sometimes *several attributes* together form a key and are grouped into a *composite attribute*

When defining a key

- **Uniqueness property must hold for every extension**
- Some entity types have more than one key attribute

Example:

Student with name and student ID (MN).

- In ER diagrams, key attribute are underlined.

Domains of Attributes

- With each simple attribute of an entity type is associated a **value set** or **domain**

  ⇒ The **domain** specifies the set of values that can be assigned to an attribute.

Example:

Range of ages for employees between 16 and 70:
Values of Age of EMPLOYEE = set of integers between 16 and 70.

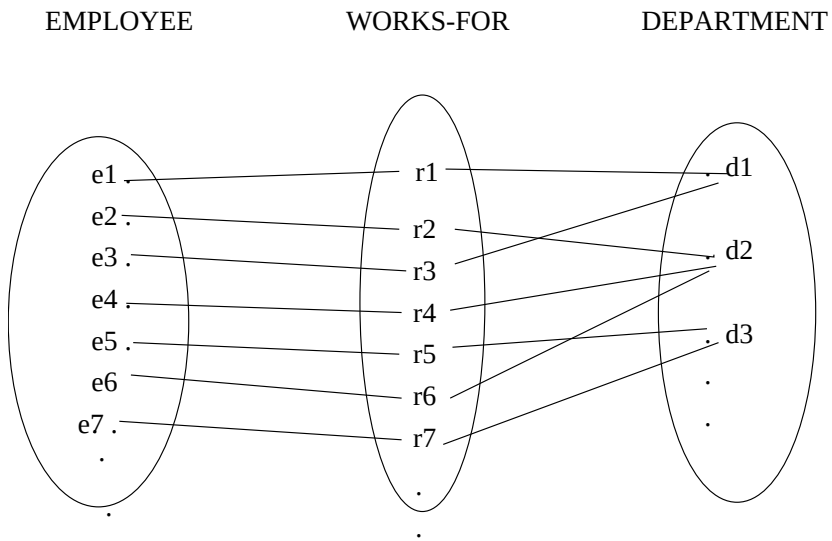
- Domains are not displayed in ER diagrams

Relationship

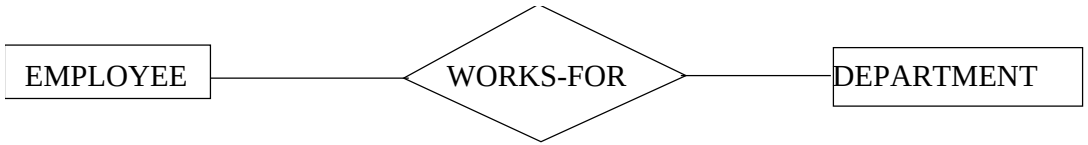
- A **relationship type** R
  among n entity types E_1, E_2, …, E_n
  defines a set of associations among entities of these types.
- R is a set of **relationship instances** r_i,
  where each r_i associates n entities (e1, e2, ..., en) and
  each entity e_j in r_i is a member of entity type E_j, $1 \leq j \leq n$.
  A relationship is a mathematical relation on
  E_1, E_2, …, E_n.
  Each of the entity types E_1, E_2, …, E_n **participates** in the relationship type R, and
  each of the individual entities e_1, e_2, …, e_n
  participates in the relationship instance $r_i = (e_1, e_2, \ldots, e_n)$.

Example Relationship

- Relationship type WORKS-FOR between 2 entity types EMPLOYEE and DEPARTMENT associates each employee with the department s/he works for.



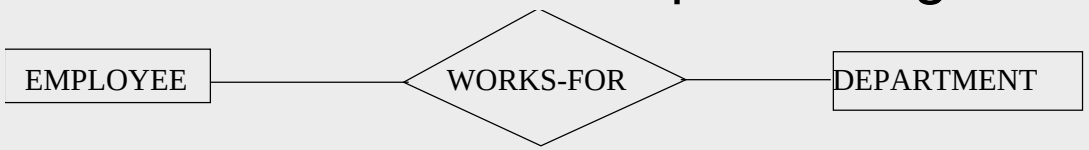
- In ER diagrams, relationship types displayed as diamond-shaped boxes (connected by straight lines to the entities).



- **Degree of a relationship type**: Number of participating entity types.

Example:

WORKS-FOR relationship is of degree 2 (binary).



Roles

- **Each entity plays a certain role in a relationship**

Example:

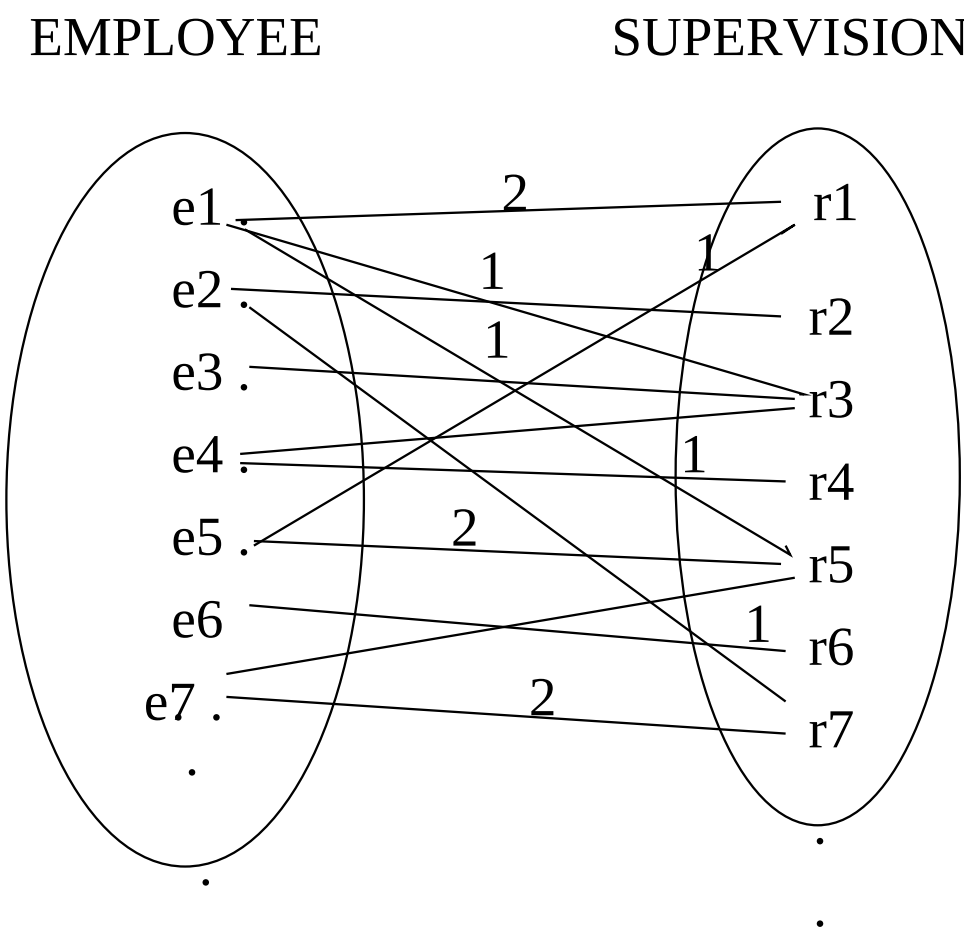
In the WORKS-FOR relationship type, EMPLOYEE plays the role of employee or worker, and DEPARTMENT plays the role of employer.

- **Recursive** relationships
  Sometimes the same entity type participates more than once in a relationship type in different *roles*

Example:

SUPERVISION relationship type relates an EMPLOYEE to a SUPERVISOR, where both employee and supervisor entities are members of the same EMPLOYEE entity type.

Roles (cont'd)



1: spervisor role

2: supervisee role

Constraints on Relationship Types

Constraints on relationship types determined by the “miniworld”.

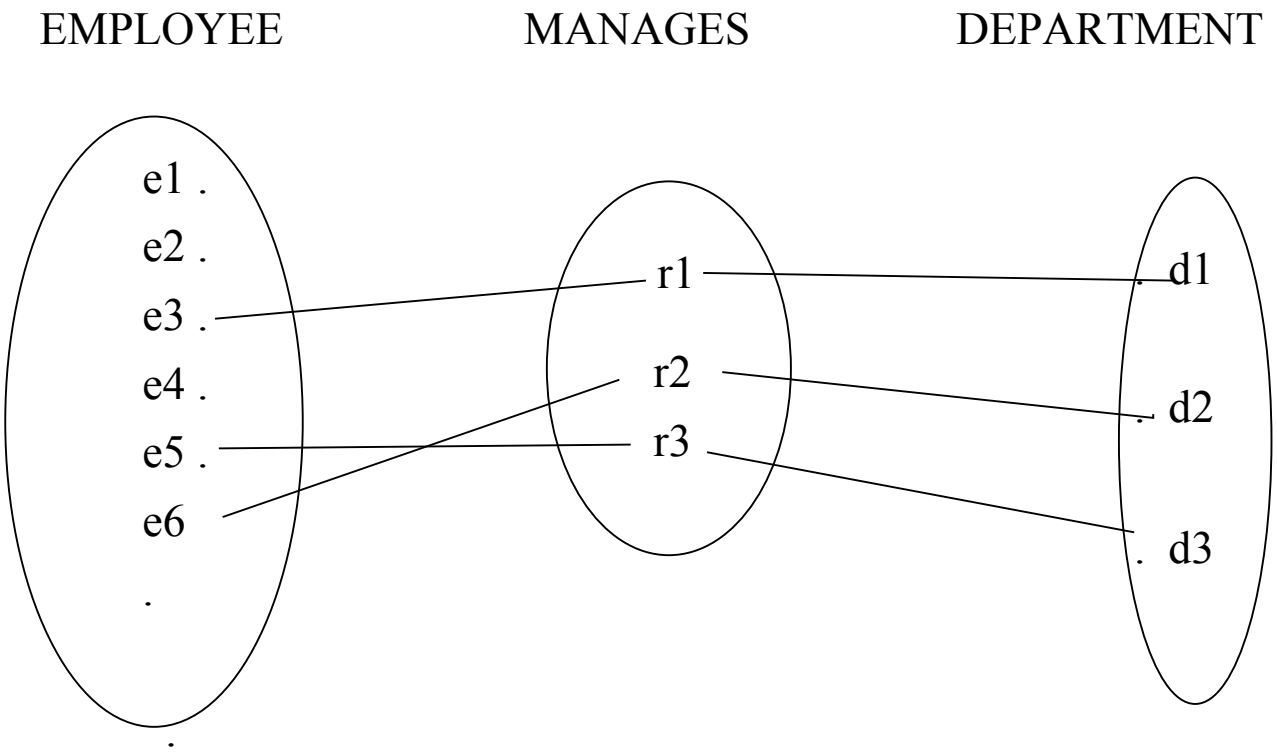
Example:

In a company, each employee must work for exactly one department.

2 main types of relationship constraints:

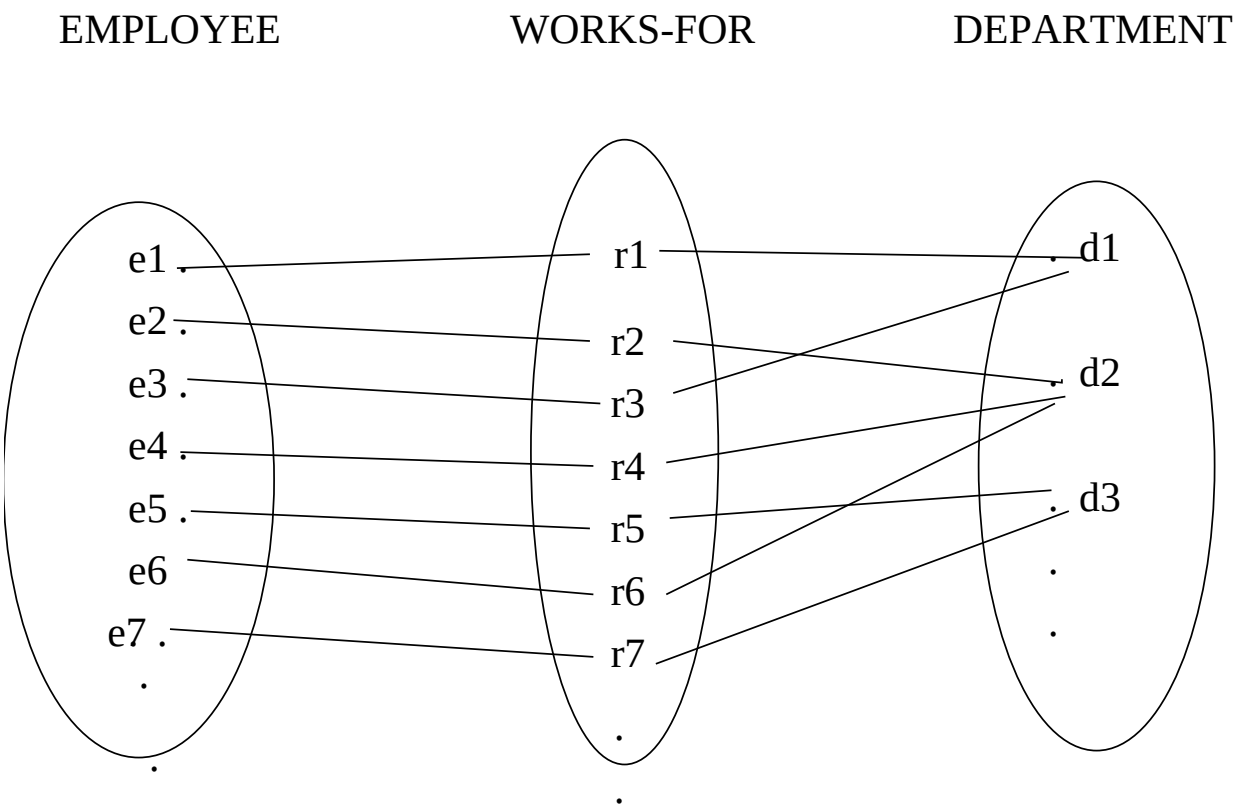
1. **Cardinality ratio** Specifies the number of relationship instances that an entity can participate in.
   - **1:1** (*one-to-one*). MANAGES between a DEPARTMENT entity and an EMPLOYEE (manager) is of cardinality ratio **1:1**. An employee can manage only *one* department and a department has only *one* manager.
   - **1:N** (*one-many)* WORKS-FOR between DEPARTMENT and EMPLOYEE is **1:N**. An employee works for one department and each department can be related to numerous employees.
   - **M:N** (*many-many)* WORKS-ON between EMPLOYEE and PROJECT is **M:N** if an employee works on several projects and several employees can work on a project.

- **1:1** relationship manages

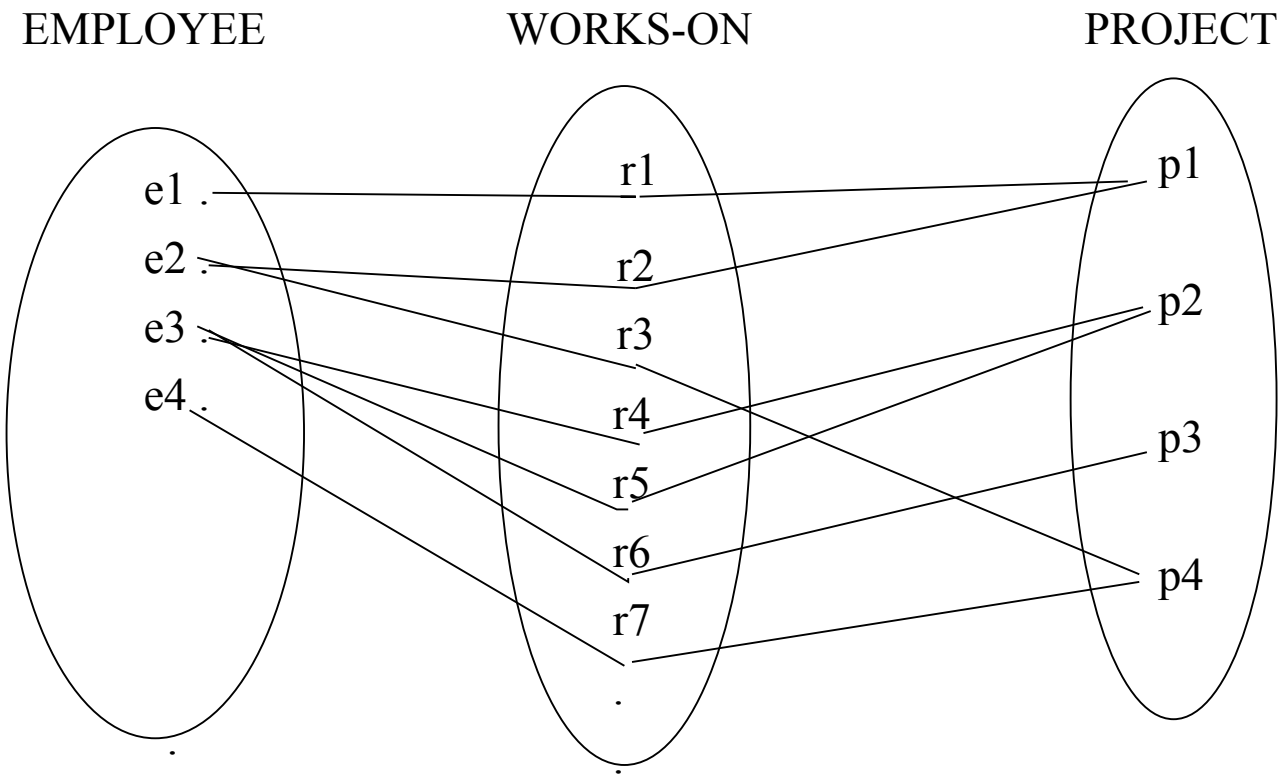


Graphic Examples (cont'd)

- **1:N** relationship works-for between department and employee.



- **M:N** relationship works-on.

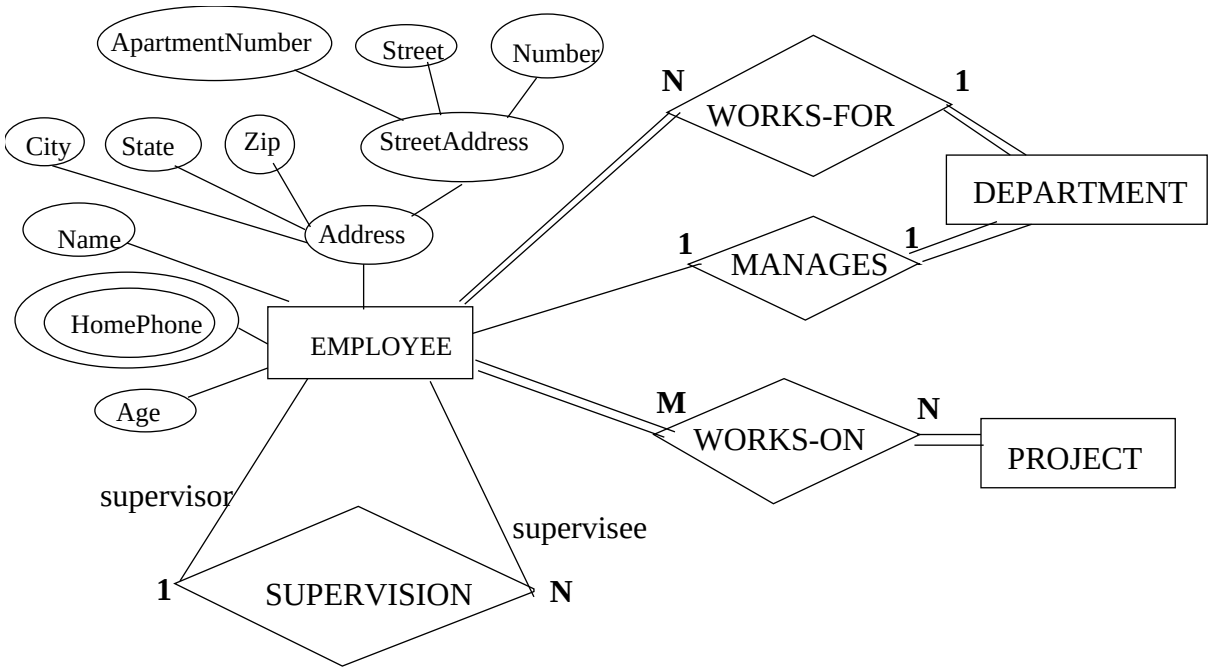


2. Participation constraint

Specifies whether the existence of an entity depends on its participation in a relationship with another entity.

- **Total** *Every employee must work for a department.*
  EMPLOYEE exists only if it participates in the WORKS-FOR relationship instance.
  Total participation of EMPLOYEE in WORKS-FOR:
  Every entity of in the total set of employees must be related to a department entity via *WORKS-FOR*.
  Also called **existence dependencies**.
- **Partial** *Not every employee manages a department*.
  EMPLOYEE in the MANAGES relationship is **partial**:
  Some employees are related to a department entity via the relationship MANAGES, but not necessarily all employees.

On ER-diagrams:
1, M, N are displayed.
Total participation: Double line between the participating entity and the relationship.
Partial: Single line.



Cf. attributes for entities

Example:

Date on which a manager started managing a department: attribute “StartDate” for the MANAGES relationship type.

- *Attributes of 1:1 relationship types* can be migrated to one of the participating entities.

Example:

StartDate can be an attribute of EMPLOYEE or DEPARTMENT.

Attributes of a Relationship Type (cont'd)

- *Attributes of 1:N relationship types* can be migrated to the entity type at the N side (only)

Example:

StartDate in the relationship WORKS-FOR between EMPLOYEE and DEPARTMENT can only be affected to EMPLOYEE.
(Each EMPLOYEE entity participates in at most one relationship instance).

Where to place a relationship attribute?
Determined subjectively by the schema designer.

Conceptual schema: least possible redundancy

Weak entity types

Some entities without any key attribute: **Weak entity types**.

Example:

DEPENDENT related to EMPLOYEE, keeps track of the dependent of each employee via a 1:N relationship.

2 dependents of *distinct employees* can have the same value for Name, Address, BirthDate and Relationships but they are still distinct.
Each EMPLOYEE entity **owns** the dependent entities related to it.

Entity type EMPLOYEE: **identifying owner**, Relationship type DEPENDENTS-OF: **identifying relationship**.
Usually has a **partial key**: set of attributes that identifies weak entities related to the *same owner entity*.

Back to the *company* example

- Company organized into departments. Each department has a unique name and a particular employee who manages it since a certain date.
  A DEPARTMENT may have several locations.
- Each employee's name, social security number, address, salary, and birthdate is stored.
  An employee is assigned to one DEPARTMENT but can work on many projects which are not necessarily controlled by the same DEPARTMENT.
  We keep track of (i) the number of hours/week an EMPLOYEE spends on each PROJECT and (ii) the direct supervisor of each employee.

Back to the *company* example (cont'd)

- A department controls a number of projects.
  A PROJECT has a unique name and a single location.
- Dependents of each employees are stored
  (name, birthdate, address, relationship).

Entities & Relationships in the *company* example (cont'd)

Entity types

- EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

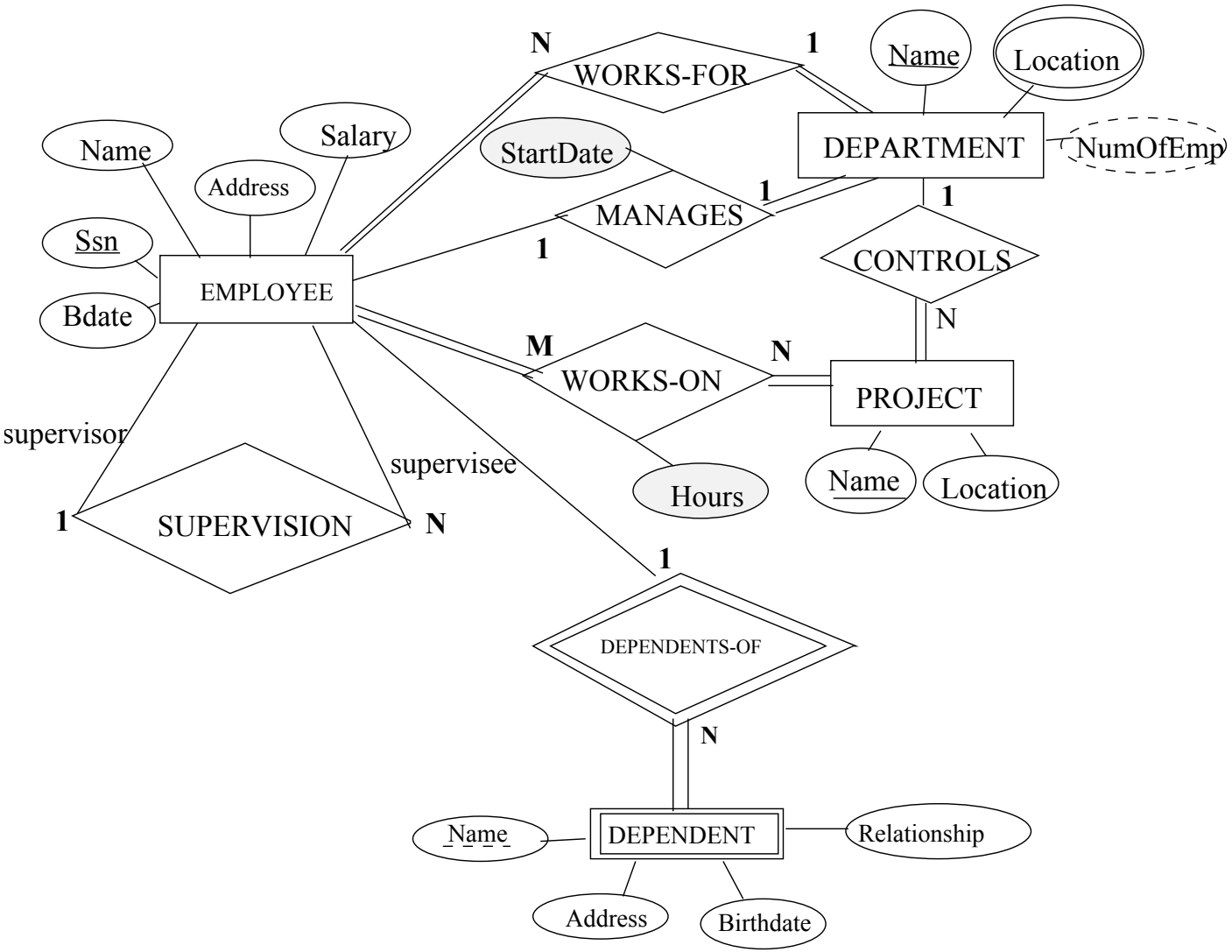
Relationship types

- MANAGES, 1:1 relationship between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. Attribute StartDate assigned to MANAGES.
- WORKS-FOR, 1:N relationship between DEPARTMENT and EMPLOYEE. Both total participation.

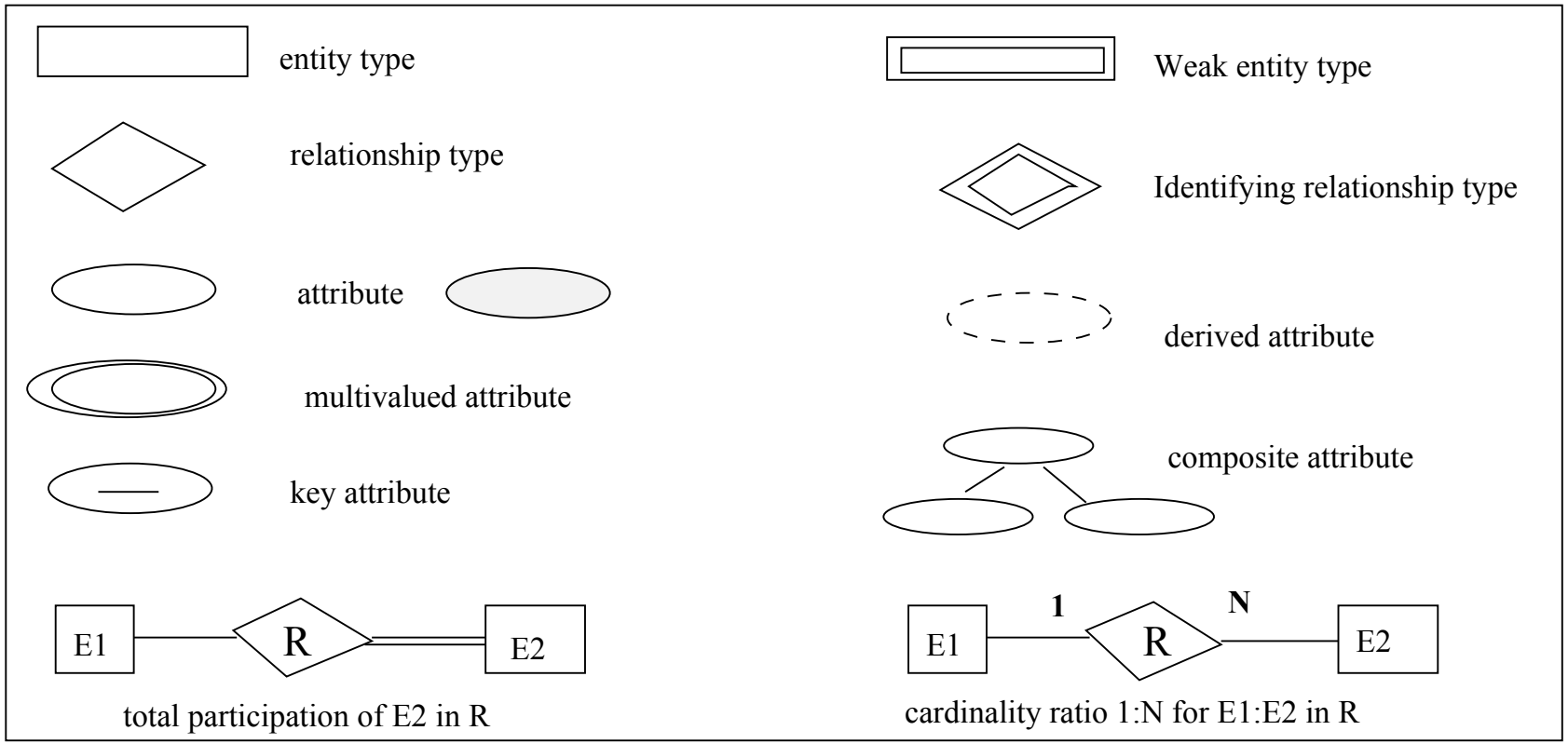
Entities & Relationships in the *company* example (cont'd)

- CONTROLS, 1:N between DEPARTMENT and PROJECT. Participation of PROJECT is total.
- SUPERVISION, 1:N between EMPLOYEE (SUPERVISOR) and EMPLOYEE (supervisee). Both partial: not every employee is a supervisor and not every employee has a supervisor.
- WORKS-ON, M:N between EMPLOYEE and PROJECT. With attribute Hours. Both total participation.
- DEPENDENTS OF, 1:N between EMPLOYEE and DEPENDENT.

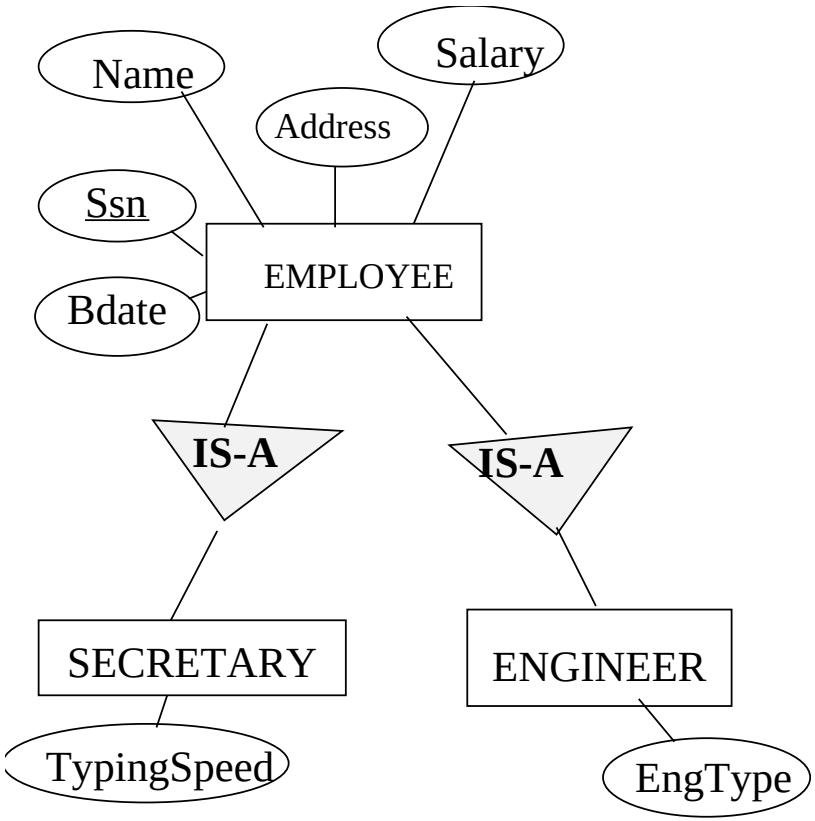
company database: complete diagram



company database: complete diagram (cont'd)



Generalization and specialization: **IS-A** relationship.



"Melany Kimm" is an EMPLOYEE and also an ENGINEER: specific role

Other ER diagram notation

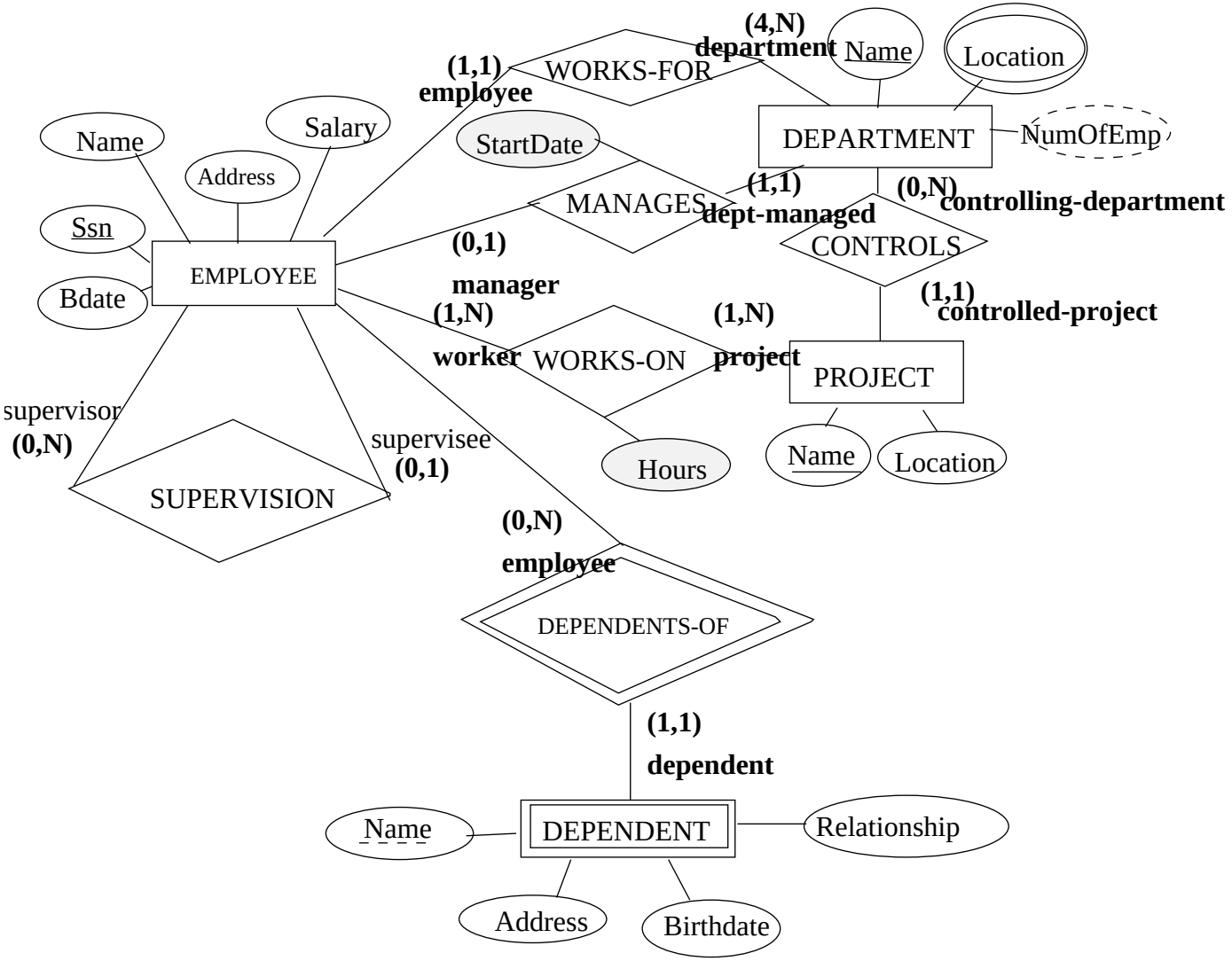
Cardinality + participation constraint sometimes noted with more precision than 1:N, etc + double line.

=> Pair of integers *(min,max)* with each participation of the entity type E in a relationship R, where $0 \leq min \leq max$ and $max \geq 1$.

Each entity e in E must participate in at least *min* and at most *max* relationships.

min = 0 implies partial participation,
$min > 0$ implies total participation.

company database: complete ER diagram



ER: high-level conceptual data model

- Types of attributes: simple/composite, single-valued/multivalued, derived/stored.
- Entity types, entity sets.
- Attributes: key attributes, domains of attributes.
- Relationship types and set of instances, role.
- Cardinality ratios (1:1, 1:N, M:N), participation constraints (total or partial).
- ER-diagrams and company database.

Questions?

What will come next?

1 Welcome to Database Systems
2 Introduction to Database Systems
3 Entity Relationship Design Diagram (ERM)
4 Relational Model
5 Relational Algebra
6 Structured Query Language (SQL)
7 Relational Database Design - Functional Dependencies
8 Relational Database Design - Normalization
9 Online Analytical Processing + Embedded SQL
10 Data Mining
11 Physical Representation - Storage and File Structure
12 Physical Representation - Indexing and Hashing
13 Transactions
14 Concurrency Control Techniques
15 Recovery Techniques
16 Query Processing and Optimization