DATABASDESIGN FÖR INGENJÖRER - 1056F

Sommar 2005

En introduktionskurs i databassystem

http://user.it.uu.se/~udbl/dbt-sommar05/
alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st05/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

UPPSALA UNIVERSITET

Introduction to Database Design Using Entity-Relationship Modeling

Elmasri/Navathe chs 3-4

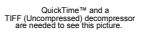
Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

UPPSALA UNIVERSITET

ER-modeling

- Aims at defining a high-level specification of the information content in the database.
- History
  - Chen, P. P. S., “The entity-relationship model: towards a unified view of data”, ACM TODS, 1, 1 1976, p. 9-36.



- Why ER-models?
  - High-level description - easier to understand for non-technicians
  - More formal than natural language - avoid misconceptions and multiple interpretations
  - Implementation independent (of DBMS) - less technical details
  - Documentation
  - Model transformation to an implementation data model

Entity type and entity

- An ***entity type*** represents a physical or abstract concept with some sort of identity. The individual instances of the concept are members of a set of ***entities*** that have the same set of attributes.
  - Entity types express the *intention,* i.e. the meaning of the concept whereas the set of entities represents the *extension* of that type.
  - Names of entity types are given in singular form.
  - The description of an entity type is called its *schema*.

    ```
    PERSON
        name, ssn, address, phoneno
    ```

  - Each attribute in an entity type is associated with a domain that indicates the allowed values of that attribute.

Attribute

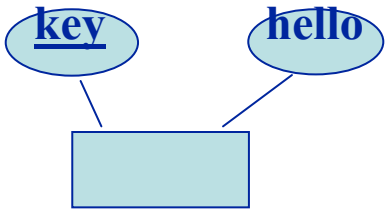
- An attribute is a characteristic or aspect that describe an entity (and is defined on entity types).
  - Every attribute has a *domain* (or *value set*).
  - A domain specifies the set of allowed *values* each individual attribute can be assigned.
  - There is (at least) six different types of values for attributes:
    - **simple/** sex: M *or* F
    - **composite** name: (Ior, Karlsson)
    - **single-valued/** name: "Ior Karlsson"
    - **multivalued** friends: {Nasse, Puh,...}
    - **stored/** birthdate: 980917
    - **derived** age : 0
    - **null**

Note!

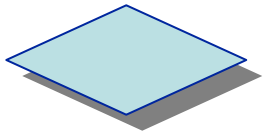
ER-diagram

Attribute cont...

- **Key**: an attribute that has unique values for every instance of an entity type is called a **key attribute**.
- Sometimes *several* attributes are used together to get a unique key.
- An entity type can have more than one key.



Relationship type and relationship



- A ***relationship type*** represents a relationship (or relation/connection), between a number of entity types.
- A relationship type R is a set of ***relationships*** (i.e. *relational instances*) or *tuples*.
- A relationship type, R, can mathematically be defined as:

  $R \subseteq E_1 \times E_2 \times \ldots \times E_n$

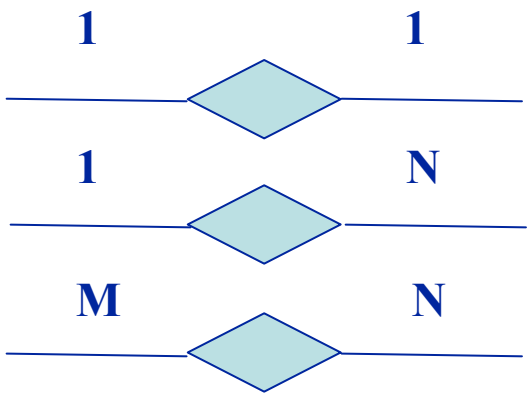
  where each Ej is a entity type.
- A tuple (or an instance) $t \in R$ is written as $(e_1, e_2, ..., e_n)$ or $\langle e_1, e_2, ..., e_n \rangle$ where $e_j \in E_j$.

Structural constraints for relationship types

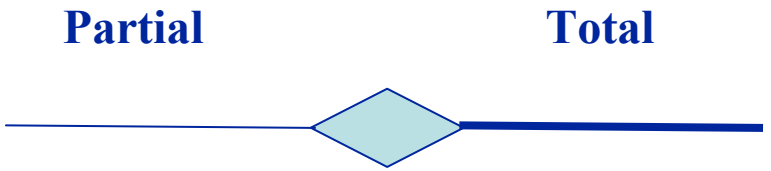
- **Cardinality ratio constraint** specifies the number of relational instances that an entity can take part in.

  For binary relationship types:

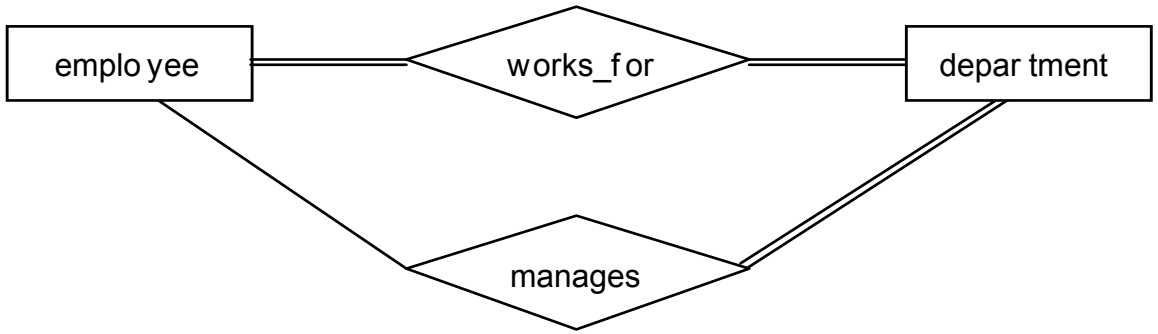
  – one-to-one (1:1)
  – one-to-many (1:N)
  – many-to-many (M:N)



Structural constraints cont. ...

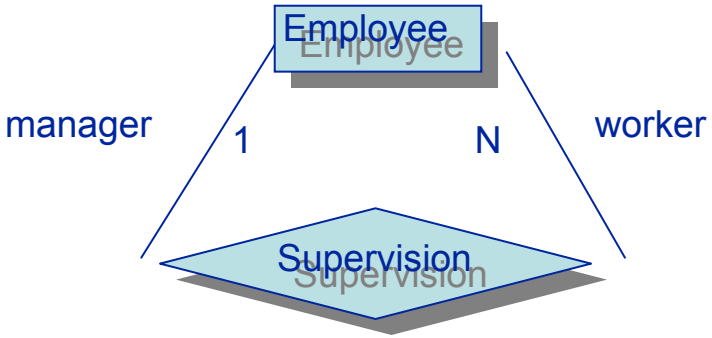


- **Participation constraint**
  - specifies whether the entity existence is dependent of another entity via a relationship type.
    E.g. can an employee exist without working for a department?
  - Partial participation: the entity can exist without this relationship
  - Total participation: the entity requires this relationship in order to exist.

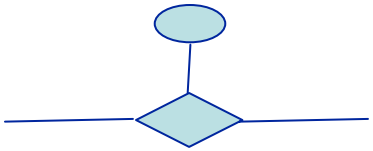


Roles of relationship types

- A role name specifies what **role** an entity type plays in a specific relationship
- Role names are sometimes used in ER-diagrams to clarify the roles of the participating entity types.



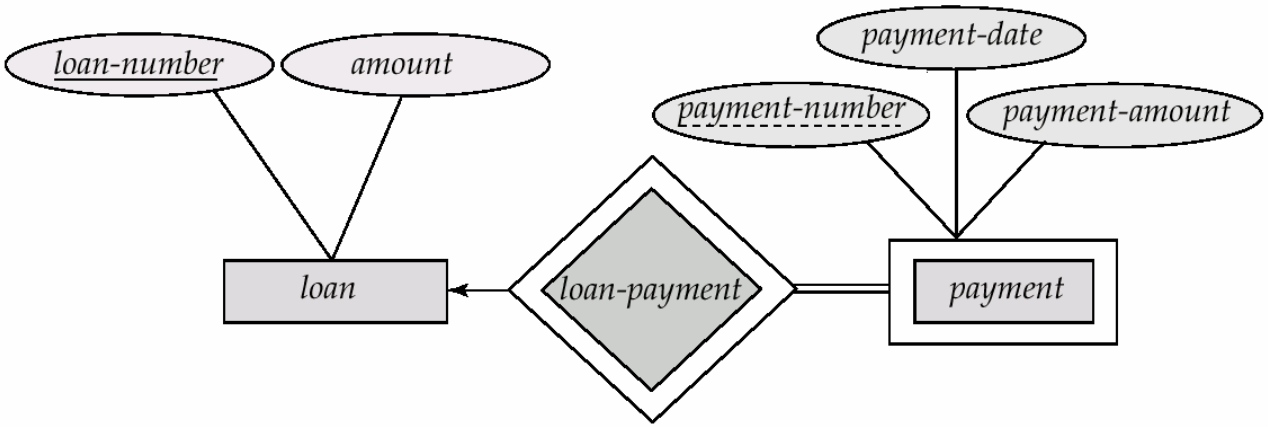
Attributes for relationship types



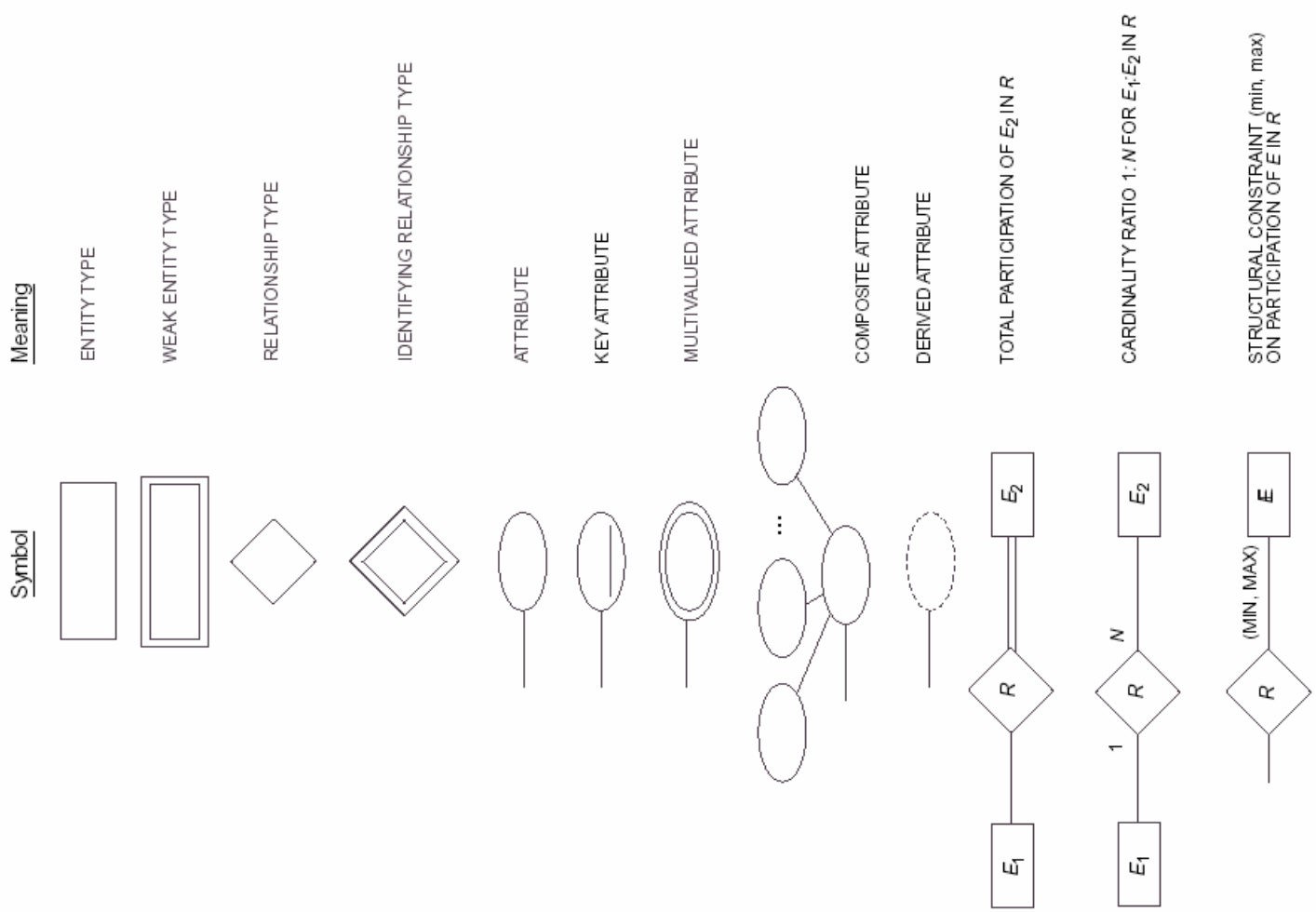
- Also a relationship type can have attributes. E.g. in the case where the weekly number of hours an employee works on a project should be kept, that can be represented for each instance of the relation “works-on”.
- If the relation is a 1:1 or 1:N relation, the attribute can be stored at one of the participating entities.
- When the relation is of the type M:N one must store the attributes with the instance of the relation.

Weak entity types

- **Weak entity types** are those that are meaningless without an owner entity type.
- Weak entities are uniquely identified in the extension with their owner's key attributes together with its own (broken) underlined attribute.
- The relationship to the owner is called the identifying *relationship.*

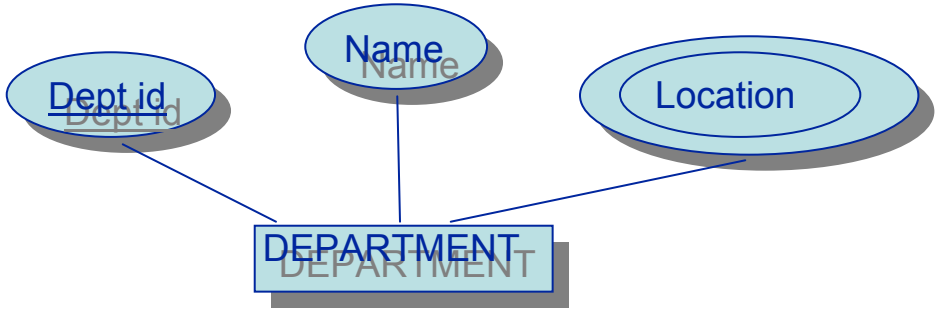


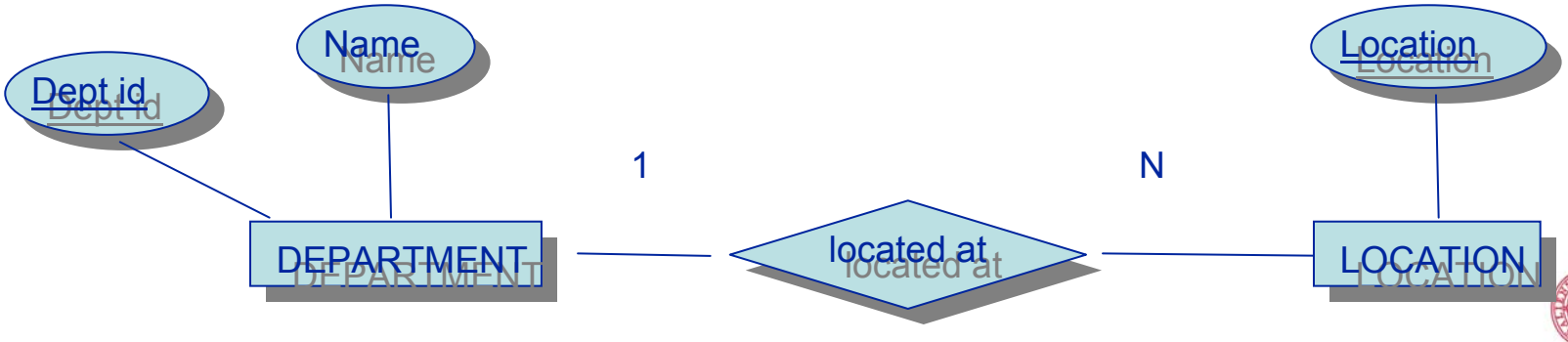
ER-notation (Elmasri/Navathe fig. 3.14)



ER model transformations

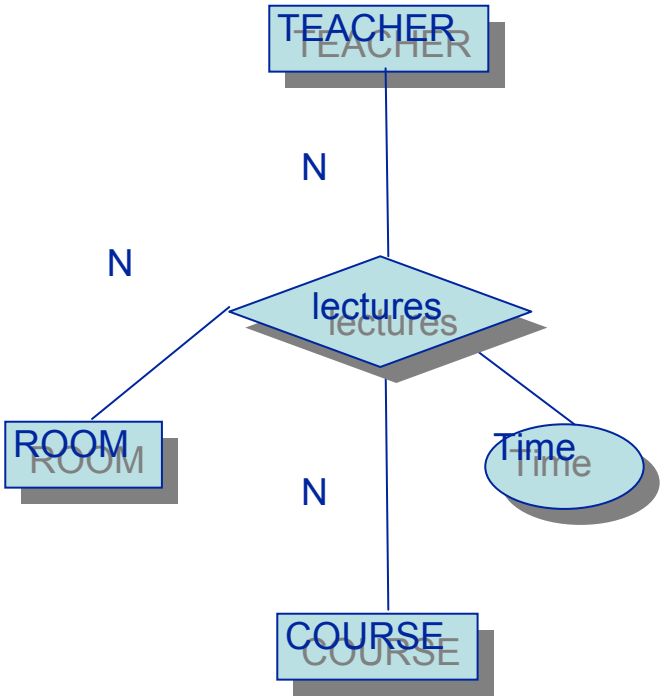
- Replacing multi-valued attributes by an entity type

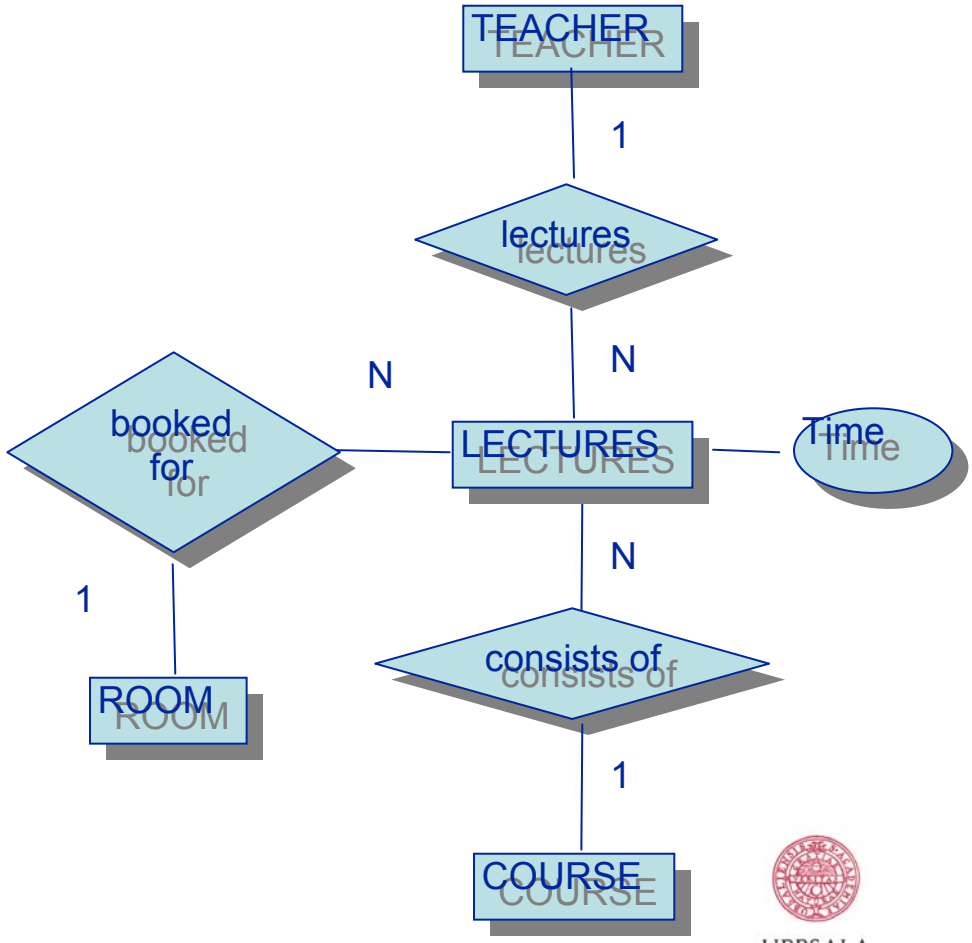




ER model transf. cont. ...

- Replacing M-N relationships with an entity type and binary relationships.

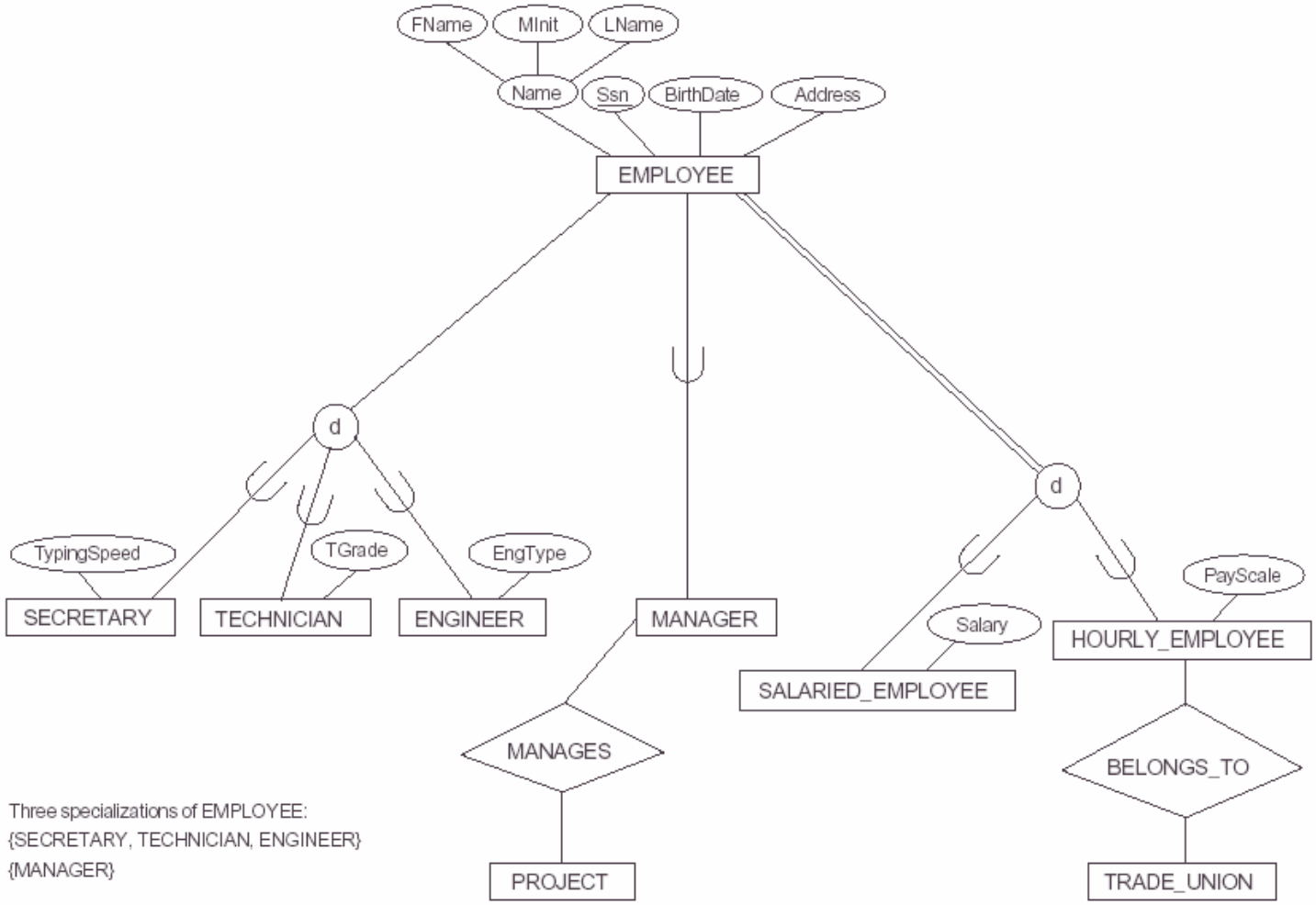




Extended Entity-Relationship (EER) modeling

- The intention of using an E-R diagram is to use it as a basis for user communication or for getting to a good design specification.
  - i.e. try to make it simple and avoid to much complexity.
- EER (extended or enhanced ER) introduces several notational extensions to deal concepts such as:
  - Superclass /subclass (supertype/subtype, is-a relationship)
    - specialization/generalization
    - constraints
  - Aggregation (whole/part or part-of relationship)
  - Union types (category)

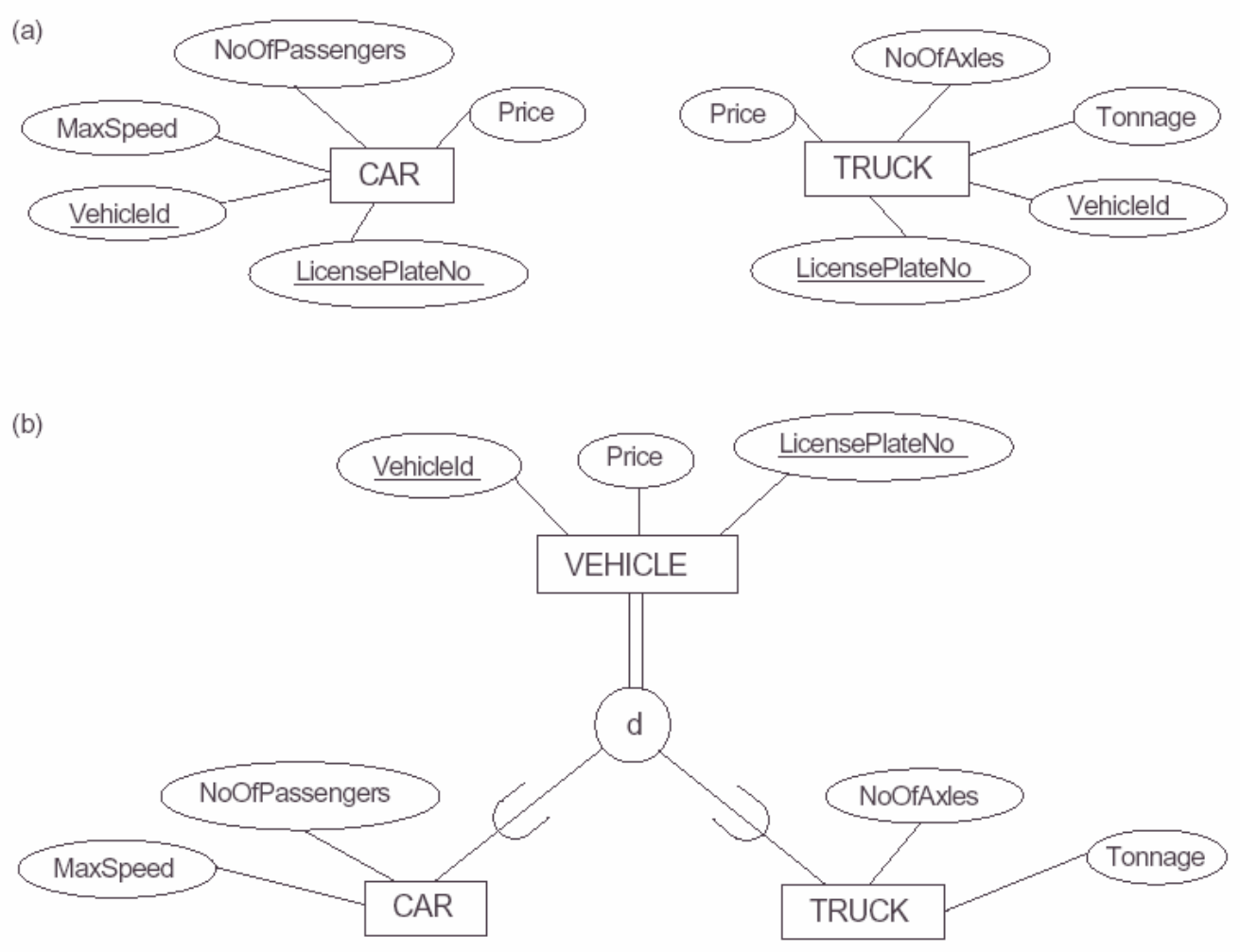
EER diagram notation for specialization and subclass (Elmasri/Navathe fig. 4.1)



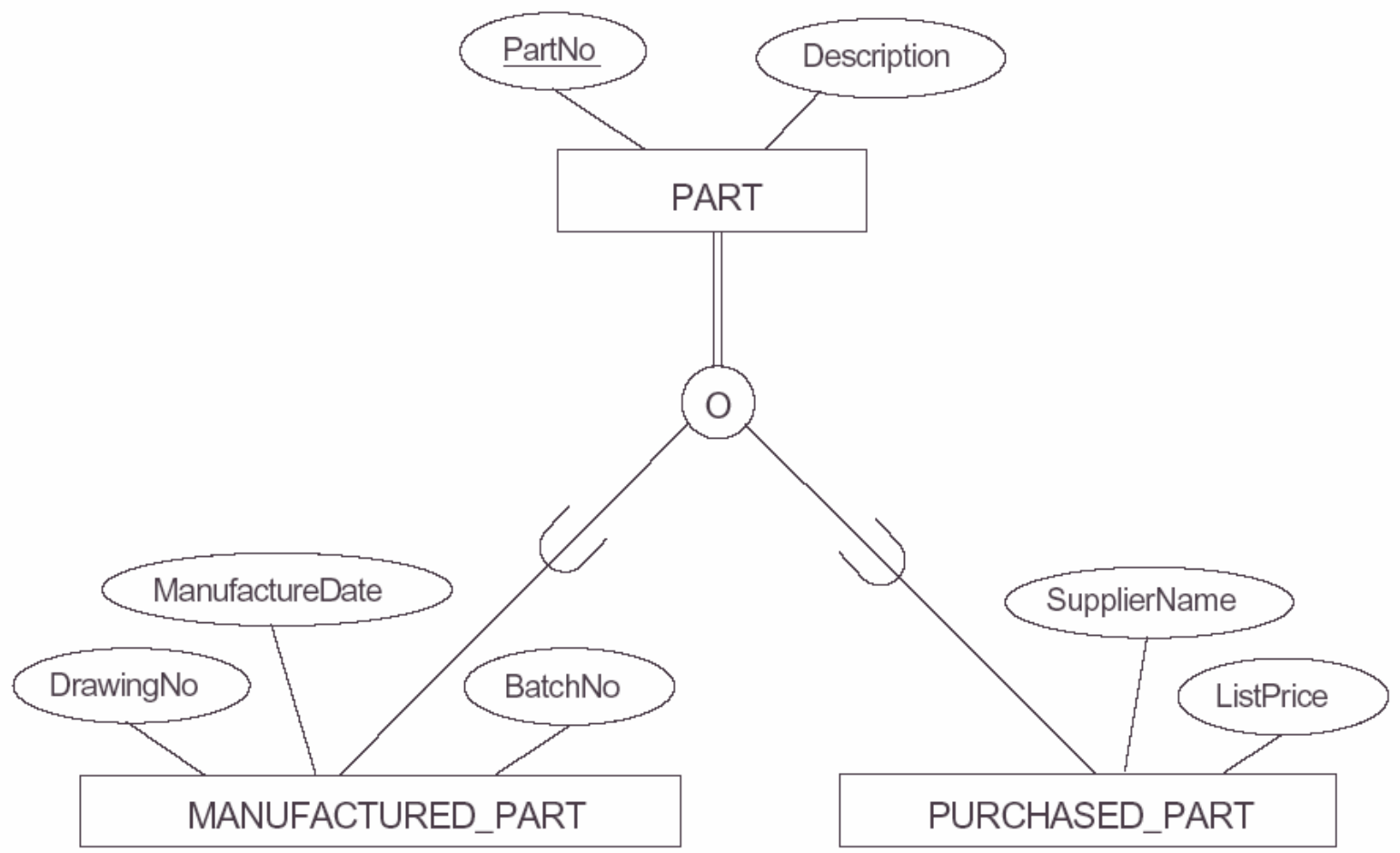
Subclasses, superclasses & inheritance

- Two generic ideas for creating superclass/subclass relationships
  - Specialization of superclass into subclasses
  - Generalization of subclasses into a superclass
- Constraints and characteristics of spec. & gen.
  - Constraints
    - Predicate-defined (condition-defined) sub-classes
    - Attribute-defined
    - User-defined
  - Disjointness
    - Disjoint
    - Overlapping
  - Completeness
    - Total
    - Partial

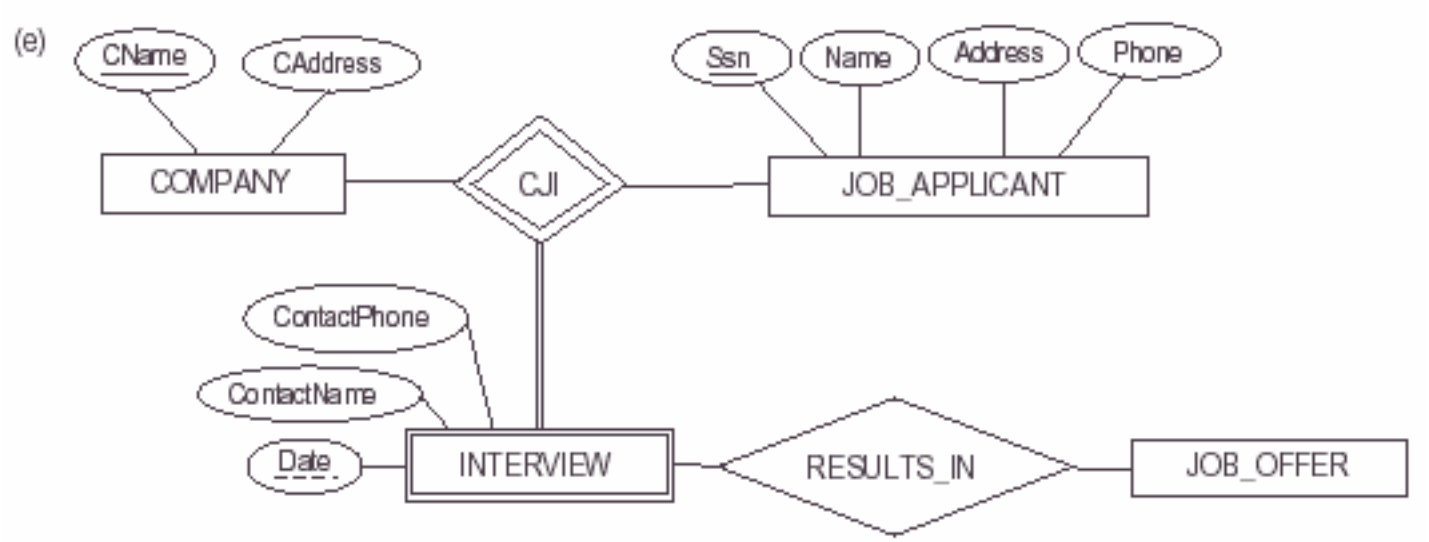
Generalization of subclasses (Elmasri/Navathe fig. 4.3)



Overlapping (nondisjoint) subclasses (Elmasri/Navathe fig. 4.5)

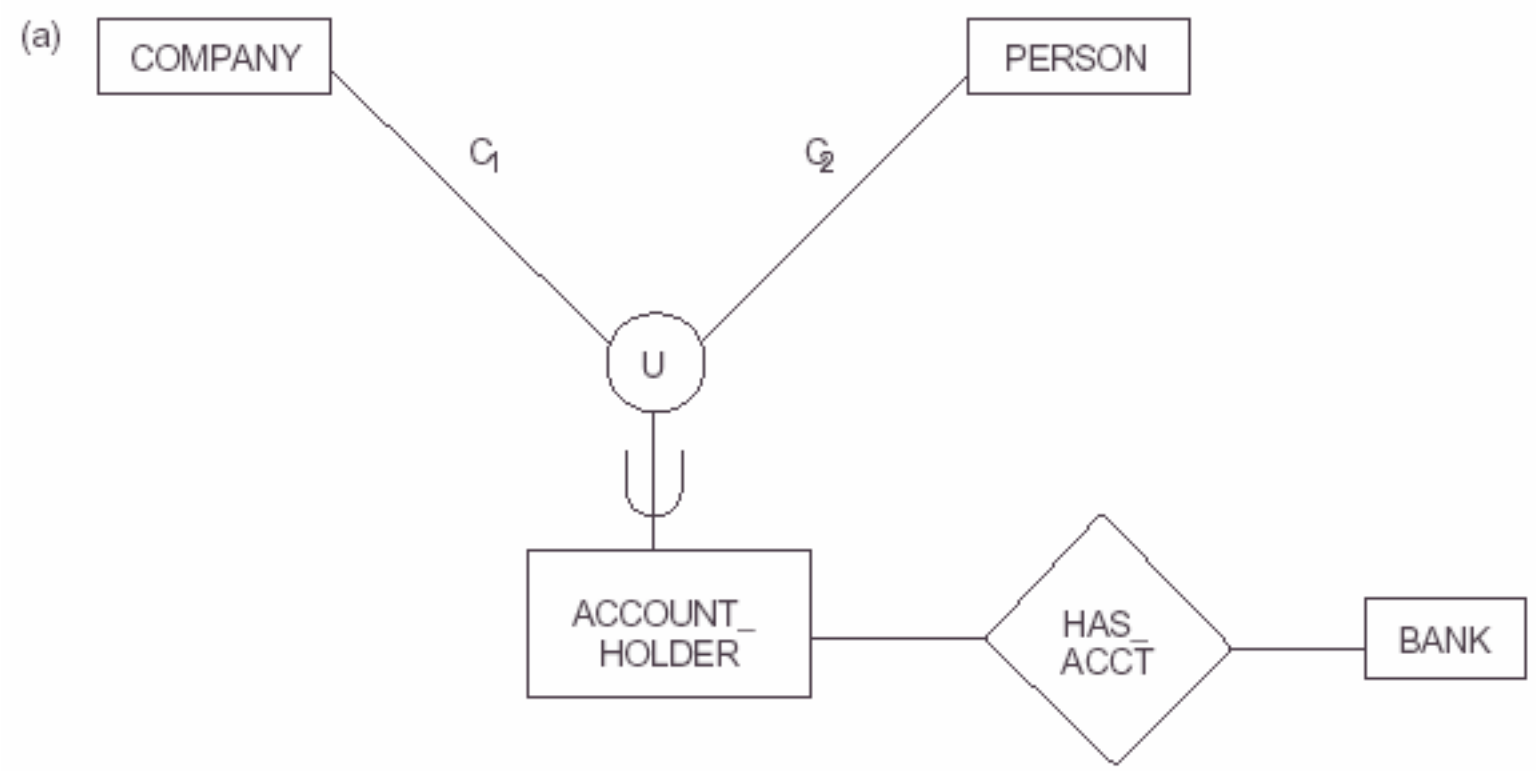


Representation of aggregation in ER notation (Elmasri/Navathe fig. 4.16e)



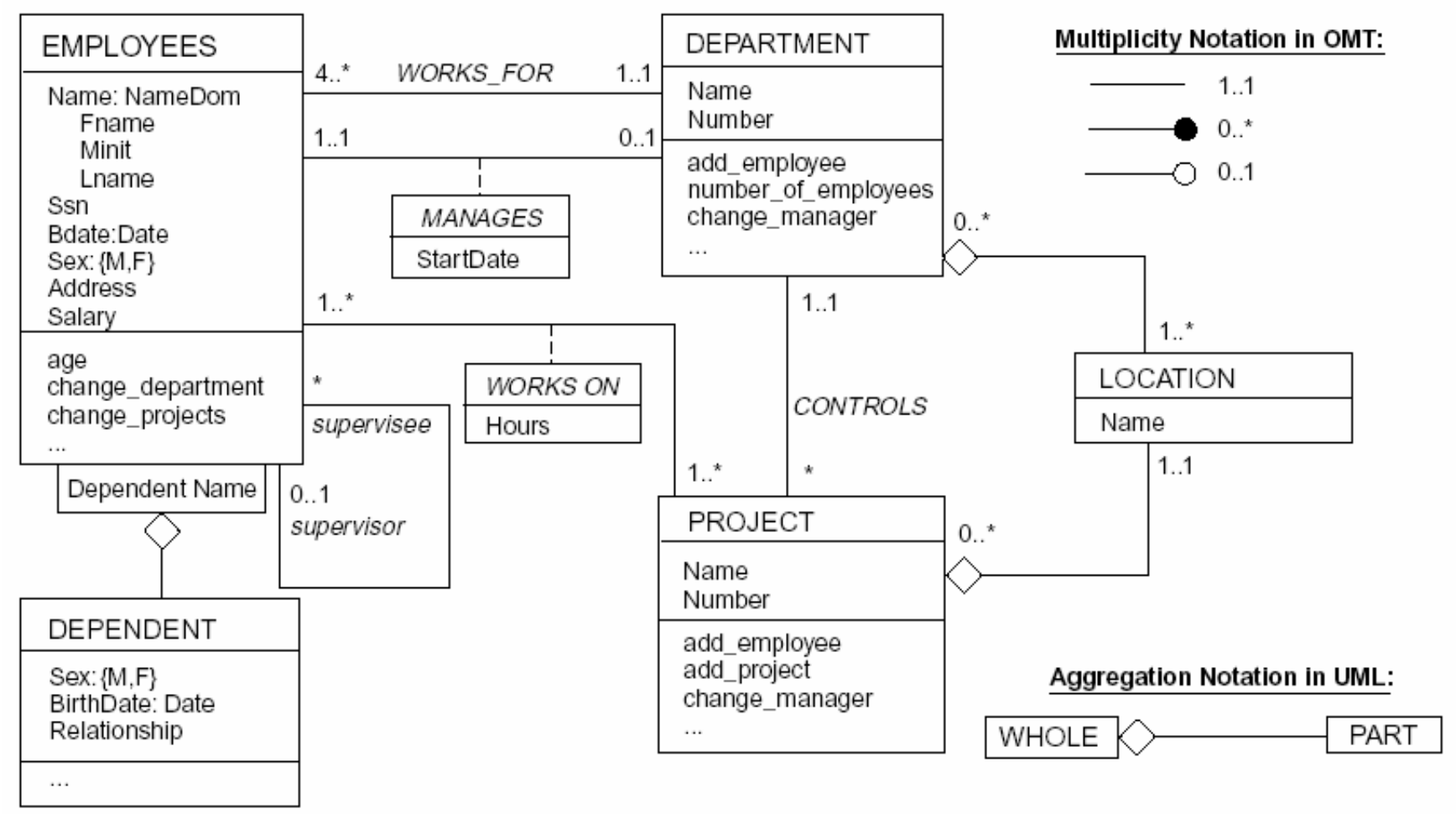
Union of two entity types

(Elmasri/Navathe fig. 4.5a)

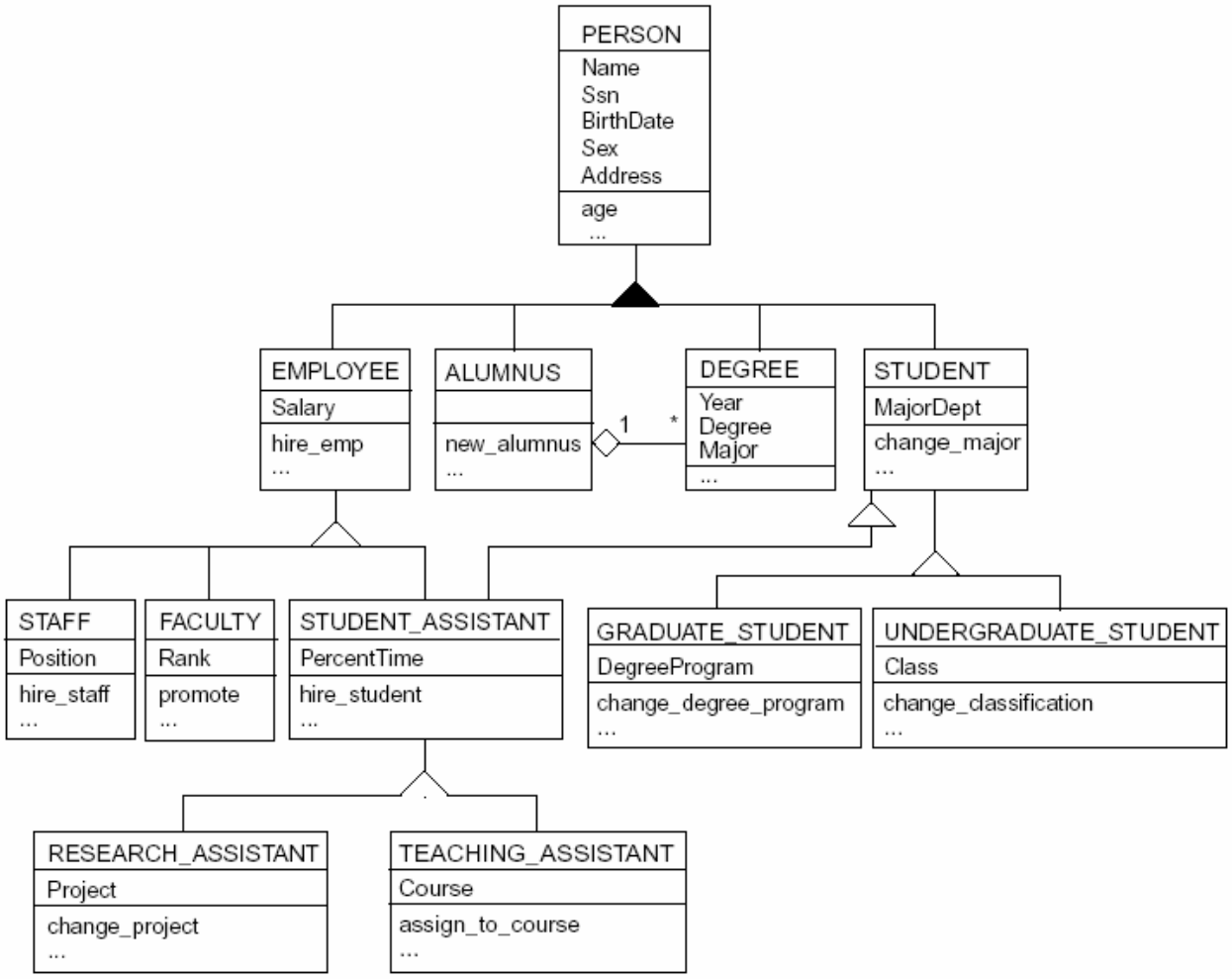


A UML conceptual schema

(Elmasri/Navathe fig. 4.11)



Specialization/generalization in UML
(Elmasri/Navathe fig. 4.12)



Alternative diagrammatic notation for ER/EER (Elmasri/Navathe fig. A.1)