Entity and Enhanced Entity Relationship Models

CS157A

Chris Pollett

Sept. 12, 2005.

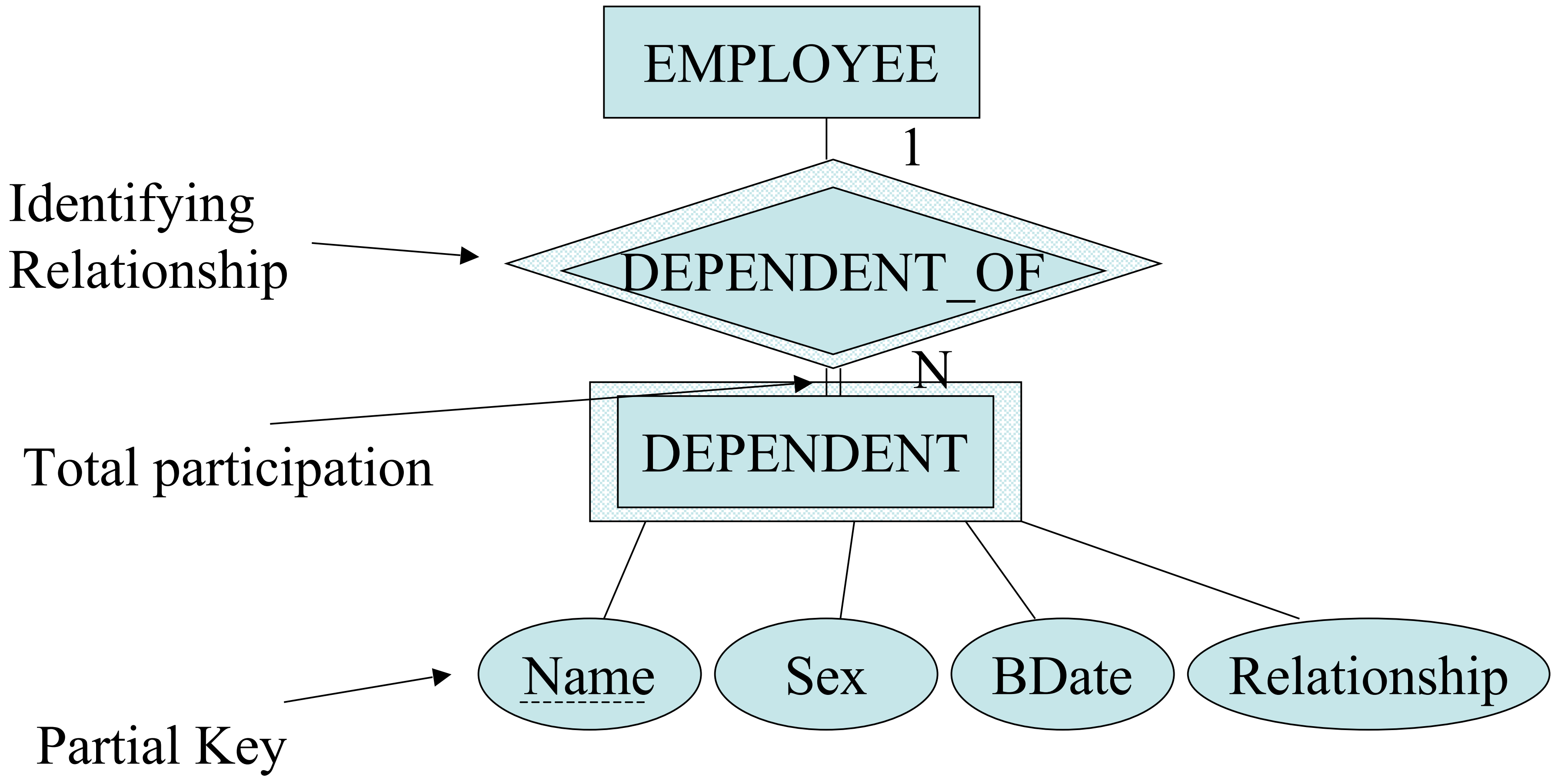
Outline

- Weak Entity Types
- Naming Conventions
- UML
- Subclasses, Superclasses, and inheritance
- Specialization and Generalization

Weak Entity Types

- Entity types which don’t have key attributes of their own are called **weak entity types**.
- Usually entity types which do have a key are sometimes called **strong entity types**.
- Entities belonging to to a weak entity type are identified by their **identifying relationship** with other strong entity types.
- These latter are called **owner types**.
- A weak entity always has total participation in this relationship.
- A weak entity type usually has a partial key which together which together with an owner entity fixes the weak entity.

Weak Entity Notation



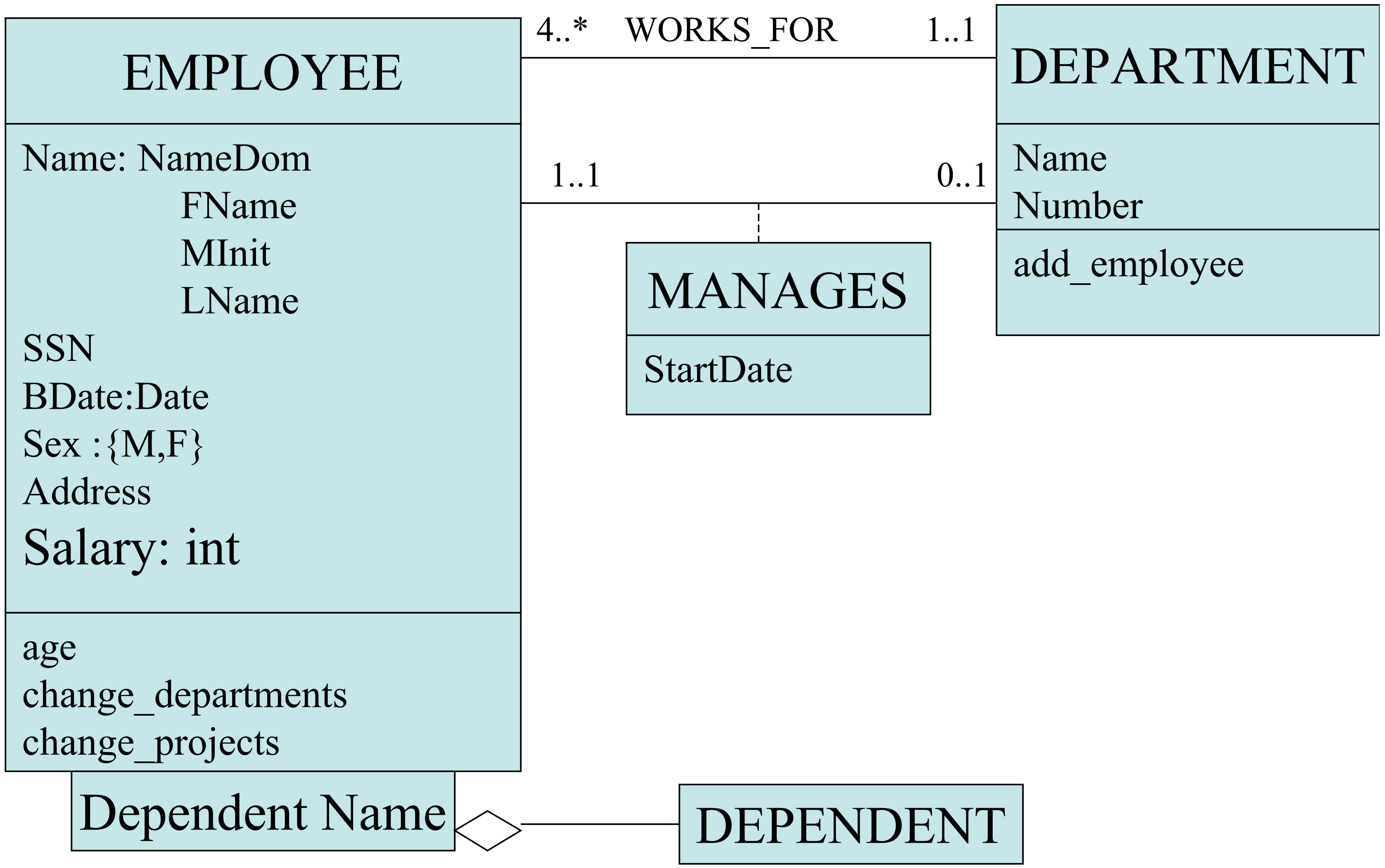
Naming Conventions

- Choose singular rather than plural names for entity types.
- Use nouns for entities; verbs for relationships
- Diagrams typically should be readable from left to right and then top down. (This effects choice of verb.)
- Entity Types and Relationship types should be written all caps.
- Attributes should be camel-cased.

Unified Modeling Language (UML)

- Software design modeling language from 1990s which can also be used to do conceptual database design.
- We will be interested in UML **class diagrams**.
- Entity types correspond to classes in UML.
- Relationship types without attributes correspond to associations. Those with attributes correspond to both classes and associations.
- Weak entity types correspond to has-a aggregations.
- UML also allows one to specify things like triggers, as well as data constraints.

Example UML Diagram



Enhanced Entity Relationship Model

- In the late 1970's and during the 1980s various semantic data models were considered in the areas of CAD/CAM, telecommunications, Geographics Information Systems,etc.
- This led to proposals on how to enhance the ER model to incorporate class/subclass relationships.
- This leads to the Enhanced ER model, EER model.
- Modeling notation for this is not at standardized but for this class we will use the books notation.

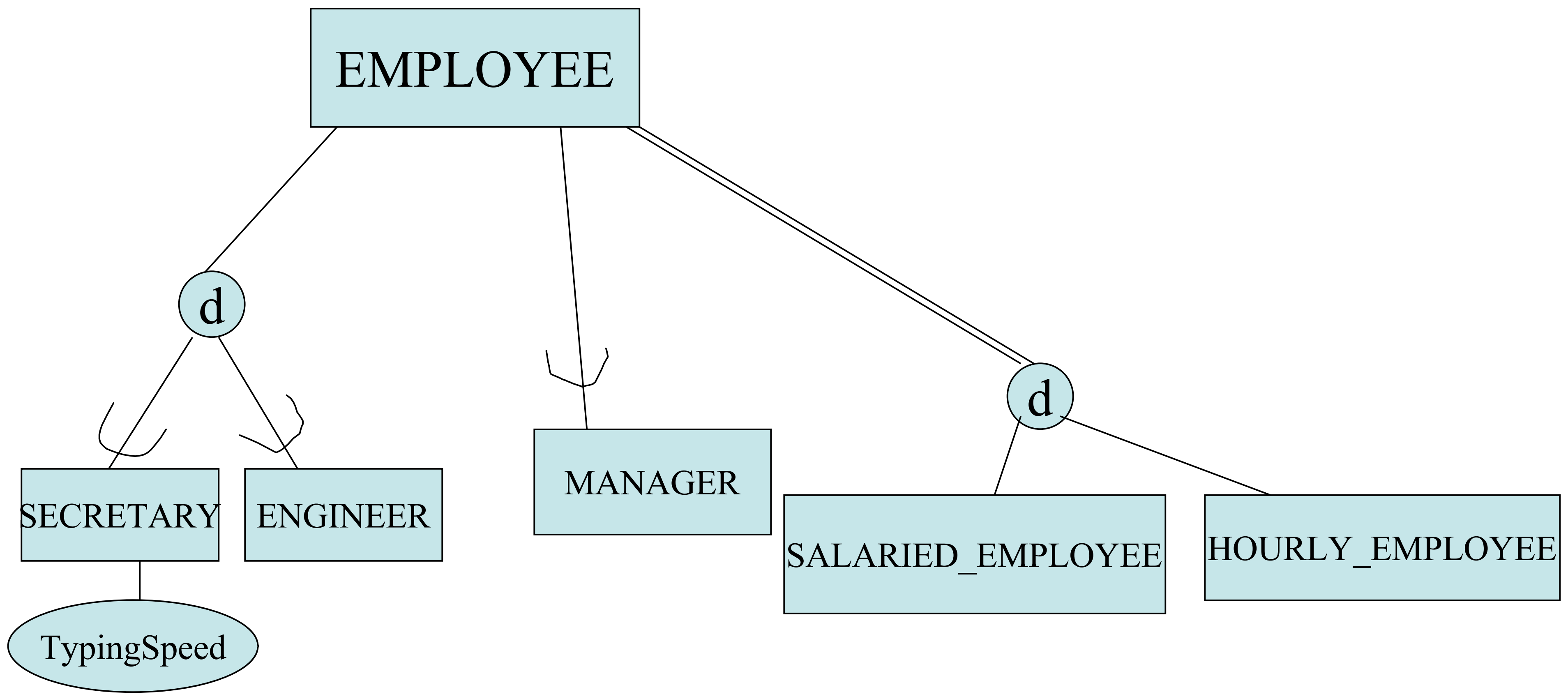
Subclasses, Superclasses, and Inheritance

- Entities belonging to an entity set of some entity type might be further classified.
- For instance, entities in EMPLOYEE, might be classified further as SECRETARY, ENGINEER, etc.
- We call SECRETARY, ENGINEER **subclasses** of EMPLOYEE; and call EMPLOYEE the **superclass** of SECRETARY and ENGINEER.
- So the entity e1 with name Sally Ryder might be both an ENGINEER and an EMPLOYEE.
- However, when stored in the database e1 must exist as an EMPLOYEE if it is to exists as an ENGINEER.
- Also, entities in a subclass possess (**inherit**) all the attributes of the superclass. This property is called **type inheritance**.
- Subclasses are allowed to have additional attributes not found in the superclass.

Specialization

- **Specialization** is the process of defining a set of subclasses of a given entity type.
- This set of subclasses is defined on the basis of some distinguishing characteristic. For example, EMPLOYEE is specialized via job type into SECRETARY and ENIGINEER.

EER Notation for Specialization



Generalization

- This is the reverse process of specialization.
- One might notice a set of entity types share many of their attributes.
- One could then create a common **generalized** superclass for them.
- This is called **generalization**.
- For example, the entity types CAR and TRUCK might be generalized to VEHICLE.