Persistent Data Design

Data design

- A local classes diagram(or ER model) required by a component, or a global classes diagram for the whole system is mapped to the design level
- Data type of attributes is defined
- Operators/methods within a class can be added based on the messages sent/received by objects in sequence diagrams, or actions in state diagrams
- The relational model is used for data design in RDBMS

Relational model

- Relation schema $R(A_1, A_2, \ldots, A_n)$,
  - each attribute A_i belongs to a domain D, $D=dom(A_i)$
- Relation $r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$

  $r=\{t_1, t_2, \ldots, t_m\}$, $t_i=<v_{i1}, v_{i2}, \ldots, v_{in}>$, $v_{ij} \in Dom(A_j)$ or Null

t is called tuple.

In a DB schema, r coressponds to a table, t corresponds to a row.

Example:

Book(ISBN, Title, Authors, Published_year, Publisher)

ISBN: domain is string (50)

Title: domain is string(300)

Relations

- A relation has a primary key and may have foreign keys
- A primary key is a minimal set of attributes and the values in these attributes uniquely identify a single tuple in the relation
- A foreign key is defined as a set of attributes in a relation, the values of which are either NULL or required to match the values of the primary key in the same or another relation.

Mapping Class/Entity to Relation

- 1.Mapping of entity class

Class/Entity E → relation R, key attributes → primary key

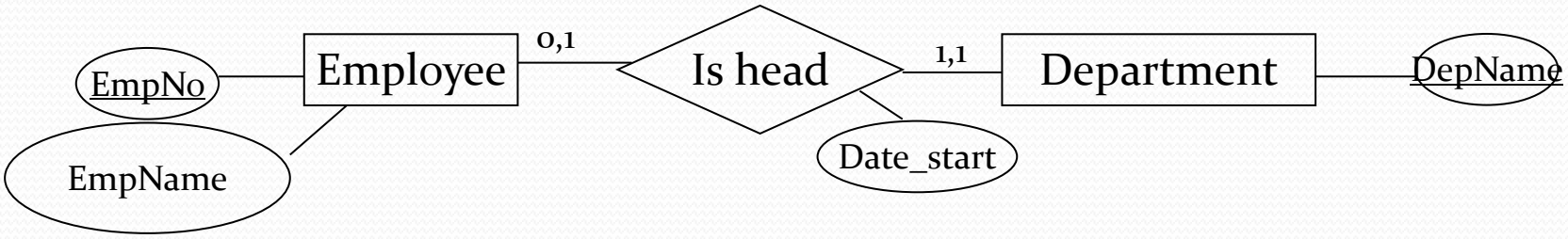
Mapping Association/Relationship

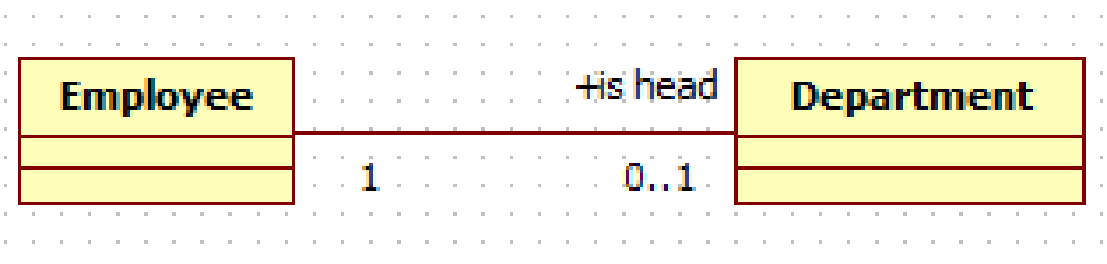
- 1. Mapping 1:1 association

  Department (DepName, EmpNo, Date_start)

  DepName is PK, EmpNo is FK

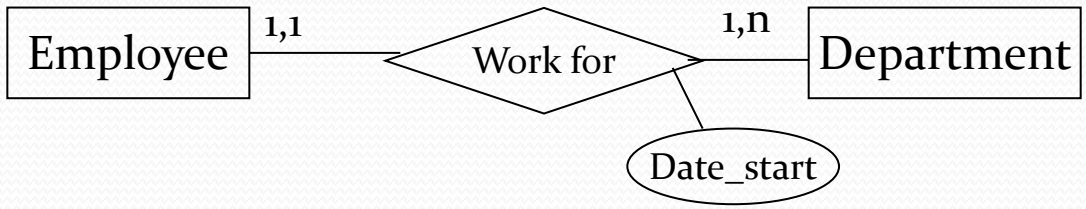
  Employee(EmpNo, Emp_Name)





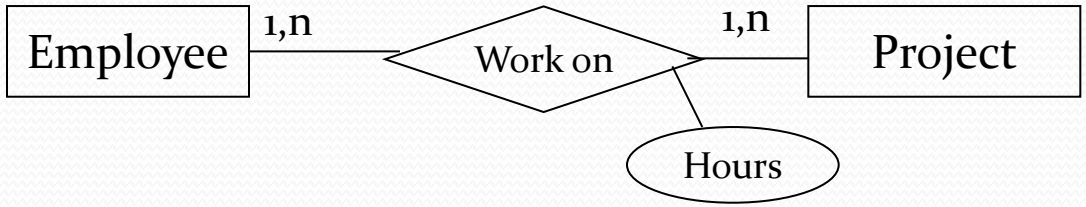
Mapping Relationship/Association

- Mapping of binary 1:N relationships types



Employee(EmpNo, DepName, Date_Start), Department(DepName)

- Mapping of binary M:N relationships types

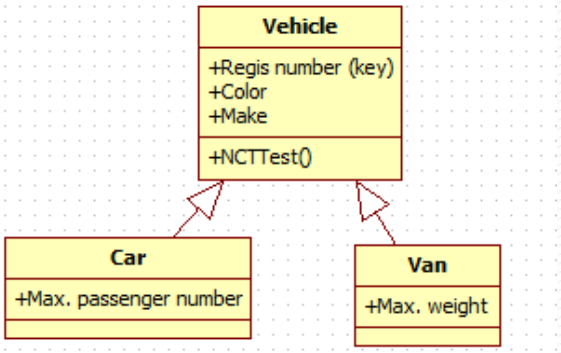


Employee(EmpNo), Project(ProNo), Work_on(EmpNo, ProNo, Hours)

Mapping Specialization

- Supper-class C, attribute of C {k, a1, a2, …,an}, sub-classes S1, S2, …, Sm. K is key attribute of C.

- Inheritance
  - Create a relation Li for each sub-class Si, Attrs(Li)=Attrs(Si) ∪ {k, a1, a2, …, an} and PK(Li)=k
  - *This option only works for non overlaping specialization. If the specialization is overlapping, then an entity(object) may be duplicated in several relations.*



Car(Registration number, Color, Make, Max. passenger nu
Van(Registration number, Color, Make, Max. weight)

Mapping Specialization (contd.)

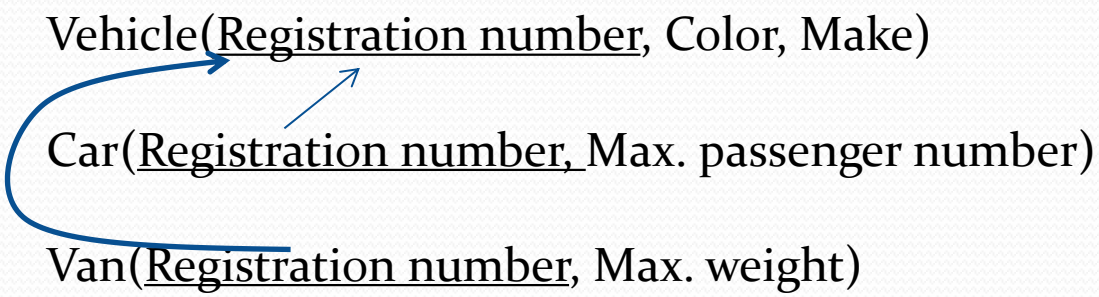
- Horizontal
  - Create a single relation L with attributes Attrs(L)= {k, a1, a2, …,an} ∪ Attrs(Si) ∪ {t} and PK(L)=k, the attribute t is a discriminating attribute to indicate the sub-class to which each tuple belongs
  - *This option only works if child entities are disjoint*.
  - If sub-classes are overlapping, Attrs(L) = {k, a1, a2, …,an} ∪ Attrs(Si) ∪ {t1, t2, …, tm}, each ti is a boolean type attribute.

Vehicle(Registration number, Color, Make, VehicleType, Max. passenger number, Max. w

VehicleType can take the value ‘car’ or ‘van’

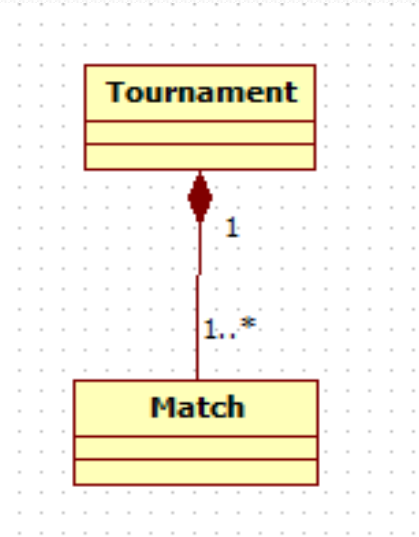
Mapping Specialization (contd.)

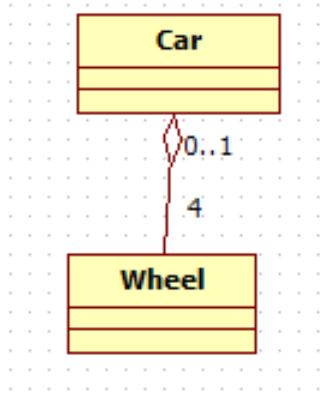
- Vertical
  - Create a relation L for C, with Attrs(L)={k, a1, a2, …,an} and PK(L)=k. Create a relation Li for each sub-class Si, Attrs(Li)={k} ∪ Attrs(Si) and PK(Li)=k
  - *This option works for any specialization*



Mapping Composition/Aggregation (contd.)

- A composition is a kind of 1: N association => using the same mapping rule as (1,1), (1,n) association
- Aggregation is a kind of (0,1) , (1,n) association





Implementation level

- At the implementation level a relation corresponds to a table, an attribute corresponds to a column of the table, a tuple is a row.