escola d'enginyeria

UAB

ESCI upf. School of International Studies

Block 4
ADVANCED ER-DESIGN (PART 3)

Debora Gil, Oriol Ramos, Carles Sanchez

Contents: Advanced Design-ER

1. Weak Entities

    1.1 Definition and Examples

    1.2 Design with Weak Entities

2. Aggregations

    2.1 Specialization

    2.2 Generalization

    2.3 Aggregation

Contents: Advanced Design-ER

1. Weak Entities

1.1 Definition and Examples

1.2 Design with Weak Entity

2. Aggregations

2.1 Specialization

2.2 Generalization

2.3 Aggregation

Specialization

Definition:

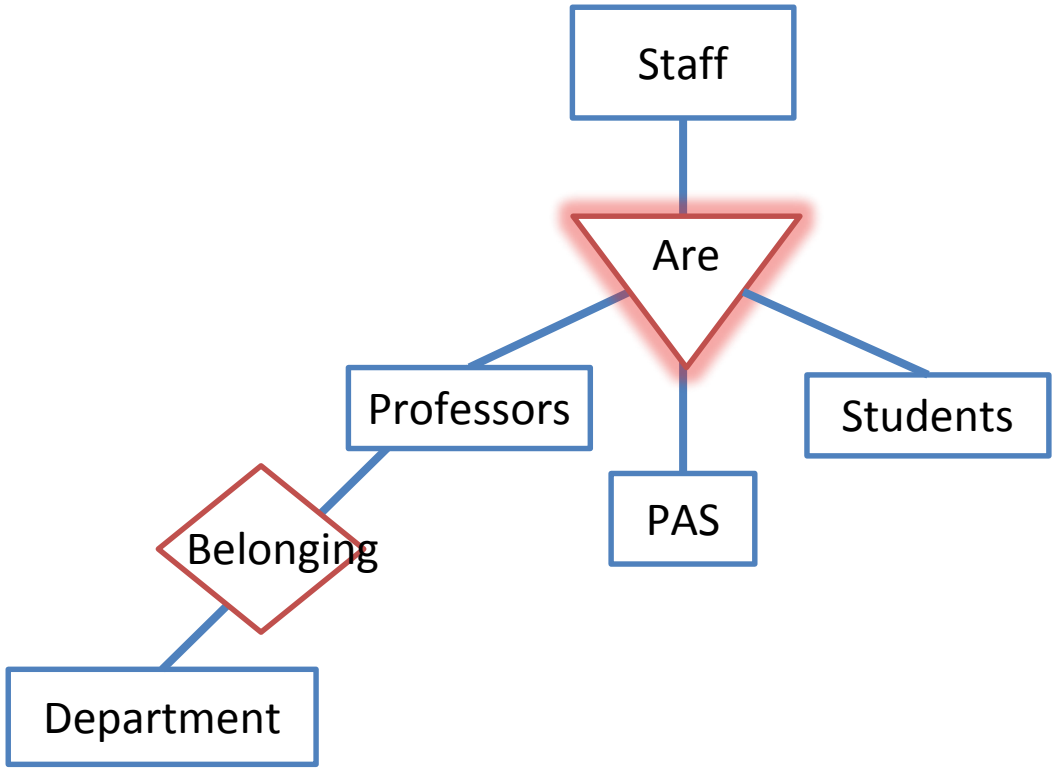
Partition of an entity in specific groups (low-level entities)

An entity may include entities subgroups that are somehow different from other entities. The process of appointment of subgroups within a set of entities is called **Specialization**.

Specialization

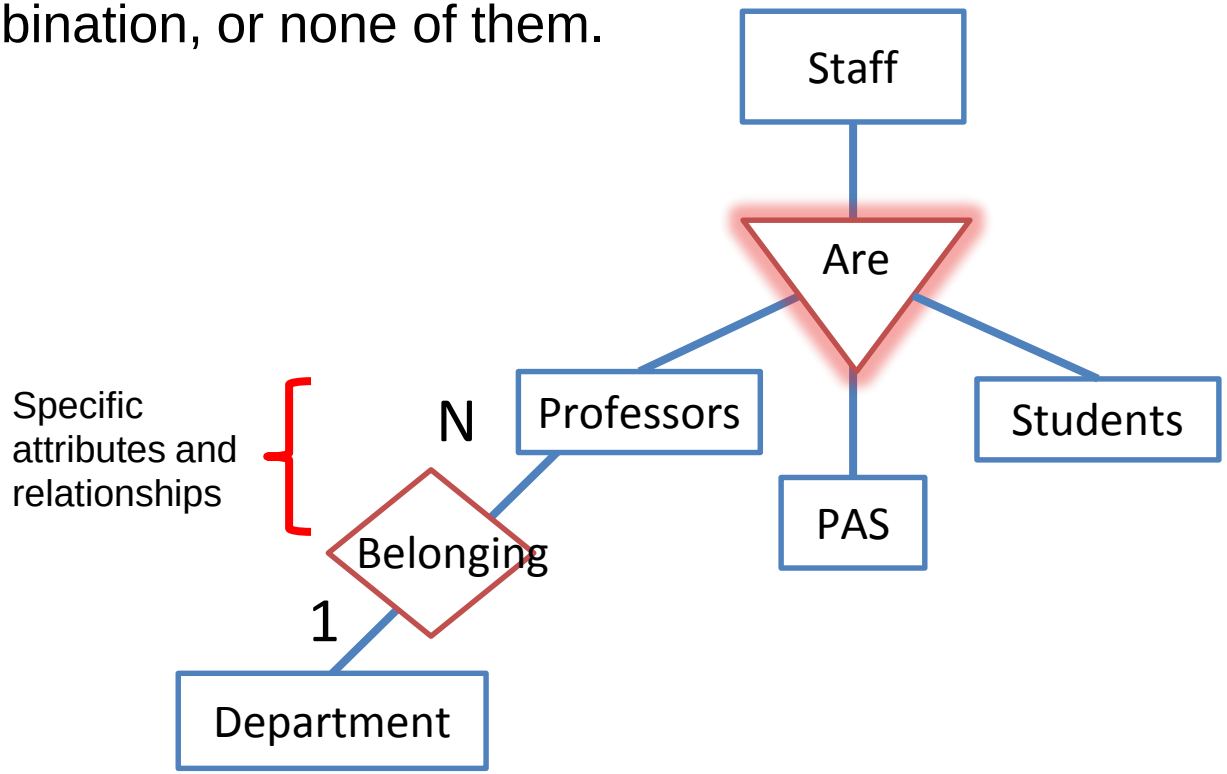
Example: instances of “Staff" entity can be classified as one of the following:

- Employees
- Students
- Administration and Services (PAS)



Specialization

The specialization of “Staff" allows us to distinguish between entities “Staff" according to whether they correspond to the “Teachers", "PAS" or "Students": In general terms, the staff could be an employee, PAS, a student, a combination, or none of them.



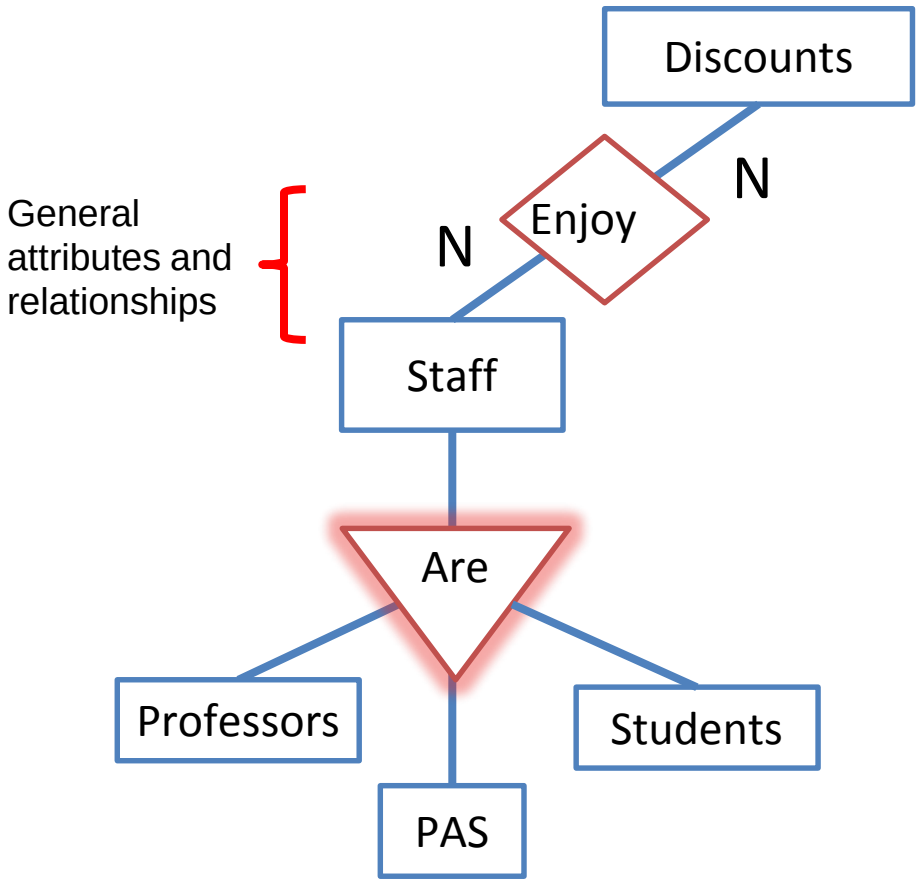
Generalization

Definition:

Process summarizing multiple entities into a high level entity based on common characteristics (the reverse process of specialization)

There may be similarities between entity "A" and entity "B", for instance, several attributes that are conceptually the same in both entities. This coincidence can be expressed by **generalization,** which is a containment relationship between an higher level entity and one or multiple lower level entities.

Generalization



The generalization is used to merge a set of entities that share the same characteristics on a set of higher-level entities.

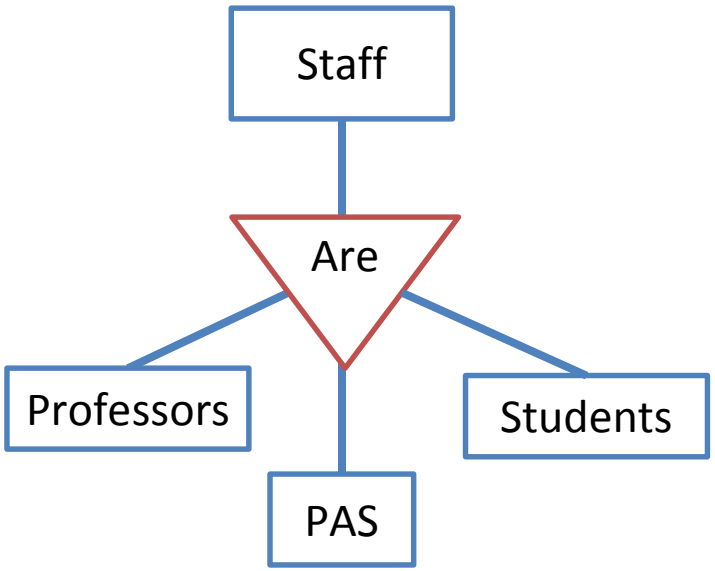
Allows to emphasize the similarities among lower level entities and to hide the differences.

Economic representation: shared attributes are not repeated

Generalization and Specialization

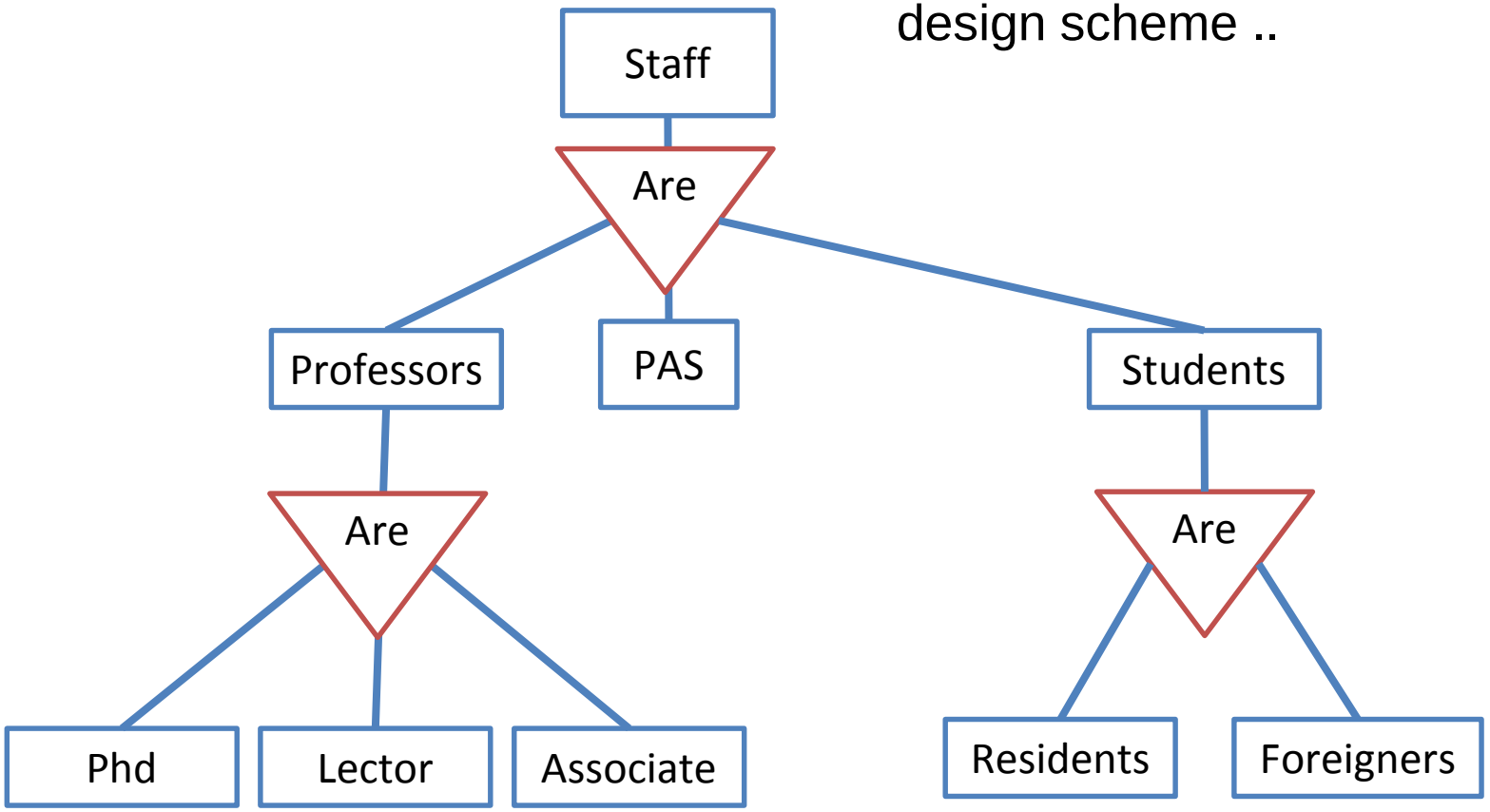
Common Features

- Grouping of entities with common attributes into a higher level entity
- Eliminate redundancies
- The ER diagram itself does not distinguish between specialization and generalization.
- Attributes of the higher level entities **are inherited** by lower level entities.
- For example, students, teachers and staff inherit “Staff” attributes.



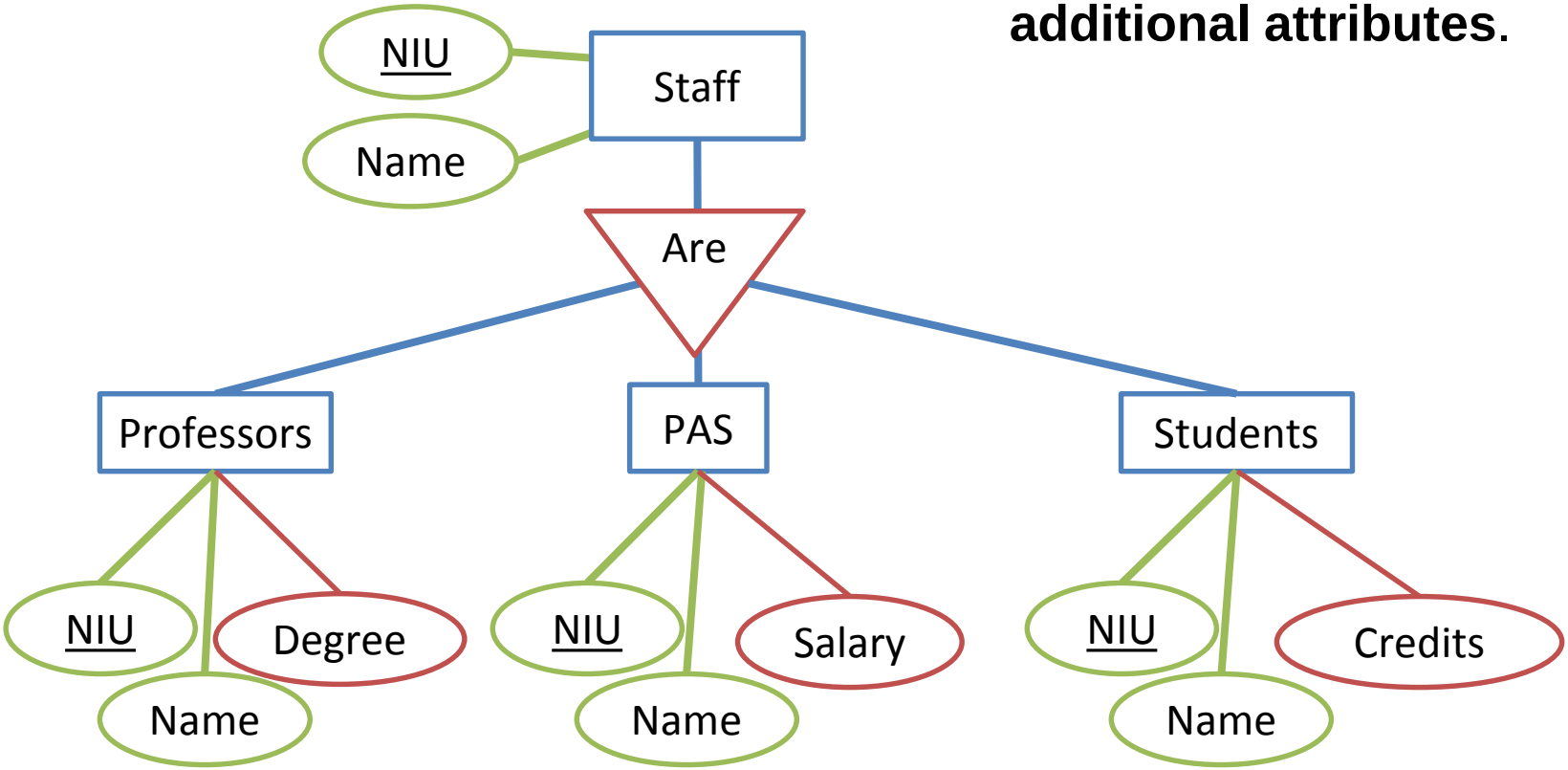
Generalization and Specialization

We can apply the specialization / generalization more than once to refine a design scheme ..



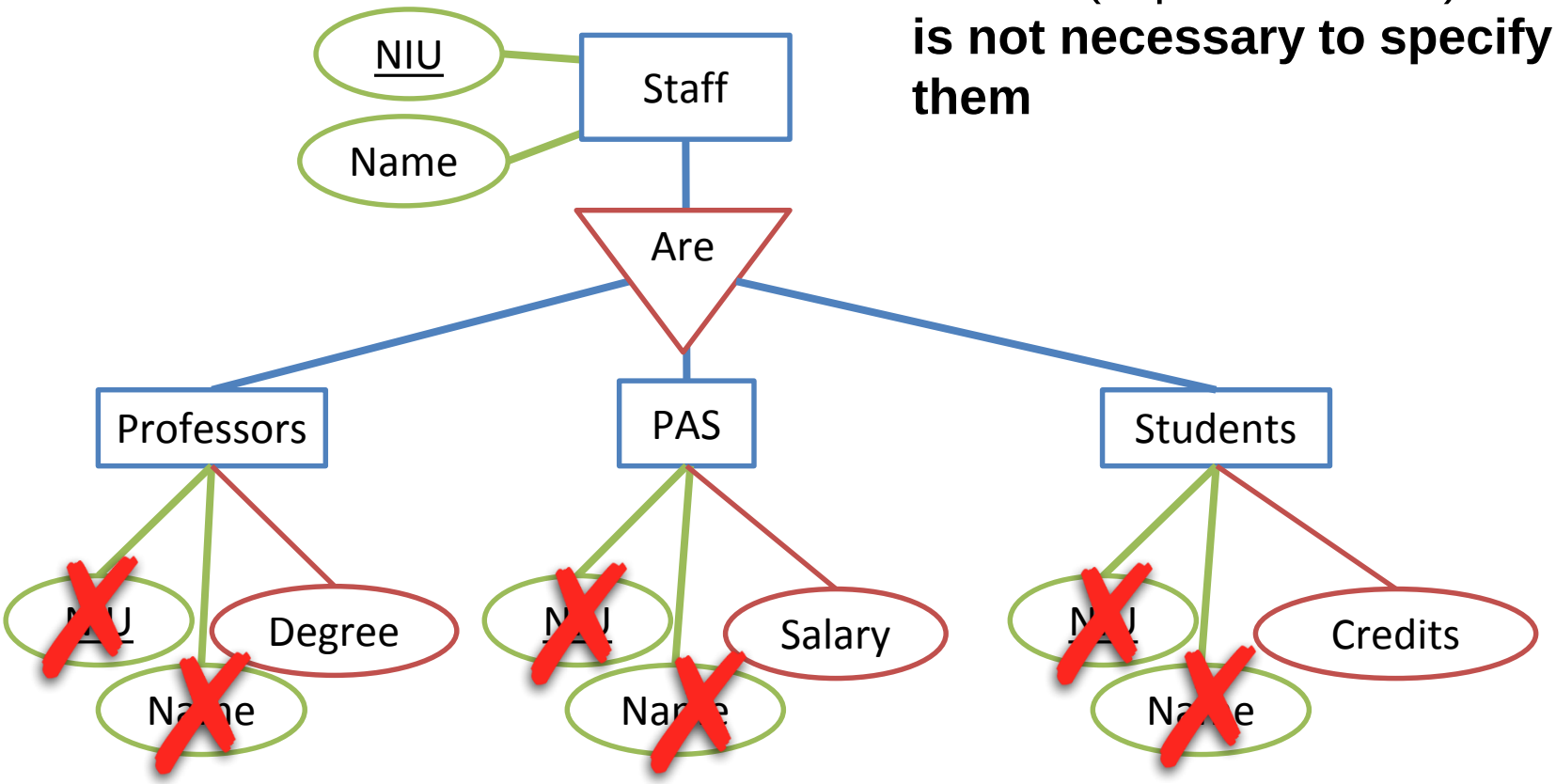
Generalization and Specialization

Each of these “Staff” sub-entity is described by a set of attributes that includes all the “Staff” attributes adding possibly **some additional attributes**.



Generalization and Specialization

Attributes and relationships of higher level entities are inherited by the low level entities (in particular CP) and **it is not necessary to specify them**

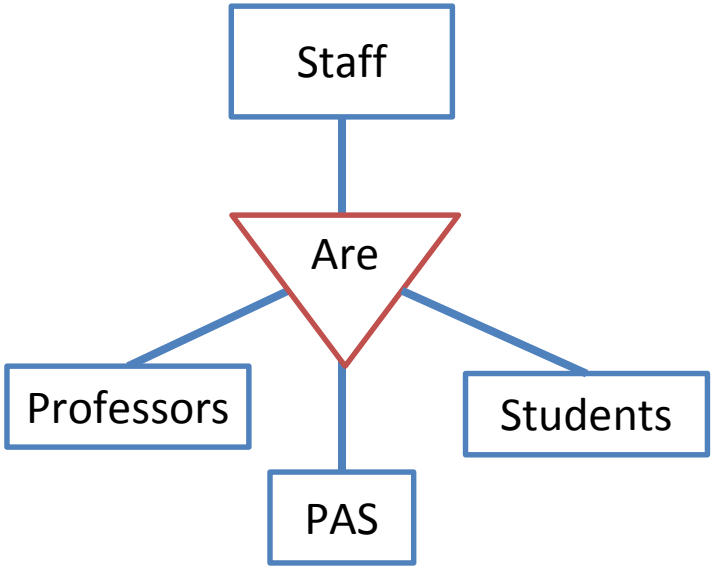


Generalization and Specialization

Restrictions

The database designer can decide to place certain restrictions on a particular specialization or generalization

- **Participation:**
  - **Total:** All the higher level entity instances must belong to an lower level entity.
  - **Partial:** There are instances of the high-level entity that can not be classified (default)



Obs: The generalizations are usually of total participation

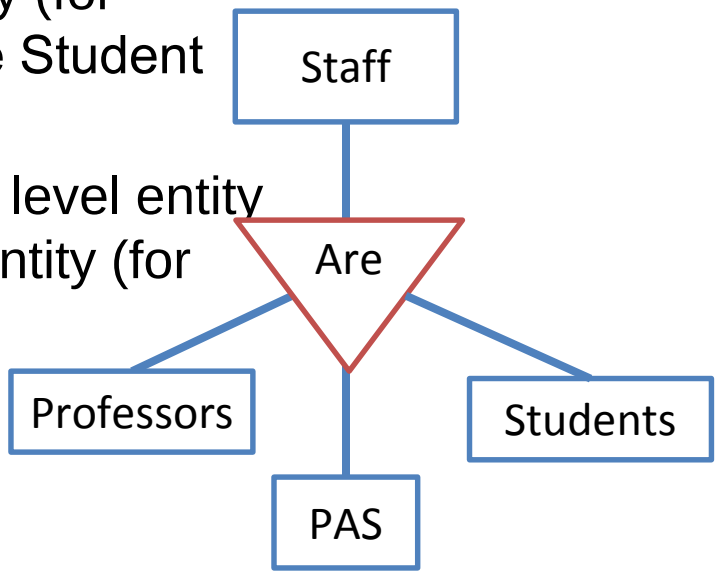
Generalization and Specialization

Restrictions

The database designer can decide to place certain restrictions on a particular specialization or generalization

Partition overlap:

- **Disjointed:** Each instantiation of the high level entity belongs to only one low level entity (for example, a member of “Staff“ can not be Student and PAS)
- **Overlapped:** Some instance of the high level entity can belong to more than one low-level entity (for instance, a “Staff” member can be both Student and PAS)



Generalization and Specialization

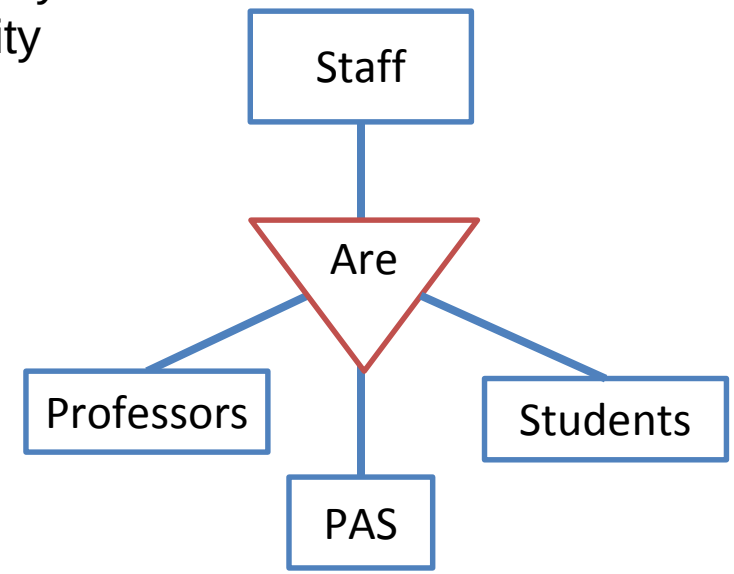
Restrictions

The database designer can decide to place certain restrictions on a particular specialization or generalization

- **Assignment:**
  - By **condition**. Some condition is mandatory over some attributes of the high level entity

Example:

- Students: people what are enroled to some course
- Professors: people what teach some course
- PAS: no teaching hired staff



  - By **user:** Instance manually assignment

Aggregation

Definition:

Grouping of relationships and participating entities into a new entity

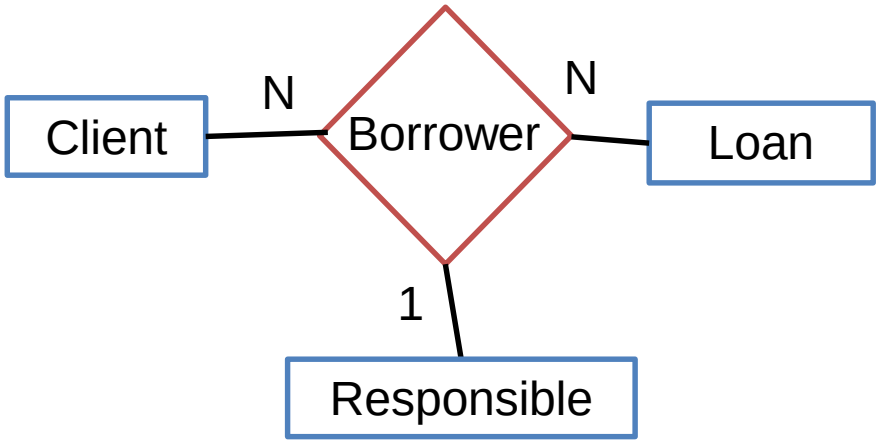
- A limitation of the ER model is that it can not express relationships between relationships
- To solve this problem, we create aggregations that are abstractions through which relationships are treated as higher level entities

Aggregation

Example 1: Bank Loan

A bank wants to keep information on loans, their clients and a bank responsible for each loan.

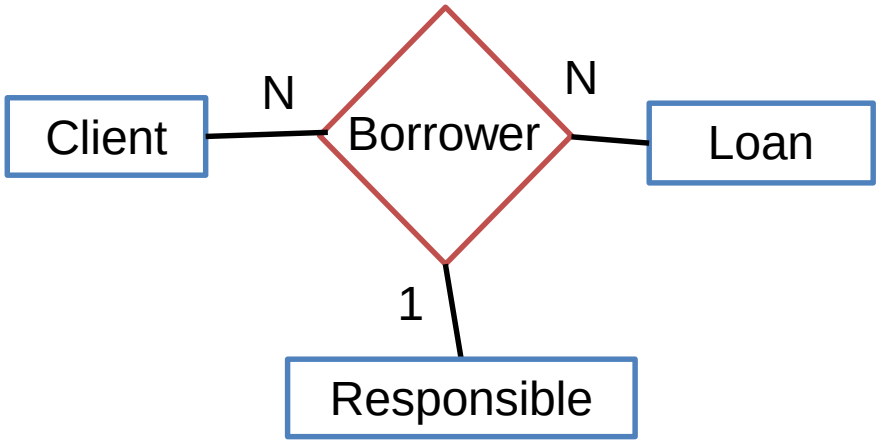
The relationship client-loan is N-N



Aggregation

Example 1: **Bank Loan**

We need to assign the responsible when the loan is accepted



(DNI1, loan1, responsible1)

(DNI1, loan2, responsible1)

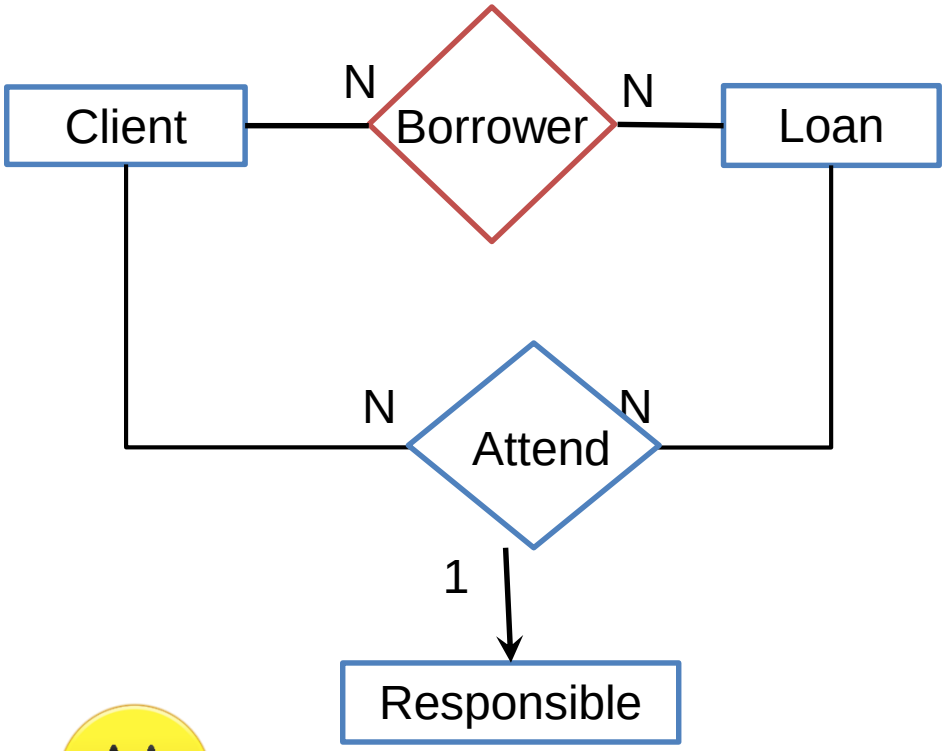
(DNI2, loan3, responsible1)

(DNI2, loan4, responsible2)

Very rigid

Aggregation

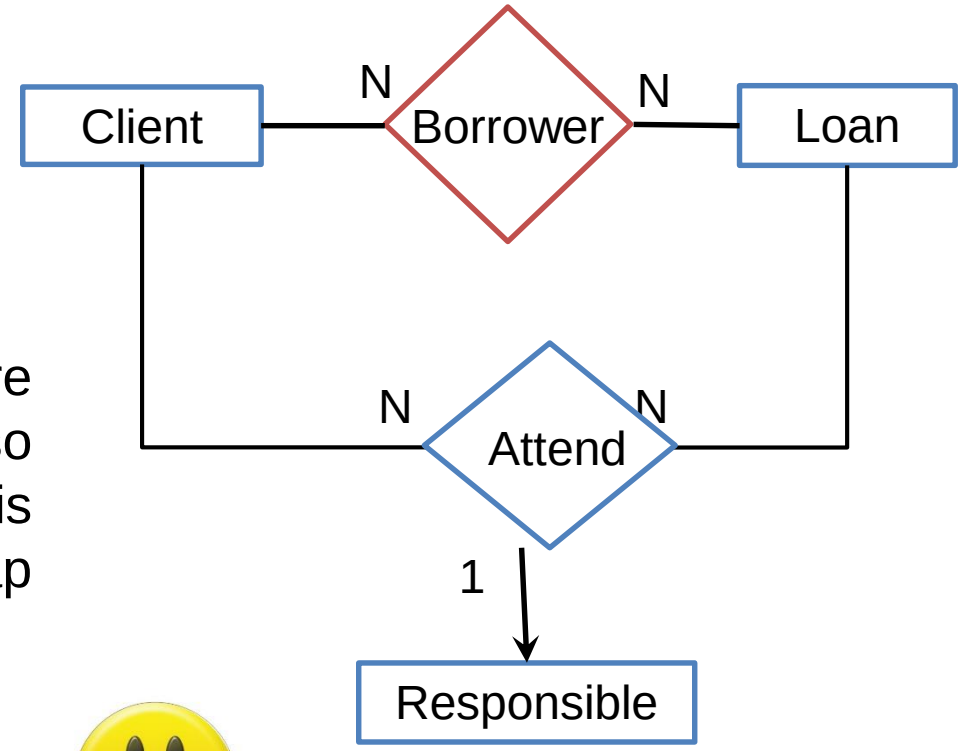
Example 1: Bank Loan



more convenient

Aggregation

Example 1: **Bank Loan**

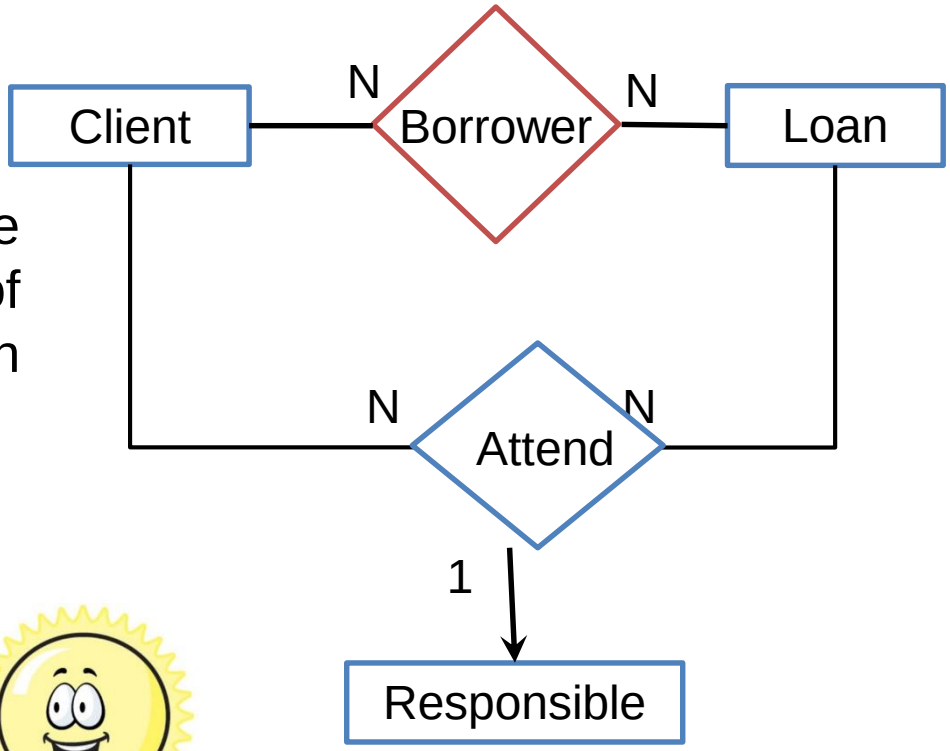


Not as it seems: we must ensure that every instance of “Attend” is also an instance of “Borrower” and this is not reflected in this diagram (Trap Connection)

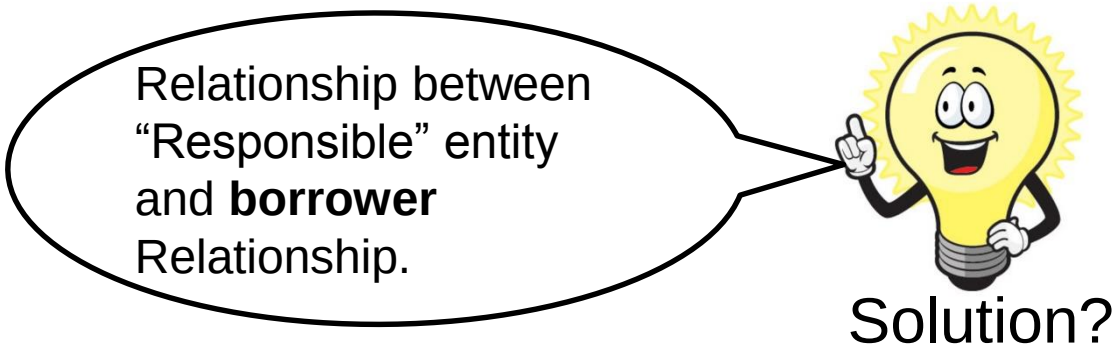
more convenient?

Aggregation

Example 1: **Bank Loan**



We must ensure that every instance of “Attend” is also an instance of “Borrower” and this is not reflected in this diagram (Trap Connection)

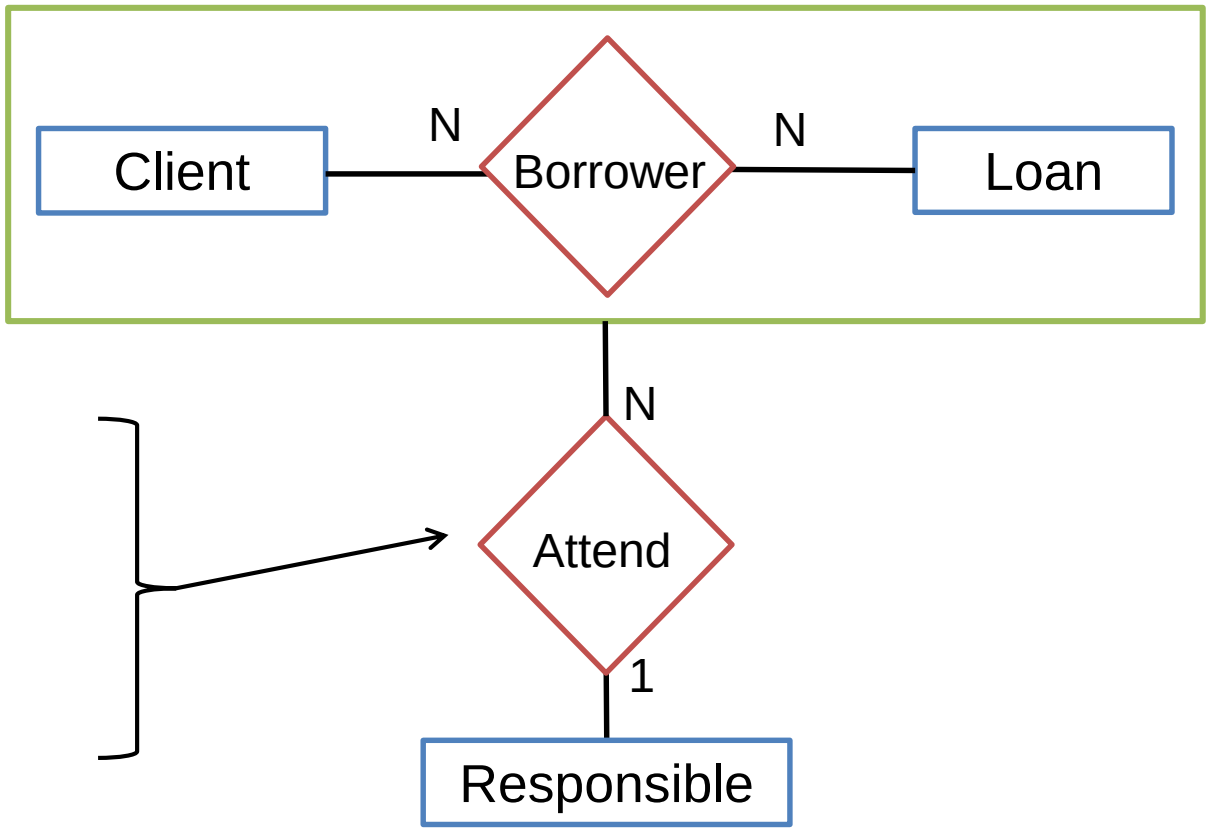


Solution?

Aggregation

Example 1: Bank Loan

(DNI1, loan1)
(DNI1, loan2)
(DNI2, loan3)
(DNI2, loan4)



((DNI1, loan1) responsible1)
((DNI1, loan2) responsible1)
((DNI2, loan3) responsible1)
((DNI2, loan4) responsible2)

Aggregation

Example 2: Classrooms booking

We want to manage ETSE classroom timetable booking and know availability. For each school day, reservations are fractions of an hour from 9:00 to 20:00.

For every classroom we keep the code, location and capacity. For each course we want to know the name and degree to which it belongs.

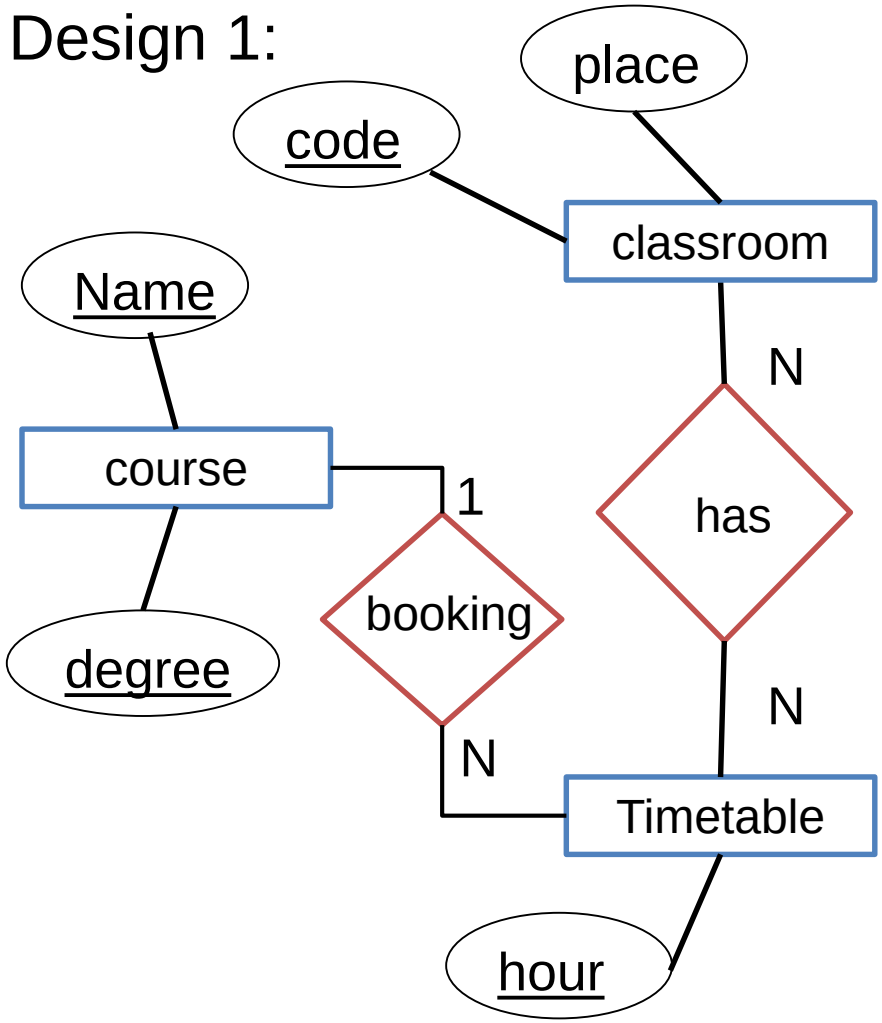
(a) There are only morning and only afternoon scheduled classrooms
(b) All classrooms have the same timetable

Aggregation

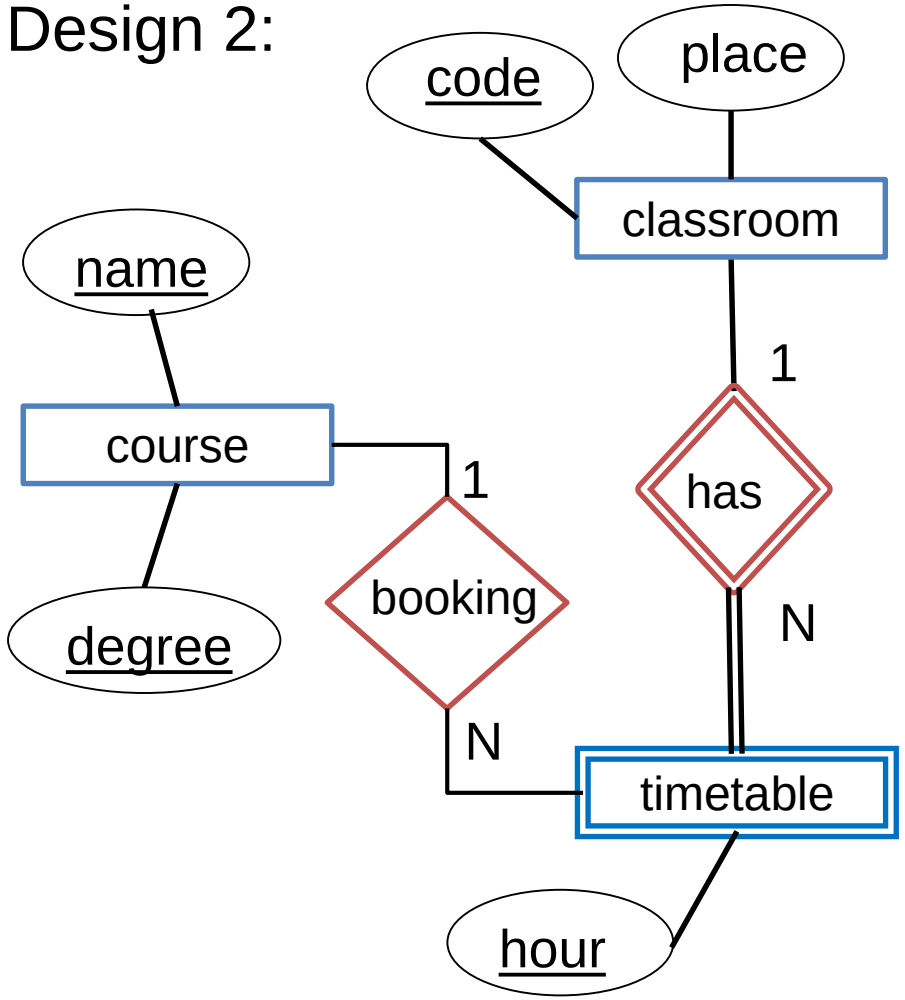
Example 2: Classrooms booking

We propose two designs:

Design 1:

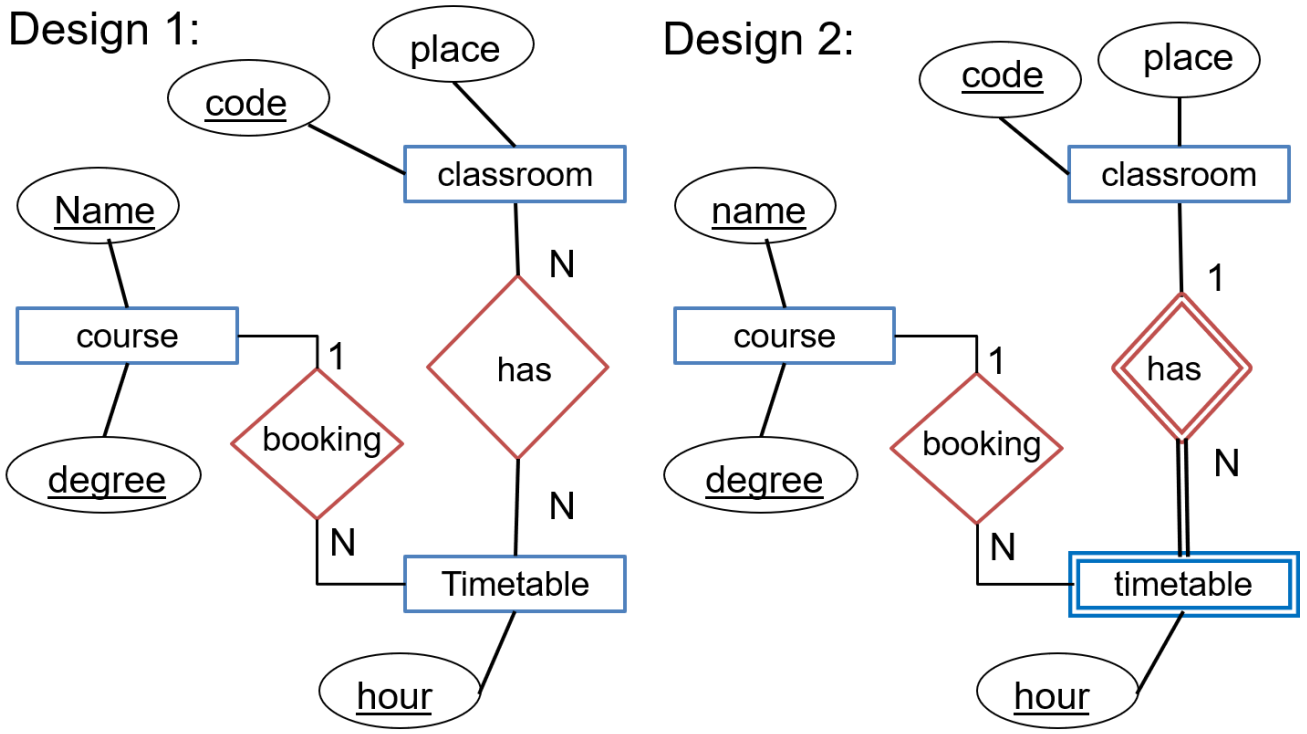


Design 2:



Aggregation

Example 2: Classrooms booking



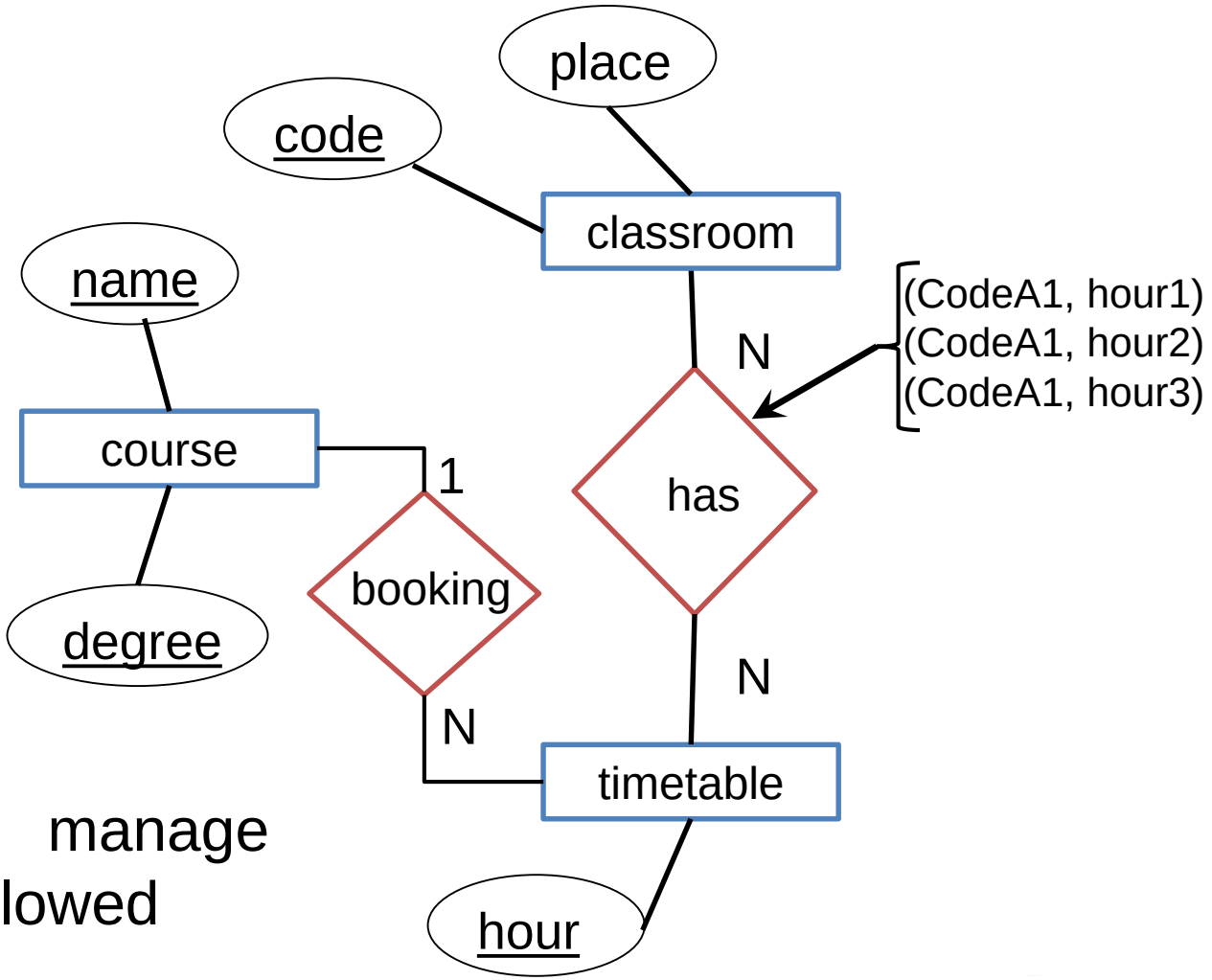
What is the difference between these 2 designs?
Can we know timetable for requirement (A)? How?

Aggregation

Example 2: Classrooms booking

Design 1:

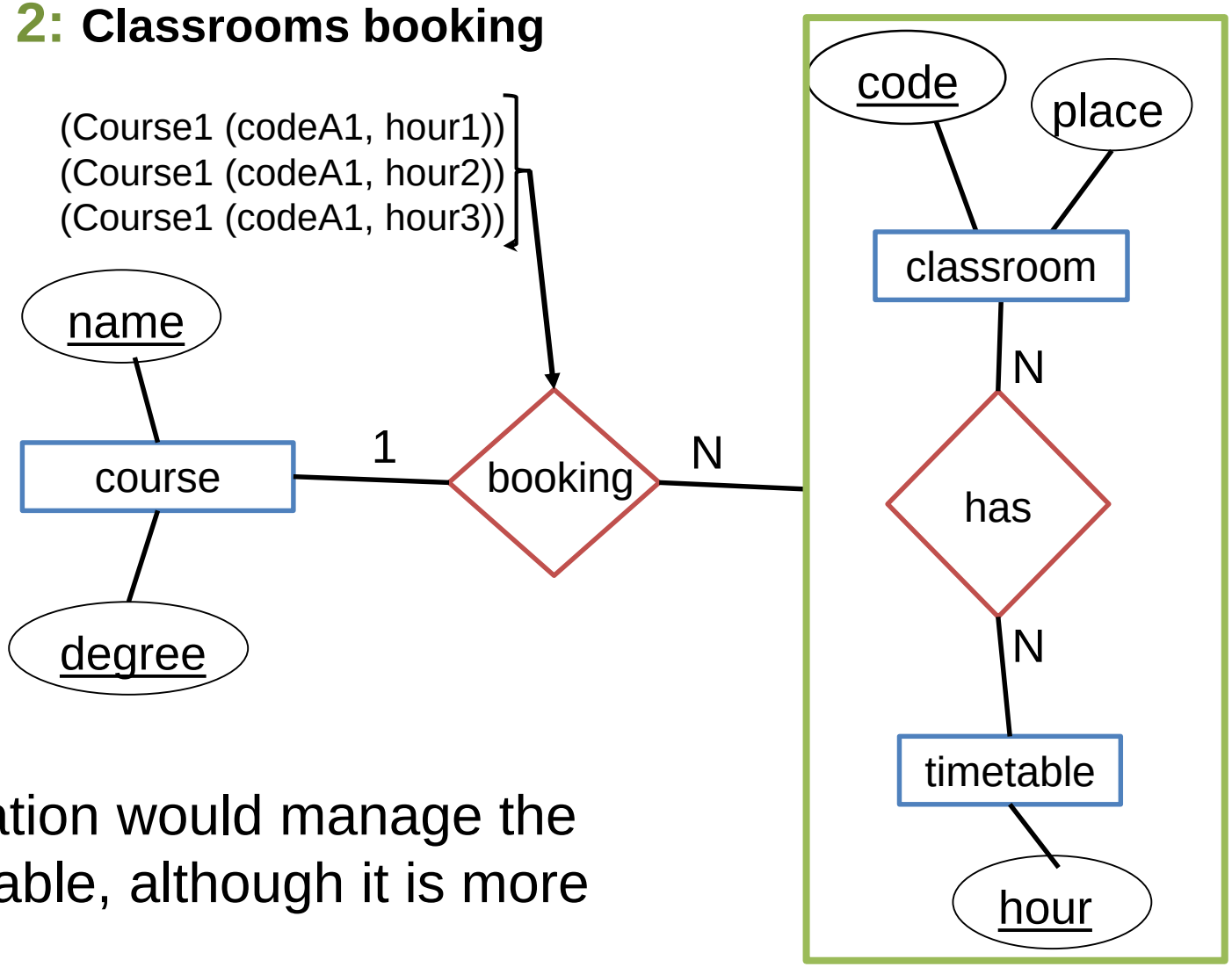
If timetable is a strong entity we have generic (list of all slots) timetables, so every classroom timetable ((a) change depending on the classroom) are in the relationship



With this design manage availability is not allowed

Aggregation

Example 2: Classrooms booking



An aggregation would manage the hours available, although it is more complex

Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform operations outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

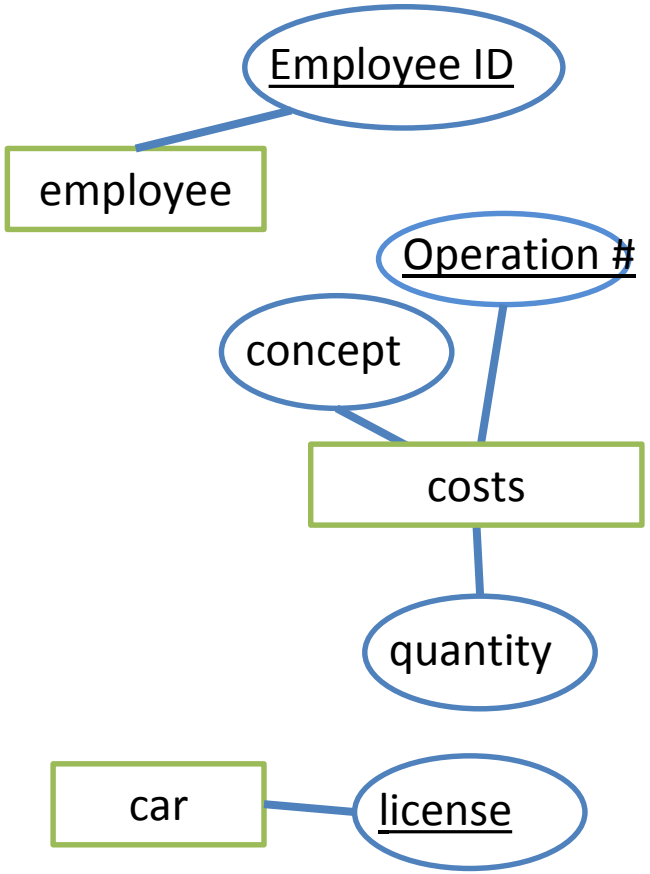
Particularly, for each operation:

- May be required more than one worker
- No limits on the number of cars in use
- The cost sheet should indicate all associated costs
- One of workers involved in the operations (the manager) is in charge (before use) to reserve cars (among those available), and once the operation has been completed, delivering the sheet with all the costs

Aggregation

Example 3: business operations

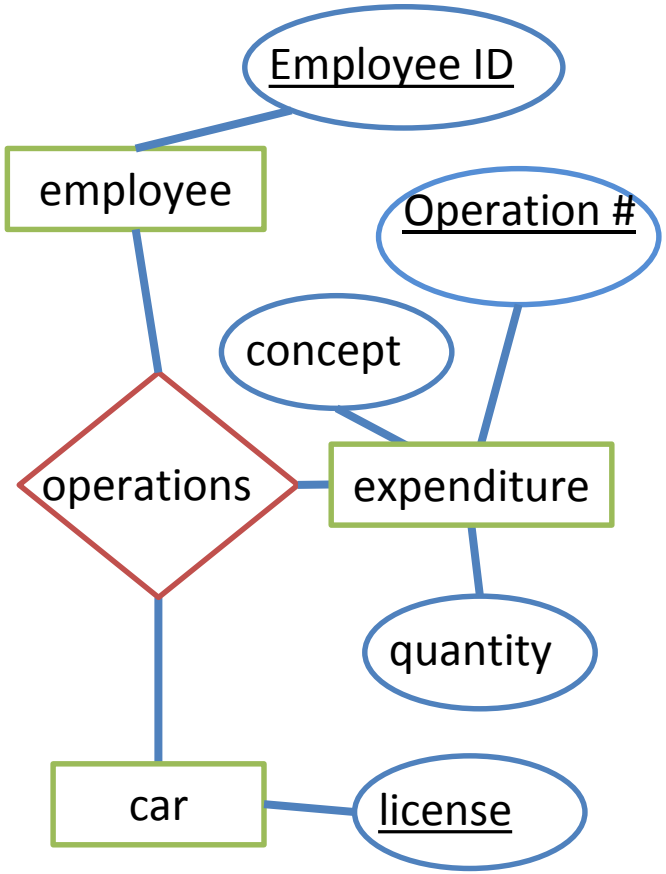
A company has a fleet of **cars** for **employees** for perform operations outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated **costs**.



Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.



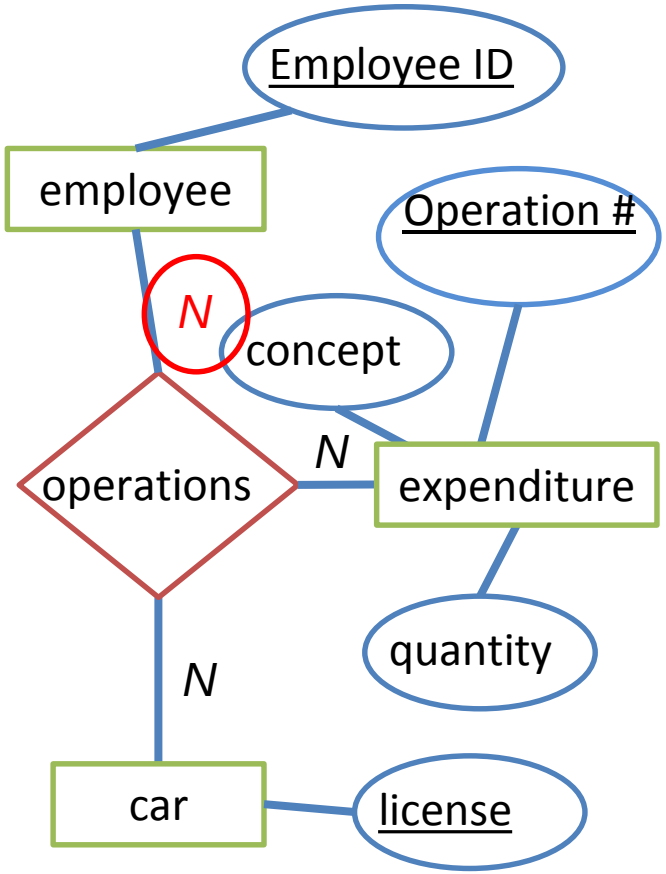
Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

Particularly, each operation:

- May be required more than one worker
- No limits on the number of cars in use
- The cost sheet should indicate all associated costs
- ……



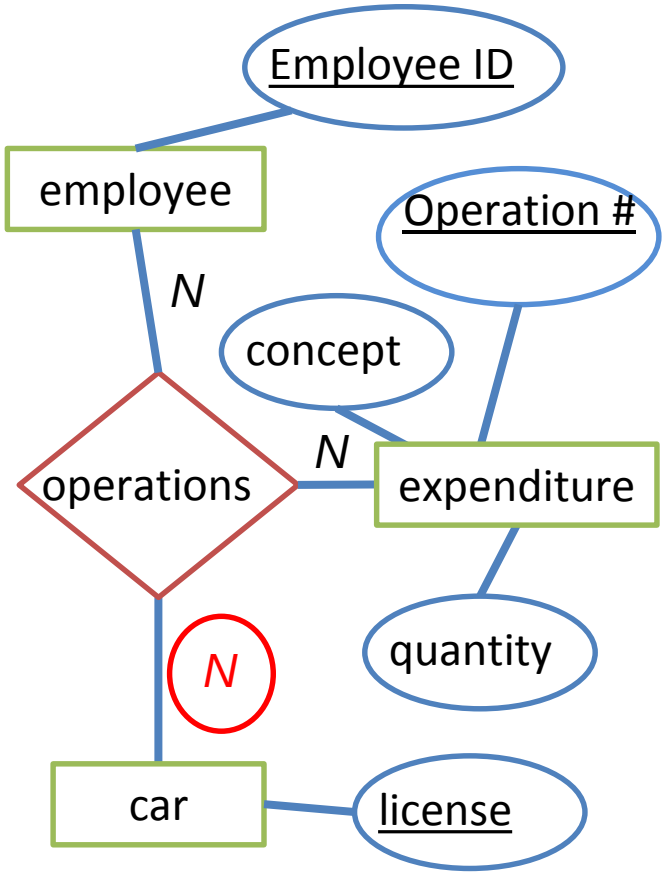
Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

Particularly, each operation:

- May be required more than one worker
- No limits on the number of cars in use
- The cost sheet should indicate all associated costs
- ……



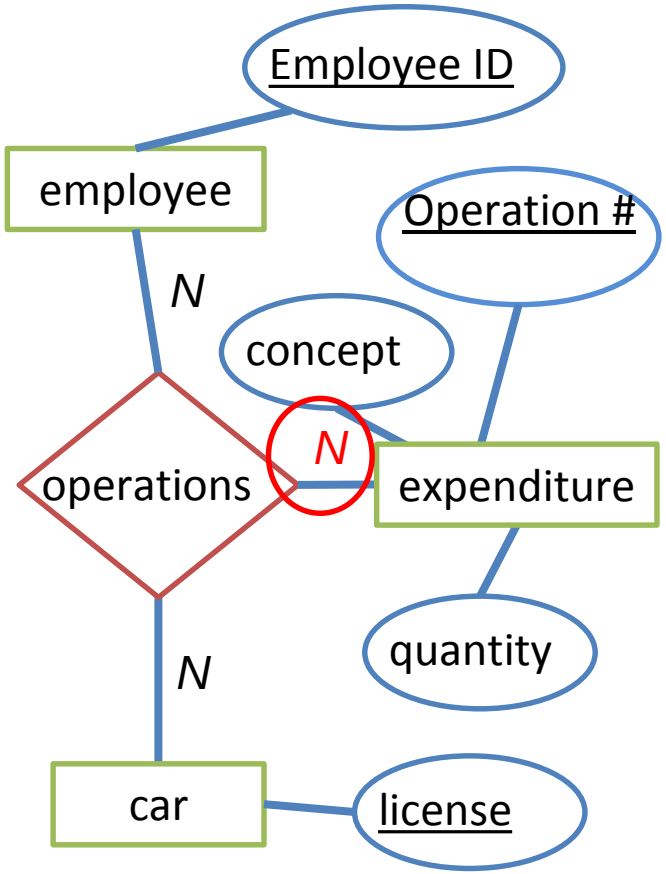
Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

Particularly, each operation:

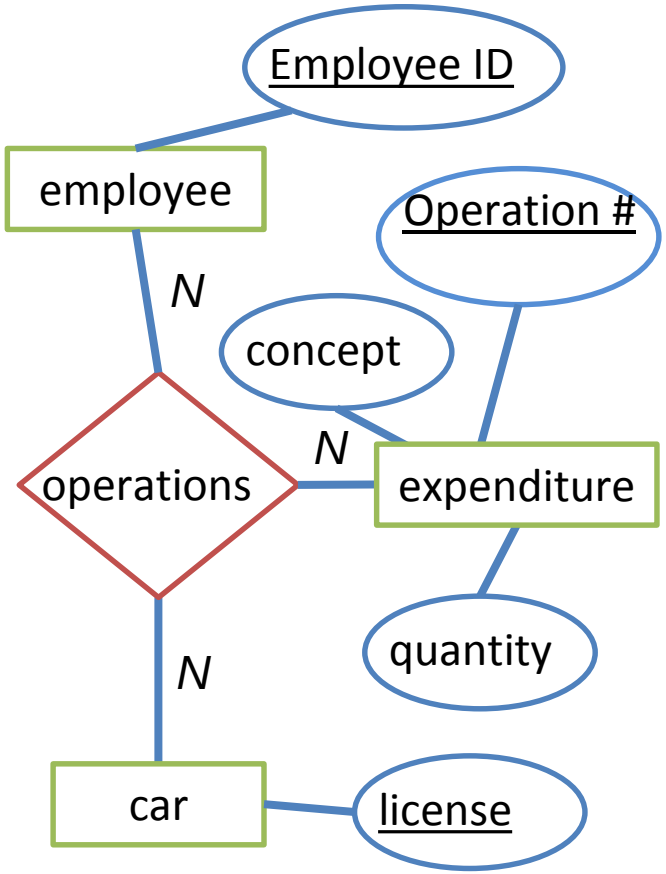
- May be required more than one worker
- No limits on the number of cars in use
- The cost sheet should indicate all associated costs
- ……



Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.



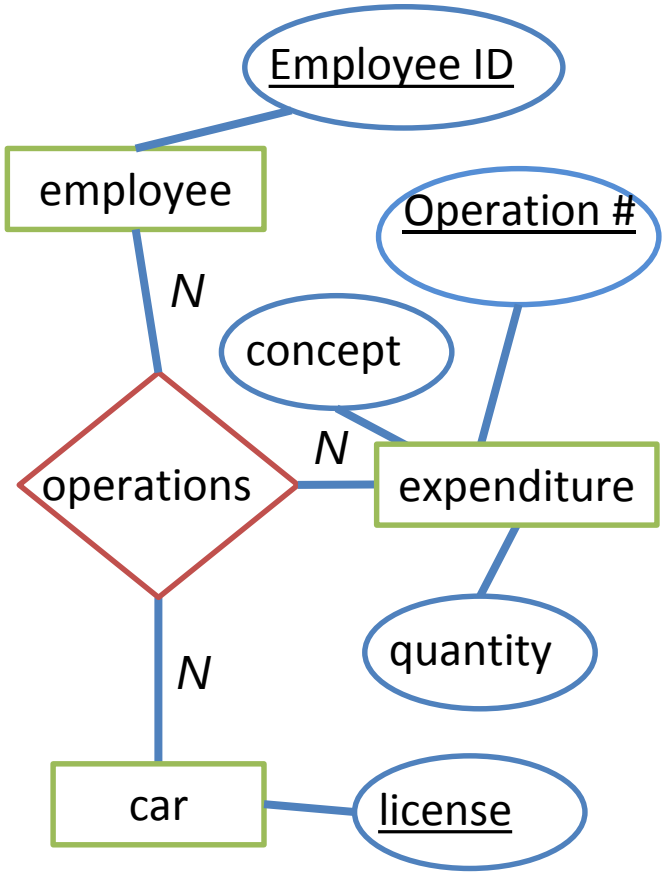
Particularly, each operation:

- ...
- One of workers involved in the operations (the manager) is in charge (before use) to reserve cars (among those available), and once the operation has been completed, delivering the sheet with all the costs

Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.



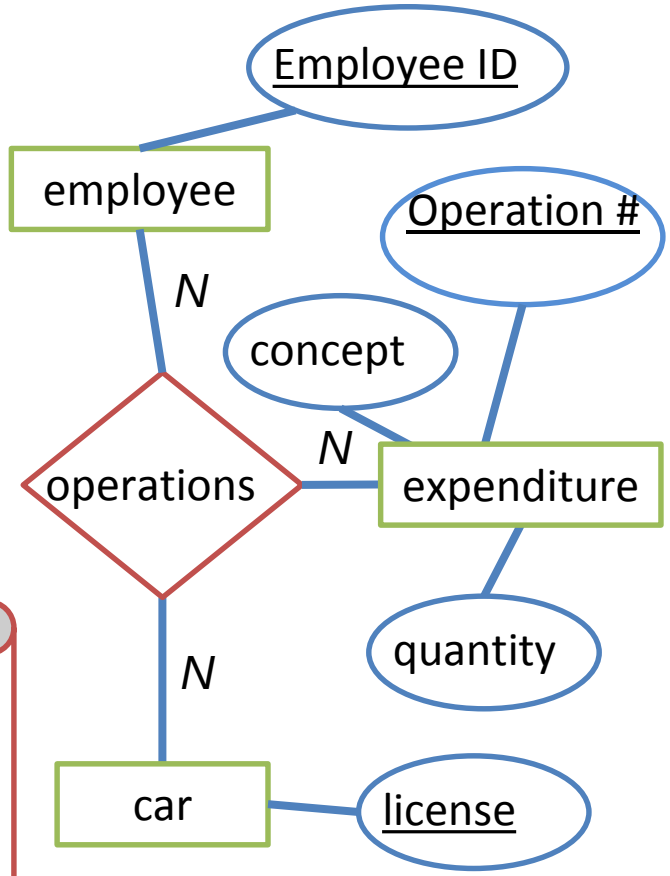
Problems:

- The ternary relationship does not allow booking first and then **to pass** expenses

Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

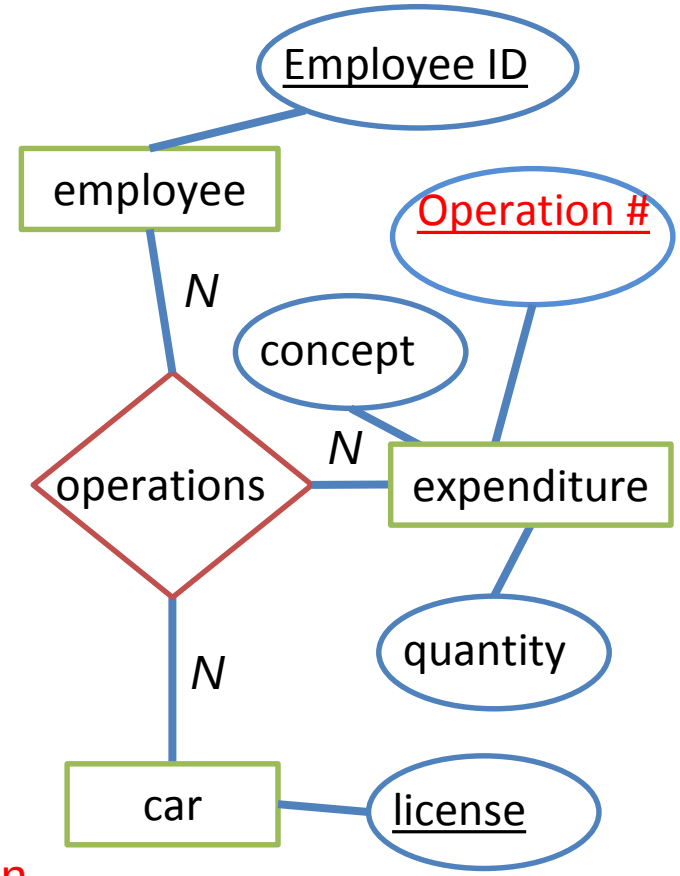


In this case should be allowed operations between **employee and car** before issuing costs (requires to be independent "operations" of "spending").

Aggregation

Example 3: business operations

A company has a fleet of cars for employees for perform **operations** outside their workplace. In addition, each time an employee uses a car generates a sheet with all the associated costs.

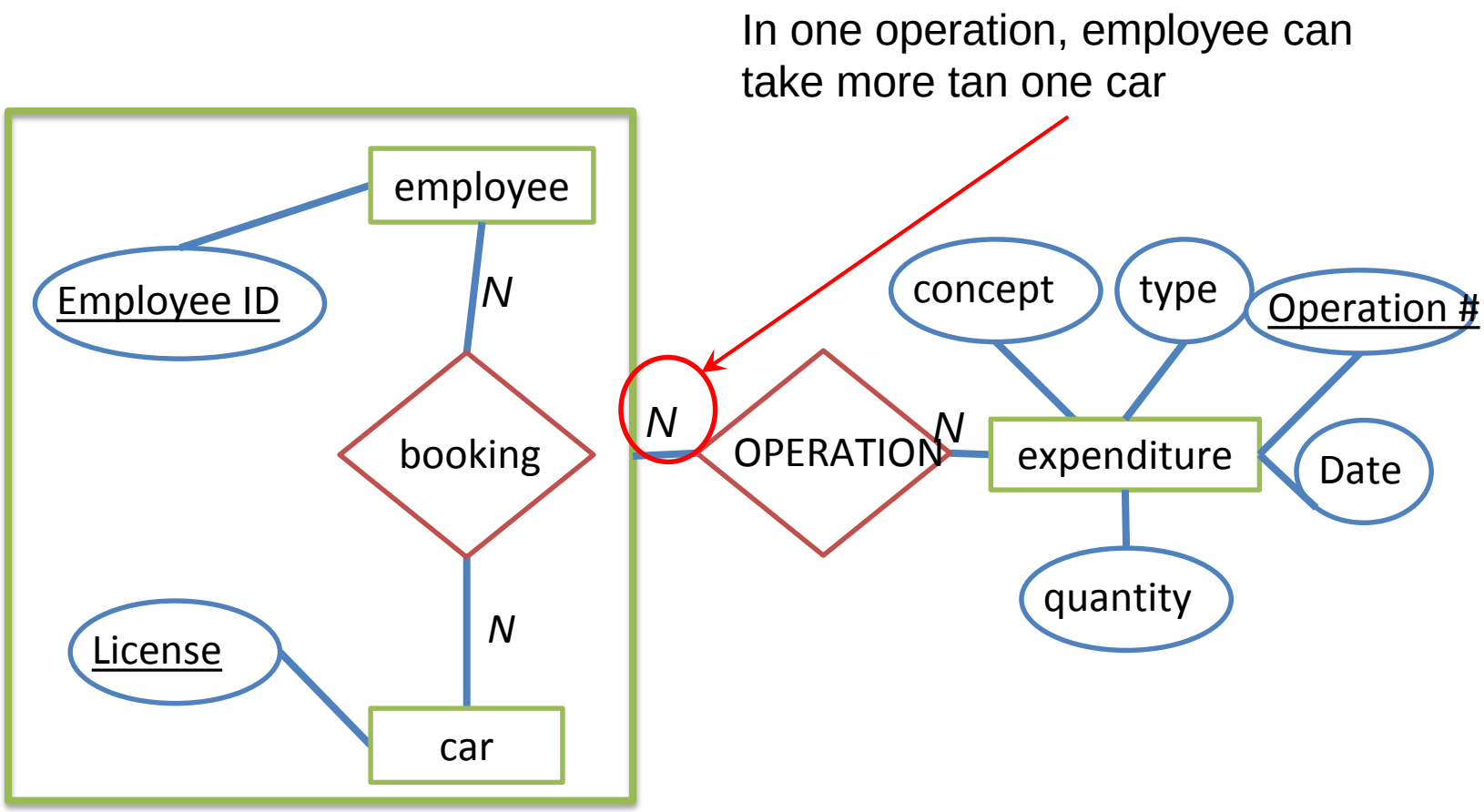


Problems:

- The ternary relationship does not allow booking first and then **to pass** expenses
- Costs associated with a real operation correspond to the same number of operation (Operation #) are not checked

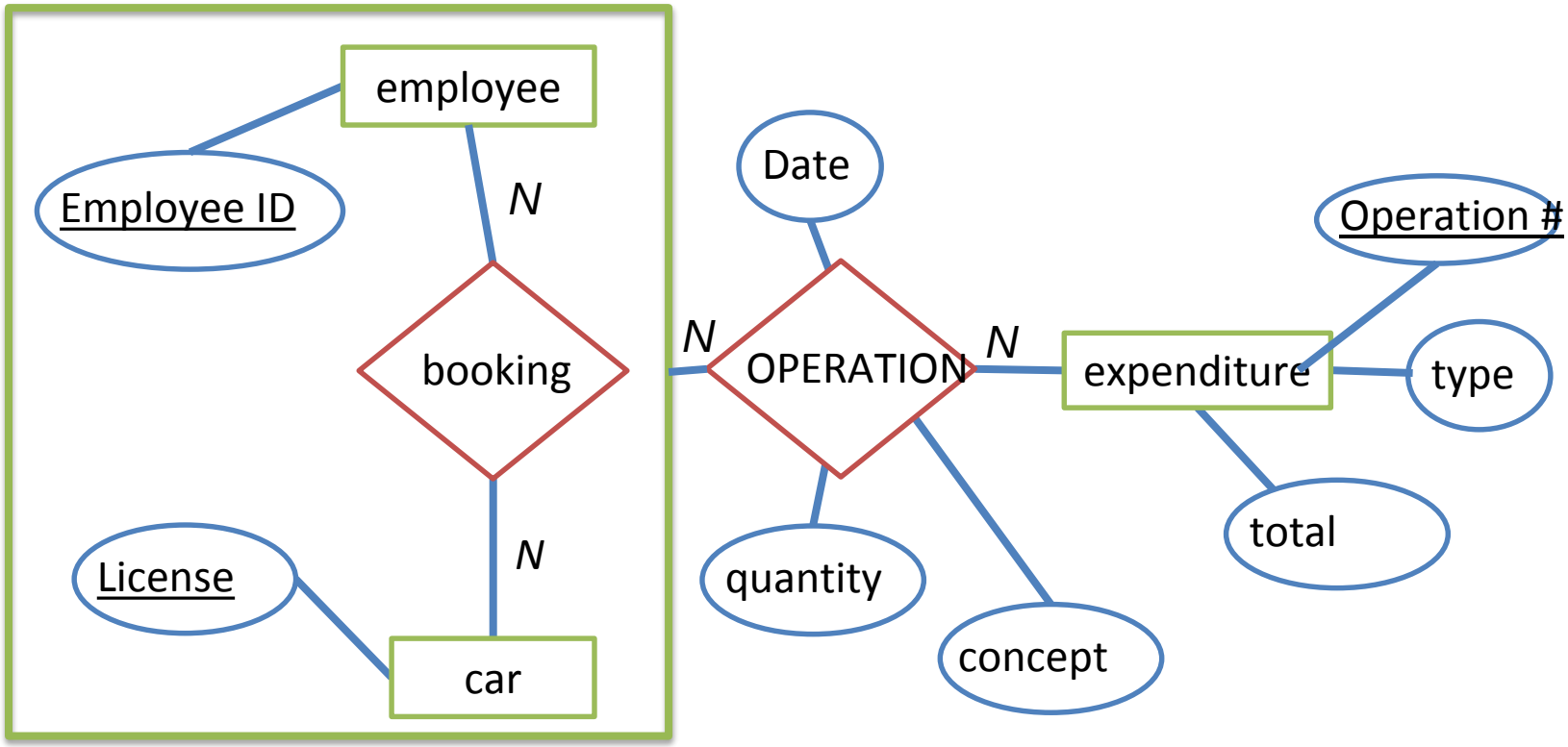
Aggregation

Example 3: business operations



Aggregation

Example 3: business operations

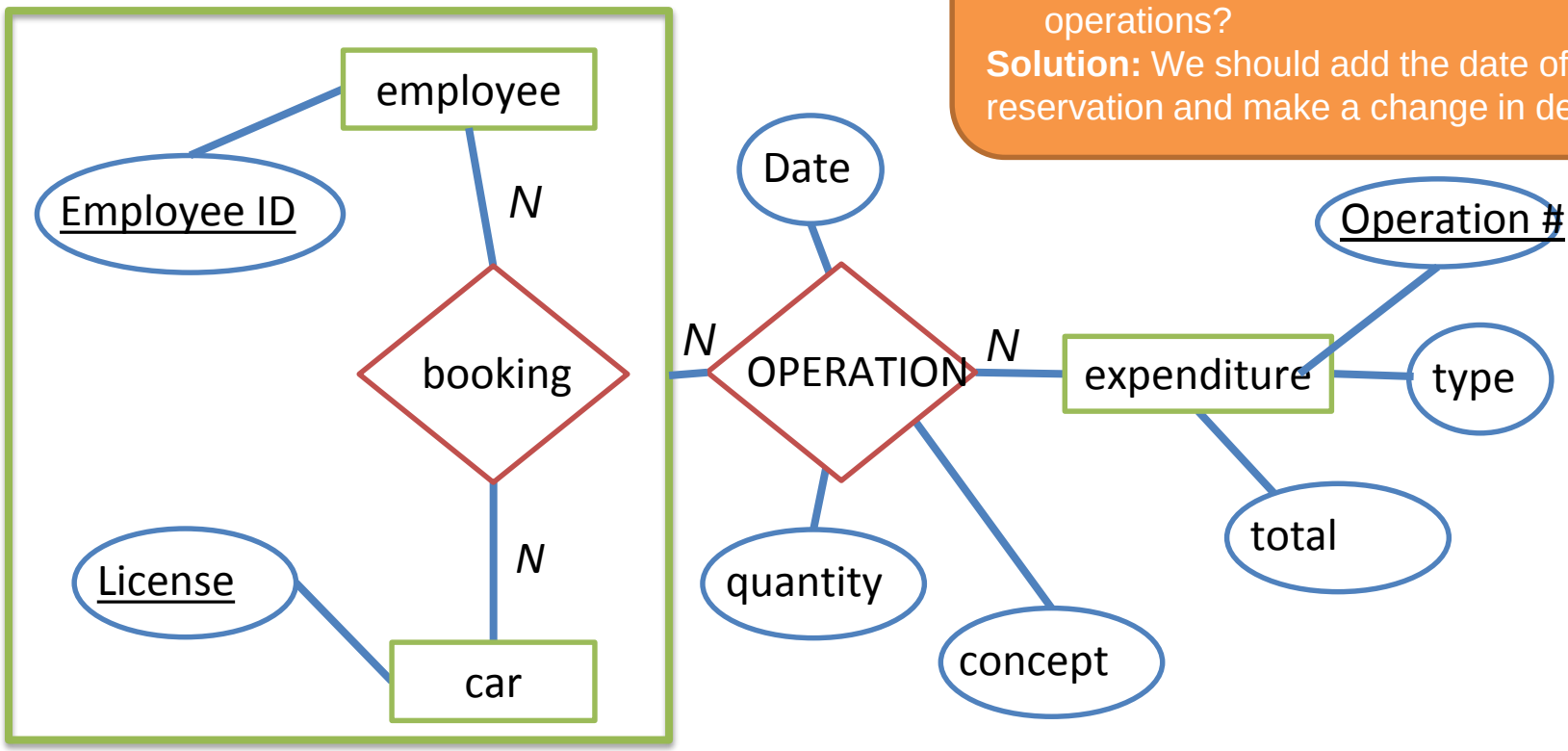


Aggregation

Example 3: business operations

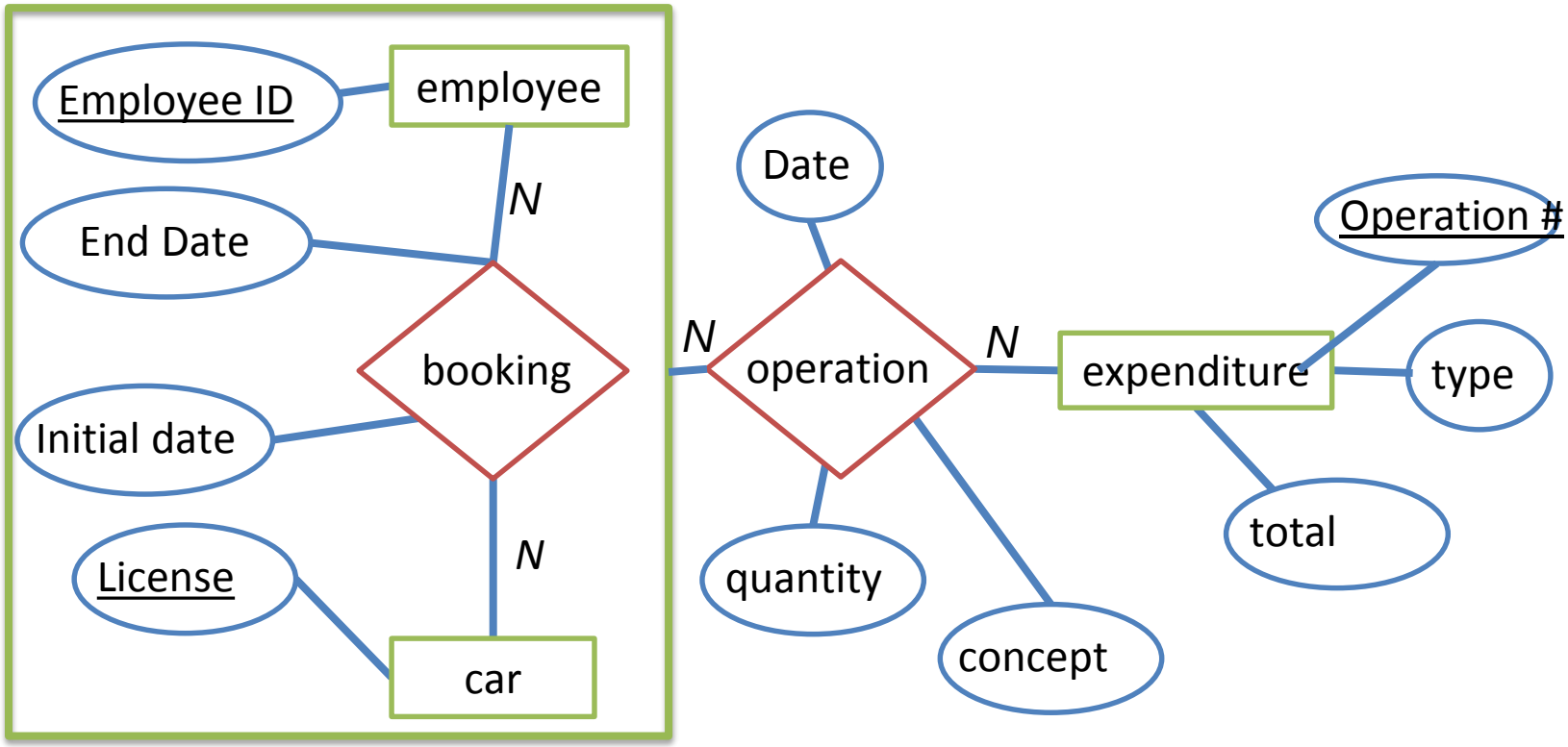
This design still has problems ...
- What happens when an employee takes the same car for different operations?

Solution: We should add the date of reservation and make a change in design.



Aggregation

Example 3: business operations (Solution 1)



Aggregation

Example 3: business operations (Solution 2)