Database design

Logical Design

DBMG

Logical Design (1/2)

- Introduction
- Restructuring of the Entity-Relationship schema
- Removing generalizations
- Partitioning of concepts
- Removing multivalued attributes
- Removing composed attributes
- Selection of primary identifiers

Logical Design (2/2)

- Translation into the relational model
  - entity and many-to-many relationships
  - one-to-many relationships
  - one-to-one relationships
  - entity with external identifier
  - ternary relationships

Logical Design

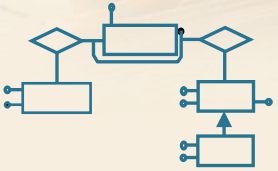
Introduction

Logical Design

- Select a logical model
  - in our case, the relational model
- Goal
  - build a relational schema that correctly and efficiently represents all the information described by the ER schema
- Not just a simple translation
  - simplification of the scheme to make it compatible with the relational model
  - optimization to increase the efficiency of queries

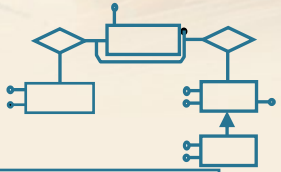
Logical design steps

ER Schema



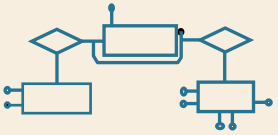
Logical design steps

ER Schema



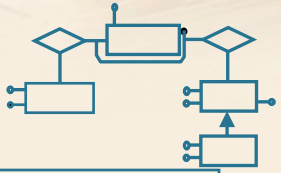
Restructuring the ER Schema

Restructured ER schema



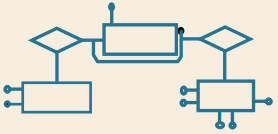
Logical design steps

Schema ER



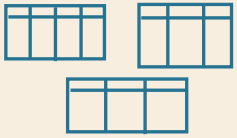
Restructuring the ER Schema

Restructured ER schema



Translation

Logical schema



Relational logical design

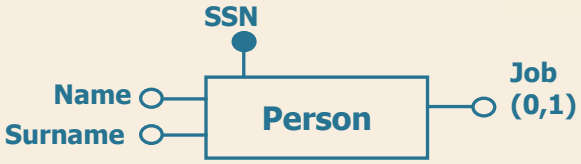
Translation in relational model: entities and many to many relationships

DBMG

Translation to the relational model

- It is executed on the restructured ER schema
  - i.e., the schema without hierarchies, multivalued attributes and compounds attributes
- Transformations
  - Each entity is translated into a table with the same attributes
  - For relations we need to consider the maximum cardinality

Entities translation



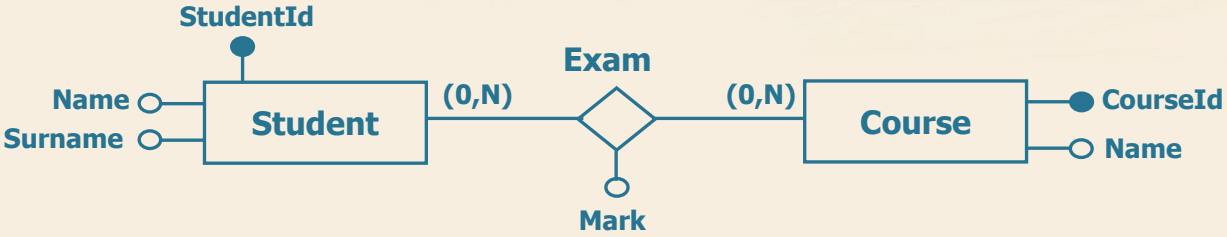
Person(SSN, Name, Surname, Job*)

- Primary key underlined
- Optional attributes indicated by * (asterisk)

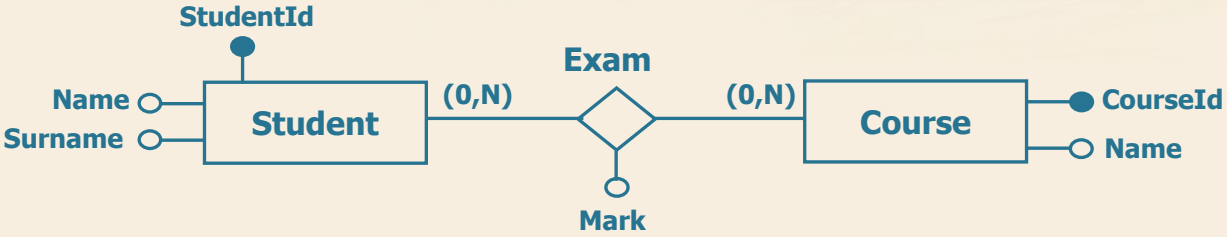
Translation: many to many binary relations

- Each many to many relationship is translated into a table
  - The primary key is the combination of the identifiers of all the linked entities
  - It is possible to rename the attributes of the table that corresponds to the relation (needed in case of recursive relations)

Many to many binary relationship

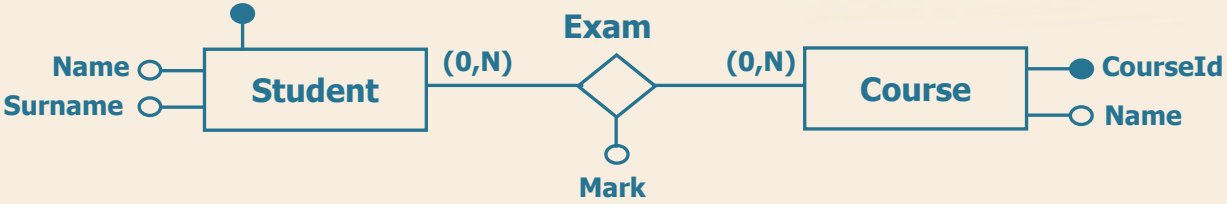


Many to many binary relationship: entity



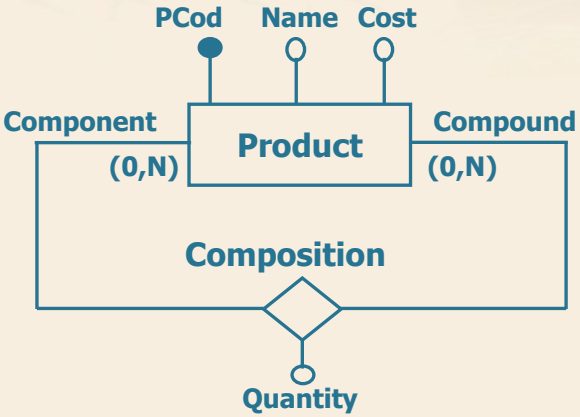
Student(<u>StudentId</u>, Name, Surname)
Course(<u>CourseId</u>, Name)

Many to many binary relationship

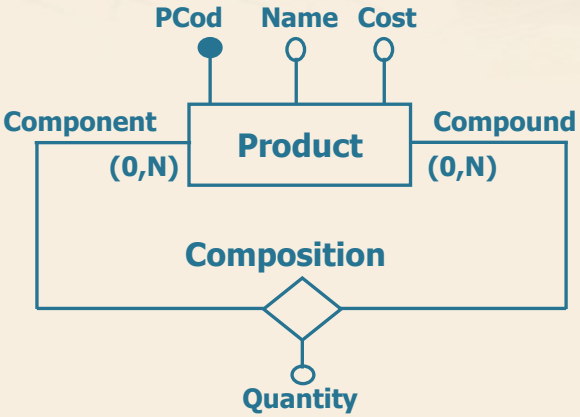


Student(StudentId, Name, Surname)
Course(CourseId, Name)
Exam(StudentId, CourseId, Mark)

Recursive many to many binary relationship

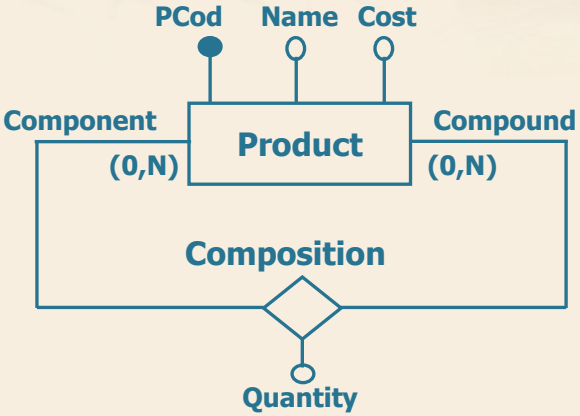


Recursive many to many binary relationship



Product(PCod, Name, Cost)

Recursive many to many binary relationship



Product(PCod, Name, Cost)
Composition(CompoundCod, ComponentCod, Quantity)

Relational logical design

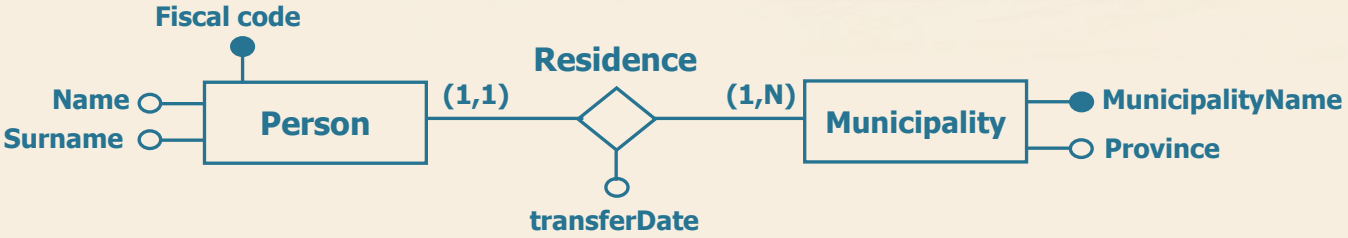
Translation to the relational model: one to many relationship

DBMG

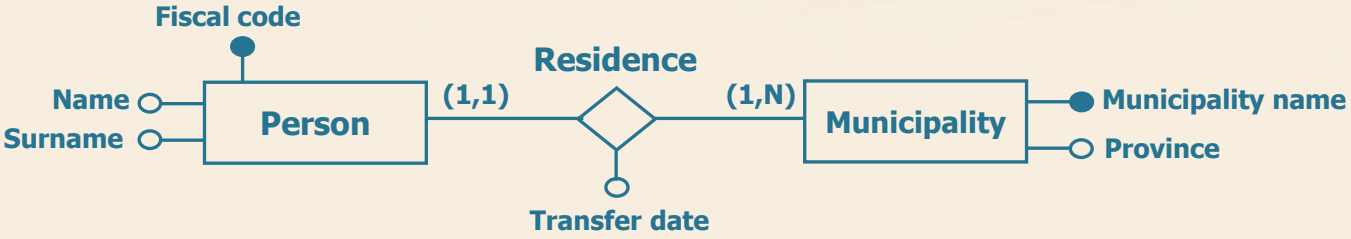
One to many binary relationships

- Two alternative ways to translate them:
  - By means of attributes
  - By means of a new table

One to many binary relationship



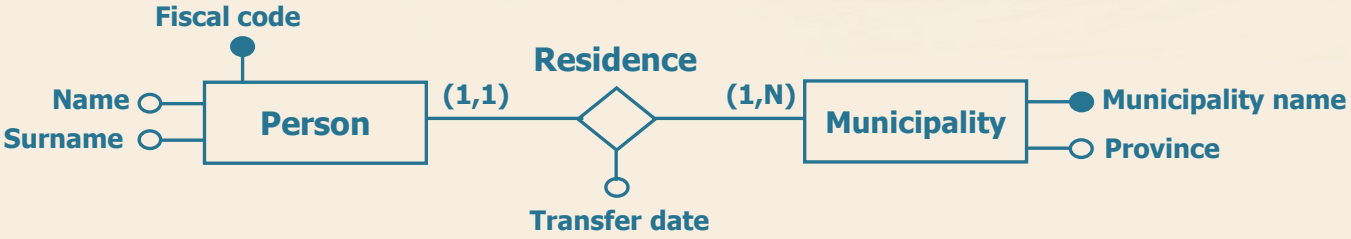
One to many binary relationship: entity



Person(FiscalCode, Name, Surname)

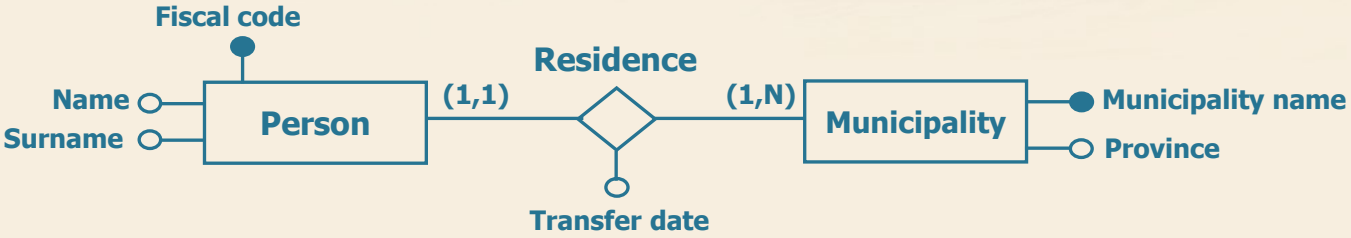
Municipality(MunicipalityName, Province)

One to many binary relationship



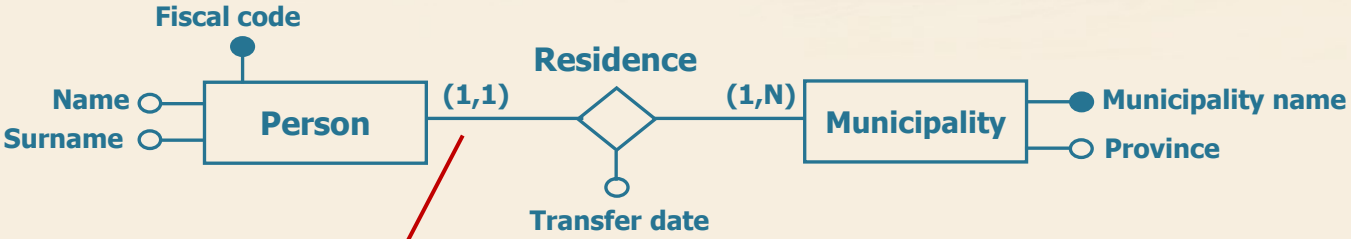
Person(<u>FiscalCode</u>, Name, Surname,
MunicipalityName)
Municipality(<u>MunicipalityName</u>, Province)

One to many binary relationship



Person(<u>FiscalCode</u>, Name, Surname,
MunicipalityName, *TransferDate*)
Municipality(<u>MunicipalityName</u>, Province)

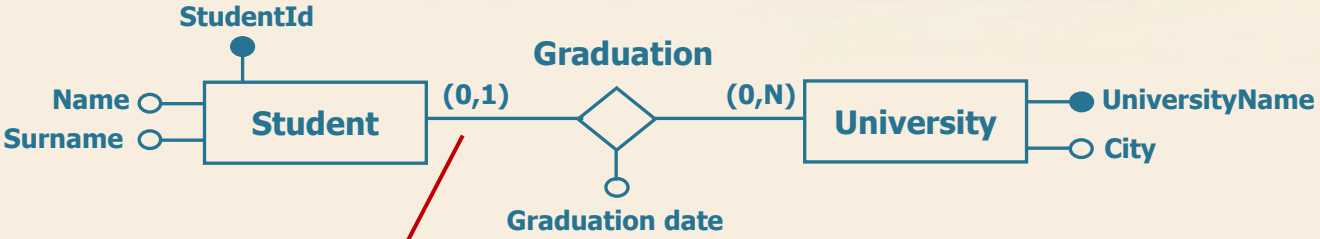
One to many binary relationship



Can be used when the cardinality is (1,1)

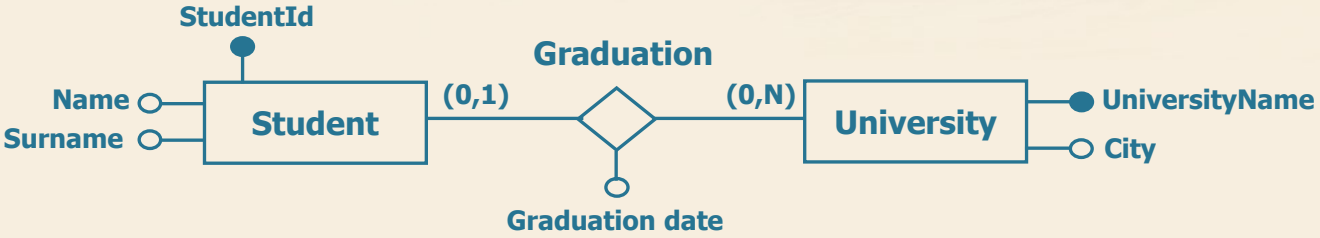
Person(FiscalCode, Name, Surname, MunicipalityName, *TransferDate*)
Municipality(MunicipalityName, Province)

One to many binary relationship



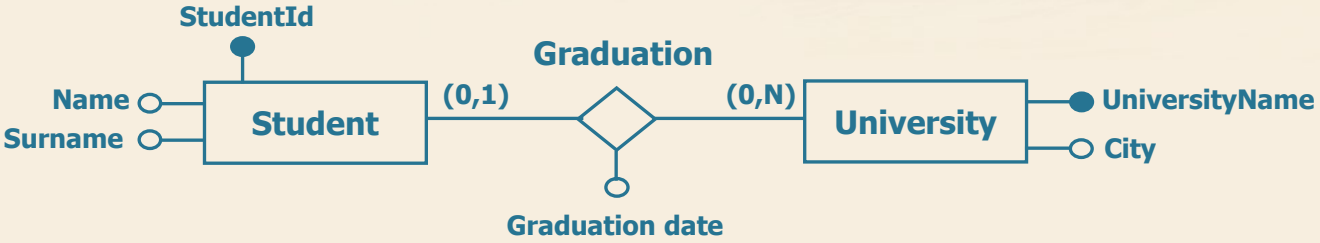
When the cardinality is (0,1), two alternative representations are possible...

Alternative n.1: new table



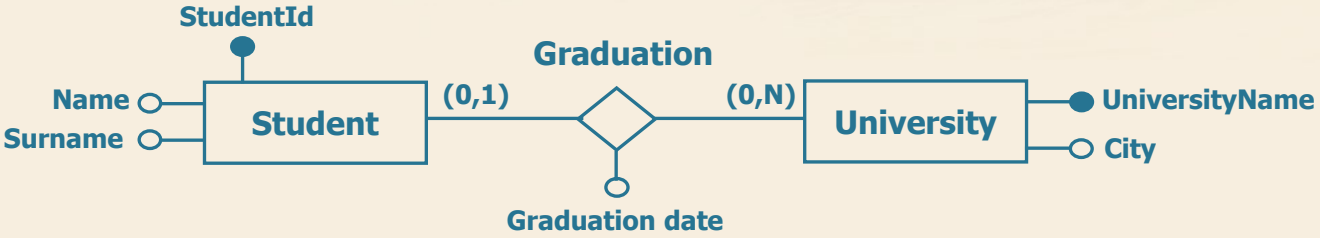
Student(<u>StudentId</u>, Name, Surname)
University(<u>UniversityName</u>, City)

Alternative n.1: new table



Student(StudentiId, Name, Surname)
University(UniversityName, City)
Graduation(StudentId, UniversityName, GraduationDate)

Alternative n.2: attributes



Student(<u>StudentId</u>, Name, Surname, FacultyName*, GraduationDate*)
University(<u>UniversityName</u>, City)

Relational logical design

Translation to the relational model: one to one relationships

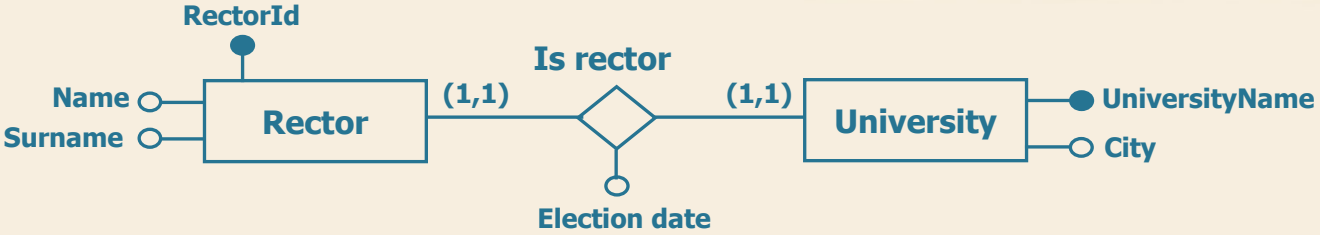
DBMG

One to one binary relationships

- Different translations are possible
  - Depending on the minimum cardinality value

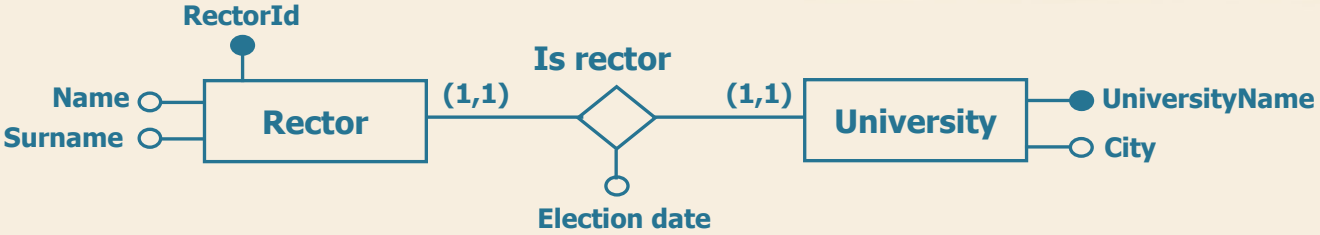
One to one binary relationship: case 1

- Mandatory partecipation from both sides



One to one binary relationship:alternative n.1

- Mandatory partecipation from both sides

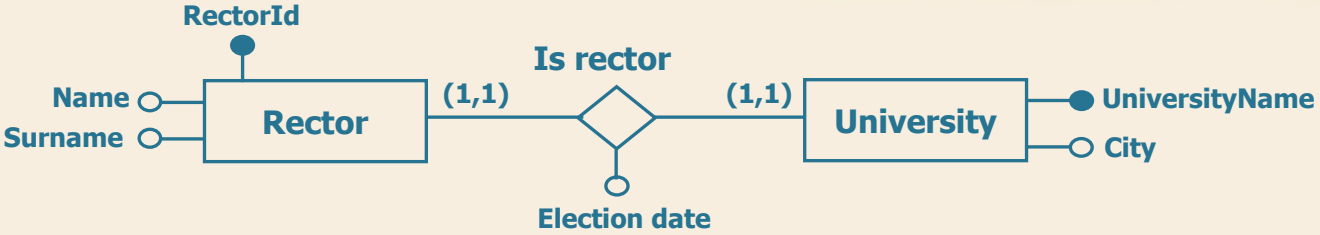


Rector(<u>RectorId</u>, Name, Surname)

University(<u>UniversityName</u>, City)

One to one binary relationship:alternative n.1

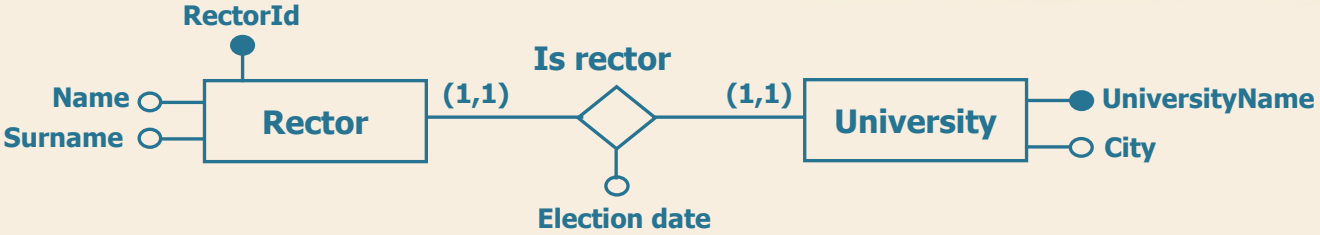
- Mandatory partecipation from both sides



Rector(<u>RectorId</u>, Name, Surname, *UniversityName, ElectionDate*)
University(<u>UniversityName</u>, City)

One to one binary relationship:alternative n.2

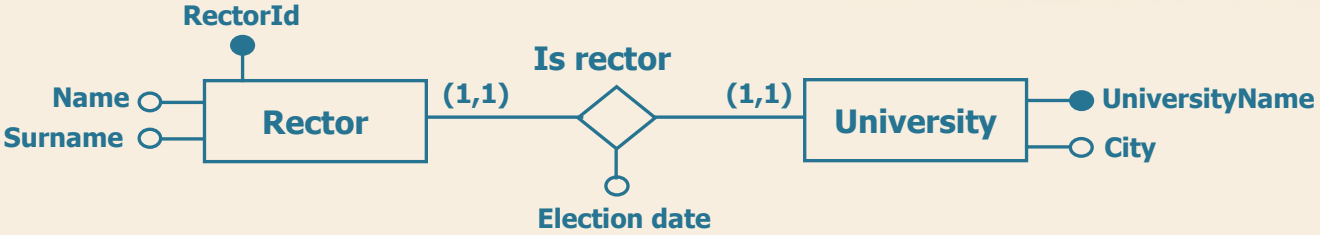
- Mandatory partecipation from both sides



Rector(<u>RectorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)

One to one binary relationship:alternative n.2

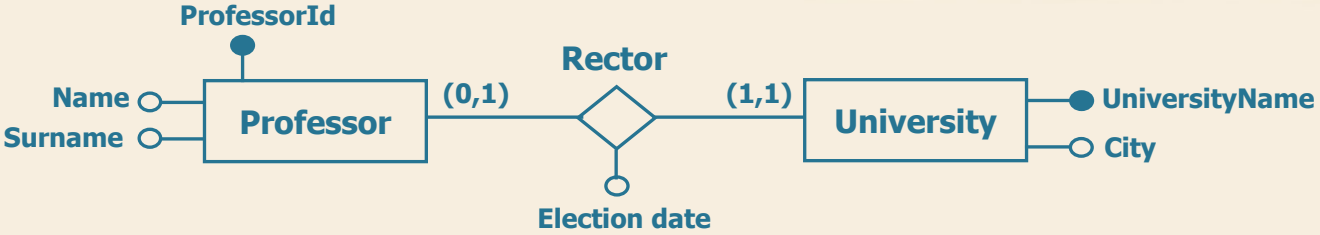
- Mandatory partecipation from both sides



Rector(<u>RectorId</u>, Name, Surname)
University(<u>UniversityName</u>, City, *RectorId*, *ElectionDate*)

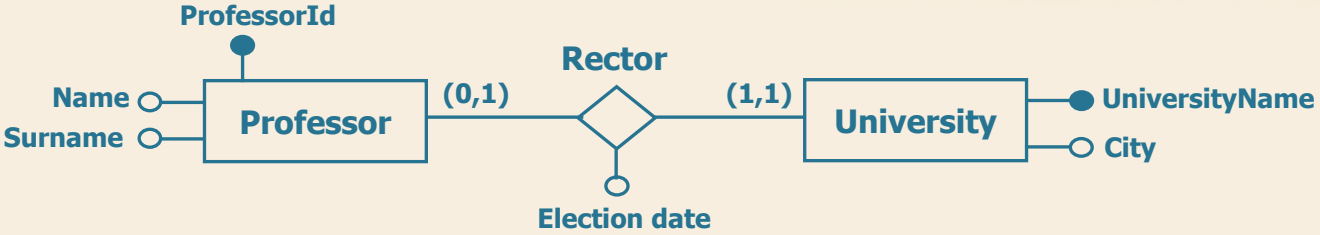
One to one binary relationship: case 2

- Optional partecipation on one side



One to one binary relationship: entity

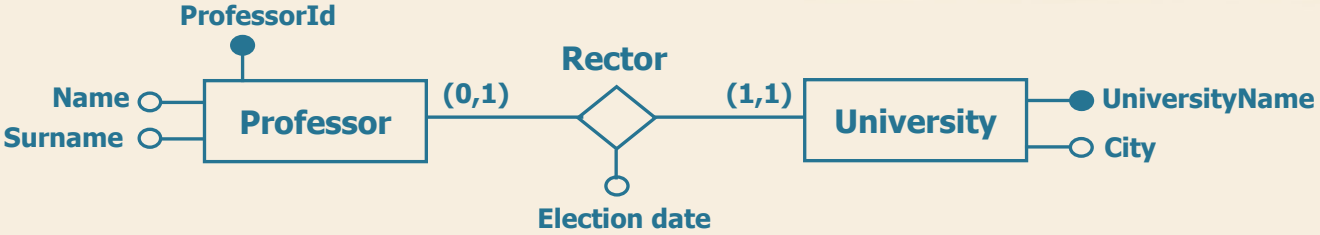
- Optional partecipation on one side



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)

One to one binary relationship

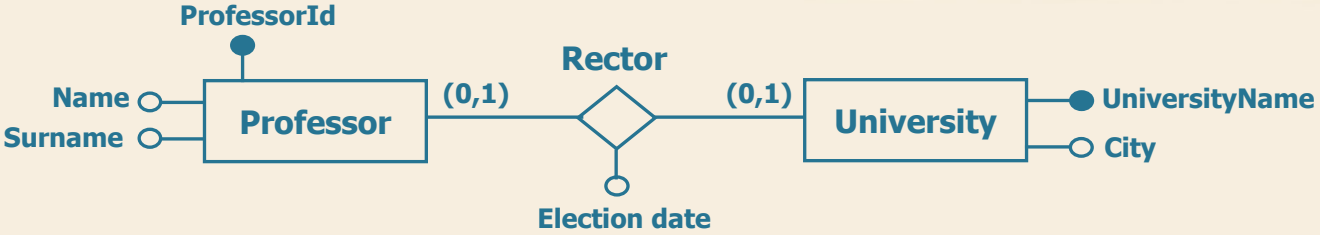
- Optional partecipation on one side



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City, *ProfessorId*, *ElectionDate*)

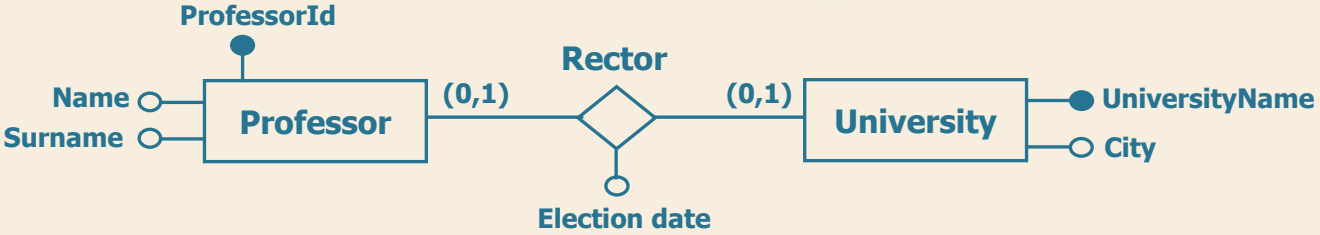
One to one binary relationship: case 3

- Optional partecipation from both sides



One to one binary relationship:alternative n.1

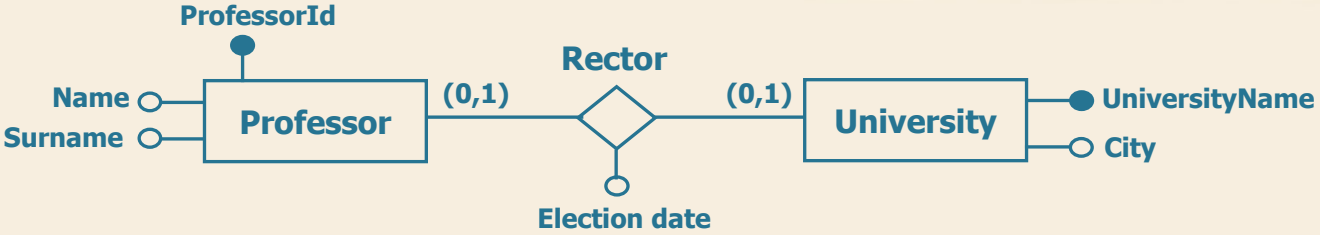
- Optional partecipation from both sides



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)

One to one binary relationship:alternative n.1

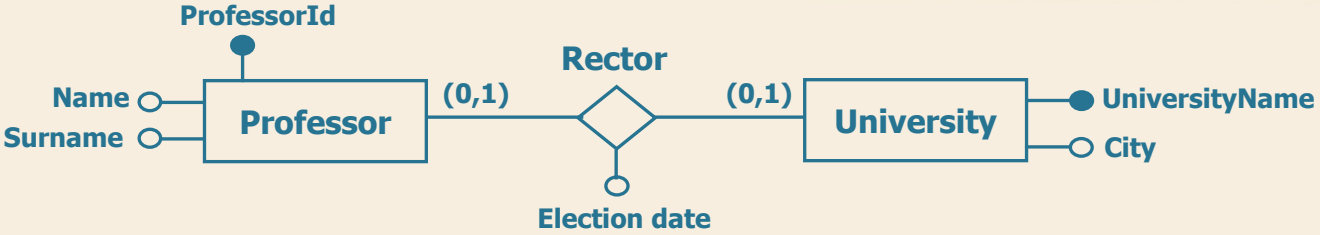
- Optional partecipation from both sides



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)
Rector(<u>ProfessorId</u>, UniversityName, ElectionDate)

One to one binary relationship:alternative n.2

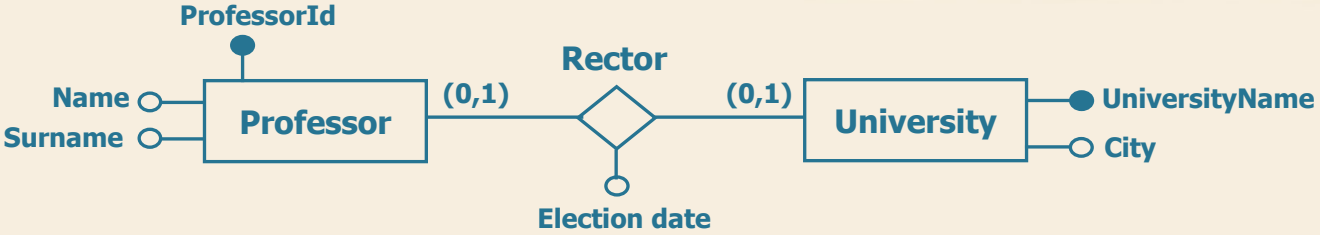
- Optional partecipation from both sides



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)
Rector(ProfessorId, <u>UniversityName</u>, ElectionDate)

One to one binary relationship:alternative n.3

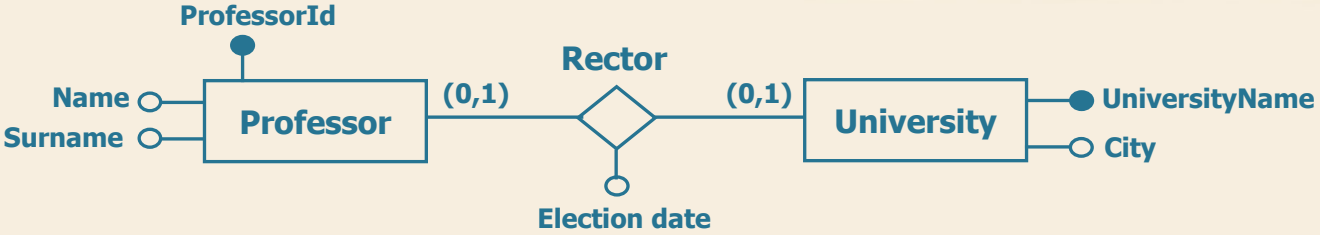
- Optional partecipation from both sides



Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>UniversityName</u>, City)

One to one binary relationship:alternative n.3

- Optional partecipation from both sides



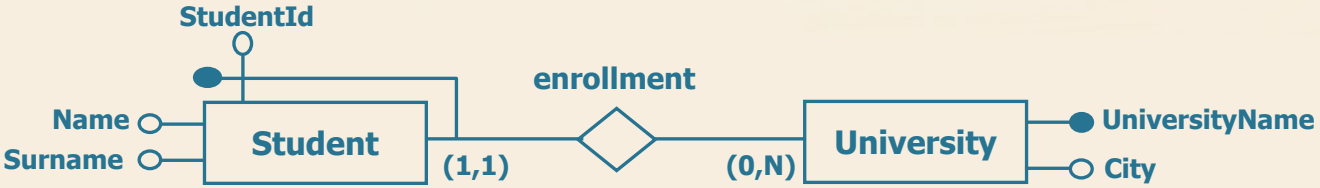
Professor(<u>ProfessorId</u>, Name, Surname)
University(<u>Name</u>, City, *ProfessorId**, *ElectionDate**)

Relational logical design

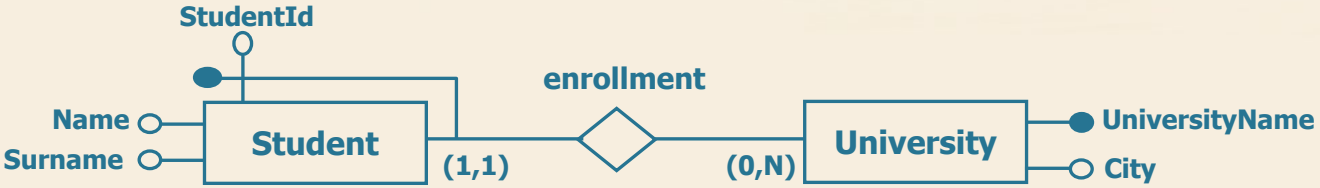
Translation to the relational model: entity with external identifier

DBMG

Entity with external identifier

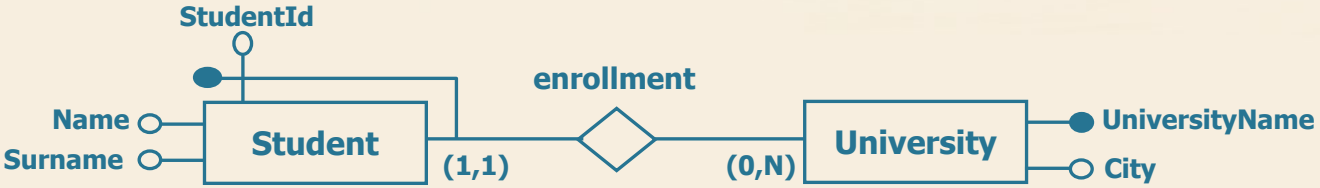


Entity with external identifier



University(<u>UniversityName</u>, City)
Student(<u>*StudentId*</u>, <u>*UniversityName*</u>, Name, Surname)

Entity with external identifier



University(<u>UniversityName</u>, City)
Student (<u>StudentId</u>, <u>UniversityName</u>, Name, Surname)

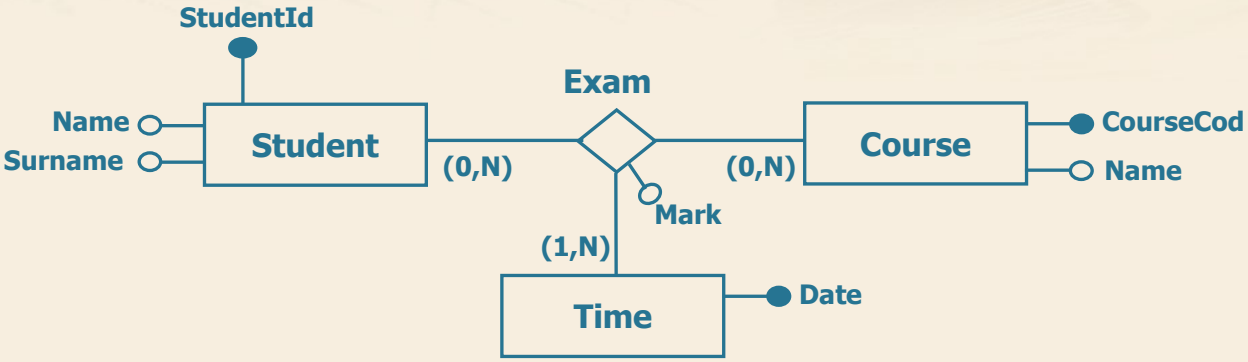
- The relationship is represented along with the external identifier

Relational logical design

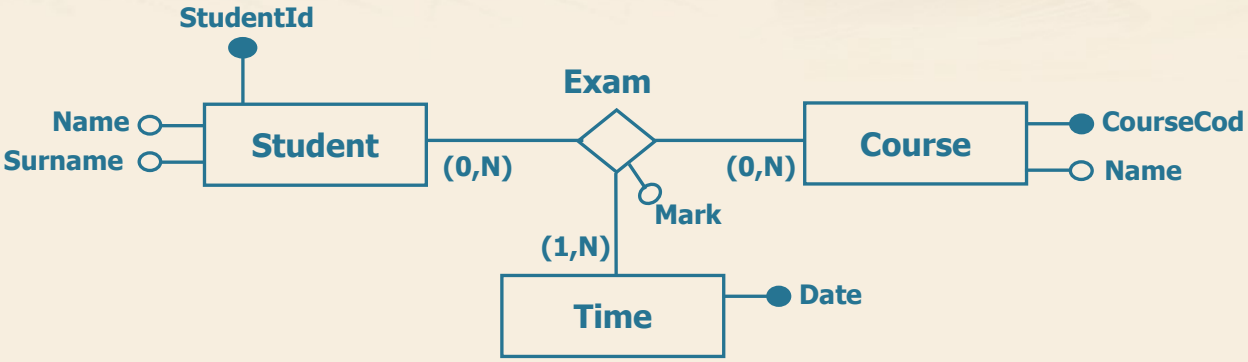
Translation to the relational model: ternary relationships

DBMG

Ternary relationship

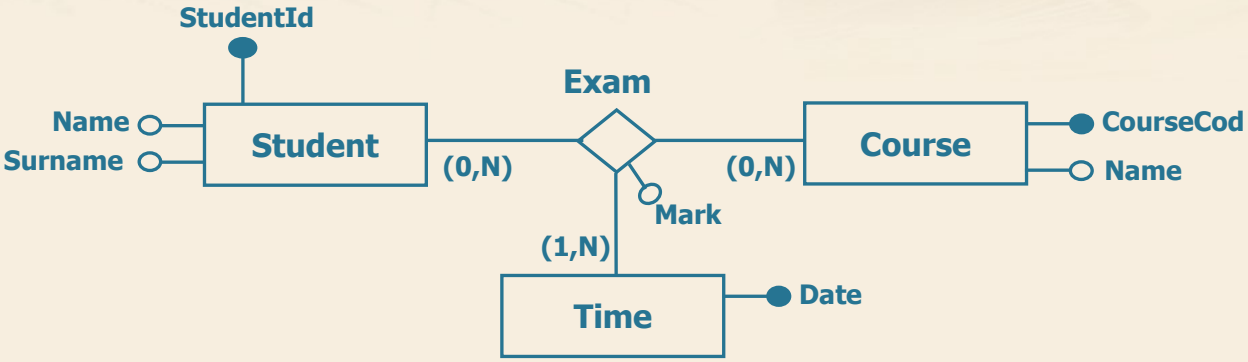


Ternary relationship: entity



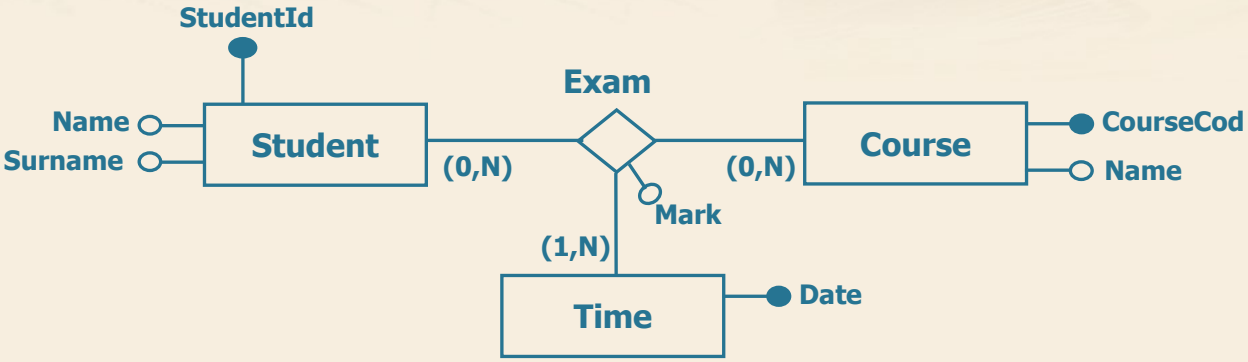
Student(<u>StudentId</u>, Name, Surname)
Course(<u>CourseCod</u>, Name)
Time(<u>Date</u>)

Ternary relationship: identificator



Student(<u>StudentId</u>, Name, Surname)
Course(<u>CourseCod</u>, Name)
Time(<u>Date</u>)
Exam(<u>StudentId, CourseCod, Date</u>)

Ternary relationship: attributes



Student(<u>StudentId</u>, Name, Surname)
Course(<u>CourseCod</u>, Name)
Time(<u>Date</u>)
Exam(<u>StudentId, CourseCod, Date</u>, Mark)

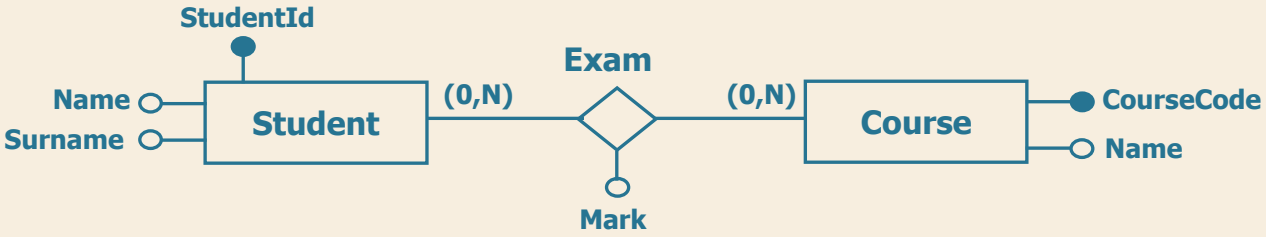
Relational logical design

Referential integrity constraints

DBMG

Referential integrity constraints

- Relationships represent referential constraints



Referential integrity: exam relationship

- Tables

Student(StudentId, Name, Surname)

Course(CourseCod, Name)

Exam(*StudentId*, CourseCod, Mark)

- Referential integrity constraints

Exam(StudentId) REFERENCES Student(StudentId)

Referential integrity: exam relationship

- Tables

Student(StudentId, Name, Surname)

Course(CourseCod, Name)

Exam(StudentId, *CourseCode*, Mark)

- Referential integrity constraints

Exam(StudentId) REFERENCES Student(StudentId)

Exam(CourseCod) REFERENCES Course(CourseCod)

Logical Design

Restructuring an ER schema

DBMG

ER schema restructuring

- Implementation aspects
  - This is not a conceptual schema
- Goals
  - Removing costructs for which there is no direct representation in the relational model
  - Optimize data access

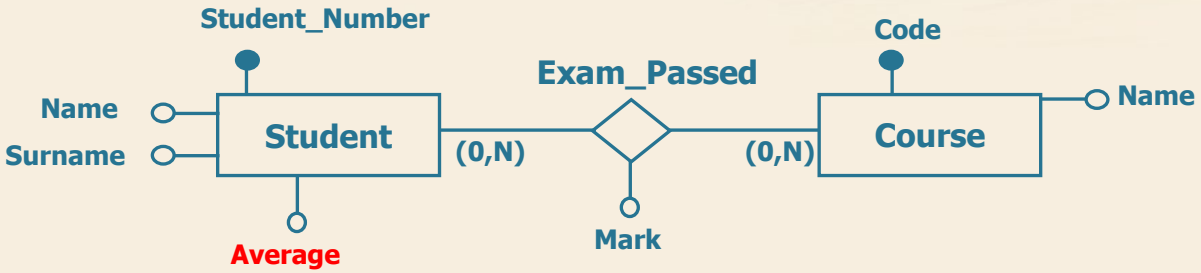
Restructuring tasks

- Analysis of redundancies
- Removing generalizations
- Partitioning and merging of entities and relationships
- Selection of primary identifiers

Analysis of redundancies

- Issue
  - To represent informations that can be derived from other data
  - To decide whether to keep or remove them
- Advantages
  - Speed up and simplify queries
- Disadvantages
  - Increased complexity of updates
  - Slower updates
  - More storage space required

Redundant attribute: example



- In this schema the attribute Average is redundant
  - It is useful for speeding up queries to calculate students' average grade.
  - If kept, the redundancy indication must be added in the relational schema.

Logical Design

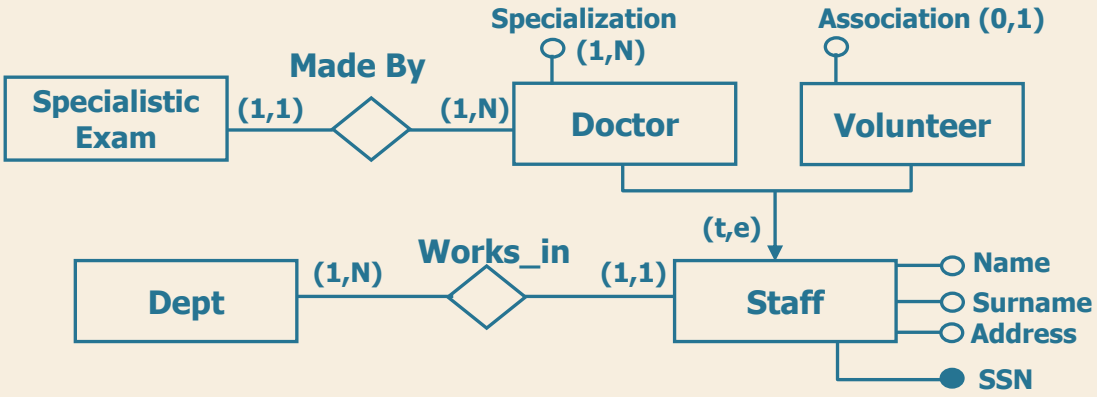
Removing generalizations

DBMG

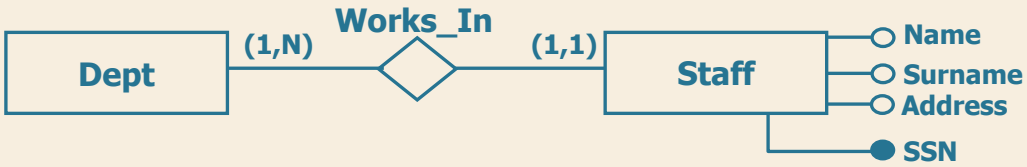
Removing Generalization

- The relational model does not allow direct representation of generalizations of the ER model
- We need, therefore, to trasform these into entities and relationships
- Possible methods:
  - Child entities merged into parent entity
  - Parent entity merged into child entities
  - Generalization translated into relatioships

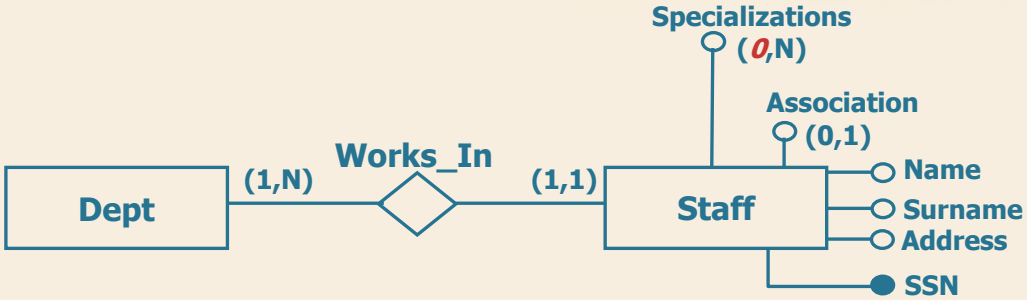
Example


Specialization
(1,N)
Association (0,1)
Made By
Specialistic Exam
(1,1)
(1,N)
Doctor
Volunteer
(t,e)
Works_in
Dept
(1,N)
(1,1)
Staff
Name
Surname
Address
SSN

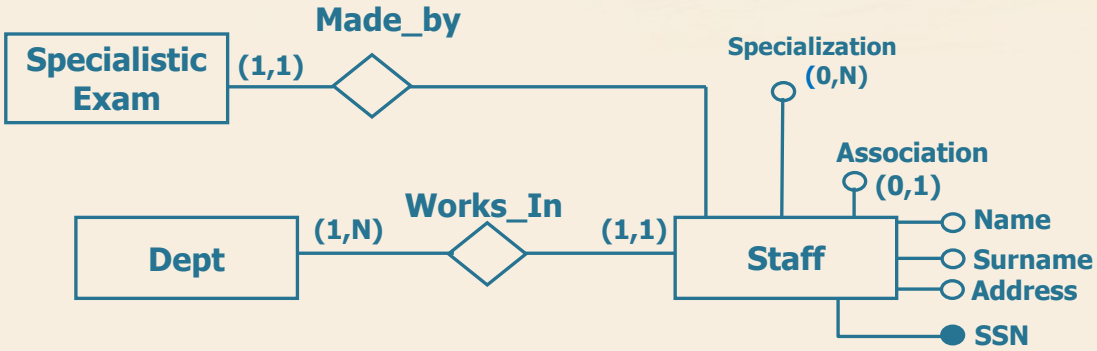
Child->Parent



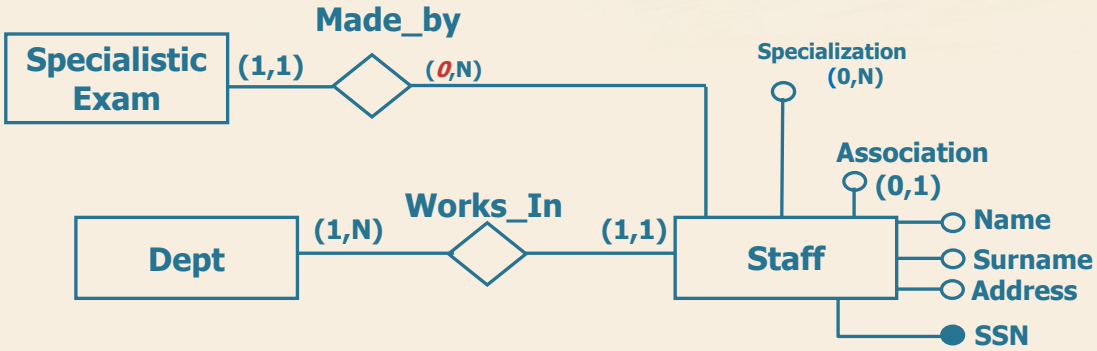
Child entities' attributes



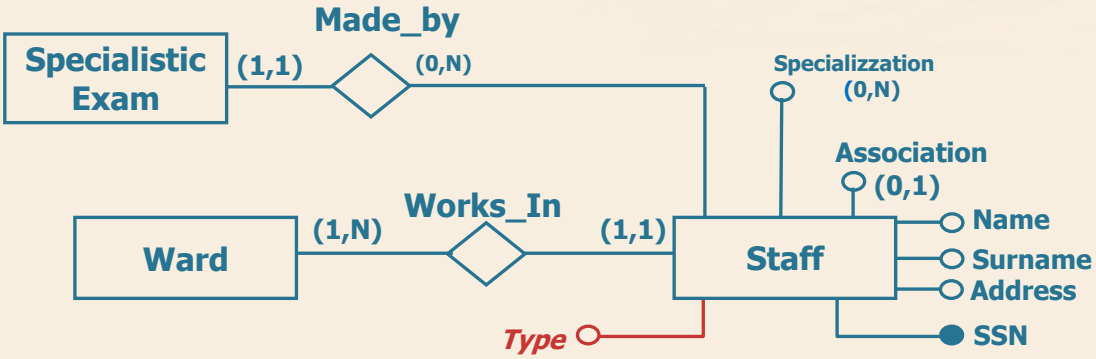
Relations with child entities



Relations with child entities

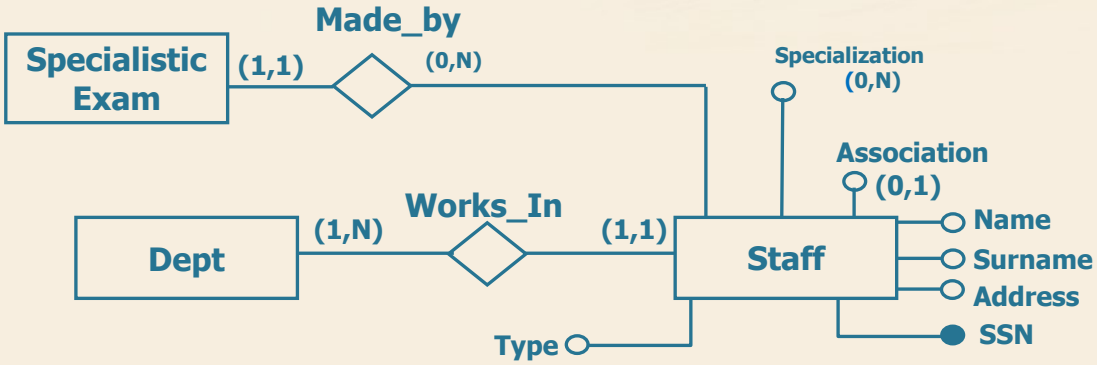


The «Type» attribute



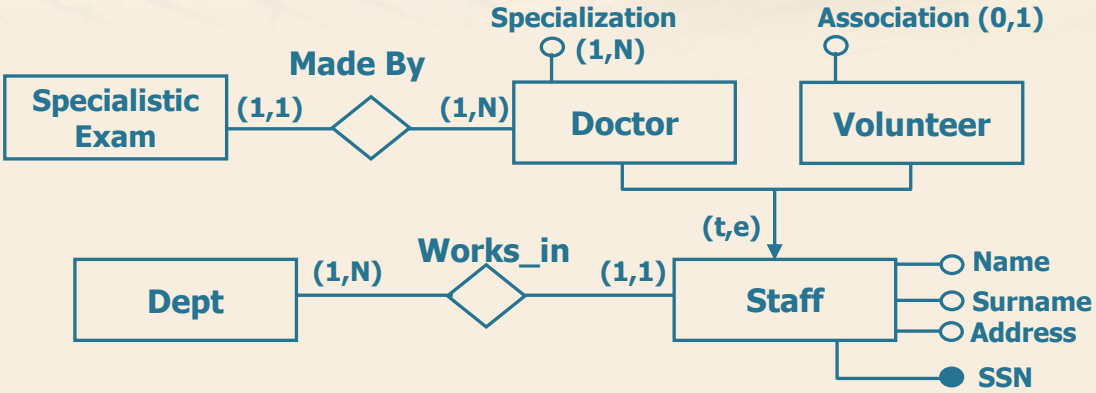
- «Type» indicates the original entity: doctor or volunteer

Child->Parent

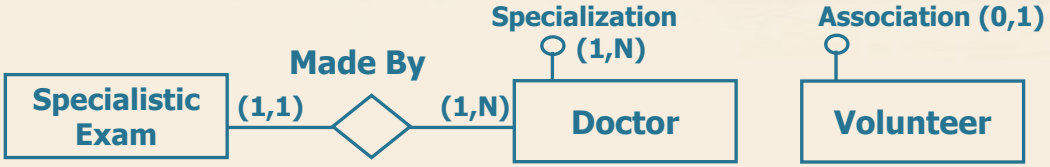


- Can be used for all type of generalization
  - in case of overlapping entities, many combinations are possible as Type values, e.g., skier and sailor

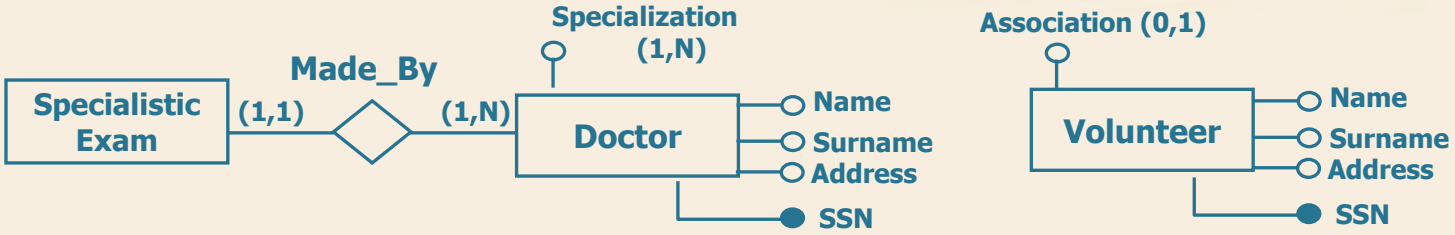
Example


Specialization
(1,N)
Association (0,1)
Made By
Specialistic Exam
(1,1)
(1,N)
Doctor
Volunteer
(t,e)
Dept
(1,N)
Works_in
(1,1)
Staff
Name
Surname
Address
SSN

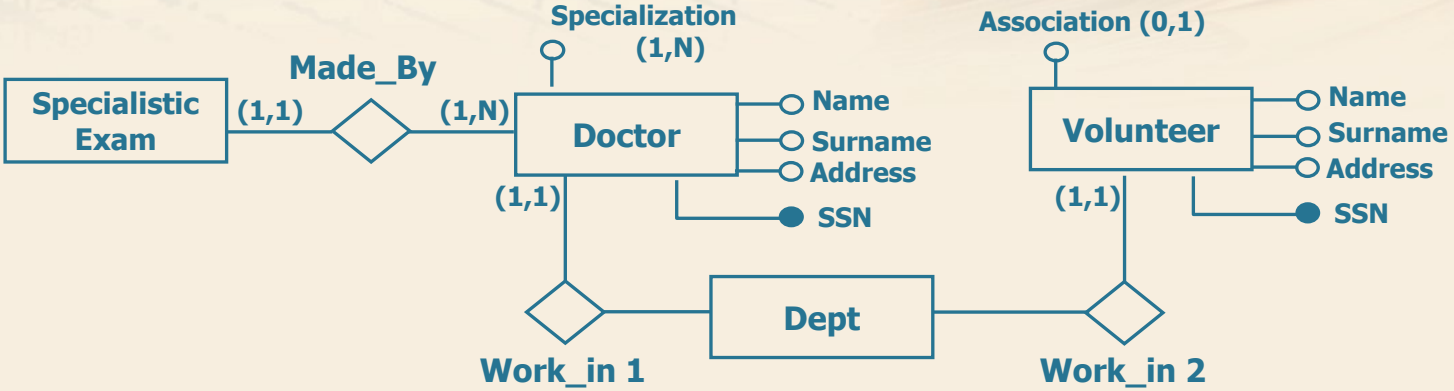
Parent->Child



Parent's attributes

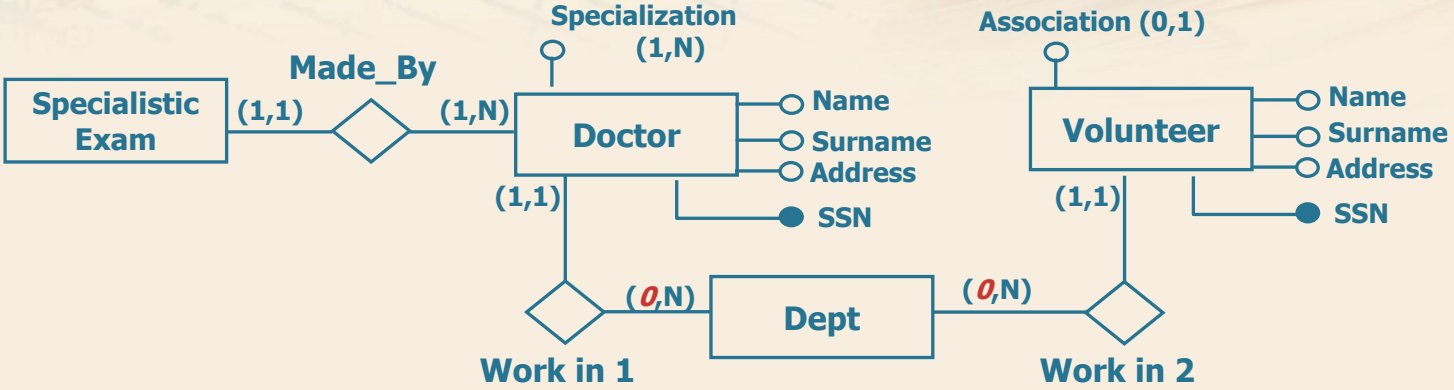


Relationships with parent

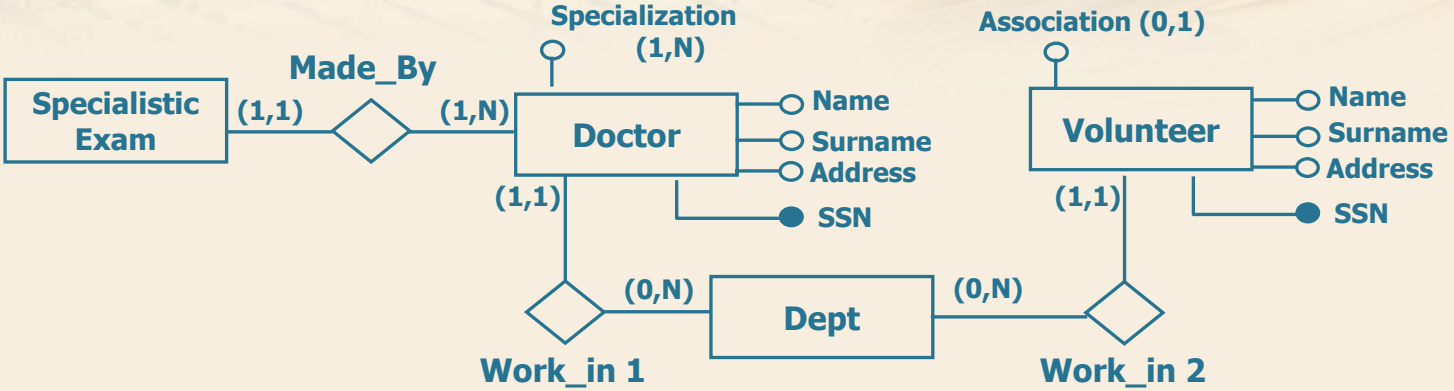


- Relationships with the parent entity need to be split

Cardinality of «work in»

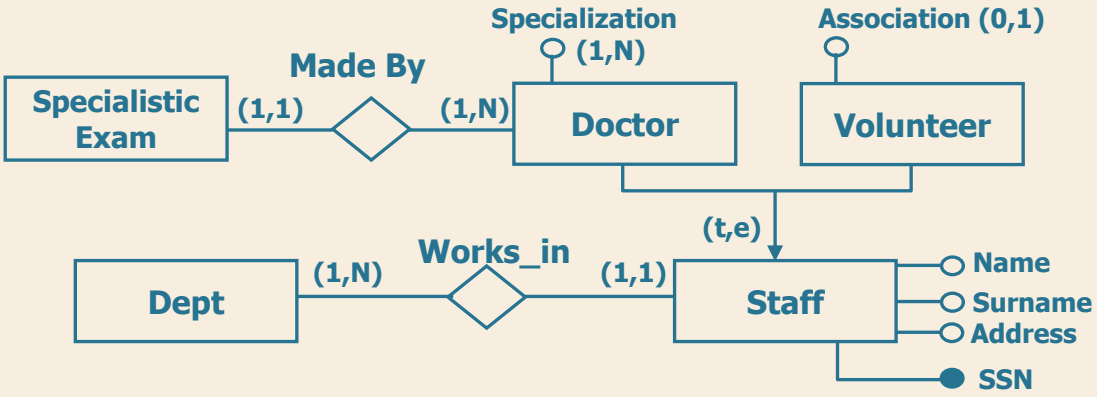


Parent -> Child

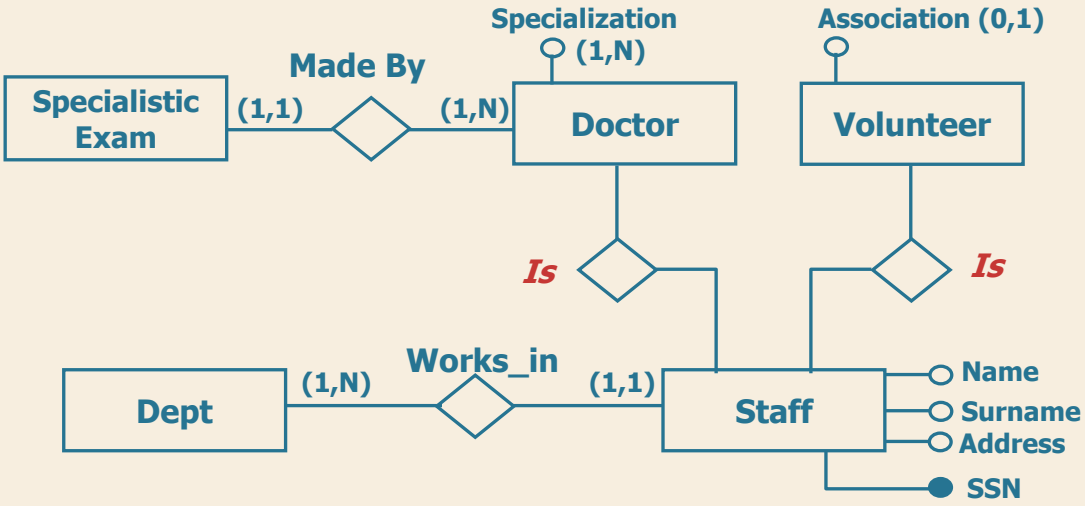


- Cannot be used for **partial** generalization
  - However, we can trasform generalizations from partial to total by adding a new entity called «Others»
- Cannot be used for **overlapping** generalization
  - Due to duplicate identifiers

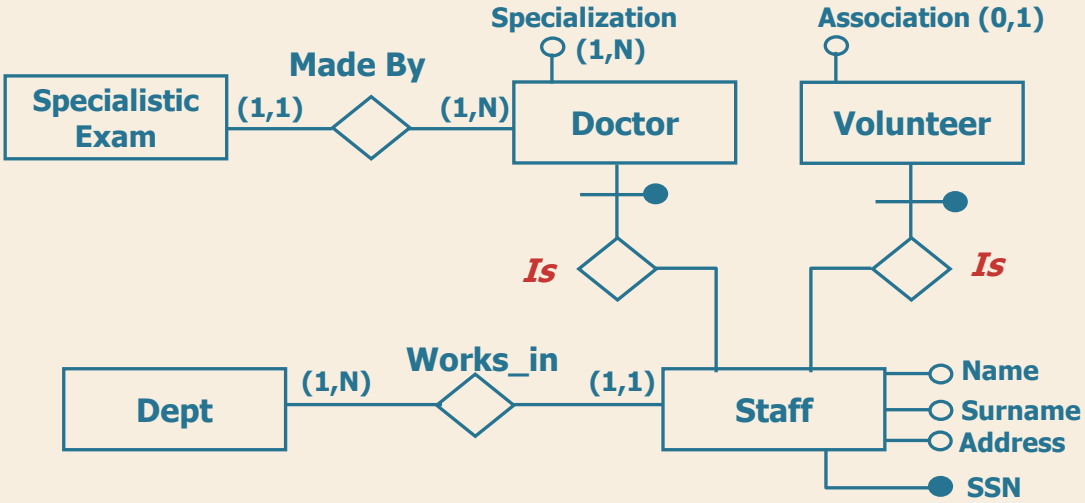
Back to the original example



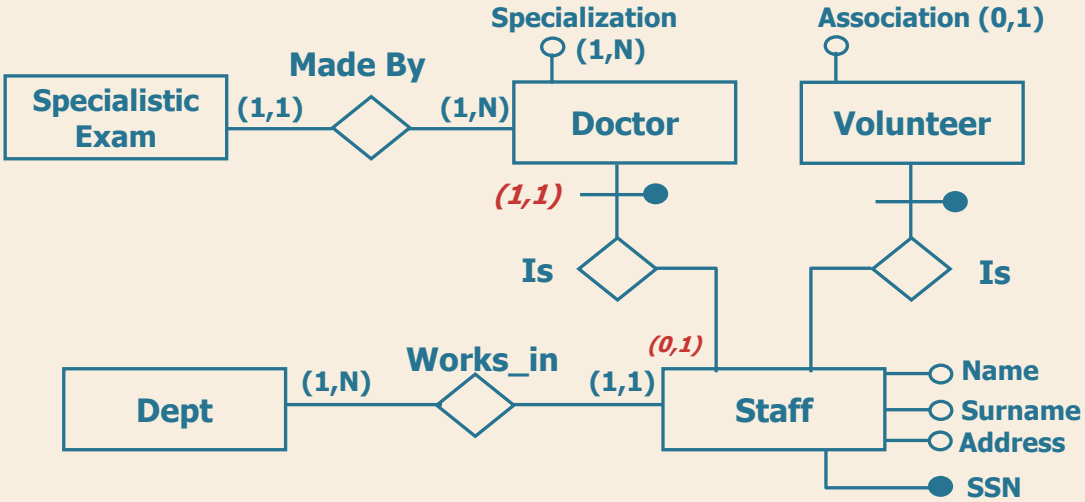
Relationship: parent and child entities



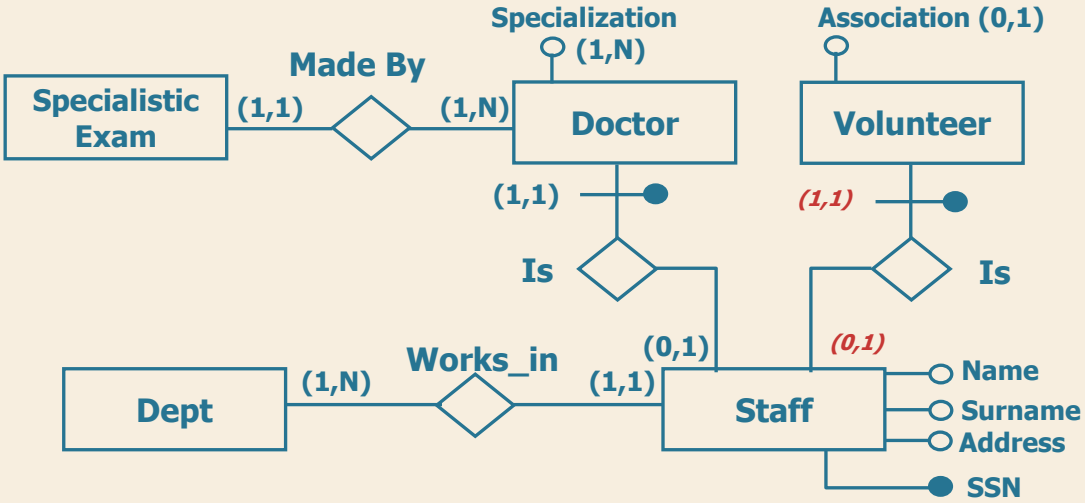
Child entities' identifier



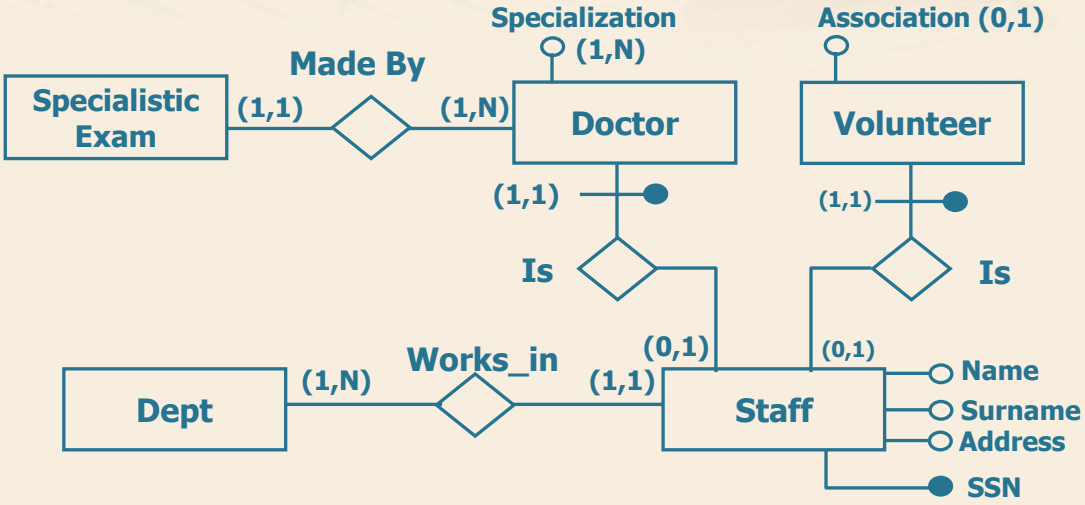
Cardinality of «is» relationship



Cardinality of «is» relationship



Generalization translated into relatioships



- This solution is more general and can be used for all generalizations
  - But it may be expensive to reconstruct the original data

Assessment of alternatives

- Merging child entities into parent entity is appropriate when:
  - Access operations apply to instances and attributes of child and parent entitites more or less in the same way (optimize data access).
  - Child entities are mildly differentiated (few null values)

Assessment of alternatives

- Merging parent entity into child entities is appropriate when:
  - The generalization is «total»
  - There are operations that refer only to occurrences of child entities and therefore it is useful to distinguish between different child entities (optimize data access).

Assessment of alternatives

- The various options can be combined
  - there are operations that refer only to instances of some child entities (optimize data access).

Assessment of alternatives

- In the presence of hierarchical generalization:
  - Apply the same procedure
  - Starting from the lower levels.

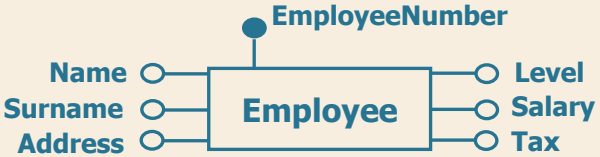
Logical Design

Partitioning of concepts

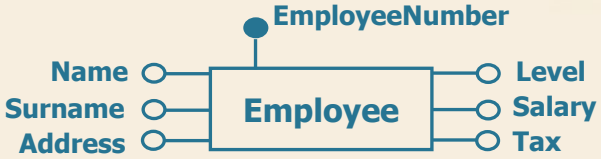
Partitioning of concepts

- Partitioning of entities and relationships
  - Better representation of different concepts
  - Separating attributes of the same concept that are accessed by different operation.
  - Improve the efficency of the operations.

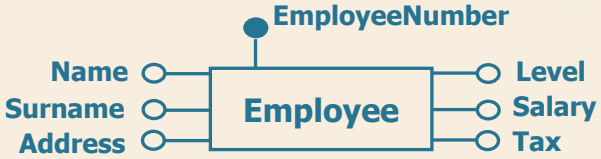
Entity partitioning



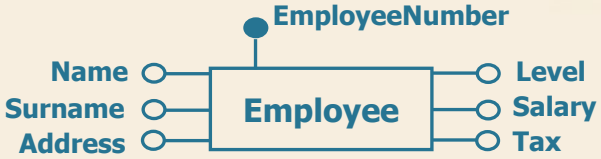
Entity partitioning



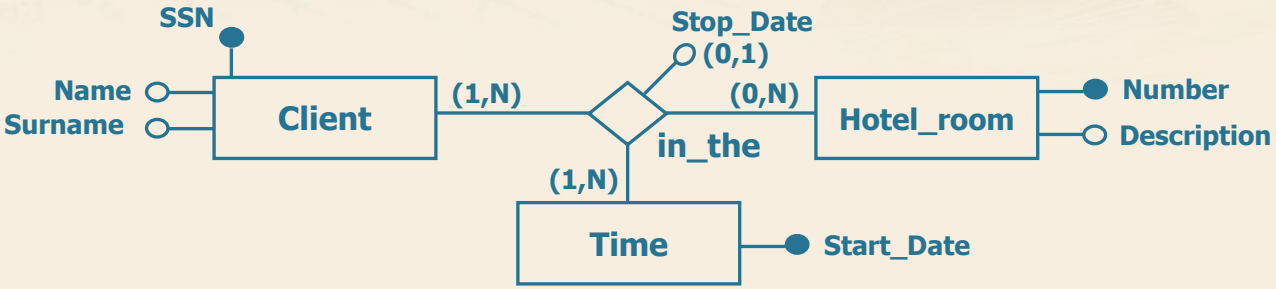
Cardinality of "Employment Data"



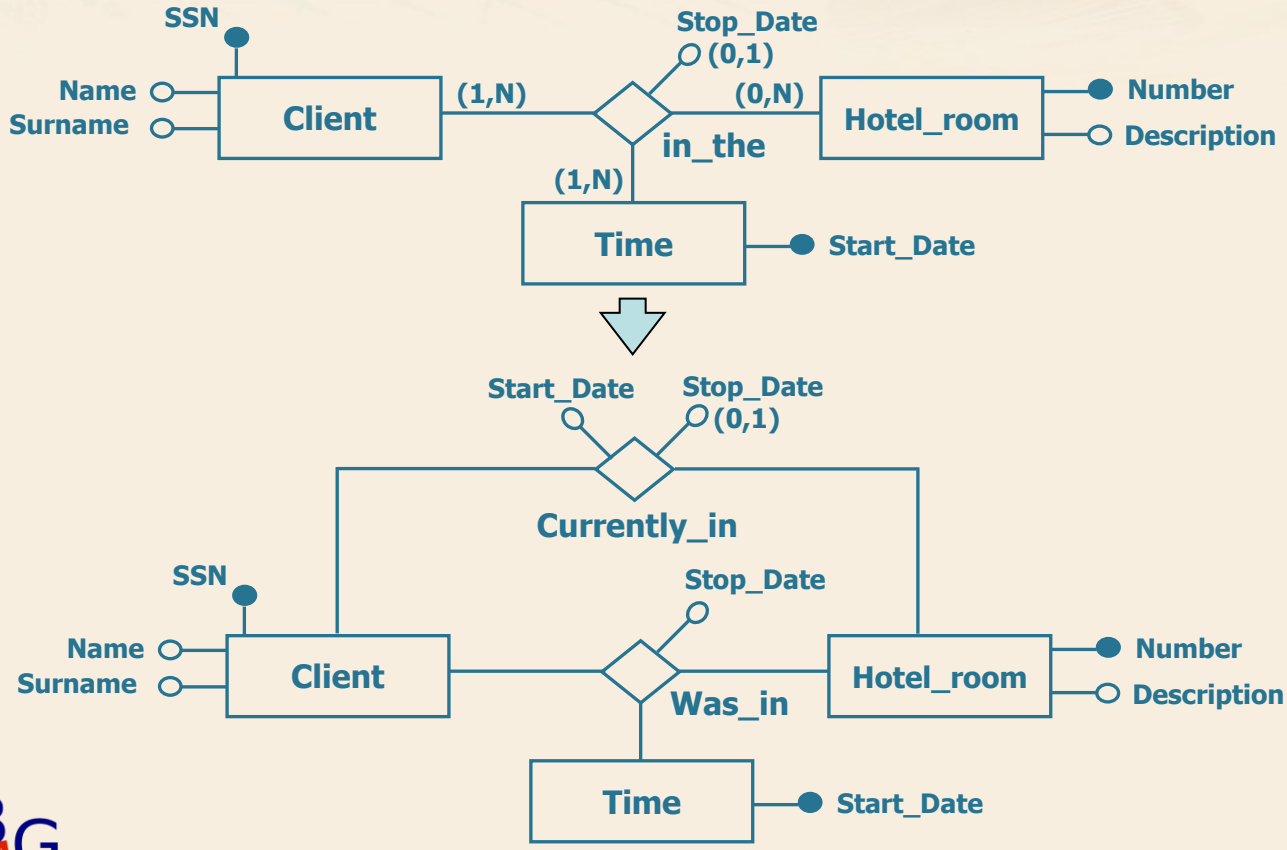
Cardinality of «Employement Data»



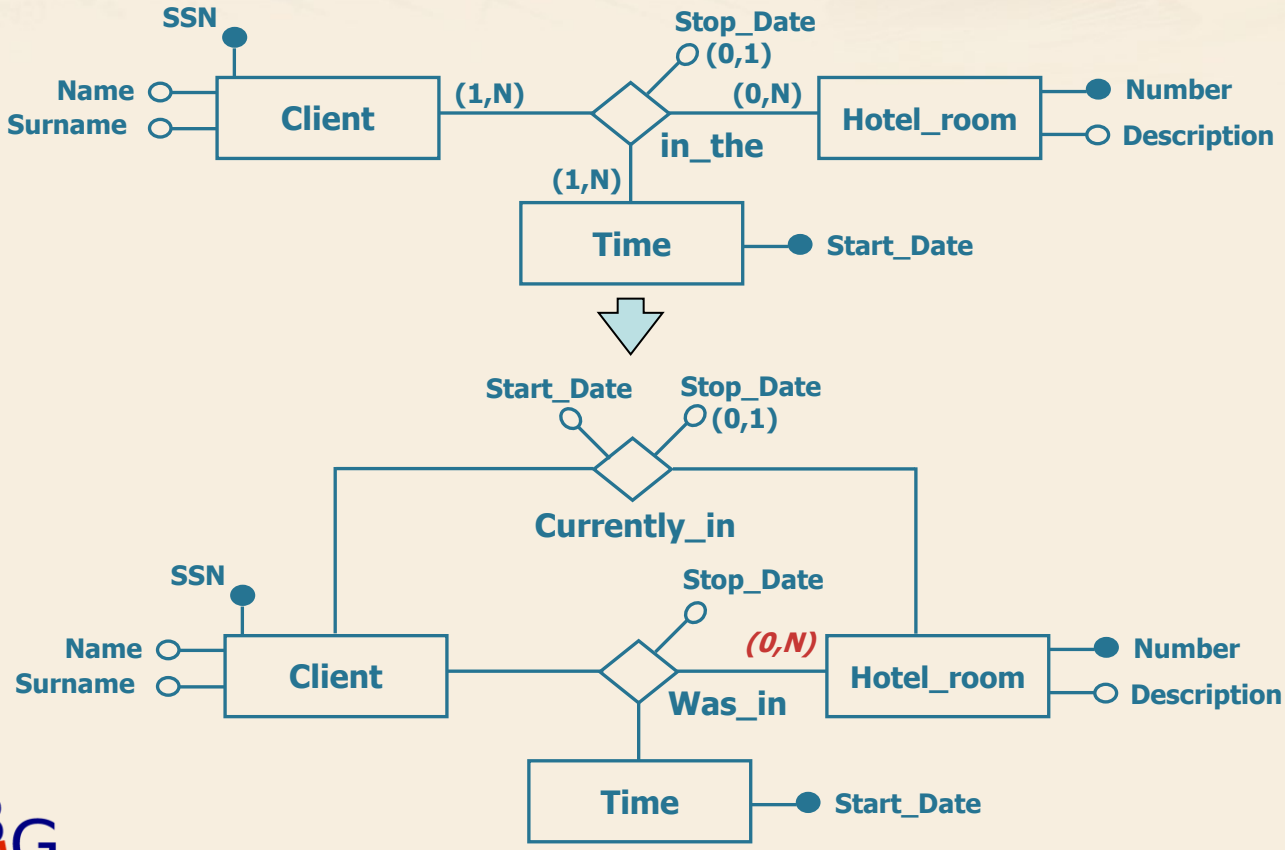
Relationships' partitioning



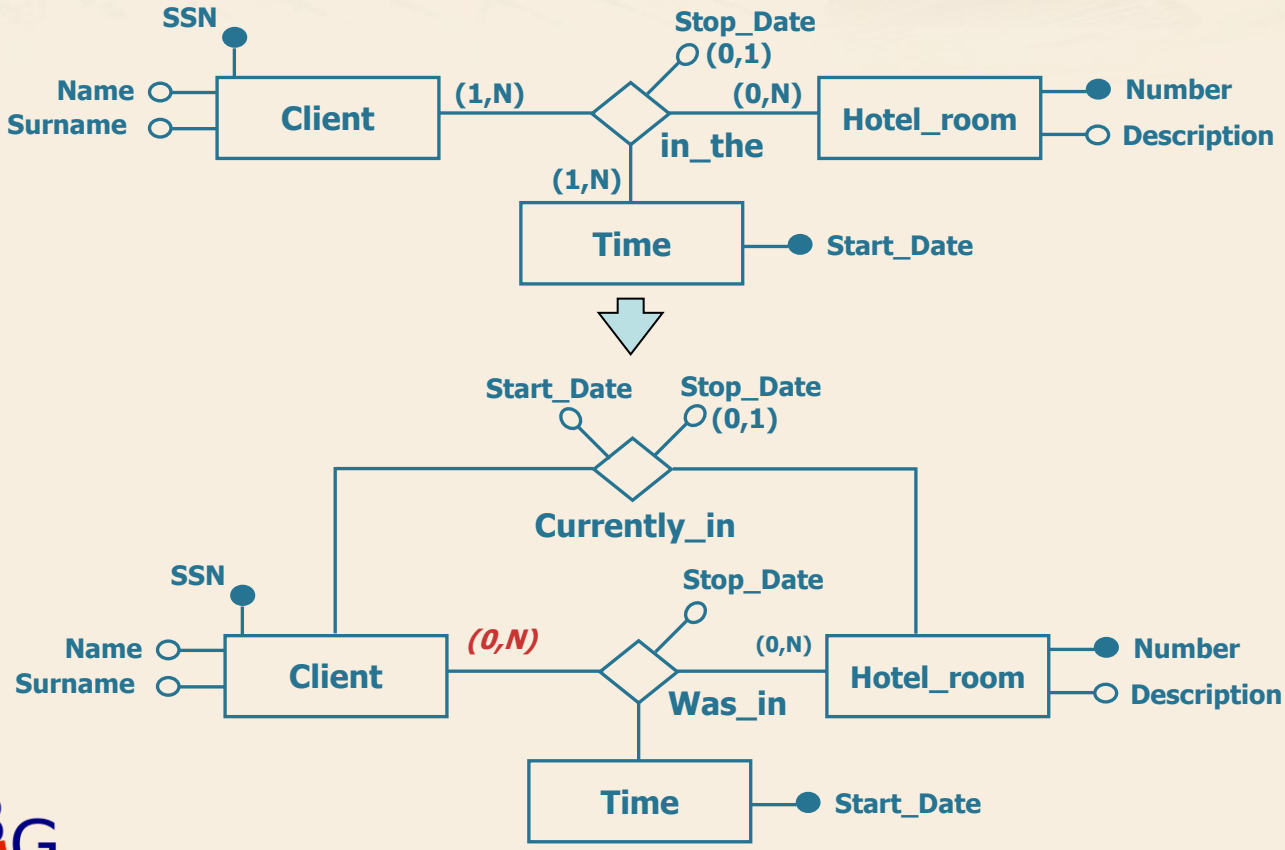
Relationships' partitioning



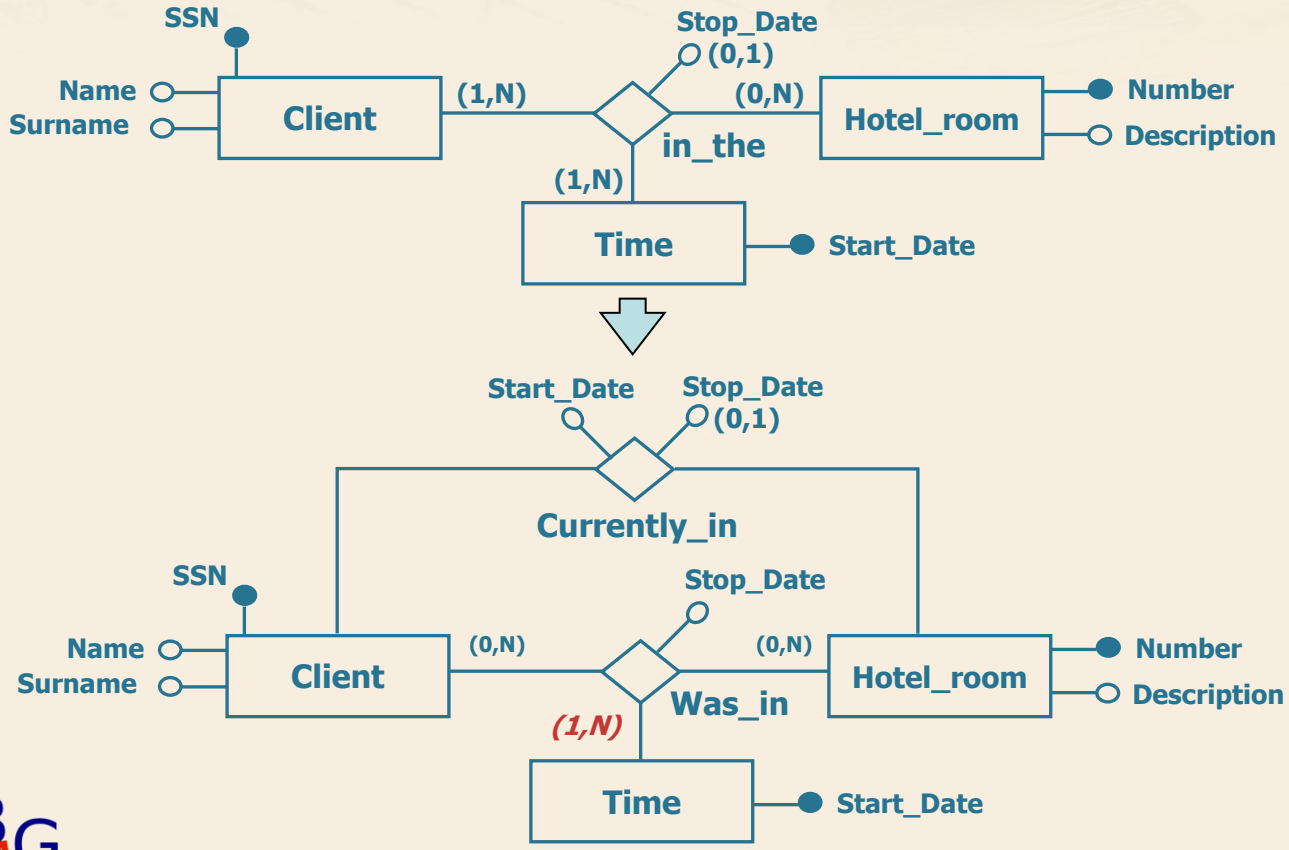
Cardinality of «Was in»



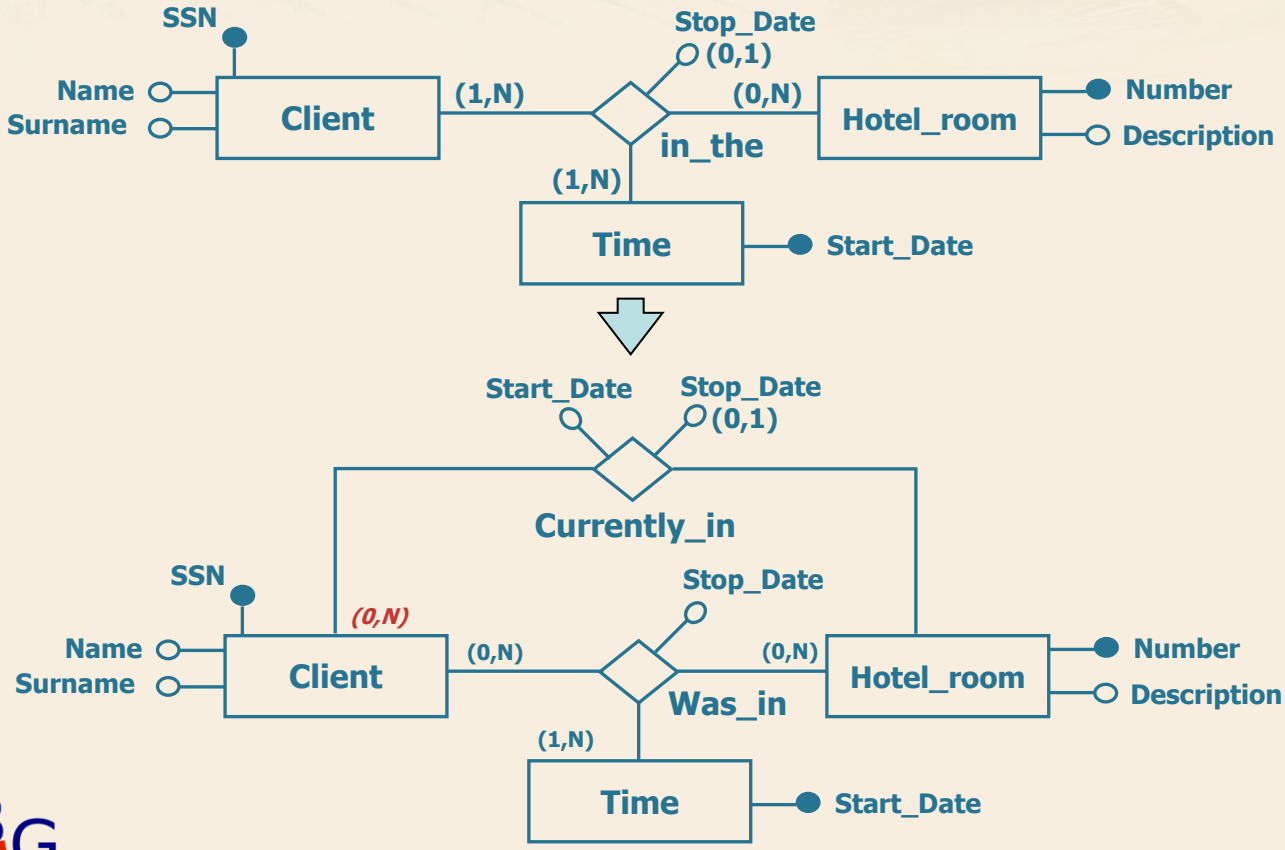
Cardinality of «Was in»



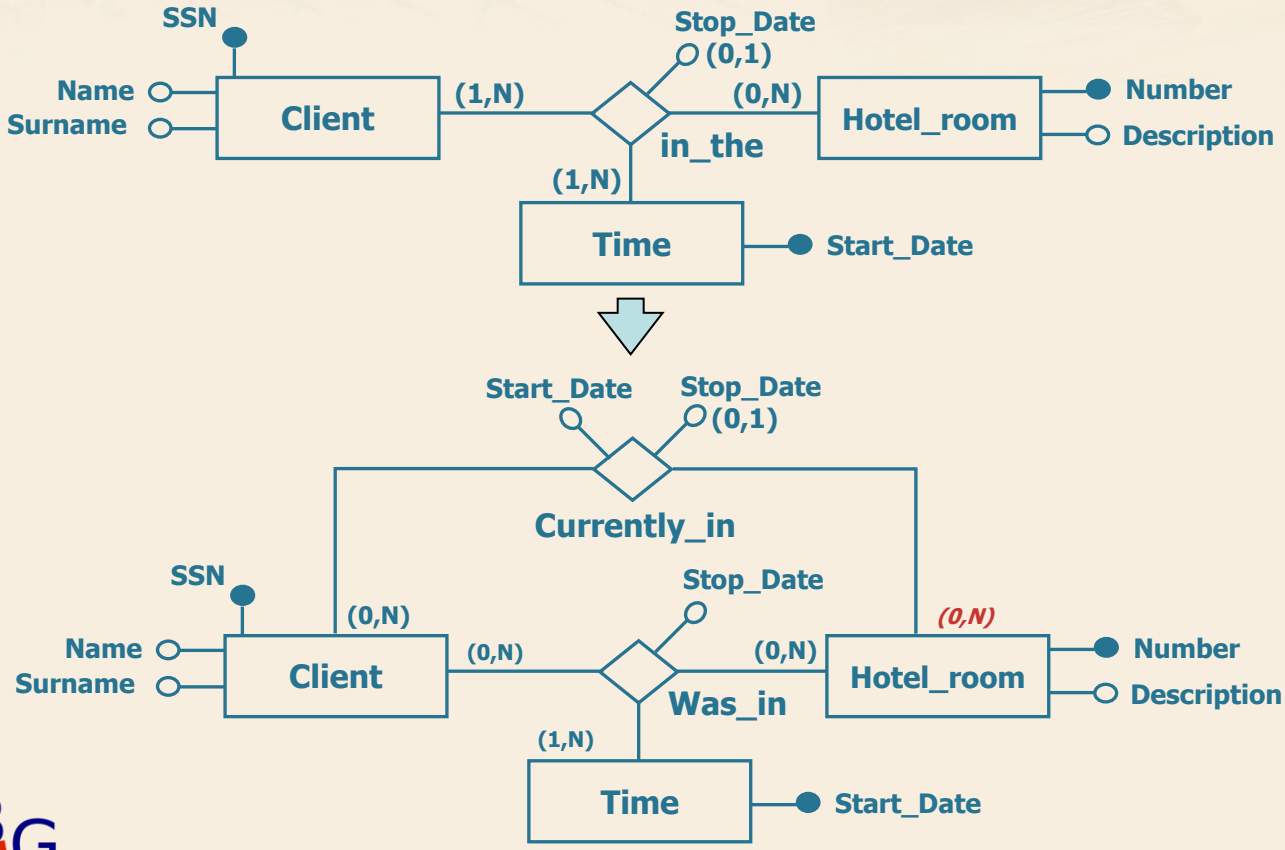
Cardinality of «Was in»



Cardinality of «Currently in»



Cardinality of «Currently in»



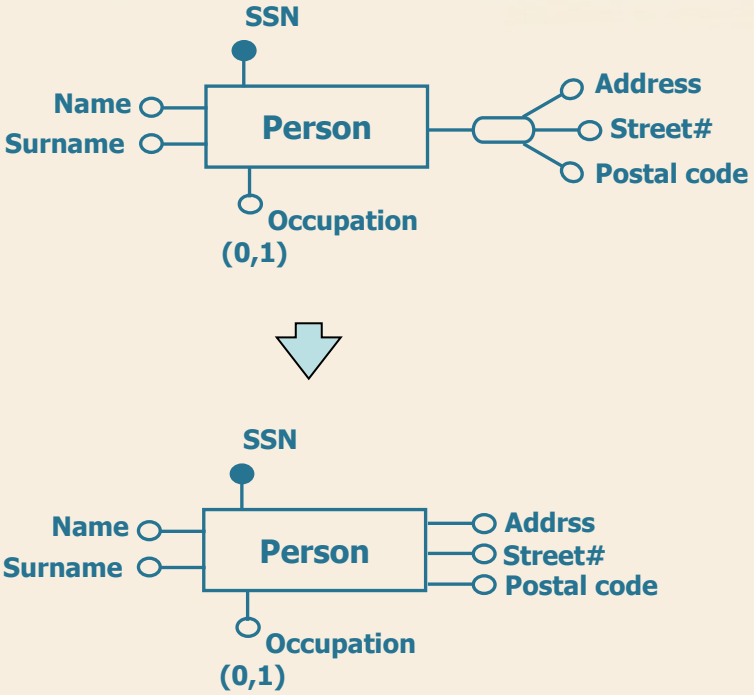
Logical Design

Removing composed attributes
Selection of primary identifiers

Removing composed attributes

- Composed (or compound) attributes are not representable in the relational model
- Two options
  - Split them in «individual» attributes
    - useful if you need to access each attribute separately

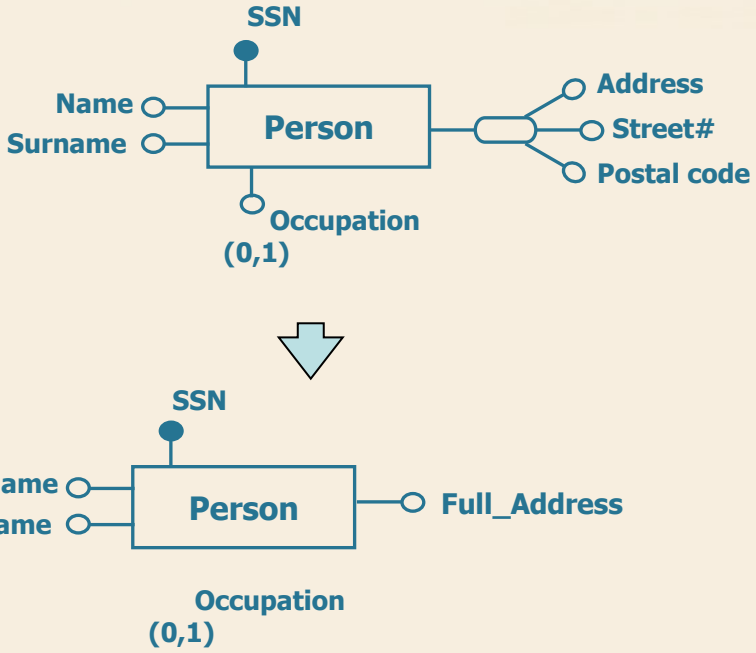
Split composed attributes



Removing composed attributes

- Composed (or compound) attributes are not representable in the relational model
- Two ways:
  - Split them in «individual» attributes.
    - useful if you need to access each attribute separately.
  - Use one attribute as a «link»
    - useful if access to comprehensive information is enough

Example


SSN
Name
Surname
Person
Address
Street#
Postal code
Occupation
(0,1)
SSN
Name
Surname
Person
Full_Address
Occupation
(0,1)

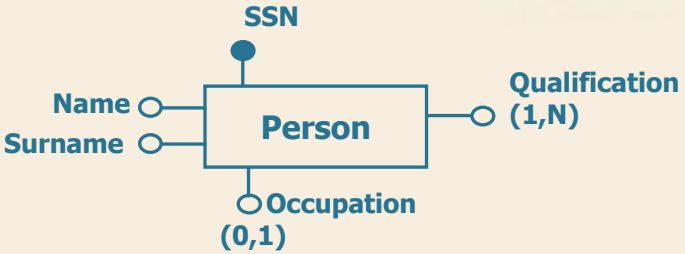
Logical Design

Removing multivalued attributes

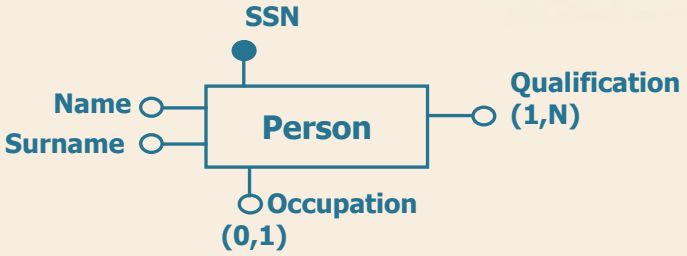
Removing multivalued attributes

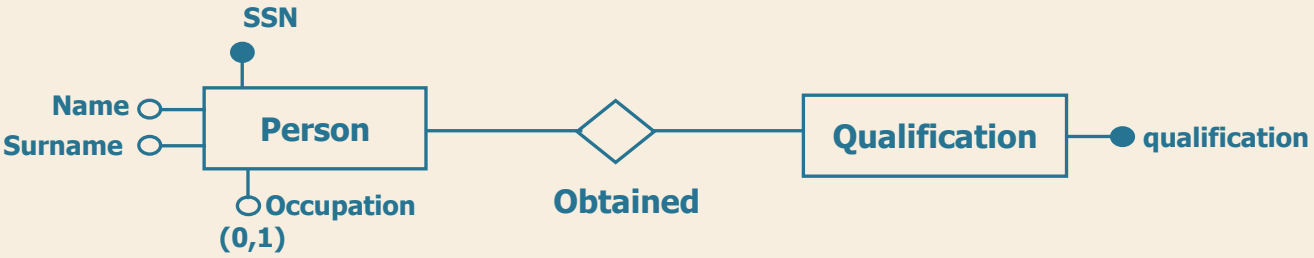
- Multi-valued attributes are not representable in the relational model
- A multi-valued attribute is represented by a relationship between
  - the original entity
  - a new entity
- Pay attention to the cardinality of the new relationship

Removing multivalued attributes

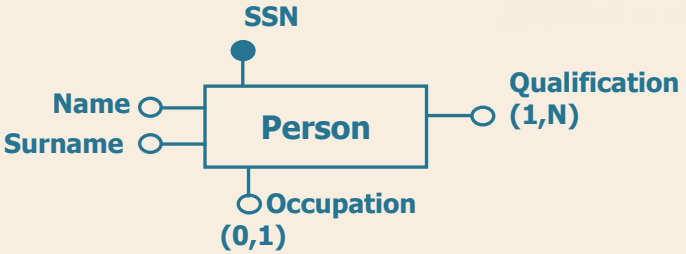


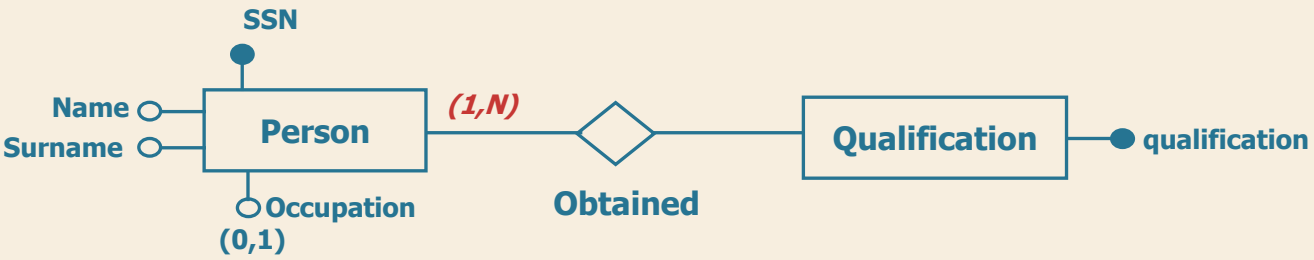
Removing multivalued attributes



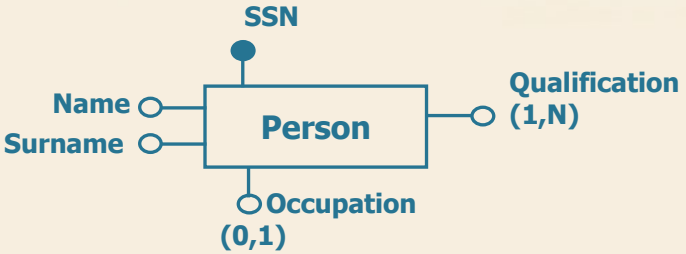


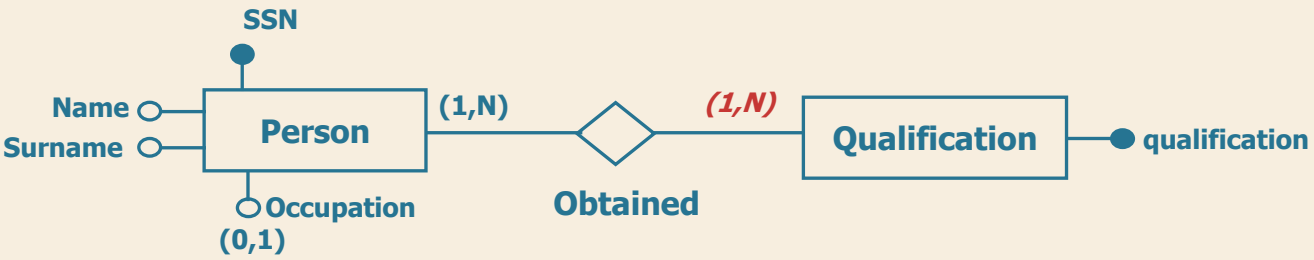
Cardinality of «Obtained»



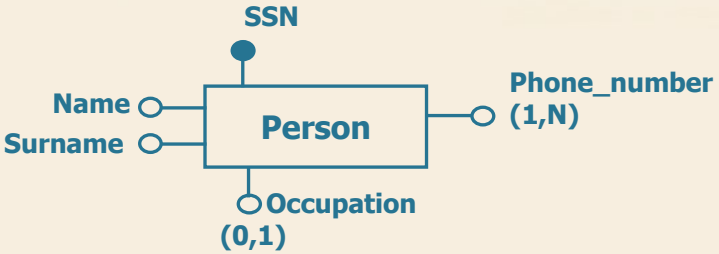


Cardinality of «Obtained»

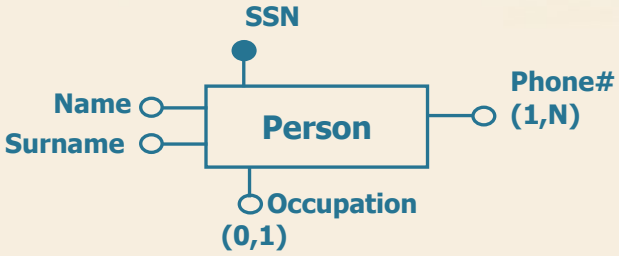


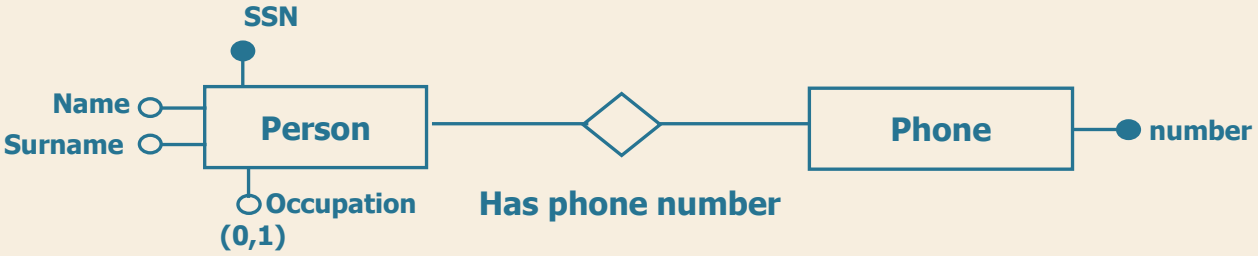


Removing multivalued attributes

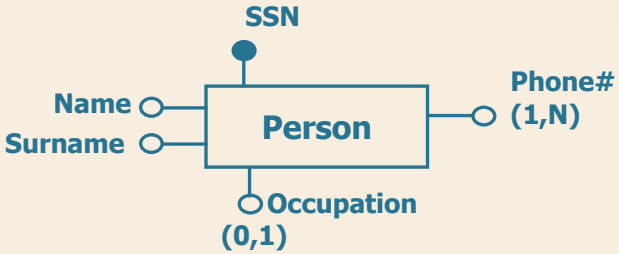


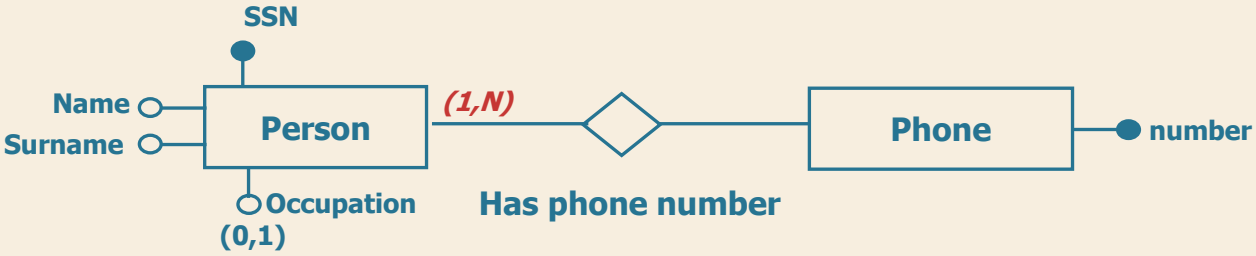
Removing multivalued attributes



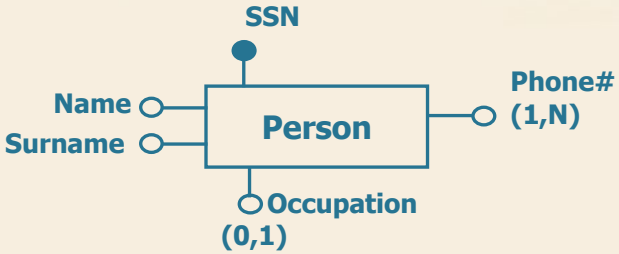


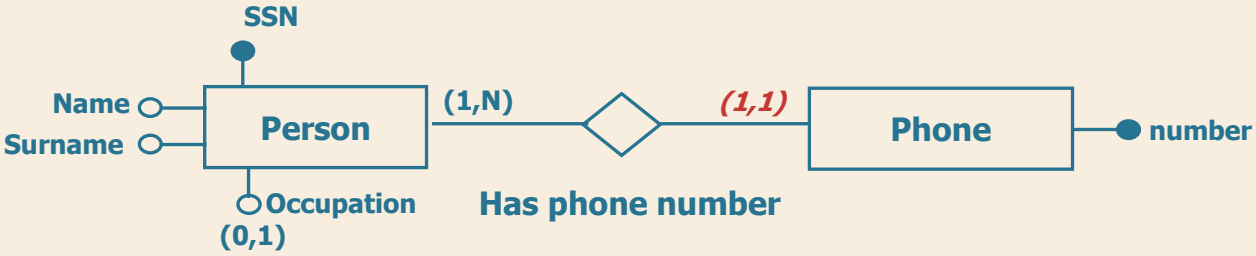
Cardinality of «Has phone number»





Cardinality of «Has phone number»





Logical Design

Selection of primary identifiers

DBMG

Selection of primary identifiers

- It is necessary to define the *primary key*
- The criteria for this decision are as follows
  - Attributes with **null** values **cannot** form primary identifiers.
  - Just **one** (better) or **few** attributes.
  - An **internal** identifier is preferable to an external one
  - It is used by many operations to access the occurrences
- It may be useful to introduce an additional attribute to represent the entity, often called code or ID, e.g. «ProductCode»