Database design

Logical Design

DBMG

1

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

DBMG 2

2

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

DBMG 3

3

Logical Design

Introduction

DBMG

4

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

DBMG 5

5

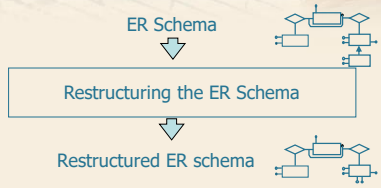
Logical design steps

ER Schema

DBMG 6

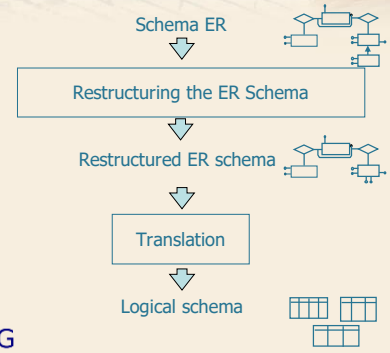
6

Logical design steps



7

Logical design steps



8

Relational logical design

Translation in relational model: entities and many to many relationships

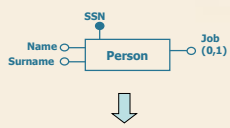
9

Translation in 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

10

Entities translation



Person(SSN, Name, Surname, Job*)

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

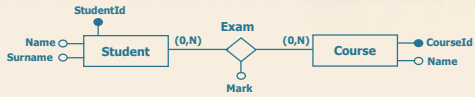
11

Translation of many to many binary relations

- Each many to many relationship is translated into a table
  - The primary key is the combination of the identificators 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)

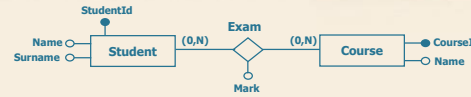
12

Many to many binary relationship



13

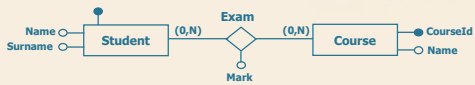
Many to many binary relationship: entity



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

14

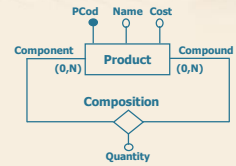
Many to many binary relationship



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

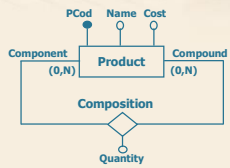
15

Recursive many to many binary relationship



16

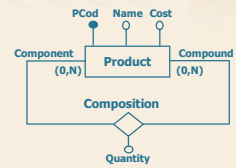
Recursive many to many binary relationship



Product(PCod, Name, Cost)

17

Recursive many to many binary relationship



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

18

Relational logical design

Translation in relational model: one to many relationship

DBMG

19

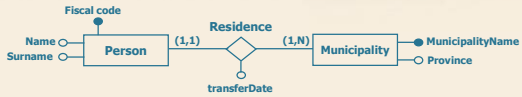
One to many binary relationships

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

DBMG 20

20

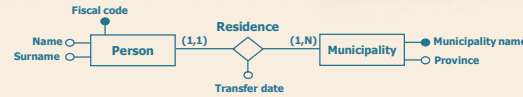
One to many binary relationship



DBMG 21

21

One to many binary relationship: entity



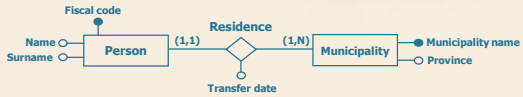
Person(FiscalCode, Name, Surname)

Municipality(MunicipalityName, Province)

DBMG 22

22

One to many binary relationship

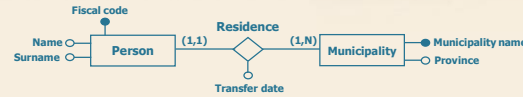


Person(FiscalCode, Name, Surname, *MunicipalityName*)
Municipality(MunicipalityName, Province)

DBMG 23

23

One to many binary relationship

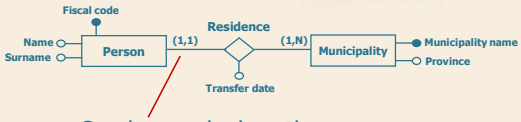


Person(FiscalCode, Name, Surname, MunicipalityName, *TransferDate*)
Municipality(MunicipalityName, Province)

DBMG 24

24

One to many binary relationship

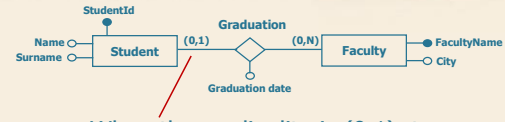


Can be used when the cardinality is (1,1)

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

25

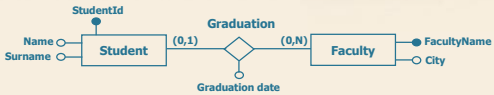
One to many binary relationship



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

26

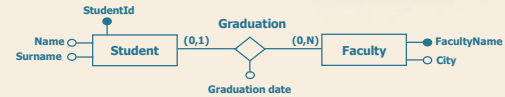
Alternative n.1: new table



Student(StudentiId, Name, Surname)
Faculty(FacultyName, City)

27

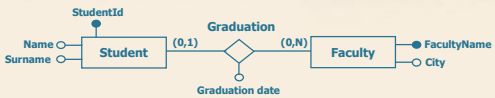
Alternative n.1: new table



Student(StudentiId, Name, Surname)
Faculty(FacultyName, City)
Graduation(StudentId, FacultyName, GraduationDate)

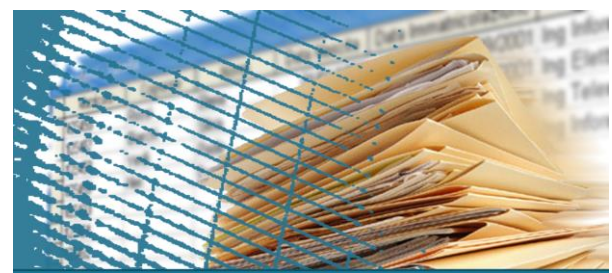
28

Alternative n.2: attributes



Student(StudentId, Name, Surname, FacultyName*, GraduationDate*)
Faculty(FacultyName, City)

29



Relational logical design

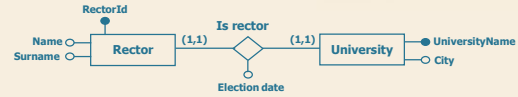
Translation in relational model: one to one relationship

One to one binary relationship

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

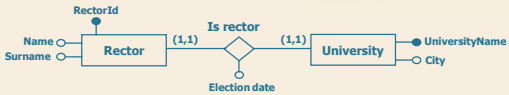
One to one binary relationship: case 1

- Mandatory partecipation from both sides



One to one binary relation: alternative n.1

- Mandatory partecipation from both sides

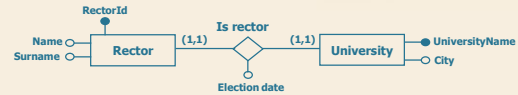


Rector(<u>RectorId</u>, Name, Surname)

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

One to one binary relation: alternative n.1

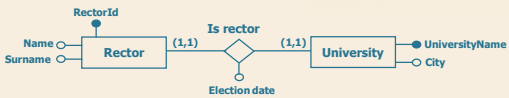
- Mandatory partecipation from both sides



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

One to one binary relation: alternative n.2

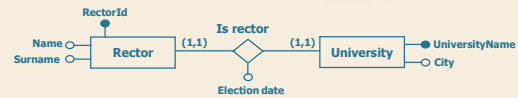
- Mandatory partecipation from both sides



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

One to one binary relation: alternative n.2

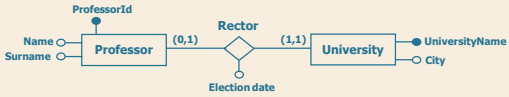
- Mandatory partecipation from both sides



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

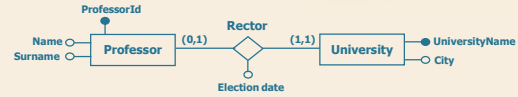
One to one binary relationship: case 2

- Optional partecipation on one side



One to one binary relationship: entity

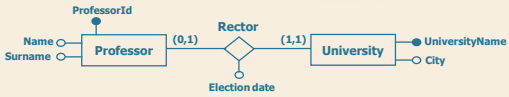
- Optional partecipation on one side



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

One to one binary relationship

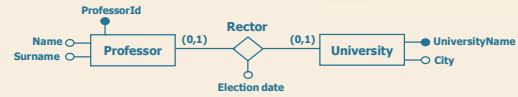
- Optional partecipation on one side



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

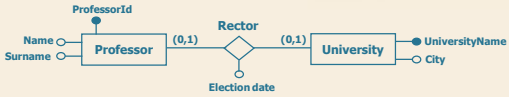
One to one binary relationship: case 3

- Optional partecipation from both sides



One to one binary relationship:alternative n.1

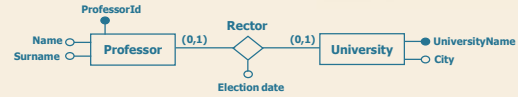
- Optional partecipation from both sides



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

One to one binary relationship:alternative n.1

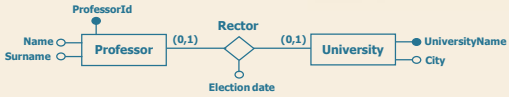
- Optional partecipation from both sides



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

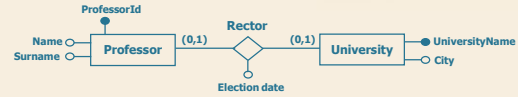
- Optional partecipation from both sides



Professor(ProfessorId, Name, Surname)
University(UniversityName, City)
Rector(ProfessorId, UniversityName, ElectionDate)

One to one binary relationship:alternative n.3

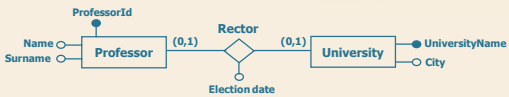
- Optional partecipation from both sides



Professor(ProfessorId, Name, Surname)
University(UniversityName, City)

One to one binary relationship:alternative n.3

- Optional partecipation from both sides

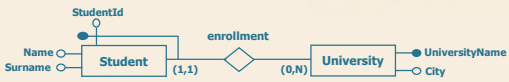


Professor(ProfessorId, Name, Surname)
University(Name, City, *ProfessorId**, *ElectionDate**)

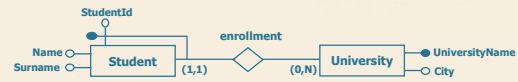
Relational logical design

Translation in relational model: entity with external identifier

Entity with external identifier

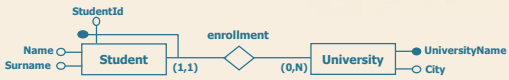


Entity with external identifier



University(UniversityName, City)
Student(*StudentId*, *UniversityName*, Name, Surname)

Entity with external identifier



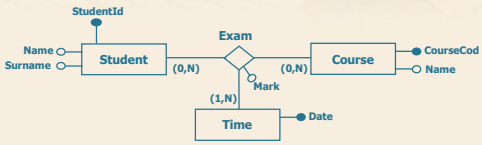
University(UniversityName, City)
Student (StudentId, UniversityName, Name, Surname)

- The relationship is represented along with the identificator

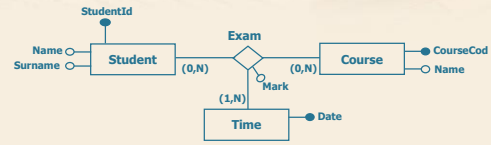
Relational logical design

Translation in relational model: ternary relationships

Ternary relationship

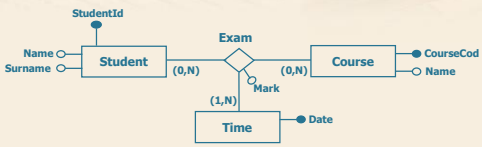


Ternary relationship: entity



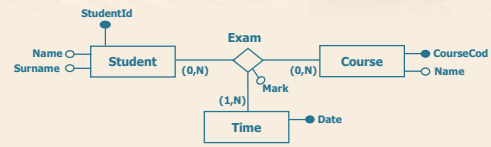
Student(StudentId, Name, Surname)
Course(CourseCod, Name)
Time(Date)

Ternary relationship: identificator



Student(StudentId, Name, Surname)
Course(CourseCod, Name)
Time(Date)
Exam(StudentId, CourseCod, Date)

Ternary relationship: attributes



Student(StudentId, Name, Surname)
Course(CourseCod, Name)
Time(Date)
Exam(StudentId, CourseCod, Date, Mark)

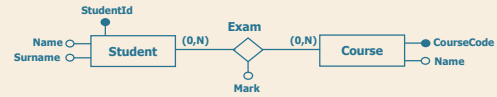
Relational logical design

Referential integrity constraints

55

Referential integrity constraints

- Relationships represent referential constraints



56

56

Referential integrity: exam relationship

- Tables

Student(StudentId, Name, Surname)
Course(CourseCod, Name)
Exam(*StudentId*, CourseCod, Mark)

- Referential integrity constraints

Exam(StudentId) REFERENCES Student(StudentId)

57

57

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)

58

58

Logical Design

Restructuring
an ER schema

59

ER scheme 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

60

60

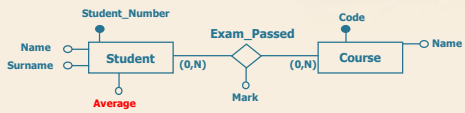
Restructuring tasks

- 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
  - Decide whether to keep or remove them
- Advantages
  - Speed up and simplify queries
- Disadvantages
  - increased complexity of updates
  - slowing down of 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 studendt's average.
  - if kept, the redundancy indication must be added in the relational schema.

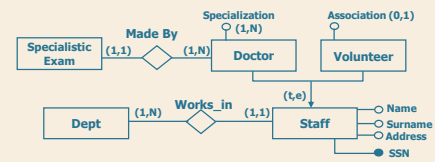
Logical Design

Removing generalizations

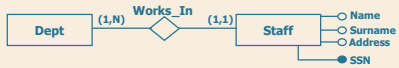
Removing Generalization

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

Example

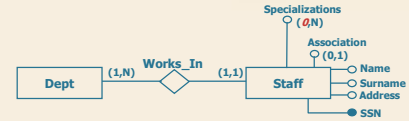


Child->Parent



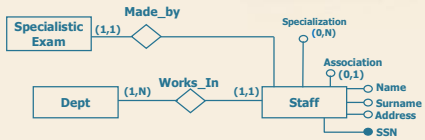
67

Child entities' attributes



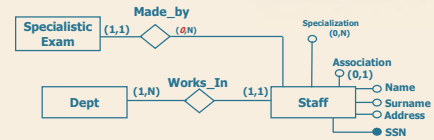
68

Relations with child entities



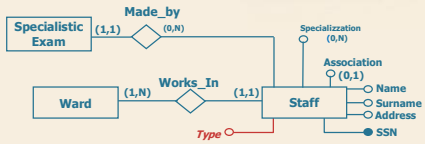
69

Relations with child entities



70

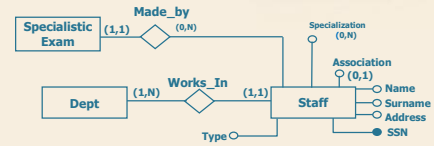
The «Type» attribute



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

71

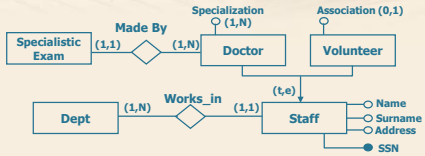
Child->Parent



- 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

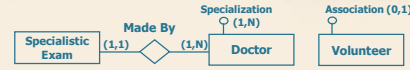
72

Example



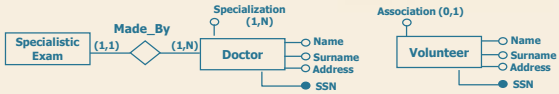
73

Parent->Child



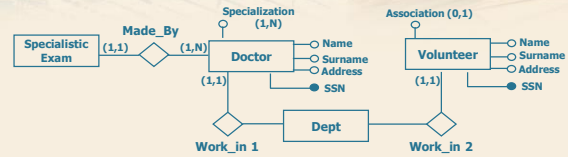
74

Parent's attributes



75

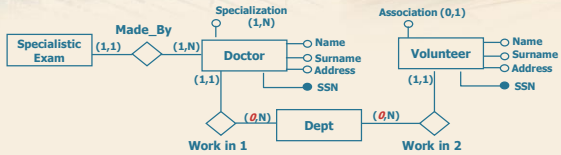
Relationships with parent



- Relationships with the parent entity need to be split

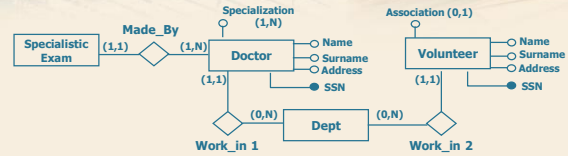
76

Cardinality of «work in»



77

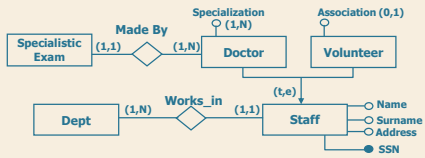
Parent -> Child



- 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

78

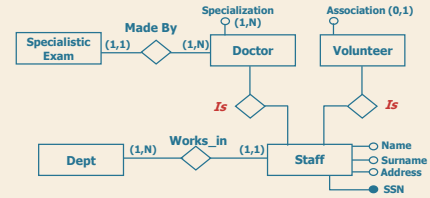
Back to the original example



DBMG 79

79

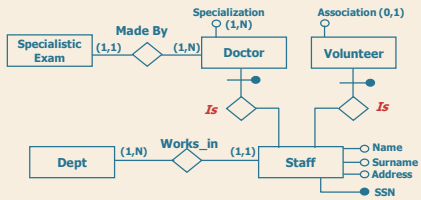
Relationship: parent and child entities



DBMG 80

80

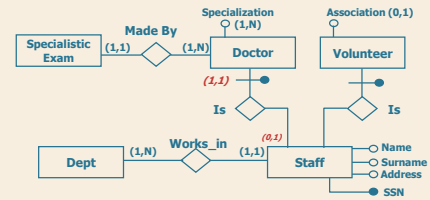
Child entities' identifier



DBMG 81

81

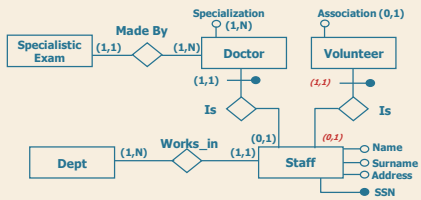
Cardinality of «is» relationship



DBMG 82

82

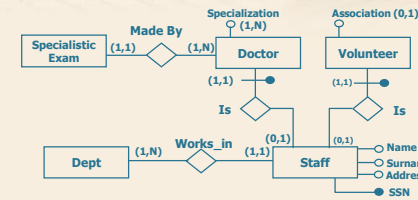
Cardinality of «is» relationship



DBMG 83

83

Generalization translated into relatioships



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

DBMG 84

84

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)

DBMG 85

85

Assessment of alternatives

- Merging parent entity into child entities is appropriate when:
  - The generalizzation 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).

DBMG 86

86

Assessment of alternatives

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

DBMG 87

87

Assessment of alternatives

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

DBMG 88

88

Logical Design

Partitioning of concepts

DBMG

89

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.

DBMG 90

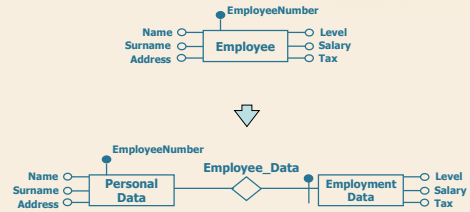
90

Entity partitioning

91

91

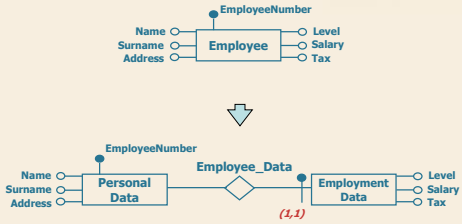
Entity partitioning



92

92

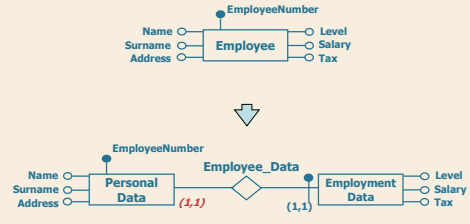
Cardinality of "Employment Data"



93

93

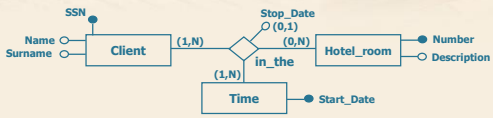
Cardinality of «Employement Data»



94

94

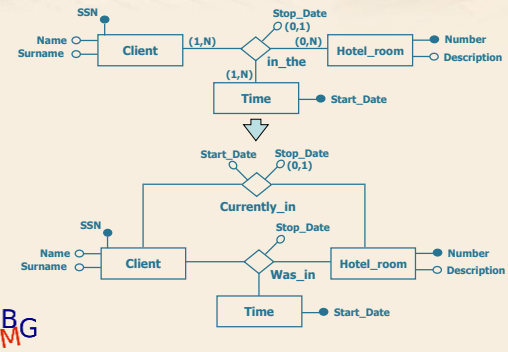
Relationships' partitioning



95

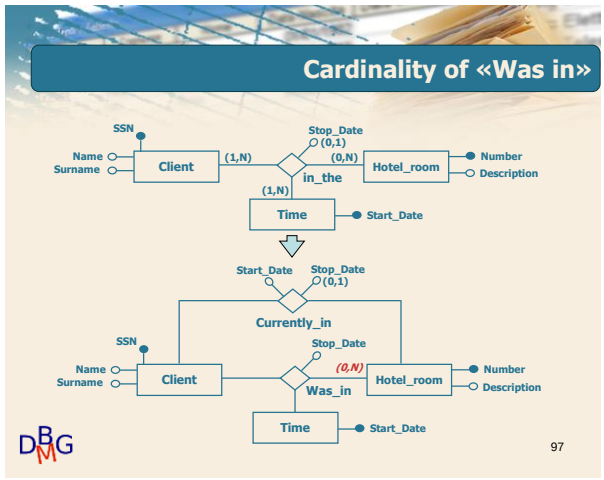
95

Relationships' partitioning

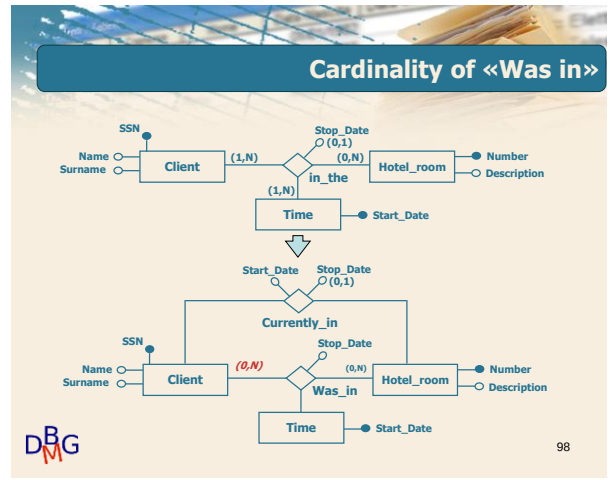


96

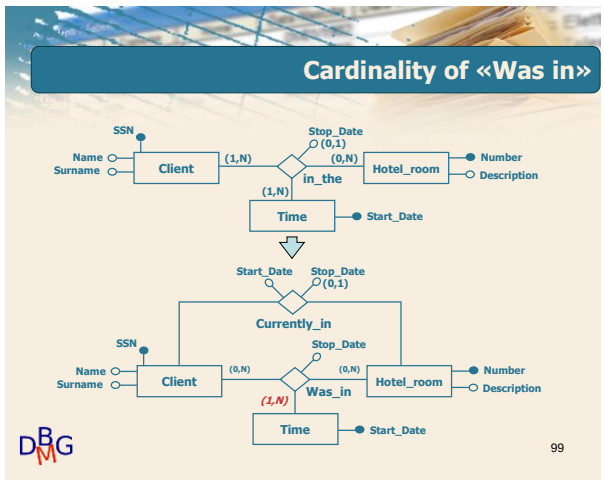
96



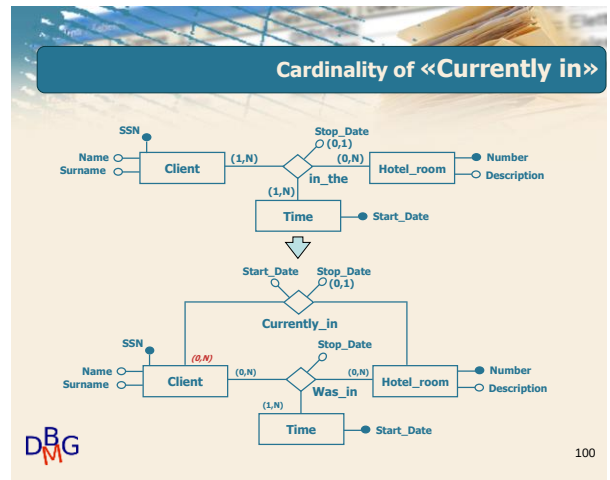
97



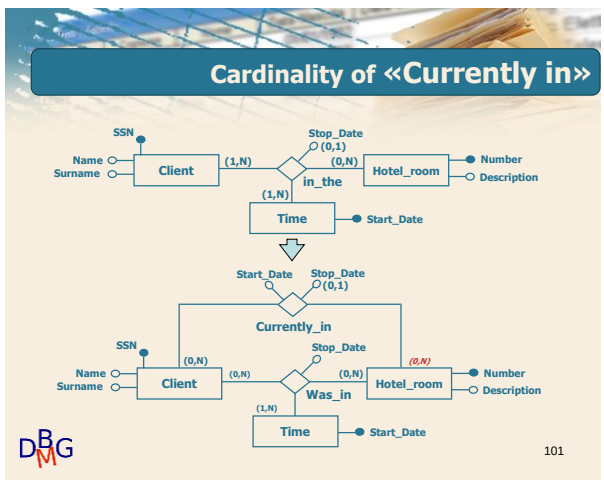
98



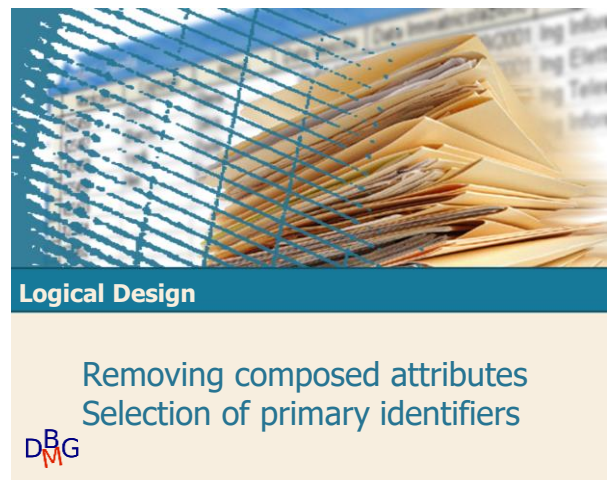
99



100



101

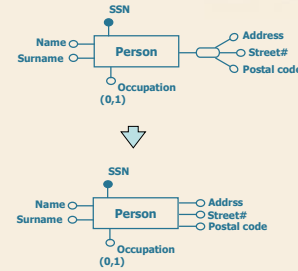


102

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

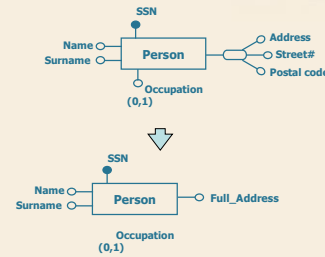
Split composed attributes



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



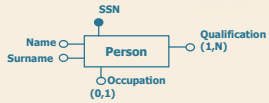
Logical Design

Removing multivalued attributes

Removing multivalued attributes

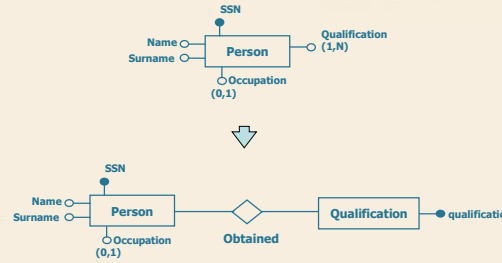
- Multi-valued attributes are not representable in the relational model
- A multi-valued attribute is represented by a relationship between
  - the original entity
  - a new entity
- Pay attention to the cardinality of the new relationship

Removing multivalued attributes



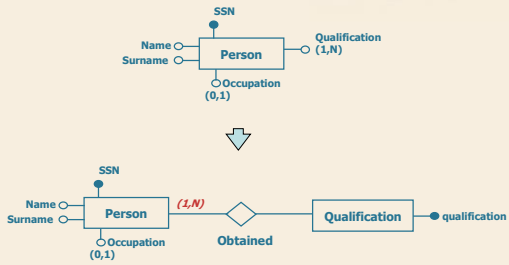
109

Removing multivalued attributes



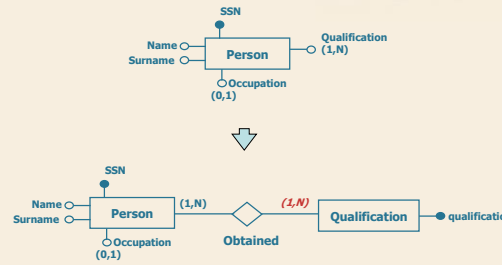
110

Cardinality of «Obtained»



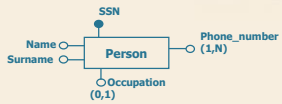
111

Cardinality of «Obtained»



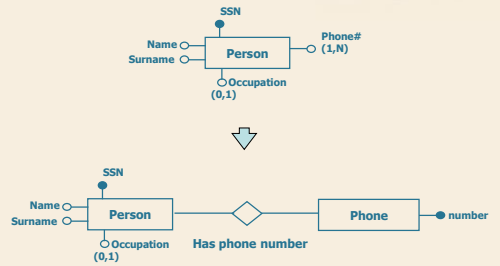
112

Removing multivalued attributes



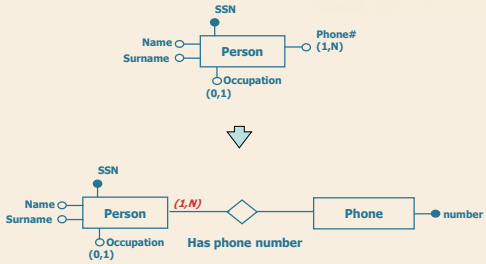
113

Removing multivalued attributes



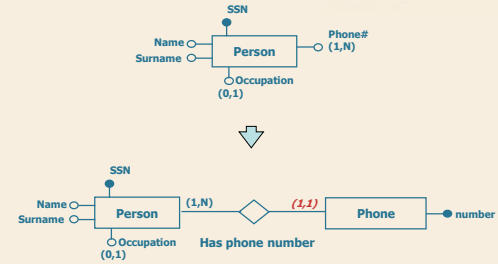
114

Cardinality of «Has phone number»



115

Cardinality of «Has phone number»



116

Logical Design

Selection of primary identifiers

117

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»

118