Database design

0

Database design

- Entity-Relationship Model
- Conceptual design
- Logical design
- Normalization

1

Entity-Relationship Model

Database design

2

Entity-Relationship Model

- Life cycle of an information system
- Database design
- Entities and Relationships
- Attributes
- Identifiers
- Generalization
- Documenting E-R Schematics
- UML and E-R

3

Database design

- The design of a database is one of the activities of the process of developing an information system
  - must be seen in the broader context of the life cycle of an information system

Life cycle of an information system

Database design

4

5

Life cycle of an information system

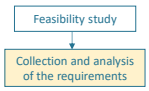
Determination of the costs of the different alternatives and the priorities for the implementation of each system component

6

6

Life cycle of an information system

- Definition of the properties and functionalities of the information system
- Requires user interaction
- Produces a comprehensive, but informal, description of the system to be implemented

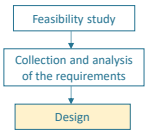


7

7

Life cycle of an information system

- Divided into data and application design
- Produces formal descriptions

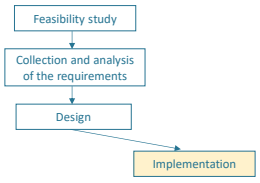


8

8

Life cycle of an information system

- Implementation of the information system according to the characteristics defined in the design phase

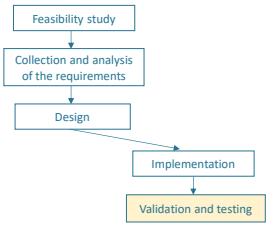


9

9

Life cycle of an information system

- Verification of the correct functioning and quality of the information system
- It can lead to changes in requirements or design revision

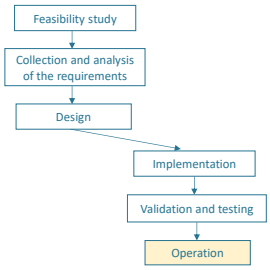


10

10

Life cycle of an information system

- System operation
- Requires maintenance and managing operations

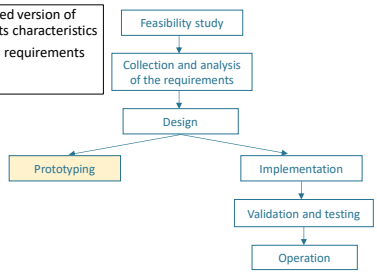


11

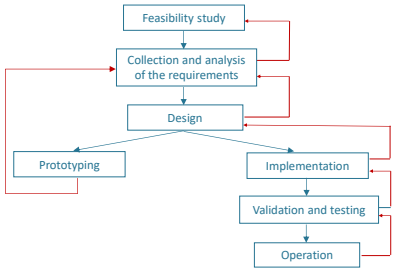
11

Life cycle of an information system

- Quickly create a simplified version of the system to evaluate its characteristics
- It can lead to changes in requirements or design revision



Life cycle of an information system



Database design

Database design

Database design

- The database is an important component of the entire system
- Data-driven design methodology
  - the design of the database precedes that of the applications that use it
  - greater attention to the design phase than to the other phases

Design Methodology

- A design methodology consists of
  - decomposition of the project activity into successive and independent phases
  - strategies to be followed in the various phases and criteria for choosing the best strategy
  - reference models to describe the input and output data of the various phases

Properties of the methodology

Generality
- can be used regardless of the problem and the tools available

Quality of the result
- in terms of correctness, completeness and efficiency with respect to the resources used

Ease of use
- of both strategies and reference models

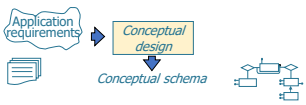
Data-driven design

- For databases, methodology based on separating two key decisions
  - *what* to represent in the database
    - conceptual design
  - *how* to represent it
    - logical and physical design

Stages of database design

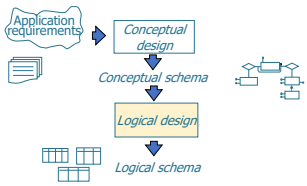
Informal specification of the reality of interest
- Application properties
- Application functionalities

Stages of database design



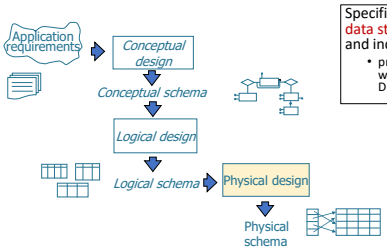
Representation of informal specifications in the form of a conceptual diagram
- formal and complete description, referring to a conceptual model
- Independent from implementation aspects (data model)
- the aim is to represent the information content of the database

Stages of database design



Translating the conceptual schema into the logical schema
- depends on the chosen data logic model
- takes into account the optimization of data processing operation
- schema quality verified by formal techniques (normalization)

Stages of database design



Specification of the physical data storage parameters (File and index organization)
- produces a physical model, which depends on the chosen DBMS

Entity-Relationship Model

Database design

The E-R (Entity-Relationship) model

- It is the most widely used conceptual model
- Provides constructs to describe specifications about data structure
  - in a simple and understandable way
  - with a graphic formalism
  - independent of the data model, which can be chosen later
- Several variants are available

Main elements of the E-R model

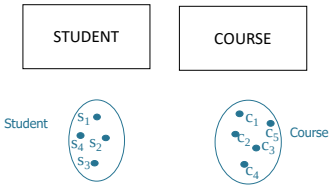
- Entity
- Relations
- Attributes
- Identifiers
- Generalizations and subsets

Entity

Entity name

- It represents classes of real-world objects (people, things, events, ...), which have
  - common Properties
  - autonomous existence
- Examples: Employee, Student, Article
- An occurrence of an entity is an object of the class that the entity represents

Example of entities

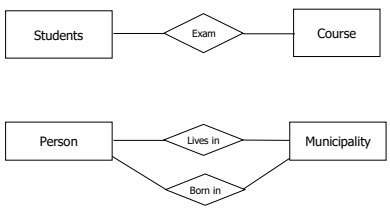


Relationship

RELATIONSHIP NAME

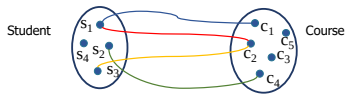
- Represents a logical link between two or more entities
- Examples: exam between student and course, residence between person and municipality
- Not to be confused with the relation of the relational model
  - sometimes referred to as association

Relationship examples



Occurrences of a relationship

- An occurrence of a relationship is an n-tuple (pair in the case of a binary relationship) consisting of occurrences of entities, one for each of the entities involved
- There can be no identical n-tuples

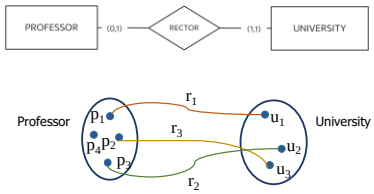


Cardinality of binary relationships

- They are specified for each entity that participates in a relationship
- Describe the minimum and maximum number of occurrences of a relationship in which an occurrence of an entity can participate
  - minimum can be either
    - 0 (optional participation)
    - 1 (participation required)
  - maximum varies between
    - 1 (at most one occurrence)
    - N (arbitrary number of occurrences)

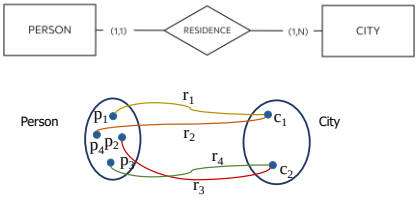
Cardinality of binary relationships

- 1-to-1 relationship



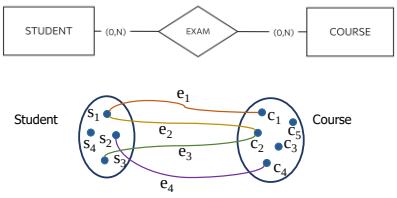
Cardinality of binary relationships

- 1-to-N (many) relationship



Cardinality of binary relationships

- N-to-N (Many to Many) relationship

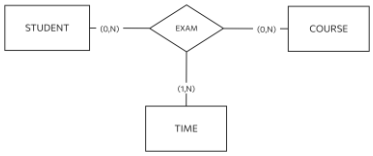


Limitations of binary relationships

- It is not possible for a student to take the same exam more than once

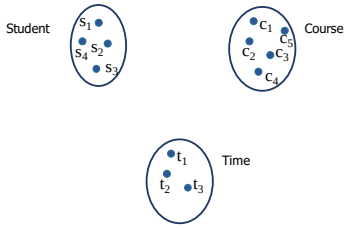
Ternary relationship

- A student may take the same exam more than once at different times.

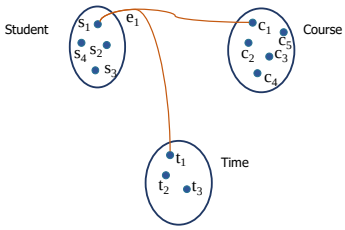


- Example of an instance of the EXAM report $s_1\ c_1\ t_1$, $s_1\ c_1\ t_2$, ...

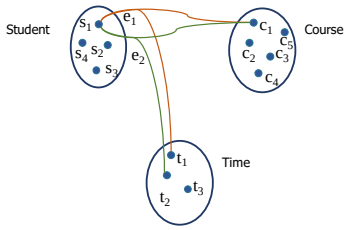
Occurrences of a ternary relationship



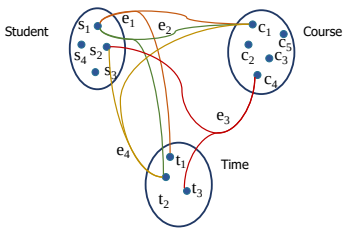
Occurrences of a ternary relationship



Occurrences of a ternary relationship



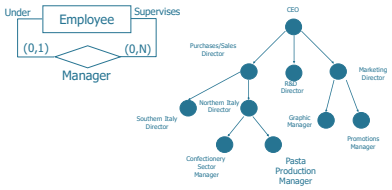
Occurrences of a ternary relationship



Cardinality of ternary relationships

- Minimum cardinalities are rarely 1 for all entities involved in a relationship
- The maximum cardinalities of an n-ary relationship are (practically) always N
  - if the participation of an entity E has a maximum cardinality of 1, it is possible to eliminate the n-ary relationship and associate the entity E with the others by binary relations

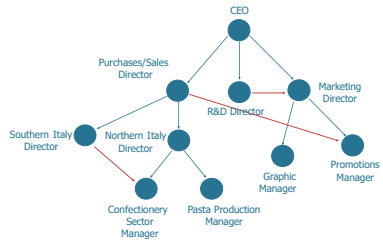
Recursive relationship



- Relationship between an entity and itself
- If the relationship is not symmetrical, the two roles of the entity must be defined

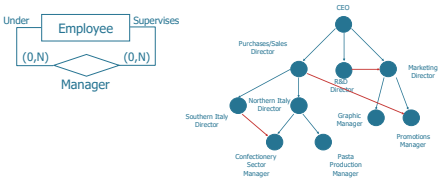
42

Recursive relationship



43

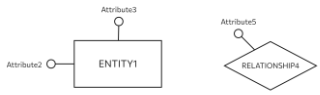
Recursive relationship



- An employee might have several managers

44

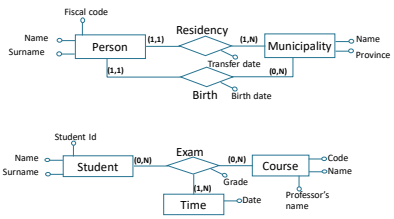
Attribute



- Describes an elementary property of an entity or relationship
- Examples
  - surname, first name, student ID are attributes that describe the student entity
  - grade is an attribute that describes the exam relationship
- Each attribute is characterized by the domain, the set of admissible values for the attribute

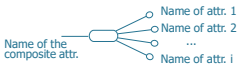
45

Examples of attributes



46

Composite attribute

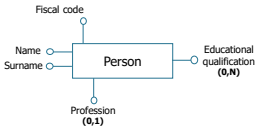


- Group of attributes that have closely connected meanings or uses.
- Example:

47

Cardinality of an attribute

- Can be specified for entity or relationship attributes
- Describes the minimum and maximum number of attribute values associated with an occurrence of an entity or relationship
  - if it is omitted it corresponds to (1,1)
  - minimum 0 corresponds to an attribute that admits a null value
  - maximum N corresponds to an attribute that can have more than one value for the same occurrence (multivalued attribute)



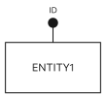
Identifier

- It is specified for each entity
- Describes the concepts (attributes and/or entities) of the schema that allow you to uniquely identify the occurrences of the entities
  - each entity must have at least one identifier
  - there can be more than one appropriate identifier for an entity
- The identifier can be
  - internal or external
  - simple or composite

Internal identifier

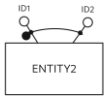
Simple

- consisting of a single attribute



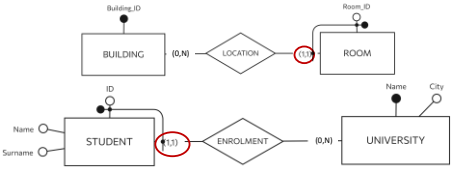
Composite

- consisting of multiple attributes



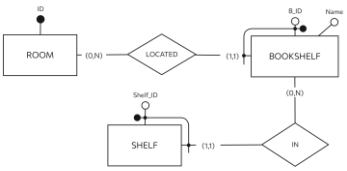
External identifier

- An entity that does not have internal attributes sufficient to define an identifier is called a weak entity
- The weak entity must participate with cardinality (1,1) in each of the relationships that provide part of the identifier



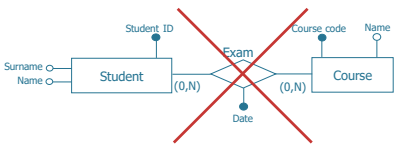
Remarks

- An external identifier can involve an entity that is itself externally identified
  - No identification cycles should be generated



Remarks

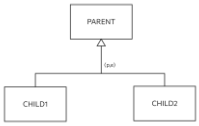
- Relationships do *not* have identifiers



Generalization

It describes a logical link between an entity E and one or more entities E_1, E_2,..., E_n, that are particular cases of E.

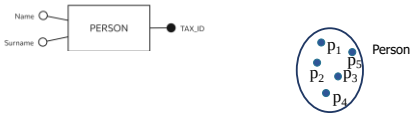
- E is called parent entity, and is a generalization of E_1, E_2,..., E_n
- E_1, E_2,..., E_n are called child entities, and are specializations of E



54

54

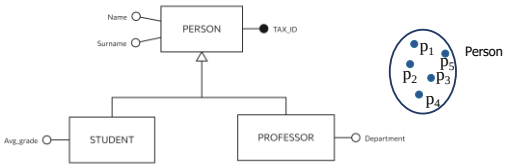
Generalization: example



55

55

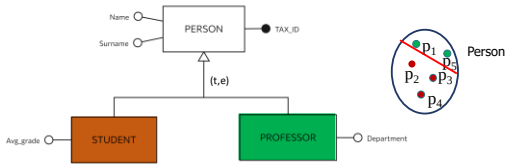
Generalization: example



56

56

Generalization: example



57

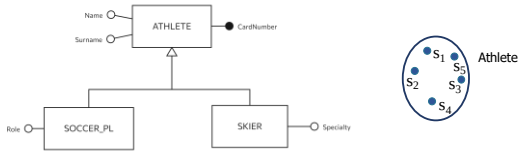
57

Generalization: example

58

58

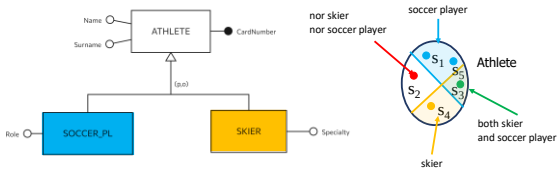
Generalization: example



59

59

Generalization: example



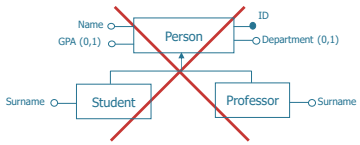
Generalization: properties

- Each occurrence of a child entity is also an occurrence of the parent entity
- Each property of the parent entity (attributes, identifiers, relationships, other generalizations) is also a property of each child entity
  - property known as inheritance
- An entity can be involved in multiple different generalizations

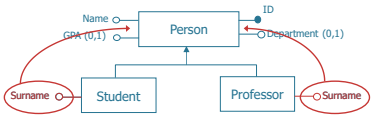
Generalization: properties

- Orthogonal characteristics
  - *total* generalization if each instance of the parent entity is an instance of at least one of the child entities, *partial* otherwise.
  - *exclusive* if each instance of the parent entity is at most one instance of one of the child entities, *overlapping* otherwise.

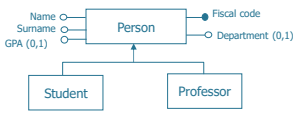
Generalization: incorrect example



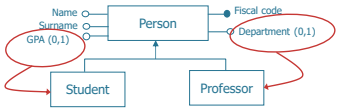
Generalization: incorrect example



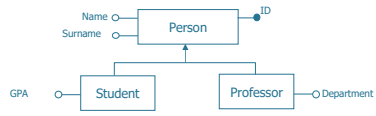
Generalization: incorrect example



Generalization: incorrect example

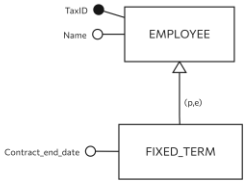


Generalization: correct example



Subset

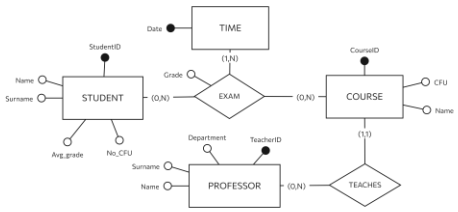
- Particular case of generalization with only one child entity
  - the generalization is always partial and exclusive



ER Model Documentation

Database design

Documenting E-R models



Documenting E-R models

Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data dictionary: example

Entity	Description	Attributes	Identifier
Student	University student	Student ID, Surname, Name, CFU acquired, Grades average	Student ID
Professor	University professor	Professor ID, Department, Surname, Name	Professor ID
Course	Courses offered by the university	Course code, Name, CFU	Course code
Time	Dates on which exams were taken	Date	Date

Data dictionary: example

Relationship	Description	Entities involved	Attributes
Exam	It associates a student to the exams taken and memorizes the mark obtained	Student (0,N), Course (0,N), Time (1,N)	Grade
Holder	It associates each course to the professor who teachers the course.	Course (1,1), Professor (0,N)	

Documenting E-R models

Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data Integrity Constraints

They cannot always be explicitly stated in an E-R scheme

They can be described in natural language

Data integrity constraints: examples

Integrity constraints	
RV1	The grade of an exam can only take values between 0 and 30
RV2	Each student cannot pass the same exam twice
RV3	A student may not take more than three exams for the same course during the same academic year

Documenting E-R models

Data Dictionary

Enrich the E-R schema with natural language descriptions of entities, relationships, and attributes

Data Integrity Constraints

They cannot always be explicitly stated in an E-R scheme

They can be described in natural language

Data Derivation Rules

Explicitly define that a concept of the schema can be obtained (by inference or arithmetic calculation) from other concepts of the schema

Derivation rules: examples

Derivation rules	
RD1	The number of credits acquired by a student is obtained by adding the number of credits of the courses for which the student has passed the exam
RD2	The average mark is obtained by calculating the average of the marks of the exams passed by a student

UML vs ER

Database design

78

UML and ER

UML (Unified Modeling Language)

- Modeling a software application
  - structural and behavioural aspects (data, operations, processes and architectures)
- Rich formalism
  - class diagram, actor diagram, sequence diagram, communication diagram, state diagram,...

ER

- Modeling a database
  - structural aspects of an application
- elements tailored to the modelling of a database

79

UML vs ER

- Different formalisms
- The class diagram of an application is different from the E-R schema of the database
- The class diagram, even if designed for different uses, can be adapted for the description of the conceptual design of a database
- Main Differences of UML vs ER
  - no standard notation to define identifiers
  - ability to add notes to comment on diagrams
  - possibility to indicate the direction of navigation of an association (not relevant in the design of a database)

80

Examples of time representation in the ER scheme

Database design

0

Time in E-R models

- Time is needed to represent:
  - Events
  - Temporal changes in values and/or relationships
- Time can be modelled using:
  - Temporal attributes
  - Binary relationships
  - Ternary relationships
  - Entity historicization

1

1

Temporal attributes

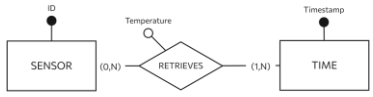
- Temporal information related to a single entity or relationship
- Unique events for each entity instance
  - Example: birth date, film release date

2

2

Binary relationship

- Example: A set of sensors are available, each identified by a unique code and present within a building.
  - It is requested store the different temperature values detected by each sensor at different time points.



3

3

Binary relationship

- We want to represent a time series of events related to an Entity E of the ER diagram
- The information of interest is represented by introducing
  - a ***Time*** entity
    - identified by time information about ***when*** the event occurs/starts (e.g. timestamp, date, date and time)
  - a **binary relationship *R*** that connects the ***Time*** entity to the ***E*** entity
    - The (if any) information on the duration and/or time of the end of the event and/or on other aspects that characterize the occurrence of the event at different time points are attributes of the relationship ***R***

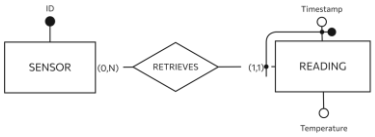
4

4

Workaround: Weak Entity

Example: A set of sensors are available, each identified by a unique code and present within a building.

- It is requested store the different temperature values detected by each sensor at different time points.

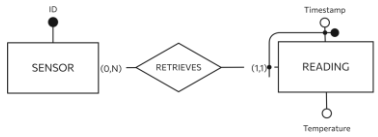


5

5

Workaround: Weak Entity

- The Event Entity **E**
  - is a weak entity identified internally by the set of attributes that represent the instant in which the event itself begins/occurs
  - the characteristics of the event are attributed directly to the weak entity **E**

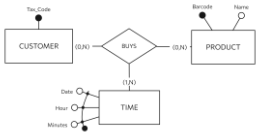


6

Ternary relationship

Example: It is requested to store the product purchases made by each customer.

- Each customer is uniquely identified by their tax code.
- Each product is uniquely identified by the barcode and characterized by the name.
- Suppose each customer can buy the same product at different times of the same day.



7

Ternary relationship

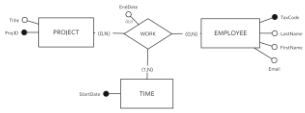
- We want to represent a time series of events expressed through an association/relationship between two Entities ***E1*** and ***E2*** of the ER diagram
- The information of interest is represented by introducing
  - a ***Time*** entity
    - identified by time information about ***when*** the event occurs/starts (e.g. timestamp, date, date and time)
  - a ***ternary relationship R*** that connects the ***Time*** entity, the ***E1*** entity, and the ***E2*** entity
    - The (if any) information on the duration and/or time of the end of the event and/or on other aspects that characterize the occurrence of the event at different time points are attributes of the relationship ***R***

8

Ternary Relationship

Example: A company that provides IT consulting wants to store the work done by its employees for each project.

- Each project is identified by an alphanumeric code and characterized by a title.
- Employees who work at the company are identified by their tax code and characterized by their first and last name and email.
- It is requested to store the time periods (start date, end date) in which an employee works on a project. Multiple employees can work at the same time on the same project.



9

Historicized Entity

Example: It is requested to store the lectures given by each teacher for each course.

- Each teacher is uniquely identified by a numeric ID and is characterized by surname and first name.
- Each course is identified by an alphanumeric code and characterized by its name.
- Each lesson is characterized by the date and time slot (start time and end time) in which it is held and by the course for which the lesson is delivered. Suppose that each teacher can deliver a maximum of one lecture in the same time slot.

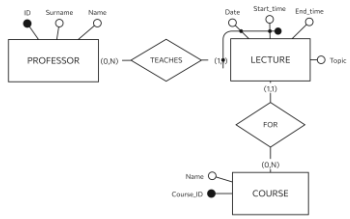
10

Historicized entity

- You want to represent
  - An event that involves two entities
  - There are constraints on an entity's participation in multiple events
- The information of interest is represented by introducing
  - a weak entity **E** identified internally by the time information about **when** the event occurs/starts (e.g. timestamp, date, date and time) and externally by the relationship linked to the entity that **cannot** participate in two events at the same time
  - the characteristics of the event are attributed directly to the weak entity **E**
  - entity **E** participates with cardinality (1,1) in relationships with other entities

11

Historicized entity



12

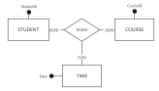
Recap

Binary

A student can take the exam for each course only once.

Translation:
STUDENT(StudentID)
COURSE(CourseID)
Exam(StudentID, CourseID, Date)

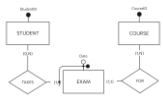
Ternary



A student can take the exam for each course multiple times on different dates.

Translation:
STUDENT(StudentID)
COURSE(CourseID)
~~TIME(Date)~~ Can be omitted
Exam(StudentID, CourseID, Date)

Historicized



A student can take the exam for each course multiple times. A student can only take one exam on a given date.

Translation:
STUDENT(StudentID)
COURSE(CourseID)
Exam(StudentID, Date, CourseID)

13

Politecnico di Torino

Logical Design

Database Design

0

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

1

1

Logical Design (2/2)

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

2

2

Logical Design

Introduction

3

3

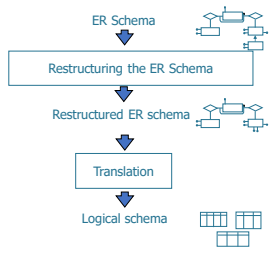
Logical design

Selection of the logical model	***Relational model***
Objective	***Definition*** of a relational logical schema corresponding to the starting ER schema
Important	***Simplification*** of the ER schema to make it representable by the relational model ***Optimization*** to increase the efficiency of queries

4

4

Logical design steps



5

5

Translation to the relational model

entities and many-to-many relationships

6

Translation to the relational model

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

7

Entity Translation

Translating the ER Schema into the Relational Model

8

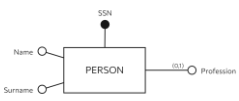
Entity Translation

- Each entity corresponds to a table with the same attributes
  - the attributes of the entity constitute the schema of the table
- The identifier (simple or composite) of the Entity becomes the primary key of the table
- Optional Entity attributes are attributes that can be NULL
  - They are highlighted with "*" in the table schema

9

Entity

Conceptual model



Logical model

Person(SSN, Name, Surname, Profession*)

- Underlined primary key
- Optional attributes indicated with an asterisk

10

Relationship translation

Translating the ER Schema into the Relational Logic Model

11

Relationship translation

- To translate a relationship
  1. Step 1: The Entities participating in the Relationship are first translated
  2. Step 2: The Relationship is then translated
     - Different translation rules for binary and ternary Relationships
     - For a Binary Relationship, it is necessary to consider the maximum and minimum cardinality with which the Entities participate in the Relationship

12

12

Translation of Binary Relationships

Translating the ER Schema into the Relational Model

13

13

Many-to-many binary relationship

Conceptual model

Logical model

Student(StudentID, Name, Surname)
Course(CodC, Name)
Exam(StudentID, CodC, Grade)

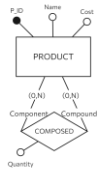
- Each many-to-many relationship corresponds to a table
  - The primary key is the combination of the identifiers of the two linked entities
  - The attributes of the table that corresponds to the relationship can be renamed (required in case of recursive relationships)

14

14

Recursive many-to-many binary relationship

Conceptual model



Logical model

Product (P_ID, Name, Cost)
Composed(CodCompound, CodComponent, Quantity)

- Each many-to-many relationship corresponds to a table
  - The primary key is the combination of the identifiers of the two linked entities
  - The attributes of the table that corresponds to the relationship can be renamed (required in case of recursive relationships)

15

15

One-to-many binary relationship

- Two translation modes are possible
  - by means of attributes
  - by means of a new table

16

16

One-to-Many Binary Relationship: using attributes

Conceptual model

Logical model

Person (SSN, Name, Surname, CityID, Date)
City (CityID, Name)

- It is used when participation of the entity that participates with a maximum cardinality of 1 is mandatory (minimum cardinality of 1)

17

17

One-to-many binary relationship: using attributes or a new table

Conceptual model

Logical model

Alternative 1: Translation using attributes

Student (StudentID, Name, Surname, NameUniv*, Date*)
University (Name, City)

Alternative 2: Translation using a new table

Student (StudentID, Name, Surname)
University (Name, City)
Graduation (StudentID, NameUniv, Date)

- It is used when participation of the entity that participates with a maximum cardinality of 1 is optional (minimum cardinality of 0)

18

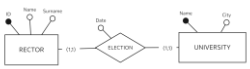
One-to-one binary relationship

- Multiple translations are possible
  - depends on the value of the minimum cardinality

19

One-to-one binary relationship

Conceptual model



Logical model

Alternative 1

Rector (ID, Name, Surname, UnivName, Date)
University (Name, City)

Alternative 2

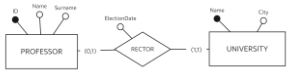
Rector (ID, Name, Surname)
University (Name, City, ID, Date)

- It is used when both entities participate with a maximum cardinality of 1 in the relationship, and participation is mandatory for both entities (minimum cardinality of 1)

20

One-to-one binary relationship

Conceptual model



Logical model

Professor (ID, Name, Surname)
University (Name, City, RectorID, ElectionDate)

- It is used when both entities participate with a maximum cardinality of 1 in the relationship, but participation is mandatory only for one entities (minimum cardinality of 1)

21

One-to-one binary relationship

Conceptual model

Logical model

Alternative 1

Professor (ID, Name, Surname)
University (Name, City)
Rector (RectorID, UniversityName, ElectionDate)

Alternative 2

Professor (ID, Name, Surname)
University (Name, City)
Rector (RectorID, UniversityName, ElectionDate)

Alternative 3

Professor (ID, Name, Surname)
University (Name, City, RectorID*, ElectionDate*)

- It is used when both entities participate with a maximum cardinality of 1 in the relationship, and participation is optional for both entities (minimum cardinality of 0)

22

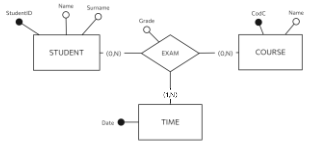
Translation of Ternary Relationships

Translating the ER Schema into the Relational Model

23

Ternary Relationship

Conceptual model



Logical model

Student(StudentID, Name, Surname)
Course(CodC, Name)
Time(Date)
Exam(StudentID, CodC, Date, Grade)

24

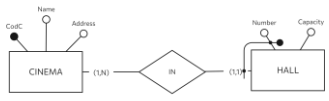
Translating Entities with External Identifier

Translating the ER Schema into the Relational Model

25

Entities with an external identifier

Conceptual model



Logical model

Cinema (CodC, Name, Address)
Hall (Number, CodC, Capacity)

- The relationship is represented together with the identifier
- The relationship contributes to the definition of the weak entity identifier

26

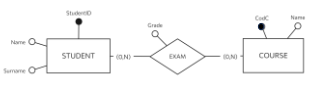
Referential integrity constraints

Translating the ER Schema into the Relational Model

27

Referential integrity constraints

Conceptual model



Logical model

Student(StudentID, Name, Surname)
Course(CodC, Name)
Exam(StudentId, CodC, Grade)

- Relationships Represent Referential Integrity Constraints

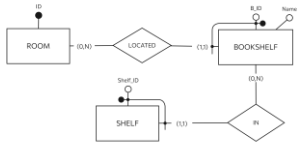
  Exam(StudentID) REFERENCES Student(StudentID)

  Exam(CodC) REFERENCES Course(CodC)

28

Referential integrity constraints

Conceptual model



Logical model

Room (ID)
Bookshelf (ID, B_ID, Name)
Shelf (ID, B_ID, Shelf_ID)

- If the referenced key consists of multiple attributes, the referential integrity constraint is imposed on the attribute set

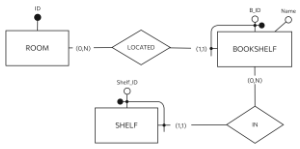
  Bookshelf(ID) REFERENCES Room(ID)

  Shelf (ID, B_ID) REFERENCES Bookshelf (ID, B_ID)

29

Referential integrity constraints

Conceptual model



Logical model

Room (ID)
Bookshelf (*ID*, B_ID, Name)
Shelf (*ID*, *B_ID*, Shelf_ID)

- If the referenced key consists of multiple attributes, the referential integrity constraint is imposed on the attribute set

Bookshelf(ID) REFERENCES Room(ID)
Shelf (ID) REFERENCES Bookshelf (ID)
Shelf(B_ID) REFERENCES Bookshelf (B_ID)

Wrong constraints!

30

Restructuring the ER model

Restructuring the ER model

31

Restructuring the ER model

- The restructured ER model takes into account implementation aspects
  - It is no longer a conceptual model
- Objectives
  - To eliminate constructs for which there is no direct representation in the relational model
  - To transform the data representation in order to increase the efficiency of data access operations

32

Restructuring tasks

- Eliminating composite attributes
- Eliminating multivalued attributes
- Eliminating generalizations
- Analysis of redundancies
- Partitioning concepts (Entities, Relationships)
- Choosing primary identifiers

33

Eliminating composite attributes

Restructuring the ER model

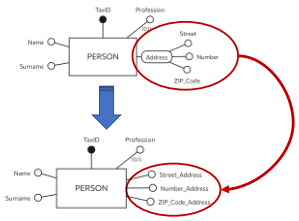
34

Eliminating composite attributes

- Composite attributes are not representable in the relational model.
- Attributes can be deleted by:
  - **separately representing individual sub-attributes**
    - if you need to access each attribute separately
  - **Introducing a single attribute that represents the concatenation of the composite attributes**
    - if access to the overall information is sufficient

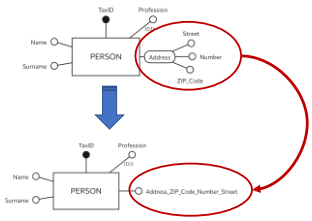
35

Option 1: separate attributes



36

Option 2: single attribute



37

Eliminating multivalued attributes

Restructuring the ER model

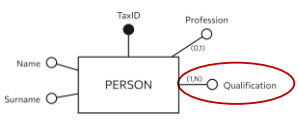
38

Eliminating multivalued attributes

- They cannot be represented in the relational model
- Multivalued attributes are represented using a relationship between:
  - the initial entity
  - a new entity
- Pay attention to the cardinality of the new relationship

39

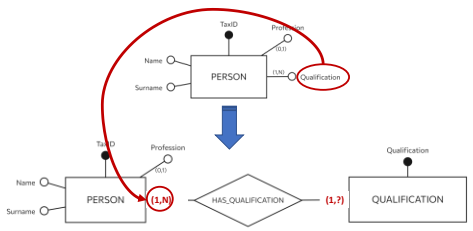
Shared information



A person can have more than one educational qualification and that the same educational qualification can be held by several people

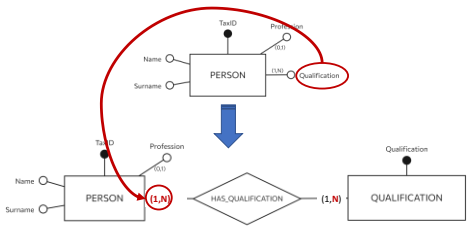
40

Shared information: *Has_qualification* cardinality



41

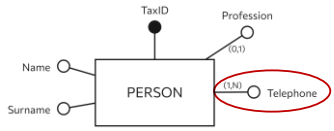
Shared information: *Has_qualification* cardinality



A person can have more than one educational qualification and that the same educational qualification can be held by several people

42

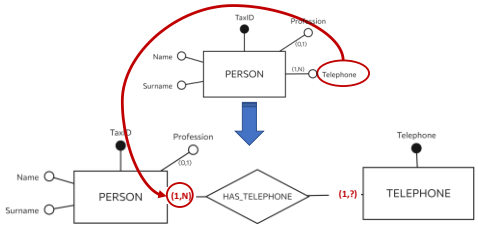
Unique information



A person can have more than one telephone number, but a given telephone number can be held only by one person

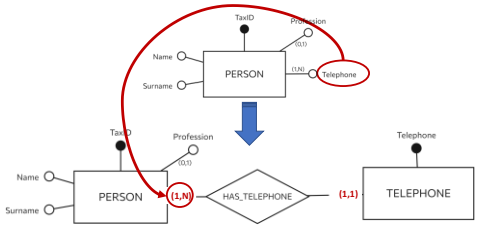
43

Unique information: *Has_telephone* cardinality



44

Unique information: *Has_telephone* cardinality



A person can have more than one telephone number, but a given telephone number can be held only by one person

45

Removing generalizations

Restructuring the ER model

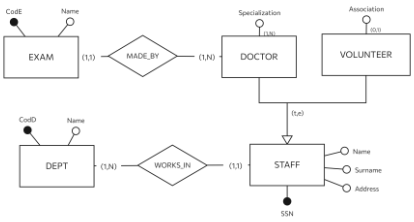
46

Removing generalizations

- 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 relationships

47

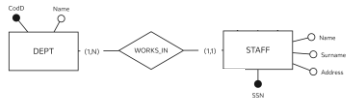
Example



48

48

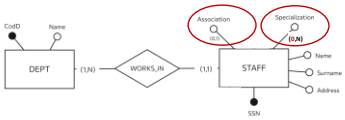
Merging child entities into the parent entity



49

49

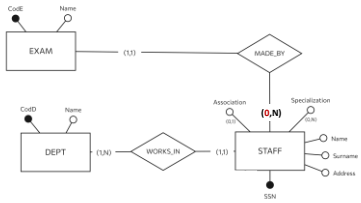
Attributes of child entities



50

50

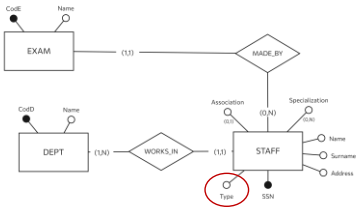
Relationships with child entities



51

51

The «Type» attribute

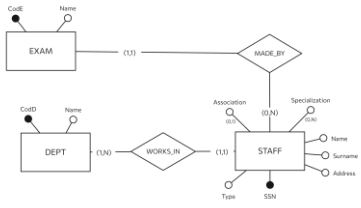


- The Type attribute indicates the original entity (doctor or volunteer) to which each occurrence of the parent entity (staff) belongs

52

52

Merging child entities into the parent entity

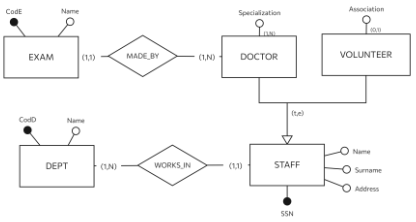


- Can be used for all types of generalization
  - in case of overlapping entities, many combinations are possible as Type values, e.g., skier and sailor

53

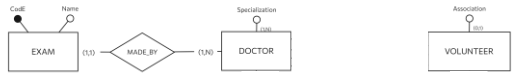
53

Back to the example



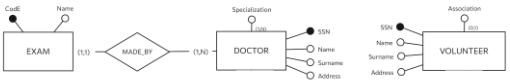
54

Merging the parent into the child entities



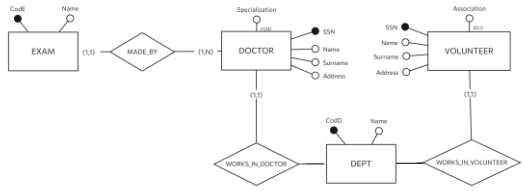
55

Attributes of the parent entity



56

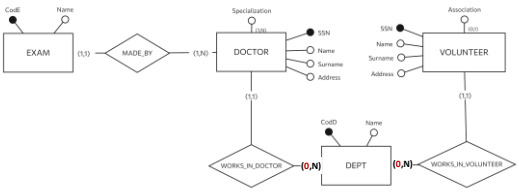
Relationships with parent entity



- Relationships with the parent entity need to be split

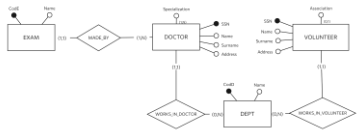
57

Cardinality of *Works in* relationship



58

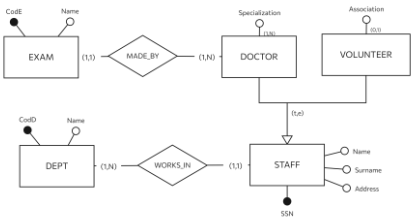
Merging the parent into the child entities



- Cannot be used for **partial** generalizations
  - however, generalizations can be transformed from partial to total by adding a new entity Others
- Cannot be used for **overlapping** generalizations
  - due to duplicate identifiers

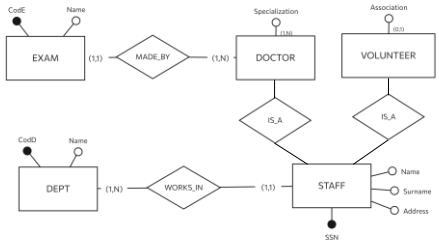
59

Back to the original example



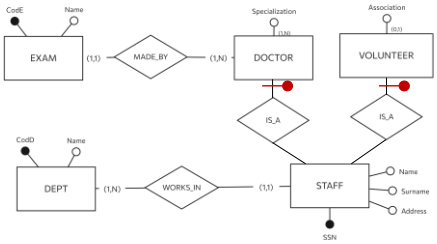
60

Generalization translated into a relationship



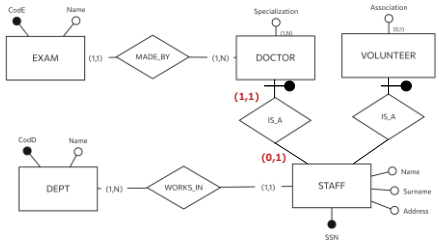
61

Child entities' identifier



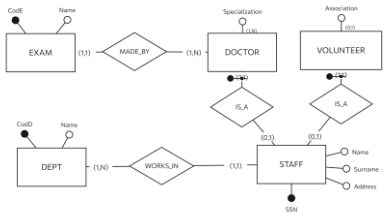
62

Cardinality of *is a* relationship



63

Generalization translated into relationships



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

64

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)
- Merging parent entity into child entities is appropriate when:
  - the generalization is total
  - there are operations that refer only to specific child entities and therefore it is useful to distinguish between different child entities (optimize data access)
- "Mixed" representations are also possible:
  - there are operations that refer only to instances of some child entities (optimize data access)
- In the presence of hierarchical generalization, apply the same procedure, starting from the lower levels

65

Redundancy analysis

Restructuring the ER model

66

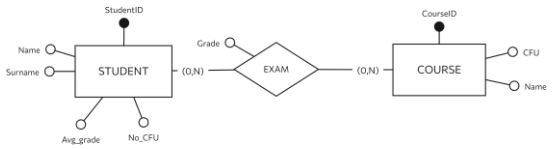
Redundancy analysis

- They represent information that is relevant to the application, but can be derived from other concepts
  - it must be decided whether to keep them
- Effects of redundancies on the logical schema
  - simplifying and speeding up queries
  - increased complexity and slower updates
  - increased storage requirements

67

Redundant attribute example

- The Avg_grade attribute is redundant:
  - Useful for speeding up queries that require calculating the average of students' grades
  - if preserved, the relational schema must be supplemented with proper documentation that the attribute is redundant (and derivation rules)



68

Partitioning concepts

Restructuring the ER model

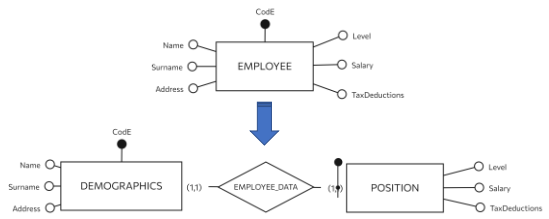
69

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

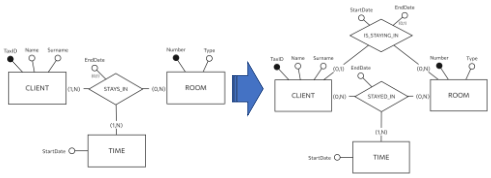
70

Entity partitioning



71

Relationship partitioning



72

Choosing Primary Identifiers

Restructuring the ER model

73

Selection of primary identifiers

- It is necessary to define the relation *primary keys*
- 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»

74

Politecnico di Torino

Database design

Example

0

Conceptual design (1/2)

- Database design stages
- Design example: problem specifica
- Design example: main concepts
- Design example: model refinement (I)
- Design example: model refinement (II)
- Design example: model refinement (III)

1

1

Conceptual Design (2/2)

- Design example: representation of time (I)
- Design example: representation of time (II)
- Design example: representation of time (III)

2

2

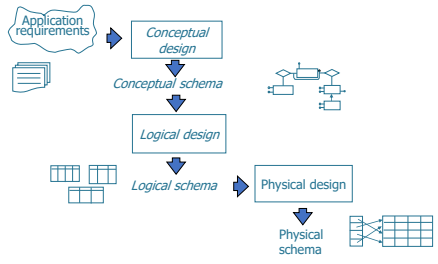
Database design

Database design stages

3

3

Database design stages



4

4

Requirements collection and analysis

- Requirements collection
  - problems that the application must solve
  - static and dynamic features of the application
- Requirements analysis
  - clarification and organization of the specifications
- Interconnected and scarcely standardizable activities

5

5

Requirements sources

- Application users
  - interviews
  - written documentation
- Existing documentation
  - regulations
  - internal rules
  - forms
- Existing systems
  - applications to replace or interact with

6

Requirements collection

- System users play an important role
  - high-level users have a more general view, but they often do not know the details
  - different users can provide different information (complementary or contradictory)

7

Requirements collection

- In practice:
  - verify that the collected information is consistent and correct
  - verify also with examples (related to general and borderline cases)
  - require precise definitions and classifications
  - distinguish the essential requirements from nice-to-have and less stringent requirements
  - proceed through subsequent refinements

8

Requirements analysis

- In practice:
  - choose the right abstraction level
  - standardize the structure of the sentences
  - avoid convoluted sentences
  - identify synonyms/homonyms and unify the terms
  - make explicit all references between terms
  - build a glossary of terms

9

Conceptual design

- Various project strategies have been proposed
- The most effective strategy is hybrid
  - the basic concepts are identified (the most relevant entities and relationships)
  - the initial project is progressively refined by adding the attributes, the cardinality of the relationships, the hierarchies, other entities and relationships
- If the problem is highly complex, it can be broken down into subproblems to solve separately and integrate later

10

Conceptual design: general criteria

- If a concept has significant properties or describes classes of objects with autonomous existence
  - entity
- If a concept has a simple structure and has no relevant properties
  - attribute (possibly multivalued)
- If two or more concepts are related
  - relation
- If a concept is a particular case of another one
  - hierarchy

11

Quality of a conceptual scheme

- Correctness
  - use of appropriate model constructs
  - absence of syntactic and semantic errors
- Completeness
  - representation of all the concepts of interest
- Minimality
  - every requirement is represented only once in the schema
  - all redundancies are verified and documented
- Legibility

12

12

Database design

Problem requirements

13

13

Problem requirements

- We want to represent a database for the management of a medical examination booking system within a Local Health Authority (ASL), considering the following information.
- Each patient is characterized by the health card number, name, surname, address, date of birth, place of birth and age.
- ASL hospitals are characterized by a numerical code, name and address.

14

14

Problem requirements

- Each hospital is divided into departments identified by a unique numeric code within the hospital and characterized by department name and telephone number.
- The department staff is identified by a Social Security Number (SSN). Name, surname and address are also known.
- Among the staff, for doctors the list of specializations achieved is known and for the volunteers the name of the association they belong to (if available) is stored.

15

15

Problem requirements

- The medical examinations that can be performed are characterized by a numerical code and a textual description (e.g., X-ray exam, etc.).
- For specialistic examinations, the doctor who carries out the visit and a description of the diet to follow (if necessary) are also stored.
- The laboratories that perform the examinations are identified by a unique code within the hospital and they are characterized by the name of the laboratory, the location plan and the room number.

16

16

Problem requirements

- For each member of the laboratory staff, the days and laboratories in which he/she works are stored. Pay attention to the fact that during the same day each staff member can work in more than one laboratory.

17

17

Problem requirements

- Each exam requires a reservation. For each exam reservation made by a patient, the date and time of the exam, the laboratory where it is performed, the cost of the ticket and the information about the exam being urgent or not are stored.
- Please note that each patient can make multiple reservations for the same exam on different dates and the same exam cannot be repeated on the same day by the same patient, even in different laboratories.

18

Problem requirements

- Each doctor can take on different roles during his/her career (e.g. assistant, head physician, etc.). We want to keep track of the roles each doctor has taken on during his/her career and the related time periods (start date, end date).
- Keep in mind that each doctor cannot take on more than one role at the same time, but he/she can take on the same role in different time periods.

19

Database design

Main concepts

20

Main concepts identification

- Text analysis to identify the most important concepts
  - the main entities of the ER diagram
  - any links between entities

21

Patient Concept

- Each *patient* is characterized by health card number, name, surname, address, date of birth, place of birth and age.

22

Patient Concept

23

Hospital concept

- ASL *hospitals* are characterized by a numeric code, a name and an address.

24

24

Hospital concept

25

25

Department concept

- Each hospital is divided into *departments* identified by a unique numeric code within the hospital and characterized by the department name and the telephone number.

26

26

Department concept

27

27

Staff concept

- The *department staff* is identified by a Social Security Number (SSN).
- Name, surname and address are also known.
- Among the staff, for doctors the list of specializations achieved is known and for the volunteers the name of the association they belong to (if available) is stored.

28

28

Staff concept

29

29

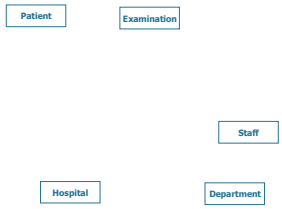
Examination concept

- The *medical examinations* that can be performed are characterized by a numerical code and a textual description (e.g. X-ray exam, etc.).
- For specialist examinations, the doctor who carries out the visit and a description of the diet to follow (if necessary) are also stored.

30

30

Examination concept



31

31

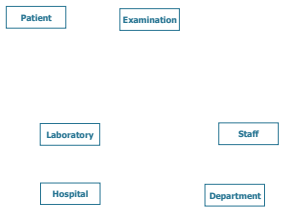
Laboratory concept

- The *laboratories* that perform the tests are identified by a unique code within the hospital and are characterized by the name of the laboratory, the location plan and the room number.

32

32

Laboratory concept

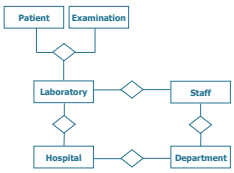


33

33

Main concepts

- Main concepts
  - patient
  - examination
  - laboratory
  - hospital
  - department
  - staff



34

34

Database design

Model refinement (I)

35

35

Concepts refinement

- Refinement of concepts
  - introduction of hierarchies
  - attributes definition
  - characterization of relationships with cardinality

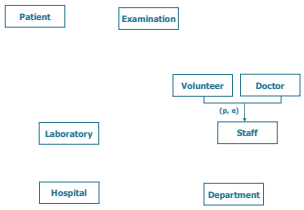
36

Staff hierarchy

- The department staff is identified by a Social Security Number (SSN). Name, surname and home address are also known.
- Among the staff, for *doctors* the list of specializations achieved is known and for the *volunteers* the name of the association they belong to (if available) is stored.

37

Staff hierarchy



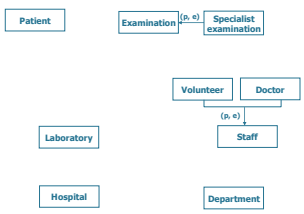
38

Exam hierarchy

- The medical examinations that can be performed are characterized by a numerical code and a textual description (e.g. radiography, etc.). For *specialist examinations*, the doctor who carries out the visit and a description of the diet to follow (if necessary) are also stored.

39

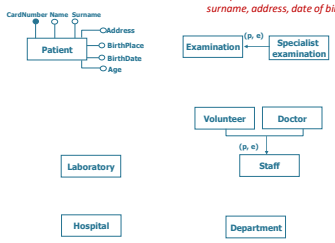
Exam hierarchy



40

Refinement of the Patient entity

Each patient is characterized by health card number, name, surname, address, date of birth, place of birth and age.



42

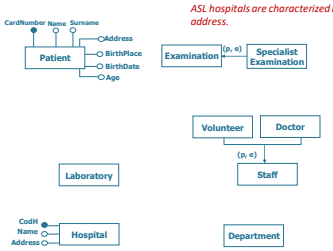
Date of birth and Age Attributes

- The Age attribute is redundant because it can be easily calculated starting from the date of birth (BirthDate)
- This information must be attached to the conceptual model documentation
  - Age derivation rule from BirthDate: Age = Year (Today () - BirthDate)
- Elimination of the Age attribute will be evaluated during the restructuring phase of the ER scheme

43

Refinement of the Hospital entity

ASL hospitals are characterized by a numerical code, name and address.



45

Database design

Model refinement (II)

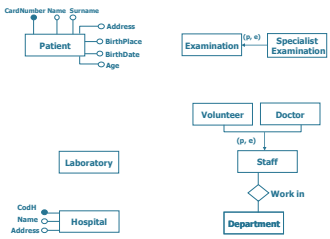
46

Relationship between staff and department

- *The department staff is identified by a Social Security Number (SSN). Name, surname and address are also known.*
  Among the staff, for doctors the list of specializations achieved is known and for volunteers the name of the association they belong to (if available) is stored.

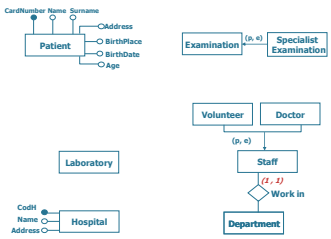
47

Relationship between staff and department



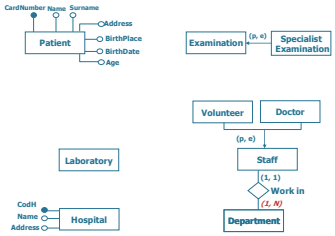
48

Cardinality of the *work in* Relationship



49

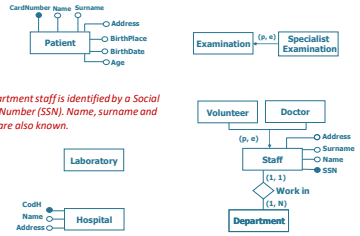
Cardinality of the *work in* Relationship



50

50

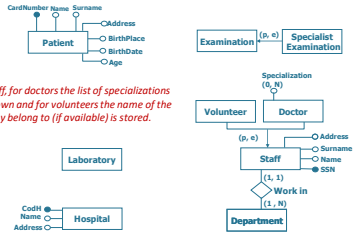
Refinement of the Staff entity



51

51

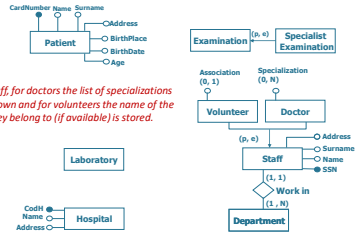
Refinement of the Doctor and Volunteer entities



53

53

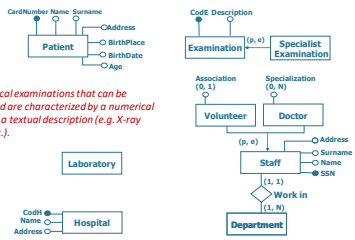
Refinement of the Doctor and Volunteer entities



54

54

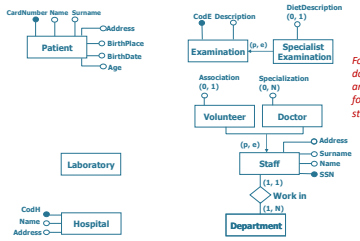
Refinement of the Examination entity



56

56

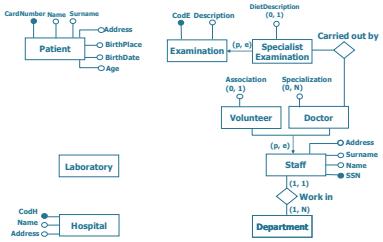
Refinement of the Specialistic Examination entity



58

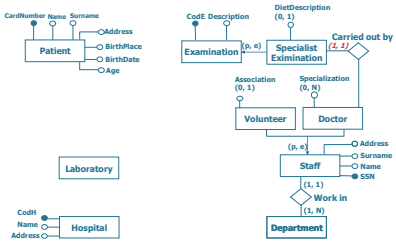
58

Refinement of the Specialistic Examination entity



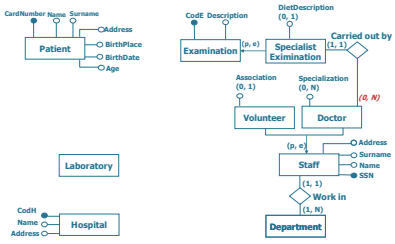
59

Cardinality of the *Carried out by* relationship



60

Cardinality of the *Carried out by* relationship



61

Database design

Model refinement (III)

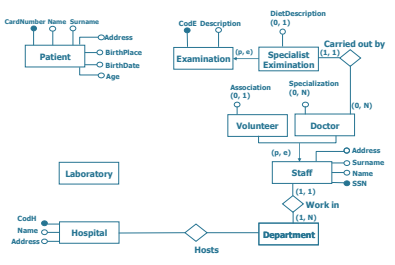
62

Relationship between Department and Hospital

- Each hospital is divided into departments identified by a unique numeric code within the hospital and characterized by the department name and the telephone number.

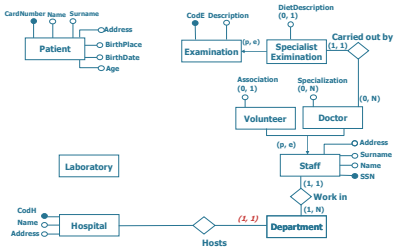
63

Relationship between Department and Hospital



64

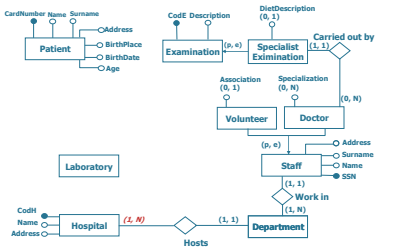
Cardinality of the *Hosts* relationship



65

65

Cardinality of the *Hosts* relationship



66

66

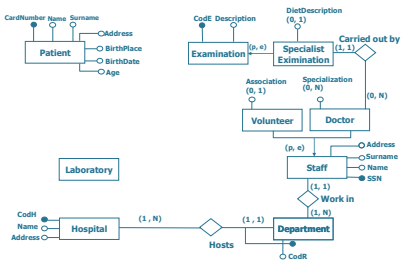
Department entity identifier

- Each hospital is divided into departments identified by a unique numeric code within the hospital and characterized by the department name and the telephone number.

67

67

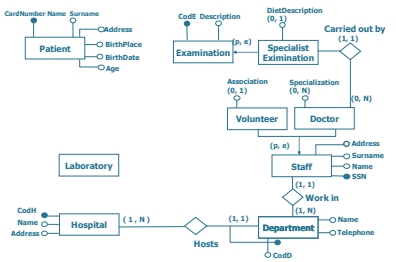
Department entity identifier



68

68

Refinement of the Department entity



69

69

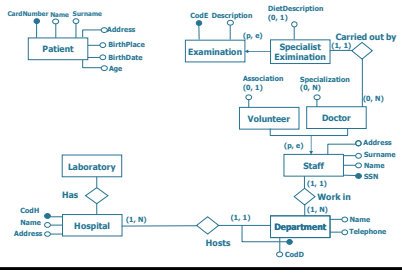
Relationship between Laboratory and Hospital

- The laboratories that perform the tests are identified by a unique code within the hospital and are characterized by the name of the laboratory, the location plan and the room number.

70

70

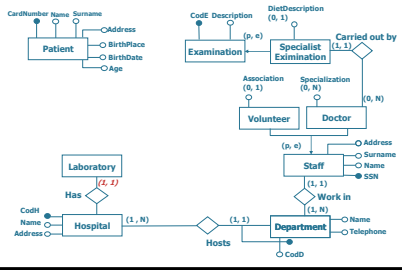
Relationship between Laboratory and Hospital



71

71

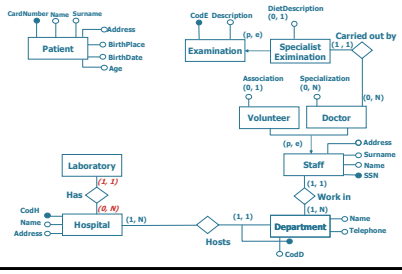
Cardinality of the *Has* relationship



72

72

Cardinality of the *Has* relationship



73

73

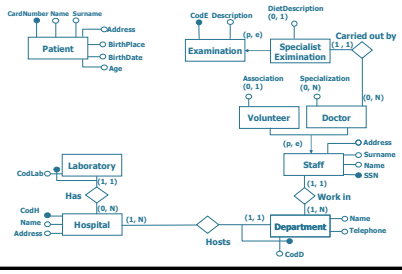
Relationship between Laboratory and Hospital

- The laboratories that perform the tests are identified by a unique code within the hospital and they are characterized by the name of the laboratory, the location plan and the room number.

74

74

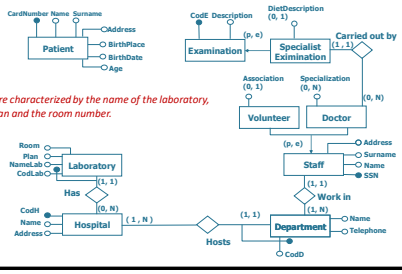
Laboratory entity identifier



75

75

Refinement of the Laboratory entity



76

76

Database design

Representation of time (I)

77

Time representation

- It is necessary to explicitly represent time-related information in the case of
  - representation of events
  - changes in the information content of entities or time attributes
- Various patterns are possible
  - by means of N-ary relationship with a time entity
  - through historicized entities
  - through binary relationships with a time entity

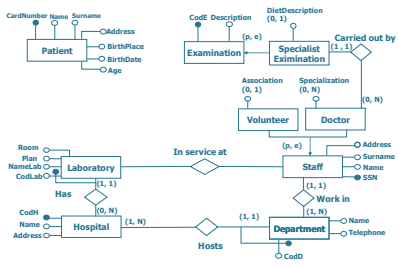
78

Relationship between Staff and Laboratory

- For each member of the laboratory staff, the days and laboratories in which he/she works are stored. It should be borne in mind that during the same day each member of the staff can work in several laboratories.

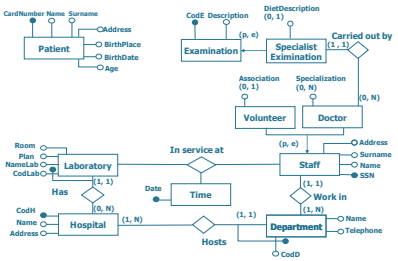
79

Relationship between Staff and Laboratory



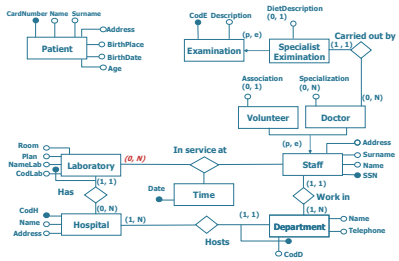
80

Relationship between Staff and Laboratory



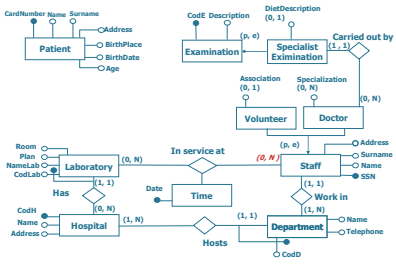
81

Cardinality of the *In service at* relationship



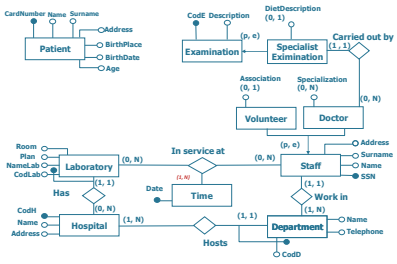
82

Cardinality of the *In service at* relationship



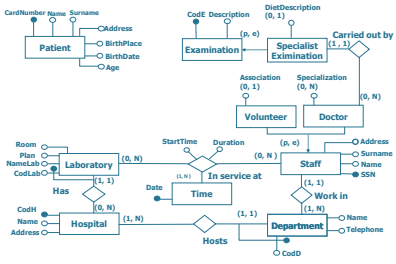
83

Cardinality of the *In service at* relationship



84

Cardinality of the *In service at* relationship



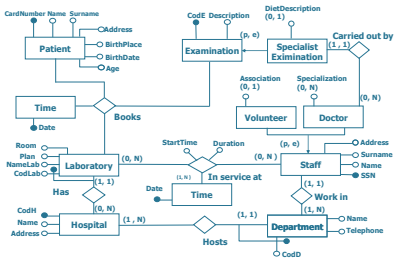
85

Representation of the reservation

- Each exam requires a reservation.
- For each exam reservation made by a patient, the date and time of the exam, the laboratory where it is performed, the cost of the ticket and the information about the exam being urgent or not are stored.
- Please note that each patient can make multiple reservations for the same exam on different dates and the same exam cannot be repeated on the same day by the same patient, even in different laboratories.

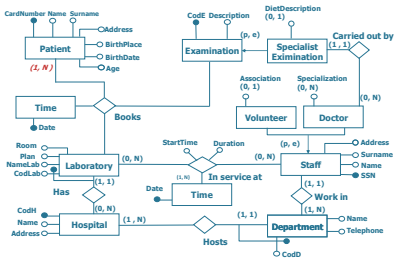
86

Representation of the reservation



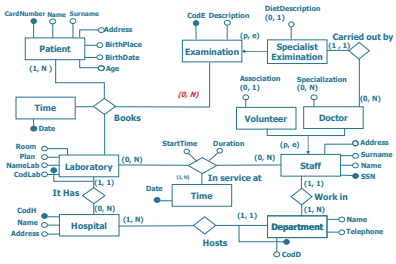
87

Cardinality of the *Books* relationship



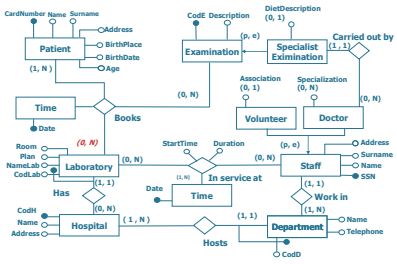
88

Cardinality of the *Books* relationship



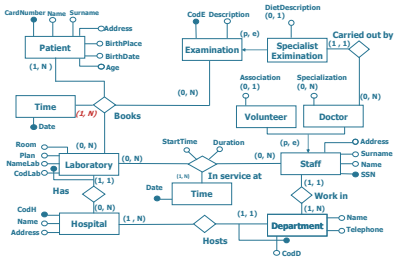
89

Cardinality of the *Books* relationship



90

Cardinality of the *Books* relationship



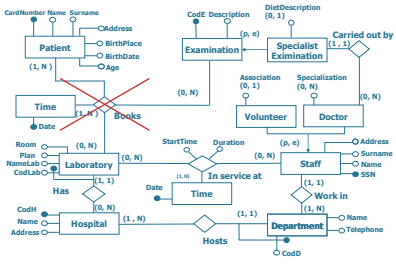
91

Reservation constraints

- Each exam requires a reservation.
- For each exam reservation made by a patient, the date and time of the exam, the laboratory where it is performed, the cost of the ticket and the information about the exam being urgent or not are stored.
- *Please note that each patient can make multiple reservations for the same exam on different dates and the same exam cannot be repeated on the same day by the same patient, even in different laboratories.*

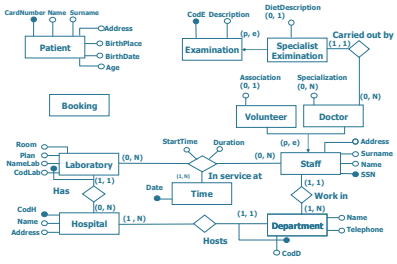
92

Representation of the reservation



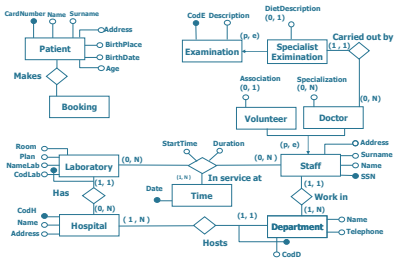
93

Introducing the Booking entity



94

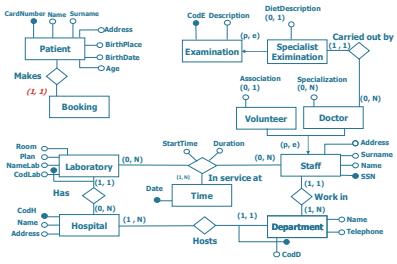
Relationship between Booking and Patient



95

95

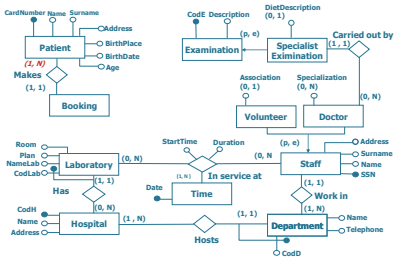
Cardinality of the *Makes* relationship



96

96

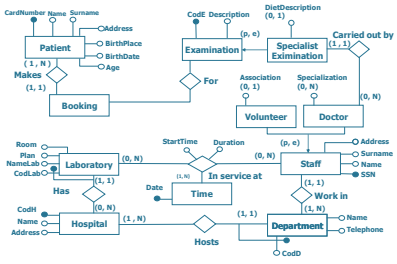
Cardinality of the *Makes* relationship



97

97

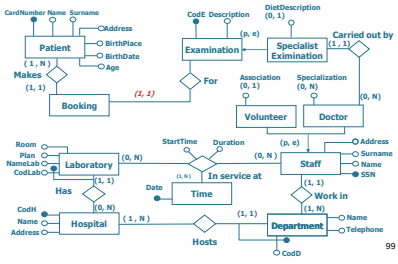
Relationship between Booking and Patient



98

98

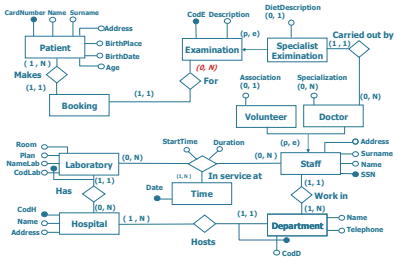
Cardinality of the *For* relationship



99

99

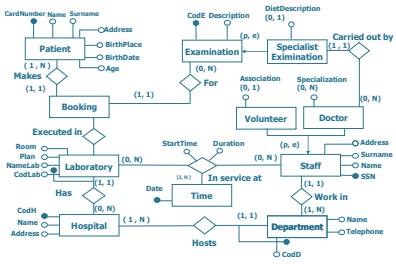
Cardinality of the *For* relationship



100

100

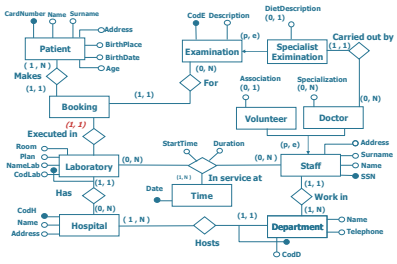
Relationship between Booking and Laboratory



101

101

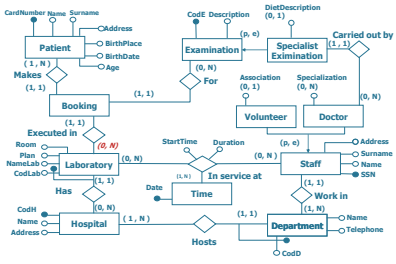
Cardinality of the *Executed in* relationship



102

102

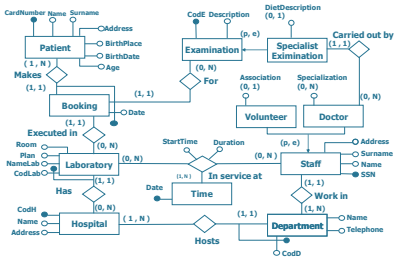
Cardinality of the *Executed in* relationship



103

103

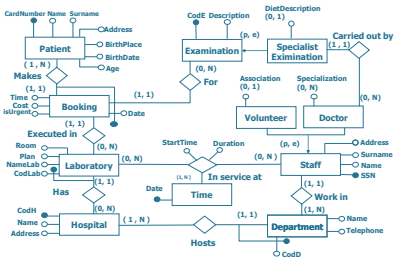
Booking entity identifier



104

104

Refinement of the Booking entity



105

105

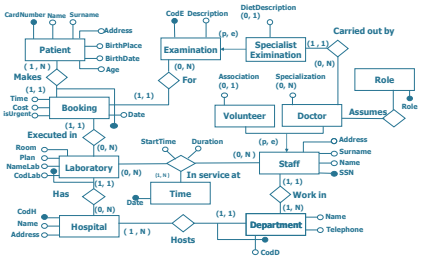
Relationship between Doctor and Role

- Each doctor can take on different roles during his/her career (e.g. assistant, head physician, etc.).
  We want to keep track of the roles each doctor has taken on during his/her career and the related time periods (start date, end date).
  Keep in mind that each doctor cannot take on more than one role at the same time, but he/she can take on the same role in different time periods.

106

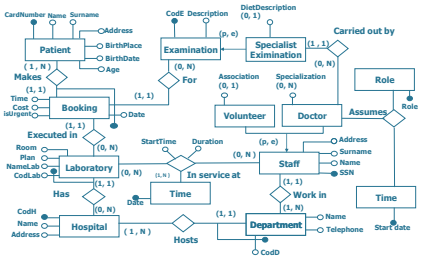
106

Relationship between Doctor and Role

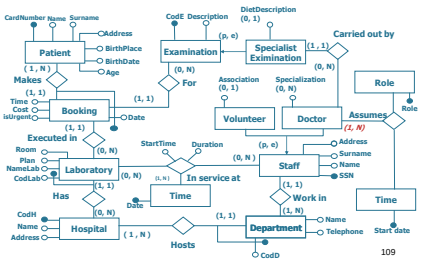


107

Relationship between Doctor and Role

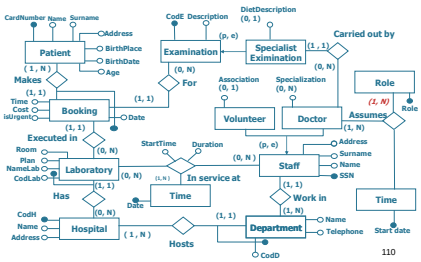


108



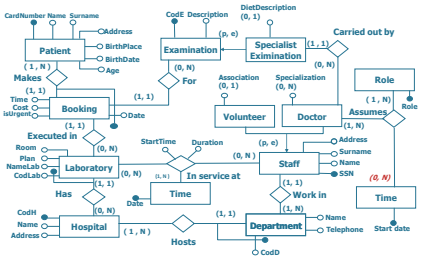
109

Cardinality of the *Assumes* relationship



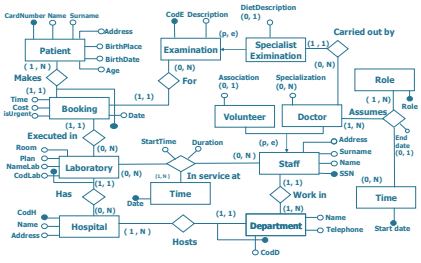
110

Cardinality of the *Assumes* relationship



111

Cardinality of the *Assumes* relationship



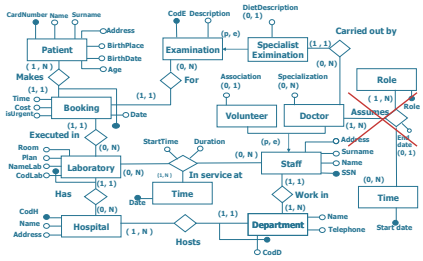
112

Constraints on the *Assumes* relationship

- Each doctor can take on different roles during his/her career (e.g. assistant, head physician, etc.).
  We want to keep track of the roles each doctor has taken on during his/her career and the related time periods (start date, end date).
  Keep in mind that each doctor cannot take on more than one role at the same time, but he/she can take on the same role in different time periods.

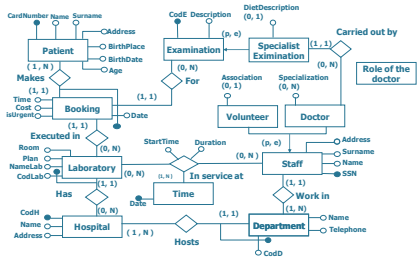
113

Historicization of the role of the doctor



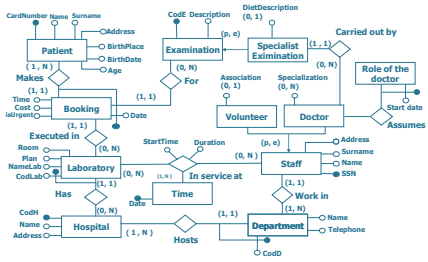
114

Historicization of the role of the doctor



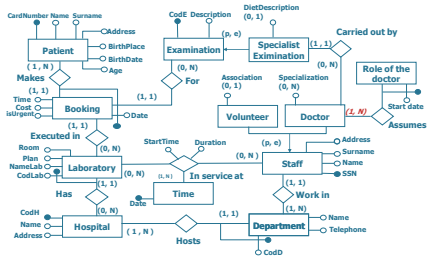
115

Historicization of the role of the doctor



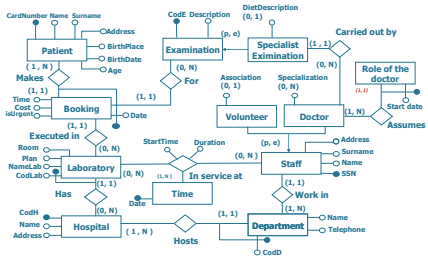
116

Historicization of the role of the doctor



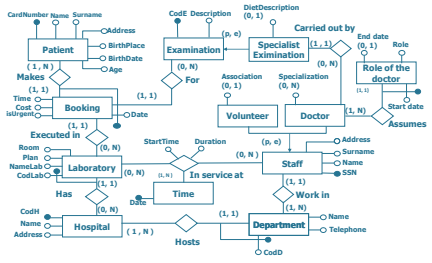
117

Historicization of the role of the doctor



118

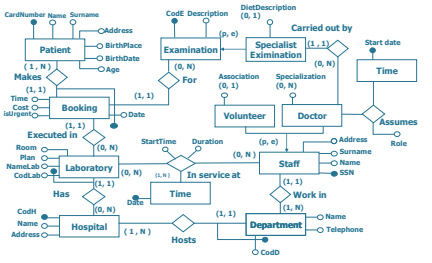
Historicization of the role of the doctor



119

119

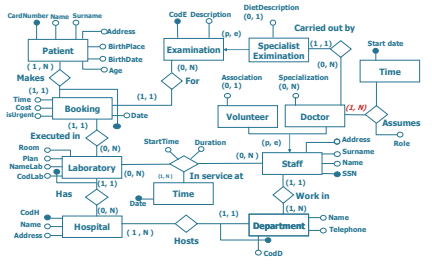
Role history: alternative modeling



120

120

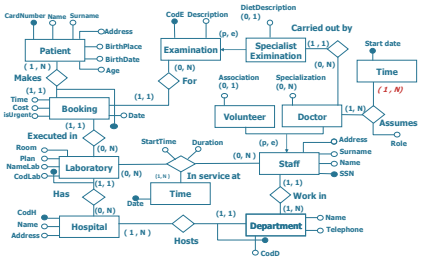
Cardinality of the *Assumes* relationship



121

121

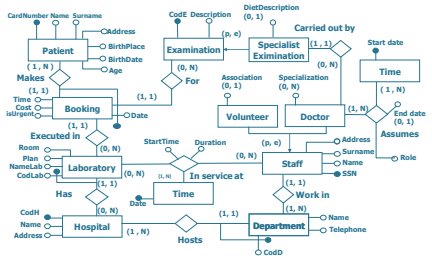
Cardinality of the *Assumes* relationship



122

122

Cardinality of the *Assumes* relationship



123

123

Politecnico di Torino

Logic design

Example

0

Example of relational logic design

- Introduction
- ER schema restructing
- Translation of the entities without an external identifier
- Translation of the entities with an external identifier
- Translation of the relationships

1

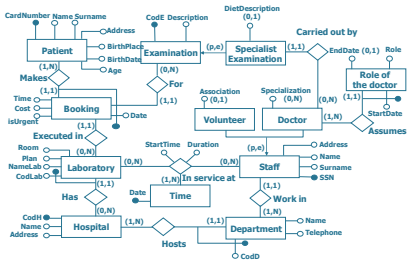
Politecnico di Torino

Introduction

Example of relational logic design

2

Starting conceptual model



3

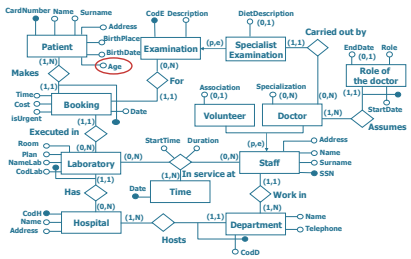
Politecnico di Torino

Restructing the ER model

Example of relational logic design

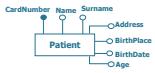
4

Analysis of the derived attributes



5

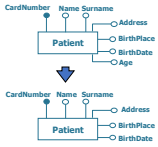
Derived Age attribute



- The Age attribute can be removed since
  - it can be easily calculated from the date of birth (BirthDate)
  - it is not generally present in a query

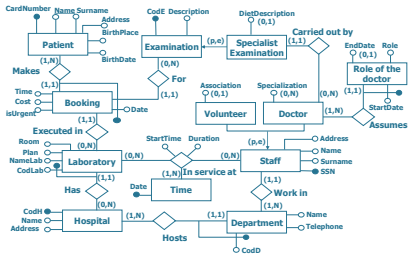
6

Eliminating the Age attribute



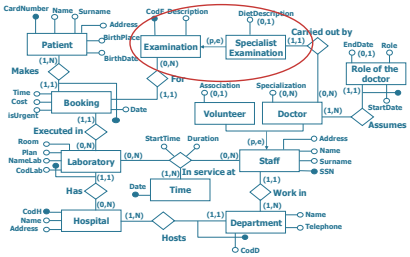
7

Restructured schema (n. 1)



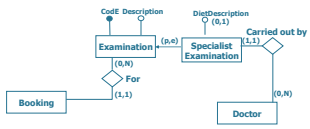
8

Removing generalizations



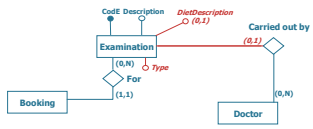
9

Removing generalizations



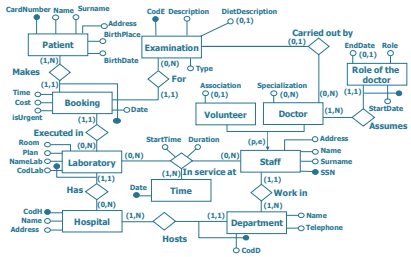
10

Merge into the parent entity



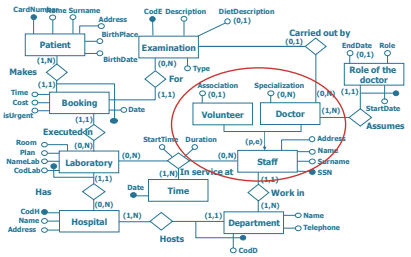
11

Restructured schema (n.2)



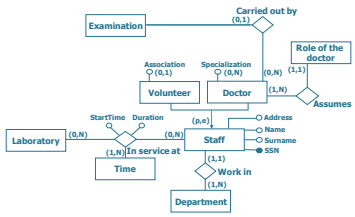
12

Removing generalizations



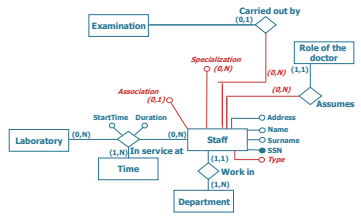
13

Staff generalization



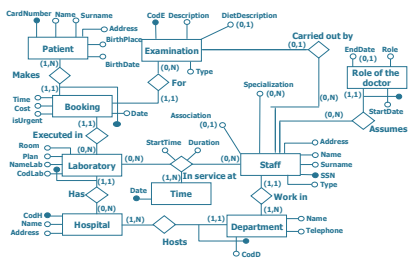
14

Merge into the parent entity



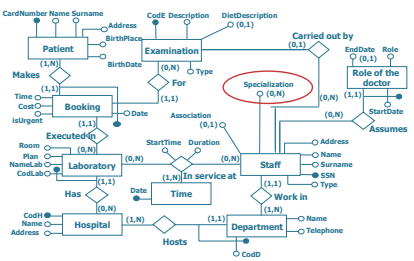
15

Restructured schema (n.3)



16

Removing multivalued attributes

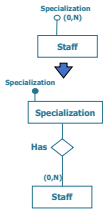


17

Removing multivalued Spacialization attribute

18

New specialization entity

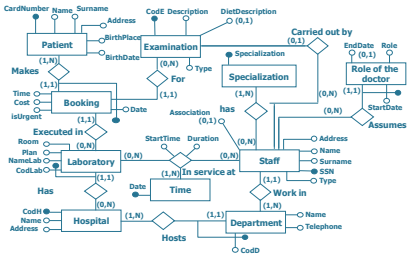


19

Cardinality of the *Has* relationship

20

Restructured schema (final)



21

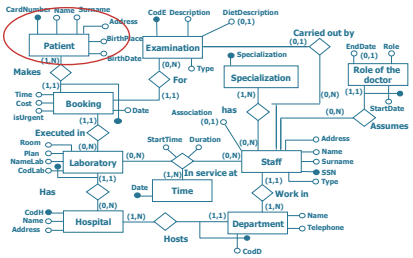
Politecnico di Torino

Traslating entities without an external identifier

Example of relational logic design

22

Translation of the Patient entity



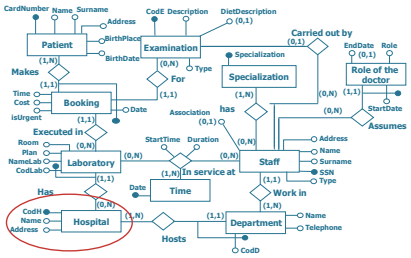
23

Translation of the Patient entity

Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)

24

Translation of the Hospital entity



25

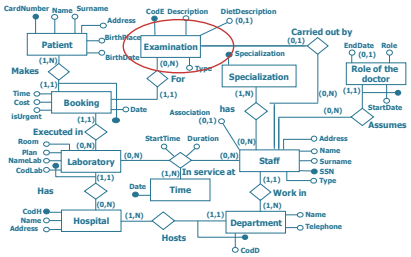
Translation of the Hospital entity

Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)

Hospital(<u>CodH</u>, Name, Address)

26

Translation of the Examination entity



27

Translation of the Examination entity

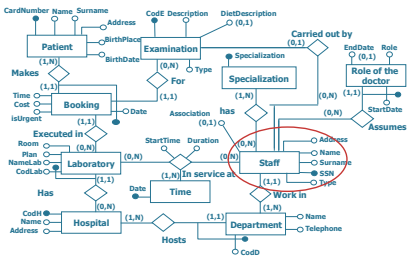
Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)

Hospital(<u>CodH</u>, Name, Address)

Examination(<u>CodE</u>, Description, DietDescription, Type)*

28

Translation of the Staff entity



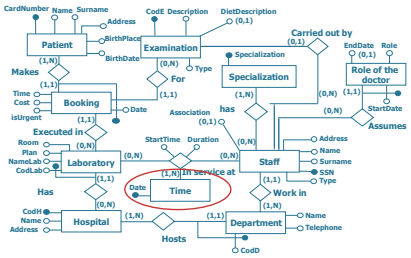
29

Translation of the Staff entity

Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(<u>CodH</u>, Name, Address)
Examination(<u>CodE</u>, Description, DietDescription*, Type)
Staff(<u>SSN</u>, Name, Surname, Address, Association,Type)*

30

Translation of the Time entity



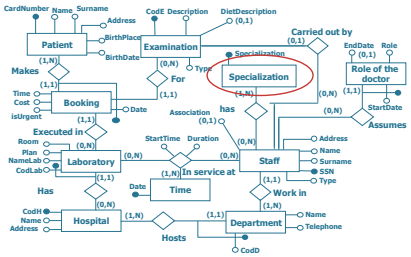
31

Translation of the Time entity

Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(<u>CodH</u>, Name, Address)
Examination(<u>CodE</u>, Description, DietDescription*, Type)
Staff(<u>SSN</u>, Name, Surname, Address, Association*,Type)
Time(<u>Date</u>)

32

Translation of the Specialization entity



33

Translation of the Specialization entity

Patient(<u>CardNumber</u>, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(<u>CodH</u>, Name, Address)
Examination(<u>CodE</u>, Description, DietDescription*, Type)
Staff(<u>SSN</u>, Name, Surname, Address, Association*,Type)
Time(<u>Date</u>)
Specialization(<u>Specialization</u>)

34

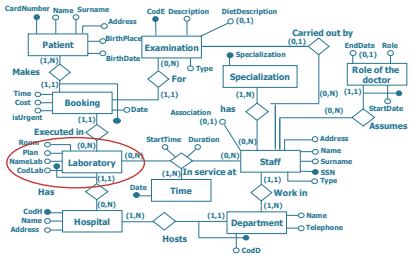
Politecnico di Torino

Translating entities with an external identifier

Example of relational logic design

35

Translation of the Laboratory entity



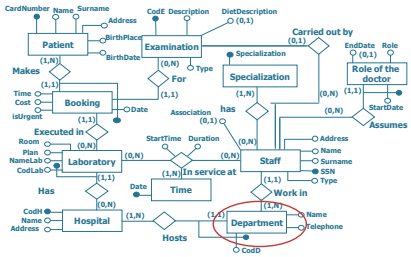
36

Translation of the Laboratory entity

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type)
Staff(SSN, Name, Surname, Address, Association*,Type)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)

37

Translation of the Department entity



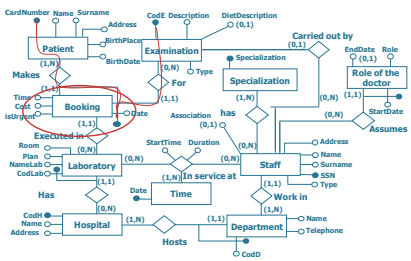
38

Translation of the Department entity

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type)
Staff(SSN, Name, Surname, Address, Association*,Type)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)

39

Translation of the Booking entity



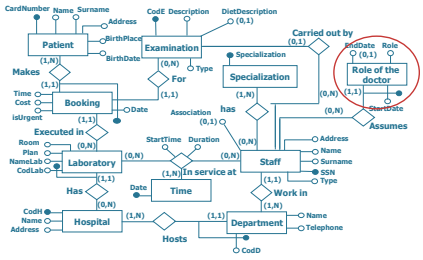
40

Translation of the Booking entity

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type)
Staff(SSN, Name, Surname, Address, Association*,Type)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent)

41

Translation of the Role of the doctor entity



42

Translation of the Role of the doctor entity

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type)
Staff(SSN, Name, Surname, Address, Association*,Type)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent)
DoctorRole(SSN, StartDate, EndDate, Role)*

43

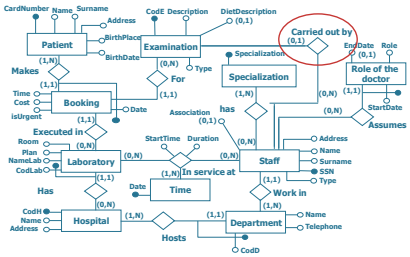
Politecnico di Torino

Translating relationships

Example of relational logic design

44

Binary one-to-many *Carried out by* relationship



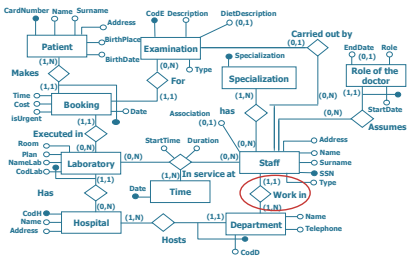
45

Binary one-to-many *Carried out by* relationship

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, *SSN**)
Staff(SSN, Name, Surname, Address, Association*,Type)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent)
DoctorRole(SSN, StartDate, EndDate*, Role)

46

Binary one-to-many *Work in* relationship



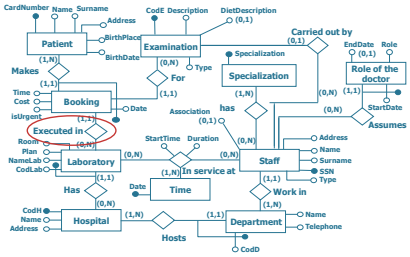
47

Binary one-to-many *Work in* relationship

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, *CodD, CodH*)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent)
DoctorRole(SSN, StartDate, EndDate*, Role)

48

Binary one-to-many *Executed in* relationship



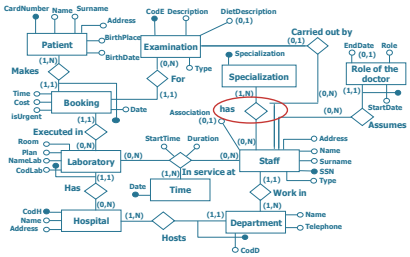
49

Binary one-to-many *Executed in* relationship

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, *CodLab, CodH*)
DoctorRole(SSN, StartDate, EndDate*, Role)

50

Binary many-to-many *has* relationship



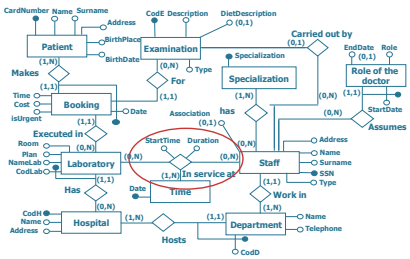
51

Binary many-to-many *has* relationship

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
DoctorRole(SSN, StartDate, EndDate*, Role)
HasSpecialization(SSN, Specialization)

52

Ternary many-to-many *In service at* relationship



53

Ternary many-to-many *In service at* relationship

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)
Time(Date)
Specialization(Specialization)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
DoctorRole(SSN, StartDate, EndDate*, Role)
HasSpecialization(SSN, Specialization)
InServiceAt(SSN, CodLab, CodH, Date, StartTime, Duration)

54

Eliminating redundant tables

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)
Time(Date)
~~*Specialization(Specialization)*~~
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
DoctorRole(SSN, StartDate, EndDate*, Role)
HasSpecialization(SSN, Specialization)
InServiceAt(SSN, CodLab, CodH, Date, StartTime, Duration)

55

Eliminating redundant tables

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)
~~*Time(Date)*~~
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
DoctorRole(SSN, StartDate, EndDate*, Role)
HasSpecialization(SSN, Specialization)
InServiceAt(SSN, CodLab, CodH, Date, StartTime, Duration)

56

Final relational schema

Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)
Hospital(CodH, Name, Address)
Examination(CodE, Description, DietDescription*, Type, SSN*)
Staff(SSN, Name, Surname, Address, Association*, Type, CodD, CodH)
Laboratory(CodLab, CodH, NameLab, Plan, Room)
Department(CodD, CodH, Name, Telephone)
Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
DoctorRole(SSN, StartDate, EndDate*, Role)
HasSpecialization(SSN, Specialization)
InServiceAt(SSN, CodLab, CodH, Date, StartTime, Duration)

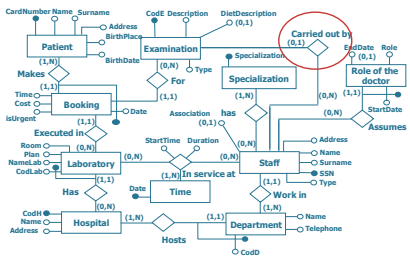
57

Politecnico di Torino

Referential integrity constraints

Example of relational logic design

58



59

Referential integrity: *Carried out by* relationship

- Involved tables

Examination(CodE, Description, DietDescription*, Type, *SSN**)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)

- Referential integrity constraint

Examination(SSN) REFERENCES Staff(SSN)

Referential integrity: *Work in* relationship

- Involved tables

Staff(SSN, Name, Surname, Address, Association*,Type, *CodD, CodH*)
Department(CodD, CodH, Name, Telephone)

- Referential integrity constraint

Staff(CodD,CodH) REFERENCES Department(CodD,CodH)

Referential integrity: *Hosts* relationship

- Involved tables

Department(CodD, *CodH*, Name, Telephone)
Hospital(CodH, Name, Address)

- Referential integrity constraint

Department(CodH) REFERENCES Hospital(CodH)

Referential integrity: *Has* relationship

- Involved tables

Laboratory(CodLab, *CodH*, NameLab, Plan, Room)
Hospital(CodH, Name, Address)

- Referential integrity constraint

Laboratory(CodH) REFERENCES Hospital(CodH)

Referential integrity: *Makes* relationship

- Involved tables

Booking(*CardNumber*, CodE, Date, Time, Cost, isUrgent, CodLab, CodH)
Patient(CardNumber, Name, Surname, Address, BirthPlace, BirthDate)

- Referential integrity constraint

Booking(CardNumber) REFERENCES Patient(CardNumber)

Referential integrity: *For* relationship

- Involved tables

Booking(CardNumber, *CodE*, Date, Time, Cost, isUrgent, CodLab, CodH)
Examination(CodE, Description, DietDescription*, Type, SSN*)

- Referential integrity constraint

Booking(CodE) REFERENCES Examination(CodE)

Referential integrity: *Executed in* relationship

- Involved tables

Booking(CardNumber, CodE, Date, Time, Cost, isUrgent, *CodLab, CodH*)
Laboratory(CodLab, CodH, NameLab, Plan, Room)

- Referential integrity constraint

Booking(CodLab,CodH) REFERENCES Laboratory(CodLab,CodH)

66

Referential integrity: *Assumes* relationship

- Involved tables

DoctorRole(*SSN*, StartDate, EndDate*, Role)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)

- Referential integrity constraint

DoctorRole(SSN) REFERENCES Staff(SSN)

67

Referential integrity: *Has* relationship

- Involved tables

HasSpecialization(*SSN*, Specialization)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)

- Referential integrity constraint

HasSpecialization(SSN) REFERENCES Staff(SSN)

68

Referential integrity: *In service at* relationship

- Involved tables

InServiceAt(*SSN*, CodLab, CodH, Date, StartTime, Duration)
Staff(SSN, Name, Surname, Address, Association*,Type, CodD, CodH)

- Referential integrity constraint

InServiceAt(SSN) REFERENCES Staff(SSN)

69

Referential integrity: *In service at* relationship

- Involved tables

InServiceAt(SSN, *CodLab*, *CodH*, Date, StartTime, Duration)
Laboratory(CodLab, CodH, NameLab, Plan, Room)

- Referential integrity constraint

InServiceAt(CodLab,CodH) REFERENCES Laboratory(CodLab,CodH)

70

Referential integrity constraints

Examination(SSN) REFERENCES Staff(SSN)
Staff(CodD,CodH) REFERENCES Department(CodD,CodH)
Department(CodH) REFERENCES Hospital(CodH)
Laboratory(CodH) REFERENCES Hospital(CodH)
Booking(CardNumber) REFERENCES Patient(CardNumber)
Booking(CodE) REFERENCES Examination(CodE)
Booking(CodLab,CodH) REFERENCES Laboratory(CodLab,CodH)
DoctorRole(SSN) REFERENCES Staff(SSN)
HasSpecialization(SSN) REFERENCES Staff(SSN)
InServiceAt(SSN) REFERENCES Staff(SSN)
InServiceAt(CodLab,CodH) REFERENCES Laboratory(CodLab,CodH)

71

Politecnico di Torino

Normalization

Database design

0

Normalization

- Introduction
- Normal form of Boyce Codd
- Decomposition in normal form
- Properties of decompositions
- Lossless decomposition
- Conservation of dependencies

1

1

Introduction

Normalization

2

2

Normalization

- Normalization is a process which, starting from a non-normalized relational schema, obtains a normalized relational schema
- Normalization is not a design methodology, but a verification tool
- The design methodology based on ER schemas normally produces normalized relational schemas
- Normalization checks can also be applied to ER schemas

3

3

Example

Exam Passed

StudentID	Residence	CourseID	CourseName	Grade
s94539	Milan	04FLYCY	Electronic calculators	30
s94540	Turin	01FLTCY	Database design	26
s94540	Turin	01KPNCY	Computer network	28
s94541	Pescara	01KPNCY	Computer network	29
s94542	Lecce	04FLYCY	Electronic calculators	25

- Constraints
  - The primary key is the pair StudentID, CourseID
  - The place of residence of each student is unique and is an attribute of the student alone, regardless of the exams he or she has passed
  - The name of the course is unique and is a function of the course only, regardless of which students pass the corresponding exam

4

4

Example: Redundancy

Exam Passed

StudentID	Residence	CourseID	CourseName	Grade
s94539	Milan	04FLYCY	Electronic calculators	30
s94540	Turin	01FLTCY	Database design	26
s94540	Turin	01KPNCY	Computer network	28
s94541	Pescara	01KPNCY	Computer network	29
s94542	Lecce	04FLYCY	Electronic calculators	25

- Redundancy
  - In all rows where a student appears, his or her place of residence is repeated
  - In all rows where the same course appears, its name is repeated

5

5

Example: Anomalies

Exam Passed

StudentID	Residence	CourseID	CourseName	Grade
s94539	Milan	04FLYCY	Electronic calculators	30
s94540	Turin	01FLTCY	Database design	26
s94540	Turin	01KPNCY	Computer network	28
s94541	Pescara	01KPNCY	Computer network	29
s94542	Lecce	04FLYCY	Electronic calculators	25

- Update anomaly
  - If a student's place of residence changes, all the rows in which it appears must be modified at the same time
- Insertion anomaly
  - If a new student enrolls at university, he or she cannot be entered in the database until he or she passes the first exam
- Deletion anomaly
  - If a student withdraws from studies, it is not possible to keep track of his place of residence

6

6

Redundancy

- A single relation is used to represent heterogeneous information
  - some data are repeated in different tuples without adding new information
    - redundant data

7

7

Anomalies

- Redundant information must be updated atomically (all at the same time)
- The deletion of a tuple implies the deletion of all concepts represented in it
  - including those that might still be valid
- The insertion of a new tuple is only possible if at least the complete information about the primary key exists
  - it is not possible to insert the part of the tuple relating to only one concept

8

8

Boyce-Codd normal form

Normalization

9

9

Functional dependency

- It is a special type of integrity constraint
- It describes functional links between the attributes of a relation
- Example: the place of residence is unique for each student
  - each time the same student appears, the value is repeated
  - the value of StudentID determines the value of Residence

Exam Passed

StudentID	Residence	CourseID	CourseName	Grade
s94539	Milan	04FLYCY	Electronic calculators	30
s94540	Turin	01FLTCY	Database design	26
s94540	Turin	01KPNCY	Computer network	28
s94541	Pescara	01KPNCY	Computer network	29
s94542	Lecce	04FLYCY	Electronic calculators	25

10

10

Functional dependency

- A relation r satisfies the functional dependency X → Y if, for each pair t1, t2 of tuples of r, having the same values for attributes in X, t1 and t2 also have the same values for attributes in Y
  - X determines Y (in r)
- Examples

StudentID → Residence
CourseID → CourseName
StudentID CourseID → CourseName

11

11

Non-trivial dependency

- The dependency

  StudentID CourseID → CourseID

  is trivial because CourseID is part of both sides
- A functional dependency X → Y is non-trivial if no attribute in X appears among the attributes in Y

12

Functional dependencies and keys

- Given a key K of a relation r

  K → any other attribute of r (or set of attributes)
- Examples
  - StudentID CourseID → Residence
  - StudentID CourseID → CourseName
  - StudentID CourseID → Grade

13

Functional dependencies and anomalies

- Anomalies are caused by attribute properties involved in functional dependencies
  - Examples
    - StudentID → Residence
    - CourseID → CourseName
- Functional dependencies on keys do not give rise to anomalies
  - Example
    - StudentID CourseID → Grade

14

Functional dependencies and anomalies

- The anomalies are caused by
  - the inclusion of mutually independent concepts in the same relation
  - functional dependencies X → Y allowing for multiple tuples with the same value of X
    - X does not contain a key

15

Boyce Codd normal form (BCNF)

- BCNF = Boyce Codd Normal Form
- A relation r is in BCNF if, for every (non-trivial) functional dependency X → Y defined on it, X contains a key of r (X is superkey of r)
- Anomalies and redundancies are not present in BCNF relations because independent concepts are separated in different relations

16

Normal form decomposition

Normalization

17

BCNF decomposition

- Normalization
  - process of replacing a non-normalised relation by two or more relations in BCNF
- Criteria
  - a relation representing several independent concepts is decomposed into smaller relations, one for each concept, by means of functional dependencies
- The new relations are obtained by projections onto the sets of attributes corresponding to the functional dependencies
- The keys of the new relations are the left parts of the functional dependencies
  - the new relations are in BCNF

18

Example

- Functional dependencies in the example
  - StudentID → Residence
  - CourseID → CourseName
  - StudentID CourseID → Grade

Exam Passed

StudentID	Residence	CourseID	CourseName	Grade
s94539	Milan	04FLYCY	Electronic calculators	30
s94540	Turin	01FLTCY	Database design	26
s94540	Turin	01KPNCY	Computer network	28
s94541	Pescara	01KPNCY	Computer network	29
s94542	Lecce	04FLYCY	Electronic calculators	25

19

Example

- By
  R (StudentID, Residence, CourseID CourseName, Grade)
- Functional dependencies in the example
  - StudentID → Residence
  - CourseID → CourseName
  - StudentID CourseID → Grade
- The relations in BCNF are
  R1 (StudentID, Residence) = $\pi_{\text{StudentID, Residence}}$R
  R2 (CourseID, CourseName) = $\pi_{\text{CourseID, CourseName}}$ R
  R3 (StudentID, CourseID, Grade) = $\pi_{\text{StudentID, CourseID, Grade}}$ R

20

Example

R_1

StudentID	Residence
s94539	Milan
s94540	Turin
s94540	Turin
s94541	Pescara

R_2

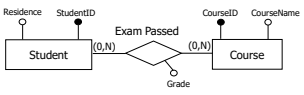
CourseID	CourseName
04FLYCY	Electronic calculators
01FLTCY	Database design
01KPNCY	Computer network

R_3

StudentID	CourseID	Grade
s94539	04FLYCY	30
s94540	01FLTCY	26
s94540	01KPNCY	28
s94541	01KPNCY	29
s94542	04FLYCY	25

21

Example: corresponding ER scheme



Student (StudentID, Residence)
Course (CourseID, CourseName)
Exam Passed (StudentID, CourseID, Grade)

22

Decomposition properties

Normalization

23

Decomposition properties

- Are all decompositions acceptable?
  - essential properties for "good" decomposition
- Problems
  - information loss
  - loss of dependencies

Example

Employee	Category	Salary
Rossi	2	1800
Verdi	3	1800
Bianchi	4	2500
Neri	5	2500
Bruni	6	3500

R (Employee, Category, Salary)

Employee → Category
Employee → Salary
Category → Salary

Lossless Decomposition

Normalization

Example: decomposition (n.1)

R (Employee, Category, Salary)

- Decomposition based on functional dependencies

Employee→ Salary
Category→ Salary

Example: decomposition (n.1)

R (Employee, Category, Salary)

- Decomposing

R_1 (Employee, Salary) = $\pi_{Employee, Salary}$ R

Employee	Salary
Rossi	1800
Verdi	1800
Bianchi	2500
Neri	2500
Bruni	3500

R_2 (Category, Salary) = $\pi_{Category, Salary}$ R

Category	Salary
2	1800
3	1800
4	2500
5	2500
6	3500

Example: recomposition (n.1)

- Recomposing

$R_1 \bowtie R_2$

Employee	Category	Salary	
Rossi	2	1800	
Rossi	*3*	*1800*	← "spurious" tuples
Verdi	*2*	*1800*	← "spurious" tuples
Verdi	3	1800	
Bianchi	4	2500	
...	...	...	

- Reconstruction with loss of information

Decomposition without loss

- The decomposition of a relation r into two sets of attributes X1 and X2 is lossless if the join of the projections of r into X1 and X2 is equal to r itself (no "spurious" tuples)
- A decomposition performed to normalize a relation must be lossless

30

Lossless decomposition

- Given the relation r(X) and sets of attributes X1 and X2 such that
  $X = X_1 \cup X_2$
  $X_0 = X_1 \cap X_2$
  if r satisfies the functional dependency
  $X_0 \rightarrow X_1$ or $X_0 \rightarrow X_2$
  the decomposition of r on X1 and X2 is lossless
- Common attributes form a key to at least one of the decomposed relations

31

Example: loss of information

R_1 (Employee, Salary) R_2 (Category, Salary)

- Verification of condition for lossless decomposition

  X_1 = Employee, Salary
  X_2 = Category, Salary
  X_0 = Salary

- The attribute Salary does not satisfy the condition for lossless decomposition

32

Example: decomposition (n.2)

R (Employee, Category, Salary)

- Decomposition based on functional dependencies

  Employee→ Category
  Employee→ Salary

- Decomposing

  R_1 (Employee, Category) = $\pi_{\text{Employee, Salary}}$ R

Employee	Category
Rossi	2
Verdi	3
Bianchi	4
Neri	4
Bruni	5

  R_2 (Employee, Salary) = $\pi_{\text{nCategory, Salary}}$ R

Employee	Salary
Rossi	1800
Verdi	1800
Bianchi	2500
Neri	2500
Bruni	3500

33

Example: lossless decomposition?

R_1 (Employee, Category) R_2 (Employee, Salary)

$R_1 \bowtie R_2$

- Is the decomposition lossless?
- Verifying the condition for lossless decomposition

  X_1 = Employee, Category
  X_2 = Employee, Salary
  X_0 = Employee

- The attribute Employee satisfies the condition for lossless decomposition

34

Conservation of dependencies

Normalization

35

Example: inserting a new tuple

R_1 (Employee, Category) R_2 (Employee, Salary)

- Inserting the tuple
  - Employee: Gialli – Category: 3 – Salary: 3500

Employee	Category
Rossi	2
Verdi	3
Bianchi	4
Neri	4
Bruni	5
Gialli	3

Employee	Salary
Rossi	1800
Verdi	1800
Bianchi	2500
Neri	2500
Bruni	3500
Gialli	3500

36

Example: inserting a new tuple

- What happens if we insert the tuple (Gialli, 3500) in R_2?
  - in the original relation insertion is forbidden because it violates the dependency Category → Salary
  - in the decomposition it is no longer possible to detect the violation, since the attributes Category and Salary are in separate relations
- The dependency between Category and Salary has been lost

37

Conservation of dependencies

- A decomposition preserves dependencies if each of the functional dependencies of the original schema is present in one of the decomposed relations
- Dependencies should be retained to ensure that the same constraints are satisfied in the decomposed schema as in the original schema

38

Example: decomposition (n.3)

R (Employee, Category, Salary)

- Decomposition based on functional dependencies

  Employee → Category
  Category → Salary

- Decomposing

  R_1 (Employee, Category) = $\pi_{\text{Employee, Category}}$ R

Employee	Category
Rossi	2
Verdi	3
Bianchi	4
Neri	4
Bruni	5

  R_2 (Category, Salary) = $\pi_{\text{Category, Salary}}$ R

Category	Salary
2	1800
3	1800
4	2500
5	2500
6	3500

39

Example: Lossless decomposition

- Recomposing

  $R_1 \bowtie R_2$

- Condition check for lossless decomposition

  X_1 = Employee, Category
  X_2 = Category, Salary
  X_0 = Category

- The attribute Category satisfies the condition for lossless decomposition

40

Example: Conservation of functional dependencies

- Recomposing

  $R_1 \bowtie R_2$

- Conserved functional dependencies

  Employee → Category
  Category → Salary

- Functional dependency

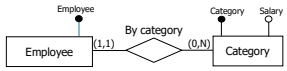
  Employee → Salary

  can be reconstructed from

  Employee → Category
  Category → Salary

41

Example: corresponding ER scheme



Employee (<u>Employee</u>, Category)
Category (<u>Category</u>, Salary)

42

Quality of a decomposition

- Decompositions must always satisfy the properties
  - lossless decomposition
    - ensures that the information in the original relation is accurately reconstructed (without spurious tuples) from the information in the decomposed relations
  - conservation of dependencies
    - ensures that the decomposed relations have the same capacity as the original relation to represent the integrity constraints

43