DIUM, University of Udine

Basi di dati
Logical design

Andrea Brunello
andrea.brunello@uniud.it

Database design

The process of designing a database is typically articulated into different phases:

1. Brainstorming meetings with IT personnel and all interested stakeholders
   - Collection of requirements
   - Design of a conceptual model (e.g., E-R model)
   - The E-R model has a specific notation, the *E-R diagram*, that helps all involved parts to discuss about the future database
2. Translation of the conceptual model into a logical model
   - Typically, a set of translation rules is followed
   - At this stage, a specific DBMS technology must be chosen (e.g., relational DB)
3. Addition to the logical model of details regarding the physical, low-level aspects (e.g., usage of indexes)
   - The physical schema is thus obtained

Some context

- We have just seen how to develop the conceptual schema of a database
- We have also seen the relational model as our logical model of choice
- In this set of slides, we will discuss the translation from the conceptual to the logical model

Logical design steps

There are two main steps involved in the relational logical design, when starting from a conceptual schema:

1. Entity-Relationship schema restructuring
2. Translation of the schema into a logical schema

Entity-Relationship schema restructuring

- It involves a simplification of the original E-R schema, to remove constructs that are not supported by the logical model
- As we shall see, the simplification operations consider also the intended use of the database, i.e., the operations that will be performed on the data
- The result is an E-R schema that takes into account some implementation aspects; thus, it is not, strictly speaking, a conceptual schema

The restructuring operations are as follows:

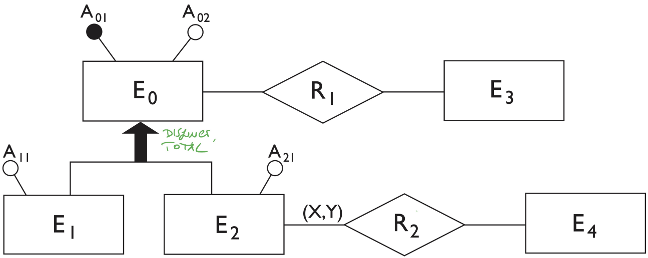
- Analysis of the redundancies
- Removal of generalizations
- Removal of multi-valued attributes
- Removal of composite attributes
- Choice of the main identifiers

- A redundancy in a conceptual schema corresponds to the presence of information that can be derived from others that are present in the schema
- This is the case with **derived attributes**: should they be kept or not?
  - **Reasons to keep:** a lot of read operations are performed over them; they are the result of computationally complex operations
  - **Reasons to remove:** they waste memory; they have to be kept up-to-date
- As we shall see, it is not necessary to encode the derived attributes as table attributes in the relational logical model; *views* can be used instead

Removing generalizations

- The relational model does not allow the direct representation of generalizations of the E-R model.
- We need, therefore, to transform these constructs into other constructs that are easier to translate: entities and relationships.

Example of a schema with generalization



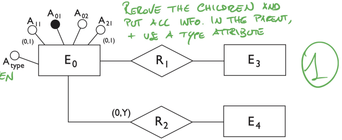
Possible restructurings of the previous schema

IF THE SPECIALIZATION IS TOTAL I HAVE SEVERAL OPTIONS. THE FINAL CHOICE FOR THE RESTRUCTURING DEPENDS ON

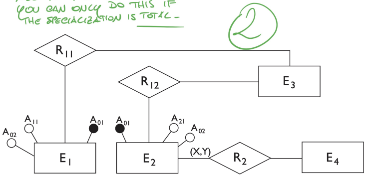
THE QUERIES THAT ARE EXPECTED TO BE PERFORMED ON THE DB

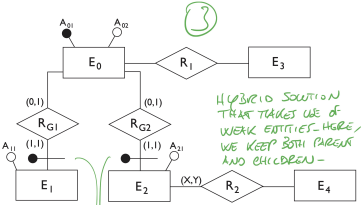
THE # OF ATTRIBUTES AND RELATIONSHIPS OF THE PARENT AND THE CHILDREN

REMOVE THE CHILDREN AND PUT ALL INFO. IN THE PARENT, + USE A TYPE ATTRIBUTE



REMOVE PARENT AND MOVE ALL TO THE CHILDREN. YOU CAN ONLY DO THIS IF THE SPECIALIZATION IS TOTAL.





HYBRID SOLUTION THAT TAKES USE OF WEAK ENTITIES. HERE, WE KEEP BOTH PARENT AND CHILDREN.

OBSERVE THAT THESE TWO WEAK ENTITIES HAVE NO PARTIAL KEY ATTRIBUTES

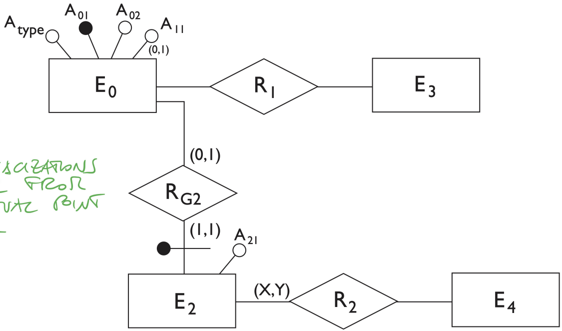
General rules about generalization removal

- Option 1 is useful when the operations involve the occurrences and the attributes of E_0, E_1 and E_2 more or less in the same way.
- Option 2 is possible only if the generalization is total and is useful when there are operations that refer only to occurrences of E_1 or of E_2, and so they make distinctions between these entities.
- Option 3 is useful when the generalization is not total and the operations refer to either occurrences and attributes of E_1 (E_2) or of E_0, and therefore make distinctions between child and parent entities.
- The various options can be combined.

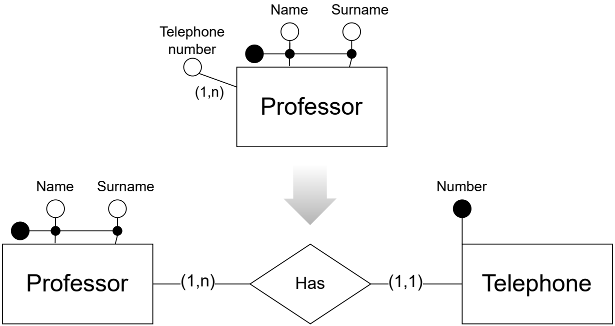
Database Systems
Chapter 7: Logical design

Possible restructuring of the previous schema

STILL, SPECIALIZATIONS ARE USEFUL FROM A CONCEPTUAL POINT OF VIEW.



- Multi-valued attributes are not supported by the relational logical model (first normal form)
- They are converted in a rather simple way



- Composite attributes cannot be represented in the relational model (first normal form property)
- Thus, we have two choices:
  - Merge all composite attributes together into a single one
  - Keep only the composing fields, modeling them as distinct attributes

- In the E-R model, an entity set may have more than one identifier
- Before performing the translation into the relational logical model, it is necessary to select one of them to act as the primary key
- Typically, the smallest one in terms of composing attributes, for convenience reasons involving also foreign key constraints and querying of the database
- This is not a strict rule, and it mostly depends on the modeled domain

- Let us now restructure the Entity-Relationship schema of Exercise 2, University DB
- Then, you will restructure the Entity-Relationship schema of the Airport exercise!

Translation into a logical schema

- Now we have an E-R diagram
  - devoid of generalizations and multi-valued attributes, and
  - with a single identifier per entity set

- We can proceed with its translation into an *equivalent* relational logical schema

- The translation follows a set of pre-defined rules, that we will describe starting from the simplest case, i.e., that of entity sets

Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set. (TYPICALLY, AS WE SHALL SEE IT IS NOT ALWAYS THE CASE)
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

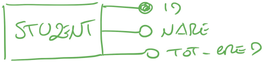
⟹ REMEMBER THAT IN THE RELATIONAL MODEL WE JUST HAVE THE FUNDAMENTAL NOTION OF RELATION (TABLE)

Representing Entity Sets

AS AN EXCEPTION TO THIS RULE, WE WILL SEE HOW TO DEAL WITH ONE-TO-ONE RELATIONSHIPS

- A strong entity set reduces to a schema with the same attributes

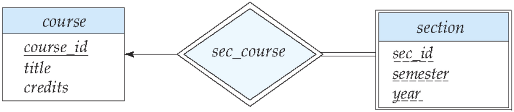
  student(<u>ID</u>, name, tot_cred)

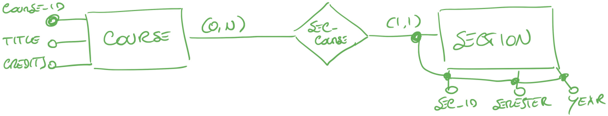


+: NOT NULL CONSTRAINTS OVER NON-OPTIONAL ATTRIBUTES

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

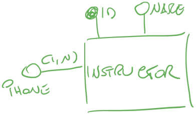
  section (<u>course_id, sec_id, sem, year</u>)





Representation of Entity Sets with Multivalued Attributes

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
- Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)



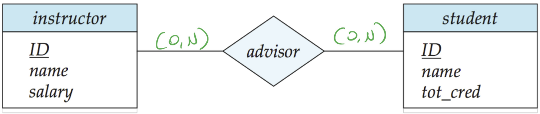
INSTRUCTOR (ID, NAME)

INSTRUCTOR-PHONE (INSTRUCTOR-ID, PHONE)

Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

advisor = (*s_id, i_id*)

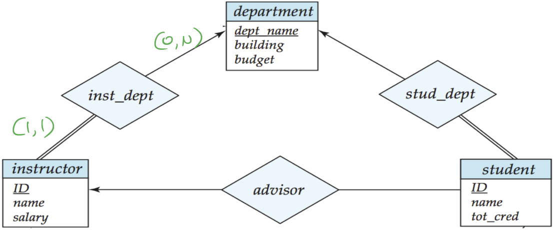


IN THE MANY-TO-MANY CASE, YOU ALWAYS ADD A NEW RELATION CORRESPONDING TO THE RELATIONSHIP SET, IRRESPECTIVE FROM THE PARTICIPATION CONSTRAINTS

CREATING A SINGLE, UNIVERSAL TABLE WITH ALL ATTRIBUTES OF INSTRUCTOR AND STUDENT IS NOT A VIABLE OPTION, DUE TO REDUNDANCY ISSUES

Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*



INSTRUCTOR (ID, NAME, SALARY, DEPT_NAME)

IF PARTICIPATION OF INSTRUCTOR IS TOTAL, THEN ADD A NOT-NULL CONSTRAINT HERE

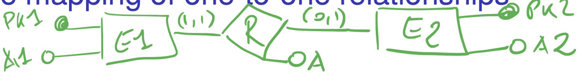
A note on the ER-to-relational mapping: the case of one-to-one relationships

Angelo Montanari
Dept. of Mathematics, Computer Science, and Physics
University of Udine, Italy

Course on Data Management for Big Data

Translation into a logical schema

The mapping of one-to-one relationships



We have to distinguish among 3 different cases:

- $E1(\mathbf{PK1}, A1) - (1,1) - R(A) - (0,1) - E2(\mathbf{PK2}, A2)$

  where $PK1$ is the key of entity $E1$, $A1$ is an attribute of $E1$, A is an attribute of relationship R, $PK2$ is the key of entity $E2$, and $A2$ is an attribute of $E2$

  (the case

  $E1(\mathbf{PK1}, A1) - (0,1) - R(A) - (1,1) - E2(\mathbf{PK2}, A2)$

  is completely symmetric, and thus ignored)

- $E1(\mathbf{PK1}, A1) - (0,1) - R(A) - (0,1) - E2(\mathbf{PK2}, A2)$

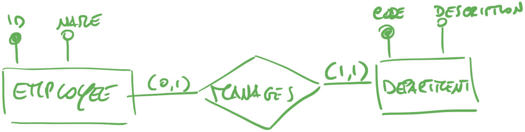
- $E1(\mathbf{PK1}, A1) - (1,1) - R(A) - (1,1) - E2(\mathbf{PK2}, A2)$

The case of (1, 1) – (0, 1) relationships (AND VICE-VERSA)

How can we map the following ER schema into a corresponding relational one (introducing no redundancy, and preserving as much as possible information/constraints)?

$E1(\mathbf{PK1}, A1) - (1, 1) - R(A) - (0, 1) - E2(\mathbf{PK2}, A2)$

E.G.,



The case of $(1, 1) - (0, 1)$ relationships

How can we map the following ER schema into a corresponding relational one (introducing no redundancy, and preserving as much as possible information/constraints)?

$E1(\mathbf{PK1}, A1) - (1, 1) - R(A) - (0, 1) - E2(\mathbf{PK2}, A2)$

Relational schema:

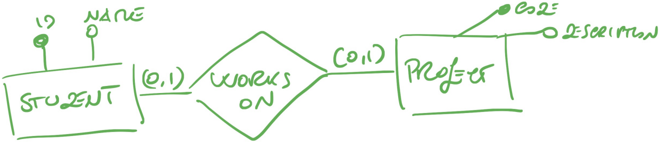
$E1(\underline{PK1}, A1, PK2, A)$ and $E2(\underline{PK2}, A2)$, where $PK2$ is a foreign key of relation $E1$ that refers to the primary key $PK2$ of relation $E2$.

- $(1, \_)$ on $E1$ side: $PK2$ NOT NULL in $E1$
- $(\_, 1)$ on $E1$ side: $PK1$ is the primary key
- $(0, \_)$ on $E2$ side: $PK2$ foreign key in $E1$ referring to the primary key of $E2$
- $(\_, 1)$ on $E2$ side: $PK2$ UNIQUE in $E1$

The case of $(0, 1) - (0, 1)$ relationships

$$E1(\mathbf{PK1}, A1) - (0, 1) - R(A) - (0, 1) - E2(\mathbf{PK2}, A2)$$

E.G.,



The case of $(0,1) - (0,1)$ relationships

$E1(\mathbf{PK1}, A1) - (0,1) - R(A) - (0,1) - E2(\mathbf{PK2}, A2)$

Relational schema:

$E1(\underline{PK1}, A1, PK2, A)$ and $E2(\underline{PK2}, A2)$, where $PK2$ is a foreign key of relation $E1$ that refers to the primary key $PK2$ of relation $E2$.

- $(0,\_)$ on $E1$ side: $PK2$ **can be NULL in** $E1$
- $(\_,1)$ on $E1$ side: $PK1$ is the primary key
- $(0,\_)$ on $E2$ side: $PK2$ foreign key in $E1$ referring to the primary key of $E2$
- $(\_,1)$ on $E2$ side: $PK2$ UNIQUE in $E1$

$E1(\underline{PK1}, A1)$ and $E2(\underline{PK2}, A2, PK1, A)$, where $PK1$ is a foreign key of relation $E2$ that refers to the primary key $PK1$ of relation $E1$, works as well.

SO, BASICALLY IT IS THE SAME AS THE PREVIOUS SOLUTION, BUT WE DROP THE NOT-NULL CONSTRAINT

The case of $(0, 1)$ – $(0, 1)$ relationships (contn'd)

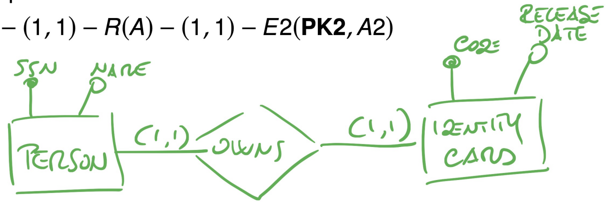
How do we choose between the two options? Participation of entities in the relationship can be a criterion.

EXTEND THE RELATION THAT CORRESPONDS TO THE ENTITY THAT "PARTICIPATES MOST" TO THE RELATIONSHIP, SO TO AVOID WASTING TOO MANY NULL VALUES.

The case of $(1,1)-(1,1)$ relationships

No one of the previous solutions works with
$E1(\mathbf{PK1}, A1) - (1,1) - R(A) - (1,1) - E2(\mathbf{PK2}, A2)$

E.G.,



The case of $(1,1)$ – $(1,1)$ relationships

No one of the previous solutions works with

$E1(\mathbf{PK1}, A1)$ – $(1,1)$ – $R(A)$ – $(1,1)$ – $E2(\mathbf{PK2}, A2)$

Relational schema:

$R(\underline{PK1}, A1, PK2, A2, A)$ (or $R(PK1, A1, \underline{PK2}, A2, A)$)

→ SIMPLY PUT ALL IN A SINGLE TABLE

→ EXCEPTION TO THE RULE "EACH ENTITY SET MAPS TO ITS OWN TABLE"

The case of $(1,1) - (1,1)$ relationships

No one of the previous solutions works with

$E1(\mathbf{PK1}, A1) - (1,1) - R(A) - (1,1) - E2(\mathbf{PK2}, A2)$

Relational schema:

$R(\underline{PK1}, A1, PK2, A2, A)$ (or $R(PK1, A1, \underline{PK2}, A2, A)$)

- $(1, \_)$ on $E1$ side: $PK2$ NOT NULL in R
- $(\_, 1)$ on $E1$ side: $PK1$ is the primary key (UNIQUE)
- $(1, \_)$ on $E2$ side: $PK1$ is the primary key (NOT NULL)
- $(\_, 1)$ on $E2$ side: $PK2$ UNIQUE in R

- Let us now turn our restructured Entity-Relationship schema of Exercise 2, University DB, into a relational logical model
- Then, you will translate the restructured Entity-Relationship schema of the Airport exercise!