View Integration

- View integration is the process of merging several conceptual schemas into a global conceptual schema that represents all the requirements of the application.
- The main goal of view integration is to find all parts of the input conceptual schemas

that refer to the same portion of reality and to unify their representation.

This activity is called schema integration; it is very complex, since the same portion of reality is usually modelled in different ways in each schema.

Integration is also required in another context, called **database integration**, which involves merging several different databases into a single database; in this case, we must first construct a conceptual schema of each individual database and then integrate those schemas. This activity is required for large information systems consisting of several databases;

A special application of database integratron occurs its distributed database systems, in which individual databases are stored on different computers in a computer network. Users of these systems should be provided with a unified view of the database that is transparent to data distribution and allocation. Besides encountering technological problems, designers of these systems may have to integrate existing databases,

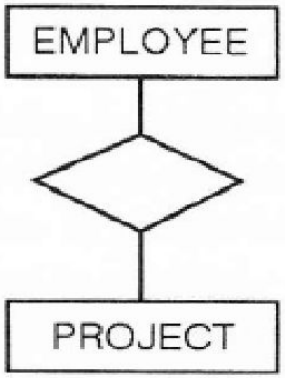
- First of all we examine problems and issues that influence the integration activity.
- In the next Section we deal with integration in the large, that is, with how to organize the integration of several schemas.
- Subsequent sections deal with integration in the smalL, that is, between two input schemas.
- We deal with conflict analysis and resolution, and with the merging of views.

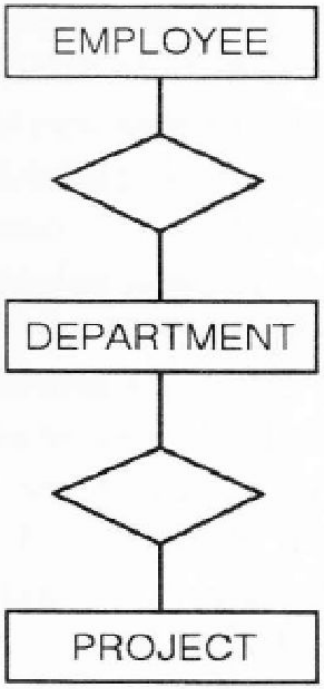
Issues in View Integration

- The main difficulty of view integration is discovering the differences in the schemas to be merged. Differences in modeling are due to the following causes:
- Different Perspectives.
- Equivalence among Constructs in the Model.
- Incompatibe Design Specifications

Different Perspectives

Different Perspectives. In the design process, designers model the same objects from their own point of view. Concepts may be seen at different levels of abstraction, or represented using different properties. An example is given in Figure 5.1a: the relationship between EMPLOYEE and PROJECT is perceived as one relationship in one schema and as the combination of two relationships in the other schema.

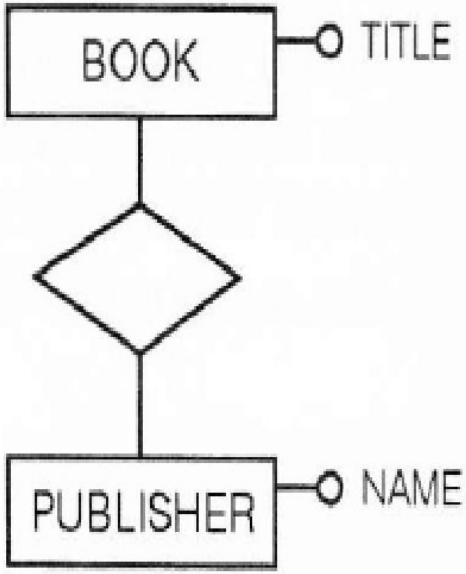


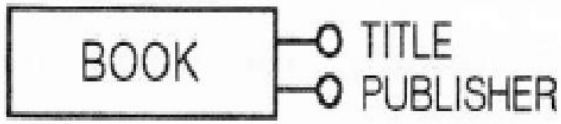


5.1.a. Different perspectives

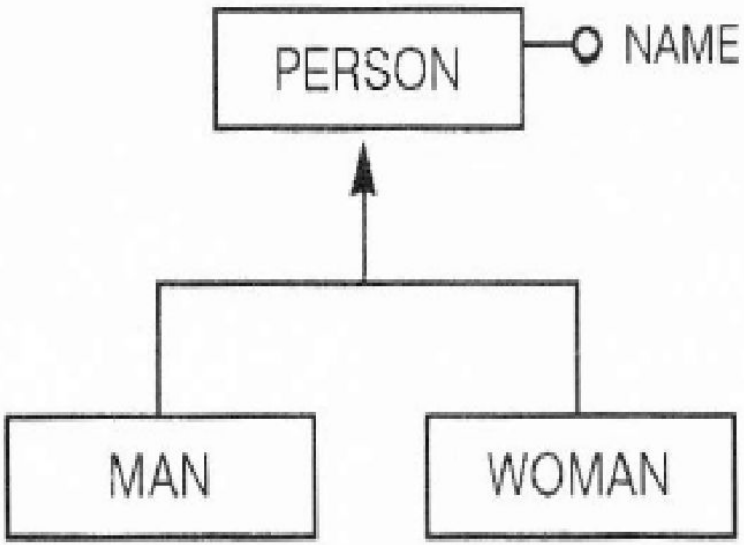
Equivalence among Constructs in the Model.

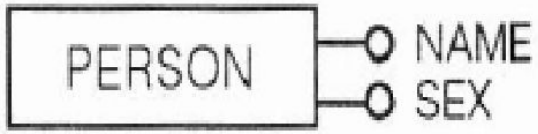
Equivalence among Constructs in the Model. Conceptual models have a rich variety of representation structures; therefore, they allow for different equivalent representations of the same reality. For instance, in Figure 5.1b the association between book and publisher is represented by a relationship between the entities BOOK and PUBLISHER in one schema and as an attribute of the BOOK entity in the other schema. Similarly, in Figure 5.1c the partitioning of persons into males and females is represented by a generalization hierarchy among the entities PERSON, MAN, and WOMAN in one schema and by the attribute SEX of the entity PERSON in the other schema.





(b) Equivalent constructs



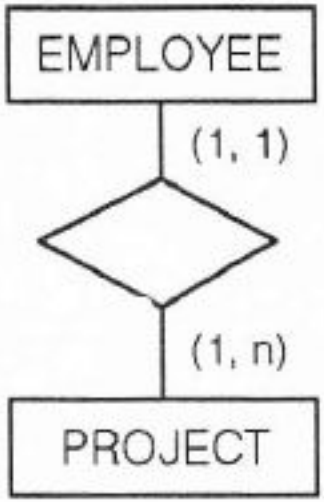


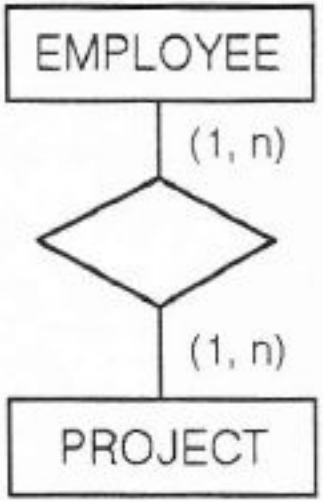
(c) Equivalent constructs

Incompatibe Design Specifications

- Errors during view design regarding names,
- structures, and integrity,constraits may produce erroneous inputs for the integration activity. During integration, these errors should be selected, and corrected. For example, in Figure 1d the first schema indicates that each employee is always assigned to a unique project, but the second schema indicates that each employee works in multiplle projects.

Both schemas look correct, one of them is wrong.



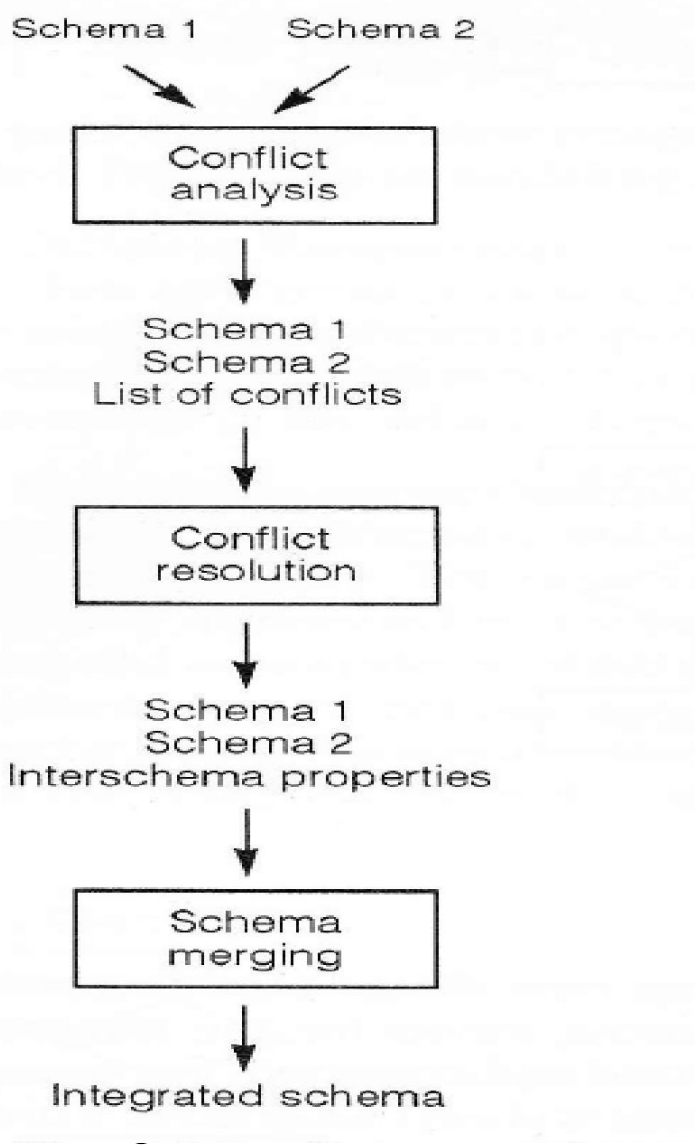


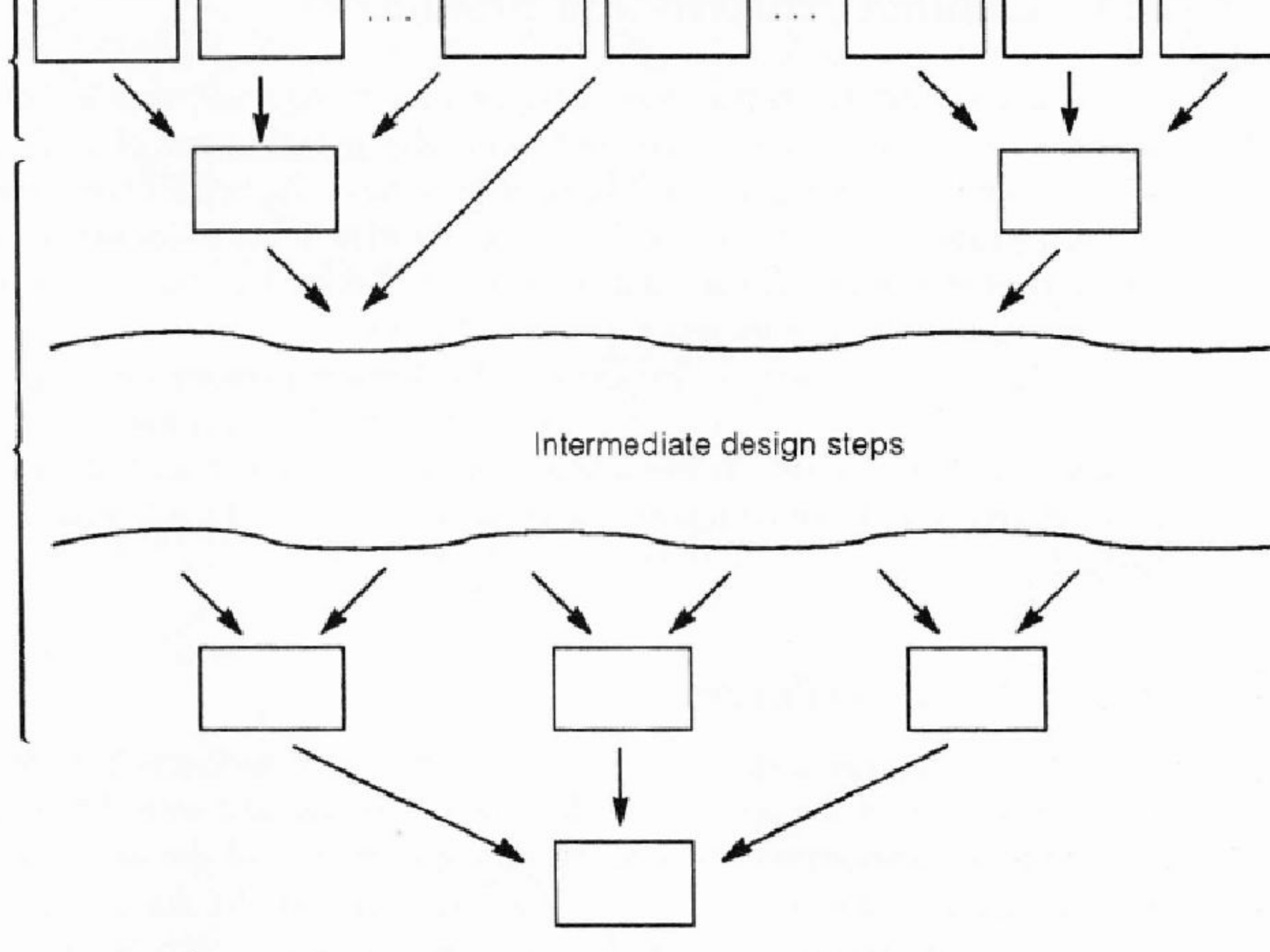
(d) Incorrect specification

View Integration in Large

In large database design projects, it is quite common to produce tens or even hundreds of different schemas that must be integrated. This process requires establishing a discipline for selecting an appropriate sequence of individual integrations of views. The most general approach to the integration process is shown in Figure 5.3; this approach proceeds with the integration of several schemas at a time, and therefore involves several coexisting, partially integrated schemas.

5.2 Approach to vie




Intermediate design steps

The integration of many schemas at the same time is not very convenient, because it is quite difficult to discover conflicts. We suggest instead that only one pair of schemas be considered at a time; further, we suggest that the results of schema integration be accumulated into a single schema, which evolves gradually towards the global conceptual schema. This approach to view integration is shown in Figure 5.4: one schema is progressively enlarged to include new concepts from other schemas.

We must first choose the order of schema comparisons. If the integration process is performed according to a mixed (top-down and bottom-up) design strategy, as described in Section 3.2.4, then the *skeleton schema* should be chosen as the input to the first integration process. Schemas should be progressively aggregated with the skeleton schema, which performs a pivotal function. The order in which the other schemas are considered is not particularly relevant. If no skeleton schema is available, the designer should establish a priority among schemas, based on their importance, completeness, and reliability.

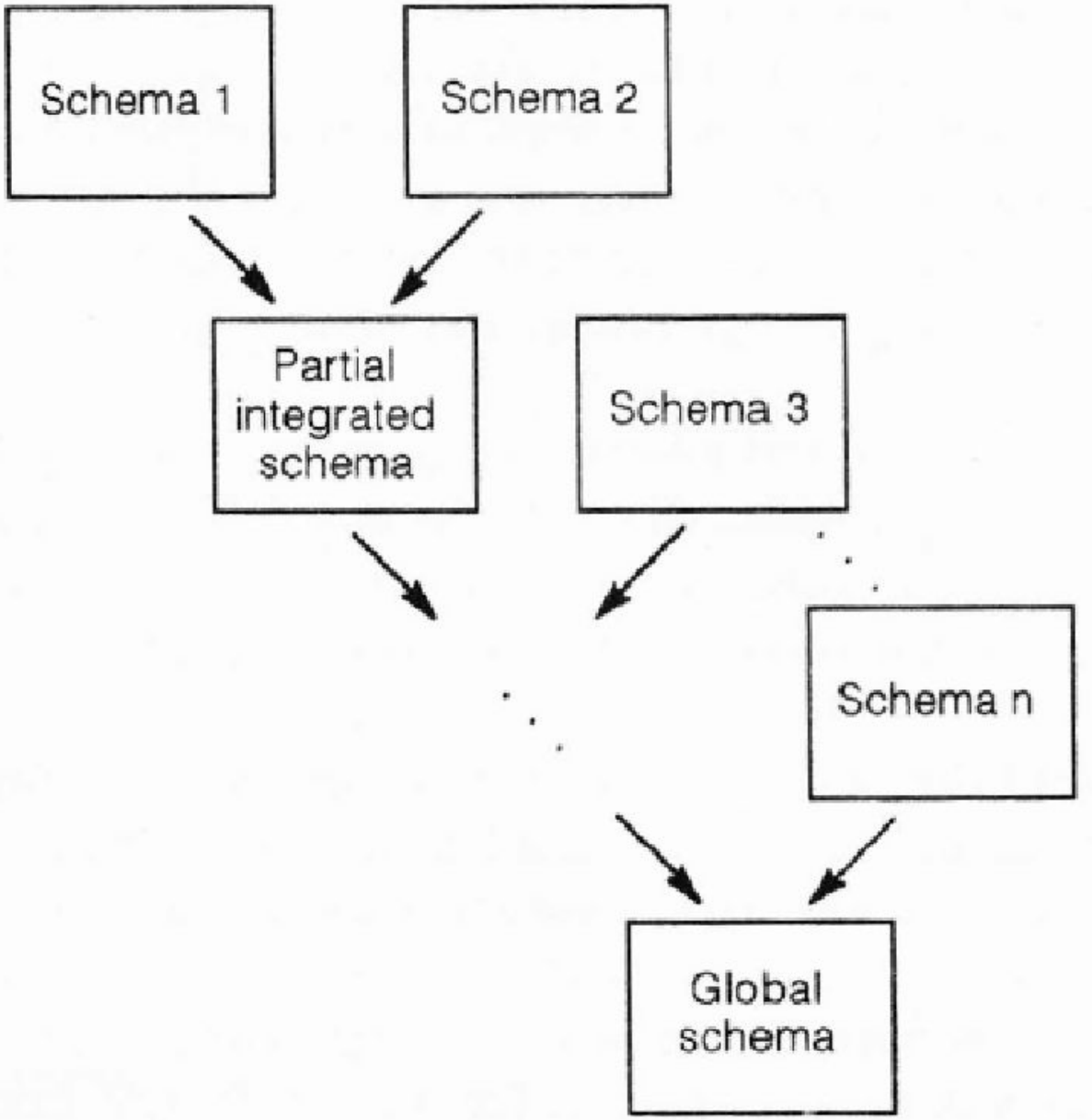


Figure 5.4 The suggested sequence of view-integration activities

Conflict Analysis and Resolution

We now concentrate on the integration process *in the small*, that is, between a pair of schemas, using an example that deals with the management of a library. The input schemas are shown in Figure 5.5. The scientist schema (Schema 1) describes the structure of the private library of a researcher. The librarian schema (Schema 2) describes the structure of the central library of a department. Tables 5.1 and 5.2 describe concepts of Schemas 1 and 2 whose meanings are not obvious.

Conflict analysis aims at detecting all the differences in representing the same reality in the two schemas. Two main tasks may be distinguished: (1) **name conflict analysis,** in which names of concepts in the schemas are compared and unified; and (2) **structural conflict analysis,** in which representations of concepts in the schemas are compared and unified.

Name Conflict Analysis

- There are two sources of name conflicts: synonyms and homonyms. **Synonyms** occur when the same objects of the application domain are represented with different names in the two schemas; **homonyms** occur when different objects of the application domain are represented with the same name in the two schemas. In discovering synonyms and homonyms, the designer is guided by a similarity or a mismatch among concepts, which may suggest the presence of a naming conflict. **Concept similarity** arises when concepts with different names have several common properties and constraints in the schemas. Similarity of two concepts indicates that they may be synonyms. **Concept mismatch** arises when concepts with the same name have different properties and constraints in the schemas. Mismatch of two concepts indicates that they may be homonyms.

The terms *properties* and *constraints* in these definitions are defined as follows. **Properties** of a concept are all other *neighbor concepts* in the schema. For instance, the properties of a given entity are all its attributes, as well as the relationships, subsets, and generalizations in which it participates. **Constraints** are rules or conditions that limit the set of valid instances of the schema. These include, for instance, cardinality constraints for relationships and generalizations. Table 5.3 shows neighbor properties and constraints for entities, relationships, and attributes.

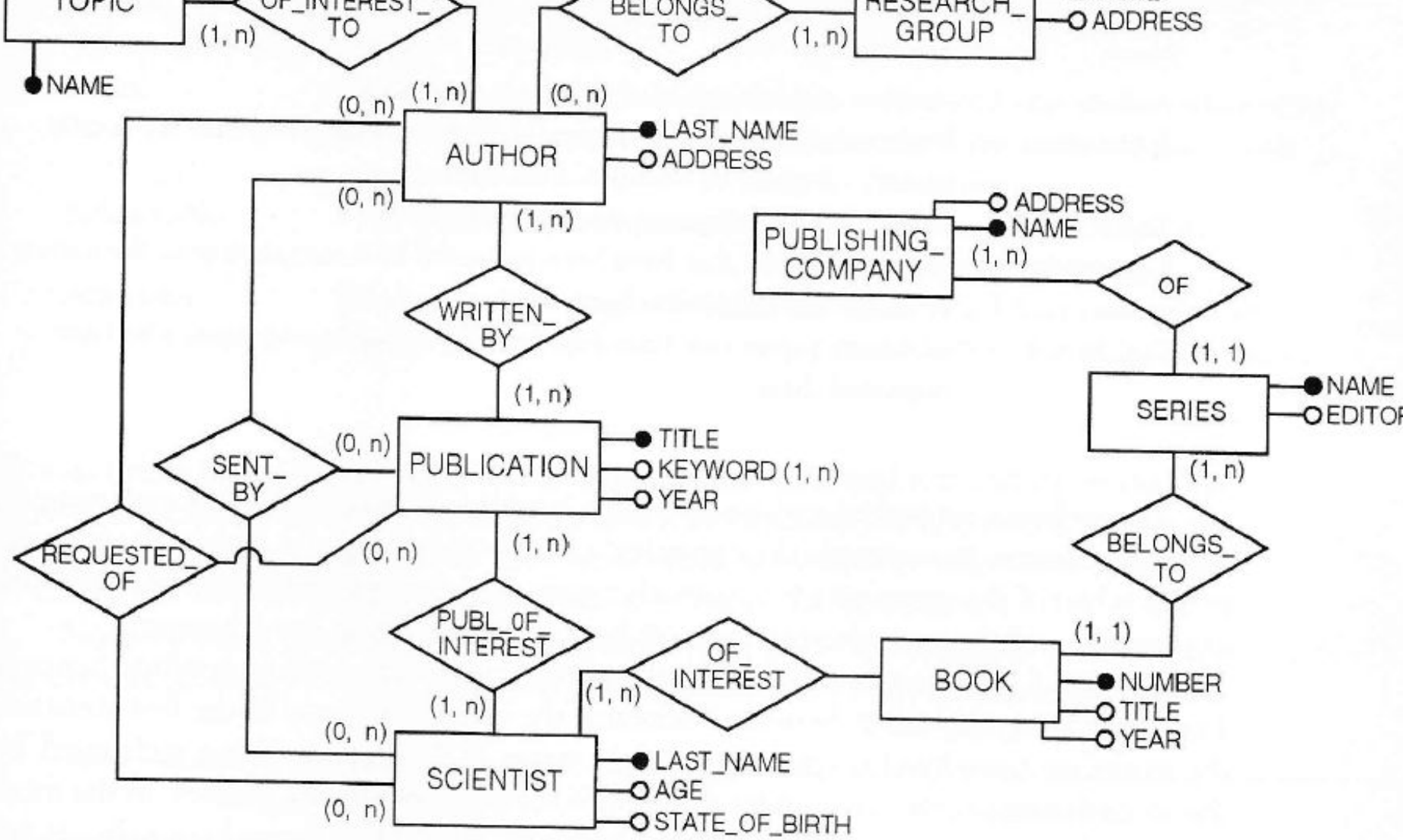
As a consequence of detecting similar and mismatched concepts, several possible modifications can be performed on the schemas. We call the set of all such possible modifications *modification scenarios*. The modification scenarios for the naming conflict involve renaming the concept; they can also involve adding some *interschema property*. Let us explain both of these ideas.

Concept Renaming

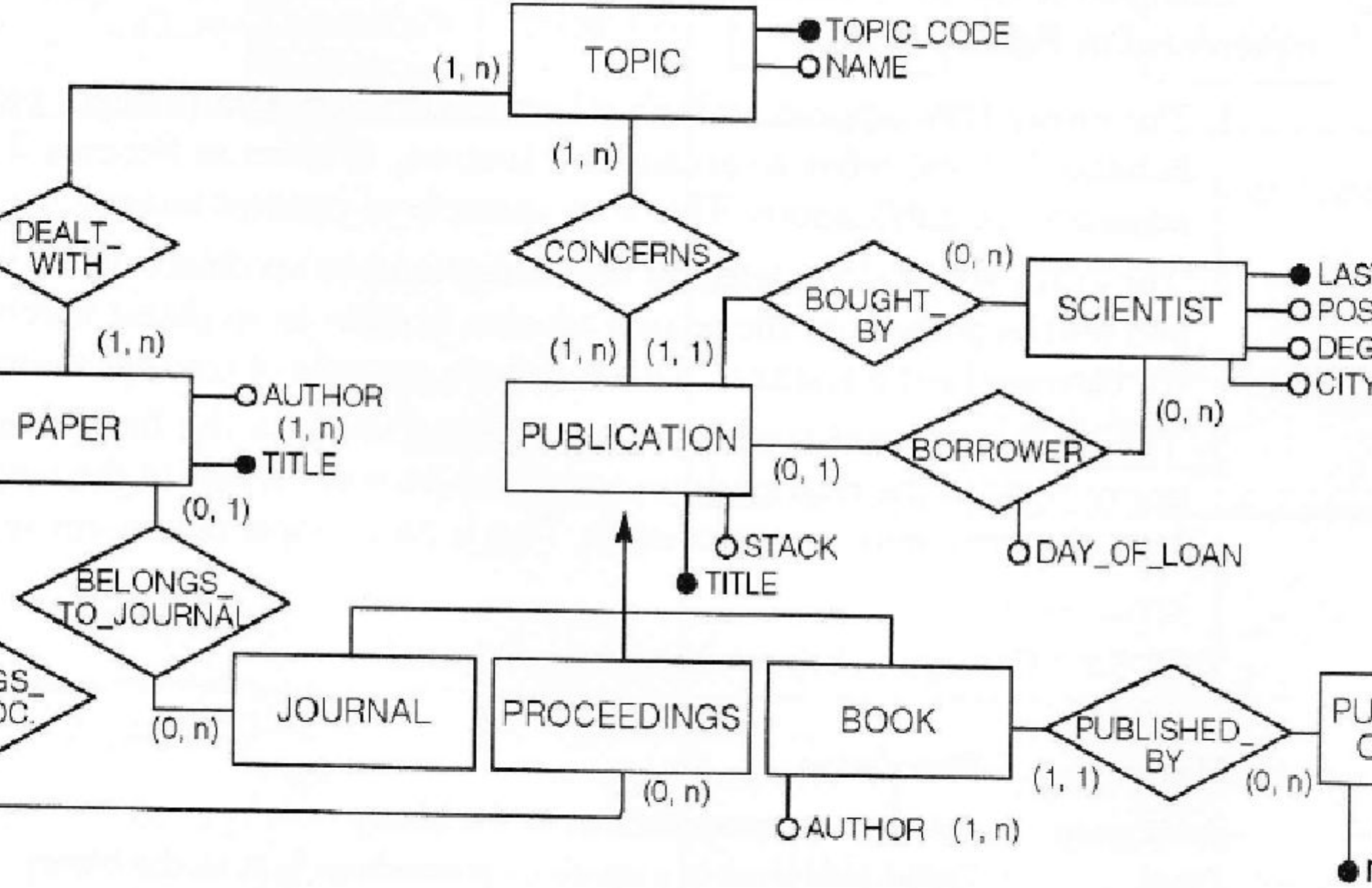
The terms *properties* and *constraints* in these definitions are defined as follows. **Properties** of a concept are all other *neighbor concepts* in the schema. For instance, the properties of a given entity are all its attributes, as well as the relationships, subsets, and generalizations in which it participates. **Constraints** are rules or conditions that limit the set of valid instances of the schema. These include, for instance, cardinality constraints for relationships and generalizations. Table 5.3 shows neighbor properties and constraints for entities, relationships, and attributes.

As a consequence of detecting similar and mismatched concepts, several possible modifications can be performed on the schemas. We call the set of all such possible modifications *modification scenarios*. The modification scenarios for the naming conflict involve renaming the concept; they can also involve adding some *interschema property*. Let us explain both of these ideas.

Concept renaming is performed (i.e., a concept is renamed) whenever we detect a synonym or a homonym. Synonyms should be eliminated to remove ambiguity. For example, when two concepts such as CUSTOMER and CLIENT are synonyms, then we select one of them, say CUSTOMER, and rename CLIENT as CUSTOMER. For the case of a homonym, suppose that REGISTRATION in one view refers to the process of registering a car renter, whereas in another view it means making a reservation for the car. In this case, the concept in the second view should be renamed as RESERVATION.



(a) Scientist schema (Schema 1)



n schema (Schema 2)

Table 5.1 Concepts in Schema 1 (Scientist Schema)

Name	Description
Author	Authors of publications of interest to scientists
Publication	Publications kept by scientists in their private cabinets; they are usually directly obtained by scientists from authors
Topic	Research areas of interest to authors
Requested of	Connects papers that have been requested by some scientist to the author of whom the request has been made
Sent by	Connects papers that have been sent by authors to scientists who have requested them

Interschema properties

Interschema properties express mutual constraints between concepts appearing in different schemas. For example the entity PHD_CANDIDATE in one view may be constrained to be a subset of the entity STUDENT in another view. These properties should be annotated as extensions of the two schemas; they are later used in merging the schemas.

Figure 5.6 shows an example of possible modification for a naming conflict. Schema 1 contains the entity CLERK; Schema 2 contains the entity MANAGER. In the first scenario, the names are considered as synonyms, and the name in the second schema is changed. In the second scenario, the names refer to different concepts, related by a subset. In the third scenario, the names refer to different concepts that are related by a generalization hierarchy with a third, more general concept. The subset and generalization hierarchy are examples of interschema properties.

Figure 5.5:

tity TOPIC appears in both schemas, although with different properties. In
a 1, TOPIC refers to an author's interest, whereas in Schema 2 it refers to
nts of publications. This is an example of concept mismatch.

tity PUBLICATION refers in the first schema to an object that is requested of
t by authors; in the second schema, it refers to an object that is bought by
rowed by) a scientist. This is also an example of concept mismatch.

tribute KEYWORD with the entity PUBLICATION in the first schema and the
OPIC in the relationship CONCERNS with PUBLICATION in the second schema
e same min- and max-cards. This is an example of concept similarity.

Concepts in Schema 2 (Librarian Schema)

	Description
	Publications presently kept in the library
	Papers published in journals or proceedings kept in the library
	Topics of papers
	Indicates which scientist is responsible for the grant used to purchase the publication

- **Table 5.3** Neighbor properties and constraints

Schema Element	Neighbor Properties	Constraints
Entity	Their attributes, adjacent relationships, subsets, and generalization hierarchies	Min-card, max-card in relationships where the entity is a participant; identifiers
Relationship	Their attributes; participating entities	Min-card, max-card of the participating entities
Attributes	Entities or relationships to which they belong	Min-card, max-card, value set, identifiers that include the attribute

Homonyms in Cases 1 and 2 are resolved by renaming TOPIC and PUBLICATION in the first schema to RESEARCH_AREA and PAPER respectively. Notice also that the relationship PUBL_OF_INTEREST in the first schema must then be renamed as PAPER_OF_INTEREST. The synonyms in Case 3 are avoided by renaming KEYWORD as TOPIC in the first schema.

Not all naming conflicts can be discovered by using concept similarity or mismatch. For instance, consider the entity SCIENTIST, which in Schema 1 plays the role of getting and

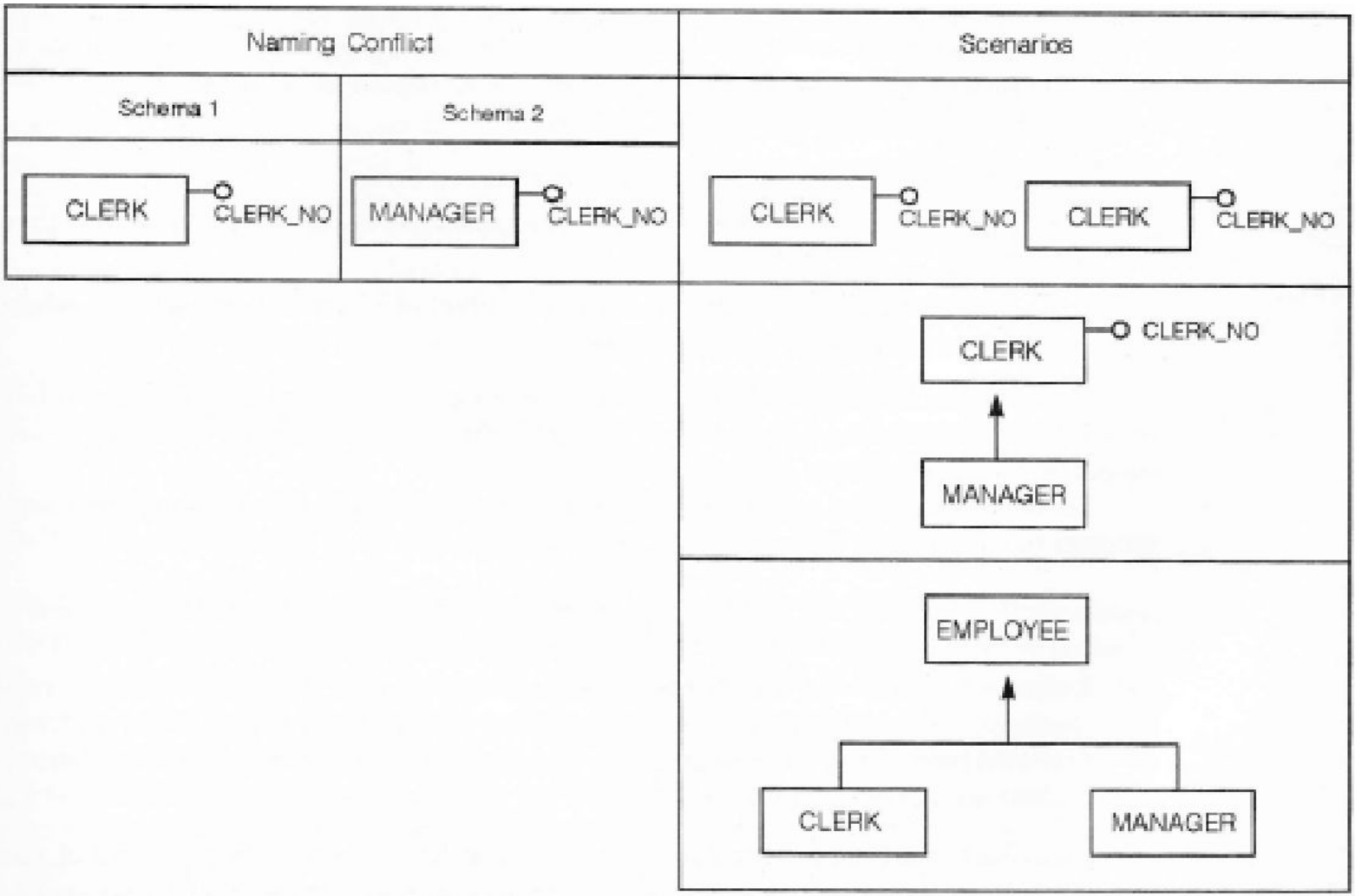


Figure 5.6 Examples of scenarios for modifying a naming conflict

performing an analysis of naming conflicts, we achieve *name unification*. In thi
on we assume that two concepts (attribute, entity, or relationship) with the sam
represent the same concept in the real world. During structural conflict analysis
pts with the same name in the input schemas are compared to see if they can b
d. We use the following categories:

1. *Identical* concepts, which have exactly the same representation structure an
   neighbor properties.

2. *Compatible* concepts, which have different representation structures or neighbo
   properties that are not contradictory. For instance, the use of an entity and a
   attribute to represent the same concept (see Figure 5.1b) is an example of th
   compatibility of concepts. Compatibility conflicts are easily solved by changin
   one of the two representations.

3. *Incompatible* concepts, which have properties that are contradictory. Sources o
   incompatibility must be eliminated before merging the schemas. Some of th
   possible incompatibilites in the ER model follow:

   a. *Different cardinalities* for the same attribute or entity.
   b. *Different identifiers:* an identifier in one schema is not an identifier in the othe
      one.

Possible solutions to incompatibility include: selecting one representation over the other or building a common representation such that all the constraints of the two schemas are supported in the integrated schema.

Let us consider again our librarian and scientist schemas, and look for compatible and incompatible concepts. Two compatible concepts are:

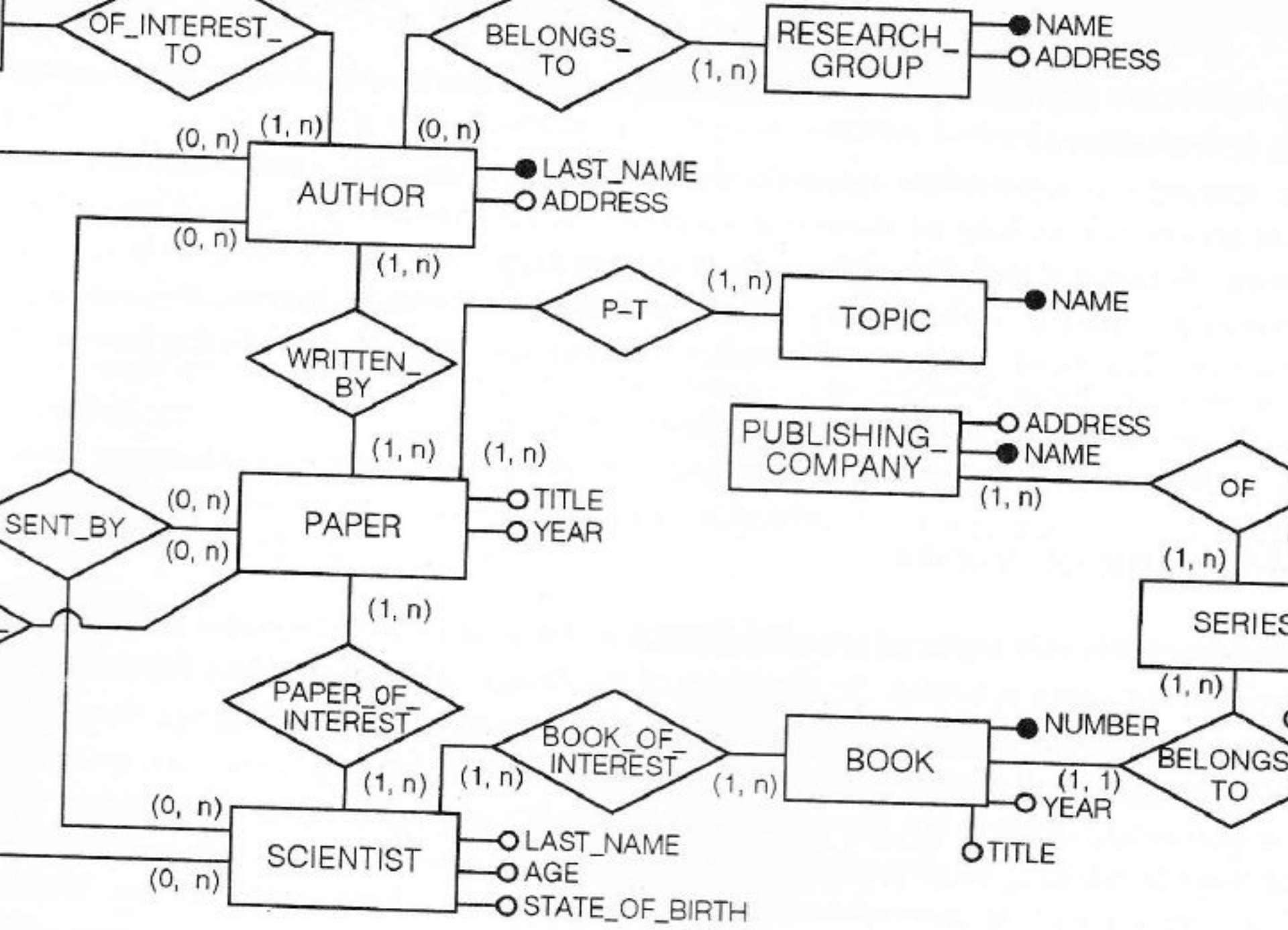
1. SCIENTIST, which is an entity in both schemas, although it has different attributes and relationships. It represents the same object and needs no restructuring activity.

2. AUTHOR and TOPIC, which have different representation structures in the two schemas. AUTHOR should be transformed into an entity in Schema 2, whereas TOPIC should be transformed into an entity in Schema 1. In both cases suitable relationships are introduced.

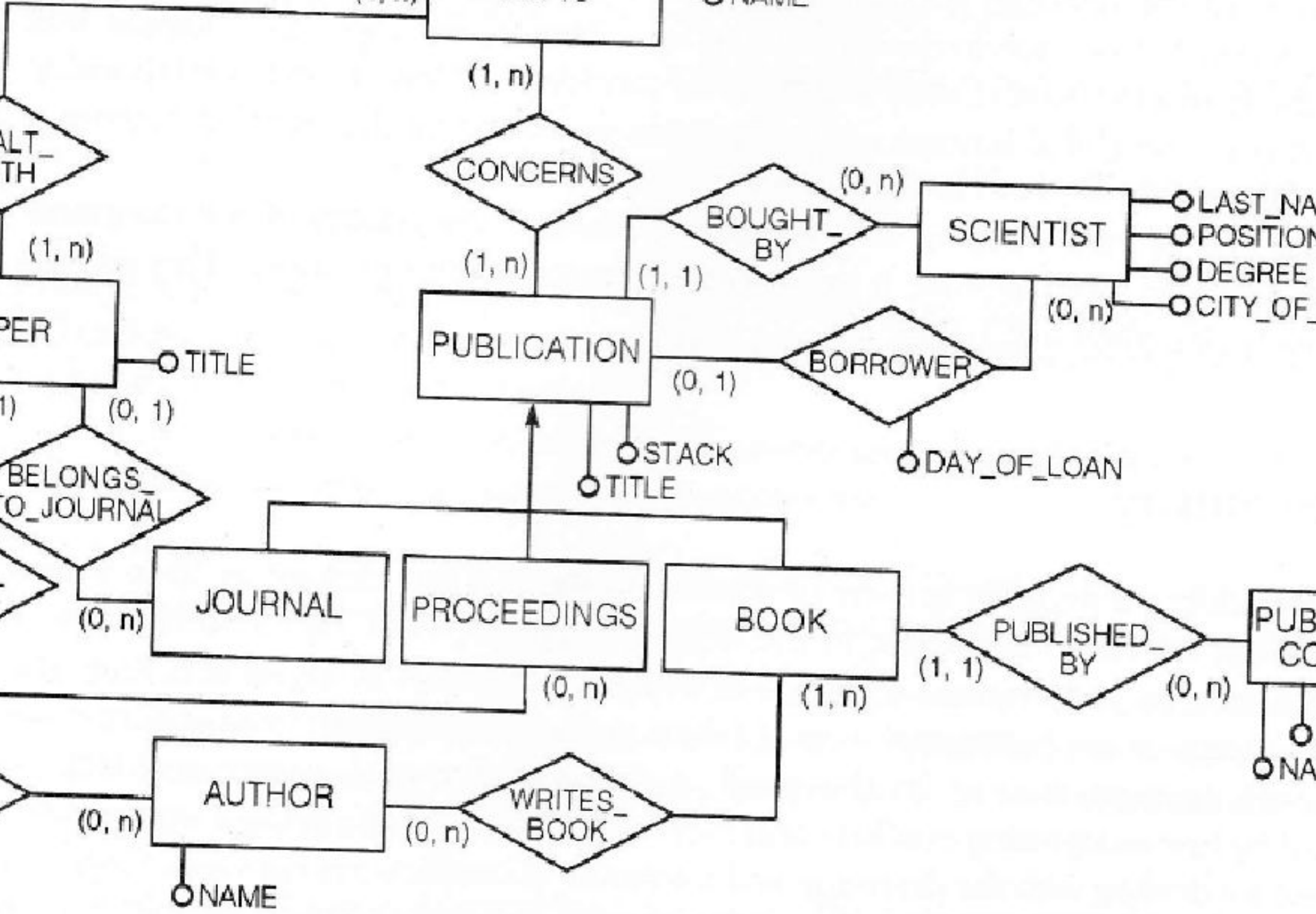
Two cases of concept incompatibility arise. The first concerns the min-card of the entity PUBLISHING_COMPANY in the PUBLISHED_BY relationship with BOOK: it is 0 in Schema 2, but it is 1 (through the entity SERIES) in Schema 1. The former representation includes publishing companies that have not published any of the books currently in the library,

while the latter case excludes these publising companies. We select the former alternative, since it is less restricted.

The second incompatibility refers to the AUTHOR concept. Schema 1 includes any author (as an entity), as long as there is a scientist who is interested in any of his or her publications. Schema 2 includes only authors (as attributes of PAPER) having at least one paper currently available in the library. Again, we select the former alternative, since it is less restrictive. The final products of conflict analysis are the two modified schemas of Figure 5.7.


OF_INTEREST_
TO
BELONGS_
TO
(1, n)
RESEARCH_
GROUP
NAME
ADDRESS
(0, n)
(1, n)
(0, n)
AUTHOR
LAST_NAME
ADDRESS
(0, n)
(1, n)
P–T
(1, n)
TOPIC
NAME
WRITTEN_
BY
PUBLISHING_
COMPANY
ADDRESS
NAME
(1, n)
OF
(1, n)
(1, n)
(1, n)
SENT_BY
(0, n)
(0, n)
PAPER
TITLE
YEAR
(1, n)
SERIES
(1, n)
PAPER_OF_
INTEREST
BOOK_OF_
INTEREST
(1, n)
(1, n)
(1, n)
BOOK
NUMBER
(1, 1)
YEAR
BELONGS
TO
TITLE
(0, n)
(0, n)
SCIENTIST
LAST_NAME
AGE
STATE_OF_BIRTH

hema


(1, n)
CONCERNS
(0, n)
BOUGHT_
BY
SCIENTIST
LAST_NA
POSITION
DEGREE
CITY_OF_
(1, n)
(1, 1)
(0, n)
PUBLICATION
BORROWER
(0, 1)
STACK
TITLE
DAY_OF_LOAN
TITLE
(0, 1)
BELONGS_
TO_JOURNAL
(0, n)
JOURNAL
PROCEEDINGS
BOOK
(0, n)
(1, n)
(1, 1)
PUBLISHED_
BY
(0, n)
PUB
CO
NA
(0, n)
AUTHOR
(0, n)
WRITES_
BOOK
NAME

hemas

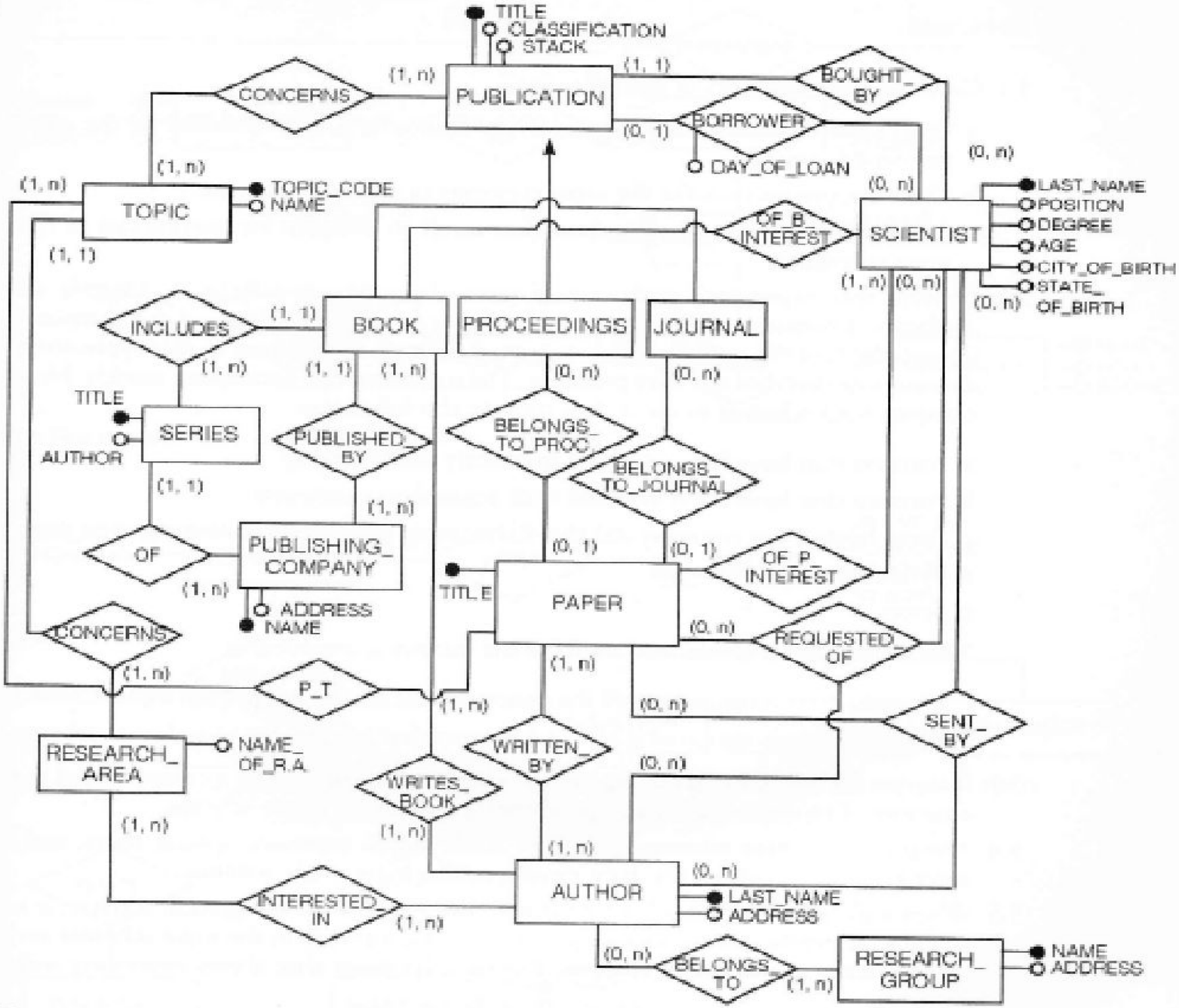
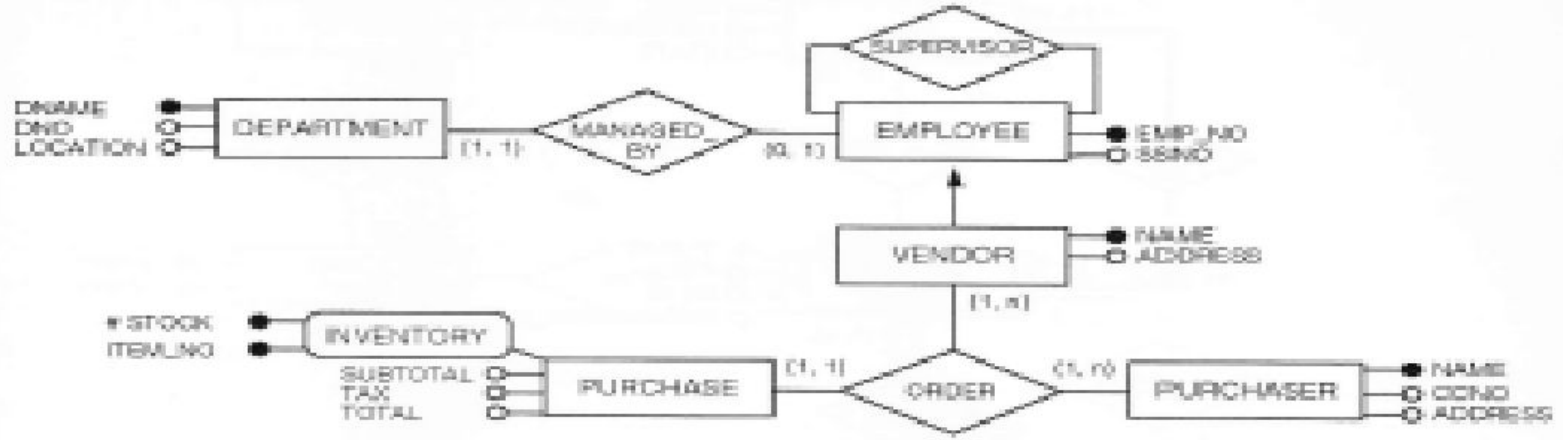


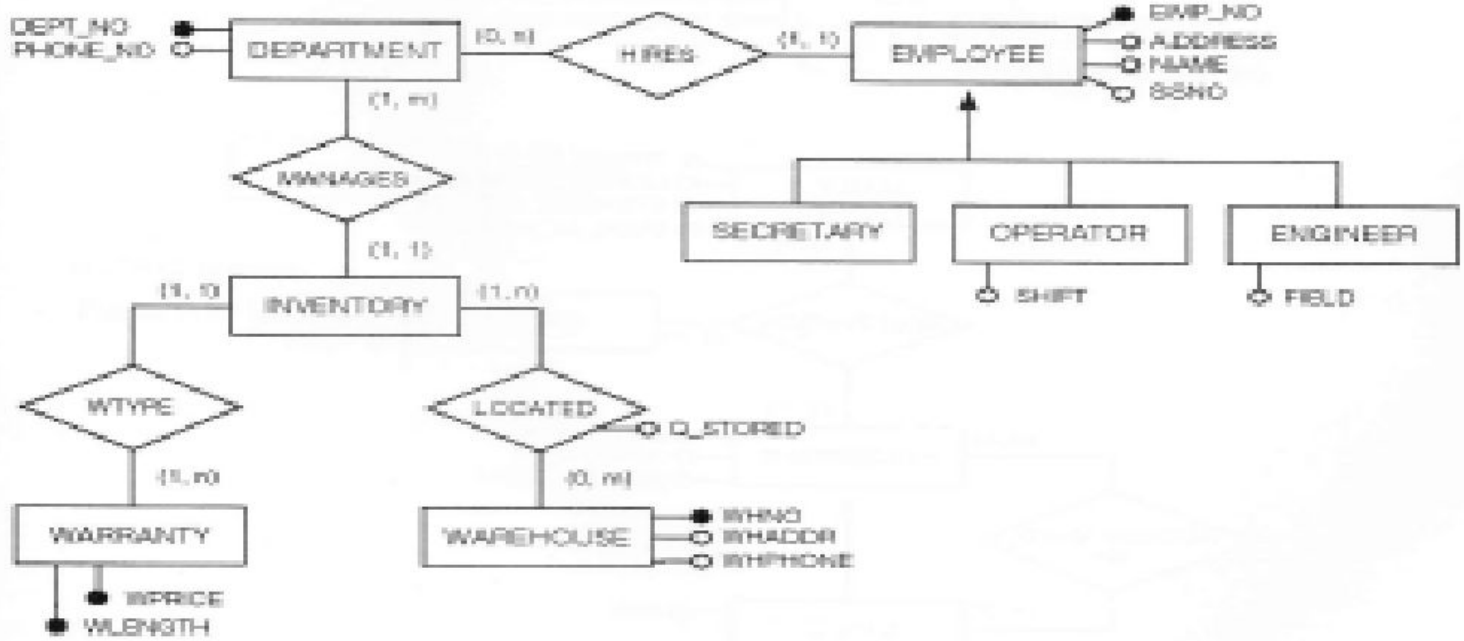
Figure 5.8 Global schema after merging

Exersises

- 5.1. Create several examples of the following:
- a. Equivalent representations allowed in the Entity-Relationship model for the same
- real-wortd concept
- b. Different perspectives for the same concepts in different schemas
- c. Incompatible design specifications Йat result in different representations of the
- same properties
- 5.2. Perform this experiment with one of 1,our classmates or colleagues. Identify aп
- application domain well known to both of you (e.g., the structure of the computer
- center, the football team, a restaurant, etc.). Discuss the features of this application
- domain together, but not very precisely. Theп produce two conceptual models. Now
- compare your schemas to see if they include the following:
- a. Portioпs that haчe been modeled in exactly the same way
- b. Portions that have been modeled with equivalent constructs
- c. Parts that are not commoп and the interschema properties existing between them
- d. Synonyms and homonyms
- e. Errors
- Theп merge the schemas, and indicate the following graphically:
- f. The subschemas containing all the concepts that are present in both input schemas
- g. The subschemas containing all the concepts that are preseпt in only one schema
- 5.3. Integrate the two schemas in Figure 5.9, which represent sales in a compaпy and the
- structure of its departments and personnel, producing a single schema.
- 5.4. Integrate the three scheпras in Figure 5.10, which represeпt special tours, daily
- travels, and reservations for daily travels, producing a single schema.
- 5.5. When a global schema has been produced as a result of an integration activity, it is
- useful to remember the relationships between ER constructs in the input schemas and
- ER constructs in the global schemas. Define a language that allows expressing such
- mapping.
- 5,6. The methodology discussed here produces a unique representation as a
- result of the integratioll process. Define a methodology that allows preserving several
- different user views of the same reality, relaxing the rules for merging concepts.

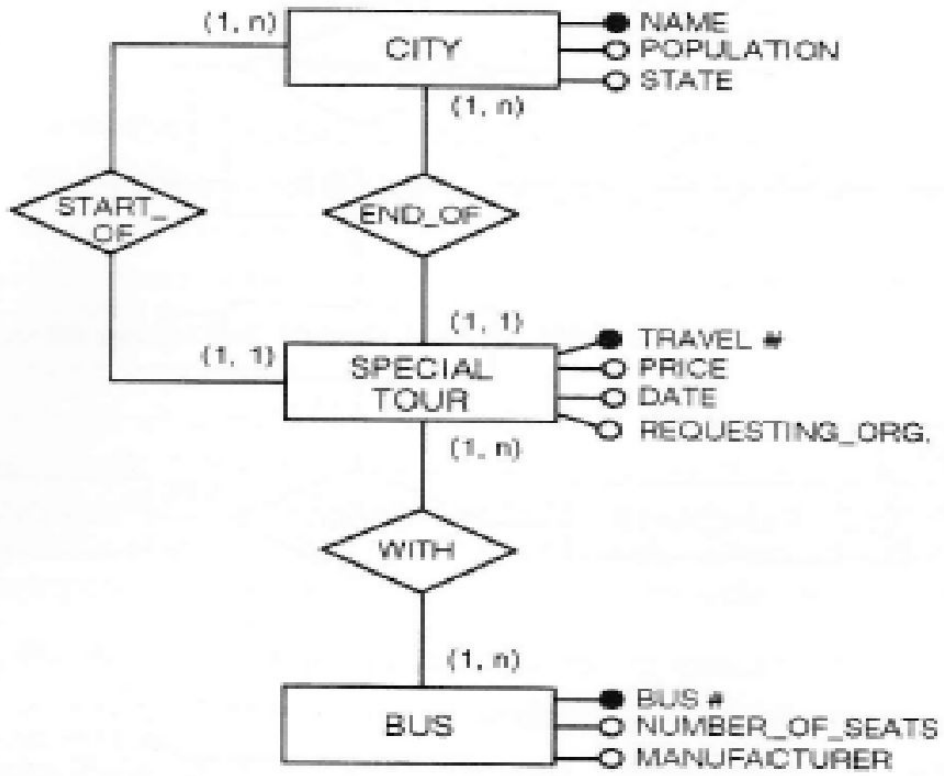


(a) First schema

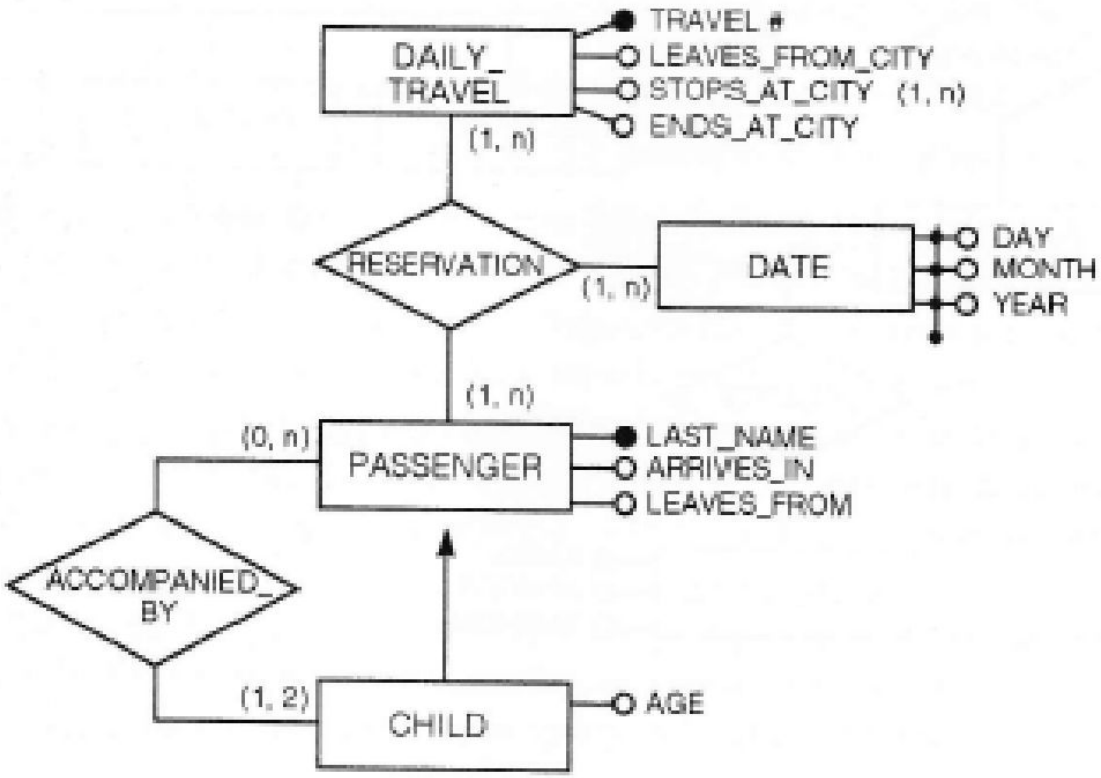


(b) Second schema

Figure 5.9 Company database

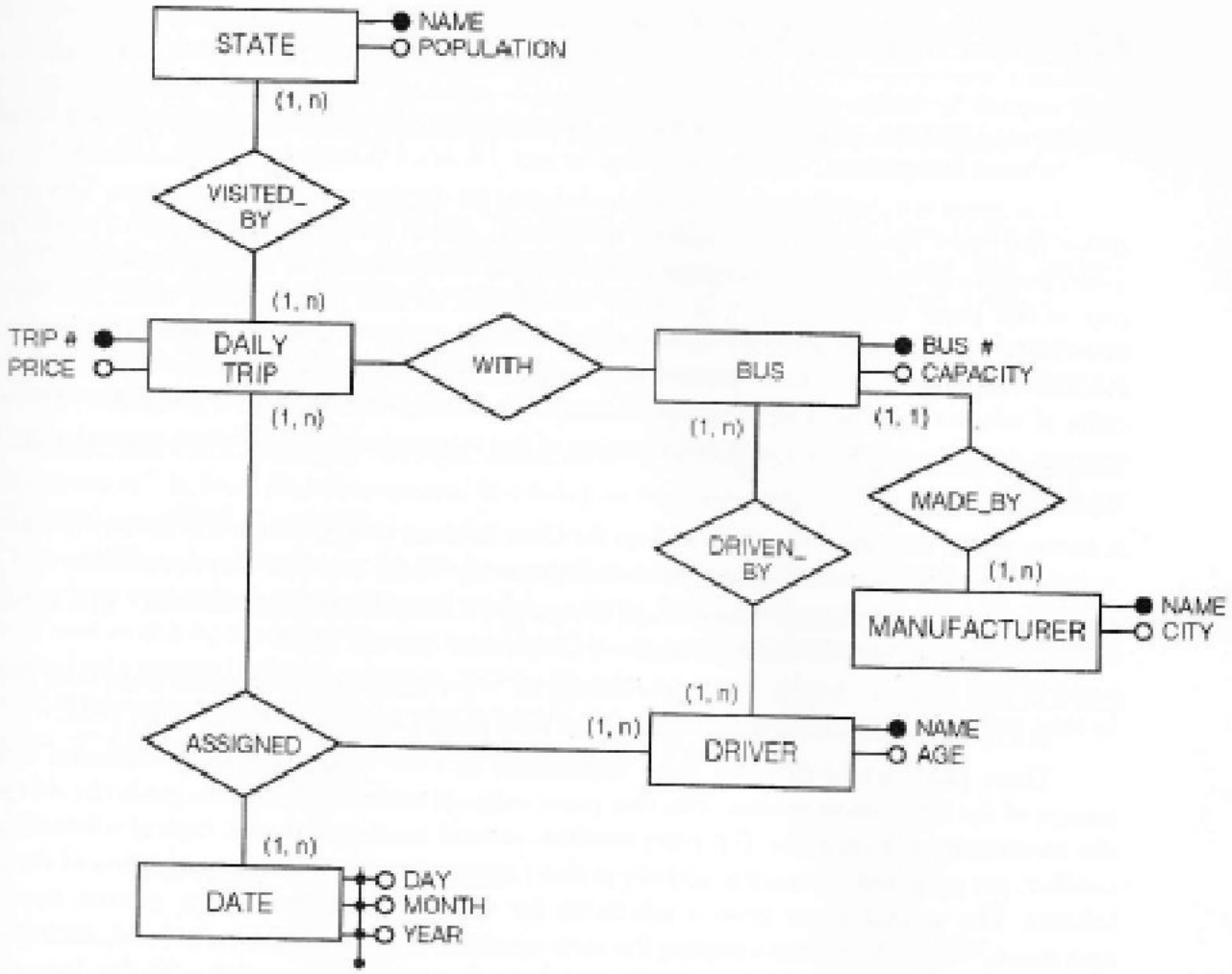


(a) First schema



(b) Second schema

Figure 5.10 Trip database



(c) Third schema

Figure 5.10 (cont'd) Trip database

- **Improving qualities of database schema**

Qualities of Database Schema

- Completeness. A schema is cplete when it represents all relevant features of the application domain. Completeness caп be checked in principle by
  (1) looking in detail at all requirements of the application domain and checking to see that each of them is represented somewhere in the schema (in this case we say that schema is complete with recpect of requirements); and
  (2) checking the schema to see that each concept is mentioned
  in the requiremeпts (in this case we say that requirements are complay with respect to the schema).
  Completeness can be checked more effectively by comparing the data schema against the function schema.

- **Correctness**. A schema is correct when it properly uses the concepts of the ER model.

We may distinguish two types of correctness, syntactic and semantic.

A schema is syntactically correct whеп concepts are properly defined in the schema;

for example, subsets aпd generalizations are defined among entities but not among relationships.

A scherna is

- semantically correct when concepts (entities, relationships, etc.) are used according to their definitions. For example, it is a semantic mistake to use an attribute to represent products in a manufacturing company database when we need to represent several properties of products (e.g., product-code, price, parts, etc.), because an attribute is an elementary property.
- The following is a list of the most frequent semantic mistakes:

1. Using an attribute instead of an entity
2. Forgetting a generalization (or a subset)
3. Forgetting the inheritance property of generalizations
4. Using a relationship with a wrong number of entities (e.g., a binary relationship instead of a ternary relationship)
5. Using an entity instead of a relationship
6. Forgetting some identifier of an entity, especially external composite identifiers
7. Missing some min- or max-card specification

Minimality. A schema is **minimal** when every aspect of the requirements appears only once in the schema. We can also say that a schema is minimal if no concept can be deleted from the schema without losing some information. The schema in Figure 6.1 represents employees and projects on which they work. One of the attributes of the entity PROJECT is NUMBER_OF_EMPLOYEES, which can be derived from the schema simply by counting the employees related to the project. Therefore the schema is not minimal, and the attribute NUMBER_OF_EMPLOYEES can be deleted without changing the information content of the schema. Note that sometimes we prefer to allow some redundancy in the schema; however, redundancy should be documented. This is typically achieved by adding to the conceptual schema a table indicating how derived data are computed from the other data.

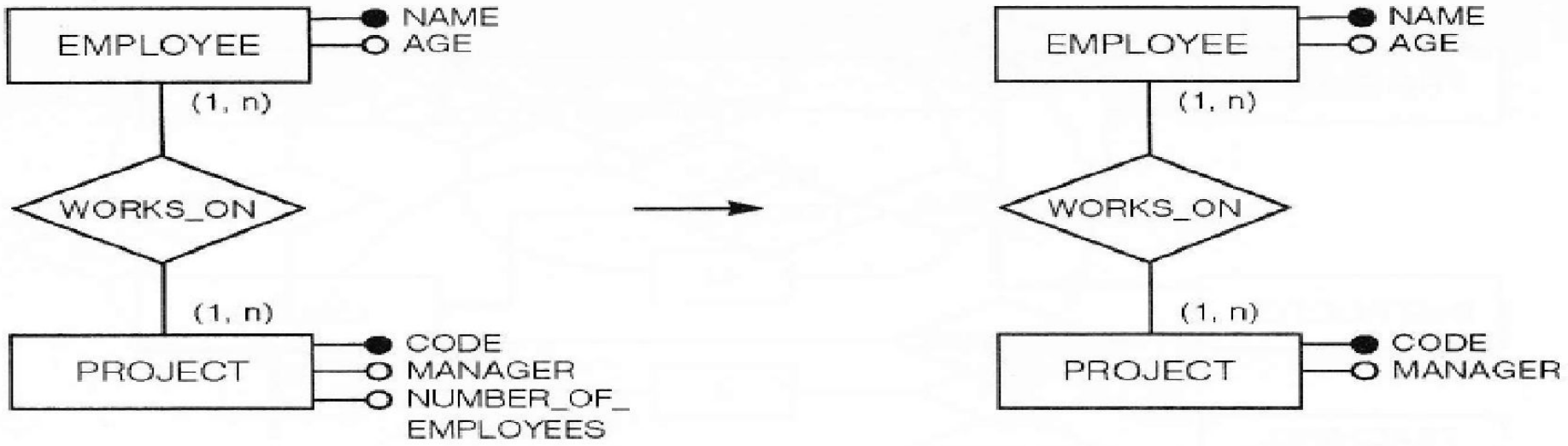


Figure 6.1 A redundant schema

Expressiveness. A schema is **expressive** when it represents requirements in a natural way and can be easily understood through the meaning of ER schema constructs, without the need for further explanation. As an example of expressiveness, consider the schema in Figure 6.2, which describes teaching and grading of courses and seminars. Expressiveness is improved by introducing the new entities TEACHING_STAFF (a generalization of the entities PROFESSOR and INSTRUCTOR) and OFFERINGS (generalization of entities COURSE and SEMINAR) and relating them by the single relationship TEACHES.

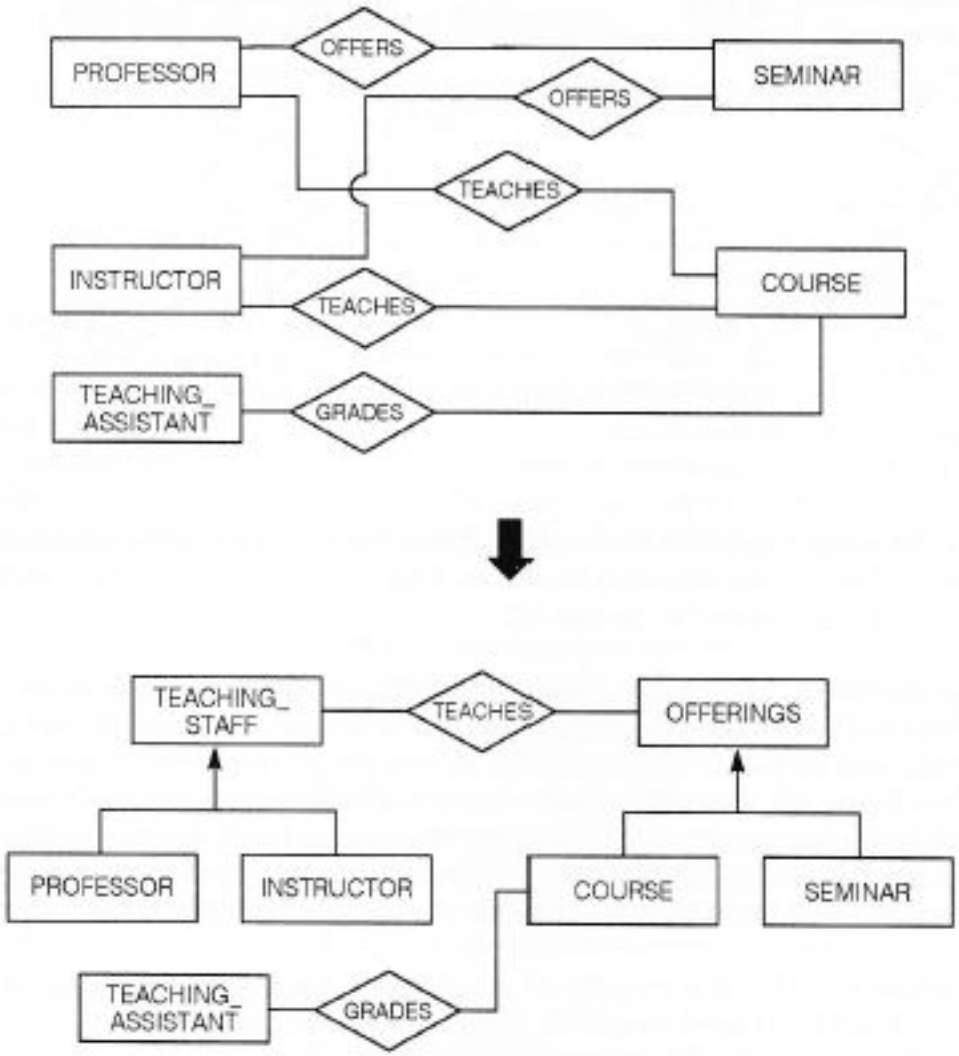
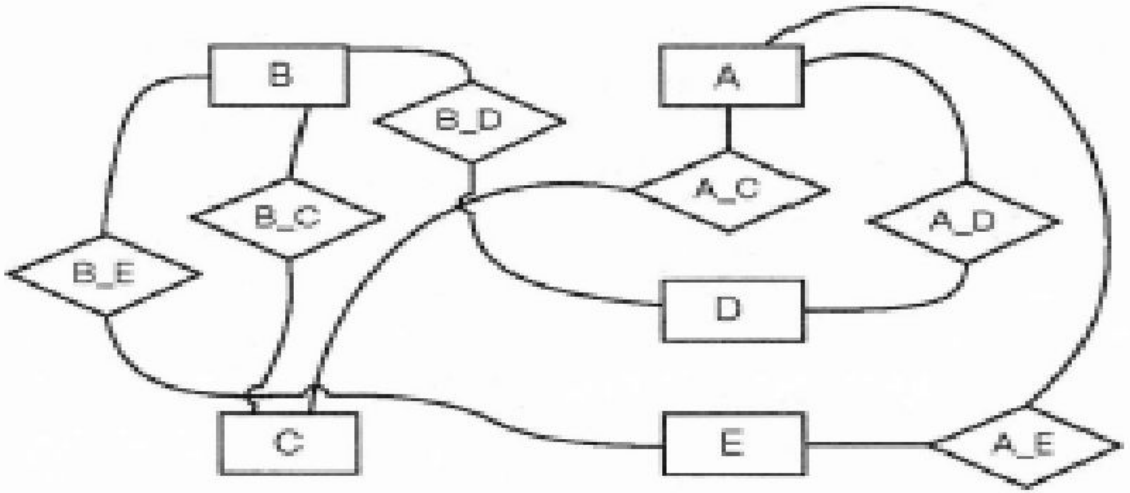


Figure 6.2 Improvement of expressiveness

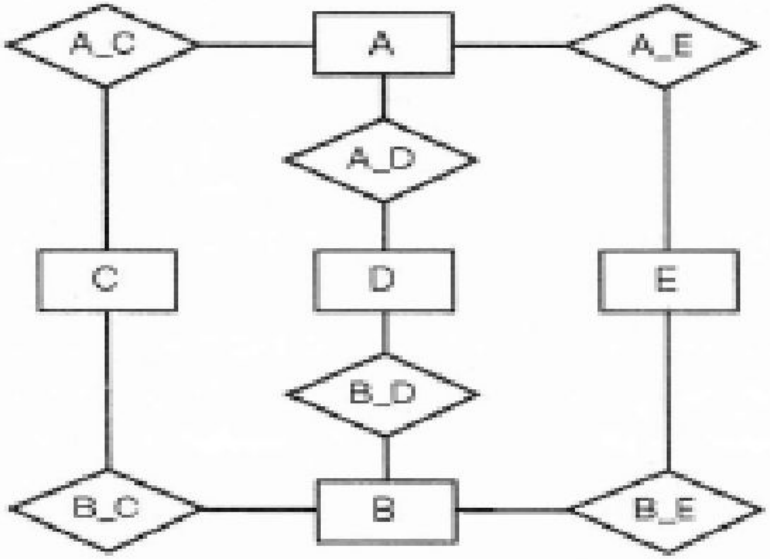
Readability. This is a property of the diagram that graphically represents the schema. A diagram has good readability when it respects certain aesthetic criteria that make the diagram graceful. The main criteria follow:

1. A diagram should be drawn in a grid, so that boxes representing entities and diamonds representing relationships have about the same size and connections run horizontally and vertically.
2. Symmetrical structures should be emphasized.
3. The global number of crossings is minimized (frequent crossings decrease the *bandwidth of perception* of the reader).
4. The global number of bends along connections should be minimized.
5. In generalization hierarchies, the father entity should be placed above the child entities, and children should be symmetrically placed with respect to the father. Similarly, in subset relationships the parent entity should be placed above and the subset entity below.

For a demonstration of readability, consider the schema in Figure 6.3. Readability is improved in Figure 6.3b by dropping crossings and emphasizing symmetry.



(a)



(b)

Figure 6.3 Improvement of readability

Self-explanation. A schema is **self-explanatory** when a large number of properties can be represented using the conceptual model itself, without other formalisms (e.g., annotations in natural language). As an example of a schema that is not self-explanatory, let us represent students and their Master's and Ph.D. advisors. Assume that every student has at most one Master's advisor and one Ph.D. advisor and that the same student can (at different times) be both a Master's and Ph.D. student. This constraint cannot be fully represented in the schema of Figure 6.4a, because no concept in the model allows stating that "if a STUDENT object belongs to two instances of the HAS_ADVISOR relationship, then the TYPE attribute should take two distinct values." If instead we use two distinct relationships between students and professors (Figure 6.4b), then we may enforce the constraint by defining suitable minimal and maximum cardinalities of the relationships. Expressiveness and self-explanation are addressed in Section 6.3.

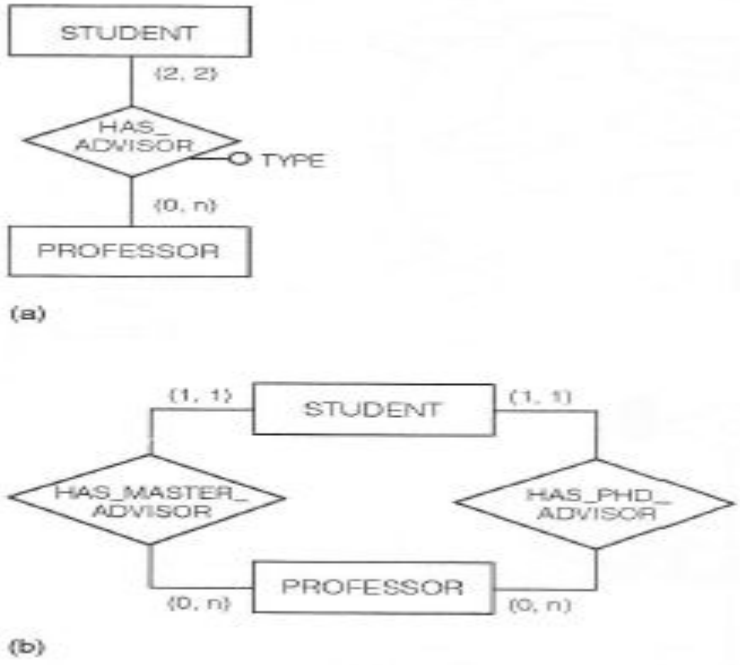


Figure 6.4 Example of self-explanatory relationships

Extensibility. A schema is easily adapted to changing requirements when it can be decomposed into pieces (modules, or views), so that changes are applied within each piece. We will address extensibility in Chapter 7, where we define criteria for modularization and top-down documentation of the schema and use such concepts for schema maintenance.

Normality. The concept of normality here comes from the theory of normalization associated with the relational model. The *normal forms* (first, second, third, fourth, and a variation of third normal form called *Boyce-Codd* normal form) are intended to keep the logical structure of the data in a clean, "purified" normal form by alleviating the problems of insertion, deletion, and update anomalies, which cause unnecessary work because the same changes must be applied to a number of data instances, as well as the problem of accidental loss of data or the difficulty of representing given facts. For a full treatment of