View Design

Activities:

1.Analysis of requirements to capture objects&classes

2.Representation of objects&classes&properties using
concepts of ER-model

View Design from Natural Language requirements

1. Requirements analysis
   1.1. Analyze requirements and filter ambiguities
   1.2. Partition sentences into homogeneous sets
2. Initial design
   2.1. Build a global skeleton schema
3. Schema design—to each concept in the skeleton schema, apply
   3.1. Top-down primitives
   3.2. Bottom-up primitives
   3.3. Inside-out primitives

   until all requirements have been expressed in the schema.

Requirements for University Data Base

In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, places and states where they lived before (with the period they lived there), courses that they have passed, with name, code, professor, grade, and date. We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once in one day). For graduate students, we represent the name of the advisor and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name, age, place and state of birth, name of the department they belong to, telephone number, title, status, and topics of their research.

Rules for searching inaccuracies and ambiguities (1)

1. **Choose the Appropriate Level of Abstraction for Terms.** Abstract terms are frequently used in real-life sentences where specific terms would be more appropriate in order to clarify the situation. General categories are common in natural language because they lead to fast, effective communication that is usually disambiguated by the context. However, in conceptual design one should use terms at the correct abstraction level, especially if the designer is not an expert in the application domain. In our example, the following abstract terms appear: *places*, *period*, and *status*; the corresponding appropriate terms are: *cities*, *number of years*, *marital status*.

2. **Avoid Using Instances Instead of General Concepts.** This rule prevents the opposite source of ambiguity; users of the information system sometimes adopt terms that are more specific than needed. For example, in an electronics company a storekeeper may say, "Every day I need to know the stock quantity of chips." The term *chips* does not describe a concept, but rather an instance of the correct concept, that is, components. Thus the preferred term would be *components*.

Rules for searching inaccuracies and ambiguities (2)

3. Avoid Roundabout Expressions. In natural language we frequently use deliberate repetition and roundabout expressions. We may say, "Look at the person sitting at the booking window" instead of "Look at the booking clerk." The latter sentence indicates a specific class of entities (clerk), whereas the former refers to the same class by indicating a relationship with another class of entities (person). Thus, the second sentence enables a clearer classification of concepts. If we use circumlocutions, we incur the risk of expressing the meaning of concepts in terms of implicit references to other concepts instead of explicit references to concepts themselves.

4. Choose a Standard Sentence Style. In free conversation we use many syntactic styles to achieve more effective communication. This variety of styles should be avoided in texts describing requirements; using simple syntactic categories enables a more straightforward (and unique) modeling of requirements. Ideally, we should produce sentences that have some standard style; for instance, data descriptions should be of the form <*subject*> <*verb*> <*specification*>. Sentences describing operations should use, as much as possible, unambiguous syntactic structures similar to those of programming languages, like <*if*> <*condition*> <*then*> <*action*> <*else*> <*action*> or <*when*> <*condition*> <*do*> <*action*>. Thorough application of this rule is not always possible or convenient; the designer should select an appropriate style as a trade-off between standardization and expressiveness.

Rules for searching inaccuracies and ambiguities (3)

5. Check Synonyms and Homonyms. Requirements usually result from the contributions of several users. Different persons may give the same meaning to different words (*synonyms*), or different meaning to the same words (*homonyms*). In general, the risk of homonyms is higher when the vocabulary of terms is small, while the risk of synonyms is higher when the vocabulary of terms is rich. Moreover, if two different users adopt vocabularies at different abstraction levels, they incur the risk of synonyms. In our example, the three different terms *teacher*, *professor*, and *advisor* refer to the same concept (they are synonyms). *Places* is used two times with different meanings (homonym).

6. Make References among Terms Explicit. Some ambiguities arise because references among terms are not specified. In our example, it is not clear whether *telephone number* is a property of professors or of departments. Notice that referred concepts may be either explicitly mentioned in requirements (*professors* and *departments*) or not mentioned at all (this is true for *day*, which can be interpreted as *day of the week* or *day of the month*; the terms *week* and *month* do not appear in requirements).

Rules for searching inaccuracies and ambiguities (4)

7. Use a Glossary. Building a **glossary of terms** is a good way (though rather time-consuming) to understand the meaning of terms and remove ambiguities from requirements. After building a comprehensive glossary, only the terms from the glossary should be used in requirement descriptions. The glossary should include for each term (1) its name; (2) a short definition (5 to 20 words) that is acceptable to all users of the term; (3) possible synonyms, that is, terms that have the same meaning for users (synonyms express the *area of equivalence* of the term); and (4) possible *key words*, that is, words logically close to the term (key words express the *area of influence* of the term).

Ambiguous terms in requirements and possible corrections

Line	Term	New Term	Reason for the Correction
5	Places	Cities	*Place* is a generic word
6	Period	Number of years	*Period* is a generic word
9	Presently	In the current year	*Presently* is ambiguous
9	Day	Day of the week	More specific
9	Places	Rooms	Homonym for *places* in line 5
10	Classes	Courses	Synonym for *courses* in line 8
15	Teacher	Professor	Synonym for *professor* in line 2
16	Place	City	Same as in line 5
17	Telephone	Telephone of the department	More specific
18	Status	Marital status	*Status* is ambiguous
18	Topic	Research area	Synonym for *research area* at line 14

Rewritten requirements

- In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, cities and states where they lived before (with the number of years they lived there), courses that they have passed, with name, code, professor, grade, and date. We also represent courses they are attending in the current year, and for each day of the week, rooms and hours where courses are held (each course meets at most once in one day). For graduate students, we represent the name of the advisor and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For professors, we represent their last name, age, city and state of birth, name of the department they belong to, telephone number of the department, title, marital status, and research area.

- To do it we analise the text and decompose it into the set of sentences so that each set of sentences refers to the same concept

(partitioning sentences into gomogenous groups)

In a university database, we represent data about students and professors.
General sentences
For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, cities and states where they lived before (with the number of years they lived there), courses that they have passed, with name, code, professor, grade, and date.
Sentences on students
We also represent courses they are attending in the current year, and for each day of the week, rooms and hours where courses are held (each course meets at most once in one day).
Sentences on courses

For graduate students, we represent the name of the advisor and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis.

Sentences on specific types of students

For professors, we represent their last name, age, city and state of birth, name of the department they belong to, telephone number of the department, title, marital status, and research area.

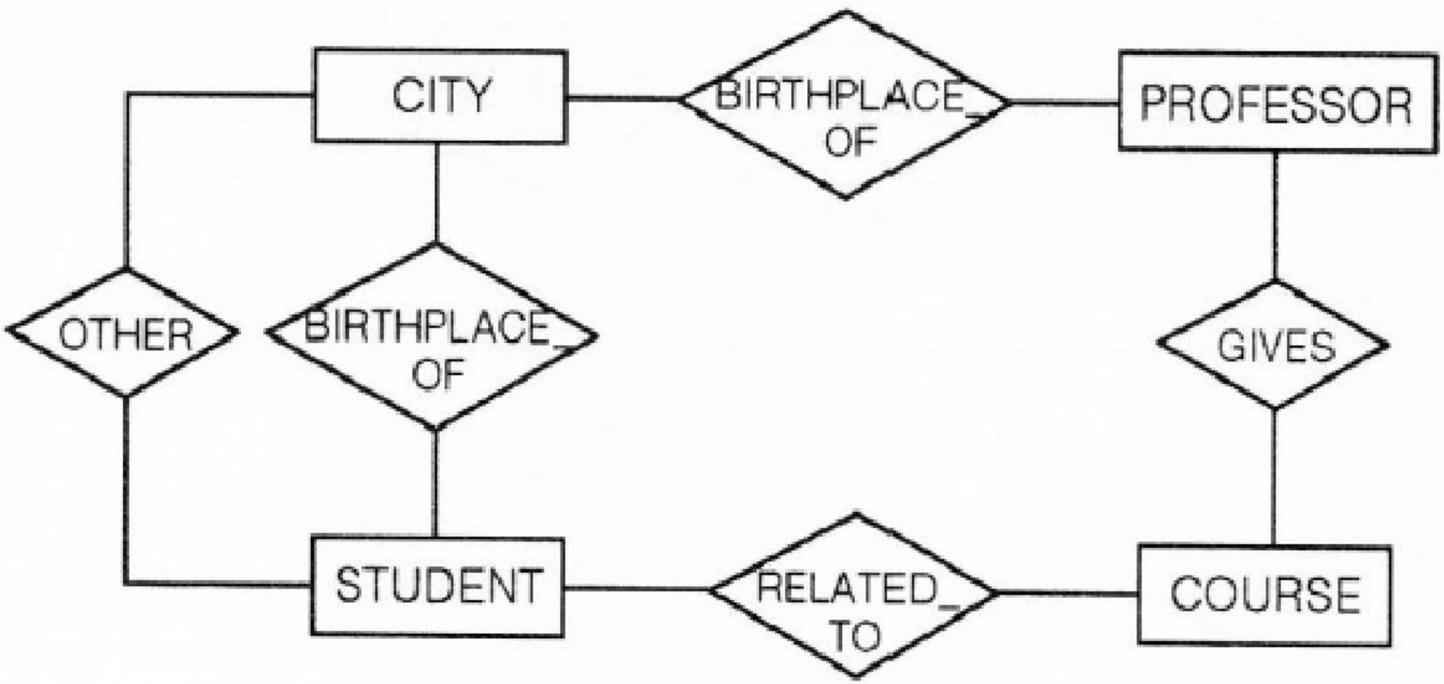
Sentences on professors

Initial Design

•

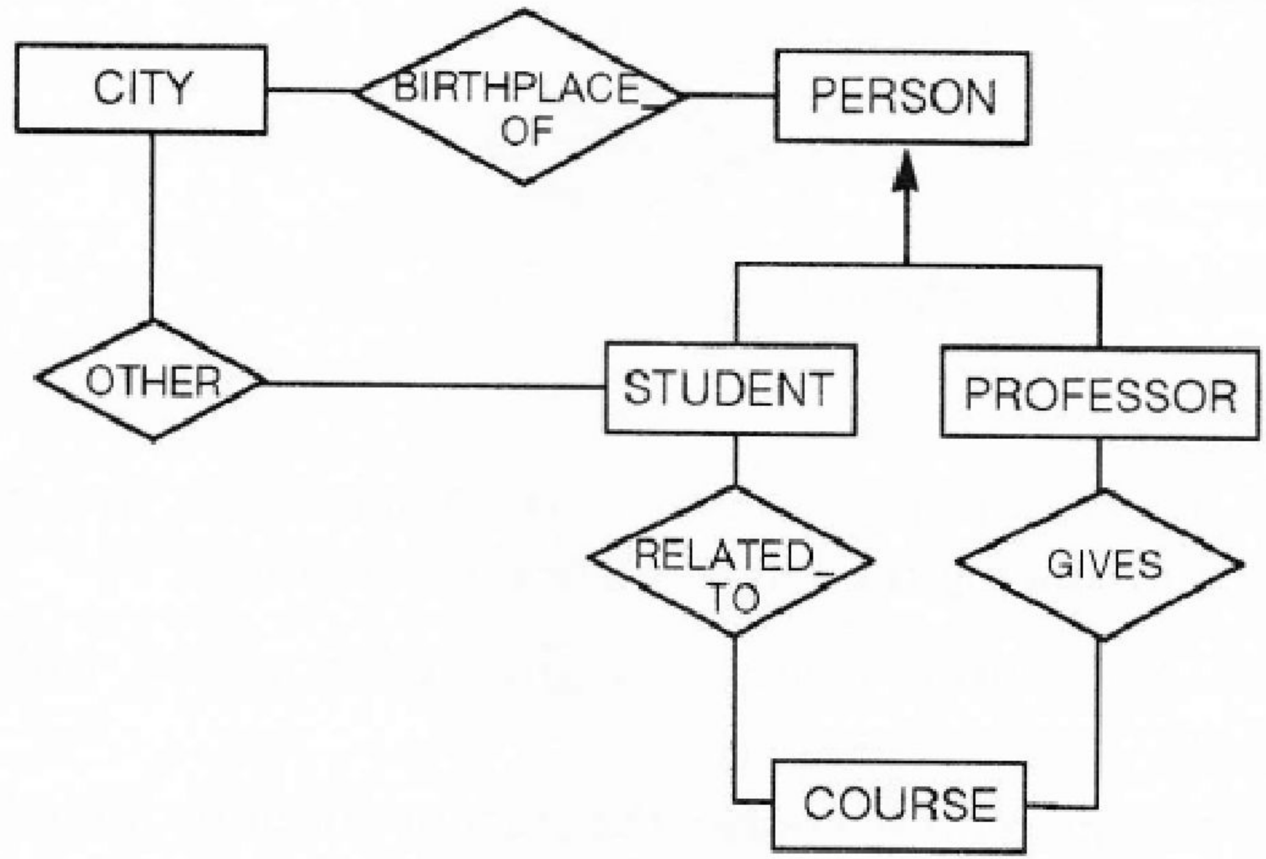
The goal of initial design is to build a skeleton schema. Concepts that appear in the skeleton schema are the most *evident* concepts referenced in requirements. We start by considering the grouping of sentences determined during requirements analysis: concepts referred to in each group are good candidates to become entities of the skeleton schema; in our example, they are STUDENT, PROFESSOR, and COURSE. We add CITY, which is an easily recognizable entity.

Once an initial group of entities is chosen, we can superimpose on them an initial network of relationships, corresponding to logical links among groups of sentences. Thus, the relationship BIRTHPLACE_OF connects CITY with STUDENT and PROFESSOR, GIVES connects PROFESSOR and COURSE, RELATED_TO connects COURSE and STUDENT, and OTHER connects CITY and STUDENT. The last two relationships are intentionally vague and will be refined later.



(a) First skeleton schema

Checking and restructuring the first skeleton schema (introducing entity PERSON)

- 

(b) Refined skeleton schema

Schema Design

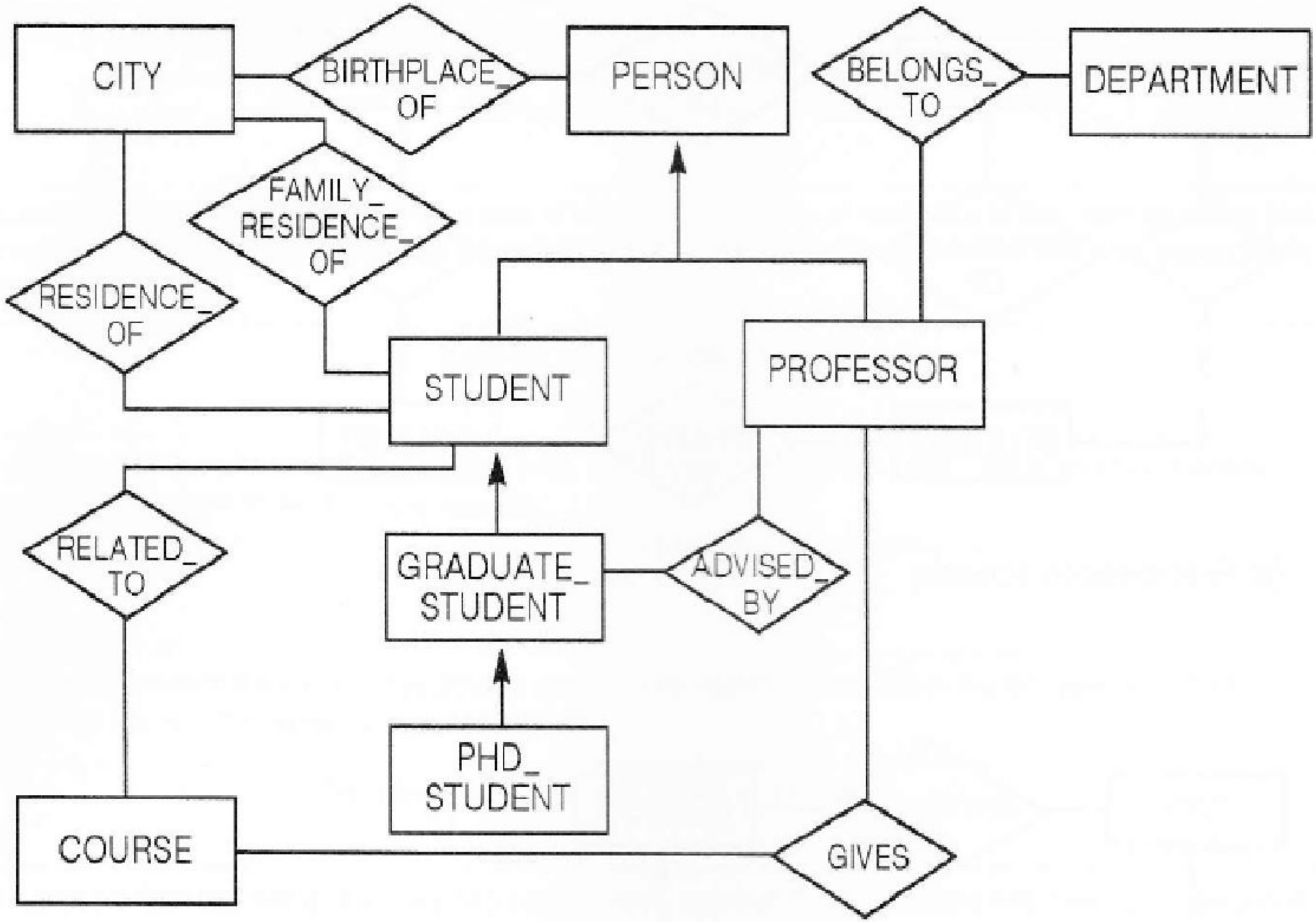
1. Top-down refinements.

   a. The entity STUDENT can be refined in terms of two subsets, GRADUATE_STUDENT and PHD_STUDENT.

   b. Relationship OTHER, between entities CITY and STUDENT, can be refined in terms of two relationships: RESIDENCE and FAMILY_RESIDENCE.

2. Bottom-up refinements. Having inserted in the schema the entity GRADUATE_STUDENT, we look at the requirements and note that a relationship exists between graduate students and their advising professors (advisors). This relationship, named ADVISED_BY, can now be inserted in the schema.

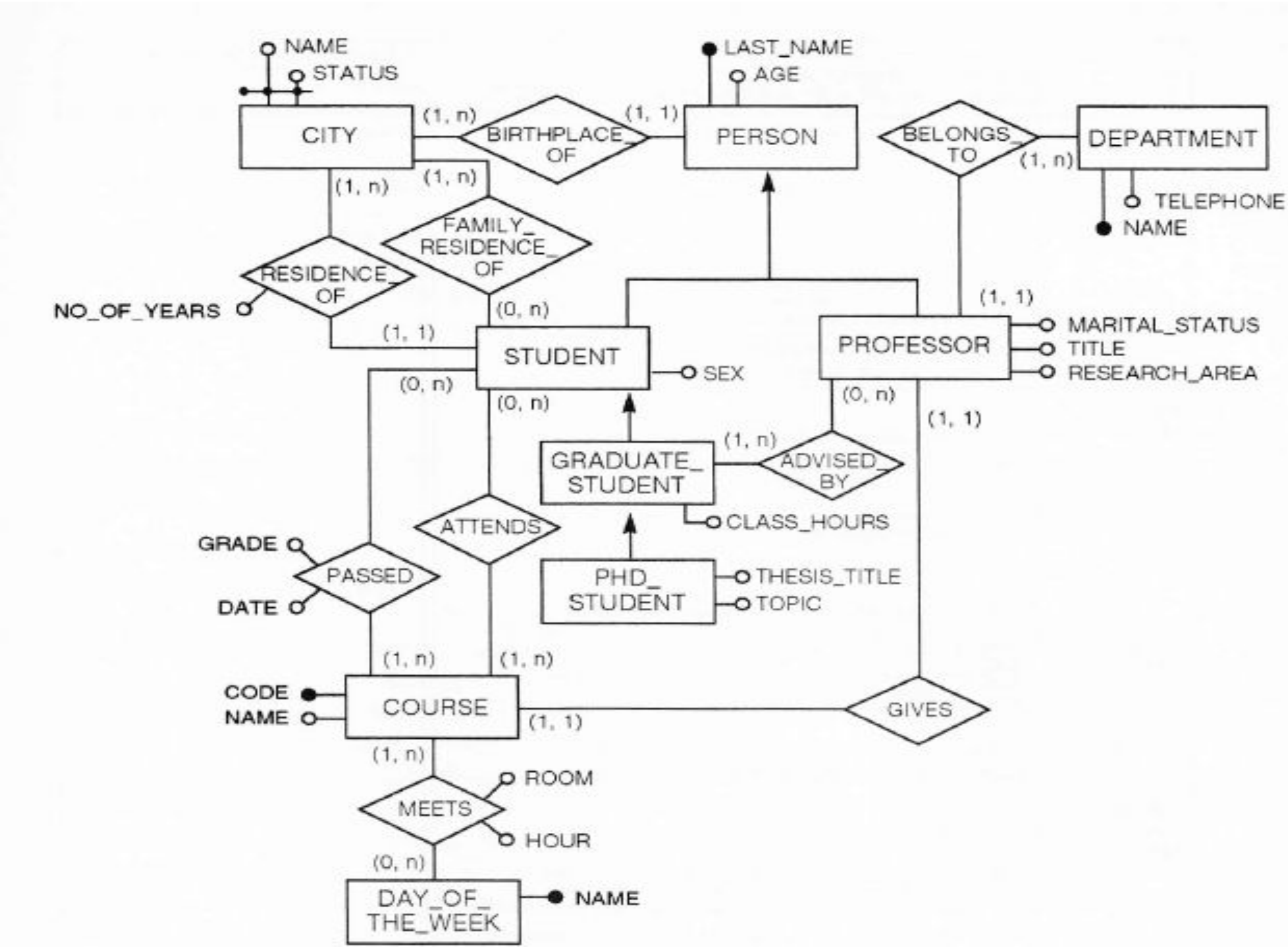
3. Inside-out refinements. One of the properties of the entity PROFESSOR is DEPARTMENT. Since several properties are associated with departments (name, address, and telephone number), we can represent DEPARTMENT as an entity and represent the logical link between PROFESSOR and DEPARTMENT by means of the relationship BELONGS_TO. The schema resulting from the application of these primitives is shown


CITY
BIRTHPLACE_OF
PERSON
BELONGS_TO
DEPARTMENT
FAMILY_RESIDENCE_OF
RESIDENCE_OF
STUDENT
PROFESSOR
RELATED_TO
GRADUATE_STUDENT
ADVISED_BY
PHD_STUDENT
COURSE
GIVES

- In order to proceed to the final refinements we may now focus on each concept of schema and check for completeness. Thus we define attributes for each entity or relationship and we specify identifiers and mappings.

- We notice that textual requirements poorly expressed by the RELATED-TO relationship between STUDENT and COURSE. In fact, this relationship must be refined by introducing the following new relationships:
- 1. The relationship PASSED, representing courses that the student passed, with two attributes: GRADE and DATE.
- 2. The relationship ATTENDS, representing courses the student currently attends.
- 3. The relationship MEETS, beween COURSE and the пew entity DAY OF_THE_WEEK, reptesenting the weekly schedule of classes attended by students in the current year, with two attributes: ROOM and HOUR.
- We complete the schema by adding some other attributes, cardinalities of relationships, and identifiers.

The final schema



View design starting from forms

Forms are structured documents used for exchanging information within organizations, in particular for providing data entry information to automated information systems. Since forms are user-oriented, they should be easily understandable.

Form structure

- Certificating part
- Extentional part
- Intentional part
- Descriptive part

Details of parts definition

- Certification part contains information that certify existence and correctness of the form, such as identifiers, date of issue, stamps, marks, and signatures. Usually this part does not convey relevant semantic information, and we make no further reference to it. The *extensional part* is the set of fields that are filled by user-provided values when the form is compiled. This is the information that a person enters on a preprinted form. The *intensional part* is the set of implicit or explicit references to names of fields on the form. This is the information that is preprinted on paper forms. The *descriptive part* contains instructions or rules that should be followed in order to fill the fields of the extensional part.

Name(s) shown on Form 1040A. (Do not complete if shown on other side.) | Your social security number

Part I (continued)

Complete lines 13 through 20 only if you received employer-provided dependent care benefits. Be sure to also complete lines 1 and 2 of Part I.

13	Enter the total amount of employer-provided dependent care benefits you received for 1989. (This amount should be separately shown on your W-2 form(s) and labeled as "DCB.") DO NOT include amounts that were reported to you as wages in Box 10 of Form(s) W-2.	13	
14	Enter the total amount of **qualified** expenses incurred in 1989 for the care of a qualifying person. (See page 34 of the instructions.)	14	
15	Compare the amounts on lines 13 and 14. Enter the **smaller** of the two amounts here.	15	
16	You **must** enter your **earned income.** (See page 34 of the instructions for the definition of earned income.)	16	
17	If you were married at the end of 1989, you **must** enter your spouse's earned income. (If your spouse was a full-time student or disabled, see page 34 of the instructions for the amount to enter.)	17	
18	• If you were married at the end of 1989, compare the amounts on lines 16 and 17 and enter the **smaller** of the two amounts here. • If you were unmarried, enter the amount from line 16 here.	18	
19	**Excluded benefits.** Enter here the **smallest** of the following: • The amount from line 15, or • The amount from line 18, or • $5,000 ($2,500 if married filing a separate return).	19	
20	**Taxable benefits.** Subtract line 19 from line 13. Enter the result. (If zero or less, enter -0-.) Include this amount in the total on Form 1040A, line 7. In the space to the left of line 7, write "DCB."	20	

Note: *If you are also claiming the child and dependent care credit, first fill in Form 1040A through line 20. Then complete lines 3-12 of Part I.*

Part II

Note: *If you received a Form 1099-INT or Form 1099-OID from a brokerage firm, enter the firm's name and the total interest shown on that form.*

Interest income (see page 24 of the instructions)

Complete this part and attach Schedule 1 to Form 1040A if you received over $400 in taxable interest.

1 List name of payer		Amount	
	1		
2 Add amounts on line 1. Enter the total here and on Form 1040A, line 8a.	2		

Part III

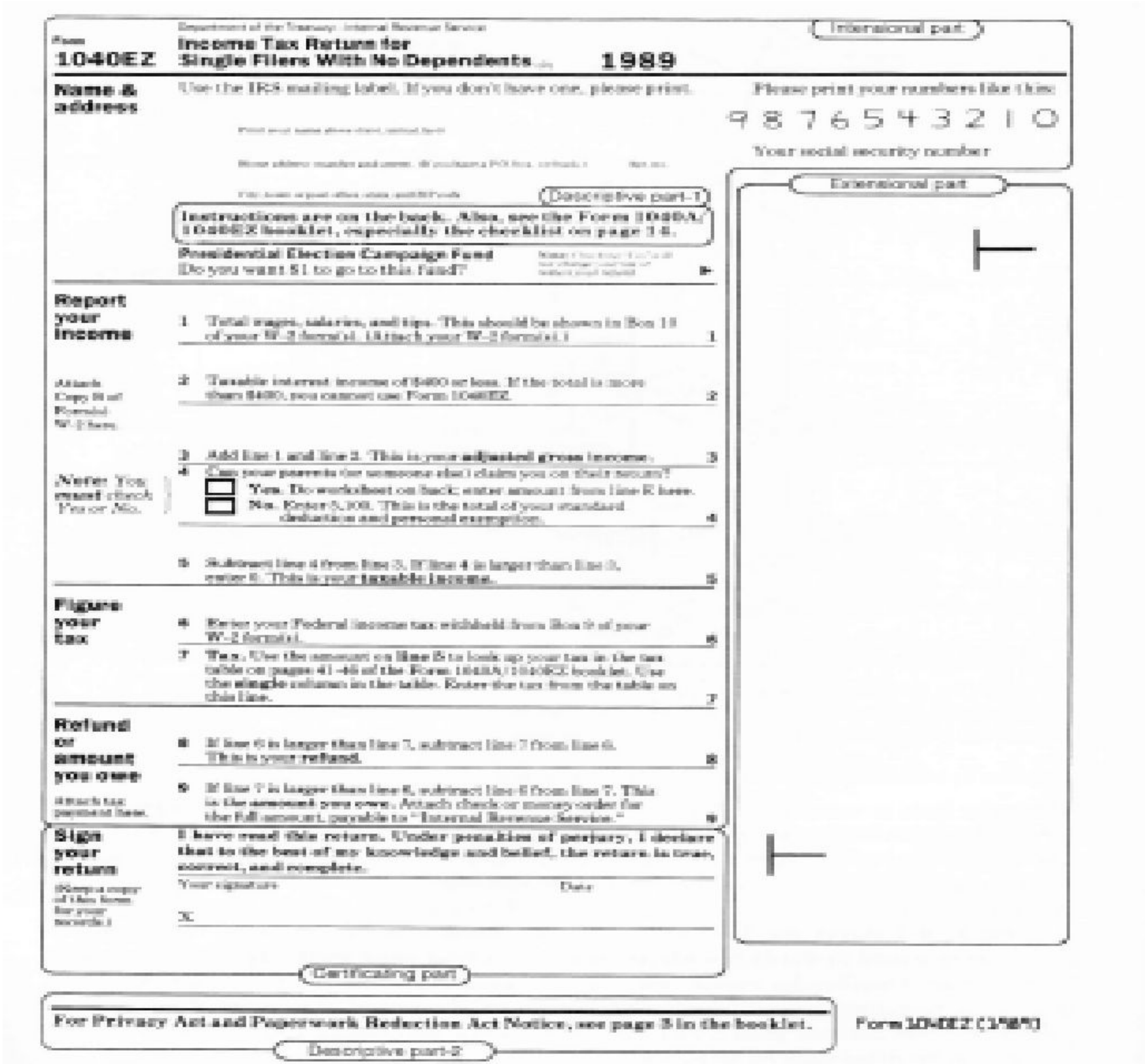
Note: *If you received a Form 1099-DIV from a brokerage firm, enter the firm's name and the total dividends shown on that form.*

Dividend income (see page 24 of the instructions)

Complete this part and attach Schedule 1 to Form 1040A if you received over $400 in dividends.

1 List name of payer		Amount	
	1		
2 Add amounts on line 1. Enter the total here and on Form 1040A, line 9.	2		

Income tax return form

- 

Department of the Treasury - Internal Revenue Service

Form 1040EZ **Income Tax Return for Single Filers With No Dependents** 1989

(Interrogational part)

Name & address

Use the IRS mailing label. If you don't have one, please print.

Please print your numbers like this: 9 8 7 6 5 4 3 2 1 0

Your social security number

(Descriptive part-1)

Instructions are on the back. Also, see the Form 1040A/1040EZ booklet, especially the checklist on page 14.

Presidential Election Campaign Fund

Do you want $1 to go to this fund?

(Extensional part)

Report your income

1 Total wages, salaries, and tips. This should be shown in Box 10 of your W-2 form(s). (Attach your W-2 form(s).) 1

Attach Copy B of Form(s) W-2 here.

2 Taxable interest income of $400 or less. If the total is more than $400, you cannot use Form 1040EZ. 2

3 Add line 1 and line 2. This is your **adjusted gross income**. 3

4 Can your parents (or someone else) claim you on their return?

Note: You must check Yes or No.

☐ **Yes.** Do worksheet on back; enter amount from line E here.

☐ **No.** Enter 5,100. This is the total of your standard deduction and personal exemption. 4

5 Subtract line 4 from line 3. If line 4 is larger than line 3, enter 0. This is your **taxable income**. 5

Figure your tax

6 Enter your Federal income tax withheld from Box 9 of your W-2 form(s). 6

7 **Tax.** Use the amount on **line 5** to look up your tax in the tax table on pages 41-46 of the Form 1040A/1040EZ booklet. Use the **single** column in the table. Enter the tax from the table on this line. 7

Refund or amount you owe

8 If line 6 is larger than line 7, subtract line 7 from line 6. This is your **refund**. 8

9 If line 7 is larger than line 6, subtract line 6 from line 7. This is the **amount you owe**. Attach check or money order for the full amount, payable to "Internal Revenue Service." 9

Attach tax payment here.

Sign your return

I have read this return. Under penalties of perjury, I declare that to the best of my knowledge and belief, the return is true, correct, and complete.

(Keep a copy of this form for your records.)

Your signature Date

X

(Certificating part)

For Privacy Act and Paperwork Reduction Act Notice, see page 3 in the booklet.

Form 1040EZ (1989)

(Descriptive part-2)

Department of the Treasury - Internal Revenue Service

Form **1040EZ** **Income Tax Return for Single Filers With No Dependents** (0) **1989**

Intensional part

Name & address

Use the IRS mailing label. If you don't have one, please print.

Print your name above (first, initial, last)

Home address (number and street). (If you have a P.O. box, see back.) Apt. no.

City, town or post office, state, and ZIP code

Please print your numbers like this:

9 8 7 6 5 4 3 2 1 0

Your social security number

Descriptive part-1

Instructions are on the back. Also, see the Form 1040A/1040EZ booklet, especially the checklist on page 14.

Extensional part

Presidential Election Campaign Fund

Do you want $1 to go to this fund?

Note: Checking "Yes" will not change your tax or reduce your refund. ▶

Report your income

1 Total wages, salaries, and tips. This should be shown in Box 10 of your W-2 form(s). (Attach your W-2 form(s).) 1

Attach Copy B of Form(s) W-2 here.

2 Taxable interest income of $400 or less. If the total is more than $400, you cannot use Form 1040EZ. 2

	3	Add line 1 and line 2. This is your **adjusted gross income.**	3
Note: You ***must*** *check Yes or No.*	4	Can your parents (or someone else) claim you on their return? ☐ **Yes.** Do worksheet on back; enter amount from line E here. ☐ **No.** Enter 5,100. This is the total of your standard deduction and personal exemption.	4
	5	Subtract line 4 from line 3. If line 4 is larger than line 3, enter 0. This is your **taxable income.**	5
Figure your tax	6	Enter your Federal income tax withheld from Box 9 of your W-2 form(s).	6
	7	**Tax.** Use the amount on **line 5** to look up your tax in the tax table on pages 41-46 of the Form 1040A/1040EZ booklet. Use the **single** column in the table. Enter the tax from the table on this line.	7
Refund or amount you owe	8	If line 6 is larger than line 7, subtract line 7 from line 6. This is your **refund.**	8
Attach tax payment here.	9	If line 7 is larger than line 6, subtract line 6 from line 7. This is the **amount you owe.** Attach check or money order for the full amount, payable to "Internal Revenue Service."	9

Sign your return

(Keep a copy of this form for your records.)

I have read this return. Under penalties of perjury, I declare that to the best of my knowledge and belief, the return is true, correct, and complete.

Your signature   Date

X

Certificating part

For Privacy Act and Paperwork Reduction Act Notice, see page 3 in the booklet.

Form 1040EZ (1989)

Descriptive part-2

View design from forms

1. Requirements analysis

   1.1 Distinguish extensional, intensional, and descriptive parts of the form
   1.2 Select areas and subareas

2. Initial design

   2.1 Build a global skeleton schema

3. Schema design—for each area:

   3.1 Build the area schema
   3.2 Merge the area schema with the skeleton schema

Form analysis

•

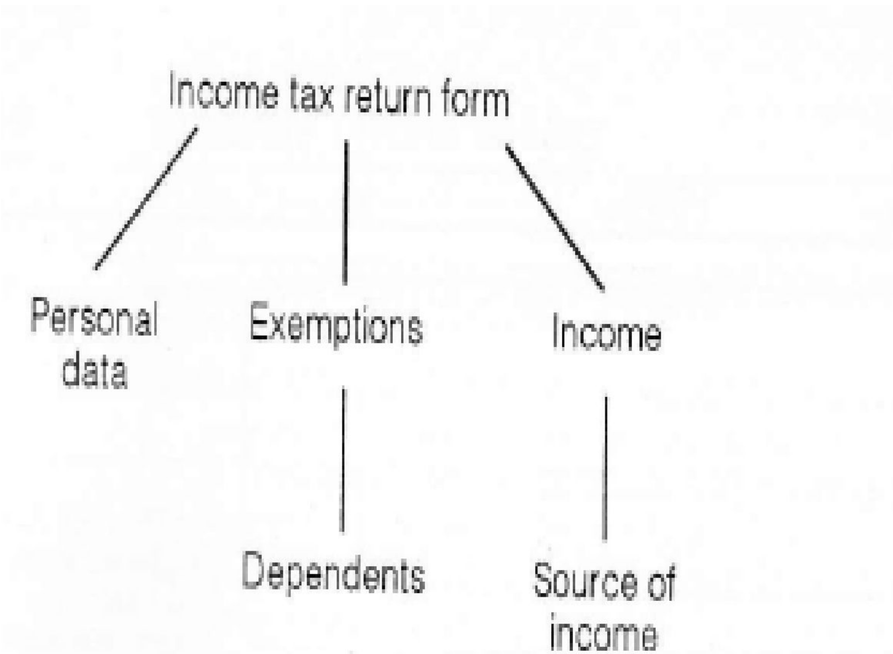
The first goal of form analysis is to understand the structure and meaning of the form; to this end, it is useful to distinguish its extensional, intensional, and descriptive parts. Additional information about the structure of forms is obtained by subdividing forms into areas. Since forms are used to facilitate information exchange, the placement of fields in forms is generally well studied, and homogeneous information is contiguous. An *area* is simply a portion of the form that deals with data items closely related to one another.

Form Analysis for Income tax return form

Let us consider a portion of the 1989 U.S. Individual Income Tax Return Form 1040A, shown in Figure 4.11. We distinguish three areas: area 1 concerns personal data, area 2 concerns exemptions, and area 3 concerns income evaluation. Areas may be further divided into subareas. In area 2, we detect a subarea about dependents of the filer, and in area 3 we detect a subarea about sources of income. As a general rule, designers prefer to use area decompositions that partition each form into pieces of similar complexity; the same applies to the decomposition of areas into subareas. Thus, areas and subareas of forms

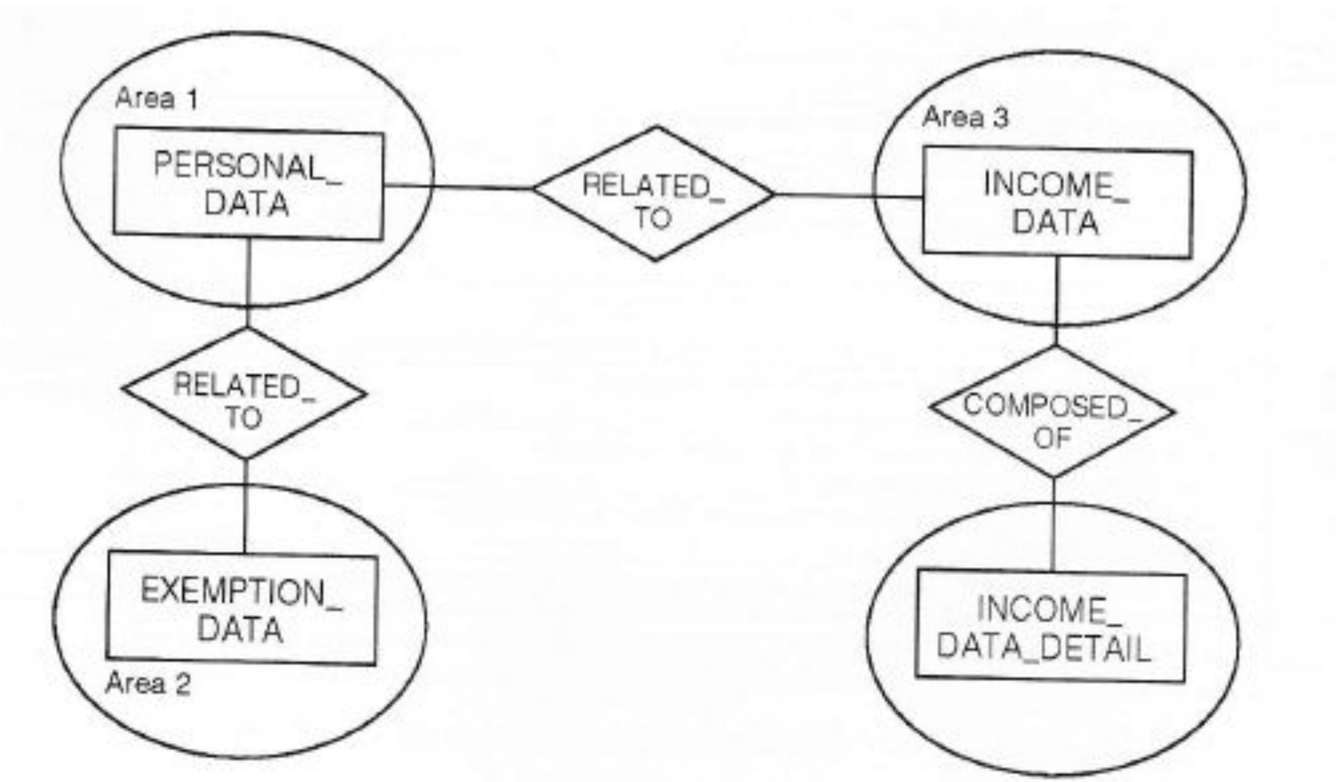
become good candidates for decomposing design activity

Tree of areas of Income tax return form



Skeleton schema for income tax return form

-



Schema Design

- Parametric Text. A parametric text is the text in the natural language with some empty fields that are to be filled with values taken from subtable domains/ This text is completed by additional indications

  about the values that are to be entered in the fields; both the text and the additional indications constitute the intensional part. When the fields are filled, the text becomes complete and coherent. An example of parametric text follows:

  We certify that ________________ (First name) ________________ (Last name), born in ________________ (City/Town)

  on ___ / ___ / 1__ , served in the army from ___ / ___ / 19___ to ___ / ___ / 19___ as

  a(n) ____________________ (Officer, Soldier)

Notice the different types of captions that are used in parametric text to express the properties of data. In the first line of text, *First name* and *Last name* are *unique names* of the concepts in the form. In the same line, City and *Town* are the two *possible names* of the corresponding concept. Finally, in the last line, the list *Officer*, *Soldier* indicates the possible *values* for the corresponding concept. The corresponding ER schema should contain four attributes: LAST_NAME, FIRST_NAME, BIRTH_CITY, and MILITARY_RANK.

Structured text such as *from* __ / __ / 19__ *to* __ / __ / 19__ indicates explicitly the existence of two attributes, START_MILITARY_SERVICE and END_MILITARY_SERVICE and also gives information about the structure of data (e.g., 6 bytes required) that will be useful in subsequent phases of the design.

Lists. In a list, all possible values of a concept are exhaustively presented; some of them are selected (e.g., checked) when the form is completed. Figure 4.14 shows a list from the Income Tax Return Form 1040A.

Step 2 Check your filing status (Check only one.)	1 ☐	Single (See if you can use Form 1040EZ.)
	2 ☐	Married filing joint return (even if only one had income)
	3 ☐	Married filing separate return. Enter spouse's social security number above and spouse's full name here. ____________
	4 ☐	Head of household (with qualifying person). (See page 16.) If the qualifying person is your child but not your dependent, enter this child's name here. ____________
	5 ☐	Qualifying widow(er) with dependent child (year spouse died ➤19___).

Figure 4.14 Example of a list

When a list is translated into ER concepts, it is important to understand whether the alternatives presented to the user are *exclusive*; in this case, a single attribute with a single value can be used to represent the list of alternatives. If instead *multiple choices* are possible, then it is necessary to introduce either a single attribute with multiple values, or one attribute for each choice (of type Boolean). In the example of Figure 4.14, choices are exclusive; therefore the single attribute FILING_STATUS is introduced.

Tables. Tables are conveniently modeled by introducing specific entities having two sets of attributes: the *identifiers* and the *values*. Identifiers uniquely select each position of the table, while the values correspond to the table's content. In the Income Tax Return Form 1040A, Part II and Part III present one table each, the interest income and dividend income tables. In this case, we refine the entity INCOME_DATA_DETAIL, present in the skeleton schema, into two separate entities INTEREST_DATA_DETAIL and DIVIDEND_DATA_DETAIL, each corresponding to one of the tables; the identifier for each row of the table is given by the combination of the filer's Social Security number and the row number. The value attributes are PAYER_NAME and AMOUNT.

Tables can be more complex (multidimensional arrays); for instance, Figure 4.15 shows an example of a three-dimensional array that represents the expenses of a company for a three-year period. Each expense refers to one year, a month in the year, and a period in the month. With multidimensional arrays, the number of attributes required for identification is larger. In this example, the identification attributes are COMPANY_IDENTIFIER, YEAR, MONTH, and PERIOD; the value attribute is EXPENSE.

Expenses for the last three years

	Year	Month											
	1988	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Period	1–15												
	16–31												
	1989	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Period	1–15												
	16–31												
	1990	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Period	1–15												
	16–31												

Figure 4.15 Example of multidimensional table

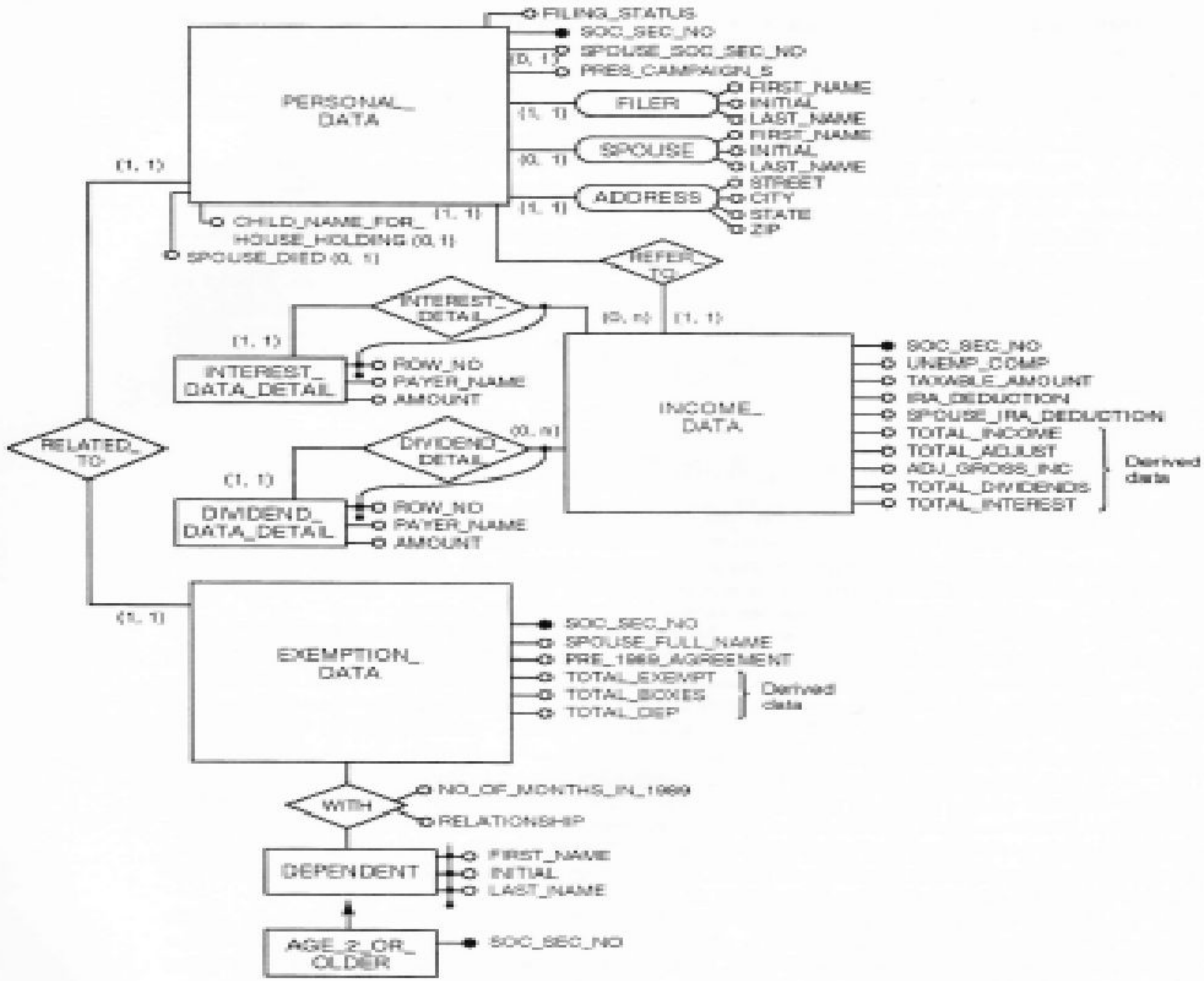
Optional Parts of the Form. A field of the form is optional when it can either be filled or left empty, depending on rules that usually appear explicitly but are sometimes implicit. For instance, consider the sentence on the top of Part III of the tax return form (Figure 4.11): *Complete this part and attach Schedule 1 to Form 1040A if you received over $400 in dividends*. This sentence indicates that the entire DIVIDEND_INCOME table may be left empty. This optionality is translated in the ER model by setting to 0 the min-card of the entity INCOME_DATA in the relationship DIVIDEND_DETAIL.

Most optionalities refer to attributes. Consider again the portion of the form in Figure 4.14 and notice that choices 3 and 4 require filling some empty space. This corresponds to introducing some additional attributes: SPOUSE_SOC_SEC_NO, CHILD_NAME_FOR_HOUSEHOLDING, and the composite attribute SPOUSE_FULL_NAME. However, since the filling of spaces is required only in the case of specific choices, these attributes are certainly optional (e.g., with min-card of 0).

Derived Data. A field contains derived data when its value can be computed from other data in the form. The tax return form contains many examples of derived data: for example, the number of boxes checked in the exemption area can be derived from the individual boxes. Likewise, fields 11, 12c, and 13 of the income area correspond to computed data. Finally, the total of amounts in the tables for interest income and dividend income can be derived as the summation of individual entries.

It is important to note that derived data should not necessarily be stored in the database, because it can be computed by a program. However, recomputation may be expensive; thus, in some applications, derived data is in fact stored within database records. Perhaps the most reasonable thing to do at the conceptual level is to include derived data and indicate clearly how the data items can be computed.

Figure 4.16a shows the final conceptual schema. All details of the schema should be carefully considered, particularly the cardinality values of attributes and composite attributes. Notice that we model dependents in the exemption subareas as an entity, since several properties (attributes, relationships, subentities) can be associated with dependents. Figure 4.16b indicates the rules that are used to compute derived data.



(a) Final conceptual schema

Figure 4.16 Schema for Income Tax Return Form

ATTRIBUTE	DERIVATION
TOTAL_INCOME	TOTAL_WAGES + INTEREST_INCOME + DIVIDEND_INCOME + UNEMP_COMP
TOTAL_ADJUST	IRA_DEDUCTION + SPOUSE_IRA_DEDUCTION
ADJ_GROSS_INC	TOTAL_INCOME – TOTAL_ADJUST
TOTAL_DIVIDENDS	summation of AMOUNT in INTEREST_DATA_DETAIL connected by the INTEREST_DETAIL relationship
TOTAL_INTEREST	summation of AMOUNT in DIVIDEND_DATA_DETAIL connected by the DIVIDEND_DETAIL relationship
TOTAL_BOXES	1 + cardinality of SPOUSE_SOC_SEC_NO
TOTAL_DEP	cardinality of DEPENDENT
TOTAL_EXEMPT	TOTAL_BOXES + TOTAL_DEP

(b) Derived data in the conceptual schema

Figure 4.16 (cont'd) Schema for Income Tax Return Form

View Designs Starting from record formats

Commercial applications implemented on computers invariably use *files*, that is, collections of records stored in secondary memory. Each record consists of a group of fields; fields may in turn be composed of subfields. As a consequence, records usually have a hierarchical structure, and each field is placed at a given level in the hierarchy. The most common languages used to write applications are COBOL, PL/1, FORTRAN, and C.

The structure of files is declared in the programs that use them: for instance, COBOL files are declared in a particular portion of COBOL programs, called the DATA DIVISION. Figure 4.17 shows the declaration in COBOL of an ORDER file. Each record corresponds to an order. Some fields (e.g., PART-CODE, UNIT-OF-MEASURE, QUANTITY) correspond to atomic pieces of information (elementary fields); other fields (e.g., VALUE, DATE-OF-ISSUE) are in turn structured into subfields (compound fields).

```
01 ORDER.

   02 ORDER-NUMBER                    PIC X(10).

   02 DATE-OF-ISSUE.
      03 YEAR-OF-ISSUE                PIC 9(2).
      03 MONTH-OF-ISSUE               PIC 9(2).
      03 DAY-OF-ISSUE                 PIC 9(2).

   02 DATE-OF-DELIVERY.
      03 YEAR-OF-DELIVERY             PIC 9(2).
      03 MONTH-OF-DELIVERY            PIC 9(2).
      03 DAY-OF-DELIVERY              PIC 9(2).

   02 VALUE.
      03 PRICE                        PIC 9(6)V99.
      03 CURRENCY-CODE                PIC X(2).
      03 CHANGE-RATE                  PIC 9(6)V99.

   02 ORDER-LINE OCCURS 10 TIMES.

      03 PART-CODE                    PIC 9(6).
      03 LINE-KEY                     PIC 9(3).
      03 UNIT-OF-MEASURE              PIC X(2).
      03 QUANTITY                     PIC 9(6) COMPUTATIONAL.

   02 STORE-CODE                      PIC X(3).
   02 SUPPLIER-CODE                   PIC X(4).
   02 CLIENT                          PIC X(15).
   02 FACTORY                         PIC X(2).
```

Figure 4.17 COBOL description of an ORDER file

In COBOL, as in other languages that deal with files, several clauses of the file definition specify the role of the field, its storage allocation, the type of accesses provided to the file, and other features. This information is of great importance in determining the meanings of fields, their inner logical relationships, and the abstractions defined among them so that we can represent the file in terms of an ER schema. Application programs that do not use a DBMS typically repeat the file definition in their initial parts.

In the design of ER schemas from record formats, we start by introducing a single entity to represent the file and give it the same name as the file. This choice is quite natural, since a file is a collection of data with the same structure. We then consider parts (clauses) of the file definition in order to deduce additional structural properties of the file. Hence, the initial simple representation of the file is progressively enriched by introducing new entities, relationships, generalizations, attributes, and so on. In this section we examine some of the clauses that can appear in a file definition and give general guidelines for their translation into features of the ER model. To make the ideas concrete, we will use the terminology of the COBOL language.

Simple Fields

A field is simple when it has a single occurrence in each record instance and is subscripted (or repetitive) otherwise. Simple fields can be elementary or compound. Simple elementary fields are translated into simple attributes of the ER model; compound fields are translated into compound attributes. Consider the record format of Figure 4.17. The following lines are translated into simple attributes of the ORDER entity.

```
02 CLIENT              PIC X(15).
02 FACTORY             PIC X(2).
```

The following lines are translated into a compound attribute of the ORDER entity.

```
02 DATE-OF-ISSUE.
   03 YEAR-OF-ISSUE    PIC 9(2).
   03 MONTH-OF-ISSUE   PIC 9(2).
   03 DAY-OF-ISSUE     PIC 9(2).
```

•

```
01  ORDER.
    02  ORDER-NUMBER                PIC X(10).
    02  DATE-OF-ISSUE.
        03  YEAR-OF-ISSUE           PIC 9(2).
        03  MONTH-OF-ISSUE          PIC 9(2).
        03  DAY-OF-ISSUE            PIC 9(2).
    02  DATE-OF-DELIVERY.
        03  YEAR-OF-DELIVERY        PIC 9(2).
        03  MONTH-OF-DELIVERY       PIC 9(2).
        03  DAY-OF-DELIVERY         PIC 9(2).
    02  VALUE.
        03  PRICE                   PIC 9(6)V99.
        03  CURRENCY-CODE           PIC X(2).
        03  CHANGE-RATE             PIC 9(6)V99.
    02  ORDER-LINE OCCURS 10 TIMES.
        03  PART-CODE               PIC 9(6).
        03  LINE-KEY                PIC 9(3).
        03  UNIT-OF-MEASURE         PIC X(2).
        03  QUANTITY                PIC 9(6) COMPUTATIONAL.
    02  STORE-CODE                  PIC X(3).
    02  SUPPLIER-CODE               PIC X(4).
    02  CLIENT                      PIC X(15).
    02  FACTORY                     PIC X(2).
```

Figure 4.17 COBOL description of an ORDER file

Subscripted (repetitive) fields

- COBOL subscripted fields have multiple occurrences, and each occurrence is identified by a progressive number. In COBOL, subscripted fields are defined in an OCCURS clause, which specifies the number of occurrences of the field in each record instance. In Figure 4.17 ORDER-LINE is repetitive and occurs 10 times in an ORDER record. Subscripted fields with a single OCCURS clause are translated into a single attribute, with both min-card and max-card set to the value of the OCCURS clause.

  A data structure frequently used in COBOL is the table, or n-dimensional array. Arrays with n dimensions are expressed in COBOL by using n subordinate OCCURS clauses. Records with more than one subordinate OCCURS clause are best represented by introducing a new entity. Consider the table in Figure 4.18a, which shows the quantity on hand of a product, classified by month and year. This table is described in COBOL as a subscripted field defined using two instances of the OCCURS clause, as shown in Figure 4.18b. As already discussed in the previous section, a table is represented in the ER model by introducing a new entity; in our case, we add QUANTITY_ON_HAND, having as identifying attributes PROD_CODE, YEAR, and MONTH, with the value attribute QUANTITY. The entity QUANTITY_ON_HAND is connected to the entity PRODUCT by the relationship AVAILABILITY (Figure 4.18c).

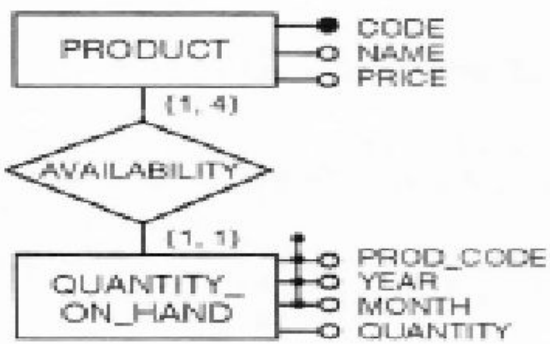
QUANTITY ON HAND OF A PRODUCT

Month	1983	1984	1985	1986
Jan	12	25	27	43
Feb	23	12	43	45
Mar	12	24	26	27
Apr	34	34	25	07
May	33	56	07	77
Jun	55	13	23	33
Jul	66	22	55	69
Aug	34	56	98	34
Sep	48	44	23	11
Oct	77	23	16	17
Nov	89	67	50	23
Dec	07	58	44	18

(a)

```
01 PRODUCT.

   02 NAME     PIC X(20).
   02 CODE     PIC X(4).
   02 PRICE    PIC 9(5).

   02 QUANTITY-ON-HAND-TABLE.
      03 QUANTITY-ON-HAND-BY-YEAR OCCURS 4 TIMES.
         04 QUANTITY-ON-HAND-BY-MONTH PIC 99 OCCURS 12 TIMES.
```

(b)



(c)

Figure 4.18 Table showing the quantity on hand of a product, corresponding record format, and ER schema

Field redefinition

- Field redefinition enables programmers to define the same portion of a record using different field definition clauses. It may be used for two different purposes: (1) to view the same data according to different viewpoints, and (2) to optimize the physical storage space.

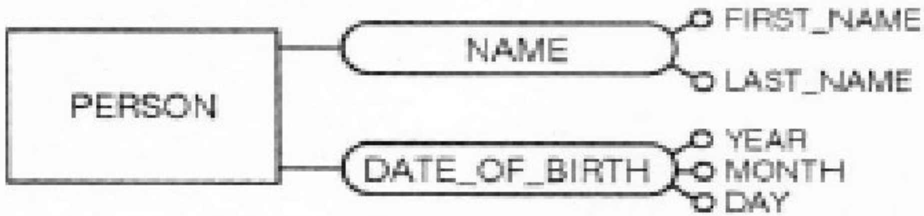
  The first application of the REDEFINES clause is shown in Figure 4.19: the same set of fields is aggregated into two different groups according to their use in procedures. The designer should select the best conceptual representation, which can be either of the two or a combination of them. In this example SEARCH subfields are used by an update procedure that does not distinguish first and last name and does not require day and month of birth; in the conceptual schema it is preferable to have all attributes explicitly mentioned; hence the first alternative is selected.

  The second application of the REDEFINES clause usually indicates the presence of a generalization hierarchy among the concepts described in the file. In Figure 4.20 the field DATA-OF-WORKER is redefined two times into the fields DATA-OF-SECRETARY and DATA-OF-MANAGER. We can translate the file into a schema with the entity EMPLOYEE and a generalization hierarchy with subentities WORKER, SECRETARY, and MANAGER.

•

```
01  PERSON.

    02  PERSONAL-DATA.

        03  NAME.
            04  LAST-NAME        PIC X(20).
            04  FIRST-NAME       PIC X(20).

        03  DATE-OF-BIRTH.
            04  YEAR         PIC 99.
            04  MONTH        PIC 99.
            04  DAY          PIC 99.

    02  PERSONAL-DATA-BIS      REDEFINES PERSONAL-DATA.

        03  SEARCH.

            04  NAME-S       PIC X(40).
            04  YEAR-S       PIC 99.

        03  FILLER           PIC 9(4).
```

(a)



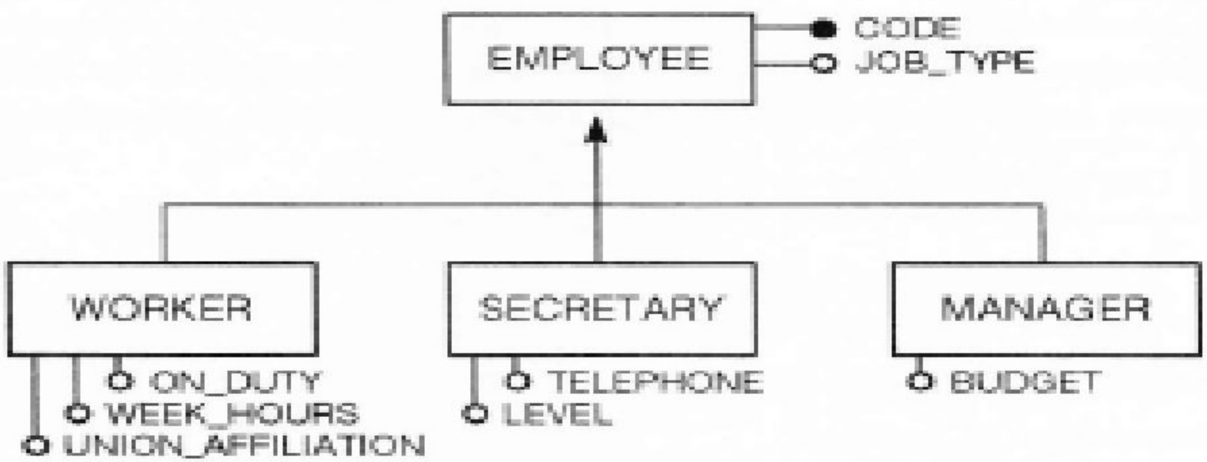
(b)

Figure 4.19 First example of the use of the REDEFINES clause

•

```
01 EMPLOYEE.

   02 CODE          PIC X(7).
   02 JOB-TYPE      PIC X.

   02 DATA-OF-WORKER.
      03 WEEK-HOURS              PIC 99.
      03 ON-DUTY                 PIC X.
      03 UNION-AFFILIATION       PIC X(6).

   02 DATA-OF-SECRETARY REDEFINES DATA-OF-WORKER.
      03 LEVEL   PIC 9.
         04 TELEPHONE   PIC 9(7).

   02 DATA-OF-MANAGER REDEFINES DATA-OF-SECRETARY.
      03 BUDGET   PIC 9(8).
```

(a)



(b)

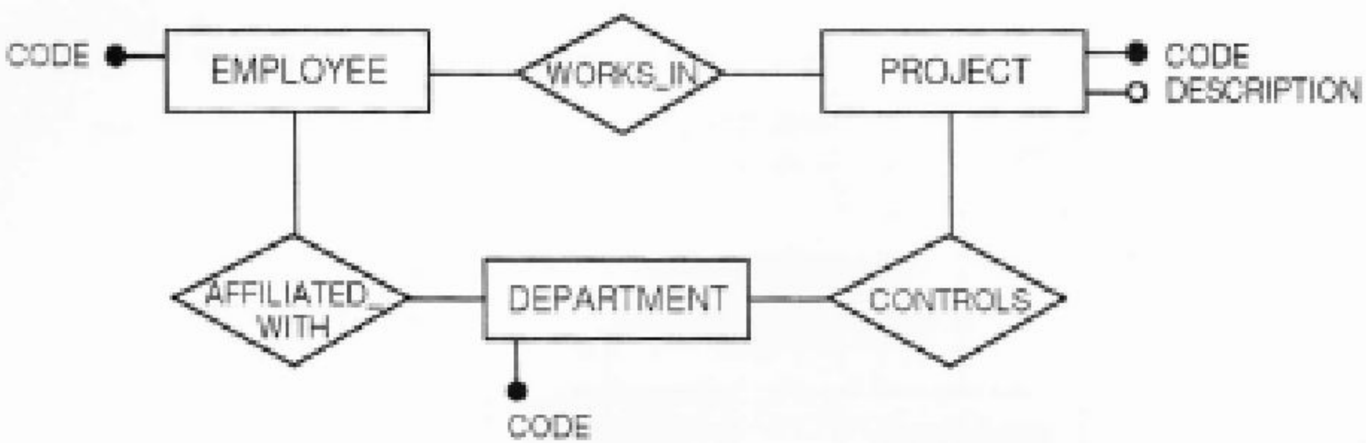
Figure 4.20 Second example of the use of the REDEFINES clause

Symbolic Pointers

A symbolic pointer is a field of a record that denotes the identifier of another record. Symbolic pointers are typically used in COBOL to express logical relationships among files. For instance, in Figure 4.21 three record formats are defined, referring to employees, departments, and projects. The relationships among employees, departments, and projects on which they work are expressed by means of three different fields, which are used as pointers: (1) DEPARTMENT-CODE links employees to their departments, (2) PROJECT-CODE links employees to projects on which they work, and (3) DEPT-CODE links projects to their controlling departments.

```
02 EMPLOYEE.
   03 CODE                  PIC X(10).
   03 DEPARTMENT-CODE       PIC X(5).
   03 PROJECT-CODE          PIC X(7)  OCCURS 10 TIMES.

02 DEPARTMENT.
   03 CODE       PIC X(5).

02 PROJECT.
   03 CODE              PIC X(7).
   03 DEPT_CODE         PIC X(5).
   03 DESCRIPTION       PIC X(30).
```

(a)

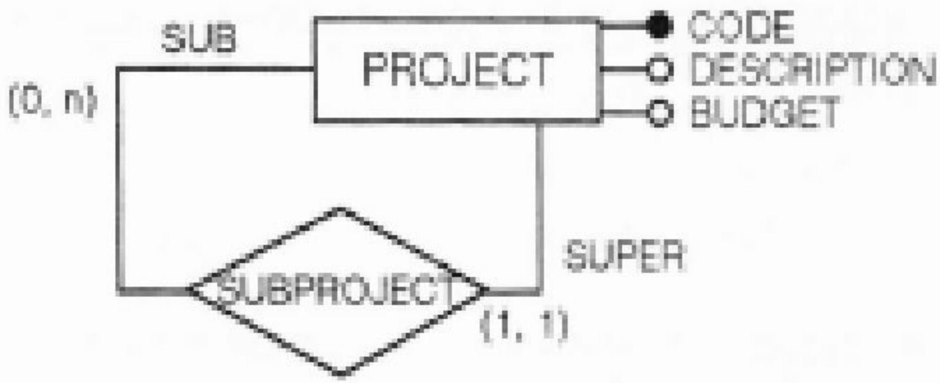


(b)

Figure 4.21 First example of translation of pointers

```
01 PROJECT.

   02 CODE.
   02 DESCRIPTION.
   02 SUPERPROJECT-CODE.
   02 BUDGET.
```

(a)



(b)

Figure 4.22 Second example of translation of pointers

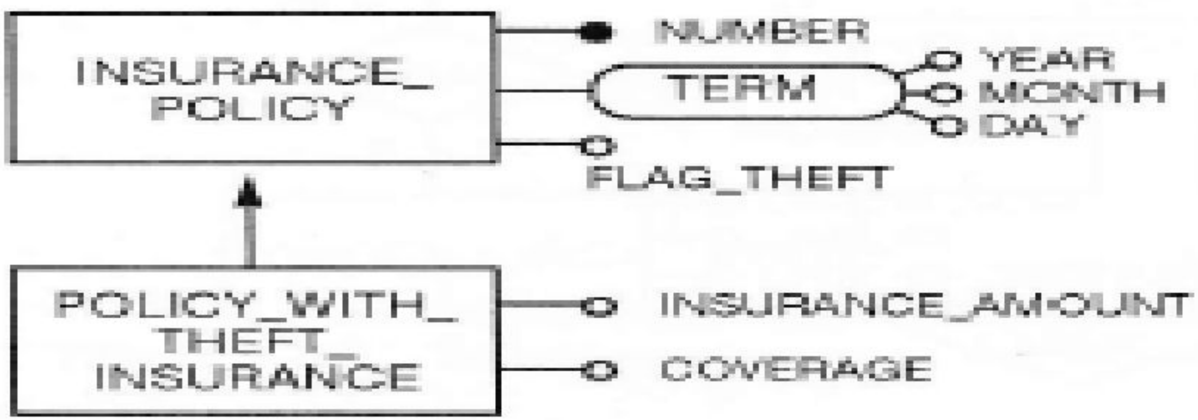
Flags

Flags refer to a field or a group of fields; they are typically used by COBOL programmers to indicate whether subsequent field(s) of a record instance take a value or are left empty. Flags may indicate the presence of a subset between two (or more) different concepts expressed in the file.

As an example, consider in Figure 4.23 the file of insurance policies of a company. Some fields are valid for any type of policy (NUMBER, DATE-OF-TERM, etc.); only some policies include the risk of theft. This property is pointed out by the FLAG_THEFT field: the value is 0 for policies covering theft, 1 otherwise. If the NO_THEFT value is 0, then fields INSURANCE_AMOUNT and COVERAGE should be specified. Note that this is a convention assumed by the programmer; however, no run-time checking is provided in COBOL to ensure it. We can translate the file declaration in two different ways: (1) with a unique entity, in which case the fields referred to by the flag are translated in terms of attributes with optional (0) min-card; and (2) with two entities related by a subset, as shown in Figure 4.23

```
01  INSURANCE-POLICY.

    02  NUMBER        PIC X(10).

    02  DATE-OF-TERM.
        03  YEAR        PIC 9(2).
        03  MONTH       PIC 9(2).
        03  DAY         PIC 9(2).

    02  FLAG-STEALING       PIC 9.
        88  NO-THEFT        VALUE 0.
        88  YES-THEFT       VALUE 1.

    02  INSURANCE-AMOUNT        PIC 9(10).
    02  COVERAGE                PIC 9(10).
```

(a)



(b)

Figure 4.23 Example of translation of a flag

Rules of field values

As an example, we show a file that deals with the bookkeeping of a company. Three levels of accounting are defined: the division, the cost center, and the specific account. Accounts are aggregated by cost centers, and cost centers are aggregated by divisions. Hence, the field COST-CENTER (indicating the code of a cost center) is only meaningful in account records, and the field DIVISION (indicating the code of a division) is only meaningful in cost-center records. The record format of the file is shown in Figure 4.24a. Specific code values establish whether a record instance belongs to one of the three levels. The rules for code values are shown in Figure 4.24b. We can conclude that (1) the file is the result of merging three logically different types of records, hierarchically related, and (2) the fields COST-CENTER and DIVISION may be considered as pointers to other records of the same file.

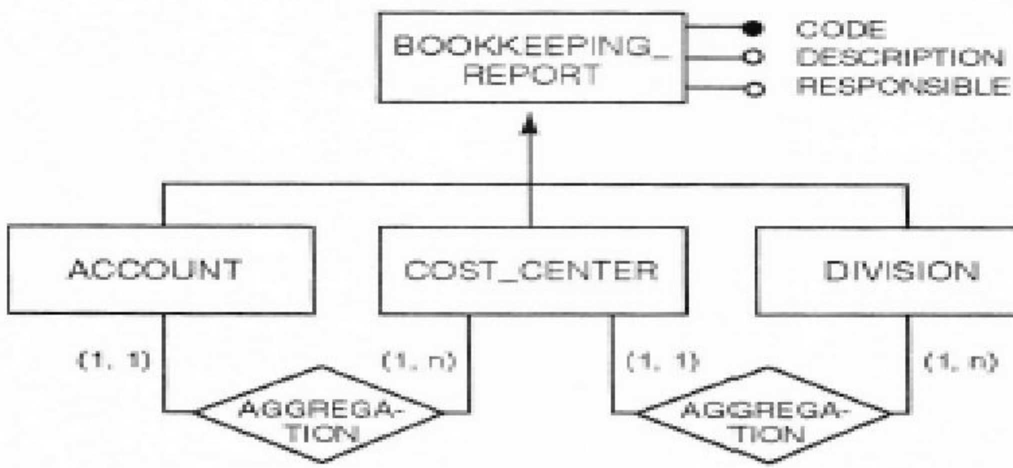
•

```
01 ACCOUNT-REPORT

   02 CODE           PIC 9(4).
   02 DESCRIPTION    PIC X(30).
   02 COST-CENTER    9(3).
   02 DIVISION       9(2).
   02 RESPONSIBLE    X(30).
```

(a) The bookkeeping file of a company

IF THE CODE IS BETWEEN	THEN THE RECORD REFERS TO
1000 and 9999	an account
100 and 999	a cost center
1 and 99	a division

(b) Rules defined for the bookkeeping file



(c) Conceptual schema

Figure 4.24 Example of translation of files with value-dependent rules

•

Starting from this analysis, a possible ER schema for the file definition is shown in igure 4.24c. The BOOKKEEPING_REPORT entity represents the root of a generalization with ubentities ACCOUNT, COST_CENTER, and DIVISION. The two one-to-many relationships be-ween subentities express the hierarchical structure defined among concepts.

We complete this section by showing in Figure 4.25 the ER schema that represents the RDER file introduced in Figure 4.17. The file is translated by introducing two entities, the RDER and the ORDER_LINE, respectively. Note that the two attributes STORE_CODE and UPPLIER_CODE are in fact pointers and could be conveniently represented by relationships vith STORE and SUPPLIER entities.

•

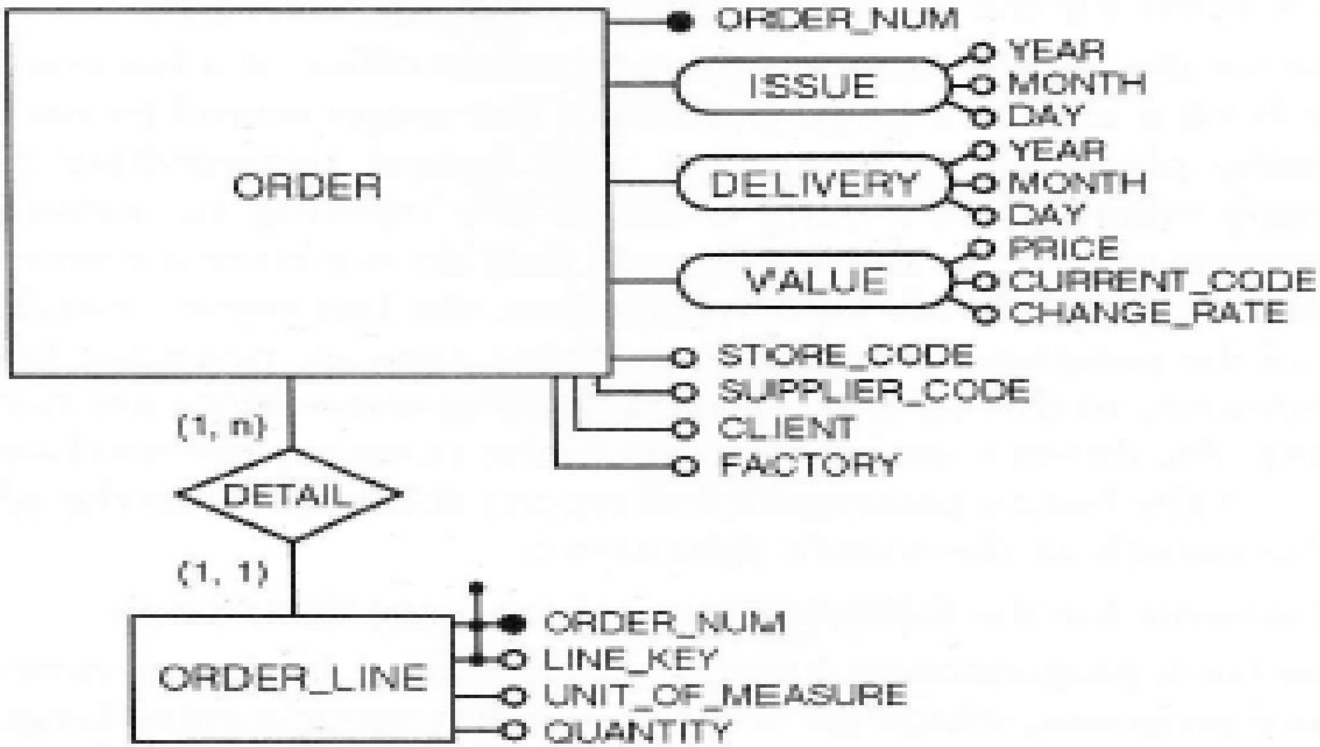


Figure 4.25 ER representation of the ORDER file