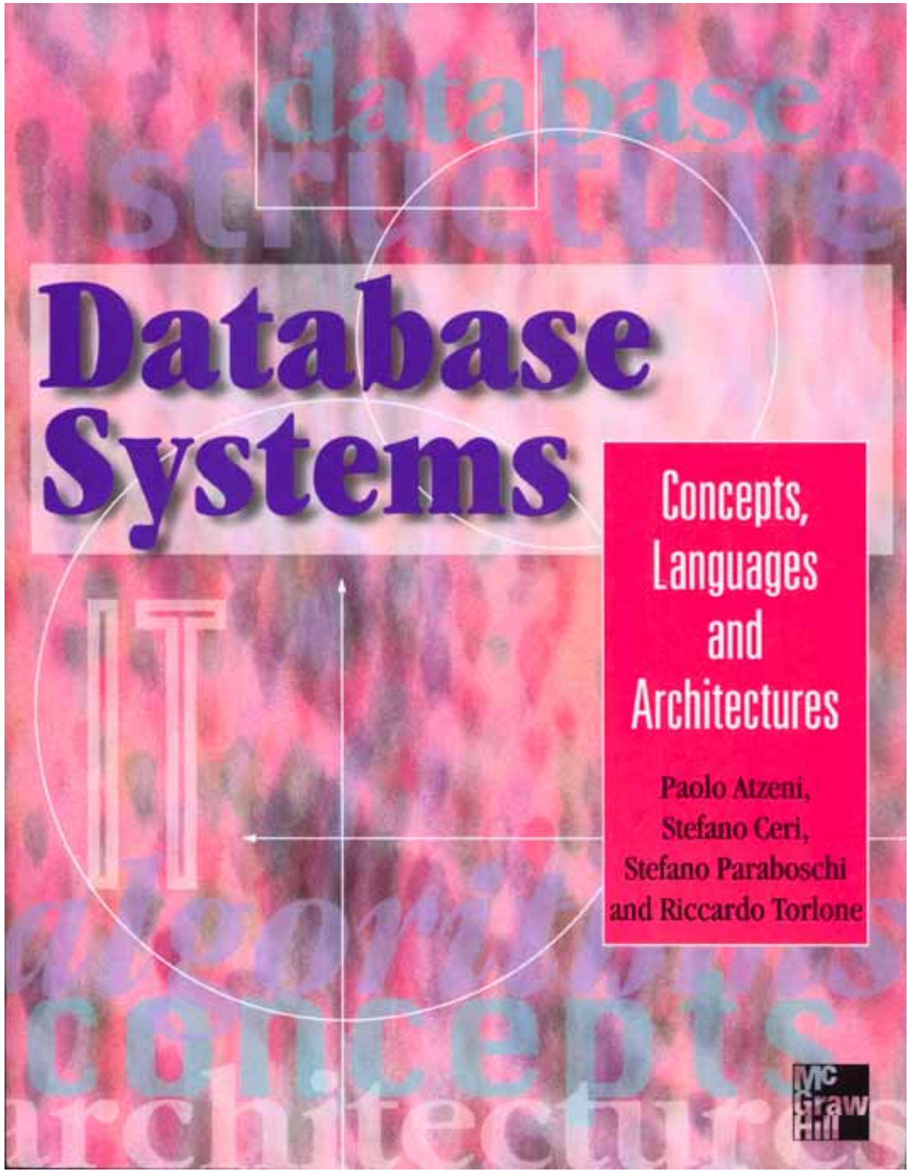

Database Systems
Concepts, Languages and Architectures
Paolo Atzeni, Stefano Ceri, Stefano Paraboschi and Riccardo Torlone
Mc Graw Hill

Chapter 6

Conceptual design

Click here for help

Conceptual design

- The conceptual design of a database consists of the construction of an Entity-Relationship schema, providing an optimal description of the user requirements;
- before we begin to discuss conceptual design, it is worth devoting some attention to the activity that precedes the design process itself: the collection and analysis of the requirements.

Requirements collection and analysis

- The *requirements collection* consists of the complete identification of the problems that the application must solve, and the features that should characterize such an application.
- The *requirements analysis* consists of the clarification and organization of the requirements specification.
- These activities that are closely related to one another.

Main sources of requirements

The requirements generally come from different sources, as follows.

- The users of the application.
- All the existing documentation that has some connection with the problem: forms, internal rules, business procedures, laws and regulations, etc.
- Possible earlier applications that are to be replaced or that must interact in some way with the new application.

Natural language specifications

- The requirements specifications are often written in natural language, at least in the first draft.
- Natural language is, by nature subject to ambiguity and misinterpretation.
- We need to carry out an in-depth analysis of the specification document in order to remove any inaccuracies and ambiguous terms.

Example of requirements in natural language

We wish to create a database for a company that runs training courses. For this, we must store data about the trainees and the instructors. For each course participant (about 5000), identified by a code, we want to store the social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and period employed), the courses attended (there are about 200 courses) and the final assessment of each course. We need also to represent the seminars that each participant is attending at present and, for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants. If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held. For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

Some rules for requirements analysis

- Choose the appropriate level of abstraction.
- Standardize sentence structure.
- Avoid complex phrases.
- Identify synonyms and homonyms, and standardize terms. → POSSONO CREARE AMBIGUITÀ
- Make cross-references explicit.
- Construct a glossary of terms.

An example of a glossary of terms

Term	Description	Synonym	Links
Trainee	Participant in a course. Can be an employee or self-employed.	Participant	Course, Employer
Instructor	Course tutor. Can be freelance.	Tutor	Course
Course	Course offered. Can have various editions.	Seminar	Instructor, Trainee
Employer	Company by which a trainee is employed or has been employed.		Trainee

Rewriting and structuring of requirements (I)

Phrases of a general nature
We wish to create a database for a company that runs training courses. We wish to hold the data for the trainees and the instructors.
Phrases relating to the trainees
For each trainee (about 5000), identified by a code, we will hold the social security number, surname, age, sex, town of birth, current employer, previous employers (along with the start date and the end date of the period employed), the editions of the courses the trainee is attending at present and those he or she has attended in the past, with the final marks out of ten.
Phrases relating to the employers of the trainees
For each employer of a trainee we will hold the name, address and telephone number.

Rewriting and structuring of requirements (II)

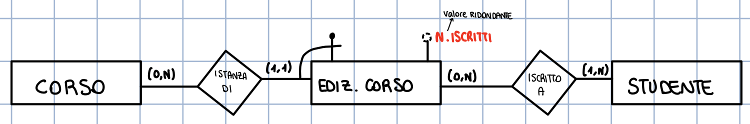
Phrases relating to the courses
For each course (about 200), we will hold the name and code. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we will hold the start date, the end date, and the number of participants. For the editions currently in progress, we will hold the dates, the classrooms and the times in which the classes are held.
Phrases relating to specific types of trainee
For a trainee who is a self-employed professional, we will hold the area of expertise and, if appropriate, the professional title. For a trainee who is an employee, we will hold the level and position held.
Phrases relating to the instructors
For each instructor (about 300), we will hold surname, age, town of birth, all telephone numbers, the edition of courses taught, those taught in the past and the courses the instructor is qualified to teach. The instructors can be permanently employed by the training company or can be freelance.

Example of operational requirements

- **operation 1**: insert a new trainee including all his or her data (to be carried out ~~approximately~~ 40 times a day);
  ↳ il PROF vuole il VALORE MEDIO (le analisi si fanno sui valori medi)
- **operation 2**: assign a trainee to an edition of a course (50 times a day);
- **operation 3**: insert a new instructor, including all his or her data and the courses he or she is qualified to teach (twice a day);
- **operation 4**: assign a qualified instructor to an edition of a course (15 times a day);
- **operation 5**: display all the information on the past editions of a course with title, class timetables and number of trainees (10 times a day);
- **operation 6**: display all the courses offered, with information on the instructors who are qualified to teach them (20 times a day);
- **operation 7**: for each instructor, find the trainees all the courses he or she is teaching or has taught (5 times a week);
- **operation 8**: carry out a statistical analysis of all the trainees with all the information about them, about the editions of courses they have attended and the marks obtained (10 times a month).

ATOMICITÀ: devo fare operaz. atomiche (es. creaz. di un nuovo corso, cose basilari), non essere vaghe

devo fare per forza 1 op. di MODIFICA e 1 op. di LETTURA sul dato ridondante (per fare l'analisi successivamente) esempio:



devi fare almeno 1 OPERAZ di LETTURA (→ conviene averlo) sull'attr. ridondante: Data un'ediz. di corso, visualizza tutti i dati compresi il # di iscritti (senza accedere a STUDENTE)

SCRITTURA (→ non conviene averlo) " " : Dati uno studente (già presente nel DB) ed un altro corso (già presente nel DB) iscrivere lo studente all'ediz. del corso

è statico, non cambia

CORSO	E	20
ED. CORSO	E	200
ISCRITTO A	R	2000
ISTANZA DI	R	200

varia, aumenta linearmente ogni anno

il valore che penso ci sarà in 10 anni, ovvero 10 riedizioni per ogni corso

circa 10 studenti a corso

General criteria for data representation

- If a concept has significant properties and/or describes classes of objects with an autonomous existence, it is appropriate to represent it by an entity.
- If a concept has a simple structure, and has no relevant properties associated with it, it is convenient to represent it by an attribute of another concept to which it refers.
- If the requirements contain a concept that provides a logical link between two (or more) entities, it is convenient to represent this concept by a relationship.
- If one or more concepts are particular cases of another concept, it is convenient to represent them by means of a generalization.

Design strategies for conceptual design

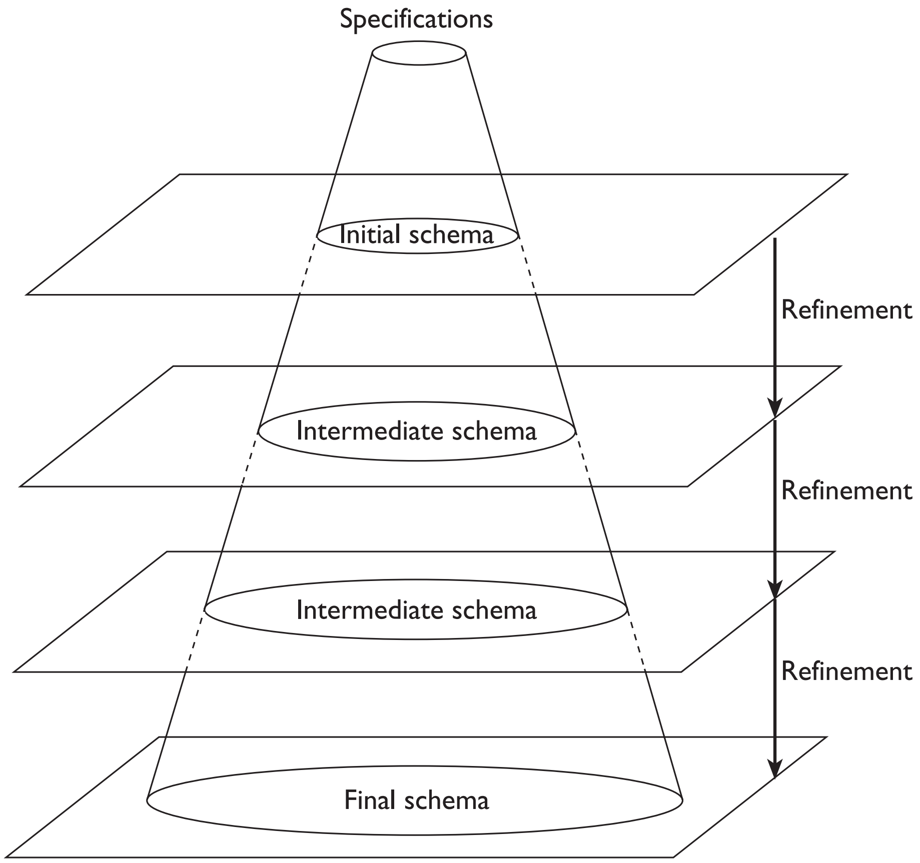
- The development of a conceptual schema based on its specification must be considered to all intents and purposes an engineering process, and, as such, design strategies used in other disciplines can be applied to it.

  – Top-down
  – Bottom-up
  – Inside-out
  – Mixed

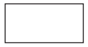
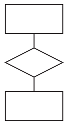
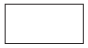
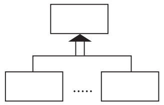
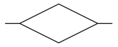
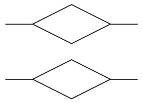
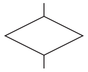
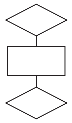
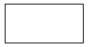
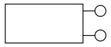
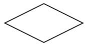
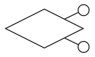
Top-down strategy

- The conceptual schema is produced by means of a series of successive refinements, starting from an initial schema that describes all the requirements by means of a few highly abstract concepts.
- The schema is then gradually expanded by using appropriate modifications that increase the detail of the various concepts.
- Moving from one level to another, the schema is usually modified using some basic transformations called *top-down transformation primitives*.

The top-down strategy



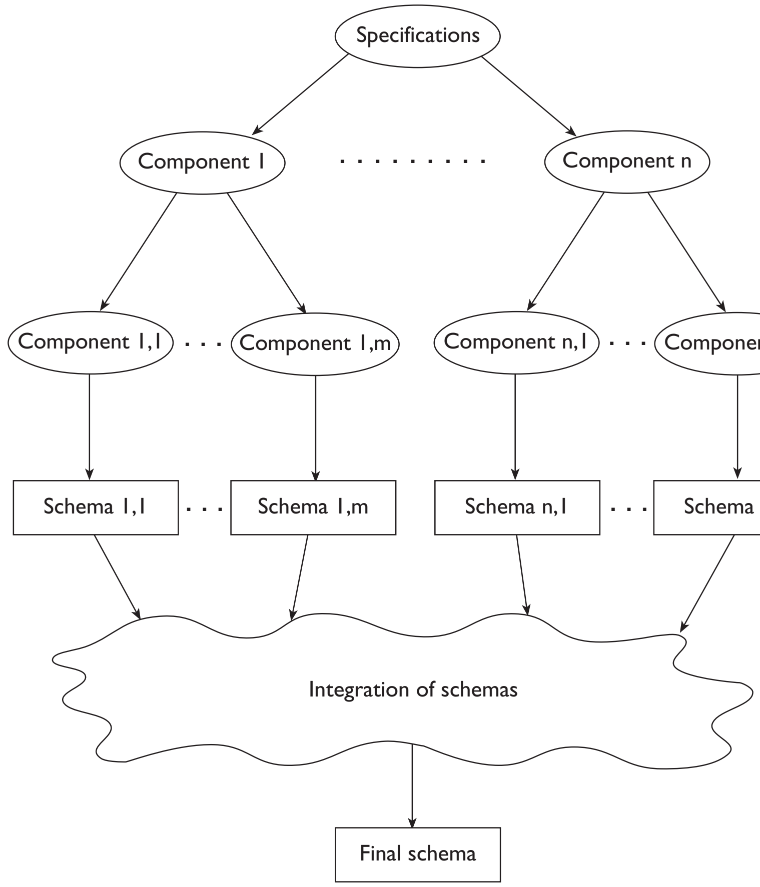
Top-down transformation primitives

Transformation	Initial concept	Result
T_1 From one entity to two entities and a relationship between them		
T_2 From one entity to a generalization		
T_3 From one relationship to multiple relationships		
T_4 From one relationship to an entity with relationships		
T_5 Adding attributes to an entity		
T_6 Adding attributes to a relationship		

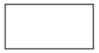
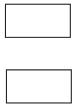
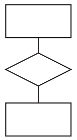
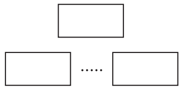
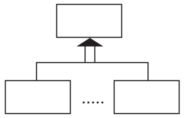
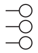
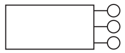
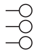
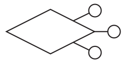
Bottom-up strategy

- The initial specifications are decomposed into smaller and smaller components, until each component describes an elementary fragment of the specifications.
- The various components are then represented by simple conceptual schemas that can also consist of single concepts.
- The various schemas thus obtained are then integrated until a final conceptual schema is reached.
- The final schema is usually obtained by means of some elementary transformations, called *bottom-up transformation primitives*.

The bottom-up strategy



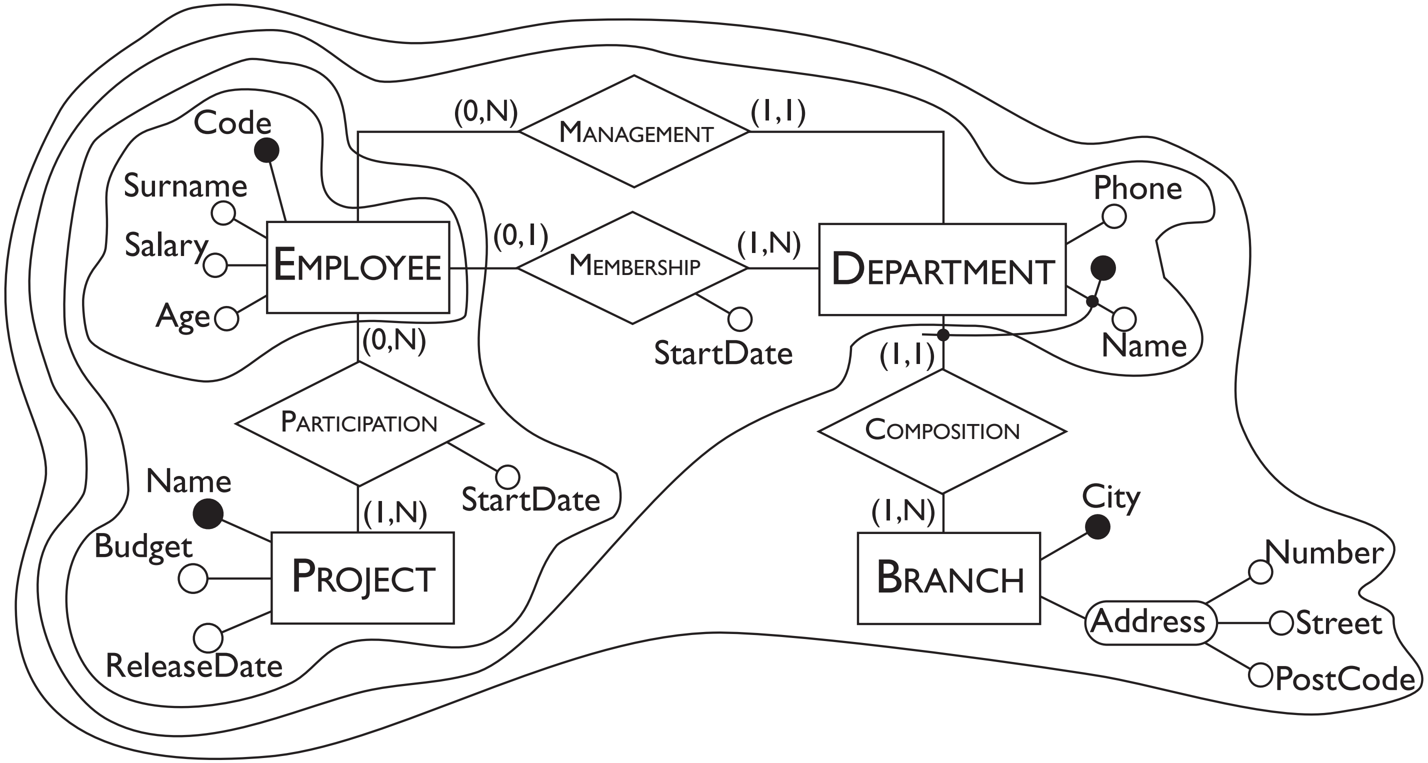
Bottom-up transformation primitives

Transformation	Initial concept	Result
T_1 Generation of an entity		
T_2 Generation of a relationship		
T_3 Generation of a generalization		
T_4 Aggregation of attributes on an entity		
T_5 Aggregation of attributes on a relationship		

Inside-out strategy

- This strategy can be regarded as a particular type of bottom-up strategy.
- It begins with the identification of only a few important concepts and, based on these, the design proceeds, spreading outward ‘radially’.
- First the concepts nearest to the initial concepts are represented, and we then move towards those further away by means of ‘navigation’ through the specification.

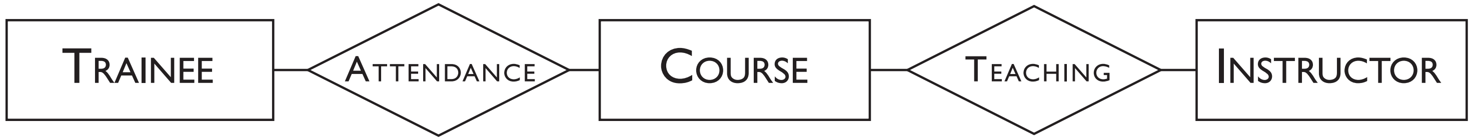
An example of the inside-out strategy



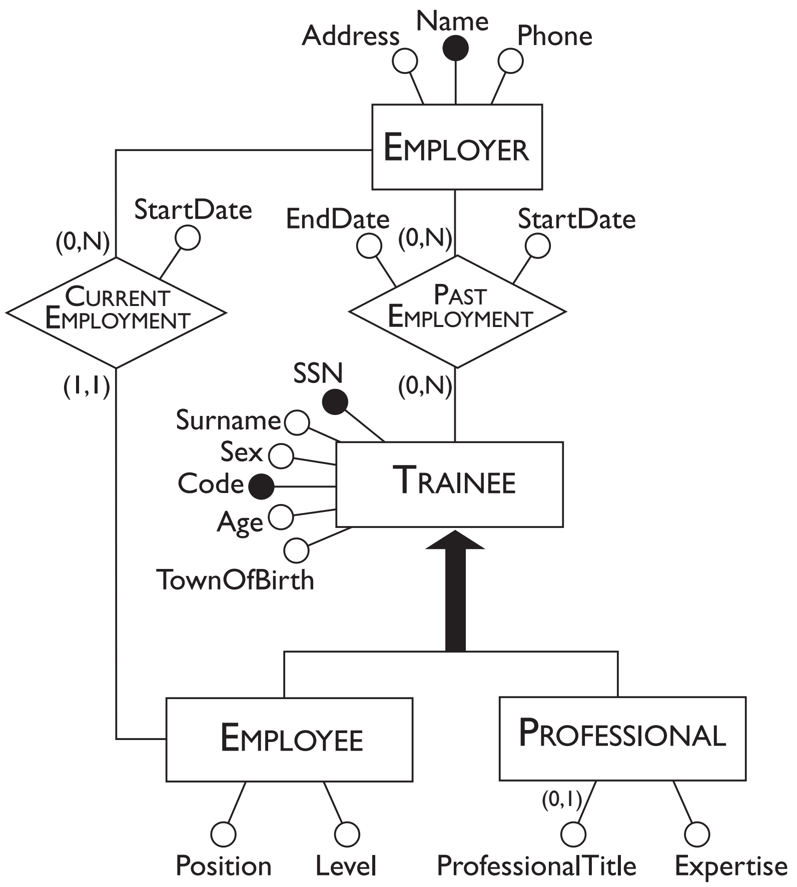
Mixed strategy

- In a mixed strategy the designer decomposes the requirements into a number of components, as in the bottom-up strategy, but not to the extent where all the concepts are separated.
- At the same time he or she defines a *skeleton schema* containing the main concepts of the application. This skeleton schema gives a unified view of the whole design and helps the integration of schemas developed separately.
- Then the designer examines separately these main concepts and can proceed with gradual refinements (following the top-down strategy) or extending a portion with concepts that are not yet represented (following the bottom-up strategy).

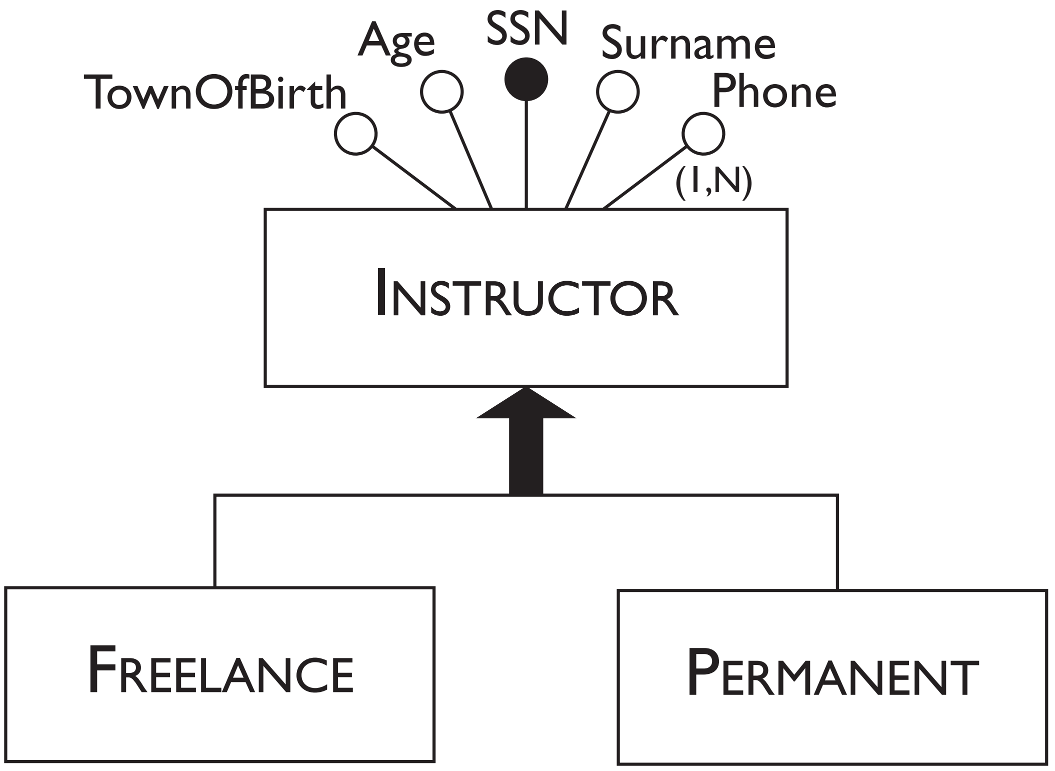
Skeleton schema for a training company



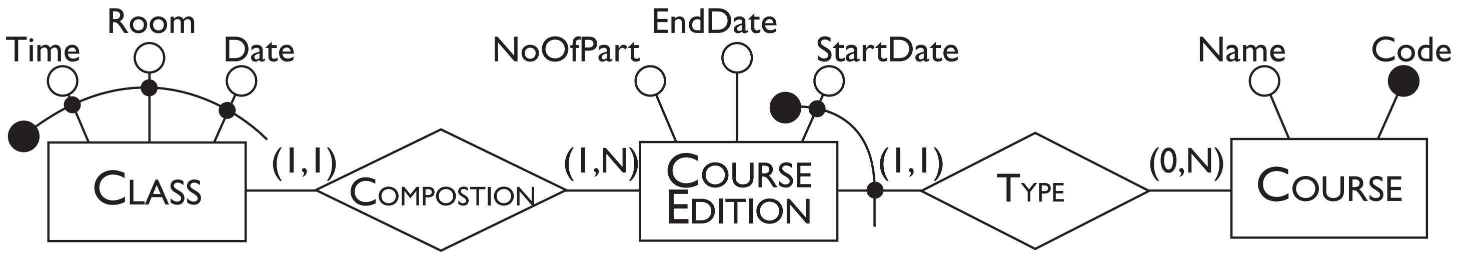
The refinement of a portion of the skeleton schema



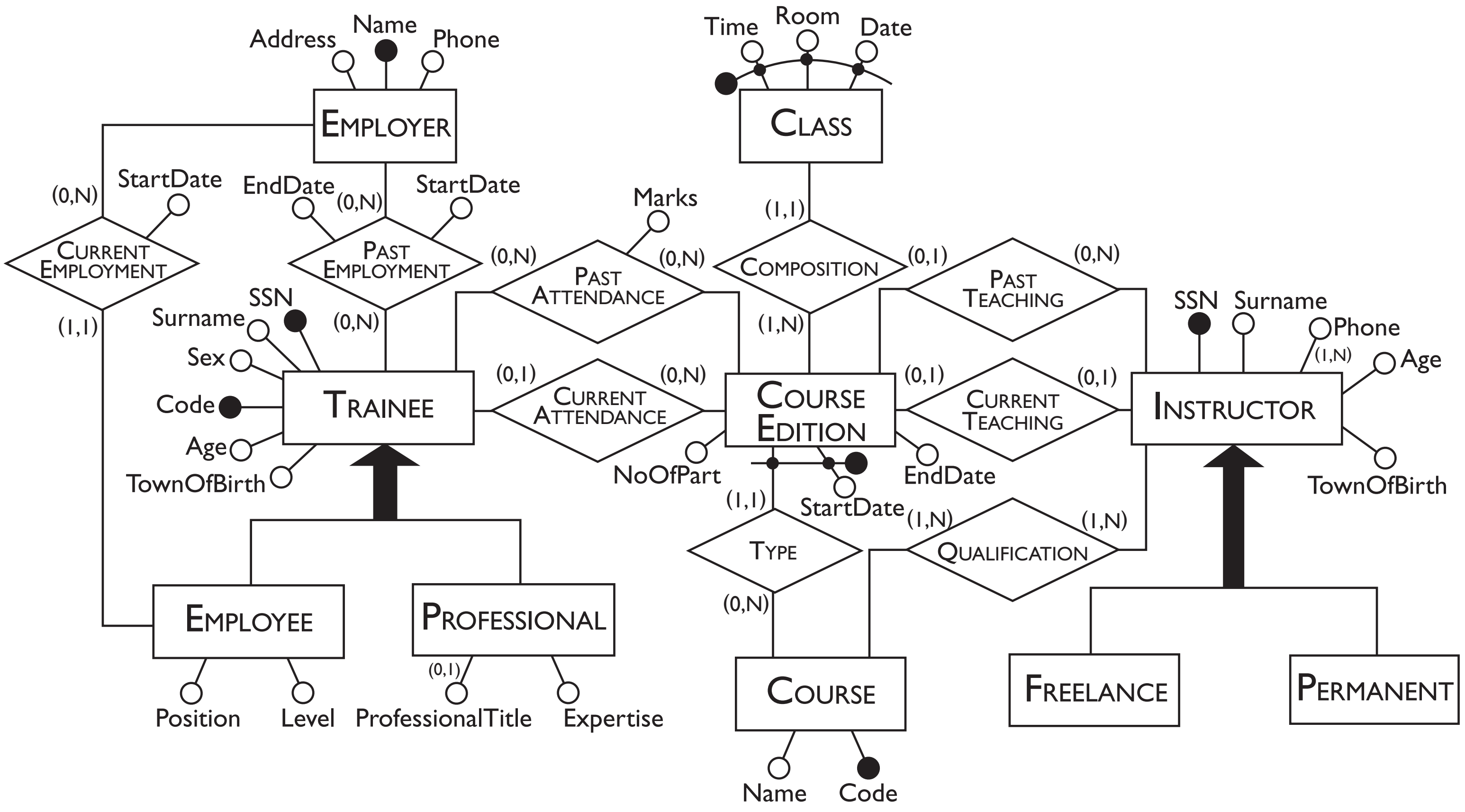
The refinement of another portion of the skeleton schema



The refinement of another portion of the skeleton schema



The final E-R schema for the training company



Quality of a conceptual schema

- Correctness
- Completeness
- Readability
- Minimality

A comprehensive method for conceptual design (I)

1. Analysis of requirements
   (a) Construct a glossary of terms.
   (b) Analyze the requirements and eliminate any ambiguities.
   (c) Arrange the requirements in groups.
2. Basic step
   (a) Identify the most relevant concepts and represent them in a skeleton schema.
3. Decomposition step (to be used if appropriate or necessary).
   (a) Decompose the requirements with reference to the concepts present in the skeleton schema.

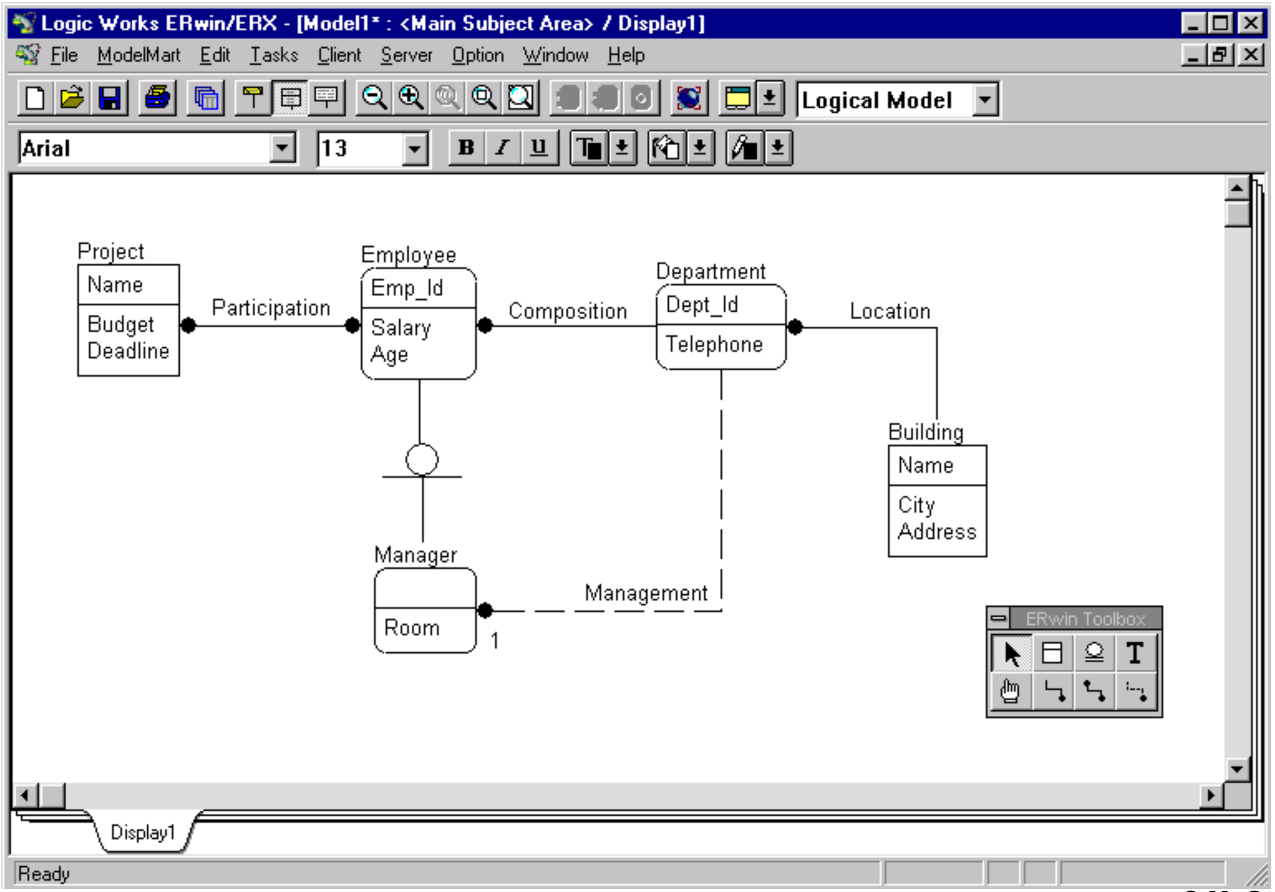
A comprehensive method for conceptual design (II)

4. Iterative step (to be repeated for all the schemas until every specification is represented)
   - (a) Refine the concepts in the schema, based on the requirements.
   - (b) Add new concepts to the schema to describe any parts of the requirements not yet represented.
5. Integration step (to be carried out if step 3 has been used)
   - (a) Integrate the various subschemas into a general schema with reference to the skeleton schema.
6. Quality analysis
   - (a) Verify the correctness of the schema.
   - (b) Verify the completeness of the schema.
   - (c) Verify the minimality of the schema.
   - (d) Verify the readability of the schema.

CASE tools for database design

- There are CASE tools expressly designed for the creation and development of databases.
- These systems provide support for the main phases of the development of a database (conceptual, logical and physical design).
- They also support specific design tasks like the automatic layout of diagrams, correctness and completeness tests, quality analysis, automatic production of DDL code for the creation of a database.

Conceptual design using a CASE tool


Logic Works ERwin/ERX - [Model1* : <Main Subject Area> / Display1]
File ModelMart Edit Tasks Client Server Option Window Help
Logical Model
Arial
13
Project
Name
Budget
Deadline
Participation
Employee
Emp_Id
Salary
Age
Composition
Department
Dept_Id
Telephone
Location
Building
Name
City
Address
Manager
Room
Management
1
ERwin Toolbox
Display1
Ready