Advanced DB

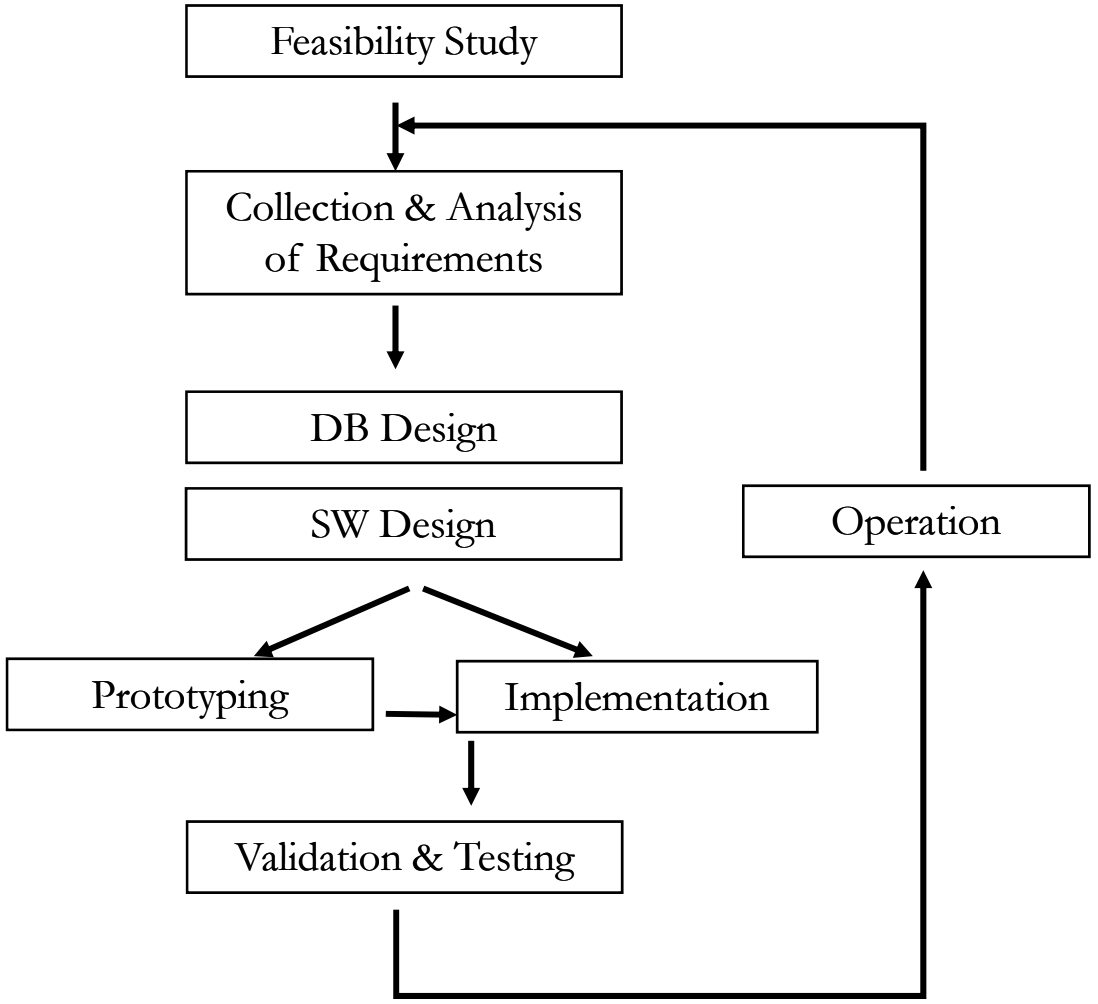
CONCEPTUAL DB DESIGN

Conceptual Database Design

- Entity Relationship Model
- Collection & Analysis of Requirements
- Design Strategies
- Qualities of DB Schema
- The Conceptual Design Process
- Logical DB Design

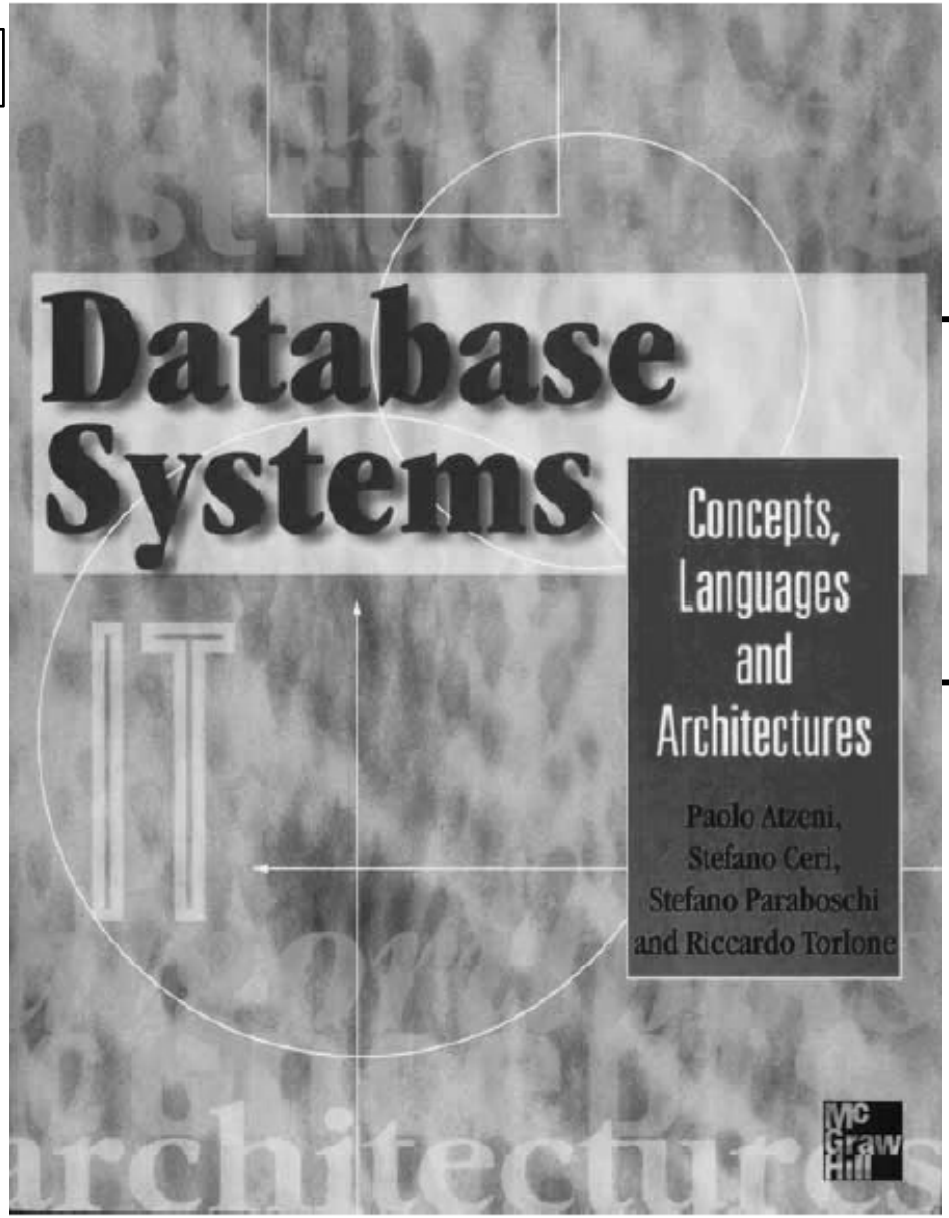
Database Design Objectives

- Decide on the DB schema that
  - is able to hold all information in consideration,
  - with minimal (or no) redundancy, and
  - allows for effective & efficient data operations
- Critical in reducing operations and maintenance costs of SW systems



[SW Development Lifecycle]

Examples from Atzeni, et al, 2000.



Materials and examples from
[Atzeni et al, 2000]
(or [Batini, et al 1992])

Entity-Relationship Model

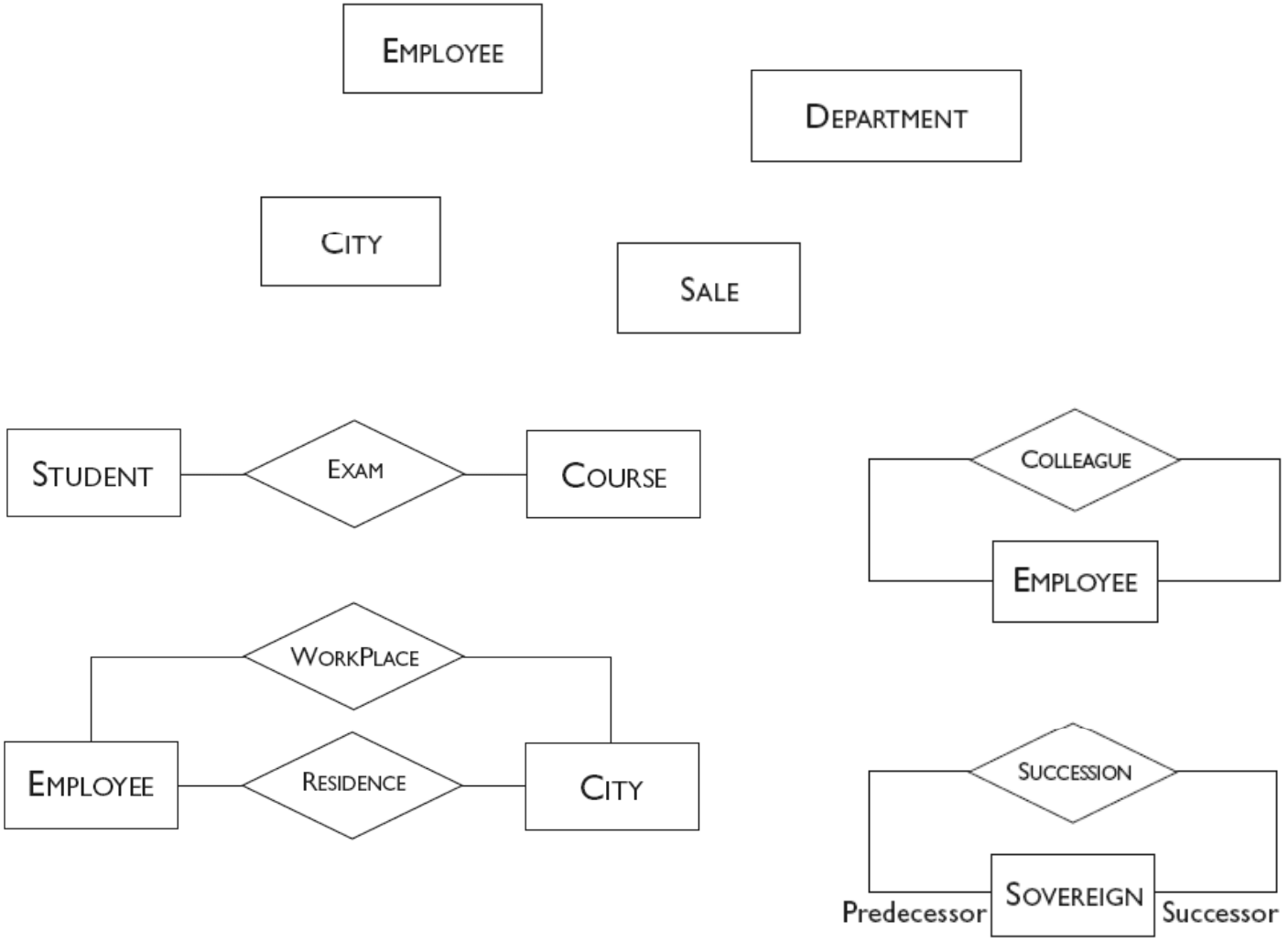
- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
  - Database design in E-R model usually converted to schema in the relational model which is used for storage and processing

Constructs of ER Model

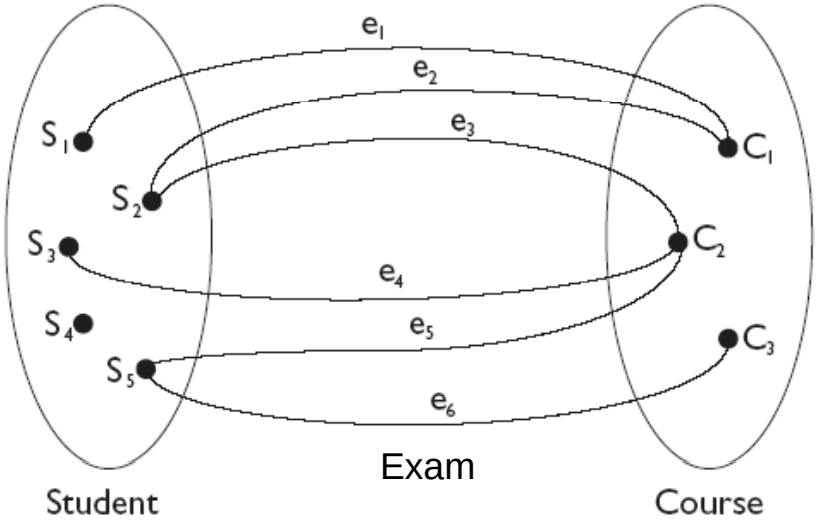
Construct	Graphical representation
Entity	
Relationship	
Simple attribute	
Composite attribute	
Cardinality of a	
Cardinality of an attribute	
Internal identifier	
External identifier	
Generalization	
Subset	

Examples from Atzeni, et al, 2000.

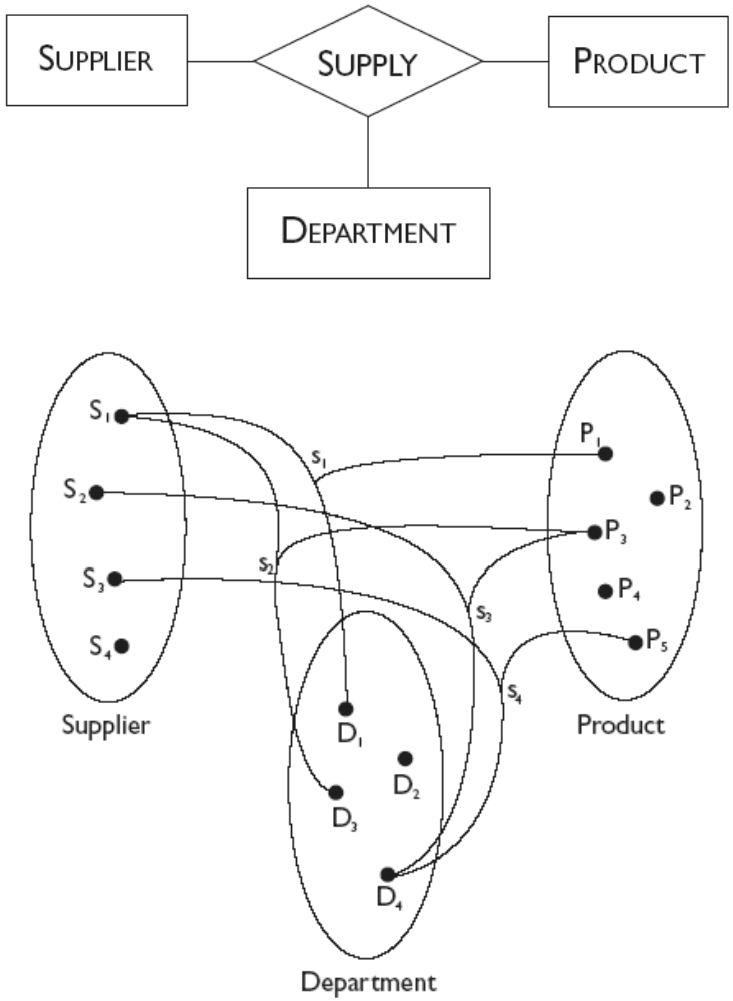
Entities & Relationships



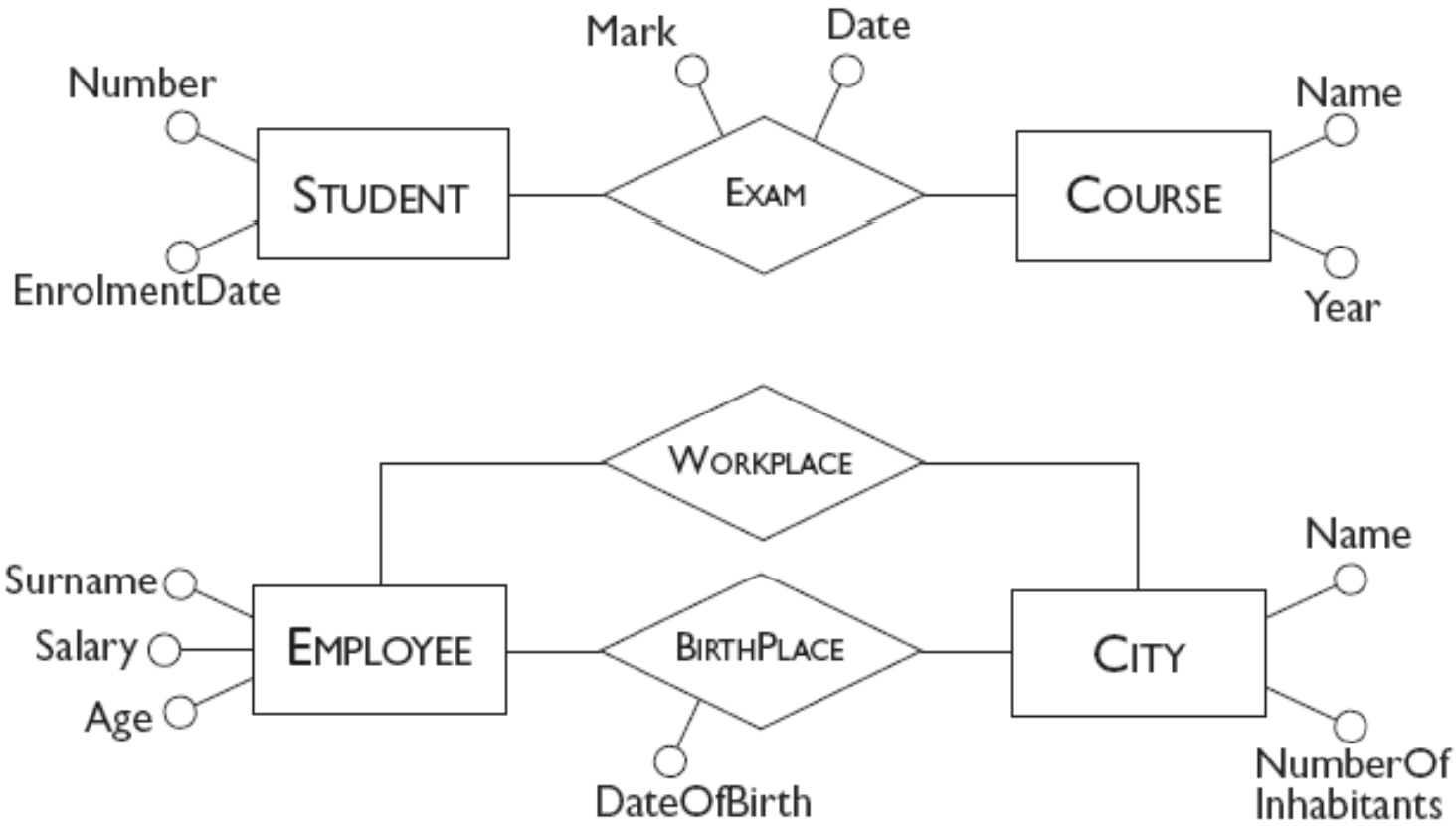
Entity & Relationship Occurrences



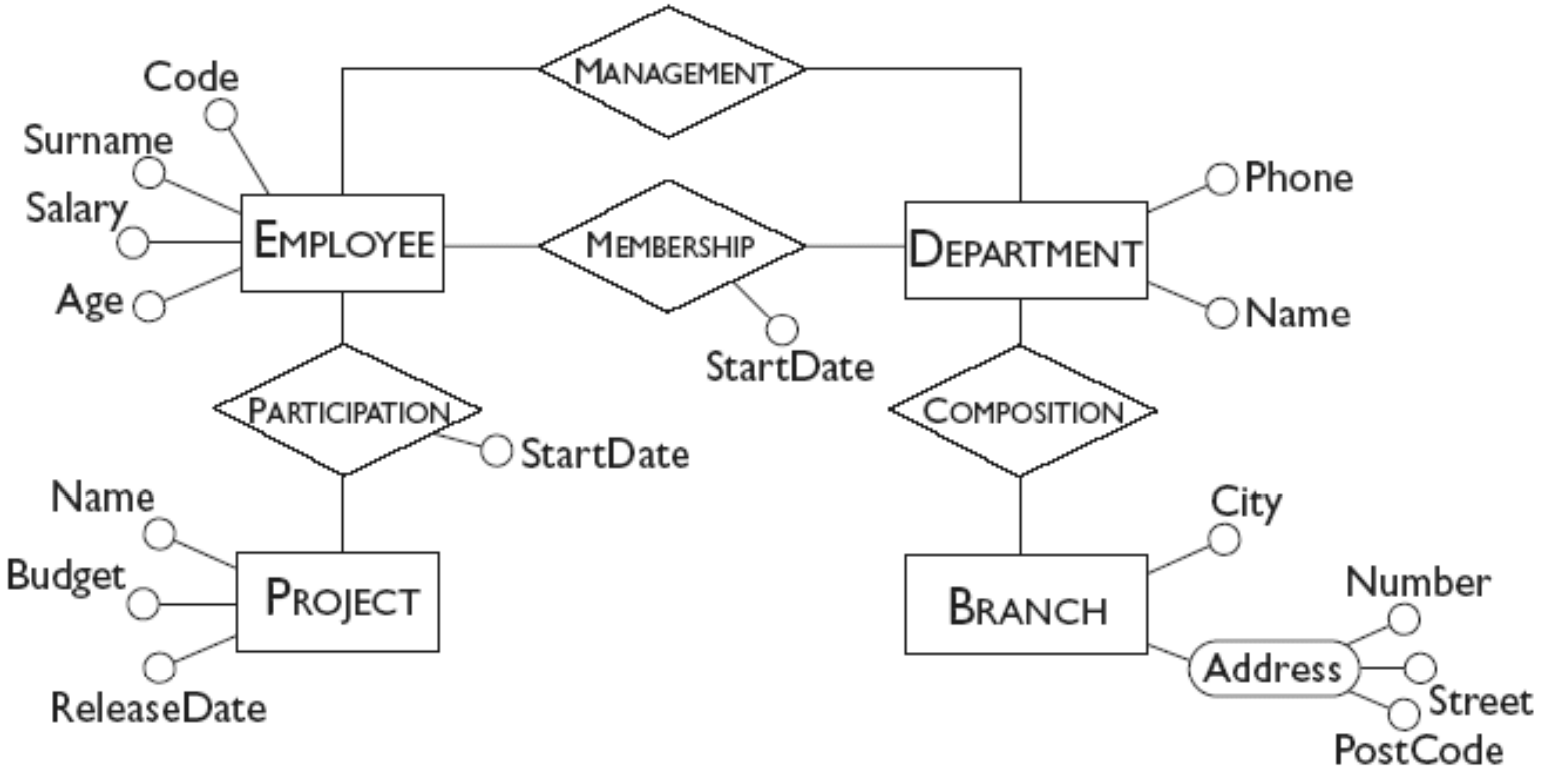
Ternary Relationship



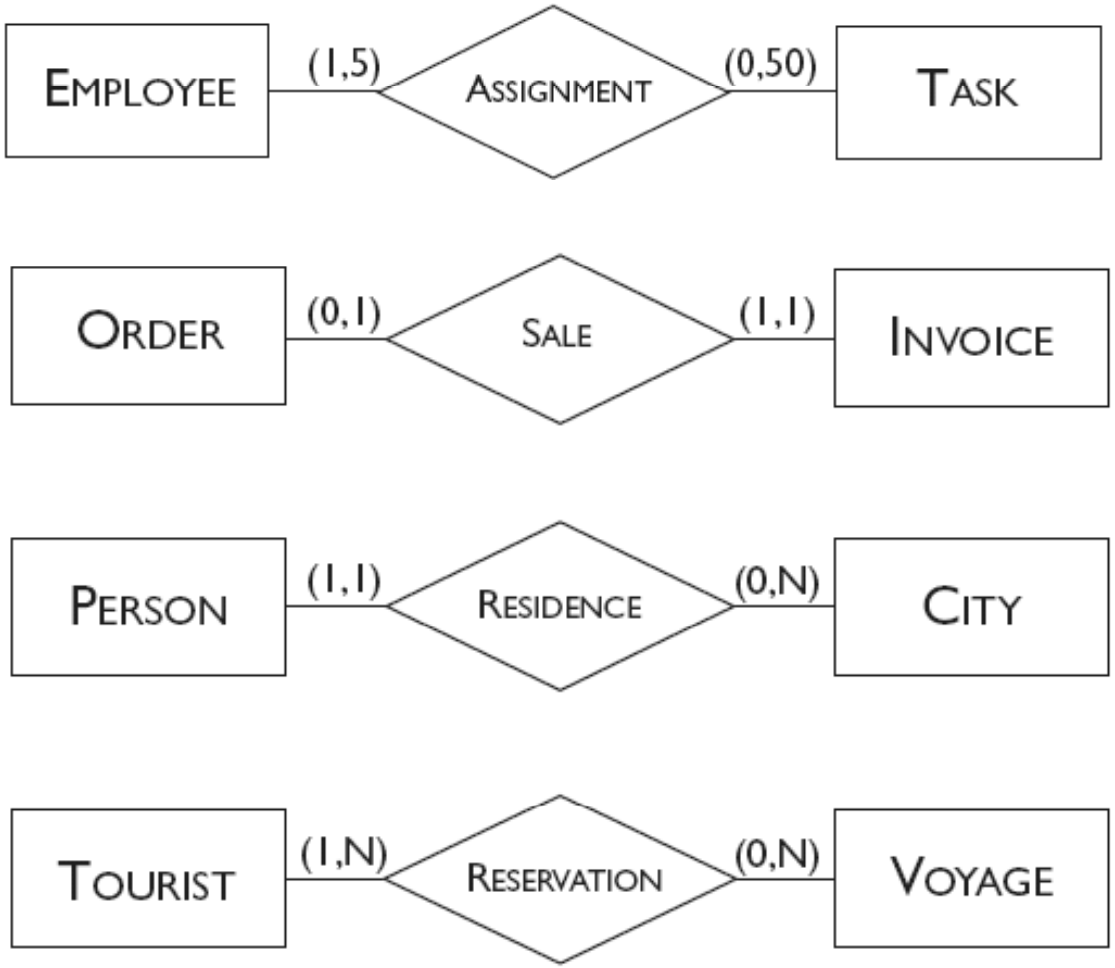
Attributes



An ER Schema

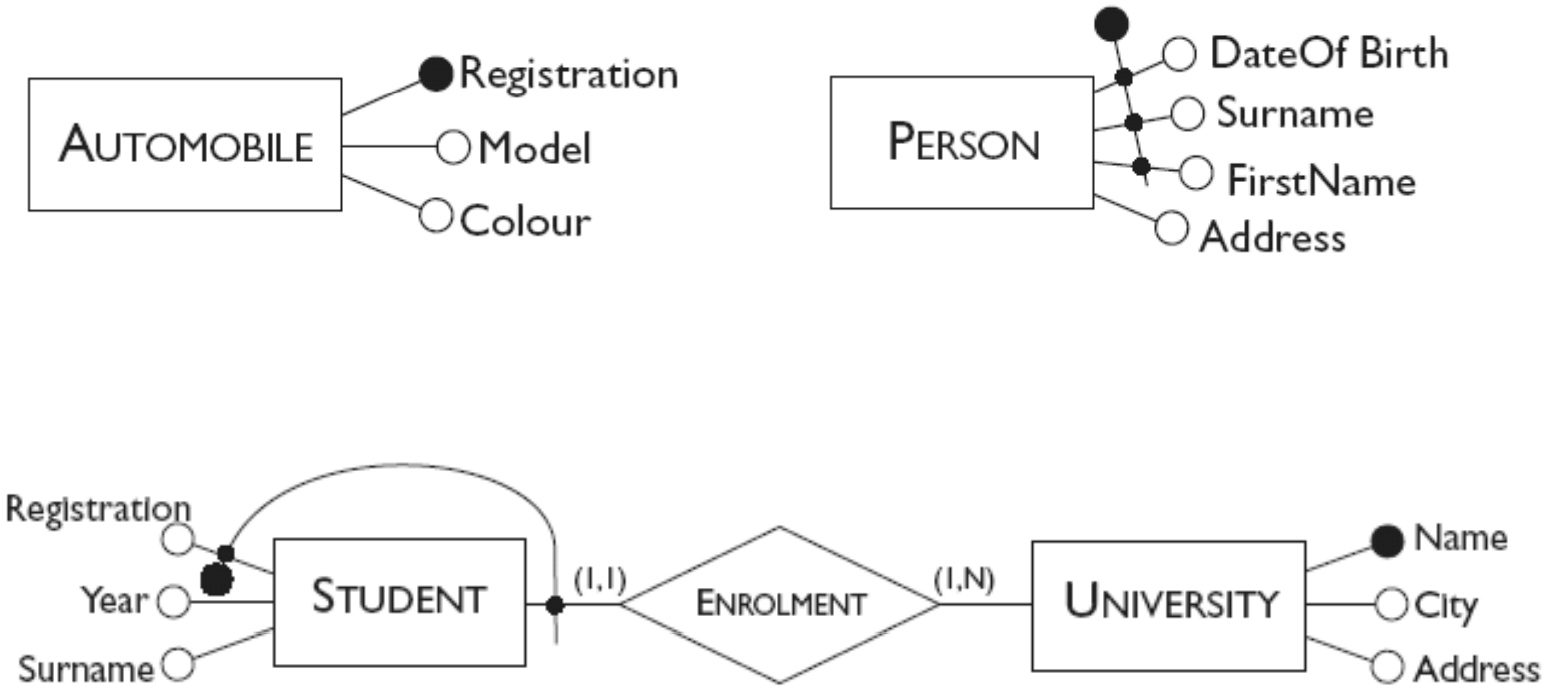


Relationship Cardinalities

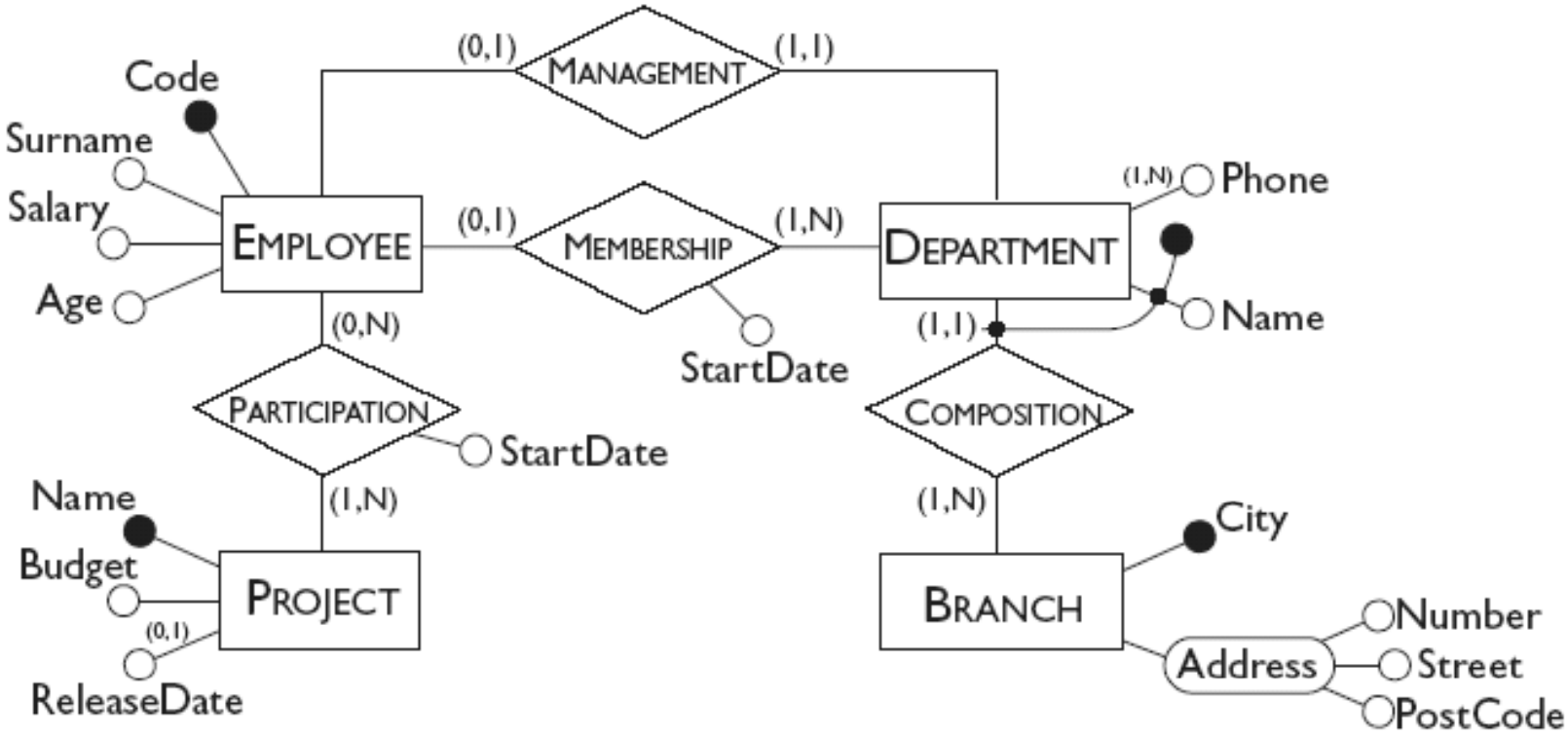


Examples from Atzeni, et al, 2000.

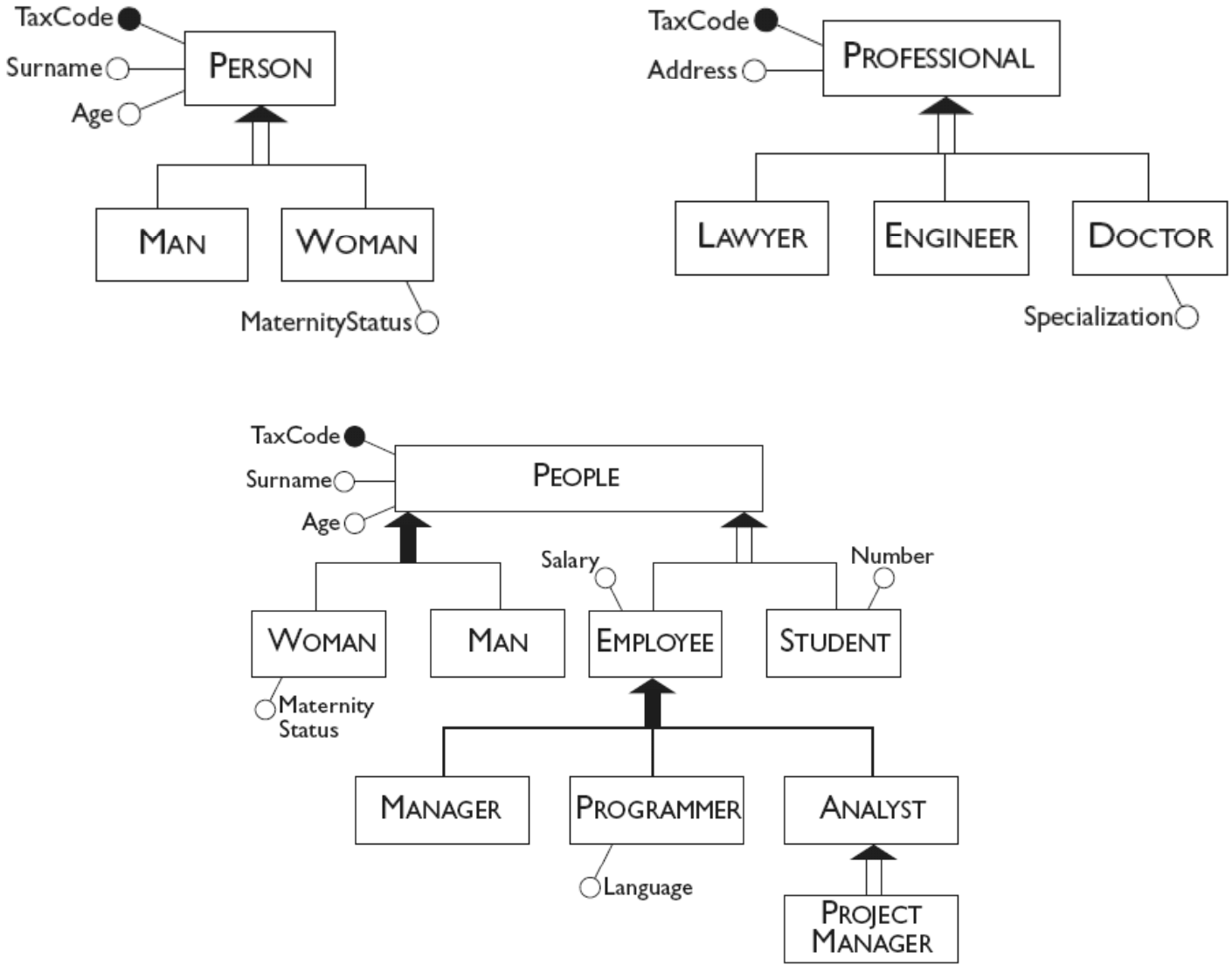
Identifiers (Keys)



ER Schema with Keys

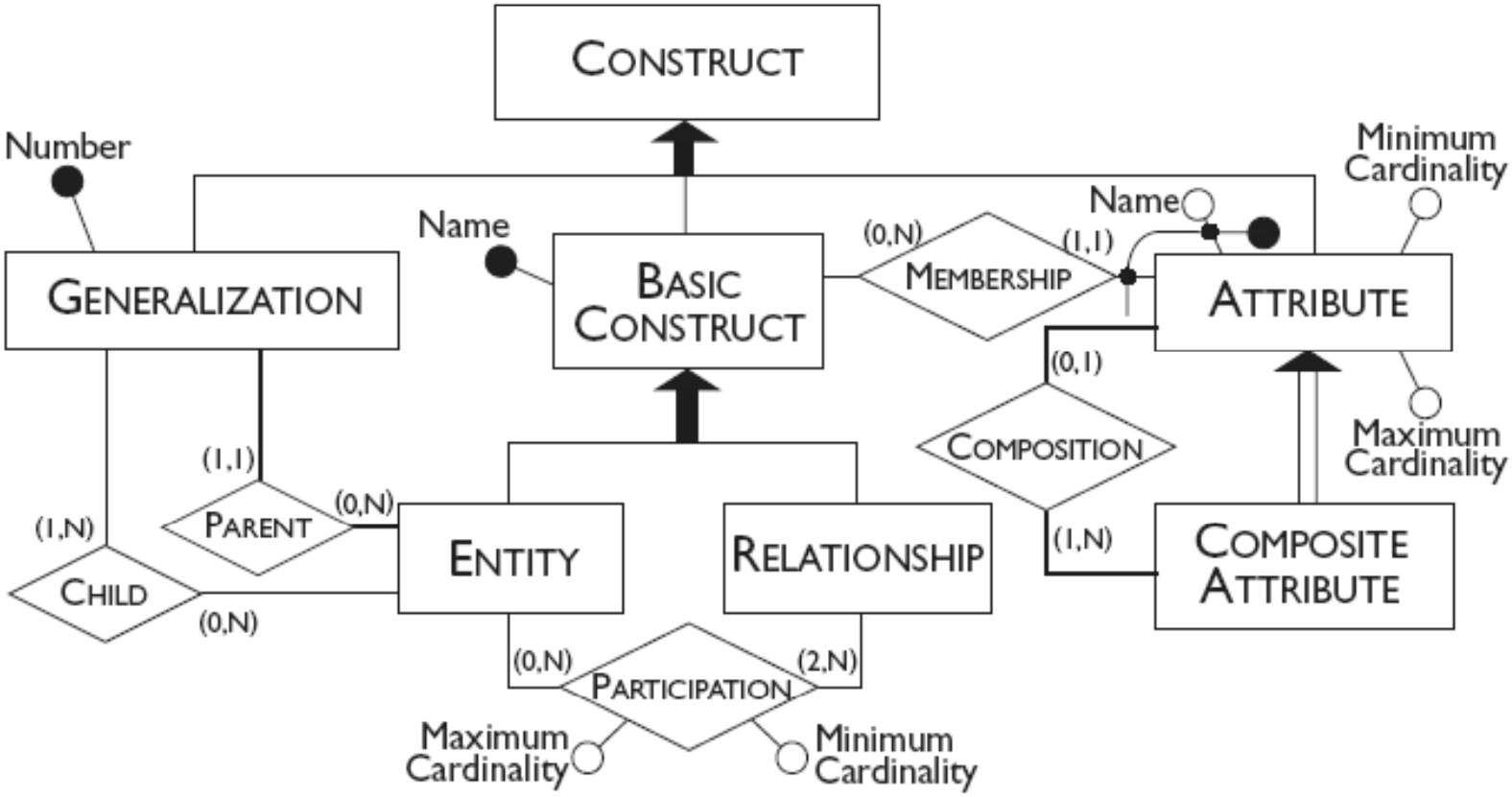


Generalization / Specialization

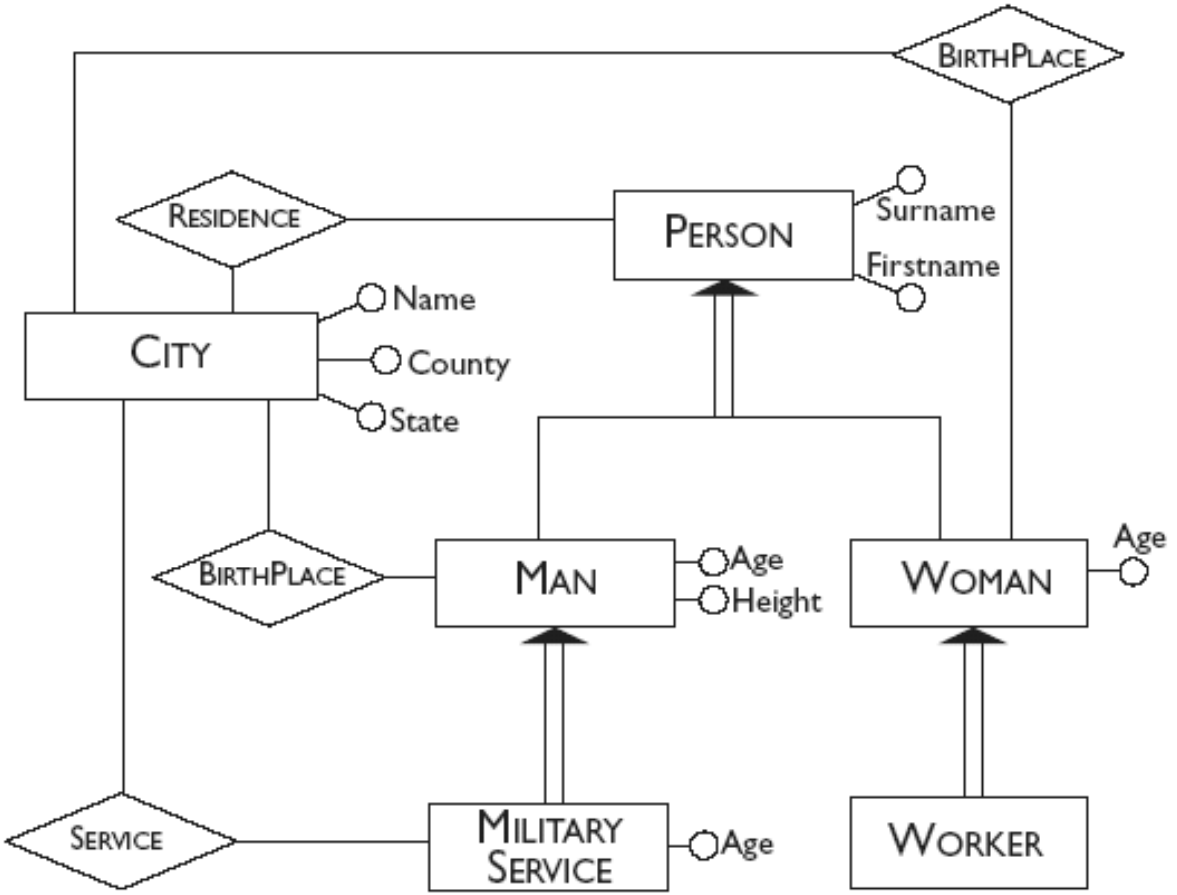


Examples from Atzeni, et al, 2000.

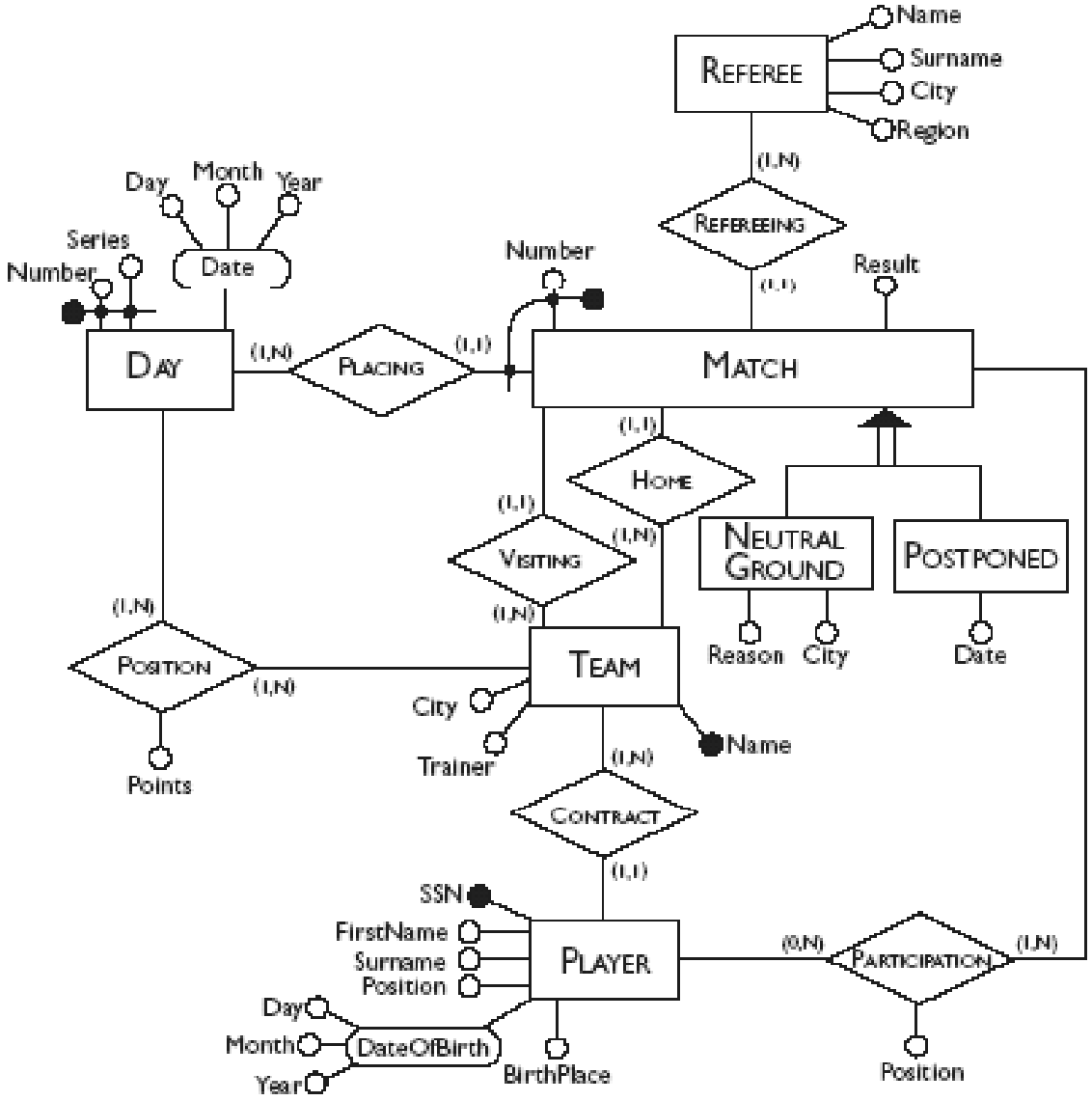
An ER Schema



Exercise: Correct/Refine the ER Schema



Exercise: Interpret the ER Diagram



Exercise: Create an ER Schema

- A local airline offers flights, each of which has a number that identifies the flight (for example Seoul-Tokyo), a date (25 Mar 2008), a departure time (14:00) and an arrival time (15:00), a departure airport and a destination airport. There are domestic and international flights. The international flights can have one or more stopovers. For completed flights, information to be recorded is the actual time of departure and arrival (for example with reference to the flight given above, 14:05 and 15:07). For future flights, the number of seats available must be known.

Requirements Collection and Analysis

- Collection and analysis of the requirements proceeds conceptual design
- *Requirements collection*
  - consists of the complete identification of the problems that the application must solve, and the features that should characterize such an application.
- *Requirements analysis*
  - consists of the clarification and organization of the requirements specification
- Main sources of requirements
  - 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.

Requirements Analysis

- Natural language specifications
  - 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.
- Some rules for requirements analysis
  - Choose the appropriate level of abstraction.
  - Standardize sentence structure.
  - Avoid complex phrases.
  - Identify synonyms and homonyms, and standardize terms.
  - Make cross-references explicit.
  - Construct a glossary of terms.

Examples from Atzeni, et al, 2000.

Example

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.

Examples from Atzeni, et al, 2000.

Glossary of Terms

Term	Description	Synonym	Links
Trainee	Participant in a course. Can be an employee or selfemployed.	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

Examples from Atzeni, et al, 2000.

Rewriting & Structuring of Requirements

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.

Examples from Atzeni, et al, 2000.

Rewriting & Structuring of Requirements (cont.)

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.

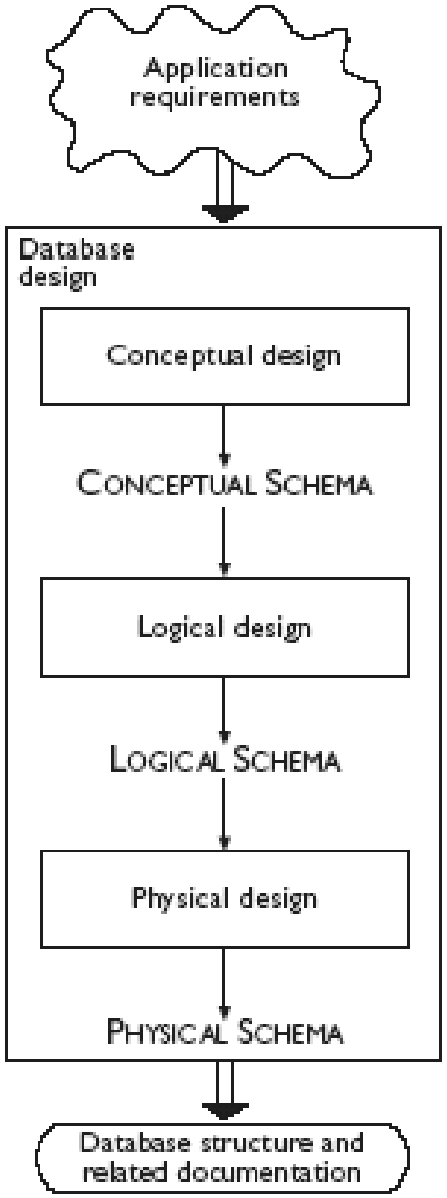
Examples from Atzeni, et al, 2000.

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);
- *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).

Phases of Database Design

- The conceptual design of a database
  - consists of the construction of an ER schema
  - to provide an optimal description of the user requirements



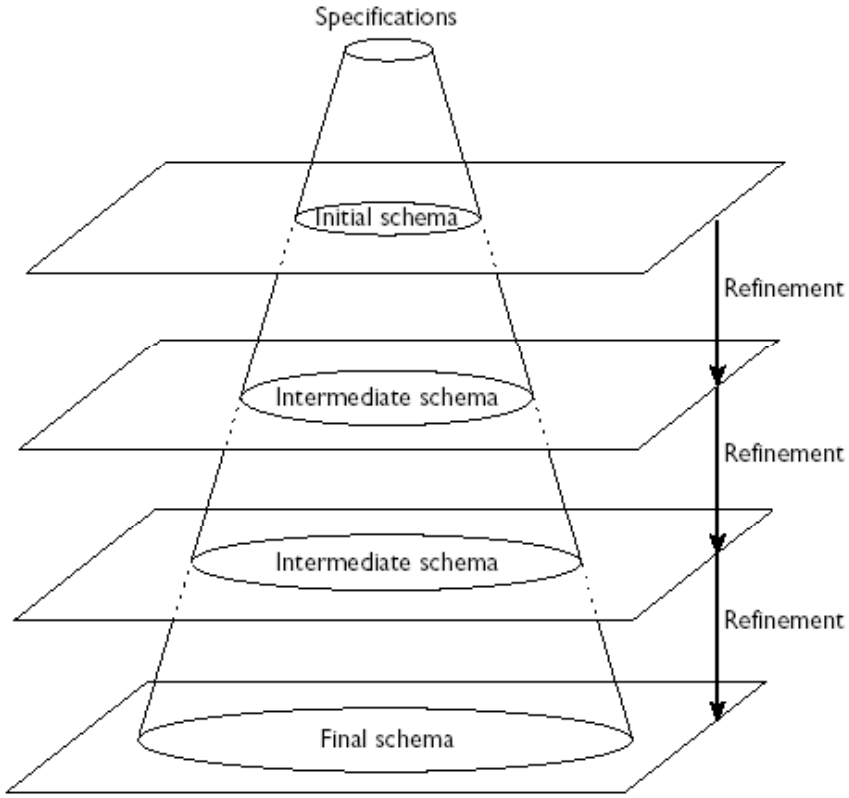
Examples from Atzeni, et al, 2000.

Design Strategies for Conceptual Design

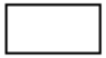
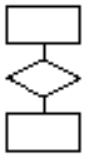
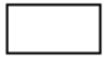
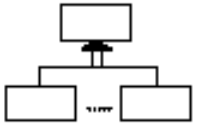
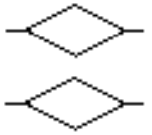
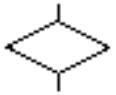
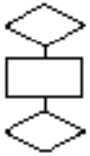
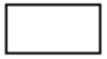
- The development of a conceptual schema must be considered an engineering process
  - design strategies used in other disciplines can be applied to it
- Design strategies
  - Top-down
  - Bottom-up
  - Inside-out
  - Mixed

Top-down Strategy

- The conceptual schema is produced by
  - 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.
- The schema is usually modified using some basic transformations called *top-down transformation primitives*.

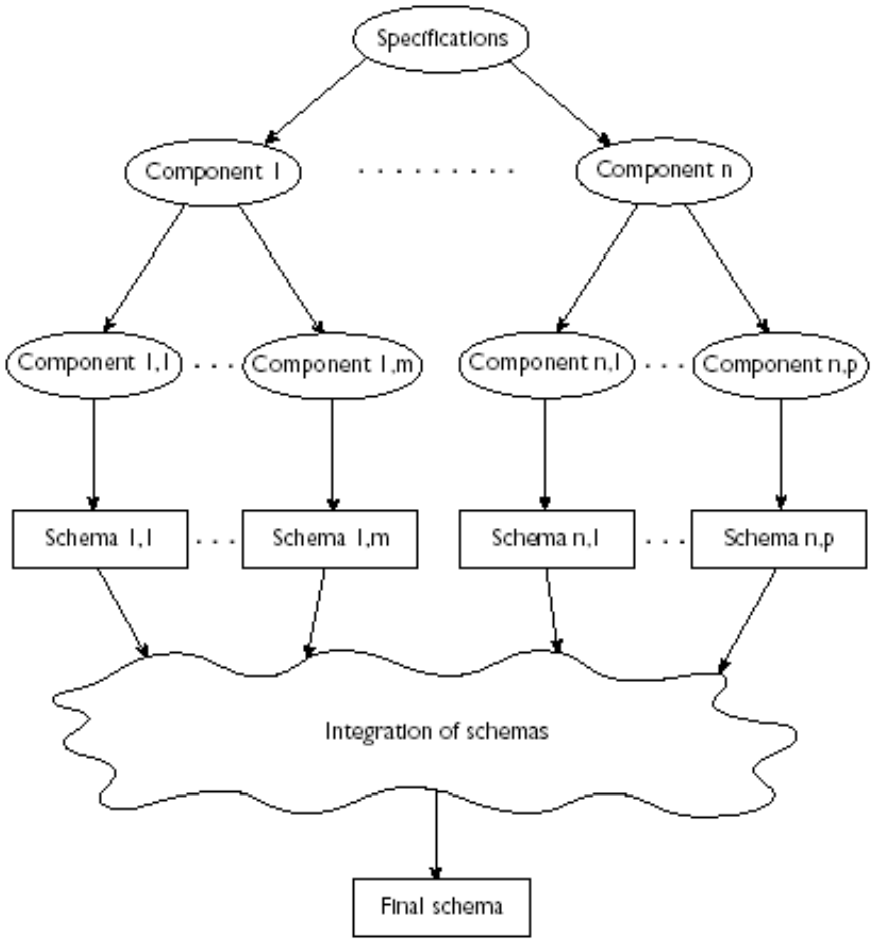


Top-down Transformation Primitives

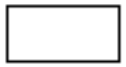
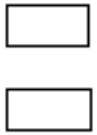
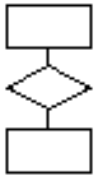
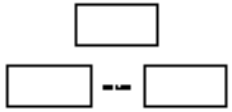
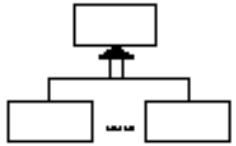
Transformation	Initial concept	Result
T1 From one entity to two entities and a relationship between them		
T2 From one entity to a generalization		
T3 From one relationship to multiple Relationships		
T4 From one relationship to an entity with Relationships		
T5 Adding attributes to an entity		
T6 Adding 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*.



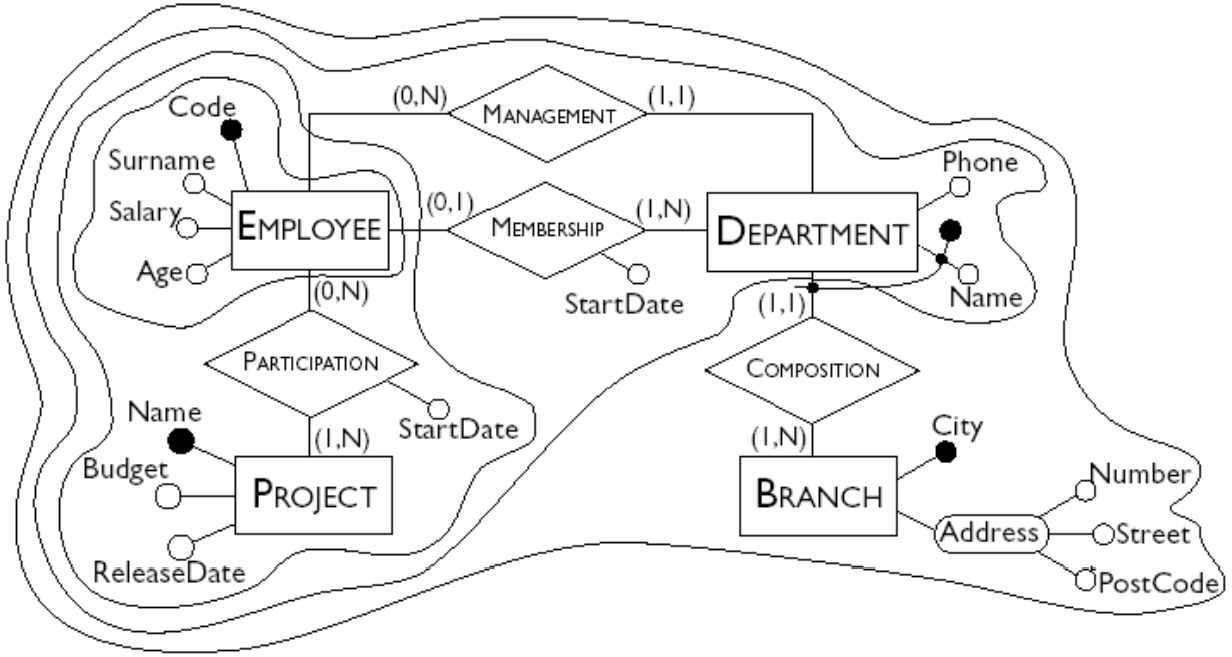
Bottom-up Transformation Primitives

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		

Examples from Atzeni, et al, 2000.

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.

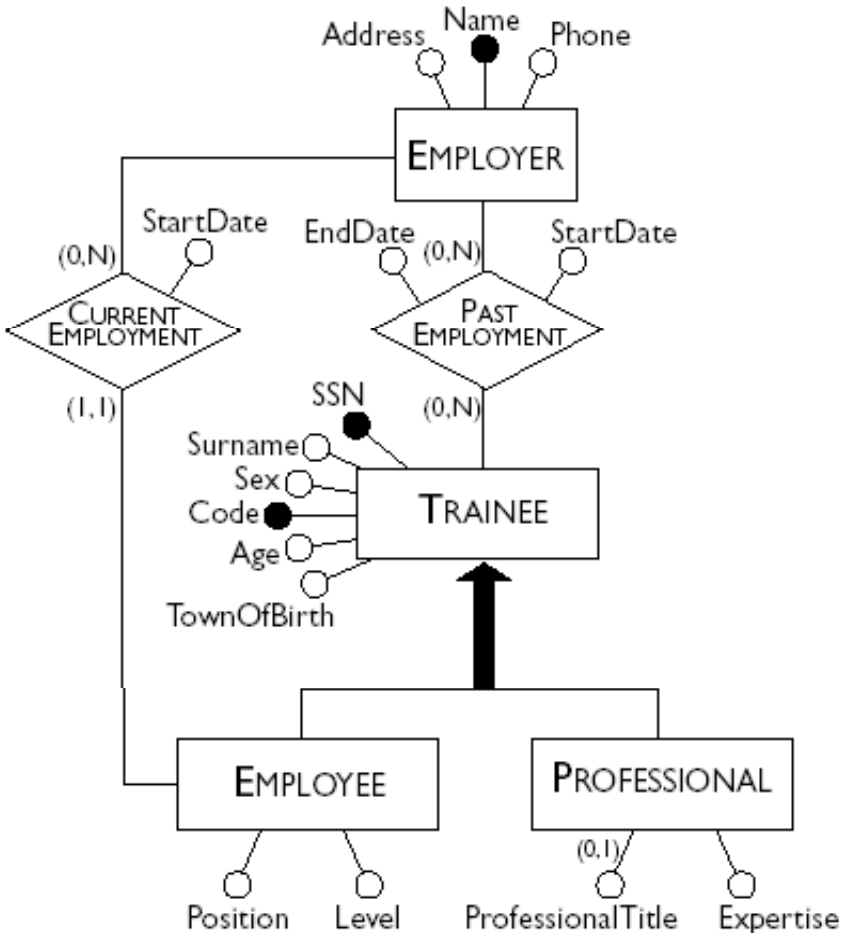


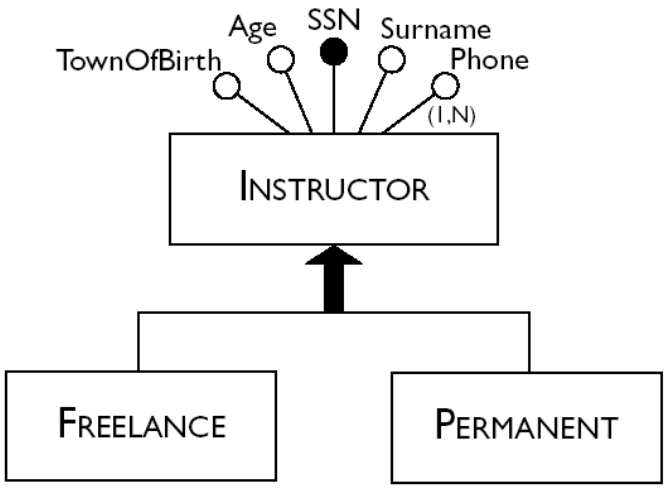
Mixed Strategy

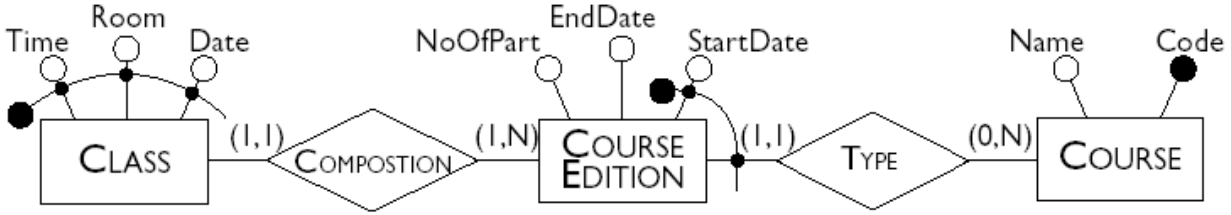
- In a mixed strategy the designer decomposes the requirements into a number of components, as in the bottom-up strategy, but not to the extent where all the concepts are separated.
- At the same time he or she defines a *skeleton schema* containing the main concepts of the application. This skeleton 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).

- Example: Skeleton schema

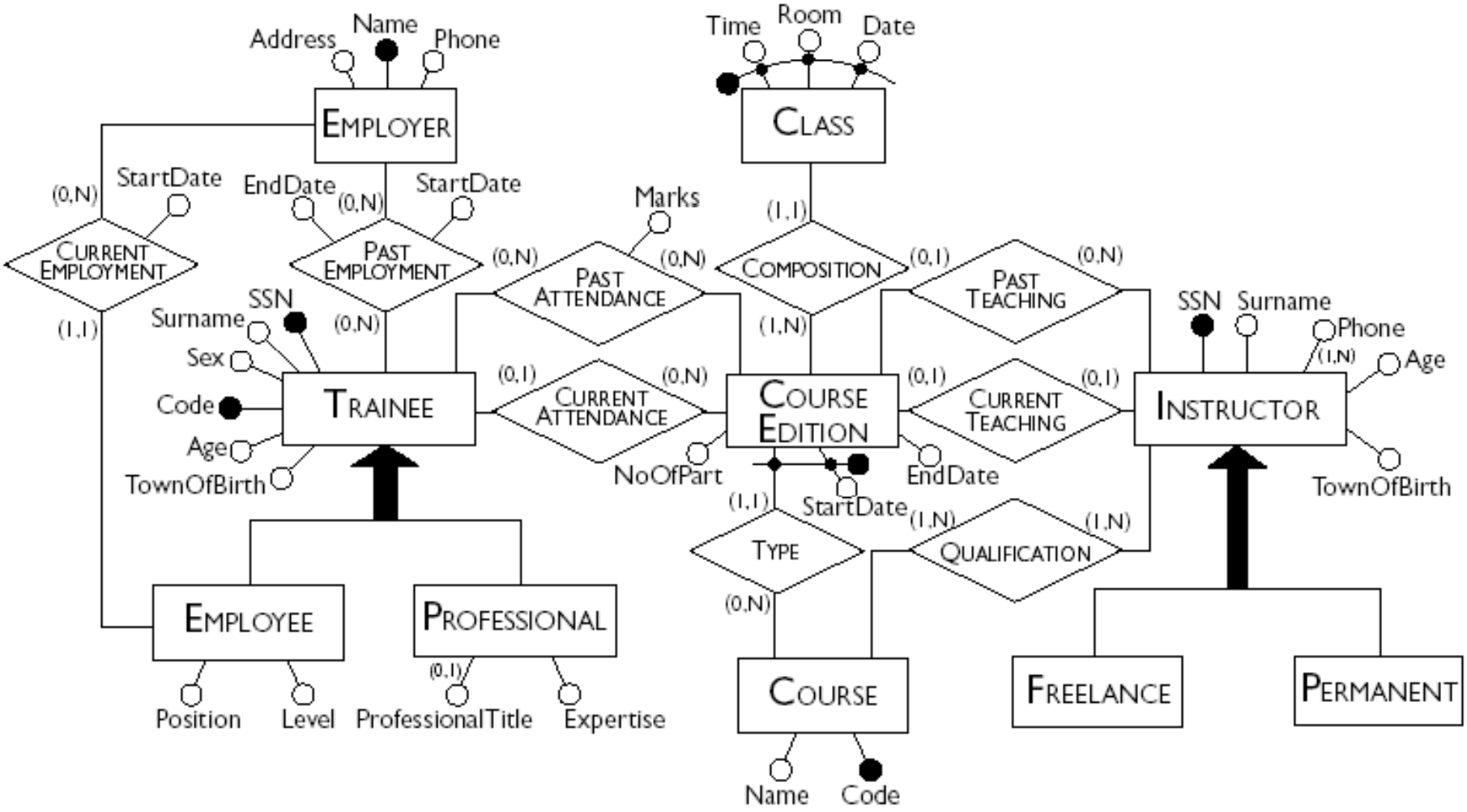
Refinement of Portions of Skeleton Schema







Final E-R schema

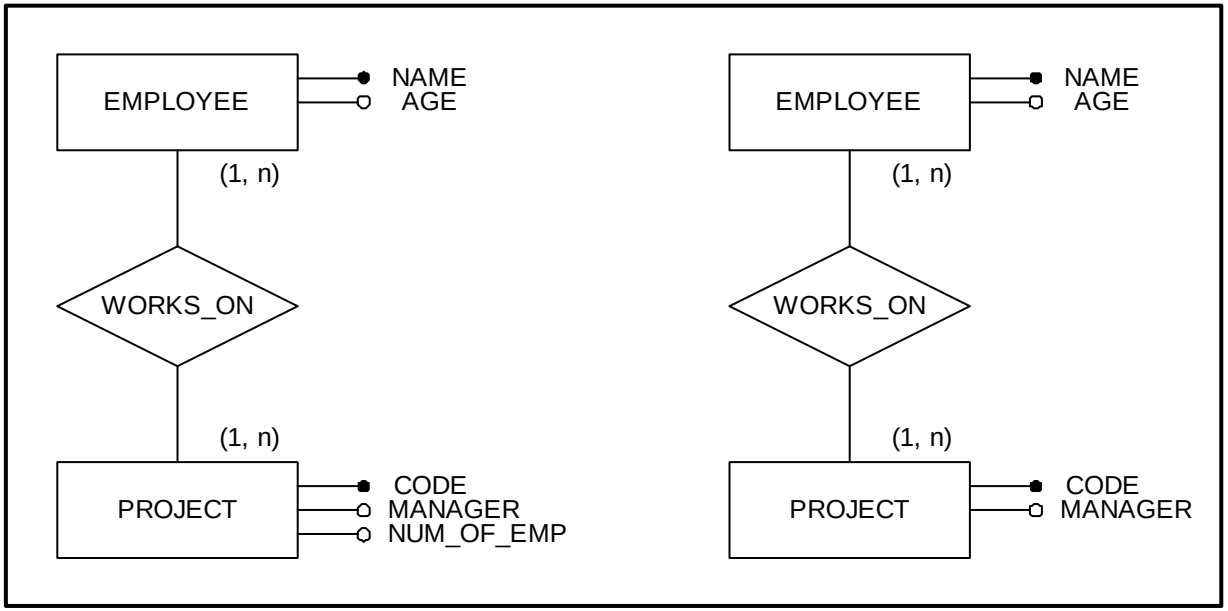


Qualities of a Database Schema

1. Completeness
   - A schema represents all relevant features of the application domain.
2. Correctness
   - A schema properly uses the concepts of the ER model.
   - Types of correctness
     1) Syntactic correctness
        - Concepts are properly defined in the schema.
     2) Semantic correctness
        - Concepts are used according to their definitions.
        - For example, To represent products, using an attribute when we need to represent several properties of products. (e.g., code, price, parts…)

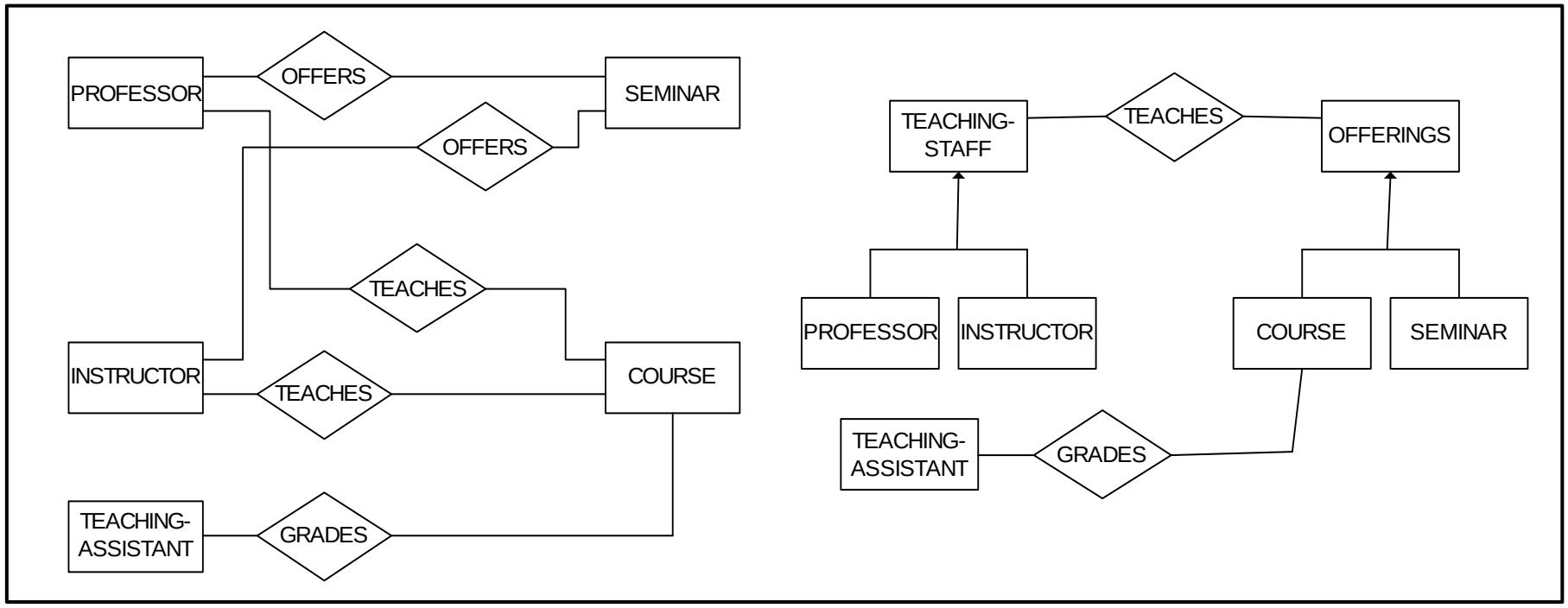
Qualities of a Database Schema (cont.)

3. Minimality
   - Every aspect of the requirements appears *only once* in the schema.
   - No concept can be deleted from the schema without losing some information.



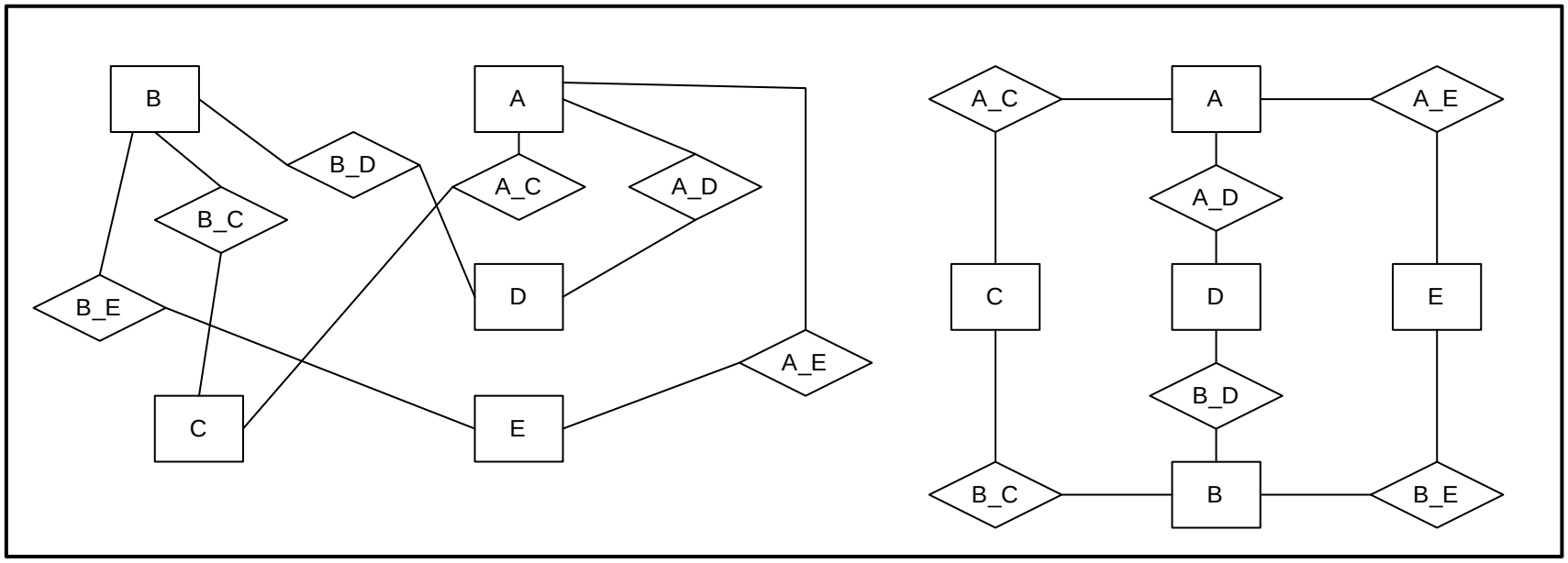
Qualities of a Database Schema (cont.)

4. Expressiveness
   - A schema represents requirements in a natural way and can be easily understood through the meaning of ER, without the need for further explanation.



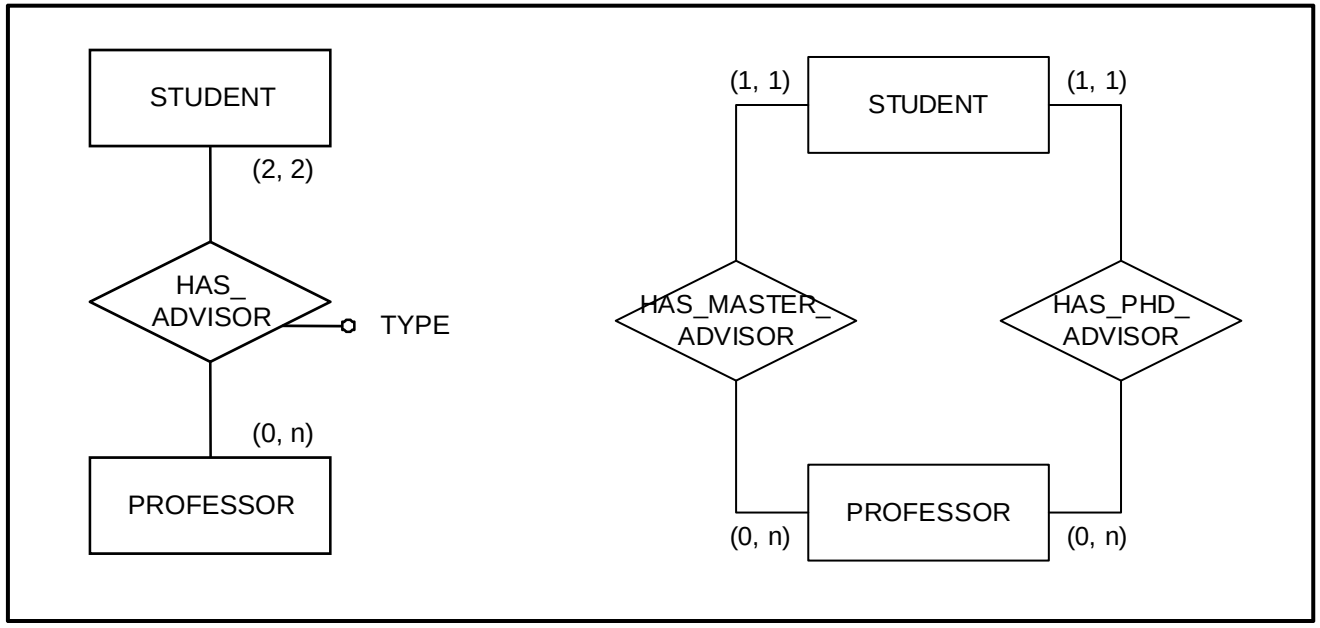
Qualities of a Database Schema (cont.)

5. Readability
   - A schema respects certain aesthetic criteria that make the diagram graceful



Qualities of a Database Schema (cont.)

6. Self-explanation
   - A large number of properties can be represented using the conceptual model itself, without other formalisms (e.g., annotations in natural language.)



Qualities of a Database Schema (cont.)

7. Extensibility
   - A schema is easily adapted to changing requirements when it can be decomposed into pieces (modules, or views).
8. Normality
   - 3NF, BCNF, 4NF, …

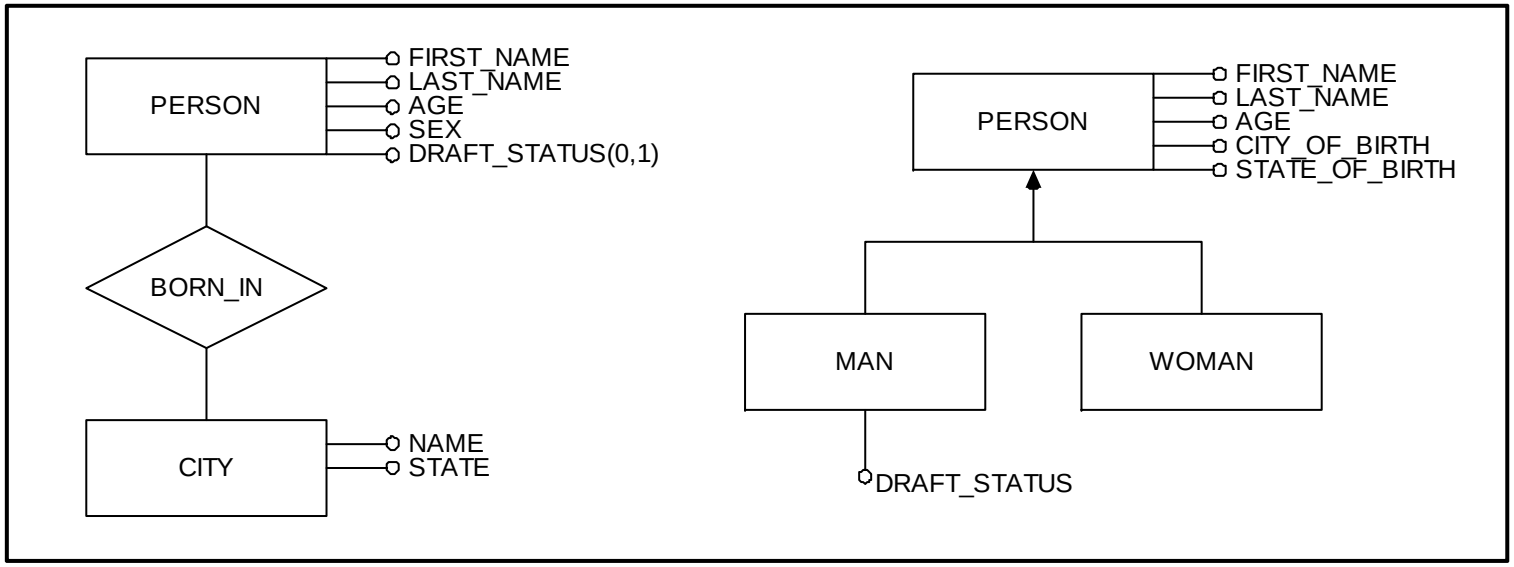
General Criteria for Concept Representation

- 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.

Examples from Atzeni, et al, 2000.

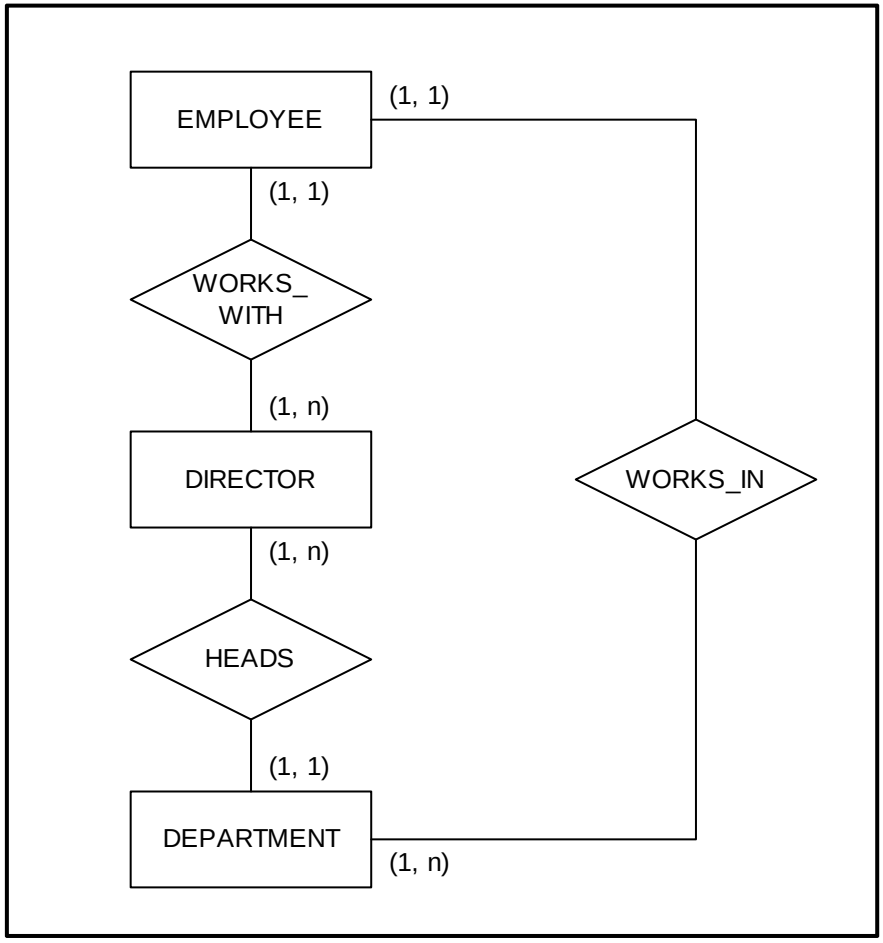
Schema Transformations

- Information content
  - for each query Q that can be expressed on schema S_1, there is a query Q' that can be expressed on schema S_2 giving the same answer, and vice versa
- Classification of transformations
  1. Information-preserving transformations
  2. Information-changing transformations



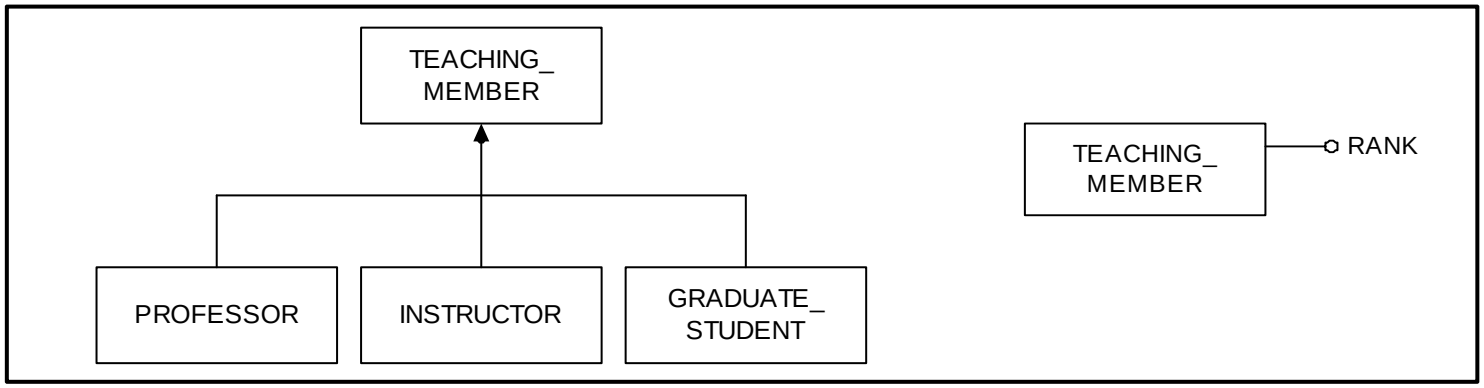
Achieving Minimality

- Cycles of Relationships

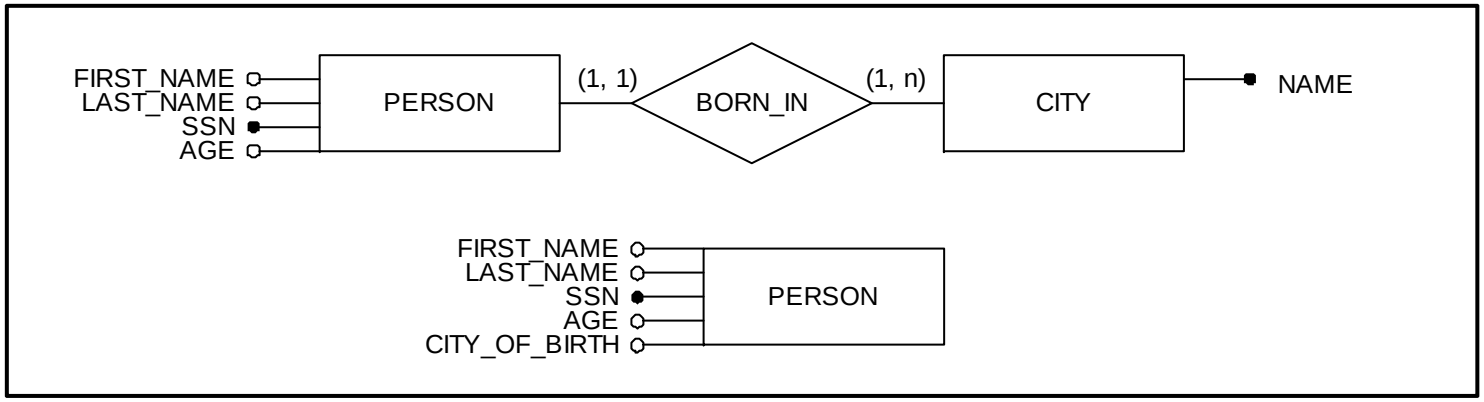


Achieving Expressiveness/Self-Explanation

- Elimination of Dangling Sub-entities in Generalization Hierarchies

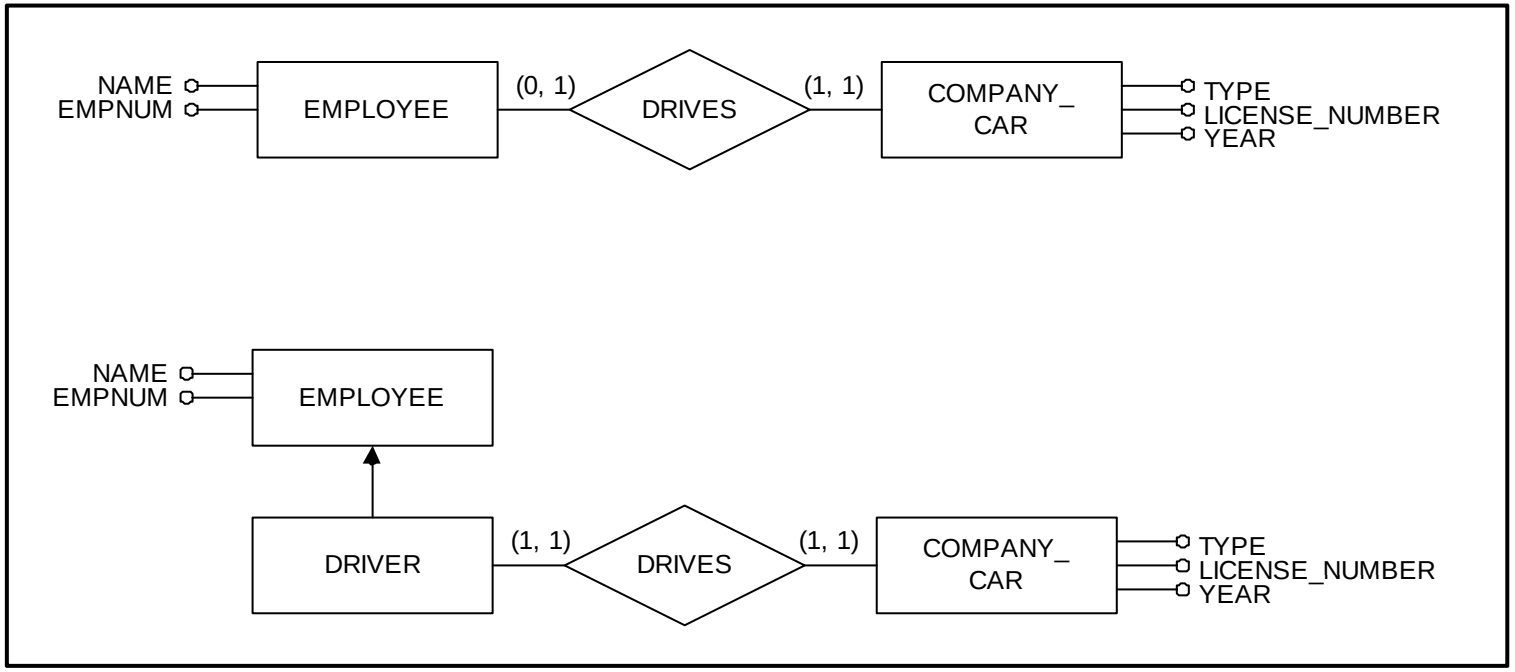


- Elimination of Dangling Entities



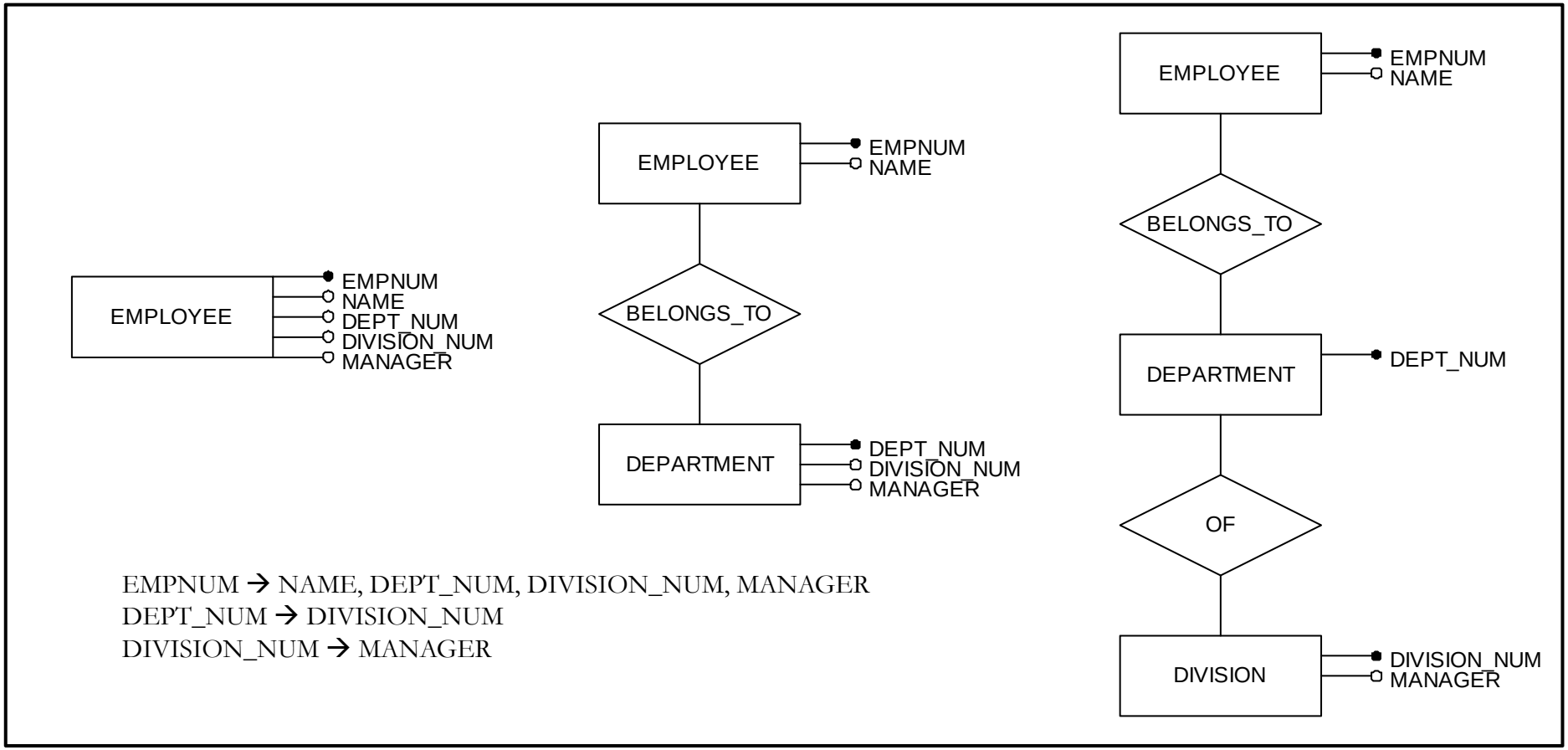
Achieving Expressiveness/Self-Explanation (cont.)

- Creation of Generalization
- Creation of a New Subset



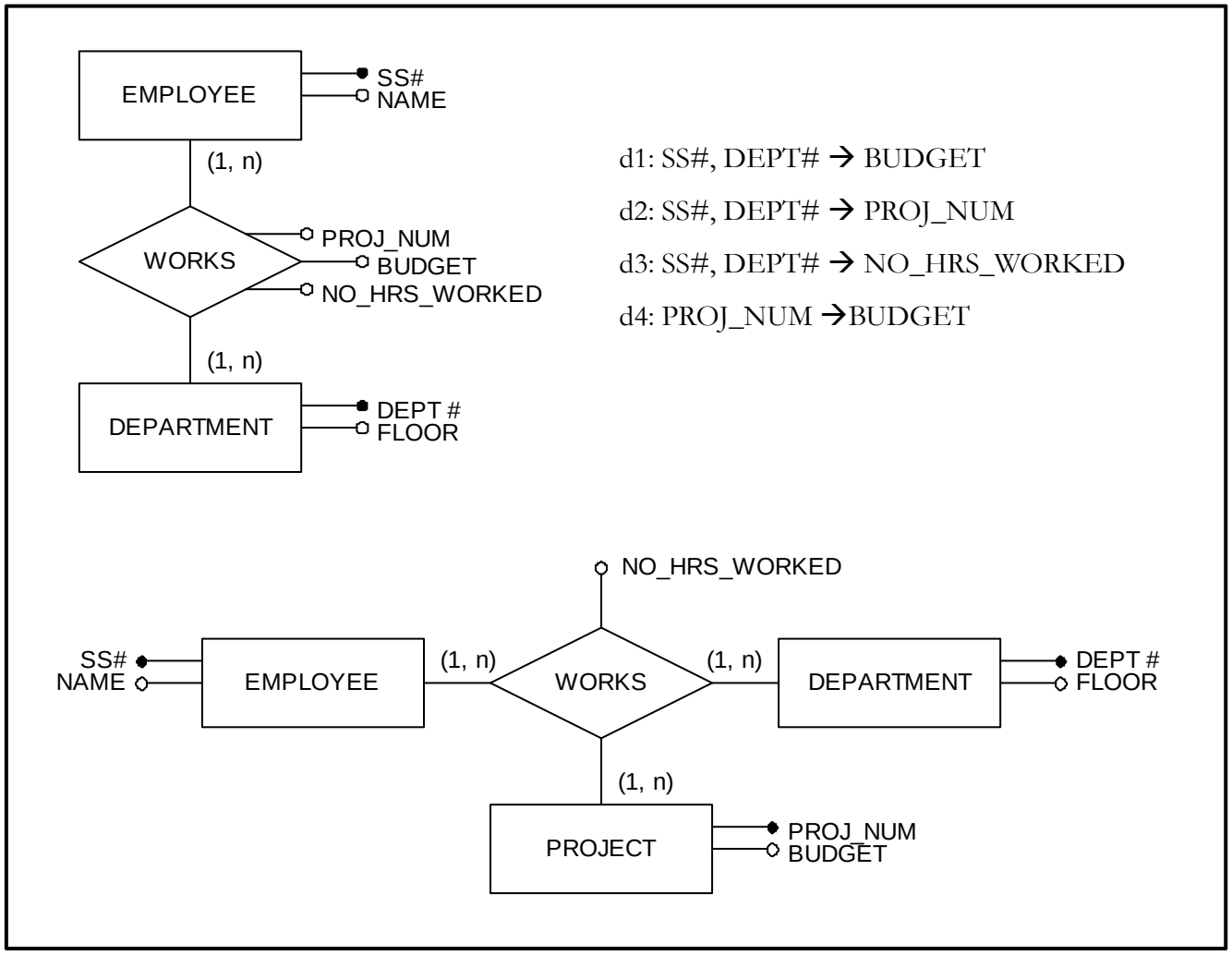
Achieving Normality

- BCNF & 3NF
  - By use of functional dependency



Examples from Atzeni, et al, 2000.

Achieving Normality (cont.)



The Conceptual Design Process

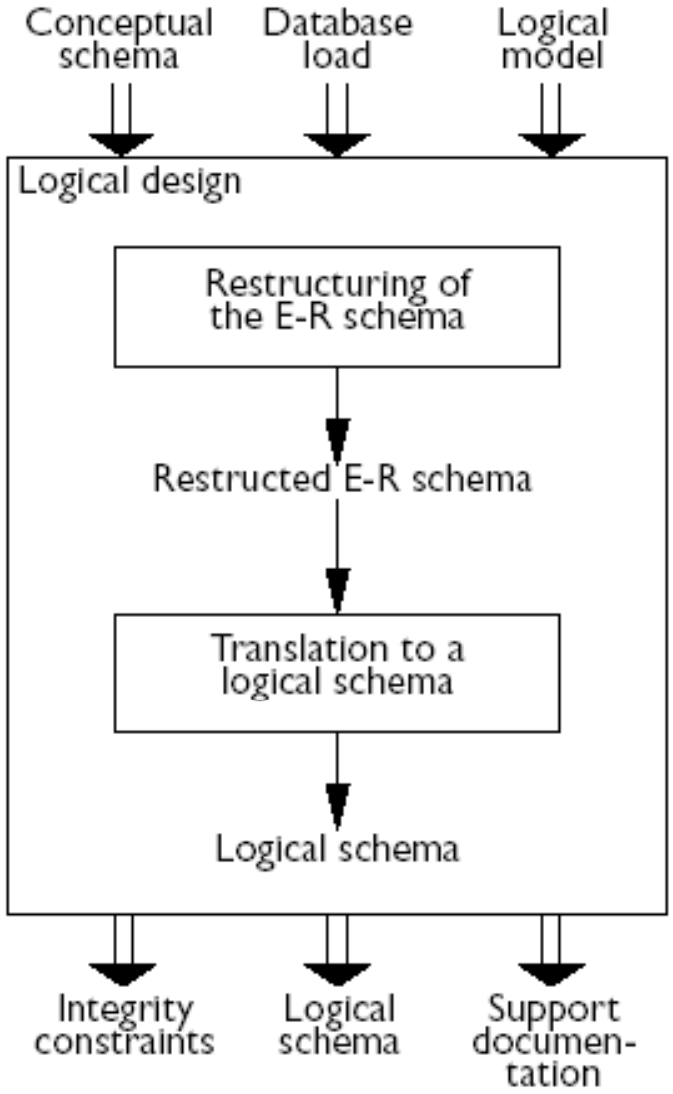
1. Analysis of requirements
   (a) Construct a glossary of terms.
   (b) Analyze the requirements and eliminate any ambiguities.
   (c) Arrange the requirements in groups.
2. Basic step
   (a) Identify the most relevant concepts and represent them in a skeleton schema.
3. Decomposition step (to be used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

The Conceptual Design Process (cont.)

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 sub-schemas 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.

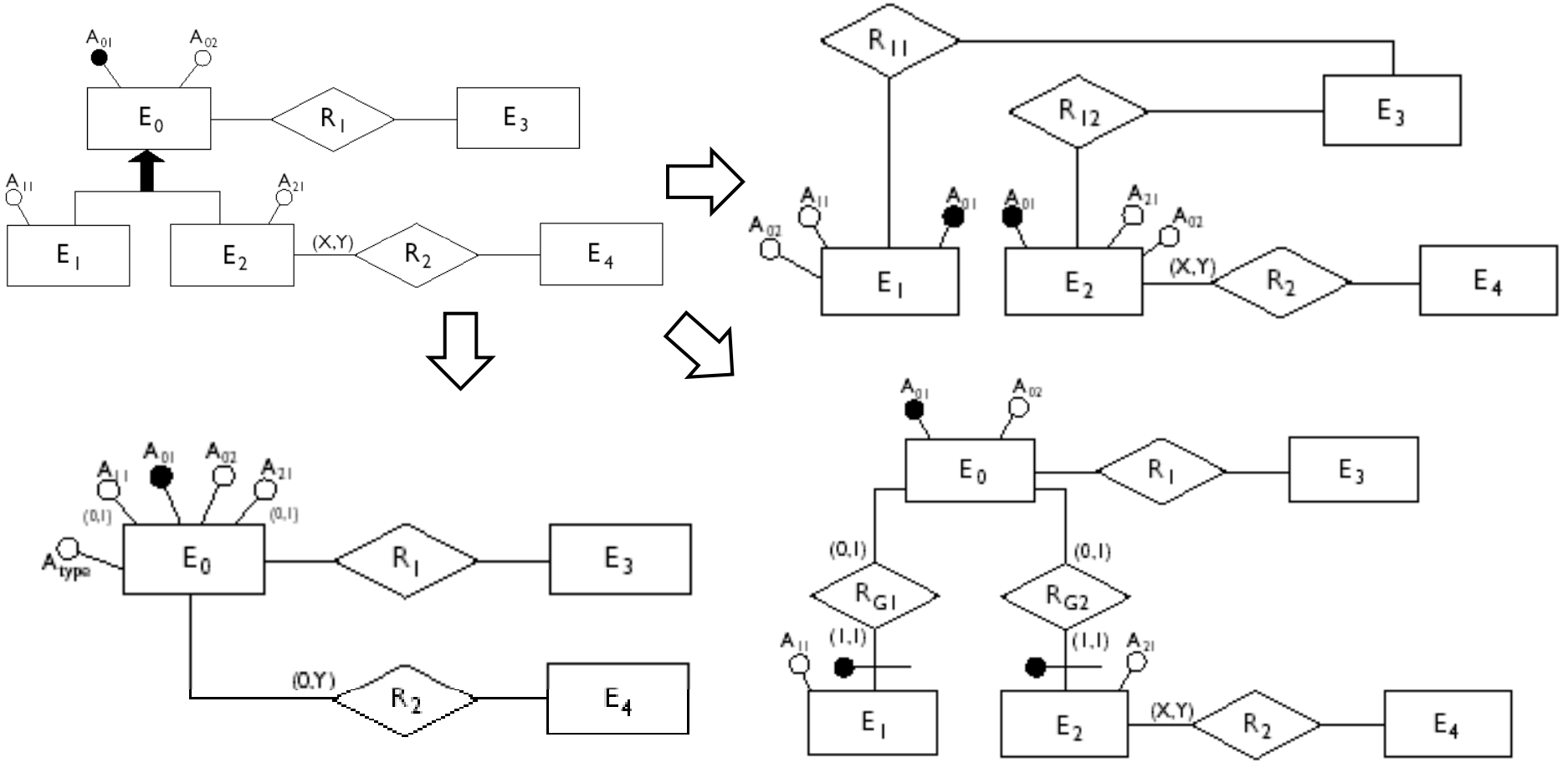
Logical Database Design

- The aim is to *construct a relational schema* that correctly and efficiently represents all of the information described by a conceptual (ER) schema.
  - not all the constructs of the Entity-Relationship model can be translated naturally into the relational model;
  - the schema must be restructured in such a way as to make the execution of the projected operations as efficient as possible.

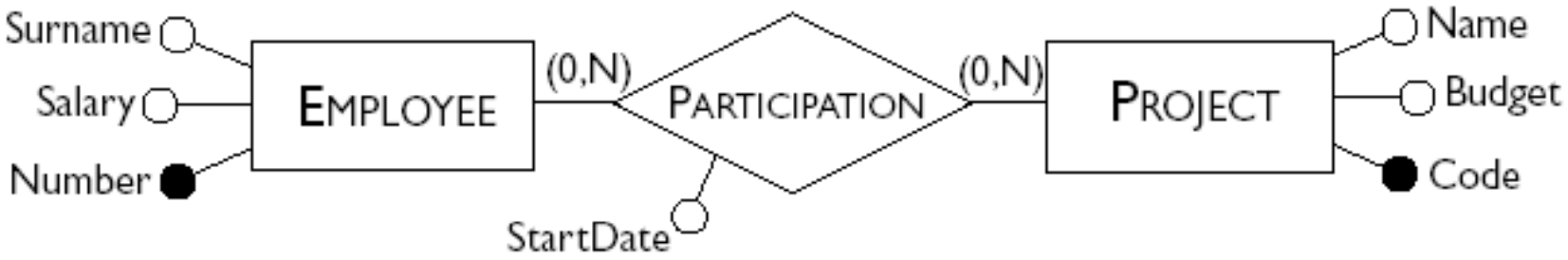


Restructuring Generalizations

- The relational model does not allow the direct representation of generalizations of the E-R model.

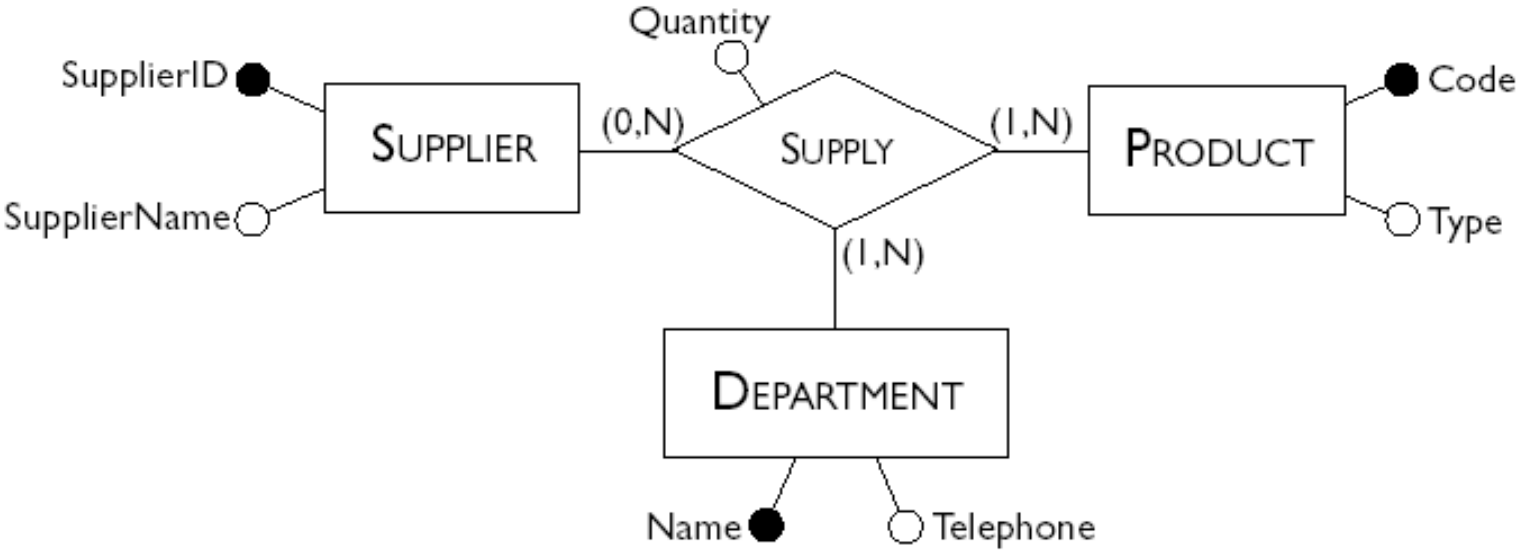


Many-to-Many Relationships



- EMPLOYEE(Number, Surname, Salary)
- PROJECT(Code, Name, Budget)
- PARTICIPATION(Employee, Project, StartDate)

Ternary Relationships



- SUPPLIER(SupplierID, SupplierName)
- PRODUCT(Code, Type)
- DEPARTMENT(Name, Telephone)
- SUPPLY(Supplier, Product, Department, Quantity)

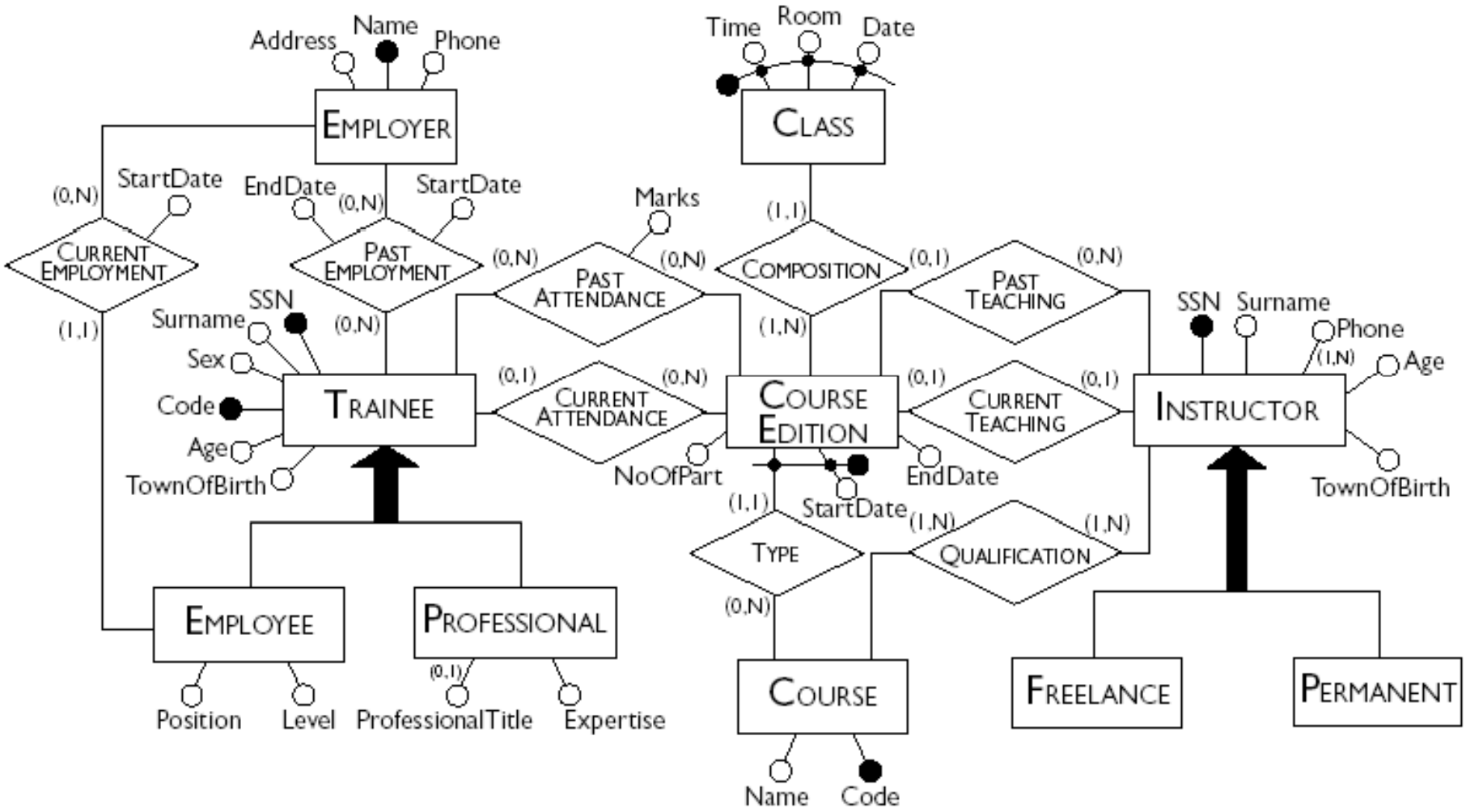
1-to-1 Relationships

- HEAD(Number, Name, Salary, Department, StartDate)
- DEPARTMENT(Name, Telephone, Branch)

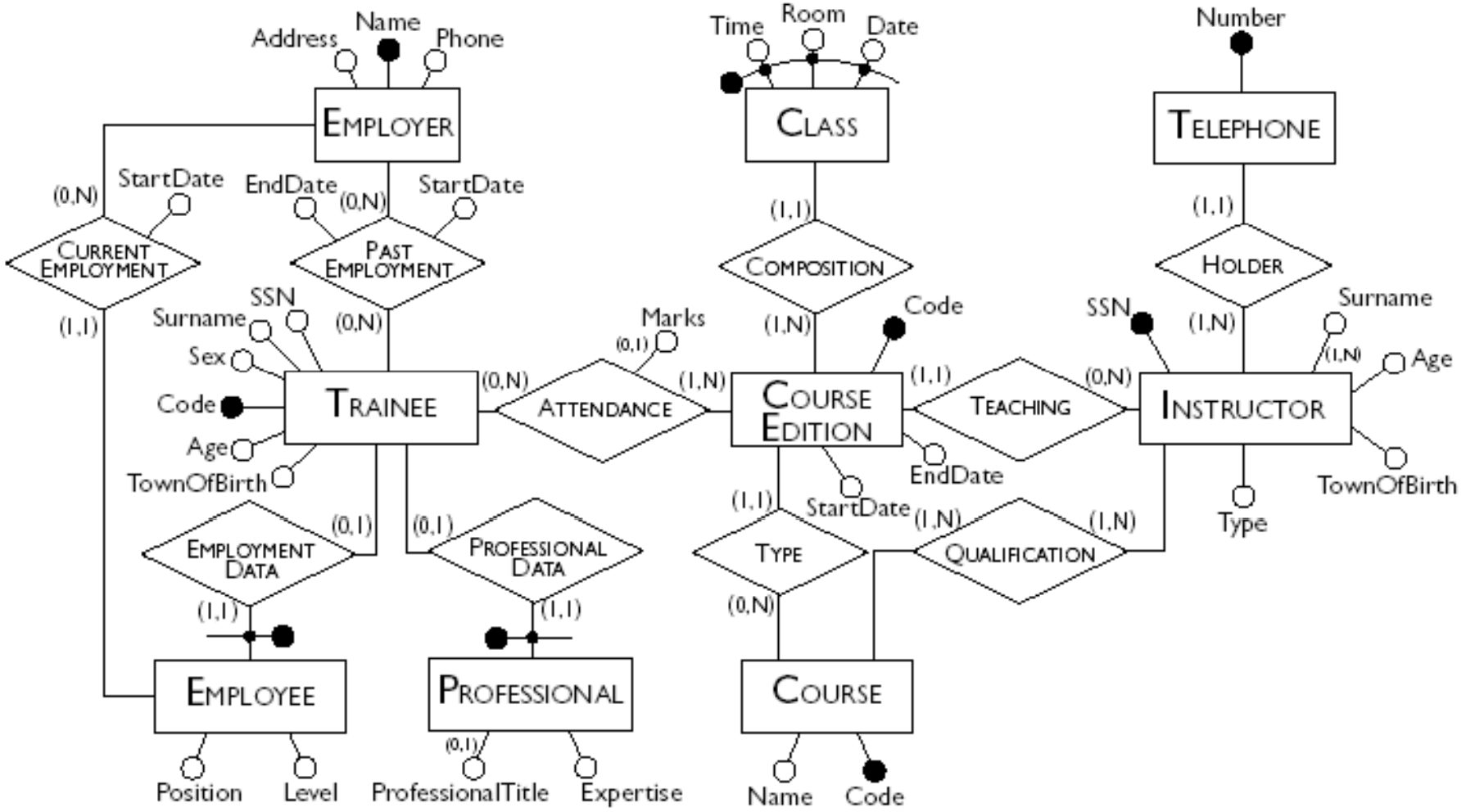
or

- HEAD(Number, Name, Salary)
- DEPARTMENT(Name, Telephone, Branch, Head, StartDate)

E-R schema of Training Company



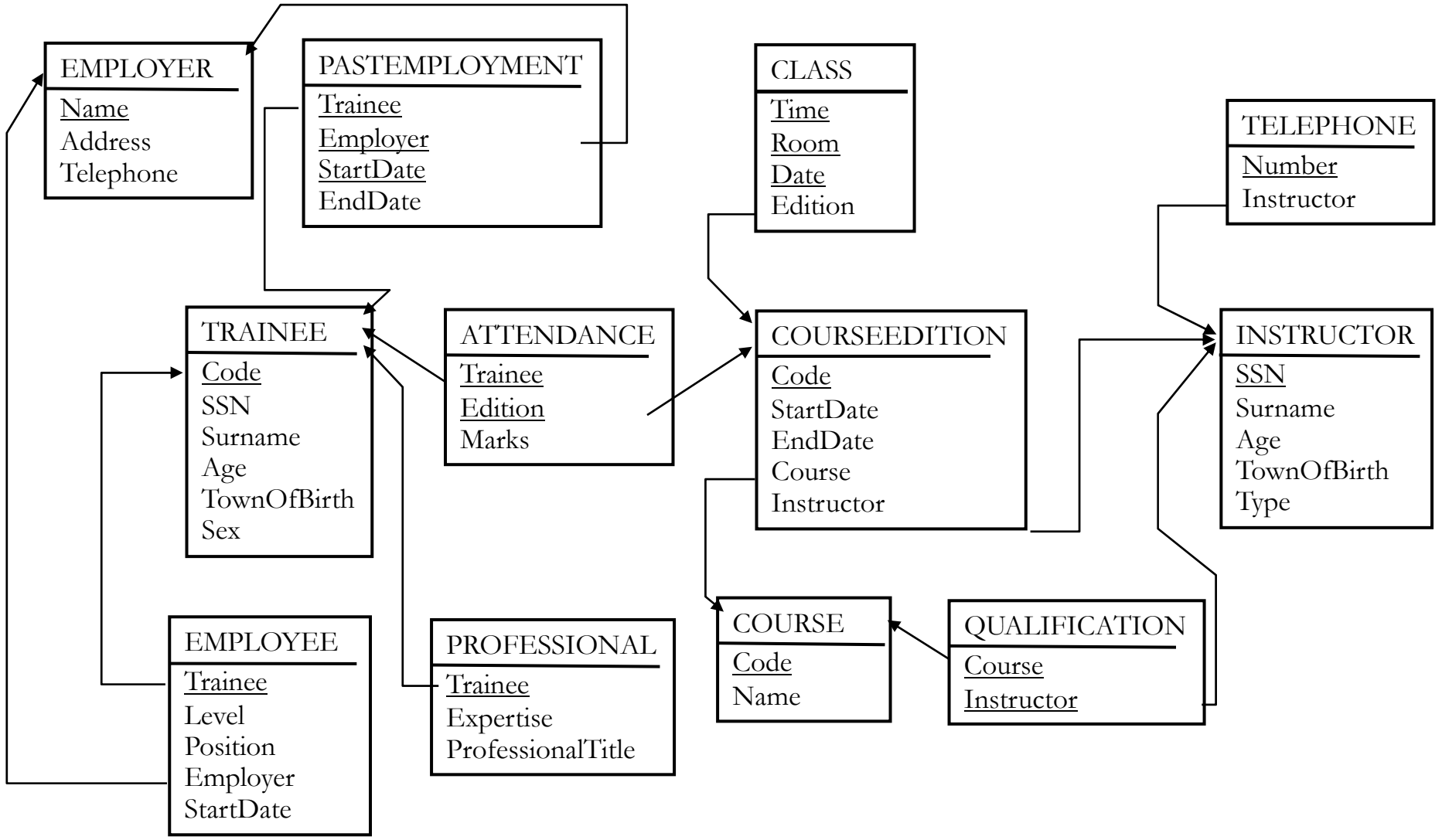
E-R schema After Restructuring



Translation into Relational Schema

- COURSEEDITION(Code, StartDate, EndDate, Course, Instructor)
- CLASS(Time, Room, Date, Edition)
- INSTRUCTOR(SSN, Surname, Age, TownOfBirth, Type)
- TELEPHONE(Number, Instructor)
- COURSE(Code, Name)
- QUALIFICATION(Course, Instructor)
- TRAINEE(Code, SSN, Surname, Age, TownOfBirth, Sex)
- ATTENDANCE(Trainee, Edition, Marks*)
- EMPLOYER(Name, Address, Telephone)
- PASTEMPLOYMENT(Trainee, Employer, StartDate, EndDate)
- PROFESSIONAL(Trainee, Expertise, ProfessionalTitle*)
- EMPLOYEE(Trainee, Level, Position, Employer, StartDate)

Relational Schema Diagram



END OF CONCEPTUAL DB DESIGN