Logical Design

- The aim of logical design is to construct a relational schema that correctly and efficiently represents all of the information described by an Entity-Relationship schema produced during the conceptual design phase.

- This is not just a simple translation from one model to another for two main reasons:

  – not all the constructs of the Entity-Relationship model can be translated naturally into the relational model;

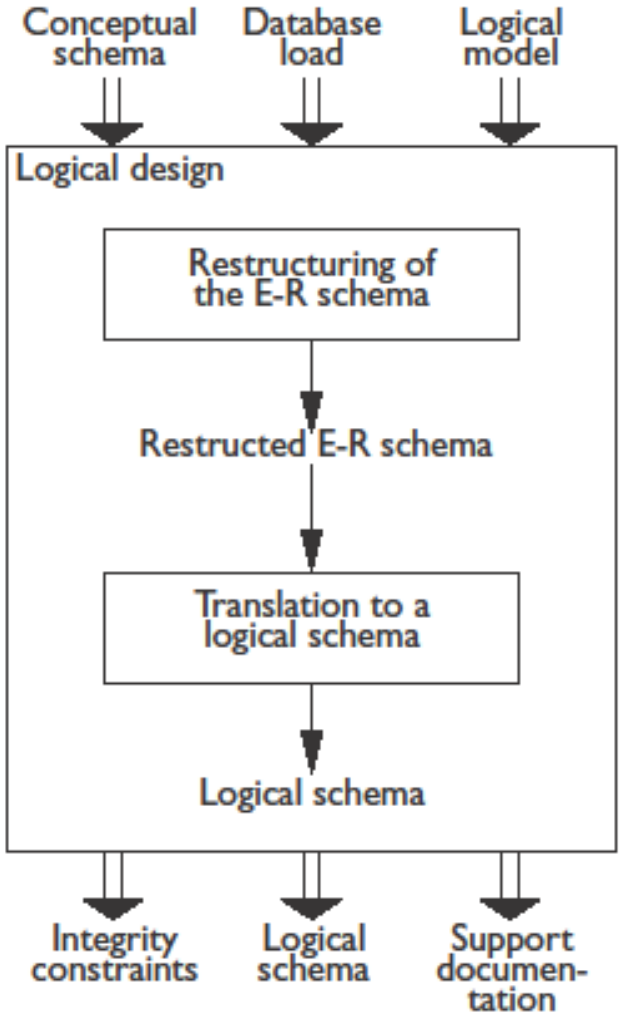
  – the schema must be restructured in such a way as to make the execution of the projected operations as efficient as possible.

Logical design steps

It is usually helpful to divide the logical design into two steps:

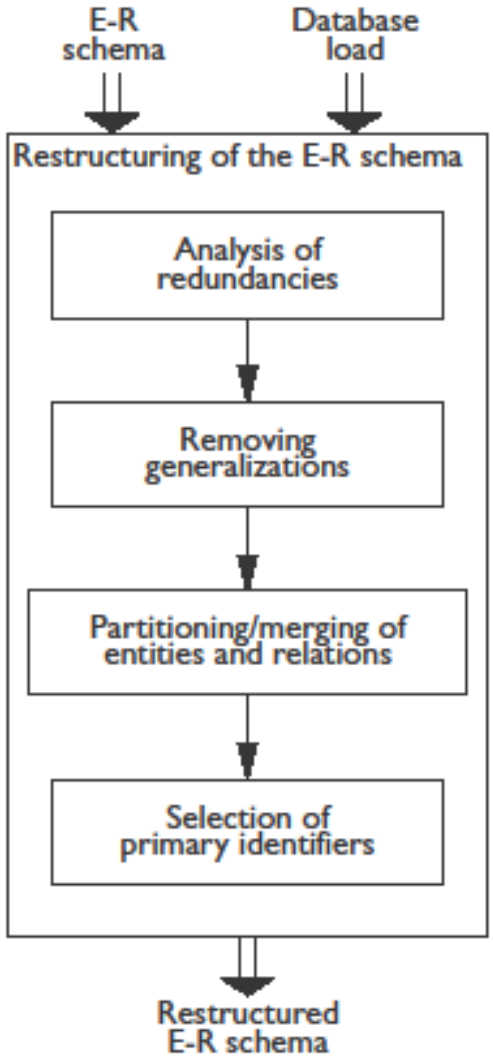
- **restructuring of the Entity-Relationship schema**, based on criteria for the optimization of the schema and the simplification of the following step;
- **translation into the logical model,** based on the features of the logical model (in our case, the relational model).

Logical database design



Restructuring tasks of an E-R schema

The restructuring step of an E-R schema can be sub-divided into a series of tasks

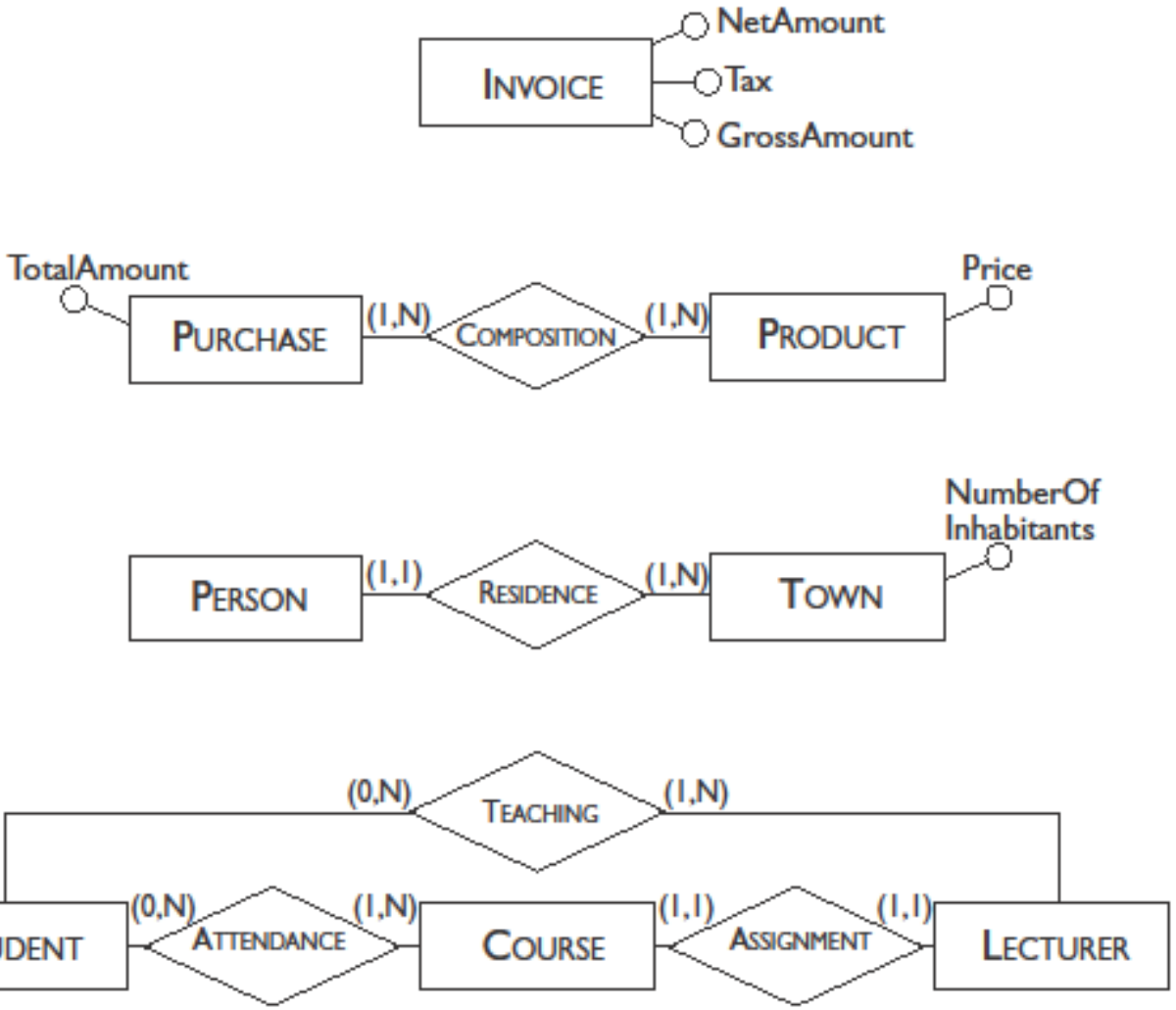


Analysis of redundancies

- A redundancy in a conceptual schema corresponds to a piece of information that can be derived (that is, obtained by a series of retrieval operations) from other data.

- An Entity-Relationship schema can contain various forms of redundancy.

Examples of schemas with redundancies



Taking a decision about redundancies

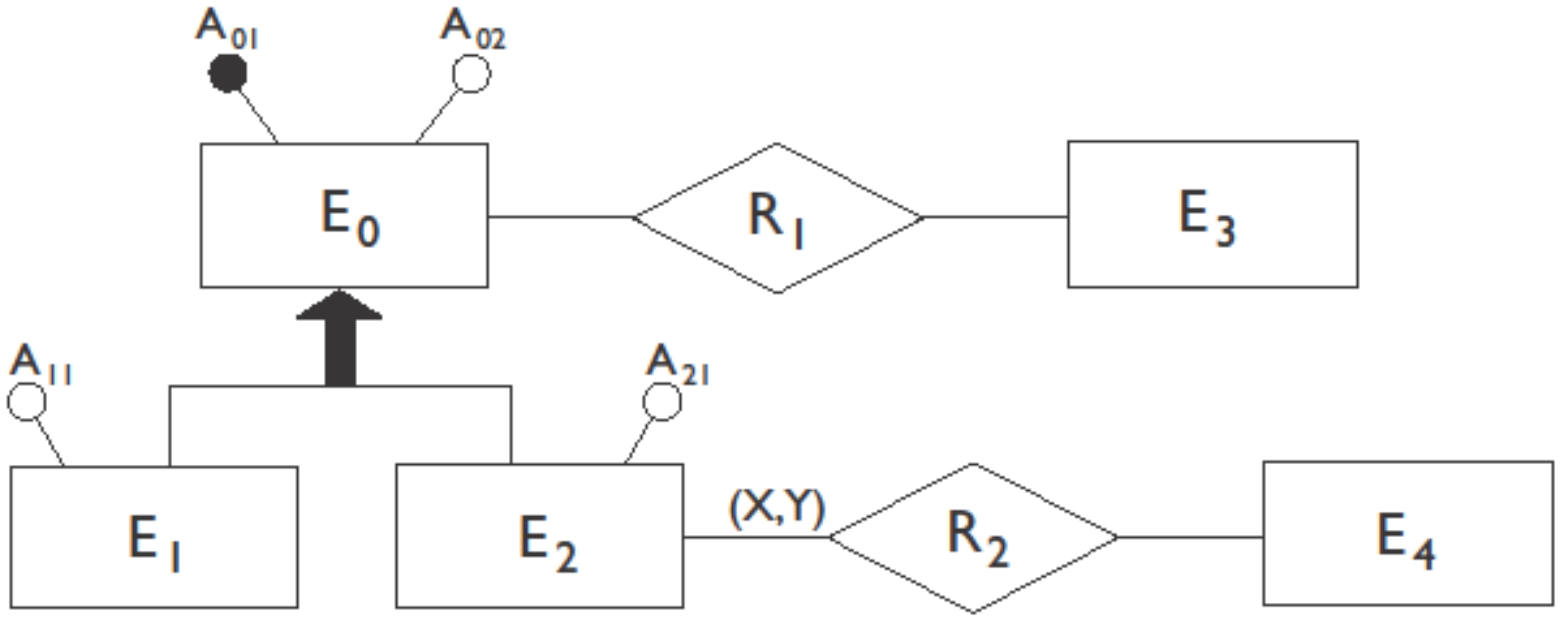
- The presence of a derived piece of information in a database presents
  - an advantage: a reduction in the number of accesses necessary to obtain the derived information;
  - some disadvantages: a larger storage requirement (which is often a negligible cost) and the necessity for carrying out additional operations in order to keep the derived data up to date.
- The decision to maintain or delete a redundancy is made by comparing the cost of operations that involve the redundant information and the storage needed, in the case of presence or absence of redundancy.

Removing generalizations

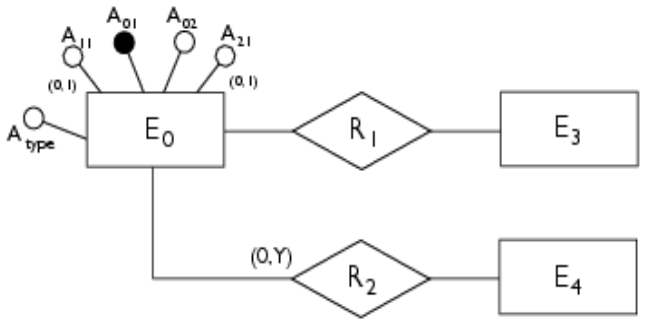
- The relational model does not allow the direct representation of generalizations of the E-R model.

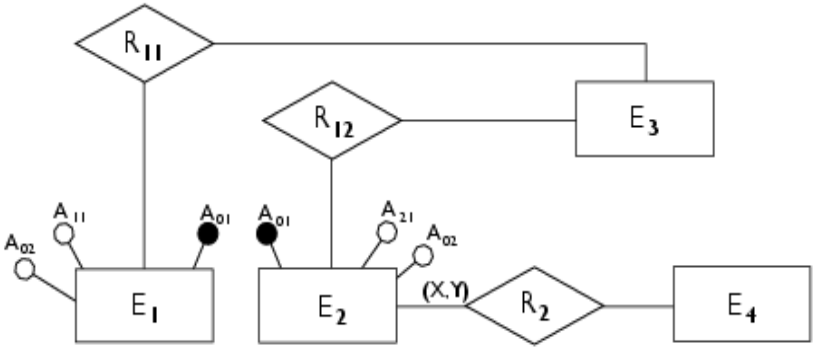
- We need, therefore, to transform these constructs into other constructs that are easier to translate: entities and relationships.

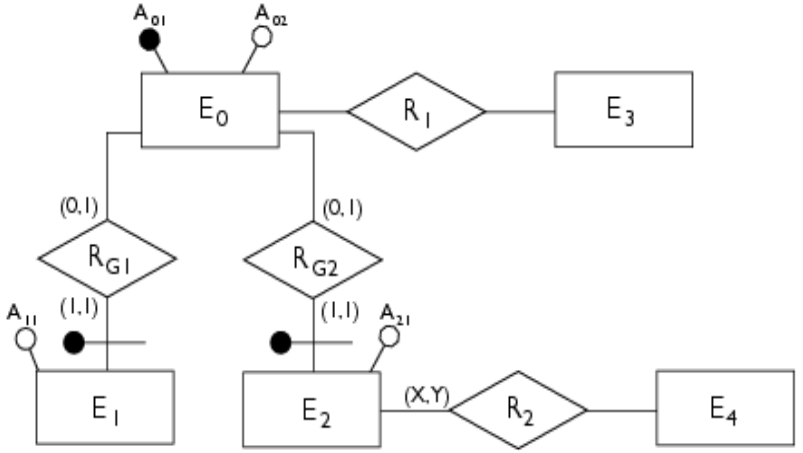
Example of a schema with generalization



Possible restructurings of the previous schema







Notes on Removing Generalizations

1. Collapse child entities into parent entity

   NOTE: Some attributes and relationships will have minimum cardinality = 0

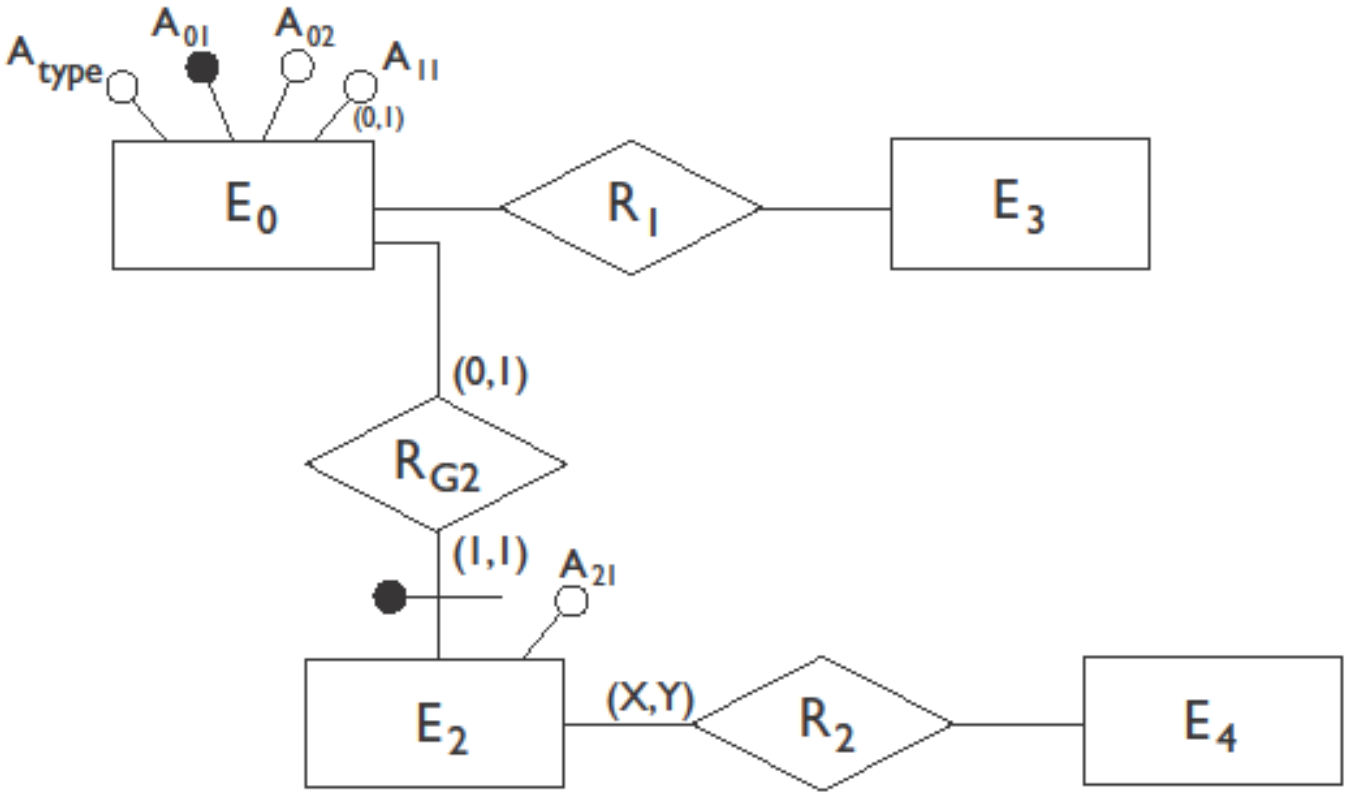
2. Collapse parent entity into child entities: *inheritance*

3. Substitute generalization with *relationships*

General rules about generalization removal

- Option 1 is useful when the operations involve the occurrences and the attributes of E0, E1 and E2 more or less in the same way.
- Option 2 is possible only if the generalization is total and is useful when there are operations that refer only to occurrences of E1 or of E2, and so they make distinctions between these entities.
- Option 3 is useful when the generalization is not total and the operations refer to either occurrences and attributes of E1 (E2) or of E0, and therefore make distinctions between child and parent entities.
- The various options can be combined.

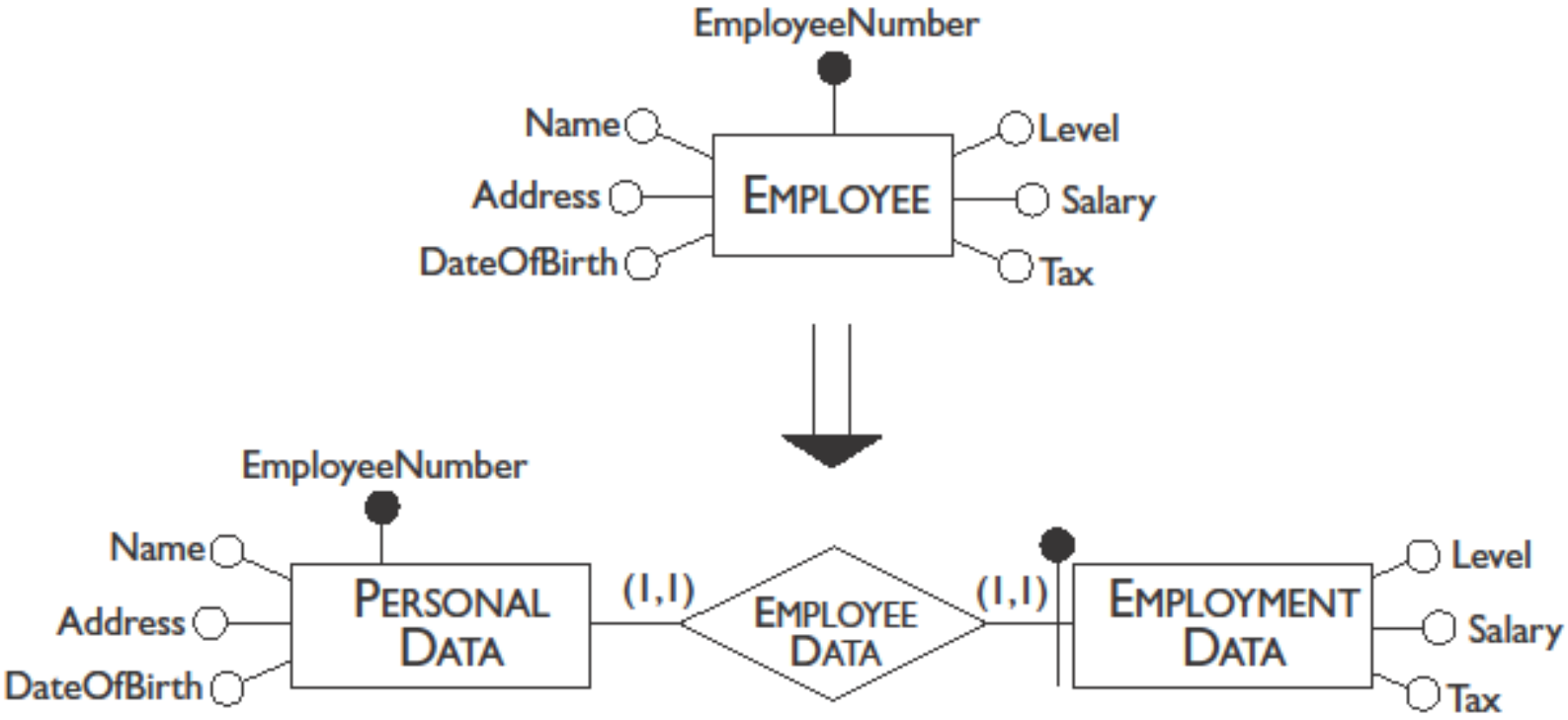
Possible restructuring of the previous schema



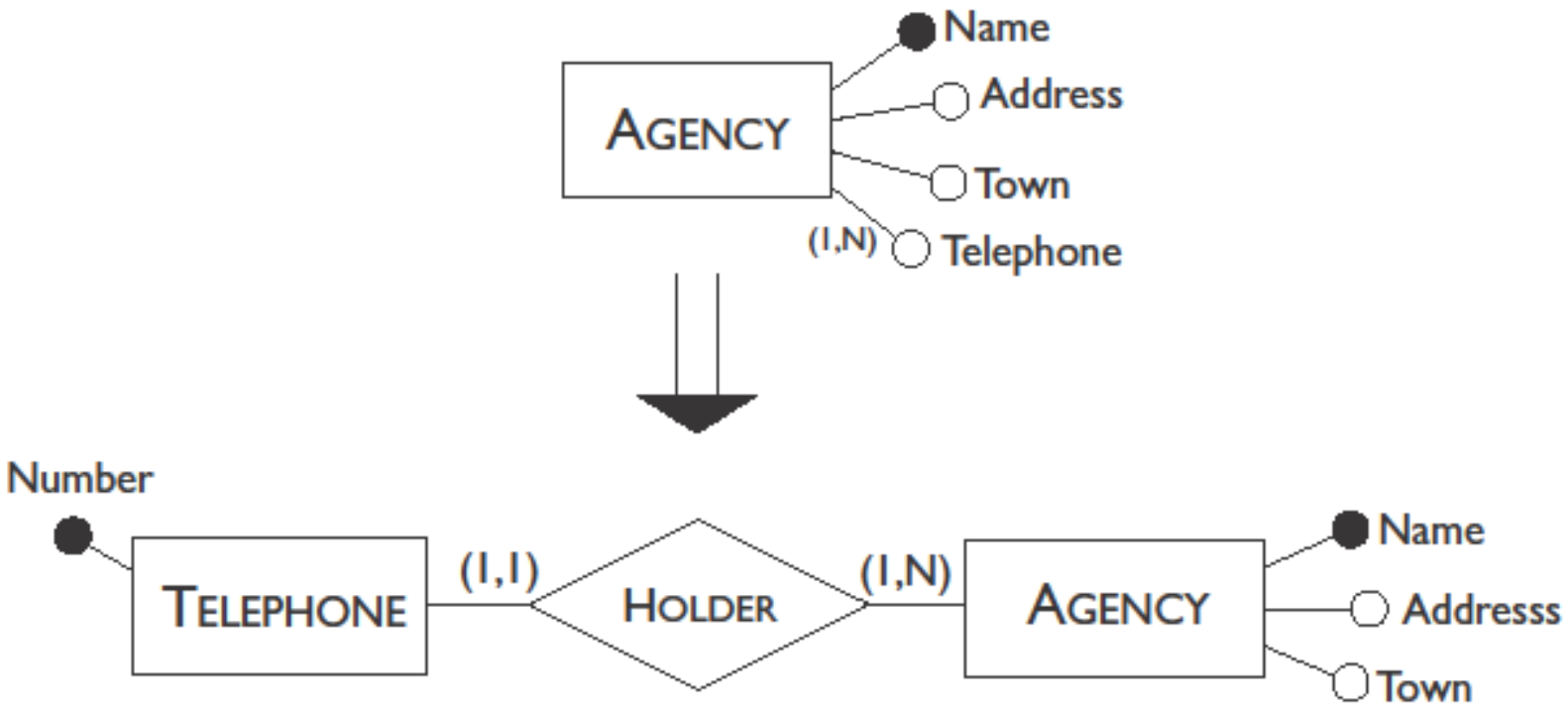
Partitioning and merging of entities and relationships

- Entities and relationships of an E-R schema can be partitioned or merged to improve the efficiency of operations, using the following principle.
  - Accesses are reduced by separating attributes of the same concept that are accessed by different operations and by merging attributes of different concepts that are accessed by the same operations.
- The same criteria as those discussed for redundancies are valid in making a decision about this type of restructuring.

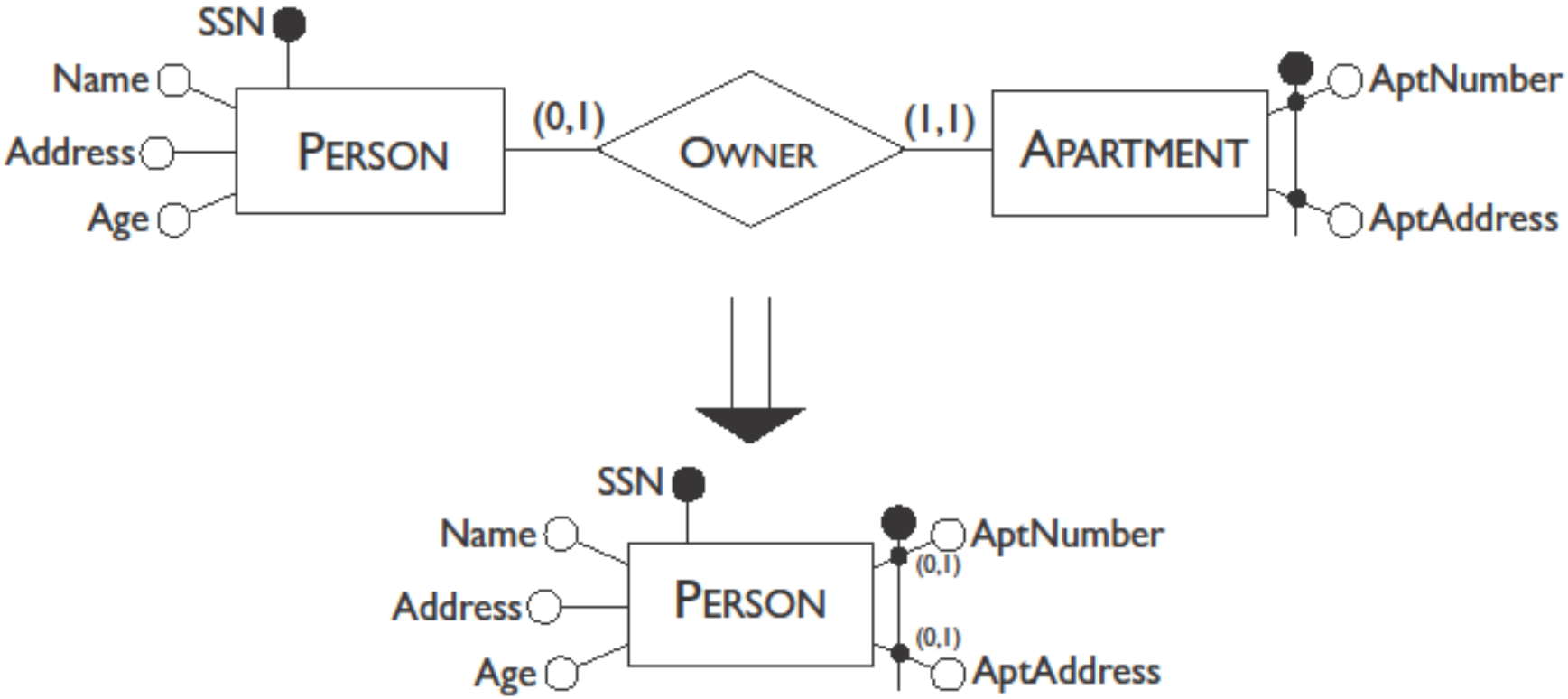
Example of partitioning of entities



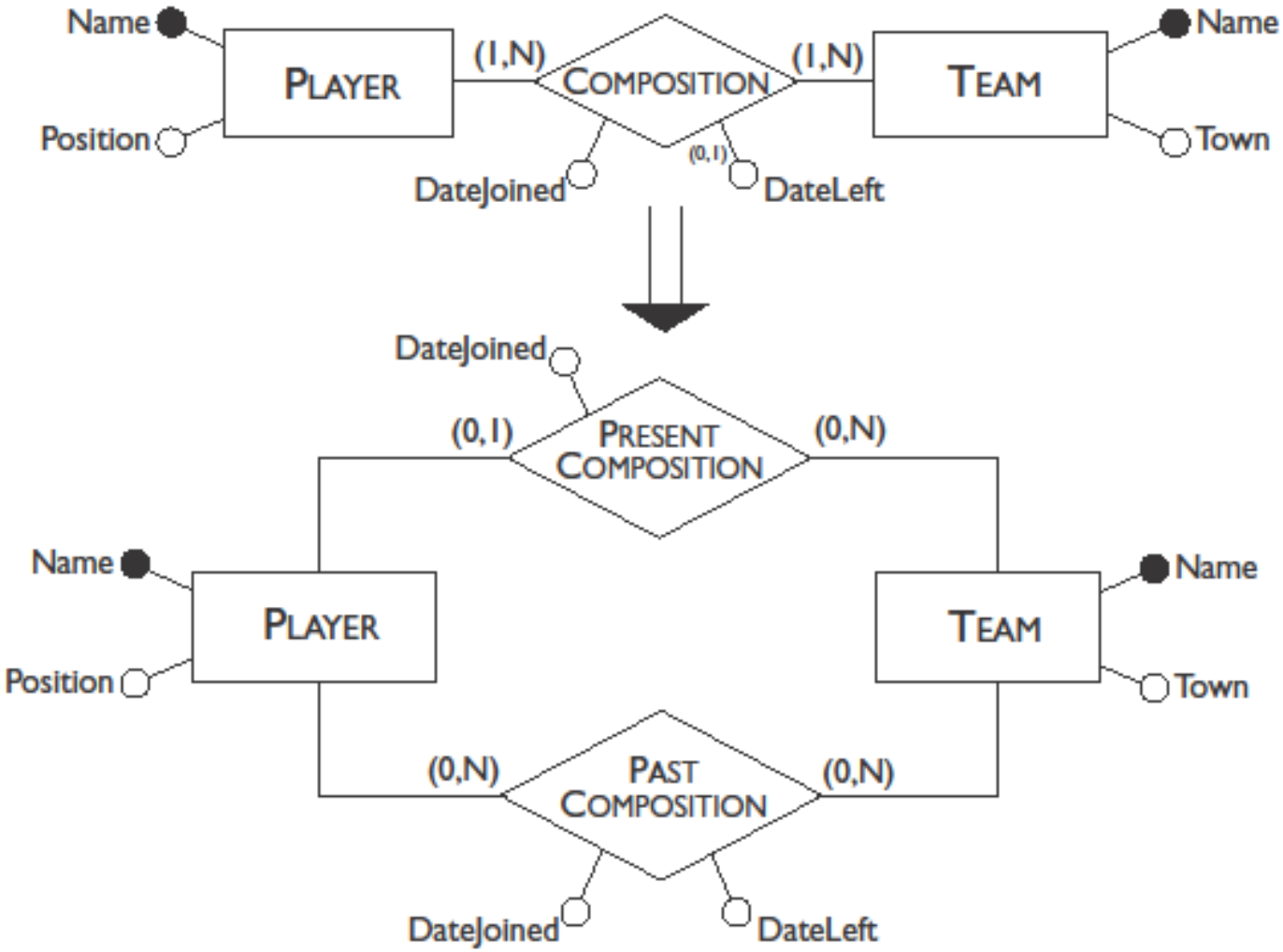
Example of deletion of multi-value attributes



Example of merging of entities



Example of partitioning of a relationship

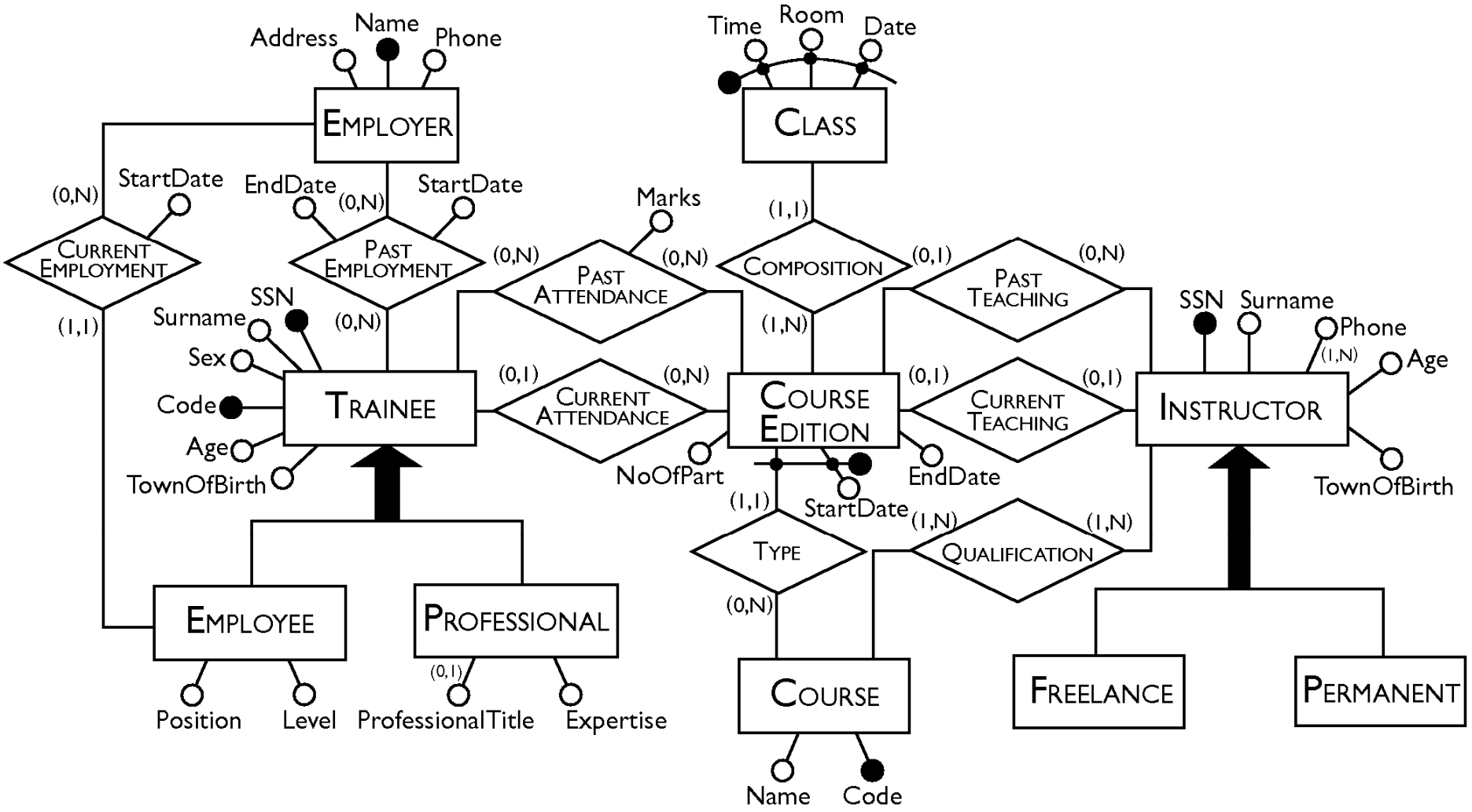


Selection of primary identifiers

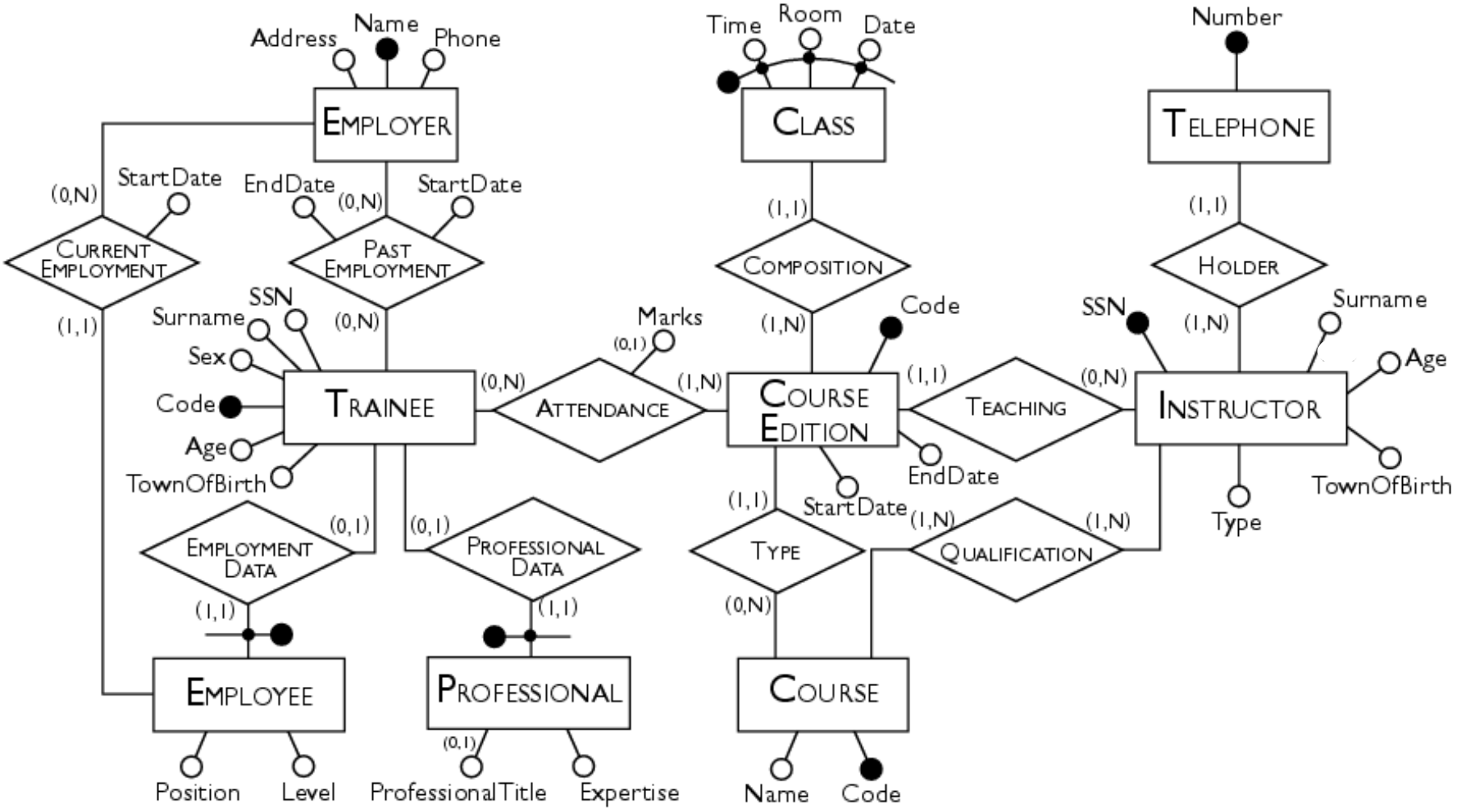
- The criteria for this decision are as follows.
  - Attributes with null values cannot form primary identifiers.
  - One or few attributes are preferable to many attributes.
  - An internal identifier with few attributes is preferable to an external one, possibly involving many entities.
  - An identifier that is used by many operations to access the occurrences of an entity is preferable to others.

- At this stage, if none of the candidate identifiers satisfies the above requirements, it is possible to introduce a further attribute to the entity. This attribute will hold special values (often called *code*s) generated solely for the purpose of identifying occurrences of the entity.

E-R schema of Training Courses Company



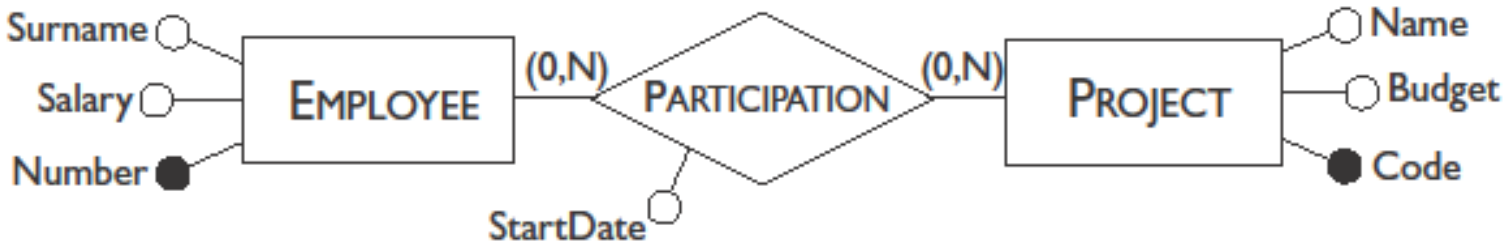
The previous E-R schema after the restructuring phase



Translation into the relational model

- The second step of logical design corresponds to a translation between different data models.
- Starting from an E-R schema, an equivalent relational schema is constructed. By equivalent, we mean a schema capable of representing the same information.
- We will deal with the translation problem systematically, beginning with the fundamental case, that of entities linked by many-to-many relationships.

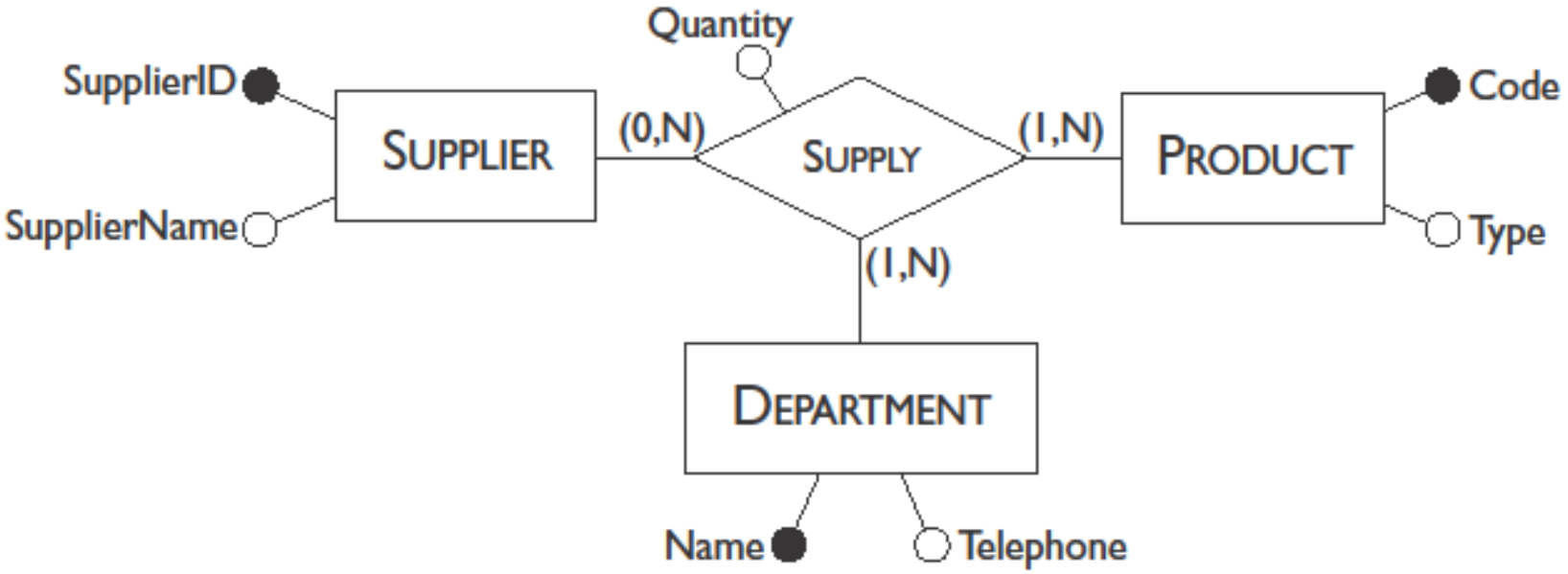
An E-R schema with a many-to-many relationship



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

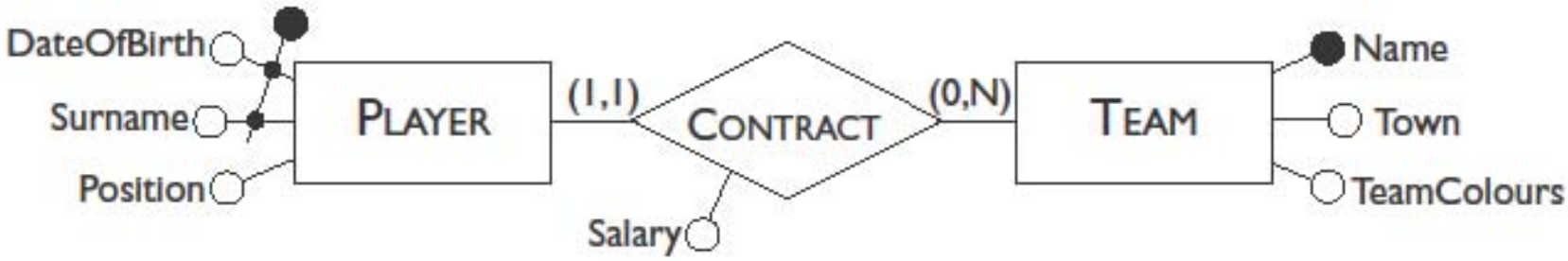
NOTE: Renaming!! PARTICIPATION(Number, Code, StartDate)

E-R schema with ternary relationship



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

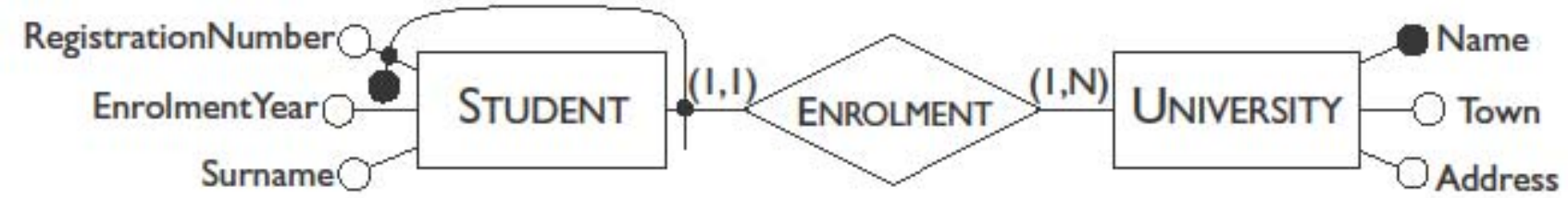
E-R schema with one-to-many relationships



PLAYER(Surname, DateofBirth, Position)
TEAM(Name, Town, TeamColours)
CONTRACT(PlayerSurname, PlayerDateOfBirth, Team, Salary)

Not part of the key

E-R schema with external identifier



STUDENT(<u>RegistrationNumber</u>, <u>University</u>, Surname, EnrolmentYear)
UNIVERSITY(<u>Name</u>, Town, Address)

Referencial constrainst: foreign key!

Note: information about Enrolment relationship is incorporated into table STUDENT

E-R schema with one-to-one relationship

HEAD(<u>Number</u>, Name, Salary, Department, StartDate)
DEPARTMENT(<u>Name</u>, Telephone, Branch)

or

HEAD(<u>Number</u>, Name, Salary)
DEPARTMENT(<u>Name</u>, Telephone, Branch, Head, StartDate)

E-R schema with one-to-one relationship

EMPLOYEE(Number, Name, Salary)
DEPARTMENT(Name, Telephone, Branch, Head, StartDate)

If both the entities have optional participation:

EMPLOYEE(Number, Name, Salary)
DEPARTMENT(Name, Telephone, Branch)
MANAGEMENT(Head, Department, StartDate)