VII. Logical Design

Logical design

Restructuring an ER schema

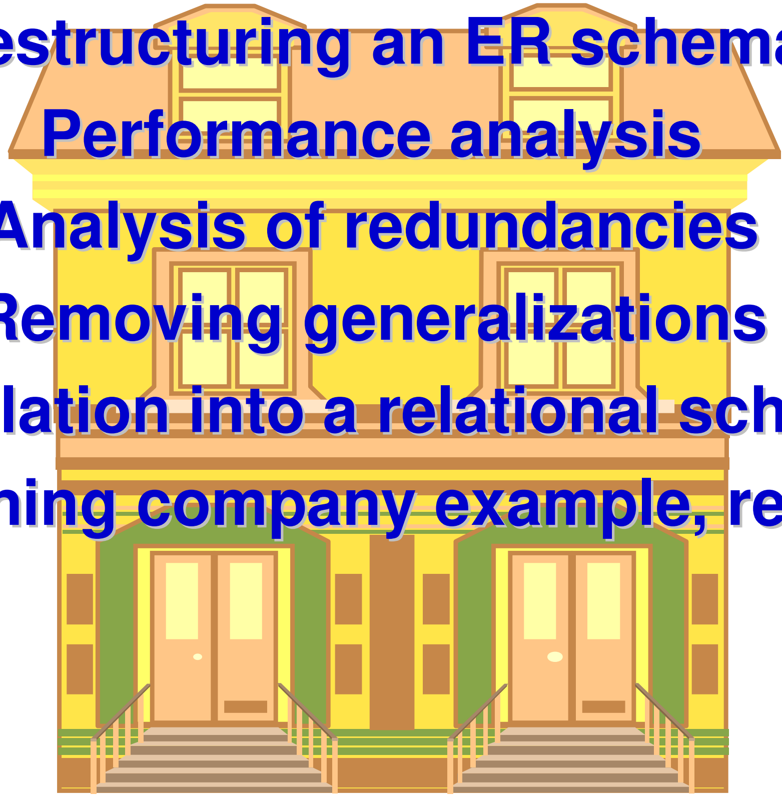
Performance analysis

Analysis of redundancies

Removing generalizations

Translation into a relational schema

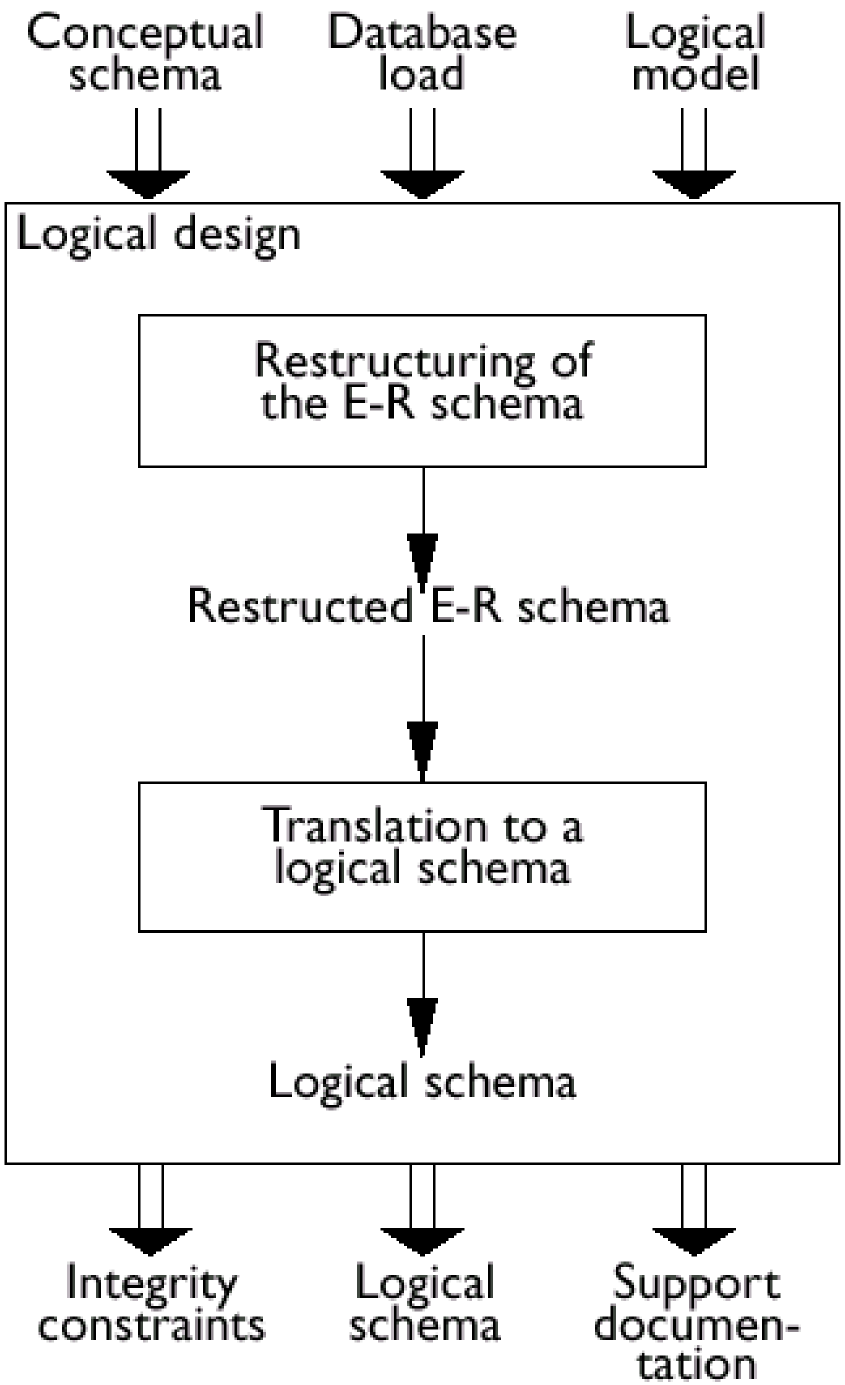
The training company example, revisited



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 Process



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).

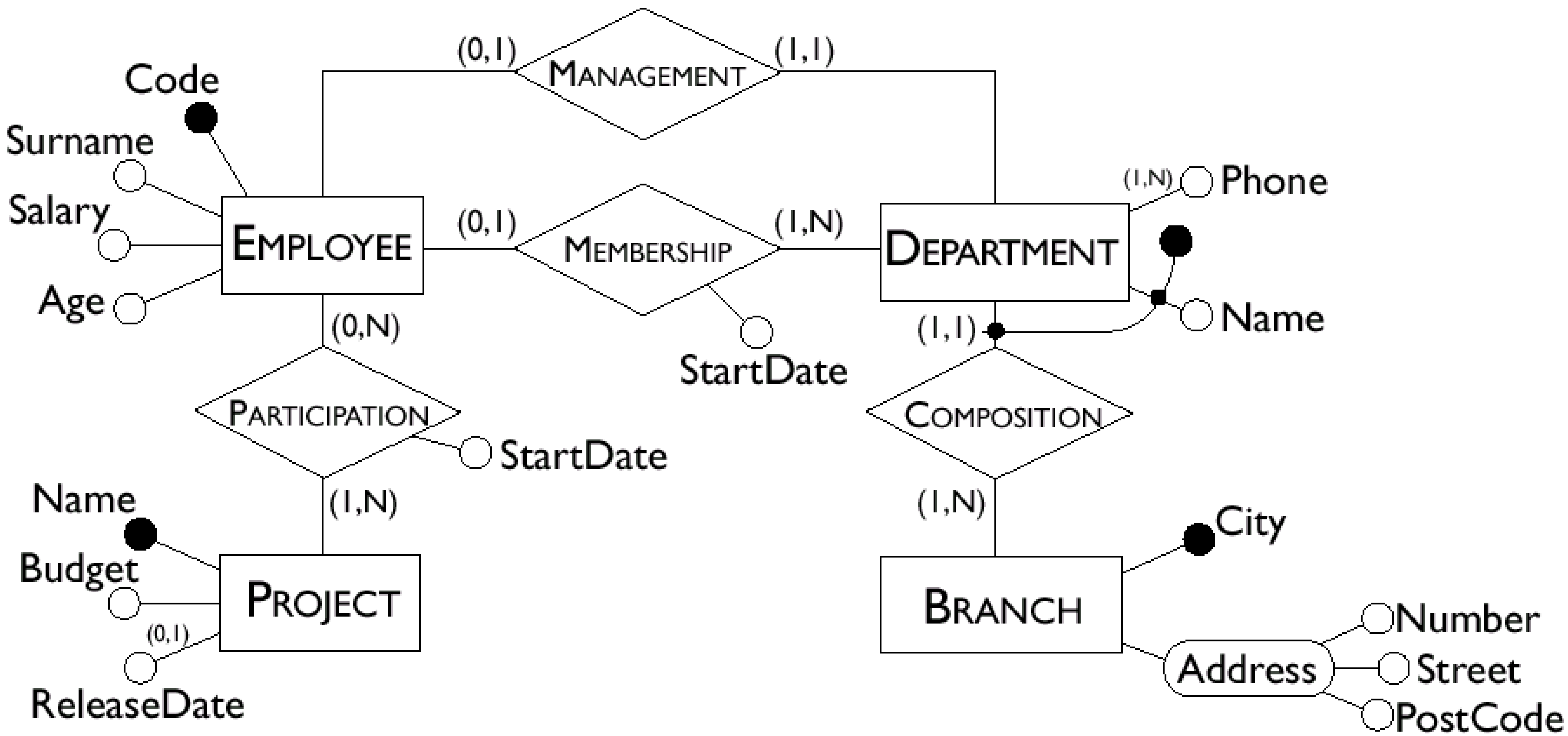
Performance Analysis

- An ER schema can be restructured to optimize two parameters:
  - ✓ **Cost of an operation** (evaluated in terms of the number of occurrences of entities and relationships that are visited to execute an operation on the database);
  - ✓ **Storage requirements** (evaluated in terms of number of bytes necessary to store the data described by the schema).
- In order to study these parameters, we need to know:
  - ✓ the volume of data;
  - ✓ the operation characteristics.

Cost Model

- The cost of an operation is measured in terms of disk accesses required. A disk access is, generally orders of magnitude more expensive than in-memory accesses, or CPU operations.
- For a coarse estimate of cost, we will assume that
  - ✓ a Read operation requires 1 disk access
  - ✓ A Write operation requires 2 disk accesses (read from disk, change, write back to disk)

Employee-Department Example, Again



Typical Operations

- Operation 1: Assign an employee to a project.
- Operation 2: Find the record of an employee, including the department where she works, and the projects she works for.
- Operation 3: Find the records of all employees for a given department.
- Operation 4: For each branch, retrieve its departments, and for each department, retrieve the last names of their managers, and the list of their employees.

Tables of Volumes and Operations

The volume of data and the general characteristics of the operations can be summed up using two special tables.

Table of volumes

Concept	Type	Volume
Branch	E	10
Department	E	80
Employee	E	2000
Project	E	500
Composition	R	80
Membership	R	1900
Management	R	80
Participation	R	6000

Table of operations

Operation	Type	Frequency
Operation 1	I	50 per day
Operation 2	I	100 per day
Operation 3	I	10 per day
Operation 4	B	2 per day

I - Interactive

B - Batch

Navigation Schema for Operation 2

A navigation schema starts from the inputs to an operation and moves (via arrows) towards its outputs.

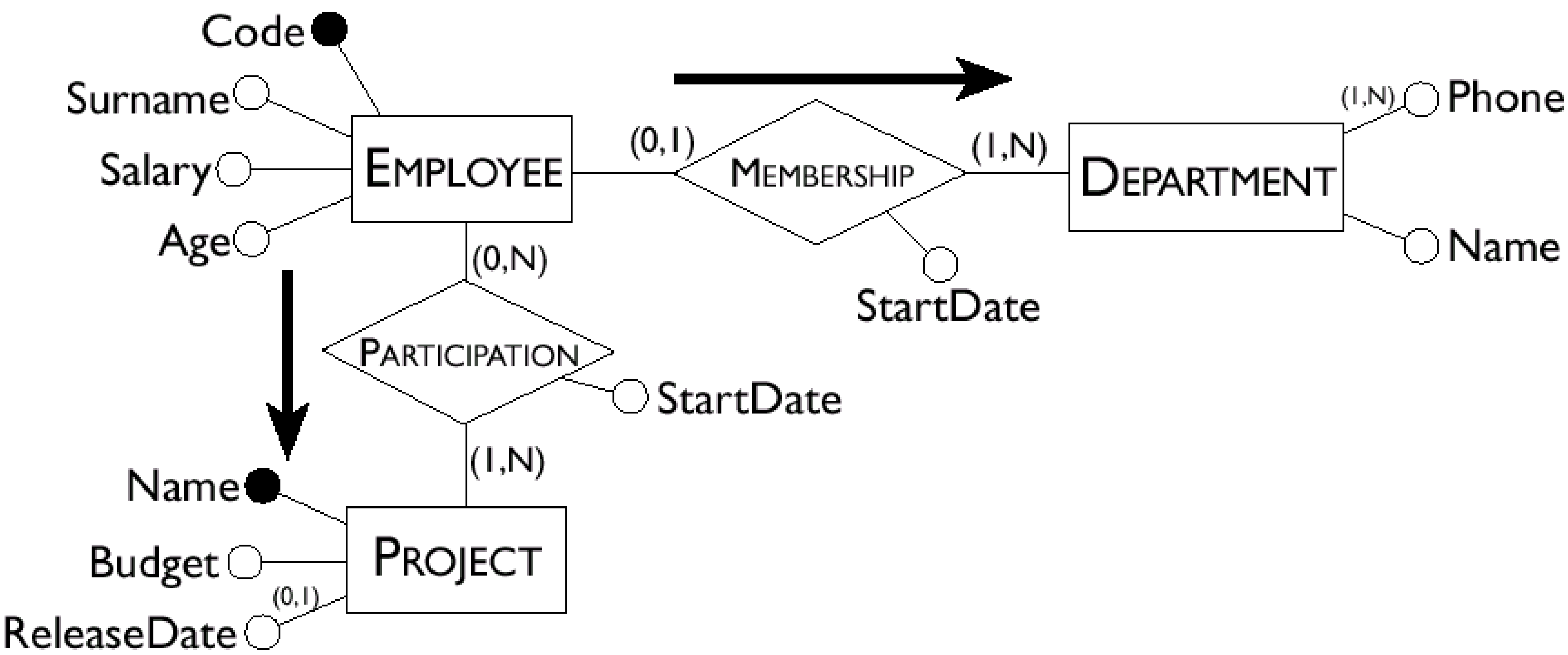


Table of Accesses

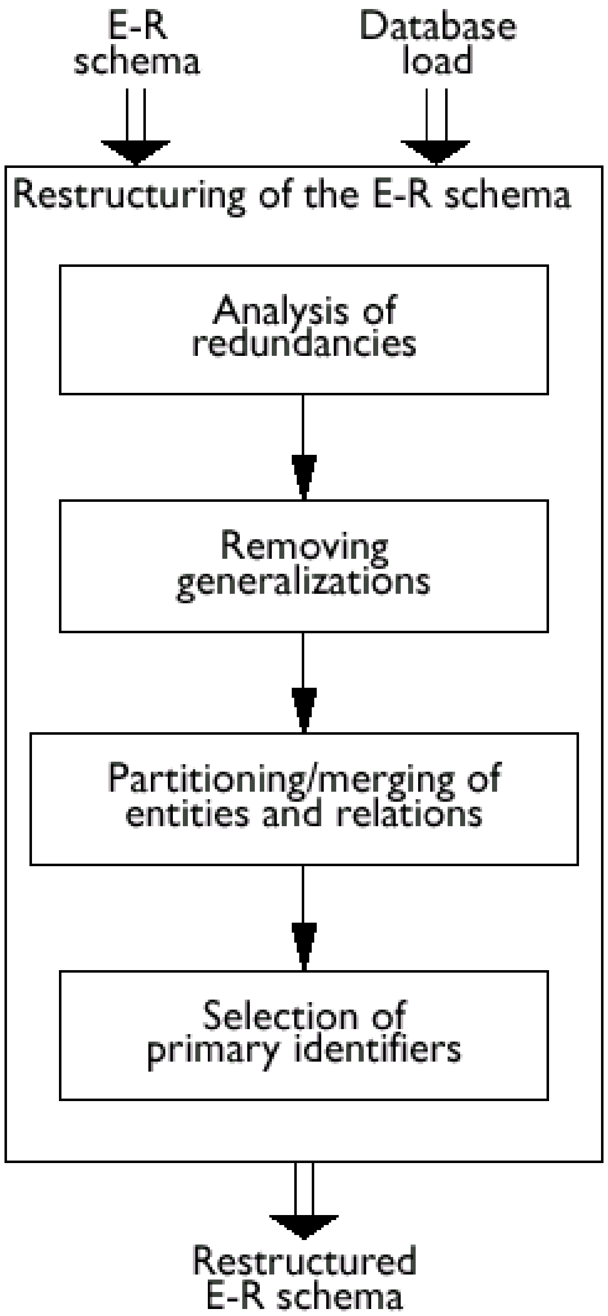
This table evaluates the cost of an operation, using the table of volumes and the navigation schema.

Operation 2

Concept	Type	Accesses	Type
Employee	Entity	1	R
Membership	Relationship	1	R
Department	Entity	1	R
Participation	Relationship	3	R
Project	Entity	3	R

Average # of participations and projects per employee

Type: R - Read. W - Write, RW - Read&Write

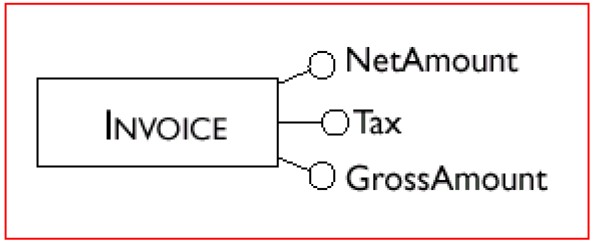


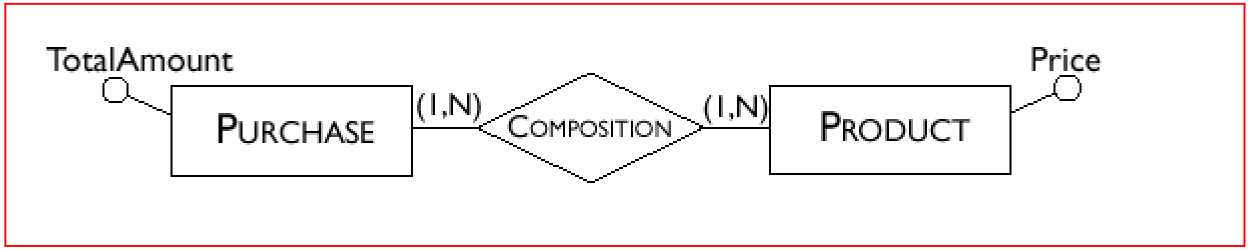
Analysis Steps

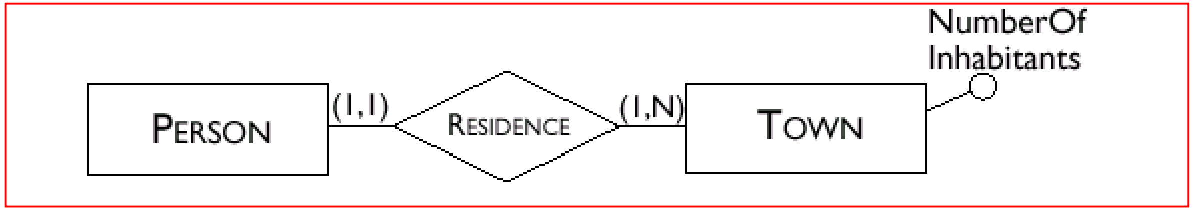
Analysis of Redundancies

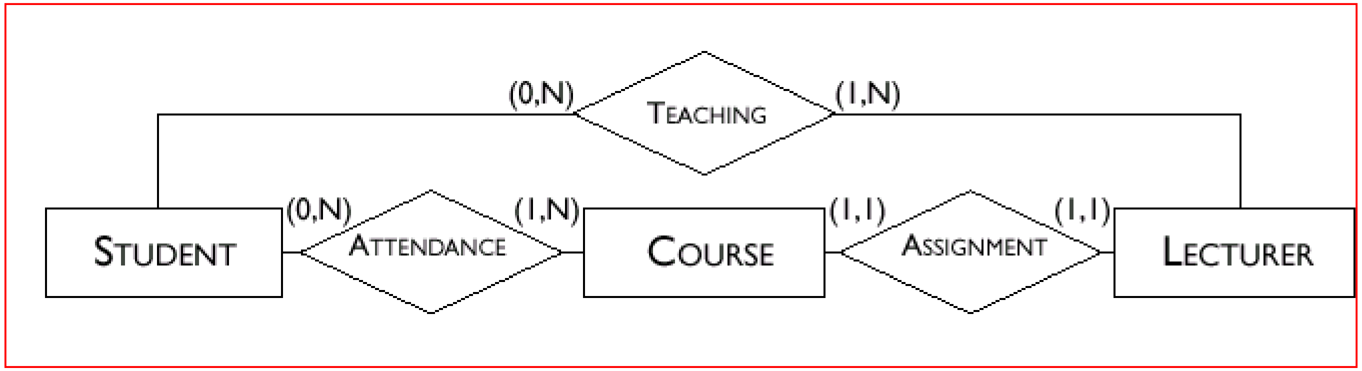
- A redundancy in a conceptual schema corresponds to a piece of information that can be derived (that is, obtained by a series of retrieval operations) from other data in the database.
- An Entity-Relationship schema can contain various forms of redundancy.

Examples of Redundancies





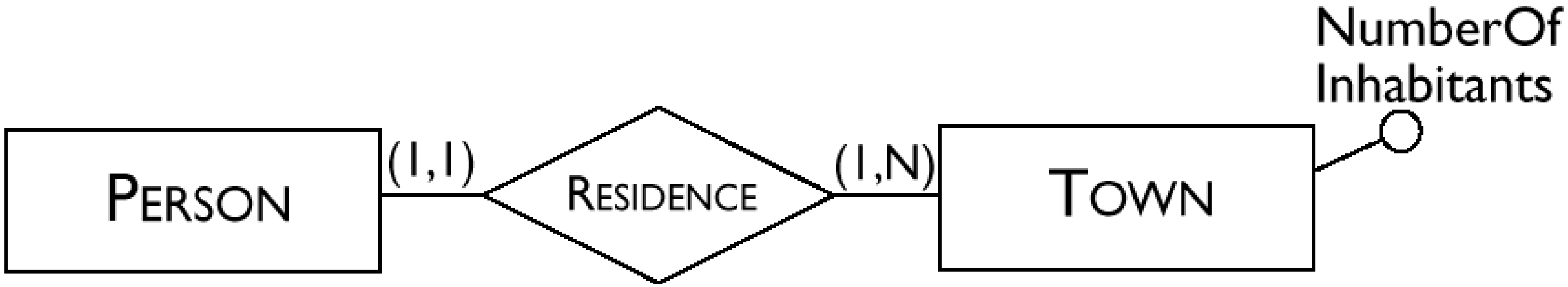




Deciding About Redundancies

- The presence of a redundancy in a database may be
  - ✓ ***an advantage***: a reduction in the number of accesses necessary to obtain the derived information;
  - ✓ ***a disadvantage***: because of larger storage requirements, (but, usually at negligible cost) and the necessity to carry out additional operations in order to keep the derived data consistent.
- 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.

Cost Comparison: An Example



In this schema the attribute NumberOfInhabitants is redundant.

Load and Operations for the Example

Table of volumes

Concept	Type	Volume
Town	E	200
Person	E	1000000
Residence	R	1000000

Table of operations

Operation	Type	Frequency
Operation 1	I	500 per day
Operation 2	I	2 per day

- **Operation 1**: add a new person with the person's town of residence.
- **Operation 2**: print all the data of a town (including the number of inhabitants).

Table of Accesses, with Redundancy

Operation 1

Concept	Type	Accesses	Type
Person	Entity	1	W
Residence	Relationship	1	W
Town	Entity	1	R
Town	Entity	1	W

Operation 2

Concept	Type	Accesses	Type
Town	Entity	1	R

Table of Accesses, without Redundancy

Operation 1

Concept	Type	Accesses	Type
Person	Entity	1	W
Residence	Relationship	1	W

Operation 2

Concept	Type	Accesses	Type
Town	Entity	1	R
Residence	Relationship	5000	R

Comparing the Cost of Operations

Presence of redundancy:

- ✓ Operation 1: 1500 write and 500 read accesses per day.
- ✓ The cost of operation 2 is almost negligible.
- ✓ Counting twice the write accesses, we have a total of 3500 accesses a day.

Absence of redundancy.

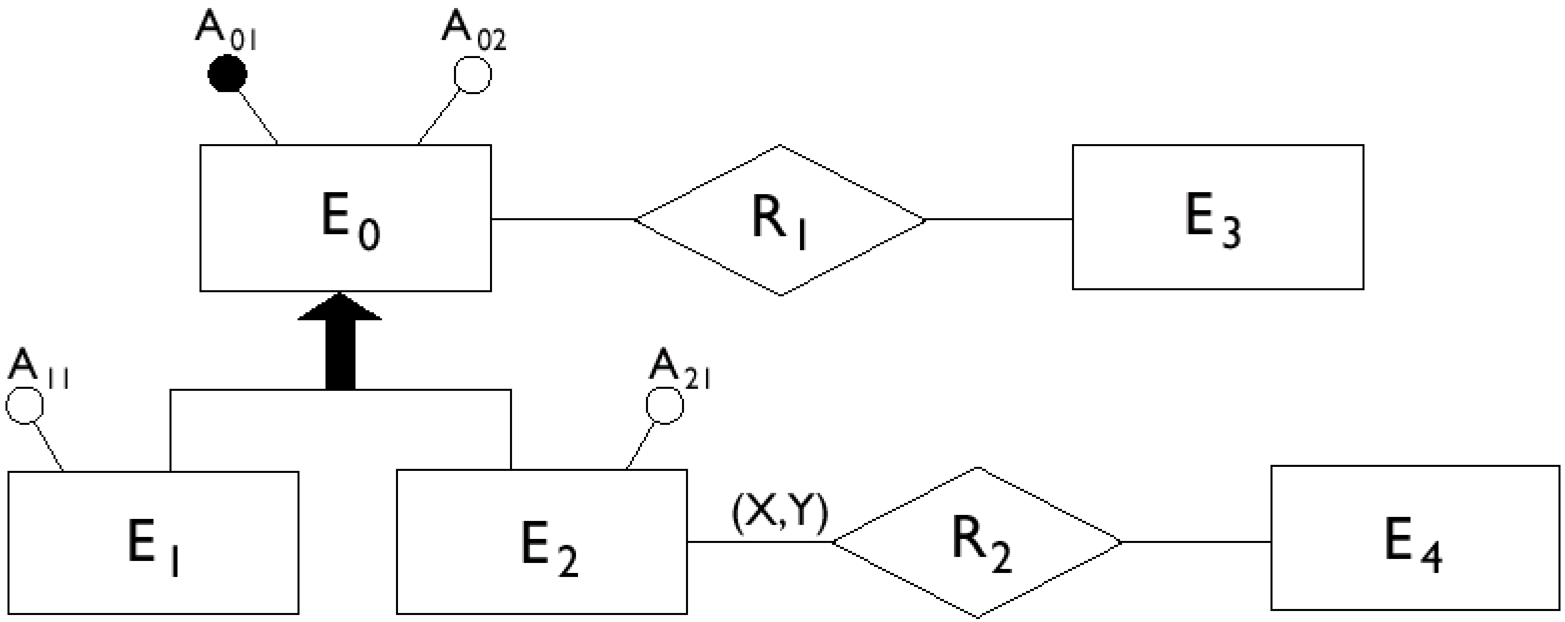
- ✓ Operation 1: 1000 write accesses per day.
- ✓ Operation 2 however requires a total of 10000 read accesses per day.
- ✓ Counting twice the write accesses, we have a total of 12000 accesses per day.

Redundant data can improve performance, sometimes!

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.

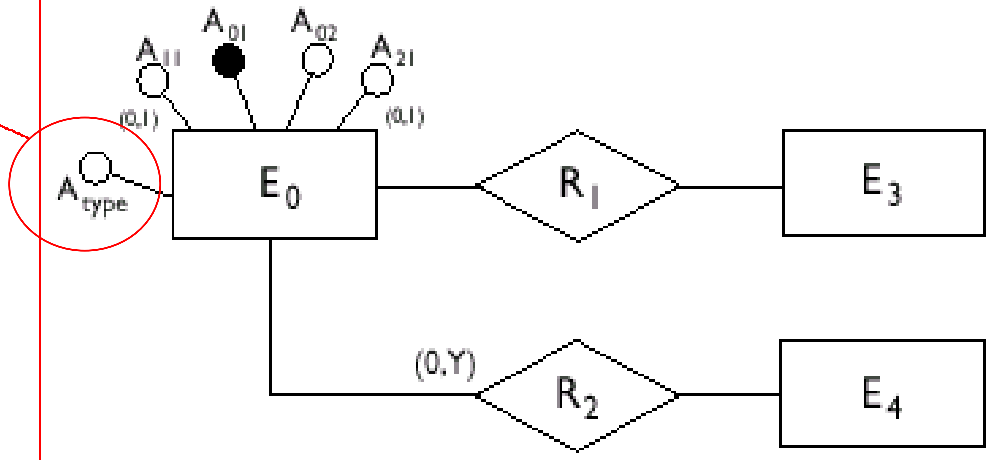
A Schema with Generalizations



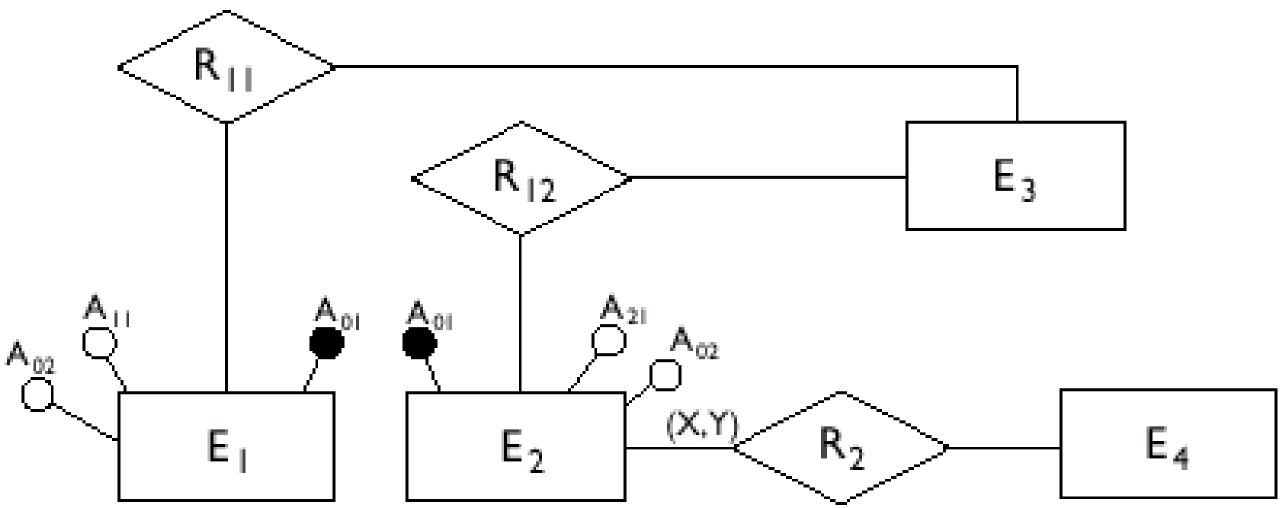
Possible Restructurings

Note!

Option 1

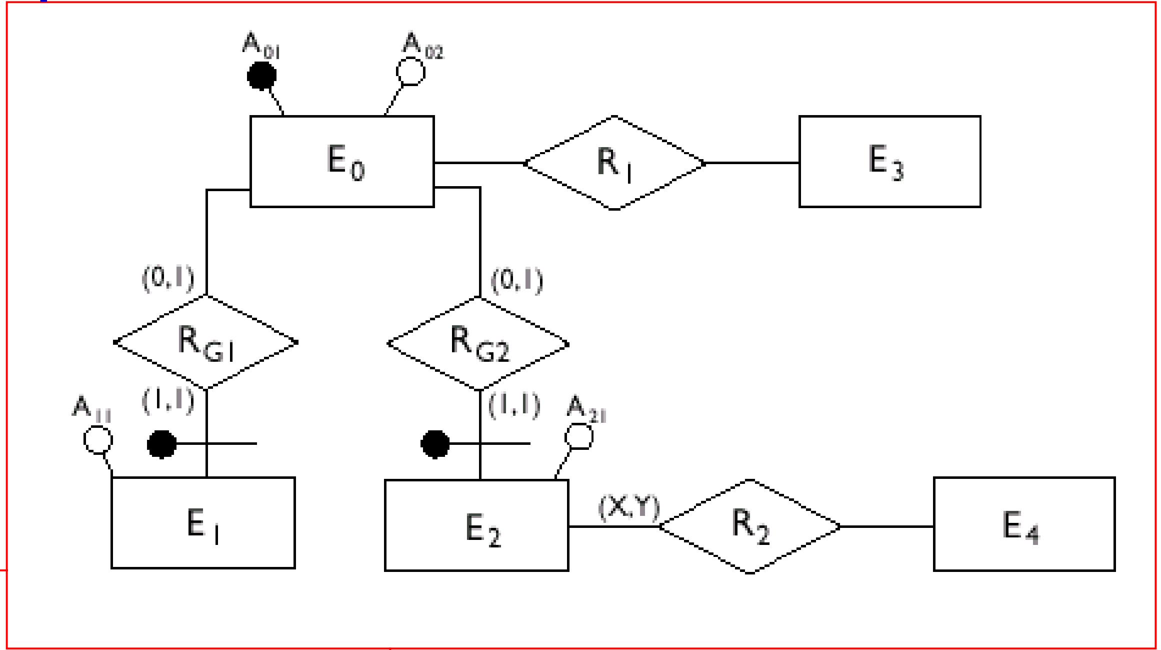


Option 2

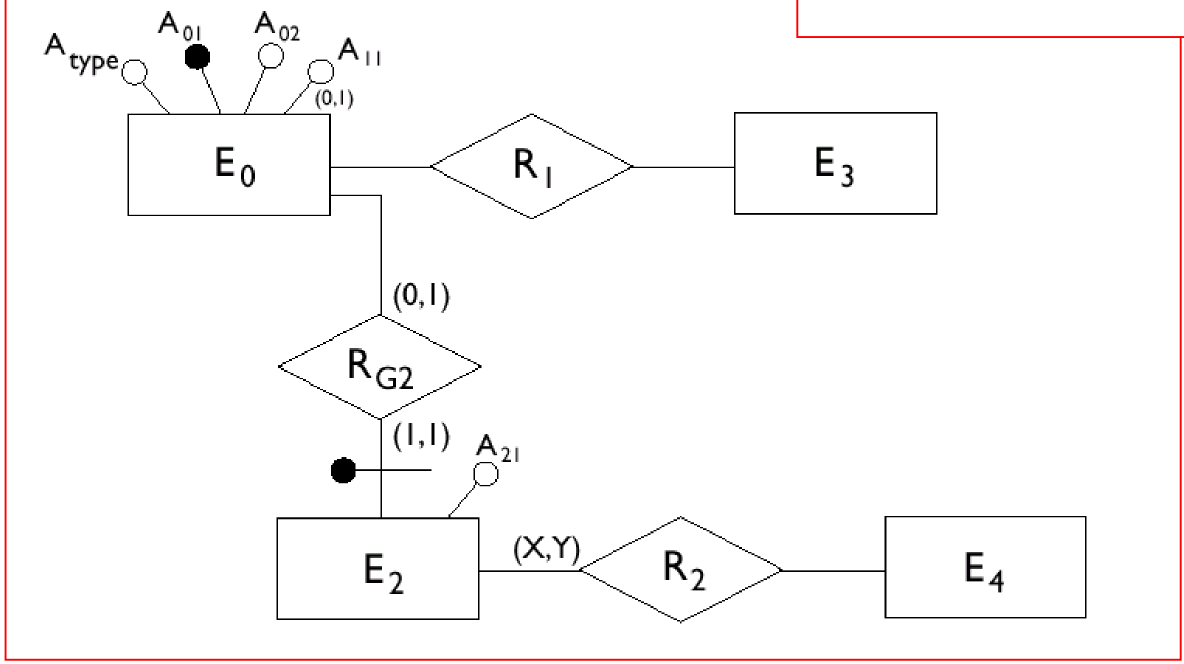


...Two More...

Option 3



Option 4 (combination)



General Rules For Removing Generalization

- Option 1 is convenient when the operations involve the occurrences and the attributes of E_0, E_1 and E_2 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 apply only to occurrences of E_1 or of E_2.
- Option 3 is useful when the generalization is not total and the operations refer to either occurrences and attributes of E_1 (E_2) or of E_0, and therefore make distinctions between child and parent entities.
- Available options can be combined (see option 4)

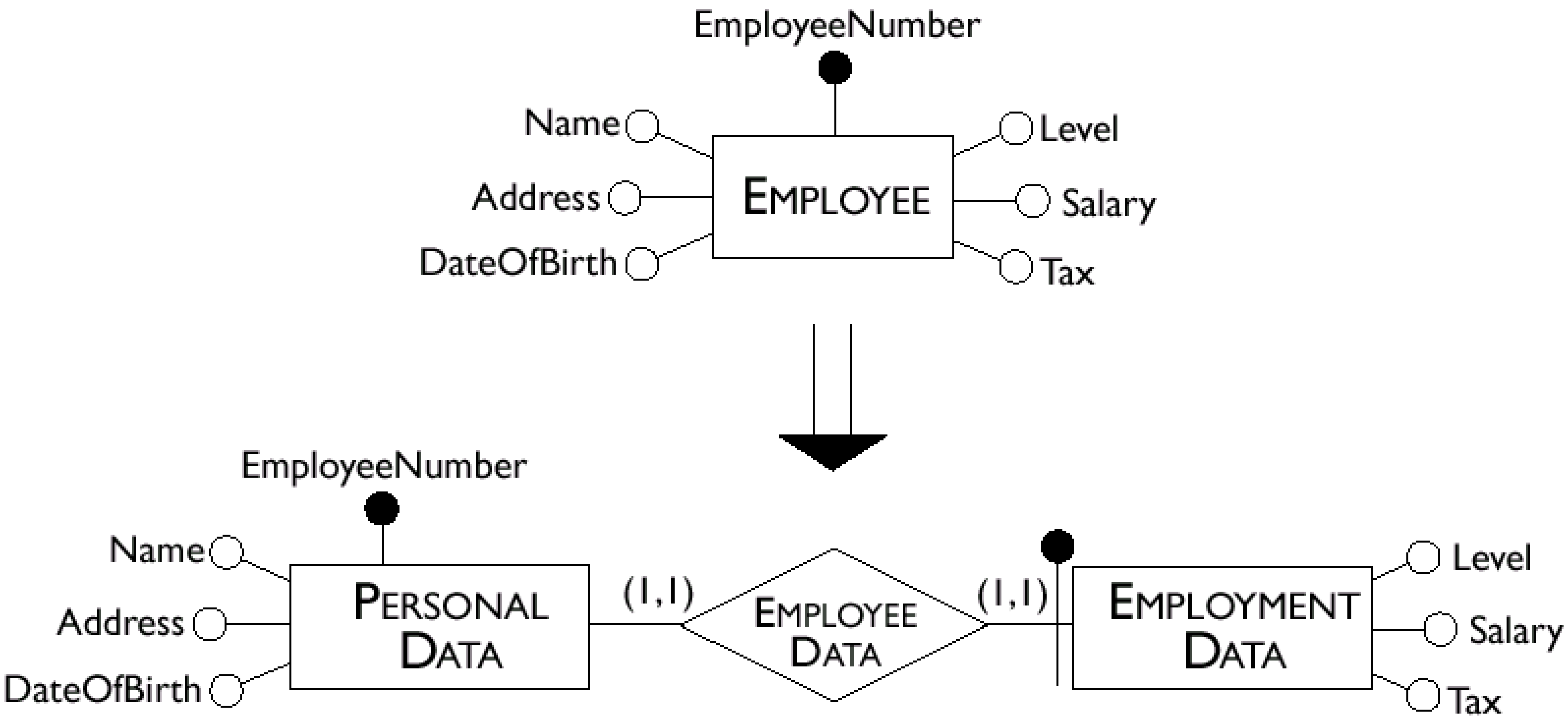
Partitioning and Merging of Entities and Relationships

- Entities and relationships of an E-R schema can be partitioned or merged to improve the efficiency of operations, using the following principle.

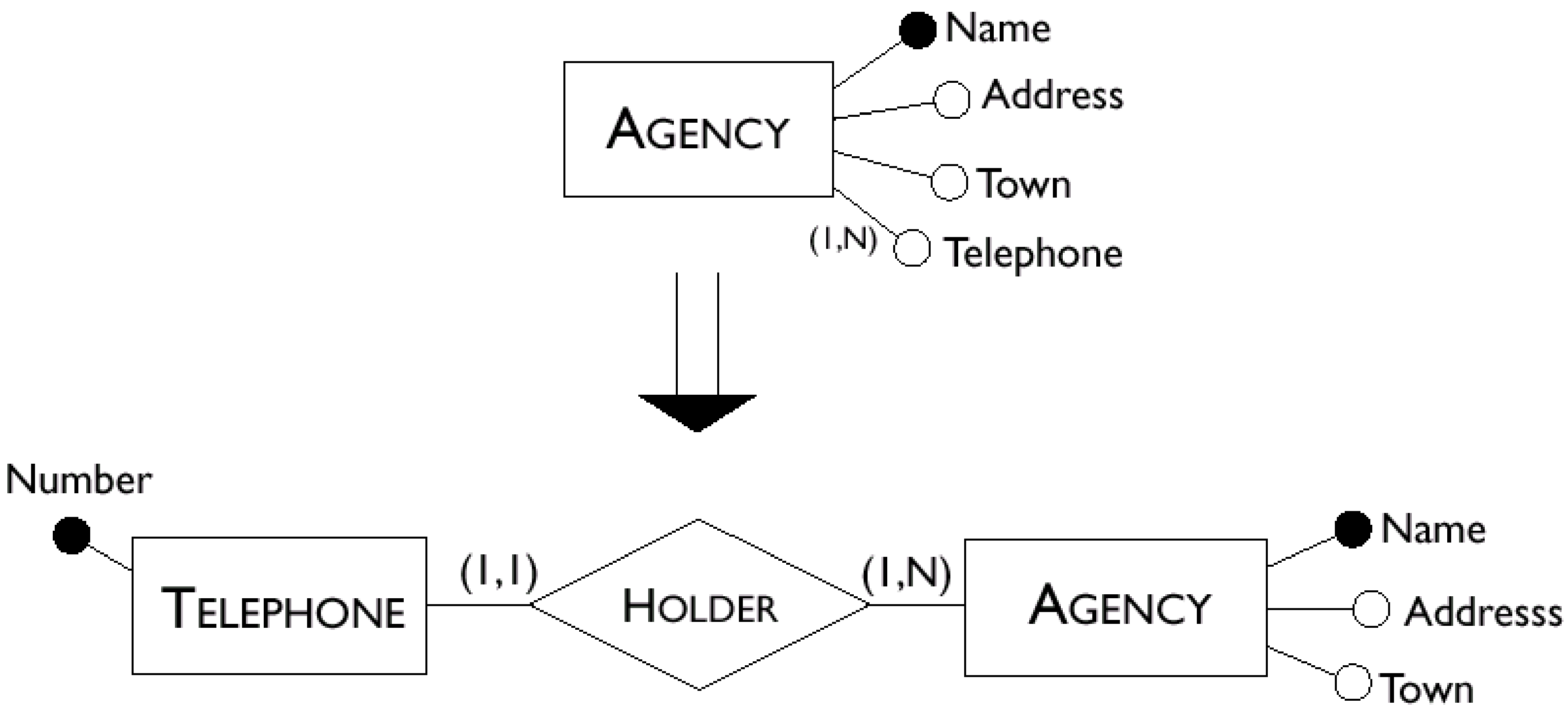
Accesses are reduced by separating attributes of the same concept that are accessed by different operations and by merging attributes of different concepts that are accessed by the same operations.

- The same criteria as those discussed for redundancies are valid in making a decision about this type of restructuring.

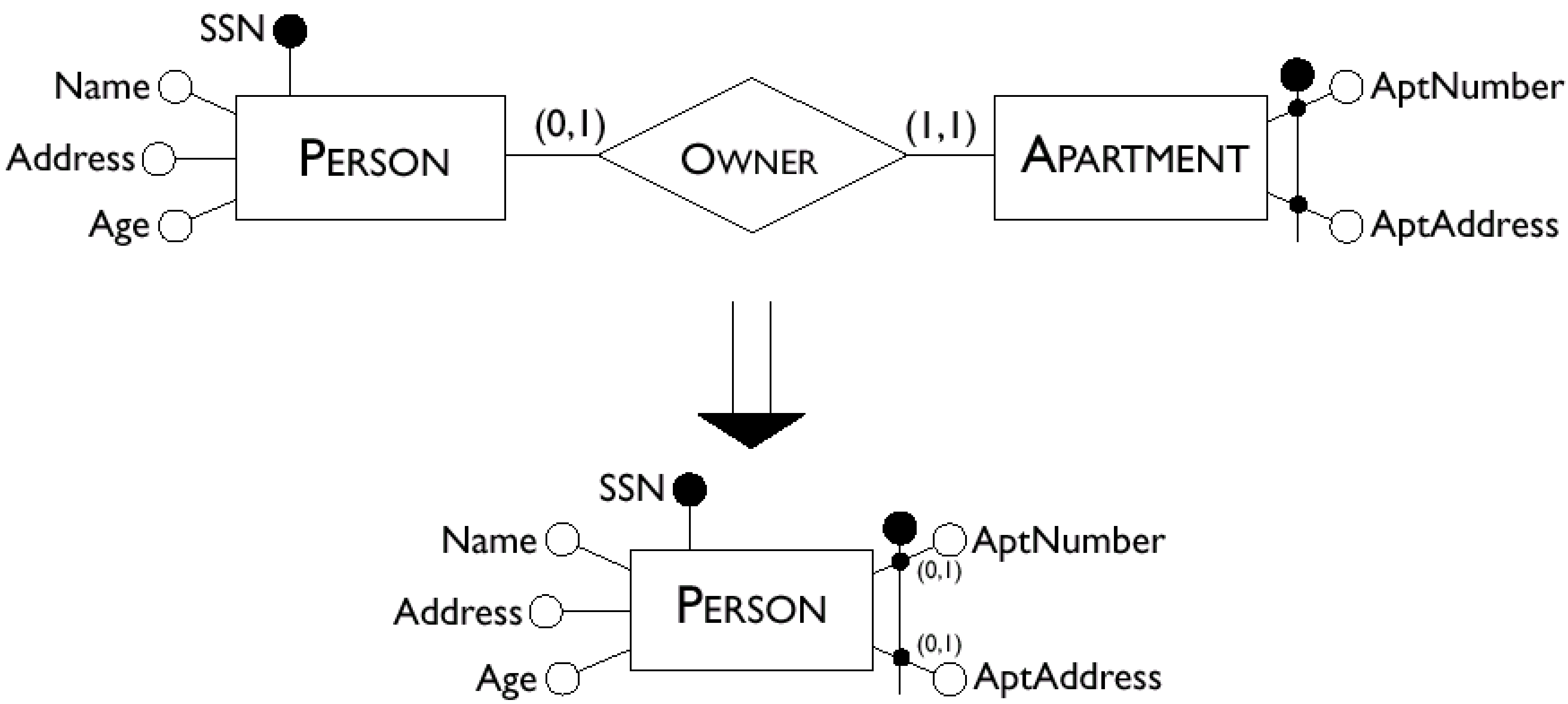
Example of Partitioning



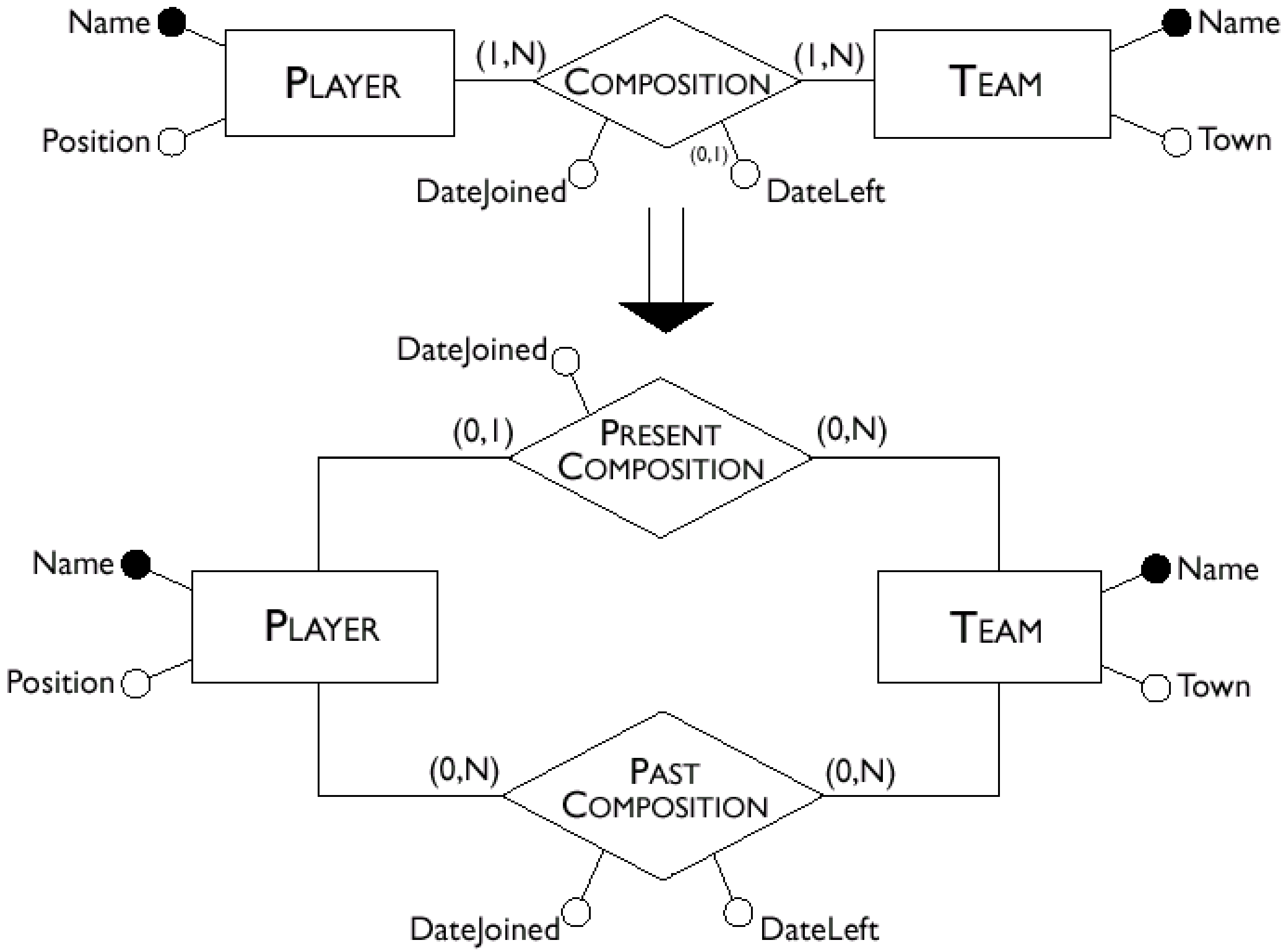
Deletion of Multi-Valued Attribute



Merging of Entities



Partitioning of a Relationship



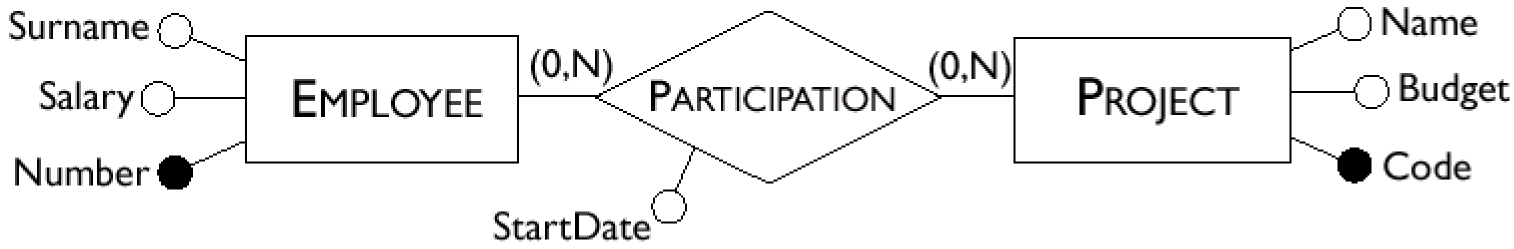
Selection of Primary Identifiers

- 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 *codes*) generated solely for the purpose of identifying occurrences of the entity.

Translation into a Logical Schema

- 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.

Many-to-Many Relationships

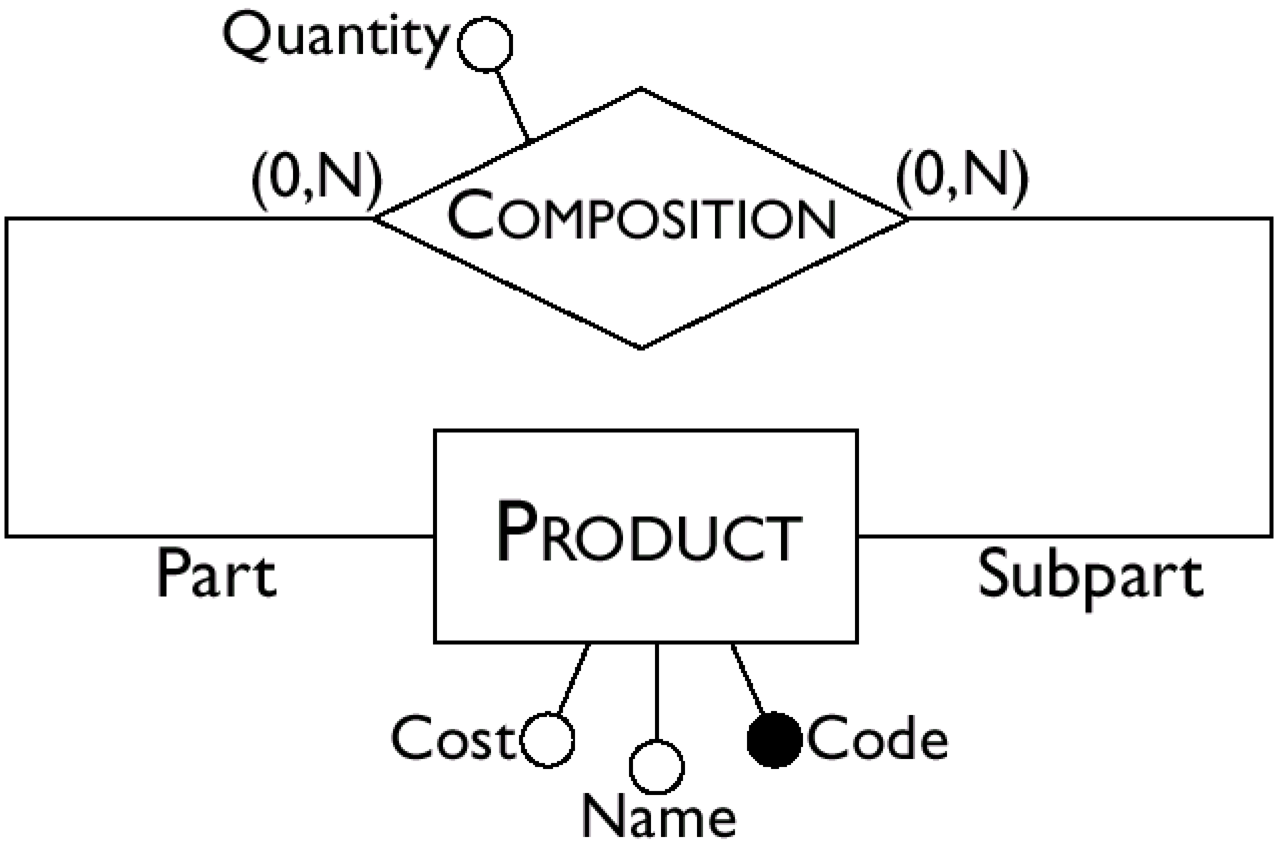


Employee(<u>Number</u>, Surname, salary)

Project(<u>Code</u>, Name, Budget)

Participation(<u>Number</u>, <u>Code</u>, StartDate)

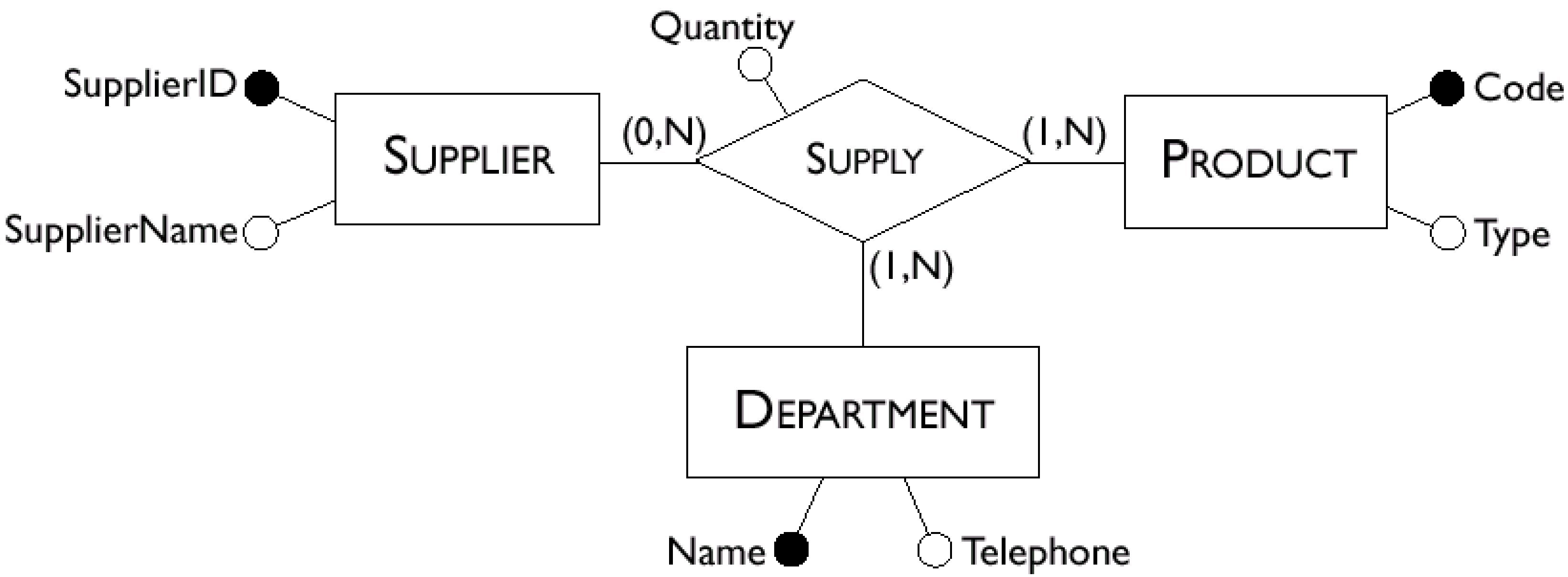
Recursive Relationships



Product(<u>Code</u>, Name, Cost)

Composition(<u>Part</u>, <u>SubPart</u>, Quantity)

Ternary Relationships



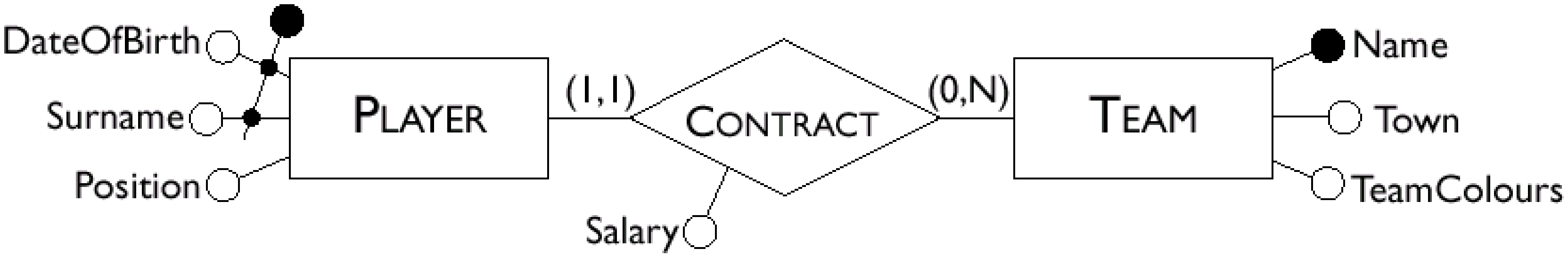
Supplier(<u>SupplierID</u>, SupplierName)

Product(<u>Code</u>, Type)

Department(<u>Name</u>, Telephone)

Supply(<u>Supplier</u>, <u>Product</u>, Department, Quantity)

One-to-Many Relationships

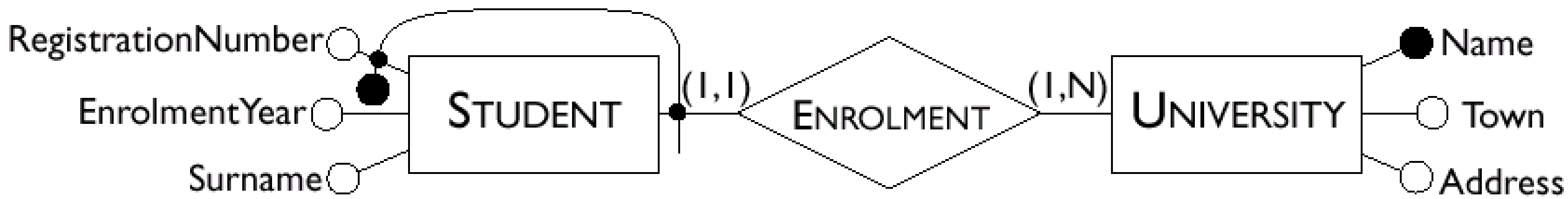


Player(<u>Surname</u>,<u>DateOfBirth</u>, Position)

Team(<u>Name</u>, Town, TeamColours)

Contract(<u>PlayerSurname</u>, <u>PlayerDateOfBirth</u>, Team, Salary)

Entities with External Identifiers



Student(<u>RegistrationNumber</u>, <u>University</u>, Surname, EnrolmentYear)

University(<u>Name</u>, Town, Address)

One-to-One Relationships

Head(<u>Number</u>, Name, Salary, Department, StartDate)

Department(<u>Name</u>, Telephone, Branch)

Or

Head(<u>Number</u>, Name, Salary, StartDate)

Department(<u>Name</u>, Telephone, HeadNumber, Branch)

Optional One-to-One Relationships

Employee(<u>Number</u>, Name, Salary)

Department(<u>Name</u>, Telephone, Branch, Head, StartDate)

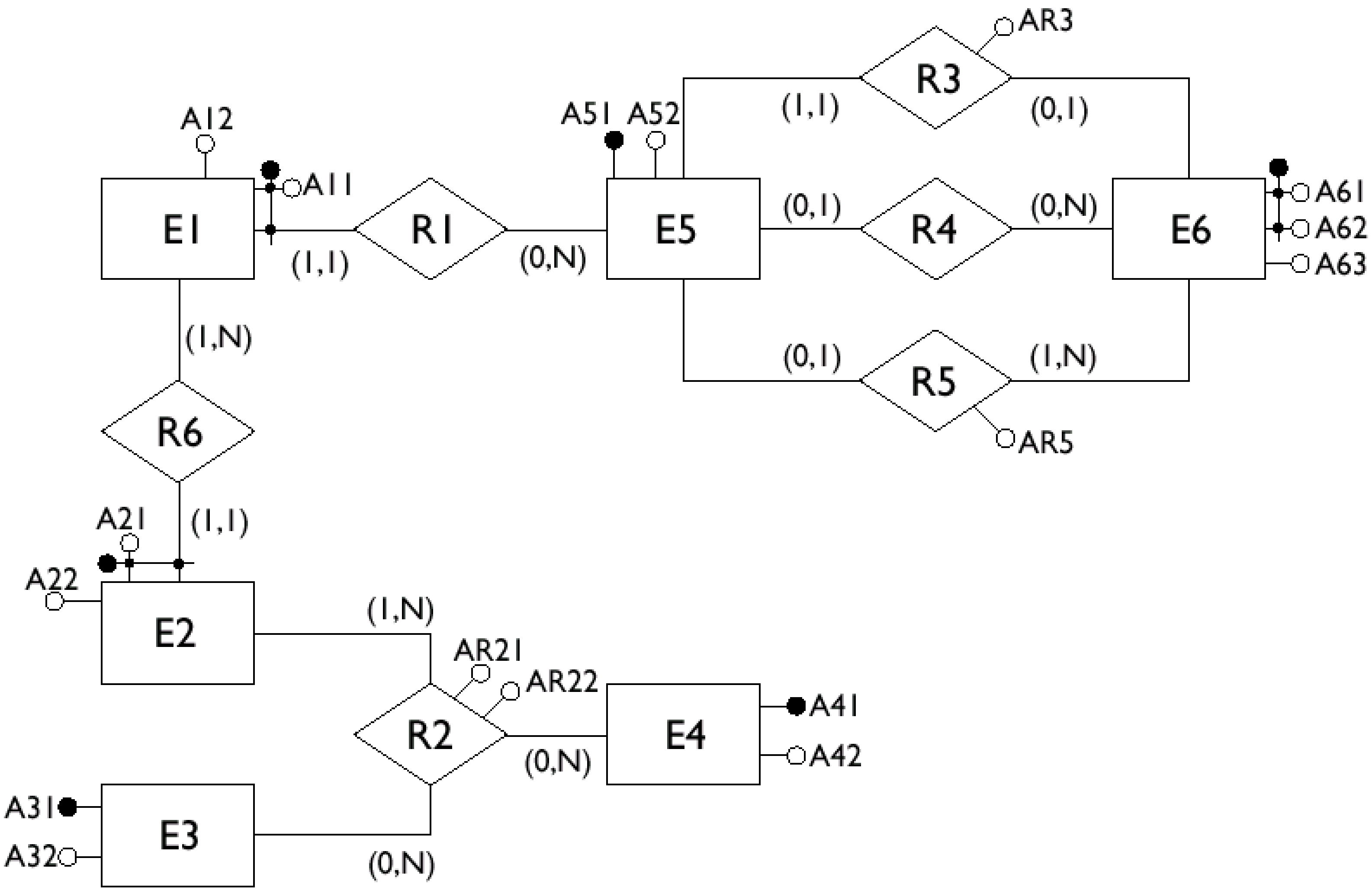
Or, if both entities are optional

Employee(<u>Number</u>, Name, Salary)

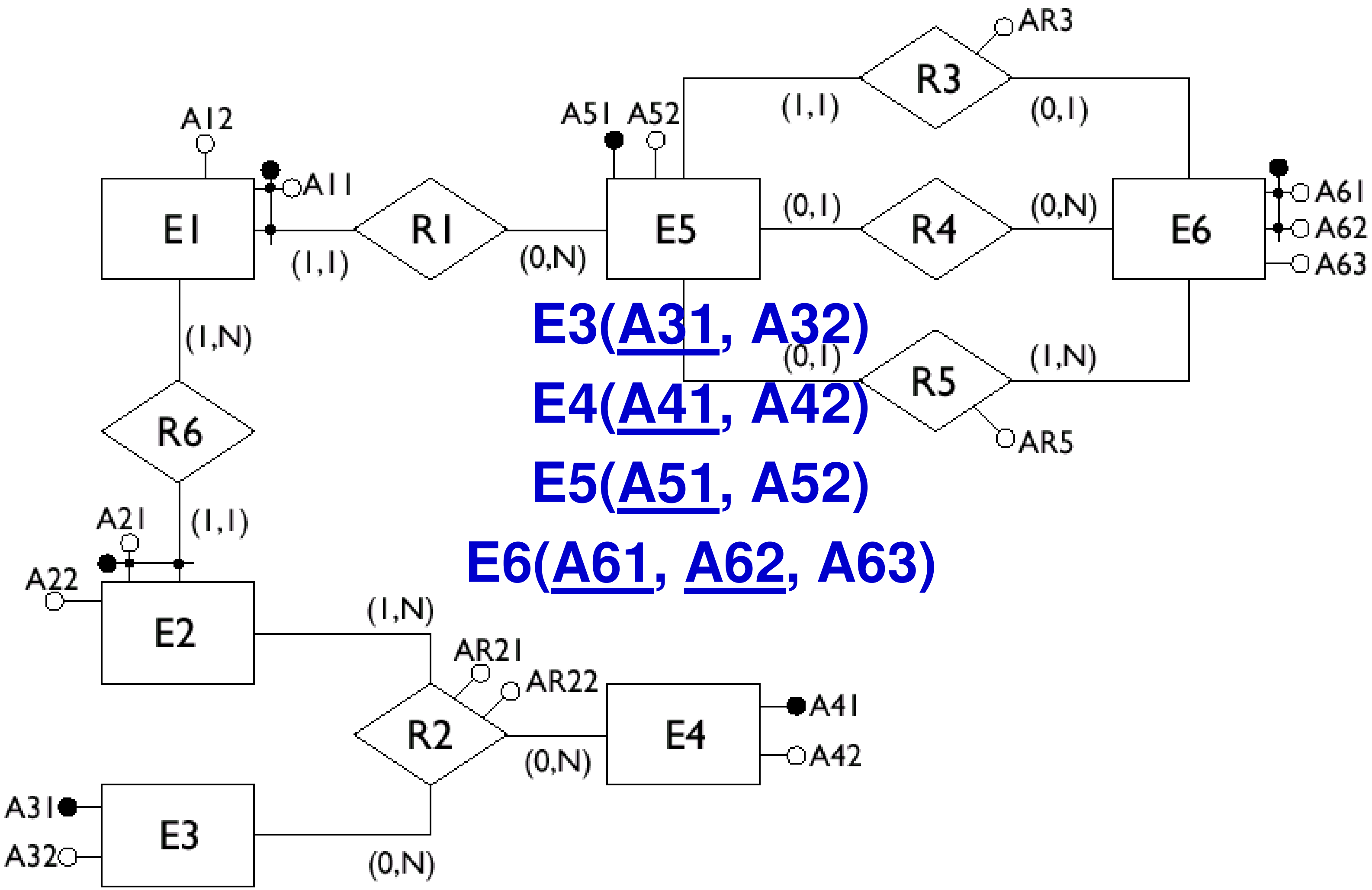
Department(<u>Name</u>, Telephone, Branch)

Management(<u>Head</u>, Department, StartDate)

A Sample ER Schema



Entities with Internal Identifiers



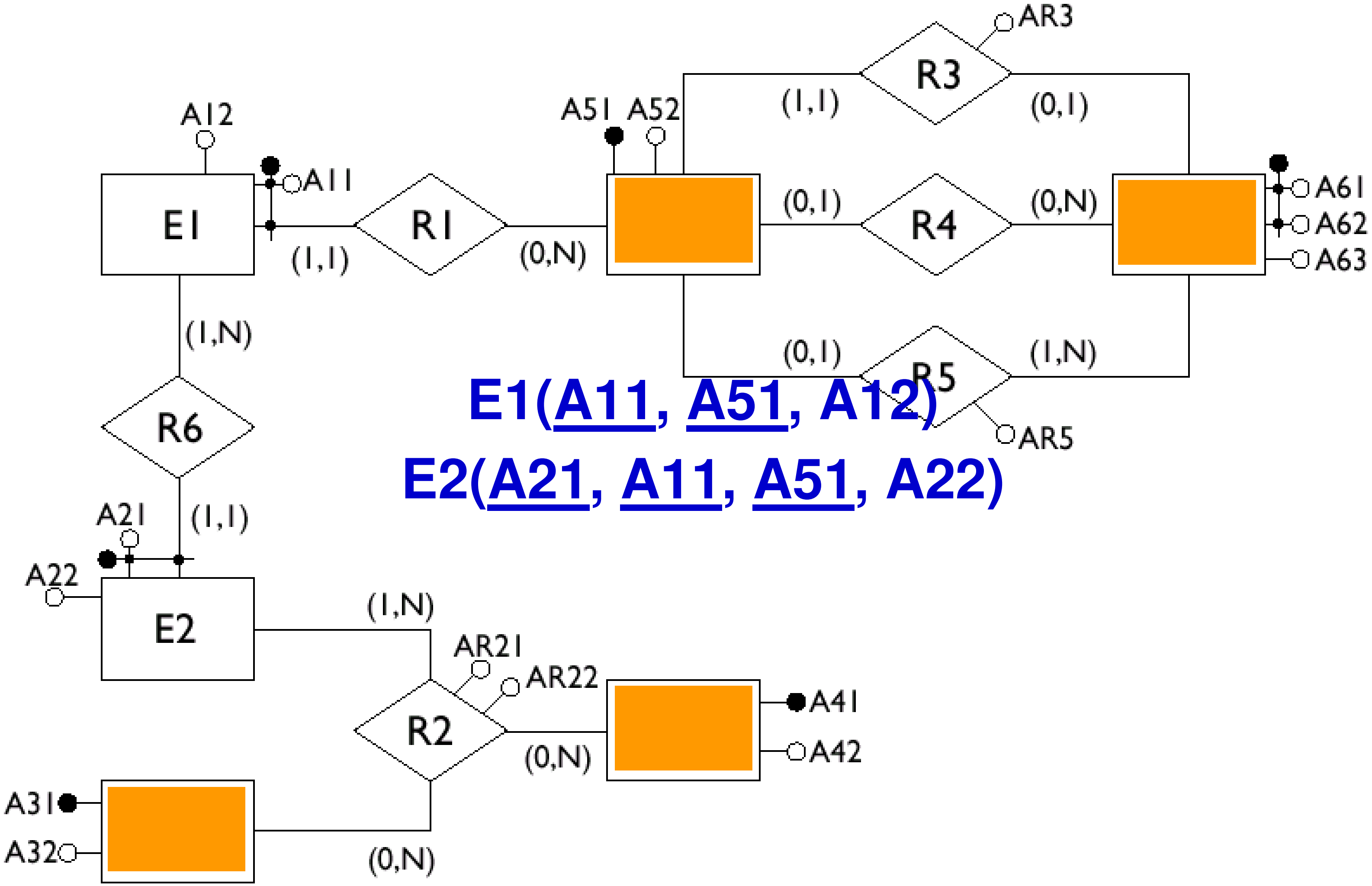
E3(<u>A31</u>, A32)

E4(<u>A41</u>, A42)

E5(<u>A51</u>, A52)

E6(<u>A61</u>, <u>A62</u>, A63)

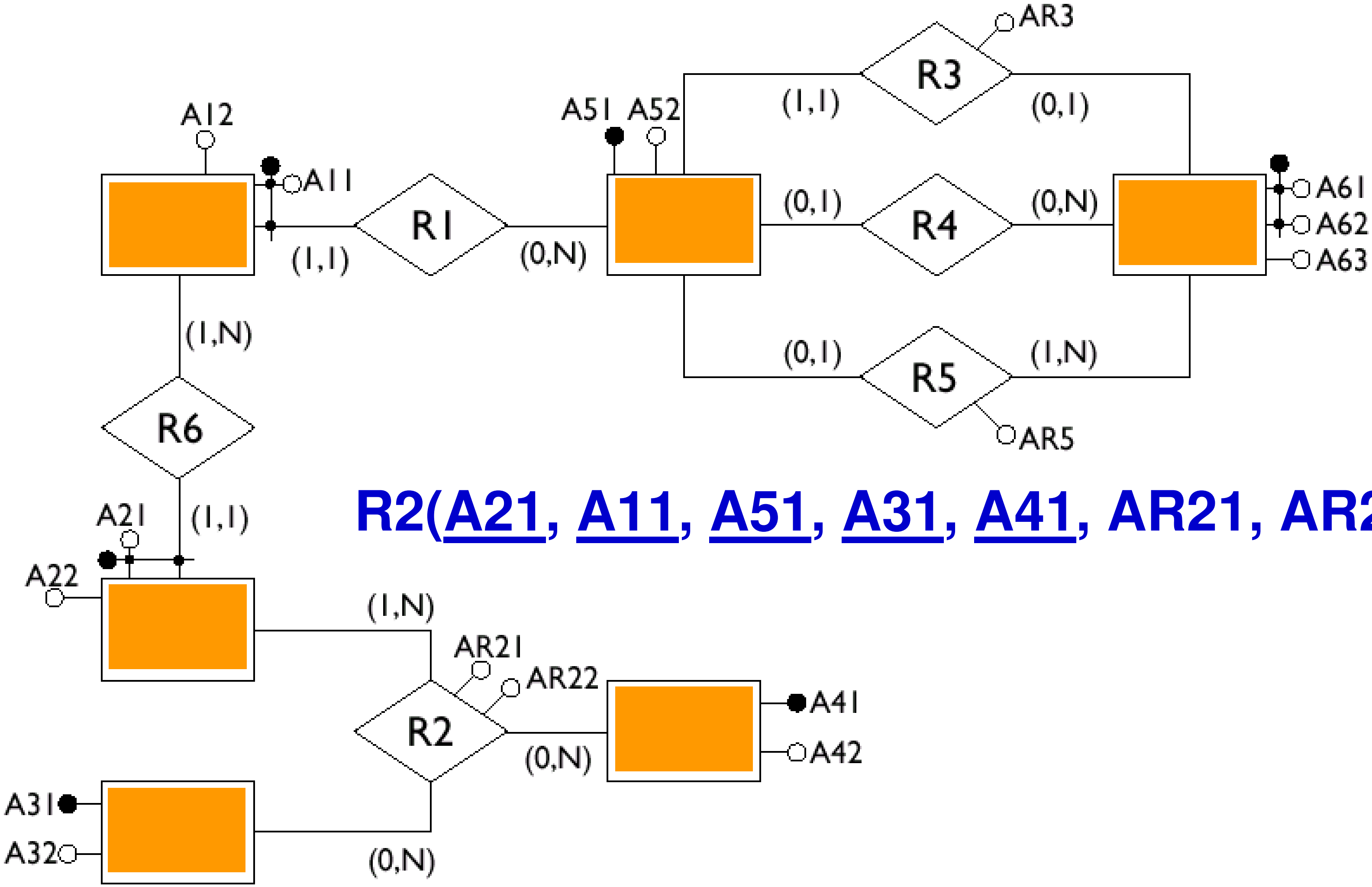
Entities with External Identifiers



E1(A11, A51, A12)

E2(A21, A11, A51, A22)

Many-to-Many Relationships



R2(<u>A21</u>, <u>A11</u>, <u>A51</u>, <u>A31</u>, <u>A41</u>, AR21, AR22)

Result of the Translation

E1(<u>A11</u>, <u>A51</u>, A12)

E2(<u>A21</u>, <u>A11</u>, <u>A51</u>, A22)

E3(<u>A31</u>, A32)

E4(<u>A41</u>,A42)

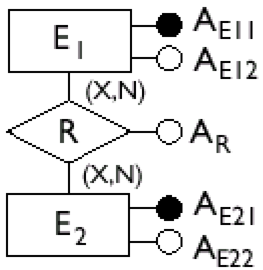
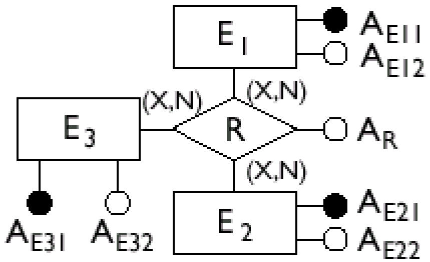
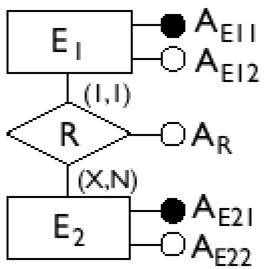
E5(<u>A51</u>, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

E6(<u>A61</u>, <u>A62</u>, A63)

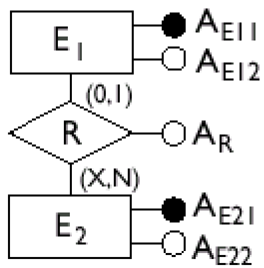
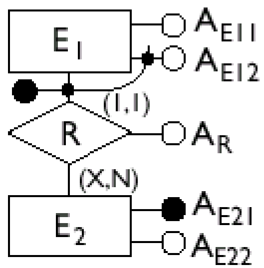
R2(<u>A21</u>, <u>A11</u>, <u>A51</u>, <u>A31</u>, <u>A41</u>, AR21, AR22)

<u>1-1 or optional 1-1 relationships can lead to messy transformations</u>

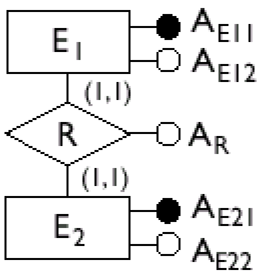
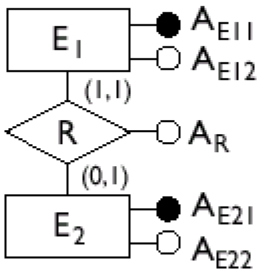
Summary of Transformation Rules

Type	Initial schema	Possible translation
Binary many-to-many relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Ternary many-to-many relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$
One-to-many relationship with mandatory participation		$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

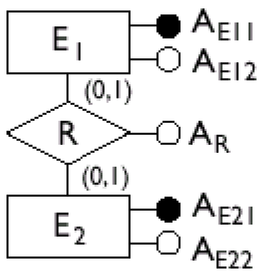
...More Rules...

Type	Initial schema	Possible translation
One-to-many relationship with optional participation		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E21}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Relationship with external identifiers		$E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

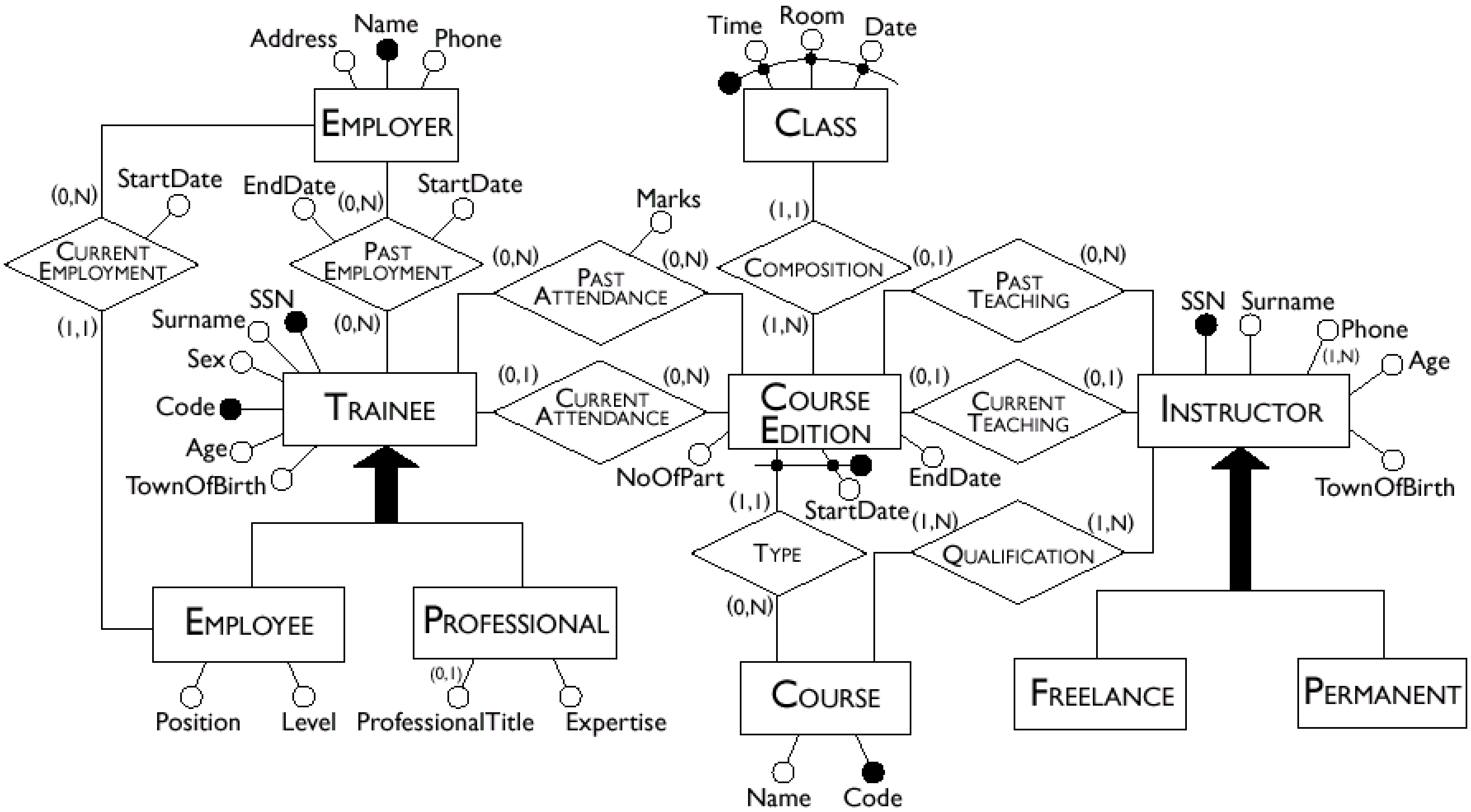
...Even More Rules...

<table>
<tr><th>Type</th><th>Initial schema</th><th>Possible translation</th></tr>
<tr><td>One-to-one relationship with mandatory participation for both entities</td><td>
</td><td>$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$
$E_2(\underline{A_{E21}}, A_{E22})$
Alternatively:
$E_2(\underline{A_{E21}}, A_{E22}, A_{E11}, A_R)$
$E_1(\underline{A_{E11}}, A_{E12})$</td></tr>
<tr><td>One-to-one relationship with optional participation for one entity</td><td>
</td><td>$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$
$E_2(\underline{A_{E21}}, A_{E22})$</td></tr>
</table>

…and the Last One...

Type	Initial schema	Possible translation
One-to-one relationship with optional participation for both entities		$E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A^*_{E11}, A^*_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$

The Training Company Revisited



Operational Requirements, Revisited

- **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).

Database Load

Table of volumes

Concept	Type	Volume
Class	E	8000
CourseEdition	E	1000
Course	E	200
Instructor	E	300
Freelance	E	250
Permanent	E	50
Trainee	E	5000
Employee	E	4000
Professional	E	1000
Employer	E	8000
PastAttendance	R	10000
CurrentAttendance	R	500
Composition	R	8000
Type	R	1000
PastTeaching	R	900
CurrentTeaching	R	100
Qualification	R	500
CurrentEmployment	R	4000
PastEmployment	R	10000

Table of operations

Operation	Type	Frequency
Operation 1	I	40 per day
Operation 2	I	50 per day
Operation 3	I	2 per day
Operation 4	I	15 per day
Operation 5	I	10 per day
Operation 6	I	20 per day
Operation 7	I	5 per day
Operation 8	B	10 per month

Access Tables

The attribute `NumberOfParticipants` in `CourseEdition` can be derived from the relationships `CurrentAttendance` and `PastAttendance`.

Operation 2 with redundancy

Concept	Type	Acc	Type
Trainee	E	1	R
CurrentAtt'nce	R	1	W
CourseEdition	E	1	R
CourseEdition	E	1	W

Operation 5 with redundancy

Concept	Type	Acc	Type
CourseEdition	E	1	R
Type	R	1	R
Course	E	1	R
Composition	R	8	R
Class	E	8	R

Operation 2 without redundancy

Concept	Type	Acc	Type
Trainee	E	1	R
CurrentAtt'nce	R	1	W

Operation 5 without redundancy

Concept	Type	Acc	Type
CourseEdition	E	1	R
Type	R	1	R
Course	E	1	R
Composition	R	8	R
Class	E	8	R
PastAtt'nce	E	10	R

Analysis of Redundancy

- From the access tables we obtain (giving double weight to the write accesses):
  - ✓ presence of redundancy: for operation 2 we have 100 read accesses and 100 write accesses per day; for operation 5 we have 190 read accesses per day, for a total of 490 accesses per day;
  - ✓ without redundancy: for operation 2 we have 50 read accesses per day and 100 write accesses per day; for operation 5, we have 290 read accesses per day, for a total of 440 accesses per day.
- Thus, when the redundancy is present, we have disadvantages both in terms of storage and access time.
- We therefore delete the attribute `NumberOfParticipants` from the entity `CourseEdition`.

Removing Generalizations

- For the generalization on instructors:
  - ✓ the relevant operations make no distinction between the child entities and these entities have no specific attributes;
  - ✓ we can therefore delete the child entities and add an attribute Type to the parent entity.
- For the generalization on trainees:
  - ✓ the relevant operations make no distinction between the child entities, but these entities have specific attributes;
  - ✓ we can therefore leave all the entities and add two relationships to link each child with the parent entity: in this way, we will have no attributes with possible null values on the parent entity and the dimension of the relations will be reduced.

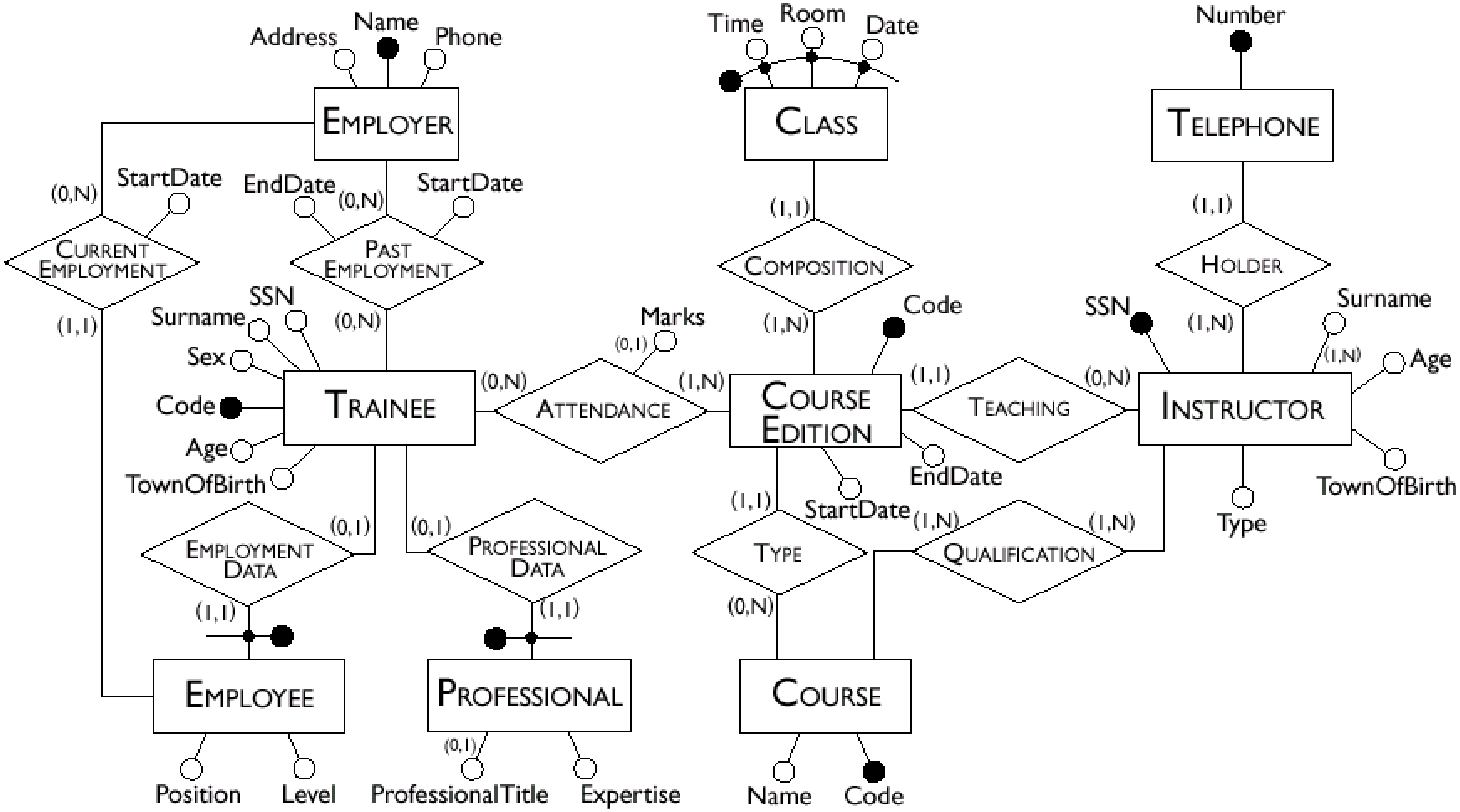
Partitioning and Merging of Concepts

- The relationships `PastTeaching` and `PresentTeaching` can be merged since they describe similar concepts between which the operations make no difference. A similar consideration applies to the relationships `PastAttendance` and `PresentAttendance`.
- The multi-valued attribute `Telephone` can be removed from the `Instructor` entity by introducing a new entity `Telephone` linked by a one-to-many relationship to the `Instructor` entity.

Choice of Main Identifiers

- `Trainee` entity:
  - ✓ there are two identifiers: the social security number and the internal code;
  - ✓ it is far preferable to choose the latter: a social security number will require several bytes whereas an internal code, which serves to distinguish between 5000 occurrences, requires a few bytes.
- `CourseEdition` entity:
  - ✓ it is identified externally by the `StartDate` attribute and by the `Course` entity;
  - ✓ we can see however that we can easily generate for each edition a code from the course code: this code is simpler and can replace the external identifier.

After Restructuring



Translation into the Relational Model

CourseEdition(<u>Code</u>, StartDate, EndDate, Course, Instructor)

Class(<u>Time</u>, <u>Room</u>, <u>Date</u>, Edition)

Instructor(<u>SSN</u>, Surname, Age, TownOfBirth, Type)

Telephone(<u>Number</u>, Instructor)

Course(<u>Code</u>, Name)

Qualification(<u>Course</u>, <u>Instructor</u>)

Trainee(<u>Code</u>, SSN, Surname, Age, TownOfBirth, Sex)

Attendance(<u>Trainee</u>, <u>Edition</u>, Marks*)

Employer(<u>Name</u>, Address, Telephone)

PastEmployment(<u>Trainee</u>, <u>Employer</u>, StartDate, EndDate)

Professional(<u>Trainee</u>, Expertise, ProfessionalTitle*)

Employee(<u>Trainee</u>, Level, Position, Employer, StartDate)

Logical Design Using CASE Tools

- The logical design phase is partially supported by all database design tools:
  - ✓ the translation to the relational model it is carried out by these systems almost automatically;
  - ✓ the restructuring step is difficult to automate and the various products provide little or no support for it.
- All the systems are able to generate automatically the SQL code for the creation of the database.
- Some systems allow direct connection with a DBMS and can construct the corresponding database automatically.

Logical Design with a CASE tool