SAPIENZA
UNIVERSITÀ DI ROMA

Data Management and Analysis
Unit 2

Logical Design

Dott. Franco Liberati
liberati@di.uniroma1.it

Logical Design

General

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

Secondly, the aim of conceptual design is to represent the data accurately and naturally from a high-level, computer-independent point of view

Logical Design

General

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

Restructuring of the Entity-Relationship schema, which is independent of the chosen logical model and is based on criteria for the optimization of the schema and the simplification of the following step

Translation into the logical model, which refers to a specific logical model (in our case, the relational model) and can include a further optimization, based on the features of the logical model itself

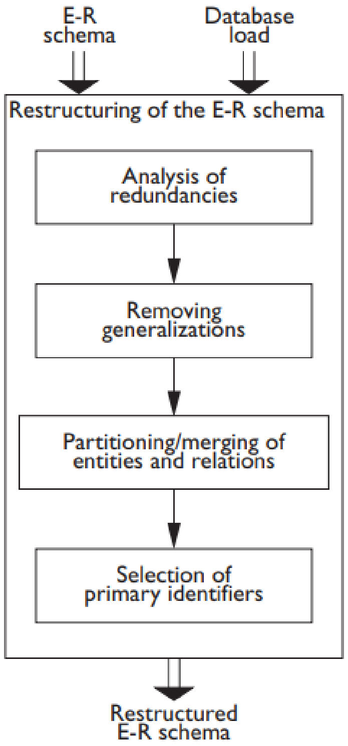
LOGICAL DESIGN: Restructuring of the Entity-Relationship schema

Logical Design

Restructuring

The restructuring step of an E-R model can be sub-divided into a series of tasks to be carried out in sequence:

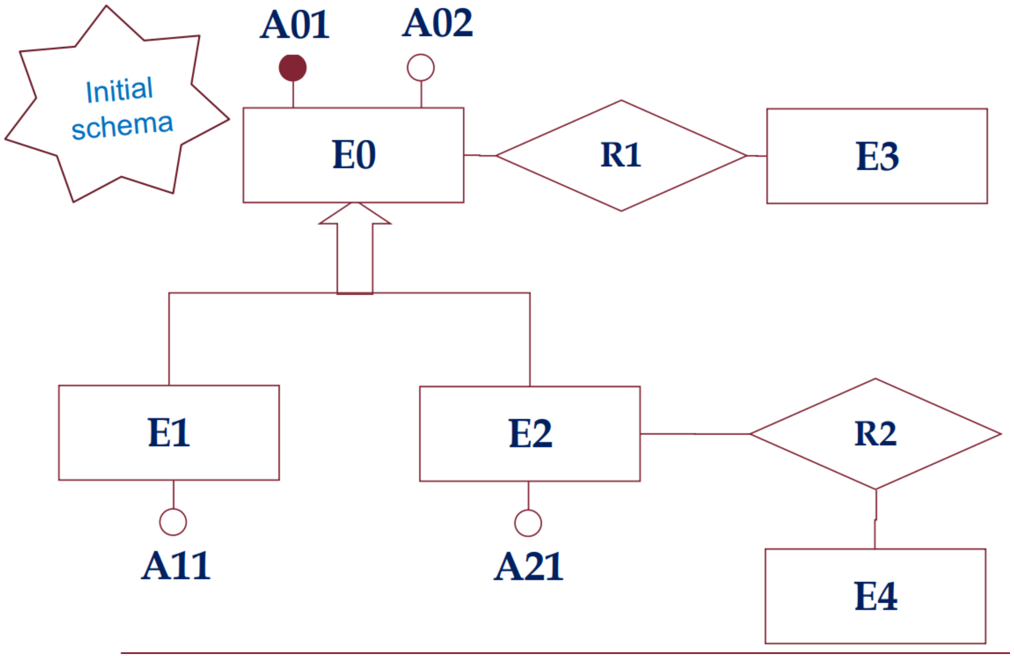
- **Analysis of redundancies** decides whether to delete or retain possible redundancies present in the schema
- **Removing generalizations** replaces all the generalizations in the schema by other constructs



Logical Design

Restructuring: redundancies sample

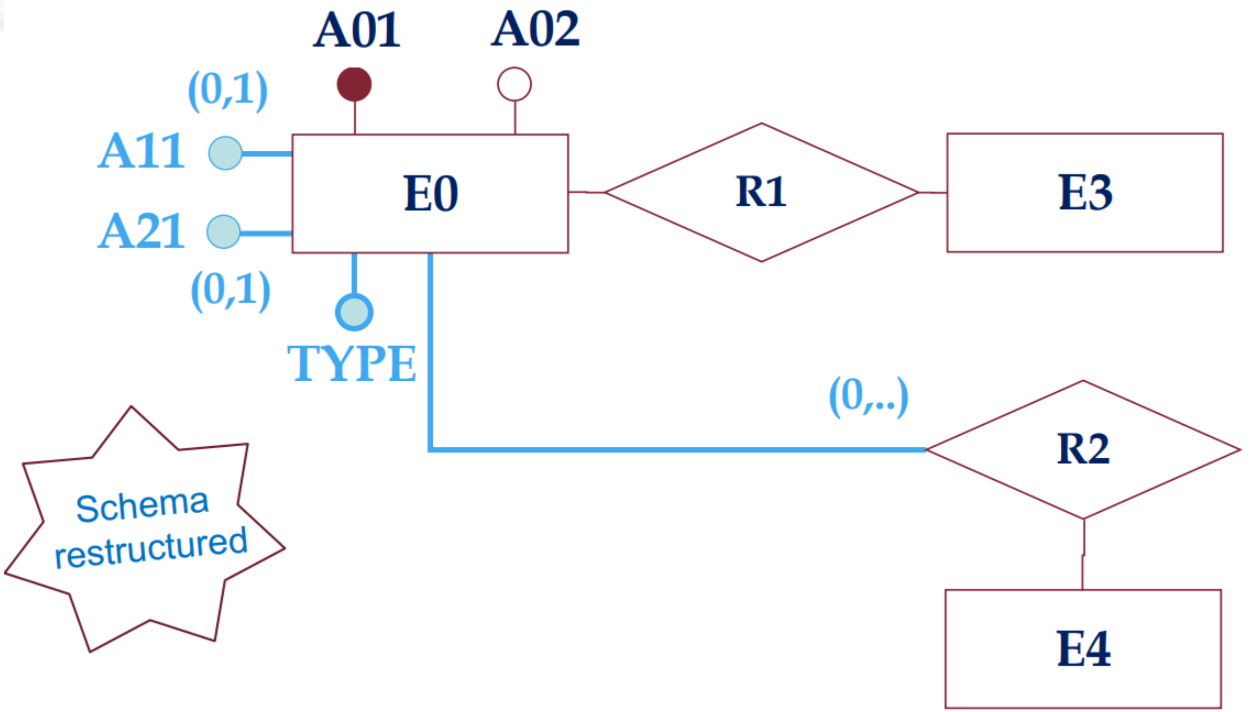
Manipulating generalizations: Merging Child Entities into the Parent



Logical Design

Restructuring: redundancies sample

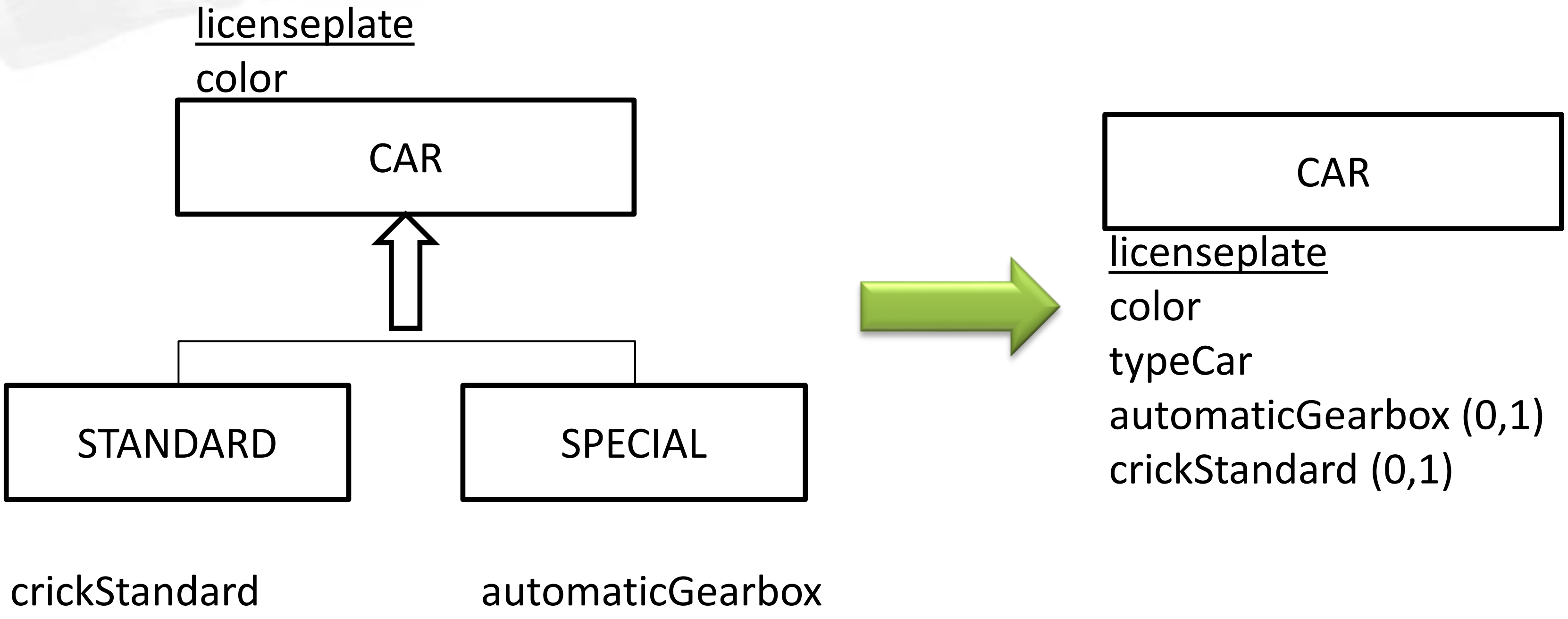
Manipulating generalizations : Merging Child Entities into the Parent



It is convenient if access to the father and children is contextual

Logical Design
Restructuring: redundancies sample

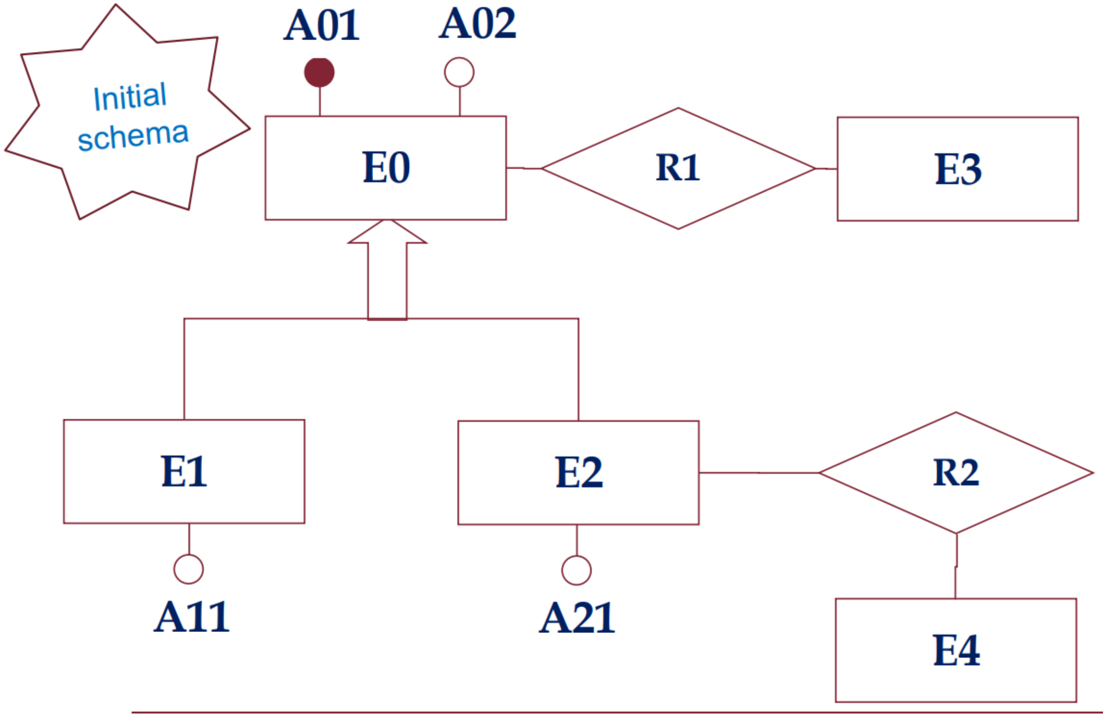
Manipulating generalizations : Merging Child Entities into the Parent



Logical Design

Restructuring: redundancies sample

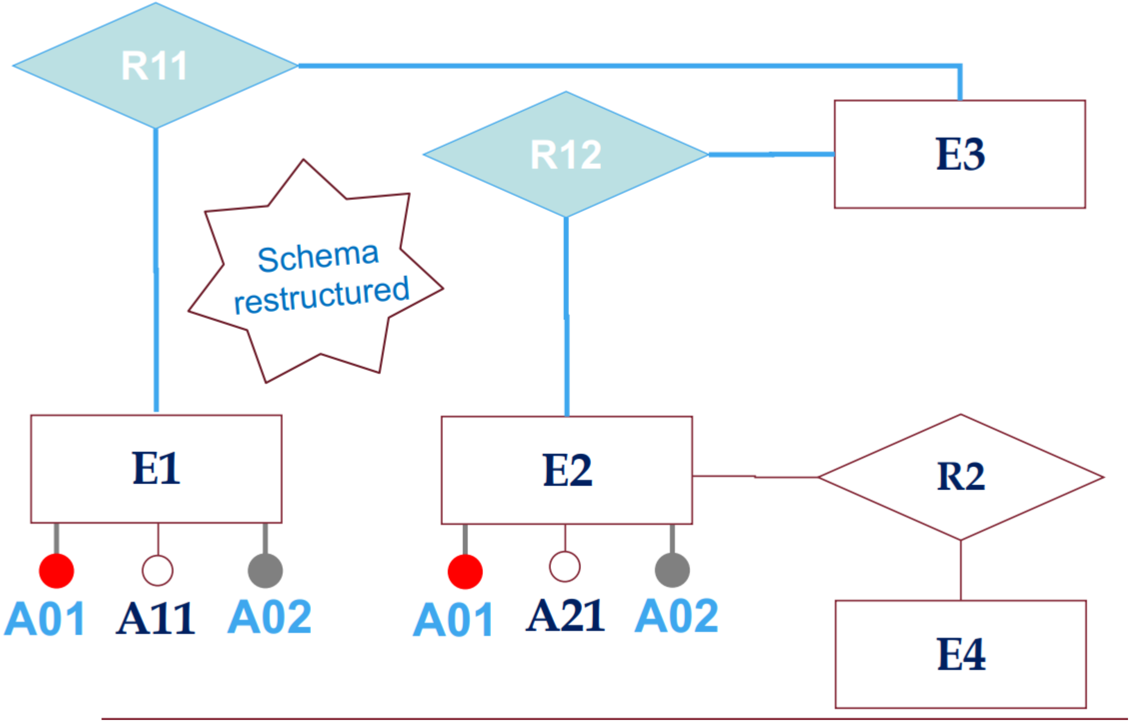
Manipulating generalizations : Merging the parent into the child entities



Logical Design

Restructuring: redundancies sample

Manipulating generalizations : Merging the parent into the child entities

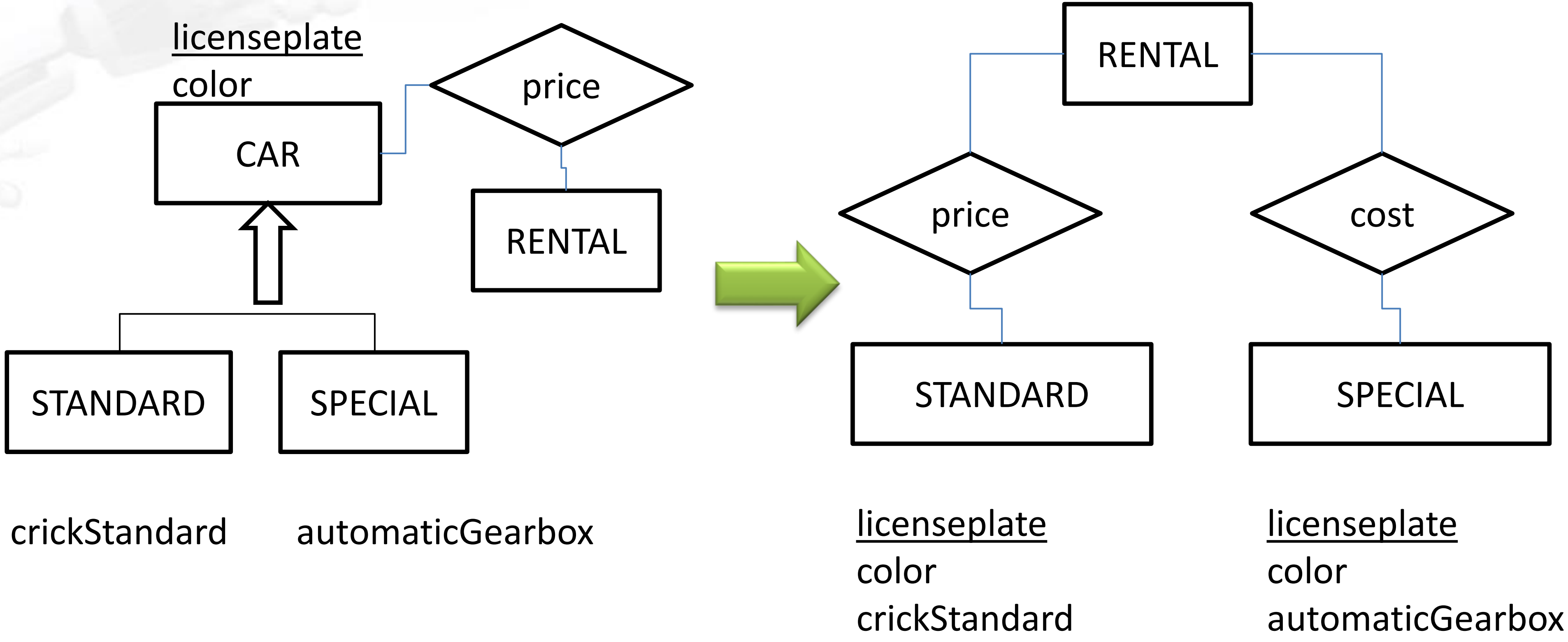


It is appropriate if the accesses to the children are distinct

Logical Design

Restructuring: redundancies sample

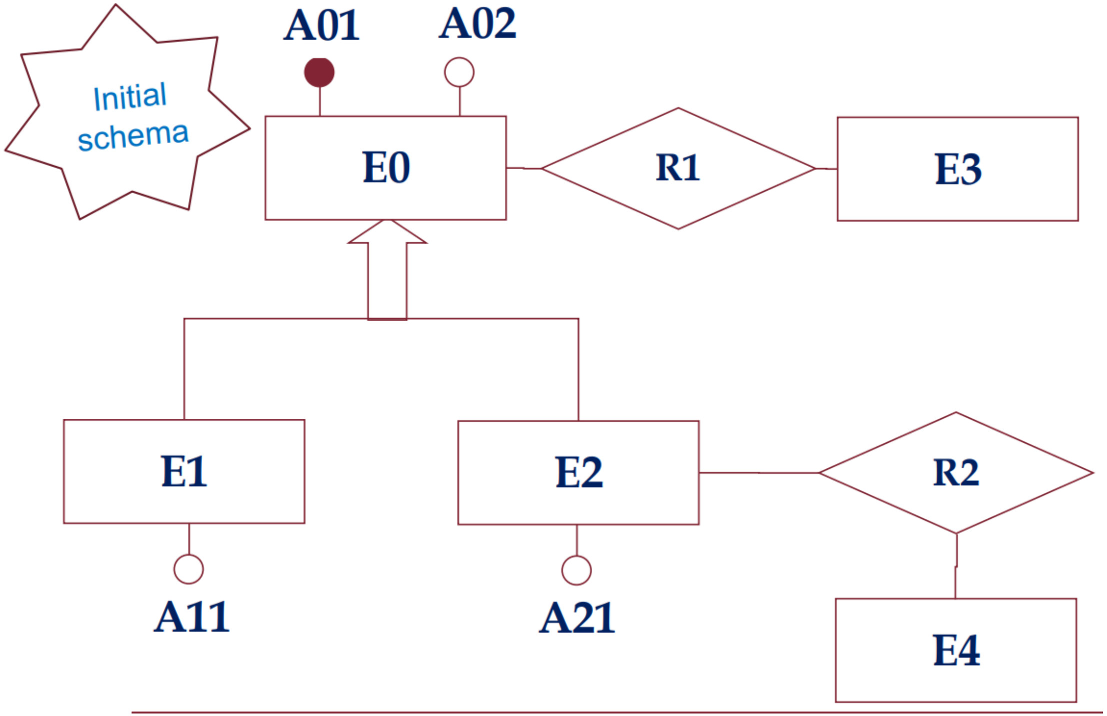
Manipulating generalizations : Merging the parent into the child entities



Logical Design

Restructuring: redundancies sample

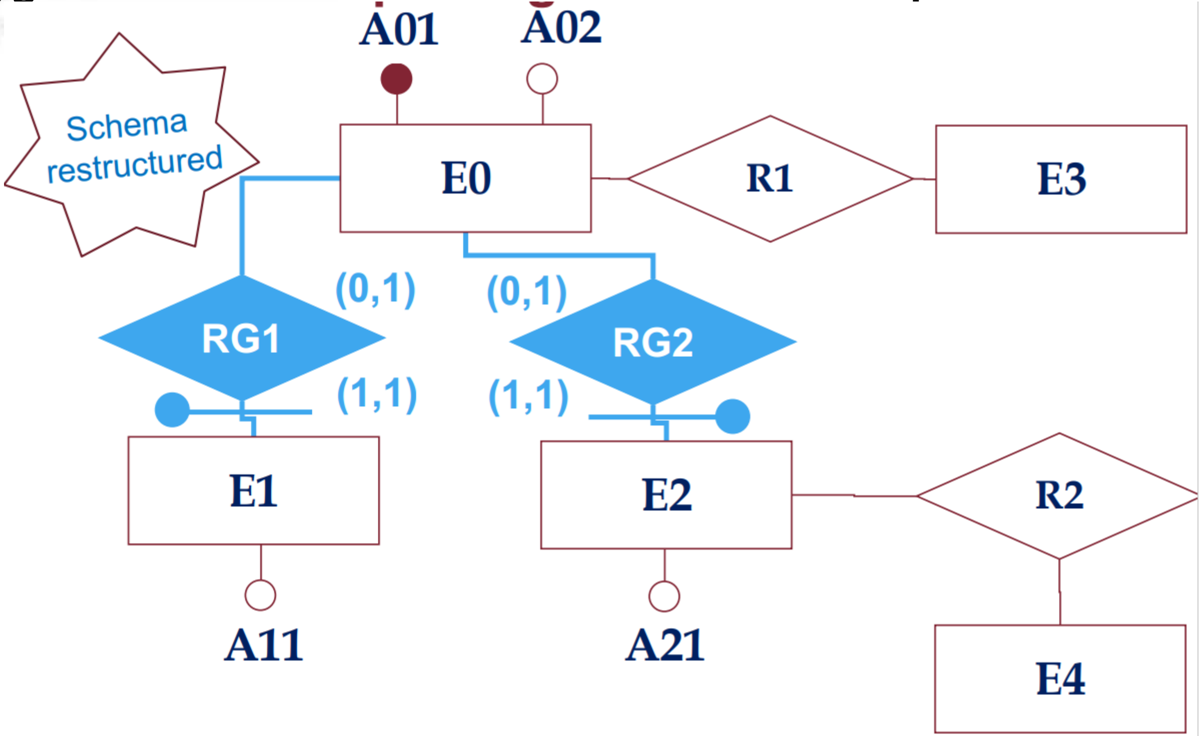
Manipulating generalizations : substitution with relationships



Logical Design

Restructuring: redundancies sample

Manipulating generalizations : substitution with relationships

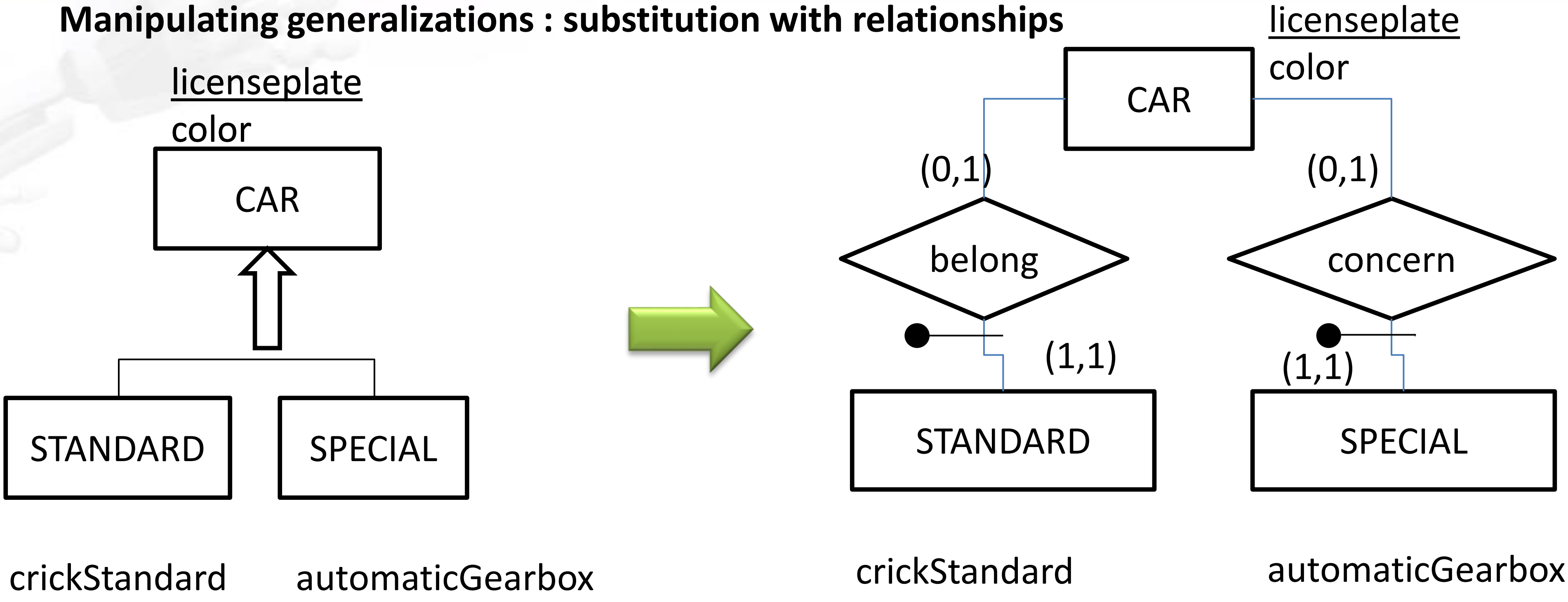


It s appropriate if accesses to child entities are separated from accesses to the parent

Logical Design

Restructuring: redundancies sample

Manipulating generalizations : substitution with relationships

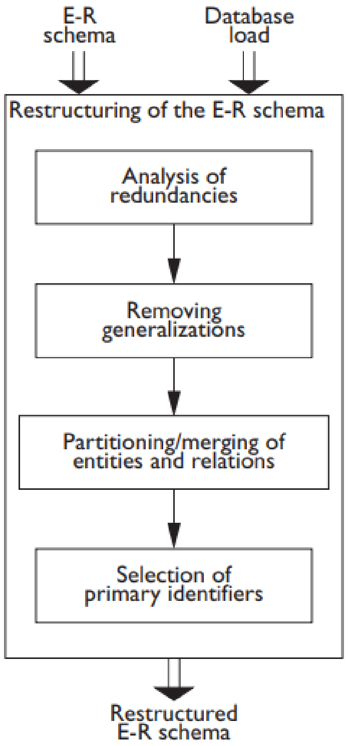


Logical Design

Restructuring

The restructuring step of an e-r schema can be sub-divided into a series of tasks to be carried out in sequence:

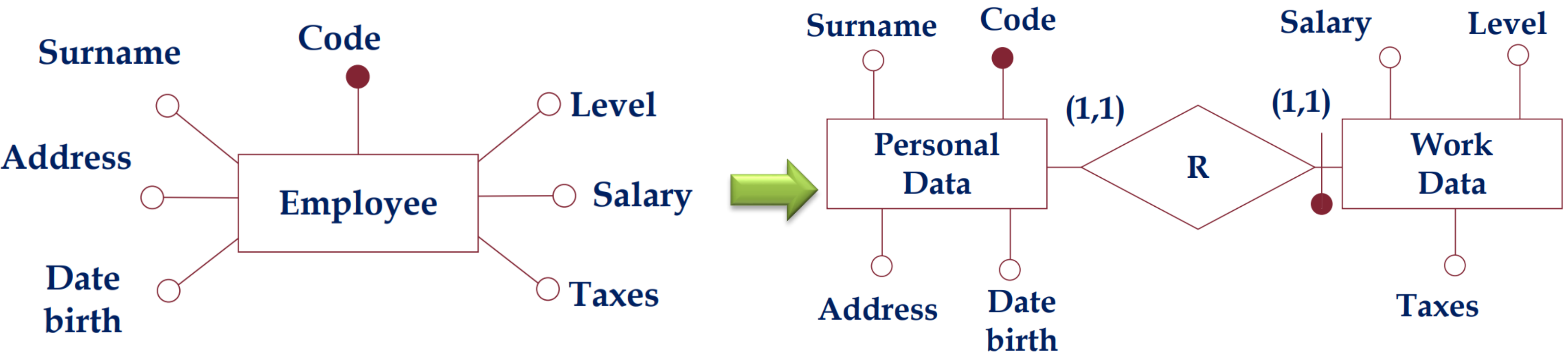
- **Analysis of redundancies**
- **Removing generalizations**
- **Partitioning and merging of entities and relationships** decides whether is it convenient to partition concepts in the schema into more than one concept or to merge several separate concepts into a single one



Logical Design

Partitioning and merging of entities and relationships

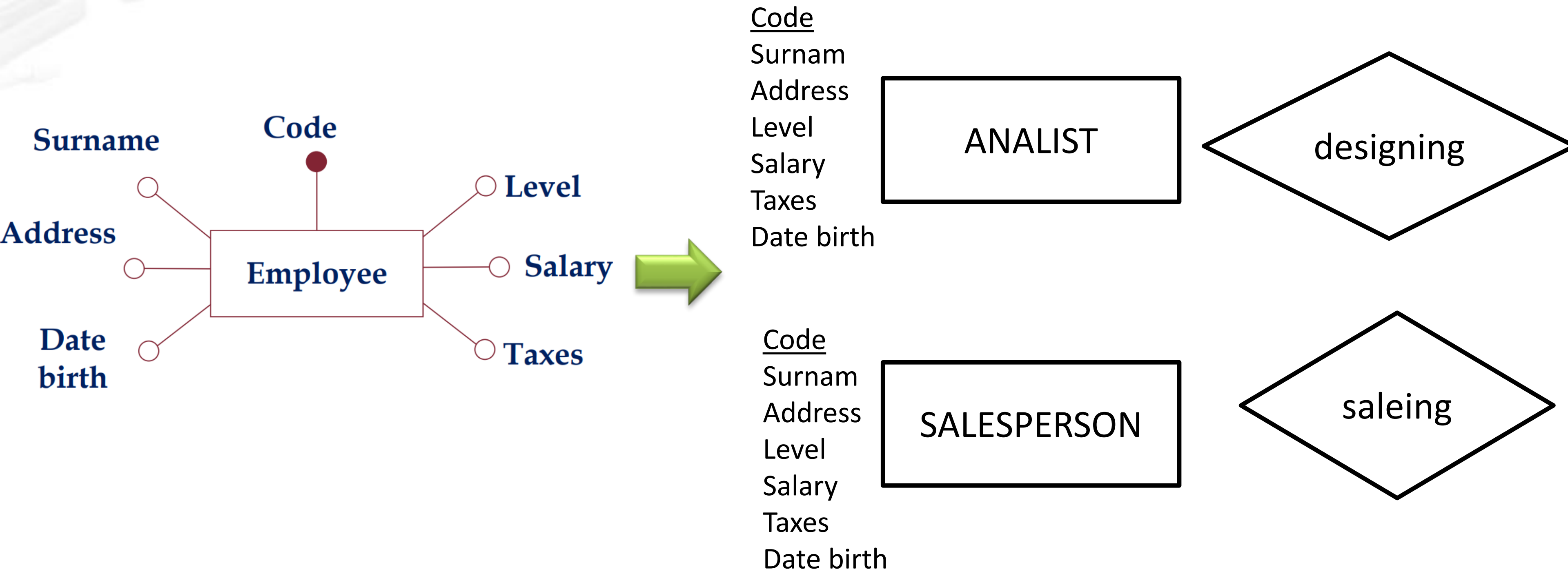
Partitioning and merging of entities and relationships: Vertical partitioning of an entity, in the sense that the concept is sub-divided according to its attributes



Logical Design

Partitioning and merging of entities and relationships

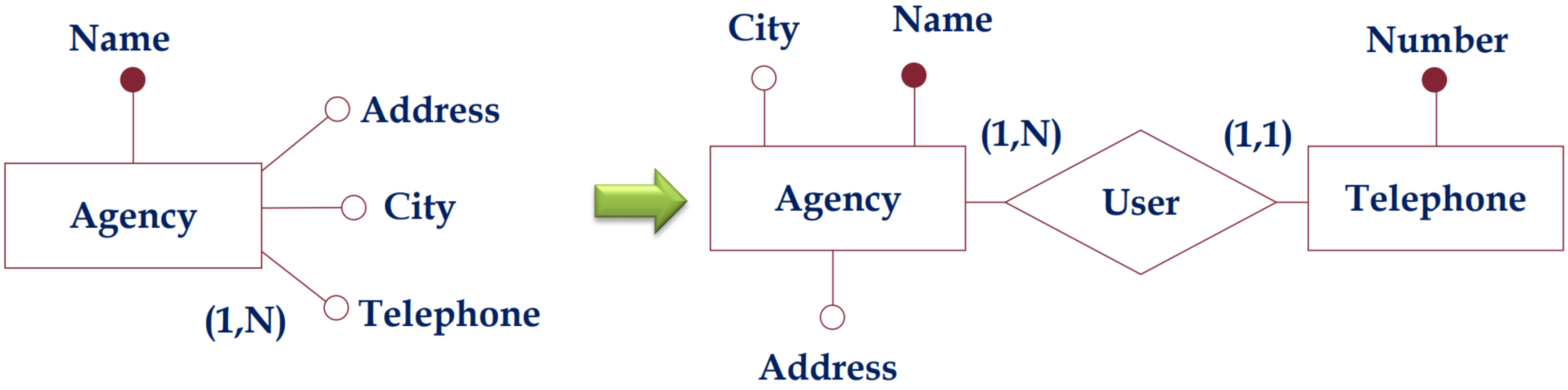
Partitioning and merging of entities and relationships: in horizontal partitioning the sub-division works on the occurrences of entities



Logical Design

Partitioning and merging of entities and relationships

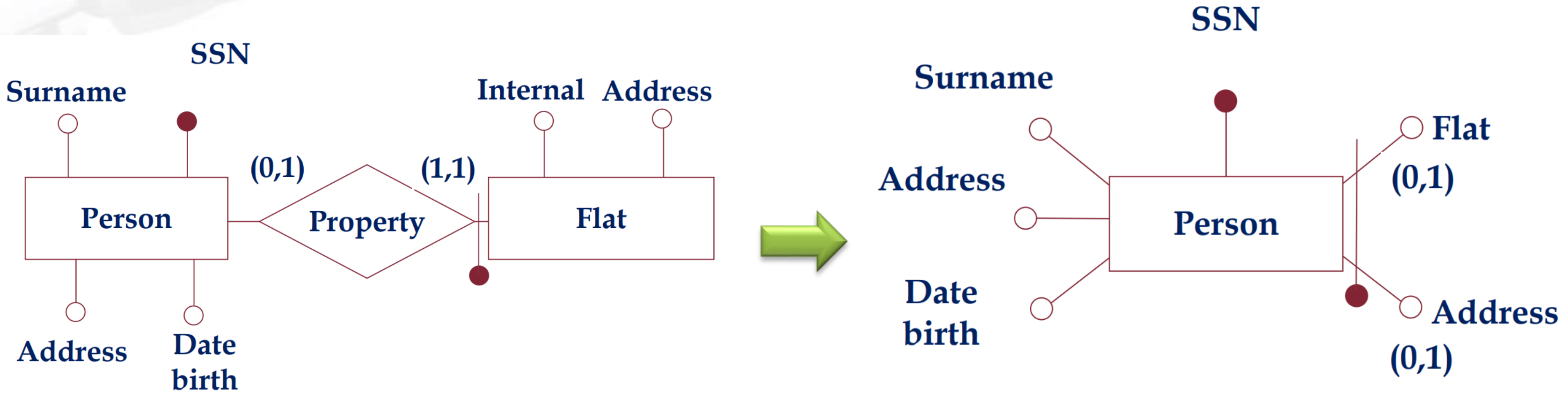
Deletion of multi-valued attributes



Logical Design

Partitioning and merging of entities and relationships

Merging of entities: Merging is the reverse operation of partitioning



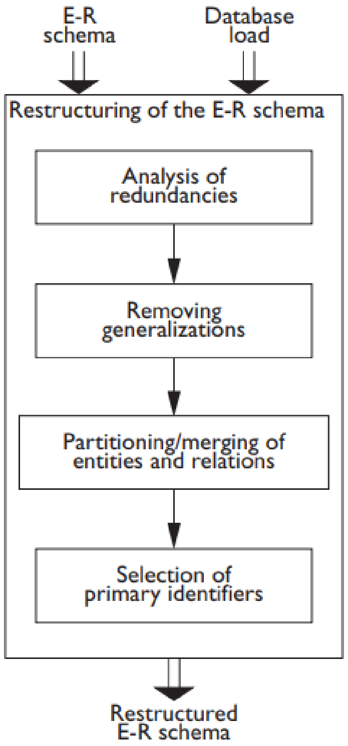
Merging is generally carried out on one-to-one relationships (because redundancy is present), rarely on one-to-many relationships and hardly ever on many-to-many relationships.

Logical Design

Restructuring

The restructuring step of an e-r schema can be sub-divided into a series of tasks to be carried out in sequence:

- **Analysis of redundancies**
- **Removing generalizations**
- **Partitioning and merging of entities and relationships**
- **Selection of primary identifiers** chooses an identifier for those entities that have more than one.



Logical Design

Selection of primary identifiers

The choice of an identifier for each entity is essential to the translation into the relational model, because of the major role keys play in a value-based data model

Logical Design

Selection of primary identifiers

The criteria for this decision are as follows.

Attributes with null values cannot form primary identifiers.

These attributes do not guarantee access to all the occurrences of the corresponding entity, as we pointed out while discussing keys for the relational model.

One or few attributes are preferable to many attributes

This ensures that the indices are of limited size, less storage is needed for the creation of logical links among the various relations, and join operations are facilitated.

An internal identifier with few attributes is preferable to an external one, possibly involving many entities

This is because external identifiers are translated into keys comprising the identifiers of the entities involved in the external identification. Thus, keys with many attributes would be generated.

An identifier that is used by many operations to access the occurrences of an entity is preferable to others

In this way, these operations can be executed efficiently, since they can take advantage of the indices automatically built by the DBMS

LOGICAL DESIGN:
Translation into the relational model

Logical Design

Translation into the relational model

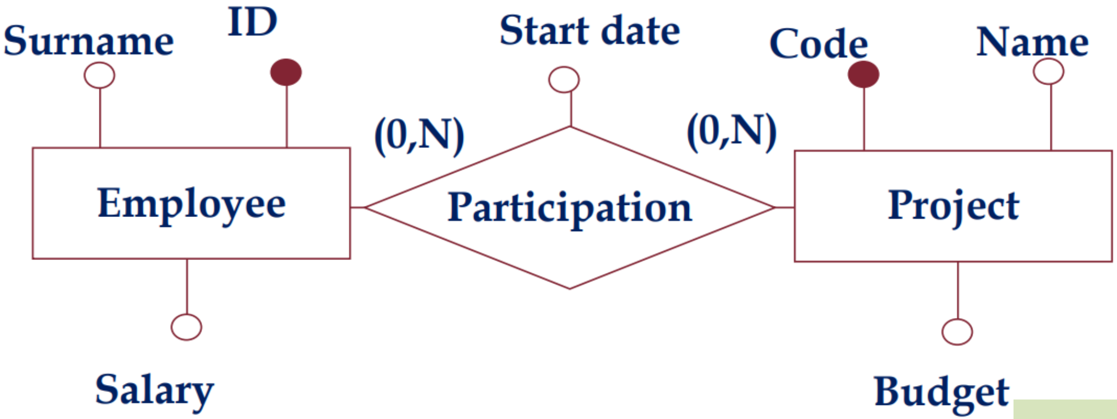
Starting from an E-R Model, an equivalent Relational Schema is constructed

Case:

- ❑ Entities and many-to-many relationships
- ❑ One-to-many relationships
- ❑ Entities with external identifiers
- ❑ One-to-one relationships

Logical Design

Entities and many-to-many relationships



Employee(<u>ID</u>, Surname, Salary)

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

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

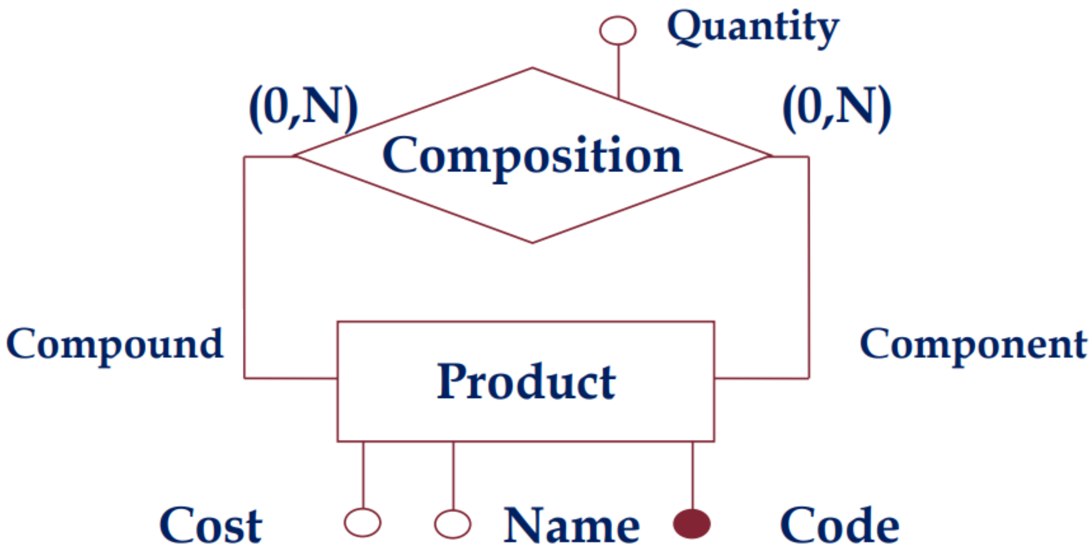
Referential integrity constraints between:

- ID in Participation and (the key of) Employee
- Code in Participation and (the key of) Project

NB: It is better to use more expressive names that make the constraints more visible
Es.:Participation(Employee, Project, StartDate)

Logical Design

Entities and many-to-many relationships RECURSIVE REL

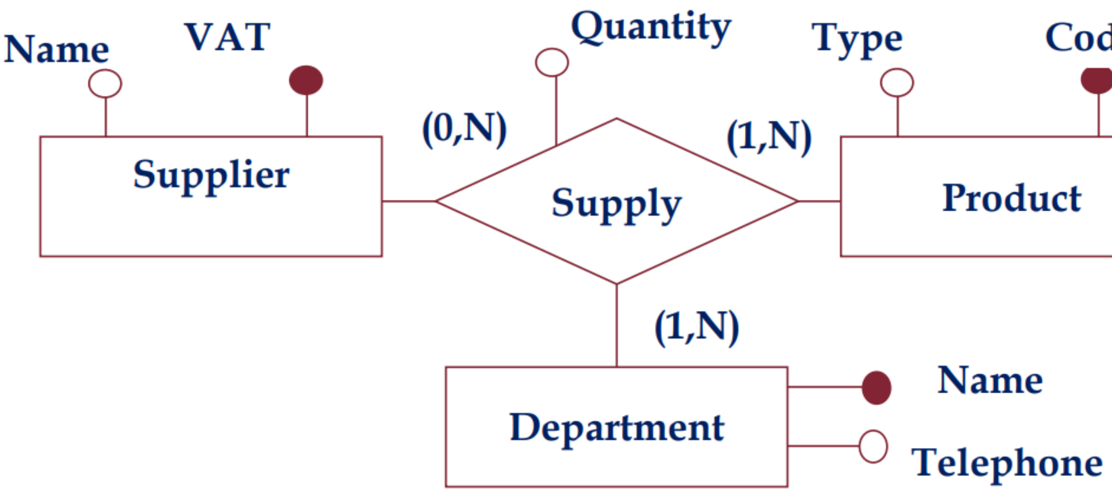


Product(<u>Code</u>, Name, Cost)

Composition(<u>Compound</u>, <u>Component</u>, Quantity)

Logical Design

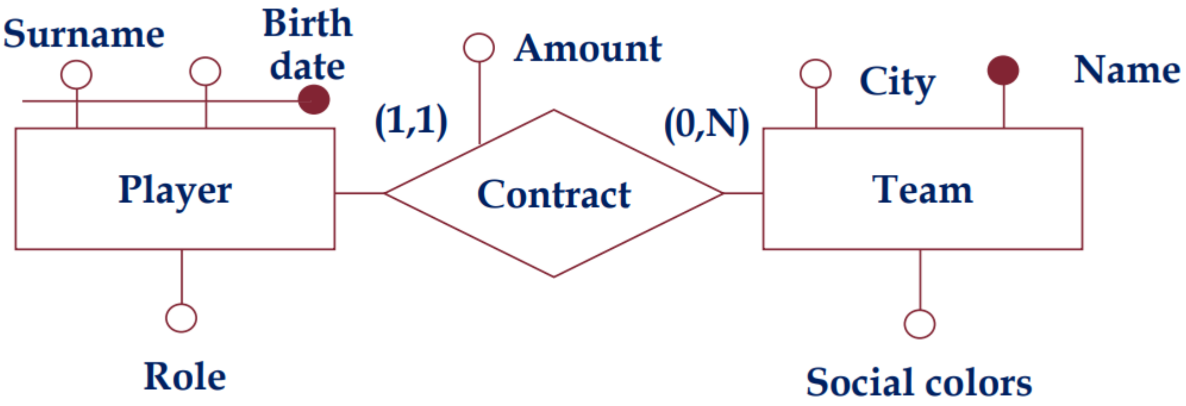
Entities and many-to-many relationships N-ary REL



Supplier(<u>VAT</u>, Name)
Product(<u>Code</u>, Type)
Department(<u>Name</u>, Telephone)
Supply(<u>Supplier</u>, <u>Product</u>, <u>Department</u>, Quantity)

Logical Design

One-to-many relationships

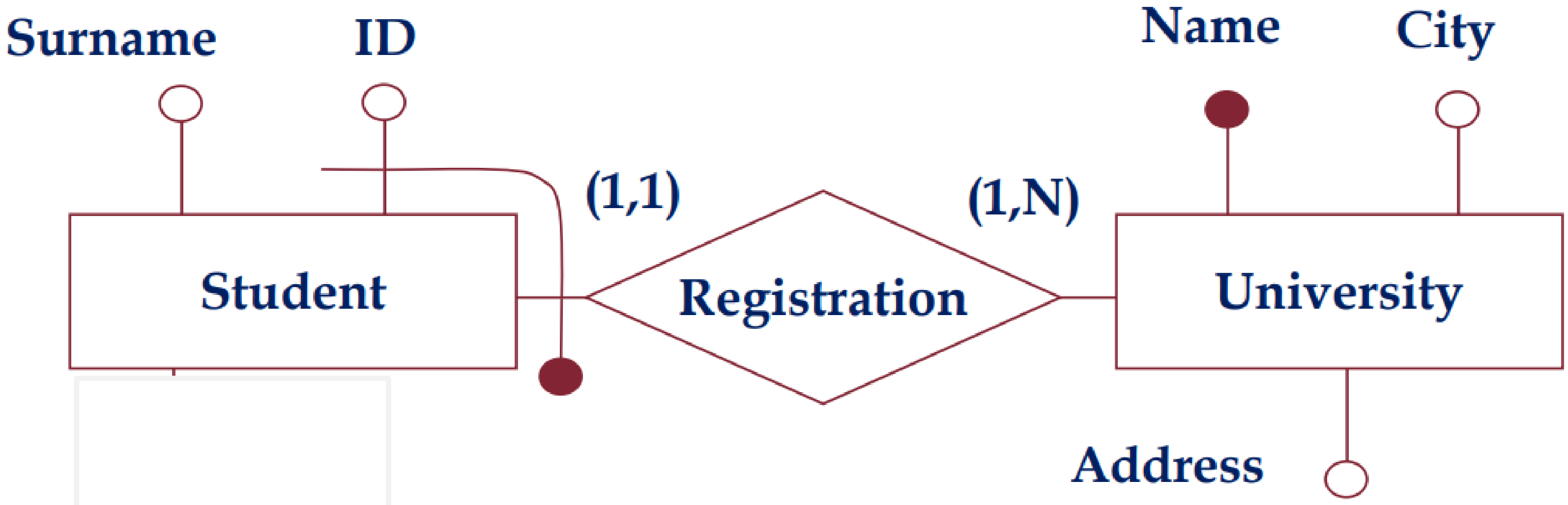


Player(Surname, BirthDate, Role)
Contract(SurPlayer, BirthDateP, Team, Amount)
Team(Name, City, SocialColors)

Compact solution:
Player(Surname, BirthDate, Role, Team, Amount)
Team(Name, City, SocialColors)

Logical Design

Entities with external identifiers

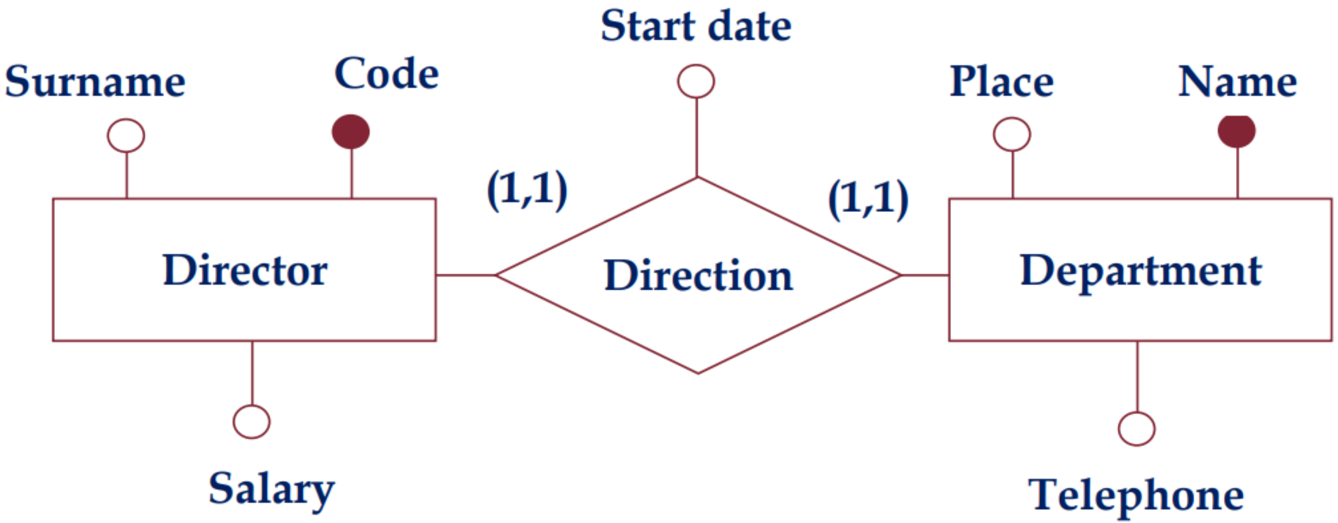


Student(<u>ID</u>, <u>University</u>, Surname)

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

Logical Design

One-to-one relationships



Director(<u>Code</u>, Surname, Salary)

Department(<u>Name</u>, Place, Telephone, Director, startD)

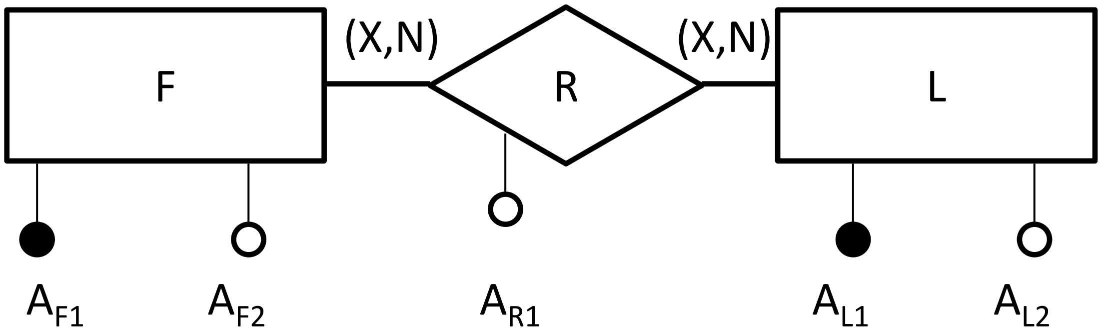
With referential integrity constraint, without null values

LOGICAL DESIGN:
Trasformation rules

Logical Design

Transformation rules

BINARY MANY TO MANY RELATIONSHIP



F($\underline{A_{F1}}$, A_{F2})

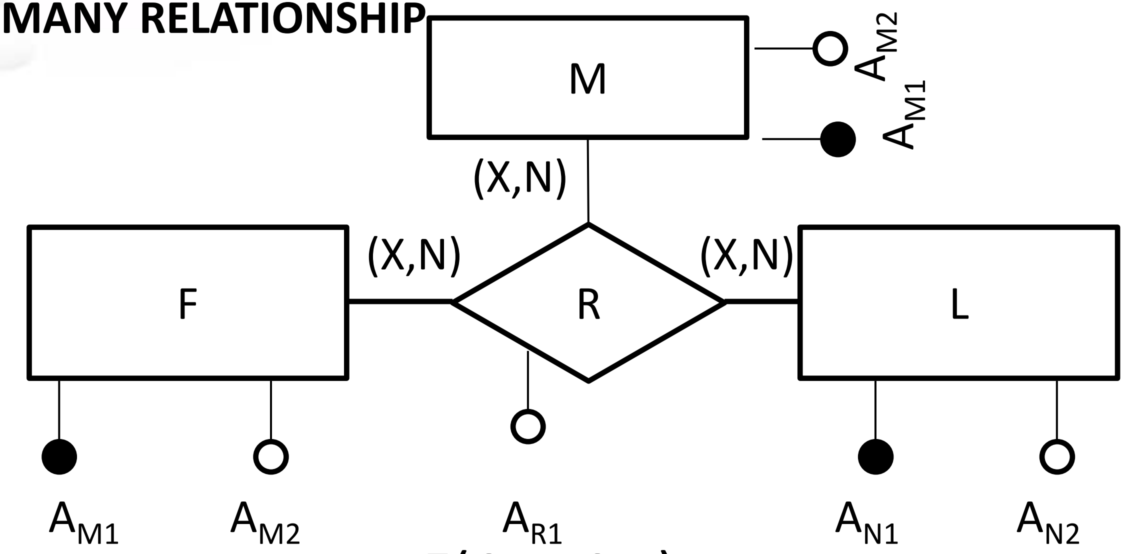
L($\underline{A_{L1}}$, A_{L2})

R($\underline{A_{F1}}$, $\underline{A_{L1}}$, A_{R1})

Logical Design

Transformation rules

TERNARY MANY TO MANY RELATIONSHIP



F($\underline{A_{F1}}$, A_{F2})

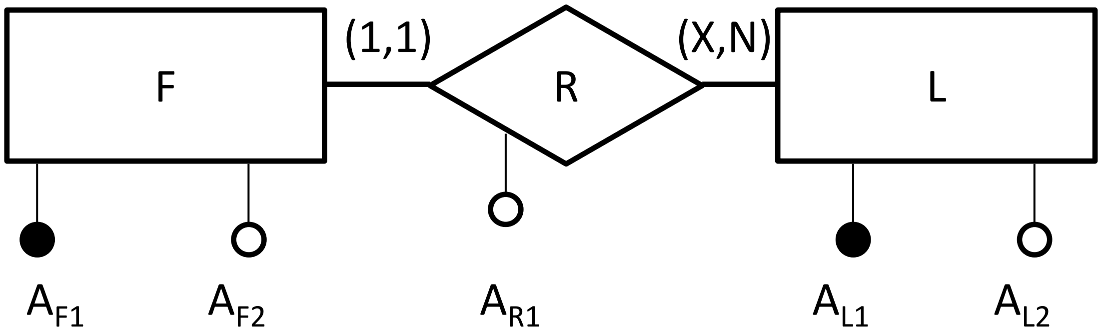
L($\underline{A_{L1}}$, A_{L2})

M($\underline{A_{M1}}$, A_{M2})

R($\underline{A_{F1}}$, $\underline{A_{L1}}$, $\underline{A_{M1}}$, A_{R1})

Logical Design
Transformation rules

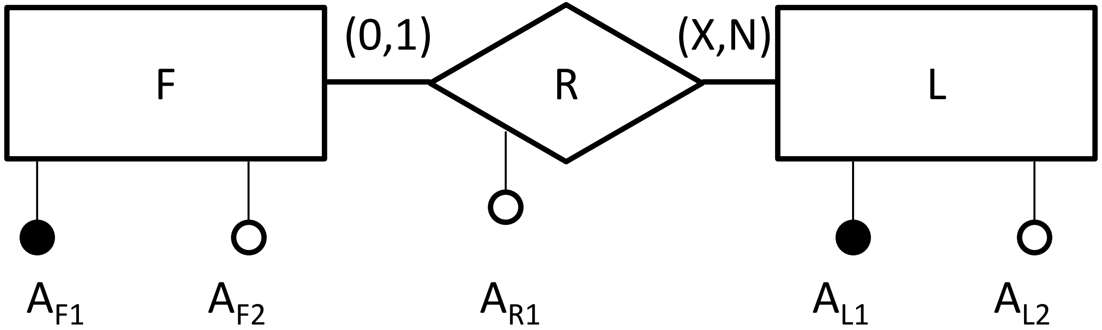
ONE TO MANY RELATIONSHIP WITH MANDORY PARTICIPATION



$\mathbf{F}(\underline{A_{F1}}, A_{F2}, A_{L1}, A_{R1})$
$\mathbf{L}(\underline{A_{L1}}, A_{L2})$

Logical Design
Transformation rules

ONE TO MANY RELATIONSHIP WITH OPTIONAL PARTICIPATION

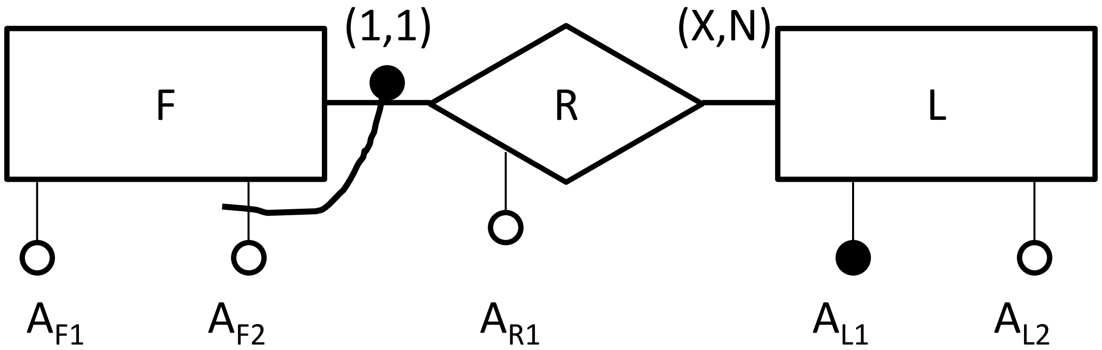


$\mathbf{F}(\underline{A_{F1}}, A_{F2})$
$\mathbf{L}(\underline{A_{L1}}, A_{L2})$
$\mathbf{R}(\underline{A_{F1}}, \underline{A_{L1}}, A_{R1})$

$\mathbf{F}(\underline{A_{F1}}, A_{L1}, A^*_{L1}, A^*_{R1})$
$\mathbf{L}(\underline{A_{L1}}, A_{L2})$

Logical Design
Transformation rules

RELATIONSHIP WITH EXTERNA DENTIFIER

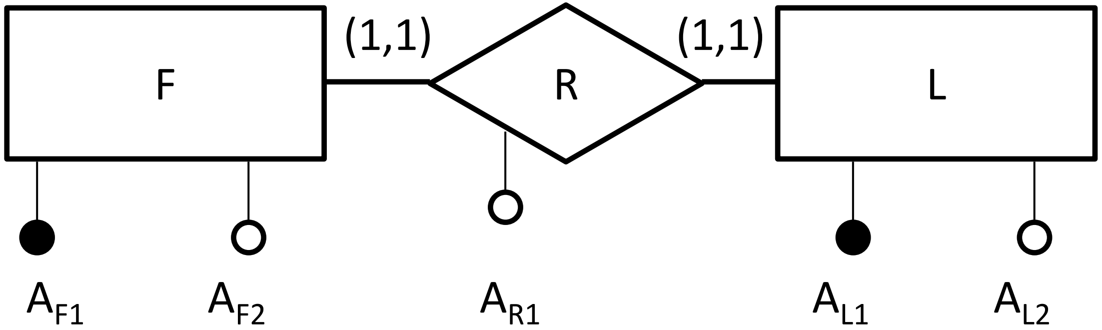


F($\underline{A_{F2}}$, $\underline{A_{L1}}$, A_{F1}, A_{R1})
L($\underline{A_{L1}}$, A_{L2})

Logical Design

Transformation rules

ONE TO ONE RELATIONSHIP WITH MANDATORY PARTICIPATION FOR BOTH ENTITIES



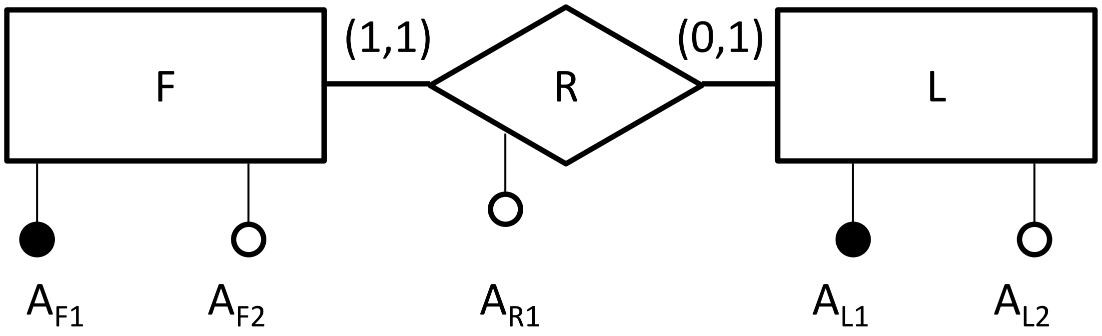
$\mathbf{F}(\underline{A_{F1}}, A_{F2}\ A_{L1}, R_{R1})$
$\mathbf{L}(\underline{A_{L1}}, A_{L2})$

$\mathbf{F}(\underline{A_{F1}}, A_{F2})$
$\mathbf{L}(\underline{A_{L1}}, A_{L2}\ A_{F1}, R_{R1})$

Logical Design

Transformation rules

ONE TO ONE RELATIONSHIP WITH OPTIONAL PARTICIPATION FOR ONE ENTITY

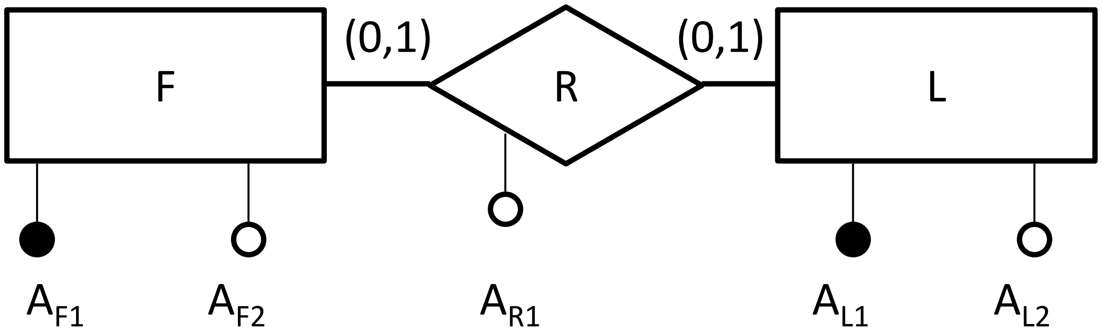


F($\underline{A_{F1}}$, A_{F2}, A_{L1}, R_{R1})
L($\underline{A_{L1}}$, A_{L2})

Logical Design

Transformation rules

ONE TO ONE RELATIONSHIP WITH OPTIONAL PARTICIPATION FOR BOTH ENTITIES



F($\underline{A_{F1}}$, A_{L1})
L($\underline{A_{L1}}$, A_{L2} , A^*_{F1} , A^*_{R1})

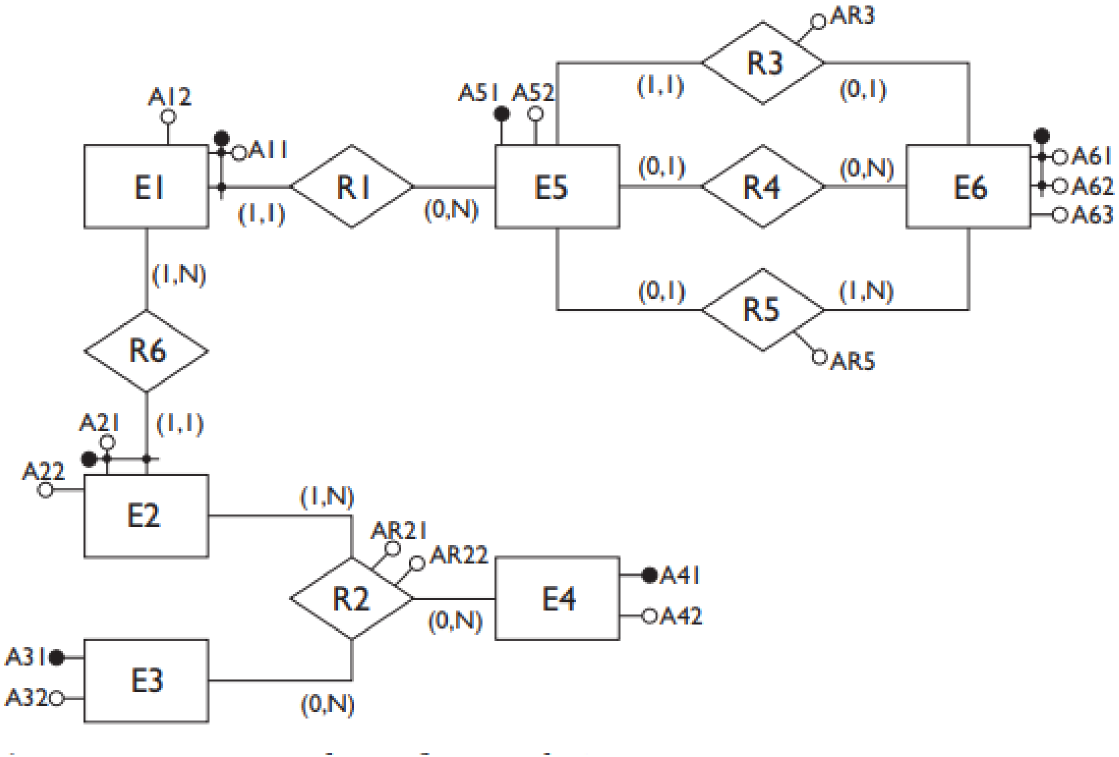
F($\underline{A_{F1}}$, A_{F2}, A^*_{L1} , A^*_{R1})
L($\underline{A_{L1}}$, A_{L2})

F($\underline{A_{F1}}$, A_{F2})
L($\underline{A_{L1}}$, A_{L2})
R($\underline{A_{F1}}$, A_{L1} , A_{R1})

LOGICAL DESIGN:
Translation into the relational model
SAMPLE

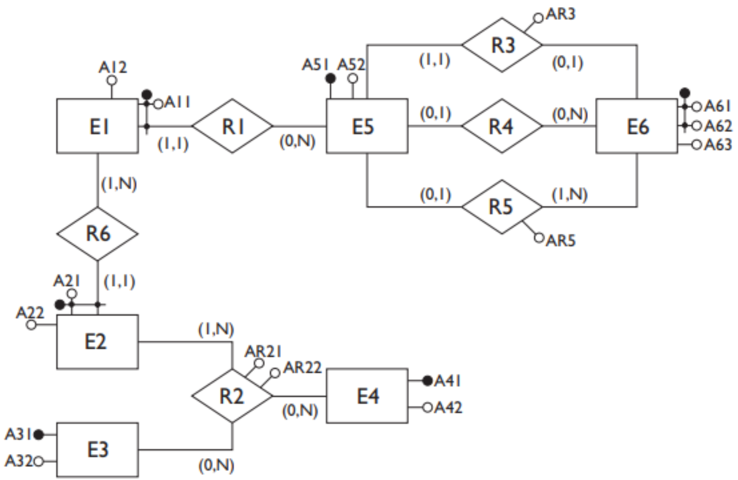
Logical Design

Traslation: sample



Logical Design

Traslation: step 1



Translate each entity into a relation.

The **translation of the entities with internal identifiers** is immediate:

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

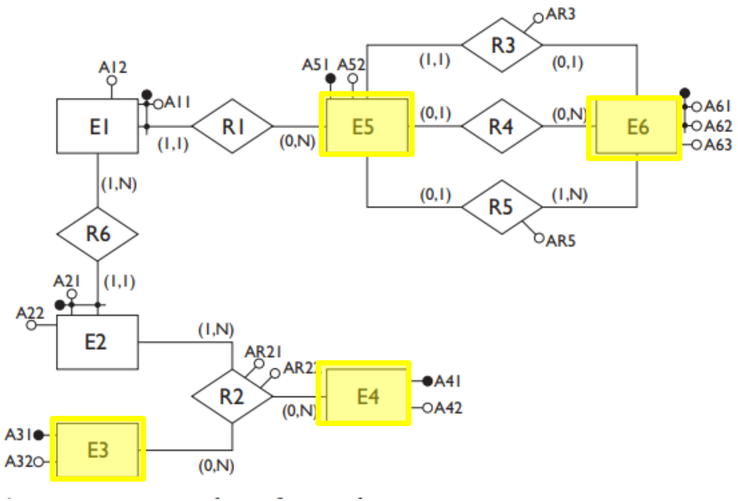
E4(<u>A41</u>,A42)

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

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

Logical Design

Traslation: step 2

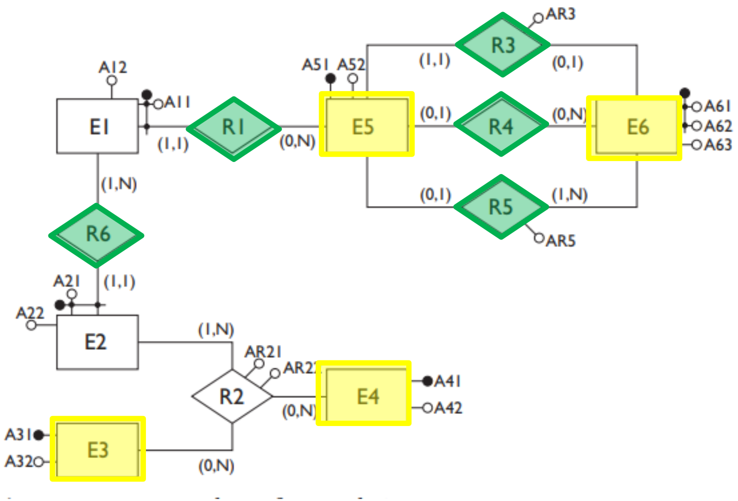


Transform 1-1 and Optional 1-1 Relationships

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

Logical Design

Traslation: step 3

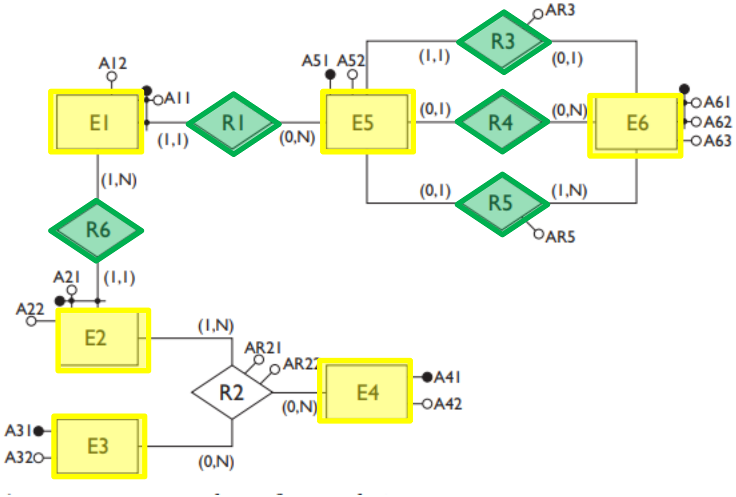


Entities with External Identifiers

E1(A11, A51, A12)
E2(A21, A11, A51, A22)

Logical Design

Traslation: step 4

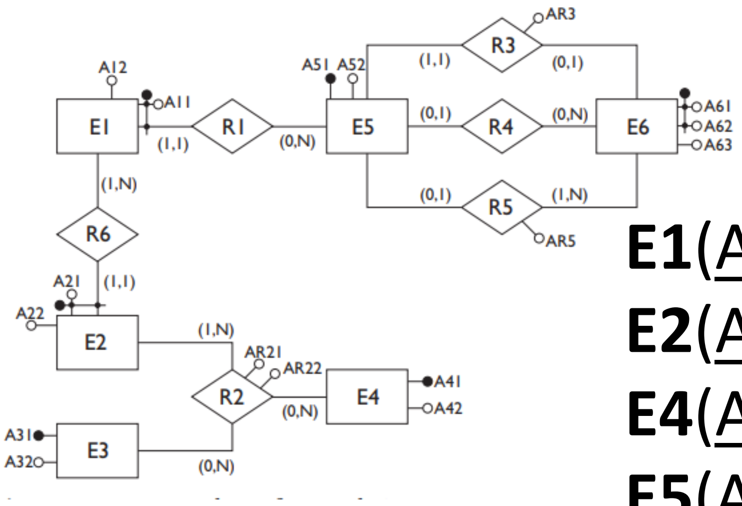


Many-to-Many Relationships

R2(A21, A11, A51, A31, A41, AR21, AR22)

Logical Design

Traslation: final result



E1(A11, A51, A12)

E2(A21, A11, A51, A22) E3(A31, A32)

E4(A41,A42)

E5(A51, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

E6(A61, A62, A63)

R2(A21, A11, A51, A31, A41, AR21, AR22)


End