The Entity/Relationship (ER) Model & DB Design

Programming on the Web

CSC309 Winter 2016

Thanks to Ryan Johnson, John Mylopoulos, Arnold Rosenbloom and Renee Miller for material in these slides

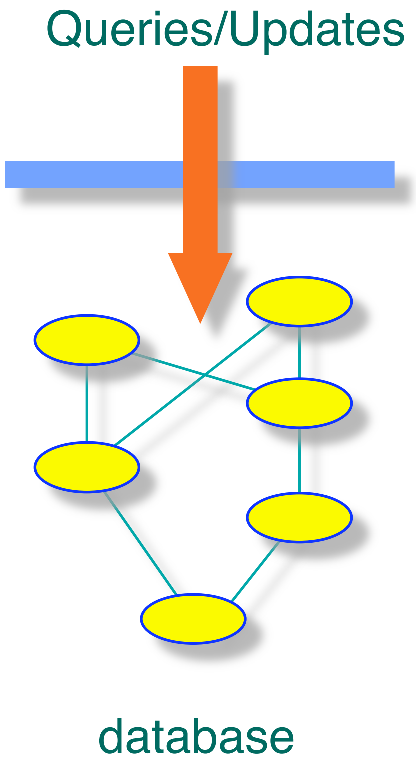
Overview

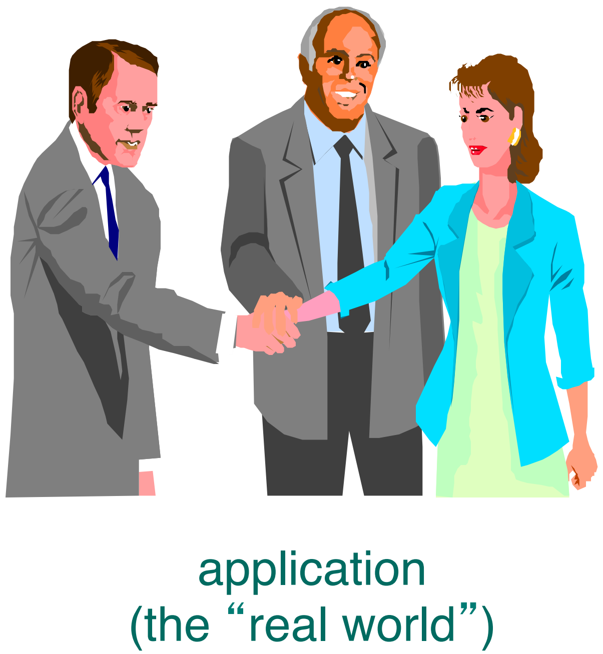
- The Entity/Relationship (ER) Model
- Designing a database schema
  - Restructuring of an E/R model
  - Translation of an E/R model into the logical model (DB Schema)

THE ENTITY/RELATIONSHIP (ER) MODEL

Conceptualizing the real-world

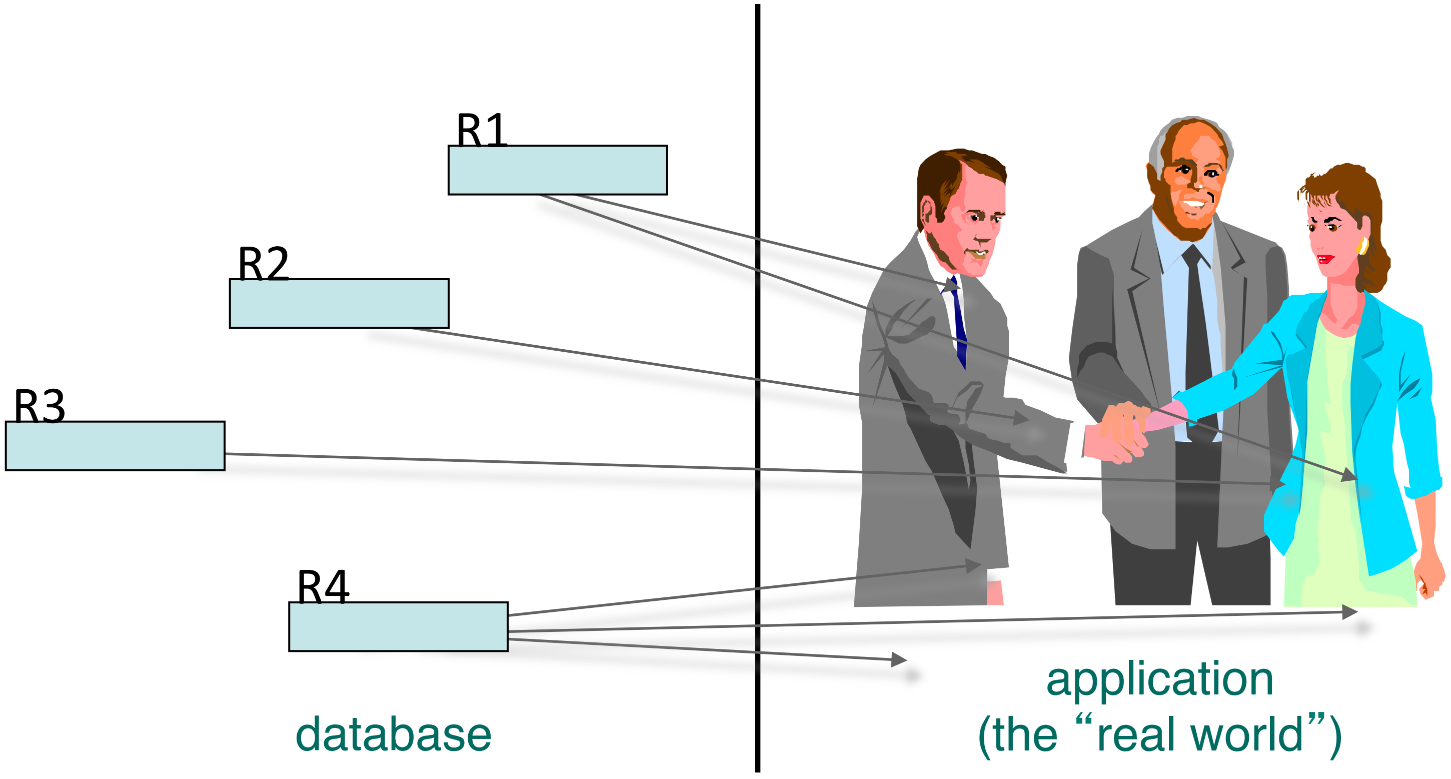
Modeling is about mapping entities and relationships of the world into the concepts of a database





Mapping is Not Deterministic

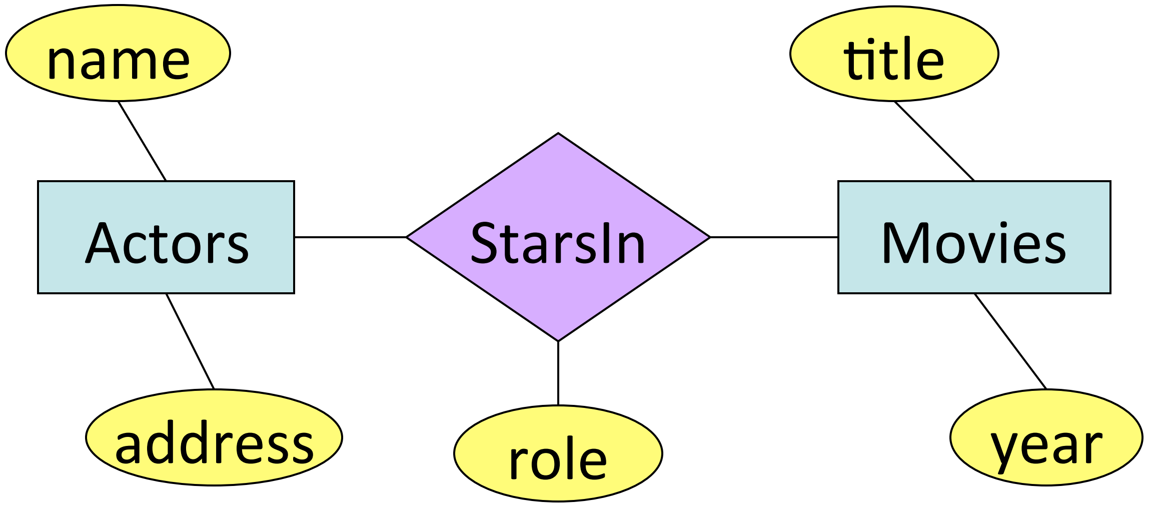
The Relational Model uses relations to represent entities, relationships, or combinations thereof



The mapping process is not always clear

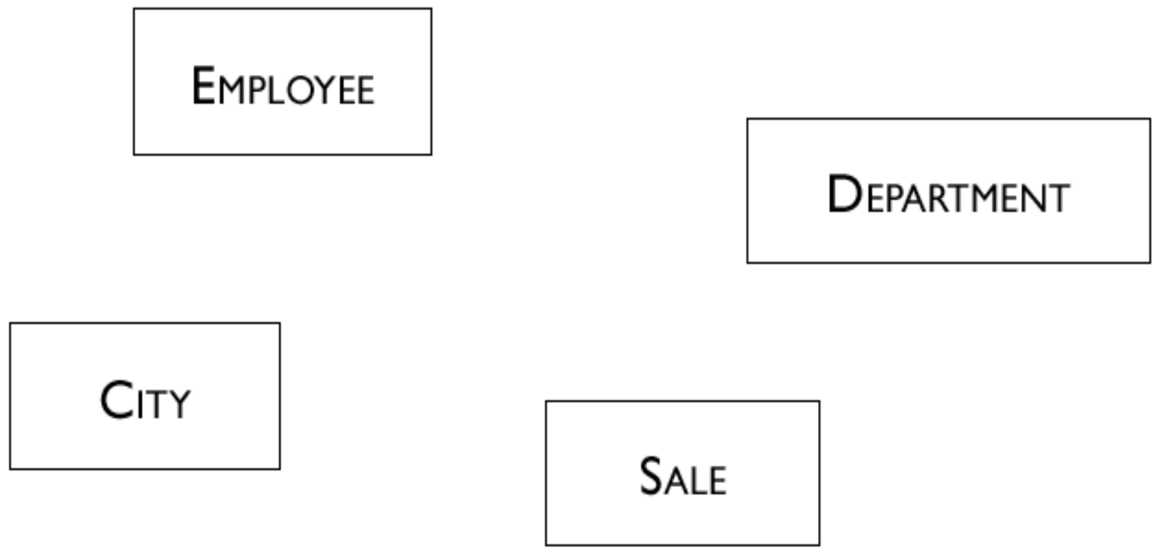
Entity/Relationship Model

- Visual data model (diagram-based)
  - Quickly "chart out" a database design
  - Easier to "see" big picture
  - Comparable to class diagrams in UML
- Basic concept: *entities* and their *relationships*, along with the *attributes* describing them



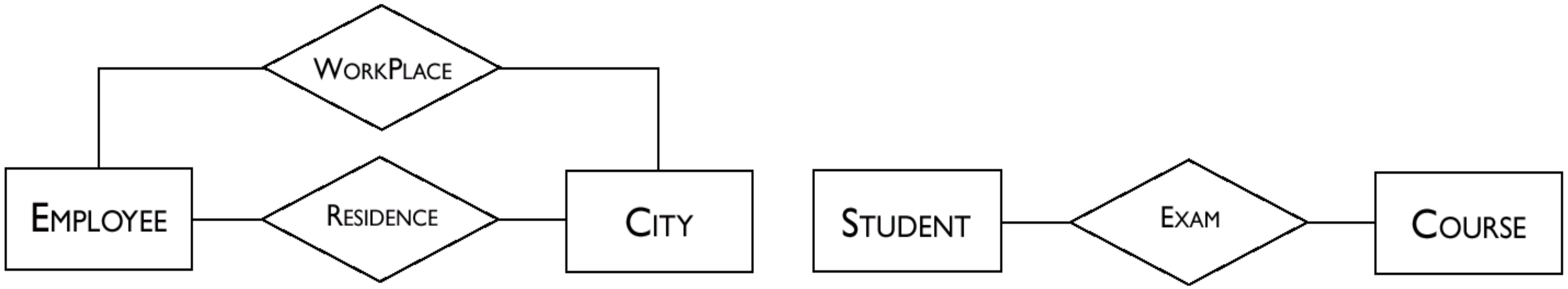
Entity Sets

- An entity set represents a class of objects that have properties in common and an autonomous existence (e.g., City, Department, Employee, Sale)
- An entity is an instance of an entity set (e.g., Stockholm is a City; Peterson is an Employee)

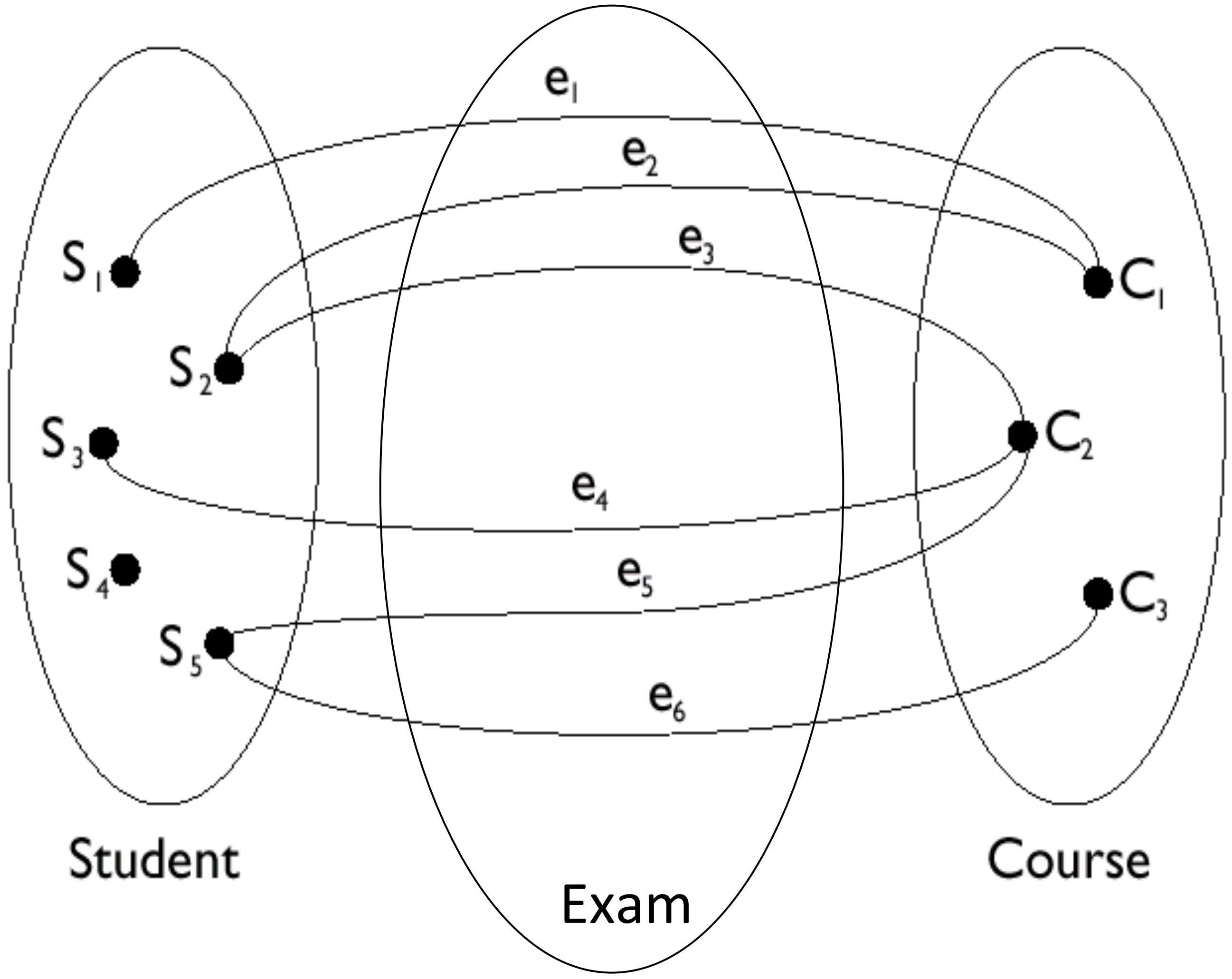


Relationship Sets

- A relationship set is an association between 2+ entity sets (e.g., Residence is a relationship set between entity sets City and Employee)
- A relationship is an instance of a n-ary relationship set (e.g., the pair <Johanssen, Stockholm> is a relationship instance of Residence)



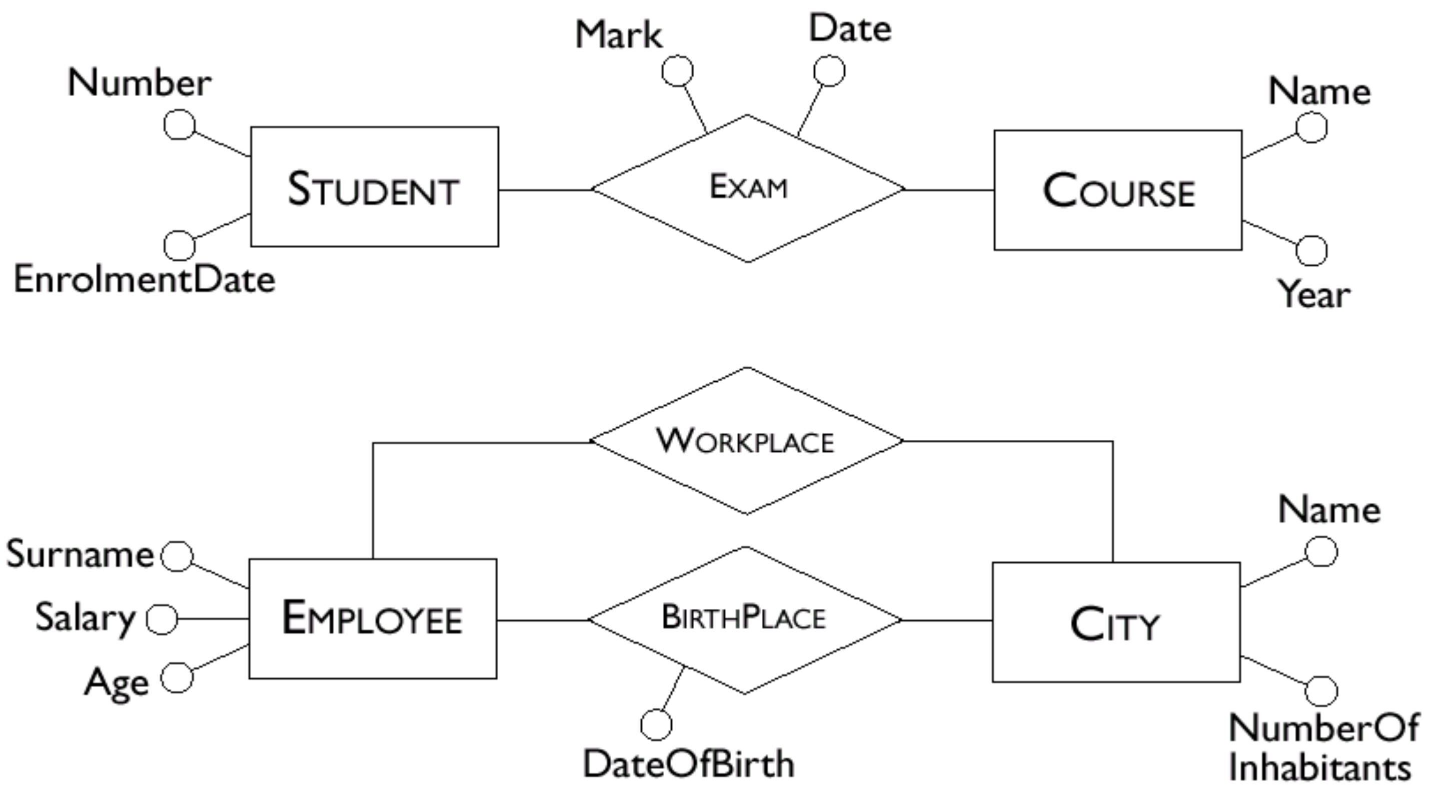
Example of Instances for Exam



A student can’t take more than one exam for a particular course

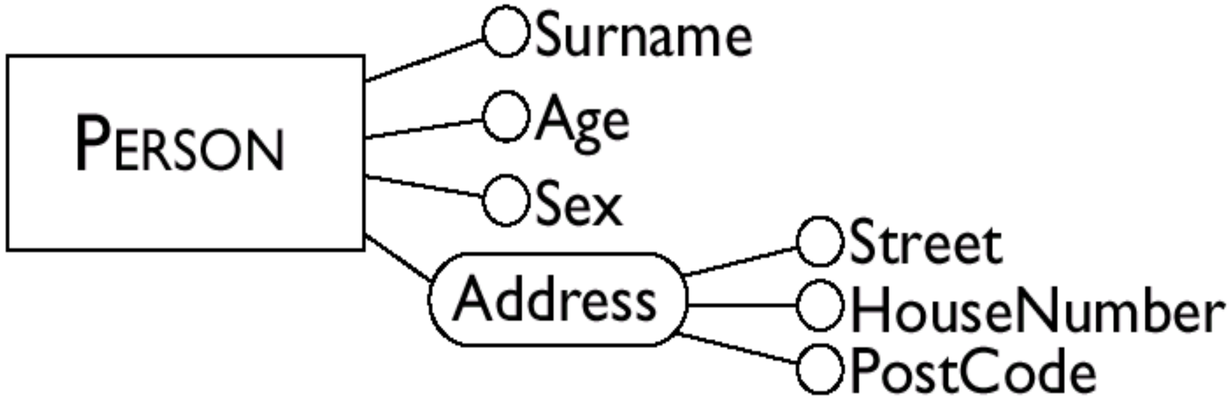
Attributes

- Describe elementary properties of entities or relationships (e.g., Surname, Salary and Age are attributes of Employee)
- May be single-valued, or multi-valued

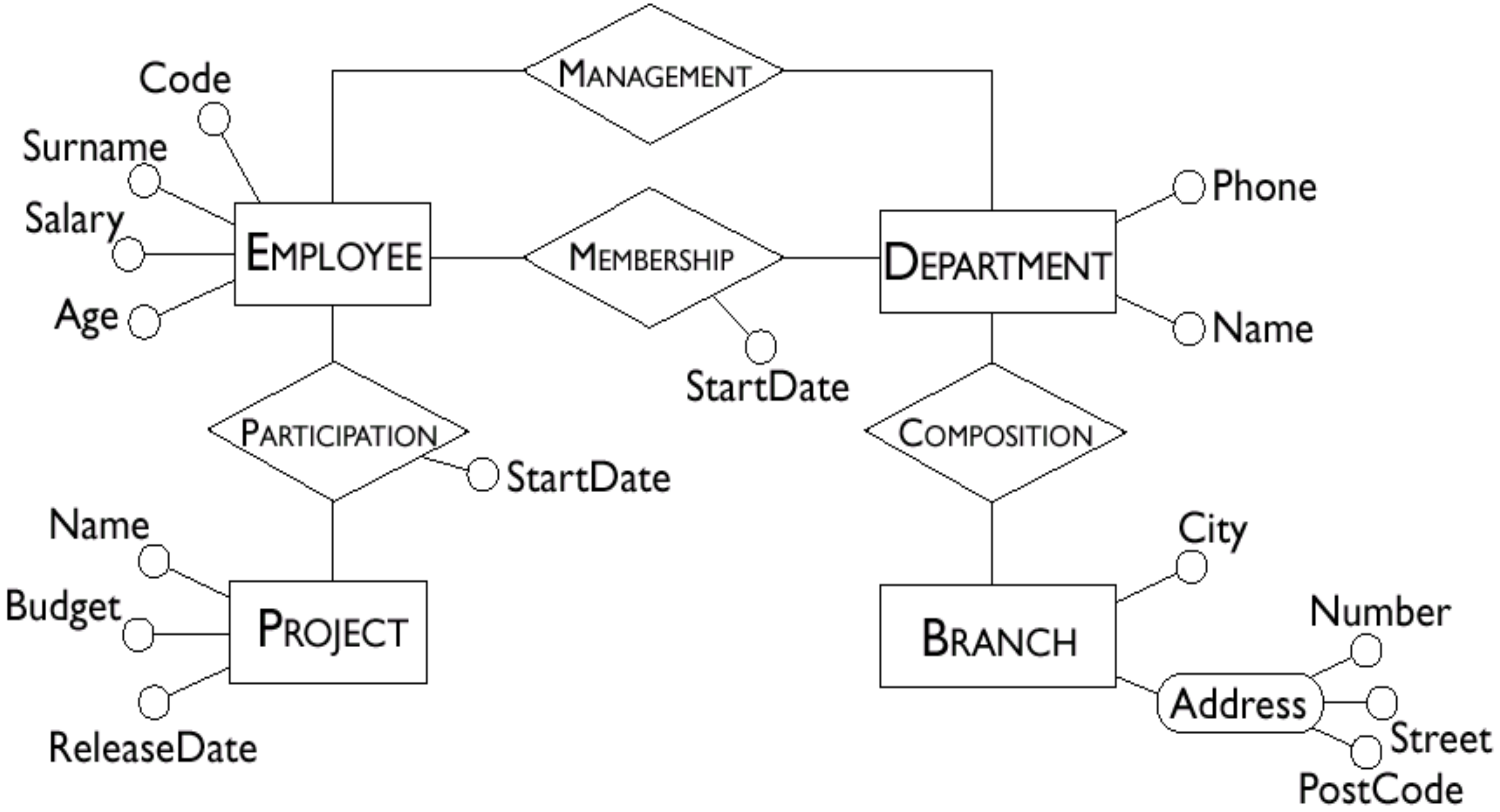


Composite Attributes

- composite attributes are grouped attributes of the same entity or relationship that have closely connected meaning or uses

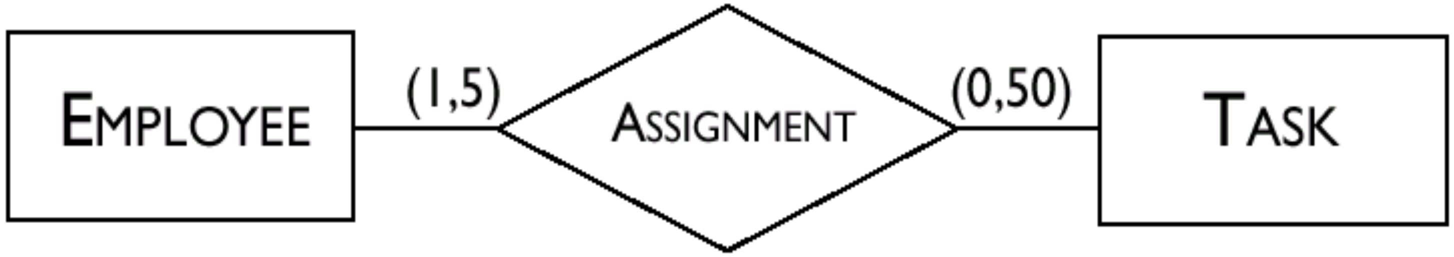


Example Schema with Attributes



Cardinalities

- Each entity set participates in a relationship set with a minimum (min) and a maximum (max) cardinality
- Cardinalities constrain how entity instances participate in relationship instances
- Graphical representation in E/R Diagrams: pairs of (min, max) values for each entity set

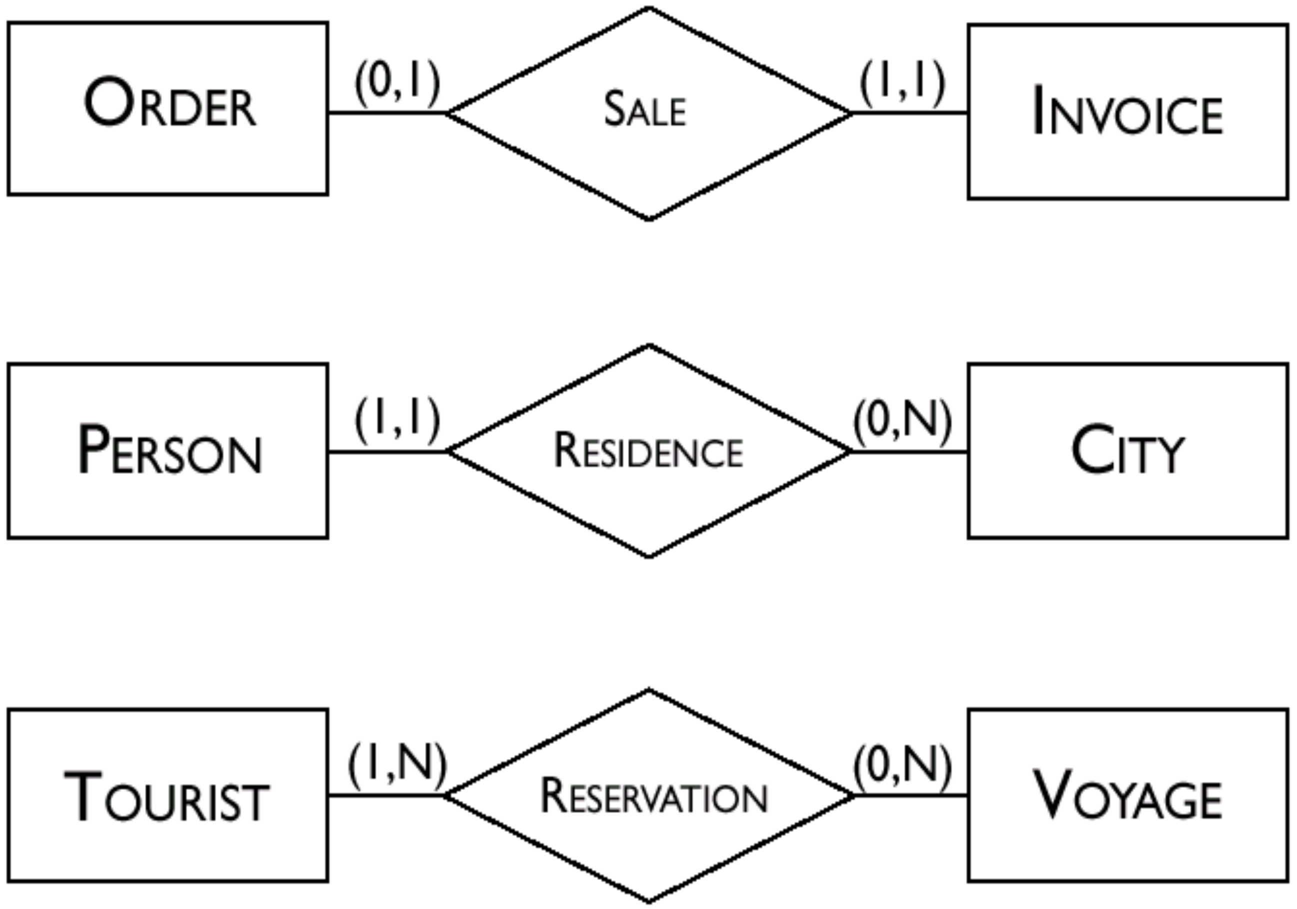


An entity might not participate in any relationship

Cardinalities (cont.)

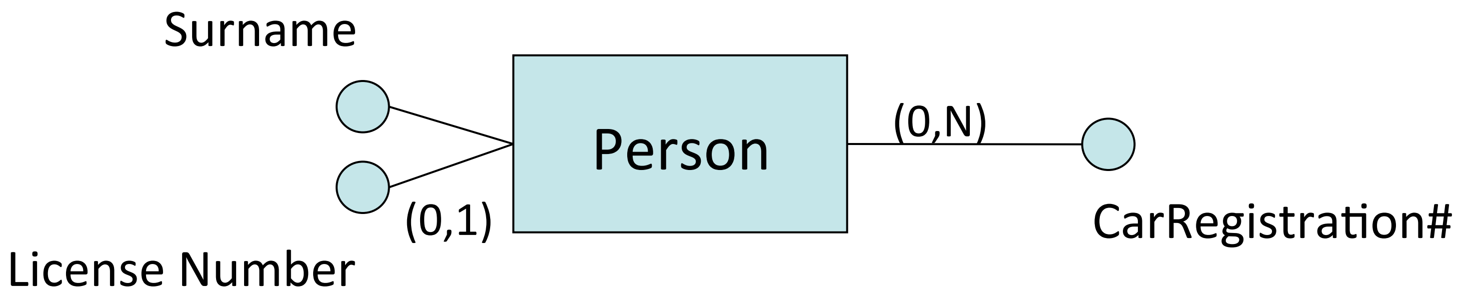
- In principle, cardinalities are pairs of non-negative integers (n, N) such that $n \leq N$, where N means "any number"
- minimum cardinality n:
  - If 0, entity participation in a relationship is optional
  - If 1, entity participation in a relationship is mandatory
- maximum cardinality N:
  - If 1, each instance of the entity is associated at most with a single instance of the relationship
  - If N, then each instance of the entity is associated with many instances of the relationship

Cardinality Examples



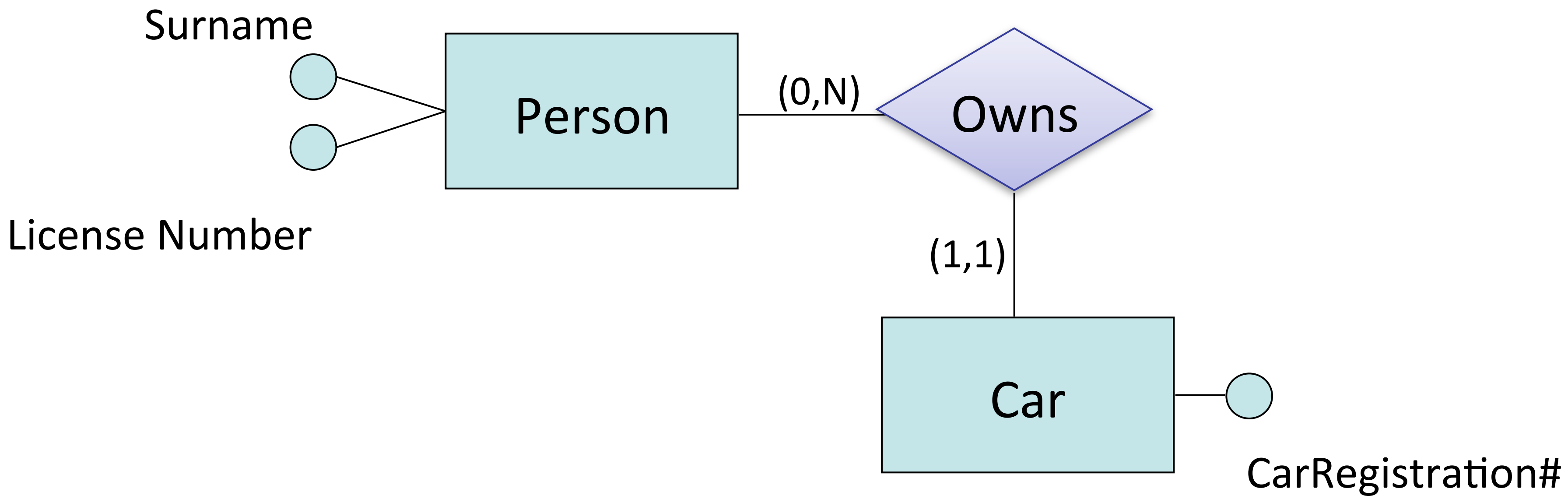
Cardinalities of Attributes

- Describe min/max number of values an attribute can have
- When the cardinality of an attribute is (1, 1) it can be omitted (single-valued attributes)
- The value of an attribute, may also be null, or have several values (multi-valued attributes)



Cardinalities of Attributes (cont.)

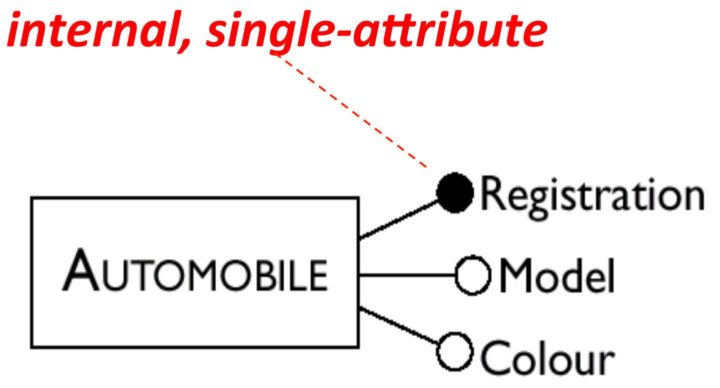
- Multi-valued attributes often represent situations that can be modeled with additional entities. E.g., the ER schema of the previous slide can be revised into:

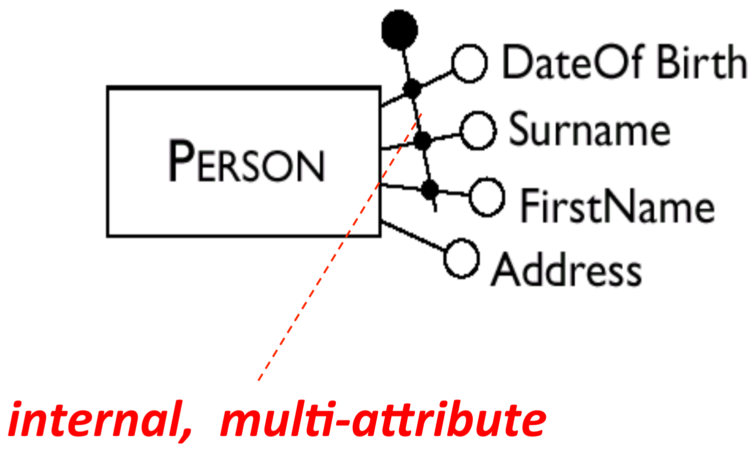


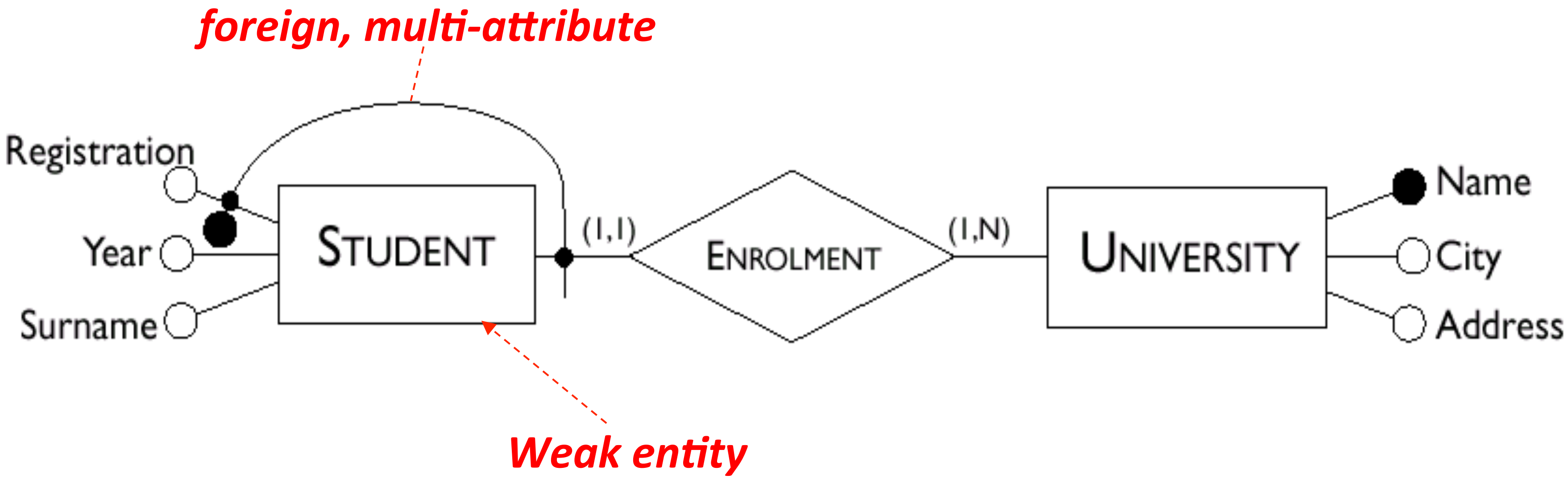
Keys in E/R

- Keys consist of minimal sets of attributes which identify uniquely instances of an entity set
  - socialInsurance# may be a key for Person
  - firstName, middleName, lastName, address may be a key for Person
- In most cases, a key is formed by one or more attributes of the entity itself (*internal* keys)
- Sometimes, other entities are involved in the identification (*foreign keys*, *weak entities*)
- A key for a relationship consists of keys of entities it relates

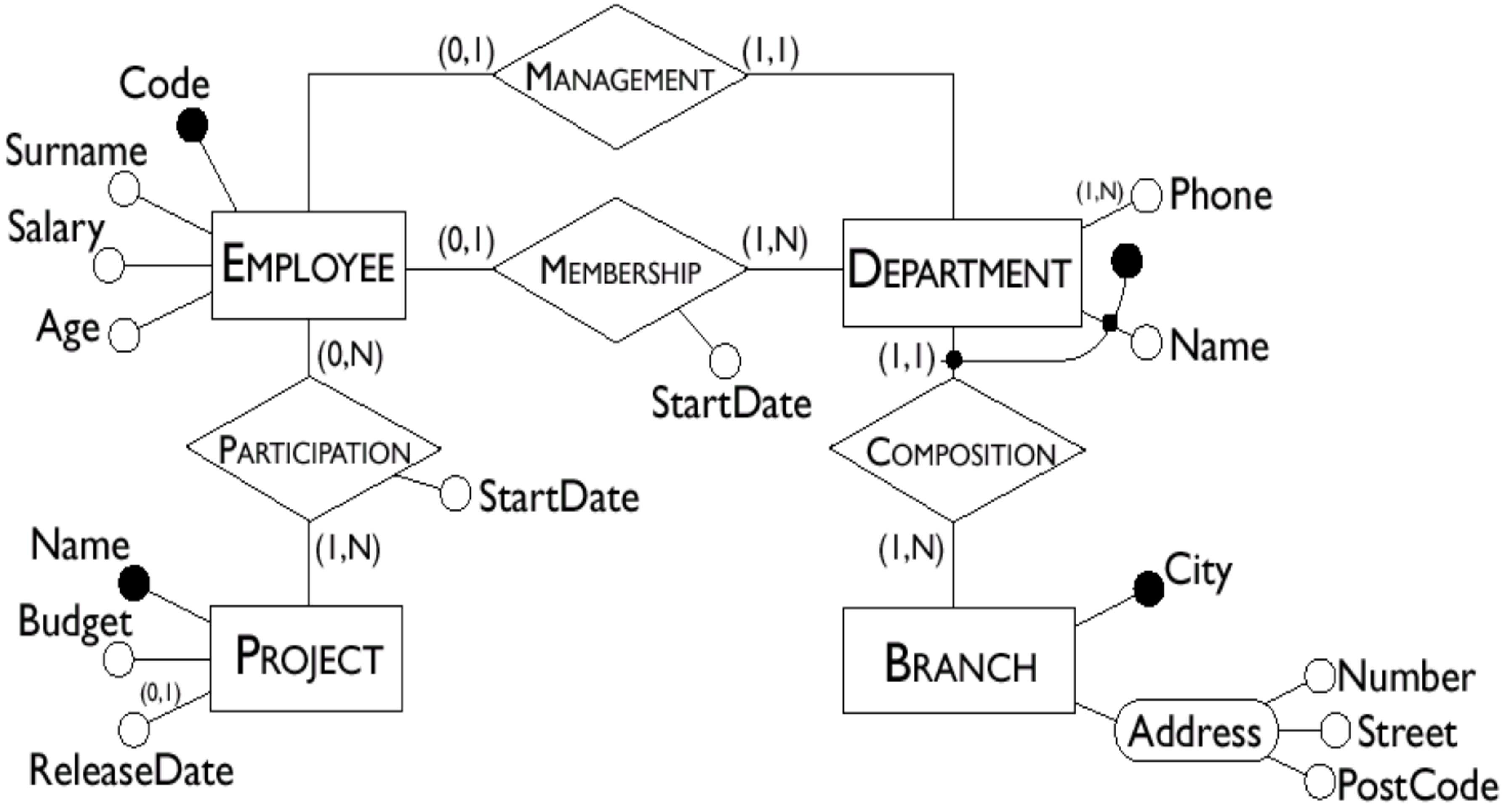
Examples of Keys in E/R





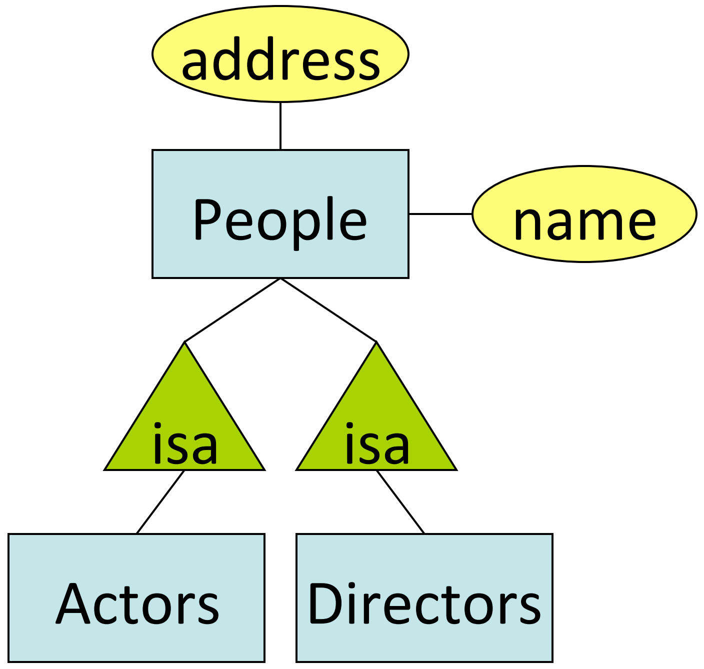


Schema with Cardinalities & Keys



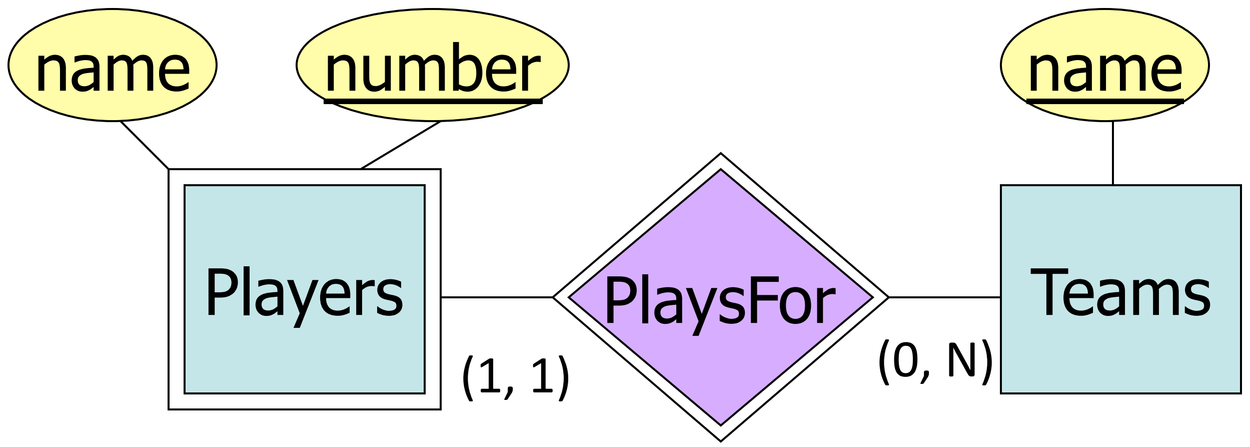
Subclasses in E/R

- Subclass = special case → Inheritance
  - Fewer instances, more attributes (usually)
  - One-one relationship between classes
  - Attributes: union of classes involved



Weak entity sets – example

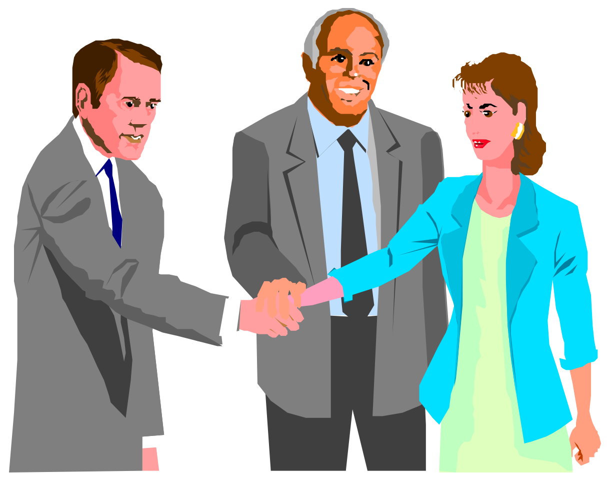
- name is almost a key for football players, but there might be two with the same name
- number is certainly not a key, since players on two teams could have the same number
- But number, together with the team name related to the player by PlaysFor should be unique



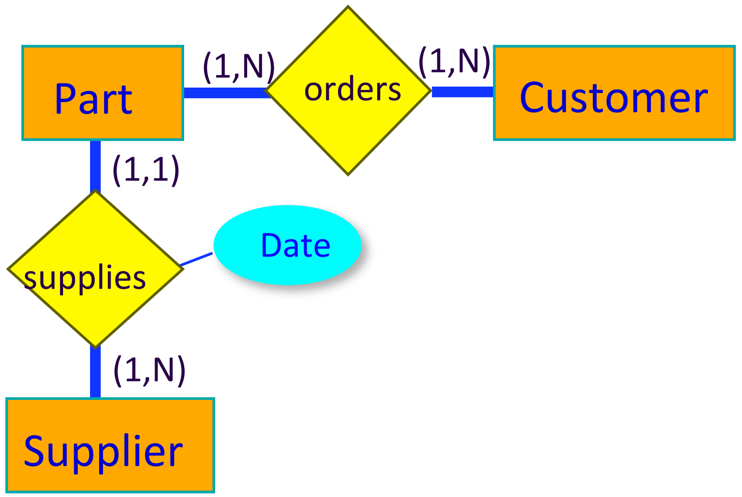
Double rectangle for the weak entity set
Double diamond for *supporting* many-one relationship

Designing a Database Schema

The "real world"



The E/R Model (Conceptual Model)



The Relational Schema

Part (Name,Description,Part#)
Supplier (Name, Addr)
Customer (Name, Addr)
Supplies (Name,Part#, Date)
Orders (Name,Part#)

(Relational) Database Design

- Given a conceptual schema (ER, but could also be UML), generate a logical (relational) schema
- This is *not* just a simple translation from one model to another for two main reasons:
  - not all the constructs of the ER 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 helpful to divide the design into two steps:

- *Restructuring of the Entity-Relationship schema*, based on criteria for the optimization of the schema
- *Translation into the logical model*, based on the features of the logical model (in our case, the relational model)

RESTRUCTURING OF AN E/R MODEL

Restructuring Overview

Input: E/R Schema

Output: Restructured E/R Schema

Restructuring parts:

- Analysis of Redundancies
- Removing Generalizations (Subclasses)
- Partitioning/Merging of Entities and Relations
- Limit the Use of Weak Entity Sets
- Selection of Primary Identifiers (Keys)

Analysis of Redundancies

- *Redundancy* = saying the same thing in two (or more) different ways
- Wastes space and (more importantly) encourages inconsistency
  - Two representations of the same fact become inconsistent if we change one and forget to change the other
- Usually indicates a design flaw as well
  - Example: storing actor's address with movies

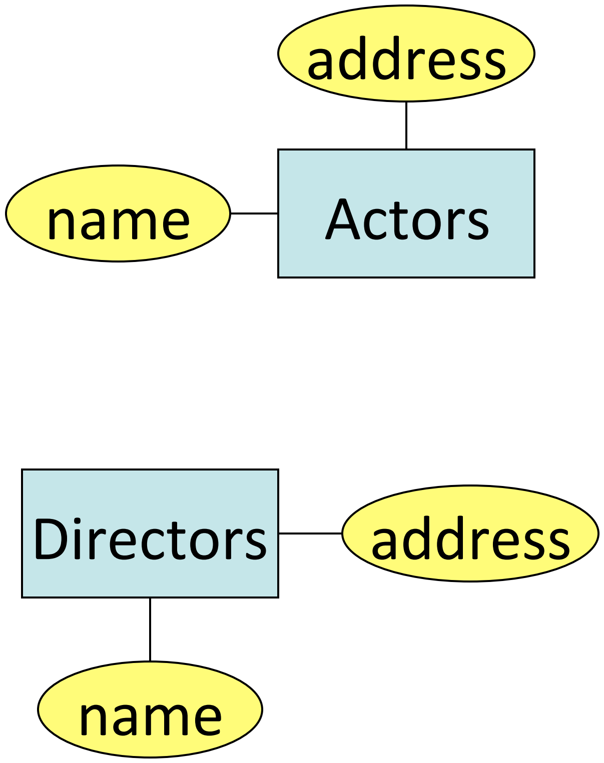
  => Address at time of filming? Now? Hotel near studio?

Two types of redundancy

- Repeated information

name	address	role
James Jones	Villa, CA	Vader
James Jones	Villa, CA	Vader
James Jones	Villa, CA	Vader
James Jones	Villa, CA	Greer
James Jones	Villa, CA	Mustafa

- Repeated designs (same or similar attributes)



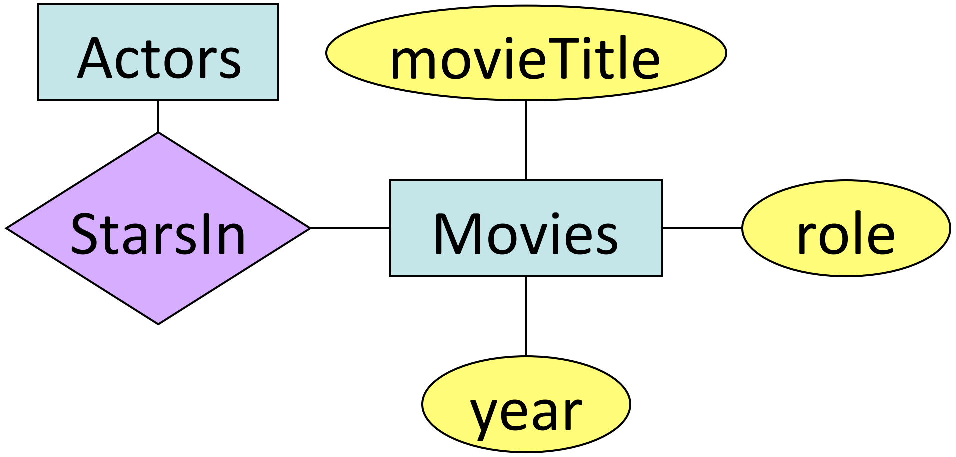
Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.

    or
  - It is the "many" in a many-one or many-many relationship.
- Rules of thumb
  - A "thing" in its own right => Entity Set
  - A "detail" about some other "thing" => Attribute
  - A "detail" correlated among many "things" => Entity Set

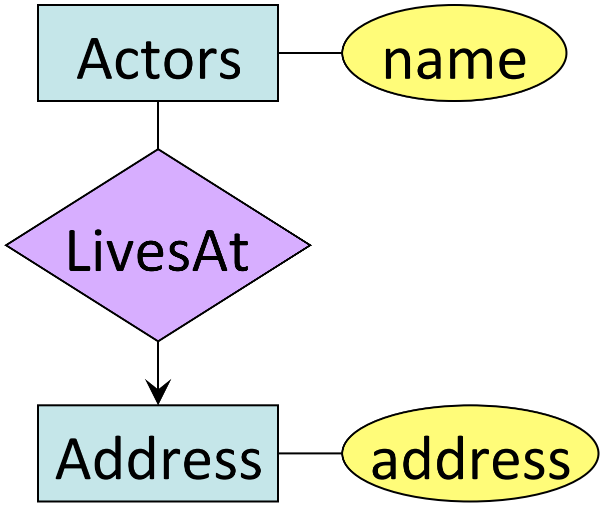
Really this is just about avoiding redundancy

E.S. vs. attributes: bad examples



Many movies, one role?

Many roles, one movie?



Redundant Entity Set

and Relationship Set

Deciding about Redundancy

The presence of a redundancy in a database may be

- an advantage: a reduction in the number of accesses necessary to obtain 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

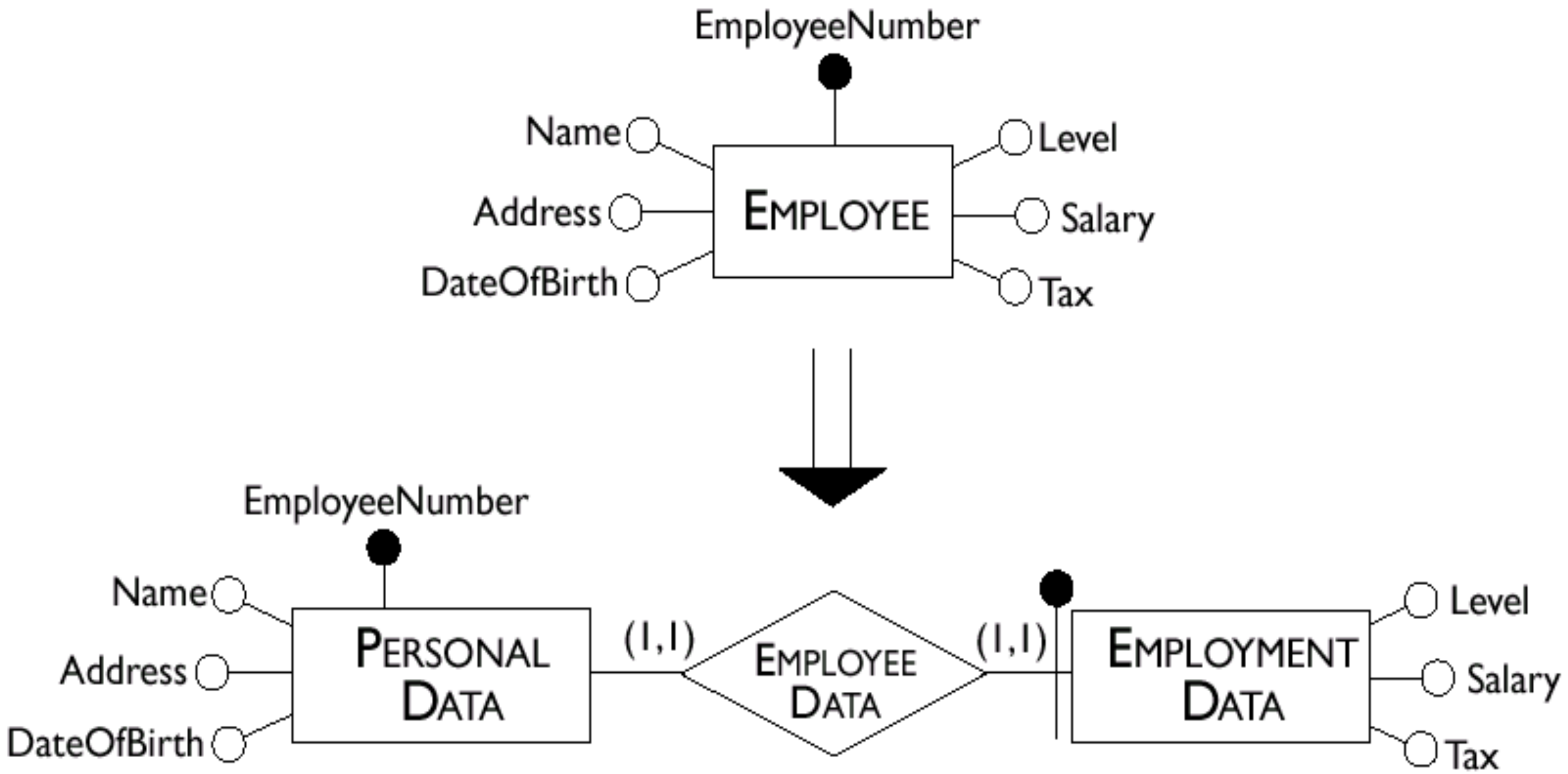
The decision to maintain or eliminate a redundancy is made by comparing the cost of operations that involve the redundant information and the storage needed, in the case of presence or absence of redundancy.

Performance analysis is required to decide about redundancy

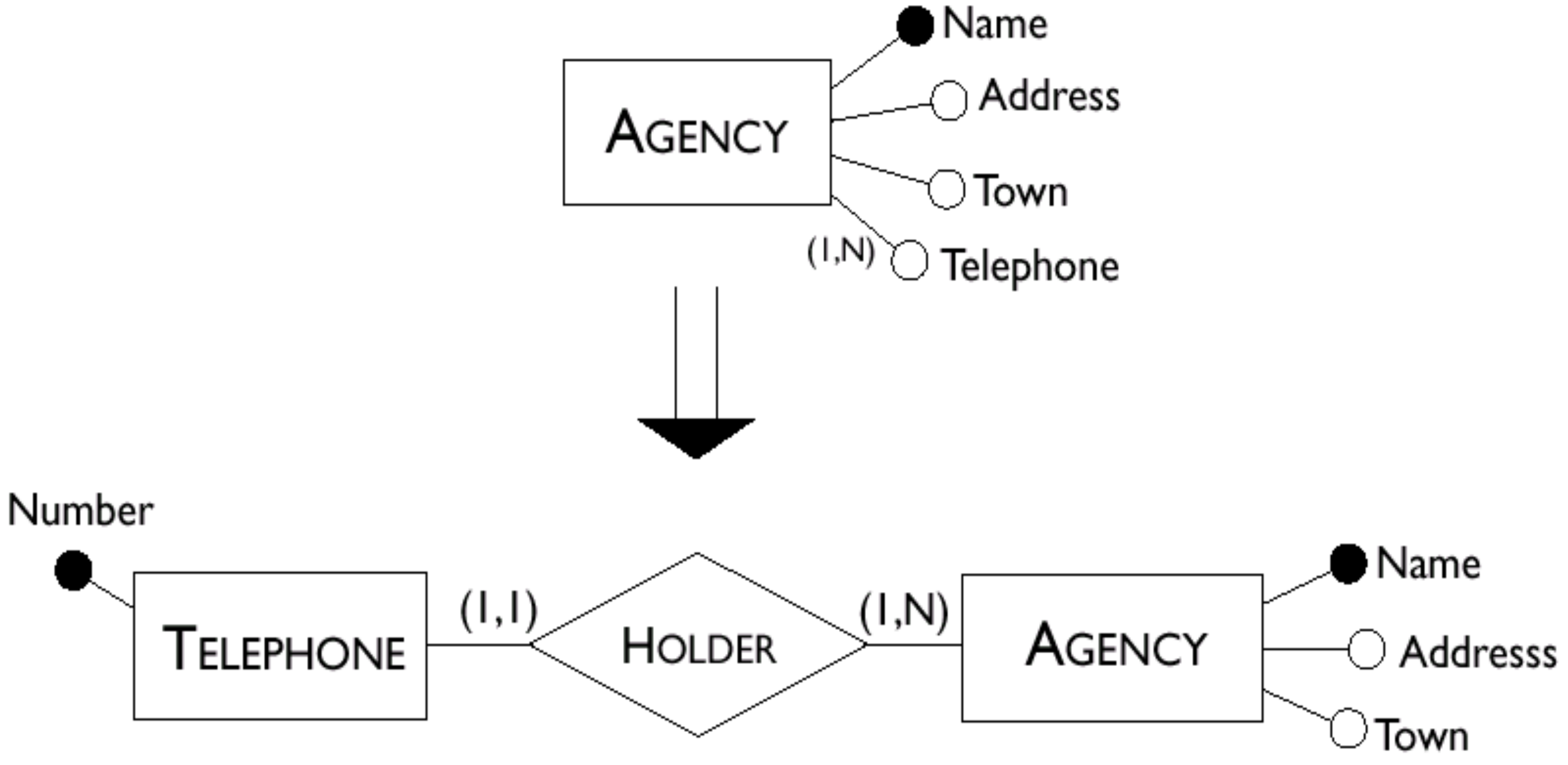
Partitioning and Merging of E/R

- Entities and relationships of an E-R schema can be partitioned or merged to improve the efficiency of operations
- Accesses are reduced by:
  - separating attributes of the same concept that are accessed by different operations and
  - merging attributes of different concepts that are accessed by the same operations

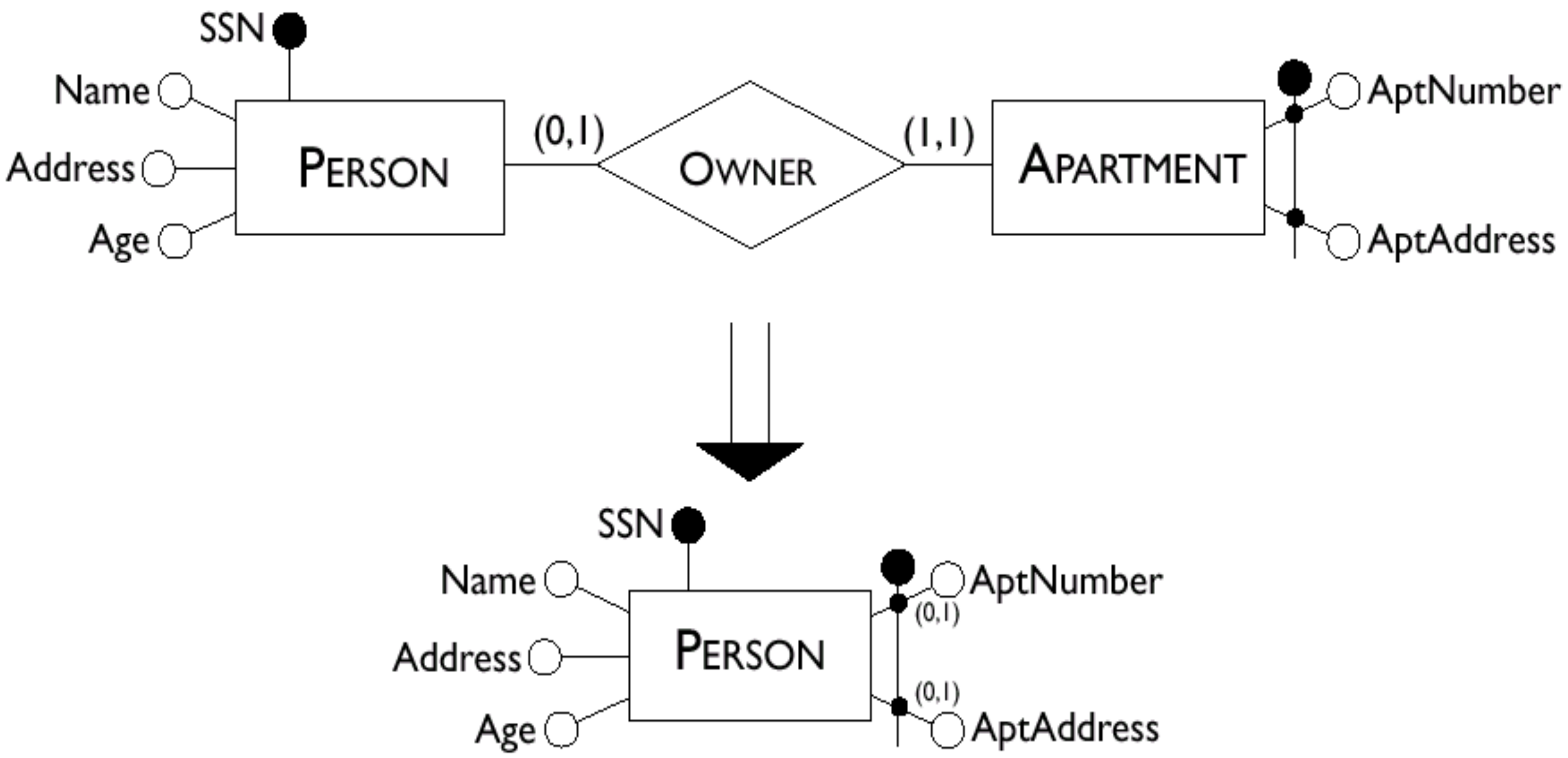
Example of Partitioning



Elimination of Multi-valued Attrib.



Merging Entities



When to use weak entity sets?

- The usual reason is that there is no global authority capable of creating unique ID’s
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world

Don't Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself
  - They make all entity sets weak, supported by all other entity sets to which they are linked
- In reality, each entity gets a unique ID anyway
  - Social insurance number, automobile VIN, etc.
  - Useful for many reasons (next slide)

Selecting a Primary Key

- Every relation must have a unique primary key
- The criteria for this decision are as follows:
  - Attributes with null values cannot form primary keys
  - One/few attributes is preferable to many attributes
  - Internal keys preferable to external ones (weak entities depend for their existence on other entities)
  - A key that is used by many operations to access instances of an entity is preferable to others

Keeping keys simple

Multi-attribute and/or string keys…

- … are redundant
  - e.g. Movies(title, year, …): 2 attributes, ~16 bytes
  - Number of movies ever made << 2^{32} (4 bytes)

  => Integer movieID key saves 75% space and a lot of typing
- … break encapsulation
  - e.g. Patient(firstName, lastName, phone, …)
  - Security/privacy hole

  => Integer patientID prevents information leaks
- … are brittle (nasty interaction of above two points)
  - Name or phone number change? Parent and child with same name?
  - Patient with no phone? Two movies with same title and year?

  => Internal ID always exists, immutable, unique

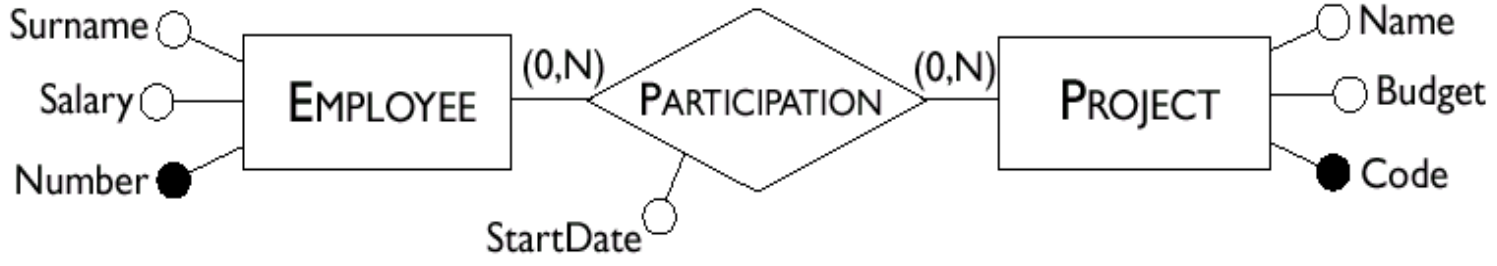
Also: computers are really good at integers…

TRANSLATION OF AN E/R MODEL INTO THE LOGICAL MODEL (DB SCHEMA)

Translation into a Logical Schema

- The second step of logical design consists of a translation between different data models
- Starting from an E-R schema, an equivalent relational schema is constructed
  - “equivalent”: 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

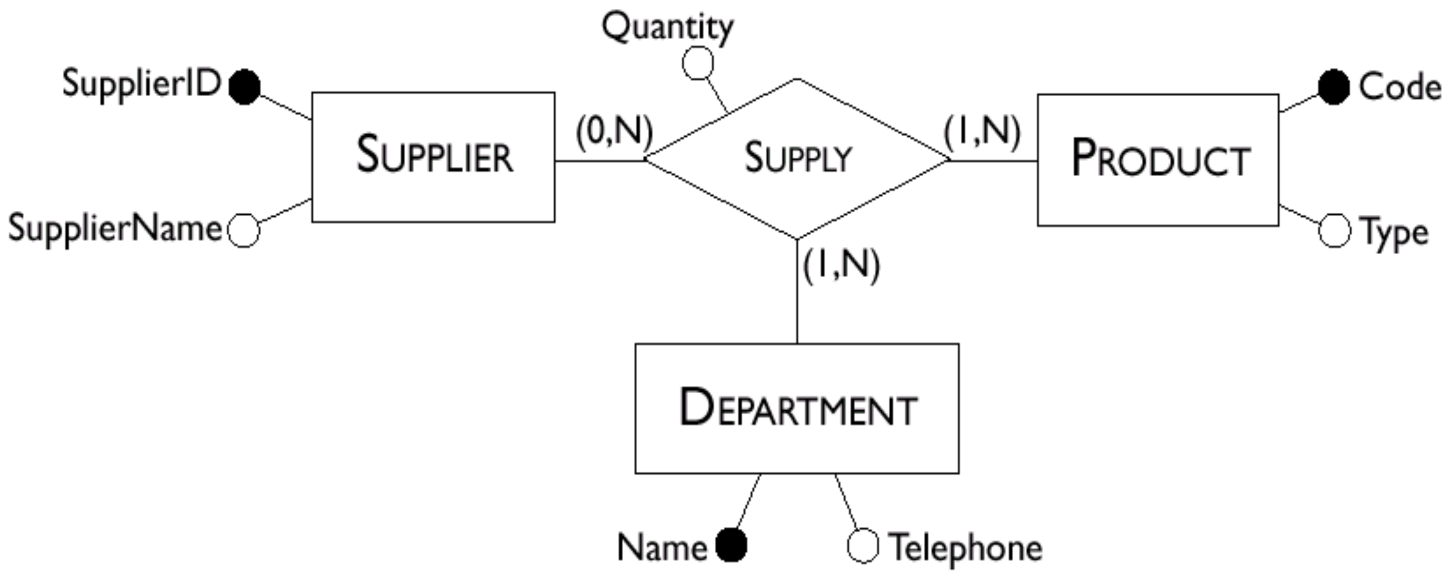


Employee(<u>Number</u>, Surname, Salary)

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

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

Ternary Relationships



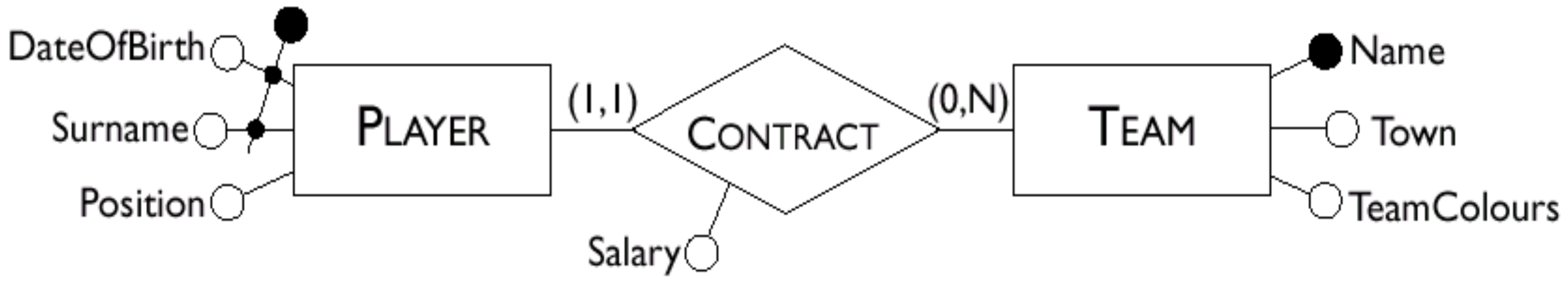
Supplier(<u>SupplierID</u>, SupplierName)

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

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

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

One-to-Many Relationships

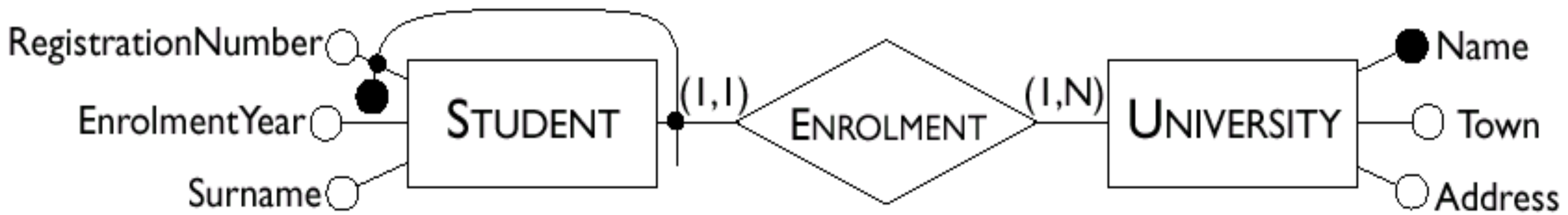


Player(<u>Surname</u>,<u>DateOfBirth</u>, Position)
Team(<u>Name</u>, Town, TeamColours)
Contract(<u>Surname</u>, <u>DateOfBirth</u>, TeamName, Salary)

OR

Player(<u>Surname</u>,<u>DateOfBirth</u>, Position, TeamName, Salary)
Team(<u>Name</u>, Town, TeamColours)

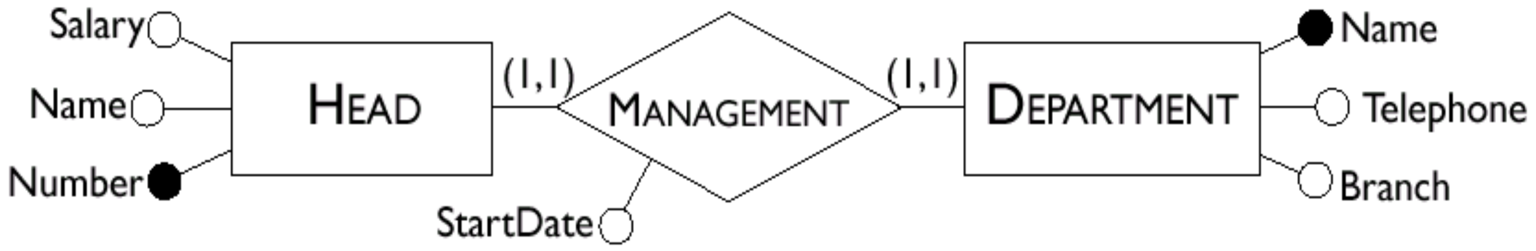
Weak Entities



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)

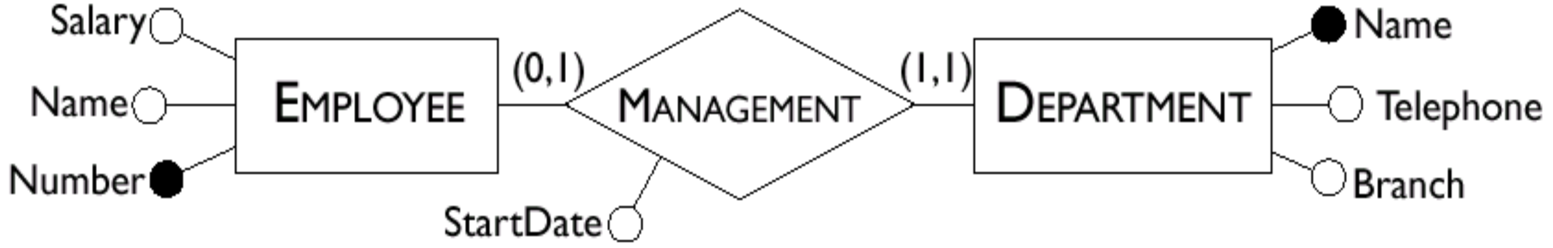
Department(<u>Name</u>, Telephone, Branch)

Or

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

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

Optional One-to-One Relationships

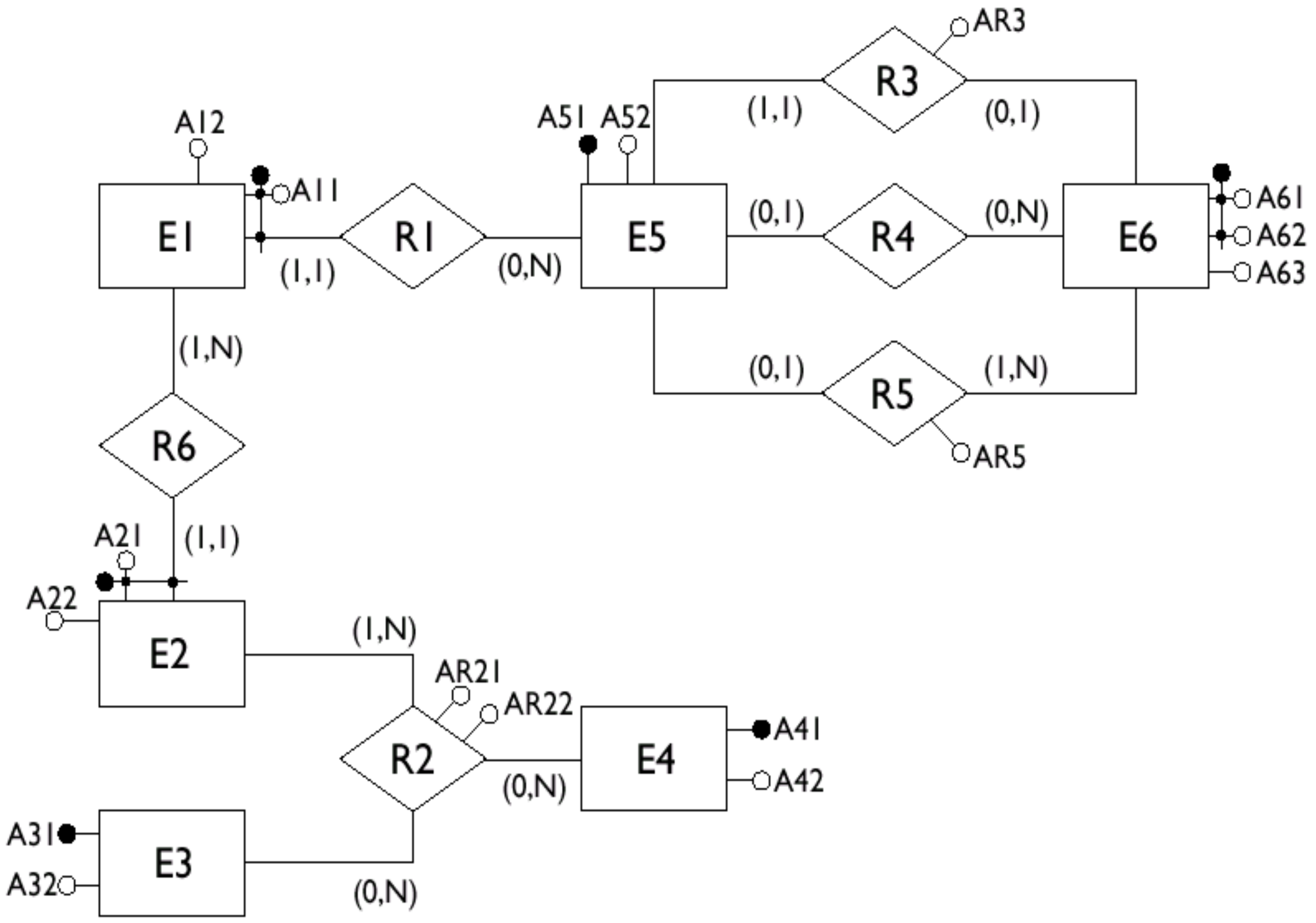


Employee(<u>Number</u>, Name, Salary)
Department(<u>Name</u>, Telephone, Branch, Number, StartDate)

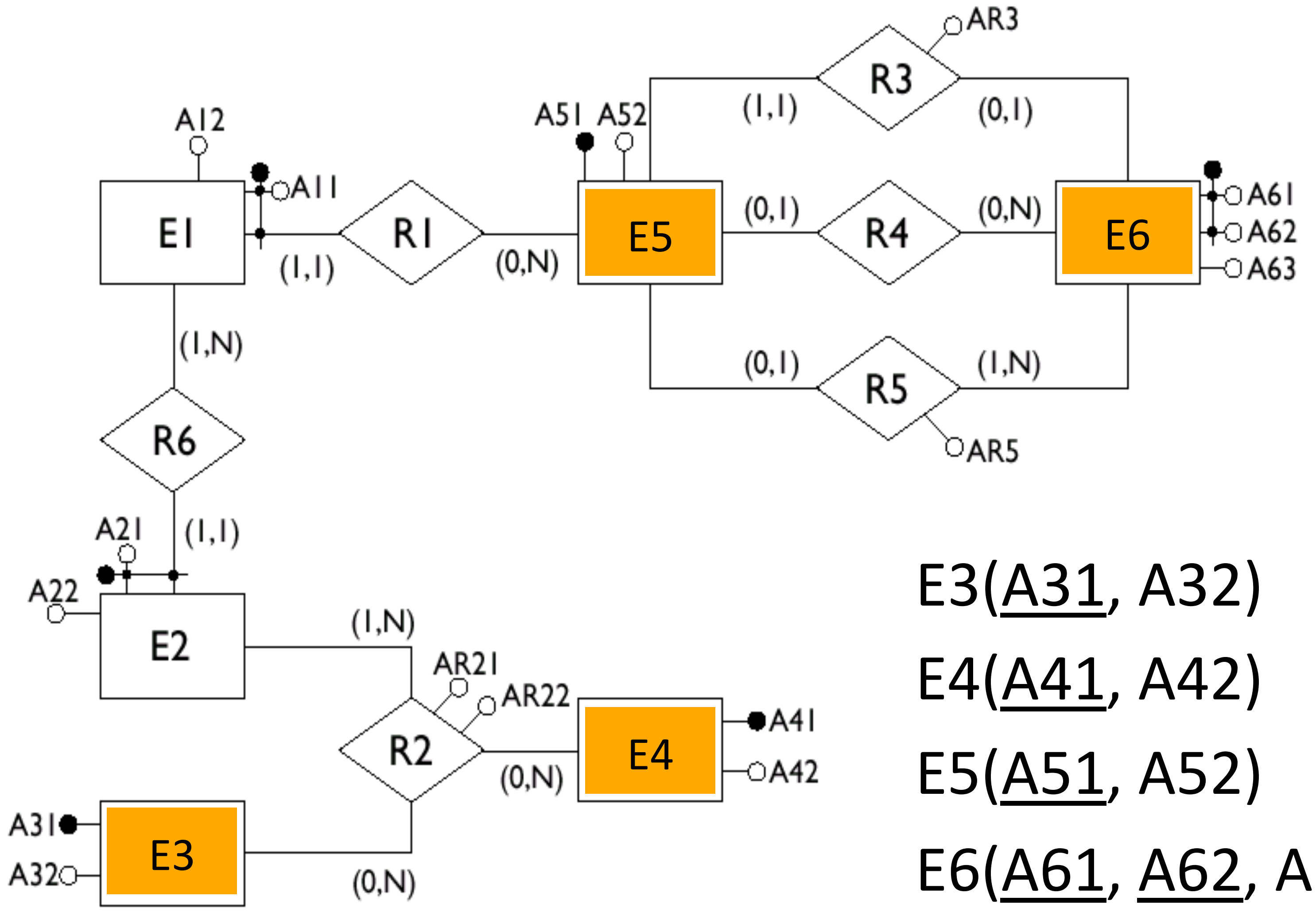
Or, if both entities are optional

Employee(<u>Number</u>, Name, Salary)
Department(<u>Name</u>, Telephone, Branch)
Management(Emp<u>Number</u>, DeptName, StartDate)

A Sample ER Schema



Entities with Internal Identifiers



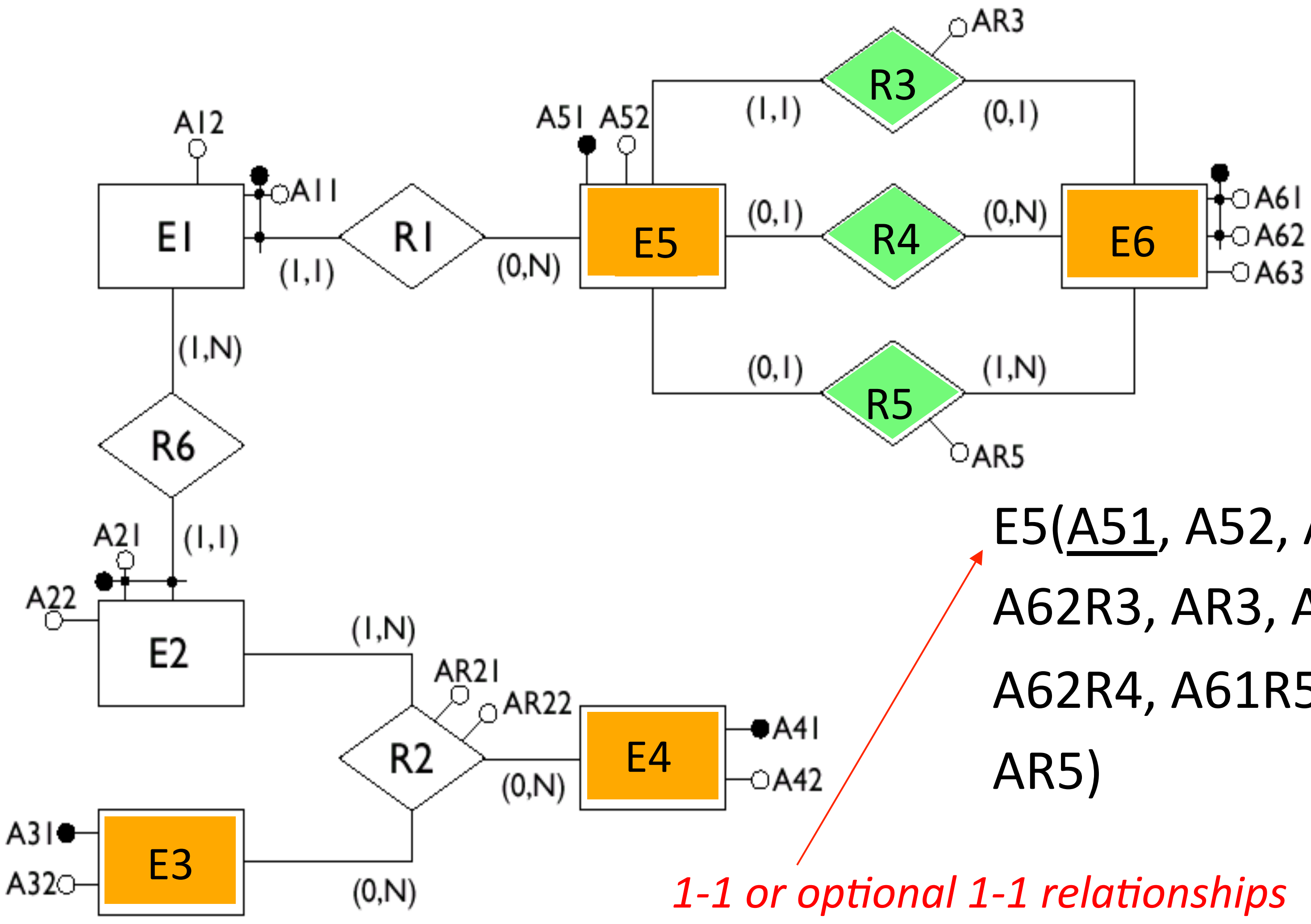
E3(<u>A31</u>, A32)

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

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

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

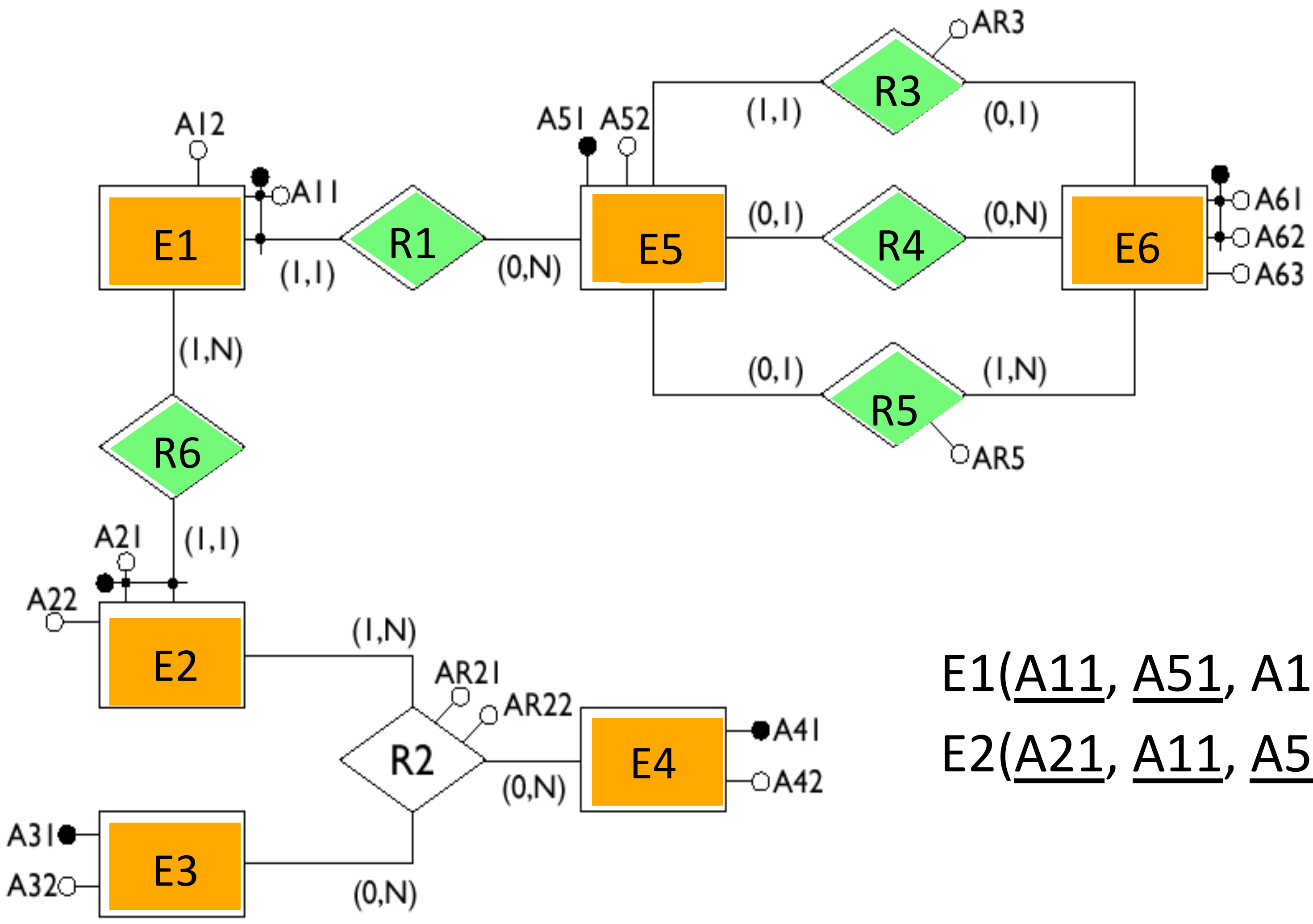
1-1 and Optional 1-1 Relationships



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

1-1 or optional 1-1 relationships
Can lead to messy transformations

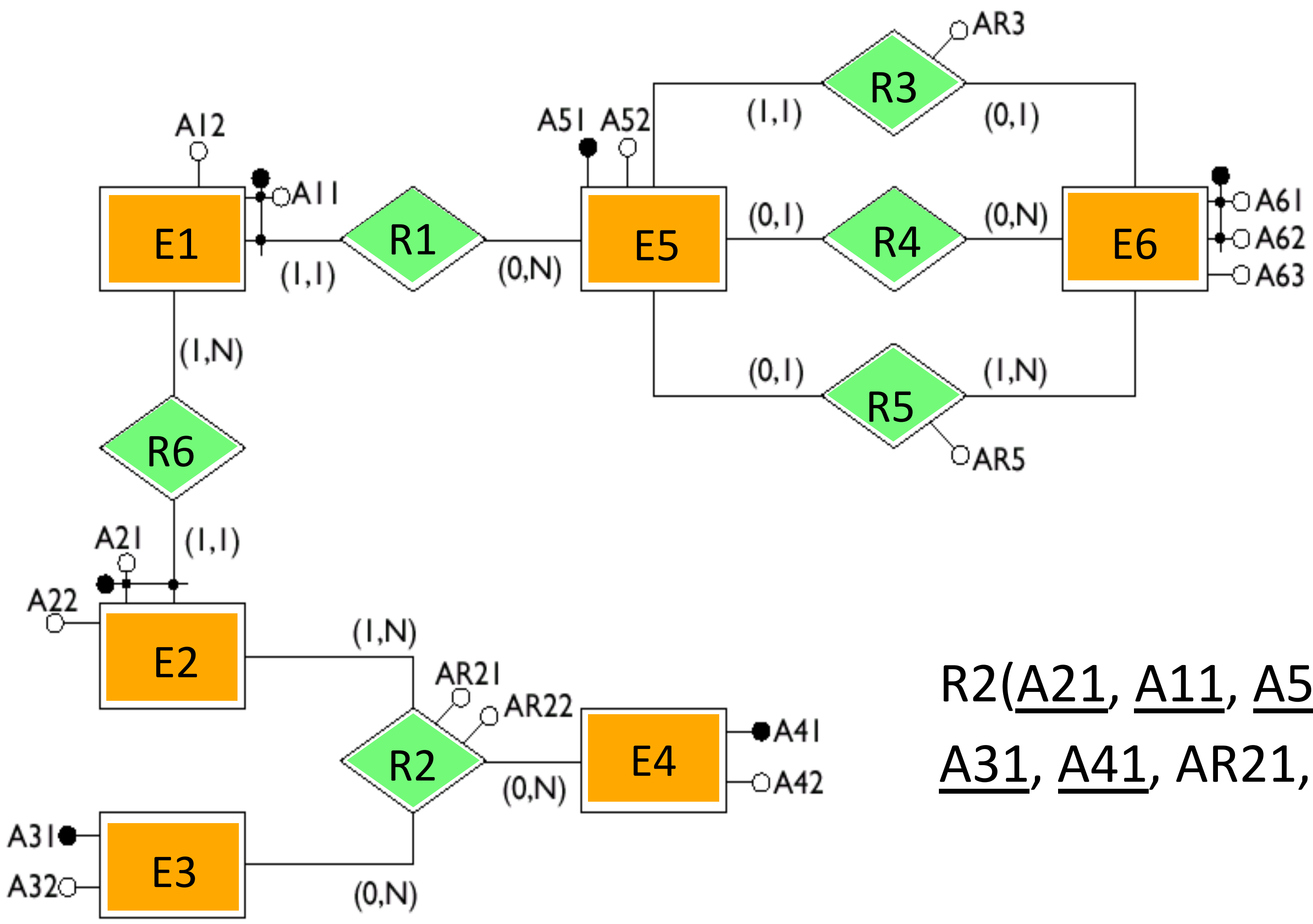
Weak Entities



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

E2(<u>A21</u>, <u>A11</u>, <u>A51</u>, 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($\underline{\text{A11}}$, $\underline{\text{A51}}$, A12)

E2($\underline{\text{A21}}$, $\underline{\text{A11}}$, $\underline{\text{A51}}$, A22)

E3($\underline{\text{A31}}$, A32)

E4($\underline{\text{A41}}$,A42)

E5($\underline{\text{A51}}$, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

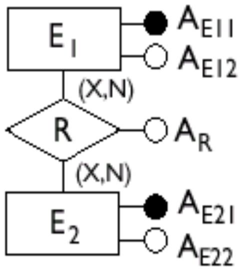
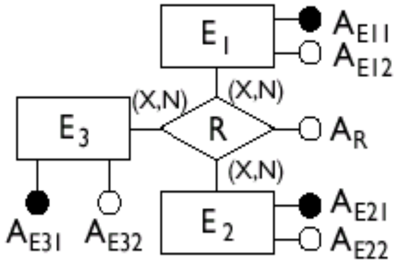
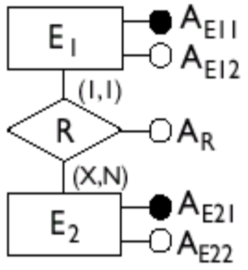
E6($\underline{\text{A61}}$, $\underline{\text{A62}}$, A63)

R2($\underline{\text{A21}}$, $\underline{\text{A11}}$, $\underline{\text{A51}}$, $\underline{\text{A31}}$, $\underline{\text{A41}}$, AR21, AR22)

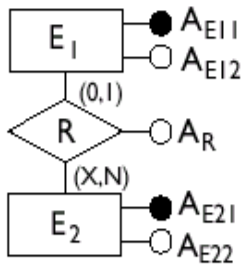
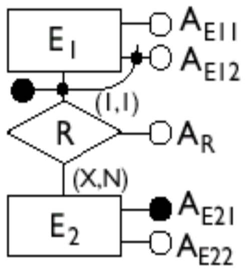
We have a Database Schema!

Ready to create our tables in the DBMS!

Summary of Transformation Rules

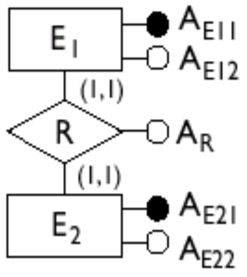
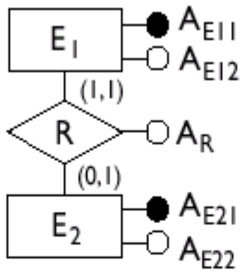
Type	Initial schema	Possible translation
Binary many-to-many relationship	 E1 A E11 A E12 (X,N) R A R (X,N) E2 A E21 A E22	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Ternary many-to-many relationship	 E1 A E11 A E12 (X,N) (X,N) E3 R A R (X,N) A E31 A E32 E2 A E21 A E22	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$
One-to-many relationship with mandatory participation	 E1 A E11 A E12 (1,1) R A R (X,N) E2 A E21 A E22	$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

...More Rules...

Type	Initial schema	Possible translation
One-to-many relationship with optional participation	 E1 E2 R (0,1) (X,N) A E11 A E12 A R A E21 A E22	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E21}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Relationship with external identifiers	 E1 E2 R (1,1) (X,N) A E11 A E12 A R A E21 A E22	$E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

_ : Primary keys
...: Alternative primary keys
*: NULL values are allowed

…Even More Rules…

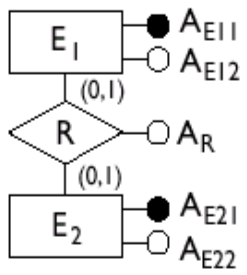
Type	Initial schema	Possible translation
One-to-one relationship with mandatory participation for both entities	 E1 E2 R (1,1) (1,1) AE11 AE12 AR AE21 AE22	$E_1(\underline{A_{E11}}, A_{E12}, \underset{- - -}{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_2(\underline{A_{E21}}, A_{E22}, \underset{- - -}{A_{E11}}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$
One-to-one relationship with optional participation for one entity	 E1 E2 R (1,1) (0,1) AE11 AE12 AR AE21 AE22	$E_1(\underline{A_{E11}}, A_{E12}, \underset{- - -}{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

_ : Primary keys

…: Alternative primary keys

*: NULL values are allowed

…and the Last One…

Type	Initial schema	Possible translation
One-to-one relationship with optional participation for both entities	 E_1 A_{E11} A_{E12} (0,1) R A_R (0,1) E_2 A_{E21} A_{E22}	$E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A^*_{E11}, A^*_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$