XV. The Entity-Relationship Model

The Entity-Relationship Model

Entities, Relationships and Attributes

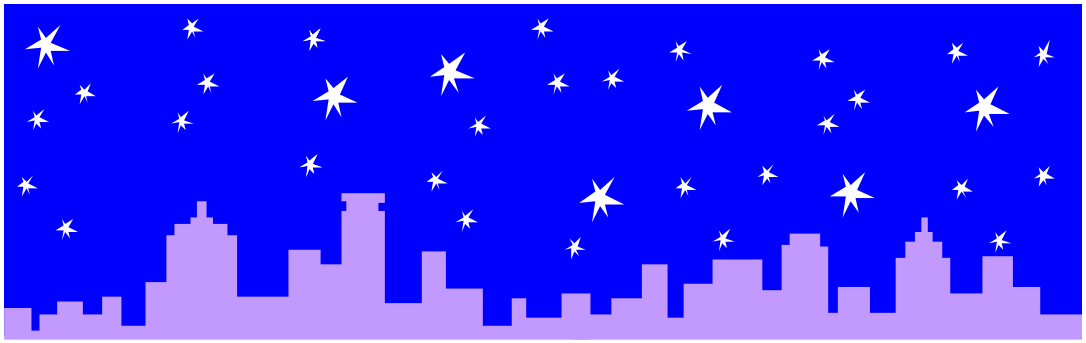
Cardinalities, Identifiers and Generalization

Documentation of E-R Diagrams and Business Rules

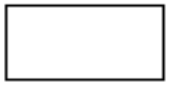
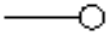
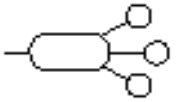
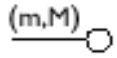
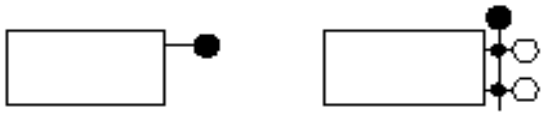
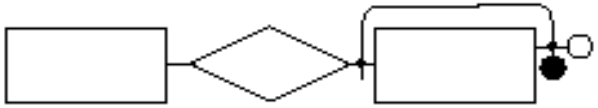
Acknowledgment: these slides are based on Prof. John Mylopoulos slides which are used to teach a similar course in the University of Toronto – St. George campus. Used with Permission.

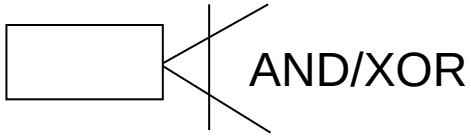
The Entity Relationship Model

- The ***Entity-Relationship*** (ER) ***model*** is a *conceptual* data model, capable of describing the data requirements for a new information system in a direct and easy to understand graphical notation.
- Data requirements are described in terms of a ***conceptual*** *(or,* ***ER****)* ***schema****.*
- ER schemata are comparable to class diagrams.



The Constructs of the E-R Model

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



Entities

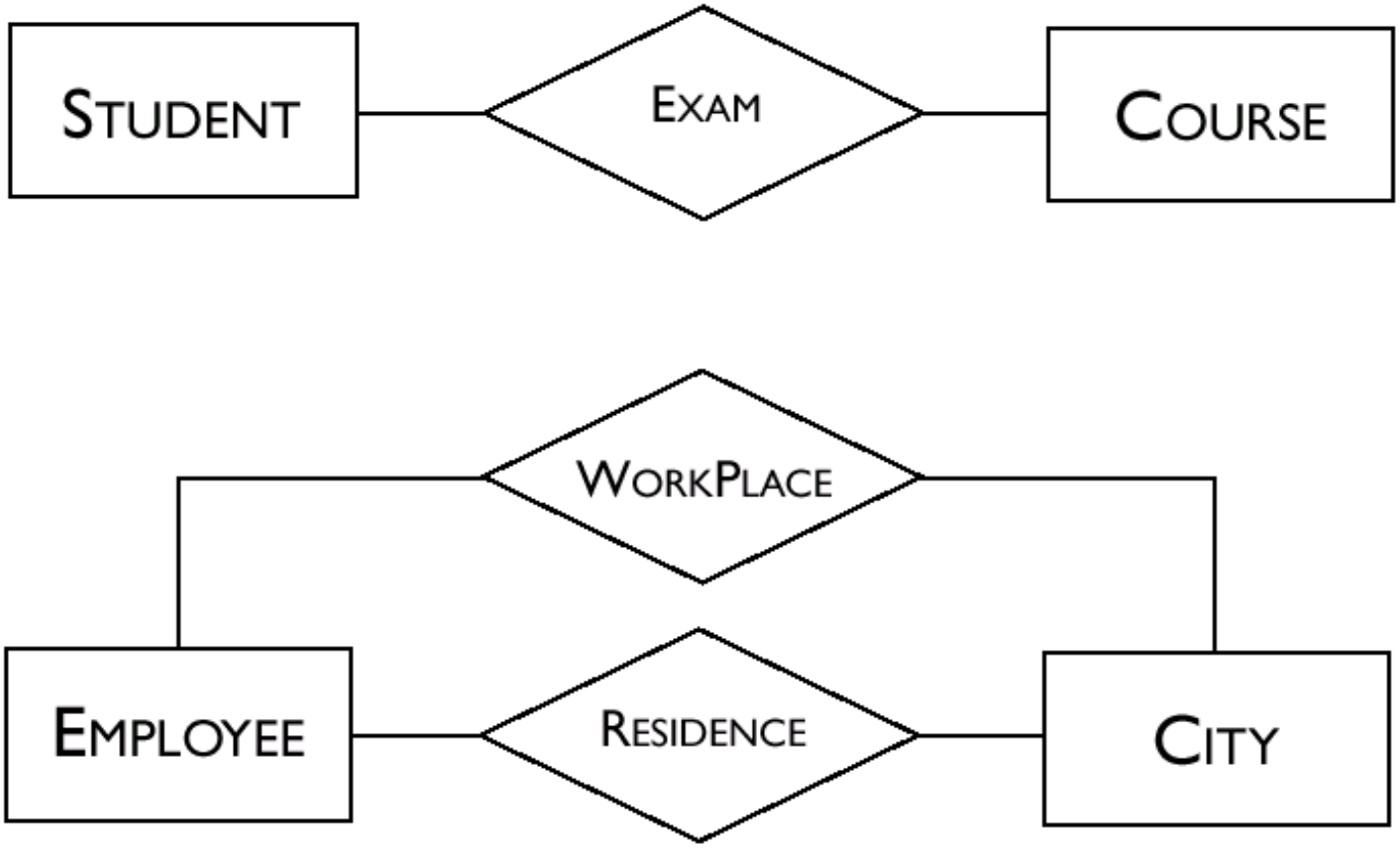
- These represent classes of objects (facts, things, people,...) that have properties in common and an autonomous existence.
- City, Department, Employee, Purchase and Sale are examples of entities for a commercial organization.
- An instance of an entity is an object in the class represented by the entity.
- Stockholm, Helsinki, are examples of instances of the entity City, and the employees Peterson and Johanson are examples of instances of the Employee entity.
- The E-R model is very different from the relational model in which it is not possible to represent an object without knowing its properties (an employee is represented by a tuple containing the name, surname, age, and other attributes.)

Examples of Entities

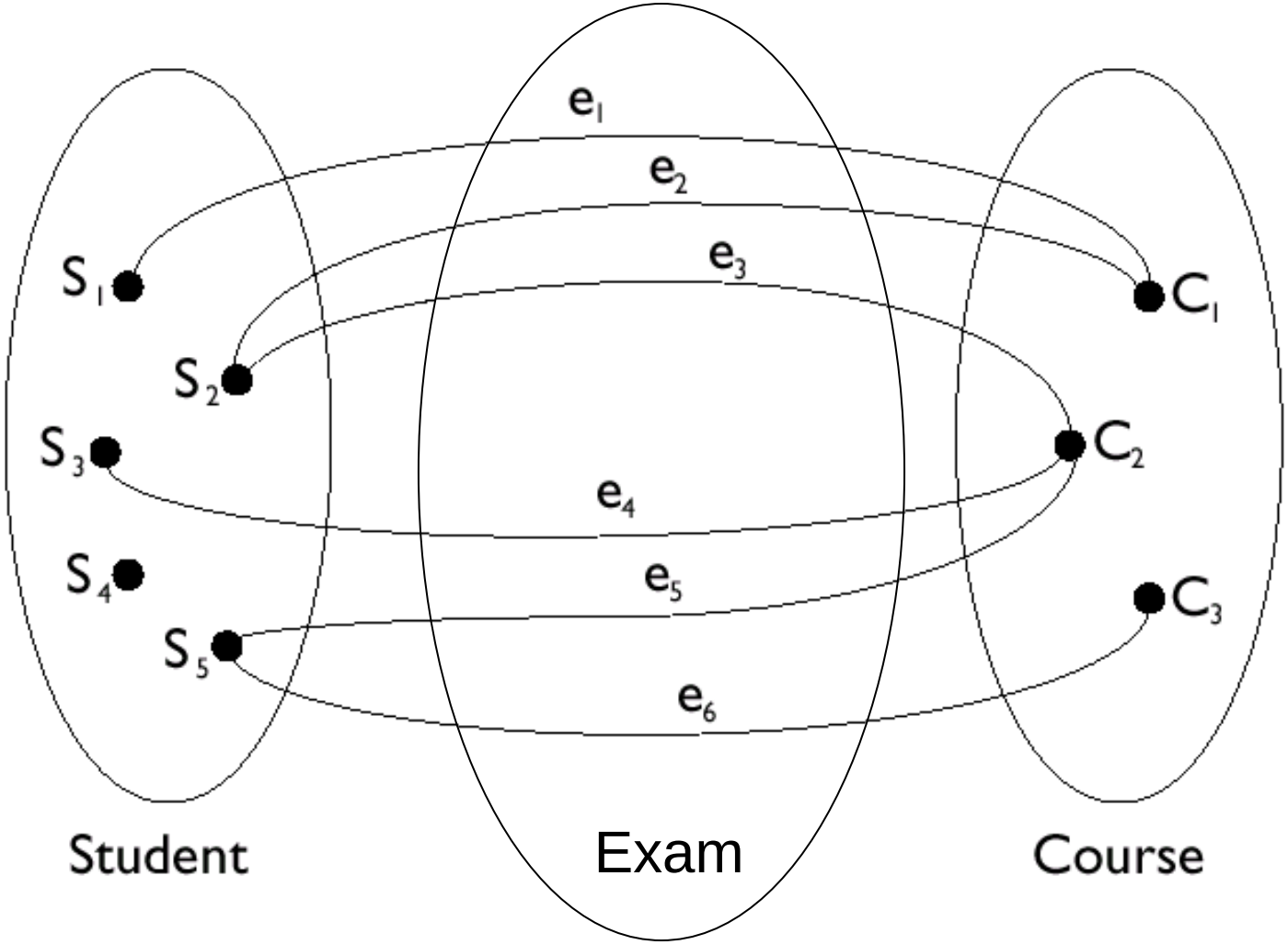
Relationships

- They represent logical links between two or more entities.
- Residence is an example of a relationship that can exist between the entities City and Employee; Exam is an example of a relationship that can exist between the entities Student and Course.
- An instance of a relationship is an n-tuple made up of instances of entities, one for each of the entities involved.
- The pair (Johanssen,Stockholm), or the pair (Peterson,Oslo), are examples of instances of the relationship Residence.

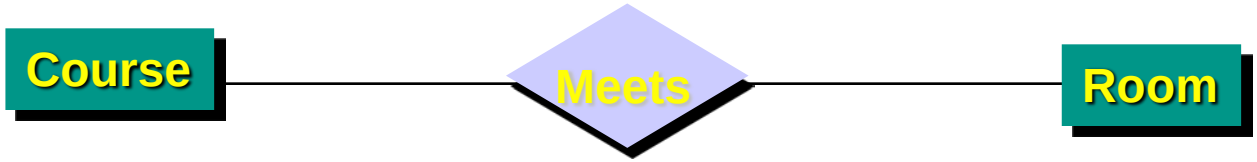
Examples of Relationships



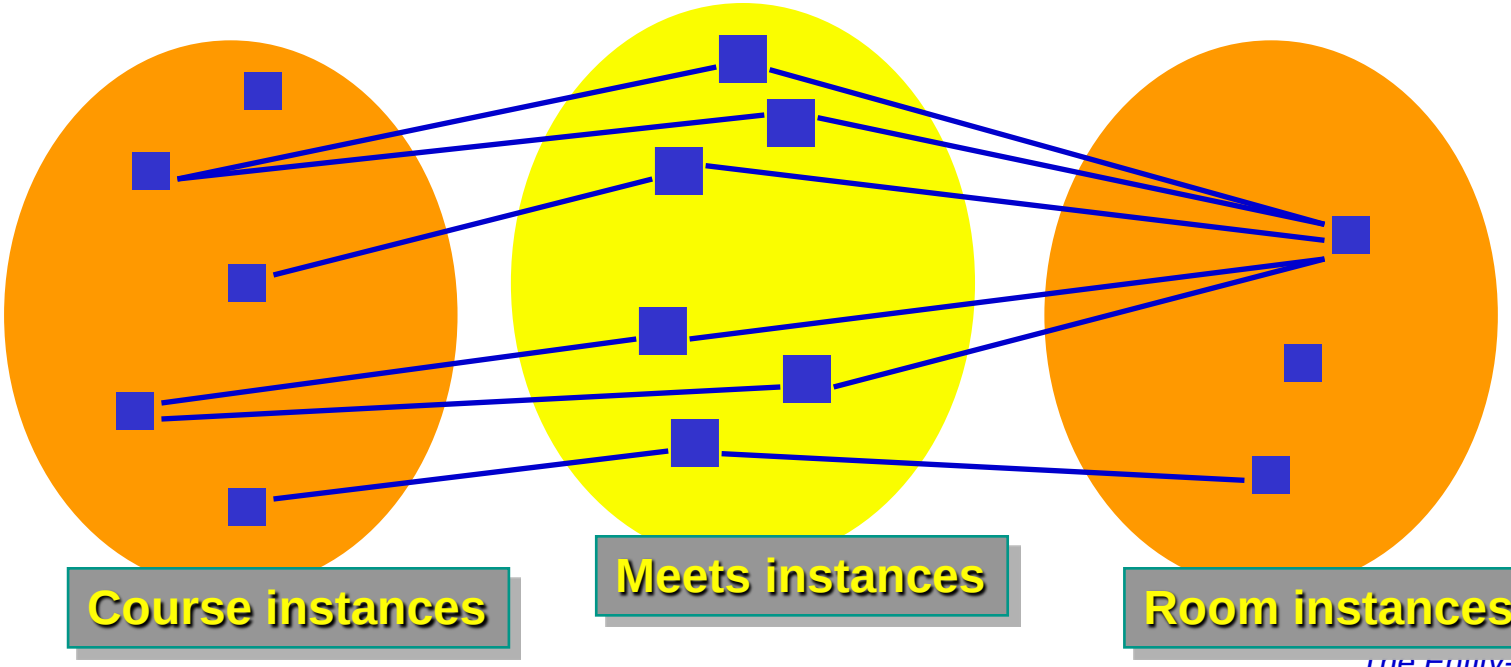
Example of Instances for Exam



What Does An E-R Diagram Really Mean?

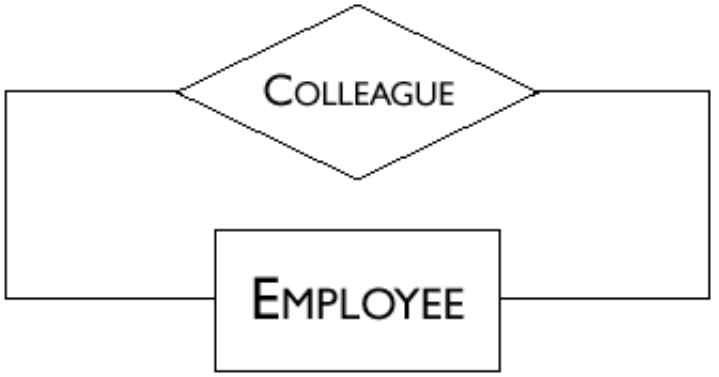


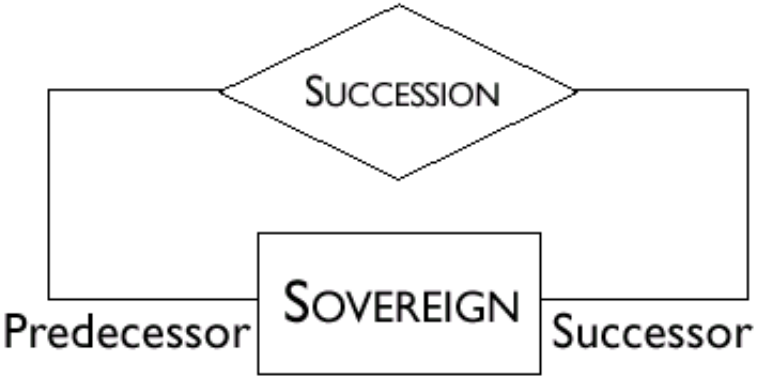
- Course and Room are entities. Their instances describe particular courses (e.g., CSC340S) and particular rooms (e.g., WB116).
- Meets is a relationship. Its instances describe particular meetings. Each meeting has exactly one associated course and room



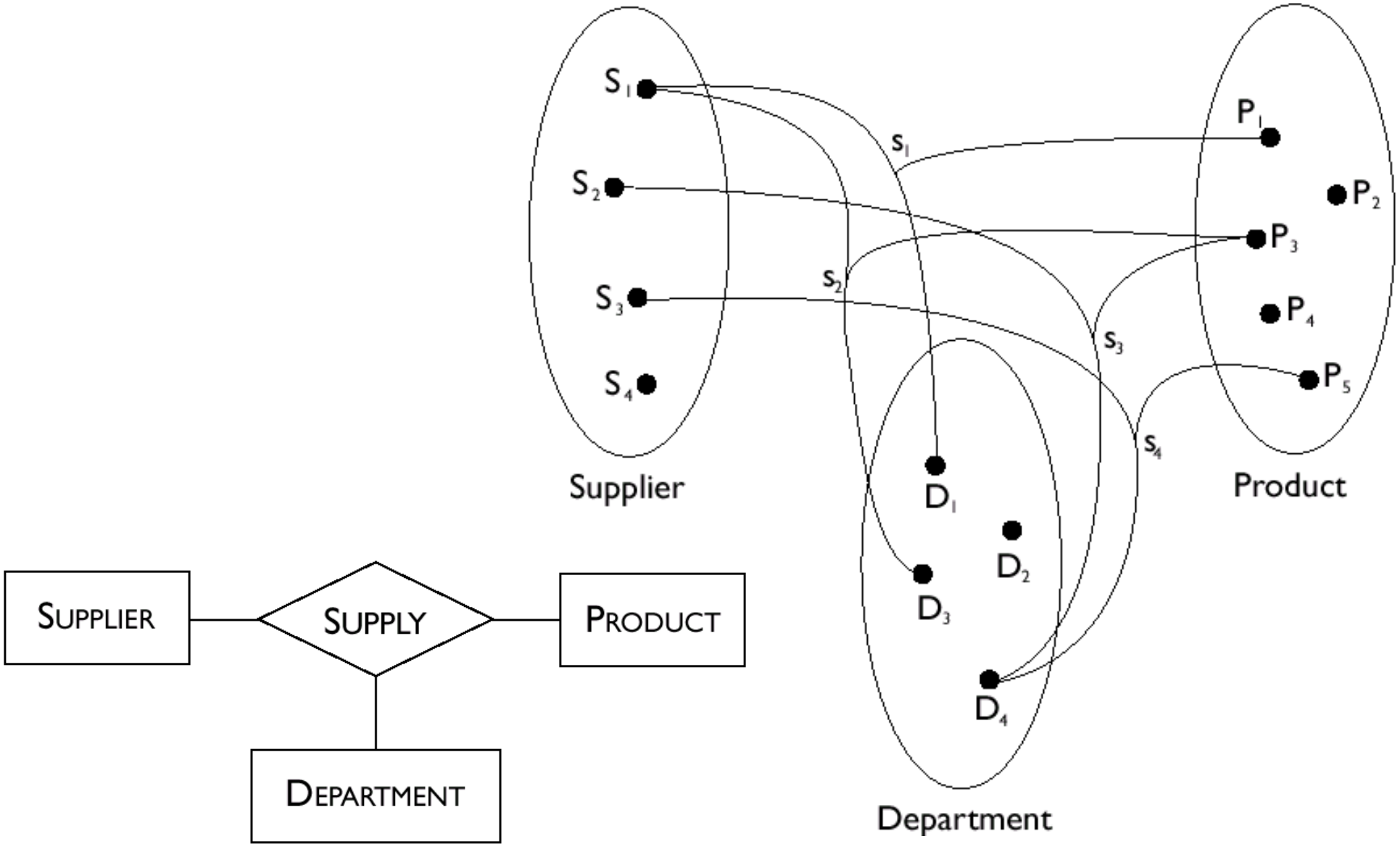
Recursive Relationships

- Recursive relationships are also possible, that is relationships between an entity and itself.
- Note in the second example that the relationship is not symmetric. In this case it is necessary to indicate the two roles that the entity involved plays in the relationship.

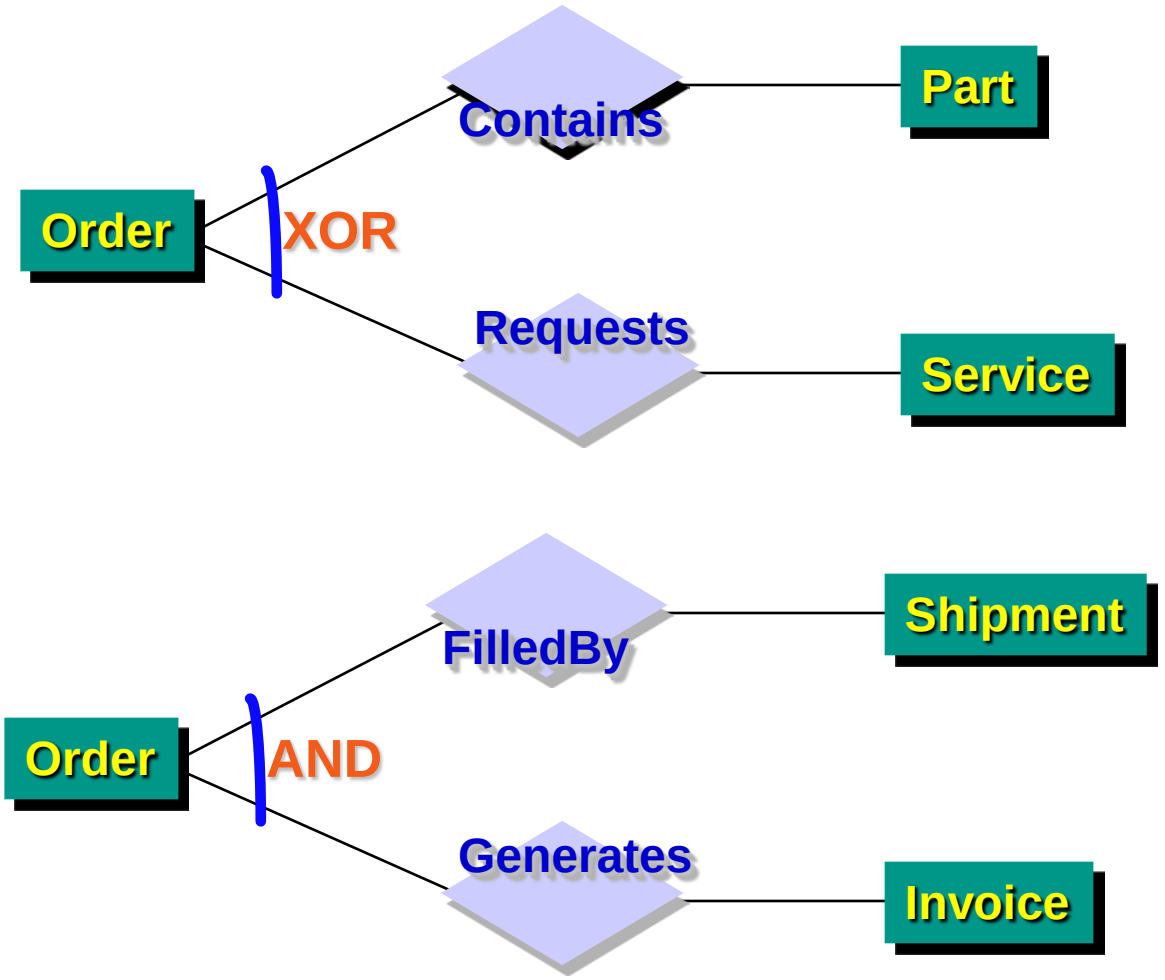




Ternary Relationships

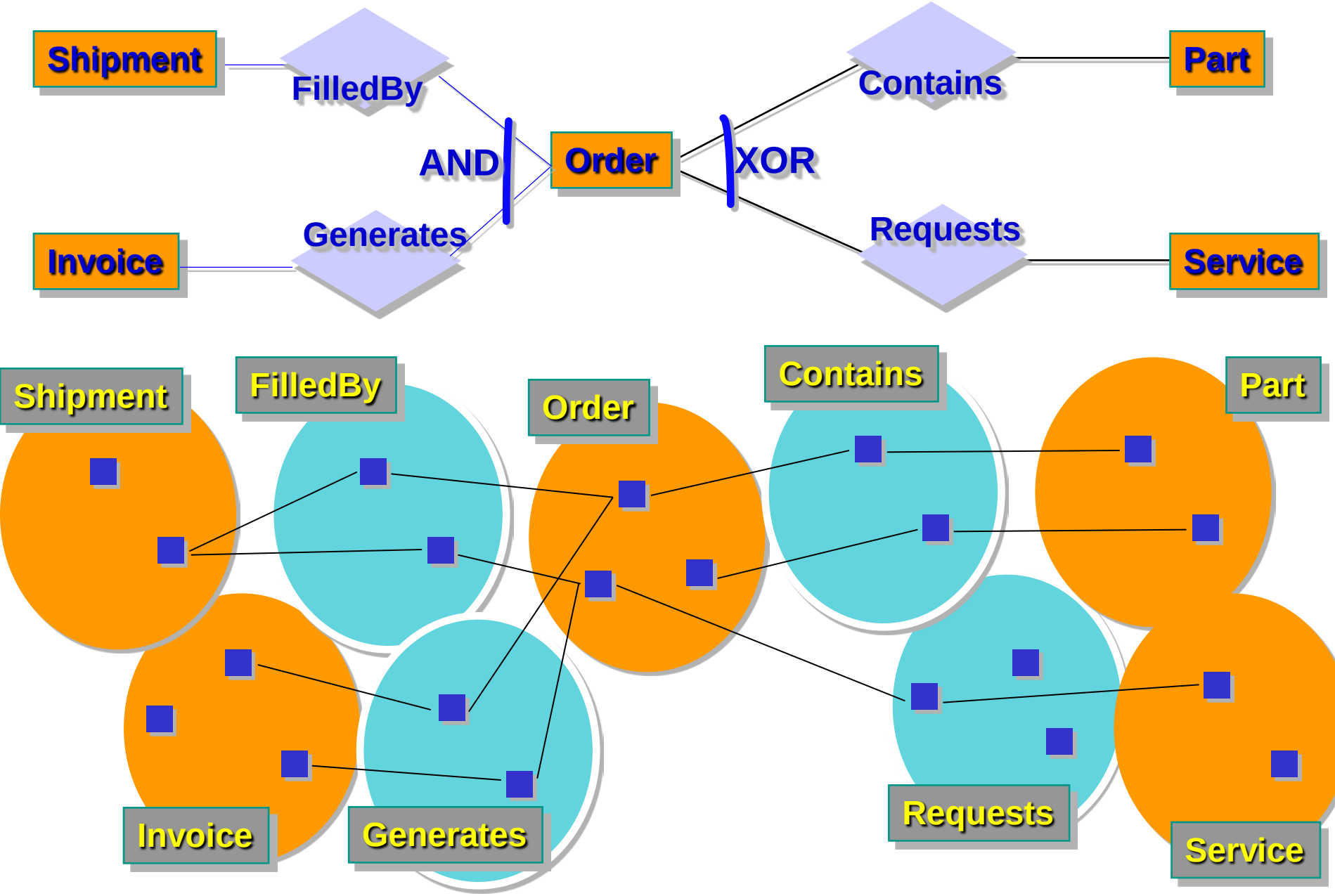


AND/XOR for Relationships



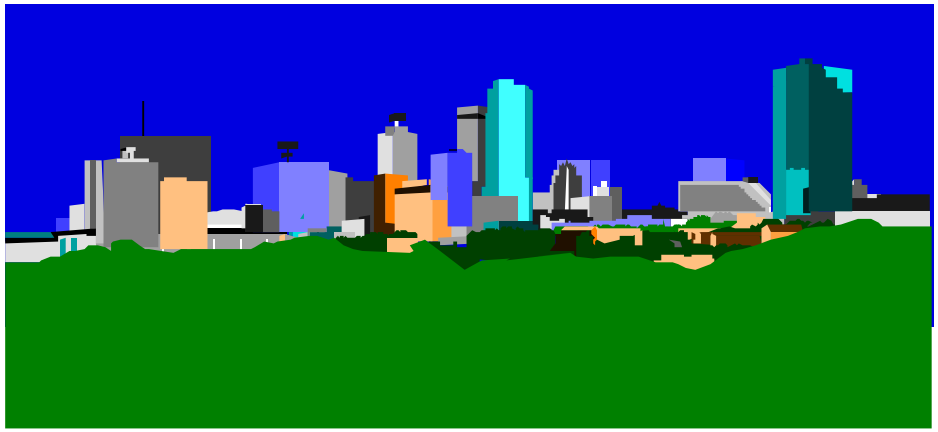
"Orders either order a part or request a service, but not both"

"For any given order, whenever there is at least one invoice there is also at least one shipment and vice versa"

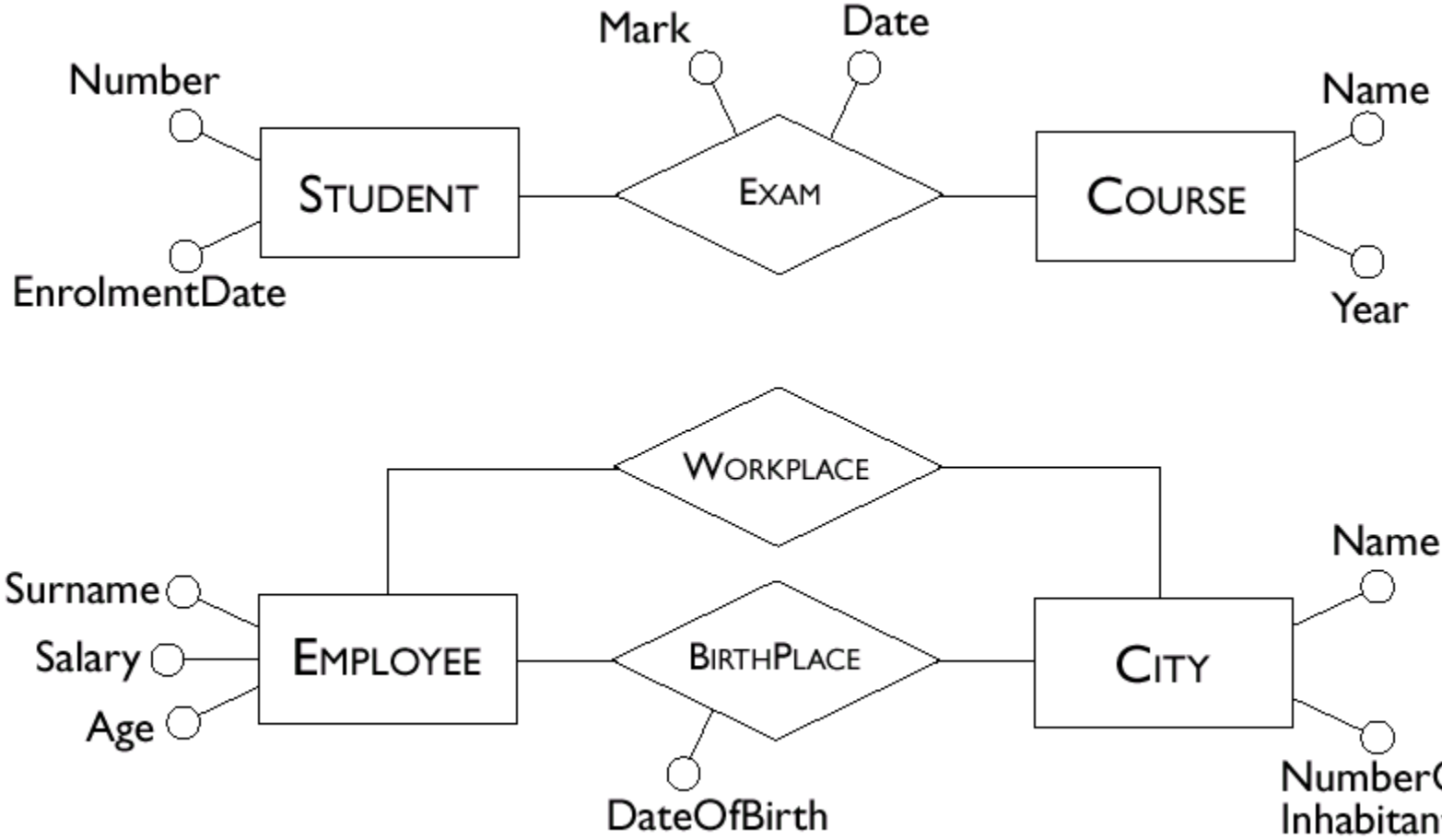

Shipment
FilledBy
AND
Order
XOR
Contains
Part
Invoice
Generates
Requests
Service
Shipment
FilledBy
Order
Contains
Part
Invoice
Generates
Requests
Service

Attributes

- These describe the elementary properties of entities or relationships.
- For example, Surname, Salary and Age are possible attributes of the Employee entity, while Date and Mark are possible attributes for the relationship Exam between Student and Course.
- An attribute associates with each instance of an entity (or relationship) a value belonging to a set known as the ***domain*** of the attribute.
- The domain contains the admissible values for the attribute.

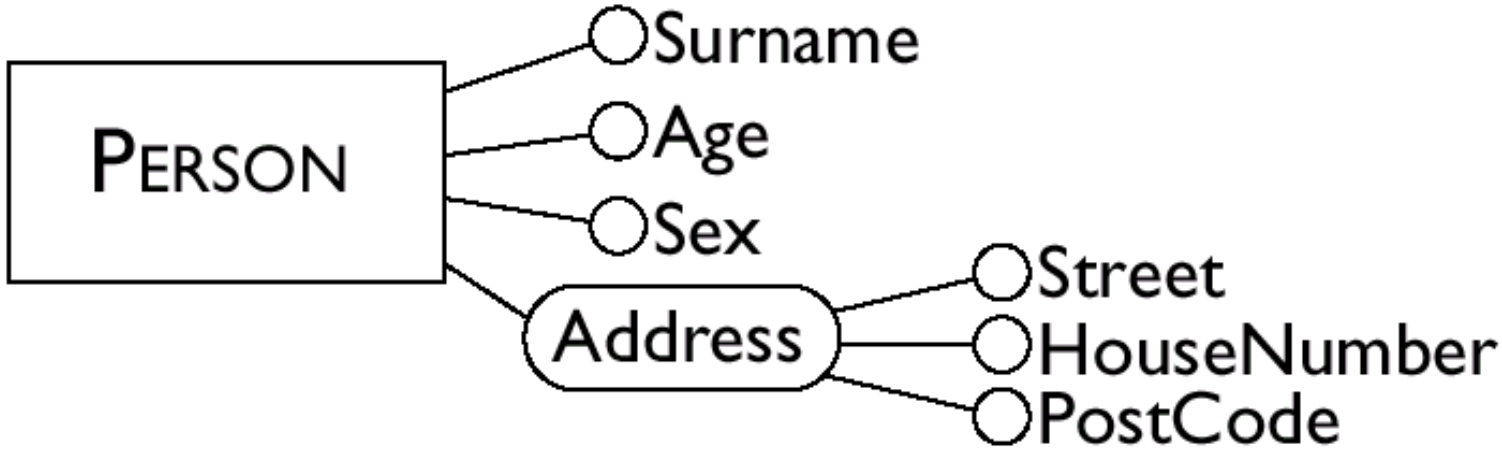


Attribute Examples

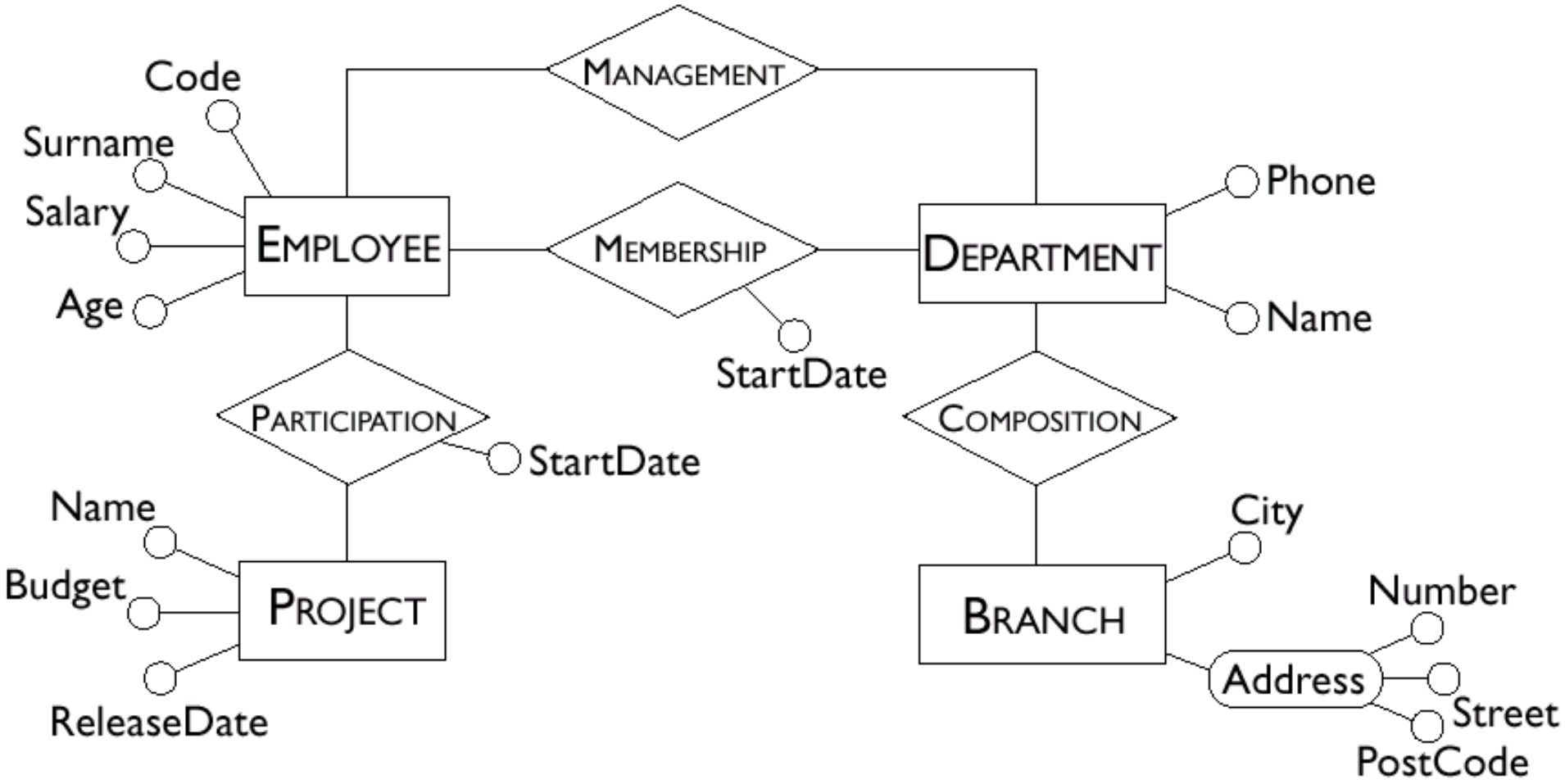


Composite Attributes

- It is sometimes convenient to group attributes of the same entity or relationship that have closely connected meanings or uses. Such groupings are called ***composite attributes***.



Schema with Attributes



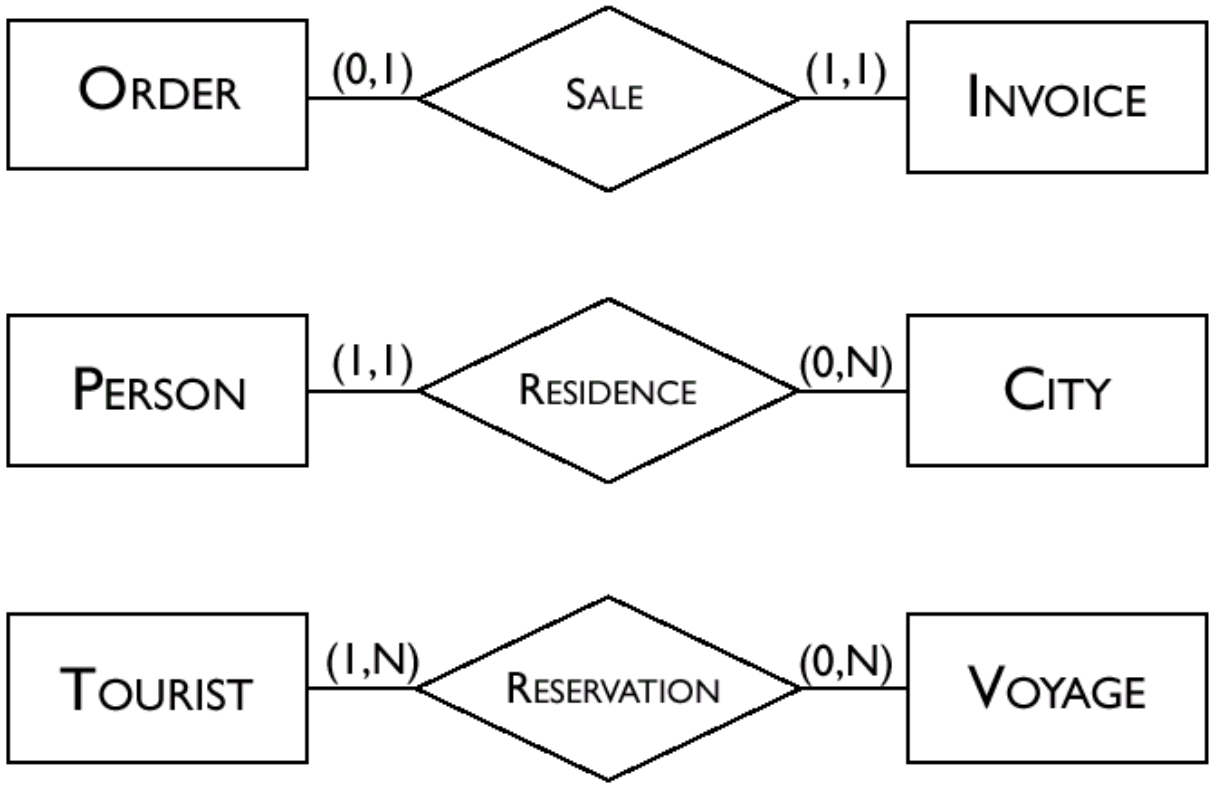
Cardinalities

- These are specified for each entity participating in a relationship and describe the maximum and minimum number of relationship occurrences in which an entity occurrence can participate.
- Cardinalities state how many times can an entity instance participate in instances of a given relationship.

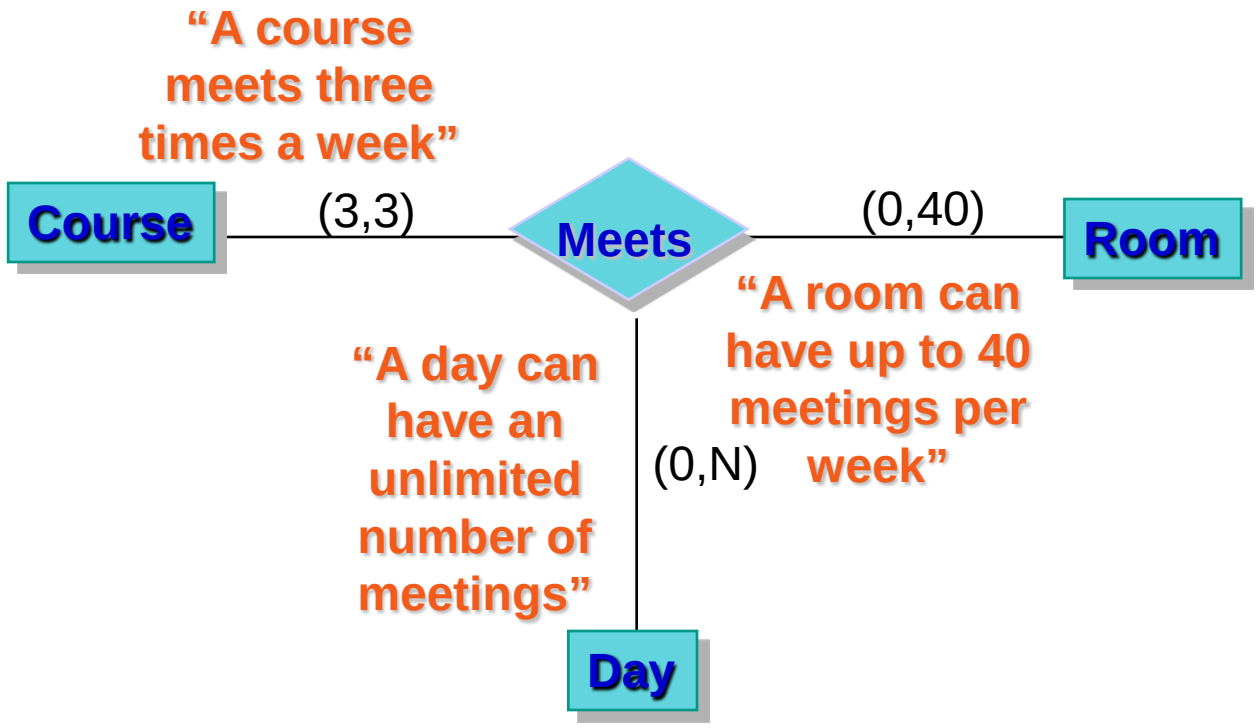
Cardinalities (cont'd)

- In principle, a cardinality is any pair of non-negative integers (n,m) such that n≤m. or a pair of the form (n,N) where N means "any number".
- If minimum cardinality is 0, we say that entity participation in a relationship is optional. If minimum cardinality is 1, we say that entity participation in a relationship is mandatory.
- If maximum cardinality is 1, each instance of the entity is associated at most with a single instance of the relationship; if maximum cardinality is N, then each instance of the entity is associated with an arbitrary number of instances of the relationship.

Cardinality Examples

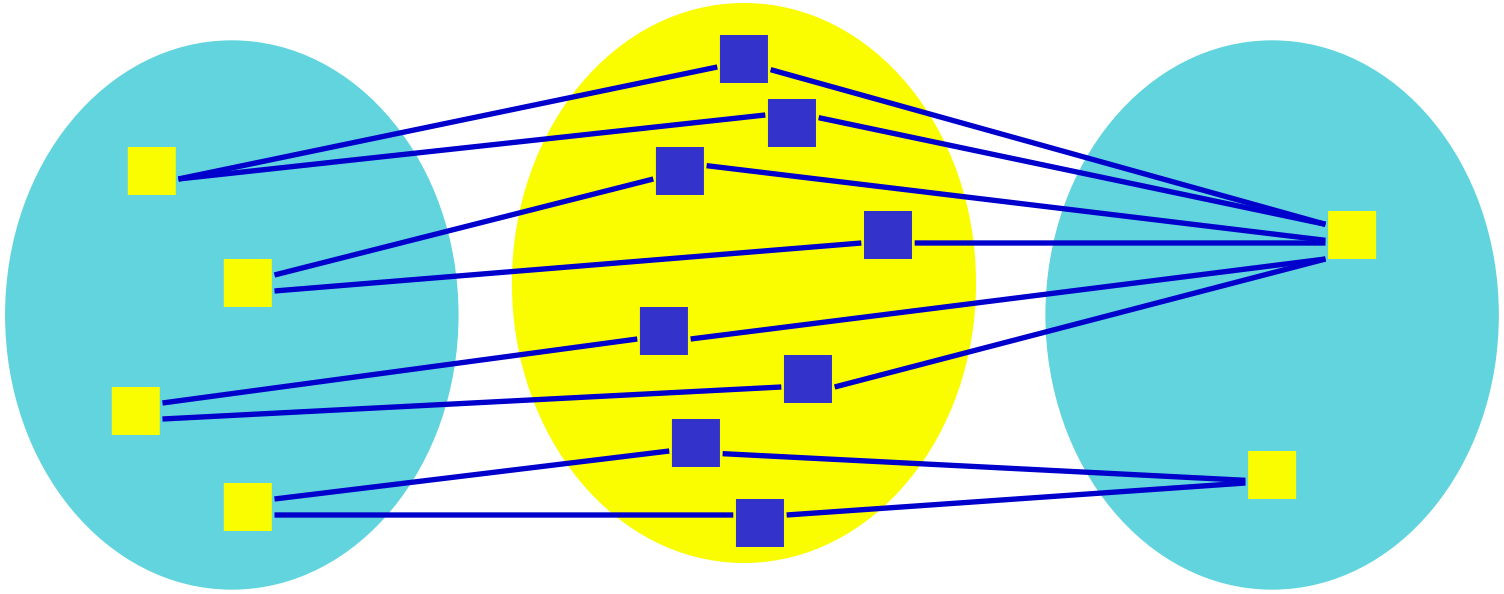


Cardinality Example

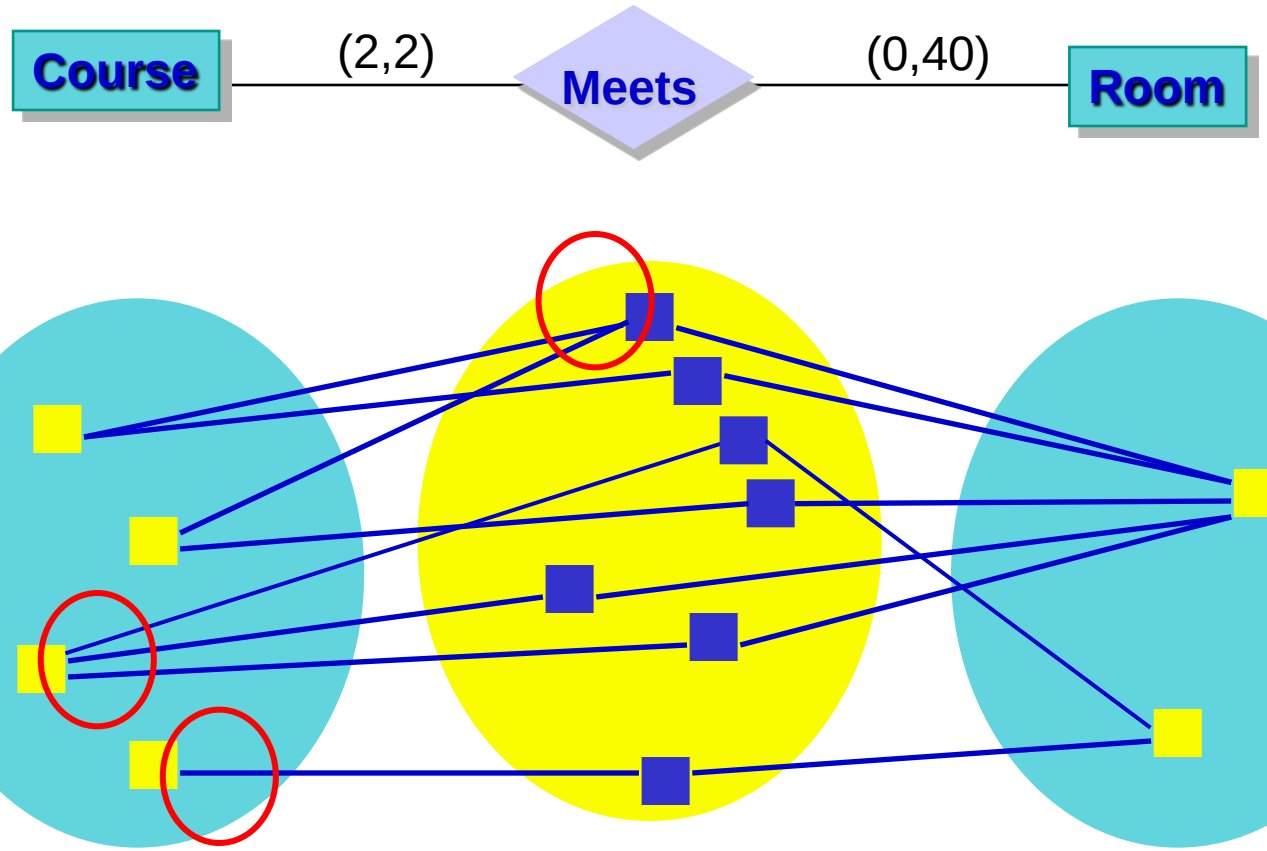


Instantiating ER Diagrams

- An ER diagram specifies what states are possible in the world being modeled

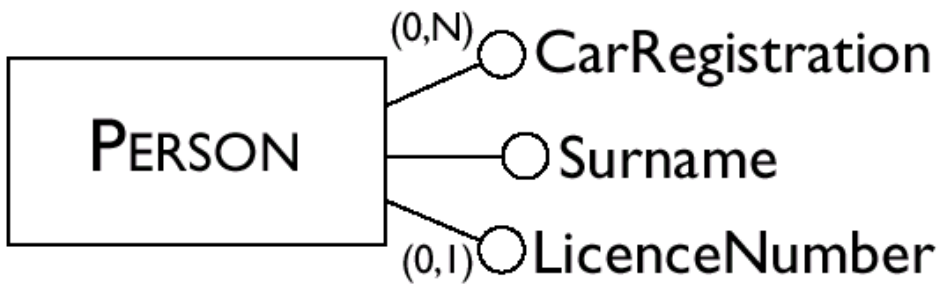


Illegal Instantiations



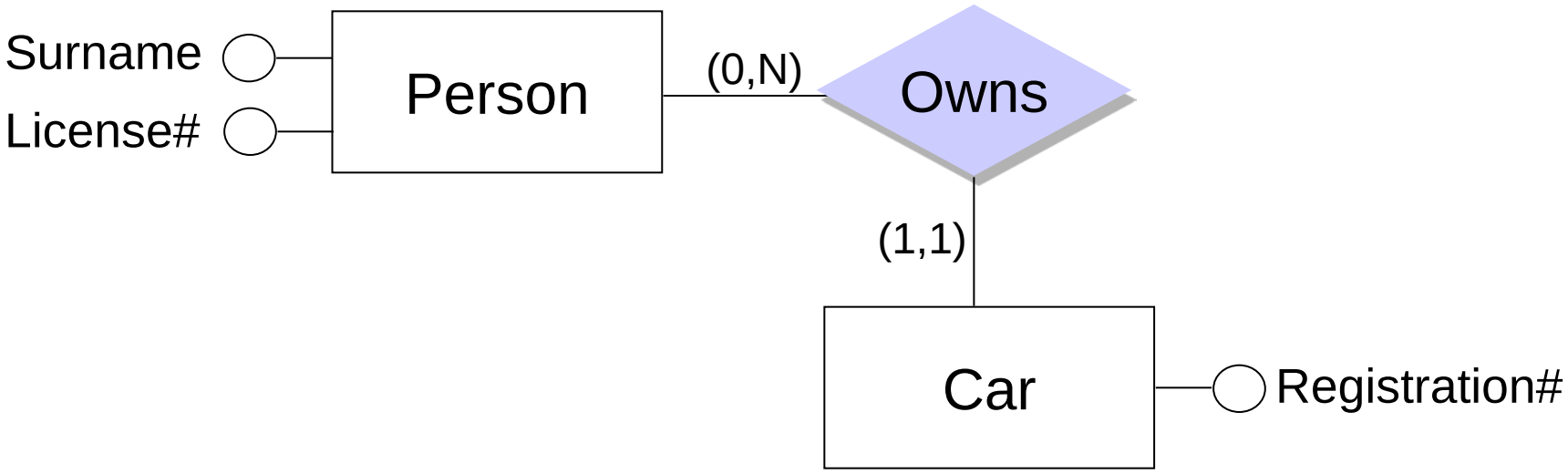
Cardinalities of Attributes

- They are specified for the attributes of entities (or relationships) and describe the minimum and maximum number of values of the attribute associated with instances of an entity or a relationship.
- In most cases, the cardinality of an attribute is equal to (1,1) and is omitted (***single-valued*** attributes)
- The value of a certain attribute however, may also be null, or there may exist several values of a certain attribute for an entity instance (***multi-valued*** attributes)



Cardinalities (cont'd)

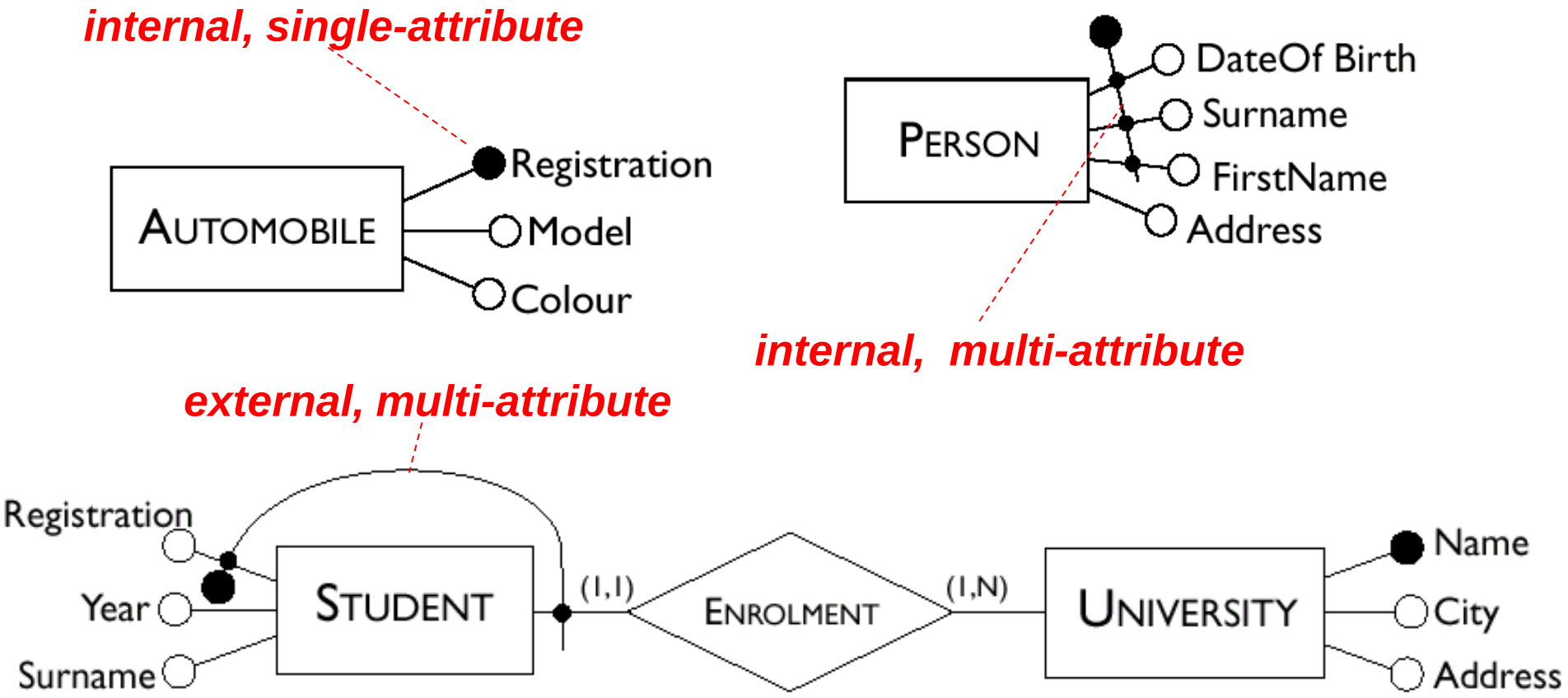
- Multi-valued attributes should be used with great caution, because they often represent situations that can be modelled in many cases with additional entities linked by one-to-many (or many-to-many) relationships to the entity to which they refer.



Identifiers

- ***Identifiers*** (or ***keys***) consist of one or more attributes which identify uniquely instances of an entity.
- For example, the entity Person may be identified by the attribute socialInsurance#. Alternatively, it may be identified by the three attributes firstName, middleName, lastName, address.
- In many cases, an identifier is formed by one or more attributes of the entity itself: in this case we talk about an ***internal*** identifier.
- Sometimes, however, the attributes of an entity are not sufficient to identify its instances unambiguously and other entities are involved in the identification. Identifiers of this type are called ***external*** identifiers.
- An identifier for a relationship consists of identifiers for all the entities it relates. For example, the identifier for the relationship (Person-) Owns(-Car) is a combination of the Person and Car identifiers.

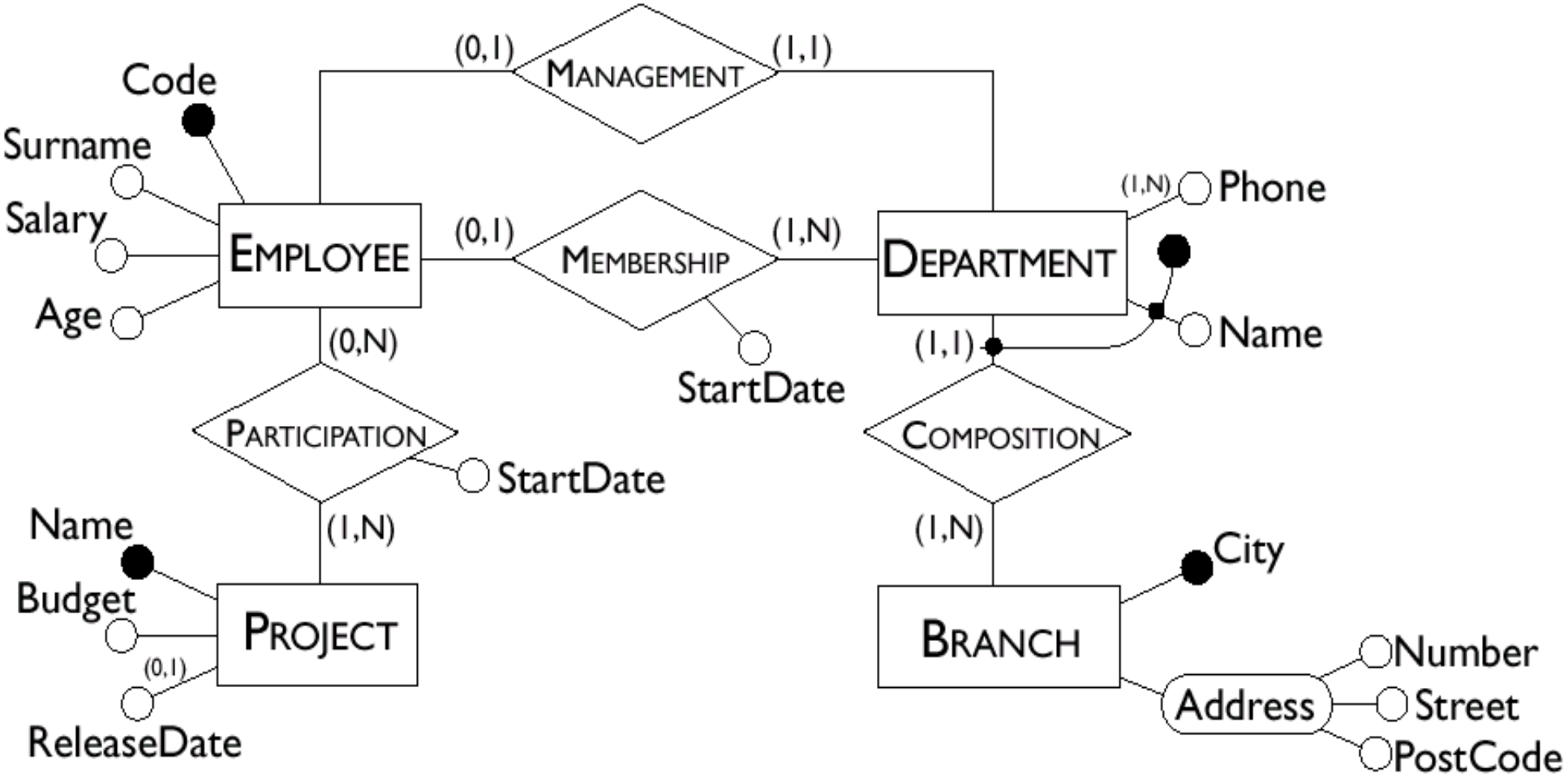
Examples of Identifiers



General Observations on Identifiers

- An identifier can involve one or more attributes, provided that each of them has (1,1) cardinality.
- An external identifier can involve one or more entities, provided that each of them is member of a relationship to which the entity to identify participates with cardinality equal to (1,1).
- An external identifier can involve an entity that is in its turn identified externally, as long as cycles are not generated.
- Each entity must have one (internal or external) identifier, but can have more than one. Actually, if there is more than one identifier, then the attributes and entities involved in an identification can be optional (minimum cardinality equal to 0).

Schema with Identifiers

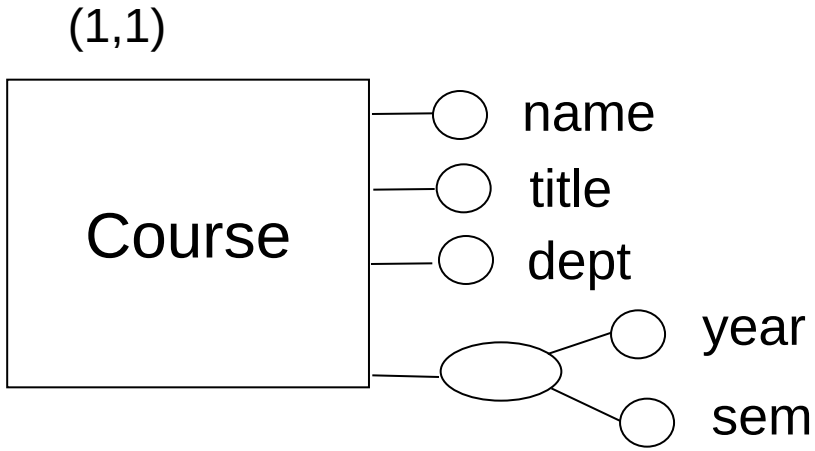


Modeling an Application with Identifiers

Identifiers constitute a powerful mechanism for modeling an application. Assume we want a database storing information about lecture meetings.

- Suppose first that we use the identifier coursename,day,hour for the Meeting entity. This says, that there can only be one meeting at any one time for a given course name, day, hour; in other words, we can't have two sections of the same course meeting at the same day+hour.
- Suppose now we use only coursename as identifier for Meeting. This says that there can only be one meeting per given course name (unreasonable!)
- If we use courseinstructor,room as identifier for Meeting, we are stating that there can only be one meeting for a given instructor+room combination, so an instructor must have all her meetings in different rooms!
- Finally, if the attribute courseinstructor by itself forms an identifier for Meeting, then the diagram we have built is stating that each instructor participates in at most one meeting, again this is unreasonable.

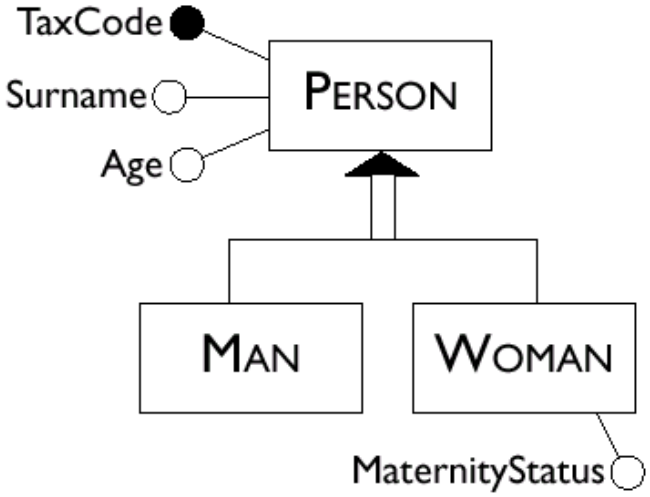
Consider...

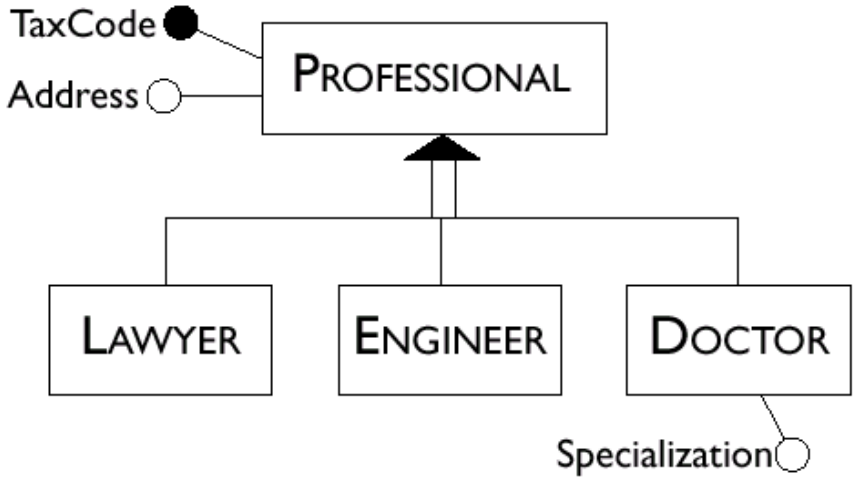


- Assume that we want to keep track of course offerings over the years.
- What does each of the following combinations of attributes say about the application?
  - ✓ name;
  - ✓ name, sem, year;
  - ✓ sem, year;
  - ✓ name, dept;
  - ✓ dept, year.

Generalizations

- These represent logical links between an entity E, known as ***parent*** entity, and one or more entities $E_1,...,E_n$ called ***child*** entities, of which E is more general, in the sense that they are a particular case.
- In this situation we say that E is a ***generalization*** of $E_1,...,E_n$ and that the entities $E_1,...,E_n$ are ***specializations*** of E.





Properties of Generalization

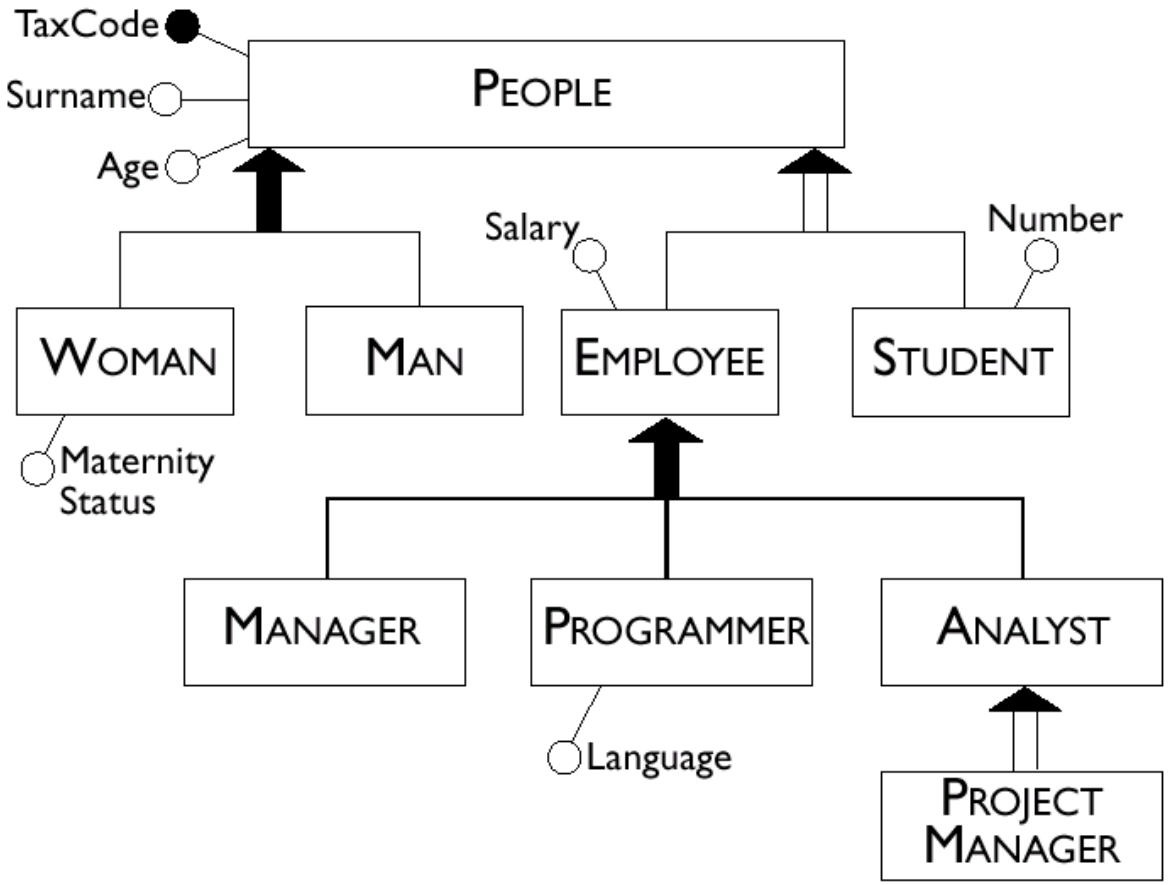
- Every instance of a child entity is also an instance of the parent entity.
- Every property of the parent entity (attribute, identifier, relationship or other generalization) is also a property of a child entity. This property of generalizations is known as ***inheritance***.

Types of Generalizations

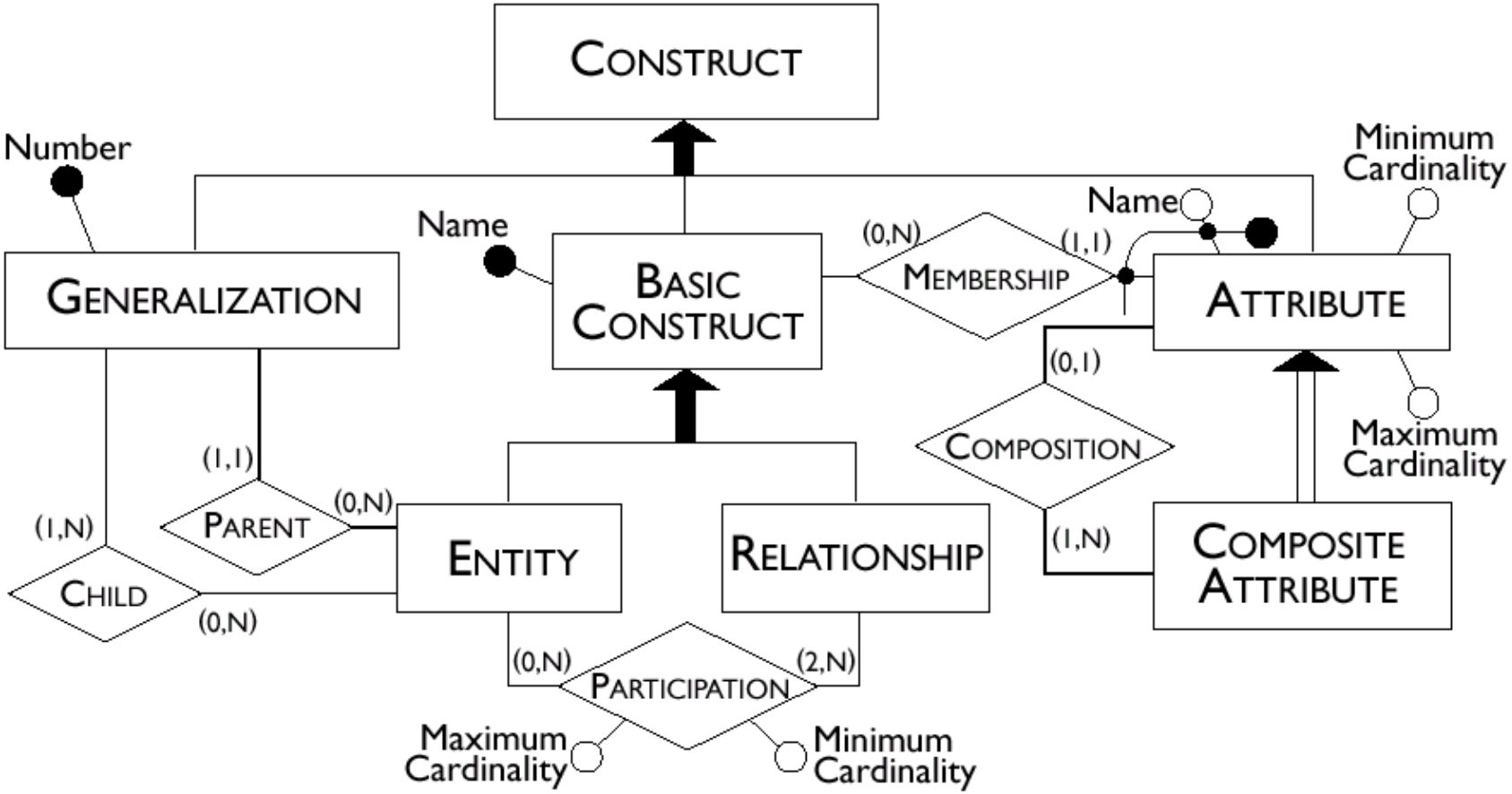
- A generalization is ***total*** if every instance of the parent entity is also an instance of one of its children, otherwise it is ***partial***.
- A generalization is ***exclusive*** if every instance of the parent entity is at most an instance of one of the children, otherwise it is ***overlapping***.
- The generalization Person, of Man and Woman is total (the sets of men and the women constitute 'all' the people) and exclusive (a person is either a man or a woman).
- The generalization Vehicle of Automobile and Bicycle is partial and exclusive, because there are other types of vehicle (for example, motor bike) that are neither cars nor bicycle.
- The generalization Person of Student and Employee is partial and overlapping, because there are students who are also employed.

Generalization Hierarchies

- Total generalization is represented by a solid arrow.
- In most applications, modeling the domain involves a *hierarchy* of generalizations that includes several levels



The E-R Model, as an E-R Diagram



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 5,000), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for 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 self-employed, we need to know her area of expertise, and, if appropriate, 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 freelance.

Example, Annotated

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 5,000), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for 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 self-employed, we need to know her area of expertise, and, if appropriate, 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 freelance.

More Annotations

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 5,000), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for 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 self-employed, we need to know her area of expertise, and, if appropriate, 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 freelance.

Glossary Example

Term	Description	Synonym	Links
Trainee	Participant in a course. Can be an employee or self-employed.	Participant	Course, Company
Instructor	Course tutor. Can be freelance.	Tutor	Course
Course	Course offered. Can have various editions.	Seminar	Instructor, Trainee
Company	Company by which a trainee is employed or has been employed.		Trainee

More Annotations

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 5,000), identified by a code, we want to store her social security number, surname, age, sex, place of birth, employer's name, address and telephone number, previous employers (and periods employed), the courses attended (there are about 200 courses) and the final assessment for 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 self-employed, we need to know her area of expertise, and, if appropriate, 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 freelance.

Structuring of Requirements (I)

Phrases of a general nature
We wish to create a database for a company that runs training courses. We wish to maintain data for the trainees and the instructors.
Phrases relating to trainees
For each trainee (about 5000), identified by a code, the database 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 trainees
For each employer of a trainee the database will store name, address and telephone number.

Structuring of Requirements (II)

Phrases relating to 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 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.

Operational Requirements

- **operation 1**: insert a new trainee including all 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 for 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).

Conceptual Design with the ER Model

Design choices:

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- Identifying relationships: Binary or ternary? Aggregation?

Constraints on the ER Model:

- A lot of data semantics can (and should) be captured.
- But some constraints cannot be captured by ER diagrams.

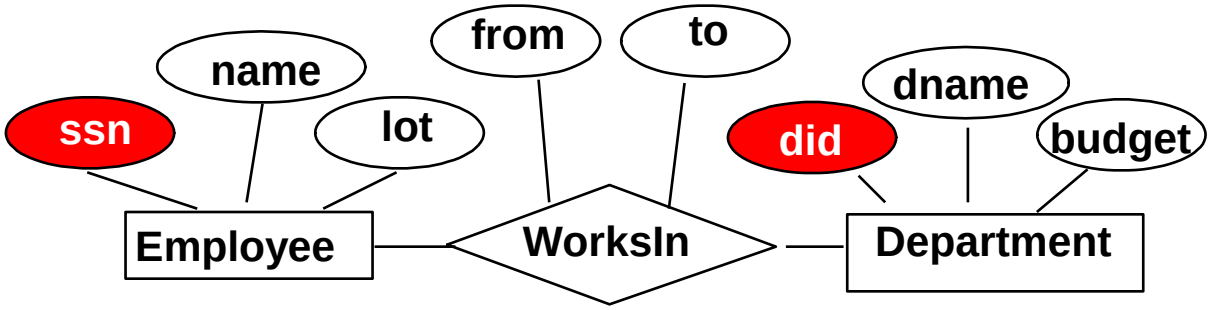
Some Rules of Thumb

- If a concept has significant properties and/or describes classes of objects with an autonomous existence, it is appropriate to represent it as an entity.
- If a concept has a simple structure, and has no relevant properties associated with it, it is convenient to represent it with an attribute of another concept to which it refers.
- If a concept provides a logical link between two (or more) entities, it is convenient to represent it with a relationship.
- If one or more concepts are particular cases of another concept, it is convenient to represent them in terms of a generalization relationship.

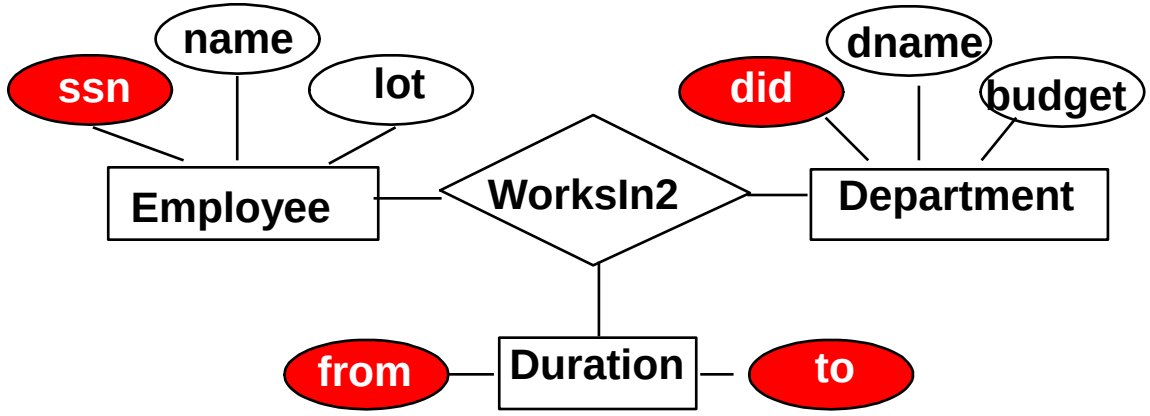
Examples

- Consider **address** of a trainee. Is it an entity or relationship?
- Consider **address** for a telephone company database, which has to keep track of how many and what type of phones are available in any one household, who lives there (there may be several phone bills going to the same address) etc.
- How do we represent employment of a trainee by a particular employer?
- How do we represent a course edition?

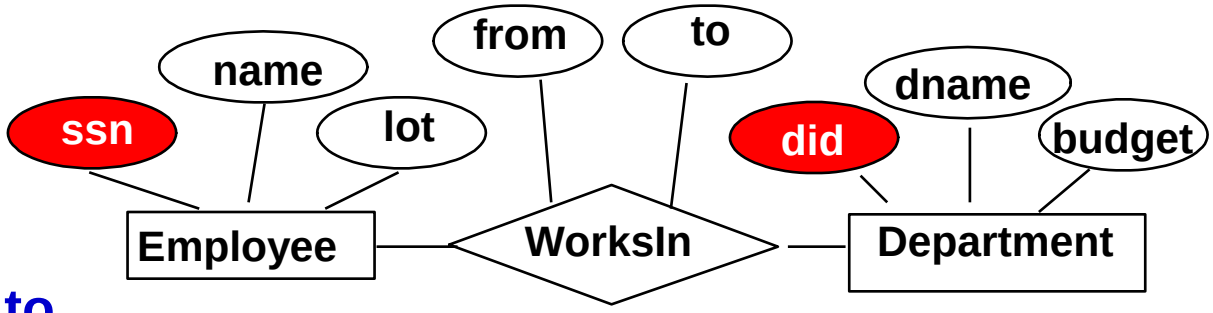
Entity vs. Attribute



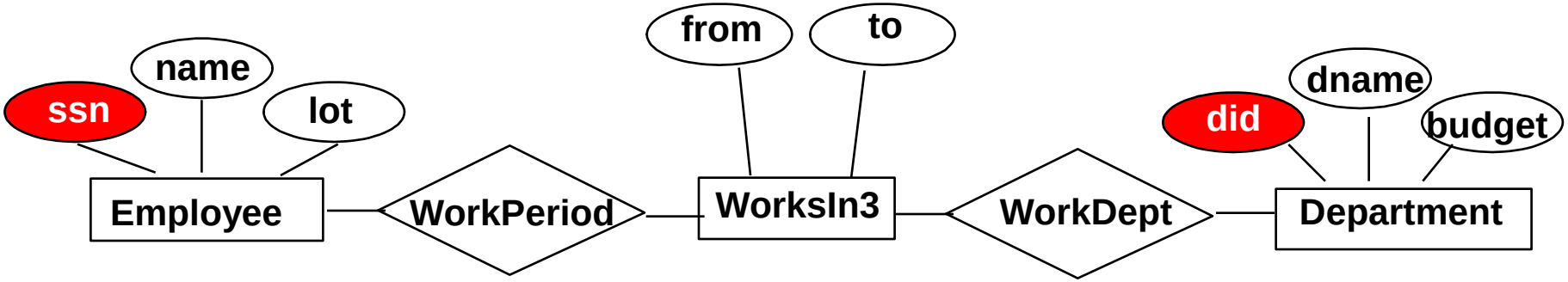
WorksIn does not allow an employee to work in a department for two or more periods.



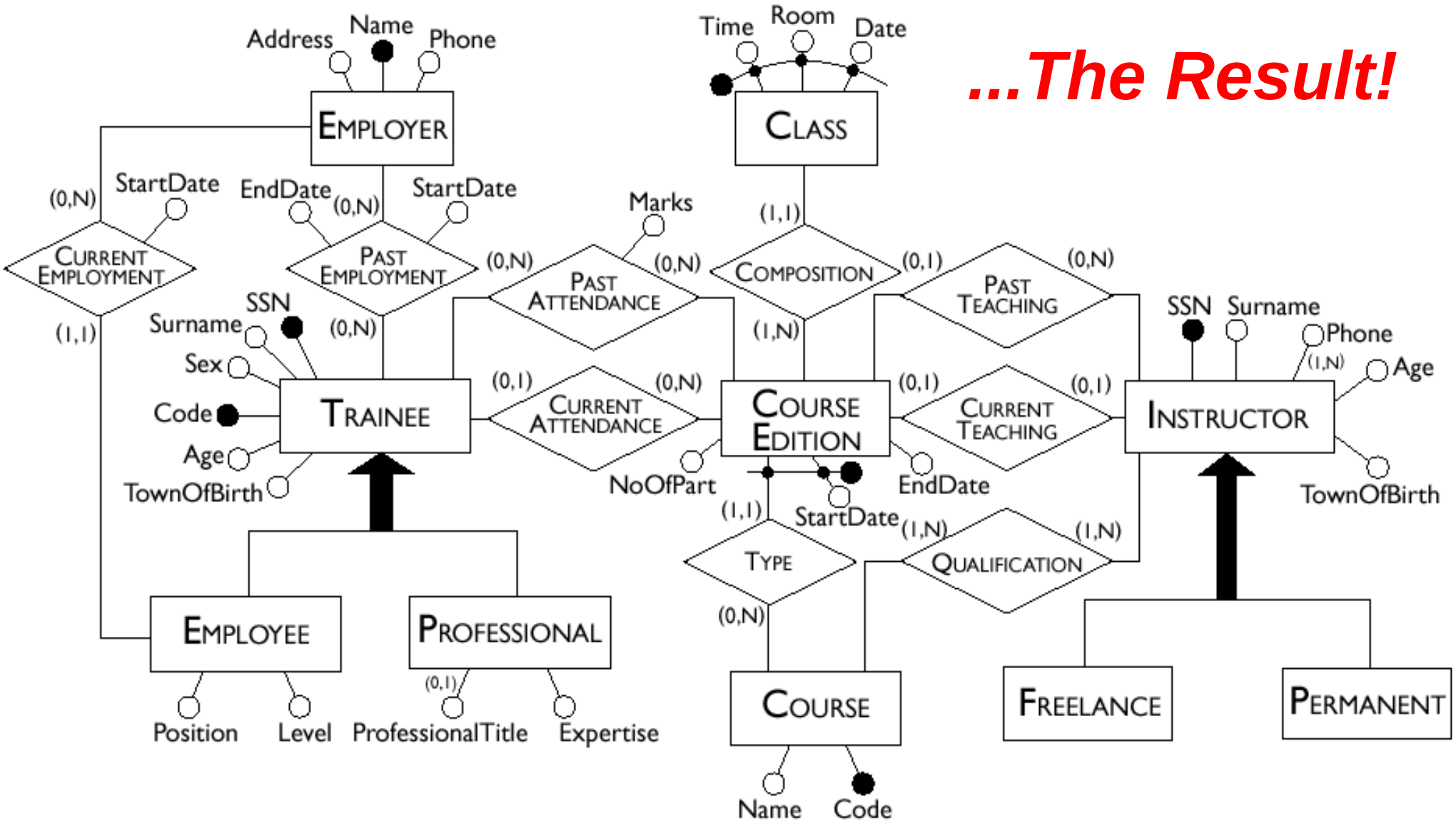
Entity vs. Relationship



WorksIn can also be turned into an entity to avoid the problem mentioned in the previous slide



...The Result!


Address
Name
Phone
EMPLOYER
Time
Room
Date
CLASS
(0,N)
StartDate
CURRENT EMPLOYMENT
EndDate
(0,N)
StartDate
PAST EMPLOYMENT
Marks
(0,N)
PAST ATTENDANCE
(0,N)
(1,1)
COMPOSITION
(0,1)
(0,N)
PAST TEACHING
(1,1)
SSN
Surname
Sex
Code
Age
TownOfBirth
(0,N)
TRAINEE
(0,1)
(0,N)
CURRENT ATTENDANCE
(1,N)
COURSE EDITION
(0,1)
(0,1)
CURRENT TEACHING
INSTRUCTOR
SSN
Surname
Phone
(1,N)
Age
TownOfBirth
NoOfPart
StartDate
EndDate
(1,1)
TYPE
(1,N)
(1,N)
QUALIFICATION
(0,N)
EMPLOYEE
PROFESSIONAL
(0,1)
Position
Level
ProfessionalTitle
Expertise
COURSE
Name
Code
FREELANCE
PERMANENT

Documentation of an E-R Schema

- An Entity-Relationship schema is rarely sufficient by itself to represent all the aspects of an application in detail.
- It is therefore important to complement every E-R schema with support documentation, which can facilitate the interpretation of the schema itself and describe properties of the data that cannot be expressed directly by the constructs of the model.
- A widely-used documentation concept for conceptual schemas is the ***business rule***.

Business Rules

- Business rules are used to describe the properties of an application, e.g., the fact that an employee cannot earn more than his or her manager.
- A business rule can be:
  - ✓ the ***description*** of a concept relevant to the application (also known as a ***business object***),
  - ✓ an ***integrity constraint*** on the data of the application,
  - ✓ a ***derivation rule***, whereby information can be derived from other information within a schema.

Documentation Techniques

- Descriptive business rules can be organized as a *data dictionary*. This is made up of two tables: the first describes the entities of the schema, the others describes the relationships.
- Business rules that describe constraints can be expressed in the following form:

  <concept> must/must not <expression on concepts>
- Business rules that describe derivations can be expressed in the following form:

  <concept> is obtained by <operations on concepts>

Example of a Data Dictionary

Entity	Description	Attributes	Identifier
EMPLOYEE	Employee working in the company.	Code, Surname, Salary, Age	Code
PROJECT	Company project on which employees are working.	Name, Budget, ReleaseDate	Name
....	....	....	....

Relationship	Description	Entities involved	Attributes
MANAGEMENT	Associate a manager with a department.	Employee (0,1), Department (1,1)	
MEMBERSHIP	Associate an employee with a department.	Employee (0,1) Department (1,N)	StartDate
....	....	....	....

Examples of Business Rules

Constraints
(BR1) The manager of a department must belong to that department. **(BR2)** An employee must not have a salary greater than that of the manager of the department to which he or she belongs. **(BR3)** A department of the Rome branch must be managed by an employee with more than 10 years' employment with the company. **(BR4)** An employee who does not belong to a particular department must not participate in any project. ....
Derivations
(BR5) The budget for a project is obtained by multiplying the sum of the salaries of the employees who are working on it by 3. ....

Comparison of ER and Class Diagrams

- ER diagrams allow N-ary relationships, N≥2; Class diagrams only allow binary relationships.
- ER diagrams allow multi-valued attributes, class diagrams do not.
- ER diagrams allow the specification of identifiers (an often-encountered type of constraint), while class diagrams do not.
- Class diagrams allow dynamic classification, but ER diagrams do not.