The Entity-Relationship Model

The Entity-Relationship 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

Entities

- Represent classes of objects (facts, things, people,...) that have properties in common and an autonomous existence, e.g., City, Department, Employee, Purchase and Sale.
- An instance of an entity is an object in the class represented by the entity, e.g., Stockholm, Helsinki, are examples of instances of the entity City; Peterson and Johnson 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, e.g., an employee is represented by a tuple containing the name, surname, age, and other attributes.]

Relationships

- They represent associations 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 (Johannes, Stockholm), or the pair (Peterson, Oslo), are examples of instances of the relationship Residence.
- Note: The collection of instances of a relationship is by definition a set, not a bag; i.e., no duplicates!
Examples of Relationships

- **STUDENT** \(\rightarrow\) **EXAM** \(\rightarrow\) **COURSE**
- **WORKPLACE** \(\rightarrow\) **RESIDENCE** \(\rightarrow\) **CITY**

Example of Instances for Exam

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

What Does A 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

- Contains, Contains Part
- Requests, Service
- XOR, FillsBy
- AND, Generates

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!"
Attributes

- Describe elementary properties of entities or relationships.
- For example, Surname, Salary, and Age are attributes of Employee, while Date and Mark are attributes of relationship Exam between Student and Course.
- An attribute associates with each instance of an entity (or relationship) one or more values belonging to its domain.
- Attributes may be single-valued, or multi-valued.

Attribute Examples

- 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.

| Composite Attributes

Cardinalities

- These are specified for each entity participating in a relationship and describe the maximum and minimum number of relationship instances that an entity instance can participate in.
Cardinalities (cont’d)

- In principle, a cardinality is any pair of non-negative integers \((n,m)\) such that \(n \leq m\), or a pair of the form \((n,N)\) where \(N\) means "any number".
- If minimum cardinality is 0, entity participation in a relationship is optional. If minimum cardinality is 1, 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

- **Course** \((3,3)\)
  - "A course meets three times a week"
- **Room** \((0,40)\)
  - "A room can have up to 40 meetings per week"
- **Day** \((0,N)\)
  - "A day can have an unlimited number of meetings"

Cardinalities of Attributes

- Describe the minimum and maximum number of values an attribute can have.
- In most cases, the cardinality of an attribute is \((1,1)\) and is omitted (single-valued attributes)
- The value of an attribute, however, may also be null, or there may be several values (multi-valued attributes)
Cardinalities (cont’d)
- Multi-valued attributes often represent situations that can be modeled with additional entities.
- For example, the ER diagram of the previous slide can be revised into:

```
Surname Licensed# Person (0,N) Owns (1,1) Car Registration#
```

Identifiers
- Identifiers (or keys) consist of one or more attributes which identify uniquely instances of an entity.
- For example, Person may be identified by the attribute socialInsurance. Alternatively, it may be identified by firstName, middleName, lastName, address.
- In most cases, an identifier is formed by one or more attributes of the entity itself (internal identifier).
- Sometimes, the attributes of an entity are not sufficient to identify its instances and other entities are involved in the identification (external identifiers).
- An identifier for a relationship consists of identifiers for all the entities it relates.

Examples of Identifiers
- Internal, single-attribute:
  - COURSE: courseName, day, hour
- Internal, multi-attribute:
  - PERSON: firstName, middleName, lastName, address
- External, multi-attribute:
  - UNIVERSITY: name, city, address

General Observations on Identifiers
- An identifier may consist of 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 may involve an entity that is itself identified externally, as long as cycles are not generated.
- Each entity must have at least one (internal or external) identifier. 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!)
More Examples

- 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...

Suppose we want information on course offerings over the years. What does each of the following identifiers say about the application?
- name;
- name, sem, year;
- sem, year;
- name, dept;
- dept, year.

Generalizations

- Represent logical links between an entity E, known as parent entity, and one or more entities E₁,...,Eₙ, called child entities, of which E is more general.
- We then say that E is a generalization of E₁,...,Eₙ and that the entities E₁,...,Eₙ are specializations of E.

Properties of Generalization

- Every instance of a child is also an instance of the parent.
- 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 is an instance of one of its children, otherwise it is partial.
- A generalization is exclusive if every instance of the parent is at most an instance of one child, otherwise overlapping.
- The generalization Person, of Man and Woman is total and exclusive (assuming a person is either a man or a woman).
- The generalization Vehicle of automobile and bicycle is partial and exclusive.
- 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 involves a hierarchy of generalizations, including several levels.
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 store 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)

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 has attended in the past, with the final marks out of ten;
- **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

**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 instructors**

  - 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 specific types of trainees**

  - For each instructor (about 50), we will hold surname, age, sex, birth, areas of expertise and, if appropriate, the professional title. In addition, we will hold all telephone numbers, the edition of courses taught, those bought 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 freelancers.

- **Phrases relating to employers of trainees**

  - For each employer of a trainee the database will store name, address and telephone number.

- **Phrases relating to trainees**

  - For each trainee (about 5000), identified by a code, the database will hold the social security number, surname, age, sex, 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 instructors**

  - For each instructor (about 50), we will hold surname, age, sex, birth, all telephone numbers, the edition of courses taught, those bought 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 freelancers.

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.

**Documentation of an E-R Schema -- The Data Dictionary Revisited**

- 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.

**The Data Dictionary**

- 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

<table>
<thead>
<tr>
<th>Entity</th>
<th>Description</th>
<th>Attributes</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>Employee working in the company.</td>
<td>Code, Surname, Salary, Age</td>
<td>Code</td>
</tr>
<tr>
<td>PROJECT</td>
<td>Company project on which employees are working.</td>
<td>Name, Budget, ReleaseDate</td>
<td>Name</td>
</tr>
</tbody>
</table>

### 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.

### ER vs Class Diagrams

- ER diagrams allow N-ary relationships, N\geq2; 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), class diagrams do not.
- Class diagrams allow dynamic classification, but ER diagrams do not.
- Class diagrams can have associated methods, constraints and pre/post-conditions.