The Entity-Relationship Model

Entitles, Relationships and Attributes
Cardinalities, Identifiers and Generalization
Documentation of E-R Diagrams and Business Rules

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

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.)

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 (Johansen, Stockholm), or the pair (Peterson, Oslo), are examples of instances in the relationship Residence.

Examples of Entities
Examples of Relationships

- **STUDENT** connected to **EXAM** connected to **COURSE**
- **EMPLOYEE** connected to **RESIDENCE** connected to **CITY**

Example of Instances for Exam

- **Student** connected to **Exam** connected to **Course**

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

- Course connected to Supplier connected to Department
- Department connected to Product connected to Supplier

AND/XOR for Relationships

- **AND**
  - Orders connected to Supplier
  - Supplier connected to Department

- **XOR**
  - Fulfilled connected to Shipment
  - Shipment connected to invoice

- **OR**
  - Orders connected to Product

- **Generated by**
  - Product connected to Supplier
  - Supplier connected to Department
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.

Attributes 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.

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 \(n \leq m\) 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

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

Instantiating ER Diagrams

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

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 represent situations that can be modeled 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.
- 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

- Internal, single-attribute
- Internal, multi-attribute
- External, multi-attribute

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

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 the 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, motorbike) that are neither cars nor bicycles.
- 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.
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.

For each trainee (about 5,000), identified by a code, we will store the surname, age, sex, place of birth, employer's name, address and telephone number. For each employer of a trainee the database will store name, address and telephone number.

For each course offered, we will store the title of the course, the course taught, those taught in the past, with the final marks out of ten. If a course 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.
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:
  - 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.

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.

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. Relationship

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

...The Result!

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

<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>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Description</th>
<th>Attributes</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>Employee manager with a department.</td>
<td>Department (1..N)</td>
<td>Department (1..N)</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>Associate an employee with a department.</td>
<td>Department (1..N)</td>
<td>StartDate</td>
</tr>
</tbody>
</table>
Examples of Business Rules

<table>
<thead>
<tr>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BR1) The manager of a department must belong to that department.</td>
</tr>
<tr>
<td>(BR2) An employee must not have a salary greater than that of the manager of the department to which he or she belongs.</td>
</tr>
<tr>
<td>(BR3) A department of the Rome branch must be managed by an employee with more than 10 years' employment with the company.</td>
</tr>
<tr>
<td>(BR4) An employee who does not belong to a particular department must not participate in any project.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Derivations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(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.</td>
</tr>
</tbody>
</table>

Comparison of ER and Class Diagrams

- ER diagrams allow N-ary relationships, No2: 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.