Part 3: ER-Model I

Elmasri/Navathe:

- Chap. 3, Data Modeling Using the Entity-Relationship Model

I have also used the following sources:

- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Ed., Ch. 2, “Entity-Relationship Model”.
- Ramakrishnan: Database Management Systems, Mc-Graw Hill, 1998, Ch. 14, “Conceptual Design and the ER-Model”
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 2, Oldenbourg, 1997.
- Rauh/Stickel: Konzeptuelle Datenmodellierung (in German), Teubner, 1997.
- Teorey: Database Modeling and Design, Third Edition, 1999.
- Barker: CASE*Method, Entity Relationship Modelling, Oracle/Addison-Wesley, 1990.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.

Objectives

- You should be able to explain the three phases of database design (and why it is done in these phases).
- You should be able to explain the significance of the ER-model for database design.
- You should be able to name and explain the basic constructs of the entity-relationship model.
- You should be able to develop ER-diagrams (schemas) for a given application (will certainly be tested in the exam).
- You should be able to compare and critically evaluate given ER-diagrams (likely to be tested in the exam).

Overview

Database Design (1)

- We want to support some tasks in the real world by programs which need persistantly stored data.
- Methods from software engineering should be used, or specialized/modified for such data-intensive programs.
- The specification of programs and data is intertwined.
- Database design is the process of developing a database schema for a given application. It is a subtask of the overall software engineering effort.
- Data is an independent resource: Often later additional programs should be developed based on the collected data.

Database Design (2)

- During database design, we have to build a formal model of some aspects of the real world ("Miniworld").
- The real world is the measure of correctness for our database: Database states should correspond to states of the real world.
- Database design is not easy:
  - You have to learn about the application domain.
  - Exceptions: The real world is very flexible.
  - Size: Database schemas can be very big.

Database Design (3)

- As any complicated task, DB design is done in several steps.
- There are usually three schema design phases:
  - Conceptual Database Design produces the initial model of the miniworld in a conceptual data model (like the Entity-Relationship-Model).
  - Logical Database Design consists of the transformation of this schema into the data model supported by the DBMS which will be used (often the relational model).
  - Physical Database Design aims at improving the performance of the final system. For instance, indexes and storage parameters are selected during this phase.

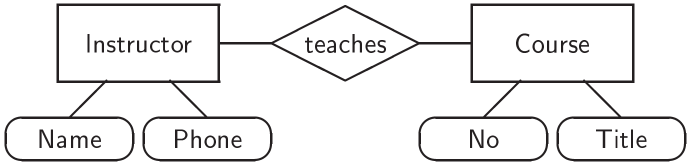
Database Design (4)

Why multiple design phases?

- In general, it is good to separate problems, and to attack them one after the other.
- E.g., during conceptual design, one does not have to worry about performance aspects / limitations of a specific DBMS. One can concentrate on producing a correct model of the real world (this is difficult enough).
- Also, special features of the DBMS do not influence conceptual design, and the logical design only in part. Thus, if one later should use a different DBMS, the conceptual design is not invalidated.

Example (1)

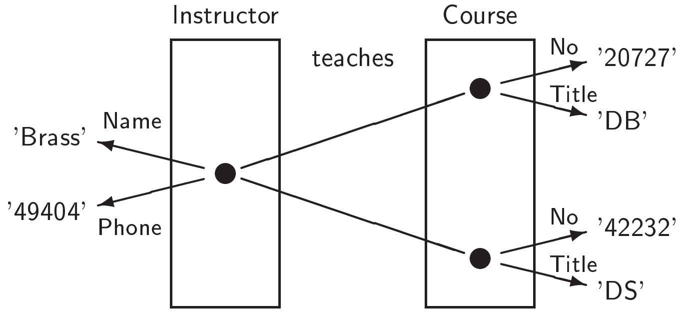
ER-Schema in Graphical Notation:



- This miniworld contains instructors and courses.
- Instructors teach courses.
- Instructors have a name and a phone number.
- Courses have a number (like "20727") and a title.

Example (2)

Possible State:



The Entity-Relationship Model (1)

- It is called a “semantic data model”, because it is closer to the real world than e.g. the relational model.
  - For instance, in the ER-model, one talks about persons. In the relational model, only about their names/numbers.
  - In the ER-model, there is a distinction between entities and relationships. In the relational model, both are represented by relations.

    So this expressiveness/semantical richness is not needed to satisfy the information requirements of the applications. But it makes the correspondence between the database schema and the real world clearer (like a comment).

The Entity-Relationship Model (2)

- Proposed by Peter P.-S. Chen (1976).
- An International Conference on the Entity-Relationship Approach has occurred almost every year since 1979.
- There is a useful graphical notation. This helps to get a better overview, to “see” the structure of the data.
  It also helps to communicate with the future users.
- There is no (commercial) entity-relationship DBMS.
  So a schema transformation into another data model is unavoidable.
- Many variations of the ER-Model have been proposed.

The Entity-Relationship Model (3)

- There are specialized graphical editors and other design tools.
- E.g. Oracle Designer is a CASE tool for DB-applications, and one component of it is the ER Diagrammer.
- The ER-model is a standard tool for conceptual DB design.
  However, recently, object-oriented methods are also used.

  Many people believe that UML (unified modelling language) is “the future”.
- But: All design formalisms are based on the ER-model.
  Knowledge of the ER-model is an important foundation for any DB design.

Overview

1. Introduction (Database Design Overview)
2. ER-Model: Entities, Relationships, Attributes
3. Style and Quality

Basic ER-Model Concepts (1)

Entities:

- Objects in the miniworld about which information has to be stored. E.g. persons, books, courses, exams.
- It does not matter whether entities have a physical existence (so that you can touch them) or only a conceptual existence.
- At each time instant, the miniworld that has to be modelled can contain only a finite number of entities.

  E.g. “all numbers” (infinitely many) cannot be entities.
- It must be possible to distinguish entities from each other, i.e. they must have some identity.

  So ants in a heap do not qualify as entities.

Basic ER-Model Concepts (2)

Data Type Elements:

- Values from some possibly infinite set, which you can store and print.
- E.g. strings, numbers, date values, lengths, pictures, ...
- You cannot store a person (an entity), but you can store his/her name (a data type element).
- Data types have names D (like `string`) and have value sets, which we will denote by $|D|$ (e.g. the set of all strings).

  In addition, data types also have operations, e.g. $+$, $-$, $*$, $/$, or concatenation.

Basic ER-Model Concepts (3)

Non-Standard Data Types:

- Most current DBMS have some predefined set of data types which they support. A typical selection are strings, numbers, date values, and containers for binary data of any size.
- There are extensible database management systems, which allow to define new data types (e.g. for geometrical data).
- You may use non-standard data types in ER-schemas, but:
  - You must clearly document its semantics.
  - During logical DB design, you have to devise a representation via the data types offered by the DBMS.

Basic ER-Model Concepts (4)

Attribute:

- A property or characteristic of an entity (or a relationship).
- E.g. the title of this course is “Database Management”.
- The value of an attribute is an element of a data type like string, integer, date. So it has a printable representation.

Relationship:

- Relation between pairs of entities (“binary relationship”).

  Some authors allow relationships involving more than two entities. My experience shows that this often leads to errors.
- E.g. I (a person) teach “Database Management” (a course).

Basic ER-Model Concepts (5)

Entity-Type:

- Set of similar entities (with respect to the information which has to be stored about them), i.e. entities which have the same attributes.
- E.g. all faculty members of this university.

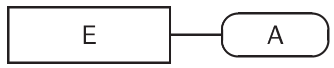
Relationship-Type:

- Set of similar relationships.
- E.g. “X teaches course Y”.
- We talk sometimes of “Relationship” when we really mean a “Relationship-Type”.

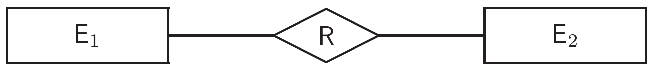
ER-Schemas: Graphical Syntax (1)

- Entity-Type E:

- Attribute A of Entity-Type E:

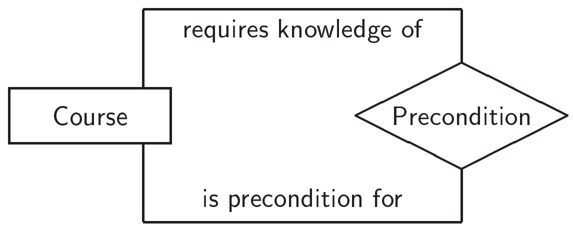


- Relationship R between Entity-Types E_1 and E_2:



ER-Schemas: Graphical Syntax (2)

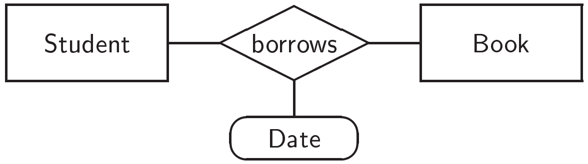
- Relationships can also exist between entities of the same type (“recursive relationships”):



- In this case, “role names” must be attached to the connecting edges.
- You may indicate roles in this way for any relationship.

ER-Schemas: Graphical Syntax (3)

- Relationships can have attributes, too:



- This means that we store a date for every pair of student X and book Y such that X borrowed Y.

Formal Graphical Syntax (1)

- An ER-Diagram contains boxes, diamonds, and ovals, plus interconnecting lines.
- Interconnecting lines are only allowed between
  - a box and a diamond,
  - a box and an oval, and
  - a diamond and a oval.
- In addition, these constraints must be satisfied:
  - A diamond must have exactly two connecting lines to boxes. There may be any number to ovals.
  - An oval must have exactly one connecting line.

Formal Graphical Syntax (2)

- Boxes, diamonds, and ovals are each labelled by a string.

  The string is written into the construct.
- The names of boxes must be unique in the entire diagram.

  I.e. you cannot have two boxes with the same label.
- The names of ovals must only be unique for a single box or diamond.

  I.e. you cannot have two ovals with the same name which are linked to the same box (resp. diamond).
- Diamonds must be uniquely identified by name and connection.

  I.e. you cannot have two diamonds which are linked to the same two boxes and have the same name.

Exercise

Define an ER-schema (diagram) for the following application:

- We want to create a database with information about researchers in the field of databases.
- About each researcher, we want to store his/her last name, first names, email address, and homepage (URL).
- We also want to store his/her current affiliation (university, company, research institute).
- About each possible affiliation (university etc.) we want to store its name, URL, and country.

  Is additional information needed for creating a sorted list? E.g. the “Univ. of Pittsburgh” should be listed under “P”.

ER-Schemas vs. ER-Diagrams (1)

- Usually, not all schema information is denoted in an ER-diagram:
  - The attributes have a data type, which must be specified in a complete ER-schema.
  - Sometimes an ER-diagram becomes clearer if the attributes are not shown.

    It is not uncommon in practice that an entity-type has 50 attributes.
  - If you want to attach comments, this also does not look good in ER-diagrams.

ER-Schemas vs. ER-Diagrams (2)

- So there should be somewhere a “real schema” (e.g. in textual form), and the ER-diagrams are only excerpts used for illustration purposes.
- However, there is no agreement for a textual syntax for writing down complete schemas, whereas there is some agreement on ER-diagrams.
- You should learn here the graphical syntax (and the fact that it might not show everything).
- Modern design tools solve the problem: For instance, if you click on an entity-type in the diagram, further information will be shown.

ER-Schemas: Semantics (1)

Definition:

A DB state $\mathcal{S}$ interprets the symbols in the schema by defining

- a finite set $\mathcal{S}(\mathsf{E})$ for every entity-type E,
- a mapping $\mathcal{S}(\mathsf{A})\colon \mathcal{S}(\mathsf{E}) \to |\mathsf{D}|$ for every attribute A of an entity-type E, where D is the data-type of A and $|\mathsf{D}|$ is the domain (value set) of this data type,
- a relation $\mathcal{S}(\mathsf{R}) \subseteq \mathcal{S}(\mathsf{E}_1) \times \mathcal{S}(\mathsf{E}_2)$ for every relationship R between entity types E_1 and E_2,
- a mapping $\mathcal{S}(\mathsf{A})\colon \mathcal{S}(\mathsf{R}) \to |\mathsf{D}|$ for every attribute A of a relationship R.

ER-Schemas: Semantics (2)

Consequences for Attributes:

- There will later be extensions, but the basic notion of a attribute requires that
  - Attributed values are defined (no unknown values).
  - Attributes are single valued (no multiple/set values).
  - Attribute values are atomic (no record structures).

    If you have a data type which is a record structure (like date values), you can access the single components (day, month, year) with the data type operations. But the basic ER-model treats all attribute values as atomic, i.e. it adds no record structures besides those already given by the data types.

ER-Schemas: Semantics (3)

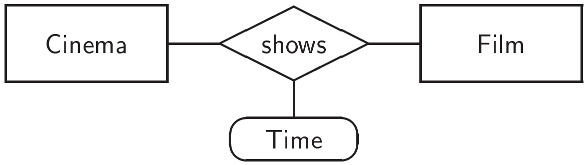
Consequence for Relationships:

- A binary relationship is interpreted by a set of entity-pairs.
- A set either contains an element or does not contain it. A element cannot be contained "two times".
- This means that a relationship either exists between two entities or it does not exist between these entities.
- There cannot be multiple connections between the same two entities with respect to the same relationship type.
- So if the relationship has an attribute, its value must be unique for any pair of entities.

ER-Schemas: Semantics (4)

Example:

- Consider this example:



- Suppose a cinema shows the same film at 3pm and at 6pm.
- Can we store this information in given schema?

Overview

Proper Naming (1)

General Recommendations:

- Names should be understandable / self-documenting.
  The correspondence to the real world must be clear. If necessary, attach comments, explanations, examples.
- Names should not be too long.
  Names longer than 15–20 characters become unwieldy.
- Choosing good names sometimes needs some thought. But the invested time will later pay off.
  Discussing the names with other people might help.
- Whatever you do, try to be consistent:
  Use abbreviations consistently. Use "_" consistently. If you develop in a team, all members should use the same style.

Proper Naming (2)

Entity-Types:

- Usually, nouns are used for entity-types.
- I am using the singular form, which names a single entity (in conformance with Elmasri/Navathe).

  Some authors use plural forms. After the translation into the relational model, the plural form for tables seems first more approriate. However, when you introduce a tuple variable in SQL, the singular form is more intuitive. Only be consistent.
- Names like “`Customer_Data`” are not good: You should create a model of the real world, and not of the database.

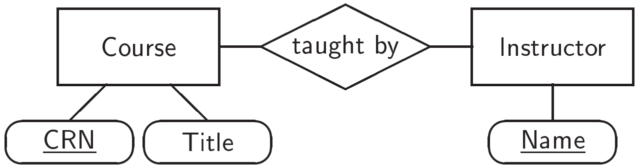
Proper Naming (3)

Relationships:

- Often, verbs are used for relationship-names.
- Relationship-names (e.g. “teaches”) should read from top to bottom and from left to right.
- Prepositions like “of” are quite unspecific. However, they can be combined with another word to make them clearer (e.g. “teacher of”, “lies in”).
- Some people also use nouns like “`Teaching_Assignment`”.

Attribute vs. Entity (1)

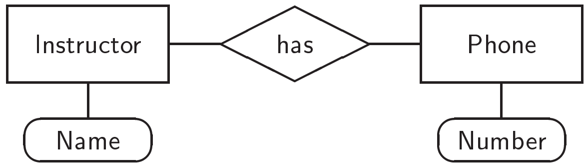
- Sometimes it is not clear whether a new entity-type is needed:



- One could add the name of the instructor as an attribute to the course.
- However, if later more information has to be stored about instructors (e.g. phone number), a bigger change is needed.

Attribute vs. Entity (2)

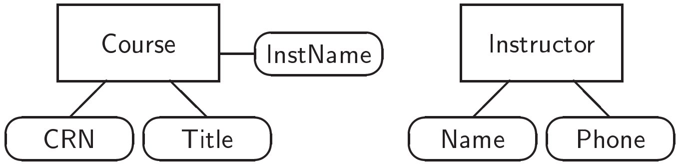
- In principle, every attribute can be moved to an entity-type of its own:



- However, this design would only be interesting for the phone company. Otherwise, it is unnecessarily complicated.
- Try to avoid entities which are only data type values.

Attribute vs. Relationship

- In principle, a relationship can be represented implicitly by attributes with the same value:



- This is not right in the ER-model. Relationships should be explicitly shown.

  In the relational model, it would be right. There is no explicit "relationship" construct in the relational model.

Redundant Information (1)

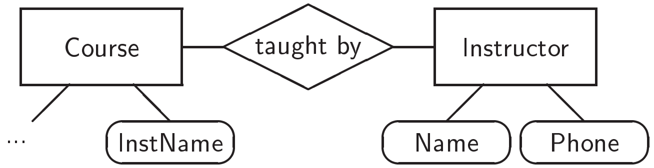
- Redundant information in an ER-schema is bad, because it is an unnecessary complication during conceptual design.

  You have to specify at least an integrity constraint which ensures that both copies of the information are always consistent.

- Redundant information may be introduced during physical design for performance reasons. Also, you may later define views which contain derived information.

Redundant Information (2)

- For instance, you should not repeat attributes from other entity types which can be reached via a relationship:

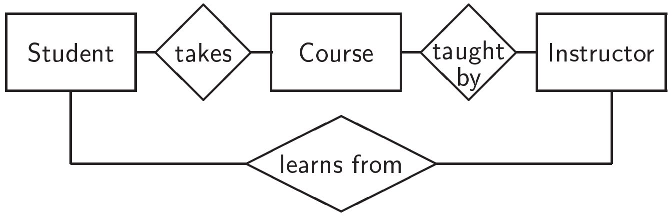


- Many students who are already trained in relational database design make this mistake.

  There will be such a column when the schema is translated into the relational model (see below). But there are no relationships in the relational model. Having a relationship <u>and</u> an attribute pointing to another entity is redundant.

Redundant Information (3)

- For instance, it is a mistake to define the composition of two relationships again as a relationship ("shortcut"):



- Different paths of relationships between two entity-types should be carefully inspected for possible redundancies.

  Some dependency often exists ($\rightarrow$ integrity constraint).