Part 3: ER-Model I

References:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999. Chapter 3, "Data Modeling Using the Entity-Relationship Model"
- 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

- The three phases of database design.
  Plus reason/motivation for multiple phases.
- Significance of the ER-model for database design.
- Basic constructs of the entity-relationship model.
- Development of ER-diagrams (schemas in the ER-model) for a given application.
- Comparison and critical evaluation of given ER-diagrams.

Overview

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

Database Design (1)

- Overall goal: Develop programs to support given real world tasks. These programs need persistently 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 will be developed based on the collected data.
  Also, ad-hoc queries may be used.

Database Design (2)

- During DB design, a formal model of some aspects of the real world ("Miniworld", "Domain of Discourse") must be built.

  Questions about the real world should be answered from the database. A list of such questions can be an important input for database design.
- The real world is the measure of correctness for the schema: DB states should correspond to states of the real world.
- Database design is not easy:
  - The designer must 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 transforming this schema into the data model supported by the DBMS to 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?

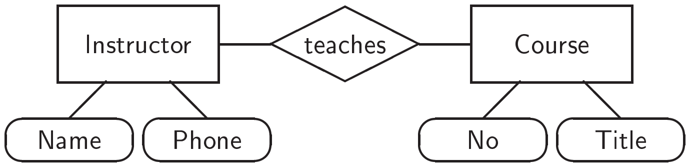
- Allows problems to be separated and attacked one after the other.
- E.g., during conceptual design, there is no need to worry about performance aspects / limitations of a specific DBMS.

  Focus is on producing a correct model of the real world (this is difficult enough).
- DBMS features do not influence conceptual design, and only partially influence the logical design.

  Thus, the conceptual design is not invalidated, if a different DBMS is later used.

Example (1)

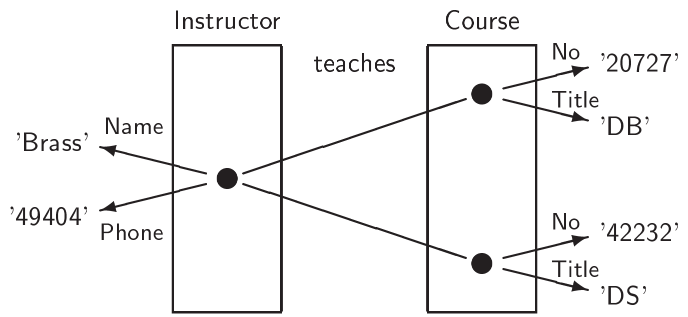
ER-Schema in Graphical Notation:



- 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 more closely resembles the real world than e.g. the relational model.
  - For instance, in the ER-model, persons are modelled. 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.

    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 Pin-Shan Chen (1976).
- An International Conference on the Entity-Relationship Approach has occurred almost every year since 1979.
- There is a useful graphical notation which helps to establish 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 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".
- 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 (and can be touched) or only a conceptual existence.
- At each 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 can be stored and printed.
- E.g. strings, numbers, date values, lengths, pictures, ...
- A person cannot be stored (an entity), but his/her name can be stored (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 definition of new data types (e.g. for geometrical data).
- Non-standard data types can be used in ER-schemas, but:
    - Their semantics must be clearly documented.
    - During logical DB design, a representation via the data types offered by the DBMS must be devised.

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.

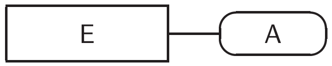
Relationship-Type:

- Set of similar relationships.
- E.g. "X teaches course Y".
- Often "Relationship" actually means "Relationship-Type".

ER-Schemas: Graphical Syntax (1)

- Entity-Type E:

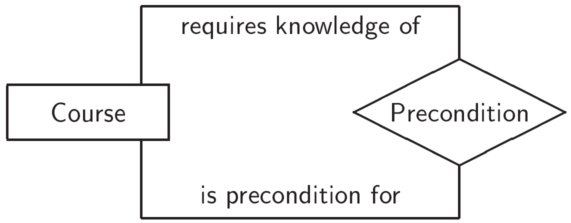
- Attribute A of Entity-Type E:



- Relationship R between Entity-Types E_1 and E_2:

ER-Schemas: Graphical Syntax (2)

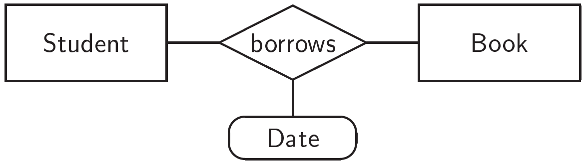
- Relationships can also exist between entities of the same type ("recursive relationships"):



- In this case, "role names" must be attached to the connecting edges.
- Roles may be indicated in this way for any relationship.

ER-Schemas: Graphical Syntax (3)

- Relationships can have attributes, too:



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

Formal Graphical Syntax, Ver. 1 (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, Ver. 1 (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.

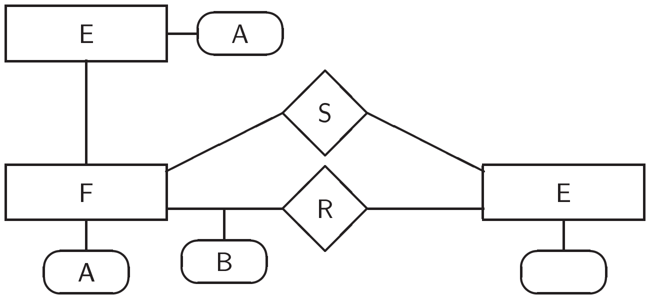
  I.e. you cannot have two boxes with the same label.
- The names of ovals must only be unique for a single box or diamond.

  I.e. you cannot have two ovals with the same name which are linked to the same box (or the same 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

Which errors does this diagram contain?



Exercise

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

- Create a database with information about researchers in the database field.
- For each researcher, store his/her last name, first name, email address, and homepage (URL).
- Also store his/her current affiliation (for simplicity, we assume that all researchers work at universities).
- For each university, 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.
  - Comments are useful, but do not look good in ER-diagrams.

ER-Schemas vs. ER-Diagrams (2)

- There should exist a "real schema" (e.g. in textual form). The ER-diagrams are then 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.
- The important thing to learn is the graphical syntax (and the fact that it might not show everything).
- Modern design tools solve the problem: E.g. clicking on an entity-type in the diagram shows further information.

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.

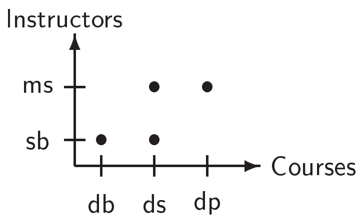
ER-Schemas: Semantics (2)

Example State:

- $\mathcal{S}$(Instructor) $= \{$sb, ms$\}$
- $\mathcal{S}$(Name) $= f_1$, where f_1(sb) = 'Brass', f_1(ms) = 'Spring'.
- $\mathcal{S}$(Phone) $= f_2$, where f_2(sb) $= 49404$, f_2(ms) $= 49429$.
- $\mathcal{S}$(Course) $= \{$db, ds, dp$\}$
- $\mathcal{S}$(No) $= g_1$, where
  g_1(db) $= 20727$, g_1(ds) $= 42232$, g_1(dp) $= 40492$.
- $\mathcal{S}$(Title) $= g_2$, where
  g_2(db) = 'Database Management',
  g_2(ds) = 'Data Structures',
  g_2(dp) = 'Document Processing'.

ER-Schemas: Semantics (3)

- $\mathcal{S}$(teaches) $= \big\{$(sb, db), (sb, ds), (ms, ds), (ms, dp)$\big\}$.
- The cartesian product $\times$ constructs the set of all pairs, e.g. $\mathbb{R} \times \mathbb{R}$ is the set of all (X, Y)-pairs where X and Y are real numbers.
- Here we construct (X, Y)-pairs where X is an instructor and Y is a course:



ER-Schemas: Semantics (4)

Consequences for Attributes:

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

    For a data type which is a record structure (like date values), single components (day, month, year) can be accessed 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 (5)

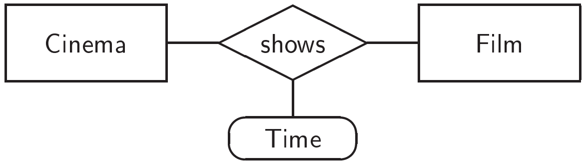
Consequence for Relationships:

- A binary relationship is interpreted by a set of entity-pairs.
- A set either contains an element or does not contain it. An 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 (6)

Example/Exercise:

- Consider this schema:



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

  ☐ Yes ☐ No

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 requires careful thought.
  But the invested time will pay off.

  Discussing the names with other people might help.

- Aim for consistency.

  Use abbreviations and "_" consistently. If developing 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 more approriate (except when declaring tuple variables). Oracle Designer asks for singular and plural forms (singular is used in the ER-diagram, plural for tables).

- Names like "`Customer_Data`" are not recommended:
  Create a model of the real world, not of the database.

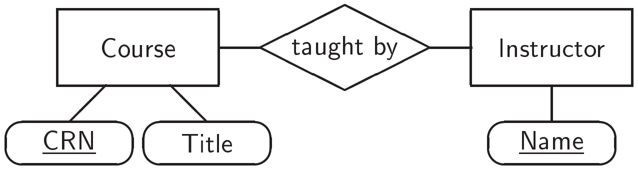
Proper Naming (3)

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").

  Actually, only "many-to-many relationsships" (see below) need later tables of their own and thus a unique name. For other relationships, simple prepositions might suffice.
- Some people also use nouns like "`Teaching_Assignment`".

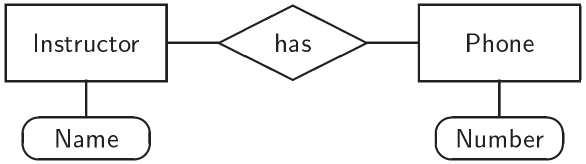
Attribute vs. Entity (1)



- Alternative: Instructor name as an attribute to the course, no entity type for instructors.
- However, if later more information has to be stored about instructors (e.g. phone number), a bigger change is needed.
- Entities are explicitly inserted/deleted. With the attribute, different spellings of the same name might be entered.

Attribute vs. Entity (2)

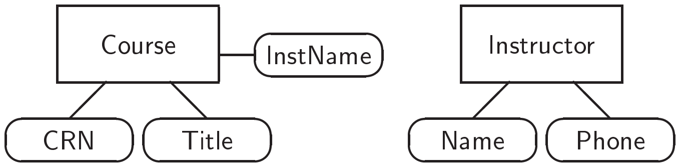
- In principle, every attribute can be moved to an entity-type of its own:



- 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)

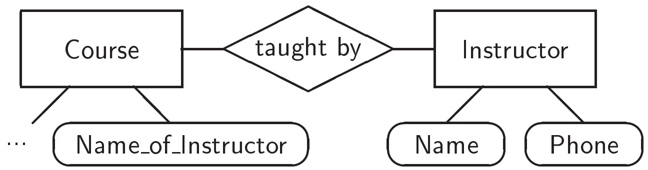
- An entity type, attribute, or relationship is redundant if it does not contain any new information, i.e. can be derived from the rest of the schema (e.g. date of birth and age).
- Redundant information in an ER-schema is an unnecessary complication during conceptual design.

  Specify at least an integrity constraint which ensures that both copies of the information are always consistent. Otherwise the schema is incorrect, because it allows states which can never happen in the real world.

  Derived constructs must be clearly marked (e.g. dashed).
- Redundant information may be introduced during physical design for performance reasons.

Redundant Information (2)

- Do not repeat attributes from other entity types which can be reached via a relationship:

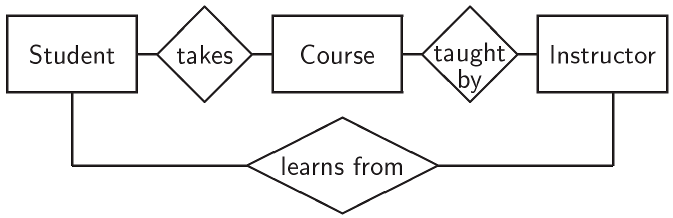


- Many students already trained in relational database design make this mistake.

  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)

- 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 (→ integrity constraint).