
Introduction to Databases, Fall 2003
IT University of Copenhagen

Lecture 3: E/R modeling

September 9, 2003

Lecturer: Rasmus Pagh

— Today's lecture —

- Practical info.
- What you should remember from last week.
- Data modeling - why and how?
- The basic entity-relationship (E/R) data model.
- E/R concepts and examples.
- Rules for designing a good E/R model.
- How to go from an E/R data model to a relational database schema.
- Advanced E/R concepts and examples.

— Practical info on hand-ins —

The first mandatory hand-in is due on Friday September 12, 11.59 AM.

You should put your hand-in in one of the teaching assistant's compartments on top of the cabinet in front of the student administration.

We will distribute the hand-ins to the teaching assistants. From week 4 you are expected to attend the exercises of the teaching assistant who corrected your hand-in. If you have a study group that would like to attend the same exercises, please **wrap up all hand-ins of the group together** (for the first hand-in).

From now on, a teaching assistant will be present in room 1.60 each Thursday 10-11 AM to answer questions about the hand-in for Friday.

— Practical info on the course book —

The course assumes that you buy (or have access to) the course book. It is your responsibility that you are able to read the mandatory exercises in the book, and that you know the course curriculum (lecture slides + course book + extra hand-outs) at the time of the exam.

If you plan on using another database textbook I should warn you that many database books do NOT cover our curriculum. I thus highly recommend that you get the book.

If you want to save money, you could share a book with others, and/or try to find a used book.

— What you should remember from last week —

In this lecture I will assume that you remember:

- The concepts of the relational data model:
 - Data model
 - Relation schema
 - Database schema (i.e., the collection of relation schemas)
 - Attributes
 - Relation instance
 - Atomic type

— Why data modeling? —

The way data is stored is *very important* for the way in which it can be accessed and manipulated using SQL.^a

A good data model is one where:

- It is easy to write correct and understandable queries.
- If the problem domain changes, it is easy to modify the data model and associated programs.

Sometimes it is also necessary to consider the data model's impact on the efficiency of database operations.

^aThis observation is also true for other programming languages such as Java.

— Database design methodology —

Today and next week we will cover the dominant design methodology for relational databases, which consists of three steps:

1. Identify all relevant **Entities** and **Relationships**, and describe them using so-called **E/R diagrams**.
2. Convert the E/R model to a number of relation schemas.
3. Eliminate (or reduce) redundancy by splitting relations. This process is called **normalization**. (Next week.)

Next: The basic entity-relationship (E/R) data model

— What is an entity? —

An **entity** is an object of some sort (not necessarily physical).

Examples of entities:

“Die Hard” (an abstract object), “Bruce Willis” (a physical object).

We group similar entities into **entity sets**.

Examples of entity sets: “Movies”, “stars”.

— Attributes of entity sets —

To every entity set we associate a number of **attributes**, which can be thought of as properties of the entity set.

Example:

The entity set “stars” might be given attributes *name*, *address*, *gender*, and *birthdate*.

Values of attributes:

For each entity in the entity set, the attributes will be associated with values (e.g., the name, address, gender and birthdate of Bruce Willis).

Like in the relational data model, we assume these values to be **atomic**.

— Relationships between entity sets —

Two entity sets may be connected in some way. In the E/R model this can be expressed as a **relationship**.

Example:

The “movies” and “stars” entity sets are connected by actors starring in movies. For example, “Die Hard” is connected to “Bruce Willis”.

Formally, a relationship between entity sets A and B is a set of tuples (a,b), where entity a belongs to A, and b belongs to entity B.

Didn't we see this definition before?
Answer in two slides...

— How to identify entities —

In most cases, an entity can be identified **uniquely** within its entity set by the values of its attributes.

That is, there are no two entities with the same attribute values.

A set of attributes whose values uniquely identify an entity in its entity set is called a **key** of the entity set.

That is, there can be no two entities with the same values on the key attributes.

If there are several possible keys, we choose one of them as the **primary key**.

— Relationships are relations —

What is called a relationship in the E/R model is *exactly* what we defined last time as a relation.

- Entity sets correspond to the domains of the values.
- Tuples correspond to connections between entities.
- We can draw relationship instances as a table, like relation instances:

<i>Movies</i>	<i>Stars</i>
Basic Instinct	Sharon Stone
Total Recall	Arnold Schwarzenegger
Total Recall	Sharon Stone

Note: We use the attribute values of a key for each entity set to identify entities (unrealistically assuming that name is a key for Stars and Movies).

— E/R diagrams

E/R diagrams is a graphical notation for E/R data models.

- Attributes are represented by ovals.
- Relationships are represented by diamonds.
- Entity sets are represented by rectangles, with lines to its attributes and relationships.

If an entity set has one or more keys, we underline the attributes in one of its keys.

[Figure 2.17 shown on slide]

— Problem session (10 minutes) —

Suppose we want to design a database for a bank, with information on costumers and their accounts, including:

- Names
- Addresses
- Phone numbers
- CPR numbers (social security numbers)
- Account numbers
- Account types
- Account balances

Design an E/R diagram for the database.

Next: E/R concepts and examples

— Multi-way relationships

Relationships may connect any number of entity sets.

[Figure 2.4 from Example 2.5 shown on slide]

— Multiplicity of relationships —

We use arrows to indicate the **multiplicity** of a relationship.

An arrow from a relationship R to an entity set E means that:

Every set of entities in the other entity sets of R is connected by R to **at most one** entity in E.

[Figure 2.3 from Example 2.4 shown on slide]

The multiplicity of a **binary** relationship (i.e., involving two entity sets) can be categorized as **one-one**, **many-one**, or **many-many**.

— Attributes on relationships —

Sometimes it is convenient to associate attributes with a relationship.

This is done in the obvious way.

[Figure 2.7 shown on slide]

— Parallel and self-relationships —

It is perfectly possible that there are several (different) relationships between the same entity sets.

[Figure 2.9 shown on slide]

Also, an entity set may be involved several times in the same relationship. In this case we label the lines with different name (called **roles**).

[Figure 2.5 from Example 2.6 shown on slide]

[Figure 2.6 from Example 2.7 shown on slide]

Next: Rules for designing a good E/R model

— Design principles for data modeling —

Good data modeling can be difficult, as there may be many choices of what data model to choose.

The following design principles can be useful when considering a design:

- Be faithful to the specification of the application.
- Avoid duplication and other redundant information.
- The KISS principle (Keep It Simple, Stupid). [Figure 2.11]
- Choose the right relationships. [Figure 2.17, 2.12]
- Use attributes when possible. [Figure 2.17]

— Problem session (10 minutes) —

Consider the E/R model of a bank database in Figure 2.14 on page 45 of GUW.

It tries to model the fact that accounts may be held jointly by several costumers, by letting each account belong to an “account set” .

Criticize the design:

- What design rules are violated and why?
- What modifications would you suggest?

Next: From an E/R model to a relational database schema

— From an entity set to a relation —

We first consider *entity sets that have keys*, i.e., where there can be no two entities with the same attributes.

The corresponding relation then simply has the attributes of the entity set as its attributes. Easy!

[Figure 2.17]

— From an a relationship to a relation —

We assume that the entity sets connected by the relationship all have a key.

The relation corresponding to a relationship then has the following attributes:

- The primary key attributes for each entity set involved in the relationship.
- Any attributes of the relationship itself.

It may be necessary to rename attributes if there are duplicate names.

[Figure 2.17, 2.12]

— Possibilities for combining relations —

Sometimes the translation to relation schemas results in more relations than necessary (or convenient).

Example: If there is a *many-one relationship* R between entity sets E and F , we can combine the relations corresponding to E and R into a single relation with a schema consisting of:

- The attributes of E ,
- the key attributes of F , and
- any attributes belonging to R .

[Figure 2.17]

— Problem session (5-10 minutes) —

Consider our revised version of the E/R diagram in Figure 2.14.

Convert the E/R data model into a relational database schema.

Next: Advanced E/R concepts and examples

— Modeling of constraints —

In addition to *what data* is stored, it is often important for a data model to specify *properties* of the data stored.

Such properties are expressed as **constraints**:

“Not all values are possible, only those where. . .”.

We already saw examples of **single value constraints** in the E/R model:

- Keys.
- Many-one relationships.

As we saw, such constraints are useful when converting the E/R model to a relational schema.

— Other kinds of constraints —

- *Referential integrity constraints*: Exactly one value exists in a given role.
- *Domain constraints* restricting the value of some attribute.
- *Multiplicity constraints* on the number of entities connected to a single entity by a relationship.
- *General constraints*, expressed in a constraint-expression language (more on this in the lecture covering chapter 7).

Examples:

- In E/R diagrams we may indicate by a rounded arrow the referential integrity constraint that the entity connected by some relationship is required to exist. [Figure 2.18]
- In E/R diagrams we may indicate multiplicity constraints on the lines connecting a relationship. [Figure 2.19]

— Constraints in the database schema —

Often, constraints are part of the database schema, and checked by the DBMS: Updates that violate a schema constraint are not allowed.

Example: In SQL, the schema

```
MovieStar(  
  name CHAR(30) PRIMARY KEY,  
  address CHAR(255),  
  gender CHAR(1),  
  birthday DATE)
```

declares the constraint that name is the primary key for MovieStar.

— Weak entity sets —

Sometimes entities in an entity set are not uniquely identified by their attribute values.

This may happen, e.g., when entities in one entity set are subunits of other entities. [Figure 2.20]

To get a key for a weak entity set E , we need to use attributes of other entity sets, namely the key attributes of the entities to which it has **supporting relationships**.

— Supporting relationships —

Suppose that E is a weak entity set, and the F is an entity set connected to E by a relationship R.

R is a supporting relationship for E if:

- It is a binary, many-one relationship from E to F.
- We can put a rounded arrow on the line from R to F in the E/R diagram, i.e., there is a unique entity in F corresponding to every entity of E.

Notes:

- If F has several supporting relationships to E we need to include the key attributes of E several times.
- E may itself be a weak entity set, so its keys may come from other entity sets.

— Weak entity sets in E/R diagrams —

- A weak entity set is drawn as a rectangle with **double** border.
- A supporting relationship is drawn as a diamond with **double** border.
- We underline any attributes of a weak entity set that are part of its key.

[Figure 2.20]

— Most important points in this lecture —

As a minimum, you should after this week:

- Be able to use the E/R data model to describe a database, including:
 - Choice of keys.
 - Deciding whether relationships are one-one, many-one, or many-many.
 - Deciding referential integrity constraints.
 - Drawing the E/R data model as an E/R diagram.
- Be able to convert an E/R data model to a relational database schema.

Next time

Next week we will see how to use so-called **normalization** to improve a relational database schema:

- Functional dependencies – the “bad guys” we wish to eliminate.
- Splitting relations to remove functional dependencies.
- Goal: Get the schema in “normal form” (3NF, BCNF, or 4NF).