Databases	Lecture 2
ENGR 3599	Jan 31, 2019

The Entity-Relationship Model

How do we understand the data that we want to manipulate in a given application? To understand the data, we need a framework that provides concepts and structures that we can use to understand properties of the data.

Such a model will help us determine, among others, how we can represent that data for an application, which manipulations we can perform on the data, or which queries will be easy and which will be hard.

Three classes of models:

- Conceptual models: abstract description of the data
- Logical models: implementation of a conceptual model in some kind of database model (relational, key-value, hierarchichal, ...)
- Physical models: implementation of a logical model in a specific database (MySQL, Microsoft SQL Server, ...) — each specific database has its own characteristics in terms of how to specify data layout to optimize various kinds of queries

We will mostly focus on conceptual and logical models in the course.

The most important conceptual model out there is the *Entity-Relationship Model*. It consists of entities, relationships, attributes, and constraints.

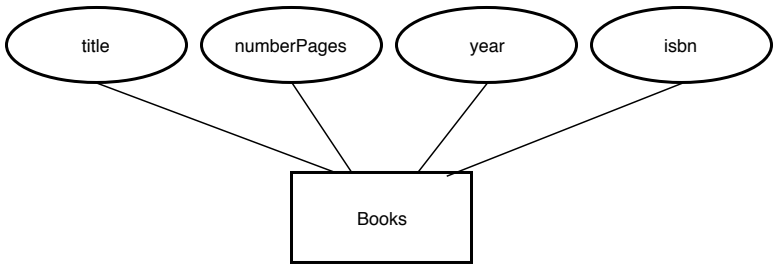
Entities

An entity is a "thing" that can be distinguished from other things. An entity is described by a set of attributes. Each attributes has a domain (integers, floating point numbers between 0 and 1, character strings, etc). An entity has a given value for each attribute.

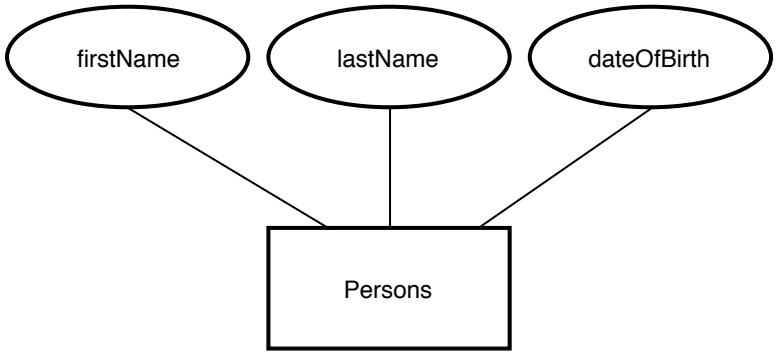
An entity-set is a named collection of distinct entities of the same kind (and therefore, with the same attributes).

We diagram an entity set with a named rectangle, linked to attributes each represented by a named oval.

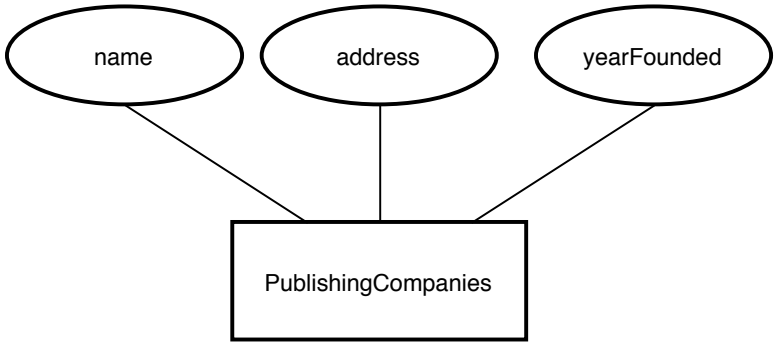
For example, a book can be an entity, with attributes *title*, *year*, *numberPages*, and *isbn*. We can diagram an entity set *Books* as:



We can diagram an entity set *Persons* as:



We can diagram an entity set *PublishingCompanies* as:



Relationships

A relationship is an association between two or more entities. For instance, an association between a book and a person (its author), or an associate between a book and a publishing company (its publisher).

A relationship set is a named collection of relationships of the same kind (associations between the same entities).

We say an entity *participate* in a relationship to indicate that said relationship associates that entity to other entities. Similarly, we say that entity sets participate in relationship sets.

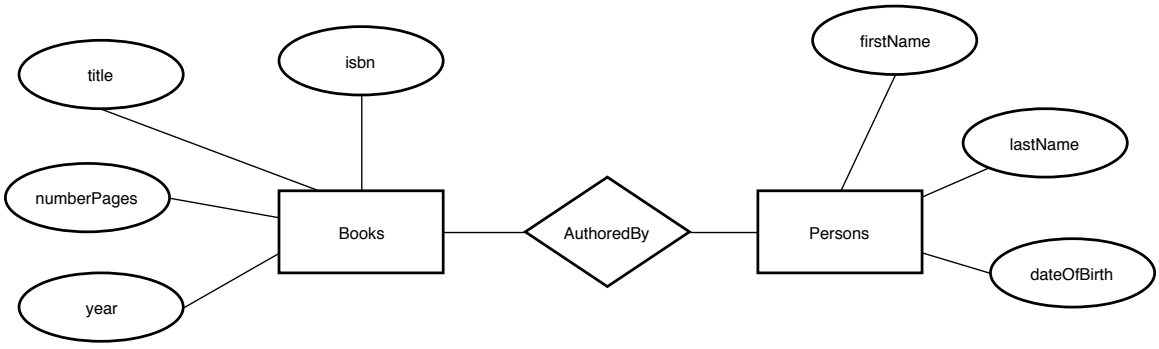
Formally, a relationship set is an n-ary mathematical relation over n entity sets. If $E_1, \ldots E_n$ are entity sets, a relationship set R over $E_1, \ldots, E_n$ is a subset of

$$E_1 \times \ldots \times E_n = \{(e_1, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}.$$

Each $(e_1, \ldots, e_n)$ in R is a relationship.

We can diagram a relationship set by a named diamond, with links to the entity sets that participate in the relationship set.

For example, we can associate books and persons via a *AuthoredBy* relationship set, associating a book with its author (or authors). The corresponding diagram would be:



We can similarly associate books and publishing companies via a *PublishedBy* relationship set. Drawing the diagram is left as an exercise to the reader.

Most of the relationships we will consider are binary — involving two entities. But ternary relationships do arise (for instance, in an organization, a *Purchased* relationship may associate a department, a part, and a supplier).

The entities participating in a relationship naturally have a role: for *AuthoredBy* relationship set, the roles could be *book* and *author*. We only really point out roles when we need to disambiguate the entities participating in a relationship set.

A relationship may have attributes (which can be used to describe particular relationship). For example, the *PublishedBy* relationships may have a *publicationDate* attribute.

Attributes

There is a lot of flexibility in the kind of attributes we can have for entities and relationships.

Attributes may be:

- *simple*: a single atomic value like an integer. These are diagramed as the normal named ovals. (It is the default.)
- *composite*: a composite value made up of parts, such as a date with a year, month, and day part. These are diagramed as a named oval for the composite attribute, linked to named ovals for each part.

Attributes may be:

- *single-valued*: a single value. These are diagramed as the normal named ovals. (It is the default.)
- *multi-valued*: a set of values. These are diagramed as double-lined named ovals.

Attributes may be:

- *base*: the value is stored in the data. These are diagramed as the normal named ovals. (It is the default.)
- *derived*: the value is computed from others and therefore need not be stored in the data (for instance, you can derive an age attribute from a date of birth attribute) These are diagramed as dashed named ovals.

Constraints

Constraints allow you to get your model to more accurately reflect the data it is meant to model.

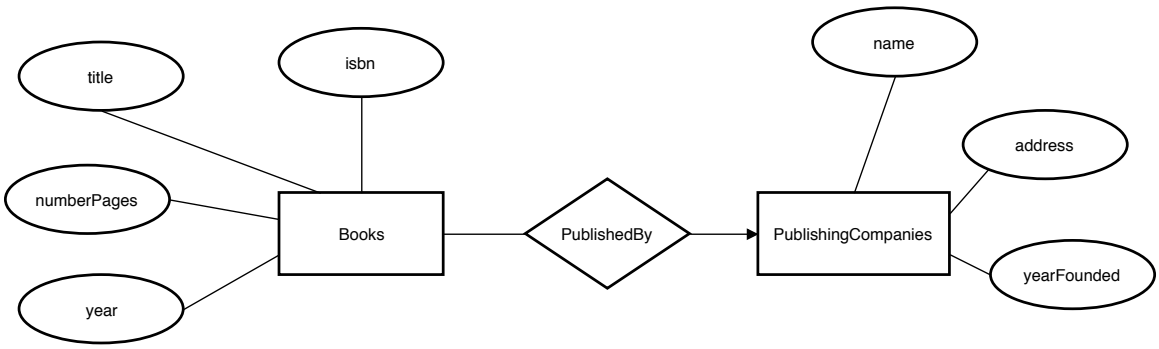
There are many kind of constraints that can be imposed. Here are the most common ones.

Mapping cardinality constraints on a relationship set R restrict how many entities can be associated with an entity via relationships in R. For example, how many authors can a book have, or how many publishing companies, or how many books a person may author.

Let's describe mapping cardinality constraints for binary relationships, since it's easiest. For more general relationships, the generalization should be straightforward. Let's assume a relationship set R between entity sets A and B.

- *One-one (1:1) mapping*: every entity in A can be associated with at most one entity in B, and every entity in B can be associated with at most one entity in A. It is diagramed by adding an arrow head to the links from the relationship diamond to each of entity sets A and B.
- *One-many (1:M) mapping*: every entity in A can be associated with zero or more entities in B, and every entity in B can be associated with at most one entity in A. It is diagramed by adding an arrow head to the link from the relationship diamond to entity set B.
- *Many-one (M:1) mapping*: every entity in A can be associated with at most one entity in B, and every entity in B can be associated with zero or more entities in A. It is diagramed by adding an arrow head to the link from the relationship diamond to entity set B.
- *Many-many (M:M) mapping*: every entity in A can be associated with zero or more entities in B, and every entity in B can be associated with zero or more entities in A. It is diagramed in the normal way, with no arrow heads. (It is the default.)

For example, if we agree that a given book may only have one publisher, but that publishers may publish many books, the corresponding mapping cardinality constraint on the *PublishedBy* relationship set would be diagramed:



Participation constraints on a relationship set R describe how many entities in an entity set E may participate in a given relationship in R.

- *Partial participation*: every entity in E may participate in zero or more relationships in R. This is diagramed in the usual way. (It is the default.)
- *Total participation*: every entity in E is required to participate in some relationship in R. This is diagramed using a double-lined link from relationship set R to entity set E.

We can generalize participation constraints into more general numerical participation constraints, which let us enforce minimum and maximum number of relationships an entity may participate in.

More advanced kinds of constraints become difficult to enforce in a logical model that implements an entity-relationship model.