CS352 Lecture - Data Models

Last revised January 18, 2017

Objectives:

1. To briefly introduce the entity-relationship model
2. To introduce the relational model.
3. To introduce relational algebra

Materials:

1. E-R Diagram example projectable

I. Introduction

A. We have seen that a database management system typically describes a database at three levels of description.

1. The physical level - how the data is stored in files
2. The conceptual level - the "big picture"
3. The view level - individual views of the database for each application

B. In order to be able to describe a database, we need some system of notation and representation - a data model. This is true at all levels; but, we are particularly concerned with description at the conceptual and view levels. We must describe two things:

1. Data objects
2. Relationships between data objects

C. Historically, there have been five major types of data model - all of which are represented by DBMS's in use today, though two are mostly in legacy systems:

1. The hierarchical model (largely obsolete, but still used in some legacy systems -we will not look at)

2. The network model (largely obsolete, but still used in some legacy systems -we will not look at)

3. The relational model (the dominant model, and the focus of much of the course)

4. Various object-oriented and object-relational models (we will look at briefly in this course)

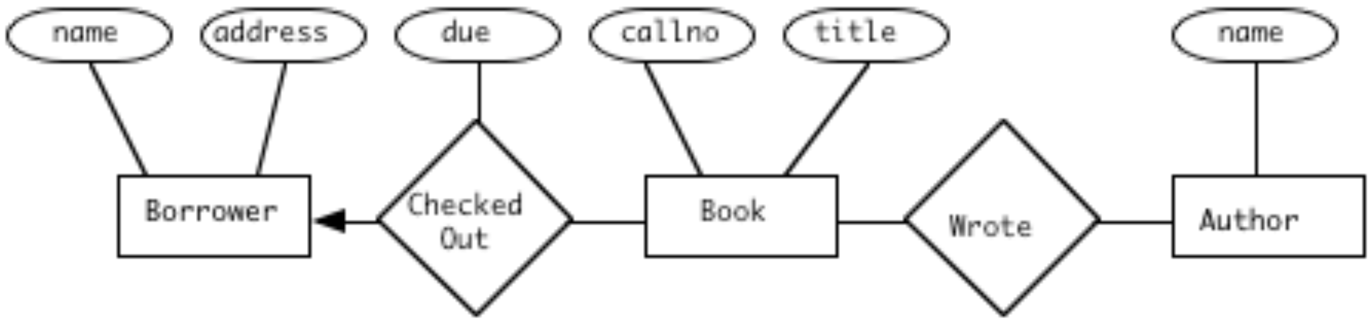
5. Various semi-structured models used in situations where performance is critical (we will look at briefly in this course)

D. There is another model, called the entity-relationship model, which is is not, per se, a basis for commercial products; but it is a very useful tool for DESIGNING databases. Also, once the E-R model is understood it gives us a language we can use in talking about the other models. Thus, we will introduce it briefly today, though we will cover it more extensively later in the course.

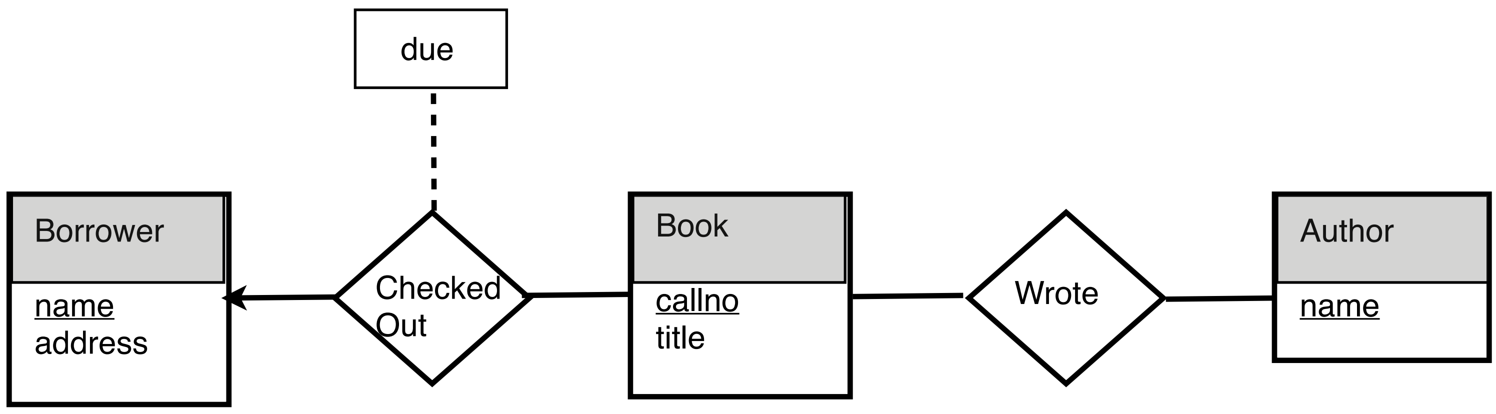
E. The textbook uses a university registration system as a source of examples. We will use a college library for our examples today.

II. The E-R Model

A. When the E-R model is used for describing a database schema, the information is generally presented in the form of an E-R diagram, like the following (very simple) example. You will note its resemblance to OO class diagrams. But the two styles of diagrams do use very distinct notation, as we shall see.



can also be drawn this way



PROJECT

B. Basic definitions

1. Entity - an entity is an object that we wish to represent information about.

   a) Example in the above: Borrower, Book, Author

   b) In an E-R diagram, an entity is represented by a rectangle.

2. Relationship - a relationship is some connection between two or more entities:

   a) Example: In the above, the “Checked Out” relationship between a borrower and a book; the “Wrote” relationship between a book and its authors.

b) Relationships can be 1:1, 1:many, or many:many.

(1) In the above, Checked out is 1 to many from Borrower to Book - one borrower can have many books checked out, but each book can only be checked out to one borrower at a time.

(2) In the above, Wrote is many to many - a given book can have multiple authors, and a given author can write multiple books.

(3) Not illustrated in the above is a 1:1 relationship

c) In an E-R diagram, a relationship is represented by a diamond, Multiplicity is represented by an arrow pointing to the "1" in a 1;1 or 1:many relationship. (Not at all the same meaning as an arrow in UML!) Review question: How does UML represent multiplicity?

ASK

Numbers or "*" on the ends of associations. (Sometimes E-R diagrams are written this way as well)

3. Attribute - Individual facts that we store concerning an entity or relationship.

a) Example: In the above, we record for a Book entity its call number and title. (In practice, we'd record a lot more as well)

b) Example: Relationships do not always have attributes, but sometimes they do.

(1) For example, for a checkout, we want to record the date due. (Note that this is a property of the relationship, not of either of the participating entities - a given borrower may have books checked out that are due on different dates, and a given book only has a date due if it is currently checked out.)

(2) However, there are no attributes for the Wrote relationship.

c) In an E-R diagram, an attribute is represented by a rounded rectangle connected to the entity or relationship it pertains to. In practice, attributes can be omitted from E-R diagrams because they make the diagram too cluttered.

C. We have noted that the E-R model is really a design tool, not the basis for actual commercial systems. One reason for this is that there is no natural physical representation for an E-R model.

1. There does exist a natural physical representation for sets of entities: a file of records, wherein each record is an entity and each field an attribute.

2. However, this is not true for relationships;

3. One of the major differences between different data models used in commercial systems is how they physically represent relationships.

   a. In the hierarchical model, relationships are modeled by <u>physical proximity.</u> An entity and the entities it is related to are stored in the same place in a physical file.

      (This works fine for 1:1 and 1:many relationships, but creates a problem for many-many relationships. The hierarchical model avoids the need to repeat entities by using what are called virtual records.)

   b. In the network model, relationships are modeled by links (pointers). A similar approach can be used in object-oriented models.

   c. Like the network model, some OO models use pointers to model relationships.

d. Like the hierarchical model, some semi-structured models also use physical proximity to model relationships. (We will look at this later.)

III. The Relational Model

A. In the relational model, both entities and relationships are modeled the same way, using relations or tables. (Two different names for the same thing)

B. An entity set is represented by a table that has one row for each entity and one column for each attribute. Typically, rows are stored in successive records in a disk file.

1. For example, the following table might be used to model borrowers (with attributes borrower_id, last_name, first_name, and address) in our library.

borrower_id		last_name	first_name	address
12345	Aardvark	Anthony	Jenks	subbasement
20174	Cat	Charlene	Frost	Basement
...				

Note: relational databases require their attributes to be atomic - hence we have separated name into last_name and first_name.

2. An important property of each relational database table is that it has a primary key - some subset of its attributes which serve to distinguish one entity from all others. (In our example, we have added a borrower_id attribute to serve this role, assuming no two borrowers can have the same id, but we can't guarantee that names will be unique.)

C. A relationship set is represented by a table which has one row for each relationship. It has one or more columns holding the primary key of each of the entities it relates, plus additional columns for any attributes of its own, if there are any.

1. For example, the following table might be used to model the CheckedOut relationship, assuming the primary key of Borrower is borrower_id and the primary key of Book is callno

   ```
   borrower_id callno      date_due

   12345       QA76.91     09-01-12
   12345       QA76.92     09-01-12
   12345       QA76.93     09-01-12
   ```

   ...

2. This simple structure has some profound efficiency implications, though, especially when compared to the other models. For example, consider the question "what are the titles of the books that Anthony Aardvark has checked out?"

   a) To answer this in a relational database.

      (1) We first find the record corresponding to the row for Anthony Aardvark in the file holding the Borrower table to find out what his id is.

      (2) Then we find the records in the file holding the CheckedOut table for the rows that contain this value as the borrower_id.

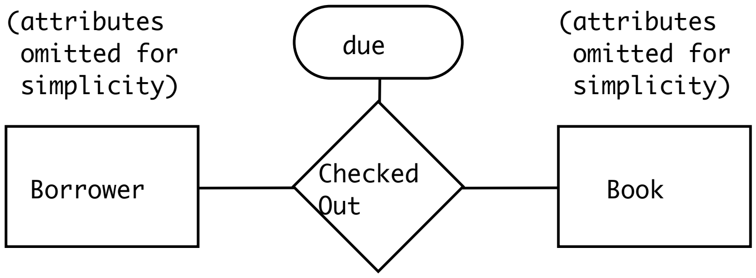
      (3) For each such book, we then find the records in the file holding row the Book table for the rows that have the corresponding callno values, in order to get the titles.

   b) By way of contrast, with the hierarchical model, we still have to find the right record for Anthony Aardvark, but then the book records come physically right after it in on disk.

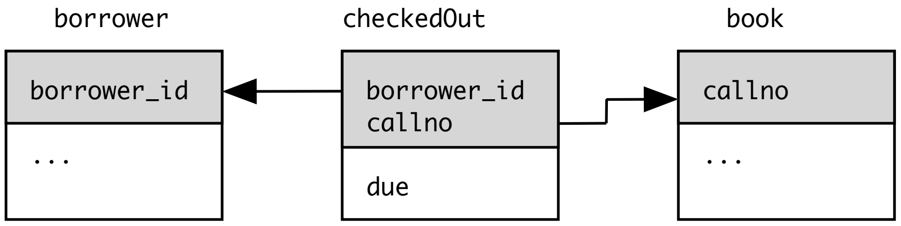
   c) With the network model, we still have to find the right record for Anthony Aardvark. But this record now holds a pointer to the first Book record for books he has out, which in turn holds a pointer to the second record ...

3. Historically, the amount of searching needed to locate desired records in a relational database was one of the main reason why the older models persisted. Early relational databases often exhibited much poorer performance than network or hierarchical ones.

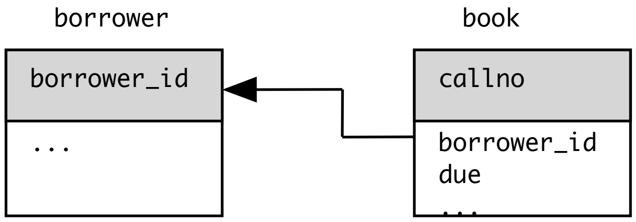
   a) One way to minimize this is to store 1:1 and 1:many relationships in the table corresponding to the "1" entity rather than in a separate table - e.g. instead of representing


(attributes omitted for simplicity)
due
(attributes omitted for simplicity)
Borrower
Checked Out
Book

by



represent it by



by taking advantage of the fact that a given book can only be checked out to a single borrower at a time.

(1) Of course, if the relationship is optional (as is the case with a book being checked out), this requires the use of a null value for an entity that is not in the relationship (for the foreign key, even if there are no attributes for the relationship)

(2) In the case where the relationship is 1:1, it might be tempting to store foreign keys in *both* entity tables - but this is problematic:

(a) Consistency - must update two tables if relationship changes.

(b) DBMS enforcement of foreign key constraints on insert (we will come to this later)

b) A lot of work has gone into indexing strategies to produce major improvements in the performance of relational systems, to the point where they dominate the database world except in certain applications , typically involving massive databases, where performance is a significant issue.

4. While modern relational database systems generally give very good performance, for extremely large datasets efficiency considerations have dictated a return to approaches similar to those used in the hierarchical and network models - but only in situations where the nature of the data is appropriate.