CS352 Lecture - Data Models

Last revised July 24, 2008

Objectives:

1. To introduce the entity-relationship model
2. To note the distinctive features of the hierarchical and network models.
3. To introduce the relational model.

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. In this course will be talking about seven different data models that can be used in designing and implementing databases (listed here in the order we will cover them):

1. The entity-relationship model
2. The hierarchical model (largely obsolete - will only be looked at briefly)
3. The network model (largely obsolete - will only be looked at briefly)
4. The relational model
5. Various OO models
6. The Object-Relational Model
7. XML

Five have been and are used as bases for commercial DBMS's. These we will cover in roughly their historical order of development.

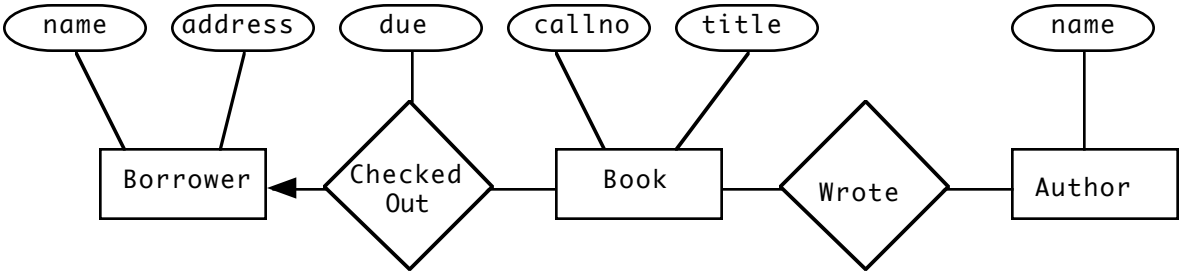
D. The entity-relationship model is not, per se, a basis for commercial products; but it is a very useful tool for DESIGNING databases. Also, once the E-R model is understood it gives us a language we can use in talking about the other models. Thus, we will introduce it briefly today, though we will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

E. We will only introduce three of the "commercial" models today. We will discuss the OO, Object-Relational, and XML models later in the course.

F. The textbook uses a banking enterprise 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:



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.

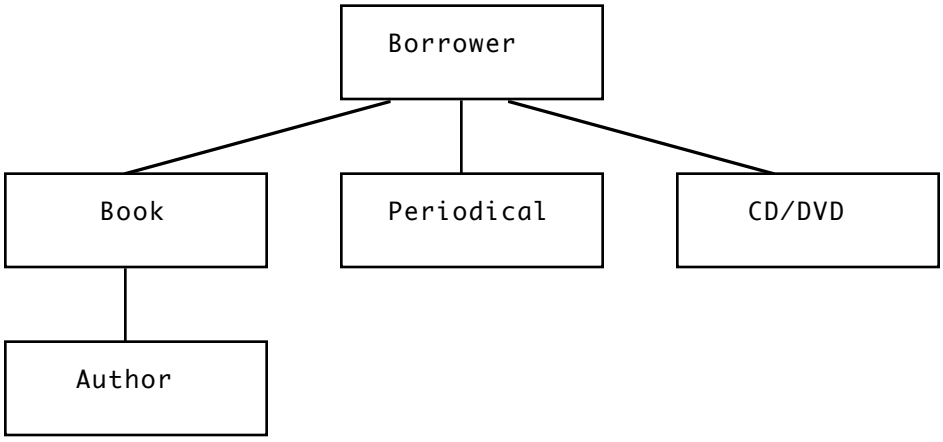
2. However, this is not true for relationships;

3. One of the major differences between different data models used in commercial systems is how they physcially represent relationships. We will note this briefly as we discuss each model.

III. The Hierarchical Model

A. Historically, the hierarchical model is the oldest model used for commercial DBMSs. IBM's Hierarchical Database Product - IMS (Information Management System) is over 40 years old (developed in 1966) - but IBM continues to develop and sell it today. (How many other software products can make that sort of claim?)

B. The hierarchical model gets its name from the fact that the structure of the information is modelled as a tree. For example, we might model our library example as follows. (I've added a few nodes lest the "tree" degenerate into a list!)



C. Generally, multiple hierarchies are needed to model a given reality. For example, if we also kept information about employees, we would almost certainly use a separate tree for this. (It makes no sense to have borrowers "own" employees or vice versa!)

D. Not only is the structure of the data modelled as a hierarchy, but the actual data itself is also represented by trees

1. Entities are stored as physical records in the database file - i.e. a sequence of locations holding the variious fields.

2. Because the data is tree-structured, physical proximity can be used to model relationships. Basically, a "parent" record is immediately followed by each of the "child" records it owns.

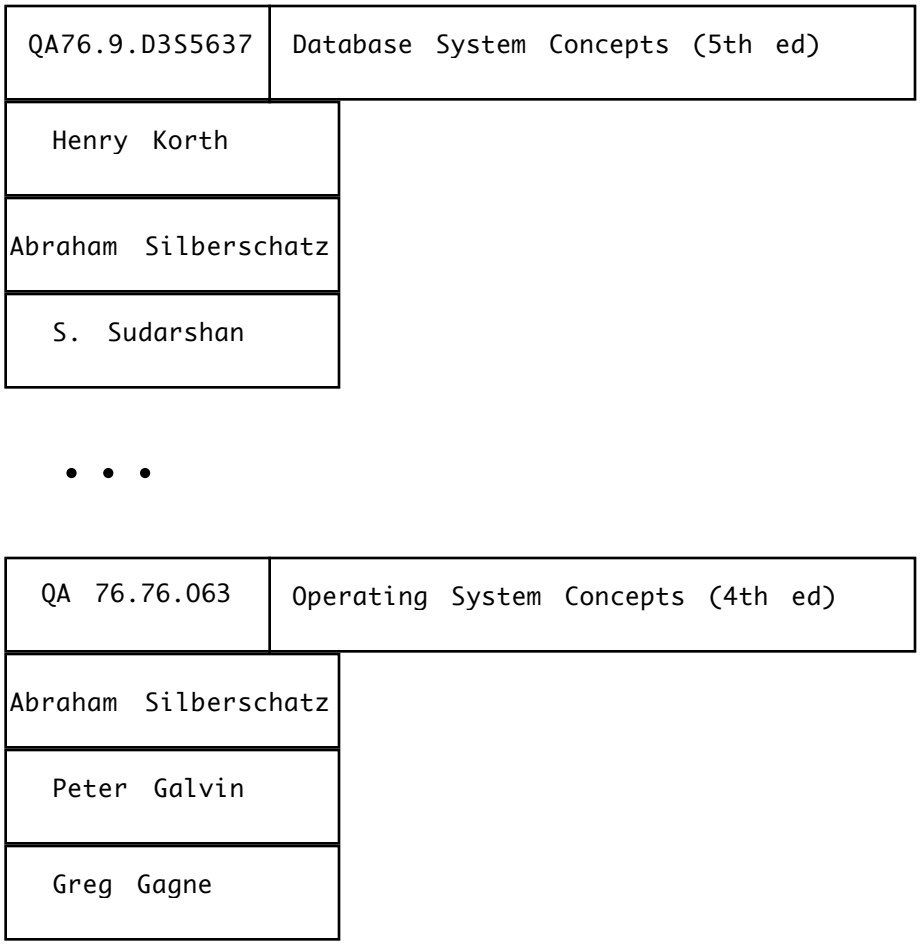
3. Example: If borrower Aardvark has books QA76.91, QA76.92 and QA76.93 checked out, this might be modelled by physically storing the data like this:

Aardvark	Jenks Subbasement	
QA76.91	Fun with Databases	9/1/08
QA76.92	Databases for Fun and Profit	9/1/08
QA76.93	More Database Stuff	9/1/08

   a) Note how the attribute of the relationship has been folded into the Book entity, which works because a given book can only be checked out to one borrower at a time - the relationship is 1:many.

   b) What do we do about books that are not checked out? One solution is to create a dummy borrower that "owns" all of the books that are on the shelf.

4. This poses a problem, of course, for many:many relationships. For example, our textbook has three authors. Both Korth and Silberschatz are the authors of multiple books. One solution used in a case like this to replicate records - a record for an author could be stored with the record for each of the books he has written, etc.



The data redudancy this creates can be lessened by using *virtual records* - which are very small records that point to the real record stored somewhere else.

E. What if the data doesn't naturally lend itself to being structured as a set of trees? (Real situations of any complexity rarely do). In this case, the information is structured as a forest of hierarchies - a primary hierarchy, and some number of secondary hierarchies.

F. Although the hierarchical model is still being used today, we will not study it further here. (You can read more about it in Appendix B of the book if you wish)

IV. The Network Model

A. The network model was historically the successor to the hierarchical model In many ways it resembles the hierarchical model.

1. Indeed, the key difference is the removal of the requirement that the database scheme be hierarchical - the network model supports a scheme that is an arbitrary graph structure.

2. The network model was standardized as what came to be known as the Codasyl DBTG model, which was developed by the Database Task Group (DBTG) or the Conference on Data Systems Languages (Codasyl) (whew!)

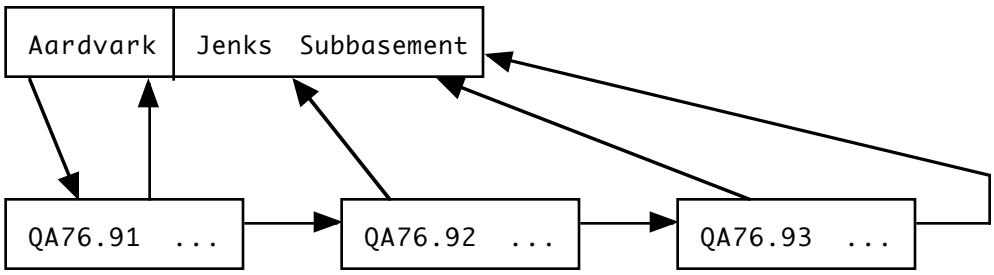
B. As in the hierarchical model, entities are stored as physical records in the database file - i.e. a sequence of locations holding the variious fields.

C. However, relationships are modelled by <u>links</u> - a pointer in one record to the physical location of another record.

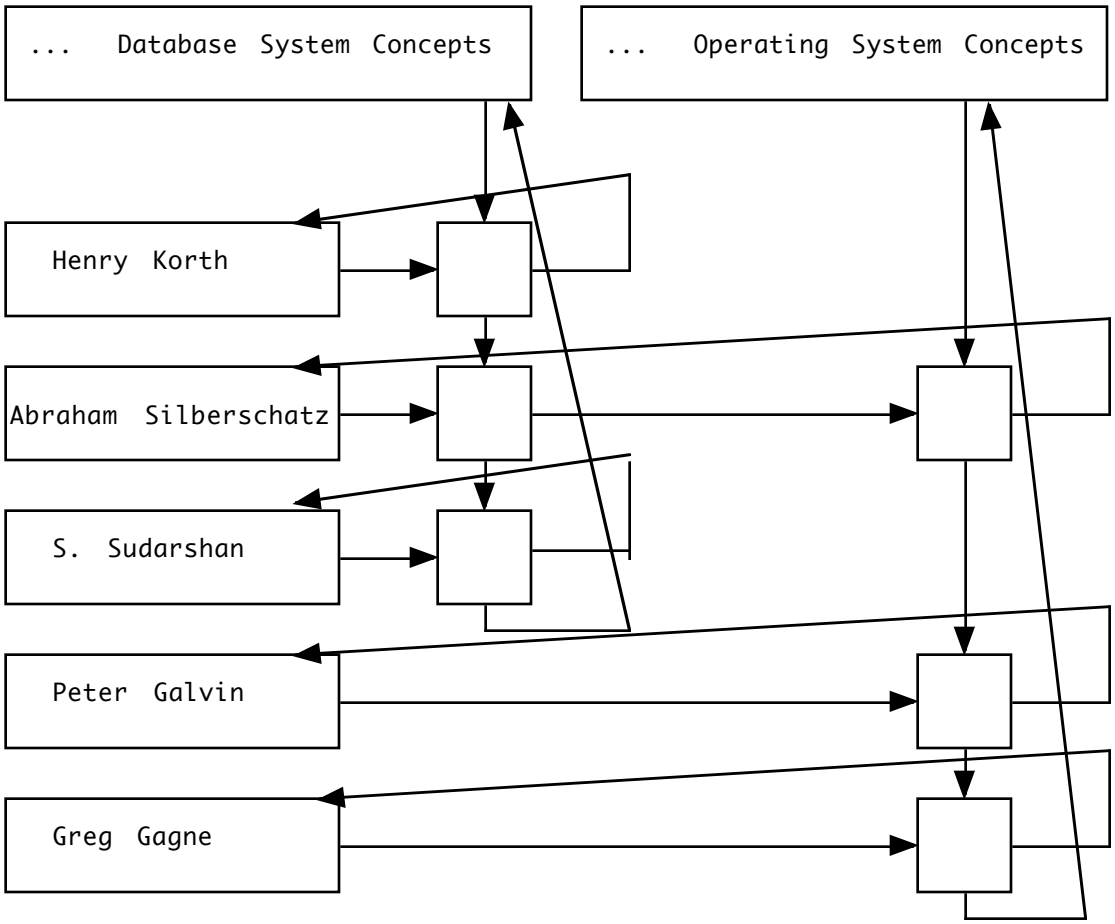
1. For example, a 1:1 relationship could be modelled by each entity record holding a pointer containing the physical location of the record for the entity it is related to.

2. What about 1:many relationships? In the Codasyl model, these are handled by having the record for the "1" type hold a pointer to the first record of the "many" type, which in turn holds a pointer to the next record ..., with the last record generally holding a pointer back to the "1". The resulting circularly-linked list is commonly called an "owner-coupled set". Of course, in addition each record of the "many" type can hold a single pointer back to the record of the "1" type is it related to.

   Example:



3. What about a many to many relationship? In this case, each instance of the relationship is represented by a special kind of record called a "link record", which participates in two sets - one associated with each record. Example:



While this is quite cumbersome to draw, it actually leads to fairly fast access to the information, because "pointer chasing" is fast.

D. Again, we will not discuss the network model further in this course, though you can study it further on your own using Appendix A of the book or various online sources.

While the hierarchical model seems to have maintained a niche in the marketplace (through products like IBM's IMS) the same does not seem to be as true for the network model. (If you do a Google search on Network Database, you'll get lots of hits for relational DBMS's holding information about various networks, as opposed to network model databases!)

V. The Relational Model

A. In the relational model, both entities and relationships are modelled 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     09-01-08
12345          QA76.92     09-01-08
12345          QA76.93     09-01-08
…
```

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.

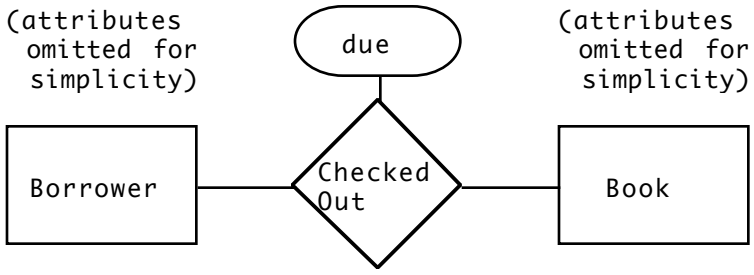
      (3) For each such book, we then find the records in the file holding row the Book table for the rows that have the corresponding callno values, in order to get the titles.

   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.

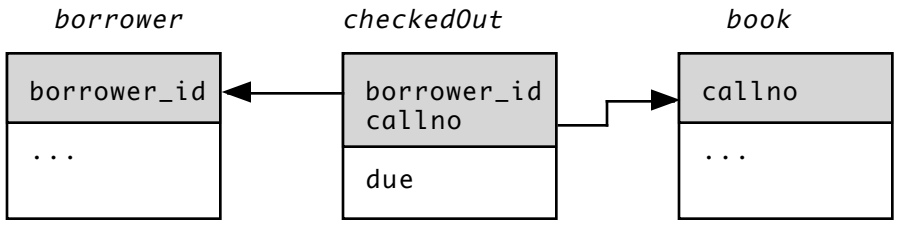
   c) With the network model, we still have to find the right record for Anthony Aardvark. But this record now holds a pointer to the first Book record for books he has out, which in turn holds a pointer to the second record ...

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 (and still do). Early relational databases often exhibited much poorer perfromance than network or hierarchical ones.

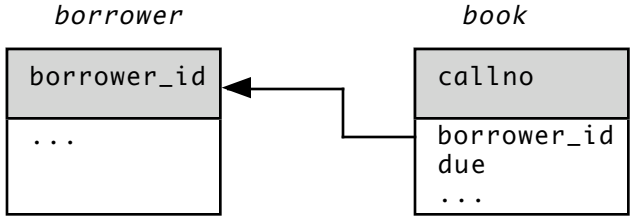
   a) One way to minimize this is to store 1:1 and 1:many relationships in the table corresponding to the "1" entity rather than in a separate table - e.g. instead of representing



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.

c) Interestingly, some of the OO database approaches have gone back to using pointers to model relationships, rather than keys - though such OO models have not attracted significant interest.

VI. Summary

A. Regardless of what basic data model one uses, one must be able to represent two kinds of things:

1. Entities - real or abstact “things”

2. Relationships between entities

B. All actual systems represent entities in a basically similar way - as records in a file. A key place where they differ is in how they represent relationships.

1. The hierarchical model represents relationships by physical proximity. It makes use of a controlled form of redundancy or - more often - virtual records to handle many:many relationships.

2. The network model represents relationships by sets that are based on pointers - in which one record holds a reference to the physical location of another record with which it is related.

   In the case of many:many relationships, it makes use of a special kind of record - called a link record - which in effect turns a relationship into an entity.

3. The relational model does not draw a distinction between how it represents entities and how it represents relationships. Both are represented using tables. A record refers to another record by means of a key.