

CPS352 Lecture -The Entity-Relationship Model last revised January 4, 2019

Objectives:

1. To discuss using an ER model to think about a database at the conceptual design level.
2. To show how to convert an ER design to a relational scheme

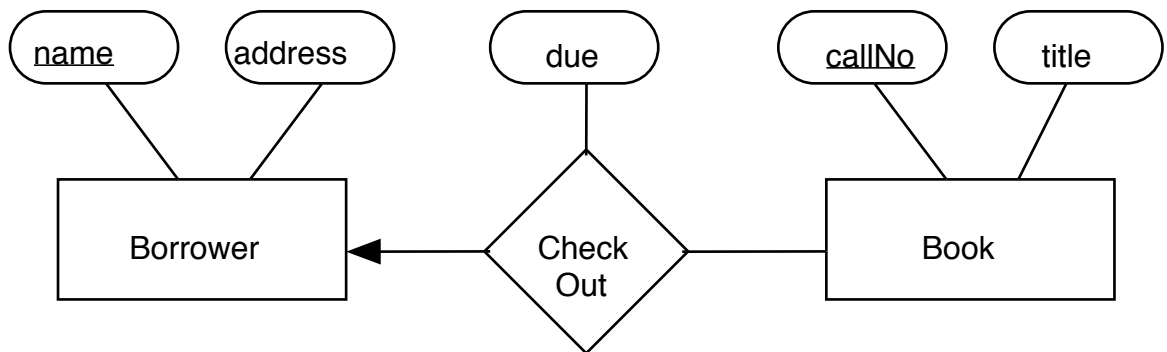
Materials:

1. Projectable of various ER diagram notations.
2. Projectable of Borrower ER diagram showing various kinds of attributes
3. Projectable of ternary relationship
4. Projectable of recursive relationship
5. Projectable of Cartesian join
6. Projectable of total participation constraint
7. Projectable of weak entity
8. Projectable of representations of generalization and specialization
9. Projectable of relational table equivalents of various ER elements

I. Introduction

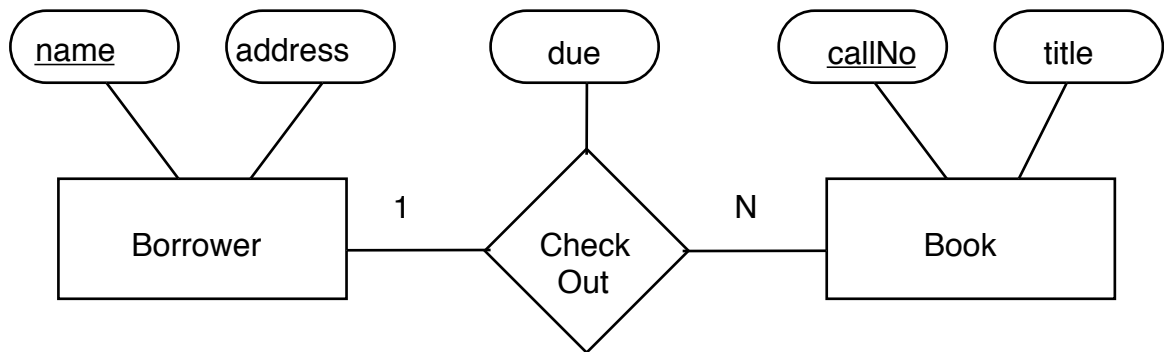
- A. In the last series of lectures, we introduced the entity-relationship model as one way of describing a database. We now return to it in more detail. We noted that the entity-relationship model is not, per se, a basis for commercial products; but it is a very useful tool for DESIGNING databases. In particular, we will focus on learning how to picture the conceptual level design of a database using Entity-Relationship (E-R) diagrams.
- B. We will also look at how to convert a conceptual design - developed as an ER diagram - into a logical design represented as a relational scheme.
- C. An ER diagram consists of two basic things - entities and relationships.
- D. However, there are several different notation systems that are used for representing these in ER diagrams. We can illustrate this by showing several different ways of drawing a diagram representing a simplified version of a portion of the library database we used in the previous lecture.

1. The original notation scheme - called "Chen" notation after its inventor - Peter Chen. This is the notation most commonly used in textbooks for a course like this - including earlier editions of the book we are using. The arrow connecting the CheckOut relationship to borrower indicates that a given book may only be checked out to one borrower at a time - i.e. given a book you can ask for the one and only borrower (if any) to whom it is checked out. We call this a one to many relationship.



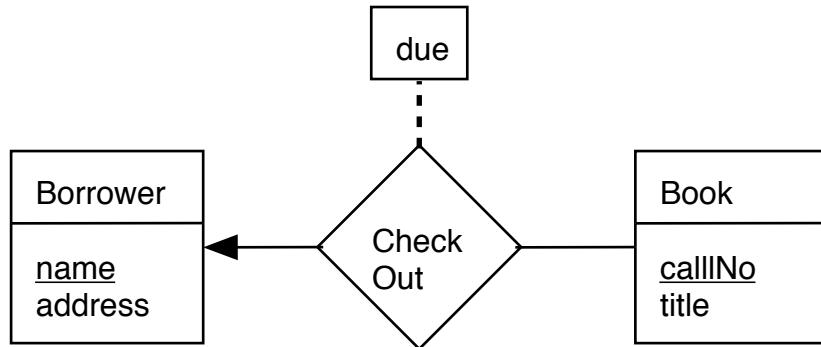
PROJECT

or, using a different representation for multiplicity of relationships, likewise indicating that a borrower can have any number of books out, but a given book can only be checked out to one borrower at a time. (Sometimes * is used in place of N - or M for a many-many relationship).



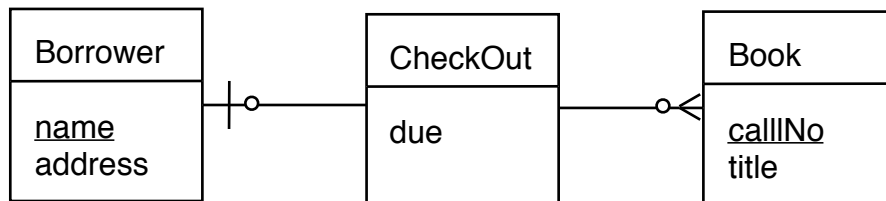
PROJECT

2. A variant of Chen notation used in our textbook. The attributes are listed in the boxes for entities and relationships "UML--style" rather than in bubbles, which is more compact but does not allow for certain features we will discuss



PROJECT

3. A system known as "crows foot" notation. Note the symbols used to denote "one" and "many". The small circles indicate that the relationship is optional on both sides - a book does not need to be checked out, and a borrower does not need to have any books checked out.



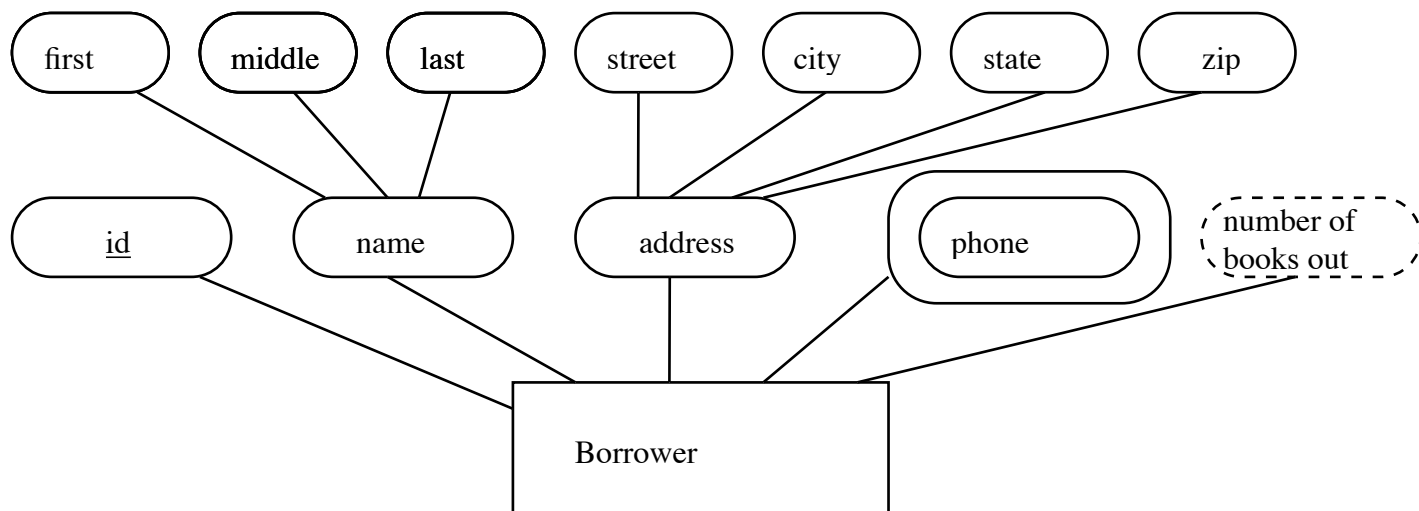
PROJECT

4. For the rest of this lecture, we will stick with the original form of the the Chen notation, and this will be the style you should use in the design project and homework (hand-drawn if you don't have an appropriate drawing tool to use.)

II. Expansion of Definitions

A. Entities and related concepts

1. Entity - an entity is an object that we wish to represent information about.
2. Entity Set - an entity set is the set of all objects of a given kind
3. Attributes - Individual facts that we store concerning an entity. To illustrate the various possibilities, we will use the following diagram:



PROJECT

- a. Often, the attributes are simple, atomic, single values; but sometimes they may not be. An attribute may be COMPOSITE - i.e. it may have internal structure

Example: The example shows that the "name" attribute is composed of a first, middle, and last name; the "address" attribute is composed of a street, a city, state, and ZIP code. Note how this is shown in a Chen diagram.

Note: we have seen that the relational model requires attributes to be atomic. But in doing conceptual design, it may be helpful to think in terms of some composite attributes, even though they will later need to be converted to an atomic form.

- b. For a given entity, a given attribute normally has a single value, but sometimes an attribute needs to be MULTIVALUED

Example: A person may have multiple phone numbers. Again, note how this is shown in a Chen diagram.

Note: the relational model requires attributes to be single-valued. But in doing conceptual design, it may be helpful to think in terms of multivalued attributes even though they will later need to be converted to a single-valued form.

- c. Sometimes, a given attribute can be calculated from other information in the database - in which case, instead of storing it we may compute it upon demand. Such an attribute is called a DERIVED attribute. Note how this is shown in a Chen diagram.

Example: As we shall see, the number of books a given borrower currently has checked out can be computed by counting whenever the value is needed; so we might include books-out as a derived attribute of the borrower entity, though no value for this stored explicitly in the database.

- d. Sometimes, we will not know the value of a particular attribute for a particular entity, or it somehow does not apply in a particular case - in which case the value of that attribute is said to be NULL. (There is no special diagrammatic notation for this, since it is just a value for particular entity).

- 4. Domain - the set of possible values for each attribute of an entity is called the domain of that attribute
- 5. Keys: Since the members of a set must be distinct, for any entity set, there must be a set of attributes that serve to uniquely distinguish one entity from all others in the set. Recall our discussion of these in connection with the relational model.

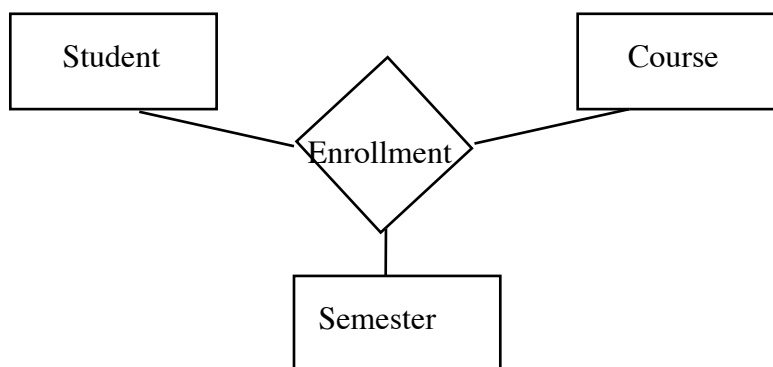
- a. Such an attribute or set of attributes is called a SUPERKEY.
Typically, an entity set has many superkeys.
- b. A superkey that has no proper subset that is also a superkey is called a CANDIDATE KEY.
- c. The candidate key chosen by the designer to uniquely identify entities in an entity set is called the PRIMARY KEY. (This is the origin of the term "candidate key" - it is a key that is a candidate for primary key.)
The primary key attribute(s) for an entity are underlined, as is the case for id in Borrower.
- d. Some kinds of entities - called weak entities - don't have any primary key. We will look at an example and how this is handled below.

B. Relationships and related concepts

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

- a. Relationships can be binary, ternary, quaternary etc - i.e. they can involve two, three, four or more entities. However, binary relationships (such as CheckOut) are by far the most common.

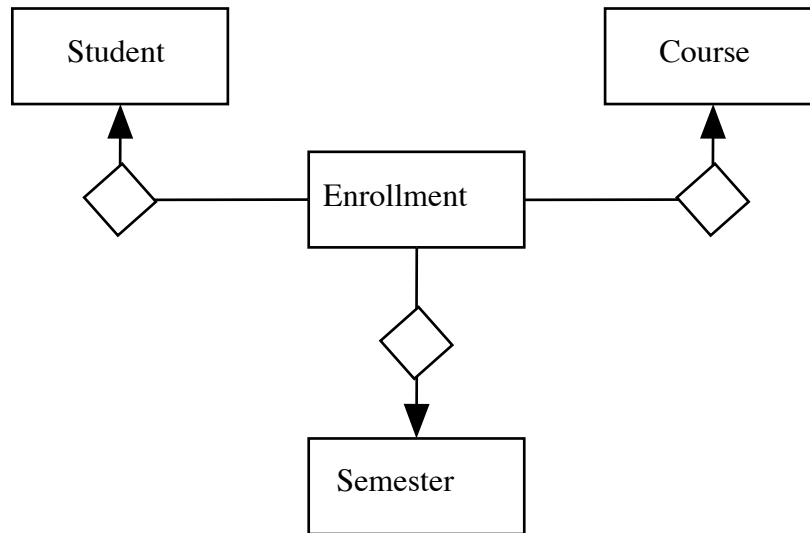
(1)Example of a ternary relationship: student enrollment in a course could be thought of as a ternary relationship involving the student, the course, and the semester.



PROJECT

A relationship set involving n entities ($n > 2$) can always be converted to a single entity set plus n binary relationships. The original relationship is converted to an entity, connected to each of the other entities through the appropriate set.

Example: Alternate approach to representing the above.



PROJECT

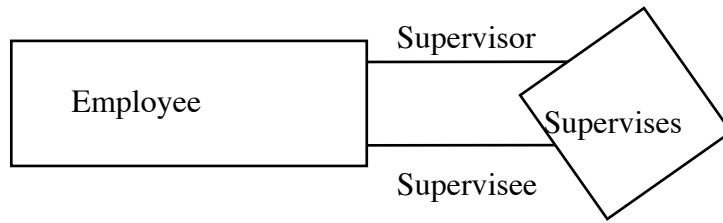
(Note that each of the new binary relationships is “one” on the side connecting to the original entity. Each of the new enrollment entities is associated with exactly one Student, Course, and Semester)

(2) Because a conversion like this is always possible, it is actually rare to see ternary and higher degree relationships in ER diagrams.

- b. Normally, the entities in a relationship come from distinct entity sets. Sometimes, though, we can have a relationship in which both entities are from the same entity set. Then, we need to distinguish the role each entity plays in the relationship.

Example: consider the relationship "supervised by" between employees of the library. Both entities in the relationship are from the same entity set (employees), but there are two distinct roles: supervisor

and supervisee. In cases such as this, it is common to explicitly include the role names in an ER diagram.



PROJECT

(Note: as drawn we allow an employee to have more than one supervisor, and a supervisor to supervise more than one employee. If an employee had only one supervisor, we would use an arrow on the Supervisor side of the relationship)

- 2. Relationship set - a relationship set is the set of all relationships of a given type - just as an entity set is the set of all entities of a given type.

Example: When a borrower checks out a book, that establishes a relationship between the borrower and the book. The set of all such relationships, together, constitutes the CheckOut relationship set.

- 3. Formally, a relationship set is a subset of the cartesian product of the entity sets. The cartesian product is formed by taking every possible combination of elements from the sets.

Example: Suppose our sets of borrowers and check outs were as follows (this time using just names and titles as the keys for simplicity):

Borrower	Book
Anthony Aardvark	Herbs and Shrubs
Ralph Raccoon	Popular Trees
Zelda Zebra	Zoo Guidebook

The Cartesian product of these sets is

Anthony Aardvark,	Herbs and Shrubs
Anthony Aardvark,	Popular Trees
Anthony Aardvark,	Zoo Guidebook
Ralph Raccoon,	Herbs and Shrubs
Ralph Raccoon,	Popular Trees
Ralph Raccoon,	Zoo Guidebook
Zelda Zebra,	Herbs and Shrubs
Zelda Zebra,	Popular Trees
Zelda Zebra,	Zoo Guidebook

PROJECT

This set represents the set of ALL POSSIBLE RELATIONSHIPS that could ever occur; but obviously the CheckOut relationship set at any one time forms a subset of this set. (Indeed, it could be empty, and - in this particular case - could never have more than three elements since only one person can check out a given book at any one time.)

- a. Thus, the join operation of relational algebra - which forms the cartesian product - can be thought of as creating an entity set which includes every possible relationship.
 - b. Likewise, the natural join operation can be thought of as creating an actual instance of the relationship set corresponding to the current instances of the entity set.
4. There exists a natural physical representation for entity sets: a file of records, wherein each record is an entity and each field an attribute.
 5. However, this is not true for relationships; one of the major differences between different types of DBMS's is how they represent relationships.
 - a. Hierarchical DBMS's often use physical proximity or pointers to model relationships.

- b. Network DBMS's often use circularly linked lists for relationships
 - c. OO Databases may use references or pointers to model relationships.
 - d. As we have seen, relational DBMS's do not distinguish between entities and relationships; a relationship is, in effect, just another entity. Thus, the same physical representation is used for both. Note that the handling of relationships represents a key difference between relational databases and the other kinds of systems.
6. Like entities, relationships can have attributes (called descriptive attributes) associated with them.

Note that we want to associate an attribute with a relationship just when it is a property of the relationship, not of the participating entities - e.g. in the case of date due for a book:

- a) It makes no sense to say that “date due” is a property of a borrower. A borrower may have several different books out, with different dates due.
- b) A date due is not really a property of a book, in the sense that a given book may be checked out by different borrowers at different times, with different dates due.

Note how this was represented in the Chen diagram example for borrowers and books we looked at earlier.

PROJECT (Original Chen illustration used earlier)

C. Mapping Constraints - We have seen that a relationship set is a subset of the cartesian product of the entity sets it relates. Often, the logic of the data being modeled will impose some restrictions as to what kind subsets are possible.

Example: We would not expect to find the following in the CheckOut relationship:

Anthony Aardvark,
Zelda Zebra,

Herbs and Shrubs
Herbs and Shrubs

Why?

ASK

1. Mapping Cardinalities - the most common constraints are mapping cardinalities.

a. In general, a binary relationship can be

(1)One to one - the most tightly constrained

(2)One to many or many to one

(3)Many to many - the least constrained

Example: The relationship set CheckOut is one to many from borrowers to books - a borrower can have many books checked out, but a book can only be checked out to one borrower at a time.

b. If a given relationship set is one to one, any member of either entity set involved in the relationship in question can participate in at most one relationship of the kind in question.

Example: heterosexual marriage is a one to one relationship between the entity sets men and women

Note that a one to one mapping constraint for a given relationship set does not imply that a given entity **MUST** participate in a relationship of that kind - only that it **CAN** participate in at most one such relationship. (The relationship marriage allows for bachelors and bachelorettes, widows and widowers etc.)

c. If a relationship set is one to many, then any entity in the first entity set can participate in any number of relationships, but an entity in the second set can participate in at most one.

Example: as we have already noted, the relationship CheckOut is one to many from borrowers to books.

- d. Of course a one to many relationship from A to B is a many to one relationship from B to A - i.e. there is no fundamental difference between the two concepts - its just a matter of how we refer to it.
- e. If a relationship set is many to many, then any entity of either set can participate in any number of relationships.

Example: Suppose we defined the relationship "has taken out" between borrowers and books, which records every book a given borrower has ever taken out (whether or not he has it out now.) (This is actually a very bad idea on privacy grounds, of course - but allows us to illustrate a point about database design!) This is many to many, since:

- Any given borrower can take out any number of books
 - Over time, any given book can be checked out by any number of borrowers
- f. Note that one-to-one and one-to-many / many-to-one relationships are most common. In fact, the earlier hierarchical and network models all had some difficulty with many-to-many relationships, and the relational model can often use a much more efficient representation for one-to-one and one-to-many relationships than what must be used for many-to-many relationship as we shall see.
 - g. Primary keys for relationship sets - just as any strong entity set has a primary key, so does any relationship set. The primary key of a relationship set depends on its cardinality.

(1)If it is many-to-many, then its primary key is simply the union of the key attributes of the entities it relates.

Example: Suppose we want to record all borrowers who have ever checked out a given book. Since the primary key for borrowers is borrower-id, and that for books is call number, the primary key for has ever checked out is the two attributes borrower id and call number together.

Note: Since this is a relationship set, we will record the fact that a given borrower has checked out a given book just once in it, even if the same borrower takes out the same book again. (Or we could include a date attribute in the relationship and make it part of the primary key)

(2) If it is one to many or many to one, then its primary key is the primary key of the “one” entity.

Example: A book can only be checked out to one borrower at a time, but a given borrower can check out many books - so CheckOut is many to one from books to borrowers. Suppose the primary key for books is call number. Then the primary key for CheckOut, like that for books, is call number.

(3) If it is one to one, then the primary key is the primary key of *either* of the entities.

h. Mapping cardinalities are shown in a Chen diagram, by the use of an arrowhead pointing to the "one" entity(s) - i.e.

- In a one to one relationship, arrowheads point to both entities
- In a one to many relationship, an arrowhead points to the first
- In a many to one relationship, an arrowhead points to the second
- In a many to many relationship, the diagram contains no arrowheads.

(As an example, in the case of CheckOut, think of the arrowhead as indicating that we can define a function that, given a book, returns its

borrower, if any. The lack of an arrowhead going the other way says we cannot define a function that, given a borrower, returns the book (s)he holds, since he can hold many books and by definition a function is single-valued.)

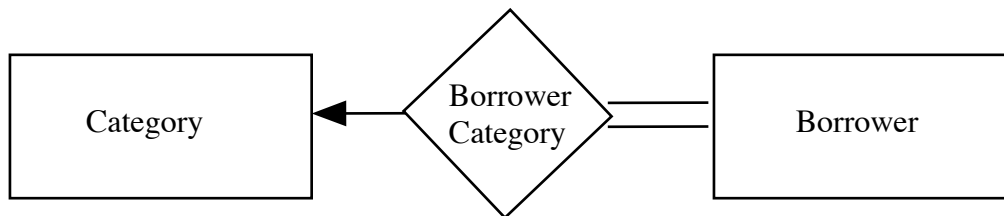
2. Participation constraints: in some cases, the underlying reality dictates that every entity in one of the entity sets must participate in an instance of the relationship.

Example: Suppose we have the notion of a borrower category, with different categories of borrowers allowed different privileges such as length of time to keep a book out. (This is, in fact, the case with the Gordon library - in fact, a faculty borrower used to be able to keep a book out for a year.)

We might want to require that every borrower be related to some category. This is called a total participation constraint (as opposed to partial participation. It

This is represented in a Chen diagram by using a double line for the connection between the relationship and the entity that must participate.

Example:



PROJECT

(Note: this diagram says each Borrower must not only participate in an instance of the relationship, but must participate exactly once - it would not make sense to have a borrower be simultaneously in two or more categories!)

3. Weak Entities and Existence dependencies - another form of constrained relationship, wherein the existence of an entity in one entity set is dependent on the existence of an entity in the other entity set.

a. We said earlier that - in almost every case - the complete set of attributes for an entity is a superkey for the entity set.

(1) In many cases, this is uninteresting, though there are some entity sets for which the full set of attributes is the only possible superkey.

(2) There is a certain kind of entity set - called a *weak* entity set - in which even the full set of attributes may not be sufficient to be a superkey. (There is no superkey for a weak entity set.) A weak entity is always subordinated to some other strong entity, on whose very existence it depends. (More about this later).

Example: In a library database, a fine can be represented as a weak entity depending on the borrower entity who owes it, if the fine entity only stores the amount.

We call the entity whose existence depends on another SUBORDINATE, and the other entity DOMINANT.

b. The key property of an existence dependency relationship is that if a dominant entity in the relationship is deleted from the database for some reason, then all subordinate entities depending on it must also be deleted

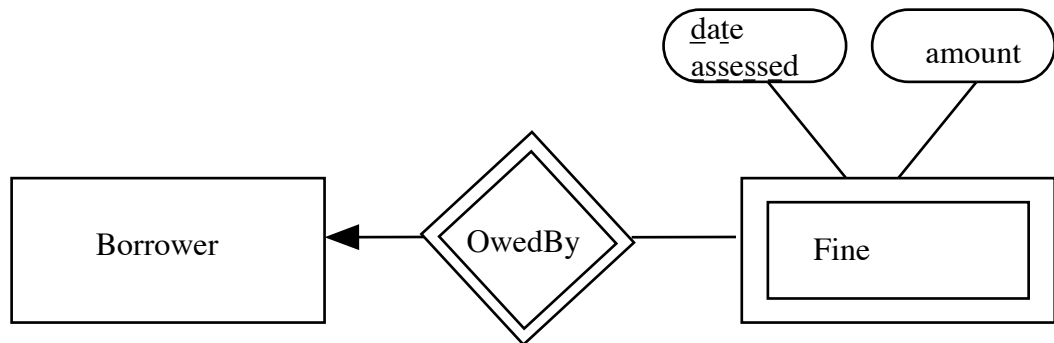
Example: if the borrower owing some fine is removed from the database, we must also remove the record of the fine owed. [An off-campus borrower who moves out of state can get away with leaving outstanding fines behind him that can never be collected.]

c. Entities within a weak entity set are uniquely identified by the combination of some or all of their own attributes plus the entity on which they depend.

Example: Suppose that the library assesses all fines once a day, lumping together into one amount all the amounts owed for all overdue books returned that day by a given borrower. Then a given fine - represented as amount due and date assessed - can be uniquely specified by specifying the date it was assessed plus the borrower who owes it.

- d. The attributes of a weak entity that identify it uniquely among all the weak entities related to a given strong entity are called its DISCRIMINATOR or PARTIAL KEY. The primary key of a weak entity consists of its partial key plus the primary key of the entity on which it depends.
- e. In an ER diagram, a weak entity is represented by a double box, and the corresponding existence-dependency relationship by a double diamond. The discriminator attributes of the weak entity are underlined using a dashed line.

Example (attributes of borrower omitted for simplicity)



PROJECT

III. Generalization and Specialization

- A. Sometimes, a given entity set may contain several distinguishable groups, with some attributes in common and some distinct to each group.

Example: The entity set borrowers may be composed of student borrowers, faculty/staff borrowers, and community borrowers. All borrowers have a name and address. Student borrowers have a student id and a class year (freshman, sophomore etc.). Faculty/staff borrowers have a faculty/staff id and a campus department they are employed by. Community borrowers may have a special id generated just for library use, plus some relationship to the college that explains why they are granted borrowing privileges here (e.g. NECCUM affiliation.)

- B. This kind of situation give rise to either generalization or specialization.

1. If every member of the general group must also be a member of exactly one special group (as was the case in the example we just talked about), we call the situation generalization - the general group (borrower) is a generalization of the particular kinds of borrower (student, faculty/staff, and community).
2. If some, but not all, the members of the general group also belong to some special group, we call the situation specialization.

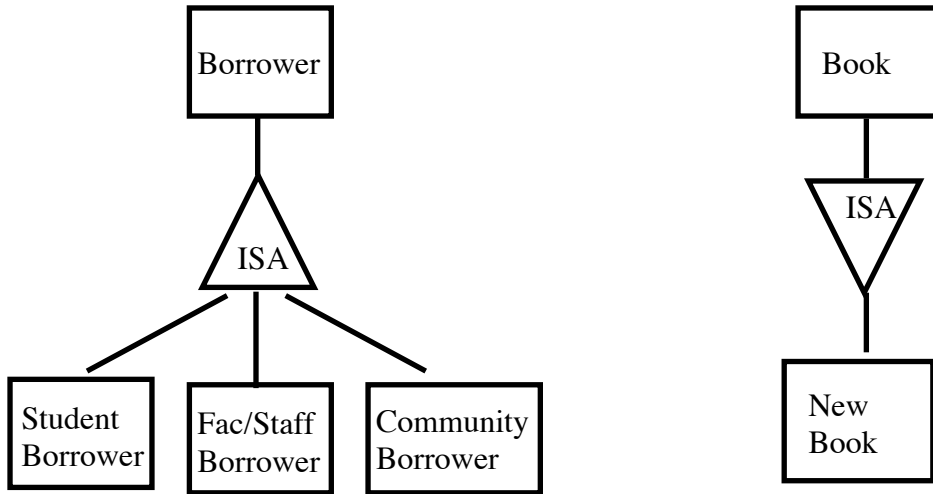
Example: Newly-received books in our library are members of a group called "new books". Such books are housed in a special place (in the lobby) and have shorter check-out periods.

3. If this sounds a lot like inheritance in OO, that's because that's really what it is! (Generalization is what gives rise to abstract classes.)

- C. The original Chen diagram notation did not have a notation for generalization and specialization. Various notational devices are used to represent these ideas:

1. Triangles as in UML. (This is what the book does)

2. Some writers use much larger triangles pointing up for generalization and down for specialization - e.g.



PROJECT

IV. Representing Entities and Relationships by Relational Tables

- A. Any database scheme consisting of entities and relationships can be represented by a series of TABLES - one for each entity set, plus one for each relationship set. These often correspond to the relations of a relational database, though in some cases some adjustments need to be made.
- B. A strong entity set can be represented by a table with one row for each entity, and one column for each attribute. When we write such a table out, we often label the columns with the names of the attributes, or an abbreviation for them.

EXAMPLE: If the entity set borrowers has attributes id, last name and first name, - it might be represented by:

<u>ID</u>	<u>lastName</u>	<u>firstName</u>
12345	Aardvark	Anthony
20174	Cat	Charlene

...

PROJECT

C. A weak entity set can be represented similarly - except that we add a column or columns containing the primary key(s) of the strong entity(s) on which the weak entity depends:

EXAMPLE: if the entity set fines - with attributes amount owed and date assessed - is a weak entity set subordinate to borrower, and the primary key of borrowers is id, then fines can be represented as follows:

<u>ID</u>	<u>amount</u>	<u>dateAssessed</u>
12345	\$0.25	12-1-18
12345	\$0.50	12-2-18
20174	\$0.10	11-15-18
...		

PROJECT

D. A relationship set can be represented by a table with one row for each relationship, and with one column for each of its own attributes, plus one column for each primary key attribute of each entity participating in the relationship.

EXAMPLE: If the CheckOut relationship has attribute due, and relates borrowers and books, and the primary key of borrowers is id, and of books is callNo then CheckOut can be represented as:

<u>ID</u>	<u>callNo</u>	<u>due</u>
89754	RZ12.905	2-10-19
89754	LM925.04	2-10-19
...		

PROJECT

E. If the relationship is one-to-one or one-to-many, an alternate representation is possible. We might “fold” the relationship into the “many” entity (by including the foreign key of the “one” entity and any attributes of the relationship into its attributes), with the understanding that these will be null for an entity that is not in any relationship.

EXAMPLE: since CheckOut is one to many from borrowers to books, it could also be represented by folding it into the book entity as follows:

callNo	title	author	ID	due
QA76.093	Wenham Zoo Guide	elephant	null	null
RZ12.905	Fire Hydrants I Have Known	dog	89754	
2-10-19				

...

PROJECT

1. This has the advantage of reducing the number of joins needed for many queries, which can be very useful since joins are computationally expensive.

Example: to find out the name of the borrower who has the book with title "Wenham Zoo Guide" out, we would need to join three tables (Book, CheckOut, and Borrower) if CheckOut is represented by a separate table, but only two if it is folded into Book.

2. However, this doesn't offer any improvement for some queries.

Example: to find out the name of the borrower who has the book with call number QA76.093 out, we would need to join two tables with either representation for CheckOut (either CheckOut and Borrower or Book and Borrower).

3. It also requires the use of null values if a given Book is not checked out.