CS344
Database Management Systems

ER modelling, Weak Entities, Class Hierarchies, Aggregation

Aug 2nd - Lecture Notes (Summary)

Submitted by -

N. Vishnu Teja
09010125

Saurabh Saxena
09010145

(Most the information has been taken from Class Notes, Silberschatz and Ramakrishnan)

ER Modelling

Recap:

Entity: A 'thing' or an 'object' that is distinguishable from all the other objects. Each entity has a set of properties. For example, each person in an enterprise is an entity with properties person-id, name, age etc.

Entity Set: Set of entities of the same type that share the same properties or attributes. For example, employees of a store. Attributes maybe single or multivalued, simple or composite, derived.

Relationship: Is an association among several entities. For example, a car is owned by a person. Relationships can be recursive. For example, a student helps a student. Both belong to the same entity set of students. Relationships can have descriptive attributes. For example, the relationship 'car owned by a person' can have attribute 'since'.

Relationship Set: Set of relationships of the same type. For example, a relationship set of 'owns' between two entities, owners and cars. Relationships can have mapping cardinality i.e. they can be one to one, many to one, one to many and many to many.

Entity sets can have full participation or partial participation in a relationship. The participation of an entity set *E* in a relationship set *R* is said to be **full** if every entity in *E* participates in at least one relationship in *R*. If only some entities in *E* participate in relationships in *R*, the participation of entity set *E* in relationship *R* is said to be ***partial***.

Keys: A key is a set of attributes whose values allow us to uniquely distinguish entities from each other.

SuperKey: *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

Candidate Key: Superkey *K* is a **candidate key** if *K* is minimal i.e. it has the least number of attributes possible.

Primary Key**:** One of the candidate keys is selected to be the **primary key.**

Relationships can involve multiple entities. For example, a professor guides a student for multiple projects.

Design of ER Diagrams

(Notations are followed from Silberschatz)

Notations:

- ***Rectangles*** Represent entity sets
- ***Ellipses*** Represent attributes
- ***Diamonds*** Represent relationship sets
- ***Lines*** Link attributes to entity sets and entity sets to relationship sets
- ***Double ellipses*** Represent multivalued attributes
- ***Dashed ellipses*** Denote derived attributes
- ***Double lines*** Indicate total participation of an entity in a relationship set
- ***Double rectangles*** Represent weak entity sets (Discussed later)

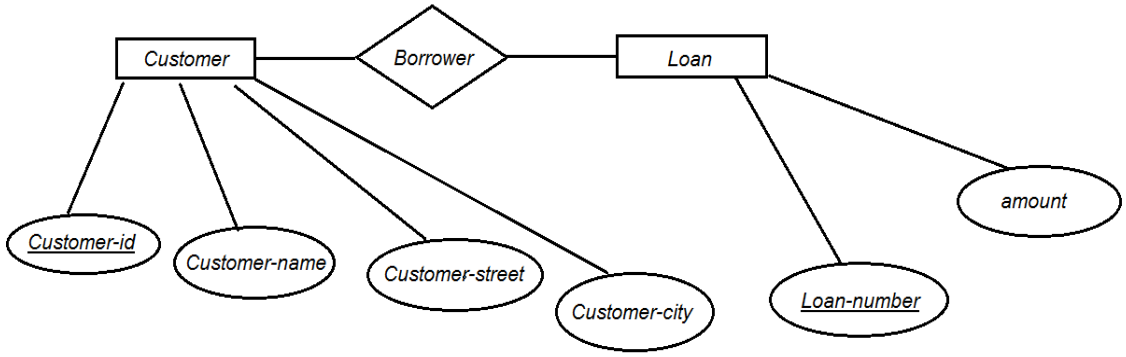
A Simple ER Diagram Construction

(Taken from Silberschatz for better understanding)

Consider an Entity relationship diagram that consists of two entities called *customer* and *loan* related through a binary relationship called *borrower*. The attributes associated with the customer are customer-id, customer-name, customer-street and customer-city. The attributes associated with loan are loan-number and amount. Attributes of an entity set that are members of primary key must be underlined.

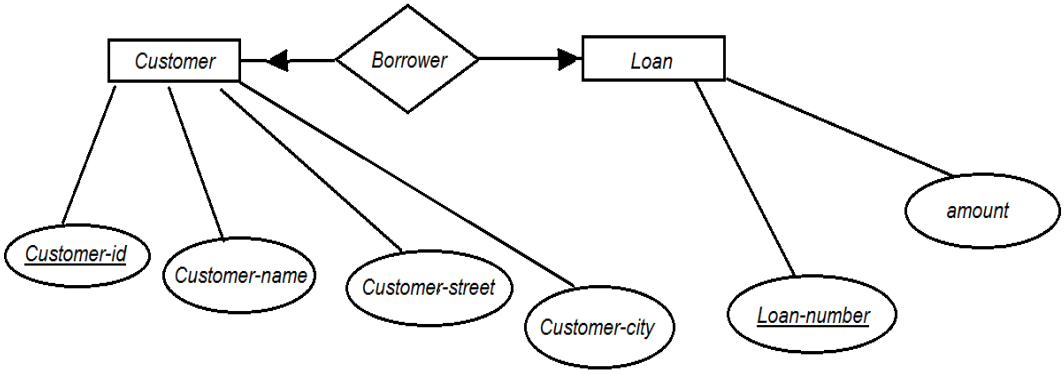
A directed line (→) from a relationship set towards an entity set also indicates a key constraint i.e. it specifies that *the relationship* is either a one-to-one or many-to-one relationship set from the first entity to the entity directed by the arrow.

An undirected line (—) from the relationship set to the entity set specifies that the relationship is either a many-to-many or one-to-many relationship set from the first entity to the directed entity.



ER Diagram corresponding to customers and loans

Here, the relationship borrower may be a many to many or a one-to-many relationship from customer to loan as it is connected without an arrow. The ‘Customer-id’ is the primary key that uniquely identifies the entities of the set ‘customer’ and the ‘Loan-number’ is the primary key that uniquely identifies the entities of the set ‘loan’.



ER diagram indicating a one to one relationship between ‘customer’ and ‘loan’ entities.

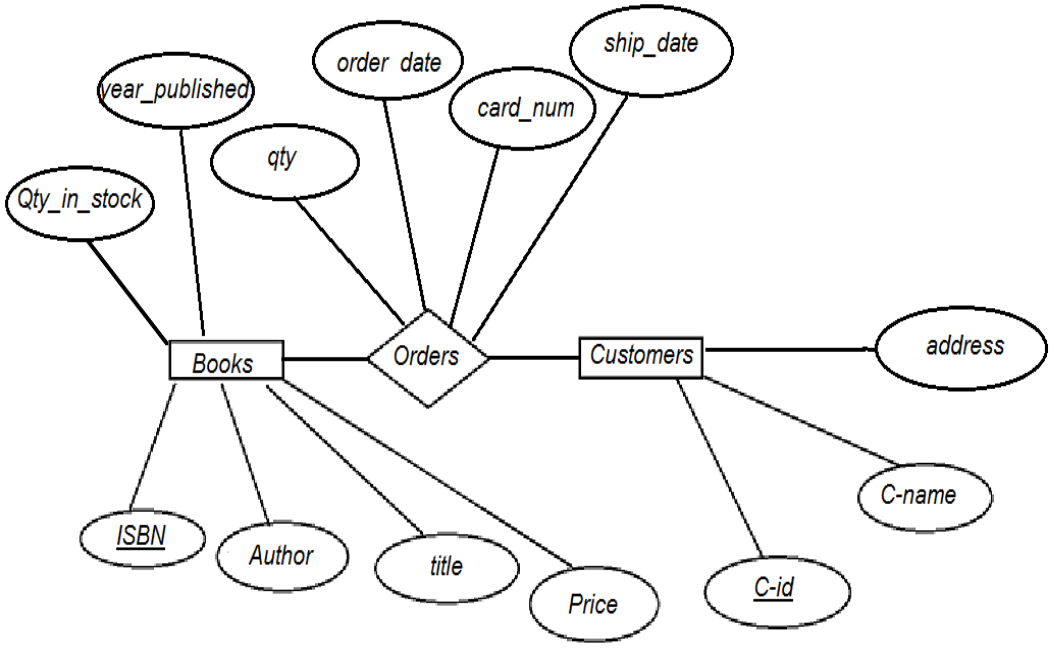
ER modeling of an online book shop

The Problem:

"I would like my customers to be able to browse my catalog of books and place orders over the internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the ISBN number of a book and quantity; they often pay by credit card. I then prepare a shipment that contains the books they ordered. If I don't have enough copies in stock, I order additional copies from the publisher and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its ISBN number, title, author, purchase price, sales price, and the year the book was published. Most of my customers are regulars, and I have records with their names and addresses. New customers have to call me first and establish an account before they can use my website.

On my new website, customers should first identify themselves by their unique customer identification number. Then they should be able to browse and to place orders online. "

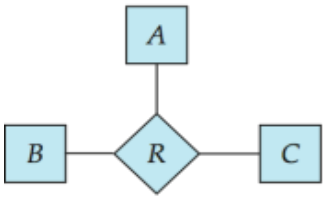
Possible Solution:



The *ISBN* code identifies each book uniquely and is hence the primary key of the book entity set. Same is the case with the *C-id* attribute of the customers entity set. *Card_num* has been placed as an attribute of the order because a customer may have more than one card number. There are no directed lines as one customer can order any number of books and one book may be ordered by any number of customers. Since *qty*, *order_date*, *ship_date* are specific to the order, they are listed as attributes of the relationship.

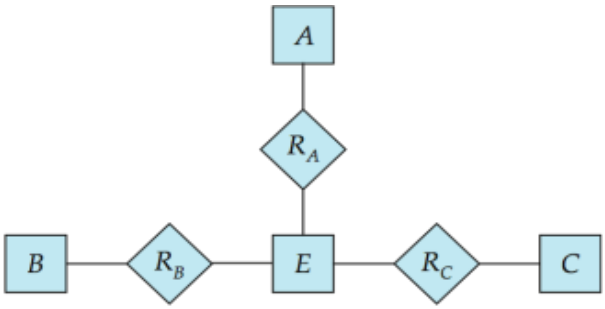
N-ary Vs Binary relationships

N-ary relationships are usually converted to binary relationships for the sake of implementation ease. This can usually be done by the addition of new entity sets.



(a)

N-ary Relationship



(b)

Binary Relationship (New entity E)

N-ary relationships can have only one outgoing arrow.

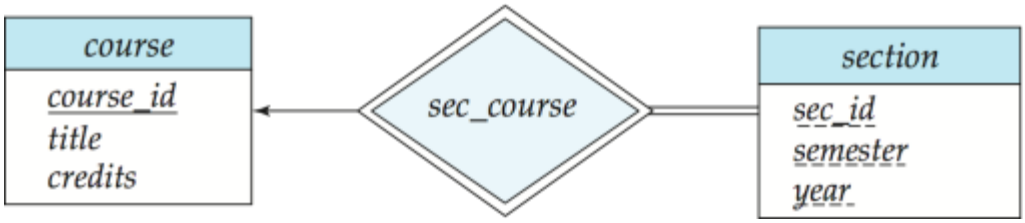
Consider an n-ary relationship among entities student, department and course. The most plausible way is to have the arrow directed toward the student. This is because both the department and the course can have multiple students. A department can have multiple courses etc. A student can only belong to one particular department.

Weak Entities

Weak entities depend on other strong entity sets to generate its Primary Key.

A very good example of this is the entity set of dependants of an employee of a bank. The bank may want to store the dependants information for various reasons such as policies etc. In this case the dependants are not uniquely identified by their attributes (name, age, gender (most likely)); But together with the employee-id of the employee they can be distinguished. In this case the dependants are the weak entity set, the employees are the 'Identifying Owner' and the policy relationship is called the 'Identifying relationship'. Note that the weak entity must always have full participation and also each entity in this set can have only one owner (Key constraint) otherwise, they are not uniquely determined by the strong entity set.

Another Example is the Course and Section entity sets.



In this case, the weak entity is the section which is dependent on the strong entity course. Together with the course-id and sec-id, year and semester the section can be uniquely determined. As stated before, a double line is used to indicate a weak entity and an identifying relationship.

Why not convert each weak entity to a strong entity?
For this we require multiple copies of the primary key, which leads to storage redundancy. (Wastage of a large amount of storage)

Class Hierarchies

(Specialization and Generalization)

Class hierarchies can be viewed in one of two ways:

People

--Students

-Graduate

-Undergraduate

-Part time

-Full time

--Employees

-Academic

-Non-academic

-Permanent

-Contract

1. People can be specialized into Students and Employees (Specialization-Top Down Approach).
   Specialization is the process of identifying the subsets of a superclass (entity set) that share some special attributes.

2. Student and Employees can be generalized into People. (Generalization-Bottom Up Approach).
   Generalization is the process of identifying common characteristics from a collection of entity sets and creating a new set that possesses these characteristics.

Overlapping vs. Disjoint:

Disjoint - There is no entity common to the subclasses

Overlapping – There is at least one entity common to the subclasses

Full vs. Partial:

Full – All the entities of the super class belong to at least one of the subclasses. Also known as **covering**.

Partial – At least one of the entities of the super class doesn't belong to either of the subclasses.

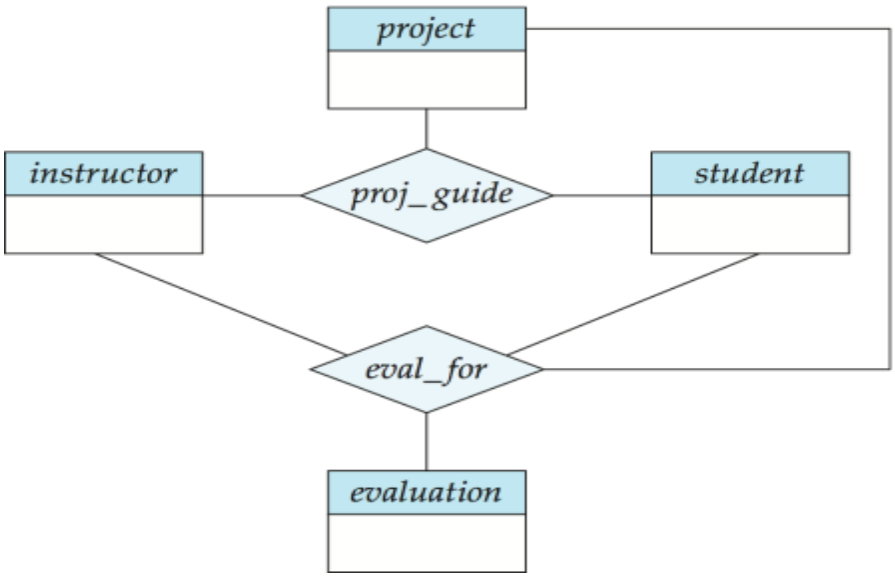
The two main reasons for identifying subclasses:

1. To add descriptive attributes that makes sense only for the entities in the subclasses.
2. To identify the set of entities that participates in some relationship.

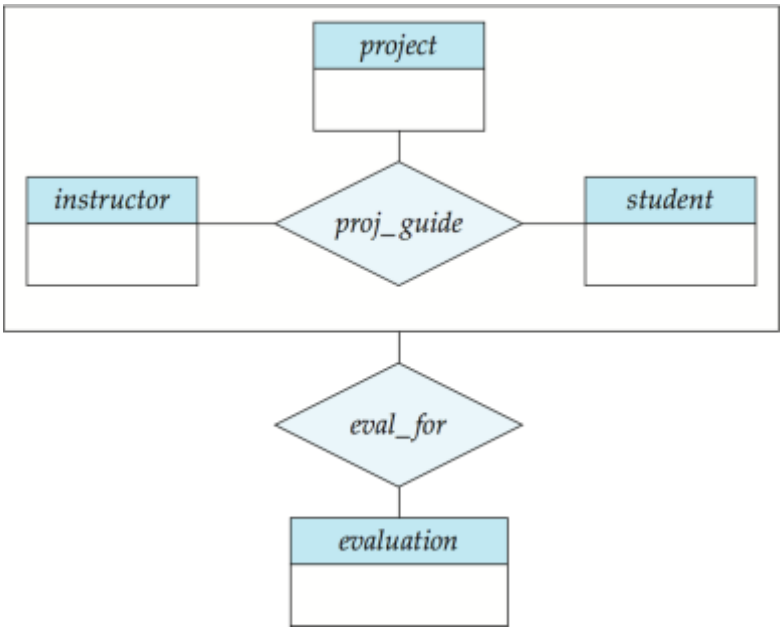
Aggregation

Aggregation is an abstraction that treats relationships as entities. Sometimes we have to model relationship between a collection of entities and relationships.

Consider the case where we have to carry out evaluation for a project for which Professor X is guiding students Y and Z.



We have a relationship called *proj_guide* which maps project, student, instructor entity sets. The evaluation is done for all i.e. the student, the instructor and the project. So basically the relationship *eval_for* maps the relationship *proj_guide* to the entity set evaluation. Using aggregation we can treat the *proj_guide* relation along with the instructor, student, project entity sets as a single entity set on which we can define a relationship called *eval_for*.



This is shown by enclosing the relationship and the corresponding entity sets in a box and then treating it as a single entity. In the first case, *eval_for* has been defined on the instructor, student and the project entities separately though it is really a characteristic of the three put together. Aggregation rectifies this.

Design Issues during ER modeling

Entity vs. Attribute:

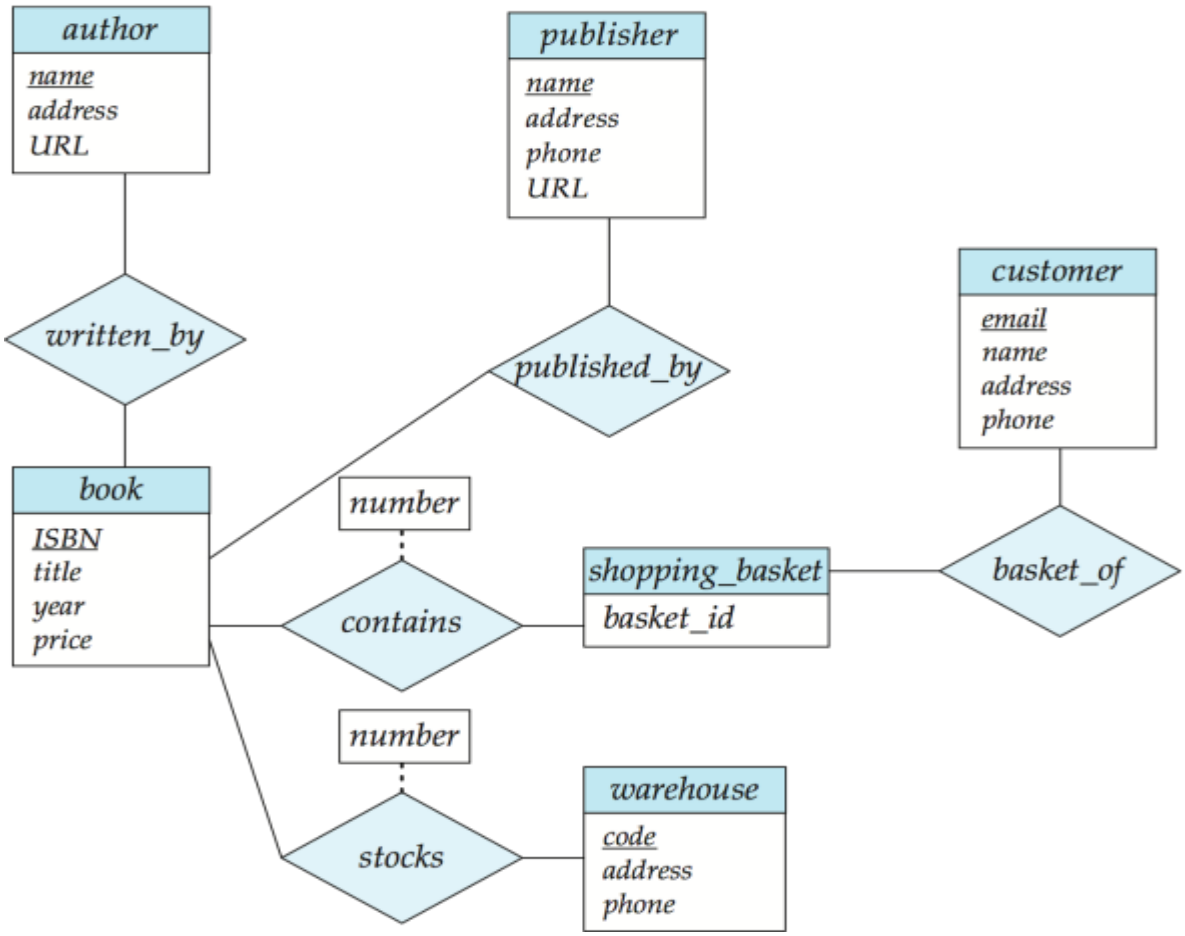
While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modelled as an attribute or as an entity set. For example, the phone number of an employee. One option is to use the attribute *phone_no*. This is suitable is we only have to store one phone number per employee.
Another option is to create an entity set called *phone_nos* and have a relationship with employees. This is useful in two cases

1. We have to record more than one phone number.
2. We have to capture the structure of the phone number like country code, city code, number etc. This can be useful to conduct searches like all the phone numbers of a particular city.

Entity vs. Relationship:

Consider a relationship manages between two entity sets, employees and departments.(Some of the employees manage some departments). Suppose that each manager is given some budget to manage a department. In this case , given a department we know the manager (Assuming a key constraint) and also the budget given. But what if the budget is the sum that covers all the departments of the manager? In this case budget is the attribute of a specific manager and hence we must create a new entity set called managers which is a subclass of the employees entity and then associate the budget with the new entity set.

Practice Problem



What if this online shop wanted to sell e-books?

Hints for solving the problem:

1. Remove the warehouse entity and the stocks relation. (E-books do not require warehouse storage)
2. The book entity must have full participation (double-line) with the *written_by* relationship (every book must have an author or a group of authors)
3. The number attribute may remain on the *contains* relationship but it now indicates the number of digital copies.

END OF LECTURE