

## ER DIAGRAM

*ER) diagram*, a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

An Entity Relationship Diagram (ERD) is a visual representation of different data using conventions that describe how these data are related to each other

## ER Diagram

An entity is a piece of data-an object or concept about which data is stored.

A relationship is how the data is shared between entities

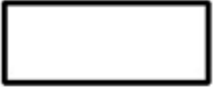
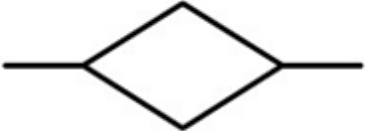


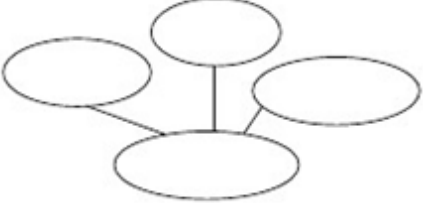

1976 proposed by Peter Chen

ER diagram is widely used in database design

Represent conceptual level of a database system

Describe things and their relationships in high level

# ER diagram Symbols

	Entity
	Relationship
	Simple Attribute
	<u>Multivalued Attribute</u>
	Composite Attribute
	Derived Attribute



**Entity**



**Attribute**



**Relationship**



**Weak  
Entity**



**Multivalued  
Attribute**



**Weak  
Relationship**

# What is An Entity(e.g. car, student)

- **An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.**

# Types of Entity

- **Strong entities** exist independently from other entity types. They always possess one or more attributes that uniquely distinguish each occurrence of the entity.
- **Weak entities** depend on some other entity type. They don't possess unique attributes (also known as a primary key) and have no meaning in the diagram without depending on another entity.

# Weak Entity

- **The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set..** Consider an entity set Payment which has three attributes: payment\_number, payment\_date and payment\_amount. Although each payment entity is distinct but payment for different loans may share the same payment number. Thus, this entity set does not have a primary key.

# Weak Entity

- **A weak entity is an entity that depends on the existence of another entity.** In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributes to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of order.



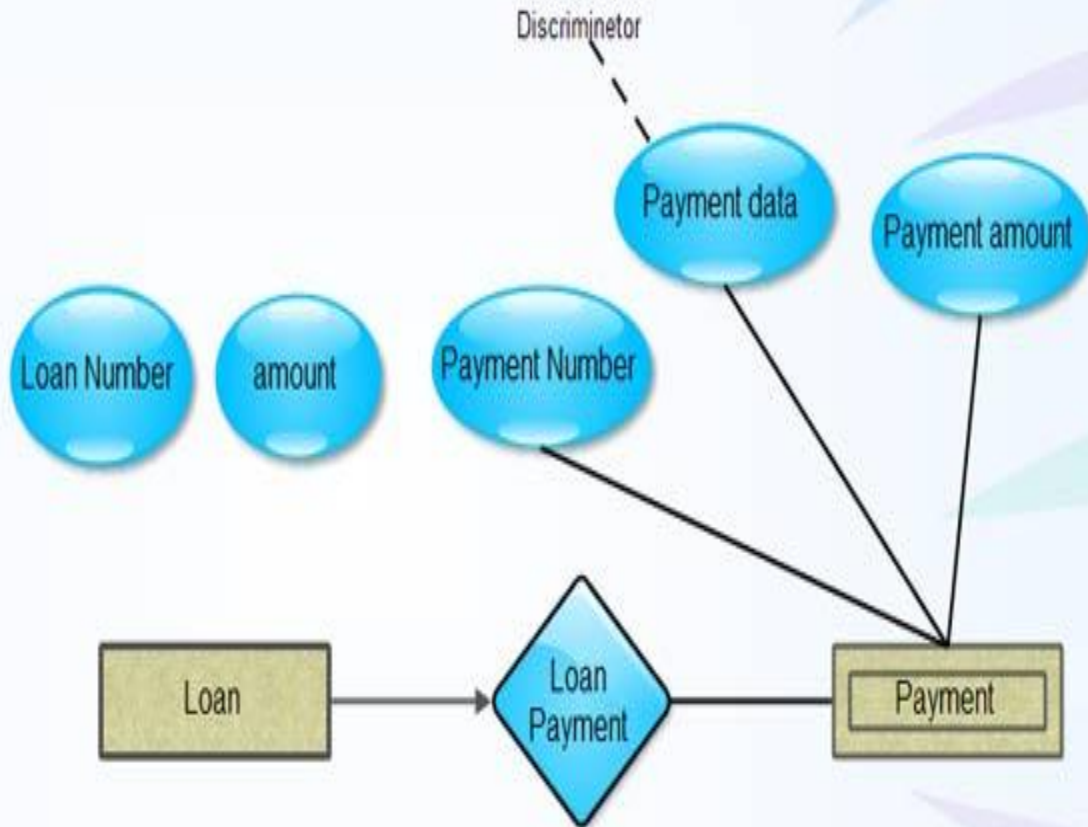
**Order**

```
graph LR; Order[Order] --- OrderItem[Order Item];
```

The diagram consists of two yellow rectangular boxes with brown borders. The box on the left is labeled 'Order' and the box on the right is labeled 'Order Item'. A horizontal brown line connects the right side of the 'Order' box to the left side of the 'Order Item' box. The 'Order Item' box has a double brown border, while the 'Order' box has a single border.

**Order Item**

## Relation Between strong and weak entity set



# Strong Entity

- **An entity set that has a primary key is called as Strong entity set.**

# Difference between strong and weak entity

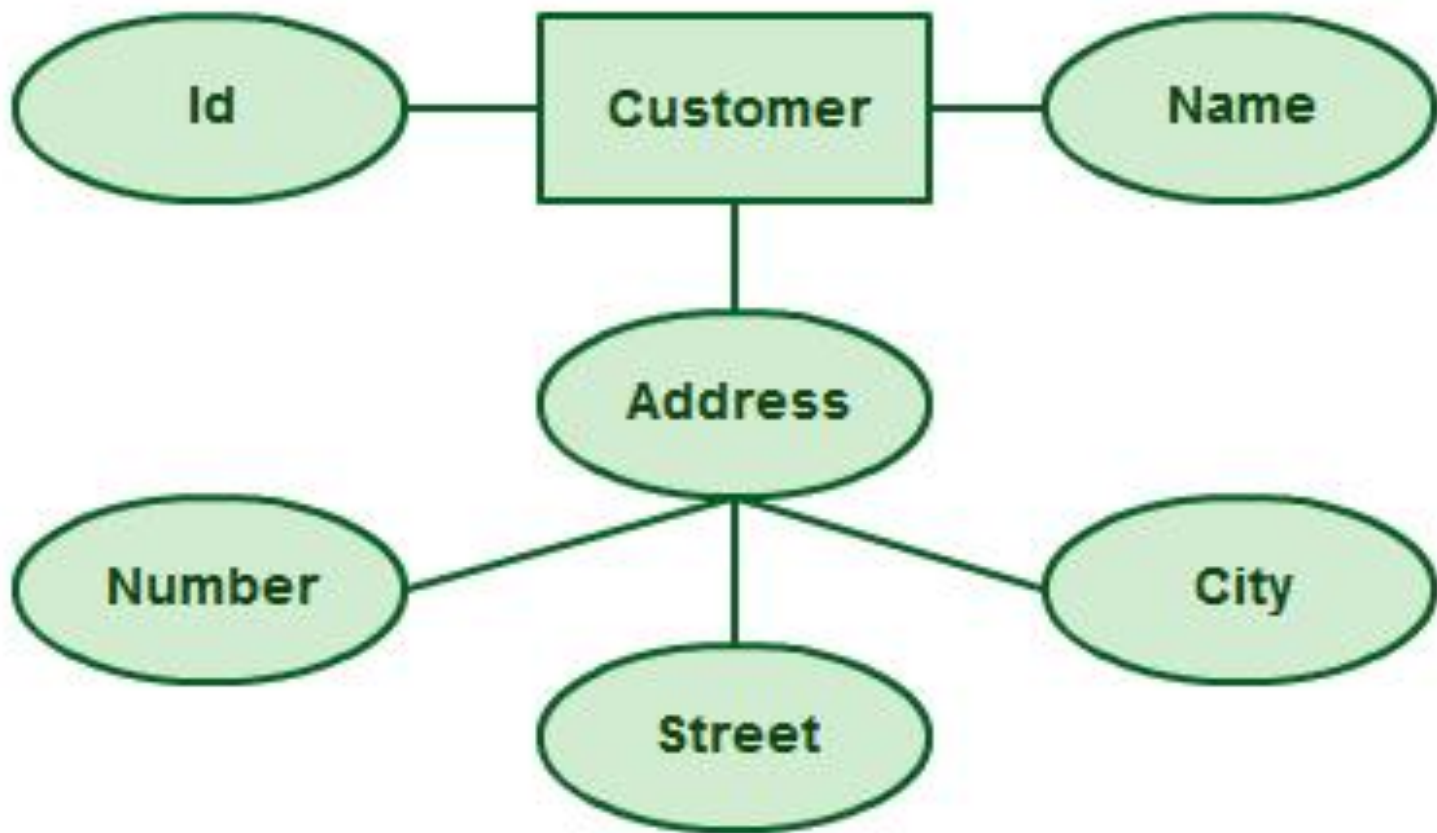
Strong Entity Set	Weak Entity Set
it has its own primary key.	It does not have sufficient attributes to form a primary Key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its attributes which uniquely Identifies its member.	The Primary Key of weak entity set is a combination of partial Key and Primary Key of the strong entity set.
The relationship between two strong entity set is represent by a diamond symbol.	The relationship between one strong and a weak entity set is represented by a double diamond sign. It is known as identifying relationship.
The line connecting strong entity set with the relationship is single	The line connecting weak entity set with the identifying relationship is double.

# Attribute (Properties of entity)

- An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item.

# Types of Attributes

- **Multivalued attributes** are those that are capable of taking on more than one value.
- **Derived attributes** are attributes whose value can be calculated from related attribute values.



# Multivalued Attributes

- If an attribute can have more than one value it is called an multivalued attribute. It is important to note that this is different to an attribute having its own attributes. For example a teacher entity can have multiple subject values.
-

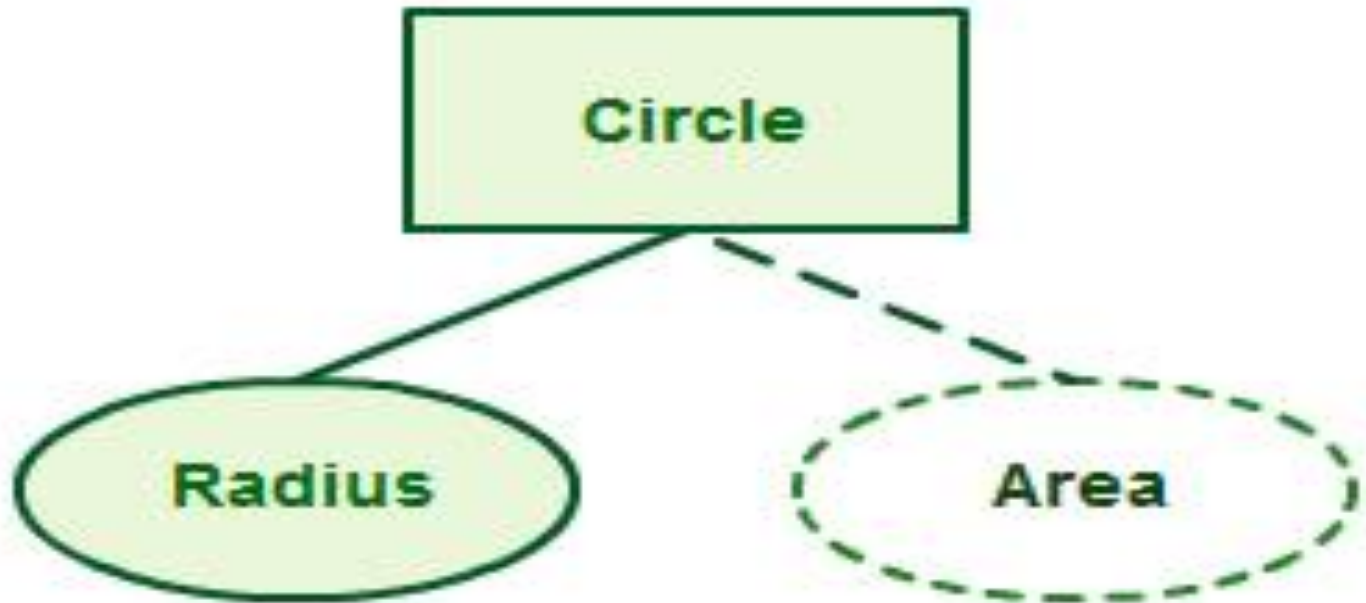




# Derived Attribute

- An attribute based on another attribute. This is found rarely in ER diagrams. For example for a circle the area can be derived from the radius.

# Derived attribute



# Relationship

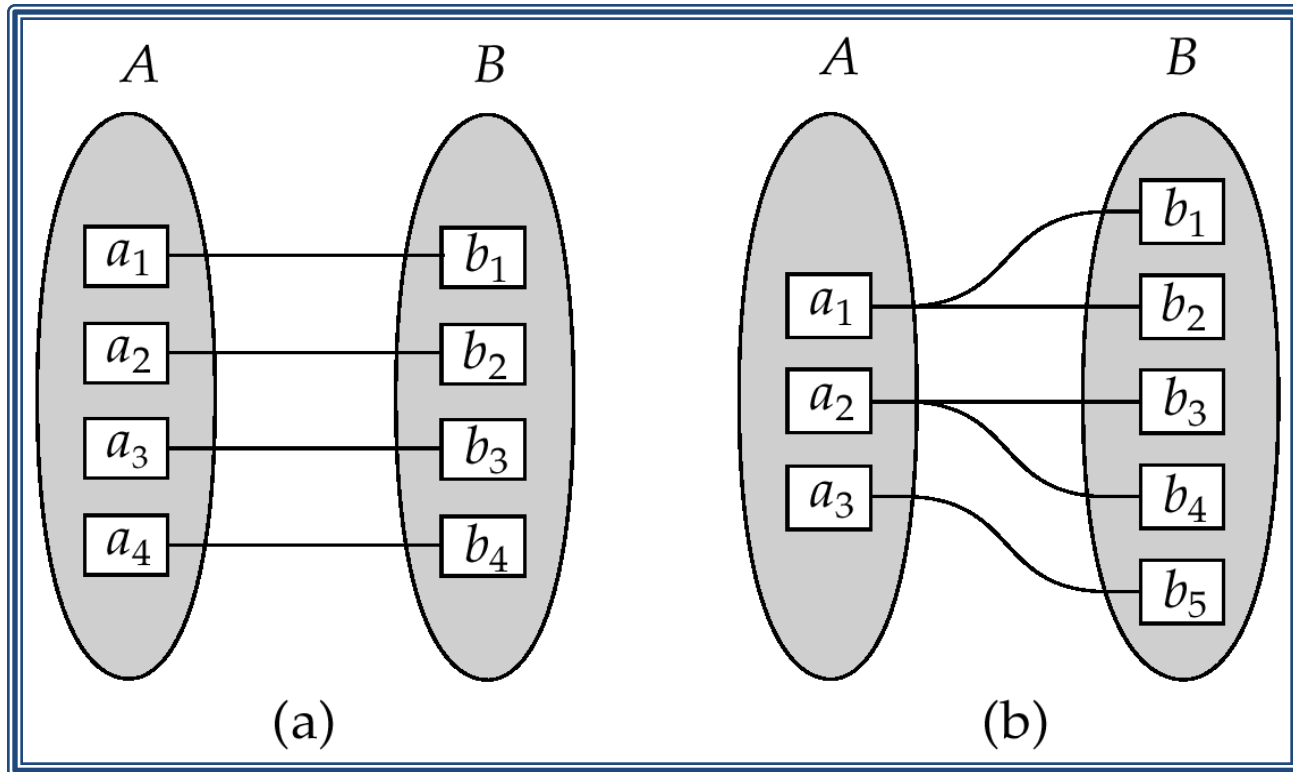
- A relationship describes how entities interact. **For example, the entity “carpenter” may be related to the entity “table” by the relationship “builds” or “makes”.** Relationships are represented by diamond shapes and are labeled using verbs.



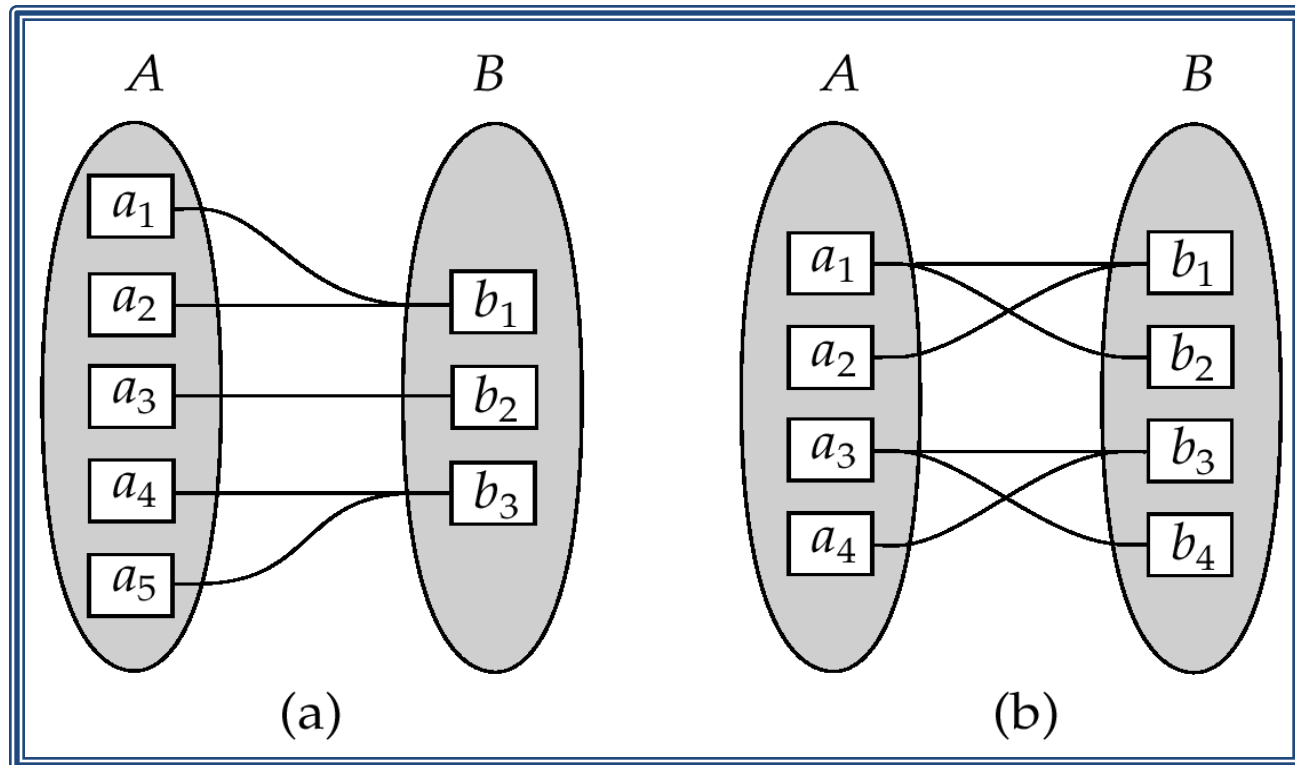
# Relationship

- The degree of a relationship = the number of entity sets that participate in the relationship
  - Mostly binary relationships
  - Sometimes more
- Mapping cardinality of a relationship
  - 1 – 1
  - 1 – many
  - many – 1
  - Many-many

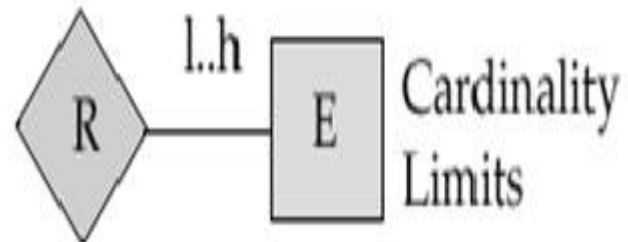
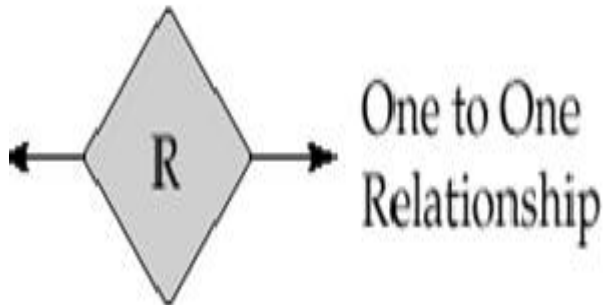
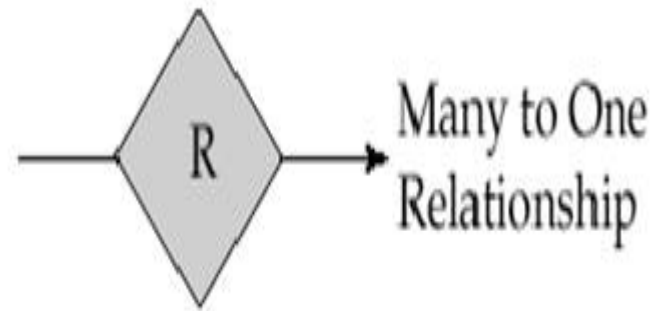
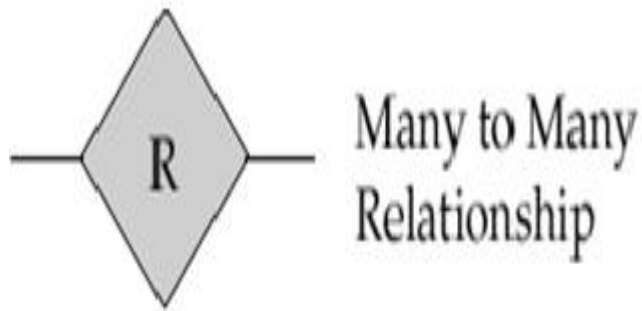
# One-One and One-Many



# Many-one and many-many







# Cardinality

- In the relational model, tables can be related as any of "one-to-many" or "many-to-many." This is said to be the **cardinality** of a given table in relation to another.
- For example, consider a database designed to keep track of hospital records. Such a database could have many tables like:
  - a *doctor* table with information about physicians;
  - a *patient* table for medical subjects undergoing treatment;
  - and a *department* table with an entry for each division of a hospital.
- In that model:
  - There is a **many-to-many** relationship between the records in the doctor table and records in the patient table because doctors have many patients, and a patient could have several doctors;
  - A **one-to-many** relation between the department table and the doctor table because each doctor may work for only one department, but one department could have many doctors.

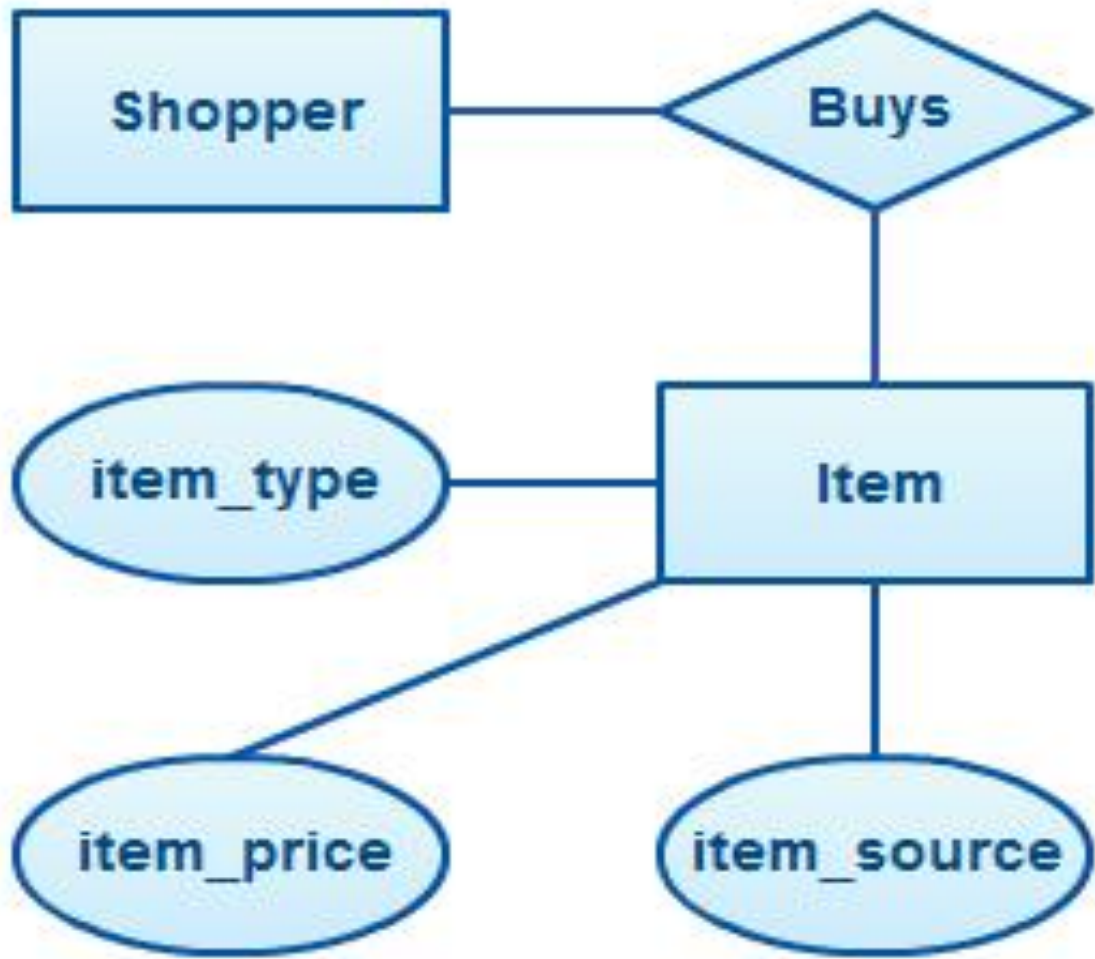
# Cardinality

- These two further defines relationships between entities by placing the relationship in the context of numbers. In an email system, for example, one account can have multiple contacts. **Cardinalities are used when you are creating an E/R diagram, and show the relationships between entities/ tables.**



# What is the use of ER Diagram

- ER diagrams are most often associated with complex databases that **are used in software engineering and IT networks**. In particular, **ER diagrams are frequently used during the design stage of a development process in order to identify different system elements and their relationships with each other**. For example, an inventory software used in a retail shop will have a database that monitors elements such as purchases, item, item type, item source and item price. Rendering this information through an ER diagram would be something like this:



# Why ER diagram

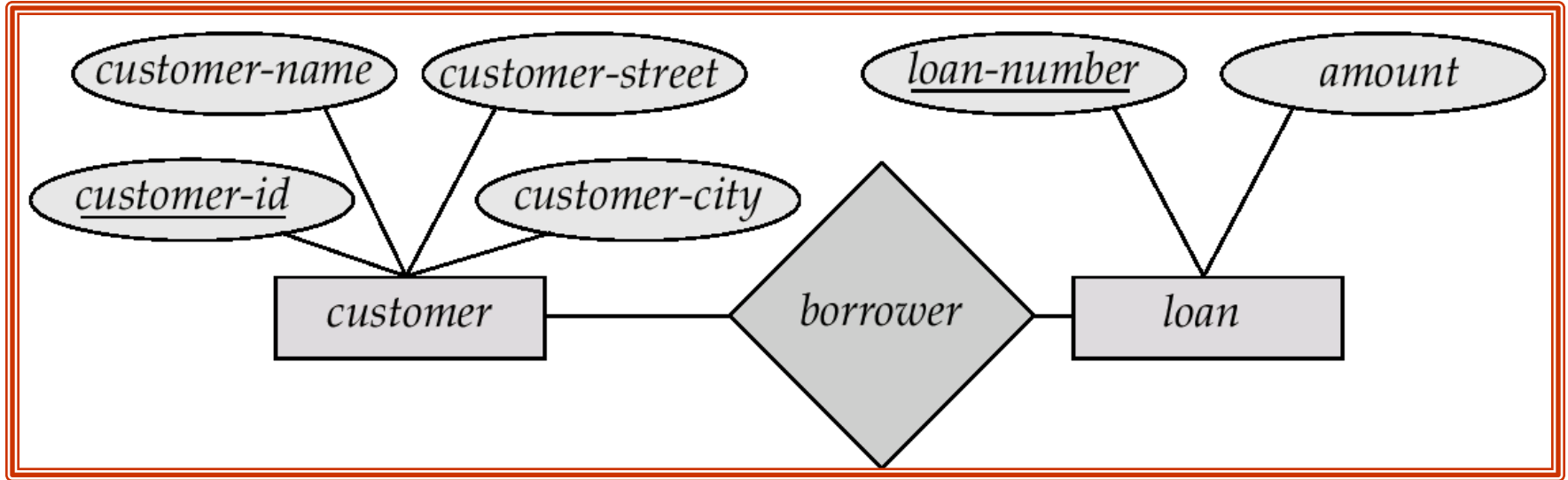
- ER diagrams constitute a very useful framework for creating and manipulating databases. First, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately. This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency. Second, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the blueprint for implementing data in specific software applications. Lastly, ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.

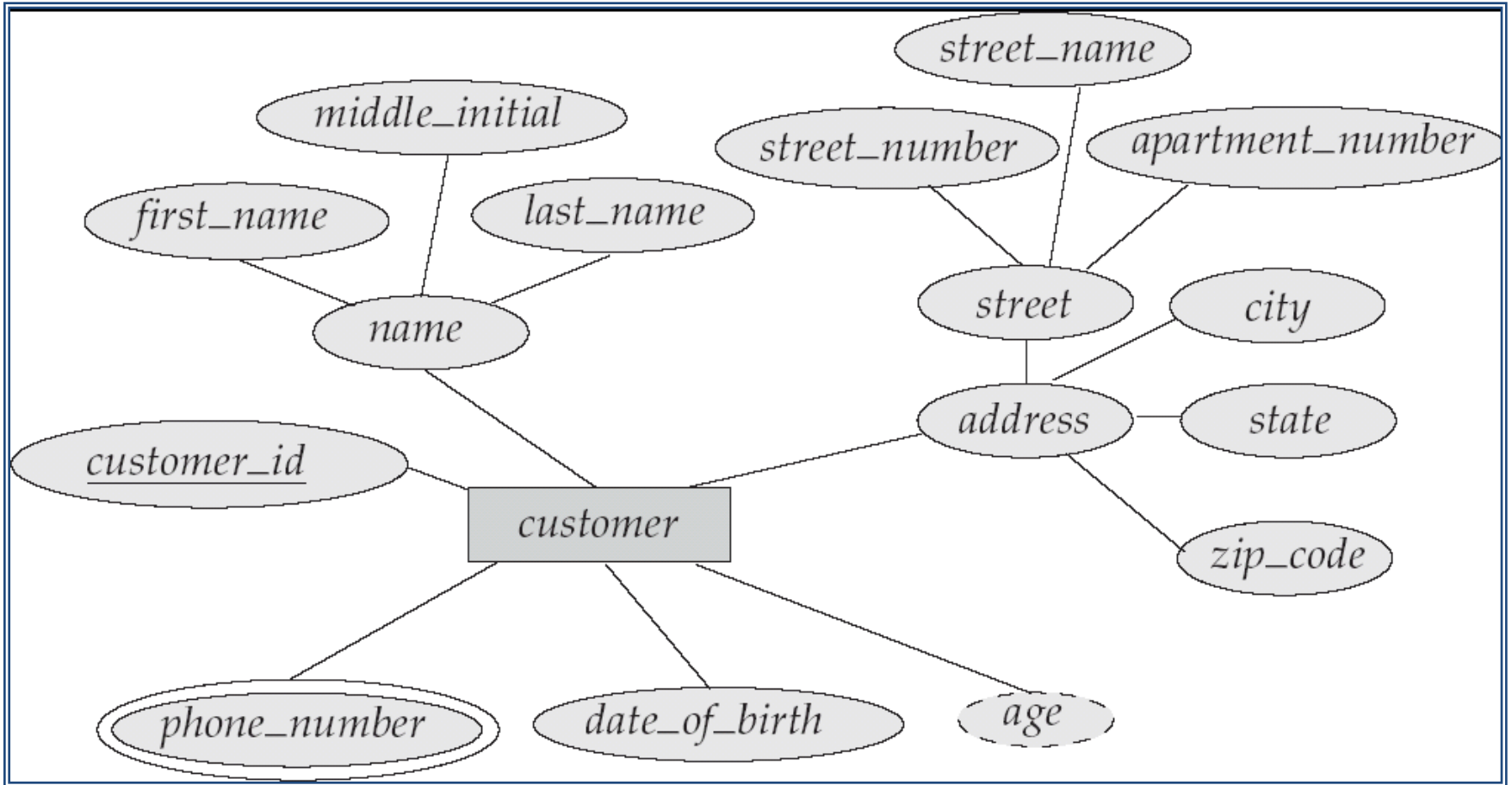
# Tips on How to Draw ER Diagrams

- **Identify all the relevant entities in a given system** and determine the relationships among these entities.
- An entity should appear only once in a particular diagram.
- **Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram.** Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
- **Remove vague, redundant or unnecessary relationships between entities.**
- **Never connect a relationship to another relationship.**
- **Make effective use of colors.** You can use colors to classify similar entities or to highlight key areas in your diagrams.

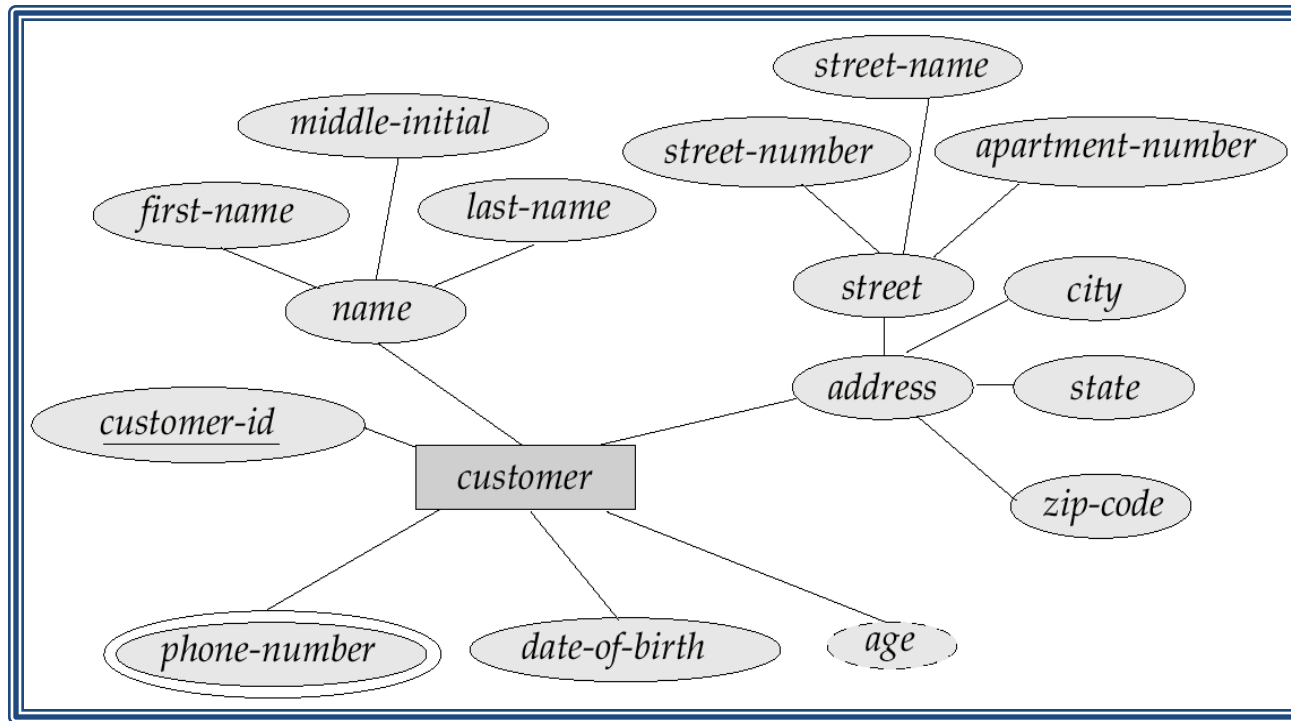


# Example of Customer ER diagram with primary key

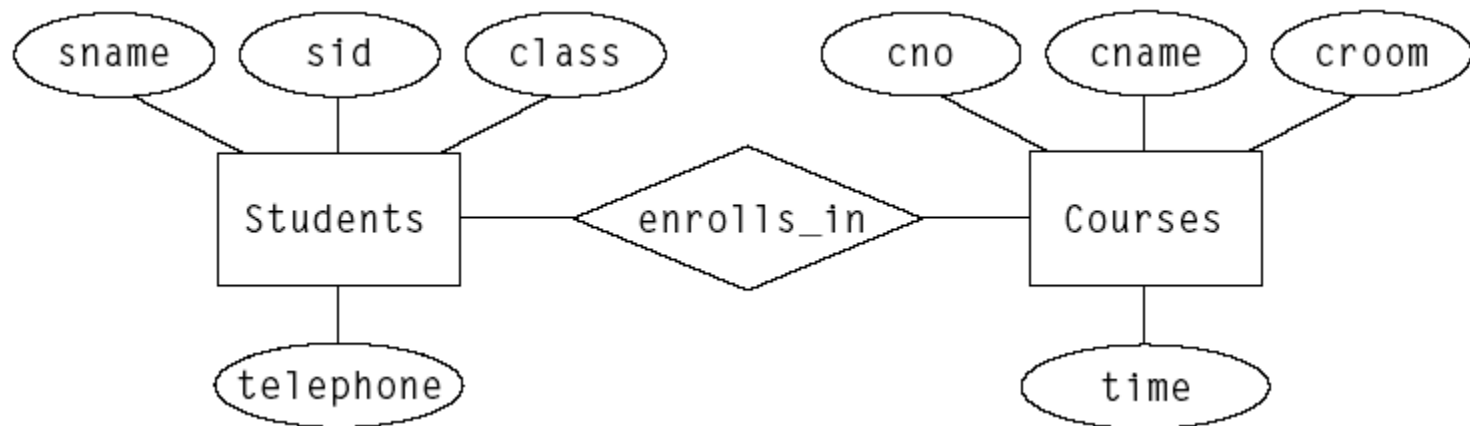




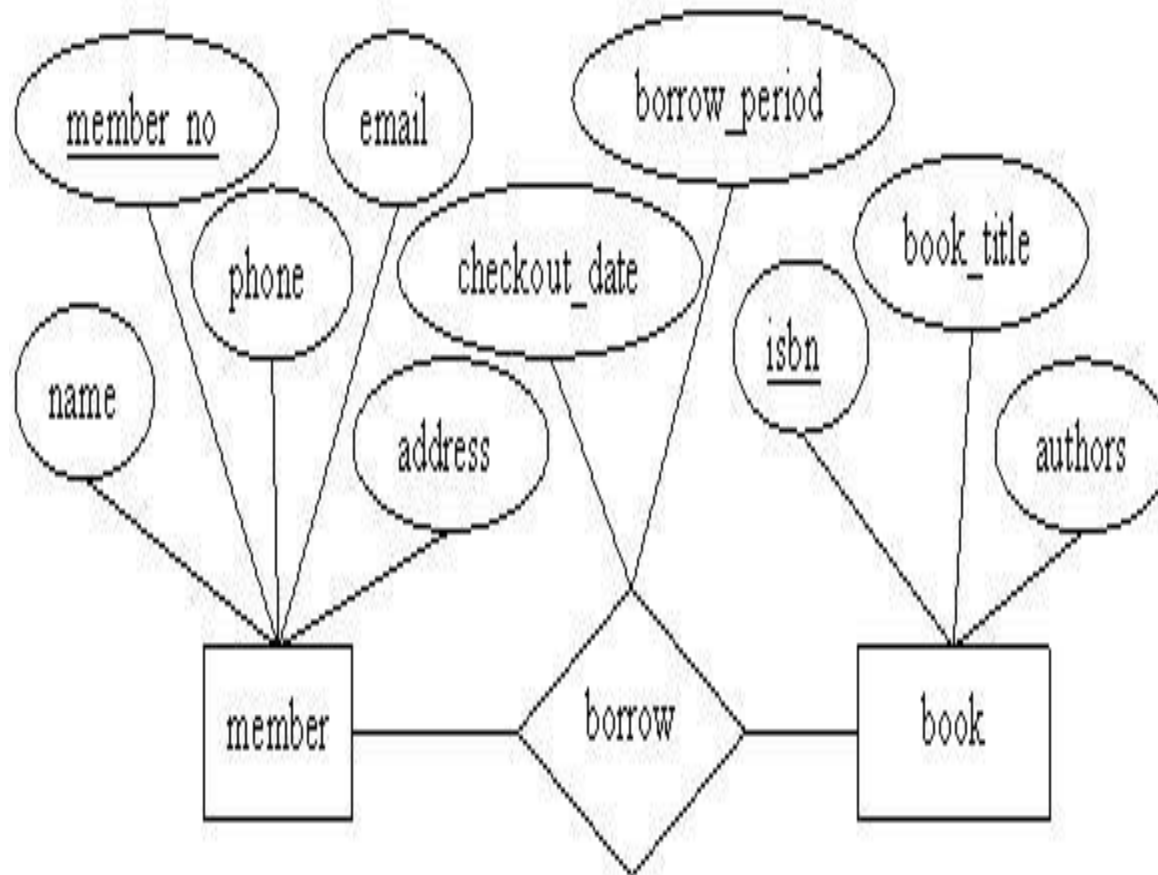
# Another Example with multivalued and Derived Attribute



# Simple ER diagram for student

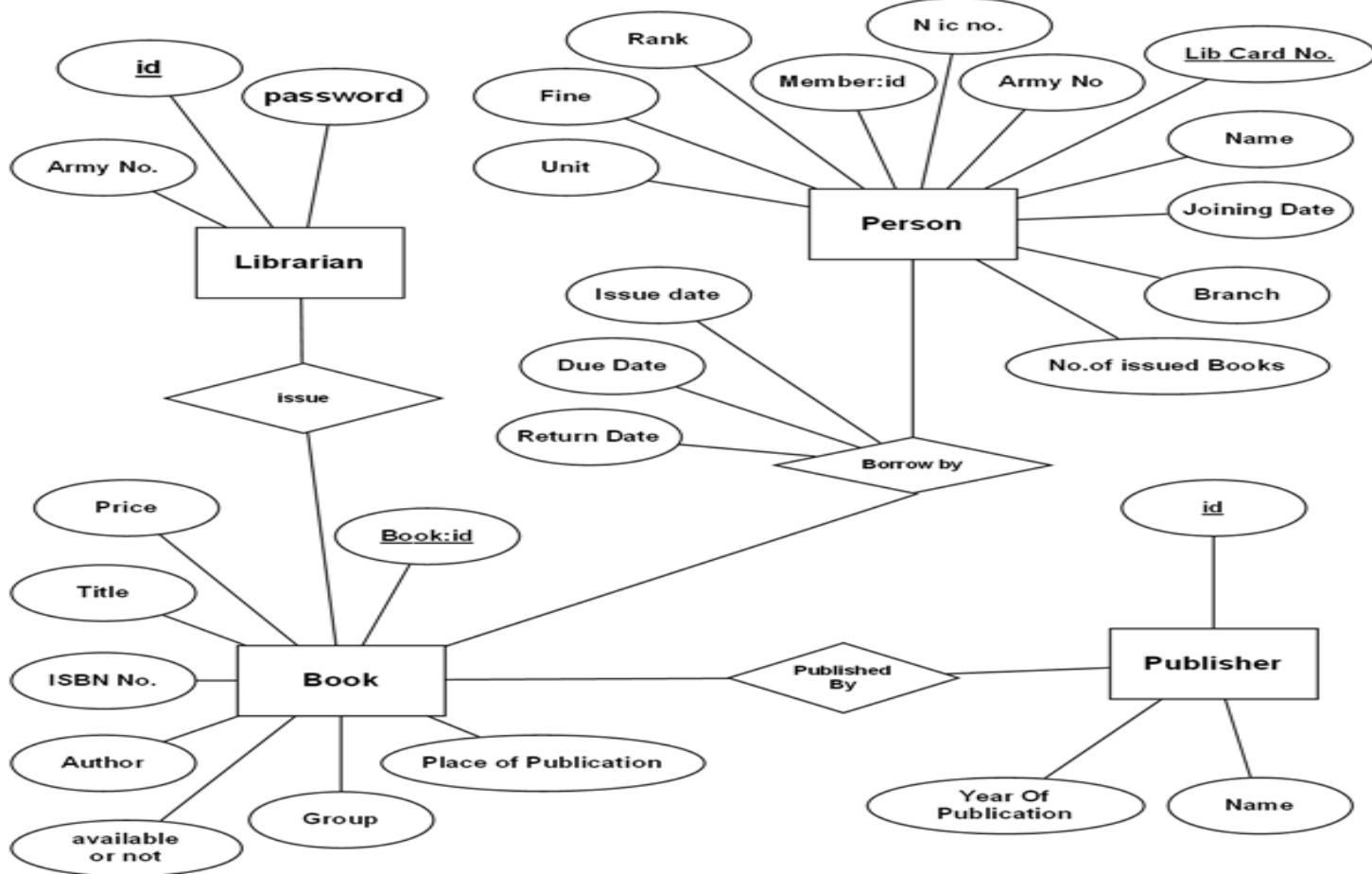


# Library Management System

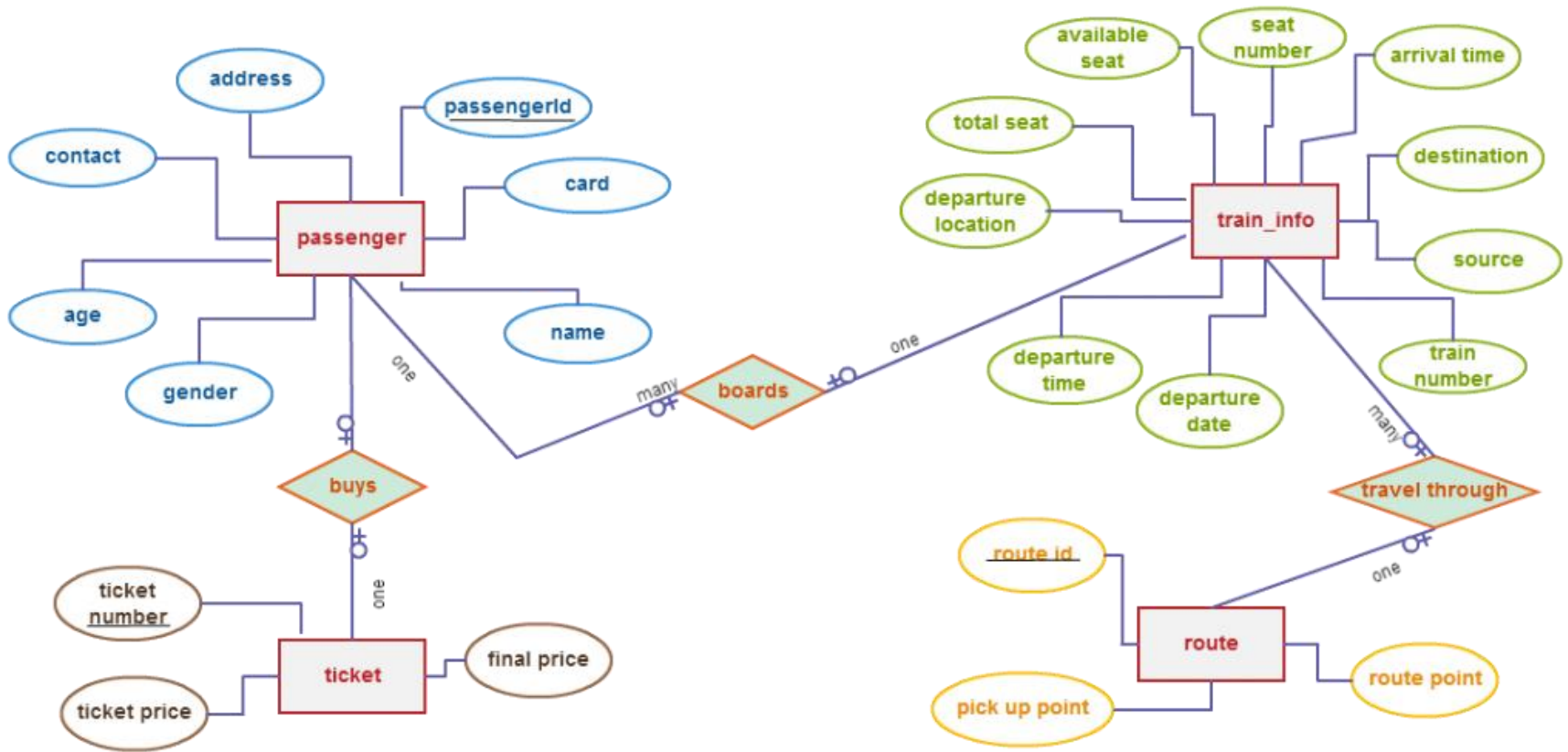


# Library management system

Library Management System For Army Erd (Srs Assg Uos)

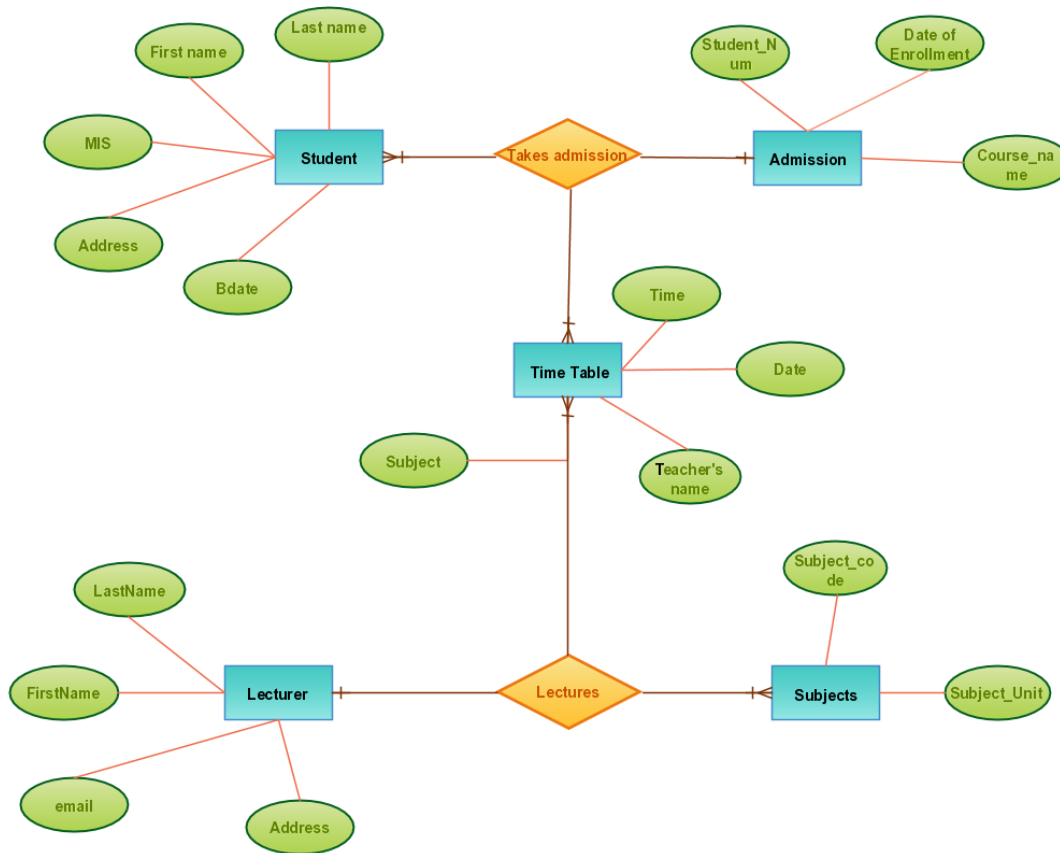


# Railway reservation system



# College Management System

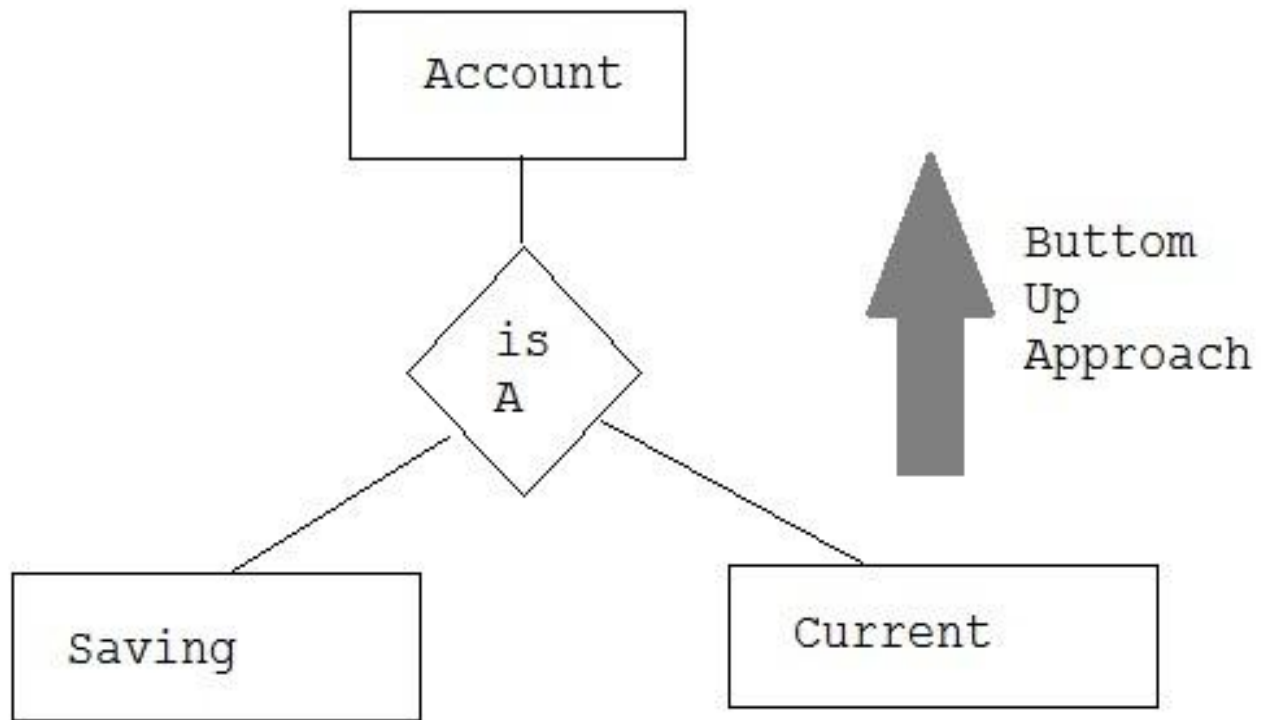
ER DIAGRAM FOR COLLEGE MANAGEMENT SYSTEM





# Generalization

Bottom up approach in which two low level entities are combined together to form high level entities



# Specialization

- Opposite to specialization. It is a top down approach in which high level entities can be broken down into lower level entities.

