I. Overview of Design Process

What is the ER Model?

ENTITY RELATIONAL (ER) MODEL is a high-level conceptual data model diagram. ER modelling helps you to analyse data requirements systematically to produce a well-designed database. The Entity-Relation model represents real-world entities and the relationship between them.

It is considered a best practice to complete ER modelling before implementing your database.

ER modelling helps you to analyse data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modelling before implementing your database.

History of ER models

ER diagrams are a visual tool which is helpful to represent the ER model. It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network. He aimed to use an ER model as a conceptual modelling approach.

What is ER Diagrams?

ENTITY-RELATIONSHIP DIAGRAM (ERD) displays the relationships of entity set stored in a database.

In other words, we can say that ER diagrams help you to explain the logical structure of databases.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

Facts about ER Diagram Model:

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modelling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modelling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD is allowed you to communicate with the logical structure of the database to users

II. The Entity Relationship Model

A database can be modelled as: –

- A collection of **entities**
- **Relationships** among entities

• An **entity** is an object that exists and is distinguishable from other objects.
Example: specific person, company, event, plant

• An **entity set** is a set of entities of the same type that share the same properties.
Example: set of all persons, companies, trees, holiday

Relationship Sets

• A relationship is an association among several entities
Example:
Ankit is depositor in Axis Bank and also has Savings Account

• A relationship set is a mathematical relation among n ≥ 2 entities,

Each taken from entity sets {(e1, e2, ..., en) | e1 ∈ E1, e2 ∈ E2, ..., en ∈ En}
Where (e1, e2, ..., en) is a relationship –

Example:
(Ankit, Savings Account) ∈ Bank (name, Account)

Attributes

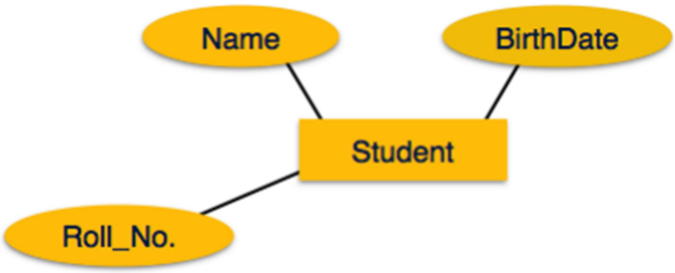
• An entity is represented by a set of attributes, that is, descriptive properties possessed by all members of an entity set.

Example:

Customer = (customer-name, social-security, customer-street, customer-city)
Account = (account-number, balance)

Domain

The set of permitted values for each attribute
• Attribute types:
– Simple and composite attributes
– Single-valued and multi-valued attributes
– Null attributes
– Derived attributes



An attribute can also be a property of a relationship set.

For instance, the Student relationship set between entity sets Name and Roll_no may have the attribute Birth-date

III. Constraints

We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set

A relationship where two entities are participating is called **a binary relationship**.

Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

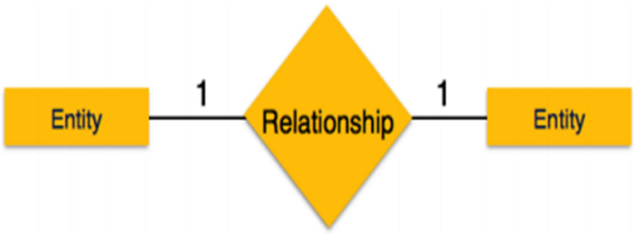
There are four types of it

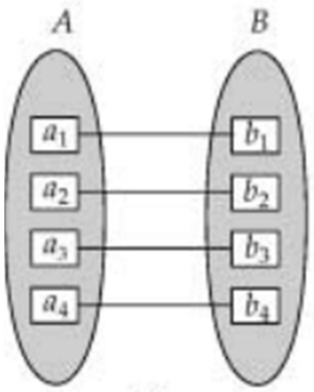
a) One to One
b) One to Many
c) Many to One
d) Many to Many

Let us discuss them one by one

One-to-one Constraint

When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance

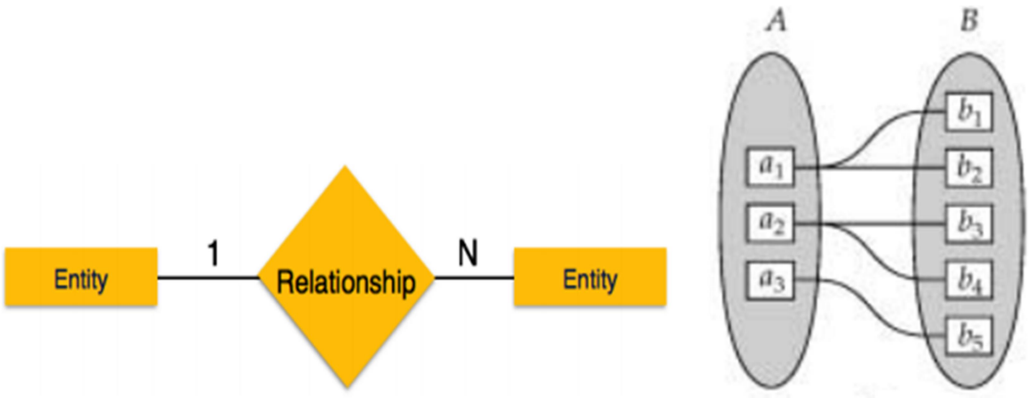




One-to-many Constraint

When more than one instance of an entity is associated with a relationship, it is marked as '1:N'.

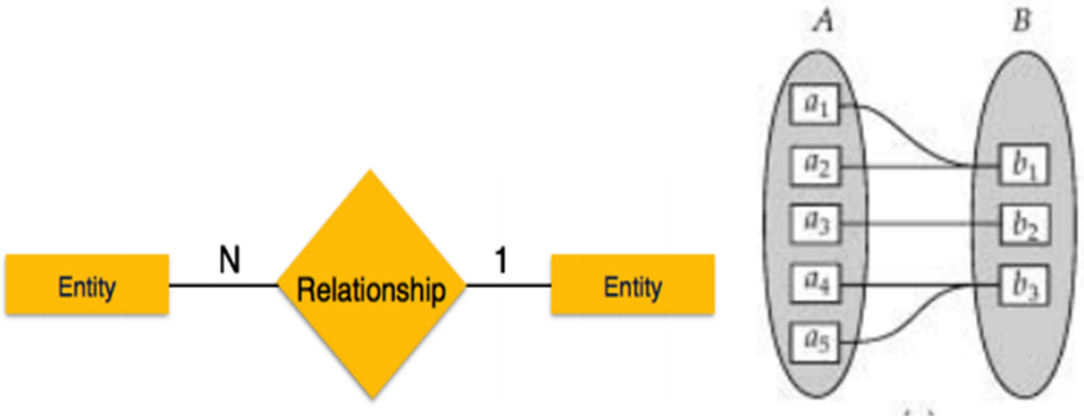
The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship.



Many-to-one Constraint

When more than one instance of entity is associated with the relationship, it is marked as 'N:1'.

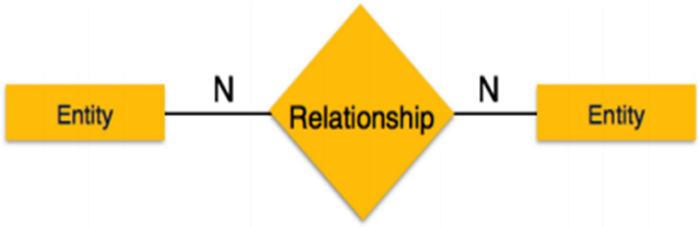
The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.

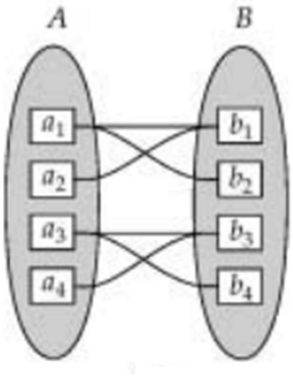


Many-to-many Constraint

The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship.

It depicts many-to-many relationship





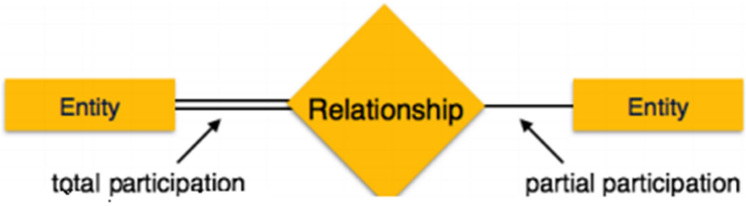
Participation Constraints

Total Participation –

- Each entity is involved in the relationship.
- Total participation is represented by double lines.

Partial participation-

- Not all entities are involved in the relationship.
- Partial participation is represented by single lines.



IV. Removing redundant attributes in Entity sets

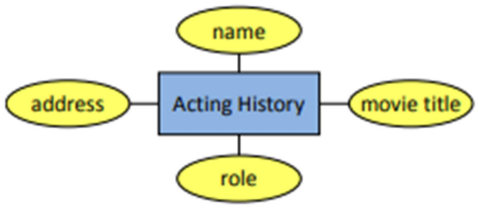
Eliminating redundancy

We say that a model is redundant if it captures the same (or very similar) information multiple times. Redundancy usually manifests in one of two ways:

There may be a structural redundancy, where the same type of information is used in different parts of the model,
Or
Data redundancy, where the same information is stored multiple times.

Data redundancy

We'll examine data redundancy first, using the example below:
Note: Example below is totally for illustration, no hard feelings ☺



Name	Address	Role	Movie Title
Irfan Khan	Mumbai	Teacher	Lunch Box
Irfan Khan	Mumbai	Father/Common Man	Madari
Irfan Khan	Mumbai	Shop Owner	English Medium
Irfan Khan	Mumbai	Police	Neerja
Irfan Khan	Mumbai	Athelete	Paan Singh Tomar
Irfan Khan	Mumbai	Survivor	Life Of Pie

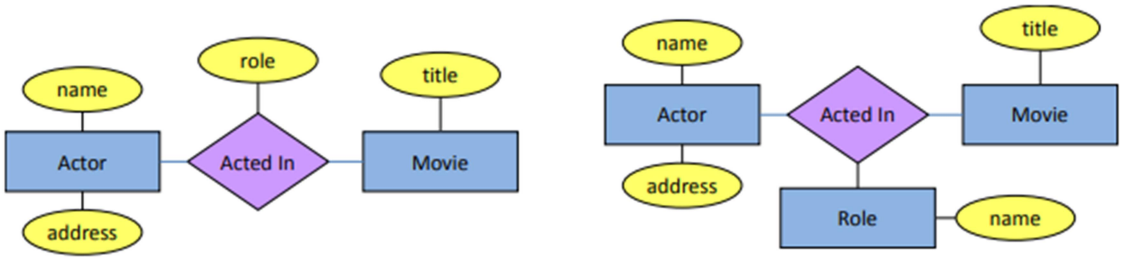
Here, we're trying to model the acting history of an actor.

However, if you look at the table, it becomes clear that there is significant redundancy in this model: The actor's name and address are repeated many times. The problem with this redundancy arises when we want to modify the data.

Nothing in the current model prevents us from giving Irfan Khan a different address for every row of the table, for example.

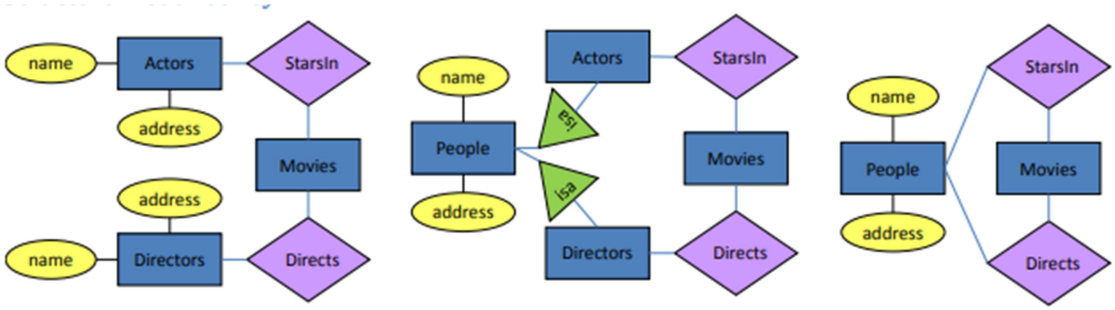
A secondary concern is that the above representation wastes storage space.

The solution for the same can be as below:-



We can remove the redundancy by splitting the acting history into two entity sets: actors and movies.
This lets us decouple actor addresses from movies, since the two really aren't related.
This can be seen in the example above to the left.
However, even though there is now less redundancy than before, we still have redundancy in the roles played by actors.
Looking again at the table above, we can see that Irfan Khan played the same role in multiple movies.
We can capture that information by splitting role into a separate entity set is well, and making acted in a ternary relationship. The final models are shown in on the right side of the above example.

Structural redundancy



In the example above, actors and directors both have names and addresses; this structural redundancy suggests that actors and directors share a similarity that is not captured in the model (namely that both are people).

In this case, it might be better to model directors and actors as subclasses of people, and to move the name and address attributes to the parent class.

Once we do this, however, there are no attributes left in directors and actors, which further suggests that perhaps directors and actors are unnecessary in our model.

In this particular case, the real information is stored in the two relationships, and we can simplify the model significantly by having people star in movies and direct movies. Doing so also eliminates the risk of data redundancy that could arise if somebody is both an actor and a director, which is quite common in the movie industry.

Before we continue, it's important to note that database design tools cannot provide very much help in removing structural redundancies. The tool simply has no way to know whether two name attributes are really the same. Consider if we had an entity set "movie studio" that also had a name and address. Even though the two entities sets—actor and movie studio—would look the same, it does not make sense to combine them.

V. ER Diagram

Entity

A real-world thing either living or non-living that is easily recognizable and non-recognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key.

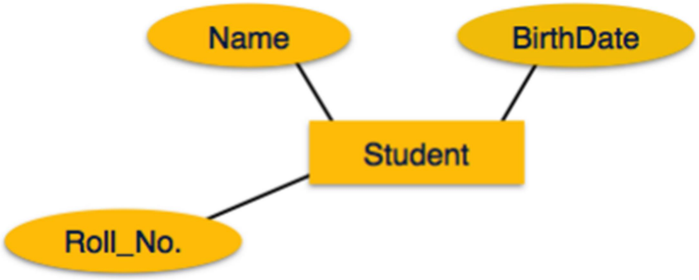
Every entity is made up of some 'attributes' which represent that entity.

- **Person**: Employee, Student, Patient
- **Place**: Store, Building
- **Object**: Machine, product, and Car
- **Event**: Sale, Registration, Renewal
- **Concept**: Account, Course

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Relationship

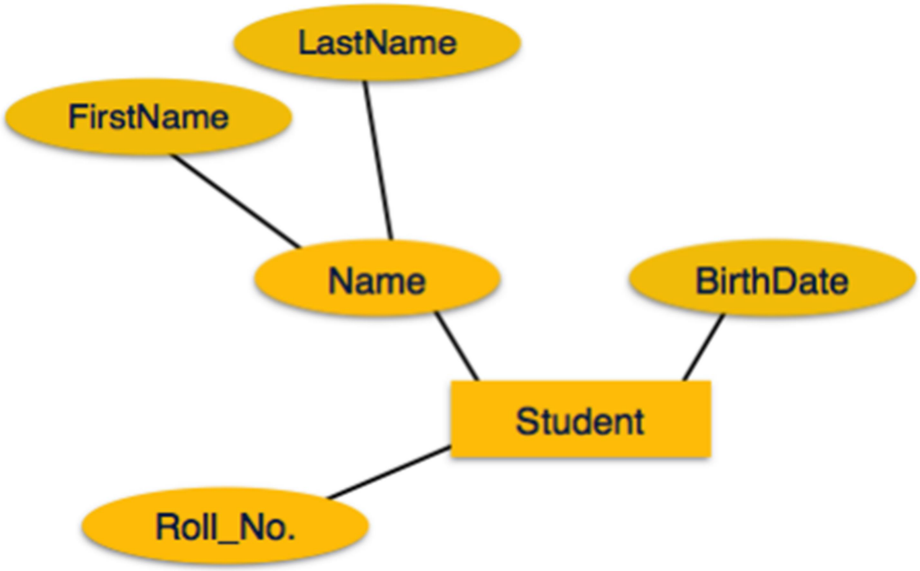
Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities rectangles participating in a relationship are connected to it by a line.



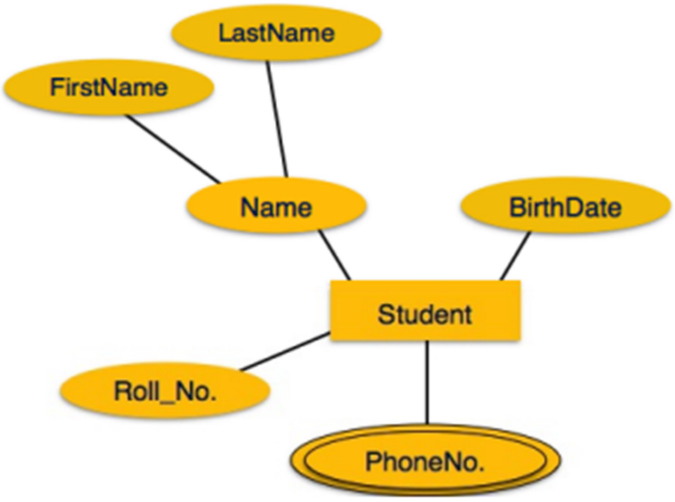
Attributes

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity rectangle.

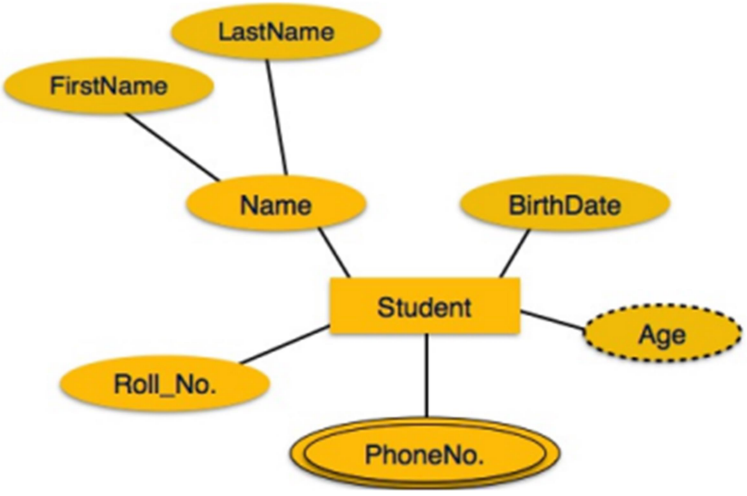
If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.

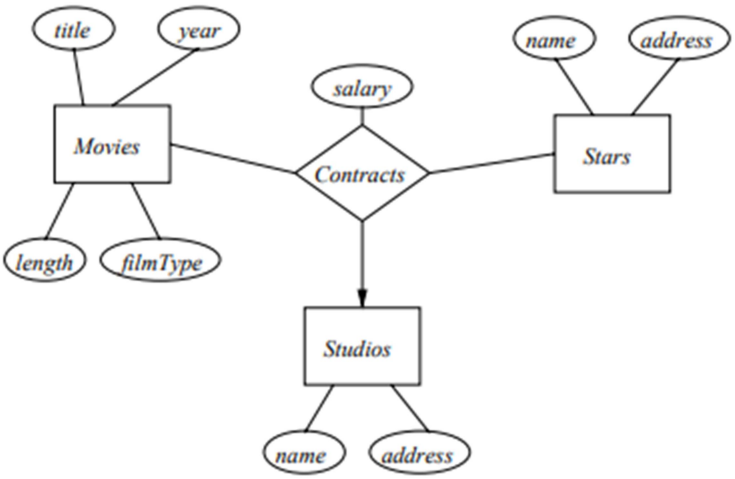


Derived attributes are depicted by dashed ellipse.



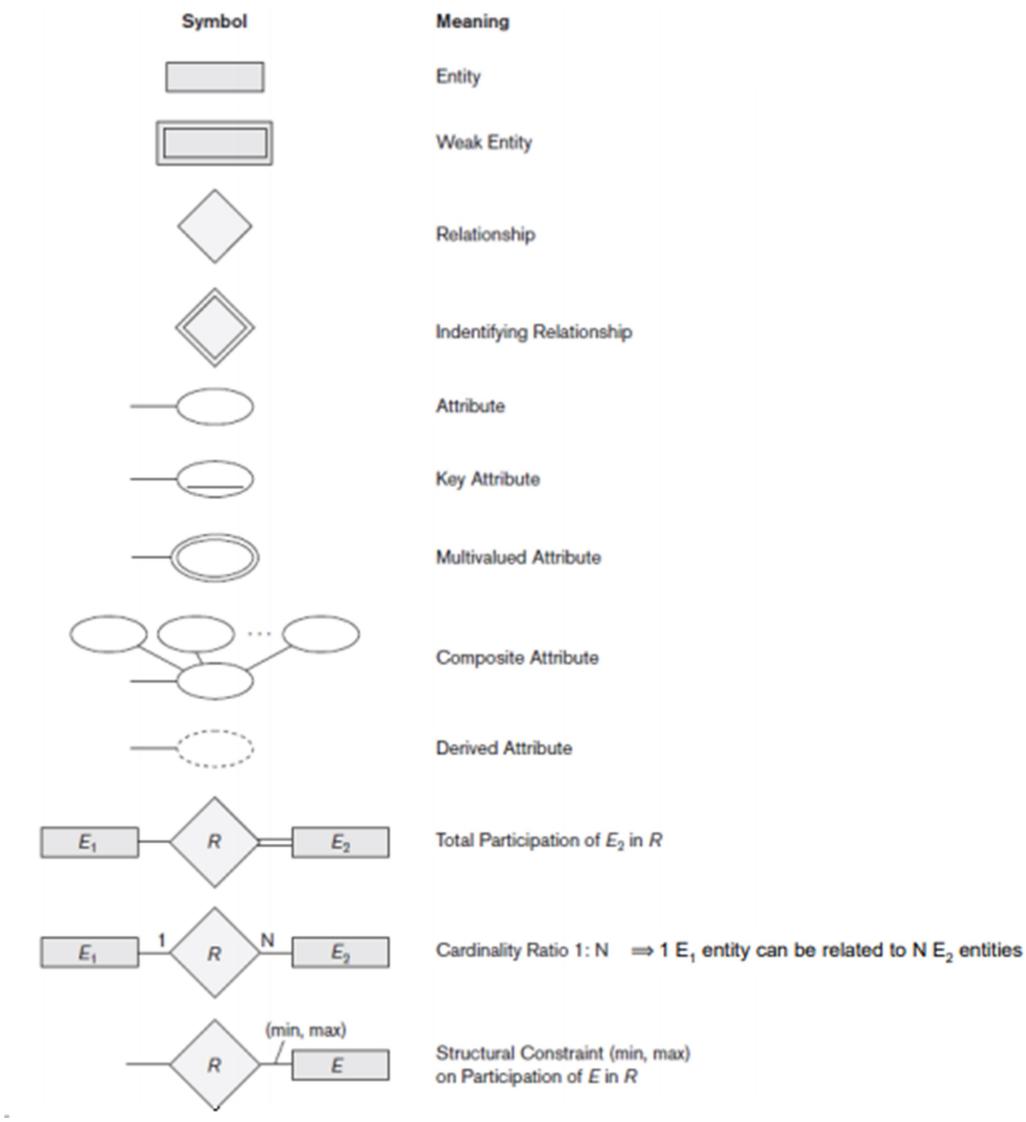
Example:
We shall use as a running example a database about movies, their stars, the studios that produce them, and other aspects of movies.
Each movie is an entity, and the set of all movies constitutes an entity set.
Likewise, the stars are entities, and the set of stars is an entity set.

A studio is another kind of entity, and the set of studios is a third entity set that will appear in our Example.



I want few more innovative examples that we can have from class at-least 1 from each ;) ☺

Below is the list of few symbols that we can use for making the ER Diagrams, each of the symbol represents a different meaning which we will talk about in class and also see examples on the same.



VI. Reduction to relation schemas

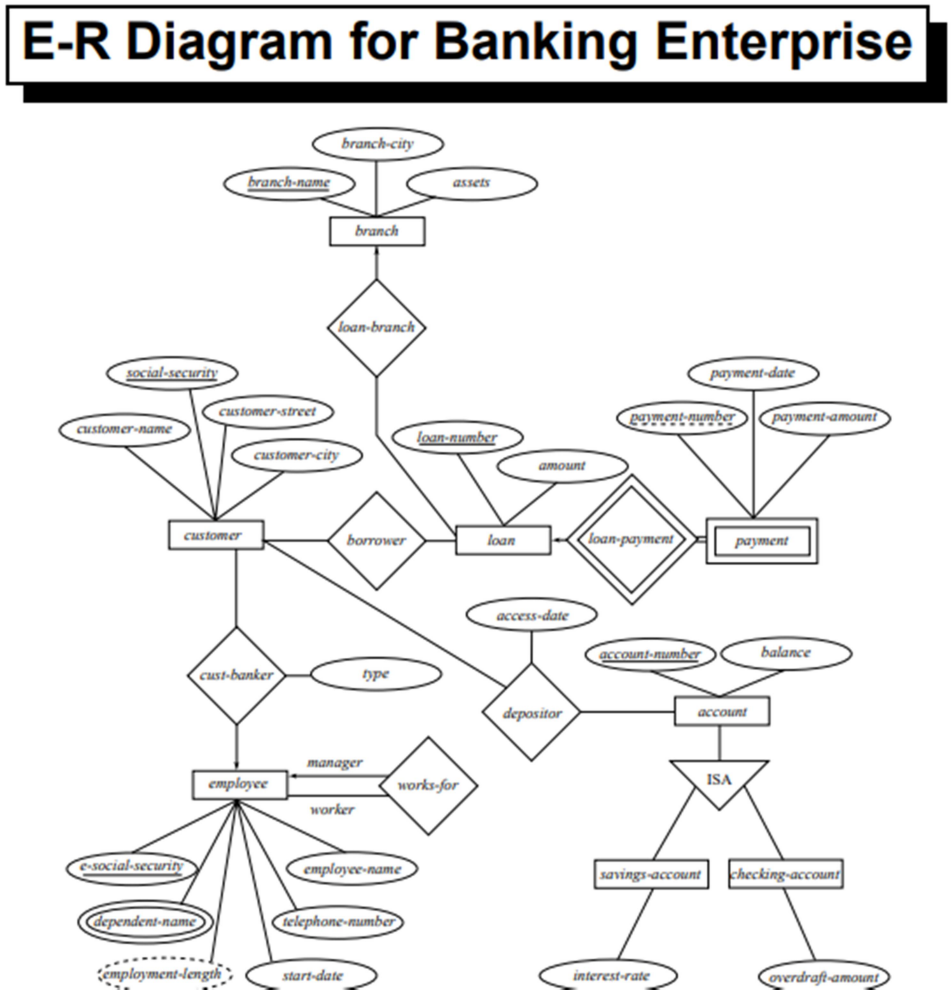
Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database.

A database which conforms to an E-R diagram can be represented by a collection of tables.

For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.

Each table has a number of columns (generally corresponding to attributes), which have unique names.

Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.



A strong entity set reduces to a table with the same attributes.

The customer table

Customer Name	Unique Number	Street	City
Ankit	123-456-789	Vikhroli	Mumbai
Keval	456-789-123	Vile-Parle	Mumbai
Ameya	789-123-456	Bhandup	Mumbai

Payment Table

• A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set.

Loan Number	Payment Number	Payment-Date	Payment Amount
L-17	5	10-May-2019	5000
L-23	11	17-May-2020	7500
L-15	22	23-May-2020	30000

The payment table is weak because? Expected answer from Class

VII. Entity relationship design issues

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 if 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:

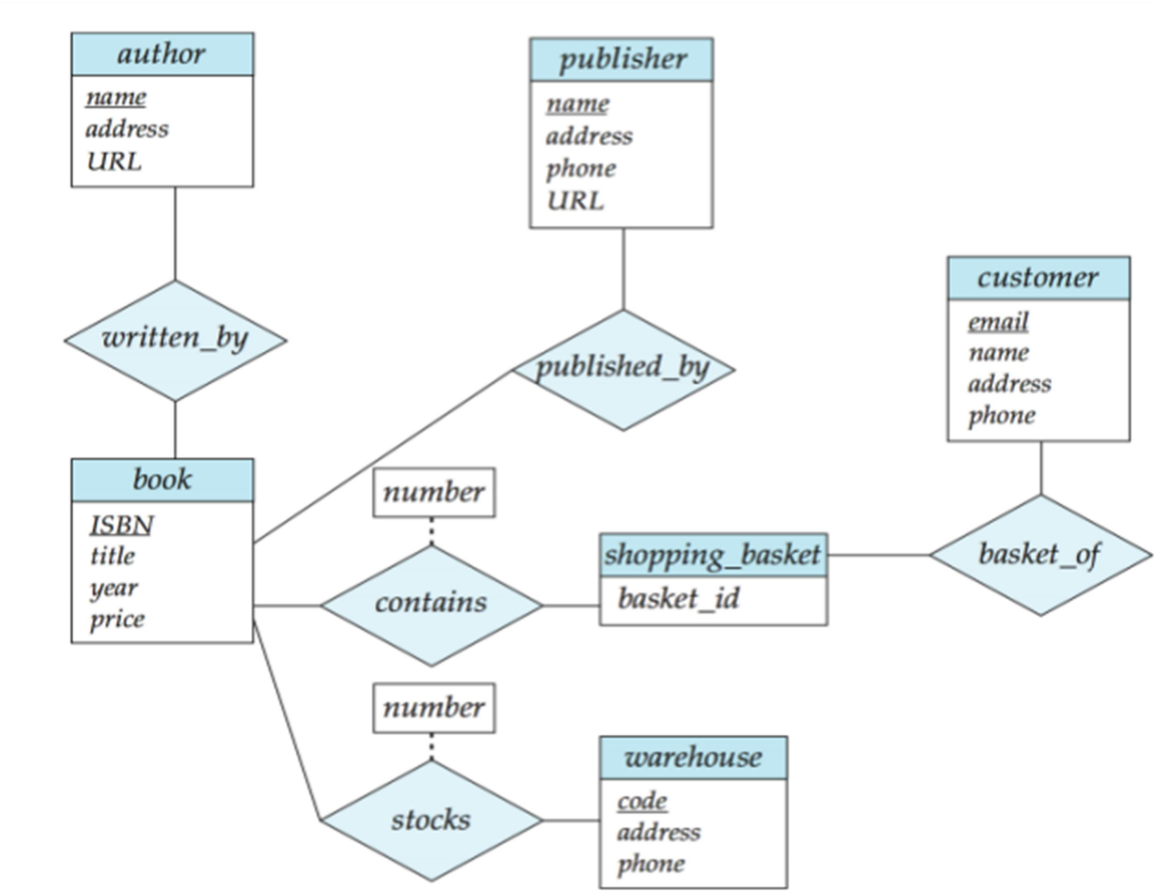
Consider a relationship manages between two entity sets, employees and departments.(Some of the employees manage some departments).

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 employee's entity and then associate the budget with the new entity set.

Let us understand the issue with Example



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

Let's discuss the solution in our class!!!

VIII. Extended ER features

The ER Model that is supported with the additional semantic concepts is called extended entity relationship model or EER model.

The EER model includes concepts of the original ER model together with the following additional concepts

1) Generalization
2) Specialization
3) Aggregation

Generalization

Generalization is the process of defining a more general entity type from a set more specialized entity types

A “bottom up” approach – Taking similar entity-sets and unifying their common features – Start with specific entities, then create generalizations from them

Specialization

- a “top down” approach – Creating general purpose entity-sets, then providing specializations of the general idea – Start with general notion, then refine it
- Terms are basically equivalent – Book refers to generalization as overarching concept
- An entity-set might contain distinct subgroups of entities
  – Subgroups have some different attributes, not shared by entire entity-set
- E-R model provides specialization to represent such entity-sets
- Example: bank account categories – Checking accounts – Savings accounts – Have common features, but also unique attributes

Bank Account Example

- Checking and savings accounts have:
  – Account number
  – Balance
  – Owner

• Checking accounts also have:

– Overdraft limit and associated account

– Check transactions

• Savings accounts also have:

– Minimum balance

Aggregation

An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.

In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.

For Example,

Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY.

Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationships REQUIRE is created between aggregated entity and MACHINERY.

3-4 Live Examples to be discussed in Class

This concludes the second chapter where we learnt about the use of ER model and how it is similar to Flow Charts, now we will switch to R-DBMS Model, which is in detail use of Relational Model on Next Sunday, after which you should be clear with basic concepts of DBMS and feature that we will start using in practical.

Feel free to reach out to me in case of any queries