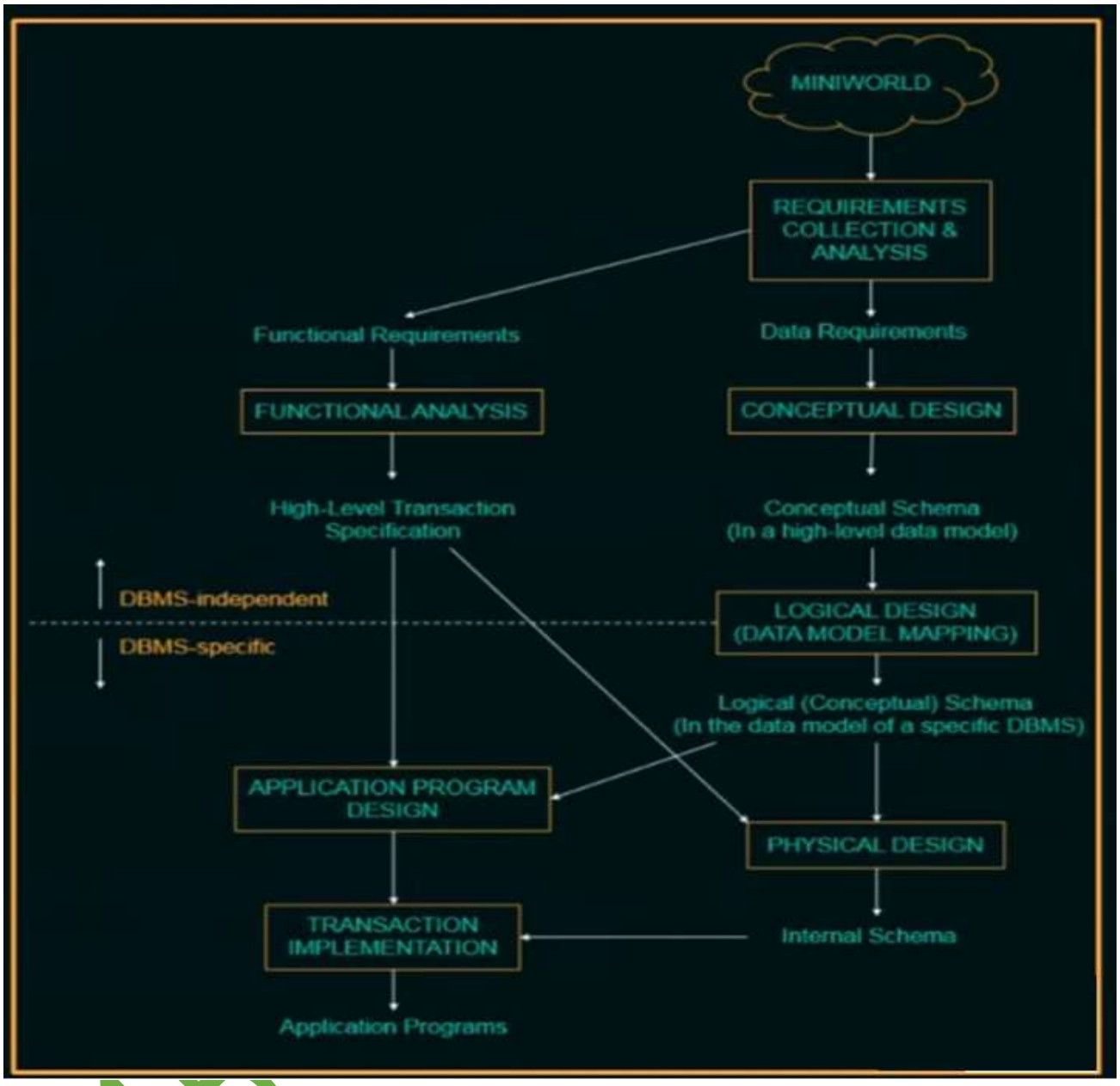
INTRODUCTION TO ER MODEL



Database Design

Requirements Collection & Analysis:

- Database Database designers understand & document the data requirements of the database users.

Functional Requirements:

Consists of user-defined operations.

Conceptual Design:

- Creating conceptual schema.

Conceptual Schema:

Concise description of the data requirements & detailed description of the entity types, relationships & constraints.

Logical Design:

Actual implementation of the database, using commercial DBMS.

Physical Design:

The internal storage structures, indexes, access paths specified.

The Entity Relationship Model

- ER model was introduced by Peter Chenn in 1976
- The ER model defines the conceptual (or logical) view of a database.
- It is used for designing database.
- It works around real-world entities and the relationships among them.
- A database schema in the ER Model can be represented pictorially as ER diagrams.

Why ER Model is Useful?

- An ER Model maps well to the relational model i.e., the constructs used in ER Model can be easily transformed into relational tables.
- An ER Model can be used by the database designer to communicate the database design to the user.
- An ER Model can be used as a design plan by the database developer to implement a data model in specific DBMS software.
- ER Diagram gives a better understanding of the information to be stored in a database.
- It allows users to get a preview of the logical structure of the database.

ER Model Constructs

1. Entities
2. Attributes
3. Relationships

1.Entity

- An Entity is a "thing" or ' 'object" in the real world that is distinguishable from other objects.
- Entity can be anything that has an independent existence and about which we collect data. It is also known as entity type.
  - Example: In a school database, the students, teachers, classes, courses or projects can be taken as an entity
- Entities are represented by means of rectangles.

- Entities have attributes that give them their identity.
  - Example: Students have roll no., names, and addresses
- Entities becomes table in relational model.

Entity Set

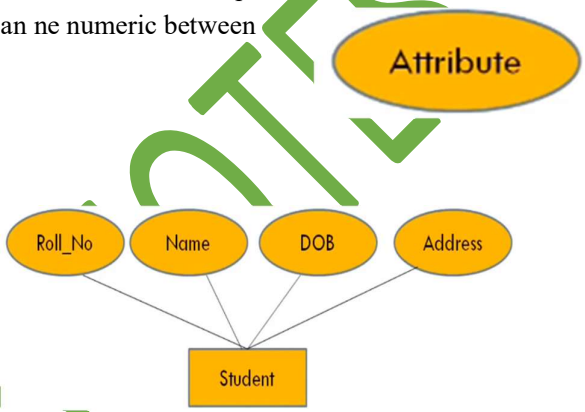
- An Entity Set is a collection of similar type of entities that share same attributes.
  - Example: a student's set may contain all the students of a school.
    a teachers set may contain all the teachers of a school from all faculties.
- **Entity sets need not be disjoint.**
- Example: the entity set Employee (all employees of a bank) and the entity set Customer (All customers of the bank) may have members in common.

Entities are of two types:

1. **Strong Entity** – A strong entity is an entity type that has a key attribute. It doesn't depend on other entities in the schema. A strong entity always has a primary key, and it is represented by a single rectangle in the ER diagram. **Example** – roll_number identifies each student of the organization uniquely and hence; we can say that the student is a strong entity type.
2. **Weak Entity** – Weak entity type doesn't have a key attribute and so we cannot uniquely identify them by their attributes alone. Therefore, a foreign key must be used in combination with its attributes to create a primary key. They are called Weak entity types because they can't be identified on their own. It relies on another powerful entity for its unique identity. A weak entity is represented by a double-outlined rectangle in ER diagrams.

2.Attributes

- An entity is represented by a set of attributes
- Attributes are Used to describe the property of an entity
    - Example: a Student entity may have Roll_No, Name, DOB, Age, Address, Mobile No as attribute
- For each attribute there is a set of permitted values, called domain (or range) of that attribute
    - Example: a student's name cannot be a numeric value. It has to be alphabetic.
- A student's age cannot be negative, etc. A student roll_no can ne numeric between some range like (0-10000)
- Attributes are represented by Ellipse



Examples

Schema (Entity type):

Student (Roll _ No, Name, DOB, Address)

Entity 1: → 101, Sharma, 12-10-2005, Delhi

Entity 2: → 102, Seema Singh, 15-12-2007, Jaipur

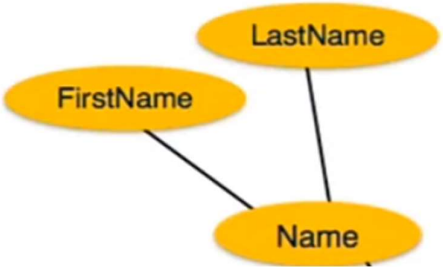
Attribute Types

1. Simple and Composite attributes
2. Single-valued and multi-valued attributes
3. Stored and Derived attributes
4. Key Attribute

Simple attributes

- Simple attributes are atomic values, which cannot be divided further.
- For example:
    - a student's mobile number is an atomic value of 10 digits.
    - a Birth Date is an atomic value of date-month-year.

Composite attributes



- Composite attributes are made of more than one simple attribute.
- A composite attribute is divided in a tree like structure.
- For example:
    - A student's complete name may have first _ name and last_name.
    - An address may have street, city, state, country, and pin code.

Single-valued and multi-valued attributes

- Single-value attributes contain single value.
    - For example: → Social _ Security _ Number, Aadhar_card_no, Roll _ no

Multi-valued attribute

- Multi-valued attributes may contain more than one values.
- Represented by double-ellipse.
- For example: a person can have more than one phone number, email address, etc.

Stored and Derived attributes

- Stored attributes are physically stored in the database.
- Mostly all attributes are stored in database except few ones.
- For example: Roll_no, Name, hirth_date, phone_no

Derived attribute

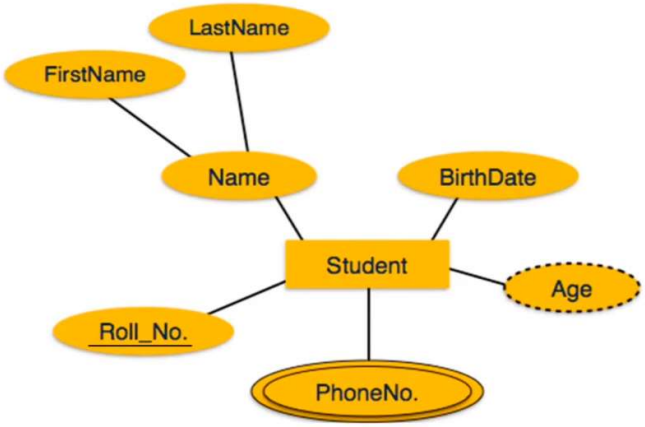
- Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
- Derived attributes are depicted by dashed ellipse.
- For example:
  - age can be derived from data_of_birth.
  - average_salary in a department should not be saved directly in the database, instead it can be derived.

Key Attribute

Key Attribute: The attribute which uniquely identifies each entity in the entity set is called key attribute.

- It represents a primary key.
- Key attribute is represented by an ellipse with underlying lines.
- For example: Roll_No. will be unique for each student.

E-R diagram of Student entity type with its attributes can be represented as:

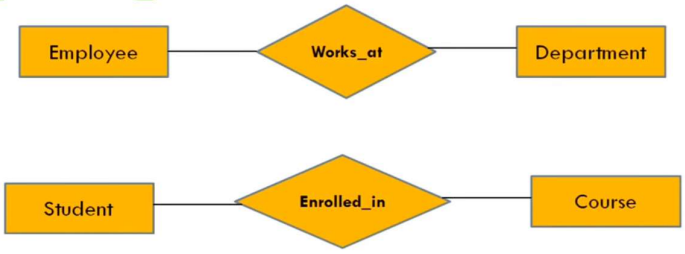


3.Relationships

- A Relationship is an association among entities.
  - For example:
    - an employee works_at a department
    - a student enrolls in a course.

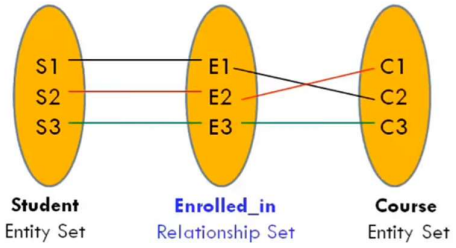
  Here, Works_at and enrolls are called relationships.
- Relationships are represented by diamond-shaped box.

Example



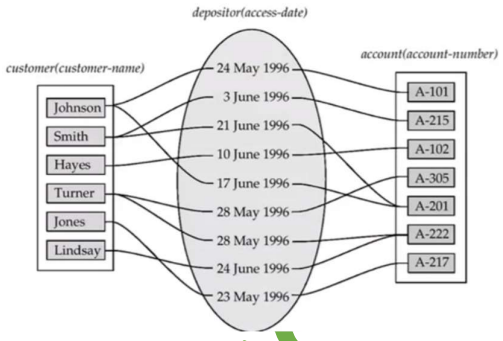
Relationship Set:

- A set of relationships of similar type is called relationship Set
- The following relationship set Enrolls (E 1, E2, E3) depicts:
- S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3



Relationship: Descriptive Attribute

- Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.
- For instance, the depositor relationship set between entity sets customer and account ma have the attribute access-date.

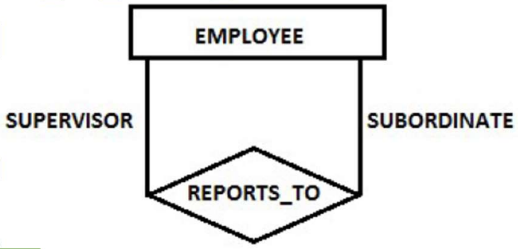


Degree of Relationship:

- The number of different entity sets participating in a relationship set is called as degree of a relationship set.
  - Unary
  - Binary
  - Ternary
  - N-ary

Unary Relationship: (degree = 1)

- A unary relationship is only one entity participate in a relationship; the relationship is called as unary relationship.
- Example:

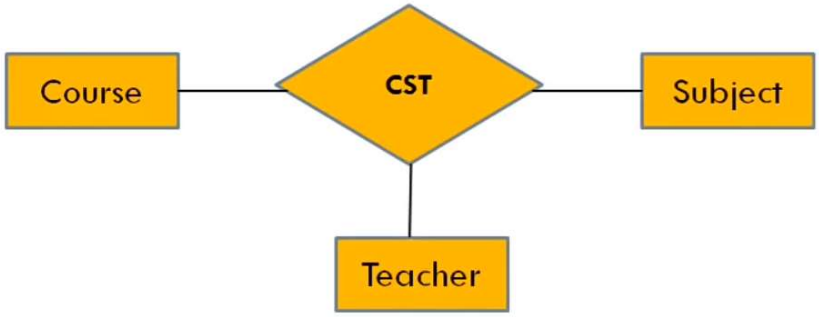


Binary Relationship: (degree = 2)

- A binary relationship is when two entities participate in a relationship and is the most common relationship degree.
- For example, Student is enrolled in Course.

Ternary Relationship: (degree=3)

- A ternary relationship is when three entities participate in the relationship.
- For example, The University might need to record which teachers taught which subjects in which courses.



n - ary Relationship: (degree=n)

- When there are n entities set participating in a relation, the relationship is called as n-ary relationship

Mapping Cardinalities (Cardinality Ratio)

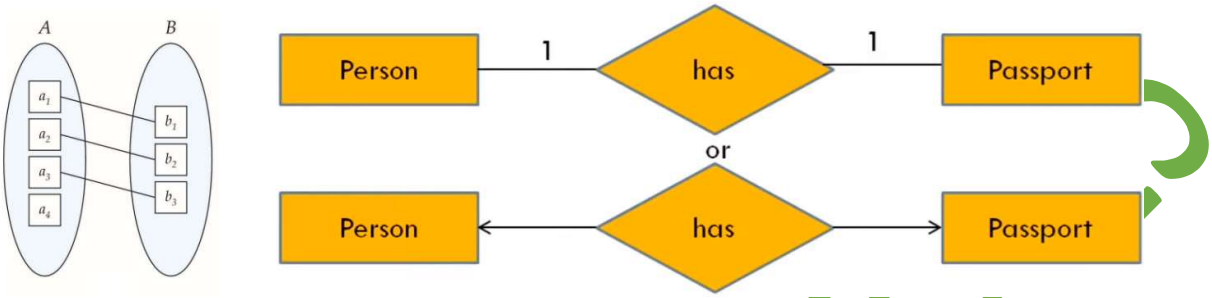
- Cardinality defines the number of entities of an entity set participates in a relationship set.
- Most Useful in describing binary relationship.

- Cardinality can be of different types or there are four types of relationships:

1. One-to-One (1-1)
2. One-to-Many (1 -M)
3. Many-to-One (M-l)
4. Many-to-Many (M-N)

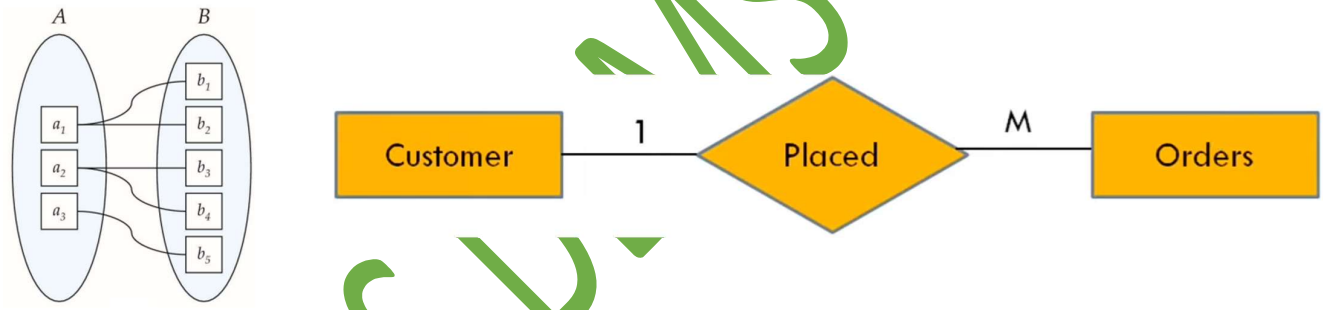
One-to-One (1-1) Relationship:

- One entity from entity set A can be associated with at most one entity of entity set B and vice versa.
  - For example, a person has only one passport and a passport is given to one person.



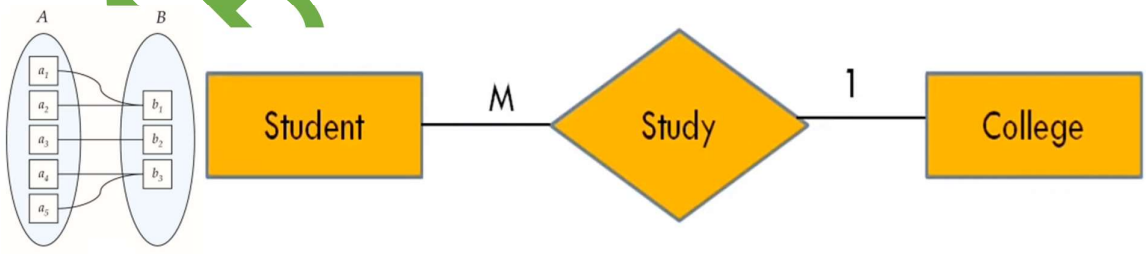
One-to-Many (1 -M) Relationship:

- One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity
- For example — a customer can place many orders but a order cannot be placed by many customers.



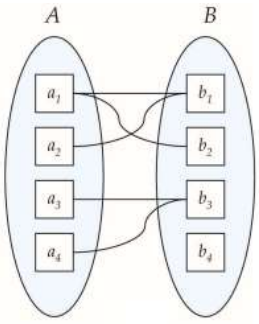
Many-to-One (M -1) Relationship:

- More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A
- For example — many students can study in a single college, but a student cannot study in many colleges at the same time.



Many-to-Many (M -M) Relationship:

- One entity from A can be associated with more than one entity from B and vice versa.
- For example, an Employee can be assigned to many Projects and a Project can have many Employee.



How to Choose Relationships?

- The appropriate mapping cardinality for a particular relationship set depends on the real world situation being modelled.

Consider Borrower relationship set in a Bank.

- If in a particular bank, a loan can belong to only one customer and a customer can have only one loans then the relationship set from customer to loan is 1:1

- If a loan can belong to only one customer and a can have several loans then the relationship set from customer to loan is 1:M

- If a loan can belong to several customers and a customer can have only one loan, then the relationship set from customer to loan is M:1

- If a loan can belong to several customers (as loans can be taken jointly by several business partners) then the relationship set is M:N

Participation Constraint:

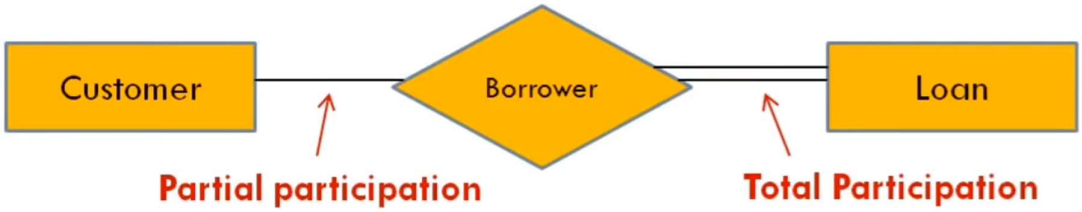
- Participation Constraint is applied on the entity participating in the relationship set.
  - Total Participation
  - Partial Participation

Total Participation

- Each entity is involved in the relationship. Total participation is represented by double lines.
  - E.g., participation of loan in borrower is total.
    - every loan must have a customer associated to it via borrower.

Partial participation

- Not all entities are involved in the relationship.
  - Partial participation is represented by single lines.
  - participation of customer in borrower is partial.
    - A customer may have no loans.



KEYS in DBMS

- Definition: A key is an attribute or set of attributes that uniquely identifies any record (or tuple) from the table.
- Purpose: a Key is used to uniquely identify any record or row of data from the table.
- It is also used to establish and identify relationships between tables.

Types of Keys

- Super Key
- Candidate Key
- Primary Key
- Alternate Key
- Foreign Key
- Composite Key

1. Super Key
   - A super key is a combination of all possible attributes that can uniquely identify the rows (or tuple) in the given relation.
     - Super key is a superset of a candidate key.
     - A table can have many super keys.
     - A super key may have additional attribute that are not needed for unique identity.

Emp_Id	Name	Aadhar_No	Email_Id	Dept_Id
01	Aman	775762540011	aa@gmail.com	1
02	Neha	876834788522	nn@gmail.com	2
03	Neha	996677898677	ss@gmail.com	2
04	Vimal	796454638800	vv@gmail.com	3

- Super Keys:

1. {Emp_Id} 2. {Aadhar _ No}3. {Email_Id} 4. {Emp_Id, Aadhar_No} 5. {Aadhar _ No, Email_Id}
6. {Emp_Id ,Email_Id} 7. {Emp_Id, Aadhar_No, Email_Id} 8. {Emp_Id, Name} 9. {Emp_id, Name, Dep_Id}
10. Aaadhar_No, Email_Id, Dept_Id}

2. Candidate Key
   - A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
   - A candidate key is a minimal super key; or a Super key with no redundant attributes.
     - It is called a minimal super key because we select a candidate key from a set of super key such that selected candidate key is the minimum attribute required to uniquely identify the table
   - Candidate keys are defined as distinct set of attributes from which primary key can be selected not allowed to have NULL values.

3. Primary Key
   - A primary key is one of the candidate key chosen by the database designer to uniquely identify the tuple in the relation.
     - The value of primary key can never be NULL.
     - The value of primary key must always be unique (not duplicate).

- The values of primary key can never be changed i.e., no updation is possible.
- The value of primary key must be assigned when inserting a record.
- A relation is allowed to have only one primary key.

4. Alternate Keys
   - Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate keys.
   - In the Employee table
     - Emp_Id is best suited for the primary key.
     - Rest of the attributes like Aadhar No, Email Id are considered as a alternate keys.

5. Foreign keys
   - A Foreign Key is:
     - A used to link two tables together.
     - An attribute (or set of attributes) in one table that refers to the Primary Key in another table.
   - The purpose of the foreign key is:
     - to ensure (or maintain) referential integrity of the data.

Foreign Key:

- Foreign key references the primary key Of the table
- Foreign key can take only those values which are present in the primary key of the referenced relation.
- Foreign key may have a name other than that of a primary key.
- Foreign key can take the NULL value.
- There is no restriction on a foreign key to be unique.
- In fact, foreign key is not Unique most of the time.
- Referenced relation may also be called as the master table or table.
- Referencing relation may also be called as the foreign table.

6. Composite Key
   - A key that has more than one attributes is known as composite key. It is also known as compound key.

Cust_Id	Order_Id	Product_Code	Product_Count
C01	001	P111	5
C02	012	P111	8
C02	012	P222	6
C01	001	P333	9

   - Composite Key:
     - {Cust_Id, Product _ Code}

Extended Entity Relationship (ER) Features:

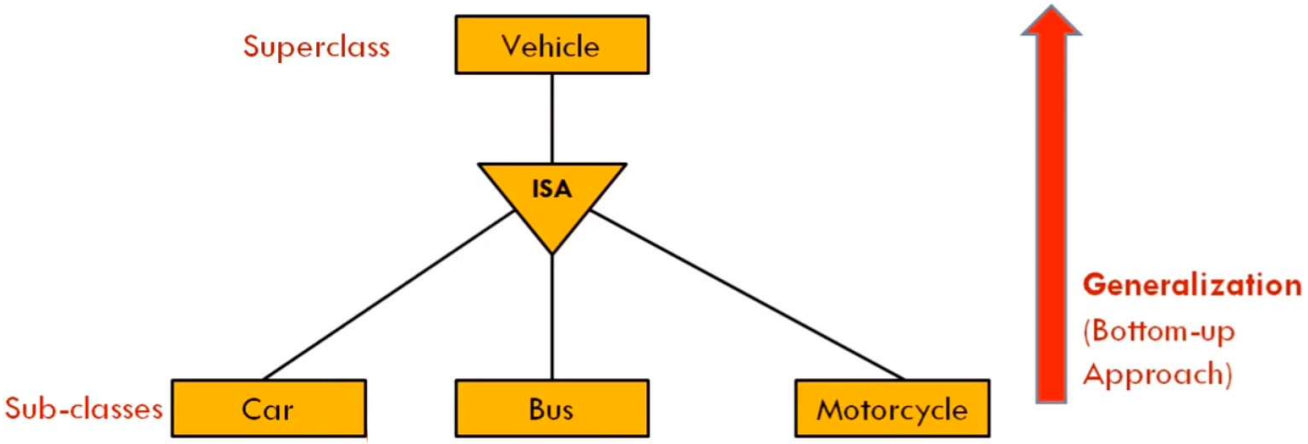
- As the complexity of data increased in the late 1 980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.
- Hence, as part of the Extended ER Model, along with other improvements, three new concepts were added to the existing ER Model:
  1. Generalization
  2. Specialization
  3. Aggregation

Generalization

- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.
- Generalization is a "bottom-up approach" in which two or more entities can be combined to form a higher-level entity if they have some attributes in common.
  - subclasses are combined to make a superclass.
- Generalization is used to emphasize the similarities among lower-level entity set and to hide differences in the schema.

Example: Generalization

- Consider we have 3 sub entities Car, BUS and Motorcycle. Now these three entities can be generalized into one higher-level entity (or super class) named as Vehicle.

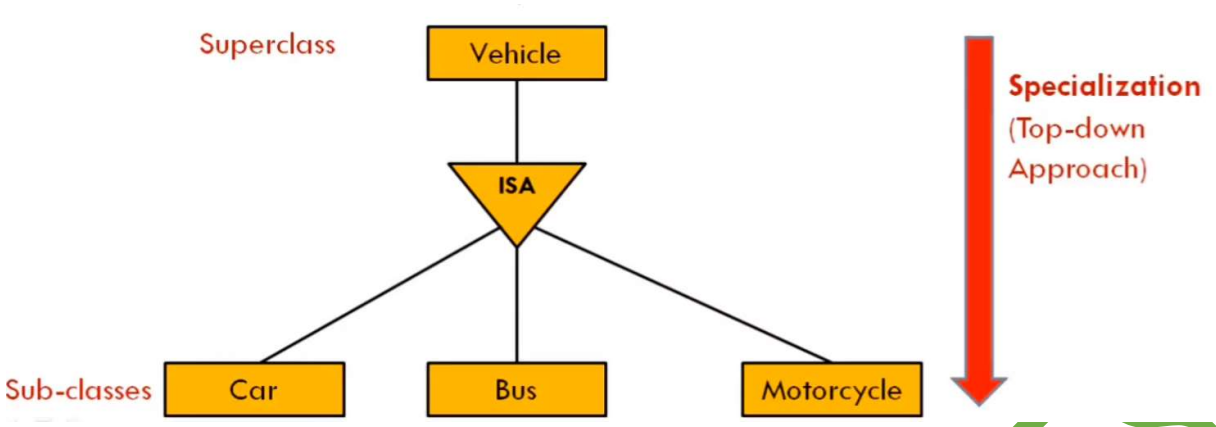


Specialization

- Specialization is opposite of Generalization In Specialization, an entity is broken down into sub-entities based on their characteristics.
- Specialization is a "Top-down approach" where higher level entity is specialized into two or more lower-level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Specialization can be repeatedly applied to refine the design of schema.
- Depicted by triangle component labelled as ISA.

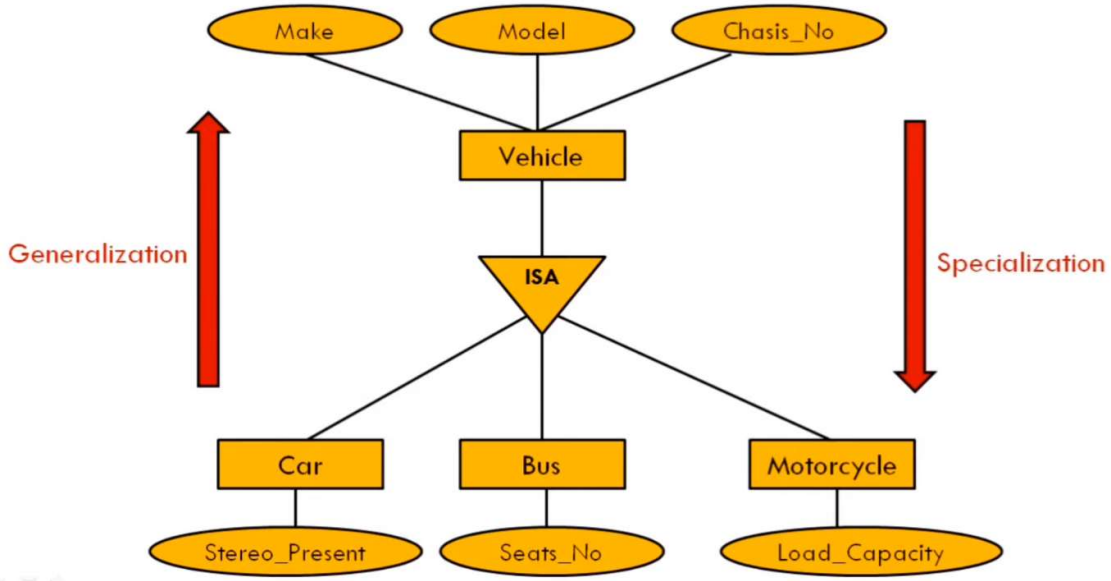
Example: Specialization

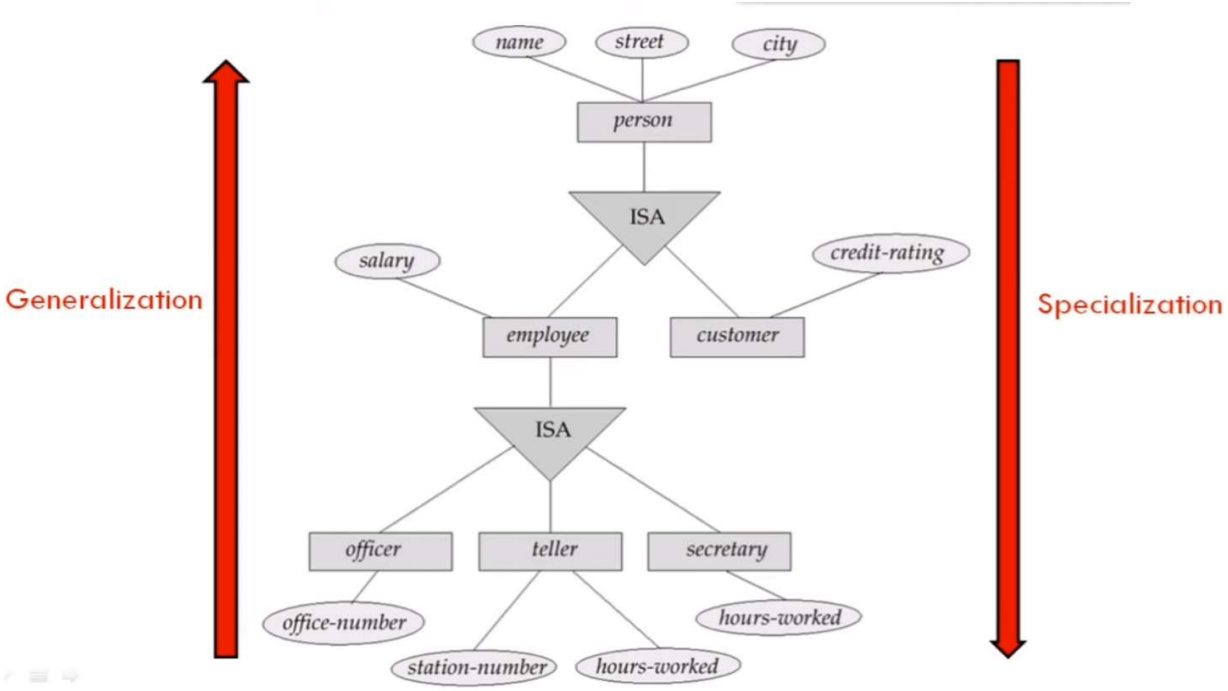
- Vehicle entity can be a Car, Truck or Motorcycle.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.



Inheritance:

- Inheritance is an important feature of generalization and specialization.
- **Attribute inheritance** allows lower-level entities to inherit the attributes of higher-level entities.
    - For example, consider relations Car and Bus inheriting the attributes of Vehicle. Thus, Car is described by attributes of super-class Vehicle as well as its own attributes.
- This also extends to **Participation Inheritance** in which relationships involving higher-level entity-sets are also inherited by lower-level entity-sets.
    - A lower-level entity-set can participate in its own relationship-sets, too.





How Schema or Tables can be formed?

Four tables can be formed:

1. customer (name, street, city, credit _ rating)
2. officer (name, street, city, salary, office_number)
3. teller (name, street, city, salary, station_number, hours _ worked)
4. secretary (name, street, city, salary, hours_worked)

Constraints on specialization and generalization

There are three constraints that may apply to specialization/generalization which are as follows:

- Membership constraints
  - Condition-defined membership constraint
  - User-defined membership constraint
- Disjoint constraints
  - Disjoint constraint
  - Overlapping constraint
- Completeness constraints
  - Total completeness constraint
  - Partial completeness constraint

Membership constraints:

Condition-defined member constraint

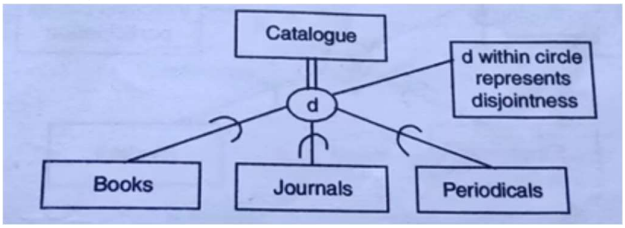
- In condition-defined lower-level entity sets, membership is evaluated based on whether or not an entity satisfies an explicit condition or predicate.
- For example, if the higher level-entity set employee has an attribute job_type, all entities that satisfy the condition job_type= "secretary" is included in secretary, and job_type= "instructor" is included in instructor.
- Since all the lower-level entities are evaluated based on the same attribute (job_type), this type of generalization is said to be attribute-defined.

User-defined member constraint

- User-defined lower-level entity sets are not considered by a membership condition; rather, the database user assigns entities to a given entity set.
- For example, after 3 months of employment, if university employees are supposed to be assigned to one of four work teams, we represent the teams as four lower-level entity sets of the higher-level employee entity set.

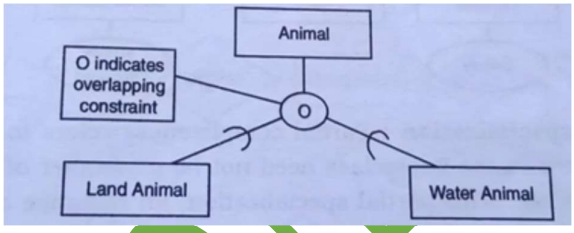
- A given employee is not assigned to a specific team entity automatically based on an explicit defining condition.
- Instead, the user in charge of this decision makes the team assignment on an individual basis.

Disjoint constraints



Disjoint constraints

- It refers that an entity belongs to no more than one lower-level entity set i.e, it specifies that the subclass of the specialization must be disjoint.



Overlapping constraints

- It refers to that the same entity may belong to more than one lower-level entity set i.e., it specifies that the subclasses are not constrained to be disjoint.

Completeness constraint:

Total completeness constraint

- Each higher-level entity must belong to a lower-level entity set.
- It is represented by a double line in the EER diagram.

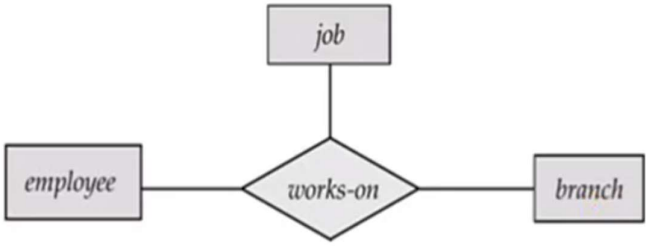
Partial completeness constraint

- Some higher-level entities may not belong to any lower-level entity set.
- It is represented by a single line in the EER diagram.

Aggregation:

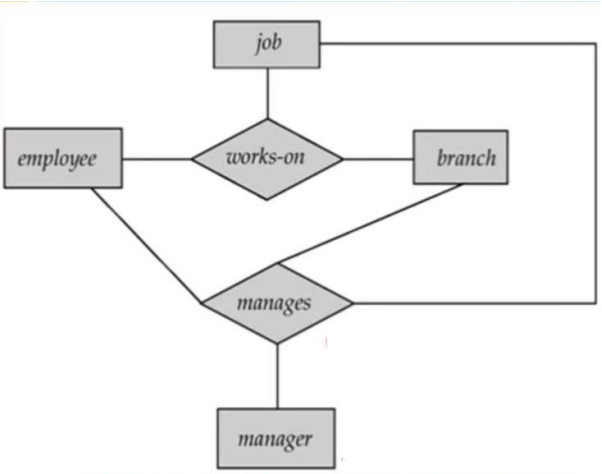
- Aggregation is used when we need to express a relationship among relationships.
- Aggregation is an abstraction through which relationships are treated as higher level entities.
- Aggregation is a process when a relationship between two entities is considered as a single entity and again this single
- entity has a relationship with another entity.

Example: (Relationship of Relations)



- Basic E-R model can't represent relationships involving other relationships.
- Consider a ternary relationship works_on between Employee, Branch and Job.
- Am employee works on a particular job at a particular branch.
- Suppose we want to assign a manager for Jobs performed by an employee at a branch (i.e. want to assign managers to each employee, job, branch combination)
- Need a separate manager entity-set a
- Relationship between each manager, employee branch and -job entity.

Example: (Redundant Relationship)

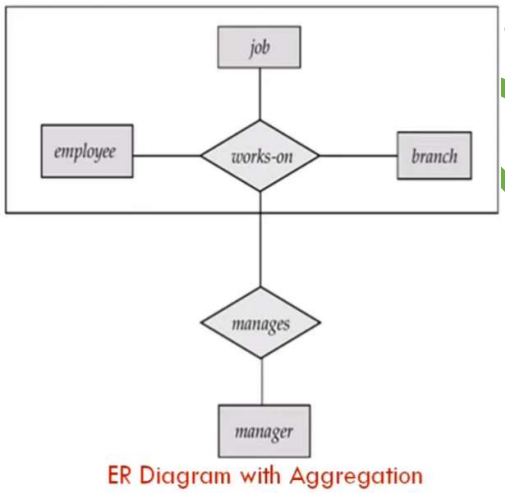


ER Diagram with Redundant Relationship

- Relationship sets works-on and manages represent overlapping (redundant) information.
  - Every manages relationship corresponds to a works-on relationship.
  - However, some works-on relationships may not correspond to any manages relationships.
    - So we can't discard the works-relationship

Aggregation:

- Eliminate this redundancy via aggregation.
  - Treat relationship as an abstract entity a
  - Allows relationships between relationships.
  - Abstraction of relationship into new entity.



ER Diagram with Aggregation

With Aggregation (without introducing redundancy) the ER diagram can be represented as:

- An employee works on a particular job at a particular branch.
- An employee, branch, job combination
- may have an associated manager.

E-R Design Principles

- Faithfulness
  - **Entities, attributes, and relationships should reflect reality.**
  - **Sometimes the correct approach is not obvious,**
    - E.g., course and instructor entities and teaching relationship
    - What are the cardinality constraints? It depends on…
- Avoiding Redundancy
  - **No information should be repeated.**
- Simplicity
  - **Some relationships may be unnecessary.**
- Choosing the right kind of element
  - **The use of an attribute or entity set to represent an object.**
  - **Whether a real-world concept is best expressed by an entity set or a relationship set**

- Choosing the right relationships
  - The use of a ternary relationship versus a pair of binary relationships
  - The use of a strong or weak entity set.
  - The use of specialization/generalization – contributes to modularity in the design.
  - The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

Conceptual Design Using the ER Model

❖ Design choices:

➢ Should a concept be modelled as an **entity or an attribute?**

- How to consider your data property as Entity or Attribute?
- Depends on data description.
- Example: Address is our data property
  - Single Address
  - Muliple addresses to a single person.
  - Records like city, state, country, zipcode.

➢ Should a concept be modelled as an **entity or a relationship?**

- Binary
- Ternary (Ternary Relationship versus Binary Relationship)
- one to one
- one to many
- many to one.
- aggregation versus Ternary Relationships

➢ Identifying relationships: **Binary or ternary? Aggregation?**

- Aggregation versus Ternary Relationship:
- Departments Produces Projects
- Departments Produces Projects and manages by employees.
- Departments Produces Projects and manages by employees until some date and time

Conceptual Design for Large Database

- Designing databases for large organizations takes the effort of more than a single designer. It diagrammatically represents the complete database and enables the user who provides inputs to database, to understand the complete functionality of database.
- Large databases are modelled in two methodologies.
  - The requirements of all the users are collected. The conflicting requirements are resolved, and a final conceptual view is generated to satisfy the requirements of all users.
  - In the other method, the user provides his requirements; the designer generates a conceptual view for the requirements. Likewise, all the conceptual views from all user requirements are generated and a comprehensive conceptual view that satisfies all the requirements is generated.

How to Draw ER Diagram?

We have read all the basic terms of E-R Diagram. Now, let's understand how to draw E-R diagram? In ER Model, objects of similar structures are collected into an **entity set**. The relationships between an entity sets is represented by a named **E-R relationship**, which may be **(one-to-one, one-to-many, many-to-one, many-to-many)**, which maps one entity set to another entity set.

Steps - How to Draw ER Diagram -

1. Identify all the entities of the given problem.
2. Identify all the attributes of the entities identified in step 1.
3. Identify the Primary Keys of entities identified in Step 1.

4. Identify the Attribute Types of attributes identified in step 2
5. Identify relationship between the entities and constraints on the entities and implement them.

Need of ER Diagram –

The ER Diagrams are useful in representing the relationship among entities. It helps to show basic data structures in a way that different people can understand. Many types of people are involved in the database environment, including programmers, designers, managers, and end users. But not all of these people work with database and might not be as skilled as others to understand the making of a software or a program etc, so, a conceptual model like the ERD helps show the design to many different people in a way they can all understand.

CASE STUDY:

ER Diagram for Banking Enterprise

- The bank is organized into branches. Each branch is located in a particular city and is identified by a unique name. The bank monitors the assets of each branch.
- Bank customers are identified by their customer id values. The bank stores each customer's name and the street and city where the customer lives. Customers may have accounts and can take out loans. A customer may be associated with a particular bank employee, who may act as a loan officer or personal banker for that customer.
- Bank employees are identified by their employee id values. The bank administration stores the name and telephone number of each employee, the names of the employee's dependents, and the employee id number of the employee's manager. The bank also keeps the track of the employee's start date and, thus, length of employment.
- The bank offers two types of accounts - savings and checking accounts. Accounts can be held by more than one customer, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of account's balance and the most recent date on which the account was accessed by each customer holding the account. In addition, each savings account has. m interest rate and overdrafts are recorded for each checking account.
- A loan originates at a particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount and the loan payments. Although a loan payment number does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a particular payment for a specific loan. The date and amount are recorded for each payment.

1. **Identifying the ENTITIES**

   Banking Database consists of 6 entities:
   - Branch
   - Customer
   - Employee
   - Account
   - Loan
   - Payment (it is a weak entity depends on Loan entity)

2. **Identify the ATTRIBUTES**
   - ✓ branch: branch-name, branch-city, assets
   - ✓ customer: customer-id, customer-name, customer-street, customer-city
   - ✓ employee: employee-id, employee-name, telephone-number, start-date, dependent-name, employment-length
   - ✓ account: account-number, balance
   - ✓ loan: loan-number, amount
   - ✓ payment: payment-number, payment-date, payment-amount

3. **Identify the PRIMARY KEY**

Entity	Primary Key
Branch	Branch_name

4. **Identify the RELATIONSHIPS and constraints.**

A customer can have multiple accounts and an account can be held by multiple customers. Cardinality will be M: N

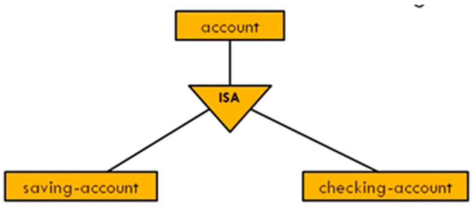
Customers are allowed to take loans from the bank, a loan can be held by one or more customers. Cardinality will be M: N

Each loan is paid back with the multiple having payment number. Cardinality will be 1: N

A customer can be an employee of the bank. Cardinality will be M: N

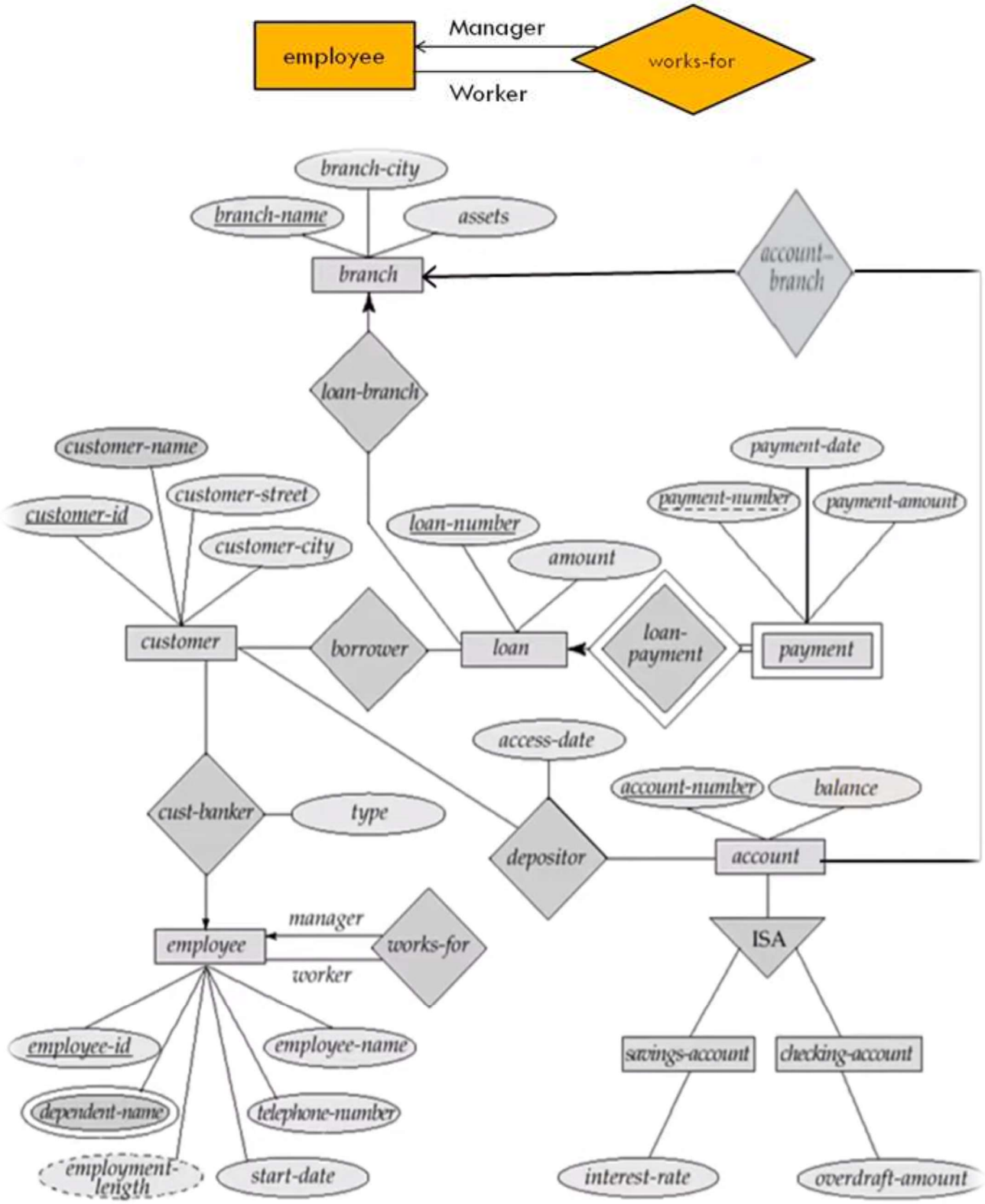
A branch can have multiple accounts. Cardinality will be 1: N.

An account can be divided into two accounts: saving account and checking account.



A branch of bank give loans to the customers. Cardinality will be 1: N

Employees can be a manger or a worker of that particular branch.(Recursive Relationship exist)


employee
Manager
works-for
Worker
branch-city
branch-name
assets
branch
account-branch
loan-branch
customer-name
customer-id
customer-street
customer-city
loan-number
amount
payment-date
payment-number
payment-amount
customer
borrower
loan
loan-payment
payment
access-date
account-number
balance
cust-banker
type
depositor
account
manager
employee
works-for
worker
ISA
employee-id
employee-name
dependent-name
telephone-number
employment-length
start-date
savings-account
checking-account
interest-rate
overdraft-amount