

Debere Berhan Polytechnic College

Database management service Level iii

MODULE TITLE: **Testing Physical Database Implementation**

LO1: **Undertake database management system modelling**

1.1 **Using Entity relationship Modelling High level coding data Model**

1.1.1. **Entity types, entity sets, attributes, and sets**

What is **entity** in a database?

- * **An Entity** is a thing in real-world (an object with a physical existence, e.g. A particular person, car, house, or employee or an object with a conceptual existence.
 - It can be roles, events, locations, tangible things or concepts.

Example: A person/an employee, department, payment, book, project, campus, organization, or business.
 - Each entity must have its own identity that distinguishes it from every other entity, called **unique identifier**.

Example: Each Student has a unique ID that distinguishes one from others.

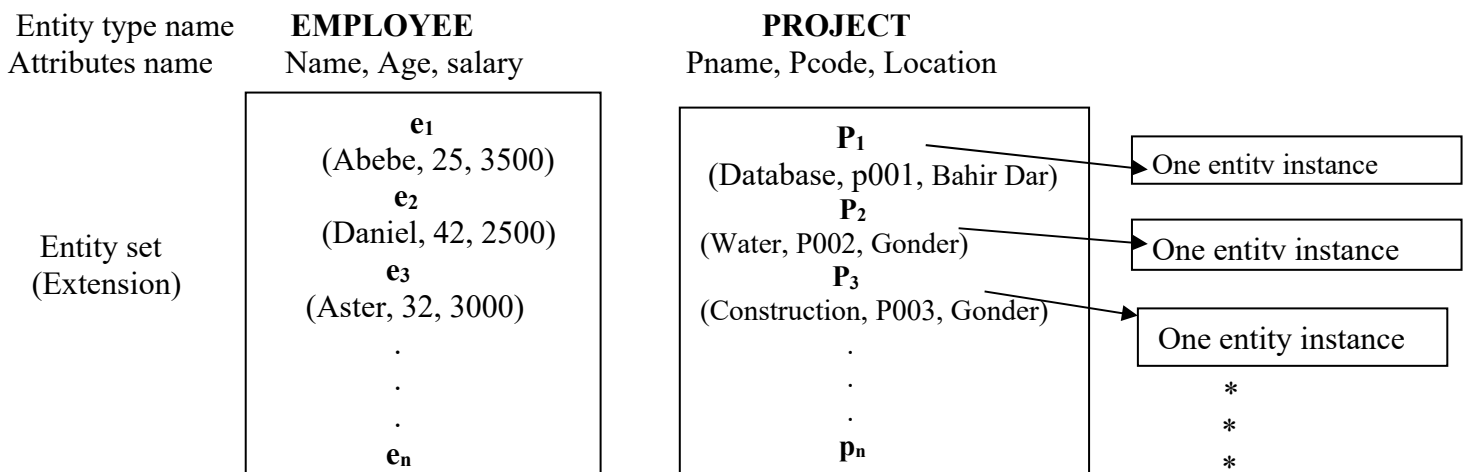
In database terms, an **entity** is a **table** which is responsible for storing data in the database.

- * **Entity type** is the set of all possible values for an entity is called **entity type**.

Example: - student is an entity type with common attributes to all students such as student_ID, Name, etc.
- * An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).
 - An **entity set** is the collection of instances represented by (e1, e2, e3, en).
 - It is also called an **extension** of the entity type

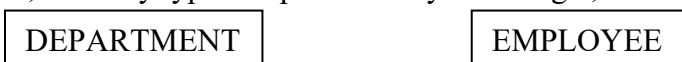
Example: - a company have many employees, and these employees are defined as entities (e1, e2, e3,) and all these entities having the same attributes which are defined under ENTITY TYPE employee, and set{e1, e2,} is called **entity set**.
- * **Entity Instance** is a single occurrence of a particular entity type. An entity type is described just one time in the data model, but many instances of that entity type may be represented by data stored in the database.

Example: - There may be hundreds or thousands of employees in an organization – each one is an instance of the Employee entity type.



In E-R Diagram, an entity type is represented by a rectangle, and the name is indicated in capital letters.

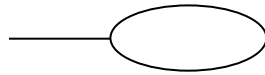
Example:



- * **Attributes**
 - An attribute is a property that describes an entity.

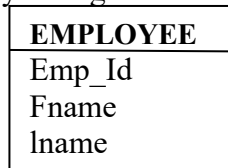
- Each attributes has a particular value based on the defined data type.
- The set of all possible (allowable) values of an attribute is called **attribute domain**.
- In E-R Diagram, attributes are named with an initial capital letter followed by lowercase letters.
- An attribute can be represented by an ellipse (oval) shape with a line connected it to the associated entity.

Example:



- It can also be represented by listing them within the entity rectangle, under the entity name(entity type).

Example:



✓ Types of attributes

An attribute can be: - simple or composite, single-value or multi-value, stored or derived or Null Values.

1) Simple (Atomic) vs Composite attribute

- Simple attributes cannot be further divisible.

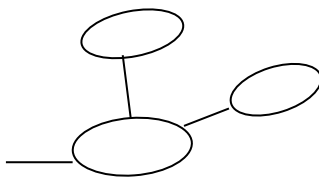
Example: SSN, Sex

- In E-R Diagram, simple attributes can be represented by an ellipse shape. Example: 

- Composite attributes can be divided into smaller subparts.

Example: - address (street address, city, state, zip code)

- Name (first name, last name, initial name)

- In E-R Diagram, composite attributes can be represented by: 
- The value of composite attribute is the concatenation of the values of its constituent simple attributes.

2) Single-valued vs Multi-valued attributes

- Most attributes have a single value for a particular entity.

- In some cases, an attribute can have a set of value for the same entity, called **multi-valued**.

Example: colour attribute for a car, college degree for a person.

- In E-R Diagram, **Multi-valued** attributes can be represented by double ellipse (oval) shape. 

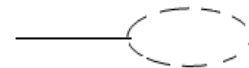
3) Stored vs derived attributes.

- In some cases, some attributes have a reference (derived from other attributes' value).

Example: - The **Age** attribute is derived from the **Birth-Date** attribute.

- So, the **Age** attribute is **derived** attribute, and the **Birth-Date** attribute is **stored** attribute.

In E-R Diagram, **Derived** attributes can be represented by dotted ellipse.



4) Null Values attributes

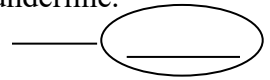
- Not applicable: e.g. Apartment Number, College Degree.
- Unknown: - the attribute value exists but is missing. E.g. Height
 - It is not known whether the value exists. E.g. Home Phone

✓ Key attributes

- Attributes can be classified as identifiers or descriptors.
- A **descriptor** describes a non-unique characteristic of an entity instance.
- **Identifiers** (more commonly called keys or key attributes) uniquely identify each instance of an entity, called **candidate key**. If such an attribute doesn't exist naturally, a new attribute is defined for that purpose.

Example: an ID number of **STUDENT** OR **EMPLOYEE** Entity type.
- In some cases, more than one attribute is required to identify a unique entity, called **composite candidate key**.

- In E-R Diagram, key attributes (identifiers) can be represented by ellipse shape with underline.



When selecting identifiers, you should follow these guidelines:

- Choose a candidate key that has values that do not change over the life of each instance of the entity type.
- Choose a candidate key that is guaranteed to always have valid values and not be null for each instance.
- If a candidate key is a composite of two or more attributes, consider creating a new key with a single value.

✓ Types of Keys in Relational Database Model

- Super key
- Primary key
- Foreign key
- Candidate key
- Alternate key

Super Key is an attribute or a composite attribute which functionally determines all of the entity's attributes.

- **Since, a super key may contain additional columns that are not necessary for unique identification; we're interested in identifying super keys that contain only the minimum number of columns necessary for unique identification.**

Candidate Key is a super key whose values are not repeated in the table records.

- **It is a super key that contains only the minimum number of columns necessary for unique identification.**
- It is possible to have more than one choice of candidate key in a particular table. In that case, the selection of the primary key would be driven by the designer's choice or by end user requirements.

Primary Key is a candidate key which doesn't have repeated nor NULL values in the table.

- A primary key in any table is both a super key as well as a candidate key.
- It is an attribute or a set of attributes that uniquely identify a specific instance of an entity.
- Every entity in the data model must have a primary key whose values uniquely identify instances of the entity.

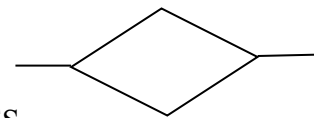
Alternate key is the **candidate keys that are not selected as the primary key of the entity.**

Foreign Key is a table's primary key attribute which is repeated in another related table (having related data) to maintain the required data relationship.

1.1.2. Relationship type, relationship sets, roles and structural constraints

A **relationship** is an association that exist b/n two or more participating entities.

- The function of relationship is to share data b/n entities.
- In E-R Diagram, a relationship type is represented by a diamond shape with the relationship verb in it.
- Some example of relationships are:
 - a DEPARTEMENT **has** EMPLOYEES
 - an EMPLOYEE **manages** PROJECTS
 - EMPLOYEES **works_on** PROJECTS
 - EMPLOYEES **process** PAYMENTS
 - EMPLOYEES **works_for** DEPARTEMENT



There are three types of relationships b/n entities

- **One-to-one:** - one instance of an entity (A) is associated with one other instance of another entity (B).
Example: - the **manages** relationship b/n department and employee
 - President to country, husband to wife, people to religion
- **One-to-many:** - one instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity (B) there is only one instance of entity (A).
Example: - the **has** relationship b/n department and employee
 - Country to people, mother to children, department to students
- **Many-to-many:** - one instance of an entity (A) is associated with zero, one or many instances of another entity (B).

(B), and one instance of entity (B) is associated with zero, one or many instances of entity (A).

Example: - the **works_for** relationship b/n employee and project

- Students to course, employee to projects, people to language

- **Relationship Types**

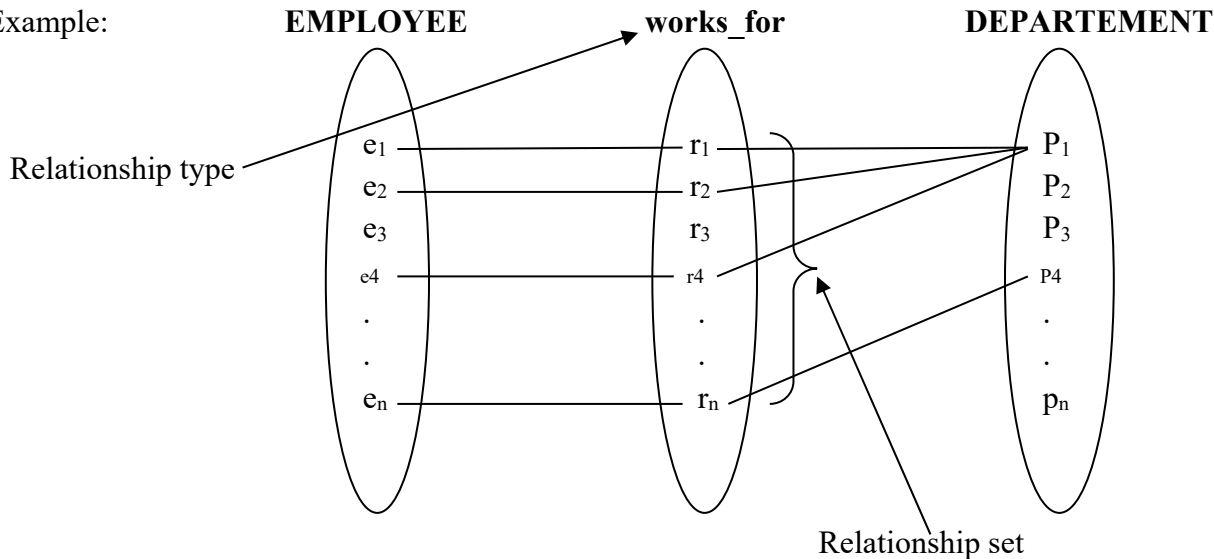
A Relationship Type defines a relationship set among entities of certain entity types.

A relationship type R among n entity types E_1, E_2, \dots, E_n defines a set of associations or a relationship set-among entity from these entity types.

- **Relationship sets**

A Relationship Set is a collection of relationships all belonging to one relationship type.

Example:



- **Roles and structural constraints**

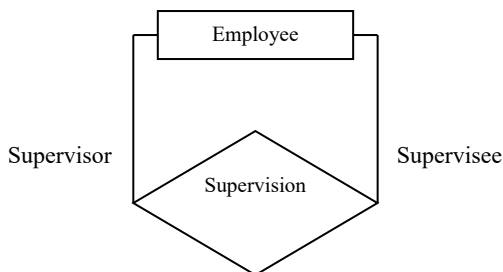
Role Names and Recursive Relationships.

- Each entity type in a relationship plays a particular role. The role name specifies the role that a participating entity type plays in the relationship and explains what the relationship means.

Example: in the relationship between Employee and Department (in the WORKS_FOR relationship type), EMPLOYEE entity type plays the role of *employee* or *worker* and DEPARTMENT entity type plays the role of *department* or *employer*.

- In most cases, the role names do not have to be specified, but in cases where the same entity participates more than once in a relationship type in different roles.

For example, each employee has a supervisor, we need to include role name “**Supervise**” and “**supervisor**”, both of them are employees. Since the employee entity type participates twice in the relationship, once as an employee and once as a supervisor, we can specify two roles, employee and supervisor.

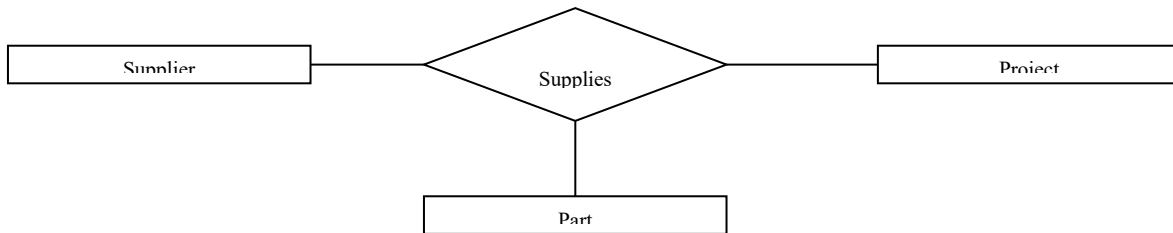


Degree of Relationship Type

- The degree of a relationship type is the number of participating entity types. Meaning if the relationship is between two entity types (for Example, Employee and Department), then the relationship is binary, or has a degree of two.

- If the relationship is between three participating entities, it has a degree of three, and therefore is a ternary relationship.

Example: The relationship b/n Supplier, Project and Part. Each part is supplied by a unique supplier, and is used for



a given project within a company; the relationship “Supplies” is a ternary (degree of three) between Supplier, Project and Part.

Constraints on Relationship Types

- Relationship types have certain constraints that limit the possible combination of entities that may participate in relationship.
- An example of a constraint on relationship is that if we have the entities Doctor and Patient, the organization may have a rule that a patient cannot be seen by more than one doctor. This constraint needs to be described in the schema.
- There are two main types of relationship constraints: cardinality ratio, and participation.

Cardinality for Binary Relationship

- Binary relationships are relationships between exactly two entities.
- The cardinality ratio specifies the maximum number of relationship instances that an entity can participate in.
- The possible cardinality ratios for binary relationship types are: 1:1, 1:N, N:1, M:N.
- Cardinality ratios are shown on ER diagrams by displaying 1, M and N on the diamonds.

Participation Constraints and Existence Dependencies

- The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- The constraint specifies the minimum number of relationship instances that each entity can participate in.
- There are two types of participation constraints:
 - **Total:**
 - If an entity can exist, only if it participates in at least one relationship instance, then that is called total participation.

Example: in the relationship between Employee and Department, if company policy states that every employee must work for a department, then an employee can exist only if he/she participates in at least one relationship instance (i.e. an employee can't exist without a department).

- It is also sometimes called an existence dependency.
- Total participation is represented by a double line.

- **Partial:**

- If only a part of the set of entities participate in a relationship, then it is called partial participation.
- Example: in the relationship between Employee and Department, if company policy states that every employee will not be a manager of a department, then the participation of an employee in the “Manages” relationship is partial.
- Partial participation is represented by a single line.

Attributes of Relationship Types

- Relationships can have attributes similar to entity types.

Example: - in the relationship Works_On, between the Employee entity and the Department entity, we would like to keep track of the number of hours an employee works on a project. Therefore we can include Number of Hours as an attribute of the relationship.

- In the “manages” relationship between employee and department, we can add Start Date as an attribute of the Manages relationship.

For some relationships (1:1, or 1: N), the attribute can be placed on one of the participating entity types.

Example: the “Manages” relationship is 1:1, so **StartDate** can either be migrated to Employee or Department.

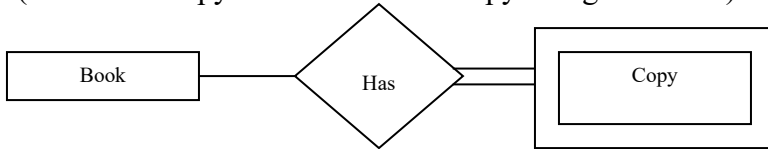
For some relationships (1:1, or 1:N), the attribute can be placed on one of the participating entity types.

Example: the “works_for” relationship is N:1, so **Since** can be migrated to Employee entity type.

1.1.3. Weak Entity Types

- Entity types that do not have key attributes (strong key) are called weak entity types.
- The relationship that relates the identifying entity type (strong entity type) with the weak entity type is called an identifying relationship.
- A weak entity type always has a total participation constraint with respect to the identifying relationship, because a weak entity cannot exist without its owner.
- A weak entity type usually has a partial key, which is the set of attributes that can uniquely identify weak entities.

Example: let’s assume in a library database, we have an entity type Book. For each book, we keep track of the author, and title. The library may own several copies of the same book, and for each copy, it keeps track of the copy number (a different copy number for each copy of a given book) and price of each copy.



The copy number is only unique for each book, meaning Book 123 may have copy 1, copy 2, copy 3, and book 456 may also have copy 1, copy 2 and copy 3. The copy number cannot be considered unique for each copy.

Therefore, the Copy entity does not have a key attribute; it is considered a weak entity type where as the book entity is the identifying entity.

- A copy cannot exist without the identifying entity (Book)
- The Copy entity type has a total participation constraint with respect to the identifying relationship.
- The partial key of the Copy entity is Copy Number; for each owner entity Book, the Copy Number uniquely identifies the copy for each book.

1.1.4. Refining ER designing

Refining is the process of analyzing of the entity relationship design of a database.

1.1.5. ERD naming, conventions, and design issues

Specify structural constraints on relationships

- Replaces cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints

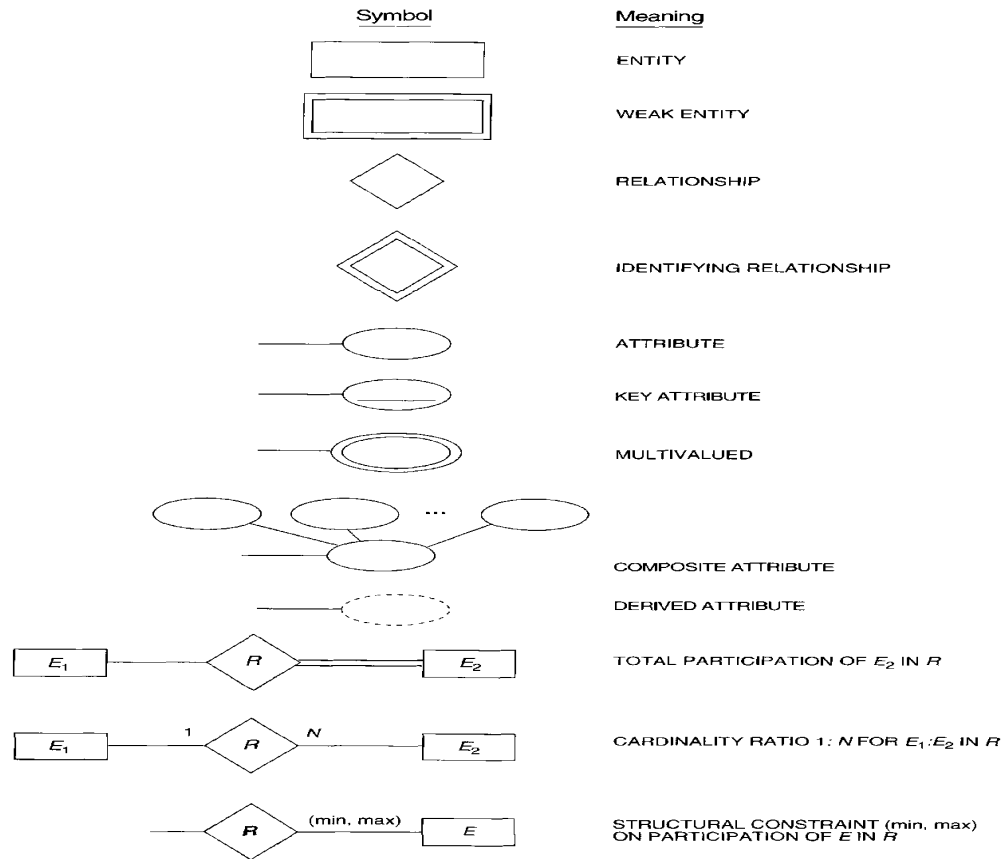
Entity naming convention

- Entity names should be a noun, singular (written in upper case).
- Where necessary, underscores should join the words.
- Where possible, avoid using the organization’s name as part of the table name.
- For physical implementation, Entity Names can have a maximum of 44 characters.
- Names for the entities must be meaningful and must not conflict with other entity names.
- Where abbreviations are used, the full words must be obvious.
- Each entity name should be unique in the database

Attribute Naming convention

- Attributes’ name initial character should be upper case followed by lower case.
- Where necessary, underscores should join the words.

- No two attributes of an entity type having the same attribute name, but Attributes of *different entities* can have the same name.



1.1.6. Notation for UML class diagrams

What is UML?

The UML (Unified Modeling Language) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the object of a software system. It captures decisions and understanding about systems that must be constructed.

The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components."

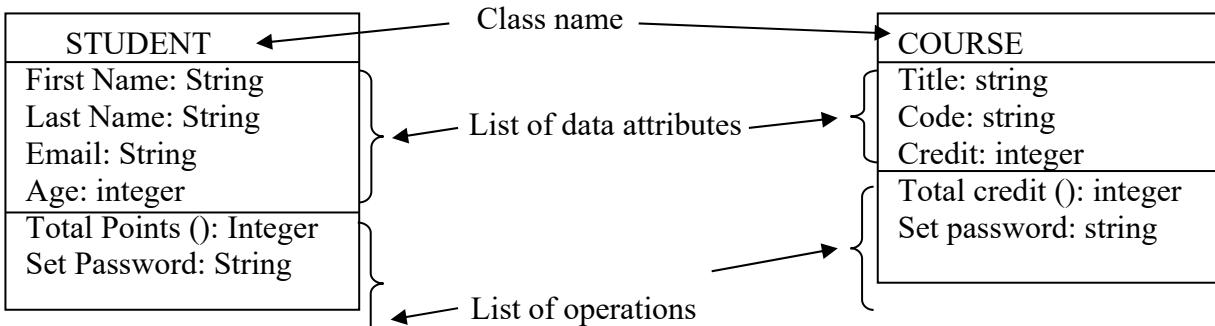
What is class?

- A class is a set of objects that share the same attributes, operations, and relationships. So, a class is similar to an entity type, only operations are added.
- It is the most important building block of any object-oriented system.

A class is symbolized by a rectangle with normally three "compartments" (sections) that contain:

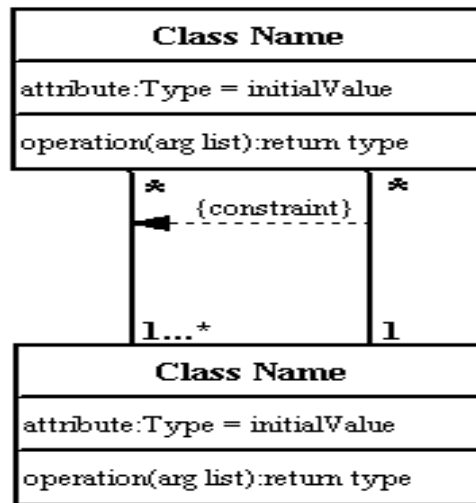
- Top section (class name).
- Middle section (attributes).
- Bottom section (operations that can be applied to individual objects).

For example, see the figure below:



- Relationships in UML are called “**associations**”.
- Cardinalities in UML are called multiplicities.

Place multiplicity notations (association names) above, on, or below the association line near the ends of an association in the form of (min, max). These symbols indicate the number of instances of one class linked to one instance of the other class.



Operation (interface):

An operation is a specification of a transformation or query that an object may be called to execute. It has a name and a list of parameters.

1.2. Enhanced ER diagram and UML modeling

Enhanced ER diagram includes all modeling concepts of the ER model.

In addition, EER diagram includes:

- Subclasses and superclasses
- Specialization and generalization
- Category or union type
- Attribute, relationship, and inheritance

EER diagram is used to model concepts more accurately than the ER diagram

➤ Subclasses, superclasses and inheritance

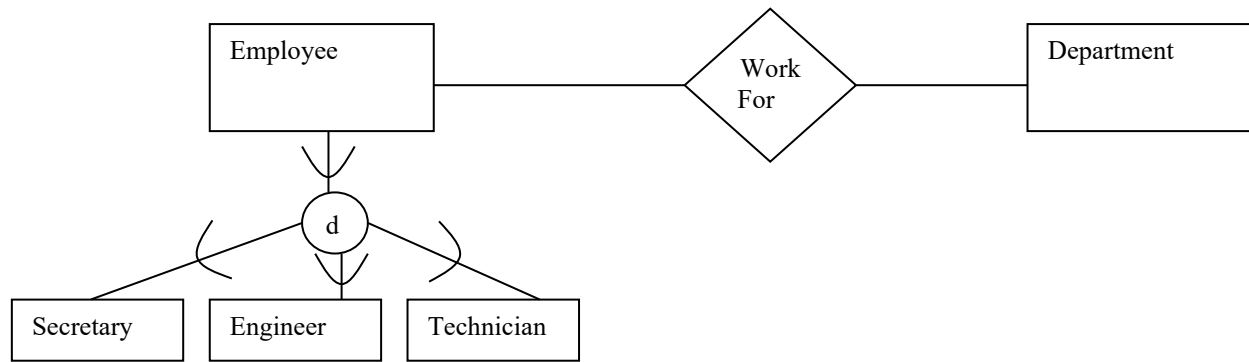
Entity type Y is a subtype (subclass) of an entity type X if and only if every Y is necessarily in X.

An Entity type X is a supertype (superclass) of an entity type Y if and only if an entity Y is member of an entity X.

Therefore, Entity type Y is a subclass of entity type X and entity type X is a superclass of entity type Y.

Example:

EMPLOYEE may be further grouped into {SECRETARY, ENGINEER, MANAGER, and TECHNICIAN}.



{**SECRETARY, ENGINEER, TECHNICIAN**} is a subset (subclass) of entity type **EMPLOYEE** and **EMPLOYEE** is the superclass for each of those subclasses.

These are called superclass/subclass relationships.

Example: **EMPLOYEE/SECRETARY**, **EMPLOYEE/TECHNICIAN**, and **EMPLOYEE/ENGINEER**.

The SubClass is also called the “child class” of the BaseClass, and the BaseClass called “parent class”, or SuperClass for the SubClass.

Inheritance is the process that allows a class to acquire the properties of another class. The class that inherits the properties is called the *subclass* or *subtype*.

➤ Specialization and generalization

Specialization is the process of defining a set of subclasses from a class.

- It is process of creating new subclasses from an existing based upon some distinguishing characteristics of the entities in the superclass.

Example: {**SECRETARY, ENGINEER, TECHNICIAN**} is a specialization of **EMPLOYEE** based upon job type. The Same superclass may have several specializations, for example, {**SALARIED_EMPLOYEE**, and **HOURLY_EMPLOYEE**} is another specialization of **EMPLOYEE** based upon method of pay.

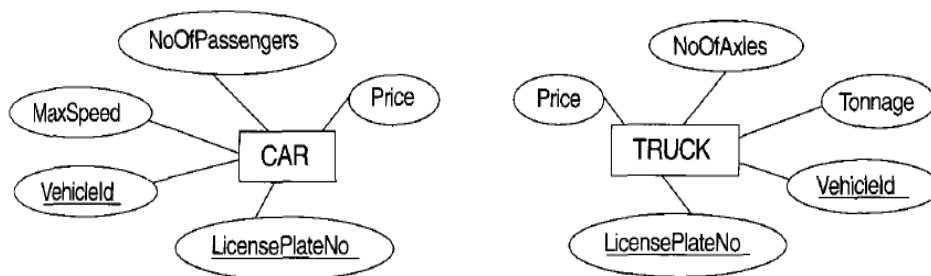
- Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams.
- Attributes of a subclass are called specific attributes.

Example: Typing Speed of **SECRETARY**

Generalization is the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass.

- It is the reverse of specialization process.
- Several classes with common features are generalized into a super class.
- Shared characteristics can be attributes, associations, or methods.

Example: the entity types Car and Truck share common attributes License_PlateNo, VehicleID and Price. Therefore, they can be generalized into the super class Vehicle with the common attributes.



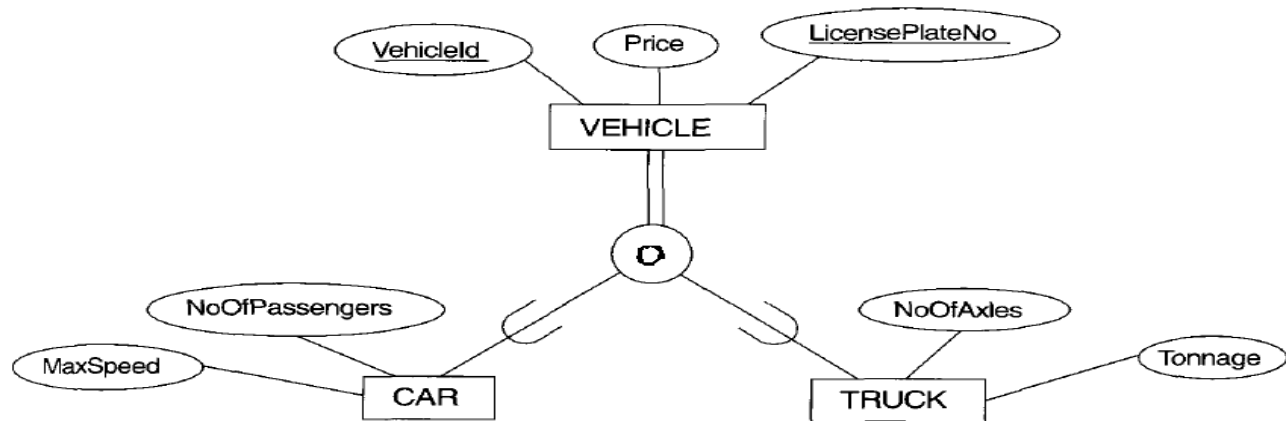


FIGURE 4.3 Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.]

➤ Constraints and characteristics of specialization and generalization

• Constraints on specialization and generalization

Two conditions can be applied to a specialization/generalization:

- **Disjointness** Constraint – This specifies that the subclasses of the specialization must be disjointed.
 - disjoint can be specified by **d** in EER diagram.
 - Disjoint constraint can be total or partial.
- **Overlap** constraint – this specifies that that is the same entity may be a member of more than one subclass of the specialization.
 - Overlapping constraint can be specified by **O** in EER diagram.
 - Overlapping constraint can be total or partial.
- **Characteristics of specialization**
 - . A subclass entity inherits all attributes and relationships of its superclass entity.
 - . A subclass entity may have its own specific attributes and relationships (together with all the attributes and relationships it inherits from the superclass).

Example: {SECRETARY, ENGINEER, TECHNICIAN}, inherits all attributes and relationship of an entity type EMPLOYEE.

• Characteristics of generalization

- . A superclass entity can take shared characteristics (Attributes, Association or methods) of its subclass entity.
- . A superclass entity may have its own specific attributes and relationships (together with all common attributes and relationships it take from the subclass).

➤ Modeling of union types using categories

Superclasses represent different entity types (subclasses). Such a subclass is called a category or UNION TYPE

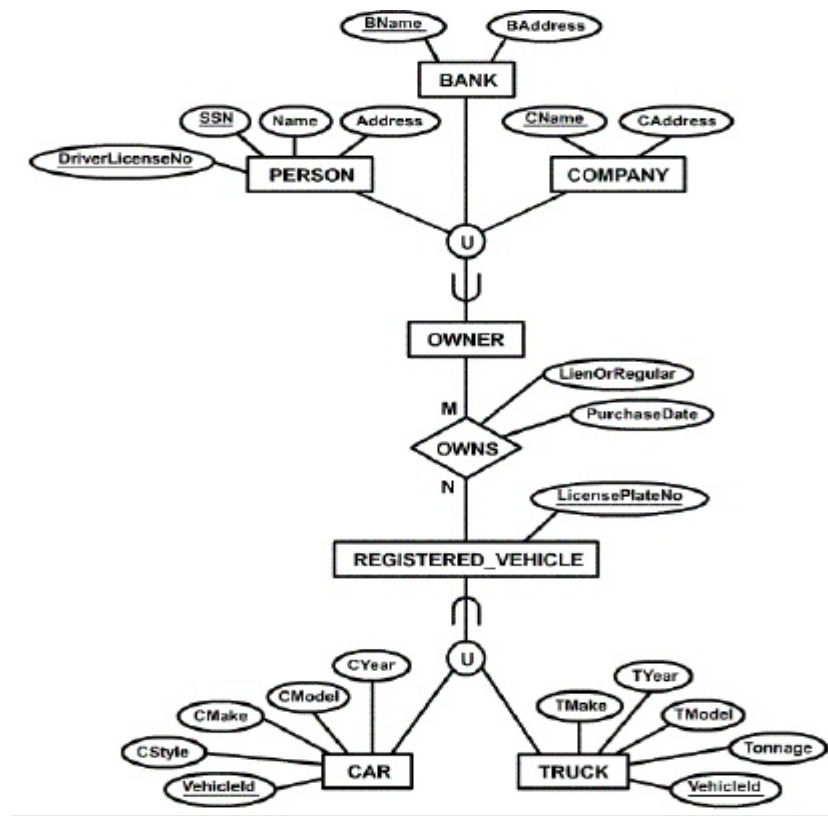
Example:

Database for vehicle registration, vehicle owner can be a person, a bank (holding a lien on a vehicle) or a company.

- Category (subclass) OWNER is a subset of the union of the three superclasses COMPANY, BANK, and PERSON
- A category member must exist in at least one of its superclasses

Union type or a category

- Represents a single superclass/subclass relationship with more than one superclass
- Subclass represents a collection of objects that is a subset of the UNION of distinct entity types
- Attribute inheritance works more selectively
- Category can be **total** or **partial**
- Some modeling methodologies do not have union types

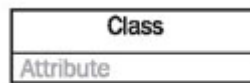


➤ **Represent specialization/Generalization and inheritance in UML class diagrams**

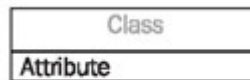
Specialization, generalization and inheritance in UML class diagrams include Basic notation for **superclasses, Subclasses, association and multiplicity**.

An UML class diagram includes the following elements:

Class - A class represents a relevant concept from the domain, a set of persons, objects, or ideas.



Attribute- An attribute of a class represents a characteristic of a class .



Generalization - *Generalization* is the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass.



Association - An association represents a relationship between two classes:

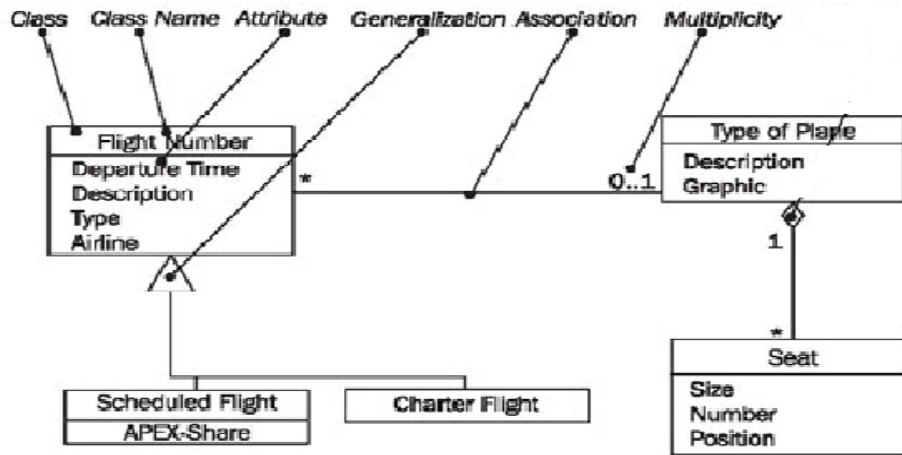
Works_on ►

An association indicates that objects of one class have a relationship with objects of another class, in which this connection has a specifically defined meaning.

Multiplicity - A multiplicity allows for statements about the number of objects that are involved in an association:

* ————— 0..1

Aggregation - An aggregation is a special case of an association (see above) meaning “consists of”:



➤ **Relationship types of degree higher than two**

The entities that are participating in a given relationship are known as participants. The number of participants in a given relationship is called **degree of relationship**.

In E-R model, the most common relationship types (with respect to degree of relationships) are:

- (1) Binary relationship
- (2) Unary relationship
- (3) Ternary (or higher degree) relationship

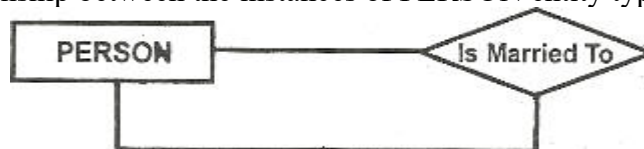
• **Unary (degree one) relationship**

The relationship between the instances of only one entity type (or participant) is known as **unary relationship**. It is also referred to as recursive relationship. This relationship has degree one.

a) **One-to-One Unary Relationship**

The **one-to-one unary relationship** is represented in the similar way as one-to-one binary relationship. Some examples are given below to explain the concept of one-to-one unary relationship.

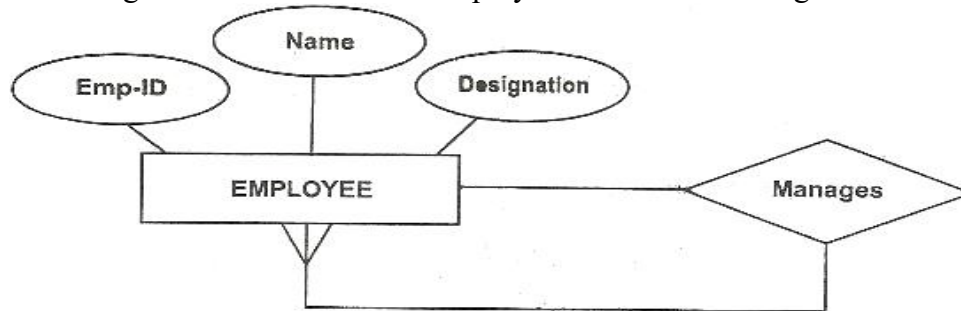
Examples: The recursive relationship between the instances of PERSON entity type is:



The above relationship shows that a person is married to a person. On the other hand, a person is married by another person.

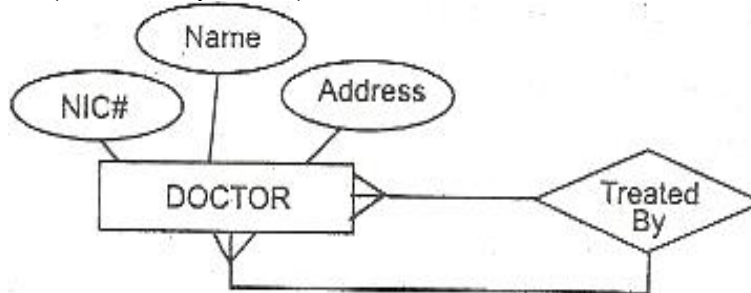
b) One-to-Many Unary Relationship

The *one-to-many unary relationship* is shown in figure below. This relationship is named as 'Manages' that associates employees of an organization with another employee who is their manager.



c) Many-to-Many Unary Relationship

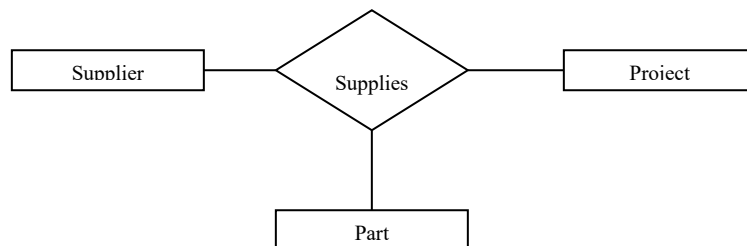
The *many-to-many unary relationship* "Treated-By" is given below. It represents the situation in which doctors give treatments to other doctors (doctors as patients).



The above relationship shows that a doctor is treated by many other doctors. On the other hand, a doctor treats many other doctors.

- Ternary (degree three) relationship

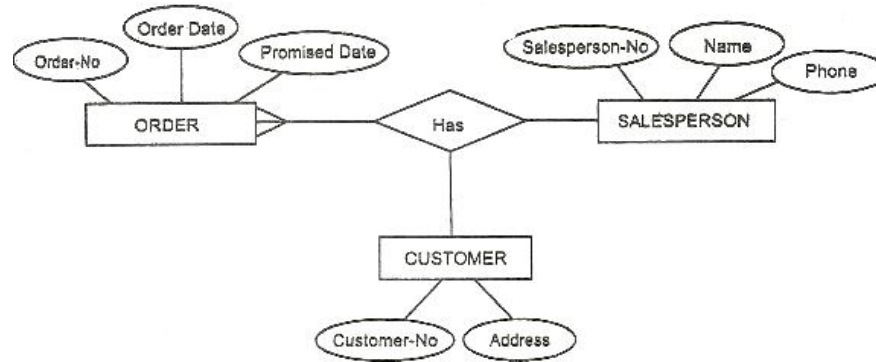
A *ternary relationship* is a simultaneous relationship among instances of three entity types. This type of relationship has degree three. Example:



The name of relationship is "Supplies". The Supplier supplies Parts to Project.

The above ternary relationship "Supplies" can be read as "A Supplier can supply many Parts to a particular Project".

Another example of ternary relationship "Has" is given below, in which three entities ORDER, CUSTOMER and SALESPERSON are associated with each other.



The above ternary relationship "Has" can be read as "A salesperson has many orders from many customers"

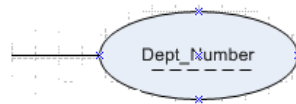
LO2. Test Database Performance

➤ Naming standard of Keys

In E-R Diagram, any candidate key attributes of an entity type can be represented by ellipse shape with underline the name.



In E-R Diagram, any foreign key attributes of an entity type can be represented by ellipse shape with underline with a dot line for the name.



➤ Audit Trails

An **audit trail** (or **audit log**) is a security-relevant chronological (in order) record, set of records, or destination and source of records that provide documentary evidence of the sequence of activities that have affected at any time a specific operation, procedure, or event.

Audit records typically result from activities such as financial transactions, scientific research and health care data transactions, or communications by individual people, systems, accounts, or other entities.

The process that creates an audit trail is typically required to always run in a privileged mode, so it can access and supervise all actions from all users; a normal user should not be allowed to stop/change it. Furthermore, for the same reason, trail file or database table with a trail should not be accessible to normal users.

Database auditing involves observing a database so as to be aware of the actions of database users. Database administrators and consultants often set up auditing for security purposes, for example, to ensure that those without the permission to access information do not access it

"Audit trails help promote data integrity by enabling the detection of *security* breaches.

One of the most significant aspects of database security involves setting up auditing to record user activities.

➤ Maintaining equipment inventory

Having an accurate equipment inventory is an important part of the maintenance process. If users reporting problems can select from a complete and accurate list of equipment items, they will more quickly and accurately report their problems, and the system can more effectively route the problems to the appropriate approval and execution processes.

An accurate equipment inventory optimizes the maintenance process and thereby minimizes the direct and indirect costs associated maintaining equipment.

Equipment Standards

An important step of maintaining an equipment inventory is to establish the types of equipment at your company -- the equipment standards

Setting up Equipment Standards

Select the View and Edit Equipment Standards task. A list of all defined equipment standard will show. Select an equipment standard, and the system displays detailed information on the selected equipment standard in the right frame.

You can make changes to the information if needed and click on the Save button to submit the changes.

To add a new equipment standard, click on the Add New button and enter your details.

View and Edit Inventory

Select the View and Edit Equipment Inventory task. The system presents a three-panel view.

- In the top panel, you can restrict the equipment listing.
- In the left panel, the restricted equipment is listed. Click the Show button in the panel header of the top panel to populate the list.
- Select an equipment item in the list, and its details display in the right panel.

You can make these edits:

- To update equipment details, make changes in the details panel and click the Save button.
- To add a new equipment to the inventory, click the Add New button in the inventory listing panel, enter data in the details panel, and save your changes.

➤ Client training and satisfaction reports

Client training increase productivity or enhance current skill, the training sessions provide a valuable opportunity to turn client risk management knowledge into a competitive advantage.

Client/Customer satisfaction report is a measure of how products and services supplied by a company meet or surpass customer expectation. Customer satisfaction is defined as "the number of customers, or percentage of total customers, whose reported experience with a firm, its products, or its services (ratings) exceeds specified satisfaction goals

LO3. Seek client feedback

➤ Implementing feedback mechanisms in line with organization policies

Client feedback can be implementing in line with the organization policies through:

- | | |
|---------------------|--------------------|
| - Prototyping | - Formal feedback |
| - Verbal feedback | - Questionnaire |
| - Informal feedback | - Survey |
| | - Group discussion |

• Prototyping

A **prototype** is an early sample or model built to test a concept or process or to act as a thing to be replicated or learned from. A prototype is designed to test and trial a new design to enhance precision by system analysts and users. Prototyping serves to provide specifications for a real, working system rather than a theoretical one.

Some Advantages of Prototyping:

- Reduces development time.
- Reduces development costs.
- Requires user involvement.
- Developers receive quantifiable user feedback.
- Facilitates system implementation since users know what to expect.
- Results in higher user satisfaction.
- Exposes developers to potential future system enhancements.

- **Verbal feedback**

Verbal feedback should be used to supplement, support or as part of other evaluation activity. It can be formal or informal.

Informal verbal feedback is most often used to evaluate just after a training event. Sometimes it may be in the form of a throw away comment or something more specific.

Informal verbal feedback is good for:

- Evaluating at validation, learning and meeting of objectives levels.
- Carrying out formative assessment of training programmes.

Formal verbal feedback can assist you to evaluate for validation, whether objectives met or for learning.

Formal verbal feedback is good for:

- Informing the tutor of any immediate reactions to the training.
- Allowing the facilitator to focus subsequent evaluations around specific issues.
-

- **Informal feedback**

Informal feedback is a type of feedback which offer daily encouragement to team members and discuss comments from customers. It can often give the employee a sense of job performance and can give motivation.

- **Formal feedback**

Formal Feedback is documented feedback. In some cases this may be a form of corrective counselling intended to make an employee aware of their performance or lack of performance.

- **Questionnaire**

Questionnaires are effective mechanisms for efficient collection of certain kinds of information.

Questionnaires may have only two options (yes/no) or multiple options or rank scaling, etc.

All the readers need to do is tick the most appropriate answer according to them.

Example: How satisfied were you with the training?

1. Overall quality of the training

Excellent	Good	Satisfactory	Poor
-----------	------	--------------	------

1. Attitude of the trainer

Excellent	Good	Satisfactory	Poor
-----------	------	--------------	------

2. Trainer's knowledge of the topics

Excellent	Good	Satisfactory	Poor
-----------	------	--------------	------

3. Handouts and training aids

Excellent	Good	Satisfactory	Poor
-----------	------	--------------	------

4. What would you recommend to the trainer to be done differently? (Please use the back of the paper if you need more space.)

- **Survey**

A **survey** is a data collection tool that used to gather information about individuals.

A **feedback survey** is designed to collect feedback from clients about their interactions with a business.

- It provides space for customers to voice their compliments, complaints, and suggestions.
- Feedback surveys help businesses improve the quality of their information, products, and services.

- **Group discussion**

After group discussion, the group provides feedback to the trainer or facilitator.

It is rated on various parameters such as attitude, confidence, communication, interpersonal skills, and flow of thoughts.