

INTRODUCTION

- DBMS- Collection of interrelated data & a set of programs to access those data.
- Goal of DMBS- To provide a way to store and retrieve information conveniently and efficiently.
- Designed to manage large bodies of information.
- Involves defining structures for the storage of information & providing mechanisms for the manipulation of information
- To ensure safety of the stored information incase of system crash or unauthorized access.

- **Database System Applications**

- **Finance (Banking)**
- **Transport (Airlines/Trains/Roadways)**
- **Educational Institutes/Universities**
- **Healthcare**
- **Credit Card Transactions**
- **Telecommunications**
- **Manufacturing**
- **Sales**
- **Human Resources**
- **Entertainment**

DBMS vs File System

- Disadvantages of File System:
 - Data Redundancy & inconsistency
 - Difficulty in accessing data
 - Data isolation
 - Integrity problems
 - Atomicity problems
 - Concurrent-access anomalies
 - Security problems

View of Data

- A major purpose of a DBMS is to provide users with an abstract view of data.
- Data Abstraction:
 - Developers hides the complexity from users through several levels of abstraction.
 - **Physical Level:** describes how the data are actually stored
 - **Logical level:** describes what data are stored in the database & what relationship exists among those data. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

➤ **View level:** The highest level of abstraction describes only part of the entire database.

➤ Example: shows the relationship among the three levels of abstraction.

in a Pascal like language we may declare a record as follows:

```
type customer = record  
    customer-id: string;  
    customer-name: string;  
    customer-street: string;  
    customer-city: string;  
end;
```

This code defines a record type called *customer* with four fields

Each field has a name & type associated with it.

A banking enterprise may have several such record types including,

account, with fields *account-number* & *balance*

employee, with fields *employee-name* & *salary*

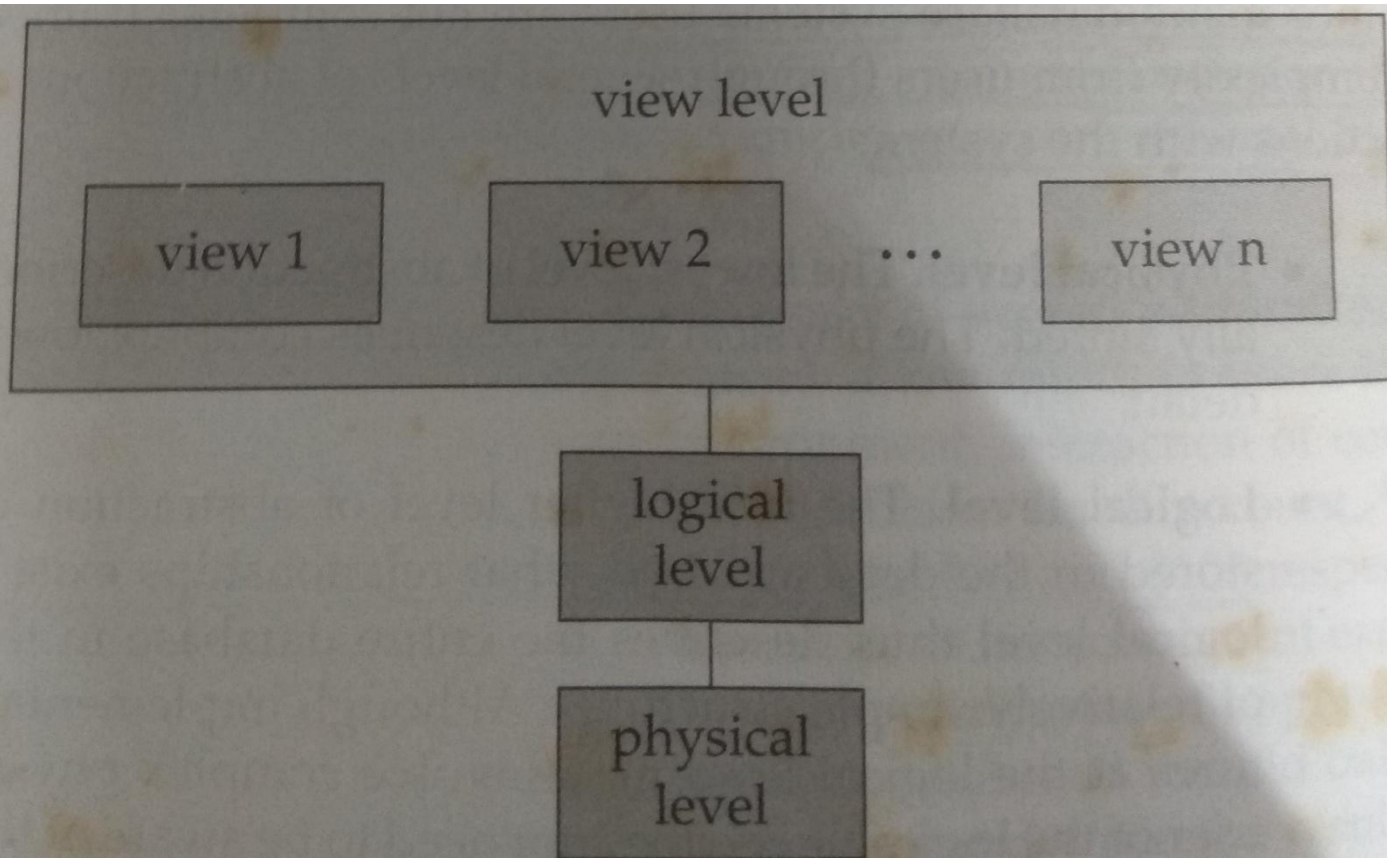


Figure 1.1 The three levels of data abstraction.

- At the physical level, a customer, account or employee record can be described as a block of consecutive storage locations (e.g. words or bytes)
- The language compiler hides this level of detail from programmer.
- At the logical level, each such record is described by a type definition, as shown in previous code segment.
- Programmers using the programming language work at this level of abstraction.
- Finally, at the view level Computer users see a set of application programs that hides the details of the data types.
- At view level several views of the database are defined. Also provide a security mechanisms to prevent unauthorized access.

Instances and Schemas

- Database changes over time as information is inserted and deleted.
- The collection of information stored in the database at particular moment is called an **instance** of the database.
- The overall design of the database is called as the database **schema**.
- In analogy to a computer program- a database schema corresponds to the variable declarations . Each variable has a value at a given instant.
- The value of the variables in a program at a point in time corresponds to an *instance* of the database *schema*.

- Database systems have several schemas partitioned according to the levels of abstraction.
- The physical schema describes the database design at the physical level.
- The logical schema describes the database design at the logical level.
- A database may also have several schemas at view level, that describes different views of the database.
- Programmers construct applications using logical schemas.
- The physical schema is hidden beneath the logical schema and can be changed easily without affecting the application program.

Data Models

- Underlying the structure of a database is the data model: a collection of conceptual tools for describing the data, data relationships, data semantics and consistency constraints.
- Two data models-
 - The entity-relationship model
 - Relational model

The Entity-Relationship Model

- The E-R model is based on a perception of a real world that consists of a collection of basic objects, called *entities*, and of *relationships* among these objects.
- An entity is a 'thing' or 'object' in the real world that is distinguishable from other objects.
- E.g. a person, a bank account, a car etc.
- Entities are described by a set of attributes.
- E.g. a bank account may have attributes such as account-number & balance.
- Customer-name, customer-address, customer-email are the attributes of the Customer entity.

- A relationship is an association among several entities.
- E.g. depositor relationship associates customer with account.
- The set of all entities & relationships of the same type are termed as an entity set and relationship set, respectively.
- The structure of database can be represented graphically by an E-R diagram.
- Components of the E-R diagram:
 - **Rectangles** – represents entity sets
 - **Ellipse**- represents attributes
 - **Lines**- link attributes to entity set and entity sets to relationships.

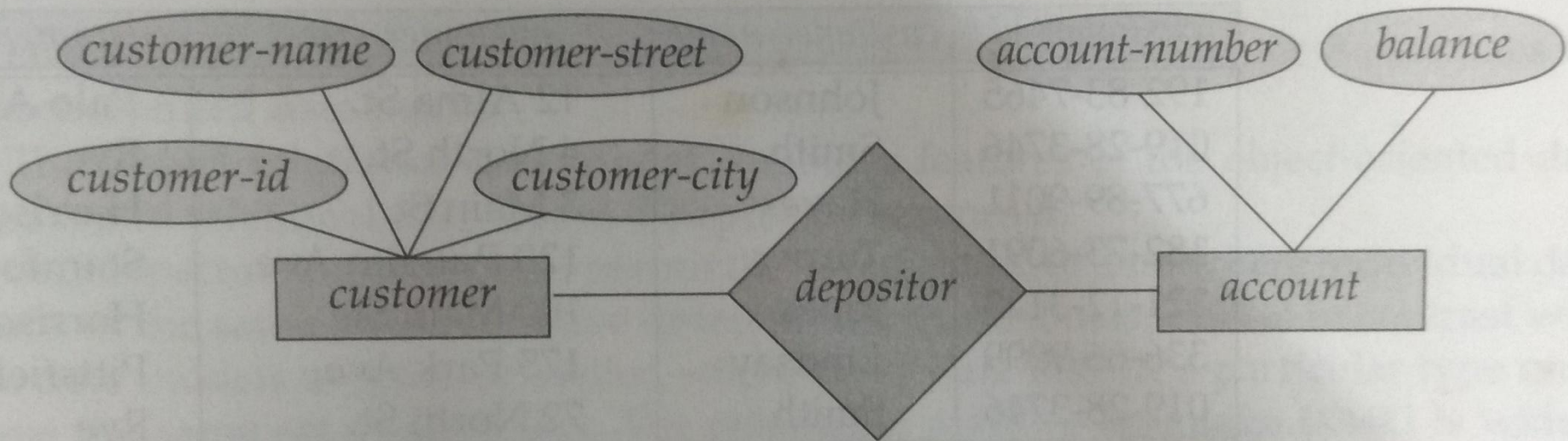


Figure 1.2 A sample E-R diagram.

The Relational Model

- It uses a collection of tables to represent both data and the relationships among the data.
- Each table has multiple columns with unique name.
- The relational model is an example of a record based model.
- It is possible to create schemas in the relational model that have problems such as unnecessary duplicated information.
- See figure:

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

<i>account-number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

<i>customer-id</i>	<i>account-number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

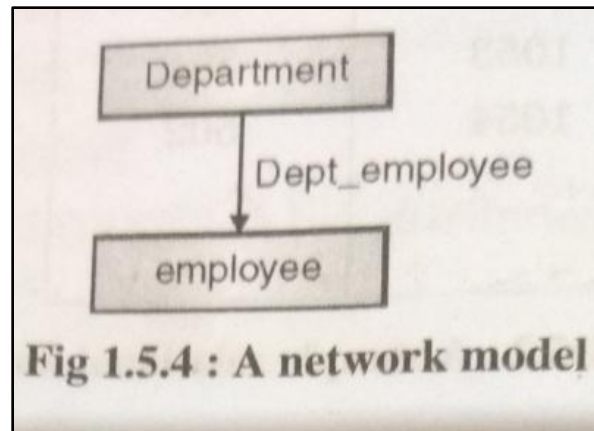
(c) The *depositor* table

Figure 1.3 A sample relational database.

Other Data Models

- Network Model

- It uses two different data structures:
- 1. A record type is used to represent an entity set. Attributes are represented by data items.
- 2. A set type is used to represent a direct relationship between two record types. It specifies 1:M relationship between owner record type and the member record type.
- Sample record type and set type are shown in below fig:



- The arrow direction is from the owner record type to member record type.
- Figure shows two record types:
 - 1) Department
 - 2) Employee

And the set type Dept_employee

- A one-to-many relationship is shown by a set type arrow that starts from the owner field in the owner record type. The arrow points to the member field within the member record type.

- Object Relational Data Model:
 - It combines the features of the object oriented data model and relational data model.
 - Existing relational database management system can be extended to handle objects:
 - To provide support for complex objects
 - To provide user extensibility for data types, operators and access methods
 - To provide active database facilities and inference support

The main advantage of ORDM is reuse and sharing. Standard functions can be implemented centrally, rather than coding it in each application.

Database Languages

- DBMS provides necessary interfaces for users.
- Users can interact with DBMS using database language.
- Two facilities are provided for interaction with DBMS:
 - Data Definition Language (DDL)
 - Data Manipulation Language (DML)
- A database system provides a **data definition language** to specify the database *schema* and **data manipulation language** to express database *queries and updates*.
- These are not two separate languages.
- They simply form parts of a single database language, such as widely used **SQL** language.

Data-Definition Language

- We specify the database schema by a set of definitions expressed by a special language called **Data-Definition Language (DDL)**.
- E.g.

create table account

(account-number **char**(10),

Balance **integer**)

- Execution of the above DDL statement creates account table .
- In addition it updates a special set of tables called the *data dictionary or data directory*.
- A data dictionary contains metadata – that is data about data.
- The schema of a table is an example of metadata.
- The database system consults the data dictionary before reading or modifying actual data.

Data-Manipulation Language

- **Used to:**

- The retrieval of information stored in the database.
- The insertion of new information to the database.
- The deletion of information from the database.
- The modification of information stored in to the database.

- **There are basically two types:**

- Procedural DMLs: require a user to specify what data are needed and how to get those data.
- Declarative DMLs (Non-Procedural): require a user to specify what data are needed without specifying how to get those data.
- The DML component of SQL is non-procedural.

select customer.customer-name

from customer

where customer-id=192-83-7465

select account.balance

from depositor,account

where depositor.customer-id=192-83-7465 **and**

depositor.account-number=account.account-number

Components of DBMS

- The major components of a DBMS are
 - 1. **DML pre-compiler**
 - 2. **DDL compiler**
 - 3. **File manager**
 - 4. **Database manager**
 - 5. **Query processor**

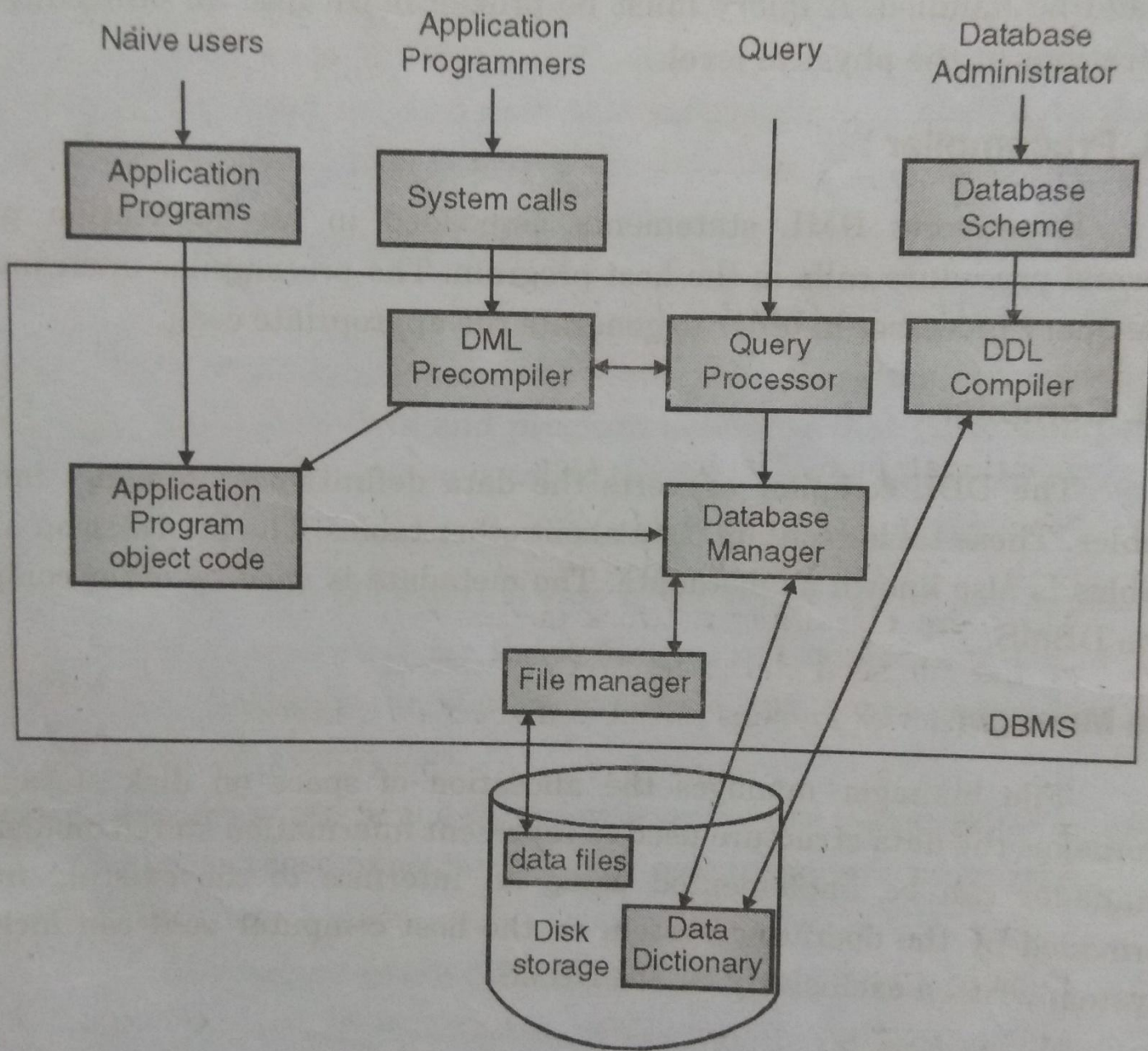


Fig 1.6.1 : DBMS architecture

- A DBMS is a complex software and has to provide several services.
- Divided into several components each assigned a specific task.
- A DBMS has to interact with user queries and physical files containing data records.
- The functional components can be broadly classified in to:
 - *1. Storage Manager*
 - *2. Query Processor*

- A DBMS requires large amount of storage space.
- Data is moved between disk storage and main memory.
- Disk is relatively a slow service
- DBMS has to structure the data so as to minimize the movement of data.
- The query processor facilitates access to data.
- Quick processing of database updates and retrieval of information is important.
- Embedded queries should be handled.
- A query must be broken down into an efficient sequence of operations at the physical level.

1. DML Pre-compiler:

- It converts DML statements embedded in an application program to normal procedure calls in the host program.
- The pre-compiler must interact with the query processor in order to generate the appropriate code.

2. DDL Compiler:

- Converts the data definition statements into a set of tables.
- These tables contain data about other tables (metadata).

3. File Manager:

- Manages the allocation of space on disk storage.
- Also manages the data structure used to represent the information stored on disk.

4. Database Manager:

- It is program module which provides the interface between the low level data stored in the database and the application programs and queries submitted to the system.

Important responsibilities of database manager:

- Interaction with file manager
- Integrity enforcement
- Security enforcement
- Backup and recovery
- Concurrency control

5. Query Processor:

- Users formulates queries in DML.
- Query processor interprets user's query and converts it into series of operations in a form capable of being sent to the data manager for execution.

❑ **Database Administrator (DBA)**

The person having control of data and programs accessing that data in a DBMS is called as DBA.

❑ **Role and Responsibilities of DBA:**

- 1) Administers the three levels of the database.
- 2) Specifies the external view of the various users and applications.
- 3) Responsible for the definition and implementation of the internal level including the storage structure and access methods to be used for optimum performance.
- 4) Ensures integrity of the database.

- 4) Protects database from unauthorized access. DBA creates and stores profile of each user describing permission levels.
- 5) Define procedures to recover from failure.
- 6) Schema and physical organization modification.

Data Modeling

- Data Modeling is an abstraction which allows conceptualization of the associations between various entities and their attributes.
- E-R model is very useful in modeling an application database before implementing it using any commercially available DBMS.
- E-R model is generalization of hierarchical and network models.
- Used basically in design of logical database.

➤ E-R models uses three features to describe database:

- 1) Entity Sets
- 2) Relationship Sets
- 3) Attributes

❖ **Entity Sets:**

- ✓ An entity is a distinguishable object in an application.
- ✓ It is characterized by a set of attributes.
- ✓ It may be physical object as book or student
- ✓ It may be abstract as bank account, accident, registration etc.
- ✓ An entity set is a set of entities of same type.
 - E.g. set of all students in an institute, set of all employees in a office etc.

❖ Attributes:

- ✓ They are descriptive properties of each member of an entity set.
- ✓ Each entity has a value for each of its attributes.
- ✓ E.g. Attributes of student entity are:
 - Name, Roll-no, DoB, City etc. & Values for the attributes are: (Mohan, MC22F14F001, 12/03/2000, Aurangabad)

❑ Simple or Composite Attributes:

- ✓ A simple attribute can not be further subdivided into smaller components.
- ✓ A composite attribute can further be subdivided into smaller components.

Some composite attributes are shown in Fig. 2.1.2.

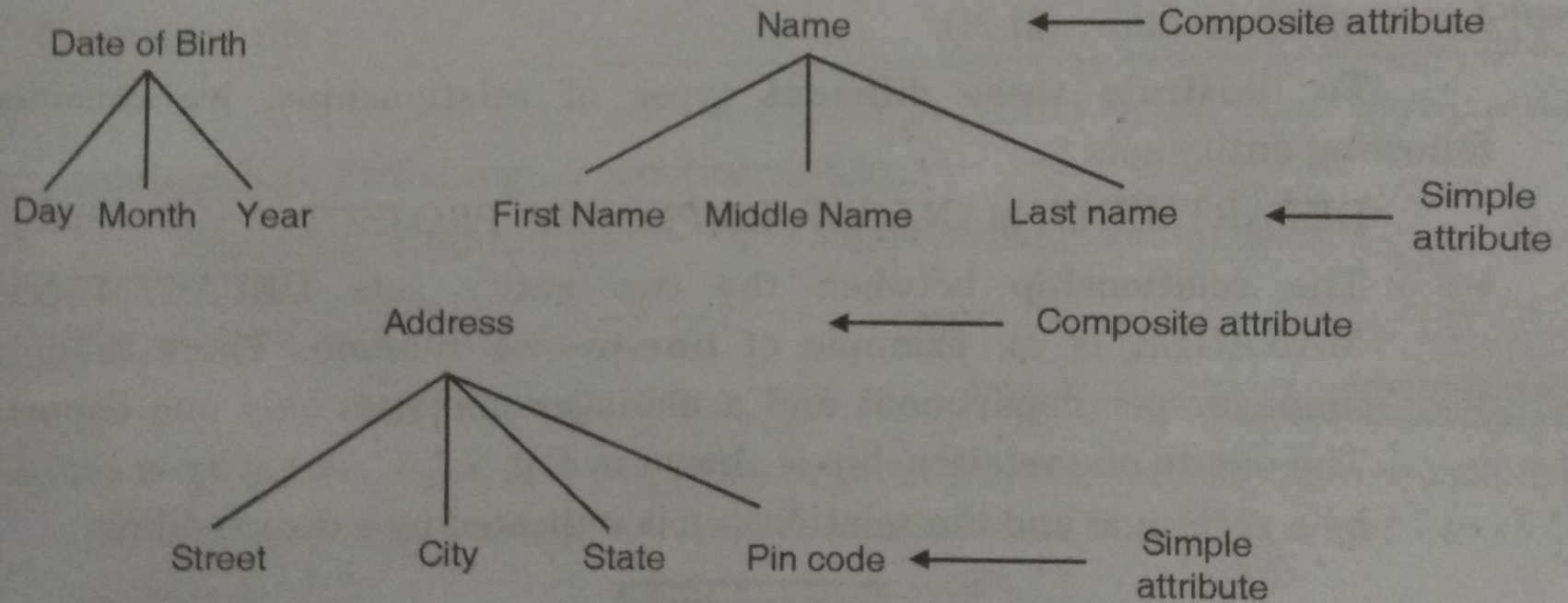


Fig. 2.1.2 : Some composite attributes

❑ **Derived Attributes:**

- ✓ The value of such an attribute is derivable from the value of a related attribute or a set of attributes.
- ✓ E.G. value of the attribute 'age' can be derived from 'DoB'.
- ✓ The value of the derived attribute is not stored but it can be computed as and when required.

❖ Relationship Sets:

- ✓ It is an association among several entities.
- ✓ A relationship can be used to model:

1. Interaction among entities

2. Constraint on the multiplicity of the association

A relation between two entity sets can be of the following types:

- 1) One-to-One (1-1)
- 2) One-to-Many (1-M)
- 3) Many-to-One (M-1)
- 4) Many-to-Many (M-M)

Consider Following Entity Sets:

DEPARTMENT, MANAGER, EMPLOYEE, PROJECT

- 1) DEPARTMENT and MANAGER is an example of **one-to-one** relationship.
- 1) DEPARTMENT to EMPLOYEE is an example of **one-to-many**
- 2) Relationship. The reverse relationship would be **many-to-one**.
- 3) EMPLOYEE to PROJECT is an example of **many-to-many**.

❖ Constraints

- ✓ Two constraints may be placed on entity types that participate in a relationship.

1. Mapping Cardinalities

2. Participation constraints.

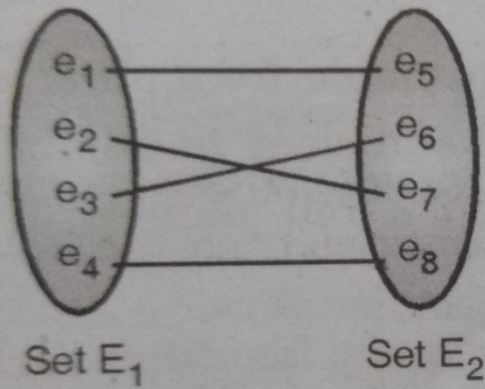
❖ Mapping Cardinalities:

- ✓ Number of entities participating in a relationship from another

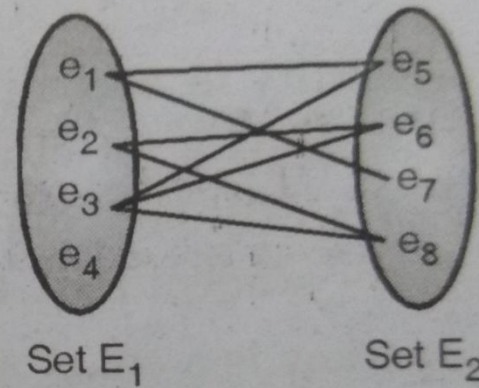
entity set.

- ✓ A binary relationship between two entity sets E1 and E2 must be one of the following:

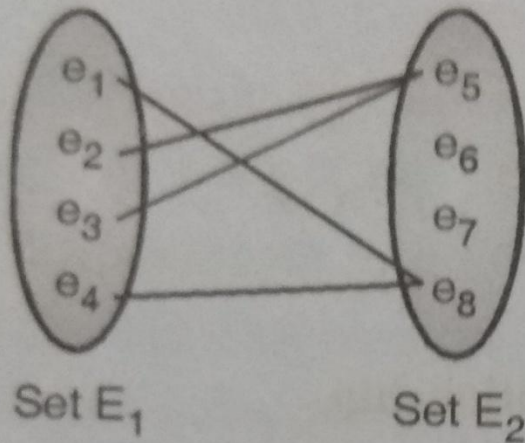
Different cases of mapping cardinalities are shown in Fig. 2.2.1 (a to



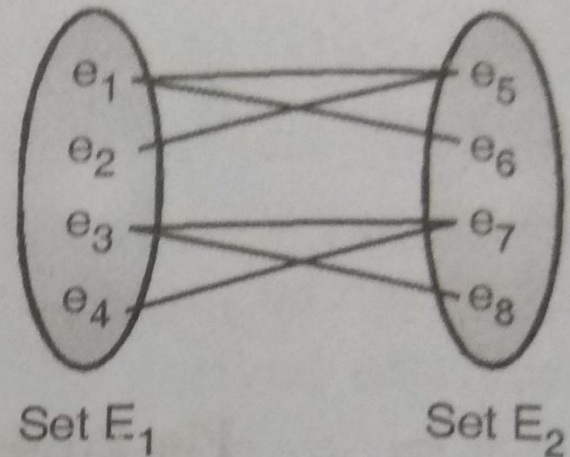
(a) one-to-one mapping



(b) one-to-many mapping



(c) many-to-one mapping

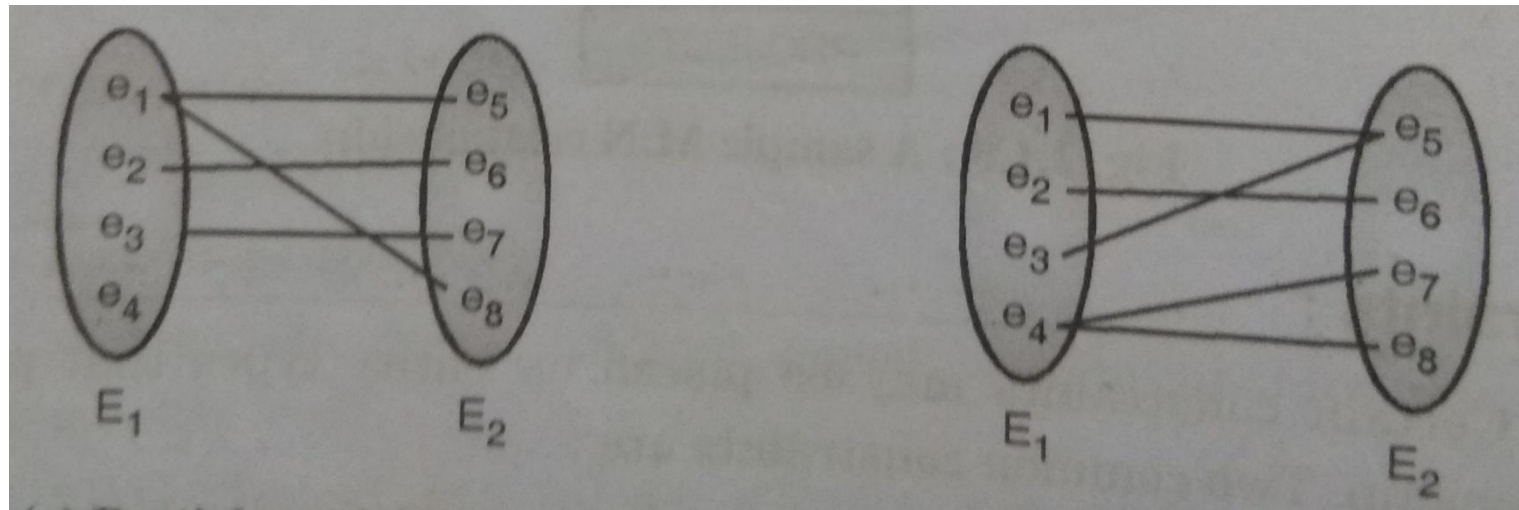


(d) many-to-many mapping

Fig. 2.2.1 : Different cases of mapping cardinalities

❖ Participation Constraints:

- ✓ It represents whether all or only some entities in an entity set participate in relationship.
- ✓ The participation of an entity set E is said to be total if every entity in set E participates in a relationship, otherwise it is said to be partial.



Partial Participation

Total Participation

❖ Keys

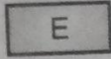
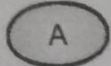

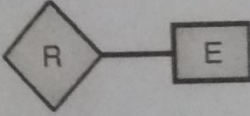
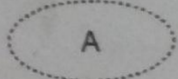
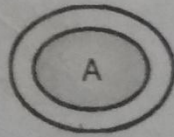
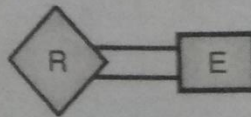
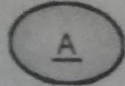
- ✓ A key is a set of attributes that is sufficient to uniquely identify an entity in an entity set.
- ✓ E.g. Student roll_no.
- ✓ A **super key** is a set of one or more attributes that uniquely identifies each entity of an entity set.
- ✓ A **key** is the minimal set of attributes that uniquely identifies each entity of an entity set.
- ✓ An entity set can have several keys called **candidate keys**.
- ✓ A **primary key** is a key chosen for database as a primary way of recognizing an entity in an entity set.

- ✓ For example, the customer-id attribute of the entity set customer is sufficient to distinguish one customer entity from another. Thus, customer-id is a superkey.
- ✓ Similarly, the combination of customer-name and customer-id is a superkey for the entity set customer.
- ✓ The customer-name attribute of customer is not a superkey, because several people might have the same name.
- ✓ If K is a superkey, then so is any superset of K . We are often interested in superkeys for which no proper subset is a superkey.
- ✓ Such minimal superkeys are called candidate keys.
- ✓ It is possible that several distinct sets of attributes could serve as a candidate key.

- ✓ Suppose that a combination of customer-name and customer-street is sufficient to distinguish among members of the customer entity set. Then, both (customer-id) and (customer-name, customer-street) are candidate keys.
- ✓ Although the attributes customer id and customer-name together can distinguish customer entities, their combination does not form a candidate key, since the attribute customer-id alone is a candidate key.
- ✓ We shall use the term primary key to denote a candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.
- ✓ A key (primary, candidate, and super) is a property of the entity set, rather than of the individual entities.

- ✓ Candidate keys must be chosen with care.
- ✓ In the United States, the social-security number attribute of a person would be a candidate key.
- ✓ The primary key should be chosen such that its attributes are never, or very rarely, changed.

E-R diagrams are used to graphically represent the logical structure of a database. An E-R diagram is drawn using the following components :

1.		Entity set	[Rectangles]
2.		Attributes	[Ellipse]
3.		Relationship	[Diamonds]
4.		Lines are used to connect attributes to an entity set and entity sets to relationship sets.	
5.		Derived attributes	[Dashed ellipses]
6.		Multi-valued attributes	[Double ellipses]
7.		Total participation	[Double lines]
8.		Key attribute	[Underlined attribute]

E-R Diagrams with Weak Entity Sets

- An entity set that does not have key attributes of its own is called a weak entity set.
- Entities belonging to a weak entity set are recognized through a relation set, relating it with an entity belonging to a strong entity set.
- **Example:** Let us consider an entity set EMPLOYEE, a list of dependents of each employee is to be maintained.
- This will need another entity set DEPENDENT, which must be related through a relationship set with the entity set EMPLOYEE.

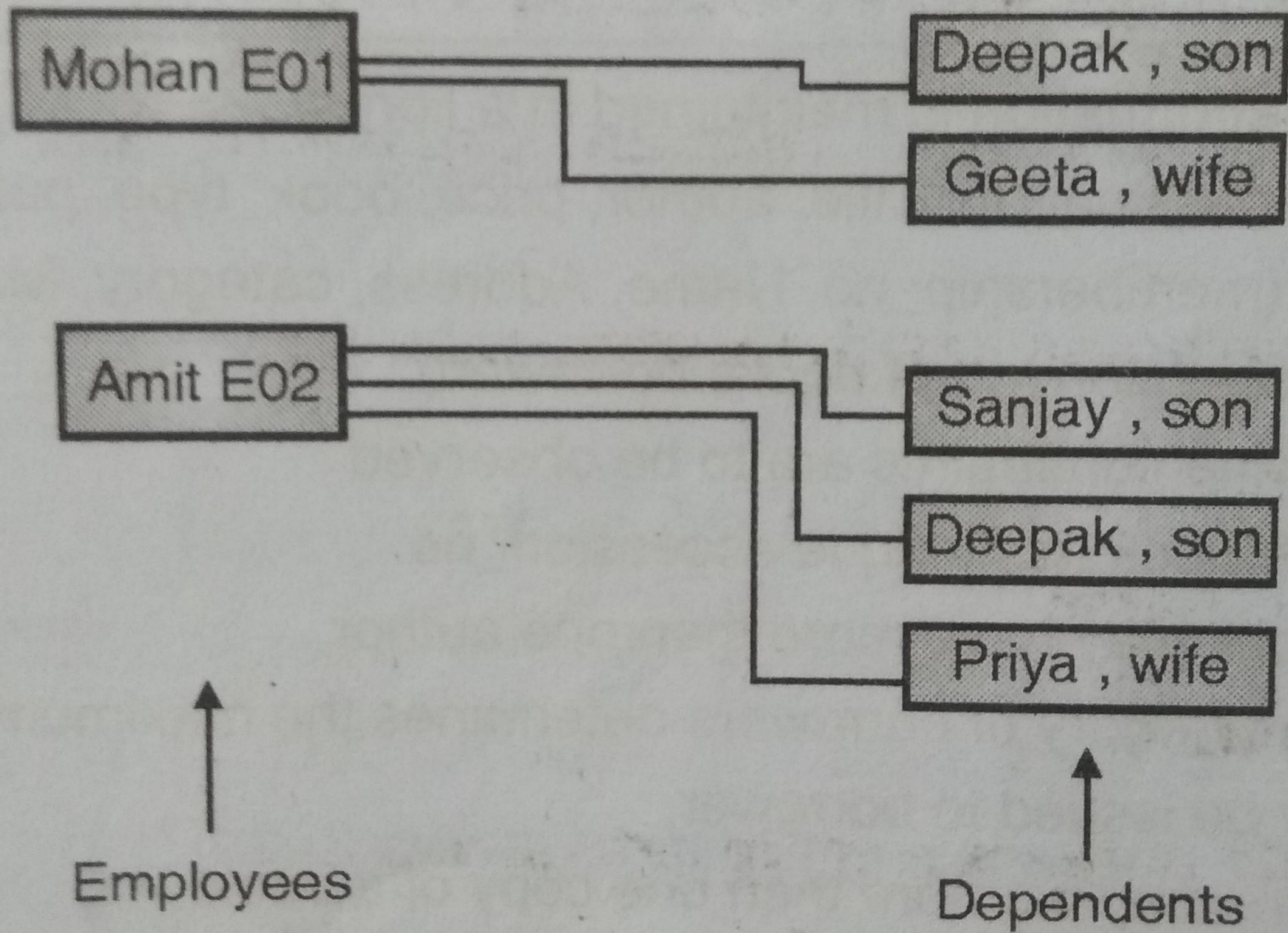
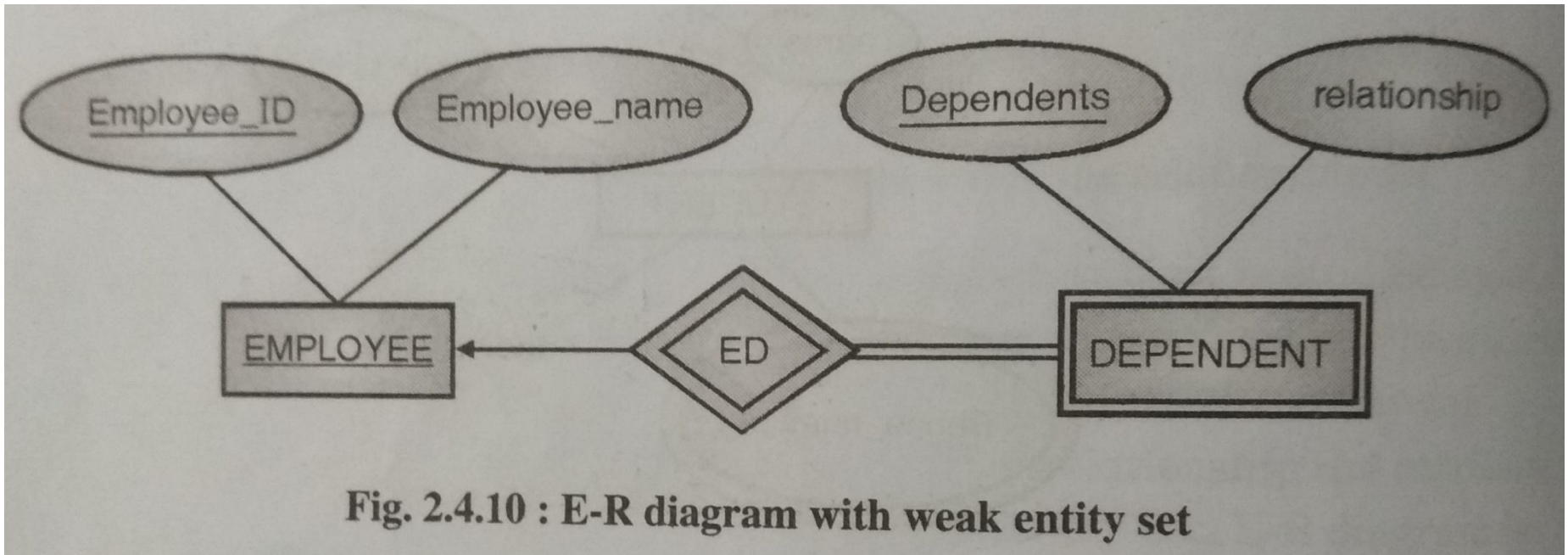


Fig. 2.4.9 : An example considered for weak entity sets

- As shown in fig. the entity set EMPLOYEE contains two entities.
- Dependents on the employees Mohan and Amit are not distinct.
- The entity set DEPENDENTS does not have a key of its own.
- The entity set DEPENDENTS will have to borrow the primary key from the entity set EMPLOYEE.

- The E-R diagram involving weak entity set is shown in following fig.



- The key for the entity set DEPENDENT will be:
(Employee_id, Dependent_name)
- A weak entity set can also be modeled as a multi-valued attribute .

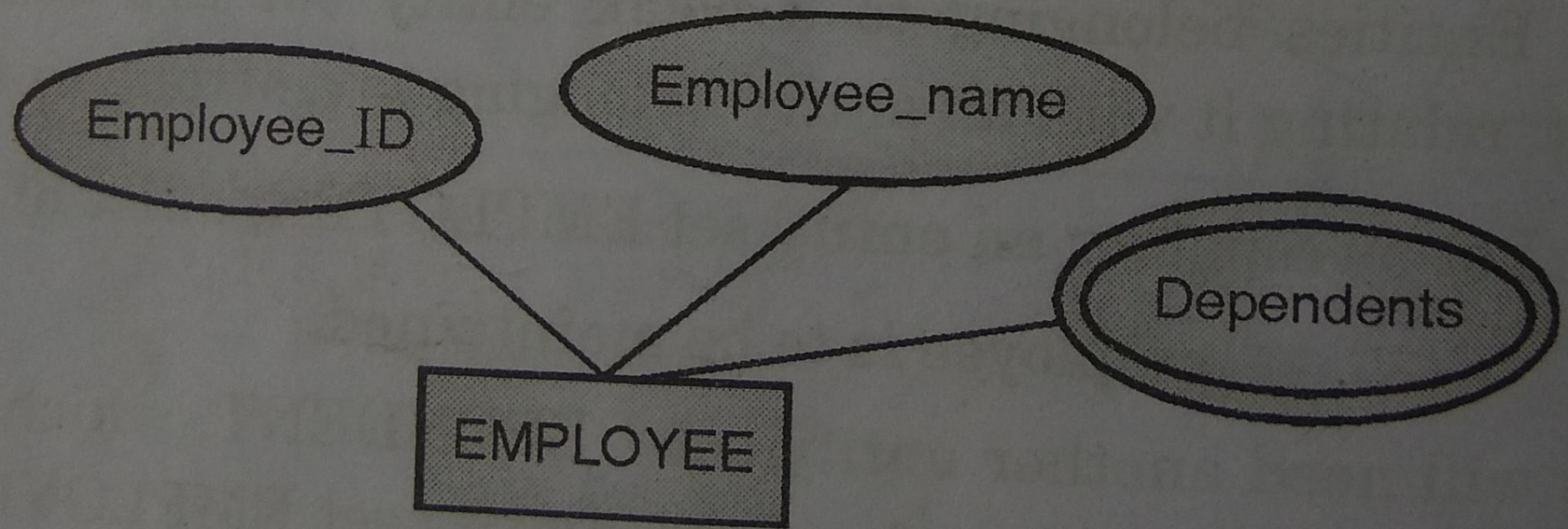


Fig. 2.4.11 : A weak entity set taken as multi-valued attribute

Extended E-R Diagrams (EER)

- Additional Concepts are added to extend E-R diagrams.
- These Concepts are:
 - 1) Specialization
 - 2) Generalization
 - 3) Aggregation
- These features are required to model complex E-R diagram.

1) Specialization:

- It is an abstracting process for introducing new attributes to an existing entity set to derive one or more entity sets.
- It involves taking a higher level entity set and using additional attributes, generating lower level entity set.
- Lower level entity set will inherit attributes of higher level entity set.

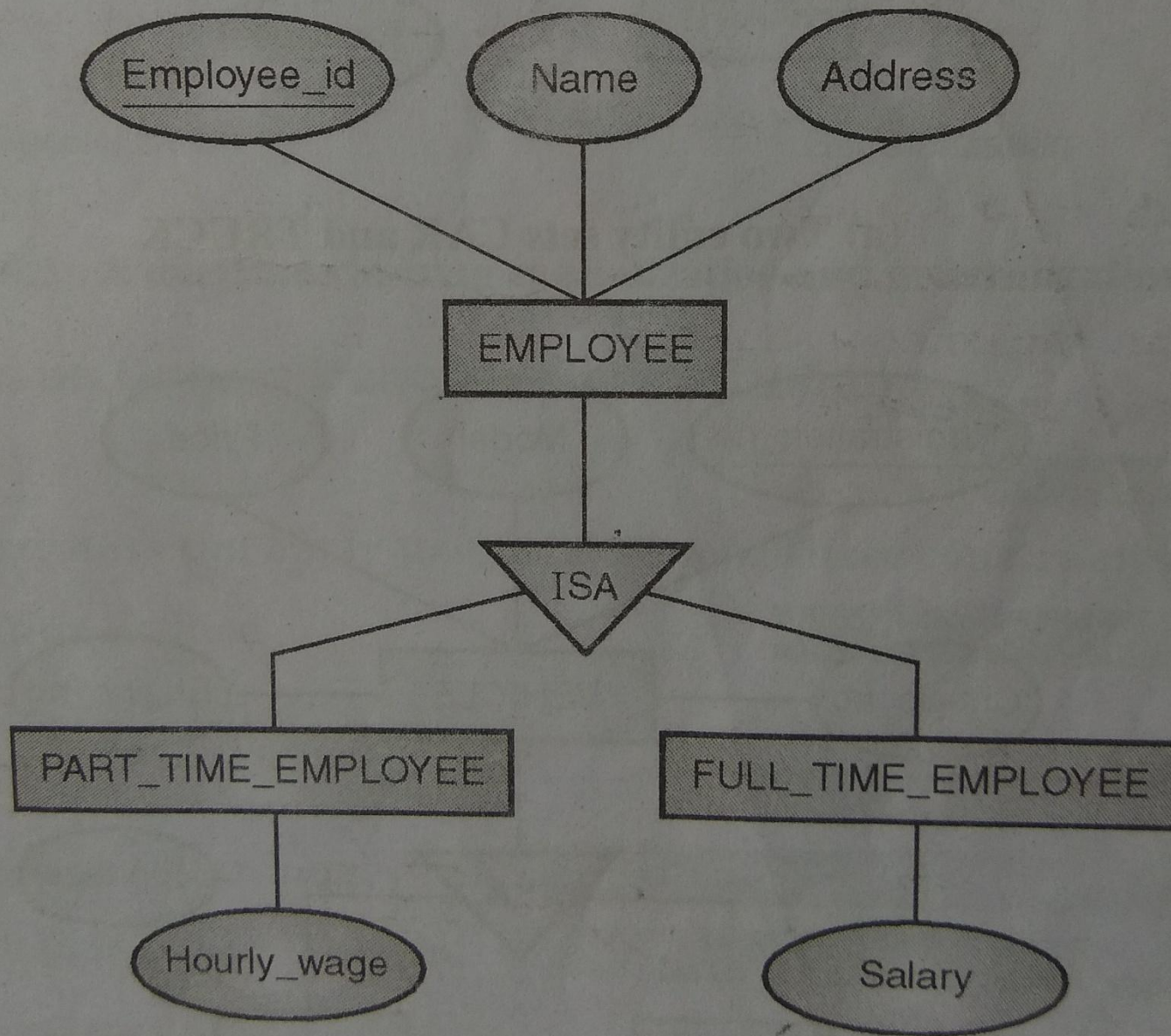


Fig. 2.5.1 : ERR diagram showing specialization

2) Generalization:

- It is a reverse of specialization.
- It is the abstracting process of creating a general class entity set by taking common attributes of constituent entity sets while ignoring their differences.
- E.g. EMPLOYEE is a general case for PART_TIME_EMPLOYEE and FULL_TIME_EMPLOYEE similarly, VEHICLE is a general case for CAR and TRUCK. (see fig.)

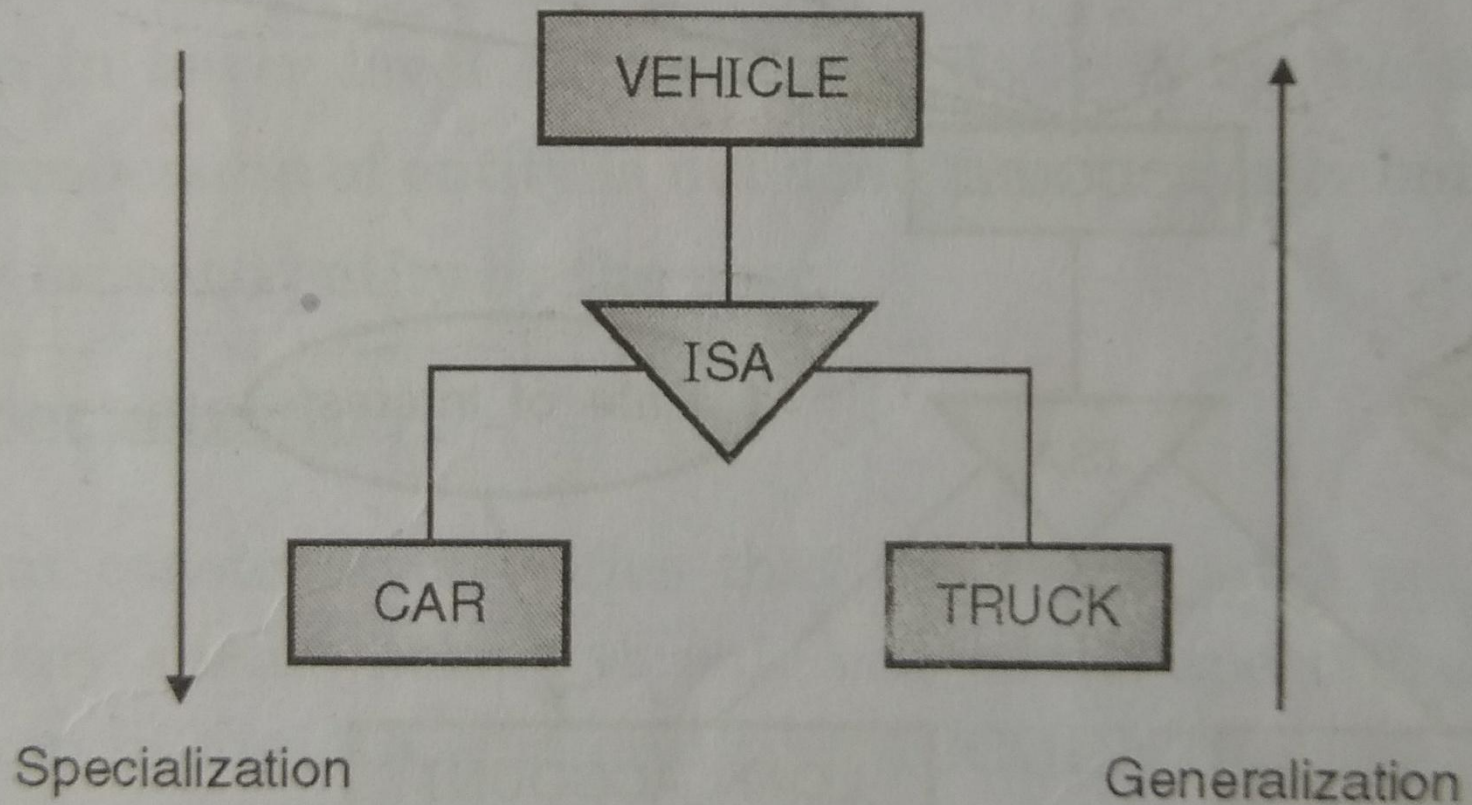


Fig. 2.5.3 : A diagram showing specialization and generalization

3) Aggregation:

- E-R models are not allowed to have a relationship between:
 1. Two relationships.
 2. A relationship set and an entity set.
- As an alternate representation, an abstract entity set is created by aggregating attributes or by treating a relationship set as an entity set.
- Thus we can define aggregation as an abstraction where a relationship set is treated as higher level entity set.

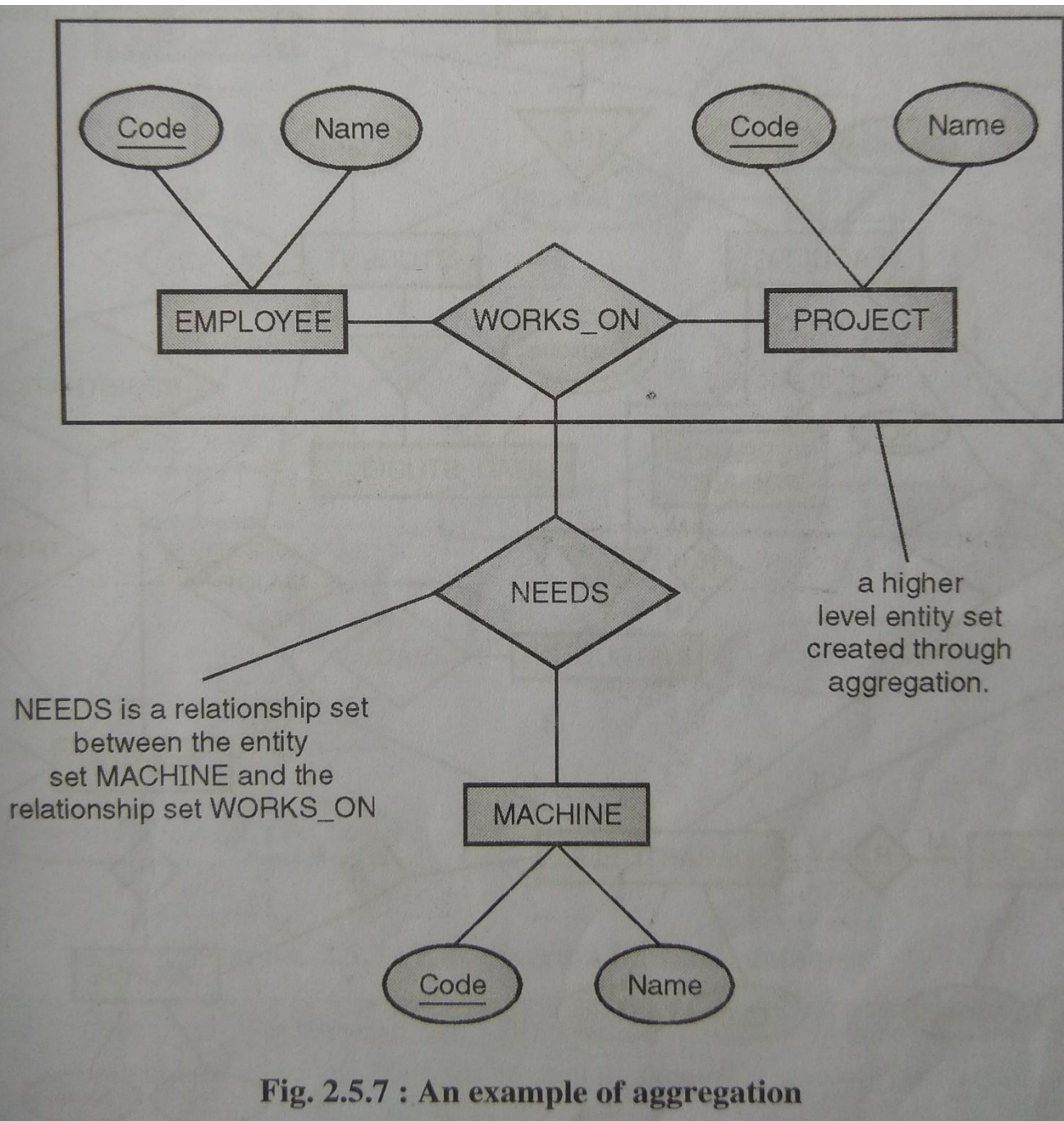


Fig. 2.5.7 : An example of aggregation

- ✓ Let us consider a case where an employee works on various projects.
- ✓ Each project requires various machines.
- ✓ A machine is assigned to an employee if the project on which he is working requires that machine.
- ✓ As shown in above diagram.