DBTG report contained specifications for three crucial database components:

- The network schema, the conceptual organization of the entire database as viewed by the database administrator (DBA). The schema included a definition of the database name, the record type of each record and the components (fields or columns) that make up the records.
- The subschema that defines the portion of the database seen by the application programs that access the database and produce information.
- A data management language to define the data characteristics and data structures and to manipulate the data.

To produce the above-mentioned standardization, the DBTG specified three distinct data management language components—schema DDL, subschema DDL and DML. The schema DDL enabled the DBA to define the schema components. The subschema DDL allowed the application programs to define the database components that will be used by the program and the DML enabled the manipulation of the database contents.

The **ANSI Standards Planning and Requirements Committee** (ANSI/SPARC) augmented the database standards in 1975. The ANSI/SPARC defined three different data models based on their degree of abstraction—conceptual, external and internal or physical. We have seen a detailed discussion of these models in Chapter 8. We have also seen that the data model that is independent of both the DBMS software and the hardware is the conceptual model. We have also mentioned that a visual representation of the conceptual model is called a conceptual schema. We also saw that the most popular conceptual model is the **Entity-Relationship** or **E-R** model and the conceptual schema associated with it is the **E-R diagram (ERD)**. We will see the E-R model and the ER diagrams in a little more detailed manner in this chapter.

E-R MODEL

The Entity–Relationship (E-R) model is a high-level conceptual data model developed by Chen in 1976 to facilitate database design. A conceptual data model is a set of concepts that describe the structure of a database and the associated retrieval and update transactions on the database. The main purpose of developing a high-level data model is to support a user's perception of the data and to conceal the more technical aspects associated with database design. Furthermore, a conceptual data model is independent of the particular DBMS and hardware platform that is used to implement the database.

The basic concepts of the ER model include entity types, relationship types and attributes. Entity type is an object or concept that is identified by the organization as having an independent existence. Entity type represents a set of objects in the real world with the same properties. An entity type has an independent existence and can be an object with a physical (real) existence such as employee, book, manager, customer, etc. or an object with a conceptual (abstract) existence like sale, inspection, satisfaction and so on. Here the important point to remember is that the above definition of entity type is only a working definition as no formal definition exists. So different designers will identify different entities in the same system.

COMPONENTS OF AN E-R MODEL

In this section we will make ourselves familiar with the components of the E-R model and their representation in the E-R diagrams. The E-R model forms the basis of the ERDs. The ERD represents the conceptual view of the database. The ERDs represent three main components—entities, attributes and relationships.

Entities

The fundamental item in any data model is the entity. An entity is viewed as the atomic real world item. An entity is an instance of an entity type that is uniquely identifiable. Each uniquely identifiable instance of an entity type is also referred to as an **entity occurrence** or **entity instance**. We identify each entity type by a name and a list of properties. A database normally contains many different entity types. Although an entity type has a distinct set of attributes, each entity has its own values for each attribute. For example a book, a publisher or a person are entities. A book is an atomic entity because it cannot be broken down into smaller pieces and still represents the real world item. This does not mean that an entity cannot be further described. As discussed earlier a book can have qualities that describe it like ISBN, Title, Author, Publisher, etc. An entity named book contains enough information to uniquely define a book. Each reference to the book entity refers to a singular representation of a book. There are numerous occurrences of books in the real world and thus the database. But in naming a general class of real world entities we use a singular form. Thus the entity describing all the books is named and called BOOK rather than books.

A data entity represents a model of a real world item specified for storage and reference in computer-readable form. The specification of a data entity is always uniquely defined in the database model of an application environment. Each entity (book, publisher, author, etc.) represents an atomic real-world item.

Attributes

As mentioned in the previous section, an entity is composed of additional information, which describes the entity. The components of an entity or the qualifiers that describe it are called attributes or data items of the entity. An attribute is a single atomic unit of information that describes something about its named entity. For example, the attributes of the entity BOOK can be ISBN, Title, Author, Publisher, Price, Year of Publication, etc. These attributes provide additional information about the BOOK entity. This additional information provides the means to uniquely define a book within the machine-usable form of the entity. In most database modeling languages the entity is singularly named and represented in uppercase characters and attributes in lowercase characters. The primary key(s) are underlined in E-R diagrams. So the BOOK entity would be represented as:

BOOK (isbn, title, author, publisher, publication_year, price)

This representation is termed as the logical description of the entity since it does not include the machine format for the entity and its attributes. A data attribute represents a computer-usable model for the components of an entity. The physical representation of an entity and its

attributes is typically defined using the concept of a named file and its physical record structure. The entity can be equated to the named file and the attributes can be equated to the records of the file. Attributes can have the same name in different entities. But within the same entity there cannot be duplicates.

As we have seen, the particular properties of entities are called attributes. For example, a student can be described by the enrollment number, name, age, sex, class, and so on. The attributes of an entity hold values that describe each entity. The values held by attributes represent the main part of the data stored in the database.

Each attribute is associated with a set of values called a **domain**. The domain defines the potential values that an attribute may hold. For example, if the age of the student in the class is between 14 and 17, then we can define a set of values for the age attribute of the student entity as the set of integers between 14 and 17.

Attributes may share a domain. For example, the date-of-birth attributes for both the teacher and student entities can share the same domain. Domains can be composed of more than one domain. For example, the domain for the date-of-birth attribute is made up of sub-domains—day, month and year. A fully developed data model includes domains for each attribute in the E-R model. We can classify attributes as following:

- Simple
- Composite
- Single-valued
- Multi-valued
- Derived

Simple Attribute

A **simple attribute** is an attribute composed of a single component with an independent existence. Simple attributes cannot be further subdivided. Examples of simple attributes include Sex, Age, Salary, etc. Simple attributes are sometimes called atomic attributes.

Composite Attribute

An attribute composed of multiple components, each with an independent existence is called a **composite attribute**. Some attributes can be further divided to yield smaller components with an independent existence of their own. For example, the Address attribute can be composed of components like Street number, Area, City, Pin code and so on. The decision to model the Address attribute as a composite attribute or subdivide the attribute into simple attributes like Street, Area, City, etc. is dependent on whether the user view of the model refers to the Address attribute as a single unit or as individual components.

Single-valued Attribute

A ***single-valued attribute*** is one that holds a single value for a single entity. The majority of attributes are single-valued for a particular entity. For example, the Classroom entity has as single value for the Room_number attribute and therefore the Room_number attribute is referred to as being single-valued.

Multi-valued Attribute

A **multi-valued attribute** is one that holds multiple values for a single entity. Some attributes have multiple values for a particular entity. For example, a Student entity can have multiple values for the Hobby attribute—reading, music, movies and so on. A multi-valued attribute may have set of numbers with upper and lower limits. For example, the Hobby attribute of a Student may have between one and five values. In other words, a student may have a minimum of one hobby and maximum of 5 hobbies.

Derived Attribute

A **derived attribute** is one that represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity. Some attributes may be related for a particular entity. For example the Age attribute can be derived from the date-of-birth attribute and therefore they are related. We refer the Age attribute as a derived attribute, the value of which is derived from the date-of-birth attribute.

E-R Diagram Conventions

There are conventions for representing the entities and attributes in the E-R diagram. These conventions are given below and shown in Figure 9.1:

- The **entities** are represented by a rectangular box with the name of the entity in the box.
- An **attribute** is shown as an ellipse attached to a relevant entity by a line and labeled with the attribute name.
- The **entity name** is written in uppercase where as the **attribute name** is written in lowercase.
- The **primary keys** (key attributes) are underlined.
- The attributes are connected using lines to the entities. If the attribute is simple or single valued a **single line** is used.
- If the attribute is derived a **dotted line** is used.
- If it is multi-valued then **double lines** are used.
- If the attribute is **composite**, its component attributes are shown as ellipses emanating from the composite attribute.

Relationships

The entity-attribute definitions only capture the static meaning of the real world items (entities). But in the real world, items have relationships to one another. For example, a book is published by a particular publisher, an employee may work for a manager, or a person has a child and the child has a cousin, etc. The association or relationships that exists between the entities relates data items to each other in a meaningful way. This information relationship must be captured by the database schema or model, if the resulting database is to be a reasonable approximation of the real world entities that it models.

The existence of data associations defines that a relationship exists between two or more entities. Relationships and associations are often used interchangeably to define this condition. In most database models the relationship between two or more entities is captured through the use of relationship qualifiers, logical pointer structures, active functions, inference rules, etc. But the most common way to represent the relationship between entities is to use an additional entity called the relationship entity. For instance, in the book-distributor-order example, the books are ordered from the distributor. To represent the relationship we form a relationship entity called ORDERS. The order entity has attributes for the BOOK and the DISTRIBUTOR from whom the books are ordered. In addition the relationship entity could have attributes of its own. For example the order quantity is an attribute of the ORDER entity. Relationships are thus used to aid the database in constructing how data are to be used within the database.

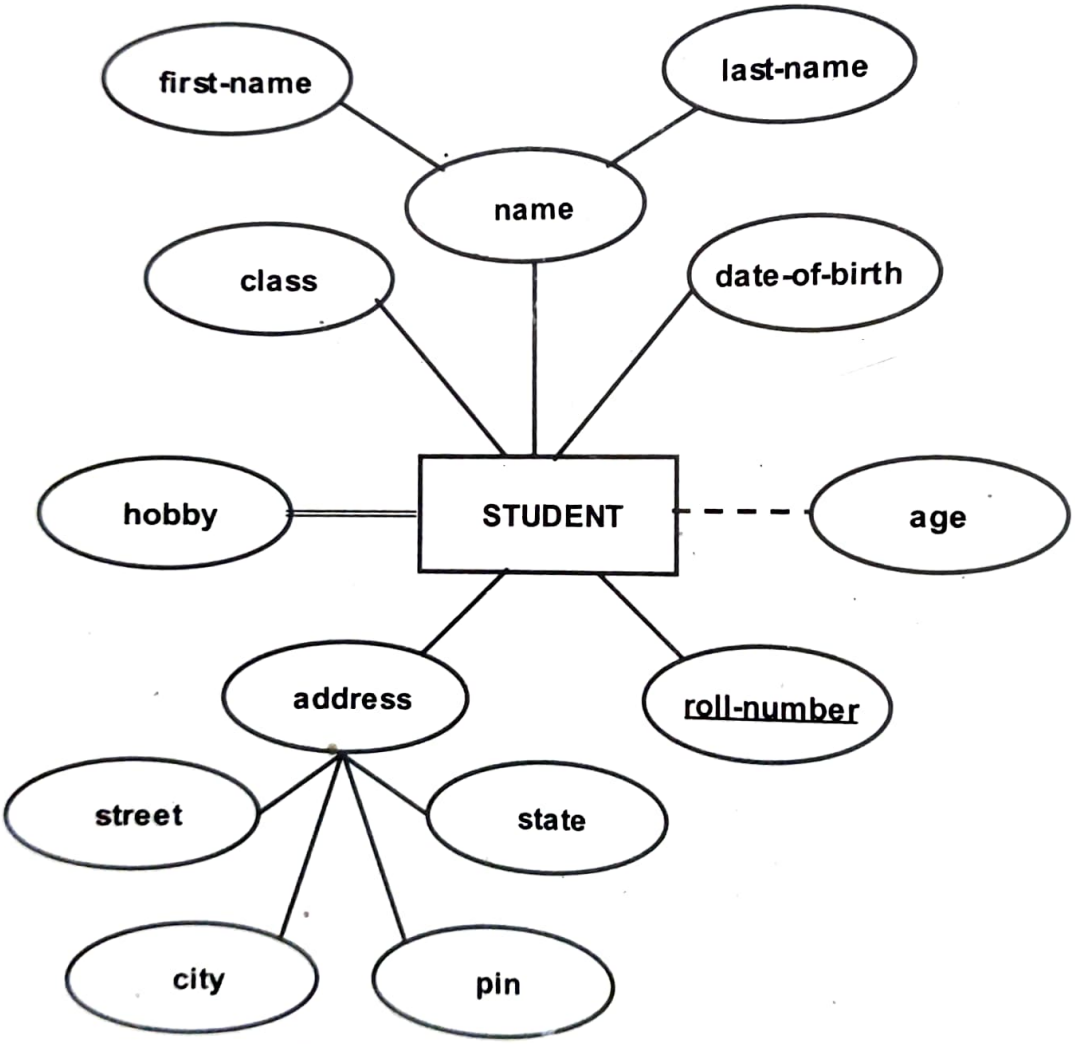


Figure 9.1 **ERD for the Student entity**

A relationship is an association between entities. Each relationship is identified so that its name is descriptive of the relationship. Usually the relationship name is an active verb, but passive verbs are also used. Relationships are represented by diamond shaped symbols with the relationship name inside the diamond. The two sides of the diamond are connected to the entities

elates. For example the Figure 9.2 shows the relationship between the entities PUBLISHER and OOK.

Figure 9.2 **PUBLISHER – BOOK Relationship**

Now we will some terms associated with entities and relationships. They are degree, onnectivity, cardinality, dependency and participation.

Degree

The **degree** of a relationship indicates the number of associated entities. A **unary** elationship exists when an association is maintained within a single entity. For example, consider he entity SUBJECT. Consider a situation where to take a particular subject, you have to take nother subject (like you should take the Mathematics to take the Statistics, or English Grammar to ake the English Drama and so on). This kind of a relationship is called a unary relationship and is epresented as shown in Figure 9.3.

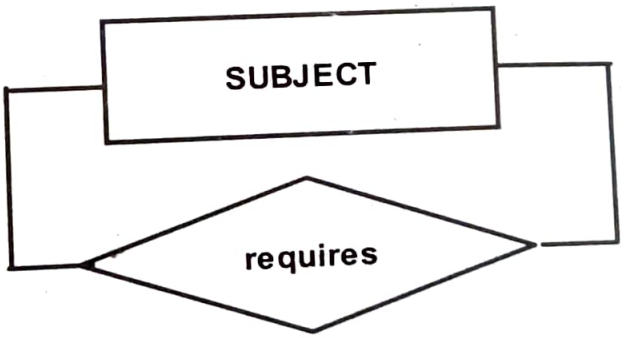


Figure 9.3 **Unary Relationship**

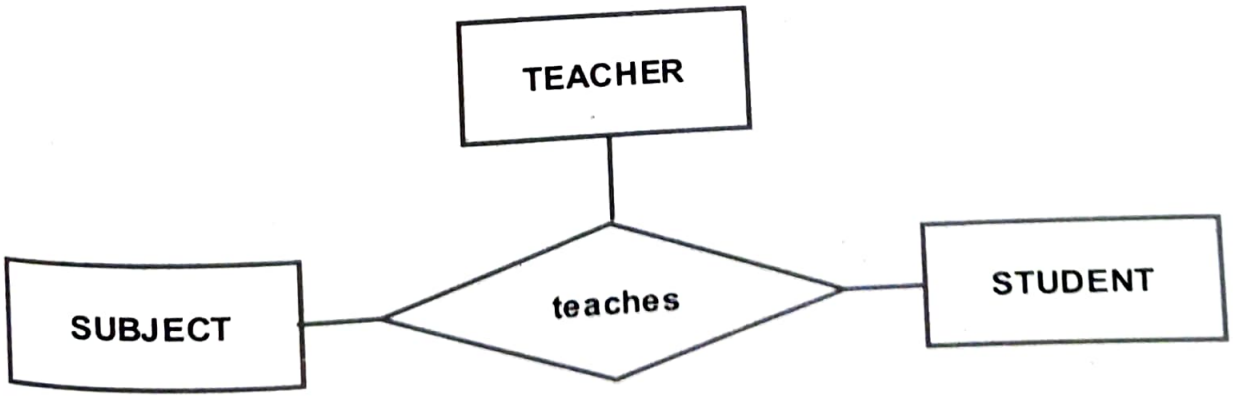


Figure 9.4 **A Ternary Relationship**

A **binary** relationship exists when two entities are associated. For example, the BOOK – PUBLISHER relationship shown in Figure 9.2 is a binary relationship. A **ternary** relationship exists when there are three entities associated. For example, the entities TEACHER, SUBJECT and

STUDENT are related using a ternary relationship called 'teaches'. This is illustrated in the Figure 9.4. A **quaternary** relationship exists when there are four entities associated. An example of a quaternary relationship is 'studies' where four entities are involved—STUDENT, TEACHER, COURSE MATERIAL and SUBJECT. This relationship (shown in Figure 9.5) represents a situation where a STUDENT taught by a TEACHER with the help of the COURSE_MATERIAL studies a SUBJECT.

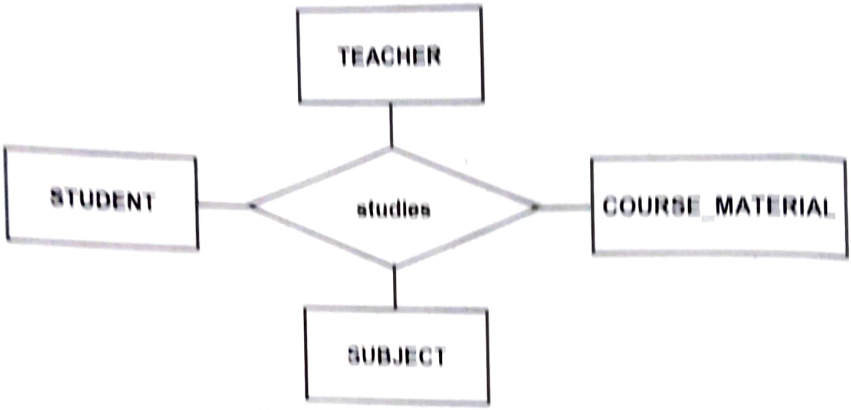


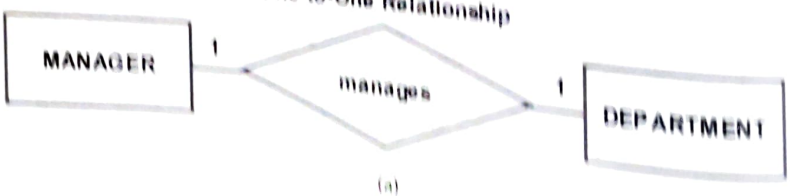
Figure 9.5 **A Quaternary Relationship**

Although higher degrees exist, they are rare and are not specifically named. Ternary and quaternary relationships are not always equivalent to a group of one-to-many relationships. The database designer must therefore consider the semantics of the problem being addressed; desirable simplifications need not always match the users' needs.

Connectivity

Relationships can be classified as one-to-one, one-to-many and many-to-many. The term **connectivity** is used to describe this relationship classification. The E-R diagram indicates the relationship's connectivity by placing a 1, M or N near the related entities as shown in Figures 9.6(a), 9.6(b) and 9.6(c). The relationships are explained as follows: A MANAGER in the company manages a single DEPARTMENT (a one-to-one connectivity). A DEPARTMENT can have more than one EMPLOYEE (a one to many connectivity). The training department of the company offers many courses to the employees as a part of the employee development program. Each EMPLOYEE can join for more than COURSE (a many to many connectivity).

One-to-One Relationship



(a)

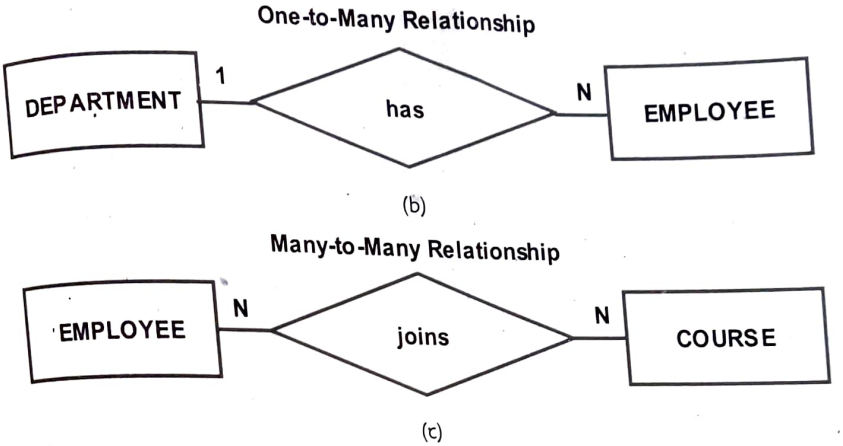


Figure 9.6 **Connectivity in Relationships**

Cardinality

Cardinality expresses the specific number of entity occurrences associated with one occurrence of the related entity. The actual number of associated entities is governed by the business rules.

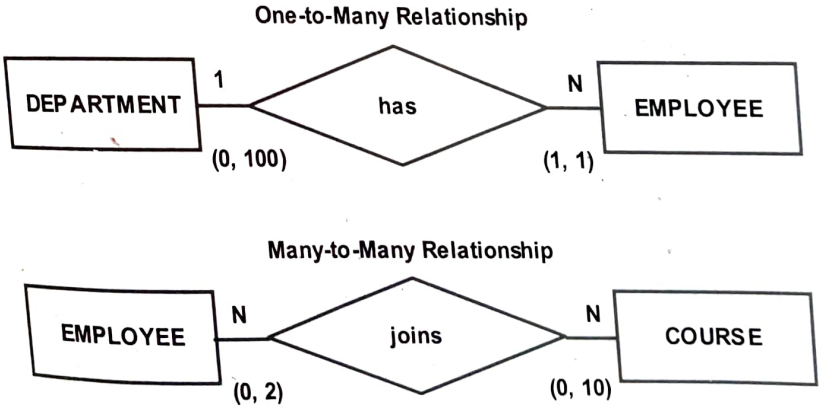


Figure 9.7 **Cardinality in Relationships**

For example, the company policy does not allow more than 100 employees in a department. Therefore the cardinality rule governing the DEPARTMENT-EMPLOYEE relationship is expressed as "One department can have a maximum of 100 employees." Also the company policy allows each employee to join a maximum of 2 courses at a time. At present the company offers 10 courses. The cardinality is indicated by placing the appropriate numbers beside the entities as shown in Figure 9.7.

Here an important point that should be noted is that the number of associated entities may be variable. For example an employee can enroll for one or two courses at a time. He also

has the choice of not enrolling for any of the courses. Similarly a department can have a maximum of 100 employees. But it is quite possible that the number of employees in a department is 0, if the department is during its formative stages.

Dependency

Entities are classified as being **strong** or **weak** entity types. An entity type that is existence-dependent on some other entity is called a weak entity type or existence-dependent and an entity type that is not existence-dependent on some other entity type is called a strong entity type. A weak entity type is dependent on the existence of another entity. Weak entities are also referred to as **child**, **dependent** or **subordinate** entities and strong entities as **parent**, **owner** or **dominant** entities. For example in the following relationship PARENT is a weak entity, as it needs the entity EMPLOYEE for its existence. The entities EMPLOYEE, COMPANY, etc, are strong entities. We represent the weak entities using a double-lined rectangle as shown in Figure 9.8.

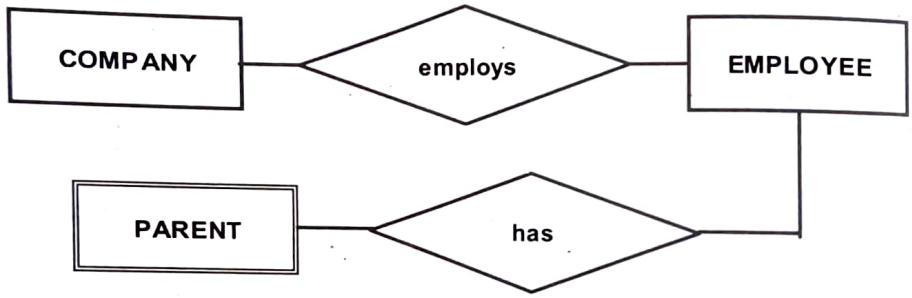


Figure 9.8 **Weak and Strong Entities**

Participation

There are two ways an entity can participate in a relationship—totally or partially. The participation is also known as mandatory or optional. The participation is total (mandatory) if an entity's existence requires the existence of an associated entity in a particular relationship. The participation is said to be optional or partial, if the occurrence of one entity does not require the occurrence of another corresponding entity in a relationship.

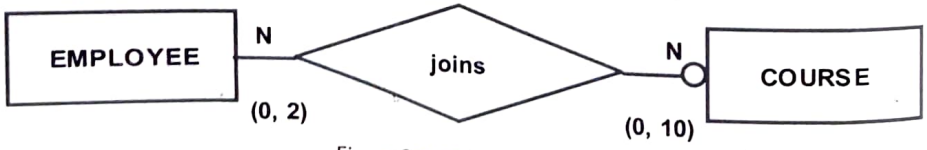


Figure 9.9 **Optional Entities**

We have seen in an earlier example that the EMPLOYEE can take up to two executive development courses, but he need not take even one. So in the relationship "EMPLOYEE joins COURSE," it is quite possible for an EMPLOYEE not to join a COURSE. Therefore COURSE is optional to the EMPLOYEE. But a COURSE must be attended by an EMPLOYEE, which makes the

MPLOYEE a mandatory entity. An optional entity is shown by drawing a small circle (O) on the de of the optional entity as shown in Figure 9.9.

It is important that you fully understand whether an entity is optional or mandatory in a lationship. Failure to do so may result in designs in which unnecessary temporary entities are eated. Also keep in mind that the terms mandatory and optional refer to the participation of an tity within the context of a relationship with another entity.

The term optional refers to a condition in which the other participating entity may or may ot be associated with occurrences of the optional entity in the relationship. But the term andatory refers to a condition in which one participating entity must be associated with one or ore occurrences of the other participating entity in the relationship.

We will now summarize what we have learned about entities and relationships using a gle example. As we have seen, relationships are represented using connecting lines. The lines are rected depending on the type of the relationship and the type of the relationship will be written the side of the line.

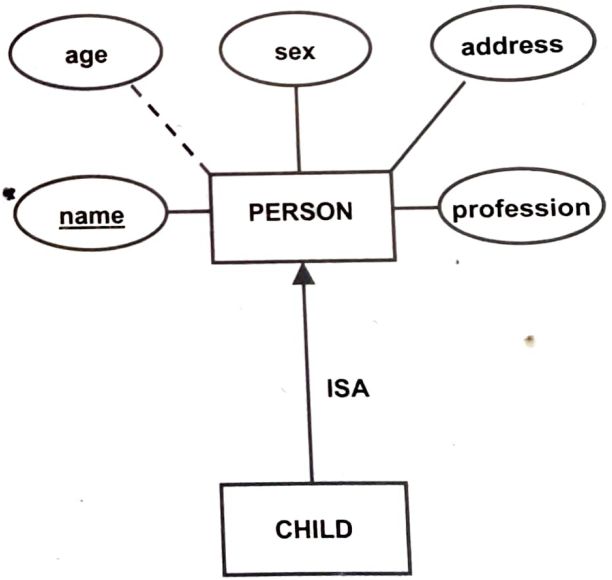


Figure 9.10 **E-R Diagram showing an ISA Relationship**

For example (see Figure 9.10) a relationship line with ISA written by the side of it indicates hat the relationship type is ISA. An ISA relationship implies that the subtype must have a matching uper-type to exist. For example a CHILD entity should be defined in the PERSON entity before it can exist as a CHILD.

The relationships are depicted in the data map using lines with added information to describe the cardinality of the relationship like one-to-one, one-to-many, many-to-many, etc. Example of a one-to one relationship is the relationship between the entities EMPLOYEE and CONSULTANT. Here consultant is a special type of the entity EMPLOYEE—an ISA relationship. Similarly there are one-to-many and many-to-many relationships.

For example, the relationship between DEPARTMENT and EMPLOYEE is a one-to-many relationship (assuming that an employee works for only one department and a department can have more than one employee).

An example of the many-to-many relationship is the one between AUTHOR and PUBLISHER. An author can have more than one publisher and a publisher will be publishing books of more than one author.

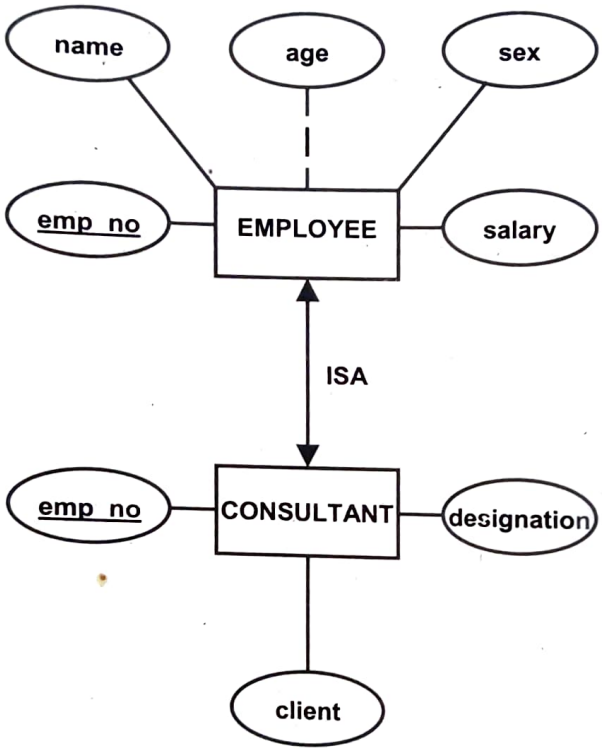


Figure 9.11 **One-to-one Relationship**

Now if we make up the entity lists of these entities, it will be something like:

- EMPLOYEE (emp_no, name, age, sex, salary)
- CONSULTANT (emp_no, client, designation)
- DEPARTMENT (dept_id, name)
- AUTHOR (name, age, sex, address, phone_number)
- PUBLISHER (name, address, editor)

The above-mentioned relationships are represented in a data maps in figures 9.11, 9.12 and 9.13. You can either indicate a many relationship using multiple lines (as indicated in the one-to-many relationship—Figure 9.12) or using double arrows (as indicated in the case of many-to-many relationship—Figure 9.13).

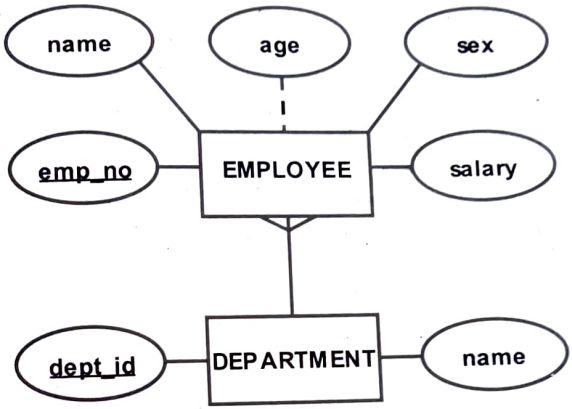


Figure 9.12 **One-to-many Relationship**

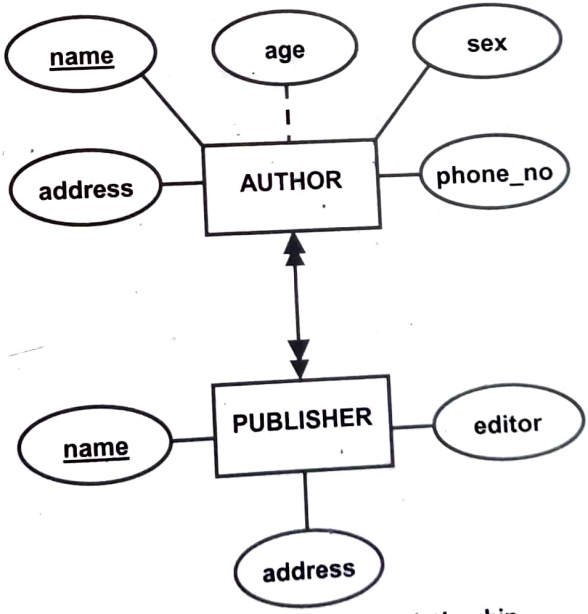


Figure 9.13 **Many-to-many Relationship**

To completely represent the relationships and constraints we need some ad features for the data map. For example if you want to specify that a relationship is man then you will need additional symbols.

For example in the case of the ISA relationship between the EMPLOYEE a CONSULTANT entities, the CONSULTANT entity cannot exist without the EMPLOYEE, EMPLOYEE can exist without the CONSULTANT. We will represent this as in figure9.14. V also seen that to make the data map complete we need to incorporate such details like ca

connectivity and so on. As an exercise, complete the Employee-Consultant incorporating all those details.

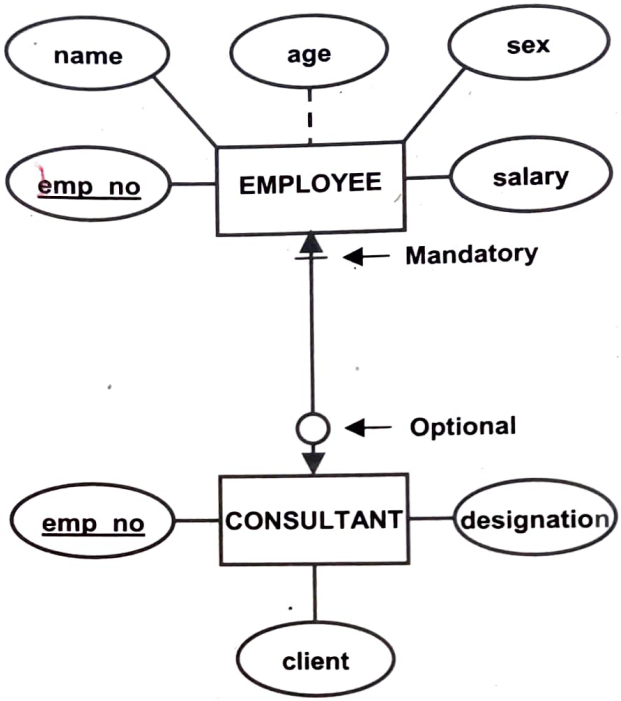


Figure 9.14 **Representation of Employee—Consultant relationship**

Composite Entities

When Chen developed the E-R model in 1975, the relationships did not contain attributes. But when we want to translate the E-R model into a relational database model, we need to transform the E-R models into one-to-many relationships, as the relational model requires the use of one-to-many relationships.

Figure 9.15 **AUTHOR-PUBLISHER Relationship**

So the many-to-many relationships in the E-R model must be broken up into one-to-many relationships. This is done using a composite entity. The composite entity is composed of (among other attributes) the primary keys of each of the entities to be connected. Composite entities are represented using a diamond shape within a rectangle.