



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-readable 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-readable 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 a 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 whereas 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.

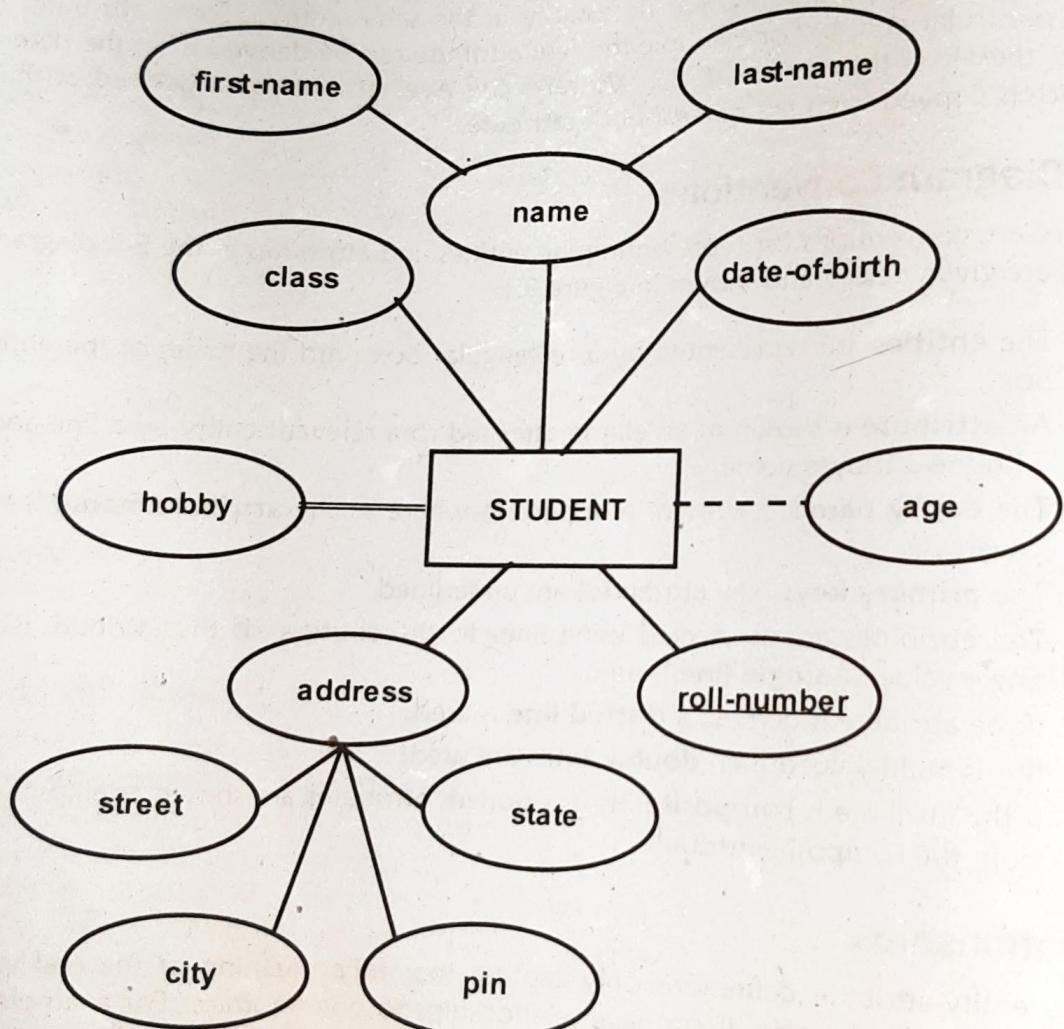


Figure 9.1 ERD for the Student entity

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 relationship name inside the diamond. The two sides of the diamond are connected to the entities

it relates. For example the Figure 9.2 shows the relationship between the entities PUBLISHER and BOOK.

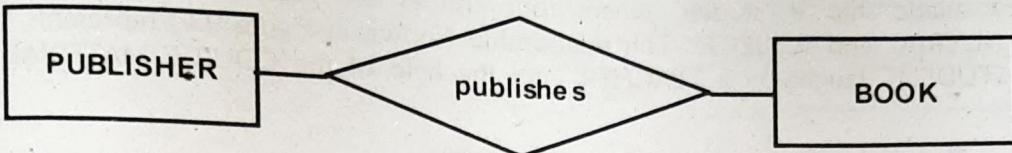


Figure 9.2 **PUBLISHER – BOOK Relationship**

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

Degree

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

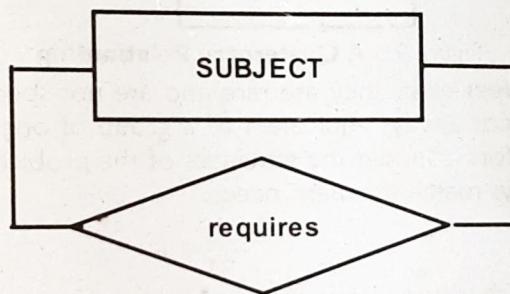


Figure 9.3 **Unary Relationship**

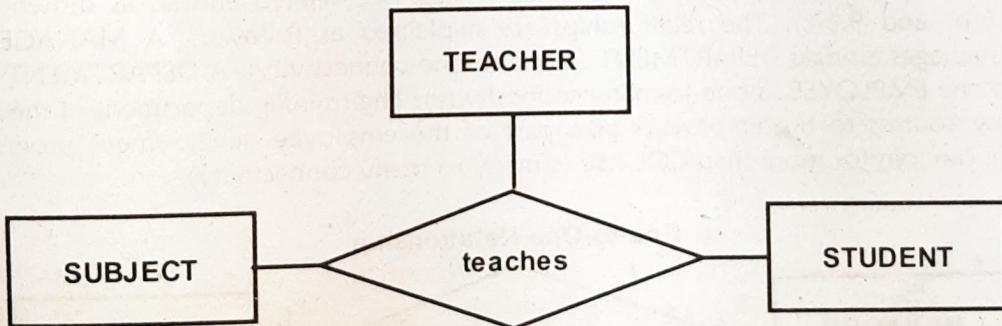


Figure 9.4 **A Ternary Relationship**

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 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 SUBJECT.

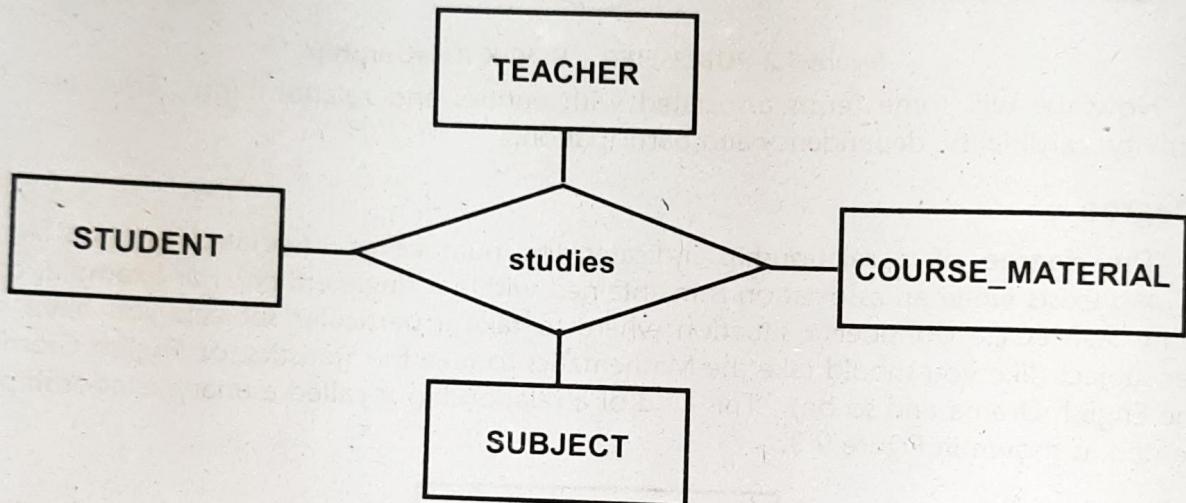
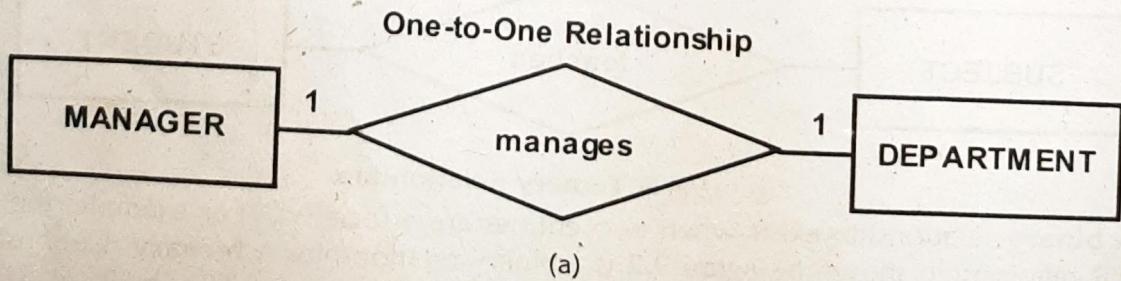


Figure 9.5 A Quaternary Relationship

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 Figure 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).



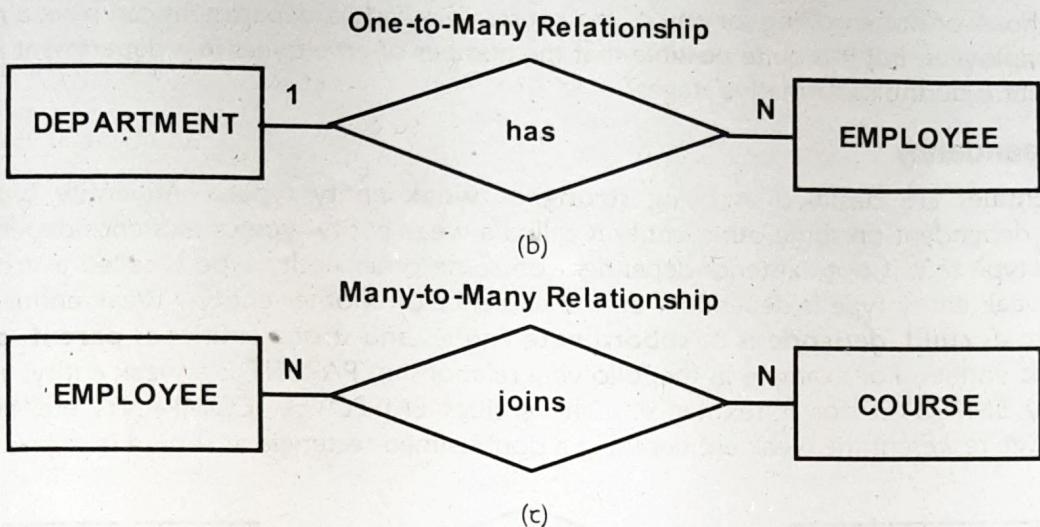


Figure 9.6 Connectivity in Relationships

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.

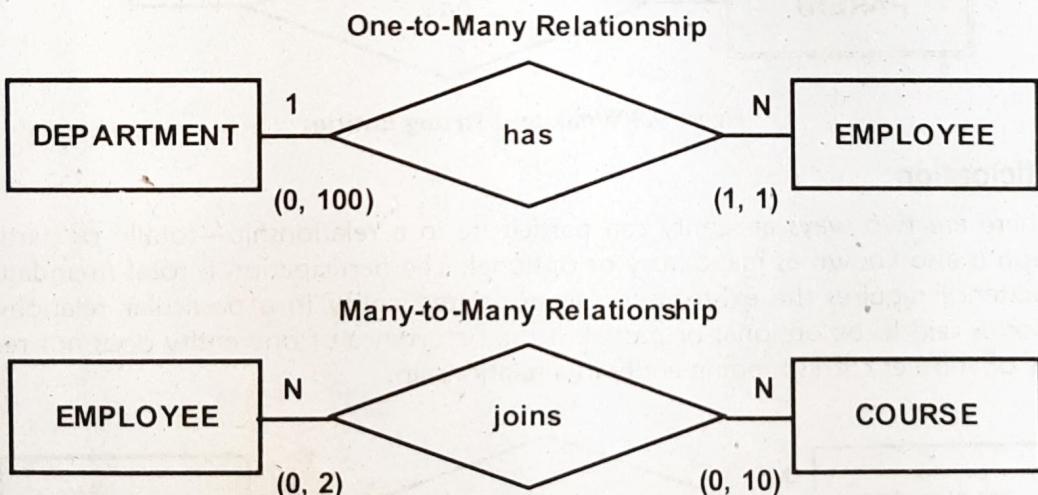


Figure 9.7 Cardinality in Relationships

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 COMPANY, COMPANY, etc, are strong entities. We represent the weak entities using a double-lined rectangle as shown in Figure 9.8.

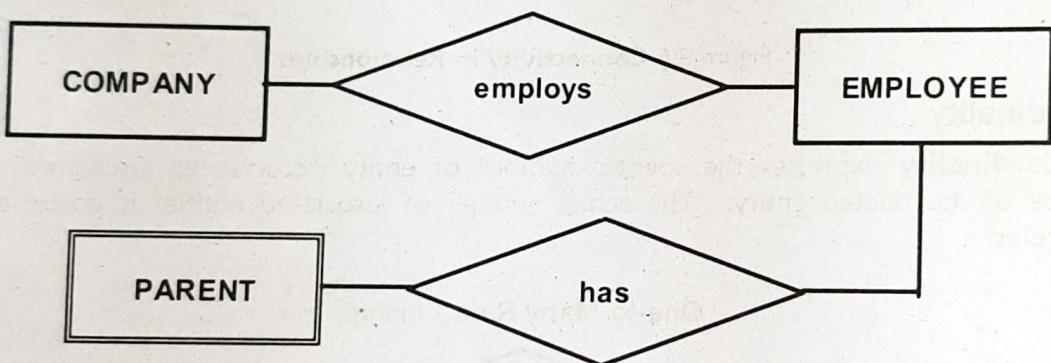


Figure 9.8 Weak and Strong Entities

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.

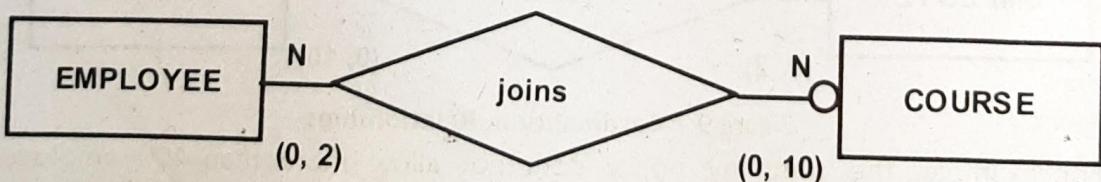


Figure 9.9 Optional Entities

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

EMPLOYEE a mandatory entity. An optional entity is shown by drawing a small circle (O) on the side 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 relationship. Failure to do so may result in designs in which unnecessary temporary entities are created. Also keep in mind that the terms mandatory and optional refer to the participation of an entity 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 not be associated with occurrences of the optional entity in the relationship. But the term mandatory refers to a condition in which one participating entity must be associated with one or more occurrences of the other participating entity in the relationship.

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

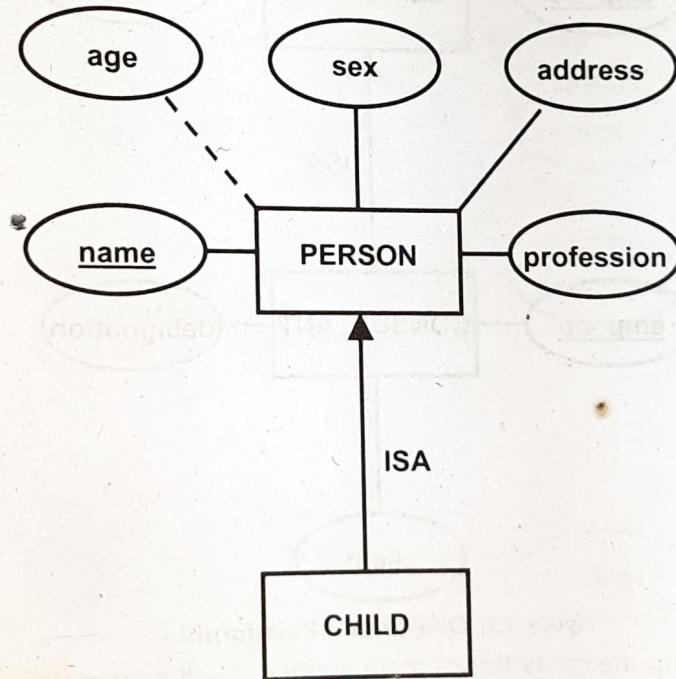


Figure 9.10 E-R Diagram showing an ISA Relationship

For example (see Figure 9.10) a relationship line with ISA written by the side of it indicates that the relationship type is ISA. An ISA relationship implies that the subtype must have a matching super-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.

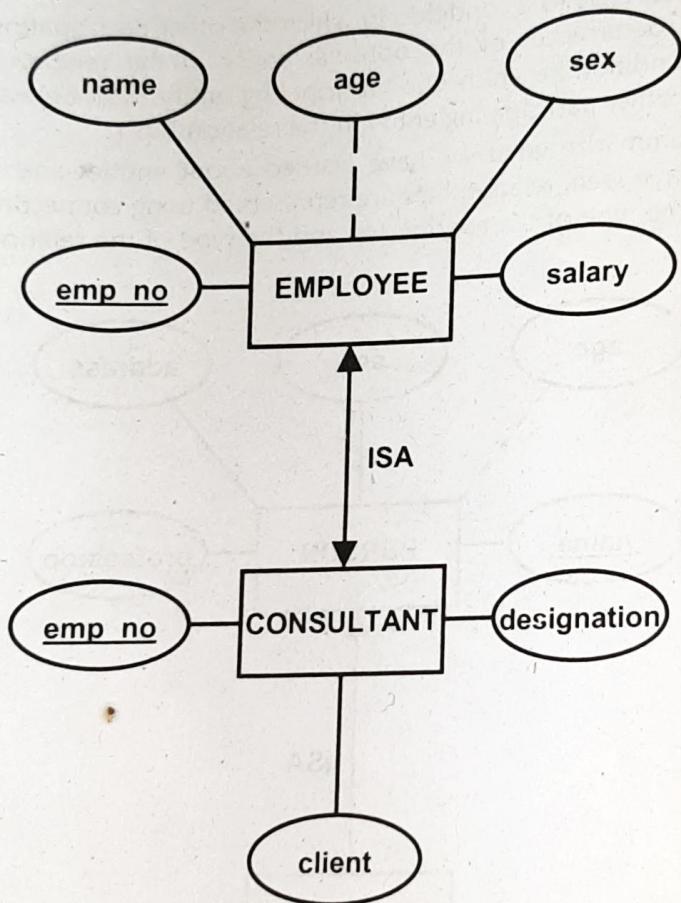


Figure 9.11 One-to-one Relationship

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).

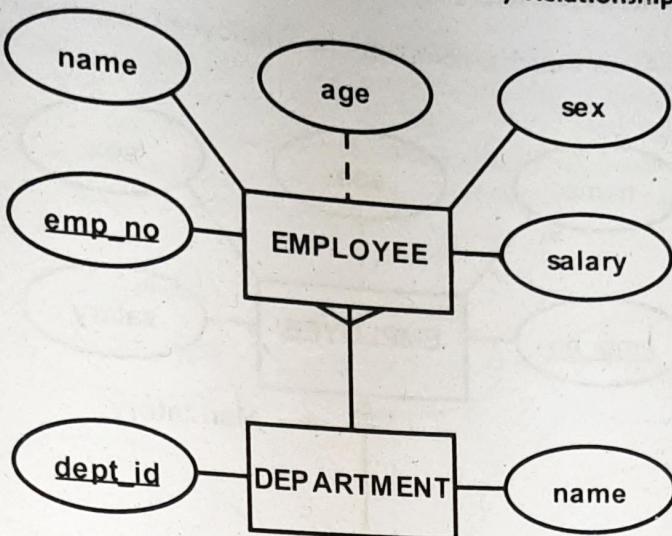


Figure 9.12 One-to-many Relationship

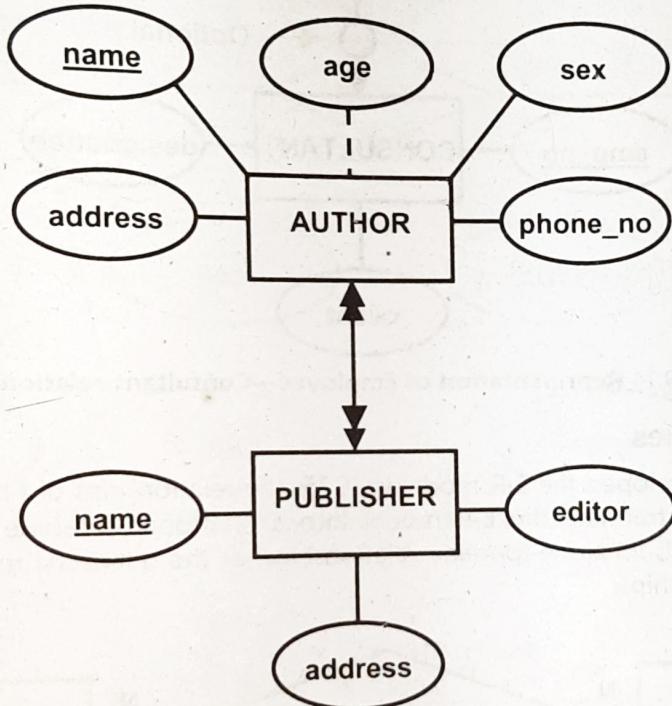


Figure 9.13 Many-to-many Relationship

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

For example in the case of the ISA relationship between the EMPLOYEE and the CONSULTANT entities, the CONSULTANT entity cannot exist without the EMPLOYEE, but the EMPLOYEE can exist without the CONSULTANT. We will represent this as in figure 9.14. We have also seen that to make the data map complete we need to incorporate such details like cardinality,

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

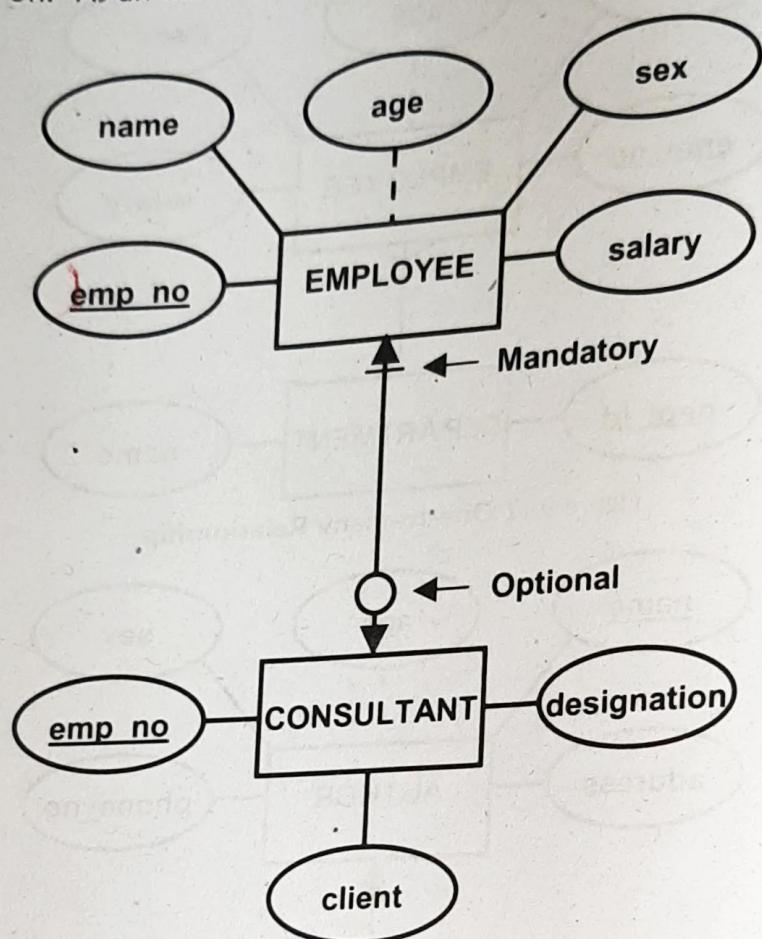


Figure 9.14 Representation of Employee—Consultant relationship



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.

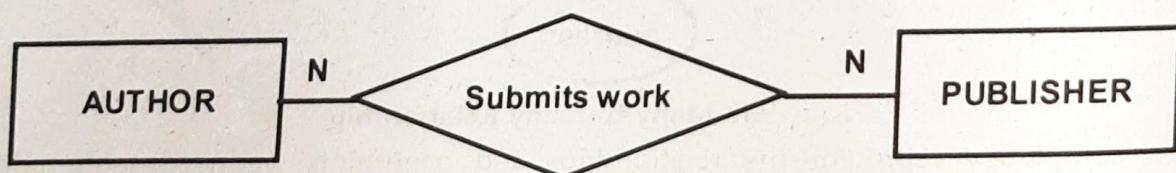


Figure 9.15 AUTHOR-PUBLISHER Relationship

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.

Consider the example of the entities AUTHOR and PUBLISHER. An author can write for many-to-many relationship as represented in Figure 9.15. But this relationship cannot be transformed into relational tables as we require one-to-many relationships.

We solve this by having a composite entity called WORK (which has among others, the primary keys of the AUTHOR and PUBLISHER tables as its attributes). The E-R diagram is shown in Figure 9.16.

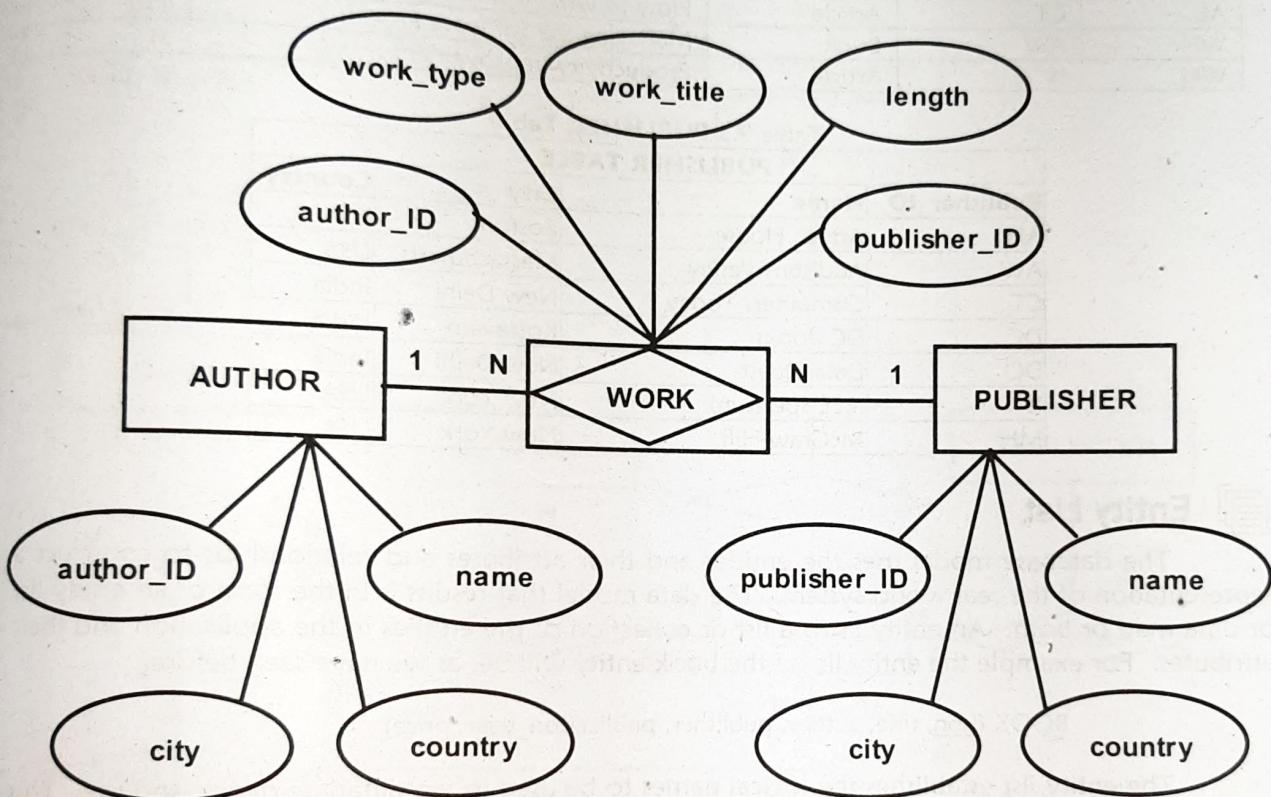


Figure 9.16 Composite Entity WORK

Now we will see the tables that are created using the above relationship—AUTHOR, WORK and PUBLISHER.

Table 9.1 AUTHOR Table

AUTHOR TABLE			
Author_ID	Name	City	Country
AD	Alan M. Davis	Colorado	USA
AL	Alexis Leon	Cochin	India
AR	Arundhati Roy	New Delhi	India
ML	Mathews Leon	Cochin	India
WH	Watts S. Humphrey	Florida	USA
RE	Ramez Elmasri	Texas	USA

Table 9.2 WORK Table

WORK				Length
Author_ID	Publisher_ID	Work_Type	Work_Title	
AD	MH	Book	201 Principles of Software Engineering	240
AL	AH	Book	A Guide to SCM	384
ML	DQ	Article	Demystifying E-business	15
AR	DC	Article	End of Imagination	45
AL	CT	Article	How to write a winning resume	30
WH	AW	Book	Managing the Software Process	450
WH	IS	Article	Productivity Improvement	31

Table 9.3 PUBLISHER Table

PUBLISHER TABLE			
Publisher_ID	Name	City	Country
AH	Artech House	Boston	USA
AW	Addison-Wesley	Massachusetts	USA
CT	Computers Today	New Delhi	India
DC	DC Books	Kottayam	India
DQ	Data Quest	New Delhi	India
IS	IEEE Spectrum	New York	USA
MH	McGraw-Hill	New York	USA



Entity List

The database model uses the entities and their attributes and relationships to construct a representation of the real world system. The data model that results is in the form of an entity list or data map or both. An entity list is a list or collection of the entities in the application and their attributes. For example the entity list of the book entity will be, as we have seen before:

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

The entity list establishes the logical names to be used in the database design and use. This list alone will not capture all the relevant and required information need for the database design. The intended use of each entity in the application, their relationship to one another and the intended use of the relationship entities should also be captured. To capture this information we may have to use additional methods or tools like a data map.



E-R Diagrams (ERDs)

E-R diagram (ERD) or data schema map or data map is another means of capturing the data and their organization. The ERD captures the entities, their attributes and the relationships between the entities. The ERD pictorially represents the interrelationships of the database schema. The major components of an ERD are entities, attributes, relationships and cardinality of the association.

Now we will see the E-R diagram of the book club. The book club has members. The book club sells books to its members. The members places orders for books, which the book club

fulfills. Each order contains one or more than one books. The books are written by author(s). The publisher publishes the book.

An author can write more than one book and a book can have more than one author. A book is published by a publisher, but a publisher publishes many books. A member can place more than one order. He also can choose not to place an order. The book club sells many books. The above book club example is converted into an E-R diagram and is shown in Figure 9.17.

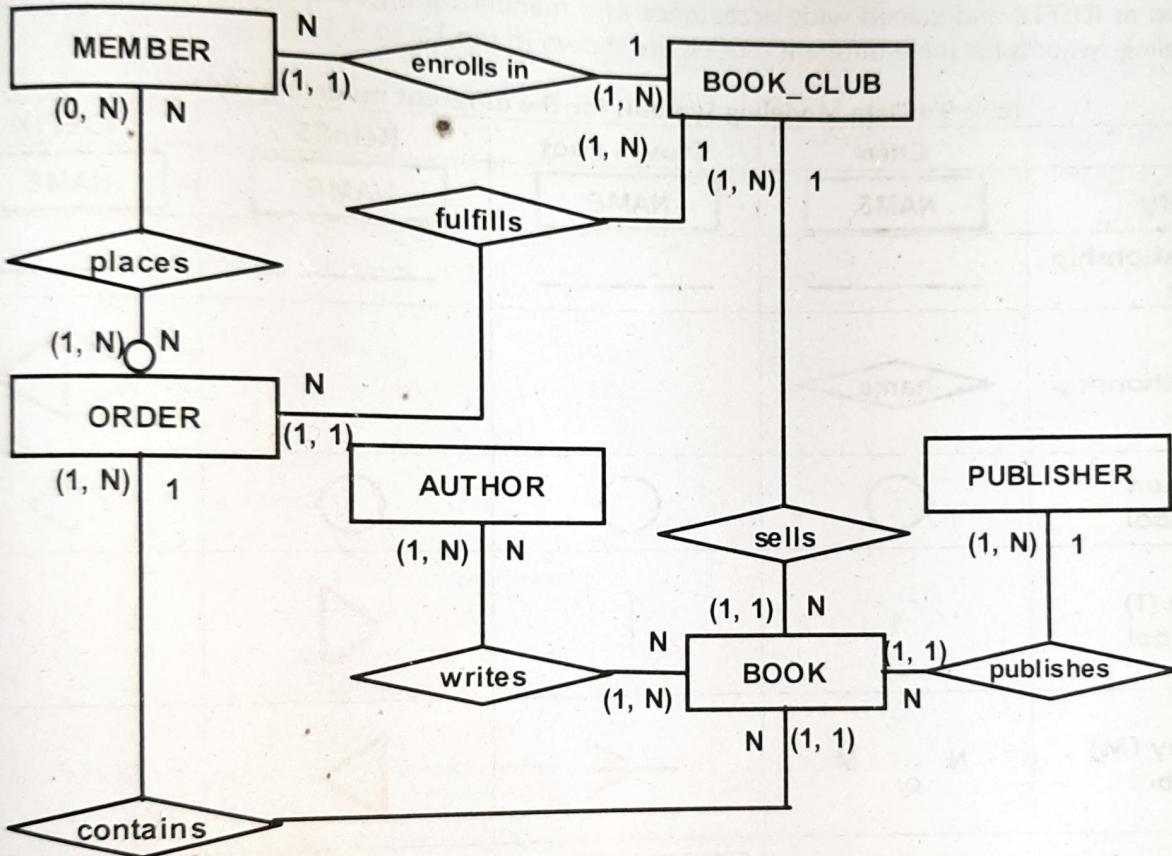


Figure 9.17 E-R Diagram of the Book Club



E-R MODELING SYMBOLS

We have been using the Chen's E-R modeling symbols so far in this chapter as they are the most popular. However, there are some other styles also that are used in many cases. So it is a good idea to be familiar with them. In this section we will make ourselves familiar with the other E-R modeling styles namely Crow's Foot, Rein85 and IDEF1X. These different styles are based on the same modeling concepts, but are used because they fit computer-based modeling tools more easily than the original Chen model.

The **Crow's Foot** model was developed by C.W. Bachman and was made popular by the Knowledgeware modeling tool. The crow's foot model combines the connectivity and cardinality information in a single symbol. But detailed cardinality cannot be represented in this model.

D. Reiner developed the **Rein85** model in 1985. The Rein85 methodology does not recognize cardinalities explicitly but rely on connectivities to lead to logical cardinality conclusions.

IDEF1X is the result of the ICAM (integrated computer-aided manufacturing) studies. The ICAM studies created a lot of graphical methods for defining functions, data structures and the dynamics of the manufacturing process. These methods were integrated into IDEF (ICAM DEFinition) by Hughes Aircraft Company and named as IDEF1. The extended version of IDEF1 is known as IDEF1X and gained wide acceptance as a manufacturing data-modeling tool. The data modeling symbols for these different models are shown in the Table 9.4.

Table 9.4 Data Modeling Symbols for the different modeling styles

	Chen	Crow's Foot	Rein85	IDEF1X
Entity	NAME	NAME	NAME	NAME
Relationship Line	—	—	—	—
Relationship	◇ name		.	◇
Option Symbol	○	○	○	◇
One (1) Symbol	1		▽	
Many (M) Symbol	N or M	↗	△	
Composite Entity	NAME	NAME	NAME	NAME
Weak Entity	NAME		NAME	



SUMMARY

In this chapter we say what is E-R modeling, what are entities, the different types of entities, what are relationships and the different properties associated with relationships. The Entity-Relationship (E-R) model is a high-level conceptual data model developed by Chen in 1976 to facilitate database design. We also saw what an E-R diagram is, how to construct an E-R diagram and so on. E-R diagram (ERD) or data schema map or data map is another means of capturing the data and their organization. The ERD captures the entities, their attributes and the relationships between the entities. The ERD pictorially represents the interrelationships of the