UNIT 2 RELATIONAL AND E-R MODELS

Structure **Page Nos.**

2.0 INTRODUCTION

In the first unit of this block, you have been provided with the details of the Database Management System, its advantages, structure, etc. This unit is an attempt to provide you information about relational and E-R models. The relational model is a widely used model for DBMS implementation. Most of the commercial DBMS products available in the industry are relational at core. In this unit we will discuss the terminology, operators and operations used in relational model.

The second model discussed in this unit is the E-R model, which primarily is a semantic model and is very useful in creating raw database design that can be further normalised. We discuss DBMS E-R diagrams, and their conversion to tables in this unit.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- describe relational model and its advantages;
- perform basic operations using relational algebra;
- draw an E-R diagram for a given problem;
- convert an E-R diagram to a relational database and vice versa.

2.2 THE RELATIONAL MODEL

A model in database system basically defines the structure or organisation of data and a set of operations on that data. Relational model is a simple model in which database is represented as a collection of "Relations", where each relation is represented by a two dimensional table. Thus, because of its simplicity it is most commonly used. The following table represents a simple relation:

PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

Figure 1: A Sample Person Relation

Following are some of the advantages of relational model:

- **Ease of use**

  The simple tabular representation of database helps the user define and query the database conveniently. For example, you can easily find out the age of the person whose first name is "Vibhu".

- **Flexibility**

  Since the database is a collection of tables, new data can be added and deleted easily. Also, manipulation of data from various tables can be done easily using various basic operations. For example, we can add a telephone number field in the table at *Figure 1*.

- **Accuracy**

  In relational databases the relational algebraic operations are used to manipulate database. These are mathematical operations and ensure accuracy (and less of ambiguity) as compared to other models. These operations are discussed in more detail in Section 2.4.

2.2.1 Domains, Attributes Tuple and Relation

Before we discuss the relational model in more detail, let us first define some very basic terms used in this model.

Tuple

Each row in a table represents a record and is called a tuple. A table containing 'n' attributes in a record is called n-tuple.

Attribute

The name of each column in a table is used to interpret its meaning and is called an attribute. Each table is called a relation.

For example, *Figure 2* represents a relation PERSON. The columns PERSON_ID, NAME, AGE and ADDRESS are the attributes of PERSON and each row in the table represents a separate tuple (record).

Relation Name: PERSON

PERSON_ID	NAME	AGE	ADDRESS	TELEPHONE
1	Sanjay Prasad	35	b-4,Modi Nagar	011-25347527
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.	023-12245678
3	Vibhu Datt	36	c-2, New Delhi	033-1601138

Figure 2: An extended PERSON relation

Domain

A domain is a set of permissible values that can be given to an attribute. So every attribute in a table has a specific domain. Values to these attributes cannot be assigned outside their domains.

In the example above if domain of PERSON_ID is a set of integer values from 1 to 1000 than a value outside this range will not be valid. Some other common domains may be age between 1 and 150. The domain can be defined by assigning a type or a format or a range to an attribute. For example, a domain for a number 501 to 999 can be specified by having a 3-digit number format having a range of values between 501 and 999. However, please note the domains can also be non-contiguous. For example, the enrolment number of IGNOU has the last digit as the check digit, thus the nine-digit enrolment numbers are non-continuous.

Relation

A relation consists of:

- Relational Schema
- Relation instance

Relational Schema:

A relational schema specifies the relation's name, its attributes and the domain of each attribute. If R is the name of a relation and A_1, $A_2 \ldots A_n$ is a list of attributes representing R then $R(A_1, A_2 \ldots A_n)$ is called a relational schema. Each attribute in this relational schema takes a value from some specific domain called Domain (Ai).

For example, the relational schema for relation PERSON as in *Figure 1* will be:

PERSON(PERSON_ID:integer, NAME: string, AGE:integer, ADDRESS:string)

Total number of attributes in a relation denotes the degree of a relation. Since the PERSON relation contains four attributes, so this relation is of degree 4.

Relation Instance or Relation State:

A relation instance denoted as **r** is a collection of tuples for a given relational schema at a specific point of time.

A relation state **r** of the relation schema R (A_1,A_2,…….A_N), also denoted by r(R) is a set of n-tuples

$r = \{t_1, t_2, \ldots\ldots\ldots\ldots t_m\}$
Where each n-tuple is an ordered list of n values
$t = \langle v_1, v_2, \ldots\ldots\ldots, v_n \rangle$
where each v_i belongs to domain (A_i) or contains null values.

The relation schema is also called *'intension'* and relation state is also called *'extension'*.

Let us elaborate the definitions above with the help of examples:

Example 1:

RELATION SCHEMA For STUDENT:

STUDENT (RollNo: string, name: string, login: string, age: integer)

RELATION INSTANCE

STUDENT

	ROLLNO	NAME	LOGIN	AGE
t_1	3467	Shikha	Noorie_jan@yahoo	20
t_2	4677	Kanu	Golgin_atat@yahoo	20

Where t_1 = (3467, shikha, Noorie-jan@yahoo.com , 20) for this relation instance, m = 2 and n = 4.

Example 2:

RELATIONAL SCHEMA For PERSON:

PERSON (PERSON_ID: integer, NAME: string, AGE: integer, ADDRESS: string)

RELATION INSTANCE
In this instance, m = 3 and n = 4

PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

Thus current relation state reflects only the valid tuples that represent a particular state of the real world. However, Null values can be assigned for the cases where the values are unknown or missing.

Ordering of tuples

In a relation, tuples are not inserted in any specific order. **Ordering of tuples is not defined as a part of a relation definition**. However, records may be organised later according to some attribute value in the storage systems. For example, records in PERSON table may be organised according to PERSON_ID. Such data or organisation depends on the requirement of the underlying database application. However, for the purpose of display we may get them displayed in the sorted order of age. The following table is sorted by age. It is also worth mentioning here that relational model **does not allow duplicate tuples**.

PERSON

PERSON_ID	NAME	AGE	ADDRESS
2	Sharad Gupta	33	Pocket 2, Mayur Vihar.
1	Sanjay Prasad	35	b-4,Modi Nagar
3	Vibhu Datt	36	c-2, New Delhi

2.2.2 Super Keys, Candidate Keys and Primary Keys for the Relations

As discussed in the previous section ordering of relations does not matter and all tuples in a relation are unique. However, can we uniquely identify a tuple in a relation? Let us discuss the concepts of keys that are primarily used for the purpose as above.

What Are Super Keys?

A **super key** is an attribute or set of attributes used to identify the records uniquely in a relation.

For Example, in the Relation PERSON described earlier PERSON_ID is a super key since PERSON_ID is unique for each person. Similarly (PERSON_ID, AGE) and (PERSON_ID, NAME) are also super keys of the relation PERSON since their combination is also unique for each record.

Candidate keys:

Super keys of a relation can contain extra attributes. Candidate keys are minimal super key, i.e. such a key contains no extraneous attribute. An attribute is called extraneous if even after removing it from the key, makes the remaining attributes still has the properties of a key.

The following properties must be satisfied by the candidate keys:

- A candidate key must be **unique**.
- A candidate key's value must **exist.** It cannot be null. (This is also called entity integrity rule)
- A candidate key is a minimal set of attributes.
- The value of a candidate key must be **stable.** Its value cannot change outside the control of the system.

A relation can have more than one candidate keys and one of them can be chosen as a **primary key.**

For example, in the relation PERSON the two possible candidate keys are PERSON-ID and NAME (assuming unique names in the table). PERSON-ID may be chosen as the primary key.

2.3 RELATIONAL CONSTRAINTS

There are three types of constraints on relational database that include:

- DOMAIN CONSTRAINT
- PRIMARY KEY CONSTRAINT
- INTEGRITY CONSTRAINT

2.3.1 Domain Constraint

It specifies that each attribute in a relation must contain an atomic value only from the corresponding domains. The data types associated with commercial RDBMS domains include:

1) Standard numeric data types for integer (such as short- integer, integer, long integer)
2) Real numbers (float, double precision floats)
3) Characters
4) Fixed length strings and variable length strings.

Thus, domain constraint specifies the condition that we want to put on each instance of the relation. So the values that appear in each column must be drawn from the domain associated with that column.
For example, consider the relation:

STUDENT

ROLLNO	NAME	LOGIN	AGE
4677	Kanu	Golgin_atat@yahoo.com	20
3677	Shikha	Noorie_jan@yahoo.com	20

In the relation above, AGE of the relation STUDENT always belongs to the integer domain within a specified range (if any), and not to strings or any other domain. Within a domain non-atomic values should be avoided. This sometimes cannot be checked by domain constraints. For example, a database which has area code and phone numbers as two different fields will take phone numbers as-

Area code	Phone
11	29534466

A non-atomic value in this case for a phone can be 1129534466, however, this value can be accepted by the Phone field.

2.3.2 Key Constraint

This constraint states that the key attribute value in each tuple must be unique, i.e., no two tuples contain the same value for the key attribute. This is because the value of the primary key is used to identify the tuples in the relation.

Example 3: If A is the key attribute in the following relation R than A1, A2 and A3 must be unique.

R

A	B
A1	B1
A3	B2
A2	B2

Example 4: In relation PERSON, PERSON_ID is primary key so PERSON_ID cannot be given as the same for two persons.

2.3.3 Integrity Constraint

There are two types of integrity constraints:

- Entity Integrity Constraint
- Referential Integrity Constraint

Entity Integrity Constraint:

It states that **no primary key value can be null**. This is because the primary key is used to identify individual tuple in the relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes. This constraint is specified on one individual relation.

Example 5: Let R be the Table

A#	B	C
Null	B1	C1
A2	B2	C2
Null	B3	C3
A4	B4	C3
A5	B1	C5

Note:

1) '#' identifies the Primary key of a relation.

In the relation R above, the primary key has null values in the tuples t_1 & t_3. NULL value in primary key is not permitted, thus, relation instance is an invalid instance.

Referential integrity constraint

It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation. This constraint is specified on two relations (not necessarily distinct). It uses a concept of foreign key and has been dealt with in more detail in the next unit.

Example 6:

R

A#	B	C^
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C3
A5	B1	C5

S

E	C#
E1	C1
E2	C3
E3	C5
E2	C2

Note:

1) '#' identifies the Primary key of a relation.
2) '^' identifies the Foreign key of a relation.

In the example above, the value of C^ in every R tuple is matching with the value of C# in some S tuple. If a tuple having values (A6, B2, C4) is added then it is invalid since referenced relation S doesn't include C4. Thus, it will be a violation of referential integrity constraint.

2.3.4 Update Operations and Dealing with Constraint Violations

There are three basic operations to be performed on relations:

- Insertion
- Deletion
- Update

The INSERT Operation:

The insert operation allows us to insert a new tuple in a relation. When we try to insert a new record, then any of the following four types of constraints can be violated:

- Domain constraint: If the value given to an attribute lies outside the domain of that attribute.
- Key constraint: If the value of the key attribute in new *tuple t* is the same as in the existing tuple in *relation R.*
- Entity Integrity constraint: If the primary key attribute value of new *tuple t* is given as *null*.
- Referential Integrity constraint: If the value of the foreign key in t refers to a tuple that doesn't appear in the referenced relation.

Dealing with constraints violation during insertion:

If the *insertion* violates one or more constraints, then two options are available:

- Default option: - *Insertion can be rejected and the reason of rejection can also be explained to the user by* DBMS.

- Ask the user to correct the data, resubmit, also give *the reason for rejecting the insertion.*

Example 7:

Consider the Relation PERSON of Example 2:

PERSON

PERSON_ID	NAME	AGE	ADDRESS
1	Sanjay Prasad	35	b-4,Modi Nagar
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.
3	Vibhu Datt	36	c-2, New Delhi

(1) Insert<1, 'Vipin', 20, 'Mayur Vihar'> into PERSON
Violated constraint: - Key constraint
Reason: - Primary key 1 already exists in PERSON.
Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

(2) Insert<'null', 'Anurag', 25, 'Patparganj'> into PERSON
Violated constraint: - Entity Integrity constraint
Reason: - Primary key is 'null'.
Dealing: - DBMS could ask the user to provide valid PERSON_ID value and accept the insertion if valid PERSON_ID value is provided.

(3) Insert<'abc', 'Suman', 25, 'IP college'> into PERSON
Violated constraint: - Domain constraint
Reason: - value of PERSON_ID is given a string which is not valid.

(4) Insert <10, 'Anu', 25, 'Patpatganj'> into PERSON
Violated constraint: - None

Note: In all first 3 cases of constraint violations above DBMS could reject the insertion.

The Deletion Operation:

Using the delete operation some existing records can be deleted from a relation. To delete some specific records from the database a condition is also specified based on which records can be selected for deletion.

Constraints that can be violated during deletion

Only one type of constraint can be violated during deletion, it is referential integrity constraint. It can occur when you want to delete a record in the table where it is referenced by the foreign key of another table. Please go through the example 8 very carefully.

Dealing with Constraints Violation

If the *deletion* violates referential integrity constraint, then three options are available:

- Default option: - *Reject the deletion.* It is the job of the DBMS to explain to the user why the deletion was rejected.
- *Attempt to cascade (or propagate) the deletion* by deleting tuples that reference the tuple that is being deleted.
- Change the value of *referencing attribute* that causes the violation.

Example 8:

Let R:

A#	B	C^
A1	B1	C1
A2	B3	C3
A3	B4	C3
A4	B1	C5

Q

C#	D
C1	D1
C3	D2
C5	D3

Note:

1) '#' identifies the Primary key of a relation.
2) '^' identifies the Foreign key of a relation.

(1) Delete a tuple with C# = 'C1' in Q.
Violated constraint: - Referential Integrity constraint

Reason: - Tuples in relation A refer to tuple in Q.
Dealing: - Options available are

1) Reject the deletion.
2) DBMS may automatically delete all tuples from relation *Q* and *S* with C # = 'C1'. This is called cascade detection.
3) The third option would result in putting NULL value in R where C1 exist, which is the first tuple R in the attribute C.

The Update Operations:

Update operations are used for modifying database values. The constraint violations faced by this operation are logically the same as the problem faced by Insertion and Deletion Operations. Therefore, we will not discuss this operation in greater detail here.

2.4 RELATIONAL ALGEBRA

Relational Algebra is a set of basic operations used to manipulate the data in relational model. These operations enable the user to specify basic retrieval request. The result of retrieval is a new relation, formed from one or more relations. **These operations can be classified in two categories:**

- Basic Set Operations
  1) UNION
  2) INTERSECTION
  3) SET DIFFERENCE
  4) CARTESIAN PRODUCT

- Relational Operations
  1) SELECT
  2) PROJECT
  3) JOIN
  4) DIVISION

2.4.1 Basic Set Operation

These are the binary operations; i.e., each is applied to two sets or relations. These two relations should be union compatible except in case of *Cartesian Product*. Two relations R (A_1, A_2,….., A_n) and S (B_1, B_2,……,B_n) are said to be union compatible if they have the ***same degree n*** and domains of the corresponding attributes are also the same i.e.***Domain (Ai) = Domain (Bi) for 1<=i<=n***.

UNION

If R1 and R2 are two union compatible relations then R3 = R1 ∪ R2 is the relation containing tuples that are either in R1 or in R2 or in both.

In other words, R3 will have tuples such that R3 = {t | R1 ϶ t ∨ R2 ϶ t}.

Example 9:

R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R3 = R1 ∪ R2 is

Q

A	B
A1	B1
A2	B2
A3	B3
A4	B4
A7	B7

Note: 1) Union is a commutative operation, i.e,

R ∪ S = S ∪ R.

2) Union is an associative operation, i.e.

R ∪ (S ∪ T) = (R ∪ S) ∪ T.

Intersection

If R1 and R2 are two union compatible functions or relations, then the result of R3 = R1 ∩R2 is the relation that includes all tuples that are in both the relations In other words, R3will have tuples such that R3 = {t | R1 ϶ t ∧ R2 ϶ t}.

Example 10:

R1

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R3 = R1 ∩ R2 is

A	B
A1	B1
A2	B2
A4	B4

Note: 1) Intersection is a commutative operation, i.e.,

$R1 \cap R2 = R2 \cap R1$.

2) Intersection is an associative operation, i.e.,

$R1 \cap (R2 \cap R3) = (R1 \cap R2) \cap R3$

Set Difference

If R1 and R2 are two union compatible relations or relations then result of R3 =R1– R2 is the relation that includes only those tuples that are in R1 but not in R2.

In other words, R3 will have tuples such that $R3 = \{t \mid R1 \ni t \wedge t \notin R2\}$.

Example 11:

R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R1-R2 =

A	B
A3	B3

R2-R1=

A	B
A7	B7

Note: -1) Difference operation is not commutative, i.e.,

$$\mathbf{R1 - R2 \neq R2 - R1}$$

2) Difference operation is not associative, i. e.,

$$\mathbf{R1 - (R2 - R3) \neq (R1 - R2) - R3}$$

2.4.2 Cartesian Product

If R1 and R2 are two functions or relations, then the result of $R3 = R1 \times R2$ is the combination of tuples that are in R1 and R2. The product is commutative and associative.

Degree (R3) =Degree of (R1) + Degree (R2).
In other words, R3 will have tuples such that $R3 = \{t_1 \| t_2 \mid R1 \ni t_1 \wedge R2 \ni t_2\}$.

Example 12:

R1

C
C1
C2

R2

A	B
A1	B1
A2	B2
A3	B3
A4	B4

$R3 = R1 \times R2$ is

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C1
A2	B2	C2
A3	B3	C1
A3	B3	C2
A4	B4	C1
A4	B4	C2

2.4.3 Relational Operations

Let us now discuss the relational operations:

SELECT

The select operation is used to select some specific records from the databse based on some criteria. This is a unary operation mathematically denoted as σ.

Syntax:

σ **<Selection condition> (Relation)**

The Boolean expression is specified in <Select condition> is made of a number of clauses of the form:

<attribute name><comparison operator><constant value> or

<attribute name><comparison operator><attribute name>

Comparison operators in the set $\{\leq, \geq, \neq, =, <, <\}$ apply to the attributes whose domains **are *ordered value* like integer.**

Example 13:

Consider the relation PERSON. If you want to display details of persons having age less than or equal to 30 than the select operation will be used as follows:

σ **AGE <=30 (PERSON)**

The resultant relation will be as follows:

PERSON_ID	NAME	AGE	ADDRESS
2	Sharad Gupta	30	Pocket 2, Mayur Vihar.

Note:

1) Select operation is commutative; i.e.,
σ <condition1> (σ <condition2> (R)) = σ <condition2> (σ <condition1> (R))
Hence, Sequence of select can be applied in any order

2) More than one condition can be applied using Boolean operators AND & OR etc.

The PROJECT operation

The project operation is used to select the records with specified attributes while discarding the others based on some specific criteria. This is denoted as Π**.**

Π **List of attribute for project (Relation)**

Example 14:

Consider the relation PERSON. If you want to display only the names of persons then the project operation will be used as follows:

Π **Name (PERSON)**

The resultant relation will be as follows:

NAME
Sanjay Prasad
Sharad Gupta
Vibhu Datt

Note: -

1) $\Pi_{<List1>}(\Pi_{<list2>}(R)) = \Pi_{<list1>}(R)$

As long as<list2> contains attributes in <list1>.

The JOIN operation

The JOIN operation is applied on two relations. When we want to select related tuples from two given relation join is used. This is denoted as ⋈. The join operation requires that both the joined relations must have at least one domain compatible attributes.

Syntax:

$R1 \bowtie_{<join\ condition>} R2$ is used to combine related tuples from two relations R1 and R2 into a single tuple.
<join condition> is of the form:
<condition>AND<condition>AND……..AND<condition>.

- Degree of Relation:

Degree ($R1 \bowtie_{<join\ condition>} R2$) <= Degree (R1) + Degree (R2).

- Three types of joins are there:

a) **Theta join**

When each condition is of the form A θ B, A is an attribute of R1 and B is an attribute of R2 and have the same domain, and θ is one of the comparison operators **{**$\leq, \geq, \neq$**, =, <, <}.**

b) **Equijoin**

When each condition appears with equality condition (=) only.

c) **Natural join (denoted by R*S)**

When two join attributes have the same name in both relations. (That attribute is called Join attribute), only one of the two attributes is retained in the join relation. The

join condition in such a case is = for the join attribute. The condition is not shown in the natural join.

Let us show these operations with the help of the following example.

Example 15:

Consider the following relations**:**

STUDENT

ROLLNO	NAME	ADDRESS	COURSE_ID
100	Kanupriya	234, Saraswati Vihar.	CS1
101	Rajni Bala	120, Vasant Kunj	CS2
102	Arpita Gupta	75, SRM Apartments.	CS4

COURSE

COURSE ID	COURSE_NAME	DURATION
CS1	MCA	3yrs
CS2	BCA	3yrs
CS3	M.Sc.	2yrs
CS4	B.Sc.	3yrs
CS5	MBA	2yrs

If we want to display name of all the students along with their course details then natural join is used in the following way:

STUDENT ⋈ COURSE

Resultant relation will be as follows:

STUDENT

ROLLNO	NAME	ADDRESS	COURSE_ID	COURSE_NAME	DURATION
100	Kanupriya	234, Saraswati Vihar.	CS1	MCA	3yrs
101	Rajni Bala	120, Vasant Kunj	CS2	BCA	3yrs
102	Arpita Gupta	75, SRM Apartments.	CS4	B.Sc.	3yrs

There are other types of joins like outer joins. You must refer to further reading for more details on those operations. They are also explained in Block 2 Unit 1.

The DIVISION operation:

To perform the division operation R1 ÷ R2, R2 should be a proper subset of R1. In the following example R1 contains attributes A and B and R2 contains only attribute B so R2 is a proper subset of R1. If we perform R1 ÷ R2 than the resultant relation will contain those values of A from R1 that are related to all values of B present in R2.

Example 16:

Let R1

A	B
A1	B1
A1	B2
A2	B1
A3	B1
A4	B2
A5	B1
A3	B2

If R2 R3= R1 ÷ R2

B
B1
B2

A
A1
A3

(a)

If R2 R3= R1 ÷ R2

B
B1

A
A1
A2
A3
A5

(b)

If R2 R3= R1 ÷ R2

A

B
B1
B2
B3

(c)

If R2 R3= R1 ÷ R2

B

A
A1
A2
A3
A4
A5

(d)

Figure 3: The Division Operation

Note:

Degree of relation: Degree (R ÷ S)=Degree of R – Degree of S.

Check Your Progress 1

1) A database system is fully relational if it supports _____ and _____.

2) A Relation resembles a _____ a tuple resembles a _____ and an attribute resembles a _____.

3) A candidate key which is not a primary key is known as a _____key.

4) Primitive operations are union, difference, product, selection and projection. The definition of A intersects B can be_____.

5) Which one is not a traditional set operator defined on relational algebra?

   a. Union
   b. Intersection
   c. Difference
   d. Join

6) Consider the tables **S**uppliers, **P**arts, pro**J**ect and **SPJ** relation instances in the relations below. (Underline represents a key attribute. The SPJ relation have three Foreign keys: SNO, PNO and JNO.)

S

SNO	SNAME	CITY
S1	Smita	Delhi
S2	Jim	Pune
S3	Ballav	Pune
S4	Sita	Delhi
S5	Anand	Agra

P

PNO	PNAME	COLOUR	CITY
P1	Nut	Red	Delhi
P2	Bolt	Blue	Pune
P3	Screw	White	Bombay
P4	Screw	Blue	Delhi
P5	Camera	Brown	Pune
P6	Cog	Grey	Delhi

J

JNO	JNAME	CITY
J1	Sorter	Pune
J2	Display	Bombay
J3	OCR	Agra
J4	Console	Agra
J5	RAID	Delhi
J6	EDS	Udaipur
J7	Tape	Delhi

SPJ

SNO	PNO	JNO	QUANTITY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J2	400
S2	P2	J7	200
S2	P3	J3	500
S3	P3	J5	400
S3	P4	J3	500
S3	P5	J3	600
S3	P6	J4	800
S4	P6	J2	900
S4	P6	J1	100
S4	P5	J7	200
S5	P5	J5	300
S5	P4	J6	400

Using the sample data values in the relations above, tell the effect of each of the following operation:

a) UPDATE Project J7 in J setting CITY to Nagpur.
b) UPDATE part P5 in P, setting PNO to P4.
c) UPDATE supplier S5 in S, setting SNO to S8, if the relevant update rule is RESTRICT.
d) DELETE supplier S3 in S, if the relevant rule is CASCADE.
e) DELETE part P2 in P, if the relevant delete rule is RESTRICT.
f) DELETE project J4 in J, if the relevant delete rule is CASCADE.
g) UPDATE shipment S1-P1-J1 in SPJ, setting SNO to S2. (shipment S1-P1-J1 means that in SPJ table the value for attributes SNO, PNO and JNO are S1, P1 and J1 respectively)
h) UPDATE shipment S5-P5-J5 in SPJ, setting JNO to J7.
i) UPDATE shipment S5-P5-J5 in SPJ, setting JNO to J8
j) INSERT shipment S5-P6-J7 in SPJ.
k) INSERT shipment S4-P7-J6 in SPJ
l) INSERT shipment S1-P2-jjj (where jjj stands for a default project number).

……

……

……

7) Find the name of projects in the relations above, to which supplier S1 has supplied.

……

……

……

……

2.5 ENTITY RELATIONSHIP (ER) MODEL

Let us first look into some of the main features of ER model.

- Entity relationship model is a high-level conceptual data model.
- It allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships.
- It is widely used to develop an initial design of a database.
- It provides a set of useful concepts that make it convenient for a developer to move from a basic set of information to a detailed and precise description of information that can be easily implemented in a database system.
- It describes data as a collection of entities, relationships and attributes.

Let us explain it with the help of an example application.

An Example Database Application

We will describe here an example database application containing COLLEGE database and use it in illustrating various E-R modeling concepts.
College database keeps track of Students, faculty, Departments and Courses organised by various departments. Following is the description of COLLEGE database:

College contains various departments like Department of English, Department of Hindi, Department of Computer Science etc. Each department is assigned a unique id and name. Some faculty members are also appointed to each department and one of them works as head of the department.

There are various courses conducted by each department. Each course is assigned a unique id, name and duration.

- Faculty information contains name, address, department, basic salary etc. A faculty member is assigned to only one department but can teach various courses of other department also.
- Student's information contain Roll no (unique), Name, Address etc. A student can opt only for one course.
- Parent (gurdian) information is also kept along with each student. We keep each guardian's name, age, sex and address.

2.5.1 Entities

Let us first be aware of the question:

What are entities?

- An entity is an object of concern used to represent the things in the real world, e.g., car, table, book, etc.
- An entity need not be a physical entity, it can also represent a concept in real world, e.g., project, loan, etc.
- It represents a class of things, not any one instance, e.g., 'STUDENT' entity has instances of 'Ramesh' and 'Mohan'.

Entity Set or Entity Type:

A collection of a similar kind of entities is called an **Entity Set or entity type.**

Example 16:

For the COLLEGE database described earlier objects of concern are Students, Faculty, Course and departments. The collections of all the students entities form a entity set STUDENT. Similarly collection of all the courses form an entity set COURSE.

Entity sets need not be disjoint. For example – an entity may be part of the entity set STUDENT, the entity set FACULTY, and the entity set PERSON.

Entity identifier key attributes

An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.

Strong entity set: The entity types containing a key attribute are called strong entity types or regular entity types.

- EXAMPLE: The Student entity has a key attribute RollNo which uniquely identifies it, hence is a strong entity.

2.5.2 Attributes

Let us first answer the question:

What is an attribute?

An attribute is a property used to describe the specific feature of the entity. So to describe an entity entirely, a set of attributes is used.

For example, a student entity may be described by the student's name, age, address, course, etc.

An entity will have a value for each of its attributes. For example for a particular student the following values can be assigned:

RollNo: 1234
Name: Supriya
Age: 18
Address: B-4, Mayapuri, Delhi.
Course: B.Sc. (H)
••

Domains:

- Each simple attribute of an entity type contains a possible set of values that can be attached to it. This is called the domain of an attribute. An attribute cannot contain a value outside this domain.
- EXAMPLE- for PERSON entity PERSON_ID has a specific domain, integer values say from 1 to 100.

Types of attributes

Attributes attached to an entity can be of various types.

Simple

The attribute that cannot be further divided into smaller parts and represents the basic meaning is called a simple attribute. For example: The 'First name', 'Last name', age attributes of a person entity represent a simple attribute.

Composite

Attributes that can be further divided into smaller units and each individual unit contains a specific meaning.

For example:-The NAME attribute of an employee entity can be sub-divided into First name, Last name and Middle name.

Single valued

Attributes having a single value for a particular entity. For Example, Age is a single valued attribute of a student entity.

Multivalued

Attributes that have more than one values for a particular entity is called a multivalued attribute. Different entities may have different number of values for these kind of attributes. For multivalued attributes we must also specify the minimum and maximum number of vales that can be attached. For Example phone number for a person entity is a multivalued attribute.

Stored

Attributes that are directly stored in the data base.

For example, 'Birth date' attribute of a person.

Derived

Attributes that are not stored directly but can be derived from stored attributes are called derived attributes. For Example, The years of services of a 'person' entity can be determined from the current date and the date of joining of the person. Similarly, total salary of a 'person' can be calculated from 'basic salary' attribute of a 'person'.

2.5.3 Relationships

Let us first define the term relationships.

What Are Relationships?

A relationship can be defined as:

- a connection or set of associations, or
- a rule for communication among entities:

Example: In college the database, the association between student and course entity, i.e., "Student opts course" is an example of a relationship.

Relationship sets

A relationship set is a set of relationships of the same type.

For example, consider the relationship between two entity sets *student* and course. Collection of all the instances of relationship opts forms a relationship set called relationship type.

Degree

The degree of a relationship type is the number of participating entity types.

The relationship between two entities is called binary relationship. A relationship among three entities is called ternary relationship.

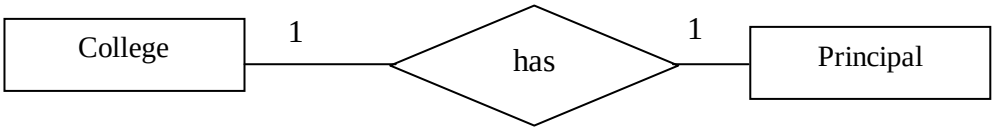
Similarly relationship among n entities is called n-ry relationship.

Relationship Cardinality

Cardinality specifies the number of instances of an entity associated with another entity participating in a relationship. Based on the cardinality binary relationship can be further classified into the following categories:

- **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

Example 17: Relationship between college and principal

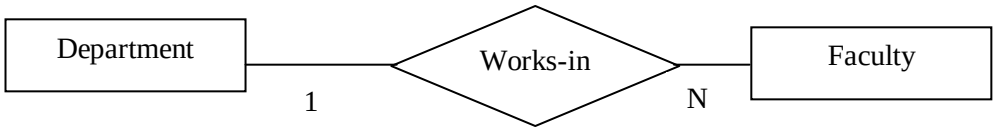


One college can have at the most one principal and one principal can be assigned to only one college.

Similarly we can define the relationship between university and Vice Chancellor.

- **One-to-many**: An entity in A is associated with any number of entities in B. An entity in B is associated with at the most one entity in A.

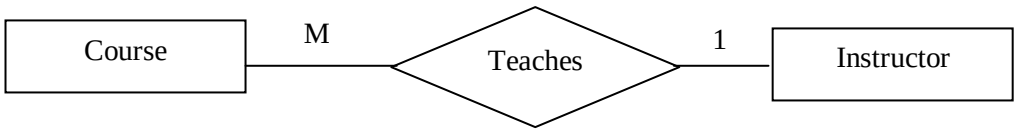
Example 18: Relationship between department and faculty.



One department can appoint any number of faculty members but a faculty member is assigned to only one department.

- **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.

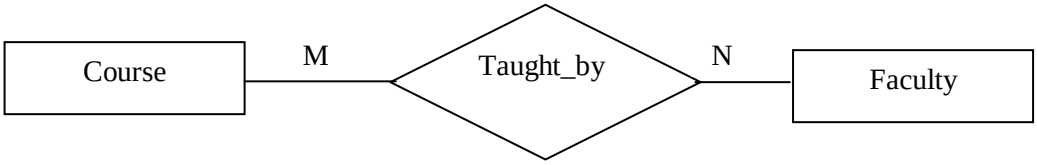
Example 19: Relationship between course and instructor. An instructor can teach various courses but a course can be taught only by one instructor. Please note this is an assumption.



Many-to-many: Entities in A and B are associated with any number of entities from each other.

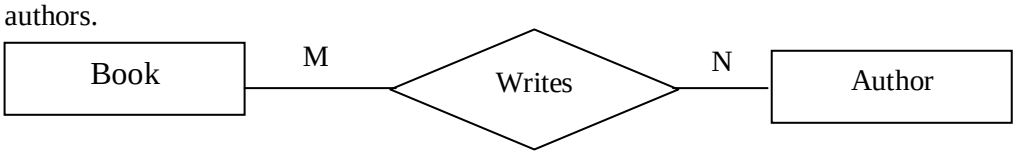
Example 20:

Taught_by Relationship between course and faculty.
One faculty member can be assigned to teach many courses and one course may be taught by many faculty members.



Relationship between book and author.

One author can write many books and one book can be written by more than one authors.



2.5.4 More about Entities and Relationships

Recursive relationships

When the same entity type participates more than once in a relationship type in different roles, the relationship types are called recursive relationships.

Participation constraints

The participation Constraints specify whether the existence of an entity depends on its being related to another entity via the relationship type. There are 2 types of participation constraints:

Total: When all the entities from an entity set participate in a relationship type, is called total participation, for example, the participation of the entity set student in the relationship set must ‘opts’ is said to be total because every student enrolled must opt for a course.

Partial: When it is not necessary for all the entities from an entity set to participate in a relationship type, it is called partial participation. For example, the participation of the entity set student in ‘represents’ is partial, since not every student in a class is a class representative.

WEAK ENTITY

Entity types that do not contain any key attribute, and hence cannot be identified independently, are called weak entity types. A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key attributes of another entity, which is called the identifying owner entity.

Generally a partial key is attached to a weak entity type that is used for unique identification of weak entities related to a particular owner entity type. The following restrictions must hold:

- The owner entity set and the weak entity set must participate in one to many relationship set. This relationship set is called the identifying relationship set of the weak entity set.
- The weak entity set must have total participation in the identifying relationship.

EXAMPLE: Consider the entity type dependent related to employee entity, which is used to keep track of the dependents of each employee. The attributes of dependents are: name, birth date, sex and relationship. Each employee entity is said to own the dependent entities that are related to it. However, please note that the 'dependent' entity does not exist of its own, it is dependent on the Employee entity. In other words we can say that in case an employee leaves the organisation all dependents related to him/her also leave along with. Thus, the 'dependent' entity has no significance without the entity 'employee'. Thus, it is a weak entity. The notation used for weak entities is explained in Section 2.6.

Extended E-R features:

Although, the basic features of E-R diagrams are sufficient to design many database situations. However, with more complex relations and advanced database applications, it is required to move to enhanced features of E-R models. The three such features are:

- Generalisation
- Specialisation, and
- Aggregation

In this unit, we have explained them with the help of an example. More detail on them are available in the further readings and MCS-043.

Example 21: A bank has an account entity set. The accounts of the bank can be of two types:

- Savings account
- Current account

The statement as above represents a specialisation/generalisation hierarchy. It can be shown as:

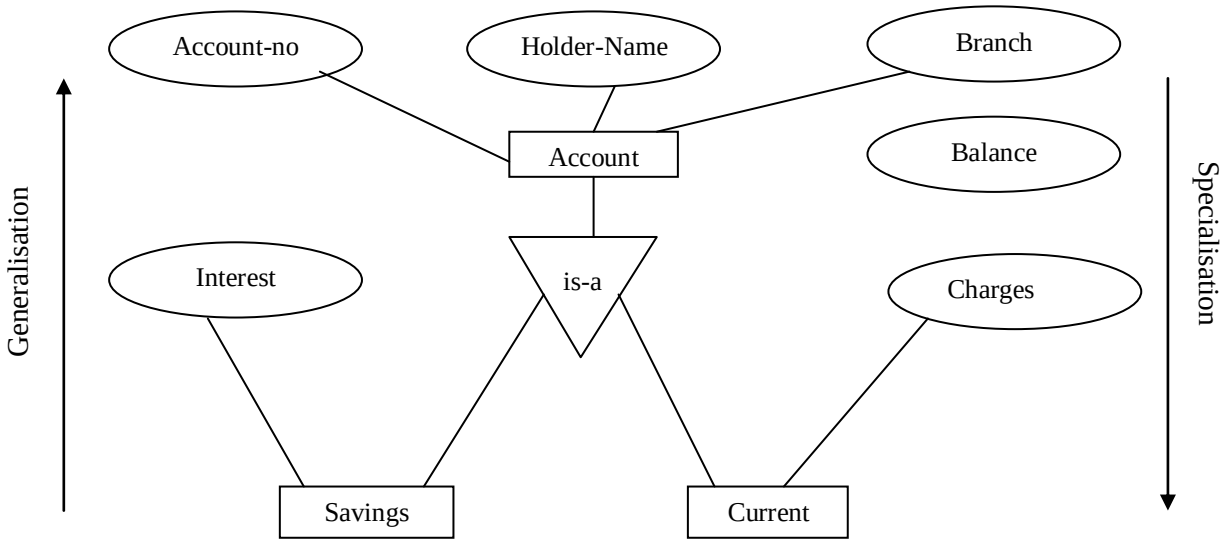


Figure 4: Generalisation and Specialisation hierarchy

But how are these diagrams converted to tables? This is shown in section 2.7.

Aggregation: One limitation of the E-R diagram is that they do not allow representation of relationships among relationships. In such a case the relationship along with its entities are promoted (aggregated to form an aggregate entity which can be used for expressing the required relationships). A detailed discussion on aggregation is beyond the scope of this unit you can refer to the further readings for more detail.

2.5.5 Defining Relationship For College Database

Using the concepts defined earlier, we have identified that strong entities in COLLEGE database are STUDENT, FACULTY, COURSE and DEPARTMENT. This database also has one weak entity called GUARDIAN. We can specify the following relationships:

1. **Head-of**, is a 1:1 relationship between FACULTY and DEPARTMENT. Participation of the entity FACULTY is partial since not all the faculty members participate in this relationship, while the participation from department side is total, since every department has one head.

2. **Works_in**, is a 1:N relationship between DEPARTMENT and FACULTY. Participation from both side is total.

3. **Opts,** is a 1:N relationship between COURSE and STUDENT. Participation from student side is total because we are assuming that each student enrolled opts for a course. But the participation from the course side is partial, since there can be courses that no student has opted for.

4. **Taught_by**, is a M: N relationship between FACULTY and COURSE, as a faculty can teach many courses and a course can be taught by many faculty members.

5. **Enrolled**, is a 1:N relationship between STUDENT and DEPARTMENT as a student is allowed to enroll for only one department at a time.

6. **Has**, is a 1:N relationship between STUDENT and GUARDIAN as a student can have more than one local guardian and one local guardian is assumed to be related to one student only. The weak entity Guardian has total participation in the relation "Has".

So now, let us make an E-R diagram for the college database in the next section.

2.6 E-R DIAGRAM

Let us now make the E-R diagram for the student database as per the description given in the previous section.

We can also express the overall logical structure of a database using ER model **graphically** with the help of an E-R diagram.

ER diagrams are composed of:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing relationship sets.

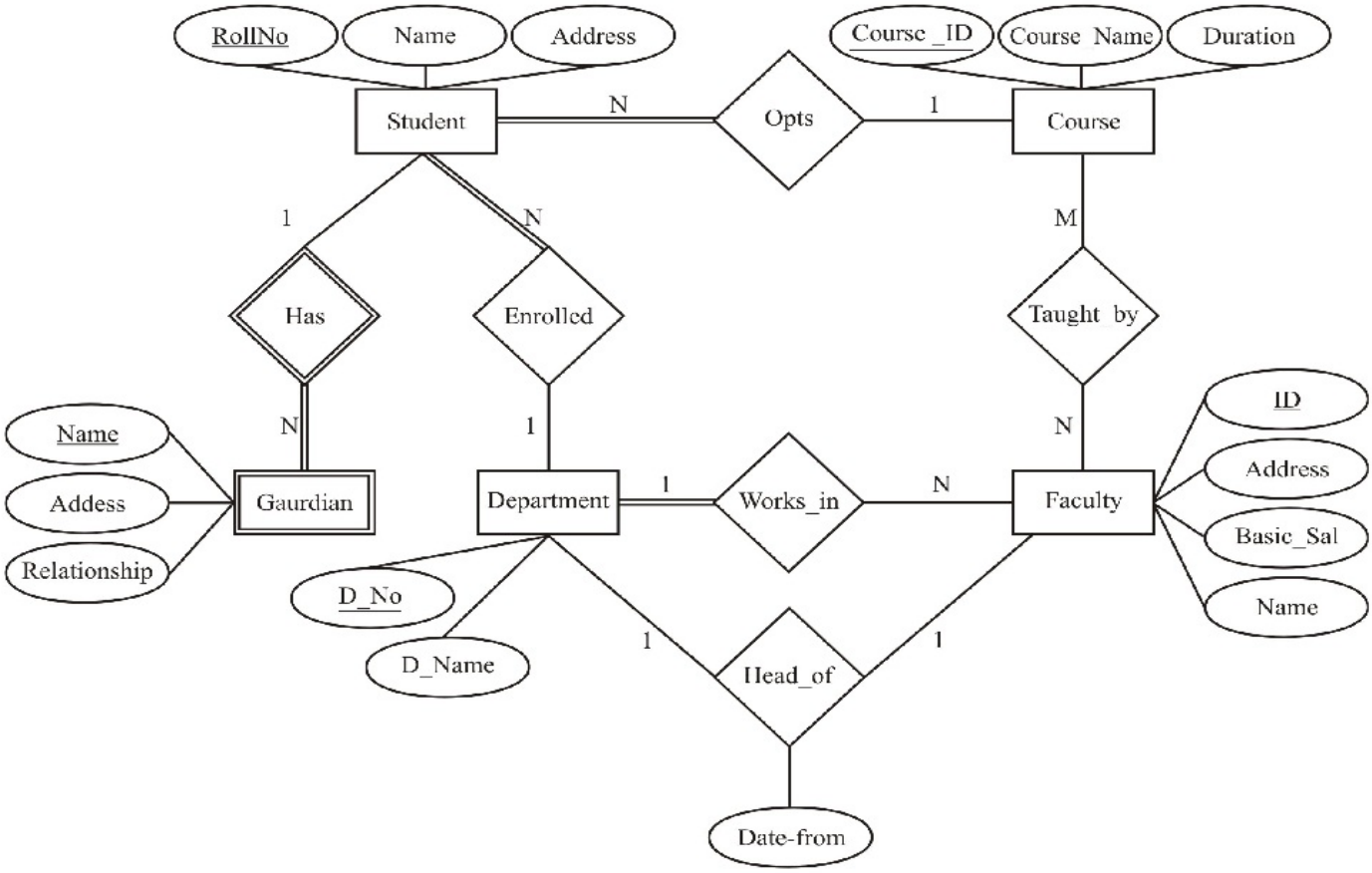


Figure 5: ER diagram of COLLEGE database

Please note that the relationship head-of has an attribute "Date-from". Let us define the symbols used in the ER diagram:

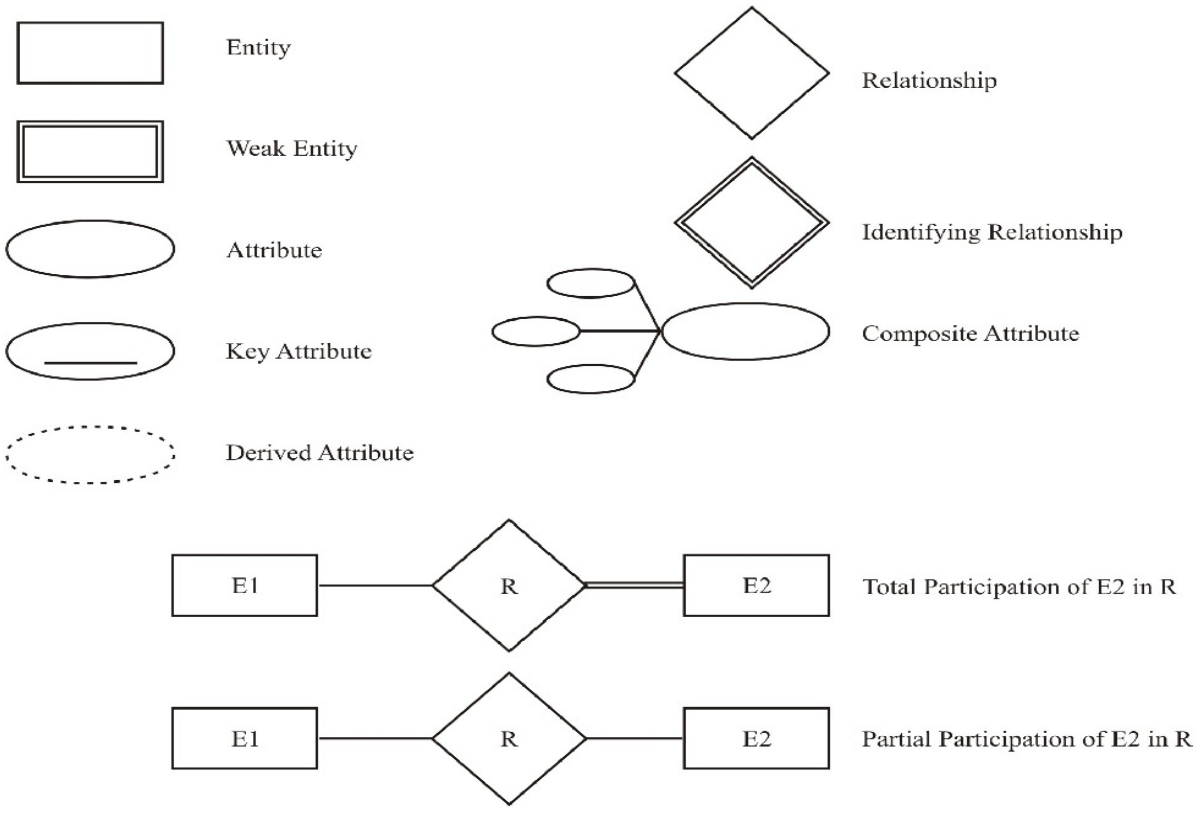


Figure 6: Symbols of E-R diagrams.

2.7 CONVERSION OF ER DIAGRAM TO RELATIONAL DATABASE

For every ER diagram we can construct a relational database which is a collection of tables. Following are the set of steps used for conversion of ER diagram to a relational database.

Conversion of entity sets:

I) For each strong entity type E in the ER diagram, we create a relation R containing all the simple attributes of E. The primary key of the relation R will be one of the key attributes of R.

For example, the STUDENT, FACULTY, COURSE and DEPARTMENT tables in *Figure 7.*

STUDENT

ROLLNO: Primary Key	NAME	ADDRESS

FACULTY

ID: Primary Key	NAME	ADDRESS	BASIC_SAL

COURSE

COURSE_ID: Primary Key	COURSE_NAME	DURATION

DEPARTMENT

D_NO: Primary Key	D_NAME

Figure 7: Conversion of Strong Entities

II) For each weak entity type W in the E R Diagram, we create another relation R that contains all simple attributes of W. If E is an owner entity of W then key attribute of E is also included in R. This key attribute of R is set as a foreign key attribute of R. Now the combination of primary key attribute of owner entity type and partial key of weak entity type will form the key of the weak entity type.

Figure 8 shows the weak entity GUARDIAN, where the key field of student entity RollNo has been added.

RollNo Name (Primary Key)		**Address**	**Relationship**

Figure 8: Conversion of weak entity to table.

Conversion of relationship sets:

Binary Relationships:

I) One-to-one relationship:

For each 1:1 relationship type R in the ER diagram involving two entities E1 and E2 we choose one of entities (say E1) preferably with total participation and add primary key attribute of another entityE2 as a foreign key attribute in the table of entity (E1). We will also include all the simple attributes of relationship type R in E1 if any.
For example, the DEPARTMENT relationship has been extended to include head-Id and attribute of the relationship. Please note we will keep information in this table of only current head and Date from which s/he is the head. (Refer to *Figure 9*).

There is one Head_of 1:1 relationship between FACULTY and DEPARTMENT. We choose DEPARTMENT entity having total participation and add primary key attribute ID of FACULTY entity as a foreign key in DEPARTMENT entity named as Head_ID. Now the DEPARTMENT table will be as follows:

DEPARTMENT

D_NO	**D_NAME**	**Head_ID**	**Date-from**

Figure 9: Converting 1:1 relationship

II) One-to-many relationship:

For each 1: n relationship type R involving two entities E1 and E2, we identify the entity type (say E1) at the n-side of the relationship type R and include primary key of the entity on the other side of the relation (say E2) as a foreign key attribute in the table of E1. We include all simple attributes (or simple components of a composite attributes of R (if any) in the table of E1).

For example, the works_in relationship between the DEPARTMENT and FACULTY. For this relationship choose the entity at N side, i.e., FACULTY and add primary key attribute of another entity DEPARTMENT, i.e., DNO as a foreign key attribute in FACULTY. Please refer to *Figure 10*.

FACULTY (CONTAINS WORKS_IN RELATIONSHIP)

ID	**NAME**	**ADDRESS**	**BASIC_SAL**	**DNO**

Figure 10: Converting 1:N relationship

III) Many-to-many relationship:

For each m:n relationship type R, we create a new table (say S) to represent R. We also include the primary key attributes of both the participating entity types as a foreign key attribute in S. Any simple attributes of the m:n relationship type (or simple components of a composite attribute) is also included as attributes of S.
For example, the m: n relationship taught-by between entities COURSE and FACULTY should be represented as a new table. The structure of the table will include primary key of COURSE and primary key of FACULTY entities. (Refer to *Figure 11*).

A new table TAUGHT-BY will be created as: Primary key of TAUGHT-By table.

ID	**Course_ID**
{Primary key of FACULTY table}	{Primary key of COURSE table}

Please note that since there are no attributes to this relationship, therefore no other fields.

Figure 11: Converting N:N relationship

n-ary Relationship:

For each n-ary relationship type R where n>2, we create a new table S to represent R. We include as foreign key attributes in s the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the n-ary relationship type (or simple components of complete attributes) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types. *Figure 11* is a special case of n-ary relationship: a binary relation.

Multivalued attributes:

For each multivalued attribute 'A', we create a new relation R that includes an attribute corresponding to plus the primary key attribute k of the relation that represents the entity type or relationship type that has as an attribute. The primary key of R is then combination of A and k.
For example, if a STUDENT entity has RollNo, Name and PhoneNumber where phone number is a multi-valued attribute then we will create a table PHONE (**RollNo, Phone-No)** where primary key is the combination. Please also note that then in the STUDENT table we need not have phoneNumber, instead it can be simply (Roll No, Name) only.

Converting Generalisation / Specialisation hierarchy to tables:

A simple rule for conversation may be to decompose all the specialised entities into tables in case they are disjoint. For example, for the *Figure 4* we can create the two tables as:

Saving_account (account-no holder-name, branch, balance, interest).

Current_account (account-no, holder-name, branch, balance, charges).

The other way might be to create tables that are overlapping for example, assuming that in the example of *Figure 4* if the accounts are not disjoint then we may create three tables:

account (account-no, holder-name, branch, balance)

saving (account-no, interest)

current (account-no, charges)

Check Your Progress 2

1) A Company ABC develops applications for the clients and their own use. Thus, the company can work in "user mode" and "developer mode". Find the possible entities and relationships among the entities. Give a step by step procedure to make an E-R diagram for the process of the company when they develop an application for the client and use the diagram to derive appropriate tables.

……

……

……

……

2) An employee works for a department. If the employee is a manager then s/he manages the department. As an employee the person works for the project, and the various departments of a company control those projects. An employee can have many dependents. Draw an E-R diagram for the above company. Try to find out all possible entities and relationships among them.

……

……

……

……

3) A supplier located in only one-city supplies various parts for the projects to different companies located in various cities. Let us name this database as "supplier-and-parts". Draw the E-R diagram for the supplier and parts database.

……

……

……

……

2.8 SUMMARY

This unit is an attempt to provide a detailed viewpoint of data base design. The topics covered in this unit include the relational model including the representation of relations, operations such as set type operators and relational operators on relations. The E-R model explained in this unit covers the basic aspects of E-R modeling. E-R modeling is quite common to database design, therefore, you must attempt as many problems as possible from the further reading. E-R diagram has also been extended. However, that is beyond the scope of this unit. You may refer to further readings for more details on E-R diagrams.

2.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. Relational databases and a language as powerful as relational algebra
2. File, Record, Field
3. Alternate
4. A minus (A minus B)

5. (d)
6.
   a) Accepted
   b) Rejected (candidate key uniqueness violation)
   c) Rejected (violates RESTRICTED update rule)

d) Accepted (supplier S3 and all shipments for supplier S3 in the relation SPJ are deleted)
e) Rejected (violates RESTRICTED delete rule as P2 is present in relation SPJ)
f) Accepted (project J4 and all shipments for project J4 from the relation SPJ are deleted)
g) Accepted
h) Rejected (candidate key uniqueness violation as tuple S5-P5-J7already exists in relation SPJ)
i) Rejected (referential integrity violation as there exists no tuple for J8 in relation J)
j) Accepted
k) Rejected (referential integrity violation as there exists no tuple for P7 in relation P)
l) Rejected (referential integrity violation – the default project number jjj does not exist in relation J).

7) The answer to this query will require the use of the relational algebraic operations. This can be found by selection supplies made by S1 in SPJ, then taking projection of resultant on JNO and joining the resultant to J relation. Let us show steps:

Let us first find out the supplies made by supplier S1 by selecting those tuples from SPJ where SNO is S1. The relation operator being:

$$SPJT = \sigma_{<SNO = 'S1'>} (SPJ)$$

The resulting temporary relation SPJT will be:

SNO	PNO	JNO	QUANTITY
S1	P1	J1	200
S1	P1	J4	700

Now, we take the projection of SPJT on PNO

$$SPJT2 = \Pi_{JNO}(SPJT)$$

The resulting temporary relation SPJT will be:

JNO
J1
J4

Now take natural JOIN this table with J:

RESULT = SPJT2 JOIN J

The resulting relation RESULT will be:

JNO	JNAME	CITY
J1	Sorter	Pune
J4	Console	Agra

Check Your Progress 2

1) Let us show the step by step process of development of Entity-Relationship Diagram for the Client Application system. The first two important entities of the system are **Client** and **Application**. Each of these terms represents a noun, thus, they are eligible to be the entities in the database.

But are they the correct entity types? Client and Application both are **independent** of anything else. So the entities, clients and applications form an entity set.

But how are these entities related? Are there more entities in the system?
Let us first consider the relationship between these two entities, if any. Obviously the relationship among the entities depends on interpretation of written requirements. Thus, we need to define the terms in more detail.
Let us first define the term **application.** Some of the questions that are to be answered in this regard are: Is the **Accounts Receivable (AR)** an application? Is the AR system installed at each client site regarded as a different application? Can the same application be installed more than once at a particular client site?

Before we answer these questions, do you notice that another entity is in the offering? The client **site** seems to be another candidate entity. This is the kind of thing you need to be sensitive to at this stage of the development of entity relationship modeling process.

So let use first deal with the relationship between Client and Site before coming back to Application.

Just a word of advice: "It is often easier to tackle what seems likely to prove simple before trying to resolve the apparently complex."

Each Client can have many sites, but each site belongs to one and only one client.

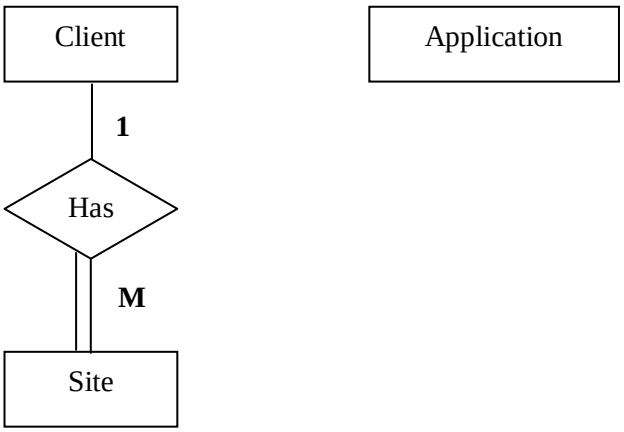


Figure 12: A One-to-Many Relationship

Now the question arises what entity type is Site? We cannot have a site without a client. If any site exists without a client, then who would pay the company? This is a good example of an existential dependency and one-to-many relationship. This is illustrated in the diagram above.

Let us now relate the entity Application to the others. Please note the following fact about this entity:

An application may be installed at many client sites. Also more than one application can be installed at these sites. Thus, there exists a many-to-many relationship between Site and Application:

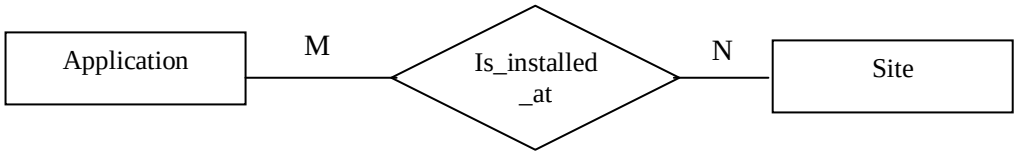


Figure 13: A Many-to-Many Relationship

However, the M: M relationship "is_installed_at" have many attributes that need to be stored with it, specifically relating to the available persons, dates, etc. Thus, it may be a good idea to promote this relationship as an Entity.

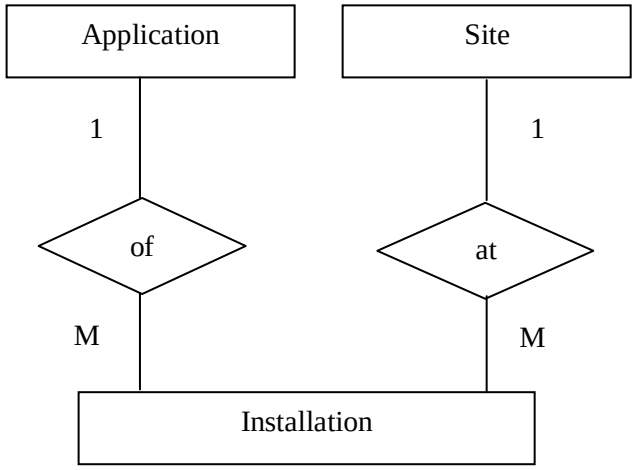


Figure 14: The INSTALLATION ENTITY

The *Figure* 14 above consists of the following relationships:

of - relationship describing installation of applications and
at - relationship describing installation at sites.

Please note that entities can arise in one of two ways. Either we draw them because they were named in the specification, or as a result of resolving a M:M relationship, as in this case. When we create a new entity in this way, we have to find a suitable name for it. This can sometimes be based on the **verb** used to describe the M:M. Thus, the statement **we can install the application at many sites** we can choose the verb install and covert it to related noun **Installation**.

But what is the type of this Installation entity?

The best person to do so is the user of this entity. By identifying type we mean to define the most effective way of uniquely identifying each record of this type. So now the questions that need to be answered are:

- How do we want to identify each Installation?
- Is an Installation independent of any other entity, that is, can an entity Installation exist without being associated with the entities Client, Site and Application?

In the present design, there cannot be an Installation until we can specify the Client, Site and Application. But since Site is existentially dependent on Client or in other words, Site is subordinate to Client, the Installation can be identified by (Client) Site (it means Client or Site) and Application. We do not want to allow more than one record for the same site and application.

But what if we also sell multiple copies of packages for a site? In such a case, we need to keep track of each individual copy (license number) at each site. So we need another entity. We may even find that we need separate entities for Application and Package. This will depend on what attributes we want to keep in each as our requirements differ in respect of each of these.

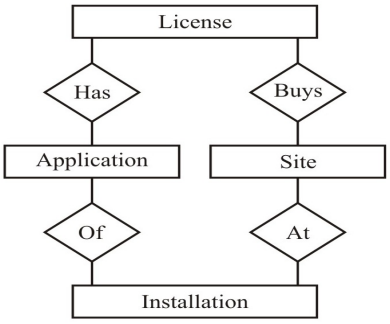


Figure 15: **Adding More Entities**

Let us define the Relationships in the Figure 15.
Has: describes that each application has one or more licenses
Buys: describes each site buys the licensed copies.

We might decide that License should be subordinate to Package: the best unique identifier for the entity could be Package ID and License serial number. We could also define License as identified by Client Site, Application and the serial number of the licensed copy. The only objection to the subordinate identification is that one of the key elements has an externally assigned value: the license numbers issued by the companies to whom the package belongs. We do not issue license numbers, and thus have no control over their length and data type (Can it be up to 15 mixed alpha-numeric characters?). Also we have no control over their uniqueness and changeability. Please note that **it is always safer to base primary keys on internally assigned values.**

What if we make License subordinate to Package, but substitute our own Copy serial number for the software manufacturer's License number? It also seems that the client site is not an essential part of the identifier, as the client is free to move the copy of a package to another site. Thus, we should definitely not make client/site part of the primary key of License.

Hence the discussion goes on till you get the final E-R diagram for the problem above. Please keep thinking and refining your reasoning. Here we present the final E-R diagram for the problem. Please note that knowing and thinking about a system is essential for making good ER diagrams.

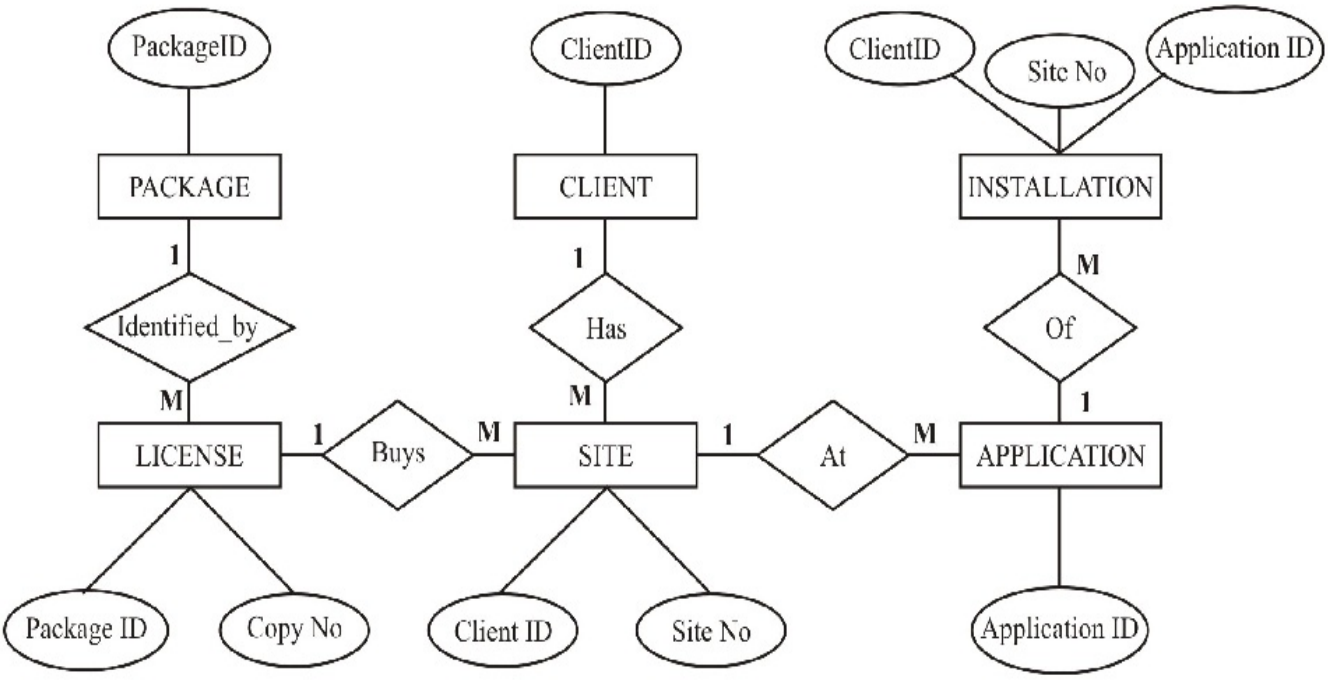


Figure 16: The Entity Types along with the attributes and keys

Let us list each entity identified so far, together with its entity type, primary keys, and any foreign keys.

Entity	Type	Primary Key	Foreign Keys
Client	Independent	Client ID	
Site	Subordinate	Client ID, Site No	
Application	Independent	Application ID	

Package	Independent	Package ID	
Installation	Combination	Client ID, Site No, Application ID	Client ID, Site No, Application ID
License	Subordinate	Package ID, Copy No	Package ID

2) The E-R diagram is given below:

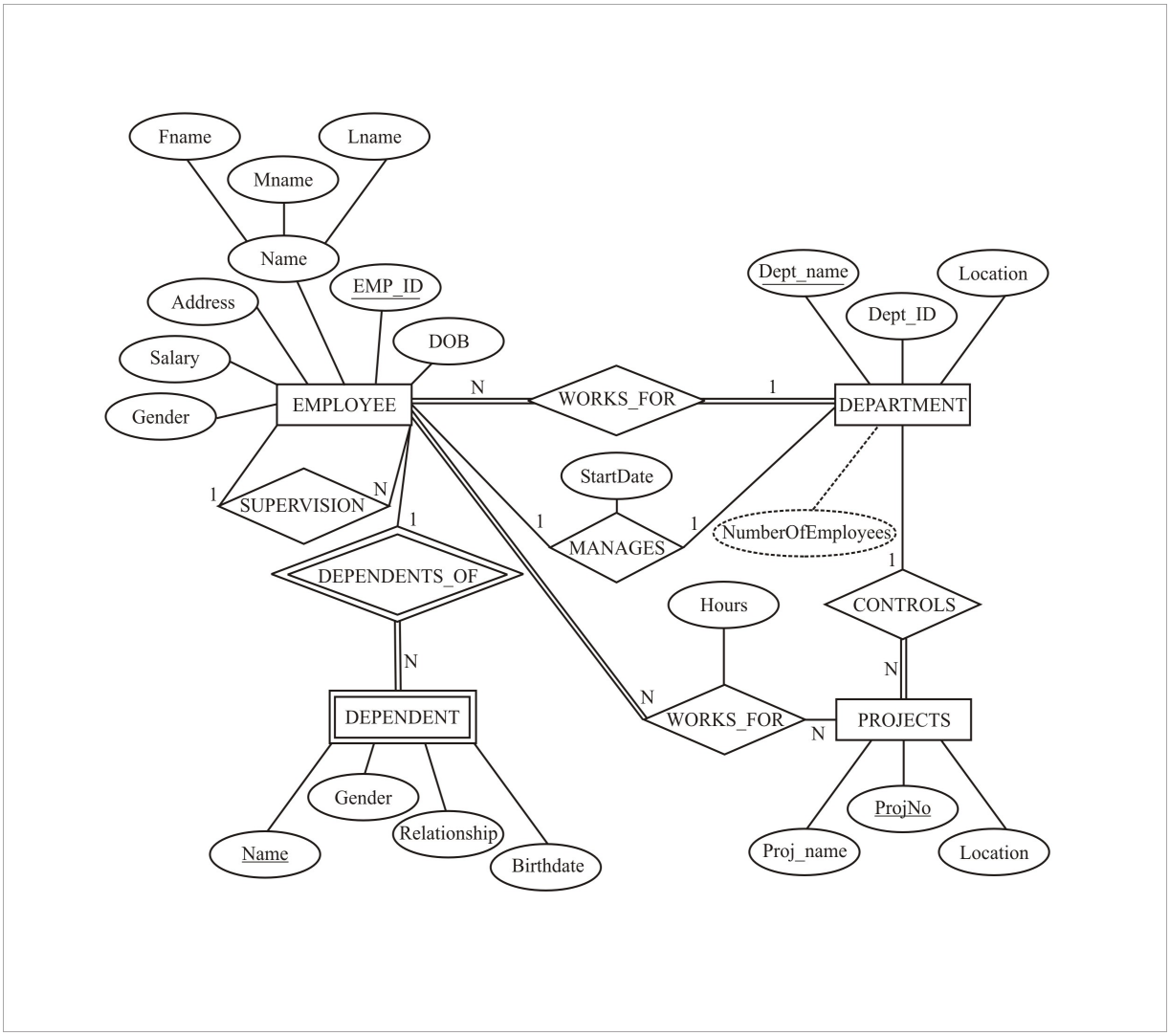


Figure 17: The E-R diagram for EMPLOYEE-DEPARTMENT database

In the above E-R diagram, EMPLOYEE is an entity, who works for the department, i.e., entity DEPARTMENT, thus **works_for** is many-to-one relationship, as many employees work for one department. Only one employee (i.e., Manager) manages the department, thus **manages** is the one-to-one relationship. The attribute Emp_Id is the primary key for the entity EMPLOYEE, thus Emp_Id is unique and not-null. The candidate keys for the entity DEPARTMENT are **Dept_name** and **Dept_Id**. Along with other attributes, NumberOfEmployees is the derived attribute on the entity DEPT, which could be recognised the number of employees working for that particular department. Both the entities EMPLOYEE and DEPARTMENT participate totally in the relationship works_for, as at least one employee work for the department, similarly an employee works for at least one department.

The entity EMPLOYEES work for the entity PROJECTS. Since many employees can work for one or more than one projects simultaneously, thus works_for is the N:N relationship. The entity DEPARTMENT controls the entity PROJECT, and since one department controls many projects, thus, controls in a 1:N relationship. The entity EMPLOYEE participate totally in the relationship works_for, as at least one employee

works for the project. A project has no meaning if no employee is working in a project.

The employees can have many dependents, but the entity DEPENDENTS cannot exist without the existence of the entity EMPLOYEE, thus, DEPENDENT is a weak entity. We can very well see the primary keys for all the entities. The underlined attributes in the eclipses represent the primary key.

3. The E-R diagram for supplier-and-parts database is given as follows:

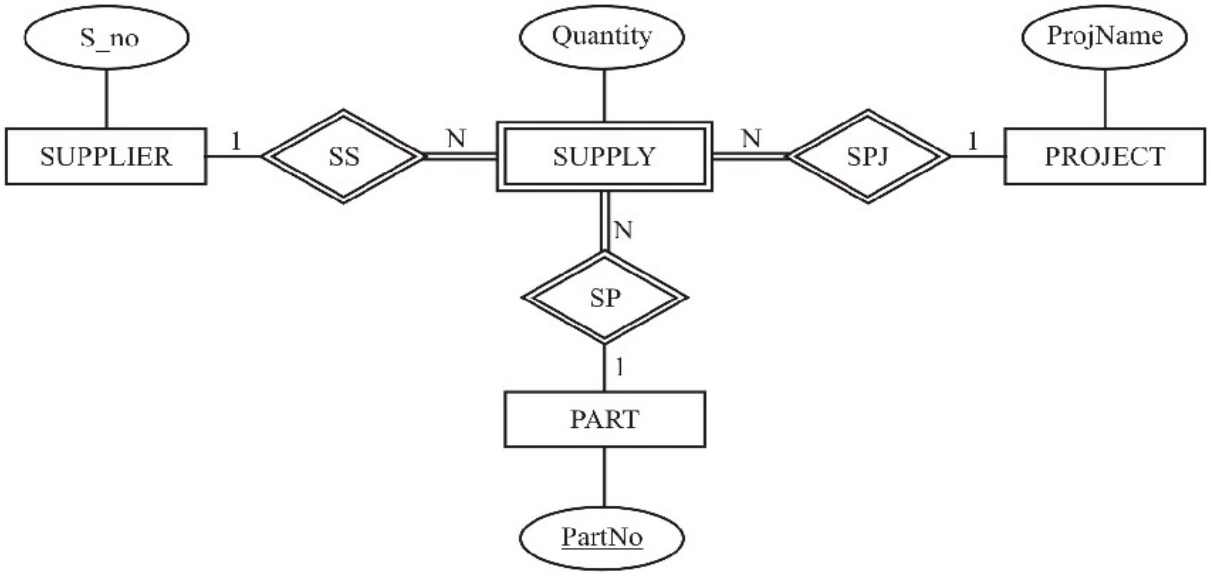


Figure 18: E-R diagram for Supplier-Project and entities