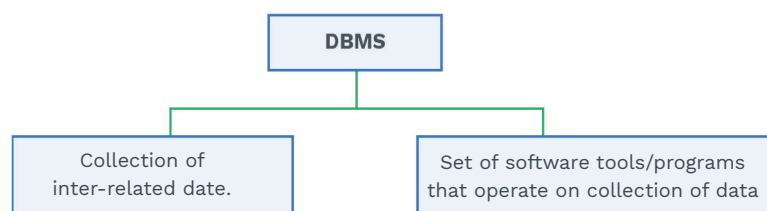


1. INTRODUCTION TO DBMS

A system which manages all such inter-related data is called “Database Management System”. It plays a vital role in almost all areas where computers are used.

For example, a company can have a database of their employees, including their name, address, age, date of birth and salaries.

DBMS:



Database management system can be defined as software that helps to maintain and utilize a huge amount of data.

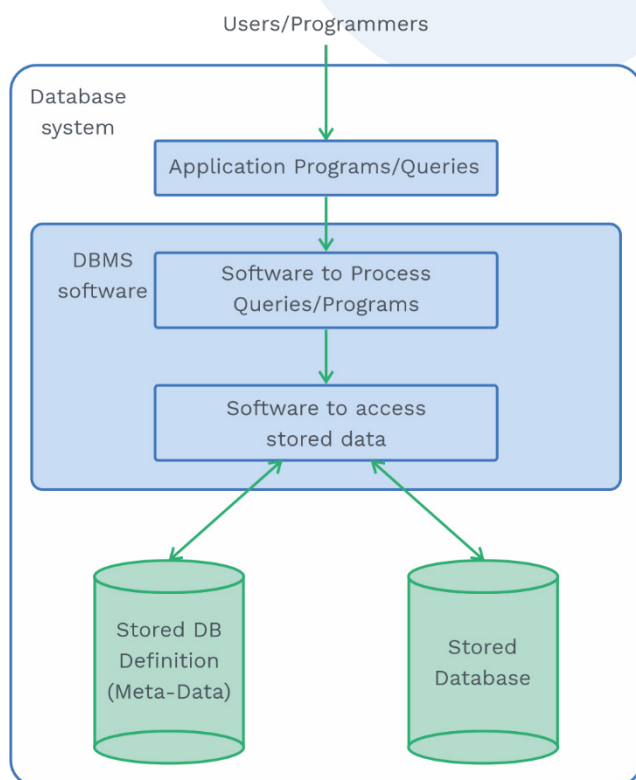


Fig. 1.1 A Simplified Database System Environment

**DBMS helps in:**

- 1) **Database design:** It helps to decide how to organise the stored data.
- 2) **Data analysis:** It tells to execute queries over data in DBMS.
- 3) **Concurrency and robustness:** DBMS allows multiple users to access data concurrently and protects data in case of system failures.
- 4) **Efficiency and scalability:** DBMS efficiently give answers to the queries and the cost-effectiveness, ability to accommodate increased workload and makes it scalable.

File systems versus a DBMS:

To store a large amount of data, we need a database management system. For example, a company has a large amount of data on employees, departments etc. In this case, only DBMS can give us the results, not file systems.

Let us consider an example: A company has a large amount of data, say 500 GB, on employees. Now, this data needs to be accessed concurrently by several employees, changes can be made to a certain parts of data, and these should be applied consistently and access to certain part of data must be restricted.

Using operating system files to store this huge amount of data, this may cause certain drawbacks.

- It is possible that we may not have 500 GB of main memory to hold all the data.
- We must write special programs to answer each question.
- Operating systems are inflexible to enforce security.

We can store data using DBMS instead of files to manage the data efficiently.

File System	DBMS
Data is stored on the disk	Data is being stored by database management system using indexing
Accessing any information is time consuming	Time required to access is comparatively less.
It can have redundant data	DBMS reduces redundancy

Table 1.1 File System Vs DBMS

**Note:**

DBMS stores everything in a file, but DBMS also includes a piece of software that helps manage these data efficiently.

The usefulness of different representations of the database:

- i) We know how entities are related. Therefore, the databases can be pictorially represented using E-R diagrams.
- ii) We need tables to store data. The model which talks about it is called the relational model.
- iii) Searching and retrieving is fast. Therefore, we use B-Trees.
- iv) The accessing of data can be made simple using transaction and concurrency control system.

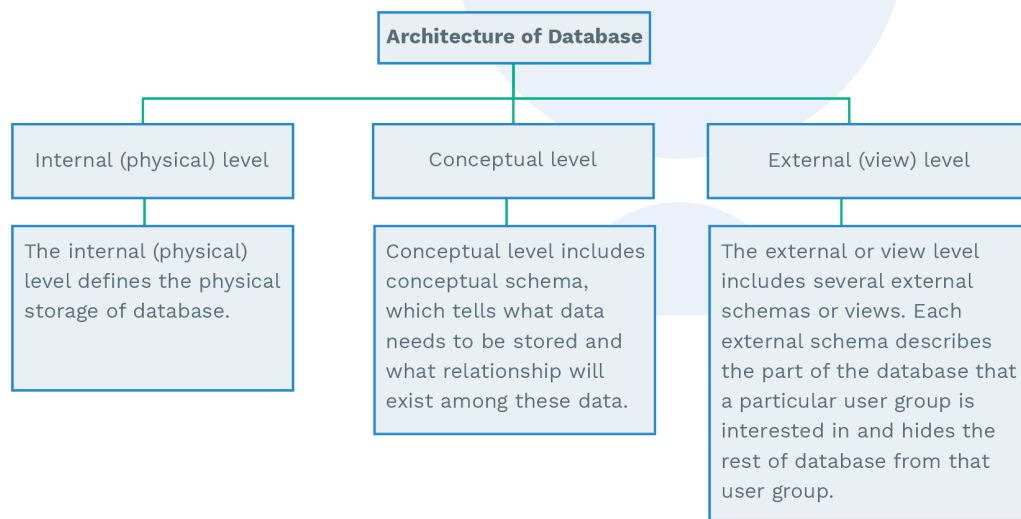
Architecture of database:

Fig. 1.2 Architecture of Database

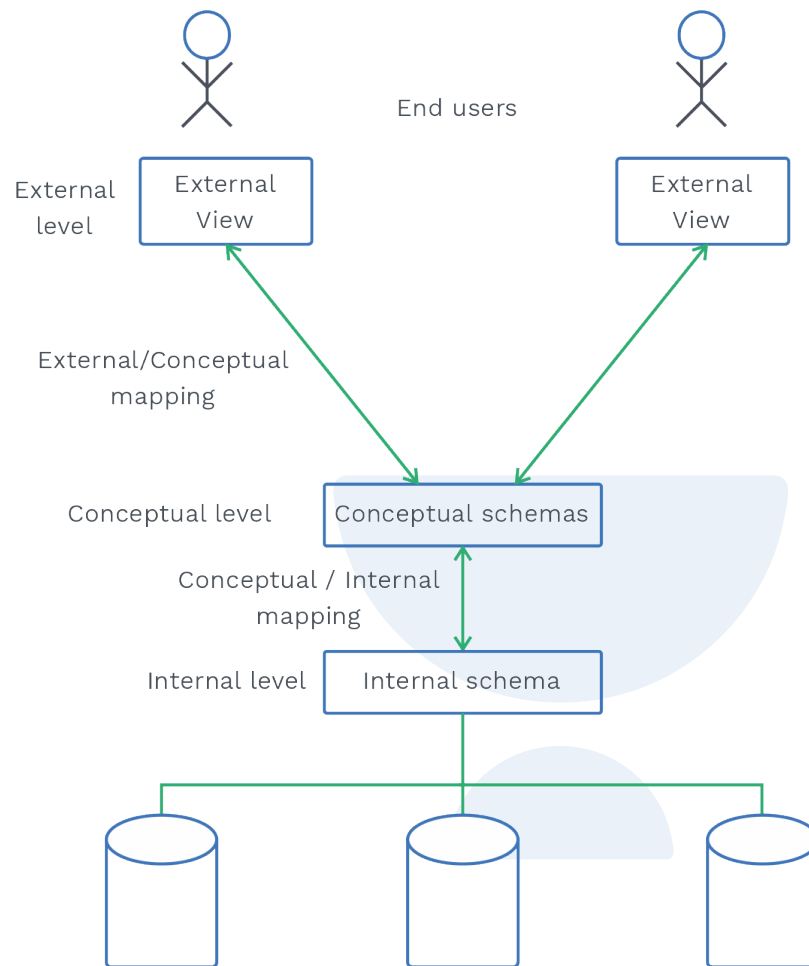


Fig. 1.3 Stored Database

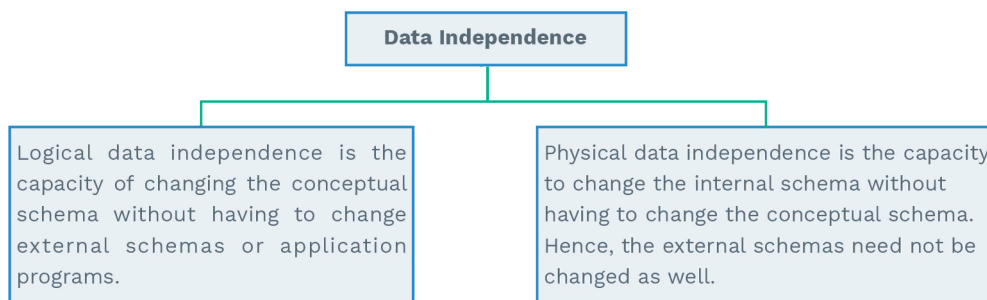


Fig. 1.4 Data Independence



Advantages of DBMS:

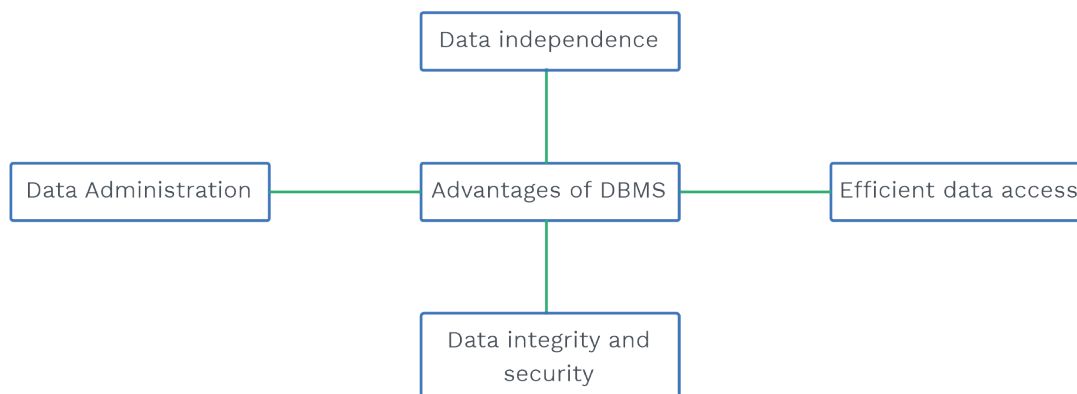


Fig. 1.5 Advantage of DBMS

Concept of keys and constraints:

Keys

An attribute or a set of attributes which helps in uniquely identifying a tuple (a row) in a relation (a table) is called a key.

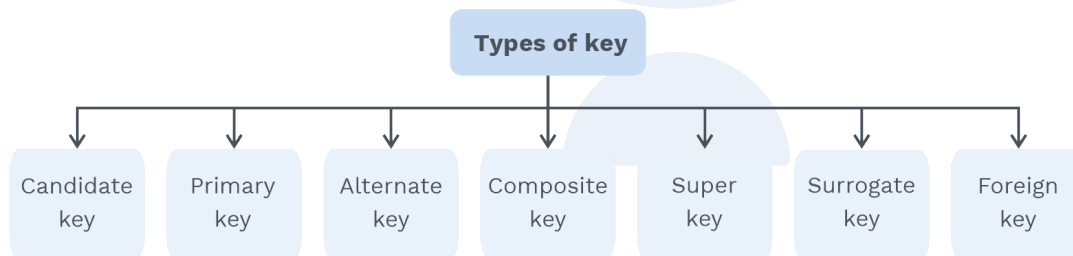


Fig. 1.6 Types of Keys

Compound key: Key with multiple attributes/columns.

Example: (emp_name, emp_dept) is a compound key.

Candidate key: The set of all unique keys are candidate keys.

Example: {emp-id, {emp-name, emp-dept}}

Rules while closing primary key:

- It can not be null of any tuple/row.
- There should be at most one primary key per table.
- It should be unique for each row/tuple.

Above mentioned conditions are called entity integrity constraints. Constraints are properties to be satisfied while inserting, deleting, or modifying the data in a relational table.

- **Foreign key:** It is an attribute/group of attributes in a relation database which relates two tables.



Example: Consider following two tables.

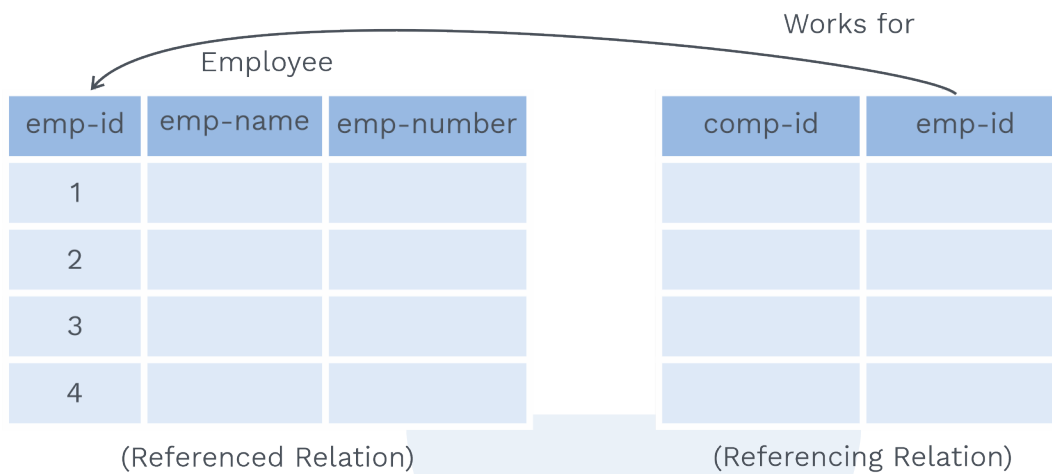


Table 1.2 Foreign Key

'Emp-id' is a foreign key in Works for relation. A foreign key should be the primary key in referenced relation, but it may or may not be the primary key in referencing relation.

Self-referential foreign keys: It defines relationships within the same table.

Example:

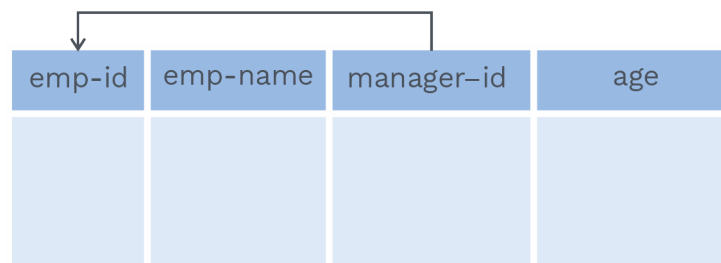


Table 1.3 Self-referential Foreign Keys

Here, the emp-dept id is a self-referential foreign key.

Surrogate key: It is an artificial key which uniquely identifies each record.

Super-key: It's a collection of one or more attributes that allow us to identify an entity in the relation uniquely. The superset of the candidate key is known as the super key in a relation.

Example: consider Emp-id as a candidate key in a relation Employee. Then, {Emp-id, Emp-name} is a super-key.



Q1 How many super keys are possible in $R (A_1, A_2 \dots A_n)$ when candidate key is $\{A_1\}$?

Sol: Super key = ((candidate key) \cup (other attributes))

Here, candidate key = A_1

Now, $A_1 \cup \phi, A_1 \cup A_2, \dots A_1 \cup A_2, A_3 \dots A_n$ all are possible super keys. It means A_1 can be combined with $(n-1)$ ways to other attributes in a given relation.

Using the concept of power sets:

Number of super keys = 2^{n-1} .

Q2 How many super keys are possible in relation $R (A_1, A_2, \dots A_n)$ if candidate keys are $\{A_1, A_2\}$?

Sol: As we have discussed in the above example

- Taking A_1 as a candidate key.
Number of super keys = 2^{n-1} , say A_1
- Taking A_2 as a candidate key.
Number of super keys = 2^{n-1} , say A_2
Using inclusion-exclusion principal

$$(|A_1 \cup A_2|) = |A_1| + |A_2| - |A_1 \cap A_2| = 2^{n-1} + 2^{n-1} - 2^{n-2} = 2^n - 2^{n-2}.$$



Rack Your Brain

How many super keys will be in a relation $R (A_1, A_2, \dots A_n)$ when the candidate keys are $\{\{A_1, A_2\}, \{A_3, A_4\}\}$



Rack Your Brain

Consider $R (A, B, C, D)$; how many superkeys are possible if

- A is the candidate key.
- $\{A, B, C\}$ are candidate keys.



Candidate key: The minimal superkey is defined as the candidate key.

Primary key: Randomly chosen candidate key is the primary key.

Let us consider two tables, student and subset, with mentioned attributes.

Student					Subjects	
S-id	S-name	S-acc no	S-roll no	Sub id	Sub-id	Sub-name

Table 1.4 Tables

Here,

- Candidate keys = {S-id, S-acc no, S- roll no}
- Primary key = {S-id}
- Alternate keys = {S-acc no, S-roll no}
- Super keys = {S-id, {S-id, S-name}, {S-id, S-name, S-roll no}...}
- Foreign key = {Sub-id}

Alternate Key: All other keys are alternate keys from the set of candidate keys, except the primary key.

Constraints:

As we have already discussed, constraints are conditions which must be satisfied before performing any operation on the database.

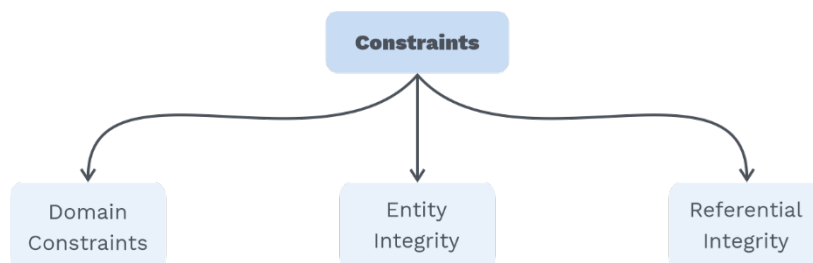


Fig. 1.7 Constraints

i) Entity integrity constraints:

- Every relation must have a primary key
- Every primary key must be unique
- A primary key should not be NULL



ii) Domain integrity constraints:

Domain integrity constraints just give a valid value for the attribute.

Example: An employee's age must be between 24 and 35, and the domain should be (integer) numeric value.

Referential integrity constraints:

- It is based on a foreign key concept.

Example:

Employee	emp-id	emp-name	dept-id
	1	Jannat	101
	2	Faisu	102
	3	Muskan	103

Table 1.5 Referencing Relation

Department	dept-id	dept-name
	101	Acting
	102	Singing

Table 1.6 Referenced Relation

Now, various operations can be performed on referencing and referenced relations.

The three basic operations are:

- Insertion
- Deletion
- Updation



Referenced Relation	Referencing Relation
<div>On insertion</div> <p>On inserting in referenced relation, there will be no change in referencing relation.</p>	<div>On inserting</div> <p>On inserting in referencing relation, first it will check if it is not violating referential integrity constraint.</p>
<div>On deletion</div> <p>On deleting a row, changes will be made:</p> <ul style="list-style-type: none">i) On delete no actionii) On delete cascade. This will be a deletion of a row from referencing relation corresponds to particular key.iii) On delete set null: It will set value null corresponding referencing tuple.	<div>On deletion</div> <p>On deletion in referencing relation, it will bring no change.</p>
<div>On updation</div> <ul style="list-style-type: none">i) On updation no action.ii) On updation cascade It will update the changes in referencing relation.iii) On update set null is not widely used.	<div>On updation</div> <p>It will also check for violation.</p>

Table 1.7 Referenced Vs Referencing Relation**Note:**

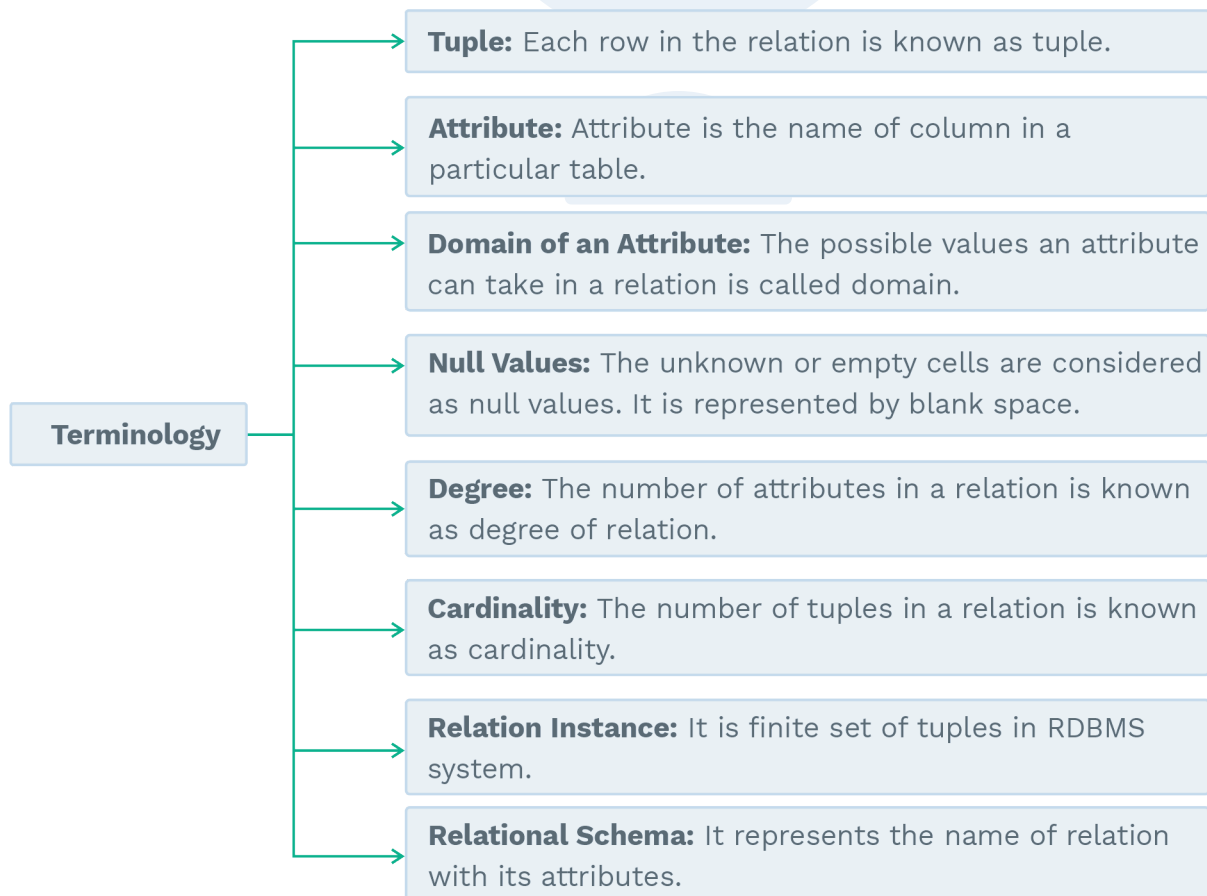
- On delete cascade can result in loss of data.
- On delete, no action violates referential integrity.

Relational model:

The database is essentially represented as a collection of relations in the relational paradigm. E.F Codd proposed it to model data in the form of relations or tables.

**Example:**

emp-id	Name	Address	Contact	Dept	Age
1	Sushant	Mumbai	9213213400	CS/IT	27
2	Rhea	Bhopal	9132123400	Civil	31
3	Showick	West Bengal	9123912300	Mechanical	24
4	Kangana	Himachal	9989998900		37
5	Diljit	Punjab	9789978900	Electrical	29
6	Arnab	Delhi	8760876000	Communication	42

Table 1.8 Relational Model**Terminology:****Fig. 1.8 Terminology**

**Note:**

Relation instances never have duplicate tuples.

Example: Employee

emp-id	Name	Address	Contact	Dept	Age
1	Sushant	Mumbai	9213213400	CS/IT	27
2	Rhea	Bhopal	9132123400	Civil	31
3	Showick	West Bengal	9123912300	Mechanical	24
4	Kangana	Himachal	9989998900		37
5	Diljit	Punjab	9789978900	Electrical	29
6	Arnab	Delhi	8760876000	Communication	42

Table 1.9 Employee Data

Tuple: 1 Sushant Mumbai 9213213400 CS/IT 27

Attributes: emp-id, Name, Address, Contact, Dept, Age.

The domain of attribute Age: The age range should be greater than 20 and less than 45.

Null values: emp-id = 4 has no dept assigned.

Cardinality: 6

Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called RDBMS. There are 13 popular rules known as Codd's 12 rules:

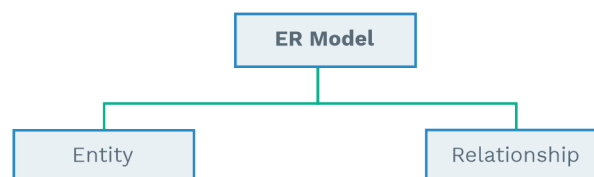


Fig. 1.9 ER Model



The E-R model consists of three basic terms:

- Entity sets
- Relationship sets
- Attributes



Fig. 1.10 Codd Rules

Entity sets:

An entity is a distinct object that can be differentiated from other items. It has a set of properties, some of which can be used to identify an entity. For example, an employee will have an id which uniquely identifies one



particular employee. In addition, an entity can be concrete (person, book) or abstract (loan, holiday or concept).

A set of entities of the same type with the same features or attributes is referred to as an entity set. Individual entities that make up a set are referred to as the extension of the entity set in a database management system (DBMS). All individual bank clients, for example, are an extension of the entity set customer.

It is not required for entity sets to be disjoint. A set of attributes represents an entity. Each of the attributes of an entity has a value. For example, emp-id, emp-name, and emp-address may have value for a certain employee object.

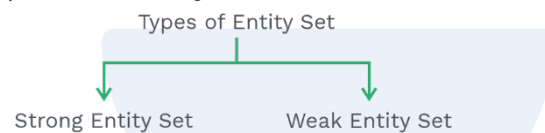


Fig. 1.11 Types of Entity Set

Strong entity set:

It has enough properties to identify each of its entities uniquely. So, formally we can say that a strong entity set has a main key. A rectangle represents it.

Example:

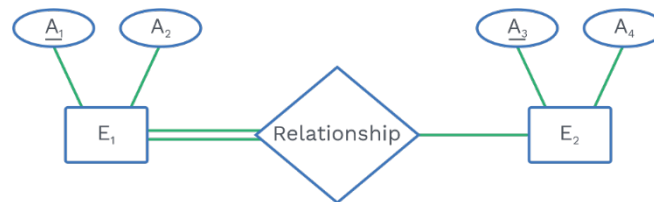


Fig. 1.12 Strong Entity Set

Where, E = Entity set
A = Attribute

Weak entity set:

It lacks sufficient properties to allow its entities to be uniquely identified. Although the weak entity set does not have a primary key, it does have a partial key, known as a discriminator, that can identify a group of entities from the entity set. A dashed line is used to depict it.

Weak entity sets are represented by a double rectangle, while the relationship between strong and weak entity sets is represented by a double diamond symbol. This relationship is known as the identifying relationship.

**Note:**

Total participation exists in a relationship involving weak entities, from the weak entity terminal.

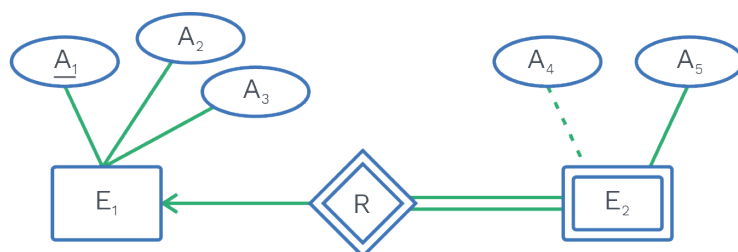
Example:

Fig. 1.13 Weak Entity Set

Where, E_2 is a weak entity set
 E_1 is a strong entity set
 A_4 is the discriminator of a weak entity set E_2

Relationship sets:

A connection is essentially an association between two or more entities. A relationship set is a collection of similar relationships.

The association between entity sets is referred to as participation. The entity's function in a relationship is called that entity's role.

Note:

A relationship may also have attributes called descriptive attributes.

The number of entity sets that participates in a relationship set is called the degree of a relationship set. A binary relationship set is degree 2, a ternary relationship set is degree 3.

Attributes:**Attributes can be of the following types:**

- 1) Simple and composite attributes:** The attributes which cannot be decomposed into a set of smaller independent attributes are classified as simple attributes. Composite attributes can be further split into more than one simple attribute.

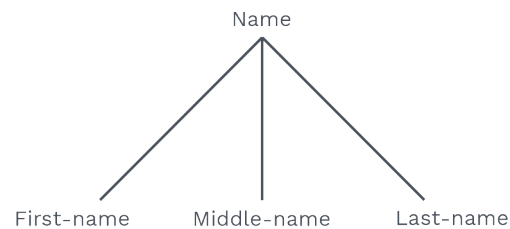


Fig. 1.14 Composite Attributes Employee Name

- 2) **Single-valued and multivalued attributes:** Single valued characteristics are attributes that have a single value for a specific entity. A multivalued attribute is one that has a collection of values for a certain entity.
- 3) **Derived attribute:** An attribute whose field can be derived from other attributes is classified as a derived attribute.
Consider the following scenario: The date of birth can be used to calculate age.

Constraints:

There are two major types of constraints which we will discuss:

- Mapping cardinalities
 - Participation constraints
- i) **Mapping cardinalities:** Mapping cardinality denotes the number of associativity from one entity set to another. The following forms of mapping cardinalities for a binary relationship set R on entity sets A and B are possible:
 - 1) **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.

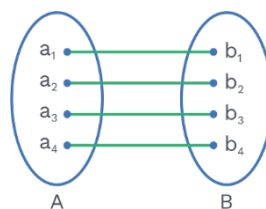


Fig. 1.15 One-to-one Mapping

- 2) **One-to-many:** A single entity in set A can be linked to any number of entities in set B, but one entity in set B can be linked to at most one entity in set A.

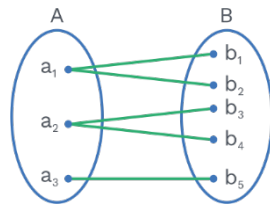


Fig. 1.16 One-To-Many Mapping

- 3) **Many-to-one:** Every component in set A must have at most one correspondence in set B. Every component in set B may have zero or more correspondences in set A.

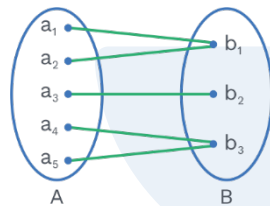


Fig. 1.17 Many-to-one Mapping

- 4) **Many-to-many:** Every component in set A may have any number of correspondences in set B and conversely.

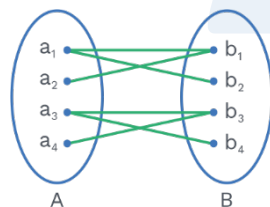


Fig. 1.18 Many-to-many Mapping

ii) **Participation constraints:**

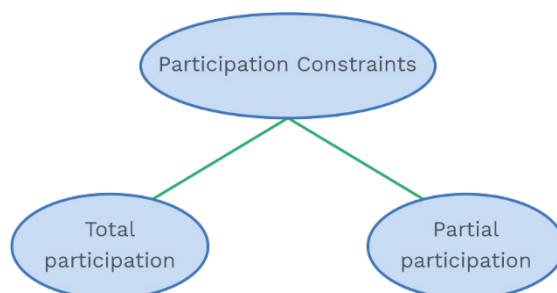


Fig. 1.19 Participation Constraints



If every entity in E participates in at least one relationship in R , the participation is said to be total.



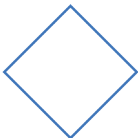




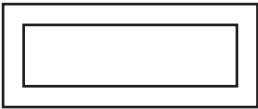
However, if just some entities participate in relationship R , entity set E 's participation is considered partial.

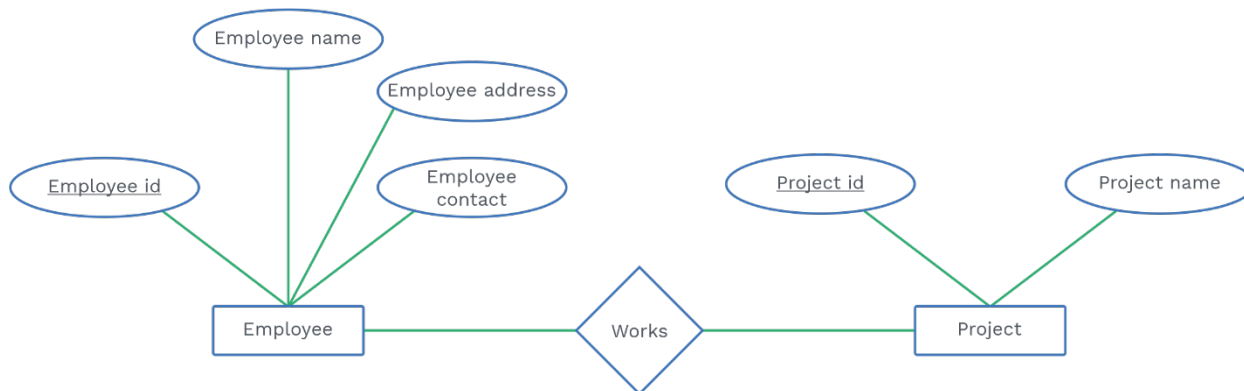
Relationship sets:

Let R be a set of relationships involving entity sets E_1, E_2, \dots, E_n . Let the primary key (E_i) signify the set of attributes that make up the entity set E_i 's primary key.

Entity-relationship diagram:

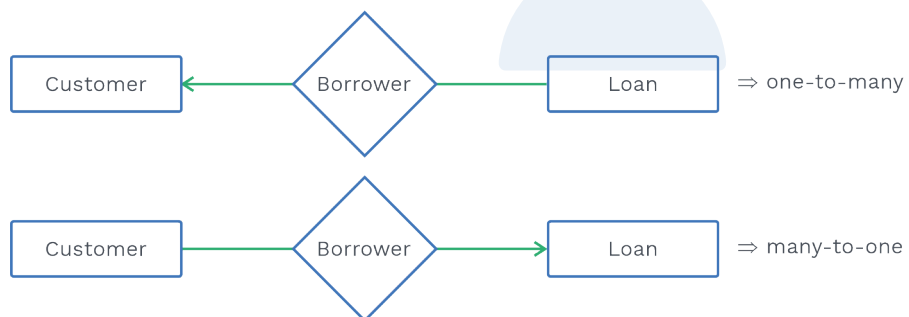
ER models consist of the following major components.

- Rectangle  Represent entity set
- Eclipse  Represent attributes
- Diamonds  represent relationship sets
- Lines  Attributes are linked to entity sets, while entity sets are linked to relationship sets.
- Double eclipse  represent Multivalued attributes
- Dash oval  represents derived attributes
- Double lines  Total participation
- Double rectangles  Weak entity sets

**Example of ER diagram:****Fig. 1.20 ER Diagram**

Here in the above diagram:

- Underlined entity is the primary key.
- An undirected line represents a many-to-many relationship.
- A directed line may represent a one-to-many or many-to-one relationship.

Example 1:**Fig. 1.21 Example**

Example 2: The following figure shows ER diagram with composite, multivalued and derived attributes.

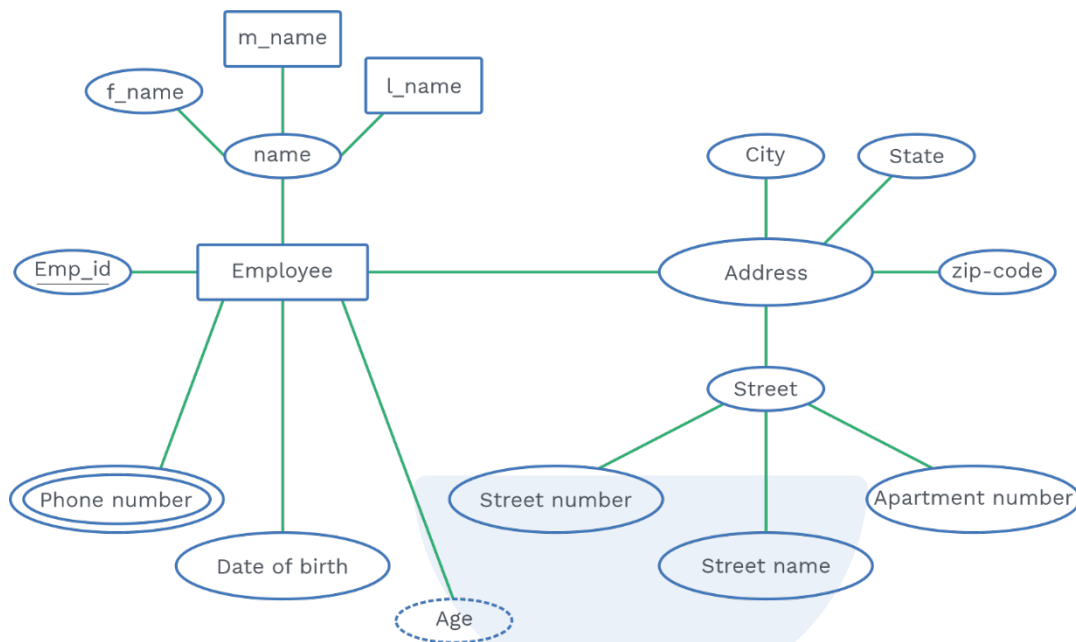


Fig. 1.22 ER Diagram

Minimization of ER diagrams:

We can minimize ER diagrams into tables because RDMS can easily organise tables.

To minimize ER diagrams, following rules should be kept under consideration.

- I) A strong entity with only simple attribute will require only one table.
- Attributes of the entity set will be the attributes of table.
 - Primary attributes of the entity set will be primary key of table.

For example:

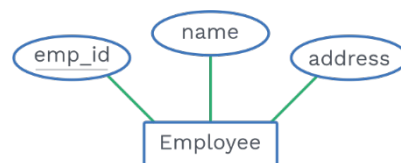


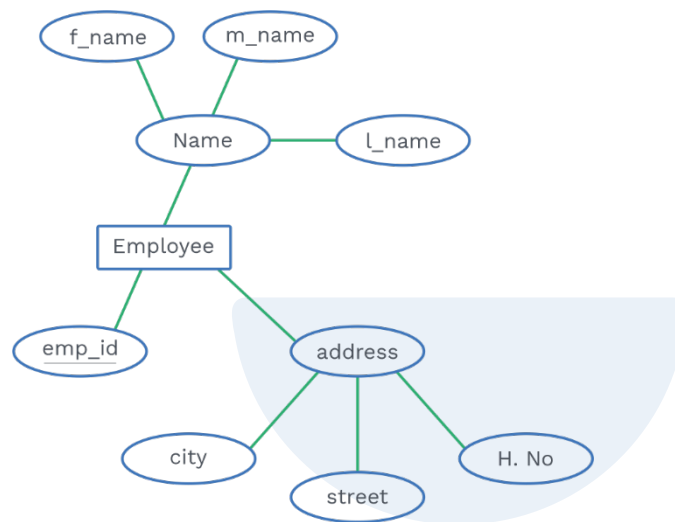
Fig. 1.23 ER Diagram

⇒

<u>emp_id</u>	name	address

**II) Check for strong entity set with composite attributes**

- A strong entity with any number of composite attributes will require only one table.

Example:**Fig. 1.24 ER Diagram**

⇒

<u>emp_id</u>	f_name	m_name	l_name	City	Street	H.No.

III) Check for strong entity set with multivalued attributes.

This requires two tables:

- One table will contain all simple attributes with primary key.
- Other table will contain primary key all the multivalued attributes.

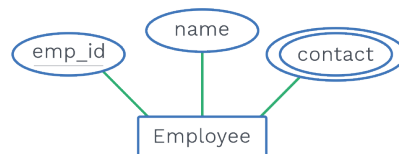
Example:**Fig. 1.25 ER Diagram**



Table 1:

<u>emp_id</u>	name

Table 2:

<u>emp_id</u>	contact

- IV)** Translation of relationship set into a table:
It requires one table in relational model and its attributes should be:
- Primary key attributes of entity sets
 - Descriptive attributes if any

Example:

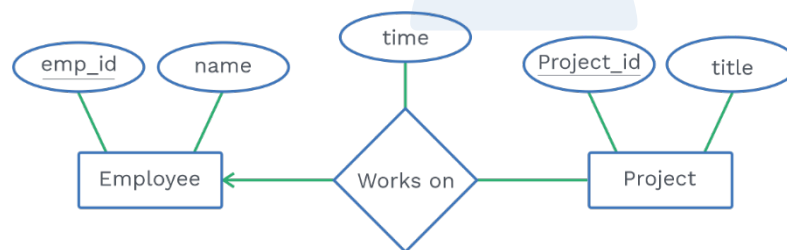


Fig. 1.26 ER Diagram

<u>emp_id</u>	<u>proj_id</u>	time

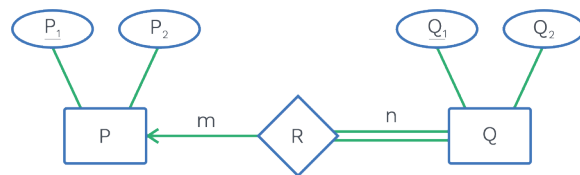
This design requires three relations- Employee, Project, Works on.

- V)** Check for binary relationships with cardinality ratios.

**Case I:**

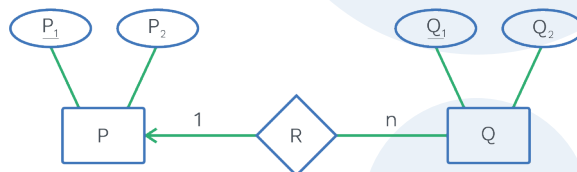
Many-to-many

Example:

**Fig. 1.27 Many-to-one Example**

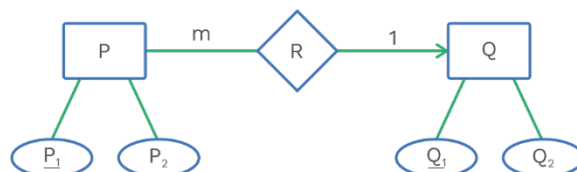
Here, three tables will be required:

- P (P₁, P₂)
- Q (P₁, Q₁)
- R (Q₁, Q₂)

Case II: One-to-many:**Fig. 1.28 One-to-many Example**

Here, two tables will be required

- QR (P₁, Q₁, Q₂)
- P (P₁, P₂)

Case III: Many-to-one:**Fig. 1.29 Many-to-one Example**

Here, two tables will be required:

- PR (P₁, P₂, Q₁)
- Q (Q₁, Q₂)



Case IV: One-to-one:

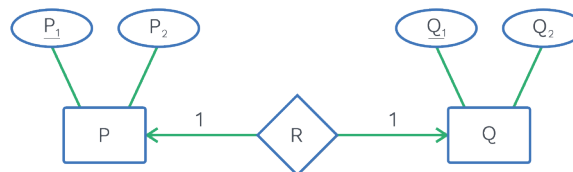
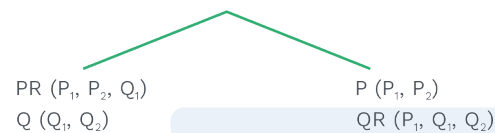


Fig. 1.30 One-To-One Example

In this case, there are two ways



VI) Check for the binary relationship between cardinality and participation constraints.

Case I: On one side, there is a binary relationship between a cardinality constraint and a total participation constraint.

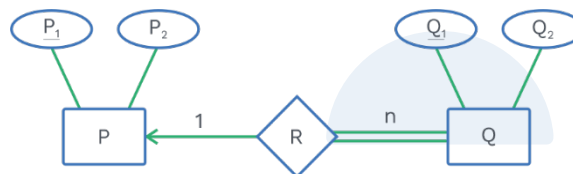


Fig. 1.32 ER Diagram

As we have already discussed, for one-to-many, two tables will be required

- $P(P_1, P_2)$
- $QR(Q_1, P_1, Q_2)$

But, because of total participation, the foreign key acquires Not NULL constraint, i.e., the new foreign key cannot be null.

Case II: Binary relationship with cardinality constraint and total participation constraint from both sides.

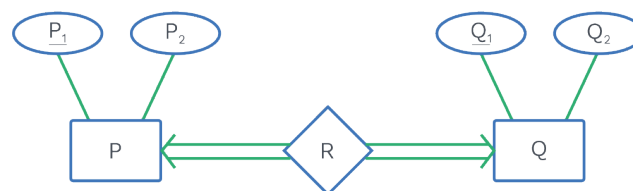


Fig. 1.33 Binary Relationship Diagram



Please keep in mind that if both sides of an entity set have a key constraint with total participation, the binary connection is represented using only one table.

Therefore,

PRQ is the table that will generate in this case (P_1, P_2, Q_1, Q_2)

VII) With a weak entity set, view for binary relationships. Please remember that a weak entity set is always associated with an identifying connection that has a total participation constraint.

Example:

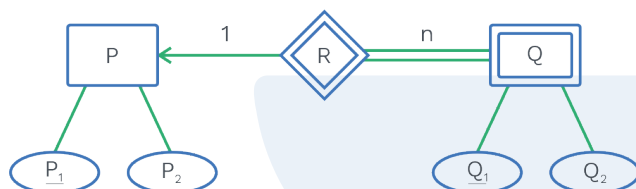


Fig. 1.34 ER Diagram

Therefore, here two tables will be required:

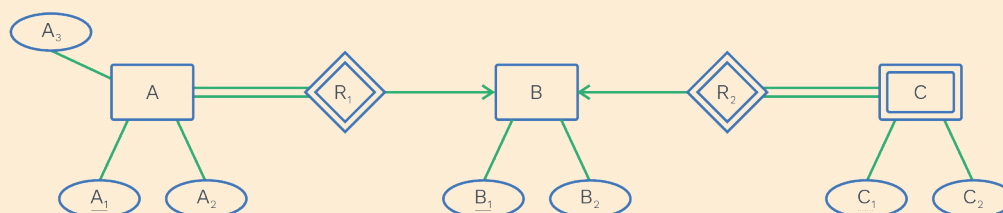
- P (P_1, P_2)
- QR (P_1, Q_1, Q_2)

The above mentioned seven rules are the basic and main rules used in reducing of E-R diagrams into tables.

SOLVED EXAMPLES

Q3

Find the minimum number of tables required for the following ER diagram in relational model.



Sol: With the help of the above discussed rules, we can conclude that a minimum of 3 tables will be required.

**Table 1:**
$$AR_1 =$$

<u>A₁</u>	A ₂	A ₃	B ₁

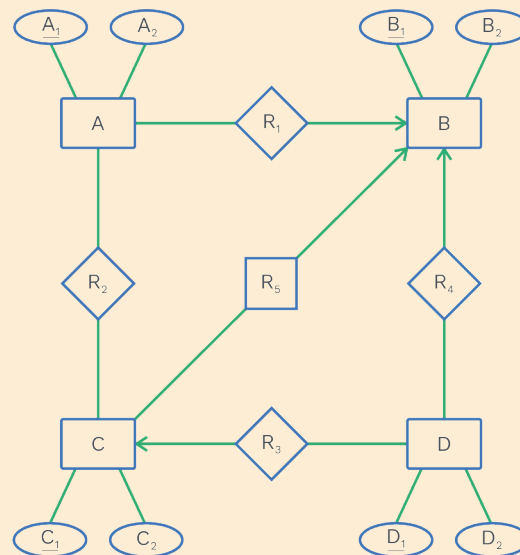
Table 2:
$$B =$$

<u>B₁</u>	B ₂

Table 3:
$$CR_2 =$$

<u>B₁C₁</u>	<u>C₁</u>	C ₂

Q4 Consider the below ER model:



How many are the minimum number of relations needed to organise the above ER model into RDBMS design?

Sol: Considering the above given ER diagram and applying rules, minimum 5 tables will be required.



Table 1:

AR_1	$\underline{A_1}$	A_2	B_1

Table 2:

B	$\underline{B_1}$	B_2

Table 3:

CR_5	$\underline{C_1}$	C_2	B_1

Table 4:

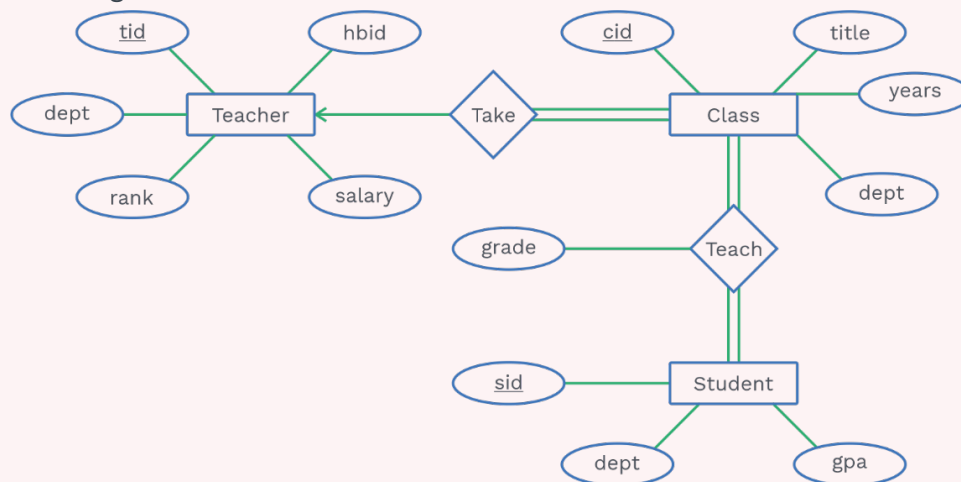
DR_3R_4	$\underline{D_1}$	D_2	C_1	B_1

Table 5:

$R_2 =$	$\underline{A_1}$	$\underline{C_1}$

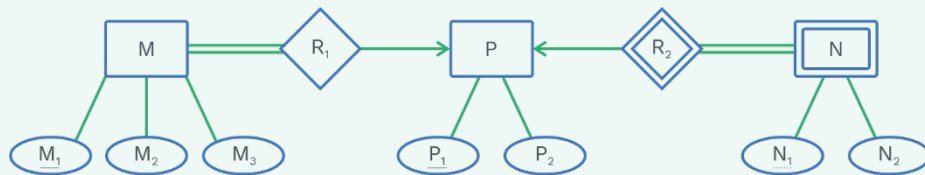
**Rack Your Brain**

Q. Find the minimum number of tables required for the given ER diagram.



**Previous Years' Question**

Consider the following ER diagram.



Which of the following is a correct attribute set for one of the tables for the minimum number of tables needed to represent M, N, P, R₁, R₂?

- 1) {M₁, M₂, M₃, P₁}
- 2) {M₁, P₁, N₁, N₂}
- 3) {M₁, P₁, N₁}
- 4) {M₁, P₁}

Sol: 1)

(GATE-CSE 2008)

Self-referential relationship:

It is the relation from an entity to itself. For example, a manager being an employee managing another employee.



General diagram of show the self-referential relationship:

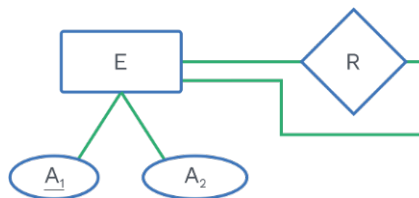


Fig. 1.35 Self-Referential Relationship



Case 1: One-to-one relationship:

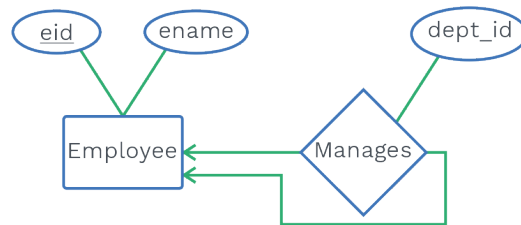


Fig. 1.36 One-to-one Relationship

Only one table will be required.

Case 2: One-to-many relationship

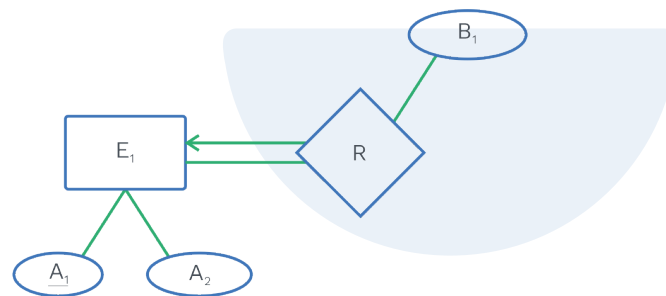


Fig. 1.37 One-to-many Relationship

Only one table will sufficient.

Case 3: Many-to-many relationship:

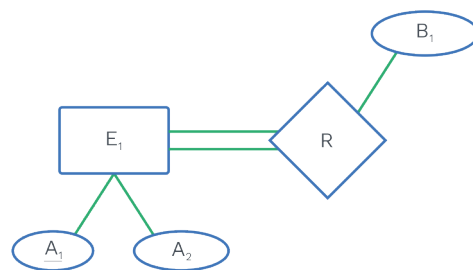


Fig. 1.38 Many-to-many Relationship

Two table will be required.

Extended ER-features:

Specialization:

- Specialization refers to the presence of more detailed attribute set belonging to an entity that uniquely defers it from other entities.
- Consider a person entity set with attributes such as name, street, and city. A person can also be described as one of the following:



- Customer & Employee
- Specialization is the process of identifying subgroupings within an entity set. We can discriminate between employees and customers using person specialization.
- Specialization is represented in the E-R diagram by the triangle component labelled ISA, which stands for “is-a” and symbolises, for example, a client is a person. A superclass-subclass relationship is another name for this relationship.

Generalization:

Generalization refers to the similarity of attributes shared between two or more entities.

For all the practical purposes, generalization is simple inversion of specialization. “Difference in the two approaches may be characterized by their starting point and our all goal.

Specialization stems from a single entity set, it emphasises difference among entities within the set by creating distinct lower-level entity sets. Generalization proceeds from the recognition that a number of entity sets share some common features.”

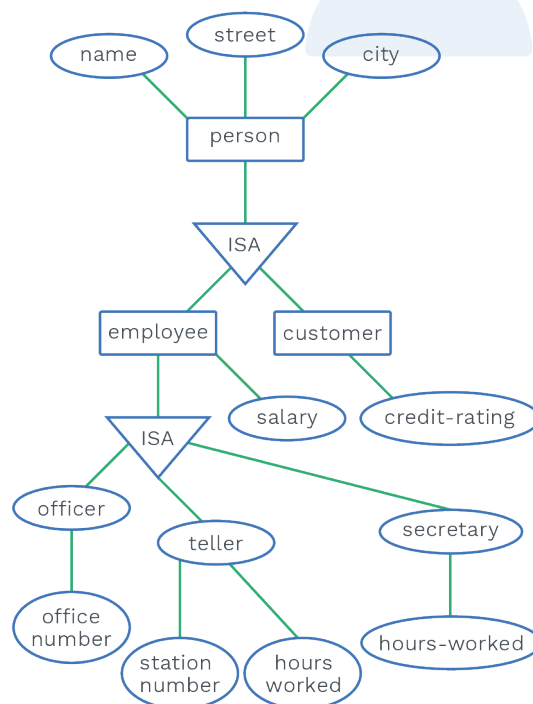


Fig. 1.39 Specialization and Generalization



Function dependency

In this topic we are going to study the following things

- Functional Dependencies
- Types of functional dependency
- Rules of functional dependency
- Attribute closure
- Minimal cover
- Problem caused by redundancy
- Equivalence set of functional dependency

Here, we will concentrate on an important class of constraints called functional dependencies.

Definition

Functional dependency (FD) is a kind of integrity constraint that generalizes the concept of a key.

Let R be a relation and let X & Y be non-empty sets of attributes in R . If the following holds for every Tuple t_1 and t_2 in r , we claim that an instance r of R fulfils the FD $X \twoheadrightarrow Y$.

If $t_1 \times X = t_2 \times X$ then $t_1 \times Y = t_2 \times Y$.

Where t_1 and t_2 are two different tuples.

Consider an entity set employee with attributes emp-id, emp-name, emp-address.

Employee (emp-id, emp-name, emp-address)

Now,

- 1) If given an employee id, we can uniquely determine the employee's name. It can be represented as:
 $\text{emp-id} \rightarrow \text{emp-name}$... (i)
- 2) Similarly, given an id, we can uniquely determine the address
Therefore, $\text{emp-id} \rightarrow \text{emp-address}$... (ii)
Now, combining (i) & (ii) we get
 $\text{emp-id} \rightarrow \text{emp-name, emp-address}$

**Note:**

‘ \rightarrow ’ is a notation which means LHS functionally determines RHS.

Generalization, given a relation $R(A_1, A_2, \dots, A_n)$

Let $\{A_1, A_2, A_3\} \rightarrow \{A_4, A_5\}$

Now, let's try to understand this concept using set theory.

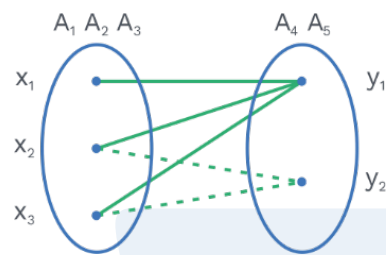
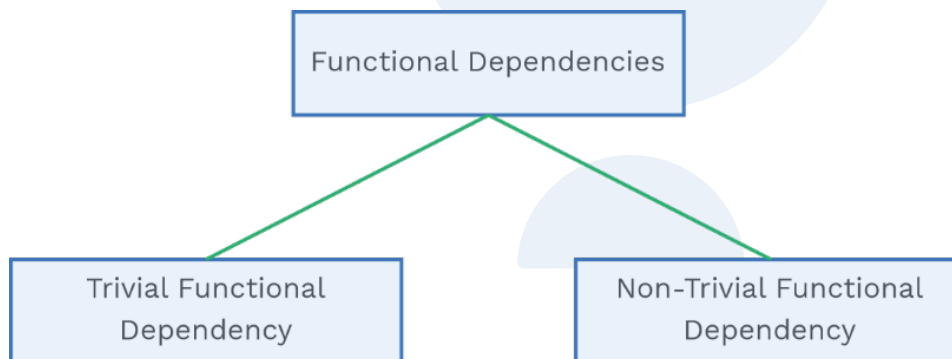
**Types of functional dependencies:**

Fig. 1.40 Types of Functional Dependencies

- Any Functional dependency is said to be Trivial iff LHS is a superset of RHS.
Considering the above example:
 $\text{emp-id} \rightarrow \text{emp-id}$
 $\text{emp-id, emp-name} \rightarrow \text{emp-id}$
 $\text{emp-id, emp-name} \rightarrow \text{emp-name}$
- Any Functional dependency is said to be non-trivial if LHS is not a superset of RHS :
For example, $\text{emp-id} \rightarrow \text{emp-name}$

Rules of functional dependency:

Over a relation schema R , we employ X , Y , and Z to denote sets of attributes:



1. Reflexivity: If $X \supseteq Y$, then $X \rightarrow Y$

2. Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

3. Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

There are some additional rules as well.

1) Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

2) Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Attribute closure of functional dependency:

A collection of all Functional dependencies implied by a particular set F of Functional Dependency is defined as the closure of F . To check if a given Functional dependency is in the closure of a set F of FDs, we can do this using the following algorithm.

Closure = X

Repeat until there is no change: {

If there is an FD: $U \rightarrow V$ in F such that

$U \subseteq \text{closure}$,

then set closure = closure $\cup V$

}

Attribute closure of X : It is a set of attributes that can be functionally determined using attributes in X .

Example 1: Given relation $R(A, B, C)$ and $FD = A \rightarrow B, B \rightarrow C$

Now, let's compute the closure of A (A^+), which is a set of all attributes which can be determined using A .

$$A^+ = \{A, B, C\}$$

Similarly:

$$B^+ = \{B, C\}$$

$$C^+ = \{C\}$$

Example 2: $R(A, B, C, D, E, F, G)$ and set of $FD =$

{

$$AB \rightarrow CD$$

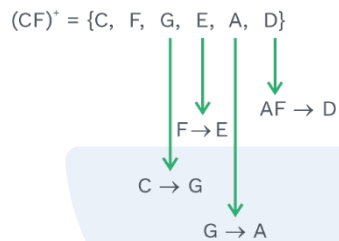


$AF \rightarrow D$
 $DE \rightarrow F$
 $C \rightarrow G$
 $F \rightarrow E$
 $G \rightarrow A$

}

Find closure of CF?

Sol:



Closure of attributes helps us to determine super keys and candidate keys.

- If F^+ gives all attributes of R, then F is super key.

Note:

Attributes which are part of atleast one candidate key are called prime attribute, non-prime otherwise, and the concept of prime attribute helps us to determine all possible candidate keys.

SOLVED EXAMPLES

Q5

Consider a relation R with the attributes (P, Q, M, S, T, U, V, W) with the following functional dependencies :

$P \rightarrow Q, S \rightarrow V, TU \rightarrow W, MS \rightarrow T, Q \rightarrow U, U \rightarrow S, S \rightarrow P, T \rightarrow M$. Which of the following is/are keys for relation R?

1) MS

2) STU

3) PS

4) PU

Sol: 1) and 2)

1) $MS \rightarrow T, S \rightarrow V, S \rightarrow P, P \rightarrow Q, Q \rightarrow U, TU \rightarrow W$.

$(MS)^+ = \{P, Q, M, S, T, U, V, W\}$

\therefore MS is a key for R.

2) $(STU)^+ = \{S, T, U, V, W, P, M, Q\}$

\therefore STU is a key for R.



- 3) $(PS)^+ = \{P, S, Q, V, U\}$
 $\therefore PS$ is not a key for R .
- 4) $(PU)^+ = \{P, U, Q, S, V\}$
 $\therefore PU$ is not a key for R .

Q6 $R(ABCDE)$
FD = $\{A \rightarrow BCDE,$
 $BC \rightarrow ADE,$
 $D \rightarrow E\}$
Find candidate key?

Sol: $A^+ = \{A, B, C, D, E\}$
Now, $B^+ = \{B\}$
 $C^+ = \{C\}$
 $D^+ = \{D\}$
 $E^+ = \{E\}$

But $BC \rightarrow A$, now we will check for BC ,
 $(BC)^+ = \{B, C, A, D, E\}$
Therefore, BC is a candidate key.

Properties of functional dependency set:

1) Membership: Let F be an FD set on R we can get $X \rightarrow Y$ using FD in F then, $X \rightarrow Y$ is a member of F .

Example: $R\{ABC\}$ and $FD = \{P \rightarrow Q, Q \rightarrow R\}$

Is $P \rightarrow R$ is member of F ?

Sol: $P^+ = \{P, Q, R\}$

Using closure of P , we can say that

$$A \rightarrow C$$

Therefore, $P \rightarrow R$ is a member of F .

2) Closure of an FD set: It is a set of all functional dependencies that can be determined using Functional dependency in F .

Note:

F^+ includes all trivial and non-trivial dependencies that are possible.



3) Equality of functional dependency set: Two Functional dependency, F and G, is said to be equal iff are

- $F^+ = G^+$
- F covers G, and G covers F (If all FDs in G can be determined by all FD's in F and all FDs in F can be determined by all FDs in G).

SOLVED EXAMPLES

Q5 For R(PQRS), given $F = \{PQ \rightarrow RS, Q \rightarrow R, R \rightarrow S\}$
 $G = \{PQ \rightarrow R, PQ \rightarrow S, R \rightarrow S\}$
Is $F = G$?

Sol: F covers G

$$(PQ)_F^+ = \{P, Q, R, S\}$$

$$(R)_F^+ = \{R, S\}$$

Hence F covers G, but G does not cover F

Hence F covers G, but G does not cover F as the closure of Q using the FD set G is not the same as using the FD set F.

G does not the cover F as

$Q^+ = Q$, but in F, FD : $Q \rightarrow R$ that G does not hold.

$R^+ = RS$

$PQ^+ = PQRS$

Therefore, $F \neq G$



Rack Your Brain



R(A, B, C, D, E) has two FD sets F and G

$$F = \left\{ \begin{array}{l} A \rightarrow B, \quad D \rightarrow AC \\ AB \rightarrow C \quad D \rightarrow E \end{array} \right\} \quad G = \left\{ \begin{array}{l} A \rightarrow BC \\ D \rightarrow AE \end{array} \right\}$$

Check if F covers G

Finding the minimal cover:

Canonical/minimal cover is the reduced form of functional dependency also known as irreducible set.

- It is free from all extraneous FDs.
- The closure of canonical cover is same as that of the given set of functional dependencies.
- Canonical cover is not unique.
- It reduces the computation time.

Steps to find canonical cover:

Given a set of Functional dependencies, F:

1) Start with F.

2) Remove all trivial functional dependencies.

3) Repeated apply until no changes are possible.

- Union simplification
- RHS simplification
- LHS simplification
- 4) Result is a minimal cover.



SOLVED EXAMPLES

Q6 $FD = \{ E \rightarrow G, \\ G \rightarrow S, \\ E \rightarrow S \\ \}$

Sol: Using union rule:



This is the canonical cover obtained from given Functional Dependencies.

OR

There is one more explanatory way to solve this.

- I) Write the given set of FDs such that each FD on LHS has only one attribute.
- II) Check each FD one by one to see if the obtained set of Functional dependencies is essential or not.

Now, there are two cases:

Case I: If the results are the same, it means that FD is non-essential and can be removed.

Case II: If the results are different, it indicates that Functional dependency is necessary, and FD should not be deleted.

- III) Now, check for any FD that contains more than one attribute on its left side. If there exists such FD, then check if their LHS can be reduced with the help of the following steps:



- Compute the closure of possible subsets of LHS of that FD.
- Replace the LHS with the subset if any subset yields the same closure result as the complete left side.

Previous Years' Question



Suppose the following Functional dependencies hold on a relation U with attributes P, Q, R, S, and T:

- $P \rightarrow QR$
- $RS \rightarrow T$

Which of the following Functional dependencies can be inferred from the above Functional dependencies?

- 1) $PS \rightarrow T$ 2) $R \rightarrow T$
3) $P \rightarrow R$ 4) $PS \rightarrow Q$

Sol: 1, 2, 3

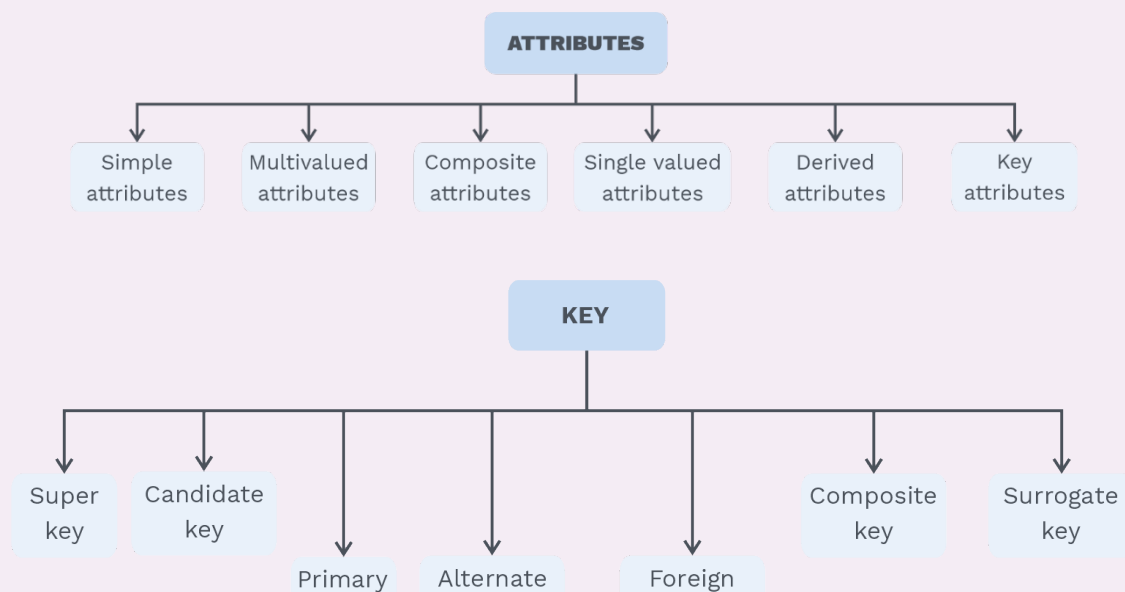
(GATE-CSE Set-2 2021)



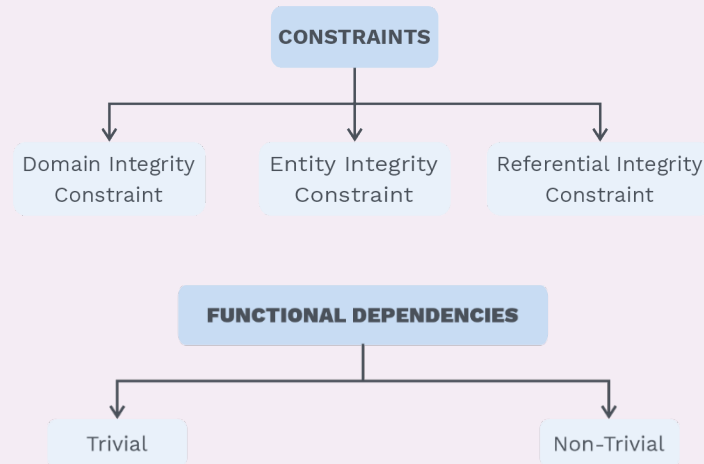
Chapter Summary



- A DBMS is a software that supports the management of a large collection of data.
- DBMS provides the user with data independence, efficient data access, automatic data integrity and security.
- Database design as six steps:
 - a) Requirement analysis
 - b) Conceptual database design
 - c) Logical database design
 - d) Schema refinement
 - e) Physical database design
 - f) Security design



- The relational database modelling represents the database as a collection of relations.
- Properties of relation:
 - a) Name of relation should be distinct
 - b) Tuple should not have duplicate value
 - c) Name of every attribute should be distinct



- **Armstrong's axioms:**

- a) Reflexivity: If $B \subseteq A$, then $A \rightarrow B$
- b) Transitivity: If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
- c) Augmentation: If $A \rightarrow B$ then $AC \rightarrow BC$ where C is an attribute-sets

From the above axioms, we can derive :

- d) Decomposition: IF $A \rightarrow BC$, then $A \rightarrow B$ and $A \rightarrow C$
- e) Composition: IF $A \rightarrow B$ and $C \rightarrow D$, then $AC \rightarrow BD$
- f) Additive: IF $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$

