UNIT-III: Relational Model

- Relational Model was proposed by **E.F. Codd** to model data in the form of relations or tables.
- Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.
- After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model to relational model which can be implemented using any RDBMS languages like Oracle SQL, MySQL etc.

CUSTOMER TABLE

ID	NAME	AGE	ADDRESS	SALARY
1	Akshay	25	Delhi	30000
2	Manish	27	Mumbai	35000
3	Kushagra	26	Kolkata	30000
4	Mukesh	31	Hyderabad	32000
5	Himanshu	29	Chennai	40000
6	Neeraj	30	Noida	36000
7	Nishant	32	Delhi	30000

Figure: Relational Model of the Database

Key Terms:

1. **Tables** – In the Relational model the, relations are saved in the table format.
   - A table has two properties rows and columns.
   - Rows represent records and columns represent attributes.
2. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, Name, Marks,etc.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain. For Example, integer domain of attribute can only take integer values etc.
5. **Relation key** – Every row has one, two or multiple attributes, which is called relation key.
6. **Degree or Arity:** The total number of attributes present in the relation is called the degree of the relation.
   For Example, Arity of relation (Customer) = 5
7. **Cardinality:** Total number of rows or tuples present in the Table.
   For Example, Cardinality of relation (Customer)= 7
8. **Relation Schema (or Database Schema):** A relation schema represents the name of the relation, its attributes and type of attributes.

Table Name: **Customer**

ID	**Number(1)**
Name	**Varchar(15)**
Age	**Number(2)**
Address	**Varchar(50)**
Salary	**Number(10,2)**

9. **Relation instance** – Relation instance is a finite set of tuples at a particular time in the RDBMS system. Relation instances never have duplicate tuples.

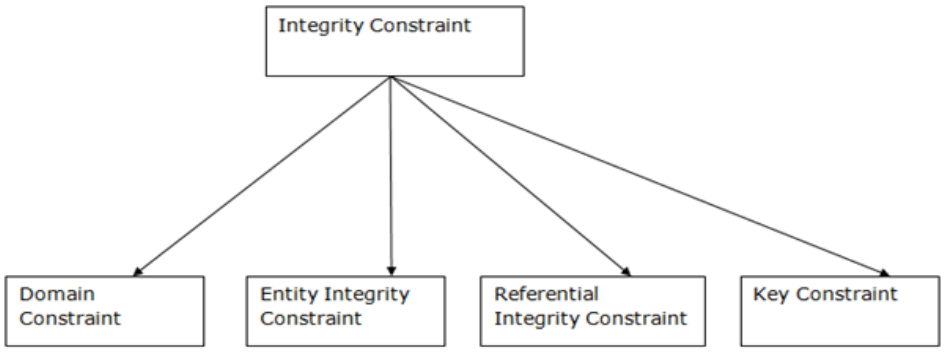
For Example, for the time being Customer table having 7 instances**.**

Properties of Relations:

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Tuple has no duplicate value

Relational Integrity Constraints

- Relational Integrity constraints in DBMS are referred to conditions which must be present for a valid relation.
- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database. Some of the Constraints are as follows:



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

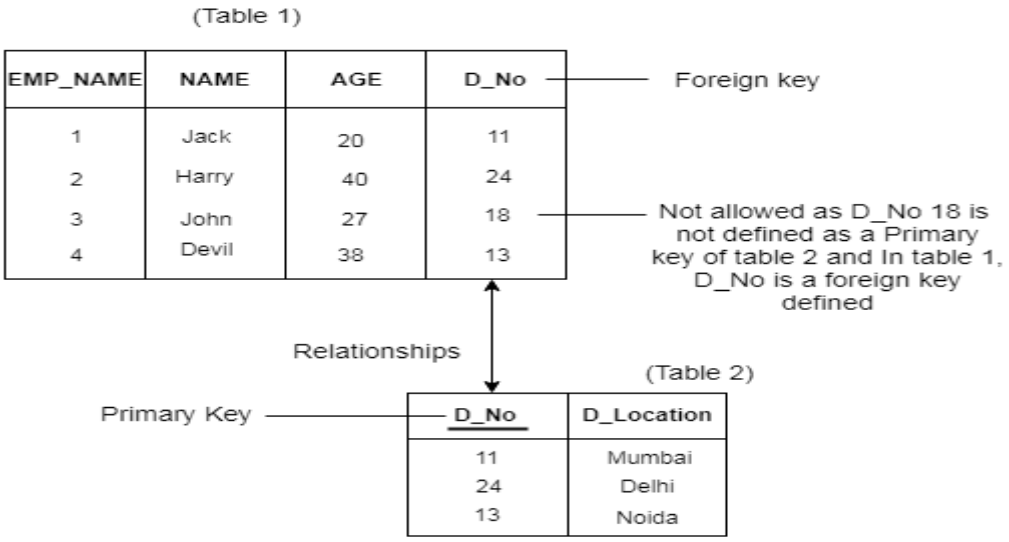
EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.



4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and not null value in the relational table.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Operations in Relational Model

Four basic operations performed on relational database model are

Insert, update, delete and select.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

INSERT

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Update Operation

You can see that in the below-given relation table CustomerName= ‘Apple’ is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

UPDATE

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Delete Operation

To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

In the above-given example, CustomerName= “Apple” is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

Select Operation

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
2	Amazon	Active

In the above-given example, CustomerName=”Amazon” is selected.

Dr Edgar F Codd Rules

Dr E.F.Codd, also known to the world as the ‘Father of Database Management Systems’ had propounded 12 rules which are in-fact 13 in number. The rules are numbered from zero to twelve. His twelve rules are fondly called ‘E.F.Codd’s Twelve Commandments’.

- If a management system or software follows any of 5-6 rules proposed by E.F.Codd, it qualifies to be a Database Management System (DBMS).
- If a management system or software follows any of 7-9 rules proposed by E.F.Codd, it qualifies to be a semi-Relational Database Management System (semi- RDBMS).
- If a management system or software follows 9-12 rules proposed by E.F. Codd, it qualifies to be a complete Relational Database Management System (RDBMS).

Here is brief note on E.F Codd’s twelve rules:

Rule 0 − Foundation rule

The database must be in relational form. So that the system can handle the database through its relational capabilities.

Rule 1: Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: Systematic treatment of NULL

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

Rule 4: Active/Dynamic Online Catalog based on the relational model

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Rule 5: Powerful and Well-Structured Language

One well-structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc. If the database allows access to the data without the use of this language, then that is a violation.

Rule 6: View Updation Rule

All the view that are theoretically updatable should be updatable by the system as well.

Rule 7: Relational Level Operation

There must be Insert, Delete, and Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8: Physical Data Independence Rule

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

Rule 9: Logical Data Independence Rule

It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

Rule 10: Integrity Independence

The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

Rule 11: Distribution Independence Rule

The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

Rule 12: Non-subversion Rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

Functional Dependencies

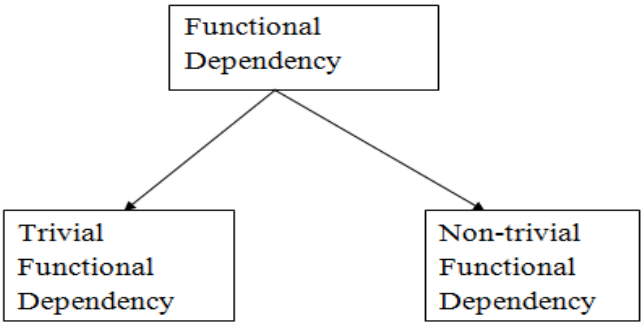
Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system.

- Functional dependency helps you to maintain the quality of data in the database.
- A functional dependency is denoted by an arrow →. The functional dependency of X on Y is represented by X → Y.
- Functional Dependency plays a vital role to find the difference between good and bad database design.

X → Y

The left side of FD is known as a **Determinant/Antecedent**, the right side of the production is known as a **Dependent/Consequent**.

Types of Functional Dependencies:



Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Inference Rule (IR):

- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.
- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.

The Functional dependency has 6 types of inference rules:

1. Reflexive Rule (IR_1)

In the reflexive rule, if **Y is a subset of X, then X determines Y.**

$$\text{i.e., If } X \supseteq Y \text{ then } X \rightarrow Y$$

2. Augmentation Rule (IR_2)

The augmentation is also called as a **partial dependency**. In augmentation, if X determines Y, then XZ determines YZ for any Z.

$$\text{i.e., If } X \rightarrow Y \text{ then } XZ \rightarrow YZ$$

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

$$\text{i.e., If } X \rightarrow Y \text{ and } Y \rightarrow Z \text{ then } X \rightarrow Z$$

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

$$\text{i.e., If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

5. Decomposition Rule (IR_5)

Decomposition rule is also known as **project rule**. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

$$\text{i.e., If } X \rightarrow YZ \text{ then } X \rightarrow Y \text{ and } X \rightarrow Z$$

6. Pseudo transitive Rule (IR_6)

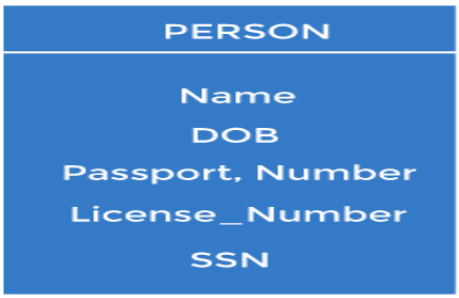
In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

$$\text{i.e., If } X \rightarrow Y \text{ and } YZ \rightarrow W \text{ then } XZ \rightarrow W$$

Keys in Relational Database Management system

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the **Student table** because it is unique for each student. In the **PERSON table**, passport_number, license_number and SSN are keys since they are uniquefor each person.



1. Super Key

Super key is an attribute or set of attributes that has to identify or derive all the attributes of a Relation.

Table: Employee

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

The above table has following super keys. All of the following sets of super key are able to uniquely identify a row of the employee table.

I. {Emp_SSN}
II. {Emp_Number}
III. {Emp_SSN, Emp_Number}
IV. {Emp_SSN, Emp_Name}
V. {Emp_SSN, Emp_Number, Emp_Name}
VI. {Emp_Number, Emp_Name}

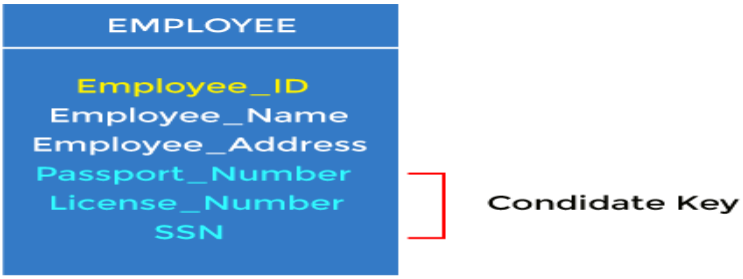
2. Candidate key

A candidate key is an attribute or set of attributes if and only if:

i) It has to derive all the attributes of the relation (or it has to identify a unique record / Tuple in a relation) and
ii) It should be a Minimal Subset of super keys.

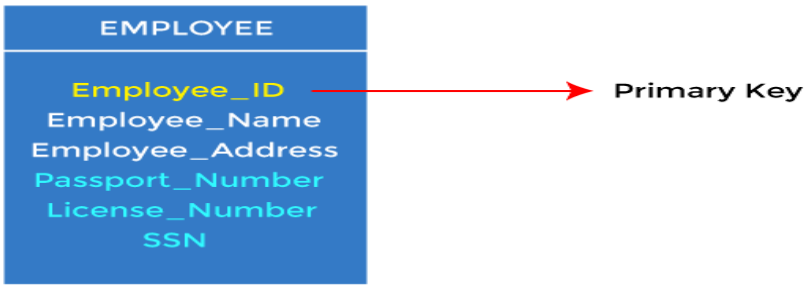
- Except for the primary key, the remaining attributes are considered a candidate key.
- The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, Employee_ID is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Primary Key:

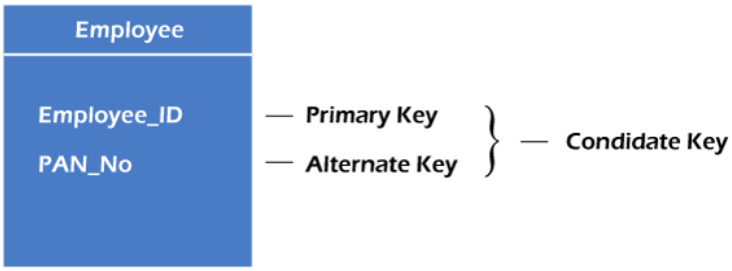
- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).
- There is no significance between Candidate and Primary key, but the primary key selection is based on requirements and developers.



4. Alternate key

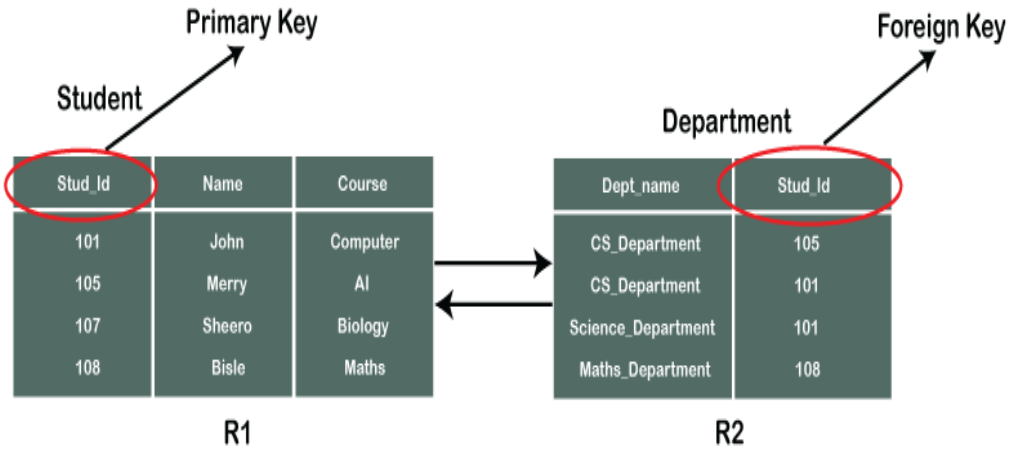
The total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.



5. Foreign Key:

- The FOREIGN KEY constraint is used to maintain Parent-Child relationship i.e., to maintain Referential Integrity.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the **child table**, and the table with the primary key is called the **referenced or parent table.**
- The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.



Prime (or) Key and Non-Key (or) Non-Prime Attributes

Prime attribute − An attribute, which is a part of the Candidate-key, is known as a prime attribute.
Non-prime attribute − An attribute, which is not a part of the Candidate-key, is said to be a non-prime attribute.

Example:
R (A,B,C,D,E,F) {
C→F
E→A
EC→D
A→B
}
Closure: $(EC)^+$=ECDAFB
Candidate Key: EC
Prime attributes: {E,C}
Non-prime attributes: {A,B,D,F}

Normalization

Anomalies in DBMS
There are three types of anomalies that occur when the database is not normalized.

These are: Insertion, updation and deletion anomaly.

Let's take an example to understand this.
Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.
Updation anomaly: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.
Insertion anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.
Deletion anomaly: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of

employee Maggie since she is assigned only to this department.

To overcome these anomalies, we need to normalize the data. In the next section we will discuss about normalization.

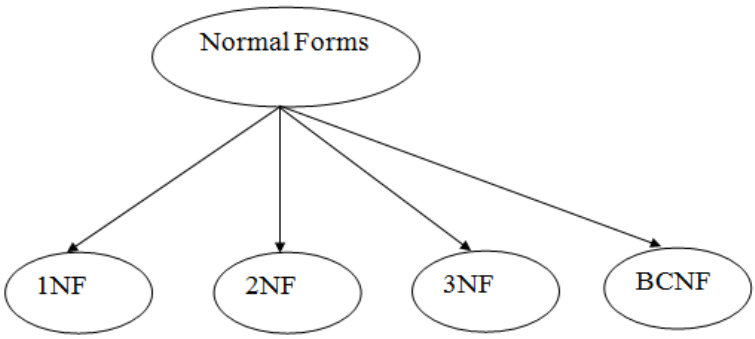
- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- **Normal form** indicates the amount of redundancy present in a relation and is used to eliminate or reduce redundancy in database tables.

Steps to find the highest normal form of a relation:

1. Find all possible candidate keys of the relation.
2. Divide all attributes into two categories: prime attributes and non-prime attributes.
3. Check for 1^{st} normal form then 2^{nd} and so on. If it fails to satisfy n^{th} normal form condition, highest normal form will be n-1.

Types of Normal Forms

There are the four types of normal forms:



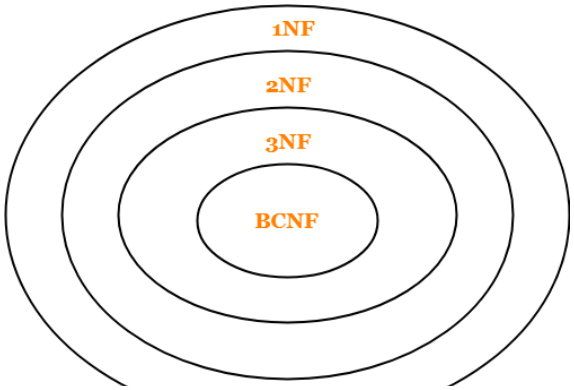


Figure: **Venn Diagram Representation of Normal Forms**

First Normal Form (1NF)

- A relation will be 1NF if it contains an **atomic values** i.e., it should not contain multi-valued attributes.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

  Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

  EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

  The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Second Normal Form (2NF)

A relation is in Second Normal Form (2NF):

i. relation must be in 1NF and
ii. It should not contain Partial-Dependency (P.D),

i.e., Non-Key attributes depending on the only the Part of the Candidate key is called **Partial Dependency.**

Third Normal Form (3NF)

A relation is in Third Normal Form (3NF):

i. if it is in 2NF, and
ii. It should not have Transitive Dependency.

i.e., Non-Key attributes depending on the Non-Key attribute or Non-Key attribute depending on Non-Key attribute along with Part of the Candidate Key is called **Transitive Dependency**.

Boyce Codd normal form (BCNF)

A relation is in BCNF:

i. if it is in 3NF, and
ii. All the FD's are in Full-Dependency

i.e., LHS of FD is depends on either a super key or Candidate Key.

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- If a relation is in BCNF means it always has Zero redundancy.
- Every Binary Relation always in BCNF

Fourth Normal Form (4NF)

A relation R is in 4NF if and only if the following conditions are satisfied:

i. It should be in the Boyce-Codd Normal Form (BCNF) and
ii. The table should not have any Multi-valued Dependency.

i.e., if we consider A→B, A→C here **A** multi-determining **B** and **C**. So, 4NF involves removal of **Multi-Valued Dependencies.**

Fifth Normal Form (5NF)

- The 5NF (Fifth Normal Form) is also known as **project-join normal form.**
- A relation is in Fifth Normal Form (5NF):
  i. if it is in 4NF, and
  ii. It should not have lossless decomposition into smaller tables.

Relational Algebra

- Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances.
- Query Language is a technique of accessing data from the database. It's mainly of two types :
  1. **Procedural Query Language:** This is a formal way of accessing the database. The information regarding what has to be accessed and how it has to be accessed are provided along with the queries, so that the data can be accessed from the database.

     Examples: FORTRAN, COBOL, ALGOL, BASIC, C and Pascal.

     Relational Algebra (RA) is a type of Procedural Query Language. It consists of a set of operators which take one or two relations as input and a new relation is provided as the output.

     - RA is the component of relational query engine. RA also provides a framework for query optimization.
     - SQL queries are internally translated into RA operations first.
     - Relational Algebra eliminates duplicate tuples from the result by default.

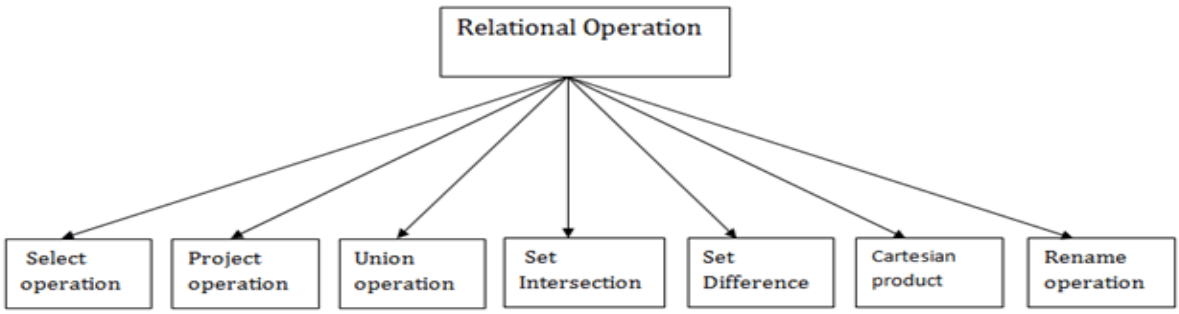
  2. **Non Procedural Query Language**: This is an informal way of accessing the data from the database.
     - In the case of Non Procedural Query Language, information is required only about what data has to be accessed from the database**.**
     - **SQL** and **Tuple Relational Calculus** are types of Non Procedural Query Language.

     Examples: SQL, PROLOG, LISP.

Procedural Language	Non-Procedural Language
It is command-driven language.	It is a function-driven language
It works through the state of machine.	It works through the mathematical functions.
Its semantics are quite tough.	Its semantics are very simple.
It returns only restricted data types and allowed values.	It can return any data type or value
Overall efficiency is very high.	Overall efficiency is low as compared to Procedural Language.
Size of the program written in Procedural language is large.	Size of the Non-Procedural language programs are small.
It is not suitable for time critical applications.	It is suitable for time critical applications.
Iterative loops and Recursive calls both are used in the Procedural languages.	Recursive calls are used in Non-Procedural languages.

Types of Relational Algebra operations

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.



Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation: $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like:**=, ≠, ≥, < , >, ≤.**

For example −

$\sigma_{subject = "database"}(Books)$

Output: Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output: Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output: Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation: $\prod_{A1, A2, An}(r)$

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated in a relation.

For example :

$$\prod_{subject,\ author}(Books)$$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as:

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notation: r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\prod_{author}(Books) \cup \prod_{author}(Articles)$$

Output: Projects the names of the authors who have either written a book or an article or both.

Intersection Operation (∪)

It performs binary union between two given relations and is defined as:

$$r \cap s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notation: r ∩ s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\prod_{author}(Books) \cap \prod_{author}(Articles)$$

Output: Projects the names of the authors who have written a book and an article both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation: r − s

Finds all the tuples that are present in **r** but not in **s**.

$$\prod_{author}(Books) - \prod_{author}(Articles)$$

Output: Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation: r X s

Where **r** and **s** are relations and their output will be defined as:

r X s = { q t | q ∈ r and t ∈ s}

$$\sigma_{author = 'tutorialspoint'}(Books \ X \ Articles)$$

Output: Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The rename operation allows us to rename the output relation. 'Rename' operation is denoted with small Greek letter **rho*ρ***.

Notation: ρ_x (E)

Where the result of expression **E** is saved with name of **x**.

Advantages and Limitations of RA

Advantages:

- Relational algebra is based on the set theory which is a mathematical concept due to which it has a scope of development.

- Like mathematics there can be many expressions for the same operation, in a similar way if there are two relational algebraic expressions for the same operation then the query optimizer will switch to the most efficient query.
- It is a high-level query language.

Limitations:

- Relational algebra cannot perform arithmetic operations.
- It is unable to do aggregation operations.
- Also, Transitive closure of a binary relation cannot be expressed.
- It cannot modify the data present in the database.

Relational Calculus

Before understanding Relational calculus in DBMS, we need to understand **Procedural Language** and **Declarative Language**.

1. **Procedural Language** - Those Languages which clearly define how to get the required results from the Database are called Procedural Language. **Relational algebra** is a Procedural Language.
2. **Declarative Language** - Those Language that only cares about what to get from the database without getting into how to get the results are called Declarative Language. **Relational Calculus** is a Declarative Language.

So Relational Calculus is a Declarative Language that uses Predicate Logic or First-Order Logic to determine the results from Database.

Relational Calculus is of Two Types:

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

Tuple Relational Calculus (TRC)

- Tuple Relational Calculus in DBMS uses a tuple variable (t) that goes to each row of the table and checks if the predicate is true or false for the given row.
- Depending on the given predicate condition, it returns the row or part of the row.

Syntax: `{ T | P(T)}`

Where **T** is the tuple variable that runs over every Row, and **P(T)** is the predicate logic expression or condition.

For Example: Customer Table

Customer_id	Name	Zip_code
1	Rohit	12345
2	Rahul	13245
3	Rohit	56789
4	Amit	12345

Q:Write a TRC query to get all the data of customers whose zip code is 12345.

TRC Query:{T | T∈ Customer ∧T.Zipcode = 12345}or

TRC Query:{T| Customer (T) ∧ T [Zipcode] = 12345 }

Workflow of query - The tuple variable "T" will go through every tuple of the Customer table. Each row will check whether the Cust_Zipcode is 12345 or not and only return those rows that satisfies the Predicate expression condition.

The TRC expression above can be read as **"Return all the tuple which belongs to the Customer Table and whose Zipcode is equal to 12345."**

Result of the TRC expression above:

Customer_id	Name	Zip code
1	Rohit	12345
4.	Amit	12345

Domain Relational Calculus (DRC)

Domain Relational Calculus uses domain Variables to get the column values required from the database based on the predicate expression or condition.

The Domain realtional calculus expression syntax:

{< X1, X2, X3,…..,Xn> | P (X1,X2,X3,….., Xn) }

where,

< X1, X2, X3,…..,Xn>are domain variables used to get the column values required, and **P(X1, X2, X3,…..,Xn)** is predicate expression or condition.

Q:Write a DRC query to get the data of all customers with Zip code 12345.

DRC query:{<x1,x2,x3>| <x1,x2>∈ Customer ∧ x3 = 12345 }

Workflow of Query: In the above query x1, x2, x3 (ordered) refers to the attribute or column which we need in the result, and the predicate condition is that the first two domain variables x1 and x2 should be present while matching the condition for each row and the third domain variable x3 should be equal to 12345.

Result of the DRC query will be:

Customer_id	Name	Zip code
1	Rohit	12345
4	Amit	12345

Comparison between Tuple Relational Calculus (TRC) and Domain Relational Calculus (DRC):

Tuple Relational Calculus (TRC)	Domain Relational Calculus (DRC)
In TRS, the variables represent the tuples from specified relation.	In DRS, the variables represent the value drawn from specified domain.
A tuple is a single element of relation. In database term, it is a row.	A domain is equivalent to column data type and any constraints on value of data.
In this, filtering variable uses tuple of relation.	In this, filtering is done based on the domain of attributes.
Query cannot be expressed using a membership condition.	Query can be expressed using a membership condition.
The QUEL or Query Language is a query language related to it,	The QBE or Query-By-Example is query language related to it.
Notation : {T | P (T)} or {T | Condition (T)}	Notation : { a1, a2, a3, …, an | P (a1, a2, a3, …, an)}

Summary about TRC and DRC:

- Relational Calculus in DBMS tells us what we want from the database and not how to get that.
- Relational Calculus is a Declarative Language.
- TRC uses tuple variable and checks every Row with the Predicate expression condition.
- DRC uses domain variables and returns the required attribute or column based on the condition.
- For any requirement both, TRC and DRC can be written.

References

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan,“Database System Concepts”, Seventh Edition.
2. Jeffrey A. Hoffer, V. Ramesh, HeikkiTopi, “Modern Database Management”, Tenth Edition.
3. Raghu Ramakrishnan, “Database Management Systems”.