Unit-2

Relational model, Algebra and calculus

The relational model:

RELATIONAL MODEL

Relational model is simple model is simple model in which database is represented as a collection of "relations" where each relation is represented by two-dimensional table.

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

The relational model was founded by E.F.Codd of the IBM in 1972.The basic concept in the relational model is that of a relation.

Properties:

- It is column homogeneous. In other words, in any given column of a table, all items are of the same kind.
- Each item is a simple number or a character string. That is a table must be in first normal form.
- All rows of a table are distinct.
- The ordering of rows with in a table is immaterial

The column of a table is assigned distinct names and the ordering of these columns in immaterial.

Domain, attributes tuples and relational:

Tuple:

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

Attributes:

The name of each column in a table is used to interpret its meaning and is called anattribute.Each table is called a relation.
In the above table, account_number, branch name, balance are the attributes.

Domain:

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

Relation:

A relation consist of

- **Relational schema**
- **Relation instance**

Relational schema:

A relational schema specifies the relation' name, its attributes and the domain of eachattribute. If R is the name of a relation and A1,A2,… and is a list of attributes representing R then R(A1,A2,…,an) is called a relational schema. Each attribute in this relational schema takes a value from some specific domain called domain(Ai).

Example:

PERSON(PERSON_IDinteger,NAME:
STRING,AGE:INTEGER,ADDRESS:string)
Total number of attributes in a relation denotes the degree of a relation.since the PERSON relation schemea contains four attributes ,so this relation is of degree 4.

Relation Instance:

A relational instance denoted as r is a collection of tuples for a given relationalschema at a specific point of time.
A relation state r to the relations schema R(A1,A2…,An) also denoted by r® is a setof n-tuples
R{t1,t2,…tm}

Where each n-tuple is an
ordered list of n values
T=<v1,v2,….vn>
Where each vi belongs to domain (Ai) or contains null values.
The relation schema is also called 'intension' and the relation state is also called'extension'.
Eg:

Relation schema for student:

STUDENT(rollno:strinhg,name:string,city:string,age:integer

Relation instance:
Student:

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

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

For example , customer-id,(cname,customer-id),(cname,telno)
Candidate key:
Super keys of a relation can contain extra attributes . candidate keys are minimal super keys. 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.

In a relation R, a candidate key for R is a subset of the set of attributes of R, which have the following properties:

- *Uniqueness:* no two distinct tuples in R have the same values for the candidate key
- *Irreducible:* No proper subset of the candidate key has the *uniqueness property that is the candidate key.*
- *A candidate key's values must exist. It can't be null.*
- *The values of a candidate key must be stable. Its value can not change outside the control of the system.*

Eg: (cname,telno)

Primary key:

The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any are called ***alternate key*.**

RELATIONAL CONSTRAINTS:

There are three types of constraints on relational database that include

- DOMAIN CONSTRAINTS
- KEY CONSTRAINTS
- INTEGRITY CONSTRAINTS

DOMAIN CONSTRAINTS:

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

Standard numeric data types for integer

- Real numbers
- Characters
- Fixed length strings and variable length strings

Thus, domain constraints specifies the condition that we 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.

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

Key constraints:

This constraints states that the key attribute value in each tuple msut be unique .i.e, no two tuples contain the same value for the key attribute.(null values can allowed)

Emp(empcode,name,address) . here empcode can be unique

Integrity constraints:

There are two types of integrity constraints:

- Entity integrity constraints
- Referential integrity constraints

Entity integrity constraints:

It states that no primary key value can be null and unique. 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.

Referential integrity constraints:

It states that the tuple in one relation that refers to another relation must refer to an existing tuple in that relation. This constraints is specified on two relations .
If a column is declared as foreign key that must be primary key of another table.

Department(deptcode,dname) Here the deptcode is the primary key.
Emp(*empcode*,name,city,*deptcode*). Here the deptcode is foreign key.

CODD'S RULES

Rule 1 : The information Rule.
"All information in a relational data base is represented explicitly at the logical level and in exactly one way - by values in tables."
Everything within the database exists in tables and is accessed via table access routines.
Rule 2 : Guaranteed access Rule.
"Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name."
To access any data-item you specify which column within which table it exists, there is no reading of characters 10 to 20 of a 255 byte string.
Rule 3 : Systematic treatment of null values.
"Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."
If data does not exist or does not apply then a value of NULL is applied, this is understood by the RDBMS as meaning non-applicable data.
Rule 4 : Dynamic on-line catalog based on the relational model.
"The data base description is represented at the logical level in the same way as-ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data."
The Data Dictionary is held within the RDBMS, thus there is no-need for off-line volumes to tell you the structure of the database.
Rule 5 : Comprehensive data sub-language Rule.
"A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all the following items

- Data Definition
- View Definition
- Data Manipulation (Interactive and by program).
- Integrity Constraints
- Authorization.

Every RDBMS should provide a language to allow the user to query the contents of the RDBMS and also manipulate the contents of the RDBMS.
Rule 6 : .View updating Rule
"All views that are theoretically updateable are also updateable by the system."
Not only can the user modify data, but so can the RDBMS when the user is not logged-in.
Rule 7 : High-level insert, update and delete.
"The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data."
The user should be able to modify several tables by modifying the view to which they act as base tables.
Rule 8 : Physical data independence.
"Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods."
The user should not be aware of where or upon which media data-files are stored
Rule 9 : Logical data independence.
"Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un-impairment are made to the base tables."
User programs and the user should not be aware of any changes to the structure of the tables (such as the addition of extra columns).
Rule 10 : Integrity independence.
"Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs."
If a column only accepts certain values, then it is the RDBMS which enforces these constraints and not the user program, this means that an invalid value can never be entered into this column, whilst if the constraints were enforced via programs there is always a chance that a buggy program might allow incorrect values into the system.
Rule 11 : Distribution independence.
"A relational DBMS has distribution independence."
The RDBMS may spread across more than one system and across several networks, however to the end-user the tables should appear no different to those that are local.
Rule 12 : Non-subversion Rule.
"If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity Rules and constraints expressed in the higher level relational language (multiple-records-at-a-time)."

Enforcing Integrity Constraints in DBMS

Introduction

Integrity constraints are rules that specify the conditions that must be met for the data in a database to be considered valid. These constraints help to ensure the accuracy and consistency of the data by limiting the values that can be entered for a particular attribute and specifying the relationships between entities in the database.

There are several types of integrity constraints that can be enforced in a DBMS:

- **Domain constraints:** These constraints specify the values that can be assigned to an attribute in a database. For example, a domain constraint might specify that the values for an "age" attribute must be integers between 0 and 120.
- **Participation constraints:** These constraints specify the relationship between entities in a database. For example, a participation constraint might specify that every employee must be assigned to a department.
- **Entity integrity constraints**: These constraints specify rules for the primary key of an entity. For example, an entity integrity constraint might specify that the primary key cannot be null.
- **Referential integrity constraints:** These constraints specify rules for foreign keys in a database. For example, a referential integrity constraint might specify that a foreign key value must match the value of the primary key in another table.
- **User-defined constraints:** These constraints are defined by the database administrator and can be used to specify custom rules for the data in a database.

Ways to Enforce Integrity Constraints

There are several ways to enforce integrity constraints in a DBMS:

- **Declarative referential integrity:** This method involves specifying the integrity constraints at the time of database design and allowing the DBMS to enforce them automatically.
- **Triggers:** A trigger is a special type of stored procedure that is executed automatically by the DBMS when certain events occur (such as inserting, updating, or deleting data). Triggers can be used to enforce integrity constraints by checking for and rejecting invalid data.
- **Stored procedures:** A stored procedure is a pre-defined set of SQL statements that can be executed as a single unit. Stored procedures can be used to enforce integrity constraints by performing checks on the data before it is inserted, updated, or deleted.
- **Application-level code:** Integrity constraints can also be enforced at the application level by writing code to check for and reject invalid data before it is entered into the database.

It is important to carefully consider the appropriate method for enforcing integrity constraints in a DBMS in order to ensure the accuracy and consistency of the data.

Querying relational data:

A relational database query is a question about the data, and the answer consists of a new relation containing the result. For example, we might want to find all students AGE less than 18 or all students enrolled in perticular course.

The ***SELECT*** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax in Mysql

SELECT column1, column2, ...

FROM table_name;

If you want to select all the fields available in the table, use the following syntax:

Syntax in Mysql

*SELECT * FROM table_name;*

The symbol ´*****´ means that we retain all fields of selected tuples in the result.

We can retrieve rows corresponding to students who are younger than 18 withthe following SQL query:

Example:

SELECT * FROM Students WHERE age < 18;

The condition **age < 18** in the WHERE clause specifies that we want to select only tuples in which the age field has a value less than 18.

In addition to selecting a subset of tuples, a query can extract a subset of the fields of each selected tuple. we can compute the student_id and First_name of students who are younger than 18 with the following query:

Example:

SELECT ID,FirstName FROM Students WHERE age < 18;

SQL Aliases

Aliases are the temporary names given to tables or columns. An alias is created with the **AS** keyword.

Alias Column Syntax in Mysql

SELECT column_name AS alias_name

FROM table_name;

Alias Table Syntax in Mysql

SELECT column_name(s)

FROM table_name AS alias_name;

Example:

SELECT studentID AS ID,

FROM students AS S;

Aliases can be useful when:

- There are more than one table involved in a query

- Functions are used in the query
- Column names are big or not very readable
- Two or more columns are combined togeth

SELECT data from Multiple Tables

We can also combine information from multiple tables.

Syntax in Mysql

SELECT table1.column1, table2.column2

FROM table1, table2

WHERE table1.column1 = table2.column1;

Example:

SELECT S.name, E.cid

FROM Students AS S, Enrolled AS E

WHERE S.sid = E.sid;

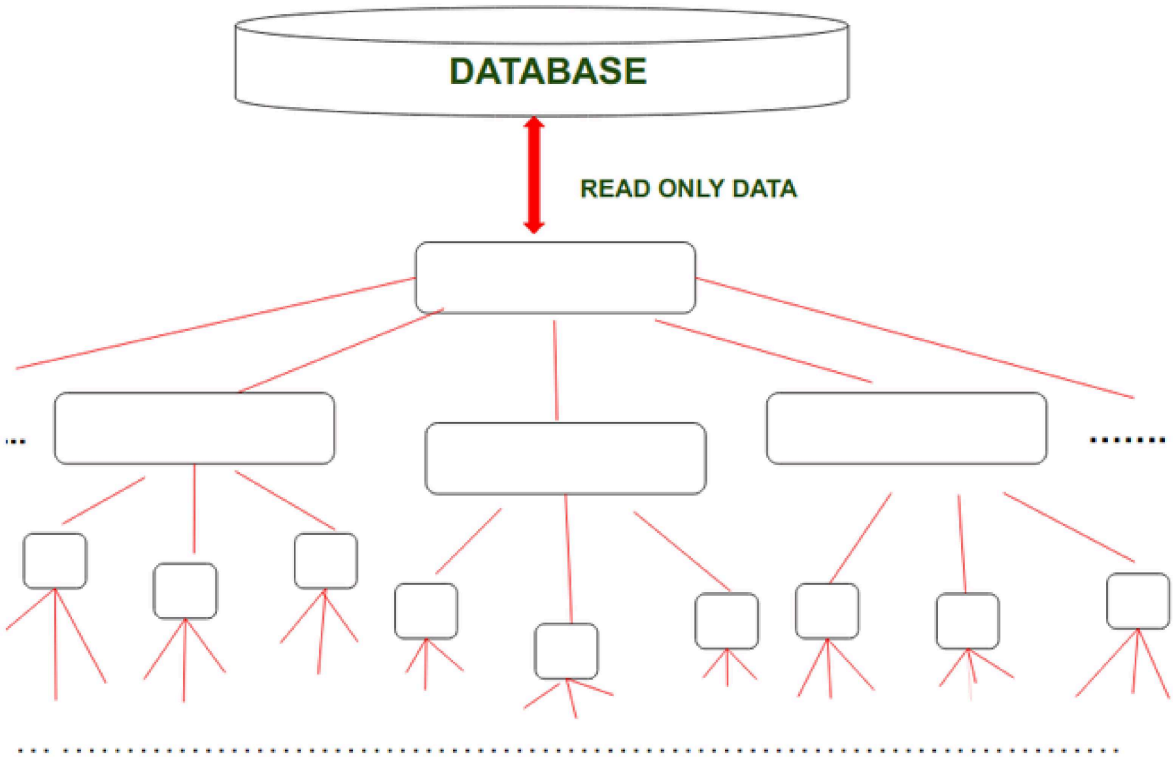
Logical database design:

A Logical Database is a special type of ABAP (Advance Business Application and Programming) that is used to retrieve data from various tables and the data is interrelated to each other. Also, a logical database provides a read-only view of Data.

Structure Of Logical Database:

A Logical database uses only a hierarchical structure of tables i.e. Data is organized in a Tree-like Structure and the data is stored as records that are connected to each other through edges (Links). Logical

Database contains Open SQL statements which are used to read data from the database. The logical database reads the program, stores them in the program if required, and passes them line by line to the application program.



Structure of Logical database

Features of Logical Database:

In this section, let us look at some features of a logical database:

- We can select only that type of Data that we need.
- Data Authentication is done in order to maintain security.

- Logical Database uses hierarchical Structure due to this data integrity is maintained.

Goal Of Logical Database:

The goal of Logical Database is to create well-structured tables that reflect the need of the user. The tables of the Logical database store data in a non-redundant manner and foreign keys will be used in tables so that relationships among tables and entities will be supported.

Tasks Of Logical Database:

Below is some important task of Logical Database:

- With the help of the Logical database, we will read the same data from multiple programs.
- A logical database defines the same user interface for multiple programs.
- Logical Database ensures the Authorization checks for the centralized sensitive database.
- With the help of a Logical Database, Performance is improved. Like in Logical Database we will use joins instead of multiple SELECT statements, which will improve response time and this will increase the Performance of Logical Database.

Data View Of Logical Database:

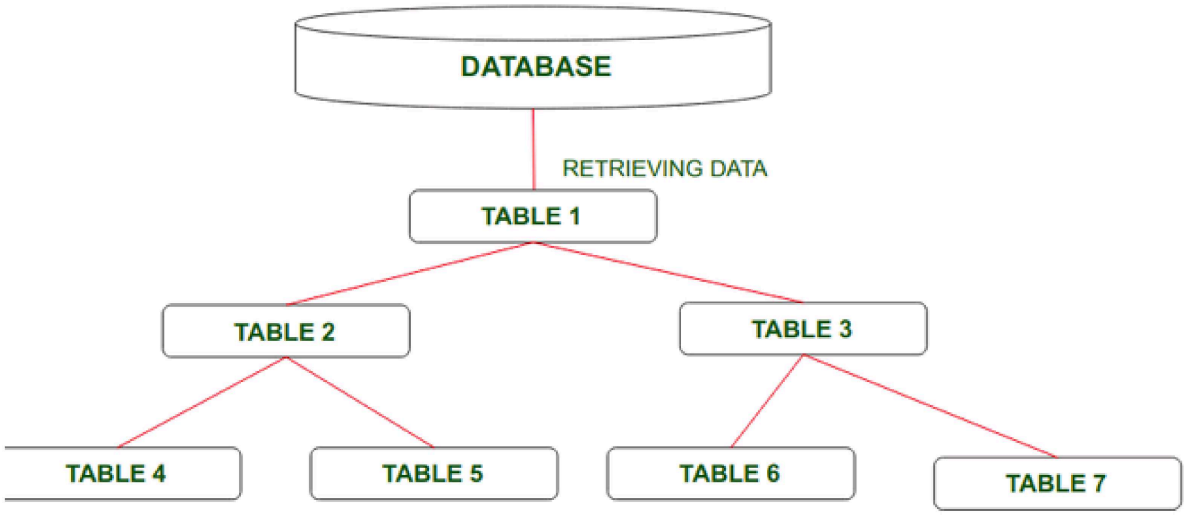
Logical Database provides a particular view of Logical Database tables. A logical database is appropriately used when the structure of the Database is Large. It is convenient to use flow i.e

- SELECT

- READ
- PROCESS
- DISPLAY

In order to work with databases efficiently. The data of the Logical Database is hierarchical in nature. The tables are linked to each other in a Foreign Key relationship.

Diagrammatically, the Data View of Logical Database is shown as:

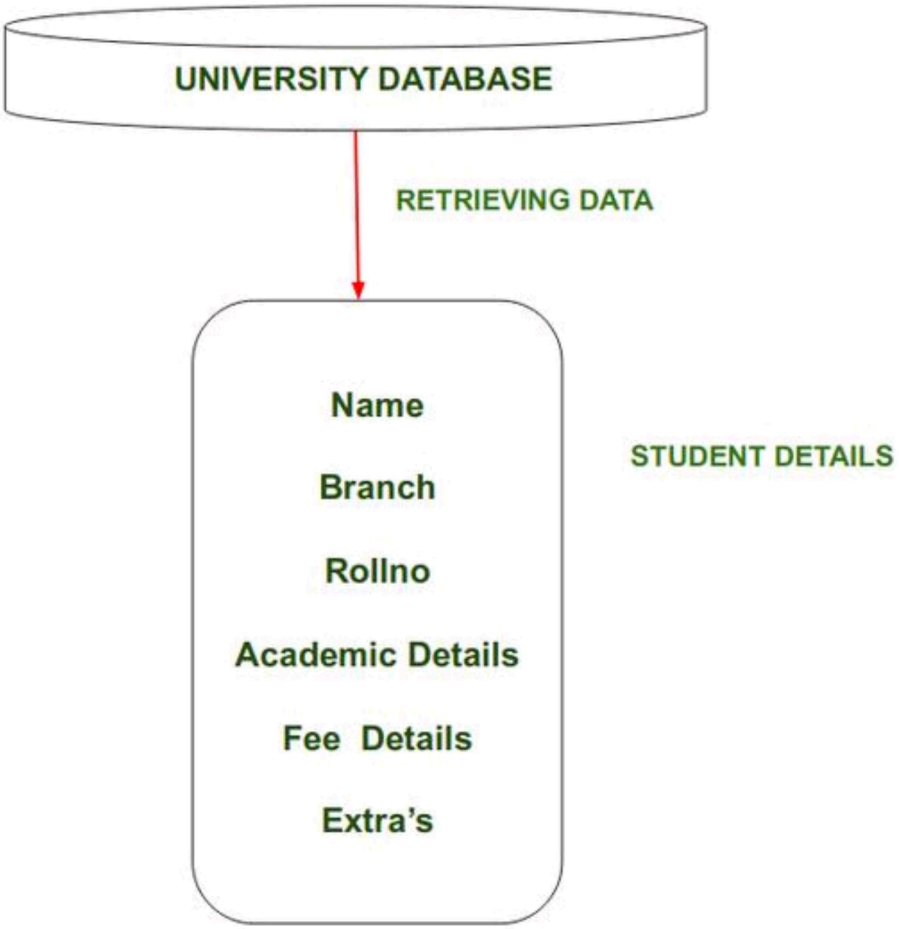


Points To Remember:

- Tables must have Foreign Key Relationship.

- A logical Database consists of logically related tables that are arranged in a hierarchical manner used for reading or retrieving Data.
- Logical Database consist of three main elements:
    - Structure of Database
    - Selections of Data from Database
    - Database Program
- If we want to improve the access time on data, then we use VIEWS in Logical Database.

Example:
Suppose in a University or College, a HOD wants to get information about a specific student. So for that, he firstly retrieves the data about its batch and Branch from a large amount of Data, and he will easily get information about the required Student but didn’t alter the information about it.



Advantages Of Logical Database:

Let us look at some advantages of the logical database:

- In a Logical database, we can select meaningful data from a large amount of data.
- Logical Database consists of Central Authorization which checks for Database Accesses is Authenticated or not.
- In this Coding, the part is less required to retrieve data from the database as compared to Other Databases.

- Access performance of reading data from the hierarchical structure of the Database is good.
- Easy to understand user interfaces.
- Logical Database firstly check functions which further check that user input is complete, correct, and plausible.

Disadvantages Of Logical Database:

This section shows the disadvantages of the logical database:

- Logical Database takes more time when the required data is at the last because if that table which is required at the lowest level then firstly all upper-level tables should be read which takes more time and this slows down the performance.
- In Logical Database ENDGET command doesn't exist due to this the code block associated with an event ends with the next event statement.

Views:

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Advantages of View:

1. **Complexity:** Views help to reduce the complexity. Different views can be created on the same base table for different users.
2. **Security:** It increases the security by excluding the sensitive information from the view.
3. **Query Simplicity:** It helps to simplify commands from the user. A view can draw data from several different tables and present it as a single table.
4. **Consistency:** A view can present a consistent, unchanged image of the structure of the database. Views can be used to rename the columns without affecting the base table.
5. **Data Integrity:** If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets the specified integrity constraints.
6. **Storage Capacity:** Views take very little space to store the data.
7. **Logical Data Independence:** View can make the application and database tables to a certain extent independent.

   Disadvantages of View:

   The DML statements which can be performed on a view created using single base table have certain restrictions are:

1. You cannot INSERT if the base table has any not null column that do not appear in view.
2. You cannot INSERT or UPDATE if any of the column referenced in the INSERT or UPDATE contains group functions or columns defined by expression.

3. You can't execute INSERT, UPDATE, DELETE statements on a view if with read only option is enabled.
4. You can't be created view on temporary tables.
5. You cannot INSERT, UPDATE, DELETE if the view contains group functions GROUP BY, DISTINCT or a reference to a psuedocolumn rownum.
6. You can't pass parameters to the SQL server views.
7. You can't associate rules and defaults with views.

Sample table:

Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19

2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

1. **CREATE VIEW** view_name **AS**
2. **SELECT** column1, column2.....
3. **FROM** table_name
4. **WHERE** condition;

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.

Query:

1. **CREATE VIEW** DetailsView **AS**
2. **SELECT NAME**, ADDRESS
3. **FROM** Student_Details

4. **WHERE** STU_ID < 4;

Just like table query, we can query the view to view the data.

1. **SELECT** * **FROM** DetailsView;

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:

1. **CREATE VIEW** MarksView **AS**
2. **SELECT** Student_Detail.**NAME**, Student_Detail.ADDRESS, Student_Marks.MARKS
3. **FROM** Student_Detail, Student_Mark
4. **WHERE** Student_Detail.**NAME** = Student_Marks.**NAME**;

To display data of View MarksView:

1. **SELECT** * **FROM** MarksView;

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax

1. **DROP VIEW** view_name;

Example:

If we want to delete the View **MarksView**, we can do this as:

1. **DROP VIEW** MarksView;

Significance of Views:

Views are highly significant, as they can provide advantages over tasks. Views can represent a subset of data contained in a table. Consequently they can limit the degree of exposure of the underlying base table to the outer world. They are used for security purpose in database and act as an intermediate between

real table schemas and programmability. They act as aggregate tables.

Types of Views:

There are two types of views.

1. **Join View:** A join view is a view that has more than one table or view in its from clause and it does not use any Group by Clause, Rownum, Distinct and set operation.
2. **Inline View:** An inline view is a view which is created by replacing a subquery in the from clause which defines the data source that can be referenced in the main query. The sub query must be given an alias for efficient working.

SQL ALTER TABLE:

The ALTER TABLE statement in Structured Query Language allows you to add, modify, and delete columns of an existing table. This statement also allows database users to add and remove various SQL constraints on the existing tables.

Any user can also change the name of the table using this statement.

ALTER TABLE ADD Column statement in SQL

In many situations, you may require to add the columns in the existing table. Instead of creating a whole table or database again

you can easily add single and multiple columns using the ADD keyword.

Syntax of ALTER TABLE ADD Column statement in SQL

1. ALTER TABLE table_name ADD column_name column-definition;

The above syntax only allows you to add a single column to the existing table. If you want to add more than one column to the table in a single SQL statement, then use the following syntax:

1. ALTER TABLE table_name
2. ADD (column_Name1 column-definition,
3. column_Name2 column-definition,

---

4. column_NameN column-definition);

Examples of ALTER TABLE ADD Column statement in SQL

Here, we have taken the following two different SQL examples, which will help you how to add the single and multiple columns in the existing table using ALTER TABLE statement:

Example 1: Let's take an example of a table named **Cars:**

Car	Color	rate
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

Table: Cars

- Suppose, you want to add the new column Car_Model in the above table. For this, you have to type the following query in the SQL:

1. ALTER TABLE Cars ADD Car_Model Varchar(20);

   This statement will add the Car_Model column to the Cars table.

Example 2: Let's take an example of a table named **Employee:**

Emp_Id	Emp_Name	Emp_Salary	Emp_City
201	Abhay	25000	Goa

202	Ankit	45000	Delhi
203	Bheem	30000	Goa
204	Ram	29000	Goa
205	Sumit	40000	Delhi

Table: Employee

- Suppose, you want to add two columns, **Emp_ContactNo.** and **Emp_EmailID,** in the above Employee table. For this, you have to type the following query in the SQL:

1. ALTER TABLE Employee ADD (Emp_ContactNo. Number(13), Emp _EmailID varchar(50) ;

   This statement will add Emp_ContactNo. and Emp_EmailID columns to the Employee table.

   ALTER TABLE MODIFY Column statement in SQL

   The MODIFY keyword is used for changing the column definition of the existing table.

   Syntax of ALTER TABLE MODIFY Column statement in SQL

1. ALTER TABLE table_name MODIFY column_name column-definition;

   This syntax only allows you to modify a single column of the existing table. If you want to modify more than one column of the table in a single SQL statement, then use the following syntax:

1. ALTER TABLE table_name
2. MODIFY (column_Name1 column-definition,
3. column_Name2 column-definition,
4. .....
5. column_NameN column-definition);

Examples of ALTER TABLE MODIFY Column statement in SQL

Here, we have taken the following two different SQL examples, which will help you how to modify single and multiple columns of the existing table using ALTER TABLE statement:

Example 1: Let's take an example of a table named **Cars:**

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000

Swift Dezire	Red	7,95,000

Table: Cars

- Suppose, you want to modify the datatype of the **Car_Color** column of the above table. For this, you have to type the following query in the SQL:

```
1. ALTER TABLE Cars ADD Car_Color Varchar(50);
```

Example 2: Let's take an example of a table named **Employee:**

Emp_Id	Emp_Name	Emp_Salary	Emp_City
201	Abhay	25000	Goa
202	Ankit	45000	Delhi
203	Bheem	30000	Goa
204	Ram	29000	Goa
205	Sumit	40000	Delhi

Table: Employee

- Suppose, you want to modify the datatypes of two columns **Emp_ContactNo.** and **Emp_EmailID** of the above Employee table. For this, you have to type the following query in the SQL:

```
1. ALTER TABLE Employee ADD ( Emp_ContactNo. Int, Emp_EmailID varchar(80) ;
```

ALTER TABLE RENAME Column statement in SQL:

The RENAME keyword is used for changing the name of columns or fields of the existing table.

Syntax of ALTER TABLE RENAME Column statement in SQL

1. ALTER TABLE table_name RENAME COLUMN old_name to new_name;

Examples of ALTER TABLE RENAME Column statement in SQL

Here, we have taken the following two different SQL examples, which will help you how to change the name of a column of the existing table using ALTER TABLE statement:

Example 1: Let's take an example of a table named **Cars:**

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

Table: Cars

- Suppose, you want to change the name of the **Car_Color** column of the above Cars table. For this, you have to type the following query in the SQL:

```
1. ALTER TABLE Cars RENAME COLUMN Car_Color to Colors;
```

This statement will change the name of a column of the Cars table. To see the changes, you have to type the following query:

```
1. SELECT * FROM Cars;
```

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

Table: Cars

Example 2: Let's take an example of a table named **Employee:**

Emp_Id	Emp_Name	Emp_Salary	Emp_City
201	Abhay	25000	Goa

202	Ankit	45000	Delhi
203	Bheem	30000	Goa
204	Ram	29000	Goa
205	Sumit	40000	Delhi

Table: Employee

- Suppose, you want to change the name of **the Emp_City** column of the above Employee table. For this, you have to type the following query in the SQL:

```
1. ALTER TABLE Employee RENAME COLUMN Emp_City to Emp_Add
   ress;
```

This statement will change the name of a column of the Employee table. To see the changes, you have to type the following query:

```
1. SELECT * FROM Employee;
```

Emp_Id	Emp_Name	Emp_Salary	Emp_Address
201	Abhay	25000	Goa
202	Ankit	45000	Delhi
203	Bheem	30000	Goa
204	Ram	29000	Goa

205	Sumit	40000	Delhi

Table: Employee

Destroying tables:

Drop

Truncate

ALTER TABLE DROP Column statement in SQL

In many situations, you may require to delete the columns from the existing table. Instead of deleting the whole table or database you can use DROP keyword for deleting the columns.

Syntax of ALTER TABLE DROP Column statement in SQL

1. ALTER TABLE table_name DROP Column column_name ;

Examples of ALTER TABLE DROP Column statement in SQL

Here, we have taken the following two different SQL examples, which will help you how to delete a column from the existing table using ALTER TABLE statement:

Example 1: Let's take an example of a table named **Cars:**

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000

Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

Table: Cars

- Suppose, you want to delete the Car_Color column from the above table. For this, you have to type the following query in the SQL:

```
1. ALTER TABLE Cars DROP COLUMN Car_Color ;
```

- Let's check using the following statement that the Car_Color column is deleted from the table or not:

```
1. SELECT * FROM Cars;
```

Car Name	Car Cost
Hyundai Creta	10,85,000
Hyundai Venue	9,50,000
Hyundai i20	9,00,000
Kia Sonet	10,00,000
Kia Seltos	8,00,000

Swift Dezire	7,95,000

Table: Cars

SQL DROP TABLE

A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

This is very important to know that once a table is deleted all the information available in the table is lost forever, so we have to be very careful when using this command.

Let's see the syntax to drop the table from the database.

1. **DROP TABLE** "table_name";

Let us take an example:

First we verify STUDENTS table and then we would delete it from the database.

1. SQL> **DESC** STUDENTS;

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
ID	Int(11)	NO	PRI		
NAME	Varchar(20)	NO			
AGE	Int(11)	NO			
ADDRESS	Varchar(25)	YES		NULL	

1. 4 rows in set (0.00 sec)

This shows that STUDENTS table is available in the database, so we can drop it as follows:

1. SQL>**DROP TABLE** STUDENTS;

Now, use the following command to check whether table exists or not.

1. SQL> **DESC** STUDENTS;

1. Query OK, 0 rows affected (0.01 sec)

As you can see, table is dropped so it doesn't display it.

SQL DROP TABLE Example in MySQL

Let's see the command to drop a table from the MySQL database.

1. **DROP TABLE** table_name;

SQL DROP TABLE Example in Oracle

Let's see the command to drop a table from Oracle database. It is same as MySQL.

1. **DROP TABLE** table_name;

SQL DROP TABLE Example in Microsoft SQLServer

Let's see the command to drop a table from SQLServer database. It is same as MySQL.

1. **DROP TABLE** table_name;

SQL DELETE TABLE

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

1. **DELETE FROM** table_name [**WHERE** condition];

But if you do not specify the WHERE condition it will remove all the rows from the table.

1. **DELETE FROM** table_name;

There are some more terms similar to DELETE statement like as DROP statement and TRUNCATE statement but they are not exactly same there are some differences between them.

Difference between DELETE and TRUNCATE statements

There is a slight difference b/w delete and truncate statement.
The **DELETE statement** only deletes the rows from the table based on the condition defined by WHERE clause or delete all the rows from the table when condition is not specified.

But it does not free the space containing by the table.

The **TRUNCATE statement:** it is used to delete all the rows from the table **and free the containing space.**

Let's see an "employee" table.

Emp_id	Name	Address	Salary
1	Aryan	Allahabad	22000
2	Shurabhi	Varanasi	13000
3	Pappu	Delhi	24000

Execute the following query to truncate the table:

1. **TRUNCATE TABLE** employee;

Difference b/w DROP and TRUNCATE statements

When you use the drop statement it deletes the table's row together with the table's definition so all the relationships of that table with other tables will no longer be valid.

When you drop a table:

- Table structure will be dropped
- Relationship will be dropped
- Integrity constraints will be dropped
- Access privileges will also be dropped

On the other hand when we **TRUNCATE** a table, the table structure remains the same, so you will not face any of the above problems.

SQL TRUNCATE TABLE

A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

TRUNCATE TABLE Vs DELETE TABLE

Truncate table is faster and uses lesser resources than DELETE TABLE command.

TRUNCATE TABLE Vs DROP TABLE

Drop table command can also be used to delete complete table but it deletes table structure too. TRUNCATE TABLE doesn't delete the structure of the table.

Let's see the syntax to truncate the table from the database.

1. **TRUNCATE TABLE** table_name;

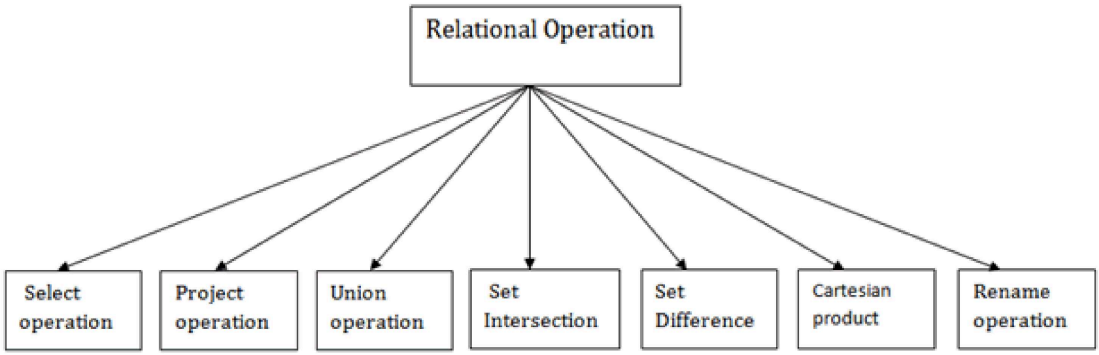
For example, you can write following command to truncate the data of employee table

1. **TRUNCATE TABLE** Employee;

Relational Algebra:

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.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: σ p(r)

Where:

σ is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1. σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏.

1. Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. ∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93

Smith	L-11
Williams	L-17

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams

Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) -
   ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT

1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal

2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

Relational Calculus:

There is an alternate way of formulating queries known as Relational Calculus. Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results. The relational calculus tells what to do but never explains how to do. Most commercial relational languages are based on aspects of relational calculus including SQL-QBE and QUEL.

Why it is called Relational Calculus?

It is based on Predicate calculus, a name derived from branch of symbolic language. A predicate is a truth-valued function with arguments. On substituting values for the arguments, the function result in an expression called a proposition. It can be either true or false. It is a tailored version of a subset of the Predicate Calculus to communicate with the relational database.

Many of the calculus expressions involves the use of Quantifiers. There are two types of quantifiers:

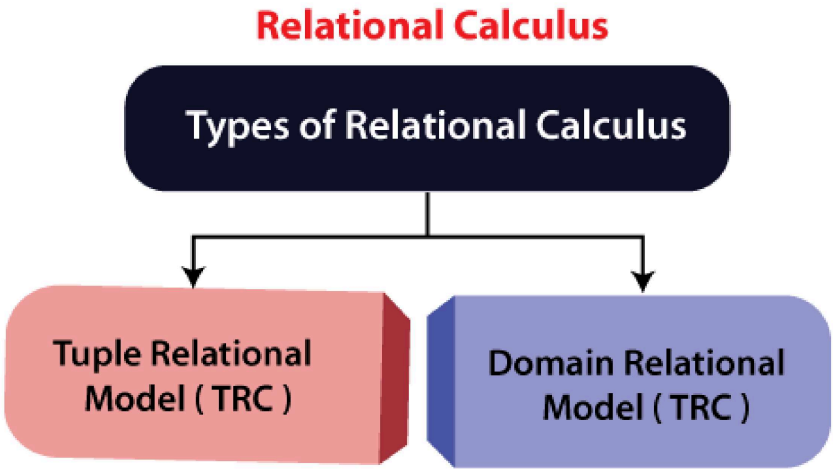
- **Universal Quantifiers:** The universal quantifier denoted by ∀ is read as for all which means that in a given set of tuples exactly all tuples satisfy a given condition.
- **Existential Quantifiers:** The existential quantifier denoted by ∃ is read as for all which means that in a given set of tuples there is at least one occurrences whose value satisfy a given condition.

Before using the concept of quantifiers in formulas, we need to know the concept of Free and Bound Variables.

A tuple variable t is bound if it is quantified which means that if it appears in any occurrences a variable that is not bound is said to be free.

Free and bound variables may be compared with global and local variable of programming languages.

Types of Relational calculus:



1. Tuple Relational Calculus (TRC)

It is a non-procedural query language which is based on finding a number of tuple variables also known as range variable for which predicate holds true. It describes the desired information without giving a specific procedure for obtaining that information. The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.

Notation:

A Query in the tuple relational calculus is expressed as following notation

1. {T | P (T)} or {T | Condition (T)}

Where

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

1. { T.name | Author(T) AND T.article = 'database' }

Output: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

For example:

1. { R| ∃T ∈ Authors(T.article='database' AND R.name=T.name)}

Output: This query will yield the same result as the previous one.

2. Domain Relational Calculus (DRC)

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes. Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ¬ (not). It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable. The QBE or Query by example is a query language related to domain relational calculus.

Notation:

1. { a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where

a1, a2 are attributes
P stands for formula built by inner attributes

For example:

1. {< article, page, subject > | ∈ dbms ∧ subject = 'database'}