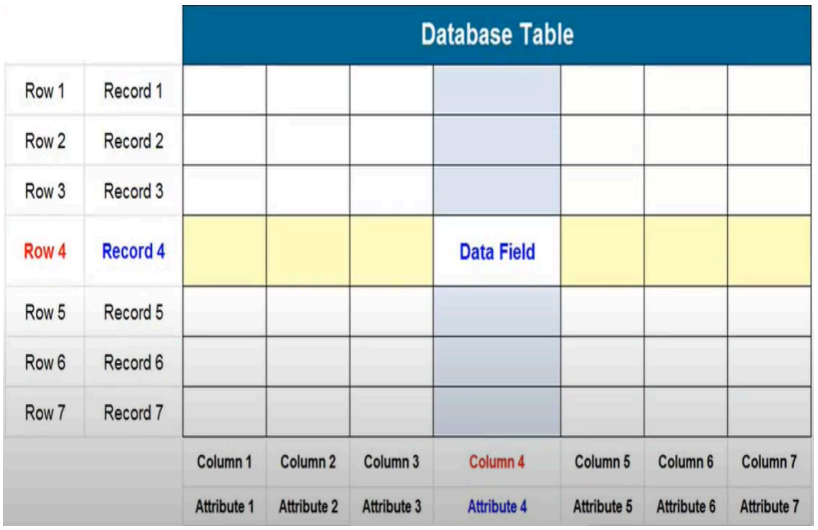
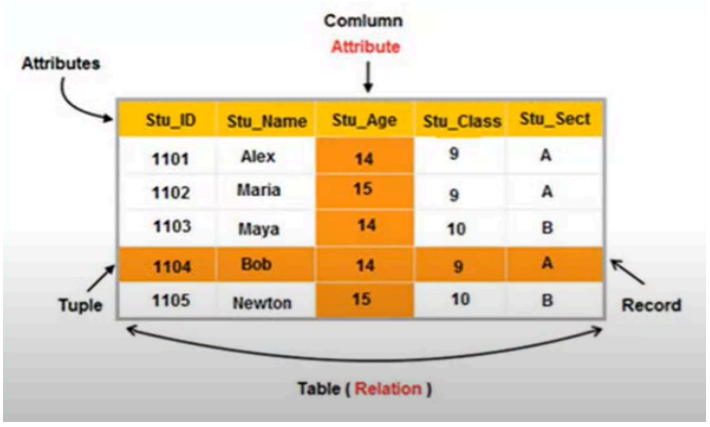
UNIT 02:

Structure of Relational Databases - Relational Algebra Fundamentals - Introduction to SQL: Basic Structure - Set operations - Aggregate functions - Nested Sub queries - Complex queries – Join Expressions-Views - Modification of the database - Integrity constraints - Referential Integrity - Triggers – Assertions- Formal Relational Query Languages: The Tuple Relational Calculus- The Domain Relational Calculus.

Relational Database

- Database is organized as group of related tables
- Each table represents a database entity
- Each table is a group of column and rows, where
  - column represents attribute of an entity
  - rows represents records.

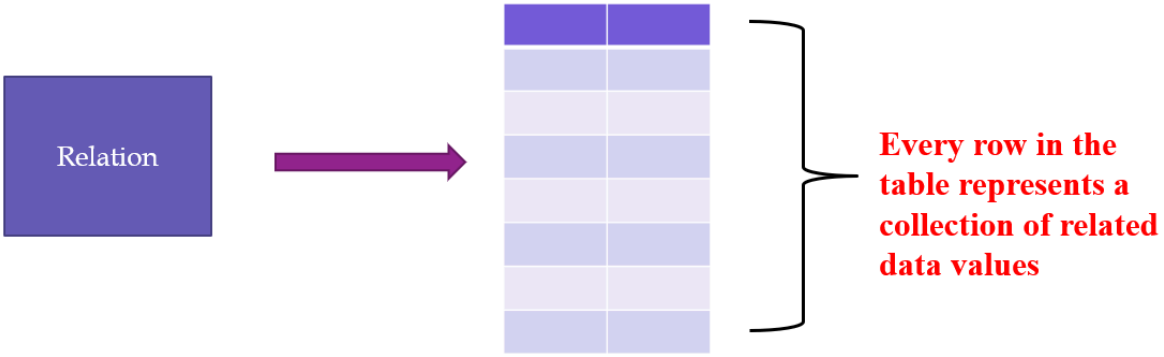




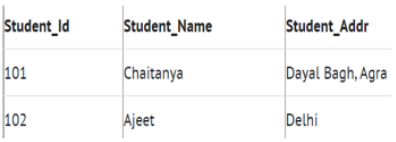
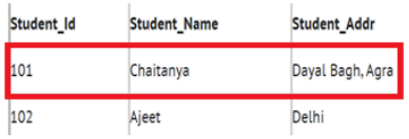
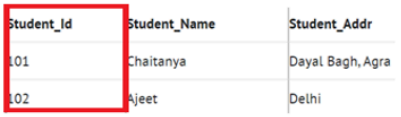
- Model data in the form of relations or tables

Relational Model

- Relational model represents **how data is stored** in relational databases
- Represents the **database** as a **collection of relations**
- In relational model, the data and relationships are represented by collection of inter-related tables.



Relational Model Terminologies

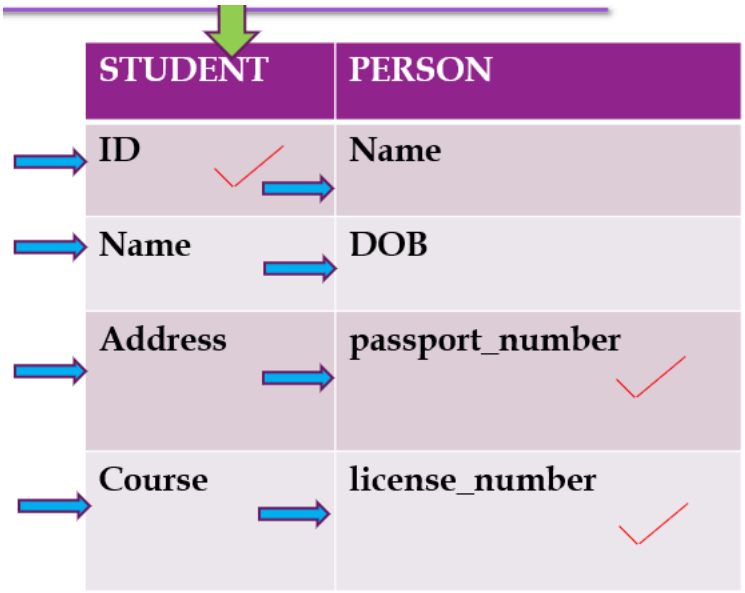
Table or Relation	collection of data represented in rows and columns	
Record or Tuple	Each row of a table	
Field or Column name or Attribute	Each column in a Table	

Domain	set of permitted values for an attribute in table	For e.g., domain of month-of-year can accept January, February,…December
Degree	total number of attributes	-
Cardinality	total number of rows present in the Table	-
Relation Schema	represents name of the relation with its attributes	e.g.; **STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, AGE)** is relation schema for STUDENT
NULL Values	value which is not known or unavailable is called NULL value	represented by blank space

relation instance	specific instance of a relation, i.e., containing a specific set of rows at that moment of time	-
Keys	they can uniquely identify the tuple	-

Keys

- 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:** In STUDENT table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_numberare keys since they are unique for each person.



Types of Keys

- Primary Key
- Candidate Key
- Alternate Key
- Super Key
- Foreign Key
- Unique Key
- Composite key

Super Key

- Like Super Set
- **Uniquely Identify** the Tuple
- It may contain NULL values
- {Name} is not a super Key – Because no uniqueness
- It May Contain **extraneous attributes**.

- SuperKeys :

{ID},{SSN},{ID,Name},

{ID,SSN},{ID,PHONE},

{NAME,PHONE},{ID,EMAIL}

{NAME,SSN,PHONE}

{NAME,EMAIL}

{ID,SSN,PHONE}….

Candidate Key

- Minimal super Keys are called candidate keys.
- Candidate Keys:
- {ID},{SSN},{NAME,PHONE},{EMAIL}

Primary Key

- To denote a candidate key.
- Candidate Keys:
  {ID},{SSN},{NAME,PHONE},{EMAIL}
- Primary Key: {ID}
- Chosen with care by DBA
- Values present in that attribute Never or very rarely changed.
- Candidate key with NULL value is NOT the primary key.
- Primary Key = UNIQUE + NOT NULL

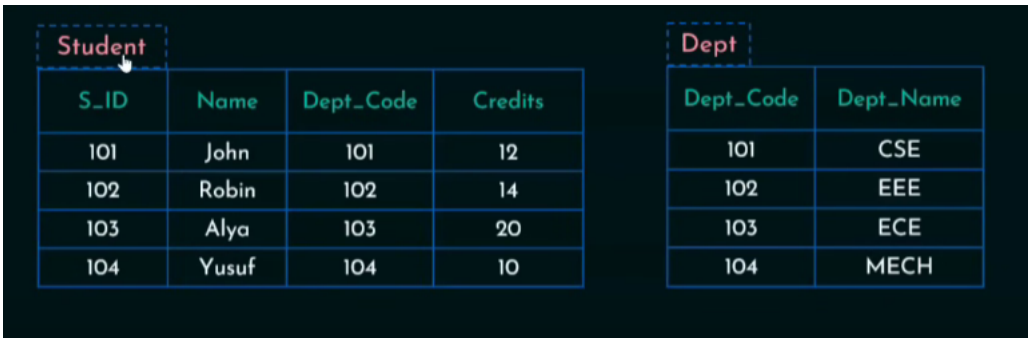
Alternate Key

- The candidate key other than the primary key.
- All the keys which are not primary keys.
- Candidate Keys:
- {ID},{SSN},{NAME,PHONE},{EMAIL}
- Primary Key: {ID}
- Alternate Keys:
  {SSN},{NAME,PHONE},{EMAIL}

Unique Key

- Candidate Keys:
  {ID},{SSN},{NAME,PHONE},{EMAIL}
- Primary Key: {ID}
- Alternate Keys:
  {SSN},{NAME,PHONE},{EMAIL}…
- Unique Key: {NAME,PHONE}},{EMAIL}…(Unique in terms of values and may have null values)
- **Composite Key:** {NAME,PHONE}…(Comprised of one or more attribute)

Foreign Key

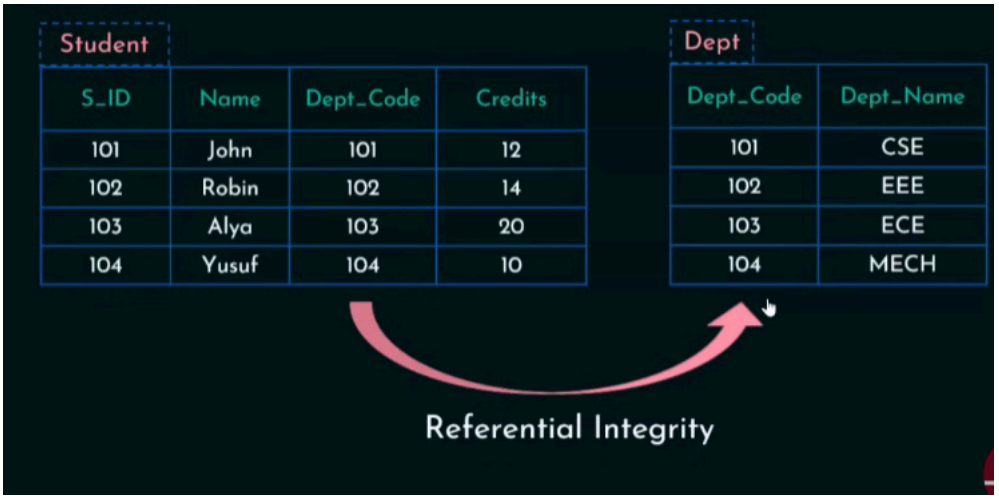


Student

S_ID	Name	Dept_Code	Credits
101	John	101	12
102	Robin	102	14
103	Alya	103	20
104	Yusuf	104	10

Dept

Dept_Code	Dept_Name
101	CSE
102	EEE
103	ECE
104	MECH



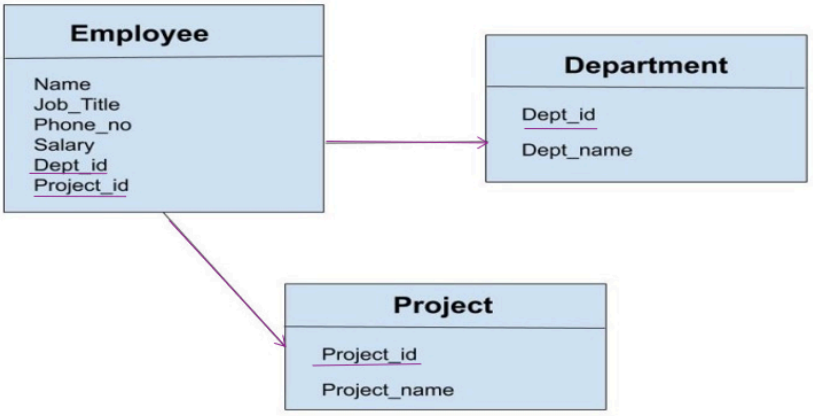
Schema

- The **design of the database** is called a **schema.** It is the structural view of the database. It gives us an **overall description of the database**.
- A database schema defines how the **data is organised** using the schema diagram. A schema diagram is a diagram which contains **entities and the attributes** that will define that schema.
- A schema diagram only shows us the **database design**. It does **not show** the **actual data** of the database.
- **Schema** can be a **single table** or it can have **more than one table** which is related. The **schema represents the relationship** between these tables.

Schema Diagram

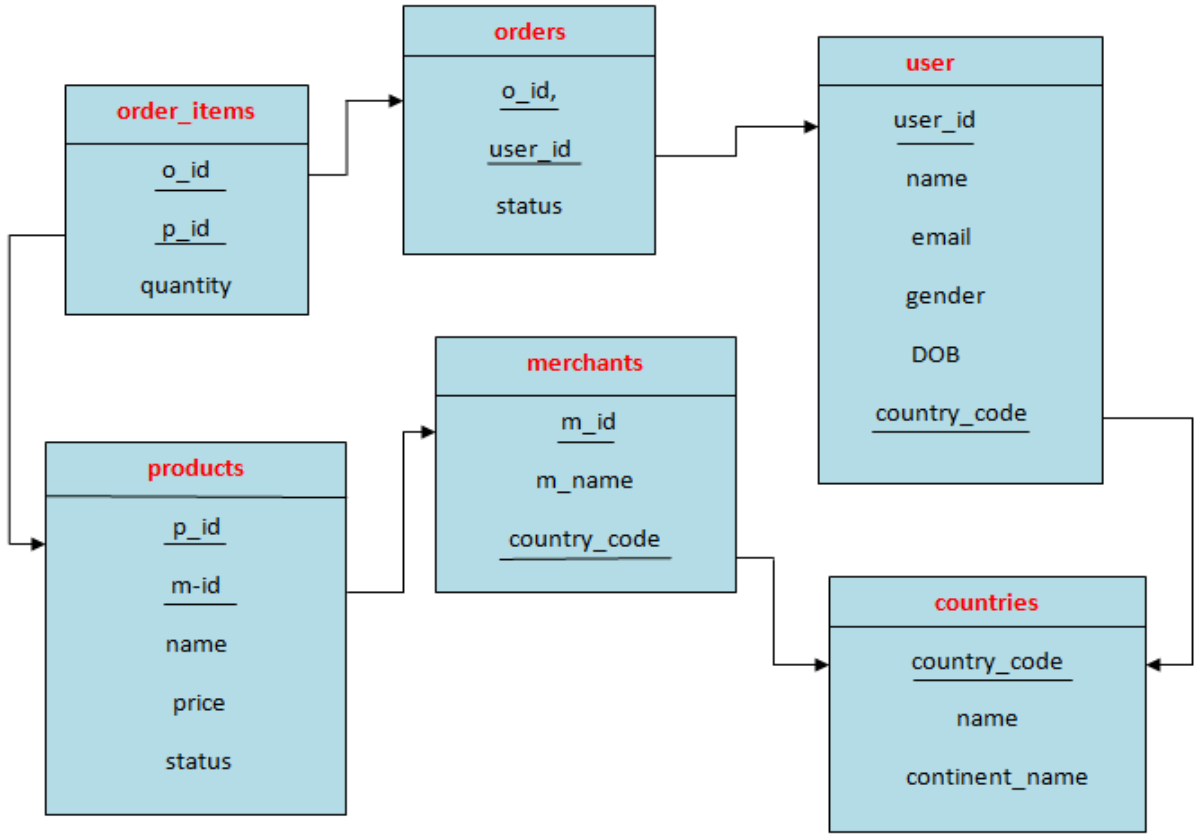
- A database schema, along with **primary key and foreign key dependencies**, can **be depicted by schema diagrams**
- Each **relation** appears as a **box**, with the relation **name at the top**, and the **attributes listed inside the box**
- **Primary key** attributes are shown **underlined**
- **Foreign key** dependencies appear as **arrows** from the foreign key attributes of the **referencing** relation to the **primary key** of the referenced relation.

Example: Three tables Employee, Department and Project tables are related.



Draw Schema diagram for the following relational schema:

1. user(user_id, name, email, gender, DOB, country_code)
2. products(p_id, m-id, name, price, status)
3. orders (o_id, user_id, status)
4. order_items(o_id, p_id, quantity)
5. merchants(m_id, m_name, country_code)
6. countries(country_code, name, continent_name)



Query Language

- A query language is a language in which a user requests information from the database

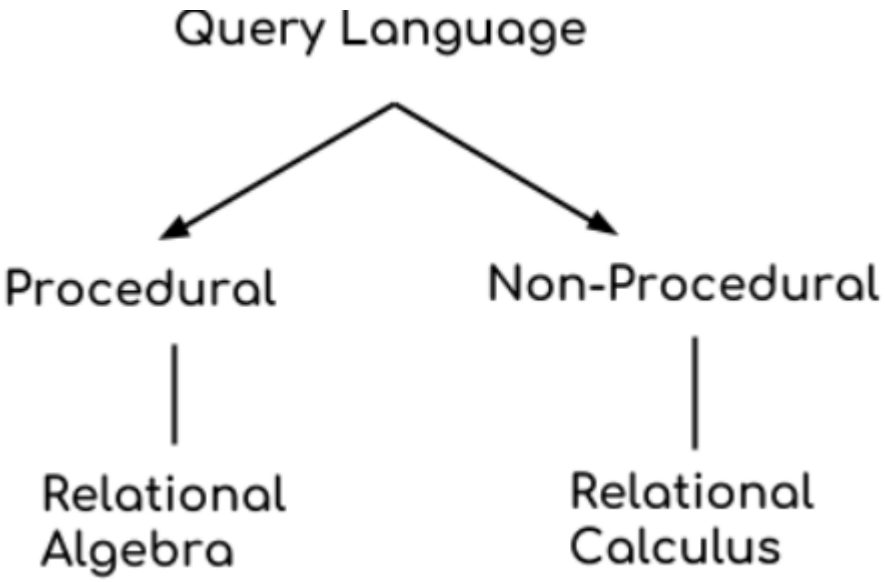
  a Language which is used to store and retrieve data from database
- **Eg: SQL**
- Two types:
  - Procedural Query language
  - Non-procedural query language

Procedural Query language	Non-procedural query language
User instructs the system to perform a sequence of operations on the database to compute the desired result	User describes the desired information without giving a specific procedure for obtaining that information
Users tells what data to be retrieved from database and how to retrieve it.	Users tells what data to be retrieved from database but doesn't tell how to retrieve it.

- **Relational Algebra**

  Conceptual procedural query language used on relational model
- **Relational Calculus**

  Conceptual non-procedural query language used on relational model
- Relational algebra and calculus are the theoretical concepts used on relational model
- SQL is a practical implementation of relational algebra and relational calculus



INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

SELECT:

Question 1: Write an RA expression to find all the instructors working in 'Finance' department.

Question 2: Find all the instructors with salary greater than $87000

Question 3: Find all the instructors who are working in "Finance" department and drawing the salary greater than $87000

Question 4: Write an RA expression to find all the instructors in the INSTRUCTOR relation.

PROJECT:

Question 1: List all the INSTRUCTORS ID,Name,Salary ,but do not care about the dept_name

Question 2: Find the name of all instructors in the cse dept.

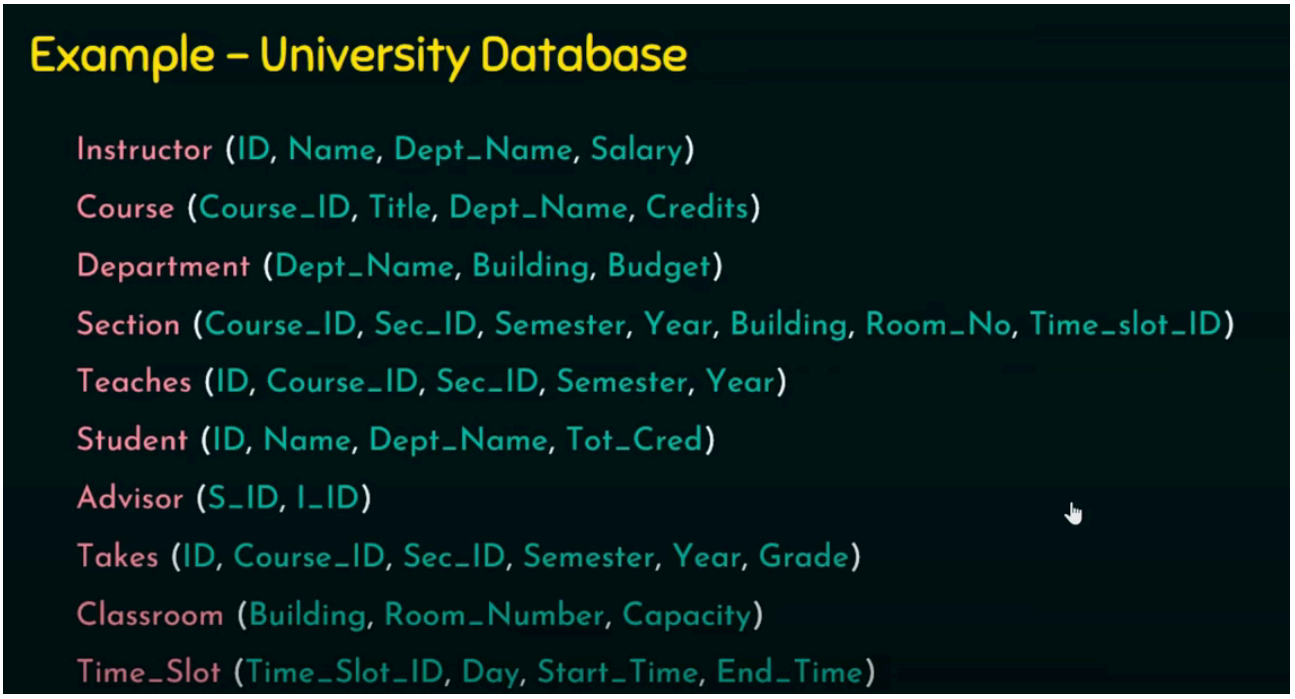
UNION:

Question 1: List all the customer names associated with the bank either as an account holder or a loan borrower.

Question 2: Find the set of all courses taught in the Fall 2009 semester, the spring 2010 semester, and both.

Question 3: List all the course IDs which are taken in spring 2020 ,Fall 2021.

Question 4: List all the instructors' ID who taught courses in spring 2020 , Fall 2021.



SET DIFFERENCE

Question 1: List all customer names those who have a deposit account but not availed loan.

Question 2: Find all the courses taught in the Fall 2009 semester but not in spring 2010.

CARTESIAN PRODUCT

Question 1: Find the names of all instructors in the physics department together with the course id of all courses they taught**.**

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
78787	Raj	Physics	87000

TEACHES

ID	Course_ID	Sec_ID	Semester	Year
10101	BIO-108	1	Summer	2009
20202	CS-103	1	Spring	2010
78787	PHY-101	2	Fall	2011
12345	EE-203	1	Spring	2009

INSTRUCTOR X TEACHES

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
10101	John	Biology	65000	10101	BIO-108	1	Summer	2009
10101	John	Biology	65000	20202	CS-103	1	Spring	2010
10101	John	Biology	65000	78787	PHY-101	2	Fall	2011
10101	John	Biology	65000	12345	EE-203	1	Spring	2009
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

$\sigma_{Instructor.ID = Teaches.ID}$ ($\sigma_{Dept\_Name = "Physics"}$ (INSTRUCTOR X TEACHES))

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

$\sigma_{Instructor.ID = Teaches.ID}$ ($\sigma_{Dept\_Name = "Physics"}$ (INSTRUCTOR X TEACHES))

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011

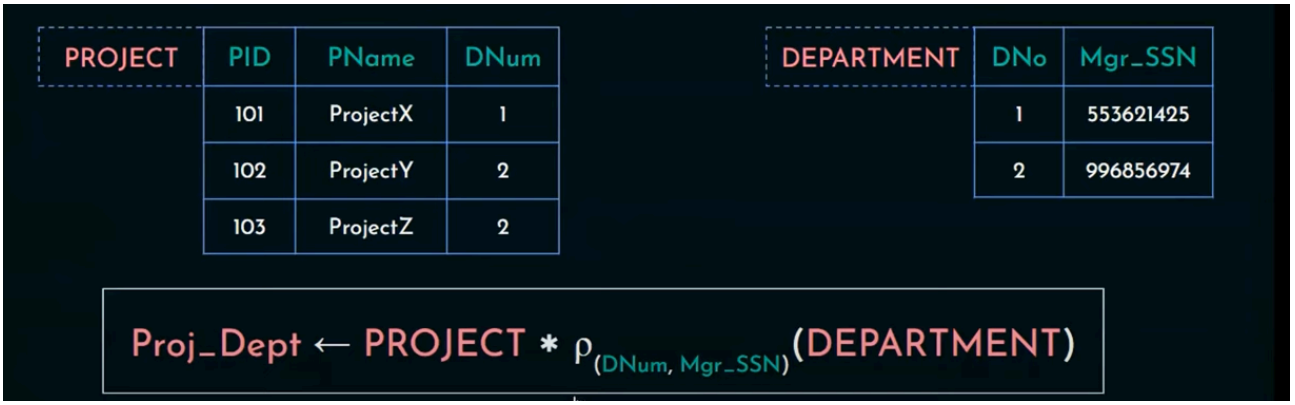
$\Pi_{name, course\_id}$ ($\sigma_{Instructor.ID = Teaches.ID}$ ($\sigma_{Dept\_Name = "Physics"}$ (INSTRUCTOR X TEACHES)))

Name	Course_ID
Raj	PHY-101

SET INTERSECTION:

Question 1: Find the names of all customers who have deposited money and also availed loan.

NATURAL JOIN



SET OPERATORS:

Four Different types of set operators are:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

UNION

- UNION is used to combine the results of two or more SELECT statements.
- It will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Example :

ID	Name
1	AB
2	CD

ID	Name
2	CD

3	EF

Syntax :

SELECT column_name FROM table1

UNION

SELECT column_name FROM table2;

Query :

SELECT * FROM First

UNION

SELECT * FROM Second;

Union All

- Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

SELECT column_name FROM table1

UNION ALL

SELECT column_name FROM table2;

Intersect

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax

SELECT column_name FROM table1

INTERSECT

SELECT column_name FROM table2;

Minus

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax:

SELECT column_name FROM table1

MINUS

SELECT column_name FROM table2;

Aggregate Functions

These functions return a single value after performing calculations on a group of values. Following are some of the frequently used Aggregrate functions.

AVG() Function

Average returns average value after calculating it from values in a numeric column.

Syntax:

SELECT AVG(column_name) FROM table_name

\

COUNT() Function

Count returns the number of rows present in the table either based on some condition or without condition.

Syntax:

SELECT COUNT(column_name) FROM table-name

MAX() Function

MAX function returns maximum value from selected column of the table.

Syntax:

SELECT MAX(column_name) from table-name;

MIN() Function

MIN function returns minimum value from a selected column of the table.

Syntax:

SELECT MIN(column_name) from table-name;

SUM() Function

SUM function returns total sum of a selected columns numeric values.

Syntax:

SELECT SUM(column_name) from table-name;

FIRST() Function

First function returns first value of a selected column

Syntax:

SELECT FIRST(column_name) FROM table-name;

LAST() Function

LAST function returns the return last value of the selected column.

Syntax:

SELECT LAST(column_name) FROM table-name;

SUB QUERY

1. A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
2. You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
3. A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
4. Subqueries are on the right side of the comparison operator.
5. A subquery is enclosed in parentheses.

Select Statement

SQL subqueries are most frequently used with the Select statement.

Syntax

SELECT column_name

FROM table_name

WHERE column_name expression operator

(SELECT column_name from table_name WHERE ...);

INSERT Statement

- SQL subquery can also be used with the Insert statement. In the insert statement, data returned from the subquery is used to insert into another table.
- In the subquery, the selected data can be modified with any of the character, date functions

Syntax:

INSERT INTO table_name (column1, column2, column3....)

SELECT *

FROM table_name

WHERE VALUE OPERATOR

UPDATE Statement

The subquery of SQL can be used in conjunction with the Update statement. When a subquery is used with the Update statement, then either single or multiple columns in a table can be updated.

Syntax

UPDATE table

SET column_name = new_value

WHERE VALUE OPERATOR

(SELECT COLUMN_NAME

FROM TABLE_NAME

WHERE condition);

DELETE Statement

The subquery of SQL can be used in conjunction with the Delete statement just like any other statements mentioned above.

Syntax

DELETE FROM TABLE_NAME

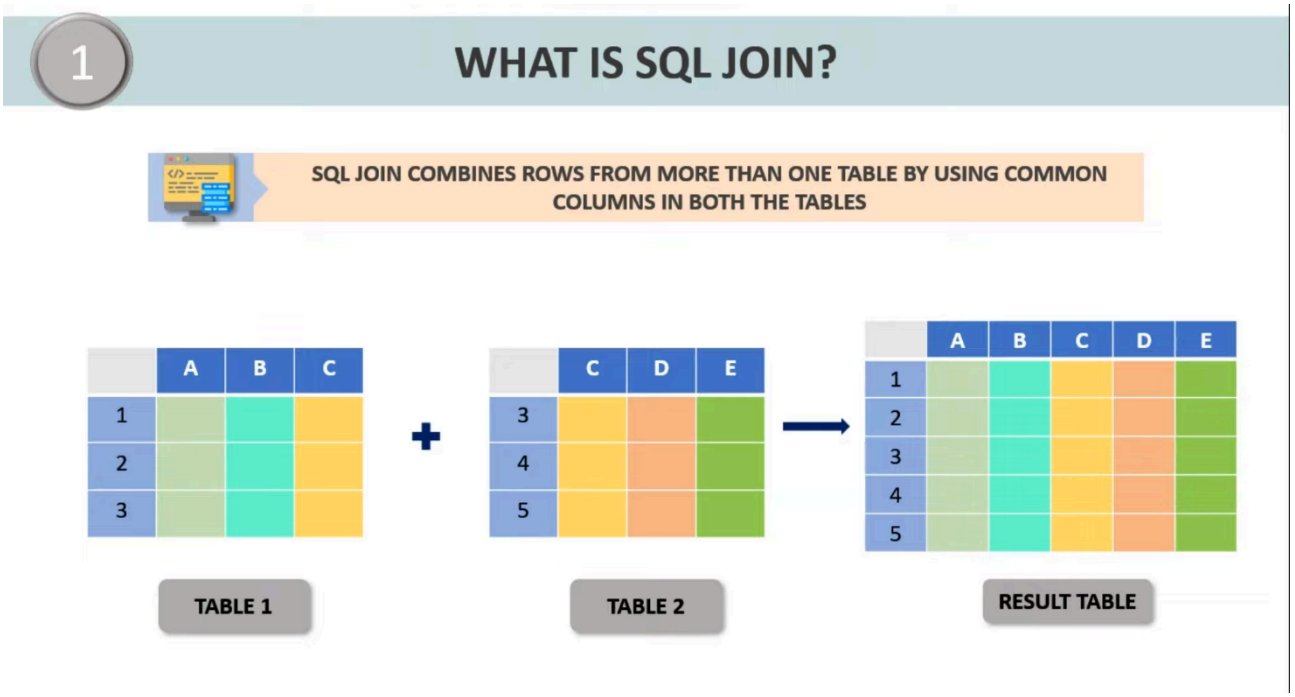
WHERE VALUE OPERATOR

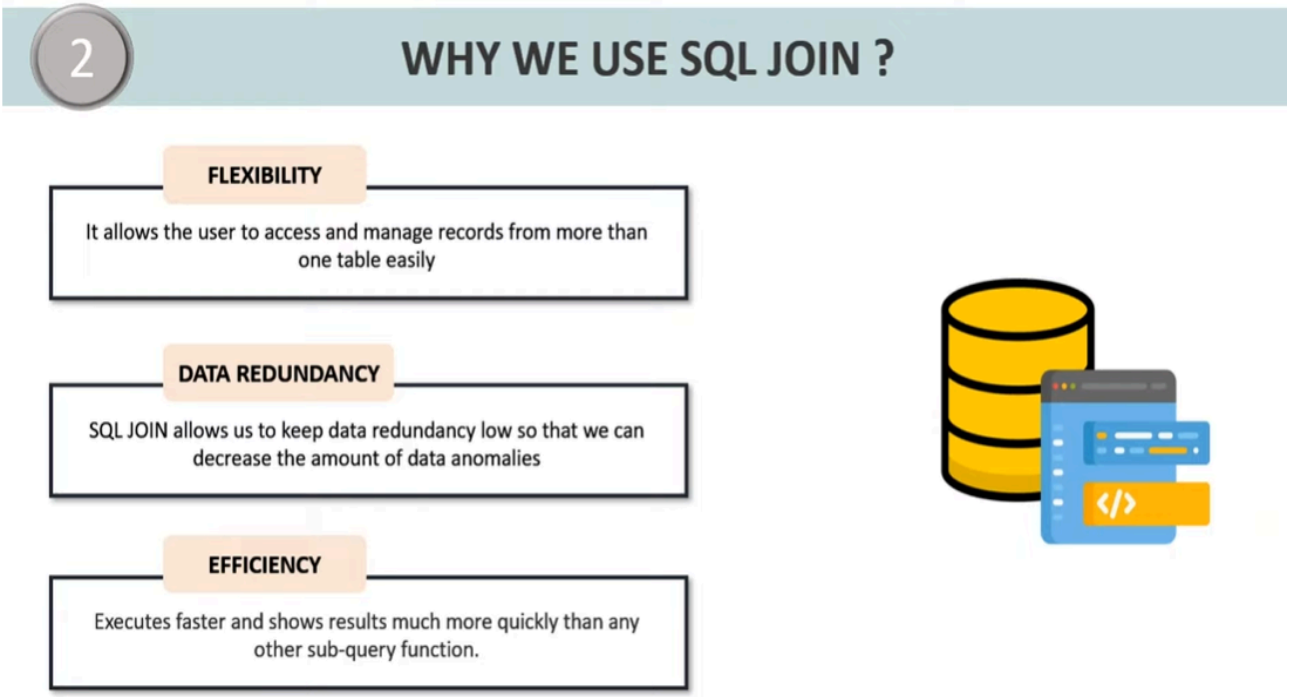
(SELECT COLUMN_NAME

FROM TABLE_NAME

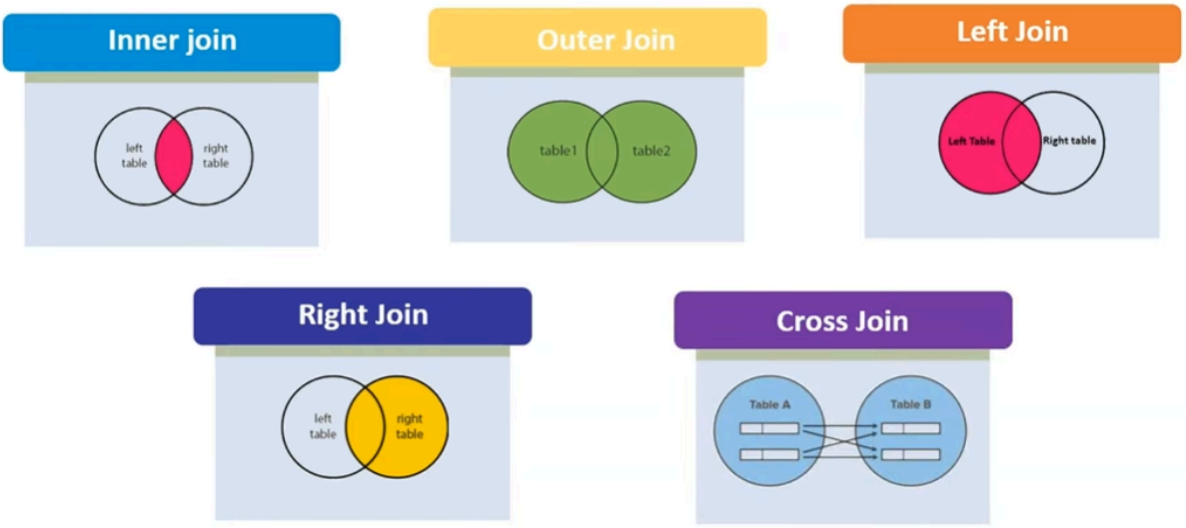
WHERE condition);

SQL JOINS

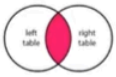

1
WHAT IS SQL JOIN?
SQL JOIN COMBINES ROWS FROM MORE THAN ONE TABLE BY USING COMMON COLUMNS IN BOTH THE TABLES
A B C
1
2
3
TABLE 1
C D E
3
4
5
TABLE 2
A B C D E
1
2
3
4
5
RESULT TABLE


2
WHY WE USE SQL JOIN ?
FLEXIBILITY
It allows the user to access and manage records from more than one table easily
DATA REDUNDANCY
SQL JOIN allows us to keep data redundancy low so that we can decrease the amount of data anomalies
EFFICIENCY
Executes faster and shows results much more quickly than any other sub-query function.

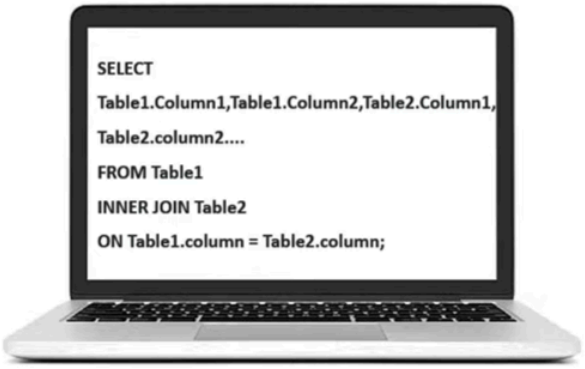
3 TYPES OF SQL JOINS



4 SQL INNER JOIN

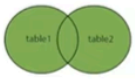


INNER JOIN RETURNS A NEW TABLE BY COMBINING RECORDS THAT ONLY HAVE MATCHING VALUES IN BOTH THE TABLES.

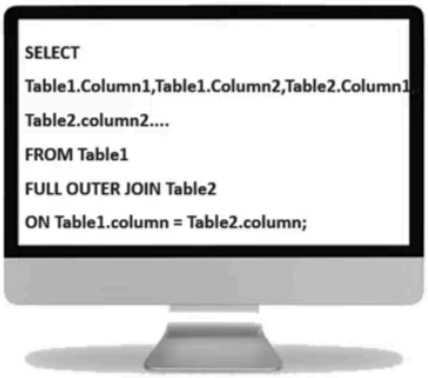


SYNTAX

5 SQL OUTER JOIN

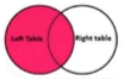


OUTER JOIN RETURNS ALL THOSE RECORDS WHICH ARE IN EITHER THE LEFT TABLE OR RIGHT TABLE



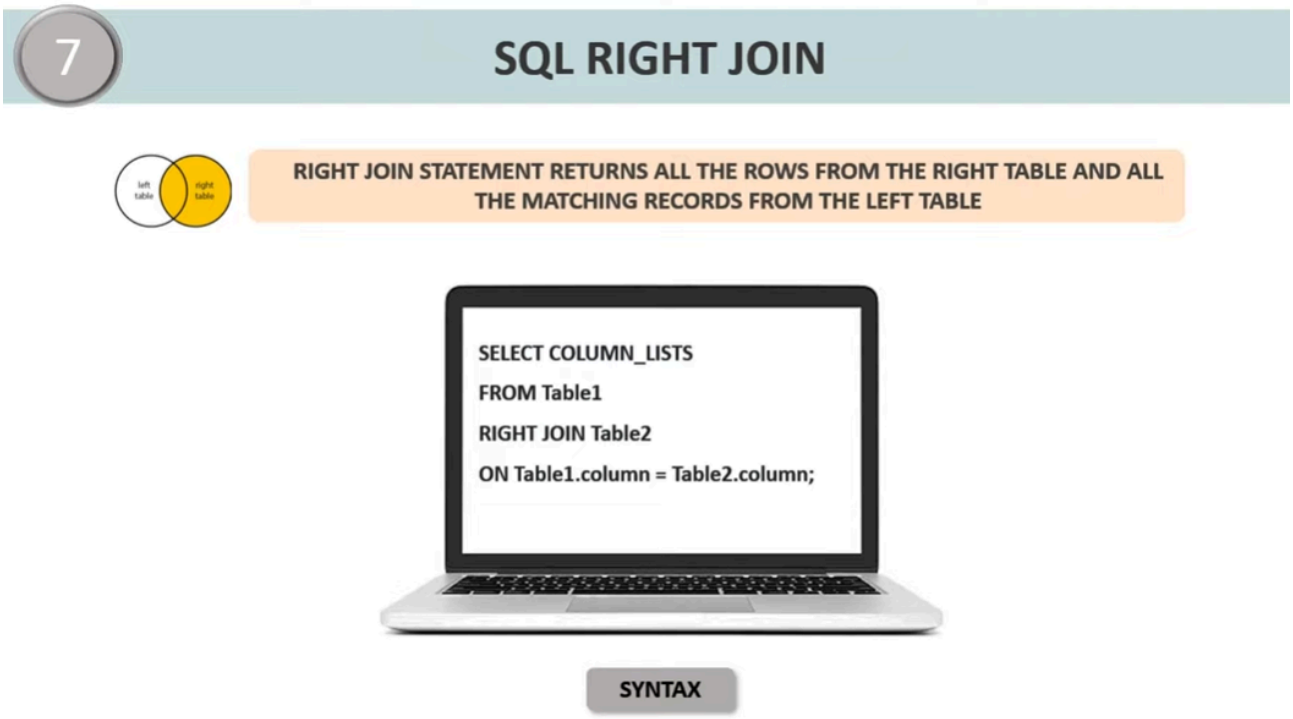
SYNTAX

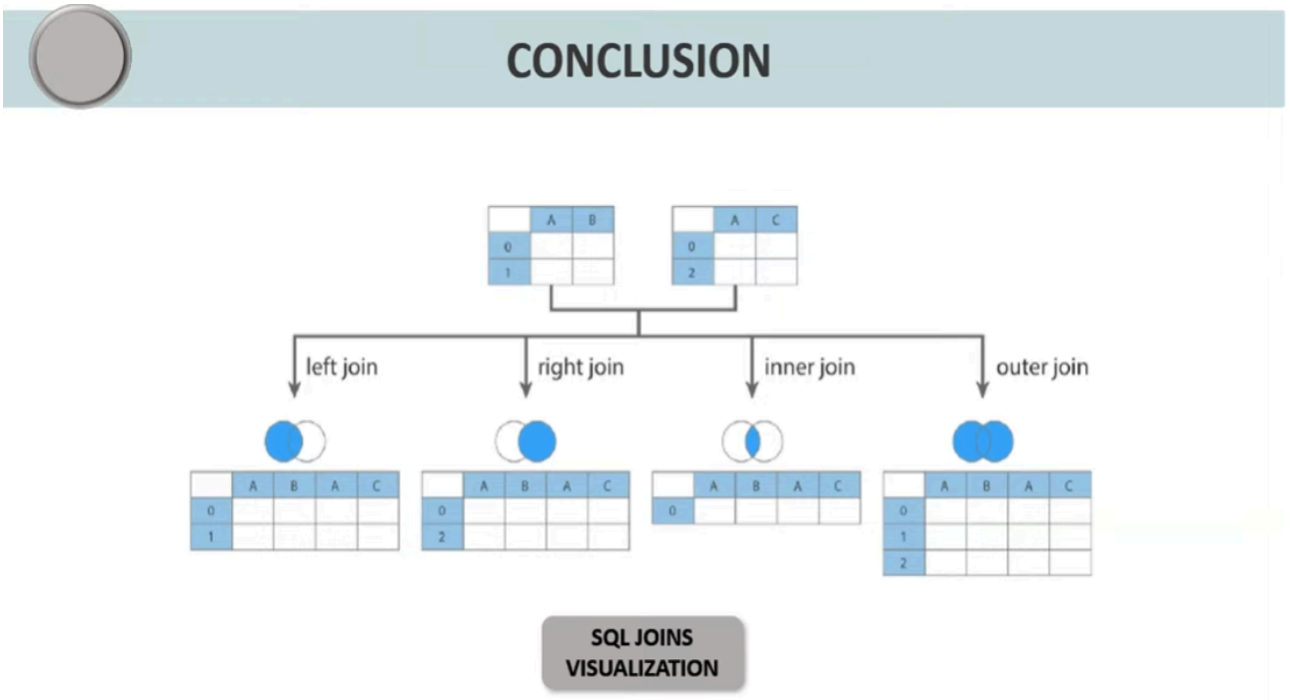
6 SQL LEFT JOIN



LEFT JOIN RETURNS ALL THE ROWS FROM THE LEFT TABLE AND MATCHING ROWS FROM THE RIGHT TABLE

SYNTAX





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.

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:

CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE condition;

Creating View from a single table

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

Query:

CREATE VIEW DetailsView AS

SELECT NAME, ADDRESS

FROM Student_Details

WHERE STU_ID < 4;

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:

CREATE VIEW MarksView AS

SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS

FROM Student_Detail, Student_Mark

WHERE Student_Detail.NAME = Student_Marks.NAME;

Deleting View

A view can be deleted using the Drop View statement.

Syntax

DROP VIEW view_name;

Integrity Constraints

- 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.

Types

1. Domain Constraint
2. Entity Integrity Constraint
3. Referential Integrity Constraint
4. Key Constraint

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

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.

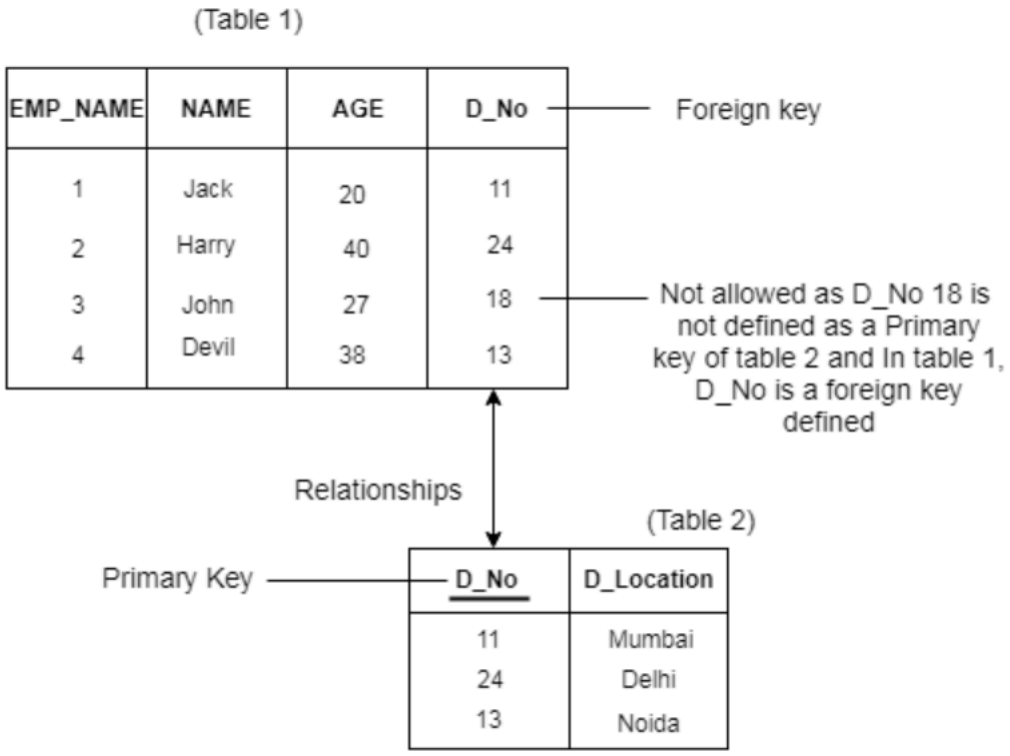
EMPLOYEE

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

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.



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 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