DATABASE CONCEPT

1. DATA
2. DBMS
3. DATA MODEL
4. DBA
5. DATABASE USER
6. E-R MODEL
7. ENTITY
8. KEYS
9. RELATIONSHIP (UNARY, BINARY, TERNARY, N-RAY)
10. CARDINALITY (ONE TO ONE, ONE TO MANY, MANY TO ONE, MANY TO MANY)
11. NORMALIZATION (1NF, 2NF, 3NF, BCNF, 5NF)
12. DATA TYPE (Numeric, Character, Date and Time, Boolean, binary)
13. INTEGRITY CONSTRAINTS
14. SQL COMMANDS (DDL, DML, DCL, DTL)
15. CONCEPT OF TRANSACTION
16. JOINING TABLES (Inner, Left, Right, Cross)
17. SUB QUERY
18. Procedure
19. Trigger
20. Cursor

DATA ?

Data means any Raw facts and figures that are incomplete by themselves, such as —

- Numbers (123, 45.6)
- Words (names, place names)
- Images
- Videos
- Audio
- Symbols
- Facts

DBMS

A database Management System is an organised collection of data, stored in a way that makes it easy to access, manage, and update.

Database Element

- **Data -** Raw facts and figures are called Data.
- **DBMS (Database management system) -**Software that helps create, manage, and access the database. Examples include MySQL, Oracle, and SQL Server.
- **Table -**Data is organized into tables with rows and columns.
- **Schema -** The structure of the database that defines how the data will be organized.

Type of Database

- **Relational Database -** Data is organized in tables, and relationships (relations) are established between different tables. Examples: SQL, PostgreSQL.
- **NoSQL Database -** Data is stored in formats other than tables, such as documents, graphs, or key-value pairs. Examples: MongoDB, Cassandra.

DBMS SOFTWARE

- **MySQL**
- **Office access**
- **Oracle**
- **FoxPro**
- **DB2**

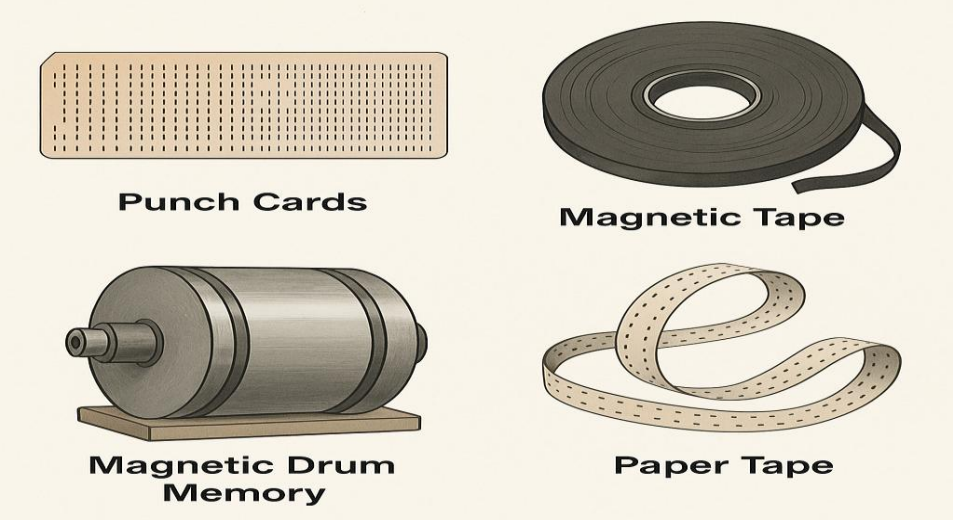
HISTORY OF DBMS

It began in the 1950s when data was stored and managed manually using file-based systems. As the volume and complexity of data grew, the need for a more efficient, organized, and systematic approach to data management led to the development of DBMS.

Development Timeline of DBMS

1950s - The Era of File-Based Systems

- Data was stored in **file-based systems**, where information was organized into separate files.
- These systems were manual and lacked a centralized management system, making it difficult to share or integrate data.
- There were no standard rules for accessing or managing the data.



The Birth of the Relational Model

- **Dr. Edgar Frank Codd** introduced the **Relational Database Model** in 1970 through his paper, *"A Relational Model of Data for Large Shared Data Banks.*.
- The relational model organized data into **tables (relations)**, using rows (tuples) and columns (attributes)
- The model was based on **set theory and relational algebra**, making it logical and easy to use.
- **SQL (Structured Query Language)** was developed to query relational databases.

DATA Model

- A **data model** is a conceptual framework that defines how data is structured, organized, and interrelated in a database.

Types of Data Model

- **Hierarchical data model**
- **Network data model**
- **Relational model**
- **Object-Oriented Data Model**
- **Document Data Model**
- **Entity-Relationship Model (ER Model)**

Hierarchical data model

- In this model, data is organized in a tree-like structure with a hierarchy of parent-child relationships.
- Each child node has only one parent, but a parent can have multiple children.
- **Example**: Organizational structures, file systems.

Network Data Model

- Similar to the hierarchical model, but allows more complex relationships with multiple parent nodes.
- Uses a graph structure where entities are represented as nodes, and relationships are edges.
- **Example**: Social networks, transport routes.

Relational Data Model

- In this model, data is organized into tables (also called relations), with rows (records) and columns (attributes).
- Each table represents an entity, and tables can be related to each other through primary and foreign keys.
- **Example**: Most modern databases like MySQL, PostgreSQL, Oracle

Object-Oriented Data Model

- Data is represented as objects, similar to object-oriented programming.
- Supports data types, methods, and inheritance, allowing complex data structures
- **Example**: Databases used in complex applications, like CAD (Computer-Aided Design) and multimedia systems.

Document Data Model

- Commonly used in NoSQL databases, where data is stored as documents in formats like JSON, XML, or BSON.
- Documents can store nested data structures and are flexible with schema changes.
- Example: MongoDB, CouchDB.

Entity-Relationship Model (ER Model)

- This is a high-level, conceptual data model that defines data elements as entities, attributes, and relationships.
- Often used in the design phase to map out how entities relate to each other.
- Example: Designing a relational database schema**.**

DBA

A **Database Administrator (DBA)** is a professional responsible for managing, maintaining, and securing an organization's databases. The DBA ensures that the database is functional, efficient, and accessible while safeguarding its integrity and security.

Role of DBA

- **Database Design and Implementation**:
- **Performance Tuning and Optimization**:
- **Backup and Recovery**:
- **Security Management**:
- **Database Maintenance**:
- **Data Integrity and Consistency**:
- **Collaboration with Teams**:

Types of DBAs

- **System DBA**: Focuses on the technical aspects of database management, such as installing, configuring, and maintaining the DBMS software.
- **Application DBA**: Works closely with application developers to optimize database queries and ensure compatibility with software applications.

- **Development DBA**: Involved in designing, developing, and testing new database systems.
- **Operational DBA**: Handles routine tasks like backups, recovery, and system monitoring.

Database Users

Database users are individuals or groups who interact with a database system to perform various operations like storing, retrieving, updating, and managing data.

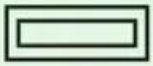
Types of Database Users

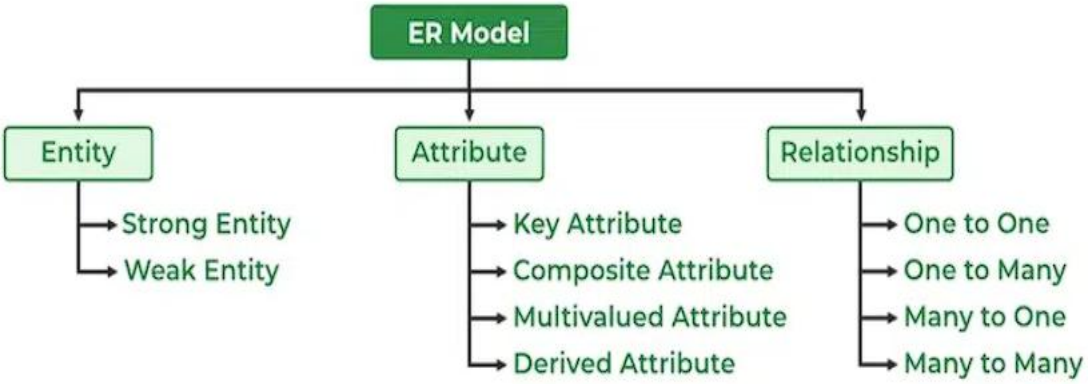
- Database Administrators (DBA)
- End Users
- Application Programmers
- System Analysts
- Database Designers
- Data Scientists and Analysts
- Testers

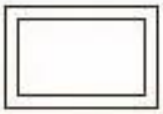
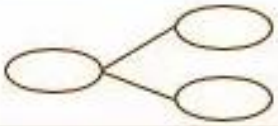
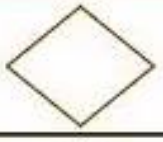
E.R. MODEL

Entity Relationship

The Entity-Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies an enterprise schema that represents the overall logical structure of a database graphically.

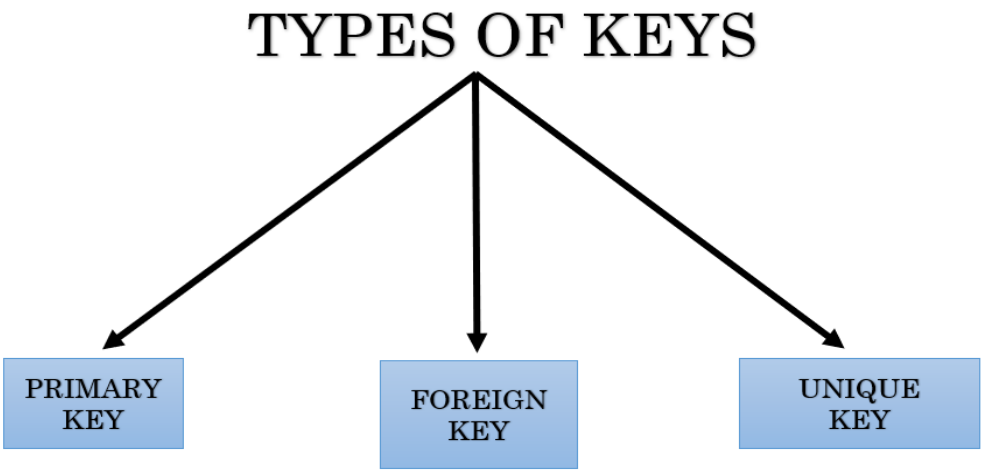
Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

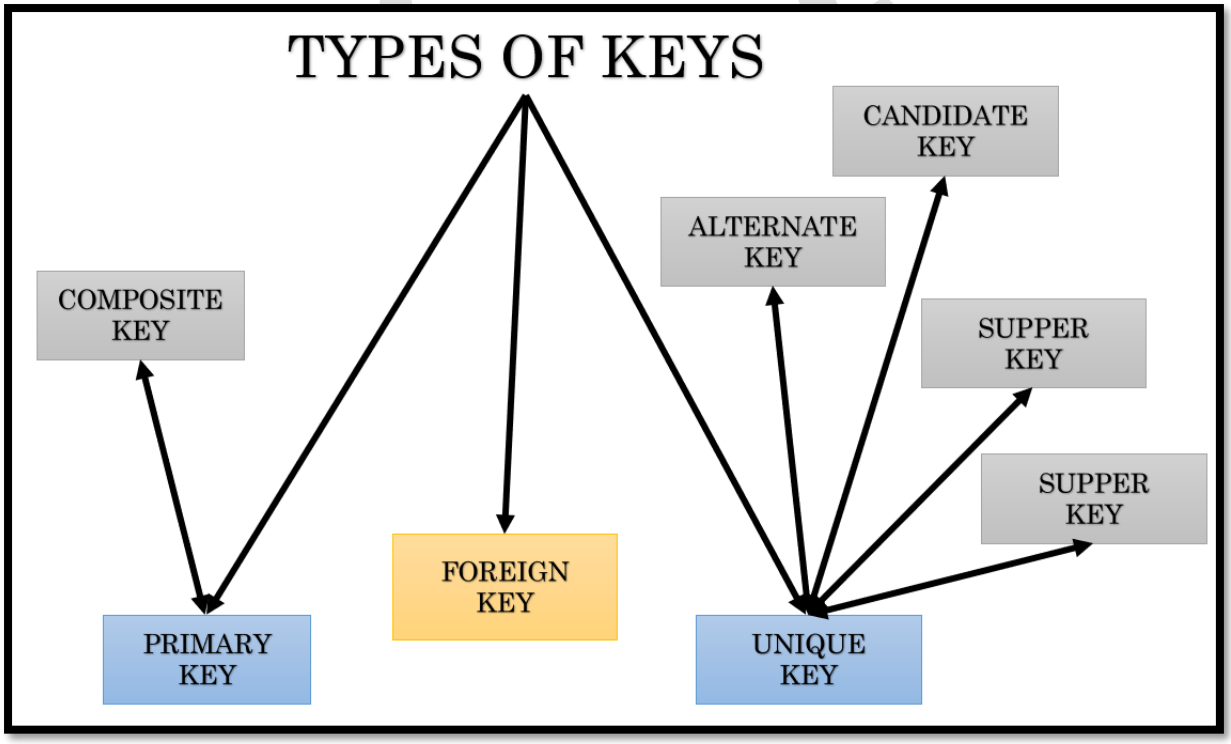


Figures	Represents	Symbols
Entity Set 	Strong Entity Set	
	Weak Entity Set	
Attributes 	Simple Attribute	
	Composite Attribute	
	Single-valued Attribute	
	Multivalued Attribute	
	Derived Attribute	
	Null Attribute	
Relationship	Strong Relationship	
	Weak Relationship	

KEYS

A key is a set of one or more attributes, which is used to uniquely identify within a table





Roll.no	Name	Dob	Mobile	Email	Address
1	Abhishek	09-05-2001	7856985624	xyzj@gmail.com	Delhi
2	Vijay	01-08-2000	8824414632	xyzd@gmail.com	Mumbai
3	Ajay	06-09-2001	7856248218	xyzs@gmail.com	Howrah
4	Satish	17-03-1998	9912542482	xyzq@gmail.com	Ayodhya
5	Navin	16-04-1999	7812545552	xyjj@gmail.com	Jodhpur
6	Suraj	07-12-1997	8615236524	xzjk@gmail.com	Banaras
7	Suneel	25-11-1993	8812458795	yz@gmail.com	Jaipur

ENTITY

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Strong → **Key Attributes**

Week → **Attributes**

STRONG ENTITY

A super key is a set of one or more attributes that uniquely identify each record within a table.

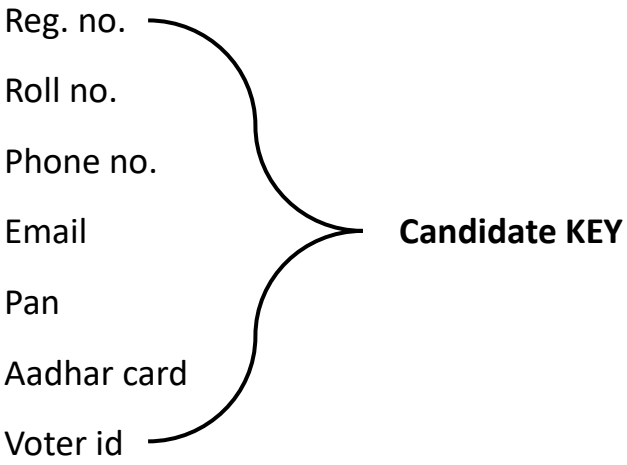
Roll No.
Name
Dob
Mobile
Email

Super Key

Candidate KEY

```
CREATE TABLE Student (
    Roll_no INT UNIQUE,
    email VARCHAR(50) UNIQUE,
    phone VARCHAR(20),
    name VARCHAR(50),
    PRIMARY KEY (Roll_no)
        );
```

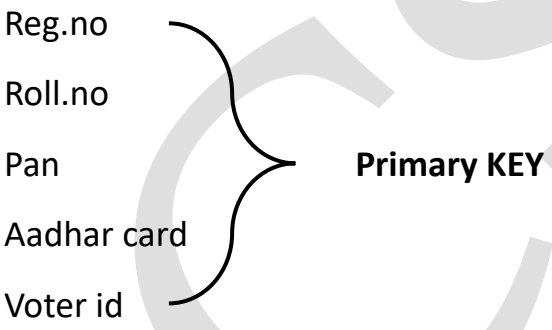
It consists of maximum possible attributes, which is Uniquely identify.




```
CREATE TABLE student (
        roll_no INT UNIQUE,
        email VARCHAR(100),
        name VARCHAR(50)
            );
```

Primary Key

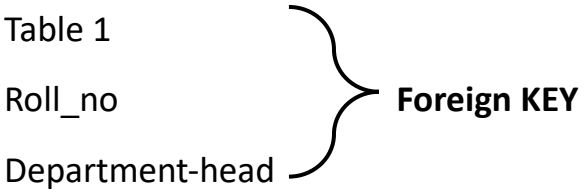
Unique + Not Null




```
CREATE TABLE T1 (
  Roll_no INT PRIMARY KEY,
   name VARCHAR(50),
   phone bigint
        );
```

Foreign Key

A foreign key are attributes in a table, whose value match as primary key in another table.




```
CREATE TABLE T1 ( Roll_no INT PRIMARY KEY, Name VARCHAR(50) );

CREATE TABLE T2 (Roll_no INT PRIMARY KEY, Name VARCHAR(50), Roll_no INT, FOREIGN KEY (Roll_no)

REFERENCES T1 (Roll_no)

        );
```

Composite Key

Two or more columns together form the primary key.

Student_id
Course _id
Enrollment _date

Composite Key

```
CREATE TABLE enrollment (
    Student_id INT,
    Course_id INT,
    enrollment_date DATE,
    PRIMARY KEY (Student_id,
Course_id)
        );
```

Alternate Key

Two or more columns together form the primary key.

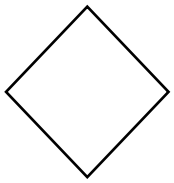
Roll_no
Name
Email
Phone

Alternat Key

```
CREATE TABLE employee (
        Roll_no INT ,
        email VARCHAR(50) UNIQUE,
        phone VARCHAR(20) UNIQUE,
        name VARCHAR(50),
        PRIMARY KEY (Roll_no)
            );
```

RELATIONSHIP

A Relationship Type represents the association between entity types.



A Relationship Type represents the association between entity types.

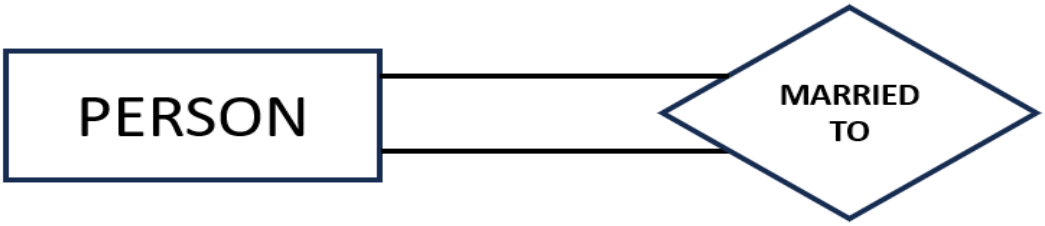
DEGREE OF A RELATIONSHIP SET

The number of different entity sets participating in a relationship set is called the degree of a relationship set.

- Unary Relationship
- Binary Relationship
- Ternary Relationship
- N-ary Relationship

Unary Relationship

When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



Binary Relationship

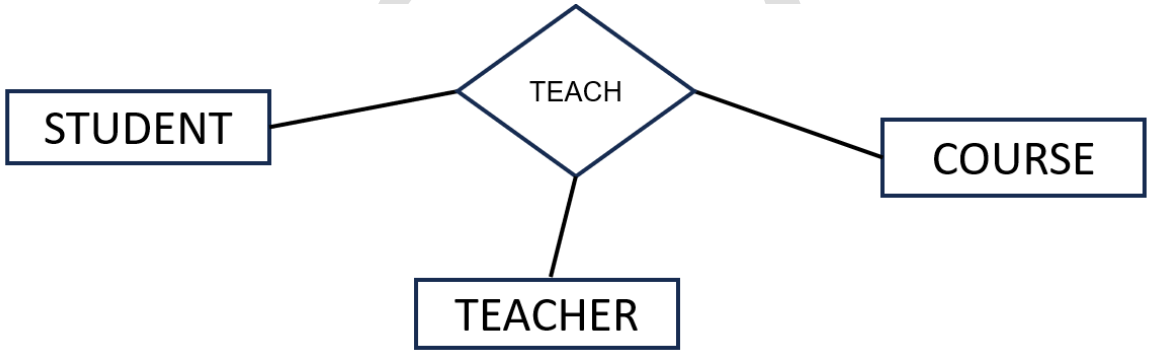
When there are TWO entities set participating in a relationship, the relationship is called a binary relationship.

For example, a Student is enrolled in a Course.

Ternary Relationship

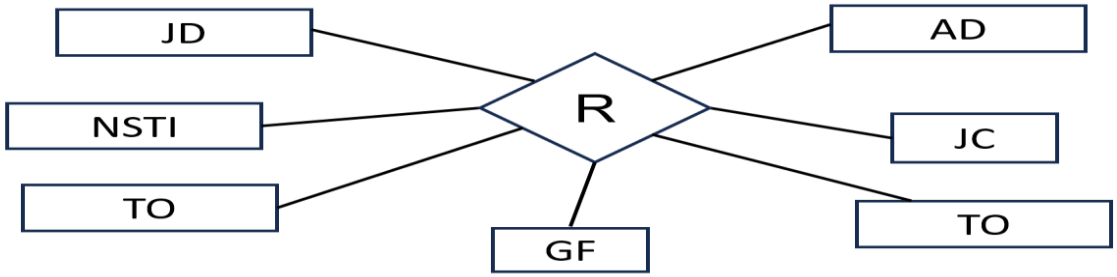
In the Ternary relationship, there are three types of entity associates.

So, we can say that a Ternary relationship exists when there are three types of entity and we call them a degree of Relationship.



N-ary Relationship

When there are n entities set participating in a relation, the relationship is called an n-ary relationship.



RELATIONSHIP 

CARDINALITY

Defines the numerical attributes of the relationship between two entities or entity sets.

- ONE TO ONE
- ONE TO MANY
- MANY TO ONE
- MANY TO MANY

CARDINALITY 

ONE TO ONE

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.

STUDENT

Roll.no	Name
1	Abhishek
2	Vijay
3	Ajay

AADHAR

ROLL.NO.	AADHAR NO.
1	111111111111
2	222222222222
3	333333333333

ONE TO MANY

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.

COSTMER

C.ID	Name	Mobile
1	Abhishek	7856985624
2	Vijay	8824414632
3	Ajay	7856248218

ORDER

C.ID	ORDER	SLOT 1
1	LAPTOP	09-01-2024
1	MOBILE	07-01-2024
1	BOOK	06-01-2024

MANY TO ONE

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.

STUDENT

COLLAGE CODE	Name	Mobile
E1006	Abhishek	7856985624
E1006	Vijay	8824414632
E1006	Ajay	7856248218

COLLEGE

ROLL.NO	ORDER	TRADE
E1006	NSTI HIWRAH	ELECTRICIAN
N103	NSTI KANPUR	CSA
N104	NSTI MUMBAI	FITTER

MANY TO MANY

When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationships.

ACTOR

ACTOR ID	Name
A1	Sahid Kapoor
A2	Ranvir Singh
A3	Ajay Devgan

ASSIGN

ACTOR ID	MOVIE ID
A1	M1
A2	M2
A3	M3
A1	M3
A2	M3

MOVIE

MOVIE ID	MOVIE NAME
M1	Jab we Met
M2	Padmavat
M3	Housefull 4

NORMALIZATION

It's a technique of organizing the data into multiple related tables, To minimize DATA REDUNDANCY

WHAT IS DATA REDUNDANCY

- DATA REDUNDANCY is nothing but repetition of similar data at multiple places.
- Repetition of data increases the size of databases.
- Other issues like

- Insertion problem
- Deletion problem
- Updation problem

STUDENT

Name	**Father name**	Trade	**T.O**
Rajes	A	CSA	Atanu ghosh
Jaswant	A	CSA	Atanu ghosh
Sorav	B	CSA	Atanu ghosh
Rajesh	C	CSA	Atanu ghosh
Manmohan	C	CSA	Atanu ghosh
Hira	D	CSA	Atanu ghosh
Aarrav	E	CSA	Atanu ghosh

- Insertion problem
- Deletion problem
- Updation problem

STUDENT

Name	Father name	Trade	T.O
Rajes	A	CSA	Atanu ghosh
Jaswant	A	CSA	Atanu ghosh
Sorav	B	CSA	Atanu ghosh
Rajesh	C	CSA	Atanu ghosh
Manmohan	C	CSA	Atanu ghosh
Hira	D	CSA	Atanu ghosh
Aarrav	E	CSA	Atanu ghosh

Un-normalize data

1NF –First normal form

2NF – Second normal form

3NF – Third normal form

BCNF – Boyce normal form

4NF -Fourth normal form

5NF -Fifth normal form

1NF

There are 4 rules

➢ Rule 1

- Each column should contain atomic values.

Column 1	Column 2
A	A,B
B	B, C
C	C, D
D	D, E

Entries like x,y and a,b violate this rule

Column 1	Column 2
A	B
B	A
C	D
D	C

1NF

➢ Rule 2

- A column should contain values that are of the same type.

Name	dob
A	20-12-99
B	12-10-98
16-11-97	15-02-99
D	A

Do not-mix defferent types of values in any column

Name	dob
A	20-12-99
B	12-10-98
C	15-02-99
D	16-10-98

1NF

➢ Rule 3

- Each column should have a unique name.

Name	Name
A	B
C	C
E	D
F	G

Same name leads to confusion at the time of data retrieval

F_Name	L-Name
A	B
C	C
E	D
F	G

1NF

➢ Rule 4

- Order in which data is saved doesn't matter.

Sl.no	F_Name	L-Name
3	A	B
5	C	C
4	E	D
1	F	G
2	H	I

Using SQL query, you can easily fetch data in any order from a table

Example

Roll_no	Name	Subject
101	Aarav	Math, English
102	Abhishek	English, Hindi
103	Ved	Math, Hindi
104	Ram	English

After converting 1NF

Roll_no	Name	Subject
101	Aarav	Math
102	Abhishek	English
103	Ved	Hindi
104	Ram	English
101	Aarav	English

102	Abhishek	Hindi
103	Ved	Math

2NF

There are 2 rules

- **Rule 1**
  - It should be in 1st Normal form

Roll_no	Name	Subject
101	Aarav	Math
102	Abhishek	English
103	Ved	Hindi
104	Ram	English
101	Aarav	English
102	Abhishek	Hindi
103	Ved	Math

2NF

- **Rule 2**
  - It should not have any partial dependencies

StudentID	CourseID	StudentName	CourseName
101	C001	ABHISHEK	CSA
102	C002	ASHUTOSH	ELECTRICIAN
101	C002	RISHI	FITTER

After 2NF

Student_ID	Student name
101	Ashutosh
102	Rishi
103	Raju

Course_ID	Course_Name
C001	CSA
C002	Electrician

Student_ID	Course_ID
101	C001
102	C002
101	C002

3NF

There are 2 rules

Rule 1:- It should be in 2nd Normal form

Rule 2:- It should not have Transitive Dependency

3NF

Transitive Dependency

Roll_no	Name	Age	Marks	Grade
101	Aarav	26	91	A
102	Abhishek	25	89	A
103	Ved	37	55	B
104	Ram	21	39	C

3NF

Remove Transitive Dependency

Roll_no	Name	Age	Marks
101	Aarav	26	91
102	Abhishek	25	89
103	Ved	37	55
104	Ram	21	39

Roll_no	Grade
101	A
102	A
103	B
104	C

BCNF (Boyce-Codd Normal Form)

There are 2 rules

Rule 1:- It should be in 3rd Normal form

Rule 2:- For any dependency A→B, A should be a super key

Student	Course	Teacher
A	DBMS	Satyam
B	JS	Aarav
A	JS	Aarav

Teacher	Course
Satyam	DBMS
Aarav	JS

Student	Teacher
A	Satyam
B	Aarav

4NF

There are 2 rules

Rule 1:- It should be in BCNF Normal form

Rule 2:- It should not have Multi-Valued Dependency

Student	Hobby	Skill
Aarav	Cricket	java
Aarav	Music	python
Aarav	Cricket	java
Aarav	Music	python

AFTER 4NF

Student_Name
karan

Student_Name	Student Hobby
karan	cricket
karan	music

Student_Name	skill
karan	java
karan	python

5NF

There are 2 rules

Rule 1:- It should be in 4NF

Rule 2:- It should not have Non-trivial join dependency

Student	Course	Teacher
Karan	DBMS	Sharma
Karan	DBMS	Mehta
Rahul	OS	Sharma

StudentID	Name
pk	

Course id	Title
pk	

Teacher id	Teacher name
pk	

StudentID	Course id	Teacher id
pk	pk	pk
fk	fk	fk

DATA TYPES

DBMS (Database Management System), data types define the kind of data that can be stored in a column of a table. Each data type has a specific range and characteristics.

1. **Numeric Data Types**
2. **Character Data Types**
3. **Date and Time Data Types**
4. **Boolean Data Type**
5. **Binary Data Types**

Numeric Data Types

1. **INTEGER (INT):-** Used to store whole numbers.

   RANGE:- -2147483648 to 2147483648 (4 bytes)

   CREATE TABLE employees (id INT ,age INT);

2. **SMALLINT:-** Stores small integers.

   RANGE:- -32,768 to 32,767 (2 bytes)

   CREATE TABLE small_table (code SMALLINT);

3. **TINYINT:-** Stores very small integers.
   RANGE:- 0 to 255 (1 byte)
   CREATE TABLE flags (is_active TINYINT);

4. **DECIMAL/NUMERIC:-** Stores numbers with a decimal point.
   Range:- User-defined (p = total digits, s = digits after the decimal)
   CREATE TABLE products (price DECIMAL(10, 2));

5. **FLOAT/REAL:-** Stores floating-point numbers (with decimal).
   RANGE:- ±1.17549 × 10^-38 to ±3.40282 × 10^38
   CREATE TABLE measurements (value FLOAT);

Character Data Types

1. **CHAR(N):-** Fixed-length strings.

   Range:- 1 to 255 characters

   CREATE TABLE countries (country_code CHAR(2));

2. **VARCHAR(N):-** Variable-length strings.
   RANGE:- 1 to 65,535 characters (depending on the database system)

- **CREATE TABLE users (username VARCHAR(50), email VARCHAR(100));**

3. **TEXT:-** Stores very large text data.
   RANGE:- Up to 2GB
   CREATE TABLE articles (content TEXT);

Date and Time Data Types

1. **DATE:-** Stores date only.
   RANGE:- 1000-01-01'to' 9999-12-13
   CREATE TABLE events (event_date DATE);

2. TIME:- Stores time only.
   RANGE:- 00:00:00 'to' 23:59:59
   CREATE TABLE schedules (start_time TIME);

3. TIMESTAMP:- Stores date and time in UNIX time (Epoch time)
   RANGE:- 1970-01-01 00:00:01 'UTC to' 2038-01-19 03:14:07 UTC

- **CREATE TABLE logs (created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);**

Boolean Data Type

1. Boolean:- Stores TRUE or FALSE values
   RANGE:- 0 (FALSE) and 1 (TRUE)

- **CREATE TABLE tasks (task_id INT,is_completed BOOLEAN);**

Binary Data Types

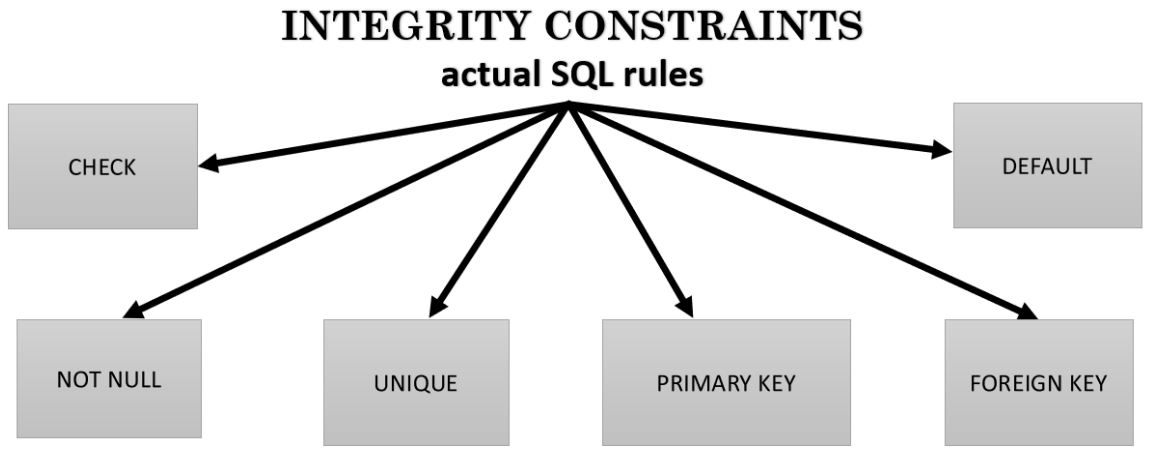
1. BINARY:- Stores fixed-length binary data
   CREATE TABLE secure_data (token BINARY(16));

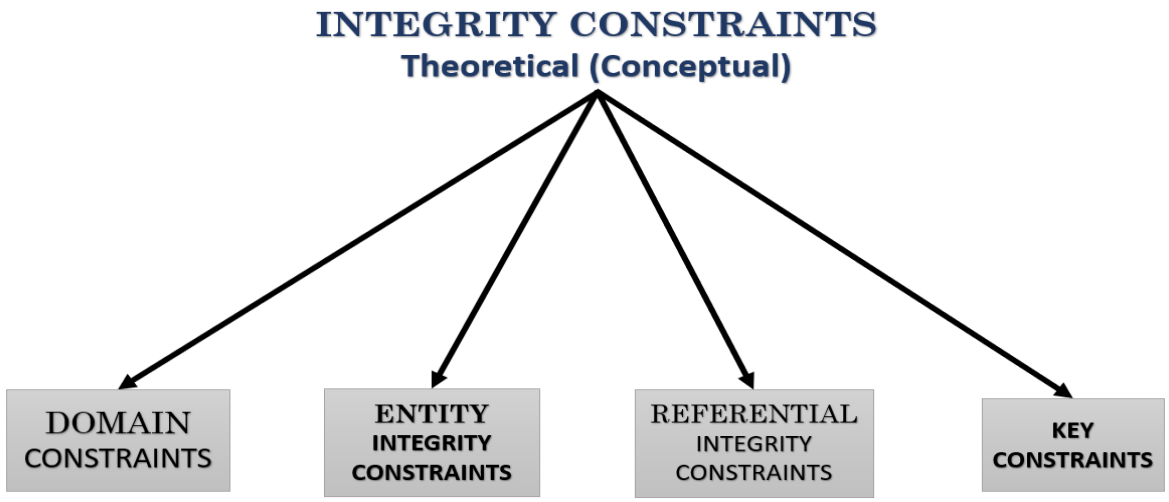
2. VARBINARY:- Stores variable-length binary data.
   CREATE TABLE secure_data (hash VARBINARY(64));

3. BLOB:- Stores large binary data (e.g., images, videos).
   RANGE:- Up to 2GB
   CREATE TABLE documents (doc_name VARCHAR(100),doc_data BLOB);

INTEGRITY CONSTRAINTS

1. Integrity constraints are rules defined in a database to maintain data Accuracy, consistency and reliability.
2. They ensure that data stored in the database follows predefined rules or conditions.





DOMAIN CONSTRAINTS

Domain constraints are related to attributes

ENTITY INTEGRITY CONSTRAINTS

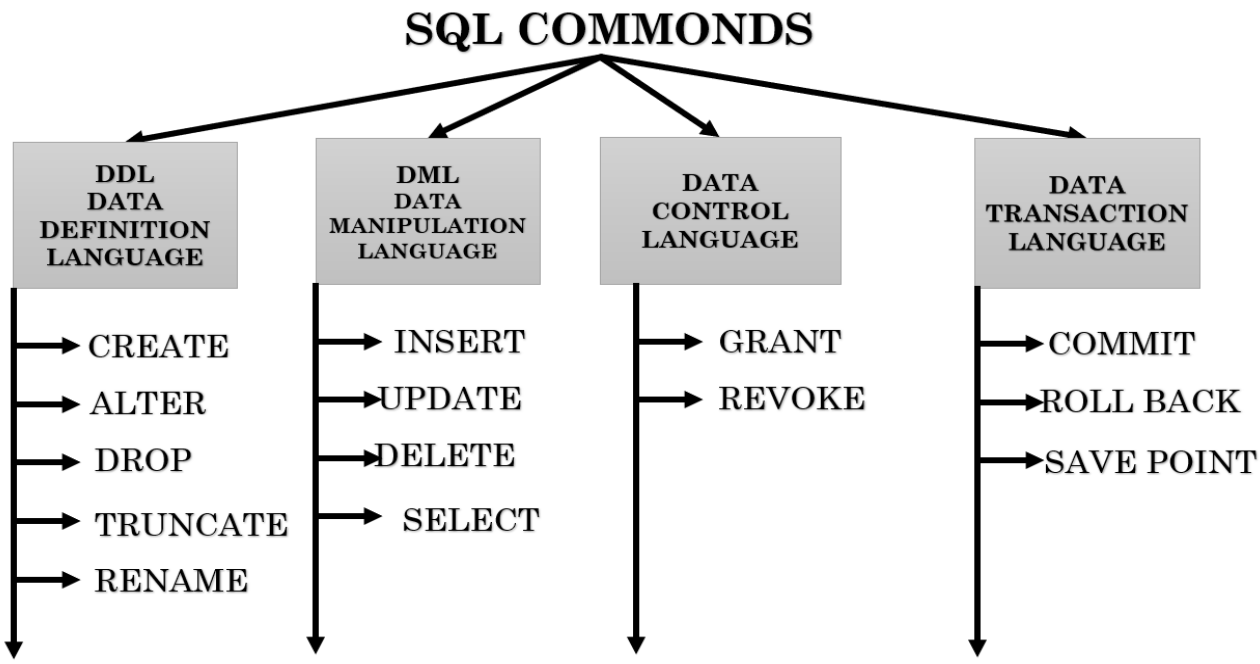
1. Entity integrity constraints are related to the primary key
2. In the table minimum one attribute are primary key

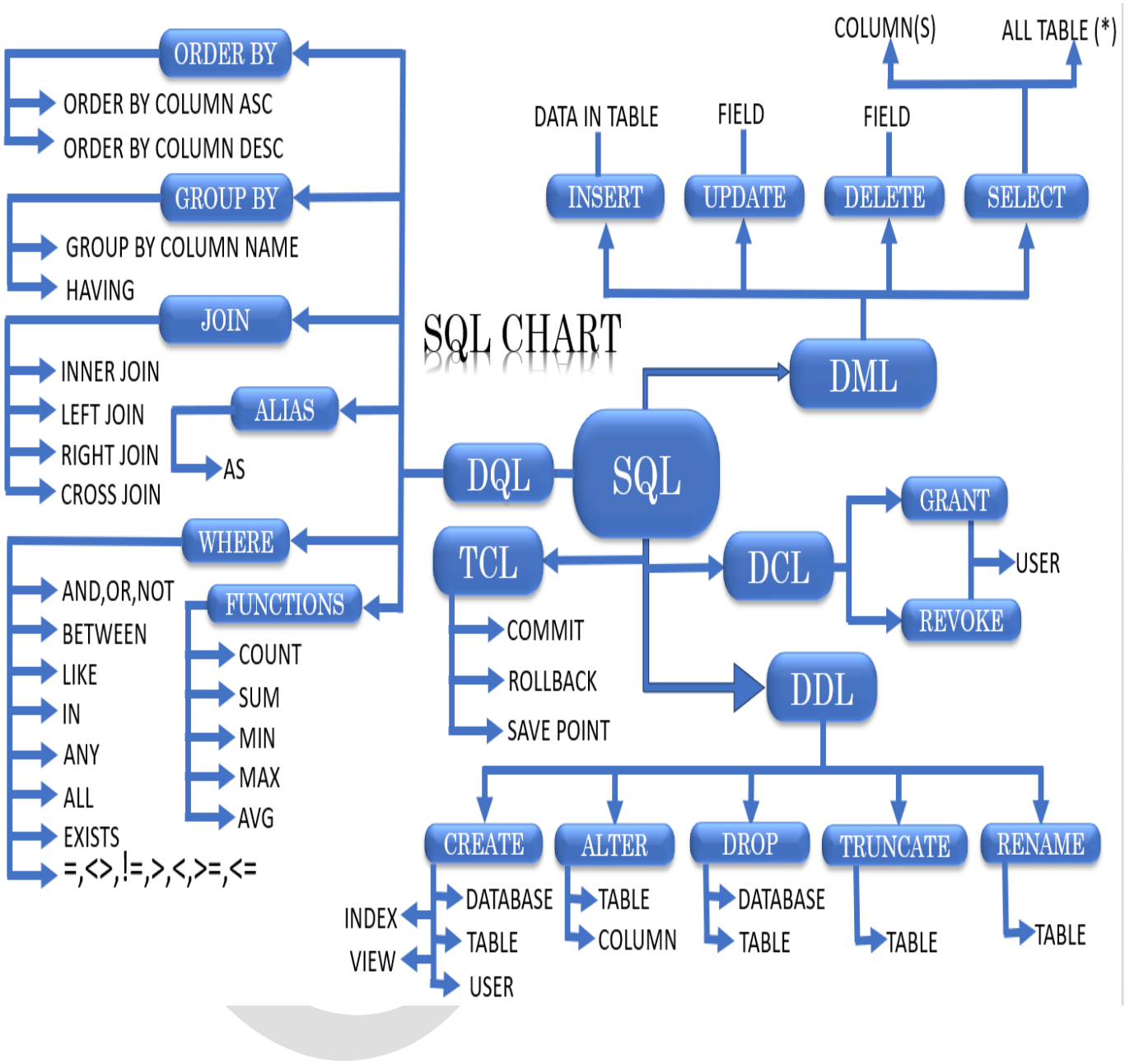
REFERENTIAL INTEGRITY CONSTRAINTS

- Referential integrity constraints is related to the foreign key
- In the table minimum one attribute a foreign key

SQL (STRUCTURE QUERY LANGUAGE)

- SQL is a domain-specific language
- SQL is a declarative language
- DDL, DML, DCL, TCL,





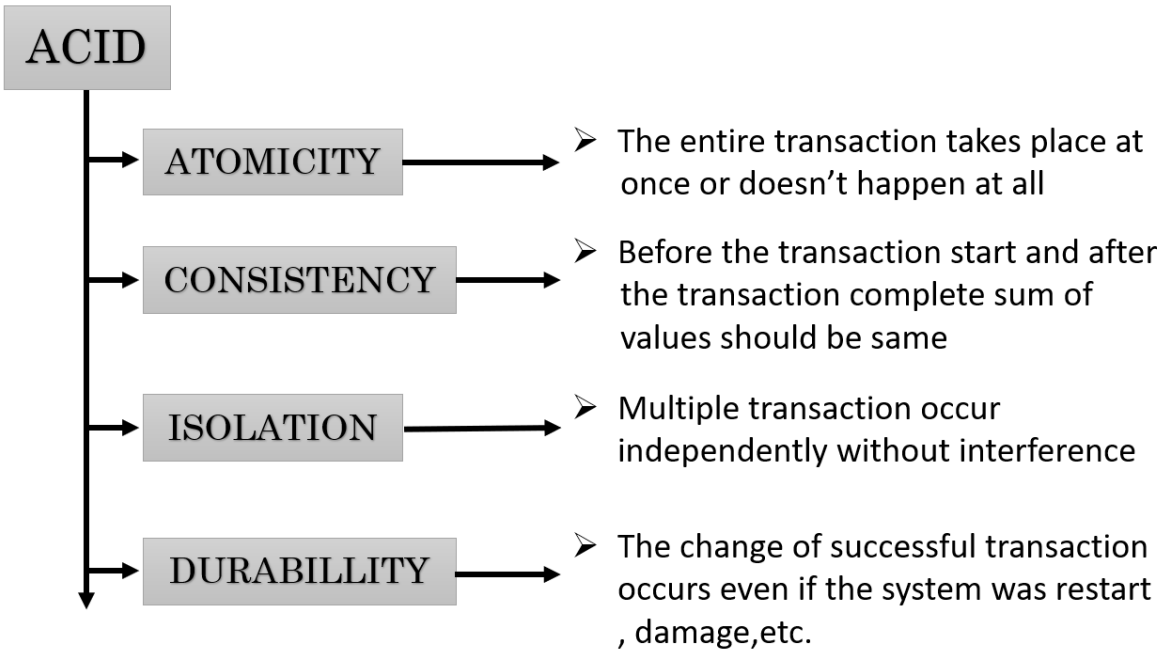
CONCEPT OF TRANSACTIONS

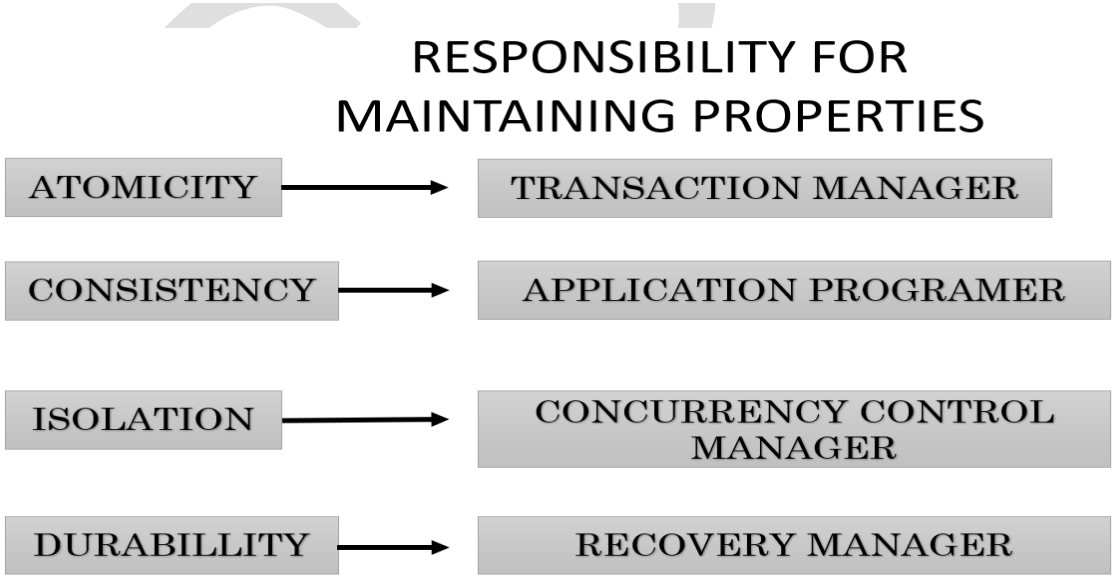
- It is a set of operations used to perform a logical unit of work.
- A transaction is generally represents changes values in the database.

PROPERTIES OF TRANSACTIONS

- Transactions have the following four standard properties,

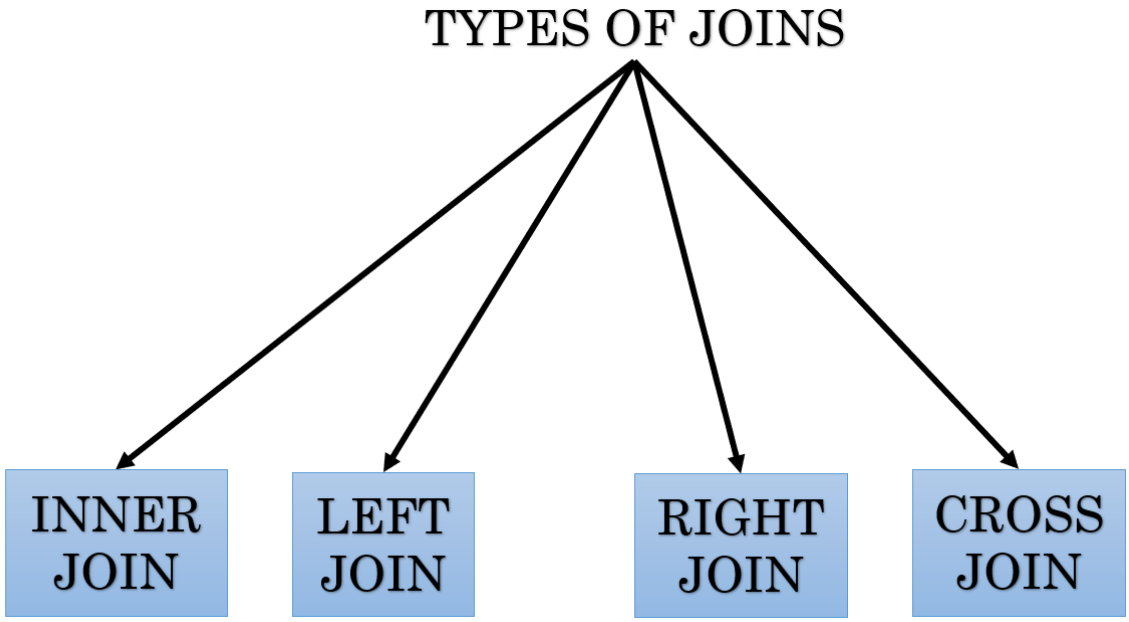
It is referred by ACID.





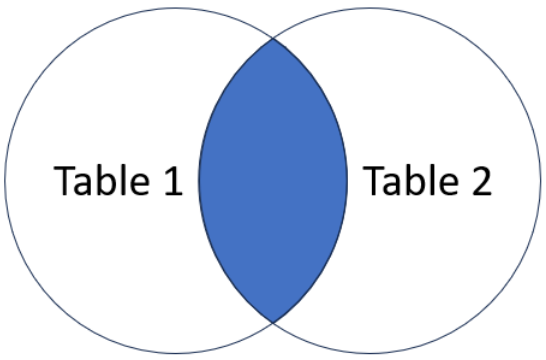
JOINING TABLES

A join clause is used to combine row from two or more tables, based on a related column between them.



INNER JOIN

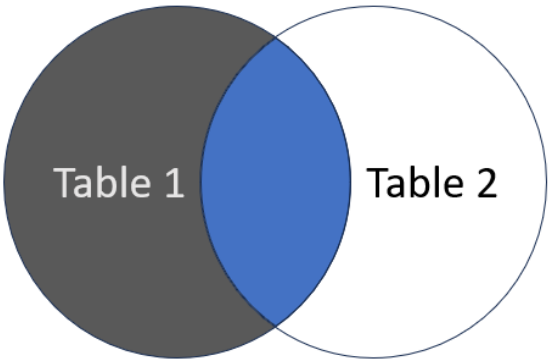
- Returns all records that have matching values in both tables



- Syntex- SELECT ‘column name’ FROM ‘table1’ INNER JOIN ‘table2’ ON table1.column=table2.column;

LEFT JOIN

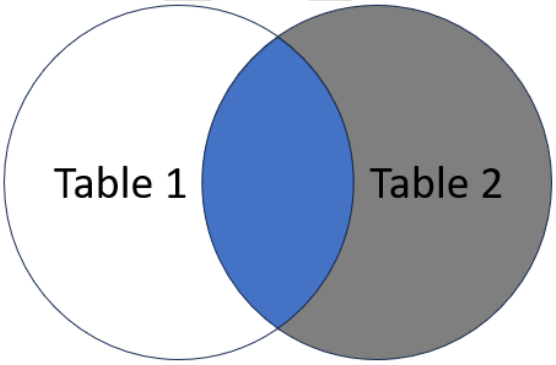
- Returns all records from the left table And the matched records from the right table



- Syntex- SELECT ‘column name’ FROM ‘table1’LEFT JOIN ‘table2’ ON table1.column=table2.column;

RIGHT JOIN

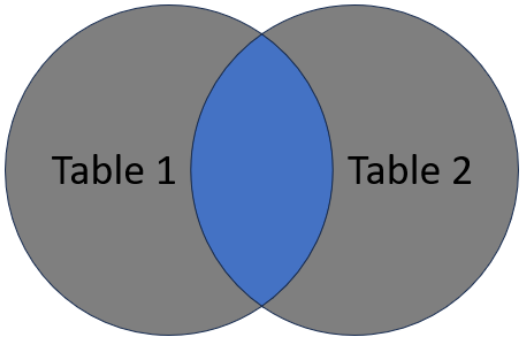
- Returns all records from the right table And the matched records from the left table



- **Syntex:-** SELECT ‘column name’ FROM ‘table1’RIGHT JOIN ‘table2’ ON table1.column=table2.column;

CROSS JOIN

- Returns all records from both table



- **Syntex:-** SELECT 'column name' FROM 'table1'CROSS JOIN 'table2' ON table1.column=table2.column;

Student

Roll_no	name	age	mobile
1	arun	26	8854585685
2	prabhakar	21	8547896589
3	rajesh	23	7885478546
4	kundan	25	8845874582
5	vikrant	21	8654785412
6	ram	22	7345685698

Address

Roll_no	address	email
1	kolkata	ab@gmail.com
2	delhi	cd@gmail.com
3	up	ef@gmail.com
4	mumbai	gh@gmail.com
5	bihar	ij@gmail.com
	rajsthan	kl@gmail.com

Trade

Roll_no	trade
1	csa
2	electrician
3	fitter
4	welder
5	plumber
7	it

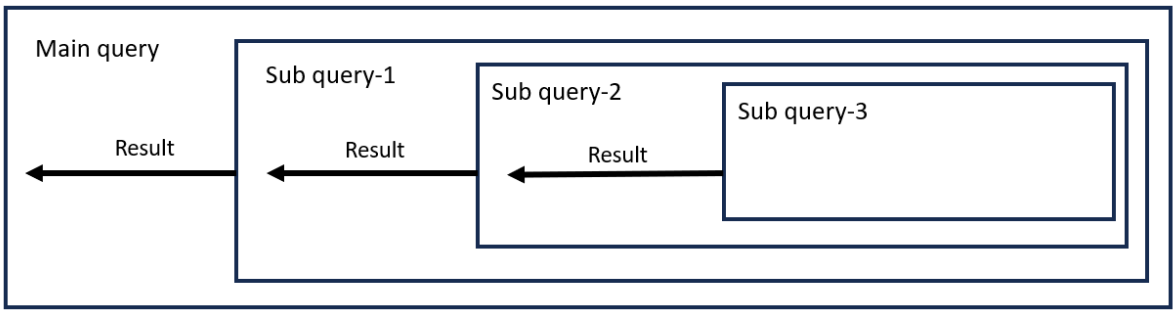
SUB QUERY

- A subquery or inner query or nested query allows us to create a complex query on the Output of another query.
- Sub query syntax involves two select statements.

RULES

- A Sub query must be enclosed with in parentheses.
- You can use the comparison operators such as

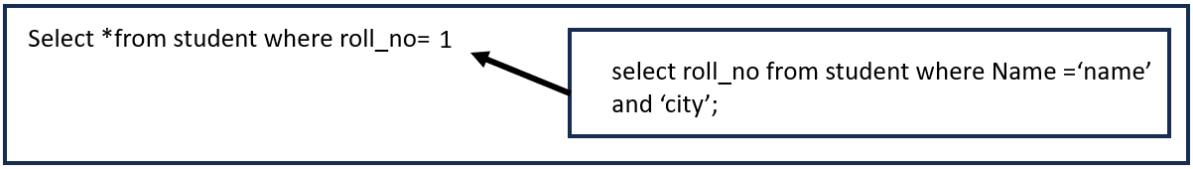
  >,<,=,IN,ANY,ALL



Select *from student where roll_no=1;

select roll_no from student where Name =‘name’ and ‘city’;

Select *from student where roll_no in (select roll_no from student where Name =‘name’ and ‘city’);



TYPE OF SUBQUERY

- SINGLE ROW
- MULTIPLE ROW
- CORRELATED SUBQUERY

Single row sub query

- Return zero or one row
- A sub query must be return a single row value. An equal (=) operator is used for a single row return

Select *from student where roll_no= 1

select roll_no from student where Name ='value' and NAME ='city';

Multiple row subquery

- Return one or more rows
- A (IN) operator is used for multiple rows

Select *from student where roll_no IN1,3,5

select roll_no from student where Name ='city';

Correlated subquery

- Return the subquery depends on the outer query for its values
- Correlated subqueries are used for Row-by-Row processing. Each subquery is executed once for every row of the outer query.

roll_no	name	address
1	sunil	kolkata
2	ajay	mumbai
3	vijay	kanpur
4	manoj	bihar
5	geeta	rajsthan
6	sita	chennai
7	sunil	kolkata
8	vijay	bihar
9	sunil	mumbai
11	arvind	mumbai
12	raju	up
13	reta	up
14	anil	rajsthan
16	shyam	rajsthan
17	ram	up

roll_no	science	math	english	computer	total
1	87	89	91	91	358
2	91	96	93	91	361
3	96	93	91	83	363
4	78	83	90	81	332
5	81	98	93	80	352
6	98	97	89	83	367
7	97	92	93	86	368
8	83	82	73	96	334
9	93	72	83	76	324
10	73	92	76	86	327
11	71	83	86	96	336
12	81	93	96	66	NULL
13	61	98	91	99	NULL
14	63	91	75	89	NULL
15	73	81	85	99	NULL

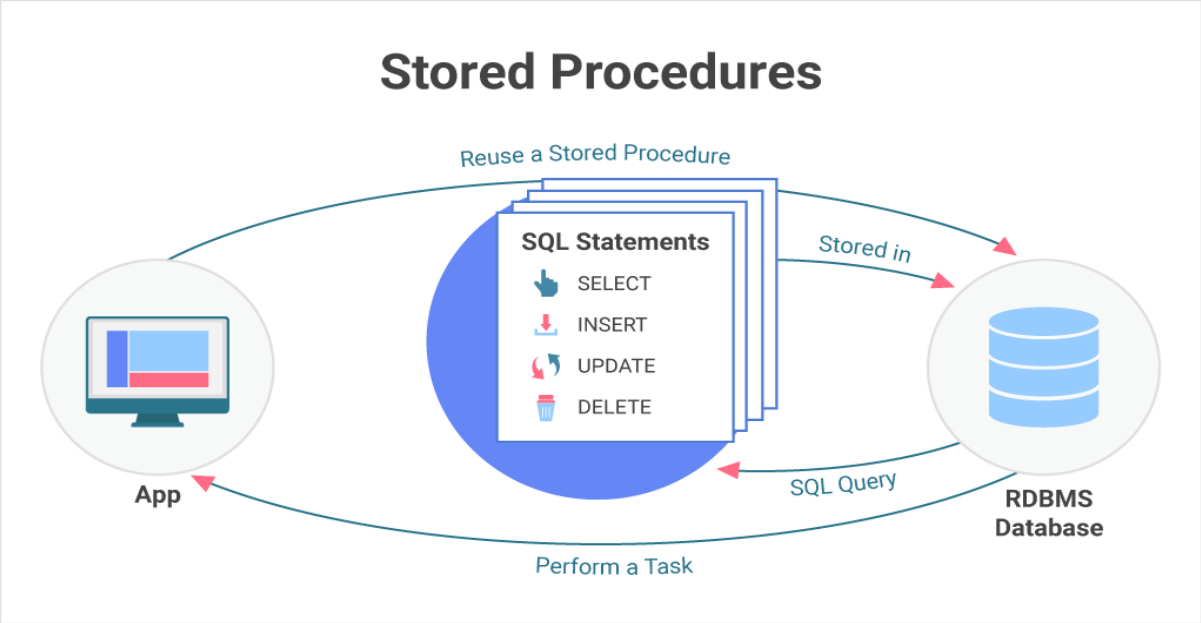
Select student.roll_no,marks.roll_no from student inner join marks on student.roll_no=marks.roll_no where 2=

select count(distinct total) from marks where total >=marks.total;

```
Select student.roll_no,marks.roll_no from student inner join marks
on student.roll_no=marks.roll_no  where 2=
```

```
select count(distinct total) from marks where
    total >=marks.total;
```

Procedure



A **procedure** in **Database Management System** is a **stored set of SQL statements that performs a specific task in the database**. It can be **saved and executed whenever needed**.

A procedure is a saved SQL program used to perform database operations automatically.

Features of a Procedure

1. **Stored in the database** – The procedure is saved inside the DBMS.
2. **Reusable** – It can be executed many times without rewriting the SQL code.
3. **Improves performance** – Since it is precompiled, execution is faster.
4. **Reduces code repetition** – Same logic can be reused in different programs.
5. **Supports parameters** – Input values can be passed to the procedure.

Ex-

```
CREATE PROCEDURE sp_name()
 BEGIN
 SELECT * FROM STUDENT;
 END ;//
```

Call procedure syntax;
Call p_name( )//
Call p_name(type your input)//

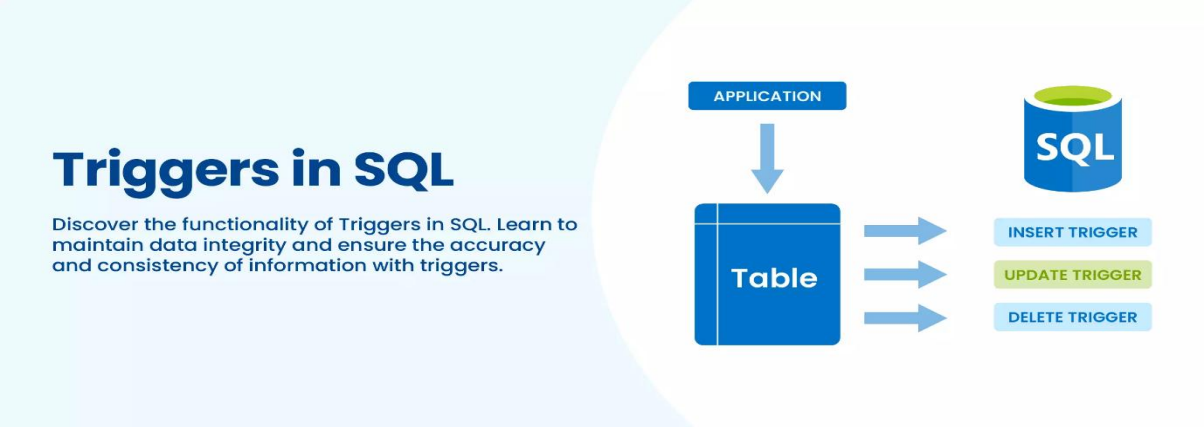
SHOW PROCEDURE
SHOW CREATE PROCEDURE HELLO()//
SHOW PROCEDURE STATUS WHERE Db = 'database Name'//

DROP PROCEDURE
DROP PROCEDURE name;

Ex-

```
Delimiter //
create procedure grade()
      begin
    select roll_no,total,
    case
    when total >= 80 then "A"
    when total >= 60 then "B"
    when total >= 40 then "C"
    else "FAIL"
    end as grade from marks3;
    end //
```

Trigger



- A trigger in Database Management System is a special stored program that automatically runs when an event (INSERT, UPDATE, or DELETE) happens in a table.
- A trigger is a set of SQL statements that automatically executes when a change occurs in a database table.

Feature of Trigger

- **Automatic Execution** – A trigger runs automatically when a specified event occurs in a table.
- **Event-Based** – It works when actions like **INSERT, UPDATE, or DELETE** happen.
- **Stored in Database** – The trigger is saved inside the database.
- **Data Integrity** – Helps maintain accuracy and consistency of data.
- **No Manual Call Needed** – It executes automatically without being called by the user.

Ex-

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name FOR EACH
ROW
 BEGIN
SQL statements;
END //
```

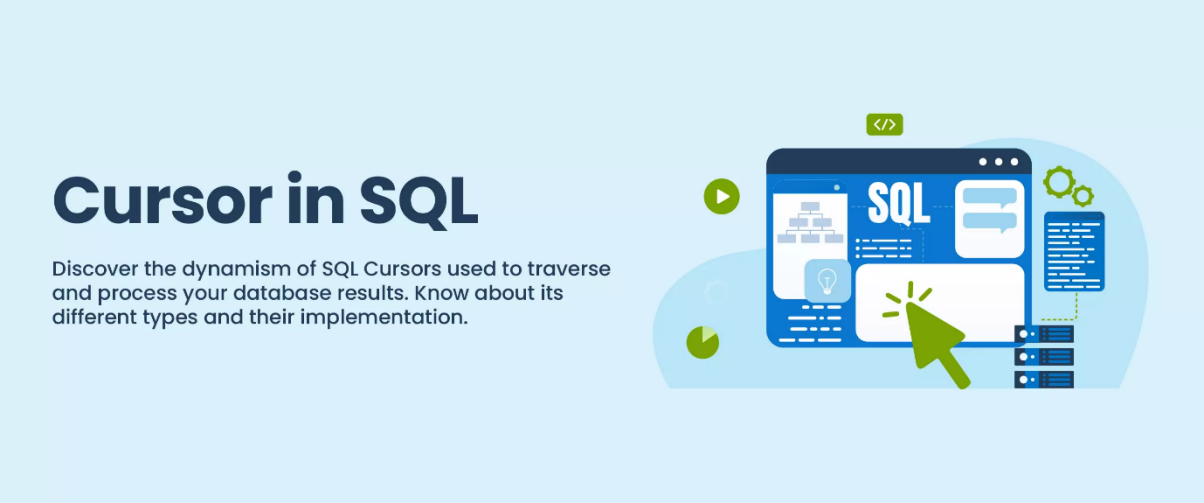
Ex- 2

```
DELIMITER //
CREATE TRIGGER before_insert_marks
BEFORE INSERT ON marks
FOR EACH ROW
BEGIN
  SET NEW.total = NEW.math + NEW.science + NEW.english;
END//
```

Ex- Time and Date

```
DELIMITER //
 CREATE TRIGGER trigger_name
   AFTER INSERT  ON table_name
   FOR EACH ROW
   BEGIN
   INSERT INTO emp_audit VALUES (null, concat('A row is inserted in student table at
', date_format(now(), '%d-%m-%y%h:%i:%s %p')));
  END //
```

Cursor



- A cursor in Database Management System is a database object used to retrieve and process rows of a query one by one.
- A cursor is used to handle query results row by row in a database.

Ex-

```
create procedure cur25()
 begin
 declare c1 varchar(50);
 declare c2 varchar(50);
 declare cursor1 cursor for select name, address from student1;
 open cursor1;
 fetch cursor1 into c1,c2;
 insert into S_table_name (name,address)values(c1,c2);
 close cursor1;
 end//
```