DBMS Interview Questions & Ans

By - shubham Maurya

debugwithshubham debugwithshubham debugwithshubham

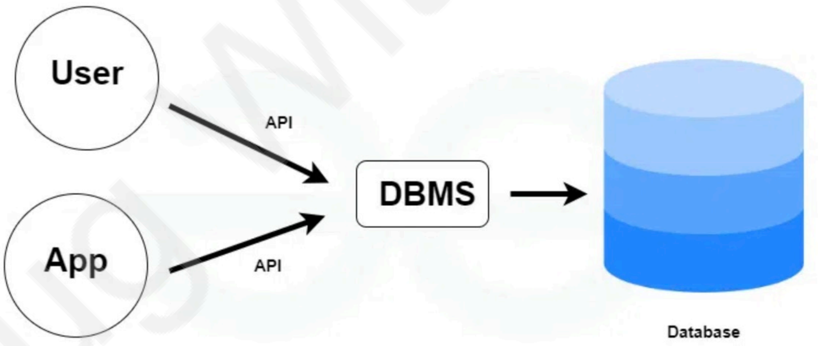
What is DBMS?

Database Management System (DBMS) is a software used to manage data from a database.

- A DBMS is a software that allows to create, update and retrieval of data in an organized ways . It also provides security to the database.

Examples of relational DBMS are MySQL, Oracle, Microsoft SQL Server, Postgre SQL and Snowflake.

Examples of NoSQL DBMS are MongoDB, Cassandra, DynamoDB and Redis.



Components of a DBMS

Data

Database access language

Query language

Management resources:

Query processing

Data Integrity and Security:

Benefits of DBMS

Reduces data redundancy

Ensures data security

Eliminates data inconsistency

Ensures data sharing

Maintains data integrity

Ensures data recovery

Low maintenance cost

Saves time

Allows multiple user interfaces

Organisation of data

Types of DBMS

1. Hierarchical database management system
2. Relational database management system
3. Network database management system
4. Object-oriented database management system
5. NoSQL DBMS

Disadvantages of DBMS

Complexity

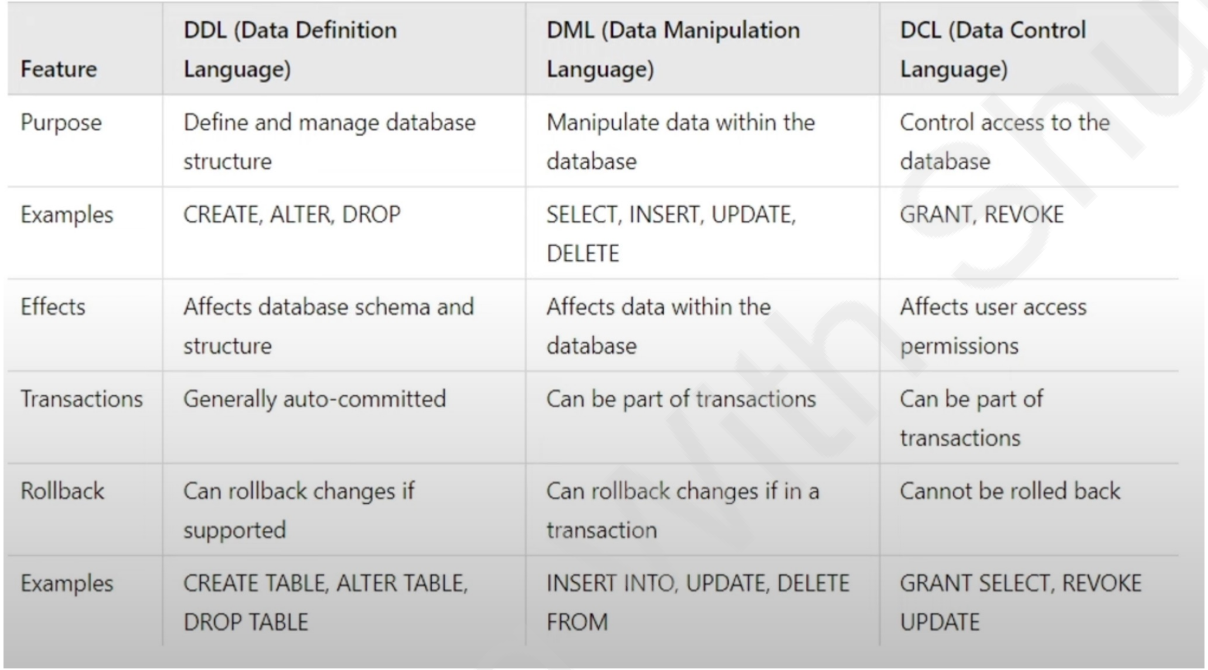
Performance Overhead

Scalability

Cost

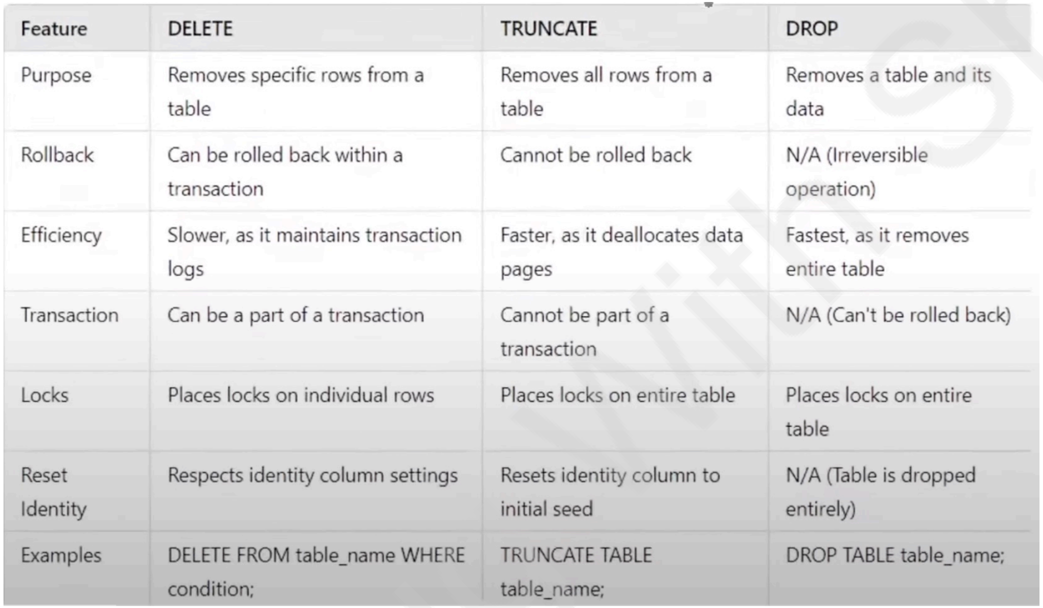
Limited Use Cases

What are DDL, DML and DCL Commands



Feature	DDL (Data Definition Language)	DML (Data Manipulation Language)	DCL (Data Control Language)
Purpose	Define and manage database structure	Manipulate data within the database	Control access to the database
Examples	CREATE, ALTER, DROP	SELECT, INSERT, UPDATE, DELETE	GRANT, REVOKE
Effects	Affects database schema and structure	Affects data within the database	Affects user access permissions
Transactions	Generally auto-committed	Can be part of transactions	Can be part of transactions
Rollback	Can rollback changes if supported	Can rollback changes if in a transaction	Cannot be rolled back
Examples	CREATE TABLE, ALTER TABLE, DROP TABLE	INSERT INTO, UPDATE, DELETE FROM	GRANT SELECT, REVOKE UPDATE

Differentiate between DELETE, TRUNCATE and DROP Commands in SQL

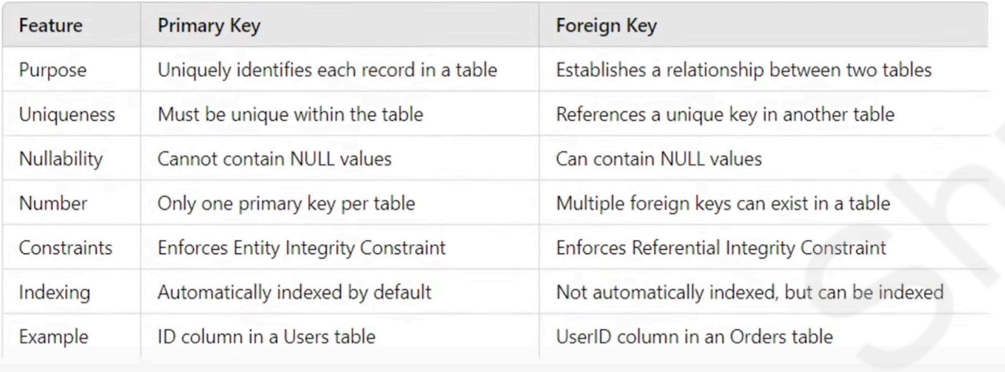


Feature	DELETE	TRUNCATE	DROP
Purpose	Removes specific rows from a table	Removes all rows from a table	Removes a table and its data
Rollback	Can be rolled back within a transaction	Cannot be rolled back	N/A (Irreversible operation)
Efficiency	Slower, as it maintains transaction logs	Faster, as it deallocates data pages	Fastest, as it removes entire table
Transaction	Can be a part of a transaction	Cannot be part of a transaction	N/A (Can't be rolled back)
Locks	Places locks on individual rows	Places locks on entire table	Places locks on entire table
Reset Identity	Respects identity column settings	Resets identity column to initial seed	N/A (Table is dropped entirely)
Examples	DELETE FROM table_name WHERE condition;	TRUNCATE TABLE table_name;	DROP TABLE table_name;

Most Important SQL Commands

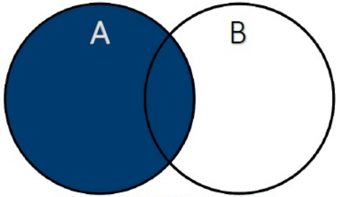
Command	Description
SELECT	Retrieves data from one or more tables.
INSERT	Adds new rows (records) to a table.
UPDATE	Modifies existing data in a table.
DELETE	Removes specific rows from a table.
CREATE TABLE	Creates a new table in the database.
ALTER TABLE	Modifies the structure of an existing table (e.g., add or remove columns).
DROP TABLE	Permanently deletes a table and its data.
TRUNCATE TABLE	Removes all rows from a table but keeps its structure intact.
WHERE	Filters records based on a condition.
ORDER BY	Sorts the result set in ascending or descending order.
GROUP BY	Groups rows that have the same values in specified columns.
HAVING	Filters grouped data (used with `GROUP BY`).
JOIN	Combines rows from two or more tables based on a related column.
DISTINCT	Removes duplicate values from the result set.
IN / BETWEEN / LIKE	Used for advanced filtering conditions.
UNION	Combines the result of two or more `SELECT` queries.
GRANT	Gives user privileges or permissions.
REVOKE	Removes user privileges.
COMMIT	Saves all changes made in the current transaction.
ROLLBACK	Undoes changes if something goes wrong in a transaction.
SAVEPOINT	Sets a point in a transaction to roll back to if needed.

What is the difference between Primary key and Foreign Key.

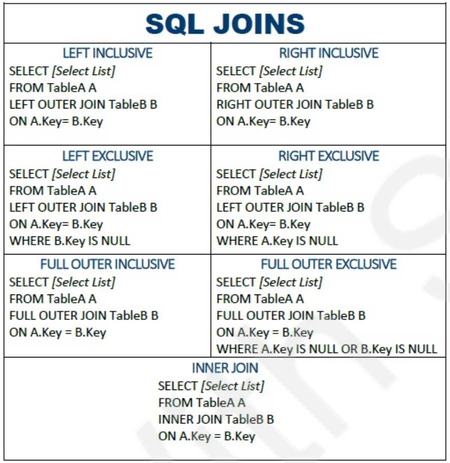


Feature	Primary Key	Foreign Key
Purpose	Uniquely identifies each record in a table	Establishes a relationship between two tables
Uniqueness	Must be unique within the table	References a unique key in another table
Nullability	Cannot contain NULL values	Can contain NULL values
Number	Only one primary key per table	Multiple foreign keys can exist in a table
Constraints	Enforces Entity Integrity Constraint	Enforces Referential Integrity Constraint
Indexing	Automatically indexed by default	Not automatically indexed, but can be indexed
Example	ID column in a Users table	UserID column in an Orders table

What are Different Type of joins in SQL



LEFT INCLUSIVE

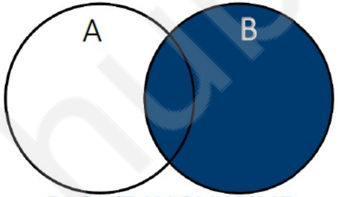


SQL JOINS	
LEFT INCLUSIVE SELECT *[Select List]* FROM TableA A LEFT OUTER JOIN TableB B ON A.Key= B.Key	RIGHT INCLUSIVE SELECT *[Select List]* FROM TableA A RIGHT OUTER JOIN TableB B ON A.Key= B.Key

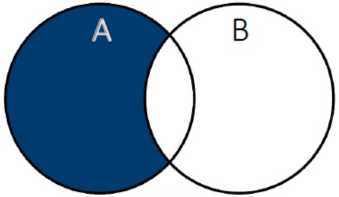
LEFT EXCLUSIVE
SELECT *[Select List]*
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key= B.Key
WHERE B.Key IS NULL
	RIGHT EXCLUSIVE
	SELECT *[Select List]*
	FROM TableA A
	LEFT OUTER JOIN TableB B
	ON A.Key= B.Key
	WHERE A.Key IS NULL

FULL OUTER INCLUSIVE
SELECT *[Select List]*
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
	FULL OUTER EXCLUSIVE
	SELECT *[Select List]*
	FROM TableA A
	FULL OUTER JOIN TableB B
	ON A.Key = B.Key
	WHERE A.Key IS NULL OR B.Key IS NULL

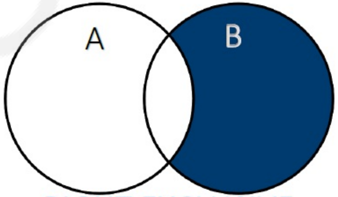
INNER JOIN SELECT *[Select List]* FROM TableA A INNER JOIN TableB B ON A.Key = B.Key	



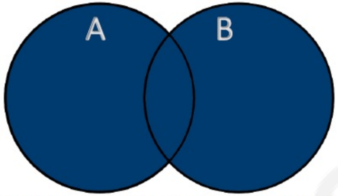
RIGHT INCLUSIVE



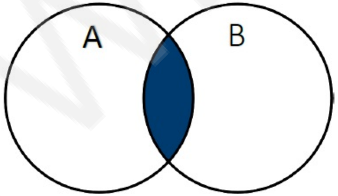
LEFT EXCLUSIVE



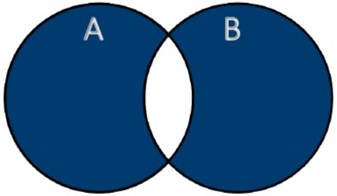
RIGHT EXCLUSIVE



FULL OUTER INCLUSIVE



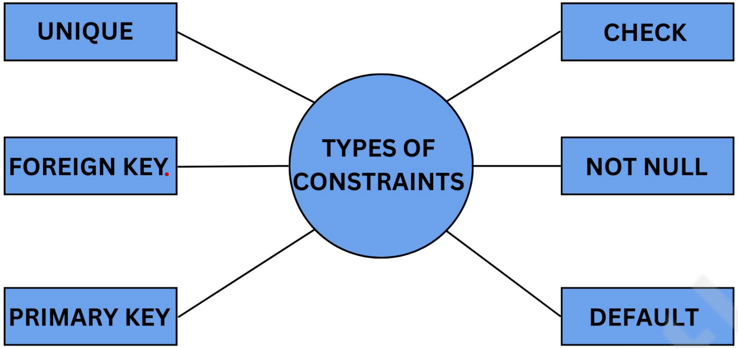
INNER JOIN



FULL OUTER EXCLUSIVE

What are different types of constraints in SQL

CONSTRAINTS IN SQL



TYPES OF CONSTRAINTS

UNIQUE

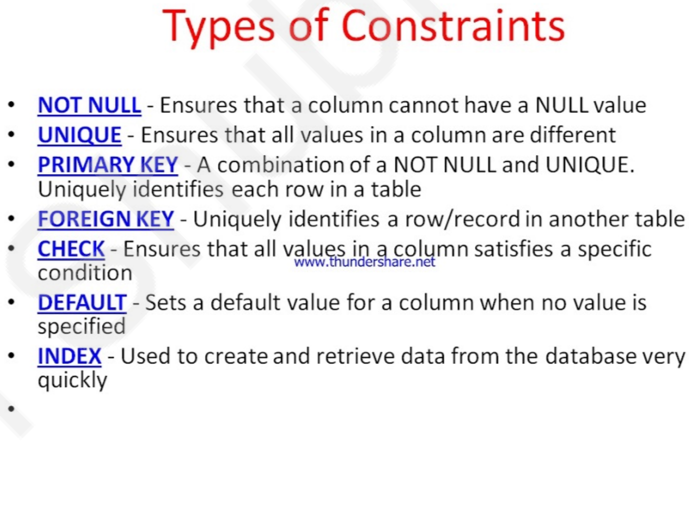
FOREIGN KEY

PRIMARY KEY

CHECK

NOT NULL

DEFAULT



Types of Constraints

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified
- INDEX - Used to create and retrieve data from the database very quickly
-

Define Normalisation, Why do we use it and its types

Normalization in database design is a process of organizing data to reduce redundancy and improve data integrity by minimizing data duplication and dependency.

Why is Normalization Important?

Reduces Data Redundancy:
Improves Data Integrity
Simplifies Database Design
Optimizes Performance

There are different types of normalization

1NF (First Normal Form): all columns contain atomic values , Each row & column is unique, order in which data is stored does not matter.

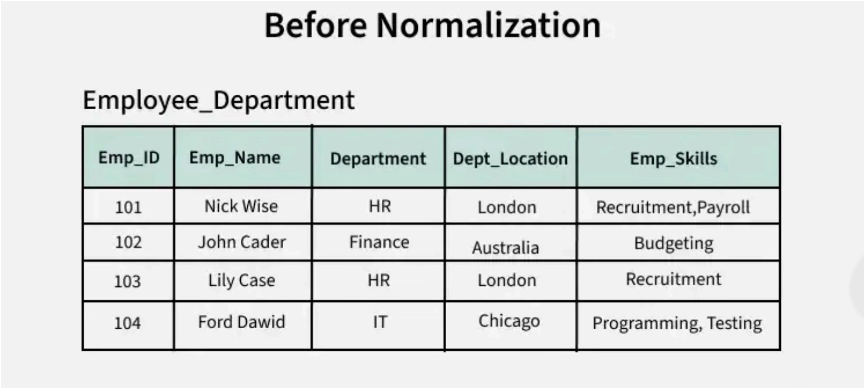
2NF (Second Normal Form): additionally. **No partial dependency** exists,

3NF (Third Normal Form):additionally, there are **no transitive dependencies**.

BCNF (Boyce-Codd Normal Form):is a stricter version of **3NF** where for every **non-trivial functional dependency** (X → Y), **X** must be a **superkey** (a unique identifier for a record in the table).

4NF (Fourth Normal Form):has no **multi-valued dependencies**

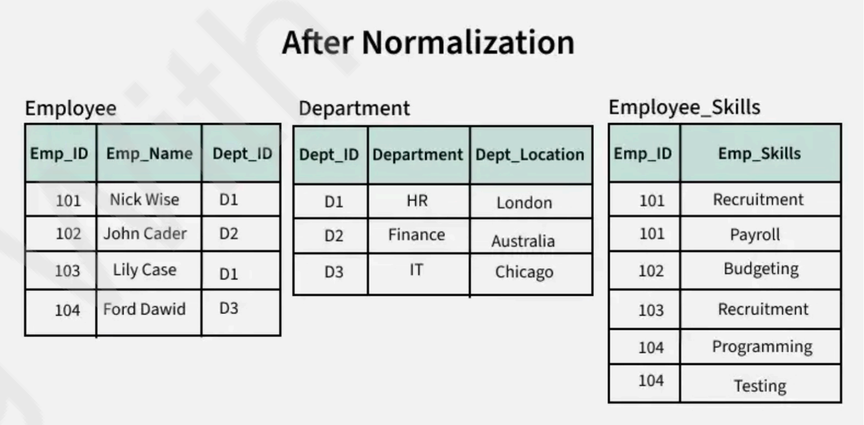
5NF (Fifth Normal Form): all **join dependencies** are removed



Before Normalization

Employee_Department

Emp_ID	Emp_Name	Department	Dept_Location	Emp_Skills
101	Nick Wise	HR	London	Recruitment,Payroll
102	John Cader	Finance	Australia	Budgeting
103	Lily Case	HR	London	Recruitment
104	Ford Dawid	IT	Chicago	Programming, Testing



After Normalization

Employee

Emp_ID	Emp_Name	Dept_ID
101	Nick Wise	D1
102	John Cader	D2
103	Lily Case	D1
104	Ford Dawid	D3

Department

Dept_ID	Department	Dept_Location
D1	HR	London
D2	Finance	Australia
D3	IT	Chicago

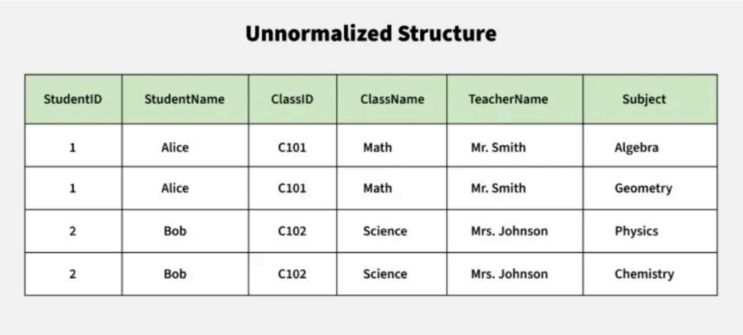
Employee_Skills

Emp_ID	Emp_Skills
101	Recruitment
101	Payroll
102	Budgeting
103	Recruitment
104	Programming
104	Testing

What is denormalizations, and when is it used?

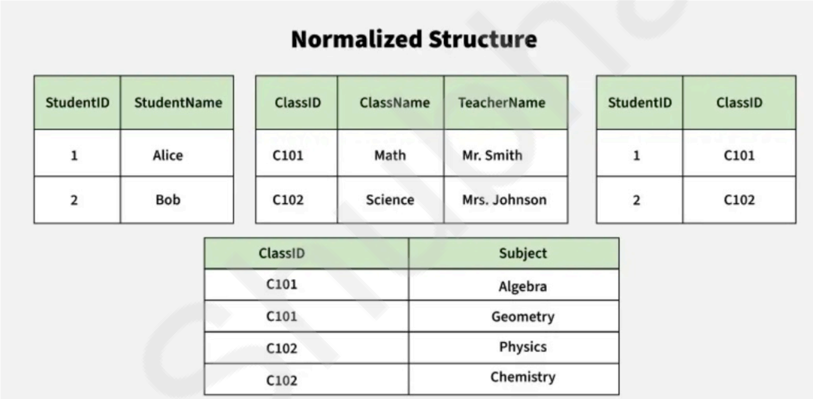
Denormalization is the process of intentionally introducing redundancy into a database design for performance optimization

it use is to improve query performance by reducing the need for joins and simplifying data retrieval



Unnormalized Structure

StudentID	StudentName	ClassID	ClassName	TeacherName	Subject
1	Alice	C101	Math	Mr. Smith	Algebra
1	Alice	C101	Math	Mr. Smith	Geometry
2	Bob	C102	Science	Mrs. Johnson	Physics
2	Bob	C102	Science	Mrs. Johnson	Chemistry



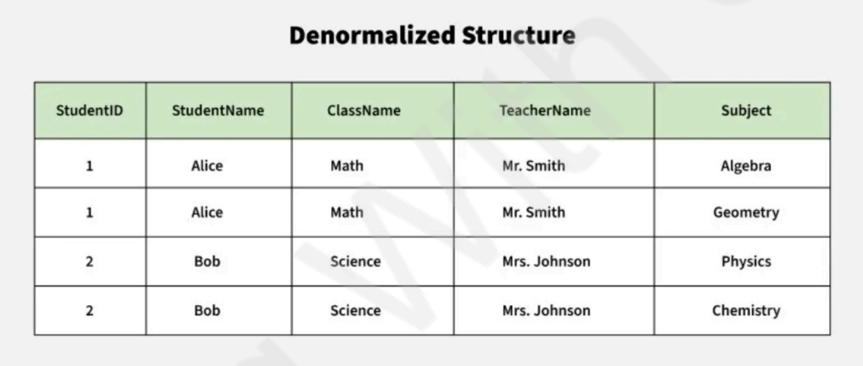
Normalized Structure

StudentID	StudentName
1	Alice
2	Bob

ClassID	ClassName	TeacherName
C101	Math	Mr. Smith
C102	Science	Mrs. Johnson

StudentID	ClassID
1	C101
2	C102

ClassID	Subject
C101	Algebra
C101	Geometry
C102	Physics
C102	Chemistry



Denormalized Structure

StudentID	StudentName	ClassName	TeacherName	Subject
1	Alice	Math	Mr. Smith	Algebra
1	Alice	Math	Mr. Smith	Geometry
2	Bob	Science	Mrs. Johnson	Physics
2	Bob	Science	Mrs. Johnson	Chemistry

What is a transaction in a database

In a Database Management System (DBMS), a transaction is a sequence of operations performed as a single logical unit of work. These operations may involve reading, writing, updating, or deleting data in the database. A transaction is considered complete only if all its operations are successfully executed, Otherwise the transaction must be **rolled back**, ensuring the database remains in a consistent state.

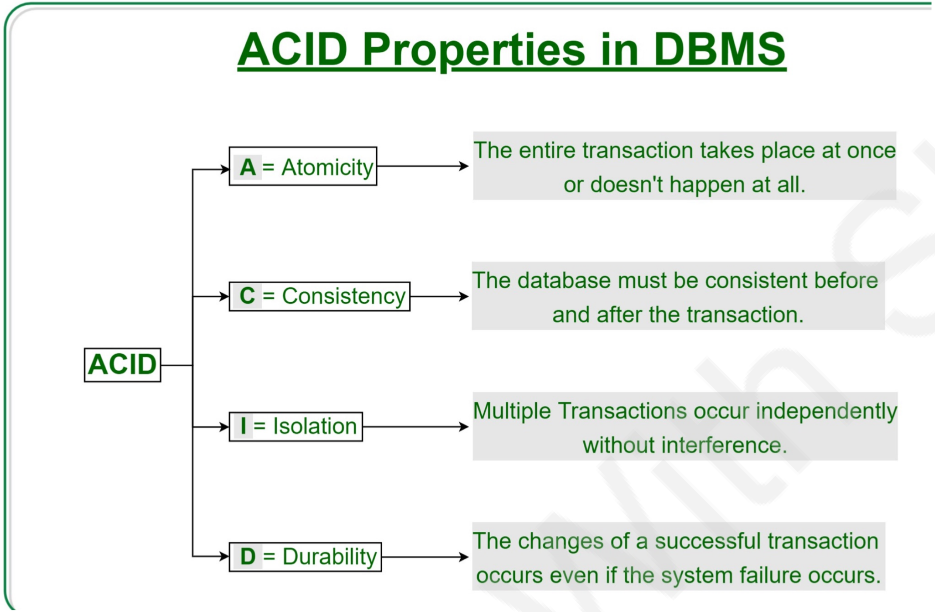
What is indexing, and why is it important

An index in SQL is a schema object that improves the speed of data retrieval operations on a table. It works by creating a separate Data structure that provides pointers to the rows in a table, making it faster to look up rows based on specific column values. Indexes act as a table of contents for a database, allowing the server to locate data quickly and efficiently, reducing disk I/O operations.

Benefits of Indexes:

Faster Queries: Speeds up SELECT and JOIN operations.
Lower Disk I/O: Reduces the load on your database by limiting the amount of data scanned.
Better Performance on Large Tables: Essential when working with millions of records.

What are ACID Properties



ACID Properties in DBMS

ACID

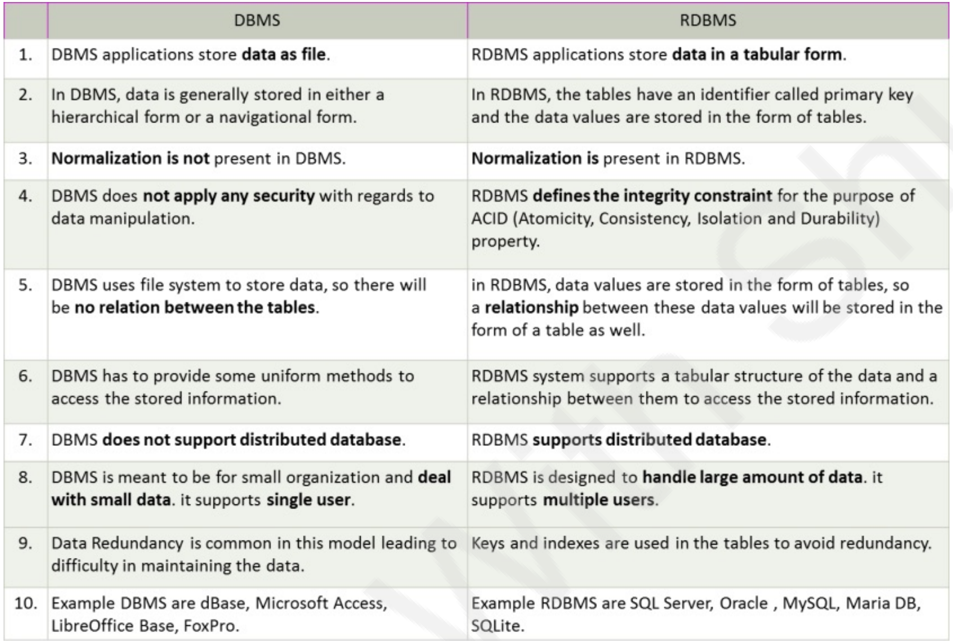
A = Atomicity → The entire transaction takes place at once or doesn't happen at all.

C = Consistency → The database must be consistent before and after the transaction.

I = Isolation → Multiple Transactions occur independently without interference.

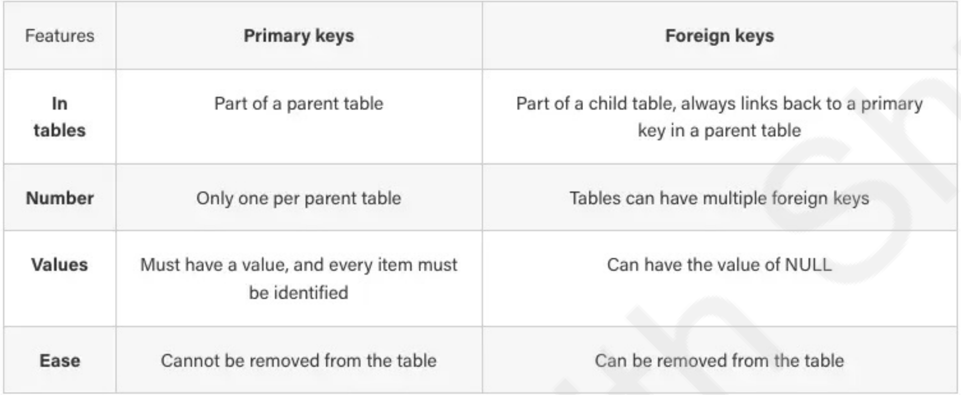
D = Durability → The changes of a successful transaction occurs even if the system failure occurs.

Difference Between DBMS and RDBMS

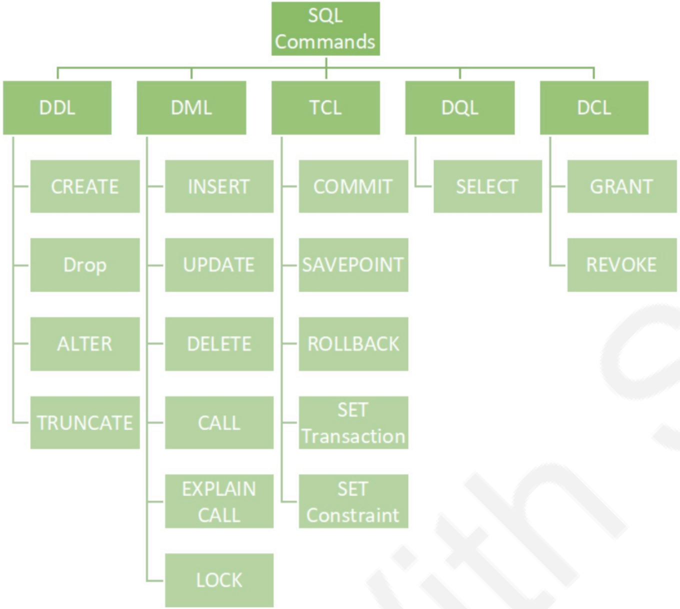


	DBMS	RDBMS
1.	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2.	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3.	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.
4.	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomicity, Consistency, Isolation and Durability) property.
5.	DBMS uses file system to store data, so there will be **no relation between the tables**.	in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
6.	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7.	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8.	DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
9.	Data Redundancy is common in this model leading to difficulty in maintaining the data.	Keys and indexes are used in the tables to avoid redundancy.
10.	Example DBMS are dBase, Microsoft Access, LibreOffice Base, FoxPro.	Example RDBMS are SQL Server, Oracle , MySQL, Maria DB, SQLite.

Difference Primary Key and Foreign Key



Features	Primary keys	Foreign keys
In tables	Part of a parent table	Part of a child table, always links back to a primary key in a parent table
Number	Only one per parent table	Tables can have multiple foreign keys
Values	Must have a value, and every item must be identified	Can have the value of NULL
Ease	Cannot be removed from the table	Can be removed from the table



SQL Commands

- DDL
  - CREATE
  - Drop
  - ALTER
  - TRUNCATE
- DML
  - INSERT
  - UPDATE
  - DELETE
  - CALL
  - EXPLAIN CALL
  - LOCK
- TCL
  - COMMIT
  - SAVEPOINT
  - ROLLBACK
  - SET Transaction
  - SET Constraint
- DQL
  - SELECT
- DCL
  - GRANT
  - REVOKE