

Database Management System Course Syllabus

Introduction

👉 Lesson 1	What is DBMS? — Application, Types & Example
👉 Lesson 2	Database Architecture in DBMS — Types of DBMS Architecture
👉 Lesson 3	DBMS Schemas — Internal, Conceptual & External
👉 Lesson 4	Relational Data Model in DBMS — Concepts, Constraints & Example

Advanced Stuff

👉 Lesson 1	ER Diagram — Learn with Example
👉 Lesson 2	Relational Algebra in DBMS — Operations with Examples
👉 Lesson 3	DBMS Transaction Management — What are ACID Properties?
👉 Lesson 4	DBMS Concurrency Control — Timestamp & Lock-Based Protocols
👉 Lesson 5	DBMS Keys — Learn with Example
👉 Lesson 6	Functional Dependency in DBMS — What is, Types & Examples
👉 Lesson 7	Data Independence in DBMS — Physical & Logical with Examples
👉 Lesson 8	Hashing in DBMS — Static & Dynamic with Examples
👉 Lesson 9	SQL Commands — DML, DDL, DCL, TCL, DQL with Query Example
👉 Lesson 10	DBMS Joins — Types of Join Operations
👉 Lesson 11	Indexing in DBMS — What is, Types of Indexes with EXAMPLES

Must Know!

👉 Lesson 1	DBMS vs RDBMS — What’s the Difference?
👉 Lesson 2	File System vs DBMS — Key Differences
👉 Lesson 3	SQL vs NoSQL — What’s the Difference?
👉 Lesson 4	Clustered vs Non-clustered Index — Key Differences with Example
👉 Lesson 5	Primary Key vs Foreign Key — What’s the Difference?
👉 Lesson 6	Primary Key vs Unique Key — What’s the Difference?
👉 Lesson 7	Row vs Column — Key Differences
👉 Lesson 8	DDL vs DML — What’s the Difference?
👉 Lesson 9	BEST Database Software — 13 BEST Free Database Software
👉 Lesson 10	Best Database Design Tools — 15 Best Database Design Tools
👉 Lesson 11	Top Database Interview Questions — Top 50 DBMS Interview Q & A

1.Introduction

What is DBMS (Database Management System)? Application, Types & Example

Before the introduction to Database Management System (DBMS), let's understand-

What is a Database?

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

What is DBMS?

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the [database](#) and other application programs. It provides an interface between the data and the software application.

In this Database Management System tutorial tutorial, you will learn DBMS concepts like-

- [What is a Database?](#)
- [What is DBMS \(Database Management System\)?](#)
- [Example of a DBMS](#)
- [History of DBMS](#)
- [Characteristics of DBMS](#)
- [DBMS vs. Flat File](#)
- [Users of DBMS](#)
- [Popular DBMS Software](#)
- [Application of DBMS](#)
- [Types of DBMS](#)
- [Advantages of DBMS](#)
- [Disadvantage of DBMS](#)
- [When not to use a DBMS system?](#)

Example of a DBMS

Let us see a simple example of a university database. This database is maintaining information concerning students, courses, and grades in a university environment. The database is organized as five files:

- The STUDENT file stores data of each student
- The COURSE file stores contain data on each course.
- The SECTION stores the information about sections in a particular course.
- The GRADE file stores the grades which students receive in the various sections
- The TUTOR file contains information about each professor.

To define DBMS:

- We need to specify the structure of the records of each file by defining the different types of data elements to be stored in each record.
- We can also use a coding scheme to represent the values of a data item.
- Basically, your Database will have 5 tables with a foreign key defined amongst the various tables.

History of DBMS

Here, are the important landmarks from the history:

- 1960 – Charles Bachman designed first DBMS system
- 1970 – Codd introduced IBM'S Information Management System (IMS)

- 1976- Peter Chen coined and defined the Entity-relationship model also know as the ER model
- 1980 – Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

Characteristics of DBMS

Here are the characteristics and properties of Database Management System:

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Database Management Software allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports multi-user environment that allows users to access and manipulate data in parallel.

DBMS vs. Flat File

DBMS	Flat File Management System
Multi-user access	It does not support multi-user access
Design to fulfill the need for small and large businesses	It is only limited to smaller DBMS system.
Remove redundancy and Integrity	Redundancy and Integrity issues
Expensive. But in the long term Total Cost of Ownership is cheap	It’s cheaper
Easy to implement complicated transactions	No support for complicated transactions

Users of DBMS

Following are the various category of users of DBMS

Component Name	Task
Application Programmers	The Application programmers write programs in various programming languages to interact with databases.
Database Administrators	Database Admin is responsible for managing the entire DBMS system. He/She is called Database admin or DBA.
End-Users	The end users are the people who interact with the database management system. They conduct various operations on database like retrieving, updating, deleting, etc.

Popular DBMS Software

Here, is the list of some popular DBMS system:

- [MySQL](#)
- Microsoft Access
- Oracle
- [PostgreSQL](#)
- dBASE
- FoxPro
- SQLite
- IBM DB2
- LibreOffice Base
- [MariaDB](#)
- Microsoft SQL Server etc.

Application of DBMS

Below are the popular database system applications:

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

Types of DBMS



Types of DBMS

Tha main Four Types of Database Management System are:

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database

Hierarchical DBMS

In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent.

Network Model

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.

Relational Model

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.

Object-Oriented Model

In Object-oriented Model data stored in the form of objects. The structure which is called classes which display data within it. It is one of the components of DBMS that defines a database as a collection of objects which stores both data members values and operations.

Advantages of DBMS

- DBMS offers a variety of techniques to store & retrieve data
- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Uniform administration procedures for data
- Application programmers never exposed to details of data representation and storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
- Reduced Application Development Time

Disadvantage of DBMS

DBMS may offer plenty of advantages but, it has certain flaws-

- Cost of Hardware and Software of a DBMS is quite high which increases the budget of your organization.
- Most database management systems are often complex systems, so the training for users to use the DBMS is required.
- In some organizations, all data is integrated into a single database which can be damaged because of electric failure or database is corrupted on the storage media
- Use of the same program at a time by many users sometimes lead to the loss of some data.
- DBMS can't perform sophisticated calculations

When not to use a DBMS system?

Although, DBMS system is useful. It is still not suited for specific task mentioned below:

Not recommended when you do not have the budget or the expertise to operate a DBMS. In such cases, Excel/CSV/Flat Files could do just fine.

Summary

- DBMS definition: A database is a collection of related data which represents some aspect of the real world
- The full form of DBMS is [Database Management System](#). DBMS stands for Database Management System is a software for storing and retrieving users' data by considering appropriate security measures.
- DBMS Provides security and removes redundancy
- DBMS has many advantages over tradition Flat File management system
- Some Characteristics of DBMS are Security, Self-describing nature, Insulation between programs and data abstraction, Support of multiple views of the data, etc.
- End-Users, Application Programmers, and Database Administrators are they type of users who access a DBMS
- DBMS is widely used in Banking, Airlines, Telecommunication, Finance and other industries
- The main Four DBMS types are 1) Hierarchical 2) Network 3) Relational 4) Object-Oriented DBMS
- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Cost of Hardware and Software of a DBMS is quite high which increases the budget of your organization

Database Architecture in DBMS: 1-Tier, 2-Tier and 3-Tier

What is Database Architecture?

A **Database Architecture** is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

A [Database](#) stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

- [Types of DBMS Architecture](#)
- [1-Tier Architecture](#)
- [2-Tier Architecture](#)
- [3-Tier Architecture](#)

Types of DBMS Architecture

There are mainly three types of DBMS architecture:

- One Tier Architecture (Single Tier Architecture)
- Two Tier Architecture
- Three Tier Architecture

Now, we will learn about different architecture of DBMS with diagram.

1-Tier Architecture

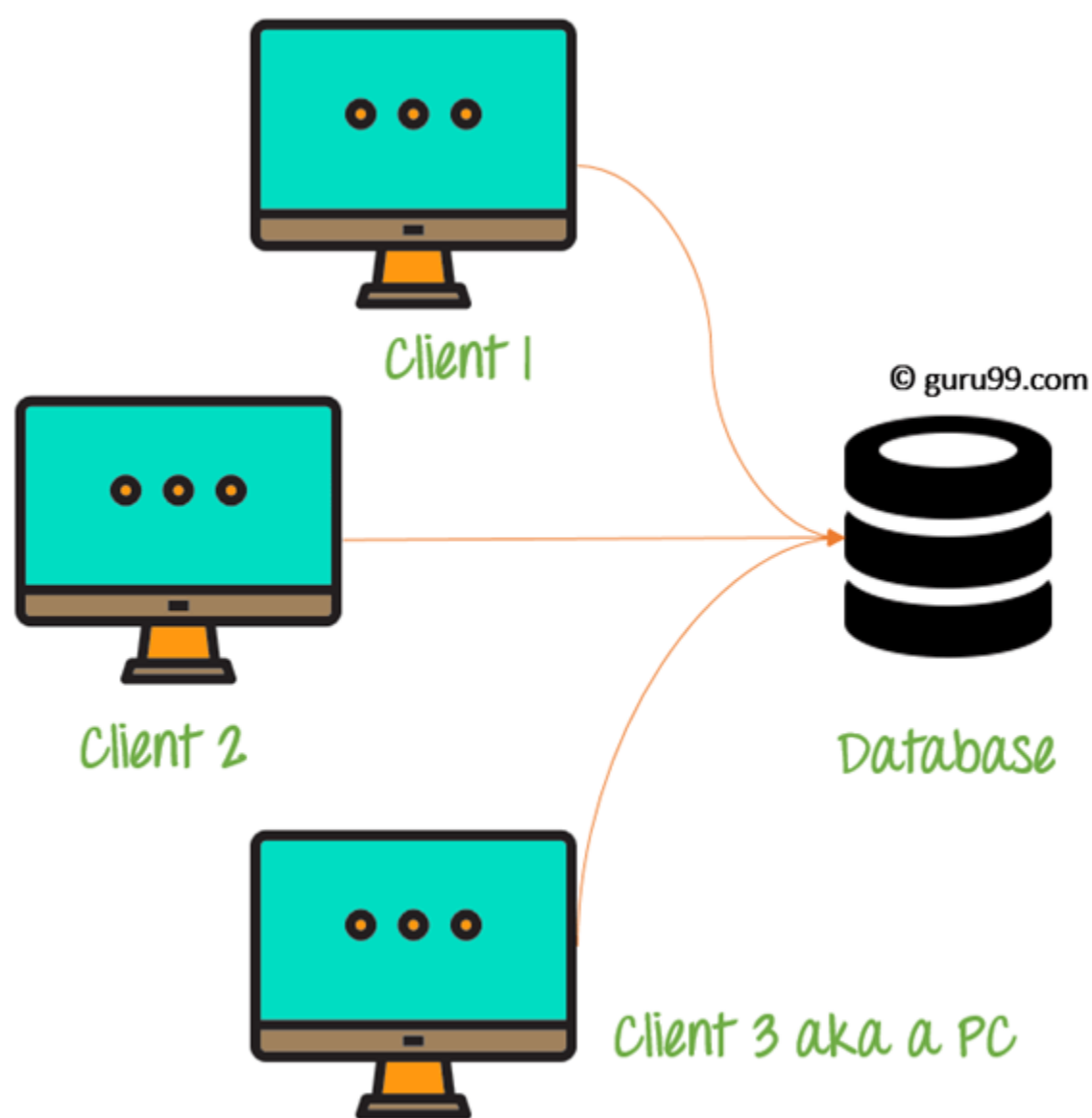
1 Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.



1 Tier Architecture Diagram

2-Tier Architecture

A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



2 Tier Architecture Diagram

In the above 2 Tier client-server architecture of database management system, we can see that one server is connected with clients 1, 2, and 3.

Two Tier Architecture Example:

A Contact Management System created using MS- Access.

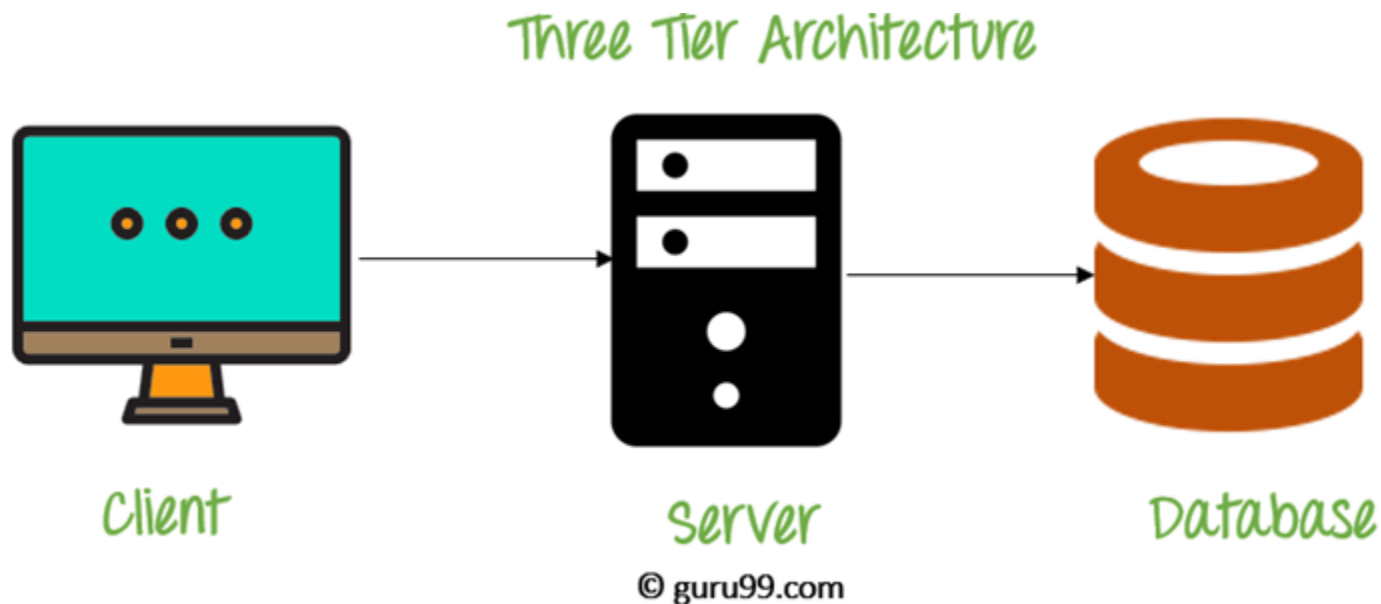
3-Tier Architecture

A **3 Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done

independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

3-Tier database Architecture design is an extension of the 2-tier client-server architecture. A 3-tier architecture has the following layers:

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server



3 Tier Architecture Diagram

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.

The goal of Three Tier client-server architecture is:

- To separate the user applications and physical database
- To support DBMS characteristics
- Program-data independence
- Supporting multiple views of the data

Three Tier Architecture Example:

Any large website on the internet, including **guru99.com**.

Summary

- An Architecture of DBMS helps in design, development, implementation, and maintenance of a database
- The simplest database system architecture is 1 tier where the Client, Server, and Database all reside on the same machine
- A two-tier architecture is a database architecture in DBMS where presentation layer runs on a client and data is stored on a server
- Three-tier client-server architecture consists of the Presentation layer (PC, Tablet, Mobile, etc.), Application layer (server) and Database Server

DBMS Schemas: Internal, Conceptual, External

Database systems comprise of complex data structures. Thus, to make the system efficient for retrieval of data and reduce the complexity of the users, developers use the method of Data Abstraction.

There are mainly three levels of data abstraction:

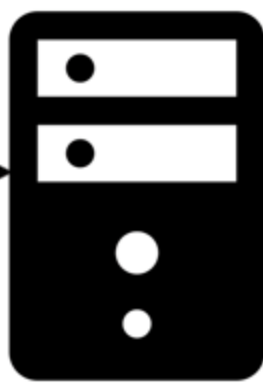
1. Internal Level: Actual PHYSICAL storage structure and access paths.
2. Conceptual or Logical Level: Structure and constraints for the entire database
3. External or View level: Describes various user views

External Level/ View Level



Client

Conceptual Level/ Logical Level



Server

© guru99.com

Internal Level



Database

Three Tier Architecture

Let's study them in detail

Internal Level/Schema

The internal schema defines the physical storage structure of the database. The internal schema is a very low-level representation of the entire database. It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called “stored record”.

Facts about Internal schema:

- The internal schema is the lowest level of data abstraction
- It helps you to keep information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records
- The internal view tells us what data is stored in the database and how
- It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages

Conceptual Schema/Level

The conceptual schema describes the Database structure of the whole database for the community of users. This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.

This logical level comes between the user level and physical storage view. However, there is only single conceptual view of a single database.

Facts about Conceptual schema:

- Defines all database entities, their attributes, and their relationships
- Security and integrity information
- In the conceptual level, the data available to a user must be contained in or derivable from the physical level

External Schema/Level

An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be “n” number of external views for each database.

Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.

An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

Facts about external schema:

- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user
- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group

Goal of 3 level/schema of Database

Here, are some Objectives of using Three schema Architecture:

- Every user should be able to access the same data but able to see a customized view of the data.
- The user need not to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without disturbing the user's views
- The internal structure of the database should remain unaffected when changes made to the physical aspects of storage.

Advantages Database Schema

- You can manage data independent of the physical storage
- Faster Migration to new graphical environments
- DBMS Architecture allows you to make changes on the presentation level without affecting the other two layers
- As each tier is separate, it is possible to use different sets of developers
- It is more secure as the client doesn't have direct access to the database business logic
- In case of the failure of the one-tier no data loss as you are always secure by accessing the other tier

Disadvantages Database Schema

- Complete DB Schema is a complex structure which is difficult to understand for every one
- Difficult to set up and maintain
- The physical separation of the tiers can affect the performance of the Database

Summary

- There are mainly three levels of data abstraction: Internal Level, Conceptual or Logical Level or External or View level
- The internal schema defines the physical storage structure of the database
- The conceptual schema describes the Database structure of the whole database for the community of users
- An external schema describe the part of the database which specific user is interested in
- DBMS Architecture allows you to make changes on the presentation level without affecting the other two layers

Relational Data Model in DBMS | Database Concepts & Example

What is Relational Model?

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

Some popular Relational Database management systems are:

- DB2 and Informix Dynamic Server – IBM
- Oracle and RDB – Oracle
- SQL Server and Access – Microsoft

In this tutorial, you will learn

- [Relational Model Concepts in DBMS](#)
- [Relational Integrity Constraints](#)
- [Operations in Relational Model](#)

- [Best Practices for creating a Relational Model](#)
- [Advantages of Relational Database Model](#)
- [Disadvantages of Relational Model](#)

Relational Model Concepts in DBMS

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** – Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

Table also called Relation

Primary Key

Domain
Ex: NOT NULL

© guru99.com

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

Tuple OR Row
Total # of rows is Cardinality

Column OR Attributes
Total # of column is Degree

Relational Integrity Constraints

Relational Integrity constraints in DBMS are referred to conditions which must be present for a valid relation. These Relational constraints in DBMS are derived from the rules in the mini-world that the database represents.

There are many types of Integrity Constraints in DBMS. Constraints on the Relational database management system is mostly divided into three main categories are:

1. Domain Constraints
2. Key Constraints
3. Referential Integrity Constraints

Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example:

```
Create DOMAIN CustomerName
```

CHECK (value not NULL)

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

Key Constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

Example:

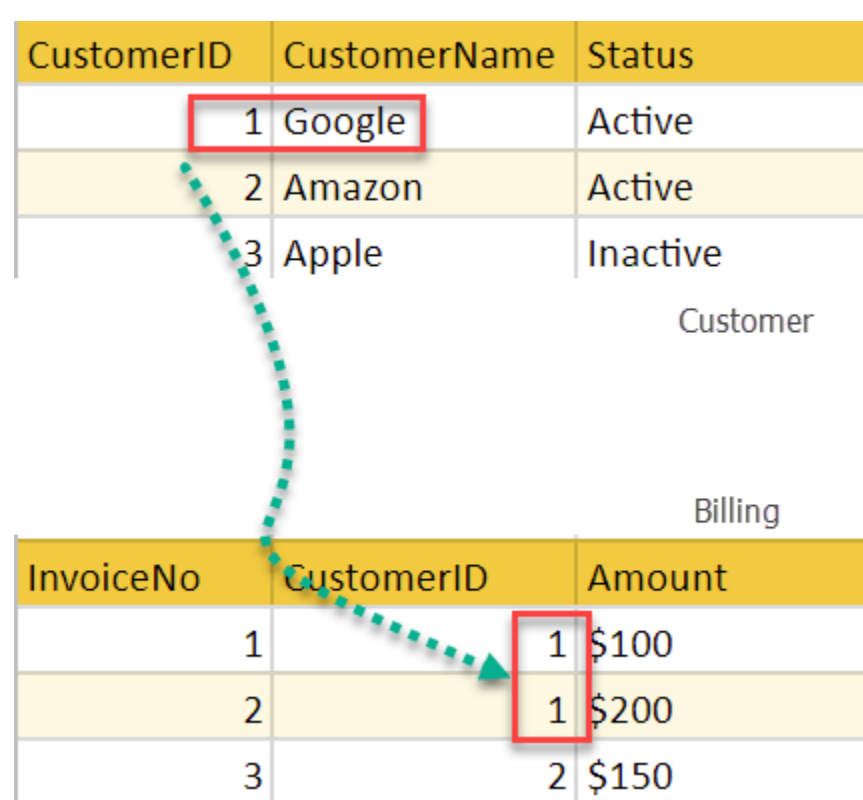
In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =” Google”.

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

Referential Integrity Constraints

Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Example:



In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount \$300

Operations in Relational Model

Four basic update operations performed on relational database model are

Insert, update, delete and select.

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

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

Insert Operation

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

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

INSERT

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

Update Operation

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

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

UPDATE

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

Delete Operation

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

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

DELETE

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

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

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

Select Operation

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

SELECT

CustomerID	CustomerName	Status
2	Amazon	Active

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

Best Practices for creating a Relational Model

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name
- No two rows can be identical
- The values of an attribute should be from the same domain

Advantages of Relational Database Model

- **Simplicity:** A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand

- **Query capability:** It makes possible for a high-level query language like [SQL](#) to avoid complex database navigation.
- **Data independence:** The Structure of Relational database can be changed without having to change any application.
- **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of Relational Model

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

Summary

- The Relational database modelling represents the database as a collection of relations (tables)
- Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation instance, are some important components of Relational Model
- Relational Integrity constraints are referred to conditions which must be present for a valid Relation approach in DBMS
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type
- Insert, Select, Modify and Delete are the operations performed in Relational Model constraints
- The relational database is only concerned with data and not with a structure which can improve the performance of the model
- Advantages of Relational model in DBMS are simplicity, structural independence, ease of use, query capability, data independence, scalability, etc.
- Few relational databases have limits on field lengths which can't be exceeded.

2.Advanced Stuff

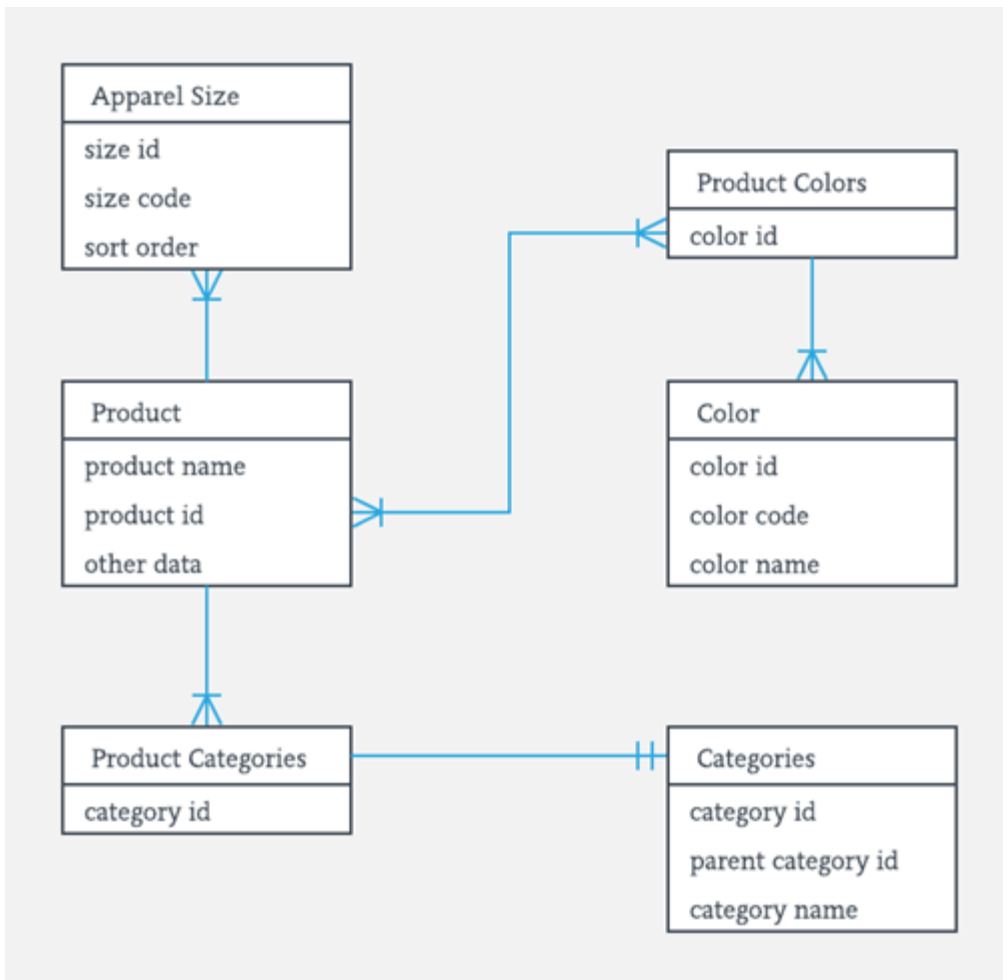
ER Diagram: Entity Relationship Diagram Model | DBMS Example

What is ER Diagram?

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



Entity Relationship Diagram Example

What is ER Model?

ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.

[ER Modeling](#) helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

In this Entity Relationship Diagram tutorial, you will learn-

- [What is ER Diagram?](#)
- [What is the ER Model?](#)
- [History of ER models](#)
- [Why use ER Diagrams?](#)
- [Facts about ER Diagram Model](#)
- [ER Diagrams Symbols & Notations](#)
- [Components of ER Diagram](#)
- [Relationship](#)
- [Weak Entities](#)
- [Attributes](#)
- [Cardinality](#)
- [How to Create an ER Diagram \(ERD\)](#)
- [Best Practices for Developing Effective ER Diagrams](#)

History of ER models

ER diagrams are visual tools that are helpful to represent the ER model. Peter Chen proposed ER Diagram in 1971 to create a uniform convention that can be used for relational databases and networks. He aimed to use an ER model as a conceptual modeling approach.

Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications

- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users

Facts about ER Diagram Model

Now in this ERD Diagram Tutorial, let’s check out some interesting facts about ER Diagram Model:

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

ER Diagrams Symbols & Notations

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



ER Diagram Symbols

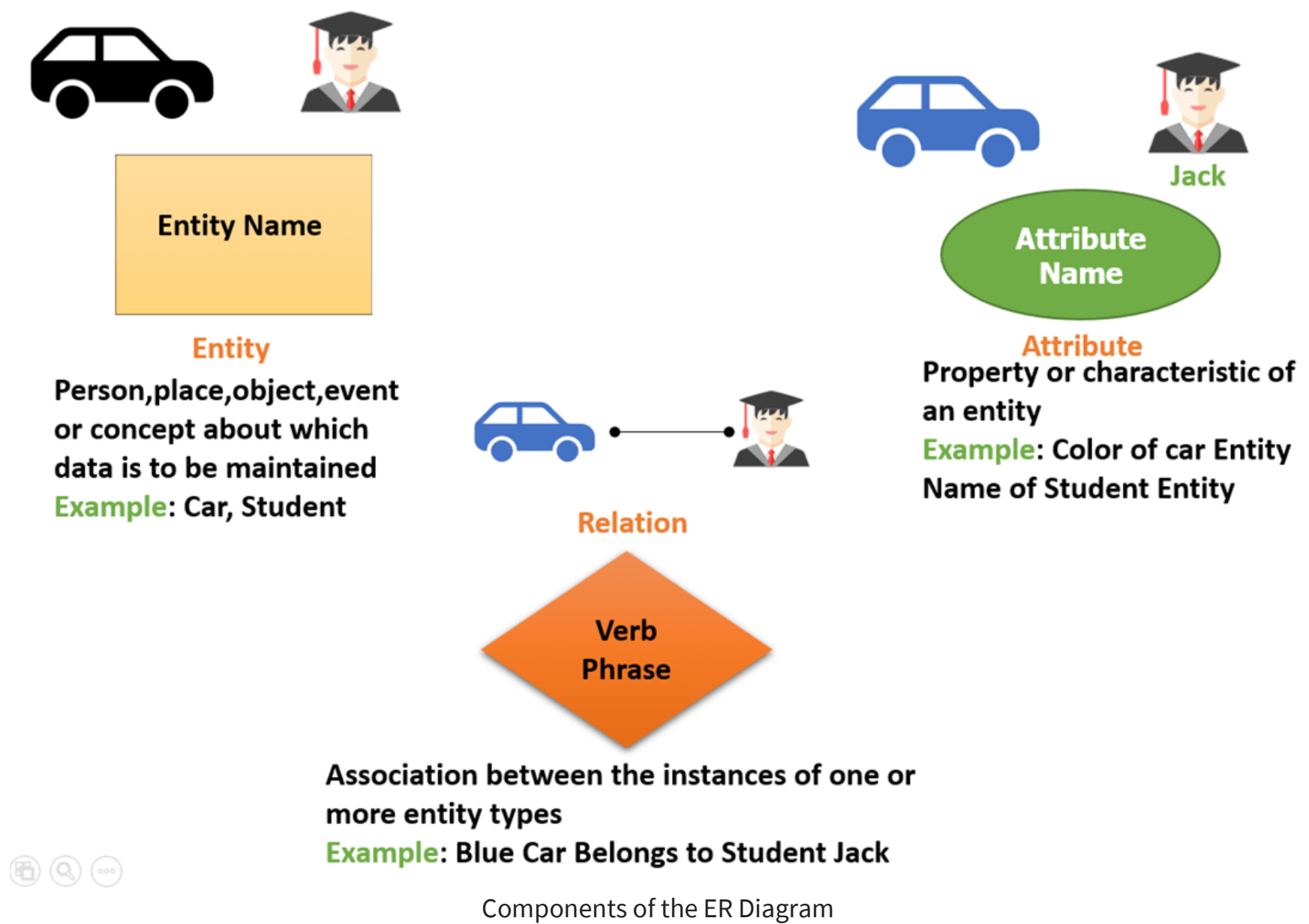
Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

ER Diagram Examples

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.



WHAT IS ENTITY?

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Examples of entities:

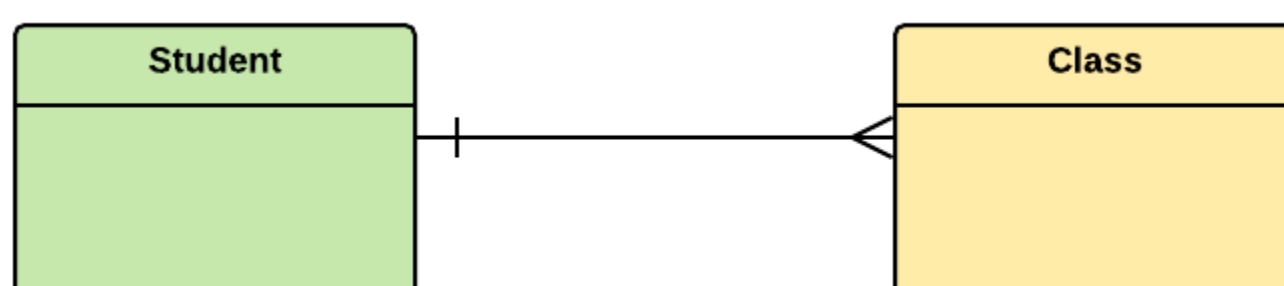
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Notation of an Entity

Entity set:

Student

An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.



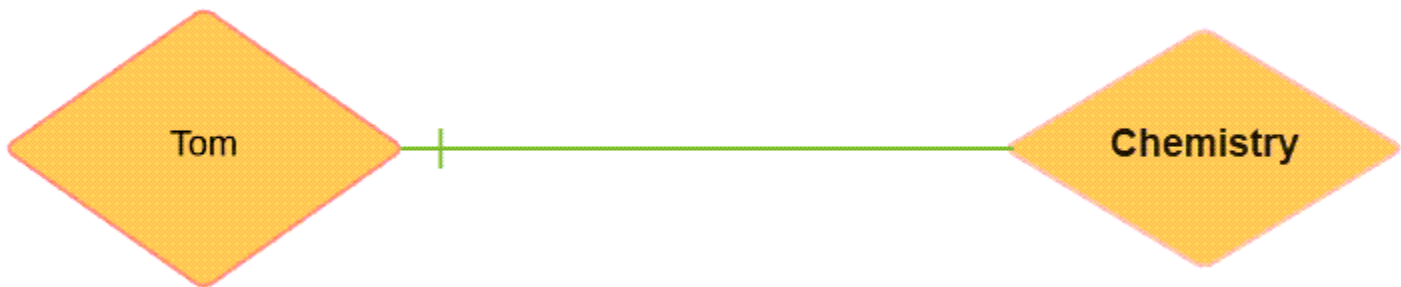
Example of Entities:

A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Relationship

Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.



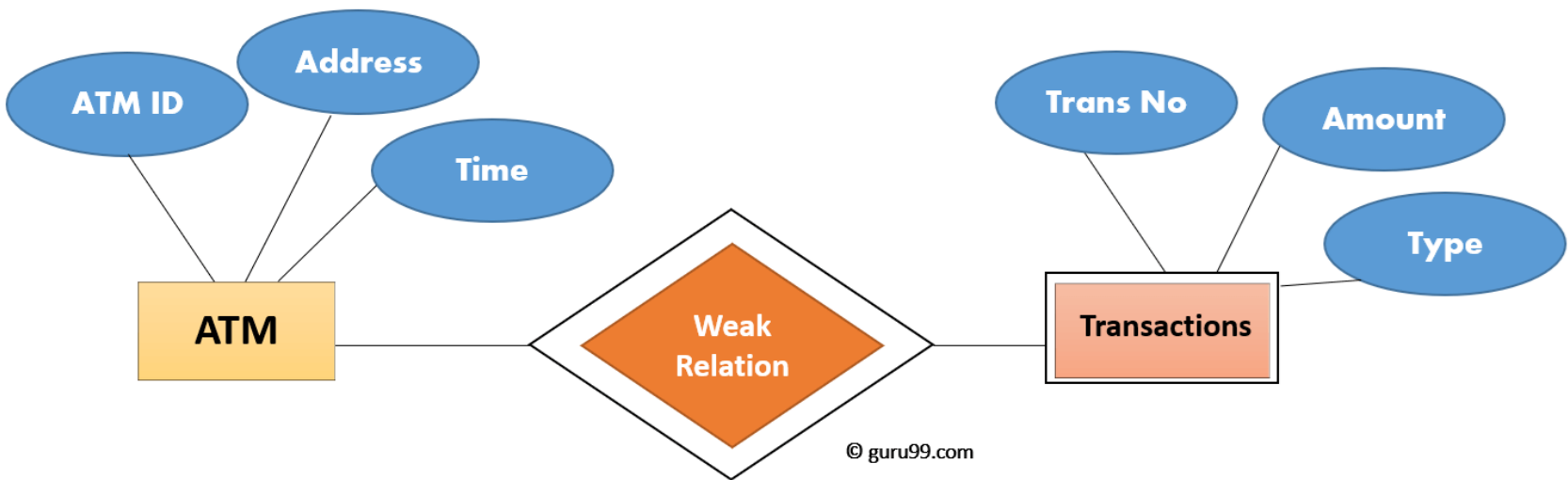
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

For example:

- You are attending this lecture
- I am giving the lecture
- Just like entities, we can classify relationships according to relationship-types:
- A student attends a lecture
- A lecturer is giving a lecture.

Weak Entities

A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.



In above ER Diagram examples, “Trans No” is a discriminator within a group of transactions in an ATM.

Let’s learn more about a weak entity by comparing it with a Strong Entity

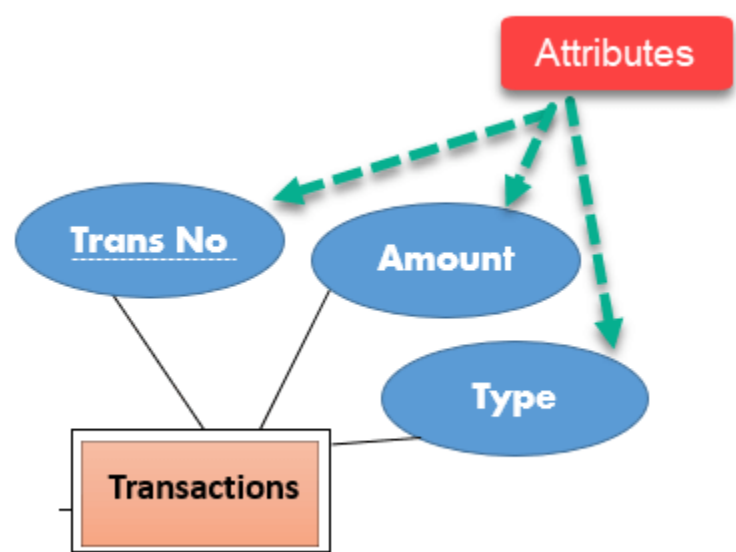
Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

Attributes

It is a single-valued property of either an entity-type or a relationship-type.

For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute in ER Diagram examples, is represented by an Ellipse



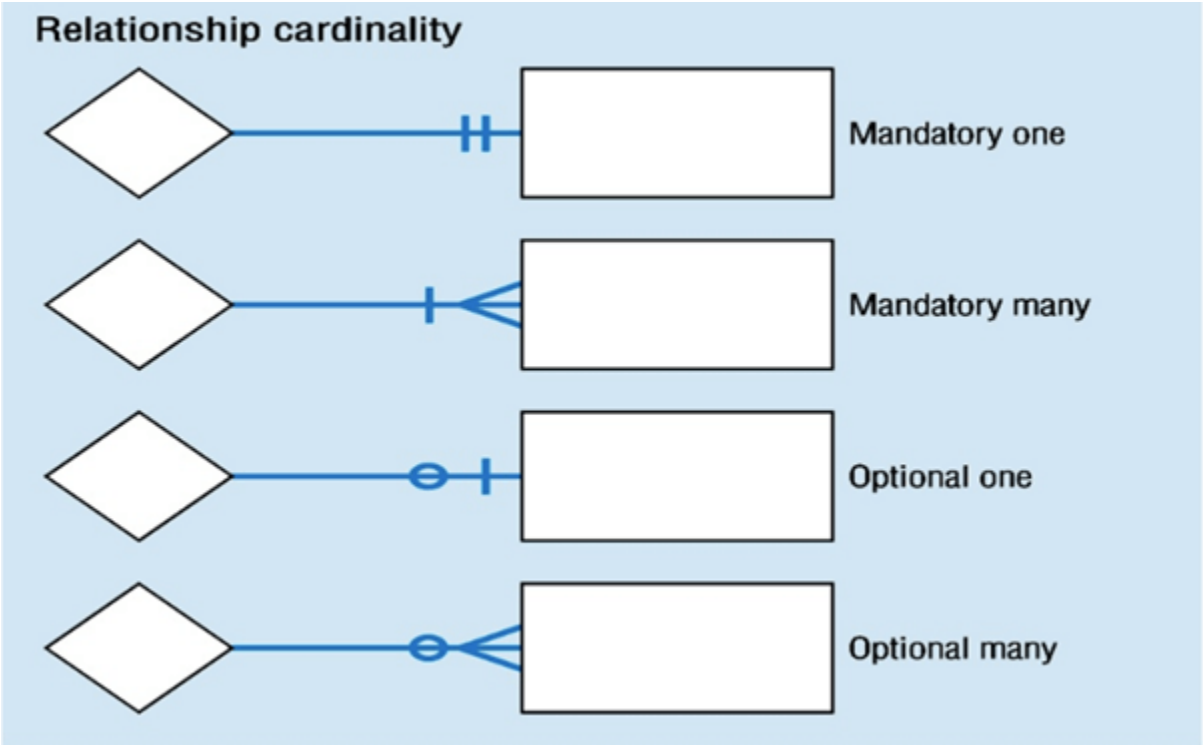
Types of Attributes	Description
Simple attribute	Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

Cardinality

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

Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships



1.One-to-one:

One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.

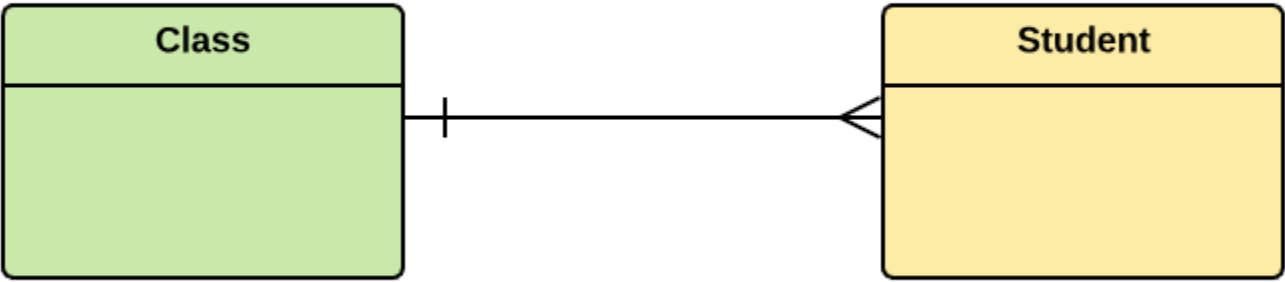
Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.



2. One-to-many:

One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

For example, one class is consisting of multiple students.



3. Many to One

More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

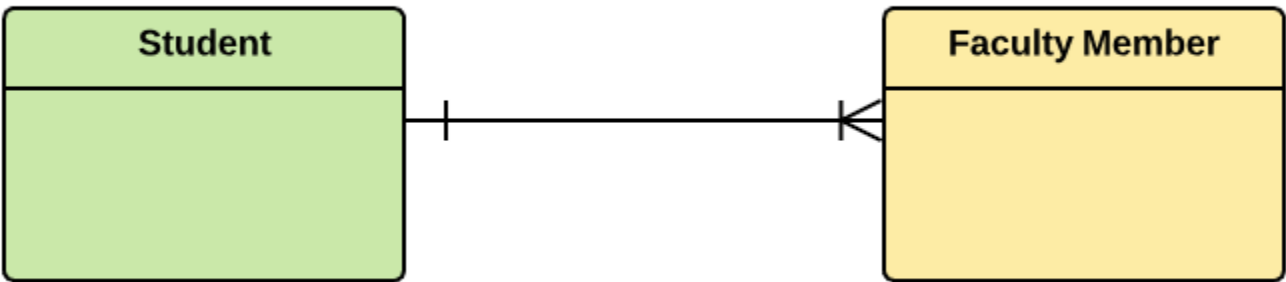
For example, many students belong to the same class.



4. Many to Many:

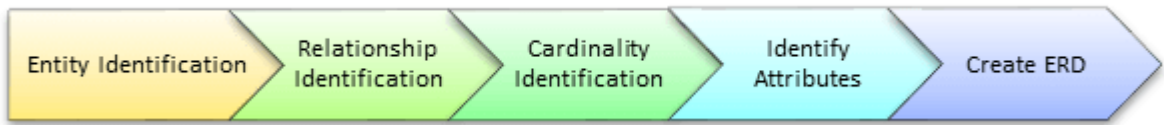
One entity from X can be associated with more than one entity from Y and vice versa.

For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.



How to Create an Entity Relationship Diagram (ERD)

Now in this ERD Diagram Tutorial, we will learn how to create an ER Diagram. Following are the steps to create an ER Diagram:



Steps to Create an ER Diagram

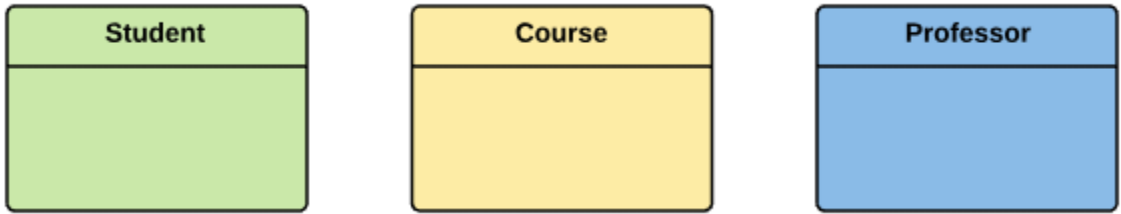
Let’s study them with an Entity Relationship Diagram Example:

In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

Step 1) Entity Identification

We have three entities

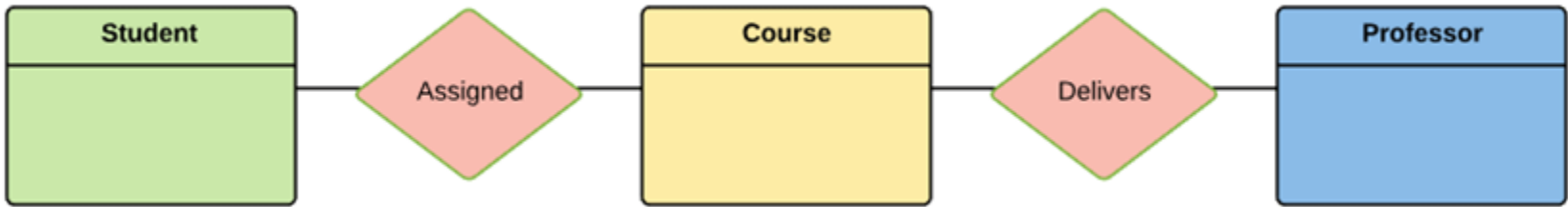
- Student
- Course
- Professor



Step 2) Relationship Identification

We have the following two relationships

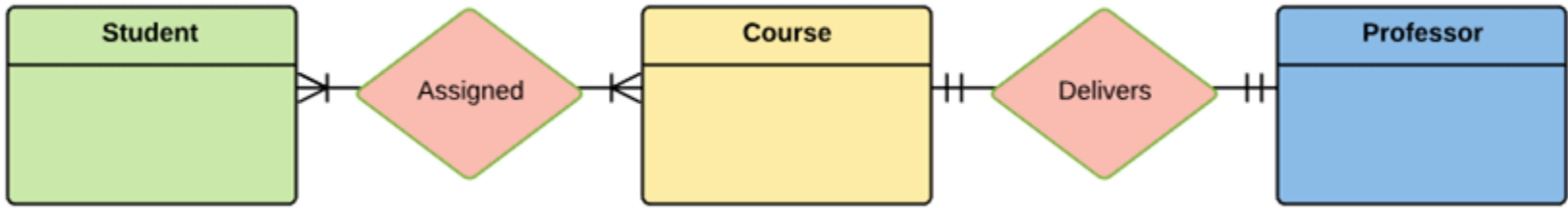
- The student is **assigned** a course
- Professor **delivers** a course



Step 3) Cardinality Identification

For them problem statement we know that,

- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



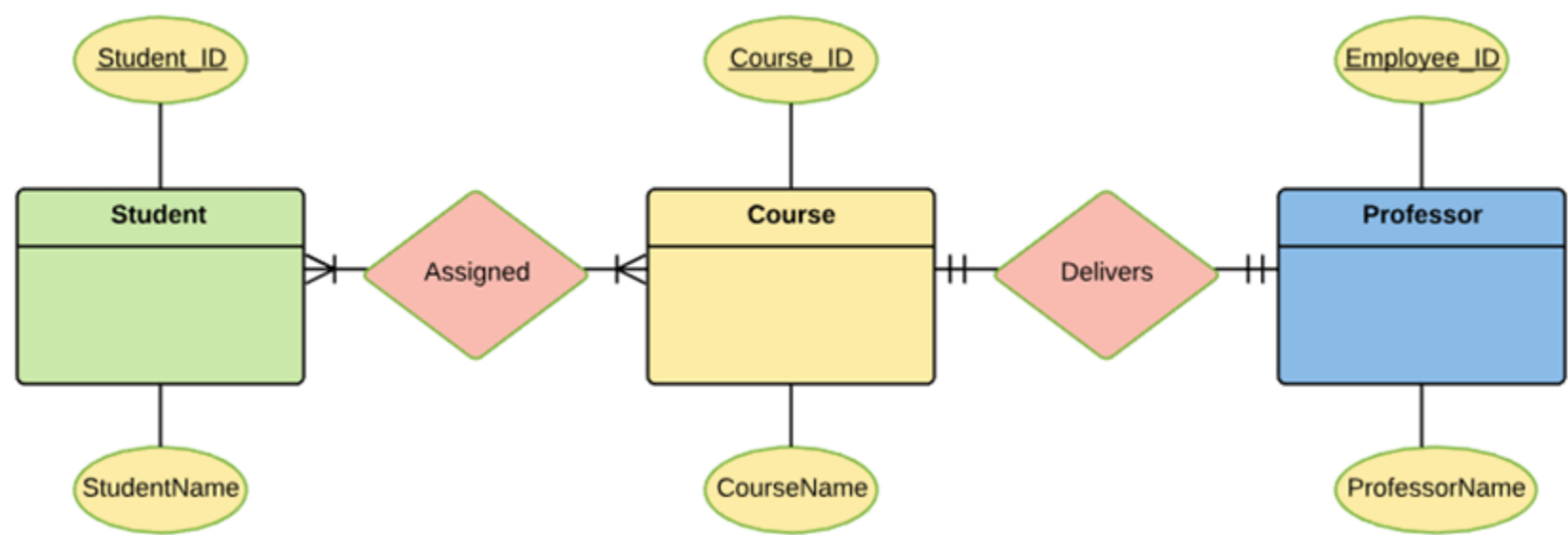
Step 4) Identify Attributes

You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it’s important to identify the attributes without mapping them to a particular entity.

Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

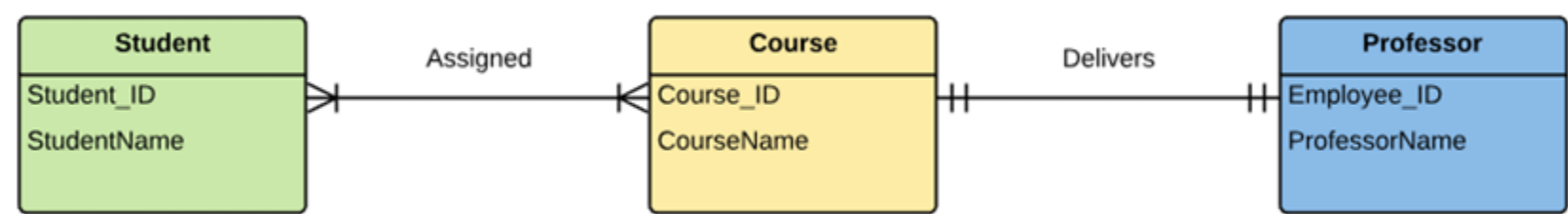
Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.

Step 5) Create the ERD Diagram

A more modern representation of Entity Relationship Diagram Example



Best Practices for Developing Effective ER Diagrams

Here are some best practice or example for Developing Effective ER Diagrams.

- Eliminate any redundant entities or relationships
- You need to make sure that all your entities and relationships are properly labeled
- There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
- You should assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on your diagram
- Never connect relationships to each other
- You should use colors to highlight important portions of the ER diagram

Summary

- ER Model in DBMS stands for an Entity-Relationship model
- The ER model is a high-level data model diagram
- ER diagrams are a visual tool which is helpful to represent the ER model
- ER diagrams in DBMS are blueprint of a database
- Entity relationship diagram DBMS displays the relationships of entity set stored in a database
- ER diagrams help you to define terms related to entity relationship modeling
- ER Model in DBMS is based on three basic concepts: Entities, Attributes & Relationships
- An entity can be place, person, object, event or a concept, which stores data in the database (DBMS)
- Relationship is nothing but an association among two or more entities
- A weak entity is a type of entity which doesn't have its key attribute
- It is a single-valued property of either an entity-type or a relationship-type
- It helps you to defines the numerical attributes of the relationship between two entities or entity sets
- ER- Diagram DBMS is a visual representation of data that describe how data is related to each other

- While Drawing ER diagrams in DBMS, you need to make sure all your entities and relationships are properly labeled.

Relational Algebra in DBMS: Operations with Examples

Relational Algebra

RELATIONAL ALGEBRA is a widely used procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operations to perform this action. SQL Relational algebra query operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

In this tutorial, you will learn:

- [Relational Algebra](#)
- [SELECT\(\$\sigma\$ \)](#)
- [Projection\(\$\pi\$ \)](#)
- [Rename \(\$\rho\$ \)](#)
- [Union operation \(\$\cup\$ \)](#)
- [Set Difference \(\$-\$ \)](#)
- [Intersection](#)
- [Cartesian product\(\$\times\$ \)](#)
- [Join Operations](#)
- [Inner Join:](#)
- [Theta Join:](#)
- [EQUI join:](#)
- [NATURAL JOIN \(\$\bowtie\$ \)](#)
- [OUTER JOIN](#)
- [Left Outer Join\(\$A \Joinr B\$ \)](#)
- [Right Outer Join: \(\$A \Joinl B\$ \)](#)
- [Full Outer Join: \(\$A \Join B\$ \)](#)

Basic SQL Relational Algebra Operations

Relational Algebra devided in various groups

Unary Relational Operations

- [SELECT \(symbol: \$\sigma\$ \)](#)
- [PROJECT \(symbol: \$\pi\$ \)](#)
- [RENAME \(symbol: \$\rho\$ \)](#)

Relational Algebra Operations From Set Theory

- [UNION \(\$\cup\$ \)](#)
- [INTERSECTION \(\$\cap\$ \),](#)
- [DIFFERENCE \(\$-\$ \)](#)
- [CARTESIAN PRODUCT \(\$\times\$ \)](#)

Binary Relational Operations

- [JOIN](#)
- [DIVISION](#)

Let’s study them in detail with solutions:

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operator selects tuples that satisfy a given predicate.

$\sigma_p(r)$

σ is the predicate

r stands for relation which is the name of the table

p is propositional logic

Example 1

$\sigma_{\text{topic} = \text{"Database"}}(\text{Tutorials})$

Output – Selects tuples from Tutorials where topic = ‘Database’.

Example 2

$\sigma_{\text{topic} = \text{"Database"} \text{ and } \text{author} = \text{"guru99"}}(\text{Tutorials})$

Output – Selects tuples from Tutorials where the topic is ‘Database’ and ‘author’ is guru99.

Example 3

$\sigma_{\text{sales} > 50000}(\text{Customers})$

Output – Selects tuples from Customers where sales is greater than 50000

Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminates duplicate values. (π) symbol is used to choose attributes from a relation. This operator helps you to keep specific columns from a relation and discards the other columns.

Example of Projection:

Consider the following table

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

Here, the projection of CustomerName and status will give

$\pi_{\text{CustomerName, Status}}(\text{Customers})$

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

Rename (ρ)

Rename is a unary operation used for renaming attributes of a relation.

$\rho_{(a/b)}R$ will rename the attribute ‘b’ of relation by ‘a’.

Union operation (\cup)

UNION is symbolized by \cup symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result $\leftarrow A \cup B$

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

- R and S must be the same number of attributes.
- Attribute domains need to be compatible.

- Duplicate tuples should be automatically removed.

Example

Consider the following tables.

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

A U B gives

Table A U B	
column 1	column 2
1	1
1	2
1	3

Set Difference (-)

- Symbol denotes it. The result of A – B, is a relation which includes all tuples that are in A but not in B.

- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be defined relation consisting of the tuples that are in relation A, but not in B.

Example

A – B

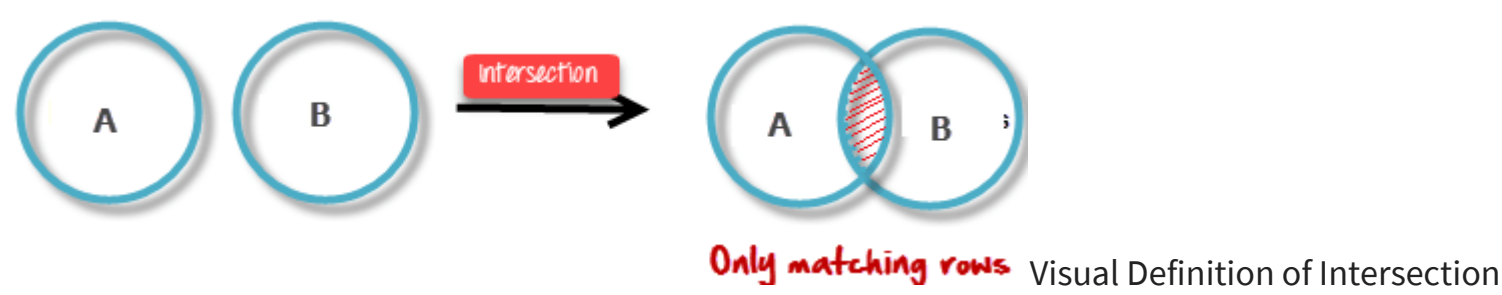
Table A – B	
column 1	column 2
1	2

Intersection

An intersection is defined by the symbol \cap

$A \cap B$

Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union-compatible.



Example:

$A \cap B$

Table $A \cap B$	
column 1	column 2
1	1

Cartesian Product(X) in DBMS

Cartesian Product in DBMS is an operation used to merge columns from two relations. Generally, a cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations. It is also called Cross Product or Cross Join.

Example – Cartesian product

$\sigma_{column\ 2 = '1'}(A \times B)$

Output – The above example shows all rows from relation A and B whose column 2 has value 1

$\sigma_{column\ 2 = '1'}(A \times B)$	
column 1	column 2
1	1
1	1

Join Operations

Join operation is essentially a cartesian product followed by a selection criterion.

Join operation denoted by \bowtie .

JOIN operation also allows joining variously related tuples from different relations.

Types of JOIN:

Various forms of join operation are:

Inner Joins:

- Theta join
- EQUI join
- Natural join

Outer join:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Inner Join:

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let’s study various types of Inner Joins:

Theta Join:

The general case of JOIN operation is called a Theta join. It is denoted by symbol θ

Example

$A \bowtie_{\theta} B$

Theta join can use any conditions in the selection criteria.

For example:

$A \bowtie_{A.column\ 2 > B.column\ 2} (B)$

$A \bowtie_{A.column\ 2 > B.column\ 2} (B)$	
column 1	column 2
1	2

EQUI join:

When a theta join uses only equivalence condition, it becomes a equi join.

For example:

$A \bowtie_{A.column\ 2 = B.column\ 2} (B)$

$A \bowtie_{A.column\ 2 = B.column\ 2} (B)$	
column 1	column 2
1	1

EQUI join is the most difficult operations to implement efficiently using SQL in an RDBMS and one reason why RDBMS have essential performance problems.

NATURAL JOIN (\bowtie)

Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

Example

Consider the following two tables

C	
Num	Square
2	4
3	9

D	
Num	Cube
2	8
3	27

C \bowtie D		
Num	Square	Cube
2	4	8
3	9	27

OUTER JOIN

In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

Left Outer Join($A \Joinr B$)

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.



Consider the following 2 Tables

A	
Num	Square
2	4
3	9
4	16

B	
Num	Cube
2	8
3	18
5	75

A \Joinr B

A ⋈ B		
Num	Square	Cube
2	4	8
3	9	18
4	16	–

Right Outer Join: (A ⋈⋈ B)

In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.



A ⋈⋈ B

A ⋈⋈ B		
Num	Cube	Square
2	8	4
3	18	9
5	75	–

Full Outer Join: (A ⋈⋈⋈ B)

In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

A ⋈⋈⋈ B

A ⋈⋈⋈ B		
Num	Cube	Square
2	4	8
3	9	18
4	16	–
5	–	75

Summary

Operation(Symbols)	Purpose
Select(σ)	The SELECT operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(\cup)	UNION is symbolized by symbol. It includes all tuples that are in tables A or in B.
Set Difference($-$)	$-$ Symbol denotes it. The result of $A - B$, is a relation which includes all tuples that are in A but not in B.
Intersection(\cap)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.
Cartesian Product(\times)	Cartesian operation is helpful to merge columns from two relations.
Inner Join	Inner join, includes only those tuples that satisfy the matching criteria.
Theta Join(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ .
EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.

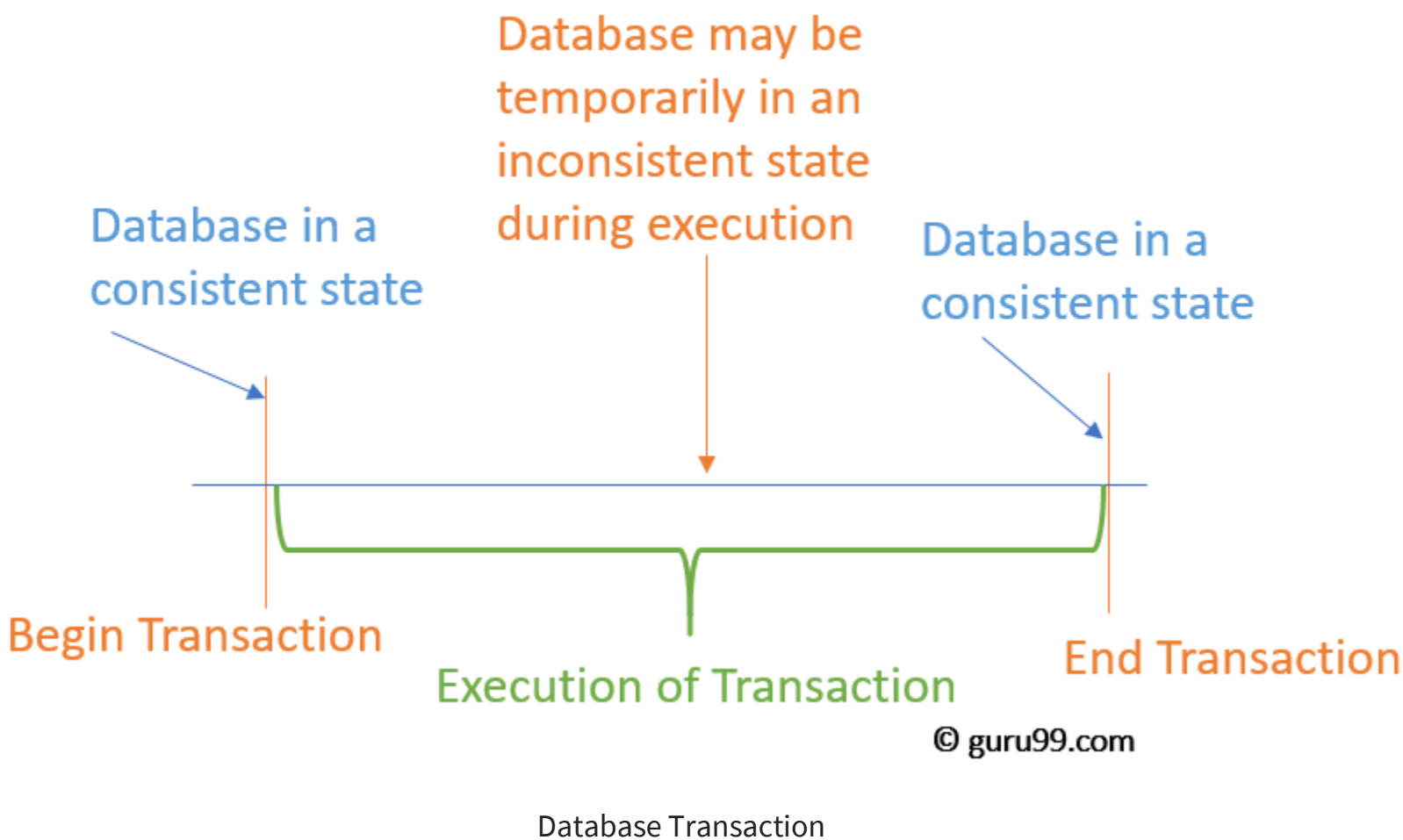
Natural Join(\bowtie)	Natural join can only be performed if there is a common attribute (column) between the relations.
Outer Join	In an outer join, along with tuples that satisfy the matching criteria.
Left Outer Join(\Join_{\leftarrow})	In the left outer join, operation allows keeping all tuple in the left relation.
Right Outer join(\Join_{\rightarrow})	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join(\Join_{\times})	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

Transaction Management in DBMS: What are ACID Properties?

 By [Richard Peterson](#) Updated February 12, 2022

What is a Database Transaction?

A **Database Transaction** is a logical unit of processing in a DBMS which entails one or more database access operation. In a nutshell, database transactions represent real-world events of any enterprise. All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction in DBMS. During the transaction the database is inconsistent. Only once the database is committed the state is changed from one consistent state to another.



In this tutorial, you will learn:

- [What is a Database Transaction?](#)
- [Facts about Database Transactions](#)
- [Why do you need concurrency in Transactions?](#)
- [States of Transactions](#)
- [What are ACID Properties?](#)
- [Types of Transactions](#)
- [What is a Schedule?](#)

Facts about Database Transactions

- A transaction is a program unit whose execution may or may not change the contents of a database.
- The transaction concept in DBMS is executed as a single unit.
- If the database operations do not update the database but only retrieve data, this type of transaction is called a read-only transaction.
- A successful transaction can change the database from one CONSISTENT STATE to another
- DBMS transactions must be atomic, consistent, isolated and durable

- If the database were in an inconsistent state before a transaction, it would remain in the inconsistent state after the transaction.

Why do you need concurrency in Transactions?

A database is a shared resource accessed. It is used by many users and processes concurrently. For example, the banking system, railway, and air reservations systems, stock market monitoring, supermarket inventory, and checkouts, etc.

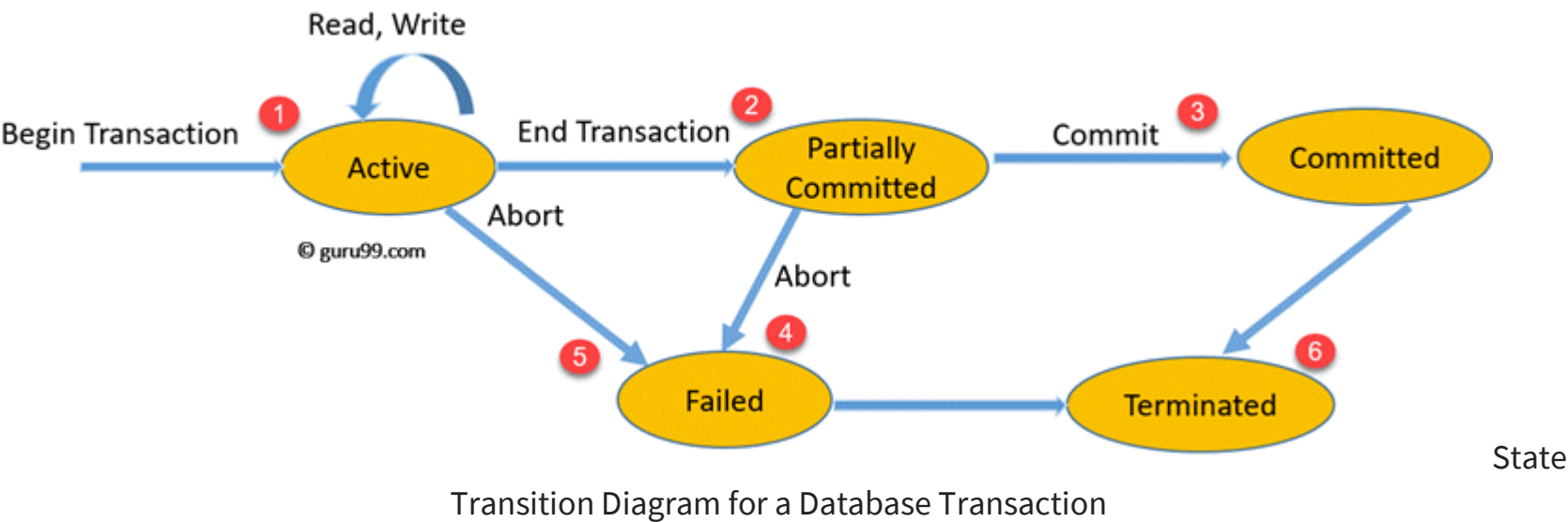
Not managing concurrent access may create issues like:

- Hardware failure and system crashes
- Concurrent execution of the same transaction, [deadlock](#), or slow performance

States of Transactions

The various states of a transaction concept in DBMS are listed below:

State	Transaction types
Active State	A transaction enters into an active state when the execution process begins. During this state read or write operations can be performed.
Partially Committed	A transaction goes into the partially committed state after the end of a transaction.
Committed State	When the transaction is committed to state, it has already completed its execution successfully. Moreover, all of its changes are recorded to the database permanently.
Failed State	A transaction considers failed when any one of the checks fails or if the transaction is aborted while it is in the active state.
Terminated State	State of transaction reaches terminated state when certain transactions which are leaving the system can't be restarted.



Let's study a [state transition diagram](#) that highlights how a transaction moves between these various states.

1. Once a transaction states execution, it becomes active. It can issue READ or WRITE operation.
2. Once the READ and WRITE operations complete, the transactions becomes partially committed state.
3. Next, some recovery protocols need to ensure that a system failure will not result in an inability to record changes in the transaction permanently. If this check is a success, the transaction commits and enters into the committed state.
4. If the check is a fail, the transaction goes to the Failed state.
5. If the transaction is aborted while it's in the active state, it goes to the failed state. The transaction should be rolled back to undo the effect of its write operations on the database.
6. The terminated state refers to the transaction leaving the system.

What are ACID Properties?

ACID Properties are used for maintaining the integrity of database during transaction processing. ACID in DBMS stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

- **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
- **Consistency:** Once the transaction is executed, it should move from one consistent state to another.
- **Isolation:** Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)

- **Durability:** • After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

ACID Property in DBMS with example:

Below is an example of ACID property in DBMS:

```
Transaction 1: Begin X=X+50, Y = Y-50 END
Transaction 2: Begin X=1.1*X, Y=1.1*Y END
```

Transaction 1 is transferring \$50 from account X to account Y.

Transaction 2 is crediting each account with a 10% interest payment.

If both transactions are submitted together, there is no guarantee that the Transaction 1 will execute before Transaction 2 or vice versa. Irrespective of the order, the result must be as if the transactions take place serially one after the other.

Types of Transactions

Based on Application areas

- Non-distributed vs. distributed
- Compensating transactions
- Transactions Timing
- On-line vs. batch

Based on Actions

- Two-step
- Restricted
- Action model

Based on Structure

- Flat or simple transactions: It consists of a sequence of primitive operations executed between a begin and end operations.
- Nested transactions: A transaction that contains other transactions.
- Workflow

What is a Schedule?

A Schedule is a process creating a single group of the multiple parallel transactions and executing them one by one. It should preserve the order in which the instructions appear in each transaction. If two transactions are executed at the same time, the result of one transaction may affect the output of other.

Example

```
Initial Product Quantity is 10
Transaction 1: Update Product Quantity to 50
Transaction 2: Read Product Quantity
```

If Transaction 2 is executed before Transaction 1, outdated information about the product quantity will be read. Hence, schedules are required.

Parallel execution in a database is inevitable. But, Parallel execution is permitted when there is an equivalence relation amongst the simultaneously executing transactions. This equivalence is of 3 Types.

RESULT EQUIVALENCE:

If two schedules display the same result after execution, it is called result equivalent schedule. They may offer the same result for some value and different results for another set of values. For example, one transaction updates the product quantity, while other updates customer details.

View Equivalence

View Equivalence occurs when the transaction in both the schedule performs a similar action. Example, one transaction inserts product details in the product table, while another transaction inserts product details in the archive table. The transaction is the same, but the tables are different.

CONFLICT Equivalence

In this case, two transactions update/view the same set of data. There is a conflict amongst transaction as the order of execution will affect the output.

What is Serializability?

Serializability is the process of search for a concurrent schedule whose output is equal to a serial schedule where transactions are executed one after the other. Depending on the type of schedules, there are two types of serializability:

- Conflict
- View

Summary:

- Transaction management is a logical unit of processing in a DBMS which entails one or more database access operations
- It is a transaction is a program unit whose execution may or may not change the contents of a database.
- Not managing concurrent access may create issues like hardware failure and system crashes.
- Active, Partially Committed, Committed, Failed & Terminate are important transaction states.
- The full form of ACID Properties in DBMS is Atomicity, Consistency, Isolation, and Durability
- Three DBMS transaction types are based on Application Areas, Action, & Structure.
- A Schedule is a process creating a single group of the multiple parallel transactions and executing them one by one.
- Serializability is the process of search for a concurrent schedule whose output is equal to a serial schedule where transactions are executed one after the other.

DBMS Concurrency Control: Timestamp & Lock-Based Protocols

What is Concurrency Control?

Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other. It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database.

Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical Database, it would have a mix of READ and WRITE operations and hence the concurrency is a challenge.

DBMS Concurrency Control is used to address such conflicts, which mostly occur with a multi-user system. Therefore, Concurrency Control is the most important element for proper functioning of a Database Management System where two or more database transactions are executed simultaneously, which require access to the same data.

In this tutorial, you will learn

- [What is Concurrency Control?](#)
- [Potential problems of Concurrency](#)
- [Why use Concurrency method?](#)
- [Concurrency Control Protocols](#)
- [Lock-based Protocols](#)
- [Two Phase Locking \(2PL\) Protocol](#)
- [Timestamp-based Protocols](#)
- [Validation Based Protocol](#)
- [Characteristics of Good Concurrency Protocol](#)

Potential problems of Concurrency

Here, are some issues which you will likely to face while using the DBMS Concurrency Control method:

- **Lost Updates** occur when multiple transactions select the same row and update the row based on the value selected
- Uncommitted dependency issues occur when the second transaction selects a row which is updated by another transaction (**dirty read**)

- **Non-Repeatable Read** occurs when a second transaction is trying to access the same row several times and reads different data each time.
- **Incorrect Summary issue** occurs when one transaction takes summary over the value of all the instances of a repeated data-item, and second transaction update few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

Why use Concurrency method?

Reasons for using Concurrency control method is DBMS:

- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.
- Concurrency control helps to ensure serializability

Example

Assume that two people who go to electronic kiosks at the same time to buy a movie ticket for the same movie and the same show time.

However, there is only one seat left in for the movie show in that particular theatre. Without concurrency control in DBMS, it is possible that both moviegoers will end up purchasing a ticket. However, concurrency control method does not allow this to happen. Both moviegoers can still access information written in the movie seating database. But concurrency control only provides a ticket to the buyer who has completed the transaction process first.

Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose. Following are the Concurrency Control techniques in DBMS:

- Lock-Based Protocols
- Two Phase Locking Protocol
- Timestamp-Based Protocols
- Validation-Based Protocols

Lock-based Protocols

Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock. Lock based protocols help to eliminate the concurrency problem in DBMS for simultaneous transactions by locking or isolating a particular transaction to a single user.

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks in DBMS help synchronize access to the database items by concurrent transactions.

All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

Binary Locks: A Binary lock on a data item can either locked or unlocked states.

Shared/exclusive: This type of locking mechanism separates the locks in DBMS based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

1. Shared Lock (S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because you will never have permission to update data on the data item.

For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, shared lock prevent it until the reading process is over.

2. Exclusive Lock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction. Transactions may unlock the data item after finishing the 'write' operation.

For example, when a transaction needs to update the account balance of a person. You can allow this transaction by placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevents this operation.

3. Simplistic Lock Protocol

This type of lock-based protocol allows transactions to obtain a lock on every object before beginning operation. Transactions may unlock the data item after finishing the 'write' operation.

4. Pre-claiming Locking

Pre-claiming lock protocol helps to evaluate operations and create a list of required data items which are needed to initiate an execution process. In the situation when all locks are granted, the transaction executes. After that, all locks are released when all of its operations are over.

Starvation

Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.

Following are the reasons for Starvation:

- When waiting scheme for locked items is not properly managed
- In the case of resource leak
- The same transaction is selected as a victim repeatedly

Deadlock

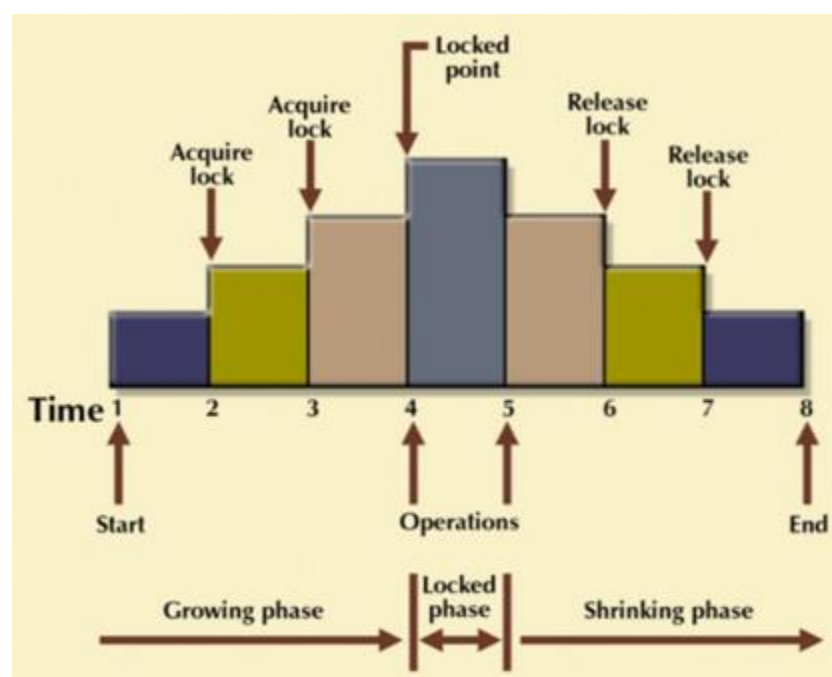
Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

Two Phase Locking Protocol

Two Phase Locking Protocol also known as 2PL protocol is a method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously. Two Phase Locking protocol helps to eliminate the concurrency problem in DBMS.

This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.



The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- **Growing Phase:** In this phase transaction may obtain locks but may not release any locks.

- **Shrinking Phase:** In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.

In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Centralized 2PL

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primary copy 2PL

Primary copy 2PL mechanism, many lock managers are distributed to different sites. After that, a particular lock manager is responsible for managing the lock for a set of data items. When the primary copy has been updated, the change is propagated to the slaves.

Distributed 2PL

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locks for data at that site. If no data is replicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL

Timestamp-based Protocols

Timestamp based Protocol in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions. The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order.

The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.

Example:

```
Suppose there are there transactions T1, T2, and T3.
T1 has entered the system at time 0010
T2 has entered the system at 0020
T3 has entered the system at 0030
Priority will be given to transaction T1, then transaction T2 and lastly Transaction T3.
```

Advantages:

- Schedules are serializable just like 2PL protocols
- No waiting for the transaction, which eliminates the possibility of deadlocks!

Disadvantages:

Starvation is possible if the same transaction is restarted and continually aborted

Validation Based Protocol

Validation based Protocol in DBMS also known as Optimistic Concurrency Control Technique is a method to avoid concurrency in transactions. In this protocol, the local copies of the transaction data are updated rather than the data itself, which results in less interference while execution of the transaction.

The Validation based Protocol is performed in the following three phases:

1. Read Phase
2. Validation Phase
3. Write Phase

Read Phase

In the Read Phase, the data values from the database can be read by a transaction but the write operation or updates are only applied to the local data copies, not the actual database.

Validation Phase

In Validation Phase, the data is checked to ensure that there is no violation of serializability while applying the transaction updates to the database.

Write Phase

In the Write Phase, the updates are applied to the database if the validation is successful, else; the updates are not applied, and the transaction is rolled back.

Characteristics of Good Concurrency Protocol

An ideal concurrency control DBMS mechanism has the following objectives:

- Must be resilient to site and communication failures.
- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.
- It must enforce some constraints on the structure of atomic actions of transactions.

Summary

- Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another.
- Lost Updates, dirty read, Non-Repeatable Read, and Incorrect Summary Issue are problems faced due to lack of concurrency control.
- Lock-Based, Two-Phase, Timestamp-Based, Validation-Based are types of Concurrency handling protocols
- The lock could be Shared (S) or Exclusive (X)
- Two-Phase locking protocol which is also known as a 2PL protocol needs transaction should acquire a lock after it releases one of its locks. It has 2 phases growing and shrinking.
- The timestamp-based algorithm uses a timestamp to serialize the execution of concurrent transactions. The protocol uses the **System Time or Logical Count as** a Timestamp.

DBMS Keys: Candidate, Super, Primary, Foreign Key Types with Example

What are Keys in DBMS?

KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

In this tutorial, you will learn:

- [What are Keys?](#)
- [Why we need a Key?](#)
- [Various Keys in Database Management System](#)
- [What is Super key?](#)
- [What is Primary Key?](#)
- [What is Alternate key?](#)
- [What is Candidate Key?](#)

- [What is Foreign key?](#)
- [What is Compound key?](#)
- [What is Composite key?](#)
- [What is Surrogate Key?](#)
- [Difference Between Primary key & Foreign key](#)

Why we need a Key?

Here are some reasons for using sql key in the DBMS system.

- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.

Types of Keys in DBMS (Database Management System)

There are mainly Eight different types of Keys in DBMS and each key has it’s different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Let’s look at each of the keys in DBMS with example:

- **Super Key** – A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key** – is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key** – is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key** – is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key** – is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key** – has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key** – is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.
- **Surrogate Key** – An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don’t have any natural primary key.

What is the Super key?

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

What is a Primary Key?

PRIMARY KEY in [DBMS](#) is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can’t be a duplicate meaning the same value can’t appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can’t have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, <code>StudID</code> is a Primary Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

What is the Alternate key?

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.

Example:

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

What is a Candidate Key?

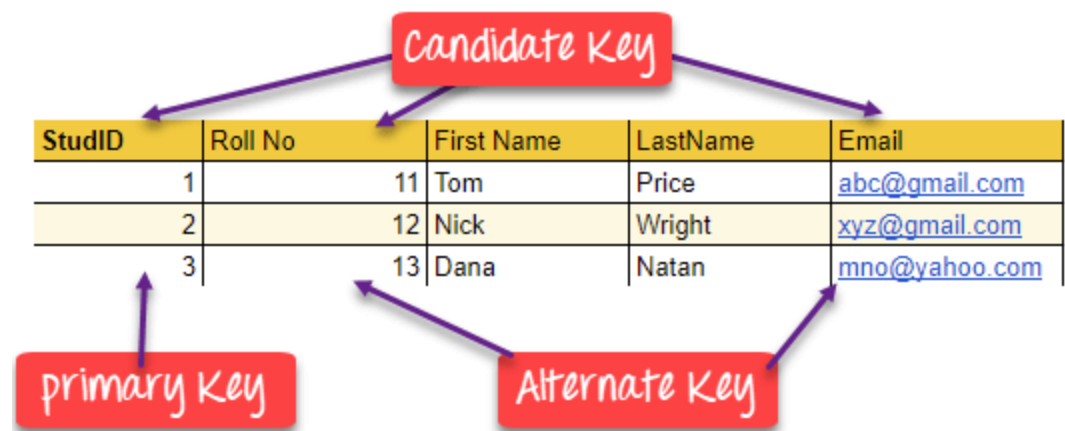
CANDIDATE KEY in SQL is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

Properties of Candidate key:

- It must contain unique values
- Candidate key in SQL may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Candidate key Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



Candidate Key in DBMS

What is the Foreign key?

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

Example:

DeptCode	DeptName	
001	Science	
002	English	
005	Computer	

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this key in dbms example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

What is the Compound key?

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the table.

Example:

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

What is the Composite key?

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table. The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

What is a Surrogate key?

SURROGATE KEYS is An artificial key which aims to uniquely identify each record is called a surrogate key. This kind of partial key in dbms is unique because it is created when you don’t have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key in DBMS is usually an integer. A surrogate key is a value generated right before the record is inserted into a table.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys in sql are allowed when

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

Difference Between Primary key & Foreign key

Following is the main difference between primary key and foreign key:

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accept null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.

Summary

- What is key in DBMS: A key in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table)
- Keys in RDBMS allow you to establish a relationship between and identify the relation between tables
- Eight types of key in DBMS are Super, Primary, Candidate, Alternate, Foreign, Compound, Composite, and Surrogate Key.
- A super key is a group of single or multiple keys which identifies rows in a table.
- A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key
- All the different keys in DBMS which are not primary key are called an alternate key
- A super key with no repeated attribute is called candidate key
- A compound key is a key which has many fields which allow you to uniquely recognize a specific record
- A key which has multiple attributes to uniquely identify rows in a table is called a composite key
- An artificial key which aims to uniquely identify each record is called a surrogate key
- Primary Key never accept null values while a foreign key may accept multiple null values.

Functional Dependency in DBMS: What is, Types and Examples

What is Functional Dependency?

Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System (DBMS). Functional Dependency helps to maintain the quality of data in the database. It plays a vital role to find the difference between good and bad database design. A functional dependency is denoted by an arrow “ \rightarrow ”. The functional dependency of X on Y is represented by $X \rightarrow Y$. Let’s understand Functional Dependency in DBMS with example.

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

In this tutorial, you will learn:

- [Key terms](#)
- [Rules of Functional Dependencies](#)
- [Types of Functional Dependencies in DBMS](#)
- [Multivalued dependency in DBMS](#)
- [Trivial Functional dependency in DBMS](#)
- [Non trivial Functional dependency in DBMS](#)
- [Transitive dependency in DBMS](#)
- [What is Normalization?](#)
- [Advantages of Functional Dependency](#)

Key terms

Here, are some key terms for Functional Dependency in Database:

Key Terms	Description
Axiom	Axioms is a set of inference rules used to infer all the functional dependencies on a relational database.
Decomposition	It is a rule that suggests if you have a table that appears to contain two entities which are determined by the same primary key then you should consider breaking them up into two different tables.
Dependent	It is displayed on the right side of the functional dependency diagram.
Determinant	It is displayed on the left side of the functional dependency Diagram.
Union	It suggests that if two tables are separate, and the PK is the same, you should consider putting them. together

Rules of Functional Dependencies

Below are the Three most important rules for Functional Dependency in Database:

- Reflexive rule –. If X is a set of attributes and Y is_subset_of X, then X holds a value of Y.
- Augmentation rule: When $x \rightarrow y$ holds, and c is attribute set, then $ac \rightarrow bc$ also holds. That is adding attributes which do not change the basic dependencies.
- Transitivity rule: This rule is very much similar to the transitive rule in algebra if $x \rightarrow y$ holds and $y \rightarrow z$ holds, then $x \rightarrow z$ also holds. $X \rightarrow y$ is called as functionally that determines y.

Types of Functional Dependencies in DBMS

There are mainly four types of Functional Dependency in DBMS. Following are the types of Functional Dependencies in DBMS:

- **Multivalued Dependency**
- **Trivial Functional Dependency**
- **Non-Trivial Functional Dependency**

- **Transitive Dependency**

Multivalued Dependency in DBMS

Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table. A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation. Consider the following Multivalued Dependency Example to understand.

Example:

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, maf_year and color are independent of each other but dependent on car_model. In this example, these two columns are said to be multivalue dependent on car_model.

This dependence can be represented like this:

car_model -> maf_year

car_model-> colour

Trivial Functional Dependency in DBMS

The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.

So, X -> Y is a trivial functional dependency if Y is a subset of X. Let’s understand with a Trivial Functional Dependency Example.

For example:

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Consider this table of with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

Non Trivial Functional Dependency in DBMS

Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A. In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

Example:

{Company} -> {CEO} (if we know the Company, we knows the CEO name)

But CEO is not a subset of Company, and hence it’s non-trivial functional dependency.

Transitive Dependency in DBMS

A Transitive Dependency is a type of functional dependency which happens when “t” is indirectly formed by two functional dependencies. Let’s understand with the following Transitive Dependency Example.

Example:

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

{Company} -> {CEO} (if we know the compay, we know its CEO’s name)

{CEO } -> {Age} If we know the CEO, we know the Age

Therefore according to the rule of rule of transitive dependency:

{ Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Note: You need to remember that transitive dependency can only occur in a relation of three or more attributes.

What is Normalization?

Normalization is a method of organizing the data in the database which helps you to avoid data redundancy, insertion, update & deletion anomaly. It is a process of analyzing the relation schemas based on their different functional dependencies and primary key.

Normalization is inherent to relational database theory. It may have the effect of duplicating the same data within the database which may result in the creation of additional tables.

Advantages of Functional Dependency

- Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
- It helps you to maintain the quality of data in the database
- It helps you to defined meanings and constraints of databases
- It helps you to identify bad designs
- It helps you to find the facts regarding the database design

Summary

- Functional Dependency is when one attribute determines another attribute in a DBMS system.
- Axiom, Decomposition, Dependent, Determinant, Union are key terms for functional dependency
- Four types of functional dependency are 1) Multivalued 2) Trivial 3) Non-trivial 4) Transitive
- Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table
- The Trivial dependency occurs when a set of attributes which are called a trivial if the set of attributes are included in that attribute
- Nontrivial dependency occurs when A->B holds true where B is not a subset of A
- A transitive is a type of functional dependency which happens when it is indirectly formed by two functional dependencies
- Normalization is a method of organizing the data in the database which helps you to avoid data redundancy

Data Independence in DBMS: Physical & Logical with Examples

What is Data Independence of DBMS?

Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level. Data independence helps you to keep data separated from all programs that make use of it.

You can use this stored data for computing and presentation. In many systems, data independence is an essential function for components of the system.

In this tutorial, you will learn:

- [What is Data Independence of DBMS?](#)
- [Types of Data Independence](#)
- [Levels of Database](#)
- [Physical Data Independence](#)
- [Logical Data Independence](#)
- [Difference between Physical and Logical Data Independence](#)
- [Importance of Data Independence](#)

Types of Data Independence

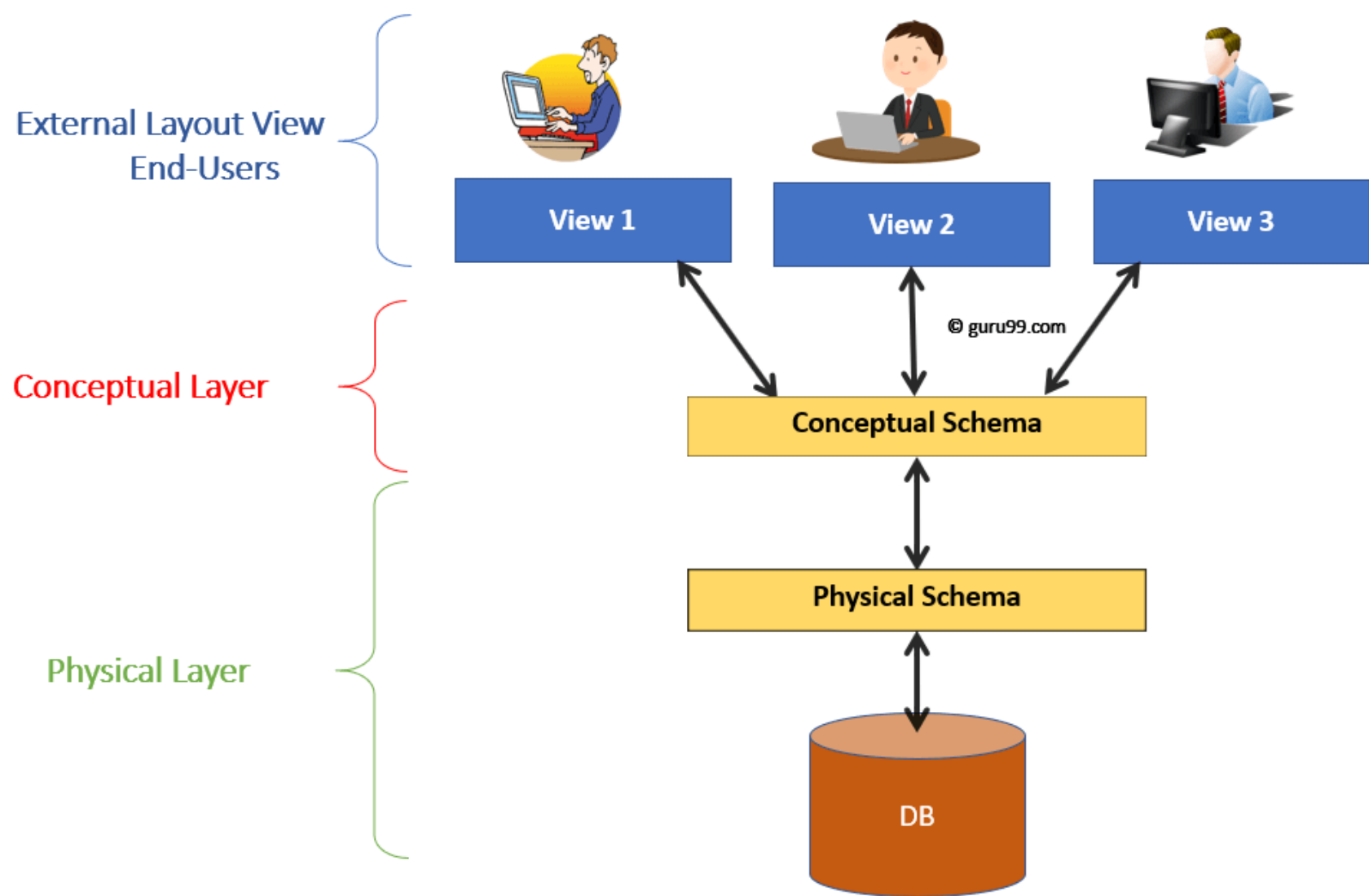
In DBMS there are two types of data independence

1. Physical data independence
2. Logical data independence.

Levels of Database

Before we learn Data Independence, a refresher on Database Levels is important. The database has 3 levels as shown in the diagram below

1. Physical/Internal
2. Conceptual
3. External



Levels of DBMS Architecture Diagram

Consider an Example of a University Database. At the different levels this is how the implementation will look like:

Type of Schema	Implementation
External Schema	View 1: Course info(cid:int,cname:string) View 2: studeninfo(id:int. name:string)
Conceptual Shema	Students(id: int, name: string, login: string, age: integer) Courses(id: int, cname:string, credits:integer) Enrolled(id: int, grade:string)
Physical Schema	<ul style="list-style-type: none">• Relations stored as unordered files.

- Index on the first column of Students.

Physical Data Independence

Physical data independence helps you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures. Compared to Logical Independence, it is easy to achieve physical data independence.

With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

Examples of changes under Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

Logical Data Independence

Logical Data Independence is the ability to change the conceptual scheme without changing

1. External views
2. External API or programs

Any change made will be absorbed by the mapping between external and conceptual levels.

When compared to Physical Data independence, it is challenging to achieve logical data independence.

Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

1. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
2. Merging two records into one
3. Breaking an existing record into two or more records

Difference between Physical and Logical Data Independence

Logica Data Independence	Physical Data Independence
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.
Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.
Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

Summary

- Data Independence is the property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Two levels of data independence are 1) Physical and 2) Logical
- Physical data independence helps you to separate conceptual levels from the internal/physical levels
- Logical Data Independence is the ability to change the conceptual scheme without changing
- When compared to Physical Data independence, it is challenging to achieve logical data independence
- Data Independence Helps you to improve the quality of the data

Hashing in DBMS: Static and Dynamic Hashing Techniques

What is Hashing in DBMS?

In DBMS, hashing is a technique to directly search the location of desired data on the disk without using index structure. Hashing method is used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value. Data is stored in the form of data blocks whose address is generated by applying a hash function in the memory location where these records are stored known as a **data block or data bucket**.

In this DBMS tutorial, you will learn,

- [What is Hashing technique in DBMS?](#)
- [Why do we need Hashing?](#)
- [Important Terminologies in Hashing](#)
- [Types of Hashing Techniques](#)
- [Static Hashing](#)
- [Dynamic Hashing](#)
- [Difference between Ordered Indexing and Hashing](#)
- [What is Collision?](#)
- [How to deal with Hashing Collision?](#)

Why do we need Hashing?

Here, are the situations in the DBMS where you need to apply the Hashing method:

- For a huge database structure, it's tough to search all the index values through all its level and then you need to reach the destination data block to get the desired data.
- Hashing method is used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value.
- Hashing is an ideal method to calculate the direct location of a data record on the disk without using index structure.
- It is also a helpful technique for implementing dictionaries.

Important Terminologies in Hashing

Here, are important terminologies which are used in Hashing:

- **Data bucket** – Data buckets are memory locations where the records are stored. It is also known as Unit Of Storage.
- **Key:** A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table). This allows you to find the relationship between two tables.

- **Hash function:** A hash function, is a mapping function which maps all the set of search keys to the address where actual records are placed.
- **Linear Probing** – Linear probing is a fixed interval between probes. In this method, the next available data block is used to enter the new record, instead of overwriting on the older record.
- **Quadratic probing**– It helps you to determine the new bucket address. It helps you to add Interval between probes by adding the consecutive output of quadratic polynomial to starting value given by the original computation.
- **Hash index** – It is an address of the data block. A hash function could be a simple mathematical function to even a complex mathematical function.
- **Double Hashing** –Double hashing is a computer programming method used in hash tables to resolve the issues of has a collision.
- **Bucket Overflow:** The condition of bucket-overflow is called collision. This is a fatal stage for any static has to function.

Types of Hashing Techniques

There are mainly two types of SQL hashing methods/techniques:

1. Static Hashing
2. Dynamic Hashing

Static Hashing

In the static hashing, the resultant data bucket address will always remain the same.

Therefore, if you generate an address for say **Student_ID = 10** using hashing function **mod(3)**, the resultant bucket address will always be **1**. So, you will not see any change in the bucket address.

Therefore, in this static hashing method, the number of data buckets in memory always remains constant.

Static Hash Functions

- **Inserting a record:** When a new record requires to be inserted into the table, you can generate an address for the new record using its hash key. When the address is generated, the record is automatically stored in that location.
- **Searching:** When you need to retrieve the record, the same hash function should be helpful to retrieve the address of the bucket where data should be stored.
- **Delete a record:** Using the hash function, you can first fetch the record which is you wants to delete. Then you can remove the records for that address in memory.

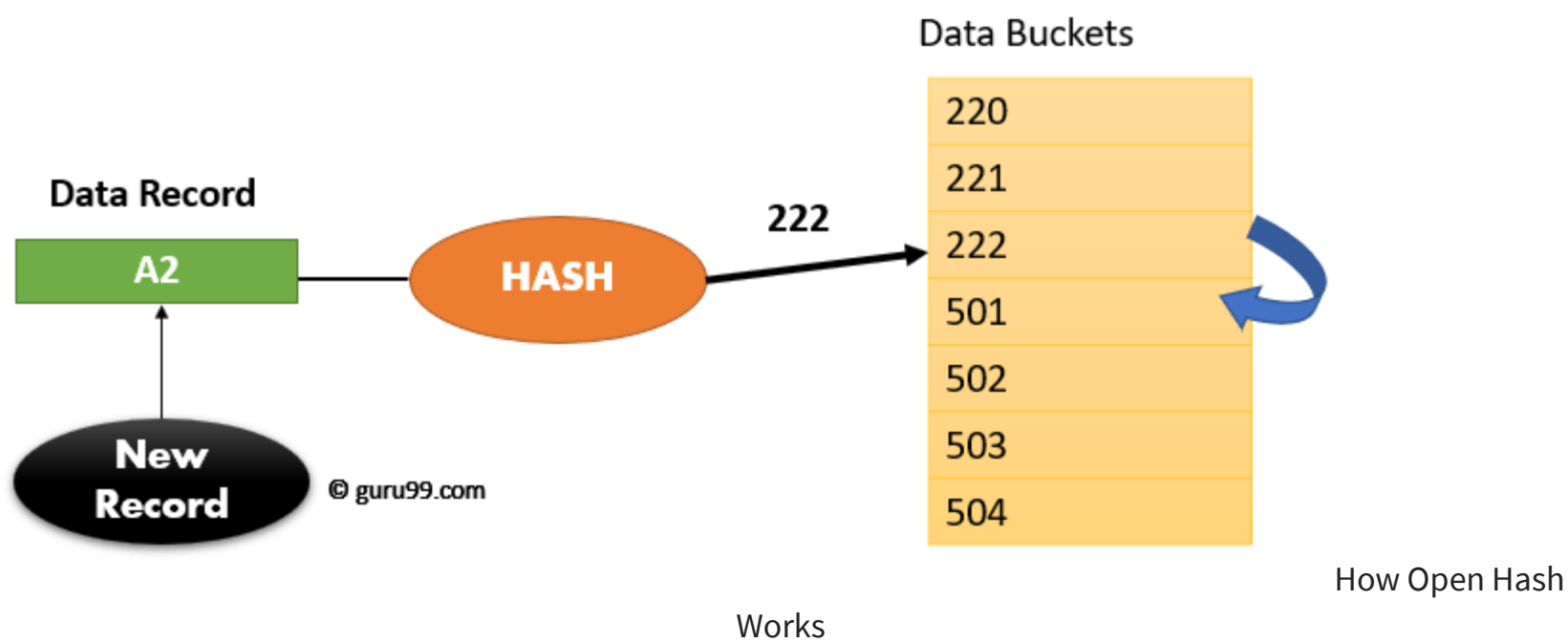
Static hashing is further divided into

1. Open hashing
2. Close hashing.

Open Hashing

In Open hashing method, Instead of overwriting older one the next available data block is used to enter the new record, This method is also known as linear probing.

For example, A2 is a new record which you wants to insert. The hash function generates address as 222. But it is already occupied by some other value. That's why the system looks for the next data bucket 501 and assigns A2 to it.



Close Hashing

In the close hashing method, when buckets are full, a new bucket is allocated for the same hash and result are linked after the previous one.

Dynamic Hashing

Dynamic hashing offers a mechanism in which data buckets are added and removed dynamically and on demand. In this hashing, the hash function helps you to create a large number of values.

Difference between Ordered Indexing and Hashing

Below are the key differences between Indexing and Hashing

Parameters	Order Indexing	Hashing
Storing of address	Addresses in the memory are sorted according to a key value called the primary key	Addresses are always generated using a hash function on the key value.
Performance	It can decrease when the data increases in the hash file. As it stores the data in a sorted form when there is any (insert/delete/update) operation performed which decreases its performance.	Performance of hashing will be best when there is a constant addition and deletion of data. However, when the database is huge, then hash file organization and its maintenance will be costlier.
Use for	Preferred for range retrieval of data- which means whenever there is retrieval data for a particular range, this method is an ideal option.	This is an ideal method when you want to retrieve a particular record based on the search key. However, it will only perform well when the hash function is on the search key.
Memory management	There will be many unused data blocks because of the delete/update operation. These data blocks can't be released for re-use. That's why regular maintenance of the memory is required.	In static and dynamic hashing methods, memory is always managed. Bucket overflow is also handled perfectly to extend static hashing.

What is Collision?

Hash collision is a state when the resultant hashes from two or more data in the data set, wrongly map the same place in the [hash table](#).

How to deal with Hashing Collision?

There are two technique which you can use to avoid a hash collision:

1. **Rehashing:** This method, invokes a secondary hash function, which is applied continuously until an empty slot is found, where a record should be placed.
2. **Chaining:** Chaining method builds a Linked list of items whose key hashes to the same value. This method requires an extra link field to each table position.

Summary:

- In [DBMS](#), hashing is a technique to directly search the location of desired data on the disk without using index structure.
- Hashing method is used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value.

- Data bucket, Key , Hash function, Linear Probing, Quadratic probing , Hash index, Double Hashing, Bucket Overflow are important terminologies used in hashing
- Two types of hashing methods are 1) static hashing 2) dynamic hashing
- In the static hashing, the resultant data bucket address will always remain the same.
- Dynamic hashing offers a mechanism in which data buckets are added and removed dynamically and on demand.
- In order Indexing addresses in the memory are sorted according to a critical value while in hashing addresses are always generated using a hash function on the key value.
- Hash collision is a state when the resultant hashes from two or more data in the data set, wrongly map the same place in the hash table.
- Rehashing and chaining are two methods which help you to avoid hashing collision.

SQL Commands: DML, DDL, DCL, TCL, DQL with Query Example

What is SQL?

SQL is a database language designed for the retrieval and management of data in a relational database.

SQL is the standard language for database management. All the RDBMS systems like MySQL, MS Access, Oracle, Sybase, Postgres, and SQL Server use SQL as their standard database language. SQL programming language uses various commands for different operations. We will learn about the like DCL, TCL, DQL, DDL and DML commands in SQL with examples.

In this SQL commands in DBMS tutorial, you will learn:

- [What is SQL?](#)
- [Why Use SQL?](#)
- [Brief History of SQL](#)
- [Types of SQL](#)
- [What is DDL?](#)
- [What is Data Manipulation Language?](#)
- [What is DCL?](#)
- [What is TCL?](#)
- [What is DQL?](#)

Why Use SQL?

Here, are important reasons for using SQL

- It helps users to access data in the RDBMS system.
- It helps you to describe the data.
- It allows you to define the data in a database and manipulate that specific data.
- With the help of SQL commands in DBMS, you can create and drop databases and tables.
- SQL offers you to use the function in a database, create a view, and stored procedure.
- You can set permissions on tables, procedures, and views.

Brief History of SQL

Here, are important landmarks from the history of SQL:

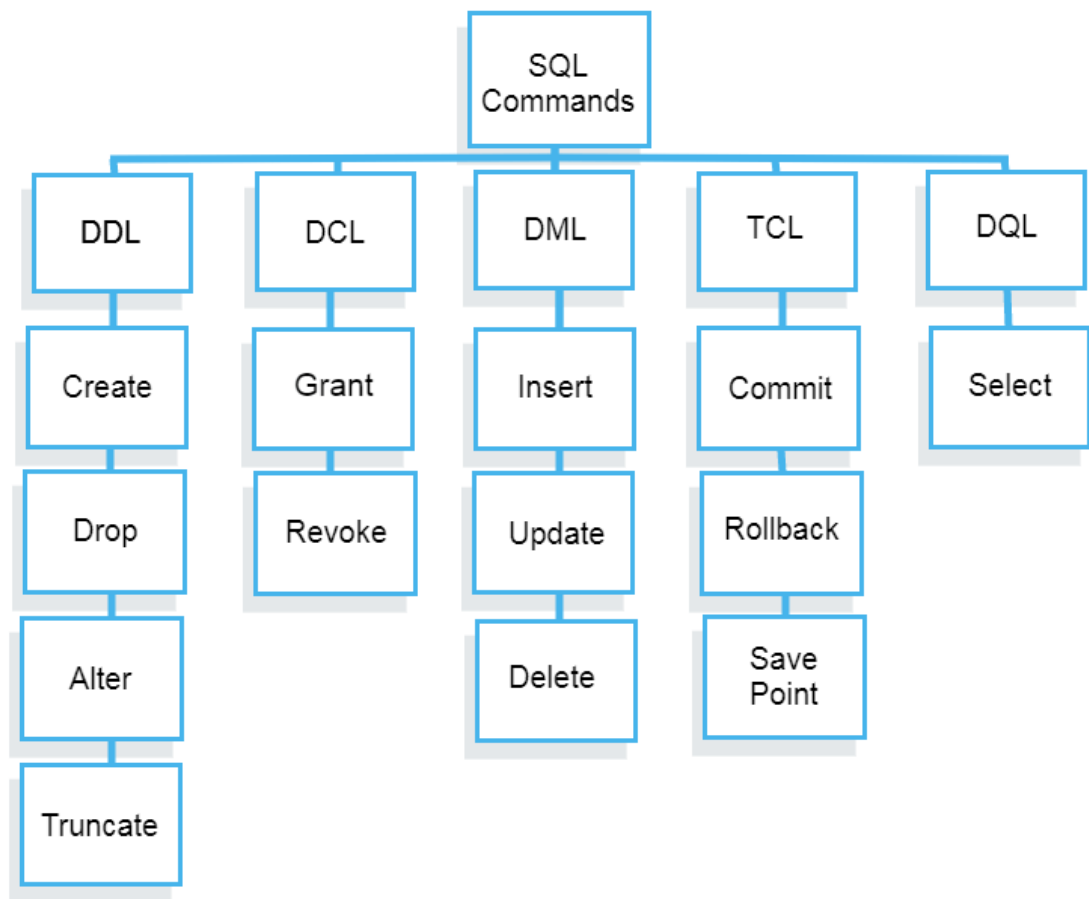
- 1970 – Dr. Edgar F. “Ted” Codd described a relational model for databases.
- 1974 – Structured Query Language appeared.
- 1978 – IBM released a product called System/R.
- 1986 – IBM developed the prototype of a relational database, which is standardized by ANSI.
- 1989- First ever version launched of SQL
- 1999 – SQL 3 launched with features like triggers, object-orientation, etc.
- SQL2003- window functions, XML-related features, etc.
- SQL2006- Support for XML Query Language
- SQL2011-improved support for temporal databases

Types of SQL

Here are five types of widely used SQL queries.

- Data Definition Language (DDL)

- Data Manipulation Language (DML)
- Data Control Language(DCL)
- Transaction Control Language(TCL)
- Data Query Language (DQL)



Types of SQL

Let see each of them in detail:

What is DDL?

Data Definition Language helps you to define the database structure or schema. Let’s learn about DDL commands with syntax.

Five types of DDL commands in SQL are:

CREATE

CREATE statements is used to define the database structure schema:

Syntax:

```
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
```

For example:

```
Create database university;
Create table students;
Create view for_students;
```

DROP

Drops commands remove tables and databases from RDBMS.

Syntax

```
DROP TABLE ;
```

For example:

```
Drop object_type object_name;
Drop database university;
Drop table student;
```

ALTER

Alters command allows you to alter the structure of the database.

Syntax:

To add a new column in the table

```
ALTER TABLE table_name ADD column_name COLUMN-definition;
```

To modify an existing column in the table:

```
ALTER TABLE MODIFY (COLUMN DEFINITION....);
```

For example:

```
Alter table guru99 add subject varchar;
```

TRUNCATE:

This command used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE table students;
```

What is Data Manipulation Language?

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

There are three basic constructs which allow database program and user to enter data and information are:

Here are some important DML commands in SQL:

- INSERT
- UPDATE
- DELETE

INSERT:

This is a statement is a SQL query. This command is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME (col1, col2, col3,... col N)
VALUES (value1, value2, value3, .... valueN);
Or
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, .... valueN);
```

For example:

```
INSERT INTO students (RollNo, FirstName, LastName) VALUES ('60', 'Tom', Erichsen');
```

UPDATE:

This command is used to update or modify the value of a column in the table.

Syntax:

```
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
```

For example:

```
UPDATE students
SET FirstName = 'Jhon', LastName= 'Wick'
WHERE StudID = 3;
```

DELETE:

This command is used to remove one or more rows from a table.

Syntax:

```
DELETE FROM table_name [WHERE condition];
```

For example:

```
DELETE FROM students
WHERE FirstName = 'Jhon';
```

What is DCL?

DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give “rights & permissions.” Other permission controls parameters of the database system.

Examples of DCL commands:

Commands that come under DCL:

- Grant
- Revoke

Grant:

This command is use to give user access privileges to a database.

Syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

For example:

```
GRANT SELECT ON Users TO 'Tom'@'localhost';
```

Revoke:

It is useful to back permissions from the user.

Syntax:

```
REVOKE privilege_name ON object_name FROM {user_name | PUBLIC | role_name}
```

For example:

```
REVOKE SELECT, UPDATE ON student FROM BCA, MCA;
```

What is TCL?

Transaction control language or TCL commands deal with the transaction within the database.

Commit

This command is used to save all the transactions to the database.

Syntax:

```
Commit;
```

For example:

```
DELETE FROM Students  
WHERE RollNo =25;  
COMMIT;
```

Rollback

Rollback command allows you to undo transactions that have not already been saved to the database.

Syntax:

```
ROLLBACK;
```

Example:

```
DELETE FROM Students  
WHERE RollNo =25;
```

SAVEPOINT

This command helps you to sets a savepoint within a transaction.

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

Example:

```
SAVEPOINT RollNo;
```

What is DQL?

Data Query Language (DQL) is used to fetch the data from the database. It uses only one command:

SELECT:

This command helps you to select the attribute based on the condition described by the WHERE clause.

Syntax:

```
SELECT expressions  
FROM TABLES  
WHERE conditions;
```

For example:

```
SELECT FirstName  
FROM Student  
WHERE RollNo > 15;
```

Summary:

- SQL is a database language designed for the retrieval and management of data in a relational database.
- It helps users to access data in the RDBMS system
- In the year 1974, the term Structured Query Language appeared
- Five types of SQL queries are 1) Data Definition Language (DDL) 2) Data Manipulation Language (DML) 3) Data Control Language(DCL) 4) Transaction Control Language(TCL) and, 5) Data Query Language (DQL)
- Data Definition Language(DDL) helps you to define the database structure or schema.
- Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data.
- DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give “rights & permissions.”
- Transaction control language or TCL commands deal with the transaction within the database.
- Data Query Language (DQL) is used to fetch the data from the [database](#).

DBMS Joins: Inner, THETA, Outer, Equi Types of Join Operations

What is Join in DBMS?

Join in DBMS is a binary operation which allows you to combine join product and selection in one single statement. The goal of creating a join condition is that it helps you to combine the data from two or more DBMS tables. The tables in DBMS are associated using the primary key and foreign keys.

In this DBMS tutorial, you will learn:

- [Types of Join](#)
- [Inner Join](#)
- [Theta Join](#)
- [EQUI join:](#)
- [Natural Join \(\$\bowtie\$ \)](#)
- [Outer Join](#)
- [Left Outer Join \(\$A \bowtie\!\!\!\!\!\lrcorner B\$ \)](#)
- [Right Outer Join \(\$A \bowtie\!\!\!\!\!\rceil B\$ \)](#)
- [Full Outer Join \(\$A \bowtie\!\!\!\!\!\Bumpeq B\$ \)](#)

Types of Join

There are mainly two types of joins in DBMS:

1. Inner Joins: Theta, Natural, EQUI
2. Outer Join: Left, Right, Full

Let’s see them in detail:

Inner Join

Inner Join is used to return rows from both tables which satisfy the given condition. It is the most widely used join operation and can be considered as a default join-type

An Inner join or equijoin is a comparator-based join which uses equality comparisons in the join-predicate. However, if you use other comparison operators like “>” it can’t be called equijoin.

Inner Join further divided into three subtypes:

- Theta join
- Natural join
- EQUI join

Theta Join

Theta Join allows you to merge two tables based on the condition represented by theta. Theta joins work for all comparison operators. It is denoted by symbol θ . The general case of JOIN operation is called a Theta join.

Syntax:

$A \bowtie_{\theta} B$

Theta join can use any conditions in the selection criteria.

Consider the following tables.

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

For example:

$A \bowtie_{A.column\ 2 > B.column\ 2} B$

$A \bowtie A.column\ 2 > B.column\ 2\ (B)$

column 1	column 2
1	2

EQUI Join

EQUI Join is done when a Theta join uses only the equivalence condition. EQUI join is the most difficult operation to implement efficiently in an RDBMS, and one reason why RDBMS have essential performance problems.

For example:

$A \bowtie_{A.column\ 2 = B.column\ 2} B$

$A \bowtie A.column\ 2 = B.column\ 2\ (B)$

column 1	column 2
1	1

Natural Join (\bowtie)

Natural Join does not utilize any of the comparison operators. In this type of join, the attributes should have the same name and domain. In Natural Join, there should be at least one common attribute between two relations. It performs selection forming equality on those attributes which appear in both relations and eliminates the duplicate attributes.

Example:

Consider the following two tables

C	
Num	Square
2	4
3	9

D	
Num	Cube
2	8
3	18

$C \bowtie D$

$C \bowtie D$

Num	Square	Cube
2	4	8
3	9	18

Outer Join

An **Outer Join** doesn't require each record in the two join tables to have a matching record. In this type of join, the table retains each record even if no other matching record exists.

Three types of Outer Joins are:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join (A ⋈ B)

Left Outer Join returns all the rows from the table on the left even if no matching rows have been found in the table on the right. When no matching record is found in the table on the right, NULL is returned.



Consider the following 2 Tables

A

Num	Square
2	4
3	9
4	16

B

Num	Cube
2	8
3	18
5	75

A ⋈ B

A ⋈ B

Num	Square	Cube
2	4	8
3	9	18
4	16	–

Right Outer Join (A ⋈ B)

Right Outer Join returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned. RIGHT outer JOIN is the opposite of LEFT JOIN

In our example, let’s assume that you need to get the names of members and movies rented by them. Now we have a new member who has not rented any movie yet.



A ⋈ B

A ⋈ B

Num	Cube	Square
2	8	4
3	18	9
5	75	–

Full Outer Join (A ⋈ B)

In a **Full Outer Join** , all tuples from both relations are included in the result, irrespective of the matching condition.

Example:

A  B
A ⋈ B

Num	Square	Cube
2	4	8
3	9	18
4	16	–
5	–	75

Summary:

- There are mainly two types of joins in DBMS 1) Inner Join 2) Outer Join
- An inner join is the widely used join operation and can be considered as a default join-type.
- Inner Join is further divided into three subtypes: 1) Theta join 2) Natural join 3) EQUI join
- Theta Join allows you to merge two tables based on the condition represented by theta
- When a theta join uses only equivalence condition, it becomes an equi join.
- Natural join does not utilize any of the comparison operators.
- An outer join doesn’t require each record in the two join tables to have a matching record.
- Outer Join is further divided into three subtypes are: 1)Left Outer Join 2) Right Outer Join 3) Full Outer Join
- The LEFT Outer Join returns all the rows from the table on the left, even if no matching rows have been found in the table on the right.
- The RIGHT Outer Join returns all the columns from the table on the right, even if no matching rows have been found in the table on the left.
- In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

Indexing in DBMS: What is, Types of Indexes with EXAMPLES

What is Indexing?

Indexing is a data structure technique which allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of [pointers](#) for holding the address of the disk block where that specific key value stored.

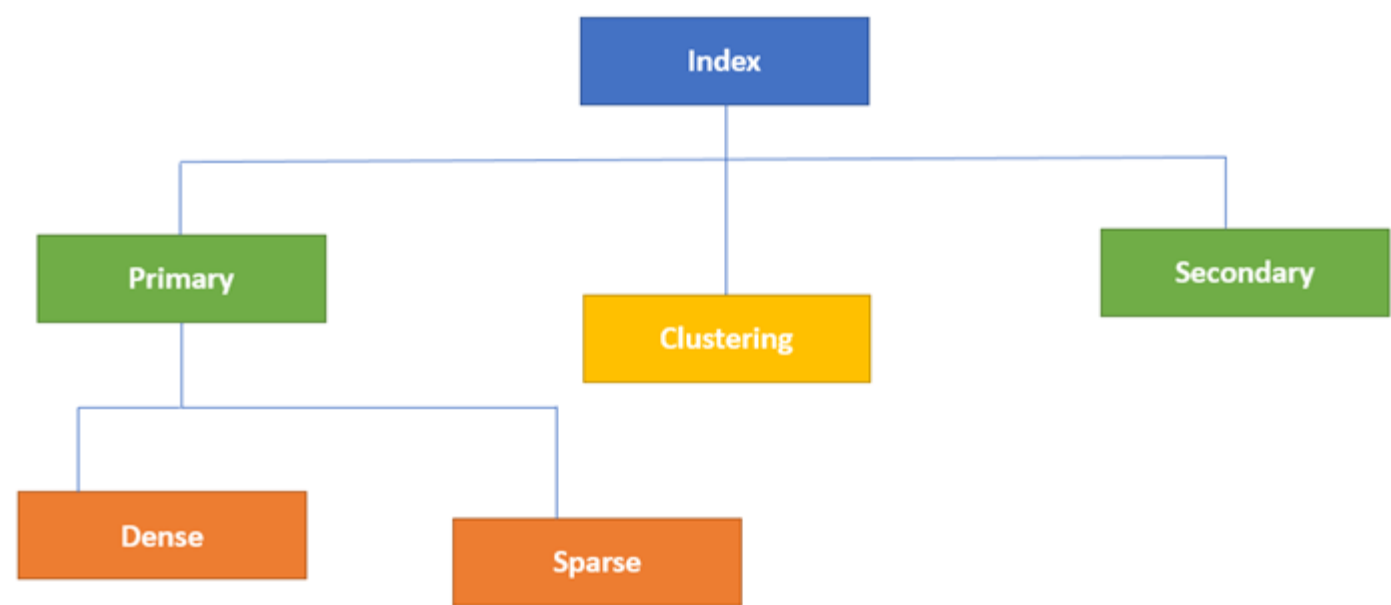
An index –

- Takes a search key as input
- Efficiently returns a collection of matching records.

In this DBMS Indexing tutorial, you will learn:

- [Types of Indexing in DBMS](#)
- [Primary Index in DBMS](#)
- [Secondary Index in DBMS](#)
- [Clustering Index in DBMS](#)
- [What is Multilevel Index?](#)
- [B-Tree Index](#)
- [Advantages of Indexing](#)
- [Disadvantages of Indexing](#)

Types of Indexing in DBMS



Type of Indexes in

Database

Indexing in Database is defined based on its indexing attributes. Two main types of indexing methods are:

- Primary Indexing
- Secondary Indexing

Primary Index in DBMS

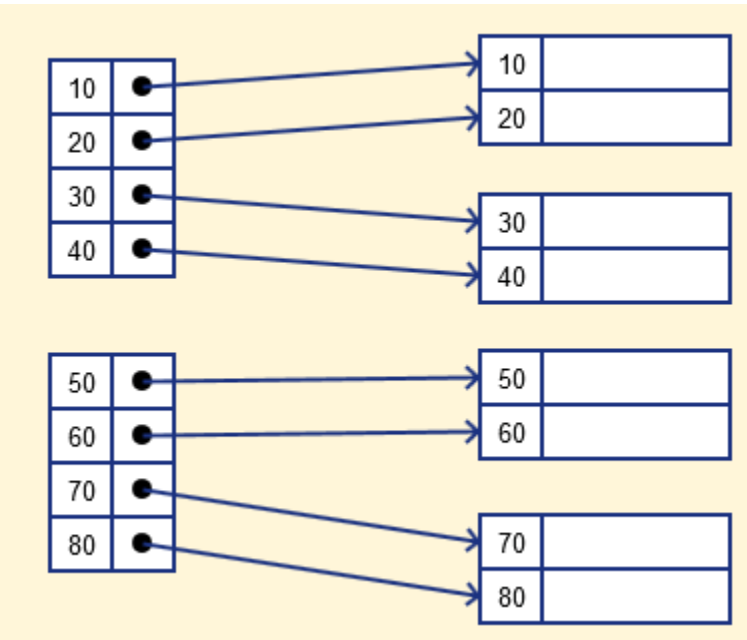
Primary Index is an ordered file which is fixed length size with two fields. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

The primary Indexing in DBMS is also further divided into two types.

- Dense Index
- Sparse Index

Dense Index

In a dense index, a record is created for every search key valued in the database. This helps you to search faster but needs more space to store index records. In this Indexing, method records contain search key value and points to the real record on the disk.

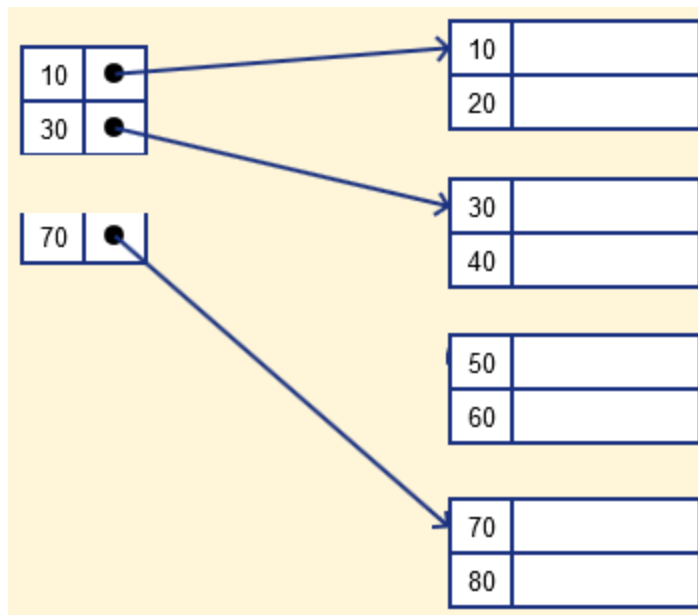


Sparse Index

It is an index record that appears for only some of the values in the file. Sparse Index helps you to resolve the issues of dense Indexing in DBMS. In this method of indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.

However, sparse Index stores index records for only some search-key values. It needs less space, less maintenance overhead for insertion, and deletions but It is slower compared to the dense Index for locating records.

Below is an database index Example of Sparse Index



Secondary Index in DBMS

The secondary Index in DBMS can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.

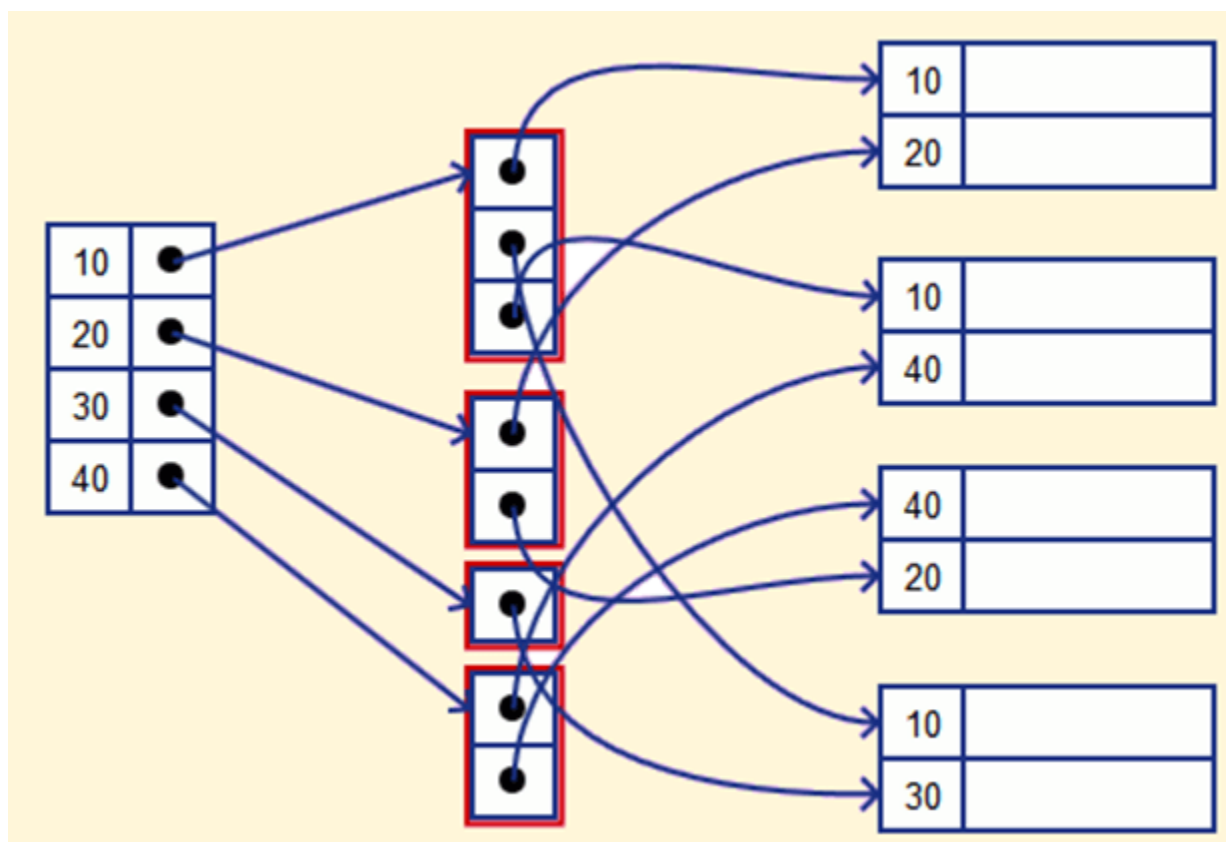
This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

Secondary Index Example

Let's understand secondary indexing with a database index example:

In a bank account database, data is stored sequentially by acc_no; you may want to find all accounts in of a specific branch of ABC bank.

Here, you can have a secondary index in DBMS for every search-key. Index record is a record point to a bucket that contains pointers to all the records with their specific search-key value.



Clustering Index in DBMS

In a clustered index, records themselves are stored in the Index and not pointers. Sometimes the Index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

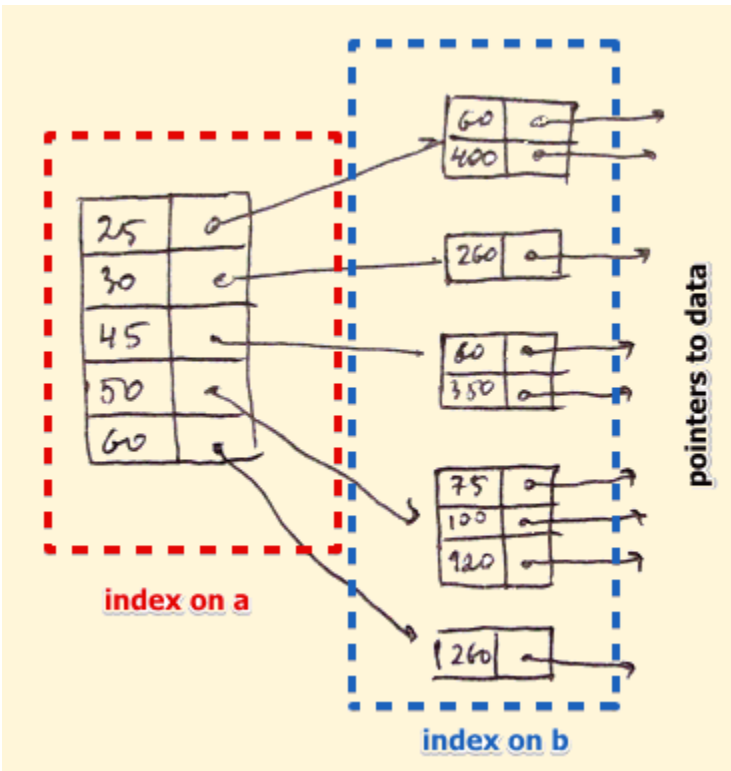
Example:

Let's assume that a company recruited many employees in various departments. In this case, clustering indexing in DBMS should be created for all employees who belong to the same dept.

It is considered in a single cluster, and index points point to the cluster as a whole. Here, Department_no is a non-unique key.

What is Multilevel Index?

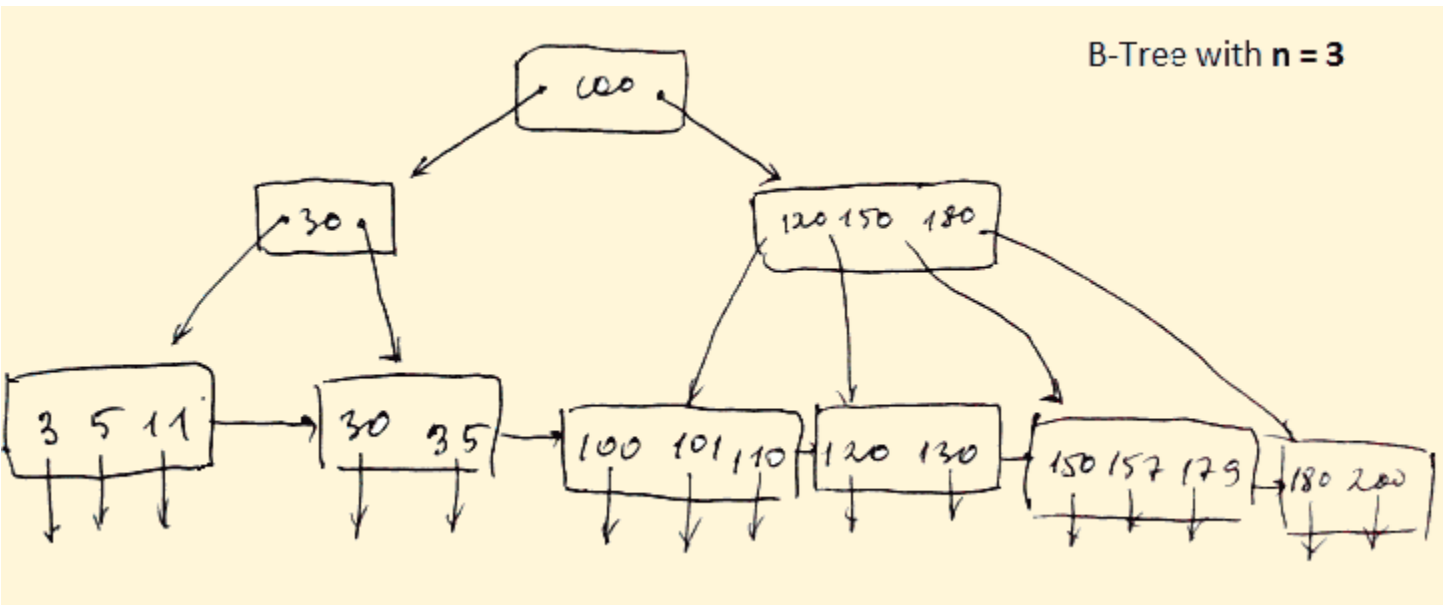
Multilevel Indexing in Database is created when a primary index does not fit in memory. In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.



B-Tree Index

B-tree index is the widely used data structures for tree based indexing in DBMS. It is a multilevel format of tree based indexing in DBMS technique which has balanced [binary search trees](#). All leaf nodes of the B tree signify actual data pointers.

Moreover, all leaf nodes are interlinked with a link list, which allows a B tree to support both random and sequential access.



- Leaf nodes must have between 2 and 4 values.
- Every path from the root to leaf are mostly on an equal length.
- Non-leaf nodes apart from the root node have between 3 and 5 children nodes.
- Every node which is not a root or a leaf has between $n/2$] and n children.

Advantages of Indexing

Important pros/ advantage of Indexing are:

- It helps you to reduce the total number of I/O operations needed to retrieve that data, so you don't need to access a row in the database from an index structure.
- Offers Faster search and retrieval of data to users.
- Indexing also helps you to reduce tablespace as you don't need to link to a row in a table, as there is no need to store the ROWID in the Index. Thus you will able to reduce the tablespace.

- You can't sort data in the leaf nodes as the value of the primary key classifies it.

Disadvantages of Indexing

Important drawbacks/cons of Indexing are:

- To perform the indexing database management system, you need a primary key on the table with a unique value.
- You can't perform any other indexes in Database on the Indexed data.
- You are not allowed to partition an index-organized table.
- SQL Indexing Decrease performance in INSERT, DELETE, and UPDATE query.

Summary:

- Indexing is a small table which consists of two columns.
- Two main types of indexing methods are 1) Primary Indexing 2) Secondary Indexing.
- Primary Index is an ordered file which is fixed length size with two fields.
- The primary Indexing is also further divided into two types 1) Dense Index 2) Sparse Index.
- In a dense index, a record is created for every search key valued in the database.
- A sparse indexing method helps you to resolve the issues of dense Indexing.
- The secondary Index in DBMS is an indexing method whose search key specifies an order different from the sequential order of the file.
- Clustering index is defined as an ordered data file.
- Multilevel Indexing is created when a primary index does not fit in memory.
- The biggest benefit of Indexing is that it helps you to reduce the total number of I/O operations needed to retrieve that data.
- The biggest drawback to performing the indexing database management system, you need a primary key on the table with a unique value.

3. Must Know!

DBMS vs RDBMS: Difference between DBMS and RDBMS

In this RDBMS vs DBMS tutorial, we will learn about main RDBMS and DBMS difference. But before that, let's learn:

What is DBMS?

A DBMS is a software used to store and manage data. The DBMS was introduced during 1960's to store any data. It also offers manipulation of the data like insertion, deletion, and updating of the data.

DBMS system also performs the functions like defining, creating, revising and controlling the database. It is specially designed to create and maintain data and enable the individual business application to extract the desired data.

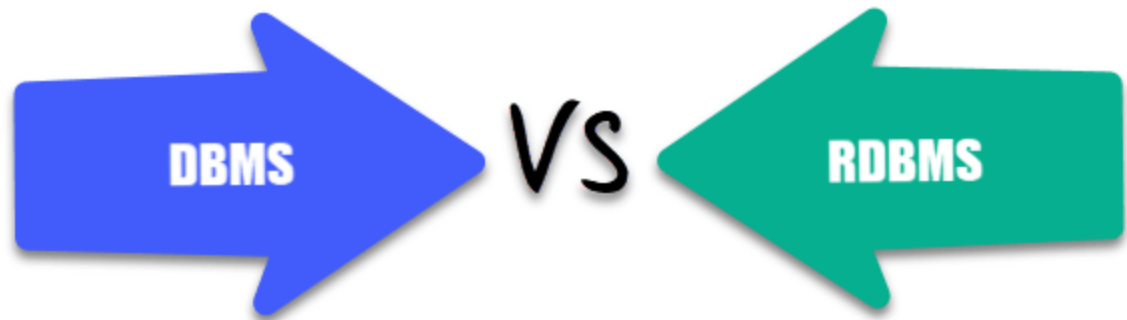
What is RDBMS?

Relational Database Management System (RDBMS) is an advanced version of a DBMS system. It came into existence during 1970's. RDBMS system also allows the organization to access data more efficiently than DBMS.

RDBMS is a software system which is used to store only data which need to be stored in the form of tables. In this kind of system, data is managed and stored in rows and columns which is known as tuples and attributes. RDBMS is a powerful data management system and is widely used across the world.

KEY DIFFERENCE

- DBMS stores data as a file whereas in RDBMS, data is stored in the form of tables.
- DBMS supports single users, while RDBMS supports multiple users.
- DBMS does not support client-server architecture but RDBMS supports client-server architecture.
- DBMS has low software and hardware requirements whereas RDBMS has higher hardware and software requirements.
- In DBMS, data redundancy is common while in RDBMS, keys and indexes do not allow data redundancy.



Difference between DBMS and RDBMS

Difference between DBMS vs RDBMS

The below table demonstrates the main difference between RDBMS and DBMS:

Parameter	DBMS	RDBMS
Storage	DBMS stores data as a file.	Data is stored in the form of tables.
Database structure	DBMS system, stores data in either a navigational or hierarchical form.	RDBMS uses a tabular structure where the headers are the column names, and the rows contain corresponding values
Number of Users	DBMS supports single user only.	It supports multiple users.
ACID	In a regular database, the data may not be stored following the ACID model. This can develop inconsistencies in the database.	Relational databases are harder to construct, but they are consistent and well structured. They obey ACID (Atomicity, Consistency, Isolation, Durability).
Type of program	It is the program for managing the databases on the computer networks and the system hard disks.	It is the database systems which are used for maintaining the relationships among the tables.
Hardware and software needs.	Low software and hardware needs.	Higher hardware and software need.
Integrity constraints	DBMS does not support the integrity constants. The integrity constants are not imposed at the file level.	RDBMS supports the integrity constraints at the schema level. Values beyond a defined range cannot be stored into the particular RDMS column.
Normalization	DBMS does not support Normalization	RDBMS can be Normalized.
Distributed Databases	DBMS does not support distributed database.	RBMS offers support for distributed databases.
Ideally suited for	DBMS system mainly deals with small quantity of data.	RDMS is designed to handle a large amount of data.
Dr. E.F. Codd Rules	Dbms satisfy less than seven of Dr. E.F. Codd Rules	Dbms satisfy 8 to 10 Dr. E.F. Codd Rules
Client Server	DBMS does not support client-server architecture	RDBMS supports client-server architecture.
Data Fetching	Data fetching is slower for the complex and large amount of data.	Data fetching is rapid because of its relational approach.
Data Redundancy	Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
Data Relationship	No relationship between data	Data is stored in the form of tables which are related to each other with the help of foreign keys.
Security	There is no security.	Multiple levels of security. Log files are created at OS, Command, and object level.
Data Access	Data elements need to access individually.	Data can be easily accessed using SQL query. Multiple data elements can be accessed at the same time.
Examples	Examples of DBMS are a file system, XML, Windows Registry, etc.	Example of RDBMS is MySQL, Oracle, SQL Server, etc.

File System vs DBMS: Key Differences

What is a File system?

A file system is a technique of arranging the files in a storage medium like a hard disk, pen drive, DVD, etc. It helps you to organizes the data and allows easy retrieval of files when they are required. It mostly consists of different types of files like mp3, mp4, txt, doc, etc. that are grouped into directories.

A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and Linux.

What is DBMS?

Database Management System (DBMS) is a software for storing and retrieving user’s data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the DBMS engine to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

KEY DIFFERENCES:

- A file system is a software that manages and organizes the files in a storage medium, whereas DBMS is a software application that is used for accessing, creating, and managing databases.
- The file system doesn’t have a crash recovery mechanism on the other hand, DBMS provides a crash recovery mechanism.
- Data inconsistency is higher in the file system. On the contrary Data inconsistency is low in a database management system.
- File system does not provide support for complicated transactions, while in the DBMS system, it is easy to implement complicated transactions using SQL.
- File system does not offer concurrency, whereas DBMS provides a concurrency facility.

Features of a File system

Here are important elements of the file system:

- It helps you to store data in a group of files.
- Files data are dependent on each other.
- C/C++ and COBOL languages were used to design the files.
- Shared File System Support
- Fast File System Recovery.

Features of DBMS

Here, are essential features of DBMS:

- A user-accessible catalog of data
- Transaction support
- Concurrency control with Recovery services
- Authorization services
- The value of data is the same at all places.
- Offers support for data communication
- Independent utility services
- Allows multiple users to share a file at the same time

Difference between filesystem vs. DBMS



Here, are the difference between File System and DBMS

File System	DBMS
A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.	DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases.
The file system provides the details of data representation and storage of data.	DBMS gives an abstract view of data that hides the details
Storing and retrieving of data can’t be done efficiently in a file system.	DBMS is efficient to use as there are a wide variety of methods to store and retrieve data.
It does not offer data recovery processes.	There is a backup recovery for data in DBMS.

The file system doesn't have a crash recovery mechanism.	DBMS provides a crash recovery mechanism
Protecting a file system is very difficult.	DBMS offers good protection mechanism.
In a file management system, the redundancy of data is greater.	The redundancy of data is low in the DBMS system.
Data inconsistency is higher in the file system.	Data inconsistency is low in a database management system.
The file system offers lesser security.	Database Management System offers high security.
File System allows you to stores the data as isolated data files and entities.	Database Management System stores data as well as defined constraints and interrelation.
Not provide support for complicated transactions.	Easy to implement complicated transactions.
The centralization process is hard in File Management System.	Centralization is easy to achieve in the DBMS system.
It doesn't offer backup and recovery of data if it is lost.	DBMS system provides backup and recovery of data even if it is lost.
There is no efficient query processing in the file system.	You can easily query data in a database using the SQL language.
These system doesn't offer concurrency.	DBMS system provides a concurrency facility.

Advantages of File system

Here are pros/benefits of file system:

- Enforcement of development and maintenance standards.
- Helps you to reduce redundancy
- Avoid inconsistency across file maintenance to get the integrity of data independence.
- Firm theoretical foundation (for the relational model).
- It is more efficient and cost less than a DBMS in certain situations.
- The design of file processing is simpler than designing Database.

Advantages of DBMS system

Here, are pros/benefits of DBMS system:

- DBMS offers a variety of techniques to store & retrieve data
- Uniform administration procedures for data
- Application programmers never exposed to details of data representation and Storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- Reduced Application Development Time
- Consume lesser space
- Reduction of redundancy.
- Data independence.

Application of File system

Here, are an important application of the file system:

- Language-specific run-time libraries
- API programs using it to make requests of the file system
- It is used for data transfer and positioning.
- Helps you to update the metadata
- Managing directories.

Application of the DBMS system

Here, are important applications of the DBMS system:

- Admission System Examination System Library System
- Payroll & Personnel Management System
- Accounting System Hotel Reservation System Airline Reservation System
- It is used in the Banking system for Customer information, account activités, Payments, dépositions, loans, etc.
- Use for Airlines for reservations and schedules
- DBMS system also used by universities to keep call records, monthly bills, maintaining balances, etc.

- Finance for storing information about stock, sales, and purchases of financial instruments like stocks and bonds.

Disadvantages of File system

Here, are cons/drawback of the file system:

- Each application has its data file so, the same data may have to be recorded and stored many times.
- Data dependence in the file processing system are data-dependent, but, the problem is incompatible with file format.
- Limited data sharing.
- The problem with security.
- Time-consuming.
- It allows you to maintain the record of the big firm having a large number of items.
- Required lots of labor work to do.

Disadvantages of the DBMS system

Here, are some cons/drawbacks of the DBMS system:

- Cost of Hardware and Software of a DBMS is quite high, which increases the budget of your organization.
- Most database management systems are often complex systems, so the training for users to use the DBMS is required.
- The use of the same program at a time by many users sometimes lead to the loss of some data.
- DBMS can't perform sophisticated calculations
- Data-sets begins to grow large as it provides a more predictable query response time.
- It required a processor with the high speed of data processing.
- The database can fail because of power failure or the whole system stops.
- The cost of DBMS is depended on the environment, function, or recurrent annual maintenance cost.

SQL vs NoSQL: What's the Difference Between SQL and NoSQL

This tutorial of the difference between SQL and NoSQL databases will discuss key SQL and NoSQL differences. But before discussing NoSQL and SQL difference, let us first look at them individually. Let's start with SQL:

What is SQL?

Structured Query language ([SQL](#)) **pronounced as “S-Q-L” or sometimes as “See-Quel”** is the standard language for dealing with Relational Databases. A relational database defines relationships in the form of tables.

SQL programming can be effectively used to insert, search, update, delete database records.

That doesn't mean SQL cannot do things beyond that. It can do a lot of things including, but not limited to, optimizing and maintenance of databases.

Relational databases like MySQL Database, Oracle, Ms SQL Server, Sybase, etc. use SQL.

What is NoSQL?

[NoSQL](#) is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would NoREL NoSQL caught on. Carl Strozzi introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

Next, we will discuss the key diff between SQL and NoSQL.

KEY DIFFERENCE

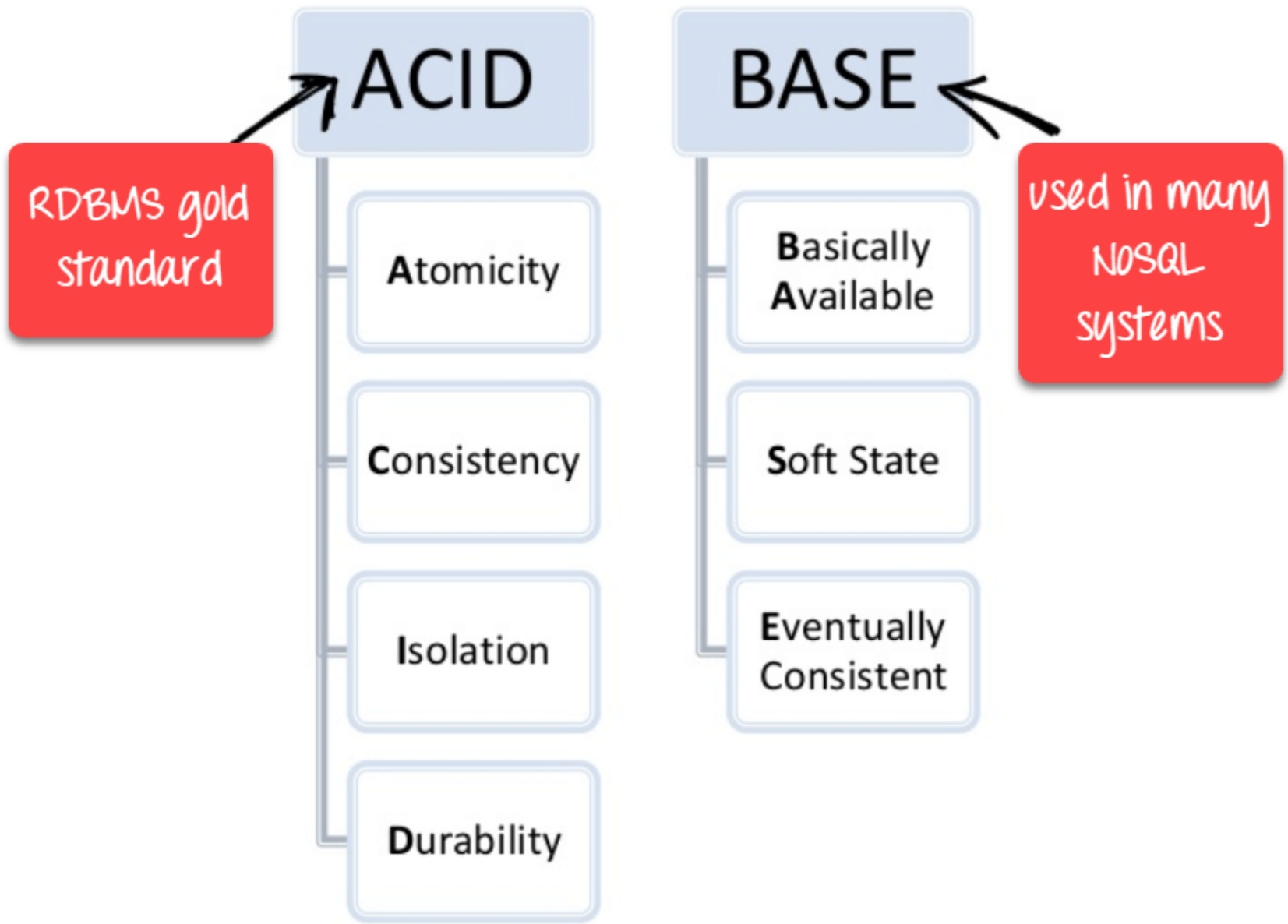
- **SQL** pronounced as “S-Q-L” or as “See-Quel” is primarily called RDBMS or Relational Databases whereas **NoSQL** is a Non-relational or Distributed Database.
- Comparing SQL vs NoSQL database, SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
- SQL databases are vertically scalable while NoSQL databases are horizontally scalable.
- SQL databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.
- Comparing NoSQL vs SQL performance, SQL requires specialized DB hardware for better performance while NoSQL uses commodity hardware.

Difference between SQL and NoSQL

Below is the main difference between NoSQL and SQL:

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB , Redis, Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra .
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it’s more important to have fast data than correct data
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developer is \$84,328 per year in the U.S.A.	The average salary for “NoSQL developer” ranges from approximately \$72,174 per year

Parameter	SQL	NOSQL
ACID vs. BASE Model	ACID (Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems



Difference between ACID vs BASE in DBMS

When use SQL?

The below image shows Stackoverflow questions for SQL vs NoSQL databases:

(MySQL) Stackoverflow Questions NoSQL DB (Mongo) Vs RDBMS DB

- SQL is the easiest language used to communicate with the RDBMS
- Analyzing behavioral related and customized sessions
- Building custom dashboards
- It allows you to store and gets data from the database quickly

- Preferred when you want to use joins and execute complex queries

When use NoSQL?

The below image shows the Google trends for NoSQL vs SQL:



- When ACID support is not needed
- When Traditional RDBMS model is not enough
- Data which need a flexible schema
- Constraints and validations logic not required to be implemented in database
- Logging data from distributed sources
- It should be used to store temporary data like shopping carts, wish list and session data

Clustered vs Non-clustered Index: Key Differences with Example

What is an Index?

An Index is a key built from one or more columns in the database that speeds up fetching rows from the table or view. This key helps a Database like Oracle, SQL Server, MySQL, etc. to find the row associated with key values quickly.

Two types of Indexes are:

- Clustered Index
- Non-Clustered Index

In this tutorial, you will learn:

- [What is an Index?](#)
- [What is a Clustered index?](#)
- [What is Non-clustered index?](#)
- [Characteristic of Clustered Index](#)
- [Characteristics of Non-clustered Indexes](#)
- [An example of a clustered index](#)
- [An example of a non-clustered index](#)
- [Differences between Clustered Index and NonClustered Index](#)
- [Advantages of Clustered Index](#)
- [Advantages of Non-clustered index](#)
- [Disadvantages of Clustered Index](#)
- [Disadvantages of Non-clustered index](#)

What is a Clustered index?

Cluster index is a type of index which sorts the data rows in the table on their key values. In the Database, there is only one clustered index per table.

A clustered index defines the order in which data is stored in the table which can be sorted in only one way. So, there can be an only a single clustered index for every table. In an RDBMS, usually, the primary key allows you to create a clustered index based on that specific column.

What is Non-clustered index?

A Non-clustered index stores the data at one location and indices at another location. The index contains pointers to the location of that data. A single table can have many non-clustered indexes as an index in the non-clustered index is stored in different places.

For example, a book can have more than one index, one at the beginning which displays the contents of a book unit wise while the second index shows the index of terms in alphabetical order.

A non-clustering index is defined in the non-ordering field of the table. This type of indexing method helps you to improve the performance of queries that use keys which are not assigned as a primary key. A non-clustered index allows you to add a unique key for a table.

KEY DIFFERENCE

- Cluster index is a type of index that sorts the data rows in the table on their key values whereas the Non-clustered index stores the data at one location and indices at another location.
- Clustered index stores data pages in the leaf nodes of the index while Non-clustered index method never stores data pages in the leaf nodes of the index.
- Cluster index doesn't require additional disk space whereas the Non-clustered index requires additional disk space.
- Cluster index offers faster data accessing, on the other hand, Non-clustered index is slower.

Characteristic of Clustered Index

- Default and sorted data storage
- Use just one or more than one columns for an index
- Helps you to store Data and index together
- Fragmentation
- Operations
- Clustered index scan and index seek
- Key Lookup

Characteristics of Non-clustered Indexes

- Store key values only
- Pointers to Heap/Clustered Index rows
- Allows Secondary data access
- Bridge to the data
- Operations of Index Scan and Index Seek
- You can create a nonclustered index for a table or view
- Every index row in the nonclustered index stores the nonclustered key value and a row locator

An example of a clustered index

In the example below, SalesOrderDetailID is the clustered index. Sample query to retrieve data

```
SELECT CarrierTrackingNumber, UnitPrice
FROM SalesData
WHERE SalesOrderDetailID = 6
```


SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice
43659	1	4911-403C-98	1	776	1	2024.994
43659	2	4911-403C-98	3	777	1	2024.994
43659	3	4911-403C-98	1	778	1	2024.994
43659	4	4911-403C-98	1	771	1	2039.994
43659	5	4911-403C-98	1	772	1	2039.994
43659	6	4911-403C-98	2	773	1	2039.994
43659	7	4911-403C-98	1	774	1	2039.994
43659	8	4911-403C-98	3	714	1	28.8404
43659	9	4911-403C-98	1	716	1	28.8404
43659	10	4911-403C-98	6	709	1	5.70
43659	11	4911-403C-98	2	712	1	5.1865
43659	12	4911-403C-98	4	711	1	20.1865
43660	13	6431-4D57-83	1	762	1	419.4589
43660	14	6431-4D57-83	1	758	1	874.794
43661	15	4E0A-4F89-AE	1	745	1	809.76



An example of a non-clustered index

In the below example, a non-clusted index is created on OrderQty and ProductID as follows

```
CREATE INDEX myIndex ON
SalesData (ProductID, OrderQty)
```

SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice
43659	1	4911-403C-98	1	776	1	2024.994
43659	2	4911-403C-98	3	777	1	2024.994
43659	3	4911-403C-98	1	778	1	2024.994
43659	4	4911-403C-98	1	771	1	2039.994
43659	5	4911-403C-98	1	772	1	2039.994
43659	6	4911-403C-98	2	773	1	2039.994
43659	7	4911-403C-98	1	774	1	2039.994
43659	8	4911-403C-98	3	714	1	28.8404
43659	9	4911-403C-98	1	716	1	28.8404
43659	10	4911-403C-98	6	709	1	5.70
43659	11	4911-403C-98	2	712	1	5.1865
43659	12	4911-403C-98	4	711	1	20.1865
43660	13	6431-4D57-83	1	762	1	419.4589
43660	14	6431-4D57-83	1	758	1	874.794
43661	15	4E0A-4F89-AE	1	745	1	809.76



The following query will be retrieved faster compared to the clustered index.

```
SELECT Product ID, OrderQty
FROM SalesData
WHERE ProductID = 714
```

43659	7	4911-403C-98	1	774	1	2039.994
43659	8	4911-403C-98	3	714	1	28.8404
43659	9	4911-403C-98	1	716	1	28.8404

Differences between Clustered Index and NonClustered Index



Parameters	Clustered	Non-clustered
Use for	You can sort the records and store clustered index physically in memory as per the order.	A non-clustered index helps you to creates a logical order for data rows and uses pointers for physical data files.
Storing method	Allows you to stores data pages in the leaf nodes of the index.	This indexing method never stores data pages in the leaf nodes of the index.

Parameters	Clustered	Non-clustered
Size	The size of the clustered index is quite large.	The size of the non-clustered index is small compared to the clustered index.
Data accessing	Faster	Slower compared to the clustered index
Additional disk space	Not Required	Required to store the index separately
Type of key	By Default Primary Keys Of The Table is a Clustered Index.	It can be used with unique constraint on the table which acts as a composite key.
Main feature	A clustered index can improve the performance of data retrieval.	It should be created on columns which are used in joins.

Advantages of Clustered Index

The pros/benefits of the clustered index are:

- Clustered indexes are an ideal option for range or group by with max, min, count type queries
- In this type of index, a search can go straight to a specific point in data so that you can keep reading sequentially from there.
- Clustered index method uses location mechanism to locate index entry at the start of a range.
- It is an effective method for range searches when a range of search key values is requested.
- Helps you to minimize page transfers and maximize the cache hits.

Advantages of Non-clustered index

Pros of using non-clustered index are:

- A non-clustering index helps you to retrieves data quickly from the database table.
- Helps you to avoid the overhead cost associated with the clustered index
- A table may have multiple non-clustered indexes in RDBMS. So, it can be used to create more than one index.

Disadvantages of Clustered Index

Here, are cons/drawbacks of using clustered index:

- Lots of inserts in non-sequential order
- A clustered index creates lots of constant page splits, which includes data page as well as index pages.
- Extra work for SQL for inserts, updates, and deletes.
- A clustered index takes longer time to update records when the fields in the clustered index are changed.
- The leaf nodes mostly contain data pages in the clustered index.

Disadvantages of Non-clustered index

Here, are cons/drawbacks of using non-clustered index:

- A non-clustered index helps you to stores data in a logical order but does not allow to sort data rows physically.
- Lookup process on non-clustered index becomes costly.
- Every time the clustering key is updated, a corresponding update is required on the non-clustered index as it stores the clustering key.

Primary Key vs Foreign Key: What’s the Difference?

Before learning the difference between primary key and foreign key, let’s learn:

What are Keys?

Keys are attribute that helps you to identify a row(tuple) in a relation(table). They allow you to find the relationship between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. The database key is also helpful for finding a unique record or row from the table.

What is Database Relationship?

The database relationship is associations between one or more tables that are created using join statements. It is used to efficiently retrieve data from the database. There are primarily three types of relationships 1) One-to-One, 2) One-to-many, 3) Many-to-many.

What is Primary Key?

A primary key constrain is a column or group of columns that uniquely identifies every row in the table of the relational database management system. It cannot be a duplicate, meaning the same value should not appear more than once in the table.

A table can have more than one primary key. Primary key can be defined at the column or the table level. If you create a composite primary key, it should be defined at the table level.

What is Foreign Key?

Foreign key is a column that creates a relationship between two tables. The purpose of the Foreign key is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table. Every relationship in the database should be supported by a foreign key.

KEY DIFFERENCES:

- A primary key constrain is a column that uniquely identifies every row in the table of the relational database management system, while foreign key is a column that creates a relationship between two tables.
- Primary Key never accepts null values whereas foreign key may accept multiple null values.
- You can have only a single primary key in a table while you can have multiple foreign keys in a table.
- The value of the primary key can't be removed from the parent table whereas the value of foreign key value can be removed from the child table.
- No two rows can have any identical values for a primary key on the other hand a foreign key can contain duplicate values.
- There is no limitation in inserting the values into the table column while inserting any value in the foreign key table, ensure that the value is present into a column of a primary key.

Why use Primary Key?

Here are the cons/benefits of using primary key:

- The main aim of the primary key is to identify each and every record in the database table.
- You can use a primary key when you do not allow someone to enter null values.
- If you delete or update records, the action you specified will be undertaken to make sure data integrity.
- Perform restrict operation to rejects delete or update operation for the parent table.
- Data are organized in a sequence of clustered index whenever you physically organize DBMS table.

Why use Foreign Key?

Here are the important reasons of using foreign key:

- Foreign keys help you to migrate entities using a primary key from the parent table.
- A foreign key enables you to link two or more tables together.
- It makes your database data consistent.
- A foreign key can be used to match a column or combination of columns with primary key in a parent table.
- SQL foreign key constraint is used to make sure the referential integrity of the data parent to match values in the child table.

Example of Primary Key

Syntax:

Below is the syntax of Primary Key:

```
CREATE TABLE <Table-Name>
(
Column1 datatype,
```

```
Column2 datatype, PRIMARY KEY (Column-Name)
);
```

Here,

- Table_Name is the name of the table you have to create.
- Column_Name is the name of the column having the primary key.

Example:

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

In the above example, we have created a student table with columns like StudID, Roll No, First Name, Last Name, and Email. StudID is chosen as a primary key because it can uniquely identify other rows in the table.

Example of Foreign Key

Syntax:

Below is the syntax of Foreign Key:

```
CREATE TABLE <Table Name>(
column1      datatype,
column2      datatype,
constraint (name of constraint)
FOREIGN KEY [column1, column2...]
REFERENCES [primary key table name] (List of primary key table column) ...);
```

Here,

- The parameter Table Name indicates the name of the table that you are going to create.
- The parameters column1, column2... depicts the columns that need to be added to the table.
- Constraint denotes the name of constraint you are creating.
- References indicate a table with the primary key.

Example:

DeptCode		DeptName	
001		Science	
002		English	
005		Computer	
Teacher ID		Fname	Lname
B002		David	Warner
B017		Sara	Joseph
B009		Mike	Brunton

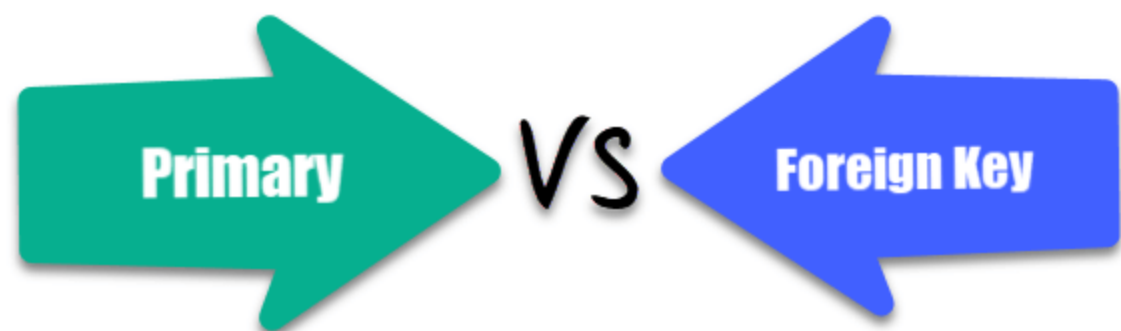
In the above example, we have two tables, a teacher and a department in a school. However, there is no way to see which search works in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

Difference between Primary key and Foreign key



Here is the important difference between Primary key and Foreign key:

Primary Key	Foreign Key
A primary key constrain is a column or group of columns that uniquely identifies every row in the table of the relational database management system.	Foreign key is a column that creates a relationship between two tables.
It helps you to uniquely identify a record in the table.	It is a field in the table that is a primary key of another table.
Primary Key never accepts null values.	A foreign key may accept multiple null values.
The primary key is a clustered index, and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered, or non-clustered.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.
The value of the primary key can't be removed from the parent table.	The value of foreign key value can be removed from the child table.
You can define the primary key implicitly on the temporary tables.	You cannot define foreign keys on the local or global temporary tables.
Primary key is a clustered index.	By default, it is not a clustered index.
No two rows can have any identical values for a primary key.	A foreign key can contain duplicate values.
There is no limitation in inserting the values into the table column.	While inserting any value in the foreign key table, ensure that the value is present into a column of a primary key.

Primary Key vs Unique Key: What’s the Difference?

What is Primary Key?

A primary key constrain is a column or group of columns in a table that uniquely identifies every row in that table. The Primary key can't be a duplicate, meaning the same value can't appear more than once in the table.

A table can have only one primary key. Primary key can be defined at the column or the table level. If you create a composite primary key, it should be defined at the table level.

In this tutorial, you will learn:

- [What is Primary Key?](#)
- [What is Unique Key?](#)
- [Why use Primary Key?](#)
- [Why use Unique Key?](#)
- [Features of Primary Key](#)
- [Features of Unique key](#)
- [Example of Creating Primary Key](#)
- [Example of Creating Unique Key](#)
- [Difference between Primary key and Unique key](#)
- [What is better?](#)

What is Unique Key?

A unique key is a group of one or more than one fields or columns of a table which uniquely identify database record.

A unique key is the same as a primary key, but it can accept one null value for a table column. It also cannot contain identical values. Unique constraints are referenced by the foreign key of other tables.

KEY DIFFERENCES

- There can be one primary key in a table while there can be multiple unique keys in the table.
- The purpose of the primary key is to enforce entity integrity on the other hand the purpose of unique key is to enforce unique data.
- In primary key, default Index is clustered whereas in unique key, default index is not-clustered
- Primary key does not allow null columns whereas unique allows null columns.
- In the primary key, duplicate keys are not allowed while in a unique key, if one or more key parts are null, then duplicate keys are allowed.

Why use Primary Key?



Primary Key

Here are the important reasons to use primary key:

- The main aim of the primary key is to identify each and every record in the database table.
- You can use a primary key when you do not allow someone to enter null values.
- If you delete or update a record, the action you specified will be undertaken to make sure database data integrity.
- Perform restrict operation to rejects delete or update operation for the parent table.
- Data are organized in sequence of clustered index whenever you physically organize DBMS table.

Why use Unique Key?

Here are the important reasons to use unique key:

- The purpose of a unique key is to make sure that information in the column for each table record is unique.
- When you allow the user to enter the null value.
- Unique key is used because it creates a non-clustered index by default.
- Unique key can be used when you have to keep null values in column.
- When one or more than one field/columns of a table that uniquely identify a record in a database table.

Features of Primary Key

Here, are the important features of primary key:

- The primary key implements the entity integrity of the table.
- You can keep only one primary in the table.
- The primary key contains of one or more table columns.
- Columns are defined as not null.

Features of Unique key

Here, are the important features of unique key:

- You can define more than one unique key in the table.
- By default, unique Keys are in non-clustered unique indexes.
- It constitutes of one or more table columns.
- The table column can be null, but only one null per column is preferable.
- A unique constraint can be easily referenced by a foreign key constraint.

Example of Creating Primary Key

The following example describes that there is a table called student. It contains five attributes, 1) StudID, 2) Roll No, 3) First Name, 4) Last Name, and 5) Email.

The Roll No attribute can never contain a duplicate or null value. It is because every student enrolled in a university can have unique roll number. You can easily identify each row of a table with student’s roll number. So it is considered as a primary key.

Table: Student



Primary Key example

Example of Creating Unique Key

Consider the same student table with attributes, 1) StudID, 2) Roll No, 3) First Name, 4) Last Name, and 5) Email.

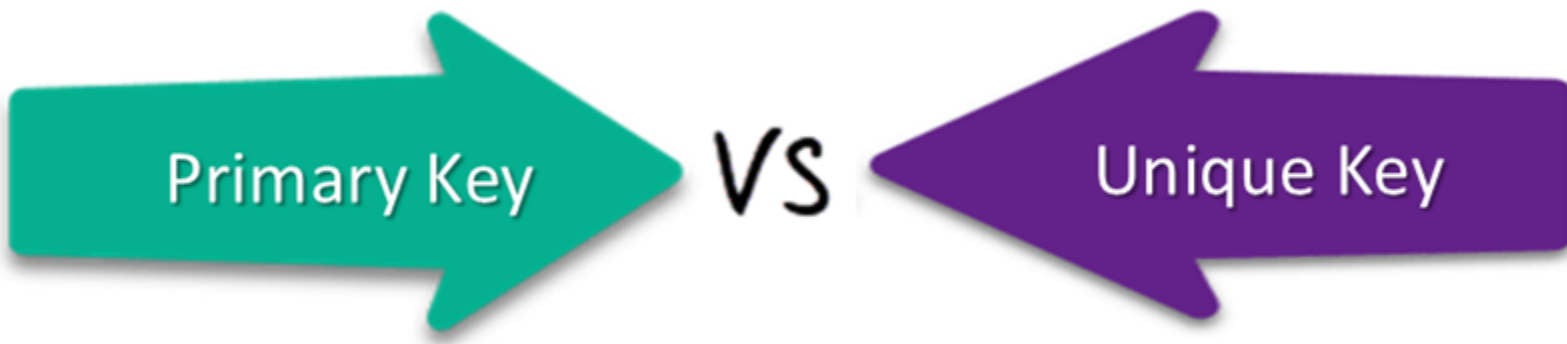
Stud ID can have a unique constraint where entries in Stud ID column can be unique because each student of a university must have a unique ID number. In case, if student is changing the university, in that case, he or she would not have any stud ID. The entry may have a null value as only one null is allowed in the unique key constraint.

Table: Student



Unique key example

Difference between Primary key and Unique key



Here are the important differences between primary key and unique key:

Primary Key	Unique Key
There can be one primary key in a table	There can be multiple unique keys in the table
It does not allow null columns.	It allows null columns.
Default Index is clustered	Default Index is no-clustered
The purpose of the primary key is to enforce entity integrity.	The purpose of unique key is to enforce unique data.
Primary key can be created using syntax: <pre>CREATE TABLE Employee (ID int PRIMARY KEY, Name varchar(255), City varchar(150))</pre>	Unique key can be created using syntax: <pre>CREATE TABLE Employee (ID int UNIQUE. Name varchar(255) NOT NULL. City varchar(150))</pre>
It is SQL constraint which allows you to uniquely identify each record or row in the database table.	It is SQL constraint that doesnot allow the same value tobe assigned to two isolatedRecords in a database table.

In the primary key, duplicate keys are not allowed.

In a unique key, if one or more key parts are null, then duplicate keys are allowed.

What is better?

- Unique key is better when you have columns you know shouldn’t contain duplication. This becomes a good way to make sure data validation.
- Primary key is ideal when you cannot keep null in the table. It can also use when you have a foreign key in another table for creating a relationship.

Row vs Column: What’s the Difference?

What is Row?

A row is a series of data placed out horizontally in a table or spreadsheet. It is a horizontal arrangement of the objects, words, numbers, and data. In Row, data objects are arranged face-to-face with lying next to each other on the straight line.

What is Column?

A column is a vertical series of cells in a spreadsheet, chart, table, or spreadsheet. It is an arrangement of figures, facts, words, etc.

Columns are mostly placed one after another in the continuous sequence. In a table, columns are mostly separated from each other by lines, which help to enhance readability and attractiveness.

			Rows			
Columns						

Row Column Example

KEY DIFFERENCES

- A row is a series of data put out **horizontally** in a table or spreadsheet while a column is a **vertical** series of cells in a chart, table, or spreadsheet.
- Rows go across left to right. On the other hand, Columns are arranged from up to down.
- In a spreadsheet such as MS Excel WPS, LibreOffice, or Google sheet, the row heading is indicated by numbers whereas column headings are denoted by letters.
- You can define the row as an order in which objects are placed alongside or horizontally while you can define a column as a vertical division of objects based on category.
- In the database, the information like name, gender, names, are placed in the rows while in the column contains information about someone who you are mentioning in the rows.

Row Examples:

Here are real-life examples of Row:

- The children are standing one after another.
- Group of people standing in a row at the back of the room.
- Building a row of houses alongside the river.
- Students are standing on the front row of the stalls.

Column Examples:

Here are real-life examples of the column:

- Weekly recipe Item

- Pillar in the front of a building
- A facade with marble columns

When to Use Row-Oriented Storage

Here are some common uses of row storages:

- Row-oriented storage is good if you need to touch one Row.
- This type of storage is also beneficial when most columns of a row need to be read or written.
- Reads are done page-wise so not many rows may fit on a page when rows are big
- Pages are normally not filled, which leads to reading lots of unused and unwanted areas.
- Row storage should be used when recorded headers need to be read too but do not contain actual row data

When to use Column-oriented storage

Here are some common uses of column storage:

- Column helps you to make a side-by-side comparison between two columns.
- Column-oriented databases primarily work on columns
- It is benefited if you want to store the value of a single column
- It helps array-processing the value of a column
- It helps you to improve the performance of queries that only touch a small amount of columns.

Difference between Row and Columns



Here are the main differences between Row and Columns

Row	Column
A row is a series of data banks put out horizontally in a table or spreadsheet.	A column is a vertical series of cells in a chart, table, or spreadsheet.
Rows go across left to right	Columns are arranged from up to down.
In a spreadsheet such as MS Excel WPS, LibreOffice, or Google sheet, the row heading is indicated by numbers.	In spreadsheet programs like excel, LiberOfifce column headings are denoted by letters.
Total number of Row is placed in the extreme right corner of the respective Row.	The total numbers of the column are shown at the bottom.
You can define the Row as an order in which objects are placed alongside or horizontally.	You can define a column as a vertical division of objects based on category.
A stub that is the farthest left part of the table describes the Row.	A caption that is the topmost of the table describes the column.
In DBMS, rows are known as records that contain fields.	In DBMS, columns are called fields which contain the collection of characters.
In a matrix, the horizontal arrays are also called rows.	In a matrix, the vertical arrays are also called columns.
In the database, the information like name, gender, names, are placed in the rows.	The column contains information about someone who you are mentioning in the rows.

Difference Between DDL and DML Command in DBMS: What is?

What is DDL?

Data Definition Language helps you to define the database structure or schema. DDL commands help you to create the structure of the database and the other database objects. Its commands are auto-committed so, the changes are saved in the database permanently. The full form of DDL is Data Definition Language.

In this difference between DML and DDL tutorial, you will learn:

- [What is DDL?](#)
- [What is DML?](#)
- [Why DDL?](#)
- [Why DML?](#)
- [Difference Between DDL and DML in DBMS](#)
- [Commands for DDL](#)
- [Commands for DML](#)
- [DDL Command Example](#)
- [DML Command Example](#)

What is DML?

DML commands it to allow you to manage the data stored in the database, although DML commands are not auto-committed. Moreover, they are not permanent. So, It is possible to roll back the operation. The full form of DML is Data Manipulation Language.

Below is the key difference between DDL and DML in DBMS:

KEY DIFFERENCES:

- Data Definition Language (DDL) helps you to define the database structure or schema while Data Manipulation language (DML command) allows you to manage the data stored in the database.
- DDL command is used to create the database schema while DML command is used to populate and manipulate database
- Comparing DDL vs DML, DDL statements affect the whole table whereas DML commands only affect one or more rows.
- In DDL, SQL Statement can't be rollbacked while in DML SQL Statement can be a rollbacked.
- DDL is a declarative method while DML is an imperative method.
- Important DDL commands are: 1) CREATE, 2) ALTER, 3) DROP, 4) TRUNCATE, etc. while important DML commands are: 1) INSERT, 2) UPDATE, 3) DELETE, 4) MERGE, etc.

Why DDL?

Here, are reasons for using DDL method:

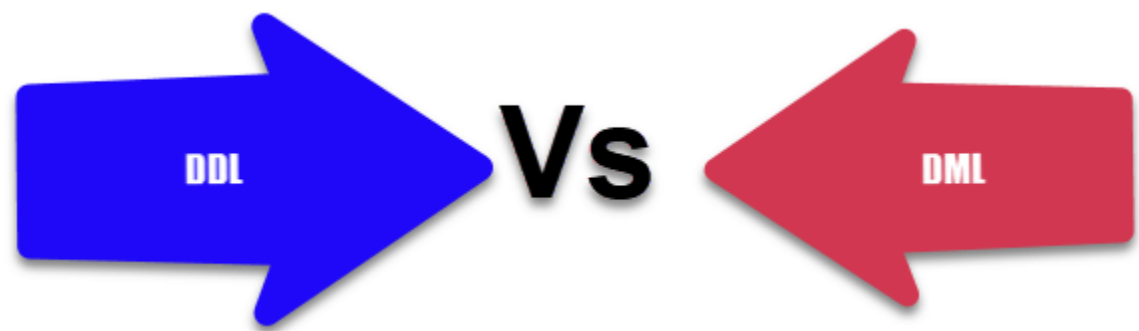
- Allows you to store shared data
- Data independence improved integrity
- Allows multiple users
- Improved security efficient data access

Why DML?

Here, benefits/ pros of DML:

- The DML statements allow you to modify the data stored in a database.
- Users can specify what data is needed.
- DML offers many different flavors and capabilities between database vendors.
- It offers an efficient human interaction with the system.

Difference Between DDL and DML in DBMS



DDL vs DML

Here is the main difference between DDL and DML commands:

DDL	DML
-----	-----

Data Definition Language (DDL) helps you to define the database structure or schema.	Data Manipulation Language (DML command) allows you to manage the data stored in the database.
DDL command is used to create the database schema.	DML command is used to populate and manipulate database
DDL is not classified further.	DML is classified as Procedural and Non and Procedural DMLs.
CREATE, ALTER, DROP, TRUNCATE AND COMMENT and RENAME, etc.	INSERT, UPDATE, DELETE, MERGE, CALL, etc.
It defines the column of the table.	It adds or updates the row of the table
DDL statements affect the whole table.	DML effects one or more rows.
SQL Statement can't be rollback	SQL Statement can be a rollback
DDL is declarative.	DML is imperative.

Commands for DDL

Five types of DDL commands are:

CREATE

CREATE statements is used to define the database structure schema:

Syntax:

```
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
```

For example:

```
Create database university;
Create table students;
Create view for_students;
```

DROP

Drops commands remove tables and databases from RDBMS.

Syntax:

```
DROP TABLE ;
```

For example:

```
Drop object_type object_name;
Drop database university;
Drop table student;
```

ALTER

Alters command allows you to alter the structure of the database.

Syntax:

To add a new column in the table

```
ALTER TABLE table_name ADD column_name COLUMN-definition;
```

To modify an existing column in the table:

```
ALTER TABLE MODIFY(COLUMN DEFINITION....);
```

For example:

```
Alter table guru99 add subject varchar;
```

TRUNCATE:

This command used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE table students;
```

Commands for DML

Here are some important DML commands:

- INSERT
- UPDATE

- DELETE

INSERT:

This is a statement that is a SQL query. This command is used to insert data into the row of a table.

Syntax:

```
INSERT INTO TABLE_NAME (col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
Or
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, .... valueN);
```

For example:

```
INSERT INTO students (RollNo, FirstName, LastName) VALUES ('60', 'Tom', 'Erichsen');
```

UPDATE:

This command is used to update or modify the value of a column in the table.

Syntax:

```
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
```

For example:

```
UPDATE students
SET FirstName = 'Jhon', LastName=' Wick'
WHERE StudID = 3;
```

DELETE:

This command is used to remove one or more rows from a table.

Syntax:

```
DELETE FROM table_name [WHERE condition];
```

For example:

```
DELETE FROM students
WHERE FirstName = 'Jhon';
```

DDL Command Example

CREATE

Syntax:

```
CREATE TABLE tableName
(
    column_1 datatype [ NULL | NOT NULL ],
    column_2 datatype [ NULL | NOT NULL ],
    ...
);
```

Here,

- The parameter tableName denotes the name of the table that you are going to create.
- The parameters column_1, column_2... denote the columns to be added to the table.
- A column should be specified as either NULL or NOT NULL. If you don't specify, SQL Server will take NULL as the default

Example:

```
CREATE TABLE Students
(
    Student_ID Int,
    Student_Name Varchar(10)
)
```

ALTER

Syntax:

```
Alter TABLE <Table name> ADD Column1 datatype, Column2 datatype;
```

Example:

```
ALTER TABLE University.Students_Name ADD Course_Duration VARCHAR(20);
```

DROP

Syntax:

```
DROP TABLE <tableName>;
```

The parameter tableName is the name of the table that is to be deleted.

Example:

```
DROP TABLE COURSE_NAMES;
```

DML Command Example

INSERT

In [PL/SQL](#), we can insert the data into any table using the [SQL](#) command INSERT INTO. This command will take the table name, table column, and column values as the input and insert the value in the base table.

The INSERT command can also take the values directly from another table using ‘SELECT’ statement rather than giving the values for each column. Through ‘SELECT’ statement, we can insert as many rows as the base table contains.

Syntax:

```
BEGIN
  INSERT INTO <table_name>(<column1 >,<column2>,...<column_n>)
    VALUES(<value1><value2>,...:<value_n>);
END;
```

The above syntax shows the INSERT INTO command. The table name and values are mandatory fields, whereas column names are not mandatory if the insert statements have values for all the columns of the table.

The keyword ‘VALUES’ is mandatory if the values are given separately, as shown above.

Syntax:

```
BEGIN
  INSERT INTO <table_name>(<column1>,<column2>,...,<column_n>)
    SELECT <column1>,<column2>,.. <column_n> FROM <table_name2>;
END;
```

The above syntax shows the INSERT INTO command that takes the values directly from the <table_name2> using the SELECT command.

The keyword ‘VALUES’ should not be present in this case, as the values are not given separately.

DELETE

Below is the Syntax to delete table

Syntax:

```
DROP TABLE <TableName>;
```

The parameter TableName is the name of the table that is to be deleted.

Example:

```
DROP TABLE COURSE_NAMES;
```

SELECT

To view data in SQL Server, we use the SELECT statement.

Syntax:

```
SELECT expression
FROM tableName
[WHERE condition];
```

Example:

```
SELECT * FROM Course;
```

13 BEST Free Database Software (SQL Databases List) in 2022

A Database is a systematic collection of data which supports storage and manipulation of information. It is usually managed by a Database Management System (DBMS). Data within a database is typically modeled in rows and columns in tables to make data querying and processing more efficient.

Following is the best Database Software list, with popular features and download links. This comparison database name list contains open-source tools that may have freemium features of the Top Free database.

1) InterBase

InterBase is a full-featured, high-performance, scalable, lightweight, embeddable and encryptable relational database that can be embedded into applications on Android, iOS, Windows, OS X, Linux, and Solaris.



Platforms: Windows, iOS, macOS, Android, Linux

Languages: Java, C, C++, .NET, Delphi, Object Pascal, PHP and Ruby.

Features:

- Security: Cross-Platform Encryption, Separate Security Login, Reduced Exposure to Risk, Role-Based User Security, Encrypted Backups
- Admin-Free: Near-Zero Maintenance, Rapid Crash Recovery, Disaster Recovery
- Resilience: Live Backups, Distinguished Data Dumps, Fast Restores, Write-Ahead Logging, Point-in-Time Recovery
- Efficiency: Log-Less Replication, Safe Data Change Subscriptions, Best-in-Class Change Tracking, Role-Based User Security, Early Fetch Data Deltas
- Flexibility: Multi-Language Support, Cross-Platform Support, Server Data Storage, Client Data Storage, Single On-Disk Format
- Lightweight: Small Footprint, Fast Install, Faster Data, Write-Ahead Logging

[More Information >>](#)

2) Microsoft SQL



SQL Server is RDBMS developed by Microsoft. SQL Server supports ANSI SQL, which is the standard SQL (Structured Query Language) language. However, SQL Server comes with its implementation of the SQL language, T-SQL (Transact-SQL).

Platform: Docker Engine, Ubuntu, SUSE Linux Enterprise Server, and Red Hat Enterprise Linux.

Languages: C, C++, Java, and C#

Cloud Version: Yes

Features:

- It provides integration of structured and unstructured data with the power of SQL Server and Spark.
- The tool offers scalability, performance, and availability for mission-critical, intelligent applications, data warehouses, and data lakes.
- It offers advanced security features to protect your data.
- Access to rich, interactive Power BI reports, to make a faster and better decision.

[More Information >>](#)

3) MySQL



MySQL is an open-source relational database which runs on a number of different platforms such as Windows, Linux, and Mac OS, etc.

Platform: Linux, Windows, and Mac.

Languages: C, Java, SQL, C++, Perl, Python, Tcl, and PHP.

Cloud Version: Yes

Features:

- This open source database tool provides Scalability and Flexibility
- This free SQL database tool has web and [data warehouse](#) strengths
- It provides high Performance
- This free database software for Windows 10 has Robust Transactional Support

Verdict: MySQL can be used for packaged software, and business-critical systems and high-volume websites.

Link: <https://www.mysql.com/>

4) PostgreSQL



PostgreSQL is an enterprise-class open source database management system. It is one of the best free databases that supports both SQL for relational and JSON for non-relational queries. It is backed by an experienced community of developers who have made a tremendous contribution to make it highly reliable Database management software.

Platform: Mac, Windows, and Linux.

Languages: PL/Tcl, PL/pgSQL, PL/Python, and PL/Perl.

Cloud Version: No.

Features:

- Compatible with various platforms using all major languages and middleware
- Standby server and high availability
- The tool has mature server-side programming Functionality
- Log-based and trigger-based replication SSL.
- It offers a most sophisticated locking mechanism.
- Support for multi-version concurrency control
- It provides support for client-server network architecture
- The tool is Object-oriented and ANSI-SQL2008 compatible
- PostgreSQL allows linking with other data stores like NoSQL, which act as a federated hub for polyglot databases.

Verdict: PostgreSQL enables you to create custom data types and range of query methods. You can run a store procedure in different programming languages.

Link: <https://www.postgresql.org/>

5) MongoDB



MongoDB is a document-oriented NoSQL database used for high volume data storage. It is an open source database which came into light around the mid-2000s. It is one of the best free database that falls under the category of a NoSQL database.

Platform: Cross-platform

Languages: C#, C, Java, C++, Perl, Scala, Ruby, etc.

Cloud Version: Yes

Features:

- Fully Automated scale provision, and manage complex, highly available, multi-node clusters with easily with API calls.
- This tool allows you to create globally distributed clusters.
- Easy to restore data when needed.
- It offers visualization, monitor, and alert on more than 80 metrics which track your cluster's health or integrate with third-party monitoring solutions.
- The tool has a powerful query language
- It uses rich JSON documents to store tables in a relational database.
- It provides MongoDB Atlas, which is a global cloud database.

Verdict: MongoDB allows you to validate the document. It is not suitable for those applications having complex transactions.

Link: <https://www.mongodb.com/>

6) OrientDB



OrientDB is an open-source NoSQL multi-model database which helps organizations to unlock the power of graph databases without deploying multiple systems to handle other data types. This helps you to increase performance and security while supporting scalability.

Platform: Linux, Mac OS X, Windows, Solaris, and HP-UX

Languages: Java, PHP, Nodejs, .NET, Python, C, JavaScript, Ruby, Scala, Elixir, Android, Perl.

Cloud Version: Yes

Features:

- Unified Multi-Model API – for quicker deployment
- TinkerPop 3 for the state of fast and effective upgrades.
- The tool focused on Scalability and performance.
- This free cloud database offers enhanced query planner.

Verdict: OrientDB has the ability to do multi-master replication, shared data using clusters, and automate distributed queries and transactions.

Link: <https://www.orientdb.org/>

7) MariaDB



MariaDB is a fork of the MySQL database management system. It is created by its original developers. This DBMS tool provides data processing capabilities for both small and enterprise tasks.

Platform: Windows, Linux, and Mac.

Languages: C#, Java, C++, Python, etc.

Cloud Version: Yes

Features:

- It operates under GPL, BSD or LGPL licenses.
- It comes with many storage engines, including the high-performance ones that can be integrated with other relational database management systems.
- It is one of the best open source database softwares which provides the Galera cluster technology.
- MariaDB can run on different operating systems, and it supports numerous [programming languages](#).

Verdict: MariaDB is an alternate software to MySQL. It provides high scalability through easy integration.

Link: <https://mariadb.org/>

8) SQLite



SQLite is an open-source, embedded, relational database management system, designed circa 2000. It is a database, with zero configuration, no requirements of a server or installation. Despite its simplicity, it is laden with popular features of database management systems.

Platform: Blackberry, Symbian, Maemo, Android, MeeGo, WebOS, NetBSD, FreeBSD, illumos, Solaris 10, Windows, and Tizen.

Languages: C, C#, C++, Java, PHP, Python, Ruby, etc.

Cloud Version: Yes

Features:

- SQLite is very lightweight compared to other database management systems like SQL Server, or Oracle.
- It has an in-memory library that you can call and use directly without installation or configuration.
- You do not require any dedicated server to store database. The database is stored in the hard disk of a computer.

Verdict: SQLite is a C programming language library which

offers self-contained, reliable and full-featured SQL database engine.

Link: <https://www.sqlite.org/index.html>

9) Cassandra



Cassandra is a free tool which is designed to manage a large amount of data across a wide range of servers. The tool offers support for replicating across multiple datacenters.

Platform: Cross-platform

Languages: SQL, Go, C++, Python, and Node JS

Cloud Version: Yes

Features:

- Data is copied to numerous nodes to provide a fault-tolerance system.
- There are no network bottlenecks as every node in the cluster is separate.
- The tool supports for contracts and services from third parties.
- It allows you to choose between synchronous or asynchronous replication for the update.

Verdict: Cassandra is for those people who want scalability and high availability without decreasing performance.

Link: <http://cassandra.apache.org/>

10) CouchDB



CouchDB is open-source software that is based on the common standards to access your important data. It stores data on your server or with a leading service provider of your choice.

Platform: Cross-Platform

Languages: Java, Python, C++, Perl, C, JavaScript, PHP, etc..

Cloud Version: No

Features:

- It enables you to run a logical database server on any virtual machines.
- CouchDB tool works with external tools like load balancers, HTTP, and proxy servers.
- The tool provides support of authentication and session.
- CouchDB cluster enables you to save data redundantly.

Verdict: CouchDB offers a scalable solution. It also provides flexibility for storing data.

Link: <https://couchdb.apache.org/>

11) Oracle



Oracle is self-repairing, self-securing, and self-driving designed to eliminate manual data management. It is an intelligent, secure, and highly available database in the cloud that helps you to grow your business.

Platform: Windows and Linux

Languages: C++, COBOL, C, Java, Visual Basic, and PL/SQL.

Cloud Version: Yes

Features:

- Oracle Cloud is optimized for high-performance database workloads, streaming workloads, and Hyperscale big data.
- You can easily migrate to the Cloud.
- This free SQL software provides the services based on how you like to operate, in order to run Oracle cloud in your data center.

Verdict: Oracle database stores and retrieve information. Database server solves the problem related to information management.

Link: <https://www.oracle.com/in/database/>

12) DynamoDB



Amazon DynamoDB is a nonrelational database. This database system provides consistent latency and offers built-in security and in-memory caching. DynamoDB is a serverless database which scales automatically and backs up your data for protection.

Platform: Cross-platform

Languages: Go, Node.js, Java, .NET, C#, Ruby, Python, PHP, and Perl.

Cloud Version: Yes

Features:

- Key-value and document data model support.
- DynamoDB offers secondary indexes which provide the flexibility to query on any attribute.
- Amazon DynamoDB Accelerator delivers fast read performance for your DynamoDB.
- DynamoDB integrates with AWS Lambda to provide triggers.

Verdict: DynamoDB is a document database and can be used for various purpose.

Link: <https://aws.amazon.com/dynamodb/>

13) Neo4j



Neo4j is an open-source NoSQL graph database, implemented in Java. It saves your data in graphs rather than in tables.

Platform: Cross-Platform

Languages: Java, Cypher Query Language, JavaScript, Java, etc.

Cloud Version: Yes

Features:

- It supports graph analytics and transactional applications.
- Continuous-time traversals for a relationship in the graph both in breadth and depth because of double linking on the storage level between node and relationship.
- Relationship in Neo4j is fast and allows you to materialize and use new relationships later to “shortcut” and speed up the domain data when the new requirement arises
- Memory caching for graphs that provide compact storage, resulting in efficient scale-up.

Verdict: Neo4j allows any organization to unlock its business value of connections, relationships in data, and influences through a new application.

Link: <https://neo4j.com/>

14) Firebirdsql



Firebird is an open-source SQL RDBMS that runs on, Microsoft Windows, macOS, Linux, and several Unix platforms.

Platform: Linux, Windows, HP-UX, macOS, Solaris, and AIX.

Languages: C/C++ and COBOL

Cloud Version: Yes

Features:

- Firebird allows you to build a custom version.
- It is a free to download, registration as well as deployment.
- This simple database software tool has enhanced multi-platform RDBMS.
- Provides a range of funding options from firebird memberships to sponsorship commitments.

Verdict: Firebird has development-friendly language support, stored procedures, and triggers.

Link: <https://firebirdsql.org/>

FAQs

? What is Database Software?

A Database Software is a tool or application that helps users to manage SQL server and database server infrastructure. The Database Software allows users to configure, manage, monitor, and administer SQL servers and databases with ease. It is designed to extract information from databases.

📖 Which are the best FREE Database Software?

Below are some of the best Free database software:

- Microsoft SQL
- MySQL
- PostgreSQL
- MongoDB
- OrientDB
- MariaDB
- SQLite

⚡ How to choose the best Database Software?

You should consider the following factors while choosing the best Database Software:

- Support for both structured and unstructured data types
- Security features
- Integration with other software
- Scalability
- Performance
- User Interface and Navigation
- Support for multiple OS and Frameworks
- Server Administration and Server reporting
- Customization features

List of Popular Databases Software Free/Paid

Name	Link	Platform	DB Capacity Limit
InterBase	https://www.embarcadero.com/es/products/interbase	Windows, iOS, macOS, Android, Linux	Multi-CPU, 20 TB database.
Microsoft SQL	https://www.microsoft.com/en-in/sql-server/sql-server-2019	Windows, Linux.	1 GB RAM & 10 GB database. 1 CPU.
MySQL	https://www.mysql.com/	Windows, Linux, Mac.	No limitation
PostgreSQL	https://www.postgresql.org/	Windows, Linux, and Mac	No limitation
mongodb	https://www.mongodb.com/	Windows, Linux, Mac.	No limitation
OrientDB	https://www.orientdb.org/	Windows, Linux, Mac.	No limitation
Mariadb	https://mariadb.org/	Windows, Linux, Mac.	No limitation
SQLite	https://www.sqlite.org/index.html	Windows, Linux, Mac.	No limitation
Cassandra	http://cassandra.apache.org/	Windows, Linux.	No limitation
couchdb	https://couchdb.apache.org/	Windows, Linux.	maximum size is 4 GB
oracle	https://www.oracle.com/in/database/	Windows, Linux	1 GB RAM 11 GB database. 1CPU.
DynamoDB	https://aws.amazon.com/dynamodb/	Windows, Linux.	25 GB, 25 write capacity units and 25 read capacity units for AWS Free Tier
Neo4j	https://neo4j.com/	Windows, Linux, Mac.	free Startup License
firebirdsql	https://firebirdsql.org/	Windows, Linux, and Mac.	Multi-CPU, 20 TB database.


15 Best Database Design Tools | ER Diagram Tool [Free/Paid]

Database Design is a collection of processes that facilitate the design, development, implementation, and maintenance of database management systems (DBMS). Properly designed databases help you to improve data consistency for disk storage.

There are a wide range of software that helps you to design your database diagrams with ease. These database design tools can be used to create a physical model or ERD of your database so that you can quickly create tables and relationships.

Following is a handpicked list of Database Diagram Design Tools, with their popular features and website links. The list contains both open source(free) and commercial(paid) database design tools.

Top ERD Diagram Tool / Database Diagram Tools

Name	Price	Link
 DbSchema	Free Trial + Paid Plan	Learn More
Slickplan	Free Trial + Paid Plan	Learn More
Dbdiagram.io	Free + Paid Plan	Learn More
SqlDBM	Free + Paid Plan	Learn More
Dbdesigner.net	Free + Paid Plan	Learn More

1) DbSchema



DbSchema is a visual database designer & manager for any SQL, NoSQL, or Cloud database. The tool enables you to visually design & interact with the database schema, design the schema in a team and deploy it on multiple databases, generate HTML5 diagram documentation, visually explore the data and build queries, and so much more.

Features:

- Design schema in the team and deploy on multiple databases
- Compare different versions of the schema, generate migration scripts
- HTML5 Documentation , Interactive Diagrams, Relational Data Explorer & Visual Query Builder
- Schema Synchronization, Random Data Generator, Data Loader, Database Reports
- SQL Editor with autocompletion

More Information >>

2) Slickplan

[Slickplan](#) is an easy-to-use diagram and flowchart software that lets you create professional diagrams, flowcharts, and organizational charts. You can use Slickplan for simple task management or much more complex tasks.



Features:

- Build the optimal flow of your website by plotting diagrams in Slick plan.
- Visualize complex user flows through your website.
- Plot conditional elements and transition between pages with ease.
- Quickly and easily draw your diagrams.
- Draw diagrams with just a few mouse clicks

More Information >>

3) Dbdiagram.io



Dbdiagram.io is a simple database design tool to draw ER (Entity Relationship) diagrams by just writing code. It is one of the free erd tools designed for developers and data analysts.

Features:

- You can directly generate the SQL statements.
- It enables you to share your diagrams with your colleagues.
- Integrate with web frameworks like Django, Rails, etc.
- Generating diagrams from SQL databases is possible.
- Dbdiagram.io diagrams can be exported to image and PDF files.
- It supports both forward engineering and reverse engineering.

Link: <https://dbdiagram.io/home>

4) SqlDBM



SqlDBM is one of the best database diagram design tools that provides an easy way to design your database on any browser. You do not require any other database engine or database modeling tools or apps to use this program.

Features:

- It is one of the best database design tools which allows you to import an existing database schema.
- You can manage large and small databases and data models easily.
- Zoom in or out diagrams is possible.
- SqlDBM has two themes, dark and light.
- You can customize your project view by using modes like table names only, the description only, keys only.
- It enables you to copy or move columns across tables.
- This program helps you to share company's projects with your colleagues.
- You can create a physical model or ERD of your database.

Link: <https://sqldb.com/Home/>

5) Dbdesigner.net



Dbdesigner.net is an online database schema design and modeling tool. This database diagram tool allows you to create a database without wiring a single SQL code.

Features:

- It has a user-friendly UI for designing database structure.
- The tool offers team collaboration & sharing of projects within your organization.
- It allows you to import an existing database or start from scratch.
- You can collaborate with your team and work on the model together.
- Dbdesigner.net helps you to export your database in PNG and PDF file formats.

Link: <https://www.dbdesigner.net/>

6) Visual Paradigm



Visual Paradigm is a database design and management tool. This database diagram tool helps the product development team to build applications faster.

Features:

- It has a Drag-and-drop diagram editor.
- This tool enables you to export the database from ERD (Entity Relationship Diagram).
- It contains REST API for designing a database.
- You can work with your team on the same project simultaneously.
- Visual Paradigm enables you to build your report.
- It provides a wizard to make a database step by step.
- You can use Visual Paradigm on Windows, macOS, and Linux OS.

Link: <https://www.visual-paradigm.com/features/database-design-with-erd-tools/>

7) Erwin Data Modeler



Erwin is a tool which is used to create logical, physical, and conceptual data models. It provides centralize model management to business and technical users.

Features:

- It allows you to extract data from CRM, ERP, etc. for accurate modeling.
- Erwin Data Modeler has easy to use graphical environment.
- It automatically compares the model and database.
- You can manage structured and unstructured data from any database.

Link: <http://erwin.com/products/erwin-data-modeler/>

8) Lucidchart



Lucidchart is a HTML5 based erd diagram tool that allows you to create a complex database diagram. You can permanently delete any data or diagrams associated with your enterprise account.

Features:

- This database diagram tool allows you to connect live data with your diagrams.
- Lucidchart keeps your diagram secure using encryption.
- It is Integrated seamlessly with MS Office, G Suite, Atlassian (Issue tracking app), etc.
- You can work with your team on any device across various platforms.
- This er diagram tool helps you to easily manage user accounts.
- You can import data to automatically build organization charts.

More Information >>

9) QuickDBD



QuickDBD is a program that helps you to quickly draw a database diagram. It helps you to make your document looks professional.

Features:

- You can share your diagrams online.
- It enables you to draw schema (database structure) without leaving your keyboard.
- This database diagram tool has a user-friendly GUI.
- Diagrams can be dawn by typing.

Link: <https://www.quickdatabasediagrams.com/>

10) Toad World



Toad World is a database modeling software that helps you to tune application performance using an automated query rewriting facility. This software manages code change and promotes the highest levels of quality.

Features:

- Access key data quickly for analysis.
- It can easily identify differences by comparing and syncing servers, data, and schemas.
- Rollback transactions directly from the transaction log without need to restore from a backup.
- Get powerful query tuning capabilities.
- Execute scripts and T-SQL snippets for numerous instances and servers.
- Automate repetitive processes like data and schema comparisons.

Link: <https://www.toadworld.com/products/toad-data-modeler>

11) Dataedo



Dataedo is an app that enables you to create data dictionaries, ER diagrams, and document server scripts. It is an er diagram tool that enables you to easily document your relational databases.

Features:

- You can share documentation in interactive HTML.
- It helps you to visualize your data with database diagrams.
- This erd diagram tool allows you to add meaningful information about your database.
- It enables you to share documents in PDF, Excel, and HTML file formats.
- You can create table relationships (One to one, one to many, and many to many) with ease.

Link: <https://dataedo.com/>

12) Vertabelo



Vertabelo is online visual database design tool. It helps you to design your database at a logical and physical level.

Features:

- You can access database models anytime.
- It enables you to import an existing database.
- Vertabelo allows you to share the model with three access levels, like the owner, editor, or viewer.
- You can generate SQL script to create or remove elements from the database.
- Vertabelo automatically set the diagram layout.
- This app helps you to validate your model and workflow.
- You can provide a public link to your clients or partner so that they can view your design.

Link: <https://www.vertabelo.com/>

13) Dmodelaid



DModelAid is an online program for documenting database design in an interactive diagram. It helps you to retrieve large amounts of records from the database using SQL queries.

Features:

- You can visualize a table with tables with keys, indexes, and relationships.
- It supports keyboard shortcuts for easy access.

- This tool automatically documents your database project.
- You can create a project with Oracle, SQLite, MySQL, etc.
- DModelAid enables you to export script from the project to create the physical database.
- You can change the database any time you like, and it will map with the data type.

Link: <https://www.dmodelaid.com/>

14) SchemaSpy



SchemaSpy is a Java-based software that analyzes the metadata of a schema in your database. It is an er diagram tool which helps you to simplify the database designing process.

Features:

- This erd diagram tool supports JDBC (Java Database Connectivity) compliant DBMS.
- You can generate ER diagram for foreign keys.
- Schemaspy can produce database to HTML.

Link: <http://schemaspy.org/>

15) DeZign



DeZign is an er diagram tool that allows you to visualize your data structures to create a new database. This app also helps you to understand your existing database tables and relationships.

Features:

- It provides easy to use and robust data modeling tool for developers.
- You can use this tool to visually make Entity Relationship Diagram (ERD).
- Navigate large diagram with pan and zoom window.
- It supports a range of data modeling techniques.
- The latest version of this erd diagram tool helps you to reduce faults in database development.
- DeZign uses ERD to graphically design database.
- Exporting diagrams to bitmap, PNG, JPEG, and GIF is possible.

Link: <https://www.datanamic.com/dezign/index.html>

16) Database Designer for MySQL



Database Designer for MySQL is an erd diagram tool that helps you to construct build a graphical representation of tables and relationships.

Features:

- You can create and maintain the database effortlessly.
- It can generate reports that describe database objects within a diagram.
- It allows you to edit entity relationship diagram.
- Customization of diagrams and object appearance is possible.

- You can export a diagram to numerous formats, including BMP, JPEG, PNG, and more.
- Database Designer for MySQL helps you to edit and execute SQL scripts.
- Supports views (Virtual table), stored procedures (Set of SQL statements).
- This er diagram tool has a built-in database connection manager.

Link: <https://www.microolap.com/products/database/mysql-designer/>

17) Draw.io



Draw.IO is one of the free erd tools for online diagram design. It helps you to create and manage the drawing easily. This entity relationship diagram tool is compatible with all browsers like Chrome, Firefox, etc.

Features:

- No limit on the number of sizes.
- It allows you to save the model in your preferred location.
- This app provides a drag and drop feature.
- You can create a wide range of database diagrams, including UML (Unified Modeling Language), ERD, and much more.
- It provides readymade templates to design a database.
- You can work online and offline.
- Draw.io can be accessed from desktop and mobile devices.

Link: <https://app.diagrams.net/>

FAQ

? What is Database Design?

Database Design is a collection of processes that facilitate the designing, development, implementation, and maintenance of enterprise data management systems.

✓ Which are the Best Database Design Tools?

Here are some of the Best Database Design Tools:

- [DbSchema](#)
- [Slickplan](#)
- [Dbdiagram.io](#)
- [SqlDBM](#)
- [Dbdesigner.net](#)
- [Visual Paradigm](#)
- [Erwin Data Modeler](#)
- [Lucidchart](#)

⚡ What is Database Design Tools?

Database design tools can be used to create a physical model or ERD of your database so that you can quickly create tables and relationships.

🏆 Which factors should you consider while selecting Database Design Tool?

You should consider the following factors before selecting Database Design Tool:

- Ease of use.
- License Cost, if applicable.
- Quality of Customer support.
- The cost involved in training employees on the tool.
- Hardware/Software requirements of the tool.
- Support and Update policy of the tool vendor.

- Reviews of the company.

Top 50 Database (DBMS) Interview Questions & Answers (2022)

1) Define Database.

A prearranged collection of figures known as data is called database.

2) What is DBMS?

Database Management Systems (DBMS) are applications designed especially which enable user interaction with other applications.

3) What are the various kinds of interactions catered by DBMS?

The various kind of interactions catered by DBMS are:

- Data definition
- Update
- Retrieval
- Administration

4) Segregate database technology's development.

The development of database technology is divided into:

- Structure or data model
- Navigational model
- SQL/ relational model

5) Who proposed the relational model?

Edgar F. Codd proposed the relational model in 1970.



6) What are the features of Database language?

A database language may also incorporate features like:

DBMS-specific Configuration and management of storage engine

Computations to modification of query results by computations, like summing, counting, averaging, grouping, sorting and cross-referencing
Constraint enforcement
Application Programming Interface

7) What do database languages do?

As special-purpose languages, they have:

- Data definition language
- Data manipulation language
- Query language

8) Define database model.

A data model determining fundamentally how data can be stored, manipulated and organised and the structure of the database logically is called database model.

9) What is SQL?

Structured Query Language (SQL) being ANSI standard language updates database and commands for accessing.

10) Enlist the various relationships of database.

The various relationships of database are:

- One-to-one: Single table having drawn relationship with another table having similar kind of columns.
- One-to-many: Two tables having primary and foreign key relation.
- Many-to-many: Junction table having many tables related to many tables.

11) Define Normalization.

Organized data void of inconsistent dependency and redundancy within a database is called normalization.

12) Enlist the advantages of normalizing database.

Advantages of normalizing database are:

- No duplicate entries
- Saves storage space
- Boasts the query performances.

13) Define Denormalization.

Boosting up database performance, adding of redundant data which in turn helps rid of complex data is called denormalization.

14) Define DDL and DML.

Managing properties and attributes of database is called Data Definition Language(DDL).

Manipulating data in a database such as inserting, updating, deleting is defined as Data Manipulation Language. (DML)

15) Enlist some commands of DDL.

They are:

CREATE:

Create is used in the CREATE TABLE statement. Syntax is:

```
CREATE TABLE [column name] ( [column definitions] ) [ table parameters]
```

ALTER:

It helps in modification of an existing object of database. Its syntax is:

```
ALTER objecttype objectname parameters.
```

DROP:

It destroys an existing database, index, table or view. Its syntax is:

```
DROP objecttype objectname.
```

16) Define Union All operator and Union.

Full recordings of two tables is Union All operator. A distinct recording of two tables is Union.

17) Define cursor.

A database object which helps in manipulating data row by row representing a result set is called cursor.

18) Enlist the cursor types.

They are:

- Dynamic: it reflects changes while scrolling.
- Static: doesn't reflect changes while scrolling and works on recording of snapshot.
- Keyset: data modification without reflection of new data is seen.

19) Enlist the types of cursor.

They types of cursor are:

- Implicit cursor: Declared automatically as soon as the execution of SQL takes place without the awareness of the user.
- Explicit cursor: Defined by PL/ SQL which handles query in more than one row.

20) Define sub-query.

A query contained by a query is called Sub-query.

21) Why is group-clause used?

Group-clause uses aggregate values to be derived by collecting similar data.

22) Compare Non-clustered and clustered index

Both having B-tree structure, non-clustered index has data pointers enabling one table many non-clustered indexes while clustered index is distinct for every table.

23) Define Aggregate functions.

Functions which operate against a collection of values and returning single value is called aggregate functions

24) Define Scalar functions.

Scalar function is depended on the argument given and returns sole value.

25) What restrictions can you apply when you are creating views?

Restrictions that are applied are:

- Only the current database can have views.
- You are not liable to change any computed value in any particular view.
- Integrity constants decide the functionality of INSERT and DELETE.
- Full-text index definitions cannot be applied.
- Temporary views cannot be created.
- Temporary tables cannot contain views.
- No association with DEFAULT definitions.
- Triggers such as INSTEAD OF is associated with views.

26) Define “correlated subqueries”.

A ‘correlated subquery’ is a sort of sub query but correlated subquery is reliant on another query for a value that is returned. In case of execution, the sub query is executed first and then the correlated query.

27) Define Data Warehousing.

Storage and access of data from the central location in order to take some strategic decision is called Data Warehousing. Enterprise management is used for managing the information whose framework is known as Data Warehousing.

28) Define Join and enlist its types.

Joins help in explaining the relation between different tables. They also enable you to select data with relation to data in another table.

The various types are:

- INNER JOINS: Blank rows are left in the middle while more than equal to two tables are joined.
- OUTER JOINS: Divided into Left Outer Join and Right Outer Join. Blank rows are left at the specified side by joining tables in other side.

Other joins are CROSS JOINS, NATURAL JOINS, EQUI JOIN and NON-EQUI JOIN.

29) What do you mean by Index hunting?

Indexes help in improving the speed as well as the query performance of database. The procedure of boosting the collection of indexes is named as Index hunting.

30) How does Index hunting help in improving query performance?

Index hunting helps in improving the speed as well as the query performance of database. The followed measures are achieved to do that:

- The query optimizer is used to coordinate the study of queries with the workload and the best use of queries suggested based on this.
- Index, query distribution along with their performance is observed to check the effect.
- Tuning databases to a small collection of problem queries is also recommended.

31) Enlist the disadvantages of query.

The disadvantages of query are:

- No indexes
- Stored procedures are excessively compiled.
- Triggers and procedures are without SET NOCOUNT ON.
- Complicated joins making up inadequately written query.
- Cursors and temporary tables showcase a bad presentation.

32) Enlist ways to efficiently code transactions.

Ways to efficiently code transactions:

- User input should not be allowed while transactions.
- While browsing, transactions must not be opened of data.
- Transactions must be kept as small as possible.
- Lower transaction segregation levels.
- Least information of data must be accessed while transacting.

33) What is Executive Plan?

Executive plan can be defined as:

- SQL Server caches collected procedure or the plan of query execution and used thereafter by subsequent calls.
- An important feature in relation to performance enhancement.
- Data execution plan can be viewed textually or graphically.

34) Define B-trees.

A data structure in the form of tree which stores sorted data and searches, insertions, sequential access and deletions are allowed in logarithmic time.

35) Differentiate Table Scan from Index Scan.

Iterating over all the table rows is called Table Scan while iterating over all the index items is defined as Index Scan.

36) What do you mean by Fill Factor concept with respect to indexes?

Fill Factor can be defined as being that value which defines the percentage of left space on every leaf-level page that is to be packed with data. 100 is the default value of Fill Factor.

37) Define Fragmentation.

Fragmentation can be defined as a database feature of server that promotes control on data which is stored at table level by the user.

38) Differentiate Nested Loop, Hash Join and Merge Join.

Nested loop (loop over loop)

An outer loop within an inner loop is formed consisting of fewer entries and then for individual entry, inner loop is individually processed.

E.g.

- Select col1.*, col2.* from coll, col2 where coll.col1=col2.col2;

It's processing takes place in this way:

For i in (select * from col1) loop

For j in (select * from col2 where col2=i.col1) loop

Results are displayed;

End of the loop;

End of the loop;

The Steps of nested loop are:

- Identify outer (driving) table
- Assign inner (driven) table to outer table.
- For every row of outer table, access the rows of inner table.

Nested Loops is executed from the inner to the outer as:

- outer_loop
- inner_loop
- Hash join

While joining large tables, the use of Hash Join is preferred.

Algorithm of Hash Join is divided into:

- Build: It is a hash table having in-memory which is present on the smaller table.
- Probe: this hash value of the hash table is applicable for each second row element.
- Sort merge join

Two independent sources of data are joined in sort merge join. Their performance is better as compared to nested loop when the data volume is big enough but it is not good as hash joins generally. The full operation can be divided into parts of two:

Sort join operation :

Get first row R1 from input1

Get first row R2 from input2.

Merge join operation:

‘while’ is not present at either loop’s end.

if R1 joins with R2

next row is got R2 from the input 2

return (R1, R2)

else if R1 < style="”"> next row is got from R1 from input 1

else

next row is got from R2 from input 2
end of the loop

39) What is Database partitioning?

Division of logical database into independent complete units for improving its management, availability and performance is called Database partitioning.

40) Explain the importance of partitioning.

Splitting of one table which is large into smaller database entities logically is called database partitioning. Its benefits are:

- To improve query performance in situations dramatically when mostly rows which are heavily accessed are in one partition.
- Accessing large parts of a single partition
- Slower and cheaper storage media can be used for data which is seldom used.

41) Define Database system.

DBMS along with database is called Database system.

42) What do you mean by Query Evaluation Engine?

Query Evaluation Engine executes the low-level instructions that are generated by the compiler.

43) Define DDL Interpreter.

DDL statements are interpreted and recorded in tables called metadata.

44) Define Atomicity and Aggregation.

Atomicity: It's an all or none concept which enables the user to be assured of incomplete transactions to be taken care of. The actions involving incomplete transactions are left undone in DBMS.

Aggregation: The collected entities and their relationship are aggregated in this model. It is mainly used in expressing relationships within relationships.

45) Enlist the various transaction phases.

The various transaction phases are:

- Analysis Phase.
- Redo Phase
- Undo Phase

46) Define Object-oriented model.

Compilations of objects make up this model in which values are stored within instance variables which is inside the object. The object itself comprises bodies of object for its operation which are called methods. Objects containing same kind of variables and methods are called classes.

47) Define Entity.

It can be defined as being a 'thing' with an independent existence in the real world.

48) What do you mean by Entity type?

A set of entries having similar attributes are entity types.

49) Define Entity Set.

Compilation of all entries of any particular type of entry in the database is called Entity Set.

50) What do you mean by Entity type extension?

Compilation of similar entity types into one particular type which is grouped together as an entity set.

