### **What is a database?**

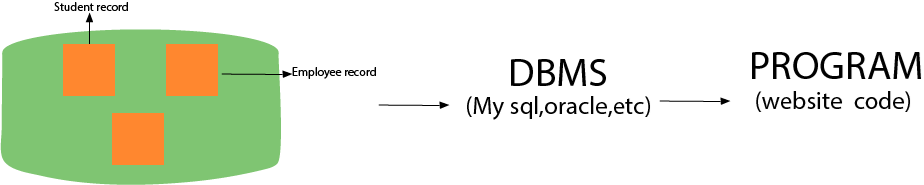
A Database is a logical, consistent and organized collection of data that it can easily be accessed, managed and updated. Databases, also known as electronic databases are structured to provide the facility of creation, insertion, updating of the data efficiently and are stored in the form of a file or set of files, on the magnetic disk, tapes and another sort of secondary devices. Database mostly consists of the objects (tables), and tables include of the records and fields. Fields are the basic units of data storage, which contain the information about a particular aspect or attribute of the entity described by the database. DBMS is used for extraction of data from the database in the form of the queries.

For more information: [Click here](https://www.javatpoint.com/what-is-database)

### **What is DBMS?**

DBMS is a collection of programs that facilitates users to create and maintain a database. In other words, DBMS provides us an interface or tool for performing different operations such as the creation of a database, inserting data into it, deleting data from it, updating the data, etc. DBMS is a software in which data is stored in a more secure way as compared to the file-based system. Using DBMS, we can overcome many problems such as- data redundancy, data inconsistency, easy access, more organized and understandable, and so on. There is the name of some popular Database Management System- MySQL, Oracle, SQL Server, Amazon simple DB (Cloud-based), etc.

Working of DBMS is defined in the figure below.



**What is a database system?**

The collection of database and DBMS software together is known as a database system. Through the database system, we can perform many activities such as-

The data can be stored in the database with ease, and there are no issues of data redundancy and data inconsistency.

The data will be extracted from the database using DBMS software whenever required. So, the combination of database and DBMS software enables one to store, retrieve and access data with considerate accuracy and security.

**What is a checkpoint in DBMS?**

The Checkpoint is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.

There are two ways which can help the DBMS in recovering and maintaining the ACID properties, and they are- maintaining the log of each transaction and maintaining shadow pages. So, when it comes to log based recovery system, checkpoints come into existence. Checkpoints are those points to which the database engine can recover after a crash as a specified minimal point from where the transaction log record can be used to recover all the committed data up to the point of the crash.

### **When does checkpoint occur in DBMS?**

A checkpoint is like a snapshot of the DBMS state. Using checkpoints, the DBMS can reduce the amount of work to be done during a restart in the event of subsequent crashes. Checkpoints are used for the recovery of the database after the system crash. Checkpoints are used in the log-based recovery system. When due to a system crash we need to restart the system then at that point we use checkpoints. So that, we don't have to perform the transactions from the very starting.

### **What do you mean by transparent DBMS?**

The transparent DBMS is a type of DBMS which keeps its physical structure hidden from users. Physical structure or physical storage structure implies to the memory manager of the DBMS, and it describes how the data stored on disk.

### **What is RDBMS?**

RDBMS stands for Relational Database Management Systems. **RDBMS**, was developed to facilitate easier data access and storage compared to DBMS. In RDBMS, data is stored in tables consisting of rows and columns. Examples of popular RDBMS systems are MySQL, Oracle DB, and others. RDBMS is the system that enables you to perform different operations such as- update, insert, delete, manipulate and administer a relational database with minimal difficulties. Most of the time RDBMS use SQL language because it is easily understandable and is used for often.

**Q1: What is the purpose of a Database Management System (DBMS)?**

Ans. A DBMS is a collection of tools that simplify the creation and maintenance of databases. It provides an interface for database creation, data entry, deletion, and updates. DBMS ensures data security, reduces redundancy and inconsistency, enables quick access, and organizes data effectively.

**Q3: What are the limitations of traditional file-based systems make DBMS a preferable choice?**

Ans. Traditional file-based systems lack indexing, leading to time-consuming content retrieval. Redundancy and inconsistency arise due to duplicate data, and updating one file causes inconsistency across all duplicates. Disorganized data makes it difficult to access information. Furthermore, unlike DBMS, traditional systems lack concurrency management, which allows simultaneous operations. DBMS addresses integrity checks, data isolation, atomicity, and security issues.

**Q4: Enumerate some advantages of using a DBMS.**

Ans. Here are a few benefits of utilizing a database management system (DBMS):

**Data sharing:** Multiple users can access data from the same database simultaneously.

**Integrity restrictions:** DBMS allows refined data storage with integrity constraints.

**Data redundancy control**: It provides a system for managing data redundancy by consolidating it into a single database.

**Data independence**: The data structure can be changed without impacting running applications.

**Backup and recovery**: DBMS offers automatic data backup and recovery capabilities.

**Data security**: DBMS ensures secure data storage and transfers through authentication and encryption techniques.

**Q5: What are the different languages used in DBMS?**

Ans. DBMS utilizes various languages for different purposes:

**Data Definition Language (DDL):** It includes commands for defining databases, such as CREATE, ALTER, DROP, TRUNCATE, and RENAME.

**Data Manipulation Language (DML):** This language allows data manipulation in a database using commands like SELECT, UPDATE, INSERT, DELETE, etc.

**Data Control Language (DCL):** DCL commands manage user permissions and controls within the database system, including GRANT and REVOKE.

**Transaction Control Language (TCL):** TCL provides commands to manage database transactions, such as COMMIT, ROLLBACK, and SAVEPOINT.

**Q6: What do the ACID properties signify in DBMS?**

Ans. The ACID properties in DBMS are fundamental principles that ensure data integrity. They are as follows:

**Atomicity** - It guarantees that a transaction is treated as a single unit of execution, either completing entirely or not at all.

**Consistency** - This property ensures that the database remains in a consistent state before and after each transaction.

**Isolation** - It allows multiple transactions to occur concurrently without interfering with each other.

**Durability** - Once a transaction is committed, its changes are permanently saved in non-volatile memory.

**Q7: Are NULL values in a database equivalent to zero or blank space?**

Ans. No, NULL values are distinct from zero and blank space. A NULL value represents an unknown, missing, or inapplicable value, whereas zero represents a numeric value and blank space represents a character. For example, a NULL value in the "number of courses" field indicates an unknown value, while a zero value indicates that no courses have been taken.

**Q8: What are the definitions of super key, primary key, candidate key, and foreign key?**

**Super Key:** A super key is a set of one or more columns (attributes) that can uniquely identify a row in a table. It can include additional attributes that are not necessary for unique identification. In other words, a super key is any combination of columns that uniquely identifies a record in a table.

**Primary Key:** A primary key is a specific type of super key that uniquely identifies each record in a table. A table can have only one primary key, and it cannot contain null values. The primary key is often chosen from the candidate keys and ensures that no duplicate records exist in the table.

**Candidate Key:** A candidate key is a minimal super key, meaning it is a set of attributes that uniquely identifies a record in a table, and no subset of these attributes can uniquely identify a record. A table can have multiple candidate keys, and one of them is selected as the primary key.

**Foreign Key:** A foreign key is a column or a set of columns in one table that establishes a link between the data in two tables. It acts as a cross-reference between tables by referencing the primary key of another table. Foreign keys ensure referential integrity, meaning that a value in the foreign key column must match a value in the primary key column of the referenced table or be null.

**Q9: What distinguishes a primary key from unique constraints?**

Ans. While a primary key cannot contain NULL values, unique constraints can. A table can have only one primary key, whereas it can have multiple unique constraints.

**Q13: What is meant by a Data Model?**

Ans. A data model is an abstract representation of the data structures and relationships within a database. It helps in defining the relationship between data entities and their attributes. Some common data models include hierarchical data models, network models, entity relationship models, and relational models. It defines how data is organized, stored, and manipulated, and includes elements such as entities, attributes, and relationships. Data models serve as blueprints for designing databases and are essential for ensuring data consistency, integrity, and efficient retrieval.

**Q15: How do entities differ from attributes in a database?**

Ans. In a database, an entity represents a real-world object, such as an employee, department, or designation. On the other hand, an attribute describes a specific characteristic or property of an entity. For example, an employee entity may have attributes like name, ID, and age, which provide additional information about the entity.

**Q16: What types of interactions can be handled by a DBMS?**

Ans. A DBMS can handle various interactions, including data definition (creating and modifying the structure of databases and tables), update (adding, modifying, or deleting data), retrieval (retrieving specific data from databases), and administration (managing user permissions, security, backups, etc.).

**Q17: Explain query optimization in DBMS.**

Ans. Query optimization is the process of identifying the most efficient plan for evaluating a query, minimizing the estimated cost. It involves selecting the best algorithms and approaches among multiple options to achieve the desired outcome. Query optimization aims to deliver query results faster, handle a larger number of queries in less time, and reduce the complexity of time and space requirements.

**Q18: Are NULL values equivalent to zero or blank space?**

Ans. No, NULL values are distinct from zero or blank space. While zero represents a numerical value and blank space represents a character, NULL denotes a value that is unavailable, unknown, assigned, or not applicable.

**Q19: Define aggregation and atomicity.**

Ans: **Aggregation** is a feature in the Entity-Relationship (E-R) model that allows one relationship set to interact with another relationship set.

**Atomicity** is a property that specifies a database alteration must adhere to all rules or none at all. If any part of a transaction fails, the entire transaction fails.

**Q20: What are the different levels of abstraction in DBMS?**

Ans. DBMS operates at three levels of data abstraction:

Physical Level: This is the lowest level of abstraction that defines how data is stored in the database system.

Logical Level: After the physical level, the logical level defines what data is stored in the database and how the data pieces relate to each other.

View Level: The highest level of abstraction, the view level, describes a specific portion of the entire database, focusing on a user's perspective or requirements.

**Q21: How would you define an entity-relationship model?**

Ans. An entity-relationship model is a graphical approach to database design where real-world objects are represented as entities, and the relationships between them are depicted. It provides a visual representation that allows the database administrators (DBAs) to quickly understand the schema.

**Q22: Explain the terms Entity, Entity Type, and Entity Set in DBMS.**

**Entity**: Think of an entity as a single thing or object. In our school database example, a student named "John Doe" is an entity. Another example could be a specific course like "Math 101."

**Entity Type**: An entity type is like a category or group that describes similar entities. For instance, "Student" is an entity type that defines all the students in the school. It has common characteristics or attributes like StudentID, Name, Age, and Grade. Similarly, "Course" is another entity type with attributes like CourseID, CourseName, and Credits.

**Entity Set:** An entity set is a collection of all entities of a particular entity type. In our school example: The entity set for "Student" includes all the students in the school, such as John Doe, Jane Smith, and others. The entity set for "Course" includes all the courses offered, such as Math 101, English 102, and Science 103.

**Simplified Example:**

**Entity:**

A specific student, John Doe.

A specific course, Math 101.

**Entity Type:**

"Student" (defines common attributes like StudentID, Name, Age, Grade).

"Course" (defines common attributes like CourseID, CourseName, Credits).

**Entity Set:**

All students in the school: John Doe, Jane Smith, etc.

All courses offered: Math 101, English 102, Science 103.

In a table format for the "Student" entity type, it might look like this:

| **StudentID** | **Name** | **Age** | **Grade** |
| --- | --- | --- | --- |
| 1 | John Doe | 15 | 10 |
| 2 | Jane Smith | 16 | 11 |
| 3 | Alice Brown | 14 | 9 |

**Here**:

Each row represents an entity (a specific student).

The table represents the entity set (all students).

The structure (columns) represents the entity type (attributes defining students).

**Q24: What are the unary operations in Relational Algebra?**

Ans. In relational algebra, unary operations are operations that work on a single operand. The two common unary operations are PROJECTION and SELECTION. These operations manipulate and retrieve data from a single relation. Additionally, RENAME is another unary operation used to rename attributes or relations in a relational algebra expression.

**Q27: Explain Relation Schema and Relation.**

A **relation schema** is a blueprint or template that defines the structure of a relation (table) in a database. It includes:

* **Relation Name**: The name of the table.
* **Attributes**: The columns or fields in the table, each with a specific name and data type.

The relation schema outlines the structure without containing any actual data. It specifies what kind of data can be stored and how the data is organized.

A **relation** is a table in a database that contains the actual data organized according to the structure defined by the relation schema. It consists of:

* **Tuples (Rows)**: Each row in the table represents a single record, with data for each attribute.
* **Attributes (Columns)**: Each column corresponds to an attribute defined in the relation schema.

A relation is essentially a collection of tuples that conform to the schema.

**Q28: What is the Degree of relation?**

In DBMS, the degree of a relationship indicates the number of entities involved in that relationship. This concept is crucial in Entity-Relationship (ER) modeling, which is used to design and structure databases.

### **Types of Relationship Degrees**

1. **Unary (Degree 1) Relationship**:
   * A unary relationship involves only one entity type.
   * Example: An employee is managed by another employee (self-referencing relationship).
     + Entity: Employee
     + Relationship: Manages (an employee manages another employee)
2. **Binary (Degree 2) Relationship**:
   * A binary relationship involves two different entity types.
   * This is the most common type of relationship in databases.
   * Example: A student enrolls in a course.
     + Entities: Student, Course
     + Relationship: Enrolls
3. **Ternary (Degree 3) Relationship**:
   * A ternary relationship involves three different entity types.
   * Example: A supplier supplies a part to a project.
     + Entities: Supplier, Part, Project
     + Relationship: Supplies
4. **N-ary (Degree N) Relationship**:
   * An N-ary relationship involves N different entity types.
   * Example: An academic consortium consisting of multiple universities, research projects, and funding bodies.
     + Entities: University, Research Project, Funding Body
     + Relationship: Collaborates

### **Examples in ER Diagrams**

* **Unary Relationship**:

Employee

|

Manages

|

Employee

* **Binary Relationship**:

Student ---- Enrolls ---- Course

* **Ternary Relationship**:

Supplier ---- Supplies ---- Part

|

Project

**Q29: Define Relationship in a database context.**

Ans. A relationship in a database refers to an association between two or more entities. It represents the way entities are related to each other. There are three types of relationships commonly used in database management systems:

**One-to-One:** A single record of one entity can be linked to a single record of another entity.

**One-to-Many (Many-to-One):** A single record of one entity can be linked to multiple records of another entity, and vice versa.

**Many-to-Many**: Multiple records of one entity can be linked to multiple records of another entity.

**Q30: What are the disadvantages of file processing systems?**

Data redundancy

Lack of security

Inconsistency

Difficulty in accessing data

Limited data sharing

Data integrity issues

Lack of concurrent access

Data isolation problems

Atomicity challenges

**Q31: Define Data Abstraction in DBMS.**

Ans. Data abstraction in DBMS refers to the process of hiding unnecessary details from users, allowing for simplified user interaction with the complex data structures of a database system. It involves presenting data in a simplified manner through layers of abstraction, ensuring that users can easily access and manipulate the data while focusing on the relevant aspects of the system.

**Q32: Why is the use of DBMS recommended? List some of its major advantages.**

**Controlled Redundancy**: DBMS allows for controlled data redundancy by storing data in a centralized database, reducing duplication and improving data consistency.

**Data Sharing:** DBMS enables simultaneous data sharing among multiple users and applications, as all users access the same shared database.

**Backup and Recovery Facility:** DBMS includes a built-in 'backup and recovery' feature, automating the process of creating data backups and restoring them when needed.

**Enforced Integrity Constraints:** DBMS enforces integrity constraints to maintain data integrity and ensure that only valid and consistent data is stored in the database.

**Data Independence:** DBMS provides data independence, allowing modifications to the data structure without affecting the structure of existing applications.

**Q33: What is the distinction between the HAVING and WHERE clause?**

Ans. In a SELECT statement, the WHERE clause is used to filter rows before grouping occurs. On the other hand, the HAVING clause is used to set conditions for groups or aggregate functions after grouping. The WHERE clause cannot contain aggregate functions, while the HAVING clause is specifically designed for such purposes.

**Q35: What is a Join?**

Ans. In SQL, a join is a technique used to combine data from two or more tables based on a common field. It allows for the retrieval of related data from multiple tables into a single result set.

**Q36: What does Identity represent in a database context?**

In a database context, **identity** often refers to a feature used to create unique identifiers for records in a table. This is typically implemented as an **identity column** or **auto-increment column**, which automatically generates a unique value for each new record inserted into the table. Here are key points about identity in databases:

**Q37: Define a view in SQL.**

A view in SQL is a virtual table that is based on the result set of an SQL query. It provides a way to simplify complex queries, encapsulate complex logic, and enhance security by restricting access to certain data. Views do not store data themselves; instead, they display data stored in other tables. Here are the key aspects of a view:

**View Definition**: Defined using the CREATE VIEW statement with a SELECT query.

**Virtual Nature**: Views do not store data themselves; they display data from the underlying tables.

**Query Simplification**: Simplifies complex queries by encapsulating them within a view.

**Security and Access Control**: Restricts user access to certain parts of the data.

**Updatability**: Some views can be updated directly, but this depends on the complexity and the SQL database system.

**Example :-**

CREATE VIEW EmployeeDepartments AS

SELECT e.EmployeeID, e.Name, d.DepartmentName

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID;

**Q38: What are the uses of views?**

Ans. Views have several uses in a database:

1. **Data Subset**: Views can represent a subset of data from one or more tables, limiting the exposure of underlying tables to users and allowing them to query specific portions of the data.

2. **Simplification**: Views can combine and simplify multiple tables into a single virtual table, making complex data structures more manageable and easier to work with.

3. **Aggregation**: Views can be used as aggregated tables, where the database engine performs aggregate functions (e.g., sum, average) and displays the results alongside the data.

4. **Data Complexity Hiding**: Views can hide the complexity of underlying tables by providing a simplified and user-friendly representation of the data.

5. **Minimal Storage**: Views occupy minimal storage space since they only store the definition of the view, not a copy of the entire data it displays.

6. **Enhanced Security**: Depending on the SQL engine used, views can provide additional security by restricting access to specific columns or rows of the underlying tables.

**Q39: What is a Trigger?**

Ans. A trigger is a piece of code associated with insert, update, or delete operations on a table. It is automatically executed when the associated query is run. A trigger is a special type of stored procedure in a database that automatically executes or fires when a specified event occurs in the database. Triggers are used to enforce data integrity, automate system tasks, and maintain business rules within a database. They are commonly used to monitor and respond to changes in data within a table.

CREATE TABLE EmployeeChanges (

ChangeID INT IDENTITY(1,1) PRIMARY KEY,

EmployeeID INT,

ChangeDate DATETIME DEFAULT GETDATE(),

OldName VARCHAR(50),

NewName VARCHAR(50)

);

CREATE TRIGGER trgAfterUpdate

ON Employees

AFTER UPDATE

AS

BEGIN

INSERT INTO EmployeeChanges (EmployeeID, OldName, NewName)

SELECT

d.EmployeeID,

d.Name AS OldName,

i.Name AS NewName

FROM

deleted d

INNER JOIN

inserted i ON d.EmployeeID = i.EmployeeID;

END;

**Q40: What is a stored procedure?**

A stored procedure is a named set of SQL statements that are stored in a database and can be executed multiple times. It is a precompiled collection of one or more SQL statements and procedural logic, which is stored as a schema object in a relational database management system (RDBMS). Stored procedures allow you to group and execute multiple SQL statements as a single unit, which provides several benefits:

Here's a simple example of a stored procedure in SQL Server that retrieves the list of employees from an Employees table:

CREATE PROCEDURE GetEmployees

AS

BEGIN

SELECT EmployeeID, Name, DepartmentID

FROM Employees;

END;

Once created, the stored procedure GetEmployees can be executed by simply calling its name:

EXEC GetEmployees;

**Q41: How does a Trigger differ from a Stored Procedure?**

Ans. Triggers cannot be directly called, unlike stored procedures. They are only associated with specific queries or actions.

**Explain dependency in DBMS.**

* A dependency is a constraint that governs or defines the relationship between two or more attributes.
* In a database, it happens when information recorded in the same table uniquely determines other information stored in the same table.

### **Functional Dependencies:**

A functional dependency (FD) is a relationship that exists between two attributes in a database, typically the primary key and additional non-key attributes. Consider it a link between two qualities of the same relation.

A dependency is denoted by an arrow "→".

If C determines D functionally, then C→D.

Functional dependency, indicated as C→ D, is a relationship between two sets of attributes, C and D. In this case, C is referred to as the "**determinant**", and Dis referred to as the "**dependent**".

### **Terms:**

**Dependent:** It is shown on the functional dependency diagram's right side.

**Determinant:** It is shown on the functional dependency Diagram's left side.

**Non-normalized table:** A table containing redundant data.

## Types of Functional Dependencies

### **1. Trivial Functional Dependency:**

1. A "dependent" in Trivial functional dependency is always a subset of the "determinant".
2. A functional dependency is said to be trivial if the attributes on its right side are a subset of the attributes on its left side.
3. If D is a subset of C, C→D is referred to as a Trivial Functional Dependency.

**Example: Take a look at the Student table below.**

|  |  |  |
| --- | --- | --- |
| **Roll\_No** | **S\_Name** | **S\_Age** |
| 1 | John | 13 |
| 2 | Riya | 12 |
| 3 | Giya | 15 |
| 4 | Jolly | 16 |

* {Roll\_No, S\_ Name} →S\_Name is a Trivial functional dependency in this case because the dependant S\_Name is a subset of the determinant {Roll\_No, S\_Name}.
* { Roll\_No } → { Roll\_No }, { S\_Name } → { S\_Name } and { S\_Age } → { S\_Age } are also Trivial.

### **2. Non-Trivial Functional Dependency**

* It is the inverse of Trivial functional dependence. Formally, a Dependent is a Non-Trivial functional dependency if it is not a subset of the determinant.
* If D is not a subset of C, C→D is said to have a non-trivial functional dependency. Non-trivial functional dependency is defined as a functional dependency C→ D where C is a set of attributes and D is also a set of attributes but not a subset of C.

|  |  |  |
| --- | --- | --- |
| **Roll\_No** | **S\_Name** | **S\_Age** |
| 1 | John | 13 |
| 2 | Riya | 12 |
| 3 | Giya | 15 |
| 4 | Jolly | 16 |

* Roll\_No→S\_Name is a non-trivial functional dependency in this case since S\_Name (dependent) is not a subset of Roll\_No (determinant).
* Similarly, {Roll\_No, Name}→ Age are non-trivial functional dependencies.

### **3. Multivalued Functional Dependency**

* In multivalued functional dependency, attributes in the dependent set are not dependent on one another.
* For example, C {D, Z} is referred to as a Multivalued functional dependency if there is no functional dependency between D and Z.

|  |  |  |
| --- | --- | --- |
| **Roll\_No** | **S\_Name** | **S\_Age** |
| 1 | John | 13 |
| 2 | Riya | 12 |
| 3 | Giya | 15 |
| 4 | Jolly | 16 |

* {Roll\_No}→ {S\_Name, S\_Age) is a Multivalued functional dependency in this case because the "dependent values" S\_ Name and S\_Age are not functionally dependent (i.e. S\_Name→S\_Age or S\_Age→S\_ Name does not exist)

### **4. Transitive Functional Dependency**

* Consider two functional dependencies, C→ D and D→Z; C→Z must exist according to the transitivity principle. This is referred to as a **Transitive Functional dependency**.
* In transitive functional dependency, the dependent is dependent on the determinant indirectly.

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll\_No** | **S\_Name** | **S\_Department** | **Street\_No** |
| 1 | John | AC | 12 |
| 2 | Riya | BH | 11 |
| 3 | Giya | MV | 14 |
| 4 | Jolly | CD | 18 |

* Roll\_No→S\_Department and S\_Department→Street\_No are correct here. As a result, Roll\_No→Street\_Number is a valid functional dependency, according to the principle of transitivity.

## Fully Functional Dependency

A functional dependency C→D, a fully functional dependency is one in which, if any attribute x from C is removed, the "dependency" no longer exists.

If D is "fully functional dependent" on C, it is not functionally dependent on any of the valid subsets of C.

i.e. Attribute Z in the relation CDE->Z is "fully functionally dependent" on CDE and not on any appropriate subset of CDE. That is, CDEsubsets such as CD, DE, C, D, and so on cannot determine Z.

|  |  |  |
| --- | --- | --- |
| **Seller\_Id** | **Product\_id** | **T\_price** |
| 1 | 1 | 530 |
| 2 | 1 | 535 |
| 1 | 2 | 100 |
| 2 | 2 | 101 |
| 3 | 1 | 342 |

According to the Table, neither **Seller\_id** nor **Product\_id** can uniquely determine the price, but both **Seller\_id** and **Product\_id** combined can.

As a result, we can say that **T\_price** is "fully functionally dependent" on **Seller\_id** nor **Product\_id**.

This outlines and demonstrates our fully functional dependency:

1. { Seller\_id , Product\_id } →T\_Price

## Partial Functional Dependency

A functional dependency C → D, If the dependency doe shold after removing any attribute x from C, then it is said to be a **Partial Functional Dependency**.

A functional dependency C→Y, If D is functionally dependent on C and may be determined by any appropriate subset of C, there is a partial dependency.

**Example:** We have a table called **Student** here.

**<Student>**

|  |  |  |
| --- | --- | --- |
| **Roll\_No** | **S\_Name** | **S\_Course** |
| 1 | John | DBMS |
| 2 | Riya | C++ |
| 3 | Giya | Java |
| 4 | Jolly | C |

We can see that the attributes S\_Name and Roll\_No can both uniquely identify a S\_Course. As a result, we might argue that the relationship is partly dependent.

### **Benefits of Functional Dependency**

* Functional Dependency prevents data duplication. As a result, the same data does not appear several times in that database.
* It assists you in maintaining the database's data quality.
* It assists you in defining database semantics and constraints.
* It aids you in spotting flawed designs.
* It aids you in locating database design information.
* The Normalization method begins with identifying the potential keys in the relation. It is impossible to locate candidate keys and normalize the database without functional dependencies.

**Q42: Explain the concept of database normalization.**

A large database defined as a single relation may result in data duplication. This duplication may result in:

* Making relations very large.
* It isn't easy to maintain and update data as it would involve searching many records in relation.

**Why do we need Normalization?**

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

## What is Normalization?

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.

**Data modification anomalies can be categorized into three types:**

* **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
* **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
* **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

## Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

**Following are the various types of Normal forms:**



|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| BCNF | A stronger definition of 3NF is known as Boyce Codd's normal form. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless. |

### **1NF - First Normal Form**

The most basic form of data normalization is 1NF which ensures there are no two same entries in a group. For a table to be in the first normal form, it should satisfy the following rules:

* Each cell should contain a single value
* Each record should be unique

**The table in 1NF will look like this:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Salutation** | **Full Name** | **Address** | **Skills** |
| Mr. | John Denver | 12, Bates Brothers Road | Content writing |
| Mr. | John Denver | 12, Bates Brothers Road | Social media marketing |
| Ms. | Mary Ann | 34,  Shadowman Drive | Machine Learning |
| Ms. | Mary Ann | 34,  Shadowman Drive | Data science |
| Ms. | Nancy Drew | 4, First Plot Street | DBMS |

### **2NF - Second Normal Form**

In a 2NF table, all the subsets of data that can be placed in multiple rows are placed in separate tables. For a table to be in the second normal form, it should satisfy the following rules:

* It should be in 1F
* The primary key should not be functionally dependant on any subset of candidate key

Let’s divide the 1NF table into two tables - Table 1 and Table 2. Table 1 contains all the employee information. Table 2 contains information on their key skills.

#### **Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee ID** | **Salutation** | **Full Name** | **Address** |
| 1 | Mr. | John Denver | 12, Bates Brothers Road |
| 2 | Ms. | Mary Ann | 34,  Shadowman Drive |
| 3 | Ms. | Nancy Drew | 4, First Plot Street |

#### **Table 2**

|  |  |
| --- | --- |
| **Employee ID** | **Key skills** |
| 1 | Content marketing |
| 1 | Social media marketing |
| 2 | Machine learning |
| 2 | Data science |
| 3 | DBMS |

We have introduced a new column called Employee ID which is the primary key for Table 1. The records can be uniquely identified using this primary key.

In Table 2, Employee ID is the foreign key.

### **3NF - Third Normal Form**

For a table to be in the third normal form, it should satisfy the following rules:

* It should be in 2F
* It should not have any transitive functional dependencies

A transitive functional dependency is when a change in a column (which is not a primary key) may cause any of the other columns to change.

In our example, if there is a name change (male to female), there may be a change in the salutation (Mr., Ms., Mrs., etc.). Hence we will introduce a new table that stores the salutations

#### **Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee ID** | **Full Name** | **Address** | **Salutation** |
| 1 | John Denver | 12, Bates Brothers Road | 1 |
| 2 | Mary Ann | 34,  Shadowman Drive | 2 |
| 3 | Nancy Drew | 4, First Plot Street | 2 |

#### **Table 2**

|  |  |
| --- | --- |
| **Employee ID** | **Key skills** |
| 1 | Content marketing |
| 1 | Social media marketing |
| 2 | Machine learning |
| 2 | Data science |
| 3 | DBMS |

#### **Table 3**

|  |  |
| --- | --- |
| **Salutation ID** | **Salutation** |
| 1 | Mr. |
| 2 | Ms. |
| 3 | Mrs. |

Now, there are no transitive functional dependencies and our table is now in 3F. Salutation ID is the primary key in Table 3. Salutation ID in Table 1 is foreign to the primary key in Table 3.

### **BCNF - Boyce and Codd Normal Form**

Boyce and Codd Normal Form is a higher version of 3NF and is also known as 3.5NF. A BCNF is a 3NF table that does not have multiple overlapping candidate keys. For a table to be in BCNF, it should satisfy the following rules:

* It should be in 3F
* For each functional dependency ( X → Y ), X should be a super key

# **Boyce Codd normal form (BCNF)**

* BCNF is the advance version of 3NF. It is stricter than 3NF.
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

**Example:** Let's assume there is a company where employees work in more than one department.

**EMPLOYEE table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_COUNTRY** | **EMP\_DEPT** | **DEPT\_TYPE** | **EMP\_DEPT\_NO** |
| 264 | India | Designing | D394 | 283 |
| 264 | India | Testing | D394 | 300 |
| 364 | UK | Stores | D283 | 232 |
| 364 | UK | Developing | D283 | 549 |

**In the above table Functional dependencies are as follows:**

1. EMP\_ID  →  EMP\_COUNTRY
2. EMP\_DEPT  →   {DEPT\_TYPE, EMP\_DEPT\_NO}

**Candidate key: {EMP-ID, EMP-DEPT}**

The table is not in BCNF because neither EMP\_DEPT nor EMP\_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

**EMP\_COUNTRY table:**

|  |  |
| --- | --- |
| **EMP\_ID** | **EMP\_COUNTRY** |
| 264 | India |
| 264 | India |

**EMP\_DEPT table:**

|  |  |  |
| --- | --- | --- |
| **EMP\_DEPT** | **DEPT\_TYPE** | **EMP\_DEPT\_NO** |
| Designing | D394 | 283 |
| Testing | D394 | 300 |
| Stores | D283 | 232 |
| Developing | D283 | 549 |

**EMP\_DEPT\_MAPPING table:**

|  |  |
| --- | --- |
| **EMP\_ID** | **EMP\_DEPT** |
| D394 | 283 |
| D394 | 300 |
| D283 | 232 |
| D283 | 549 |

**Functional dependencies:**

1. EMP\_ID   →    EMP\_COUNTRY
2. EMP\_DEPT   →   {DEPT\_TYPE, EMP\_DEPT\_NO}

**Candidate keys:**

**For the first table:** EMP\_ID  
**For the second table:** EMP\_DEPT  
**For the third table:** {EMP\_ID, EMP\_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

**Q43: What are indexes in a database?**

Ans. Indexes in a database are data structures designed to improve the speed of data retrieval operations on database tables. They trade off increased writes and storage space for faster access to data. Indexes allow for quicker searches based on specific values by organizing and storing data in a specific order.

**Q44: Differentiate between clustered and non-clustered indexes.**

Ans. Clustered indexes determine the physical order in which data is stored on a disk. Each database table can have only one clustered index. Non-clustered indexes, on the other hand, establish logical ordering of data rather than physical ordering. They often involve the creation of tree structures, such as B-trees or B+ trees, where leaves point to disk records.

**Q45: What is denormalization in a database?**

Ans. Denormalization is a technique used for database optimization, where duplicate data is intentionally introduced into one or more tables. This approach aims to improve performance by reducing the need for complex joins and increasing data retrieval speed.

### Examples:

**Merging Tables**:

1. **Normalized Form**:
   * Customers (CustomerID, CustomerName, Address, City)
   * Orders (OrderID, CustomerID, OrderDate, TotalAmount)
2. **Denormalized Form**:
   * CustomerOrders (CustomerID, CustomerName, Address, City, OrderID, OrderDate, TotalAmount)

**Q46: What is a clause in SQL?**

Ans. In SQL, a clause is a component of a query that allows for filtering or customizing the data being queried. It helps specify conditions or actions to be applied to the data.

**Q47: Define Livelock.**

Ans. Livelock occurs when two or more processes repeatedly engage in a futile interaction in response to changes in other processes. In this situation, the processes are continuously executing without making any progress. Unlike a deadlock, where processes are waiting, livelock involves active execution without achieving a desirable outcome.

**Q48: What is QBE in the context of databases?**

Ans. Query-by-example (QBE) is a visual/graphical technique used to retrieve information from a database by using skeleton tables as query templates. QBE allows users to express what they want to achieve by entering example values into the template, without the need for programming languages. It simplifies the process of accessing desired information by employing a two-dimensional syntax that resembles tables.

**Q49: Why are cursors necessary in embedded SQL?**

Ans. Cursors are used in embedded SQL to store the result of a query, allowing application programs to process the data row by row. SQL statements operate on sets of data and return another set of data, while host language programs typically work with data one row at a time. Cursors enable traversal through a set of rows generated by a SQL SELECT statement within the code, acting as a pointer for efficient row-by-row processing.

**Q51: How does a database schema differ from a database state?**

Ans. A database state refers to the actual collection of data stored in a database at a specific moment, while a database schema refers to the overall design and structure of the database.

**Q52: What is the purpose of SQL?**

Ans. SQL (Structured Query Language) serves as a language used to interact with relational databases. Its primary purpose is to perform operations such as querying, updating, and modifying data in the database.

**Q53: Explain the concepts of a Primary Key and Foreign Key.**

Ans. A Primary Key is a unique identifier for records in a database table, ensuring their uniqueness and serving as a reference point for data retrieval. A Foreign Key, on the other hand, establishes a relationship between two or more tables by referencing the Primary Key of another table.

**Q54: What are the main differences between a Primary Key and Unique Key?**

Ans. Here are the key differences between a Primary Key and Unique Key:

- A Primary Key cannot contain a null value, whereas a Unique Key can.

- Each table can have only one Primary Key, while multiple Unique Keys can exist in a table.

**Q55: What is the concept of a subquery in SQL?**

Ans. A subquery, also known as an inner query, is a query that is nested within another query. It is used to retrieve data that is based on the results of another query.

SELECT EmployeeName, Salary

FROM Employees

WHERE Salary > (

SELECT AVG(Salary)

FROM Employees );

**Q56: How is the DROP command used, and what are the differences between DROP, TRUNCATE, and DELETE commands?**

Ans. The DROP command is used in SQL to delete an existing table, database, index, or view from a database. The main differences between DROP, TRUNCATE, and DELETE commands are:

- DROP and TRUNCATE are DDL (Data Definition Language) commands used to delete tables and their associated indexes, while DELETE is a DML (Data Manipulation Language) command used to delete specific rows from a table.

- DROP removes the entire table structure, while TRUNCATE deletes all rows from a table but keeps the structure intact.

- DELETE allows the operation to be rolled back (undo), while DROP and TRUNCATE cannot be undone.

**Q57: What is the main difference between UNION and UNION ALL?**

Ans. UNION and UNION ALL are used to combine data from multiple tables, but they differ in terms of duplicate row handling. UNION removes duplicate rows and selects only distinct rows after merging the data, while UNION ALL does not remove duplicates and selects all rows from the tables.

**Q58: Define Correlated Subquery in DBMS.**

Ans. A correlated subquery is a subquery that is executed for each row of the outer query. It is a nested query where the inner query depends on the values from the outer query.

**OR**

A **correlated subquery** is a type of subquery in SQL where the subquery depends on the outer query for its values. Unlike a regular subquery, which is executed once and its result is used by the outer query, a correlated subquery is executed repeatedly, once for each row processed by the outer query.

#### Example:

Find the names of employees whose salaries are above the average salary in their respective departments.

| **EmployeeID** | **EmployeeName** | **DepartmentID** | **Salary** |
| --- | --- | --- | --- |
| 1 | John Doe | 1 | 60000 |
| 2 | Jane Smith | 2 | 75000 |
| 3 | Alice Jones | 1 | 55000 |
| 4 | Bob Brown | 3 | 80000 |
| 5 | Carol White | 2 | 72000 |

### Correlated Subquery Example:

SELECT EmployeeName, Salary

FROM Employees e

WHERE Salary > (

SELECT AVG(Salary)

FROM Employees

WHERE DepartmentID = e.DepartmentID

);

**Q59: What integrity rules exist in a DBMS?**

Ans. In a DBMS, two major integrity rules are:

- Entity Integrity: This rule ensures that the value of a primary key cannot be NULL.

- Referential Integrity: This rule is associated with foreign keys and ensures that the foreign key value is either NULL or matches the primary key of another relation.

**Q60: What is the E-R model in DBMS?**

Ans. The E-R (Entity-Relationship) model is a conceptual model used in relational databases. It represents entities, their attributes, and the relationships between them, providing a visual representation of the database structure and data dependencies.

**Q61: What is the significance of a functional dependency in a DBMS?**

Ans. A functional dependency in a DBMS is a constraint that describes the relationship between different attributes within a relation. For example, if a relation 'R1' has attributes Y and Z, the functional dependency between them can be represented as Y->Z, indicating that Z is functionally dependent on Y.

**Q62: How is pattern matching performed in SQL?**

Ans. Pattern matching in SQL is achieved using the LIKE operator. The 'percent' symbol (%) is used to match zero or more characters, while the underscore symbol (\_) is used to match a single character.

**Q63: What is Data Warehousing?**

Ans. Data warehousing involves collecting, extracting, processing, and importing data from various sources into a single database. It serves as a central repository for data analytics, receiving data from transactional systems and other relational databases. Data warehousing enables historical data storage and supports decision-making processes.

**Q64: Explain the distinction between intension and extension in a database.**

Ans. In a database, intension refers to the database schema or design, which remains relatively unchanged. Extension, on the other hand, represents the actual data stored in the database at a specific point in time, fluctuating as tuples are created, updated, or deleted.

**Q65: Compare the DELETE and TRUNCATE commands in a DBMS.**

Ans. The DELETE command is used to remove specific rows from a table based on conditions specified in the WHERE clause. In contrast, the TRUNCATE command is used to delete all data from a table without conditions.

**Q66: What is a lock, and how does a shared lock differ from an exclusive lock during a transaction in a database?**

Ans. A lock in a database prevents multiple users or sessions from updating the same data simultaneously. A shared lock allows multiple transactions to read data concurrently, while an exclusive lock ensures only one transaction can perform write operations to maintain data consistency.

**Q67: Describe the different normalization forms in a DBMS.**

Ans. The primary normalization forms in a DBMS include:

1. 1NF: Ensures atomicity of data and eliminates duplicate columns.

2. 2NF: Builds upon 1NF by ensuring non-key attributes are fully functionally dependent on the primary key.

3. 3NF: Extends 2NF by eliminating transitive dependencies between non-key attributes.

4. BCNF: Further refines 3NF by ensuring that no non-prime attribute is functionally dependent on another non-prime attribute.

**Q68: Explain the various types of keys in a database.**

Ans. In a database, there are several types of keys:

**Candidate Key**: A set of attributes that uniquely identifies each record in a table. A table can have multiple candidate keys, but each must uniquely identify the records.

**Super Key**: A superset of a candidate key, meaning it includes the candidate key and possibly additional attributes. It still uniquely identifies each record in the table.

**Primary Key**: A specific candidate key chosen to uniquely identify each record in the table. A primary key cannot contain NULL values and must be unique.

**Unique Key**: Similar to a primary key in that it must contain unique values across records, but unlike a primary key, it can contain NULL values. A table can have multiple unique keys.

**Alternate Key**: Candidate keys that are not chosen as the primary key. These are still capable of uniquely identifying records.

**Foreign Key**: An attribute or a set of attributes in one table that references the primary key of another table, establishing a relationship between the two tables.

**Q69: Differentiate between a 2-tier and 3-tier architecture in a DBMS.**

Ans. A 2-tier architecture involves direct interaction between client applications and the database server. In contrast, a 3-tier architecture adds an intermediary layer, providing a graphical user interface and enhanced security by separating the client-side application from the server-side application.

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Two-Tier Database Architecture** | **Three-Tier Database Architecture** |
| 1 | It is a Client-Server Architecture. | It is a Web-based application. |
| 2 | In two-tier, the application logic is either buried inside the user interface on the client or within the database on the server (or both). | In three-tier, the application logic or process resides in the middle-tier, it is separated from the data and the user interface. |
| 3 | Two-tier architecture consists of two layers : Client Tier and Database (Data Tier). | Three-tier architecture consists of three layers : Client Layer, Business Layer and Data Layer. |
| 4 | It is easy to build and maintain. | It is complex to build and maintain. |
| 5 | Two-tier architecture runs slower. | Three-tier architecture runs faster. |
| 6 | It is less secured as client can communicate with database directly. | It is secured as client is not allowed to communicate with database directly. |
| 7 | It results in performance loss whenever the users increase rapidly. | It results in performance loss whenever the system is run on Internet but gives more performance than two-tier architecture. |
| 8 | Example – Contact Management System created using MS-Access or Railway Reservation System, etc. | Example – Designing registration form which contains text box, label, button or a large website on the Internet, etc. |

**Q70: Explain the distinction between logical database design and physical database design and how this separation leads to data independence.**

Ans. Logical database design involves transforming the conceptual schema into a relational database structure, while physical database design focuses on storage, indexing, and optimization. This separation leads to data independence because the logical design is independent of the physical design, allowing changes to the physical implementation without affecting the logical representation.

**Q71: What are temporary tables and when are they beneficial?**

Ans. Temporary tables are tables that are used for a single session or for the duration of a transaction. They are commonly used to support unique rollups or specific application processing needs. Unlike permanent tables, temporary tables do not have pre-allocated space and dynamically allocate space as rows are added. In Oracle, you can create temporary tables using the CREATE GLOBAL TEMPORARY TABLE command.

**Q72: Define entity type extension.**

Ans. Entity type extension involves combining similar entity types into a single type and grouping them together as an entity set.

**Q73: What is conceptual design in DBMS?**

Ans. Conceptual design is the initial stage of the database design process. It aims to create a database that is independent of database software and physical implementation details. During conceptual design, a conceptual data model is developed, describing the primary data items, properties, relationships, and constraints within a specific domain.

**Q74: Explain the various types of failures that can occur in an Oracle database.**

Ans. Different types of failures that can occur in an Oracle database include:

- Bad data type

- Insufficient space

- Instance failure

- Media failure

- User process failure

- User error

- Statement failure

- Insufficient privileges

**Q75: What is the primary goal of RAID technology?**

Ans. The main goal of RAID (Redundant Array of Inexpensive Disks) technology is to improve fault tolerance and performance in storage systems. RAID combines multiple hard drives into a single logical unit, providing greater fault tolerance and throughput compared to individual drives. It is crucial for enhancing data reliability and system performance in various client/server applications.

**Q76: Define database partitioning and its significance.**

Ans. Database partitioning is the process of dividing a logical database into independent components to enhance availability, performance, and manageability. It allows for efficient access to specific partitions, enables data storage on low-cost storage devices, and improves query performance.

**Q77: Explain the functionality of a DML compiler.**

Ans. A DML (Data Manipulation Language) compiler is responsible for converting DML statements into a query language that can be understood by the query evaluation engine. Since DML has grammar elements similar to other programming languages, a DML compiler is necessary to compile the code into a language understood by the query evaluation engine, facilitating proper query execution.

**Q78: What is Relational Algebra?**

Ans. Relational Algebra is a procedural query language that consists of a set of operations applied to one or two relations to produce a new relation as output. It represents the basic set of operations in the relational model. Relational algebra resembles algebraic operations performed on numbers and includes operations like set difference, projection, selection, union, rename, and more.

**Q79: What is Relational Calculus?**

Ans. Relational Calculus is a non-procedural query language that utilizes mathematical predicate calculus instead of algebraic operations. It is not concerned with mathematical essentials like algebra, differential equations, integration, etc. Relational calculus is divided into two types: tuple relational calculus and domain relational calculus.

**Q80: Define durability in DBMS.**

Ans. Durability, in the context of DBMS, refers to the property that ensures the effects of a committed transaction persist even in the event of a system failure. Once a DBMS confirms the successful completion of a transaction, its changes are stored in non-volatile memory, providing durability by safeguarding the data against system failures.

**Q81: What is the significance of System R and what are its two major subsystems?**

Ans. System R was developed by IBM San Jose Research Centre from 1974 to 1979. It was the first relational database management system (RDBMS) to demonstrate improved transaction processing performance and implement SQL, the standard query language for relational data. System R served as a working prototype to address real-world problems.

System R consists of two major subsystems:

- Relational Data System

- Research Storage System

**Q82: How do you interact with an RDBMS?**

Ans. To interact with an RDBMS, you use Structured Query Language (SQL). SQL queries are used to provide input to the database, and the database processes these queries to generate the desired output.

**Q83. Explain the concepts of Proactive, Retroactive, and Simultaneous Update.**

Ans. - Proactive Update: Changes made to the database before it is operational in the real world.

- Retroactive Update: Updates applied to a database after it has been in use.

- Simultaneous Update: Updates applied to the database at the same time they become effective in the real world.

**Q84: Define specialization and generalization in the context of database design.**

**Specialization**: Specialization is a top-down design process where we define subclasses based on a superclass. It involves identifying subsets of entities that have unique characteristics or attributes. These subsets (subclasses) inherit attributes and relationships from a higher-level entity (superclass) but may also have their own specific attributes or relationships that differentiate them from others in the superclass.

**Generalization**: Generalization is the reverse process of specialization, where it involves combining common attributes and relationships from multiple entity types into a higher-level entity. It represents the relationship between a superclass and its subclasses, highlighting shared characteristics and allowing for abstraction of common features.

**Q85: What is the concept of Fill Factor in relation to indexes?**

Ans. Fill Factor refers to the percentage of space left on each leaf-level page of an index that is densely packed with data. The default value is typically 100, indicating full packing of index data.

**Q86: What is Index Tuning and how does it enhance query performance?**

Ans. Index Tuning involves optimizing a set of indexes to improve query performance and query processing time. It helps in query performance enhancement through the following steps:

- Suggesting optimal queries using query optimizers.

- Measuring the impact with indices, query distribution, and performance metrics.

- Optimizing databases for a small number of frequently executed queries.

**Q87: Explain what a deadlock is and how it can be resolved.**

Ans. A deadlock occurs when two transactions are waiting for resources that are mutually unavailable or when one transaction is waiting for another operation to complete. Deadlocks can be resolved by ensuring that all transactions acquire all required locks at the same time. If a deadlock is detected, one of the transactions needs to be aborted to break the deadlock and remove the incomplete work.

**Q88: When should you use an index?**

Ans. An index should be used when you want to enforce uniqueness in a database or when you need to simplify sorting and retrieve data quickly. Columns that are frequently used in queries are often good candidates for indexing.

**Q89: What is the difference between a clustered and a non-clustered index?**

Ans. A clustered index determines the physical order of data in a table, whereas a non-clustered index is a separate structure that points to the data in the table.

**Q90: What is a database transaction? Explain the concept of atomicity.**

A database transaction is a set of operations performed as a single logical unit of work, which must either be entirely completed or entirely failed (rolled back). Transactions ensure data integrity and consistency in databases, especially in environments where multiple users or processes may be accessing and modifying the data concurrently.

**Q91: What is a foreign key constraint?**

Ans. A foreign key constraint is a rule that ensures the referential integrity between two tables. It establishes a relationship between a column in one table and the primary key column in another table, enforcing data consistency and preventing orphaned records.

**Q92: What is the difference between a primary key and a unique key?**

Ans. A primary key is a column or set of columns that uniquely identifies each row in a table. It enforces entity integrity and cannot contain null values. On the other hand, a unique key ensures that the values in a column or set of columns are unique but allows null values.

**Q93: What is the purpose of the GROUP BY clause in SQL?**

Ans. The GROUP BY clause is used to group rows based on one or more columns in a SELECT statement. It is typically used in conjunction with aggregate functions (e.g., SUM, COUNT, AVG) to perform calculations on groups of data.

**Q94: What is the difference between a view and a table?**

Ans. A table is a storage structure that holds data in a relational database, while a view is a virtual table derived from one or more tables or views. Unlike tables, views do not store data physically but provide a way to query and manipulate data from multiple tables as a single entity.

**Q95: What is the purpose of the HAVING clause in SQL?**

Ans. The HAVING clause is used to filter rows in a SELECT statement based on conditions applied to groups defined by the GROUP BY clause. It is similar to the WHERE clause but operates on grouped data rather than individual rows.

**Q96: Explain the concept of referential integrity in database management.**

Ans. Referential integrity is a rule that ensures the consistency and integrity of data between related tables. It requires that foreign key values in one table match primary key values in another table, preventing the creation of orphaned or invalid records.

**Q97: What are the ACID properties in database transactions?**

Ans. ACID stands for Atomicity, Consistency, Isolation, and Durability. Atomicity ensures that a transaction is treated as a single unit of work. Consistency ensures that a transaction brings the database from one valid state to another. Isolation ensures that concurrent transactions do not interfere with each other. Durability ensures that once a transaction is committed, its changes are permanent and can survive system failures.

**Q98: What is the purpose of database normalization?**

Ans. Database normalization is the process of organizing data in a database to minimize redundancy and dependency issues. It aims to eliminate data anomalies and improve data integrity by dividing tables into smaller, well-structured entities and establishing relationships between them.

**Q99: What is the role of a database administrator (DBA) in a DBMS?**

Ans. A database administrator (DBA) is responsible for managing and maintaining a database system. Their roles include designing and implementing database structures, ensuring data security and integrity, optimizing database performance, managing user access and permissions, and performing backups and recovery.

**Q100: What is the difference between OLTP and OLAP databases?**

Ans. OLTP (Online Transaction Processing) databases are optimized for transactional processing, supporting day-to-day operations with a focus on fast and concurrent data modifications. OLAP (Online Analytical Processing) databases, on the other hand, are designed for analytical processing and reporting, providing complex querying capabilities and aggregations for decision-making and data analysis purposes.

### **1. What is a database transaction?**

A database transaction is a logical work unit performed within a database management system (DBMS). It represents a sequence of one or more operations, such as inserts, updates, or deletions, treated as a single, indivisible unit. Transactions ensure data integrity and consistency by completing all the operations successfully (commit) or rolling back the changes if an error occurs (rollback).

### **2. Explain the ACID properties of a transaction.**

ACID properties ensure that database transactions are reliable and maintain data integrity:

* Atomicity: A transaction is atomic, meaning it either completes in its entirety or not at all. If any part fails, the entire [transaction](https://www.simplilearn.com/online-transaction-processing-oltp-article) returns to its original state.
* Consistency: The database remains consistent before and after the transaction. All constraints, rules, and relationships defined in the database are enforced during the transaction.
* Isolation: Each transaction is isolated from other transactions until it is completed. This ensures that the intermediate state of one transaction is invisible to other concurrent transactions.
* Durability: Once a transaction is committed, its changes are permanent and persist even in system failure. The changes are stored permanently in non-volatile memory (e.g., disk).

### **3. What is concurrency control, and why is it important?**

Concurrency control manages simultaneous access to shared resources in a database by multiple users or transactions. It ensures that transactions execute correctly and maintain data consistency despite running concurrently. Concurrency control is essential because it prevents data corruption, maintains data integrity, and ensures the isolation of transactions from one another.

### **4. Describe the difference between optimistic and pessimistic locking.**

* Optimistic Locking: In optimistic locking, the system does not acquire locks on data resources until the transaction is ready to commit. It assumes that conflicts between transactions are rare. Instead of locking, it checks whether any other transaction has modified the data since it was last read. The transaction is aborted if a conflict is detected and the user is prompted to retry.
* Pessimistic Locking: In pessimistic locking, locks are acquired on data resources as soon as they are accessed. It assumes that conflicts between transactions are common. This approach ensures that other transactions cannot access the locked resources until the lock is released. Pessimistic locking can lead to reduced concurrency but guarantees data consistency.

### **5. What are deadlocks, and how can they be avoided?**

A deadlock occurs when two or more transactions wait for each other to release the resources needed to proceed. As a result, none of the transactions can continue, leading to a deadlock situation. Deadlocks can be avoided by implementing techniques such as:

* Deadlock Detection: Periodically check for deadlocks and resolve them automatically by rolling back one transaction.
* Deadlock Prevention: Use techniques like locking hierarchy, timeouts, or resource ordering to prevent deadlocks from occurring in the first place.
* Deadlock Avoidance: Use algorithms like the wait-for graph to ensure that transactions are ordered so that deadlocks cannot occur.

### **6. Explain the concept of transaction isolation levels.**

Transaction isolation levels define the extent to which a transaction is shielded from the influences of other concurrent transactions. The ANSI/ISO SQL standard defines four isolation levels:

* Read Uncommitted: Allows transactions to read data modified but not yet committed by other transactions. It has the lowest level of isolation and can lead to dirty reads.
* Read Committed: This ensures that transactions only read data other transactions commit. However, it may still result in non-repeatable reads and phantom reads.
* Repeatable Read: This guarantees that if a transaction reads a row once, it will see the same value every time it reads that row again within the same transaction. It prevents non-repeatable reads but may still allow phantom reads.
* Serializable: Provides the highest isolation level by ensuring that transactions are completely isolated. It prevents dirty, non-repeatable, and phantom reads but can result in reduced concurrency.

### **7. How does a database maintain data integrity during transactions?**

A database maintains data integrity during transactions by enforcing the [ACID](https://www.simplilearn.com/acid-properties-in-dbms-article) properties:

### **8. What is a two-phase commit?**

A two-phase commit (2PC) protocol ensures the atomicity of distributed transactions involving multiple databases or resources. It consists of two phases:

* Prepare Phase: In this phase, the transaction coordinator (typically the database management system) asks all participants (databases or resources involved in the transaction) to prepare to commit the transaction.
* Commit Phase: If all participants are prepared to commit, the transaction coordinator sends a commit command to all participants. If any participant is not prepared to commit, the coordinator sends a rollback command to all participants to abort the transaction.
* The two-phase commit protocol ensures that all participants commit the transaction, or none do, preventing inconsistencies in distributed systems.

### **9. Describe the role of a transaction log in a DBMS.**

A [transaction log](https://www.simplilearn.com/tutorials/sql-tutorial/sql-truncate-vs-delete) (also known as a redo log or audit trail) is a file that records all changes made to the database during transactions. It serves several important purposes:

* Recovery: In the event of a system failure, the transaction log can recover the database to a consistent state by replaying or undoing transactions.
* Concurrency Control: The transaction log can support concurrency control mechanisms such as locking and rollback, ensuring that transactions are isolated and maintaining data integrity.
* Audit Trail: The transaction log records all changes made to the database, enabling auditing and compliance with regulatory requirements.

### **10. What are savepoints in a transaction?**

Savepoints are markers within a transaction that allow you to define points you can roll back without rolling back the entire transaction. They provide a way to divide a transaction into smaller units and selectively undo parts of the transaction if necessary. Savepoints are useful in complex transactions where certain parts may fail but can be recovered without aborting the entire transaction. They allow for finer control over transaction management and recovery.

### **1. What is a distributed database, and what are its advantages?**

A distributed database is a system in which data is stored and managed across multiple computing devices or nodes, often in different geographical locations. Each node in a distributed database system contains a subset of the data, and nodes communicate with each other to provide users with a unified view of the data. The advantages of distributed databases include:

* Improved Scalability: Distributed databases can scale horizontally by adding more nodes, allowing them to handle larger volumes of data and higher numbers of users.
* Increased Availability: Data replication across multiple nodes improves fault tolerance and availability. If one node fails, other nodes can still access the data.
* Geographic Distribution: Distributed databases can store data closer to where needed, reducing latency and improving users' performance in different locations.
* Better Performance: Distributing data and processing across multiple nodes can improve query performance by parallelizing data retrieval and processing tasks.

### **2. Explain the concept of database replication.**

Database replication is copying data from one database to another in real-time or near real-time. Its primary purpose is to improve data availability, fault tolerance, and disaster recovery. In replication, changes made to the data in one database (the source database) are propagated to one or more other databases (the target databases) to ensure that they contain identical copies of the data.

### **3. What is a NoSQL database, and how does it differ from a relational database?**

A [NoSQL](https://www.simplilearn.com/rise-of-nosql-and-why-it-should-matter-to-you-article" \t "_blank) (Not Only SQL) database offers a method for storing and accessing data, diverging from the tabular structures employed in relational databases. It is designed to handle large volumes of structured, semi-structured, and unstructured data and is optimized for horizontal scalability and distributed data architectures. Unlike relational databases, NoSQL databases do not strictly adhere to the ACID properties and use different data models, such as key-value, document, columnar, or graph-based models.

### **4. Describe the CAP theorem and its implications for distributed systems.**

The CAP theorem states that it is impossible for a distributed computer system to provide all three of the following guarantees simultaneously:

* Consistency
* Availability
* Partition Tolerance

The CAP theorem implies sacrificing one of the three guarantees in a distributed system. Most distributed systems sacrifice consistency or availability in favor of partition tolerance, depending on the system's specific requirements.

### **5. How does sharding work in a database?**

[Sharding](https://www.simplilearn.com/tutorials/mongodb-tutorial) is a technique used in distributed databases to horizontally partition data across multiple servers or nodes. Each shard contains a subset of the data, comprising the entire dataset. Sharding can improve scalability and performance by distributing the workload across multiple nodes, allowing the database to handle larger volumes of data and higher numbers of concurrent users.

Sharding typically involves partitioning data based on a shard key, which determines which shard a particular piece of data belongs to. Each shard operates independently and can be located on a different physical server, providing fault tolerance and high availability.

### **6. What is a data warehouse, and how does it differ from a database?**

A data warehouse is a centralized storage facility that accumulates extensive structured and unstructured data from diverse origins, including transactional databases, CRM systems, and external sources. It is designed for querying and analysis rather than transaction processing. Data warehouses typically use a denormalized schema to optimize query performance, and they often employ techniques like data aggregation, indexing, and partitioning to improve query speed.

### **7. Explain the concept of data mining.**

Data mining involves uncovering patterns, trends, and valuable insights from extensive datasets using diverse methodologies such as statistical analysis, machine learning, and artificial intelligence. Its goal is to extract actionable information from raw data, enabling data-driven decision-making and predictive analytics. [Data mining](https://www.simplilearn.com/what-is-data-mining-article) techniques can uncover hidden patterns, relationships, and anomalies in data that may not be apparent through traditional methods. Data mining applications include customer segmentation, market basket analysis, fraud detection, and predictive modeling.

### **8. What is Big Data, and how does DBMS handle it?**

Big Data refers to large volumes of structured, semi-structured, and unstructured data that cannot be processed or analyzed using traditional database management tools and techniques. Big Data is characterized by its volume, velocity, variety, and veracity. Traditional DBMS may struggle to handle Big Data due to scalability, performance, and flexibility limitations.

Specialized Big Data platforms and technologies, such as Hadoop, Spark, and NoSQL databases, are used to handle Big Data. These technologies are designed to scale horizontally, process data in parallel, and handle diverse data types and sources.

### **9. Describe the role of an ORM (Object-Relational Mapping) tool.**

An Object-Relational Mapping (ORM) tool is a programming technique to convert data between incompatible type systems, such as object-oriented programming languages and relational databases. ORM tools provide a mechanism for mapping objects in the application code to tables in the database and mapping relationships between objects to foreign key relationships in the database schema.

ORM tools abstract away the complexities of database interaction, allowing developers to work with objects and classes rather than SQL queries. ORM tools also provide automatic schema generation, query building, and caching to improve developer productivity and performance.

### **10. How do database triggers work?**

Database triggers are special stored procedures automatically executed in response to certain events or actions in a database. [Triggers](https://www.simplilearn.com/tutorials/sql-tutorial/triggers) are used to enforce business rules, maintain data integrity, and automate database tasks. Database triggers are commonly used with constraints, stored procedures, and other objects to enforce complex business logic and ensure data consistency.

### **1. How can database performance be monitored and improved?**

Database performance can be monitored and improved through various techniques, including:

* Performance Monitoring: Regularly monitor key performance metrics such as CPU usage, memory usage, disk I/O, query execution times, and throughput.
* Query Optimization: Identify and optimize slow-running queries by using query execution plans, indexing, and rewriting queries for better performance.
* Database Indexing: Create and maintain indexes on columns frequently used in WHERE clauses and JOIN conditions to speed up data retrieval.
* Database Tuning: Configure database parameters, such as buffer sizes, cache sizes, and concurrency settings, to optimize performance for specific workloads.
* Hardware Upgrades: Upgrade hardware components, such as CPU, memory, storage, and network infrastructure, to improve overall system performance.
* Data Partitioning: This process involves partitioning large tables or indexes into smaller chunks to distribute data across multiple disks or servers, improving query performance.
* Regular Maintenance: Perform routine maintenance tasks such as vacuuming, reindexing, and updating statistics to ensure database health and optimal performance.

### **2. Explain the role of caching in database systems.**

Caching in database systems involves storing frequently accessed data or query results in memory for fast retrieval. By caching data in memory, database systems can reduce the need to access disk storage, which is slower than memory access. By serving data directly from memory rather than fetching it from disk, the cache can improve query performance and reduce latency for read-heavy workloads.

Common caching techniques include query caching, result caching, and data caching. However, if not managed properly, caching can lead to stale data, so cache invalidation mechanisms are often used to ensure that cached data remains up-to-date.

### **3. What are the common security threats to a database?**

Common security threats to a database include:

* Unauthorized Access: Unauthorized users gain access to sensitive data or database resources.
* SQL Injection: Attackers inject malicious SQL code into input fields to manipulate database queries and gain unauthorized access.
* Data Breaches: Unauthorized access or disclosure of sensitive data, often due to inadequate access controls or [encryption](https://www.simplilearn.com/data-encryption-methods-article).
* Data Manipulation: Malicious users modify or delete data, leading to data loss or corruption.
* Denial of Service (DoS): Attackers flooding the database server with requests to overload and disrupt its normal operation.
* Insider Threats: Malicious or negligent actions by employees or trusted users, such as stealing data or leaking sensitive information.

### **4. How does encryption protect database data?**

Encryption protects database data by converting it into a ciphertext that can only be decrypted with the appropriate decryption key. Encrypted data is unreadable and unintelligible to unauthorized users or attackers who gain unauthorized access to the database.

Encryption helps ensure data confidentiality by preventing unauthorized access to sensitive information, even if the database is compromised. Common encryption techniques used in database systems include column-level encryption, transparent data encryption (TDE), and data encryption in transit using SSL/TLS protocols.

### **5. What is SQL injection, and how can it be prevented?**

SQL injection is a type of cyber attack in which malicious SQL code is injected into input fields or parameters of a web application to manipulate the database query and gain unauthorized access to the database. SQL injection attacks can result in data leakage, data loss, unauthorized access, and database corruption. SQL injection attacks can be prevented by:

* Using parameterized queries or prepared statements to sanitize user input and prevent injection of malicious SQL code.
* Implementing input validation and data sanitization ensures user input conforms to expected formats and does not contain malicious characters.
* Escaping special characters in user input before including them in database queries.
* Limiting database privileges and access rights to minimize the impact of a successful SQL injection attack.
* Regularly updating and patching web applications and database systems to fix vulnerabilities that attackers could exploit.

### **6. Describe the purpose of database audits.**

Database audits monitor and track database activity, access, and changes to database objects and data. The purpose of database audits includes:

* Ensuring compliance with regulatory requirements and industry standards, such as [GDPR](https://www.simplilearn.com/what-is-gdpr-compliance-and-how-to-follow-it-article), HIPAA, PCI DSS, and SOX.
* Detecting and investigating security breaches, unauthorized access, and suspicious activities in the database.
* Identifying and mitigating security vulnerabilities, misconfigurations, and unauthorized changes to database objects.
* Providing an audit trail and forensic evidence for investigations, legal proceedings, and internal reviews.
* Improving accountability, transparency, and governance of database operations and data handling practices.

### **7. How can data redundancy be managed in a DBMS?**

Data redundancy can be managed in a DBMS through various techniques, including:

* Normalization: Organizing data into separate tables and eliminating redundant data by breaking it down into smaller, related tables.
* Denormalization: Introducing controlled redundancy by duplicating some data to improve query performance or simplify data retrieval.
* Use of Foreign Keys: Establishing relationships between tables using foreign keys to ensure data integrity and prevent redundant data.
* Data Deduplication: Identifying and removing duplicate records or data elements from the database to reduce redundancy.
* Data Compression: Using compression techniques to store data more efficiently and reduce storage requirements for redundant data.
* Regular Maintenance: Performing routine cleanup, data archiving, and data purging to remove outdated or unnecessary data from the database.

### **8. What is a database backup, and why is it important?**

A database backup is a copy of the database at a specific time, stored separately from the production database, typically on secondary storage devices or in the cloud. Database backups are important for several reasons:

* Disaster Recovery: Database backups are essential for recovering data in the event of data loss, database corruption, hardware failure, or other disasters.
* Data Protection: Database backups safeguard against accidental deletion, data corruption, or malicious attacks that could compromise data integrity.
* Business Continuity: Database backups help ensure business continuity by minimizing downtime and data loss in a disaster or system failure.
* Regulatory Compliance: Many regulatory requirements and industry standards mandate regular backups and data retention policies to protect sensitive information and ensure data availability.

### **9. How do you restore a database from a backup?**

To restore a database from a backup, follow these general steps:

* Identify the most recent backup of the database that you want to restore.
* Prepare the environment for the database restore, including ensuring enough storage space and the database server is available.
* Stop any services or applications that access the database to prevent data loss or corruption during restoration.
* Restore the database backup using the appropriate backup and restore tools or commands the database management system provides.
* Verify the integrity and completeness of the restored database by running consistency checks and testing data access and functionality.
* Restart services or applications that access the database once the restore process is complete and verified.

### **10. What are the best practices for database disaster recovery planning?**

Best practices for database disaster recovery planning include:

* Regular Backups: Implement regular database backups and test backup and restore procedures regularly to ensure data availability and integrity.
* Redundancy and Failover: Deploy redundant database [servers](https://www.simplilearn.com/what-is-a-cloud-server-article), storage systems, and network infrastructure to minimize single points of failure and ensure high availability.
* Disaster Recovery Site: Establish a disaster recovery site or secondary data center in a geographically separate location to ensure business continuity during a regional disaster.
* Automated Monitoring: Implement automated monitoring and alerting systems to detect and respond to potential issues, such as hardware failures, network outages, or data corruption.
* Documented Procedures: Document disaster recovery procedures, including roles and responsibilities, escalation paths, and contact information, to ensure a coordinated response during a disaster.
* Regular Testing: Conduct regular disaster recovery drills and tabletop exercises to test the effectiveness of disaster recovery plans and identify areas for improvement.
* Compliance and Governance: Ensure that disaster recovery plans comply with regulatory requirements and industry standards, and regularly review and update plans to address changing business needs and technology landscapes.

### **What do you understand by aggregation and atomicity?**

|  |  |
| --- | --- |
| **Aggregation** | **Atomicity** |
| This is a feature of the E-R  model which allows a relationship set to participate in another relationship set. | This property states that a database modification must either follow all the rules or nothing at all. So, if one part of the transaction fails, then the entire transaction fails. |

### **Q24. What do you understand by Proactive, Retroactive and Simultaneous Update?**

* **Proactive Update:** These updates are applied to the database before it becomes effective in the real-world environment.
* **Retroactive Update:** These retroactive updates are applied to a database after it becomes effective in the real-world environment.
* **Simultaneous Update:** These updates are applied to the database at the same instance of time as it becomes effective in a real-world environment.

### **Q27. What do you understand by cursor? Mention the different types of cursor**

A cursor is a database object which helps in manipulating data, row by row and represents a result set.

The types of cursor are as follows:

* **Implicit cursor:** This type of cursor is declared automatically as soon as the execution of [SQL](https://www.edureka.co/blog/sql-commands) takes place. Here, the user is not indicated about the declaration of the cursor.
* **Explicit cursor:** This type of cursor is defined by the PL/ SQL, as it handles a query in more than a single row.

### **Q28. Explain the terms specialization and generalization**

* **Specialization:** Specialization is a process of defining a set of subclasses of the entity type. Here, each subclass will contain all the attributes and relationships of the parent entity. Apart from this, the subclasses may contain additional attributes and relationships specific to itself.
* **Generalization:** Generalization is a process of finding relations, common attributes for a particular set of entities; and finally defining a common superclass for them.

### **Q29. What do you understand by Data Independence?**

When you say an application has data independence, it implies that the application is independent of the storage structure and data access strategies of data.

### **What are the differences between network and hierarchical database model?**

|  |  |
| --- | --- |
| **Network Database Model** | **Hierarchical Database Model** |
| Each parent node can have multiple children nodes and vice versa. | A top-down structure where each parent node can have many child nodes. But, a child node can have only a single parent node. |
| Supports one-to-one, one-to-many, and many-to-many relationships | Supports one-tone and one-to-many relationships |