**Database Management Concepts**

* **Introduction to Database System**

**What is Data?**

* **The raw facts are called as data. The word “raw” indicates that they have not been processed.**
* **Ex: For example 89 is the data.**

**What is information?**

* **The processed data is known as information.**
* **Ex: Marks: 89; then it becomes information.**

**DIFFERENCE BETWEEN DATA AND INFORMATION:**

| **DATA** | **INFORMATION:** |
| --- | --- |
| **1.Raw facts** | **1.Processed data** |
| **2. It is in unorganized form** | **2. It is in organized** |
| **3. Data doesn’t help in Decision** | **3. Information helps in Decision making process making proces** |

**FILE ORIENTED APPROACH:**

**The earliest business computer systems were used to process business records and produce information. They were generally faster and more accurate than equivalent manual systems. These systems stored groups of records in separate files, and so they were called file processing systems.**

**1. File system is a collection of data. Any management with the file system, user has to write the procedures**

**2.File system gives the details of the data representation and Storage of data.**

**3.In File system storing and retrieving of data cannot be done efficiently. 4. Concurrent access to the data in the file system has many problems like a Reading the file while other deleting some information, updating some information**

**5.File system doesn’t provide crash recovery mechanism. Eg. While we are entering some data into the file if System crashes then content of the file is lost.**

**6. Protecting a file under file system is very difficult. The typical file-oriented system is supported by a conventional operating system. Permanent records are stored in various files and a number of different application programs are written to extract records from and add records to the appropriate files.**

**DISADVANTAGES OF FILE-ORIENTED SYSTEM:**

**The following are the disadvantages of File-Oriented System:**

**Data Redundancy and Inconsistency:**

**Since files and application programs are created by different programmers over a long period of time, the files are likely to be having different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several places. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency.**

**Difficulty in Accessing Data:**

**The conventional file processing environments do not allow needed data to be retrieved in a convenient and efficient manner. Better data retrieval system must be developed for general use.**

**Data Isolation:**

**Since data is scattered in various files, and files may be in different formats, it is difficult to write new application programs to retrieve the appropriate data.**

**Concurrent Access Anomalies:**

**In order to improve the overall performance of the system and obtain a faster response time, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data.**

**Security Problems:**

**Not every user of the database system should be able to access all the data. For example, in banking system, payroll personnel need only that part of the database that has information about various bank employees. They do not need access to information about customer accounts. It is difficult to enforce such security constraints.**

**Integrity Problems:**

**The data values stored in the database must satisfy certain types of consistency constraints. For example, the balance of a bank account may never fall below a prescribed amount. These constraints are enforced in the system by adding appropriate code in the various application programs. When new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items for different files.**

**Atomicity Problem:**

**A computer system like any other mechanical or electrical device is subject to failure. In many applications, it is crucial to ensure that once a failure has occurred and has been detected, the data are restored to the consistent state existed prior to the failure**

**Example:**

**Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including:  A program to debit or credit an account  A program to add a new account  A program to find the balance of an account  A program to generate monthly statements**

**INTRODUCTION TO DATABASES:**

**History of Database Systems:**

**1950s and early 1960s:**

* **Magnetic tapes were developed for data storage**
* **Data processing tasks such as payroll were automated, with data stored on tapes.**
* **Data could also be input from punched card decks, and output to printers.**
* **Late 1960s and 1970s: The use of hard disks in the late 1960s changed the scenario fordata processing greatly, since hard disks allowed direct access to data.**
* **With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.**
* **With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.**
* **In the 1970’s the EF CODD defined the Relational Model.**

**In the 1980’s:**

* **Initial commercial relational database systems, such as IBM DB2, Oracle, Ingress, and DEC Rdb, played a major role in advancing techniques for efficient processing of declarative queries.**
* **In the early 1980s, relational databases had become competitive with network and hierarchical database systems even in the area of performance.**
* **The 1980s also saw much research on parallel and distributed databases, as well as initial work on object-oriented databases.**

**Early 1990s:**

* **The SQL language was designed primarily in the 1990’s.  And this is used for the transaction processing applications.**
* **Decision support and querying re-emerged as a major application area for databases.**
* **Database vendors also began to add object-relational support to their databases.**

**Late 1990s:**

* **The major event was the explosive growth of the World Wide Web.**
* **Databases were deployed much more extensively than ever before. Database systems now had to support very high transaction processing rates, as well as very high reliability and 24 \*7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities).**
* **Database systems also had to support Web interfaces to data.**

**The Evolution of Database systems:**

**The Evolution of Database systems are as follows:**

**1.File Management System**

**2.Hierarchical database System**

**3.Network Database System**

**4.Relational Database System**

**DATABASE:**

**A database is a collection of related data.**

**(OR)**

**A database is a collection of information that is organized so that it can be easily accessed,managed and updated.**

**Examples / Applications of Database Systems:**

**The following are the various kinds of applications/organizations uses databases for their business processing activities in their day-to-day life. They are:**

**1.Banking: For customer information, accounts, and loans, and banking transactions.**

**2. Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.**

**3. Universities: For student information, course registrations, and grades.**

**4.Credit Card Transactions: For purchases on credit cards and generation of monthly statements.**

**5. Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.**

**6. Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.**

**7. Sales: For customer, product, and purchase information.**

**8. Manufacturing: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.**

**9. Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.**

**10. Railway Reservation Systems: For reservations and schedule information.**

**11. Web: For access the Back accounts and to get the balance amount.**

**12. E –Commerce: For Buying a book or music CD and browse for things like watches, mobiles from the Internet**

**CHARACTERISTICS OF DATABASE:**

**The database approach has some very characteristic features which are discussed in detail below:**

**Structured and Described Data:**

**Fundamental feature of the database approach is that the database system does not only contain the data but also the complete definition and description of these data. These descriptions are basically details about the extent, the structure, the type and the format of all data and, additionally, the relationship between the data. This kind of stored data is called metadata ("data about data").**

**Separation of Data and Applications:**

**Application software does not need any knowledge about the physical data storage like encoding, format, storage place, etc. It only communicates with the management system of a database (DBMS) via a standardized interface with the help of a standardized language like SQL. The access to the data and the metadata is entirely done by the DBMS. In this way all the applications can be totally separated from the data.**

**Data Integrity:**

**Data integrity is a byword for the quality and the reliability of the data of a database system. In a broader sense data integrity includes also the protection of the database from unauthorized access (confidentiality) and unauthorized changes. Data reflect facts of the real world.**

**Transactions:**

**A transaction is a bundle of actions which are done within a database to bring it from one consistent state to a new consistent state. In between the data are inevitable inconsistent. A transaction is atomic what means that it cannot be divided up any further. Within a transaction all or none of the actions need to be carried out. Doing only a part of the actions would lead to an inconsistent database state.**

**Example: One example of a transaction is the transfer of an amount of money from one bank account to another.**

**Data Persistence:**

**Data persistence means that in a DBMS all data is maintained as long as it is not deleted explicitly. The life span of data needs to be determined directly or indirectly be the user and must not be dependent on system features. Additionally data once stored in a database must not be lost. Changes of a database which are done by a transaction are persistent. When a transaction is finished even a system crash cannot put the data in danger.**

**INTRODUCTION TO DATABASE-MANAGEMENT SYSTEM:**

**Database Management System:**

* **A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data.**
* **The DBMS is a general purpose software system that facilitates the process of defining constructing and manipulating databases for various applications.**

**Goals of DBMS:**

**The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient**

**1.Manage large bodies of information**

**2. Provide convenient and efficient ways to store and access information**

**3. Secure information against system failure or tampering**

**4. Permit data to be shared among multiple users**

**Properties of DBMS:**

**1.A Database represents some aspect of the real world. Changes to the real world reflected in the database.**

**2.A Database is a logically coherent collection of data with some inherent meaning.**

**3.A Database is designed and populated with data for a specific purpose.**

**Need of DBMS:**

**1. Before the advent of DBMS, organizations typically stored information using a “File Processing Systems”.**

**Example of such systems is File Handling in High Level Languages like C, Basic and COBOL etc., these systems have Major disadvantages to perform the Data Manipulation. So to overcome those drawbacks now we are using the DBMS.**

**2. Database systems are designed to manage large bodies of information.**

**3. In addition to that the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results**

**ADVANTAGES OF A DBMS OVER FILE SYSTEM:**

**Using a DBMS to manage data has many advantages:**

**Data Independence:**

**Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.**

**Efficient Data Access:**

**A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.**

**Data Integrity and Security:**

**If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.**

**Concurrent Access and Crash Recovery:**

**A database system allows several users to access the database concurrently. Answering different questions from different users with the same (base) data is a central aspect of an information system. Such concurrent use of data increases the economy of a system. An example for concurrent use is the travel database of a bigger travel agency. The employees of different branches can access the database concurrently and book journeys for their clients. Each travel agent sees on his interface if there are still seats available for a specific journey or if it is already fully booked. A DBMS also protects data from failures such as power failures and crashes etc. by the recovery schemes such as backup mechanisms and log files etc.**

**Data Administration:**

**When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals, who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and fine-tuning the storage of the data to make retrieval efficient.**

**Reduced Application Development Time:**

**DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of applications. Such applications are also likely to be more robust than applications developed from scratch because many important tasks are handled by the DBMS instead of being implemented by the application.**

**DISADVANTAGES OF DBMS:**

**Danger of a Overkill:**

**For small and simple applications for single users a database system is often not advisable.**

**Complexity:**

**A database system creates additional complexity and requirements. The supply and operation of a database management system with several users and databases is quite costly and demanding.**

**Qualified Personnel:**

**`The professional operation of a database system requires appropriately trained staff.Without a qualified database administrator nothing will work for long.**

**Costs:**

**Through the use of a database system new costs are generated for the system itself but also for additional hardware and the more complex handling of the system.**

**Lower Efficiency:**

**A database system is a multi-use software which is often less efficient than specialized software which is produced and optimized exactly for one problem.**

**DBMS FUNCTIONS:**

**DBMS performs several important functions that guarantee the integrity and consistency of the data in the database.**

**Those functions transparent to end users and can be accessed only through the use of DBMS.**

**They include:-**

* **Data Dictionary Management**
* **Data Storage Management**
* **Data transformation and Presentation**
* **Security Management**
* **Multiple Access Control**
* **Backup and Recovery Management**
* **Data Integrity Management**
* **Database Access Languages**
* **Databases Communication Interfaces**

**Data Dictionary Management:**

**DBMS stores definitions of database elements and their relationship (Metadata) in the data dictionary.The DBMS uses the data dictionary to look up the required data component structures and relationships.**

**Any change made in database structure is automatically recorded in the data dictionary.**

**Data Storage Management:**

**Modern DBMS provides storage not only for data but also for related data entities. Data Storage Management is also important for database “performance tuning”.Performance tuning related to activities that make database more efficiently.**

**Data Transformation and Presentation:**

**DBMS transforms entered data to confirm to required data structures.**

**DBMS formats the physically retrieved data to make it confirms to user’s logical expectations. DBMS also presents the data in the user’s expected format.**

**Security Management:**

**DBMS creates a security system that enforces the user security and data privacy.Security rules determines which users can access the database, which data items each user can access etc.DBA and authenticated user logged to DBMS through username and password or through Biometric authentication such as Finger print and face reorganization etc.**

**Multiuser Access Control:**

**To provide data integrity and data consistency, DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of database.**

**Backup and Recovery Management:**

**DBMS provides backup and recovery to ensure data safety and integrity.Recovery management deals with the recovery of database after failure such as bad sector in the disk or power failure. Such capability is critical to preserve database integrity.**

**Data Integrity Management:**

**DBMS provides and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. Ensuring data integrity is especially important in transaction- oriented database systems.**

**Data Dictionary:-**

**Data Dictionary is made up of two words, data which means the collected information through multiple sources, and dictionary meaning the place where all this information is made available.**

**A data dictionary is a crucial part of a relational database as it provides additional information about the relationships between multiple tables in a database. The data dictionary in DBMS helps the user to arrange data in a neat and well-organized way, thus preventing data redundancy.**

**Below is a data dictionary describing the table containing employee details.**

| **Attribute Name** | **Data Type** | **Max Field Size** | **Description** | **isRequired** |
| --- | --- | --- | --- | --- |
| **Employee ID** | **Integer** | **10** | **A unique ID for each Employee** | **Yes** |
| **Name** | **Text** | **25** | **Name of the Employee** | **Yes** |
| **Date of Birth** | **DateTime** | **10** | **Date of Birth of the Employee** | **Yes** |
| **Mobile Number** | **Integer** | **10** | **Contact Number of the Employee** | **Yes** |

**Some advantages of using a data dictionary are:**

1. [**Data models in DBMS**](https://www.scaler.com/topics/dbms/data-models-in-dbms/)**provide very little information about the database, so a data dictionary is very essential to have proper knowledge about entities,**[**relationships**](https://www.scaler.com/topics/types-of-relationship-in-dbms/)**, and attributes that are present in a data model.**
2. **The Data Dictionary provides consistency by reducing data redundancy in the collection and use of data across various members of a team.**
3. **The Data Dictionary provides structured analysis and design tools by enforcing the use of data standards. Data standards are the set of rules that govern the way data is collected, recorded, and represented.**
4. **Using a Data Dictionary helps to define naming conventions that are used in a model.**

* **Types of Data Dictionary**

**Now let us talk about different types of it :**

**1. Active Data dictionary**

**A huge responsibility of the database management system is to make sure that the database structure change should immediately be reflected in the data dictionary. It follows this immediate update methodology known as an active data dictionary as they are self-updating.**

**2. Passive data dictionary**

**These dictionaries are more helpful and easy to handle than active data dictionaries. Every time someone modifies the database, they need to manually update the dictionary to keep it in sync with the database. The responsibility of updating the dictionary lies with the individuals making the modifications. While it may seem counterintuitive that the database does not automatically update the dictionary, there are several advantages to using a passive data dictionary for storing metadata as a central repository. Hence, it can be used for multiple databases that are virtually the same simultaneously.**

**DATABASE USERS & DATABASE ADMINISTRATORS:**

**People who work with a database can be categorized as database users or database administrators.**

**Database Users:**

**There are four different types of database-system users, differentiated by the way they expect to interact with the system.**

**Naive users:**

**Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.**

**For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.**

**Application programmers:**

**Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.**

**Sophisticated users:**

**Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category.**

**Specialized users:**

**Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.**

**Database Administrator:**

**One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a database administrator (DBA)**

* **Database Administrator Functions/Roles:**

**The functions of a DBA include:**

**Schema definition:**

**The DBA creates the original database schema by executing a set of data definition statements in the DDL, Storage structure and access-method definition. Schema and physical-organization modification: The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.**

**Granting of authorization for data access:**

**By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.**

**Routine maintenance:**

**Examples of the database administrator’s routine maintenance activities are:**

**1. Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.**

**2. Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.**

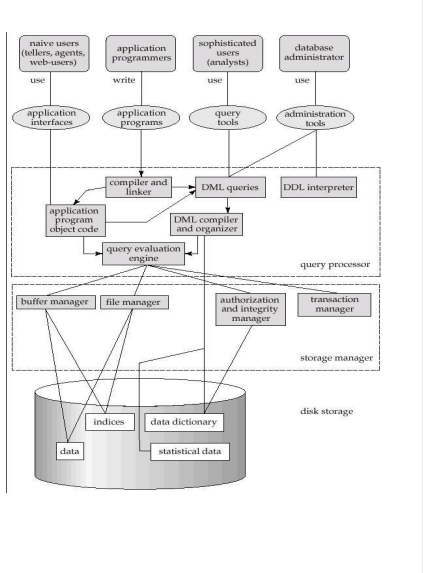
**3. Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.**

**DATABASE SYSTEM STRUCTURE:**

**A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.**

**The storage manager is important because databases typically require a large amount of storage space. Some Big organizations Database ranges from Giga bytes to Tera bytes. So the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.**

**The query processor also very important because it helps the database system simplify and facilitate access to data. So quick processing of updates and queries is important. It is the job of the database system to translate updates and queries written in a nonprocedural language,**

****

**Storage Manager:**

**A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager translates the various DML statements into low-level file-system commands.**

**Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.**

**Storage Manager Components:**

**Authorization and integrity manager which tests for the satisfaction of integrity constraints and checks the authority of users to access data.**

**Transaction manager which ensures that the database itself remains in a consistent state despite system failures, and that concurrent transaction executions proceedwithout conflicting.**

**File manager: which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.**

**Buffer manager which is responsible for fetching data from disk storage into main memory. Storage manager implements several data structures as part of the physical system implementation. Data files are used to store the database itself. Data dictionary is used to stores metadata about the structure of the database, in particular the schema of the database.**

**Query Processor Components:**

**DDL interpreter: It interprets DDL statements and records the definitions in the data**

**dictionary.**

**DML compiler: It translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.**

**Query evaluation engine: It executes low-level instructions generated by the DML compiler.**

**Application Architectures:**

**Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, onwhich remote database users’ work, and server machines, on which the database system runs.**

**Database applications are usually partitioned into two or three parts. They are:**

**1.Two – Tier Architecture**

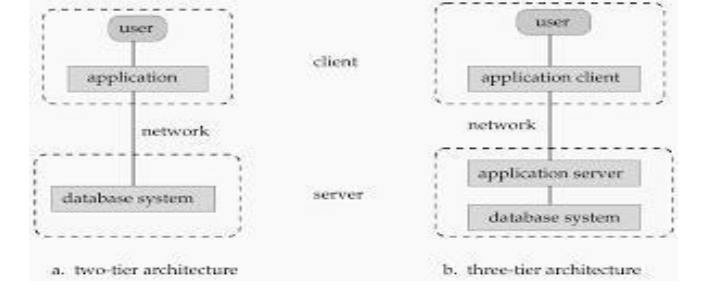
**2.Three – Tier Architecture.**

**Two-Tier Architecture:**

**The application is partitioned into a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.**

**Three-Tier Architecture:**

**The client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide**



**DATA ABSTRACTION:**

**Hiding certain details of how the data are stored and maintained. A major purpose of database system is to provide users with an “Abstract View” of the data. In DBMS there are 3 levels of data abstraction. The goal of the abstraction in the DBMS is to separate the users request and the physical storage of data in the database.**

**Levels of Abstraction:-**

**Physical Level:**

* **The lowest Level of Abstraction describes “How” the data are actually stored.**
* **The physical level describes complex low level data structures in detail.**

**Logical Level**

* **This level of data Abstraction describes “What” data are to be stored in the database and what relationships exist among those data.**
* **Database Administrators use the logical level of abstraction.**

**View Level:**

* **It is the highest level of data Abstracts that describes only part of entire database.**
* **Different users require different types of data elements from each database.**
* **The system may provide many views for the some database.**

**SQL Commands:**

**The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their nature.**

**They are:**

* **DDL Commands**
* **DML Commands**
* **DCL Commands**
* **DRL/DQLCommands**
* **TCL Commands**
* **Data Definition Language (DDL) Commands:**

**CREATE Creates a new table, a view of a table, or other object in database**

**ALTER Modifies an existing database object, such as a table.**

**DROP Deletes an entire table, a view of a table or other object in the database.**

**TRUNCATE Truncates the table values without delete table structure**

* **Data Manipulation Language (DML) Commands:**

**INSERT Creates a record**

**UPDATE Modifies records**

**DELETE Deletes records**

* **Data Control Language (DCL) Commands:**

**GRANT Gives a privilege to user**

**REVOKE Takes back privileges granted from user**

* **Data Query Language (DQL) Commands:**

**SELECT Retrieves certain records from one or more tables**

* **Transaction Control Language (TCL) Commands:**

**commit Save work done**

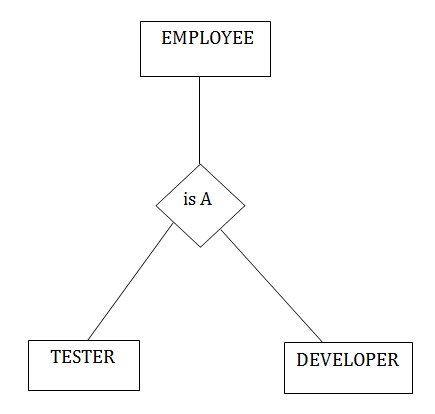
**Save point Identify a point in a transaction to which we can later roll back.**

**Roll backs Restore database to original since the last Commit**

# Specialization

* **Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.**
* **Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.**
* **Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.**

**For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.**

****

## Specialization in DBMS

A software developer is a person who can develop software. Now the software can be a web application, a mobile application, or, say, a desktop application. Hence, we can have developers specializing in a particular field, whether it's desktop applications, mobile applications, or desktop applications. This intuition of categorizing or dividing a higher-level entity into multiple lower-level entities of a similar kind is known as *specialization*.

### Specialization Characteristics:

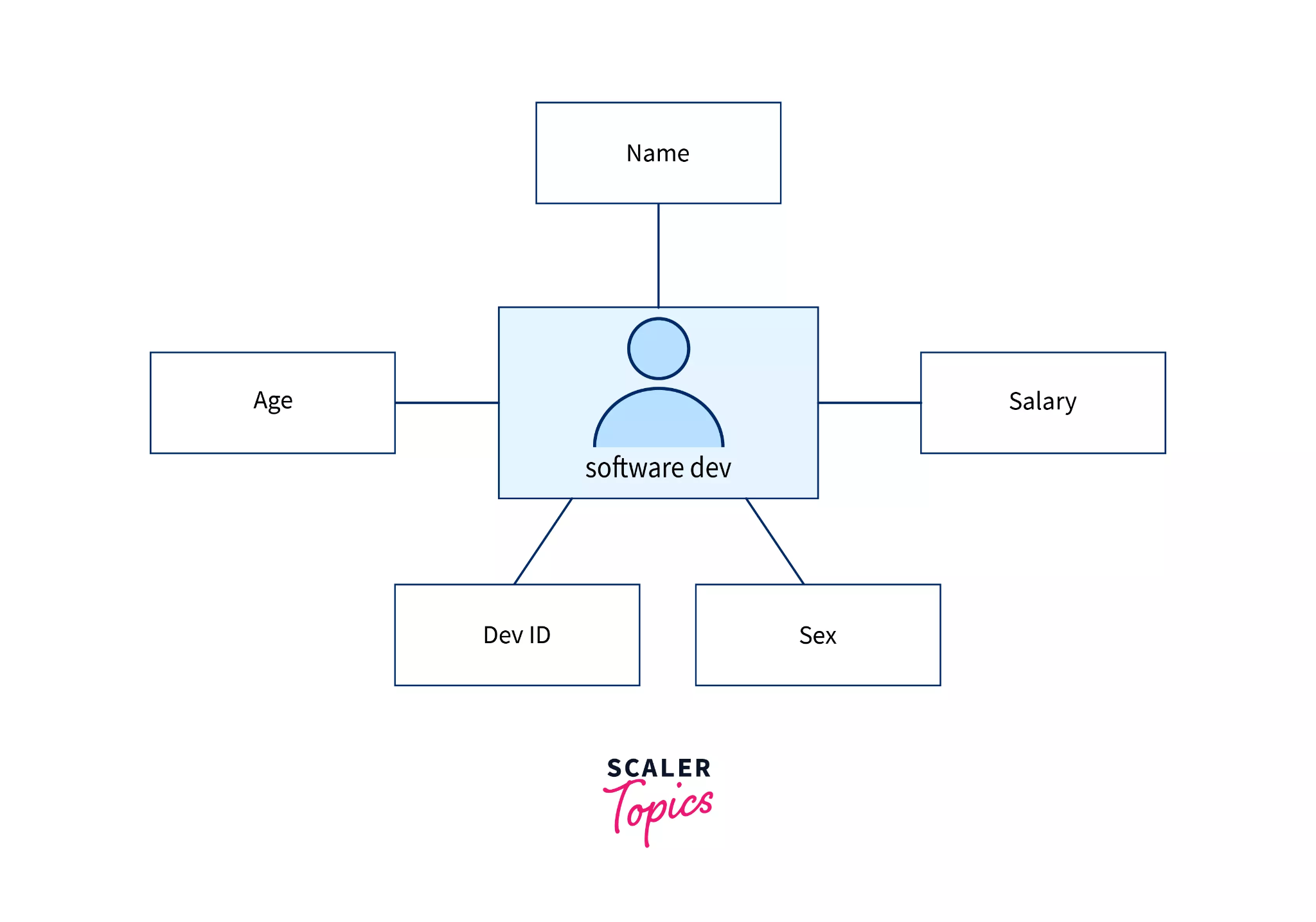
* The top-down technique is used for specialization.
* It divides or specializes in the entity.
* Higher-level entities and lower-level entities can be linked.

**Let us understand it with the help of an example.**

### Entities Before Specialization

Below will be the entities before specialization:

Software Developer: Software developers can have attributes like name, age, salary, etc. We will apply specialization to this entity and will divide it into entities.

****

## Generalization in DBMS

**When someone says, "Social media", what comes to mind?**

**Facebook, Twitter, Instagram, or maybe LinkedIn**

**In short, if we want to talk about Social media in general, we generalize and refer to them as "Social Media". This method of combining multiple lower-level entities of a similar kind and referring to them as a single higher-level entity is what we know as *generalization*.**

### Generalization Characteristics:

* **The bottom-up technique is used for generalization.**
* **It simplifies or generalizes the entities.**
* **Higher-level entities and lower-level entities can be linked.**

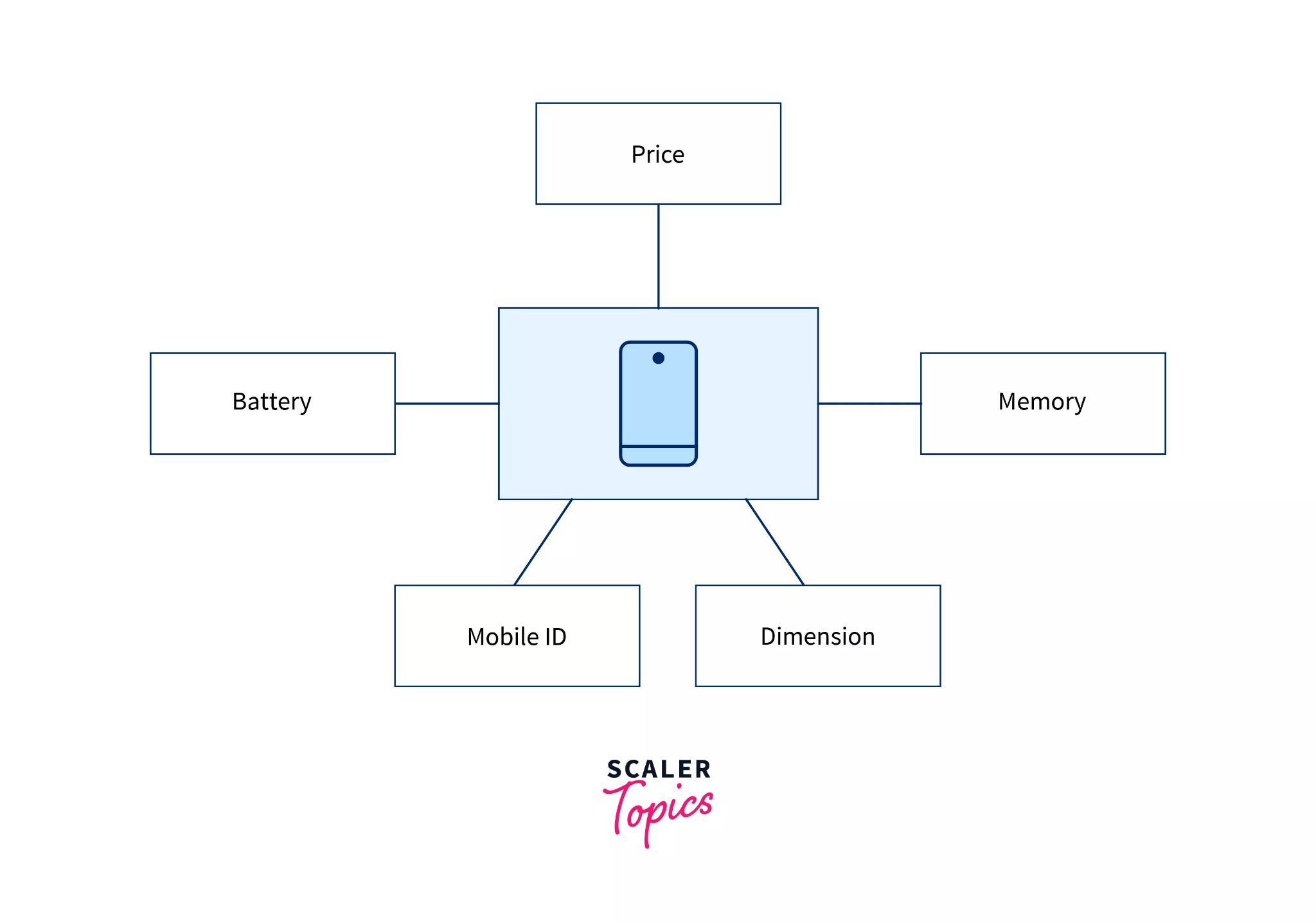
**Let us take an example.**

**Let's say we have entities as mobile and laptop and we need to generalize it based on the common entities between them so before generalization, we will have two separate entities for mobiles and laptops.**

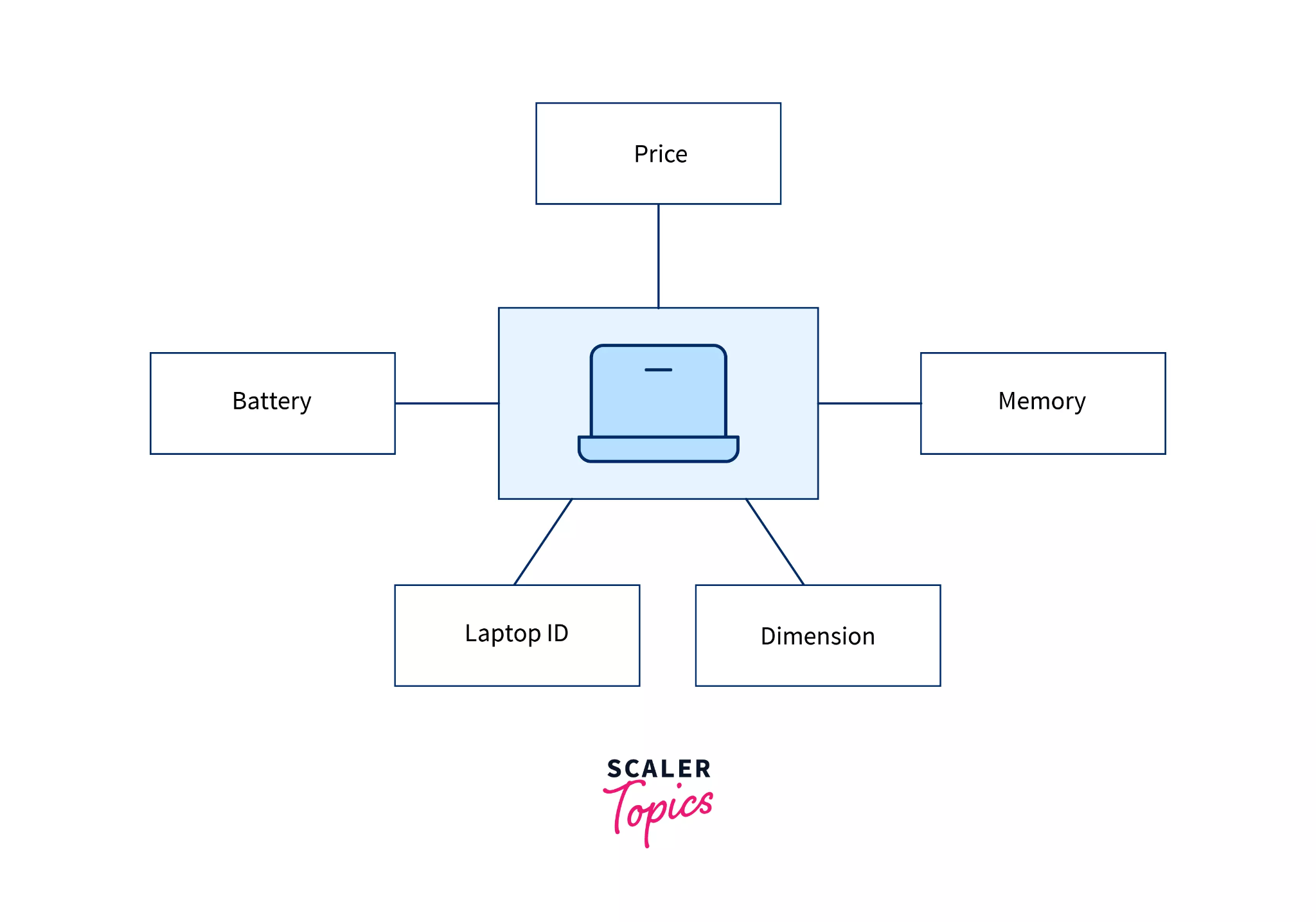
### Entities before Generalization

**Below will be the entities before generalization:**

**Mobile Entity: Mobile entities can have attributes like mobile\_ID, price, memory, battery, and dimension to which we can apply the generalization procedure.**

****

**Laptop Entity: Laptop entity can have attributes like laptop\_ID, price, memory, battery, and dimension to which we can apply the generalization procedure.**

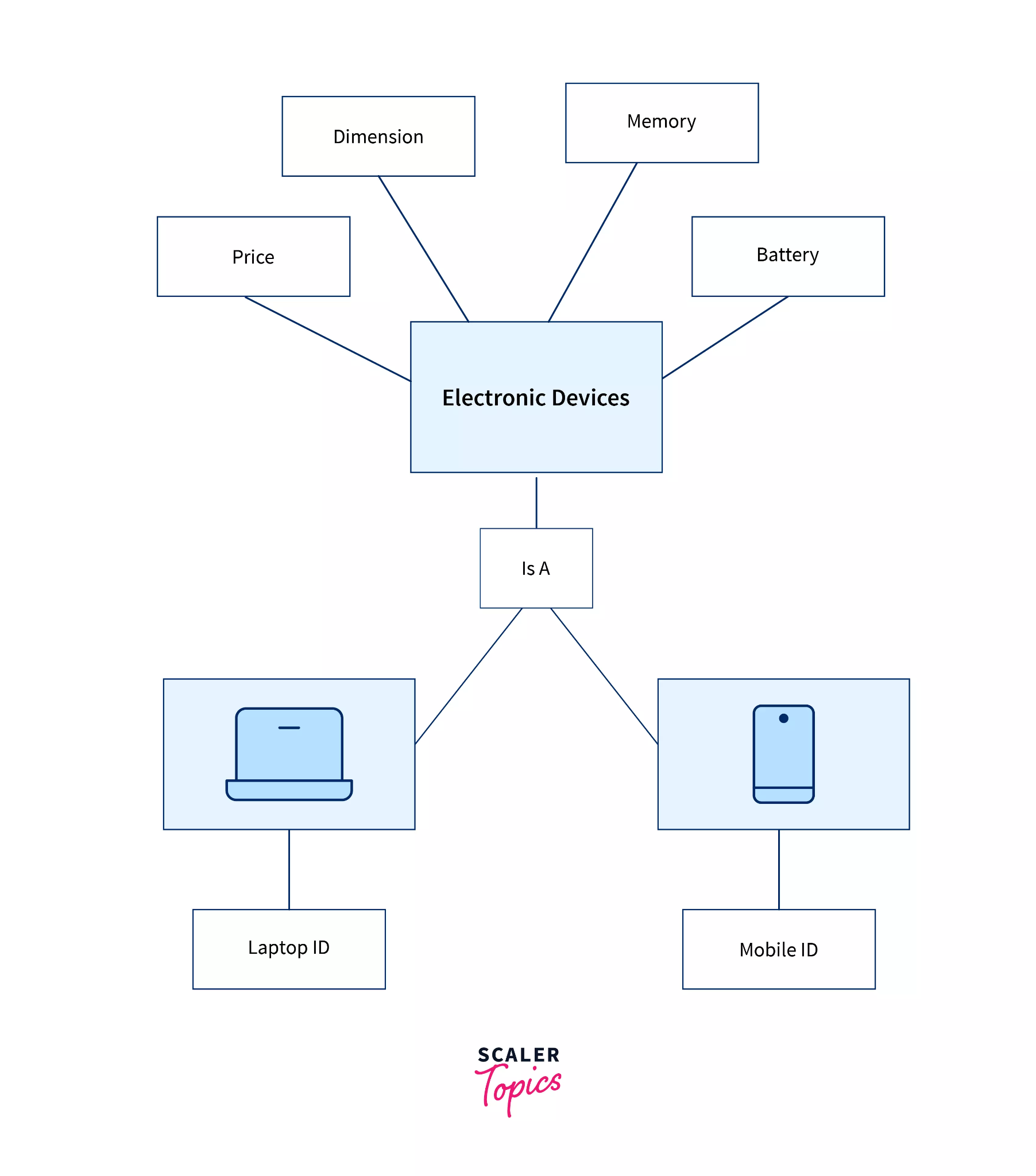
****

### ER Diagram after Generalization

**We can see that for laptops and mobiles some attributes like price, memory, battery, and dimension are the same so we can apply generalization and can group them into an entity called Electronic Devices.**

**The**

***Electronic* *Devices* entity will be having these common attributes and the laptop and mobile entity will be related to them using "Is A" relationship table, so using generalization we can represent ER diagram by:**

****

# Normalization

**A large database defined as a single relation may result in data duplication. This repetition of data 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.**
* **Wastage and poor utilization of disk space and resources.**
* **The likelihood of errors and inconsistencies increases.**

**So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.**

## 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.**

**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.**

# First Normal Form (1NF)

* **A relation will be 1NF if it contains an atomic value.**
* **It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.**
* **First normal form disallows the multi-valued attribute, composite attribute, and their combinations.**

**Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.**

**EMPLOYEE table:**

| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| --- | --- | --- | --- |
| **14** | **John** | **7272826385,**  **9064738238** | **UP** |
| **20** | **Harry** | **8574783832** | **Bihar** |
| **12** | **Sam** | **7390372389,**  **8589830302** | **Punjab** |

**The decomposition of the EMPLOYEE table into 1NF has been shown below:**

| **EMP\_ID** | **EMP\_NAME** | **EMP\_PHONE** | **EMP\_STATE** |
| --- | --- | --- | --- |
| **14** | **John** | **7272826385** | **UP** |
| **14** | **John** | **9064738238** | **UP** |
| **20** | **Harry** | **8574783832** | **Bihar** |
| **12** | **Sam** | **7390372389** | **Punjab** |
| **12** | **Sam** | **8589830302** | **Punjab** |

# Second Normal Form (2NF)

* **In the 2NF, relational must be in 1NF.**
* **In the second normal form, all non-key attributes are fully functional dependent on the primary key**

**Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.**

**TEACHER table**

| **TEACHER\_ID** | **SUBJECT** | **TEACHER\_AGE** |
| --- | --- | --- |
| **25** | **Chemistry** | **30** |
| **25** | **Biology** | **30** |
| **47** | **English** | **35** |
| **83** | **Math** | **38** |
| **83** | **Computer** | **38** |

**In the given table, non-prime attribute TEACHER\_AGE is dependent on TEACHER\_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.**

**To convert the given table into 2NF, we decompose it into two tables:**

**TEACHER\_DETAIL table:**

| **TEACHER\_ID** | **TEACHER\_AGE** |
| --- | --- |
| **25** | **30** |
| **47** | **35** |
| **83** | **38** |

**TEACHER\_SUBJECT table:**

| **TEACHER\_ID** | **SUBJECT** |
| --- | --- |
| **25** | **Chemistry** |
| **25** | **Biology** |
| **47** | **English** |
| **83** | **Math** |
| **83** | **Computer** |

# Third Normal Form (3NF)

* **A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.**
* **3NF is used to reduce the data duplication. It is also used to achieve the data integrity.**
* **If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.**

**A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.**

1. **X is a super key.**
2. **Y is a prime attribute, i.e., each element of Y is part of some candidate key.**
3. **Example:  
   EMPLOYEE\_DETAIL table:**

| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| --- | --- | --- | --- | --- |
| **222** | **Harry** | **201010** | **UP** | **Noida** |
| **333** | **Stephan** | **02228** | **US** | **Boston** |
| **444** | **Lan** | **60007** | **US** | **Chicago** |
| **555** | **Katharine** | **06389** | **UK** | **Norwich** |
| **666** | **John** | **462007** | **MP** | **Bhopal** |

1. **Super key in the table above:**
   1. **{EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on**
2. **Candidate key: {EMP\_ID}  
   Non-prime attributes: In the given table, all attributes except EMP\_ID are non-prime.  
   Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.  
   That's why we need to move the EMP\_CITY and EMP\_STATE to the new <EMPLOYEE\_ZIP> table, with EMP\_ZIP as a Primary key.  
   EMPLOYEE table:**

| **EMP\_ID** | **EMP\_NAME** | **EMP\_ZIP** |
| --- | --- | --- |
| **222** | **Harry** | **201010** |
| **333** | **Stephan** | **02228** |
| **444** | **Lan** | **60007** |
| **555** | **Katharine** | **06389** |
| **666** | **John** | **462007** |

1. **EMPLOYEE\_ZIP table:**

| **EMP\_ZIP** | **EMP\_STATE** | **EMP\_CITY** |
| --- | --- | --- |
| **201010** | **UP** | **Noida** |
| **02228** | **US** | **Boston** |
| **60007** | **US** | **Chicago** |
| **06389** | **UK** | **Norwich** |
| **462007** | **MP** | **Bhopal** |

# 

# 

# 

# 

# 

# Boyce Codd normal form (BCNF)

* **BCNF is the advanced 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.**

## Why is RDBMS Required?

**RDBMS on the other hand is a type of DBMS, as the name suggests it deals with relations as well as various key constraints. So here we have tables which are called schema and we have rows which are called tuples. It also aids in the reduction of data redundancy and the preservation of database integrity.**

**Relational Database Management System is an advanced version of a DBMS.**

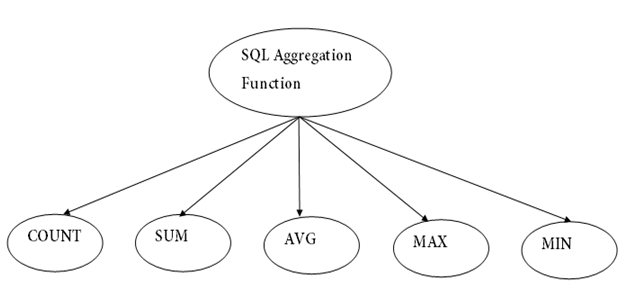
# Difference Between DBMS and RDBMS

| **DBMS** | **RDBMS** |
| --- | --- |
| [**DBMS**](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/) **stores data as file.** | [**RDBMS**](https://www.geeksforgeeks.org/rdbms-architecture/) **stores data in tabular form.** |
| **Data elements need to access individually.** | **Multiple data elements can be accessed at the same time.** |
| **No relationship between data.** | **Data is stored in the form of tables which are related to each other.** |
| **Normalization is not present.** | **Normalization is present.** |
| **DBMS does not support distributed database.** | **RDBMS supports distributed database.** |
| **It stores data in either a navigational or hierarchical form.** | **It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.** |
| **It deals with small quantity of data.** | **It deals with large amount of data.** |
| **Data redundancy is common in this model.** | **Keys and indexes do not allow Data redundancy.** |
| **It is used for small organization and deal with small data.** | **It is used to handle large amount of data.** |
| **Not all Codd rules are satisfied.** | **All 12 Codd rules are satisfied.** |
| **Security is less** | **More security measures provided.** |
| **It supports single user.** | **It supports multiple users.** |
| **Data fetching is slower for the large amount of data.** | **Data fetching is fast because of relational approach.** |
| **The data in a DBMS is subject to low security levels with regards to data manipulation.** | **There exists multiple levels of data security in a RDBMS.** |
| **Low software and hardware necessities.** | **Higher software and hardware necessities.** |
| **Examples:** [**XML**](https://www.geeksforgeeks.org/xml-basics/)**, Window Registry, Forxpro, dbaseIIIplus etc.** | **Examples:** [**MySQL**](https://www.geeksforgeeks.org/architecture-of-mysql/)**,** [**PostgreSQL**](https://www.geeksforgeeks.org/what-is-postgresql-introduction/)**,** [**SQL**](https://www.geeksforgeeks.org/what-is-sql/) **Server, Oracle, Microsoft Access etc.** |

# Aggregate Functions

* **SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.**
* **It is also used to summarize the data.**

## Types of SQL Aggregation Function

****

**Now let us understand each Aggregate function with a example:**

**Id Name Salary**

**-----------------------**

**1 A 80**

**2 B 40**

**3 C 60**

**4 D 70**

**5 E 60**

**6 F Null**

**Count():**

***Count(\*):* Returns total number of records .i.e 6.  
*Count(salary):* Return number of Non Null values over the column salary. i.e 5.  
*Count(Distinct Salary):*  Return number of distinct Non Null values over the column salary .i.e 4**

**Sum():**

***sum(salary):*  Sum all Non Null values of Column salary i.e., 310  
*sum(Distinct salary):* Sum of all distinct Non-Null values i.e., 250.**

**Avg():**

***Avg(salary)* = Sum(salary) / count(salary) = 310/5  
*Avg(Distinct salary)* = sum(Distinct salary) / Count(Distinct Salary) = 250/4**

**Min():**

***Min(salary):* Minimum value in the salary column except NULL i.e., 40.  
*Max(salary):* Maximum value in the salary i.e., 80.**

# Functional Dependency

**The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.**

1. **X → Y**

**The left side of FD is known as a determinant, the right side of the production is known as a dependent.**

**For example:**

**Assume we have an employee table with attributes: Emp\_Id, Emp\_Name, Emp\_Address.**

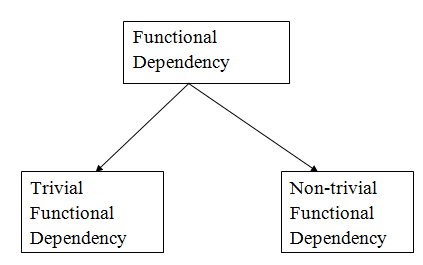
**Here Emp\_Id attribute can uniquely identify the Emp\_Name attribute of employee table because if we know the Emp\_Id, we can tell that employee name associated with it.**

**Functional dependency can be written as:**

1. **Emp\_Id → Emp\_Name**

**We can say that Emp\_Name is functionally dependent on Emp\_Id.**

## Types of Functional dependency

****

### 1. Trivial functional dependency

* **A → B has trivial functional dependency if B is a subset of A.**
* **The following dependencies are also trivial like: A → A, B → B**

**Example:**

1. **Consider a table with two columns Employee\_Id and Employee\_Name.**
2. **{Employee\_id, Employee\_Name} → Employee\_Id is a trivial functional dependency as**
3. **Employee\_Id is a subset of {Employee\_Id, Employee\_Name}.**
4. **Also, Employee\_Id → Employee\_Id and Employee\_Name → Employee\_Name are trivial dependencies too.**

### 2. Non-trivial functional dependency

* **A → B has a non-trivial functional dependency if B is not a subset of A.**
* **When A intersection B is NULL, then A → B is called as complete non-trivial.**

**Example:**

1. **ID → Name,**
2. **Name → DOB**