**Question No-01: What is Data?**

In general, **data** is any set of [characters](https://www.computerhope.com/jargon/c/charact.htm) that is gathered and translated for some purpose, usually analysis. If data is not put into context, it doesn't do anything to a human or computer.

There are multiple types of data. Some of the more common types of data include the following:

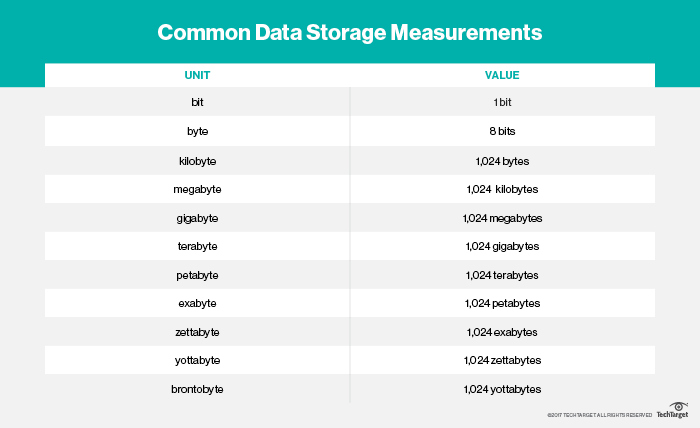
* Single character
* [Boolean](https://www.computerhope.com/jargon/b/boolean.htm) (true or false)
* Text ([string](https://www.computerhope.com/jargon/s/string.htm))
* Number ([integer](https://www.computerhope.com/jargon/i/integer.htm) or [floating-point](https://www.computerhope.com/jargon/f/floapoin.htm))
* [Picture](https://www.computerhope.com/jargon/p/picture.htm)
* [Sound](https://www.computerhope.com/jargon/s/sound.htm)
* [Video](https://www.computerhope.com/jargon/v/video.htm)

In a computer's storage, digital data is a sequence of [bits](https://www.computerhope.com/jargon/b/bit.htm) ([binary digits](https://www.computerhope.com/jargon/b/binary.htm)) that have the value one or zero. Data is processed by the [CPU](https://www.computerhope.com/jargon/c/cpu.htm), which uses [logical](https://www.computerhope.com/jargon/l/logic.htm) operations to produce new data ([output](https://www.computerhope.com/jargon/o/output.htm)) from source data ([input](https://www.computerhope.com/jargon/i/input.htm)).

**How data is stored:**

Computers represent data, including video, images, sounds and text, as binary values using patterns of just two numbers: 1 and 0. A [bit](https://www.techtarget.com/whatis/definition/bit-binary-digit) is the smallest unit of data, and represents just a single value. A byte is eight binary digits long. Storage and memory is measured in [megabytes](https://www.techtarget.com/searchstorage/definition/megabyte) and [gigabytes](https://www.techtarget.com/searchstorage/definition/gigabyte).

The units of data measurement continue to grow as the amount of data collected and stored grows. The relatively new term "[brontobyte](https://www.techtarget.com/searchstorage/definition/brontobyte)," for example, is [data storage](https://www.techtarget.com/searchstorage/definition/storage) that is equal to 10 to the 27th power of [bytes](https://www.techtarget.com/searchstorage/definition/byte).



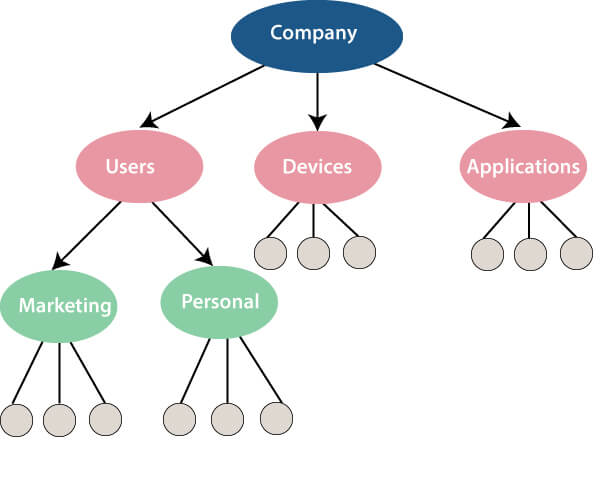
**Question – 02:**

**What is Database?**

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.

Let us discuss a database example: An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc.

Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.



## Types of Databases

Here are some popular types of databases.

* Distributed Database
* Relational Database
* Object-Oriented Database
* Centralized Database
* Open-Source Database
* Cloud Database
* Data Warehouse
* NoSql Database
* Graph Database
* OLTP Database
* Personal Database
* Multimodal Database
* Document/JSON Database
* Hierarchical Database

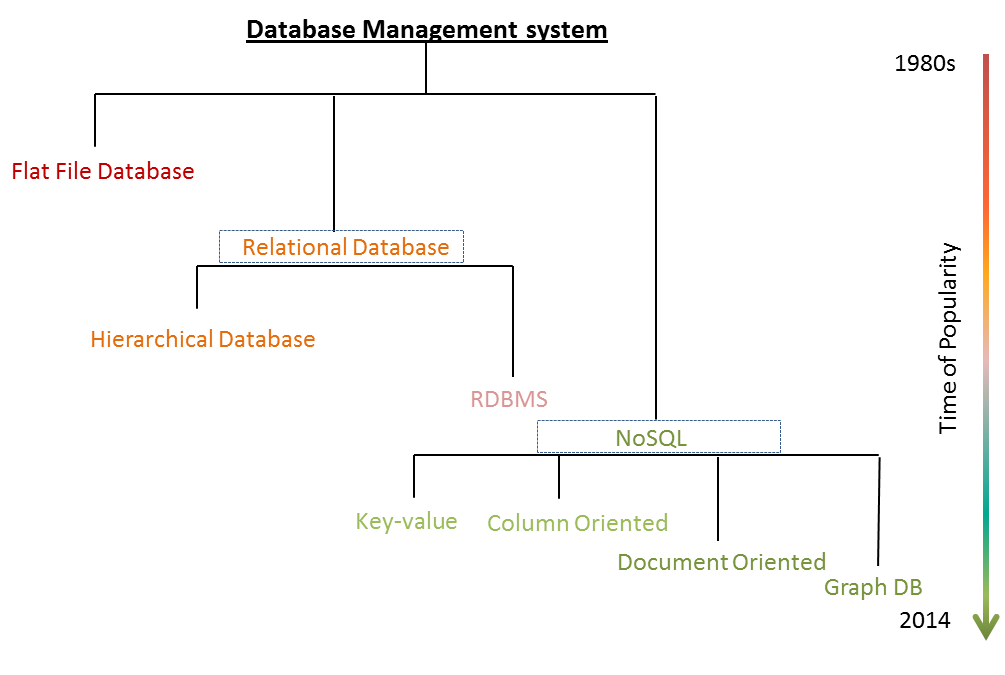
**Question-03:**

**Evolution of Database:**

A database management system is a set of programs that allows users to create and maintain a data base and interact with various users, including data base administrators, online users, application programmers, and naive users, as well as other users.

Hence we can say that basically database is a compilation of database files and each database file is further a collection of records.

**Database models:**

****

**1) Flat files (1960s – 1980s)**

**Implementation of a flat file database**

Flat file database is implemented in:

* Berkeley DB
* SQLite
* Mimesis
* TheIntegrationEngineer etc.

**2) Hierarchical database (1970s – 1990s)**

**3)** **Network database (1970s – 1990s)**

**Implementation of network database**

Network database is implemented in:

* Digital Equipment Corporation DBMS-10
* Digital Equipment Corporation DBMS-20
* RDM Embedded
* Turbo IMAGE
* Univac DMS-1100 etc.

**4)** **Relational database (1980s – present)**

Implementation of Relational Database:

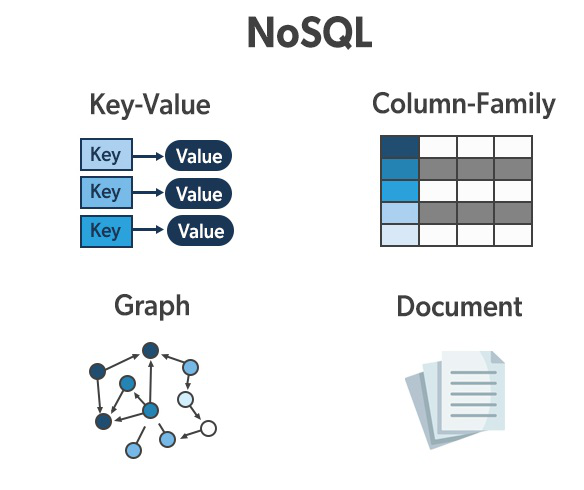
* Oracle
* Microsoft
* IBM
* My SQL
* PostgreSQL
* SQLite

**5)** **Object-oriented database (1990s – present)**

**Question No-04:**

**NoSQL Database:**

**NoSQL database** technology stores information in JSON documents instead of columns and rows used by relational databases. To be clear, NoSQL stands for “not only SQL” rather than “no SQL” at all. This means a NoSQL JSON database can store and retrieve data using literally “no SQL.” Or you can combine the flexibility of JSON with the power of SQL for the best of both worlds. Consequently, NoSQL databases are built to be flexible, scalable, and capable of rapidly responding to the data management demands of modern businesses. The following defines the four most-popular types of NoSQL database:



* **Document databases** are primarily built for storing information as documents, including, but not limited to, JSON documents. These systems can also be used for storing XML documents, for example.
* **Key-value stores** group associated data in collections with records that are identified with unique keys for easy retrieval. Key-value stores have just enough structure to mirror the value of relational databases while still preserving the benefits of NoSQL.
* **Wide-column databases** use the tabular format of relational databases yet allow a wide variance in how data is named and formatted in each row, even in the same table. Like key-value stores, wide-column databases have some basic structure while also preserving a lot of flexibility
* **Graph databases** use graph structures to define the relationships between stored data points. Graph databases are useful for identifying patterns in unstructured and semi-structured information.

## Why use NoSQL?

 Customer experience has quickly become the most important competitive differentiator and ushered the business world into an era of monumental change. As part of this revolution, enterprises are interacting digitally – not only with their customers, but also with their employees, partners, vendors, and even their products – at an unprecedented scale. This interaction is powered by the internet and other 21st century technologies – and at the heart of the revolution are a company’s cloud, mobile, social media, big data, and IoT applications.

How are these applications different from legacy enterprise applications like ERP, HR, and financial accounting? Today’s web, mobile, and IoT applications share one or more (if not all) of the following characteristics. They need to:

 Support large numbers of concurrent users (tens of thousands, perhaps millions)

* Deliver highly responsive experiences to a globally distributed base of users
* Be always available – no downtime
* Handle semi- and unstructured data
* Rapidly adapt to changing requirements with frequent updates and new features

**Question No- 05:**

**Advantage of NoSQL:**

**What are the advantages of NoSQL?**

Like every other technology, NoSQL databases also offer some benefits and suffer from some limitations too.

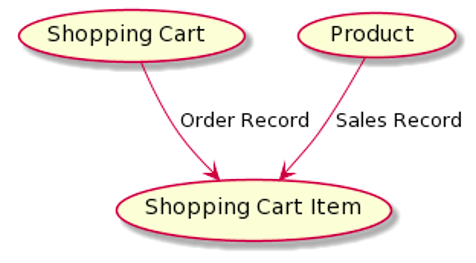
In an era where relational databases are mainly used for data storage and retrieval, modern web technologies posed a major challenge in the form of unstructured data, high scale data, enormous concurrency etc.

Relational databases struggled especially to represent highly unstructured data and high scalability and thus came into being the NoSQL databases.

**Major advantages of NoSQL databases include:**

**(i) Flexible Data Model:**

NoSQL databases are highly flexible as they can store and combine any type of data, both structured and unstructured, unlike relational databases that can store data in a structured way only.



**(ii) Evolving Data Model :**

NoSQL databases allow you to dynamically update the schema to evolve with changing requirements while ensuring that it would cause no interruption or downtime to your application.

**(iii) Elastic Scalability:**

NoSQL databases can scale to accommodate any type of data growth while maintaining low cost.

**(iv) High Performance:**

NoSQL databases are built for great performance, measured in terms of both throughput (it is a measure of overall performance) and latency (it is the delay between request and actual response).

**(v) Open-source:**

NoSQL databases don’t require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.

**Question No- 06:**

**Disadvantage of NoSQL:**

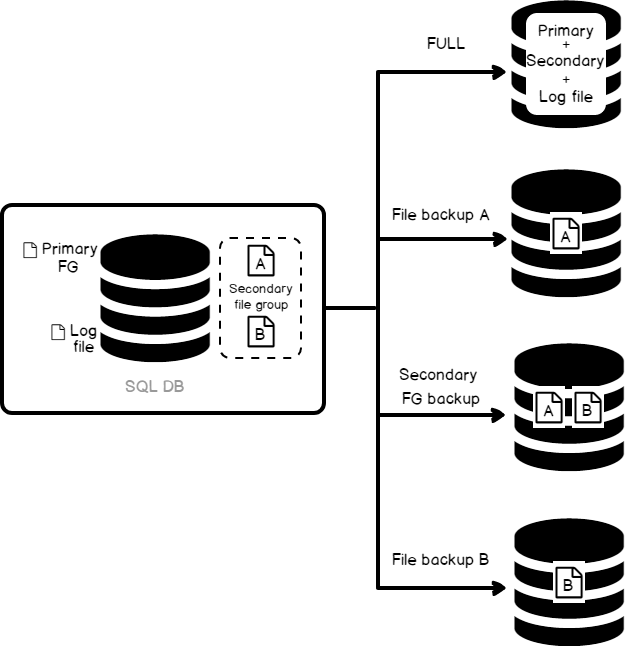
**Major disadvantages of NoSQL databases are:**

**(i) Lack of Standardization:**

There is no standard that defines rules and roles of NoSQL databases. The design and query languages of NoSQL databases vary widely between different NoSQL products – much more widely than they do among traditional SQL databases.

**(ii) Backup of Database:**

Backups are a drawback in NoSQL databases. Though some NoSQL databases like MongoDB provide some tools for backup, these tools are not mature enough to ensure proper complete data backup solution.



**(iii) Consistency:**

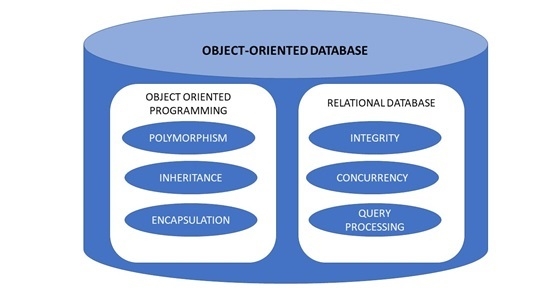
NoSQL puts a scalability and performance first but when it comes to a consistency of the data NoSQL doesn’t take much consideration so it makes it little insecure as compared to the relational database e.g., in NoSQL databases if you enter same set of data again, it will take it without issuing any error whereas relational databases ensure that no duplicate rows get entry in databases.

Now that you have an idea of what NoSQL databases are.

**Question No – 07:**

**The Object-Oriented Database:**

**Object-oriented databases** are a type of database management system. Different database management systems provide additional functionalities. Object-oriented databases add the [database](https://phoenixnap.com/kb/what-is-a-database) functionality to object programming languages, creating more manageable code bases.



## Object-Oriented Programming Concepts

Object-oriented databases closely relate to object-oriented programming concepts. The four main ideas of object-oriented programming are:

* **Polymorphism**
* **Inheritance**
* **Encapsulation**
* **Abstraction**

These four attributes describe the critical characteristics of object-oriented management systems.

### Polymorphism

**Polymorphism** is the capability of an object to take multiple forms. This ability allows the same program code to work with different data types. Both a car and a bike are able to break, but the mechanism is different. In this example, the action break is a polymorphism. The defined action is **polymorphic** — the result changes depending on which vehicle performs.

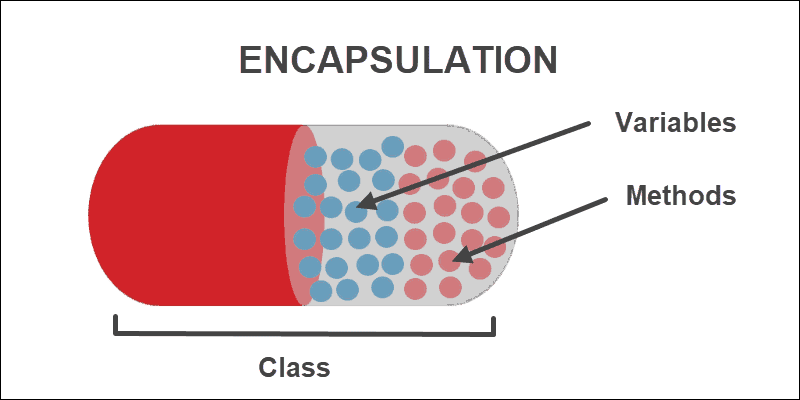
### Inheritance

**Inheritance** creates a hierarchical relationship between related classes while making parts of code reusable. Defining new types inherits all the existing class fields and methods plus further extends them. The existing class is the **parent** class, while the **child** class extends the parent.

For example, a parent class called Vehicle will have child classes Car and Bike. Both child classes **inherit** information from the parent class and **extend** the parent class with new information depending on the vehicle type.

### Encapsulation

**Encapsulation** is the ability to group data and mechanisms into a single object to provide access protection. Through this process, pieces of information and details of how an object works are **hidden**, resulting in data and function security. Classes interact with each other through methods without the need to know how particular methods work.



**Question No – 08:**

**Graph Database:**

Graph databases are purpose-built to store and navigate relationships. Relationships are first-class citizens in graph databases, and most of the value of graph databases is derived from these relationships. Graph databases use nodes to store data entities, and edges to store relationships between entities. An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have.

A graph in a graph database can be traversed along specific edge types or across the entire graph. In graph databases, traversing the joins or relationships is very fast because the relationships between nodes are not calculated at query times but are persisted in the database. Graph databases have advantages for use cases such as social networking, recommendation engines, and fraud detection, when you need to create relationships between data and quickly query these relationships.

The following graph shows an example of a social network graph. Given the people (nodes) and their relationships (edges), you can find out who the "friends of friends" of a particular person are—for example, the friends of Howard's friends.

### 5.png

### The structure of a graph database

Traditionally classified as a type of [NoSQL database](https://preview.techtarget.com:8080/searchDataManagement/definition/NoSQL-Not-Only-SQL), graph databases are sometimes referred to as triple stores. That's because this type of database uses a special index that stores information about nodes, edges and the relationship between them in groups of three.

A triple, which may also be referred to as an assertion, has three main fields: a subject, a predicate and an object. Each subject, predicate or object is represented by a unique resource identifier

**Question No – 09:**

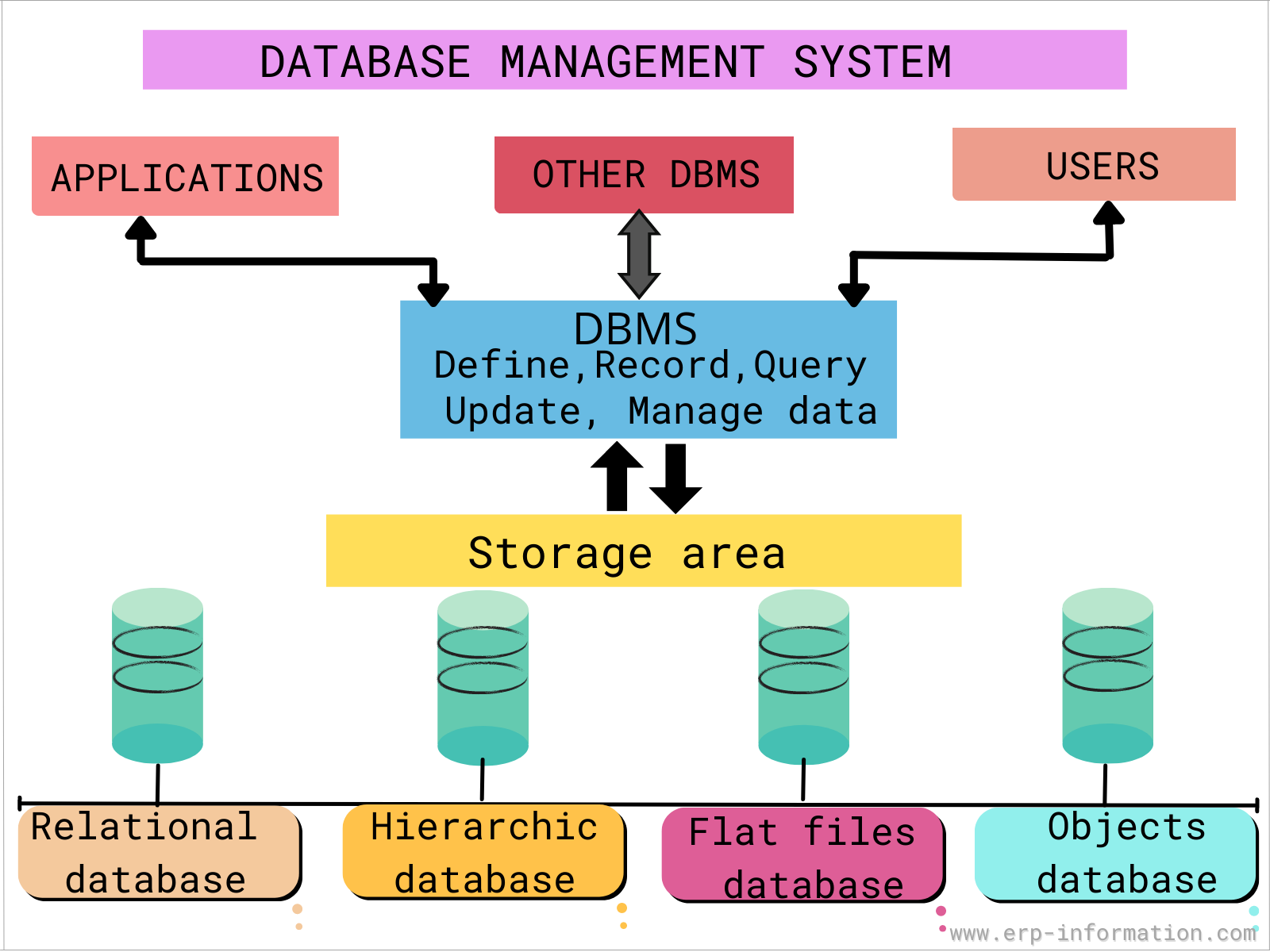
**DBMS (Data Base Management System):**

Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.

DBMS manage the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users and other programs. This helps provide data security, data integrity, concurrency, and uniform data administration procedures.

DBMS optimizes the organization of data by following a database schema design technique called normalization, which splits a large table into smaller tables when any of its attributes have redundancy in values. DBMS offer many benefits over traditional file systems, including flexibility and a more complex backup system.

Database management systems can be classified based on a variety of criteria such as the data model, the database distribution, or user numbers. The most widely used types of DBMS software are relational, distributed, hierarchical, object-oriented, and network.



**Distributed database management system**

A distributed DBMS is a set of logically interrelated databases distributed over a network that is managed by a centralized database application. This type of DBMS synchronizes data periodically and ensures that any change to data is universally updated in the database.

**Hierarchical database management system**

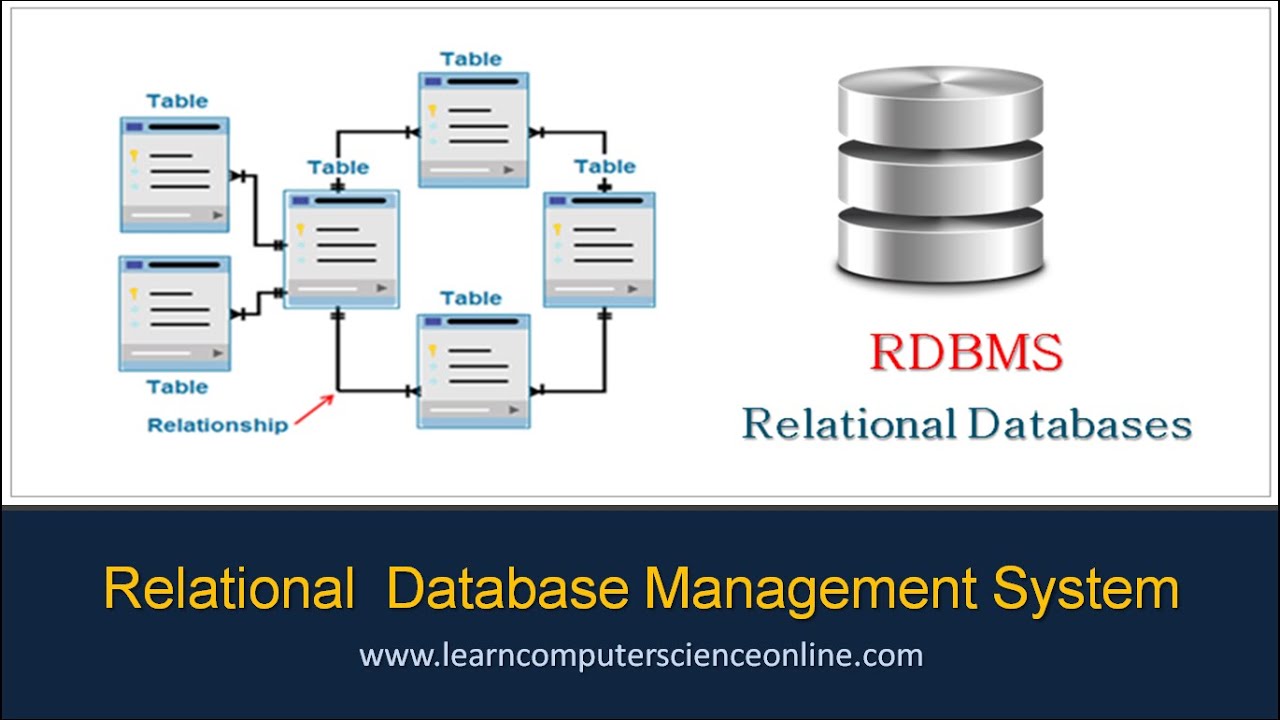
Hierarchical databases organize model data in a tree-like structure. Data storage is either a top-down or bottom-up format and is represented using a parent-child relationship.

**Network database management system**

The network database model addresses the need for more complex relationships by allowing each child to have multiple parents. Entities are organized in a graph that can be accessed through several paths.

**Relational database management system**

Relational database management systems (RDBMS) are the most popular data model because of its user-friendly interface. It is based on normalizing data in the rows and columns of the tables. This is a viable option when you need a data storage system that is scalable, flexible, and able to manage lots of information.



**Object-oriented database management system**

Object-oriented models store data in objects instead of rows and columns. It is based on object-oriented programming (OOP) that allows objects to have members such as fields, properties, and methods.

## Examples of DBMS

There is a wide range of database software solutions, including both enterprise and open source solutions, available for database management.

Here are some of the most popular database management systems:

### Oracle

Oracle Database is a commercial relational database management system. It utilizes enterprise-scale database technology with a robust set of features right out of the box. It can be stored in the cloud or on-premises.

### MySQL

MySQL is a relational database management system that is commonly used with open-source content management systems and large platforms like Facebook, Twitter, and Youtube.

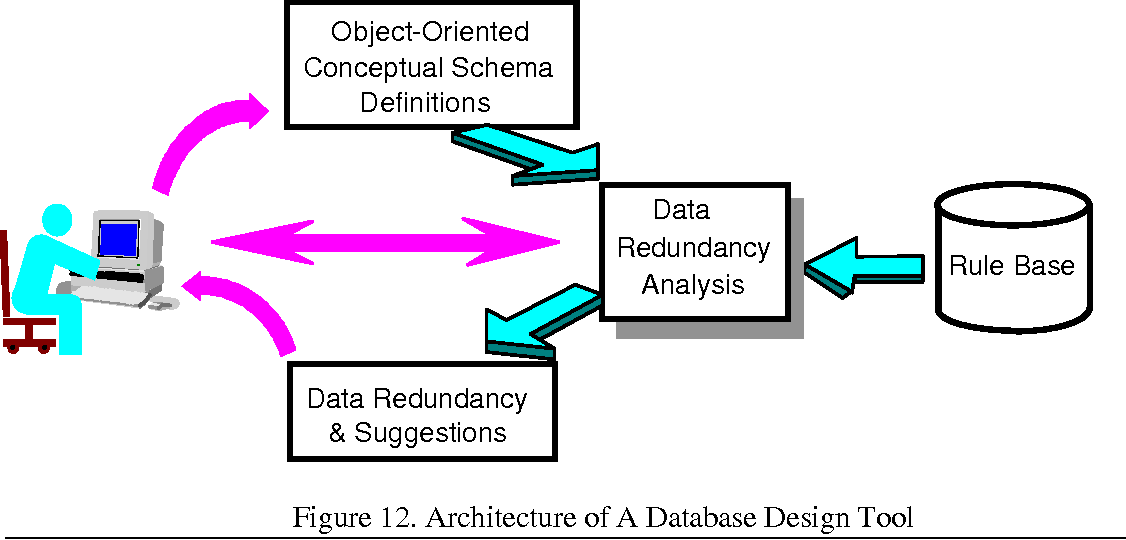
### SQL Server

Developed by Microsoft, SQL Server is a relational database management system built on top of structured query language (SQL), a standardized programming language that allows database administrators to manage databases and query data.

**Question No- 10:**

**Advantage of DBMS:**

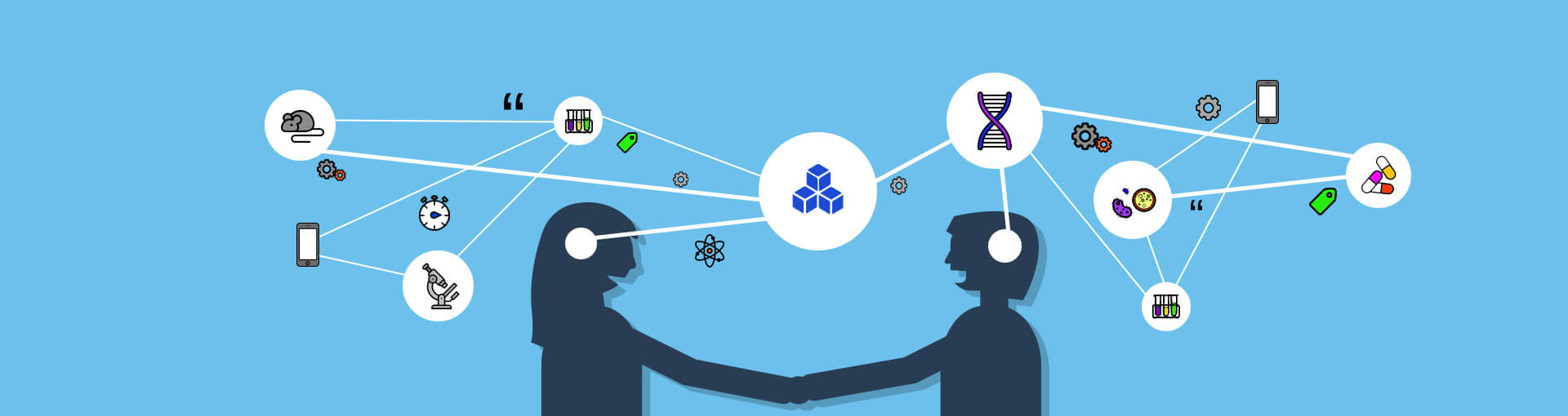
## Reducing Data Redundancy



The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes multiple copies of the same file which lead to data redundancy.

This is prevented in a database as there is a single database and any change in it is reflected immediately. Because of this, there is no chance of encountering duplicate data.

## Sharing of Data



In a database, the users of the database can share the data among themselves. There are various levels of authorization to access the data, and consequently the data can only be shared based on the correct authorization protocols being followed.

Many remote users can also access the database simultaneously and share the data between themselves.

## Data Integrity

Data integrity means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All of these databases contain data that is visible to multiple users. So it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.

## Data Security

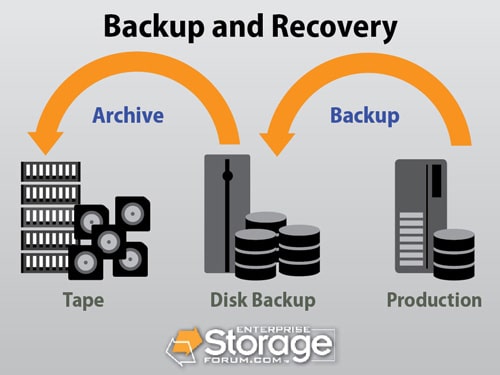


Data Security is vital concept in a database. Only authorized users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorized users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

## Privacy

The privacy rule in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he is allowed to. For example - In social networking sites, access constraints are different for different accounts a user may want to access.

## Backup and Recovery



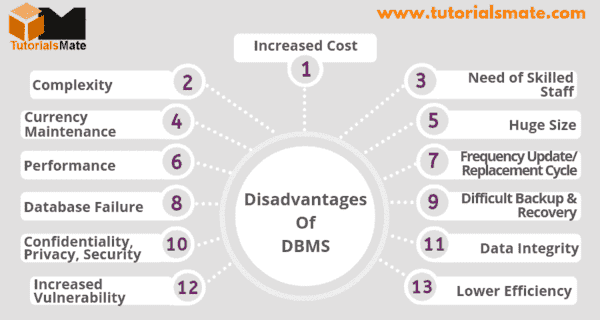
Database Management System automatically takes care of backup and recovery. The users don't need to backup data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.

## Data Consistency

Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

**Question No- 11:**

**Disadvantage of DBMS:**



With the vast list of advantages, there are some following disadvantages or limitations of the database management system.

**1. High Cost**

The high cost of software and hardware is the main disadvantage of the database management system.

Database users require a high-speed processor and huge memory size to use the database on the DBMS. Sometimes, users require costly machines for maintaining databases.

Organizations need a trained and highly paid technical database administrator for using and maintaining the large database systems.

**2. Huge Size**

The size of the database is not big at the initial state, but when the user stores a large amount of data, then it creates many problems. Due to the huge data, database systems do not provide good results and do not run efficiently. That's why the size is another limitation of the database systems.

**3. Database Failure**

In the database systems, all the data or information of an organization is stored in one centralized database. If the database of that organization fails, then the data is lost, and the organization will collapse. So, database failure is a big problem with the database management system.

**4. Complexity**

Database management system (DBMS) is so complex for non-technical users. So, it isn't easy to manage and maintain database systems. Therefore, training for the designers, users, and administrators is necessary to efficiently run the database systems.

**5. Increased Staff Cost**

DBMS requires an educated and skilled staff for managing and maintaining the databases. So, we need to spend a lot of money to get this level of trained and experienced staff.

**6. Requirement of Technical Staff**

A non-technical people can't understand the complexity of the database. So, the technical staff is required for maintaining and handling the database management system.

**7. Cost of Data Conversion**

It is one of the big disadvantages of the database management system because the cost of data conversion is very high. There is a requirement for trained, skilled, and experienced database administrators for converting the data smoothly.

**8. Performance**

Performance is another big disadvantage of database systems because the speed of the database systems for small firms and organizations is very slow. Hence, the performance of the database systems in small organizations is poor.

**Question No – 12:**

**RDBMS (Relational Database Management System):**



RDBMS stands for Relational Database Management System. It is an information management system that is oriented on a data model. Here all the information is properly stored as tables. RDBMS Example systems are SQL Server, Oracle, MySQL, MariaDB, and SQLite.

### ****Basic Features of RDBMS:****

* Offers information to be saved in the tables
* Numerous users can access it together which is managed by a single user
* Virtual tables are available for storing the insightful data
* In order to exclusively find out the rows, the primary key is used
* The data are always saved in rows and columns
* To retrieve the information the indexes are used
* Columns are being shared between tables using the keys

## ****RDBMS CONCEPTS****

Relation in the RDBMS is the deposit of tuples having similar characteristics. A relation in RDBMS means the set of the entities and data contained in them. The entities are different instances and their relation is organized in different rows and columns in the data table.  The related information is of identical domains and constraints. By deleting old data and by inserting new data, relations are altered in the database model.

### ****Domain in RDBMS****

The domain describes probable terms collected jointly that always have the same characteristics and also constraints. A domain can be involved in a feature but only if the feature is a factor of a particular set.

### ****RDBMS Database****

### 6.jpg

The association between different databases is called the relational database. Here all the data are related in terms of the tables they are stored in. The users will be able to access them. RDBMS database is different from that of [DBMS](https://intellipaat.com/blog/what-is-dbms/).

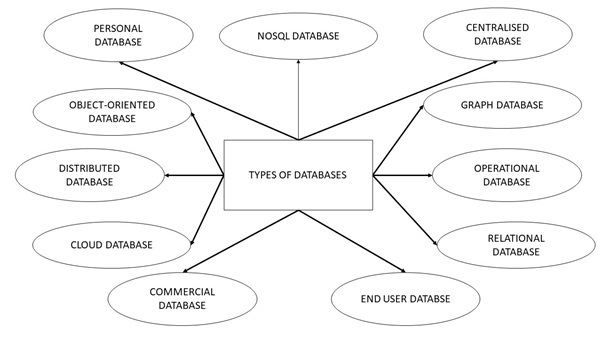
The RDBMS speaks about relations between different entities present in the database. Normal Database assists only the tables while RDBS along with the tables tells their connections too. Normal databases give consistent methods but databases of RDBMS do not offer any methodologies but instead give connections that connect one single object with another.

The stored database is called executable code. The [database](https://intellipaat.com/blog/what-is-database/) collects and saves data and that particular procedure is called the stored procedure. During this, the codes are also stored which are used for encapsulation, delete, etc. The programmers are able to add extensions of the syntaxes of [SQL](https://intellipaat.com/blog/tutorial/sql-tutorial/introduction-to-sql/) due to the applications of the APIs in the stored processes.

**Question No – 13**

**Types of Database:**

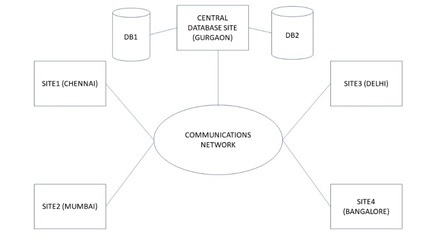
There are various types of databases used for storing different varieties of data:



## 1. Centralised Database

The information(data) is stored at a centralized location and the users from different locations can access this data. This type of database contains application procedures that help the users to access the data even from a remote location.

Various kinds of authentication procedures are applied for the verification and validation of end users, likewise, a registration number is provided by the application procedures which keeps a track and record of data usage. The local area office handles this thing.

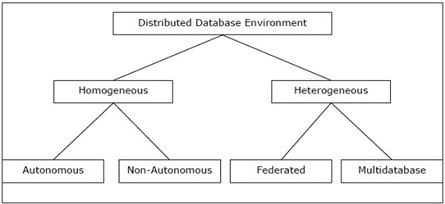


## 2.Distributed Database

Just opposite of the centralized database concept, the distributed database has contributions from the common database as well as the information captured by local computers also. The data is not at one place and is distributed at various sites of an organization. These sites are connected to each other with the help of communication links which helps them to access the distributed data easily.

You can imagine a distributed database as a one in which various portions of a database are stored in multiple different locations(physical) along with the application procedures which are replicated and distributed among various points in a network.

There are two kinds of distributed database, viz. homogenous and heterogeneous. The databases which have same underlying hardware and run over same operating systems and application procedures are known as homogeneous DDB, for eg. All physical locations in a DDB. Whereas, the operating systems, underlying hardware as well as application procedures can be different at various sites of a DDB which is known as heterogeneous DDB.



## 3.Personal Database

Data is collected and stored on personal computers which is small and easily manageable. The data is generally used by the same department of an organization and is accessed by a small group of people.

## 4.End User Database

The end user is usually not concerned about the transaction or operations done at various levels and is only aware of the product which may be a software or an application. Therefore, this is a shared database which is specifically designed for the end user, just like different levels’ managers. Summary of whole information is collected in this database.

## 5.Commercial Database

These are the paid versions of the huge databases designed uniquely for the users who want to access the information for help. These databases are subject specific, and one cannot afford to maintain such a huge information. Access to such databases is provided through commercial links.

## 6.NoSQL Database

These are used for large sets of distributed data. There are some big data performance issues which are effectively handled by relational databases, such kind of issues are easily managed by NoSQL databases. There are very efficient in analyzing large size unstructured data that may be stored at multiple virtual servers of the cloud.

## 7.Operational Database

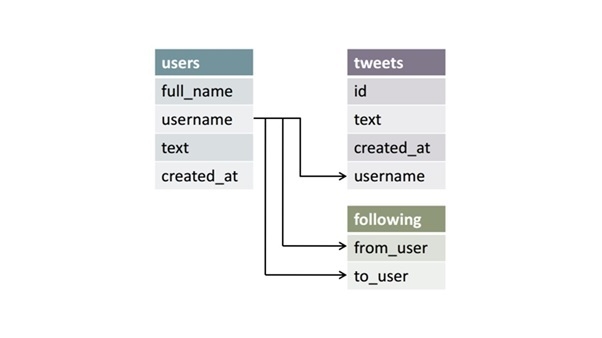
Information related to operations of an enterprise is stored inside this database. Functional lines like marketing, employee relations, customer service etc. require such kind of databases.



## 8.Relational Databases

These databases are categorized by a set of tables where data gets fit into a pre-defined category. The table consists of rows and columns where the column has an entry for data for a specific category and rows contains instance for that data defined according to the category. The Structured Query Language (SQL) is the standard user and application program interface for a relational database.

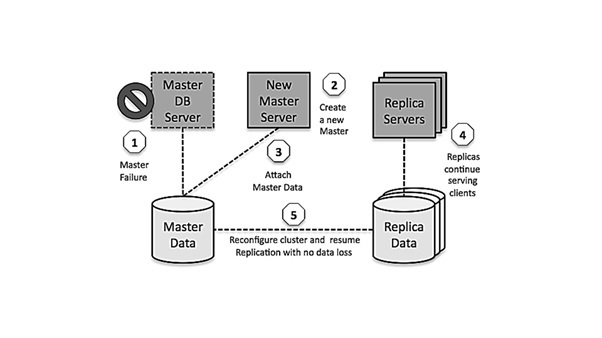
There are various simple operations that can be applied over the table which makes these databases easier to extend, join two databases with a common relation and modify all existing applications.



## 9.Cloud Databases

Now a day, data has been specifically getting stored over clouds also known as a virtual environment, either in a hybrid cloud, public or private cloud. A cloud database is a database that has been optimized or built for such a virtualized environment. There are various benefits of a cloud database, some of which are the ability to pay for storage capacity and bandwidth on a per-user basis, and they provide scalability on demand, along with high availability.

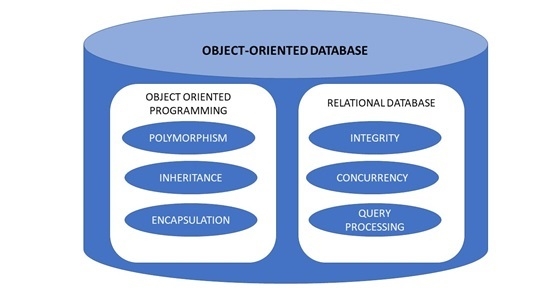
A cloud database also gives enterprises the opportunity to support business applications in a software-as-a-service deployment.



## 10.Object-Oriented Databases

An object-oriented database is a collection of object-oriented programming and relational database. There are various items which are created using object-oriented programming languages like C++, Java which can be stored in relational databases, but object-oriented databases are well-suited for those items.

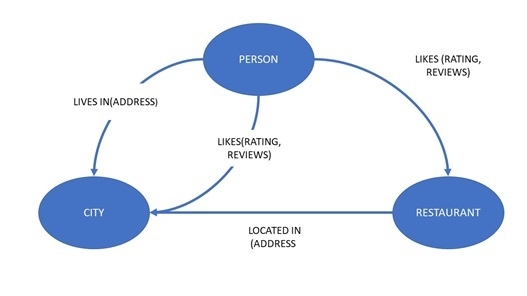
An object-oriented database is organized around objects rather than actions, and data rather than logic. For example, a multimedia record in a relational database can be a definable data object, as opposed to an alphanumeric value.



## 11.Graph Databases

The graph is a collection of nodes and edges where each node is used to represent an entity and each edge describes the relationship between entities. A graph-oriented database, or graph database, is a type of NoSQL database that uses graph theory to store, map and query relationships.

Graph databases are basically used for analyzing interconnections. For example, companies might use a graph database to mine data about customers from social media.



## Question No – 14:

**What is RDBMS:**

A relational [database](https://www.techtarget.com/searchdatamanagement/definition/database) management system (RDBMS) is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a [relational database](https://www.techtarget.com/searchdatamanagement/definition/relational-database). RDBMSes store data in the form of tables, with most commercial relational database management systems using [Structured Query Language](https://searchsqlserver.techtarget.com/definition/SQL) (SQL) to access the database. However, since SQL was invented after the initial development of the relational model, it is not necessary for RDBMS use.

The RDBMS is the most popular database system among organizations across the world. It provides a dependable method of storing and retrieving large amounts of data while offering a combination of system performance and ease of implementation.

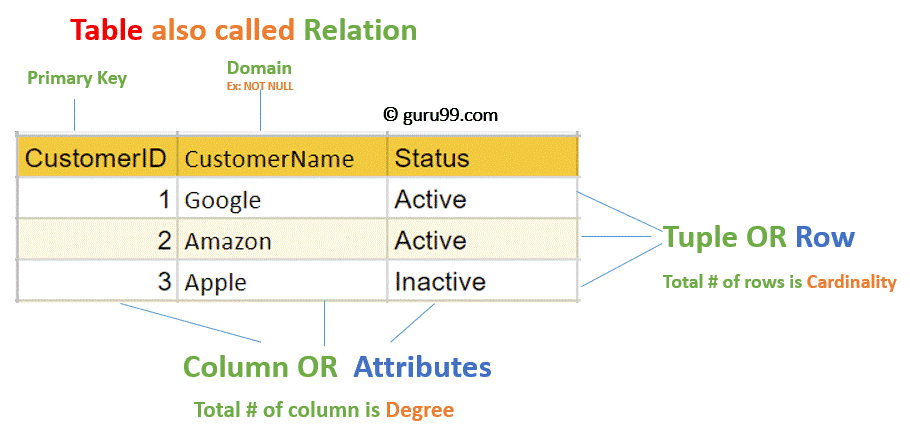
**Question No – 15:**

**How it Works:**

Data is stored in a relational database in the form of multiple tables. A key question here arises, how does a database structure work, and how is it implemented? Let’s understand this in detail.

A database structure works by arranging every table into rows (known as records/ tuples) and columns (known as fields/attributes). Tables, columns, and rows are the three major components of a relational database.

Here is an example of a type of business database to process customers’ orders. In this type of database, the first column contains customer ID, which is the primary key. The primary key is used to identify each unique record in a table. Each column (or attribute) stores bits of information, and the database assigns a unique customer ID to each row.



Relational database example. Source: [Guru99](https://www.guru99.com/relational-data-model-dbms.html)

The data structure used to implement a relational database is as follows:

As seen in the example above, the data structure of a single table. While most commonly, data is more complex and is stored in multiple tables. The tables are then connected based on pre-existing relationships.

The tables can be connected in different ways, such as:

* A record in one table could be related to another record in a different table (1:1 relationship)
* A record in one table may be related to several records in other tables (1:M relationship)
* Several table records could be linked to multiple records in a different table (M: N relationship)

However, before tables are created, a relational database management system must ensure that:

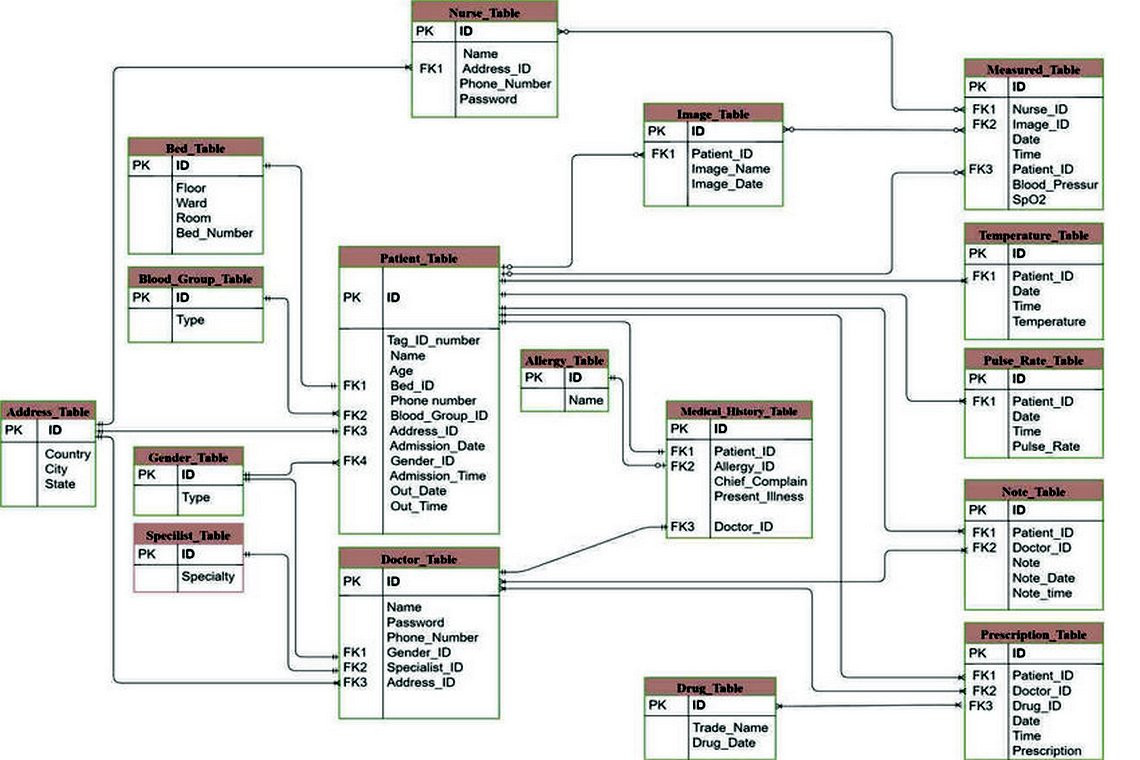
* Each table has a unique primary key, which has no null values.
* The foreign key, which is used to relate to 2 tables, is preserved in one table and refers to the primary key of another table.
* No column has a null value (empty cell).

An RDBMS usually comes with [data dictionaries](https://searchapparchitecture.techtarget.com/definition/data-dictionary) and collections of metadata, which are beneficial in data management. A data dictionary defines the data objects of every user within the database. As a result, it helps users identify all the objects that exist in the database and who can access them.

For instance, it’s humanly impossible to remember all the tables stored in a huge database. If you want to search for a specific table, such as Products, you can use a data dictionary to look up all the tables named Products. Then, you can identify your required one from the list of tables.

**Question No – 16:**

**Brief History of RDBMS**:



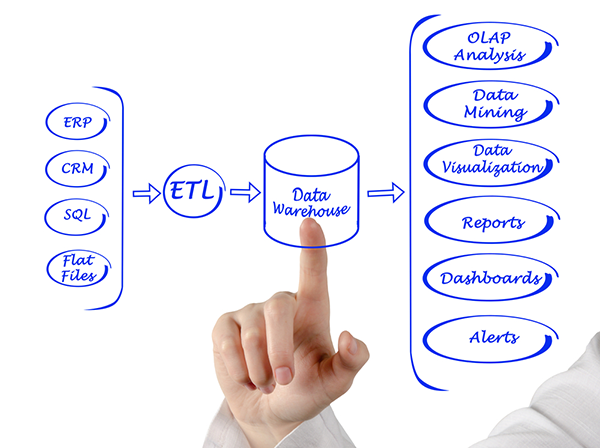
A brief history of relational databases helps us appreciate the importance of this technology and helps us understand Oracle Corporation’s decisions. The Oracle database is a huge product and contains mistakes. Some of those mistakes are unimportant historical curiosities, but others are huge pitfalls we need to avoid.

Relational databases are built on the relational model and relational algebra, first publically described by E.F. Codd in 1970. The relational model is built on set theory, a mathematical way of dealing with collections of objects. The relational model is discussed in more detail in the next section.

IBM started working on relational technology and products in 1968. Here’s the first history lesson: best is the enemy of good enough. Larry Ellison heard about IBM’s project, implemented it, and released the first commercially available SQL database in 1979. He has a huge presence in the database world and is still involved with many database decisions to this day.

Oracle Corporation has certainly used its first-mover advantage. The Oracle database has been the most popular database product for a long time. There are current trends away from Oracle and SQL, but we shouldn’t overlook how incredibly popular they are. The database scores high on almost any database popularity metric.

Oracle’s age explains many of its unexpected behaviors. The following list contains the features that are most likely to confuse database developers who are new to Oracle.

* **(+)**: Oracle originally used syntax like (+) instead of keywords like LEFT JOIN.
* **Date**: An Oracle date contains a time and should be called a DATETIME. Date formatting was awkward before the introduction of ANSI date literals.
* **Empty string**: Oracle treats an empty string as null, instead of a distinct value as most programmers expect. (Although I would argue that Oracle is not necessarily wrong here. We don’t have zero-length dates, or zero-length numbers; why should we have zero-length strings?)
* **30-byte name limit**: The SQL and PL/SQL languages have an English-like syntax, but we’ll quickly hit the 30-byte limit if we use regular words for names. Good variable names are important to make our programs more readable. Luckily this problem is fixed in version 12.2, which allows 128 bytes.
* **SQL\*Plus quirks**: SQL\*Plus is a great tool for some tasks, but it’s really showing its age in many ways.
* 

 In Oracle’s defense, those mistakes were made before any standards existed. On the other hand, Oracle doesn’t always make a good effort to comply with standards. For example, Oracle used to claim “partial” compliance for allowing long names. While 30 bytes is “part” of a bigger number, that doesn’t really meet the spirit of the standard.

More important than excusing Oracle’s mistakes, it helps to see how Oracle Corporation responds to industry trends. Sometimes it feels like they have a “fire and motion” strategy for their technologies. They add so many features that nobody can possibly keep up with them. However, adding a huge number of features may be backfiring now. The Unix philosophy of building small tools for each task seems to be taking over.

The following list shows the largest architectural and conceptual changes made to Oracle, along with the version it was introduced. These are not necessarily the most important features, but the features that tried to redefine what a database is.

* Multiversion concurrency control (MVCC): 4
* PL/SQL: 6
* Object-relational: 8
* Java: 8
* OLAP: 9
* XML, JSON, documents: 9, 12, 18
* RAC, sharding: 9, 12
* In-Memory: 12
* Containers (multitenant): 12
* Property graph: 18
* Autonomous database: 18

Adding new features rarely hurts sales. But some of those features move Oracle in a wrong direction. For example, object-relational, Java in the database, and multitenant containers all have problems.

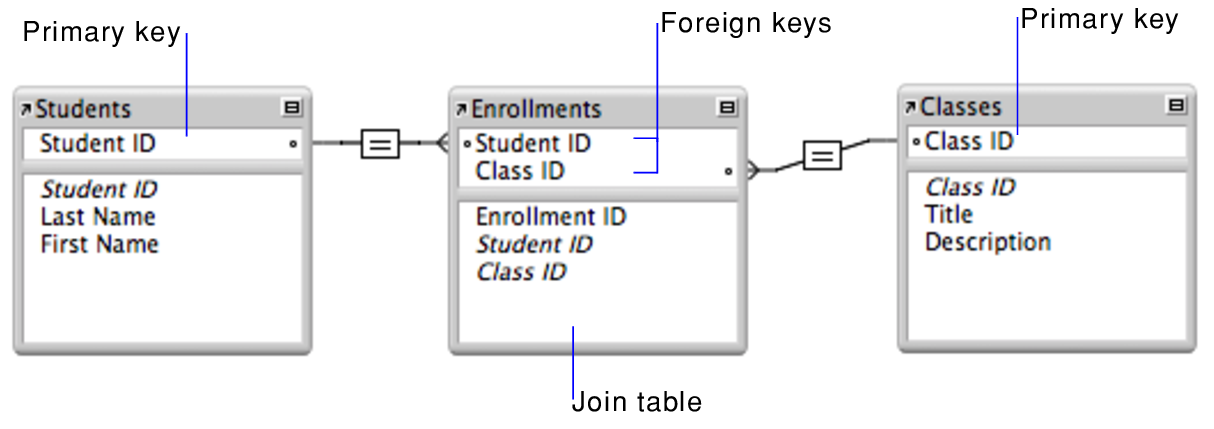
Oracle will always add a new feature to catch up with competitors, even if that feature doesn’t make sense. Not everything Oracle Corporation does is the “future.” Oracle is a huge product that sometimes moves in multiple, contradictory directions at the same time. We need to remember to not drink the Kool-Aid and not throw out proven technology for shiny new things.

On the other hand, it’s good that Oracle supports almost everything. It’s a Swiss-army knife for database solutions. We don’t need a new database for each new technology trend.

The pace of technological change is accelerating, and nobody can predict the future. Given the past, it’s safe to say that Oracle will add or invent new and important features. We don’t always want to start using new features immediately. But we should at least take the time to read about them in the “New Features” chapter of the manual.

**Question No – 17:**

**What is Table / Relation?**



Relationships are meaningful associations between tables that contain related information — they’re what make databases useful. Without some connection between tables in a database, you may as well be working with disparate spreadsheet files rather than a database system.

As we covered in our [short overview of databases](https://www.metabase.com/learn/databases/database-basics), databases are collections of tables, and those tables have fields (also known as columns). Every table contains a field known as an entity (or primary) key, which identifies the rows within that table. By telling your database that the key values in one table correspond to key values in another, you create a relationship between those tables; these relationships make it possible to run powerful queries across different tables in your database. When one table’s entity key gets linked to a second table, it’s known as a foreign key in that second table.

Identifying the connections you’ll need between tables is part of the data modeling and schema design process — that is, the process of figuring out how your data fits together, and how exactly you should configure your tables and their fields. This process often involves creating a visual representation of tables and their relationships, known an [entity relationship diagram (ERD)](https://www.metabase.com/glossary/erd), with different notations specifying the kinds of relationships. Those relationships between your tables can be:

* [One-to-one](https://www.metabase.com/learn/databases/table-relationships#one-to-one-relationship)
* [One-to-many](https://www.metabase.com/learn/databases/table-relationships#one-to-many-relationship)
* [Many-to-many](https://www.metabase.com/learn/databases/table-relationships#many-to-many-relationship)

Giving some thought to how your tables should relate to each other also helps ensure data integrity, data accuracy, and keeps redundant data to a minimum.

## One-to-one relationship

In a **one-to-one** relationship, a record in one table can correspond to only one record in another table (or in some cases, no records). One-to-one relationships aren’t the most common, since in many cases you can store corresponding information in the same table. Whether you split up that information into multiple tables depends on your overall data model and design methodology; if you’re keeping tables as narrowly-focused as possible (like in a [normalized database](https://www.metabase.com/learn/databases/normalization.html)), then you may find one-to-one relationships useful.

### Example one-to-one relationship

### 2.png

Let’s say you’re organizing employee information at your company, and you also want to keep track of each employee’s computer. Since each employee only gets one computer and those computers are not shared between employees, you could add fields to your Employee table that hold information like the brand, year, and operating system of each computer. However, that can get messy from a semantic standpoint — does computer information really belong in a table about employees? That’s for you to decide, but another option is to create a Computers table with a one-to-one relationship to the Employee table, like in the diagram below:

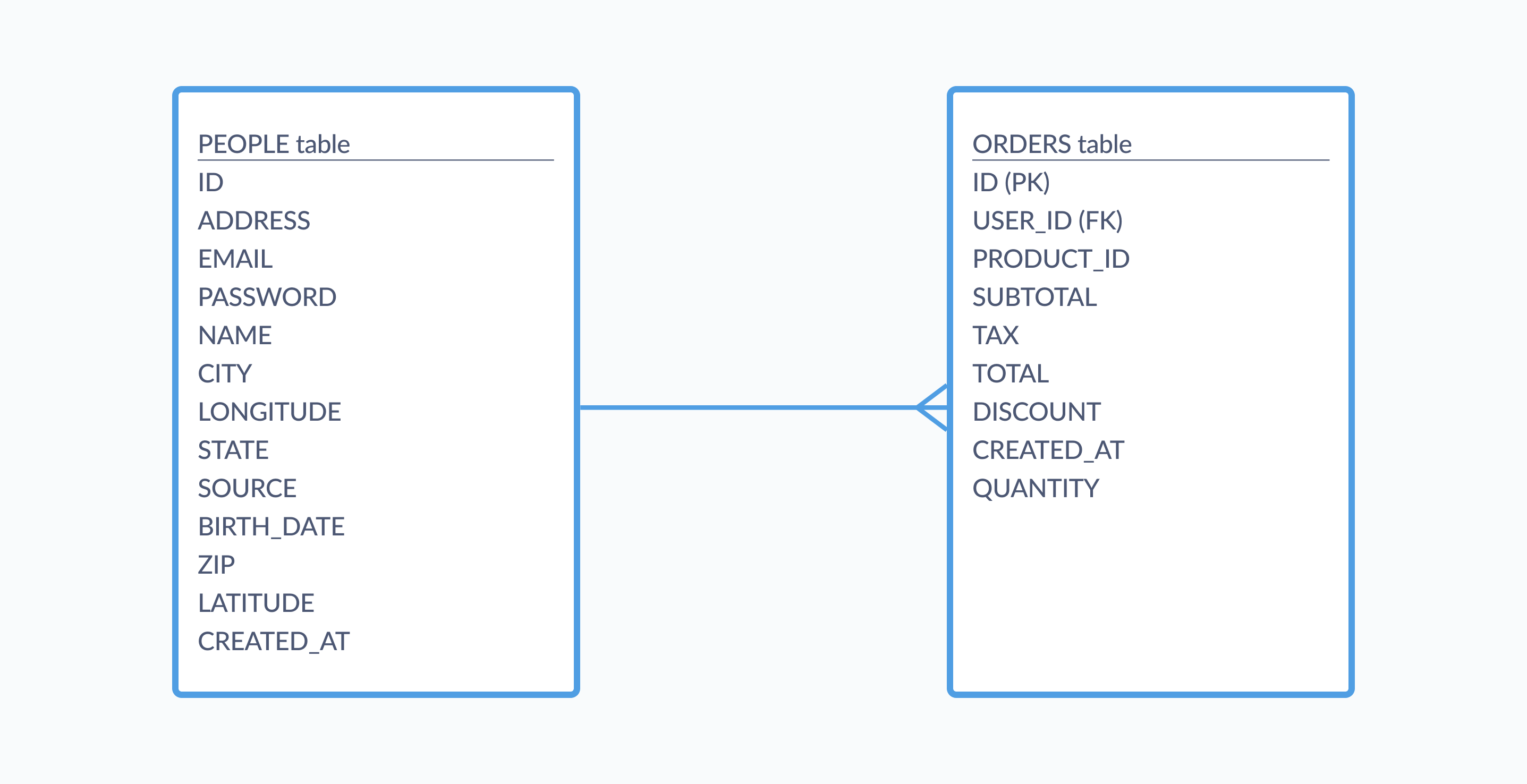
In this case, the entity key from our Employee table serves as the foreign key for the Computers table. You may have computers that aren’t yet assigned to employees, and this modeling ensures that you can still keep records for them the Computer table. And if an employee leaves the company, you’ll only need to update one field, and you can easily link a computer to a new employee.

The exact formatting of the lines used to connect tables in an ERD (known as crow’s foot notation) varies; sometimes you’ll see plain lines indicating a one-to-one relationship, other times those lines will have crosshatches like in figure 1.

One-to-one relationships can be also useful for security purposes, like if you want to store sensitive customer information in a separate table, which you link to your main Customers table with a foreign key.

## One-to-many relationship

**One-to-many** relationships are the most common type of relationships between tables in a database. In a one-to-many (sometimes called many-to-one) relationship, a record in one table corresponds to zero, one, or many records in another table.

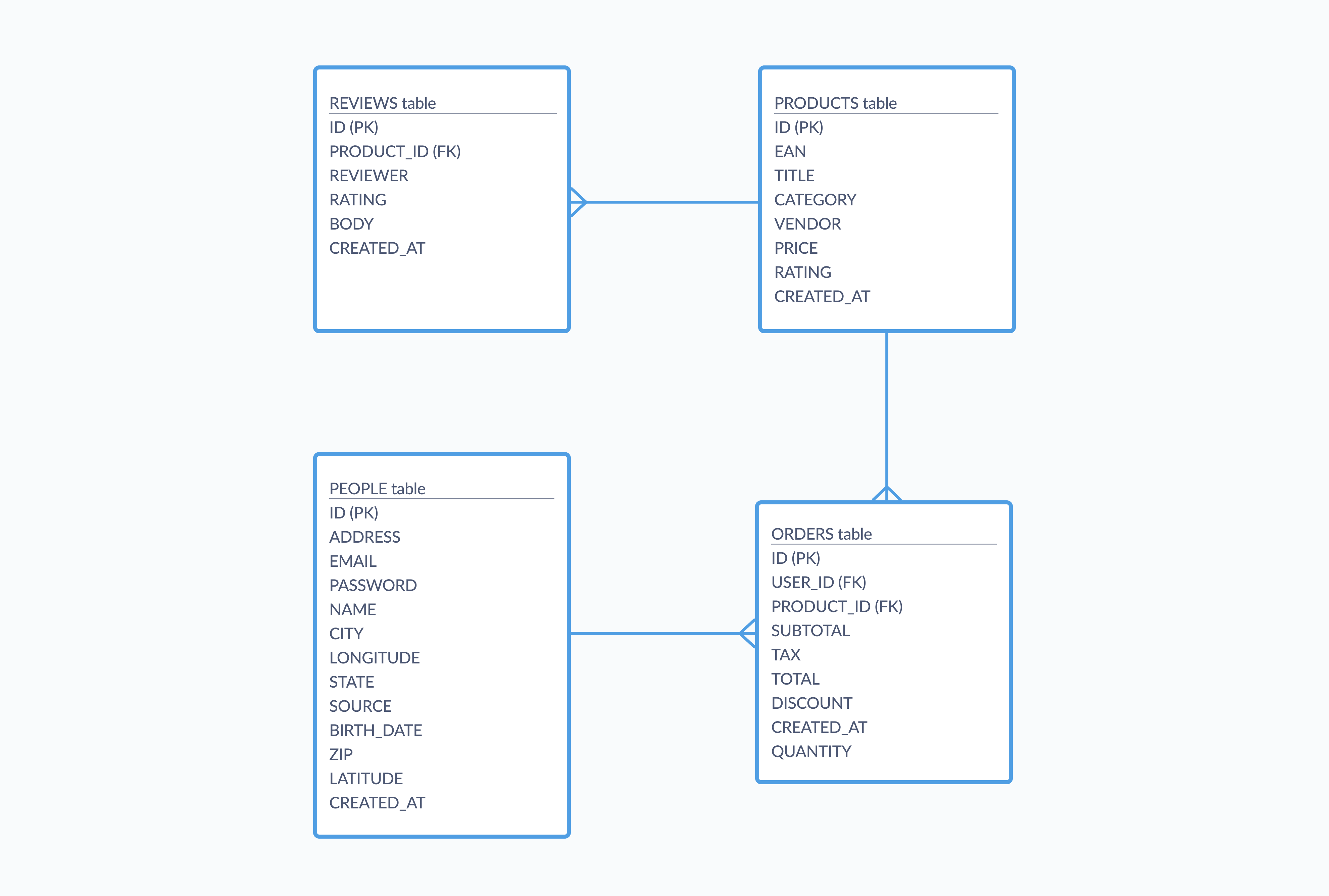


### Example one-to-many relationship

For example, think about tables for customers and their orders, like in Metabase’s Sample Database, where one record from the People table can be linked to many records in the Orders table. In this case, one customer can place many orders, and those multiple order records will all link back to a single record in the People table. That connection is codified through the User\_ID field, which is a primary key in the People table and a foreign key in the Orders table. The diagram below shows how these two tables relate to each other:

While the one person can be linked to many orders, the reverse is not true — orders are only linked to a single record in the People table, and don’t have many customers.

## Many-to-many relationship



A **many-to-many** relationship indicates that multiple records in a table are linked to multiple records in another table. Those records may only be associated with a single record (or none at all) but the key is that they can and often are linked to more than one. Many-to-many relationships aren’t very common in practical database use cases, since adhering to normalization often involves breaking up many-to-many relationships into separate, more focused tables.

In fact, your database system may not even allow for the creation of a direct many-to-many relationship, but you can get around this by creating a third table, known as a **join table**, and create one-to-many relationships between it and your two starting tables.

In this sense, the Orders table in Metabase’s Sample Database acts as a join table, creating an intermediate link between People and Products. An ERD of the Sample Database would look something like the image below, where each relationship is specified by the type of line used to connect the tables:

Technically speaking the Products and Orders tables have a one-to-many relationship, in that one product can be associated with many orders. But according to our fake company’s database, people seem to only order a single product (they’ll buy like five Lightweight Wool Computers for whatever reason). A real-world (and perhaps more business-savvy) implementation of this database would probably include a join table between the two, making it so orders could contain many different products.

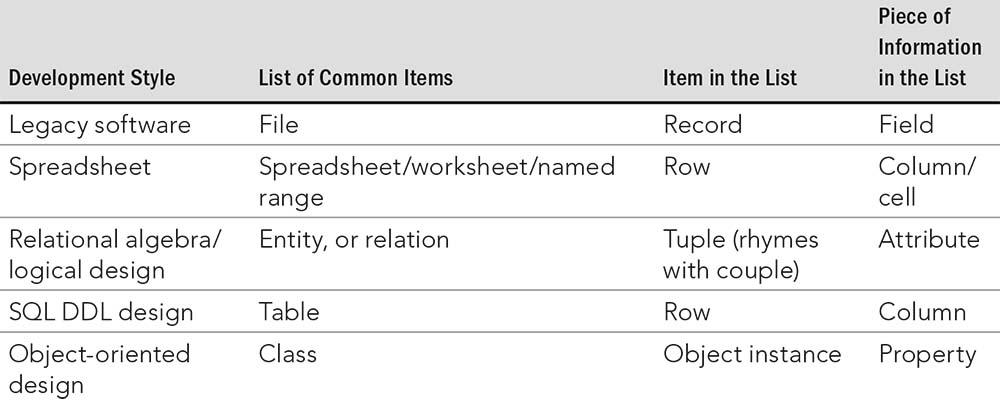
**Question No – 18:**

**What is a Row or Record:**

*Is it a row or is it a record?*

Row is a more accurate term. Some databases use the term record to describe a row, but they do have slightly different meanings. However, the terms are often used interchangeably.

To alter data in a table, you use UPDATE table\_name. Specify which values you want to change with SET field\_name = value, and add a WHERE condition to limit the table rows updated.



SQL Server developers generally refer to database elements as tables, rows, and columns when discussing the SQL Data Defi nition Language (DDL) layer or physical schema and sometimes use the terms entity, tuple, and attribute when discussing the logical design. The rest of this book uses the SQL terms (table, row, and column), but this chapter is devoted to the theory behind the design, so the relational algebra terms (entity, tuple, and attribute) are also used.

A table row is a collection of columns that comprise an information unit. A record is a representation of an object. The specifics of the distinction between rows and records are best left to the theorists. It can be confusing though. For example, query results are stored in a recordset object, implying it is storing a set of records. However, it's really storing a result set, which by its definition, represents all the rows and columns resulting from a query. The following is right out of Microsoft documentation on the recordset object: "As its name implies, the Recordset object has features that you can use, depending on your query constraints, for retrieving and displaying a set of database rows, or *records*." For practical purposes, they're the same thing.

**Question No – 19:**

**What is a column / Attribute:**

The Column attribute is applied to a property to specify the database column that the property should map to when the entity's property name and the database column name differ. The following example maps the Title property in the Book entity to a database column named Description:

1. public class Book
2. {
3. public int BookId { get; set; }
4. [Column("Description")]
5. public string Title { get; set; }
6. public Author Author { get; set; }
7. }

You can also use the Column attribute to specify the ordinal position (Order property) of the column in the resulting table, and its database-specific data type (TypeName property):

1. public class Book
2. {
3. public int BookId { get; set; }
4. [Column("Description", Order = 2, TypeName = "nvarchar(100)")]
5. public string Title { get; set; }
6. public Author Author { get; set; }
7. }

#### Fluent API

The Fluent API equivalent to the Column attribute is [HasColumnName](https://www.learnentityframeworkcore.com/configuration/fluent-api/hascolumnname-method). The Fluent API equivalent to specifying the data type is [HasColumnType](https://www.learnentityframeworkcore.com/configuration/fluent-api/hascolumntype-method). Column ordering is not implemented in Fluent API.

### On this page

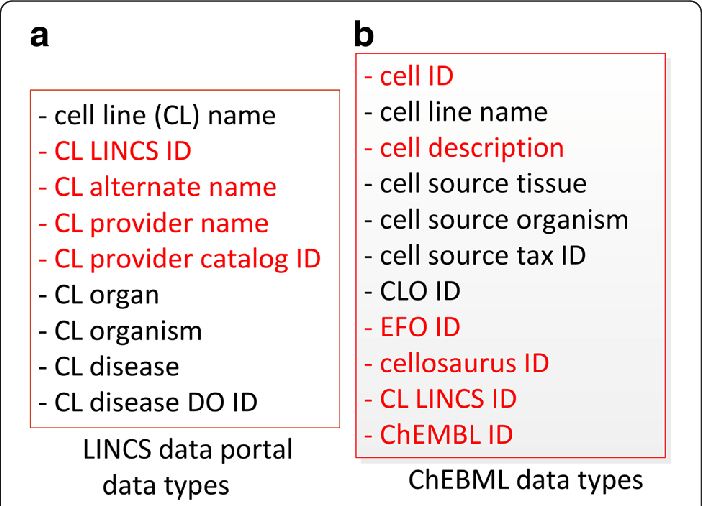
* [The Column Attribute](https://www.learnentityframeworkcore.com/configuration/data-annotation-attributes/column-attribute#the-column-attribute)

### Latest Updates

* [Package Manager Console Commands](https://www.learnentityframeworkcore.com/migrations/commands/pmc-commands)
* [Command Line Interface commands](https://www.learnentityframeworkcore.com/migrations/commands/cli-commands)
* [Commands in Entity Framework Core](https://www.learnentityframeworkcore.com/migrations/commands)
* [The Fluent API WithOne Method](https://www.learnentityframeworkcore.com/configuration/fluent-api/withone-method)
* [The Fluent API WithMany Method](https://www.learnentityframeworkcore.com/configuration/fluent-api/withmany-method)
* [The Fluent API ValueGeneratedOnAddOrUpdate Method](https://www.learnentityframeworkcore.com/configuration/fluent-api/valuegeneratedonaddorupdate-method)

**Question No – 20:**

**What is data item / cells:**



**A data item represents the piece of information you find in one cell of a data table, representing one trait of one observation.**

So, data items are just data in a cell,l. But are their different kinds, or types of data items? The short answer is yes. [Data analysts categorize](https://analystanswers.com/data-analyst-resources/) data items based on their features. Numerical data, for example, are different from textual data. Let’s look at these \_\_\_\_\_ types of data items:

* **Integer** – any number that doesn’t have a decimal point
* **Date** – a date of a given year and month
* **Time** – the time of day
* **Text** – often referred to as “string,” means simply any combination of letters instead of numbers or other symbols
* **Boolean** – TRUE or FALSE data, often migrated to YES or NO text, or 1 and 0 numbers. It is, in simple terms, binarydata.

The above examples are simple, big-picture data item types, but they’re far from comprehensive. In fact, we can dig deeper with the following list:

* **Numeric Data Item Types**
  + Integer – any number that is not a decimal. Examples include -11, 34, 0, 100.
  + Tinyint – an integer, but only numbers from 0 to 255
  + Bigint – an integer bigger than 1 trillion
  + Float – numbers too big to write out, and the scientific method is needed
  + Real – any fixed point on a line
* **Date and Time Data Item Types**
  + Date – the date sorted in different forms, including “mm/dd/yyyy” (US), “dd/mm/yyyy” (Europe), “mmmm dd, yyyy”, and “mm-dd-yy” among many more.
  + Time – the time of day, broken down as far as milliseconds
  + Date time – the date and time value of an event
  + Timestamp – stores number of **seconds passes since** 1970-01-01 00:00:00’ UTC
*  Year – stores years ranging from 1901 to 2155 in two-digit or four-digit ranges

 **Character and String Data Item Types**

* Char – fixed length of characters, with a maximum of 8,000
* Varchar – max of 8,000 characters like char, but each entry can differ in length (variable)
* Text – similar to varchar, but the maximum is 2GB instead of a specific length

 **Unicode Character and String Item Types** – unicode is a way of structuring data in the form of U+0000, where the 0’s can be any type

* nchar – fixed length with maximum length of 8,000 characters
* nvarchar – variable length with maximum of 8,000 characters
* ntext – variable length storage, only now the maximum is 1GB rather than a specific length

 **Binary Data Item Types** – a combination of 0s and 1s

* binary – fixed length with maximum of 8,000 *bytes*
* varbinary – variable length storage with maximum bytes, topped at 8,000

 **Miscellaneous Data Item Types**

* clob – also known as **C**haracter **L**arge **O**bject, is a type of sub-character that carries Unicode texts up to 2GB
* blob – carries big binary objects
* xml – a specific data type that stores XML data. XML stands for extensible markups language, and is common in data bases

**Question No – 21:**

**Different between DBMS and RDBMS:**

The main differences between DBMS and RDBMS are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **DBMS** | **RDBMS** |
| 1) | DBMS applications store **data as file**. | RDBMS applications store **data in a tabular form**. |
| 2) | In DBMS, data is generally stored in either a hierarchical form or a navigational form. | In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables. |
| 3) | **Normalization is not** present in DBMS. | **Normalization is** present in RDBMS. |
| 4) | DBMS does **not apply any security** with regards to data manipulation. | RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property. |
| 5) | DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well. |
| 6) | DBMS has to provide some uniform methods to access the stored information. | RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information. |
| 7) | DBMS **does not support distributed database**. | RDBMS **supports distributed database**. |
| 8) | DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of data**. it supports **multiple users**. |
| 9) | Examples of DBMS are file systems, **xml** etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc. |

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

**Question No – 22:**

**DBMS vs File System:**

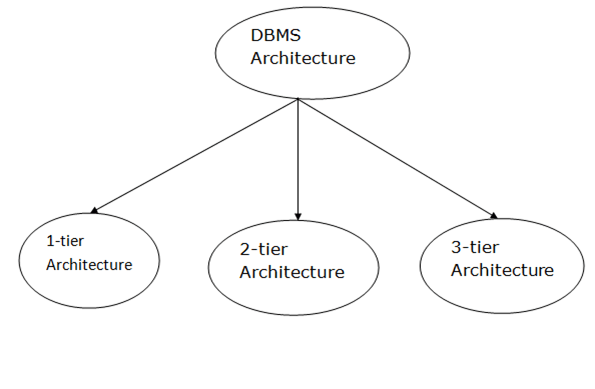
**There are the following differences between DBMS and File systems:**

|  |  |  |
| --- | --- | --- |
| **Basis** | **DBMS Approach** | **File System Approach** |
| **Meaning** | DBMS is a collection of data. In DBMS, the user is not required to write the procedures. | The file system is a collection of data. In this system, the user has to write the procedures for managing the database. |
| **Sharing of data** | Due to the centralized approach, data sharing is easy. | Data is distributed in many files, and it may be of different formats, so it isn't easy to share data. |
| **Data Abstraction** | DBMS gives an abstract view of data that hides the details. | The file system provides the detail of the data representation and storage of data. |
| **Security and Protection** | DBMS provides a good protection mechanism. | It isn't easy to protect a file under the file system. |
| **Recovery Mechanism** | DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from system failure. | The file system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will be lost. |
| **Manipulation Techniques** | DBMS contains a wide variety of sophisticated techniques to store and retrieve the data. | The file system can't efficiently store and retrieve the data. |
| **Concurrency Problems** | DBMS takes care of Concurrent access of data using some form of locking. | In the File system, concurrent access has many problems like redirecting the file while deleting some information or updating some information. |
| **Where to use** | Database approach used in large systems which interrelate many files. | File system approach used in large systems which interrelate many files. |
| **Cost** | The database system is expensive to design. | The file system approach is cheaper to design. |
| **Data Redundancy and Inconsistency** | Due to the centralization of the database, the problems of data redundancy and inconsistency are controlled. | In this, the files and application programs are created by different programmers so that there exists a lot of duplication of data which may lead to inconsistency. |
| **Structure** | The database structure is complex to design. | The file system approach has a simple structure. |
| **Data Independence** | In this system, Data Independence exists, and it can be of two types.   * Logical Data Independence * Physical Data Independence | In the File system approach, there exists no Data Independence. |
| **Integrity Constraints** | Integrity Constraints are easy to apply. | Integrity Constraints are difficult to implement in file system. |
| **Data Models** | In the database approach, 3 types of data models exist:   * Hierarchal data models * Network data models * Relational data models | In the file system approach, there is no concept of data models exists. |
| **Flexibility** | Changes are often a necessity to the content of the data stored in any system, and these changes are more easily with a database approach. | The flexibility of the system is less as compared to the DBMS approach. |
| **Examples** | Oracle, SQL Server, Sybase etc. | Cobol, C++ etc. |

**Question No – 23:**

**DBMS Architecture:**

* The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
* The client/server architecture consists of many PCs and a workstation which are connected via the network.
* DBMS architecture depends upon how users are connected to the database to get their request done.



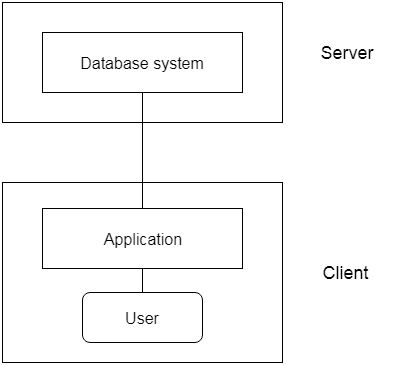
Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

**1-Tier Architecture**

* In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
* Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
* The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

**2-Tier Architecture**

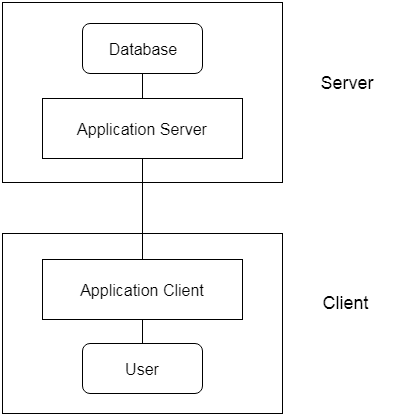
* The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
* The user interfaces and application programs are run on the client-side.
* The server side is responsible to provide the functionalities like: query processing and transaction management.
* To communicate with the DBMS, client-side application establishes a connection with the server side.



**Fig: 2-tier Architecture**

**3-Tier Architecture**

* The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
* The application on the client-end interacts with an application server which further communicates with the database system.
* End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
* The 3-Tier architecture is used in case of large web application.



**Question No – 24:**

**Types of DBMS Architecture:**

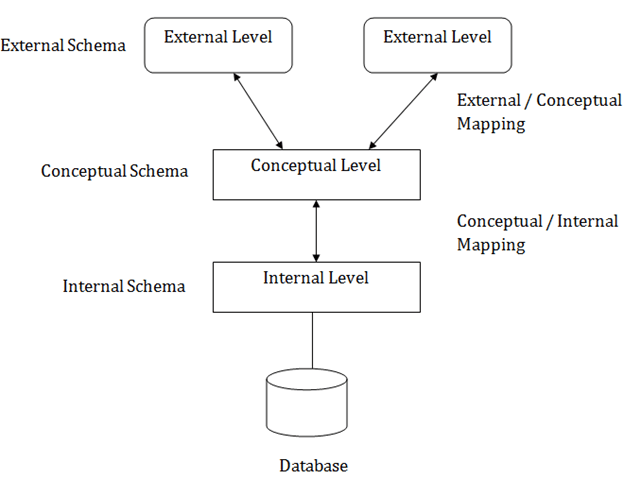
**To understand the structure of DBMS, they are classified into three types based on their build architecture:**

* One-Tier Architecture. ...
* Two-Tier Architecture. ...
* Three-Tier Architecture. ...
* Hierarchical Model. ...
* Network Model. ...
* Relational Model.

**Question No – 25:**

**Three Schema Architecture:**

* The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
* This framework is used to describe the structure of a specific database system.
* The three schema architecture is also used to separate the user applications and physical database.
* The three schema architecture contains three-levels. It breaks the database down into three different categories.



**In the above diagram:**

* It shows the DBMS architecture.
* Mapping is used to transform the request and response between various database levels of architecture.
* Mapping is not good for small DBMS because it takes more time.
* In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
* In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

## Objectives of Three schema Architecture

The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once. Thus it separates the user's view from the physical structure of the database. This separation is desirable for the following reasons:

* Different users need different views of the same data.
* The approach in which a particular user needs to see the data may change over time.
* The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
* All users should be able to access the same data according to their requirements.
* DBA should be able to change the conceptual structure of the database without affecting the user's
* Internal structure of the database should be unaffected by changes to physical aspects of the storage.
* **1. Internal Level**
* The internal level has an internal schema which describes the physical storage structure of the database.
* The internal schema is also known as a physical schema.
* It uses the physical data model. It is used to define that how the data will be stored in a block.
* The physical level is used to describe complex low-level data structures in detail.
* Storage space allocations.  
  **For Example:** B-Trees, Hashing etc.
* Access paths.  
  **For Example:** Specification of primary and secondary keys, indexes, pointers and sequencing.
* Data compression and encryption techniques.
* Optimization of internal structures.
* Representation of stored fields.

**2. Conceptual Level**

* The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
* The conceptual schema describes the structure of the whole database.
* The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
* In the conceptual level, internal details such as an implementation of the data structure are hidden.
* Programmers and database administrators work at this level.

### 3. External Level

DBMS Three schema Architecture

* At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
* An external schema is also known as view schema.
* Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
* The view schema describes the end user interaction with database systems.

## Mapping between Views

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

**There are basically two types of mapping in the database architecture:**

* Conceptual/ Internal Mapping
* External / Conceptual Mapping

**Conceptual/ Internal Mapping**

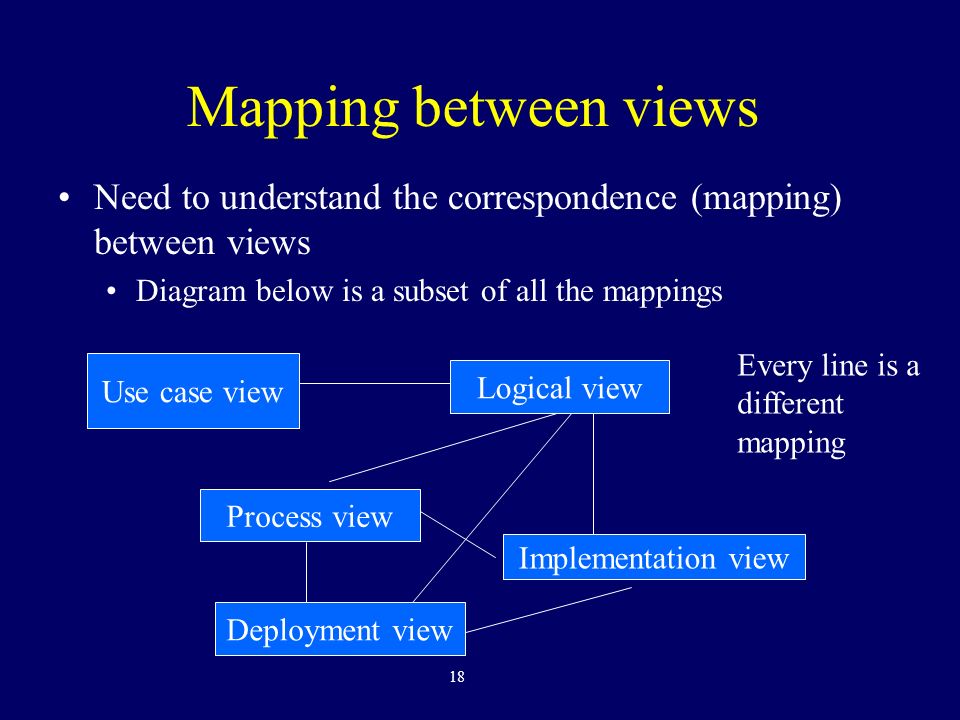
The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

**External/ Conceptual Mapping**

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

**Question No – 26:**

**Mapping between Views**:



Each of the views specified in [Views](https://wiki.sei.cmu.edu/confluence/display/SAD/Views) provides a different perspective and design handle on a system, and each is valid and useful in its own right. Although the views give different system perspectives, they are not independent. Elements of one view will be related to elements of other views, and we need to reason about these relations. For example, a module in a decomposition view may be manifested as one, part of one, or several components in one of the component-and-connector views, reflecting its runtime alter-ego. In general, mappings between views are many to many. This section describes the relations that exist among the [view](https://wiki.sei.cmu.edu/confluence/display/SAD/Views). As required by ANSI/IEEE 1471-2000, it also describes any known inconsistencies among the views.

### Mapping Between View1 and View2

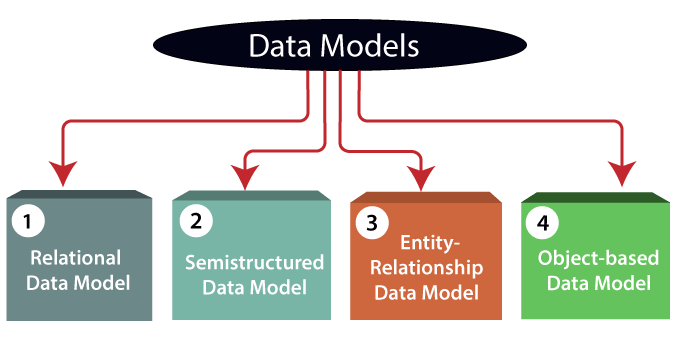
...

### Mapping Between View1 and View3

### Question No – 27:

### Data Model:

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:



**1) Relational Data Model:** This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

**2) Entity-Relationship Data Model:** An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student\_name, student\_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

**3) Object-based Data Model:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

**4) Semistructured Data Model:** This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

**Question No – 28:**

**Data Model Schema and Instance:**

An instance of the database is the data stored in a particular database for a limited time. The storage is for the finite interval of time. When we talk about the overall design of the database, then it is called a schema. It is the skeleton structure of the database, which represents the logical outlook of the complete mentioned database. A schema consists of some schema objects such as – table, primary key, foreign key, columns, views, stored procedure, data types, etc.

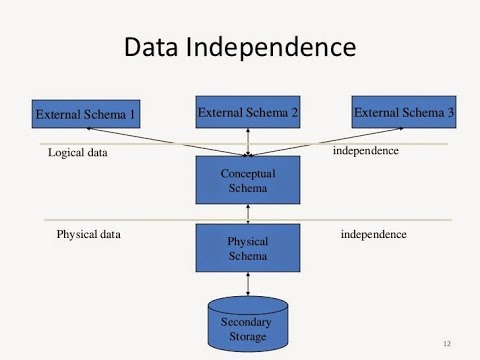
A database schema can also be in the form of visual diagrams. The diagram reflects all the database objects and their relationship with each one of them. The design of the database schema represents the designs of the database designers that can help the programmers. The software of these programs will interact with the respective database, and the process of the database creation is indeed known as data modeling.

The property of a schema diagram is that it can display only some of the aspects of a schema. The elements cover the name of data type, constraints, and record type. Other factors can’t be in specification with the help of a schema diagram. For example, in the given figure, there is no data type of every data item and no relationship between the different files.

The actual data of the database change at a regular rate; for example, in the other figure, the database changes whenever we try to add a new grade.

**Question No – 29:**

**Data Independence:**



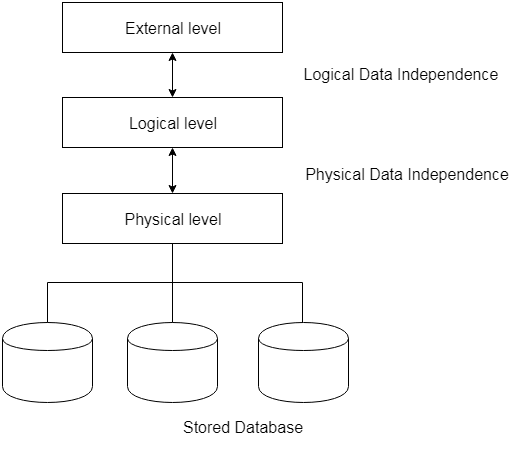
Data independence is the ability to modify the scheme without affecting the programs and the application to be rewritten. Data is separated from the programs, so that the changes made to the data will not affect the program execution and the application.

We know the main purpose of the three levels of data abstraction is to achieve data independence. If the database changes and expands over time, it is very important that the changes in one level should not affect the data at other levels of the database. This would save time and cost required when changing the database.

There are two levels of data independence based on three levels of abstraction. These are as follows −

* Physical Data Independence
* Logical Data Independence

## Physical Data Independence



Physical Data Independence means changing the physical level without affecting the logical level or conceptual level. Using this property, we can change the storage device of the database without affecting the logical schema.

The changes in the physical level may include changes using the following −

* A new storage device like magnetic tape, hard disk, etc.
* A new data structure for storage.
* A different data access method or using an alternative files organization technique.
* Changing the location of the database.

## Logical Data Independence

Logical view of data is the user view of the data. It presents data in the form that can be accessed by the end users.

Codd’s Rule of Logical Data Independence says that users should be able to manipulate the Logical View of data without any information of its physical storage. Software or the computer program is used to manipulate the logical view of the data.

Database administrator is the one who decides what information is to be kept in the database and how to use the logical level of abstraction. It provides the global view of Data. It also describes what data is to be stored in the database along with the relationship.

The data independence provides the database in simple structure. It is based on application domain entities to provide the functional requirement. It provides abstraction of system functional requirements. Static structure for the logical view is defined in the class object diagrams. Users cannot manipulate the logical structure of the database.

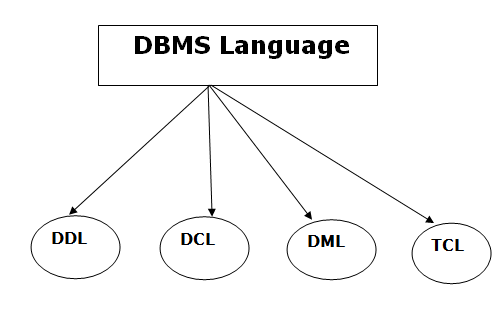
The changes in the logical level may include −

* Change the data definition.
* Adding, deleting, or updating any new attribute, entity or relationship in the database.

**Question No – 30:**

**Database Language**:

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.



**1. Data Definition Language**

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

**2. Data Manipulation Language**

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

**3. Data Control Language**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

**4. Transaction Control Language**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

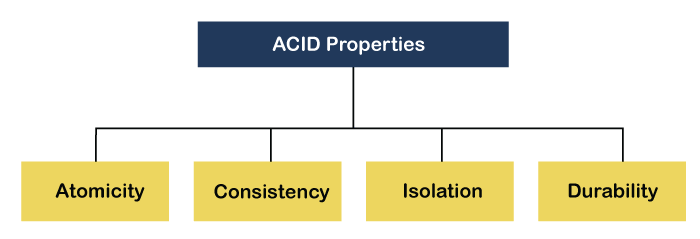
* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

**Question No – 31:**

**Acid Properties in DBMS:**

DBMS is the management of data that should remain integrated when any changes are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the **ACID** properties. The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties.

In this section, we will learn and understand about the ACID properties. We will learn what these properties stand for and what does each property is used for. We will also understand the ACID properties with the help of some examples.



**Project Name: Data Base Management System.**

**Designed by: Bipul Dutta.**

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

**14.08.2022**