What is Database?

A database is an organised collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just a database.

What is a database system? What is RDBMS ? Properties..

A Relational Database Management system (RDBMS) is a database management system that is based on the relational model. It has the following major components: Table, Record/Tuple/Row, Field, and Column/Attribute. Examples of the most popular RDBMS are MYSQL, Oracle, IBM DB2, and Microsoft SQL Server database.

Relational databases have the following properties:

- Values are atomic.
- All of the values in a column have the same data type.
- Each row is unique.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.
- Each column has a unique name.
- Integrity constraints maintain data consistency across multiple tables.

Difference between vertical and horizontal scaling

Scaling alters the size of a system. In the scaling process, we either compress or expand the system to meet the expected needs. The scaling operation can be achieved by adding resources to meet the smaller expectation in the current system, or by adding a new system in the existing one, or both.

Vertical scaling keeps your existing infrastructure but adds computing power. Your existing pool of code does not need to change — you simply need to run the same code on machines with better specs. By scaling up, you increase the capacity of a single machine and increase its throughput. Vertical scaling allows data to live on a single node, and scaling spreads the load through CPU and RAM resources for your machines.

Horizontal scaling simply adds more instances of machines without first implementing improvements to existing specifications. By scaling out, you share the processing power and load balancing across multiple machines.

What is sharding

Sharding is a method of splitting and storing a single logical dataset in multiple databases. By distributing the data among multiple machines, a cluster of database systems can store larger dataset and handle additional requests. Sharding is necessary if a dataset is too large to be stored in a single database. Moreover, many sharding strategies allow additional machines to be added. Sharding allows a database cluster to scale along with its data and traffic growth.

Keys in DBMS

A key is a set of attributes that can identify each tuple uniquely in the given relation.

Types of Keys:

- **Super Key** – A superkey is a set of attributes that can identify each tuple uniquely in the given relation. A super key may consist of any number of attributes.
- **Candidate Key** – A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called a candidate key.
- **Primary Key** – A primary key is a candidate key that the database designer selects while designing the database. Primary Keys are unique and NOT NULL.
- **Alternate Key** – Candidate keys that are left unimplemented or unused after implementing the primary key are called alternate keys.
- **Foreign Key** – An attribute 'X' is called as a foreign key to some other attribute 'Y' when its values are dependent on the values of attribute 'Y'. The relation in which attribute 'Y' is present is called the referenced relation. The relation in which attribute 'X' is present is called the referencing relation.
- **Composite Key** – A primary key composed of multiple attributes and not just a single attribute is called a composite key.
- **Unique Key** – It is unique for all the records of the table. Once assigned, its value cannot be changed i.e. it is non-updatable. It may have a NULL value.

Types of relationship

A relationship is defined as an association among several entities.

- Unary Relationship Set – Unary relationship set is a relationship set where only one entity set participates in a relationship set.
- Binary Relationship Set – Binary relationship set is a relationship set where two entity sets participate in a relationship set.
- Ternary Relationship Set – Ternary relationship set is a relationship set where three entity sets participate in a relationship set.
- N-ary Relationship Set – N-ary relationship set is a relationship set where 'n' entity sets participate in a relationship set.

Indexing in DBMS

Indexing is a way to optimise the performance of a database by minimising the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Read normalisation in details

What is denormalization ?

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean not doing normalisation. It is an optimization technique that is applied after doing normalisation.

What is functional dependency ?

A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets. It is denoted as X → Y, where X is a set of attributes that is capable of determining the value of Y. The attribute set on the left side of the arrow, X is called Determinant, while on the right side, Y is called the Dependent.

If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called **non trivial Functional dependency**.

Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For e.g.

X -> Z is a **transitive dependency** if the following three functional dependencies hold true:

- X->Y
- Y does not ->X
- Y->Z

Conflict Serializability in DBMS ..

Serializability is a concept that helps us to check which schedules are serializable. A serializable schedule is the one that always leaves the database in consistent state.

A schedule is called conflict serializability if after swapping of non-conflicting operations, it can transform into a serial schedule. The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

What is CCP ? (Concurrency Control Protocols)

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

The concurrency control protocols ensure the atomicity, consistency, isolation, durability and serializability of the concurrent execution of the database transactions.

Therefore, these protocols are categorised as:

- Lock Based Concurrency Control Protocol
- Timestamp Concurrency Control Protocol

- Validation Based Concurrency Control Protocol

Entity, Entity Type, Entity Set, Weak Entity Set..

Entity in DBMS can be a real-world object with an existence, For example, in a College database, the entities can be Professor, Students, Courses, etc.

The entity type is a collection of the entity having similar attributes.

Types of Entity type:

- **Strong Entity Type**:Strong entities are those entity types which have a key attribute. The primary key helps in identifying each entity uniquely. It is represented by a rectangle.
- **Weak Entity Type**: Weak entity type doesn't have a key attribute. Weak entity types can't be identified on their own. It depends upon some other strong entity for its distinct identity.
- **Entity Set:** Entity Set is a collection of entities of the same entity type. We can say that entity type is a superset of the entity set as all the entities are included in the entity type.

What are SQL commands ? Types of them..

https://www.javatpoint.com/dbms-sql-command

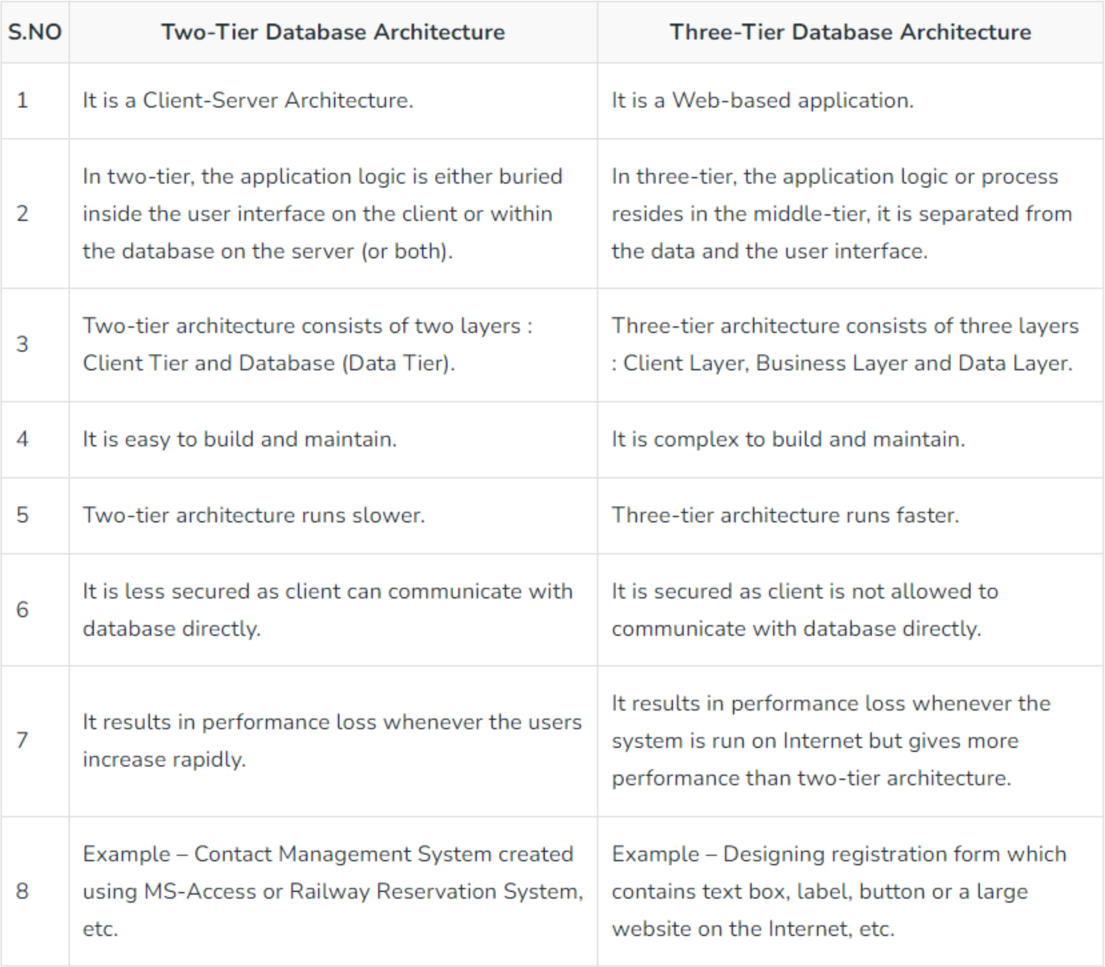
Nested Queries in SQL ?

https://www.tutorialspoint.com/explain-about-nested-queries-in-dbms

Diff between 2 tier and 3 tier architecture

1. Two-Tier Database Architecture –
In two-tier, the application logic is either buried inside the User Interface on the client or within the database on the server (or both). With two-tier client/server architectures, the user system interface is usually located in the user's desktop environment and the database management services are usually in a server that is a more powerful machine that services many clients.

2. Three-Tier Database Architecture –
In three-tier, the application logic or process lives in the middle-tier, it is separated from the data and the user interface. Three-tier systems are more scalable, robust and flexible. In addition, they can integrate data from multiple sources. In the three-tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers.



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

Difference between Intension and Extension in a Database

In a database context, intension and extension are terms used to describe different aspects of data.

Intension:
- Intension refers to the definition or description of a concept or data entity in the database.
- It represents the structure, attributes, relationships, and constraints of the data.
- Intension focuses on the logical or conceptual view of the data.
- It is also referred to as the schema or metadata of the database.
- Changes to the intension represent modifications to the database's structure or schema.

Extension:
- Extension refers to the actual instances or values of the data that exist in the database at a given point in time.
- It represents the concrete data stored in the database.
- Extension focuses on the physical or actual view of the data.
- It is also referred to as the data or records of the database.
- Changes to the extension represent modifications to the data itself, such as insertions, updates, or deletions.

To better understand the difference, consider an example of a database table representing "Employees":

Intension:
- The intension of the "Employees" table would define its attributes like "EmployeeID," "Name," "Age," and "Salary," along with any constraints or relationships with other tables.
- It describes the structure and organization of the table, but it does not contain any actual data.

Extension:
- The extension of the "Employees" table would contain the specific rows or records representing actual employees in the database.
- It includes the actual values for each attribute, such as "123," "John Doe," "30," and "5000" for an employee record.

In summary, intension describes the structure and definition of the data in a database, while extension represents the actual instances or values of the data stored in the database. Intension focuses on the logical view, while extension focuses on the physical view of the data.

Difference between share lock and exclusive lock, definition of lock

In database management systems (DBMS), a lock is a mechanism used to control concurrent access to data and ensure data consistency. Locks prevent conflicts and maintain the integrity of the database. There are two commonly used types of locks: share lock (also known as read lock) and exclusive lock (also known as write lock).

1. Share Lock (Read Lock):
   - A share lock allows multiple transactions or processes to acquire the lock simultaneously.
   - Transactions holding a share lock can read the locked data but cannot modify it.
   - Share locks are used when concurrent read access is required, allowing multiple users to access the data concurrently without conflicts.
   - Share locks are compatible with other share locks, meaning multiple transactions can hold share locks on the same data simultaneously.

2. Exclusive Lock (Write Lock):
   - An exclusive lock allows only one transaction or process to acquire the lock at a time.
   - Transactions holding an exclusive lock have both read and write access to the locked data.
   - Exclusive locks are used when a transaction needs exclusive access to modify the data, ensuring that no other transaction can read or write the data concurrently.
   - Exclusive locks are not compatible with other locks. If an exclusive lock is held on data, no other transaction can acquire a share lock or exclusive lock on the same data until the lock is released.

Locks play a vital role in concurrency control, ensuring that concurrent transactions operate on data without causing inconsistencies or conflicts. The choice of lock type depends on the requirements of the transaction and the desired level of concurrency and isolation. Share locks allow multiple readers but no writers, promoting concurrent read access. Exclusive locks ensure exclusive access for writing, preventing concurrent modifications that could result in inconsistent or incorrect data.

It's worth noting that lock behavior and mechanisms may vary across different DBMS implementations and configurations.

1. Cursor in PL/SQL :
A cursor can be basically referred to as a pointer to the context area.Context area is a memory area that is created by Oracle when SQL statement is processed.The cursor is thus responsible for holding the rows that have been returned by a SQL statement.Thus the PL/SQL controls the context area by the help of cursor.An Active set is basically the set of rows that the cursor holds. The cursor can be of two types: Implicit Cursor, and Explicit Cursor.

Advantages of Cursor:

- They are helpful in performing the row by row processing and also row wise validation on each row.
- Better concurrency control can be achieved by using cursors.
- Cursors are faster than while loops.

Disadvantages of Cursor:

- They use more resources each time and thus may result in network round trip.
- More number of network round trips can degrade the performance and reduce the speed.

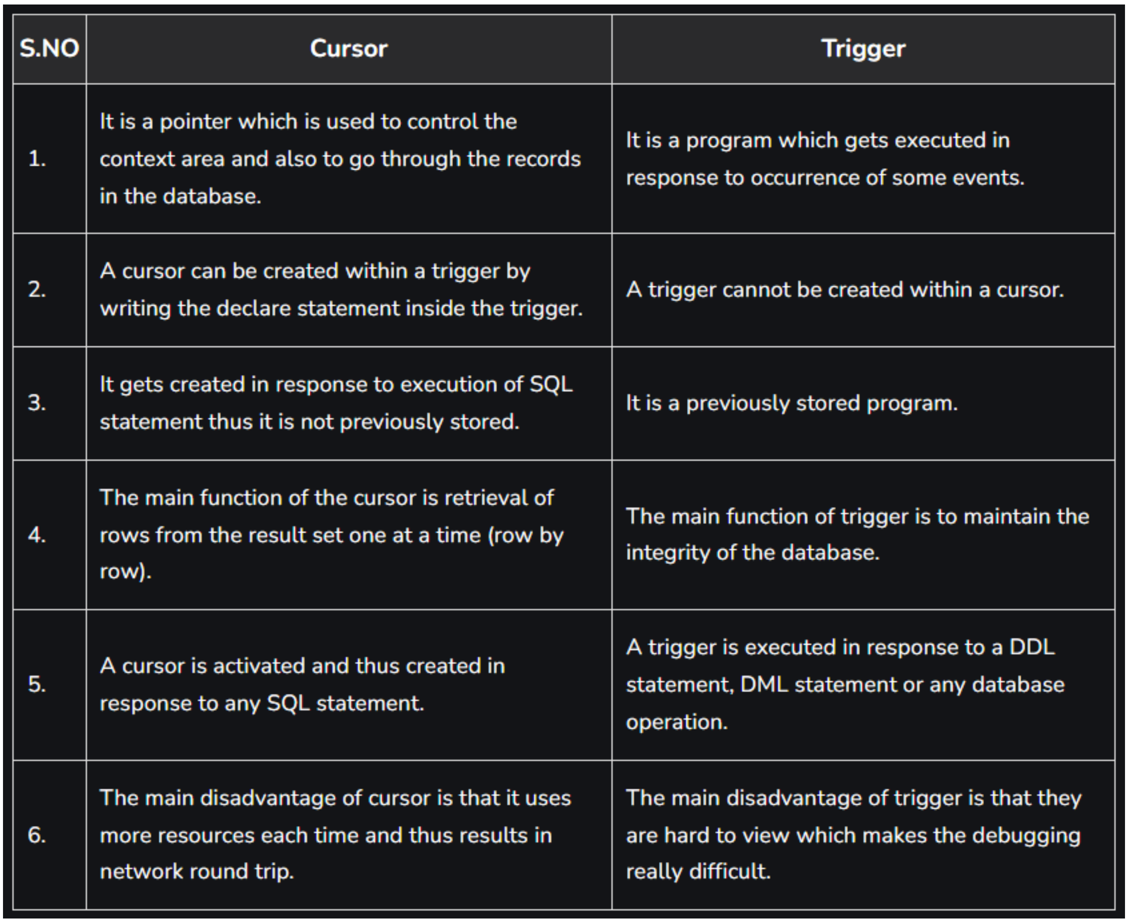
2. Trigger in PL/SQL :
A Trigger is basically a program which gets automatically executed in response to some events such as modification in the database.Some of the events for their execution are DDL statement, DML statement or any Database operation.Triggers are thus stored within the database and come into action when specific conditions match.Hence, they can be defined on any schema, table, view etc. There are six types of triggers: BEFORE INSERT, AFTER INSERT, BEFORE UPDATE, AFTER UPDATE, BEFORE DELETE, and AFTER DELETE.

Advantages of Trigger:

- They are helpful in keeping the track of all the changes within the database.
- They also help in maintaining the integrity constraints.

Disadvantages of Trigger:

- They are very difficult to view which makes the debugging also difficult.
- Too much use of the triggers or writing complex codes within a trigger can slow down the performance.



S.NO	Cursor	Trigger
1.	It is a pointer which is used to control the context area and also to go through the records in the database.	It is a program which gets executed in response to occurrence of some events.
2.	A cursor can be created within a trigger by writing the declare statement inside the trigger.	A trigger cannot be created within a cursor.
3.	It gets created in response to execution of SQL statement thus it is not previously stored.	It is a previously stored program.
4.	The main function of the cursor is retrieval of rows from the result set one at a time (row by row).	The main function of trigger is to maintain the integrity of the database.
5.	A cursor is activated and thus created in response to any SQL statement.	A trigger is executed in response to a DDL statement, DML statement or any database operation.
6.	The main disadvantage of cursor is that it uses more resources each time and thus results in network round trip.	The main disadvantage of trigger is that they are hard to view which makes the debugging really difficult.

A **stored procedure** in SQL is a type of pre-written code that can be stored for later execution and then used many times hence, saving time. It is a group of SQL statements that performs the task. The stored procedure can be invoked explicitly whenever required. It may accept some inputs in the form of parameters, these may be one parameter or multiple parameters.

Syntax:

```
CREATE PROCEDURE stored_procedure_name
AS
sql_statement
GO;
EXEC stored_procedure_name;
```

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. In this article we will learn about creating , deleting and updating Views.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```