TYPES OF INDEXES AND PARTITIONS

Types of Partitions:

- **Horizontal partitioning** (often called *sharding*)
  In this strategy, each partition is a separate data store, but all partitions have the same schema. Each partition is known as a *shard* and holds a specific subset of the data.
  The most important factor is the choice of this type (sharding key). It can be difficult to change the key after the system is in operation. The key must ensure that data is partitioned to spread the workload as evenly as possible across the shards.

- **Vertical partitioning**
  In this strategy, each partition holds a subset of the fields for items in the data store. The fields are divided according to their pattern of use.
  - Relatively slow-moving can be separated from the more dynamic data. Slow moving data is a good candidate for an application to cache in memory.
  - Sensitive data can be stored in a separate partition with additional security controls.
  - Vertical partitioning can reduce the amount of concurrent access that's needed.

- **Functional partitioning**
  In this strategy, data is aggregated according to how it is used by each bounded context in the system. Itis a way to improve isolation and data access performance.

Types of Indexes:

Indexing is a way to optimize the performance of a database by minimizing 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.

Different types of indexes can be created for different purposes and performance benefits.

- **Clustering**
  Indexes can improve the performance of most query operations because they provide a more linear access path to data, which is stored in pages. In addition, because rows with similar index key values are stored together, sequential detection prefetching is more efficient when clustering indexes are used. Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be unique for each record.

- **Nonclustering**
  Index just tells us where the data lies, it gives us a list of virtual pointers or references to the location where the data is actually stored. Data is not physically stored in the order of the index. Instead, data is present in leaf nodes. It requires more time as compared to the clustered index because some amount of extra work is done in order to extract the data by further following the pointer.

- **Multilevel Indexing**
  Segregates the main block into various smaller blocks so that the same can stored in a single block. The outer blocks are divided into inner blocks which in turn are pointed to the data blocks. This can be easily stored in the main memory with fewer overheads.