1. SSD: How does the Flash Translation Level (FTL) work in SSDs?

Ans: The main factor that made adoption of SSDs so easy is that they use the same host interfaces as HDDs, Though the way Logical Block Addresses works in HDDs, does not work in SSDs. For this reason, an additional component is required to hide the inner characteristics of NAND flash memory and expose only an array of LBAs to the host. This component is called the Flash Translation Layer (FTL). The FTL has two main purposes: logical block mapping and garbage collection.

- Logical block mapping: The logical block mapping translates logical block addresses (LBAs) from the host space into physical block addresses (PBAs) in the physical NAND-flash memory space. This mapping takes the form of a table, which for any LBA gives the corresponding PBA.

- Garbage collection: The garbage collection process in the SSD controller ensures that "stale" pages are erased and restored into a "free" state so that the incoming write commands can be processed.

2. SSD: Why are sequential writes important for performance on SSDs?

Ans: Sequential writes is very much related with garbage collection. The need for garbage collection affects an SSD's performance because any write operation needs to await the availability of new free space created through the garbage collection process. Because garbage collection occurs at the block level, there is also a significant performance difference, depending on whether sequential or random data is involved. Sequential files fill entire blocks, which dramatically simplifies garbage collection. The situation is very different for random data.

3. SSD: Discuss the effect of alignment of blocks to SSD pages.

Ans: We know that in each block there could be many pages. I think the alignment of Block is related to page mapping. The block alignment saves SSD Some serious space and manufacturing cost. Let's assume that an SSD drive has 256 pages per block. This means that block-level mapping requires 256 times less memory than page-level mapping, which is a huge improvement for space utilization.

4. RocksDB: Describe the layout of MemTable and SSTable of RocksDB.

Ans: From a technical point of view both MemTable and SSTable seems exactly same – a table. In RocksDB MemTable usages as an initial storage point. Whenever data comes in, that initially stores in MemTable. Which are unsorted. Once MemTable is filled then the entire data is flushed to SSTable. Which data stored string key ordered. Also linked with some index and a compactor to reduce the redundant data. Also, while reading data, the query directly goes to MemTable first then SSTable.

5. RocksDB: What happens during compaction in RocksDB?

Ans: The main/basic job of a compactor is to merge SSTable by removing redundant and delete keys and creating a compact/merged SSTable. In most certain cases compaction happens in RocksDB every 30 minutes. Compaction is a good program for RocksDB. But there are some downsides as well. During compaction the write process could be significantly slow.

6. LSM-trees vs B+-trees. Give some reasons for why LSM-trees are regarded as more efficient than B+-trees for large volumes of inserts.

Ans: LSM tree and B+ tree is the most used algorithm in current DB systems. But both has different strong and week points. Where B+ tree is very efficient for reading data along with sequence, on the other hand LSM tree is very efficient for insert a large volume of data. Why, I believe mostly because of our recent technological advancement, or short term in memory storage. LSM tree usage MemTable for the initial storage of data, which is very insert efficient. So, while data is coming continuously LSM tree outperform B+ tree.

7. Regarding fault tolerance, give a description of what hardware, software and human errors may be?

Ans: Fault tolerance refers to the ability of a system to continue operating without interruption when one or more component failed. For hardware error, it's the situation when a Computer Hard drive or RAM goes down and for software error, its could be when a software code runs for the first time or a untrace bugs occurs and crash the whole system. And for human error when human make an error by mistake or more formally human error refers to something having been done that was "not intended by the actor.

8. Give an overview of tasks/techniques you may take/do to achieve fault tolerance.

Ans: Most of the cases fault tolerance could be achieve by introducing more backup systems. For example, instate of once PC we could impose multiple PC, or we could do both horizontal and vertical scaling of the hardware to make fault tolerance. Or in case of software, we could employ multiple versions of the same software or could create multiple virtual instances of the same software. And for human errors, impose the limit of the system to prevent some known human errors.

9. Compare SQL and the document model. Give advantages and disadvantages of each approach. Give an example which shows the problem with many-to-many relationships in the document model, e.g., how would you model that a paper has many sections and words, and additionally it has many authors, and that each author with name and address has written many papers?

Ans: SQL and document model, both has its advantage and disadvantage. While SQL mostly focus on relationships and structure and fixed schema, the document model mostly focusses on NOSQL and non-structure data. Where there is a critical relationship, SQL performs better. While each model has a very distinctive query language. In case of many to many relationships in the paper I would choose SQL. As, it can create relationship very well with predefine schema, we can also choose document mode, but document model is not very focused on relationship and keys and indexes.

10. When should you use a graph model instead of a document model? Explain why. Give an example of a typical problem that would benefit from using a graph model.

Ans: Document model is best for over changing data structure and graph model is best when each object is highly connected with other objects in a model. For example, financial data or social media data. In social media data each object, either human, entity, organization is highly connected with each other. In many ways each entity has connected with every other and in both ways.

11. Column compression: You have the following values for a column: 43 43 43 87 87 63 63 32 33 33 33 33 89 89 89 33

a) Create a bitmap for the values.

b) Create a runlength encoding for the values

Ans: Bitamp Table -

Values	43	43	43	87	87	63	63	32	33	33	33	33	89	89	89	33
=32	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
=33	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1
=43	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
=63	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
=87	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
=89	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

RangeLength Encoding -

=32: 7, 1 (7 zeros, 1 one, rest zero)

=33: 8,4,3,1 (8 zeros, 4 ones, 3 zeros, 1 one)

=43: 3 (3 ones, rest zeros)

=63: 5,2 (5 zeros, 2 ones, rest zeros)

=87: 3,2 (3 zeros, 2 ones, rest zeros)

=89: 12,3 (12 zeros, 3 ones, rest zeros)

12. We have different binary formats / techniques for sending data across the network:

- **MessagePack**
- **Apache Thrift**
- **Protocol Buffers**
- **Avro**

In case we need to do schema evolution, e.g., we add a new attribute to a Person structure:

Labour union, which is a String. How is this supported by the different systems? How is forward

and backward compatibility supported?

Ans: Here all the given formats are binary encoding. Regarding a new attribute adding to person, I think Avro is better and backward compatible. If we are talking about dynamic schema change then. And, about others MessagePack is very non-human readable format and schema less. While others must have to have a schema either static or dynamic. Apache and Protocol buffer works almost exactly same while Avro works for dynamic schema. But Avro also save schema in a large, so that is not very memory efficient.