Assignment 1

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

   The FTL in SSDs is responsible for mapping logical block addresses (LBAs) to physical flash memory locations.

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

   Sequential writes are important for SSD performance because NAND flash memory can write data faster in large, contiguous blocks. In contrast, random writes are slower due to the need for read-modify-write operations.

3. **SSD: Discuss the effect of alignment of blocks to SSD pages. See e.g. Figure 2.5 of Dybvik.**

   Aligning blocks to SSD pages is crucial to prevent unnecessary read-modify-write operations. Misalignment can lead to decreased performance and increased write amplification as the SSD may have to read and write entire pages instead of just the necessary data.

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

   MemTable is an in-memory data structure that stores recently written data in RocksDB. they store in sorted key-value pairs.

5. **RocksDB: What happens during compaction in RocksDB?**

   Compaction in RocksDB is the process of merging and organizing SSTables to reduce the number of files and improve read performance.

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.**

   LSM-trees are often considered more efficient for large volumes of inserts because they minimize random writes, use sequential disk I/O, and perform well with write-intensive workloads.

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

- Hardware errors: Failures in storage devices, CPUs, memory, power supplies, etc.
- Software errors: Bugs, crashes, or data corruption in the software stack.
- Human errors: Mistakes made by administrators or users, such as accidental data deletion.

8. **Give an overview of tasks/techniques you may take/do to achieve fault tolerance.**
   - Error detection: Employ mechanisms to detect errors.
   - Error correction: Employ mechanisms to correct 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?**
   - SQL:
     - Advantages: the possibility complex querying.
     - Disadvantages: Less flexible for schema changes, not ideal for hierarchical or semi-structured data.
   - Document Model (e.g., MongoDB):
     - Advantages: Schema flexibility, good for hierarchical and semi-structured data.
     - Disadvantages: Limited support for complex joins and transactions.
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.

    Use a graph model when relationships between data entities are complex and important. Example: Twitter, to get relations between user and a tweet which makes it easy to join the tweets table and user table to get all tweets with a “where clause”.

For a user to get the most relevant tweets for their home page based on their followings, relevancy and popularity

11. **What are the situations that decide that you should use textual encodings instead of binary encodings for sending data?**

    Textual encodings are much easier or humans to understand and read, so it will make it easier to debug, but the tradeoff is runtime, so when working in smaller datasets, it might be more appropriate to use textual encodings because the tradeoff will not be significant

12. **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.**

       Example of a bitmap, not using prefix encoding

       - 43: 001
       - 87: 010
       - 63: 011
       - 32: 100
       - 33: 101
       - 89: 110

       result: 00100100101001001101110010110110110111010110101

       for readability: 001 001 001 010 010 011 011 100 101 101 101 101 110 110 110 101

    b. **Create a runlength encoding for the values**

       (43, 3), (87, 2), (63, 2), (32, 1), (33, 4), (89, 3), (33, 1).

13. **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?

MessagePack:
MessagePack has limited support for schema evolution, and adding a new attribute may not break backward compatibility if older code can ignore unknown fields.

Apache Thrift:
Apache Thrift provides better schema evolution support, allowing you to add new fields without breaking backward compatibility, as long as the older code can ignore unknown fields.

Protocol Buffers (protobuf):
Protocol Buffers offer excellent schema evolution support, enabling you to add new fields, including "Labour union," with both backward and forward compatibility while using numeric tags for identification.

Avro:
Avro is designed for schema evolution, and you can easily add a new attribute like "Labour union" without breaking backward or forward compatibility as schemas are stored with the data and support evolution.