Redis Study Guide

🔍 Quiz

1. What is the primary characteristic of Redis that makes it perform operations quickly?
2. Explain the basic structure of data storage in Redis.
3. Name two non-string data structures supported by Redis and provide a brief use case for each.
4. How does Redis support real-time messaging?
5. Briefly describe the two main persistence mechanisms in Redis.
6. What is the purpose of Redis Sentinel in a Redis deployment?
7. Explain the concept of atomic operations in Redis.
8. How can you set a time limit for how long a key will exist in Redis?
9. Identify one significant limitation of Redis regarding dataset size.
10. Why is Redis generally not ideal for complex relational data or detailed analytical queries?

✅ Quiz Answer Key

1. Redis is an in-memory data store, meaning it keeps all data in RAM. This allows for very fast read and write operations compared to disk-based storage.
2. Redis stores data in a key-value pair format. Each data item is identified by a unique key, and the associated value can be of various data types.
3.

- **Lists**: For ordered collections like queues
- **Sets**: For unique collections like tags
- **Sorted Sets**: For ranked lists like leaderboards
- **Hashes**: For sub-collections like user profiles

4. Redis supports real-time messaging through its Publish/Subscribe (Pub/Sub) feature, where clients can publish messages to channels and other clients subscribed to those channels receive the messages instantly.
5.

- **RDB (Redis Database file)**: Takes periodic snapshots of the data to disk
- **AOF (Append-Only File)**: Logs every write command sequentially

6. Redis Sentinel is a service used for high availability. It monitors Redis master nodes and automatically promotes a replica node to become the new master if the original master fails.
7. Atomic operations in Redis mean that each command is executed completely or not at all, ensuring that there are no partial updates or race conditions.
8. You can set a TTL (Time-To-Live) using commands like `EXPIRE` or `SETEX`. After the

specified time passes, the key is automatically deleted.
9. A significant limitation is that the dataset size is limited by the amount of available RAM on the server. Storing very large datasets can become costly.
10. Redis lacks built-in support for complex SQL-like queries, joins, and advanced aggregation functions. It is designed for fast key-based access, not analytical queries.

📝 Essay Format Questions

1. Compare and contrast the RDB and AOF persistence mechanisms in Redis, discussing their respective trade-offs in terms of data durability and performance.
2. Discuss the various scalability options available in Redis (Replication, Sentinel, Cluster) and explain how they address different aspects of scaling (read capacity, high availability, data distribution).
3. Explain how Redis's rich data structures (Lists, Sets, Sorted Sets, Hashes) enable its use in diverse application scenarios beyond simple caching. Provide specific examples for at least three different data structures.
4. Analyze the primary weaknesses of Redis, particularly its memory dependency and limitations in querying and analytics, and discuss the types of use cases where these weaknesses would make Redis unsuitable as the primary data store.
5. Describe the concept of atomic operations in Redis and how it contributes to data consistency. Additionally, discuss the role of `MULTI/EXEC` transactions and Lua scripting in achieving atomic execution of multiple commands, noting any important caveats.

📚 Glossary of Key Terms

- **In-memory**: Data is stored directly in the computer's RAM for very fast access.
- **Key-value store**: A NoSQL database that stores data as key-value pairs.
- **NoSQL**: "Not Only SQL" — databases that don't follow the traditional relational model.
- **RAM**: Random Access Memory — high-speed memory used for temporary storage.
- **Caching**: Storing frequently accessed data for faster retrieval.
- **Session storage**: Data related to user sessions in apps or websites.
- **Real-time analytics**: Immediate analysis of incoming data streams.
- **Leaderboard**: Ranked list of scores or performance metrics.

📁 Data Structures in Redis

- **Lists**: Ordered collections of strings (e.g., message queues).
- **Sets**: Unordered collections of unique strings (e.g., tags).
- **Sorted Sets**: Sets ordered by a numeric score (e.g., leaderboards).
- **Hashes**: Field-value pairs in a single key (e.g., user profiles).
- **Bitmaps**: Binary tracking of flags or counters.
- **HyperLogLog**: Estimates unique counts with minimal memory.

- **Geospatial Indexes**: Location-based data and queries.

🔄 Messaging & Persistence

- **Pub/Sub**: Real-time message distribution via channels.
- **Persistence**: Retaining data between restarts.
- **RDB**: Snapshot-based data persistence.
- **AOF**: Write-log-based data persistence.

⚙️ Replication & Scaling

- **Replication**: Master-slave data copying for scalability.
- **Master-slave model**: One master for writes, replicas for reads.
- **Redis Sentinel**: Automated monitoring and failover tool.
- **Redis Cluster**: Shards data across nodes for horizontal scaling.

🔐 Transactions & Security

- **Atomic Operations**: All-or-nothing command execution.
- **MULTI/EXEC**: Transactional command grouping.
- **Lua Scripting**: Server-side scripting for custom logic.
- **TTL**: Auto-expiry timer for keys.
- **EXPIRE / SETEX**: Commands for setting TTL.
- **ACL**: Access control for command and key permissions.

🧠 Performance & ACID

- **Single-threaded**: Redis uses one thread for operations.
- **ACID**: Redis ensures atomicity but lacks full ACID compliance.
- **Sharding**: Dividing data across nodes in Redis Cluster.
