📘 DynamoDB Study Guide

🧠 Quiz

1. What makes DynamoDB a "serverless" database, and what is the main benefit of this characteristic for a developer?
2. Explain the difference between the Key-Value and Document data models in DynamoDB.
3. How does automatic partitioning contribute to DynamoDB's scalability and performance?
4. What is the primary function of Global Tables in DynamoDB, and what SLA does AWS offer for them?
5. Describe how IAM roles and policies enhance security in DynamoDB.
6. What is the purpose of DynamoDB Streams, and how can they be integrated with AWS Lambda?
7. Explain the concept of Time to Live (TTL) in DynamoDB and provide an example use case.
8. What are the key differences between a Query operation and a Scan operation in DynamoDB, and which is generally more efficient for large tables?
9. Briefly explain the difference between a Global Secondary Index (GSI) and a Local Secondary Index (LSI).
10. What is a "Hot Partition" in DynamoDB, and how can it impact performance?

✅ Quiz Answer Key

1. **Serverless** means AWS manages all the underlying infrastructure, including servers and patching. This allows developers to focus on data logic without worrying about infrastructure maintenance.
2. **Key-Value Model** uses unique keys for storing data like a dictionary. **Document Model** stores data as JSON or map attributes, supporting complex, flexible schemas.
3. **Automatic partitioning** spreads data across partitions to balance the load, enabling horizontal scalability and high performance.
4. **Global Tables** replicate data across AWS regions in real-time, ensuring high availability. AWS offers a **99.999% SLA** for Global Tables.
5. **IAM roles/policies** define who can access what in DynamoDB, allowing fine-grained access control to items or attributes.
6. **DynamoDB Streams** log every change (insert/update/delete) and can trigger **AWS Lambda** for real-time, event-driven processing.
7. **TTL (Time to Live)** automatically deletes items after a set expiry time. Example: auto-expiring session tokens.
8. **Query** uses partition (and optional sort) keys to fetch items efficiently. **Scan** reads the entire table and is inefficient for large datasets.

9. **GSI** allows querying with a different partition/sort key across the entire table. **LSI** uses the same partition key but allows a different sort key within the partition.
10. A **Hot Partition** is over-accessed compared to others, leading to throttling and degraded performance due to uneven request distribution.

✍️ Essay Format Questions

1. Discuss the trade-offs and considerations when designing a DynamoDB schema, particularly highlighting the importance of planning access patterns before modeling.
2. Explain how DynamoDB's built-in security features (IAM, encryption at rest, TLS in transit) help secure sensitive data.
3. Analyze the pros and cons of **Provisioned** vs. **On-Demand** modes in capacity planning, especially for cost predictability and handling unpredictable spikes.
4. Describe use cases where DynamoDB is more suitable than relational databases, such as real-time, global, or event-driven applications.
5. Discuss DynamoDB challenges like item size limits, lack of joins/aggregations, and index complexity, including potential workarounds.

📚 Glossary of Key Terms

- **AWS-Managed**: AWS handles infrastructure, patching, and scaling.
- **Serverless**: No need to manage servers; focus only on data logic.
- **Key-Value Model**: Data stored as key-value pairs.
- **Document Model**: Semi-structured JSON-style records.
- **Flexible Schema**: Schema can vary per item.
- **Single-Digit Millisecond Latency**: Fast read/write response (1–9 ms).
- **High Throughput**: Can process millions of requests/sec.
- **Automatic Horizontal Scaling**: Load-balanced data distribution.
- **On-Demand Provisioning**: Auto-scale resources based on usage.
- **Scale-to-Zero**: Cost-saving when idle.
- **Automatic Partitioning**: Data evenly split across partitions.
- **Scale-Out**: Add more partitions/servers as needed.
- **Global Tables**: Real-time multi-region data replication.
- **99.999% SLA**: Ultra-high uptime guarantee.
- **Multi-Active Access**: Concurrent access from multiple regions.
- **IAM Roles/Policies**: Secure, fine-grained access controls.
- **Data at-Rest Encryption**: Stored data is encrypted.
- **TLS In-Transit Encryption**: Data encrypted during transfer.
- **Fine-Grained Access**: Permissions down to item/attribute level.
- **Backup & Restore**: Scheduled or manual backups.
- **Point-in-Time Recovery**: Restore to any past second.
- **DynamoDB Streams**: Real-time logs of data changes.
- **Event-Driven**: React to changes with Lambda, etc.
- **TTL (Time to Live)**: Auto-delete old items.
- **Change Data Capture**: Track item changes via streams.

- **Stream Records**: Logged data change events.
- **Real-Time Processing**: Near-instant processing of new data.
- **Query**: Efficient targeted fetch using keys.
- **Scan**: Full-table read; slow on large datasets.
- **Filters**: Narrow down results after Query/Scan.
- **Projection**: Fetch only selected attributes.
- **Secondary Indexes**: Alternative query paths.
- **Global Secondary Index (GSI)**: Alternate partition/sort keys.
- **Local Secondary Index (LSI)**: Same partition key, different sort key.
- **ACID Transactions**: Reliable, consistent multi-item operations.
- **Atomic Transactions**: All-or-nothing multi-step operations.
- **Hot Partitions**: Overloaded partitions due to uneven access.
- **Data Skew**: Uneven data distribution.
- **400 KB Item Size Limit**: Max item size is 400 KB.
- **Provisioned Mode**: Pre-allocated capacity (predictable costs).
- **On-Demand Mode**: Pay per request (scalable but cost-variable).
- **DynamoDB Local**: Offline local version for development.