All about DynamoDB!

1: DynamoDB Kya Hai?

- **NoSQL DB on AWS:** Puri tarah se AWS manage karega (serverless), aapko infrastructure ka tension nahi sirf data ke logic pe dhyaand do
- **Key-Value & Document Model:** Data ko key-value pairs ya JSON documents mein store karo, flexible schema hai
- **High Performance:** Single-digit millisecond latency, bahut tezi se read/write hota hai
- **Fully Scalable:** Automatic horizontal scaling (partitioning) karta hai, high traffic ko easily handle karta hai

Strengths:

1: AWS-Managed Infrastructure & Serverless

- **AWS-Managed:** Infrastructure, servers, patching sab AWS karta hai; aapko maintenance karne ki need hee nahi hai
- **Serverless Scaling:** On-demand provisioning (resources ko optimize karte hue scale-to-zero); koi manual capacity setup nahi karna padta

2: Key-Value aur Document Data Model

- **Key-Value Store:** Har item ko unique key se store aur access kar sakte ho (jaise dictionary mein key se value milegi)
- **Document Store:** JSON ya map attributes mein complex items store ho sakte hain (jaise ek user profile me name, age, address sab ek item me ho sakta hai)
- **Flexible Schema:** Schema kabhi fix nahi karna padta; har item me alag attributes ho sakte hain jisse agar future me naye attributes add karne ho, asaani se kar sakte ho

3: Performance & Scalability

- **Single-Digit Millisecond Latency:** DynamoDB bohot fast response deta hai (read/write mostly 1-5 millisecond)
- **High Throughput:** Millions of requests per second tak handle kar sakta hai using auto-sharding
- **Automatic Partitioning:** Data khud-b-khud partitions me divide hota hai, load evenly distribute hota hai
- **Scale-Out:** Data aur traffic ke saath tables automatically aur partitions add kar leti hai

4: High Availability (Global Tables)

- **Multi-Region Replication:** Global Tables ke through data multiple AWS regions me real-time replicate hota hai
- **99.999% SLA:** AWS high availability guarantee deta hai (99.999% uptime), downtime almost zero
- **Multi-Active Access:** Alag regions me ek hi time read/write ho sakta hai, data hamesha in sync rahega

5: Security & Backup

- **Built-in Security:** IAM roles/policies se access control (kaun read/write kar sakta), data at-rest encryption aur TLS in-transit encryption by default hoti hai
- **Fine-Grained Access:** Alag-alag users/roles ko specific items/attributes pe permissions de sakte ho
- **Backup & Restore:** Scheduled ya on-demand backup le sakte ho; Point-in-Time Recovery se kisi bhi past moment ka data wapas la sakte ho

6: AWS Ecosystem Integration

- **DynamoDB Streams:** Har data change ka ek stream record banta hai (CCTV footage jaisa)
    - **Stream Records:** Har insert/update/delete ke baad table ka ek record stream me add hota hai (old aur new image dono capture hote hain)
    - **Real-Time Processing:** Streams ko Lambda, Kinesis, ya MSK me pipe karke real-time alerts, analytics aur dashboards update karo
    - **Use Cases:** Audit logs maintain karna, change feeds, real-time analytics pipeline, cache invalidation (jaise item update hone par cache clear karna)

- **Lambda & Event-Driven:** Streams ke through AWS Lambda functions trigger kar sakte ho (jaise naye order pe automatic notification ya payment processing)
- **Integration with Services:** Asani se S3, Kinesis, Elasticsearch, Glue, Athena, aur analytics tools se connect hota hai

Working: DynamoDB AWS ecosystem ke saath easily integrate ho jata hai. Sabse pehla feature hai DynamoDB Streams: jab bhi koi record insert, update, ya delete hota hai, wo event ek ordered stream me chala jata hai. Phir aap is stream ko AWS Lambda ke saath connect kar sakte ho – matlab jaise hi koi naya record banega, us par automatic code chal jayega.

7: TTL (Time to Live)

- **Automatic Expiration:** Har item ko ek TTL (expiry timestamp) assign karo; expiry ke baad item automatically delete ho jayega (jaise Instagram/Snapchat story expire hoti hai)
- **Cleanup of Old Data:** Purana ya unused data apne aap clean ho jata hai, manual cleanup ka jhanjhat nahi
- **Use-Cases:** Session tokens, temporary cache entries, IoT sensor data jise kuch time ke baad irrelevant ho jata hai

8: Query aur Scan Operations

Two ways to read data in DynamoDB – Query and Scan

- **Query:** Partition key (aur optional sort key) se records fetch karo – bohot efficient read operation (jaise library me ek specific shelf se kitaab nikaalna)
- **Scan:** Puri table ko read karo (saare items check karo) – inefficient hai large tables ke liye (jaise library ki saari shelves me search karna)
- **Filters & Projection:** Query/Scan ke results par filter lagao (jaise age > 20) aur sirf required attributes fetch karo (data transfer kam hoga)

10: Secondary Indexes (GSI & LSI)

- **Global Secondary Index (GSI):** Alag partition key + sort key define karo, table se independent ek naya index bana dega; table ke full data pe queries chalata hai
- **Local Secondary Index (LSI):** Base table ka partition key same hi use hota hai, aur ek alag sort key use kar sakte hai; matlab same partition ke items ko alag tarah se sort kar sakte ho -**Use Cases:** Naya query pattern chahiye, toh additional index create karo (jaise phonebook ka alternate index jahan last name se search ho)
- **Costs & Limits:** Index maintain karne ke liye extra write capacity lagti hai; maximum 5 LSIs aur 20 GSIs per table

12: ACID Transactions Support

- **Atomic Transactions:** Multiple items par ek saath all-or-nothing operations (batch update ya multi-item write) kar sakte hai
- **Consistency & Isolation:** Agar transaction me koi step fail ho, toh pura rollback; sab changes ek saath apply hote hain
- **Use Cases:** Financial transfers, inventory updates, leaderboards jahan multiple updates atomic hone chahiye

Weaknesses

1: Design Requirements

- **Access Pattern Pehle Plan Karo:** DynamoDB relational DB jaisa model nahi hai; pehle apne expected queries decide karo, phir schema design karo
- **No Joins/Aggregations:** Table joins support nahi; agar complex relations hain toh application side me multiple queries karke data combine karna padta hai
- **Indexing Complexity:** Indexes flexibility ke liye hai, lekin sahi index design karna challenging hai; galat design se queries slow ya expensive ho sakti hai

2: Scaling Challenges

- **Hot Partitions:** Agar bahut saari requests ek hi partition key ko target karte hai, toh woh partition throttle ho sakta hai (jaise sab log ek hi food stall par line laga dein)
- **Throughput Limits:** Har partition ke paas maximum read/write capacity hoti hai; ek key ke liye bohot high sustained throughput possible nahi
- **Data Skew:** Agar key distribution uneven ho, toh kuch partitions heavy load me hai aur poore table ka performance impact ho sakta hai

3: Item Size & Limits

- **400 KB Item Size Limit:** Ek item ka total size (sare attributes mila ke) 400 KB se zyada nahi ho sakta – bohot large data ko tod ke store karna padega jaise agar large images ya JSON blobs store karna chahte ho, toh unko S3 jaisi service me rakh ke uska reference yahan store kar sakte ho
- **Attribute Limits:** Har item me attribute count aur size par limits hai; bahut badi nested structure ya values store karne par problem aa sakti hai
- **Batch/API Limits:** Single batch write me max 25 items; Scan/Query ke results pagination hoti hai; large operations ke liye multiple calls karni padti hai

4: Complexity & Cost

- **Learning Curve:** DynamoDB ke concepts (partition/sort keys, GSIs/LSIs, capacity modes) RDBMS se alag hai; naye users ko samajhne me time lag sakta hai
- **Capacity Planning:** Provisioned mode me reads/writes ki capacity set karni padti hai (galat estimate se throttling/idle cost); On-Demand mode flexible hai par unpredictable peaks par cost zyada ho sakti hai
- **Debugging & Local Dev:** DynamoDB Local emulator hai lekin production jaisa exact behavior nahi deta (Streams, transactions waghairah me differences); local testing thoda tricky ho sakta hai
- **Cost Spikes:** Agar access patterns inefficient hai (jaise heavy scans ya duplicate indexes), aap zyada read/write units consume karoge aur AWS bill unexpected high aa sakta hai

Use-Case

- **Real-Time Applications:** Chat apps (messages store + notification via Streams), multiplayer games (leaderboard updates) – jahan low latency & real-time updates chahiye
- **Session/Token Store:** Temporary data jaise login session tokens, OTPs, verification codes – TTL lagao aur auto-expire hone do
- **Global Apps:** E-commerce product catalog with users worldwide (Global Tables use karo for fast access), social network profiles & feeds with low-latency access
- **Event-Driven Pipelines:** DynamoDB Streams se Lambda ya Kinesis jod ke notifications, alerts aur analytics workflows banao