Day 33 - 04th Sept 2025

Name : Aravind Kasanagottu

ID : mvsnarav

---

## ✅ Task 01: DynamoDB Architecture

DynamoDB is a fully managed NoSQL database service provided by AWS.

It is designed for \*\*high availability, scalability, and low-latency performance\*\*.

### 🔹 Architecture Overview

1. \*\*Tables\*\* → Store data as collections of items.

2. \*\*Items\*\* → Individual records, like rows in RDBMS.

3. \*\*Attributes\*\* → Fields in the record (columns in RDBMS).

4. \*\*Partition Key\*\* → Determines how data is distributed across storage nodes.

5. \*\*Sort Key (optional)\*\* → Allows range queries within a partition.

6. \*\*Storage Nodes\*\* → Automatically replicated across \*\*multiple AZs\*\* for durability.

7. \*\*DynamoDB Streams\*\* → Capture changes for triggers or replication.

8. \*\*Indexes (GSI & LSI)\*\* → Allow additional query flexibility.

### 🖼️ Architecture Diagram (Text Representation)

┌───────────────────────────┐

│ Application │

│ (API / CLI / SDK) │

└─────────────┬─────────────┘

│

▼

┌───────────────────────────┐

│ DynamoDB Service │

│ ─ Tables │

│ ─ Items │

│ ─ Partition & Sort Keys │

│ ─ Streams & Indexes │

└─────────────┬─────────────┘

│

┌─────────────▼─────────────┐

│ Replication Across AZs │

│ High Availability + DR │

└───────────────────────────┘

---

## ✅ Task 02: Multi-value Sorts / Filters

- \*\*Multi-value Sort\*\* → Use \*\*Sort Key\*\* with range queries (`BETWEEN`, `<`, `>`, etc.).

- \*\*Multi-value Filter\*\* → Apply \*\*FilterExpression\*\* on non-key attributes to get multiple values (like `IN` clause).

### Example Table: `Employee`

| EmpID | JoiningDate | Name | Department |

|-------|-------------|--------|------------|

| 101 | 2019-12-15 | John | HR |

| 101 | 2020-03-10 | Alice | IT |

| 101 | 2020-09-22 | Raj | Finance |

| 101 | 2021-06-11 | Meher | IT |

| 102 | 2020-05-20 | Anita | HR |

---

### 🔹 1. Multi-value Sort (Range on Sort Key)

```bash

aws dynamodb query \

--table-name Employee \

--key-condition-expression "EmpID = :id AND JoiningDate BETWEEN :start AND :end" \

--expression-attribute-values '{

":id": {"N": "101"},

":start": {"S": "2020-01-01"},

":end": {"S": "2021-12-31"}

}'

aws dynamodb scan \

--table-name Employee \

--filter-expression "Department IN (:dept1, :dept2)" \

--expression-attribute-values '{

":dept1": {"S": "HR"},

":dept2": {"S": "IT"}

}'

aws dynamodb query \

--table-name Employee \

--key-condition-expression "EmpID = :id AND JoiningDate BETWEEN :start AND :end" \

--filter-expression "Department = :dept" \

--expression-attribute-values '{

":id": {"N": "101"},

":start": {"S": "2020-01-01"},

":end": {"S": "2021-12-31"},

":dept": {"S": "IT"}

}'

**✅ Task 03: How to Lower DynamoDB Costs?**

1. **Choose On-Demand vs Provisioned Capacity Wisely**
   * Use **On-Demand** for unpredictable workloads.
   * Use **Provisioned with Auto-Scaling** for steady workloads.
2. **Use Filters Carefully**
   * Filters scan all items first → costly.
   * Instead, design keys & indexes to minimize scans.
3. **Efficient Table Design**
   * Use **Composite Keys (Partition + Sort Key)**.
   * Avoid hot partitions by distributing keys evenly.
4. **Use Secondary Indexes (GSI/LSI) only when needed**
   * Extra indexes = extra cost.
5. **Use DAX (DynamoDB Accelerator)**
   * Cache frequently accessed items → reduces read costs.
6. **Enable TTL (Time To Live)**
   * Automatically delete old/unused items → saves storage costs.
7. **Backup Strategy**
   * Use **on-demand backups** only when needed instead of frequent snapshots.
8. **Optimize Queries**
   * Query by Partition Key instead of full scans.
   * Retrieve only needed attributes using --projection-expression.

#!/bin/bash

# ================================

# \*\*1. Create Employee Table\*\*

# ================================

aws dynamodb create-table \

--table-name Employee \

--attribute-definitions \

AttributeName=EmpID,AttributeType=N \

AttributeName=JoiningDate,AttributeType=S \

--key-schema \

AttributeName=EmpID,KeyType=HASH \

AttributeName=JoiningDate,KeyType=RANGE \

--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

echo "✅ \*\*Employee table created.\*\*"

# \*\*Wait until the table is active\*\*

aws dynamodb wait table-exists --table-name Employee

echo "✅ \*\*Employee table is now ACTIVE.\*\*"

# ================================

# \*\*2. Insert Sample Data\*\*

# ================================

aws dynamodb put-item --table-name Employee --item '{

"EmpID": {"N": "101"},

"JoiningDate": {"S": "2019-12-15"},

"Name": {"S": "John"},

"Department": {"S": "HR"}

}'

aws dynamodb put-item --table-name Employee --item '{

"EmpID": {"N": "101"},

"JoiningDate": {"S": "2020-03-10"},

"Name": {"S": "Alice"},

"Department": {"S": "IT"}

}'

aws dynamodb put-item --table-name Employee --item '{

"EmpID": {"N": "101"},

"JoiningDate": {"S": "2020-09-22"},

"Name": {"S": "Raj"},

"Department": {"S": "Finance"}

}'

aws dynamodb put-item --table-name Employee --item '{

"EmpID": {"N": "101"},

"JoiningDate": {"S": "2021-06-11"},

"Name": {"S": "Meher"},

"Department": {"S": "IT"}

}'

aws dynamodb put-item --table-name Employee --item '{

"EmpID": {"N": "102"},

"JoiningDate": {"S": "2020-05-20"},

"Name": {"S": "Anita"},

"Department": {"S": "HR"}

}'

echo "✅ \*\*Sample records inserted.\*\*"

# ================================

# \*\*3. Multi-value Sort Query\*\*

# ================================

echo "🔍 \*\*Query: Employees 101 joined between 2020-01-01 and 2021-12-31\*\*"

aws dynamodb query \

--table-name Employee \

--key-condition-expression "EmpID = :id AND JoiningDate BETWEEN :start AND :end" \

--expression-attribute-values '{

":id": {"N": "101"},

":start": {"S": "2020-01-01"},

":end": {"S": "2021-12-31"}

}'

# ================================

# \*\*4. Multi-value Filter Query\*\*

# ================================

echo "🔍 \*\*Filter: Employees in HR or IT\*\*"

aws dynamodb scan \

--table-name Employee \

--filter-expression "Department IN (:dept1, :dept2)" \

--expression-attribute-values '{

":dept1": {"S": "HR"},

":dept2": {"S": "IT"}

}'

# ================================

# \*\*5. Sort + Filter Query\*\*

# ================================

echo "🔍 \*\*Query: Employees 101 in IT dept joined 2020-2021\*\*"

aws dynamodb query \

--table-name Employee \

--key-condition-expression "EmpID = :id AND JoiningDate BETWEEN :start AND :end" \

--filter-expression "Department = :dept" \

--expression-attribute-values '{

":id": {"N": "101"},

":start": {"S": "2020-01-01"},

":end": {"S": "2021-12-31"},

":dept": {"S": "IT"}

}'