DevOps

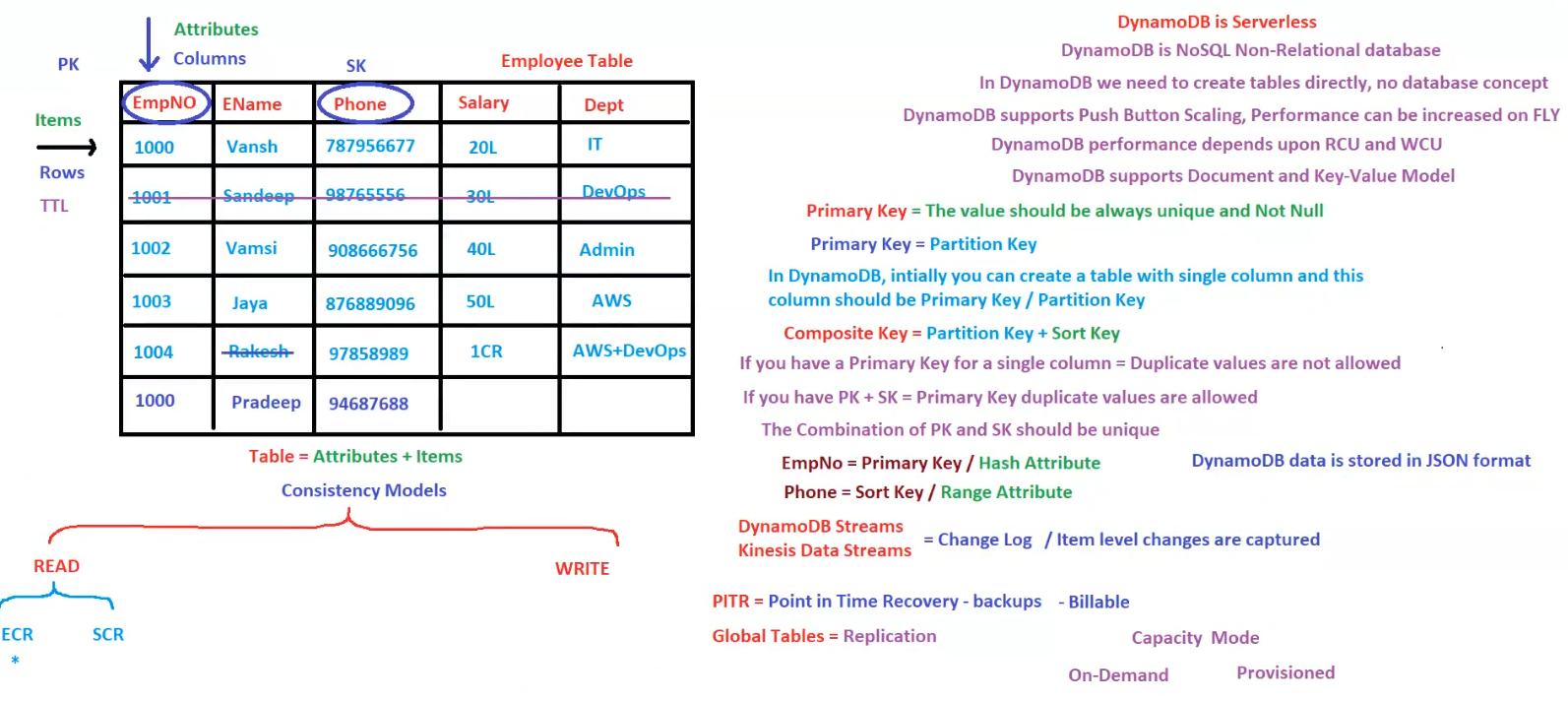
[Dynamo DB 2](#_Toc193689807)

[Amazon DynamoDB - Serverless NoSQL Database 2](#_Toc193689808)

[DynamoDB - Indexing & Capacity Units 4](#_Toc193689809)

# Dynamo DB

## Amazon DynamoDB - Serverless NoSQL Database



**Amazon DynamoDB** is a **serverless, NoSQL, non-relational database** designed for **high-performance, scalability, and low-latency** workloads.

**1. DynamoDB Table Structure**

* **Table** = Collection of **Attributes + Items (Rows)**.
* **Attributes (Columns)**: Fields in the table.
* **Items (Rows)**: Individual records in the table.
* **TTL (Time-To-Live)**: Used to automatically delete expired records.

**Example: Employee Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EmpNO (PK)** | **EName** | **Phone (SK)** | **Salary** | **Dept** |
| 1000 | Vansh | 787956677 | 20L | IT |
| 1001 | Sandeep | 987655556 | 30L | DevOps |
| 1002 | Vamsi | 908667756 | 40L | Admin |
| 1003 | Jaya | 876889096 | 50L | AWS |
| 1004 | Rakesh | 97858989 | 1CR | AWS+DevOps |
| 1000 | Pradeep | 94687688 | - | - |

**2. Keys in DynamoDB**

**A. Primary Key**

* **Must always be unique and Not Null**.
* **Types**:
  1. **Partition Key (PK)** (Single-column Primary Key)
     + Must be unique.
     + Duplicate values **not allowed**.
  2. **Composite Key** (Partition Key + Sort Key)
     + **PK + SK together must be unique**.
     + PK **can have duplicate values** if SK is different.

**B. Key Attributes in Example**

* **EmpNo = Partition Key (PK) / Hash Attribute**.
* **Phone = Sort Key (SK) / Range Attribute**.

**3. Data Storage Format**

✅ **DynamoDB stores data in JSON format**.

**4. Read & Write Consistency Models**

**A. Read Consistency**

* **ECR (Eventual Consistency Read)**: Faster but may return stale data.
* **SCR (Strong Consistency Read)**: Always returns the most up-to-date data.

**B. Write Operations**

* DynamoDB **automatically replicates data across multiple availability zones**.

**5. Performance & Scaling**

✅ **Push-Button Scaling**: Performance can be increased **on the fly**.  
✅ **Capacity Mode**:

* **On-Demand**: Pay per request.
* **Provisioned**: Set Read/Write Capacity Units (RCU/WCU).

**6. Advanced Features**

**A. DynamoDB Streams**

* **Tracks item-level changes**.
* Can be integrated with **Kinesis Data Streams**.

**B. PITR (Point-in-Time Recovery)**

* **Provides backups** for data recovery.
* **Billable feature**.

**C. Global Tables**

* **Replication** of DynamoDB tables across multiple AWS Regions.

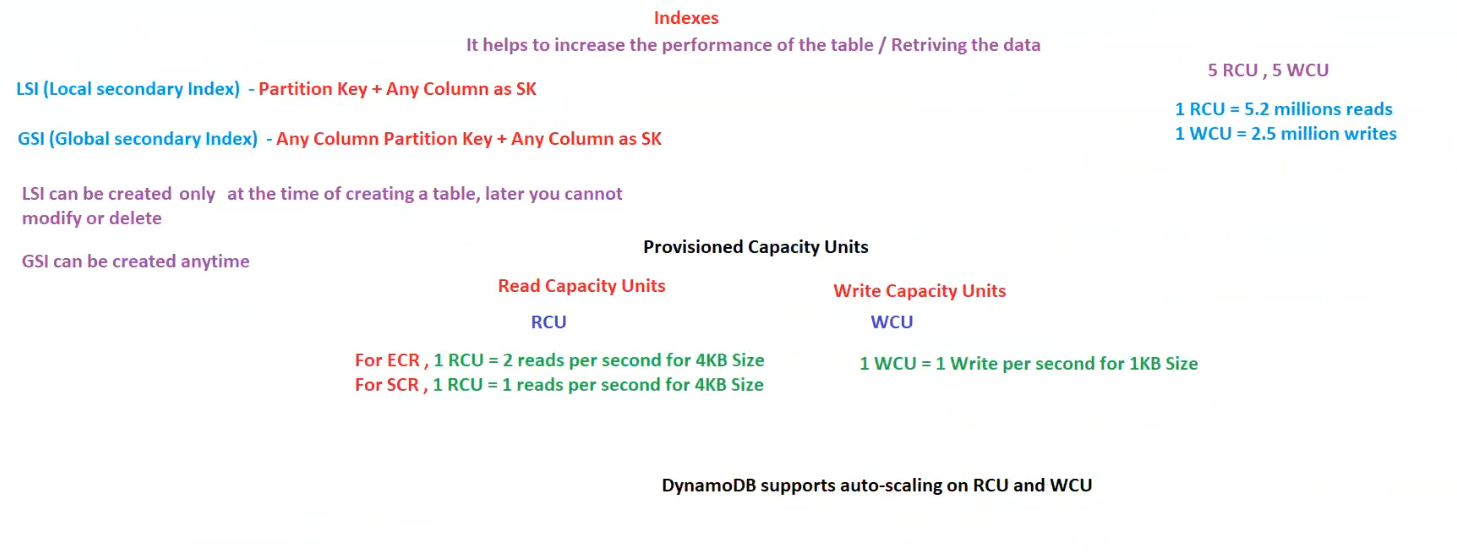
**7. Summary & Takeaways**

✅ **DynamoDB is serverless and scalable**.  
✅ **Supports Key-Value & Document models**.  
✅ **Primary Key is always unique** (Single-column PK or Composite PK+SK).  
✅ **Supports Strong and Eventual Consistency Reads**.  
✅ **DynamoDB Streams & Global Tables** enable real-time data replication.

**Conclusion**

AWS **DynamoDB** is a **fully managed NoSQL database** that provides **high availability, scalability, and flexibility**, making it ideal for **real-time applications, IoT, and event-driven architectures**.

## DynamoDB - Indexing & Capacity Units



**Amazon DynamoDB - Indexing & Capacity Units**

**DynamoDB** is a **serverless NoSQL database** that supports **indexes and capacity provisioning** for **performance optimization and scalability**.

**1. Indexes in DynamoDB**

Indexes **improve query performance** by allowing fast data retrieval without scanning the entire table.

**A. Local Secondary Index (LSI)**

* **Structure**: **Partition Key + Any Column as Sort Key (SK)**.
* **Limitation**:
  + **LSI must be created at the time of table creation**.
  + Cannot be **modified or deleted** later.
* **Use Case**: When queries **require sorting** on a different column than the primary Sort Key.

**B. Global Secondary Index (GSI)**

* **Structure**: **Any Column as Partition Key + Any Column as Sort Key**.
* **Advantage**:
  + **GSI can be created anytime** (unlike LSI).
  + Provides **flexibility** to query data based on different attributes.
* **Use Case**: When **querying data using attributes other than the primary key**.

**2. Provisioned Capacity Units**

DynamoDB provides **Provisioned Capacity Mode**, where Read and Write operations consume capacity units.

**A. Read Capacity Units (RCU)**

* **ECR (Eventually Consistent Read)**:
  + **1 RCU = 2 reads per second for 4KB data size**.
* **SCR (Strongly Consistent Read)**:
  + **1 RCU = 1 read per second for 4KB data size**.

**B. Write Capacity Units (WCU)**

* **1 WCU = 1 write per second for 1KB data size**.

**3. Auto-Scaling & High Throughput**

✅ **DynamoDB supports Auto-Scaling** for RCU and WCU.  
✅ **High Read/Write Capacity**:

* **1 RCU can handle 5.2 million reads per month**.
* **1 WCU can handle 2.5 million writes per month**.

**4. Summary & Takeaways**

✅ **LSI is created at table creation & cannot be modified later**.  
✅ **GSI can be created anytime & allows flexible queries**.  
✅ **Provisioned Capacity Mode allows fine-tuned read/write scaling**.  
✅ **Auto-scaling ensures cost optimization & performance stability**.

**Conclusion**

DynamoDB indexing and capacity management enable **high-performance, cost-effective, and scalable applications**. Choosing **LSI vs GSI** and **ECR vs SCR** depends on **query patterns and workload**.