Day 33

Employee ID: 201933938

Login ID: iamasif

Name: Shaik Asif

As we have learnt about the DynamoDb Archietecture..

Task 01👍

What do you understand by DynamoDb Archietecture.. ? explain using a diagram

It is a **fully managed NoSQL database** that delivers fast and predictable performance at any scale. Its architecture is built on **partitioning, replication, and request routing** to provide **high availability, durability, and scalability**.

Key components:

**Tables** – Logical containers for data.

**Items** – Individual records (like rows).

**Attributes** – Fields inside each item.

**Primary Key** – Partition key (or Partition + Sort key) decides how data is stored.

**Partitions** – Physical storage units across multiple servers.

**Request Router** – Routes client requests to the correct partition.

**Replication** – Data automatically replicated across multiple Availability Zones (AZs).

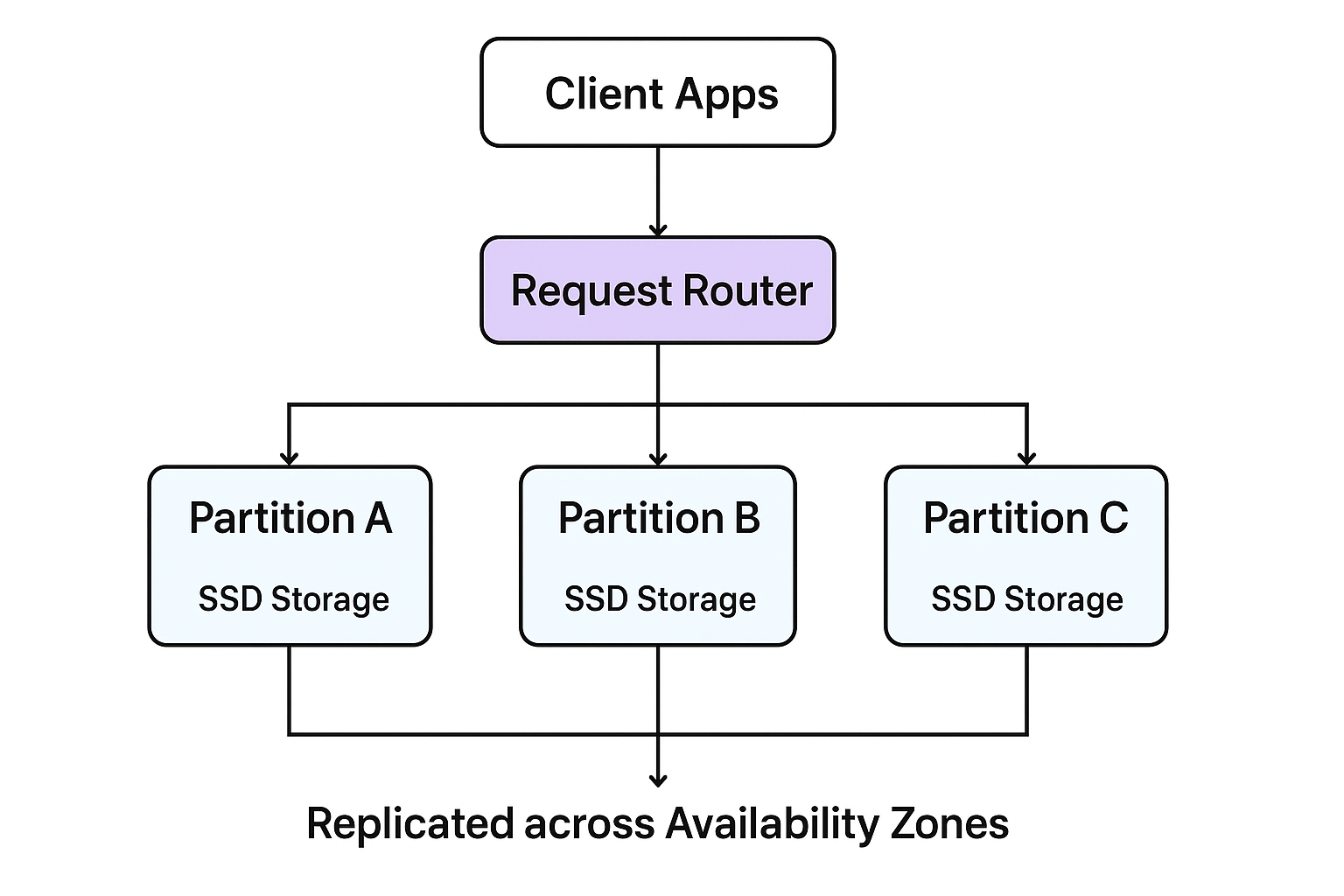
**Secondary Indexes** – GSIs and LSIs allow flexible queries.

works:

When an app sends a request, the **request router** uses the **partition key’s hash value** to find the correct partition.

Data is stored on **SSD-backed partitions**, replicated across **3 AZs** for durability.

Scaling is automatic: DynamoDB adds partitions when storage or throughput demand grows.



**Scalability**: Data automatically spreads across partitions.

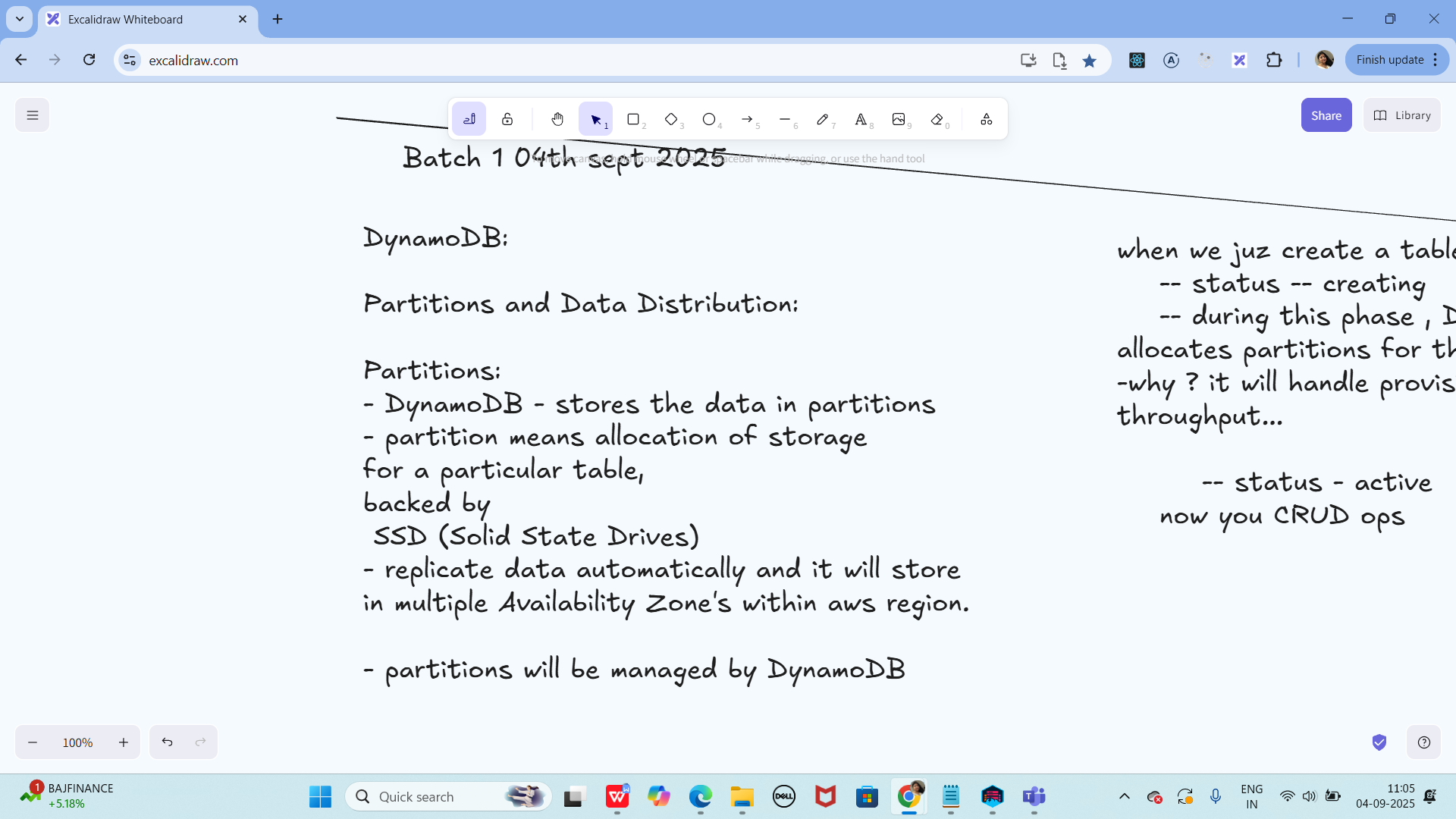
**Availability**: Replication across multiple AZs ensures uptime.

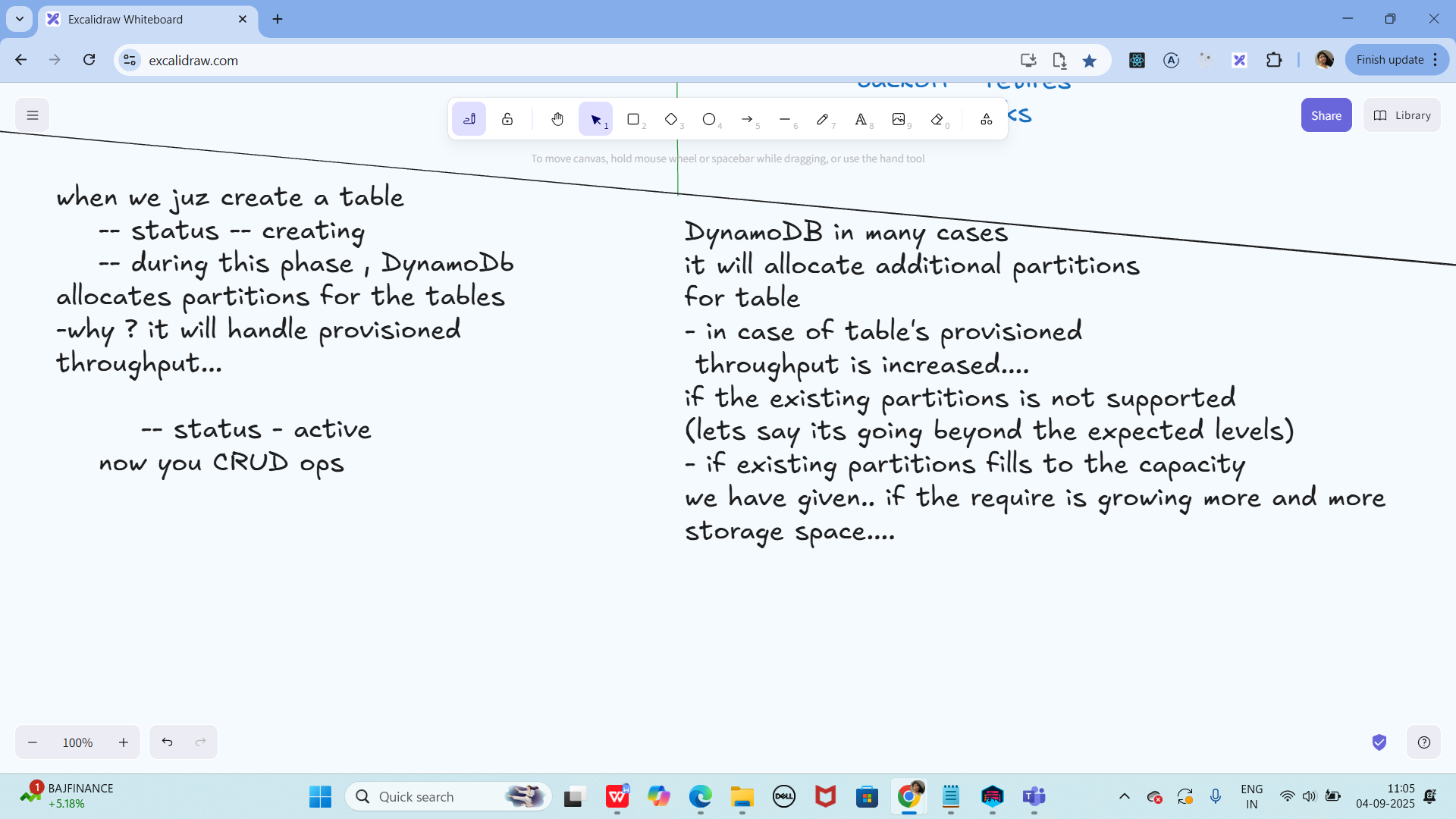
**Performance**: Single-digit millisecond latency.

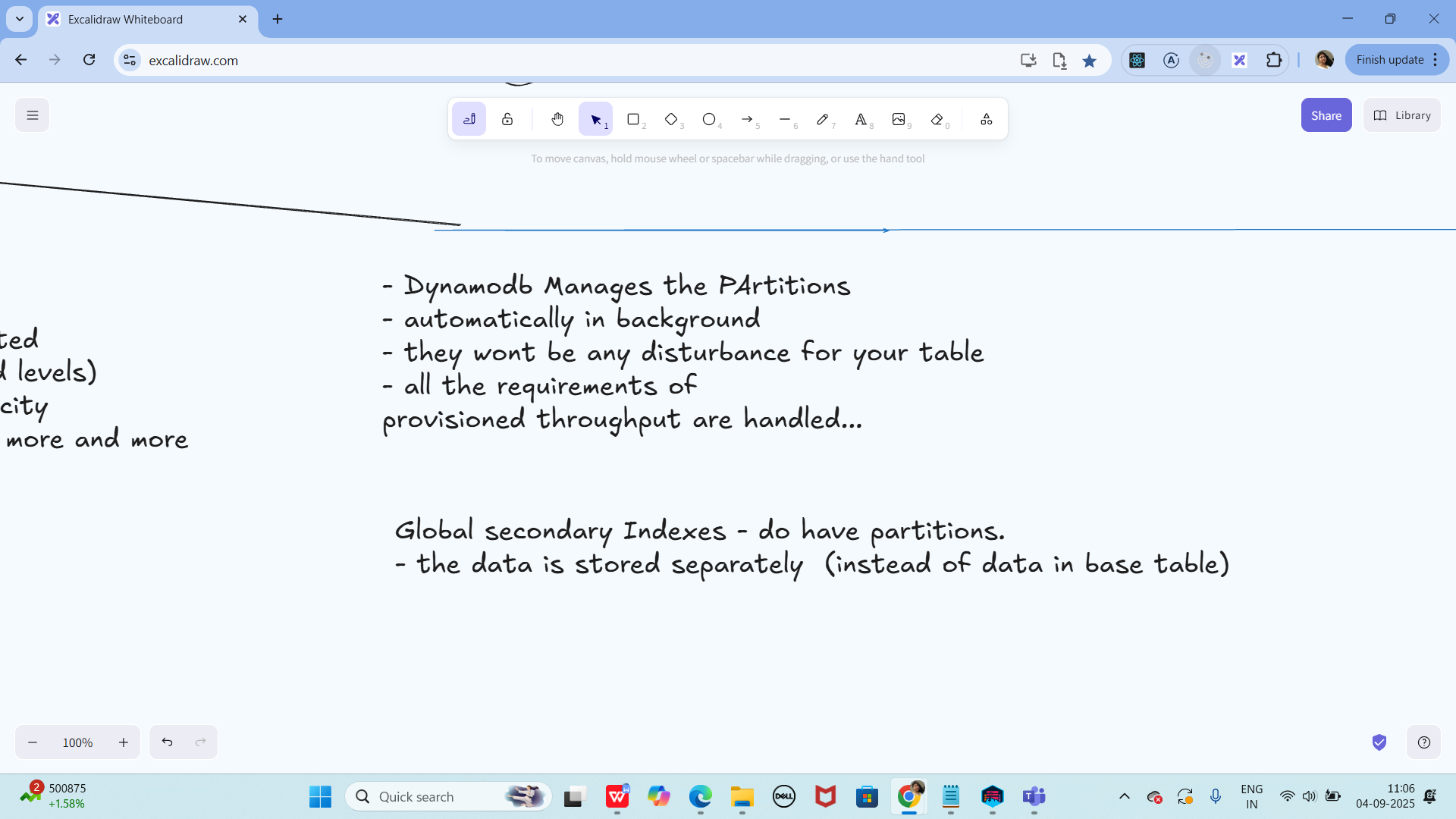
**Flexibility**: Secondary indexes for querying.

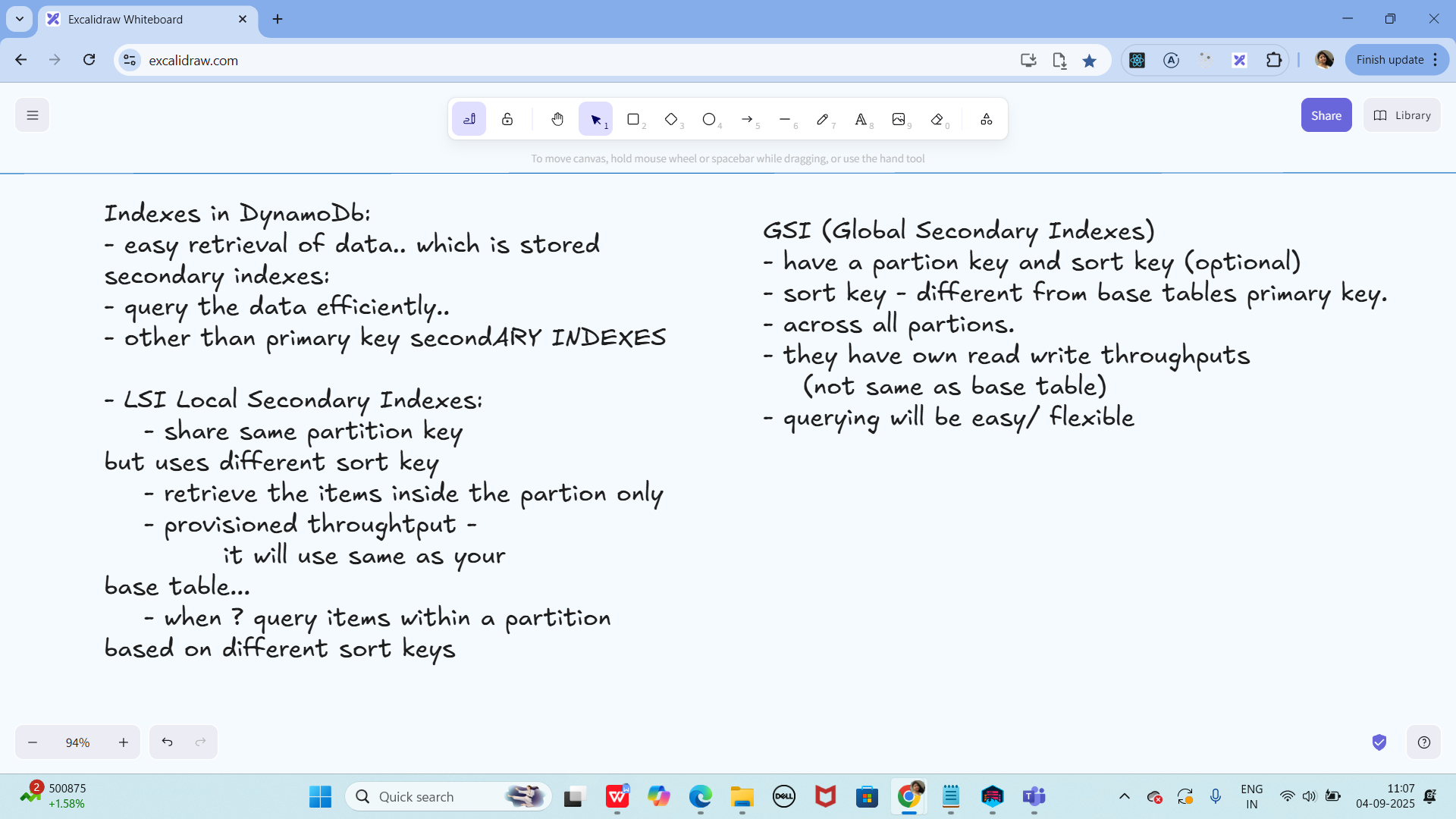
**Security**: IAM, encryption at rest and in transit.

DynamoDB’s architecture is based on **partitioning data, routing requests efficiently, and replicating across multiple AZs** to ensure **scalability, performance, and reliability**.









Task 02:

What are multivalue sorts or filters..? Explain briefly..

**1. Multi-value Sorts**

In DynamoDB, the sort key is used to arrange items within the same partition key.

* A sort key can have multiple values for the same partition key, which means you can store multiple related items under the same partition.
* This allows you to query and sort items by different attributes.

**ex:**

PK (UserID) SK (OrderDate) OrderID

USER#123 2025-09-01 ORDER#1

USER#123 2025-09-02 ORDER#2

USER#123 2025-09-05 ORDER#3

# **2. Filters**

Filters are conditions applied **after DynamoDB reads the data**.

* A **FilterExpression** refines results returned by a query or scan.
* Important: Filters don’t reduce the amount of data read; they only reduce the **results returned**.

**ex:**

Query: Get all orders for USER#123

Filter: Status = 'Shipped'

**Multi-value sorts**: Multiple items with the same partition key but different sort keys → allows querying by ranges, prefixes, or ordering.

**Filters**: Conditions applied after fetching data → refine the results, but do not reduce read capacity usage.

Task 03:

How do you lower the cost of DynomaDB..

## **1. Choose the Right Capacity Mode**

* **On-Demand Mode**
  + Pay per request. Best for unpredictable workloads.
  + Avoids overprovisioning, but costly for high steady traffic.
* **Provisioned Mode (with Auto Scaling)**
  + Set baseline Read/Write Capacity Units (RCUs/WCUs).
  + Auto Scaling adjusts when traffic increases.
  + Cheaper for steady, predictable workloads.

## **2. Use Efficient Data Modeling**

* **Design partition keys carefully** → distribute traffic evenly, avoid hot partitions.
* **Single Table Design** → reduces need for multiple scans across tables.
* **Sparse Indexes** → only index items that need to be queried, not the whole dataset.
* **Avoid large attributes** → store big files in **S3**, and keep only metadata in DynamoDB.

## **3. Minimize Read/Write Capacity Usage**

* **Batch Operations** → use BatchGetItem or BatchWriteItem to reduce request overhead.
* **Projection Expressions** → fetch only required attributes, not entire items.
* **Filters vs Queries** → design queries to use **partition & sort keys** instead of relying on filters (filters still consume RCU/WCU).
* **DAX (DynamoDB Accelerator)** → cache hot items, reduce read costs.

## **4. Optimize Index Usage**

* Global Secondary Indexes (GSI) and Local Secondary Indexes (LSI) cost extra.
* Keep indexes **only for essential access patterns**.
* Project only the attributes you really need (KEYS\_ONLY or INCLUDE projection instead of ALL).

## **5. Use TTL (Time to Live)**

* Enable **TTL** to automatically delete old/unused items.
* Saves storage costs.
* Example: Session tokens, logs, temporary data.

**6. Use DynamoDB Streams Wisely**

* Streams cost extra when you process them with Lambda.
* Use them only when needed (replication, event-driven workflows).

## **7. Monitor & Optimize**

* Use **CloudWatch metrics** and **Contributor Insights** to find hot keys and wasted capacity.
* Adjust **RCU/WCU** or switch capacity modes based on usage trends.
* Use **Reserved Capacity** if you have long-term, predictable workloads (up to 75% cheaper).

=======================================================================================================================================================================

Info box:

Excalidraw updated at 11.05

<https://excalidraw.com/#json=QkKS8kaMr61IO2OuVmyy5,UGs5LYSkL0C5LM1eFOTffQ>

Excalidraw updated at 13.01

<https://excalidraw.com/#json=M67BNiIL9iNISXSRzaZpW,mfnzHw8ap2eEWQ57QEM0_w>

Ways to reduce code - dynamodb — doc in Module 4 Dynamodb folder

<https://drive.google.com/drive/folders/19jURMCjQRnl-j56VLupBhuCgkSr7oLrf?usp=sharing>

Amazon AWS dynamoDb doc for reference…

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide>