



# NoSQL and Other Database Solutions in AWS





# Today's Takeaways

- ▶ Redshift
- ▶ ElastiCache
- ▶ DynamoDB


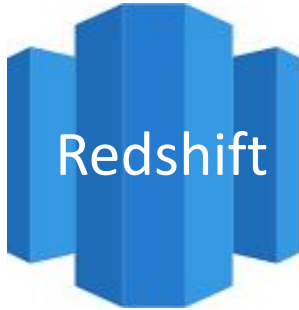


2

# Amazon Redshift

# Amazon Redshift



Database	Data Warehouse
 RDS	 Redshift
OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Used for data storing and managing	Used for data analyzing

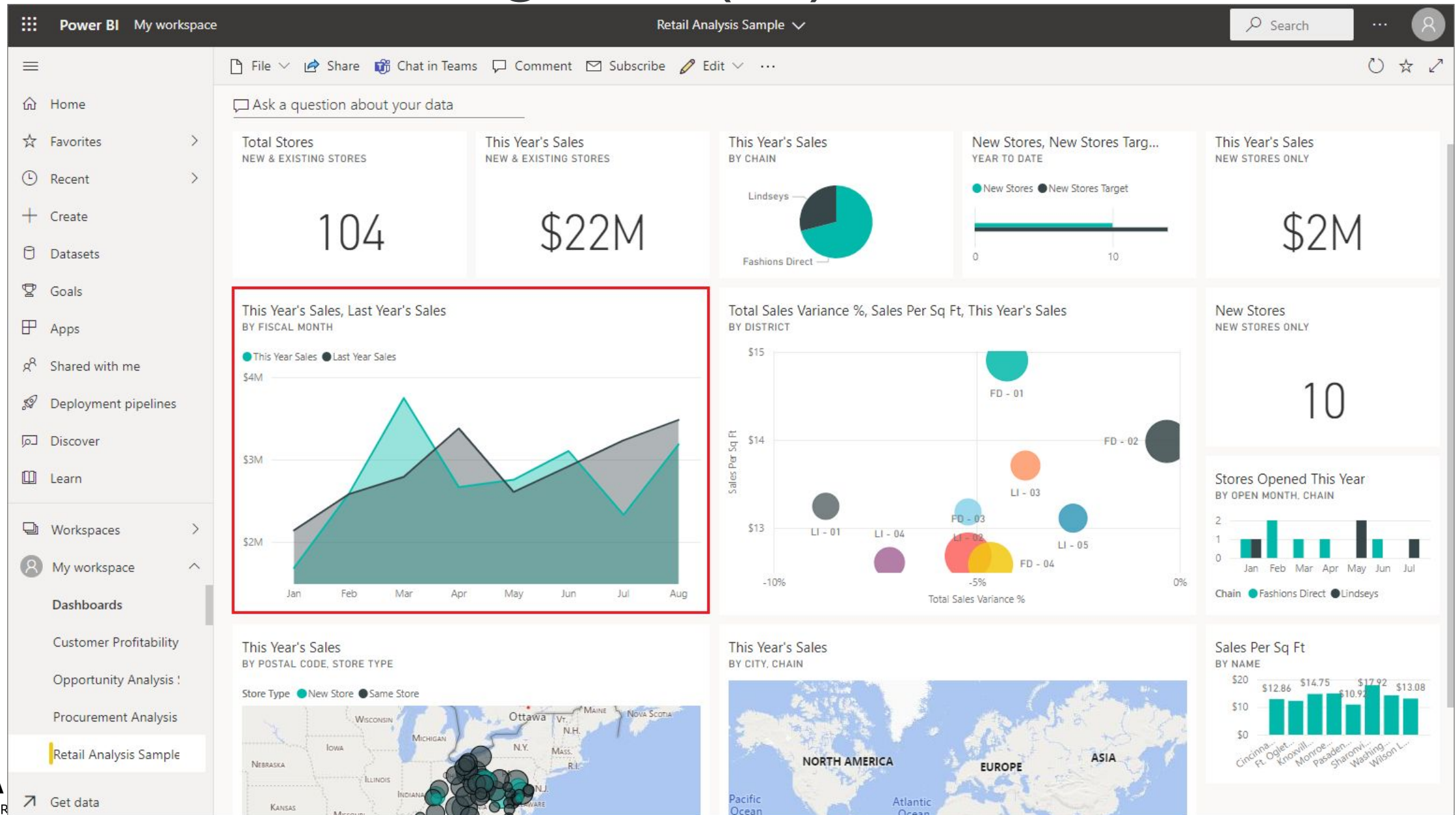
- Since the **analyzing process causes an extra workload on database** we prefer to use data warehouse
- Amazon Redshift is a fully managed, cloud-based, petabyte-scale **data warehouse** service by Amazon Web Services (AWS).
- Amazon Redshift is an efficient solution to **collect and store** all your data to **analyze**.

# Amazon Redshift



- Amazon Redshift is a fully managed, **petabyte-scale data warehouse** service in cloud. It's a **very large relational database** traditionally used in big data applications.
- **Incredibly** big - it can hold up **16 PB** of data. This means you **don't have to split** up your large dataset.
- This database is **relational**. You use your **standard SQL** and business intelligence(BI) tools to interact with it.
- While Redshift is fantastic tool for BI applications, It's **not a replacement** for standard RDS databases.

# Business intelligence(BI) tools



# Exam Tips



- Amazon Redshift is **not a high available** service, meaning it only comes online **at one AZ**.
- If you want it in **multiple AZs**, you're gonna have to create **multiple copies**.
- Keep in mind it's for **BI applications**, it's **relational**, and it can store up to **16 PB** of data

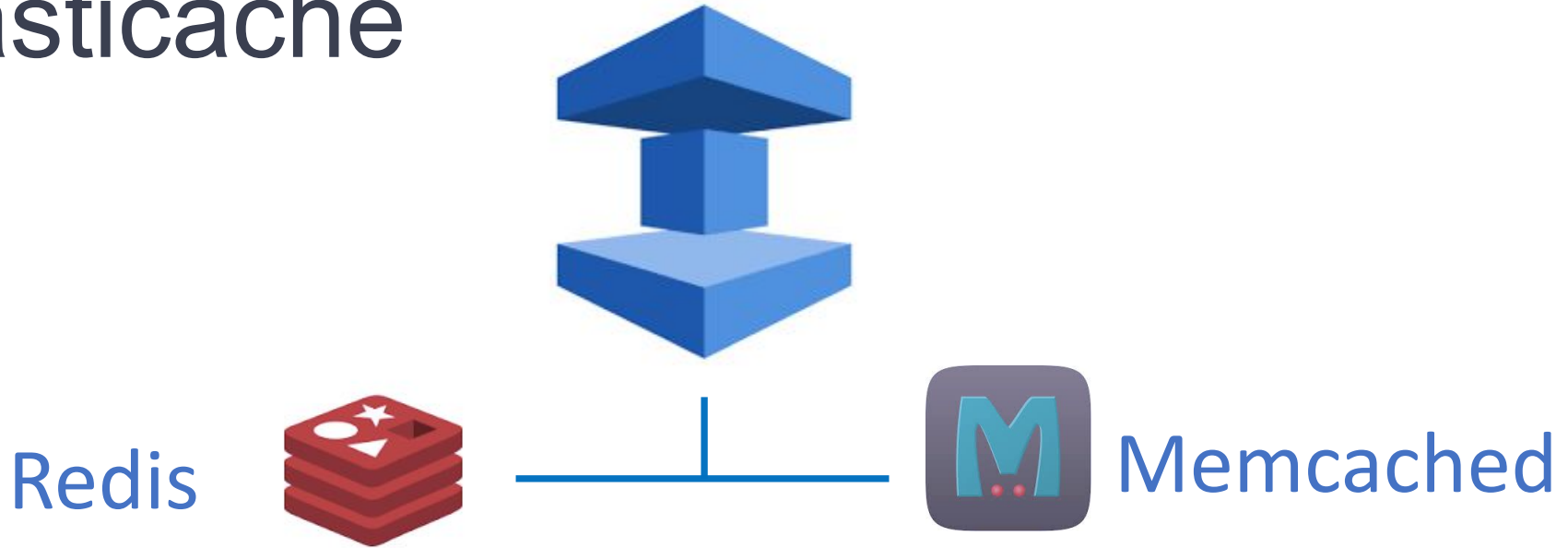


3

# AWS ElastiCache



# AWS ElastiCache



- ElastiCache is an **In-Memory Cache service of AWS**.
- In-Memory Cache is a **temporary and fast** storage component. These components are used to reduce the workload of the main data storage device such as a database.
- AWS offers Redis and Memcached in-memory cache option which are popular in market.

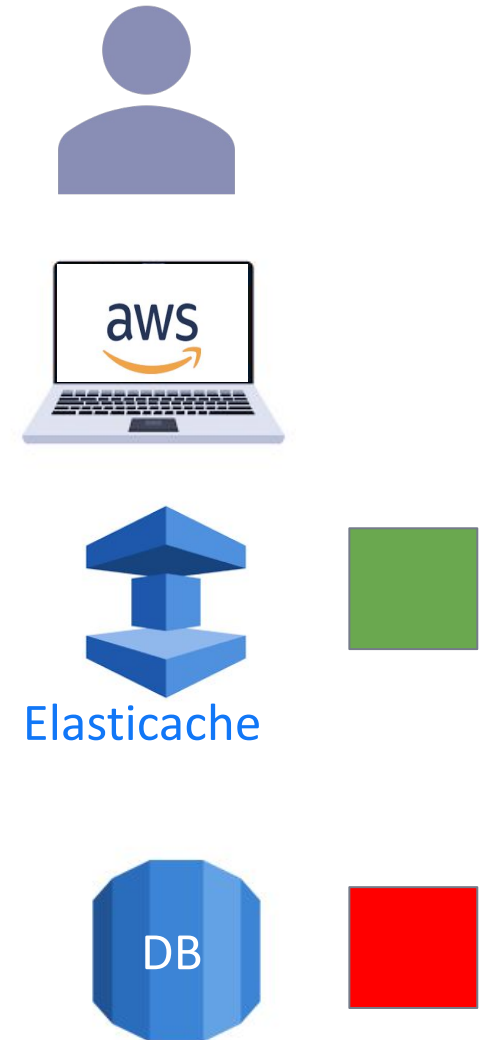
# AWS ElastiCache



After ElastiCache - **First Query**

After ElastiCache - **Second Query**

Before ElastiCache



# Redis



# Memcached

Sub-millisecond latency	+	+	Sub-millisecond latency
User friendly syntax	+	+	User friendly syntax
It supports many different programming languages C, C++, java, python, etc.	+	+	It supports many different programming languages C, C++, java, python, etc.
Redis supports strings ,lists, sets, sorted sets, hashes, bit arrays, and hyperloglogs	+	-	Memcached supports only strings
It doesn't support multithreaded architecture	-	+	It supports multithreaded architecture. It means that it has multiple processing cores. This allows you to handle multiple operation.
It supports Snapshot	+	-	It doesn't support Snapshot
It supports Replica	+	-	It doesn't support Replica

# AWS ElastiCache



- ElastiCache gives you a bit more flexibility. It can front just about any database, but really **excels** being placed in **front of RDS**.
- Redis can be more than just a cache - it can be a standalone database as well (**More Than a Cache**.)
- **Memcached**    >>    No Failover or Multi-AZ supported
- **Redis**            >>    Failover or Multi-AZ supported



1

# Amazon DynamoDB

# DynamoDB

## What is DynamoDB?



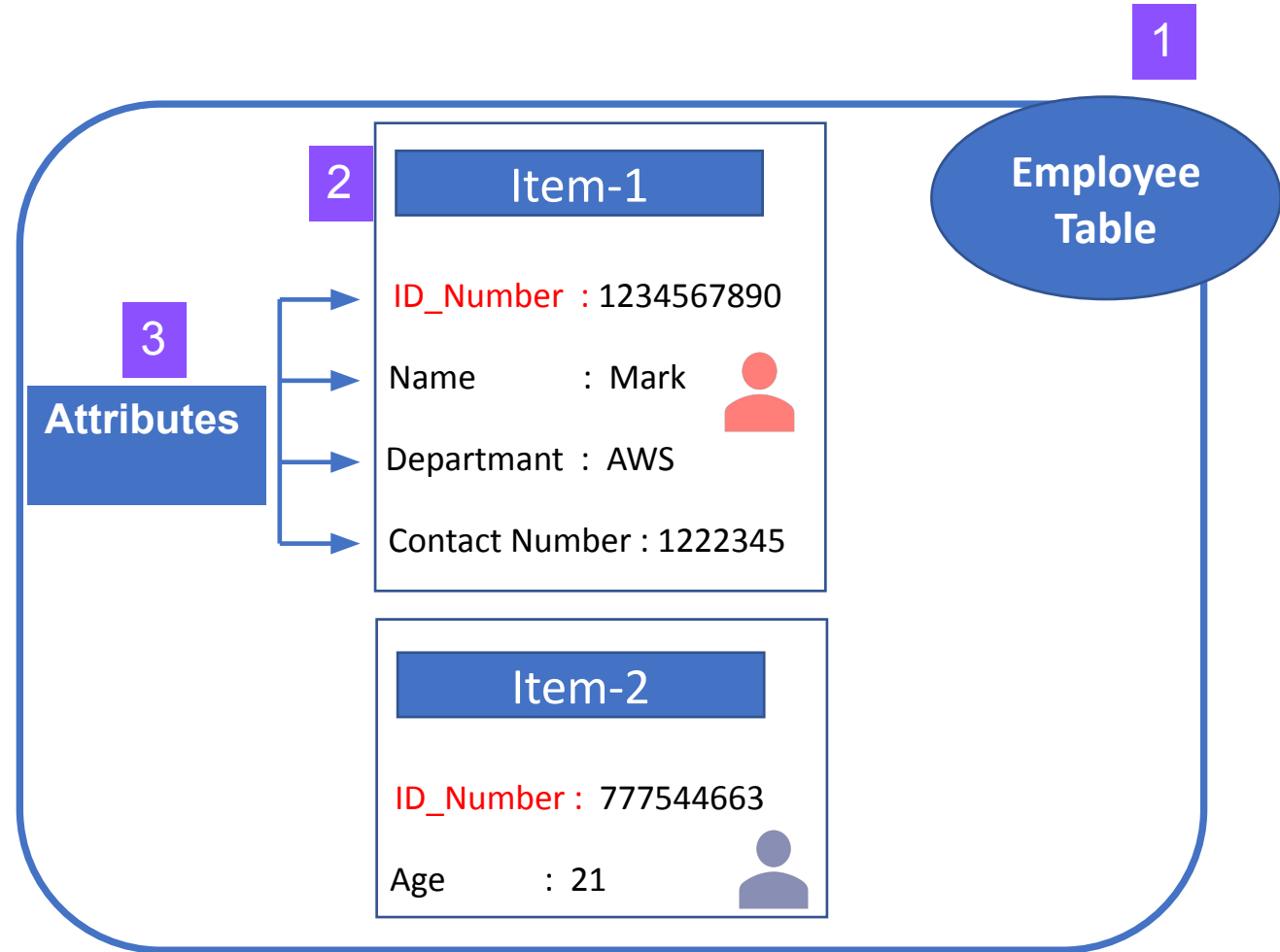
**amazon**  
DynamoDB

- Amazon DynamoDB is a **NoSQL database** service
- Unlike RDS, you don't need to stick pre-determined schema. Instead of Schema, DynamoDB uses **flexible tables**.
- Amazon DynamoDB is a **fully-managed** database. (also **Serverless**)
- DynamoDB doesn't **have Join function**.

# DynamoDB

## Structure of DynamoDB?

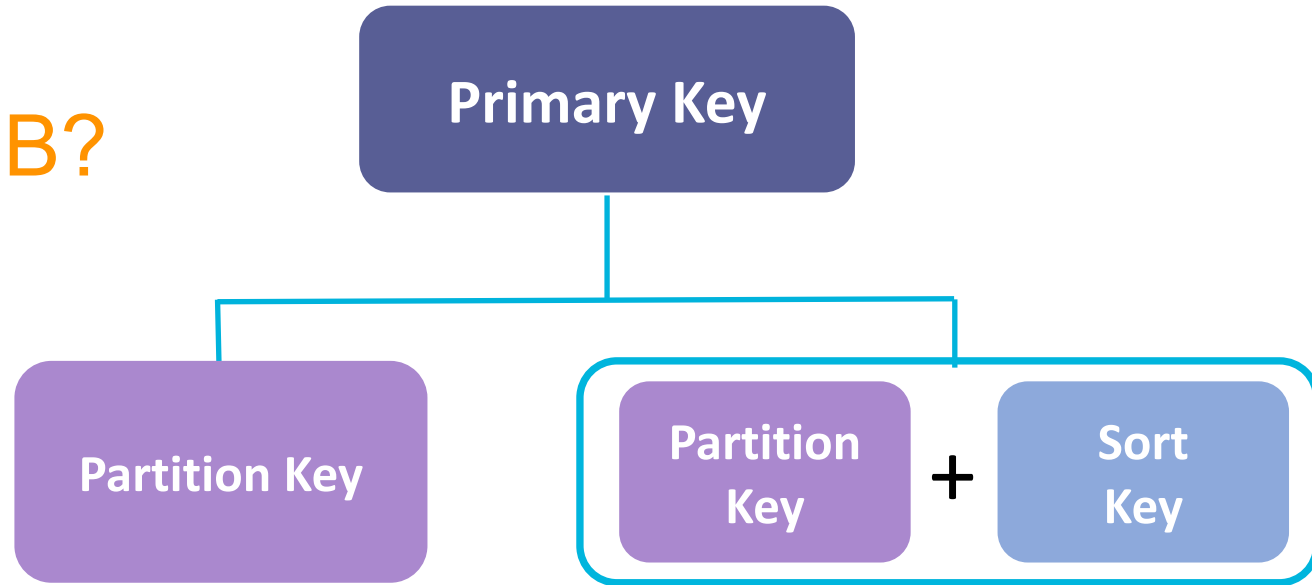
- 1- **Table** is a collection of data.
- 2- Each table consist of **items**. In the Picture, **item represents a person**.
- 3- **Attributes** are specific feature of the items.



Unlike RDS, you can enter different **attributes** for each people.

# DynamoDB

## Structure of DynamoDB?



DynamoDB uses **Primary Keys** to **uniquely identify each item** in a table. When you create a table, in addition to the table name, you must specify the primary key of the table.

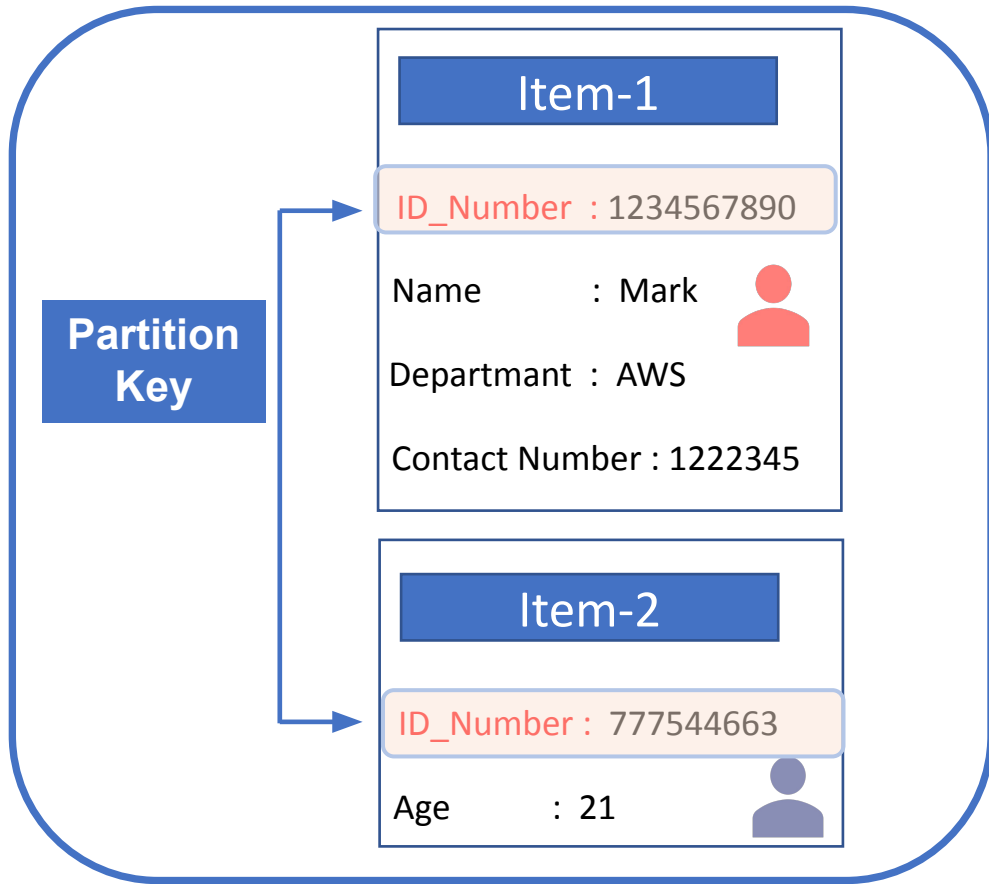
**The sort key** is used to sort and order **items** in a partition. Multiple items with the same partition key value are feasible, but they should have different sort key values. This means you can have multiple items with the same partition keys, but the sort key can not be the same.

There are two different kinds of Primary Key model : **Partition Key** and **Partition Key & Sort Key**.

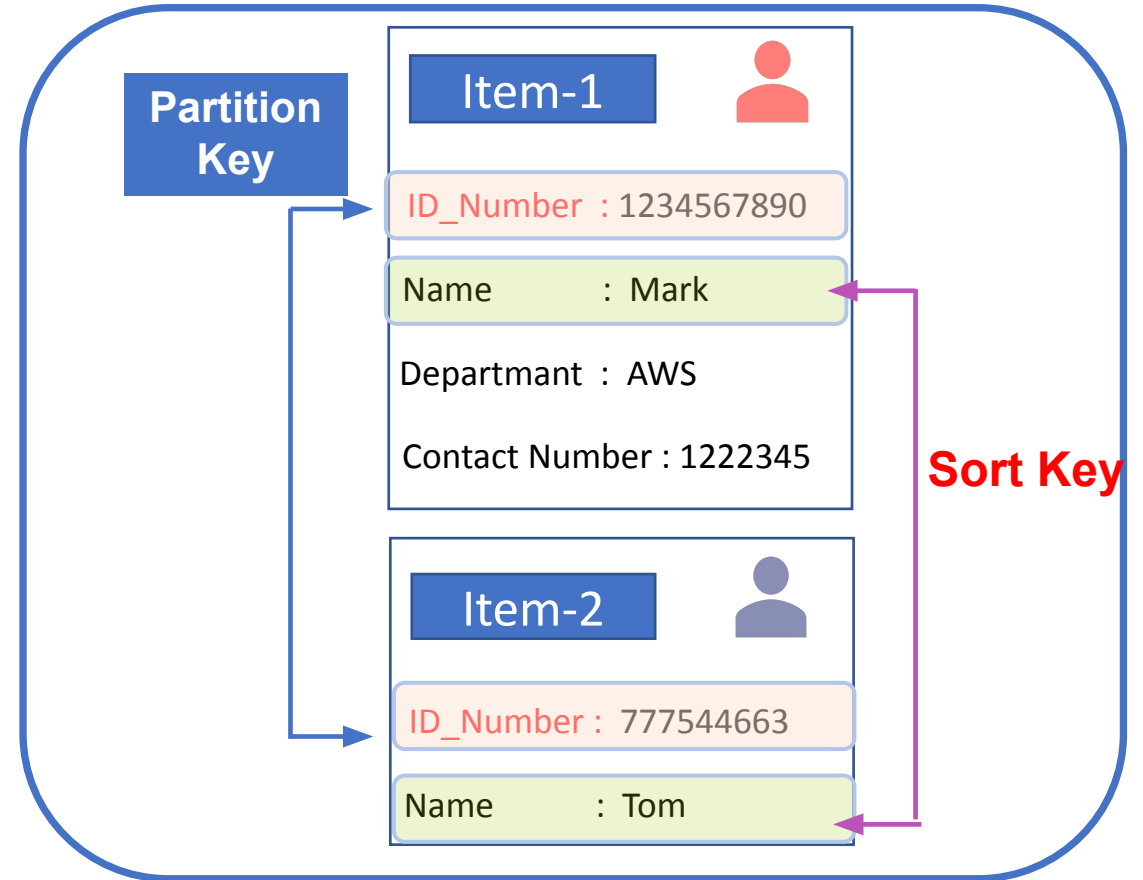


# DynamoDB

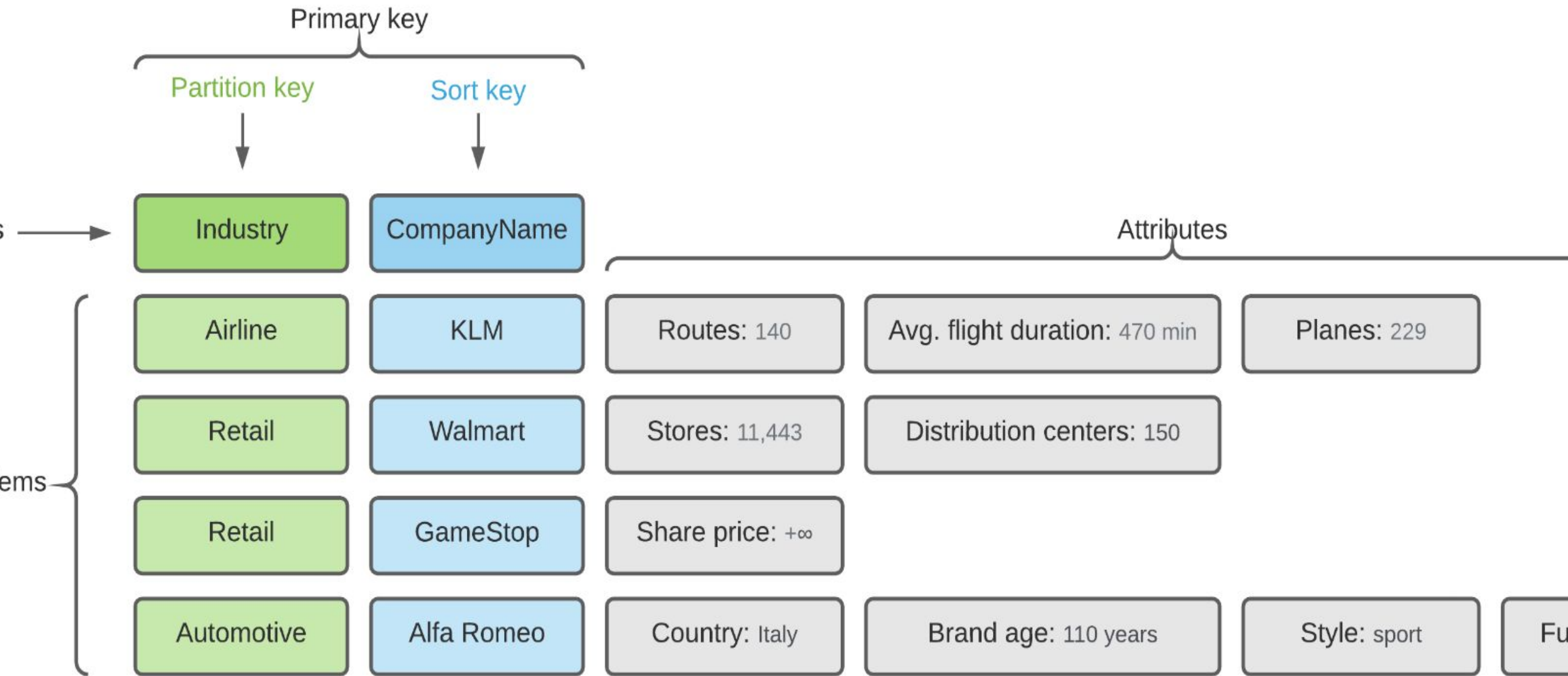
## Structure of DynamoDB?



Partition Key



Partition Key&Sort Key



# DynamoDB

## Capacity Modes

### Eventually Consistent Reads

- When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data.
- If you repeat your read request after a short time, the response should return the latest data.

### Strongly Consistent Reads

- When you request a strongly consistent read, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful.

# DynamoDB

## Capacity Modes

### DynamoDB RCU

- Read Capacity tells us how much data can be read from a DynamoDB table. Read Capacity is measured in RCUs. (Read Capacity Unit)

DynamoDB read requests can be either strongly consistent, eventually consistent, or transactional.

- An *eventually consistent* read request of an item up to 8 KB requires **one read** request unit.
- A *strongly consistent* read request of an item up to 8 KB requires **two read** request unit.
- A *transactional* read request of an item up to 8 KB requires **four read** request units.

### DynamoDB Write Capacity

- DynamoDB Write Capacity tells us how much data can be written to a DynamoDB table. Write Capacity is measured in WCUs.
- WCU or "Write Capacity Unit" represents one write per second, for an item up to 1 KB in size.

# DynamoDB

## Secondary Indexes

- In a relational database, an index is a data structure that lets you perform fast queries on different columns in a table. You can use the CREATE INDEX SQL statement to add an index to an existing table, specifying the columns to be indexed.
- You can create one or more secondary indexes on a dynamodb table. A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key. DynamoDB doesn't require that you use indexes, but they give your applications more flexibility when querying your data.
- After you create a secondary index on a table, you can read data from the index in much the same way as you do from the table.

DynamoDB supports two kinds of indexes:

- Global secondary index – An index with a partition key and sort key that can be different from those on the table.
- Local secondary index – An index that has the same partition key as the table, but a different sort key.

Each table in DynamoDB has a quota of 20 global secondary indexes (default quota) and 5 local secondary indexes.

# DynamoDB Streams



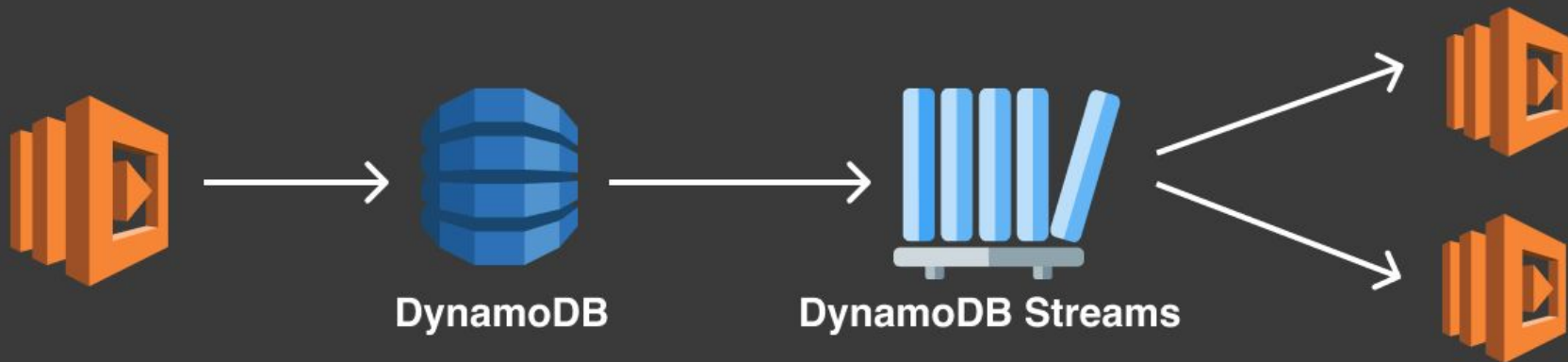
DynamoDB Streams is an optional feature that captures data modification events in DynamoDB tables. The data about these events appear in the stream in near-real time, and in the order that the events occurred.

Each event is represented by a *stream record*. If you enable a stream on a table, DynamoDB Streams writes a stream record whenever one of the following events occurs:

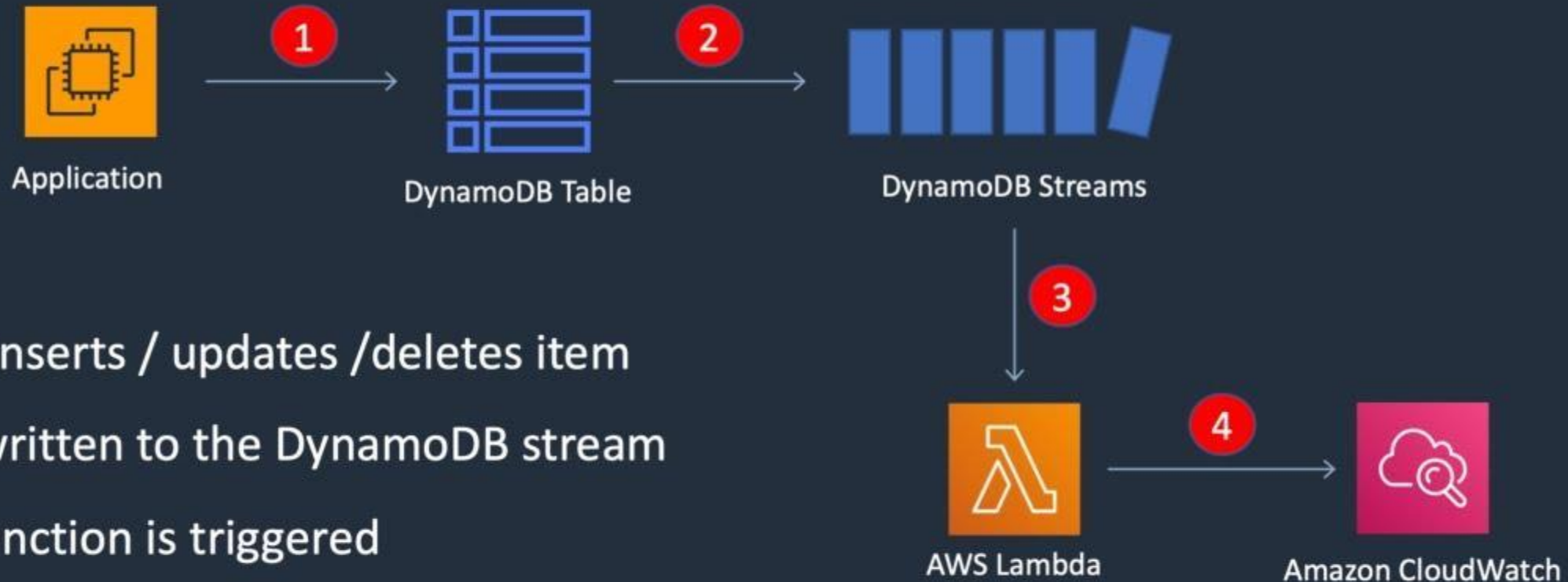
- A new item is added to the table: The stream captures an image of the entire item, including all of its attributes.
- An item is updated: The stream captures the "before" and "after" image of any attributes that were modified in the item.
- An item is deleted from the table: The stream captures an image of the entire item before it was deleted.

# DynamoDB Streams

- Time-ordered sequence of item-level changes in table
- Stored for 24 hours
- Inserts, updates, and deletes
- Combine with Lambda functions for functionality like stored procedures
- FIFO time sequence



# Amazon DynamoDB Streams



1. Application inserts / updates / deletes item
2. A record is written to the DynamoDB stream
3. A Lambda function is triggered
4. The Lambda function writes to CloudWatch Logs



# Global Tables



it's a way of replicating your DynamoDB tables from one region to another.

You will need **DynamoDB streams turned on** in order to enable this.

## Managed Multi-Master, Multi-Region Replication

- Globally distributed applications
- Based on DynamoDB streams
- Multi-region redundancy for disaster recovery or high availability
- No application rewrites
- Replication latency under **1 second**

# DynamoDB Accelerator (DAX)



## What is DynamoDB Accelerator (DAX) ?

**In-Memory Cache** : DAX can reduce DynamoDB response times from milliseconds to microseconds.

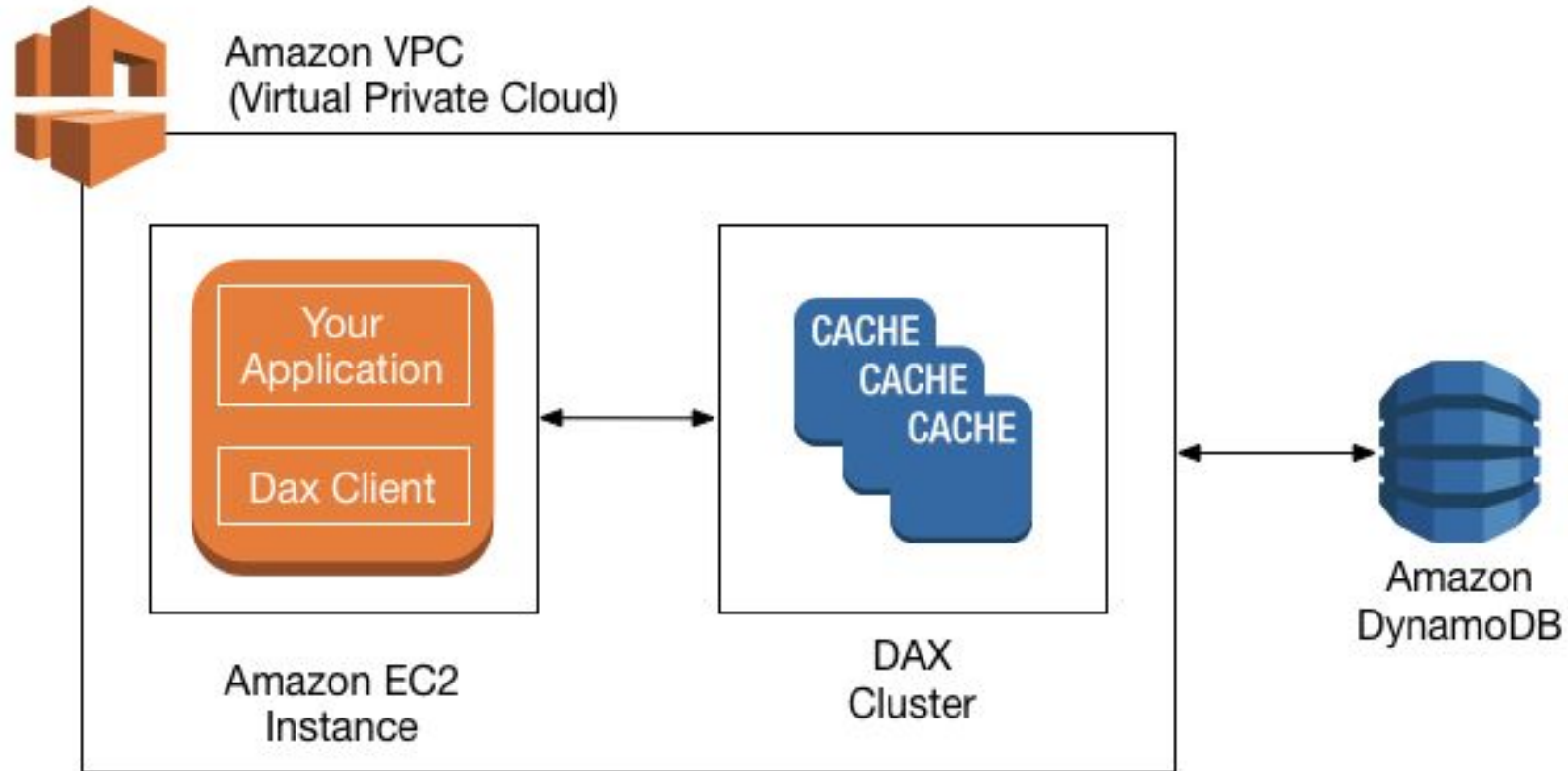
Fully managed, high available, in-memory cache

10x performance improvement

Reduces request time from millisecond to **microseconds** – even under load

Compatible with DynamoDB API calls

# DynamoDB Accelerator (DAX)



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



# DynamoDB



Let's get our hands dirty!

- Create a DynamoDB table



# THANKS!

## Any questions?

You can find me at:

- ▶ @Paul - Instructor
- ▶ paul@clarusway.com

