What is DynamoDB?

amazon DynamoDB

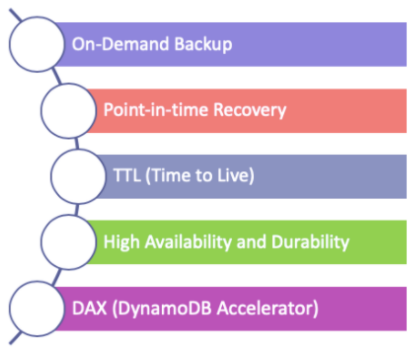
There are two types of databases, **Relational** and **Non-Relational**. As you remember from the previous lessons, we just learned the Relational database (SQL) service, RDS. Now, It's time to cover **DynamoDB** which is the **Non-Relational (NoSQL)** database solution of AWS.

Amazon DynamoDB is a **NoSQL database** service that supports **key-value** and **document** data models. It is NoSQL database, so you don't need to stick pre-determined schema. Instead of Schema, DynamoDB uses **flexible tables**. You can create database tables that can store and retrieve any amount of data and serve any level of request traffic.

Unlike RDS, Amazon DynamoDB is a **fully-managed** database. It means, there are no servers to provision, patch, or manage, and no software to install, maintain or operate. DynamoDB automatically scales tables to adjust for capacity and maintains performance with **zero administration**.

DynamoDB is designed to run high-performance, internet-scale applications that would overburden traditional relational databases like mobile, web, gaming, ad-tech, IOT, and many other applications.

Features of DynamoDB



Features of DynamoDB

On-Demand Backup:

DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Point-in-time Recovery:

After creating on-demand backups, you can enable point-in-time recovery for your Amazon DynamoDB tables. Point-in-time recovery helps protect your tables from accidental write or delete operations. With point-in-time recovery, you can restore that table to any point in time during the last 35 days.

TTL(Time to Live)

DynamoDB has also **TTL** function. DynamoDB TTL allows you to delete expired items from tables automatically so that you can reduce storage usage and the cost of storing data that is no longer relevant.

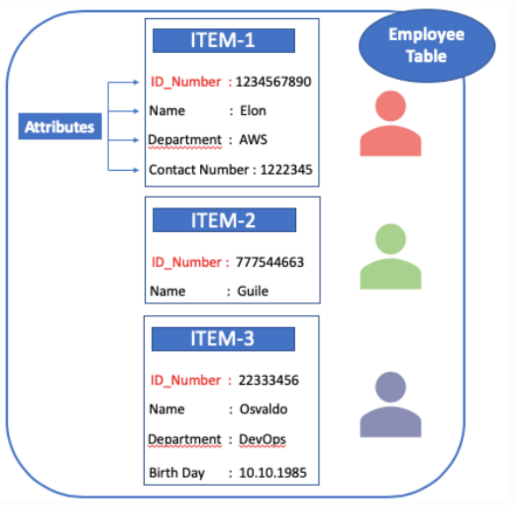
High Availability and Durability

In DynamoDB all of your data is stored on solid-state disks (SSDs) and is automatically replicated across **multiple Availability Zones** in an AWS Region. but also you can use **Global Tables** to keep DynamoDB tables in **multiple AWS Regions**.

DAX (DynamoDB Accelerator)

Although Amazon DynamoDB is designed for performance, you may need to require response times in **microseconds** for certain use cases. So, DynamoDB Accelerator (DAX) delivers fast response times for these use cases. In short, you can speed up your transactions with this feature.

Tables, Items and Attributes



Tables & Items & Attributes

In DynamoDB, **tables**, **items**, and **attributes** are the key components that you work with. A table is a collection of items, and each item is a collection of attributes.

- **Tables:**

Similar to other database systems, DynamoDB stores data in tables. A table is a collection of data. As you see in the picture above, there is a table called Employee and you can see the contact information of Employee in this table.

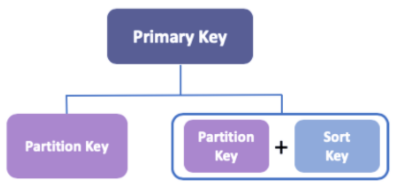
- **Items:**

Each table contains items. An item is a group of attributes that is uniquely identifiable among all of the other items. In an Employee Table, **each item represents a person**. Items in DynamoDB are similar in many ways to **rows and records** in other database systems. In DynamoDB, there is **no limit** to the number of items you can store in a table. But each item can not exceed **400 KB**.

- **Attributes:**

Each item is composed of one or more attributes. An attribute is a **key data element** for DynamoDB table. For example, an item in an Employee Table contains attributes called **PersonID, LastName, FirstName, and so on**.

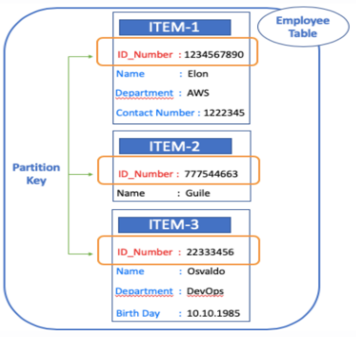
Primary Key



Primary Key

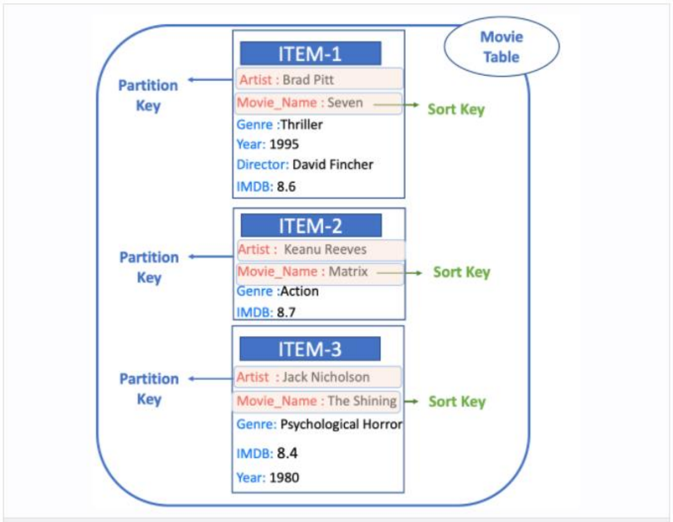
DynamoDB uses **Primary Keys** to **uniquely identify each item** in a table. When you create a table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key.

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



Partition Key

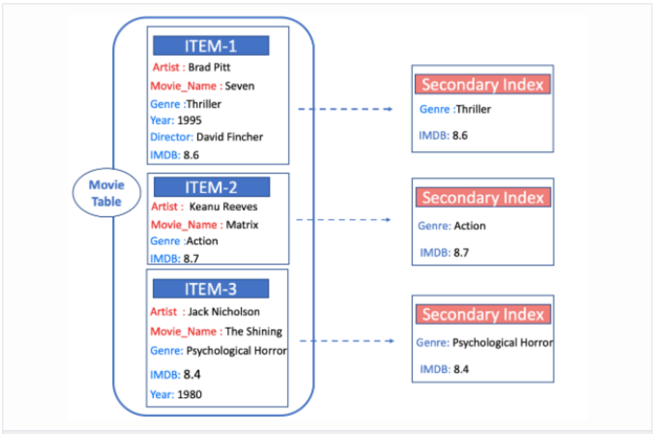
- **Partition Key:**
  - Partition Key is a simple primary key that composed of **one attribute**. If you determine one of the attributes as a Primary Key (Partition Key), there is no way to add an item into the table without Partition Key. Partition Key is **mandatory** value in the table.
  - As you see in the picture above, there is an **Employee Table** described in tables, items, and attributes. In this table, the primary key(Partition Key) is PersonID. All items (person) have PersonID value. You can access any item in the Employee Table directly by providing the PersonID value for that item.



Partition Key and Sort Key

- **Partition Key and Sort Key**
  - Unlike the Primary Key, this type of key is composed of **two attributes**. The first attribute is the **partition key**, and the second attribute is the **Sort Key**.
  - In fact, **Sort Key** is an optional key. We use the Sort Key to enable rich query capabilities. If there are 2 items that have the same Partition Key, thanks to the Sort Key, you can distinguish these two items from each other.
  - As you see in the picture above, there is a Movie Table described in tables, items, and attributes. The primary key for Movie consists of two attributes (Artist and Movie_Name). Each item in the table must have these two attributes. The combination of Artist and Movie_Name distinguishes each item in the table from all of the others.

Secondary Indexes

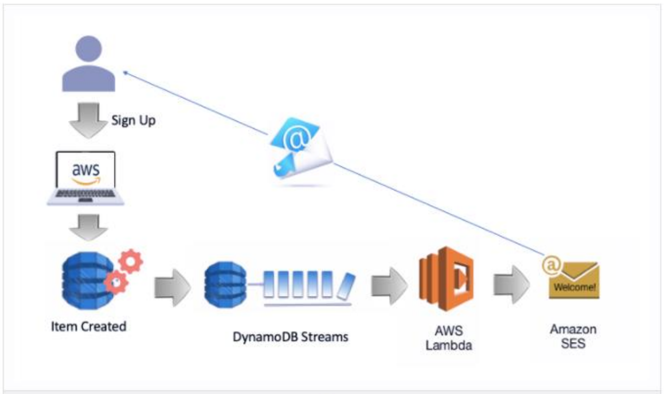


Secondary Indexes

- DynamoDB uses **Secondary Indexes** to **enrich queries**.
- 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.
- In other words, Secondary Indexes are a way to create an **alternative query option** with current attributes of the item.
- In the example Movie Table shown above, you can query data items by Artist (partition key) or by Artist and Movie_Name (partition key and sort key). But, what if you also wanted to query the data by **Genre and IMDB Rate**? To do this, you can create a **Secondary Index** on Genre and IMDB, and then you can read data from the index in much the same way as you do from the Movie 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 limit of **20 global secondary indexes** (default limit) and **5 local secondary indexes** per table.

DynamoDB Streams



DynamoDB Streams

DynamoDB Streams is an optional feature that **captures data modification events** in DynamoDB tables. It is a kind of transaction record.

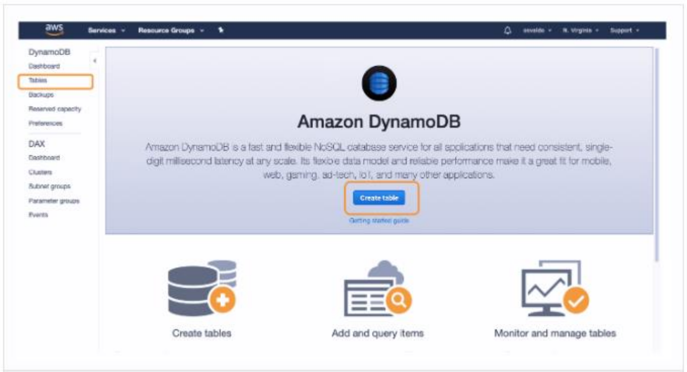
Each event is represented by a stream record. If you enable a stream on a table, DynamoDB writes a stream record whenever events such as **adding, updating or deleting an item** occurs in the table.

Stream records have a lifetime of **24 hours**; after that, they are automatically removed from the stream.

But, why do we need streams?

- **First**, you can use DynamoDB Streams **together with AWS Lambda to create a trigger**. For example, in the picture above:
  - Thanks to the stream whenever a new customer is added to the customer table, DynamoDB Streams creates a Stream Record,
  - Since we associate the stream with a Lambda function, the Lambda function will trigger Amazon Simple Email Service (Amazon SES)
  - Then Amazon Simple Email Service (Amazon SES) will send a welcome email to the customer.
- **Another reason** for using DynamoDB Streams is to **enable the Global Tables** which means Multi-Region Replication with DynamoDB

Creating DynamoDB Table

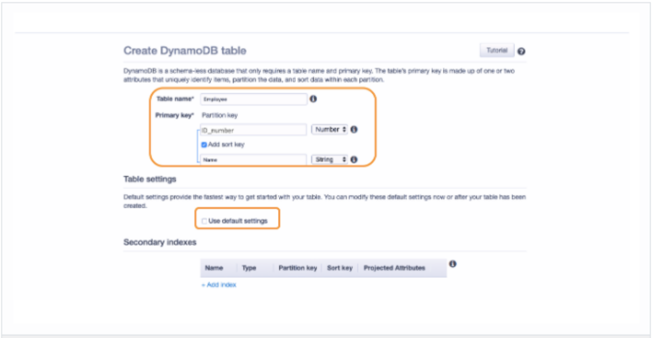


Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Dashboard

- First, go to the DynamoDB service on AWS Management Console and Click **Create Table** tab as you seen in the picture above. You can also reach **Create Table** tab, by clicking **Table** option from the left-hand menu.

- Then set the DynamoDB Table on the page opened.



Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*: Employee

Primary key*: Partition key

ID_number — Number

[✓] Add sort key

Name — String

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

[ ] Use default settings

Secondary indexes

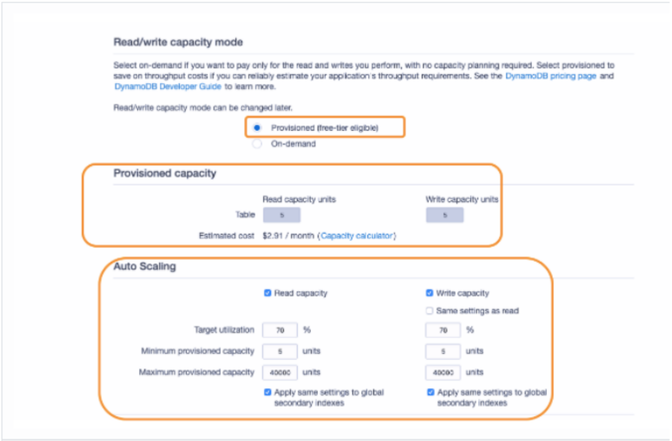
Name	Type	Partition key	Sort key	Projected Attributes

+ Add index

Create DynamoDB Table

- **Table Name:** Let's name our first table as **Employee** in which we plan to store our employee's information.
- **Primary Key:** We need to determine the Primary Key. As you remember from the previous lesson, we can choose only Partition Key or we can choose **Sort Key** together with **Partition Key** as Primary Key. We prefer to choose both of them for now.
  - **Partition Key:** We determine **ID_Number** as Partition Key. Since ID_number is consists of numbers we choose **Number** option from the **Type Box** near the Partition Key.
  - **Add Sort Key:** To add Sort Key, first check the **Add sort Key** box. Then enter **Name** as Sort Key. After that choose the **String** from the **Type Box** near the Sort Key.
- **Table Settings: Use default settings** box is checked as default. Since we want to see detailed options, let's **uncheck** the box. Then options will open below.

- **Secondary Indexes:** Since our table is simple, we don't need to use its option. Leave it as is.



Read/write capacity mode

Select on-demand if you want to pay only for the read and writes you perform, with no capacity planning required. Select provisioned to save on throughput costs if you can reliably estimate your application's throughput requirements. See the DynamoDB pricing page and DynamoDB Developer Guide to learn more.

Read/write capacity mode can be changed later.

[✓ Provisioned (free-tier eligible)] On-demand

Provisioned capacity

	Read capacity units	Write capacity units
Table	5	5

Estimated cost $2.91 / month (Capacity calculator)

Auto Scaling

	[✓] Read capacity		[✓] Write capacity	
			[ ] Same settings as read	
Target utilization	70	%	70	%
Minimum provisioned capacity	5	units	5	units
Maximum provisioned capacity	40000	units	40000	units
	[✓] Apply same settings to global secondary indexes		[✓] Apply same settings to global secondary indexes	

Read/Write Capacity Mode

- **Read/Write Capacity Mode:**

Since theDynamoDB is a fully managed service, unlike RDS, you don't need to select any instance type to determine your capacity in DynamoDB. Instead of selecting an instance, you determine a capacity interval. Consequently, you will be charged for **reading**, **writing**, and **storing data** in your DynamoDB tables

DynamoDB has two capacity options: **On-Demand** and **Provisioned**.

1-Provisioned Capacity Mode:

With provisioned capacity mode, you **specify the number of reads and writes per second** that you expect your application to require. You can use **Auto-Scaling to automatically adjust your table's capacity** based on the specified utilization rate to ensure application performance while reducing costs.

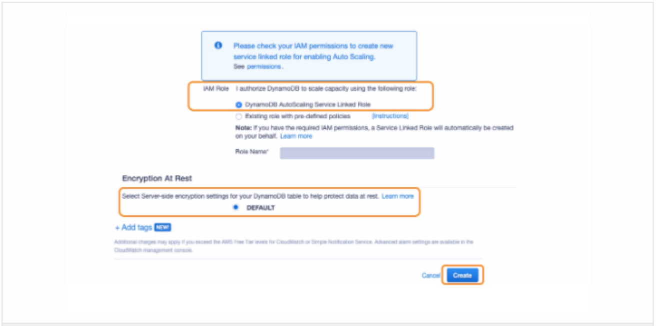
The provisioned capacity mode might be best if you have **predictable application traffic**.

2- On-Demand Capacity Mode:

With on-demand capacity mode, DynamoDB charges you for the data reads and writes your application performs on your tables. You do not need to specify how much read and write throughput you expect your application to perform because DynamoDB instantly accommodates your workloads as they ramp up or down.

It might be best if you create new tables with **unknown workloads** and you have **unpredictable** application traffic.

- **Provisioned Capacity:** Capacity of 5 read and write per second is enough for know. If we need more capacity we can provide it via Auto-Scaling seen below.
- **Auto-Scaling** Auto-Scaling automatically adjust our table's capacity from 5 to 40000 read/write per second based on the specified utilization rate (default %70). So, leave it as default.



Please check your IAM permissions to create new service linked role for enabling Auto Scaling. See permissions.

IAM Role: I authorize DynamoDB to scale capacity using the following role:

[✓ DynamoDB AutoScaling Service Linked Role] Existing role with pre-defined policies [instructions]

Note: If you have the required IAM permissions, a Service Linked Role will automatically be created on your behalf. Learn more

Role Name*

Encryption At Rest

Select Server-side encryption settings for your DynamoDB table to help protect data at rest. Learn more

[✓ DEFAULT]

+ Add tags NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

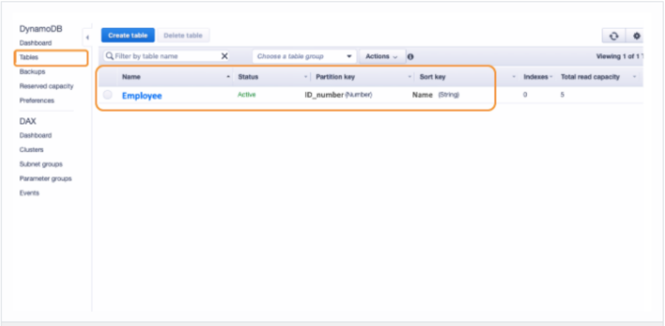
Cancel Create

IAM Role&Encryption

IAM Role: We need to assign the IAM role to DynamoDB to use Auto-scaling for adjusting the capacity. We select **DynamoDB AutoScaling Service Linked Role** option and AWS automatically assigns a role for us.

- **Encryption:** Leave it as default for now.
- **Add Tags**: We don't need to add a tag for now.

Then Click **Create.**



DynamoDB: Dashboard, Tables, Backups, Reserved capacity, Preferences

DAX: Dashboard, Clusters, Subnet groups, Parameter groups, Events

Create table | Delete table

Filter by table name | Choose a table group | Actions | Viewing 1 of 1

Name	Status	Partition key	Sort key	Indexes	Total read capacity
Employee	Active	ID_number (Number)	Name (String)	0	5

Table

After creating DynamoDB Table, click the **Tables** option from left-hand menu. As you see in the picture above, the name of the table is **Employee**. Partition Key is **ID_number** and Sort Key is **Name**

Congratulations!!!

Complementary Lesson about AWS DynamoDB