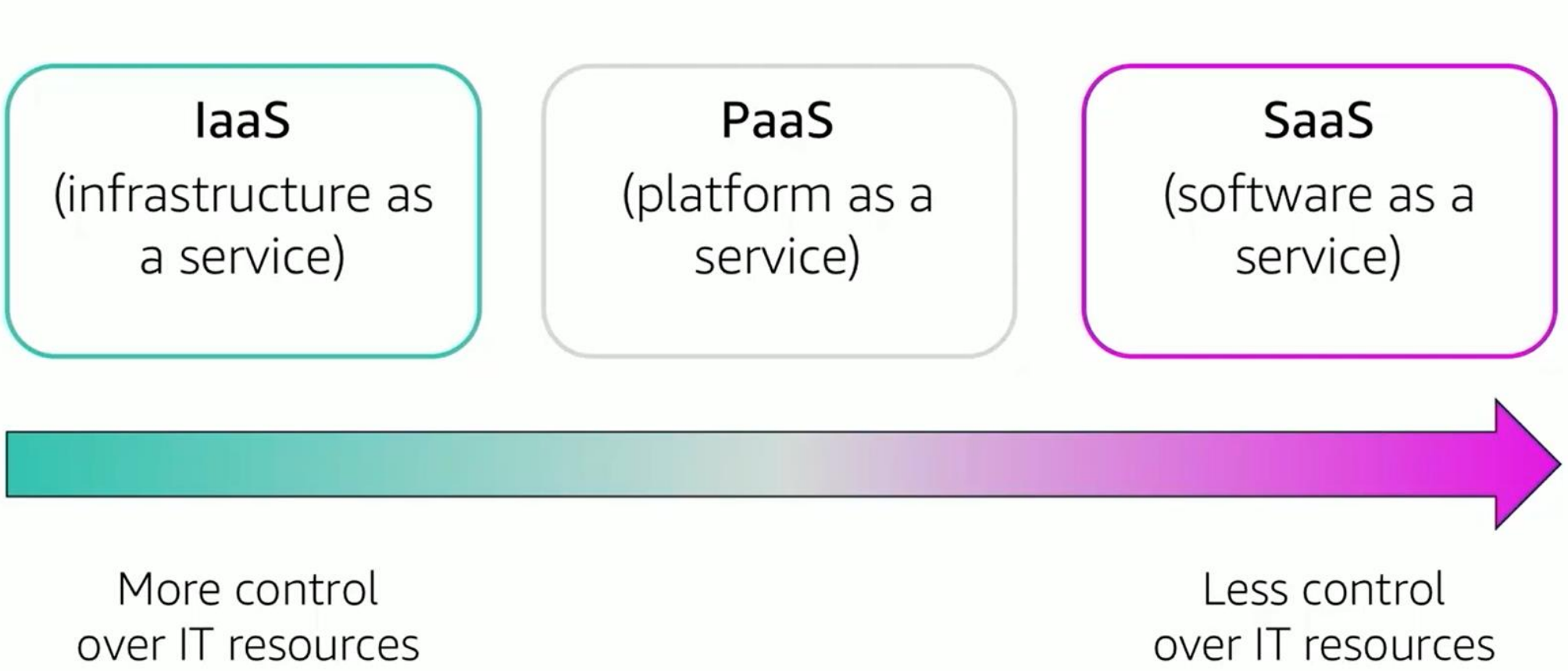
FLORIDA POLYTECHNIC UNIVERSITY

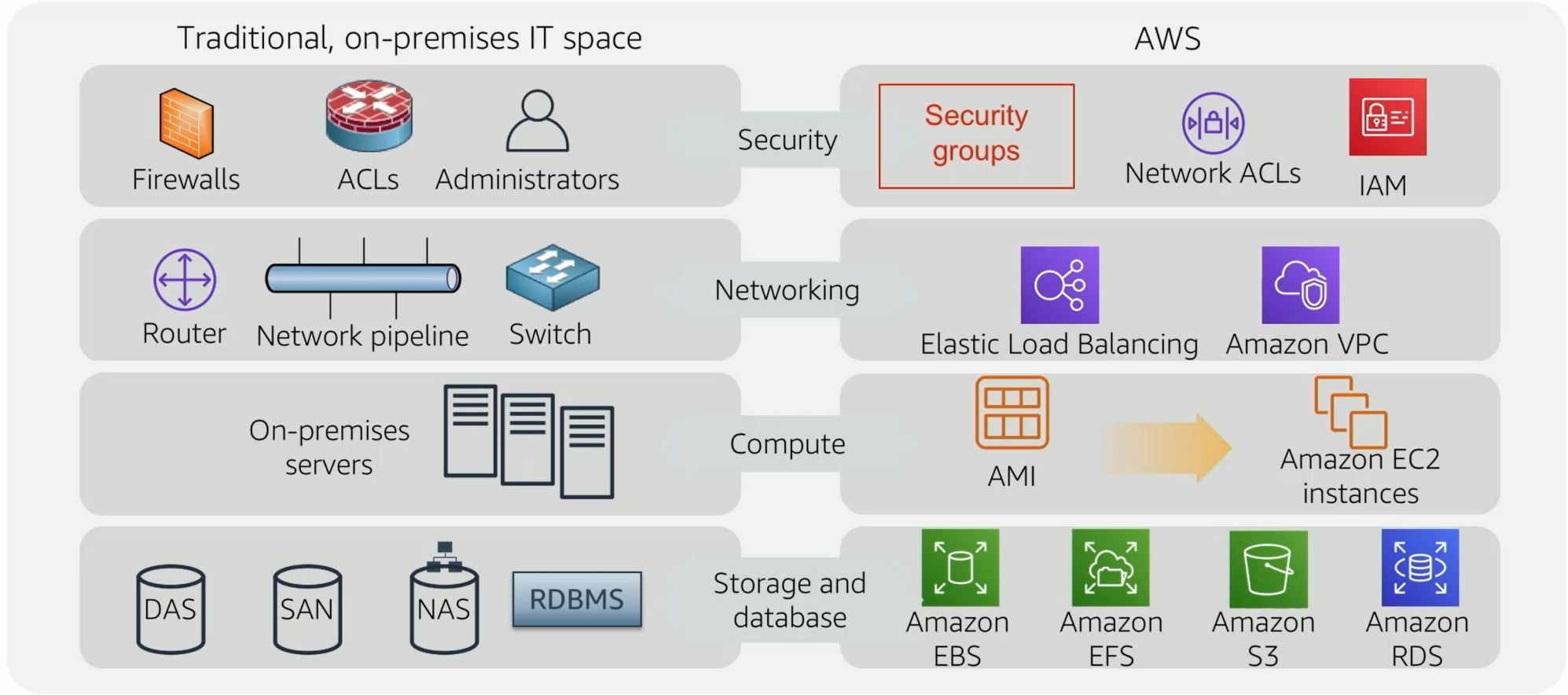
DynamoDB

Intro

What is Cloud



What is AWS

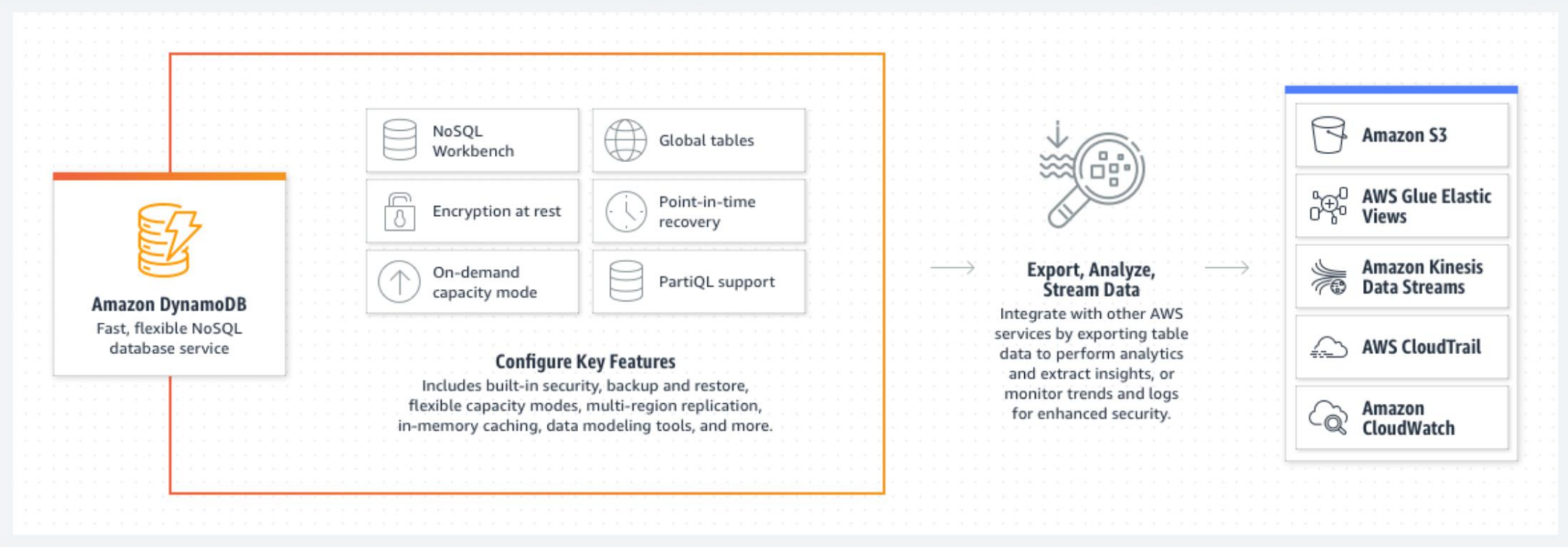


What is DynamoDB

- Cloud based
- Hosted on AWS
- More than 100,000 AWS customers have chosen DynamoDB
- Extremely Low Latency
- Scales to $ you have
- Millions of requests per second
- High availability and durability
- Originally built to serve as Amazons shopping cart

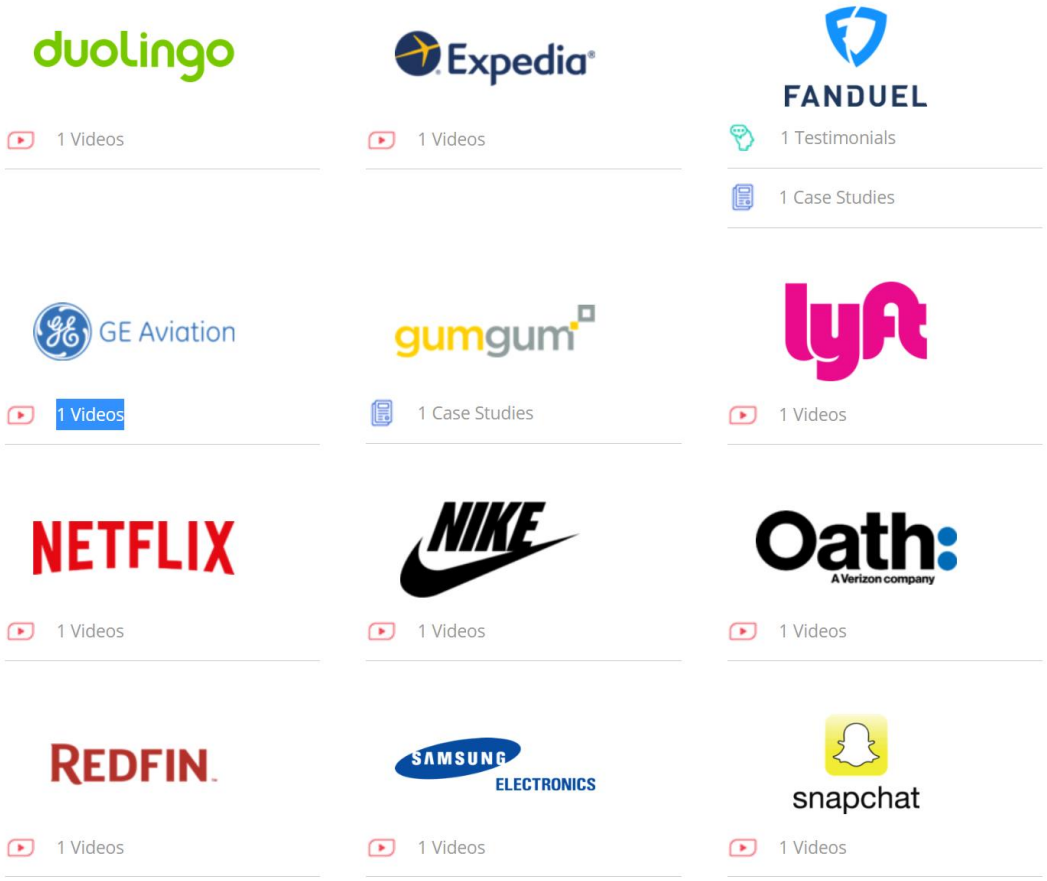
How DynamoDB works

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-region replication, in-memory caching, and data export tools.



FLORIDA POLYTECHNIC UNIVERSITY

Who Uses Dynamo



FLORIDA POLYTECHNIC UNIVERSITY

What do they use it for

Develop software applications

Build internet-scale applications supporting user-content metadata and caches that require high concurrency and connections for millions of users, and millions of requests per second.

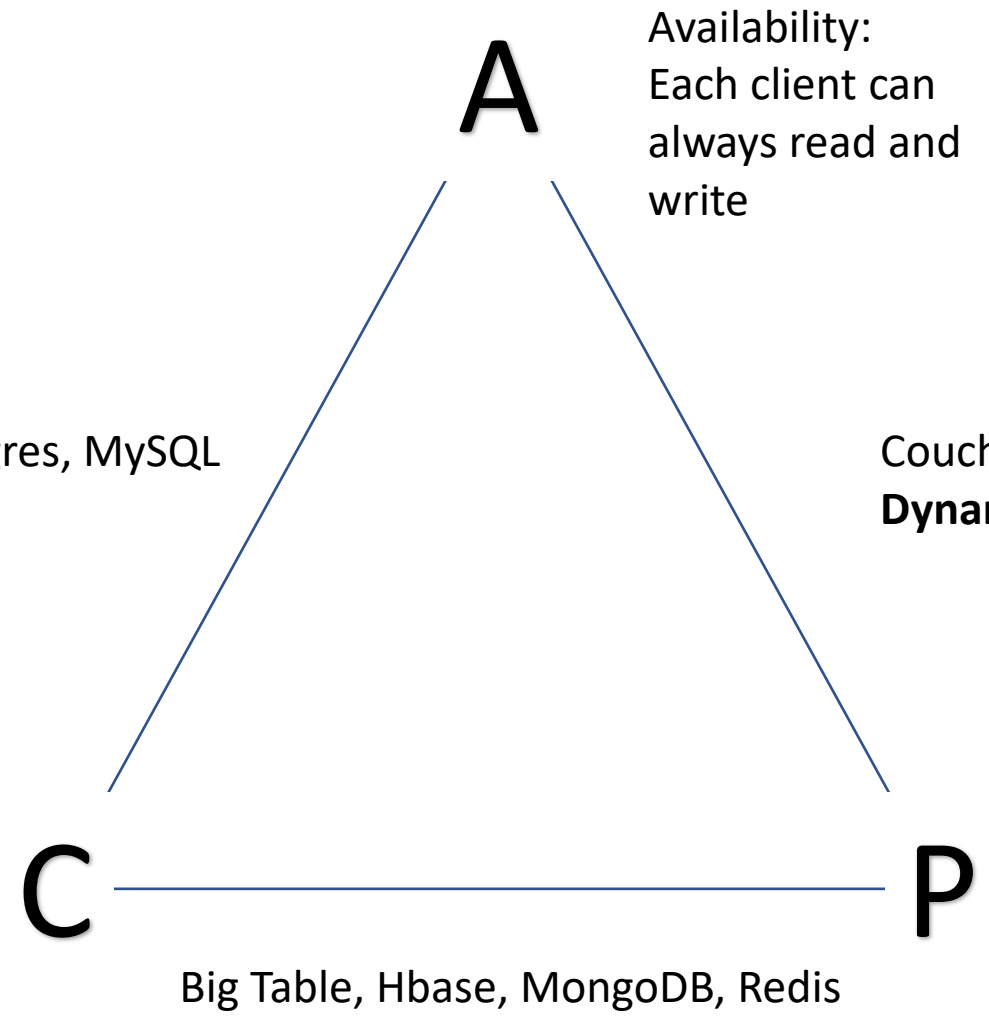
Create media metadata stores

Scale throughput and concurrency for media and entertainment workloads such as real-time video streaming and interactive content, and deliver lower latency with multi-region replication across AWS Regions.

Deliver seamless retail experiences

Use design patterns for deploying shopping carts, workflow engines, inventory tracking, and customer profiles. DynamoDB supports high-traffic, extreme-scaled events and can handle millions of queries per second.

Where does it fit in CAP Theorem



FLORIDA POLYTECHNIC UNIVERSITY

Consistent Types

- **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. However, this consistency comes with some disadvantages:

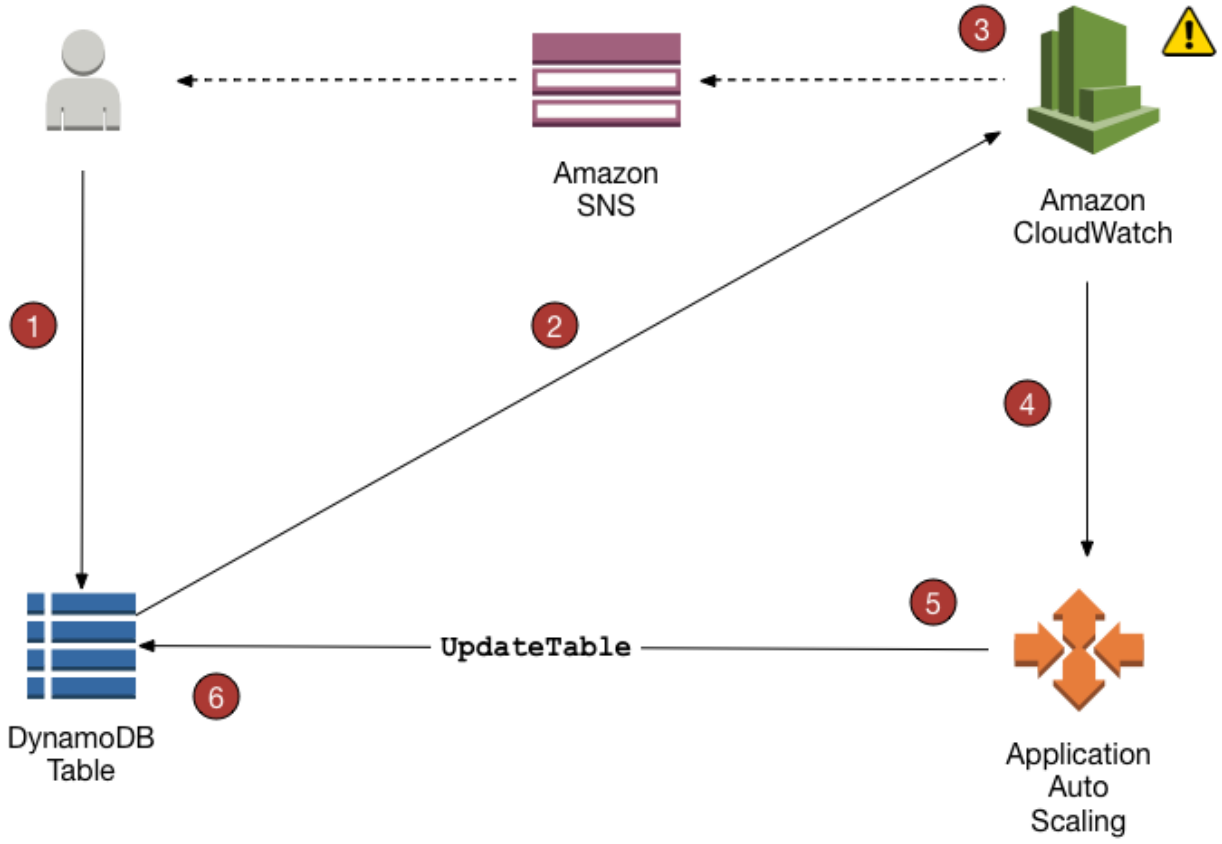
Fully Managed Service

- No operational burden
- Consistent, Predictable service
  - Global Table SLA 99.999%
  - Standard Table SLA 99.9%
- Focus on your app
- Auto Scaling
- On-Demand Backups
  - With point-in-time recovery, you can restore that table to any point in time during the last 35 days

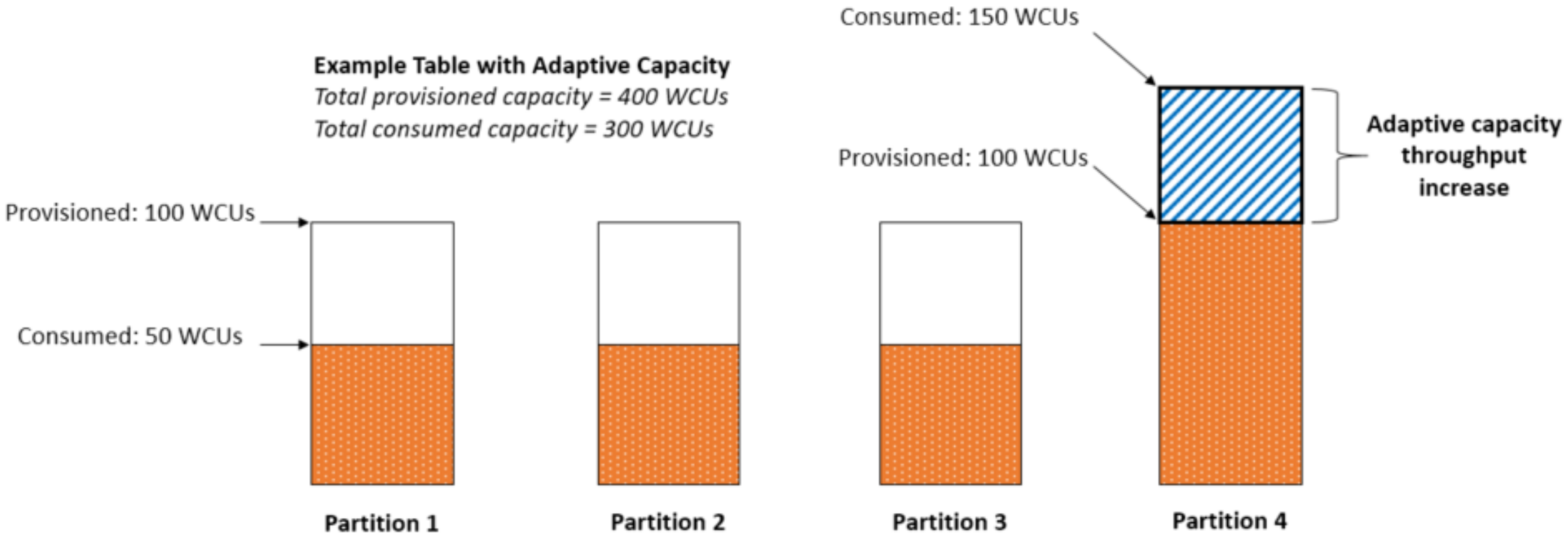
AWS Regions

Region Name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	dynamodb.us-east-2.amazonaws.com	HTTP and HTTPS
US East (N. Virginia)	us-east-1	dynamodb.us-east-1.amazonaws.com	HTTP and HTTPS
US West (N. California)	us-west-1	dynamodb.us-west-1.amazonaws.com	HTTP and HTTPS
US West (Oregon)	us-west-2	dynamodb.us-west-2.amazonaws.com	HTTP and HTTPS

Auto Scaling



Adaptive Capacity



FLORIDA POLYTECHNIC UNIVERSITY

Tables, Items, and Attributes

- **Tables** – A *table* is a collection of data.
- **Items** – Each table contains zero or more items. An *item* is a group of attributes that is uniquely identifiable among all of the other items.
- **Attributes** – Each item is composed of one or more attributes. An *attribute* is a fundamental data element, something that does not need to be broken down any further.

Naming Rules

- All names must be encoded using UTF-8, and are case-sensitive.
- Table names and index names must be between 3 and 255 characters long, and can contain only the following characters:
  - a-z
  - A-Z
  - 0-9
  - _ (underscore)
  - - (dash)
  - . (dot)
- Attribute names must be between 1 and 255 characters long.

Data Types

- Scalar Types – A scalar type can represent exactly one value. The scalar types are number, string, binary, Boolean, and null.
- Document Types – A document type can represent a complex structure with nested attributes, such as you would find in a JSON document. The document types are list and map.
  - Lists are enclosed in square brackets: [...]
  - Maps are enclosed in curly braces: { ... }
- Set Types – A set type can represent multiple scalar values. The set types are string set, number set, and binary set.
  - ["Black", "Green", "Red"]
  - [42.2, -19, 7.5, 3.14]

Example Table

People


```
{
    "PersonID": 101,
    "LastName": "Smith",
    "FirstName": "Fred",
    "Phone": "555-4321"
}
```

```
{
    "PersonID": 102,
    "LastName": "Jones",
    "FirstName": "Mary",
    "Address": {
        "Street": "123 Main",
        "City": "Anytown",
        "State": "OH",
        "ZIPCode": 12345
    }
}
```

```
{
    "PersonID": 103,
    "LastName": "Stephens",
    "FirstName": "Howard",
    "Address": {
        "Street": "123 Main",
        "City": "London",
        "PostalCode": "ER3 5K8"
    },
    "FavoriteColor": "Blue"
}
```

Example Table

Music


```
{
    "Artist": "No One You Know",
    "SongTitle": "My Dog Spot",
    "AlbumTitle": "Hey Now",
    "Price": 1.98,
    "Genre": "Country",
    "CriticRating": 8.4
}
```

```
{
    "Artist": "No One You Know",
    "SongTitle": "Somewhere Down The Road",
    "AlbumTitle": "Somewhat Famous",
    "Genre": "Country",
    "CriticRating": 8.4,
    "Year": 1984
}
```

```
{
    "Artist": "The Acme Band",
    "SongTitle": "Still in Love",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 2.47,
    "Genre": "Rock",
    "PromotionInfo": {
        "RadioStationsPlaying": [
            "KHCR",
            "KQBX",
            "WTNR",
            "WJJH"
        ],
        "TourDates": {
            "Seattle": "20150625",
            "Cleveland": "20150630"
        },
        "Rotation": "Heavy"
    }
}
```

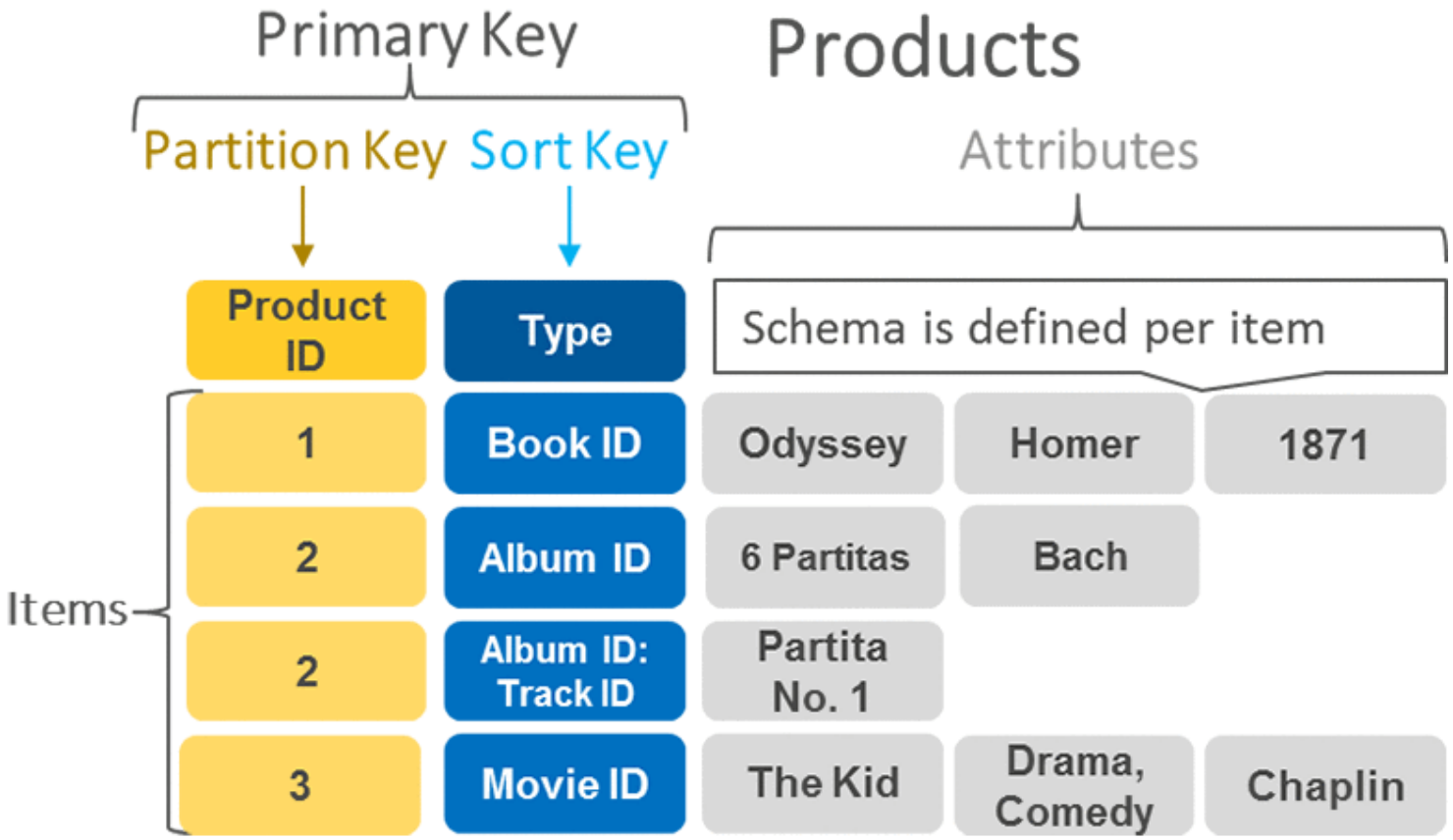
```
{
    "Artist": "The Acme Band",
    "SongTitle": "Look Out, World",
    "AlbumTitle": "The Buck Starts Here",
    "Price": 0.99,
    "Genre": "Rock"
}
```


FLORIDA POLYTECHNIC UNIVERSITY

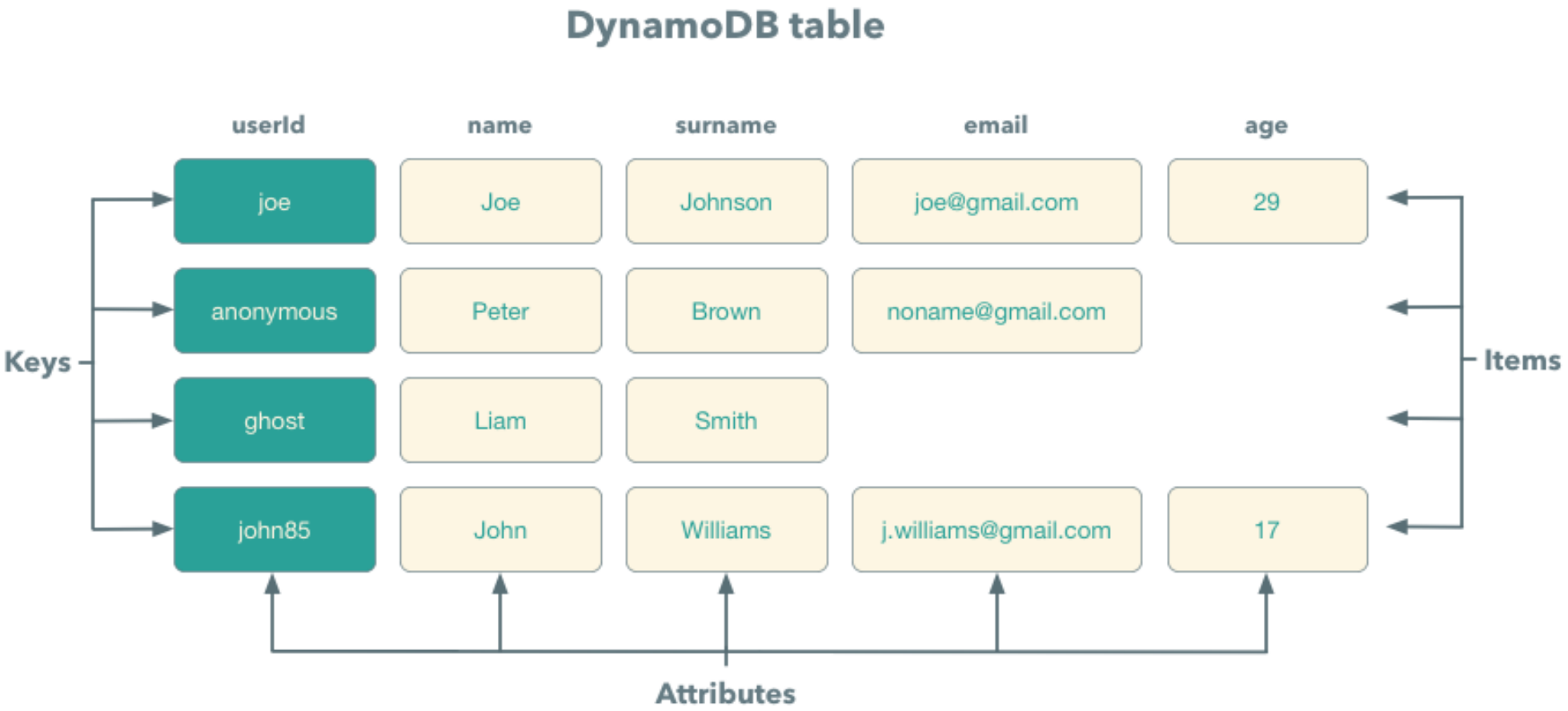
Primary Key

- **Partition key** – A simple primary key, composed of one attribute known as the *partition key*.
- In a table that has only a partition key, no two items can have the same partition key value.
- **Partition key and sort key** – Referred to as a *composite 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*.

Dynamo Structure



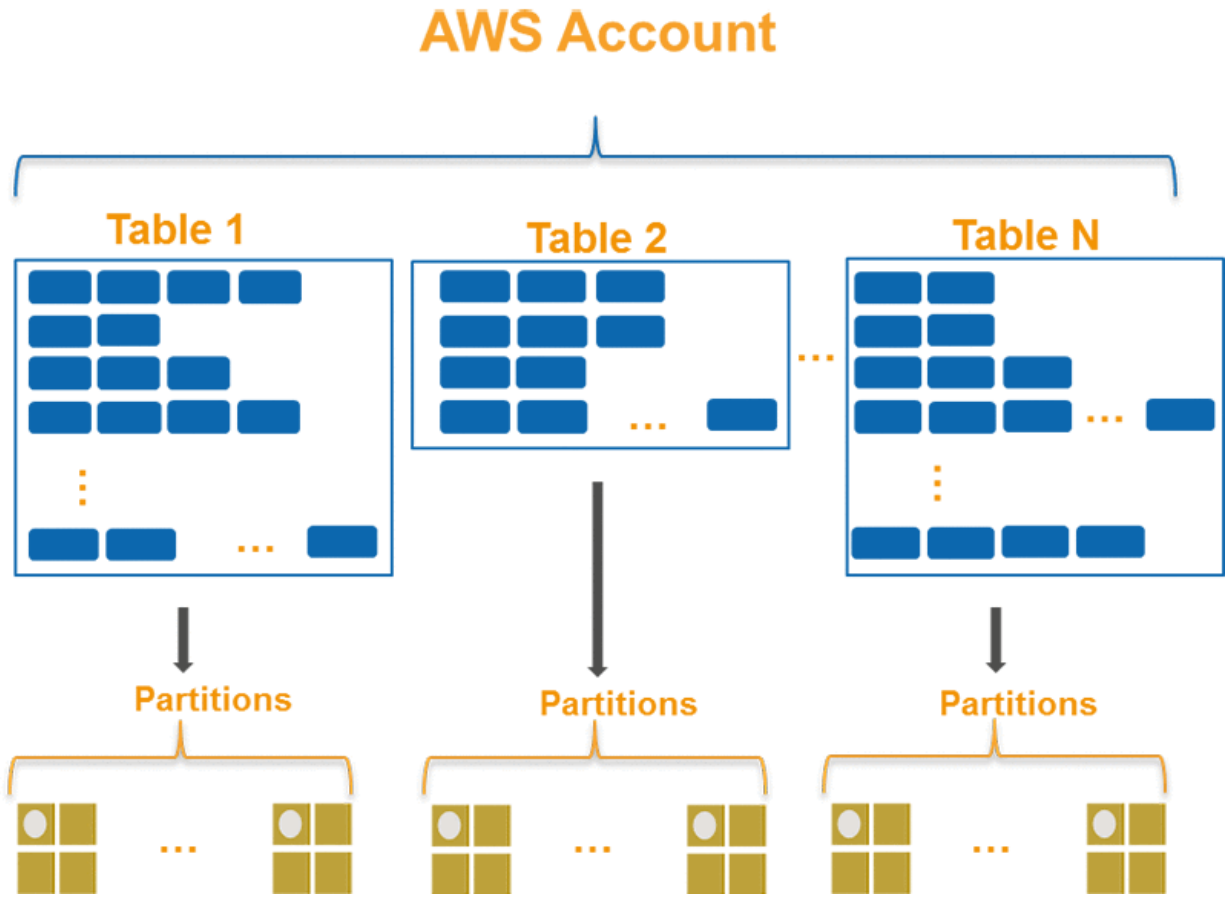
Tables



FLORIDA POLYTECHNIC UNIVERSITY

Partition Key

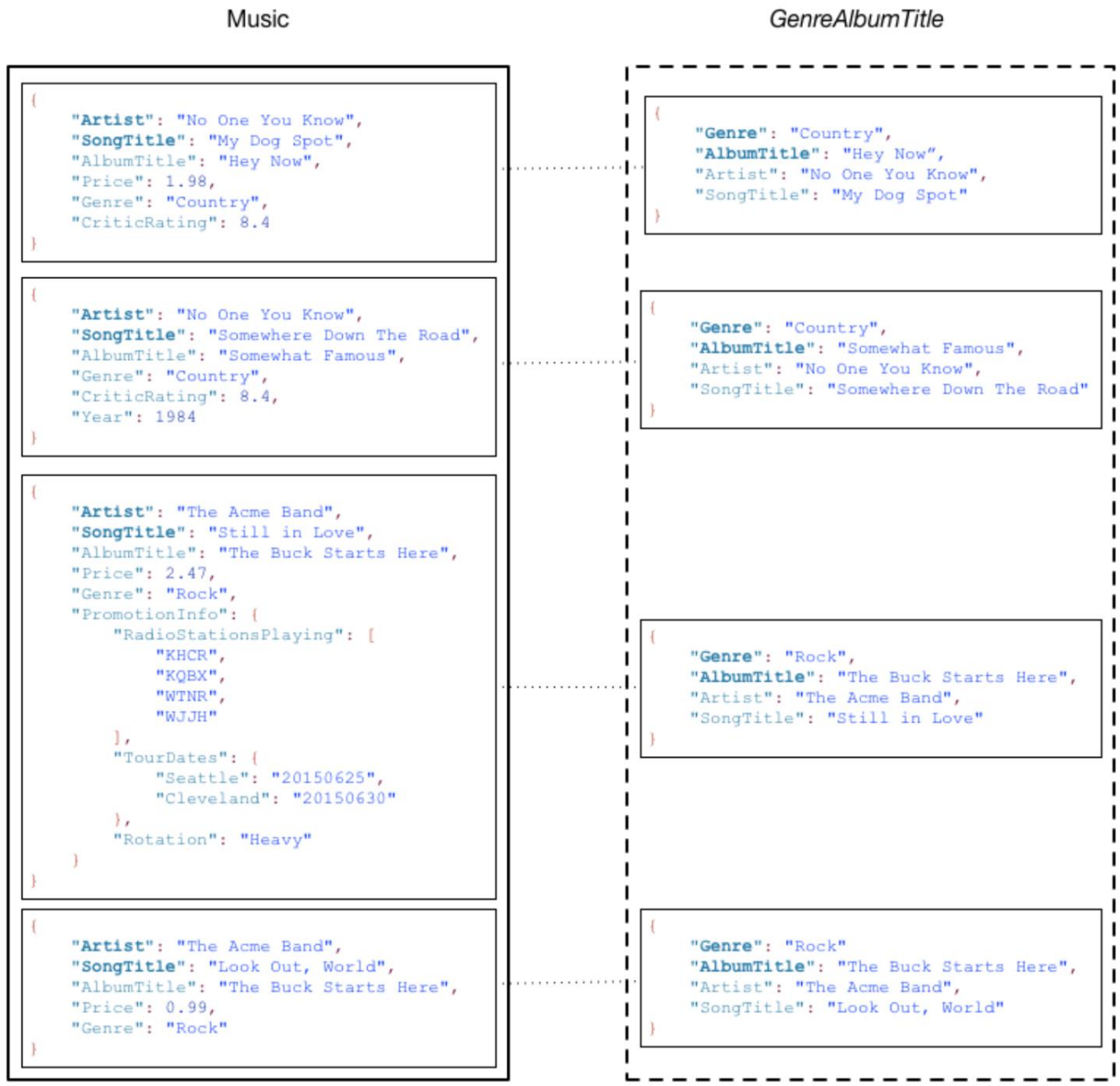
- Use High-cardinality attributes
- Use Composite attributes



Secondary 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.
- You can define up to 5 global secondary indexes and 5 local secondary indexes per table.

Example Index



FLORIDA POLYTECHNIC UNIVERSITY

Global Tables

- Multi-region
- Multi-master
- Automatic replication
- No Code

DAX In Memory Accelerator

- As an in-memory cache, DAX reduces the response times of eventually-consistent read workloads by an order of magnitude, from single-digit milliseconds to microseconds.
- DAX reduces operational and application complexity by providing a managed service that is API-compatible with Amazon DynamoDB, and thus requires only minimal functional changes to use with an existing application.
- For read-heavy or bursty workloads, DAX provides increased throughput and potential operational cost savings by reducing the need to over-provision read capacity units. This is especially beneficial for applications that require repeated reads for individual keys.

Encryption at Rest

- Can only be enabled when creating a new table
- Cannot turn on for existing tables
- encrypts your data using 256-bit AES encryption
- automatically integrates with AWS Key Management Service (AWS KMS)
- Encrypts table and indexes
- Decryption handled transparently by dynamoDB

Pricing

- https://aws.amazon.com/dynamodb/pricing/
- First 25 GB stored per month is free
- $0.25 per GB-month thereafter

Provisioned Throughput Type	Price per hour	Performance
Write capacity unit (WCU)	$0.00065 per WCU	1 WCU provides up to 3,600 writes per hour
Read capacity unit (RCU)	$0.00013 per RCU	1 RCU provides up to 7,200 reads per hour

Limits

- **Table Size**
  - There is no practical limit on a table's size. Tables are unconstrained in terms of the number of items or the number of bytes.
- **Tables Per Account**
  - For any AWS account, there is an initial limit of 256 tables per region.
- **US Regions:**
  - Per table – 40,000 read capacity units and 40,000 write capacity units
  - Per account – 80,000 read capacity units and 80,000 write capacity units
- **Item Size**
  - The maximum item size in DynamoDB is 400 KB

Control Plane

- CreateTable – Creates a new table. Optionally, you can create one or more secondary indexes, and enable DynamoDB Streams for the table.
- DescribeTable– Returns information about a table, such as its primary key schema, throughput settings, index information, and so on.
- ListTables – Returns the names of all of your tables in a list.
- UpdateTable – Modifies the settings of a table or its indexes, creates or remove new indexes on a table, or modifies DynamoDB Streams settings for a table.
- DeleteTable – Removes a table and all of its dependent objects from DynamoDB.

Data Plane

- PutItem – Writes a single item to a table. You must specify the primary key attributes, but you don't have to specify other attributes.
- BatchWriteItem – Writes up to 25 items to a table. This is more efficient than calling PutItem multiple times because your application only needs a single network round trip to write the items. You can also use BatchWriteItem for deleting multiple items from one or more tables.
- GetItem – Retrieves a single item from a table. You must specify the primary key for the item that you want. You can retrieve the entire item, or just a subset of its attributes.
- BatchGetItem – Retrieves up to 100 items from one or more tables. This is more efficient than calling GetItem multiple times because your application only needs a single network round trip to read the items.
- Query – Retrieves all items that have a specific partition key. You must specify the partition key value. You can retrieve entire items, or just a subset of their attributes. Optionally, you can apply a condition to the sort key values, so that you only retrieve a subset of the data that has the same partition key. You can use this operation on a table, provided that the table has both a partition key and a sort key. You can also use this operation on an index, provided that the index has both a partition key and a sort key.
- Scan – Retrieves all items in the specified table or index. You can retrieve entire items, or just a subset of their attributes. Optionally, you can apply a filtering condition to return only the values that you are interested in and discard the rest.

Data Plane

- UpdateItem – Modifies one or more attributes in an item. You must specify the primary key for the item that you want to modify. You can add new attributes and modify or remove existing attributes. You can also perform conditional updates, so that the update is only successful when a user-defined condition is met. Optionally, you can implement an atomic counter, which increments or decrements a numeric attribute without interfering with other write requests.
- DeleteItem – Deletes a single item from a table. You must specify the primary key for the item that you want to delete.
- BatchWriteItem – Deletes up to 25 items from one or more tables. This is more efficient than calling DeleteItem multiple times because your application only needs a single network round trip to delete the items. You can also use BatchWriteItem for adding multiple items to one or more tables.

Data Plane

- Transactions provide atomicity, consistency, isolation, and durability (ACID) enabling you to maintain data correctness in your applications more easily.
- TransactWriteItems – A batch operation that allows Put, Update, and Delete operations to multiple items both within and across tables with a guaranteed all-or-nothing result.
- TransactGetItems – A batch operation that allows Get operations to retrieve multiple items from one or more tables.