Amazon DynamoDB

What is NoSQL

- NoSQL databases are non-tabular databases and store data differently than relational tables
- NoSQL databases store data in documents (ex: json) rather than relational tables
- NoSQL databases do not support join
- NoSQL databases scale horizontally
- Benefits of NoSQL Databases
  - Flexible data models
  - Horizontal scaling
  - Faster queries
  - Easy to work

NoSQL Database

- Most-popular types of NoSQL database
  - Document databases
  - Key-value stores
  - Wide-column databases
  - Graph databases
- Popular NoSQL Databases
  - MongoDB
  - Couchbase
  - Redis
  - Amazon DynamoDB
  - IBM Cloudant
  - RavenDB
  - Cassandra
  - HBase
  - Azure Cosmos DB

Amazon DynamoDB

- Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale.
- Single-digit millisecond performance at any scale
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Each table has a **primary key**
- Each table can have an infinite number of items
- Maximum size of a item is 400KB
- Data types supported are:
  - Scalar Types: String, Number, Binary, Boolean, Null
  - Document Types: List, Map
  - Set Types: String Set, Number Set, Binary Set

Components of Amazon DynamoDB

- The core components of DynamoDB are **tables**, **items**, and **attributes**
- A table is a collection of items, and each item is a collection of attributes.
- DynamoDB uses primary keys to uniquely identify each item in a table and secondary indexes to provide more querying flexibility.
- **Tables**: A table is a collection of data. DynamoDB stores data in tables.
- **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.

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"
}
```


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"
}
```

DynamoDB – Primary Keys

- When you create a table, table name and the primary key will be given.
- The primary key uniquely identifies each item in the table, so that no two items can have the same key
- DynamoDB supports 2 types of primary keys
  - Partition key
    - A simple primary key, composed of one attribute known as the partition key
  - Partition key and sort key
    - A composite primary key, composed of two attributes. partition key and sort key.
    - The combination must be unique

DynamoDB – Provisioned Throughput

- Table must have provisioned read and write capacity units
- Read Capacity Units (RCU): throughput for reads
- Write Capacity Units (WCU): throughput for writes
- Option to setup auto-scaling of throughput to meet demand
- ReadCapacityUnits
  - The maximum number of strongly consistent reads consumed per second
- WriteCapacityUnits
  - The maximum number of writes consumed per second

DynamoDB – Write Capacity Units

- One write capacity unit represents one write per second for an item up to 1 KB in size.
- If the items are larger than 1 KB, more WCU are consumed
- **Example 1**: we write 10 objects per seconds of 2 KB each.

  We need 2 * 10 = 20 WCU
- **Example 2**: we write 6 objects per second of 4.5 KB each

  We need 6 * 5 = 30 WCU (4.5 gets rounded to the upper KB)
- **Example 3**: we write 120 objects per minute of 2 KB each

  We need 120 / 60 * 2 = 4 WCU

DynamoDB – Read Capacity Units

- One read capacity unit represents **one strongly consistent read** per second, or **two eventually consistent** reads per second, for an item up to 4 KB in size.
- If the items are larger than 4 KB, more RCU are consumed
- **Example 1**: 10 **strongly consistent** reads per seconds of 4 KB each

  We need 10 * 4 KB / 4 KB = 10 RCU
- **Example 2**: 16 **eventually consistent** reads per seconds of 12 KB each

  We need (16 / 2) * (12 / 4) = 24 RCU
- **Example 3**: 10 strongly consistent reads per seconds of 6 KB each

  We need 10 * 8 KB / 4 = 20 RCU (we have to round up 6 KB to 8 KB)

DynamoDB – Write, Delete Data

- PutItem - Write data to DynamoDB
- UpdateItem – Update data in DynamoDB
- DeleteItem
  - Delete an individual row
  - Ability to perform a conditional delete
- DeleteTable
  - Delete a whole table and all its items
  - Much quicker deletion than calling DeleteItem on all items

DynamoDB – Batching Writes

- BatchWriteItem
  - Up to 25 PutItem and / or DeleteItem in one call
  - Up to 16 MB of data written
  - Up to 400 KB of data per item
- Batching allows you to save in latency by reducing the number of API calls done against DynamoDB
- Operations are done in parallel for better efficiency

DynamoDB – Reading Data

- GetItem:
  - Read based on Primary key
  - Primary Key = HASH or HASH-RANGE
  - Eventually consistent read by default
  - Option to use strongly consistent reads (more RCU - might take longer)
  - ProjectionExpression can be specified to include only certain attributes
- BatchGetItem:
  - Up to 100 items
  - Up to 16 MB of data
  - Items are retrieved in parallel to minimize latency

Amazon DynamoDB Accelerator (DAX)

- It is a fully managed, highly available, in-memory cache for Amazon DynamoDB that delivers up to a **10 times performance** improvement—**from milliseconds to microseconds**—even at millions of requests per second.
- DAX does all the heavy lifting required to add in-memory acceleration to your DynamoDB tables, without requiring developers to manage cache invalidation, data population, or cluster management.