Introduction to NoSQL – Dynomo DB

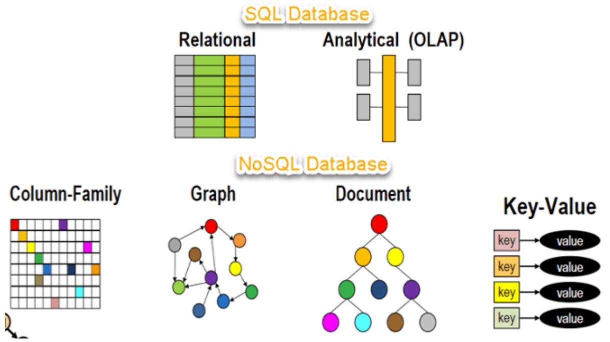
Agenda

- What is NoSQL?
- Why NoSQL?
- Brief History of NoSQL Databases
- Features of NoSQL
- Types of NoSQL Databases
- Query Mechanism tools for NoSQL
- What is the CAP Theorem?
- Eventual Consistency
- Advantages & Disadvantage of NoSQL
- Introduction to Amazon Dynomo DB

What is NoSQL?

What is NoSQL ?

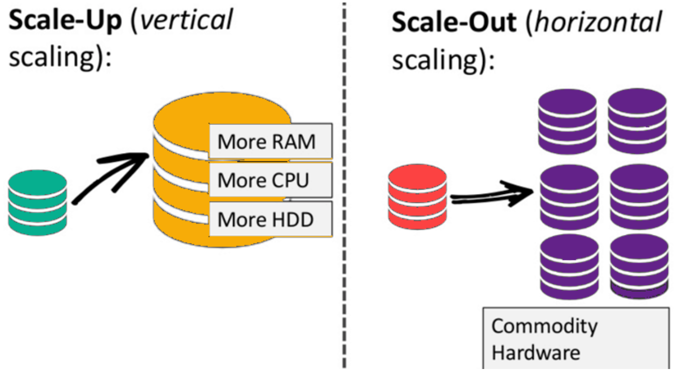
- NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale.
- NoSQL database is used for distributed data stores with humongous data storage needs.
- NoSQL is used for Big data and real-time web apps.
- For example, companies like Twitter, Facebook, Google that collect terabytes of user data every single day.
- NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would NoREL NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.
- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.



Why NoSQL?

Why NoSQL ?

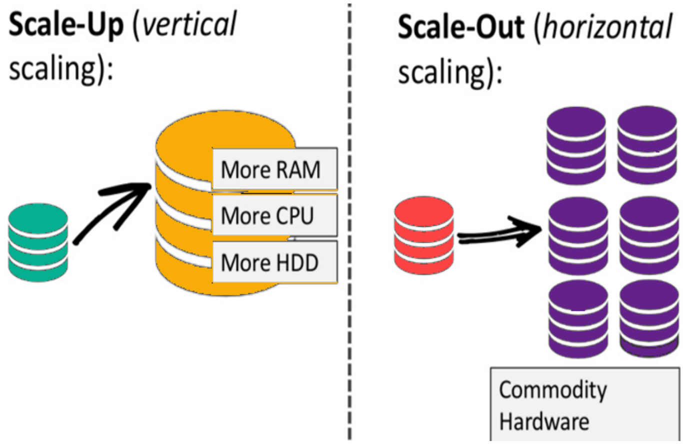
- The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.
- To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.
- The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out".
- NoSQL database is non-relational, so it scales out better than relational databases as they are designed with web applications in mind.



Brief History of NoSQL Databases

Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google BigTable is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced


Scale-Up (*vertical* scaling):
More RAM
More CPU
More HDD
Scale-Out (*horizontal* scaling):
Commodity Hardware

Features of NoSQL

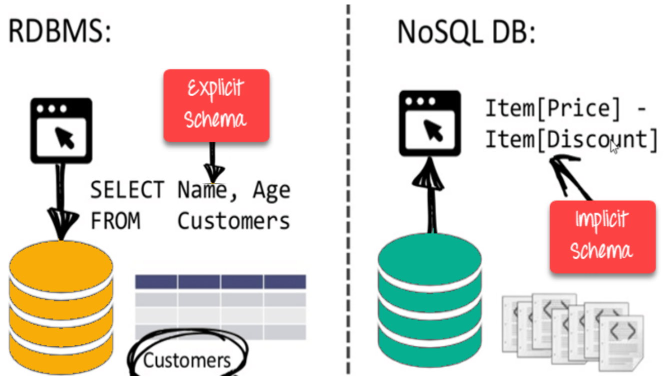
Features of NoSQL

Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners,
- referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain



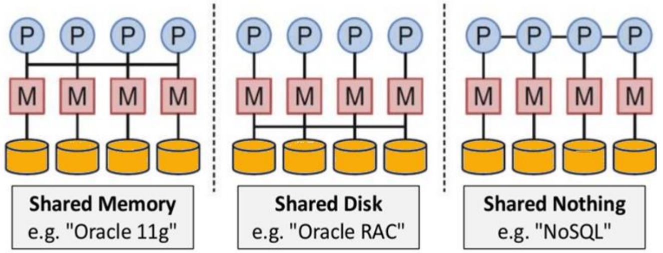
Features of NoSQL

Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based query language
- Web-enabled databases running as internet-facing services

Distributed

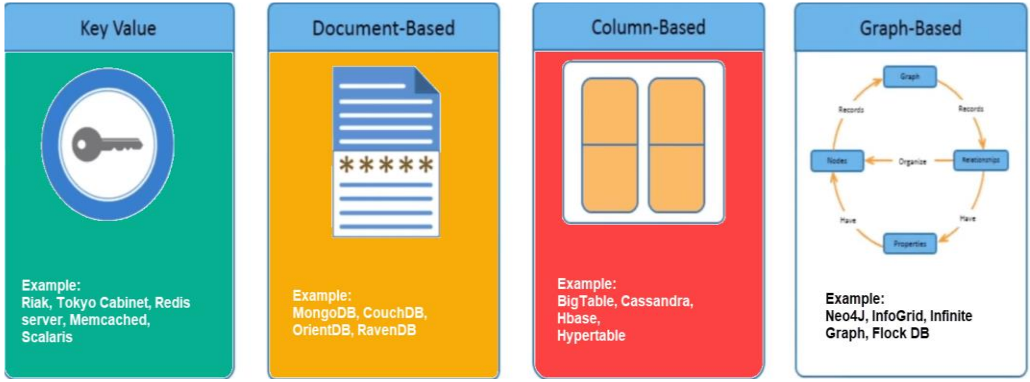
- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.



Types of NoSQL Databases

Types of NoSQL

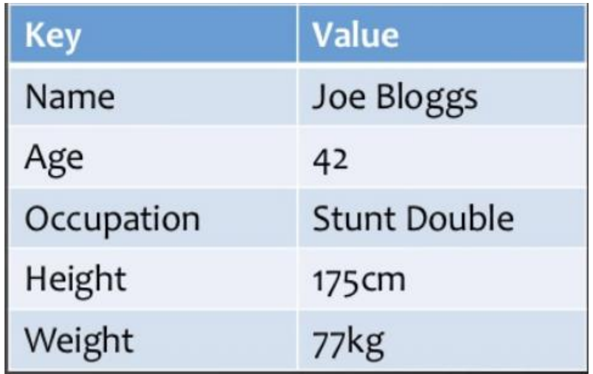
- There are mainly four categories of NoSQL databases. Each of these categories has its unique attributes and limitations.
- No specific database is better to solve all problems. You should select a database based on your product needs.
- Let see all of them:
- Key-value Pair Based
- Column-oriented Graph
- Graphs based
- Document-oriented


Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
*****
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Records
Nodes
Organize
Relationships
Have
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Types of NoSQL

Key-value Pair Based

- Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.
- It is one of the most basic types of NoSQL databases.
- This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data.
- They work best for shopping cart contents.
- Redis, Dynamo, Riak are some examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

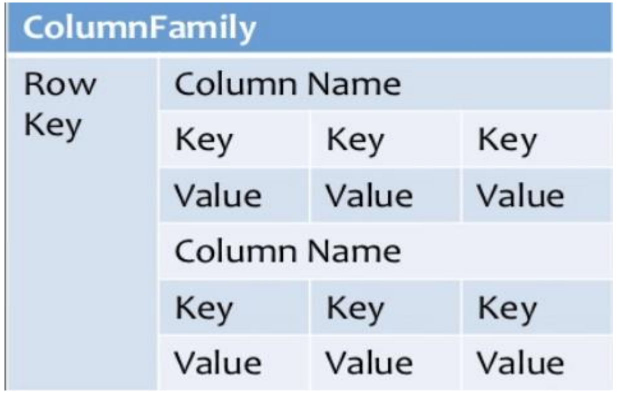


Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Types of NoSQL

Column-based

- Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs, HBase, Cassandra, HBase, Hypertable are examples of column based database.

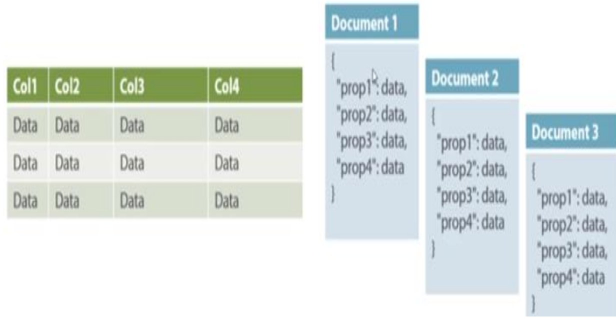


ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

Types of NoSQL

Document-Oriented:

- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.
- In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.
- The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.
- Amazon SimpleDB, CouchDB, MongoDB,Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.



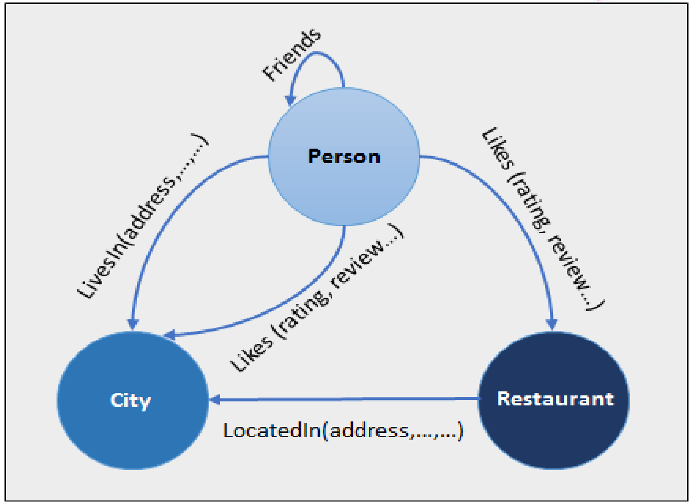
Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

```
Document 1
{
"prop1": data,
"prop2": data,
"prop3": data,
"prop4": data
}

Document 2
{
"prop1": data,
"prop2": data,
"prop3": data,
"prop4": data
}

Document 3
{
"prop1": data,
"prop2": data,
"prop3": data,
"prop4": data
}
```

Types of NoSQL

Graph-Based

- A graph type database stores entities as well the relations amongst those entities.
- The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes.
- Every node and edge has a unique identifier.
- Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature.
- Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.
- Graph base database mostly used for social networks, logistics, spatial data.
- Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.



Friends
Person
LivesIn(address,...,...)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

Query Mechanism tools for NoSQL

Query Mechanism tools for NoSQL

- The most common data retrieval mechanism is the REST-based retrieval of a value based on its key/ID with GET resource

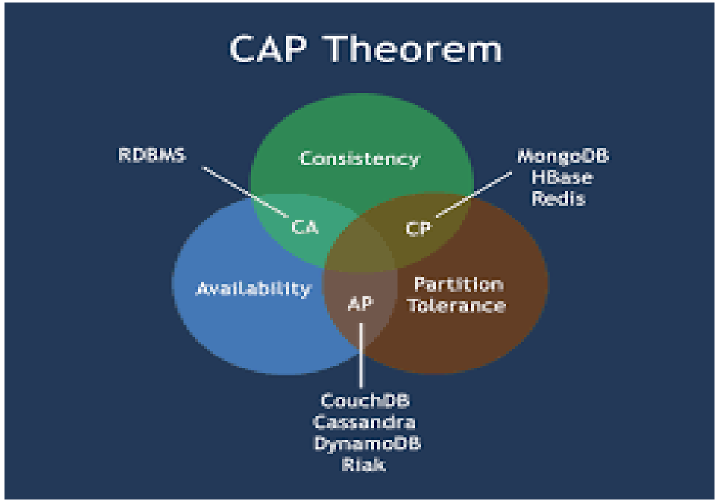
- Document store Database offers more difficult queries as they understand the value in a key-value pair. For example, CouchDB allows defining views with MapReduce

What is the CAP Theorem ?

What is the CAP Theorem?

- CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

- Consistency
- Availability
- Partition Tolerance



What is the CAP Theorem?

Consistency

- The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability

- The database should always be available and responsive. It should not have any downtime.

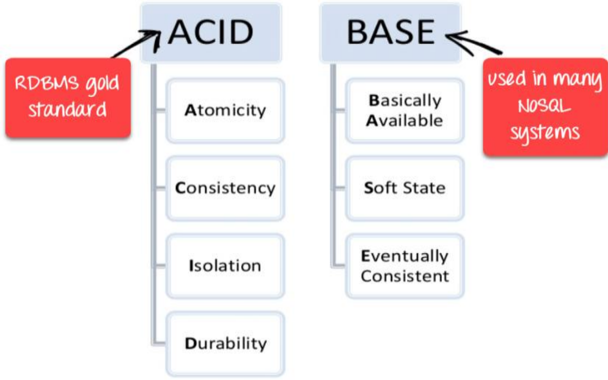
Partition Tolerance

- Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

Eventual Consistency

Eventual Consistency

- The term **"eventual consistency"** means to have copies of data on multiple machines to get high availability and scalability.
- Thus, changes made to any data item on one machine has to be propagated to other replicas.
- Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time.
- These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.
- BASE: Basically Available, Soft state, Eventual Consistency.
- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time



Advantages & Disadvantage of NoSQL

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantage of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

Introduction to Amazon Dynomo DB

Introduction to Amazon Dynomo DB

- **Amazon DynamoDB** is a key-value and document database that delivers single-digit millisecond performance at any scale.
- It's a fully managed, multiregion, multimaster, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications.
- DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.
- **Amazon DynamoDB** is a fully managed NoSQL database service that allows to create database tables that can store and retrieve any amount of data.
- It automatically manages the data traffic of tables over multiple servers and maintains performance.

Terminologies Associated With DynamoDB

Table, Items, and Attributes

- A table can be visualized as a group of items. Taking an example of Employee records, you will have Employee Name, Employee ID, Address and Phone Number all such items will be stored in a table.
- An item is a set of attributes in a table. You can also understand an item as a set of attributes that can uniquely define your entry in a table. For example, an item in Employee records will identify a single employee.
- An attribute is a single field that is attached to an item. E.g. Employee Name.

Terminologies Associated With DynamoDB

Primary Key

- A primary key is a unique attribute that is necessary while creating a table, it cannot be null at any given point. Hence, while inserting an item into the table, a primary key attribute is a must. E.g. Employee ID is the primary key for the table Employee records. Two items cannot have a similar primary key. DynamoDB supports two types of Primary key.

Simple Primary Key

- A simple primary key is also known as Partition key, this is basically a single attribute. DynamoDB uses Partition key's value to distinguish items in a table. E.g. Employee ID in Employee records table.

Composite Primary Key

- A composite primary key is also known as Partition key and Sort key. This type of key is generally made up of two items. The primary component is the Partition key and the secondary component is the Sort key. E.g. Car Details table with Brand name and Model number as a composite primary key.

Terminologies Associated With DynamoDB

Secondary Index

- A secondary index can be understood as the attribute that lets you query the data, with or without the help of a Primary key.
- DynamoDB has these secondary indexes that help you achieve this additional access.

DynamoDB Streams

- This is an additional/optional feature provided by DynamoDB to keep a track of data modification events in a table.
- Here, each event is represented by a stream record and if this service is enabled, then you get a new event every time when there is a new item created, an item is updated or an item is deleted.

Accessing Amazon DynamoDB

Accessing DynamoDB is very easy and can be done using the following methods:

Console

- You can access DynamoDB simply by clicking here.

CLI(Command Line Interface)

- Using CLI, you can simply open your command prompt and type the relevant commands and access the table. For more details, click here.

Using API

- Using AWS SDKs you can make the most of DynamoDB. AWS SDK supports a variety of languages like Java, JavaScript, .NET, Python, PHP etc. For more details, click here.

Features Of Amazon DynamoDB

DynamoDB is designed in such a way that the user can get high-performance, run scalable applications that would not be possible with the traditional database system.

- **On-demand capacity mode**: The applications using the on-demand service, DynamoDB automatically scales up/down to accommodate the traffic.
- **Built-in support for ACID transactions**: DynamoDB provides native/ server-side support for transactions.
- **On-demand backup**: This feature allows you to create a complete backup of your work at any given point of time.
- **Point-in-time recovery**: This feature helps you with the protection of your data in case of accidental read/ write operations.
- **Encryption at rest**: It keeps the data encrypted even when the table is not in use. This enhances security with the help of encryption keys.

Amazon DynamoDB API

DynamoDB is a database tool and to interact with an application, it requires API.

The APIs in DynamoDB are:

- Control Plane
- Data Plane
- DynamoDB Stream

Amazon DynamoDB API

Control Plane

- Control Plane consists of operations responsible for "*Creating*" and "*Managing*" a DynamoDB table.

The API operations that can be used are as follows:

- **CreateTable**: Creates a new table.
- **DescribeTable**: Provides information about the table.
- **ListTable**: Returns all the table names in your list.
- **DeleteTable**: Deletes the table and all its dependencies from DynamoDB.

Amazon DynamoDB API

Data Plane

- Data Plane consists of “CRUD” operation, i.e. “Create“, “Read“, “Update“, and “Delete” options to perform different actions on your table.
- Here in Data Plane, there are multiple operations that can be done on a table. The operations here are as follows:

Creating Data

- **PutItem:** You can write a single data item to your table with the help of Primary key.
- **BatchWriteItem:** It is a better version of PutItem, with this you can write upto 25 items to your table.

Amazon DynamoDB API

Data Plane

Reading Data

- **GetItem:** It retrieves a single item from a table with the help of the primary key.
- **BatchGetItem:** The better version of GetItem it can retrieve upto 100 items from multiple tables.
- **Query:** It is basically a command that retrieves an item which has a specific partition key.
- **Scan:** Works in a similar way as Query but doesn’t require partition key aslo it works on a specific table.

Updating Data

- **UpdateItem:** It modifies a single or multiple data items in a table with the help of Primary Key.

Deleting Data

- **DeleteItem**: It deletes a single item from the table with the help of Primary Key.
- **BatchWriteItem**: The better version of DeleteItem it can delete upto 25 items in a table.

Amazon DynamoDB API

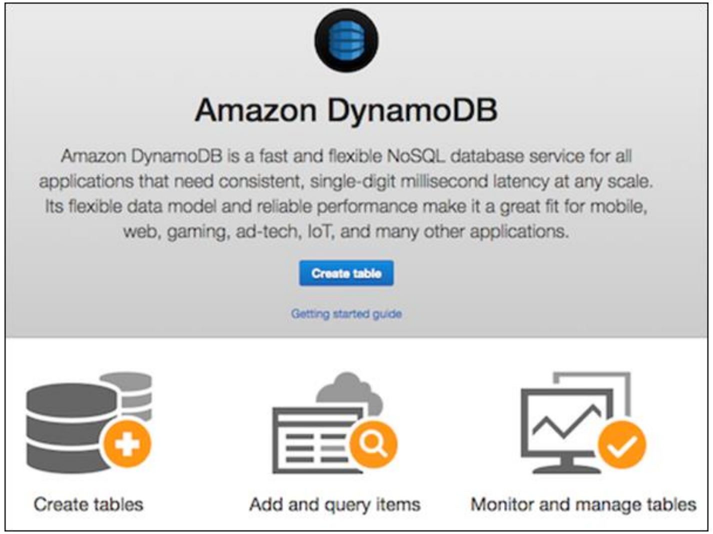
DynamoDB Stream

DynamoDB Stream is nothing but a service used to track data stream that is loaded into a table and retrieved from a table.

- To modify the streaming, the user can use the following commands:
  - **ListStream**: It gives a list of all streams.
  - **DescribeStream**: It gives detail about the stream and the resources used.
  - **GetShardIterator**: It gives a Shard iterator that is a data structure to store information about the stream.
  - **GetRecords**: Using Shard iterator GetRecords retrieves information about streams.

Demo: Creating, Inserting And Querying A Table In DynamoDB

Step 1: Navigate to the DynamoDB section in AWS or click here. Select "Create Table".

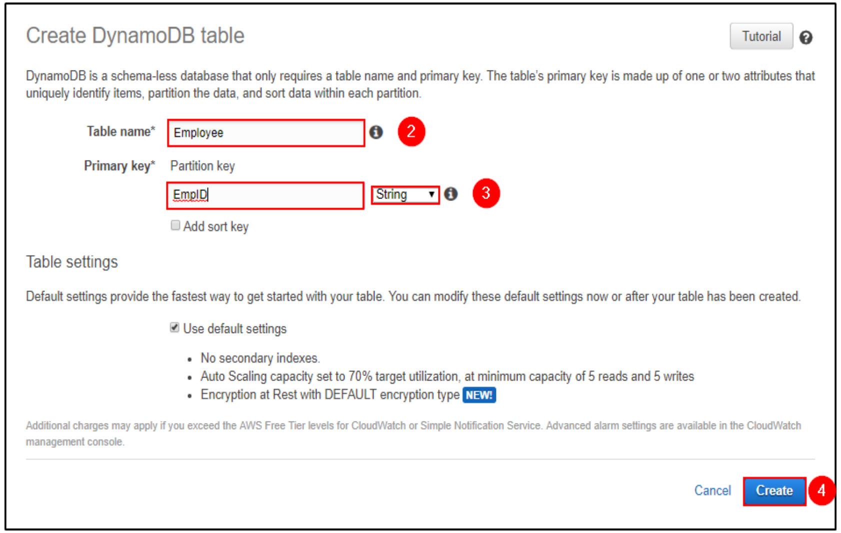


Amazon DynamoDB

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.

Demo: Creating, Inserting And Querying A Table In DynamoDB

Step 2: Fill in with the necessary details and click on "Create".



Create DynamoDB table

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

EmpID String

☐ Add sort key

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

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- Encryption at Rest with DEFAULT encryption type 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.

Demo: Creating, Inserting And Querying A Table In DynamoDB

Step 3: You can view your table being created. Click on “Overview” to understand your table, click on “Items” to edit, insert and query on the table. There are many more options you can use to understand your table better.

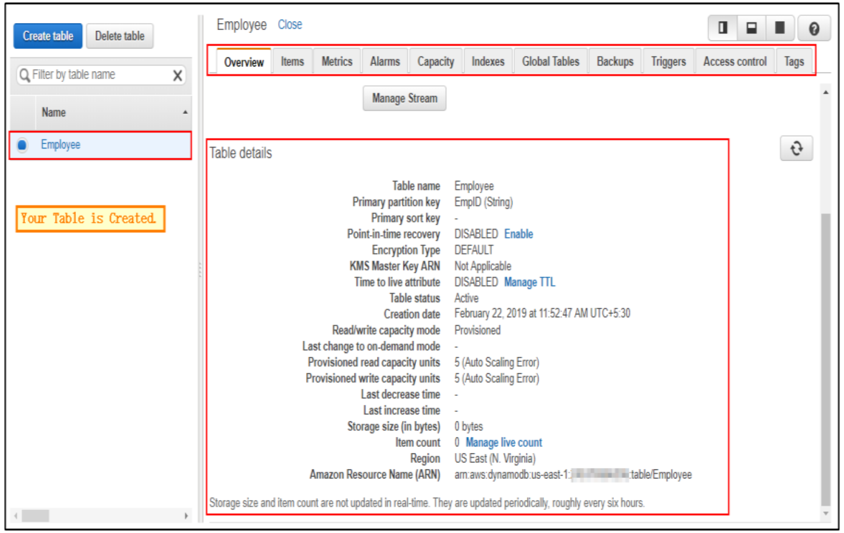


Table details

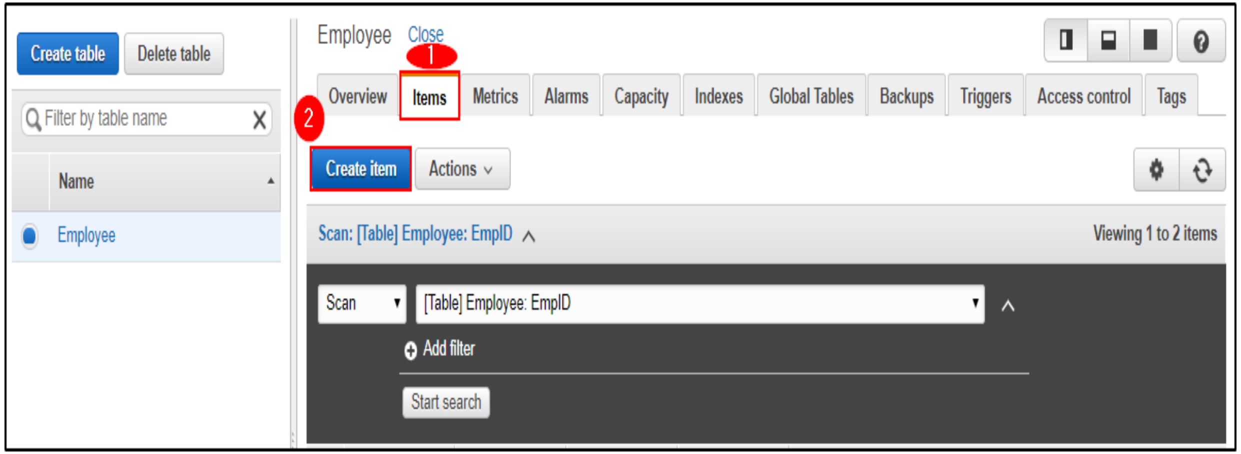
Table name	Employee
Primary partition key	EmpID (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	February 22, 2019 at 11:52:47 AM UTC+5:30
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5 (Auto Scaling Error)
Provisioned write capacity units	5 (Auto Scaling Error)
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0 Manage live count
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:[redacted]:table/Employee

Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

Demo: Creating, Inserting And Querying A Table In DynamoDB

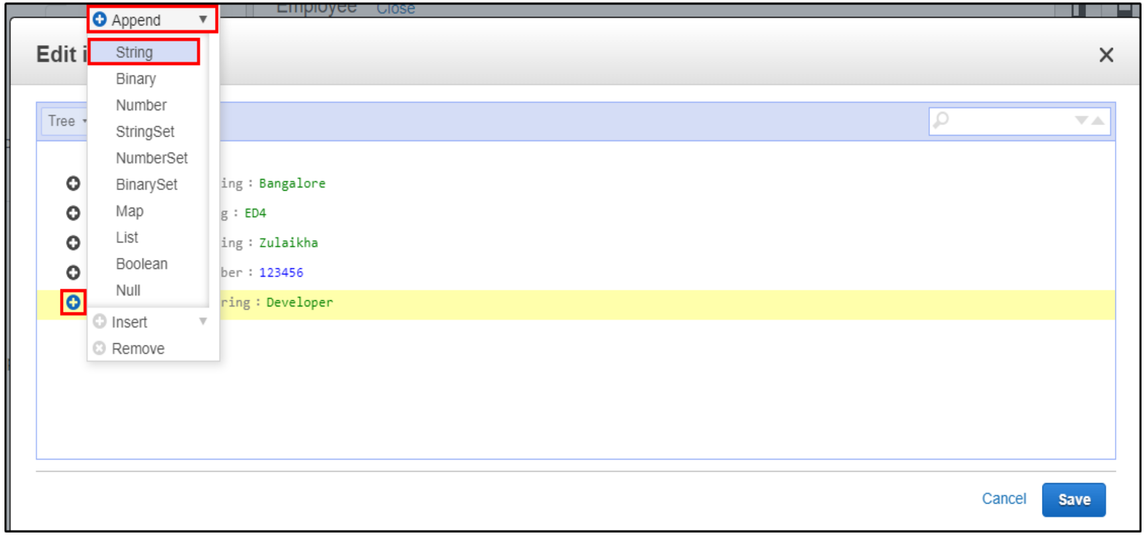
Now that you have created a table, let's go ahead and insert a few items and understand how NoSQL works.

Step 1: Navigate to "Items" and click on "Create item".



Demo: Creating, Inserting And Querying A Table In DynamoDB

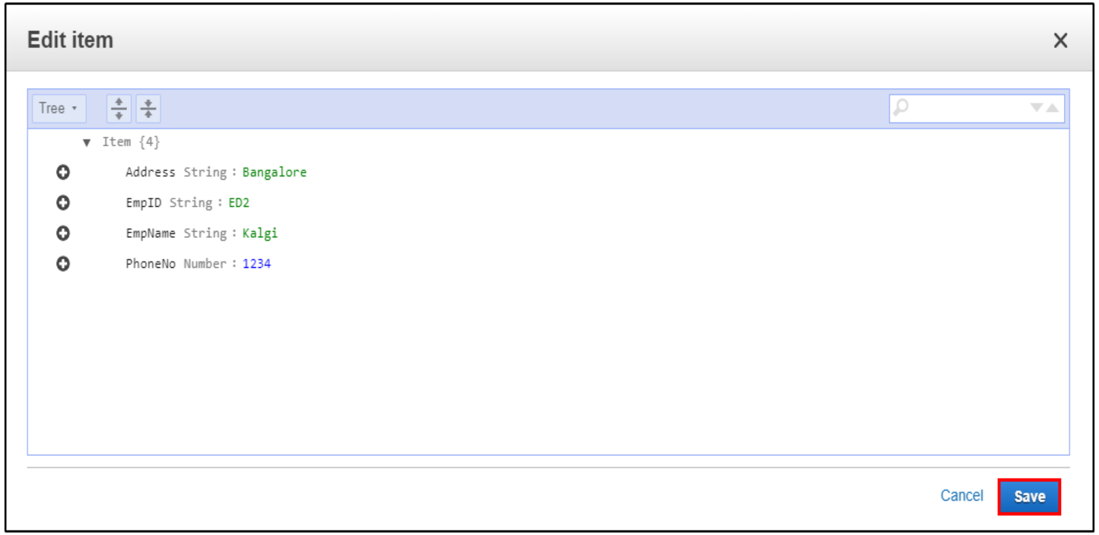
Step 2: It will open a JSON file where you can add different items. Click on the "+" symbol and select "Append" and select what type of data you want to enter.



```
ing : Bangalore
g : ED4
ing : Zulaikha
ber : 123456
ring : Developer
```

Demo: Creating, Inserting And Querying A Table In DynamoDB

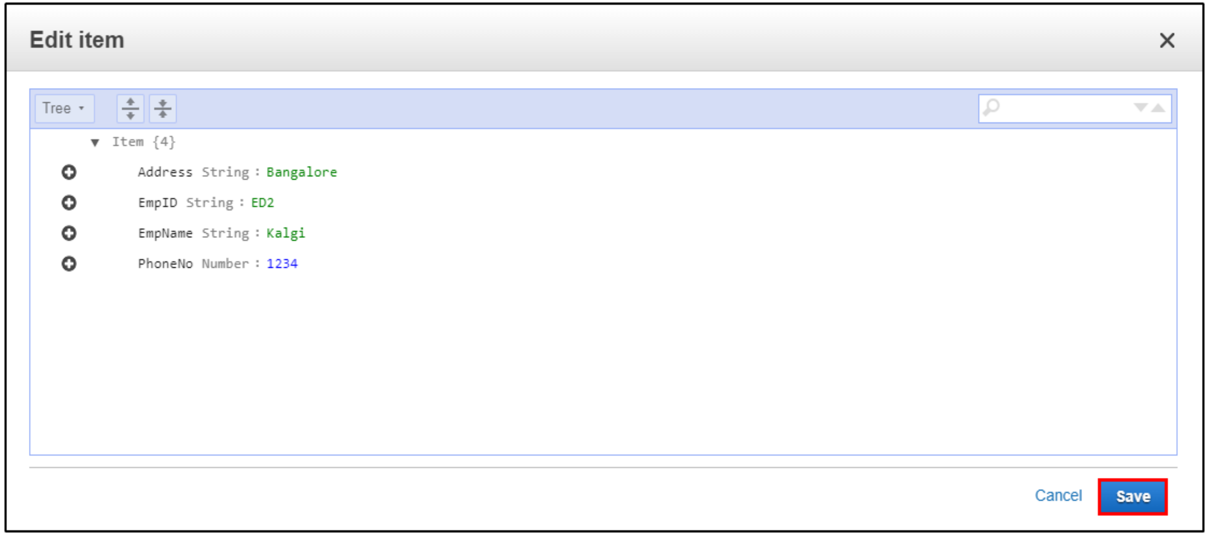
Step 3: This is what it looks like after adding multiple columns to your table. Click on "Save".



```
▼ Item {4}
    Address String : Bangalore
    EmpID String : ED2
    EmpName String : Kalgi
    PhoneNo Number : 1234
```

Demo: Creating, Inserting And Querying A Table In DynamoDB

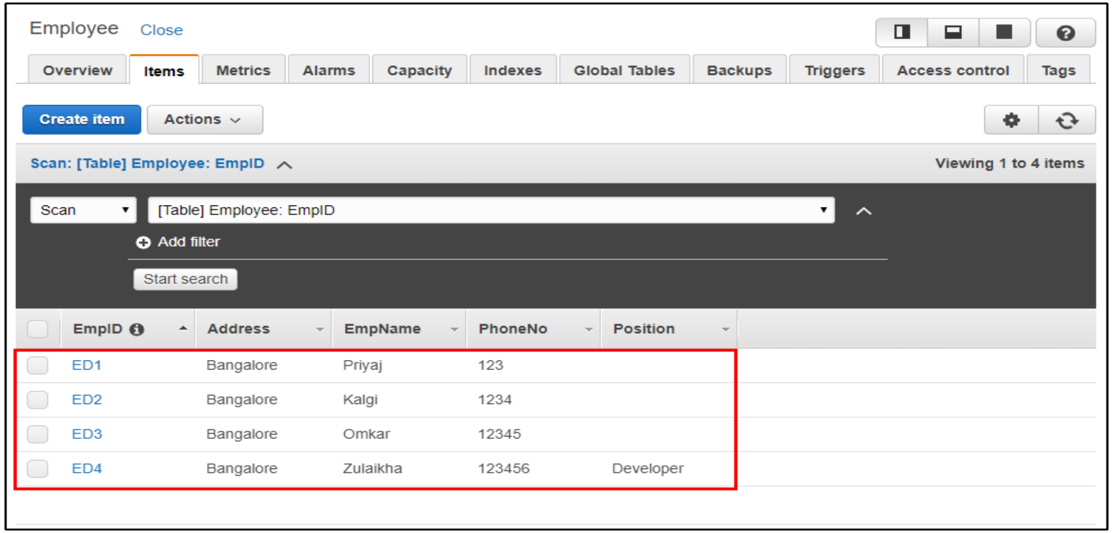
Step 4: Since it is a NoSQL architecture, you can play around with the columns you add to the table. E.g. “Position“.



```
▼ Item {4}
    Address String : Bangalore
    EmpID String : ED2
    EmpName String : Kalgi
    PhoneNo Number : 1234
```

Demo: Creating, Inserting And Querying A Table In DynamoDB

Step 5: This is how your table will look like once you have inserted the data.

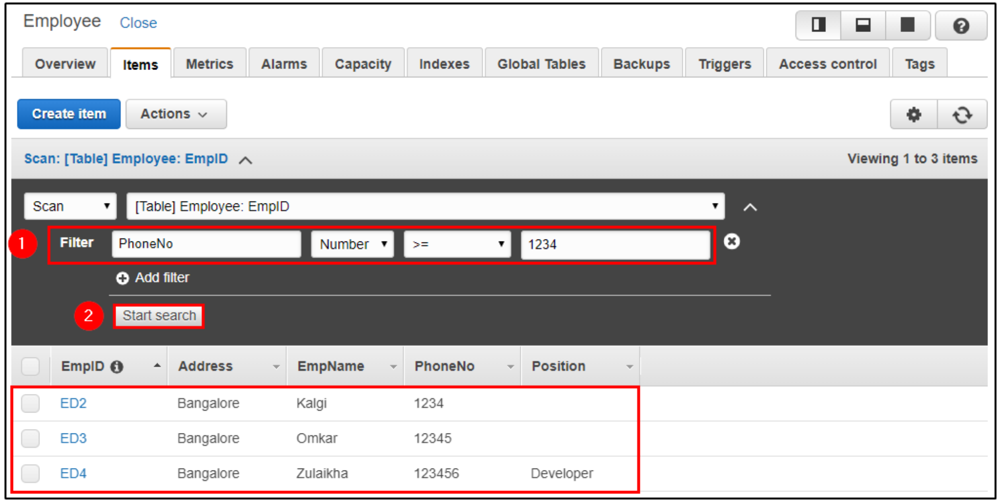


Employee Close
Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Triggers Access control Tags
Create item Actions
Scan: [Table] Employee: EmpID
Viewing 1 to 4 items
Scan [Table] Employee: EmpID
Add filter
Start search

EmpID	Address	EmpName	PhoneNo	Position
ED1	Bangalore	Priyaj	123	
ED2	Bangalore	Kalgi	1234	
ED3	Bangalore	Omkar	12345	
ED4	Bangalore	Zulaikha	123456	Developer

Demo: Creating, Inserting And Querying A Table In DynamoDB

Now that we have a table ready, let's go ahead and look at some basic queries.
Step 1: Here you can frame your query and click on "*Start Search*" to get the desired result.
E.g. I am searching for all the mobile numbers that are greater than or equals to "*1234*"



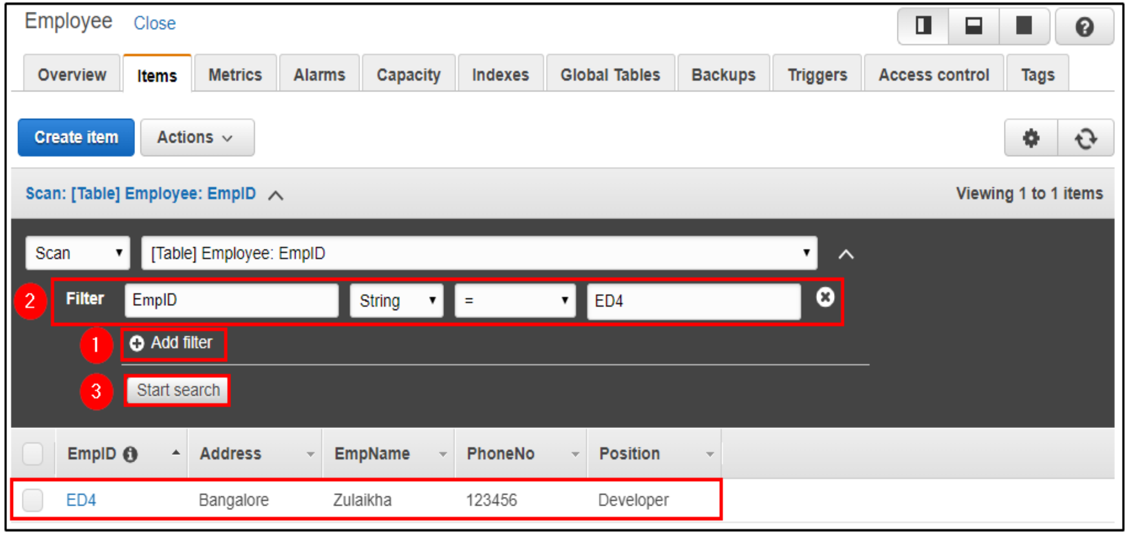
Scan | [Table] Employee: EmpID

Filter | PhoneNo | Number | >= | 1234

EmpID	Address	EmpName	PhoneNo	Position
ED2	Bangalore	Kalgi	1234	
ED3	Bangalore	Omkar	12345	
ED4	Bangalore	Zulaikha	123456	Developer

Demo: Creating, Inserting And Querying A Table In DynamoDB

Step 2: Here, I am searching for the record which has EmpId as “ED4“.



Scan: [Table] Employee: EmpID

Viewing 1 to 1 items

Scan · [Table] Employee: EmpID

2 Filter: EmpID · String · = · ED4

EmpID	Address	EmpName	PhoneNo	Position
ED4	Bangalore	Zulaikha	123456	Developer

Thank You!