Amazon DynamoDB: Fast And Scalable NoSQL Database

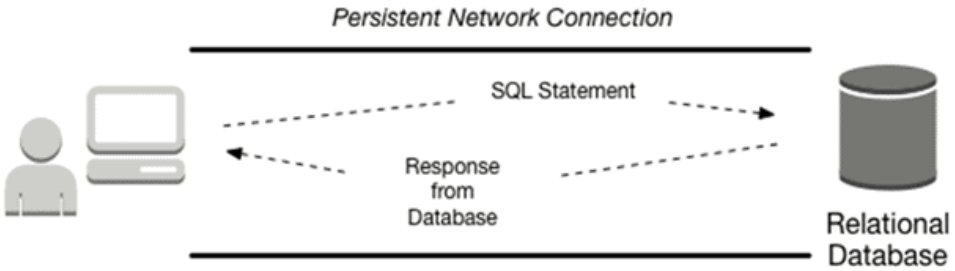
Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multi-region, multi-active, durable database with built-in security, backup and restores, and in-memory caching for internet-scale applications.

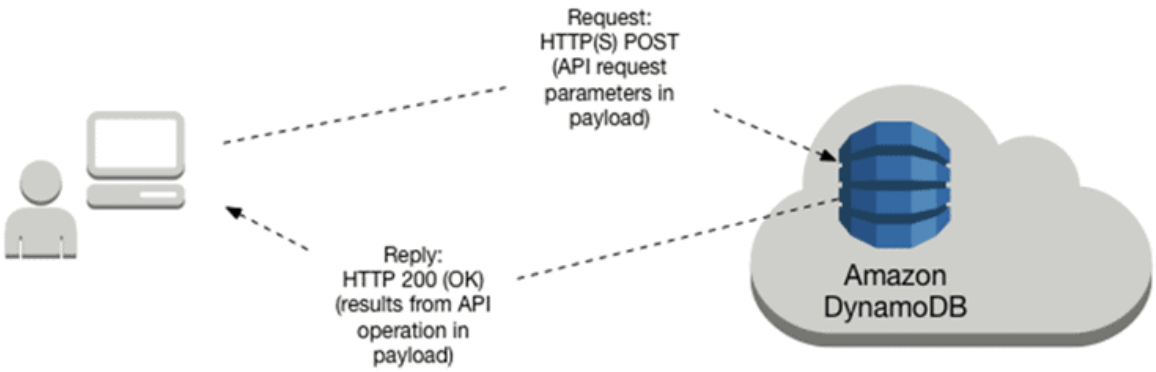
Non-Relational Database

Non-relational databases (often called **NoSQL databases**) are different from traditional relational databases in that they store their data in a non-tabular form**. NoSQL** is a term used to describe nonrelational database systems that are highly available, scalable, and optimized for high performance. Instead of the relational model, NoSQL databases (like DynamoDB) use alternate models for data management, such as key-value pairs or document storage.

What is DynamoDB?

Amazon DynamoDB is a fully managed NoSQL service that works on key-value pair and other data structure documents provided by Amazon. It requires only a primary key and doesn't require a schema to create a table. It can store any amount of data and serve any amount of traffic to any extent. With DynamoDB, you can expect great performance even when it scales up. It's a really simple and small API that follows the key-value method to store, access, and perform advanced data retrieval.



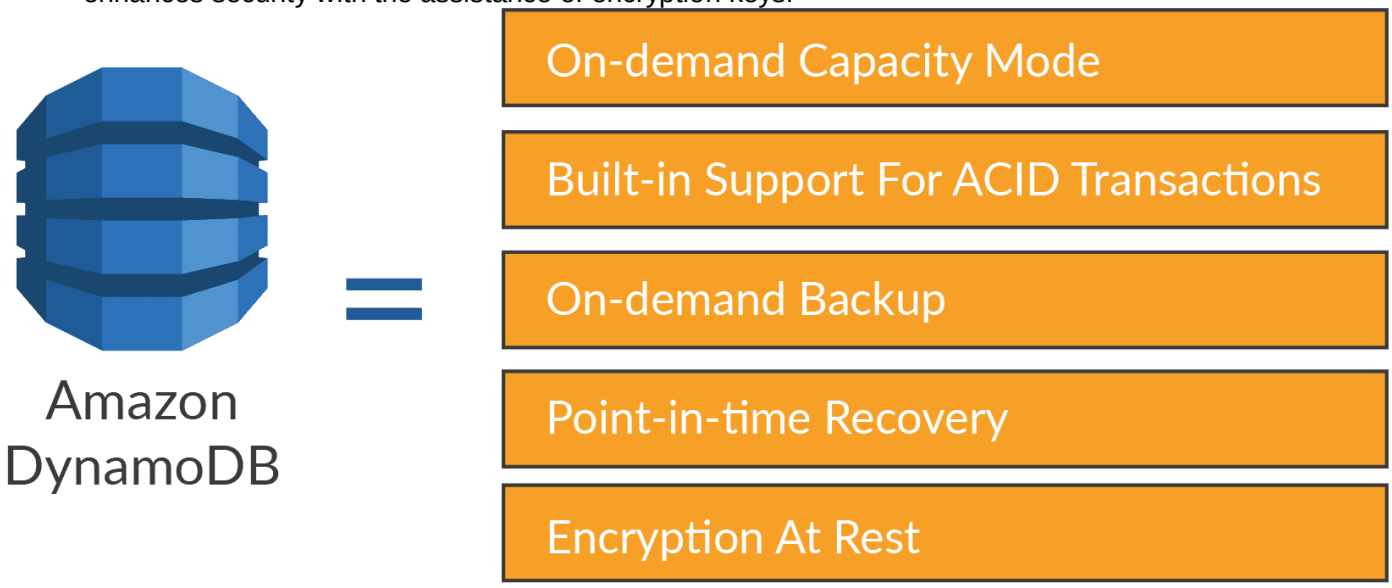


DynamoDB is a web service, and interactions with it are stateless. Applications are not required to maintain persistent network connections. Instead, interaction with DynamoDB occurs using **HTTP(S)** requests and responses. To know more about Databases please check our blog on AWS Database Services.

Features of DynamoDB

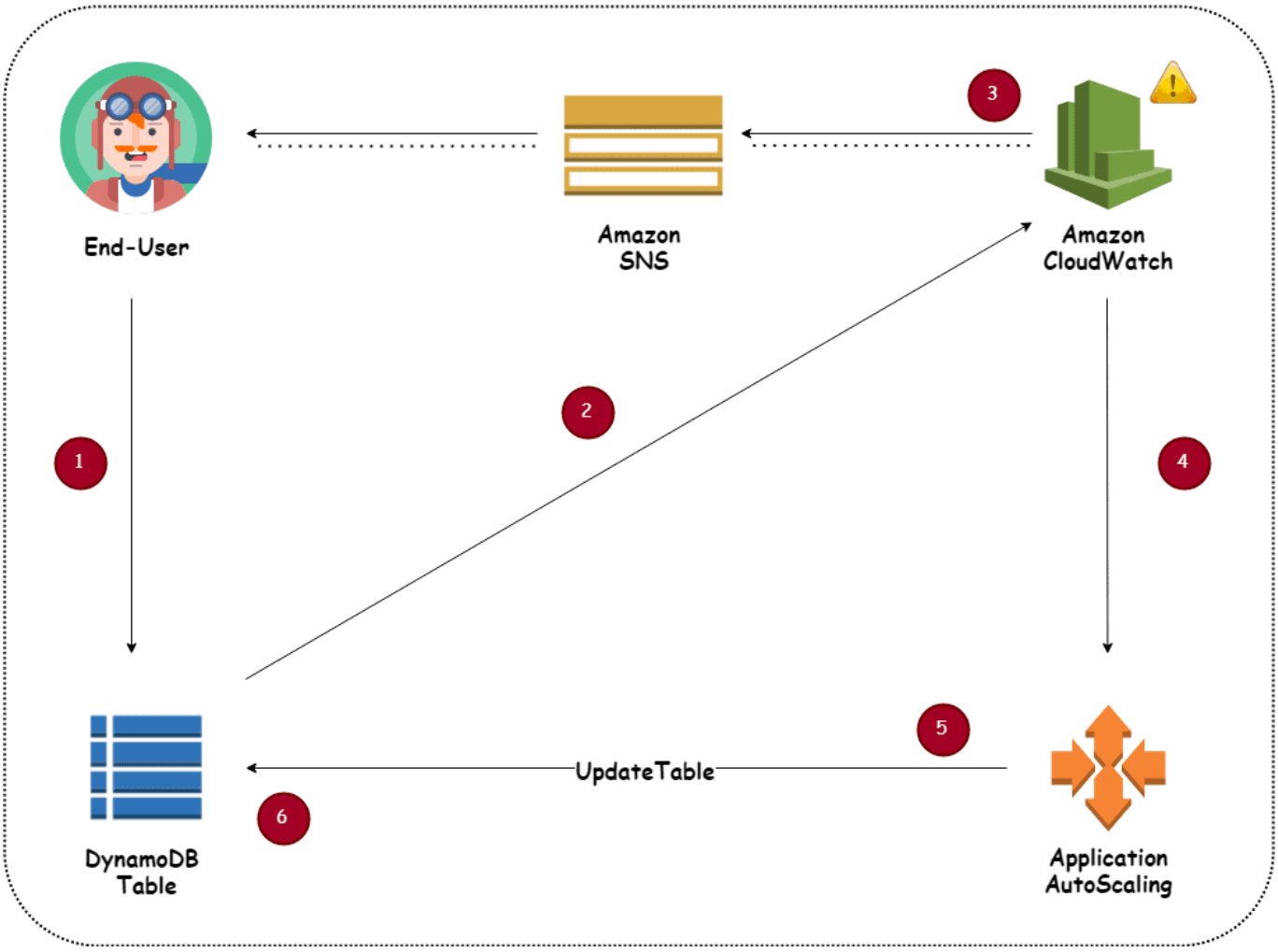
DynamoDB is designed in such a way that the user can get high-performance, run scalable applications that might not be possible with the normal database system. These additional features of DynamoDB are often seen under the subsequent categories:

- **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 make an entire backup of your work om any given point in time.
- **Point-in-time recovery**: This feature helps you with the protection of your data just in case of accidental read/ write operations.
- **Encryption at rest**: It keeps the info encrypted even when the table is not in use. This enhances security with the assistance of encryption keys.



DynamoDB Auto Scaling

Amazon DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns. This enables a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling. When the workload decreases, Application Auto Scaling decreases the throughput in order that you do not pay for unused provisioned capacity.



- You create an Application Auto Scaling policy for your DynamoDB table.
- DynamoDB publishes consumed capacity metrics to Amazon CloudWatch.
- If the table's consumed capacity exceeds your **target utilization** (or falls below the target) for a particular length of time, Amazon **CloudWatch triggers an alarm**. You'll view the alarm on the console and receive notifications using Amazon Simple Notification Service (Amazon SNS).
- The CloudWatch alarm invokes Application Auto Scaling to gauge your scaling policy.
- Application Auto Scaling issues a **UpdateTable** request to regulate your table's provisioned throughput.
- DynamoDB processes the **UpdateTable** request, dynamically increasing (or decreasing) the table's provisioned throughput capacity in order that it approaches your target utilization.

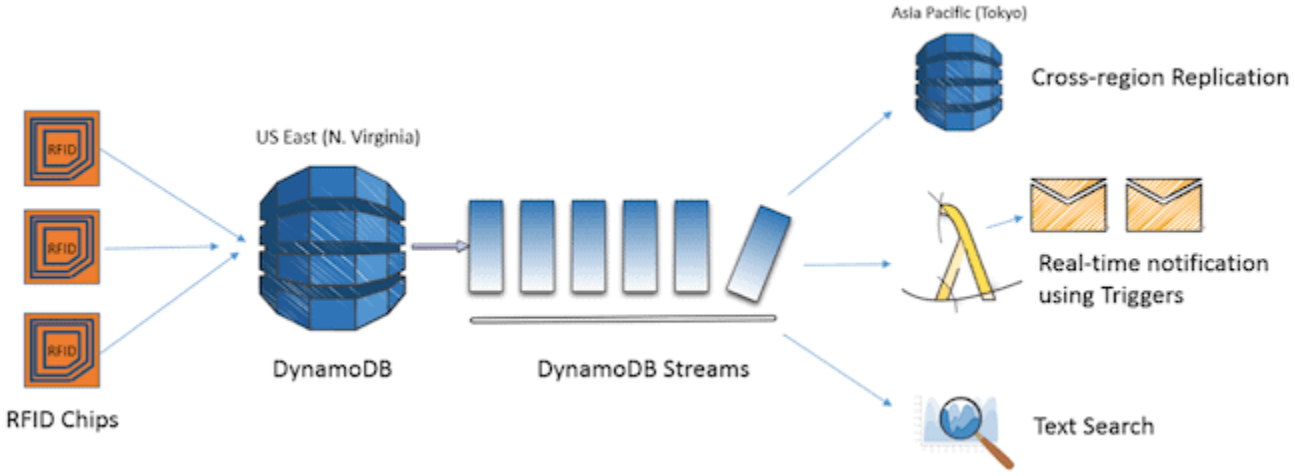
DynamoDB Streams

DynamoDB Streams – an optional feature that captures data modification events in DynamoDB tables.

- Each event is represented by a stream record, and captures the subsequent events:
  A new item is added to the table: captures an image of the entire item, including all of its attributes.
  An item is updated: captures the "before" and "after" images of any attributes that were

modified within the item.
An item is deleted from the table: captures an image of the entire item before it had been deleted.

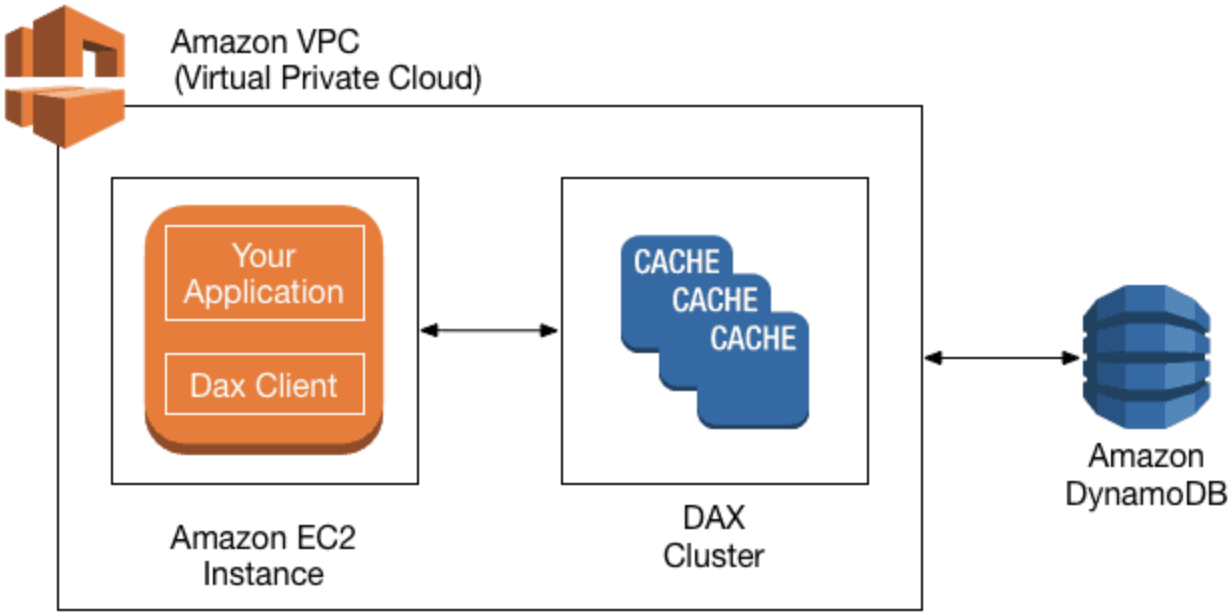
- Stream records are organized into groups or **shards**. Each shard acts as a container for multiple stream records and contains information required for accessing and iterating through these records.
- The naming convention for DynamoDB Streams endpoints is **streams.dynamodb.<region>.amazonaws.com**



- Stream records have a lifetime of 24 hours; then, they're automatically removed from the stream
- You can use DynamoDB Streams alongside with AWS Lambda to create a trigger, which is a code that executes automatically whenever an event of interest appears in a stream.
- DynamoDB Streams enables powerful solutions such as data replication within and across Regions, materialized views of data in DynamoDB tables, data analysis using Kinesis materialized views, and much more.

DynamoDB Accelerators (DAX)

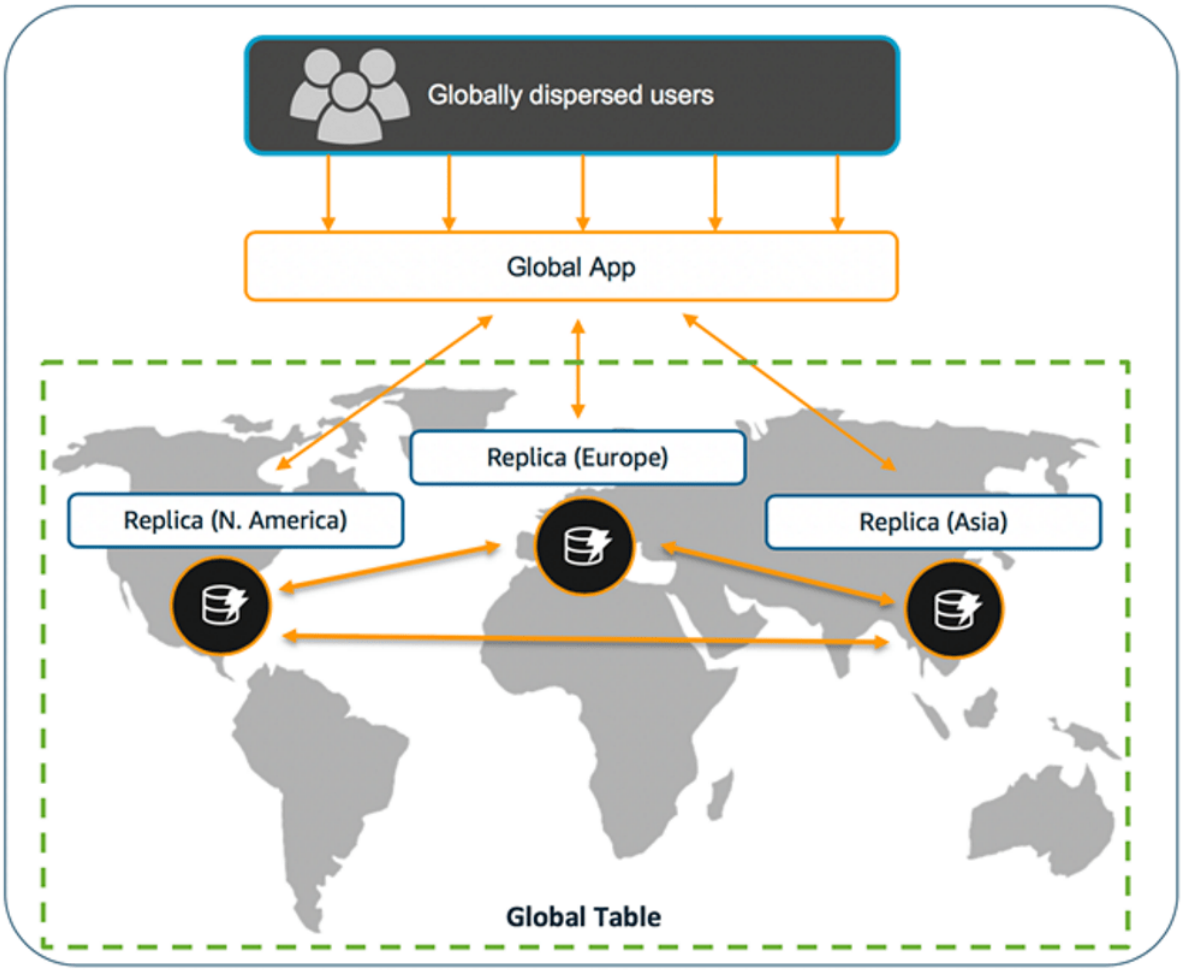
- DAX is a fully managed, highly available, in-memory cache for DynamoDB.
- **DynamoDB Accelerator (DAX)** delivers microsecond response times for accessing eventually consistent data.
- It requires only minimal functional changes to use DAX with an existing application since it is API-compatible with DynamoD3.
- For read-heavy or bursty workloads, DAX provides increased throughput and potential cost savings by reducing the necessity to overprovision read capacity units.
- DAX allows you to scale on-demand.
- DAX supports server-side encryption but not TLS(Transport Layer Security).
- DAX is fully managed. You are not required to do hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed cache cluster, or replicating data over multiple instances as you scale.
- DAX isn't recommended if you require strongly consistent reads.
- DAX is only useful for read-intensive workloads, but not for write-intensive ones.



DynamoDB Global Tables

Global tables build on the global Amazon DynamoDB footprint to provide you with a fully managed, multi-region, and multi-active database that delivers quick, local, read and write performance for massively scaled, global applications. Global tables replicate your DynamoDB tables automatically across any of AWS Regions of your choice.

When you create a DynamoDB global table, it consists of multiple replica tables (one per AWS Region) that DynamoDB treats as a single unit. Every replica has the same table name and same primary-key schema. When an application writes data to a replica table in one Region, DynamoDB propagates the write to the other replica tables within the other AWS Regions automatically.



Capacity Provisioning

1. Read Capacity Units

On-demand mode tables

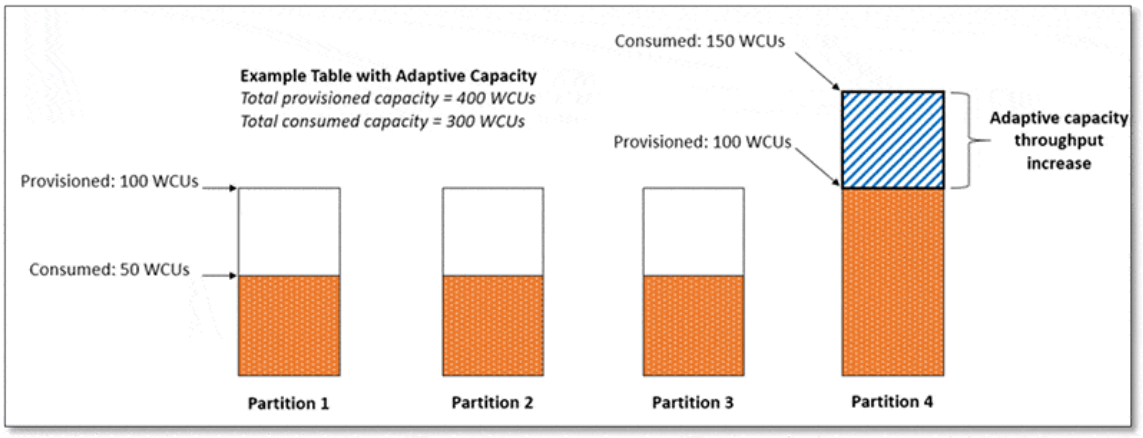
You don't need to specify how much read throughput you expect your application to perform. DynamoDB charges you for the reads that your application performs on your tables in terms of read request units.

- 1 read request unit (RRU) = 1 strongly consistent reads of up to 4 K8/s = 2 eventually consistent reads of up to 4 KB's per read.
- 2 RRUs = 1 transactional read requests (one read per second) for items up to 4 KB.
- For reads on items greater than 4 KB, a total number of reads required = (total item size / 4 KB) rounded up.

Provisioned mode tables

You specify throughput capacity in terms of read capacity units (RCUs).

- 1 read capacity unit 1 strongly consistent reads of up to 4 KB's = 2 eventually consistent reads of up to 4KB's per read.
- 2 RCUs = 1 transactional read requests (one read per second) for items up to 4 KB.
- For reads on items greater than 4 KB, a total number of reads required = (total item size / 4 KB) rounded up.



2. Write Capacity Units

On-demand mode tables

You don't need to specify how much write throughput you expect your application to perform. DynamoDB charges you for the writes that your application performs on your tables in terms of write request units.

- 1 write request unit (WRU) = 1 write of up to 1 KB's.
- 2 WRUs = 1 transactional write request (one write per second) for items up to 1 KB.
- For writes greater than 1 KB, the total number of writes required = (total item size / 1 KB) rounded up

Provisioned mode tables

You specify throughput capacity in terms of write capacity units (WCUs).

- 1 write capacity unit (WCU) = 1 write of up to 1 KB's.
- 2 WCUs = 1 transactional write requests (one write per second) for items up to 1 KB.
- For writes greater than 1 KB, the total number of writes required = (total item size / 1 KB) rounded up