Amazon DynamoDB

General Amazon DynamoDB Concepts

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

Multi-AZ NoSQL data store with Cross-Region Replication option.

Push button scaling means that you can scale the DB at any time without incurring downtime.

Defaults to eventual consistency reads but can request strongly consistent read via SDK parameter.

Priced on throughput, rather than compute.

Provision read and write capacity in anticipation of need.

Autoscale capacity adjusts per configured min/max levels.

On-Demand Capacity provides flexible capacity at a small premium cost.

Can achieve ACID compliance with DynamoDB Transactions.

SSD based and uses limited indexing on attributes for performance.

DynamoDB is a Web service that uses HTTP over SSL (HTTPS) as a transport and JSON as a message serialisation format.

Amazon DynamoDB stores three geographically distributed replicas of each table to enable high availability and data durability.

Data is synchronously replicated across 3 facilities (AZs) in a region.

Cross-region replication allows you to replicate across regions:

* Amazon DynamoDB global tables provides a fully managed solution for deploying a multi-region, multi-master database.
* When you create a global table, you specify the AWS regions where you want the table to be available.
* DynamoDB performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.

Provides low read and write latency.

Scale storage and throughput up or down as needed without code changes or downtime.

DynamoDB is schema-less.

DynamoDB can be used for storing session state.

Provides two read models.

Eventually consistent reads (Default):

* The eventual consistency option maximises your read throughput (best read performance).
* An eventually consistent read might not reflect the results of a recently completed write.
* Consistency across all copies reached within 1 second.

Strongly consistent reads:

* A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read (faster consistency).

Users/applications reading from DynamoDB tables can specify in their requests if they want strong consistency (default is eventually consistent).

Attributes consists of a name and a value or set of values.

Attributes in DynamoDB are similar to fields or columns in other database systems.

The primary key is the only required attribute for items in a table and it uniquely identifies each item.

A primary key can either be one of the following types.

Partition key:

* A simple primary key, composed of one attribute known as the partition key.

Partition key and sort key:

* Referred to as a composite primary key.
* Composed of two attributes: partition key and sort key.

An item is a collection of attributes.

The aggregate size of an item cannot exceed 400KB including keys and all attributes.

Can store pointers to objects in S3, including items over 400KB.

Tables are a collection of items and items are made up of attributes (columns).

Supports key-value and document data structures.

Supports fast, in-place Atomic updates.

Stores structured data in tables, indexed by a primary key.

Supports GET/PUT operations using a user-defined primary key.

DynamoDB provides flexible querying by letting you query on non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.

You can create one or more secondary indexes on a table.

A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key.

DynamoDB supports two kinds of 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 search using one of the following methods:

* Query operation – find items in a table or a secondary index using only the primary keys attributes.
* Scan operation – reads every item in a table or a secondary index and by default will return all items.

Use DynamoDB when relational features are not required and the DB is likely to need to scale.

Not ideal for the following situations:

* Traditional RDS apps.
* Joins and/or complex transactions.
* BLOB data.
* Large data with low I/O rate.

DynamoDB Streams

DynamoDB Streams help you to keep a list of item level changes or provide a list of item level changes that have taken place in the last 24hrs.

Amazon DynamoDB is integrated with AWS Lambda so that you can create triggers—pieces of code that automatically respond to events in DynamoDB Streams.

If you enable DynamoDB Streams on a table, you can associate the stream ARN with a Lambda function that you write.

DynamoDB DAX

Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement .

Improves performance 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.

You do not need to modify application logic, since DAX is compatible with existing DynamoDB API calls.

You can enable DAX with just a few clicks in the AWS Management Console or using the AWS SDK.

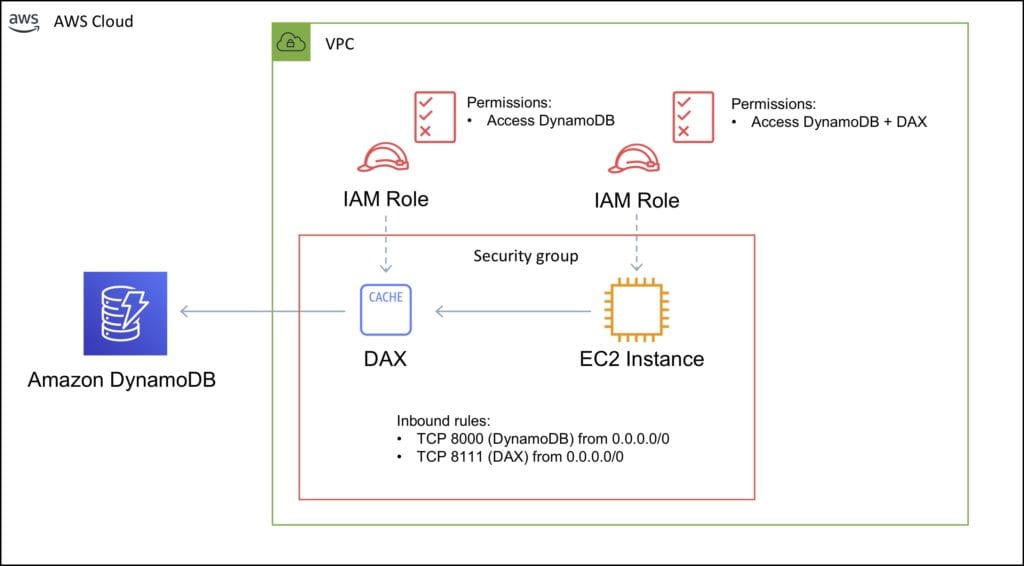
Just as with DynamoDB, you only pay for the capacity you provision.

Provisioned through clusters and charged by the node (runs on EC2 instances).

Pricing is per node-hour consumed and is dependent on the instance type you select.

The following diagram depicts the Amazon DynamoDB DAX service. Note the following:

* You can apply an IAM role to the the DAX nodes
* You can apply Security Groups to the DAX nodes
* DynamoDB DAX sits within your VPC



Best practices

Keep item sizes small.

If you are storing serial data in DynamoDB that will require actions based on date/time use separate tables for days, weeks, months.

Store more frequently and less frequently accessed data in separate tables.

If possible compress larger attribute values.

Store objects larger than 400KB in S3 and use pointers (S3 Object ID) in DynamoDB.

Integrations

ElastiCache can be used in front of DynamoDB for performance of reads on infrequently changed data.

Triggers integrate with AWS Lambda to respond to triggers.

Integration with RedShift:

* RedShift complements DynamoDB with advanced business intelligence.
* When copying data from a DynamoDB table into RedShift you can perform complex data analysis queries including joins with other tables.
* A copy operation from a DynamoDB table counts against the table’s read capacity.
* After data is copied, SQL queries do not affect the data in DynamoDB.

DynamoDB is integrated with Apache Hive on EMR. Hive can allow you to:

* Read and write data in DynamoDB tables allowing you to query DynamoDB data using a SQL-like language (HiveQL).
* Copy data from a DynamoDB table to an S3 bucket and vice versa.
* Copy data from a DynamoDB table into HDFS and vice versa.
* Perform join operations on DynamoDB tables.

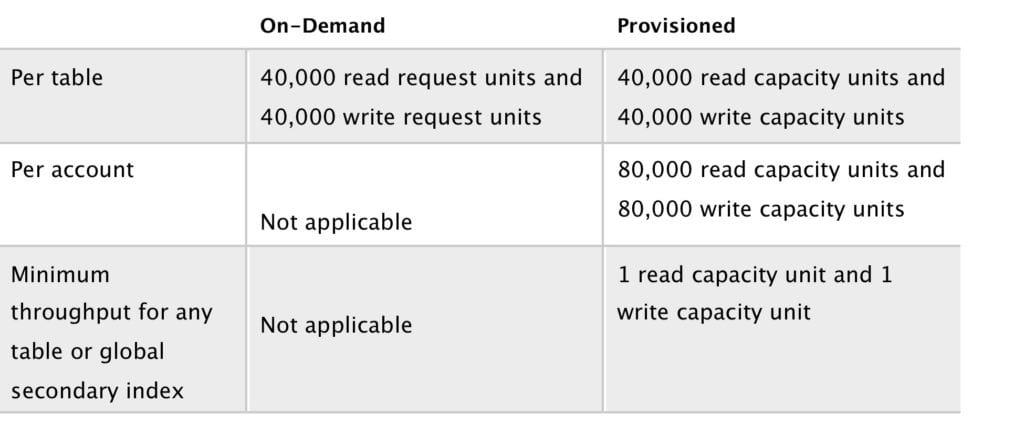
Scalability

Push button scaling without downtime.

You can scale down only 4 times per calendar day.

AWS places some default limits on the throughput you can provision.

These are the limits unless you request a higher amount:



DynamoDB can throttle requests that exceed the provisioned throughput for a table.

DynamoDB can also throttle read requests for an Index to prevent your application from consuming too many capacity units.

When a request is throttled it fails with an HTTP 400 code (Bad Request) and a ProvisionedThroughputExceeded exception.

Cross Region Replication with Global Tables

Amazon DynamoDB global tables provide a fully managed solution for deploying a multi-region, multi-master database.

When you create a global table, you specify the AWS regions where you want the table to be available.

DynamoDB performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.

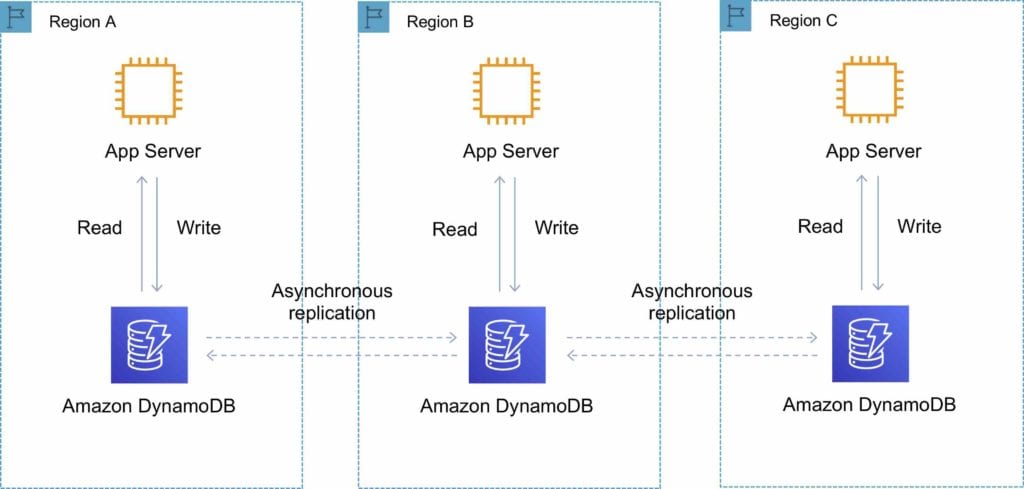
DynamoDB global tables are ideal for massively scaled applications, with globally dispersed users.

Global tables provide automatic multi-master replication to AWS regions world-wide, so you can deliver low-latency data access to your users no matter where they are located.

Definitions:

* **A *global table***is a collection of one or more replica tables, all owned by a single AWS account.
* **A *replica table*** (or *replica*, for short) is a single DynamoDB table that functions as a part of a global table. Each replica stores the same set of data items. Any given global table can only have one replica table per region.

The following digram depicts the Amazon DynamoDB Global Tables topology:



You can add replica tables to the global table, so that it can be available in additional AWS regions.

With a global table, each replica table stores the same set of data items. DynamoDB does not support partial replication of only some of the items.

An application can read and write data to any replica table. If your application only uses eventually consistent reads, and only issues reads against one AWS region, then it will work without any modification.

However, if your application requires strongly consistent reads, then it must perform all of its strongly consistent reads and writes in the same region. DynamoDB does not support strongly consistent reads across AWS regions.

It is important that each replica table and secondary index in your global table has identical write capacity settings to ensure proper replication of data.

DynamoDB Auto Scaling

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 so that you don’t pay for unused provisioned capacity.

How Application Auto Scaling works:

* You create a *scaling policy* for a table or a global secondary index.
* The scaling policy specifies whether you want to scale read capacity or write capacity (or both), and the minimum and maximum provisioned capacity unit settings for the table or index.
* The scaling policy also contains a *target utilization*—the percentage of consumed provisioned throughput at a point in time.
* Uses a *target tracking* algorithm to adjust the provisioned throughput of the table (or index) upward or downward in response to actual workloads, so that the actual capacity utilization remains at or near your target utilization.

Currently, Auto Scaling does not scale down your provisioned capacity if your table’s consumed capacity becomes zero.

If you use the AWS Management Console to create a table or a global secondary index, DynamoDB auto scaling is enabled by default.

Limits

256 tables per account per region.

No limit on the size of a table.

Read/write capacity unit limits vary per region.

Capacity units

One read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per second for items up to 4KB.

For items larger than 4KB, DynamoDB consumes additional read capacity units.

One write capacity unit represents one write per second for an item up to 1KB.

Charges

DynamoDB charges for reading, writing, and storing data in your DynamoDB tables, along with any optional features you choose to enable.

There are two pricing models for DynamoDB:

* **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.
* **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 cost.

Additional charges include:

* Data transfer out.
* Backups per GB (continuous or on-demand).
* Global Tables.
* DynamoDB Accelerator (DAX).
* DynamoDB Streams.

High Availability Approaches for Databases

If possible, choose DynamoDB over RDS because of inherent fault tolerance.

If DynamoDB can’t be used, choose Aurora because of redundancy and automatic recovery features.

If Aurora can’t be used, choose Multi-AZ RDS.

Frequent RDS snapshots can protect against data corruption or failure and they won’t impact performance of Multi-AZ deployment.

Regional replication is also an option, but will not be strongly consistent.

If the database runs on EC2, you have to design the HA yourself.