Frequently Asked Questions About Amazon DynamoDB

## **What is Amazon DynamoDB?**

Q: What is Amazon DynamoDB?

[Amazon DynamoDB](https://aws.amazon.com/dynamodb/) is a fully managed [NoSQL database](https://aws.amazon.com/nosql/) service that provides fast and predictable performance with seamless scalability. DynamoDB enables customers to offload the administrative burdens of operating and scaling distributed databases to AWS so that they don’t have to worry about hardware provisioning, setup and configuration, throughput capacity planning, replication, software patching, or cluster scaling.

Q: What does Amazon DynamoDB manage on my behalf?

Amazon DynamoDB takes away one of the main stumbling blocks of scaling databases: the management of database software and the provisioning of the hardware needed to run it. You can deploy a nonrelational database in a matter of minutes. DynamoDB automatically scales throughput capacity to meet workload demands and partitions and repartitions your data as your table size grows. In addition, DynamoDB synchronously replicates data across three facilities in an AWS Region, giving you high availability and data durability.

Q: What does read consistency mean? Why should I care?

Amazon DynamoDB stores three geographically distributed replicas of each table to enable high availability and data durability. Read consistency represents the manner and timing in which the successful write or update of a data item is reflected in a subsequent read operation of that same item. DynamoDB exposes logic that enables you to specify the consistency characteristics you desire for each read request within your application.

Q: What is the consistency model of Amazon DynamoDB?

When reading data from Amazon DynamoDB, users can specify whether they want the read to be eventually consistent or strongly consistent:

* Eventually consistent reads (Default) – The eventual consistency option maximizes your read throughput. However, an eventually consistent read might not reflect the results of a recently completed write. Consistency across all copies of data is usually reached within a second. Repeating a read after a short time should return the updated data.
* Strongly consistent reads — In addition to eventual consistency, Amazon DynamoDB also gives you the flexibility and control to request a strongly consistent read if your application, or an element of your application, requires it. A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read.

Q: Does DynamoDB support in-place atomic updates?

Amazon DynamoDB supports fast, in-place updates. You can increment or decrement a numeric attribute in a row using a single API call. Similarly, you can atomically add or remove sets, lists, or maps. For more information about atomic updates, see [Atomic Counters](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithItems.html#WorkingWithItems.AtomicCounters).

Q: Why is Amazon DynamoDB built on solid-state drives?

Amazon DynamoDB runs exclusively on solid-state drives (SSDs). SSDs help AWS achieve the design goals of predictable low-latency response times for storing and accessing data at any scale. The high I/O (high reads/second and writes/second) performance of SSDs also enables us to serve high-scale request workloads cost-efficiently, and to pass this efficiency along in low request pricing.

Q: The storage cost of DynamoDB seems high. Is this a cost-effective service?

As with any product, we encourage potential customers of Amazon DynamoDB to consider the total cost of a solution, not just a single pricing dimension. The total cost of servicing a database workload is a function of the request traffic requirements and the amount of data stored. Most database workloads are characterized by a requirement for high I/O (high reads/second and writes/second) per GB stored. DynamoDB is built on SSD drives, which raises the cost per GB stored, relative to spinning media, but it also allows us to offer very low request costs. Based on what we see in typical database workloads, we believe that the total bill for using SSD-based DynamoDB will usually be lower than the cost of using a typical spinning media-based relational or nonrelational database. If you need to store a large amount of data that you rarely access, DynamoDB may not be right for you. We recommend that you use [Amazon S3](https://aws.amazon.com/s3/)for such use cases.

You also should note that the storage cost reflects the cost of storing multiple copies of each data item across multiple facilities in an AWS Region.

Q: Is DynamoDB only for high-scale applications?

No. Amazon DynamoDB offers seamless scaling so that you can scale automatically as your application requirements increase. If you need fast, predictable performance at any scale, DynamoDB may be the right choice for you.

## **Getting started**

Q: How do I get started with Amazon DynamoDB?

Click “Sign Up” to get started with Amazon DynamoDB today. From there, you can begin interacting with Amazon DynamoDB using either the [AWS Management Console](https://aws.amazon.com/console/) or Amazon DynamoDB APIs. If you are using the AWS Management Console, you can create a table with Amazon DynamoDB and begin exploring with just a few clicks.

Q: What kind of query functionality does DynamoDB support?

Amazon DynamoDB supports GET/PUT operations using a user-defined primary key. The primary key is the only required attribute for items in a table and it uniquely identifies each item. You specify the primary key when you create a table. In addition to that DynamoDB provides flexible querying by letting you query on non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.

A primary key can either be a single-attribute partition key or a composite partition-sort key. A single attribute partition primary key could be, for example, “UserID”. This would allow you to quickly read and write data for an item associated with a given user ID.

A composite partition-sort key is indexed as a partition key element and a sort key element. This multi-part key maintains a hierarchy between the first and second element values. For example, a composite partition-sort key could be a combination of “UserID” (partition) and “Timestamp” (sort). Holding the partition key element constant, you can search across the sort key element to retrieve items. This would allow you to use the Query API to, for example, retrieve all items for a single UserID across a range of timestamps.

For more information on Global Secondary Indexing and its query capabilities, see the [Secondary Indexes](https://aws.amazon.com/dynamodb/faqs/#gsi_anchor) section in FAQ.

Q: How do I update and query data items with Amazon DynamoDB?

After you have created a table using the AWS Management Console or CreateTable API, you can use the PutItem or BatchWriteItem APIs to insert items. Then you can use the GetItem, BatchGetItem, or, if composite primary keys are enabled and in use in your table, the Query API to retrieve the item(s) you added to the table.

Q: Does Amazon DynamoDB support conditional operations?

Yes, you can specify a condition that must be satisfied for a put, update, or delete operation to be completed on an item. To perform a conditional operation, you can define a ConditionExpression that is constructed from the following:

* Boolean functions: ATTRIBUTE\_EXIST, CONTAINS, and BEGINS\_WITH
* Comparison operators: =, <>, <, >, <=, >=, BETWEEN, and IN
* Logical operators: NOT, AND, and OR.

You can construct a free-form conditional expression that combines multiple conditional clauses, including nested clauses. Conditional operations allow users to implement optimistic concurrency control systems on DynamoDB. For more information on conditional operations, [please see our documentation](http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/WorkingWithDDItems.html).

Q: Are expressions supported for key conditions?

Yes, you can specify an expression as part of the Query API call to filter results based on values of primary keys on a table using the KeyConditionExpression parameter.

Q: Are expressions supported for partition and partition-sort keys?

Yes, you can use expressions for both partition and partition-sort keys. Refer to the [documentation page](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/QueryAndScan.html) for more information on which expressions work on partition and partition-sort keys.

Q: Does Amazon DynamoDB support increment or decrement operations?

Yes, Amazon DynamoDB allows atomic increment and decrement operations on scalar values.

Q: When should I use Amazon DynamoDB vs a relational database engine on Amazon RDS or Amazon EC2?

Today’s web-based applications generate and consume massive amounts of data. For example, an online game might start out with only a few thousand users and a light database workload consisting of 10 writes per second and 50 reads per second. However, if the game becomes successful, it may rapidly grow to millions of users and generate tens (or even hundreds) of thousands of writes and reads per second. It may also create terabytes or more of data per day. Developing your applications against Amazon DynamoDB enables you to start small and simply dial-up your request capacity for a table as your requirements scale, without incurring downtime. You pay highly cost-efficient rates for the request capacity you provision, and let Amazon DynamoDB do the work over partitioning your data and traffic over sufficient server capacity to meet your needs. Amazon DynamoDB does the database management and administration, and you simply store and request your data. Automatic replication and failover provides built-in fault tolerance, high availability, and data durability. Amazon DynamoDB gives you the peace of mind that your database is fully managed and can grow with your application requirements.

While Amazon DynamoDB tackles the core problems of database scalability, management, performance, and reliability, the datamodel, just like any NoSQL, must be designed specifically for the access patterns required by the application. In other words, running adhoc queries on DynamoDB can be inefficient. Refer to the design guidance that shows how to effectively migrate from any Relational database to DynamoDB. If your workload requires this functionality, or you are looking for compatibility with an existing relational engine, you may wish to run a relational engine on Amazon RDS or Amazon EC2. While relational database engines provide robust features and functionality, scaling a workload beyond a single relational database instance is highly complex and requires significant time and expertise. As such, if you anticipate scaling requirements for your new application and do not need relational features, Amazon DynamoDB may be the best choice for you.

Q: How does Amazon DynamoDB differ from Amazon SimpleDB?

Which should I use? Both services are non-relational databases that remove the work of database administration. Amazon DynamoDB focuses on providing seamless scalability and fast, predictable performance. It runs on solid state disks (SSDs) for low-latency response times, and there are no limits on the request capacity or storage size for a given table. This is because Amazon DynamoDB automatically partitions your data and workload over a sufficient number of servers to meet the scale requirements you provide. In contrast, a table in Amazon SimpleDB has a strict storage limitation of 10 GB and is limited in the request capacity it can achieve (typically under 25 writes/second); it is up to you to manage the partitioning and re-partitioning of your data over additional SimpleDB tables if you need additional scale. While SimpleDB has scaling limitations, it may be a good fit for smaller workloads that require query flexibility. Amazon SimpleDB automatically indexes all item attributes and thus supports query flexibility at the cost of performance and scale.

[Amazon CTO Werner Vogels' DynamoDB blog post](http://allthingsdistributed.com/2012/01/amazon-dynamodb.html) provides additional context on the evolution of non-relational database technology at Amazon.

Q: When should I use Amazon DynamoDB vs Amazon S3?

Amazon DynamoDB stores structured data, indexed by primary key, and allows low latency read and write access to items ranging from 1 byte up to 400KB. Amazon S3 stores unstructured blobs and suited for storing large objects up to 5 TB. In order to optimize your costs across AWS services, large objects or infrequently accessed data sets should be stored in Amazon S3, while smaller data elements or file pointers (possibly to Amazon S3 objects) are best saved in Amazon DynamoDB.

Q: Can DynamoDB be used by applications running on any operating system?

Yes. DynamoDB is a fully managed cloud service that you access via API. DynamoDB can be used by applications running on any operating system (e.g. Linux, Windows, iOS, Android, Solaris, AIX, HP-UX, etc.). We recommend using the AWS SDKs to get started with DynamoDB. You can find a list of the AWS SDKs on our [Developer Resources](http://aws.amazon.com/dynamodb/developer-resources/) page. If you have trouble installing or using one of our SDKs, please let us know by posting to the relevant [AWS Forum](https://forums.aws.amazon.com/index.jspa).

## **Data models and APIs**

Q: What is the Data Model?

The data model for Amazon DynamoDB is as follows:

Table: A table is a collection of data items – just like a table in a relational database is a collection of rows. Each table can have an infinite number of data items. Amazon DynamoDB is schema-less, in that the data items in a table need not have the same attributes or even the same number of attributes. Each table must have a primary key. The primary key can be a single attribute key or a “composite” attribute key that combines two attributes. The attribute(s) you designate as a primary key must exist for every item as primary keys uniquely identify each item within the table.

Item: An Item is composed of a primary or composite key and a flexible number of attributes. There is no explicit limitation on the number of attributes associated with an individual item, but the aggregate size of an item, including all the attribute names and attribute values, cannot exceed 400KB.

Attribute: Each attribute associated with a data item is composed of an attribute name (e.g. “Color”) and a value or set of values (e.g. “Red” or “Red, Yellow, Green”). Individual attributes have no explicit size limit, but the total value of an item (including all attribute names and values) cannot exceed 400KB.

Q: Is there a limit on the size of an item?

The total size of an item, including attribute names and attribute values, cannot exceed 400KB. Refer to the design guidance for using ‘Composite Sort Keys’ to design for items that exceed the 400K limit.

Q: Is there a limit on the number of attributes an item can have?

There is no limit to the number of attributes that an item can have. However, the total size of an item, including attribute names and attribute values, cannot exceed 400KB.

Q: What are the APIs?

* CreateTable – Creates a table and specifies the primary index used for data access.
* UpdateTable – Updates the provisioned throughput values for the given table.
* DeleteTable – Deletes a table.
* DescribeTable – Returns table size, status, and index information.
* ListTables – Returns a list of all tables associated with the current account and endpoint.
* PutItem – Creates a new item, or replaces an old item with a new item (including all the attributes). If an item already exists in the specified table with the same primary key, the new item completely replaces the existing item. You can also use conditional operators to replace an item only if its attribute values match certain conditions, or to insert a new item only if that item doesn’t already exist.
* BatchWriteItem – Inserts, replaces, and deletes multiple items across multiple tables in a single request, but not as a single transaction. Supports batches of up to 25 items to Put or Delete, with a maximum total request size of 16 MB.
* UpdateItem – Edits an existing item's attributes. You can also use conditional operators to perform an update only if the item’s attribute values match certain conditions.
* DeleteItem – Deletes a single item in a table by primary key. You can also use conditional operators to perform a delete an item only if the item’s attribute values match certain conditions.
* GetItem – The GetItem operation returns a set of Attributes for an item that matches the primary key. The GetItem operation provides an eventually consistent read by default. If eventually consistent reads are not acceptable for your application, use ConsistentRead.
* BatchGetItem – The BatchGetItem operation returns the attributes for multiple items from multiple tables using their primary keys. A single response has a size limit of 16 MB and returns a maximum of 100 items. Supports both strong and eventual consistency.
* Query –  Gets one or more items using the table primary key, or from a secondary index using the index key. You can narrow the scope of the query on a table by using comparison operators or expressions. You can also filter the query results using filters on non-key attributes. Supports both strong and eventual consistency. A single response has a size limit of 1 MB.
* Scan – Gets all items and attributes by performing a full scan across the table or a secondary index. You can limit the return set by specifying filters against one or more attributes.

Q: What is the consistency model of the Scan operation?

The Scan operation supports eventually consistent and consistent reads. By default, the Scan operation is eventually consistent. However, you can modify the consistency model using the optional ConsistentRead parameter in the Scan API call. Setting the ConsistentRead parameter to true will enable you make consistent reads from the Scan operation. For more information, read the [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Scan.html) for the Scan operation.

Q: How does the Scan operation work?

You can think of the Scan operation as an iterator. Once the aggregate size of items scanned for a given Scan API request exceeds a 1 MB limit, the given request will terminate and fetched results will be returned along with a LastEvaluatedKey (to continue the scan in a subsequent operation).

Q: Are there any limitations for a Scan operation?

A Scan operation on a table or secondary index has a limit of 1MB of data per operation. After the 1MB limit, it stops the operation and returns the matching values up to that point, and a LastEvaluatedKey to apply in a subsequent operation, so that you can pick up where you left off.

Q: How many read capacity units does a Scan operation consume?

The read units required is the number of bytes fetched by the scan operation, rounded to the nearest 4KB, divided by 4KB. Scanning a table with consistent reads consumes twice the read capacity as a scan with eventually consistent reads.

Q: What data types does DynamoDB support?

DynamoDB supports four scalar data types: Number, String, Binary, and Boolean. Additionally, DynamoDB supports collection data types: Number Set, String Set, Binary Set, heterogeneous List and heterogeneous Map. DynamoDB also supports NULL values.

Q: What types of data structures does DynamoDB support?

DynamoDB supports key-value and document data structures.

Q: What is a key-value store?

A key-value store is a database service that provides support for storing, querying and updating collections of objects that are identified using a key and values that contain the actual content being stored.

Q: What is a document store?

A document store provides support for storing, querying and updating items in a document format such as JSON, XML, and HTML.

Q: Does DynamoDB have a JSON data type?

No, but you can use the document SDK to pass JSON data directly to DynamoDB. DynamoDB’s data types are a superset of the data types supported by JSON. The document SDK will automatically map JSON documents onto native DynamoDB data types.

Q: Can I use the AWS Management Console to view and edit JSON documents?

Yes. The AWS Management Console provides a simple UI for exploring and editing the data stored in your DynamoDB tables, including JSON documents. To view or edit data in your table, please log in to the AWS Management Console, choose DynamoDB, select the table you want to view, then click on the “Explore Table” button.

Q: Is querying JSON data in DynamoDB any different?

No. You can create a Global Secondary Index or Local Secondary Index on any top-level JSON element. For example, suppose you stored a JSON document that contained the following information about a person: First Name, Last Name, Zip Code, and a list of all of their friends. First Name, Last Name and Zip code would be top-level JSON elements. You could create an index to let you query based on First Name, Last Name, or Zip Code. The list of friends is not a top-level element, therefore you cannot index the list of friends. For more information on Global Secondary Indexing and its query capabilities, see the [Secondary Indexes section](https://aws.amazon.com/dynamodb/faqs/#gsi_anchor) in this FAQ.

Q: If I have nested JSON data in DynamoDB, can I retrieve only a specific element of that data?

Yes. When using the GetItem, BatchGetItem, Query, or Scan APIs, you can define a ProjectionExpression to determine which attributes should be retrieved from the table. Those attributes can include scalars, sets, or elements of a JSON document.

Q. If I have nested JSON data in DynamoDB, can I update only a specific element of that data?

Yes. When updating a DynamoDB item, you can specify the sub-element of the JSON document that you want to update.

Q:What is the Document SDK?

The Document SDK is a datatypes wrapper for JavaScript that allows easy interoperability between JS and DynamoDB datatypes. With this SDK, wrapping for requests will be handled for you; similarly for responses, datatypes will be unwrapped. For more information and downloading the SDK see our GitHub respository [here](https://github.com/awslabs/dynamodb-document-js-sdk).

## **Scalability, availability, and durability**

Q: Is there a limit to how much data I can store in Amazon DynamoDB?

No. There is no limit to the amount of data you can store in an Amazon DynamoDB table. As the size of your data set grows, Amazon DynamoDB will automatically spread your data over sufficient machine resources to meet your storage requirements.

Q: Is there a limit to how much throughput I can get out of a single table?

No, you can increase the maximum capacity limit setting for Auto Scaling or increase the throughput you have manually provisioned for your table using the API or the AWS Management Console. DynamoDB is able to operate at massive scale and there is no theoretical limit on the maximum throughput you can achieve. DynamoDB automatically divides your table across multiple partitions, where each partition is an independent parallel computation unit. DynamoDB can achieve increasingly high throughput rates by adding more partitions.

If you wish to exceed throughput rates of 10,000 writes/second or 10,000 reads/second, you must first [contact Amazon through this online form](http://portal.aws.amazon.com/gp/aws/html-forms-controller/DynamoDB_Limit_Increase_Form).

Q: Does Amazon DynamoDB remain available when Auto Scaling triggers scaling or when I ask it to scale up or down by changing the provisioned throughput?

Yes. Amazon DynamoDB is designed to scale its provisioned throughput up or down while still remaining available, whether managed by Auto Scaling or manually.

Q: Do I need to manage client-side partitioning on top of Amazon DynamoDB?

No. Amazon DynamoDB removes the need to partition across database tables for throughput scalability.

Q: How highly available is Amazon DynamoDB?

The service runs across Amazon’s proven, high-availability data centers. The service replicates data across three facilities in an AWS Region to provide fault tolerance in the event of a server failure or Availability Zone outage.

Q: How does Amazon DynamoDB achieve high uptime and durability?

To achieve high uptime and durability, Amazon DynamoDB synchronously replicates data across three facilities within an AWS Region.

## **Auto Scaling**

Q. What is DynamoDB Auto Scaling?  
  
DynamoDB Auto Scaling is a fully managed feature that automatically scales up or down provisioned read and write capacity of a DynamoDB table or a global secondary index, as application requests increase or decrease.  
  
Q. Why do I need to use Auto Scaling?

Auto Scaling eliminates the guesswork involved in provisioning adequate capacity when creating new tables and reduces the operational burden of continuously monitoring consumed throughput and adjusting provisioned capacity manually. Auto Scaling helps ensure application availability and reduces costs from unused provisioned capacity.  
  
Q. What application request patterns and workload are suited for Auto Scaling?  
  
Auto Scaling is ideally suited for request patterns that are uniform, predictable, with sustained high and low throughput usage that lasts for several minutes to hours.  
  
Q. How can I enable Auto Scaling for a DynamoDB table or global secondary index?  
  
From the DynamoDB console, when you create a new table, leave the 'Use default settings' option checked, to enable Auto Scaling and apply the same settings for global secondary indexes for the table. If you uncheck 'Use default settings', you can either set provisioned capacity manually or enable Auto Scaling with custom values for target utilization and minimum and maximum capacity. For existing tables, you can enable Auto Scaling or change existing Auto Scaling settings by navigating to the 'Capacity' tab and for indexes, you can enable Auto Scaling from under the 'Indexes' tab. Auto Scaling can also be programmatically managed using CLI or AWS SDK. Please refer to the [DynamoDB developer guide](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.html) to learn more.  
  
Q. What are settings I can configure for Auto Scaling?  
  
There are three configurable settings for Auto Scaling: Target Utilization, the percentage of actual consumed throughput to total provisioned throughput, at a point in time, the Minimum capacity to which Auto Scaling can scale down to, and Maximum capacity, to which the Auto Scaling can scale up to. The default value for Target Utilization is 70% (allowed range is 20% - 80% in one percent increments), minimum capacity is 1 unit and maximum capacity is the table limit for your account in the region. Please refer to the [Limits in DynamoDB](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html#limits-capacity-units-provisioned-throughput) page for region-level default table limits.  
  
Q. Can I change the settings of an existing Auto Scaling policy?  
  
Yes, you can change the settings of an existing Auto Scaling policy at any time, by navigating to the 'Capacity' tab in the management console or programmatically from the CLI or SDK using the [AutoScaling APIs](http://docs.aws.amazon.com/ApplicationAutoScaling/latest/APIReference/Welcome.html).  
  
Q. How does Auto Scaling work?  
  
When you create a new Auto Scaling policy for your DynamoDB table, Amazon CloudWatch alarms are created with thresholds for target utilization you specify, calculated based on consumed and provisioned capacity metrics published to CloudWatch. If the table's actual utilization deviates from target for a specific length of time, the CloudWatch alarms activates Auto Scaling, which evaluates your policy and in turn makes an UpdateTable API request to DynamoDB to dynamically increase (or decrease) the table's provisioned throughput capacity to bring the actual utilization closer to the target.  
  
Q. Can I enable a single Auto Scaling policy across multiple tables in multiple regions?  
  
No, an Auto Scaling policy can only be set to a single table or a global secondary indexes within a single region.  
  
Q. Can I force an Auto Scaling policy to scale up to maximum capacity or scale down to minimum capacity instantly?  
  
No, scaling up instantly to maximum capacity or scaling down to minimum capacity is not supported. Instead, you can temporarily disable Auto Scaling, set desired capacity you need manually for required duration, and re-enable Auto Scaling later.

Q. Where can I monitor the scaling actions triggered by Auto Scaling?  
  
You can monitor status of scaling actions triggered by Auto Scaling under the 'Capacity' tab in the management console and from CloudWatch graphs under the 'Metrics' tab.  
  
Q. How can I tell if a table has an active Auto Scaling policy or not?  
  
From the DynamoDB console, click on Tables in the left menu, to bring up the list view of all DynamoDB tables in your account. For tables with an active Auto Scaling policy, the 'Auto Scaling' column shows either READ\_CAPACITY, WRITE\_CAPACITY or READ\_AND\_WRITE depending on whether Auto Scaling is enabled for read or write or both. Additionally, under the 'Table details' section of the 'Overview' tab of a table, the provisioned capacity label shows whether Auto Scaling is enabled for read, write or both.  
  
Q. What happens to the Auto Scaling policy when I delete a table or global secondary index with an active policy?  
  
When you delete a table or global secondary index from the console, its Auto Scaling policy and supporting Cloud Watch alarms are also deleted.  
  
Q. Are there any additional costs to use Auto Scaling?  
  
No, there are no additional cost to using Auto Scaling, beyond what you already pay for DynamoDB and CloudWatch alarms. To learn about DynamoDB pricing, please visit the [DynamoDB pricing page](https://aws.amazon.com/dynamodb/pricing/).  
  
Q. How does throughput capacity managed by Auto Scaling work with my Reserved Capacity?  
  
Auto Scaling works with reserved capacity in the same manner as manually provisioned throughput capacity does today. Reserved Capacity is applied to the total provisioned capacity for the region you purchased it in. Capacity provisioned by Auto Scaling will consume the reserved capacity first, billed at discounted prices, and any excess capacity will be charged at standard rates. To limit total consumption to the reserved capacity you purchased, distribute maximum capacity limit across all tables with Auto Scaling enabled, to be cumulatively less than total reserved capacity amount you have purchased.

## **Global secondary indexes**

Q: What are global secondary indexes?

Global secondary indexes are indexes that contain a partition or partition-and-sort keys that can be different from the table's primary key.

For efficient access to data in a table, Amazon DynamoDB creates and maintains indexes for the primary key attributes. This allows applications to quickly retrieve data by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available to allow efficient access to data with attributes other than the primary key. To address this, you can create one or more secondary indexes on a table, and issue Query requests against these indexes.

Amazon DynamoDB supports two types of secondary indexes:

* Local secondary index — an index that has the same partition key as the table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a table partition that has the same partition key.
* Global secondary index — an index with a partition or a partition-and-sort key that can be different from those on the table. A global secondary index is considered "global" because queries on the index can span all items in a table, across all partitions.

Secondary indexes are automatically maintained by Amazon DynamoDB as sparse objects. Items will only appear in an index if they exist in the table on which the index is defined. This makes queries against an index very efficient, because the number of items in the index will often be significantly less than the number of items in the table.

Global secondary indexes support non-unique attributes, which increases query flexibility by enabling queries against any non-key attribute in the table.

Consider a gaming application that stores the information of its players in a DynamoDB table whose primary key consists of *UserId*(partition) and *GameTitle* (sort). Items have attributes named *TopScore*, *Timestamp*, *ZipCode*, and others. Upon table creation, DynamoDB provides an implicit index (primary index) on the primary key that can support efficient queries that return a specific user’s top scores for all games.

However, if the application requires top scores of users for a particular game, using this primary index would be inefficient, and would require scanning through the entire table. Instead, a global secondary index with GameTitle as the partition key element and TopScore as the sort key element would enable the application to rapidly retrieve top scores for a game.

A GSI does not need to have a sort key element. For instance, you could have a GSI with a key that only has a partition element *GameTitle*. In the example below, the GSI has no projected attributes, so it will just return all items (identified by primary key) that have an attribute matching the *GameTitle* you are querying on.

Q: When should I use global secondary indexes?

Global secondary indexes are particularly useful for tracking relationships between attributes that have a lot of different values. For example, you could create a DynamoDB table with *CustomerID* as the primary partition key for the table and *ZipCode* as the partition key for a global secondary index, since there are a lot of zip codes and since you will probably have a lot of customers. Using the primary key, you could quickly get the record for any customer. Using the global secondary index, you could efficiently query for all customers that live in a given zip code.

To ensure that you get the most out of your global secondary index's capacity, please review our [best practices documentation on uniform workloads](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForTables.html#GuidelinesForTables.UniformWorkload%22%3Ebest%20practices%20documentation%20on%20uniform%20workloads).

Q: How do I create a global secondary index for a DynamoDB table?

GSIs associated with a table can be specified at any time. For detailed steps on creating a Table and its indexes, see [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStartedCreateTables.html). You can create a maximum of [5 global secondary indexes](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html) per table.

Q: Does the local version of DynamoDB support global secondary indexes?

Yes. The local version of DynamoDB is useful for developing and testing DynamoDB-backed applications. You can download the local version of DynamoDB [here](http://dynamodb-local.s3-website-us-west-2.amazonaws.com/dynamodb_local_latest.zip).

Q: What are projected attributes?

The data in a secondary index consists of attributes that are projected, or copied, from the table into the index. When you create a secondary index, you define the alternate key for the index, along with any other attributes that you want to be projected in the index. Amazon DynamoDB copies these attributes into the index, along with the primary key attributes from the table. You can then query the index just as you would query a table.

Q: Can a global secondary index key be defined on non-unique attributes?

Yes. Unlike the primary key on a table, a GSI index does not require the indexed attributes to be unique. For instance, a GSI on *GameTitle* could index all items that track scores of users for every game. In this example, this GSI can be queried to return all users that have played the game "TicTacToe."

Q: How do global secondary indexes differ from local secondary indexes?

Both global and local secondary indexes enhance query flexibility. An [LSI](https://aws.amazon.com/dynamodb/faqs/#lsi_anchor) is attached to a specific partition key value, whereas a GSI spans all partition key values. Since items having the same partition key value share the same partition in DynamoDB, the "Local" Secondary Index only covers items that are stored together (on the same partition). Thus, the purpose of the LSI is to query items that have the same partition key value but different sort key values. For example, consider a DynamoDB table that tracks Orders for customers, where *CustomerId* is the partition key.

An LSI on *OrderTime* allows for efficient queries to retrieve the most recently ordered items for a particular customer.

In contrast, a GSI is not restricted to items with a common partition key value. Instead, a GSI spans all items of the table just like the primary key. For the table above, a GSI on *ProductId* can be used to efficiently find all orders of a particular product. Note that in this case, no GSI sort key is specified, and even though there might be many orders with the same *ProductId*, they will be stored as separate items in the GSI.

In order to ensure that data in the table and the index are co-located on the same partition, LSIs limit the total size of all elements (tables and indexes) to 10 GB per partition key value. GSIs do not enforce data co-location, and have no such restriction.

When you write to a table, DynamoDB atomically updates all the LSIs affected. In contrast, updates to any GSIs defined on the table are eventually consistent.

LSIs allow the [Query API](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html) to retrieve attributes that are not part of the projection list. This is not supported behavior for GSIs.

Q: How do global secondary indexes work?

In many ways, GSI behavior is similar to that of a DynamoDB table. You can query a GSI using its partition key element, with conditional filters on the GSI sort key element. However, unlike a [primary key](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DataModel.html#DataModelPrimaryKey) of a DynamoDB table, which must be unique, a GSI key can be the same for multiple items. If multiple items with the same GSI key exist, they are tracked as separate GSI items, and a GSI query will retrieve all of them as individual items. Internally, DynamoDB will ensure that the contents of the GSI are updated appropriately as items are added, removed or updated.

DynamoDB stores a GSI’s projected attributes in the GSI data structure, along with the GSI key and the matching items’ primary keys. GSI’s consume storage for projected items that exist in the source table. This enables queries to be issued against the GSI rather than the table, increasing query flexibility and improving workload distribution. Attributes that are part of an item in a table, but not part of the GSI key, primary key of the table, or projected attributes are thus not returned on querying the GSI index. Applications that need additional data from the table after querying the GSI, can retrieve the primary key from the GSI and then use either the [GetItem](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_GetItem.html) or [BatchGetItem](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_BatchGetItem.html) APIs to retrieve the desired attributes from the table. As GSI’s are eventually consistent, applications that use this pattern have to accommodate item deletion (from the table) in between the calls to the GSI and GetItem/BatchItem.

DynamoDB automatically handles item additions, updates and deletes in a GSI when corresponding changes are made to the table. When an item (with GSI key attributes) is added to the table, DynamoDB updates the GSI asynchronously to add the new item. Similarly, when an item is deleted from the table, DynamoDB removes the item from the impacted GSI.

Q: Can I create global secondary indexes for partition-based tables and partition-sort schema tables?

Yes, you can create a global secondary index regardless of the type of primary key the DynamoDB table has. The table's primary key can include just a partition key, or it may include both a partition key and a sort key.

Q: What is the consistency model for global secondary indexes?

GSIs support eventual consistency. When items are inserted or updated in a table, the GSIs are not updated synchronously. Under normal operating conditions, a write to a global secondary index will propagate in a fraction of a second. In unlikely failure scenarios, longer delays may occur. Because of this, your application logic should be capable of handling GSI query results that are potentially out-of-date. Note that this is the same behavior exhibited by other DynamoDB APIs that support eventually consistent reads.

Consider a table tracking top scores where each item has attributes *UserId*, *GameTitle* and *TopScore*. The partition key is *UserId*, and the primary sort key is *GameTitle*. If the application adds an item denoting a new top score for *GameTitle* "TicTacToe" and *UserId*"GAMER123," and then subsequently queries the GSI, it is possible that the new score will not be in the result of the query. However, once the GSI propagation has completed, the new item will start appearing in such queries on the GSI.

Q: Can I provision throughput separately for the table and for each global secondary index?

Yes. GSIs manage throughput independently of the table they are based on. When you enable Auto Scaling for a new or existing table from the console, you can optionally choose to apply the same settings to GSIs. You can also provision different throughput for tables and global secondary indexes manually.

Depending upon on your application, the request workload on a GSI can vary significantly from that of the table or other GSIs. Some scenarios that show this are given below:

* A GSI that contains a small fraction of the table items needs a much lower write throughput compared to the table.
* A GSI that is used for infrequent item lookups needs a much lower read throughput, compared to the table.
* A GSI used by a read-heavy background task may need high read throughput for a few hours per day.

As your needs evolve, you can change the provisioned throughput of the GSI, independently of the provisioned throughput of the table.

Consider a DynamoDB table with a GSI that projects all attributes, and has the GSI key present in 50% of the items. In this case, the GSI’s provisioned write capacity units should be set at 50% of the table’s provisioned write capacity units. Using a similar approach, the read throughput of the GSI can be estimated. Please see DynamoDB GSI Documentation for more details.

Q: How does adding a global secondary index impact provisioned throughput and storage for a table?

Similar to a DynamoDB table, a GSI consumes provisioned throughput when reads or writes are performed to it. A write that adds or updates a GSI item will consume write capacity units based on the size of the update. The capacity consumed by the GSI write is in addition to that needed for updating the item in the table.

Note that if you add, delete, or update an item in a DynamoDB table, and if this does not result in a change to a GSI, then the GSI will not consume any write capacity units. This happens when an item without any GSI key attributes is added to the DynamoDB table, or an item is updated without changing any GSI key or projected attributes.

A query to a GSI consumes read capacity units, based on the size of the items examined by the query.

Storage costs for a GSI are based on the total number of bytes stored in that GSI. This includes the GSI key and projected attributes and values, and an overhead of 100 bytes for indexing purposes.

Q: Can DynamoDB throttle my application writes to a table because of a GSI’s provisioned throughput?

Because some or all writes to a DynamoDB table result in writes to related GSIs, it is possible that a GSI’s provisioned throughput can be exhausted. In such a scenario, subsequent writes to the table will be throttled. This can occur even if the table has available write capacity units.

Q: How often can I change provisioned throughput at the index level?

Tables with GSIs have the same [daily limits](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html) on the number of throughput change operations as normal tables.

Q: How am I charged for DynamoDB global secondary index?

You are charged for the aggregate provisioned throughput for a table and its GSIs by the hour. When you provision manually, while not required, You are charged for the aggregate provisioned throughput for a table and its GSIs by the hour. In addition, you are charged for the data storage taken up by the GSI as well as standard data transfer (external) fees. If you would like to change your GSI’s provisioned throughput capacity, you can do so using the [DynamoDB Console](https://console.aws.amazon.com/dynamodb/) or the [UpdateTable API](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_UpdateTable.html) or the [PutScalingPolicy API](http://docs.aws.amazon.com/ApplicationAutoScaling/latest/APIReference/API_PutScalingPolicy.html) for updating Auto Scaling policy settings.

Q: Can I specify which global secondary index should be used for a query?

Yes. In addition to the common query parameters, a GSI [*Query command*](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html) explicitly includes the name of the GSI to operate against. Note that a query can use only one GSI.

Q: What API calls are supported by a global secondary index?

The API calls supported by a GSI are [*Query*](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html) and [*Scan*](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Scan.html). A Query operation only searches index key attribute values and supports a subset of comparison operators. Because GSIs are updated asynchronously, you cannot use the [*ConsistentRead*](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html#DDB-Query-request-ConsistentRead) parameter with the query. Please see [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html) for details on using GSIs with queries and scans.

Q: What is the order of the results in scan on a global secondary index?

For a global secondary index, with a partition-only key schema there is no ordering. For global secondary index with partition-sort key schema the ordering of the results for the same partition key is based on the sort key attribute.

Q. Can I change Global Secondary Indexes after a table has been created?

Yes, Global Secondary Indexes can be changed at any time, even after the table has been created.

Q. How can I add a Global Secondary Index to an existing table?

You can add a Global Secondary Indexes through the console or through an API call. On the DynamoDB console, first select the table for which you want to add a Global Secondary Index and click the “Create Index” button to add a new index. Follow the steps in the index creation wizard and select “Create” when done. You can also add or delete a Global Secondary Index using the UpdateTable API call with the GlobalSecondaryIndexes parameter.You can learn more by reading our [documentation page](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.OnlineOps.html).

Q. How can I delete a Global Secondary Index?

You can delete a Global Secondary Index from the console or through an API call. On the DynamoDB console, select the table for which you want to delete a Global Secondary Index. Then, select the “Indexes” tab under “Table Items” and click on the “Delete” button next to delete the index. You can also delete a Global Secondary Index using the UpdateTable API call.You can learn more by reading our[documentation page](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.OnlineOps.html).

Q. Can I add or delete more than one index in a single API call on the same table?  
  
You can only add or delete one index per API call.

Q. What happens if I submit multiple requests to add the same index?

Only the first add request is accepted and all subsequent add requests will fail till the first add request is finished.

Q. Can I concurrently add or delete several indexes on the same table?

No, at any time there can be only one active add or delete index operation on a table.

Q. Should I provision additional throughput to add a Global Secondary Index?

With Auto Scaling, it is recommended that you apply the same settings to Global Secondary Index as the table. When you provision manually, while not required, it is highly recommended that you provision additional write throughput that is separate from the throughput for the index. If you do not provision additional write throughput, the write throughput from the index will be consumed for adding the new index. This will affect the write performance of the index while the index is being created as well as increase  the time to create the new index.

Q. Do I have to reduce the additional throughput on a Global Secondary Index once the index has been created?

Yes, you would have to dial back the additional write throughput you provisioned for adding an index, once the process is complete.

Q. Can I modify the write throughput that is provisioned for adding a Global Secondary Index?

Yes, you can dial up or dial down the provisioned write throughput for index creation at any time during the creation process.

Q. When a Global Secondary Index is being added or deleted, is the table still available?

Yes, the table is available when the Global Secondary Index is being updated.

Q. When a Global Secondary Index is being added or deleted, are the existing indexes still available?

Yes, the existing indexes are available when the Global Secondary Index is being updated.

Q. When a Global Secondary Index is being created added, is the new index available?

No, the new index becomes available only after the index creation process is finished.

Q. How long does adding a Global Secondary Index take?

The length of time depends on the size of the table and the amount of additional provisioned write throughput for Global Secondary Index creation. The process of adding or deleting an index could vary from a few minutes to a few hours. For example, let's assume that you have a 1GB table that has 500 write capacity units provisioned and you have provisioned 1000 additional write capacity units for the index and new index creation. If the new index includes all the attributes in the table and the table is using all the write capacity units, we expect the index creation will take roughly 30 minutes.

Q. How long does deleting a Global Secondary Index take?

Deleting an index will typically finish in a few minutes. For example, deleting an index with 1GB of data will typically take less than 1 minute.

Q. How do I track the progress of add or delete operation for a Global Secondary Index?

You can use the DynamoDB console or DescribeTable API to check the status of all indexes associated with the table. For an add index operation, while the index is being created, the status of the index will be “CREATING”. Once the creation of the index is finished, the index state will change from “CREATING” to “ACTIVE”. For a delete index operation, when the request is complete, the deleted index will cease to exist.

Q. Can I get a notification when the index creation process for adding a Global Secondary Index is complete?

You can request a notification to be sent to your email address confirming that the index addition has been completed. When you add an index through the console, you can request a notification on the last step before creating the index. When the index creation is complete, DynamoDB will send an SNS notification to your email.

Q. What happens when I try to add more Global Secondary Indexes, when I already have 5?

You are currently limited to 5 GSIs. The “Add” operation will fail and you will get an error.

Q. Can I reuse a name for a Global Secondary Index after an index with the same name has been deleted?

Yes, once a Global Secondary Index has been deleted, that index name can be used again when a new index is added.

Q. Can I cancel an index add while it is being created?

No, once index creation starts, the index creation process cannot be canceled.

Q: Are GSI key attributes required in all items of a DynamoDB table?

No. GSIs are sparse indexes. Unlike the requirement of having a primary key, an item in a DynamoDB table does not have to contain any of the GSI keys. If a GSI key has both partition and sort elements, and a table item omits either of them, then that item will not be indexed by the corresponding GSI. In such cases, a GSI can be very useful in efficiently locating items that have an uncommon attribute.

Q: Can I retrieve all attributes of a DynamoDB table from a global secondary index?

A query on a GSI can only return attributes that were specified to be included in the GSI at creation time. The attributes included in the GSI are those that are projected by default such as the GSI’s key attribute(s) and table’s primary key attribute(s), and those that the user specified to be projected. For this reason, a GSI query will not return attributes of items that are part of the table, but not included in the GSI. A GSI that specifies all attributes as projected attributes can be used to retrieve any table attributes. See [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html) for documentation on using GSIs for queries.

Q: How can I list GSIs associated with a table?

The [DescribeTable API](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_DescribeTable.html) will return detailed information about global secondary indexes on a table.

Q: What data types can be indexed?

All scalar data types (Number, String, Binary, and Boolean) can be used for the sort key element of the local secondary index key. Set, list, and map types cannot be indexed.

Q: Are composite attribute indexes possible?

No. But you can concatenate attributes into a string and use this as a key.

Q: What data types can be part of the projected attributes for a GSI?

You can specify attributes with any data types (including set types) to be projected into a GSI.

Q: What are some scalability considerations of GSIs?

[Performance considerations](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForTables.html#GuidelinesForTables.UniformWorkload) of the primary key of a DynamoDB table also apply to GSI keys. A GSI assumes a relatively random access pattern across all its keys. To get the most out of secondary index provisioned throughput, you should select a GSI partition key attribute that has a large number of distinct values, and a GSI sort key attribute that is requested fairly uniformly, as randomly as possible.

Q: What new metrics will be available through CloudWatch for global secondary indexes?

Tables with GSI will provide aggregate metrics for the table and GSIs, as well as breakouts of metrics for the table and each GSI.

Reports for individual GSIs will support a subset of the CloudWatch metrics that are supported by a table. These include:

* Read Capacity (Provisioned Read Capacity, Consumed Read Capacity)
* Write Capacity (Provisioned Write Capacity, Consumed Write Capacity)
* Throttled read events
* Throttled write events

For more details on metrics supported by DynamoDB tables and indexes see [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/MonitoringDynamoDB.html).

Q: How can I scan a Global Secondary Index?

Global secondary indexes can be scanned via the Console or the Scan API.

To scan a global secondary index, explicitly reference the index in addition to the name of the table you’d like to scan. You must specify the index partition attribute name and value. You can optionally specify a condition against the index key sort attribute.

Q: Will a Scan on Global secondary index allow me to specify non-projected attributes to be returned in the result set?

Scan on global secondary indexes will not support fetching of non-projected attributes.

Q: Will there be parallel scan support for indexes?

Yes, parallel scan will be supported for indexes and the semantics are the same as that for the main table.

## **Local secondary indexes**

Q: What are local secondary indexes?

Local secondary indexes enable some common queries to run more quickly and cost-efficiently, that would otherwise require retrieving a large number of items and then filtering the results. It means your applications can rely on more flexible queries based on a wider range of attributes.

Before the launch of local secondary indexes, if you wanted to find specific items within a partition (items that share the same partition key), DynamoDB would have fetched all objects that share a single partition key, and filter the results accordingly. For instance, consider an e-commerce application that stores customer order data in a DynamoDB table with partition-sort schema of customer id-order timestamp. Without LSI, to find an answer to the question “Display all orders made by Customer X with shipping date in the past 30 days, sorted by shipping date”, you had to use the Query API to retrieve all the objects under the partition key “X”, sort the results by shipment date and then filter out older records.

With local secondary indexes, we are simplifying this experience. Now, you can create an index on “shipping date” attribute and execute this query efficiently and just retieve only the necessary items. This significantly reduces the latency and cost of your queries as you will retrieve only items that meet your specific criteria. Moreover, it also simplifies the programming model for your application as you no longer have to write customer logic to filter the results. We call this new secondary index a ‘local’ secondary index because it is used along with the partition key and hence allows you to search locally within a partition key bucket. So while previously you could only search using the partition key and the sort key, now you can also search using a secondary index in place of the sort key, thus expanding the number of attributes that can be used for queries which can be conducted efficiently.

Redundant copies of data attributes are copied into the local secondary indexes you define. These attributes include the table partition and sort key, plus the alternate sort key you define. You can also redundantly store other data attributes in the local secondary index, in order to access those other attributes without having to access the table itself.

Local secondary indexes are not appropriate for every application. They introduce some constraints on the volume of data you can store within a single partition key value. For more information, see the FAQ items below about item collections.

Q: What are Projections?

The set of attributes that is copied into a local secondary index is called a projection. The projection determines the attributes that you will be able to retrieve with the most efficiency. When you query a local secondary index, Amazon DynamoDB can access any of the projected attributes, with the same performance characteristics as if those attributes were in a table of their own. If you need to retrieve any attributes that are not projected, Amazon DynamoDB will automatically fetch those attributes from the table.

When you define a local secondary index, you need to specify the attributes that will be projected into the index. At a minimum, each index entry consists of: (1) the table partition key value, (2) an attribute to serve as the index sort key, and (3) the table sort key value.

Beyond the minimum, you can also choose a user-specified list of other non-key attributes to project into the index. You can even choose to project all attributes into the index, in which case the index replicates the same data as the table itself, but the data is organized by the alternate sort key you specify.

Q: How can I create a LSI?

You need to create a LSI at the time of table creation. It can’t currently be added later on. To create an LSI, specify the following two parameters:

Indexed Sort key – the attribute that will be indexed and queried on.

Projected Attributes – the list of attributes from the table that will be copied directly into the local secondary index, so they can be returned more quickly without fetching data from the primary index, which contains all the items of the table. Without projected attributes, local secondary index contains only primary and secondary index keys.

Q: What is the consistency model for LSI?

Local secondary indexes are updated automatically when the primary index is updated. Similar to reads from a primary index, LSI supports both strong and eventually consistent read options.

Q: Do local secondary indexes contain references to all items in the table?

No, not necessarily. Local secondary indexes only reference those items that contain the indexed sort key specified for that LSI. DynamoDB’s flexible schema means that not all items will necessarily contain all attributes.

This means local secondary index can be sparsely populated, compared with the primary index. Because local secondary indexes are sparse, they are efficient to support queries on attributes that are uncommon.

For example, in the Orders example described above, a customer may have some additional attributes in an item that are included only if the order is canceled (such as CanceledDateTime, CanceledReason). For queries related to canceled items, an local secondary index on either of these attributes would be efficient since the only items referenced in the index would be those that had these attributes present.

Q: How do I query local secondary indexes?

Local secondary indexes can only be queried via the Query API.

To query a local secondary index, explicitly reference the index in addition to the name of the table you’d like to query. You must specify the index partition attribute name and value. You can optionally specify a condition against the index key sort attribute.

Your query can retrieve non-projected attributes stored in the primary index by performing a table fetch operation, with a cost of additional read capacity units.

Both strongly consistent and eventually consistent reads are supported for query using local secondary index.

Q: How do I create local secondary indexes?

Local secondary indexes must be defined at time of table creation. The primary index of the table must use a partition-sort composite key.

Q: Can I add local secondary indexes to an existing table?

No, it’s not possible to add local secondary indexes to existing tables at this time. We are working on adding this capability and will be releasing it in the future. When you create a table with local secondary index, you may decide to create local secondary index for future use by defining a sort key element that is currently not used. Since local secondary index are sparse, this index costs nothing until you decide to use it.

Q: How many local secondary indexes can I create on one table?

Each table can have up to five local secondary indexes.

Q: How many projected non-key attributes can I create on one table?

Each table can have up to 20 projected non-key attributes, in total across all local secondary indexes within the table. Each index may also specifify that all non-key attributes from the primary index are projected.

Q: Can I modify the index once it is created?

No, an index cannot be modified once it is created. We are working to add this capability in the future.

Q: Can I delete local secondary indexes?

No, local secondary indexes cannot be removed from a table once they are created at this time. Of course, they are deleted if you also decide to delete the entire table. We are working on adding this capability and will be releasing it in the future.

Q: How do local secondary indexes consume provisioned capacity?

You don’t need to explicitly provision capacity for a local secondary index. It consumes provisioned capacity as part of the table with which it is associated.

Reads from LSIs and writes to tables with LSIs consume capacity by the standard formula of 1 unit per 1KB of data, with the following differences:

When writes contain data that are relevant to one or more local secondary indexes, those writes are mirrored to the appropriate local secondary indexes. In these cases, write capacity will be consumed for the table itself, and additional write capacity will be consumed for each relevant LSI.

Updates that overwrite an existing item can result in two operations– delete and insert – and thereby consume extra units of write capacity per 1KB of data.

When a read query requests attributes that are not projected into the LSI, DynamoDB will fetch those attributes from the primary index. This implicit GetItem request consumes one read capacity unit per 4KB of item data fetched.

Q: How much storage will local secondary indexes consume?

Local secondary indexes consume storage for the attribute name and value of each LSI’s primary and index keys, for all projected non-key attributes, plus 100 bytes per item reflected in the LSI.

Q: What data types can be indexed?

All scalar data types (Number, String, Binary) can be used for the sort key element of the local secondary index key. Set types cannot be used.

Q: What data types can be projected into a local secondary index?

All data types (including set types) can be projected into a local secondary index.

Q: What are item collections and how are they related to LSI?

In Amazon DynamoDB, an item collection is any group of items that have the same partition key, across a table and all of its local secondary indexes. Traditional partitioned (or sharded) relational database systems call these shards or partitions, referring to all database items or rows stored under a partition key.

Item collections are automatically created and maintained for every table that includes local secondary indexes. DynamoDB stores each item collection within a single disk partition.

Q: Are there limits on the size of an item collection?

Every item collection in Amazon DynamoDB is subject to a maximum size limit of 10 gigabytes. For any distinct partition key value, the sum of the item sizes in the table plus the sum of the item sizes across all of that table's local secondary indexes must not exceed 10 GB.

The 10 GB limit for item collections does not apply to tables without local secondary indexes; only tables that have one or more local secondary indexes are affected.

Although individual item collections are limited in size, the storage size of an overall table with local secondary indexes is not limited. The total size of an indexed table in Amazon DynamoDB is effectively unlimited, provided the total storage size (table and indexes) for any one partition key value does not exceed the 10 GB threshold.

Q: How can I track the size of an item collection?

DynamoDB’s write APIs (PutItem, UpdateItem, DeleteItem, and BatchWriteItem) include an option, which allows the API response to include an estimate of the relevant item collection’s size. This estimate includes lower and upper size estimate for the data in a particular item collection, measured in gigabytes.

We recommend that you instrument your application to monitor the sizes of your item collections. Your applications should examine the API responses regarding item collection size, and log an error message whenever an item collection exceeds a user-defined limit (8 GB, for example). This would provide an early warning system, letting you know that an item collection is growing larger, but giving you enough time to do something about it.

Q: What if I exceed the 10GB limit for an item collection?

If a particular item collection exceeds the 10GB limit, then you will not be able to write new items, or increase the size of existing items, for that particular partition key. Read and write operations that shrink the size of the item collection are still allowed. Other item collections in the table are not affected.

To address this problem , you can remove items or reduce item sizes in the collection that has exceeded 10GB. Alternatively, you can introduce new items under a new partition key value to work around this problem. If your table includes historical data that is infrequently accessed, consider archiving the historical data to Amazon S3, Amazon Glacier or another data store.

Q: How can I scan a local secondary index?

To scan a local secondary index, explicitly reference the index in addition to the name of the table you’d like to scan. You must specify the index partition attribute name and value. You can optionally specify a condition against the index key sort attribute.

Your scan can retrieve non-projected attributes stored in the primary index by performing a table fetch operation, with a cost of additional read capacity units.

Q: Will a Scan on a local secondary index allow me to specify non-projected attributes to be returned in the result set?

Scan on local secondary indexes will support fetching of non-projected attributes.

Q: What is the order of the results in scan on a local secondary index?

For local secondary index, the ordering within a collection will be the based on the order of the indexed attribute.

## **Security and control**

Q: What is DynamoDB Fine-Grained Access Control?

Fine Grained Access Control (FGAC) gives a DynamoDB table owner a high degree of control over data in the table. Specifically, the table owner can indicate *who* (caller) can access *which* items or attributes of the table and perform *what* actions (read / write capability). FGAC is used in concert with [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/), which manages the security credentials and the associated permissions.

Q: What are the common use cases for DynamoDB FGAC?

FGAC can benefit any application that tracks information in a DynamoDB table, where the end user (or application client acting on behalf of an end user) wants to read or modify the table directly, without a middle-tier service. For instance, a developer of a mobile app named *Acme* can use FGAC to track the top score of every *Acme* user in a DynamoDB table. FGAC allows the application client to modify only the top score for the user that is currently running the application.

Q: Can I use Fine Grain Access Control with JSON documents?

Yes. You can use Fine Grain Access Control (FGAC) to restrict access to your data based on top-level attributes in your document. You cannot use FGAC to restrict access based on nested attributes. For example, suppose you stored a JSON document that contained the following information about a person: ID, first name, last name, and a list of all of their friends. You could use FGAC to restrict access based on their ID, first name, or last name, but not based on the list of friends.

Q: Without FGAC, how can a developer achieve item level access control?

To achieve this level of control without FGAC, a developer would have to choose from a few potentially onerous approaches. Some of these are:

1. Proxy: The application client sends a request to a brokering proxy that performs the authentication and authorization. Such a solution increases the complexity of the system architecture and can result in a higher total cost of ownership (TCO).
2. Per Client Table: Every application client is assigned its own table. Since application clients access different tables, they would be protected from one another. This could potentially require a developer to create millions of tables, thereby making database management extremely painful.
3. Per-Client Embedded Token: A secret token is embedded in the application client. The shortcoming of this is the difficulty in changing the token and handling its impact on the stored data. Here, the key of the items accessible by this client would contain the secret token.

Q: How does DynamoDB FGAC work?

With FGAC, an application requests a security token that authorizes the application to access only specific items in a specific DynamoDB table. With this token, the end user application agent can make requests to DynamoDB directly. Upon receiving the request, the incoming request’s credentials are first evaluated by DynamoDB, which will use IAM to authenticate the request and determine the capabilities allowed for the user. If the user’s request is not permitted, FGAC will prevent the data from being accessed.

Q: How much does DynamoDB FGAC cost?

There is no additional charge for using FGAC. As always, you only pay for the provisioned throughput and storage associated with the DynamoDB table.

Q: How do I get started?

Refer to the [Fine-Grained Access Control](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/FGAC_DDB.html) section of the DynamoDB Developer Guide to learn how to create an access policy, create an IAM role for your app (e.g. a role named AcmeFacebookUsers for a Facebook app\_id of 34567), and assign your access policy to the role. The trust policy of the role determines which identity providers are accepted (e.g. Login with Amazon, Facebook, or Google), and the access policy describes which AWS resources can be accessed (e.g. a DynamoDB table). Using the role, your app can now to obtain temporary credentials for DynamoDB by calling the AssumeRoleWithIdentityRequest API of the AWS Security Token Service (STS).

Q: How do I allow users to Query a Local Secondary Index, but prevent them from causing a table fetch to retrieve non-projected attributes?

Some Query operations on a Local Secondary Index can be more expensive than others if they request attributes that are not projected into an index. You an restrict such potentially expensive “fetch” operations by limiting the permissions to only projected attributes, using the "dynamodb:Attributes" context key.

Q: How do I prevent users from accessing specific attributes?

The recommended approach to preventing access to specific attributes is to follow the principle of least privilege, and Allow access to only specific attributes.

Alternatively, you can use a *Deny* policy to specify attributes that are disallowed. However, this is not recommended for the following reasons:

1. With a *Deny*policy, it is possible for the user to discover the hidden attribute names by issuing repeated requests for every possible attribute name, until the user is ultimately denied access.
2. *Deny* policies are more fragile, since DynamoDB could introduce new API functionality in the future that might allow an access pattern that you had previously intended to block.

Q: How do I prevent users from adding invalid data to a table?

The available FGAC controls can determine which items changed or read, and which attributes can be changed or read. Users can add new items without those blocked attributes, and change any value of any attribute that is modifiable.

Q: Can I grant access to multiple attributes without listing all of them?

Yes, the IAM policy language supports a rich set of comparison operations, including StringLike, StringNotLike, and many others.  For additional details, please see the [IAM Policy Reference](http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html).

Q: How do I create an appropriate policy?

We recommend that you use the DynamoDB Policy Generator from the DynamoDB console. You may also compare your policy to those listed in the Amazon DynamoDB Developer Guide to make sure you are following a recommended pattern. You can post policies to the AWS Forums to get thoughts from the DynamoDB community.

Q: Can I grant access based on a canonical user id instead of separate ids for the user based on the identity provider they logged in with?

Not without running a “token vending machine”. If a user retrieves federated access to your IAM role directly using Facebook credentials with STS, those temporary credentials only have information about that user’s Facebook login, and not their Amazon login, or Google login. If you want to internally store a mapping of each of these logins to your own stable identifier, you can run a service that the user contacts to log in, and then call STS and provide them with credentials scoped to whatever partition key value you come up with as their canonical user id.

Q: What information cannot be hidden from callers using FGAC?

Certain information cannot currently be blocked from the caller about the items in the table:

* Item collection metrics. The caller can ask for the estimated number of items and size in bytes of the item collection.
* Consumed throughput The caller can ask for the detailed breakdown or summary of the provisioned throughput consumed by operations.
* Validation cases. In certain cases, the caller can learn about the existence and primary key schema of a table when you did not intend to give them access. To prevent this, follow the principle of least privilege and only allow access to the tables and actions that you intended to allow access to.
* If you deny access to specific attributes instead of whitelisting access to specific attributes, the caller can theoretically determine the names of the hidden attributes if “allow all except for” logic. It is safer to whitelist specific attribute names instead.

Q: Does Amazon DynamoDB support IAM permissions?

Yes, DynamoDB supports API-level permissions through AWS Identity and Access Management (IAM) service integration.

For more information about IAM, go to:

* [AWS Identity and Access Management](https://aws.amazon.com/iam/)
* [AWS Identity and Access Management Getting Started Guide](http://docs.amazonwebservices.com/IAM/latest/GettingStartedGuide/)
* [Using AWS Identity and Access Management](http://docs.amazonwebservices.com/IAM/latest/UserGuide/)

Q: I wish to perform security analysis or operational troubleshooting on my DynamoDB tables. Can I get a history of all DynamoDB API calls made on my account?

Yes. AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. The AWS API call history produced by AWS CloudTrail enables security analysis, resource change tracking, and compliance auditing. Details about DynamoDB support for CloudTrail can be found [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/logging-using-cloudtrail.html). Learn more about CloudTrail at the [AWS CloudTrail detail page](http://aws.amazon.com/cloudtrail/), and turn it on via CloudTrail's [AWS Management Console](https://console.aws.amazon.com/cloudtrail/home) home page.

## **Pricing**

Q: How will I be charged for my use of Amazon DynamoDB?

Each DynamoDB table has provisioned read-throughput and write-throughput associated with it. You are billed by the hour for that throughput capacity if you exceed the free tier.

Please note that you are charged by the hour for the throughput capacity, whether or not you are sending requests to your table. If you would like to change your table’s provisioned throughput capacity, you can do so using the AWS Management Console, the UpdateTable API or the PutScalingPolicy API for Auto Scaling..

In addition, DynamoDB also charges for indexed data storage as well as the standard internet data transfer fees

To learn more about DynamoDB pricing, please visit the [DynamoDB pricing page](https://aws.amazon.com/dynamodb/pricing/).

Q: What are some pricing examples?

Here is an example of how to calculate your throughput costs using US East (Northern Virginia) Region pricing. To view prices for other regions, [visit our pricing page](https://aws.amazon.com/dynamodb/pricing/).

If you create a table and request 10 units of write capacity and 200 units of read capacity of provisioned throughput, you would be charged:

$0.01 + (4 x $0.01) = $0.05 per hour

If your throughput needs changed and you increased your reserved throughput requirement to 10,000 units of write capacity and 50,000 units of read capacity, your bill would then change to:

(1,000 x $0.01) + (1,000 x $0.01) = $20/hour

To learn more about DynamoDB pricing, please visit the [DynamoDB pricing page](https://aws.amazon.com/dynamodb/pricing/).

Q: Do your prices include taxes?

For details on taxes, see [Amazon Web Services Tax Help](https://aws.amazon.com/tax-help/).

Q: What is provisioned throughput?

Amazon DynamoDB Auto Scaling adjusts throughput capacity automatically as request volumes change, based on your desired target utilization and minimum and maximum capacity limits, or lets you specify the request throughput you want your table to be able to achieve manually. Behind the scenes, the service handles the provisioning of resources to achieve the requested throughput rate. Rather than asking you to think about instances, hardware, memory, and other factors that could affect your throughput rate, we simply ask you to provision the throughput level you want to achieve. This is the provisioned throughput model of service.

During creation of a new table or global secondary index, Auto Scaling is enabled by default with default settings for target utilization, minimum and maximum capacity; or you can specify your required read and write capacity needs manually; and Amazon DynamoDB automatically partitions and reserves the appropriate amount of resources to meet your throughput requirements.

Q: How does selection of primary key influence the scalability I can achieve?

When storing data, Amazon DynamoDB divides a table into multiple partitions and distributes the data based on the partition key element of the primary key. While allocating capacity resources, Amazon DynamoDB assumes a relatively random access pattern across all primary keys. You should set up your data model so that your requests result in a fairly even distribution of traffic across primary keys. If a table has a very small number of heavily-accessed partition key elements, possibly even a single very heavily-used partition key element, traffic is concentrated on a small number of partitions – potentially only one partition. If the workload is heavily unbalanced, meaning disproportionately focused on one or a few partitions, the operations will not achieve the overall provisioned throughput level. To get the most out of Amazon DynamoDB throughput, build tables where the partition key element has a large number of distinct values, and values are requested fairly uniformly, as randomly as possible. An example of a good primary key is CustomerID if the application has many customers and requests made to various customer records tend to be more or less uniform. An example of a heavily skewed primary key is “Product Category Name” where certain product categories are more popular than the rest.

Q: What is a read/write capacity unit?

How do I estimate how many read and write capacity units I need for my application? A unit of Write Capacity enables you to perform one write per second for items of up to 1KB in size. Similarly, a unit of Read Capacity enables you to perform one strongly consistent read per second (or two eventually consistent reads per second) of items of up to 4KB in size. Larger items will require more capacity. You can calculate the number of units of read and write capacity you need by estimating the number of reads or writes you need to do per second and multiplying by the size of your items (rounded up to the nearest KB).

Units of Capacity required for writes = Number of item writes per second x item size in 1KB blocks

Units of Capacity required for reads\* = Number of item reads per second x item size in 4KB blocks

\* If you use eventually consistent reads you’ll get twice the throughput in terms of reads per second.

If your items are less than 1KB in size, then each unit of Read Capacity will give you 1 strongly consistent read/second and each unit of Write Capacity will give you 1 write/second of capacity. For example, if your items are 512 bytes and you need to read 100 items per second from your table, then you need to provision 100 units of Read Capacity.

If your items are larger than 4KB in size, then you should calculate the number of units of Read Capacity and Write Capacity that you need. For example, if your items are 4.5KB and you want to do 100 strongly consistent reads/second, then you would need to provision 100 (read per second) x 2 (number of 4KB blocks required to store 4.5KB) = 200 units of Read Capacity.

Note that the required number of units of Read Capacity is determined by the number of items being read per second, not the number of API calls. For example, if you need to read 500 items per second from your table, and if your items are 4KB or less, then you need 500 units of Read Capacity. It doesn’t matter if you do 500 individual GetItem calls or 50 BatchGetItem calls that each return 10 items.

Q: Will I always be able to achieve my level of provisioned throughput?

Amazon DynamoDB assumes a relatively random access pattern across all primary keys. You should set up your data model so that your requests result in a fairly even distribution of traffic across primary keys. If you have a highly uneven or skewed access pattern, you may not be able to achieve your level of provisioned throughput.

When storing data, Amazon DynamoDB divides a table into multiple partitions and distributes the data based on the partition key element of the primary key. The provisioned throughput associated with a table is also divided among the partitions; each partition's throughput is managed independently based on the quota allotted to it. There is no sharing of provisioned throughput across partitions. Consequently, a table in Amazon DynamoDB is best able to meet the provisioned throughput levels if the workload is spread fairly uniformly across the partition key values. Distributing requests across partition key values distributes the requests across partitions, which helps achieve your full provisioned throughput level.

If you have an uneven workload pattern across primary keys and are unable to achieve your provisioned throughput level, you may be able to meet your throughput needs by increasing your provisioned throughput level further, which will give more throughput to each partition. However, it is recommended that you considering modifying your request pattern or your data model in order to achieve a relatively random access pattern across primary keys.

Q: If I retrieve only a single element of a JSON document, will I be charged for reading the whole item?

Yes. When reading data out of DynamoDB, you consume the throughput required to read the entire item.

Q: What is the maximum throughput I can provision for a single DynamoDB table?

DynamoDB is designed to scale without limits However, if you wish to exceed throughput rates of 10,000 write capacity units or 10,000 read capacity units for an individual table, you must first [contact Amazon through this online form](http://portal.aws.amazon.com/gp/aws/html-forms-controller/DynamoDB_Limit_Increase_Form). If you wish to provision more than 20,000 write capacity units or 20,000 read capacity units from a single subscriber account you must first [contact us](http://portal.aws.amazon.com/gp/aws/html-forms-controller/DynamoDB_Limit_Increase_Form) using the form described above.

Q: What is the minimum throughput I can provision for a single DynamoDB table?

The smallest provisioned throughput you can request is 1 write capacity unit and 1 read capacity unit for both Auto Scaling and manual throughput provisioning..

This falls within the free tier which allows for 25 units of write capacity and 25 units of read capacity. The free tier applies at the account level, not the table level. In other words, if you add up the provisioned capacity of all your tables, and if the total capacity is no more than 25 units of write capacity and 25 units of read capacity, your provisioned capacity would fall into the free tier.

Q: Is there any limit on how much I can change my provisioned throughput with a single request?

You can increase the provisioned throughput capacity of your table by any amount using the UpdateTable API. For example, you could increase your table’s provisioned write capacity from 1 write capacity unit to 10,000 write capacity units with a single API call. Your account is still subject to table-level and account-level limits on capacity, as described in our [documentation page](http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/WorkingWithDDItems.html). If you need to raise your provisioned capacity limits, you can visit our [Support Center](https://aws.amazon.com/support), click “Open a new case”, and file a service limit increase request.

Q: How am I charged for provisioned throughput?

Every Amazon DynamoDB table has pre-provisioned the resources it needs to achieve the throughput rate you asked for. You are billed at an hourly rate for as long as your table holds on to those resources. For a complete list of prices with examples, see the [DynamoDB pricing page](https://aws.amazon.com/dynamodb/pricing/).

Q: How do I change the provisioned throughput for an existing DynamoDB table?

There are two ways to update the provisioned throughput of an Amazon DynamoDB table. You can either make the change in the management console, or you can use the UpdateTable API call. In either case, Amazon DynamoDB will remain available while your provisioned throughput level increases or decreases.

Q: How often can I change my provisioned throughput?

You can increase your provisioned throughput as often as you want. You can decrease up to four times any time per day. A day is defined according to the GMT time zone. Additionally, if there was no decrease in the past four hours, an additional dial down is allowed, effectively bringing maximum number of decreases in a day to 9 (4 decreases in the first 4 hours, and 1 decrease for each of the subsequent 4 hour windows in a day).

Keep in mind that you can’t change your provisioned throughput if your Amazon DynamoDB table is still in the process of responding to your last request to change provisioned throughput. Use the management console or the DescribeTables API to check the status of your table. If the status is “CREATING”, “DELETING”, or “UPDATING”, you won’t be able to adjust the throughput of your table. Please wait until you have a table in “ACTIVE” status and try again.

Q: Does the consistency level affect the throughput rate?

Yes. For a given allocation of resources, the read-rate that a DynamoDB table can achieve is different for strongly consistent and eventually consistent reads. If you request “1,000 read capacity units”, DynamoDB will allocate sufficient resources to achieve 1,000 strongly consistent reads per second of items up to 4KB. If you want to achieve 1,000 eventually consistent reads of items up to 4KB, you will need half of that capacity, i.e., 500 read capacity units. For additional guidance on choosing the appropriate throughput rate for your table, see our provisioned throughput guide.

Q: Does the item size affect the throughput rate?

Yes. For a given allocation of resources, the read-rate that a DynamoDB table can achieve does depend on the size of an item. When you specify the provisioned read throughput you would like to achieve, DynamoDB provisions its resources on the assumption that items will be less than 4KB in size. Every increase of up to 4KB will linearly increase the resources you need to achieve the same throughput rate. For example, if you have provisioned a DynamoDB table with 100 units of read capacity, that means that it can handle 100 4KB reads per second, or 50 8KB reads per second, or 25 16KB reads per second, and so on.

Similarly the write-rate that a DynamoDB table can achieve does depend on the size of an item. When you specify the provisioned write throughput you would like to achieve, DynamoDB provisions its resources on the assumption that items will be less than 1KB in size. Every increase of up to 1KB will linearly increase the resources you need to achieve the same throughput rate. For example, if you have provisioned a DynamoDB table with 100 units of write capacity, that means that it can handle 100 1KB writes per second, or 50 2KB writes per second, or 25 4KB writes per second, and so on.

For additional guidance on choosing the appropriate throughput rate for your table, see our provisioned throughput guide.

Q: What happens if my application performs more reads or writes than my provisioned capacity?

If your application performs more reads/second or writes/second than your table’s provisioned throughput capacity allows, requests above your provisioned capacity will be throttled and you will receive 400 error codes. For instance, if you had asked for 1,000 write capacity units and try to do 1,500 writes/second of 1 KB items, DynamoDB will only allow 1,000 writes/second to go through and you will receive error code 400 on your extra requests. You should use CloudWatch to monitor your request rate to ensure that you always have enough provisioned throughput to achieve the request rate that you need.

Q: How do I know if I am exceeding my provisioned throughput capacity?

DynamoDB publishes your consumed throughput capacity as a CloudWatch metric. You can set an alarm on this metric so that you will be notified if you get close to your provisioned capacity.

Q: How long does it take to change the provisioned throughput level of a table?

In general, decreases in throughput will take anywhere from a few seconds to a few minutes, while increases in throughput will typically take anywhere from a few minutes to a few hours.

We strongly recommend that you do not try and schedule increases in throughput to occur at almost the same time when that extra throughput is needed. We recommend provisioning throughput capacity sufficiently far in advance to ensure that it is there when you need it.

## **Reserved capacity**

Q: What is Reserved Capacity?

Reserved Capacity is a billing feature that allows you to obtain discounts on your provisioned throughput capacity in exchange for:

* A one-time up-front payment
* A commitment to a minimum monthly usage level for the duration of the term of the agreement.

Reserved Capacity applies within a single AWS Region and can be purchased with 1-year or 3-year terms. Every DynamoDB table has provisioned throughput capacity associated with it, whether managed by Auto Scaling or provisioned manually when you create or update a table. This capacity is what determines the read and write throughput rate that your DynamoDB table can achieve. Reserved Capacity is a billing arrangement and has no direct impact on the performance or capacity of your DynamoDB tables. For example, if you buy 100 write capacity units of Reserved Capacity, you have agreed to pay for that much capacity for the duration of the agreement (1 or 3 years) in exchange for discounted pricing.

Q: How do I buy Reserved Capacity?

Log into the [AWS Management Console](https://aws.amazon.com/console/), go to the DynamoDB console page, and then click on "Reserved Capacity”. This will take you to the "Reserved Capacity Usage" page. Click on "Purchase Reserved Capacity" and this will bring up a form you can fill out to purchase Reserved Capacity. Make sure you have selected the AWS Region in which your Reserved Capacity will be used. After you have finished purchasing Reserved Capacity, you will see purchase you made on the "Reserved Capacity Usage" page.

Q: Can I cancel a Reserved Capacity purchase?

No, you cannot cancel your Reserved Capacity and the one-time payment is not refundable. You will continue to pay for every hour during your Reserved Capacity term regardless of your usage.

Q: What is the smallest amount of Reserved Capacity that I can buy?

The smallest Reserved Capacity offering is 100 capacity units (reads or writes).

Q: Are there APIs that I can use to buy Reserved Capacity?

Not yet. We will provide APIs and add more Reserved Capacity options over time.

Q: Can I move Reserved Capacity from one Region to another?

No. Reserved Capacity is associated with a single Region.

Q: Can I provision more throughput capacity than my Reserved Capacity?

Yes. When you purchase Reserved Capacity, you are agreeing to a minimum usage level and you pay a discounted rate for that usage level. If you provision more capacity than that minimum level, you will be charged at standard rates for the additional capacity.

Q: How do I use my Reserved Capacity?

Reserved Capacity is automatically applied to your bill. For example, if you purchased 100 write capacity units of Reserved Capacity and you have provisioned 300, then your Reserved Capacity purchase will automatically cover the cost of 100 write capacity units and you will pay standard rates for the remaining 200 write capacity units.

Q: What happens if I provision less throughput capacity than my Reserved Capacity?

A Reserved Capacity purchase is an agreement to pay for a minimum amount of provisioned throughput capacity, for the duration of the term of the agreement, in exchange for discounted pricing. If you use less than your Reserved Capacity, you will still be charged each month for that minimum amount of provisioned throughput capacity.

Q: Can I use my Reserved Capacity for multiple DynamoDB tables?

Yes. Reserved Capacity is applied to the total provisioned capacity within the Region in which you purchased your Reserved Capacity. For example, if you purchased 5,000 write capacity units of Reserved Capacity, then you can apply that to one table with 5,000 write capacity units, or 100 tables with 50 write capacity units, or 1,000 tables with 5 write capacity units, etc.

Q: Does Reserved Capacity apply to DynamoDB usage in Consolidated Billing accounts?

Yes. If you have multiple accounts linked with Consolidated Billing, Reserved Capacity units purchased either at the Payer Account level or Linked Account level are shared with all accounts connected to the Payer Account. Reserved capacity will first be applied to the account which purchased it and then any unused capacity will be applied to other linked accounts.

## **Cross-region replication**

Q: What is a DynamoDB cross-region replication?

DynamoDB cross-region replication allows you to maintain identical copies (called replicas) of a DynamoDB table (called master table) in one or more AWS regions. After you enable cross-region replication for a table, identical copies of the table are created in other AWS regions. Writes to the table will be automatically propagated to all replicas.

Q: When should I use cross-region replication?

You can use cross-region replication for the following scenarios.

* Efficient disaster recovery: By replicating tables in multiple data centers, you can switch over to using DynamoDB tables from another region in case a data center failure occurs.
* Faster reads: If you have customers in multiple regions, you can deliver data faster by reading a DynamoDB table from the closest AWS data center.
* Easier traffic management: You can use replicas to distribute the read workload across tables and thereby consume less read capacity in the master table.
* Easy regional migration: By creating a read replica in a new region and then promoting the replica to be a master, you migrate your application to that region more easily.
* Live data migration: To move a DynamoDB table from one region to another, you can create a replica of the table from the source region in the destination region. When the tables are in sync, you can switch your application to write to the destination region.

Q: What cross-region replication modes are supported?

Cross-region replication currently supports single master mode. A single master has one master table and one or more replica tables.

Q. How can I set up single master cross-region replication for a table?

You can create cross-region replicas using the [DynamoDB Cross-region Replication library](http://github.com/awslabs/dynamodb-cross-region-library).

Q: How do I know when the bootstrapping is complete?

On the replication management application, the state of the replication changes from Bootstrapping to Active.

Q: Can I have multiple replicas for a single master table?

Yes, there are no limits on the number of replicas tables from a single master table. A DynamoDB Streams reader is created for each replica table and copies data from the master table, keeping the replicas in sync.

Q: How much does it cost to set up cross-region replication for a table?

[DynamoDB cross-region replication](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.CrossRegionRepl.html) is enabled using the [DynamoDB Cross-region Replication Library](https://github.com/awslabs/dynamodb-cross-region-library). While there is no additional charge for the cross-region replication library, you pay the usual prices for the following resources used by the process. You will be billed for:

* Provisioned throughput (Writes and Reads) and storage for the replica tables.
* Data Transfer across regions.
* Reading data from DynamoDB Streams to keep the tables in sync.
* The EC2 instances provisioned to host the replication process. The cost of the instances will depend on the instance type you choose and the region hosting the instances.

Q: In which region does the Amazon EC2 instance hosting the cross-region replication run?

The cross-region replication application is hosted in an Amazon EC2 instance in the same region where the cross-region replication application was originally launched. You will be charged the instance price in this region.

Q: Does the Amazon EC2 instance Auto Scale as the size and throughput of the master and replica tables change?

Currently, we will not auto scale the EC2 instance. You will need to pick the instance size when configuring DynamoDB Cross-region Replication.

Q: What happens if the Amazon EC2 instance managing the replication fails?

The Amazon EC2 instance runs behind an auto scaling group, which means the application will automatically fail over to another instance. The application underneath uses the Kinesis Client Library (KCL), which checkpoints the copy. In case of an instance failure, the application knows to find the checkpoint and resume from there.

Q: Can I keep using my DynamoDB table while a Read Replica is being created?

Yes, creating a replica is an online operation. Your table will remain available for reads and writes while the read replica is being created. The bootstrapping uses the Scan operation to copy from the source table. We recommend that the table is provisioned with sufficient read capacity units to support the Scan operation.

Q: How long does it take to create a replica?

The time to initially copy the master table to the replica table depends on the size of the master table, the provisioned capacity of the master table and replica table. The time to propagate an item-level change on the master table to the replica table depends on the provisioned capacity on the master and replica tables, and the size of the Amazon EC2 instance running the replication application.

Q: If I change provisioned capacity on my master table, does the provisioned capacity on my replica table also update?

After the replication has been created, any changes to the provisioned capacity on the master table will not result in an update in throughput capacity on the replica table.

Q: Will my replica tables have the same indexes as the master table?

If you choose to create the replica table from the replication application, the secondary indexes on the master table will NOT be automatically created on the replica table. The replication application will not propagate changes made on secondary indices on the master table to replica tables. You will have to add/update/delete indexes on each of the replica tables through the AWS Management Console as you would with regular DynamoDB tables.

Q: Will my replica have the same provisioned throughput capacity as the master table?

When creating the replica table, we recommend that you provision at least the same write capacity as the master table to ensure that it has enough capacity to handle all incoming writes. You can set the provisioned read capacity of your replica table at whatever level is appropriate for your application.

Q: What is the consistency model for replicated tables?

Replicas are updated asynchronously. DynamoDB will acknowledge a write operation as successful once it has been accepted by the master table. The write will then be propagated to each replica. This means that there will be a slight delay before a write has been propagated to all replica tables.

Q: Are there CloudWatch metrics for cross-region replication?

CloudWatch metrics are available for every replication configuration. You can see the metric by selecting the replication group and navigating to the Monitoring tab. Metrics on throughput and number of record processed are available, and you can monitor for any discrepancies in the throughput of the master and replica tables.

Q: Can I have a replica in the same region as the master table?

Yes, as long as the replica table and the master table have different names, both tables can exist in the same region.

Q: Can I add or delete a replica after creating a replication group?

Yes, you can add or delete a replica from that replication group at any time.

Q: Can I delete a replica group after it is created ?

Yes, deleting the replication group will delete the EC2 instance for the group. However, you will have to delete the DynamoDB metadata table.

## **Triggers**

Q. What is DynamoDB Triggers?

DynamoDB Triggers is a feature which allows you to execute custom actions based on item-level updates on a DynamoDB table. You can specify the custom action in code.

Q. What can I do with DynamoDB Triggers?

There are several application scenarios where DynamoDB Triggers can be useful. Some use cases include sending notifications, updating an aggregate table, and connecting DynamoDB tables to other data sources.

Q. How does DynamoDB Triggers work?

The custom logic for a DynamoDB trigger is stored in an AWS Lambda function as code. To create a trigger for a given table, you can associate an AWS Lambda function to the stream (via DynamoDB Streams) on a DynamoDB table. When the table is updated, the updates are published to DynamoDB Streams. In turn, AWS Lambda reads the updates from the associated stream and executes the code in the function.

Q: What does it cost to use DynamoDB Triggers?

With DynamoDB Triggers, you only pay for the number of requests for your AWS Lambda function and the amount of time it takes for your AWS Lambda function to execute. Learn more about AWS Lambda pricing [here](http://aws.amazon.com/lambda/pricing). You are not charged for the reads that your AWS Lambda function makes to the stream (via DynamoDB Streams) associated with the table.

Q. Is there a limit to the number of triggers for a table?

There is no limit on the number of triggers for a table.

Q. What languages does DynamoDB Triggers support?

Currently, DynamoDB Triggers supports Javascript, Java, and Python for trigger functions.

Q. Is there API support for creating, editing or deleting DynamoDB triggers?

No, currently there are no native APIs to create, edit, or delete DynamoDB triggers. You have to use the AWS Lambda console to create an AWS Lambda function and associate it with a stream in DynamoDB Streams. For more information, see the [AWS Lambda FAQ page](http://aws.amazon.com/lambda/faqs).

Q. How do I create a DynamoDB trigger?

You can create a trigger by creating an AWS Lambda function and associating the event-source for the function to a stream in DynamoDB Streams. For more information, see the AWS Lambda FAQ page.

Q. How do I delete a DynamoDB trigger?

You can delete a trigger by deleting the associated AWS Lambda function. You can delete an AWS Lambda function from the AWS Lambda console or throughput an AWS Lambda API call. For more information, see the [AWS Lambda FAQ](http://aws.amazon.com/lambda/faqs) and [documentation page](http://docs.aws.amazon.com/lambda/latest/dg/API_DeleteFunction.html).

Q. I have an existing AWS Lambda function, how do I create a DynamoDB trigger using this function?

You can change the event source for the AWS Lambda function to point to a stream in DynamoDB Streams. You can do this from the DynamoDB console. In the table for which the stream is enabled, choose the stream, choose the Associate Lambda Function button, and then choose the function that you want to use for the DynamoDB trigger from the list of Lambda functions.

Q. In what regions is DynamoDB Triggers available?

DynamoDB Triggers is available in all AWS regions where AWS Lambda and DynamoDB are available.

## **Streams**

Q: What is DynamoDB Streams?

DynamoDB Streams provides a time-ordered sequence of item-level changes made to data in a table in the last 24 hours. You can access a stream with a simple API call and use it to keep other data stores up-to-date with the latest changes to DynamoDB or to take actions based on the changes made to your table.

Q: What are the benefits of DynamoDB Streams?

Using the DynamoDB Streams APIs, developers can consume updates and receive the item-level data before and after items are changed. This can be used to build creative extensions to your applications built on top of DynamoDB. For example, a developer building a global multi-player game using DynamoDB can use the DynamoDB Streams APIs to build a multi-master topology and keep the masters in sync by consuming the DynamoDB Streams for each master and replaying the updates in the remote masters. As another example, developers can use the DynamoDB Streams APIs to build mobile applications that automatically notify the mobile devices of all friends in a circle as soon as a user uploads a new selfie. Developers could also use DynamoDB Streams to keep data warehousing tools, such as Amazon Redshift, in sync with all changes to their DynamoDB table to enable real-time analytics. DynamoDB also integrates with Elasticsearch using the Amazon DynamoDB Logstash Plugin, thus enabling developers to add free-text search for DynamoDB content.

You can read more about DynamoDB Streams in our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html).

Q: How long are changes to my DynamoDB table available via DynamoDB Streams?

DynamoDB Streams keep records of all changes to a table for 24 hours. After that, they will be erased.

Q: How do I enable DynamoDB Streams?

DynamoDB Streams have to be enabled on a per-table basis. To enable DynamoDB Streams for an existing DynamoDB table, select the table through the AWS Management Console, choose the Overview tab, click the Manage Stream button, choose a view type, and then click Enable.

For more information, see our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html).

Q: How do I verify that DynamoDB Streams has been enabled?

After enabling DynamoDB Streams, you can see the stream in the AWS Management Console. Select your table, and then choose the Overview tab. Under Stream details, verify Stream enabled is set to Yes.

Q: How can I access DynamoDB Streams?

You can access a stream available through DynamoDB Streams with a simple API call using the DynamoDB SDK or using the Kinesis Client Library (KCL). KCL helps you consume and process the data from a stream and also helps you manage tasks such as load balancing across multiple readers, responding to instance failures, and checkpointing processed records.

For more information about accessing DynamoDB Streams, see our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html).

Q: Does DynamoDB Streams display all updates made to my DynamoDB table in order?

Changes made to any individual item will appear in the correct order. Changes made to different items may appear in DynamoDB Streams in a different order than they were received.

For example, suppose that you have a DynamoDB table tracking high scores for a game and that each item in the table represents an individual player. If you make the following three updates in this order:

* Update 1: Change Player 1’s high score to 100 points
* Update 2: Change Player 2’s high score to 50 points
* Update 3: Change Player 1’s high score to 125 points

Update 1 and Update 3 both changed the same item (Player 1), so DynamoDB Streams will show you that Update 3 came after Update 1. This allows you to retrieve the most up-to-date high score for each player. The stream might not show that all three updates were made in the same order (i.e., that Update 2 happened after Update 1 and before Update 3), but updates to each individual player’s record will be in the right order.

Q: Do I need to manage the capacity of a stream in DynamoDB Streams?

No, capacity for your stream is managed automatically in DynamoDB Streams. If you significantly increase the traffic to your DynamoDB table, DynamoDB will automatically adjust the capacity of the stream to allow it to continue to accept all updates.

Q: At what rate can I read from DynamoDB Streams?

You can read updates from your stream in DynamoDB Streams at up to twice the rate of the provisioned write capacity of your DynamoDB table. For example, if you have provisioned enough capacity to update 1,000 items per second in your DynamoDB table, you could read up to 2,000 updates per second from your stream.

Q: If I delete my DynamoDB table, does the stream also get deleted in DynamoDB Streams?

No, not immediately. The stream will persist in DynamoDB Streams for 24 hours to give you a chance to read the last updates that were made to your table. After 24 hours, the stream will be deleted automatically from DynamoDB Streams.

Q: What happens if I turn off DynamoDB Streams for my table?

If you turn off DynamoDB Streams, the stream will persist for 24 hours but will not be updated with any additional changes made to your DynamoDB table.

Q: What happens if I turn off DynamoDB Streams and then turn it back on?

When you turn off DynamoDB Streams, the stream will persist for 24 hours but will not be updated with any additional changes made to your DynamoDB table. If you turn DynamoDB Streams back on, this will create a new stream in DynamoDB Streams that contains the changes made to your DynamoDB table starting from the time that the new stream was created.

Q: Will there be duplicates or gaps in DynamoDB Streams?

No, DynamoDB Streams is designed so that every update made to your table will be represented exactly once in the stream.

Q: What information is included in DynamoDB Streams?

A DynamoDB stream contains information about both the previous value and the changed value of the item. The stream also includes the change type (INSERT, REMOVE, and MODIFY) and the primary key for the item that changed.

Q: How do I choose what information is included in DynamoDB Streams?

For new tables, use the CreateTable API call and specify the ViewType parameter to choose what information you want to include in the stream.  
For an existing table, use the UpdateTable API call and specify the ViewType parameter to choose what information to include in the stream.

The ViewType parameter takes the following values:

*ViewType: {  
                    { KEYS\_ONLY,   
                      NEW\_IMAGE,   
                      OLD\_IMAGE,   
                      NEW\_AND\_OLD\_IMAGES}  
                }*

The values have the following meaning: KEYS\_ONLY: Only the name of the key of items that changed are included in the stream.

* NEW\_IMAGE: The name of the key and the item after the update (new item) are included in the stream.
* OLD\_IMAGE: The name of the key and the item before the update (old item) are included in the stream.
* NEW\_AND\_OLD\_IMAGES: The name of the key, the item before (old item) and after (new item) the update are included in the stream.

Q: Can I use my Kinesis Client Library to access DynamoDB Streams?

Yes, developers who are familiar with Kinesis APIs will be able to consume DynamoDB Streams easily. You can use the DynamoDB Streams Adapter, which implements the Amazon Kinesis interface, to allow your application to use the Amazon Kinesis Client Libraries (KCL) to access DynamoDB Streams. For more information about using the KCL to access DynamoDB Streams, please see our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html).

Q: Can I change what type of information is included in DynamoDB Streams?

If you want to change the type of information stored in a stream after it has been created, you must disable the stream and create a new one using the UpdateTable API.

Q: When I make a change to my DynamoDB table, how quickly will that change show up in a DynamoDB stream?

Changes are typically reflected in a DynamoDB stream in less than one second.

Q: If I delete an item, will that change be included in DynamoDB Streams?

Yes, each update in a DynamoDB stream will include a parameter that specifies whether the update was a deletion, insertion of a new item, or a modification to an existing item. For more information on the type of update, see our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html).

Q: After I turn on DynamoDB Streams for my table, when can I start reading from the stream?

You can use the DescribeStream API to get the current status of the stream. Once the status changes to ENABLED, all updates to your table will be represented in the stream.

You can start reading from the stream as soon as you start creating it, but the stream may not include all updates to the table until the status changes to ENABLED.

Q: What is the Amazon DynamoDB Logstash Plugin for Elasticsearch?

Elasticsearch is a popular open source search and analytics engine designed to simplify real-time search and big data analytics. Logstash is an open source data pipeline that works together with Elasticsearch to help you process logs and other event data. The Amazon DynamoDB Logstash Plugin make is easy to integrate DynamoDB tables with Elasticsearch clusters.

Q: How much does the Amazon DynamoDB Logstash Plugin cost?

The Amazon DynamoDB Logstash Plugin is free to download and use.

Q: How do I download and install the Amazon DynamoDB Logstash Plugin?

The Amazon DynamoDB Logstash Plugin is available on [GitHub](https://github.com/awslabs/logstash-input-dynamodb). Read our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.DynamoDBLogstash.html) page to learn more about installing and running the plugin.

## **Storage backend for Titan**

Q: What is the DynamoDB Storage Backend for Titan?

The DynamoDB Storage Backend for Titan is a plug-in that allows you to use DynamoDB as the underlying storage layer for Titan graph database. It is a client side solution that implements index free adjacency for fast graph traversals on top of DynamoDB.

Q: What is a graph database?

A graph database is a store of vertices and directed edges that connect those vertices. Both vertices and edges can have properties stored as key-value pairs.

A graph database uses adjacency lists for storing edges to allow simple traversal. A graph in a graph database can be traversed along specific edge types, or across the entire graph. Graph databases can represent how entities relate by using actions, ownership, parentage, and so on.

Q: What applications are well suited to graph databases?

Whenever connections or relationships between entities are at the core of the data you are trying to model, a graph database is a natural choice. Therefore, graph databases are useful for modeling and querying social networks, business relationships, dependencies, shipping movements, and more.

Q: How do I get started using the DynamoDB Storage Backend for Titan?

The easiest way to get started is to launch an EC2 instance running Gremlin Server with the DynamoDB Storage Backend for Titan, using the CloudFormation templates referred to in this [documentation page](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.GremlinServerEC2.html). You can also clone the project from the GitHub repository and start by following the Marvel and Graph-Of-The-Gods tutorials on your own computer by following the instructions in the [documentation here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.DownloadingAndRunning.html). When you’re ready to expand your testing or run in production, you can switch the backend to use the DynamoDB service. Please see the [AWS documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.html) for further guidance.

Q: How does the DynamoDB Storage Backend differ from other Titan storage backends?

DynamoDB is a managed service, thus using it as the storage backend for Titan enables you to run graph workloads without having to manage your own cluster for graph storage.

Q: Is the DynamoDB Storage Backend for Titan a fully managed service?

No. The DynamoDB storage backend for Titan manages the storage layer for your Titan workload. However, the plugin does not do provisioning and managing of the client side. For simple provisioning of Titan we have developed a CloudFormation template that sets up DynamoDB Storage Backend for Titan with Gremlin Server; see the instructions available [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.GremlinServerEC2.html).

Q: How much does using the DynamoDB Storage Backend for Titan cost?

You are charged the regular DynamoDB throughput and storage costs. There is no additional cost for using DynamoDB as the storage backend for a Titan graph workload.

Q: Does DynamoDB backend provide full compatibility with the Titan feature set on other backends?

A table comparing feature sets of different Titan storage backends is available in the [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.html#Table).

Q: Which versions of Titan does the plugin support?

We have released DynamoDB storage backend plugins for Titan versions 0.5.4 and 1.0.0.

Q: I use Titan with a different backend today. Can I migrate to DynamoDB?

Absolutely. The DynamoDB Storage Backend for Titan implements the Titan KCV Store interface so you can switch from a different storage backend to DynamoDB with minimal changes to your application. For full comparison of storage backends for Titan please see our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.html#Table).

Q: I use Titan with a different backend today. How do I migrate to DynamoDB?

You can use [bulk loading](http://s3.thinkaurelius.com/docs/titan/1.0.0/bulk-loading.html) to copy your graph from one storage backend to the DynamoDB Storage Backend for Titan.

Q: How do I connect my Titan instance to DynamoDB via the plugin?

If you create a graph and Gremlin server instance with the DynamoDB Storage Backend for Titan installed, all you need to do to connect to DynamoDB is provide a principal/credential set to the [default AWS credential provider chain](http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/credentials.html#credentials-default). This can be done with an EC2 instance profile, environment variables, or the credentials file in your home folder. Finally, you need to choose a DynamoDB endpoint to connect to.

Q: How durable is my data when using the DynamoDB Storage Backend for Titan?

When using the DynamoDB Storage Backend for Titan, your data enjoys the strong protection of DynamoDB, which runs across Amazon’s proven, high-availability data centers. The service replicates data across three facilities in an AWS Region to provide fault tolerance in the event of a server failure or Availability Zone outage.

Q: How secure is the DynamoDB Storage Backend for Titan?

The DynamoDB Storage Backend for Titan stores graph data in multiple DynamoDB tables, thus is enjoys the same high security available on all DynamoDB workloads. Fine-Grained Access Control, IAM roles, and AWS principal/credential sets control access to DynamoDB tables and items in DynamoDB tables.

Q: How does the DynamoDB Storage Backend for Titan scale?

The DynamoDB Storage Backend for Titan scales just like any other workload of DynamoDB. You can choose to increase or decrease the required throughput at any time.

Q: How many vertices and edges can my graph contain?

You are limited by [Titan’s limits](http://s3.thinkaurelius.com/docs/titan/1.0.0/limitations.html) for (2^60) for the maximum number of edges and half as many vertices in a graph, as long as you use the multiple-item model for edgestore. If you use the single-item model, the number of edges that you can store at a particular out-vertex key is limited by DynamoDB’s maximum item size, currently 400kb.

Q: How large can my vertex and edge properties get?

The sum of all edge properties in the multiple-item model cannot exceed 400kb, the maximum item size. In the multiple item model, each vertex property can be up to 400kb. In the single-item model, the total item size (including vertex properties, edges and edge properties) can’t exceed 400kb.

Q: How many data models are there? What are the differences?

There are two different storage models for the DynamoDB Storage Backend for Titan – single item model and multiple item model. In the single item storage model, vertices, vertex properties, and edges are stored in one item. In the multiple item data model, vertices, vertex properties and edges are stored in different items. In both cases, edge properties are stored in the same items as the edges they correspond to.

Q: Which data model should I use?

In general, we recommend you use the multiple-item data model for the edgestore and graphindex tables. Otherwise, you either limit the number of edges/vertex-properties you can store for one out-vertex, or you limit the number of entities that can be indexed at a particular property name-value pair in graph index. In general, you can use the single-item data model for the other 4 KCV stores in Titan versions 0.5.4 and 1.0.0 because the items stored in them are usually less than 400KB each. For full list of tables that the Titan plugin creates on DynamoDB please see [here](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.BestPractices.html).

Q: Do I have to create a schema for Titan graph databases?

Titan supports automatic type creation, so new edge/vertex properties and labels will get registered on the fly (see [here](http://s3.thinkaurelius.com/docs/titan/1.0.0/schema.html) for details) with the first use. The Gremlin Structure (Edge labels=MULTI, Vertex properties=SINGLE) is used [by default](http://s3.thinkaurelius.com/docs/titan/1.0.0/titan-config-ref.html).

Q: Can I change the schema of a Titan graph database?

Yes, however, you cannot change the schema of existing vertex/edge properties and labels. For details please see [here](http://s3.thinkaurelius.com/docs/titan/1.0.0/limitations.html).

Q: How does the DynamoDB Storage Backend for Titan deal with supernodes?

DynamoDB deals with supernodes via vertex label partitioning. If you define a vertex label as partitioned in the management system upon creation, you can key different subsets of the edges and vertex properties going out of a vertex at different partition keys of the partition-sort key space in the edgestore table. This usually results in the virtual vertex label partitions being stored in different physical DynamoDB partitions, as long as your edgestore has more than one physical partition. To estimate the number of physical partitions backing your edgestore table, please see guidance in the [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForTables.html#GuidelinesForTables.Partitions).

Q: Does the DynamoDB Storage Backend for Titan support batch graph operations?

Yes, the DynamoDB Storage Backend for Titan supports batch graph with the Blueprints BatchGraph implementation and through Titan’s bulk loading configuration options.

Q: Does the DynamoDB Storage Backend for Titan support transactions?

The DynamoDB Storage Backend for Titan supports optimistic locking. That means that the DynamoDB Storage Backend for Titan can condition writes of individual Key-Column pairs (in the multiple item model) or individual Keys (in the single item model) on the existing value of said Key-Column pair or Key.

Q: Can I have a Titan instance in one region and access DynamoDB in another?

Accessing a DynamoDB endpoint in another region than the EC2 Titan instance is possible but not recommended. When running a Gremlin Server out of EC2, we recommend connecting to the DynamoDB endpoint in your EC2 instance’s region, to reduce the latency impact of cross-region requests. We also recommend running the EC2 instance in a VPC to improve network performance. The [CloudFormation template](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.TitanDB.GremlinServerEC2.html) performs this entire configuration for you.

Q: Can I use this plugin with other DynamoDB features such as update streams and cross-region replication?

You can use Cross-Region Replication with the DynamoDB Streams feature to create read-only replicas of your graph tables in other regions.

## **CloudWatch metrics**

Q: Does Amazon DynamoDB report CloudWatch metrics?  
  
Yes, Amazon DynamoDB reports several table-level metrics on CloudWatch. You can make operational decisions about your Amazon DynamoDB tables and take specific actions, like setting up alarms, based on these metrics. For a full list of reported metrics, see the [*Monitoring DynamoDB with CloudWatch*](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/MonitoringDynamoDB.html) section of our documentation.  
  
Q: How can I see CloudWatch metrics for an Amazon DynamoDB table?  
  
On the Amazon DynamoDB console, select the table for which you wish to see CloudWatch metrics and then select the Metrics tab.  
  
Q: How often are metrics reported?  
  
Most CloudWatch metrics for Amazon DynamoDB are reported in 1-minute intervals while the rest of the metrics are reported in 5-minute intervals. For more details, see the [*Monitoring DynamoDB with CloudWatch*](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/MonitoringDynamoDB.html) section of our documentation.

## **Tagging**

Q: What is a tag?

A tag is a label you assign to an AWS resource. Each tag consists of a key and a value, both of which you can define. AWS uses tags as a mechanism to organize your resource costs on your cost allocation report. For more about tagging, see the [*AWS Billing and Cost Management User Guide*](http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/allocation-what.html).  
  
Q: What DynamoDB resources can I tag?

You can tag DynamoDB tables. Local Secondary Indexes and Global Secondary Indexes associated with the tagged tables are automatically tagged with the same tags. Costs for Local Secondary Indexes and Global Secondary Indexes will show up under the tags used for the corresponding DynamoDB table.  
  
Q: Why should I use Tagging for DynamoDB?

You can use Tagging for DynamoDB for cost allocation. Using tags for cost allocation enables you to label your DynamoDB resources so that you can easily track their costs against projects or other criteria to reflect your own cost structure.

Q: How can I use tags for cost allocation?

You can use cost allocation tags to categorize and track your AWS costs. AWS Cost Explorer and detailed billing reports support the ability to break down AWS costs by tag. Typically, customers use business tags such as cost center/business unit, customer, or project to associate AWS costs with traditional cost-allocation dimensions. However, a cost allocation report can include any tag. This enables you to easily associate costs with technical or security dimensions, such as specific applications, environments, or compliance programs.

Q: How can I see costs allocated to my AWS tagged resources?

You can see costs allocated to your AWS tagged resources through either Cost Explorer or your cost allocation report.

[Cost Explorer](http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-explorer-what-is.html) is a free AWS tool that you can use to view your costs for up to the last 13 months, and forecast how much you are likely to spend for the next three months. You can see your costs for specific tags by filtering by “Tag” and then choose the tag key and value (choose “No tag” if no tag value is specified).

The cost allocation report includes all of your AWS costs for each billing period. The report includes both tagged and untagged resources, so you can clearly organize the charges for resources. For example, if you tag resources with an application name, you can track the total cost of a single application that runs on those resources. More information on cost allocation can be found in [*AWS Billing and Cost Management User Guide*](http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-alloc-tags.html).

Q: Can DynamoDB Streams usage be tagged?

No, DynamoDB Streams usage cannot be tagged at present.

Q: Will Reserved Capacity usage show up under my table tags in my bill?

Yes, DynamoDB Reserved Capacity charges per table will show up under relevant tags. Please note that Reserved Capacity is applied to DynamoDB usage on a first come, first serve basis, and across all linked AWS accounts. This means that even if your DynamoDB usage across tables and indexes is similar from month to month, you may see differences in your cost allocation reports per tag since Reserved Capacity will be distributed based on which DynamoDB resources are metered first.

Q: Will data usage charges show up under my table tags in my bill?

No, DynamoDB data usage charges are not tagged. This is because data usage is billed at an account level and not at table level.

Q: Do my tags require a value attribute?

No, tag values can be null.

Q: Are tags case sensitive?

Yes, tag keys and values are case sensitive.

Q: How many tags can I add to single DynamoDB table?

You can add up to 50 tags to a single DynamoDB table. Tags with the prefix “aws:” cannot be manually created and do not count against your tags per resource limit.

Q: Can I apply tags retroactively to my DynamoDB tables?

No, tags begin to organize and track data on the day you apply them. If you create a table on January 1st but don’t designate a tag for it until February 1st, then all of that table’s usage for January will remain untagged.

Q: If I remove a tag from my DynamoDB table before the end of the month, will that tag still show up in my bill?

Yes, if you build a report of your tracked spending for a specific time period, your cost reports will show the costs of the resources that were tagged during that timeframe.

Q. What happens to existing tags when a DynamoDB table is deleted?

When a DynamoDB table is deleted, its tags are automatically removed.

Q. What happens if I add a tag with a key that is same as one for an existing tag?

Each DynamoDB table can only have up to one tag with the same key. If you add a tag with the same key as an existing tag, the existing tag is updated with the new value.

## **Time to Live (TTL)**

Q: What is DynamoDB Time-to-Live (TTL)?

DynamoDB Time-to-Live (TTL) is a mechanism that lets you set a specific timestamp to delete expired items from your tables. Once the timestamp expires, the corresponding item is marked as expired and is subsequently deleted from the table. By using this functionality, you do not have to track expired data and delete it manually. TTL can help you reduce storage usage and reduce the cost of storing data that is no longer relevant.  
  
Q: Why do I need to use TTL?

There are two main scenarios where TTL can come in handy:

* Deleting old data that is no longer relevant – data like event logs, usage history, session data, etc. when collected can get bloated over time and the old data though stored in the system may not be relevant any more. In such situations, you are better off clearing these stale records from the system and saving the money used for storing it.
* Sometimes you may want data to be kept in DynamoDB for a specified time period in order to comply with your data retention and management policies. You might want to eventually delete this data once the obligated duration expires. Please do know however that TTL works on a best effort basis to ensure there is throughput available for other critical operations. DynamoDB will aim to delete expired items within a two-day period. The actual time taken may be longer based on the size of the data.

Q: How does DynamoDB TTL work?

To enable TTL for a table, first ensure that there is an attribute that can store the expiration timestamp for each item in the table. This timestamp needs to be in the [epoch time format](https://en.wikipedia.org/wiki/Unix_time). This helps avoid time zone discrepancies between clients and servers.

DynamoDB runs a background scanner that monitors all the items. If the timestamp has expired, the process will mark the item as expired and queue it for subsequent deletion.

Note: TTL requires a numeric DynamoDB table attribute populated with an epoch timestamp to specify the expiration criterion for the data. You should be careful when setting a value for the TTL attribute since a wrong value could cause premature item deletion.

Q: How do I specify TTL?

To specify TTL, first enable the TTL setting on the table and specify the attribute to be used as the TTL value. As you add items to the table, you can specify a TTL attribute if you would like DynamoDB to automatically delete it after its expiration. This value is the expiry time, specified in [epoch time format](https://en.wikipedia.org/wiki/Unix_time). DynamoDB takes care of the rest. TTL can be specified from the console from the overview tab for the table. Alternatively, developers can invoke the TTL API to configure TTL on the table. See our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) and our [API guide](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html).

Q: Can I set TTL on existing tables?

Yes. If a table is already created and has an attribute that can be used as TTL for its items, then you only need to enable TTL for the table and designate the appropriate attribute for TTL. If the table does not have an attribute that can be used for TTL, you will have to create such an attribute and update the items with values for TTL.

Q: Can I delete an entire table by setting TTL on the whole table?

No. While you need to define an attribute to be used for TTL at the table level, the granularity for deleting data is at the item level. That is, each item in a table that needs to be deleted after expiry will need to have a value defined for the TTL attribute. There is no option to automatically delete the entire table.

Q: Can I set TTL only for a subset of items in the table?

Yes. TTL takes affect only for those items that have a defined value in the TTL attribute. Other items in the table remain unaffected.

Q: What is the format for specifying TTL?

The TTL value should use the [epoch time format](https://en.wikipedia.org/wiki/Unix_time), which is number of seconds since January 1, 1970 UTC. If the value specified in the TTL attribute for an item is not in the right format, the value is ignored and the item won’t be deleted.

Q: How can I read the TTL value for items in my table?

The TTL value is just like any attribute on an item. It can be read the same way as any other attribute. In order to make it easier to visually confirm TTL values, the DynamoDB Console allows you to hover over a TTL attribute to see its value in human-readable local and UTC time.

Q: Can I create an index based on the TTL values assigned to items in a table?

Yes. TTL behaves like any other item attribute. You can create indexes the same as with other item attributes.

Q: Can the TTL attribute be projected to an index?

Yes. TTL attribute can be projected onto an index just like any other attribute.

Q: Can I edit the TTL attribute value once it has been set for an item?

Yes. You can modify the TTL attribute value just as you modify any other attribute on an item.

Q: Can I change the TTL attribute for a table?

Yes. If a table already has TTL enabled and you want to specify a different TTL attribute, then you need to disable TTL for the table first, then you can re-enable TTL on the table with a new TTL attribute. Note that Disabling TTL can take up to one hour to apply across all partitions, and you will not be able to re-enable TTL until this action is complete.

Q: Can I use AWS Management Console to view and edit the TTL values?

Yes. The AWS Management Console allows you to easily view, set or update the TTL value.

Q: Can I set an attribute within a JSON document to be the TTL attribute?

No. We currently do not support specifying an attribute in a JSON document as the TTL attribute. To set TTL, you must explicitly add the TTL attribute to each item.

Q: Can I set TTL for a specific element in a JSON Document?

No. TTL values can only be set for the whole document. We do not support deleting a specific item in a JSON document once it expires.

Q: What if I need to remove the TTL on specific items?

Removing TTL is as simple as removing the value assigned to the TTL attribute or removing the attribute itself for an item.

Q: What if I set the TTL timestamp value to sometime in the past?

Updating items with an older TTL values is allowed. Whenever the background process checks for expired items, it will find, mark and subsequently delete the item. However, if the value in the TTL attribute contains an [epoch value](https://en.wikipedia.org/wiki/Epoch_(reference_date)) for a timestamp that is over 5 years in the past, DynamoDB will ignore the timestamp and not delete the item. This is done to mitigate accidental deletion of items when really low values are stored in the TTL attribute.

Q: What is the delay between the TTL expiry on an item and the actual deletion of that item?

TTL scans and deletes expired items using background throughput available in the system. As a result, the expired item may not be deleted from the table immediately. DynamoDB will aim to delete expired items within a two-day window on a best-effort basis, to ensure availability of system background throughput for other data operations. The exact duration within which an item truly gets deleted after expiration will be specific to the nature of the workload and the size of the table.

Q: What happens if I try to query or scan for items that have been expired by TTL?

Given that there might be a delay between when an item expires and when it actually gets deleted by the background process, if you try to read items that have expired but haven’t yet been deleted, the returned result will include the expired items. You can filter these items out based on the TTL value if the intent is to not show expired items.

Q: What happens to the data in my Local Secondary Index (LSI) if it has expired?

The impact is the same as any delete operation. The local secondary index is stored in the same partition as the item itself. Hence if an item is deleted it immediately gets removed from the Local Secondary Index.

Q: What happens to the data in my Global Secondary Index (GSI) if it has expired?

The impact is the same as any delete operation. A Global Secondary Index (GSI) is eventually consistent and so while the original item that expired will be deleted it may take some time for the GSI to get updated.

Q: How does TTL work with DynamoDB Streams?

The expiry of data in a table on account of the TTL value triggering a purge is recorded as a delete operation. Therefore, the Streams will also have the delete operation recorded in it. The delete record will have an additional qualifier so that you can distinguish between your deletes and deletes happening due to TTL. The stream entry will be written at the point of deletion, not the TTL expiration time, to reflect the actual time at which the record was deleted. See our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) and our [API guide](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html).

Q: When should I use the delete operation vs TTL?

TTL is ideal for removing expired records from a table. However, this is intended as a best-effort operation to help you remove unwanted data and does not provide a guarantee on the deletion timeframe. As a result, if data in your table needs to be deleted within a specific time period (often immediately), we recommend using the delete command.

Q: Can I control who has access to set or update the TTL value?

Yes. The TTL attribute is just like any other attribute on a table. You have the ability to control access at an attribute level on a table. The TTL attribute will follow the regular access controls specified for the table.

Q: Is there a way to retrieve the data that has been deleted after TTL expiry?

No. Expired items are not backed up before deletion. You can leverage the DynamoDB Streams to keep track of the changes on a table and restore values if needed. The delete record is available in Streams for 24 hours since the time it is deleted.

Q: How can I know whether TTL is enabled on a table?

You can get the status of TTL at any time by invoking the DescribeTable API or viewing the table details in the DynamoDB console. See our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) and our [API guide](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html).

Q: How do I track the items deleted by TTL?

If you have DynamoDB streams enabled, all TTL deletes will show up in the DynamoDB Streams and will be designated as a system delete in order to differentiate it from an explicit delete done by you. You can read the items from the streams and process them as needed. They can also write a Lambda function to archive the item separately. See our [documentation](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) and our [API guide](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html).

Q: Do I have to pay a specific fee to enable the TTL feature for my data?

No. Enabling TTL requires no additional fees.

Q: How will enabling TTL affect my overall provisioned throughput usage?

The scan and delete operations needed for TTL are carried out by the system and does not count toward your provisioned throughput or usage.

Q: Will I have to pay for the scan operations to monitor TTL?

No. You are not charged for the internal scan operations to monitor TTL expiry for items. Also these operations will not affect your throughput usage for the table.

Q: Do expired items accrue storage costs till they are deleted?

Yes. After an item has expired it is added to the delete queue for subsequent deletion. However, until it has been deleted, it is just like any regular item that can be read or updated and will incur storage costs.

Q: If I query for an expired item, does it use up my read capacity?

Yes. This behavior is the same as when you query for an item that does not exist in the table.

## **Amazon DynamoDB Accelerator (DAX)**

Q. What is DynamoDB Accelerator (DAX)?

Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory [cache](https://aws.amazon.com/caching/) for [DynamoDB](https://aws.amazon.com/dynamodb/) that enables you to benefit from fast in-memory performance for demanding applications. DAX improves the performance of read-intensive DynamoDB workloads so repeat reads of cached data can be served immediately with extremely low latency, without needing to be re-queried from DynamoDB. DAX will automatically retrieve data from DynamoDB tables upon a cache miss. Writes are designated as write-through (data is written to DynamoDB first and then updated in the DAX cache).

Just like DynamoDB, DAX is fault-tolerant and scalable. A DAX cluster has a primary node and zero or more read-replica nodes. Upon a failure for a primary node, DAX will automatically fail over and elect a new primary. For scaling, you may add or remove read replicas.

To get started, create a DAX cluster, download the DAX SDK for Java or Node.js (compatible with the DynamoDB APIs), re-build your application to use the DAX client as opposed to the DynamoDB client, and finally point the DAX client to the DAX cluster endpoint. You do not need to implement any additional caching logic into your application as DAX client implements the same API calls as DynamoDB.

Q. What does "DynamoDB-compatible" mean?

It means that most of the code, applications, and tools you already use today with DynamoDB can be used with DAX with little or no change. The DAX engine is designed to support the DynamoDB APIs for reading and modifying data in DynamoDB. Operations for table management such as CreateTable/DescribeTable/UpdateTable/DeleteTable are not supported.

Q. What is in-memory caching, and how does it help my application?

Caching improves application performance by storing critical pieces of data in memory for low-latency and high throughput access. In the case of DAX, the results of DynamoDB operations are cached. When an application requests data that is stored in the cache, DAX can serve that data immediately without needing to run a query against the regular DynamoDB tables. Data is aged or evicted from DAX by specifying a Time-to-Live (TTL) value for the data or, once all available memory is exhausted, items will be evicted based on the Least Recently Used (LRU) algorithm.

Q. What is the consistency model of DAX?

When reading data from DAX, users can specify whether they want the read to be eventually consistent or strongly consistent:

Eventually Consistent Reads (Default) – the eventual consistency option maximizes your read throughput and minimizes latency. On a cache hit, the DAX client will return the result directly from the cache. On a cache miss, DAX will query DynamoDB, update the cache, and return the result set. It should be noted that an eventually consistent read might not reflect the results of a recently completed write. If your application requires full consistency, then we suggest using strongly consistent reads.

Strongly Consistent Reads — in addition to eventual consistency, DAX also gives you the flexibility and control to request a strongly consistent read if your application, or an element of your application, requires it. A strongly consistent read is pass-through for DAX, does not cache the results in DAX, and returns a result that reflects all writes that received a successful response in DynamoDB prior to the read.

Q. What are the common use cases for DAX?  
  
DAX has a number of use cases that are not mutually exclusive:

Applications that require the fastest possible response times for reads. Some examples include real-time bidding, social gaming, and trading applications. DAX delivers fast, in-memory read performance for these use cases.

Applications that read a small number of items more frequently than others. For example, consider an e-commerce system that has a one-day sale on a popular product. During the sale, demand for that product (and its data in DynamoDB) would sharply increase, compared to all of the other products. To mitigate the impacts of a "hot" key and a non-uniform data distribution, you could offload the read activity to a DAX cache until the one-day sale is over.

Applications that are read-intensive, but are also cost-sensitive. With DynamoDB, you provision the number of reads per second that your application requires. If read activity increases, you can increase your table’s provisioned read throughput (at an additional cost). Alternatively, you can offload the activity from your application to a DAX cluster, and reduce the amount of read capacity units you'd need to purchase otherwise.

Applications that require repeated reads against a large set of data. Such an application could potentially divert database resources from other applications. For example, a long-running analysis of regional weather data could temporarily consume all of the read capacity in a DynamoDB table, which would negatively impact other applications that need to access the same data. With DAX, the weather analysis could be performed against cached data instead.

How It Works  
  
Q. What does DAX manage on my behalf?

DAX is a fully-managed cache for DynamoDB. It manages the work involved in setting up dedicated caching nodes, from provisioning the server resources to installing the DAX software. Once your DAX cache cluster is set up and running, the service automates common administrative tasks such as failure detection and recovery, and software patching. DAX provides detailed CloudWatch monitoring metrics associated with your cluster, enabling you to diagnose and react to issues quickly. Using these metrics, you can set up thresholds to receive CloudWatch alarms. DAX handles all of the data caching, retrieval, and eviction so your application does not have to. You can simply use the DynamoDB API to write and retrieve data, and DAX handles all of the caching logic behind the scenes to deliver improved performance.

Q. What kinds of data does DAX cache?

All read API calls will be cached by DAX, with strongly consistent requests being read directly from DynamoDB, while eventually consistent reads will be read from DAX if the item is available. Write API calls are write-through (synchronous write to DynamoDB which is updated in the cache upon a successful write).

The following API calls will result in examining the cache. Upon a hit, the item will be returned. Upon a miss, the request will pass through, and upon a successful retrieval the item will be cached and returned.  
  
• GetItem  
• BatchGetItem  
• Query  
• Scan

The following API calls are write-through operations.  
  
• BatchWriteItem  
• UpdateItem  
• DeleteItem  
• PutItem

Q. How does DAX handle data eviction?

DAX handles cache eviction in three different ways. First, it uses a Time-to-Live (TTL) value that denotes the absolute period of time that an item is available in the cache. Second, when the cache is full, a DAX cluster uses a Least Recently Used (LRU) algorithm to decide which items to evict. Third, with the write-through functionality, DAX evicts older values as new values are written through DAX. This helps keep the DAX item cache consistent with the underlying data store using a single API call.

Q. Does DAX work with DynamoDB GSIs and LSIs?

Just like DynamoDB tables, DAX will cache the result sets from both query and scan operations against both DynamoDB GSIs and LSIs.

Q. How does DAX handle Query and Scan result sets?

Within a DAX cluster, there are two different caches: 1) item cache and 2) query cache. The item cache manages GetItem, PutItem, and DeleteItem requests for individual key-value pairs. The query cache manages the result sets from Scan and Query requests. In this regard, the Scan/Query text is the “key” and the result set is the “value”. While both the item cache and the query cache are managed in the same cluster (and you can specify different TTL values for each cache), they do not overlap. For example, a scan of a table does not populate the item cache, but instead records an entry in the query cache that stores the result set of the scan.

Q. Does an update to the item cache either update or invalidate result sets in my query cache?

No. The best way to mitigate inconsistencies between result sets in the item cache and query cache is to set the TTL for the query cache to be of an acceptable period of time for which your application can handle such inconsistencies.

Q. Can I connect to my DAX cluster from outside of my VPC?

The only way to connect to your DAX cluster from outside of your VPC is through a VPN connection.

Q. When using DAX, what happens if my underlying DynamoDB tables are throttled?

If DAX is either reading or writing to a DynamoDB table and receives a throttling exception, DAX will return the exception back to the DAX client. Further, the DAX service does not attempt server-side retries.

Q. Does DAX support pre-warming of the cache?

DAX utilizes lazy-loading to populate the cache. What this means is that on the first read of an item, DAX will fetch the item from DynamoDB and then populate the cache. While DAX does not support cache pre-warming as a feature, the DAX cache can be pre-warmed for an application by running an external script/application that reads the desired data.

Q. How does DAX work with the DynamoDB TTL feature?

Both DynamoDB and DAX have the concept of a "TTL" (or Time to Live) feature. In the context of DynamoDB, TTL is a feature that enables customers to age out their data by tagging the data with a particular attribute and corresponding timestamp. For example, if customers wanted data to be deleted after the data has aged for one month, they would use the DynamoDB TTL feature to accomplish this task as opposed to managing the aging workflow themselves.

In the context of DAX, TTL specifies the duration of time in which an item in cache is valid. For instance, if a TTL is set for 5-minutes, once an item has been populated in cache it will continue to be valid and served from the cache until the 5-minute period has elapsed. Although not central to this conversation, TTL can be preempted by writes to the cache for the same item or if there is memory pressure on the DAX node and LRU evicts the items as it was the least recently used.

While TTL for DynamoDB and DAX will typically be operating in very different time scales (i.e., DAX TTL operating in the scope of minutes/hours and DynamoDB TTL operating in the scope of weeks/months/years), there is a potential when customers will need to be present of how these two features affect each other. For example, let's imagine a scenario in which the TTL value for DynamoDB is less than the TTL value for DAX. In this scenario, an item could conceivably be cached in DAX and subsequently deleted from DynamoDB via the DynamoDB TTL feature. The result would be an inconsistent cache. While we don’t expect this scenario to happen often as the time scales for the two features are typically order of magnitude apart, it is good to be aware of how the two features relate to each other.

Q. Does DAX support cross-region replication?

Currently DAX only supports DynamoDB tables in the same AWS region as the DAX cluster.

Q. Is DAX supported as a resource type in AWS CloudFormation?

Yes. You can create, update and delete DAX clusters, parameter groups, and subnet groups using AWS CloudFormation.

Getting Started  
  
Q. How do I get started with DAX?  
You can create a new DAX cluster through the AWS console or AWS SDK to obtain the DAX cluster endpoint. A DAX-compatible client will need to be downloaded and used in the application with the new DAX endpoint.

Q. How do I create a DAX cluster?

You can create a DAX cluster using the AWS Management Console or the DAX CLI. DAX clusters range from a 13 GiB cache (dax.r3.large) to 216 GiB (dax.r3.8xlarge) in the R3 instance types, 15.25GiB cache (dax.r4.large) to 488 GiB (dax.r4.16xlarge) in the R4 instance types, and 2 GiB (dax.t2.small) to 4 GiB (data.t2.medium) for the smaller T2 instance types. With a few clicks in the console or a single API call, you can add up to 10 replicas to your cluster for increased throughput.

The single node configuration enables you to get started with DAX quickly and cost-effectively, and then scale out to a multi-node configuration as your needs grow. The multi-node configuration consists of a primary node that manages writes, and up to nine read replica nodes. The primary node is provisioned for you automatically.

Specify your preferred subnet groups and Availability Zones (optional), the number of nodes, node types, VPC subnet group, and other system settings. After you've chosen your desired configuration, DAX will provision the required resources and set up your caching cluster specifically for DynamoDB.

Q. Does all my data need to fit in memory to use DAX?

No. DAX will utilize the available memory on the node. Using either TTL and/or LRU, items will be expunged to make space for new data when the memory space is exhausted.

Q. Which languages does DAX support?

DAX provides DAX SDKs for Java, Node.js, Python, and .NET that you can download. We are actively working on adding support for additional client SDKs.

Q. Can I use DAX and DynamoDB at the same time?

Yes, you can access the DAX endpoint and DynamoDB at the same time through different clients. However, DAX will not be able to detect changes in data written directly to DynamoDB unless these changes are explicitly populated in to DAX through a read operation after the update was made directly to DynamoDB.

Q. Can I utilize multiple DAX clusters for the same DynamoDB table?

Yes, you can provision multiple DAX clusters for the same DynamoDB table. These clusters will provide different endpoints that can be used for different use cases, ensuring optimal caching for each scenario. Two DAX clusters will be independent of each other and will not share state or updates, so users are best served using these for completely different tables.

Q. How will I know what DAX node type I'll need for my workload?

Sizing of a DAX cluster is an iterative process. It is recommended to provision a three-node cluster (for high availability) with enough memory to fit the application's working set in memory. Based on the performance and throughput of the application, the utilization of the DAX cluster, and the cache hit/miss ratio you may need to scale your DAX cluster to achieve desired results.

Q. On what kinds of Amazon EC2 instances can DAX run?

See the [Amazon DynamoDB Pricing](https://aws.amazon.com/dynamodb/pricing/) page for the latest instance types supported by DAX.

Q. Does DAX support Reserved Instances or the AWS Free Usage Tier?

Currently DAX only supports on-demand instances.

Q. How is DAX priced?

DAX is priced per node-hour consumed, from the time a node is launched until it is terminated. Each partial node-hour consumed will be billed as a full hour. Pricing applies to all individual nodes in the DAX cluster. For example, if you have a three node DAX cluster, you will be billed for each of the separate nodes (three nodes in total) on an hourly basis.

Availability

Q. How can I achieve high availability with my DAX cluster?

DAX provides built-in multi-AZ support, letting you choose the preferred availability zones for the nodes in your DAX cluster. DAX uses asynchronous replication to provide consistency between the nodes, so that in the event of a failure, there will be additional nodes that can service requests. To achieve high availability for your DAX cluster, for both planned and unplanned outages, we recommend that you deploy at least three nodes in three separate availability zones. Each AZ runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable.

Q. What happens if a DAX node fails?

If the primary node fails, DAX automatically detects the failure, selects one of the available read replicas, and promotes it to become the new primary. In addition, DAX provisions a new node in the same availability zone of the failed primary; this new node replaces the newly-promoted read replica. If the primary fails due to a temporary availability zone disruption, the new replica will be launched as soon as the AZ has recovered. If a single-node cluster fails, DAX launches a new node in the same availability zone.

Scalability

Q. What type of scaling does DAX support?

DAX supports two scaling options today. The first option is read scaling to gain additional throughput by adding read replicas to a cluster. A single DAX cluster supports up to 10 nodes, offering millions of requests per second. Adding or removing additional replicas is an online operation. The second way to scale a cluster is to scale up or down by selecting larger or smaller r3 instance types. Larger nodes will enable the cluster to store more of the application's data set in memory and thus reduce cache misses and improve overall performance of the application. When creating a DAX cluster, all nodes in the cluster must be of the same instance type. Additionally, if you desire to change the instance type for your DAX cluster (i.e., scale up from r3.large to r3.2xlarge), you must create a new DAX cluster with the desired instance type. DAX does not currently support online scale-up or scale-down operations.

Q. How do I write-scale my application?

Within a DAX cluster, only the primary node handles write operations to DynamoDB. Thus, adding more nodes to the DAX cluster will increase the read throughput, but not the write throughput. To increase write throughput for your application, you will need to either scale-up to a larger instance size or provision multiple DAX clusters and shard your key-space in the application layer.

Monitoring

Q. How do I monitor the performance of my DAX cluster?  
Metrics for CPU utilization, cache hit/miss counts and read/write traffic to your DAX cluster are available via the AWS Management Console or Amazon CloudWatch APIs. You can also add additional, user-defined metrics via Amazon CloudWatch's custom metric functionality. In addition to CloudWatch metrics, DAX also provides information on cache hit, miss, query and cluster performance via the AWS Management Console.

Maintenance

Q. What is a maintenance window? Will my DAX cluster be available during software maintenance?

You can think of the DAX maintenance window as an opportunity to control when cluster modifications such as software patching occur. If a "maintenance" event is scheduled for a given week, it will be initiated and completed at some point during the maintenance window you identify.

Required patching is automatically scheduled only for patches that are security and reliability related. Such patching occurs infrequently (typically once every few months). If you do not specify a preferred weekly maintenance window when creating your cluster, a default value will be assigned. If you wish to modify when maintenance is performed on your behalf, you can do so by modifying your cluster in the AWS Management Console or by using the UpdateCluster API. Each of your clusters can have different preferred maintenance windows.

For multi-node clusters, updates in the cluster are performed serially, and one node will be updated at a time. After the node is updated, it will sync with one of the peers in the cluster so that the node has the current working set of data. For a single-node cluster, we will provision a replica (at no charge to you), sync the replica with the latest data, and then perform a failover to make the new replica the primary node. This way, you don’t lose any data during an upgrade for a one-node cluster.

## **Global tables**

Q. What is DynamoDB Global Tables?

DynamoDB Global Tables is a new multi-master, cross-region replication capability of DynamoDB to support data access locality and regional fault tolerance for database workloads. Applications can now perform reads and writes to DynamoDB in AWS regions around the world, with changes in any region propagated to every region where a table is replicated.

Q. Why should I use Global Tables?

Global Tables enable you to build applications that take advantage of data locality to reduce overall latency. Your applications can read/write data to the region closest to your end users, thereby improving the overall responsiveness of your application. In addition, Global Tables enables your applications to stay highly available even in the unlikely event of isolation or degradation of an entire region.

Q. Which AWS regions support Global Tables?

Global Tables is currently supported in five regions: US East (Ohio), US East (N. Virginia), US West (Oregon), EU (Ireland), and EU (Frankfurt).

Q. How many AWS regions can I replicate to?

You can have one replica table per region, for as many regions in which Global Tables is supported.

Q. What consistency guarantees does Global Table provide?

Global Tables ensures eventual consistency, meaning that any update made to any item in any replica table will be replicated to all other replicas in the same global table. If there are multiple updates to the same item, all replicas in the global table will agree on the latest update, and hence all replicas will converge continually towards a state in which they store identical data.

Q. What latency can I expect when accessing a replica table?

If your application is hosted in a region with a global table, accessing the local replica table through its regional endpoint will exhibit the same single-digit millisecond latencies that you have come to expect from DynamoDB.

Q. What is the pricing for Global Tables?

Please refer to the [pricing page](https://aws.amazon.com/dynamodb/pricing/#pricing-details) for details.

Q. Can I use Global Tables with DynamoDB’s free tier?

Yes. You can create a global table with replica tables that each fall within free tier, as long as it is replicated in no more than two AWS regions.

Q. Does DynamoDB Global Tables support cross-account access (i.e. replicating data from one AWS account to another)?

A: No. Currently, Global Tables replicates data among regions within a single AWS account.

## **Backup and restore**

Q. What backup options does DynamoDB support for my tables?

You can enable point-in-time recovery (PITR) as well as create an on-demand backup for your Amazon DynamoDB tables.

PITR provides continuous backups of your DynamoDB table data. You can enable PITR with a single click from the AWS Management Console or a single API call. When enabled, DynamoDB maintains continuous backups of your table up to the last 35 days. You can recover a copy of that table to a previous state at any point in time from the moment you enable PITR, up to a maximum of the 35 preceding days. PITR provides continuous backups until you explicitly disable it.

On-demand backups allow you to create backups of DynamoDB table data and its settings. You can initiate an on-demand backup at any time with a single click from the AWS Management Console or a single API call. You can restore the backups to a new DynamoDB table in the same AWS Region at any time.

For both PITR and on-demand backups, DynamoDB encrypts, catalogs, and stores the backups automatically.

Q. Why should I use PITR and on-demand backups?

Amazon DynamoDB provides two complementary options for your data protection and archival.

PITR gives your DynamoDB tables continuous protection from accidental writes and deletes. With PITR, you do not have to worry about creating, maintaining, or scheduling backups. You enable PITR on your table and your backup is available for restore at any point in time from the moment you enable it, up to a maximum of the 35 preceding days. For example, imagine a test script writing accidentally to a production DynamoDB table. You could recover your table to any point in time within the last 35 days.

On-demand backups help with long-term archival requirements for regulatory compliance. On-demand backups give you full-control of managing the lifecycle of your backups, from creating as many backups as you need to retaining these for as long as you need.

Q. How do I enable or disable PITR?

You can enable or disable PITR by using the Amazon DynamoDB console, the AWS CLI, or any AWS SDK.

Q. How do I initiate an on-demand backup?

You can initiate an on-demand backup from the DynamoDB console, the AWS CLI, or any AWS SDK. All on-demand backup actions—create, restore, and delete—are available from the Backups tab of a DynamoDB table. You also can browse the full list of on-demand backups in your account from the navigation pane of the console. For more information about the on-demand backup and restore API, see the [DynamoDB API Reference documentation](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html).

Q. How long are on-demand backups and PITR backups retained?

DynamoDB retains on-demand backups until you delete those backups. PITR retains backups from the last 35 days.

Q. What happens with my backups after I delete the source table?

On-demand backups remain accessible even after you delete the source table. When you delete a table that has PITR enabled, the system stops tracking changes. If you need to recover a table that you deleted, you need to contact AWS Support to restore that table from a PITR backup within the 35-day recovery window.

Q. Do backups consume provisioned read or write capacity of the source table?

No. DynamoDB executes backup and restore actions within the service and does not consume any provisioned read or write capacity of the source table.

Q. Do backups affect the performance of my table?

No.

Q. Where are my backups stored?

Backups are stored using the highly durable managed storage of DynamoDB to provide a simple, high-performing, and easy experience.

Q. What does DynamoDB restore with PITR?

When you restore a backup created by PITR, DynamoDB restores your table data to the state based on the selected date and time (day:hour:minute:second) to a new table. The settings for that new table come from the current settings of the source table and include local secondary indexes, global secondary indexes, encryption, and read capacity unit and write capacity unit settings.

Q. Can I restore a table in a different account or AWS Region with PITR?

No. You can restore a table only to the same AWS Region and account where the source table resided.

Q. Can a table restore overwrite an existing table?

No. When you restore your table from backup, the process always writes to a new table.

Q. Which AWS Regions support on-demand backups and PITR?

See [Pricing Details](https://aws.amazon.com/dynamodb/pricing/#pricing-details) for the latest regional availability and pricing.

Q. What does DynamoDB retain with on-demand backups and PITR?

Along with the data, on-demand backups retain your table’s settings for read capacity units, write capacity units, local secondary indexes, global secondary indexes, streams, and encryption. On-demand backups do not preserve auto scaling policies, PITR settings, Time To Live (TTL), tags, and CloudWatch metrics and alarms.

PITR backs up the content of the table and uses the current settings of that table at the time of restore.

Q. What does DynamoDB restore from an on-demand backup?

On restore, the destination table is set with the same provisioned read capacity units and write capacity units as the source table, as recorded at the time backup was requested. The process restores the local secondary indexes and the global secondary indexes and does not restore streams and TTL data.

Q. What happens if I disable PITR and later reenable it?

If you disable PITR and later reenable it on a table, you reset the start time for which you can recover that table. For example, when you disable PITR, you remove the restore snapshot and logs. As a result, you cannot recover that table. If you later enable PITR again for that table, DynamoDB will create a new sequence of snapshots and logs.

Q. How long do backup and restore operations take to complete?

DynamoDB processes backup requests in seconds and those backups become available for restore immediately. The time it takes you to restore a table will vary based on multiple factors, and the restore times are not always correlated directly to the size of the table. For example, because of parallelization, it is possible that restoring a 300 GB table could take the same amount of time as restoring a 3 GB table. Here are some of the considerations for restore times:

1. You restore backups to a new table. It can take up to 20 minutes (even if the table is empty) to perform all the actions to create the new table and initiate the restore process.

2. For tables with even data distribution across your primary keys, the restore time is proportional to the largest single partition by item count and not the overall table size. For the largest partitions with billions of items, a restore could take less than 10 hours.

3. If your source table contains data with significant skew, the time to restore may increase. For example, if your table’s primary key is using the month of the year for partitioning and all your data is from the month of December, you have skewed data.

Q. Is there a limit to the number of on-demand backups I can create?

No. There is no limit to the number of on-demand backups you can create for a table. There is also no limit to the number of backups you can retain in your AWS account.

Q. How much does it cost to use on-demand backups and PITR?

See [Pricing Details](https://aws.amazon.com/dynamodb/pricing/#pricing-details) to learn more about pricing for on-demand backups and PITR.

Q. Can I use on-demand backups or PITR to back up my DynamoDB tables and restore these to another AWS account?

No. Currently, you must recover a table from backup to the same AWS Region within the same AWS account where you took the backup.

Q. Can I back up a DynamoDB global table?

You can create an on-demand backup or enable PITR on each local replica of your global table. When you restore from backup, it creates a new independent table that is not part of the global table.

Q. Can I back up an encrypted table?

Yes. You can create an on-demand backup or enable PITR on an encrypted table.

Q. How is on-demand backup different from import and export using AWS Data Pipeline?

The AWS Data Pipeline import and export capability uses your table’s provisioned read and write throughput capacity and impacts table performance because DynamoDB table data is moved using full table scans. On-demand backups, on the other hand, do not consume any throughput capacity, with no impact to table performance and availability. The import and export options under the Actions menu in the DynamoDB console have been removed, but they are still accessible from the AWS Data Pipeline console.

## **Encryption at rest**

Q.  What is DynamoDB encryption at rest?

DynamoDB encryption at rest provides you with the ability to enable encryption for the data persisted (data at rest) in your DynamoDB tables. This includes - base table, local secondary indexes, and global secondary indexes. Encryption at rest automatically integrates with AWS Key Management Service (KMS) for managing the keys used for encrypting your tables.

Q.  Why do I need to use encryption at rest?

Encryption at rest is a managed server side encryption feature using AWS KMS keys stored in your AWS account. You do not have to implement and maintain additional code to encrypt data before it is sent to DynamoDB and decrypt data after it is retrieved. Once encryption at rest is enabled for a DynamoDB table, your application will work seamlessly without any other changes.

Q.  How do I encrypt a table?

You can enable encryption at rest for your new DynamoDB tables using the console, AWS CLI, or API. At present, you cannot enable encryption at rest for an existing DynamoDB table.

Q.  Are my Global Secondary Indexes (GSI) and Local Secondary Indexes (LSI) encrypted in encryption at rest?

Yes, Global Secondary Indexes (GSI) and Local Secondary Indexes (LSI) associated with an encrypted table are encrypted by default using the same key that is used to encrypt the table.

Q.  What are the additional costs for using DynamoDB encryption at rest?

There are no additional DynamoDB costs for using DynamoDB encryption at rest. However, KMS charges will apply for using a service default key. These charges can be seen on the [AWS KMS pricing page](https://aws.amazon.com/kms/pricing/).

Q.  Can I encrypt DynamoDB Streams?

Currently, you cannot enable encryption at rest for DynamoDB Streams. If encryption at rest is a compliance/regulatory requirement, we recommend turning off DynamoDB Streams for encrypted tables.

Q.  Are DynamoDB On-Demand Backups encrypted as well?

Yes, On-Demand Backups of encrypted DynamoDB tables are encrypted (using S3’s Server-Side Encryption). At present, these backups are partially encrypted using your service default keys and service managed keys. We are working towards encrypting all data related to On-Demand Backups using only customer owned KMS keys.

Q.  How does encryption at rest encrypt my data?

DynamoDB uses envelope encryption to encrypt your data in which it uses a hierarchy of encryption keys to encrypt the database. You use AWS KMS to manage the top-level encryption keys in this hierarchy. Once your data is encrypted, Amazon DynamoDB handles decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Q.  How do I manage my keys used for encryption at rest?

DynamoDB is integrated with AWS KMS for ease of managing the key(s) used to encrypt your tables. DynamoDB encryption at rest uses service default keys (specific to DynamoDB) stored in your KMS account. If a service default key does not exist when creating your encrypted DynamoDB table, KMS will automatically create a new key for you that will be used with encrypted tables created in the future. For more information, see the [AWS Key Management Service Developer Guide](http://docs.aws.amazon.com/kms/latest/developerguide/overview.html).

Q.  Which encryption keys can I choose to encrypt my DynamoDB table?

Currently, you can only use the service default key used for your DynamoDB tables. If this key doesn’t exist, it will be created.

Q.  What is the role of my service default key in AWS Key Management Service (KMS) in encryption at rest?

DynamoDB cannot read your table data without access to your KMS service default key. DynamoDB uses envelope encryption and key hierarchy to encrypt data. Your KMS encryption key is used to encrypt the root key of this key hierarchy. For more information, see [How Envelope Encryption Works with Supported AWS Service](http://docs.aws.amazon.com/kms/latest/developerguide/workflow.html).

Q.  Can I use different service default keys for different tables?

No, DynamoDB uses a single service default key for encrypting all of your DynamoDB tables.

Q.  Can I encrypt only a subset of items in a table?

No. Encryption at Rest works at a table level granularity.

Q.  How can I check if encryption at rest is enabled on my table?

From the console, you can get the status of encryption from the "Table details" section of the "Overview" tab. You can also use DescribeTable command to get the status of encryption on the table.

Q.  Can I disable encryption at rest on a table once it is enabled?

No, you cannot disable encryption at rest on an encrypted table.

Q.  How is encryption at rest different from the DynamoDB client side encryption library?

The client side encryption library - Amazon DynamoDB Encryption Client for Java - performs encryption and decryption of your data at the client side (in your application using the AWS SDK). The encryption keys reside on the client side. Since DynamoDB does not have access to your encryption keys, DynamoDB cannot access your decrypted data. The server side encryption at rest feature encrypts your data just before storing it in DynamoDB tables. The encryption and decryption of your data is performed at the server side by DynamoDB using your specified KMS encryption keys. You can still use full querying capabilities for your encrypted data.

Q.  Does encryption at rest protect my data while it is being transferred over the network?

No. Encryption at rest only encrypts data while it is static (at rest) on a persistent storage media. You have to ensure protection of data while it is actively moving over a public or a private network (data in transit) by encrypting sensitive data on the client side or using encrypted connections (TLS).

Q.  What encryption algorithm does encryption at rest use?

Encryption at rest encrypts your data using 256-bit AES encryption.

Q.  How does encryption at rest work with DynamoDB Global Tables?

You can enable encryption at rest on your Global Table replicas. Note that Global Tables uses DynamoDB Streams, which does not yet support Encryption at Rest. As a result, replicated data on DynamoDB Streams will not be encrypted at rest.

## **VPC endpoints**

Q. What are VPC endpoints for Amazon DynamoDB?

Amazon Virtual Private Cloud (VPC) is an AWS service that provides users a virtual private cloud, by provisioning a logically isolated section of the AWS Cloud. VPC endpoints for Amazon DynamoDB are logical entities within a VPC that create a private connection between a VPC and DynamoDB without requiring access over the internet, through a network address translation (NAT) device, or a VPN connection. For more information about VPC endpoints, see [VPC Endpoints](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-endpoints.html).

Q. Why should I use VPC endpoints for DynamoDB?

In the past, the main way of accessing Amazon DynamoDB from within a VPC was to traverse the internet, which may have required complex configurations such as firewalls and VPNs. VPC endpoints for DynamoDB improve privacy and security for customers, especially those dealing with sensitive workloads with compliance and audit requirements, by enabling private access to DynamoDB from within a VPC without the need for an internet gateway or NAT gateway. In addition, VPC endpoints for DynamoDB support AWS Identity and Access Management (IAM) policies to simplify DynamoDB access control. You can now easily restrict access to your DynamoDB tables to a specific VPC endpoint.

Q. How do I get started using VPC endpoints for DynamoDB?

You can create VPC endpoints for Amazon DynamoDB by using the AWS Management Console, AWS SDK, or AWS Command Line Interface (CLI). You must specify the VPC and existing route tables in the VPC, and describe the IAM policy to attach to the endpoint. A route is automatically added to each of the specified VPC’s route tables.

Q. Do VPC endpoints for DynamoDB ensure that traffic will not be routed outside of the Amazon network?

Yes, when using VPC endpoints for Amazon DynamoDB, data packets between DynamoDB and your VPC will remain in the Amazon network.

Q. Can I connect to a DynamoDB table in an AWS Region different from my VPC using VPC endpoints for DynamoDB?

No, VPC endpoints can be created only for Amazon DynamoDB tables in the same AWS Region as the VPC.

Q. Do VPC endpoints for DynamoDB limit throughput to DynamoDB?

No, you will continue to get the same throughput to Amazon DynamoDB as you do today from an instance with a public IP within your VPC.

Q. What is the price of using VPC endpoints for DynamoDB?

There is no additional cost for using VPC endpoints for Amazon DynamoDB.

Q. Can I access DynamoDB Streams using VPC endpoints for DynamoDB?

Currently, you cannot access Amazon DynamoDB Streams using VPC endpoints for Amazon DynamoDB.

Q. I currently use an internet gateway and a NAT gateway to send requests to DynamoDB. Do I need to change my application code when I use a VPC endpoint?

Your application code does not need to change. Simply create a VPC endpoint, update your route table to point Amazon DynamoDB traffic at the DynamoDB VPC endpoint, and access DynamoDB directly. You can continue using the same code and same DNS names to access DynamoDB.

Q. Can I use one VPC endpoint for both DynamoDB and another AWS service?

No, each VPC endpoint supports one service. You can create one for Amazon DynamoDB and another for the other AWS service and use both of them in a route table.

Q. Can I have multiple VPC endpoints in a single VPC?

Yes, you can have multiple VPC endpoints in a single VPC. For example, you can have one VPC endpoint for Amazon S3 and one VPC endpoint for Amazon DynamoDB.

Q. Can I have multiple VPC endpoints for DynamoDB in a single VPC?

Yes, you can have multiple VPC endpoints for Amazon DynamoDB in a single VPC. Individual VPC endpoints can have different VPC endpoint policies. For example, you could have a VPC endpoint that is read-only and one that is read/write. However, a single route table in a VPC can only be associated with a single VPC endpoint for DynamoDB, because that route table will route all traffic to DynamoDB through the specified VPC endpoint.

Q. What are the differences between VPC endpoints for S3 and VCP endpoints for DynamoDB?

The main difference is that these two VPC endpoints support different services – Amazon S3 and Amazon DynamoDB.

Q. What IP address will I see in AWS CloudTrail logs for traffic coming from the VPC endpoint for DynamoDB?

AWS CloudTrail logs for Amazon DynamoDB will contain the private IP address of the Amazon EC2 instance in the VPC, and the VPC endpoint identifier (for example, sourceIpAddress=10.89.76.54, VpcEndpointId=vpce-12345678).

Q. How can I manage VPC endpoints using the AWS Command Line Interface (CLI)?

You can use the following CLI commands to manage VPC endpoints: create-vpc-endpoint, modify-vpc-endpoint, describe-vpc-endpoint, delete-vpc-endpoint, and describe-vpc-endpoint-services. You should specify the Amazon DynamoDB service name specific to your VPC and DynamoDB Region (for example, com.amazon.us.east-1.DynamoDB). For more information, see [create-vpc-endpoint](https://docs.aws.amazon.com/cli/latest/reference/ec2/create-vpc-endpoint.html).

Q. Do VPC endpoints for DynamoDB require customers to know and manage the public IP address ranges of DynamoDB?

No, customers don’t need to know or manage the public IP address ranges for Amazon DynamoDB in order to use this feature. A prefix list will be provided to use in route tables and security groups. AWS maintains the address ranges in the list. The prefix list name is: com.amazonaws.<region>.DynamoDB (for example, com.amazonaws.us-east-1.DynamoDB).

Q. Can I use AWS IAM policies on a VPC endpoint for DynamoDB?

Yes. You can attach an AWS IAM policy to your VPC endpoint and this policy will apply to all traffic through this endpoint. For example, a VPC endpoint using this policy allows only describe\* API calls:  
{  
    "Statement":  [  
       {  
            "Sid": "Stmt1415116195105",  
            "Action": "dynamodb:describe\*",  
            "Effect": "Allow",  
            "Resource": "arn:aws:dynamodb:region:account-id:table/table-name",  
            "Principal": "\*"  
        }  
    ]  
}

Q. Can I limit access to my DynamoDB table from a VPC endpoint?

Yes, you can create an AWS IAM policy to restrict an IAM user, group, or role to a particular VPC endpoint for DynamoDB tables.

This can be done by setting the IAM policy’s Resource element to a DynamoDB table and the Condition element’s key to aws:sourceVpce. For more details, see the [IAM JSON Policy Elements Reference](http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html#Condition).

For example, the following IAM policy restricts access to DynamoDB tables unless sourceVpce matches “vpce-111bbb22”

{  
    "Statement":  [  
       {  
            "Sid": "Stmt1415116195105",  
            "Action": "dynamodb:\*",  
            "Effect": "Deny",  
            "Resource": "arn:aws:dynamodb:region:account-id:\*",  
            "Condition": { "StringNotEquals" : { "aws:sourceVpce": "vpce-111bbb22" } }  
        }  
    ]  
}

Q. Do VPC endpoints for DynamoDB support IAM policy conditions for fine-grained access control?

Yes. VPC endpoints for DynamoDB support all fine-grained access control access keys. You can use AWS IAM policy conditions for fine-grained access control to control access to individual data items and attributes. For more information about fine-grained access control, see [Using IAM Policy Conditions for Fine-Grained Access Control](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/FGAC_DDB.html).

Q. Can I use the AWS Policy Generator to create VPC endpoint policies for DynamoDB?

Yes, you can use the [AWS Policy Generator](https://awspolicygen.s3.amazonaws.com/policygen.html) to create VPC endpoint policies.  
  
Q. Does DynamoDB support resource-based policies similar to Amazon S3 bucket policies?

No, Amazon DynamoDB does not support resource-based policies pertaining to individual tables, items, and the like.