DynamoDB FAQs

* [What is DynamoDB](#what-is-dynamodb)
* [Getting Started](#getting_started_anchor)
* [Data Model & APIs](#Data_Models_and_APIs)
* [Scalability, Availability, and Durability](#Scalability,_Availability_&_Durability)
* [Secondary Indexes](#Local_Secondary_Indexes)
* [Security and Control](#Security_and_Control)
* [Pricing](#Pricing)
* [Reserved Capacity](#Reserved_Capacity)
* [DynamoDB Cross-region Replication](#DynamoDB_Cross-region_Replication)
* [DynamoDB Triggers](#DynamoDB_Triggers)
* [DynamoDB Streams](#DynamoDB_Streams)
* [DynamoDB Storage Backend for Titan](#DynamoDB_Storage_Backend_for_Titan)
* [DynamoDB CloudWatch Metrics](#DynamoDB_CloudWatch_Metrics)

What is DynamoDB [(back to top)](#FAQs)

Q: What is Amazon DynamoDB?

Amazon DynamoDB is a fully managed [NoSQL database](https://aws.amazon.com/nosql/) service that provides fast and predictable performance with seamless scalability. Amazon DynamoDB enables customers to offload the administrative burdens of operating and scaling distributed databases to AWS, so they don’t have to worry about hardware provisioning, setup and configuration, 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 the database software and the provisioning of hardware needed to run it. Customers can deploy a non-relational database in a matter of minutes. DynamoDB automatically partitions and re-partitions your data and provisions additional server capacity as your table size grows or you increase your provisioned throughput. In addition, Amazon 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. Amazon 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 to sets, lists, or maps. [View our documentation for more information on atomic updates.](http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/WorkingWithDDItems.html)

Q: Why is Amazon DynamoDB built on Solid State Drives?

Amazon DynamoDB runs exclusively on Solid State Drives (SSDs). SSDs help us achieve our design goals of predictable low-latency response times for storing and accessing data at any scale. The high I/O 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: DynamoDB’s storage cost seems high. Is this a cost-effective service for my use case?

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/sec and writes/sec) per GB stored. Amazon 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 the SSD-based DynamoDB service will usually be lower than the cost of using a typical spinning media-based relational or non-relational database. If you have a use case that involves storing a large amount of data that you rarely access, then DynamoDB may not be right for you. We recommend that you use S3 for such use cases.

It should also be noted that the storage cost reflects the cost of storing multiple copies of each data item across multiple facilities within an AWS Region.

Q: Is DynamoDB only for high-scale applications?

No. DynamoDB offers seamless scaling so you can start small and scale up and down in line with your requirements. If you need fast, predictable performance at any scale then DynamoDB may be the right choice for you.

Getting Started [(back to top)](#FAQs)

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, it does not have all the functionality of a relational database. It does not support complex relational queries (e.g. joins) or complex transactions. 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 [(back to top)](#FAQs)

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.

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 & Durability [(back to top)](#FAQs)

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 throughput you have provisioned for your table using UpdateTable API or in 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 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.

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.

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">best practices documentation on uniform workloads).

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" \t "_blank) 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. You need to explicitly specify the provisioned throughput for the table and each associated GSI at creation time. You can use the [Create Table Wizard](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStartedCreateTables.html) of the [DynamoDB Console](https://console.aws.amazon.com/dynamodb/" \t "_blank) which can assist you in distributing your total throughput among your tables and indexes.

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. 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/" \t "_blank) or the [UpdateTable API](http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_UpdateTable.html" \t "_blank).

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" \l "DDB-Query-request-ConsistentRead" \t "_blank)* 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?

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" \t "_blank) 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: Can I auto-scale my tables and indexes in DynamoDB?

While this is not a native function, there are recommended third party libraries located in the [Developer Resources](http://aws.amazon.com/dynamodb/developer-resources/) section of the DynamoDB web page.

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 [(back to top)](#FAQs)

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 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 [(back to top)](#FAQs)

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 will support 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 [(back to top)](#FAQs)

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 that you provision for your table, 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 or the UpdateTable API.

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 lets you specify the request throughput you want your table to be able to achieve. 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.

Amazon DynamoDB lets you specify your throughput needs in terms of units of read capacity and write capacity for your table. During creation of a table, you specify your required read and write capacity needs and Amazon DynamoDB automatically partitions and reserves the appropriate amount of resources to meet your throughput requirements. To decide on the required read and write throughput values, consider the number of read and write data plane API calls you expect to perform per second. If at any point you anticipate traffic growth that may exceed your provisioned throughput, you can simply update your provisioned throughput values via the AWS Management Console or Amazon DynamoDB APIs. You can also reduce the provisioned throughput value for a table as demand decreases. Amazon DynamoDB will remain available while scaling it throughput level up or down.

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.

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 it four times per day. A day is defined according to the GMT time zone. For example, if you decrease the provisioned throughput for your table four times on December 12th, you won’t be able to decrease the provisioned throughput for that table again until 12:01am GMT on December 13th.

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 [(back to top)](#FAQs)

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. When you create or update a table, you specify how much read or write capacity you want it to have. 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.

DynamoDB Cross-region Replication [(back to top)](#FAQs)

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 replication management app that you launch from the AWS CloudFormation console and opening this template: https://dynamodb-cross-region.s3.amazonaws.com/dynamodb-replication-coordinator.template. Once launched, you can use the application to set up a replication group and replica to the group.

Immediately after the replication is set up, the cross-region replication application performs a one-time copy of the master table to the replica tables (called bootstrapping) and then keeps the tables in sync as the items in the master table change.

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 by a new application which you can launch using the provided [AWS CloudFormation Stack](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.CrossRegionRepl.Walkthrough.Step2.html). While there is no additional charge for the cross-region replication application, you pay the usual prices for the following resources used by the application. 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 application. The cost of the instances will depend on the instance type you choose and the region hosting the instances.
* The SQS queue that queues control commands from the application.

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.

DynamoDB Triggers [(back to top)](#FAQs)

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.

DynamoDB Streams [(back to top)](#FAQs)

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.

DynamoDB Storage Backend for Titan [(back to top)](#FAQs)

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.

DynamoDB CloudWatch Metrics [(back to top)](#FAQs)

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.