Exhibit 16

Frequently Asked Questions About Amazon DynamoDB

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.

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 for more information on which expressions work on partition and partition-sort keys.

Q: Does Amazon DynamoDB support increment or decrement operations?

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

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

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

While Amazon DynamoDB tackles the core problems of database scalability, management, performance, and reliability, the datamodel, just like any NoSQL, must be designed specifically for the access patterns required by the application. In other words, running adhoc queries on DynamoDB can be inefficient. Refer to the design guidance that shows how to effectively migrate from any Relational database to DynamoDB. If your workload requires this functionality, or you are looking for compatibility with an existing relational engine, you may wish to run a relational engine 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 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 page. If you have trouble installing or using one of our SDKs, please let us know by posting to the relevant AWS Forum.

Data models and APIs

Q: What is the Data Model?

The data model for Amazon DynamoDB is as follows:

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

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

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

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

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

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

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

Q: What are the APIs?

- CreateTable – Creates a table and specifies the primary index used for data access.
- UpdateTable – Updates the provisioned throughput values for the given table.
- DeleteTable – Deletes a table.
- DescribeTable – Returns table size, status, and index information.
- ListTables – Returns a list of all tables associated with the current account and endpoint.
- PutItem – Creates a new item, or replaces an old item with a new item (including all the attributes). If an item already exists in the specified table with the same primary key, the new item completely replaces the existing item. You can also use conditional operators to replace an item only if its attribute values match certain