AWS - DynamoDB

https://aws.amazon.com/dynamodb/

https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

AUTO-SCALING in DYNAMODB

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

Eventual Vs Strong Consistency

- Difference is time in which data synchronization happens across data centers
- **Eventual Consistent Reads** :
    - Few seconds delay for Reads
- **Strong Consistent Reads** :
    - Few milli-seconds delay for Reads

Read/Write Units Calculation

- READS
    - Rounded to 50 no's for Reads
- WRITES
    - Rounded to 10 no's for Writes
    - Eg:
    - **WRITES**
        - If 11 txns per second
        - Then 20 units per second.

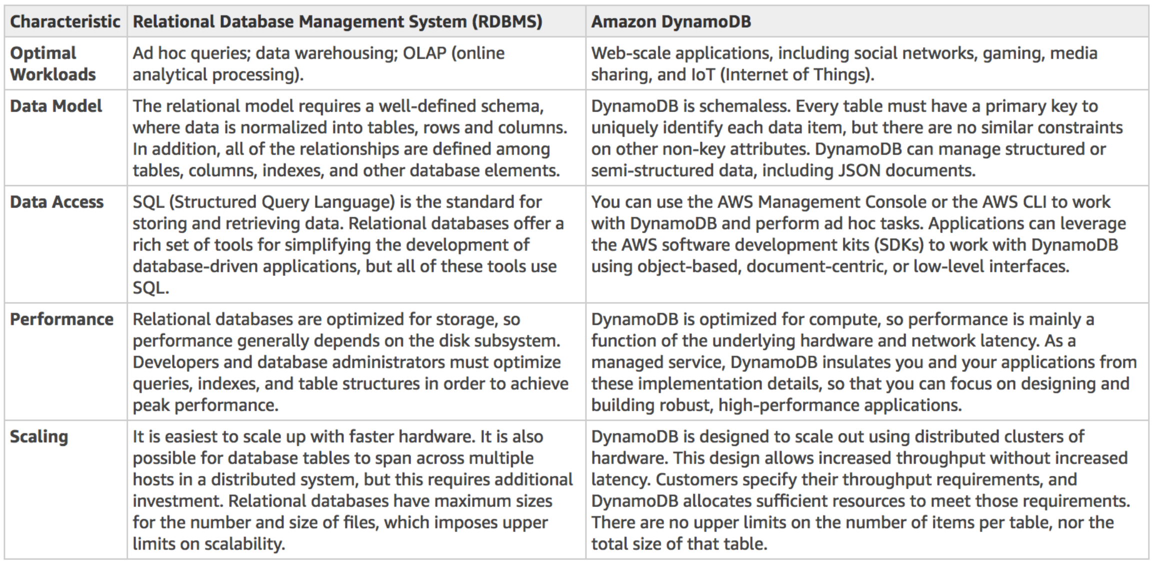
DynamoDB

DynamoDB supports two different kinds of primary keys:

Partition key – A simple primary key, composed of one attribute known as the partition key. DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored.

Partition key and sort key – Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key. DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored.

Take note that the partition key of an item is also known as its hash attribute. The term hash attribute derives from the use of an internal hash function in DynamoDB that evenly distributes data items across partitions, based on their partition key values. Hence, the correct answer is to ***use one table every week, with a composite primary key which is the sensor ID as the partition key and the timestamp as the sort key.***



Characteristic	Relational Database Management System (RDBMS)	Amazon DynamoDB
Optimal Workloads	Ad hoc queries; data warehousing; OLAP (online analytical processing).	Web-scale applications, including social networks, gaming, media sharing, and IoT (Internet of Things).
Data Model	The relational model requires a well-defined schema, where data is normalized into tables, rows and columns. In addition, all of the relationships are defined among tables, columns, indexes, and other database elements.	DynamoDB is schemaless. Every table must have a primary key to uniquely identify each data item, but there are no similar constraints on other non-key attributes. DynamoDB can manage structured or semi-structured data, including JSON documents.
Data Access	SQL (Structured Query Language) is the standard for storing and retrieving data. Relational databases offer a rich set of tools for simplifying the development of database-driven applications, but all of these tools use SQL.	You can use the AWS Management Console or the AWS CLI to work with DynamoDB and perform ad hoc tasks. Applications can leverage the AWS software development kits (SDKs) to work with DynamoDB using object-based, document-centric, or low-level interfaces.
Performance	Relational databases are optimized for storage, so performance generally depends on the disk subsystem. Developers and database administrators must optimize queries, indexes, and table structures in order to achieve peak performance.	DynamoDB is optimized for compute, so performance is mainly a function of the underlying hardware and network latency. As a managed service, DynamoDB insulates you and your applications from these implementation details, so that you can focus on designing and building robust, high-performance applications.
Scaling	It is easiest to scale up with faster hardware. It is also possible for database tables to span across multiple hosts in a distributed system, but this requires additional investment. Relational databases have maximum sizes for the number and size of files, which imposes upper limits on scalability.	DynamoDB is designed to scale out using distributed clusters of hardware. This design allows increased throughput without increased latency. Customers specify their throughput requirements, and DynamoDB allocates sufficient resources to meet those requirements. There are no upper limits on the number of items per table, nor the total size of that table.

AWS - DynamoDB autoscaling

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

AWS - DynamoDB Keys

- The partition key portion of a table's primary key determines the logical partitions in which a table's data is stored.
- This in turn affects the underlying physical partitions.
- Provisioned I/O capacity for the table is divided evenly among these physical partitions.
- Therefore a partition key design that doesn't distribute I/O requests evenly can create "hot" partitions that result in throttling and use your provisioned I/O capacity inefficiently.

- The optimal usage of a table's provisioned throughput depends not only on the workload patterns of individual items, but also on the partition-key design.
- This doesn't mean that you must access all partition key values to achieve an efficient throughput level, or even that the percentage of accessed partition key values must be high.

- It does mean that the more distinct partition key values that your workload accesses, the more those requests will be spread across the partitioned space.
- In general, you will use your provisioned throughput more efficiently as the ratio of partition key values accessed to the total number of partition key values increases.
- **Partition Key with HIGH-CARDINALITY** = Distinct values for each item
- **Partition Key with LOW-CARDINATLITY** = Few number of Distinct values for each item.

References:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-uniform-load.html

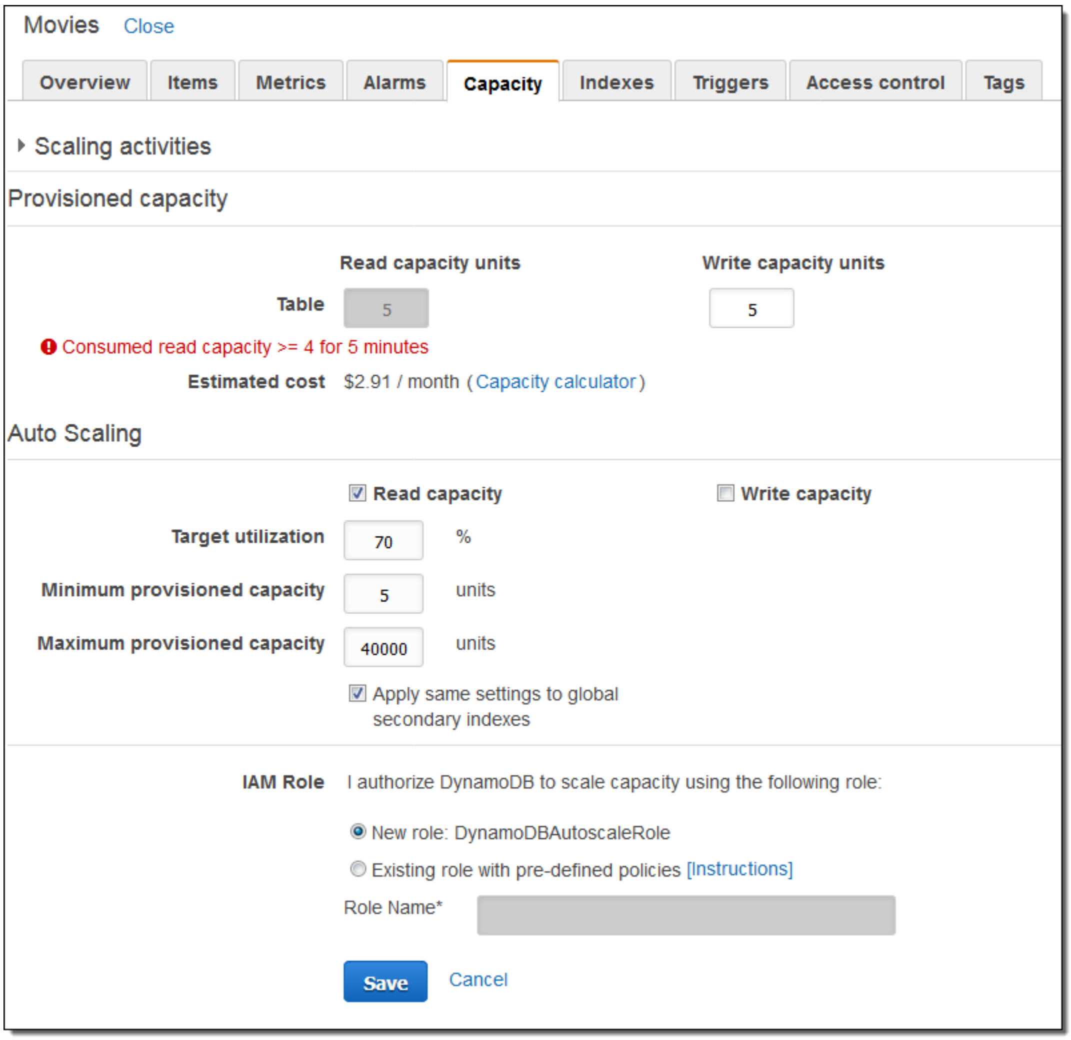
https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

Check out this Amazon DynamoDB Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/

AWS - DynamoDB Accelerator (DAX)

- **Amazon DynamoDB Accelerator (DAX)** is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement – from milliseconds to microseconds – even at millions of requests per second.
- DAX does all the heavy lifting required to add in-memory acceleration to your DynamoDB tables, without requiring developers to manage cache invalidation, data population, or cluster management.


Movies Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Triggers | Access control | Tags

Scaling activities

Provisioned capacity

	Read capacity units	Write capacity units
Table	5	5

Consumed read capacity >= 4 for 5 minutes

Estimated cost $2.91 / month (Capacity calculator)

Auto Scaling

[✓] Read capacity [ ] Write capacity

Target utilization	70	%
Minimum provisioned capacity	5	units
Maximum provisioned capacity	40000	units

[✓] Apply same settings to global secondary indexes

IAM Role I authorize DynamoDB to scale capacity using the following role:

(●) New role: DynamoDBAutoscaleRole

( ) Existing role with pre-defined policies [Instructions]

Role Name*

Save Cancel

- Amazon API Gateway lets you create an API that acts as a "front door" for applications to access data, business logic, or functionality from your back-end services, such as code running on AWS Lambda. Amazon API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. Amazon API Gateway has no minimum fees or startup costs.
- AWS Lambda scales your functions automatically on your behalf. Every time an event notification is received for your function, AWS Lambda quickly locates free capacity within its compute fleet and runs your code. Since your code is stateless, AWS Lambda can start as many copies of your function as needed without lengthy deployment and configuration delays.

https://aws.amazon.com/lambda/faqs/

https://aws.amazon.com/api-gateway/faqs/

https://aws.amazon.com/dynamodb/dax/

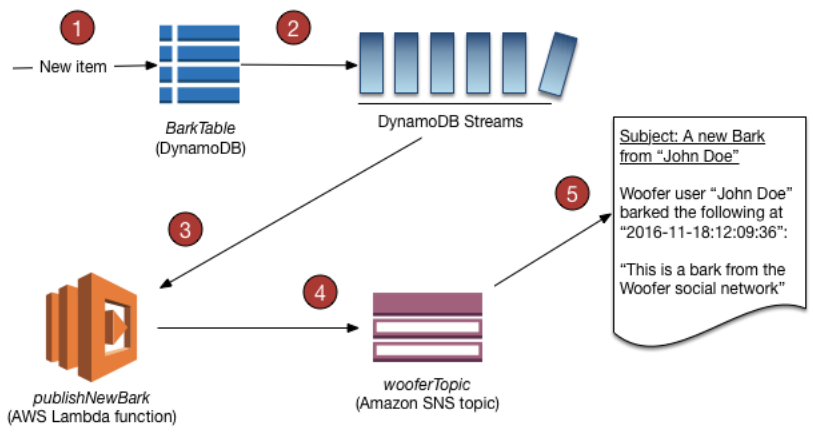
AWS - DynamoDB Streams

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

With triggers, you can build applications that react to data modifications in DynamoDB tables.

If you enable DynamoDB Streams on a table, you can associate the stream ARN with a Lambda function that you write. Immediately after an item in the table is modified, a new record appears in the table's stream.

AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.



Subject: A new Bark from "John Doe"

Woofer user "John Doe" barked the following at "2016-11-18:12:09:36":

"This is a bark from the Woofer social network"

You can create a Lambda function which can perform a specific action that you specify, such as sending a notification or initiating a workflow.

For instance, you can set up a Lambda function to simply copy each stream record to persistent storage, such as EFS or S3, to create a permanent audit trail of write activity in your table.

Suppose you have a mobile gaming app that writes to a TutorialsDojoCourses table.

Whenever the TopCourse attribute of the TutorialsDojoScores table is updated, a corresponding stream record is written to the table's stream.

This event could then trigger a Lambda function that posts a congratulatory message on a social media network. (The function would simply ignore any stream records that are not updates to TutorialsDojoCourses or that do not modify the TopCourse attribute.)

References:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.html

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

Check out this Amazon DynamoDB cheat sheet:

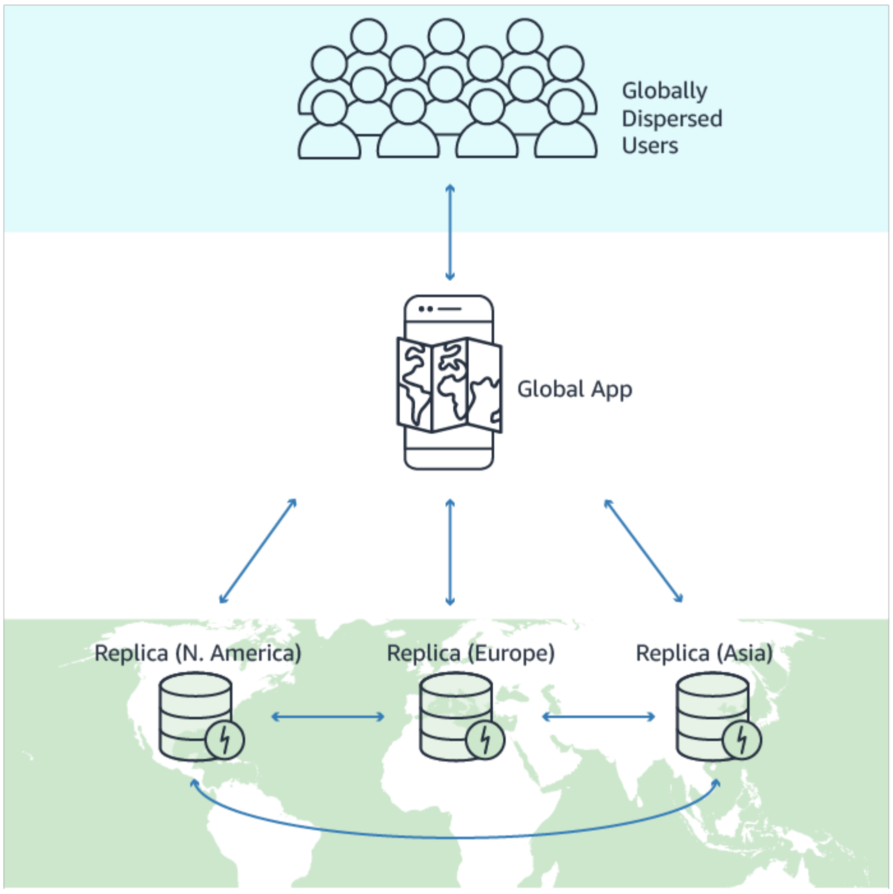
https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/

AWS - DynamoDB Global Tables

Global Tables builds upon DynamoDB's global footprint to provide you with a fully managed, multi-region, and multi-master database that provides fast, local, read and write performance for massively scaled, global applications. Global Tables replicates your Amazon DynamoDB tables automatically across your choice of AWS regions.

Global Tables eliminates the difficult work of replicating data between regions and resolving update conflicts, enabling you to focus on your application's business logic. In addition, Global Tables enables your applications to stay highly available even in the unlikely event of isolation or degradation of an entire region.

Hence, the option that says: **Create a Global DynamoDB table with replica tables across several AWS regions that you prefer. In each local region, store the individual transactions to a DynamoDB replica table in the same region. Any changes made in one of the replica tables will automatically be replicated across all other tables** is the correct answer.


Globally Dispersed Users
Global App
Replica (N. America)
Replica (Europe)
Replica (Asia)