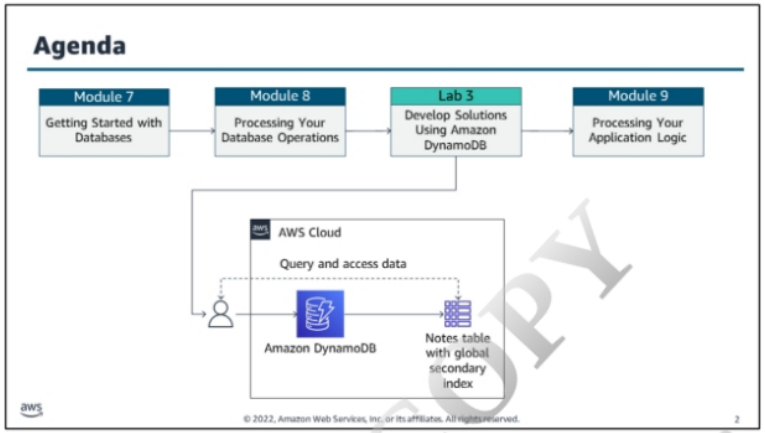
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification

Module 7: Getting Started with Databases



Module objectives

By the end of this module, you will be able to:

- Describe the key components of DynamoDB
- Explore multiple ways to connect to DynamoDB
- Define SDK dependencies and settings for your code
- Work with request and response objects

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification

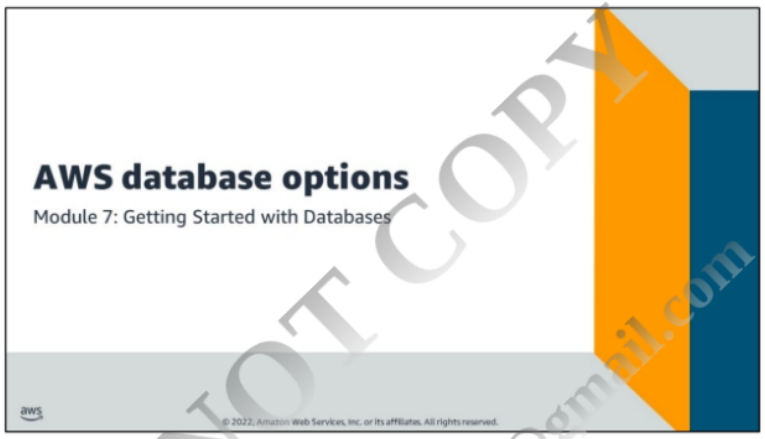
Module 7: Getting Started with Databases

Day 1 recap

- Your application
- Developer tools (IDEs, SDKs, APIs)
- Permissions
- Storing and hosting your application

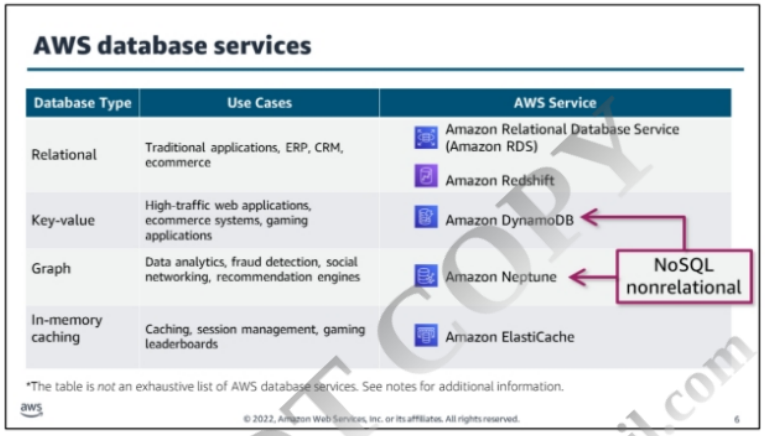
DO NOT COPY
sameersheik68@gmail.com

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



AWS provides a wide variety of database services to choose from. The AWS services are categorized as relational and nonrelational database services. Depending on the application context, type of data, and business needs, you have the following options to store application-related data:

Relational AWS database services

AWS offers the following relational database services:

- **Amazon Relational Database Service (Amazon RDS)**: Provides relational database services in the cloud, with support for the following database engines:
  - Amazon Aurora
  - PostgreSQL
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
- **Amazon Redshift**: Fast, fully managed data warehouse

Nonrelational (NoSQL) databases

Amazon DynamoDB, Amazon ElastiCache, and Amazon Neptune services are considered NoSQL databases (nonrelational databases), which use dynamic schemas for unstructured data.

- **Amazon DynamoDB** – NoSQL database that supports both document and key-value store models.
- **Amazon ElastiCache** – In-memory data cache that supports a fully managed Redis or Memcached engine.
- **Amazon Neptune** – Fast, reliable, fully managed graph database service that streamlines building and running applications that work with highly connected datasets.

***Additional AWS database services (*not *included* in the table):**

- **Amazon DocumentDB (with MongoDB compatibility)** – Document-based database service that is purpose-built for JSON data management at scale.

  Use cases: Content management, catalogs, or user profiles

- **Amazon Keyspaces (for Apache Cassandra):** A wide-column database solution that is a scalable, highly available, and managed Apache Cassandra-compatible database service.

  Use cases: High-scale industrial applications for equipment maintenance, fleet management, and route optimization

- **Amazon Timestream:** A fast, scalable, and serverless time series database service for Internet of Things (IoT) and operational applications that make streamline storing and analyzing trillions of events per day.

  Use cases: IoT applications, DevOps, and industrial telemetry

- **Amazon Quantum Ledger Database (Amazon QLDB):** A fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority.

  Use cases: Focus on systems of record, supply chain, registrations, and banking transactions

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

Comparing relational and nonrelational databases, 1 of 2

Aspect	Relational	NoSQL (Nonrelational)
Data Storage	Rows and columns	Key-value, document, wide-column, graph
Schemas	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scalability	Vertical	Horizontal
Transactions	Supported	Support varies
Consistency	Strong	Eventual and strong

Relational databases

Relational databases have the following characteristics:

- Data is stored in tables that are related to each other through a primary key/foreign key relationship. The database supports complex queries and joins to retrieve a combination of data from various tables.
- The schema of a relational database is defined in the beginning. Changes to the schema usually require a migration of data from the earlier schema to the new schema.
- Data in SQL databases is queried using structured query language (SQL), which allows for complex queries.
- Relational databases support vertical scaling, which means that a single server must be made more powerful.
- Relational databases support ACID transactions: atomicity, consistency, isolation, and durability.
- Relational databases support strong data consistency automatically because of the ACID properties of transactions.

Nonrelational databases

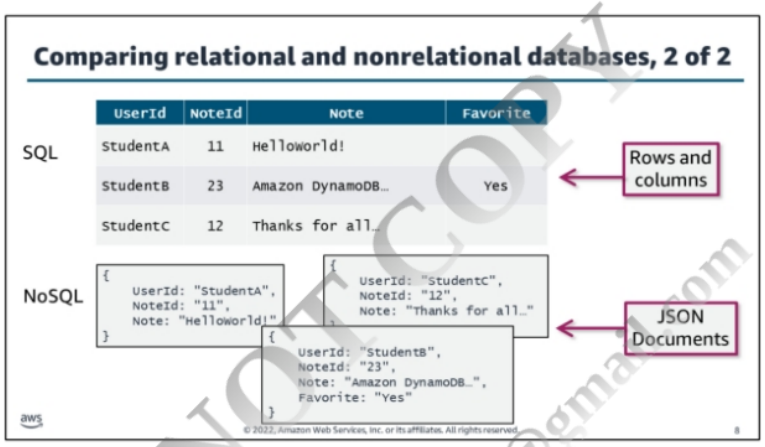
NoSQL, nonrelational databases have the following characteristics:

- NoSQL databases can support wide-column stores, document stores, key-value stores, and graph stores. The type of data stored varies by the NoSQL database.
- NoSQL databases do not have a fixed schema. Different records can have different attributes.

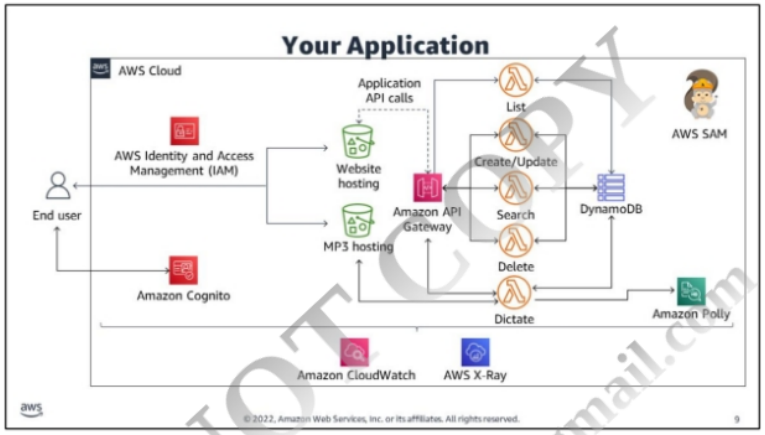
- Data in NoSQL databases is queried by focusing on collections of documents.
- NoSQL databases support horizontal scaling. With this approach, you can partition and spread data across multiple, less costly servers instead of purchasing more powerful servers. For example, if a table stores information about 100,000 users, data can be partitioned to store a subset of 10,000 users on each server.
- NoSQL databases generally deliver high performance with eventual consistency. Depending on the database, you can set it to strong consistency, if needed, for your usage scenario. An example of strong consistency is read immediately after a write returns the latest data. **Note:** DynamoDB supports ACID through DynamoDB transactions.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



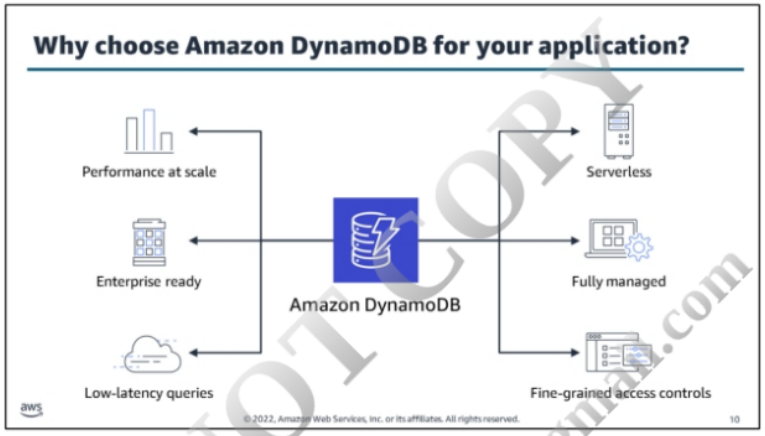
DO NOT COPY sameersheik68@gmail.com



Your application users will be reading, adding, updating, and deleting their notes multiple times a day. Amazon DynamoDB offers a viable solution to store these interactions.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



Why choose DynamoDB for your application? DynamoDB provides these benefits:

Performance at scale
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use this scalability to meet your application's fluctuating capacity needs.

Serverless
DynamoDB offers a serverless solution that does not require you to install or maintain any software. Amazon DynamoDB is also enterprise-ready for vital workloads.

Enterprise ready
Amazon DynamoDB is a fast and flexible nonrelational database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database that supports both document and key-value store models.

Fully managed
DynamoDB is a fully managed, nonrelational database service. You can create a database table, set your target use for automatic scaling, and let the service handle the rest. You no longer need to worry about database management tasks as you scale. Examples of management tasks include the following:

DO NOT COPY
sameersheik68@gmail.com

- Hardware or software provisioning
- Setup and configuration
- Software patching
- Operating a distributed database cluster
- Partitioning data over multiple instances

DynamoDB also provides point-in-time recovery, backup, and restore for all your tables. These features help you meet your corporate and regulatory archival requirements.

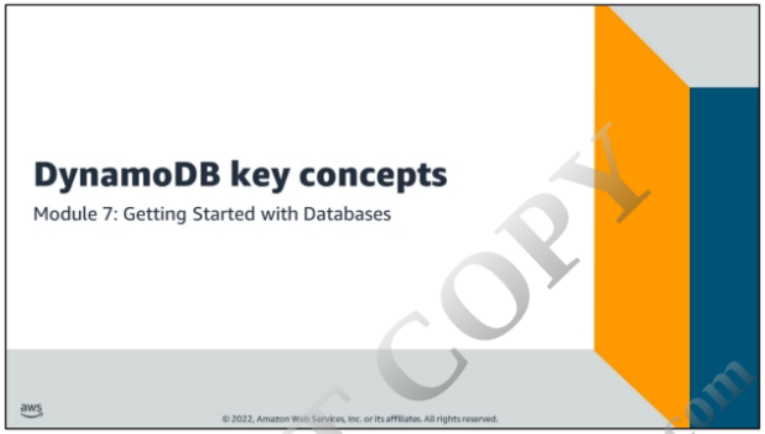
Low-latency queries
Average service-side latencies are typically single-digit milliseconds. As your data volumes grow and application performance demands increase, DynamoDB uses automatic partitioning and solid state drives (SSD) technologies. These technologies help you to meet your throughput requirements and deliver low latencies at any scale.

Fine-grained access control
DynamoDB integrates with AWS Identity and Access Management (IAM) for fine-grained access control of users in your organization. You can assign unique security credentials to each user and control each user's access to services and resources.

Flexible
DynamoDB supports storing, querying, and updating documents. By using the AWS SDK, you can write applications that store JSON documents directly into Amazon DynamoDB tables. This capability reduces the amount of new code to be written to insert, update, and retrieve JSON documents. You can also perform powerful database operations, such as nested JSON queries, by using a few lines of code.

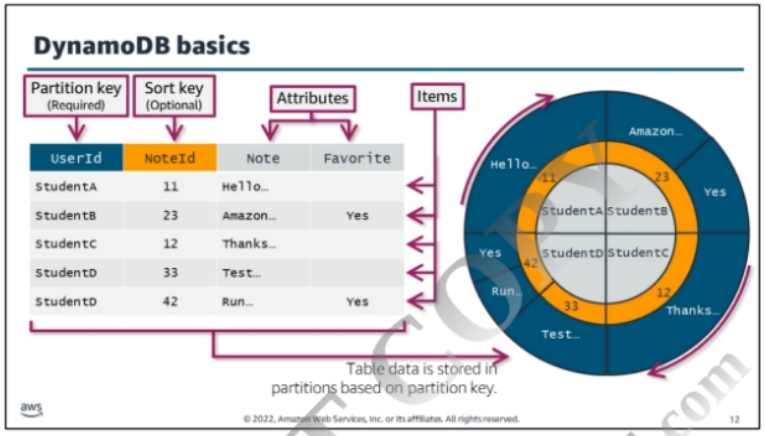
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



DO NOT COPY
sameersheik68@gmail.com

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon DynamoDB, stores data in *tables*. A table contains *items* with *attributes*. You can think of items as rows or tuples, in a relational database, and attributes as columns.

DynamoDB stores data in partitions and divides a table's items into multiple partitions based on the *partition key* value. A *partition* is an allocation of storage for a table, backed by SSDs and automatically replicated across multiple Availability Zones within an AWS Region. Partition management is handled entirely by DynamoDB. The partition key of an item is also known as its *hash attribute*.

A *sort key* can be defined to store all of the items with the same partition key value physically close together. The items can then be ordered by sort key value in the partition. It represents a one-to-many relationship based on the partition key and enables querying on the sort key attribute. The sort key of an item is also known as its *range attribute*.

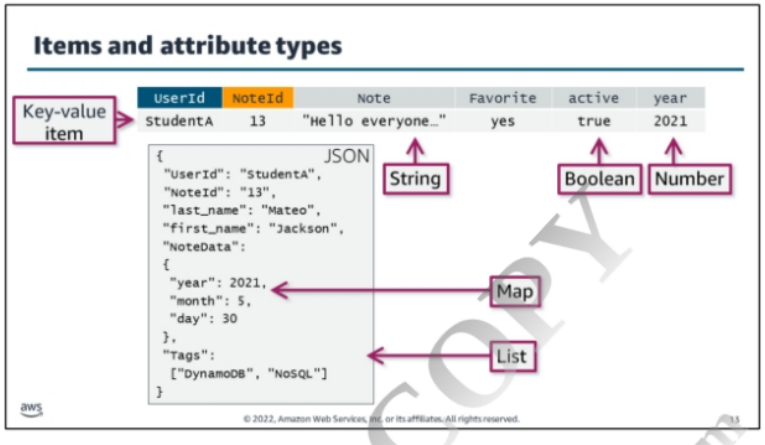
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

For information, see the following in the *Amazon DynamoDB Developer Guide*:

- "Partitions and Data Distribution" (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html)
- "Working with Tables and Data in DynamoDB" (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithTables.html)
- "DynamoDB data model" (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.Partitions.html)

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



An item is a collection of attributes. Items are analogous to rows, and attributes are analogous to columns in a relational database table.

Each attribute has a name, data type, and value. An item can have any number of attributes. Unlike a relational database, DynamoDB is not constrained by a predefined schema. Items in a table can have different types of attributes.

Data types

Attributes can have one of the following data types:

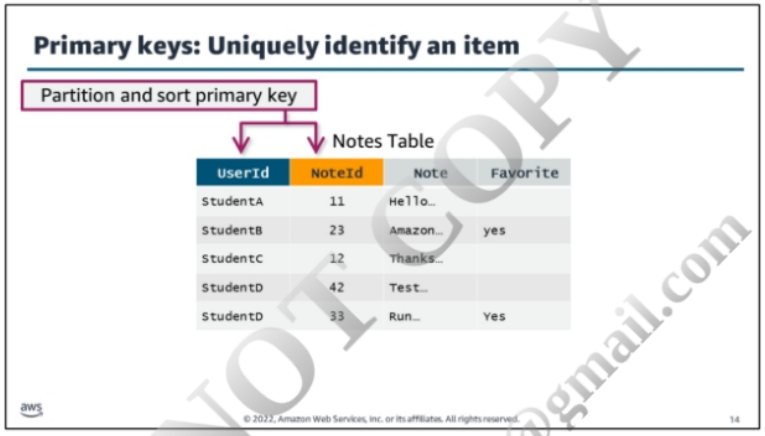
- **Scalar types** – Number, String, Binary, Boolean, and Null
- **Multi-valued types** – String Set, Number Set, and Binary Set
- **Document types** – List and Map

A document type can represent a complex structure with nested attributes, similar to the structure you would find in a JSON-formatted document. Maps are ideal for storing JSON-formatted documents in DynamoDB. You can save objects as JSON documents in a DynamoDB attribute.

The size of an item is the sum of the lengths of its attribute names and values. An item can be a maximum of 400 KB in size.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

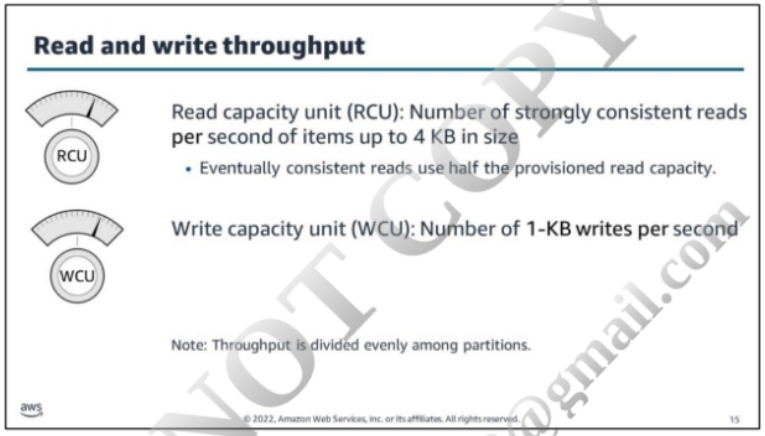


A table has a primary key that uniquely identifies each item in the table.
Types of primary keys:

- ***Partition primary key*** – The primary key consists of a single attribute, the partition key. DynamoDB builds an unordered index on this primary key attribute. Each item in the table is uniquely identified by its partition key value.
- ***Partition and sort primary key*** – The primary key is made of two attributes. The first attribute is the partition key attribute and the second attribute is the sort key attribute. DynamoDB builds an unordered index on the partition key attribute and a sorted index on the sort key attribute. Each item in the table is uniquely identified by the combination of its partition key and sort key values.

The example is the Notes table for which the primary key is composed of a partition key and a sort key. The UserId attribute is the partition key, and the NoteId attribute is the sort key. Therefore, it is a *partition and sort primary key*. For each UserId, there may be multiple notes. The combination of UserId and NoteId uniquely identifies items in the table. By implementing this design, you can query the table for all readings related to a particular user.

Remember that you can use DynamoDB as a key-value store and document store. In the examples shown, the primary key value (partition key and sort key, if present) is the key, and the remaining attributes constitute the values corresponding to the key.



To enable high throughput and low latency responses, DynamoDB requires that you specify read and write throughput values when you create a table. This ensures predictable, low-latency response times. DynamoDB reserves the necessary resources to handle your throughput requirements. If you expect spikes in your workload, you can set higher throughput values by updating the table.

Throughput is specified in terms of *read capacity units* (RCU) and *write capacity units* (WCU). A read capacity unit is the number of strongly consistent reads per second of items up to 4 KB in size. If you perform eventually consistent reads, you use half the read capacity units provisioned. In other words, for eventually consistent reads, one read capacity unit is *two* reads per second for items up to 4 KB. A write capacity unit is the number of 1-KB writes per second.

Read and write capacity unit values can be set independent of each of other. For example, you might increase only the read capacity units if you expect a spike in reads from the table. DynamoDB returns an error if the provisioned throughput has been exceeded (`ProvisionedThroughputExceeded` exception).

DynamoDB divides the throughput evenly among partitions. Throughput per partition is the total provisioned throughput divided by the number of partitions.

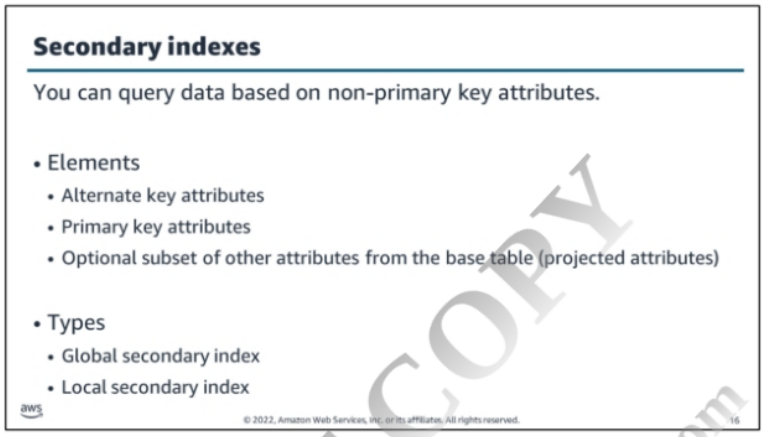
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

Set your required throughput value by setting the `ProvisionedThroughput` parameter when you create or update a table.

For information, see "Working with Tables and Data in DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithTables.html).

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



Some queries can't address the more complex data access patterns. To address these types of requests, the Query operation would have to scan the entire table. However, by specifying secondary indexes on nonkey attributes, the query wouldn't consume a large amount of provisioned read throughput.

If you want to perform queries on attributes that are not part of the table's primary key, you can create a *secondary index*. Use a secondary index to query the data in the table using an alternate key, in addition to queries against the primary key.

In addition to the alternate key attributes (partition key and sort key), a secondary index contains a subset of the other table attributes. When you create an index, you specify which attributes will be copied, or projected, from the base table to the index. At a minimum, DynamoDB projects the key attributes from the base table into the index.

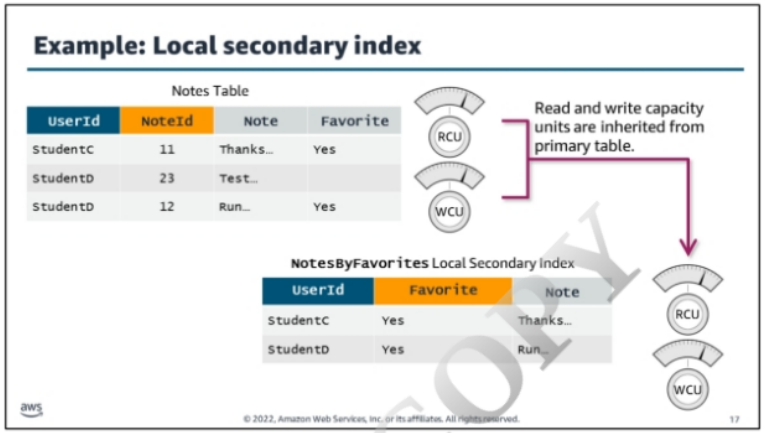
Two types of secondary indexes are available: global secondary index and local secondary index. Each table in DynamoDB can have up to 20 global secondary indexes (default quota) and 5 local secondary indexes.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

To request a service limit increase, see "Service, Account, and Table Quotas in Amazon DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html).

DynamoDB automatically creates indexes based on the primary key of a table and automatically updates all indexes whenever a table changes.



A local secondary index is considered to be *local* because the index is located on the same table partition as the items that have a given partition key value. You can query over data in only a single partition as specified by the partition key value in the query.

The following are characteristics of a local secondary index:

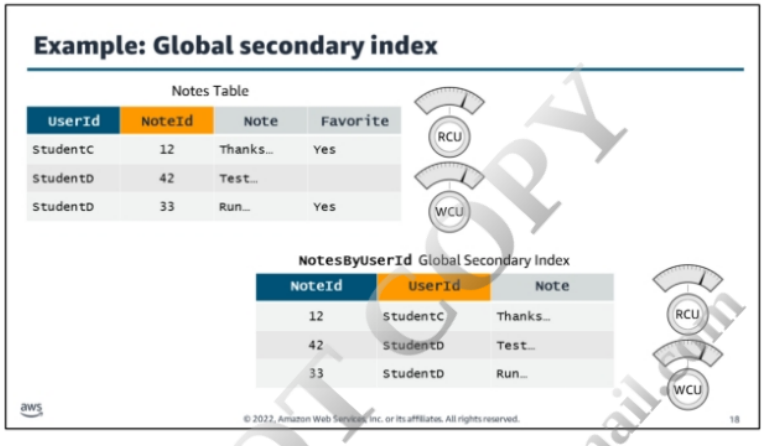
- The partition key is the same as the table's partition key. The sort key can be any scalar attribute.
- It can be created only when a table is created.
- It cannot be deleted.
- It supports eventual consistency and strong consistency.
- It does not have its own provisioned throughput. Instead, it uses the table's read and write capacity units.

- Queries can return attributes that are not projected into the index.
- All the items with a particular partition key in the table and the items in the corresponding local secondary index (together known as an *item collection*) are stored on the same partition. The total size of an item collection cannot exceed 10 GB.

For more information, see "Improving Data Access with Secondary Indexes" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

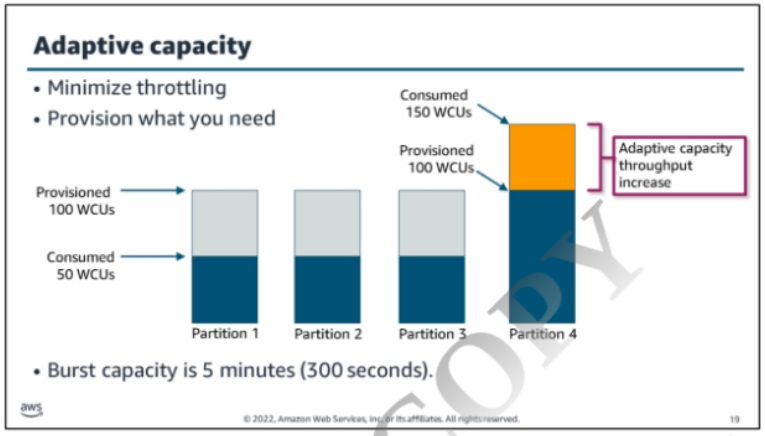


A global secondary index is considered to be *global* because queries on this index can span all the data in a table across all partitions.

The following are characteristics of a global secondary index:

- It can have a partition key and an optional sort key that are different from the partition key and sort key of the original table.
- Key values do not need to be unique.
- It can be created when a table is created, or it can be added to an existing table.
- It can be deleted.
- It supports eventual consistency only.
- It has its own provisioned throughput settings for read and write operations.
- Queries return only attributes that are projected into the index.

If eventual consistency is suitable for your application, we recommend that you use global secondary indexes.
For more information, see "Improving Data Access with Secondary Indexes" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html).



Adaptive capacity enables your DynamoDB table to run imbalanced workloads. It minimizes throttling and helps reduce your costs by enabling you to provision only what you need and not for spikes.

In this example, the table is provisioned with 400 WCUs evenly shared across four partitions. This sharing allows each partition to sustain up to 100 WCUs per second. Partitions 1, 2, and 3 each receives write traffic of 50 WCU/sec. Partition 4 receives 150 WCU/sec. This hot partition can accept write traffic while it still has unused burst capacity, but eventually it throttles traffic that exceeds 100 WCU/sec. DynamoDB adaptive capacity responds by increasing Partition 4's capacity so that it can sustain the higher workload of 150 WCU/sec without being throttled. The table's total provisioned capacity is 400 WCUs. Its total consumed capacity is 300 WCUs.

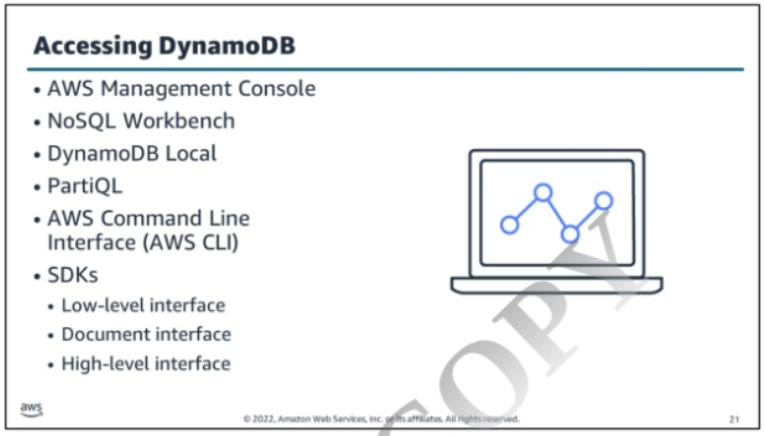
Additionally, DynamoDB provides some flexibility in portion throughput by allowing for burst capacity. Whenever a partition's throughput is not fully used, DynamoDB will reserve a portion of the unused capacity for any spikes in throughput.

For more information, see "Understanding DynamoDB Adaptive Capacity" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

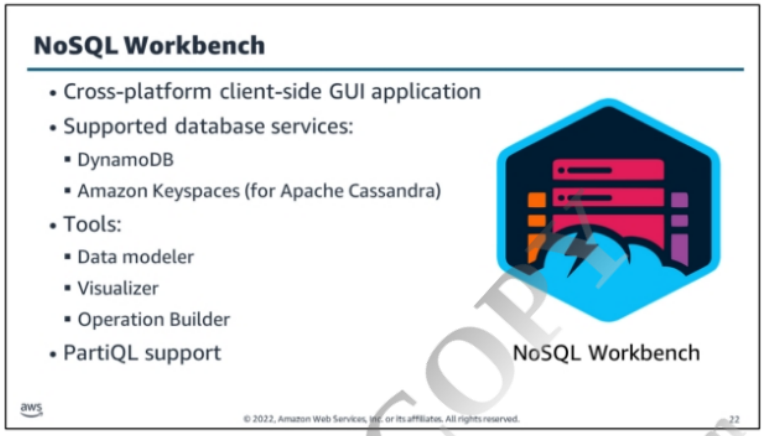
© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



In this section, you will learn how to access DynamoDB. You have already learned how to access DynamoDB using the console. Next, you will learn about additional tools you can use to access DynamoDB. Options to access and control your DynamoDB tables include:

- AWS Management Console
- NoSQL Workbench
- DynamoDB Local
- PartiQL
- AWS CLI
- SDKs
  - Low-level interface
  - Document interface
  - High-level interface

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



NoSQL Workbench for Amazon DynamoDB is a cross-platform, client-side GUI application for modern database development and operation. It currently supports both DynamoDB and Amazon Keyspaces (for Apache Cassandra). NoSQL Workbench is a unified visual IDE tool that provides data modeling, data visualization, and query development features. You can use these features to design, create, query, and manage DynamoDB tables.

Data modeler

- Build new data models
- Add tables and indexes
- Import and export models

Visualizer

- Add sample data
- Visualize data layout and structure
- Commit models to the cloud

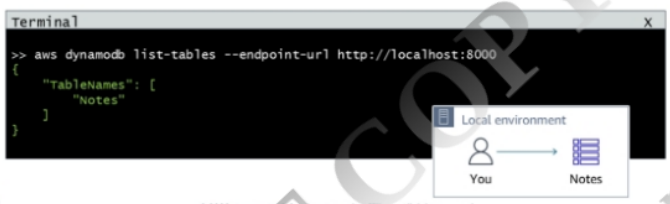
Operation Builder

- Build operations and queries (PartiQL)
- Use a guided form
- Generate code for data-plane operations

For more information, see "NoSQL Workbench for DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.html).

Developing using DynamoDB Local

- Requires no internet connection
- Saves on throughput, data storage, and data transfer fees
- Available as a download, Apache Maven dependency, or Docker image



With the downloadable version of Amazon DynamoDB, you can develop and test applications without accessing the DynamoDB web service. Instead, the database is self-contained on your computer. When you're ready to deploy your application in production, remove the local endpoint in the code. It then points to the DynamoDB web service.

Having this local version helps you save on throughput, data storage, and data transfer fees. In addition, you don't need an internet connection while you develop your application.

DynamoDB Local is available as a download (requires JRE), as an Apache Maven dependency, or as a Docker image. For more information, see "Setting Up DynamoDB Local (Downloadable Version)" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBLocal.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Interacting with PartiQL

PartiQL

- SQL-compatible query access
- Queries compatible with:
  - DynamoDB console
  - NoSQL Workbench
  - AWS CLI
  - DynamoDB APIs

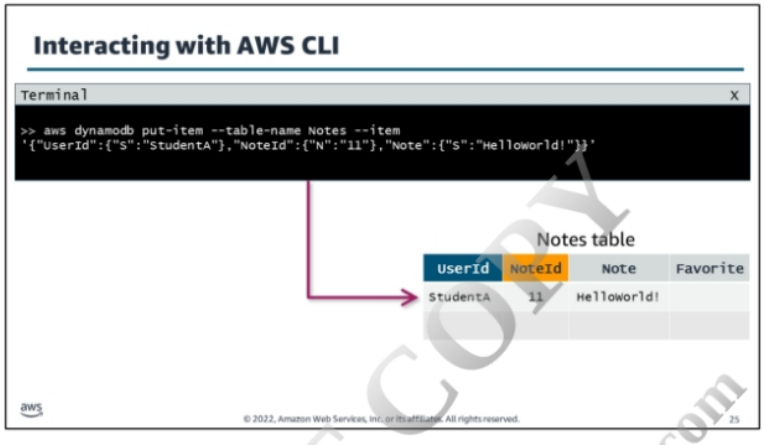
```
private static List<ParameterizedStatement> getPartiQLTransactionStatements() {
   List<ParameterizedStatement> statements = new ArrayList<ParameterizedStatement>();
   statements.add(new ParameterizedStatement()
        .withStatement("EXISTS(SELECT * FROM Notes where UserId = 'StudentA')"));
   return statements;
}
```


PartiQL is a SQL-compatible query language used to select, insert, update, and delete data in Amazon DynamoDB. Using PartiQL, you can easily interact with DynamoDB tables and run ad hoc queries. To run PartiQL queries in DynamoDB, you can use:

- DynamoDB console
- NoSQL Workbench
- AWS CLI
- DynamoDB APIs

For more information, see the following:

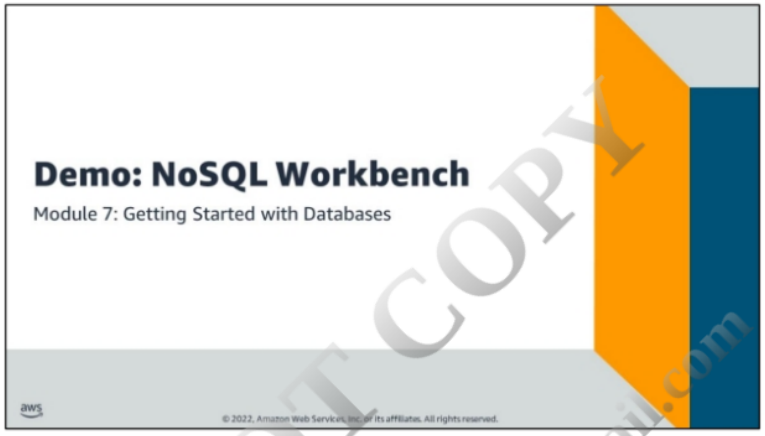
- PartiQL (https://partiql.org/)
- "PartiQL - A SQL-Compatible Query Language for Amazon DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-reference.html)



By interacting with your DynamoDB database with the AWS CLI, you can create interactions from the command line and then automate them with scripts. You can use the AWS CLI for ad hoc operations, such as creating a table or adding new items to a table.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



DO NOT COPY
sameersheik68@gmail.com

Product demonstration

NoSQL Workbench walkthrough

 27

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

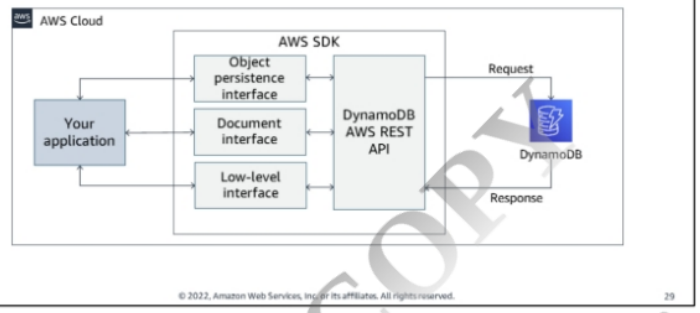
AWS Training and Certification Module 7: Getting Started with Databases

Programming with DynamoDB

Module 7: Getting Started with Databases

DO NOT COPY sameersheik68@gmail.com

AWS SDK support for DynamoDB, 1 of 2



This diagram provides a high-level overview of Amazon DynamoDB application programming using the AWS SDKs.

1. You write an application using an AWS SDK for your programming language.
2. Each AWS SDK provides one or more programmatic interfaces for working with DynamoDB. The specific interfaces available depend on which programming language and AWS SDK you use.
3. The AWS SDK constructs HTTP(S) requests for use with the low-level DynamoDB API.
4. The AWS SDK sends the request to the DynamoDB endpoint.
5. DynamoDB runs the request. If the request is successful, DynamoDB returns an HTTP 200 response code (OK). If the request is unsuccessful, DynamoDB returns an HTTP error code and an error message.
6. The AWS SDK processes the response and propagates it back to your application.

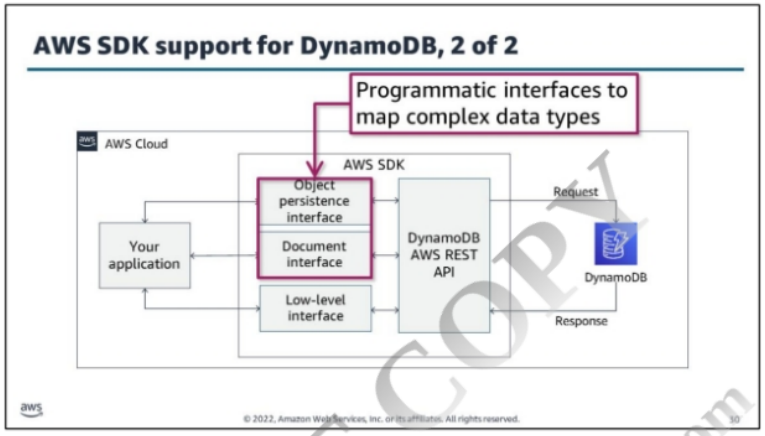
Each of the AWS SDKs provides important services to your application, including the following:

- Formatting HTTP(S) requests and serializing request parameters.
- Generating a cryptographic signature for each request.
- Forwarding requests to a DynamoDB endpoint and receiving responses from DynamoDB.
- Extracting the results from those responses.
- Implementing basic retry logic in case of errors.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases

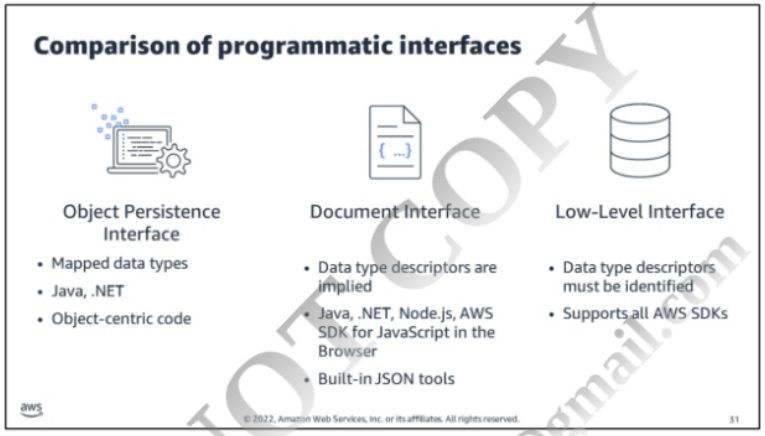
For more information, see "Overview of AWS SDK Support for DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.SDKOverview.html).



The Object persistence interface (High-level Interface) and Document interface allow for complex mapping of data types.

For more information, see "Overview of AWS SDK Support for DynamoDB" in the *Amazon DynamoDB Developer Guide* (https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.SDKOverview.html).

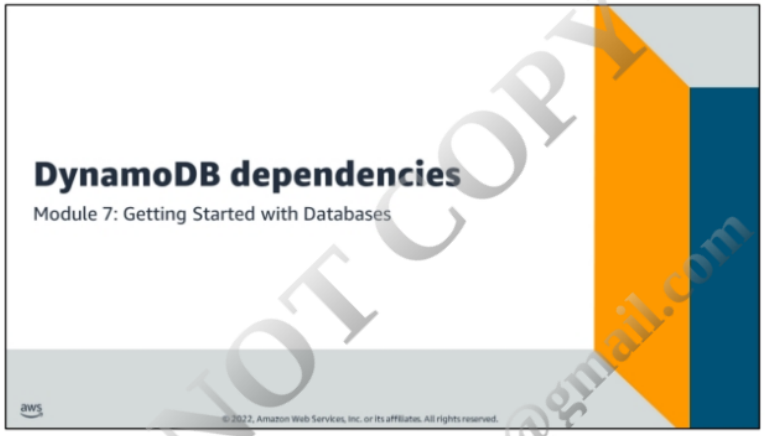
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Object persistence interface
The AWS SDK for .NET provides an object persistence model that you can use to map your client-side classes to Amazon DynamoDB tables. Each object instance then maps to an item in the corresponding tables. To save your client-side objects to the tables, the object persistence model provides the `DynamoDBContext` class an entry point to DynamoDB. This class provides you with a connection to DynamoDB. You can then access tables, perform various CRUD operations, and run queries.

Document interface
Many AWS SDKs provide a document interface that you can use to perform data plane operations (create, read, update, delete) on tables and indexes. With a document interface, you do not need to specify data type descriptors. The data types are implied by the semantics of the data itself. These AWS SDKs also provide methods to convert JSON documents to and from native Amazon DynamoDB data types.

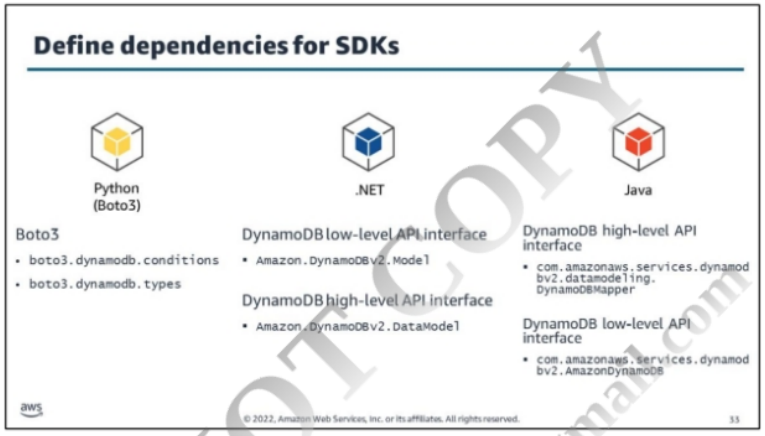
Low-level interface
Every language-specific AWS SDK provides a low-level interface for Amazon DynamoDB, with methods that closely resemble low-level DynamoDB API requests.

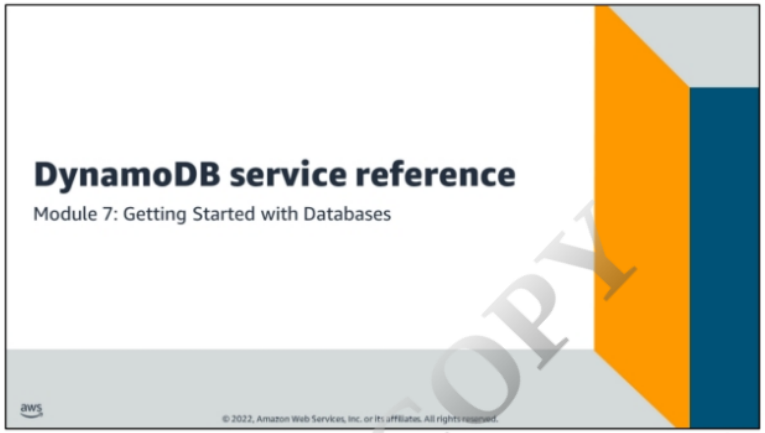


Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification

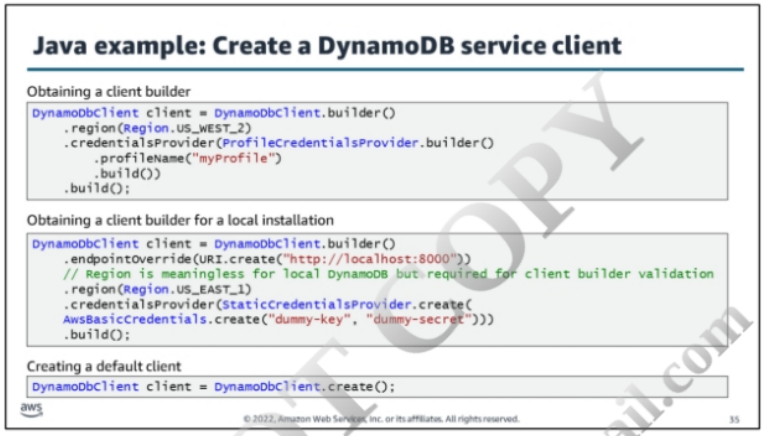
Module 7: Getting Started with Databases





Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



To make requests to AWS services, you must first create a service client object.

Obtain an instance of the client, use the static factory method builder

```
DynamoDbClient client = DynamoDbClient.builder()
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.builder()
                        .profileName("myProfile")
                        .build())
                .build();
```

Obtain an instance of the client, use the static factory method builder (Local installation)

```
DynamoDbClient client = DynamoDbClient.builder()
  .endpointOverride(URI.create("http://localhost:8000"))
  // The region is meaningless for local DynamoDb but required for client builder validation
  .region(Region.US_EAST_1)
  .credentialsProvider(StaticCredentialsProvider.create(
  AwsBasicCredentials.create("dummy-key", "dummy-secret")))
  .build();
```

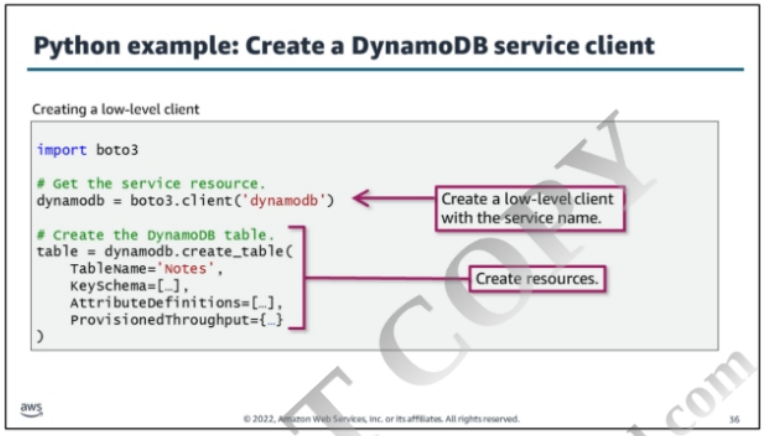
In general, avoid using this type of code in your application. If you include it in your application, take appropriate precautions to not expose plaintext keys in the code, over the network, or even in computer memory.

Create a default client
This method creates a service client with the default configuration.
DynamoDbClient client = DynamoDbClient.create();

For more information, see "Creating service clients" in the *AWS SDK for Java Developer Guide* (https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/using.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



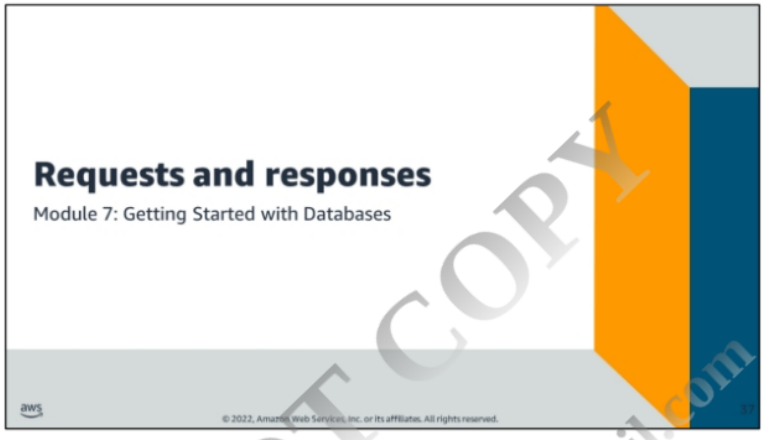
To make requests to AWS services, you must first create a service client object.

Low-level clients
Clients are created in a similar fashion to resources:

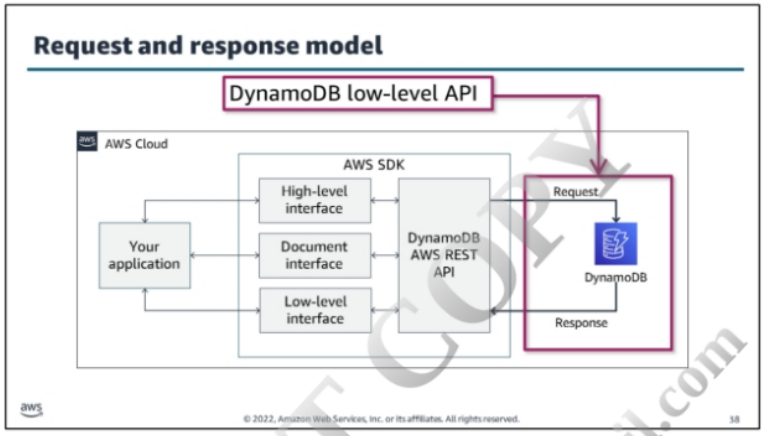
```
import boto3

# Get the service resource.
dynamodb = boto3.resource('dynamodb')

# Create the DynamoDB table.
table = dynamodb.create_table(
  TableName='Notes',
  KeySchema=[
    {
      'AttributeName': 'UserId',
      'KeyType': 'HASH'
    },
    {
      'AttributeName': 'NoteId',
      'KeyType': 'RANGE'
```

```
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': 'UserId',
            'AttributeType': 'S'
        },
        {
            'AttributeName': 'NoteId',
            'AttributeType': 'S'
        },
    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 5,
        'WriteCapacityUnits': 5
    }
)
```


Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



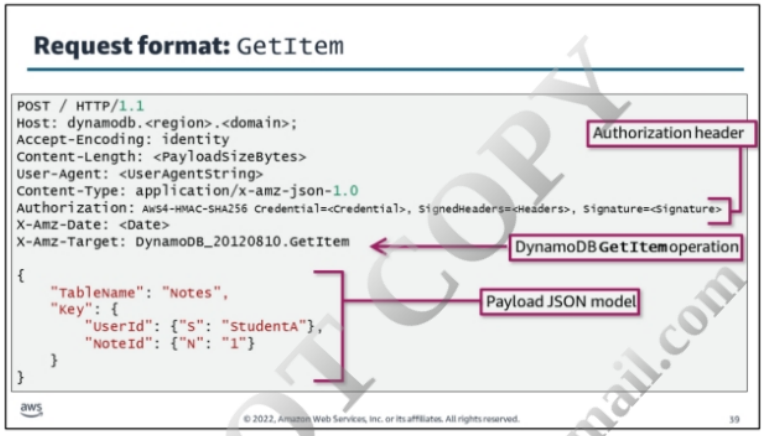
DO NOT COPY
sameersheik68@gmail.com



The Amazon DynamoDB *low-level API* is the protocol-level interface for DynamoDB. At this level, every HTTP(S) request must be correctly formatted and carry a valid digital signature. The AWS SDKs construct low-level DynamoDB API requests on your behalf and process the responses from DynamoDB. This capability lets you focus on your application logic, instead of low-level details.

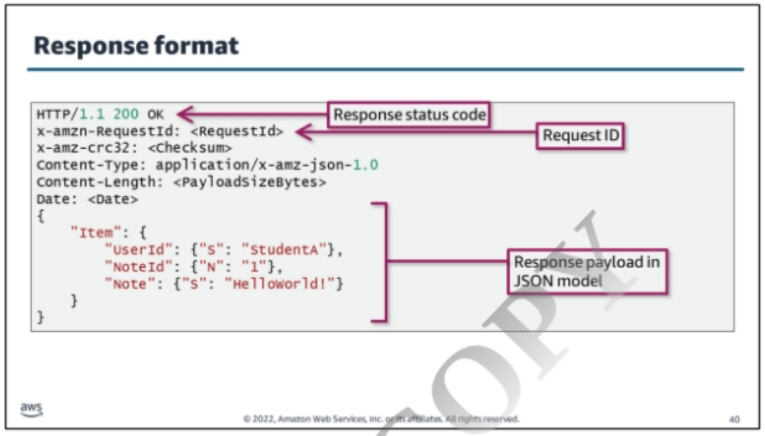
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



In this example, the `Authorization` header contains information required for DynamoDB to authenticate the request.

The `X-Amz-Target` header contains the name of a DynamoDB operation: `GetItem`. The GetItem operation returns a set of attributes for the item with the given primary key. The payload (body) of the request contains the parameters for the operation, in JSON model. For the `GetItem` operation, the parameters are `TableName` and `Key`.

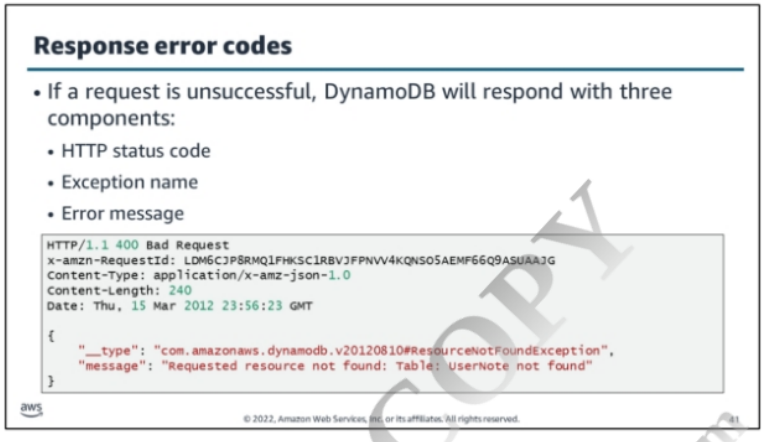


On receipt of the request, DynamoDB processes it and returns a response. For the request shown previously, the HTTP(S) response payload contains the results from the operation, as shown in the example.

The AWS SDK returns the response data to your application for further processing.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



When your program sends a request, DynamoDB attempts to process it. If the request is successful, DynamoDB returns an HTTP success status code (200 OK), along with the results from the requested operation.

Error components
If the request is unsuccessful, DynamoDB returns an error. Each error has the following components:
- An HTTP status code (such as 400)
- An exception name (such as `ResourceNotFoundException`)
- An error message (such as `Requested resource not found: Table: UserNote not found`)

Error messages and codes
The following are a few of the exceptions DynamoDB returns, grouped by HTTP status code.

HTTP status code 400
An HTTP 400 status code indicates a problem with your request, such as authentication failure, missing required parameters, or exceeding a table's provisioned throughput. You must fix the issue in your application before submitting the request again.

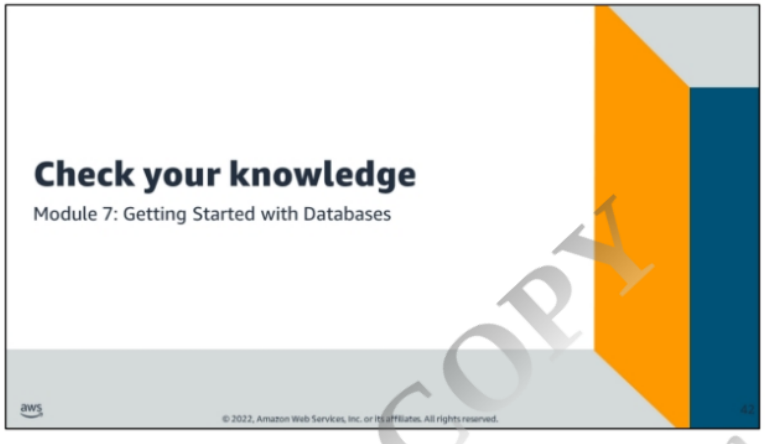
Message: *Requested resource not found.*

Example: The table that is being requested does not exist, or is too early in the CREATING state.

HTTP status code 5xx
An HTTP 5xx status code indicates a problem that must be resolved by AWS. This might be a transient error, in which case you can retry your request until it succeeds.
Example: Service unavailable (HTTP 503). DynamoDB is currently unavailable.

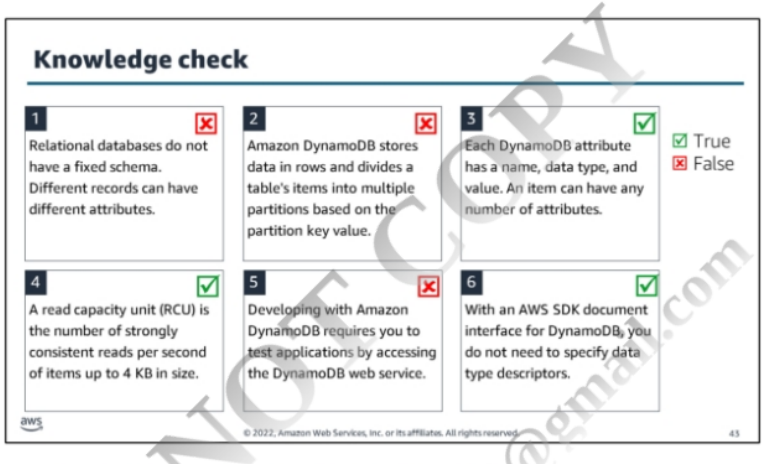
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



DO NOT COPY sameersheik68@gmail.com

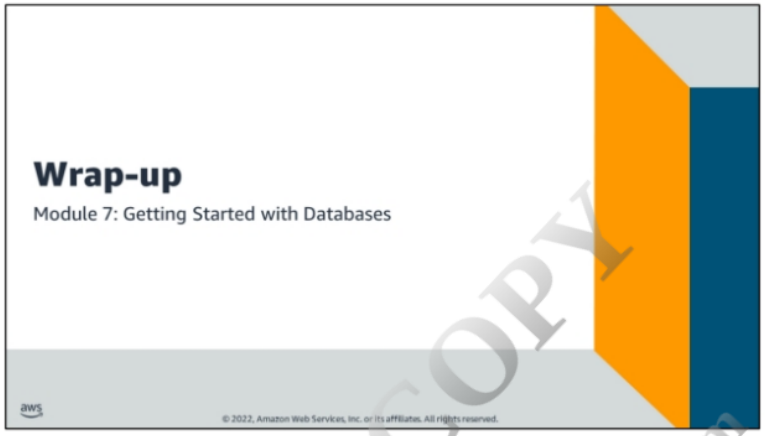
© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.



1. (False) Nonrelational databases do not have a fixed schema.
2. (False) DynamoDB stores data in partitions and divides a table's items into multiple partitions based on the *partition key* value.
3. (True)
4. (False) With the downloadable version of Amazon DynamoDB, you can develop and test applications without accessing the DynamoDB web service.
5. (True)
6. (True)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 7: Getting Started with Databases



Module summary

You are now able to:

- Describe the key components of DynamoDB
- Define SDK dependencies for your code
- Explain how to connect to DynamoDB
- Describe how to perform key table operations
- Describe how to build a request object
- Explain how to read a response object
- List the most common troubleshooting exceptions

 45

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-training.
All trademarks are the property of their owners.

 46