[ENGLISH]

Hello Cloud Gurus, and welcome to this lecture,

which is an introduction to DynamoDB,

so that you have some understanding of DynamoDB before we get our

hands dirty and create our own DynamoDB table.

So what is DynamoDB? Well,

it's a fast and flexible NoSQL database. And when we say NoSQL,

this is a type of database which stores and retrieves data

without needing to define its structure or database schema

upfront. So it's got a really flexible data model,

and it's an alternative to the more rigid relational databases

like Oracle or SQL Server,

which were first released before many of us were even born.

And DynamoDB is great for applications which need consistent

single-digit millisecond latency at any scale.

And it's a fully managed database which supports both key-value data

models and documents,

and the supported document formats are JSON, HTML,

and XML.

And DynamoDB is a great fit for mobile, web,

gaming, ad tech, IOT, and many,

many more applications. Now,

DynamoDB is also serverless and it integrates really,

really well with Lambda. It could be configured to auto scale,

and it's a really popular choice for developers and architects who are

designing serverless applications.

Now DynamoDB data is stored on solid state disks,

or SSDs,

which helped to give you consistently fast performance for reads and writes.

The underlying hardware supporting your DynamoDB table is always

spread across 3 geographically distinct data centers,

so that means it avoids any single point of failure if one of those

locations becomes unavailable.

And there's a choice of 2 consistency models for DynamoDB read

operations. First, there's eventually consistent reads,

which is the default. And second is strongly consistent reads.

Now with eventually consistent reads,

this means that consistency across all 3 copies of the data is

usually reached within one second.

So if you read data that you changed a second ago,

then DynamoDB should return the updated data.

And this gives you the best performance for DynamoDB reads.

Now with strongly consistent reads,

a strongly consistent read always reflects all successful writes.

So all writes are reflected across all 3 locations at once,

and this is best for read consistency. So if you're after read consistency,

then go for strongly consistent reads.

And there's also now support for ACID transactions,

and they call these DynamoDB transactions.

And DynamoDB transactions --

they provide the ability to perform ACID transactions.

So that's atomic, which means an all-or-nothing transaction,

consistent, which means consistent with data validation rules,

isolated,

which means that transactions must happen in isolation independently of one

another and not impacting each other,

and transactions must be durable as well,

so that they don't disappear if a system crashes.

And if you're not familiar with ACID transactions,

this is just an industry standard term.

And it's used to describe the properties of database transactions.

And it enables you to read or write multiple items across multiple

tables as an all-or-nothing operation. For example,

if you were making a credit card payment,

then you want that to happen as an all-or-nothing operation,

and you don't want your credit card to be charged,

but the person on the other end not receiving the payment.

Now DynamoDB data is made up of items or data records

stored in tables.

And here are a couple of examples of the kind of data that you might store in a

DynamoDB table. And in the first table,

we're storing player data for an online gaming application.

So we have our player ID and inventory special objects that they've collected

while playing the game. And in this game,

they can also collect gold coins as well.

So we've got the number of gold coins owned by each player as well.

And then we've got this objects table listing the available special

objects, a purchase price, whether the object is available or not,

and the current owner of the object. Now,

DynamoDB data records are called items.

So when we say items,

I want you to think about a row of data in the table.

And each item is made up of a set of attributes.

So when we think about attributes,

just think about a column of data in the table.

So for our Players table, the attributes would be things like the player ID,

the inventory, and the coins. Now,

DynamoDB supports key-value data structures. And when we say key-value,

the key is really the name of the data. So for example, player ID.

And then the value is the data itself.

So in our Players table,

that would be the name of the player that we're using as the player ID.

And the next thing we're going to talk about is primary keys.

And it's the primary key that actually allows us to query the data in the

table.

DynamoDB stores and retrieves data based on this primary

key. And there are 2 types of primary key,

and we're going to go through each of these in detail.

So we have the partition key and a composite key,

which is a combination of the partition key and a sort key.

So what does that actually mean? Well, first of all,

let's take a look at the partition key, and this one is pretty easy.

So the partition key is usually based on a unique attribute,

for example, a customer ID, a product ID,

an email address, or a vehicle registration number, or something like that.

And the reason it's called a partition key is because the value of the

partition key is input into an internal hash function within DynamoDB.

And the output of this hash function actually determines the partition or

the physical location where that data is going to be stored within DynamoDB.

And if you're using the partition key as your primary key,

then it makes sense that no 2 items are allowed to have the same

partition key. Now,

the second type of primary key is called a composite key,

and this is made up of a combination of a partition key and a

sort key.

And you would use this kind of composite key in a situation where the

partition key was not necessarily unique within your table.

So for example,

think about a table which is storing data about forum posts,

and you'd expect to have users posting multiple messages in the same forum.

So then the user ID alone is not going to be a unique identifier.

So in this case,

a combination of the user ID and then a sort key,

which will be a timestamp reflecting the time that the message was posted in the

forum. So this gives you a unique combination for your primary key,

and it means the items in the table may have the same partition key or user

ID, but they must have a different sort key.

So that is going to give you the unique combination required for a primary key.

Now, all items with the same partition key are stored together,

and then they're sorted according to the sort key value.

And this allows you to store multiple items using the same partition

key. So here is an example of a student table,

and in our first item, we've got the student ID of 1975,

and this could be our partition key because we know it's a unique ID.

But then think of another table where we've got a number of students and they're

taking multiple courses.

So they're not only enrolled in one single course.

So in our second example, we have our student named Riad.

And if he were to be enrolled in 2 different courses,

then his unique user ID would not necessarily be a good primary

key on its own,

because he's going to appear in our DynamoDB table numerous times

having completed multiple courses. So in this case,

we need to use a composite key.

So we need to use a partition key along with a sort key,

which could be the course name, in order to give us a unique primary key.

So let's take a look at some of our exam tips for DynamoDB.

Well, firstly, it's a low latency, NoSQL database,

and DynamoDB supports both document and key-value data

models, and the supported document formats are JSON, HTML,

and XML. And in terms of consistency models,

we have eventually consistent, which is the default.

And with eventually consistent, the overall read performance can be better.

However,

it can take up to one second for new writes or updates to your data

to be reflected when you read data from your DynamoDB table.

There's also strongly consistent,

which means that any writes will always be consistent across all 3

locations.

You don't need to wait up to one second to be propagated across all locations.

And there's also support for ACID transactions in the form of DynamoDB

transactions. And when you think about ACID transactions,

I want you to think about making a credit card payment.

And when you make that kind of payment,

you want it to be an all-or-nothing transaction.

You don't want your credit card to be charged,

and the person on the other end not receiving the payment.

It has to be an all-or-nothing transaction.

And DynamoDB consists of tables, items, and attributes.

And when we think of items, think of rows in the database or database records.

And when we think of attributes, just think of columns in the table.

And when it comes to primary keys, there are 2 different types of primary key.

Firstly, we've got our partition key, and secondly, a composite key,

which consists of a partition key and a sort key.

And just remember,

the partition key should be something like a user ID or a product ID.

So it should be a unique identifier.

And then a sort key could be a timestamp, or a date,

or whatever makes sense for your data to enable you to have a uniquely

identifying primary key. So that is the end of this lecture.

If you have any questions, please let me know. Otherwise,

feel free to move on to the next lecture. Thank you.

[SPANISH]

Hola Cloud Gurus, y bienvenidos a esta conferencia,

que es una introducción a DynamoDB ,

para que comprenda algo sobre DynamoDB antes de que obtengamos nuestro

manos sucias y crear nuestra propia tabla de DynamoDB .

Entonces, ¿qué es DynamoDB ? Bien,

es una base de datos NoSQL rápida y flexible . Y cuando decimos NoSQL ,

este es un tipo de base de datos que almacena y recupera datos

sin necesidad de definir su estructura o esquema de base de datos

por adelantado Así que tiene un modelo de datos muy flexible,

y es una alternativa a las bases de datos relacionales más rígidas

como Oracle o SQL Server,

que se lanzaron por primera vez antes de que muchos de nosotros naciéramos.

Y DynamoDB es ideal para aplicaciones que necesitan constante

de un solo dígito a cualquier escala.

Y es una base de datos completamente administrada que admite datos de clave-valor

modelos y documentos,

y los formatos de documentos admitidos son JSON, HTML,

y XML.

Y DynamoDB es ideal para dispositivos móviles, web,

juegos , tecnología publicitaria, IOT y muchos,

muchas más aplicaciones. Ahora,

DynamoDB también es sin servidor y se integra realmente,

muy bien con Lambda. Se puede configurar para escalar automáticamente,

y es una opción muy popular para desarrolladores y arquitectos que están

diseño aplicaciones sin servidor .

Ahora los datos de DynamoDB se almacenan en discos de estado sólido,

o SSD,

lo que ayudó a brindarle un rendimiento consistentemente rápido para lecturas y escrituras.

El hardware subyacente que admite su tabla de DynamoDB siempre está

distribuidos en 3 centros de datos geográficamente distintos,

eso significa que evita cualquier punto único de falla si uno de esos

ubicaciones deja de estar disponible.

Y hay una opción de 2 modelos de consistencia para la lectura de DynamoDB

operaciones \_ Primero, eventualmente hay lecturas consistentes,

que es el predeterminado. Y el segundo es lecturas fuertemente consistentes.

Ahora con lecturas eventualmente consistentes,

esto significa que la coherencia entre las 3 copias de los datos es

normalmente se alcanza en un segundo.

Entonces, si lees datos que cambiaste hace un segundo,

entonces DynamoDB debería devolver los datos actualizados.

Y esto le brinda el mejor rendimiento para las lecturas de DynamoDB .

Ahora con lecturas fuertemente consistentes,

una lectura fuertemente consistente siempre refleja todas las escrituras exitosas.

Entonces, todas las escrituras se reflejan en las 3 ubicaciones a la vez,

y esto es mejor para la consistencia de lectura. Entonces, si buscas consistencia de lectura,

luego busque lecturas fuertemente consistentes.

Y ahora también hay soporte para transacciones ACID,

y las llaman transacciones de DynamoDB .

Y transacciones de DynamoDB ,

proporcionan la capacidad de realizar transacciones ACID.

Así que eso es atómico, lo que significa una transacción de todo o nada,

consistente , lo que significa consistente con las reglas de validación de datos,

aislado ,

lo que significa que las transacciones deben ocurrir de forma aislada, independientemente de una

otra y no impactar entre sí,

y las transacciones también deben ser duraderas,

para que no desaparezcan si un sistema falla.

Y si no está familiarizado con las transacciones ACID,

este es solo un término estándar de la industria.

Y se usa para describir las propiedades de las transacciones de la base de datos.

Y le permite leer o escribir varios elementos a través de múltiples

tablas como una operación de todo o nada. Por ejemplo,

si estaba haciendo un pago con tarjeta de crédito,

entonces quieres que eso suceda como una operación de todo o nada,

y no quieres que se cargue tu tarjeta de crédito,

pero la persona en el otro extremo no recibe el pago.

Ahora los datos de DynamoDB se componen de elementos o registros de datos

almacenados en tablas.

Y aquí hay un par de ejemplos del tipo de datos que puede almacenar en un

de DynamoDB . Y en la primera tabla,

estamos almacenando datos de jugadores para una aplicación de juegos en línea.

Así que tenemos nuestra identificación de jugador y objetos especiales de inventario que han recolectado.

mientras juegas. Y en este juego,

también pueden recolectar monedas de oro .

Así que también tenemos la cantidad de monedas de oro que posee cada jugador.

Y luego tenemos esta tabla de objetos que enumera los especiales disponibles

objetos , un precio de compra, ya sea que el objeto esté disponible o no,

y el propietario actual del objeto. Ahora,

de DynamoDB se denominan elementos.

Así que cuando decimos artículos,

Quiero que pienses en una fila de datos en la tabla.

Y cada elemento se compone de un conjunto de atributos.

Entonces, cuando pensamos en los atributos,

solo piense en una columna de datos en la tabla.

Entonces, para nuestra tabla de jugadores, los atributos serían cosas como la identificación del jugador,

el inventario y las monedas. Ahora,

DynamoDB admite estructuras de datos de clave-valor. Y cuando decimos clave-valor,

la clave es realmente el nombre de los datos. Entonces, por ejemplo, la identificación del jugador.

Y luego el valor son los datos en sí.

Así que en nuestra tabla de Jugadores,

ese sería el nombre del jugador que estamos usando como ID de jugador.

Y lo siguiente de lo que vamos a hablar son las claves primarias.

Y es la clave principal la que realmente nos permite consultar los datos en el

mesa \_

DynamoDB almacena y recupera datos basados en este primario

llave \_ Y hay 2 tipos de clave principal,

y vamos a ir a través de cada uno de estos en detalle.

Así que tenemos la clave de partición y una clave compuesta,

que es una combinación de la clave de partición y una clave de clasificación.

Entonces, ¿qué significa eso realmente? Pues, ante todo,

echemos un vistazo a la clave de partición, y esta es bastante fácil.

Entonces, la clave de partición generalmente se basa en un atributo único,

por ejemplo, un ID de cliente, un ID de producto,

una dirección de correo electrónico, o el número de registro de un vehículo, o algo por el estilo.

Y la razón por la que se llama clave de partición es porque el valor de la

de partición se ingresa en una función hash interna dentro de DynamoDB .

Y la salida de esta función hash en realidad determina la partición o

la ubicación física donde se almacenarán esos datos dentro de DynamoDB .

Y si está utilizando la clave de partición como su clave principal,

entonces tiene sentido que no se permita que 2 elementos tengan el mismo

clave de partición Ahora,

el segundo tipo de clave primaria se llama clave compuesta,

y esta se compone de una combinación de una clave de partición y una

de clasificación .

Y usaría este tipo de clave compuesta en una situación en la que el

de partición no era necesariamente única dentro de su tabla.

Así por ejemplo,

piense en una tabla que almacena datos sobre publicaciones en el foro,

y esperaría que los usuarios publicaran varios mensajes en el mismo foro.

Entonces, la ID de usuario por sí sola no será un identificador único.

Entonces, en este caso,

una combinación del ID de usuario y luego una clave de clasificación,

que será una marca de tiempo que reflejará la hora en que se publicó el mensaje en el

foro \_ Entonces esto le da una combinación única para su clave principal,

y significa que los elementos de la tabla pueden tener la misma clave de partición o usuario

ID, pero deben tener una clave de clasificación diferente.

Eso le dará la combinación única requerida para una clave principal.

Ahora, todos los elementos con la misma clave de partición se almacenan juntos,

y luego se ordenan de acuerdo con el valor de la clave de ordenación.

Y esto le permite almacenar varios elementos usando la misma partición

llave \_ Así que aquí hay un ejemplo de una mesa de estudiante,

y en nuestro primer elemento, tenemos la identificación de estudiante de 1975,

y esta podría ser nuestra clave de partición porque sabemos que es una identificación única.

Pero luego piense en otra mesa donde tenemos varios estudiantes y están

tomando varios cursos.

Entonces no solo están inscritos en un solo curso.

Entonces, en nuestro segundo ejemplo, tenemos a nuestro estudiante llamado Riad .

Y si estuviera matriculado en 2 cursos diferentes,

entonces su ID de usuario único no sería necesariamente un buen primario

llave sola,

porque va a aparecer en nuestra tabla de DynamoDB varias veces

haber completado varios cursos. Entonces, en este caso,

Necesitamos usar una clave compuesta.

Entonces necesitamos usar una clave de partición junto con una clave de clasificación,

que podría ser el nombre del curso, para así darnos una clave primaria única.

Así que echemos un vistazo a algunos de nuestros consejos para el examen de DynamoDB .

Bueno, en primer lugar, es una base de datos NoSQL de baja latencia ,

y DynamoDB admite datos de documentos y valores clave

modelos y los formatos de documentos admitidos son JSON, HTML,

y XML. Y en términos de modelos de consistencia,

tenemos consistente, que es el predeterminado.

Y con eventualmente consistente, el rendimiento general de lectura puede ser mejor.

Sin embargo,

puede tomar hasta un segundo para nuevas escrituras o actualizaciones de sus datos

para que se refleje cuando lea datos de su tabla de DynamoDB .

También hay una fuerte consistencia,

lo que significa que cualquier escritura siempre será consistente en los 3

ubicaciones \_

No necesita esperar hasta un segundo para que se propague en todas las ubicaciones.

Y también hay soporte para transacciones ACID en forma de DynamoDB

transacciones \_ Y cuando piensa en transacciones ACID,

Quiero que pienses en hacer un pago con tarjeta de crédito.

Y cuando haces ese tipo de pago,

quieres que sea una transacción de todo o nada.

No desea que se cargue su tarjeta de crédito,

y la persona en el otro extremo no recibe el pago.

Tiene que ser una transacción de todo o nada.

Y DynamoDB consta de tablas, elementos y atributos.

Y cuando pensamos en elementos, pensemos en las filas de la base de datos o en los registros de la base de datos.

Y cuando pensamos en atributos, solo piense en las columnas de la tabla.

Y cuando se trata de claves principales, hay 2 tipos diferentes de claves principales.

En primer lugar, tenemos nuestra clave de partición y, en segundo lugar, una clave compuesta,

que consta de una clave de partición y una clave de clasificación.

Y solo recuerda,

la clave de partición debe ser algo así como una ID de usuario o una ID de producto.

Así que debería ser un identificador único.

Y luego, una clave de clasificación podría ser una marca de tiempo o una fecha,

o lo que tenga sentido para sus datos para permitirle tener una única

identificar la clave principal. Así que ese es el final de esta conferencia.

Si tiene alguna pregunta, por favor hágamelo saber. De lo contrario,

Siéntete libre de pasar a la siguiente lección. Gracias.