BOGOTÁ | 18 DE JULIO DE 2024

aws SUMMIT

DEV303

Diseño Eficiente de Modelos de Datos en Amazon DynamoDB

Guille Ojeda

(él, he/him)
Cloud Software Architect
Caylent

Derribando mitos sobre DynamoDB

Mito	Verdad
NoSQL = No Relacional	Los datos tienen relación
Los datos no están estructurados	Se almacenan estructurados y se consultan por índices
Todas las lecturas son iguales	Lecturas por PK = Query  Rápidas y muy baratas Lecturas fuera de PK = Scan  Lentas y muy caras (leen toda la tabla!)

Acerca de mí

Guille Ojeda

(él, he/him)
Cloud Software Architect
Caylent

- Cloud Software Architect en Caylent
- Publiqué 2 libros y 100+ artículos
- AWS Community Builder
- Caylent Champion

Agenda

01 Derribando mitos sobre DynamoDB

02 Conceptos básicos de DynamoDB

03 Estrategias de diseño de datos en DynamoDB

04 Analizar un escenario práctico

05 Diseñar patrones de acceso a los datos

06 Implementar cada patrón

07 Lecciones sobre DynamoDB

Conceptos básicos de DynamoDB

Base de datos NoSQL

Totalmente gestionada por AWS (managed service)

Baja latencia: millones de requests con <10ms

Lectura y escritura escalables

Altamente disponible por diseño

Conceptos básicos de DynamoDB

Concepto clave: Tabla

Ni parecido a una tabla en BDs SQL

Clave de Partición (PK)

Divide los datos en distintas particiones físicas

Clave de Ordenamiento (SK)

Permite ordenar los resultados

Conceptos básicos de DynamoDB

Concepto clave: Tabla

Ni parecido a una tabla en BDs SQL

Clave de Partición (PK)

Divide los datos en distintas particiones físicas

Clave de Ordenamiento (SK)

Permite ordenar los resultados

Juntas son la clave primaria

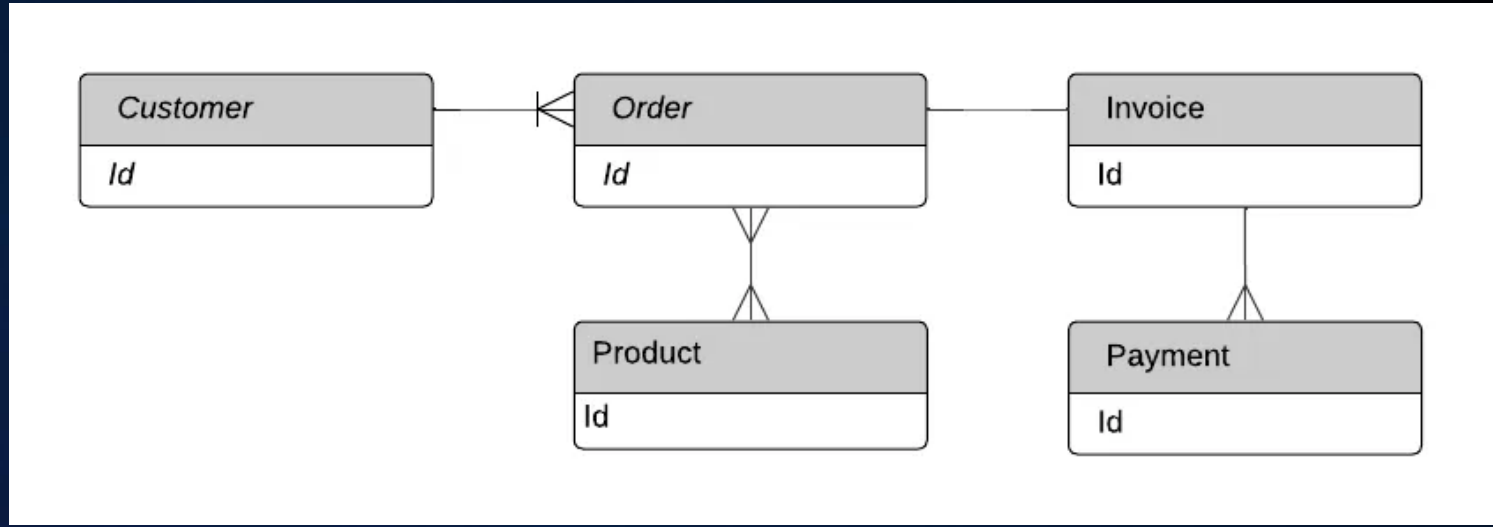
Estrategias de diseño de datos en DynamoDB

Tabla Única	Múltiples Tablas
Una sola tabla	Una tabla por grupo de entidades
Capacidad compartida	Capacidad separada
Mayor duplicación de datos	Menor duplicación de datos
Captura de Datos Modificados requiere mayor esfuerzo	Captura de Datos Modificados más fácil de gestionar
Soporta transacciones	Soporta transacciones

Analizar un escenario práctico

App e-commerce:

- Cliente entra a la web
- Se registra
- Ve distintos productos
- Hace una compra
- Paga la compra



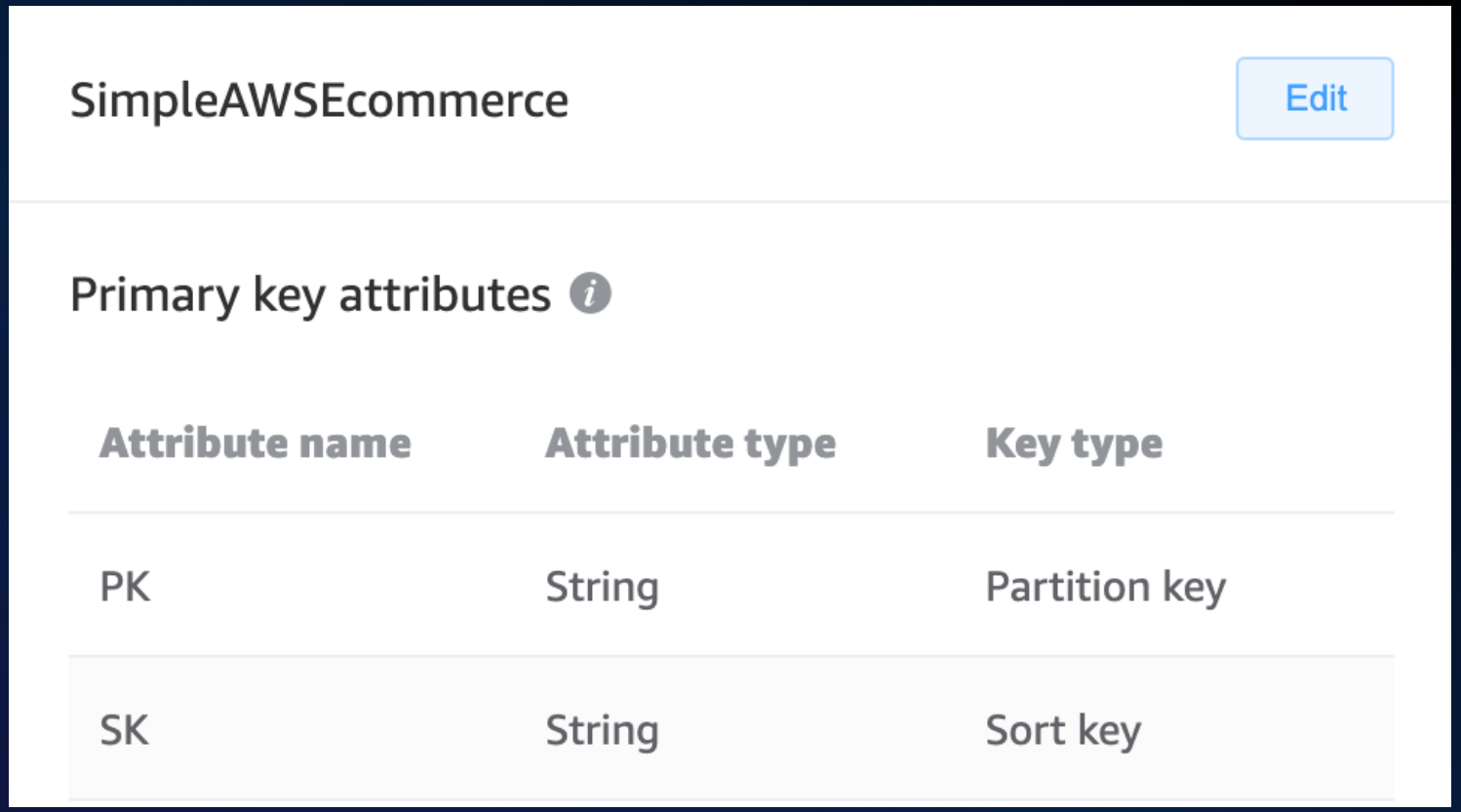
Diseñar patrones de acceso

Resultado esperado	Query
Datos de un cliente	Get Customer según customerId
Datos de un producto	Get Product según productId
Datos de una venta	Get Order según orderId
Productos vendidos en una venta	Get todos los Products según orderId
Recibos de una venta	Get Invoice según orderId
Cuánto se vendió de un producto en un rango de fecha	Get todos los OrderItems según productId en rango de fecha

Implementar cada patrón

Paso previo: Crear la tabla

Creamos una tabla vacía
Partition Key: "PK"
Sort Key: "SK"



1- Get Customer según customerId

PK=customerId y SK=customerId. Customer ID es c#12345
Consulta de ejemplo: PK="c#12345" and SK="c#12345"

Primary key				
Partition key: PK	Sort key: SK			
c#12345	c#12345	EntityType	Email	Name
		customer	guille@simpleaws.dev	Guille

2- Get Product según productId

PK=productId y SK=productId. Product ID es p#12345
Consulta de ejemplo: PK="p#12345" and SK="p#12345"

		EntityType	Detail	Price
p#12345	p#12345	product	{"Name":{"S":"Nodejs on AWS"},"Description":{"S":"Ebook with step by step to deploy Nodejs on AWS"}}	10

Comparemos

Primary key				
Partition key: PK	Sort key: SK			
c#12345	c#12345	EntityType	Email	Name
		customer	guille@simpleaws.dev	Guille
p#12345	p#12345	EntityType	Detail	Price
		product	{"Name":{"S":"Nodejs on AWS"},"Description":{"S":"Ebook with step by step to deploy Nodejs on AWS"}}	10

3- Get Order según orderId
4- Get todos los Products según orderId

Order: tiene sentido como entidad separada. Order ID o#12345

OrderItem: intersección entre Order y Product. No existe sin Order.
No necesita OrderItemID. PK=orderId, SK=productId
Otros atributos: quantity, price

Consulta de ejemplo para 3- Get Order según orderId: PK="o#12345"
Consulta de ejemplo para 4- Get todos los Products según orderId:
PK="o#12345" and SK begins_with "p#"

3- Get Order según orderId
4- Get todos los Products según orderId

Primary key		
Partition key: PK	**Sort key: SK**	
o#12345	c#12345	EntityType
		order
	p#12345	EntityType
		orderItem
	p#99887	EntityType
		orderItem

5- Get Invoice según orderId

Invoice: Invoice ID i#12345
No tiene sentido que Invoice ID sea Partition Key
PK = orderID y SK = invoiceID
Consulta de ejemplo: PK="o#12345" y SK begins_with "i#"

Primary key			
Partition key: PK	**Sort key: SK**		
o#12346	i#55443	EntityType	Amount
		invoice	120

6- Get todos los OrderItems según productId en rango de fecha

Nueva consulta sobre entidades que ya existen
Query "en rango" -> Sort Key
Product ID no es Partition Key en ningún elemento
Necesitamos guardar los mismos datos con otra Partition Key y Sort Key
Vamos a crear un Global Secondary Index (GSI)

Los índices duplican los datos automáticamente y permiten consultarlos de manera distinta. Usan distinta Partition Key y Sort Key

Nota: Local Secondary Index (LSI): la misma PK y distinta SK

6- Get todos los OrderItems según productId en rango de fecha

Nuevo índice GSI1. Partition Key = GSI1-PK y Sort Key = GSI1-SK
GSI1-PK = Product ID y GSI1-SK = Fecha
Proyectamos todos los atributos: se duplican automáticamente

Primary key		Attributes				
Partition key: GSI1-PK	**Sort key: GSI1-SK**					
p#12345	2023-04-25T19:18:00	PK	SK	EntityType	Price	Quantity
		o#12345	p#12345	orderItem	10	2
p#99887	2023-04-25T19:20:00	PK	SK	EntityType	Price	Quantity
		o#12345	p#99887	orderItem	20	5

Resultado final

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
c#12345	c#12345	EntityType	Email	Name
		customer	guille@simpleaws.dev	Guille
p#12345	p#12345	EntityType	Detail	Price
		product	{"Name":{"S":"Nodejs on AWS"},"Description":{"S":"Ebook with step by step to deploy Nodejs on AWS"}}	10
p#99887	p#99887	EntityType	Detail	Price
		product	{"Name":{"S":"Event Driven Architectures Book"},"Description": {"S":"The other book, the one about Event-Driven Architectures"}}	20

Resultado final

o#12345	c#12345	EntityType	Date			
		order	2023-04-25T19:10:00			
	i#55443	EntityType	Amount			
		invoice	120			
	p#12345	EntityType	GSI1-PK	GSI1-SK	Price	Quantity
		orderItem	p#12345	2023-04-25T19:18:00	10	2
	p#99887	EntityType	GSI1-PK	GSI1-SK	Price	Quantity
		orderItem	p#99887	2023-04-25T19:20:00	20	5

Lecciones sobre DynamoDB

- Debemos diseñar el modelo datos, y no es fácil
- No hay un único modelo de datos posible
- El modelo de datos es evolutivo
- Hay duplicación de datos
- Siempre Query 
- Nunca Scan 

El modelo de datos en DynamoDB depende de los patrones de acceso, no de los datos.

Guille Ojeda

Cloud Software Architect, Caylent

Siempre Query ⚡💵

Nunca Scan 🐢💸

Guille Ojeda

Cloud Software Architect, Caylent