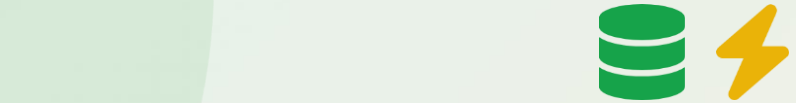
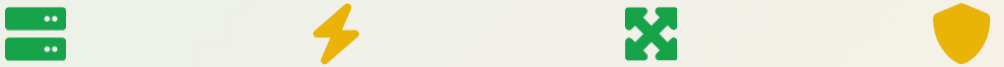


Getting Started with Amazon DynamoDB

Cơ sở dữ liệu NoSQL serverless, quản lý toàn phần bởi AWS với hiệu năng cao, độ trễ thấp và khả năng mở rộng tự động



Serverless | High Performance | Auto Scaling | Managed Service

October 18, 2025

Mục tiêu học tập

Mô tả các thành phần chính của DynamoDB

Hiểu rõ các thành phần cơ bản của DynamoDB bao gồm Table, Item, Attributes, Primary Key và Data Types.

Khám phá nhiều cách kết nối tới DynamoDB

Tìm hiểu các phương thức kết nối khác nhau đến DynamoDB bao gồm AWS SDK, CLI, Console, Lambda và môi trường local.

Xác định các dependency và cấu hình SDK trong code

Hướng dẫn cách cấu hình và sử dụng AWS SDK cho các ngôn ngữ lập trình khác nhau trong ứng dụng của bạn.

Làm việc với các đối tượng Request và Response

Hướng dẫn cách sử dụng các đối tượng request và response khi tương tác với DynamoDB trong ứng dụng.

Giới thiệu DynamoDB

Amazon DynamoDB là một dịch vụ cơ sở dữ liệu NoSQL serverless, được quản lý toàn phần bởi AWS. Được thiết kế để cung cấp hiệu năng cao và độ trễ thấp ở quy mô lớn.

Serverless

Không cần quản trị hạ tầng, loại bỏ các tác vụ như quản lý máy chủ, vá lỗi hay sao lưu thủ công.

Hiệu năng cao

Tự động phân vùng dữ liệu (partitioning) và nhân bản đa vùng sẵn sàng (Multi-AZ replication) để đảm bảo tính sẵn sàng.

Khả năng mở rộng

Tự động chia vùng dữ liệu và mở rộng tự động để đáp ứng nhu cầu biến động của ứng dụng.

Tích hợp với các dịch vụ AWS khác

- Lambda
- API Gateway
- S3
- IAM
- CloudWatch
- Kinesis

So sánh RDBMS và NoSQL

Đặc điểm	NoSQL (DynamoDB)	RDBMS
Schema	Linh hoạt, không yêu cầu schema cố định.	Cố định, yêu cầu định nghĩa trước.
Ngôn ngữ	API hoặc ngôn ngữ truy vấn chuyên biệt (ví dụ: PartiQL).	SQL, hỗ trợ các thao tác quan hệ (join, transaction).
Cấu trúc dữ liệu	Dạng key-value, document, đồ thị, hoặc cột rộng.	Dạng bảng với các hàng và cột.
Khả năng mở rộng	Scale ngang (horizontal scaling), tự động chia vùng dữ liệu.	Scale dọc (vertical scaling), có giới hạn hiệu năng.
Tính nhất quán	Thường là nhất quán cuối cùng (Eventually Consistent), có thể cấu	Thường là nhất quán mạnh (Strongly Consistent).

⚙ Thành phần chính của DynamoDB

Table (Bảng)

Tập hợp các item, tương tự như một bảng trong cơ sở dữ liệu quan hệ. Ví dụ: bảng `People` lưu trữ thông tin liên hệ cá nhân.

Item (Mục)

Bản ghi đơn lẻ trong bảng, tương tự như một hàng trong RDBMS. Mỗi item là một tập hợp các thuộc tính và có kích thước tối đa 400 KB.

Attributes (Thuộc tính)

Các cặp key-value bên trong một item, đại diện cho các phần tử dữ liệu cơ bản. Hỗ trợ lồng nhau lên đến 32 cấp độ.

Primary Key (Khóa chính)

Thuộc tính hoặc tập hợp các thuộc tính xác định duy nhất mỗi item trong bảng. Có thể là Partition Key hoặc Partition Key + Sort Key.

Data Types (Kiểu dữ liệu)

Hỗ trợ nhiều kiểu dữ liệu: Scalar Types (String, Number, Binary, Boolean, Null), Document Types (List, Map), Set Types (String Set, Number Set, Binary Set).

Mối quan hệ giữa các thành phần

- Table chứa nhiều Item
- Item chứa nhiều Attributes
- Primary Key xác định duy nhất mỗi Item

Điểm cần lưu ý

Primary Key trong DynamoDB

1. Partition Key (Hash Key)

- Là một thuộc tính duy nhất đóng vai trò là khóa chính của bảng
- DynamoDB sử dụng giá trị của Partition Key để băm (hash) và xác định phân vùng vật lý nơi item sẽ được lưu trữ
- Mọi item trong bảng phải có giá trị Partition Key khác nhau

Ví dụ:

UserID (Partition Key)	Name	Email
101	An	an@example.com
102	Binh	binh@example.com

2. Partition Key + Sort Key (Composite Key)

- Là sự kết hợp của hai thuộc tính để định danh một item
- Thuộc tính đầu tiên là Partition Key, thuộc tính thứ hai là Sort Key
- Các item có cùng giá trị Partition Key sẽ được nhóm lại và sắp xếp theo giá trị của Sort Key
- Hỗ trợ lưu trữ nhiều bản ghi có cùng nhóm logic và truy vấn hiệu quả theo phạm vi giá trị Sort Key

Ví dụ:

UserID (Partition Key)	OrderDate (Sort Key)	OrderAmount
101	2025-10-10	250
101	2025-10-12	120
102	2025-10-11	300

Partition và Data Distribution

Cách Partition hoạt động

- DynamoDB tự động chia dữ liệu thành nhiều **phân vùng vật lý** để đảm bảo hiệu năng và khả năng mở rộng.
- Dữ liệu được phân phối dựa trên giá trị **băm (hash)** của Partition Key.
- Mục tiêu chính là đảm bảo hiệu năng ổn định và khả năng mở rộng gần như vô hạn.

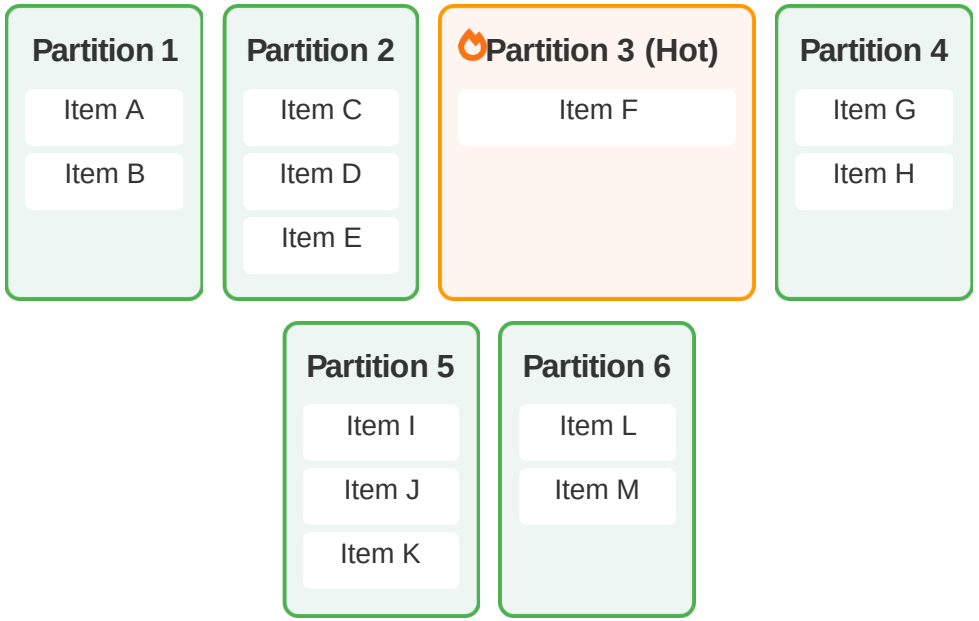
Tránh "hot partition"

> Một Partition Key được thiết kế kém có thể dẫn đến "hot partition" – tình trạng một phân vùng nhận quá nhiều yêu cầu đọc/ghi, gây ra tắc nghẽn và giảm hiệu suất.

Adaptive Capacity

> DynamoDB sử dụng khả năng thích ứng để tự động tăng dung lượng thông lượng cho các phân vùng "nóng" nhằm giảm thiểu tình trạng điều tiết (throttling).

Minh họa Partitioning



Best Practice

Chọn Partition Key có giá trị phân tán tốt để đảm bảo hiệu năng tối ưu và tránh "hot partition".

⚡ Capacity Mode (Chế độ dung lượng)

Provisioned Mode

Mô tả

Bạn xác định trước số lượng Đơn vị dung lượng đọc (RCU) và Đơn vị dung lượng ghi (WCU) mà ứng dụng yêu cầu.

Ưu điểm

Phù hợp cho các ứng dụng có workload ổn định, dễ dự đoán. Có thể tiết kiệm chi phí nếu quản lý dung lượng hiệu quả.

$ Chi phí

Trả tiền cho dung lượng đã cấp phát, bất kể bạn sử dụng bao nhiêu. DynamoDB hỗ trợ tự động điều chỉnh RCU/WCU.

On-Demand Mode

Mô tả

DynamoDB tự động điều chỉnh dung lượng để đáp ứng nhu cầu thực tế của ứng dụng. Bạn chỉ trả tiền cho số lượng yêu cầu thực tế.

Ưu điểm

Lý tưởng cho các ứng dụng có workload biến động, không thể đoán trước hoặc có lưu lượng truy cập tăng đột biến.

$ Chi phí

Đơn giản hơn trong việc quản lý vì bạn không cần lo lắng về việc cấp phát hay điều chỉnh dung lượng, nhưng có thể tốn kém hơn cho workload ổn định.

🔁 Bạn có thể chuyển đổi giữa hai chế độ này tối đa bốn lần trong khoảng thời gian 24 giờ.

Read/Write Capacity Units

Read Capacity Unit (RCU)

- 1 RCU = 1 thao tác đọc mạnh mẽ nhất quán (Strongly Consistent) mỗi giây cho item tối đa 4 KB
- Nếu dùng đọc nhất quán cuối cùng (Eventually Consistent), 1 RCU = 2 thao tác mỗi giây
- Mọi thao tác đọc đều tiêu tốn RCU, bất kể bạn có sử dụng hết dữ liệu hay không

Ví dụ tính RCU:

Size item	Loại đọc	RCU cần thiết
4 KB	Strongly Consistent	1
4 KB	Eventually Consistent	0.5 (làm tròn lên thành 1)
5 KB	Strongly Consistent	2 (5/4 = 1.25, làm tròn lên)

Write Capacity Unit (WCU)

- 1 WCU = 1 thao tác ghi mỗi giây cho item tối đa 1 KB
- Nếu item lớn hơn 1 KB, số WCU cần thiết sẽ được làm tròn lên
- Mọi thao tác ghi đều tiêu tốn WCU, bất kể bạn có ghi hết dữ liệu hay không

Ví dụ tính WCU:

Size item	WCU cần thiết
1 KB	1
1.5 KB	2 (1.5/1 = 1.5, làm tròn lên)
2 KB	2

📄 Read/Write Capacity Units

📖 Ví dụ 1 – Read Capacity

Trường hợp	Số lần đọc/giây	Loại đọc	Kích thước item	Cách tính	Kết quả
A	10	Strongly Consistent	4 KB	10 × (4/4)	**10 RCU**
B	16	Eventually Consistent	12 KB	(16 ÷ 2) × (12/4) = 8 × 3	**24 RCU**
C	10	Strongly Consistent	6 KB (làm tròn 8 KB)	10 × (8/4)	**20 RCU**

✍️ Ví dụ 2 – Write Capacity

Trường hợp	Số lần ghi/giây	Kích thước item	Cách tính	Kết quả
A	10	2 KB	10 × (2/1)	**20 WCU**
B	6	4.5 KB (làm tròn 5 KB)	6 × (5/1)	**30 WCU**
C	120/phút (2/giây)	2 KB	2 × (2/1)	**4 WCU**

Consistency Models

Eventually Consistent Read

Mô tả:

Dữ liệu có thể được trả về từ một bản sao không phải là bản sao mới nhất. Các thay đổi dữ liệu sẽ được truyền đến tất cả các bản sao trong vòng vài mili-giây.

Ưu điểm:

- Hiệu năng cao, độ trễ thấp
- Chi phí thấp (chỉ 0.5 RCU cho mỗi 4KB dữ liệu)

Trường hợp sử dụng:

- Số lượt xem bài viết, danh mục sản phẩm không thay đổi thường xuyên
- Mặc định của DynamoDB

Strongly Consistent Read

Mô tả:

Khi bạn thực hiện một thao tác đọc, DynamoDB sẽ trả về dữ liệu mới nhất, phản ánh tất cả các thao tác ghi thành công trước đó.

Ưu điểm:

- Đảm bảo bạn luôn nhận được dữ liệu cập nhật nhất
- Tốn kém hơn (1 RCU cho mỗi 4KB dữ liệu) so với Eventually Consistent Read

Trường hợp sử dụng:

- Giao dịch tài chính, quản lý hàng tồn kho
- Yêu cầu tính chính xác cao của dữ liệu ngay lập tức

Secondary Indexes

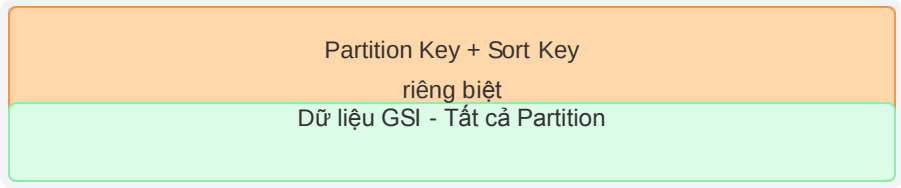
Local Secondary Index (LSI)

- **Chia sẻ** cùng Partition Key với bảng gốc
- Có Sort Key **khác** với bảng gốc
- Đặt cùng vị trí **vật lý** với dữ liệu bảng gốc
- Hỗ trợ cả **Eventually Consistent Read** và **Strongly Consistent Read**
- Giới hạn: **tối đa 5 LSI** mỗi bảng
- Phải được tạo cùng lúc khi tạo bảng, không thể thêm/xóa sau này
- **Tiêu thụ** RCU/WCU từ bảng gốc

VS

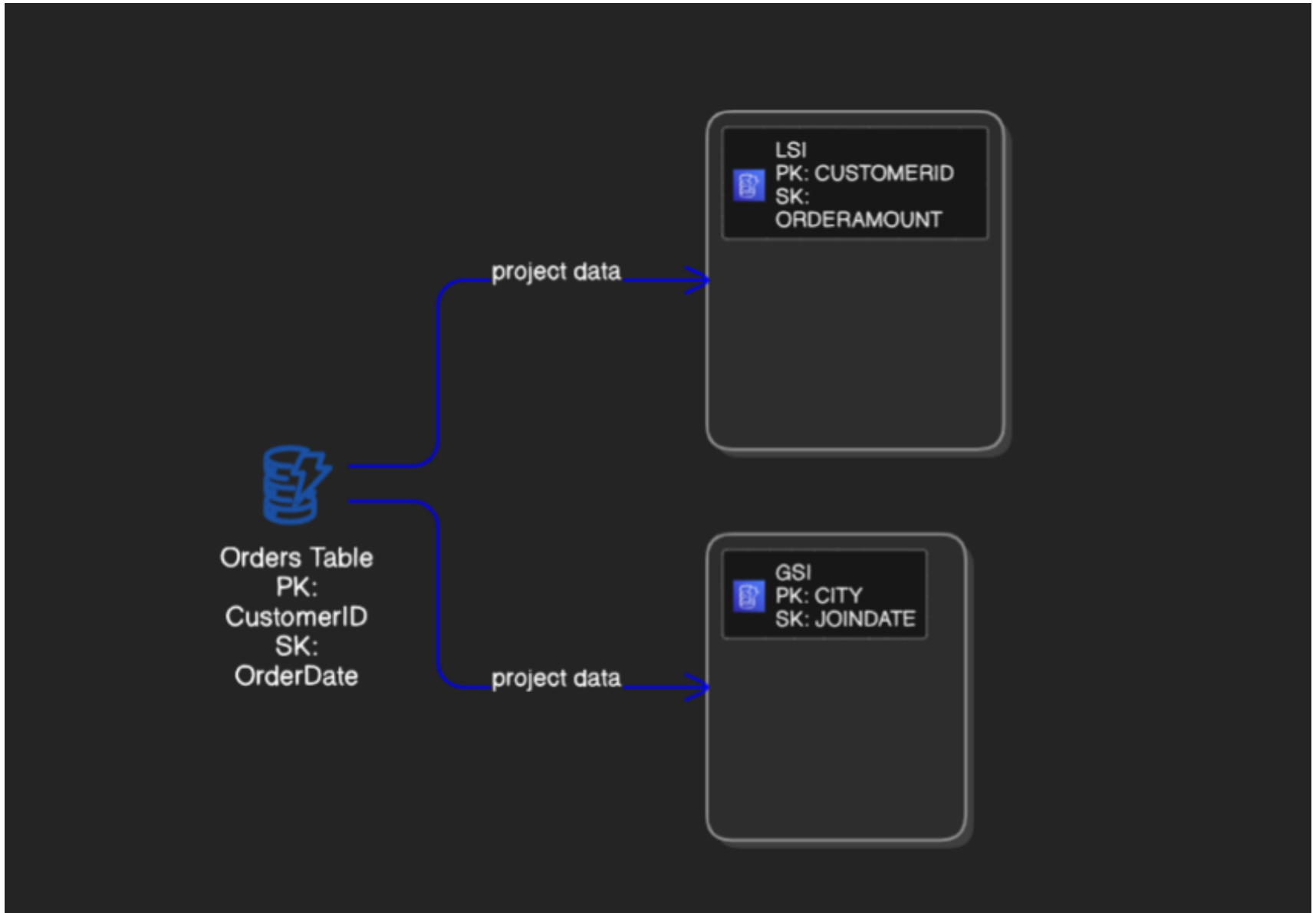
Global Secondary Index (GSI)

- Có Partition Key và Sort Key **hoàn toàn khác** với bảng chính
- Trải rộng trên **tất cả dữ liệu** của bảng gốc
- Cho phép truy vấn dựa trên **thuộc tính không phải khóa chính**
- Chỉ hỗ trợ **Eventually Consistent Read**
- Giới hạn: **tối đa 20 GSI** mỗi bảng (mặc định)
- Có thể tạo hoặc xóa **bất kỳ lúc nào** sau khi bảng đã được tạo
- Có các thiết lập **RCU/WCU riêng biệt** và độc lập



Khi nào nên sử dụng LSI: Khi bạn cần truy vấn dữ liệu theo nhiều cách sắp xếp khác nhau cho cùng một Partition Key |**Khi nào nên sử dụng GSI:** Khi bạn cần truy vấn dữ liệu dựa trên các thuộc tính không phải là khóa chính của bảng gốc

Secondary Indexes



Secondary Indexes

⚙ Use Case 1 – Local Secondary Index (LSI)
Bối cảnh: Bạn có bảng `Orders` lưu đơn hàng của khách hàng.

Partition Key	Sort Key	Attributes
`CustomerID`	`OrderDate`	`OrderAmount`, `Status`

Vấn đề:
Bạn muốn **xem lại các đơn hàng của 1 khách hàng**, nhưng **sắp xếp theo giá trị đơn hàng (OrderAmount)** thay vì ngày.
Giải pháp:
Tạo **LSI** như sau:
•Partition Key: `CustomerID` (giống bảng chính)
•Sort Key: `OrderAmount`
→ Khi truy vấn:

```sql
Query CustomerID = 12345 ORDER BY OrderAmount DESC
```

→ DynamoDB sẽ lấy dữ liệu từ LSI mà **không cần quét toàn bộ bảng**.
LSI Use Case thực tế:
•Tìm đơn hàng lớn nhất của từng khách hàng
•Phân tích giá trị mua trung bình của một khách hàng
•Sắp xếp dữ liệu theo thứ tự khác trong cùng nhóm dữ liệu

Secondary Indexes

⚙ Use Case 2 – Global Secondary Index (GSI)
Bối cảnh:
Vẫn là bảng `Orders` ở trên, nhưng giờ bạn muốn **xem tất cả đơn hàng theo thành phố** hoặc **tìm khách hàng đăng ký gần đây**.

Partition Key	Sort Key	Attributes
CustomerID	OrderDate	City, JoinDate, OrderAmount

Giải pháp:
Tạo **GSI** như sau:
•Partition Key: `City`
•Sort Key: `JoinDate`
→ Khi truy vấn:

```sql
Query City = "Hanoi" ORDER BY JoinDate DESC
```

→ Bạn sẽ nhanh chóng lấy được **tất cả khách hàng ở Hà Nội**, được sắp xếp theo **ngày đăng ký mới nhất**, mà **không cần quét toàn bộ bảng chính**.
GSI Use Case thực tế:
•Tìm khách hàng theo khu vực (City, Region)
•Lọc đơn hàng theo trạng thái (`Status = Delivered`)
•Xây dựng trang tổng quan thống kê toàn hệ thống

Query và Scan

Query

Tìm kiếm các item dựa trên Partition Key (bắt buộc) và tùy chọn Sort Key (hoặc điều kiện trên Sort Key).

- **Hiệu quả cao:** Truy cập trực tiếp vào các phân vùng chứa dữ liệu có Partition Key đó.
- **Nhanh chóng:** Không cần phải đọc toàn bộ bảng, chỉ truy cập các phân vùng cần thiết.
- **Chi phí thấp:** Tiêu thụ ít RCU hơn so với Scan.
- **Giới hạn:** Chỉ tìm kiếm theo Partition Key hoặc phạm vi Sort Key.

Scan

Đọc tất cả các item trong toàn bộ bảng (hoặc chỉ mục phụ), sau đó áp dụng dụng các bộ lọc nếu có.

- **Tốn kém:** Tiêu thụ nhiều RCU và có thể mất nhiều thời gian để hoàn thành.
- **Gây tắc nghẽn:** Có thể làm giảm hiệu năng của các yêu cầu khác.
- **Linh hoạt:** Có thể áp dụng bộ lọc phức tạp trên toàn bộ bảng.
- **Không khuyến nghị:** Dành cho mục đích thử nghiệm hoặc với các bảng nhỏ.

Best Practice:

Luôn ưu tiên sử dụng Query thay vì Scan khi có thể, đặc biệt với các bảng lớn. Thiết kế Partition Key phân tán tốt để tránh " hot partition".

⚙□ Streams

DynamoDB Streams ghi lại mọi thay đổi dữ liệu (INSERT, MODIFY, REMOVE) và cho phép bạn xây dựng các pipeline xử lý sự kiện, audit trail và phân tích.

Lưu trữ 24 giờ

Dữ liệu được giữ lại trong 24 giờ, đủ thời gian để xử lý các sự kiện và xây dựng audit trail.

Kích hoạt Lambda

Tự động kích hoạt các hàm Lambda dựa trên các thay đổi dữ liệu, hỗ trợ xử lý sự kiện tức thì.

Multi-AZ Replication

Hỗ trợ replication đa vùng, đảm bảo dữ liệu được phân phối đều và khả dụng cao.

Ứng dụng của Streams

Audit Trail — Analytics — Replication — Event Processing

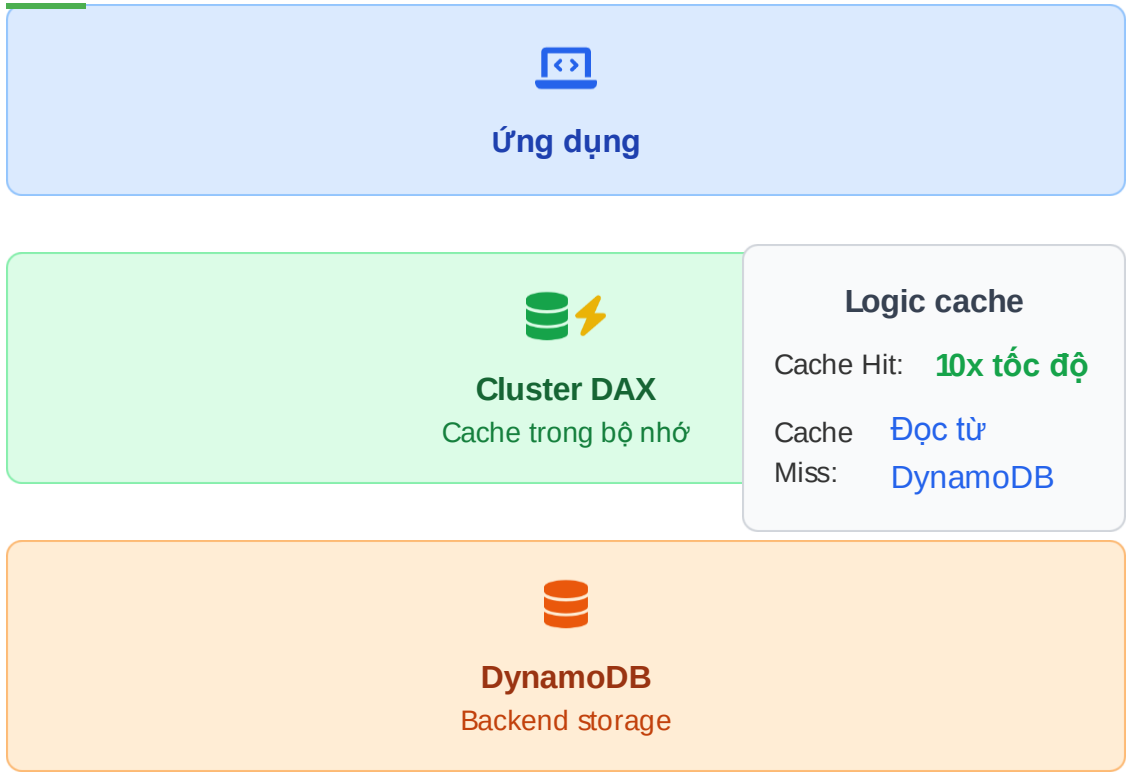
⚡ DynamoDB Accelerator (DAX)

DynamoDB Accelerator (DAX) là một giải pháp cache trong bộ nhớ hiệu quả, được thiết kế để tăng tốc độ đọc và giảm độ trễ xuống mức microsecond.

Tính năng nổi bật

- Tăng tốc độ đọc lên đến **10 lần** so với DynamoDB thông thường
- Giảm độ trễ đọc xuống mức **microsecond**
- Tương thích hoàn toàn với DynamoDB API
- Cluster gồm nhiều node cache được quản lý tự động

Cách DAX hoạt động



Security và IAM

Chính sách IAM

Kiểm soát chi tiết ai có thể truy cập tài nguyên DynamoDB của bạn và họ có thể làm gì. Cho phép cấp quyền truy cập cụ thể đến bảng, mục hoặc hành động.

```
IAM Policy Example:
{"Effect": "Allow", "Action":
"dynamodb:ReadItem", "Resource": "*"}
```

Mã hóa dữ liệu

DynamoDB tự động mã hóa tất cả dữ liệu khi lưu trữ bằng AWS Key Management Service (KMS), đảm bảo dữ liệu của bạn được bảo vệ ngay cả khi lưu trữ trên đĩa.

```
Encryption at Rest:
Automatic with AWS KMS
```

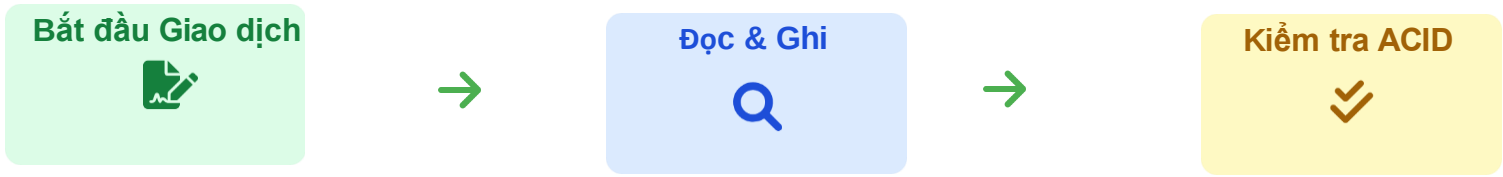
Điểm cuối VPC

Sử dụng các điểm cuối VPC để giới hạn quyền truy cập vào DynamoDB chỉ từ bên trong Mạng riêng ảo (VPC) của bạn, giúp ngăn chặn truy cập trái phép từ internet.

```
VPC Endpoint:
vpc-
endpoint.dynamodb.region.amazonaws.com
```

Transactions

DynamoDB hỗ trợ các giao dịch (transactions), cho phép bạn thực hiện nhiều thao tác đọc hoặc ghi như một đơn vị nguyên tử (atomic unit), đảm bảo tính nhất quán của dữ liệu ngay cả khi các thao tác liên quan đến nhiều mục hoặc bảng khác nhau.



Tính Nguyên Tử

Các giao dịch đảm bảo rằng tất cả các thao tác trong nhóm sẽ thành công hoặc tất cả sẽ thất bại, giữ cho dữ liệu luôn nhất quán.

Tuân thủ ACID

Các giao dịch của DynamoDB cung cấp tính nguyên tử, nhất quán, cô lập và bền vững (ACID), ngay cả đối với các nhóm mục nằm trên các Partition Key khác nhau.

Giới hạn

Một giao dịch có thể hỗ trợ tối đa 25 thao tác (operations). Điều này bao gồm cả đọc và ghi.

TTL (Time to Live)

Tính năng tự động xóa các item khỏi bảng sau một khoảng thời gian xác định

Quản lý vòng đời dữ liệu

Tự động xóa dữ liệu cũ, giữ cho bảng luôn gọn gàng và hiệu suất cao

Tiết kiệm chi phí lưu trữ

Loại bỏ dữ liệu cũ tự động, giảm chi phí lưu trữ và cải thiện hiệu năng

Cách thức triển khai

Chỉ định một thuộc tính (ví dụ: expireAt) để lưu trữ thời điểm hết hạn

```
item = {
    "id": "123",
    "expireAt": 1634567890 // Unix timestamp
}
```

Trường hợp sử dụng

- Phiên người dùng (user sessions)
- Nhật ký (logs)
- Dữ liệu cảm biến (sensor data)

SDK Integration

Hỗ trợ đa ngôn ngữ

Python boto3	Java aws-java-sdk	Node.js aws-sdk
.NET AWSSDK.DynamoDBv2	Go aws-sdk-go	Ruby aws-sdk-dynamodb

Request & Response

Request

```
table.put_item(
    Item={
        'id': '123',
        'name': 'John'
    }
)
```

Response

```
response = table.get_item(
    Key={'id': '123'}
)
item = response['Item']
```

Cấu hình cơ bản

```
import boto3

# Cấu hình AWS credentials và region
dynamodb = boto3.resource('dynamodb', region_name='us-west-2')
```

- aws configure hoặc AWS credentials file
- Region: us-west-2, eu-central-1, ap-southeast-1...

Tính năng nâng cao

Pagination

Định dạng kết quả lớn thành nhiều trang

Batch Write

Ghi nhiều item cùng lúc

Kết nối DynamoDB

AWS SDK

Sử dụng các SDK được cung cấp cho ngôn ngữ lập trình của bạn để tương tác trực tiếp với DynamoDB từ ứng dụng.

AWS CLI

Giao diện dòng lệnh AWS (CLI) cho phép bạn thực hiện các thao tác quản lý và dữ liệu trên DynamoDB thông qua các lệnh terminal.

AWS Console

Bảng điều khiển quản lý AWS cung cấp giao diện người dùng đồ họa để tạo, quản lý và giám sát các bảng DynamoDB của bạn.

AWS Lambda

DynamoDB tích hợp trực tiếp với AWS Lambda, cho phép bạn kích hoạt các hàm Lambda dựa trên các thay đổi dữ liệu trong DynamoDB Streams.

Môi trường phát triển local

Để phát triển và thử nghiệm cục bộ, bạn có thể sử dụng DynamoDB Local hoặc các công cụ như LocalStack, cung cấp một endpoint URL tương thích với API của DynamoDB.

Best Practices

Thiết kế Partition Key phân tán tốt

Chọn Partition Key có tính phân tán cao để tránh "hot partition" – tình trạng một partition nhận quá nhiều yêu cầu, dẫn đến tắc nghẽn và giảm hiệu suất.

Dùng GSI thay vì Scan toàn bảng

Global Secondary Index (GSI) cho phép truy vấn dữ liệu theo các thuộc tính không phải là khóa chính một cách hiệu quả, tránh sử dụng thao tác Scan trên toàn bộ bảng.

Bật TTL cho dữ liệu tạm thời

Kích hoạt tính năng Time to Live (TTL) để tự động xóa các item không còn cần thiết sau một khoảng thời gian xác định, giúp giảm chi phí lưu trữ và giữ cho bảng gọn gàng.

Dùng DAX cho workload đọc nặng

Đối với các ứng dụng có khối lượng đọc lớn và yêu cầu độ trễ cực thấp (microsecond), DynamoDB Accelerator (DAX) là một giải pháp cache trong bộ nhớ hiệu quả, có thể tăng tốc độ đọc lên đến 10 lần.

Giám sát qua CloudWatch metrics

Thường xuyên theo dõi các chỉ số quan trọng trên Amazon CloudWatch như Consumed RCU/WCU (đơn vị đọc/ghi đã tiêu thụ) và Throttled Requests (yêu cầu bị điều tiết) để phát hiện sớm các vấn đề về hiệu suất.

Chọn đúng chế độ dung lượng

Lựa chọn giữa Provisioned Mode (chế độ cấp phát trước) và On-Demand Mode (chế độ theo yêu cầu) dựa trên đặc điểm workload của ứng dụng để tối ưu chi phí và hiệu năng.

Use Cases điển hình

Ứng dụng IoT

- Lưu trữ dữ liệu cảm biến theo thời gian thực từ hàng triệu thiết bị IoT
- Xử lý lượng lớn dữ liệu ghi và đọc liên tục với độ trễ thấp
- Lý tưởng cho các hệ thống giám sát và phân tích IoT

Ứng dụng Mobile/Web

- Quản lý dữ liệu người dùng, phiên làm việc (session), hồ sơ (profile)
- Token xác thực và các cài đặt cá nhân
- Khả năng mở rộng linh hoạt để phục vụ hàng triệu người dùng

Gaming backend

- Hỗ trợ các bảng xếp hạng (leaderboard), trạng thái người chơi (player state)
- Dữ liệu vật phẩm trong game và lịch sử giao dịch
- Hiệu suất cao cần thiết cho trải nghiệm chơi game mượt mà và phản

Financial & eCommerce

- Lưu trữ thông tin đơn hàng, giỏ hàng, lịch sử giao dịch
- Dữ liệu sản phẩm và quản lý hàng tồn kho
- Khả năng xử lý giao dịch nguyên tử (transactions) và độ bền dữ liệu cao

Tổng kết

Key Strengths

- High performance & low latency
- Serverless architecture
- Managed service by AWS
- Infinite scalability

Notable Features

- Flexible indexing (LSI, GSI)
- In-memory caching (DAX)
- Data streaming & TTL
- ACID transactions

Best Applications

- IOT sensor data
- Mobile & web backends
- Gaming leaderboards
- Financial & e-commerce

> “Amazon DynamoDB là lựa chọn hàng đầu cho ứng dụng quy mô lớn và thời gian thực với khả năng mở rộng tự động, hiệu năng cao và độ trễ thấp.”

Cảm ơn bạn đã tham gia!