**Dynamo DB**

**+ Giới thiệu về DynamoDB:**

DynamoDB là một dịch vụ cơ sở dữ liệu quản lý hoàn toàn sử dụng NoSQL và là một cơ sở dữ liệu Key-Value được cung cấp bởi Amazon như một phần của Amazon Web Services. DynamoDB được Amazon công bố vào ngày 18 tháng 1 năm 2012 và được trình bày dưới dạng một sự phát triển của giải pháp Amazon SimpleDB.

Hiện nay Dynamo hổ trợ đa dạng các ngôn ngữ khác nhau để phù hợp với từng đối tượng khách hàng và người sử dụng. Các ngôn ngữ liên kết DynamoDB bao gồm Java, Node.js, Go, C# .NET, Perl, PHP, Python, Ruby, Haskell, …

**+ Kiến trúc của DynamoDB:**

DynamoDB được thiết kế để đạt được sự ổn định cao trong lúc xử lý các truy vẫn, có đồ bến tốt và tính khả dụng cao phù hợp với nhiều đối tượng người sử dụng. DynamoDB cũng đượ thiết kế làm sao để gia tăng hiệu suất của người sử dụng luôn đạt ở mức cao nhất để từ đó có được độ trể thấp trong việc tiếp nhận và phản hồi truy vấn cũng như đồng bộ hóa cơ sở dữ liệu.

Amazon DynamoDB được xây dựng để hoạt động trên một cụm máy chủ đươc AWS (Amazon Web Services) quản lý để tạo ra môt nền tảng lưu trử lớn với mật độ cao cùng với đó là tối ưu hóa phù hợp với nhiều người dùng. Nền tảng này giúp cải thiệt hiệu suất làm việc của DynamoDB và khiến nó không phụ thuộc vào độ lớn của bảng dữ liệu.

Đối với DynamoDB kiến trúc của nó có 2 cơ chế đặc biệt giúp dynamoDB có thể cải thiện hiệu suất và nâng cao sự ổn định của mình đó chính là cơ chế Partitioning (phân vùng) và Replication (nhân bản)

+ Partitioning (phân vùng):

DynamoDB sử dụng một hệ thống máy chủ và mỗi máy chủ chịu trách nhiệm cho việc lưu trử một phần dữ liệu của người dùng. Bằng cách sử dung hàm băm (hash function) để chia các key của dữ liệu (DynamoDB sử dụng hệ thống dữ liệu key – value) để chia các loại dữ liệu ứng với giá trị hash của key vào từng máy chủ thích hợp. Và khi giá trị băm của một vùng nào đó bắt đầu trở nên quá lớn và bị thiếu không gian lưu trử ta chỉ cần cấu hình thêm một máy chủ mới vào vùng giá đấy để giảm tải bớt cho các máy chủ ban đầu. Việc này cũng giúp ta dể dàng gộp dữ liệu lại nếu như có một vùng dữ liệu nào đó bị xóa đi hoặc trống nhiều nhằm tiết kiệm tài nguyên.

Từ những điều trên ta có thể nhận thấy rắng với thiết kế Partition chúng ta sẽ không bị giới hạn về quy mô của của DynamoDB và ta luôn có thể mở rộng nó. Và việc tìm kiếm dữ liệu cũng không hề bị ảnh hưởng khi nó được thực hiện thông qua cơ chế Key – Value còn việc phân vùng cho dữ liệu chỉ dựa trên giá trị hash của key và hoàn toàn không gây biến đổi key của dữ liệu.

+ Replication (nhân bản):

Đối với một cơ sở dữ liệu việc chỉ lưu trử dử liệu duy nhất trên một máy chủ là vô cùng nguy hiểm vì nếu như máy chủ bị hư hòng hoặc tạm thời bị tắt thì việc truy xuất sẽ bị đình trệ và khi mất mát dữ liệu thì khồng thể khôi phục được. Từ đó DynamoDB đã tạo ra một cụm máy chủ với chức năng lưu trử các bản sao của dữ liệu lưu trử ở DynamoDB. Cơ chế Replication yêu cầu máy chủ ghi nhận lại các dữ liệu của máy chủ trong cùng một cụm với nhau. Khi có một request được gửi lên nhằm lấy dử liệu thông qua cơ chế Key – Value thì máy chủ đẩu tiên tiếp nhận sẽ gửi yêu cầu kiểm tra cặp key – value đấy đến các máy chủ cùng cùm để đảm bảo rằng cặp key – value tồn tại từ đó phản hồi cho người dùng. Tuy nhiên việc này sẽ gây mất thời gian nếu như có nhiều máy chủ trong 1 cụm nên để rút ngắn thời gian thường chỉ cần 1 đến 2 máy chủ phàn hồi thì ngay lập tức response cho người dùng về giá trị của cặp key-value.

**+ Các đặc điểm nổi bật của DynamoDB:**

DynamoDB được xây dựng với các đặc điểm chính như sau:

+ Cung cấp một cơ sở dữ liệu mà người dùng không cần phải tự duy trì Database (DynamoDB sẽ hổ trơ việc duy trì).

+ Tự động xây dựng phần cứng phù hợp với kích thước của dử liệu.

+ Hổ trợ việc xây dựng một liên kết với độ trể thấp.

+ Dể dàng lấy lại dữ liệu trong trường hợp bị mất mát.

+ Giúp người dùng dễ dàng sử dụng và mở rộng Database của họ.

+ Tiếp cận được với nhiều đối tượng với giá thành hợp lý.

**+ Các tính năng của hệ quản trị CSDL Dynamo DB:**

Dynamo DB hổ trợ phần lớn tất cả cá tính năng quan trọng mà một hệ quản trị CSDL cần có. Và được xây dựng để người dùng có thể dể dàng sử dụng cũng như khắc phục khi xảy ra lổi trong hệ thống.

1. **Transactions (Giao tác):**

Đối với transaction trong Dynamo DB người dùng có thể dể dàng gom nhóm nhiều hành động và gửi đi cùng một lúc bằng 2 hàm được xây dựng sẵn là TransactWriteItems và TransactGetItems.

**+ TransacWriteItems:**

TransacWriteItems là một hàm được xây dựng dưới dạng synchronous và idempotent có nghĩa là đồng bộ và không cố định. Nó có thể gom nhóm tối đa 25 mục khác nhau trong một hoặc nhiều bảng trong Database. TransacWriteItem được xây dựng để khi thực hiện thì toàn bộ transaction đều thành công hoặc tất cả sẽ đều không thành công.

Đối với hàm TransacWriteItems người dùng có thể dể dàng thêm những hành động sau đây là vào 1 transaction:

+ Put: Thêm một đối tượng mới hoặc thay đổi một đối tượng cũ bằng một đối tượng mới trong cơ sở dữ liệu. Và người dùng có thể thêm điều kiện để thực hiện nó hoặc không.

+ Update: Điều chỉnh thuộc tính của một đối tượng hoặc tạo một đối tượng mới nếu nó chưa tồn tại trong cơ sở dữ liệu. Người dùng có thể thêm điều kiện để hàm thực hiện các công việc khác như thêm, xóa, sửa chữa thông tin ở đối tượng đấy.

+ Delete: Xóa một đối tượng khỏi cơ sở dữ liệu.

+ ConditionCheck: Kiểm tra sự tồn tại của một đối tượng hoặc một thuộc tính bên trong nó.

Ngoài ra với TransacWriteItem người dùng có thểm tạo thêm một token ở mỗi Transaction để khi có sự cố về kết nối xảy ra thì khi một lệnh gọi TransacWriteItem thành công thì các lệnh khác chỉ trả về đã thực hiện thành công và không ghi vào cơ sở dử liệu.

DynamoDB quy định những trượng hợp sau thì việc ghi Transaction sẽ không thành công:

+ Khi một điều kiện trong các điều kiện được áp dụng không đáp ứng

+ Khi lỗi xác thực xảy ra cho thấy rằng có nhiều hành động xảy ra đồng thời trên 1 đối tượng.

+ Khi một TransacWriteItem gây xung đột với một TransacWriteItem khác.

+ Khi dung lượng hiện tại của người dùng không đủ thể thêm thực hiện.

+ Khi dung lượng của 1 đối tượng trở lên quá lớn.

+ Khi xuất hiện lổi từ người dùng (Ví dụ: như chọn sai kiểu dữ liệu, …)

**+ TransacGetItem:**

Tương tự với TransacWriteItem, TransacGetItem cũng là một hàm đồng bộ và có thể thực hiện hành động của tối đa 25 đối tượng từ một hoặc nhiều bẳng tròn DynamoDB.

TransacGetItem hổ trợ hàm Get truy xuất dử liệu cho một khóa chính được cung cấp. Nếu như không tìm thấy khóa chính nó sẽ không trả về dữ liệu gì.

DynamoDB quy định những trượng hợp sau thì việc truy xuất dữ liệu sẽ không thành công:

+ Khi một TransacGetItem gây xung đột với một TransacWriteItem.

+ Khi xuất hiện lổi từ người dùng (Ví dụ: như chọn sai kiểu dữ liệu, …)

+ Khi dung lượng hiện tại của người dùng khồn đủ để thực hiện transaction.

1. **Triggers:**

Đối với DynamoDB nó được Amazon tích hợp với AWS Lambda nhờ đó người dùng có thể dể dàng tạo ra các trigger. Với các trigger được xây dựng người dùng có thể dể dang xây dựng các phản hồi tự động ứng với các hành động lên cơ sở dữ liệu.

1. **Lưu trử và Phục hồi dữ liệu trong DynamoDB (Backup and Restore):**

Để thực hiện tạo bản sao và lưu trử bản sao trên DynamDB ta cần làm các bước sau:

1. Đăng nhập vào hệ thống AWS Management Console và mở bảng điều khiển DynamoDB.
2. Tại mục **Backup** của 1 bảng ta muốn backup ta nhập tên và nhấn **Create Backup** để tạo backup.

Graphical user interface, text, application, email

Description automatically generated

1. Sau đó hệ thống sẽ xử lý việc tạo Backup và hoàn thành khi trạng thái chuyển sang Available.

Graphical user interface, text, application, email

Description automatically generated

Để thực hiện việc phục hồi dữ liệu từ backup Table ta làm như sau:

1. Đăng nhập vào hệ thống AWS Management Console và mở bảng điều khiển DynamoDB.
2. Trên thanh công cụ ta chọn **Backup**
3. Chọn backup Table ứng với Table muốn khôi phục.

Graphical user interface, application

Description automatically generated with medium confidence

1. Chọn **Restore**.
2. Nhập tên bảng sau khi Restore sau đó chọn **Restore Table**.

Graphical user interface, text, application, email

Description automatically generated

1. **Bảo mật:**

DynamoDB được thiết để lưu trử dữ liệu với mật độ cao và được sao liêu ở trên nhiều máy chủ khác nhau ở nhiều khu vực thuộc sự quản lý cảu Amazon DynamoDB. DynamoDB bảo vệ dữ liệu của người dùng dù ở bất kỳ trạng thái nào, trạng thái nghỉ hay đang trên đường truyền giữa các máy chủ với nhau, giữa máy chủ và máy khách.

1. Định danh và truy cập vào Amazon Management:

Để có thể sử dụng DynamoDB thì người dùng cần phải truy cập vào AWS Management Console. IAM administrators chịu trách nhiệm kiểm soát việc xác thực người dùng và phân quyền của dùng trên cơ sở dữ liệu để kiểm tra xem ai là người có quyền truy cập cũng như chỉnh sữa dữ liệu. Cũng như triển khai các chính sách bảo mật của Amazon đến người dùng để bảo vệ tốt hơn dử liệu.

1. Bảo mật dữ liệu:

Amazon DynamoDB bảo mật dữ liệu bằng tiêu chuẩn mã hóa nâng cao 256-bit (AES-256), giúp bảo mật dữ liệu của khỏi bị truy cập trái phép. Khi dữ liệu đang ở trang thái nghỉ nó được AWS Key Management Service (AWS KMS) quản lý và mã hóa dử liệu. Ngoài ra DynamoDB cho phép người dùng chọn lựa loại khóa mã hóa phù hợp với dữ liệu của bản thân với 3 lựa chọn là: AWS owned key (khóa cơ bản được quản lý bởi DynamoDB), AWS managed key (khóa được quản lý bởi AWS KMS), Customer managed key (Khóa do chính người dùng quản lý).

1. **Tham Khảo:**

+ <https://viblo.asia/p/gioi-thieu-ve-dynamodb-phan-1-jamoG87nMz8P>

+ <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction>

+ <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/data-protection.html>

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

+ <https://medium.com/swlh/aws-dynamodb-triggers-event-driven-architecture-61dea6336efb> (Dùng VPN để truy cập)

+ <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/transaction-apis.html>

+ <https://www.alexdebrie.com/posts/dynamodb-transactions/>