

**SAMSUNG**

# Samsung Innovation Campus

| Khoá học Big Data

Chương 4.

# Lưu trữ Big Data

Khoá học Big Data

# Mô tả chương

---

## 📌 Mục tiêu:

- ✓ Chúng ta sẽ tìm hiểu cách sử dụng các thiết bị lưu trữ khác ngoài HDFS
  - HDFS EC cho phép chúng ta đạt được cùng một mức độ chịu lỗi với khoảng một nửa chi phí sử dụng đĩa
  - Kudu cho phép các ứng dụng có cả đặc tính thông lượng cao và độ trễ nhanh
  - Ngoài ra, về lưu trữ tiền đề, có nhiều tùy chọn lưu trữ đám mây công cộng có sẵn
- ✓ We will learn what NoSQL is and how it differs from conventional relational database management systems
  - Apache HBase là một trong những công cụ NoSQL sớm nhất và được tích hợp chặt chẽ với Hadoop
  - Các công cụ NoSQL thường được phân loại là "nhất quán mạnh mẽ" hoặc "có tính sẵn sàng cao"
  - Chúng ta sẽ tìm hiểu kiến thức cơ bản về Apache Cassandra, chủ yếu là cơ sở dữ liệu NoSQL có tính sẵn sàng cao
- Để nhất quán mạnh mẽ, chúng ta sẽ tìm hiểu kiến thức cơ bản về Apache MongoDB

## 📌 Nội dung:

1. Giải pháp thay thế lưu trữ dữ liệu
2. NoSQL

Bài 1.

# Giải pháp thay thế lưu trữ dữ liệu

Lưu trữ Big Data

Bài 1.

# Giải pháp thay thế lưu trữ dữ liệu

| 1.1. Lưu trữ tại chỗ

| 1.2. Lưu trữ đám mây công cộng

# Khi Hadoop còn trẻ

- I Khi Hadoop được giới thiệu như một nền tảng và phương pháp để làm việc và xử lý lượng dữ liệu khổng lồ, nó đã làm như vậy bằng cách giới thiệu hai chức năng cốt lõi
  - ▶ Hệ thống tệp phân tán Hadoop để lưu trữ
  - ▶ MapReduce do YARN quản lý để tính toán
- I Đồng thời, Web 2.0 đang được triển khai tốt và cần một mô hình cơ sở dữ liệu mới để xử lý các yêu cầu thay đổi
  - ▶ Các trang web thế hệ thứ hai nhấn mạnh nội dung do người dùng tạo
  - ▶ Nội dung được chia sẻ giữa những người tham gia và công chúng
  - ▶ Tính năng dễ sử dụng
  - ▶ NoSQL, một thuật ngữ được đặt ra lần đầu tiên vào năm 1998, được đề xuất thay thế cho RDBMS để xử lý Web 2.0
  - ▶ Thông số kỹ thuật NoQuery hiện tại đã hoàn thiện vào khoảng năm 2009

# Hadoop HDFS

## I Hiệu quả:

- ▶ Quét hiệu quả lượng lớn dữ liệu
- ▶ Tích lũy dữ liệu với thông lượng cao
- ▶ Nhiều tùy chọn SQL
- ▶ Tất cả các động cơ xử lý

## I Hạn chế:

- ▶ Truy cập một hàng có vấn đề
- ▶ Đột biến có vấn đề
- ▶ Truy cập "Dữ liệu nhanh" có vấn đề



# Apache HBase - the Hadoop NoSQL

## I Hiệu quả:

- ▶ Tìm và viết hiệu quả các hàng riêng lẻ
- ▶ Tích lũy dữ liệu với thông lượng cao

## I Hạn chế:

- ▶ Quét có vấn đề
- ▶ Khả năng tiếp cận cardinality cao là vấn đề
- ▶ Hỗ trợ SQL không thật sự tốt do những điều trên





# Hadoop trưởng thành

- I Hadoop trưởng thành và các trường hợp sử dụng mới xuất hiện
- I Các ứng dụng truyền thống tách riêng việc xử lý dữ liệu lịch sử và dữ liệu thời gian thực
  - ▶ Xử lý hàng loạt dữ liệu lịch sử được lưu trên bộ lưu trữ HDFS
  - ▶ Truy cập và sửa đổi các điểm dữ liệu truy cập ngẫu nhiên trong thời gian thực bằng cách sử dụng các công cụ xử lý luồng với dữ liệu được lưu trữ trên NoQuery, chẳng hạn như HBase
- I Kiến trúc Lambda
  - ▶ Nhiều trường hợp sử dụng yêu cầu phân tích dữ liệu lịch sử được kết hợp với dữ liệu phát trực tuyến hiện tại được cập nhật
  - ▶ Kiến trúc Lambda đề xuất hai ứng dụng và hệ thống riêng biệt để xử lý việc này

Độ trễ nhanh

APACHE  
HBASE

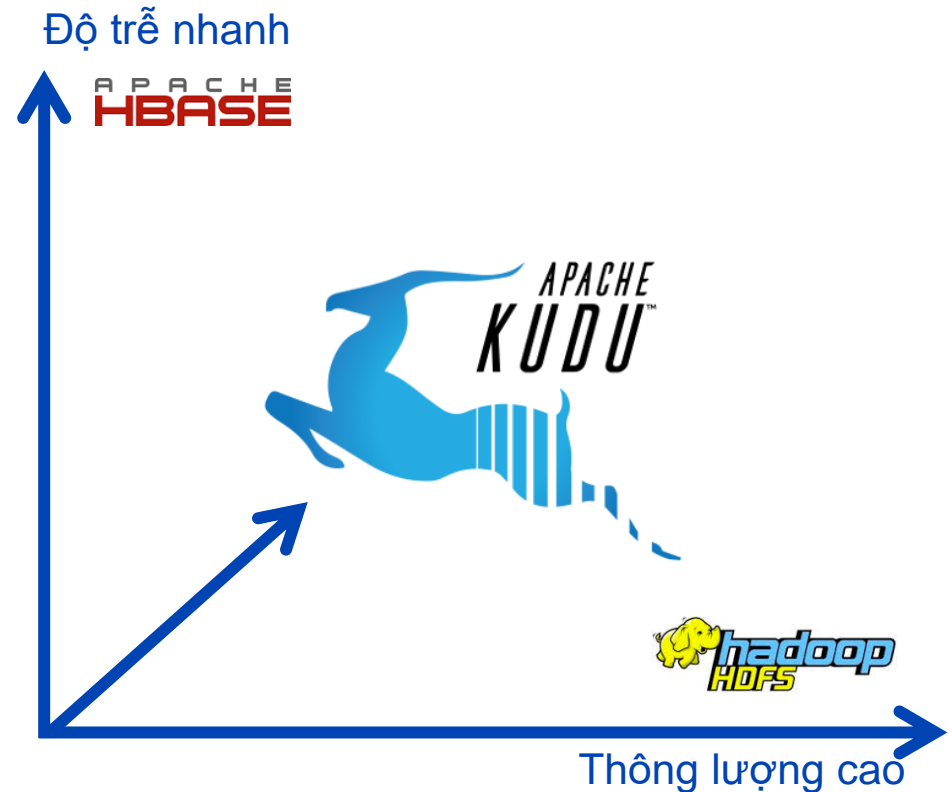
Chúng ta có thể tạo bộ lưu trữ thực hiện cả hai không?



Thông lượng cao

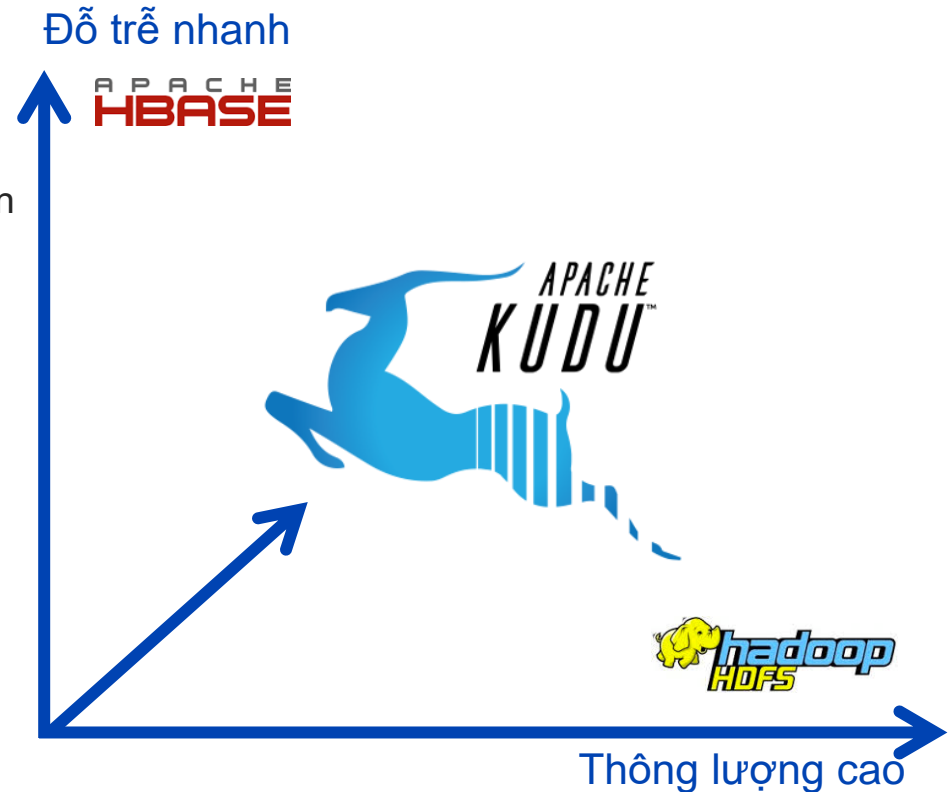
# Mục tiêu thiết kế của Kudu để đáp ứng khoảng cách (1/2)

- | Thông lượng cao cho các bản quét lớn
- | Độ trễ thấp cho truy cập ngẫu nhiên
- | Hiệu suất CPU cao để tận dụng RAM và Flash tốt hơn
- | Hiệu suất IO cao
  - ▶ Lưu trữ cột thực với mã hóa theo loại cụ thể
  - ▶ Phân tích hiệu quả khi chỉ một số cột nhất định được truy cập
- | Mô hình dữ liệu biểu cảm và có thể phát triển
- | Kiến trúc hỗ trợ vận hành đa trung tâm dữ liệu



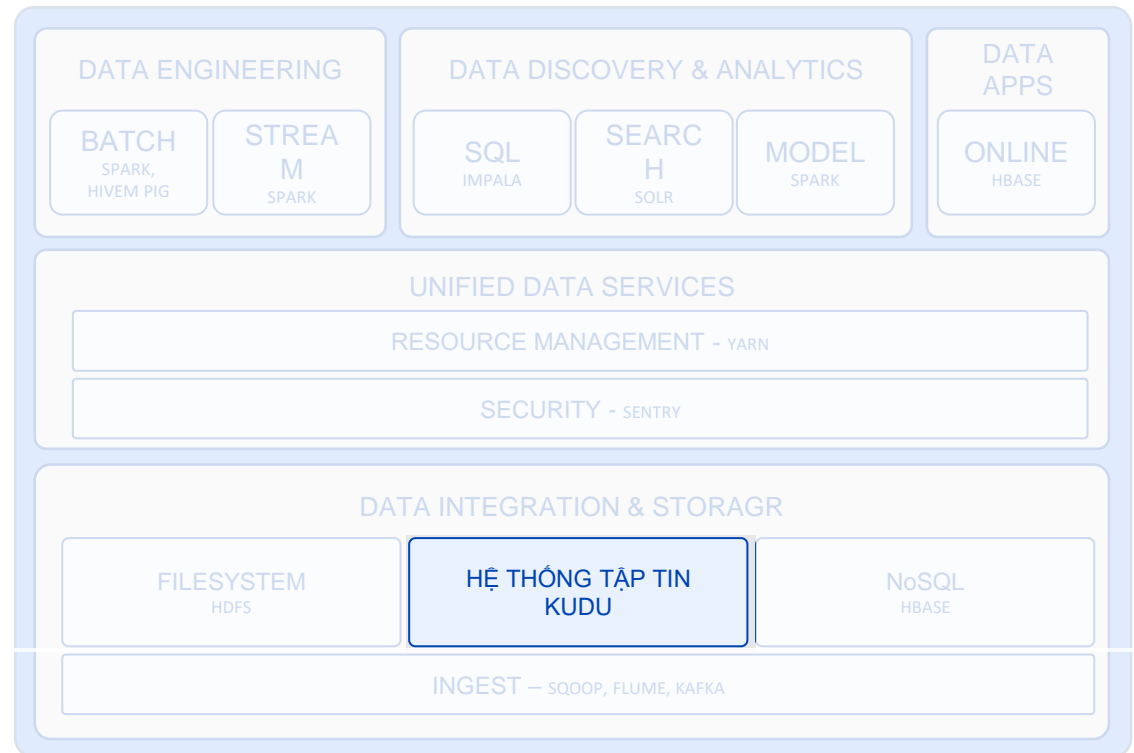
## Mục tiêu thiết kế của Kudu để đáp ứng khoảng cách (2/2)

- | Xử lý nhanh khối lượng công việc OLAP
- | Hiệu năng mạnh mẽ để chạy đồng thời khối lượng công việc tuần tự và ngẫu nhiên
- | Tích hợp với MapReduce, Spark và các thành phần hệ sinh thái Hadoop khác
- | Tích hợp chặt chẽ với Apache Impala
- | Mô hình nhất quán mạnh mẽ nhưng linh hoạt
  - ▶ Chọn yêu cầu về tính nhất quán của cơ sở theo yêu cầu
- | Khả dụng cao và khả năng chịu lỗi
  - ▶ Tiếp tục cung cấp khả năng đọc/ghi khi gặp lỗi



# Vị trí của Kudu trong Hệ sinh thái Hadoop

- I Kho lưu trữ cột có thể cập nhật cho Hadoop
- I Mã nguồn mở apache

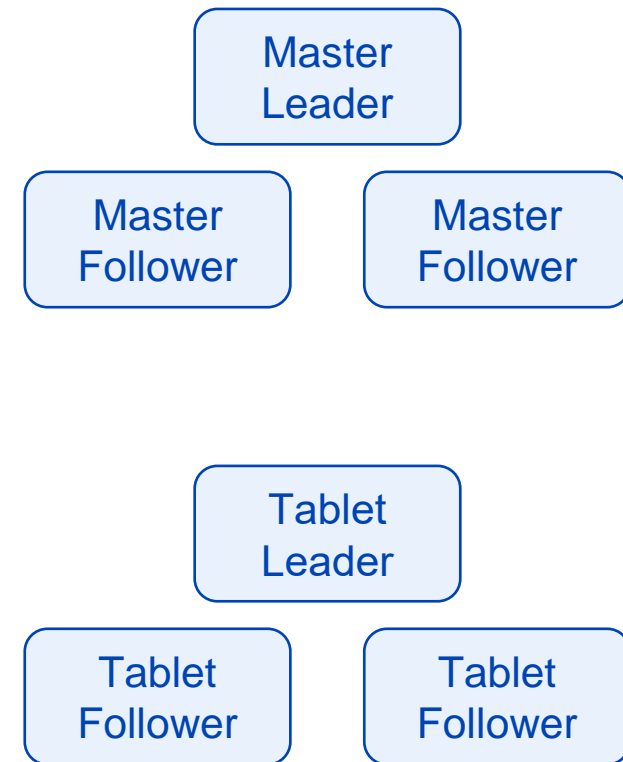


# Kudu KHÔNG PHẢI LÀ GÌ

- I Bản thân nó không phải là giao diện SQL
  - ▶ Nó chỉ là lớp lưu trữ
  - ▶ Mang theo SQL của riêng bạn (ví dụ: Impala hoặc Spark)
- I Không phải là một ứng dụng chạy trên HDFS
  - ▶ Đó là một công cụ lưu trữ Hadoop gốc, thay thế
  - ▶ Thuê vị trí với HDFS dự kiến
- I Không phải là sự thay thế cho HDFS hoặc HBase
  - ▶ Chọn đúng bộ nhớ cho đúng trường hợp sử dụng
  - ▶ Cloudera sẽ hỗ trợ và đầu tư vào cả ba

# Thiết kế cơ bản của Kudu

- I Kudu là một trình quản lý lưu trữ dữ liệu dạng cột
  - ▶ Lưu trữ dữ liệu theo cột được gõ mạnh
- I Cụm Kudu lưu trữ các bảng trông giống như các bảng cơ sở dữ liệu SQL
  - ▶ Một cặp giá trị và khóa nhị phân đơn giản
  - ▶ Phức tạp với hàng trăm thuộc tính được gõ mạnh
- I Kiến trúc Master Slave
  - ▶ Máy chủ master chịu trách nhiệm về siêu dữ liệu
  - ▶ Máy chủ tablet chịu trách nhiệm về dữ liệu thực tế
- I Kiến trúc Leader Follower
  - ▶ Leader trong số nhiều máy chủ master
  - ▶ Máy chủ tablet có thể là lãnh đạo hoặc người theo dõi



## Lưu trữ dữ liệu cột

Tên 1	Tuổi 1	Năm 1
Tên 2	Tuổi 2	Năm 2
Tên 3	Tuổi 3	Năm 3

Lưu trữ theo hàng

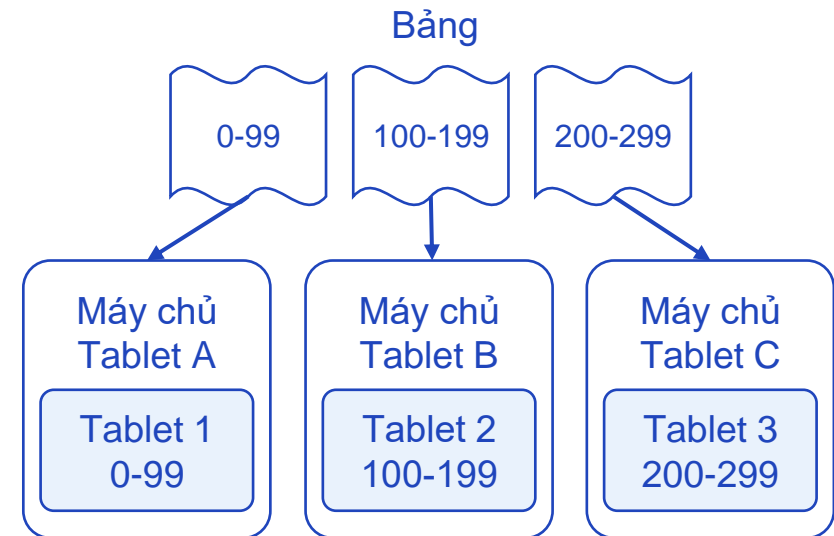


Lưu trữ theo cột



# Bảng (tables) và máy tính bảng (tablets)

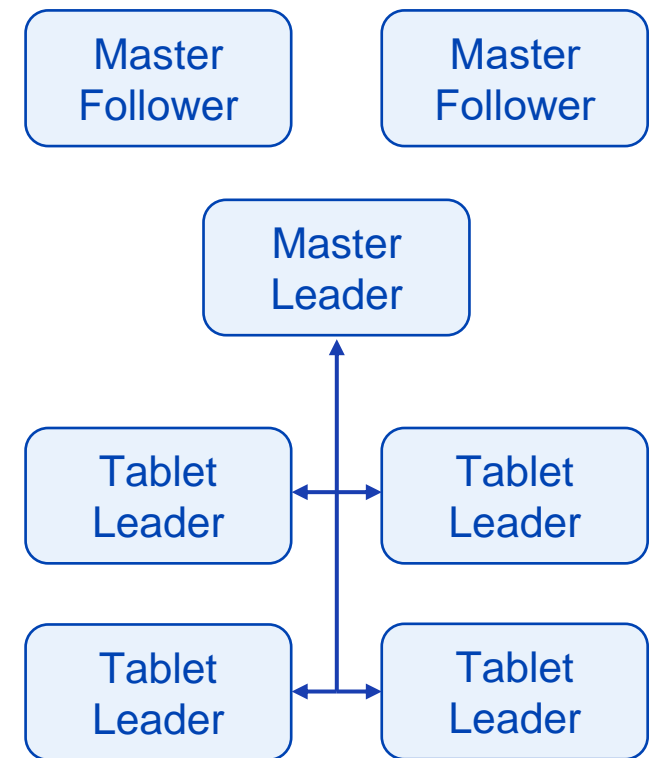
- I Mỗi bảng phải có một KHÓA CHÍNH
  - ▶ Cột đơn như mã định danh duy nhất hoặc khóa ghép phức tạp
- I Một bảng được chia thành các phân đoạn được gọi là tablet
  - ▶ Bảng được phân chia theo chiều ngang thành các tablet
  - ▶ Tablet là một phần liên kề của bảng
- I Các hàng được gán cho tablet dựa trên chiến lược phân vùng
  - ▶ Được phân vùng dựa trên một hoặc nhiều khóa phân vùng
  - ▶ Cột khóa phân vùng phải là một trong các cột khóa chính
  - ▶ Mặc định là không có phân vùng
- I Mục tiêu của chiến lược phân vùng là phân phối các lần đọc và ghi giữa các máy chủ tablet
- I Tablet được lưu trữ trên đĩa cục bộ
  - ▶ Không phải HDFS





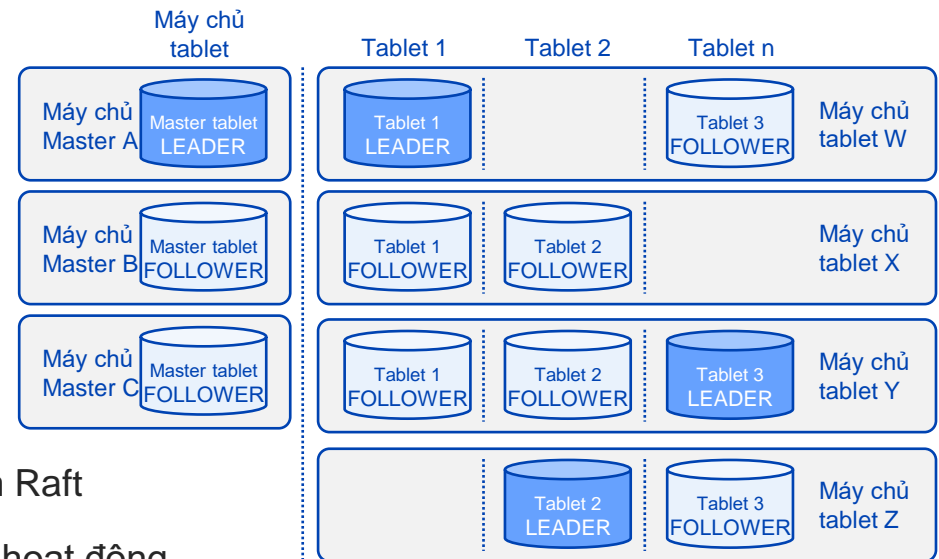
# Kiến trúc Master Slave

- | Master theo dõi tất cả các tablet, máy chủ tablet và Bảng danh mục
  - ▶ Lưu trữ thông tin về bảng và tablet
  - ▶ Bảng danh mục là một bảng đặc biệt và không thể truy cập trực tiếp
  - ▶ Có thể truy cập thông qua các hoạt động siêu dữ liệu được hiển thị trong API máy khách
- | Một cụm Kudu có thể có nhiều Master
  - ▶ Cung cấp khả năng chịu lỗi trong trường hợp thất bại
  - ▶ Tuy nhiên, chỉ có thể có một chủ hoạt động
  - ▶ Nếu Leader Master thất bại, một Master mới sẽ được bầu bằng thuật toán Raft Consensus
- | Máy chủ tablet lưu trữ và phục vụ máy tính bảng cho khách hàng
- | Đối với bất kỳ bảng cụ thể nào, một máy chủ tablet đóng vai trò là người lãnh đạo và những người khác đóng vai trò là bản sao theo dõi của tablet



# Kiến trúc Leader Follower

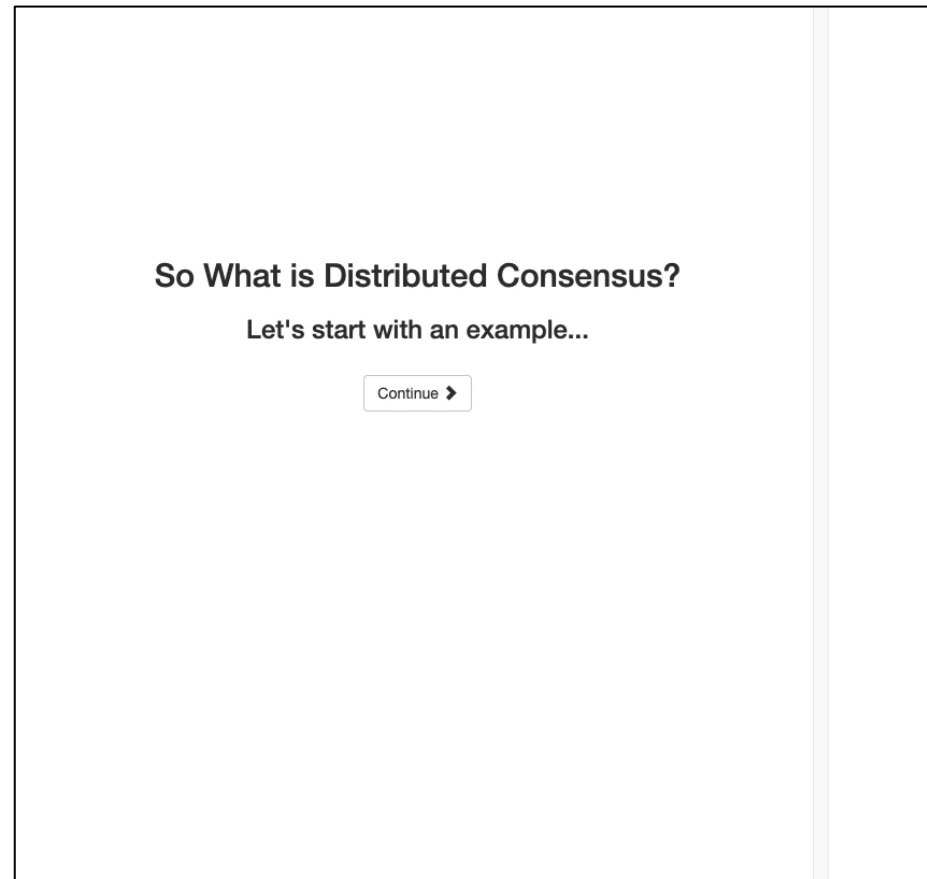
- I Mỗi tablet có nhiều bản sao
  - ▶ Thông thường là 3 hoặc 5 (số lẻ) bản sao
  - ▶ Mỗi bản sao được phục vụ bởi một máy chủ tablet
- I Một trong những máy chủ tablet, đóng vai trò là lãnh đạo
  - ▶ Chỉ dịch vụ Leader được quyền viết yêu cầu
- I Phần còn lại đóng vai trò là Người theo dõi (Follower)
  - ▶ Lãnh đạo và Người theo dõi, mỗi dịch vụ đọc yêu cầu
- I Các nhà lãnh đạo được bầu bằng Thuật toán đồng thuận Raft
- I Thuật toán đồng thuận Raft cũng được sử dụng cho các hoạt động ghi
  - ▶ Lãnh đạo gửi yêu cầu viết cho tất cả những người theo dõi
  - ▶ Sau khi bài viết được đa số người theo dõi duy trì, lãnh đạo có thể xác nhận khách hàng
  - ▶ Đảm bảo khả năng chịu lỗi và nhất quán
  - ▶ Đối với N bản sao,  $(N-1)/2$  bản sao bị lỗi có thể được chấp nhận



# Thuật toán đồng thuận Raft

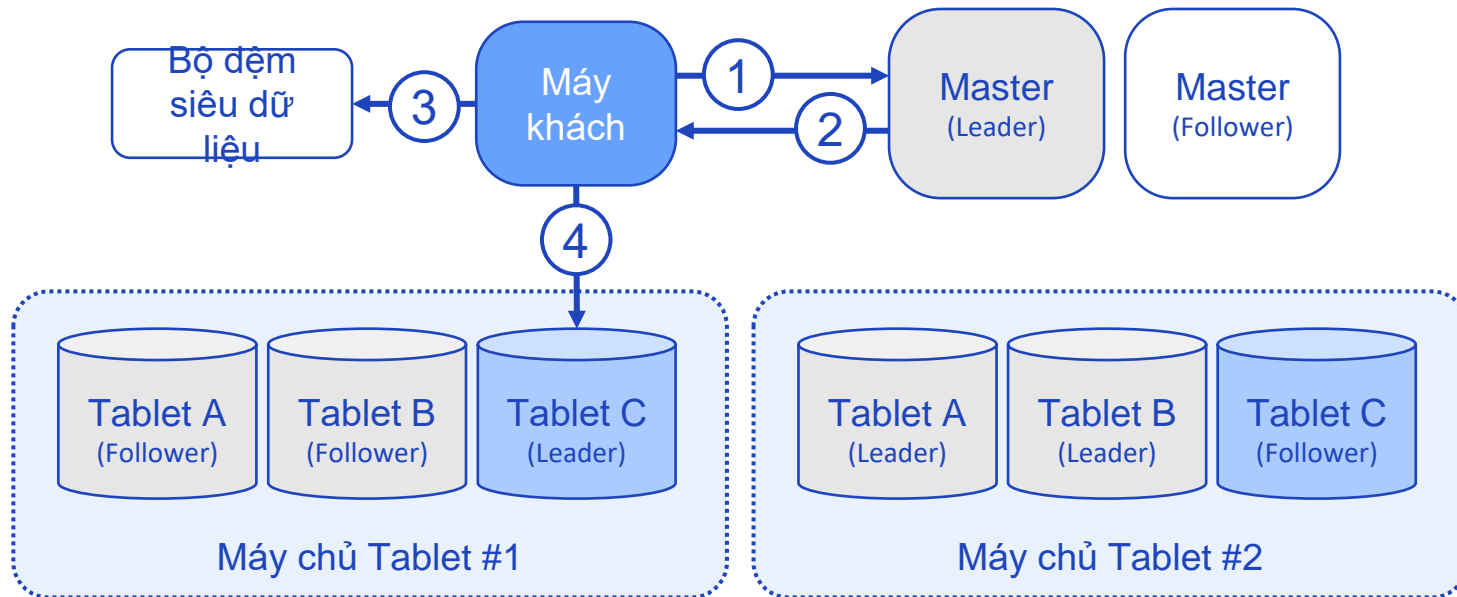
[Video]

- I Mục tiêu là đạt được sự đồng thuận phân tán
  - ▶ Sự đồng thuận giữa nhiều nút được phân phối trên một cụm
  - ▶ Giá trị của cột "tên" trong hàng 3 được lưu trữ trong máy chủ máy tính bảng A, B và C là gì?



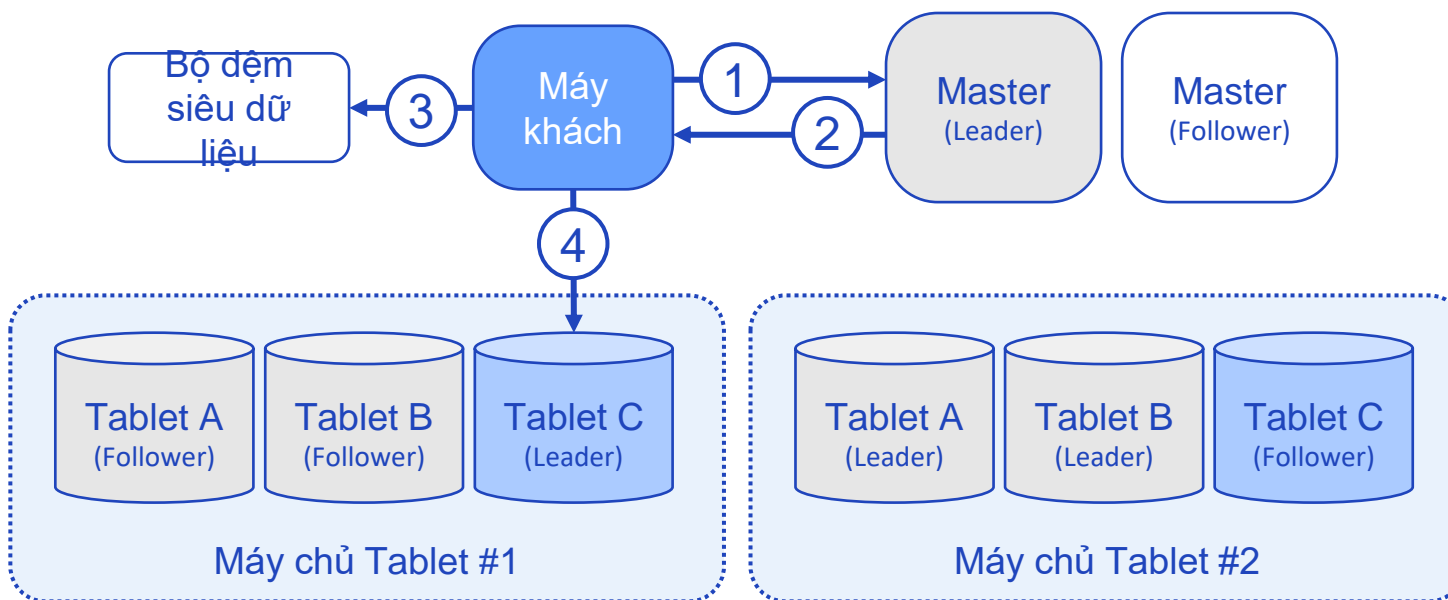
## Mẫu Truy cập Đọc/Ghi của Máy khách (1/4)

I Bước 1 : Khách hàng liên hệ với Master để biết vị trí dữ liệu để truy cập đọc/ghi



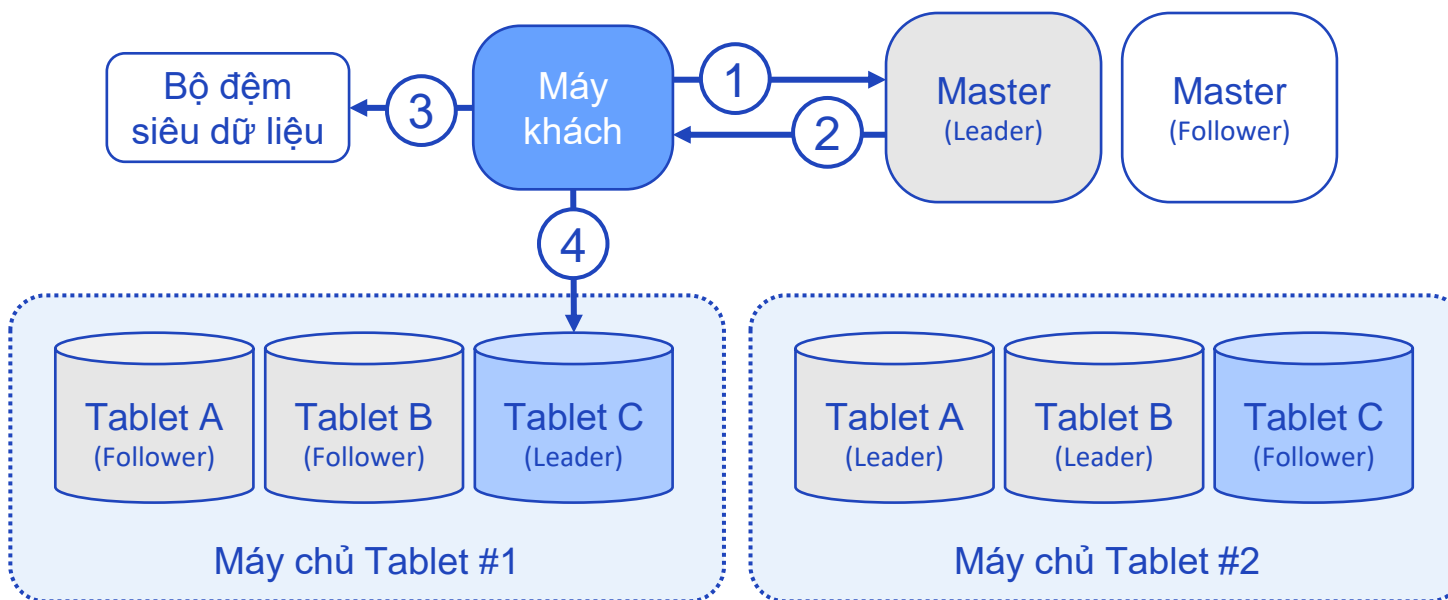
## Mẫu Truy cập Đọc/Ghi của Máy khách (2/4)

I Bước 2 : Master cung cấp vị trí của tablet và Máy chủ tablet lưu trữ dữ liệu



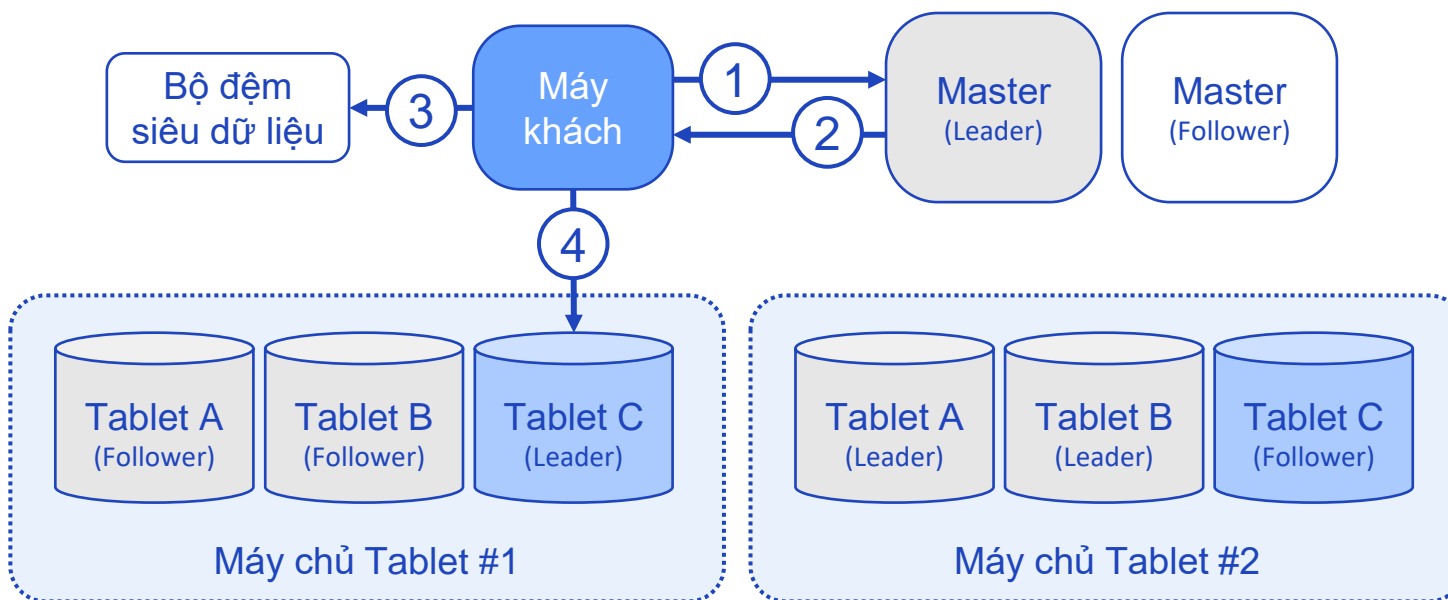
## Mẫu Truy cập Đọc/Ghi của Máy khách (3/4)

**Bước 3** : Máy khách lưu trữ Siêu dữ liệu cho tất cả máy tính bảng truy cập gần đây



## Mẫu Truy cập Đọc/Ghi của Máy khách (4/4)

Bước 4 : Khách hàng liên hệ trực tiếp Tablet Leader để truy cập máy tính bảng và bắt đầu thao tác đọc/ghi



# Khả năng chịu lỗi (1/2)

### I Lỗi FOLLOWER tạm thời:

- ▶ Leader vẫn có thể đạt được đa số
- ▶ Khởi động lại máy chủ tablet của người theo dõi trong vòng 5 phút và nó sẽ tham gia lại một cách minh bạch

### I Lỗi LEADER tạm thời:

- ▶ Những follower mong đợi được nghe nhịp tim từ leader của họ cứ sau 1,5 giây
- ▶ 3 lần lỡ nhịp: bầu chọn leader!
- ▶ LEADER mới được bầu chọn từ các nút còn lại trong vòng vài giây
- ▶ Khởi động lại trong vòng 5 phút và nó sẽ tham gia lại với tư cách là FOLLOWER

### I Xử lý N bản sao $(N-1)/2$ lỗi

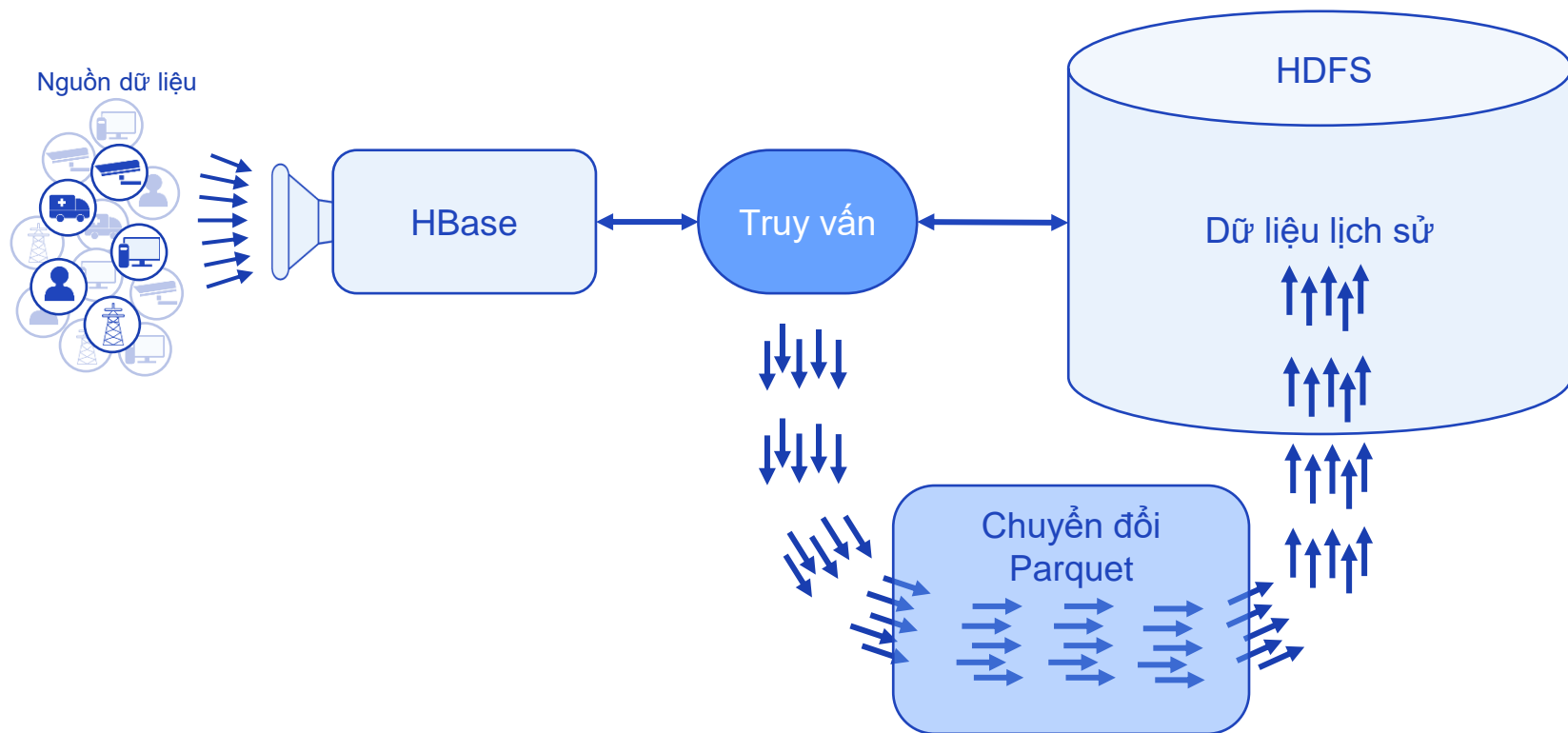


# Khả năng chịu lỗi (2/2)

### I Lỗi vĩnh viễn:

- ▶ Leader thông báo rằng một người theo dõi đã chết trong 5 phút
- ▶ Trục xuất follower đó
- ▶ Master chọn một bản sao mới
- ▶ Leader sao chép dữ liệu sang dữ liệu mới, dữ liệu này sẽ tham gia với tư cách là FOLLOWER mới

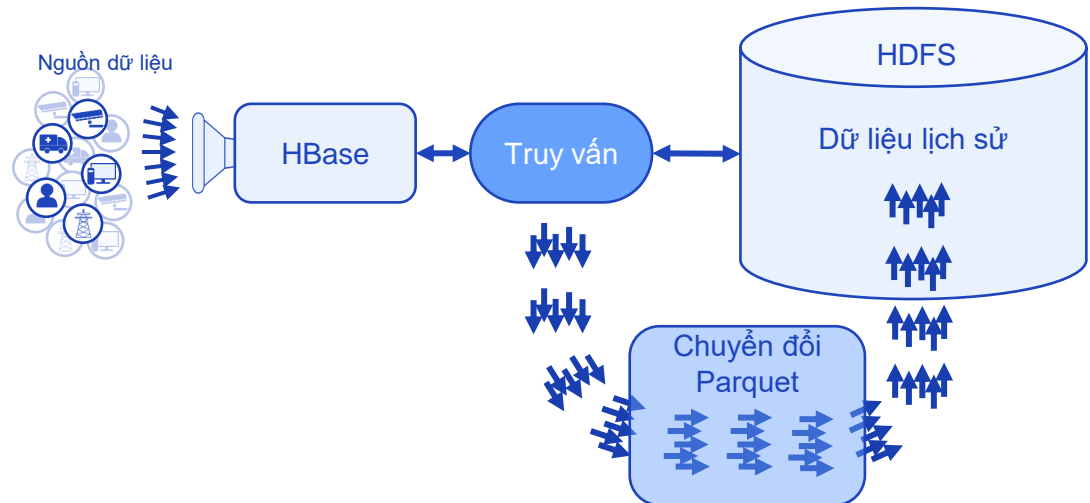
## Ví dụ về phân tích thời gian thực



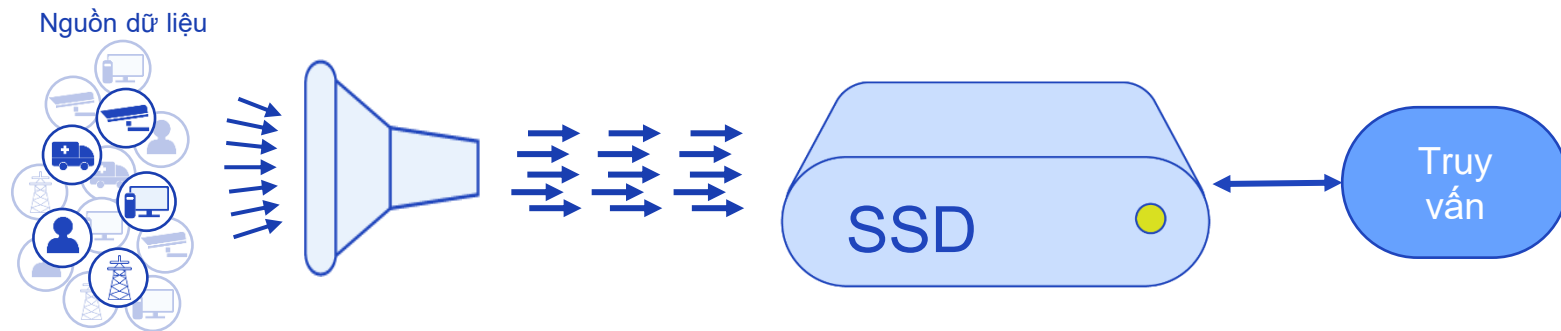
# Các vấn đề về phân tích thời gian thực

## I Cân nhắc

- ▶ Làm cách nào để xử lý lỗi trong quá trình này?
- ▶ Tần suất tôi sắp xếp lại luồng dữ liệu thành định dạng phù hợp để báo cáo là bao lâu?
- ▶ Khi báo cáo, tôi thấy dữ liệu chưa được sắp xếp lại như thế nào?
- ▶ Làm cách nào để đảm bảo rằng các công việc quan trọng không bị gián đoạn do bảo trì? Làm cách nào để xử lý lỗi trong quá trình này?
- ▶ Tần suất tôi sắp xếp lại luồng dữ liệu thành định dạng phù hợp để báo cáo là bao lâu?
- ▶ Khi báo cáo, tôi thấy dữ liệu chưa được sắp xếp lại như thế nào?
- ▶ Làm cách nào để đảm bảo rằng các công việc quan trọng không bị gián đoạn do bảo trì?



## Phân tích thời gian thực với Kudu



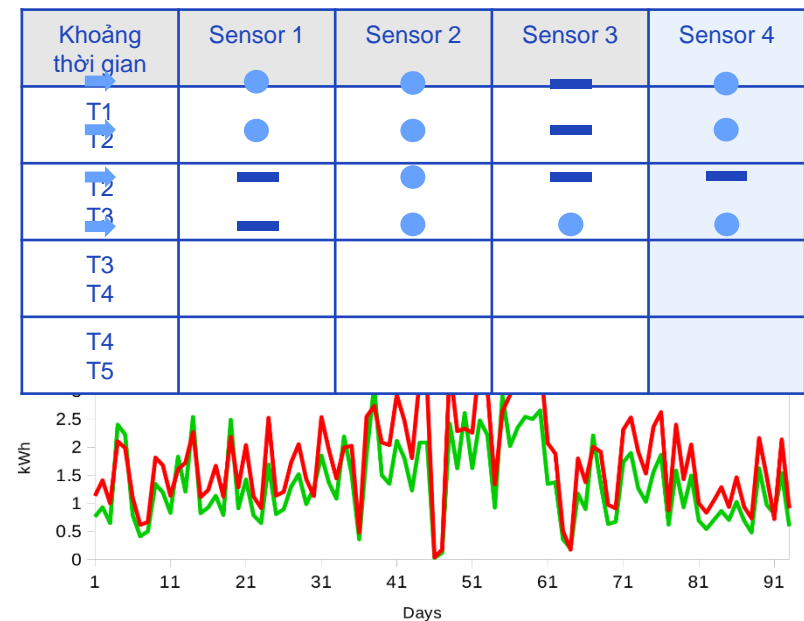
## Trường hợp sử dụng Kudu - Streaming Input

- I Streaming input với tính khả dụng gần thời gian thực
  - ▶ Dữ liệu mới đến nhanh chóng và liên tục
  - ▶ Yêu cầu chèn và cập nhật nhanh
  - ▶ Quét cột hiệu quả để cho phép phân tích thời gian thực



# Trường hợp sử dụng Kudu - Ứng dụng chuỗi thời gian

- Ứng dụng chuỗi thời gian với các mẫu truy cập đa dạng
  - Các điểm dữ liệu được sắp xếp và khóa theo thời gian chúng xảy ra
  - Điều tra hiệu suất của các số liệu theo thời gian
  - Dự đoán hành vi trong tương lai dựa trên dữ liệu trong quá khứ
  - Quét phân tích dữ liệu trong quá khứ cùng với các bản cập nhật và chèn theo thời gian thực
  - Ví dụ: Dữ liệu khách hàng theo chuỗi thời gian lưu trữ lịch sử nhấp chuột mua hàng để dự đoán các lần mua hàng trong tương lai



# Trường hợp sử dụng Kudu - Mô hình dự đoán

## I Mô hình dự đoán

- ▶ Phát triển các mô hình học tập dự đoán từ các tập dữ liệu lớn
- ▶ Mô hình và dữ liệu yêu cầu cập nhật và sửa đổi thường xuyên
- ▶ Nhà khoa học dữ liệu có thể muốn thay đổi một hoặc các yếu tố trong mô hình để quan sát các thay đổi theo thời gian
- ▶ Cập nhật trong HDFS yêu cầu dữ liệu phải được viết lại hoàn toàn
- ▶ Trong Kudu, những cập nhật và thay đổi này có thể diễn ra gần như theo thời gian thực

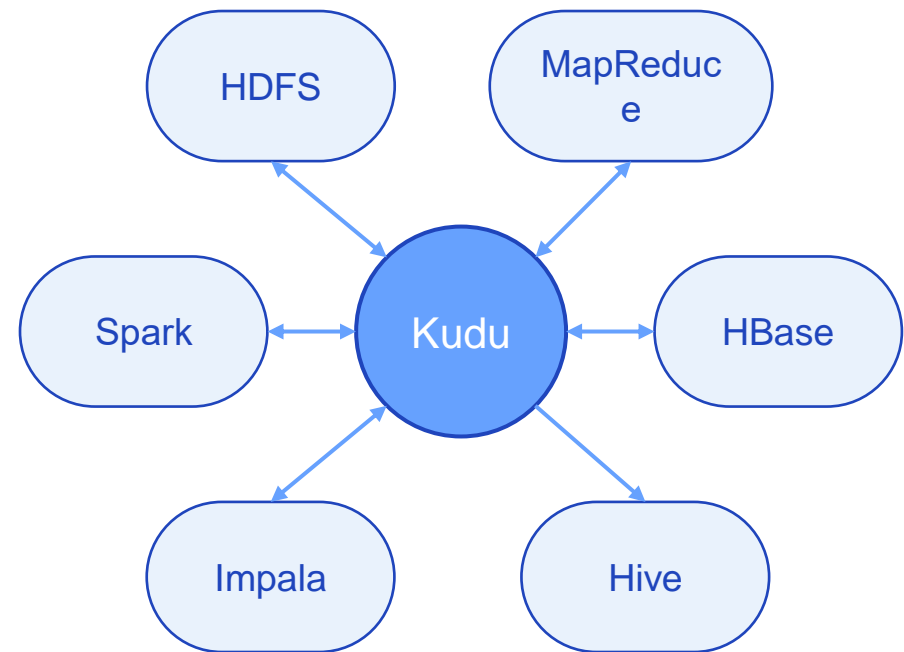
Mô hình dự đoán

Xác định đúng khách hàng và thực hiện các hành động phù hợp



# Trường hợp sử dụng Kudu - Tích hợp các hệ thống kế thừa

- I Kết hợp dữ liệu trong Kudu với các hệ thống kế thừa
  - ▶ Các công ty tạo dữ liệu từ nhiều nguồn
  - ▶ Được lưu trữ trong nhiều hệ thống và định dạng khác nhau
  - ▶ Được lưu trữ trong RDBMS truyền thống
  - ▶ Được lưu trữ trong NoSQL linh hoạt
  - ▶ Được lưu trữ trong HDFS quét cao
  - ▶ Tất cả các loại dữ liệu trên có thể được tích hợp và truy vấn bằng cách sử dụng kết hợp Impala / Kudu



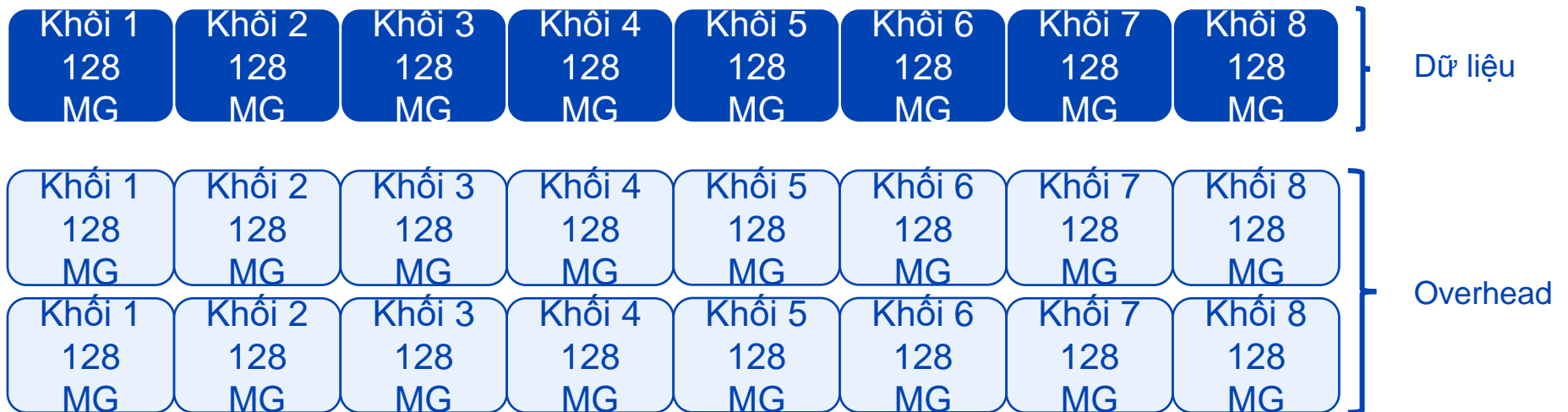


## Tại sao là Erasure mã hóa lưu trữ?

### I Sao chép Hadoop rất tốn kém

- ▶ Hệ số sao chép là 3 có nghĩa là 200% chi phí hoặc hiệu suất 33%
- ▶ Cung cấp khả năng chịu lỗi 66,67% - lượng dữ liệu có thể bị mất và vẫn khôi phục được
- ▶ Độ bền của 2 - số lần thất bại đồng thời có thể tồn tại

### I Các khối được sao chép cho dữ liệu nóng và lạnh hiếm khi được truy cập



## Ví dụ Lưu trữ EC Sử dụng Mã hóa XOR

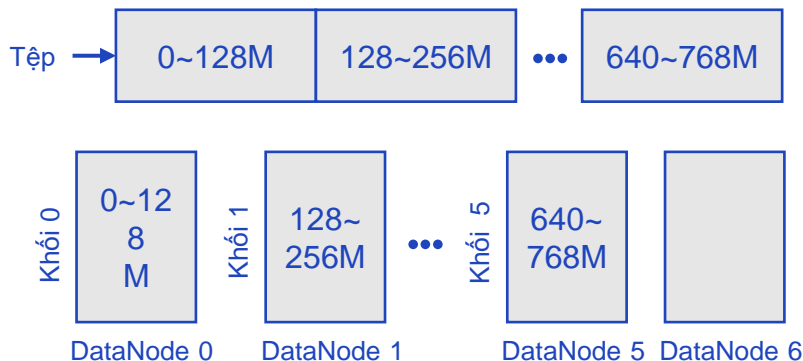
- I Ví dụ mã hóa OR độc quyền
  - ▶ Yêu cầu thêm 2 overhead bit hoặc 100% overhead bit hoặc 50% hiệu suất
  - ▶ Đạt độ bền 1
- I Độc quyền OR
  - ▶ Yêu cầu thêm 1 bit hoặc 50% overhead hoặc 67% hiệu suất
  - ▶ Đạt độ bền 1

Nhân rộng	0	0	1	1	2 bits overhead
Mã hóa XOR	0	$\oplus$	1	1	1 bit overhead

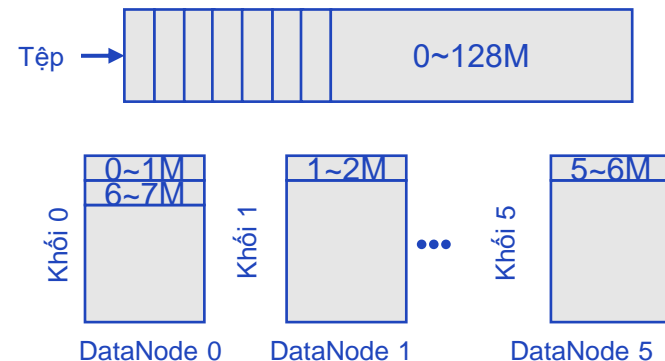
# Tách khối dữ liệu

- Các khối dữ liệu HDFS không có Bộ lưu trữ EC được lưu trữ dưới dạng các khối liên kề
  - ▶ Toàn bộ khối được lưu trữ cùng nhau
- Đối với Bộ lưu trữ EC, mỗi khối dữ liệu được chia thành các dải và được lưu trữ trong các datanode khác nhau
- Đơn vị cơ bản của dữ liệu sọc được gọi là ô
  - ▶ Ví dụ, một ô 64 KB

Bố cục khối:

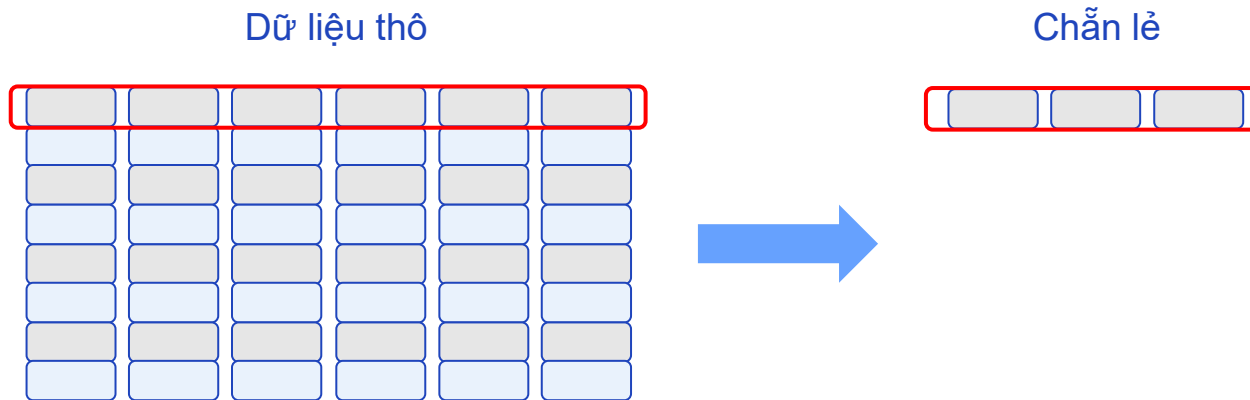


Bố cục khối:



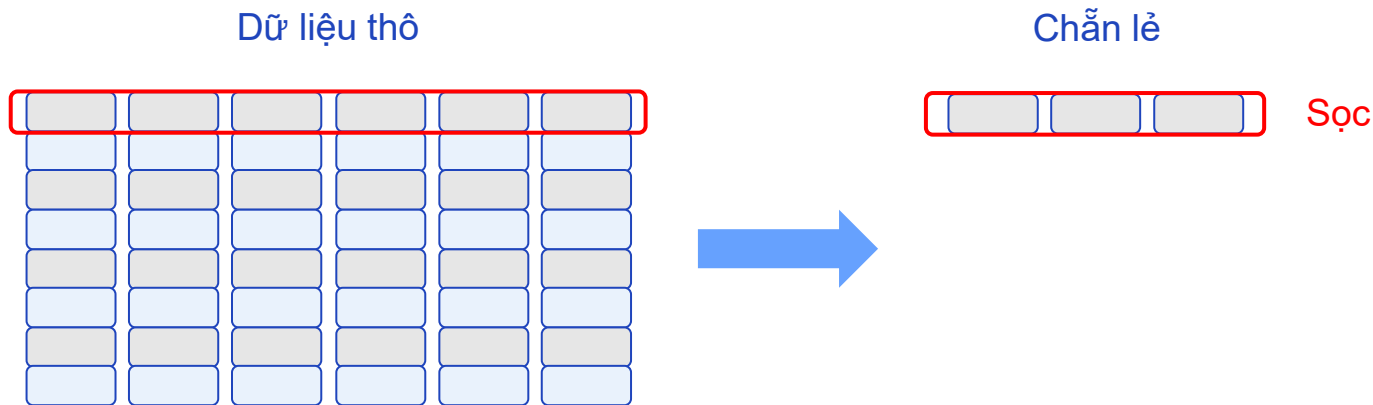
## Dữ liệu stripe với các khối chẵn lẻ

- | Codec được sử dụng trong bộ lưu trữ HDFS EC xác định cấu hình khối chẵn lẻ và dữ liệu
- | Codec Reed-Solomon (6, 3)
  - ▶ 6 ô dữ liệu sẽ được dùng để tính 3 ô chẵn lẻ



## Dữ liệu stripe với các khối chẵn lẻ

- | Hầu hết các codec được sử dụng trong bộ lưu trữ HDFS EC xác định cấu hình khối dữ liệu và khối chẵn lẻ
- | Codec Reed-Solomon (6, 3)
  - ▶ 6 ô dữ liệu sẽ được dùng để tính 3 ô chẵn lẻ
- | 6 ô dữ liệu và 3 ô chẵn lẻ được gọi là nhóm sọc hoặc EC



## So sánh các thuật toán Codec

I Các thuật toán codec khác nhau cung cấp độ bền và hiệu quả khác nhau

Độ bền dữ liệu = Có thể chịu được bao nhiêu lỗi đồng thời?

Hiệu quả lưu trữ = Bao nhiêu phần của phần lưu trữ dành cho dữ liệu hữu ích?

	Độ bền dữ liệu	Hiệu quả lưu trữ
Bản sao đơn	0	100%
Sao chép 3 chiều	2	33%
XOR với 6 ô dữ liệu	1	86%
RS (6,3)	3	67%
RS (10,4)	4	71%

# Sử dụng bộ lưu trữ EC

- | Mỗi ô trong dải EC phải nằm trên một datanode khác
  - ▶ Đối với Reed-Solomon (6,3), điều này có nghĩa là tối thiểu 9 datanode được định cấu hình trong cụm
- | Các tệp Erasure Coded được trải rộng trên các rack để chịu được rack nếu có thể
  - ▶ Hậu quả là khi đọc ghi dữ liệu stripe thì hầu hết các thao tác sẽ bị off-rack
- | Để sử dụng bộ lưu trữ HDFS EC, hãy định cấu hình hoặc chọn một trong các chính sách erasure mã hóa mặc định
- | Định nghĩa chính sách Erasure mã hóa
  - ▶ Lược đồ EC - Thuật toán codec và số lượng dữ liệu và khối chẵn lẻ trong một nhóm EC
  - ▶ Kích thước của ô phân mảnh
  - ▶ Chính sách mặc định là "RS-6-3-1024K" - Codec Reed Solomon với 6 dữ liệu và 3 ô chẵn lẻ trong đó mỗi ô có kích thước 1024KB
- | Chính sách EC được áp dụng ở cấp thư mục
  - ▶ Mọi dữ liệu hiện có trong thư mục sẽ không được chuyển đổi
  - ▶ Chỉ dữ liệu mới được lưu trữ vào thư mục mới được mã hóa EC

Bài 1.

# Giải pháp thay thế lưu trữ dữ liệu

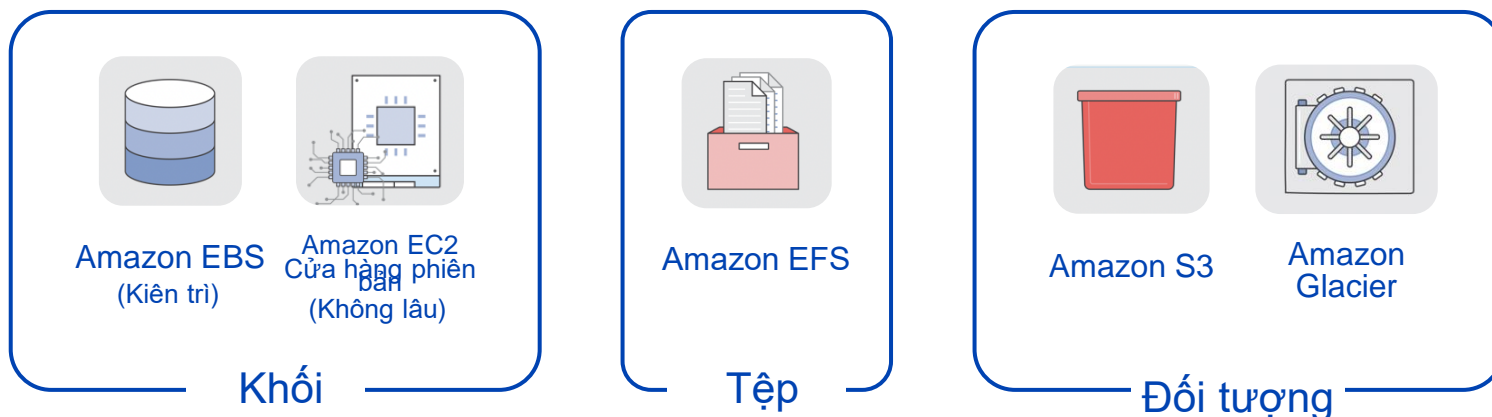
| 1.1. Lưu trữ tại chỗ

| 1.2. Lưu trữ đám mây công cộng



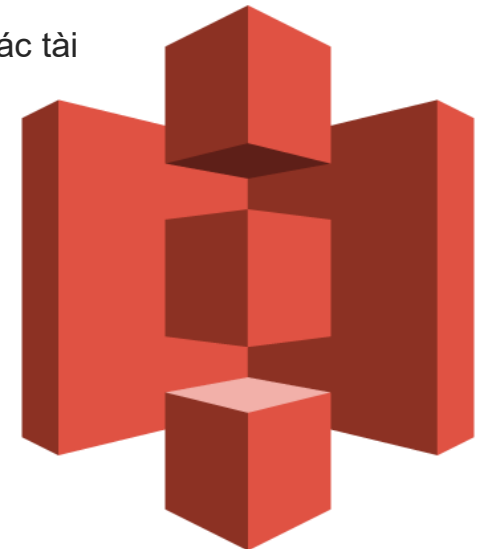
# Tổng quan về dịch vụ lưu trữ AWS

- Các loại lưu trữ
  - Lưu trữ khối - Một loại lưu trữ trong đó dữ liệu người dùng được lưu trữ và truy cập trong các đơn vị khối trên một ổ đĩa trên đĩa cục bộ hoặc bộ lưu trữ SAN
  - Lưu trữ tệp - Một loại lưu trữ (NAS) truy cập vào bộ lưu trữ bao gồm các hệ thống tệp trong Bais tệp bằng giao thức dựa trên mạng.
  - Lưu trữ đối tượng - Một vùng chứa ảo lưu trữ dữ liệu và thuộc tính được đóng gói, siêu dữ liệu và ID đối tượng.



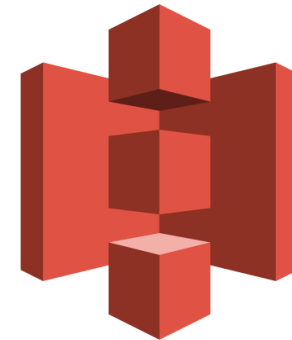
# Amazon S3

- | Dịch vụ lưu trữ đơn giản
- | Giao diện dịch vụ web đơn giản
- | Bucket (bộ chứa)
  - ▶ Để tải dữ liệu (ảnh, video, tài liệu, v.v.) lên S3, trước tiên hãy tạo một bucket S3 trong một Khu vực AWS. Tải đối tượng lên bucket này
  - ▶ Tên bộ chứa là duy nhất trên toàn cầu và không gian tên được chia sẻ bởi tất cả các tài khoản AWS.
- | Sự cho phép
- | Managing public access to buckets
  - ▶ ACL
  - ▶ Chính sách bucket



## Trường hợp sử dụng với Amazon S3

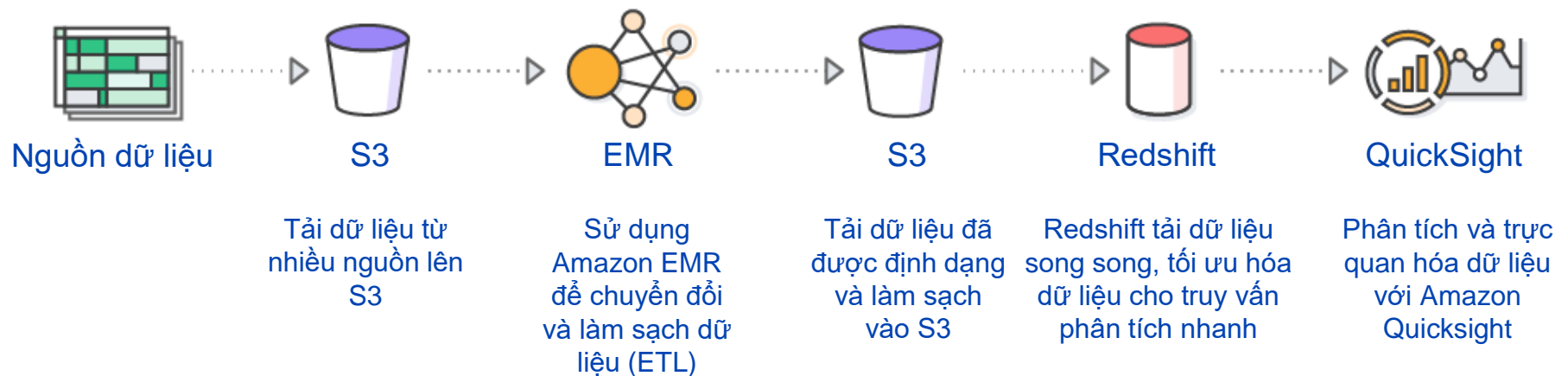
- | Lưu trữ & Phân phối Nội dung
- | Phân tích dữ liệu lớn
- | Sao lưu & Lưu trữ
- | Phục hồi sau thảm họa
- | Lưu trữ trang web tĩnh
- | Dữ liệu ứng dụng gốc trên đám mây
- | Xây dựng kho dữ liệu



# Amazon S 3

# Kho lưu trữ dữ liệu với Amazon S3

- Lưu trữ dữ liệu bằng S3
- Thực hiện các hoạt động chuyển đổi dữ liệu (ETL) và phân tích qua EMR
- Tận dụng nền tảng dịch chuyển đồ cho các ứng dụng kinh doanh thông minh



# Bucket

- | Bucket là nơi chứa các đối tượng được lưu trữ trong S3
- | Tất cả các đối tượng trong S3 được chứa trong một bucket
- | Bucket tổ chức không gian tên S3 ở mức cao nhất
- | Xác định tài khoản chịu trách nhiệm
- | Được sử dụng để kiểm soát truy cập




# Các đối tượng (Objects)

- | Các đối tượng là các thực thể cơ bản được lưu trữ trong S3
- | Các đối tượng bao gồm dữ liệu đối tượng và siêu dữ liệu
  - ▶ Phần dữ liệu mờ đục và có thể nhìn thấy
  - ▶ Siêu dữ liệu bao gồm các thuộc tính của đối tượng và được lưu trữ trong các cặp khóa-giá trị
  - ▶ Một số ví dụ về siêu dữ liệu mặc định
    - date last modified
    - HTTP metadata such as Content-Type
  - ▶ Bạn có thể xem meta theo chương trình hoặc từ bảng điều khiển
- | Mỗi đối tượng được xác định duy nhất trong một nhóm bằng một khóa và ID phiên bản

## Khóa (Keys)

- | Khóa là mã định danh duy nhất cho một đối tượng trong nhóm
- | Sự kết hợp giữa bucket, khóa, ID phiên bản xác định duy nhất mọi đối tượng
- | Sự kết hợp này được sử dụng làm điểm cuối dịch vụ web

<https://mybucket.s3.us-west-2.amazonaws.com/myobjectkey>

  
Tên Bucket    Tên vùng

  
Tên khóa

## Vùng (Regions)

- | Bạn có thể chọn Khu vực AWS địa lý nơi S3 sẽ lưu trữ các bộ chứa bạn đã tạo
- | Đặt vùng cho phép bạn tối ưu hóa độ trễ và chi phí
- | Các đối tượng được lưu trữ trong một vùng không bao giờ rời khỏi vùng trừ khi bạn chuyển chúng sang vùng khác một cách rõ ràng

AWS Global Infrastructure





# Quy tắc đặt tên bucket S3

- | Tên bucket phải dài từ 3 đến 63 ký tự.
- | Chỉ sử dụng chữ thường, số, dấu chấm và dấu gạch ngang (-)
- | Tên phải bắt đầu và kết thúc bằng một chữ cái hoặc số.
- | Không sử dụng định dạng địa chỉ IP
- | Tên phải là độc nhất trong phân vùng.
  - ▶ Một phân vùng là một nhóm Khu vực
  - ▶ AWS có ba phân vùng
    - aws (Tiêu chuẩn)
    - aws-cn (Trung Quốc)
    - aws-us-gov (AWS Govcloud tại các khu vực của Hoa Kỳ)

# Truy cập một bucket

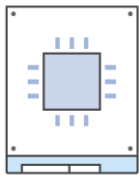
- I Truy cập kiểu lưu trữ ảo (Virtual-hosted-style)
  - ▶ <https://bucket-name.s3.Region.amazonaws.com/key name>
  
- I Truy cập theo kiểu đường dẫn (Path-style)
  - ▶ <https://s3.Region.amazonaws.com/bucket-name/key name>
  
- I Truy cập một bucket thông qua các điểm truy cập S3
  - ▶ <https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com>
  
- I Truy cập một bucket bằng S3: //
  - ▶ S3://bucket-name/key-name

# Xóa một bucket

- | Thao tác xóa khi kích hoạt phiên bản bucket S3 sẽ xóa vĩnh viễn tất cả các phiên bản của tất cả các đối tượng trong bucket.
  
- | Cần nhắc trước khi xóa
  - ▶ Tên bucket là duy nhất. Việc xóa một bucket làm cho tên của nó có sẵn cho những người dùng AWS khác.
  - ▶ Việc xóa một bucket chứa các đối tượng sẽ xóa vĩnh viễn tất cả các đối tượng trong bucket, kể cả các đối tượng đã được chuyển đổi thành lớp lưu trữ S3 Glacier.
  - ▶ Nếu bộ chứa của bạn đang nhận dữ liệu nhật ký từ Cân bằng tải đàn hồi (ELB), chúng tôi khuyên bạn nên ngừng vận chuyển nhật ký ELB tới bộ chứa trước khi xóa bucket.

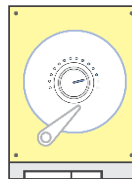
## Lớp lưu trữ S3

I Các lớp lưu trữ được thiết kế cho trường hợp sử dụng của bạn



### S3 Tiêu chuẩn

- ✓ Phân tích Big Data
- ✓ Phân phối nội dung
- ✓ Lưu trữ trang web tĩnh



### Tiêu chuẩn - IA

- ✓ Sao lưu & lưu trữ
- ✓ Phục hồi sau thảm họa
- ✓ Đồng bộ hóa và chia sẻ tệp
- ✓ Dữ liệu lưu trữ lâu



### Amazon Glacier

- ✓ Lưu trữ dài hạn
- ✓ Bảo quản kỹ thuật số
- ✓ Thay thế băng từ

# Dịch vụ lưu trữ Azure

- Microsoft Azure cung cấp các giải pháp lưu trữ, sao lưu và khôi phục trên đám mây có thể mở rộng và bền vững cho tất cả dữ liệu lớn và nhỏ của bạn.
- Nó cung cấp dung lượng lưu trữ có tính khả dụng cao, an toàn, bền bỉ, có thể mở rộng và dự phòng để lưu trữ đối tượng cho các đối tượng dữ liệu, lưu trữ đĩa cho máy ảo Azure, dịch vụ hệ thống tệp cho đám mây, lưu trữ thư và lưu trữ NoSQL.
- Cho dù đó là hình ảnh, âm thanh, video, nhật ký, tệp cấu hình hoặc dữ liệu cảm biến đều được lưu trữ theo cách có thể dễ dàng truy cập cho mục đích phân tích.
- Dịch vụ lưu trữ cốt lõi
  - ▶ Azure Blobs
  - ▶ Azure Files
  - ▶ Azure Queues
  - ▶ Azure Tables
  - ▶ Azure Disks
- Mỗi dịch vụ được truy cập thông qua một tài khoản lưu trữ

Microsoft Azure  
Blob Storage



## Tổng quan về tài khoản lưu trữ

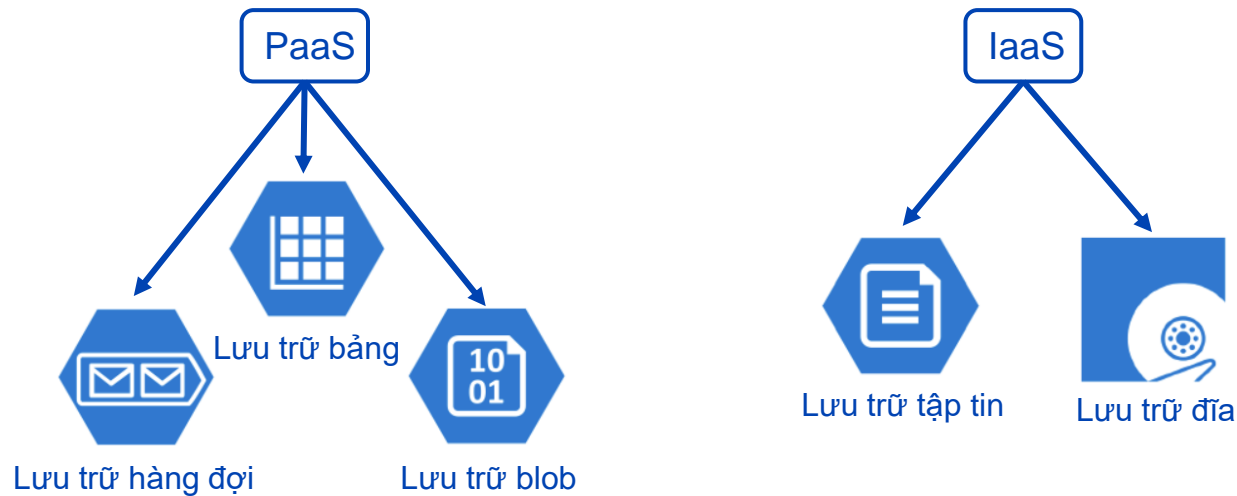
- Tài khoản lưu trữ Azure chứa tất cả các đối tượng dữ liệu Lưu trữ Azure, chẳng hạn như blob, tệp, hàng đợi, bảng và đĩa.
- Tài khoản lưu trữ cung cấp một không gian tên duy nhất cho dữ liệu Lưu trữ Azure.
- Các loại tài khoản lưu trữ

Loại tài khoản lưu trữ	Dịch vụ được hỗ trợ	Phương pháp nhân rộng	Mô hình triển khai
General – purpose V2	Blob, File, Queue, Table, Disk, and Data Lake Gen2	LRS, GRS, RA-GRS, ZRS, GZRS, RA-GZRS	Resource Manager
General – purpose V1	Blob, File, Queue, Table and Disk	LRS, GRS, RA-GRS	Resource Manager, Classic
BlockBlobStorage	Blob ( block blobs and append blobs only )	LRS, ZRS	Resource Manager
FileStorage	File only	LRS, ZRS	Resource Manager
BlobStorage	Blob ( block blobs and append blobs only )	LRS, GRS, RA-GRS	Resource Manager

## Dịch vụ lưu trữ cốt lõi

- Các dịch vụ lưu trữ lõi cung cấp khả năng lưu trữ đối tượng có thể mở rộng quy mô lớn cho các đối tượng dữ liệu, lưu trữ đĩa cho máy ảo Azure (VM), dịch vụ hệ thống tệp cho đám mây, lưu trữ nhắn tin để nhắn tin và lưu trữ NoSQL.

### Lưu trữ Microsoft Azure



# Lưu trữ Azure Blob

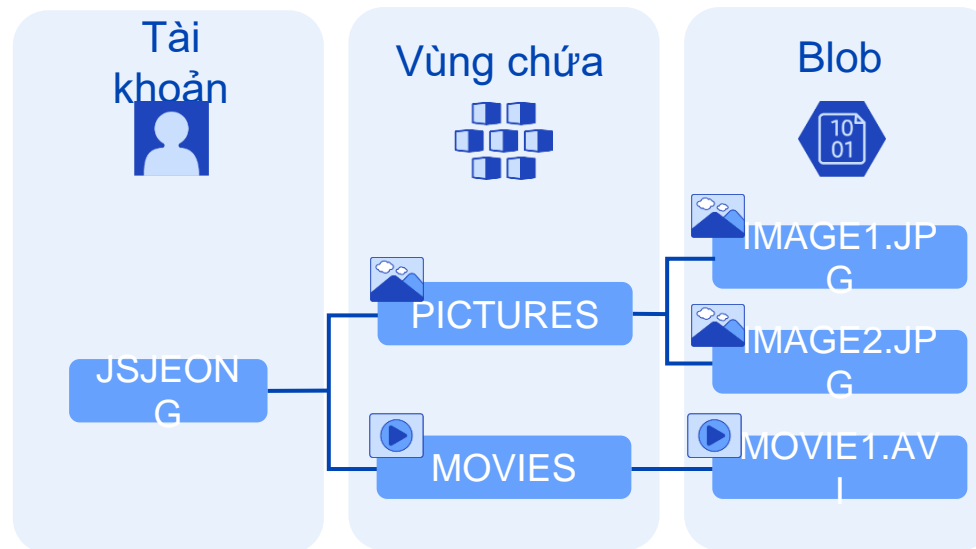
- | Azure Blob Storage là bộ lưu trữ đối tượng của Microsoft dành cho đám mây
- | Được tối ưu hóa để lưu trữ lượng lớn dữ liệu phi cấu trúc
- | Trường hợp sử dụng:
  - ▶ Cung cấp hình ảnh hoặc tài liệu trực tiếp cho trình duyệt của bạn
  - ▶ Lưu trữ tệp để truy cập phân tán
  - ▶ Truyền phát video và âm thanh
  - ▶ Ghi vào tệp nhật ký
  - ▶ Lưu trữ dữ liệu để sao lưu/khôi phục, khắc phục sự cố và lưu trữ
  - ▶ Lưu trữ dữ liệu để phân tích tại chỗ hoặc bởi các dịch vụ được lưu trữ trên Azure
- | Có thể truy cập thông qua việc sử dụng Azure storage REST API, Powershell, CLI, Azure Storage client lib



# Tài nguyên lưu trữ Blob

### I Lưu trữ blob cung cấp ba loại tài nguyên

- ▶ Tài khoản lưu trữ
- ▶ Vùng chứa trong tài khoản lưu trữ
- ▶ Một blob trong một vùng chứa (container)



## Tài nguyên lưu trữ – Tài khoản lưu trữ

- I Tài khoản lưu trữ cung cấp một không gian tên duy nhất
  - ▶ Mỗi đối tượng được lưu trữ có một địa chỉ bao gồm tên tài khoản lưu trữ duy nhất
  - ▶ Địa chỉ là sự kết hợp giữa tên tài khoản và điểm cuối blob



Các tài khoản lưu trữ

# Tài nguyên lưu trữ – Vùng chứa (container)

- I Các vùng chứa tổ chức một tập hợp các blob – tương tự như thư mục
  - ▶ Có thể bao gồm số lượng vùng chứa không giới hạn
  - ▶ Đặt tên – chữ thường và số
  - ▶ Bạn không thể tạo các vùng chứa phụ nhưng bạn có thể tạo các điểm màu có "/" trong đó để mô phỏng các thư mục con

# Tài nguyên lưu trữ – Blob

## I Các khối blob

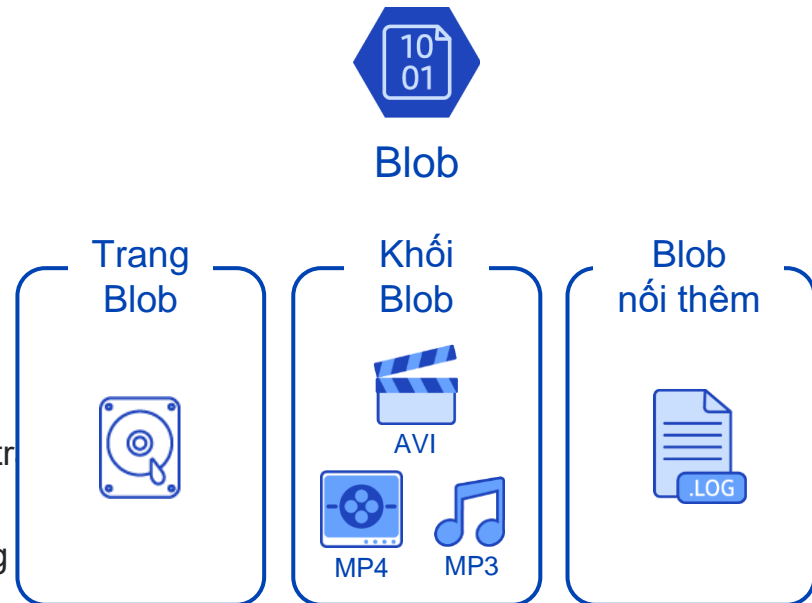
- ▶ Lưu trữ văn bản và dữ liệu nhị phân
- ▶ Được tạo thành từ các khối có thể được quản lý riêng
- ▶ Mỗi khối có thể lưu trữ 4,75 TiB dữ liệu

## I Các blob nối thêm (Append blobs)

- ▶ Tương tự như block blobs nhưng được tối ưu hóa cho append ops
- ▶ Lý tưởng để lưu trữ dữ liệu ghi nhật ký

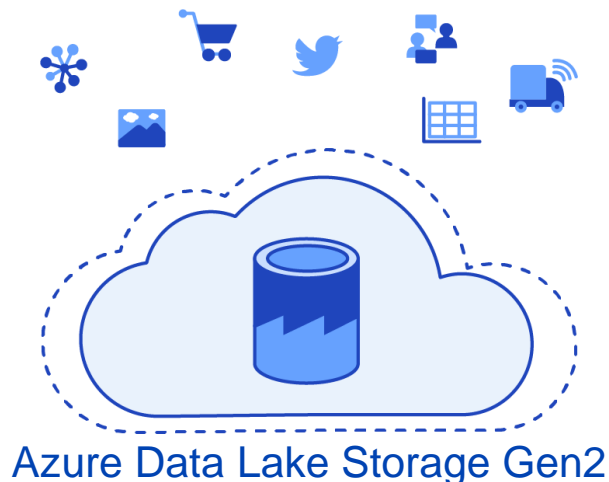
## I Các trang blob

- ▶ Lưu trữ các tệp truy cập ngẫu nhiên từ 512 byte đến 8 TB
- ▶ Lý tưởng cho các hoạt động đọc/ghi ngẫu nhiên thường xuyên
- ▶ Cơ sở cho đĩa Azure IaaS



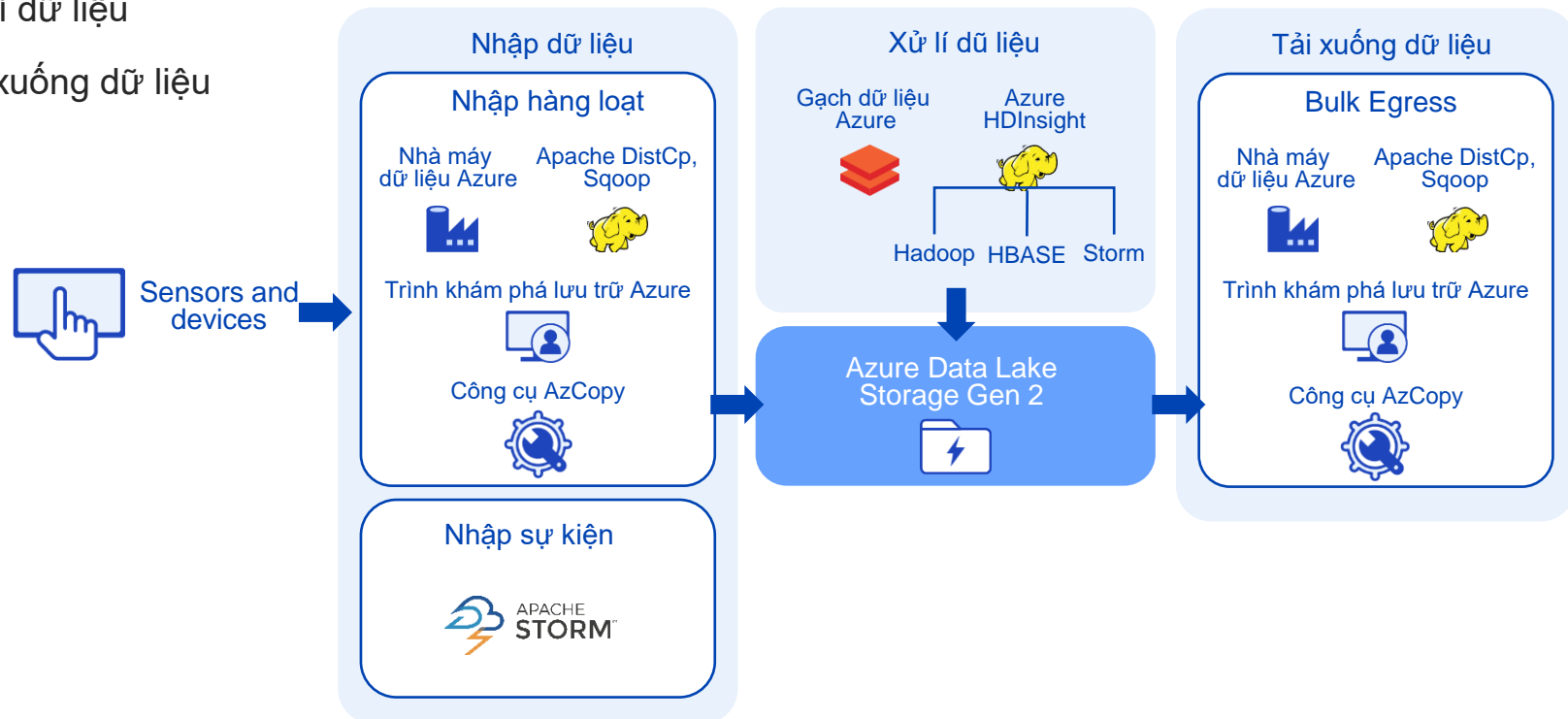
# Azure Data Lake Storage Gen2

- | Azure Data Lake Storage là một giải pháp kho dữ liệu toàn diện, có thể mở rộng và tiết kiệm chi phí để phân tích big data được tích hợp trong Azure.
- | Được xây dựng trên Azure Blob Storage
  - ▶ Sự tích hợp này cho phép khả năng phân tích hiệu suất, phân tầng và quản lý vòng đời dữ liệu của Blob Storage cũng như các tính năng về tính sẵn sàng, bảo mật và độ bền cao của Azure Storage



# ADLS Gen2 dành cho các yêu cầu về Big Data

- Thu thập lượng lớn dữ liệu theo thời gian thực hoặc theo đợt vào kho lưu trữ dữ liệu
- Xử lý dữ liệu
- Tải xuống dữ liệu



# Các tính năng chính của Data Lake Storage Gen2

- | Truy cập tương thích Hadoop
  - ▶ HDFS
  - ▶ Azure HDInsight, Gạch dữ liệu Azure , và Phân tích Synapse Azure
- | Một siêu quyền POSIX
- | Chi phí hiệu quả
- | Trình điều khiển được tối ưu hóa
- | Khả năng mở rộng
- | Tích hợp dịch vụ Azure được hỗ trợ và nền tảng nguồn mở

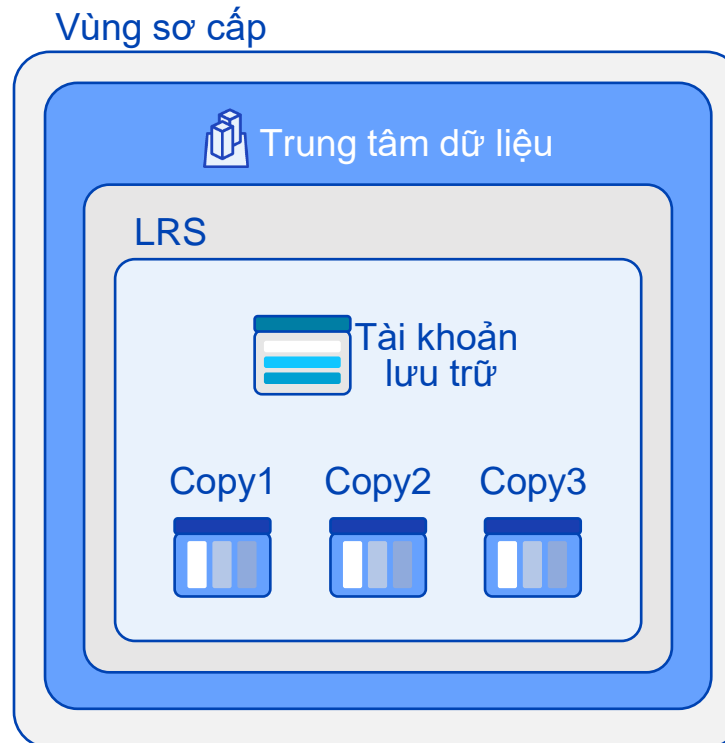
# Dịch vụ lưu trữ Blob

- | Cung cấp bộ nhớ đối tượng có thể được sử dụng để lưu trữ và cung cấp dữ liệu phi cấu trúc
- | Dữ liệu này có thể bao gồm từ dữ liệu Ứng dụng và Web đến hình ảnh, tệp, Dữ liệu lớn từ IoT và tất nhiên là các bản sao lưu và lưu trữ
- | Lưu trữ dữ liệu hàng petabyte
- | Các loại độ bền lưu trữ Blob
  - ▶ Local Redundant Storage (LRS): Lưu trữ dự phòng cục bộ
  - ▶ Zone Redundant Storage (ZRS): Vùng lưu trữ dự phòng
  - ▶ Geo Replicated Storage (GRS): Lưu trữ sao chép địa lý
  - ▶ Read Access Geo Replicated Storage (RA-GRS): Đọc truy cập bộ lưu trữ được sao chép theo địa lý



# LRS(Local Redundant Storage): Lưu trữ dự phòng cục bộ

- Bộ lưu trữ LRS lưu trữ ba bản sao dữ liệu trong một vùng duy nhất để bảo vệ chống lại lỗi đĩa, nút và giá đỡ.



# ZRS(Zone Redundant Storage): Vùng lưu trữ dự phòng

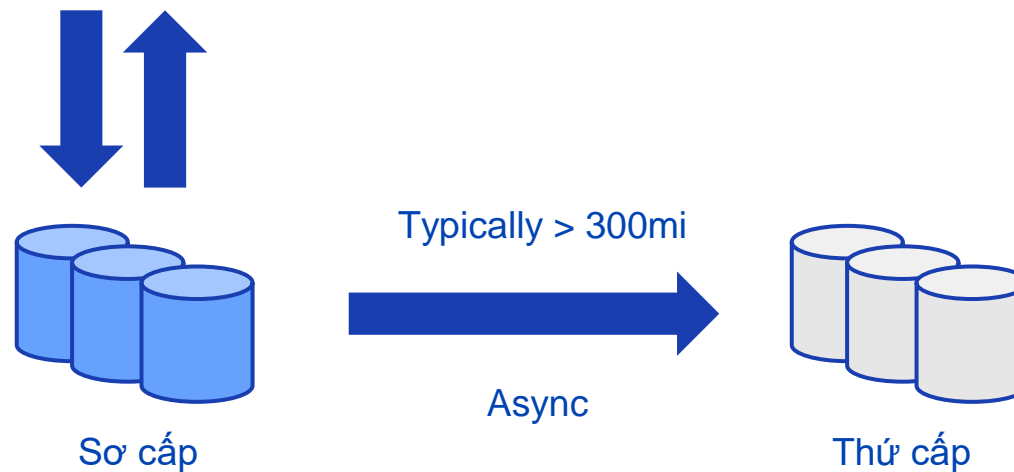
- ZRS tương tự như LRS, nhưng phân chia dữ liệu trên hai trung tâm dữ liệu trong cùng một khu vực.
- Làm cho dữ liệu của bạn linh hoạt hơn so với tùy chọn mặc định, LRS đơn giản.

Vùng sơ cấp



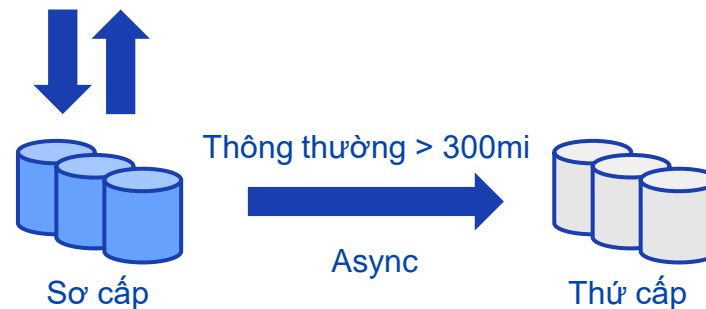
## GRS(Geo Redundant Storage): Lưu trữ sao chép địa lý

- | GRS bảo vệ chống lại các thảm họa lớn trong khu vực bằng cách lưu trữ sáu bản sao ở hai khu vực khác nhau
  - ▶ Ví dụ, Bắc Âu và Tây Âu
- | Sao chép sang vùng thứ cấp là không đồng bộ.



# RA-GRS: Đọc truy cập bộ lưu trữ được sao chép theo địa lý

- RA-GRS: Read Access Geo Replicated Storage (Đọc truy cập bộ lưu trữ được sao chép theo địa lý )
- It provides read-only access to the secondary storage through an accessible endpoint to allow for load-balanced queries



Chiến lược nhân rộng	LRS	ZRS	GRS	RA-GRS
Liệu dữ liệu có được sao chép vào nhiều trung tâm dữ liệu hay không	Không	Có	Có	Có
Vị trí thứ cấp đọc dịch vụ như sơ cấp	Không	Không	Không	Có
Số bản sao	3	3	6	6

# Các loại lưu trữ

### I Bộ nhớ tiêu chuẩn (mặc định)

- ▶ Khi quyết định các dịch vụ Lưu trữ Azure, hiệu suất và chi phí là hai yếu tố quan trọng
- ▶ Đĩa quay thông thường, hàng hóa

### I Dung lượng cao cấp

- ▶ Để hỗ trợ ổ đĩa có hiệu suất cao, độ trễ thấp
- ▶ Lưu trữ dữ liệu trên ổ cứng thể rắn (SSD)

### I Kho lưu trữ lạnh (Cold storage)

- ▶ Dịch vụ này hướng tới truy cập không thường xuyên
- ▶ Nó được thiết kế cho mục đích lưu trữ dữ liệu, chẳng hạn như sao lưu cơ sở dữ liệu và VM.
- ▶ API và SDK giữa bộ lưu trữ nóng và bộ nhớ mát giống hệt nhau
- ▶ Chi phí rẻ hơn. Nếu bạn truy cập thường xuyên thì chi phí đọc dữ liệu càng cao. (Xem xét không gian lưu trữ & tần suất sử dụng)

# [Lab1] AWS S3 và Glacier



# [Lab2] Lưu trữ AWS S3 Glacier



Bài 2.

# NoSQL

Lưu trữ Big Data



Bài 2.

# NoSQL

| 2.1. Tổng quan về NoSQL

| 2.2. Apache HBase

| 2.3. Cassandra

| 2.4. MongoDB

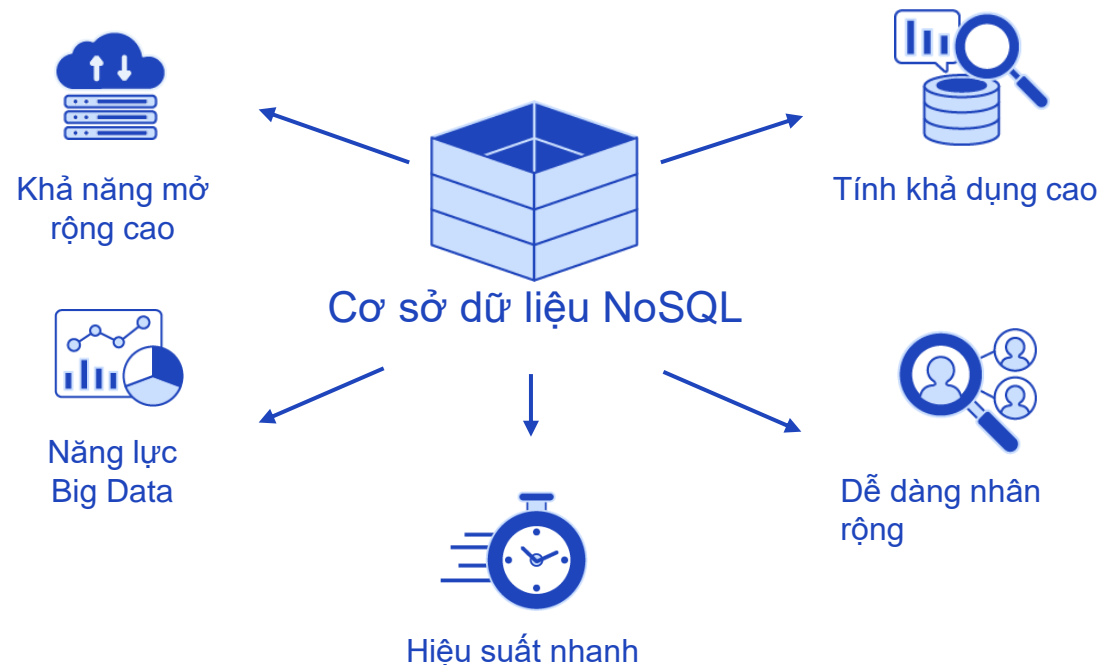
# NoSQL là gì? (1/2)

- | Viết tắt của “Không có SQL”, “**Không chỉ SQL**”, “SQL hoạt động phi quan hệ”
  - ▶ RDB không hỗ trợ giao diện SQL tiêu chuẩn
- | Năm 2009, Johan Oskarsson đã sử dụng DB phân tán dựa trên mã nguồn mở
- | Schema ít hơn, không hỗ trợ giao dịch
- | Lưu trữ dữ liệu với lược đồ DB không quan hệ

NotOnlySQL

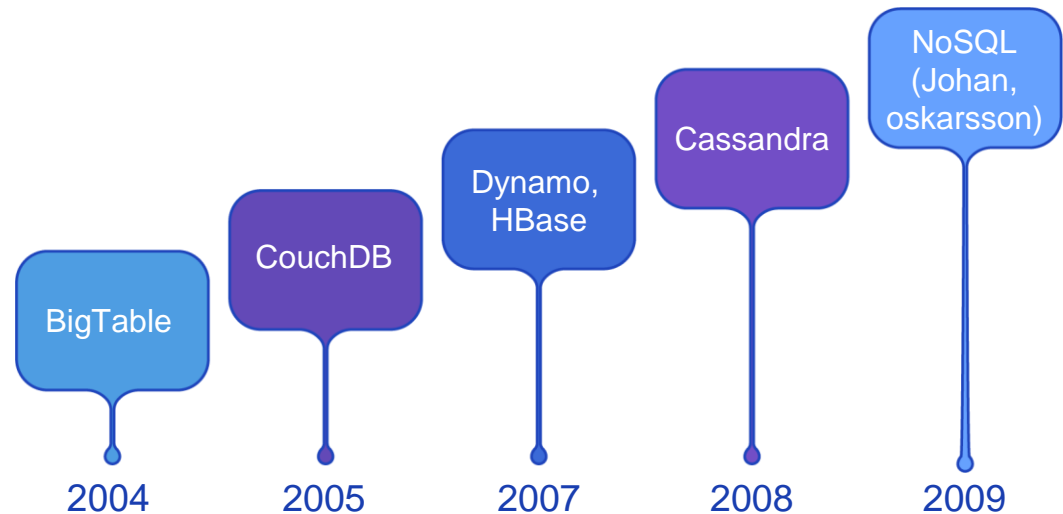
# NoSQL là gì? (2/2)

- I Bổ sung cho cơ sở dữ liệu quan hệ
- I Một hệ thống quản lý cơ sở dữ liệu linh hoạt có thể lưu trữ và xử lý cả dữ liệu có cấu trúc và bán cấu trúc



# Tóm tắt lịch sử của NoSQL

- | Carlo Strozzi sử dụng thuật ngữ NoSQL cho cơ sở dữ liệu quan hệ mã nguồn mở, nhẹ của mình. (1998)
- | Cơ sở dữ liệu đồ thị Neo4j được ra mắt (2000)
- | Google **BigTable** được ra mắt (2004)
- | CouchDB được ra mắt (2005)
- | Tài liệu nghiên cứu về **Amazon Dynamo** được phát hành (2007)
- | Facebook mã nguồn mở cho dự án **Cassandra** (2008)
- | Thuật ngữ NoSQL được giới thiệu lại bởi Johan Oskarsson (2009)



Hình : Lịch sử của HBase

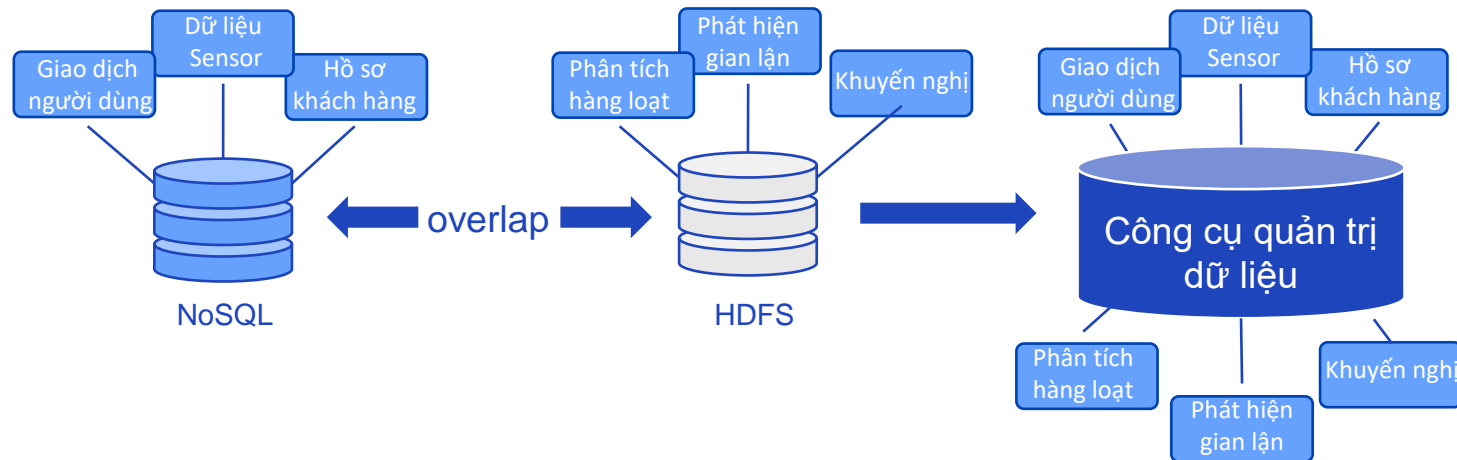
# Đặc điểm của NoSQL (1/2)

- | Truy vấn phức tạp không được hỗ trợ.
  - ▶ Nói chung, hoạt động tham gia không được hỗ trợ.
- | Sơ đồ phản ứng linh hoạt
  - ▶ Hỗ trợ không chuẩn hóa
- | Thiết kế Quy mô đám mây sẵn sàng triển khai
  - ▶ Hoạt động tốt trong các cụm
- | Không cần ACID
  - ▶ Truy vấn viết hoạt động trên tất cả các cơ sở dữ liệu và do đó lan truyền đến các máy chủ phân tán khác
  - ▶ BASE thay vì ACID.

# Đặc điểm của NoSQL (2/2)

- | BASE – Basically Available, Soft state, Eventual consistency: Có sẵn về cơ bản, trạng thái mềm, tính nhất quán cuối cùng
- | Về cơ bản, khả dụng
  - ▶ Có khả năng xảy ra lỗi, nhưng nó không trở thành lỗi của toàn bộ hệ thống.
- | Trạng thái mềm
  - ▶ Trạng thái hệ thống có thể được thay đổi mà không cần đầu vào
  - ▶ Trạng thái của một nút được xác định thông qua thông tin được truyền từ bên ngoài.
- | Tính nhất quán cuối cùng
  - ▶ Tính nhất quán cuối cùng là sự đảm bảo tính nhất quán theo thời gian.

# NoSQL VS HDFS



- ▶ Thời gian thực
- ▶ Tương Tác
- ▶ Đọc/ghi nhanh

- ▶ Hàng loạt
- ▶ Chế biến quy mô lớn
- ▶ Sức mạnh tính toán khổng lồ

# Cơ sở dữ liệu quan hệ (SQL) VS NoSQL (1/2)

## I Cơ sở dữ liệu quan hệ

- ▶ Cơ sở dữ liệu quan hệ cung cấp cơ chế lưu trữ và truy xuất dữ liệu thông qua các mối quan hệ bảng.
- ▶ Một bảng bao gồm các hàng (bản ghi, bộ dữ liệu) và cột (trường, mục).
- ▶ Tổ chức dữ liệu liên quan đến các giá trị của các trường bảng
- ▶ Ví dụ) MySQL, DB2, Oracle, MS-SQL

## I NoSQL

- ▶ Dễ dàng nhân bản và hỗ trợ API đơn giản
- ▶ Cơ sở dữ liệu phi quan hệ (Lược đồ không có bảng)
- ▶ Lý tưởng cho dữ liệu lớn và hệ thống thời gian thực
- ▶ Các loại cơ sở dữ liệu NoSQL khác nhau
- ▶ Ví dụ) HBase, Cassandra, MongoDB



## Cơ sở dữ liệu quan hệ (SQL) VS NoSQL (2/2)

	Cơ sở quan hệ dữ liệu	NoSQL
Lưu trữ dữ liệu	Mô hình quan hệ có hàng và cột	Các mô hình lưu trữ khác nhau: tài liệu, khóa-giá trị, loại cột, loại biểu đồ.
Lược đồ & tính linh hoạt	<ul style="list-style-type: none"> <li>Mỗi bản ghi tuân theo một lược đồ cố định</li> <li>Các cột phải được xác định trước</li> <li>Khi nhập dữ liệu, giá trị null được nhập ngay cả trong trường trống.</li> <li>Khi một lược đồ được thiết lập, rất khó để thay đổi nó,</li> <li>Tốn nhiều chi phí cho những thay đổi.</li> </ul>	<ul style="list-style-type: none"> <li>Lược đồ là biến</li> <li>Thay đổi theo thời gian thực khi nhập dữ liệu</li> <li>Các trường trống không có dung lượng lưu trữ</li> </ul>
Khả năng mở rộng	<ul style="list-style-type: none"> <li>TĂNG quy mô</li> <li>Yêu cầu máy chủ lớn hơn và đắt tiền hơn để mở rộng dữ liệu.</li> </ul>	<ul style="list-style-type: none"> <li>Mở rộng quy mô</li> <li>Khi mở rộng dữ liệu, có thể tăng số lượng máy chủ chung.</li> </ul>
Tuân thủ ACID	Tương thích với Tính nguyên tử, Tính nhất quán, Cách ly và Độ bền	Hy sinh ACID cho hiệu suất và khả năng mở rộng
Loại dữ liệu	Dữ liệu có cấu trúc	Cấu trúc, bán cấu trúc, phi cấu trúc
Sản phẩm	MySQL, Oracle, MS-SQL	MongoDB, HBase, Cassandra

# Khía cạnh quan trọng của RDBMS - ACID

### I Tính nguyên tử

- ▶ Mỗi giao dịch là nguyên tử. Hoạt động giao dịch không được thực hiện một phần, được thực hiện đầy đủ hoặc bị hủy bỏ (Tất cả hoặc không có gì)
- ▶ Ví dụ) Chuyển khoản ngân hàng (lỗi xử lý không thể xảy ra giữa chừng)

### I Tính nhất quán

- ▶ Tất cả các ràng buộc (ràng buộc, kích hoạt, khóa) của mô hình dữ liệu được duy trì trước và sau khi thực hiện giao dịch

### I Cách ly

- ▶ Các giao dịch không can thiệp lẫn nhau

### I Độ bền

- ▶ Khi một giao dịch được thực hiện, kết quả được đảm bảo mãi mãi.

# Ưu điểm của NoSQL so với RDBMS

- | Hỗ trợ Big data
- | Cho phép khả dụng liên tục hoặc chia nhỏ
- | Mô hình dữ liệu – hỗ trợ lược đồ linh hoạt
- | Hỗ trợ cho các cấu trúc dữ liệu khác nhau
- | Hỗ trợ khả năng mở rộng tuyến tính- mở rộng theo chiều ngang
- | Hỗ trợ cho các ngôn ngữ và nền tảng dành cho nhà phát triển chính
- | Mã nguồn mở

# Nhược điểm của NoSQL so với RDBMS

- | Thiếu sự hỗ trợ và hoàn thiện tiêu chuẩn hóa
- | Gánh nặng của các phương pháp quản lý khác nhau cho từng hệ thống
- | Tăng dung lượng lưu trữ dữ liệu với dung lượng lưu trữ dự phòng
- | Thiếu công cụ phân tích và BI

# Khi nào nên sử dụng NoSQL

- | Hỗ trợ cho nhiều ứng dụng yêu cầu các mức hiệu suất, tính nhất quán, tính khả dụng và khả năng mở rộng khác nhau
- | Các hệ thống tạo ra hàng chục terabyte dữ liệu trở lên mỗi ngày
- | Các hệ thống yêu cầu hiệu suất cao nhưng không yêu cầu tính nhất quán
- | Đảm bảo tính nhất quán cuối cùng sau một thời gian trì hoãn
- | Lưu trữ và xử lý Big Data
  - ▶ Big Data - 3V (Volume, Variety, Velocity): Khối lượng, Đa dạng, Tốc độ

# Các tính năng của NoSQL

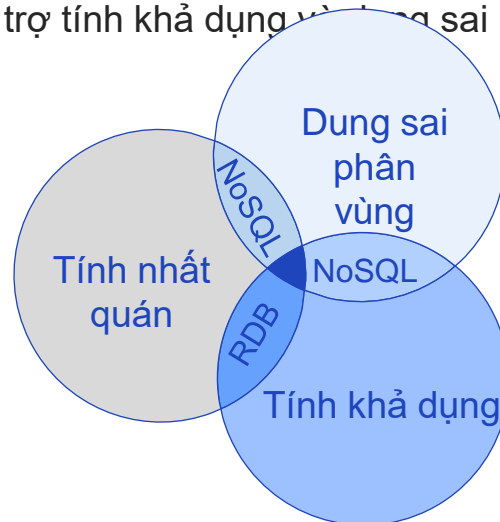
- I Không quan hệ
  - ▶ Cơ sở dữ liệu NoSQL tránh mô hình quan hệ
  - ▶ Không cung cấp bảng loại bản ghi cột cố định
  - ▶ Không cần ánh xạ quan hệ đối tượng và chuẩn hóa dữ liệu
  - ▶ Quản lý hiệu quả dữ liệu kích thước lớn và bán cấu trúc
- I Giảm đồ miễn phí
- I API đơn giản

# Định lý CAP là gì? (1/2)

- I f Theo lý thuyết CAP (C, A, P), không có hệ thống phân tán nào có thể hỗ trợ đồng thời ba thuộc tính nhất quán (consistency), khả dụng (availability) và dung sai phân vùng (partition tolerance).
  - ▶ Được đề xuất bởi Eric Brewer vào năm 2000.
- I Tính nhất quán
  - ▶ Tất cả các node cung cấp cùng một dữ liệu cho cùng một mục tại cùng một thời điểm.
- I Tính khả dụng
  - ▶ Tất cả máy khách luôn có thể thực hiện thao tác Đọc/Ghi (ví dụ: 24X7)
- I Dung sai phân vùng
  - ▶ Hệ thống hoạt động ngay cả trên các phân vùng mạng vật lý (gửi tin nhắn không thành công hoặc các bộ phận của hệ thống bị lỗi).

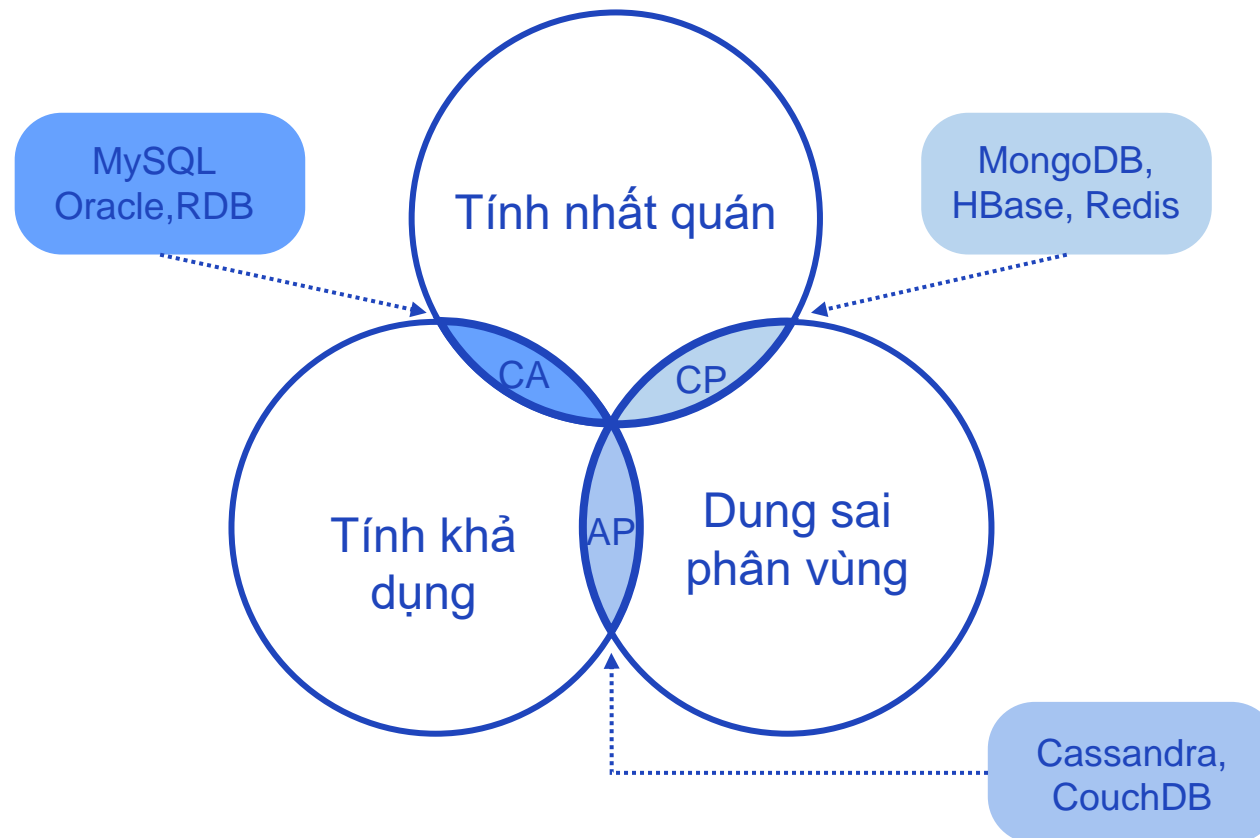
## Định lý CAP là gì? (2/2)

- | Hệ thống dữ liệu phân tán là sự cân bằng giữa tính nhất quán, tính sẵn sàng và dung sai phân vùng.
- | Mỗi cơ sở dữ liệu chỉ đảm bảo 2 trong số 3 thuộc tính trong CAP
- | Cơ sở dữ liệu quan hệ thường cung cấp tính nhất quán và tính sẵn sàng, nhưng không có dung sai phân vùng.
  - ▶ Khi một phân vùng xảy ra, hệ thống bị treo
- | Cơ sở dữ liệu NoSQL thường hỗ trợ tính khả dụng và dung sai phân vùng.



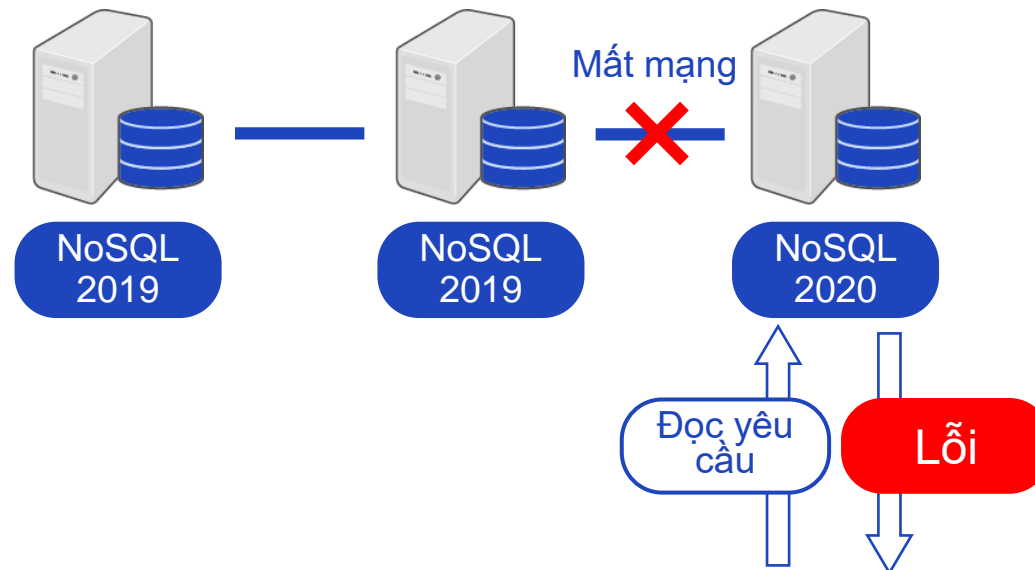


## Định lý CAP trong Cơ sở dữ liệu



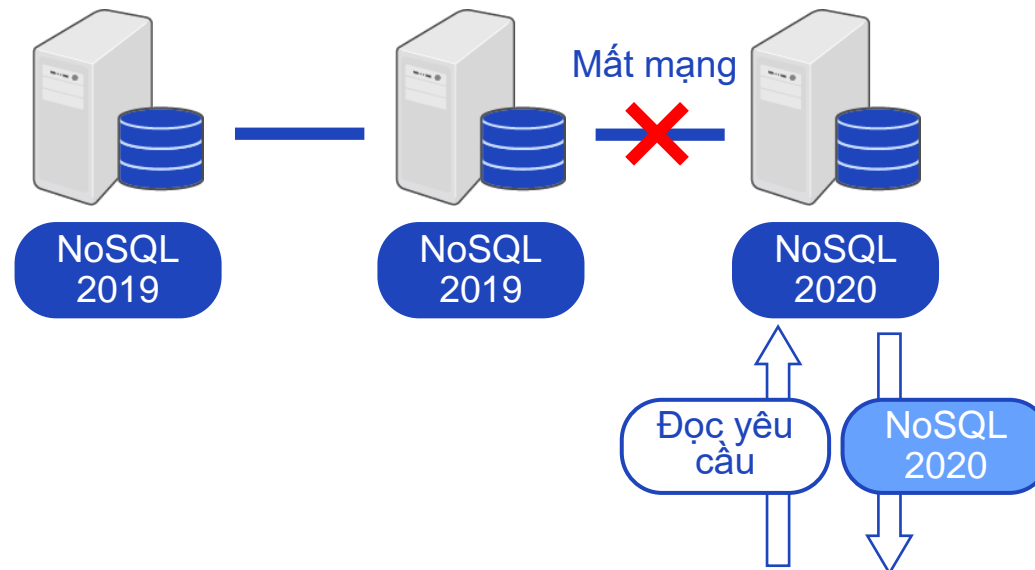
# Tính nhất quán và dung sai phân vùng

- I Đảm bảo tính nhất quán và dung sai phân vùng (CP) và từ bỏ tính khả dụng (A)
  - ▶ Cơ sở dữ liệu không khả dụng khi mạng ngừng hoạt động để đảm bảo tính nhất quán
  - ▶ Có nguy cơ một số dữ liệu không khả dụng



## Tính khả dụng và dung sai phân vùng

- I Đảm bảo tính khả dụng và dung sai phân vùng (AP) và từ bỏ tính nhất quán (C)
  - ▶ Khi nhận được yêu cầu đọc, thông tin do node được yêu cầu nắm giữ sẽ được truyền đi.
  - ▶ Bởi vì nó không kiểm tra tính nhất quán, Khách hàng có thể đọc dữ liệu không nhất quán.
- I Ví dụ) Thông tin tên miền DNS

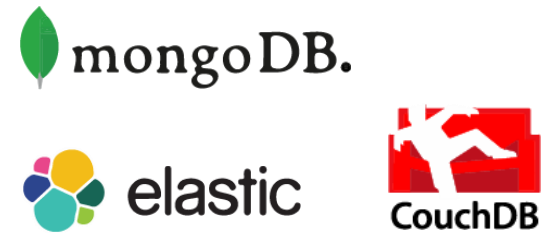


## Các loại cơ sở dữ liệu NoSQL

Loại lưu trữ cột



Loại lưu trữ tài liệu



Loại lưu trữ khóa-giá trị



Loại lưu trữ đồ thị



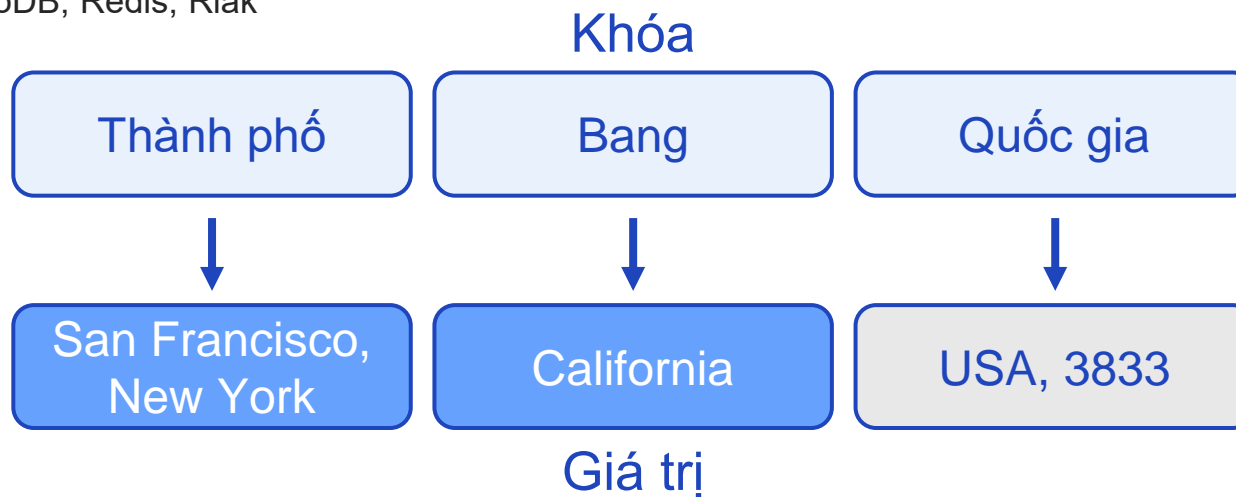
# Đặc điểm theo loại CSDL NoSQL

- | Kho lưu trữ khóa-giá trị - DynamoDB, Redis, Riak
  - ▶ Kho lưu trữ khóa-giá trị liên kết từng giá trị dữ liệu với một khóa duy nhất. Hầu hết các kho lưu trữ khóa-giá trị chỉ hỗ trợ các thao tác truy vấn, chèn và xóa đơn giản
  - ▶ Hữu ích cho truy cập ngẫu nhiên
- | CSDL hướng tài liệu- MongoDB, CouchDB, Elastic Search
  - ▶ Lưu trữ dữ liệu theo cấu trúc giống như JSON
- | Cơ sở dữ liệu cột - Apache Cassandra, HBase, BigTable
  - ▶ Sắp xếp dữ liệu của bạn thành các cột thay vì hàng
  - ▶ Thuận lợi cho việc truy vấn các tập dữ liệu lớn
- | CSDL đồ thị- neo4j, Giraph
  - ▶ Dữ liệu thành các node (đối tượng) và cạnh (mối quan hệ)

## Kho lưu trữ khóa-giá trị

I Dữ liệu được lưu trữ dưới dạng cặp khóa-giá trị

- ▶ Nó lưu trữ dữ liệu dưới dạng bảng băm trong đó mỗi khóa là duy nhất, các giá trị là JSON, BLOB (Đối tượng lớn nhị phân), chuỗi
- ▶ Một biểu mẫu có một giá trị trong một khóa duy nhất
- ▶ Làm việc dựa trên khóa chính
- ▶ Ví dụ) DynamoDB, Redis, Riak



## CSDL hướng tài liệu

### I Kho lưu trữ tài liệu/XML/đối tượng

- ▶ Dạng mở rộng của lưu trữ khóa-giá trị, không phù hợp với các truy vấn phức tạp
- ▶ Thích hợp cho các hệ thống CMS, nền tảng blog, ứng dụng phân tích thời gian thực và thương mại điện tử
- ▶ Một cấu trúc lưu trữ dữ liệu trong trường Giá trị tương ứng với Khóa và kiểu dữ liệu của Giá trị được lưu trữ là Tài liệu (XML, JSON..)
- ▶ Ví dụ) MongoDB, CouchDB, Elasticsearch

```
{
  "UID": "21f7f8de-8051-5b89-4",
  "Time": "2011-04-01T13:01:02.0",
  "Server": "A2223E",
  "Calling Server": "A2213W",
  "Type": "E100",
  "Initiating User": "dsalings@spy.net",
  "Details": {
    {
      "IP": "10.1.1.22",
      "APP": "InsertDVDQueueItem",
      "Trace": "cleansed",
      "Tags": {
        "SERVER",
        "US-West",
        "APP"
      }
    }
  }
}
```

Tài liệu 1

```
{
  "UID": "21f7f8de-8051-5b89-4",
  "Time": "2011-04-01T13:01:02.0",
  "Server": "A2223E",
  "Calling Server": "A2213W",
  "Type": "E100",
  "Initiating User": "dsalings@spy.net",
  "Details": {
    {
      "IP": "10.1.1.22",
      "APP": "InsertDVDQueueItem",
      "Trace": "cleansed",
      "Tags": {
        "SERVER",
        "US-West",
        "APP"
      }
    }
  }
}
```

Tài liệu 2

```
{
  "UID": "21f7f8de-8051-5b89-4",
  "Time": "2011-04-01T13:01:02.0",
  "Server": "A2223E",
  "Calling Server": "A2213W",
  "Type": "E100",
  "Initiating User": "dsalings@spy.net",
  "Details": {
    {
      "IP": "10.1.1.22",
      "APP": "InsertDVDQueueItem",
      "Trace": "cleansed",
      "Tags": {
        "SERVER",
        "US-West",
        "APP"
      }
    }
  }
}
```

Tài liệu 3

## CSDL cột

### I Dựa trên giấy BigTable của Google

- ▶ Dữ liệu cột được lưu cùng nhau, trái ngược với dữ liệu hàng
- ▶ Lý tưởng cho các truy vấn tổng hợp như SUM, COUNT, AVG, MIN, v.v., vì dữ liệu có sẵn trong các cột
- ▶ Ví dụ) HBase, Cassandra, BigTable

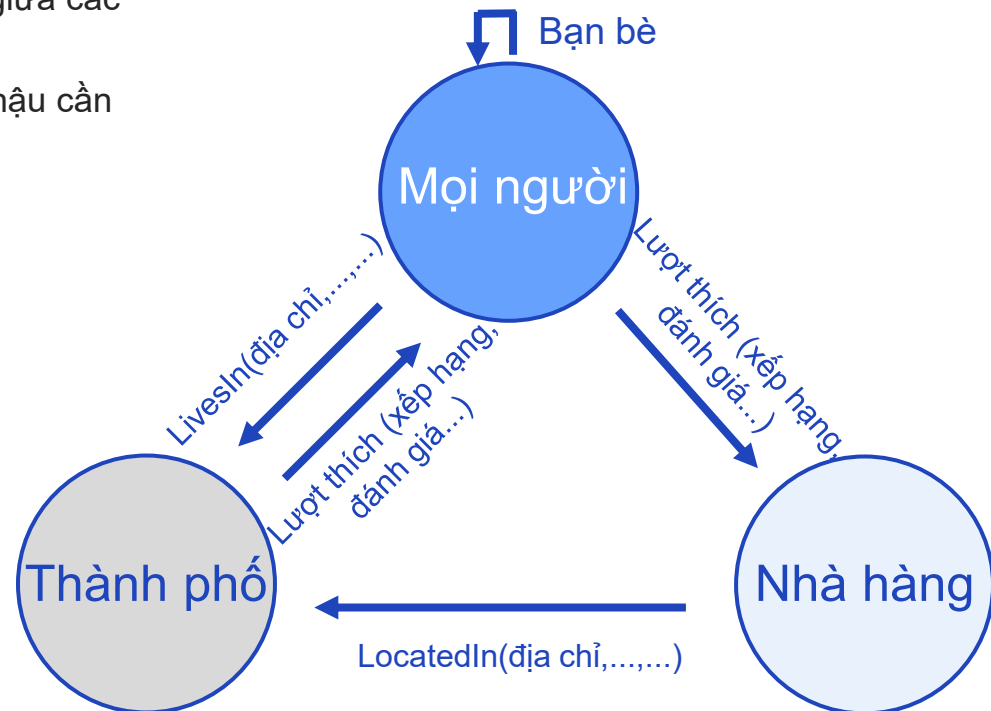
pinfo		
hàng phím	fname	lname
hkchoi	Jean	Choi
jhjeon	Scott	Jeon
jtjeon	Lonan	Jeon



# CSDL đồ thị

### I Kho lưu trữ đồ thị

- ▶ Lưu trữ dữ liệu trong cấu trúc đồ thị
- ▶ Các nút được lưu trữ độc lập và mối quan hệ giữa các node (cạnh) được lưu trữ với dữ liệu
- ▶ Chủ yếu được sử dụng cho các mạng xã hội, hậu cần và dữ liệu không gian
- ▶ Ví dụ) Neo4j, OrientDB



## Tóm tắt về Lưu trữ dữ liệu NoSQL

Mô hình dữ liệu	Hiệu suất	Khả năng mở rộng	Tính linh hoạt	Độ phức tạp	Chức năng
Lưu trữ Khóa–Giá trị	Cao	Cao	Cao	Không	Bất định (Không)
Lưu trữ hướng cột	Cao	Cao	Trung bình	Thấp	Tối thiểu
Lưu trữ hướng tài liệu	Cao	Độ bất định cao	Cao	Thấp	Bất định (Thấp)
Cơ sở dữ liệu đồ thị	Bất định	Bất định	Cao	Cao	Bất định (Không)
Cơ sở dữ liệu quan hệ	Bất định	Bất định	Thấp	Trung bình	Bất định (Không)

Bài 2.

# NoSQL

| 2.1. Tổng quan về NoSQL

| 2.2. Apache HBase

| 2.3. Cassandra

| 2.4. MongoDB

# HBase là gì?

- | HBase là NoSQL chạy trên HDFS.
- | HBase là...
  - ▶ Loại cửa hàng cột đại diện
  - ▶ Tính khả dụng cao và khả năng chịu lỗi
  - ▶ Khả năng mở rộng cao và thông lượng cao
  - ▶ Dễ dàng xử lý các bảng lớn
  - ▶ Dữ liệu với các hàng phân tán với số lượng lớn các cột
  - ▶ Mã nguồn mở, dự án Apache
- | Các tính năng được cung cấp bởi HDFS
  - ▶ Khả năng chịu lỗi
  - ▶ Khả năng mở rộng



# Thuật ngữ Hbase

- | Node
  - ▶ Một máy chủ, máy
- | Cụm
  - ▶ Một nhóm các node trong đó nhiều node được kết nối và cộng tác để thực hiện một tác vụ
- | Master Node
  - ▶ Các node thực hiện các hoạt động phối hợp
- | Worker Nodes
  - ▶ Một node thực hiện các nhiệm vụ được giao bởi master node.
- | Trình nền
  - ▶ Một quy trình hoặc chương trình chạy trong nền

# Tổng quan về HBase (1/2)

- | Lưu trữ dữ liệu trong bảng HBase
  - ▶ Tương tự như các bảng RDBMS, nhưng có những điểm khác biệt chính
- | Các bảng được lưu trữ trong HDFS
  - ▶ Dữ liệu được phân vùng thành các khối HDFS và được lưu trữ trên nhiều nút trong cụm.
- | Các bảng được tạo thành từ các hàng, cột và họ cột.
- | Tất cả các hàng bao gồm hàng phím để truy xuất nhanh
- | Các cột lưu trữ dữ liệu trong một bảng
- | Mỗi cột thuộc về một họ cột cụ thể
- | Một bảng chứa một hoặc nhiều họ cột.

# Tổng quan về Hbase (2/2)

- | Bản đồ được phân phối, sắp xếp nguyên bản
- | Phân phối  
HBase được thiết kế để sử dụng nhiều máy để lưu trữ và phục vụ dữ liệu bảng.
- | Bản đồ được sắp xếp
  - ▶ Lưu trữ dữ liệu bảng dưới dạng bản đồ, với các khóa liên kết được lưu trữ liên kết trên đĩa
- | Đa chiều
  - ▶ Hàng trăm hoặc hàng nghìn cột có khả năng được lưu trữ trong một họ cột

# Hàng HBase

- | hàng phím tương tự như khóa chính của RDBMS hiện có
- | Sắp xếp và lưu trữ các hàng theo hàng phím để truy xuất dữ liệu nhanh
  - ▶ Các hàng được sắp xếp theo thứ tự bảng chữ cái theo khóa
- | hàng phím là một chuỗi từ 100 byte trở xuống
  - ▶ VD) 1, 10, 2, ...

pinfo		
hàng phím	fname	lname
hkchang	Jean	Choi
jhjeong	Scott	Jeong
thjtjeon	Lonan	Jeon



## Cột HBase

- I Tương tự như các cột RDBMS (Hệ quản trị CSDL), nhưng có các đặc điểm khác
- I Các cột được tạo khi cần (so với RDBMS)
  - ▶ Các cột có thể được thêm vào bất kỳ lúc nào miễn là họ cột tồn tại
  - ▶ Một cột chỉ tồn tại nếu hàng chứa dữ liệu trong cột đó.
  - ▶ Một ô của bảng (giao điểm hàng-cột) là một mảng byte tùy ý (Mảng)
- I Mỗi cột trong một bảng thuộc về một họ cột cụ thể.
  - ▶ Tập hợp các cột
- I Một hoặc nhiều họ cột tồn tại trong một bảng, được viết dưới dạng định nghĩa bảng
  - ▶ Bao gồm ít nhất một
- I Các cột được nhóm thành các họ cột

k			
			X
	X		

# Họ cột HBase

- I Tất cả các cột thuộc cùng một họ cột có cùng tiền tố
  - ▶ VD) pinfo:fname and pinfo:lname
  - ▶ “:” is used as a column name and column family separator
- I Cài đặt điều chỉnh và lưu trữ cho từng họ cột
  - ▶ VD) Số lượng phiên bản của mỗi ô sẽ được lưu
- I Họ cột có thể có nhiều cột
  - ▶ Các cột trong một họ cột được sắp xếp và lưu trữ cùng nhau

# Bảng HBase

- I Các cột được tham chiếu bằng họ cột và tên cột.
- I Khi sử dụng các họ cột riêng biệt là hữu ích
  - ▶ Dữ liệu được truy cập không thường xuyên
  - ▶ Dữ liệu sử dụng các tùy chọn họ cột khác nhau
  - ▶ ví dụ: nén

Họ cột		pinfo		pimage	
Tên cột		hàng phím	fname	lname	Ảnh
Hàng		hkchoi	Jean	Choi	
		jhjeon	Scott	Jeong	<scott.jpg>
		thjeon	Lonan	Jeon	<lonan.jpg>

# Lưu trữ dữ liệu trong bảng

- Dữ liệu được lưu trữ vật lý trên đĩa theo đơn vị của họ cột.

	pinfo	
hàng phím	fname	lname
hkchoi	Jean	Choi
jhjeon	Scott	Jeon
jtjeon	Lonan	Jeon

Tập

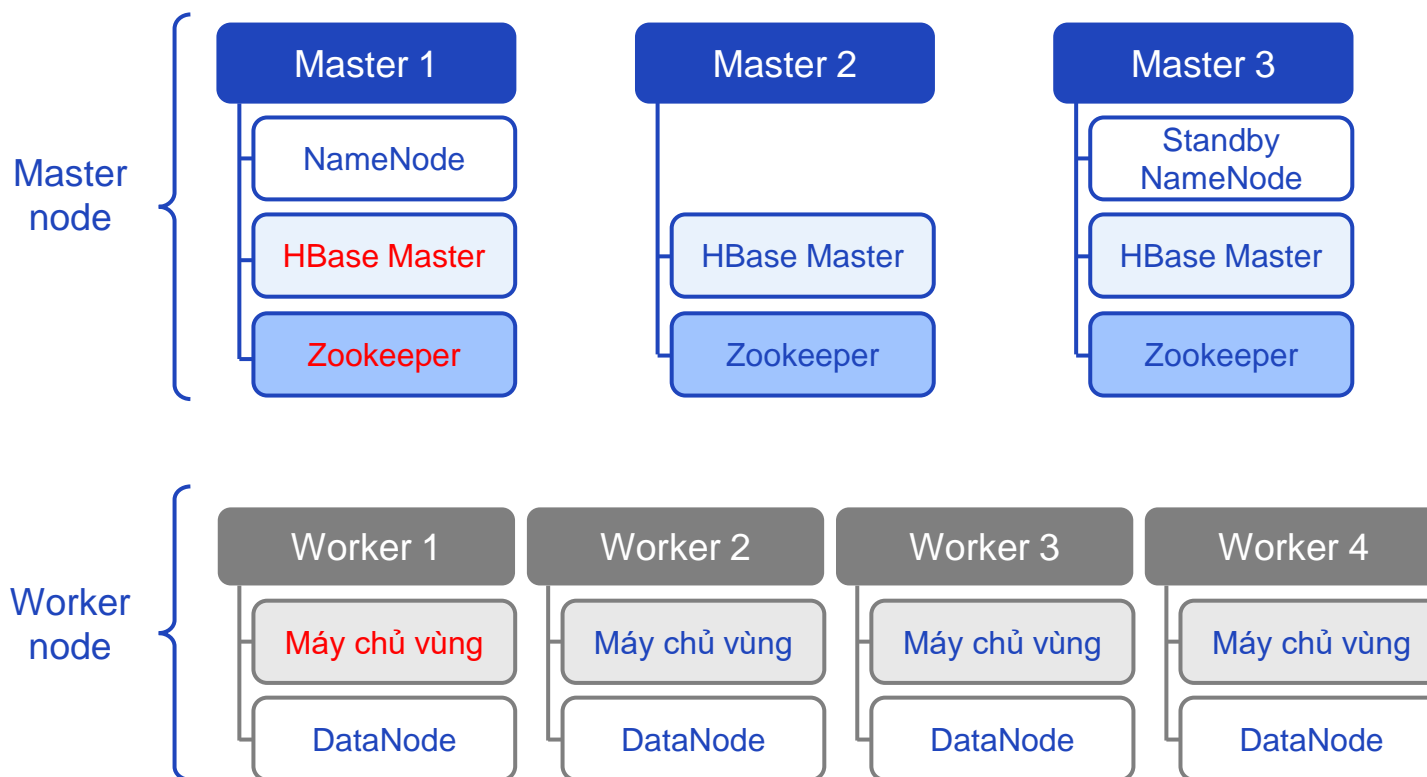
- Bảng là giá trị, nội dung của ô
  - hàng phím + cột + dấu thời gian

hàng phím	Cột	Dấu thời gian	Giá trị ô
hkchoi	pinfo:fname	102638498231	Scott
jhjeon	pinfo:lname	1026753402919	Jeong

# Thành phần HBase

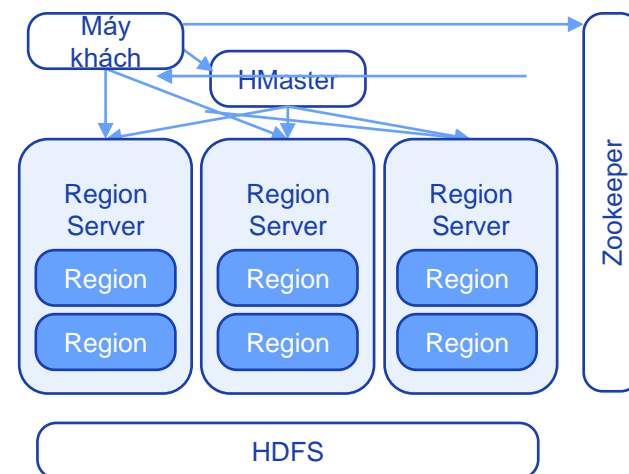
- | Máy chủ Vùng
  - ▶ Quản lý và bảo trì máy chủ Vùng
- | HBase Master
  - ▶ Giám sát tất cả các phiên bản Máy chủ vùng trên giao diện cụm để biết tất cả các thay đổi siêu dữ liệu
- | ZooKeeper
  - ▶ Khung điều phối được sử dụng để quản lý tập trung thông tin cấu hình cho HBase
  - ▶ Hỗ trợ quyết định cho một Master duy nhất
- | NameNode
  - ▶ Quy trình quản lý để quản lý siêu dữ liệu HDFS
- | DataNode
  - ▶ Một tiến trình nền lưu trữ và giữ các khối HDFS.

## Trình nền và Dịch vụ trong Cụm HBase



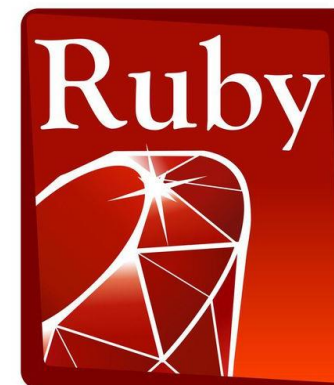
## Vùng HBase và Máy chủ Vùng

- | Bảng HBase được chia thành các Vùng
  - ▶ Phân vùng bảng
  - ▶ Tương tự như phân đoạn và phân vùng Hệ quản trị CSDL hiện có
- | Các vùng được cung cấp cho khách hàng bởi quy trình Máy chủ vùng.
- | Máy chủ vùng thường chạy trên nút làm việc của cụm
- | Các nút dữ liệu thường có một số máy chủ khu vực đang chạy.
- | Máy chủ vùng thường phục vụ nhiều vùng
  - ▶ Vùng được cung cấp thuộc về một số bảng khác nhau.
  - ▶ Không chắc rằng một Máy chủ Vùng sẽ phục vụ tất cả các vùng của một bảng cụ thể.



# Thông tin cơ bản về HBase Shell

- I HBase shell xử lý các lệnh tương tác để chuyển lệnh tới HBase.
  - ▶ Sử dụng HDFS
  - ▶ Hmaster, máy chủ vùng, liên kết vùng
  - ▶ Hầu hết các hoạt động có sẵn trong API có thể được thực hiện trong shell.
- I HBase shell sử dụng JRuby
  - ▶ Kết thúc các cuộc gọi máy khách Java trong Ruby (Chương trình Ruby trong JVM)
  - ▶ Sử dụng cú pháp Ruby cho các lệnh
  - ▶ Cách các tham số được sử dụng khác với các shell khác.
  - ▶ Các tham số lệnh được đặt trong dấu nháy đơn (').



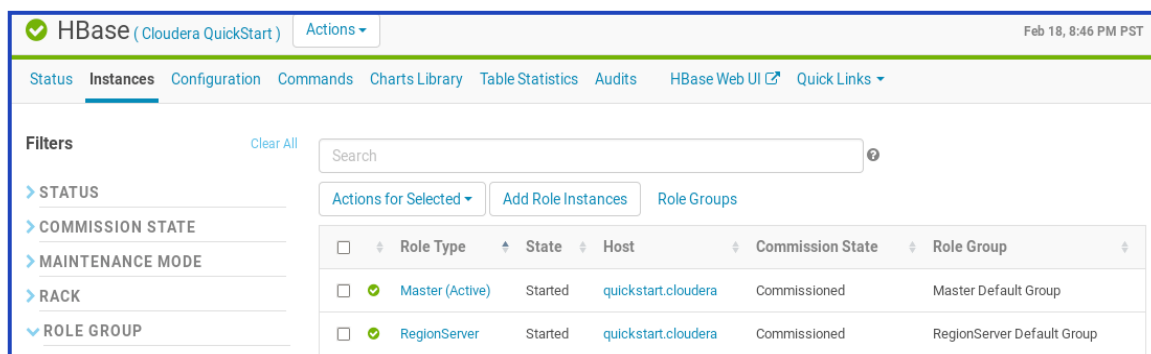
Programming  
Language



# Cách sử dụng HBase Shell (1/2)

## I Thực thi HBase shell

```
$ hbase shell
```



The screenshot shows the HBase Cloudera QuickStart web interface. The 'Instances' tab is selected, displaying a table of HBase instances. The table has columns for Role Type, State, Host, Commission State, and Role Group. Two instances are listed: Master (Active) and RegionServer, both in a 'Started' state and 'Commissioned'.

<input type="checkbox"/>	Role Type	State	Host	Commission State	Role Group
<input type="checkbox"/>	Master (Active)	Started	quickstart.cloudera	Commissioned	Master Default Group
<input type="checkbox"/>	RegionServer	Started	quickstart.cloudera	Commissioned	RegionServer Default Group

## I Lệnh cơ bản HBase

- ▶ help, status, version, whoami
- ▶ Exit(terminate) + <Return>

## Cách sử dụng HBase Shell (2/2)

### I Tính tương tác của Ruby shell

```
[cloudera@quickstart ~] $ hbase shell
20/02/18 20:37:01 INFO Configuration.deprecation: Hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0> version
1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):002:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):003:0> exit
[cloudera@quickstart ~]$ hbase shell
20/02/18 20:37:58 INFO Configuration.deprecation: Hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0>
```

## Lệnh shell (1/2)

### I Lệnh Shell có thể lấy tham số

- ▶ <return> sau khi nhập lệnh
- ▶ Không giống như trình bao Bash, dấu nháy đơn ( ' ' ) được sử dụng.
- ▶ Các cuộc gọi nhiều tham số yêu cầu dấu ngoặc nhọn, tên khóa, giá trị trong dấu nháy đơn và dấu phẩy (,)

```
Hbase> command 'param1', 'param2'
```

```
hbase(main):001:0> create 'test', 'pinfo'  
0 row(s) in 1.5330 seconds
```

```
=>Hbase::Table - test  
hbase(main):002:0> list 'test'  
TABLE  
test  
1 row(s) in 0.0310 seconds  
  
=> ["test"]
```

## Lệnh Shell (2/2)

### I Tạo và thay đổi bảng sử dụng hàm băm Ruby

- ▶ {'key1' => 'value1', 'key2' => 'value2', ... }
- ▶ Toán tử “=>” là một tên lửa băm và phân tách các khóa và giá trị.
- ▶ Hàm băm phải kết thúc bằng một dấu ngoặc nhọn khác

```
create 't1', {NAME => 'f1', VERSIONS => 2, BLOCKCACHE => true}
```

### I Loại lệnh Shell

- ▶ Lệnh chung
- ▶ Các lệnh định nghĩa dữ liệu (create, list, drop, alter...)
- ▶ Các lệnh thao tác dữ liệu (Put, Get, Scan...)

# Shell scripting

- I HBase shell chạy ở chế độ tương tác và hàng loạt
  - ▶ Viết một kịch bản với JRuby và chuyển nó vào trình bao
  - ▶ Được truyền dưới dạng tham số khi thực thi hbase shell hoặc từ shell

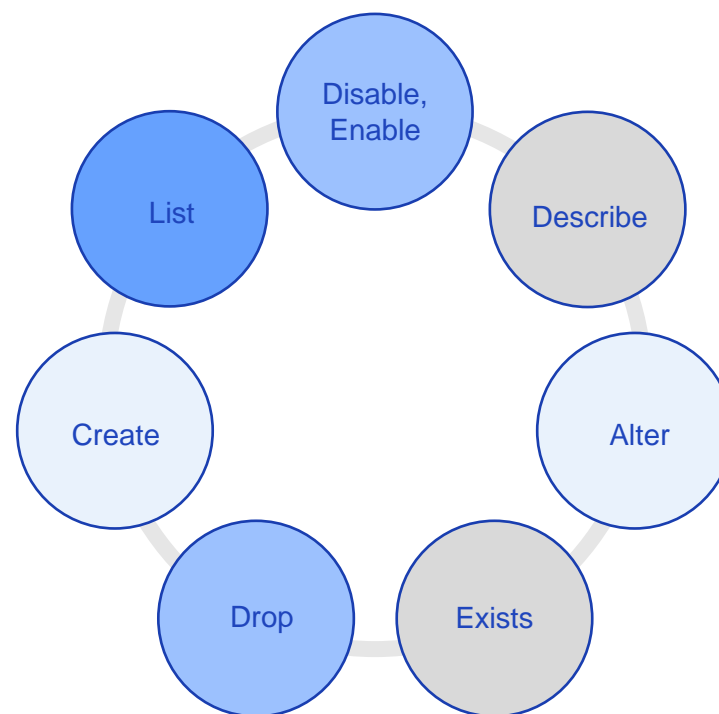
```
$ hbase shell rubyscript.rb
```

```
hbase> require 'rubyscript.rb'
```

# Các thao tác HBase trên bảng

## I Lệnh định nghĩa dữ liệu

- ▶ Create: Tạo
- ▶ List: Liệt kê
- ▶ Disable, Enable: Vô hiệu hóa, Kích hoạt
- ▶ Describe: Mô tả
- ▶ Alter: Thay đổi
- ▶ Exists: Tồn tại
- ▶ Drop: Bỏ

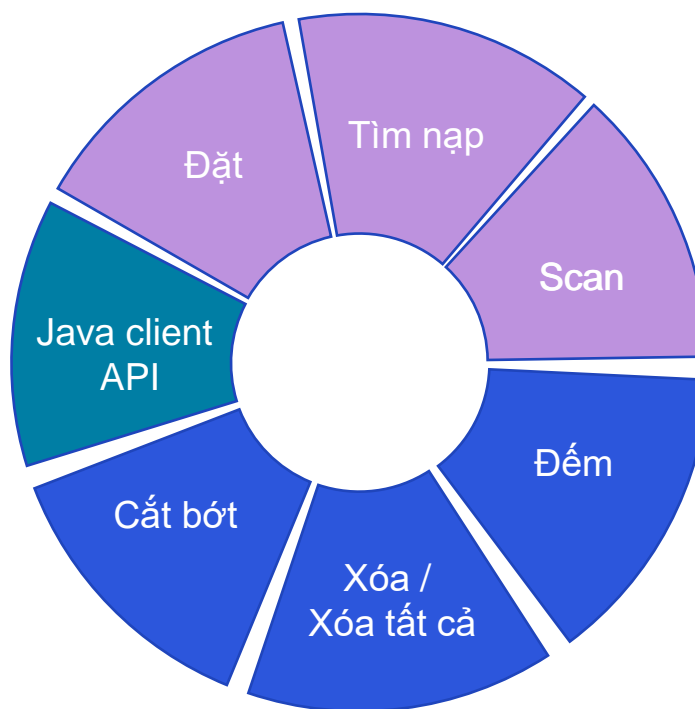


# Tạo bảng

- I Tên bảng, họ cột, tùy chọn và cấu hình bảng
  - ▶ Đặc tả yêu cầu của bảng và họ cột khi tạo bảng
  - ▶ Tạo '<ten bảng>', '<họ cột>'
  - ▶ Ít nhất một họ cột được yêu cầu cho mỗi bảng.
  - ▶ Cột là một chuỗi hoặc từ điển đơn giản và phải chứa thuộc tính NAME.
- I Các bảng có thể được nhóm thông qua tính năng không gian tên mới
  - ▶ Khái niệm “cơ sở dữ liệu” và “lược đồ” trong cơ sở dữ liệu quan hệ
- I Sự khác biệt từ Hệ quản trị CSDL
  - ▶ Không cần tạo cột hoặc mối quan hệ
  - ▶ Không cần chỉ định các mối quan hệ hoặc ràng buộc chính

## Lệnh thao tác dữ liệu

- | Put: Đặt
- | Get: Tìm nạp
- | Scan: Quét
- | Count: Đếm
- | Delete / deleteall: Xóa / Xóa tất cả
- | Truncate: Cắt bớt
- | Java client API





# Tạo dữ liệu

## I Put: Đặt

- ▶ Nhập một hàng mới
- ▶ Được sử dụng để thay đổi dữ liệu hiện có
- ▶ Nếu hàng phím tồn tại, nó sẽ sửa đổi giá trị và dấu thời gian hiện có.  
Nếu một họ cột được xác định để chỉ giữ một phiên bản của mỗi giá trị cột, thì chỉ giữ giá trị mới.  
Nếu xác định n phiên bản được lưu giữ thì n giá trị được lưu giữ, bao gồm cả giá trị mới.
- ▶ Cập nhật cho một cột cụ thể không thay đổi các cột khác trong hàng

```
Put 'table name', 'rowkey', 'columnfamily:column', 'value' [,timestamp]
```

```
hbase> put 'test','1','pinfo:name', 'Jean'  
hbase> put 'test','2','position:title', 'Engineer'
```

# Đọc dữ liệu

## I Get: Tìm nạp

- ▶ Nhận phiên bản mới nhất của nội dung hàng hoặc cột tương ứng với hàng phím
- ▶ Chỉ định hàng phím chính xác sẽ được tìm nạp

```
get 'table name', 'rowkey' [, options]
```

```
hbase(main):008:0> get 'test', '1'
COLUMN                                CELL
  pinfo:city                          timestamp=1582167226850, value=Seoul
  pinfo:name                          timestamp=1582166931093, value=Jean
  position:title                      timestamp=1582167310073, value=Manager
3 row(s) in 0.0210 seconds
```

```
hbase(main):009:0> get 'test', '2', {COLUMN => 'pinfo:name'}
COLUMN                                CELL
  pinfo:city                          timestamp=1582167455859, value=Scott
1 row(s) in 0.0050 seconds
```

```
hbase(main)010:0> get 'test', '3', {COLUMN => ['pinfo']}
COLUMN                                CELL
  pinfo:name                          timestamp=1582167555280, value=Lonan
1 row(s) in 0.0060 seconds
```

## Quét và đếm dữ liệu

### I Scan: Quét

- ▶ Nếu bạn không biết hàng phím chính xác
- ▶ Nếu bạn cần quyền truy cập vào một nhóm hàng

### I Quét toàn bộ bảng

- ▶ Có thể bị hạn chế đối với các phím dòng bắt đầu và dòng dừng
- ▶ Các hàng dừng không được bao gồm trong kết quả, chỉ có hàng trước đó được tìm kiếm.
- ▶ Có thể bị hạn chế đối với các cột và họ cột cụ thể
- ▶ Bạn có thể giới hạn kết quả bằng bộ lọc

```
scan 'table name' [, options]  
get 'table name', 'rowkey' [, options]
```

### I Count: Đếm

- ▶ Đếm số hàng

```
count 'table name'
```

# Xóa dữ liệu

## I Delete: Xóa

- ▶ Xóa một ô, cột hoặc họ cột cụ thể khỏi bảng
- ▶ Việc thêm dấu thời gian sẽ xóa cột của phiên bản đó và tất cả các phiên bản trước đó

```
delete 'table name', 'rowkey', 'column family:column name', timestamp
```

```
hbase> delete 'test','1','position:title', 1582167310073
```

## I Deleteall: Xóa tất cả

- ▶ Xóa tất cả các ô trong một hàng


```
Deleteall 'table name', 'rowkey'
```

# Phiên bản

## I HBase lưu trữ các phiên bản của giá trị trong mỗi ô

- ▶ Số lượng phiên bản được xác định khi thiết lập họ cột.
- ▶ Phiên bản được xác định bằng dấu thời gian loại dài và dấu thời gian hiện tại được sử dụng làm mặc định.
- ▶ Số lượng phiên bản ô mặc định là 3
- ▶ Mỗi hàng được sắp xếp theo dấu thời gian (giảm dần)

hàng phím	Cột	Dấu thời gian	Giá trị
1	pinfo:title	1582172284996	Kỹ sư
1	pinfo:title	1582167455859	Giám đốc
1	pinfo:title	1582167226850	Kỹ sư



## I Giá trị

- ▶ hàng phím → Họ cột → Cột → Phiên bản

## Đánh đổi thiết kế chính

### I Salted: Bảo mật

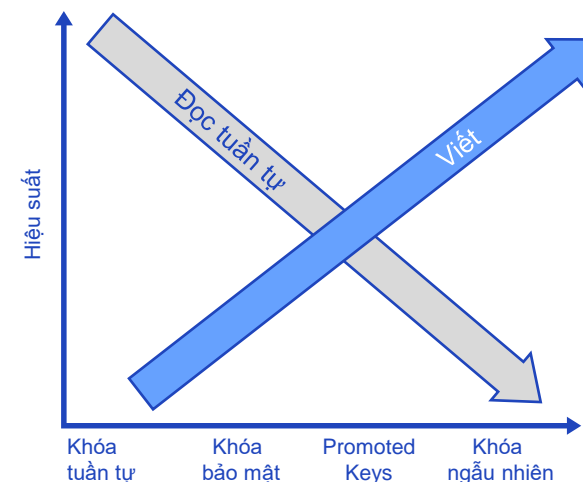
- ▶ Đặt một hàm băm nhỏ, được tính toán trước dữ liệu thực để ngẫu nhiên hóa hàng phim
- ▶ Ví dụ: <salt><timestamp> thay vì chỉ <timestamp>

### I Trường khóa được thúc đẩy

- ▶ Một trường được di chuyển trước trường gia tăng hoặc dấu thời gian
- ▶ Ví dụ: <sourceid><timestamp> thay vì <timestamp><sourceid>

### I Ngẫu nhiên

- ▶ Xử lý dữ liệu bằng hàm băm một chiều như MD5
- ▶ Ví dụ: <md5(timestamp)> thay vì chỉ <timestamp>



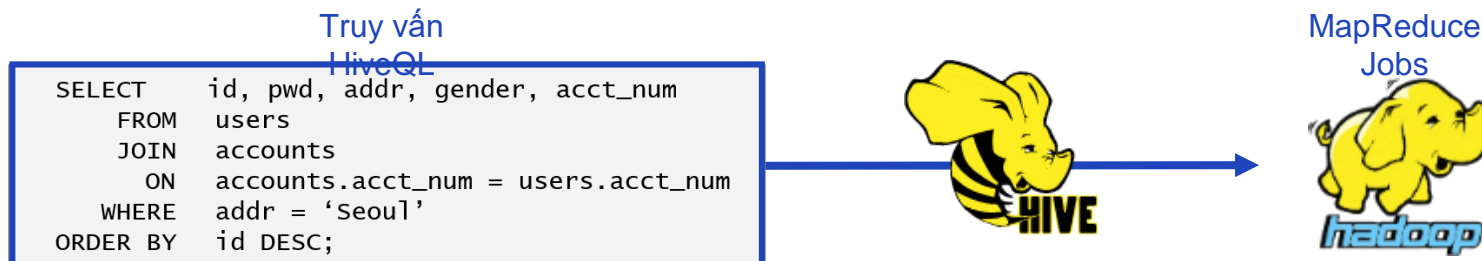
# VERSIONS & TTL(Time-To-Live): Phiên bản & TTL (Thời gian tồn tại)

- I Theo mặc định, một phiên bản ô được giữ.
  - ▶ HBase tự động chèn dấu thời gian vào từng phiên bản của ô.
  - ▶ Chỉ những ô được cam kết gần đây nhất mới được giữ lại.
- I Thuộc tính TTL được sử dụng làm thời gian hết hạn (giây) cho giá trị của ô.
  - ▶ Hàng được giữ lại cho đến khi người dùng xóa chúng, nhưng các hàng hết hạn sẽ tự động bị xóa.
  - ▶ Thuộc tính TTL có thể được đặt cho từng họ cột.

## Hive và HBase (1/2)

### I Hive

- ▶ Được sử dụng để truy cập dữ liệu HDFS bằng cú pháp giống như SQL
- ▶ Trong quá trình xử lý Hadoop, nó được thay đổi thành Map-Reduce thực tế và được thực thi.
- ▶ Sử dụng công cụ Map-Reduce và Tez
- ▶ Dịch vụ xử lý/phân tích dữ liệu do Facebook phát triển



- ### I Mã HBase thường được viết bằng java hoặc các ngôn ngữ khác sử dụng tiết kiệm
- ▶ Phân tích đơn giản bằng HBase shell



# Hive và HBase (2/2)

### I Yêu cầu sử dụng HBase

- ▶ Một lập trình viên giỏi là người hiểu mã
- ▶ Hiểu cách thức hoạt động của HBase
- ▶ Nhân sự có thể viết và duy trì mã API chương trình

### I Cách để sử dụng nó một cách dễ dàng với các công cụ

- ▶ Cách lưu trữ và truy xuất dữ liệu mà không cần biết chi tiết về HBase API
- ▶ Có sẵn qua Hive/Impala/presto

# Sử dụng Hive với HBase

## I Thực hiện Hive trong beeline

- ▶ Tạo bảng “lớp” làm bảng bên ngoài

```
$ beeline -u jdbc:hive2://localhost:10000

beeline> CREATE EXTERNAL TABLE test
(rowkey string, name string, city string, title string)
STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping"
=":key,pinfo:name, pinfo:city, position:title")
TBLPROPERTIES
("hbase.table.name" = "test");

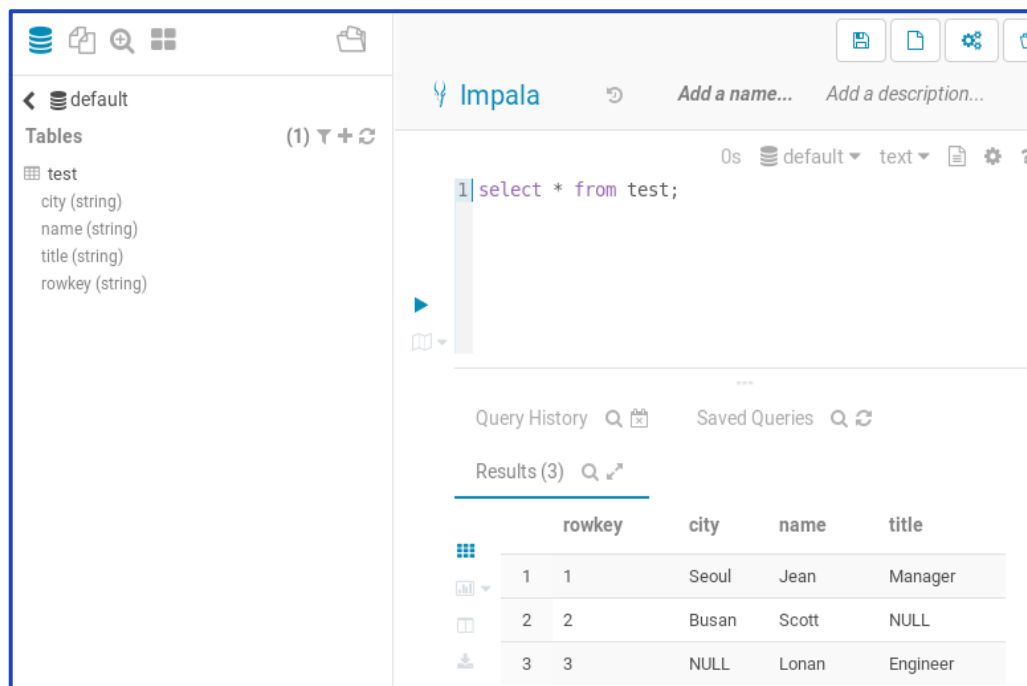
beeline> DESCRIBE test;

Beeline> select * from test;
```

# Sử dụng Impala với HBase

## I Thực thi Impala trong Hue Query Editor

- ▶ Impala chia sẻ metastore với Hive



The screenshot shows the Hue Query Editor interface. On the left, a sidebar lists tables under the 'default' database. A table named 'test' is selected, showing its schema: city (string), name (string), title (string), and rowkey (string). The main editor area shows the Impala query: `select * from test;`. Below the query, the 'Results (3)' section displays the query output as a table:

	rowkey	city	name	title
1	1	Seoul	Jean	Manager
2	2	Busan	Scott	NULL
3	3	NULL	Lonan	Engineer

Bài 2.

# NoSQL

| 2.1. Tổng quan về NoSQL

| 2.2. Apache HBase

| 2.3. Cassandra

| 2.4. MongoDB

## Định nghĩa Cassandra

Apache Cassandra là một cơ sở dữ liệu mã nguồn mở, được phân phối, phi tập trung, có khả năng mở rộng linh hoạt, có tính khả dụng cao, chịu lỗi, nhất quán có thể điều chỉnh được, dựa trên thiết kế phân phối của nó trên Dynamo của Amazon và mô hình dữ liệu của nó trên Bigtable của Google. Được tạo tại Facebook, nó hiện được sử dụng tại một số trang web phổ biến nhất trên Web

The Definitive Guide, Eben Hewitt, 2010

Google

Bigtable, 2006



cassandra

OpenSource, 2008

amazon

Dynamo, 2007

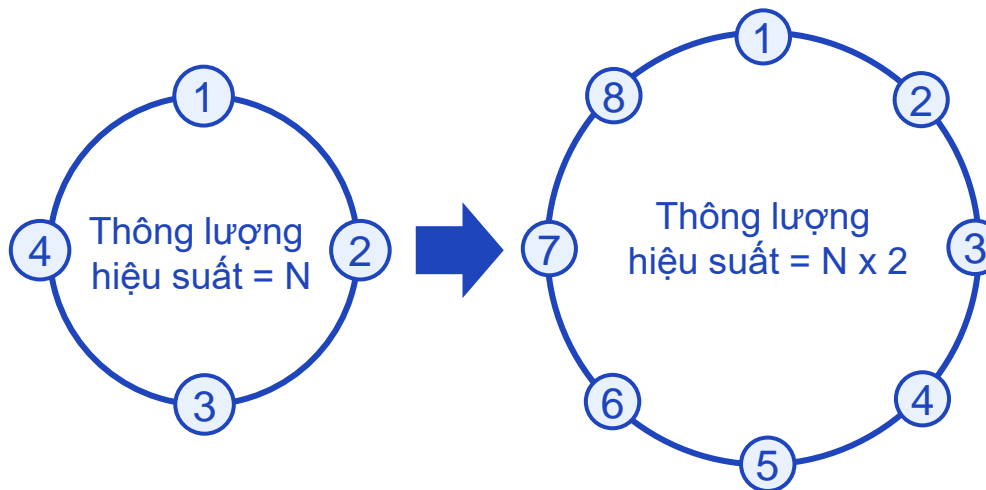
## Các tính năng chính

- I Cassandra là kho lưu trữ dữ liệu phân tán được tối ưu hóa cho khả năng mở rộng và tính khả dụng cao.
  - ▶ Giao thức Ring và Gossip
  - ▶ Phản hồi lỗi mà không có cấu trúc sharding và master-slave

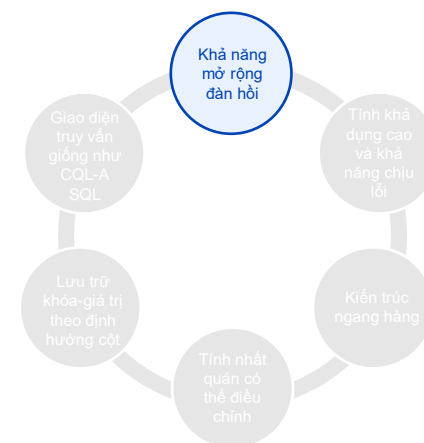


## Khả năng mở rộng đàn hồi

- Cassandra mở rộng quy mô để thêm các hệ thống chứa tất cả hoặc một phần dữ liệu
- Thêm các nút làm tăng thông lượng hiệu suất một cách tuyến tính
- Thật dễ dàng để tăng và loại bỏ số lượng nút mà không cần cấu hình lại toàn bộ cụm.



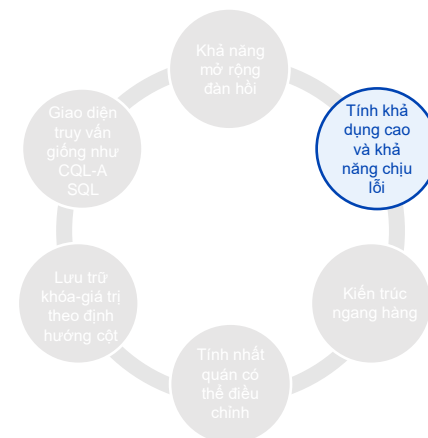
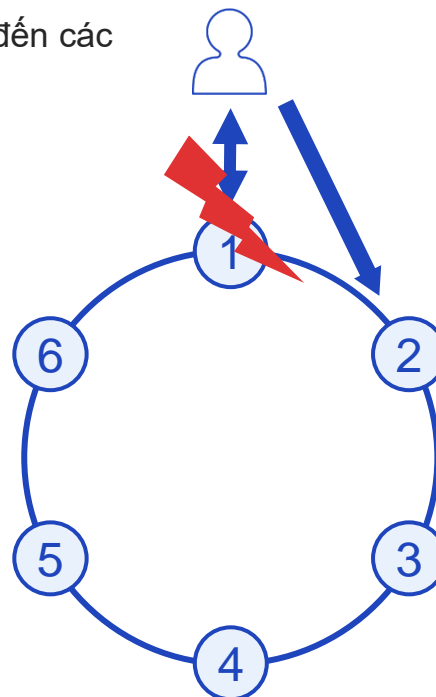
[Tỷ lệ tuyến tính thành terabyte và petabyte dữ liệu]



# Tính khả dụng cao và khả năng chịu lỗi

## I Tính khả dụng cao của Cassandra

- ▶ Nhiều máy tính mạng làm việc trong một cụm
- ▶ Nhận biết lỗi node
- ▶ Chuyển đổi dự phòng bằng cách chuyển tiếp yêu cầu đến các phần khác của hệ thống

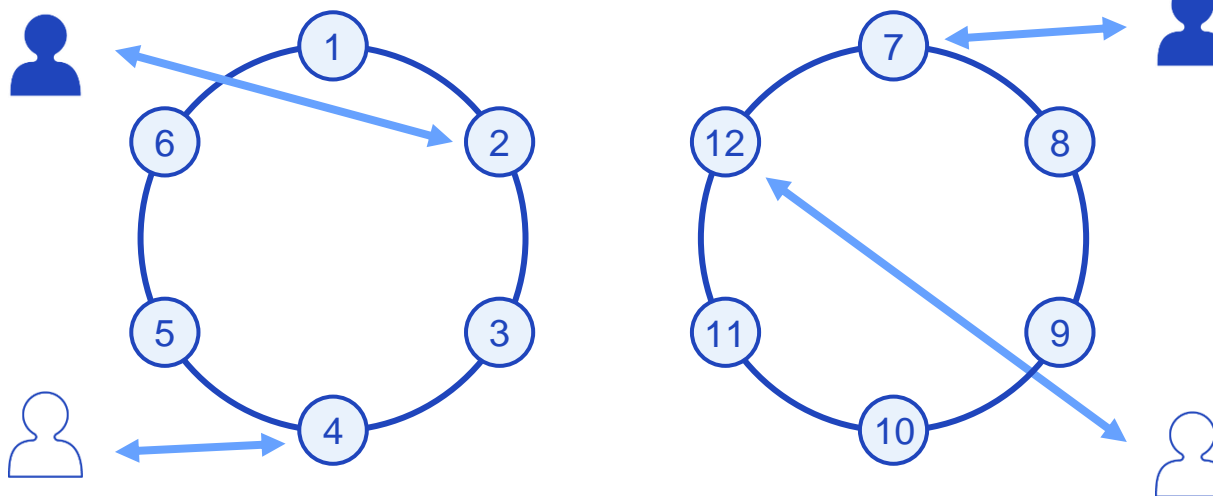




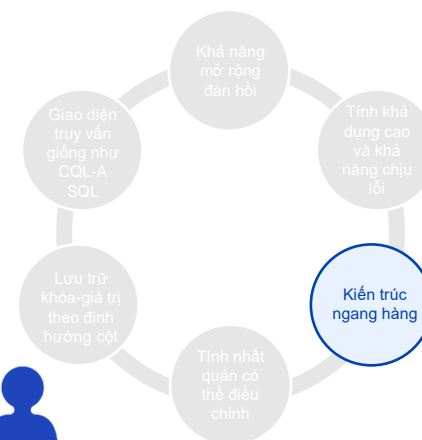
# Phân tán và phân quyền

## I Decentralized: SPOF does not occur

- ▶ No master-slave problem with peer-to-peer structure (protocol "gossip")
- ▶ A single Cassandra cluster can run in geographically dispersed data centers

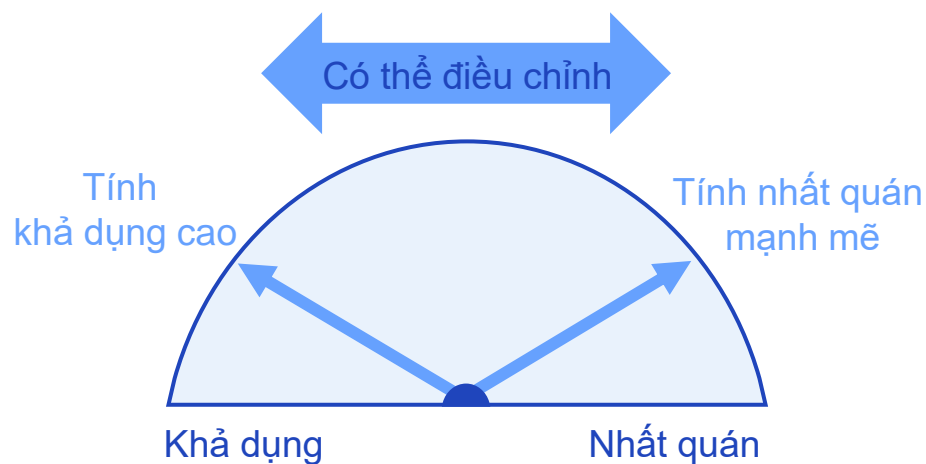


[Yêu cầu đọc và ghi tới bất kỳ node nào]

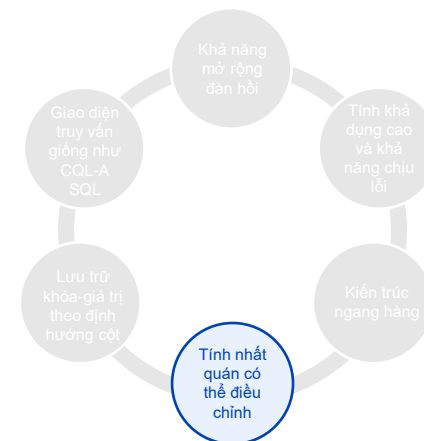


## Tính nhất quán có thể điều chỉnh

- I Chọn giữa tính nhất quán Mạnh/cuối cùng (Có sẵn so với Tính nhất quán)
- I Đọc (thao tác đọc) và Viết (thao tác ghi) có thể được điều chỉnh riêng
- I Hơi hy sinh Tính nhất quán để tăng Tính khả dụng tổng thể.
  - Điều chỉnh Mức độ nhất quán để cân bằng tính khả dụng

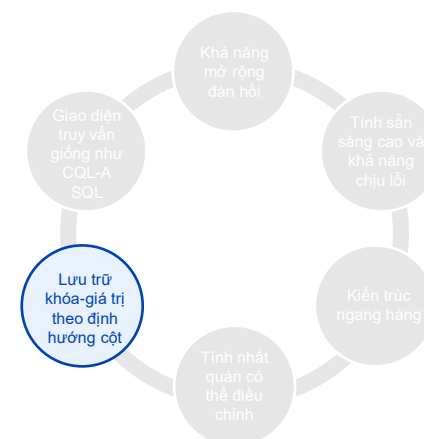


[Mức độ nhất quán phụ thuộc vào trường hợp sử dụng]



## Lưu trữ hướng cột

- I Dữ liệu được lưu trữ trong nhiều bảng băm
- I Một hàng chứa nhiều cột, nhưng mỗi hàng không nhất thiết phải có cùng số cột.
- I Mỗi hàng có một khóa duy nhất, được sử dụng để phân vùng
  - ▶ Các hàng phím cung cấp giá trị băm để phân phối dữ liệu đồng đều trên cụm.



Được lưu trữ sắp xếp theo khóa/giá trị cột →

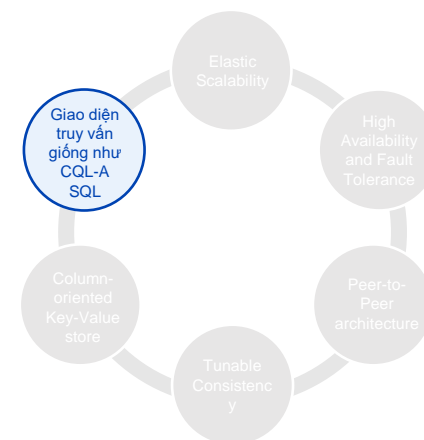
- I Không có quan hệ đối với các bảng

Được lưu trữ sắp xếp theo hàng phím ↓

Bảng				
Hàng phím 1	Khóa cột 1	Khóa cột 2	Khóa cột 3	...
	Giá trị cột 1	Giá trị cột 2	Giá trị cột 3	...
		⋮		

## Giao diện truy vấn giống như CQL-A SQL

- CQL là giao diện chính của Cassandra DBMS.
- Ảnh xạ tới công cụ lưu trữ Cassandra và sử dụng cú pháp giống như SQL để đơn giản hóa việc lập mô hình dữ liệu



```
CREATE TABLE song_col (  
  id uuid PRIMARY KEY,  
  title text,  
  album text,  
  artist text,  
  data blob,  
  tags set<text>  
);
```

```
INSERT INTO song_col (  
  (id, title, artist, album, tags)  
VALUES(  
  'a3e64f8f...',  
  'La Grange',  
  'ZZ Top',  
  'Tres Hombres'  
  {'cool', 'hot'});
```

```
SELECT *  
FROM song_col  
WHERE id = 'a3e63f8f...';
```

["giống SQL" nhưng KHÔNG phải SQL quan hệ]

# Sử dụng Cassandra ở đâu

- | Thiết lập, bảo trì và khả năng tìm kiếm mã đơn giản
- | Thao tác đọc/ghi rất nhanh
- | Nếu bạn không cần nhiều chỉ mục phụ
- | Khi các loại cột khác nhau được yêu cầu
- | Trường hợp KHÔNG sử dụng Cassandra
  - ▶ Giao dịch
  - ▶ Nhu cầu cấp phép và bảo mật nghiêm ngặt đối với dữ liệu
  - ▶ Truy vấn động trên cột

# Thuật ngữ Cassandra

- | Trung tâm dữ liệu – Một bài tạo thành một cụm với một tập hợp các nút
- | Nhật ký cam kết – Lưu đĩa cục bộ bằng cách loại bỏ lỗi trong quá trình ghi
- | MemTables – Lưu trữ dữ liệu hoạt động ghi trong bộ đệm bộ nhớ tạm thời, tuôn ra SSTable khi không gian đầy
- | SSTable (Bảng chuỗi đã sắp xếp) – Lưu trữ đĩa dữ liệu tuần tự, bất biến
- | Gossip – Một nút định kỳ trao đổi thông tin trạng thái về các nút khác mà nó được kết nối.
  - ▶ Giao thức ngang hàng, chạy mỗi giây
  - ▶ Trao đổi thông báo trạng thái với tối đa 3 node khác
- | Bộ lọc Bloom – Thuật toán kiểm tra xem một phần tử có phải là thành viên của một tập hợp hay không

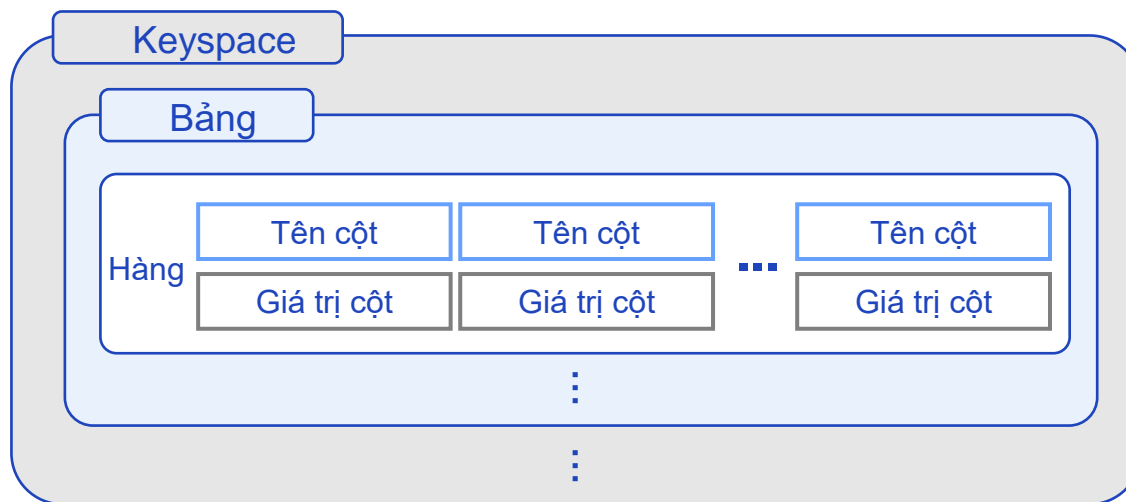
# Cấu trúc dữ liệu Cassandra

## I Mô hình dữ liệu Cassandra

- ▶ Keyspace > Table > Row > tên cột : giá trị cột
- ▶ Keyspace hoạt động như một thùng chứa dữ liệu

## I Cụm Cassandra

- ▶ Lưu trữ dữ liệu phân tán trong mỗi node ở dạng vòng
- ▶ Tiêu chí phân phối là khóa phân vùng
- ▶ Phân phối dựa trên giá trị băm dữ liệu

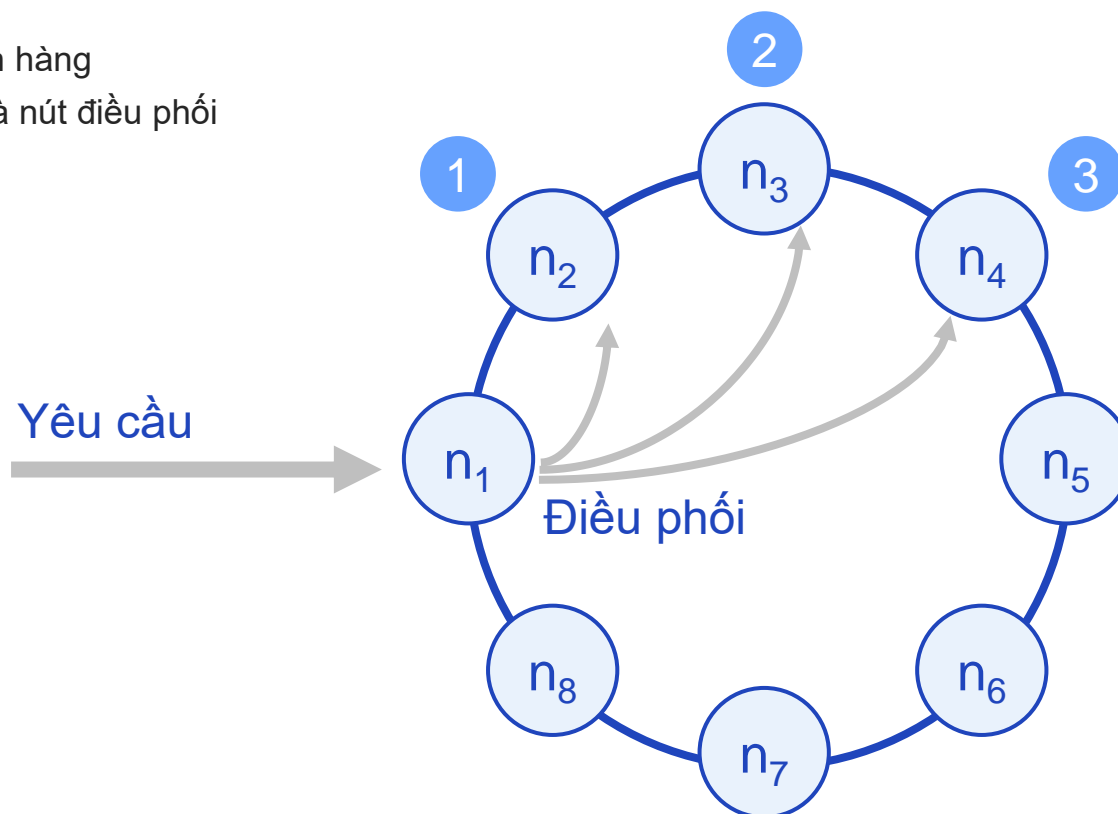


Cấu trúc dữ liệu

## Ghi/Đọc luồng dữ liệu

### I Node điều phối

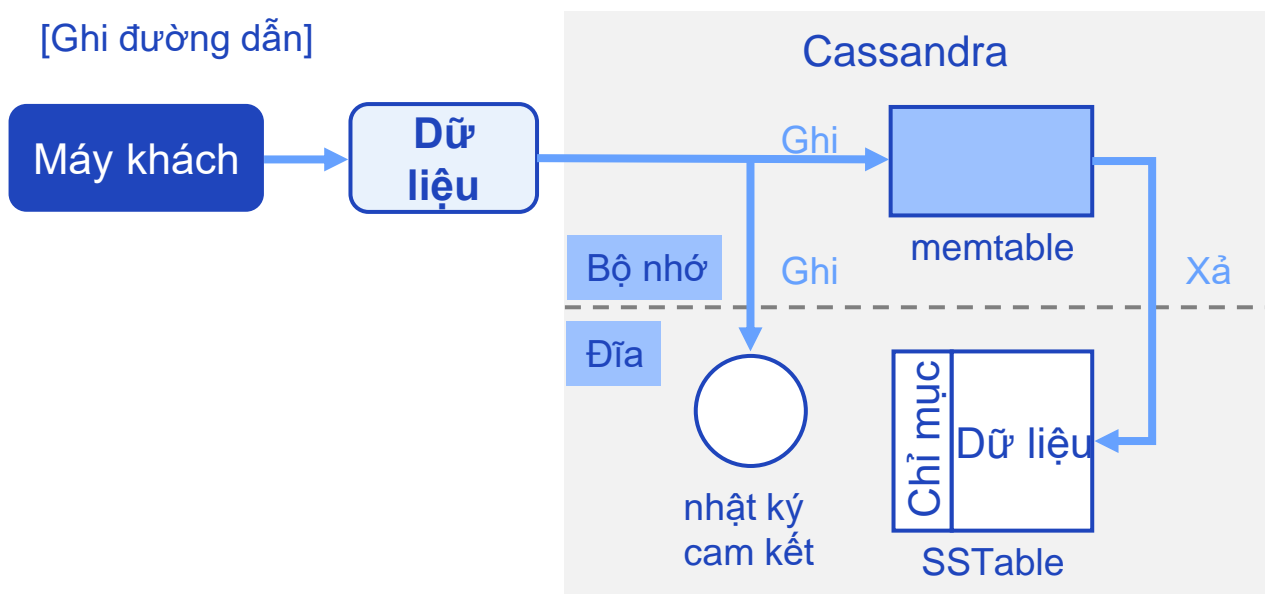
- ▶ Xử lý các yêu cầu của khách hàng
- ▶ Bất kỳ nút nào cũng có thể là nút điều phối





## Ghi đường dẫn (1/2)

- Thêm vào nhật ký Cam kết trên đĩa và lưu vào memTable
- Khi memTable đầy, nó được chuyển sang SSTable thông qua I/O.
- Sau khi xả, dữ liệu trong nhật ký cam kết sẽ bị xóa.



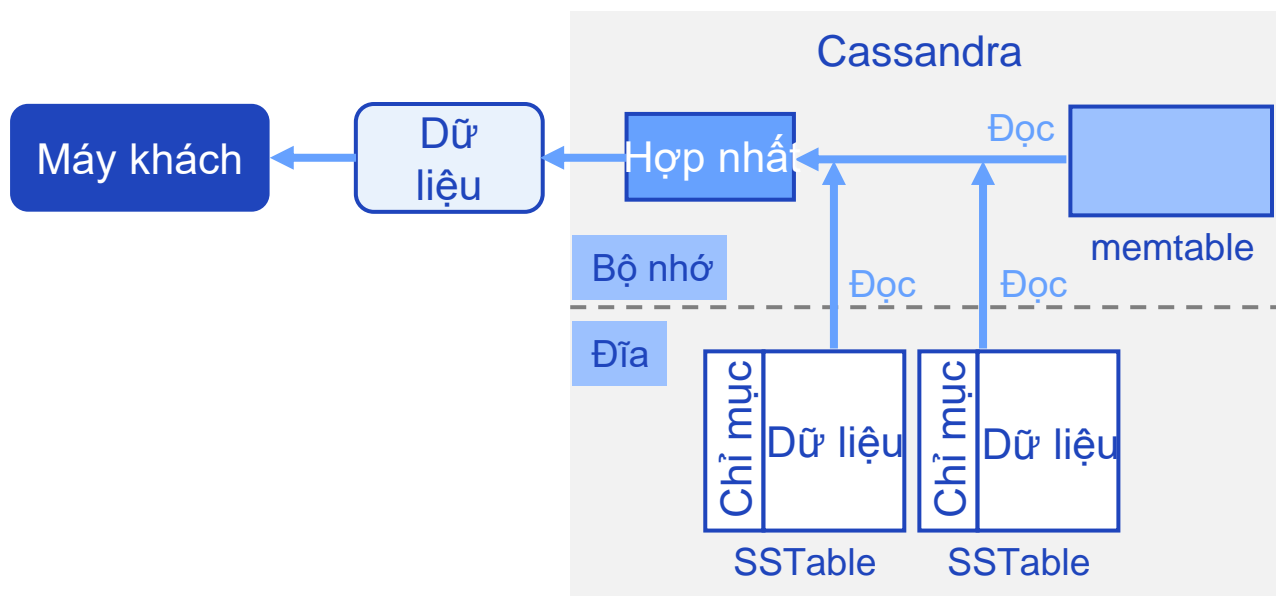
## Ghi đường dẫn (2/2)

- | Các hàng thường được lưu trữ trong các tệp SSTable
- | SSTable bao gồm chỉ mục chính và bộ lọc Bloom
  - ▶ Chỉ mục chính: Danh sách các khóa hàng và vị trí bắt đầu hàng trong tệp dữ liệu
  - ▶ Bộ lọc Bloom: Một tập hợp con của chỉ mục chính để cải thiện hiệu suất
- | Mức độ ghi nhất quán: Số lượng bản sao xác nhận ghi thành công.



## Đọc đường dẫn (1/2)

- | Khi một thao tác đọc được yêu cầu trên một nút, nó sẽ được hợp nhất từ tất cả các SSTable có liên quan và tất cả các memTable chưa được xóa
- | Các giá trị đã hợp nhất được lưu trữ trong bộ nhớ đệm hàng ghi qua.



## Đọc đường dẫn (2/2)

- I Mức độ đọc nhất quán: Số lượng bản sao được truy cập cho các hoạt động đọc nhất quán thành công.



- I Đọc sửa chữa
  - ▶ Điều phối truy cập nhiều nút cho dữ liệu (nhiều như mức nhất quán)
  - ▶ Bản sao nhanh nhất trả về dữ liệu để kiểm tra tính nhất quán thông qua so sánh bộ nhớ.
  - ▶ Điều phối kiểm tra tất cả các bản sao còn lại trong nền
  - ▶ Cập nhật cho các bản sao không nhất quán

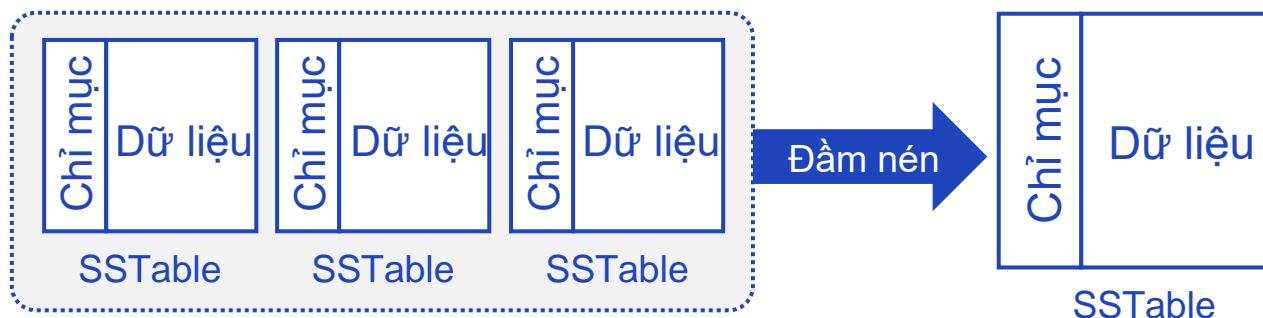
# Xóa dữ liệu

## I Tombstone

- ▶ Điểm đánh dấu của hàng cho biết cột đã xóa
- ▶ Các cột được đánh dấu không bị xóa ngay lập tức mà bị xóa khi SSTable được nén theo thời gian.
- ▶ Tác dụng phụ xảy ra (một nút phục hồi sau khi bị lỗi)

## I Đầm nén

- ▶ Hợp nhất và sắp xếp SSTable để tạo SSTable mới
- ▶ Giảm số lượng tìm kiếm cần thiết để giải phóng dung lượng và cải thiện hiệu suất
- ▶ Hợp nhất khóa, hợp nhất cột, xóa bia mộ, tạo chỉ mục mới



## CQL là gì?

### I CQL(Cassandra Query Language): Ngôn ngữ truy vấn Cassandra

- ▶ CQL là ngôn ngữ cơ bản tận dụng cơ sở dữ liệu Cassandra và tương tự như SQL tiêu chuẩn.
- ▶ Thực hiện các thao tác CRUD bằng trình bao CQL (cqlsh)
- ▶ Các lệnh kết thúc bằng dấu chấm phẩy (;) để phân tách nhiều lệnh.
- ▶ Định danh (tên) được sử dụng để xác định không gian phím, bảng, cột và các đối tượng khác.
- ▶ Tên bắt đầu bằng một bảng chữ cái
- ▶ Từ khóa CHỌN và VỚI không phân biệt chữ hoa chữ thường
- ▶ ( " ) - một mã định danh được trích dẫn được xác định bằng cách đặt các ký tự tùy ý trong dấu ngoặc kép.

## Các loại dữ liệu

- I Cassandra hỗ trợ nhiều loại dữ liệu khác nhau
  - ▶ Loại tích hợp
  - ▶ Loại bộ sưu tập
  - ▶ Loại do người dùng xác định
  - ▶ Loại tuple
  - ▶ Loại tùy chỉnh

## Kiểu dữ liệu tích hợp

Loại dữ liệu	Dữ liệu	
Boolean	Đúng, sai	Sử dụng một trong T/F
Blob	Các đối tượng lớn nhị phân	được viết bằng hệ thập lục phân
ASCII	65(A), 97(a)..	Chuỗi US-ASCII
Bigint	$-2^{32} \sim 2^{32}$	Chỉ số nguyên, 64 bit
Counter	1, 2, 3....	Giá trị bộ đếm tăng/giảm (số nguyên 64 bit)
Timestamp	2020-01-03 01:02:00+0000	Ngày và giờ với mili giây, +0000(GMT)
Double	$-2^{32} \sim 2^{32}$	Tất cả các số thực, 64 bit
Int	$-2^{16} \sim 2^{16}$	số nguyên
varchar	Chuỗi	Chuỗi mã hóa UTF8



## Loại dữ liệu thu thập (1/3)

### I Ánh xạ (Map)

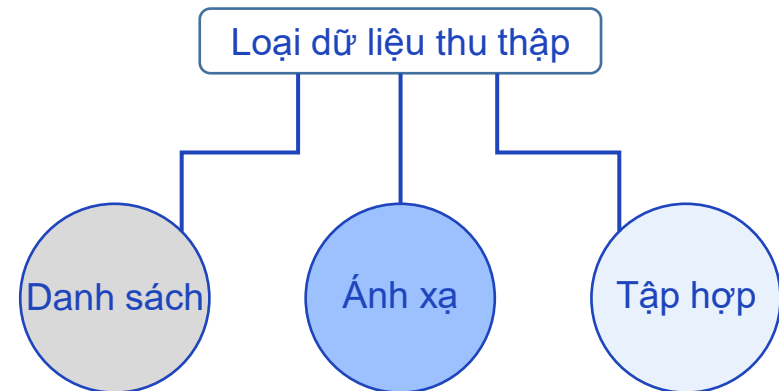
- ▶ Một bộ sưu tập các cặp khóa-giá trị được sắp xếp theo thứ tự
- ▶ Lưu khóa trong Tên và Giá trị trong Giá trị

### I Tập hợp (Set)

- ▶ Tập hợp một hoặc nhiều phần tử được sắp xếp không chồng chéo

### I Danh sách (List)

- ▶ Một tập hợp các phần tử cho phép một hoặc nhiều chồng lên nhau và sắp xếp theo thứ tự



## Loại dữ liệu thu thập (2/3)

### I Ánh xạ

```
Create table ucol1 (  
  id    integer PRIMARY KEY,  
  name  varchar,  
  items  map<text, text>  
);  
  
Insert into ucol1 (id, name, items)  
values(1,'Jason Jeong', {'color' : 'blue', 'movie' : 'star wars'});  
  
Update ucol1 set items = {'color' : 'green'}  
where id = 1;  
Update ucol1 set items['fruit'] = 'apple' where  
id = 1;
```

## Loại dữ liệu thu thập (3/3)

### I Tập hợp

```
Create table img1 (  
  id    integer PRIMARY KEY,  
  title varchar,  
  tags  set<text>  
);
```

```
Insert into img1 (id, title, tags)  
  values(1,'Dog',{'black', 'pet', 'smile'});
```

```
Update img1 set tags = {'white', 'pet', 'cute'}  
  where id = 1;
```

```
Update img1 set tags = tags + {'gray', 'small'}  
  where id = 1;
```

```
Update img1 set tags = tags - {'small'}  
  where id = 1;
```

## Kiểu dữ liệu do người dùng xác định (UDT) (1/2)

### I Kiểu dữ liệu do người dùng xác định



### I Create type statement

```
create_type_statement ::= CREATE TYPE [ IF NOT EXISTS ] udt_name  
'(' field_definition ( ',' field_definition )* ')'  
field_definition ::= identifier cql_type
```

## Kiểu dữ liệu do người dùng xác định (UDT) (2/2)

### I Alter\_type\_statement

```
alter_type_statement ::= ALTER TYPE udt_name alter_type_modification  
alter_type_modification ::= ADD field_definition |  
    RENAME identifier TO identifier ( identifier TO identifier )*
```

### I Drop\_type\_statement

```
drop_type_statement ::= DROP TYPE [ IF EXISTS ] udt_name
```

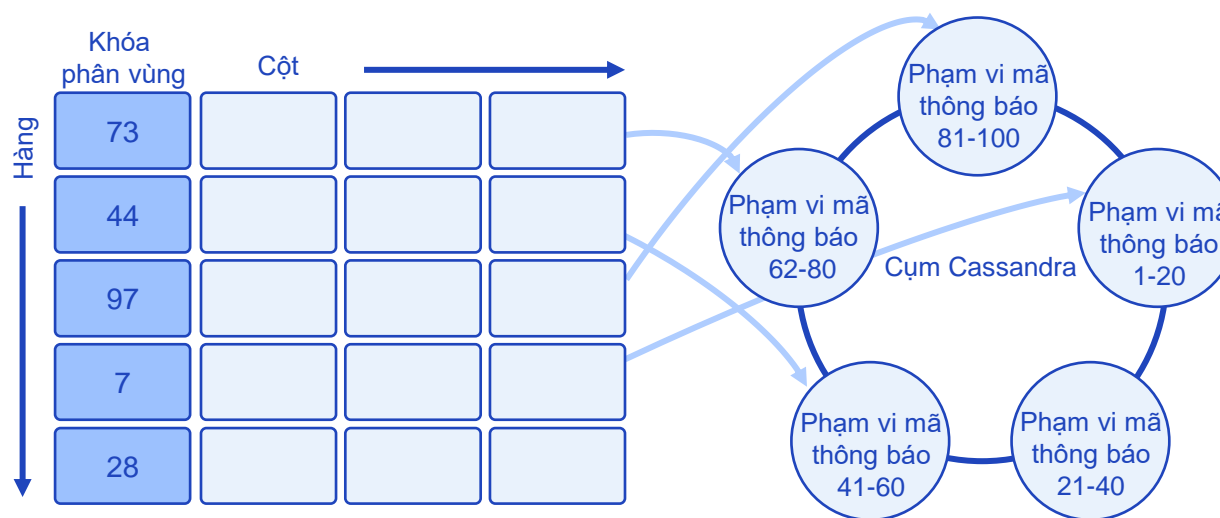
### I Describe\_type\_statement

```
describe_type_statement ::= DESCRIBE TYPE udt_name
```

## Khóa CQL (1/2)

### I Khóa phân vùng

- ▶ Khóa duy nhất để lưu trữ dữ liệu phân tán
- ▶ Khi soạn một bảng, một hoặc nhiều bảng phải được chỉ định và một số bảng có thể được chỉ định.
- ▶ Giá trị trong một cột = khóa hàng
- ▶ Kết hợp các giá trị cột tổng hợp (giá trị:giá trị) với ":" = hàng phím



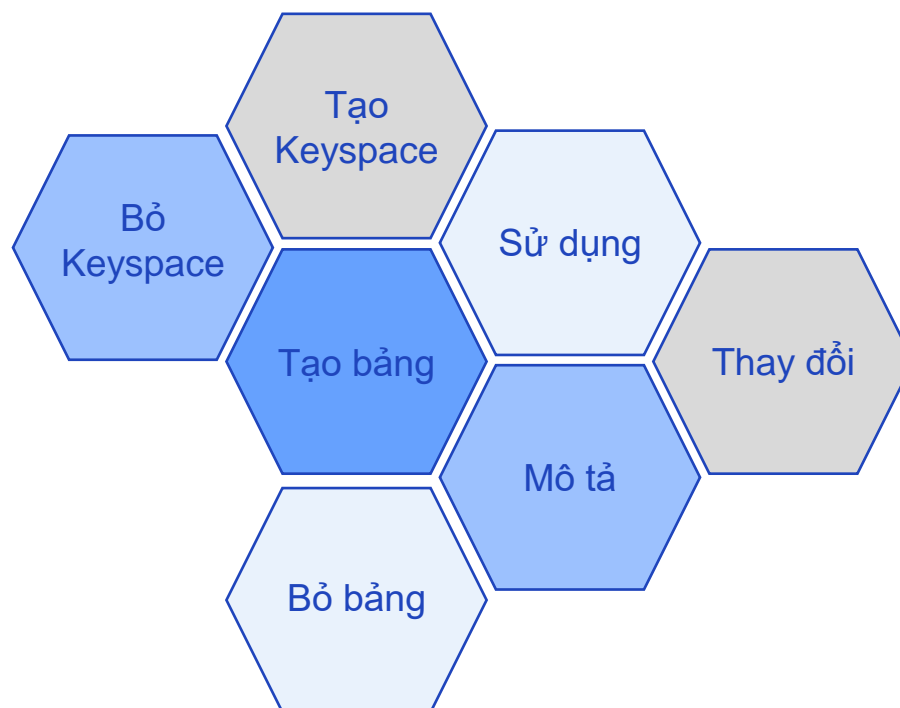
## Khóa CQL (2/2)

- | Khóa cụm
  - ▶ Một khóa để sắp xếp các cột thuộc một hàng
- | Khóa chính
  - ▶ Một cột hoặc nhóm cột xác định duy nhất một hàng trong bảng
  - ▶ Khóa phân vùng + khóa cụm
  - ▶  $PK(a,b,c) \Rightarrow a / bc$
  - ▶  $PK((a,b),c) \Rightarrow a:b / c$
- | Khóa tổng hợp (khóa ghép)
  - ▶ Khóa chính bao gồm hai hoặc nhiều cột CQL
- | Khóa phân vùng tổng hợp
  - ▶ Khóa phân vùng bao gồm hai hoặc nhiều cột CQL

# Lệnh định nghĩa dữ liệu

I Xác định việc tạo, sửa đổi và xóa các không gian phím và bảng

- ▶ Tạo keyspace
- ▶ Sử dụng
- ▶ Thay đổi keyspace
- ▶ Bỏ keyspace
- ▶ Tạo bảng
- ▶ Thay đổi bảng
- ▶ Bỏ bảng
- ▶ Mô tả





## Định nghĩa phổ biến

### I Tên bảng và keyspace

```
keyspace_name ::= name  
table_name ::= [ keyspace_name '.' ] name  
name ::= unquoted_name | quoted_name  
unquoted_name ::= re('[a-zA-Z_0-9]{1, 48}')  
quoted_name ::= "'" unquoted_name "'"
```

### I Tạo keyspace

```
CREATE KEYSAPCE [ IF NOT EXISTS ]  
Keyspace_name WITH options
```

# Tạo Keyspace

## I Keyspace ks\_test

- ▶ Chiến lược đơn giản - sử dụng 2 bản sao cho toàn bộ cụm

```
>Create keyspace 'ks_test'  
With replication = {'class':'SimpleStrategy',  
'replication_factor' : 2};
```

## I Keyspace ks\_test2

- ▶ Xác định số lượng bản sao cho mỗi trung tâm dữ liệu
- ▶ NetworkTopologyStrategy - 1 cho DC1, 2 cho DC2

```
>CREATE KEYSPACE ks_test2  
WITH replication =  
{'class':'NetworkTopologyStrategy', 'DC1':  
'1', 'DC2': '2'}
```

# Tạo bảng

## I Tạo bảng mới

```

create_table_statement ::= CREATE TABLE [ IF NOT EXISTS ] table_name '('
    column_definition
    ( ',' column_definition )*
    [ ',' PRIMARY KEY '(' primary_key ')' ]
    ')' [ WITH table_options ]
column_definition ::= column_name cql_type
[ STATIC ] [ PRIMARY KEY]
primary_key ::= partition_key
    [ ',' clustering_columns ]
partition_key ::= column_name
| '(' column_name ( ',' column_name )* ')'
clustering_columns ::= column_name ( ',' column_name )*
table_options ::= COMPACT STORAGE [ AND table_options ]
    | CLUSTERING ORDER BY '(' clustering_order ')'
    [ AND table_options ] | options
clustering_order ::= column_name (ASC | DESC)
    ( ',' column_name (ASC | DESC) )*
  
```

# Lệnh thao tác dữ liệu

I Các lệnh truy xuất, xóa, thêm, cập nhật và xử lý dữ liệu hàng loạt



# Môi trường Cassandra trong bài tập

- I Cài đặt và vận hành Cassandra
- I Môi trường phòng thí nghiệm
  - ▶ <http://cassandra.apache.org/doad>
  - ▶ Phiên bản mới nhất: 3.11
  - ▶ Linux - Centos 6.7, Ubuntu..
  - ▶ Trên trang tải xuống, hãy chọn hệ điều hành của bạn và bản phân phối phù hợp
  - ▶ Yêu cầu cài đặt Java
  - ▶ Python 2.7 để sử dụng Cqlsh

Bài 2.

# NoSQL

| 2.1. Tổng quan về NoSQL

| 2.2. Apache HBase

| 2.3. Cassandra

| 2.4. MongoDB

# MongoDB là gì?

- | MongoDB là một chương trình cơ sở dữ liệu định hướng tài liệu đa nền tảng. Được phân loại là chương trình cơ sở dữ liệu NoSQL, MongoDB sử dụng các tài liệu giống JSON với lược đồ. MongoDB được phát triển bởi MongoDB Inc.
- | Cơ sở dữ liệu NoSQL hướng tài liệu, viết bằng C++
- | 2007.10 Bắt đầu phát triển, chuyển sang mô hình phát triển mã nguồn mở vào năm 2009
- | Được phát triển như một thành phần của sản phẩm PaaS vào thế hệ thứ 10. (MongoDB Inc)
- | mongoDB (từ DB “khổng lồ”) là một cơ sở dữ liệu hiệu suất cao, có thể mở rộng.



# Tổng quan về MongoDB

- | Cơ sở dữ liệu tài liệu với cấu trúc lưu trữ dữ liệu kiểu JSON
  - ▶ Lưu trữ dữ liệu ở định dạng giống như được sử dụng trong lập trình
  - ▶ Định dạng BSON (JSON nhị phân)
- | Khả năng mở rộng dễ dàng thông qua Sharding
- | Tính khả dụng cao khi sử dụng Bộ bản sao
- | Cung cấp API bằng nhiều ngôn ngữ khác nhau
  - ▶ C, C#, C++, Java, JavaScript, Perl, PHP, Python, Ruby, Scala
- | Schema-less
  - ▶ Mỗi tài liệu được lưu trữ trong cơ sở dữ liệu chứa các trường khác nhau.
  - ▶ Lý tưởng cho dữ liệu phi cấu trúc
  - ▶ lược đồ được nhập động vì nó không được xác định trước



# Sử dụng MongoDB ở đâu

- | RDBMS thay thế cho các ứng dụng web
- | Hệ thống quản lý nội dung bán cấu trúc
- | Phân tích thời gian thực & ghi nhật ký tốc độ cao (Logging)
- | Big Data & Trung tâm dữ liệu
- | Quản lý dữ liệu người dùng
- | Ai đang sử dụng MongoDB?



# Thuật ngữ chung MongoDB (1/2)

### | \_id - Khóa chính của tài liệu

- ▶ Các trường bắt buộc cho mọi tài liệu
- ▶ Giá trị duy nhất của Tài liệu
- ▶ \_id bao gồm thời gian/ID máy/ID quá trình/số thứ tự để đảm bảo tính duy nhất
- ▶ Tự tạo nếu thiếu

### | Sưu tầm – tài liệu nhóm

- ▶ Khái niệm bảng trong cơ sở dữ liệu quan hệ
- ▶ Tồn tại trong một cơ sở dữ liệu
- ▶ Bộ sưu tập không bị giới hạn về cấu trúc

### | Con trỏ - Con trỏ tới tập kết quả của truy vấn

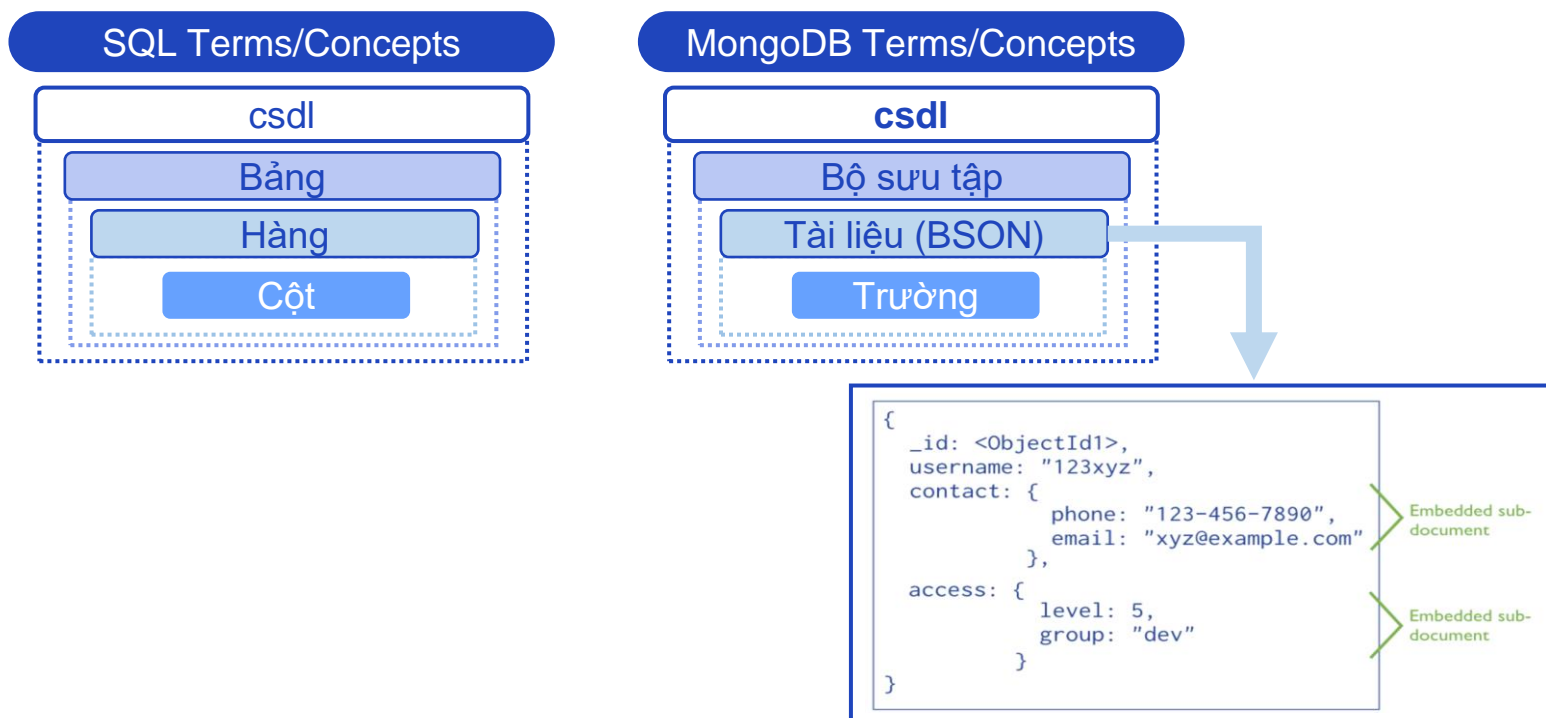
- ▶ Máy khách có thể lặp qua con trỏ để lấy kết quả

## Thuật ngữ chung MongoDB (2/2)

- I Cơ sở dữ liệu - Vùng chứa cho các bộ sưu tập
  - ▶ Mỗi CSDL có bộ tập riêng trong hệ thống tệp
  - ▶ Máy chủ MongoDB lưu trữ nhiều cơ sở dữ liệu
- I Tài liệu – Khái niệm hàng trong cơ sở dữ liệu quan hệ
  - ▶ Các bản ghi trong Bộ sưu tập MongoDB
  - ▶ Tài liệu bao gồm tên trường và giá trị (khóa-giá trị)
  - ▶ Giá trị có thể là tài liệu, mảng hoặc mảng tài liệu khác nhau
- I Trường - Một cặp tên-giá trị trong tài liệu
  - ▶ Một hoặc nhiều trường tồn tại trong tài liệu
  - ▶ Các trường tương tự như các cột trong cơ sở dữ liệu quan hệ.
- I JSON - Ký hiệu đối tượng JavaScript
  - ▶ Định dạng văn bản có thể đọc được để biểu thị dữ liệu có cấu trúc
  - ▶ Hỗ trợ nhiều ngôn ngữ lập trình

# Mô hình dữ liệu trong MongoDB

I [Mongodb : db > collection > document(record) > key:value]



## Các tính năng của MongoDB (1/3)

- I Truy vấn:
  - ▶ Hỗ trợ truy vấn đặc biệt và truy vấn dựa trên tài liệu
- I Hỗ trợ chỉ mục:
  - ▶ Tất cả các trường trong tài liệu có thể được lập chỉ mục
- I Nhân rộng:
  - ▶ Sao chép master-slave
- I Duy trì nhiều bản sao dữ liệu
  - ▶ Ngăn chặn treo cơ sở dữ liệu



## Các tính năng của MongoDB (2/3)

- I Nhiều máy chủ:
  - ▶ Cơ sở dữ liệu chạy trên nhiều máy chủ và sao chép dữ liệu để bảo vệ hệ thống trong trường hợp lỗi phần cứng.
- I Tự động phân đoạn:
  - ▶ Lưu trữ phân tán dữ liệu trong phân đoạn (phân vùng vật lý) và cân bằng tải tự động với phân đoạn
- I MapReduce:
  - ▶ Hỗ trợ phân tán/song song thông qua sự kết hợp của chức năng ánh xạ và rút gọn



## Các tính năng của MongoDB (3/3)

### I Xử lý lỗi:

- ▶ Một số lượng lớn các bản sao đảm bảo tính khả dụng của dữ liệu trong trường hợp xảy ra lỗi hệ thống, lỗi trung tâm dữ liệu hoặc lỗi phân đoạn mạng

### I GridFS:

- ▶ Chia tệp thành các phần nhỏ hơn và lưu chúng thành các tài liệu riêng biệt

### I Schema-less:

- ▶ Lược đồ không được xác định trước và tự động lưu trữ nhiều dữ liệu khác nhau khi cần.

### I Lưu trữ hướng tài liệu:

- ▶ Cơ sở dữ liệu kiểu tài liệu sử dụng định dạng BSON, lưu trữ một tài liệu trên mỗi hàng



# Tại sao lại là MongoDB

## I Ưu & Nhược điểm





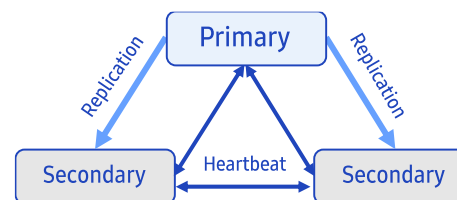
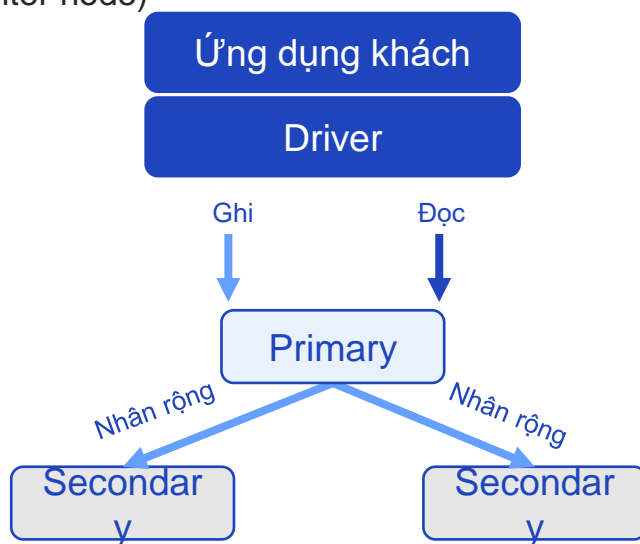
# Truy vấn Ad-Hoc

- I Hỗ trợ các loại truy vấn khác nhau như Truy vấn SQL của RDBMS
  - ▶ Trường
  - ▶ Truy vấn phạm vi
  - ▶ Tìm kiếm biểu thức chính quy
  - ▶ Kết hợp chuẩn xác
  - ▶ Hàm do người dùng định nghĩa được tạo bằng Java Script
  - ▶ Điều kiện tìm kiếm có thể được chỉ định linh hoạt
  - ▶ Từ việc tìm kiếm thông qua biểu thức chính quy, có thể kiểm tra xem một giá trị cụ thể có được bao gồm trong dữ liệu mảng hay không.

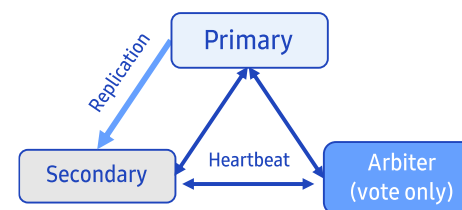
## Bản sao (Bộ bản sao)

Quản lý phân tán cùng một dữ liệu cho nhiều máy chủ để có tính sẵn sàng cao trong trường hợp máy chủ và lỗi mạng

- ▶ Primary node (Leader)
- ▶ Secondary node (follow)
- ▶ Node phân xử (Arbiter node)



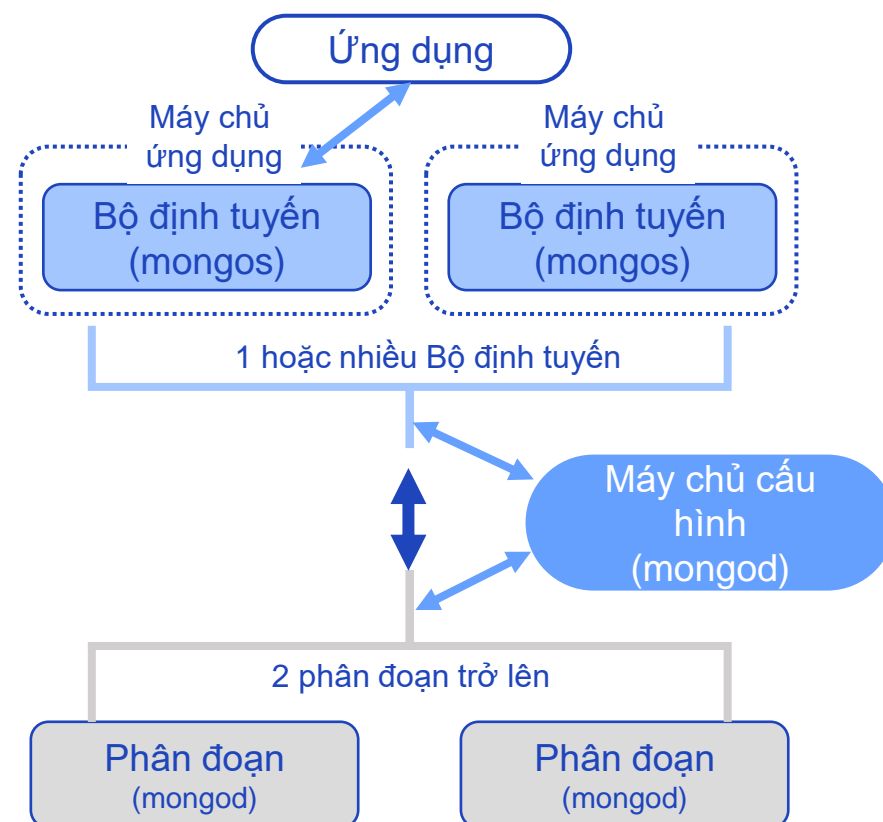
Trường hợp 1



Trường hợp 2

## Phân đoạn (sharding)

- I Phân đoạn phân phối, lưu trữ và xử lý dữ liệu trên nhiều máy chủ.
- I Cụm phân đoạn
  - ▶ Máy chủ cấu hình – lưu trữ thông tin meta như phạm vi dữ liệu và thông tin vị trí phân đoạn
  - ▶ Máy chủ bộ định tuyến (mongos) – hoạt động như một proxy để thực hiện các truy vấn
  - ▶ Phân đoạn – Tập hợp con dữ liệu được phân vùng, có thể được triển khai dưới dạng bộ bản sao



# Schema-less

- I Không có lược đồ được xác định trước
  - ▶ Ứng dụng xác định cấu trúc dữ liệu
  - ▶ Thiết kế dữ liệu có thể thay đổi thường xuyên
  - ▶ Lưu trữ dữ liệu bất cứ lúc nào bạn cần mà không cần xác định trước cột sẽ sử dụng
  - ▶ Tất cả dữ liệu có thể được lưu trữ trong một bộ sưu tập mà không cần chuẩn hóa như bảng RDB
  - ▶ Nhiều loại dữ liệu có thể được lưu trữ
  - ▶ Dễ dàng thay đổi mô hình dữ liệu, mở rộng trường

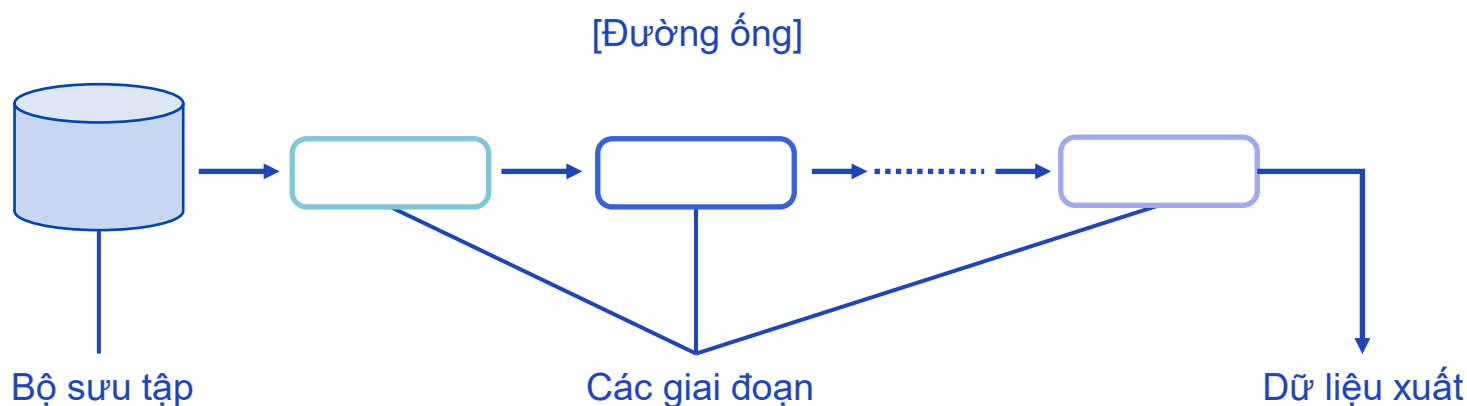
# Khung tổng hợp

- | Hoạt động tổng hợp xử lý bản ghi dữ liệu và chuyển kết quả tính toán
  - ▶ Quy trình theo thứ tự đường ống
- | Nhóm các giá trị từ nhiều tài liệu lại với nhau và thực hiện các thao tác khác nhau trên dữ liệu được nhóm để mang lại một kết quả duy nhất
- | Phương pháp thực hiện
  - ▶ Đường ống tổng hợp
  - ▶ Khung tổng hợp được mô hình hóa theo khái niệm đường ống xử lý dữ liệu
  - ▶ Chức năng Map-Reduce
  - ▶ Hoạt động tổng hợp mục đích duy nhất
  - ▶ Các hoạt động kết hợp các tài liệu trong một bộ sưu tập duy nhất
  - ▶ `db.collection.count()`, `db.collection.distinct()`

## Đường ống tổng hợp

### I Đường ống

- ▶ Truyền phát tài liệu và biến nó thành nhiều trạng thái
- ▶ Mỗi giai đoạn chuyển đổi tài liệu thông qua đường ống dẫn.
- ▶ Một số giai đoạn tạo tài liệu mới hoặc lọc tài liệu.
- ▶ 7 trạng thái - \$project, \$match, \$group, \$sort, \$skip & \$ limit, \$first & \$last, \$unwind

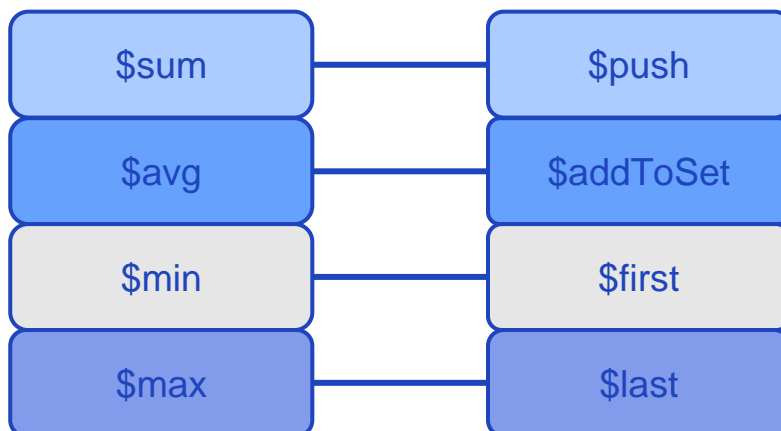


## Biểu thức tổng hợp

### I Sự biểu lộ

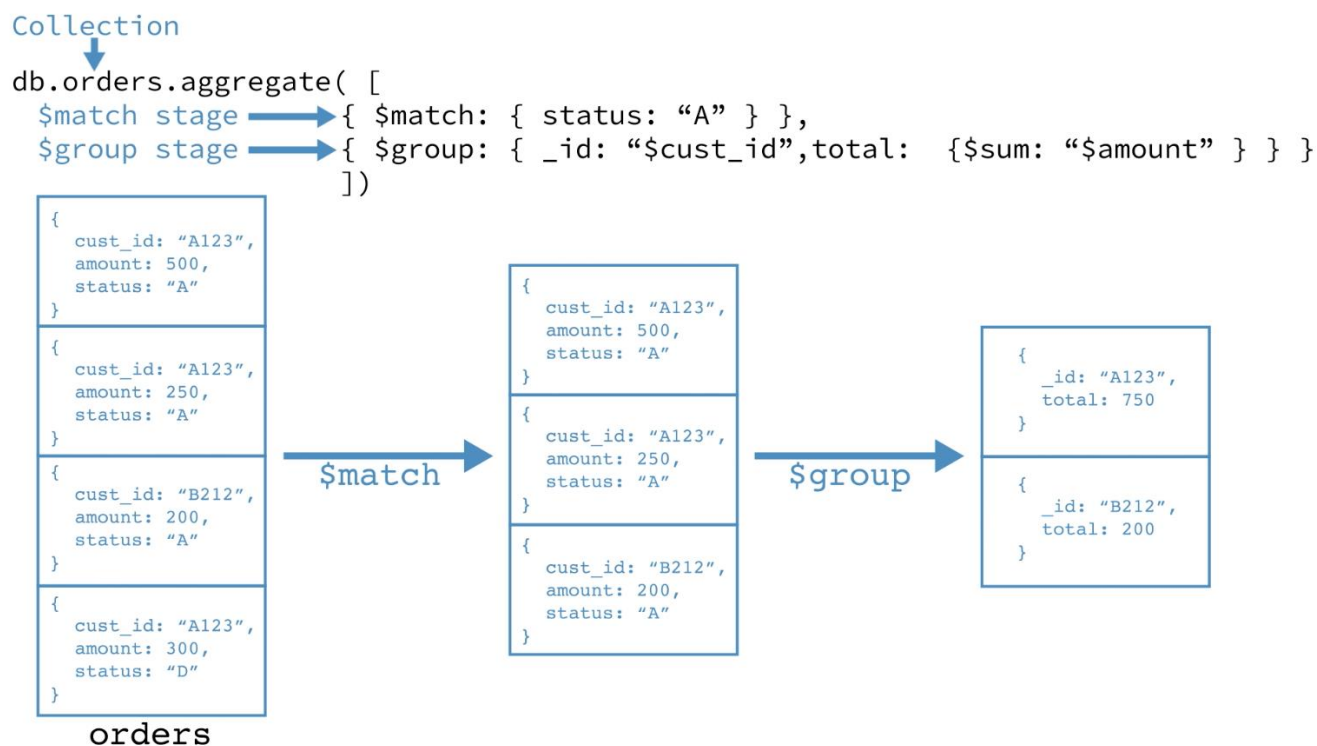
- ▶ Tạo tài liệu đầu ra dựa trên biểu thức tài liệu đầu vào

[Biểu thức tổng hợp]



# Ví dụ tập hợp

## I Tổng hợp - Kết hợp Giai đoạn (đường ống) và biểu thức





# Lập chỉ mục

- | Một cấu trúc dữ liệu để truy xuất nhanh các bản ghi từ một bảng nhất định mà không cần duyệt qua tất cả các bản ghi trong bảng.
  - ▶ Hỗ trợ tìm kiếm nhanh bằng cách sắp xếp trước thứ tự dữ liệu
  - ▶ Giới hạn số lượng tài liệu cần được quét trong một bộ sưu tập (so với quét)
  - ▶ Cấu trúc cây nhị phân
- | Chỉ có một chỉ mục hợp lệ cho mỗi truy vấn
- | Nếu bạn cần hai chỉ mục, hãy sử dụng chỉ mục tổng hợp.
  - ▶ Trong một chỉ mục tổng hợp, thứ tự của các khóa rất quan trọng.
- | Chỉ mục mặc định
  - ▶ `_id` tạo một chỉ mục duy nhất trên trường `_id` trong khi tạo bộ sưu tập
  - ▶ Chỉ mục `_id` ngăn máy khách chen hai tài liệu có cùng giá trị `_id`.

# Các loại chỉ mục trong MongoDB (1/2)

### I Chỉ mục trường đơn

- ▶ Do người dùng chỉ định riêng ngoài chỉ mục `_id`
- ▶ Cung cấp tính năng tạo chỉ mục tăng dần/giảm dần tùy chỉnh cho một trường trong một tài liệu

### I Chỉ số hợp chất

- ▶ Chỉ mục tổng hợp hỗ trợ chỉ mục do người dùng xác định trên nhiều trường
- ▶ Thứ tự của từng trường rất quan trọng - trước tiên hãy sắp xếp trường đầu tiên, sau đó là trường thứ hai

### I Chỉ mục đa khóa

- ▶ Lập chỉ mục với Mảng và tài liệu được nhúng dưới dạng trường

### I Chỉ số không gian địa lý

- ▶ Cung cấp một chỉ mục đặc biệt để cung cấp khả năng truy vấn dữ liệu tọa độ địa lý hiệu quả

# Các loại chỉ mục trong MongoDB (2/2)

## I Chỉ mục văn bản

- ▶ Chỉ mục hỗ trợ tìm kiếm chuỗi trong bộ sưu tập
- ▶ Chỉ mục văn bản chứa các trường có giá trị là chuỗi hoặc mảng các phần tử chuỗi

## I Chỉ mục băm

- ▶ Sử dụng cấu trúc dữ liệu băm thay vì cây B
- ▶ Nếu dữ liệu chính được phân phối không đồng đều, các giá trị chỉ mục băm vẫn nhất quán.
- ▶ Chỉ mục băm nhiều khóa không được phép, truy vấn theo phạm vi không được phép

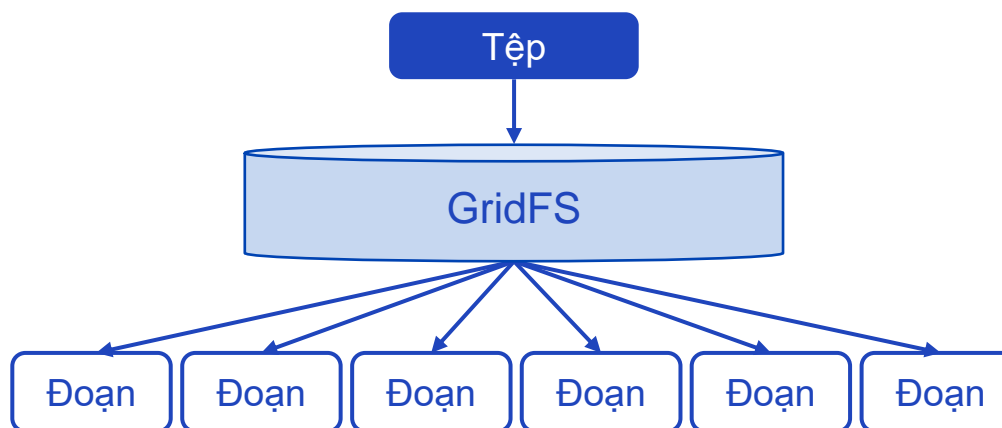
## Bộ sưu tập giới hạn

- I Chỉ lưu trữ dữ liệu trong không gian được tạo ban đầu với kích thước giới hạn
  - ▶ Bộ sưu tập kích thước cố định
  - ▶ Một bộ sưu tập được sử dụng lại khi không gian đầu tiên được sử dụng hết
  - ▶ Phù hợp với dữ liệu chỉ được lưu trữ và quản lý trong một khoảng thời gian nhất định, chẳng hạn như dữ liệu nhật ký.
  - ▶ Tài liệu được lưu theo thứ tự lưu và khi đầy, tài liệu cũ nhất sẽ bị xóa và tài liệu mới được lưu trữ (Phương thức xếp hàng)
  - ▶ Tùy chọn kích thước yêu cầu
- I Không thể xóa hoặc cập nhật tài liệu
- I Tốc độ chèn tài liệu nhanh nhưng thao tác tìm phụ thuộc vào thứ tự chèn

```
db.createCollection("capped_collection",  
    {capped: true, size: 100000})
```

## Hệ thống tệp lưới (1/2)

- I Quản lý các tệp vượt quá giới hạn kích thước tài liệu BSON là 16 MB
  - ▶ Thích hợp cho các tệp nhị phân/hình ảnh lớn, khi có giới hạn về số lượng tệp được lưu trữ trong hệ thống tệp
  - ▶ Thay vì lưu trữ tệp trong một tài liệu, hãy chia tệp thành nhiều phần nhỏ hơn và lưu trữ từng phần dưới dạng một tài liệu riêng biệt.
  - ▶ Không phù hợp khi cập nhật nguyên bản nội dung của toàn bộ tệp



## Hệ thống tệp lưới (2/2)

- I Theo mặc định, GridFS lưu trữ các tệp trong hai bộ sưu tập bằng cách sử dụng tiền tố 'fs'.
  - ▶ fs.files – thông tin meta tệp
  - ▶ fs.chunks – khối nhị phân
- I Chỉ mục GridFS
  - ▶ Chỉ mục khối - sử dụng các chỉ mục duy nhất và phức hợp cho các trường files\_id và n
  - ▶ Chỉ mục tệp – sử dụng các trường tên tệp và ngày tải lên để lập chỉ mục cho bộ sưu tập tệp

# [Lab3]

## Truy cập dữ liệu với Hbase



# [Lab4]

## Truy cập dữ liệu bằng lệnh DML





# [Lab5]

## Làm việc với Hbase



# [Lab6]

## Truy cập dữ liệu với Cassandra



# [Lab7]

## Làm việc với Cassandra





**SAMSUNG**

Together for Tomorrow!  
**Enabling People**

Education for Future Generations

©2021 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this book other than the curriculum of Samsung innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.