

****

**Báo Cáo CUỐI KÌ**

**MôN: Big Data Essentials**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Lập trình trên windows**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**TIỂU LUẬN**

**Trí Tuệ Nhân Tạo**

**TIỂU LUẬN**

**TƯ TƯỞNG HỒ CHÍ MINH**

**Giảng viên: Ths. Lê Thị Minh Châu**

**Sinh viên thực hiện**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Phạm Minh Long | 20133062 |
| 2 | Nguyễn Quốc Thắng | 20133091 |
| 3 | Trương Hoàng Anh Khôi | 20133058 |
| 4 | Nguyễn Văn Trường Tốt | 20133098 |

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO**n thực hiện**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Trương Hoàng Anh Khôi | 20133058 |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO**n thực hiện**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Trương Hoàng Anh Khôi | 20133058 |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

Thủ Đức, tháng 12 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 06 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

Thủ Đức, tháng 05 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠOThủ Đức, tháng 05 năm 2022

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA: CHÍNH TRỊ VÀ LUẬT**

**LỜI CẢM ƠN**

Lời đầu tiên, nhóm chúng em xin chân thành gửi lời cảm ơn tới trường ĐH Sư Phạm kỹ Thuật TP. HCM đã tạo mọi điều kiện cho chúng em có môi trường học tập thật tốt. Xin cảm ơn tất cả quý thầy cô đã hết lòng quan tâm, dạy dỗ và trang bị cho chúng em những kiến thức quý báu. Chúng em xin chân thành cảm ơn cô Lê Thị Minh Châu, người đã trực tiếp hướng dẫn nhóm chúng em trong quá trình triển khai và thực hiện đề tài này. Cảm ơn cô đã tận tình hướng dẫn, động viên giúp đỡ đưa ra hướng đi tối ưu nhất cho đề tài của nhóm. Cuối cùng, chân thánh cảm ơn tất cả bạn bè đã động viên, khích lệ cũng như đóng góp những ý kiến quý báu để nhóm chúng em hoàn thành tốt đề tài của mình.

Do trình độ chúng em còn hạn hẹp cũng như kinh nghiệm còn thiếu sót chúng em rất mong nhận được ý kiến đóng góp của cô cho đề tài để chúng em có thể cải thiện và học hỏi được nhiều kiến thức bổ ích. Chúng em xin cảm ơn.

**MỤC LỤC**

[**A. Tổng quan về NoSQL 5**](#_Toc122550701)

[**1. Khái niệm 5**](#_Toc122550702)

[**2. Lịch sử ra đời NoSQL 5**](#_Toc122550703)

[**3. Phân loại NoSQL 6**](#_Toc122550704)

[**3.1. Key-value stores 6**](#_Toc122550705)

[**3.2. Column-oriented databases 6**](#_Toc122550706)

[**3.3. Graph databases 7**](#_Toc122550707)

[**3.4. Document Oriented databases 7**](#_Toc122550708)

[**4. So sánh SQL và NoSQL 8**](#_Toc122550709)

[**5. Ưu điểm và nhược điểm của NoSQL 9**](#_Toc122550710)

[**5.1. Ưu điểm của NoSQL 9**](#_Toc122550711)

[**5.2. Nhược điểm của NoSQL 10**](#_Toc122550712)

[**6. Cách hoạt động của NoSQL 10**](#_Toc122550713)

[**7. Ứng dụng NoSQL 11**](#_Toc122550714)

[**B. Tìm hiểu Redis 13**](#_Toc122550715)

[**1. Khái niệm 13**](#_Toc122550716)

[**2. Ưu điểm và nhược điểm của Redis 13**](#_Toc122550717)

[**2.1. Ưu điểm của Redis 13**](#_Toc122550718)

[**2.2. Nhược điểm của Redis 14**](#_Toc122550719)

[**3. Các kiểu dữ liệu 14**](#_Toc122550720)

[**3.1 Key 14**](#_Toc122550721)

[**3.2 String 15**](#_Toc122550722)

[**3.3. List 15**](#_Toc122550723)

[**3.4. Set 16**](#_Toc122550724)

[**3.5. Hash 17**](#_Toc122550725)

[**3.6. Sorted set 17**](#_Toc122550726)

[**4. Redis Persistence là gì 18**](#_Toc122550727)

[**5. Ứng dụng Redis 18**](#_Toc122550728)

[**Tài liệu tham khảo 20**](#_Toc122550729)

2. **Tổng quan về NoSQL**
3. **Khái niệm**

No SQL có nghĩa là Non-Relational (NoRel) – Không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL – Không chỉ là SQL

Đây là thuật ngữ chung cho các hệ CSDL không sử dụng mô hình dữ liệu quan hệ. No SQl đặc biệt nhấn mạnh đến mô hình lưu trữ cặp giá trị - khoá và hệ thống lưu trữ phân tán.

NoSQL không phải là một sản phẩm hay một công nghệ duy nhất, nó đại diện cho một lớp các sản phẩm liên quan đến việc lưu trữ và xử lý dữ liệu không theo quy tắt RDBMS

1. **Lịch sử ra đời NoSQL**

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các lightweight open source relational database ( cơ sở dữ liệu quan hệ nguồn mở nhỏ) nhưng không sử dụng SQL để truy vấn.

Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu về cơ sở dữ lệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: phân tán(distributed) + không ràng buộc (non-relational).

NoSQL storage đặc biệt phổ dụng trong thời kỳ web 2.0 bùng nổ, nơi các mạng dịch vụ dữ liệu cộng đồng cho phép người dùng tạo hàng tỷ nội dung trên web. Dẫn đến việc gia tăng nhanh về kích thước dữ liệu và ngày càng yêu cầu cao về khà năng truy xuất dữ liệu. Dữ liệu lớn rất nhanh đã vượt qua giới hạn phần cứng cần phải giải quyết bằng bài toán phân tán trên nhiều máy chủ khác nhau vì hạn chế của thiết bị lưu trữ. Các RDBMS vẫn xử lý tốt nhiệm vụ lưu trữ dữ liệu tập trung, nhưng cũng chính vì tính tập trung đó mà lại tạo ra khó khăn cho hệ thống nếu muốn mở rộng. Việc người sử dụng thường xuyên tìm cách phi chuẩn hoá dữ liệu đã phá vỡ thiết kế cơ sở dữ liệu ban đầu. Hơn nữa các giao dịch phân tán trên RDBMS thường sinh ra hiện tượng thắt nút cổ chai, việc xử lý chúng gặp tương đối nhiều khó khăn khi các RDBMS không có sự hỗ trợ trực tiếp cho những tình huống như vậy. Đa số các hệ cơ sở dữ liệu NoSQL được hình thành trong thế giới của việc xử lý song song và tránh được các hiện tượng này. Do đó NoSQL nổi lên như một giải pháp sáng giá và đáp ứng được tình huống này.

1. **Phân loại NoSQL**

Cơ sở dữ liệu NoSQL có nhiều loại, mỗi loại có các đặc điểm, cách lưu trữ khác nhau được phân loại thành bốn loại: Key-values stores, Column-oriented, Graph-based và Document-oriented. Mỗi loại đều có những thuộc tính và hạn chế riêng. Không có cơ sở dữ liệu nào được cho là tốt hơn để giải quyết tất cả các vấn đề.

* 1. ***Key-value stores***

Dữ liệu được lưu trữ trong database dưới dạng key-value. Để truy vấn dữ liệu trong database, ta dựa vào key để lấy value ra. Các database dạng này có tốc độ truy vấn rất nhanh.

Database tiêu biểu: Hbase, Redis, MemCache, Project Voldemort, CouchBase

Ứng dụng: Do tốc độ truy xuất nhanh, key-value database thường được dùng để làm cache cho ứng dụng (Tiêu biểu là Redis và MemCache).

* 1. ***Column-oriented databases***

Dữ liệu được lưu trữ trong database dưới dạng key-value. Để truy vấn dữ liệu trong database, ta dựa vào key để lấy value ra. Các database dạng này có tốc độ truy vấn rất nhanh.

Database tiêu biểu: Hbase, Redis, MemCache, Project Voldemort, CouchBase

Ứng dụng: Do tốc độ truy xuất nhanh, key-value database thường được dùng để làm cache cho ứng dụng (Tiêu biểu là Redis và MemCache).

* 1. ***Graph databases***

Dữ liệu trong graph database được lưu dưới dạng các node. Mỗi node sẽ có 1 label, 1 số properties như một row trong SQL. Các node này được kết nối với nhau bằng các relationship. Graph database tập trung nhiều vào relationship giữa các node, áp dụng nhiều thuật toán duyệt node để tăng tốc độ.

Database tiêu biểu: Infinite Graph, Orient DB, FlockDB, Neo4J, Giraph.

Ứng dụng: Khi cần truy vấn các mối quan hệ, graph database truy vấn nhanh và dễ hơn nhiều so với database. Nó được dùng trong các hệ thống: mạng nơ ron, chuyển tiền bạc, mạng xã hội (tìm bạn bè), giới thiệu sản phẩm (dựa theo sở thích/lịch sử mua sắm của người dùng)

* 1. ***Document Oriented databases***

Mỗi object sẽ được lưu trữ trong database dưới dạng một document. Dữ liệu sẽ được lưu trữ dưới dạng BSON/JSON/XML dưới database. Dữ liệu không schema cứng như SQL, do đó ta có thể thêm/sửa field, thay đổi table, … rất nhanh và đơn giản. Database dạng này có tốc độ truy vấn nhanh, có thể thực hiện các câu truy vấn phức tạp, dễ mở rộng (scalability). Mỗi database có một kiểu truy vấn riêng, khá là loạn xà ngầu (RavenDB dùng Lucene, MongoDB lại dùng query document).

Database tiêu biểu: MongoDB, RavenDB, CouchDB, TerraStone, OrientDB.

Ứng dụng: Do nhanh và linh động, document database thường đóng vài trò làm database cho các ứng dụng prototype, big data, e-commerce, CMS. Ngoài ra, ta còn dùng nó để lưu log hoặc history

1. **So sánh SQL và NoSQL**

|  |  |  |
| --- | --- | --- |
| **Tính năng** | **SQL** | **NoSQL** |
| Hiệu xuất | Kém do khi truy xuất dữ liệu cần phải tính toán cũng như xử lý các mối quan hệ giữa các bảng | Tốt hơn SQL vì bỏ qua các ràng buộc |
| Khả năng mở rộng | Mở rộng theo chiều dọc, cần năng các hiệu năng của máy chủ | Mở rộng theo chiều ngang, chỉ cần thêm nhiều máy chủ hơn |
| Phần cứng | Đòi hỏi phần cứng cao | Không đòi hỏi quá cao về phần cứng |
| Thay đổi số node trong hệ thống | Vì tính nhất quán về dữ liệu nên khi thêm hay xóa 1 node cần phải shutdown hệ thống trong 1 khoảng thời gian | Vì tính nhất quán cuối nên sẽ không cần phải shutdown hệ thống |
| Truy vấn và báo cáo | Dễ dàng sử dụng ngôn ngữ SQL query để truy vấn trực tiếp dữ liệu từ database hoặc dùng công cụ hỗ trợ để lấy báo cáo | Việc lấy báo cáo dữ liệu trực tiếp từ NoSQL chưa được hỗ trợ tốt, thực hiện chủ yếu thông qua giao diện ứng dụng |
| Mở rộng dữ liệu | Khi muốn bổ xung thêm cột cho một bảng, chúng ta phải khai báo trước | Không cần khai báo trước |
| Thuộc tính cần tuân thủ | SQL tuân thủ theo ACID (Atomicity, Consistency, Isolation and Durability) | NoSQL tuân thủ theo CAP (Consistency, Availability and Partition tolerance) |

Theo những điểm được trình bày ở trên thì NoSQL có khả nhiều điểm tốt hơn SQL, vậy NoSQL có thể thay thế cũng như làm việc tốt hơn SQL hay không. Đó là câu hỏi của nhiều nhà doanh nghiệp phải đối mặt ngày nay. Mặc dù SQL là một tùy chọn khả thi ở hầu hết tình huống, nhưng đối mặt với lượng dữ liệu khổng lồ cũng như nhu cầu phân tích dữ liệu lớn ngày nay thì nó không còn phù hợp. Đây là lý do chính cho sự phổ biến của NoSQL trong các công ty Internet lớn như Google, Amazon,v.v.

Nhìn vào chức năng chính của cả hai cơ sở dữ liệu đều phục vụ cùng một mục đích là lưu trữ dữ liệu nhưng cách cả hai thực hiện thì hoàn toàn khác nhau. Trong khi SQL quản lý và truy xuất tốt cho các database có cấu trúc và mối quan hệ của bảng phức tạp thì NoSQL sẽ phát huy hiệu quả ở các database có kích thước lớn vì tính linh hoạt là một yêu cầu quan trọng được lược đồ động của chúng đáp ứng.

Do đó, giữa SQL và NoSQL không có cái nào tốt hơn cái nào, mỗi cái sẽ phát huy tốt khả năng của mình ở từng lĩnh vực riêng. Và sự lựa chọn giữa SQL và NoSQL hoàn toàn phụ thuộc vào hoàn cảnh cá nhân vì cả hai đề có ưu điểm cũng như nhược điểm riêng.

1. **Ưu điểm và nhược điểm của NoSQL**
   1. **Ưu điểm của NoSQL**

Cơ sở dữ liệu NoSQL đơn giản hóa việc phát triển ứng dụng, đặc biệt đối với các ứng dụng web thời gian thực tương tác, chẳng hạn như những ứng dụng sử dụng API [REST](https://www.techtarget.com/searchapparchitecture/definition/RESTful-API) và web services.

Các cơ sở dữ liệu này cung cấp tính linh hoạt cho dữ liệu chưa được chuẩn hóa, yêu cầu mô hình dữ liệu linh hoạt hoặc có các thuộc tính khác nhau cho các thực thể dữ liệu khác nhau.

Chúng cung cấp khả năng mở rộng cho các tập dữ liệu lớn hơn, vốn phổ biến trong các ứng dụng phân tích và trí tuệ nhân tạo (AI).

Cơ sở dữ liệu NoSQL phù hợp hơn với các yêu cầu về đám mây, di động, mạng xã hội và dữ liệu lớn.

Chúng được thiết kế cho các trường hợp sử dụng cụ thể và dễ sử dụng hơn cơ sở dữ liệu SQL hoặc quan hệ có mục đích chung cho các loại ứng dụng đó.

* 1. **Nhược điểm của NoSQL**

Mỗi cơ sở dữ liệu NoSQL có cú pháp riêng để truy vấn và quản lý dữ liệu. Điều này trái ngược với SQL, vốn là ngôn ngữ chung cho các hệ thống cơ sở dữ liệu quan hệ và SQL.

Việc thiếu lược đồ cơ sở dữ liệu cứng nhắc và các ràng buộc sẽ loại bỏ các biện pháp bảo vệ tính toàn vẹn dữ liệu được tích hợp trong các hệ thống cơ sở dữ liệu SQL và quan hệ.

Cần có một lược đồ với một số loại cấu trúc để sử dụng dữ liệu. Với NoSQL, điều này phải được thực hiện bởi nhà phát triển ứng dụng thay vì quản trị viên cơ sở dữ liệu.

Bởi vì hầu hết các cơ sở dữ liệu NoSQL sử dụng mô hình nhất quán cuối cùng, chúng không cung cấp mức độ nhất quán dữ liệu giống như cơ sở dữ liệu SQL. Đôi khi, dữ liệu sẽ không nhất quán, điều đó có nghĩa là chúng không phù hợp cho các giao dịch yêu cầu tính toàn vẹn ngay lập tức, chẳng hạn như giao dịch ngân hàng và ATM.

Bởi vì cơ sở dữ liệu NoSQL mới hơn, không có tiêu chuẩn ngành toàn diện như với SQL.

1. **Cách hoạt động của NoSQL**

Cơ sở dữ liệu NoSQL sử dụng nhiều mô hình dữ liệu để truy cập và quản lý dữ liệu. Các loại cơ sở dữ liệu này được tối ưu hóa dành riêng cho các ứng dụng yêu cầu mô hình dữ liệu linh hoạt có lượng dữ liệu lớn và độ trễ thấp, có thể đạt được bằng cách giảm bớt một số hạn chế về tính nhất quán của dữ liệu của các cơ sở dữ liệu khác. Trong cơ sở dữ liệu NoSQL, hồ sơ về một đối tượng thường được lưu trữ dưới dạng văn bản JSON. Với từng đối tượng, các thông tin về đối tượng được lưu trữ dưới dạng thuộc tính trong một văn bản duy nhất. Trong mô hình này, dữ liệu được tối ưu hóa cho việc phát triển trực quan và khả năng thay đổi quy mô theo chiều ngang.

1. **Ứng dụng NoSQL**

Đối với những tập dữ liệu lớn, đặc biệt là những tập dữ liệu cực lớn thì NoSQL chính là sự lựa chọn hoàn hảo. Việc sử dụng cơ sở dữ liệu NoSQL một cách hợp lý sẽ đem lại hiệu quả ứng dụng cao, mang lại những lợi ích thiết thực nhất. Cụ thể, những trường hợp mà chúng ta nên sử dụng cơ sở dữ liệu NoSQL như:

* *Khi cần mang tới tốc độ phát triển nhanh chóng*

Cơ sở dữ liệu NoSQL cho phép nhà phát triển có thể kiểm soát tốt được cấu trúc của dữ liệu nên phù hợp cho phương pháp phát triển Agile hiện đại. Thông qua việc phát triển nhanh, lặp lại và cập nhật code thường xuyên. Đối với cơ sở dữ liệu quan hệ SQL, khi nhà phát triển yêu cầu người quản trị cơ sở dữ liệu SQL thay đổi cấu trúc của cơ sở dữ liệu thì tải lại dữ liệu, chỉnh sửa code… có thể làm chậm quá trình phát triển.

* *Khi yêu cầu cấu trúc của nhiều dạng dữ liệu khác nhau cần được xử lý*

Cơ sở dữ liệu NoSQL phù hợp để lưu trữ và mô hình hóa dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc trong một cơ sở dữ liệu. Cơ sở dữ liệu NoSQL cho phép thực hiện lưu trữ các dữ liệu tương tự như các đối tượng được sử dụng trong những ứng dụng, cũng giảm nhu cầu dịch từ dạng dữ liệu được lưu trữ sang dạng dữ liệu lấy trong code.

* *Khi cơ sở dữ liệu SQL không đáp ứng được nhu cầu lưu trữ lớn*

Khả năng xử lý dữ liệu lớn như một phần của kiến trúc cơ bản. Bởi thế, việc sử dụng cơ sở dữ liệu NoSQL sẽ dựa trên chiến lược mở rộng quy mô (scale-out strategy), từ đó giúp việc mở rộng quy mô cho tới khối lượng dữ liệu lớn rẻ hơn nhiều so với khi sử dụng phương pháp mở rộng quy mô mà cơ sở dữ liệu SQL thực hiện.

* *Hệ thống sở hữu lượng truy cập cao, không cho phép downtime*

Cơ sở dữ liệu NoSQL đảm bảo việc hoạch định kế hoạch rõ ràng về mở rộng lưu lượng truy cập. Không chỉ vậy, những kiến trúc mở rộng quy mô lúc này cũng thực sự mang lại lợi ích như việc nâng cấp cơ sở dữ liệu hay thay đổi cấp trúc mà hoàn toàn không yêu cầu cần thời gian downtime.

* *Khi cần các mô hình ứng dụng mới hỗ trợ tốt hơn*

NoSQL cho phép một cơ sở dữ liệu cụ thể có thể phục vụ cả những tác vụ nghiệp vụ, cũng như phân tích từ cùng một cơ sở dữ liệu. Lúc này, một data warehouse riêng biệt sẽ được dùng hỗ trợ cho quá trình phân tích. Cơ sở dữ liệu NoSQL với công nghệ đám mây sẽ nhanh chóng thích nghi tốt được quá trình tự động hóa, trở thành một phần của đám mây đó.

1. **Tìm hiểu Redis**
2. **Khái niệm**

Redis (Remote Dictionary Server) là một cơ sở dữ liệu nguồn mở dùng để lưu trữ dữ liệu theo dạng key-value. Redis cũng được sử dụng để lưu trữ dữ liệu có cấu trúc như: hashes, lists, sets, sorted sets, strings…, có thể được sử dụng như một database, bộ nhớ cache hay một message broker. Ngoài ra, Redis được đánh giá cao nhờ có thể hỗ trợ được cho các tính năng lưu trữ dữ liệu dựa vào đĩa cứng. Từ đó, cho phép chúng ta có thể phục hồi được các dữ liệu ngay khi hệ thống gặp phải các sự cố. Redis được xem là một trong những cơ sở dữ liệu giá trị và quan trọng và được xếp vào nhóm dữ liệu cơ sở NoSQL.

1. **Ưu điểm và nhược điểm của Redis**
   1. **Ưu điểm của Redis**

* Hỗ trợ các chức năng thêm mới, xóa, chỉnh sửa một cách nhanh chóng.
* Cho phép lưu trữ được dữ liệu dưới dạng Key-Value với kích thước lên đến 512MB.
* Hỗ trợ nhiều cấu trúc dữ liệu linh hoạt.
* Dữ liệu được lưu trữ trên RAM giúp cho việc truy xuất dữ liệu được thực hiện nhanh chóng. Ngoài ra, chúng ta có thể cấu hình để Redis có thể lưu trữ dữ liệu được trên ổ cứng.
* Chúng ta có thể cấu hình cho Key tự động xóa trong một khoảng thời gian nhất định(expire).
* Không bị downtime và ảnh hưởng hiệu suất khi thay đổi quy mô.
* Mã nguồn mở, ổn định.
* Hỗ trợ Queue(hàng đợi) thông qua cơ chế PUB/SUB, chúng ta có thể dùng Redis để làm hệ thống queue cho website xử lý tuần tự từng request.
* Các tính năng Master - slave đều rất thích hợp cho những người dùng muốn gia tăng tính an toàn cho các dữ liệu, thu hẹp hoặc mở rộng cho các không gian lưu trữ Data.
* Hỗ trợ nhiều ngôn ngữ khác nhau: Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go,…
  1. **Nhược điểm của Redis**
* Dữ liệu được chia sẻ dựa trên các vị trí hash gán cho từng Master. Vì vậy nếu Master đang giữ slot gặp trục trặc, dữ liệu trong slot đó cũng sẽ bị mất.
* Client kết nối đến các cluster Redis nên chú ý đến cấu trúc liên kết cụm.
* Khi sử dụng Redis cần nhiều RAM vì cơ chế in-memory.

1. **Các kiểu dữ liệu**

Redis không có table tương tự như RDMS như MySQL, hay PostgreSQL,.. Nó được dùng để lưu trữ data dưới dạng key-value và trên thực tế thì memcache cũng làm như vậy nhưng kiểu dữ liệu của nó lại hạn chế hơn rất nhiều mà không đa dạng như Redis. Gồm các kiểu dữ liệu chính sau:

***3.1 Key***

Key Redis là khóa nhị phân an toàn, điều này có nghĩa là bạn có thể sử dụng bất kỳ chuỗi nhị phân nào làm khóa, từ chuỗi như "foo" đến nội dung của tệp JPEG. Chuỗi trống cũng là một khóa hợp lệ.

Một vài quy tắc khác về khóa:

* Các key quá dài không phải là một ý tưởng tốt. Chẳng hạn, một khóa 1024 byte là một ý tưởng tồi không chỉ về bộ nhớ mà còn bởi vì việc tra cứu khóa trong tập dữ liệu có thể yêu cầu một số so sánh key tốn kém.
* Các key rất ngắn thường không phải là một ý tưởng hay. Có rất ít điểm khi viết "u1000flw" làm khóa nếu thay vào đó bạn có thể viết "user:1000:followers". Cái sau dễ đọc hơn và không gian được thêm vào nhỏ hơn so với không gian được sử dụng bởi chính đối tượng khóa và đối tượng giá trị. Mặc dù các phím ngắn rõ ràng sẽ tiêu tốn ít bộ nhớ hơn một chút, nhưng công việc của bạn là tìm sự cân bằng phù hợp.
* Cố gắng gắn bó với một lược đồ. Ví dụ: "object-type:id" là một ý tưởng hay, như trong "user:1000". Dấu chấm hoặc gạch ngang thường được sử dụng cho các trường có nhiều từ, như trong "comment:4321:reply.to" hoặc "comment:4321:reply-to".
* Kích thước khóa tối đa được phép là 512 MB.

### 3.2 String

String Redis là loại giá trị đơn giản nhất mà bạn có thể liên kết với khóa Redis. Nó là kiểu dữ liệu duy nhất trong Memcached nên những người mới sử dụng nó trong Redis cũng là điều hết sức tự nhiên.

Vì các khóa Redis là string, nên khi chúng ta sử dụng string làm giá trị, chúng ta sẽ ánh xạ một string này sang một string khác. Kiểu dữ liệu string hữu ích cho một số trường hợp sử dụng, chẳng hạn như lưu vào bộ đệm các đoạn hoặc trang HTML.

Với Redis có khả năng làm việc được với string cũng như từng phần của string và thực hiện tăng/giảm cho các giá trị đích thực của integer, float.

Theo mặc định, một string Redis có thể có kích thước tối đa là 512 MB.

Các lệnh cơ bản:

* SET lưu trữ một giá trị chuỗi.
* SETNX chỉ lưu trữ một giá trị chuỗi nếu khóa chưa tồn tại. Hữu ích cho việc thực hiện khóa.
* GET lấy một giá trị chuỗi.
* MGET truy xuất nhiều giá trị chuỗi trong một thao tác.
* INCR tăng giá trị của một key nhất định

### 3.3. List

List được xem là một list của strings và nó được sắp xếp theo thứ tự insert. Do đó, Redis sẽ có thể thực hiện thêm một phần tử vào đầu hoặc cuối list một cách dễ dàng.

List này rất phù hợp cho những bài toán buộc có các thao tác với nhiều phần tử gần đầu hoặc gần cuối. Bởi vì việc thực hiện truy xuất này rất nhanh nên dù bạn phải insert cả triệu phần tử cũng vậy. Tuy nhiên nhược điểm ở đây là: việc thực hiện truy cập vào các phần tử nằm ở giữa list thường rất chậm.

Độ dài tối đa của list Redis là 2^32 - 1 (4.294.967.295) phần tử.

Các lệnh cơ bản:

* LPUSH thêm phần tử mới vào đầu list; RPUSH thêm vào đuôi.
* LPOP loại bỏ và trả về một phần tử từ phần đầu của list; RPOP làm tương tự nhưng từ phần đuôi của list.
* LLEN trả về độ dài của một list.
* LMOVE di chuyển nguyên tử các phần tử từ list này sang list khác.
* LTRIM giảm list xuống phạm vi phần tử đã chỉ định.
* BLPOP loại bỏ và trả về một phần tử từ phần đầu của list. Nếu list trống, lệnh sẽ chặn cho đến khi một phần tử có sẵn hoặc cho đến khi đạt đến thời gian chờ đã chỉ định.
* BLMOVE di chuyển nguyên tử các phần tử từ list nguồn sang list đích. Nếu list nguồn trống, lệnh sẽ chặn cho đến khi có phần tử mới.

### 3.4. Set

Nó thường tập hợp các string (đều không được sắp xếp). Redis có thể hỗ trợ cho các thao tác như: thêm, đọc, xóa đi từng phần tử, thực hiện kiểm tra cho sự xuất hiện của một số phần tử nằm trong tập hợp.

Ngoài ra Redis có khả năng hỗ trợ cho các phép toán tập hợp như: intersect/union/difference.

Kích thước tối đa của set Redis là 2^32 - 1 (4.294.967.295) thành viên.

Các lệnh cơ bản:

* SADD thêm một thành viên mới vào một set
* SREM xóa phần tử được chỉ định khỏi set
* SISMEMBER kiểm tra một chuỗi cho tư cách thành viên đã đặt.
* SINTER trả về set các thành viên mà hai hoặc nhiều set có điểm chung (tức là phần giao nhau).
* SCARD trả về kích thước (còn gọi là số lượng) của một set

### 3.5. Hash

Việc lưu trữ hash table của các cặp key-value thì trong đó các key đều sẽ được sắp xếp một cách ngẫu nhiên mà không theo bất kỳ thứ tự nào. Redis thường hỗ trợ cho những thao tác khác như thêm, đọc, xóa đi từng phần tử, cũng như đọc được tất cả giá trị.

Mỗi hash có thể lưu trữ tối đa 4.294.967.295 (2^32 - 1) cặp giá trị trường. Trên thực tế, hash chỉ bị giới hạn bởi bộ nhớ tổng thể trên các máy ảo lưu trữ triển khai Redis.

Các lệnh cơ bản:

• HSET đặt giá trị của một hoặc nhiều trường trên hash.

• HGET trả về giá trị tại một trường nhất định.

• HMGET trả về các giá trị tại một hoặc nhiều trường nhất định.

• HINCRBY tăng giá trị tại một trường nhất định theo số nguyên được cung cấp

### 3.6. Sorted set

Các sorted set là một loại dữ liệu tương tự như sự kết hợp giữa Set và Hash. Giống như các set, các sorted set bao gồm các phần tử chuỗi duy nhất, không lặp lại, do đó, theo một nghĩa nào đó, một sorted set cũng là một set.

Đây là 1 danh sách mà trong danh sách đó thì mỗi phần tử được xem là map của 1 string (member) và 1 floating-point number (score). Danh sách này được sắp xếp dựa theo score này. Những phần tử của zset đều được sắp xếp dựa theo thứ tự từ score nhỏ tới lớn.

Các lệnh cơ bản:

* ZADD thêm một thành viên mới và điểm liên quan vào một tập hợp đã sắp xếp. Nếu thành viên đã tồn tại, điểm sẽ được cập nhật.
* ZRANGE trả về các thành viên của một tập hợp đã sắp xếp, được sắp xếp trong một phạm vi nhất định.
* ZRANK trả về thứ hạng của thành viên được cung cấp, giả sử sắp xếp theo thứ tự tăng dần.
* ZREVANK trả về thứ hạng của thành viên được cung cấp, giả sử tập hợp được sắp xếp theo thứ tự giảm dần.

## 4. Redis Persistence là gì

Persistent dùng để chỉ việc ghi dữ liệu vào các bộ nhớ ổn định, chẳng hạn như ổ SSD. Redis cũng cung cấp một số tuỳ chọn như dưới đây:

* RDB (Redis Database): Thực hiện các snapshot của tập dữ liệu trong các khoảng thời gian nhất định.
* AOF (Append Only File): Viết lại mọi thao tác ghi mà server nhận được, chạy mỗi khi server khởi động. Các lệnh được log theo định dạng giống như giao thức Redis theo kiểu append-only (chỉ thêm). Redis có khả năng viết lại log nếu background có kích thước quá lớn.
* No persistence: Người dùng vẫn hoàn toàn có thể disable tín năng này nếu chỉ cần dữ liệu tồn tại khi server đang chạy.
* RDB + AOF: Nếu kết hợp AOF và RBD, cần lưu ý rằng khi Redis restart thì file AOF sẽ được dùng để xây dựng lại tập dữ liệu ban đầu.

## 5. Ứng dụng Redis

Ứng dụng của Redis ngoài tính năng lưu trữ key-value trên RAM thì Redis còn hỗ trợ tính năng sắp xếp query, backup dữ liệu trên đĩa cứng cho phép bạn có thể phục hồi dữ liệu khi hệ thống gặp sự cố…và có thể nhân bản (Chạy nhiều Server Redis cùng lúc).

* **Caching:** Ứng dụng phổ biến nhất của Redis là sử dụng như một session cache. Sử dụng làm bộ nhớ đệm và tạo ra được tốc độ ghi nhanh có thể chia sẻ được nhiều dữ liệu giữa các ứng dụng hoặc làm database tạm thời. Việc khởi động lại khi maintain, thay đổi cấu hình sẽ không làm mất dữ liệu đang cache. Một khách hàng đương nhiên là sẽ không muốn các sản phẩm của mình trong giỏ đột nhiên biến mất vì một lý do nào đó. Ngoài ra Redis có thể sử dụng để làm **Full Page Cache** cho website. Cũng vì tính nhất quán của Redis, cho dù restart Redis thì người dùng cũng không có cảm nhận chậm khi tải trang.
* **Counter:** được sử dụng làm bộ đếm. Với thuộc tính tăng giảm thông số rất nhanh trong khi dữ liệu được lưu trên RAM, sets và sorted sets được sử dụng thực hiện đếm lượt view của một website, được sử dụng thực hiện các bảng xếp hạng trong game. Redis còn hỗ trợ thread safe do đó nó có thể đồng bộ dữ liệu giữa các request.
* **Publish/Suscribe (Pub/Sub):** Tạo kênh chia sẻ dữ liệu. Redis sở hữu tính năng có thể hỗ trợ tạo ra các channel để người sử dụng có thể trao đổi dữ liệu giữa publisher và subscriber tương tự như một channel trong Socket Cluster hoặc các topic trong Apache Kafka. ***Ví dụ***: Pub/Sub được sử dụng theo dõi các kết nối trong mạng xã hội hoặc các hệ thống chat.
* **Queues:** Tạo hàng đợi để xử lý lần lượt các request. Redis cho phép lưu trữ theo list và cung cấp rất nhiều thao tác với các phần tử trong list, vì vậy nó còn được sử dụng như một message queues.

# Tài liệu tham khảo

[1] Trần Quốc Đạt (2017). *Giới thiệu về NoSQL*. Truy cập vào 15/12/2022 từ

https://viblo.asia/p/gioi-thieu-ve-nosql-database-djeZ1a9jZWz.

[2] Thanh Huyen (2019). *Tìm hiểu về Cơ sở dữ liệu Phi quan hệ - NoSQL*. Truy cập vàp 15/12/2022 từ <https://quantrimang.com/cong-nghe/co-so-du-lieu-phi-quan-he-nosql-160708>.

[4] AWS (2016). *Cơ sở dữ liệu NoSQL là gì*. Truy cập vào ngày 15/12/2022 từ

<https://bom.so/xmBVTk>.

[5] Mona media(2017) *. NoSQL là gì? Những thông tin cần biết về cơ sở dữ liệu NoSQL.* Truy cập vào 15/12/2022 từ https://mona.media/nosql-la-gi/.

[5] Nguyen Hung (2022). *Redis là gì? Toàn bộ những ưu điểm Redis mang lại.* Truy cập vào 16/12/2022 từ <https://vietnix.vn/redis-la-gi/>.

[6] dinhhoanglong91 (2015). *Tổng quan về Redis*. Truy cập vào 16/12/2022 từ [*https://viblo.asia/p/tong-quan-ve-redis-NznmMdXzMr69*](https://viblo.asia/p/tong-quan-ve-redis-NznmMdXzMr69)*.*