**CHƯƠNG I. TÌM HIỂU VỀ NOSQL DATABASE**

**Mở Đầu**

**Mục Lục**

**1. Thời Đại NoSQL**

**2. NoSQL Là Gì?**

2.1 Định Nghĩa

2.2 Một số thuật ngữ liên quan

**3. Phân loại NoSQL**

3.1 Dạng Column Family Stores

3.2 Dạng Key- Values Stores

3.3 Dạng Document Stores

3.4 Dạng Graph Database

**4. So sánh NoSQL Và SQL**

**5. Tại sao Lại có Nhiều Cơ sở dữ liệu NoSQL**

**6. Ưa điểm và nhược điểm của NoSQL**

6.1 Ưa điểm

6.2 Nhược điểm

**7.Tổng Kết**.

**8.Tài liệu và Link tham khảo**

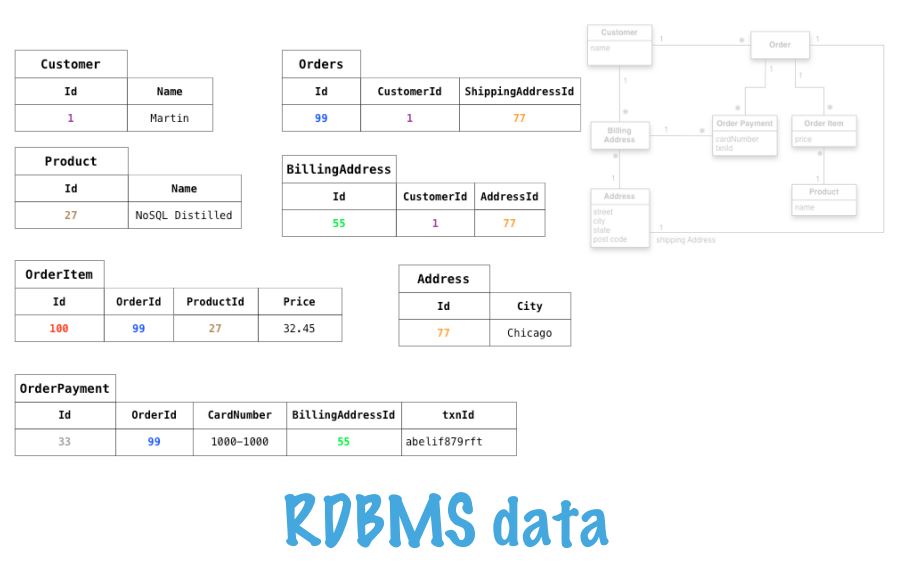
**1.Thời Đại Của NoSQL**

* Nhắc lại một chút, database(DB) là một cơ sở dữ liệu, gồm các bảng, hàng, cột.
* Còn MySQL, Microsoft SQL server, MongoDB,.. là hệ quản trị cơ sở dữ liệu (DBMS).
* SQL không phải là DB ,cũng không phải là DBMS . Nó là viết tắt của Structure Query ( Ngôn ngữ truy vấn cấu trúc). Ngôn ngữ này truy vấn trên nền một RDBMS( hệ quản trị CSDL quan hệ ) . Đây là thứ chúng ta đã được học trong môn “cơ sở dữ liệu “.
* Trong RDBMS , dự liệu được lưa thành nhiều bảng. Mỗi bảng sẽ có nhiều cột, nhiều hàng, ta sử dụng SQL để truy vấn như sau:
* **SELECT Name, Age FROM Students Where Score &gt;8**
* RDBMS (MySQL , Microsoft SQL Server , Oracle…) được sử dụng rất rộng rãi, trong hầu hết các ứng dụng , vì một số lý do sau :

1. Tính ACID ( Atomicity, Consistency, Isolation, Durability) của một transaction được đảo bảo.
2. Với databbase chuẩn 3, dữ liệu được đảm bảo tính đồng nhất và toàn vẹn ( Consistency)
3. Có rất nhiều driver cho mọi ngôn ngữ : Java , C# , PHP.
4. Số lượng lập trình viên biết dùng SQL rất nhiều

Tuy nhiên, RDBMS vẫn còn một số khuyết điểm :

* Việc mapping giữa các bảng trong database với các object trong code khá rắc rối và rất phức tạp. ( Mặc dù một số ORM như Entity Framework, Hibernate đã đơn giản hóa chuyện này ).
* Performance sẽ rất chậm khi join nhiều bảng để lấy dữ liệu ( Đó là lý do ta sử dụng “ giảm chuẩn “ để tăng hiệu suất cho RDBMS)
* Việc thay đổi cấu trúc dữ liệu ( Thêm/ xóa bảng) rất mệt moit , kéo theo vô số thay đổi trên code.
* Không làm việc được với dữ liệu không có cấu trúc ( un-structure).
* RDBMS được thiết kế để chạy trên máy chủ. Khi muốn mở rộng , nó khó chạy trên được nhiều máy ( clustering)



Dữ liệu được lưa thành nhiều bảng trong RDBMS , khi query ta phải join lại rất khó khăn

Do đó để giải quyết được những khuyết điểm của RDBMS , thế hệ database tiếp theo ra đời là Nosql Database.

* Đây là thế hệ database không rằng buộc, phân tán , mã nguồn mở , có khả năng mở rộng theo chiều ngang, có thể lưa trữ và xử lý từ một lượng rất nhỏ cho đến hàng petabytes dữ liệu trong hệ thông có độ chịu tải và chịu lỗi cao với những đòi hỏi về tài nguyên phần cứng thấp.
* Đặc điểm nhận dạng : Có schema-free , hộ trợ mở rộng dễ dàng, API đơn giản , eventual consistency ( Nhất quán cuối) và/hoặc transactions hạn chế trên các thành phần dữ liệu đơn lẻ, không giới hạn không gian dữ liệu,…

NoSQL storage đặc biệt phổ dụng cho phép người dùng tạo hàng tỷ nội dụng trên web. Do đó , dữ liệu lớn rất nhanh vượt qua giới hạn phần cứng và cần phải giải quyết bằng bài toán phân tán.

**2. NoSQL Là Gì ?**

2.1 Định Nghĩa

NoSQL là một xu hướng cơ sở dữ liệu mà không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mền. NoSQL có nghĩa là No-Relational (NoRel) – Không rằng buộc. Tuy nhiên thuật ngữ đó ít phổ dụng hơn và ngày ngày người ta thường dịch NoSQL thành Not Only SQL – Không chỉ SQL .

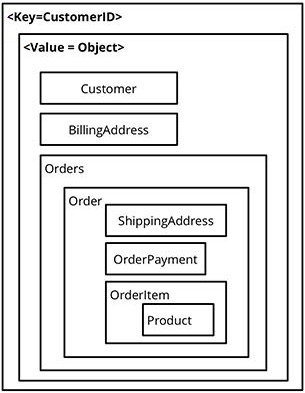
Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên chung cho các lightweight open source relational database( cơ sở dữ liệu quan hệ nguồn mở nhỏ ) nhưng không sử dụng SQL cho truy vấn

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

Là một công nghê cơ sở dữ liệu được thiết kế hộ trợ các yêu cầu của các ứng dụng điện toán đám mây và kiến trúc vượt qua quy mô, hiệu quả , mô hình dữ liệu, và các giới hạn phân phối dữ liệu của cơ sở dữ liệu quan hệ (RDBMS)

Công nghệ NoSQL ban đầu được tạo ra và được sử dụng bới các nhà lãnh đạo trên Internet như Facebook, Google ,Amazon, và những người yêu cầu hệ thống quản lý cơ sở dữ liệu có thể viết và đọc dữ liệu bất cứ nơi nào trên thế giới, trong khi mở rộng quy mô và cung cấp hiệu suất trên toàn bộ dữ liệu lớn và hằng triệu người sử dụng

Dữ liệu được lưa dưới dạng object . mặc dù bị trùng lặp nhưng truy vấn rất nhanh và đơn giản



**3. Một Số Thuật Ngữ Liên Quan.**

Relational (Rằng buộc) thuật ngữ sử dụng đến các mối quan hệ giữa các bảng trong cơ sở dữ liệu quan hệ (RDBMs) sử dụng mô hình khóa gồm 2 loại khóa: khóa chính và khóa phụ( primary key + foreign key) để rằng buộc dữ liệu nhàm thể hiện tính nhất quán dữ liệu từ các bẳng khác nhau.

Non- Relational (Không rằng buộc ) là khái niệm không sử dụng các rằng buộc dữ liệu cho nhất quán dữ liệu ở NoSQL database.

High Availability (Tính sẵn sàng) : Do chấp nhân sự trùng lặp trong lưa trữ nên nếu một node( commodity machine) nào đó bị chết cũng không ảnh hưởng tới toàn bộ hệ thống.

High Scalability ( khả năng mở rộng ) : Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ thống.

Eventual consistency (Nhất quán cuối) : Tính nhất quán của dữ liệu không cần phải đảm bảo ngay tức khác sau mỗi phép write. Một hệ thống phân tán chấp nhận những ảnh hưởng theo phương thức lan truyền và sau một khoảng thơi gian ( không phải ngay tức khắc), thay đổi sẽ di đến mọi điểm trong hệ thống, tức là cuối cùng ( eventually) dữ liệu trên hệ thống sẽ trở lại trạng thái nhất quán

Vertical scalable ( khả năng mở rộng theo chiều dọc ): Khi dữ liệu lớn về lượng, phương pháp tăng cường khả năng lưa trữ và xử lý bằng việc cải tiến phần mền và cải thiện phần cứng trên một máy tính đơn lẻ được gọi là khả năng mở rộng chiều dọc. Ví dụ tăng cường CPUs, cải thiện đĩa cứng , bộ nhớ trong một máy tính,... cho DBMs nằm trong phạm trù này. Khả năng mở rộng chiều dọc còn có một thuật ngữ khác scale up.

Horizontal scalable (khả năng mở rộng chiều ngang) : Khi dữ liệu lớn về lượng , phương pháp tăng cường khả năng lưa trữ và xử lý dùng nhiều máy tính phân tán . Phân tán dữ liệu được hộ trợ bởi các phần mền tức cơ sở dữ liệu.

Distributed Data ( phân tán dữ liệu ): Mô hình lưa trữ phân tán các file hoặc dữ liệu ra nhiều máy tính khác nhau trong mạng LAN hoặc Internet dưới sự kiểm soát của phần mền .

Deployment Flexibility ( Triển khai linh hoạt ): Việc bổ sung thêm/loại bỏ các node , hệ thống sẽ tự động nhận biết để lưa trữ mà không cần phải can thiệp bằng tay. Hệ thống cũng không đòi hỏi cấu hình mạnh , đồng nhất .

Durability ( Lưa trữ tốt) : Dữ liệu có thể tồn tại trong bộ nhớ máy tính nhưng đồng thời cũng được lưa trữ lại đĩa cứng.

- Trong khi giá thành phần cứng ngày càng giảm. tốc độ xử lý, bộ nhớ ngày tăng thì horizontal scalable là một sự lựa chọn đúng đắn . Hằng trăm máy tính nhỏ được chập lại tạo thành một hệ thống tính toán mạnh hơn nhiều sơ với vi xử lý RISC truyền thống đơn lẻ. Mô hình này tiếp tục được hộ trợ bới các công nghệ kết nối Myrinet và InfiniBand. Từ đó chúng ta có thể quản lý , bảo trì từ xa, xây dụng batch procession( Xử lý đồng loạt tập lệnh ) tốt hơn. Do những đòi hỏi về tấc độ xử lý I/O cao , lượng cực lớn dữ liệu …scale horizontally sẽ thúc đẩy các công nghệ lưa trữ mới phát triển giống như object storage devices (OSD)

4. So Sánh NoSQL và SQL

Tóm lại , NoSQL database chỉ là một kiểu database có các lưa trữ , truy vấn dữ liệu hoàn toàn khác so với RDBMS và SQL.

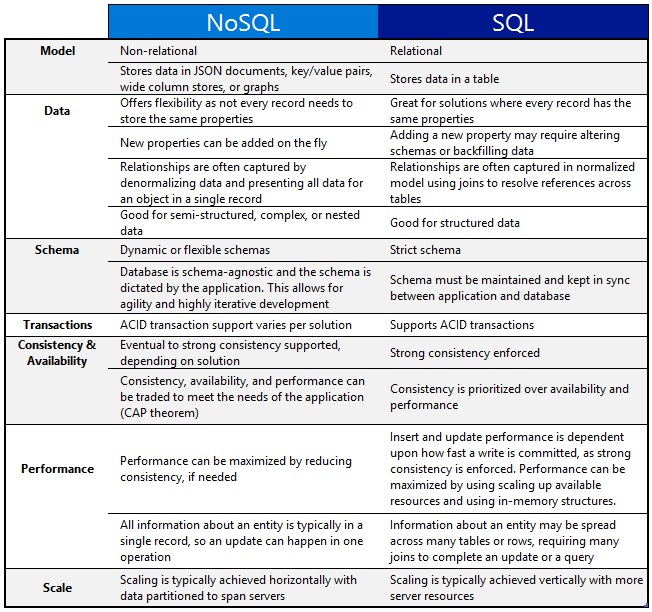
NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng (Scalability ). Chính những ưa điểm trên mà NoSQL được sử dụng nhiều trong các dự án Big Data, các dự án Real-time, số lượng dữ liệu nhiều.

* Liệu NoSQL có thay thế được hoàn toàn RDBMS và NoSQL được không ? Câu trả lời là KHÔNG. Trong tương lai, RDBMS vẫn sẽ giữ nguyên chỗ đúng của mình. Một ứng dụng không chỉ có một database duy nhất, và có thể kết hợp cả SQL lẫn NoSQL.

So sánh SQL và NoSQL

|  |  |  |
| --- | --- | --- |
| Tính Năng | SQL | NoSQL |
| Hiệu xuất | Kém hơn NoSQL vì khi truy vấn nó phải tính toán, kiểm tra và xử lý các mối quan hệ các bảng | Tốt hơn SQL vì nó bỏ qua các rằng buộc |
| Mở rộng theo chiều ngang | Có thể thực hiện được nhưng quá trình mở rộng sẽ rất phức tạp nếu đã tồn tại dữ liệu trong database | Mở rộng dễ dàng |
| Tấc độ read/write | Kém hơn NoSQL vì phảm đảm bảo tính rằng buộc dữ liệu giữa các bảng  Nếu sử dụng nhiều server thì phải bảo toàn tính nhất quán về dữ liệu ở các server với nhau | Tấc độ nhanh hơn SQL vì bỏ qua các cơ chế rằng buộc bảng  Vì dữ liệu được lưa trong RAM , sau đó mới đẩy xuống HDD và nó tính nhất quán cuối |
| 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 một node cần phải shutdown hệ thống trong một khoảng thời gian | Vì tính nhất quán cuối nên sẽ không cần phải shutdown hệ thống |
| Truy vấn báo cáo | Dễ dàng sử dụng ngôn ngữ SQL query để truy vấn trực tiếp dữ liệu từ database hoặc dùng công cụ hộ trợ để lấy báo cáo | Việc báo dữ dữ liệu trực tiếp NoSQL chưa được hộ trợ tốt , thực hiện chủ yếu thông qua giao diện ứng dụng |
| Mở rộng dữ liệu | Khi muốn bổ sung thêm cột cho một bảng ,chúng ta phải khai báo trước. | Không cần khai báo trước |
| Ứng dụng | Sử dụng để xây dựng những hệ thống có quan hệ chặt chẽ và cần tính đồng nhất về dữ liệu như : tài chính, ngân hàng ,chứng khoán | Sử dụng xây dựng những hệ thống lưa trữ thông tin lớn, không quá quan trọng vấn đề đồng nhất dữ liệu trong một thời gian nhất định. Vd như : báo trí, mạng xã hội, diễn đàn, shopping.. |

Bảng so sánh NoSQL và SQL của Microsoft . Link tham khảo bên dưới



* NoSQL và RDBMS được thiết kế để hỗ trợ các yêu cầu ứng dụng khác nhau và thường thì họ cùng tồn tại trong hầu hết các doanh nghiệp. Những điểm quyết định quan trọng khi sử dụng gồm :

|  |  |
| --- | --- |
| RDBMS | NoSQL |
| Phát triển các ứng dụng tập trung (ví dụ ERP) | Phát triển các ứng dụng phi tập trung( vi dụ web , điện thoại di động và IOT |
| Các ứng dụng vừa và nhỏ để có tính sẵn sàng phục vụ cao | Tính sẵn sàng liên tục, không có thời gian chết. |
| Truy cập vào dữ liệu với tấc độ vừa phải | Truy cập vào dữ liệu tấc độ cao.( thiết bị, cảm biến , vv) |
| Dữ liệu đến từ một hoặc vài địa điểm | Dữ liệu đến từ nhiều địa điểm |
| Chủ yếu là dữ liệu có cấu trúc | Dữ liệu có cấu trúc, bán cấu trúc, phi cấu trúc. |
| Các giao dịch phức tạp, lồng nhau | Các giao dịch đơn giản |
| Mối quan tâm chính là mở rộng việc đọc, việc đọc dễ dàng | Quan tâm cả việc đọc và ghi phải dễ dàng |
| Duy trì dữ liệu vừa phải với việc có thể xóa bỏ dễ dàng | Duy tri khối lượng dữ liệu cao , có thể giữ lại mãi mãi |

5. Tạo sao lại có nhiều NoSQL Database ?

- Theo thống kê của trang : <http://nosql-database.org/> hiện tại lơn hơn 225 loại NoSQL Database . Vậy tại sao lại có nhiều đến vậy ? Sự khác là do :

* Architecture : Một vài NoSQL databases như MongoDB được kiến trúc theo mô hình master/slave gần giống như RDBMS’s . Còn NoSQL Database khác ( như Cassandra) được thiết kế theo mô hình ‘ masterless’ nơi mà tất cả các node trong database cluster là giống nhau. Kiến trúc của NoSQL database có tác động rất lớn tới việc database hộ trợ các yêu cầu như thơi gian hoạt động liên tục , dự đoán hiệu suất…
* Data Model: NoSQL database thường được phân lớp dựa vào data model mà chúng được hộ trợ. Một vài hộ trợ lưa trữ winde-row tabular , trong khi đó một vài lại hộ trợ model kiểu document-oriented , key –value hoặc graph.
* Data Distribution Model : Bởi chúng khác nhau về architecture , NoSQL databases khác nhau về việc chúng được hộ trợ việc đọc, ghi và phân phối dữ liệu. Một vài nền tảng NoSQK như Cassandra hộ trợ việc ghi và đọc trên mọi node trong một cluster và có thể nhân rộng / đồng bộ hóa dữ liệu giữa nhiều trung tâm dữ liệu và các nhà cung cấp điện toán đám mây.
* Development Model : NoSQL databases khác nhau trên API cho nhà phát triển với vài hộ trợ ngôn ngữ SQL ( vd như Cassadra’s CQL).

6. Lợi ích Của việc dùng NoSQL

Có rất nhiều lợi ích khi dùng NoSQL Database ví dụ như :

* Nó là nguồn mở : Các sản phẩm nguồn mở được đưa ra cho những người phát triển với một vài lợi ích lơn lao, trong khi đó có tình trạng không có chi phí của chúng. Những lợi ích khác : phần mền nguồn mở có xu hướng sẽ là tin cậy hơn ,an ninh hơn và nhanh hơn để triển khai so với các lựa chọn thay thế sở hữa độc quyền.
* Việc mở rộng phạm vi là mền dẻo : NoSQL thây thế câu thần chú cũ của của các nhà quản trị cơ sở dữ liệu về ‘ mở rộng phạm vi ‘ với một thứ mới ‘ mở rộng ra ngoài ‘. Thay vì bổ sung thêm các máy chủ lớn hơn để điều khiển tải dữ liệu hơn, thì cơ sở dữ liệu NoSQL cho phép một công ty phân tán tải qua nhiều máy chủ khi mà tải gia tăng.
* Các cơ sở dữ liệu NoSQL khác nhau cho những dự án khác nhau :
  + MongoDB và Redis là những lựa chọn tốt cho việc lưa trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên , như một số đếm truy cập web chẳng hạn.
  + Hadoop , một cơ sở dữ liệu dạng tự do , phân tán tốt công việc lưa trữ các dự liệu lơn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.
  + Memcache , một cơ sở dữ liệu nhất thời chóng tàn, tuyệt vời trong lưa trữ các phiên làm việc web , các khóa , và các con số thống kê ngắn hạn.
  + Cassandra và Riak ( các lưa trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.

-NoSQL được các hãng lơn sử dụng : Các công ty như Amazon , BBC , Facebook và Google dựa vào các cơ sở dữ liệu NoSQL.

- NoSQL phù hợp với công nghệ đám mây : Các máy chủ ngày nay là không đắt và có thể dễ dàng mở rộng phạm vi được theo yêu cầu có sử dụng một dịch vụ như là Amzon EC2 . Giống như tất các công nghệ đám mây, EC2 dựa vào ảo hóa. Liên kết yếu của ảo hóa là sự thực thi của I/O , với bộ nhớ và CPU các kết nối mạnh.

Các cơ sở dữ liệu NoSQL hầu như sử dụng bộ nhớ qua địa như vị trí ghi đầu tiên – vì thế ngăn chặn được sự thực thi không ổn định của I/O . Và vì NoSQL lưa trữ dữ liệu thường thúc đẩy tính mở rộng phạm vi theo chiều ngang thông qua việc ngăn chia, chúng có khả năng tận dụng được việc cung cấp mền dẻo của đám mây.

**7. Nhược Điểm Của NoSQL**

*- Nguồn mở có thể có nghĩa là sự hộ trợ không đồng đều cho các doanh nghiệp :*

* Trong khi các nhà cung cấp chủ chốt của RMBMS như Oracle , IBM hay Sybase đưa ra sự hộ trợ tốt nổi tiếng cho khách hàng doanh nghiệp cỡ vừa , thì các doanh nghiệp nhỏ hơn, thường là các nhà cung cấp nguồn mở mới thành lập không thể được mong đợi cung cấp được sự hộ trợ có thể so sánh được – ngoại trừ một nhúm khách hàng blue chip
* Nhà cung cấp nguồn mở trung bình thiếu sự tiếp cận toàn cầu, các dịch vụ hộ trợ và sự tin cậy của một Oracle hay IBM

- *Chưa đủ “chín” cho các doanh nghiệp* : Dù chúng được triển khai tại một số công ty lơn thì các cơ sở dữ liệu NoSQL vẫn phải đối mắt với một số vấn đề vế sự tin cậy chính với nhiều doanh nghiệp. Điểm sống còn đối với sự thiếu của NoSQL về độ chín muồi và các vấn đề tính không ổn định có thể, trong khi trích ra tính chính muồi, giàu chứng năng và ổn định của các RDBMS được thiết lập từ lâu.

- *Những hạn chết về tri thức nghiệp vụ:*

Có một vài câu hỏi xung quang những khả năng về tri thức nghiệp vụ (BI) của các cơ sở dữ liệu NoSQL : Liệu cơ sở dữ liệu này có thể cung cấp dạng phân tích dữ liệu lớn và mạnh mà các doanh nghiệp đã quen với RDBMS? Cần bao nhiêu sự tinh thông về lập trình để có để tiến hành những truy vấn và phân tích hiện đại ?

Các câu trả là không tích cực. Các cơ sở dữ liệu NoSQL không có nhiều sự đeo bám tới các công cụ BI thường được sử dụng , trong khi những yêu cầu và phân tích hiện đại đơn giản nhất có liên quan khác tới sự tinh thông về lập trình. Tuy vậy , các giải pháp là sẵn sàng .Quest Software , ví dụ , đã tạo ra Toad cho các cơ sở dữ liệu đám mây, mà nó phân phối các khả năng truy vấn hiện đại tới một số cơ sở dữ liệu NoSQL.

* *Thiếu sự tính thông*

Tính rất mới mẻ của NoSQL có nghĩa là không có nhiều lập trình viên và người quản trị mà biết công nghệ này – làm cho các công ty khó khăn tìm người với sự tinh thông phù hợp. Đối lại thế giới của RDBMS có hàng ngàn người đủ tư cách.

* *Những vấn đề về tính tương thích* :

Không giống với các cơ sở dữ liệu quan hệ , các cơ sở dữ liệu NoSQL chia sẻ ít theo cách thức của tiêu chuẩn . Mỗi cơ sở dữ liệu NoSQL có các giao diện lập trình ứng dụng API riêng của mình, các giao diện truy vấn độc nhất vô nhị, và những sự riêng biệt. Sự thiếu hụt các tiêu chuẩn có nghĩ là nó không có khả năng để chuyển một các đơn giản từ một nhà cung cấp sang một nhà cung cấp khác, nếu bạn trở nên không hạnh phúc với dịch vụ

. 8. Phân loại cơ sở dữ liệu NoSQL

Tính đến thời điểm hiện tại thì có hơn 255 cơ sở dữ liệu NoSQL (<http://nosql-database.org/>) . Dựa theo đặc điểm và thuộc tính của một số loại cơ sở dữ liệu NoSQL có thể chưa ra chúng làm bốn loại. Thực chất thì việc chia chúng ra làm bốn 4 loại như vậy chỉ mang tính chất tương đối, dựa trên những đặc điểm rất khác biệt vì trên tổng thể , chúng vẫn mang tính chất khá giống nhau

* **8.1 Key – Value Store**
* Key –value stores là kiểu lưa trữ đơn giản nhất trong các loại cơ sở dữ liệu NoSQL đồng thời nó cũng là kiểu lưa trữ cho tất các hệ quản trị cơ sở dữ liệu NoSQL. Thông thường các các hệ quản trị cơ sở dữ liệu Key-value lưa trữ dữ liệu dưới dạng key( là một chuỗi duy nhât) liên kết với value có thể ở dạng chuỗi văn bản đơn giản hoặc các tập, danh sách dữ liệu phức tạp hơn. Quá trình tìm kiếm dữ liệu thường sẽ được thực hiện thông qua key, điều này đẫn đến thiếu độ chính xác .

|  |  |
| --- | --- |
| Car | |
| Key | Attributes |
| 1 | Make: Honda  Model: SH  Color: Blue  Year: 2018 |
| 2 | Make: Yamaha  Model: Exciter  Color : Green  Year: 2017 |

Key – Value Store

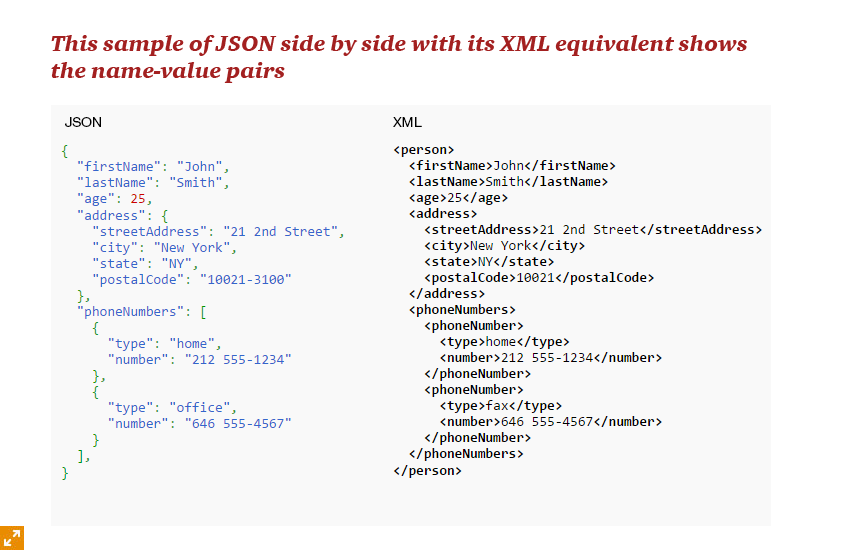
* Các API được cung cấp cho việc truy vấn dữ liệu của các cơ sở dữ liệu NoSQL thường cũng rất đơn giản , về cơ bản hầu hết các cơ sở dữ liệu NoSQL sẽ có các API sau:

**Void Put(string key , byte[] data); byte[] Get(string key) ; void Remove(string key);**

* Tuy nhiên việc triên khai thực tế của các cơ sở dữ liệu Key –Value thường rất phức tập do cấu trúc lưa trữ quá đơn giản. Vi dụ để lưa trữ thông tin một người:
* “ 1234” => “ BaLam, [nguyenbalam190@gmail.com](mailto:nguyenbalam190@gmail.com)”
* “1235” => “ Trung Nghĩa , [trungnghia@gmail.com](mailto:trungnghia@gmail.com)”
* “1236” => “ Văn Thao , [vanthao@gmail.com](mailto:vanthao@gmail.com), Tp,HCM”
* “1237” => “ Văn Thắng”
* Với kiểu lưa trữ này, ta sẽ rất dễ dàng và nhanh chóng truy xuất được thông tin của một người thông qua key, nhưng không hề đơn giản cho việc xử lý những dữ liệu phức tạp. Dễ thấy rằng hệ quản trị cơ sở dữ liệu Key-Value là đơn giản hóa việc lưa trữ dữ liệu , nghĩa là không cần quan tâm tới nội dung là gì .
* Nói cách khác , chúng ta lưa trữ thông tin mà không cần xác điịnh lược đồ. Điều này làm cho phương pháp lưa trữ dữ liệu với Key-Value trở nên đơn giản hơn rất nhiều trong việc xây dựng cũng như khả năng mở rộng là cự kì linh động và hiệu suất cho các thao tác truy vấn dữ liệu cũng nhanh.
* Với sự đơn giản của cách lưa trữ Key-Value làm cho các cơ sở dữ liệu loại này rất phù hợp với các ứng dụng cần truy xuất nhanh chóng và khả năng mở rộng cao, chẳng hạn như quản lý các phiên giao dịch (session) hoặc quản lý các thông tin về giỏ hàng vì trong trường hợp này biết được các ID của phiên giao dịch hoặc ID của khách hàng là rất cần thiết. Hay việc quản lý thông tin của sản phẩm bảo gồm các thông tin liên quan , đánh giá… sẽ được trữ dưới dạng key là mã sản phẩm chẳng hạn và value là các thông tin còn lại của sản phẩm cần lưa trữ. Điều này cho phép truy xuất được tất cả các thông tin về một sản phẩm chỉ thông qua mã sản phẩm cực kì nhanh.
* Một số cơ sở dữ liệu NoSQL tiêu biểu cho dạng này: Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB

**8.2 Document Databases**

Cơ sở dữ liệu Document được thiết kế để quản lý và lưa trữ dữ liệu ở dạng document. Nhưng document này được mã hóa về các dạng chuẩn như XML , JSON ( Javascript Option Notation) hay BSON( Binary JSON) . Khác với kiểu lưa trữ dạng Key- Value , giá trị của cột tring các cơ sở dữ liệu document chứa dữ liệu bán cấu trúc ( Semi-Structured Data) , đặc biệt là cặp thuộc tính name(key )- value . Một cột có thế chưa hàng trăm các thuộc tính như vậy, số lượng , loại thuộc tính được lưa lại có thể khác nhau giữa các dòng. Một điểm khác nữa so với kiểu lưa trữ dữ liệu Key-Value đơn giản là cả key và value đều có thể tìm kiếm trong cơ sở dữ liệu Document.



* Cơ sở dữ liệu Document phù hợp cho việc lưa trữ và quản lý dữ liệu có kích thước lớn như tài liệu văn bản, tin nhắn, cũng như biểu diễn một thực thể cơ sở dữ liệu là Product hay Customer
* Các cơ sở dữ liệu tiêu biểu cho cơ sở dữ liệu Document : CouchDB (JSON) , MongoDB (BSON) ,… đều là mã nguồn mở ( open source ), hướng document( document oriented) và có lược đồ tự do ( shema free).
* Một số cơ sở dữ liệu NoSQL tiêu biểu cho dạng này : Apache Jackrabbit, CouchDB, IBM Lotus Notes Storage Format (NSF), MongoDB, Terrastore, ThruDB, OrientDB, RavenDB,...

MongoDB

* API : BSON
* Protocol : C
* Query Method : Dynamic object – based language & MapReduce,
* Replication : Master Slave & Auto –Sharding
* Writen in: C++.
* Concurrency : Update in place . Misc : Indexing, GridFS,

CouchDB

* API : JSON
* Protocol :REST
* Query Method :MapReduce of JavaScript Funcs.
* Replication : Master Master
* Writen in:Erlang.
* Concurrency : MVCC , Misc.

**8.3 Column Family Stores**

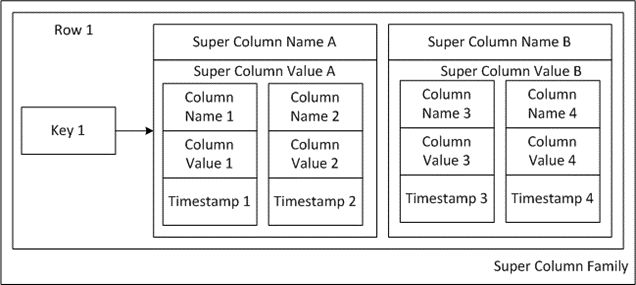
- Column Family được biết đến rộng rãi nhất qua sự triển khai BigTable của Google , nhìn bề ngoài , chúng khá giống với cơ sở dữ liệu quan hệ nhưng thực tế là hoàn toàn khác . Một số sự khác biệt dễ thấy nhất là việc lưa trữ dữ liệu theo dòng đối với các hệ quản trị cơ sở dữ liệu quan hệ với việc lưa trữ theo cột của các hệ quản trị cơ sở dữ liệu Column Family . Và các tư tưởng của cả hai loại cơ sở dữ liệu này cùng hoàn toàn khác nhau. Chúng ta không thể áp dụng cùng một giải pháp mà chúng ta đã sử dụng trong cơ sở dữ liệu quan hệ vào cơ sở dữ liệu Conlumn Family. Bởi vì cơ sở dữ liệu Column Family là các cơ sở dữ liệu phi quan hệ.

Với các cơ sở dữ liệu Column Family , chúng ta cần quan tâm đến các khái niệm sau :

Column Family ( Họ cột) : Một column family là cách thức dữ liệu được lưa trữ trên đĩa. Tất cả dữ liệu trong một cột sẽ được lưa trên cùng một file. Một column family có thể chứa super column hoặc column.

Super column ( Siêu cột) : Một super column có thể được dùng như một dictionnary ( kiểu từ điển ) . Nó là một column có thể chứa những column khác ( mà không phải là super column ).

-Column và super column trong column family database dùng thay thế nhau, có nghĩa là chúng sẽ là 0 byte nếu chúng không có chứa dữ liệu . không giống như một bảng, thứ duy nhất chúng ta cần xác đinh trong column farmily database tên cột và các tùy chọn chính ( không có lược đồ cố định).



* Cơ sở dữ liệu Column Family được thế kế để chạy trên một số lượng lớn các máy , và lưa trữ một lượng dữ liệu cực lớn. chúng ta không thể lưa trữ một lượng dữ liệu lớn dữ liệu trong cơ sở dữ liệu quan hệ vì chắc chắn chúng sẽ nhanh chóng bị sụp đổ hoặc là chết rất nhanh về kích thước của dữ liệu và những truy vấn đó được các cơ sở dữ liệu Column Family xử lý một cách dễ dàng. Các cơ sở dữ liệu Column Family loại bỏ các khái niệm trừa tượng , những thứ làm cho nó cứng nhắt khi chạy trên một cụm máy.

Một số cơ sở dữ liệu NoSQL tiêu biểu cho dạng này :Hadoop/HBase – Apache, BigTable – Google, Cassandra - Facebook/Apache, Hypertable - Zvents Inc/Baidu, Cloudera, SciDB, Mnesia, Tablets,…

**Hadoop / Hbase**

* API :Java / any write
* Protocol : any write call
* Query Method : MapReduce Java /Any exec
* Replication : HDFS Replication
* Writen in: Java

Cassandra

Massively scable , partitioned row store ,architecture , linear scale performance , no single points of failure , read/write support across multiple data centers & cloud availability zones

API /Query Method : CQL and Thrift,

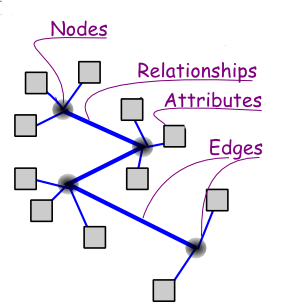
Replication : peer –to-peer

Written in : java

Concurrency : tunable consistency

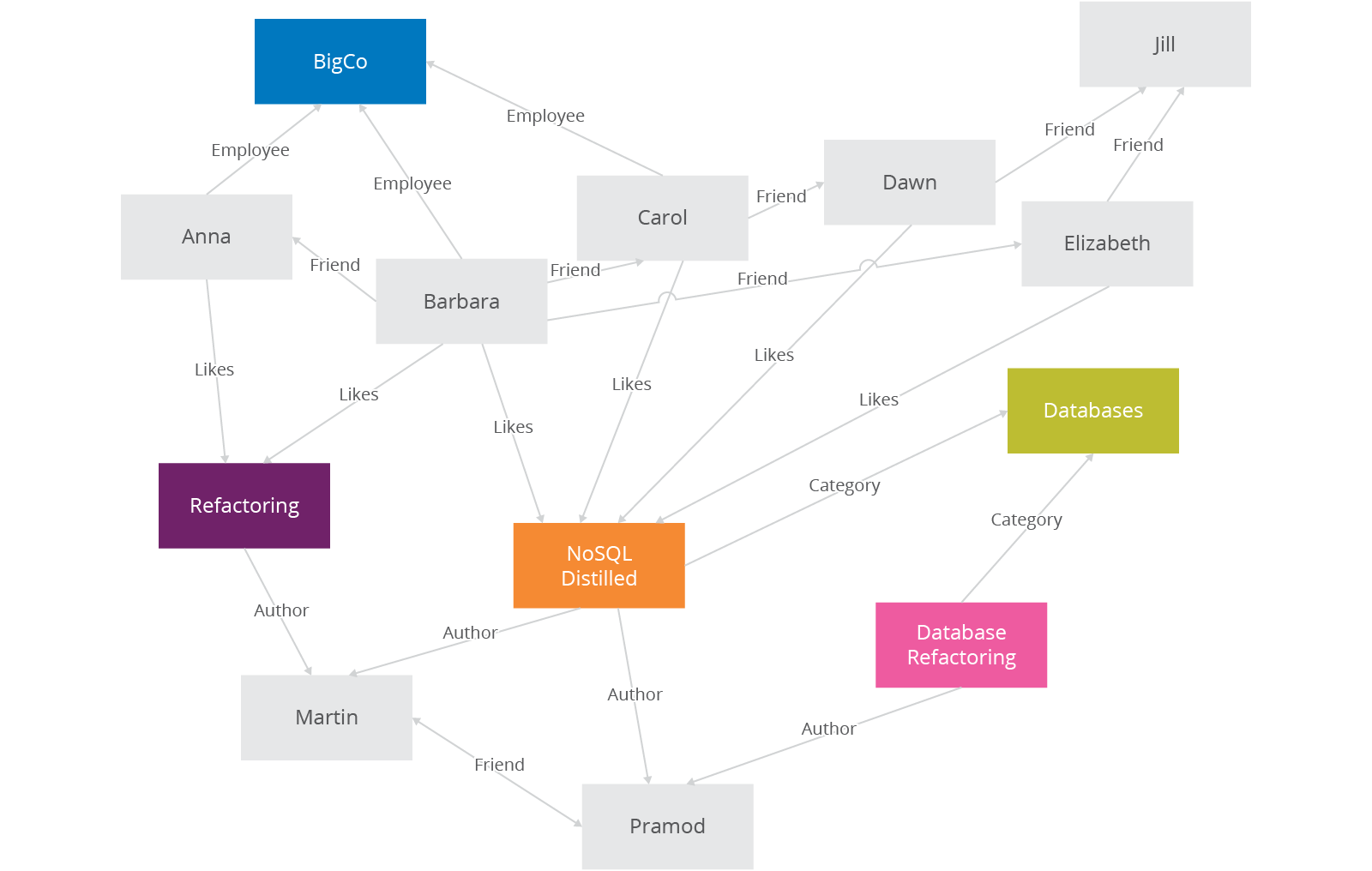
8.4 Graph Database

Cơ sở dữ liệu đồ thị là một cơ sở dữ liệu dạng đồ thị sử dụng các cấu trúc đồ thị với các nút ( nodes) các mối quan hệ ( relationships) , các thuộc tính (properties) để mô tả và lưa trữ dữ liệu



Chúng ta có thể thực hiện những truy vấn phưc tạp hơn như lọc trên các thuộc tính quan hệ , xem xét trọng số của người đó..

Graph Database thường được sử dụng để giải quyết các vấn đề về mạng. Trong thực tế , hầu hết các trang web mạng xã hội đều sử dụng một số hình thức của garph database để làm những việc mà chúng ta đã biết như :kết bạn, bạn của bạn.



Một vấn đề đối với việc mở rộng Graph Database là rất khó khăn để tìm thấy một đồ thị con độc lập, có nghĩ là rất khó khăn để ta phân tán Graph Database thành nhiều mảnh . có rất nhiều nỗ lực nghiên cứa cho việc này nhưng vẫn chưa có bất kì giải pháp nào đáng tin cậy được đưa ra .

Một số sản phẩm tiêu biểu của Graph database là : Neo4j , Sones , AllegroGraph , Core Data, Dex, FlockDB, InfoGrid, OpenLink Virtuoso…

**9. Tổng kết :**

**-**Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi loại cơ sở dữ liệu NoSQL cũng có những đặc điểm riêng. Vì thế chúng thường được dùng cụ thể cho những dự án khác nhau . Ví dụ

-MongoDB và Redis là những lựa chọn tốt cho việc lưa trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên

-Hadoop, một cơ sở dữ liệu dạng tự do, phán tán làm tốt công việc lưa trữ các dữ liệu lớn như các con số thống kê thời tiết, hoặc công việc phân tích nghiệp vụ.

Memcachedb một cơ sở dữ liệu nhất thời chóng tàn, tuyệt vời trong lưa trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.

Canssandra va Riak ( các lưa trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn

10. Tài liệu và link tham khảo

<https://toidicodedao.com/2015/09/24/nosql-co-gi-hay-ho-tong-quan-ve-nosql-phan-1/>

<https://zxthanhhuy.wordpress.com/2015/08/04/cac-loai-nosql-thong-dung/>

<https://toidicodedao.com/2015/09/29/nosql-co-gi-hay-ho-tong-quan-ve-nosql-phan-2/>

<http://nosql-database.org/>

<https://viblo.asia/p/tim-hieu-ve-nosql-Zzb7vDNYMjKd>

<https://www.datastax.com/nosql-databases>

**CHƯƠNG II . TÌM HIỂU VỀ NODE.JS**

**Mục Lục**

**I.Tổng Quan Về Node.js**

1. Node.js là gì?
2. Đặc điểm của Node.js
3. Với nodejs bạn phải làm tất cả
4. Ai sử dụng nodejs
5. Các thành phần quan trọng trong Node.js
   1. Luồng
   2. V8
   3. Quản Lý Package
   4. Unified API
   5. Vòng Lặp Sự Kiện
6. Node.js được sử dụng ở đâu và không nên sử dụng ở đâu ?
7. Cộng đồng

**II .Ưa Điểm Và Nhược Điểm Của NodeJs**

1. Ưa Điểm
2. Nhược Điểm
3. Với ưa điểm nhược điểm trên khi nào chúng ta sử dụng nodejs ?

**III Installation**

1. Installing Nodejs
2. Installing New Modules

**VI.Các Object Cơ Bản**

1. **Global Objects( đối tượng toàn cục)**
   1. Console
   2. Process
   3. Buffer
2. **Sự Kiện (Event)**
   1. EvenEmitter
   2. Kế thừa từ EventEmitter( Inheriting From EventEmitter)
   3. Loại bỏ các sự kiện lắng nge ( Removing Event Listeners)
3. **Luồng( Streams)**
   1. Readable Streams
   2. Writable Streams
4. **File System**
   1. làm việc với tập tin hệ thống
   2. File thông tin
   3. xem các tập tin
5. **HTTP**

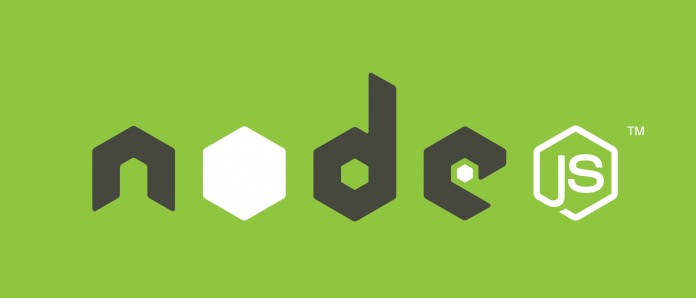
5.1 Class : http.Server

5.2 Class: http.ServerRequest

5.3 Class: http.ServerRespones

5.4 http.ClientResponse

I .Tổng Quan Về Node.JS



Hình 1. Ryan Lienhart Dahl & Node.js

**1.Node.Js Là Gì ?**

Nodejs được Ryan Lienhart Dahl giới thiệu vào năm 2009 cho phép lập trình JavaScript phía Server. Trước đây JavScript chỉ được coi là một ngôn ngữ đơn giản tạo hiệu ứng hoạt hình, tương tác phía trình duyệt. Cho đến khi Google Chrome sử dụng V8 engine , một bộ dịch , thực thi mã JavaScript nhanh nhất thể giới tính đến nay. V8 được viết bằng C/ C++ , mã nguồn mở, cho phép bên thứ 3 toàn quyền sử dụng , triển khai hệ điều hành khác nhau: Windows , Linux , Unix , MasOS . Node .js được xây dựng trên V8 và thư viện LibUV.

Định nghĩa Node.js bởi tài liệu chính thức như sau :

“ Node.js là một nền tảng dựa vào Chrome JavaScript runtime để xây dựng các ứng dụng nhanh , có độ lớn. Node.js sử dụng các phát sinh các sự kiện ( event-driven) , mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán ’’

Node.js là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía server và các ứng dụng liên quan đến mạng. ứng dụng node.js được viết bằng JavaScript và có thể chạy trong môi trường node.js trên hệ điều hành Windows, Linux …

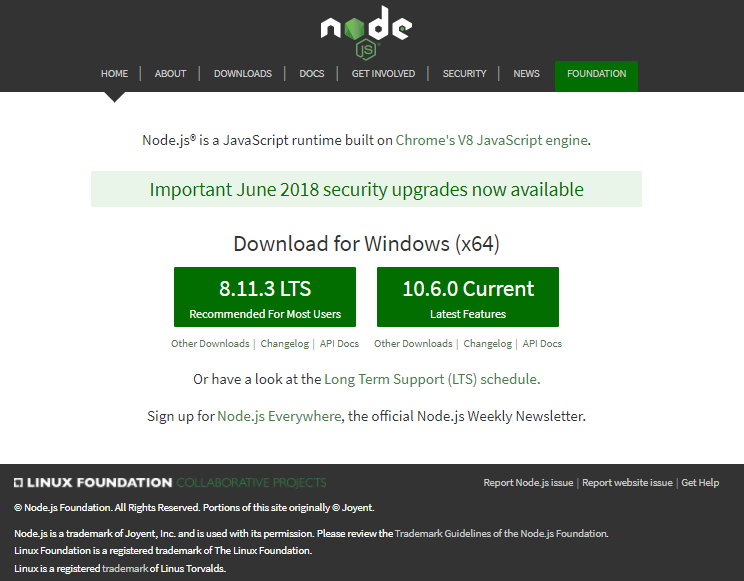
Node.js cũng cung cấp cho chúng ta các module JavaScript đa dạng, có thể đơn giản hóa, sự phát triển các ứng dụng web.

Node.js = Môi trường runtime + Các thư viện JavaScript

Nodejs phát triển nhanh đến chóng mặt. Đến nay các website lớn trên thế giới đã sử dụng Node.js gồm Microsoft , LinkedIn, Twitter, Walmart, Paypal…

Node.js liên tục được InfoWorld bình chọn là “ Technology Of the Year” từ năm 2012 đến nay.

Trang chủ : <https://nodejs.org/en/>



Phiên bản mới nhất tính đến ngày 07/8/ 2018 là v10.6.0 .Phiên bản ổn định nhất của Node.js hiện tại là :v8.11.3

**2.Đặc điểm của Node.Js**

Dưới đây là vài đặc điểm quan trọng biến Node.js thành sự lựa chọn hàng đầu trong phát triển phần mền :

**Không đồng bộ và phát sinh sự kiện ( Event Driven):** Tất cả các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dữ liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về sự kiện của Node.js giúp Server nhận được phản hồi từ các API gọi trước đó.

**Chạy rất nhanh:** Dựa trên V8 JavaScript Engine của Google Chrome , thư viện Node.js rất nhanh trong quá trình thực hiện code.

**Các tiến trình đơn giản nhưng hiệu năng cao :** Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi một cách không khóa và tạo cho server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữa hạn để quản lý request . Node.js sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các server truyền thống như Apache HTTP Server.

**Không đêm :** ứng dụng Node.js không lưa trữ các dữ liệu buffer

**Có giấy phép:** Node.js được phát hành dựa trên giấy phép MIT License.

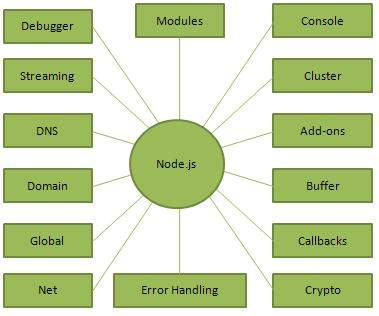
**3.Với Node.js Bạn phải làm tất cả.**

Node.js chỉ một môi trường - đều này có nghĩa bạn phải tự làm mọi thứ. Sẽ chảng có bất kì máy chủ mặc định nào cả !!!. Một đoạn script xử lý tất cả các kết nối với Client .Diều này làm giảm đáng kể số lượng tài nguyên được sử dụng ứng dụng.

**4.Ai sử dụng node.js**

<https://goo.gl/IZn6jX> chứa các danh sách các dự án , ứng dụng và các công ty sử dụng Node.js .Trong danh sách này bao gồm : eBay, GE, GoDaddy, Microsoft, Paypal, Uber..

**5.Các Thành phần quan trọng của Node.js**



5.1Luồng ( Threading)

Node.js hoạt động trên một luồng duy nhất , sử dụng các lệnh gọi Non-blocking I/O cho phép hộ trỡ hàng chục nghìn kết nối đồng thời. Thiết kế chia sẻ một luồng duy nhất giữa tất cả các yêu cầu sử dụng mô hình quan sát được sử dụng để xây dựng ứng dụng đồng thời, trong đó bất kì chức năng nào thực hiện I/O phải sử dụng một callback. Để thức ứng với vòng lặp sự kiện đơn luồng. Node.js dùng thư viện libuv để sử dụng một threadpool có kích thước cố định. Nó chịu trách nhiệm cho một số hoạt động I/O không đồng bộ không bị ngăn chặn. Một nhược điểm của cách tiếp cận đơn luồng này là Node.js không cho phép nhân rộng theo chiều dọc bằng cách tăng số lỗi CPU của máy nó đang cahyj mà không cần sử dụng một module bổ sung. Chẳng hạn như cluster, StrongLoop Process Mannager hay pm2 . Tuy nhiên , các nhà phát triển có thể tăng số lượng mặc định của các luồng trong Threadpool libuv . các luồng này có thể được phân phối qua nhiều lõi bởi hệ điều hành máy chủ. Thực hiện các tác vụ song song trong Node.js được xử lý bằng một threadpool .Luồng chính gọi các function gửi các nhiệm vụ cho hàng đợi công việc mà luồng trong threadpool kéo và thực hiện. Các chức năng hệ thống không chặn non-blocking như việc chuyển đổi mạng sang các socket non-blocking phía kernel , trong khi đó các chức năng hệ thống như I/O chạy theo luồng riêng của nó. Khi một luồng trong threadpool hoàn thành một nhiệm vụ, nó sẽ thông báo cho thread chủ chốt của việc này lần lượt đánh thức và thực hiện callback đã đăng kí. Vì Callback được xử lý nối tiếp trên luồng chính, các phép tính kéo dài và nhiệm vụ khác rằng buộc với CPU sẽ đóng băng toàn bộ vòng lặp sự kiện cho đến khi hoàn thành.

**5.2 V8**

V8 là công cụ xử lý JavaScript được xây dụng cho Google Chrome và có mã nguồn mở từ google trong năm 2008. Viết bằng C++ , V8 biên dịch mã nguồn JavaScript thành mã máy thay vì interprete nó trong thời gian thực .Node.js sử dụng libuv để xử lý sự kiện không đồng bộ . Libuv là một lớp trừa tượng cho các chức năng hệ thống mạng và tệp tin trên cả Windows và các hệ thống dựa tren POSIX như linux ,MacOS, OSS trên NonStop Và Unix. Chức năng chính của Node.js nằm trong thư viện JavaScript . Các rằng buộc Node.js được viết bằng C++ , kết nối các công nghệ này với nhau và với hệ điều hành.

* 1. **Quản lý Package**

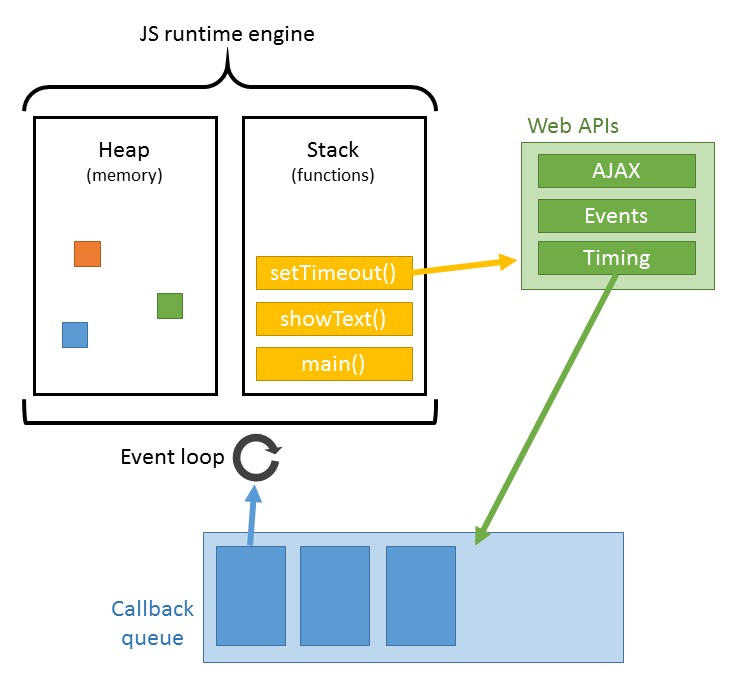
Nmp là trình quản lý gói cài đặt sẵn cho nền tảng máy chủ Node.js . Nó được sử dụng để cài đặt các chương trình Node.js từ đăng kí nmp, tổ chức cài đặt và quản lý các chương trình Node.js của bên thứ ba. NPM không được nhầm lẫn với câu lệnh yêu cầu CommonJS (). Nó không được sử dụng để nạp mã. Thay vào đó , nó được sử dụng để cài đặt mã và quản lý mã phụ thuộc từ dòng lệnh. Các gói được tìm thấy trong đăng kí nmp có thể bao gồm từ các thư viện trợ giúp đơn giản như Underscore.js cho các chương trình chạy như Grunt.

* 1. Unified API

Node.js có thể được kết hợp với một trình duyệt , một cơ sở dữ liệu hộ trợ dữ liệu JSON ( như Postgres, MongoDB hoặc CouchDB) và JSON cho một ngăn xếp phát triển thống nhất của JavaScript . Với sự thích nghi của các mô hình phát triển phía máy chủ như MVC, MVP, MVVM,… Node.js cho phép tái sử dụng cùng mô hình và giao diện dịch vụ giữa máy khách và phía máy chủ.

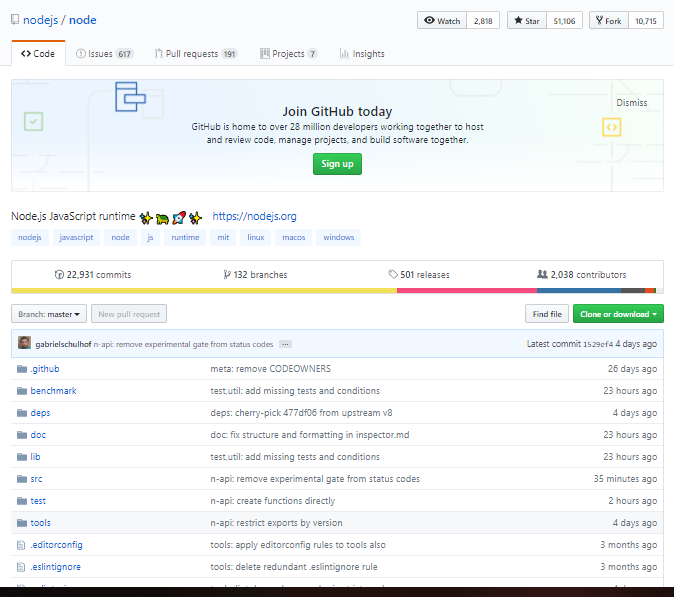
**5.5 Vòng lặp sự kiện( Event Loop)**

Node.js đăng kí chính nó với hệ điều hành để nó được thông báo khi một kết nối được thực hiện , và hệ điều hành sẽ phát hành một Callback Trong thời gian chạy Node.js mỗi kết nối là một sự phân bổ heap nhỏ . Thông thường, các tiến trình hoặc luồng tương đối nặng của hệ điều hành xử lý mỗi kết nối. Node.js sử dụng vòng lặp sự kiện cho khả năng mở rộng , thay vì các tiến trình hoặc các luồng. Trái ngược với các máy chủ điều khiển sự kiện khác, vòng lặp sự kiện của Node.js không cần phải được gọi rõ ràng. Thay vào đó, gọi lại được xác đinh, và máy chủ tự động nhập vào các vòng lặp sự kiện vào cuối định nghĩa gọi lại. Node.js thoát khỏi vòng lặp sự kiện khi không có callback khác được gọi thực hiện



**6. Cộng Đồng Phát Triển**

Địa chỉ mã nguồn mở cho dự án Node.js <https://github.com/nodejs/node>



Cộng đồng phát triển Node.js chủ yếu tập trung ở hai nhóm google : Node.js Và Node.js-dev, một kênh IRC là #node.js trên mạng Freenode.Có một hội thảo về Node.js cà NodeConf được tổ chức thường xuyên.

Hiện tại , cộng động Node.js việt nam hoạt động tại <https://nodejs.vn/>

**II. Ưa Điểm Và Nhược Điểm**

1. **Ưa Điểm**

Đặc điểm nổi bật của Node.js là nó nhận và xử lý nhiều kết nối chỉ với một single-thread. Điều này giúp hệ thống tốn ít RAM nhất và chạy nhanh nhất khi không phải tạo thread mới cho mỗi truy vấn giống PHP. Ngoài ra, tận dụng ưu điểm non-blocking I/O của Javascript mà Node.js tận dụng tối đa tài nguyên của server mà không tạo ra độ trễ như PHP

JSON APIs Với cơ chế event-driven, non-blocking I/O(Input/Output) và mô hình kết hợp với Javascript là sự lựa chọn tuyệt vời cho các dịch vụ Webs làm bằng JSON.

Ứng dụng trên 1 trang( Single page Application) Nếu bạn định viết 1 ứng dụng thể hiện trên 1 trang (Gmail?) NodeJS rất phù hợp để làm. Với khả năng xử lý nhiều Request/s đồng thời thời gian phản hồi nhanh. Các ứng dụng bạn định viết không muốn nó tải lại trang, gồm rất nhiều request từ người dùng cần sự hoạt động nhanh để thể hiện sự chuyên nghiệp thì NodeJS sẽ là sự lựa chọn của bạn.

Shelling tools unix NodeJS sẽ tận dụng tối đa Unix để hoạt động. Tức là NodeJS có thể xử lý hàng nghìn Process và trả ra 1 luồng khiến cho hiệu xuất hoạt động đạt mức tối đa nhất và tuyệt vời nhất.

Streamming Data (Luồng dữ liệu) Các web thông thường gửi HTTP request và nhận phản hồi lại (Luồng dữ liệu). Giả xử sẽ cần xử lý 1 luồng giữ liệu cực lớn, NodeJS sẽ xây dựng các Proxy phân vùng các luồng dữ liệu để đảm bảo tối đa hoạt động cho các luồng dữ liệu khác.

Ứng dụng Web thời gian thực Với sự ra đời của các ứng dụng di động & HTML 5 nên Node.js rất hiệu quả khi xây dựng những ứng dụng thời gian thực (real-time applications) như ứng dụng chat, các dịch vụ mạng xã hội như Facebook, Twitter,…

**III. Installtion**

1.Installing Node.js

Hiển nhiên là các bạn phải học cách cài đặt node trước khi muốn viết và chạy bất cứ ứng dụng nào trên nền node. Cài đặt node thì rất đơn giản, bạn là người sử dụng window hay linux thì trên website Node.js đều có những bộ cài đặt tương ứng , bạn chỉ cần download về cài đặt như bình thường.

Với Linux thì bạn sử dụng package Manager , bật cửa sổ terminal và Type:

Sudo apt-get update

Sude apt-get install node

Hoặc:

Sudo aptiude update

Sudo aptitude install node

Bạn chỉ cần thêm Node.js vào danh sách mã nguồn bằng lệnh sau.

Sudo echo deb http: //fpt.us.debian.org/debian/ sid main > /etc/apt/sources.list.d/sid.list

Cần trọng khi cài sid packgare trên những hệ thống cũ hơn có thể làm hệ thống của bạn ảnh hưởng , hãy cận thận và remove /etc/apt/source.list.d/sid.list sau khi bạn cài xong Node.

2.Installing New Modules

Node.js có một ứng ựng quản lý packgare , đó là Node Packgate Manager (NMP) .ứng dụng này tự động được cài đặt khi bạn cài Node.js và bạn dùng NPM để cài đặt các module khác. Đề cài đặt một module , bạn mở của sổ command line của node.js ra , vào đường dẫn tường ứng và nhập lệnh.

Nmp install module\_name

Không phụ thuộc vào hệ điều hành bạn dùng , lệnh trên sẽ cài module mà bạn mong muốn chỉ định.

**III . Các Object Cơ Bản**

**1.Global Objects (Đối tượng toàn cục)**

Như chúng ta đã biết, hệ thống mô-dun của node không khuyến khích việc sự dụng