**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH- ĐẠI HỌC BÁCH KHOA**

**KHOA KHỌC HỌC VÀ KĨ THUẬT MÁY TÍNH**

**BỘ MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**


# BÁO CÁO BÀI TẬP LỚN

**ĐỀ TÀI: NGHIÊN CỨU NOSQL**

**Giáo viên hướng dẫn:** Võ Thị Ngọc Châu

**Thành viên:**

- Nguyễn Văn Quân 50902143
- Phạm Minh Thành 50902476

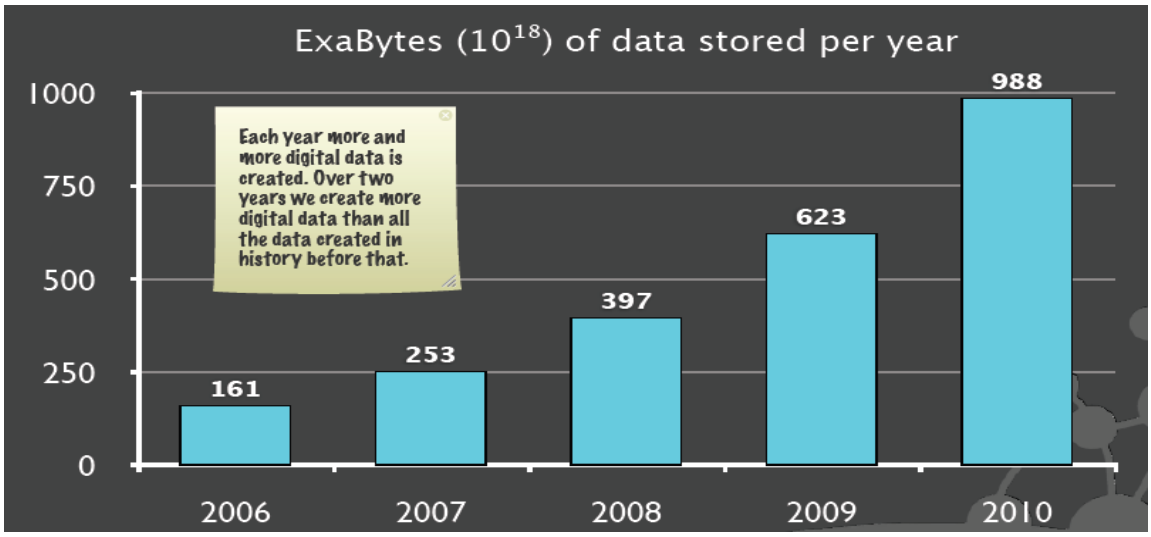
# Mục lục

# I. NoSQL là gì?

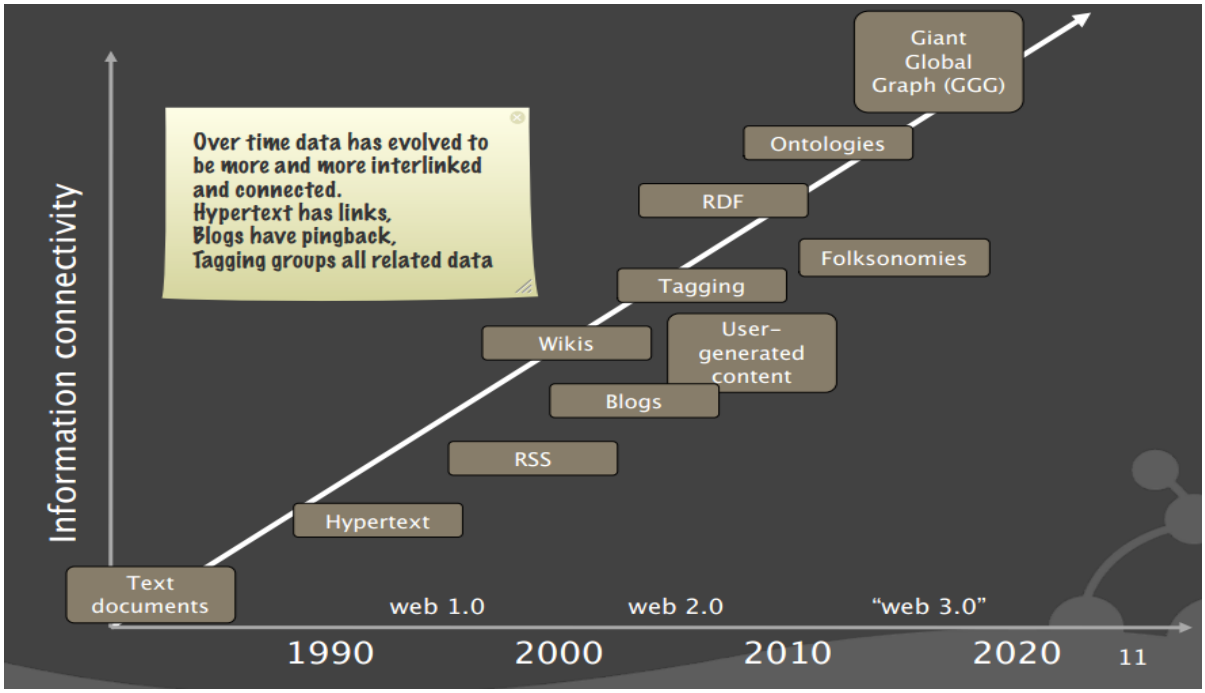
## 1. Xu hướng của dữ liệu.

### a. Khả năng lưu trữ dữ liệu cực lớn.

Dữ liệu số được tạo ra ngày càng nhiều. Trong hai năm (2009-2010) chúng ta đã tạo ra lượng dữ liệu số nhiều hơn tất cả các năm trước cộng lại.




### b. Sự liên kết dữ liệu chặt chẽ.

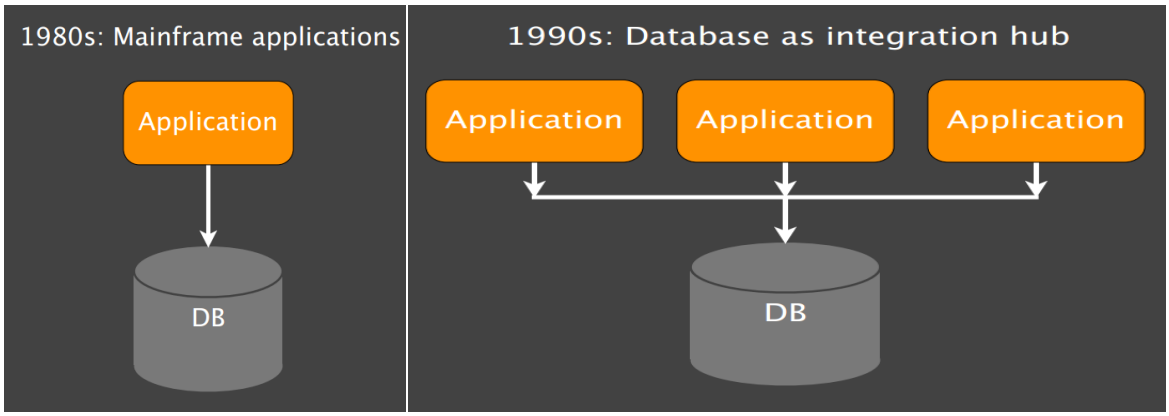


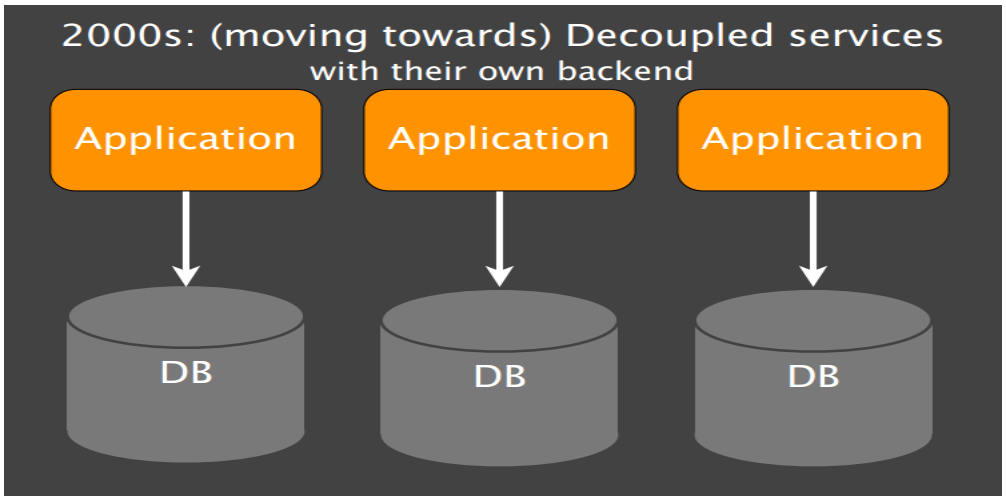
Qua thời gian, dữ liệu ngày càng liên kết chặt chẽ với nhau. Các đường dẫn web liên kết các trang web, các trang nhật ký mạng liên kết với nhau và các nhãn nhóm các thông tin có liên quan lại.

c. Bán cấu trúc

Ngày nay, dữ liệu không còn đơn giản có thể thể hiện qua các hàng và các cột mà nó bao gồm nhiều thành phần dữ liệu, và các thành phần dữ liệu đó được lồng vào nhau. Từ đó, người ta đã nghĩ ra cách lưu dữ liệu theo kiểu bán cấu trúc. Đây cũng là một thành phần quan trọng trong thời đại web 2.0.

d. Kiến trúc dữ liệu




Trong những năm đầu của sự bùng nổ dữ liệu (1980), các ứng dụng thường sử dụng cơ sở dữ liệu với sự chuyên môn: một cơ sở dữ liệu dành riêng cho một ứng dụng cụ thể.

Vào những năm 1990 thì xu hướng tận dụng cơ sở dữ liệu để tạo nên nhiều ứng dụng lên ngôi, đặt ra yêu cầu lớn cho hệ quản trị cơ sở dữ liệu.

Đến những năm đầu thế kỉ 21, thì sự chuyên môn hoá dữ liệu cho từng ứng dụng với phương thức tách riêng dịch vụ cho từng phần của ứng dụng mở ra một thời kì mới, thời kì phân tán dữ liệu và xử lý song song.

## 2. NoSQL là gì

### a. Khái niệm

NoSQL là khái niệm dùng để chỉ chung một tập các hệ cơ sở dữ liệu khác với hệ quản trị cơ sở dữ liệu quan hệ truyền thống-hệ quản trị cơ sở dữ liệu quan hệ được truy xuất bằng SQL. Vì vậy, NOSQL ngụ ý một hệ quản trị cơ sở dữ liệu không dùng SQL, hay là không dùng mô hình quan hệ.

Hệ cơ sở dữ liệu NoSQL, còn gọi là Not Only SQL, là một cách tiếp cận mới về thiết kế và quản trị dữ liệu phù hợp với những tập dữ liệu phân tán có kích thước lớn.

### b. Những thuộc tính đặc trưng của hệ cơ sở dữ liệu NOSQL

- Dữ liệu phân tán, có thể mở rộng theo chiều ngang và có thể quản lý một lượng lớn dữ liệu lên đến hàng petabyte với độ trễ thấp.
- Mềm dẻo hơn so với mô hình dữ liệu quan hệ.
- Sự đảm bảo giao tác yếu hơn mô hình dữ liệu quan hệ.
- Không hỗ trợ SQL.
- Thường lưu dữ liệu theo dạng cặp khóa – giá trị.

Trong một hệ cơ sở dữ liệu NOSQL

- Không có mô hình dữ liệu tĩnh.
- Không hỗ trợ phép kết.
- Phân tán ở mức cao.
- Có thể mở rộng dễ dàng.
- Nhất quán cuối.

### c. Ưu điểm/nhược điểm của NoSQL

- *Ưu điểm*

- Khả năng mở rộng.

- Tính sẵn sang.
- Chi phí thấp.
- Tính co giãn có thể đoán trước.
- Dữ liệu thưa và bán cấu trúc.

- *Nhược điểm*

- Khả năng truy vấn bị giới hạn.
- Tính nhất quán cuối không trực quan cho việc lập trình làm cho chương trình khách phức tạp.
- Không có sự tiêu chuẩn hóa.
- Khó khăn trong việc kiểm soát truy cập.

*d. Các hệ thống NoSQL phổ biến*

- Google
- Amazon
- Twistter
- Facebook
- MongoDB
- Basho
- Hadoop
- LinkedIn
- …

*e. Các opensource về NoSQL*

- Neo4j – http://neo4j.org
- CouchDB – http://couchdb.apache.org
- Cassandra – http://cassandra.apache.org
- Hadoop + HBase – http://hadoop.apache.org
- MongoDB http://www.mongodb.org
- Redis – http://code.google.com/p/redis
- Oracle Berkley DB – http://www.oracle.com/database/berkeley-db
- FlockDB – http://github.com/twitter/flockdb
- …

## 3. So sánh NoSQL và hệ cơ sở dữ liệu quan hệ

Các hệ cơ sở dữ liệu quan hệ (RDBMs-Relational database management systems) hiện tại bộc lộ những yếu kém trong những tác vụ như đánh chỉ mục một lượng lớn dữ liệu, phân trang, hoặc phân phối luồng dữ liệu media (phim, ảnh, nhạc...). Cơ sở dữ liệu quan hệ được thiết kế cho những mô hình dữ liệu không quá lớn trong khi các dịch vụ mạng xã hội lại có một lượng dữ liệu cực lớn và cập nhật liên tục do số lượng người dùng quá nhiều.

Thế hệ cơ sở dữ liệu mới - NoSQL - giảm thiểu tối đa các phép tính toán, tác vụ đọc-ghi liên quan kết hợp với xử lý theo lô (batch processing) đảm bảo được yêu cầu xử lý dữ liệu của các dịch vụ mạng xã hội. Hệ cơ sở dữ liệu này có thể lưu trữ, xử lý từ lượng rất nhỏ đến hàng petabytes dữ liệu với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp.

NoSQL thiết kế đơn giản, nhẹ, gọn hơn so với các hệ cơ sở dữ liệu quan hệ. Ngoài memory cached, dữ liệu nhỏ,… các cơ sở dữ liệu dạng NoSQL đặc biệt thích hợp cho thiết bị cầm tay nơi mà bộ nhớ và tốc độ xử lý hạn chế hơn so với máy tính thông thường. Khi khối lượng dữ liệu cần lưu trữ và lượng vào/ra cực lớn, cơ sở dữ liệu quan hệ đòi hỏi khắt khe và cao về phần cứng, chi phí thiết lập, vận hành đắt thì các mô hình lưu trữ phân tán trong NoSQL trở nên vượt trội. Thiết kế đặc biệt tối ưu về hiệu suất, tác vụ đọc-ghi, ít đòi hỏi về phần cứng mạnh và đồng nhất, dễ dàng thêm bớt các node không ảnh hưởng tới toàn hệ thống,…

Các mô hình dữ liệu đặc thù của N oSQL cũng cấp API(Application programming interface-giao diện lập trình ứng dụng) tự nhiên hơn so với việc dùng cơ sở dữ liệu quan hệ.

Những ràng buộc về giấy phép, bản quyền sử dụng cùng với một khoản phí không nhỏ khi sử dụng cơ sở dữ liệu quan hệ cũng là một ưu thế của NoSQL. Chấp nhận NoSQL đồng nghĩa với việc bạn tham gia vào thế giới nguồn mở nơi mà bạn có khả năng tùy biến mạnh mẽ các sản phẩm, thư viện theo đúng mục đích của mình.

Bảng dưới đây đưa ra một số so sánh giữa RDBM và NoSQL.

| **Tính năng** | **CSDL quan hệ** | **NoSQL** |
|---|---|---|
| Hiệu suất | - Kém hơn<br>- SQL<br>- Relational giữa các table | - Cực tốt<br>- Bỏ qua SQL<br>- Bỏ qua các ràng |

| | | |
|---|---|---|
| | | buộc dữ liệu |
| Khả năng mở rộng | - Hạn chế về lượng. | - Hỗ trợ một lượng rất lớn các node. |
| Hiệu suất đọc-ghi | - Kém do thiết kế để đảm bảo sự vào/ra liên tục của dữ liệu | - Tốt với mô hình xử lý theo lô và những tối ưu về đọc-ghi dữ liệu. |
| Thay đổi số node trong hệ thống | - Phải shutdown cả hệ thống.<br>- Việc thay đổi số node phức tạp. | - Không cần phải shutdown cả hệ thống.<br>- Việc thay đổi số node đơn giản, không ảnh hưởng đến hệ thống. |
| Phần cứng | - Đòi hỏi cao về phần cứng. | - Đòi hỏi thấp hơn về giá trị và tính đồng nhất của phần cứng |

## 4. NoSQL được dùng khi nào?

### a. Khi nào không nên dùng NoSQL?

- Tất cả dữ liệu được lưu đủ tại một máy và không cần phải phân tán.
- Bạn đang dùng xử lý chuyển giao dữ liệu trực tuyến và cần đảm bảo toàn vẹn dữ liệu cũng như các thuộc tính giao tác ACID mà các hệ cơ sở dữ liệu quan hệ làm rất tốt.
- Chương trình có các thực thể với mối quan hệ phức tạp với nhau.
- Việc phân tách dữ liệu ra khỏi chương trình là quan trọng.

### b. Khi nào nên dùng NoSQL?

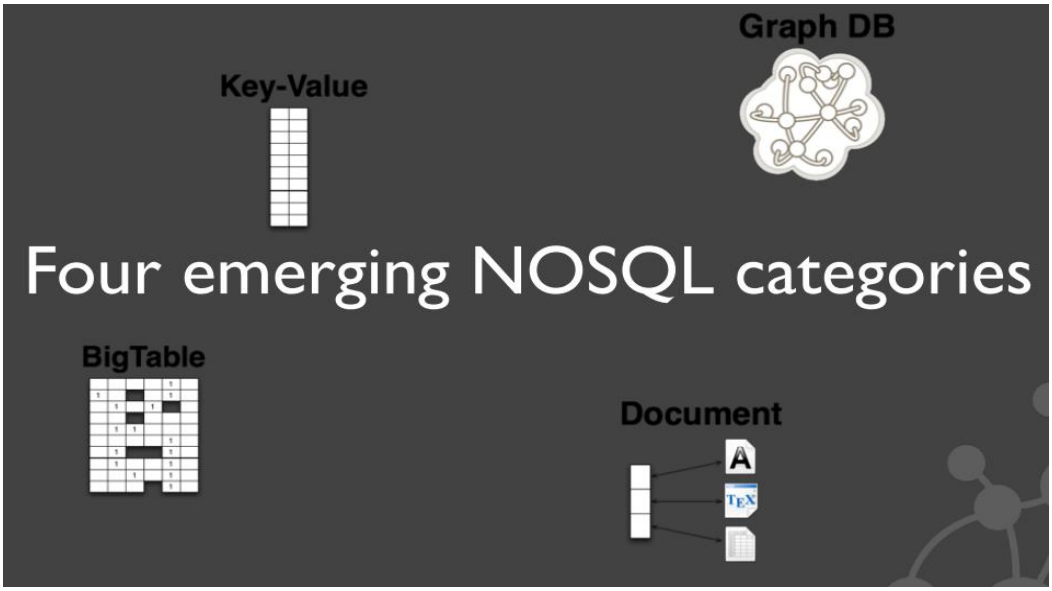
- Dữ liệu trở nên lớn và không thể quản lý tại một máy được nữa.
- Hệ cơ sở dữ liệu quan hệ không còn quản lý nổi việc trao đổi dữ liệu nữa.
- Chương trình cần ghi với hiệu suất rất cao và đọc với độ trễ thấp.
- Dữ liệu không đòi hỏi tính cấu trúc quá cao.
- Có thể chấp nhận một vài sự không nhất quán dữ liệu.

## 5. Vì sao nên chọn NoSQL?

- Khả năng mở rộng cao.
- Tính sẵn sàng cao.

- Chi phí thấp.
- Dữ liệu có cấu trúc linh hoạt.
- Triển khai dễ dàng và linh hoạt.

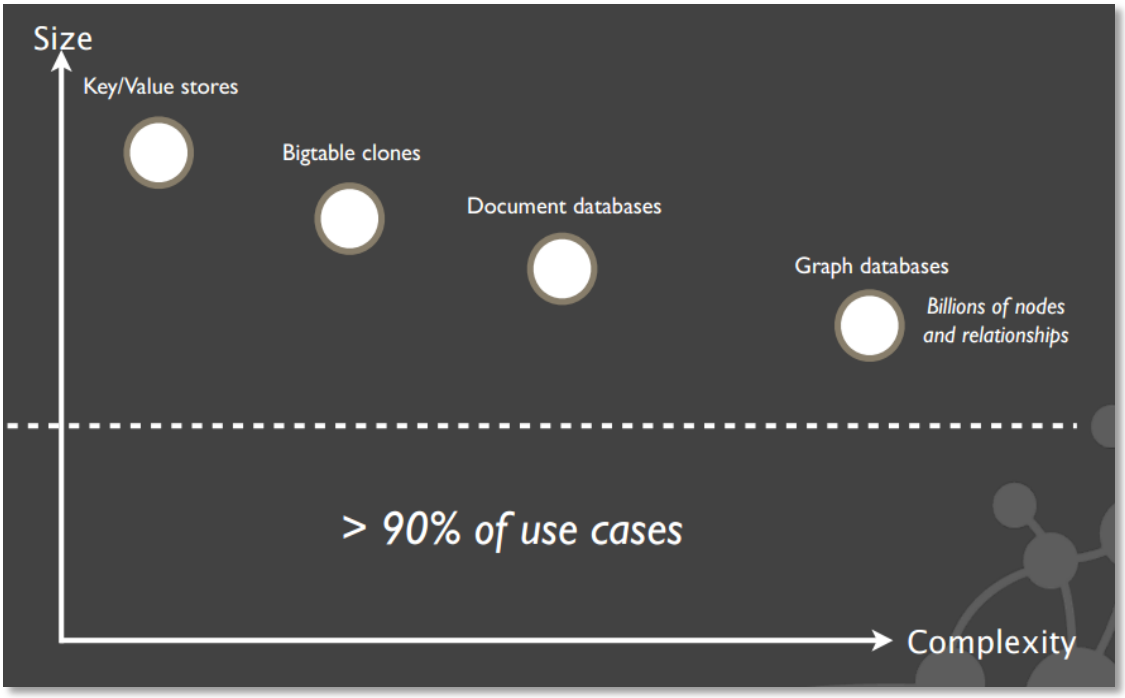
## II. Phân loại




NoSQL được phân chia thành bốn lại cơ bản:

- Lưu trữ dạng cặp khóa - giá trị (Key-Value).
- Lưu trữ theo nhóm cột (BigTable).
- Cơ sở dữ liệu hướng tài liệu (Document).
- Cơ sở dữ liệu biểu đồ (Graph DB).

Độ phức tạp và dung lượng tiêu tốn của mỗi loại là khác nhau (như bảng dưới), mặc dù sự khác nhau này không nhiều (khoảng 10%).




## 1. Lưu trữ dạng cặp khóa - giá trị

Hỗ trợ máy client đọc ghi dữ liệu bằng cách dùng khóa. Một ví dụ dùng dạng lưu trữ này là Amazon's Dynamo. Ý tưởng chính là một bảng băm gồm khóa và con trỏ tới từng phần tử dữ liệu. Các ánh xạ thường đi kèm với cơ chế bộ nhớ cache để tối đa hóa hiệu suất.

*get(key): trả về một đối tượng hoặc một danh sách đối tượng.*

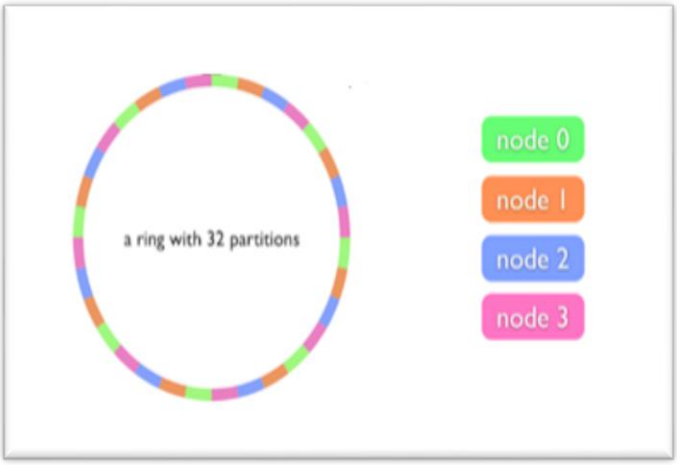
*put(key, object): lưu trữ dữ liệu*

Dynamo dùng cơ chế băm để phân chia dữ liệu trên các máy chủ khác nhau. Để đảm bảo tính sẵn sàng cao, từng thao tác ghi được nhân rộng trên một số máy chủ. Các máy chủ đều cùng cấp và không có máy quản lý. Ưu điểm của Dynamo là mô hình khóa - giá trị rất đơn giản và cho hiệu suất ghi cao.

Ví dụ: Riak

- Không xảy ra việc một điểm hỏng dẫn đến cả hệ thống hỏng.
- Không có máy nào là trung tâm.
- Xử dụng truy vấn MapReduce (bằng ngôn ngữ Erlang hoặc Javascript)
- HTTP/JSON API

- Các nốt liên kết dạng vòng và tự sao chép
- Các nốt co dãn và tự cân bằng
- Hỗ trợ các ngôn ngữ: Erlang, C, Javascript
- Phát triển bởi: Basho Technologies
- Hỗ trợ java: riak-java-client




Mô hình dữ liệu

- Các cặp khóa - giá trị được lưu trữ trong một thùng chứa.
- Một thùng chứa tương đương với một không gian tên.
- Quản lý phiên bản.
- Mỗi cập nhật được quản lý bởi giải thuật "vector clock".
- Giải thuật quy định thứ tự và phát hiện tương tranh.
- Khi có tương tranh xảy ra, có thể quy định: Đến sau thắng hoặc xử lý thủ công.

Ví dụ: REST API

Đọc một đối tượng

*GET /riak/bucket/key*

Tạo và lưu một đối tượng mới

*POST /riak/bucket*

Cập nhật một đối tượng đã có

*PUT /riak/bucket/key*

## 2. Lưu trữ theo nhóm cột

Các hệ cơ sở dữ liệu NoSQL dạng này được tạo ra để lưu trữ và xử lý lượng rất lớn dữ liệu phân tán trên nhiều máy. Các khóa vẫn được sử dụng và mỗi khóa trỏ đến nhiều cột. Với BigTable (Mô hình dữ liệu lưu trữ theo nhóm cột của Google), các dòng được xác định bởi một khóa dòng với dữ liệu đã được sắp xếp. Các cột được sắp xếp theo nhóm.

Các tác vụ đọc, ghi được thực hiện theo cột thay vì theo dòng. Đại diện nổi tiếng nhất của hệ cơ sở dữ liệu loại này là BigTable của Google. Ngoài ra, còn có HBase và Cassandra, vốn lấy cảm hứng từ BigTable. Theo tài liệu của hãng, BigTable là một ánh xạ thưa, đa chiều, ổn định, phân tán và đã được sắp xếp:

- Thưa - một số ô có thể trống
- Phân tán - dữ liệu được phân tán trên nhiều máy chủ
- Ổn định - được lưu trữ vào bộ nhớ ngoài
- Đa chiều - có nhiều hơn một chiều
- Ánh xạ - khóa và giá trị
- Sắp xếp - các ánh xạ thường không được sắp xếp, nhưng trường hợp này thì có

Ví dụ sau đây giúp ta dễ hình dung về BigTable:

```
{
row1:{
    user:{
          name: john
          id : 123
    },
    post: {
          title:This is a post
          text : xyxyxyxx
    }
}
row2:{
    user:{
          name: joe
          id : 124
    },
    post: {
          title:This is a post
          text : xyxyxyxx
    }
}
row3:{
    user:{
          name: jill
          id : 125
    },
    post: {
          title:This is a post
          text : xyxyxyxx
    }
}

}
```

Các khóa ngoài cùng (rows1, rows2, rows3) tương đương với các dòng. User và post là các nhóm cột. Nhóm cột user có các cột là name và id. Nhóm cột post có các cột title và text.

## 3. Cơ sở dữ liệu hướng tài liệu

Tương tự như kiểu lưu trữ dạng cặp khóa - giá trị, mô hình dữ liệu cơ bản là một tài liệu được đánh số phiên bản và chứa các cặp khóa - giá trị. Các văn bản bán cấu trúc được lưu ở dạng tương tự JSON (JavaScript Object Notation).

Các cặp khóa - giá trị chứa thông tin về dữ liệu được đóng gói như một tài liệu. Apache CouchDB là một ví dụ thuộc loại này. Trong CouchDB, tài liệu có nhiều trường. Mỗi trường có một khóa và giá trị. Một tài liệu có thể có dạng như sau:

```
"firstname " : " John ",
"lastname " : "Doe" ,
"street " : "1 main st",
"city " : "New york"
```

Trong CouchDB, phân tán và sao lưu được thực hiện theo nguyên tắc P2P. Cổng giao tiếp của máy khách là RESTful HTTP, vốn đã được tích hợp tốt với giải pháp cân bằng tải của HTTP hiện tại.

Ví dụ: một bài viết blog dùng MongoDB

```
db.posts.save({
    _id    : 3,
    author:"john",
    title  : "Apples, Oranges and NOSQL",
    text   :  "This article will…",
    tags   : [ "database", "nosql" ]
});
```

MongoDB hỗ trợ chỉ mục thứ cấp, chỉ mục cụm và tối ưu truy vấn.

```
db.posts.ensureIndex({ tags: 1 });
db.posts.ensureIndex({ author: 1});

db.posts.find({ author: "john", tags: "nosql" });

// Result:
{
        "_id"    : 3,
        "author" : "john",
        "title"  : "Apples, Oranges and NOSQL",
        "text"   : "This article will…",
        "tags"   : ["database", "nosql", "mongodb" ]
}
```

Cập nhật các bình luận vào bài viết:

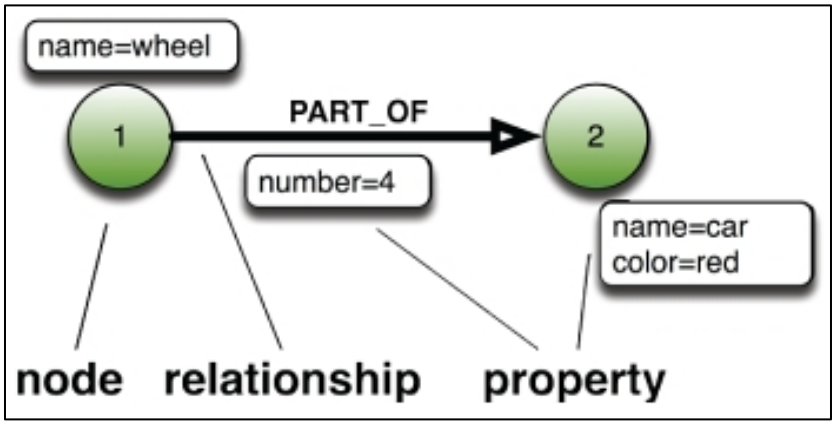
```
db.posts.update({ _id: 3 }, {
    $inc: { comments_count: 4},
    $pushAll : {
        comments: [
            { text: "Comment 1" },
            { text: "Comment 2", author: "Mr. T" },
            { text: "Comment 3" },
            { text: "Comment 4" }
        ]
    }
});
```

## 4. Graph Databases

Được xây dựng với

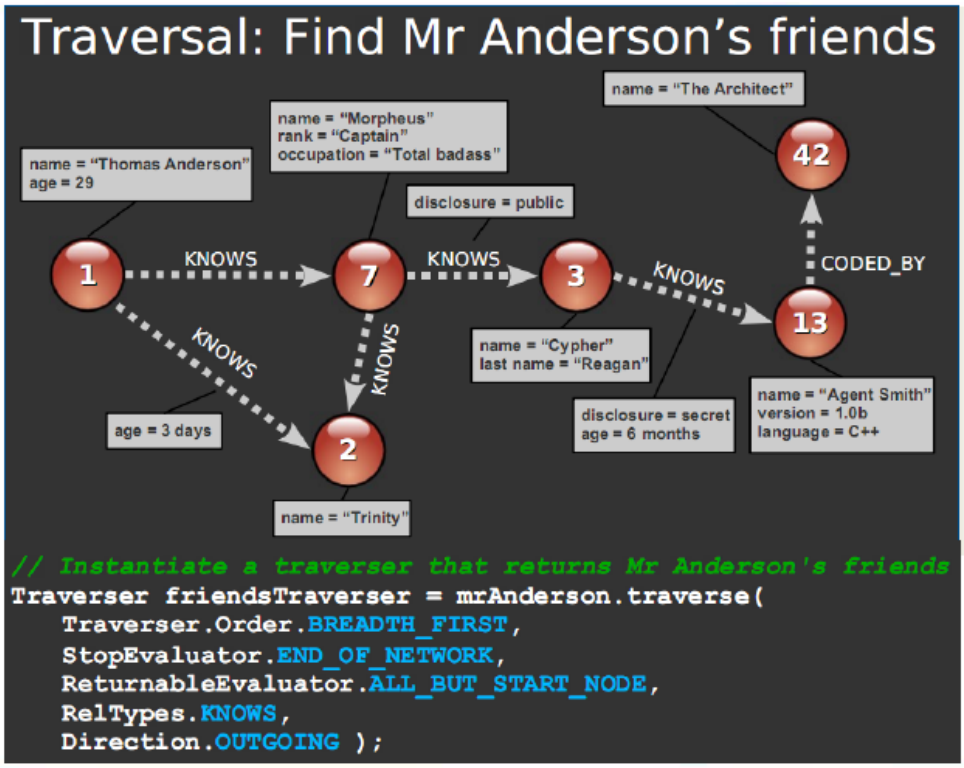
+ các node
+ quan hệ giữa các node
+ các thuộc tính của node

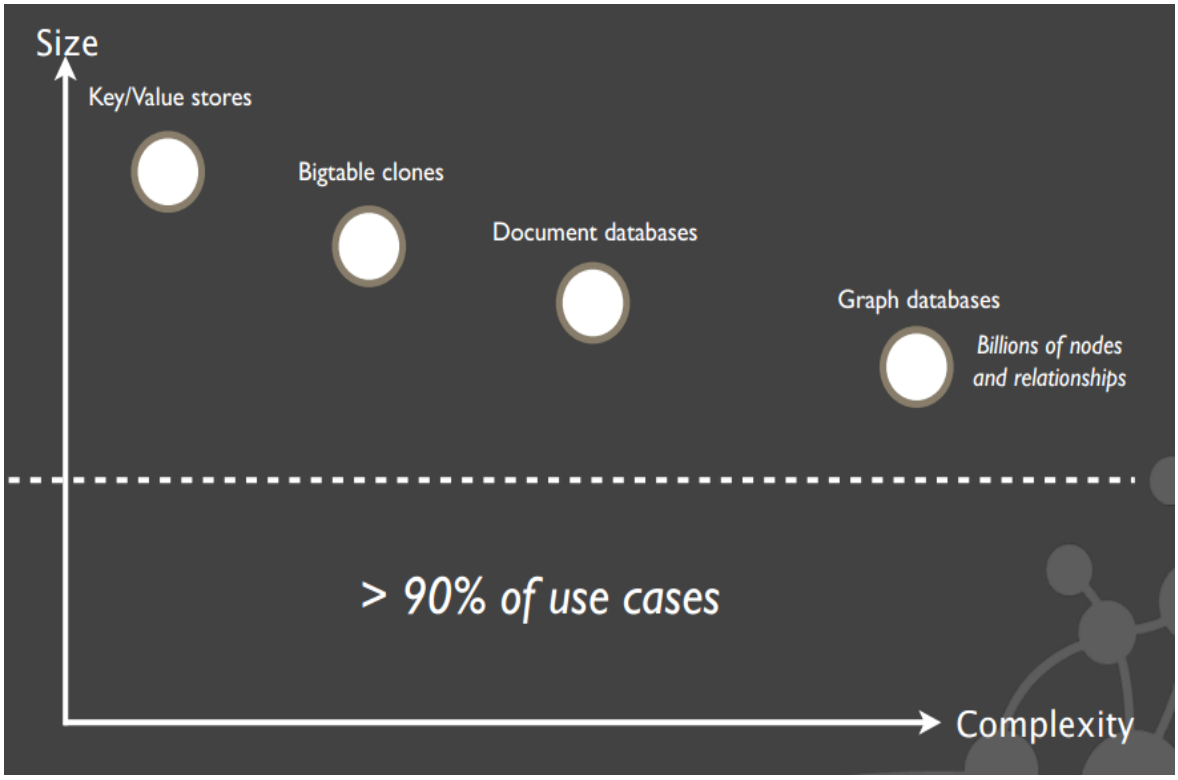
Thay vì sử dụng các bảng gồm nhiều hàng và cột - một cấu trúc cứng nhắc của SQL thì một mô hình đồ thị linh hoạt có thể được sử dụng với khả năng co dãn qua nhiều máy khác nhau.




Ví dụ: Neo4j

Neo4j là một cơ sở dữ liệu dạng graph database tuân thủ tất cả các tính chất ACID cho transaction. Neo4j được viết bằng Java. Dữ liệu được lưu ở đĩa như là một kiến trúc tối ưu cho mạng đồ thị.






## III. Demo

Sử dụng thư viện JDBM phiên bản 2.4 để hiện thực demo về NoSQL.

### 1. JDBM2

JDBM2 (Java Database Connectivity) là một thư viện cung cấp các kiểu dữ liệu như HashMap và TreeMap và lưu trữ chúng trên đĩa cứng. Sử dụng JDBM cho phép ta bảo quản và lưu trữ dữ liệu một cách dễ dàng và nhanh chóng theo mô hình NoSQL. JDBM2 là một phiên bản được dùng phổ biến của JDBM, rất nhẹ và khả năng chạy trên nhiều platform (file jar chỉ có dung lượng 145 KB).

JDBM2 được phát triển để hỗ trợ tính toán thiên văn khi mà dữ liệu quá lớn để có thể đưa vào bộ nhớ. Đồng thơi cũng cung cấp khả năng lưu trữ cho dự án thiên văn vũ trụ Asterope.

JDBM đã được phát triển 10 năm trước, được chuyển sang Java từ thư viện GDBM (viết bằng C). Phiên bản đầu tiên được phát triển bởi Cees De Groot, Alex Boisvert và một số developer khác. Phiên bản phổ biến 1.0 ra mắt vào 2005 và được sử dụng rộng rãi. JDBM2 là một phiên bản khác phát triển từ phiên bản 1.0 đó và được thêm nhiều tính năng cũng như tăng hiệu năng thực thi.

## 2. Demo

Để sử dụng demo này, cần add thư viện jdbm phiên bản 2.4 vào project.

### a. Test JDBM tree map.

Đầu tiên, ta cùng tìm hiểu về JDBM TreeMap. Trong file testJDBM.java, ta có dòng code như bên dưới sử dụng TreeMap.

Ta có thể nhận ra các function chính: put, remove trong tree map và, commit, rollback dành cho cơ sở dữ liệu.Chúng ta cần một RecordManager để quản lý treemap.

```
String fileName = "helloWorld";

RecordManager recMan = RecordManagerFactory.createRecordManager(fileName);

//Define simple tree:

String recordName = "firstTreeMap";

PrimaryTreeMap<Integer,String> treeMap = recMan.treeMap(recordName);

//Add some stuff to Tree

treeMap.put(1, "One");

treeMap.put(2, "Two");

treeMap.put(3, "Three");

recMan.commit();
```

Sau khi thực thi các dòng trên, dữ liệu chúng ta bây giờ bao gồm 3 cặp key-value, và tập key (các khoá) của tree sẽ là [1, 2, 3]

Ta thử bỏ cặp giá trị có khoá là 2 ra khỏi cây, giá trị của cây sau đó sẽ là [1, 3]:

```
treeMap.remove(2);
```

Ta cũng có thể rollback lại như các cơ sở dữ liệu khác, khi đó dữ liệu sẽ trở về trạng thái sau khi commit gần nhất.

```
recMan.rollback();
```

Khoá 2 đã được rollback, tập khoá của cây hiện tại là [1, 2, 3].

b. JDBM2 với cơ sở dữ liệu phức tạp hơn.

Trong file test.java, ta khởi tạo một cơ sở dữ liệu để lưu trữ thông tin khách hàng với 3 cột id, name, email. Class Customers định nghĩa các thông tin về một khách hàng (như một dòng). Class CustomersStorage định nghĩa cách thức chúng được lưu trên đĩa.

Trong test.java, chúng ta thử tạo 4 khách hàng và đưa chúng vào cơ sở dữ liệu. Chúng ta có thể tìm một khách hàng bằng id và hiển thị lên màn hình, cập nhật hoặc xoá chúng khỏi cơ sở dữ liệu giống như các chức năng của các cơ sở dữ liệu khác.

Sau khi chạy test.java, ta có thể thấy kết quả output:

*^^-Database has 4 record*

*Get all name in the DB*

*-->256:Nguyen Van A*

*-->257:Huynh Van B*

*-->258:Quan*

*-->505:Pham Minh Thanh*

*^^-Try to update DB:Quan->Nguyen Van Quan*

*Updated OK*

*Get all name in the DB*

*-->256:Nguyen Van A*

*-->257:Huynh Van B*

*-->258:Nguyen Van Quan*

*-->505:Pham Minh Thanh*

*^^-Try to remove from DB:Delete Nguyen Van Quan*

*Removed OK*

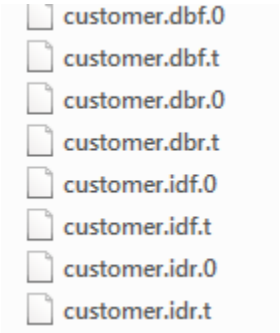
*Get all name in the DB:3*

*-->256:Nguyen Van A*

*-->257:Huynh Van B*

*-->505:Pham Minh Thanh*

Database file:




## IV. Tham khảo


+ NoSQL. http://en.wikipedia.org/wiki/NoSQL
+ What is NoSQL http://www.javacodegeeks.com/2011/11/what-is-nosql.html#ixzz1qVuRXktt
+ NoSQL Databases: Why, what and when. http://www.slideshare.net/quipo/nosql-databases-why-what-and-when
+ Apples, Oranges and NOSQL, Java World Cup 2010 seminar. http://www.slideshare.net/roialdaag/seminar2010nosql
+ NoSQL Databases, Marin Dimitrov, Tontotext - Semantic Technology Lab.
+ JDBM2 Embedded Key Value Java database. http://code.google.com/p/jdbm2
+ The World's Leading Graph Database. http://neo4j.org/learn/
+ NoSQL for dummies Tobias Ivarsson Hacker @ Neo Technology.
+ NoSQL phân tán không ràng buộc http://www.pcworld.com.vn/articles/cong-nghe/cong-nghe/2011/03/1222905/nosql-phan-tan-khong-rang-buoc/


--- THE END ---