สรุปโครงสร้างข้อมูลของ NoSQL Database  
วันนี้ได้เริ่มอ่านหนังสือ [Introducing to Data Science](https://www.manning.com/books/introducing-data-science)มีหนึ่งบททำการอธิบายเกี่ยวกับโครงสร้างข้อมูลของ NoSQL Database  
ไว้ได้อย่างน่าสนใจ และ เข้าใจง่าย  
จึงนำมาแปลไว้อ่านนิดหน่อยดังนี้

โครงสร้างข้อมูลของ NoSQL Database 4 ชนิด ประกอบไปด้วย

1. Column-Oriented
2. Key-value
3. Document
4. Graph

โดยแต่ละชนิดถูกคิดและสร้างขึ้นมาเพื่อแก้ไขปัญหาที่แตกต่างกันไป  
ซึ่งปัญหาเหล่านั้นไม่สามารถแก้ไขด้วยการใช้งาน RDBMS นั่นเอง

และพบว่า NoSQL หนึ่ง ๆ มักจะมีรูปแบบการจัดเก็บข้อมูลที่หลากหลาย  
หรือเรียกว่า **Multi-model database**  
ตัวอย่างที่เห็นได้ชัดคือ OrientDB  
ซึ่งมันคือ Graph database  
โดยที่แต่ละ node มันมีรูปแบบเป็น Document

แต่ก่อนที่จะไปดูโครงสร้างข้อมูลทั้ง 4 ของ NoSQL

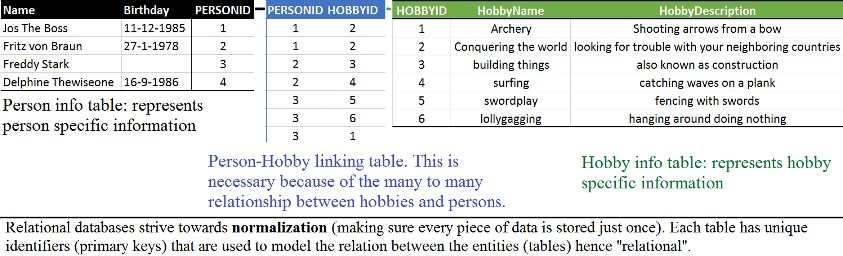
เราต้องมาดู RDBMS หรือ Relational Database ก่อนว่ามันเป็นอย่างไร ?  
เพื่อจะได้เห็นข้อแตกต่างได้ชัดเจนยิ่งขึ้น

**เราปฏิเสธไม่ได้ว่า RDBMS นั้นได้รับความนิยมอย่างสูง**  
สามารถรองรับความต้องการต่าง ๆ ได้อย่างมากมาย  
คำหนึ่งที่เรามักจะได้ยินบ่อย ๆ ในออกแบบโครงสร้างข้อมูลบน RDBMS คือ  
การ **Normalization** คือ เป็นวิธีการลดความซ้ำซ้อนของข้อมูลนั้นเอง  
มีโครงสร้างข้อมูลที่อยู่รูปแบบของตาราง  
และแต่ละตารางก็จะมีความสัมพันธ์กัน เช่น

* One-to-one
* One-to-many
* Many-to-many

เมื่อระบบมีขนาดใหญ่ขึ้น  
จำนวนตารางก็เยอะขึ้น  
รวมทั้งความสัมพันธ์ต่าง ๆ ก็มากขึ้นเช่นกัน

**แสดงความสัมพันธ์แบบ Many-to-many ระหว่าง Person และ Hobby ดังรูป**

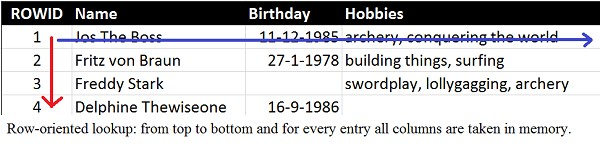


มาดูว่าโครงสร้างข้อมูลของ NoSQL Database แต่ละชนิดเป็นอย่างไร

1. Column-Oriented Database

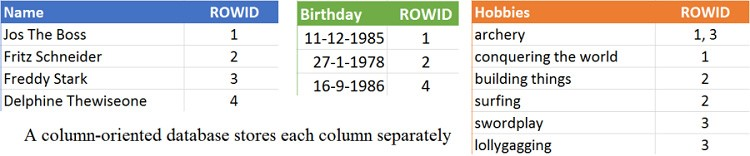
ถ้าเปรียบเทียบกับ RDBMS จะเห็นว่ามันเป็น **Row-based oriented**  
นั่นคือแต่ละ row ของข้อมูลประกอบไปด้วย  
ID ที่เป็น primary key และ field หรือ column ต่าง ๆ  
โดยแต่ละ row จะถูกจัดเก็บในตาราง

ดังนั้นในการดึงข้อมูลจากตารางจะเป็นแบบ  
อ่านจากบนลงล่าง และ ซ้ายไปขวา  
โดยข้อมูลแต่ละ row จะถูก load ไปยัง memory  
ซึ่งมันทำให้เสียเวลา และ ใช้ memory อย่างมากมาก  
แสดงดังรูป



ดังนั้น เพื่อเพิ่มความเร็วในการเข้าถึงข้อมูล  
เราจึงทำการสร้าง index ให้ตาม column ที่เราต้องการ  
แต่มันเป็นการเพิ่ม overhead ให้แก่ระบบ  
ลองคิดดูว่า ถ้าเราทำการ index ทุก ๆ column ล่ะ !!

**ดังนั้น Column-Oriented Database จึงสร้างมาเพื่อช่วยแก้ไขปัญหาเหล่านี้**  
โดยแต่ละ column จะถูกจัดเก็บแยกกัน  
ทำให้การเข้าถึงข้อมูลในแต่ละ column เร็วขึ้น  
รวมทั้งทำให้ง่ายต่อการบีบอัดข้อมูลอีกด้วย  
เนื่องจากในแต่ละตารางจัดเก็บข้อมูลเพียงชนิดเดียว  
แสดงดังรูป

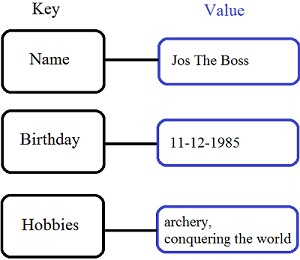


**ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Column-Oriented เช่น**

* Apache HBase
* Cassandra
* Hypertable
* Google BigTable

2. Key-Value Database

เป็นโครงสร้างข้อมูลที่เรียบง่าย และ ไม่ซับซ้อนที่สุดแล้ว  
จากความเรียบง่ายนี่เอง  
ทำให้ Key-Value มันสามารถรองรับการใช้งานจำนวนมากได้  
สามารถรองรับข้อมูลจำนวนมากได้อย่างสบาย ๆ  
แสดงโครงสร้างดังรูป



**ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Key-Value เช่น**

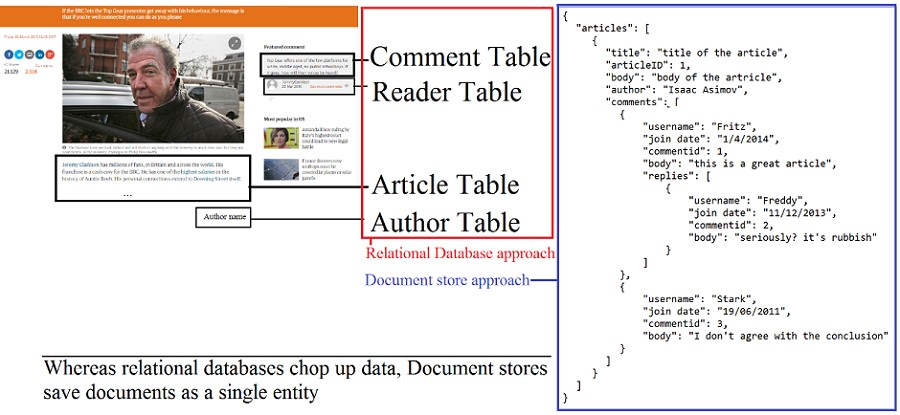
* Memcached
* Redis
* Riak
* Voldemort  
  Amazon Dynamo

3. Document Database

เป็นโครงสร้างที่ซับซ้อนมาอีกหนึ่งชั้นจาก Key-Value database  
ทำการเก็บข้อมูลในรูปแบบของเอกสาร  
แน่นอนว่า ในแต่ละเอกสารต้องมีโครงสร้างข้อมูลเช่นกัน  
เรามักจะเรียกว่า **Schema**

**โดยโครงสร้างแบบนี้จะถูกนำไปใช้อย่างมาก**  
เนื่องจากข้อมูลส่วนใหญ่จะอยู่ในรูปแบบของเอกสารอยู่แล้ว  
หรือถ้าเทียบกับ RDBMS  
เราอาจจะบอกว่าได้ มันคือข้อมูลที่ถูกทำการ Nomalization เพียงเล็กน้อย  
หรือบางคนบอกว่ามันคือ การ Denomalization นั่นเอง  
ทำให้ NoSQL ชนิดนี้เกิดมาเพื่อแก้ไขบางอย่าง  
ที่ RDBMS ไม่ตอบโจทย์นั่นเอง

**ลองคิดดูสิว่า**  
ถ้าข้อมูลของเราเป็นหนังสืมพิมพ์ หรือ นิตยสาร  
เมื่อนำมาจัดเก็บใน RDBMS แล้ว พบว่า  
ต้องทำการแยกข้อมูลไปจัดเก็บในแต่ละตาราง !!  
ทั้ง ๆ ที่เราสามารถบันทึกข้อมูลในรูปแบบของเอกสารเพียงเอกสารเดียวได้  
มันน่าจะช่วยลดงานต่าง ๆ ลงไปได้เยอะนะ ว่าไหม ?  
แสดงดังรูป



**ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Document เช่น**

* MongoDB
* CouchDB

4. Graph Database

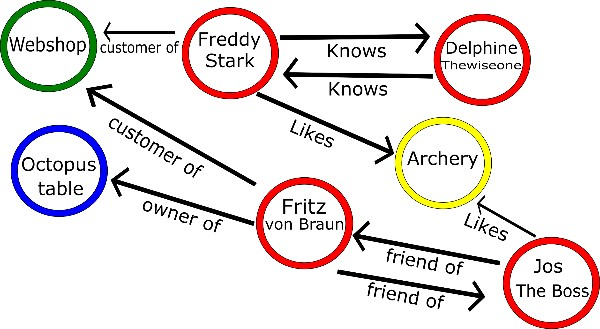
เป็นโครงสร้างข้อมูลที่มีความซับซ้อนสูงที่สุด  
เนื่องจากใช้จัดเก็บข้อมูลที่มีความสัมพันธ์ระหว่างกัน  
โดยมักจะใช้งานในเรื่องของ

* Social Networking
* Scientific paper citation
* Capital asset cluster
* Direction in map

โครงสร้งข้อมูลแบบ Graph จะประกอบไปด้วย

* **Node** คือ ข้อมูลหรือ entity หนึ่ง ๆ ตัวอย่างเช่นใน Social Network คือ ผู้ใช้งาน
* **Edge** เป็นความสัมพันธ์ระหว่าง entity ซึ่งแสดงอยู๋ในรูปแบบเส้น และ มีคุณบัติต่าง ๆ อยู่ด้วย รวมทั้งยังมีทิศทาง หรือ direction อีกด้วย

แสดงดังรูป



**ตัวอย่าง product ที่มีโครงสร้างข้อมูลเป็น Graph เช่น**

* Neo4j
* OrientDB