การพัฒนาและการประยุกต์ระบบฐานข้อมูลหลายรูปแบบแบบคลาวด์ I

Cloud-based Polyglot Application Development

ประเสต มีงบรรเจิดฤทธิ์

อังคณา สุขัยมานะเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

การพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ I

Cloud-based Polyglot Application Development I

ประณต มิ่งบรรเจิดสุข
อังคณา สุชัยมานะเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ I

Cloud-based Polyglot Application Development I

ผู้จัดทำ

1. นายประณต	มิ่งบรรเจิดสุข	รหัสนักศึกษา	53010924
2. นางสาวอังคณา	สุชัยมานะเจริญ	รหัสนักศึกษา	53011911

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. ศุภมิตร จิตตะยโศธร)

การพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ I

นาย ประณต	มิ่งบรรเจิดสุข	53010924
นางสาว อังคณา	สุชัยมานะเจริญ	53011911
รศ.ดร. ศุภมิตร	จิตตะยโศธร	อาจารย์ที่ปรึกษา

ปีการศึกษา 2556

บทคัดย่อ

ระบบจัดเก็บข้อมูลหลายรูปแบบ คือระบบจัดเก็บข้อมูลที่ใช้แบบจำลองข้อมูลมากกว่าหนึ่งแบบจำลองเพื่อสนับสนุนโปรแกรมประยุกต์ ระบบโดยทั่วไปประกอบด้วยระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ และระบบจัดการฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (NoSQL) ซึ่งอาจเป็นระบบการจัดการฐานข้อมูลแบบ ออปเจ็ค, เอกสาร หรือกราฟ ซึ่งระบบจัดเก็บข้อมูลเหล่านี้สามารถปฏิบัติงานบนคลาวด์ได้

โครงงานนี้ศึกษาระบบจัดเก็บข้อมูลหลายรูปแบบที่ปฏิบัติงานบนสภาพแวดล้อมเป็นการประมวลผลคลาวด์ ในหัวข้อการประมวลผลทรานแซกชั่น, การสร้างฟังก์ชั่นลงในฐานข้อมูล, การแทนข้อมูล และการกระจาย โดยมีการพัฒนาโปรแกรมประยุกต์ต้นแบบ ซึ่งมีการใช้งานร่วมกันของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ดีบีทู และระบบจัดการข้อมูลที่ไม่ใช่เชิงสัมพันธ์ซึ่งจัดเก็บข้อมูลเป็นเอกสาร (Document) มองโกดีบี (MongoDB) บนคลาวด์

โครงงานนี้ใช้อเมซอน คลาวด์ แพลตฟอร์ม โดยใช้อเมซอน อีลาสติก คลาวด์ สนับสนุนระบบการจัดเก็บข้อมูลหลายรูปแบบ

Cloud-based Polyglot Application Development I

Mr. Pranot	Mingbunjurdsuk	53010924
Ms. Angkana	Suchaimanacharoen	53011911
Assoc. Prof. Dr. Suphamit	Chittayasothorn	Advisor

Academic Year 2013

ABSTRACT

Polyglot storage systems are storage systems that employ more than one data model to support applications. Typical systems comprise a relational database management system and the so-called NoSQL systems. NoSQL could be object, document or graph database management systems. These storage systems could be conveniently implemented on a cloud platform.

This project studies polyglot data storage implementation on a cloud computing environment. Typical topics include transaction processing, user-defined function, data representation and distribution. Prototype application is implemented. The application uses both DB2, relational database management system, and MongoDB, document database management system.

Amazon cloud platform is used in this project. Polyglot storage systems are those which are supported by Amazon Elastic Cloud.

กิตติกรรมประกาศ

โครงงานการพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ I สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความช่วยเหลืออย่างดียิ่งจาก รศ. ดร. ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ผู้ซึ่งให้คำแนะนำ คำสั่งสอน ชี้แนะแนวทางการทำงาน ชี้ข้อปรับปรุงแก้ไข และติดตามความก้าวหน้าของโครงงานอย่างสม่ำเสมอ

ขอขอบพระคุณคณาจารย์สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนวิชาความรู้ต่าง ๆ แก่ข้าพเจ้ามาโดยตลอด ทำให้ข้าพเจ้าสามารถนำความรู้เหล่านั้นมาใช้ในโครงงานนี้

ขอขอบคุณรุ่นพี่ รวมทั้งเพื่อน ๆ ทุกคนที่ให้คำปรึกษา แลกเปลี่ยนความคิดเห็น ให้กำลังใจ และแนะนำการทำโครงงานในด้านต่าง ๆ ตลอดมา

ขอขอบพระคุณบิดา มารดาและครอบครัว ที่ให้การอบรมสั่งสอน เลี้ยงดู ให้โอกาสทางการศึกษา และให้การสนับสนุนในทุก ๆ ด้านมาโดยตลอด

สุดท้ายนี้ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้ข้าพเจ้าได้เข้ามาศึกษาหาความรู้ที่นี่ ข้าพเจ้ารู้สึกเป็นเกียรติอย่างยิ่ง คุณความดีใด ๆ ที่ปรากฏในโครงงานนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่านมา ณ ที่นี้

นาย ประณต มิ่งบรรเจิดสุข

นางสาว อังคณา สุชัยมานะเจริญ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันในขณะที่การใช้งานระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) ได้รับความนิยม แต่ยังมีหลายองกรที่เลือกใช้ระบบฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ (NoSQL) ซึ่งมีแบบจำลองข้อมูล (Data model) ต่างจากฐานข้อมูลเชิงสัมพันธ์ โดยระบบฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ แต่ละผลิตภัณฑ์จะมีรูปแบบการเก็บข้อมูลที่แตกต่างกันไป เช่น เป็นออปเจ็ค (Object), กราฟ (Graph) หรือเป็นเอกสาร (Document) และมีคำถามตามมาว่าข้อมูลแบบใดเหมาะฐานข้อมูลแบบใด ระบบฐานข้อมูลแต่ละแบบมีข้อดีข้อเสียต่างกันอย่างไร และทำอย่างไรจึงสามารถรวมเอาข้อดีของทั้งสองระบบมาใช้งานร่วมกันเพื่อเป็นทางออกใหม่ให้กับองค์กร โดยไม่ต้องพึ่งเพียงระบบฐานข้อมูลเชิงสัมพันธ์เท่านั้น

โครงงานนี้จึงจัดทำขึ้นเพื่อศึกษาถึงข้อดีและข้อเสียของการเก็บข้อมูลในฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ โดยผลิตภัณฑ์ที่เลือกใช้สำหรับฐานข้อมูลเชิงสัมพันธ์คือ ดีบีทู (DB2) และผลิตภัณฑ์สำหรับฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ คือมองโกดีบี (MongoDB) รวมถึงศึกษาความเหมาะสมว่าข้อมูลแบบใดควรเก็บในฐานข้อมูลเชิงสัมพันธ์และข้อมูลแบบใดควรเก็บในฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ เพื่อให้เกิดประสิทธิภาพสูงสุดในการพัฒนาโปรแกรมประยุกต์ และศึกษาระบบการดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor) เพื่อรักษาคุณสมบัติทั้ง 4 ประการของทรานแซกชั่น คือ ความเป็นหนึ่งเดียว (Atomicity), ความสอดคล้อง (Consistency), ความโดดเดี่ยว (Isolation) และความคงทน (Durability) ในระหว่างการปฏิบัติทรานแซกชั่นร่วมกันของทั้งสองระบบ โดยจะมีการพัฒนาโปรแกรมประยุกต์เพื่อทดสอบความสามารถในการทำงานร่วมกันของทั้งสองระบบด้วย

ทั้งนี้การทดลองพัฒนาโปรแกรมประยุกต์จะทำการพัฒนาบนระบบประมวลผลคลาวด์ (Cloud Computing) ซึ่งได้รับความสนใจอย่างมากในปัจจุบัน เนื่องจากสามารถลดต้นทุนทางด้านฮาร์ดแวร์ขององค์กร และช่วยลดความเสี่ยงในหลายด้าน เช่น การมีทรัพยากรไม่เพียงพอต่อความต้องการ ความปลอดภัย และภัยธรรมชาติ รวมถึงสามารถดึงประสิทธิภาพสูงสุดของการทำงานแบบขนานของระบบฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ บนคลัสเตอร์ หรือการทำงานพร้อมกันบนเซิร์ฟเวอร์หลายเครื่อง โดยในการทดลองจะเลือกใช้บริการระบบประมวลผลบนคลาวด์ของอเมซอนเว็บ เซอร์วิส (AWS) ซึ่งเป็นผู้ให้บริการคลาวด์แพลตฟอร์ม และจำลองการตั้งเซิร์ฟเวอร์ซึ่งสามารถเลือกขนาดและแพลตฟอร์มได้ตามที่ต้องการ เพื่อใช้ในการทดลอง

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อศึกษาการใช้งานระบบการประมวลผลคลาวด์ในอเมซอน เว็บ เซอร์วิส
2) เพื่อศึกษาการใช้งานของระบบจัดการฐานข้อมูลดีบีทู (DB2) บนคลาวด์
3) เพื่อศึกษาการใช้งานของระบบจัดการฐานข้อมูลมองโกดีบี (MongoDB) บนคลาวด์
4) เพื่อศึกษาการทำงานของระบบดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor)
5) เพื่อศึกษาและพัฒนาฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นเอง (User-defined function) เพื่อรู้จำใบหน้าในดีบีทู
6) เพื่อศึกษาและพัฒนาโปรแกรมประยุกต์ที่ใช้ระบบเก็บข้อมูลหลายรูปแบบ (Polyglot storage system) บนคลาวด์

1.3 ขอบเขตของโครงงาน

การพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์จะศึกษาการใช้งานระบบจัดการฐานข้อมูลบนคลาวด์ การใช้งานภายใต้ทรานแซกชั่นเดียวกันโดยใช้ระบบดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor) และการดึงจุดเด่นของฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ มาใช้ประโยชน์ในการพัฒนาโปรแกรมประยุกต์ที่ใช้ระบบเก็บข้อมูลหลายรูปแบบบนคลาวด์

1.4 วิธีการดำเนินการ

1) ศึกษาการใช้งานอเมซอน เว็บ เซอร์วิส (Amazon Web Services)
2) ศึกษาการติดตั้ง และการใช้งานระบบจัดการฐานข้อมูลดีบีทูและมองโกดีบีบนคลาวด์
3) ศึกษา และทดลองคุณสมบัติต่าง ๆ ของมองโกดีบี
4) ศึกษาหลักการของระบบดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor)
5) ศึกษาการพัฒนาฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นเองในดีบีทู
6) ศึกษาอัลกอริทึมในการรู้จำใบหน้า
7) ศึกษาวิธีการที่ใช้ในการพัฒนาโปรแกรมประยุกต์ที่ใช้ระบบจัดการฐานข้อมูลทั้ง 2 ผลิตภัณฑ์
8) วิเคราะห์ และออกแบบโปรแกรมประยุกต์
9) พัฒนาโปรแกรมประยุกต์ที่ใช้ระบบเก็บข้อมูลหลายรูปแบบบนคลาวด์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้ ความเข้าใจเกี่ยวกับการใช้งานระบบประมวลผลคลาวด์
2) ได้รับความรู้ ความเข้าใจเกี่ยวกับระบบจัดการฐานข้อมูลทั้งแบบเชิงสัมพันธ์, ระบบจัดการฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ และการใช้งานร่วมกันผ่านระบบควบคุมการประมวลผลทรานแซกชั่น
3) สามารถอธิบายข้อแตกต่างระหว่างฐานข้อมูลทั้งแบบเชิงสัมพันธ์ และไม่เชิงสัมพันธ์และควรใช้ฐานข้อมูลแบบใดในกรณีใดบ้าง
4) ได้รับความรู้และประสบการณ์การพัฒนาฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นเอง เพื่อรู้จำใบหน้าบนดีบีทู
5) ได้รับความรู้และประสบการณ์การพัฒนาโปรแกรมประยุกต์บนคลาวด์

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาโดยทั่วไปออกเป็น 5 บทด้วยกันคือ

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ประกอบด้วยโปรแกรมประยุกต์ระบบข้อมูลหลายรูปแบบ, ระบบควบคุมการประมวลผลทรานแซกชั่น, การประมวลผลคลาวด์, ฐานข้อมูลเชิงสัมพันธ์, ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์, คุณสมบัติของมองโกดีบี, การรู้จำใบหน้า และการสร้างชนิดข้อมูลและฟังก์ชั่นขึ้นมาใช้เองในดีบีทู

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงรายละเอียดของโครงงาน เซิฟเวอร์จำลองบนคลาวด์, การออกแบบการทดลองความสามารถในการจัดการข้อมูล, การค้นหาของมองโกดีบี, การทำทรานแซกชั่นแบบกระจาย, การทำฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นเองเพื่อรู้จำใบหน้า และการออกแบบระบบงานทะเบียนนักศึกษาจำลอง

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงข้อมูลที่ใช้ในการทดลอง สภาวะแวดล้อมในการทดลอง การทดลองความสามารถในการจัดการข้อมูล, การค้นหาของมองโกดีบี, การทดลองสร้างระบบงานทะเบียนนักศึกษาจำลองเพื่อใช้งานระบบจัดเก็บข้อมูลหลายรูปแบบ และผลจากการทดลอง

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของโครงงาน ข้อจำกัด รวมถึงปัญหาอุปสรรคต่าง ๆ ของโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 โปรแกรมประยุกต์ระบบข้อมูลหลายรูปแบบ (Polyglot Application)

โปรแกรมประยุกต์ระบบข้อมูลหลายรูปแบบเป็นโปรแกรมที่มีการนำระบบข้อมูลหลายแบบมาใช้งานร่วมกัน เช่น โปรแกรมที่มีการใช้งานระบบจัดการฐานข้อมูล 2 ผลิตภัณฑ์ร่วมกัน หรือโปรแกรมที่ใช้งานฐานข้อมูล 2 โครงสร้างร่วมกัน เช่น การใช้งานฐานข้อมูลเชิงสัมพันธ์ (relational database) ร่วมกับฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (NoSQL) โดยที่แต่ละระบบฐานข้อมูลจะมีการจัดการภายในของมันเอง ทั้งนี้การทำงานร่วมกันของหลายระบบข้อมูลจะถูกจัดการในโปรแกรมประยุกต์อีกที [1]

โปรแกรมประยุกต์ที่สร้างขึ้นในการศึกษาเป็นโปรแกรมงานทะเบียนนักศึกษาจำลอง ซึ่งมีการใช้งานระบบข้อมูลร่วมกัน 2 โครงสร้าง คือ ฐานข้อมูลเชิงสัมพันธ์ และฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ซึ่งมีโครงสร้างข้อมูลแบบเอกสาร (document) ในหนึ่งทรานแซกชั่น โดยมีทรานแซกชั่นย่อย (local transaction) ซึ่งเป็นทรานแซกชั่นที่ทำบนแต่ละฐานข้อมูล ซึ่งแต่ละฐานข้อมูลนั้นมีระบบจัดการข้อมูลของตัวเองและส่งผลการทำทรานแซกชั่นให้กับส่วนที่ประสานการทำงานของแต่ละฐานข้อมูล เช่นระบบดูแลการประมวลผลทรานแซกชั่น (TP monitor) หรือโปรแกรมประยุกต์ เพื่อจัดการทรานแซกชั่นรวม (global transaction) โดยจะมีการส่งสัญญาณกับแต่ละฐานข้อมูล เพื่อทราบผลการทำ transaction ของแต่ละฐานข้อมูล และตัดสินว่าทรานแซกชั่นรวมสำเร็จหรือไม่ จากนั้นจะติดต่อกลับไปบอกผลให้แต่ละฐานข้อมูล เพื่อทำการยืนยันการเปลี่ยนแปลงข้อมูล (commit) หรือยกเลิกการเปลี่ยนแปลง (rollback) นอกจากนี้ยังทำหน้าที่ให้การรับคำสั่งทำทรานแซกชั่นจากโปรแกรมประยุกต์อีกด้วย

นอกจากนี้การเก็บข้อมูลลงในแต่ละฐานข้อมูลจะคำนึงถึงความเหมาะสมในรูปแบบการใช้งานข้อมูลนั้น โดยในโปรแกรมประยุกต์ที่สร้างขึ้นจะเก็บข้อมูลส่วนที่มีโครงสร้างข้อมูลชัดเจน มีการเพิ่ม (insert) ลบ (delete) หรือเปลี่ยนแปลงข้อมูล (update) ลงในฐานข้อมูลเชิงสัมพันธ์ และเก็บข้อมูลสำหรับแสดงผลรวมกันไว้เป็นเอกสารในฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ กล่าวคือ การเพิ่มข้อมูลวิชา ข้อมูลนักศึกษา หรือการให้เกรดจะทำกับฐานข้อมูลเชิงสัมพันธ์ ส่วนการแสดงผลการเรียนจะทำกับฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ ทั้งนี้เนื่องจากฐานข้อมูลเชิงสัมพันธ์มีการเก็บข้อมูลในแบบตาราง มีโครงสร้างชัดเจน และมีการลดความซ้ำซ้อนของข้อมูล ทำให้ดูแลความถูกต้องของข้อมูลในการเปลี่ยนแปลงข้อมูลได้ง่ายกว่า ส่วนฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์จะเก็บข้อมูลในรูปแบบของเอกสาร ซึ่งพร้อมในการแสดงผล ทำให้ไม่เรียกดูผลได้เลยโดยไม่ต้องทำงานกับหลายตารางเมื่อผู้ใช้ต้องการดูผล

2.2 ทรานแซกชั่น

ในการปฏิบัติงานกับฐานข้อมูลบางงานอาจต้องใช้คำสั่งมากกว่าหนึ่งคำสั่ง ซึ่งแต่ละคำสั่งนั้นอาจจะมีการละเมิดกฎบังคับความถูกต้องของฐานข้อมูลนั้น หรือบางงานต้องทำกับข้อมูลจำนวนมาก ซึ่งระหว่างทำงานนั้นอาจมีการละเมิดกฎบังคับความถูกต้องของฐานข้อมูลได้เช่นกัน ดังนั้นเพื่อรักษาความถูกต้องของฐานข้อมูล ระบบฐานต้องมูลจึงมีการให้ทำงานกับฐานข้อมูลเป็นทรานแซกชั่น ซึ่งในทรานแซกชั่นอาจมีหนึ่งหรือมากกว่าหนึ่งคำสั่งที่ทำงานกับฐานข้อมูลก็ได้ และจะถูกมองเป็นหน่วยเดียว โดยยอมให้ระหว่างการทำทรานแซกชั่นมีการละเมิดกฎบังคับความถูกต้องของฐานข้อมูลได้เป็นการชั่วคราว แต่เมื่อทำทรานแซกชั่นเสร็จ ผลลัพธ์จะต้องถูกต้องสอดคล้องกับกฎบังคับความถูกต้องของฐานข้อมูล ตัวอย่างของทรานแซกชั่นที่มีคำสั่งภายในมากกว่าหนึ่งคำสั่ง เช่น การโอนเงิน ซึ่งจะมีคำสั่งภายในคือการฝากเงิน และการถอนเงิน โดยกฎของการโอนเงินนี้คือ ก่อนและหลังการโอนเงินเงินในระบบจะต้องเท่าเดิม ซึ่งการฝากเงินหรือการถอนเงินจะเป็นการละเมิดกฎนี้โดยการฝากเงินจะทำให้เงินในระบบเพิ่มขึ้น และการถอนเงินจะทำให้เงินในระบบลดลง ดังนั้นทรานแซกชั่นที่ครอบทั้ง 2 กิจกรรมนี้ จะยอมให้แต่ละกิจกรรมละเมิดกฎข้อบังคับนี้ได้เป็นการชั่วคราว แต่หลังจากทำทั้งทรานแซกชั่นเสร็จแล้ว เงินในระบบจะต้องเท่าเดิมตามกฎบังคับความถูกต้อง ตัวอย่างของ ทรานแซกชั่นที่มีคำสั่งภายในหนึ่งคำสั่ง เช่น การเปลี่ยนหมู่เลือดของประชากรทั่วประเทศจาก A เป็น Z โดยกฎข้อบังคับคือฐานข้อมูลจะต้องมีหมู่เลือด A หรือ Z เพียงอย่างเดียวเท่านั้น แต่เนื่องจากการเปลี่ยนหมู่เลือดของประชากรทั้งประเทศจะต้องทำกับข้อมูลจำนวนมาก และในระหว่างเปลี่ยนหมู่เลือด จะมีคนที่ถูกเปลี่ยนหมู่เลือดเป็น Z แล้ว กับคนที่ยังเป็นหมู่เลือด A และรอเปลี่ยนเป็น Z อยู่ ดังนั้น transaction จะยอมให้ละเมิดกฎข้อนี้เป็นการภายในจนกว่าจะทำทรานแซกชั่นเสร็จ โดยทั้ง 2 ตัวอย่างที่กล่าวมาก หากทำได้สำเร็จผลสุดท้ายจะไม่ละเมิดกฎข้อบังคับ ฐานข้อมูลจะทำการบันทึกการเปลี่ยนแปลง (committed) เพื่อให้ผลทั้งหมดคงอยู่ แต่หากไม่สำเร็จเพียงหนึ่งคำสั่ง หรือมีการละเมิดกฎข้อบังคับ ฐานข้อมูลจะยกเลิกการเปลี่ยนแปลงทั้งหมด (rollback) ให้ข้อมูลกลับมาเป็นเหมือนก่อนเริ่มทำทรานแซกชั่นทั้งนี้ระบบฐานข้อมูลจะรักษาความถูกต้องของฐานข้อมูลตามคุณสมบัติของทรานแซกชั่นดังนี้ [2]

1. ความเป็นหนึ่งเดียว (Atomicity)

ทั้งทรานแซกชั่นจะถูกมองเป็นหน่วยเดียว ไม่สามารถแบ่งย่อยได้ ถ้าทรานแซกชั่นสำเร็จแสดงว่าทุกๆกิจกรรมในทรานแซกชั่นต้องสำเร็จทั้งหมด หรือถ้ามีกิจกรรมใดล้มเหลว ทุกกิจกรรมจะล้มเหลวทั้งหมด

2. ความสอดคล้อง (Consistency)

ฐานข้อมูลจะต้องถูกต้องและสอดคล้องกันทั้งก่อนและหลังทำทรานแซกชั่นกล่าวคืออาจมีการยอมให้ละเมิดกฎบังคับความถูกต้อง หรือให้ข้อมูลมีความไม่สอดคล้องได้ในระหว่างทำ ทรานแซกชั่นแต่เมื่อเสร็จการทำทรานแซกชั่นแล้ว ฐานข้อมูลนั้นจะต้องถูกต้องตามกฎ

3. ความโดดเดี่ยว (Isolation)

ทรานแซกชั่นที่ทำกับฐานข้อมูลเดียวกันในช่วงเวลาเดียวกัน จะถูกมองเป็นการทำงานตามกัน เสมือนมีทรานแซกชั่นเดียวที่ทำงานในช่วงเวลาหนึ่งๆ และแต่ละ transaction จะไม่รบกวนกัน ทั้งนี้ผู้เขียนโปรแกรมสามารถตั้งระดับความโดดเดี่ยวได้ตามความเหมาะสมของแต่ละงาน

4. ความคงทน (Durability)

เมื่อทรานแซกชั่นสำเร็จและยืนยันการเปลี่ยนแปลงแล้ว ผลการเปลี่ยนแปลงนั้นจะต้องอยู่ถาวร แม้ระบบจะเกิดความเสียหาย

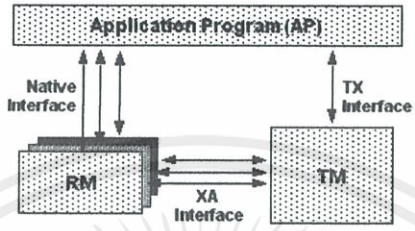
2.2.1 ทรานแซกชั่นแบบกระจาย

การทำทรานแซกชั่นแบบกระจายเป็นการทำทรานแซกชั่นร่วมกันของหลายฐานข้อมูล โดยจะมีการทำงานเป็น 2 ส่วนคือทรานแซกชั่นย่อย เป็นทรานแซกชั่นที่ปฏิบัติงานกับฐานข้อมูลใดฐานข้อมูลหนึ่ง มีระบบการจัดการของตัวเอง และไม่ติดต่อสื่อสารกันระหว่างกับฐานข้อมูลอื่นๆ และทรานแซกชั่นรวม เป็นทรานแซกชั่นรวมที่จัดการโดยผู้ประสานงานระหว่างทรานแซกชั่นโดยทำหน้าที่ติดต่อไปยังแต่ละฐานข้อมูล และรับผลการทำทรานแซกชั่นเพื่อตัดสินความสำเร็จของทรานแซกชั่นรวมโดยถ้าทรานแซกชั่นรวมสำเร็จ แสดงว่าทุกทรานแซกชั่นย่อยสำเร็จ แต่ถ้ามีทรานแซกชั่นย่อยใดไม่สำเร็จทรานแซกชั่นรวมจะไม่สำเร็จด้วย แล้วส่งผลไปให้แต่ละฐานข้อมูลเพื่อยืนยันการ หรือยกเลิกการเปลี่ยนแปลงของทรานแซกชั่นที่ดูแล ซึ่งมีผลสามารถทำให้รักษาคุณสมบัติของทรานแซกชั่นของฐานข้อมูลในการทำทรานแซกชั่นแบบกระจายได้

2.2.2 จาวาทรานแซกชั่นเอพีไอ (Java Transaction API)

จาวาทรานแซกชั่นเอพีไอ (JTA) เป็น เอพีไอ (Application Programming Interface) ที่ใช้กับจาวาอีอี (Java EE) ทำหน้าที่ในการจัดการทรานแซกชั่นแบบกระจายบนสถาปัตยกรรม เอ็กซ์/โอเพ่นเอ็กซ์เอ (X/Open XA) ซึ่งประกอบด้วยระบบควบคุมการประมวลผลทรานแซกชั่น (TP monitor) ซึ่งทำหน้าที่ประสานงานระหว่างหลายฐานข้อมูลที่นำมาใช้ร่วมกัน และติดต่อกับโปรแกรมประยุกต์เพื่อรับคำสั่งในการทำทรานแซกชั่น เช่น เริ่มทรานแซกชั่น (begin) ยืนยันการเปลี่ยนแปลง (commit) โดยฐานข้อมูลที่เข้าร่วมในทรานแซกชั่นแบบกระจายปกติจะมีระบบการจัดการทรานแซกชั่นและสามารถติดต่อเพื่อรับคำสั่งในการทำทรานแซกชั่นจากโปรแกรมประยุกต์ได้โดยตรงอยู่แล้ว เมื่อต้องการ

ทำทรานแซกชั่นแบบกระจายจะต้องตั้งค่าในทรัพยากรเอ็กซ์เอ (XAResource) เพื่อให้สามารถติดต่อกับระบบควบคุมการประมวลผลทรานแซกชั่นได้ แล้วรับคำสั่งทรานแซกชั่นผ่านระบบควบคุมการประมวลผลทรานแซกชั่นอีกที [3]



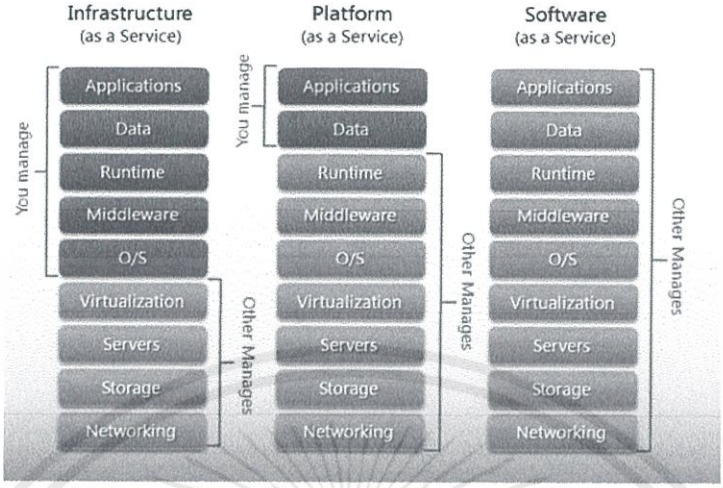
รูปที่ 2.1 การติดต่อระหว่างโปรแกรมประยุกต์ ฐานข้อมูล (RM) และระบบควบคุมการประมวลผล ทรานแซกชั่น(TM) ในทรานแซกชั่นแบบกระจาย

2.3 การประมวลผลคลาวด์ (Cloud computing)

การประมวลผลคลาวด์เป็นการให้บริการทรัพยากรทางคอมพิวเตอร์ ผู้ให้บริการจะมีทรัพยากรต่างๆที่ระบบต้องการ เช่น ฮาร์ดแวร์ ระบบปฏิบัติการ ระบบไฟล์ ระบบเครือข่าย และระบบรักษาความปลอดภัย และมีซอร์ฟแวร์สำหรับการให้บริการแบบเวอร์ชวลไลเซชัน (virtualization) ซึ่งสามารถแบ่งทรัพยากรให้ผู้ใช้บริการได้หลายๆระบบ บนฮาร์ดแวร์ชุดเดียวกัน และผู้ใช้แต่ละคนจะสามารถร้องขอทรัพยากรตามความต้องการเท่าที่ระบบมีให้ได้ และเห็นเสมือนตนเองเป็นเจ้าของทรัพยากรทั้งหมดที่ขอนั้น โดยไม่มีผู้อื่นมาร่วมใช้ด้วย [2]

การใช้บริการคลาวด์สามารถช่วยให้ผู้ใช้บริการประหยัดค่าใช้จ่ายในการซื้อทรัพยากร บำรุงรักษา และประหยัดเวลาในการติดตั้งระบบเอง อีกทั้งผู้ให้บริการสามารถใช้ทรัพยากรที่มีอยู่ได้อย่างเต็มที่ โดยสามารถแบ่งทรัพยากรที่ยังว่างอยู่ให้กับผู้ที่ต้องการใช้ทรัพยากรได้

ผู้ใช้บริการคลาวด์สามารถเลือกทรัพยากรได้เองตามที่ระบบมีให้ โดยไม่ต้องติดต่อกับผู้ให้บริการโดยตรง และสามารถเพิ่มหรือลดขนาดทรัพยากรได้ภายหลังทำให้มีความยืดหยุ่นในการใช้งานเสมือนกับไม่มีข้อจำกัดทางทรัพยากรคอมพิวเตอร์ และสามารถจ่ายค่าบริการตามการใช้งานจริงขึ้นกับขนาดและระยะเวลาที่ใช้งาน นอกจากนี้การให้บริการคลาวด์ยังถูกแบ่งย่อยเป็น 3 ระดับขึ้นกับความรับผิดชอบในการจัดการทรัพยากร ได้แก่ ซอฟต์แวร์ แอส อะ เซอร์วิส (Software as a Service: SaaS), แพลตฟอร์ม แอส อะ เซอร์วิส (Platform as a Service: PaaS) และอินฟราสตรัคเจอร์ แอส อะ เซอร์วิส (Infrastructure as a Service: IaaS) แสดงในรูปที่ 2.2 [4]



รูปที่ 2.2 แสดงความรับผิดชอบในการจัดการทรัพยากรของการให้บริการคลาวด์ในแต่ละระดับ

2.3.1 ซอฟต์แวร์ แอส อะ เซอร์วิส

ซอฟต์แวร์ แอส อะ เซอร์วิส เป็นการให้บริการคลาวด์ในรูปแบบของโปรแกรมประยุกต์ที่ให้งานได้เลย โดยมากเป็นโปรแกรมสำหรับจัดการข้อมูลทางธุรกิจ ทำให้ผู้ใช้ลดภาระในการดูแลโปรแกรม รวมถึงการอัพเดทต่างๆและการจัดการข้อมูล เนื่องจากผู้ให้บริการเป็นผู้ดูแลให้จากส่วนกลาง โดยบริการนี้เหมาะสำหรับผู้ที่ต้องการใช้โปรแกรมประยุกต์ในการทำงานทั่วๆไปไม่เฉพาะเจาะจง และไม่เป็นความลับมาก เช่น บริการอีเมลล์ โปรแกรมที่เชื่อมต่อกับมือถือ โปรแกรมสำหรับออกใบเสร็จ แต่จะไม่เหมาะกับโปรแกรมที่ต้องการการประมวลผลเร็ว ข้อมูลมีความลับ หรือต้องการรูปแบบที่เฉพาะเจาะจงกับองค์กร ตัวอย่างบริการ เช่น เซลล์ฟอร์ส (Salesforce) และเบสแคมป์ (basecamp)

2.3.2 แพลตฟอร์ม แอส อะ เซอร์วิส

แพลตฟอร์ม แอส อะ เซอร์วิส เป็นบริการคลาวด์ที่ผู้ให้บริการจะบริการระบบให้กับผู้ขอใช้บริการ โดยผู้ใช้จะดูแลเพียงโปรแกรมประยุกต์ กับข้อมูลของตัวเอง โดยสามารถพัฒนาโปรแกรมประยุกต์และเชื่อมต่อกับบริการ ผู้ให้บริการจะทำการปล่อยโปรแกรม สร้างเซิร์ฟเวอร์และให้ที่อยู่ (URL) ของโปรแกรมประยุกต์ให้ นอกจากนี้ให้ความยืดหยุ่นแก่โปรแกรมโดยสามารถเพิ่มการใช้ทรัพยากรได้เพื่อมีการเข้าถึงโปรแกรมมากขึ้น หรือโปรแกรมใหญ่ขึ้น และมีการแบ่งกันทำงาน (load balancing) โดยหลายๆเซิร์ฟเวอร์ เพื่อให้โปรแกรมสามารถทำงานได้เร็วและทำงานได้ตลอดเวลา หากทรัพยากรบางส่วนไม่สามารถใช้การได้ นอกจากนี้ยังบริการฐานข้อมูลตามมาตรฐานให้อีกด้วย

ตัวอย่างบริการ เช่น อเมซอน อีลาสติก บีนทอร์ค (Amazon Elastic Beanstalk), กูเกิ้ล แอป เอนจิน (Google App Engine) และ เมนดิซ (Mendix)

2.3.3 อินฟราสตรัคเจอร์ แอส อะ เซอร์วิส

อินฟราสตรัคเจอร์ แอส อะ เซอร์วิส เป็นบริการคลาวด์ที่ให้บริการโดยผู้ใช้บริการเป็นคนจัดการโปรแกรมประยุกต์ไปจนถึงเซิร์ฟเวอร์ โดยผู้ให้บริการจะจัดสรรทรัพยากรฮาร์ดแวร์ หน่วยความจำ ระบบเครือข่ายให้ ผู้ใช้สามารถเข้าใจงานโดยสร้างอินสแตนซ์ขึ้นมาเหมือนเป็นคอมพิวเตอร์เครื่องหนึ่งและเลือกความสามารถได้ตามต้องการเท่าที่ระบบมีให้ เช่น ความเร็วของซีพียู พื้นที่หน่วยความจำ รวมถึงระบบปฏิบัติการ และผู้ให้บริการจะให้ไอพีของแต่ละอินสแตนซ์ เพื่อใช้ในการเข้าถึงอินสแตนซ์นั้นด้วย และเนื่องจากการทำงานเป็นแบบเวอชวลไลซ์ การเก็บข้อมูลจึงเก็บในไฟล์ที่สร้างเสมือนเป็นดิสก์หน่วยความจำ ทำให้สามารถจัดการกับข้อมูลได้อย่างรวดเร็วเนื่องจากเป็นการทำงานกับไฟล์ไม่ใช่กับฮาร์ดแวร์จริงๆ ทำให้การเปลี่ยนแปลงโครงสร้างทำได้งาน รวมถึงมีการขยายขนาดทรัพยากรก็ทำได้ง่ายเช่นกัน แต่อาจมีข้อจำกัดบ้างในบางงานที่เฉพาะเจาะจงเกินไป หรือความสามารถของทรัพยากรอาจไม่พอดีกับงานที่ต้องใช้ทรัพยากรที่มีความสามารถสูงมากๆ

2.4 อเมซอน เว็บ เซอร์วิส (Amazon Web Services: AWS)

อเมซอน เว็บ เซอร์วิสเป็นผู้ให้บริการคลาวด์ ก่อตั้งขึ้นในปี 2006 เพื่อให้บริการแก่ผู้พัฒนาโปรแกรมสำหรับให้บริการเว็บไซต์ โดยสามารถเข้าถึงบริการได้ผ่านทางเอชทีทีพี (HTTP) และเก็บค่าบริการตามการใช้งาน ในปี 2003 อเมซอนมีโครงการให้บริการคลาวด์แบบ infrastructure และขยายขอบเขตการบริการ เช่น ให้บริการที่เก็บข้อมูล โครงสร้างระบบภายในและได้ก่อตั้งอเมซอน อีซีทู ซึ่งถูกพัฒนาโดยคริสต์ บราวน์ (Chris Brown) นักพัฒนาโปรแกรมจากเมืองเคปทาวน์ ประเทศแอฟริกาใต้ จากนั้นอเมซอนได้ให้นักพัฒนาโปรแกรมได้ลองใช้งานจำนวนมากกว่า 180,000 คน และมีการพัฒนาต่อมาเรื่อยๆ ในปัจจุบันผู้ใช้งานสามารถขอใช้งานได้ผ่านเว็บไซต์อเมซอนดอทคอม (Amazon.com) ซึ่งมีบริการพื้นที่สำหรับเก็บข้อมูล การบริการฐานข้อมูล การติดตั้งโปรแกรมประยุกต์บนเครือข่าย ระบบเครือข่าย การวิเคราะห์ และบริการที่เกี่ยวกับโปรแกรมประยุกต์ โดยมีศูนย์เซิร์ฟเวอร์ใน 8 พื้นที่ ได้แก่ อเมริกาตะวันออก (ตอนเหนือของเวอร์จิเนีย) อเมริกาตะวันตก (ตอนเหนือของแคลิฟอร์เนีย และ รัฐออริกอน) เซาเปาลู (ประเทศบราซิล) ไอร์แลนด์ สิงคโปร์ โตเกียว และซิดนีย์ ในพื้นที่ที่ปลอดภัยเพื่อป้องกันความปลอดภัยของข้อมูล และความต่อเนื่องของบริการ

2.4.1 อเมซอน อีซีทู

อเมซอน อีซีทู เป็นการให้บริการคลาวด์ของอเมซอน เว็บ เซอร์วิสแบบอินฟราสตรัคเจอร์แอส อะ เซอร์วิส ให้บริการเซิร์ฟเวอร์แก่ผู้ใช้งาน โดยผู้ใช้บริการสามารถสร้างอินสแตนซ์ขึ้นมาได้เหมือนเป็นคอมพิวเตอร์เครื่องหนึ่ง และสามารถเลือกระบบปฏิบัติการและความสามารถ เช่น ความเร็วในการประมวลผลของ ซีพียู พื้นที่ของอินสแตนซ์และการรักษาความปลอดภัยได้ ผ่านทางหน้าเว็บไซต์ และสามารถปรับเปลี่ยนความสามารถของอินสแตนซ์ได้ภายหลังตามความต้องการโดยไม่กระทบกระเทือนต่อระบบ นอกจากนี้ยังสามารถทำงานร่วมกับบริการอื่นๆของอเมซอนได้ เช่น อเมซอนเอสทรี (Amazon S3), อเมซอน อาร์ดีเอส (Amazon RDS), อเมซอน เอสคิวเอส (Amazon SQS) นอกจากนี้ยังมีการรักษาความปลอดภัยโดยมีการใช้คลาวด์เสมือนจริงส่วนตัว (Virtual Private Cloud: VPC) มี รายการควบคุมการเข้าถึง (ACLs) เพื่อควบคุมการเข้าถึงอินสแตนซ์ นอกจากนี้ค่าบริการเป็นไปตามการใช้งานเป็นรายชั่วโมงของแต่ละ instance [5]

2.4.2 อเมซอน อีลาสติก บีนทอร์ค

อเมซอน อีลาสติก บีนทอร์ค เป็นบริการคลาวด์ของอเมซอน เว็บ เซอร์วิสแบบแพลตฟอร์มแอส อะ เซอร์วิสช่วยให้ผู้ใช้บริการสามารถสร้างโปรแกรมประยุกต์ได้ง่าย โดยจะจัดการรายละเอียดสำหรับการสร้างรวมถึงสร้างเว็บเซิร์ฟเวอร์ตามภาษาที่ใช้เขียนโปรแกรมในการจัดเก็บโปรแกรมประยุกต์ โดยมีเอชทีทีพี เซิฟเวอร์ (HTTP Server) สำหรับ โหนด เจเอส(Node.js) พีเอชพี (PHP) และ ไพทอน (Python), พาสเซ็นเจอร์ (Passenger) สำหรับรูบี้ (Ruby), ไอไอเอส (IIS) 7.5 สำหรับดอทเน็ต (.NET) และอะแพชี ทอมแคท (Apache Tomcat) สำหรับจาวาให้ นอกจากนี้ยังให้ความยืนหยุ่นในการใช้งานเซิร์ฟเวอร์ สามารถปรับเปลี่ยนความสามารถของเซิร์ฟเวอร์ได้ หรือจัดการรายละเอียดอื่นๆได้ ทำให้ผู้ใช้สามารถควบคุมการใช้งานได้อย่างเต็มที่ รวมถึงสามารถดูล็อกไฟล์ (log file) และตรวจสอบการทำงานของโปรแกรมประยุกต์ที่ปล่อยออกไปได้อีกด้วย นอกจากนี้ยังสามารถใช้งานร่วมกับบริการอื่นๆของอเมซอน เว็บ เซอร์วิส รวมถึงขอใช้ฐานข้อมูลที่มีให้ในบริการได้ และมีการขยายพื้นที่ของเซิร์ฟเวอร์ให้อัตโนมัติเมื่อมีการขยายของโปรแกรมประยุกต์หรือมีการใช้งานเพิ่มขึ้น และลดพื้นที่การใช้งานเมื่อโปรแกรมถูกใช้งานน้อย โดยค่าใช้จ่ายคิดตามชั่วโมงที่ใช้งาน [6]

2.5 ฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

ฐานข้อมูลเชิงสัมพันธ์ คือ แบบจำลองข้อมูลเพื่อนำเสนอข้อมูลและความสัมพันธ์ของข้อมูลในรูปแบบตารางประกอบด้วยแถวและคอลัมน์ โดยหนึ่งคอลัมน์มาจากหนึ่งโดเมน และเป็นหน่วยที่เล็กที่สุด (atomic) สามารถทำงานได้ดีกับภาษาเอสคิวแอล โดยตารางสามารถออกแบบให้ไม่มีความ

ซ้ำซ้อน หรือมีความซ้ำซ้อนของข้อมูลน้อย เพื่อสะดวกในการหาคำตอบ และการเปลี่ยนแปลงข้อมูลโดยไม่กระทบต่อข้อมูลอื่นๆที่ไม่ถูกเปลี่ยนแปลง ทำให้ข้อมูลมีความถูกต้องตามกฎรักษาความถูกต้องของฐานข้อมูล (integrity constraint) นอกจากนี้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์หลายผลิตภัณฑ์ยังรองรับการทำทรานแซกชั่นหรือ การทำงานกับกลุ่มคำสั่งที่ยอมให้มีการละเมิดกฎรักษาความถูกต้องของฐานข้อมูลเป็นการภายในอีกด้วย

ในปัจจุบันฐานข้อมูลเชิงสัมพันธ์ได้รับความนิยมเป็นอย่างมาก และนิยมใช้ในงานที่ต้องการความถูกต้องของข้อมูลสูง หรือมีการทำทรานแซกชั่นหรือมีการเพิ่ม ลบแถวข้อมูล หรือปรับปรุงข้อมูล เนื่องจากสามารถรักษาความถูกต้องของข้อมูลได้ดี แต่อาจใช้เวลานานในการแสดงผลหรือหาคำตอบจากข้อมูลหลายตาราง

2.5.1 ดีบีทู (DB2)

ผลิตภัณฑ์จัดการฐานข้อมูลเชิงสัมพันธ์ที่นำมาใช้คือดีบีทูซึ่งจะใช้เป็นส่วนสำหรับเก็บข้อมูลในส่วนที่มีโครงสร้างชัดเจน และมีการเพิ่ม เปลี่ยน หรือลบข้อมูล เช่น การเพิ่มวิชาในระบบทะเบียน การเพิ่มข้อมูลนักศึกษาใหม่ หรือการให้เกรด นอกจากนี้ยังใช้เป็นฐานข้อมูลในการเก็บรูปภาพ และฟังก์ชั่นสำหรับเปรียบเทียบรูปภาพในลักษณะที่เรียกว่าฟังก์ชั่นที่ผู้ใช้กำหนดเอง (user defined function: UDF) ซึ่งเป็นส่วนที่ทำให้สามารถใช้ภาษาฐานข้อมูลในการเรียกดูข้อมูล (query) โดยใช้คำสั่งที่ต้องประมวลผลตามแบบที่ต้องการ ซึ่งฟังก์ชั่นที่ผู้ใช้กำหนดเองที่ทำการทดลองจะสามารถทำให้ภาษาฐานข้อมูลรับพารามิเตอร์เป็นรูปภาพของใบหน้าคน และเปรียบเทียบรูปภาพในฐานข้อมูลว่ารูปภาพในเป็นใบหน้าของคนในรูปภาพที่รับเข้ามาได้ หรือรูปภาพใดมีใบหน้าคล้ายคลึงกันมากที่สุด

2.5.2 ฟังก์ชั่นที่ผู้ใช้กำหนดเอง (User Defined Function)

ฟังก์ชั่นที่ผู้ใช้กำหนดเองในดีบีทูเป็นฟังก์ชั่นใหม่ที่ผู้ใช้กำหนดขึ้นได้นอกเหนือหรือต่อเติมจากฟังก์ชั่นเดิมที่มีอยู่แล้วในดีบีทู (build-in function) ซึ่งฟังก์ชั่นที่พัฒนาสามารถเขียนได้โดยใช้ภาษาโปรแกรมมิ่ง เช่น ซี, ซีพลัสพลัส และจาวา การสร้างฟังก์ชั่นขึ้นมาใช้เองจะมีประโยชน์ในการช่วยทำให้การทำงานกับฐานข้อมูลของโปรแกรมเมอร์สะดวกขึ้น โดยการเขียนฟังก์ชั่นที่ทำงานเฉพาะอย่างลงไปและประกาศให้ดีบีทูรับรู้โดยใช้คำสั่ง CREATE FUNCTION หรือคอมไพล์ฟังก์ชั่นนั้น ๆ แล้วนำไปเก็บไว้ในที่ที่ดีบีทูกำหนด เช่น C:\Program Files\IBM\SQLLIB\Function จากนั้นจะสามารถเรียกดูข้อมูลด้วยภาษาฐานข้อมูลโดยเรียกใช้ฟังก์ชั่นนั้นได้เลย เช่น select * from table where function(para) = 1 โดยฟังก์ชั่นที่ผู้ใช้กำหนดในดีบีทูมีด้วยกัน 5 ประเภท ดังนี้ [7]

2.5.2.1 ฟังก์ชั่นตามต้นกำเนิดหรือแบบฉบับ (Sourced or Template Function)

เป็นการปรับแต่งฟังก์ชั่นที่สร้างมาจากฟังก์ชั่นเดิมที่มีอยู่แล้วในดีบีทูเช่น การหาผลรวม (SUM) และการหาค่าเฉลี่ย (AVG) ให้เหมาะสมกับชนิดข้อมูลที่จะนำไปใช้งาน

โปรแกรมที่ 2.1 ตัวอย่างการสร้าง ฟังก์ชั่นตามต้นกำเนิดหรือแบบฉบับ

```
CREATE FUNCTION   AVG    (HATSIZE)
RETURNS HATSIZE
SOURCE SYSIBM.AVG (INTEGER)
```

2.5.2.2 ฟังก์ชั่นจากภาษาเอสคิวแอล (SQL Scalar, Table or Row Function)

เป็นฟังก์ชั่นใหม่ที่สร้างขึ้นมาจากคำสั่งเอสคิวแอลโดยสามารถส่งค่าคืนออกมาได้ทั้ง ค่าเดี่ยว (scalar value), ตาราง (table) และแถว (row)

ตัวอย่างการสร้างฟังก์ชั่น DEPTEMPLOYEES โดยรับพารามิเตอร์คือ DEPTNO และ return ค่าออกมาเป็นตารางกระกอบด้วยแอตทริบิวต์ EMPNO, LASTNAME และ FIRSTNAME เป็นดังโปรแกรมด้านล่างนี้

โปรแกรมที่ 2.2 ตัวอย่างการสร้างฟังก์ชั่นจากภาษาเอสคิวแอล

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
RETURNS TABLE     (EMPNO CHAR(6),
                   LASTNAME VARCHAR(15),
                   FIRSTNAME VARCHAR(12))
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN
      SELECT EMPNO, LASTNAME, FIRSTNME
      FROM EMPLOYEE
      WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

2.5.2.3 ฟังก์ชั่นสเกล่าภายนอก (External Scalar Functions)

เป็นฟังก์ชั่นที่เขียนโดยภาษาอื่นที่ไม่ใช่เอสคิวแอล ที่ส่งค่าคืนเป็นค่าเดี่ยว (scalar value) โดยฟังก์ชั่นนั้นจะต้องถูกคอมไพล์ (compile) มาก่อนที่จะเอาลงดีบีทู

โปรแกรมที่ 2.3 ตัวอย่างการสร้างฟังก์ชั่นสเกล่าภายนอก

```
CREATE FUNCTION CENTRE (INT,FLOAT)
RETURNS FLOAT
EXTERNAL NAME 'mod!middle'
LANGUAGE C
PARAMETER STYLE SQL
DETERMINISTIC
```

```
NO SQL
NO EXTERNAL ACTION
```

2.5.2.4 ฟังก์ชั่นตารางภายนอก (External Table Functions)

เป็นฟังก์ชั่นที่เขียนโดยภาษาอื่นที่ไม่ใช่เอสคิวแอล ที่ส่งค่าคืนเป็นตาราง โดยฟังก์ชั่นนั้นจะต้องถูกคอมไพล์ (compile) มาก่อนที่จะเอาลงดีบีทู

โปรแกรมที่ 2.4 ตัวอย่างการสร้างฟังก์ชั่นตารางภายนอก

```
CREATE FUNCTION MAIL()
RETURNS TABLE (TIMERECEIVED DATE,
SUBJECT VARCHAR(15),
SIZE INTEGER,
TEXT VARCHAR(30))
EXTERNAL NAME 'tfmail.header!list'
LANGUAGE OLE
PARAMETER STYLE SQL
NOT DETERMINISTIC
FENCED
CALLED ON NULL INPUT
SCRATCHPAD
FINAL CALL
NO SQL
EXTERNAL ACTION
DISALLOW PARALLEL
```

2.5.2.5 ตารางภายนอก โอแอลอี ดีบี (OLE DB External Table)

ไมโครซอฟท์ โอเอลอี ดีบี (Microsoft OLE DB) เป็น เอพีไอ (API) เพื่อช่วยในการเข้าถึงข้อมูลที่จัดการโดยระบบจัดการฐานข้อมูลยี่ห้ออื่น เช่นไมโครซอฟท์ แอคเซส (Microsoft Access), ไมโครซอฟท์ เอสคิวแอล เซิฟเวอร์ (MS SQL Server), โอราเคิล ดาตาเบส (Oracle Database) เป็นต้น โดยระบุระบบจัดการฐานข้อมูลยี่ห้ออื่น ใน EXTERNAL NAME

โปรแกรมที่ 2.5 ตัวอย่างการสร้างตารางภายนอก โอแอลอี ดีบี

```
CREATE FUNCTION favorites (varchar(600))
RETURNS TABLE     (store_id CHAR (4),
                   name VARCHAR (41),
                   sales INTEGER)
SPECIFIC favorites
LANGUAGE OLEDB
EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
                   User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;
                   Locale Identifier=1033;Use Procedure for Prepare=1;
                   Auto Translate=False;Packet Size=4096;Workstation
                   ID=WALTZ;
                   OLE DB Services=CLIENTCURSOR;';
```

```
SELECT *
      FROM TABLE (favorites
                  (' select top 3 sales.stor_id as store_id, ' CONCAT
                  ' stores.stor_name as name, ' CONCAT
                  ' sum(sales. qty) as sales ' CONCAT
                  ' from sales, stores ' CONCAT
                  ' where sales.stor_id = stores.stor_id ' CONCAT
                  ' group by sales.stor_id, stores.stor_name ' CONCAT
                  ' order by sum(sales.qty) desc ')) as f;
```

2.5.3 ฟังก์ชั่นที่ผู้ใช้กำหนดเองเป็นภาษาจาวาในดีบีทู

การเขียนฟังก์ชั่นโดยใช้รูปแบบการเขียนแบบภาษาจาวาในฟังก์ชั่นที่ผู้ใช้กำหนดเองจะมีรูปแบบตามที่ดีบีทูกำหนด คือฟังก์ชั่นจะส่งค่าคืนหนึ่งค่าและมีรูปแบบการรับค่าหรือการส่งพารามิเตอร์ (parameter style) ได้ 2 วิธีคือ จาวา (Java) และดีบีทู เจนเนอรัล (DB2GENERAL) โดยทั้ง 2 วิธีจะต้องเขียนอยู่ใน ฟังก์ชั่นที่เป็นแบบพับบลิก สแตติก (public static) ค่าที่รับส่งเข้าออกจากฟังก์ชั่นจะต้องมีชนิดข้อมูลที่สามารถแปลงไปเป็นชนิดข้อมูลในภาษาเอสคิวแอลได้ และมีการอิมพอร์ต ไลบรารี่ com.ibm.db2.app.* เพื่อให้เรียกใช้ฟังก์ชั่นต่างๆของดีบีทูได้

แต่จะมีความแตกต่างกันที่รูปแบบพารามิเตอร์จาวา จะเขียนเป็นฟังก์ชั่นที่มีรีเทิร์นค่า และรับพารามิเตอร์สำหรับใช้ในฟังก์ชั่นตามปกติเหมือนในการเขียนฟังก์ชั่นภาษาจาวาทั่วไป ส่วนรูปแบบพารามิเตอร์ดีบีทู เจนเนอรัล จะมีการสืบทอดคลาส UDF และมีอาร์กูเมนต์สำหรับรีเทิร์นค่าที่อารกูเมนต์สุดท้ายของพารามิเตอร์ใน void ฟังก์ชั่น และการรีเทิร์นค่าจะใช้คำสั่ง set () ระบุตำแหน่งอินเดกซ์ของอากูเมนต์ที่ทำหน้าที่รีเทิร์นและค่าที่ต้องการรีเทิร์น แสดงตัวอย่างรูปแบบพารามิเตอร์จาวาและรูปแบบพารามิเตอร์ดีบีทู เจนเนอรัล ในโปรแกรมที่ 2.6 และ 2.7 ตามลำดับ โดย T1 ถึง T3 เป็นชนิดข้อมูลของค่าที่รับเข้ามาในฟังก์ชั่น T4 เป็นชนิดข้อมูลของค่าที่ต้องการรีเทิร์น

โปรแกรมที่ 2.6 ฟังก์ชั่นที่ผู้ใช้กำหนดเองในภาษาจาวา ที่ใช้รูปแบบพารามิเตอร์จาวา

```
import com.ibm.db2.app.*;
public class sample {
   public static T4 test3(T1 arg1, T2 arg2, T3 arg3) { ... return
value;}
}
```

โปรแกรมที่ 2.7 ฟังก์ชั่นที่ผู้ใช้กำหนดเองในภาษาจาวา ที่ใช้รูปแบบพารามิเตอร์ดีบีทู เจนเนอรัล

```
import com.ibm.db2.app.*;
public class sample extends UDF{
   public static void test3(T1 arg1, T2 arg2, T3 arg3, T4 arg4) { ...
   set(3,value);}
}
```

2.6 ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (NoSQL)

การเพิ่มขึ้นของการใช้งานอินเตอร์เน็ตและเว็บไซต์ต่างๆในปัจจุบันทำให้ฐานข้อมูลต้องเก็บข้อมูลขนาดใหญ่และมีความหลากหลายของชนิดข้อมูลเพิ่มมากขึ้นเช่นการเก็บข้อมูลที่เป็นรูปภาพ เสียง วิดีโอ ตำแหน่งสถานที่ และอื่นๆ รวมถึงการเก็บข้อมูลที่ไม่มีโครงสร้าง ทำให้มีการคิดค้นผลิตภัณฑ์ฐานข้อมูลชนิดใหม่ เพื่อแทนที่ฐานข้อมูลเชิงสัมพันธ์แบบเดิมซึ่งไม่สะดวกในการกระจายข้อมูลไปเก็บในหลายๆแหล่ง และการเก็บข้อมูลที่มีการเปลี่ยนแปลงบ่อยและกำหนดโตรงสร้างของข้อมูลได้ยาก ซึ่งผลิตภัณฑ์ฐานข้อมูลแบบใหม่มีการเก็บข้อมูลรูปแบบต่างๆ เช่น การเก็บข้อมูลแบบออปเจ็ค, การเก็บข้อมูลแบบกราฟ, การเก็บข้อมูลแบบคู่คีย์-แวลู (key-value pair) และการเก็บข้อมูลแบบเอกสาร (document) เป็นต้น ซึ่งรวมเรียกว่าฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (NoSQL)

ตัวอย่างผลิตภัณฑ์ฐานข้อมูลสำหรับใช้งานฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ ในรูปแบบต่างๆ เช่น การเก็บข้อมูลแบบเอกสารเช่น โคชดีบี (CouchDB) และมองโกดีบี (MongoDB) การเก็บข้อมูลแบบกราฟเช่น ไฮเปอร์กราฟดีบี (HyperGraphDB), อินโฟกริช (InfoGrid) และนีโอโฟร์เจ (Neo4j) การเก็บข้อมูลแบบคู่คีย์-แวลู ได้แก่ แอชเบส (Hbase), ไดนาโม (Dynamo) และซิมเปิ้ลดีบี (SimpleDB)

2.6.1 มองโกดีบี (MongoDB)

มองโกดีบีเป็นระบบจัดการฐานข้อมูลแบบเอกสารซึ่งเป็นโอเพนซอร์ส ซึ่งถูกพัฒนาโดยบริษัทเทนเจน (10gen) ในปี ค.ศ. 2007 ด้วยภาษา ซีพลัสพลัส และมีการพัฒนามาเป็น รุ่นล่าสุดคือรุ่น 2.4 ในปี ค.ศ. 2013 ซึ่งในปัจจุบันได้มีการนำมาใช้กับผลิตภัณฑ์หรือเว็บไซต์ต่างๆอย่างแพร่หลาย เช่นเอสเอพี (SAP) ใช้มองโกดีบีเป็นส่วนประกอบหลักในแพลตฟอร์ม แอส อะ เซอร์วิส (PaaS) หรือโฟร์สแควร์ (Foursqure) ใช้มองโกดีบีเพื่อจัดการกับข้อมูลที่เกี่ยวกับพิกัดภูมิศาสตร์ [8]

2.6.1.1 แบบจำลองข้อมูล (Data model)

มองโกดีบีเก็บข้อมูลในตารางที่มีความยืดหยุ่นหรือเรียกว่าคอลเลคชั่น (collection) ซึ่งภายในจะบรรจุเอกสาร (document) ซึ่งเปรียบได้กับแต่ละแถวในฐานข้อมูลเชิงสัมพันธ์ โดยเอกสารของมองโกดีบีจะมีลักษณะการเก็บข้อมูลแบบเจสัน (JSON) เป็นคู่ฟิลล์ (field) และค่า (value) ซึ่งไม่มีการบังคับโครงสร้างภายใน เอกสารคือในแต่ละเอกสารไม่จำเป็นต้องมีฟิลล์เหมือนกัน และค่าของฟิลล์เดียวกันในแต่ละเอกสารไม่จำเป็นต้องเก็บข้อมูลชนิดเดียวกัน ซึ่งชนิดของข้อมูลสำหรับมองโกดีบีจะเป็นไปตามชนิดของไบสัน (BSON type) ดังตารางที่ 2.1

ตารางที่ 2.1 ชนิดข้อมูลที่ไบสัน (BSON) รองรับ

ชนิดของไบสัน	หมายเลข
ดับเบิ้ล	1
สตริง	2
ออปเจค	3
อาเรย์	4
ข้อมูลไบนารี่	5
ออปเจคไอดี	7
บูลีน	8
วันที่	9
ค่าว่าง	10
นิพจน์ปรกติ	11
จาวาสคริปต์	13
สัญลักษณ์	14
จาวาสคริปต์ (มีสโคป)	15
เลขจำนวนเต็ม 32 บิต	16
ประทับเวลา	17
เลขจำนวนเต็ม 64 บิต	18
คีย์ที่น้อยที่สุด	255
คีย์ที่มากที่สุด	127

ตัวอย่างการเก็บข้อมูลในเอกสาร (document) ของมองโกดีบี ในรูปที่ 2.3 แสดงการเก็บข้อมูลในเอกสารแบบเจสัน (JSON) ซึ่งประกอบไปด้วยฟิลล์และค่า [8]

```
{
  name: "sue",                    ← field: value
  age: 26,                        ← field: value
  status: "A",                    ← field: value
  groups: [ "news", "sports" ]    ← field: value
}
```


รูปที่ 2.3 ตัวอย่างเอกสาร (document) ในมองโกดีบี

ชื่อฟิลล์ในเอกสารเป็นสตริง ซึ่งไม่ประกอบด้วยช่องว่าง จุดหรือเครื่องหมายดอลลาร์ ($) โดยในหนึ่งเอกสารสามารถเก็บฟิลล์ที่มีชื่อเหมือนกันได้ และมีการสงวนฟิลล์ชื่อ _id ไว้เป็นคีย์หลัก (primary key) ซึ่งจะมีค่า ของฟิลล์ไม่ซ้ำกันในคอลเลคชั่นและสามารถเป็นชนิดข้อมูลใดก็ได้นอกเหนือจากอาเรย์ [8]

เอกสารจะมีขนาดไม่เกิน 16 เมกะไบต์ ซึ่งเป็นขนาดที่รับประกันได้ว่า 1 เอกสารจะไม่ใช้แรมหรือใช้แบนด์วิดธ์ในการส่งข้อมูลมากเกินไป โดยถ้าต้องการเก็บข้อมูลซึ่งมาขนาดใหญ่กว่า 16 เมกะไบต์สามารถทำได้โดยใช้กริดเอฟเอส (GridFS) ซึ่งเป็นระบบไฟล์ของมองโกดีบี [4]

2.7 คุณสมบัติของมองโกดีบี

คุณสมบัติของมองโกดีบีมีดังต่อไปนี้

2.7.1 Dynamic Schema

มองโกดีบีมีการเก็บข้อมูลแบบยืดหยุ่น (flexible schema) ไม่มีการบังคับโครงสร้างของเอกสาร (document) ในคอลเลคชั่น (collection) คือ ในแต่ละเอกสารไม่จำเป็นต้องมีฟิลล์ หรือโครงสร้างเหมือนกัน และในฟิลล์เดียวกันของแต่ละเอกสาร ไม่จำเป็นต้องเก็บข้อมูลชนิดเดียวกัน ทำให้ไม่มีปัญหาฟิลล์ที่มีค่าเป็นค่าว่าง (null) และเหมาะกับการเก็บข้อมูลที่ไม่มีโครงสร้าง หรือโครงสร้างมีการเปลี่ยนแปลงบ่อย [8]

ตัวอย่างของการใช้งาน คือ การเก็บข้อมูลคลังสินค้า สมมติว่ามีสินค้าอยู่ 2 ชนิดคือเพลงและภาพยนตร์ โดยมองโกดีบีสามารถเก็บข้อมูลทั้ง 2 ประเภทนี้ลงในคอลเลคชั่นเดียวกันได้ และถ้ามีสินค้าประเภทอื่นก็สามารถเก็บลงในคอลเลคชั่นนี้ได้เช่นกัน ตัวอย่างของการเก็บคือ

```
{
  sku: "00e8da9b",
  type: "Audio Album",
  title: "A Love Supreme",
  description: "by John Coltrane",
  asin: "B0000A118M",

  shipping: {
    weight: 6,
    dimensions: {
      width: 10,
      height: 10,
      depth: 1
    },
  },

  pricing: {
    list: 1200,
    retail: 1100,
    savings: 100,
    pct_savings: 8
  },

  details: {
    title: "A Love Supreme [Original Recording Reissued]",
    artist: "John Coltrane",
    genre: [ "Jazz", "General" ],
    ...
    tracks: [
      "A Love Supreme Part I: Acknowledgement",
      "A Love Supreme Part II - Resolution",
      "A Love Supreme, Part III: Pursuance",
      "A Love Supreme, Part IV-Psalm"
    ],
  },
}
```

รูปที่ 2.4 ตัวอย่างเอกสารเก็บข้อมูลเพลง

```
{
  sku: "00e8da9d",
  type: "Film",
  ...,
  asin: "B000P0J0AQ",

  shipping: { ... },

  pricing: { ... },

  details: {
    title: "The Matrix",
    director: [ "Andy Wachowski", "Larry Wachowski" ],
    writer: [ "Andy Wachowski", "Larry Wachowski" ],
    ...,
    aspect_ratio: "1.66:1"
  },
}
```

รูปที่ 2.5 ตัวอย่างการเก็บข้อมูลภาพยนตร์

2.7.2 การทำชุดสำเนา (Replica Set)

มองโกดีบีมีการเก็บชุดข้อมูลซ้ำหลายเซิร์ฟเวอร์เพื่อเป็นแบ็คอัพเป็นการสำรองข้อมูล ป้องกันการความเสี่ยงในการเสียหายของระบบและความผิดพลาดต่างๆซึ่งเป็นอันตรายต่อข้อมูล เรียกว่าการทำสำเนา โดยชุดสำเนามีการทำงานแบบนาย-บ่าว (master-slave) ซึ่งมีฐานข้อมูลหลัก ซึ่งทำตัวเป็นนาย โดยโปรแกรมประยุกต์สามารถอ่านเขียนข้อมูลในฐานข้อมูลหลักได้ จากนั้นฐานข้อมูลหลักจะคัดลอกข้อมูลให้กับฐานข้อมูลรอง ซึ่งเป็นบ่าว เพื่อเก็บข้อมูลซ้ำ โดยข้อมูลในฐานข้อมูลรองสามารถถูกอ่านได้เท่านั้นไม่สามารถเขียนข้อมูลลงในฐานข้อมูลรองได้ และหากมีฐานข้อมูลใดในชุดสำเนาได้รับความเสียหายฐานข้อมูลหลักจะทำการกู้คืนฐานข้อมูลนั้นๆ แต่ถ้าฐานข้อมูลหลักได้รับความเสียหายฐานข้อมูลรองจะทำการสื่อสารกันเพื่อเลือกตัวแทนมาเป็นฐานข้อมูลหลักตัวใหม่ เมื่อเลือกได้แล้ว ฐานข้อมูลหลักตัวใหม่จะทำการกู้คืนฐานข้อมูลหลักตัวเก่าที่ได้รับความเสียหาย [9]

2.7.3 ชาร์ดดิ้ง (Sharding)

ชาร์ดดิ้งเป็นวิธีการกระจายการเก็บข้อมูลในหลายแหล่ง ซึ่งเป็นประโยชน์ในการทำงานกับข้อมูลขนาดใหญ่หรือการรับส่งข้อมูลเป็นจำนวนมาก [10]

2.7.3.1 จุดประสงค์ของชาร์ดดิ้ง

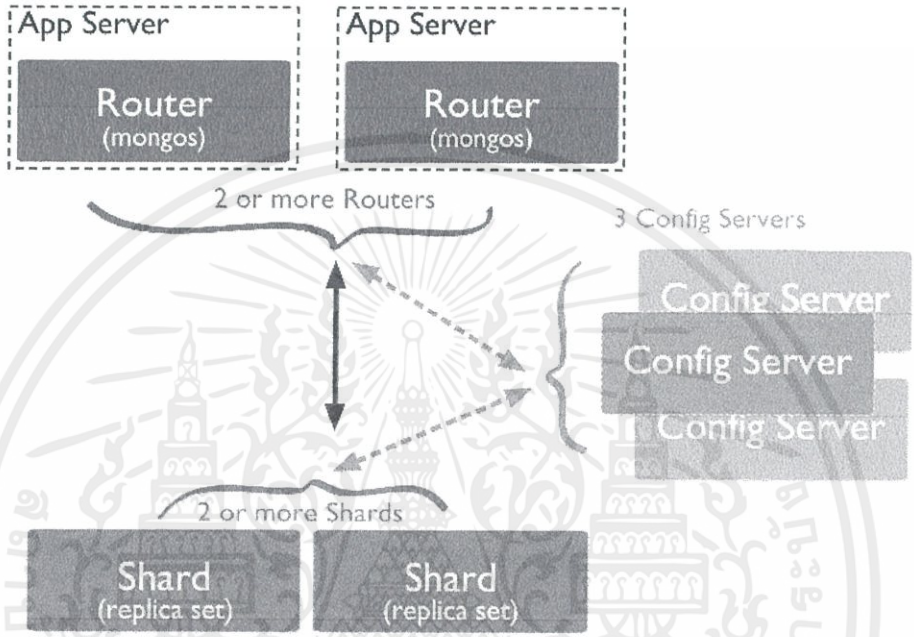
การแก้ปัญหาการทำงานกับข้อมูลขนาดใหญ่และต้องใช้ทรัพยากรจำนวนมาก มี 2 วิธีคือ

1) การทำสเกลแนวดิ่ง ซึ่งเป็นการเพิ่มความสามารถให้กับเซิร์ฟเวอร์ที่เก็บข้อมูล เช่น เพิ่มความเร็วในการประมวลผลข้อมูลและขนาดของหน่วยความจำ ซึ่งทำให้มีต้นทุนสูงและไม่คุ้มกับระบบขนาดเล็ก
2) การทำสเกลแนวราบหรือการทำชาร์ดดิ้ง ซึ่งเป็นการแบ่งและกระจายข้อมูลไปเก็บยังหลายเซิร์ฟเวอร์หรือหลายชาร์ด (shard) โดยแต่ละชาร์ดเป็นฐานข้อมูลที่ไม่ขึ้นต่อกัน หรือเรียกได้ว่าหนึ่งชาร์ดคือหนึ่งลอจิคอลฐานข้อมูล

การทำชาร์ดดิ้งช่วยลดการทำงานของแต่ละชาร์ดโดยยิ่งมีการกระจายการทำงานมากเท่าใดการทำงานของแต่ละชาร์ดจะน้อยลงเท่านั้น ซึ่งเป็นผลทำให้แต่ละชาร์ดมีพื้นที่มากขึ้นและมีความสามารถในการรับส่งข้อมูลได้ดี และช่วยลดการเก็บข้อมูลของแต่ละเซิร์ฟเวอร์ และแต่ละชาร์ดจะรับภาระการเก็บข้อมูลน้อยลงเมื่อมีการทำชาร์ดจำนวนมาก [10]

2.7.3.2 ชาร์ดดิ้งในมองโกดีบี

มองโกดีบีรองรับการทำชาร์ดดิ้งและการตั้งค่า ของ กลุ่มที่ทำชาร์ด (shard clusters) โดยกลุ่มที่ทำชาร์ด ประกอบด้วย ชาร์ด, เราเตอร์และเซิฟเวอร์ตั้งค่า [10]



รูปที่ 2.6 แผนภาพกลุ่มที่ทำชาร์ด

สำหรับองค์ประกอบที่จำเป็นในการทำชาร์ดดิ้งมีดังนี้

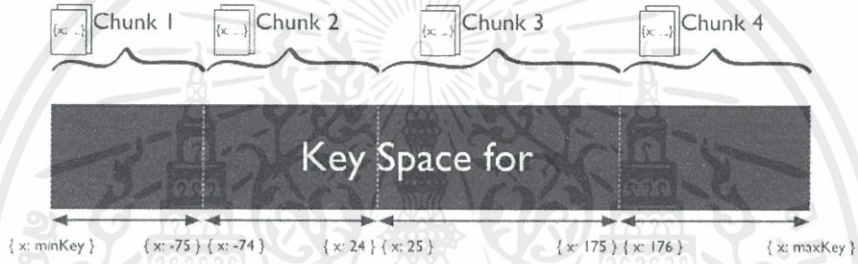
- ชาร์ดเป็นที่เก็บข้อมูล และเพื่อให้สามารถใช้งานข้อมูลได้ตลอดเวลา (High availability) และมีความสอดคล้องของข้อมูล (consistency) แต่ละชาร์ดจะต้องเป็นชุดสำเนา (Replica Set)
- เราเตอร์หรือ มองโก (mongos) เป็นตัวกลางในการเชื่อมต่อกับโปรแกรมประยุกต์ของไคลเอนต์และส่งคำสั่งการเรียดดูข้อมูลไปยังชาร์ดที่เก็บข้อมูลที่ต้องการ จากนั้นจะส่งผลกลับให้ไคลเอนต์
- เซิฟเวอร์ตั้งค่าใช้ในการเก็บข้อมูลของคลัสเตอร์ ซึ่งเป็นข้อมูลที่เราเตอร์จะใช้เป็นข้อมูลในการหาชาร์ดที่ต้องการ

2.7.3.3 การแบ่งข้อมูลของชาร์ด

มองโกดีบีมีการแบ่งข้อมูลในแต่ละคอลเลคชั่นให้กระจายไปยังชาร์ดต่าง ๆ โดยใช้ชาร์ดคีย์ซึ่งเป็นฟิลล์ หรือกลุ่มฟิลล์ ที่มีปรากฏในทุกเอกสาร ของคอลเลคชั่น โดยมองโกดีบี จะแบ่งข้อมูลตามค่าของฟิลล์ นั้นเป็นชรังค์ (chunk) และกระจายชรังค์เหล่านั้นไปยังชาร์ดต่างๆ ซึ่งมี 2 วิธีคือโดยมีวิธีแบ่งข้อมูลดังนี้ [4]

1) แบ่งโดยใช้ลำดับ (range based partitioning)

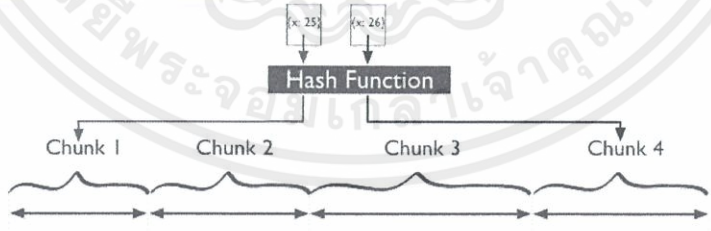
มีการแบ่งข้อมูลออกเป็นช่วงตามค่าของชาร์ดคีย์ที่เลือก ทำให้เราเตอร์ สามารถส่งการค้นหาไปยังชาร์ดที่ต้องการได้ แต่อาจมีการกระจายข้อมูลไม่เท่ากัน ทำให้บางชาร์ดต้องรับภาระข้อมูลมาก



รูปที่ 2.7 แผนภาพการแบ่งข้อมูลเป็นช่วงตามค่าของชาร์ดคีย์

2) แบ่งโดยใช้แฮช (hash base partitioning)

มีการนำค่าของชาร์ดคีย์ไปคำนวณหาค่าแฮชก่อน แล้วค่อยจัดเรียงแบ่งเป็นชรังค์ทำให้มีการกระจายข้อมูลเท่ากัน แต่การค้นหาที่เกี่ยวกับชาร์ดจะต้องทำการประมวลผลทุกชาร์ด



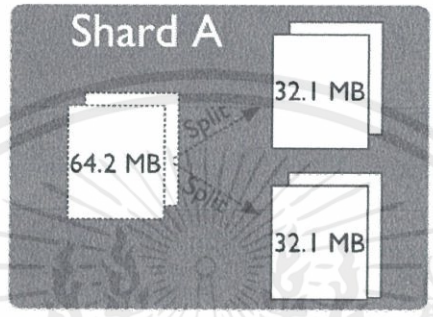
รูปที่ 2.8 แผนภาพการแบ่งข้อมูลแบบแฮช

2.7.3.4 การรักษาความสมดุลของชาร์ด

เมื่อทำการเพิ่มข้อมูล หรือการเพิ่มเซิร์ฟเวอร์ จะทำให้การกระจายข้อมูลไม่สมดุลกันใน คลัสเตอร์เช่น บางชาร์ดจะบรรจุชรังค์ (chunk) มากเกินไป หรือมีขนาดของชรังค์มากกว่าชรังค์อื่นๆมองโกดีบี จะทำการสร้างความสมดุลใหม่โดยให้กระบวนการ 2 อย่างคือ [11]

1) การแบ่ง (Splitting)

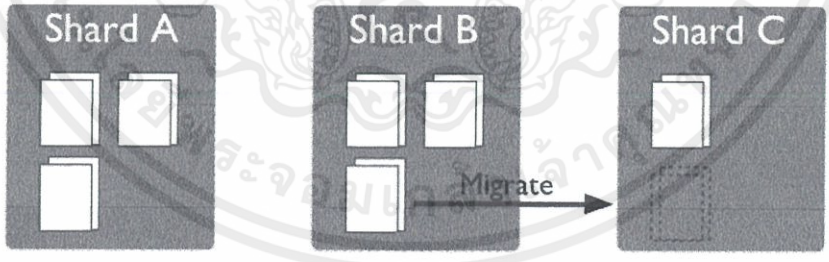
การแบ่งเป็นกระบวนการเบื้องหลังในการจัดการกับชรังค์ที่มีขนาดใหญ่เกินไป โดยมองโกดีบีจะทำการแบ่งครึ่งของชรังค์ และทำการอัพเดทข้อมูลการทำการแบ่งซึ่งการแบ่งจะทำให้เมตาดาต้าเปลี่ยนไป



รูปที่ 2.9 การรักษาความสมดุลของชาร์ดโดยวิธีการแบ่ง

2) การทำให้สมดุล (Balancer)

การทำให้สมดุลเป็นกระบวนการรักษาความสมดุลโดยจะทำการย้ายชรังค์ข้อมูลไปยังชาร์ดอื่นเมื่อมีจำนวนชรังค์ในชาร์ด มากเกินไป โดยจะใช้ค่าระดับการย้าย (migration threshold) เป็นเกณฑ์สำหรับพิจารณาว่ามีชรังค์ในชาร์ดมากเกินไปหรือไม่



รูปที่ 2.10 การรักษาความสมดุลของชาร์ดโดยวิธีการทำให้สมดุล

2.7.4 กริดเอฟเอส (กริดเอฟเอส (GridFS))

กริดเอฟเอส เป็นระบบไฟล์ซึ่งเป็น API ที่มองโกดีบีให้บริการ เพื่อเก็บไบนารี่ไฟล์ซึ่งมีขนาดใหญ่กว่าขนาดสูงสุดของเอกสารหรือ 16 เมกะไบต์ โดยกริดเอฟเอสจะนำไฟล์ข้อมูลที่ต้องการเก็บมาแบ่งเป็นชรังค์ขนาด 256 กิโลไบต์ และนำข้อมูลไปเก็บใน 2 คอลเลคชั่น ได้แก่ คอลเลคชั่นไฟล์ (files collection) ซึ่งจะเก็บข้อมูลเกี่ยวกับข้อมูลของไฟล์ (metadata) และ คอลเลคชั่นชรังค์ (chunks

collection) เก็บข้อมูลที่ถูกแบ่งไว้ในชรังค์ข้อมูล สามารถดูรายละเอียดในตารางที่ 2.2 และ 2.3 ซึ่งการนำไฟล์ข้อมูลมาแบ่งเป็นชรังค์ข้อมูลขนาดเล็กยังสามารถช่วยให้กระจายการเก็บข้อมูลบนหลายแหล่งไม่ต้องเก็บข้อมูลขนาดใหญ่ทั้งไฟล์ไว้รวมกัน สนับสนุนการทำงานแบบขนานเพื่อเพิ่มความสามารถในการส่งข้อมูล และสามารถเรียกดูข้อมูลเฉพาะชรังค์ที่ต้องการซึ่งทำให้ประหยัดพื้นที่ใช้สอยอีกด้วย [11]

ตารางที่ 2.2 ข้อมูลฟิลล์ ใน files collection

ชื่อฟิลล์	รายละเอียดข้อมูลในฟิลล์
_id	เก็บออปเจ็คไอดีซึ่งเป็นไอดีของเอกสาร (document)
length	เก็บขนาดของเอกสาร มีหน่วยเป็นไบต์
chunkSize	เก็บขนาดของแต่ละชรังค์ซึ่งโดยปกติมีขนาด 256 กิโลไบต์
uploadDate	เก็บวันที่ที่ไฟล์ถูกเก็บลงในเอกสาร (document) เป็นครั้งแรก
Md5	เก็บ MD5 hash ซึ่งได้รับจาก filemd5 API ซึ่งมีค่าเป็นสตริง
filename	เก็บชื่อของเอกสาร (document)
contentType	เก็บ MIME ของเอกสาร (document)
aliases	เก็บอาเรย์ของนามแฝง
metadata	เก็บข้อมูลอื่นๆตามที่ต้องการ

ตารางที่ 2.3 ข้อมูลฟิลล์ ใน chunks collection

ชื่อฟิลล์	รายละเอียดข้อมูลในฟิลล์
_id	เก็บสตริงของออปเจ็คไอดีของชรังค์
files_id	เก็บสตริงใน field_id ของเอกสารต้นกำเนิดซึ่งระบุในคอลเลคชั่นไฟล์
n	เก็บหมายเลขลำดับของชรังค์ซึ่งมีค่าเริ่มต้นที่ 0
data	เก็บไบนารี่ข้อมูลในชรังค์

2.7.5 การเก็บและการใช้งานข้อมูลทางภูมิศาสตร์

มองโกดีบีสามารถเก็บข้อมูลเชิงตำแหน่ง และมีฟังก์ชั่นที่เกี่ยวกับตำแหน่งรองรับด้วย สำหรับการเก็บข้อมูล จะเก็บใน 2 ลักษณะโดยจะแบ่งตามลักษณะพื้นผิวที่ใช้เก็บ ดังนี้ [12]

1) เชิงทรงกลม (Spherical) คือการเก็บข้อมูลโดยอ้างอิงพื้นผิวเป็นลักษณะทรงกลม ซึ่งจะตรงกับพื้นผิวของโลกมากกว่าอีกแบบ จะเก็บโดยใช้ออปเจ็คจีโอเจสัน (GeoJSON) ซึ่งมีโครงสร้างของฟิลล์ดังนี้

โปรแกรมที่ 2.8 โครงสร้างฟิลล์ที่ใช้เก็บข้อมูลพิกัดโดยใช้ออปเจ็คจีโอเจสัน

```
{<location field>:
            { type : "<GeoJSON type>" ,
            coordinates : <coordinates>
            }
}
```

2) เชิงราบ (Flat) คือการเก็บข้อมูลโดยอ้างอิงพื้นผิวเป็นลักษณะแบนราบ การเก็บจะเก็บพิกัดโดยใช้อาเรย์ แต่มีข้อจำกัดคือเก็บได้เพียงแค่จุดเท่านั้น

โปรแกรมที่ 2.9 โครงสร้างฟิลล์ที่ใช้เก็บข้อมูลพิกัดโดยใช้อาเรย์

```
{<location field>: [ <longitude> , <latitude> ] }
```

สำหรับหน่วยของพิกัดที่ใช้การการเก็บคือ ละติจูด (Latitude) และลองจิจูด (Longitude)

ในการเก็บข้อมูลตำแหน่งโดยใช้จีโอเจสัน ซึ่งมีชนิดข้อมูลคือจุด, เส้นตรง และรูปหลายเหลี่ยม (Polygon) โดยระบุไปในชนิดฟิลล์ของจีโอเจสันเป็น Point, LineString หรือ Polygon โดยมีตัวอย่างการใช้งานดังนี้

โปรแกรมที่ 2.10 ตัวอย่างฟิลล์ที่เก็บจุด

```
{ loc : { type : "Point" ,
coordinates : [ 40, 5 ] }
}
```

โปรแกรมที่ 2.11 ตัวอย่างฟิลล์ที่เก็บเส้นตรง

```
{ loc : { type : "LineString" ,
coordinates : [ [ 40 , 5 ] , [ 41 , 6 ] ] }
}
```

โปรแกรมที่ 2.12 ตัวอย่างฟิลล์ที่เก็บรูปหลายเหลี่ยม

```
{ loc : { type : "Polygon" ,
coordinates : [ [ [ 0 , 0 ] , [ 3 , 6 ] , [ 6 , 1 ] , [ 0 , 0 ] ] ] }
}
```

ในการสร้างอินเด็กซ์ให้กับฟิลล์ที่เก็บพิกัด เพื่อเพิ่มความเร็วในการค้นหา หรือทำให้ใช้งานฟังก์ชั่นที่เกี่ยวกับภูมิศาสตร์ได้ โดยการเก็บข้อมูลแบบจีโอเจสัน จะต้องใช้อินเด็กซ์แบบ 2dsphere ส่วนการเก็บข้อมูลเป็นอาเรย์ จะต้องใช้อินเด็กซ์เป็น 2d

โปรแกรมที่ 2.13 การสร้างอินเด็กซ์ให้กับ GeoJSON

```
db.<collection>.ensureIndex( { <location field> : "2dsphere" } )
```

โปรแกรมที่ 2.14 การสร้างอินเด็กซ์ให้กับการเก็บข้อมูลพิกัดโดยใช้อาเรย์

```
db.<collection>.ensureIndex( { <location field> : "2d" } )
```

สำหรับการใช้งานข้อมูลทางภูมิศาสตร์ก็มีฟังก์ชั่นของมองโกดีบีให้ดังนี้

1) $geoWithin เพื่อหาจุด เส้น หรือพื้นที่ ที่อยู่ในรูปทรงเรขาคณิตที่กำหนด
2) $geoIntersects เพื่อหาจุด เส้น หรือพื้นที่ ที่ตัดกับรูปทรงเรขาคณิตที่กำหนด
3) $near เพื่อหาจุด เส้น หรือพื้นที่ ที่อยู่ใกล้จุดที่กำหนด โดยคำนวณระยะห่างแบบระนาบ (flat plane)
4) $nearSphere เพื่อหาจุด เส้น หรือพื้นที่ ที่อยู่ใกล้จุดที่กำหนด โดยคำนวณระยะห่างแบบเรขาคณิตทรงกลม (spherical geometry)

สำหรับตัวอย่างการใช้งานฟังก์ชั่นทางภูมิศาสตร์ได้แสดงไว้ในหัวข้อ 4.5 ของรายงานฉบับนี้

2.8 การจัดการข้อมูลโดยมองโกดีบี

มองโกดีบีมีความสามารถในการทำคำสั่งในการอ่านและจัดการข้อมูล ได้แก่การสร้างข้อมูลในฐานข้อมูล การปรับปรุงข้อมูลในฐานข้อมูล และการลบข้อมูลออกจากฐานข้อมูล นอกจากนี้มองโกดีบี ยังให้ความสามารถในการทำฟังก์ชั่นรวมกลุ่ม (Aggregation function) เพื่อเข้าถึงข้อมูลที่ต้องทำคำสั่ง (operation) หลายขั้นตอน 2 วิธี คือไปป์ไลน์ (Pipeline Aggregation) และ แมพ-รีดิวส์ (Map-Reduce) [13]

2.8.1 การสร้าง, ปรับปรุง, อ่านและลบข้อมูล

สำหรับการสร้าง ปรับปรุง อ่านและลบข้อมูลในมองโกดีบีมีรายละเอียดดังนี้

2.8.1.1 การสร้าง (Create)

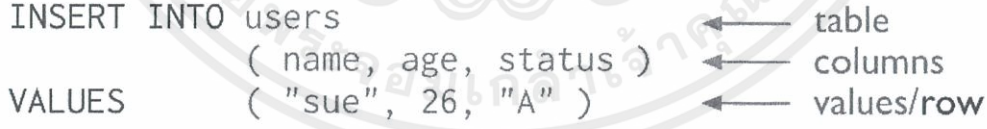
การสร้างข้อมูลเป็นความสามารถในการจัดการที่มองโกดีบีมอบให้ เพื่อใช้ในการสร้างหรือเพิ่มข้อมูลลงไปในฐานข้อมูล โดยการใส่ข้อมูลเป็นเอกสารไบสัน (BSON document) โครงสร้างในการใส่ข้อมูลดังแสดงในโปรแกรมที่ 2.15 เมื่อทำการใส่ข้อมูลลงในคอลเลคชั่น แล้ว ไคลเอนต์ ไลบรารี่ของมองโกดีบีจะทำการสร้างออปเจ็คไอดี (Object ID) ให้โดยอัตโนมัติ หากไม่มีการระบุออปเจ็คไอดีเอง ซึ่งออปเจ็คไอดีจะมีไว้ระบุตัวเอกสารและมีค่าไม่ซ้ำกันในคอลเลคชั่น [13]

โปรแกรมที่ 2.15 โครงสร้างคำสั่งการ create

```
db.collection.insert(<document>)
```

รูปที่ 2.11 และ 2.12 แสดงการเปรียบเทียบการใส่ข้อมูล ระหว่างมองโกดีบีและ ภาษาเอสคิวแอล โดยทำการใส่เอกสารที่ประกอบด้วยฟิลล์ชื่อ name, age, และ status ซึ่งมีค่าคือ “sue”, 26, และ “A” ตามลำดับ ลงในคอลเลคชั่น ชื่อ users [4]

รูปที่ 2.11 ตัวอย่างการใช้คำสั่ง insert ในมองโกดีบี



รูปที่ 2.12 ตัวอย่างการใช้คำสั่ง insert ในภาษาเอสคิวแอล

2.8.1.2 การอ่าน (Read)

การอ่านหรือการเรียกดูข้อมูล เป็นความสามารถในการค้นหาเอกสาร (document) ซึ่งสอดคล้องกับเงื่อนไขที่ต้องการ (criteria) โดยสามารถระบุฟิลล์เฉพาะที่ต้องการให้แสดงผล (projection) จำกัดจำนวนเอกสาร (document) ผลลัพธ์ และการจัดเรียงข้อมูลในขณะแสดงผลได้อีกด้วย โดยคำสั่งในการเรียกดูข้อมูลของมองโกดีบีคือคำสั่ง find ซึ่งแสดงโครงสร้างคำสั่งในโปรแกรมที่ 2.16 จะติดต่อกับอินเทอร์เฟสในการเรียกดูข้อมูลของมองโกดีบีเพื่อรับเงื่อนไขจากโปรแกรม

ประยุกต์ของผู้ใช้งาน และจะส่งผลลัพธ์เป็นเคอร์เซอร์ออกมาหาเอกสาร (document) ที่สอดคล้องเพื่อส่งให้กับโปรแกรมประยุกต์ที่มาติดต่อต่อไป [13]

โปรแกรมที่ 2.16 โครงสร้างการ query

```
db.collection.find(<query>, <projection>)
```

รูปที่ 2.13 และ 2.14 แสดงการเปรียบเทียบการเรียกดูข้อมูลระหว่างมองโกดีบีและ ภาษาเอสคิวแอล โดยทำการค้นหาเอกสาร (document) ที่ฟิลล์ age มีค่ามากกว่า 18 และแสดงผลเฉพาะชื่อฟิลล์ name และ address เรียงลำดับแบบจากน้อยไปมาก และให้ผลลัพธ์ออกมาเพียง 5 เอกสาร ซึ่งการกำหนดจำนวนเอกสาร (document) ผลลัพธ์ จะทำโดยตัวแก้เคอร์เซอร์ (cursor modifier) ซึ่งอยู่นอกคำสั่ง find

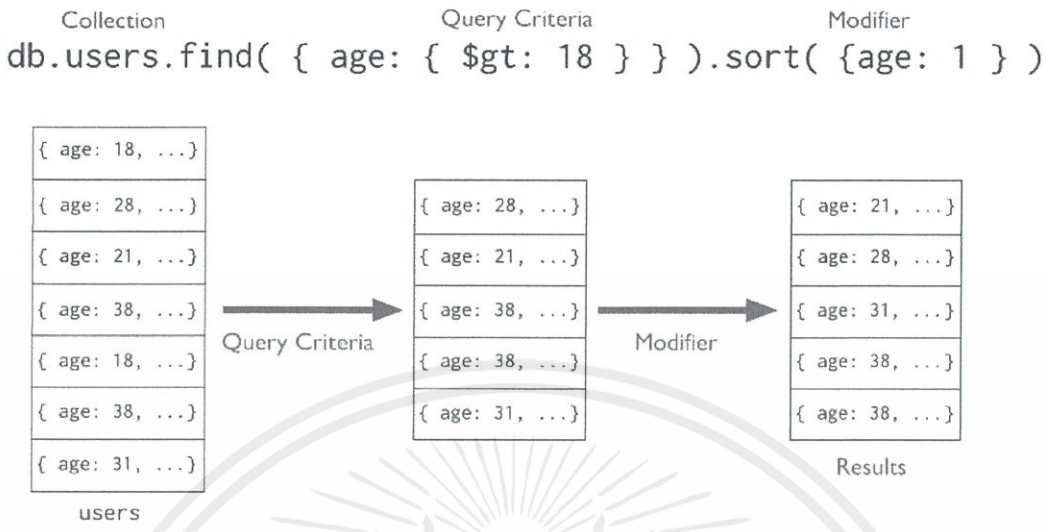
```
db.users.find(                          <-- collection
    { age: { $gt: 18 } },               <-- query criteria
    { name: 1, address: 1 }             <-- projection
).limit(5)                              <-- cursor modifier
```

รูปที่ 2.13 ตัวอย่างการใช้คำสั่ง find ในมองโกดีบี

```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier
```

รูปที่ 2.14 ตัวอย่างการเรียกดูข้อมูลในภาษาเอสคิวแอล

กระบวนการเรียกดูข้อมูลของมองโกดีบี จะมีการทำเรียงลำดับขั้นตอนโดยจะทำตามเงื่อนไขของการเรียกดูข้อมูล (criteria) ก่อน เมื่อได้ผลลัพธ์แล้วจึงนำมาปรับแต่งเพื่อแสดงผลต่อไป ดังแสดงในรูปที่ 2.15 ซึ่งแสดงขั้นตอนการทำงานของการหาเอกสาร (document) ที่ฟิลล์ age มีค่ามากกว่า 18 และเรียงลำดับการแสดงผลลัพธ์ตามฟิลล์ age จากน้อยไปมาก จะเห็นได้ว่าเงื่อนไขที่อยู่ในคำสั่ง find คือเอกสาร (document) ที่มีฟิลล์ age มีค่ามากกว่า 18 จะถูกทำก่อน และนำผลที่ได้ไปเรียงลำดับตามค่าของ age อีกที [13]



รูปที่ 2.15 แผนภาพแสดงลำดับการเรียกดูข้อมูลของมองโกดีบี

คำสั่งที่ใช้ในการเรียกดูข้อมูล (query operator) ที่มองโกดีบีมอบให้มีหลายประเภท ดังแสดงในตารางต่อไปนี้ [13]

ตารางที่ 2.4 ตารางแสดงคำสั่งในการ query สำหรับเปรียบเทียบ

คำสั่ง	คำอธิบาย
$all	ใช้สำหรับเลือกอาเรย์ซึ่งบรรจุค่าครอบคลุมค่าทั้งหมดในอาเรย์ที่เรียกดูข้อมูล
$gt	ใช้สำหรับเลือกค่าที่มีค่ามากกว่าค่าที่ใช้ในการเรียกดูข้อมูล
$gte	ใช้สำหรับเลือกค่าที่มีค่ามากกว่าหรือเท่ากับค่าที่ใช้ในการเรียกดูข้อมูล
$in	ใช้สำหรับเลือกค่าที่ปรากฏอยู่ในอาเรย์ที่ใช้ในการเรียกดูข้อมูล
$lt	ใช้สำหรับเลือกค่าที่มีค่าน้อยกว่าค่าที่ใช้ในการเรียกดูข้อมูล
$lte	ใช้สำหรับเลือกค่าที่มีค่าน้อยกว่าหรือเท่ากับค่าที่ใช้ในการเรียกดูข้อมูล
$ne	ใช้สำหรับเลือกค่าที่ไม่เท่ากับค่าที่ใช้ในการเรียกดูข้อมูล
$nin	ใช้สำหรับเลือกค่าที่ไม่ปรากฏในอาเรย์ที่ใช้ในการเรียกดูข้อมูล

สำหรับโครงสร้างและตัวอย่างคำสั่งในการเปรียบเทียบค่าเป็นดังต่อไปนี้

โปรแกรมที่ 2.17 โครงสร้างสำหรับการใช้คำสั่ง $all

```
{field: {$all: [<value>,<value1>, …]}}
```

โปรแกรมที่ 2.18 ตัวอย่างการใช้คำสั่ง $all และผลลัพธ์

```
db.inventory.find({tags: {$all: ["appliances","school","book"]}})
```

โปรแกรมที่ 2.18 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในอาเรย์ ในฟิลล์ tags ครอบคลุมอาเรย์ที่ใช้ในการเรียกดูข้อมูลคือ ["appliances" ,"school" ,"book"]

โปรแกรมที่ 2.19 โครงสร้างสำหรับการใช้คำสั่ง $gt

```
{field: {$gt: value}}
```

โปรแกรมที่ 2.20 ตัวอย่างการใช้คำสั่ง $gt และผลลัพธ์

```
db.inventory.find({qty: {$gt: 20}})
```

โปรแกรมที่ 2.20 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty มากกว่า 20

โปรแกรมที่ 2.21 โครงสร้างสำหรับการใช้คำสั่ง gte

```
{field: {$gte: value}}
```

โปรแกรมที่ 2.22 ตัวอย่างการใช้คำสั่ง $gt และผลลัพธ์

```
db.inventory.find({qty: {$gte: 20}})
```

โปรแกรมที่ 2.22 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty มากกว่าหรือเท่ากับ 20

โปรแกรมที่ 2.23 โครงสร้างสำหรับการใช้คำสั่ง $in

```
{field: {$in: [<value>,<value1>, …, <value N>]}}
```

โปรแกรมที่ 2.24 ตัวอย่างการใช้คำสั่ง $in และผลลัพธ์

```
db.inventory.find({qty: {$in: [5,15]}})
```


โปรแกรมที่ 2.24 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty เท่ากับ 5 หรือ 15 (ในกรณีนี้สามารถเลือกใช้คำสั่ง $or แทนได้)

โปรแกรมที่ 2.25 โครงสร้างสำหรับการใช้คำสั่ง $lt

```
{field: {$lt: value}}
```

โปรแกรมที่ 2.26 ตัวอย่างการใช้คำสั่ง $lt และผลลัพธ์

```
db.inventory.find({qty: {$lt: 20}})
```

โปรแกรมที่ 2.26 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty น้อยกว่า 20

โปรแกรมที่ 2.27 โครงสร้างสำหรับการใช้คำสั่ง lte

```
{field: {$lte: value}}
```

โปรแกรมที่ 2.28 ตัวอย่างการใช้คำสั่ง $lt และผลลัพธ์

```
db.inventory.find({qty: {$lte: 20}})
```

โปรแกรมที่ 2.28 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty น้อยกว่าหรือเท่ากับ 20

โปรแกรมที่ 2.29 โครงสร้างสำหรับการใช้คำสั่ง $ne

```
{field: {$ne: value}}
```

โปรแกรมที่ 2.30 ตัวอย่างการใช้คำสั่ง $lt และผลลัพธ์

```
db.inventory.find({qty: {$ne: 20}})
```

โปรแกรมที่ 2.30 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty ไม่เท่ากับ 20

โปรแกรมที่ 2.31 โครงสร้างสำหรับการใช้คำสั่ง $nin

```
{field: {$nin: [<value>,<value1>, …, <value N>]}}
```

โปรแกรมที่ 2.32 ตัวอย่างการใช้คำสั่ง $in และผลลัพธ์

```
db.inventory.find({qty: {$nin: [5,15]}})
```

โปรแกรมที่ 2.32 แสดงการเรียกดูข้อมูลเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ qty ไม่เท่ากับ 5 และไม่เท่ากับ 15 หรือ ค่าที่ไม่ปรากฏในอาเรย์ที่ใช้ในการเรียกดูข้อมูล

สำหรับคำสั่งค้นหาแบบลอจิคอล (Logical Query Operators) มีดังนี้

ตารางที่ 2.5 ตารางแสดงคำสั่งในการเรียกดูข้อมูลแบบลอจิคอล

คำสั่ง	คำอธิบาย
$or	ใช้สำหรับเชื่อมเงื่อนไขในการเรียกดูข้อมูลเพื่อเลือกผลลัพธ์ที่ตรงกับเงื่อนไขใดเงื่อนไขหนึ่งหรือทั้งหมด
$and	ใช้สำหรับเชื่อมเงื่อนไขในการเรียกดูข้อมูลเพื่อเลือกผลลัพธ์ที่ตรงกับทั้งสองเงื่อนไข
$not	ใช้สำหรับเลือกผลลัพธ์ที่ตรงข้าม หรือไม่ตรงตามเงื่อนไขในการเรียกดูข้อมูล
$nor	ใช้สำหรับเชื่อมเงื่อนไขในการเรียกดูข้อมูล เพื่อเลือกผลลัพธ์ที่ไม่ตรงกับทั้งสองเงื่อนไข

สำหรับโครงสร้างและตัวอย่างคำสั่งค้นหาแบบลอจิคอลเป็นดังต่อไปนี้

โปรแกรมที่ 2.33 โครงสร้างสำหรับการใช้คำสั่ง $or

```
{$or: [{<expression1>},{<expression2>},..., {<expressN>}]}
```

โปรแกรมที่ 2.34 ตัวอย่างการใช้คำสั่ง $or และผลลัพธ์

```
db.inventory.find({price: 1.99, $or:[{qty: {$lt: 20}},{sale: true}]})
```

โปรแกรมที่ 2.34 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ price เท่ากับ 1.99 และ มีค่าในฟิลล์ qty น้อยกว่า 20 หรือ มีค่าในฟิลล์ sale เป็นจริง

โปรแกรมที่ 2.35 โครงสร้างสำหรับการใช้คำสั่ง $and

```
{$and: [{<expression1>},{<expression2>},..., {<expressN>}]}
```

โปรแกรมที่ 2.36 ตัวอย่างการใช้คำสั่ง $and และผลลัพธ์

```
db.inventory.find({$and: [{price: 1.99},{qty: {$lt: 20}},{sale:
true}]})
```


โปรแกรมที่ 2.36 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ price เท่ากับ 1.99ค่าในฟิลล์ qty น้อยกว่า 20 และ มีค่าในฟิลล์ sale เป็นจริง

โปรแกรมที่ 2.37 โครงสร้างสำหรับการใช้คำสั่ง $not

```
{ field: {$not: {<operator-expression>}}}
```

โปรแกรมที่ 2.38 ตัวอย่างการใช้คำสั่ง $not และผลลัพธ์

```
db.inventory.find({price: {$not: {$gt: 1.99}}})
```

โปรแกรมที่ 2.38 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ price ที่มีค่าไม่เกิน 1.99 หรือไม่มีฟิลล์ price

โปรแกรมที่ 2.39 โครงสร้างสำหรับการใช้คำสั่ง $nor

```
{$nor: [{<expression1>},{<expression2>},..., {<expressN>}]}
```

โปรแกรมที่ 2.40 ตัวอย่างการใช้คำสั่ง $nor และผลลัพธ์

```
db.inventory.find({$nor: [{price: 1.99},{qty: {$lt: 20}},{sale:
true}]})
```

โปรแกรมที่ 2.40 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ price ไม่เท่ากับ 1.99 และ มีค่าในฟิลล์ qty ไม่น้อยกว่า 20 และ มีค่าในฟิลล์ sale ไม่เป็นจริง

ตารางที่ 2.6 ตารางแสดงคำสั่งในการการเรียกดูข้อมูลเกี่ยวกับฟิลล์

คำสั่ง	คำอธิบาย
$exists	ใช้สำหรับเลือกเอกสาร (document) ที่มีฟิลล์ที่ต้องการ
$mod	ทำการหารเอาเศษค่าของฟิลล์เพื่อเลือกเอกสาร (document) ที่มีผลลัพธ์ตามที่ต้องการ
$type	เลือกเอกสาร (document) ที่มีชนิดข้อมูลในฟิลล์ตามที่ต้องการ

โปรแกรมที่ 2.41 โครงสร้างสำหรับการใช้คำสั่ง $exists

```
{ field: {$exists: <boolean>}}
```

โปรแกรมที่ 2.42 ตัวอย่างการใช้คำสั่ง $exists และผลลัพธ์

```
db.inventory.find({qty: {$exists: true, $nin:[5,15]}})
```

โปรแกรมที่ 2.42 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีฟิลล์ qty และมีค่าในฟิลล์ไม่เท่ากับ 5 และไม่เท่ากับ 15

โปรแกรมที่ 2.43 โครงสร้างสำหรับการใช้คำสั่ง $mod

```
{ field: {$mod: [divisor,remainder]}}
```

โปรแกรมที่ 2.44 ตัวอย่างการใช้คำสั่ง $mod และผลลัพธ์

```
db.inventory.find({qty: {$mod: [4,0]}})
```

โปรแกรมที่ 2.44 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่ค่าของฟิลล์ qty mod 4 แล้วเท่ากับ 0 เช่นเอกสาร (document) ที่มีค่าในฟิลล์ qty เท่ากับ 0 หรือ 12

โปรแกรมที่ 2.45 โครงสร้างสำหรับการใช้คำสั่ง $type

```
{ field: {$type: <BSON type>}}
```

โปรแกรมที่ 2.46 ตัวอย่างการใช้คำสั่ง $type

```
db.inventory.find({price: {$type : 1}})
```

โปรแกรมที่ 2.46 แสดงทุกเอกสาร (document) ในคอลเลคชั่น inventory ที่มีค่าในฟิลล์ price เป็นดับเบิ้ลโดยสามารถดูชนิดข้อมูลของตัวเลขที่แทนชนิดไบสัน (BSON type) ได้จากตารางที่ 2.1

2.8.1.3 การปรับปรุง (Update)

การปรับปรุงเป็นความสามารถที่มองโกดีบีใช้ในการปรับปรุงข้อมูลต่างๆในเอกสาร (document) ซึ่งเคยถูกสร้างไว้แล้วในคอลเลคชั่น ซึ่งมองโกดีบีได้ให้ความสามารถในการค้นหาเอกสาร ที่ต้องการปรับปรุงค่า แล้วจึงทำการกำหนดค่าใหม่ให้กับเอกสาร นั้น โดยการปรับปรุงข้อมูลของเอกสาร สามารถทำได้ทั้งการเพิ่ม หรือลบฟิลล์ที่มีอยู่แล้ว รวมถึงการเปลี่ยนแปลงค่าในฟิลล์ที่มีอยู่แล้ว โดยคำสั่งที่ใช้ในการปรับปรุงค่าคือคำสั่ง update ทั้งแสดงในโปรแกรมที่ 2.47 ซึ่งประกอบพารามิเตอร์สำหรับการค้นหาเอกสาร ที่ต้องการปรับปรุง พารามิเตอร์ข้อมูลที่ต้องการปรับปรุง โดยสามารถใช้คำสั่ง $set เพื่อระบุว่าเป็นการตั้งค่าใหม่ และพารามิเตอร์สำหรับเลือกตัวเลือก [13]

จากรูปที่ 2.16 และ 2.17 แสดงการเปรียบเทียบการอัพเดทข้อมูลระหว่างมองโกดีบีและภาษาเอสคิวแอล แสดงการปรับปรุงข้อมูลของเอกสาร ที่มีค่าในฟิลล์ age มากกว่า 18 ให้มีค่าในฟิลล์ status เป็น “A” โดยทำงานมากกว่า 1 เอกสาร [13]

โปรแกรมที่ 2.47 โครงสร้างคำสั่งอัพเดต

```
db.collection.update(<query>,<update>,<options>)
```

```
db.users.update(                          <-- collection
    { age: { $gt: 18 } },                 <-- update criteria
    { $set: { status: "A" } },            <-- update action
    { $multi: true }                      <-- update option
)
```

รูปที่ 2.16 ตัวอย่างการใช้คำสั่งอัพเดตในมองโกดีบี

```
UPDATE users                <-- table
SET    status = 'A'         <-- update action
WHERE  age > 18             <-- update criteria
```

รูปที่ 2.17 ตัวอย่างการอัพเดตในภาษาเอสคิวแอล

2.8.1.4 การลบ (Delete)

การลบคือการลบเอกสารหรือกลุ่มของเอกสาร ที่ตรงกับเงื่อนไขที่กำหนด โดยจะมีการกระทำคล้ายกับการปรับปรุงข้อมูล คือมีการค้นหาเพื่อเลือกเอกสาร ที่ต้องการลบก่อน จากนั้นจะทำการลบเอกสารนั้น [13]

โปรแกรมที่ 2.48 โครงสร้างคำสั่งลบ

```
db.collection.remove(<query>,<justOne>)
```

จากรูปที่ 2.18 และ 2.19 แสดงการเปรียบเทียบการลบข้อมูลระหว่างมองโกดีบีและ ภาษาเอสคิวแอล โดยในตัวอย่างที่ใช้มองโกดีบีเป็นการลบเอกสารที่มีค่า status เป็น D ส่วนตัวอย่างที่ใช้เอสคิวแอล เป็นการลบแถวที่มี age มากกว่า 18 [4]

```
db.users.remove(          <---- collection
   { status: "D" }        <---- remove criteria
)
```

รูปที่ 2.18 ตัวอย่างการใช้คำสั่งลบในมองโกดีบี

```
DELETE FROM users         <---- table
WHERE  age > 18           <---- delete criteria
```

รูปที่ 2.19 ตัวอย่างการใช้คำสั่งลบในภาษาเอสคิวแอล

2.8.2 การทำคำสั่งรวมกลุ่ม (Aggregation) โดยมองโกดีบี

คำสั่งรวมกลุ่มประมวลผลข้อมูลและคืนค่าผลลัพธ์ โดยคำสั่งรวมกลุ่มจะจับกลุ่มค่าของหลายๆเอกสาร (document) ด้วยกัน และสามารถทำหลายคำสั่งบนข้อมูลที่จับกลุ่มไว้ ให้ได้ผลลัพธ์เดียวโดยให้ไว้สองทางเลือก คือการรวมแบบไปป์ไลน์ (pipeline aggregation) และ แมพ-รีดิวส์ (map-reduce) [14]

2.8.2.1 การรวมแบบไปป์ไลน์ (Pipeline Aggregation)

การทำไปป์ไลน์เป็นวิธีหนึ่งของการทำคำสั่งรวมกลุ่มในมองโกดีบี โดยการทำไปป์ไลน์ นั้นภายในประประกอบด้วยหลาย ๆ คำสั่งไปป์ไลน์ แล้วขั้นตอนของการทำงานจะเป็นการทำงานอย่างมีลำดับขั้นตอนตามคำสั่งไปป์ไลน์ที่ผู้ใช้เขียนขึ้น โดยมีรายละเอียดดังนี้ [15]

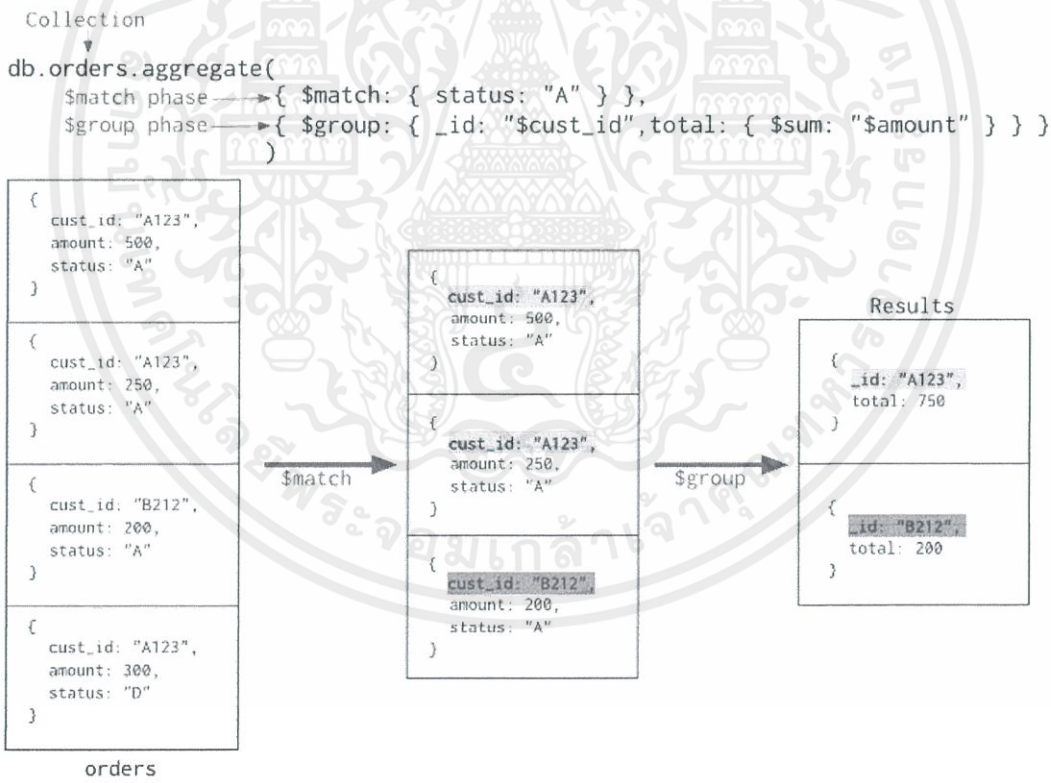
ตารางที่ 2.7 รายละเอียดของคำสั่งไปป์ไลน์

คำสั่งไปป์ไลน์	คำอธิบาย
$project	กำหนดฟิลล์ของเอกสารที่จะออกมา และสามารถเปลี่ยนชื่อเรียกของฟิลล์ที่จะออกมาได้ด้วย
$match	เป็นตัวกรองเอกสารให้ผ่านไปเฉพาะเอกสารที่ตรงกับเงื่อนไขที่อยู่ในนี้
$limit	การจำกัดจำนวนเอกสารที่ออกมา เช่น {$limit:5} จะทำให้มีเอกสารที่ออกมาเพียง 5 เอกสารแรกเท่านั้น
$skip	การข้ามเอกสารที่จะทำการรวมแบบไปป์ไลน์ เช่น {$skip:5} จะทำให้ 5 เอกสารแรกไม่ได้ทำการรวมแบบไปป์ไลน์ต่อ

ตารางที่ 2.7 (ต่อ)

คำสั่งไปป์ไลน์	คำอธิบาย
$unwind	กระจายค่าที่อยู่ในอาเรย์ให้ออกมาเป็นค่าเดี่ยว โดยจะได้เป็นเอกสารหลายเอกสาร
$group	รวมกลุ่มเอกสารจากค่าในชื่อฟิลล์ที่กำหนดเพื่อการคำนวณค่าจากเอกสาร
$sort	เรียงลำดับเอกสารแต่ละเอกสาร โดยใช้ค่าในฟิลล์ที่เลือกมาเป็นเกณฑ์ในการจัด
$geonear	เรียงลำดับเอกสารแต่ละเอกสาร โดยใช้ระยะห่างจากจุดพิกัดทางภูมิศาสตร์มาเป็นเกณฑ์ในการจัด

ตัวอย่างการทำการรวมแบบไปป์ไลน์ [15]



รูปที่ 2.20 แผนภาพการทำงานของการรวมแบบไปป์ไลน์

จากรูปแผนภาพตัวอย่างมีวัตถุประสงค์เพื่อ รวมค่า amount ของ cust_id แต่ละคนที่มี status เป็น A โดยที่ใช้คำสั่งไปป์ไลน์คือ $match และ $group ตามลำดับ เมื่อเริ่มการทำการรวม

แบบไปป์ไลน์เอกสารทั้งหมดในคอลเลคชั่น orders จะเข้าไปทำ $match เพื่อกรองเอกสารให้เหลือเฉพาะเอกสารที่มี status เป็น A จากนั้นจะเข้าไปทำ $group เมื่อจัดกลุ่มด้วยค่าในฟิลล์ ชื่อ cust_id ซึ่งจะต้องระบุในฟิลล์ _id ของ $group แล้วทำการรวมค่า amount สำหรับ cust_id แต่ละคน ด้วยคำสั่ง { $sum: "$amount"} แล้วเก็บค่าดังกล่าวในฟิลล์ชื่อ total จากนั้นจึงได้ผลลัพธ์ของการทำการรวมแบบไปป์ไลน์ออกมา

2.8.2.2 แมพ-รีดิวส์ (Map-Reduce)

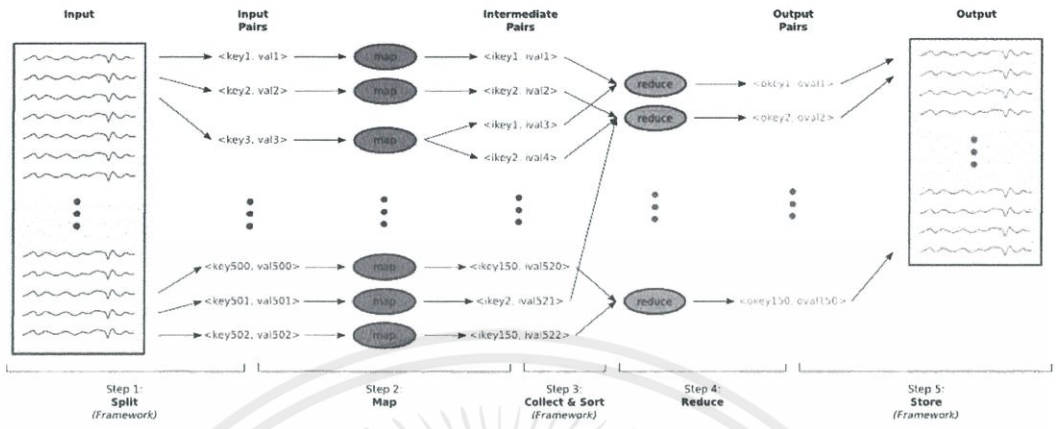
แมพ-รีดิวส์เป็น โปรแกรมมิ่ง โมเดล (programming model) สำหรับการเขียนโปรแกรมให้สามารถประมวลผลแบบขนานได้ง่าย การประมวลจะรับคู่คีย์-แวลู (key-value pair) แล้วผ่านเข้าไปยัง ฟังก์ชั่น 2 ฟังก์ชั่น คือแมพและรีดิวส์ แล้วจะได้ผลลัพธ์ออกมาเป็นคู่คีย์-แวลู [17]

ฟังก์ชั่นแมพ (Map function) ซึ่งเขียนโดยโปรแกรมเมอร์ จะประมวลผลคู่คีย์-แวลู เพื่อให้ได้ คู่คีย์-แวลูขั้นกลาง (intermediate key-value pairs) โดยจะมีการจัดกลุ่มที่มีค่าของคีย์เหมือนกันมาอยู่รวมกัน

ฟังก์ชั่นรีดิวส์ (Reduce function) ซึ่งเขียนโดยโปรแกรมเมอร์ จะประมวลผลคีย์ขั้นกลางจากเซตของค่าต่างๆ ของคีย์นั้น เพื่อทำการรวม ให้เหลือค่าที่เป็นผลลัพธ์เพียงหนึ่ง หรือศูนย์ค่า

ขั้นตอนในการทำแมพ-รีดิวส์เป็นดังนี้ [16]

1. ข้อมูลที่รับมาเช่นไฟล์ข้อความ จะถูกแบ่งไปเป็นคู่คีย์-แวลู ซึ่งคู่คีย์-แวลูเหล่านั้นจะถูกนำเข้าสู่ตัวทำแมพ (mapper) ซึ่งเป็นหน้าที่ของกรอบการทำงานแมพ-รีดิวส์
2. ตัวทำแมพประมวลผลคู่คีย์-แวลูแต่ละคู่ แล้วได้ผลลัพธ์ออกมาเป็นคู่คีย์-แวลูขั้นกลาง
3. คู่คีย์-แวลูขั้นกลางจะถูกเก็บ และจัดเรียงลำดับ และจัดกลุ่มด้วยค่าคีย์ ซึ่งเป็นหน้าที่ของกรอบการทำงานแมพ-รีดิวส์
4. ตัวทำรีดิวส์ (reducer) จะรับคีย์และรายการของแวลูที่เชื่อมโยงกับคีย์นั้น ไปประมวลผล เช่นทำการรวม, การนับ หรือการคิดค่าเฉลี่ย เป็นต้น และได้ผลลัพธ์ออกมาเป็นคู่คีย์-แวลูที่เป็นผลลัพธ์
5. กรอบการทำงาน (framework) จะรวบรวมคู่คีย์-แวลูเหล่านั้น และเก็บลงในไฟล์ผลลัพธ์



รูปที่ 2.21 แสดงขั้นตอนการทำแมพ-รีดิวส์

จากภาพจะเห็นว่า โมเดลนี้เหมาะสมสำหรับการประมวลผลแบบขนาน เนื่องจากขั้นตอนแมพและรีดิวส์สามารถแยกกันประมวลผลได้ โดยการแบ่งข้อมูลไปประมวลผลยังแต่ละโปรเซสเซอร์หรือแต่ละเครื่องได้

สำหรับมองโกดีบีก็มีกรอบการทำงานแมพ-รีดิวส์ให้ใช้งานในการทำการรวมกลุ่ม โดยมีคำสั่งดังนี้ [18]

โปรแกรม 2.49 โครงสร้างคำสั่ง แมพ-รีดิวส์ ในมองโกดีบี

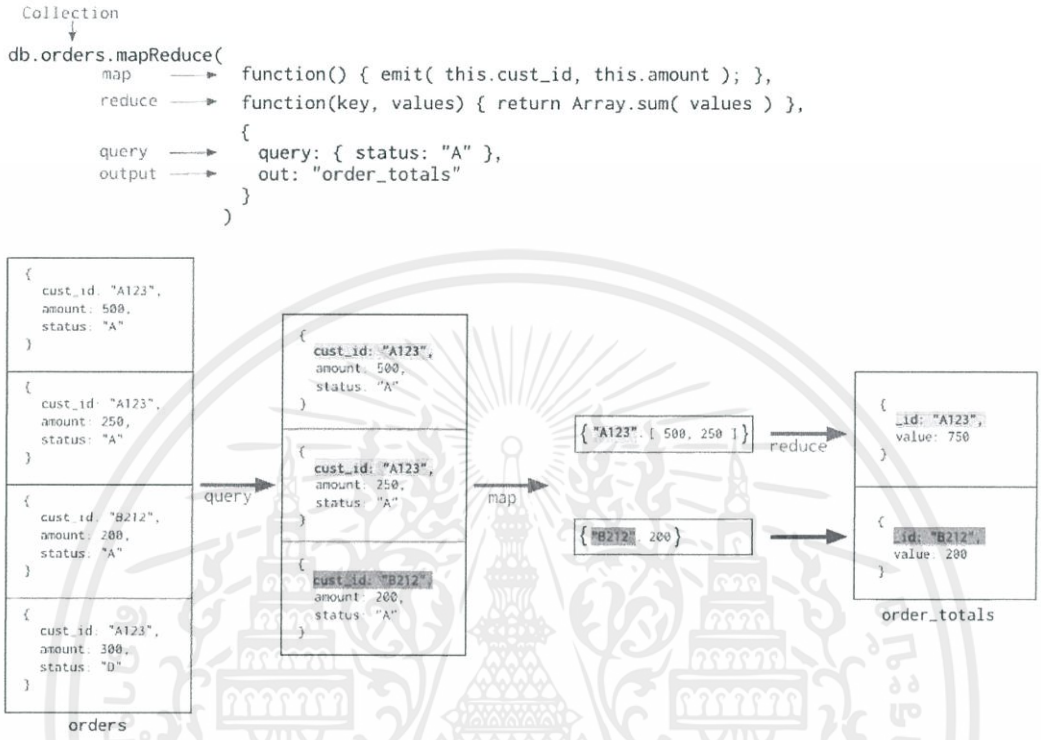
```
db.<collection name>.mapReduce(
      <map>,
      <reduce>,
      {
            out: <collection>,
            query: <document>,
            sort: <document>,
            limit: <number>,
            finalize: <function>,
            scope: <document>,
            jsMode: <boolean>,
            verbose: <boolean>
      }
)
```

พารามิเตอร์แต่ละค่ามีคำอธิบายดังนี้

ตารางที่ 2.8 ตารางอธิบายตัวแปรที่อยู่ในคำสั่ง แมพ-รีดิวส์ ในมองโกดีบี

ตัวแปร	ประเภท	คำอธิบาย
map	ฟังก์ชั่นจาวาสคริปต์	ฟังค์ชั่นภาษาจาวาสคริปต์ หรือ ชื่อฟังค์ชั่นภาษาจาวาสคริปต์ ที่ทำหน้าที่เป็นแมพฟังก์ชั่น
reduce	ฟังก์ชั่นจาวาสคริปต์	ฟังค์ชั่นภาษาจาวาสคริปต์ หรือ ชื่อฟังค์ชั่นภาษาจาวาสคริปต์ ที่ทำหน้าที่เป็นรีดิวส์ฟังก์ชั่น
out	สตริง	บอกว่าผลลัพธ์หลังจากทำ แมพ-รีดิวส์ แล้วจะเก็บไว้ในคอลเลคชั่น ใด ถ้าไม่ต้องการเก็บในคอลเลคชั่น ให้ระบุเป็น {out: {inline:1}}
query	เอกสาร	เพื่อเลือกเอกสารที่จะเข้าไปทำแมพ-รีดิวส์
Sort	เอกสาร	เพื่อเรียงลำดับค่าของฟิลล์ที่เลือกไว้
Limit	ตัวเลข	เพื่อจำกัดจำนวนเอกสารที่จะเข้าไปทำแมพ-รีดิวส์
finalize	ฟังก์ชั่นจาวาสคริปต์	ฟังค์ชั่นภาษาจาวาสคริปต์ หรือ ชื่อฟังค์ชั่นภาษาจาวาสคริปต์ ที่จะใช้ปรับแต่งผลลัพธ์หลังจากทำรีดิวส์ฟังก์ชั่นเสร็จ
scope	เอกสาร	ใส่ชื่อฟังก์ชั่นอื่น ๆ ที่ แมพ, รีดิวส์ และไฟนอลไลซ์ ฟังก์ชั่น เรียกใช้เพื่อให้ มองโกดีบีรู้จักฟังก์ชั่นนั้น
jsMode	บูลีน	กำหนดว่า หลังจากทำฟังก์ชั่นแมพ แล้วจะแปลงออปเจ็คจาวาสคริปต์มาเป็นออปเจ็คไบสันหรือไม่ ถ้าค่าเป็นเท็จจะแปลง โดยออปเจ็คไบสันที่สร้างขึ้นชั่วคราวจะถูกเก็บในหน่วยความจำ ทำให้แมพ-รีดิวส์สามารถประมวลผลข้อมูลขนาดใหญ่ได้
Verbose	บูลีน	ระบุว่าต้องการให้ผลลัพธ์มีข้อมูลเกี่ยวกับเวลาที่ใช้ในแต่ละขั้นตอนหรือไม่

เมื่อเริ่มทำคำสั่งแมพ-รีดิวส์ จะมีกระบวนการทำงานดังรูป [19]



รูปที่ 2.22 กระบวนการทำแมพ-รีดิวส์ในมองโกดีบี

จากรูปที่ 2.22 แต่ละเอกสารในคอลเลคชั่น order จะถูกคัดเหลือเฉพาะเอกสารที่มีค่า status เป็น A ซึ่งระบุในพารามิเตอร์ query จากนั้น แต่ละเอกสารที่ผ่านเข้ามา จะไปประมวลผลในฟังก์ชั่นแมพ แล้วออกมาเป็นคู่คีย์-แวลูขั้นกลาง โดยทำสั่ง emit (ในรูปตัวอย่าง จะได้ คีย์คือ cust_id และแวลูคือ amount) จากนั้นมองโกดีบีจะรวบรวมคู่ต่างๆที่มีคีย์เหมือนกันมาไว้ด้วยกัน แล้วนำไปประมวลผลในฟังก์ชั่นรีดิวส์ (ในรูปตัวอย่างคือการรวมค่า) จากนั้นนำข้อมูลไปเก็บในคอลเลคชั่นที่ระบุไว้ในพารามิเตอร์ out นั่นคือ order_totals

ในการทำแมพ-รีดิวส์ในมองโกดีบีสามารถประมวลผลแบบขนานได้ เมื่อมีการทำชาร์ดดิ้ง (Sharding) บนคอลเลคชั่นไว้ก่อนแล้ว โดยจะแบ่งการทำฟังก์ชั่นแมพ แยกไปยังชาร์ดต่าง ๆ ที่ทำไว้

ถ้ามีการทำชาร์ดดิ้ง บนคอลเลคชั่นที่เก็บผลลัพธ์มองโกดีบีก็จะกระจายงานการทำฟังก์ชั่นรีดิวส์ และฟังก์ชั่นไฟนอลไลซ์ไปยังชาร์ดต่าง ๆ ที่ทำไว้ได้

นอกจากนี้หลังจากทำแมพ-รีดิวส์ จะมีข้อมูลบอกรายละเอียดจำนวนเอกสาร หรือคู่คีย์-แวลู ที่ถูกประมวลผลในขั้นตอนต่าง ๆ ของแมพ-รีดิวส์ และเวลาที่ใช้ ดังรูป

```
    "timeMillis" : 3,
    "counts" : {
            "input" : 39,
            "emit" : 60,
            "reduce" : 10,
            "output" : 23
    },
    "ok" : 1,
}
```

รูปที่ 2.23 fields ที่บอกข้อมูลของการทำแมพ-รีดิวส์

ข้อมูลดังกล่าวมีรายละเอียดคือ

- timeMillis คือ เวลาที่ใช้ ในหน่วยมิลลิวินาที
- counts
  - input คือ จำนวนเอกสารที่เข้ามาในฟังก์ชั่นแมพ
  - emit คือ จำนวนคู่คีย์-แวลูที่ออกมาจากฟังก์ชั่นแมพ
  - reduce คือ จำนวนคู่คีย์-แวลูที่เข้าฟังก์ชั่นรีดิวส์ โดยคู่คีย์-แวลูที่ผ่านฟังก์ชั่นแมพมาแล้วมีคีย์ซ้ำกัน จะต้องเข้าฟังก์ชั่นรีดิวส์
  - output คือ จำนวนเอกสารที่ออกมาเป็นผลลัพธ์จากแมพ-รีดิวส์

2.9 การรู้จำใบหน้า (Face recognition)

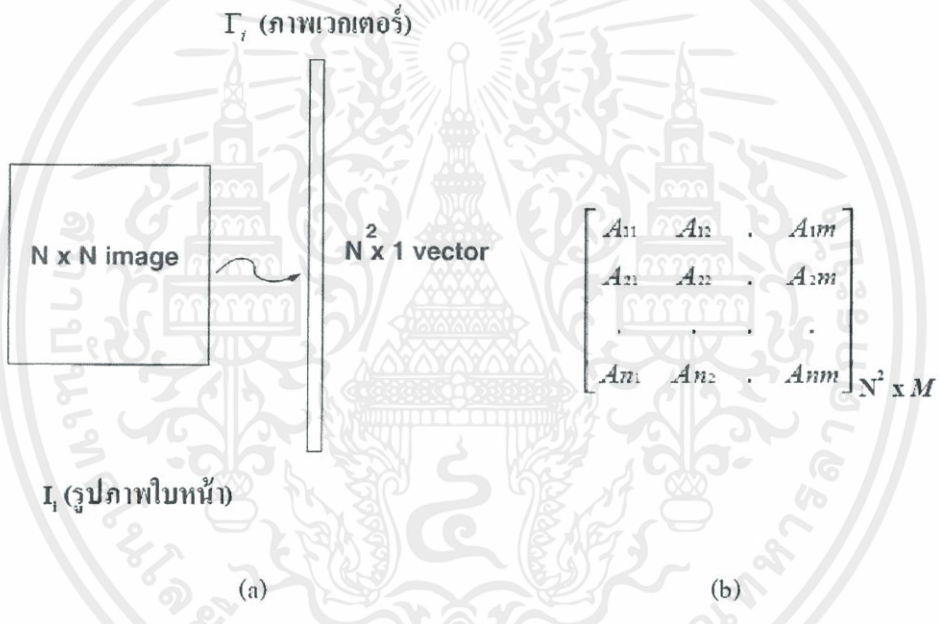
การรู้จำใบหน้าเป็นการนำเทคโนโลยีสำหรับการระบุตัวบุคคลด้วยข้อมูลภาพทางชีวมิติ (Biometric) มาใช้เป็นข้อมูลทางคอมพิวเตอร์ในการสร้างระบบการรู้จำใบหน้า โดยนำรูปภาพซึ่งรับเข้ามาผ่านกรรมวิธีหรืออัลกอริธึมในการสร้างแม่แบบซึ่งมีหลายวิธี แล้วนำไปเปรียบเทียบกับข้อมูลในฐานข้อมูล ทำให้การระบุตัวบุคคลมีความสะดวกมากขึ้น ลดความผิดพลาดในการจำข้อมูลเพื่อยืนยันตัวบุคคล ซึ่งในปัจจุบันได้มีการนำมาใช้ในงานยืนยันประวัติอาชญากร การยืนยันตัวตนแทนรหัสผ่าน [20]

2.9.1 เทคนิคการวิเคราะห์องค์ประกอบหลัก (Principal Component Analysis: PCA)

เทคนิคการวิเคราะห์องค์ประกอบหลักเป็นวิธีหนึ่งซึ่งสามารถนำมาประยุกต์ใช้กับการรู้จำใบหน้าได้ โดยใช้หลักการทางสถิติคำนวณค่าความแปรปรวนร่วมจากรูปภาพ (Covariance Matrix)

เพื่อนำไปสร้างภาพใบหน้าไอเกน (Eigenfaces) เพื่อลดมิติข้อมูลที่จะนำไปใช้ในการเปรียบเทียบความคล้ายของใบหน้าของภาพที่รับเข้ามาและภาพในฐานข้อมูล โดยวิธีนี้ถูกนำมาสร้างเป็นภาพใบหน้าไอเกน (Eigen face) ครั้งแรกโดย Sirovich และ Kirby ในปี 1987 และต่อมา Turk และ Pentland ได้สร้างระบบรู้จำใบหน้าโดยการนำภาพใบหน้าไอเกนมาใช้ [21]

สำหรับการหาใบหน้าไอเกนจะเริ่มที่การแปลงภาพที่รับเข้ามาเป็นภาพสีเทา (Gray scale) นำมาเก็บเป็นเวกเตอร์ 1 มิติ แล้วทำการ normalize จากนั้นนำเวกเตอร์ของทุกๆรูปในฐานข้อมูลมาจัดให้อยู่ในเมตริกซ์ โดยให้เวกเตอร์ของภาพที่ 1 อยู่ในคอลัมน์ที่ 1 เวกเตอร์ของชุดข้อมูลที่ 2 อยู่ในคอลัมน์ที่ 2 ไปเรื่อยๆ ถ้ามี M ภาพ จะได้เมตริกซ์ข้อมูล N^2xM เมื่อ N เป็นมิติของภาพ ดังรูปที่ 2.24



รูปที่ 2.24 การรับข้อมูลภาพมาเก็บเป็นเวกเตอร์
(a) แสดงการแปลงภาพจากเมตริกซ์เป็นเวกเตอร์
(b) เมตริกซ์ภาพขนาด n^2xm

จากนั้นหาค่าเฉลี่ยภาพใบหน้าแต่ละตำแหน่ง

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \qquad (2.1)$$

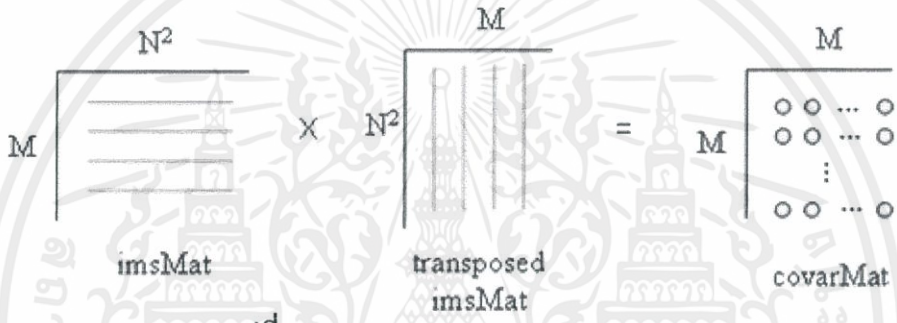
โดยที่ M เป็นจำนวนภาพในฐานข้อมูล Γ_i เป็นเวกเตอร์ภาพที่ i แล้วหาค่ากลาง

$$\Phi_i = \Gamma_i - \Psi \tag{2.2}$$

จากนั้นหาค่ากลางมาหาเมตริกซ์ความแปรปรวนร่วม (Covariance Matrix)

$$C = \frac{1}{M}\sum_1^M \Phi_i \Phi_i^r = AA^T \tag{2.3}$$

ดังรูปที่ 2.25 [21]



รูปที่ 2.25 การหา Covariance Matrix

จากนั้นหาค่าไอเกนเวกเตอร์ (υ) และค่าไอเกน (μ) ของเมตริกซ์ความแปรปรวนร่วม เนื่องจากเมตริกซ์ C มีขนาด N^2 x N^2 ซึ่งหากนำมาสร้างเป็นไอเกนเวกเตอร์จะมีขนาดใหญ่มาก จึงมีการลดมิติข้อมูลโดยใช้สร้างเมตริกซ์ A^TA แทน AA^T ได้ตามสมการดังนี้

$$A^T A\, v_i = \mu_i v_i \tag{2.4}$$

เมื่อคูณ Aทั้ง 2 ข้างของสมการจะได้

$$AA^T A\, v_i = \mu_i A v_i \tag{2.5}$$

แล้วแทน AA^T ด้วย C จะได้

$$CA\, v_i = \mu_i A v_i \tag{2.6}$$

จะเห็นได้ว่าสามารถแทน $C = \mu_i$ ได้ และเพื่อเป็นการลดขนาดข้อมูลที่ใช้ในการคำนวณ จึงใช้นำไอเกนเวกเตอร์มาใช้แทน โดยนำไอเกนเวกเตอร์ที่ได้มาเรียงลำดับตามค่าไอเกนจากมากไปหาน้อย จากนั้นนำไอเกนเวกเตอร์มาสร้างไอเกนใบหน้าโดยใช้ไอเกนเวกเตอร์ทรานสโพสต์คูณกับเมตริกซ์รูปภาพในฐานข้อมูล ดังรูปที่ 2.26 [21]

รูปที่ 2.26 คำนวณไอเกนใบหน้าจากไอเกนเวกเตอร์และเมตริกซ์ภาพ

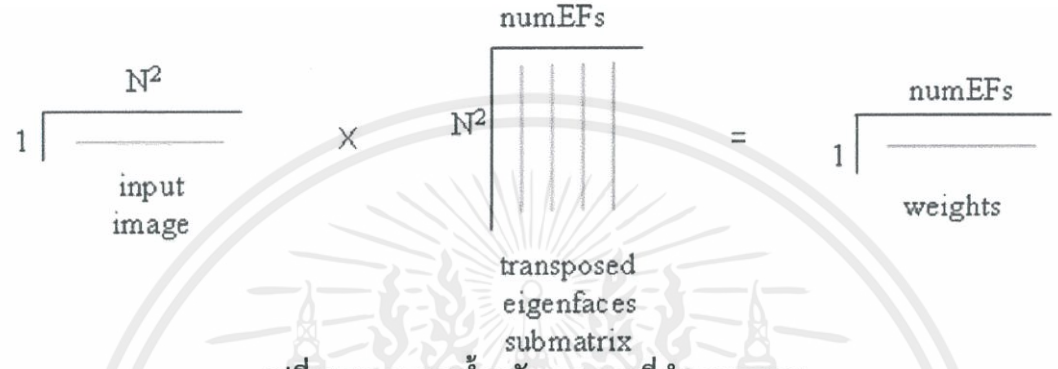
จากนั้นเวกเตอร์ในแต่ละแถวของเมตริกซ์ไอเกนใบหน้าที่ได้ จะเป็นไอเกนใบหน้าแต่ละหน้า ดังรูปที่ 2.27

รูปที่ 2.27 ไอเกนใบหน้าที่ได้จากการคำนวณ

การเทียบความคล้ายของภาพใบหน้าว่าเป็นใบหน้าของคนๆใดในฐานข้อมูลทำได้โดยการหาระยะทางแบบยุคลิด (Euclidian distance) ระหว่างน้ำหนักของภาพที่รับเข้ามากับน้ำหนักของภาพในฐานข้อมูล โดยภาพของใบหน้าที่เป็นคนเดียวกันจะมีค่าระยะทางแบบยุคลิดต่ำที่สุด วิธีการในการเปรียบเทียบเริ่มจากรับภาพใบหน้าที่ต้องการเปรียบเทียบเข้ามาแปลงเป็นภาพสีเทาจากนั้นเก็บไว้เป็นเวกเตอร์ 1 มิติแล้วทำการนอมอลไลซ์ (normalize) แล้วหาค่ากลางโดยทำค่าเฉลี่ยมาหักออกจากเวกเตอร์ภาพทดสอบ

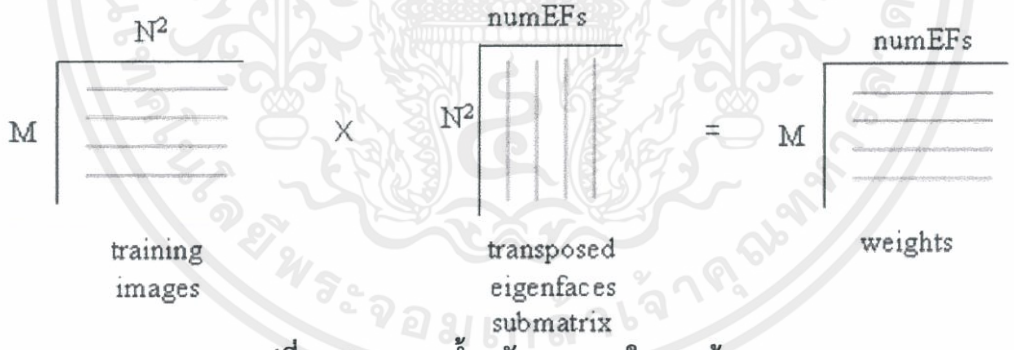
$$\Phi_{test} = \Gamma_{test} - \Psi \tag{2.7}$$

แล้วเลือกจำนวนไอเกนใบหน้าจากฐานข้อมูลที่จะนำมาคำนวณน้ำหนักโดยไม่ต้องใช้ไอเกนเวกเตอร์ทั้งหมด ทั้งนี้จำนวนไอเกนใบหน้าที่เลือกจะมีผลต่อความแม่นยำในการวัดความคล้าย โดยการหาน้ำหนักจากภาพทดสอบจะนำเวกเตอร์ภาพทดสอบคูณกับเมตริกซ์ไอเกนเวกเตอร์ที่เลือกมาทรานสโพสต์ดังรูปที่ 2.28 [22]



รูปที่ 2.28 การหาน้ำหนักจากภาพที่นำมาทดสอบ

และนำเมตริกซ์เวกเตอร์ภาพในฐานข้อมูลคูณกับเมตริกซ์ไอเกนเวกเตอร์ที่เลือกมาทรานสโพสต์เช่นกัน เพื่อหาค่าน้ำหนักของภาพในฐานข้อมูลดังรูปที่ 2.29



รูปที่ 2.29 การหาน้ำหนักจากภาพในฐานข้อมูล

จากนั้นหาผลต่างระหว่างน้ำหนักจากภาพที่นำมาทดสอบและภาพในฐานข้อมูล และเลือกภาพมีผลต่างน้ำหนักน้อยที่สุดจากฐานข้อมูลเป็นผลลัพธ์ [22]

บทที่ 3
การออกแบบ และพัฒนา

3.1 รายละเอียดของโครงงาน

ตามจุดประสงค์ของโครงงานนี้เป็นการพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ ซึ่งจะใช้ระบบการจัดการฐานข้อมูลที่เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (relational database management system) และระบบจัดการฐานข้อมูลที่ไม่เชิงสัมพันธ์ โดยโครงงานนี้เลือกใช้ดีบีทูและมองโกดีบี ดังนั้นเพื่อที่จะพัฒนาโปรแกรมประยุกต์ที่ใช้ระบบการจัดการฐานข้อมูลทั้ง 2 ชนิดนี้ จึงต้องเรียนรู้ และทำการทดลองใช้งานเพื่อหาความสามารถต่าง ๆ ของระบบการจัดการฐานข้อมูลทั้ง 2 ชนิดโดยเฉพาะมองโกดีบีว่ามีการใช้คำสั่งการจัดการข้อมูลเช่นการ เก็บข้อมูล การค้นหาข้อมูลทำอย่างไร ความสามารถในการค้นหาเมื่อเปรียบเทียบกับภาษาเอสคิวแอลแล้ว สามารถทำได้ทัดเทียมกันหรือไม่ รวมทั้งการใช้ความสามารถอื่น ๆ ที่น่าสนใจของมองโกดีบีเช่น แมพ-รีดิวส์, กริดเอฟเอส และ ฟังก์ชั่นที่เกี่ยวกับภูมิศาสตร์ เพื่อให้สามารถดึงประสิทธิภาพการใช้งานมองโกดีบี ที่จะไปทำงานร่วมกับดีบีทูให้เกิดประโยชน์เพิ่มขึ้นได้

อีกทั้งทดลองใช้ระบบควบคุมการประมวลผลทรานแซกชั่นเพื่อควบคุมความถูกต้องของทรานแซกชั่นแบบกระจายและการเขียนฟังก์ชั่นที่ผู้ใช้กำหนดเองในดีบีทูเพื่อสร้างการรู้จำใบหน้า เพื่อนำความรู้จากการทดลองมาใช้ประโยชน์ในระบบงานทะเบียนนักศึกษาจำลอง

การใช้งานระบบคลาวด์คอมพิวติ้ง (Cloud computing) ซึ่งในโครงงานนี้เลือกใช้อเมซอน อีซีทู ก็ต้องมีการศึกษา และทดลองว่าจะต้องมีการเตรียมเซิฟเวอร์จำลองที่จะติดตั้งระบบการจัดการฐานข้อมูลทั้งมองโกดีบีและดีบีทูอย่างไร การติดต่อไปยังเซิฟเวอร์นั้นทำอย่างไร และมีโปรแกรมใดที่ช่วยในการติดต่อบ้าง

3.2 เซิฟเวอร์จำลองบนอเมซอน อีซีทู

จากการศึกษาบริการต่าง ๆ ของอเมซอน เว็บ เซอร์วิส (Amazon Web Service) พบว่าบริการ อเมซอน อีซีทูเป็นบริการสำหรับการสร้างเซิฟเวอร์จำลองบนคลาวด์ ซึ่งเป็นการให้บริการประเภท อินฟราสตรัคเจอร์ แอส อะ เซอร์วิส (Infrastructure as a service) โดยผู้ใช้สามารถเลือกจัดสรรทรัพยากรให้กับเซิฟเวอร์จำลองด้วยตนเอง เช่น ระบบปฏิบัติการ, หน่วยการประมวลผล, หน่วยความจำ เป็นต้น หรือผู้ใช้สามารถเลือกอเมซอน แมคชีน อิมเมจ (Amazon Machine Image: AMI) ซึ่งจะมีซอฟต์แวร์เพิ่มเติมมาให้

สำหรับระบบการรักษาความปลอดภัยของอเมซอน อีซีทู เพื่อไม่ให้ผู้อื่นสามารถเข้าใช้งานเซิฟเวอร์จำลองที่เราสร้าง คือ การใช้ระบบการเข้ารหัสแบบคีย์สาธารณะ (public–key cryptography) โดยมีการทำงานคือ เมื่อสร้างเซิฟเวอร์จำลองขึ้นมา จะต้องสร้างกุญแจสำหรับเซิฟเวอร์นั้นด้วย โดยอเมซอน อีซีทู จะเก็บกุญแจสาธารณะ (public key) และผู้ใช้จะเก็บ กุญแจส่วนตัว (public key) ทั้งนี้การเข้ารหัสจะใช้กับข้อมูลที่เกี่ยวกับการเข้าใช้ (login information) เช่นรหัสผ่านเท่านั้น

นอกจากนี้เรายังสามารถกำหนดพอร์ต (port) และหมายเลขไอพี (IP address) ที่อนุญาตให้ติดต่อกับเซิฟเวอร์ได้

สำหรับการเชื่อมต่อไปยังเซิฟเวอร์จำลองที่สร้างขึ้น ผู้ใช้สามารถใช้โปรแกรมประเภทซีเคียวเชลล์ ไคลแอนท์ เช่น โปรแกรมพัตตี้ (PuTTY) หรือเลือกใช้จาวา ซีเคียวเชลล์ ไคลแอนท์ (Java SSH Client) ของอเมซอนได้โดยการเรียกใช้ผ่านเบราว์เซอร์ได้เลย โดยผู้ใช้ต้องระบุกุญแจส่วนตัวสำหรับการใช้เซิฟเวอร์นั้นทุกครั้งที่เริ่มการเชื่อมต่อ การเชื่อมต่อทั้ง 2 วิธีนี้ใช้ ซีเคียวเชล โพรโตคอล (Secure Shell Protocol) ในการเชื่อมต่อเหมือนกัน

รูปที่ 3.1 การเชื่อมต่อไปยังเซิฟเวอร์โดยใช้จาวา ซีเคียวเชลล์ ไคลแอนท์

3.2.1 การติดตั้งมองโกดีบี

การติดตั้งมองโกดีบีจะเริ่มจากการสร้างเซิฟเวอร์จำลองโดยเลือกใช้ระบบปฏิบัติการที่ มองโกดีบีรองรับ และเลือกใช้สถาปัตยกรรม 64 บิต เพราะว่าถ้าเป็น 32 บิตจะสามารถเก็บข้อมูลลงในมองโกดีบีได้สูงสุดเพียง 2 GB โดยหลังจากสร้างเซิฟเวอร์จำลองบนอเมซอน อีซีทู แล้ว ทำการเชื่อมต่อเพื่อดาว์นโหลดมองโกดีบีมาติดตั้ง และใช้งานเอง

3.2.2 การติดตั้งดีบีทู

การติดตั้งดีบีทูจะเป็นการติดตั้งโดยใช้อเมซอน แมคชีน อิมเมจ โดยเมื่อติดตั้งแล้วจะมีโปรแกรม ดีบีทูเอ็กซ์เพรส-ซี 10.1 ติดมาโดยอัตโนมัติ โดยสามารถเข้าไปใช้งานได้โดยใช้โปรแกรม ซีเคียวเชลล์ ไคลแอนท์ได้เช่นเดียวกับการสร้างเซิฟเวอร์จำลองด้วยตนเอง

3.3 การออกแบบการทดลองคำสั่งในการจัดการข้อมูลในมองโกดีบี

การทดลองนี้มีจุดประสงค์เพื่อสร้างความคุ้นเคยกับคำสั่งในมองโกดีบี โดยจะใช้งานในคำสั่งที่เกี่ยวกับการจัดการข้อมูล ได้แก่ การสร้าง (Create), การอ่าน (Read), การปรับปรุง (Update) และการลบข้อมูล (Delete) โดยจะทำกับระบบการจัดการฐานข้อมูลมองโกดีบีบนคลาวด์

3.4 การออกแบบการทดลองความสามารถในการค้นหาของมองโกดีบี

การทดลองนี้มีจุดประสงค์เพื่อทดสอบความสามารถในการค้นหา (query) ของมองโกดีบีว่าทำได้เท่าใดโดยนำมาเปรียบเทียบกับการค้นหาโดยใช้คำสั่งภาษาเอสคิวแอล ว่าในคำถามที่ ภาษาเอสคิวแอลตอบได้ ในมองโกดีบีสามารถตอบได้เช่นกันหรือไม่ โดยจะไม่เปรียบเทียบในแง่ของความเร็วในการค้นหา หรือความยาก-ง่ายในการเขียนคำสั่ง นอกจากนี้ยังมีการทดลองใช้แมพรีดิวส์ซึ่งมีข้อดีคือ สามารถประมวลผลแบบขนานได้

3.5 การออกแบบการทดลองใช้งานความสามารถอื่น ๆ ของมองโกดีบี

การทดลองนี้มีจุดประสงค์เพื่อใช้งานความสามารถอื่น ๆ ที่น่าสนใจ หรือสามารถนำมาประยุกต์ใช้ ในโปรแกรมประยุกต์ที่จะพัฒนา โดยจะทดลองใช้ความสามารถในส่วนต่าง ๆ ดังนี้

- การใช้งานอินเด็กซ์
- การใช้งานฟังก์ชั่นทางภูมิศาสตร์ในมองโกดีบี
- การใช้งานกริดเอฟเอส

3.6 การออกแบบการทดลองทรานแซกชั่นแบบกระจายในดีบีทู

การทดลองนี้จะใช้จาวา ทรานแซกชั่น เอพีไอ (Java transaction API: JTA) และ extended architecture (XA) ในการทำทรานแซกชั่นแบบกระจายระหว่างดีบีทู 2 เซิฟเวอร์ โดยทดลองโอนเงิน โดยถอนเงินจากบัญชีที่อยู่ในเซิฟเวอร์หนึ่ง แล้วฝากที่อีกเซิฟเวอร์ โดยทั้ง 2 เซิฟเวอร์มีรายละเอียดดังรูป

Name	Amount
Kik	13000
Mr.X	12000

Name	Amount
Pao	16000
Mr.Y	17000

รูปที่ 3.2 รายละเอียดฐานข้อมูลในการทดลอง distributed transaction

รายละเอียดของการปฏิบัติทรานแซกชั่นแบบกระจายมีดังนี้

1) เตรียมรายละเอียดของเซิฟเวอร์ฐานข้อมูลทั้งสอง
2) สร้างทรานแซกชั่นไอดี
3) เริ่มทรานแซกชั่นย่อยที่เซิฟเวอร์ฐานข้อมูล 1 ทำการถอนเงิน
4) เริ่มทรานแซกชั่นย่อยที่เซิฟเวอร์ฐานข้อมูล 2 ทำการฝากเงิน
5) ตรวจสอบสถานะของแต่ละเซิฟเวอร์ฐานข้อมูลว่าพร้อมยืนยันการเปลี่ยนแปลงข้อมูลทั้งหมดหรือไม่ เพื่อทำการยืนยันการเปลี่ยนแปลงข้อมูล หรือ ยกเลิกการเปลี่ยนแปลงของทรานแซกชั่นย่อย

3.7 การทดลองเขียนฟังก์ชั่นที่ผู้ใช้กำหนดเองในดีบีทู เพื่อทำการรู้จำใบหน้า

ในการทดลองนี้จะเขียนฟังก์ชั่นการรู้จำใบหน้าโดยใช้เทคนิคการวิเคราะห์องค์ประกอบหลัก (Principal Component Analysis: PCA) เพื่อให้โปรแกรมประยุกต์ใด ๆ สามารถใช้ฟังก์ชั่นการรู้จำใบหน้าจากดีบีทูได้ โดยที่ผู้เขียนโปรแกรมไม่จำเป็นต้องเขียนระบบรู้จำใบหน้าเอง และไม่ต้องรู้โค้ด

ของฟังก์ชั่นการรู้จำใบหน้า อีกทั้งเมื่อต้องการปรับปรุงเทคนิคในการรู้จำใบหน้า ก็เพียงแค่ไปแก้ไขฟังก์ชั่นในดีบีทู

ในการทดลองนี้จะแบ่งเป็น 4 ขั้นตอนคือ

1) การเก็บรูปลงในดีบีทู
2) การเขียนฟังก์ชั่นการรู้จำใบหน้า
3) การนำฟังก์ชั่นการรู้จำใบหน้าลงในดีบีทู
4) การทดสอบฟังก์ชั่นการรู้จำใบหน้า

3.8 การออกแบบระบบงานทะเบียนนักศึกษาจำลอง

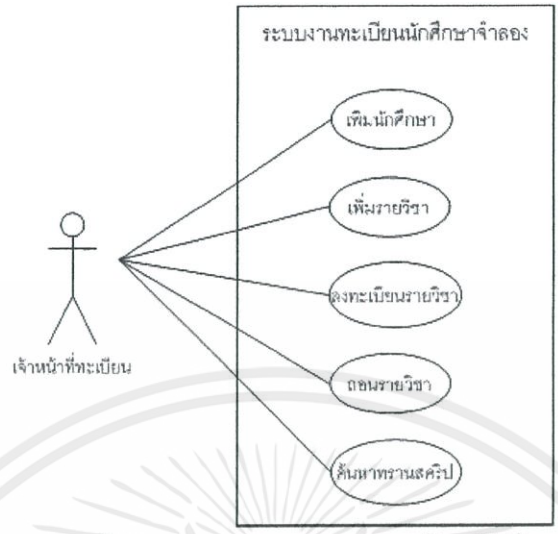
สำหรับการทดสอบระบบจัดเก็บข้อมูลหลายรูปแบบ (polyglot storage system) ที่ใช้ดีบีทูและมองโกดีบีที่ติดตั้งอยู่บนอเมซอน อีซีทู กลุ่มผู้จัดทำได้ออกแบบและพัฒนาระบบงานทะเบียนนักศึกษาจำลองที่สามารถใช้ประโยชน์จากระบบจัดเก็บข้อมูลหลายรูปแบบ และสามารถค้นหาทรานสคริปด้วยรูปใบหน้าของนักศึกษาได้

3.8.1 ภาพรวมของระบบ

ความต้องการของระบบมีดังนี้

- รองรับการเพิ่มนักศึกษา, รายวิชา, การลงทะเบียนเรียน, การถอนรายวิชา
- เก็บทรานสคริปของนักศึกษาแต่ละคนในรูปแบบเอกสารลงมองโกดีบี
- เรียกดูทรานสคริปเป็นไฟล์พีดีเอฟ (PDF) ได้โดยใช้รหัสหรือใบหน้านักศึกษา

ยูสเคสไดอะแกรม (use case diagram) ของระบบเป็นดังรูป 3.3

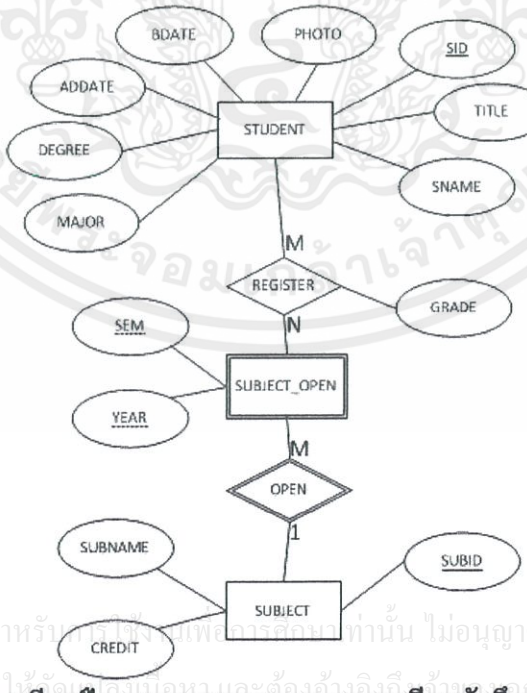


รูปที่ 3.3 ยูสเคสไดอะแกรมของระบบงานทะเบียนนักศึกษาจำลอง

3.8.2 การออกแบบฐานข้อมูล

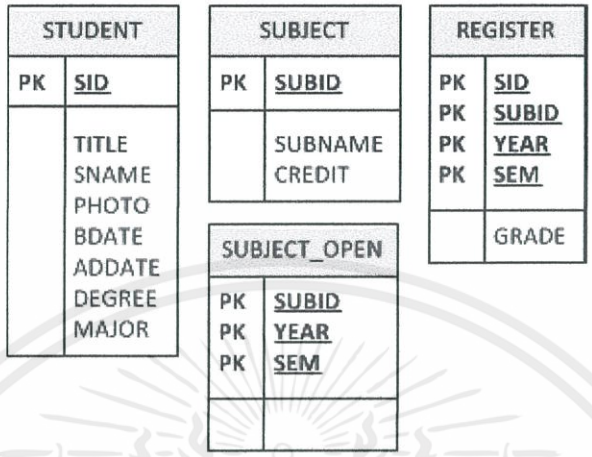
การออกแบบฐานข้อมูลในระบบงานทะเบียนนักศึกษาจำลองแบ่งเป็นส่วนที่เก็บในดีบีทู และมองโกดีบี ซึ่งมีรายละเอียดดังนี้

ข้อมูลที่เก็บในดีบีทูเขียนอธิบายด้วยอีอาร์ไดอะแกรม (entity–relationship diagram: E-R Diagram) ดังนี้



รูปที่ 3.4 อีอาร์ไดอะแกรมของระบบงานทะเบียนนักศึกษาจำลอง

เมื่อนำมาเขียนเป็นตารางจะได้ 4 ตารางดังรูป



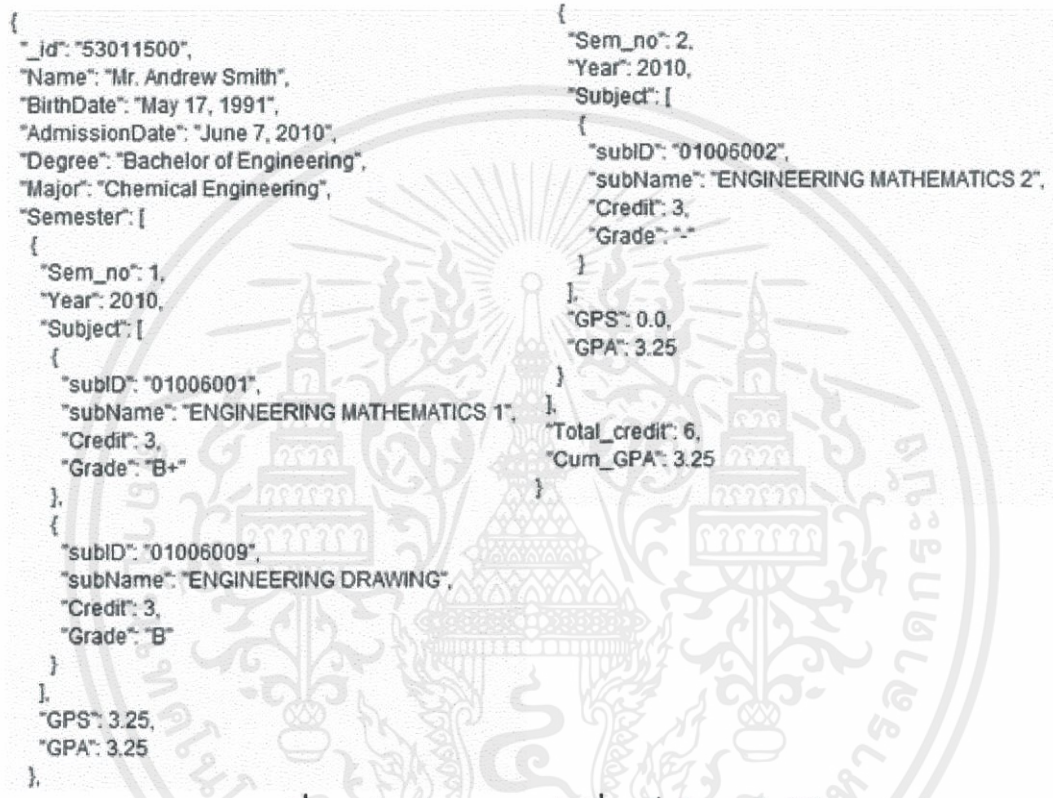
รูปที่ 3.5 ตารางของระบบงานทะเบียนนักศึกษาจำลองใน DB2

มองโกดีบีมีการสร้างคอลเลคชั่น transcripts ขึ้นมาเพื่อเก็บเอกสารทรานสคริปของนักศึกษาแต่ละคน ซึ่งมีโครงสร้างดังนี้

```
{
      _id: <รหัสนักศึกษา>
      Name: <ชื่อนักศึกษา>
      BirthDate: <วันเกิดของนักศึกษา>
      AdmissionDate: <วันที่เข้ามาเป็นนักศึกษา>
      Degree: <ระดับปริญญา>
      Major:<วิชาเอก>
      Semester:[ <เป็น array เก็บผลการเรียนแต่ละเทอม>
            {
                  Sem_no: <เทอม>
                  Year: <ปีการศึกษา>
                  Subject: [ <เป็น array เก็บผลการเรียนในเทอมนี้>
                        {
                              subID: <รหัสวิชา>
                              subName: <ชื่อวิชา>
                              Credit: <หน่วยกิต>
                              Grade: <เกรด>
                        }, ...
                  ]
                  GPS: <ผลการเรียนเฉลี่ยในเทอมนี้>
                  GPA: <ผลการเรียนเฉลี่ยตั้งแต่เริ่มจนถึงเทอมนี้>
```

```
            }, ...
        ]
        Total_credit: <หน่วยกิตรวม>
        Cum_GPA: <ผลการเรียนเฉลี่ย>
}
```

ตัวอย่างเอกสารทรานสคริปในมองโกดีบี กับทรานสคริปที่เป็นไฟล์พีดีเอฟ



```
{
 "_id": "53011500",
 "Name": "Mr. Andrew Smith",
 "BirthDate": "May 17, 1991",
 "AdmissionDate": "June 7, 2010",
 "Degree": "Bachelor of Engineering",
 "Major": "Chemical Engineering",
 "Semester": [
  {
   "Sem_no": 1,
   "Year": 2010,
   "Subject": [
    {
     "subID": "01006001",
     "subName": "ENGINEERING MATHEMATICS 1",
     "Credit": 3,
     "Grade": "B+"
    },
    {
     "subID": "01006009",
     "subName": "ENGINEERING DRAWING",
     "Credit": 3,
     "Grade": "B"
    }
   ],
   "GPS": 3.25,
   "GPA": 3.25
  },
  {
   "Sem_no": 2,
   "Year": 2010,
   "Subject": [
    {
     "subID": "01006002",
     "subName": "ENGINEERING MATHEMATICS 2",
     "Credit": 3,
     "Grade": "-"
    }
   ],
   "GPS": 0.0,
   "GPA": 3.25
  }
 ],
 "Total_credit": 6,
 "Cum_GPA": 3.25
}
```

รูปที่ 3.6 ตัวอย่างทรานสคริปที่ถูกเก็บในมองโกดีบี

EXAMPLE UNIVERSITY
BANGKOK, THAILAND

Name Mr. Andrew Smith
Student ID 53011500
Date of Birth May 17, 1991
Date of Admission June 7, 2010
Degree Bachelor of Engineering
Major Chemical Engineering

COURSE TITLE	CREDIT	GRADE	COURSE TITLE	CREDIT	GRADE
Semester 1, Year 2010					
01006001 ENGINEERING MATHEMATICS 1	3	B+			
01006009 ENGINEERING DRAWING	3	B			
GPS: 3.25 GPA: 3.25					
Semester 2, Year 2010					
01006002 ENGINEERING MATHEMATICS 2	3	-			
GPS: 0.0 GPA: 3.25					
Total Attended Credit Hours: 6					
Cumulative GPA: 3.25					

รูปที่ 3.7 ทรานสคริปที่เป็นไฟล์ PDF

ในการใช้ฐานข้อมูลทั้ง 2 ร่วมกันนั้นทุกๆ ครั้งที่มีการเพิ่มนักศึกษา, เพิ่มการลงทะเบียนรายวิชาของนักศึกษา, การให้เกรด และการถอนรายวิชาจากการลงทะเบียน จะต้องทำทั้งในดีบีทูและบันทึกทรานสคริปใหม่ที่เปลี่ยนแปลงในมองโกดีบีซึ่งการกระทำดังกล่าวต้องมีคุณสมบัติ atomicity หมายความว่าการบันทึกในดีบีทูและมองโกดีบีต้องสำเร็จทั้งคู่ แต่ถ้าการบันทึกในที่ใดที่หนึ่งไม่สำเร็จ ก็ต้องไม่สำเร็จทั้งคู่ เพื่อคงไว้ซึ่งความถูกต้องของข้อมูลในดีบีทูและมองโกดีบี

3.8.3 เครื่องมือที่ใช้ดำเนินงาน

การเขียนโปรแกรมระบบงานทะเบียนนักศึกษาจำลอง ได้พัฒนาบนจาวาแอพเพล็ต (Java Applet) และมีการฝังลงในแท็กเอชทีเอมแอล (HTML) ทำให้สามารถใช้งานผ่านเบราว์เซอร์ (Browser) ได้ และอัพโหลดขึ้นใช้งานบนคลาวด์ของอีลาสติก บีนทอล์ค (Elastic Beanstalk) ซึ่งช่วยให้สามารถจัดการโปรแกรมประยุกต์ได้ง่ายขึ้น โดยดูแลเพียงส่วนของโปรแกรมและข้อมูล ซึ่งส่วนอื่นๆ เช่น รันไทม์ (Run time), ระบบปฏิบัติการ (OS) และเซิร์ฟเวอร์ จะมีการจัดการโดยผู้ให้บริการคลาวด์โปรแกรมที่ใช้พัฒนาระบบนี้ขึ้นมาได้เลือกใช้โปรแกรม NetBeans IDE 7.4 ซึ่งมีความสามารถในการออกแบบหน้าตา GUI และการเชื่อมต่อกับ AWS Elastic Beanstalk เพื่ออัพโหลดโปรแกรมขึ้นบนคลาวด์

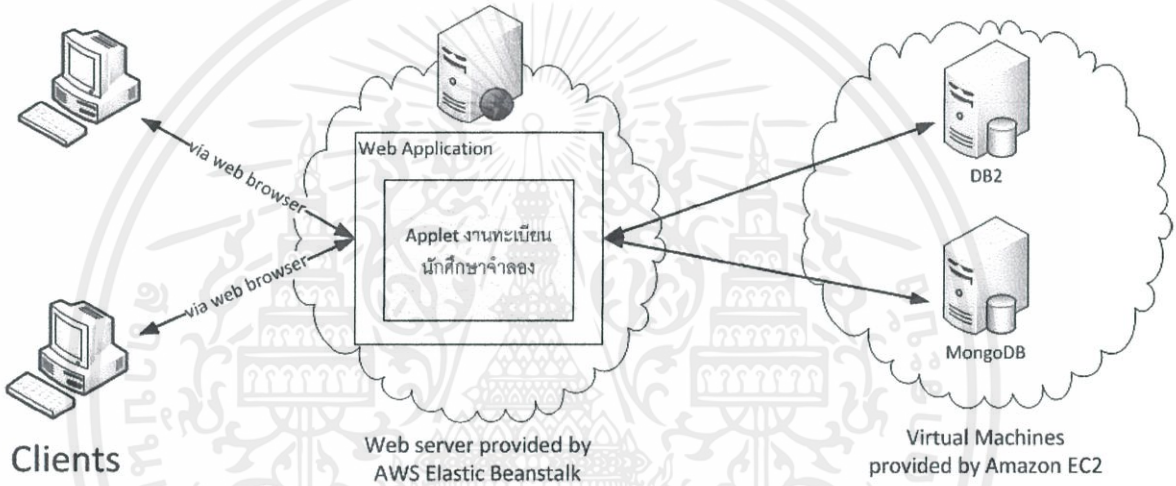
3.8.4 แผนภาพของระบบ

การออกแบบระบบงานทะเบียนนักศึกษาจำลอง ได้แบ่งเป็น 3 ส่วน ดังนี้

1) ส่วนของผู้ใช้งานที่ใช้งานโปรแกรมผ่านทางเว็บเบราว์เซอร์

2) ส่วนเว็บเซิฟเวอร์ ที่ใช้บริการของอีลาสติก บีนทอร์ค เพื่อเป็นที่อยู่ของเว็บแอพพลิเคชั่น ซึ่งมีแอพเพล็ต ของโปรแกรมระบบงานทะเบียนนักศึกษาจำลอง ติดตั้งอยู่

3) ส่วนฐานข้อมูล ซึ่งใช้บริการของ อเมซอน อีซีทู ในการสร้างเวอร์ชวลแมชชีน ที่มีดีบีทู และมองโกดีบีติดตั้งอยู่



รูปที่ 3.8 แผนภาพของระบบงานทะเบียนนักศึกษาจำลอง

บทที่ 4
การทดลองและผลการทดลอง

4.2 การทดลองคำสั่ง CRUD ในมองโกดีบี

ในการทดลองนี้จะทดลองคำสั่งที่ใช้การสร้างเอกสารลงคอลเลคชั่น, การอ่านเอกสารมาจากคอลเลคชั่น, การปรับปรุงข้อมูล และการลบข้อมูล

การทดลองนี้จะใช้คอลเลคชั่น bio จะเก็บข้อมูลส่วนตัวโดยมีโครงสร้างเอกสารดังนี้

```
{name: {
    first:
    last:
    }
 birth_date:
  age:
 favorite_color:
}
```

4.2.1 การสร้าง (Create)

คำสั่งที่ใช้ในการสร้างของมองโกดีบี เราสามารถใส่เอกสารลงในคอลเลคชั่นได้เลยโดยไม่ต้องสร้างคอลเลคชั่นไว้ก่อน และยังมีความเป็น dynamic schema คือชื่อฟิลด์และชนิดของข้อมูลบนฟิลด์ไม่จำเป็นต้องเหมือนกัน

คำสั่งที่ใช้ในการสร้าง คือคำสั่ง save ซึ่งจะใส่เอกสารลงในคอลเลคชั่นได้ทีละ 1 เอกสาร โดยถ้ายังไม่มีคอลเลคชั่นนั้นมองโกดีบี จะสร้างใหม่ให้

โปรแกรม 4.1 โครงสร้างสั่ง save

```
db.<collection name>.save(<document>)
```

เราจะทดลองใส่เอกสารลงไปในคอลเลคชั่น bio ทั้งสิ้น 3 เอกสาร ดังนี้

ตารางที่ 4.1 เอกสารต่างๆในคอลเลคชั่น bio

เอกสารที่	ข้อมูล

1	{
	name: {
	first: "Pranot",
	surname: "Mingbunjurdsuk"
	}
	birth_date: Jul 15,1992
	age: 21
	favorite_color: ["Blue","Green"]
	}

2	{
	name: {
	first: "Angkana",
	surname: "Suchaimanacharoen"
	}
	birth_date: Nov 07,1991
	age: 21
	favorite_color: ["Blue","Red","Orange"]
	}

3	{
	name: {
	first: "Bob",
	}
	age: 25
	favorite_color: ["Yellow","Red"]
	}

ใช้คำสั่ง save ในมองโกดีบีเชลล์ซึ่งเป็นส่วนที่ติดต่อกับข้อมูล

```
> db.bio.save({name:{first:"Pranot",last:"Mingbunjurdsuk"},birth_date: new Date(
"Jul 15,1992"),age:21,favorite_color: ["Blue","Green"]})
> db.bio.save({name:{first:"Angkana",last:"Suchaimanacharoen"},birth_date: new D
ate("Nov 07,1991"),age:21,favorite_color: ["Blue","Red","Orange"]})
> db.bio.save({name:{first:"Bob"},age:25,favorite_color: ["Yellow","Red"]})
```

รูปที่ 4.1 การนำเอกสารไปเก็บในคอลเลคชั่น

4.2.2 การอ่าน (Read)

คำสั่งที่ใช้ในการอ่านข้อมูลในคอลเลคชั่น คือคำสั่ง find โดยสามารถค้นหาข้อมูล (query) และ เลือกเฉพาะบางฟิลด์มาแสดงได้

โปรแกรม 4.2 โครงสร้างคำสั่ง find

```
db.<collection name>.find(<query criteria>, <projection>)
```

- <query criteria> คือเอกสารที่อธิบายสิ่งที่จะค้นหา
- <projection> คือเอกสารที่บอกว่าจะแสดงแอตทริบิวต์อะไรบ้าง

จากคอลเลคชั่น bio ที่สร้างขึ้น จะทำการทดลองใช้คำสั่ง find ทั้งสิ้น 4 กรณีดังนี้

1) อ่านเอกสารทั้งหมดในคอลเลคชั่น

ใช้คำสั่ง find ได้โดยไม่ต้องระบุ <query criteria> และ <projection> ดังรูป

```
> db.bio.find()
{ "_id" : ObjectId("523565bd6485cc2252740b9a"), "name" : { "first" : "Pranot", "
last" : "Mingbunjurdsuk" }, "birth_date" : ISODate("1992-07-15T00:00:00Z"), "age
" : 21, "favorite_color" : [  "Blue",  "Green" ] }
{ "_id" : ObjectId("523568326485cc2252740b9b"), "name" : { "first" : "Angkana",
"last" : "Suchaimanacharoen" }, "birth_date" : ISODate("1991-11-07T00:00:00Z"),
"age" : 21, "favorite_color" : [  "Blue",  "Red",  "Orange" ] }
{ "_id" : ObjectId("523568556485cc2252740b9c"), "name" : { "first" : "Bob" }, "a
ge" : 25, "favorite_color" : [  "Yellow",  "Red" ] }
```

รูปที่ 4.2 แสดงเอกสารทั้งหมดในคอลเลคชั่น bio

ทั้งสามเอกสารที่ออกมามี field _id อยู่ ซึ่งฟิลด์นั้นก็คือ ObjectID ที่มองโกดีบี สร้างให้อัตโนมัติ ซึ่งทำหน้าที่เป็น primary key ให้กับคอลเลคชั่น

เราสามารถใช้คำสั่ง pretty() ต่อท้าย คำสั่ง find เพื่อให้จัดรูปแบบเอกสารที่ออกมาให้อ่านได้ง่ายขึ้น

```
> db.bio.find().pretty()
{
        "_id" : ObjectId("523565bd6485cc2252740b9a"),
        "name" : {
                "first" : "Pranot",
                "last" : "Mingbunjurdsuk"
        },
        "birth_date" : ISODate("1992-07-15T00:00:00Z"),
        "age" : 21,
        "favorite_color" : [
                "Blue",
                "Green"
        ]
}
{
        "_id" : ObjectId("523568326485cc2252740b9b"),
        "name" : {
                "first" : "Angkana",
                "last" : "Suchaimanacharoen"
        },
        "birth_date" : ISODate("1991-11-07T00:00:00Z"),
        "age" : 21,
        "favorite_color" : [
                "Blue",
                "Red",
                "Orange"
        ]
}
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "name" : {
                "first" : "Bob"
        },
        "age" : 25,
        "favorite_color" : [
                "Yellow",
                "Red"
        ]
}
```

รูปที่ 4.3 เอกสารในคอลเลคชั่น bio ในรูปแบบที่อ่านง่าย

2) แสดงข้อมูลเฉพาะชื่อ และอายุของคนที่มีชื่อต้นคือ Bob

ใช้คำสั่ง find ได้โดยระบุ <query criteria> = {“name.first”: “Bob”} และ <projection> = {_id:0, name:1, age:1}

```
> db.bio.find({"name.first":"Bob"},{_id:0,name:1,age:1}).pretty()
{ "name" : { "first" : "Bob" }, "age" : 25 }
```

รูปที่ 4.4 ข้อมูลเฉพาะชื่อ และอายุในคอลเลคชั่น bio ของคนที่มีชื่อต้นคือ Bob

ในส่วน <projection> จะระบุ field ที่ต้องการให้ออกมาโดยให้ค่าเป็น 1 ส่วน _id จะต้องให้ค่าเป็น 0 จึงจะไม่แสดงค่าออกมา ส่วนฟิลด์อื่นๆ ที่ไม่ระบุก็จะไม่ออกมาเช่นกัน

3) แสดงเอกสารที่มีค่าอายุน้อยกว่า 25

คำสั่งที่ใช้ในการเปรียบเทียบค่าได้แก่ $gt มากกว่า, $lt น้อยกว่า, $gte มากกว่าหรือเท่ากับ, $lte น้อยกว่าหรือเท่ากับ และ $ne ไม่เท่ากับ

ใช้คำสั่ง find ได้โดยระบุ <query criteria> = {age: {$lt:25}}

```
> db.bio.find({age:{"$lt":25}}).pretty()
{
        "_id" : ObjectId("523565bd6485cc2252740b9a"),
        "name" : {
                "first" : "Pranot",
                "last" : "Mingbunjurdsuk"
        },
        "birth_date" : ISODate("1992-07-15T00:00:00Z"),
        "age" : 21,
        "favorite_color" : [
                "Blue",
                "Green"
        ]
}
{
        "_id" : ObjectId("523568326485cc2252740b9b"),
        "name" : {
                "first" : "Angkana",
                "last" : "Suchaimanacharoen"
        },
        "birth_date" : ISODate("1991-11-07T00:00:00Z"),
        "age" : 21,
        "favorite_color" : [
                "Blue",
                "Red",
                "Orange"
        ]
}
```


รูปที่ 4.5 เอกสารในคอลเลคชั่น bio ที่มีค่าอายุน้อยกว่า 25

4) แสดงเอกสารที่มี favorite_color คือ Green หรือ Yellow

ใช้คำสั่ง find ได้โดยระบุ <query criteria> = {favorite_color: {$in:["Green","Yellow"]} } โดยถ้าค่าใดค่าหนึ่งใน favorite_color ตรงกับค่าใดค่าหนึ่งใน $in จะดึงเอกสารนั้นออกมา

```
> db.bio.find({favorite_color:{$in:["Green","Yellow"]}}).pretty()
{
        "_id" : ObjectId("523565bd6485cc2252740b9a"),
        "name" : {
                "first" : "Pranot",
                "last" : "Mingbunjurdsuk"
        },
        "birth_date" : ISODate("1992-07-15T00:00:00Z"),
        "age" : 21,
        "favorite_color" : [
                "Blue",
                "Green"
        ]
}
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "name" : {
                "first" : "Bob"
        },
        "age" : 25,
        "favorite_color" : [
                "Yellow",
                "Red"
        ]
}
```

รูปที่ 4.6 เอกสารในคอลเลคชั่น bio ที่ favorite_color คือ Green หรือ Yellow

4.2.3 การปรับปรุงข้อมูล (Update)

คำสั่งที่ใช้ในการปรับปรุงข้อมูลในคอลเลคชั่น คือคำสั่ง update โดยสามารถ ปรับปรุงค่าในฟิลด์, เพิ่มฟิลด์, ลบฟิลด์ หรือ แทนที่ด้วยเอกสารใหม่

โปรแกรมที่ 4.3 โครงสร้างคำสั่ง update ในมองโกดีบี

```
db.<collection name>.update(
                    <query>,
                    <update>,
                    {
                      upsert: <Boolean>,
                      multi: <Boolean>,
                    }
                    )
```

พารามิเตอร์ upsert หมายถึง จะนำเอกสารที่จะปรับปรุงลงคอลเลคชั่นถ้าไม่มีเอกสารที่ตรงกับ <query> เลย ส่วน multi หมายถึง การปรับปรุงจะปรับปรุงทุกเอกสารที่ตรงกับ <query>

จากคอลเลคชั่น bio ที่สร้างขึ้น จะทำการทดลองใช้คำสั่ง update ทั้งสิ้น 5 กรณีดังนี้

1) ปรับปรุงค่า age ในเอกสารที่มีชื่อต้นเป็น Bob ให้เป็น 30

ใช้คำสั่ง update โดยระบุ <query> = {“name.first”:“Bob”} และ <update> = {$set: {age:30}} โดย $set จะเป็นคำสั่งที่ใช้ในการเปลี่ยนแปลงค่าใน field ที่กำหนด หรือสร้าง field ใหม่ถ้าเดิมไม่มี field นั้นอยู่ จะได้ผลลัพธ์ดังรูป

```
> db.bio.update({"name.first":"Bob"},{$set:{age:30}})
> db.bio.find({"name.first":"Bob"}).pretty()
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "name" : {
                "first" : "Bob"
        },
        "age" : 30,
        "favorite_color" : [
                "Yellow",
                "Red"
        ]
}
```

รูปที่ 4.7 การปรับปรุงค่า age ในคอลเลคชั่น bio ที่เอกสารมีชื่อต้นเป็น Bob

2) การเพิ่มฟิลด์ใหม่ลงในเอกสารที่มีชื่อต้นเป็น Bob

กรณีนี้จะเพิ่มฟิลด์ name.last ลงไป โดยใช้คำสั่ง update โดยระบุ <query> = {“name.first”:“Bob”} และ <update> = {$set: {“name.last”:“Smith”}} จะได้ผลลัพธ์ดังรูป

```
> db.bio.update({"name.first":"Bob"},{$set:{"name.last":"Smith"}})
> db.bio.find({"name.first":"Bob"}).pretty()
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "age" : 30,
        "favorite_color" : [
                "Yellow",
                "Red"
        ],
        "name" : {
                "first" : "Bob",
                "last" : "Smith"
        }
}
```


รูปที่ 4.8 การเพิ่ม field ในคอลเลคชั่น bio ที่เอกสารมีชื่อต้นเป็น Bob

3) การแทนที่เอกสารใหม่ ลงในเอกสารที่มีชื่อต้นเป็น Bob

ในการแทนที่ด้วยเอกสาร ให้ใส่เอกสารใหม่ลงใน <update> ใช้คำสั่ง update โดยระบุ <query> = {“name.first”:“Bob”} และ <update> = {name:{first: “Alice”},age: 24} จะได้ผลลัพธ์ดังรูป

```
> db.bio.update({"name.first":"Bob"},{name:{first:"Alice"},age:24})
> db.bio.find({"name.first":"Alice"}).pretty()
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "name" : {
                "first" : "Alice"
        },
        "age" : 24
}
```

รูปที่ 4.9 การแทนที่เอกสารในคอลเลคชั่น bio ที่เอกสารมีชื่อต้นเป็น Bob

4) การเพิ่มค่าลงในฟิลด์ที่เป็นอาเรย์ ให้กับทุกเอกสาร

ในกรณีนี้จะเพิ่ม White ลงในฟิลด์ favorite_color ของทุกเอกสาร จึงต้องใช้พารามิเตอร์ multi เป็น true และการนำค่าไปต่อท้ายของอาเรย์ใช้ $push

ใช้คำสั่ง update โดยระบุ <query> = {} , <update> = {$push: {favorite_color: “White”}} และ multi:true จะได้ผลลัพธ์ดังรูป

```
> db.bio.update({},{$push:{favorite_color:"White"}},{multi:true})
> db.bio.find({},{_id:0,name:1,favorite_color:1})
{ "favorite_color" : [  "White" ], "name" : { "first" : "Alice" } }
{ "favorite_color" : [  "Blue",  "Green",  "White" ], "name" : { "first" : "Pran
ot", "last" : "Mingbunjurdsuk" } }
{ "favorite_color" : [  "Blue",  "Red",  "Orange",  "White" ], "name" : { "first
" : "Angkana", "last" : "Suchaimanacharoen" } }
```

รูปที่ 4.10 การเพิ่มค่าลงใน field ที่เป็นอาเรย์ของคอลเลคชั่น bio

5) การลบฟิลด์ออกจากเอกสาร

ในกรณีนี้จะลบฟิลด์ age ออกจากเอกสารที่มีชื่อต้นเป็น Alice โดยใช้ $unset แล้วระบุชื่อฟิลด์ที่ต้องการลบออก

ใช้คำสั่ง update โดยระบุ <query> = {“name.first”:“Alice”} และ <update> = {$unset: {age:1}} จะได้ผลลัพธ์ดังรูป

```
> db.bio.update({"name.first":"Alice"},{$unset:{age:1}})
> db.bio.find({"name.first":"Alice"}).pretty()
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "favorite_color" : [
                "White"
        ],
        "name" : {
                "first" : "Alice"
        }
}
```

รูปที่ 4.11 การลบ field ออกจากเอกสาร

4.2.4 การลบ (Delete)

คำสั่งที่ใช้ในการลบเอกสารในคอลเลคชั่น คือคำสั่ง remove โดยสามารถ ลบเอกสารทั้งหมดที่ค้นหาออกมา หรือลบเฉพาะเอกสารแรกที่ค้นหาออกมา

โปรแกรมที่ 4.4 โครงสร้างของคำสั่ง remove

```
db.<collection name>.remove(
                        <query>,
                        justOne: <boolean>
                    )
```

นอกจากนี้ยังมีคำสั่งที่ใช้ลบทั้งคอลเลคชั่นคือคำสั่ง drop

โปรแกรมที่ 4.5 โครงสร้างของคำสั่ง remove

```
db.<collection name>.drop()
```

จากคอลเลคชั่น bio ที่สร้างขึ้น จะทำการทดลองใช้คำสั่ง update ทั้งสิ้น 2 กรณีดังนี้

1) การลบเอกสารที่ query ออกมา

ในกรณีนี้จะลบเอกสาร ที่มี age เป็น 21 โดยใช้คำสั่ง remove โดยระบุ <query> = {age: 21} จะได้ผลลัพธ์ดังรูป

```
> db.bio.remove({age:21})
> db.bio.find().pretty()
{
        "_id" : ObjectId("523568556485cc2252740b9c"),
        "favorite_color" : [
                "White"
        ],
        "name" : {
                "first" : "Alice"
        }
}
```

รูปที่ 4.12 การลบเอกสารที่ query ออกมา

2) การลบทั้งคอลเลคชั่น

ใช้คำสั่ง drop() ถ้าลบสำเร็จจะได้ผลลัพธ์กลับมาคือ true

```
> db.bio.drop()
true
```

รูปที่ 4.13 การลบทั้งคอลเลคชั่น

4.3 การทดลองความสามารถในการค้นหาของมองโกดีบี

การทดลองนี้จะใช้ คอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา เพื่อทดลองความสามารถในการค้นหาว่าทำได้เทียบเท่ากับการใช้ภาษา SQL หรือไม่ ซึ่งโครงสร้างของเอกสารในคอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา เป็นดังนี้

```
{
        Name:
        Birth_yr:
        Yrs_serv:
        Death_age:
        Party:
        Pres_marriage: [ {
                Spouse_name:
                Pr_age:
                Sp_age:
                Nr_childern:
                Mar_year:
        } ]
        Hobby: [ ]
        Administration:[ {
                Admin_nr:
                Year_inaugurated:
                Vice_pres_name:
        } ]
}
```

ตัวอย่างเอกสารในคอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา


```
{
        "_id" : ObjectId("52lca5le546045fc232fadlb"),
        "Name" : "Coolidge C",
        "Birth_yr" : 1872,
        "Yrs_serv" : 5,
        "Death_age" : 60,
        "Party" : "Republican",
        "State_born" : "New York",
        "Pres_marriage" : [
                {
                        "Spouse_name" : "Goodhue G A",
                        "Pr_age" : 33,
                        "Sp_age" : 26,
                        "Nr_children" : 2,
                        "Mar_year" : 1905
                }
        ],
        "Hobby" : [
                "Fishing",
                "Golf",
                "Indian Clubs",
                "Mechanical Horse",
                "Pitching Hay"
        ],
        "Administration" : [
                {
                        "Admin_nr" : 34,
                        "Year_inaugurated" : 1923
                },
                {
                        "Admin_nr" : 35,
                        "Year_inaugurated" : 1925,
                        "Vice_pres_name" : "Dawes C G"
                }
        ]
}
```


รูปที่ 4.14 ตัวอย่างเอกสารในคอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา

การทดลองจะมีทั้งการค้นหาผ่านทางมองโกเชลล์และในบางการทดลองจะมีการเขียนโปรแกรมภาษาจาวาโดยใช้เอพีไอของมองโกดีบีในการติดต่อกับมองโกดีบีบนเซิฟเวอร์จำลองเพื่อค้นหาข้อมูล

4.3.1 การค้นหาโดยใช้คำสั่ง find

มีการทดลองดังนี้

1) ประธานาธิบดีที่ดำรงตำแหน่งมากกว่า 5 ปี และ ไม่เกิดในรัฐ Texas และอยู่พรรค Republican

ใช้คำสั่ง $gt เพื่อแทนเครื่องหมายมากกว่า และ $ne แทนเครื่องหมายไม่เท่ากับคำสั่งที่ใช้ในมองโกเชลล์ คือ

โปรแกรมที่ 4.6 คำสั่งในมองโกเชลล์ที่ใช้หาประธานาธิบดีที่ดำรงตำแหน่งมากกว่า 5 ปี และไม่เกิดในรัฐ Texas และอยู่พรรค Republican

```
db.President.find({
      Yrs_serv: {$gt:5},
      State_born: {$ne:"Texas"},
      Party:"Republican"
      }).pretty()
```

คำสั่งที่ใช้ในภาษาจาวา

โปรแกรมที่ 4.7 คำสั่งในภาษาจาวาที่ใช้หาประธานาธิบดีที่ดำรงตำแหน่งมากกว่า 5 ปี และ ไม่เกิดในรัฐ Texasและอยู่พรรค Republican

```
BasicDBObject query = new BasicDBObject("Yrs_serv",
            new BasicDBObject("$gt",5))
            .append("State_born", new BasicDBObject("$ne","Texas"))
            .append("Party", "Republican");
DBCursor res = coll.find(query);
while(res.hasNext())
      System.out.println(res.next);
```

ในโปรแกรมภาษาจาวาเราจะต้องสร้างตัวแปรชนิด BasicDBObject เพื่อเอาไว้เก็บเอกสารที่จะนำไปค้นหาและตัวแปรชนิด DBCursor เอาไว้เก็บเอกสารผลลัพธ์

2) แสดงชื่อ, จำนวนปีที่ดำรงตำแหน่ง, อายุขัย ของ president ที่มีอายุขัยระหว่าง 60 ถึง 65 ปี หรือ ดำรงตำแหน่งน้อยกว่า 4 ปี

ใช้คำสั่ง $or เพื่อทำคำสั่ง logical or ระหว่างค่าในอาเรย์ของ $or คำสั่งที่ใช้ในมองโกเชลล์ คือ

โปรแกรมที่ 4.8 คำสั่งในมองโกเชลล์ที่ใช้แสดงชื่อ, จำนวนปีที่ดำรงตำแหน่ง, อายุขัย ของ president ที่มีอายุขัยระหว่าง 60 ถึง 65 ปี หรือ ดำรงตำแหน่งน้อยกว่า 4 ปี

```
db.President.find({
      $or: [
            {Death_age: {$gte:60,$lte:65}},
            {Yrs_serv: {$lt:4}}
      ]},
      {_id:0,Name:1,Yrs_serv:1,Death_age:1})
```

คำสั่งที่ใช้ในภาษาจาวา

โปรแกรมที่ 4.9 คำสั่งในภาษาจาวา ที่ใช้แสดงชื่อ, จำนวนปีที่ดำรงตำแหน่ง, อายุขัย ของ president ที่มีอายุขัยระหว่าง 60 ถึง 65 ปี หรือ ดำรงตำแหน่งน้อยกว่า 4 ปี

```
BasicDBList array = new BasicDBList();
array.add(new BasicDBObject("Death_age",new BasicDBObject("$gt",60)
      .append("$lt",65)));
array.add(new BasicDBObject("Yrs_serv",new BasicDBObject("$lt",4)));

BasicDBObject query = new BasicDBObject("$or",array);

BasicDBObject proj = new BasicDBObject("_id",0)
      .append("Name",1)
      .append("Yrs_serv",1)
      .append("Death_age",1);

DBCursor res = coll.find(query,proj);
while(res.hasNext())
      System.out.println(res.next);
```

ในโปรแกรมภาษาจาวาเราจะต้องสร้างตัวแปรชนิด BasicDBList ขึ้นมาเพื่อเก็บเอกสารที่เป็นอาเรย์

ผลลัพธ์ที่ออกมาคือ

```
{ "Death_age" : 68, "Name" : "Harrison W H", "Yrs_serv" : 0 }
{ "Name" : "Tylor J", "Yrs_serv" : 3, "Death_age" : 71 }
{ "Name" : "Taylor Z", "Yrs_serv" : 1, "Death_age" : 65 }
{ "Name" : "Fillmore M", "Yrs_serv" : 2, "Death_age" : 74 }
{ "Name" : "Pierce F", "Yrs_serv" : 4, "Death_age" : 64 }
{ "Name" : "Johnson A", "Yrs_serv" : 3, "Death_age" : 66 }
{ "Name" : "Grant U S", "Yrs_serv" : 8, "Death_age" : 63 }
{ "Name" : "Garfield J A", "Yrs_serv" : 0, "Death_age" : 49 }
{ "Name" : "Arthur C A", "Yrs_serv" : 3, "Death_age" : 56 }
{ "Name" : "Roosevelt T", "Yrs_serv" : 7, "Death_age" : 60 }
{ "Name" : "Harding W G", "Yrs_serv" : 2, "Death_age" : 57 }
{ "Name" : "Coolidge C", "Yrs_serv" : 5, "Death_age" : 60 }
{ "Name" : "Roosevelt F D", "Yrs_serv" : 12, "Death_age" : 63 }
{ "Name" : "Kennedy J F", "Yrs_serv" : 2, "Death_age" : 46 }
{ "Name" : "Johnson L B", "Yrs_serv" : 5, "Death_age" : 65 }
{ "Name" : "Ford G R", "Yrs_serv" : 2 }
{ "Name" : "Reagan R", "Yrs_serv" : 3 }
```

รูปที่ 4.15 ผลลัพธ์จากการหา president ที่มีอายุขัยระหว่าง 60 ถึง 65 ปี หรือ ดำรงตำแหน่งน้อยกว่า 4 ปี

3) หาประธานาธิบดีที่ไม่เคยแต่งงาน

ในมองโกดีบีสามารถตรวจสอบว่ามีฟิลด์นี้อยู่ในเอกสารหรือไม่โดยใช้คำสั่ง $exists คำสั่งที่ใช้ในมองโกเชลล์คือ

โปรแกรมที่ 4.10 คำสั่งในมองโกเชลล์ที่ใช้หาประธานาธิบดีที่ไม่เคยแต่งงาน

```
db.President.find({
      Pres_marriage: {$exists:false}
}).pretty()
```

คำสั่งที่ใช้ในภาษาจาวา

โปรแกรมที่ 4.11 คำสั่งในภาษาจาวา ที่ใช้หาประธานาธิบดีที่ไม่เคยแต่งงาน

```
BasicDBObject query = new BasicDBObject("Pres_marriage",
      new BasicDBObject("$exists",false));

DBCursor res = coll.find(query);
while(res.hasNext())
      System.out.println(res.next);
```

ผลลัพธ์ที่ออกมาคือ

```
{
        "_id" : ObjectId("521c930954605cb326348ea7"),
        "Name" : "Buchanan J",
        "Birth_yr" : 1791,
        "Yrs_serv" : 4,
        "Death_age" : 77,
        "Party" : "Democratic",
        "State_born" : "Pennsylvania",
        "Administration" : [
                {
                        "Admin_nr" : 18,
                        "Year_inaugurated" : 1857,
                        "Vice_pres_name" : "Breckinridge J C"
                }
        ]
}
```

รูปที่ 4.16 ผลลัพธ์จากการหาประธานาธิบดีที่ไม่เคยแต่งงาน

4) แสดง ชื่อ, อายุขัย ของประธานาธิบดีที่ชื่อขึ้นต้นด้วย A เรียงอายุขัยจากน้อยไปมาก

ในกรณีนี้เป็นการทดลองการค้นหาด้วยส่วนของสตริง (substring) ซึ่งในมองโกดีบีก็สามารถทำได้ โดยการใส่เครื่องหมาย '/' ล้อมหน้าและหลังส่วนของสตริงนั้น และถ้าต้องการหาเฉพาะคำขึ้นต้นให้ใส่ '^' หน้าส่วนของสตริง ถ้าเป็นคำลงท้ายเท่านั้นให้ใส่ '.' หน้าส่วนของสตริงเช่นเดียวกัน

สำหรับการเรียงลำดับใช้คำสั่ง sort ต่อท้าย โดยใส่พารามิเตอร์เป็นชื่อฟิลด์ที่ต้องการเรียงลำดับ (sort) แล้วใส่ค่า 1 ถ้าต้องการให้เรียงจากน้อยไปมาก หรือ -1 ถ้าต้องการให้เรียงจากมากไปน้อย

คำสั่งที่ใช้ในมองโกเชลล์ คือ

โปรแกรมที่ 4.12 คำสั่งในมองโกเชลล์ที่ใช้หาประธานาธิบดีที่ชื่อขึ้นต้นด้วย A เรียงอายุขัยจากน้อยไปมาก

```
db.President.find({
      Name: /^A/},
      {_id:0,Name:1,Death_age:1}
).sort({Death_age:1})
```

สำหรับในภาษาจาวา จะต้องใช้ $regex เพื่อทำการค้นหาด้วยส่วนของสตริง โดยไม่ต้องใส่ '/' ล้อมหน้าและหลัง ส่วนการเรียงลำดับให้ทำหลังจากการ find เสร็จก่อน

คำสั่งที่ใช้ในภาษาจาวา

โปรแกรมที่ 4.13 คำสั่งในภาษาจาวา ที่ใช้หาประธานาธิบดีที่ชื่อขึ้นต้นด้วย A เรียงอายุขัยจากน้อยไปมาก

```
BasicDBObject query = new BasicDBObject("Name",
      new BasicDBObject("$regex","^A"));

BasicDBObject proj = new BasicDBObject("_id",0)
      .append("Name",1)
      .append("Death_age",1);

DBCursor res = coll.find(query,proj);
res.sort(new BasicDBObject("Death_age",1));
while(res.hasNext())
      System.out.println(res.next);
```

ผลลัพธ์ที่ออกมาคือ

```
{ "Name" : "Arthur C A" , "Death_age" : 56}
{ "Name" : "Adams J Q" , "Death_age" : 80}
{ "Name" : "Adams J" , "Death_age" : 90}
```

รูปที่ 4.17 ผลลัพธ์จากการหาประธานาธิบดีที่ชื่อขึ้นต้นด้วย A เรียงอายุขัยจากน้อยไปมาก

4.3.2 การ query โดยใช้ Aggregation Pipeline

มีการทดลองดังนี้

1) หาว่าแต่ละพรรคมีประธานาธิบดีมาแล้วกี่คน , ดำรงตำแหน่งรวมกันกี่ปี และดำรงตำแหน่งเฉลี่ยคนละกี่ปี เรียงจากจำนวนปีที่ดำรงตำแหน่งรวมกัน จากมากไปน้อย

ในข้อนี้ เราจะต้อง $group field party โดยระบุไปที่ field _id แล้วจึงใช้ฟังก์ชั่น $sum และ $avg ซึ่งเป็นบิลท์อินฟังก์ชั่น (built-in function) โดย $sum ถ้ารวมค่าตัวเลขของฟิลด์ที่ระบุ ถ้าระบุเป็นตัวเลขก็จะนำตัวเลขนั้นมาคิดผลรวม ส่วน $avg เป็นการหาค่าเฉลี่ยของฟิลด์ที่ต้องการ

เมื่อทำคำสั่ง $group แล้วต่อมาคือการเรียงลำดับผลจากการ $group โดยใช้คำสั่ง $sort โดยคำสั่งที่ใช้ในมองโกเชลล์คือ

โปรแกรมที่ 4.14 คำสั่งในมองโกเชลล์ที่ใช้หาว่าแต่ละพรรคมีประธานาธิบดีมาแล้วกี่คน, ดำรงตำแหน่งรวมกันกี่ปี และดำรงตำแหน่งเฉลี่ยคนละกี่ปี เรียงจากจำนวนปีที่ดำรงตำแหน่งรวมกัน จากมากไปน้อย

```
db.President.aggregate([
      { $group:{
            _id:"$Party",
```

```
            Total_yrs_serv: {$sum: "$Yrs_serv"},
            Count: {$sum:1},
            Avg_serv:{$avg:"$Yrs_serv"}}
      },
      {$sort:{Total_yrs_serv:-1}}
])
```

ผลลัพธ์ที่ออกมาคือ

```
{ "_id" : "Democratic" , "Total_yrs_serv" : 73 , "Count" : 13 , "Avg_serv" : 5.615384615384615}
{ "_id" : "Republican" , "Total_yrs_serv" : 67 , "Count" : 16 , "Avg_serv" : 4.1875}
{ "_id" : "Demo-Rep" , "Total_yrs_serv" : 28 , "Count" : 4 , "Avg_serv" : 7.0}
{ "_id" : "Federalist" , "Total_yrs_serv" : 11 , "Count" : 2 , "Avg_serv" : 5.5}
{ "_id" : "Whig" , "Total_yrs_serv" : 6 , "Count" : 4 , "Avg_serv" : 1.5}
```

รูปที่ 4.18 ผลลัพธ์จากการหาว่าแต่ละพรรคมีประธานาธิบดีมาแล้วกี่คน, ดำรงตำแหน่งรวมกันกี่ปี และดำรงตำแหน่งเฉลี่ยคนละกี่ปี เรียงจากจำนวนปีที่ดำรงตำแหน่งรวมกัน จากมากไปน้อย

2) เรียงลำดับ Hobby ที่มีประธานาธิบดีเล่นจากมากไปน้อย โดยเลือกเฉพาะที่มี President เล่นมากกว่า 1 คน

ในข้อนี้จะต้อง $unwind ในฟิลด์ Hobby เพื่อกระจายค่าในอาเรย์ให้ออกมาเป็นค่าเดี่ยว จากนั้น $group ที่ field Hobby และสร้างฟิลด์ Count ขึ้นมา เพื่อนับจำนวน Hobby ที่มีประธานาธิบดีเล่น จากนั้น $match เพื่อหา Hobby ที่มีประธานาธิบดีเล่นมากกว่า 1 คน และสุดท้ายคือ $sort เพื่อเรียงลำดับผลลัพธ์ในฟิลด์ count

โปรแกรมที่ 4.15 คำสั่งในมองโกเชลล์ที่ใช้เรียงลำดับ Hobby ที่มีประธานาธิบดีเล่นจากมากไปน้อย โดยเลือกเฉพาะที่มี President เล่นมากกว่า 1 คน

```
db.President.aggregate([
      {$unwind: "$Hobby"},
      {$group:{
            _id:"$Hobby",
            Count: {$sum:1}
      }},
      {$match:{Count: {$gt:1}}},
      {$sort:{Count:-1}}
])
```

ผลลัพธ์ที่ออกมาคือ

```
{ "_id" : "Riding" , "Count" : 11}
{ "_id" : "Fishing" , "Count" : 9}
{ "_id" : "Golf" , "Count" : 6}
{ "_id" : "Walking" , "Count" : 6}
{ "_id" : "Swimming" , "Count" : 4}
{ "_id" : "Hunting" , "Count" : 3}
{ "_id" : "Sailing" , "Count" : 2}
{ "_id" : "Shooting" , "Count" : 2}
{ "_id" : "Poker" , "Count" : 2}
{ "_id" : "Billiards" , "Count" : 2}
```

รูปที่ 4.19 ผลลัพธ์จากการเรียงลำดับ Hobby ที่มี President เล่นจากมากไปน้อย โดยเลือกเฉพาะที่มี President เล่นมากกว่า 1 คน

3) แสดงชื่อ,ปีเกิด และจำนวนบุตรรวมของประธานาธิบดีที่อยู่ในพรรค Democratic ที่มีจำนวนบุตรมากกว่า 3 คน

ในข้อนี้จะต้อง $match ประธานาธิบดี ที่สังกัดพรรค Democratic ก่อน จากนั้น $unwind ในฟิลด์ Pres_marriage เพื่อกระจายค่าในอาเรย์ให้ออกมาเป็นค่าเดี่ยว แล้ว $group ที่ฟิลด์ Name และคำนวณฟิลด์ที่จะให้ออกมาเป็นผลลัพธ์คือฟิลด์ Birth_yr และ Sum_children และสุดท้ายคือ $match ให้ออกเฉพาะเอกสารที่มีบุตรรวมมากกว่า 3 คน

โปรแกรมที่ 4.16 คำสั่งในมองโกเชลล์ที่ใช้แสดงชื่อ,ปีเกิด และจำนวนบุตรรวมของประธานาธิบดีที่อยู่ในพรรค Republican ที่มีจำนวนบุตรมากกว่า 3 คน

```
db.President.aggregate([
      {$match:{Party:"Republican"}},
      {$unwind:"$Pres_marriage"},
      {$group:{
            _id:"$Name",
            Birth_yr:{$min:"$Birth_yr"},
            Sum_children: {$sum:"$Pres_marriage.Nr_children"}}},
      {$match:{Sum_children: {$gt:3}}}
])
```

ผลลัพธ์ที่ออกมาคือ

```
{ "_id" : "Reagan R" , "Birth_yr" : 1911 , "Sum_children" : 4}
{ "_id" : "Ford G R" , "Birth_yr" : 1913 , "Sum_children" : 4}
{ "_id" : "Roosevelt T" , "Birth_yr" : 1858 , "Sum_children" : 6}
{ "_id" : "Garfield J A" , "Birth_yr" : 1831 , "Sum_children" : 7}
{ "_id" : "Hayes R B" , "Birth_yr" : 1822 , "Sum_children" : 8}
{ "_id" : "Grant U S" , "Birth_yr" : 1822 , "Sum_children" : 4}
{ "_id" : "Lincoln A" , "Birth_yr" : 1809 , "Sum_children" : 4}
```

รูปที่ 4.20 ผลลัพธ์จากการแสดงชื่อ,ปีเกิด และจำนวนบุตรรวมของประธานาธิบดีที่อยู่ในพรรค Democratic ที่มีจำนวนบุตรมากกว่า 3 คน

4.3.3 การค้นหาโดยใช้แมพรีดิวส์ (Map-Reduce)

มีการทดลองดังนี้

1) แสดงชื่อ Hobby และจำนวนประธานาธิบดีที่เล่น hobby นั้น

ในข้อนี้จะสร้างแมพฟังก์ชั่น (map function) เพื่อสร้างคู่คีย์แวลู (key-value pairs) (คือคำสั่ง emit ในโปรแกรม) โดยกำหนดคีย์ คือ ชื่อ Hobby และแวลูคือ {count:1} ซึ่งจะมีไว้นับจำนวน hobby ที่มี

โปรแกรมที่ 4.17 แมพฟังก์ชั่น

```
map = function (){
      if(this.Hobby != null)
            this.Hobby.forEach(function(hobby) {
                  emit(hobby,{count:1});
            });
}
```

จะได้ตัวอย่าง key-value pair ที่ออกมาดังรูป

```
> map.apply(x)
emit
key: Billiards value: { "count" : 1 }
emit
key: Swimming value: { "count" : 1 }
emit
key: Walking value: { "count" : 1 }
```

รูปที่ 4.21 ตัวอย่าง key-value pair

รีดิวส์ฟังก์ชั่น (reduce function) จะนับจำนวนครั้งที่ปรากฏค่า {count:1} ใน hobby แต่ละชนิด เพื่อรวมเป็นจำนวนประธานาธิบดีที่เล่น hobby นั้น

โปรแกรมที่ 4.18 รีดิวส์ฟังก์ชั่น

```
reduce = function (key,values){
      return {count:values.length};
}
```

จากนั้นใช้คำสั่ง mapReduce

โปรแกรมที่ 4.19 คำสั่ง mapReduce ที่ใช้แสดงชื่อ Hobby และจำนวนประธานาธิบดีที่เล่น hobby นั้น

```
res = db.President.mapReduce(map,reduce,{out:{inline:1}})
```

ตัวอย่างบางส่วนของผลลัพธ์ที่ออกมาคือ

```
{
        "_id" : "Golf",
        "value" : {
                "count" : 6
        }
},
{
        "_id" : "Hunting",
        "value" : {
                "count" : 3
        }
},
{
        "_id" : "Indian Clubs",
        "value" : {
                "count" : 1
        }
},
                {
                        "_id" : "Wrestling",
                        "value" : {
                                "count" : 1
                        }
                }
        ],
        "timeMillis" : 3,
        "counts" : {
                "input" : 39,
                "emit" : 60,
                "reduce" : 10,
                "output" : 23
        },
        "ok" : 1,
}
```

รูปที่ 4.22 บางส่วนของผลลัพธ์ของการแสดงชื่อ Hobby และจำนวนประธานาธิบดีที่เล่น hobby นั้น

2) หาจำนวนประธานาธิบดี, จำนวนบุตรรวมและจำนวนบุตรเฉลี่ยต่อคนในพรรคแต่ละพรรค

ในข้อนี้จะสร้างแมพฟังก์ชั่นเพื่อสร้างคู่คีย์แวลู (คือคำสั่ง emit ในโปรแกรม) โดยกำหนดคีย์คือ ชื่อพรรค และแวลู คือ {nr_children: , nr_pres: 1} ซึ่งจะมีไว้เก็บจำนวนบุตรของประธานาธิบดีคนนั้น และนับจำนวนประธานาธิบดี

โปรแกรมที่ 4.20 แมพฟังก์ชั่น

```
map = function(){
      var key_a = this.Party
      var n = 0;
      if(this.Pres_marriage != null)
            this.Pres_marriage.forEach(
                  function(mar){
                        n += mar.Nr_children;
                  }
            );
      var value_b = {nr_children:n, nr_pres:1};
      emit(key_a , value_b);
}
```

รีดิวส์จะนับจำนวนประธานาธิบดีในพรรคนั้น และคิดผลรวมของจำนวนบุตรในพรรคนั้นด้วย

โปรแกรมที่ 4.21 รีดิวส์ฟังก์ชั่น

```
reduce = function (key,values){
      var total = 0;
      var count = 0;
      for(var i=0;i<values.length;i++){
            total += values[i].nr_children;
            count += values[i].nr_pres;
      }
      return{nr_children:total, nr_pres:count};
}
```

สำหรับการคิดค่าจำนวนบุตรเฉลี่ยต่อคน จะอยู่ในไฟนอลไลซ์ฟังก์ชั่น (finalize) function โดยเอาค่าใน nr_children หารกับ nr_pres สร้างเป็น field ใหม่ คือ avg

โปรแกรมที่ 4.22 นอลไลซ์ฟังก์ชั่น

```
finalize_fn = function(key,  reducedVal){
      reducedVal.avg = reducedVal.nr_children/reducedVal.nr_pres;
      return reducedVal;
}
```

จากนั้นใช้คำสั่ง mapReduce

โปรแกรมที่ 4.23 คำสั่ง mapReduce ที่ใช้หาจำนวนประธานาธิบดี, จำนวนบุตรรวมและจำนวนบุตรเฉลี่ยต่อคนในพรรคแต่ละพรรค

```
db.President.mapReduce(map,reduce,{finalize:finalize_fn,
out:{inline:1}})
```

ผลลัพธ์ที่ออกมาคือ


```
"results" : [
        {
                "_id" : "Demo-Rep",
                "value" : {
                        "nr_children" : 16,
                        "nr_pres" : 4,
                        "avg" : 4
                }
        },
        {
                "_id" : "Democratic",
                "value" : {
                        "nr_children" : 36,
                        "nr_pres" : 13,
                        "avg" : 2.769230769230769
                }
        },
        {
                "_id" : "Federalist",
                "value" : {
                        "nr_children" : 5,
                        "nr_pres" : 2,
                        "avg" : 2.5
                }
        },
        {
                "_id" : "Republican",
                "value" : {
                        "nr_children" : 56,
                        "nr_pres" : 16,
                        "avg" : 3.5
                }
        },
        {
                "_id" : "Whig",
                "value" : {
                        "nr_children" : 33,
                        "nr_pres" : 4,
                        "avg" : 8.25
                }
        }
],
"timeMillis" : 4,
"counts" : {
        "input" : 39,
        "emit" : 39,
        "reduce" : 5,
        "output" : 5
},
"ok" : 1,
```


รูปที่ 4.23 ผลลัพธ์จากการหาจำนวนประธานาธิบดี, จำนวนบุตรรวมและจำนวนบุตรเฉลี่ยต่อคนในพรรคแต่ละพรรค

3) แสดงชื่อภรรยา และปีที่แต่งงาน ของการแต่งงานครั้งล่าสุดของประธานาธิบดีแต่ละท่าน

ในข้อนี้จะสร้างแมพฟังก์ชั่น เพื่อสร้างคู่คีย์แวลู (คือคำสั่ง emit ในโปรแกรม) โดยกำหนดคีย์คือ ชื่อประธานาธิบดี และแวลู คือ {sp_name: Mar_yr:} ซึ่งจะมีไว้เก็บชื่อคู่สมรส และปีที่แต่งงาน โดยประธานาธิบดีที่ไม่เคยแต่งงานจะไม่สร้างคู่คีย์แวลูออกมา

โปรแกรม 4.24 แมพฟังก์ชั่น

```
map = function(){
      var key_a = this.Name
      if(this.Pres_marriage != null)
            this.Pres_marriage.forEach(
                  function(mar){
```

```
                        var value_b = {sp_name: mar.Spouse_name,
Mar_yr:                                 mar.Mar_year};
                        emit(key_a ,value_b);
                    }
            );
}
```

รีดิวส์ฟังก์ชั่นจะหาการแต่งงานครั้งล่าสุดจากปีที่แต่งงานที่น้อยที่สุดออกมาเป็นผลลัพธ์

โปรแกรม 4.25 รีดิวส์ฟังก์ชั่น

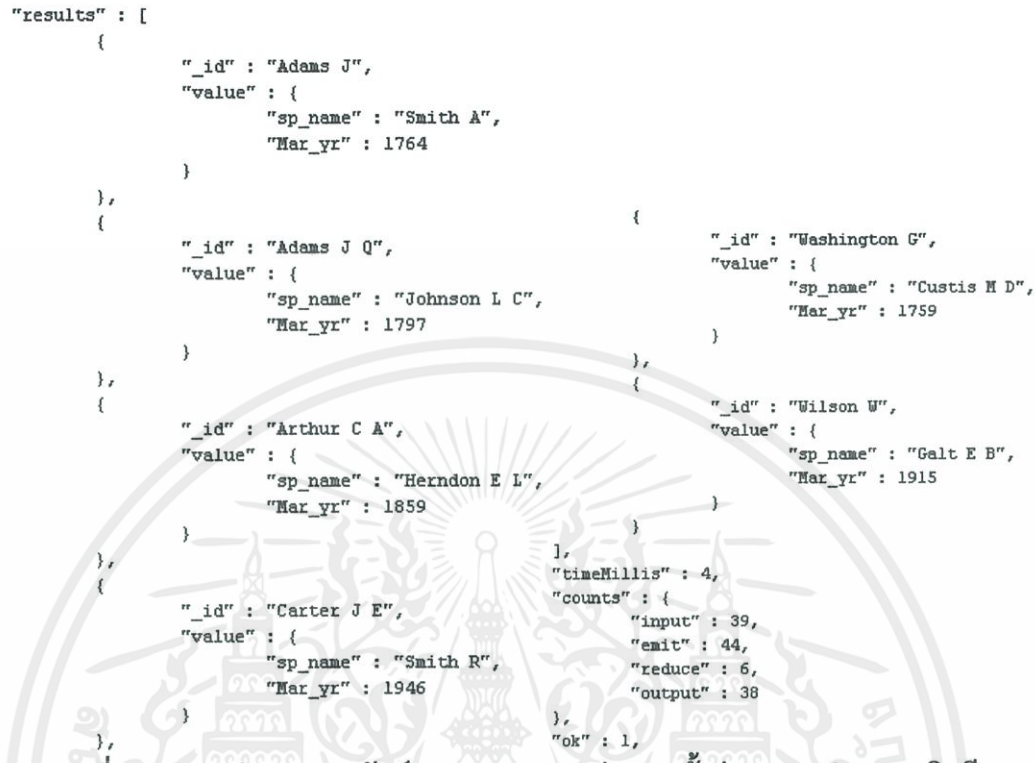
```
reduce = function (key,values){
      var latest_mar_yr = 0;
      var value_a;
      for(var i=0;i<values.length;i++){
          if(values[i].Mar_yr > latest_mar_yr)
                value_a = {sp_name: values[i].sp_name, Mar_yr:
                          values[i].Mar_yr};
      }
      return value_a;
}
```

จากนั้นใช้คำสั่ง mapReduce

โปรแกรมที่ 4.26 คำสั่งแมพรีดิวส์ที่ใช้หาภรรยา และปีที่แต่งงาน ของการแต่งงานครั้งล่าสุดของประธานาธิบดีแต่ละท่าน

```
res = db.President.mapReduce(map,reduce,{out:{inline:1}})
```

ตัวอย่างบางส่วนของผลลัพธ์ที่ออกมาคือ



รูปที่ 4.24 บางส่วนของผลลัพธ์จากการหาการแต่งงานครั้งล่าสุดของประธานาธิบดี

การทดลองความสามารถในการค้นหาว่าทำได้เทียบเท่ากับการใช้ภาษาเอสคิวแอลหรือไม่ พบว่า ในมองโกดีบีไม่มีคำสั่งที่ความสามารถเทียบเท่าการจอย (join) ของเอสคิวแอลดังนั้นจึงไม่สามารถจอยคอลเลคชั่นได้ จึงต้องเก็บข้อมูลที่ใช้ร่วมกันลงในคอลเลคชั่นเดียวกัน ซึ่งอาจจะเป็นไปได้เพราะ คอลเลคชั่นในมองโกดีบีมีคุณสมบัติ dynamic schema คือทุกเอกสารในคอลเลคชั่น ไม่จำเป็นต้องมีฟิลด์ที่เหมือนกัน นอกจากนี้มองโกดีบียังทำ subquery ไม่ได้อีกด้วย ทั้งนี้การวัดความสามารถว่าเทียบเท่าเอสคิวแอลหรือไม่ จะไม่รวมถึงความเร็วในการค้นหา และความยากในการเขียนคำสั่ง แต่ดูเฉพาะความสามารถในการตอบคำถามทีเอสคิวแอลสามารถหาคำตอบได้หรือไม่

สำหรับข้อดีของการค้นหาโดยใช้มองโกดีบี คือการทำแมพรีดิวส์ เพราะว่าสามารถทำการประมวลผลแบบขนานได้

4.4 การทดลองการใช้อินเด็กซ์ ในมองโกดีบี

จากข้อดีของการทำอินเด็กซ์ (Index) ในด้านการเพิ่มความเร็วของการค้นหา การทดลองนี้จึงเปรียบเทียบความเร็วในแง่ของจำนวนเอกสารที่ต้องสแกนเพื่อหาคำตอบ

คอลเลคชั่นที่ใช้ทำการทดลองคือ คอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา โดยจะสร้างอินเด็กซ์ที่ฟิลด์ Name โดยใช้คำสั่งคือ

โปรแกรมที่ 4.27 คำสั่งในการสร้างอินเด็กซ์บนฟิลด์ Name ของคอลเลคชั่นประธานาธิบดีสหรัฐอเมริกา

```
db.President.exsureIndex({"Name":1})
```

จากนั้นทดลองค้นหา ที่เกี่ยวข้องกับ Name เพื่อให้ query optimizer เลือกใช้อินเด็กซ์ที่สร้างขึ้นในการค้นหา และใช้คำสั่ง explain() เพื่อให้เห็น query plan ไว้เปรียบเทียบ จำนวนเอกสารที่ต้องสแกนเพื่อหาคำตอบ และความเร็วที่ใช้

ตารางที่ 4.2 เปรียบเทียบจำนวนเอกสารที่ต้องสแกนเพื่อหาคำตอบ และความเร็วที่ใช้

ไม่ใช้อินเด็กซ์บน Name	ใช้อินเด็กซ์บน Name

> db.President.find({Name:"Adams J Q"}).explain
{
"cursor" : "BasicCursor",
"isMultiKey" : false,
"n" : 1,
"nscannedObjects" : 39,
"nscanned" : 39,
"nscannedObjectsAllPlans" : 39,
"nscannedAllPlans" : 39,
"scanAndOrder" : false,
"indexOnly" : false,
"nYields" : 0,
"nChunkSkips" : 0,
"millis" : 0,
"indexBounds" : {
},
"server" : "ip-172-31-7-175:27017"
}
	> db.President.find({Name:"Adams J Q"}).explain
	{
	"cursor" : "BtreeCursor Name_1",
	"isMultiKey" : false,
	"n" : 1,
	"nscannedObjects" : 1,
	"nscanned" : 1,
	"nscannedObjectsAllPlans" : 1,
	"nscannedAllPlans" : 1,
	"scanAndOrder" : false,
	"indexOnly" : false,
	"nYields" : 0,
	"nChunkSkips" : 0,
	"millis" : 0,
	"indexBounds" : {
	"Name" : [
	[
	"Adams J Q",
	"Adams J Q"
	]
	]
	},
	"server" : "ip-172-31-7-175:27017"
	}

พบว่าเมื่อไม่ใช้อินเด็กซ์การหาคำตอบจะต้องสแกนเอกสารทุกเอกสารในคอลเลคชั่นเพื่อหาคำตอบ แต่ถ้าใช้อินเด็กซ์จะสามารถหาคำตอบได้ทันที

4.5 การทดลองฟังก์ชั่นทางภูมิศาสตร์ในมองโกดีบี

ในการทดลองนี้ใช้คอลเลคชั่น Place ในการทดลองฟังก์ชั่นทางภูมิศาสตร์ได้แก่

- $geoWithin เพื่อหาจุด เส้น หรือพื้นที่ ที่อยู่ในรูปทรงเรขาคณิตที่กำหนด
- $geoIntersects เพื่อหาจุด เส้น หรือพื้นที่ ที่ตัดกับรูปทรงเรขาคณิตที่กำหนด

- $near เพื่อหาจุด เส้น หรือพื้นที่ ที่อยู่ใกล้จุดที่กำหนด

เอกสารที่อยู่ในคอลเลคชั่น Place มีดังนี้

```
> db.place.find()
{ "_id" : "Home1", "loc" : { "type" : "Point", "coordinates" : [  40,  5 ] } }
{ "_id" : "Home2", "loc" : { "type" : "Point", "coordinates" : [  20,  20 ] } }
{ "_id" : "Home3", "loc" : { "type" : "Point", "coordinates" : [  2,  5 ] } }
{ "_id" : "Home4", "loc" : { "type" : "Point", "coordinates" : [  10,  -10 ] } }
{ "_id" : "Village_A", "loc" : { "type" : "Polygon", "coordinates" : [  [  [  0,
  0 ],  [  0,  20 ],  [  20,  20 ],  [  20,  0 ],  [  0,  0 ] ] ] } }
{ "_id" : "Home5", "loc" : { "type" : "Point", "coordinates" : [  25.5,  10 ] }
}
```

รูปที่ 4.25 เอกสารในคอลเลคชั่น Place

4.5.1 การทดลองใช้ฟังก์ชั่น $geoWithin

ใช้ฟังก์ชั่น $geoWithin หาบ้านที่อยู่ใน Village_A ใช้คำสั่งดังนี้

โปรแกรมที่ 4.28 คำสั่งที่ใช้หาบ้านที่อยู่ใน Village_A

```
db.place.find({
      loc: {$geoWithin:
            {$geometry:db.place.findOne({_id:"Village_A"}).loc}},
       _id:{$ne:"Village_A"}})
```

จากการใช้ฟังก์ชั่น $geoWithin เราจะต้องเลือก $geometry เป็นพื้นที่ที่จะหารูปทรงเรขาคณิตที่อยู่ในพื้นที่นั้น ซึ่งก็คือฟิลด์ loc ของ Village_A ซึ่งได้ผลลัพธ์ดังนี้

```
{ "_id" : "Home3", "loc" : { "type" : "Point", "coordinates" : [  2,  5 ] } }
{ "_id" : "Home2", "loc" : { "type" : "Point", "coordinates" : [  20,  20 ] } }
```

รูปที่ 4.26 ผลลัพธ์จากการหาบ้านที่อยู่ใน Village_A

4.5.2 การทดลองใช้ฟังก์ชั่น $geoIntersects

ใช้ฟังก์ชั่น $geoIntersects หาสถานที่เส้นตรง (0,5),(50,5) ตัดผ่านใช้คำสั่งดังนี้

โปรแกรมที่ 4.29 คำสั่งที่ใช้หาสถานที่เส้นตรง (0,5),(50,5) ตัดผ่าน

```
db.place.find({
      loc:{
            $geoIntersects:{
```

```
                $geometry: {type:"LineString",
                            coordinates:[[0,5],[50,5]]
                            }
            }
        }
})
```

จากการใช้ฟังก์ชั่น $geoIntersects เราจะต้องเลือก $geometry เป็นเส้นตรง คือประเภทเส้นสตริง (LineString) และโคออร์ดิเนต (coordinates) คืออาเรย์ที่เก็บจุดเริ่มต้นกับสุดสิ้นสุดของเส้นตรงนั้น คือ [[0,5],[50,5]] ซึ่งได้ผลลัพธ์ดังนี้

```
{ "_id" : "Village_A", "loc" : { "type" : "Polygon", "coordinates" : [ [ [ 0,
  0 ], [ 0, 20 ], [ 20, 20 ], [ 20, 0 ], [ 0, 0 ] ] ] } }
```

รูปที่ 4.27 ผลลัพธ์จากการหาสถานที่เส้นตรง (0,5),(50,5) ตัดผ่าน

4.5.2 การทดลองใช้ฟังก์ชั่น $near

ใช้ฟังก์ชั่น $near หาสถานที่ที่ใกล้จุด (0,0) ในระยะห่างไม่เกิน 2,000,000 เมตร ซึ่งสาเหตุที่ต้องใช้ระยะห่างสูงถึง 2,000,000 เมตร เพราะว่าหน่วยที่ใช้เก็บพิกัดเป็นหน่วยละติจูด, ลองจิจูด เมื่อคำนวณหาระยะห่างแล้วจะห่างกันมาก

คำสั่งที่ใช้คือ

โปรแกรมที่ 4.30 คำสั่งที่ใช้หาสถานที่ที่ใกล้จุด (0,0) ในระยะห่างไม่เกิน 2,000,000 เมตร

```
db.place.find({
      loc:{
            $near: {
                  $geometry: {type:"Point",
                              coordinates:[0,0]}
                  },
                  $maxDistance:2000000
      }
})
```

จากการใช้ฟังก์ชั่น $near เราจะต้องเลือก $geometry เป็นจุด คือประเภทจุด (Point) และโคออร์ดิเนต (Coordinates) คือจุดศูนย์กลางที่จะใช้วัดระยะห่าง คือ [0,0] และเลือก $maxDistance คือระยะห่างสูงสุดจากจุดที่เลือกไว้ คือ 2000000 ซึ่งได้ผลลัพธ์ดังนี้

```
{ "_id" : "Village_A", "loc" : { "type" : "Polygon", "coordinates" : [ [ [ 0,
  0 ], [ 0, 20 ], [ 20, 20 ], [ 20, 0 ], [ 0, 0 ] ] ] } }
{ "_id" : "Home3", "loc" : { "type" : "Point", "coordinates" : [ 2, 5 ] } }
{ "_id" : "Home4", "loc" : { "type" : "Point", "coordinates" : [ 10, -10 ] } }
```

รูปที่ 4.28 ผลลัพธ์จากการหาสถานที่ที่ใกล้จุด (0,0) ในระยะห่างไม่เกิน 2,000,000 เมตร

4.6 การทดลองการเก็บไฟล์โดย GridFS ในมองโกดีบี

การทดลองนี้จะทดลองใช้คุณสมบัติ GridFS ของมองโกดีบีที่สามารถเก็บไฟล์ได้ เช่น ไฟล์อักษร (text file), ไฟล์ภาพ และไฟล์เสียง เป็นต้น โดยจะเก็บข้อมูลเหล่านี้ในรูปแบบไบนารี่ โดยจะทดลองการเขียนไฟล์, การแสดงไฟล์ทั้งหมด, การค้นหาไฟล์, การโหลดไฟล์ และแสดงข้อมูลในคอลเลคชั่นไฟล์ (files) และชังค์ (chunks) ซึ่งเกี่ยวข้องกับการเก็บข้อมูล GridFS

1) การเขียนไฟล์

การทดลองนี้จะนำไฟล์อักษร 3 ไฟล์ คือ textfile1.txt, textfile2.txt และ textfile3.txt ซึ่งมีอยู่ในเครื่องที่ทำการทดลองแล้ว เข้าไปเก็บในฐานข้อมูล records ซึ่งระบุไว้ใน -d

คำสั่งที่ใช้คือคำสั่ง put ตามด้วยชื่อไฟล์

โปรแกรมที่ 4.31 การทดลองเขียนไฟล์

```
mongofiles -d records put textfile1.txt
mongofiles -d records put textfile2.txt
mongofiles -d records put textfile3.txt
```

2) การแสดงข้อมูลไฟล์

การทดลองนี้จะใช้คำสั่ง list เพื่อแสดงชื่อไฟล์ทั้งหมดในฐานข้อมูล records โดยมีคำสั่ง และผลลัพธ์ดังรูป

```
ubuntu@ip-172-31-7-175:~$ mongofiles -d records list
connected to: 127.0.0.1
textfile1.txt   36
textfile2.txt   7
textfile3.txt   30
```

รูปที่ 4.29 การแสดงข้อมูลไฟล์ใน GridFS

3) การค้นหาไฟล์

การทดลองนี้จะใช้คำสั่ง list และ search เพื่อหาไฟล์จากชื่อไฟล์ โดยคำสั่งทั้งสองมีความแตกต่างกันคือ list จะค้นหาเฉพาะชื่อต้น (prefix) ของไฟล์ แต่ search จะค้นหาเพียงส่วนใดส่วน

หนึ่งของชื่อไฟล์ตรงกับสิ่งที่จะค้นหา เช่นถ้าต้องการค้นหาไฟล์ชื่อ textfile2.txt ถ้าใช้คำสั่ง list จะต้องค้นหาด้วย “textfile2” แต่ถ้าใช้คำสั่ง search ใช้คำค้นหา “file2” ก็ใช้ได้แล้ว

```
ubuntu@ip-172-31-7-175:~$ mongofiles -d records list textfile2
connected to: 127.0.0.1
textfile2.txt   7
ubuntu@ip-172-31-7-175:~$ mongofiles -d records search file2
connected to: 127.0.0.1
textfile2.txt   7
```

รูปที่ 4.30 การค้นหาไฟล์ใน GridFS

4) การโหลดไฟล์

การทดลองนี้จะใช้คำสั่ง get เพื่อโหลดไฟล์ textfile1.txt มาเก็บในเครื่องโดยใช้ชื่อ text1.txt โดยระบุชื่อไฟล์ที่จะให้เก็บในออฟชั่น -l

```
ubuntu@ip-172-31-7-175:~$ mongofiles -d records -l textl.txt get textfilel.txt
connected to: 127.0.0.1
done write to: textl.txt
```

รูปที่ 4.31 การโหลดไฟล์ใน GridFS

5) การลบไฟล์

การทดลองนี้ใช้คำสั่ง delete ตามด้วยชื่อไฟล์ที่ต้องการลบ

```
ubuntu@ip-172-31-7-175:~$ mongofiles -d records delete textfilel.txt
connected to: 127.0.0.1
done!
ubuntu@ip-172-31-7-175:~$ mongofiles -d records list
connected to: 127.0.0.1
textfile2.txt   7
textfile3.txt   30
```

รูปที่ 4.32 การลบไฟล์ใน GridFS

6) ข้อมูลในคอลเลคชั่นไฟล์ (files)

ในคอลเลคชั่น files จะเก็บเมทาเดทา (metadata) ของไฟล์นั้นไว้ ได้แก่ ชื่อไฟล์, ขนาดของชังค์ที่เก็บไฟล์, วันเวลาที่อัพโหลด, ค่าแฮชแบบ MD5 และความยาวของข้อมูลในไฟล์

```
> db.fs.files.find().pretty()
{
        "_id" : ObjectId("51ff37a38fef88b7f2c68d16"),
        "filename" : "textfile2.txt",
        "chunkSize" : 262144,
        "uploadDate" : ISODate("2013-08-05T05:26:59.263Z"),
        "md5" : "f447b20a7fcbf53a5d5be013ea0b15af",
        "length" : 7
}
{
        "_id" : ObjectId("51ff3820fdae6fedd3ddbf52"),
        "filename" : "textfile3.txt",
        "chunkSize" : 262144,
        "uploadDate" : ISODate("2013-08-05T05:29:04.900Z"),
        "md5" : "5b72c669d1163b0c1469e41bcf22a4d2",
        "length" : 30
}
```

รูปที่ 4.33 ข้อมูลในคอลเลคชั่น files

7) ข้อมูลในคอลเลคชั่นชังค์

คอลเลคชั่นชังค์จะเก็บเนื้อข้อมูลจริงๆ โดยไฟล์หนึ่งไฟล์ ถ้ามีขนาดใหญ่จะมีหลายชังค์ได้

```
> db.fs.chunks.find().pretty()
{
        "_id" : ObjectId("51ff37a348f8199b8af7b466"),
        "files_id" : ObjectId("51ff37a38fef88b7f2c68d16"),
        "n" : 0,
        "data" : BinData(0,"MTIzNDU2Cg==")
}
{
        "_id" : ObjectId("51ff382048f8199b8af7b467"),
        "files_id" : ObjectId("51ff3820fdae6fedd3ddbf52"),
        "n" : 0,
        "data" : BinData(0,"dGV4dCBmaWxlIDMKZm9yIHRlc3Rpbmcgb25seSEK")
}
```

รูปที่ 4.34 ข้อมูลในคอลเลคชั่นชังค์

4.6 การทดลองทรานแซคชั่นแบบกระจาย (Distributed Transaction) ในดีบีทู

การทดลองนี้จะทำการทดลองทรานแซคชั่นโอนเงินระหว่างดีบีทู 2 เซิฟเวอร์ ซึ่งมีรายละเอียดในหัวข้อ 3.6 โดยใช้ภาษาจาวาในการทดลอง มีรายละเอียดการปฏิบัติทรานแซคชั่นแบบกระจาย โอนเงินจำนวน 3000 จากบัญชี Mr.X ไป บัญชี Mr.Y

1) เตรียมรายละเอียดของเซิร์ฟเวอร์ฐานข้อมูลทั้งสอง

ในเซิร์ฟเวอร์ฐานข้อมูลทั้งสอง ต้องเตรียมข้อมูลต่าง ๆ ได้แก่ ชื่อเซิฟเวอร์, หมายเลขพอร์ต, ชื่อฐานข้อมูล, ชื่อผู้ใช้, รหัสผ่าน และคำอธิบาย ลงใน DB2XADataSource แล้วแล้วสร้างการเชื่อมต่อไปยังเซิฟเวอร์นั้น

โปรแกรมที่ 4.32 การเตรียมข้อมูลก่อนการทำทรานแซคชั่นแบบกระจาย

```
//prepare DB2 server 1
DB2XADataSource db2ds = new DB2XADataSource();

db2ds.setDescription("DB2 Database 1");
db2ds.setServerName("ec2-54-204-30-239.compute-1.amazonaws.com");
db2ds.setPortNumber(50001);
db2ds.setDatabaseName("testdb");
db2ds.setUser("db2inst1");
db2ds.setPassword("pass");
db2ds.setDriverType(4);

XAConnection xaConn = db2ds.getXAConnection();
XAResource xaRes = xaConn.getXAResource();
Connection con = xaConn.getConnection();

//prepare DB2 server 2
DB2XADataSource ds_2 = new DB2XADataSource();

ds_2.setDescription("DB2 Database 2");
ds_2.setServerName("ec2-54-204-115-251.compute-1.amazonaws.com");
ds_2.setPortNumber(50001);
ds_2.setDatabaseName("testdb");
ds_2.setUser("db2inst1");
ds_2.setPassword("pass");
ds_2.setDriverType(4);

XAConnection xaConn_2 = ds_2.getXAConnection();
XAResource xaRes_2 = xaConn_2.getXAResource();
Connection con_2 = xaConn_2.getConnection();
```

2) สร้างโกบอลทรานแซคชั่นไอดี

ในการทำทรานแซคชั่นแบบกระจาย จะมีการสร้าง Xid เพื่อให้ระบุว่า transaction ใด เป็นโกบอลทรานแซคชั่น

โปรแกรมที่ 4.33 การสร้าง Xid

```
//Create Transaction ID
Xid db2xid = null;
db2xid = XidImpl.getUniqueXid(1);
```


3) เริ่มโลคอลทรานแซคชั่นที่เซิร์ฟเวอร์ฐานข้อมูล 1 ทำการถอนเงิน

การเริ่มและการสิ้นสุดโลคอลทรานแซคชั่นใด ๆ จะต้องสั่งเริ่ม และสิ้นสุดทุกครั้ง โดยใน โลคอลทรานแซกชั่นนี้ จะถอนเงินจำนวน 3000 จากบัญชี Mr.X

โปรแกรมที่ 4.34 local transaction ใน DB server 1

```
//start local transaction 1
xaRes.start(db2xid, XAResource.TMNOFLAGS);
Statement stmt = con.createStatement();

stmt.executeUpdate("update account2 set amount = amount - 3000 where
name =      'Mr.X'");

xaRes.end(db2xid, XAResource.TMSUCCESS);
//end local transaction 1
```

4) เริ่มโลคอลทรานแซคชั่นที่เซิร์ฟเวอร์ฐานข้อมูล 2 ทำการฝากเงิน

ในโลคอลทรานแซคชั่น นี้จะฝากเงินจำนวน 3000 เข้าบัญชี Mr.Y

โปรแกรมที่ 4.35 โลคอลทรานแซคชั่นในเซิร์ฟเวอร์ฐานข้อมูล 2

```
//start local transaction 2
xaRes_2.start(db2xid, XAResource.TMNOFLAGS);
stmt = con_2.createStatement();

stmt.executeUpdate("update account set amount = amount + 3000 where
name = 'Mr.Y'");

xaRes_2.end(db2xid, XAResource.TMSUCCESS);
//end local transaction 2
```

5) ตรวจสอบสถานะของแต่ละเซิร์ฟเวอร์ฐานข้อมูลว่าพร้อมคอมมิต ทั้งหมดหรือไม่ เพื่อทำการคอมมิต (commit) หรือโรลแบ็ค (rollback) โลคอลทรานแซคชั่น

ในขั้นตอนนี้จะเขียนโปรแกรมเพื่อตรวจสถานะของแต่ละเซิร์ฟเวอร์ฐานข้อมูล ซึ่งถ้าหากทั้งคู่ทำโลคอลทรานแซคชั่นเรียบร้อยก็จะคอมมิต ทั้ง 2 โลคอลทรานแซคชั่น แต่ถ้าไม่เรียบร้อยก็จะโรลแบ็ค ทั้ง 2 โลคอลทรานแซคชั่น

โปรแกรมที่ 4.36 การตรวจสถานะของแต่ละฐานข้อมูลเซิรฟเวอร์ เพื่อคอมมิตหรือโรลแบ็ค

```
if((xaRes.prepare(db2xid) == XAResource.XA_OK) &&
(xaRes_2.prepare(db2xid) != XAResource.XA_OK)){
      System.out.println("OK");
      xaRes.commit(db2xid, false);
      xaRes_2.commit(db2xid, false);
}else{
      System.out.println("Rollback");
```

```
        xaRes.rollback(db2xid);
        xaRes_2.rollback(db2xid);
}
```

ผลลัพธ์จากการทำทรานแซคชั่นแบบกระจายโอนเงินนี้ แบ่งเป็น 2 กรณีคือทำสำเร็จ และไม่สำเร็จดังรูป

```
Before running Transaction
========DB2 Server 1========
Kik             13000
Mr.X            12000
========DB2 Server 2========
Mr.Y            17000
Pao             16000
Start running Transaction...
OK
After running Transaction
========DB2 Server 1========
Kik             13000
Mr.X            9000
========DB2 Server 2========
Mr.Y            20000
Pao             16000
```

(ก)

```
Before running Transaction
========DB2 Server 1========
Kik             13000
Mr.X            12000
========DB2 Server 2========
Mr.Y            17000
Pao             16000
Start running Transaction...
========DB2 Server 1 Intermediate===
Kik             13000
Mr.X            9000
Rollback
After running Transaction
========DB2 Server 1========
Kik             13000
Mr.X            12000
========DB2 Server 2========
Mr.Y            17000
Pao             16000
```

(ข)

รูปที่ 4.35 ข้อมูลก่อนและหลังปฏิบัติทรานแซคชั่นแบบกระจาย
(ก) กรณีที่ปฏิบัติสำเร็จ
(ข) กรณีที่ปฏิบัติไม่สำเร็จ

4.7 การทดลองเขียนฟังก์ชั่นที่ผู้ใช้กำหนดเอง (User Defined Function) ในดีบีทูเพื่อทำการรู้จำใบหน้า

การทดลองนี้แบ่งเป็น 4 ขั้นตอนคือ

4.7.1 การเก็บรูปลงในดีบีทู

ในดีบีทูมีชนิดข้อมูลที่เอาไว้เก็บข้อมูลไบนารี่ ซึ่งเรียกว่าบล็อบ (BLOB) ย่อมาจาก binary large object ซึ่งสามารถเก็บข้อมูลที่เป็นภาพและเสียงได้เป็นต้น ซึ่งในการเก็บรูปนั้นจะต้องมีขั้นตอนการแปลงไฟล์รูปซึ่งมีนามสกุลต่างๆ เช่น .jpg และ .png เป็นต้น ให้เป็นอาเรย์ของไบต์เสียก่อน แล้วค่อยแปลงเป็นบล็อบจึงจะเอาข้อมูลลงในดีบีทูได้ ซึ่งในการทดลองนี้ได้ทดลองสร้าง

ตาราง Person ขึ้นมาประกอบด้วยแอตทริบิวต์ 3 ตัวคือ ID, name และ photo ซึ่งมีคำสั่งการสร้างตารางดังนี้

โปรแกรมที่ 4.37 คำสั่งเอสคิวแอลในการสร้างตาราง Person

```
create table person (
id int not null,
name char(40),
photo blob,
primary key (id))
```

ในการแปลงไฟล์รูปภาพเป็นอาเรย์ของไบต์จะใช้ฟังก์ชั่นซึ่งเขียนโดยใช้ภาษาจาวาดังนี้

โปรแกรมที่ 4.38 ฟังก์ชั่นที่แปลงไฟล์รูปภาพเป็นอาเรย์ของไบต์

```
private static byte[] readImage(String imageFileName) {
      byte[]  imageData = null;
      FileInputStream file = null;
      try {
            file = new FileInputStream(imageFileName);
            int size = file.available();
            imageData = new byte[size];
            file.read(imageData);
            if (file != null)
             file.close();
      } catch (IOException e) {
            e.printStackTrace();
      }
      return imageData;
}
```

ส่วนการเอาข้อมูลเข้าไปในตาราง Person จะใช้ฟังก์ชั่นซึ่งเขียนโดยใช้ภาษาจาวาดังนี้

โปรแกรมที่ 4.39 ฟังก์ชั่นที่เอาข้อมูลใส่ในตาราง Person

```
private static void insertData(Connection con, int id, String name,
String imageFileName) {
      try {
            byte[] imageData = readImage(imageFileName);
            Blob blob = new SerialBlob(imageData);

            PreparedStatement stmt = con.prepareStatement("INSERT
INTO person VALUES(?,?,?)");
            stmt.setInt(1, id);
            stmt.setString(2, name);
            stmt.setBlob(3, blob);
            stmt.executeUpdate();
      } catch (SQLException e) {
            e.printStackTrace();
```

```
        }
}
```

4.7.2 การเขียนฟังก์ชั่นการรู้จำใบหน้า

ในขั้นตอนนี้จะเขียนฟังก์ชั่นการรู้จำใบหน้าโดยใช้เทคนิคการวิเคราะห์องค์ประกอบหลัก (Principal Component Analysis: PCA) ซึ่งฟังก์ชั่นที่เขียนขึ้นจะต้องเหมาะสมกับการนำไปใช้งานในฐานข้อมูล โดยฟังก์ชั่นนี้จะรับพารามิเตอร์ 2 ค่าคือ ใบหน้า 2 ใบหน้า เพื่อนำไปคำนวณหา Euclidian distance ระหว่างภาพทั้งสอง

4.7.3 การนำฟังก์ชั่นการรู้จำใบหน้าลงในดีบีทู

จากฟังก์ชั่นการรู้จำใบหน้าในหัวข้อที่แล้ว การนำไปใช้ในดีบีทู จะต้องเขียนรูปแบบของฟังก์ชั่นซึ่งในการทดลองจะใช้รูปแบบของพารามิเตอร์แบบ DB2General ซึ่งอธิบายไว้ในหัวข้อ 2.5.3 จากนั้นนำไฟล์ .class ซึ่งได้มาจากการคอมไพล์โปรแกรมภาษาจาวาให้ออกมาเป็น byte code เอามาเก็บในโฟลเดอร์ที่เอาไว้เก็บฟังก์ชั่นที่ดีบีทู มีให้ เช่น /home/ec2-user/sqllib/function/

ในฟังก์ชั่นการรู้จำใบหน้านี้มีการเรียกใช้งานไลบรารี่อื่นๆ เป็นไฟล์ .JAR ด้วยคือ utils ซึ่งใช้ในการแปลงไฟล์ข้อมูลบล็อบให้เป็นรูปภาพ และ colt ซึ่งช่วยในการคำนวณที่เกี่ยวกับเวกเตอร์ และเมทริกซ์ โดยมีคำสั่งในการติดตั้งลงในดีบีทู ดังนี้

โปรแกรมที่ 4.40 คำสั่งการติดตั้งไฟล์ .JAR ลงในดีบีทู

```
CALL SQLJ.INSTALL_JAR('file:/home/ec2-
user/sqllib/function/Utils.jar', 'Utils')
CALL SQLJ.INSTALL_JAR('file:/home/ec2-user/sqllib/function/colt.jar',
'colt')
```

จากคำสั่งข้างต้น พารามิเตอร์ตัวแรกคือที่อยู่ของไฟล์ ตัวที่สองคือชื่อเรียกที่ตั้งขึ้นมา จากนั้นใช้คำสั่ง SQL เพื่อสร้างฟังก์ชั่นที่ผู้ใช้กำหนดเองดังนี้

โปรแกรมที่ 4.41 คำสั่งเอสคิวแอลการสร้างฟังก์ชั่น euclidianDistance

```
create function euclidianDistance (Blob,Blob)
returns double
language Java
external name 'DB2FaceCompare!faceCompare'
PARAMETER STYLE db2general
```

จากคำสั่งข้างต้นเป็นการสร้างฟังก์ชั่นชื่อ euclidianDistance มีการรับพารามิเตอร์ 2 ค่า ชนิดข้อมูลเป็นบล็อบชนิดข้อมูลที่ออกมาเป็นดับเบิล ซึ่งก็คือ Euclidian distance ที่คำนวณได้จากรูป 2 รูปนั้น ภาษาที่ใช้คือจาวา ชื่อคลาส และเมธอดในการทำงานคือ DB2FaceCompare และ faceCompare ตามลำดับ และสุดท้ายคือ รูปแบบของพารามิเตอร์เป็นแบบ DB2General

จากนั้นจะสร้างฟังก์ชั่น faceCompare ขึ้นมาบนดีบีทู โดยฟังก์ชั่นนี้จะช่วยหาว่าแถวใดที่คำนวณค่า Euclidian distance เทียบกับใบหน้าที่รับเข้ามาแล้วมีค่าน้อยที่สุด ซึ่งในการเขียนจะใช้ภาษา SQL PL (SQL Procedural Language) ดังนี้

โปรแกรมที่ 4.42 คำสั่ง SQL การสร้างฟังก์ชั่น faceCompare

```
CREATE FUNCTION faceCompare (a Blob,b Blob)
RETURNS int
LANGUAGE sql
BEGIN
      DECLARE minDistance double;
      DECLARE distance double;
      SET minDistance = (select min(euclidianDistance(photo,b)) from
people);
      SET distance = euclidianDistance(a,b);

      IF (distance = minDistance) THEN RETURN 1;
      ELSE RETURN 0;
      END IF;
END
```

ในฟังก์ชั่นดังกล่าวจะมีพารามิเตอร์ 2 ค่าคือ a คือรูปที่อยู่ในฐานข้อมูล ส่วน b คือรูปที่ต้องการจะจับคู่ โดยจะมีค่าที่ออกมาได้ 2 ค่าคือ 1 ถ้ารูปในแถวนั้นมี Euclidian distance น้อยที่สุดในทุกๆ รูป และ 0 ถ้าไม่ใช่รูปที่ Euclidian distance น้อยที่สุด

ตัวอย่างการใช้งานฟังก์ชั่น faceCompare เช่น SELECT * FROM person WHERE faceCompare(photo,input_face) = 1 โดย photo คือชื่อของ attribute ที่เก็บรูป ส่วน input_face คือรูปที่ต้องการจะเปรียบเทียบ

4.7.4 การทดสอบฟังก์ชั่นการรู้จำใบหน้า

สำหรับการทดสอบจะใช้ตาราง Person ซึ่งอธิบายไว้ในหัวข้อ 4.7.1 โดยมีข้อมูลของคน 10 คน อยู่ในตาราง และจะทดสอบการใช้ฟังก์ชั่น euclidianDistance และ faceCompare ดังนี้

4.7.4.1 การทดสอบฟังก์ชั่น euclidianDistance

ทดลองแสดงไอดี, ชื่อ และ Euclidian distance เมื่อเทียบกับรูปตัวอย่าง ซึ่งในโปรแกรมนี้จะทดลองด้วยรูปของ Hill

โปรแกรมที่ 4.43 โปรแกรมที่แสดงไอดี, ชื่อ และ Euclidian distance เมื่อเทียบกับรูปตัวอย่าง

```
private static void queryData(Connection con) throws SQLException,
IOException {
      byte[] imageData = readImage("C:\\Users\\user\\Desktop\\face
                  recog image\\all image\\hill4.png");
      Blob blob = new SerialBlob(imageData);

      PreparedStatement stmt = con.prepareStatement("select id, name,
                  euclidianDistance(photo,?) from people order by 3");
      stmt.setBlob(1, blob);

      ResultSet rs = stmt.executeQuery();
      while(rs.next()){
            System.out.println("id: " + rs.getInt("id") + "\tName: "
                        + rs.getString("name").trim() + "\tEuclidian
                              distance : " + rs.getDouble(3));
      }
}
```

ผลลัพธ์จะได้ ลำดับคนที่มีหน้าใกล้เคียงกับหน้าที่เข้ามาเทียบมากที่สุดจากมากไปน้อย (Euclidian distance ยิ่งน้อยยิ่งเหมือน) ซึ่งจะพบว่าใกล้เคียงกับ Hill มากที่สุดดังรูป

```
id: 7   Name: hill      Euclidian distance : 1.0312658437358317
id: 5   Name: erin      Euclidian distance : 1.2024126598796323
id: 4   Name: andy      Euclidian distance : 1.2270132229724524
id: 3   Name: andrew    Euclidian distance : 1.2642668162478883
id: 9   Name: kik       Euclidian distance : 1.270471567710302
id: 1   Name: amber     Euclidian distance : 1.272098334600651
id: 8   Name: zach      Euclidian distance : 1.3257740784699816
id: 6   Name: gabe      Euclidian distance : 1.3623167247447678
id: 2   Name: amy       Euclidian distance : 1.5161284290460606
```

รูปที่ 4.36 ผลลัพธ์จากฟังก์ชั่น euclidianDistance

4.7.4.2 การทดสอบฟังก์ชั่น faceCompare

ทดลองแสดงไอดี และชื่อของคนที่มีหน้าเหมือนหน้าที่ต้องการมากที่สุด ซึ่งในโปรแกรมนี้จะทดลองด้วยรูปของ Hill

โปรแกรมที่ 4.44 โปรแกรมแสดงไอดี และชื่อของคนที่มีหน้าเหมือนหน้าที่ต้องการมากที่สุด

```
private static void queryData(Connection con) throws SQLException,
IOException {
      byte[] imageData = readImage("C:\\Users\\user\\Desktop\\face
recog              image\\all image\\hill4.png");
      Blob blob = new SerialBlob(imageData);

      PreparedStatement stmt = con.prepareStatement("select * from
people              where faceCompare(photo,?) = 1");
      stmt.setBlob(1, blob);

      ResultSet rs = stmt.executeQuery();
      while(rs.next()){
            System.out.println("id: " + rs.getInt("id") + "\tName: "
                  + rs.getString("name").trim());
      }
}
```

ผลลัพธ์จะได้ id: 7 Name: hill ซึ่งมีหน้าที่คล้ายกับหน้าที่ต้องการมากที่สุด

จากการทดลองเรื่อง UDF การรู้จำใบหน้าที่ผ่านมา จะเห็นว่าเป็นฟังก์ชั่นที่มีประโยชน์ โดยสามารถประยุกต์ใช้ในงานต่างๆ ได้เช่น งานบริการลูกค้าซึ่งต้องกรอกแบบฟอร์มก่อนเข้ารับบริการก็สามารถรับบริการได้เร็วขึ้นด้วยการใช้หน้าของลูกค้าในการยืนยันตนโดยไม่ต้องกรอกแบบฟอร์ม หรือแม้แต่งานรักษาความปลอดภัย ก็สามารถติดตั้งกล้องที่ตรวจจับหน้าคนแล้วตรวจสอบว่าคนคนนั้นเป็นคนใน หรือว่าเป็นบุคคลภายนอกได้ ทำให้การรักษาความปลอดภัยทำได้ดียิ่งขึ้นเป็นต้น

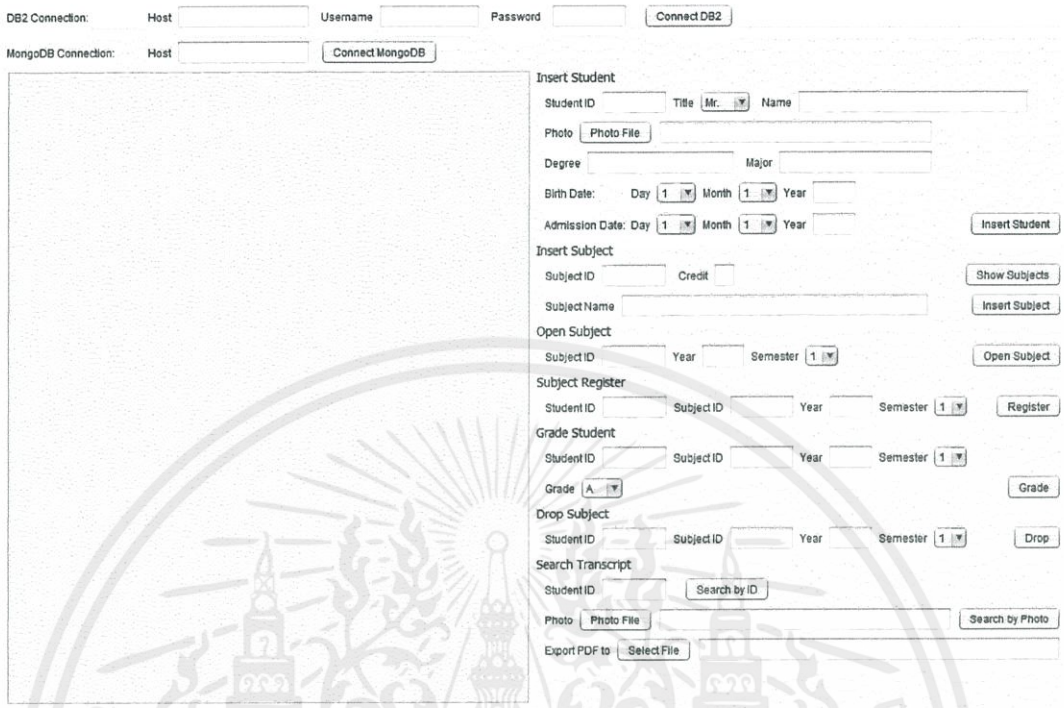
4.8 การใช้งานระบบจัดเก็บข้อมูลหลายรูปแบบในระบบงานทะเบียนนักศึกษาจำลอง

4.8.1 ส่วนประกอบของระบบงานทะเบียนนักศึกษาจำลอง

ระบบงานทะเบียนนักศึกษาจำลองเป็นระบบที่ประกอบด้วยส่วนประกอบคือ

4.8.1.1 ส่วนที่ติดต่อกับผู้ใช้

ส่วนนี้ถูกสร้างขึ้นมาเป็นเว็บไซด์ ซึ่งใช้บริการของ AWS Elastic Beanstalk แล้วภายในจะเป็นจาวาแอพเพล็ต ซึ่งฝังอยู่บนเว็บไซด์



รูปที่ 4.37 หน้าของแอพเพล็ต งานทะเบียนนักศึกษาจำลอง

ในหน้าจอแอพเพล็ต นี้แบ่งเป็น 3 ส่วนใหญ่ได้แก่

1) ส่วนบน: เป็นส่วนที่สร้างการเชื่อมต่อกับดีบีทู และมองโกดีบี ซึ่งจะต้องเชื่อมต่อก่อนปฏิบัติงานอื่นๆ
2) ส่วนขวา: เป็นเมนูในการทำงานต่างๆ ได้แก่ การเพิ่มนักศึกษา, เพิ่มรายวิชา, เปิดสอนวิชา, การลงทะเบียนของนักศึกษา, การให้เกรดนักศึกษา, การดรอป และการค้นหาทรานสคริปด้วยรหัสนักศึกษาหรือหน้านักศึกษา
3) ส่วนซ้าย: เป็นหน้าจอแสดงรายละเอียดของการทำรายงานต่างๆ

4.8.1.2 ส่วนที่เก็บข้อมูลที่ระบบใช้

ส่วนที่เก็บข้อมูลใช้ระบบจัดเก็บข้อมูลหลายรูปแบบ ซึ่งใช้ระบบการจัดการฐานข้อมูลดีบีทูและมองโกดีบี ในการเก็บข้อมูล โดยในดีบีทู จะเก็บข้อมูลนักศึกษา, รายวิชา, วิชาที่เปิดในแต่ละเทอม และการลงทะเบียนของนักศึกษา ทั้งหมดลงในตาราง 4 ตาราง ส่วนมองโกดีบี ได้เก็บทรานสคริปของนักศึกษาแต่ละคนไว้ เพื่อเวลาจัดรูปแบบเป็น ไฟล์ PDF จะได้ไม่ต้อง join ตารางในดีบีทู

4.8.2 ความสามารถของระบบ

เป็นระบบที่มีผู้ใช้งานคือเจ้าหน้าที่สำนักทะเบียน ซึ่งสามารถเพิ่มนักศึกษา, เพิ่มรายวิชา, เปิดสอนวิชา, การลงทะเบียนของนักศึกษา, การให้เกรดนักศึกษา, การดรอป และการค้นหาทรานสคริปด้วยรหัสนักศึกษาหรือหน้านักศึกษาได้ โดยการปฏิบัติงานบางอย่างได้แก่ การเพิ่มนักศึกษา,การลงทะเบียนของนักศึกษา, การให้เกรดนักศึกษา, การดรอป จำเป็นต้องเกี่ยวข้องกับระบบจัดการฐานข้อมูลทั้งดีบีทู และมองโกดีบี ซึ่งการปฏิบัติงานดังกล่าวจำเป็นต้องรักษาความถูกต้องของข้อมูลเพื่อไม่ให้เกิดภาวะที่ข้อมูลไม่ตรงกันซึ่งอาจเกิดจากการแก้ไขข้อมูลในดีบีทู ทำสำเร็จ แต่การแก้ไขข้อมูลในมองโกดีบี ทำไม่สำเร็จ ดังนั้นในระบบนี้จึงจำเป็นต้องมีคุณสมบัติความเป็นหนึ่งเดียว (Atomicity)

4.8.2.1 คุณสมบัติความเป็นหนึ่งเดียว (Atomicity)

ในการปฏิบัติงานที่ต้องเกี่ยวข้องกับดีบีทู และมองโกดีบี จำเป็นต้องมีคุณสมบัติความเป็นหนึ่งเดียว ซึ่งทำขึ้นมาเองภายในระบบโดยที่หลังจากปฏิบัติงานในดีบีทู เสร็จให้มีการแจ้งว่าทำงานเสร็จเรียบร้อยหรือไม่ ถ้าไม่ก็จะยกเลิกการปฏิบัติงานในมองโกดีบี และแจ้งผู้ใช้ แต่ถ้าปฏิบัติงานในดีบีทู เรียบร้อยก็สามารถปฏิบัติงานในมองโกดีบี ต่อได้เลย โดยหลังจากปฏิบัติงานในมองโกดีบี เสร็จก็ให้มีการแจ้งว่าทำงานเสร็จเรียบร้อยหรือไม่เช่นเดียวกัน ถ้าไม่งานที่ปฏิบัติในดีบีทู จะถูกโรลแบ็ค แต่ถ้าปฏิบัติงานในมองโกดีบี เรียบร้อย งานที่ปฏิบัติในดีบีทู จะถูกคอมมิต

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุปของโครงงาน

ในการพัฒนาโครงงานการพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์ได้ข้อสรุปของโครงงานดังนี้

จากการศึกษาคุณสมบัติของระบบจัดการฐานข้อมูลมองโกดีบี ซึ่งใช้ฐานข้อมูลแบบเอกสาร (Document Database) พบว่ามีความเป็น dynamic schema โดยผู้ใช้งานสามารถเก็บข้อมูลต่างชนิดกันภายในคอลเลคชั่นเดียวกันได้ การเพิ่ม หรือลบ field ต่าง ๆ ทำได้โดยไม่กระทบกับข้อมูลอื่น ๆ ซึ่งต่างจาก ระบบจัดการฐานข้อมูลดีบีทู ซึ่งใช้ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) คือเมื่อสร้างตารางจะต้องกำหนด Attribute และชนิดข้อมูลก่อนที่จะใส่ข้อมูลลงตาราง

ในเรื่องความสามารถในการค้นหา (Query) ของมองโกดีบีพบว่าสามารถค้นหาได้ผ่านทางคำสั่ง find และมีความสามารถในการทำ Aggregation ได้โดยการใช้ฟังก์ชั่น Pipeline และ Map-reduce ซึ่ง Map-reduce เป็นวิธีในการประมวลผลแบบขนาน โดยผู้ใช้จะต้องเขียนฟังก์ชั่นแมพและฟังก์ชั่นรีดิวส์ แล้วมองโกดีบีจะจัดการให้สามารถทำการประมวลผลแบบขนานได้ แต่ว่ามองโกดีบีไม่สามารถทำการ join ระหว่างคอลเลคชั่นได้ ทำให้ผู้ใช้ต้องเก็บข้อมูลลงในคอลเลคชั่นเดียว

ในเรื่องความสามารถเกี่ยวกับการให้ผู้ใช้สามารถสร้างชนิดตัวแปร และฟังก์ชั่นเองได้นั้น ในดีบีทูสามารถทำได้ทั้งคู่ แต่ว่ามองโกดีบีไม่สามารถสร้างชนิดตัวแปรเองได้ แต่สามารถสร้างฟังก์ชั่นเองได้ และมองโกดีบียังมีการเก็บข้อมูลเป็นพิกัดทางภูมิศาสตร์ และมีฟังก์ชั่นทางภูมิศาสตร์ให้ใช้งานอีกด้วย

การสร้างฟังก์ชั่นการรู้จำใบหน้าโดยใช้เทคนิคการวิเคราะห์องค์ประกอบหลัก (PCA) สามารถใช้ร่วมกันในดีบีทูได้ ซึ่งมีประโยชน์ในงานที่ต้องการเปรียบเทียบหน้าคน ซึ่งในอนาคตจะมีเทคนิคการรู้จำใบหน้าที่ดีกว่านี้ก็สามารถเปลี่ยนแปลงฟังก์ชั่นการรู้จำใบหน้าในดีบีทูเลย โดยที่ไม่ต้องแก้ไขโปรแกรมที่ติดต่อกับฐานข้อมูลนั้น

โดยการใช้งานระบบจัดการฐานข้อมูลทั้งสองบนคลาวด์นั้นยังไม่พบปัญหาใด ๆ ระหว่างการใช้งาน เพียงแต่ว่ายังไม่สามารถใช้ระบบดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor) ซึ่งทำหน้าที่ประสานงานระหว่างระบบจัดการฐานข้อมูลต่าง ๆ บนคลาวด์ได้ เนื่องจากมองโกดีบีไม่สามารถทำทรานแซกชั่นได้ จึงแก้ไขโดยการสร้างคุณสมบัติความเป็นหนึ่งเดียวบนโปรแกรมประยุกต์แทน

ในการทดลองสร้างโปรแกรมประยุกต์ขึ้นมาใช้ระบบจัดเก็บข้อมูลหลายรูปแบบบนคลาวด์นั้น พบว่าการนำมองโกดีบีมาเก็บเอกสารเช่นทรานสคริป มีประโยชน์ในการเรียกค้นทรานสคริป เพราะว่าจะได้ไม่ต้องจอยหลายตารางในดีบีทู

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข

1) การใช้งานระบบควบคุมการประมวลผลทรานแซกชั่นบนคลาวด์ ในการทดลองที่ใช้ JTA/XA นั้นสามารถทำทรานแซกชั่นแบบกระจายระหว่างดีบีทูบนคลาวด์ได้ แต่ว่าไม่สามารถทำบนมองโกดีบีได้ จึงแก้ไขโดยการสร้างคุณสมบัติความเป็นหนึ่งเดียวบนโปรแกรมประยุกต์แทน
2) ระบบจัดการฐานข้อมูลทั้งดีบีทู และมองโกดีบี ไม่มีคำสั่งในการเปรียบเทียบข้อมูลประเภทมัลติมีเดียเช่นรูปภาพ ให้ใช้ (แต่สามารถเก็บข้อมูลประเภทนี้ได้) แต่โครงงานนี้ต้องการพัฒนาโปรแกรมประยุกต์ให้สามารถใช้ข้อมูลประเภทมัลติมีเดียให้ได้ ดังนั้นจึงเลือกเขียนฟังก์ชั่นการรู้จำใบหน้าบนดีบีทูขึ้นมาเอง
3) การใช้งานฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นเองในดีบีทู พบว่าไม่สามารถทำได้บนบริการที่อเมซอนมีไว้ให้ (Amazon Machine Image) จึงต้องติดตั้งโปรแกรมดีบีทูเองโดยการสร้างเซิฟเวอร์จำลองบนบริการอเมซอนอีซีทู
4) ในการใช้งานบริการคลาวด์นั้นมีค่าใช้จ่ายซึ่งสัมพันธ์กับปริมาณการใช้งาน และความสามารถของเซิฟเวอร์จำลองที่สร้างขึ้น จึงต้องศึกษารายละเอียดให้ชัดเจน รวมทั้งลดค่าใช้จ่ายโดยการปิดเซิฟเวอร์จำลองเมื่อไม่ได้ใช้งาน หรือใช้เซิฟเวอร์จำลองในรุ่นที่ใช้ได้ฟรี 750 ชั่วโมงต่อเดือน ที่อเมซอนมีให้
5) การใช้งานเซิฟเวอร์จำลองบนคลาวด์นั้นจำเป็นจะต้องติดต่อผ่านระบบเครือข่ายอินเตอร์เน็ต ซึ่งถ้าระบบเครือข่ายอินเตอร์เน็ตไม่สามารถใช้งานได้ ก็จะไม่สามารถใช้งานเซิฟเวอร์จำลองนั้นได้เช่นเดียวกัน

5.3 แนวทางในการพัฒนาต่อ

การพัฒนาและการประยุกต์ระบบข้อมูลหลายรูปแบบบนคลาวด์สามารถนำไปประยุกต์ใช้งานได้ ซึ่งสามารถศึกษาและพัฒนาต่อได้ดังนี้

1) ศึกษาและใช้งาน ระบบดูแลการประมวลผลทรานแซกชั่น (Transaction Processing Monitor) เพื่อให้สามารถทำ ทรานแซกชั่นที่มีการติดต่อระหว่างระบบจัดการฐานข้อมูลดีบีทู (DB2) และมองโกดีบี (MongoDB)

2) ศึกษาการทำฐานข้อมูลแบบกระจาย (Distributed Database) และการทำซ้ำ (Replication) ในฐานข้อมูลบนคลาวด์
3) ศึกษาเทคนิคการรู้จำหน้าคนที่ดีมากกว่าเดิม เพื่อเพิ่มความแม่นยำเมื่อต้องการเทียบหน้าคน
4) ศึกษาการใช้งานฟังก์ชั่นที่ผู้ใช้กำหนดเองในงานด้านอื่นๆ เช่นการเปรียบเทียบเสียงและลายนิ้วมือ เป็นต้น
5) ศึกษาการนำรูปแบบการจัดเก็บข้อมูลอื่นๆ เช่นเอ็กซ์เอ็มแอล (XML) มาใช้ประโยชน์ในระบบจัดเก็บข้อมูลหลายรูปแบบเช่นการเก็บข้อมูลเอ็กซ์เอ็มแอลบนมองโกดีบี
6) ศึกษาการนำเอกสารในงานต่างๆ ที่ต้องใช้บ่อย และเป็นงานที่เป็นการอ่านข้อมูลเป็นส่วนใหญ่ เช่น ใบเสร็จ, ใบส่งของ และใบกำกับภาษี เป็นต้น มาเก็บลงในมองโกดีบี เพื่อเวลาที่เรียกดูข้อมูลจะได้ไม่ต้องจอยตาราง

บรรณานุกรม

[1] John Wood. **"Polyglot Persistence – Two Great Tastes That Taste Great Together"** [Online] Available : http://www.slideshare.net/jwoodslideshare/polyglot-persistence -two-great-tastes-that-taste-great-together-4625004

[2] Mr.Nutchanon Leelapornudom, Ms. Tipwimon Jiwiriyawat, **"Transaction processing on DB2 cloud database system"**, 2012

[3] Wikipedia. (2013). **"Java Transaction API"** [Online] Available : http://en.wikipedia.org/wiki/Java_Transaction_API

[4] Brightpattern. **"Separation of Responsibilities"** [Online] Available : http://www.brightpattern.com/2013/clearest-explanation-yet-of-saas-paas-iaas/

[5] AWS Amazon Web Services. **"Amazon EC2"** [Online] Available : http://aws.amazon.com/ec2/

[6] AWS Amazon Web Services. **"AWS Elastic Beanstalk (Beta)"** [Online] Available : http://aws.amazon.com /elasticbeanstalk/

[7] IBM. **"Java user-defined scalar functions"** [Online] Available : http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=%2Frzaha%2Fwriteudf.htm

[8] MongoDB Documentation Release 2.4.6. **"MongoDB Documentation Project"**, 2013

[9] MongoDB Manual 2.4. **"Replication"** [Online] Available : http://docs.mongodb.org/manual/replication/

[10] MongoDB Manual 2.4. **"Sharding"** [Online] Available : http://docs.mongodb.org/manual/sharding/

[11] MongoDB Manual 2.4. **"GridFS"** [Online] Available : http://docs.mongodb.org/manual/core/gridfs/

[12] MongoDB Manual 2.4. **"2d Index Internals"** [Online] Available : http://docs.mongodb.org/manual/core/geospatial-indexes/

[13] MongoDB Manual 2.4. **"MongoDB CRUD Operations"** [Online] Available : http://docs.mongodb.org/manual/crud/

[14] Reference Operators. **"Aggregation Framework Operators"** [Online] Available : http://docs.mongodb.org/manual/reference/aggregation/operators/

[15] Aggregation concepts. **"Aggregation Pipeline"** [Online] Available : http://docs.mongodb.org/manual/core/aggregation-pipeline/

[16] Jeffrey Dean, Sanjay Ghemawat. **"MapReduce: Simplified Data Processing on Large Clusters"**, 2004

[17] WebMapReduce 2.0 documentation. **"What is Map-Reduce?"** [Online] Available : http://webmapreduce.sourceforge.net/docs/using/intro.html

[18] Aggregation commands, mapReduce. **"mapReduce"** [Online] Available : http://docs.mongodb.org/manual/reference/command/mapReduce/

[19] Aggregation concepts. **"Map-Reduce"** [Online] Available : http://docs.mongodb.org/manual/core/map-reduce/

[20] รุ้งนภา จิ่นจันทร์, วโรธร เชยโภคา. **"การตรวจสอบระบุตัวบุคคลด้วยโทรศัพท์มือถือ"**,2012

[21] ความรู้พื้นฐานของระบบรู้จำใบหน้า

[22] Dr. Andrew Davison. **"Chapter VBI-10. Face Recognition"** [Online] Available : http://fivedots.coe.psu.ac.th/~ad/jg/nui08/

ภาคผนวก ก
คู่มือการติดตั้ง

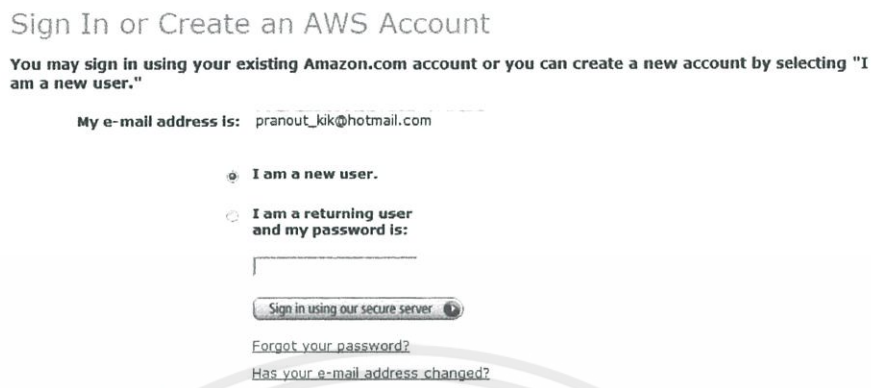
ก1 การลงทะเบียนใช้งาน Amazon Web Services

มีขั้นตอนการลงทะเบียนดังนี้

1) เปิดเว็บเบราว์เซอร์ เข้าไปที่เว็บไซต์ http://aws.amazon.com/ แล้วคลิกปุ่ม Sign Up

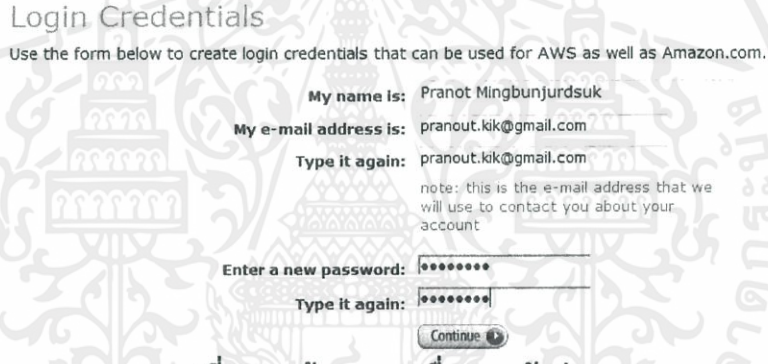
รูปที่ ก.1 หน้าเว็บไซต์ http://aws.amazon.com/

2) จะพบหน้าจอการเข้าสู่ระบบ ให้ผู้ใช้เลือก I am a new user. และกรอกอีเมล์ลงไปในช่อง My e-mail address is แล้วคลิก Sign in using our secure server



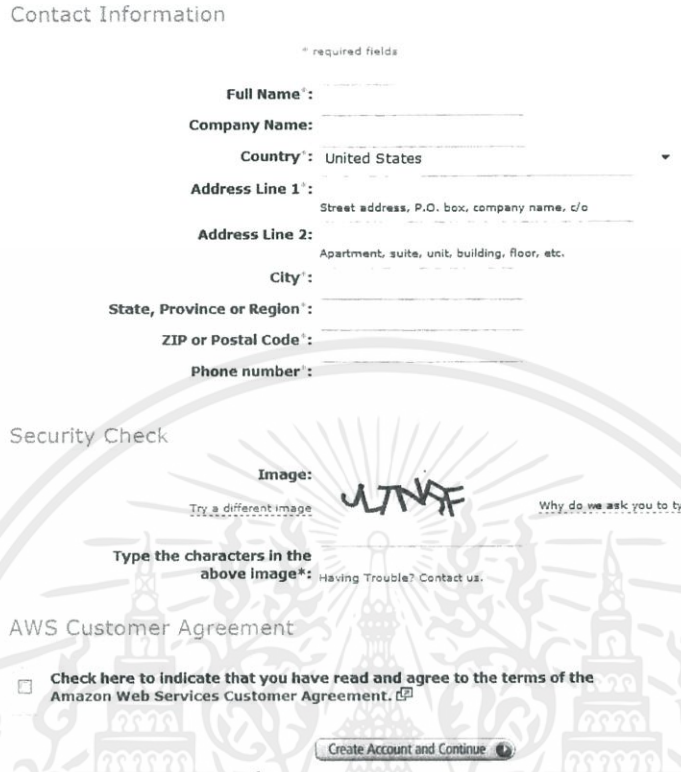
รูปที่ ก.2 หน้าจอการเข้าระบบ Amazon Web Services

3) กรอกชื่อ และรหัสผ่านที่ต้องการ



รูปที่ ก.3 หน้าจอกรอกชื่อและรหัสผ่าน

4) กรอกรายละเอียดข้อมูลส่วนตัว และยอมรับข้อตกลงการใช้งาน แล้วคลิก Create Account and Continue

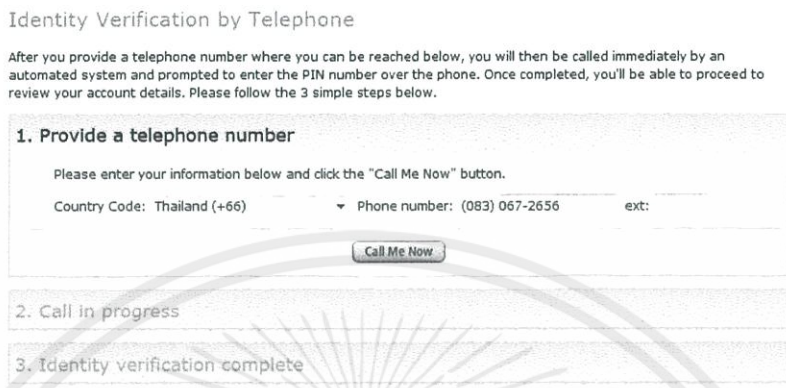


รูปที่ ก.4 หน้าจอกรอกข้อมูลส่วนตัว

5) กรอกรายละเอียดการชำระเงิน

รูปที่ ก.5 หน้าจอกรอกรายละเอียดการชำระเงิน

6) จากนั้นจะทำการยืนยันตัวบุคคลผ่านทางโทรศัพท์



รูปที่ ก.6 หน้าจอการยืนยันตนผ่านโทรศัพท์

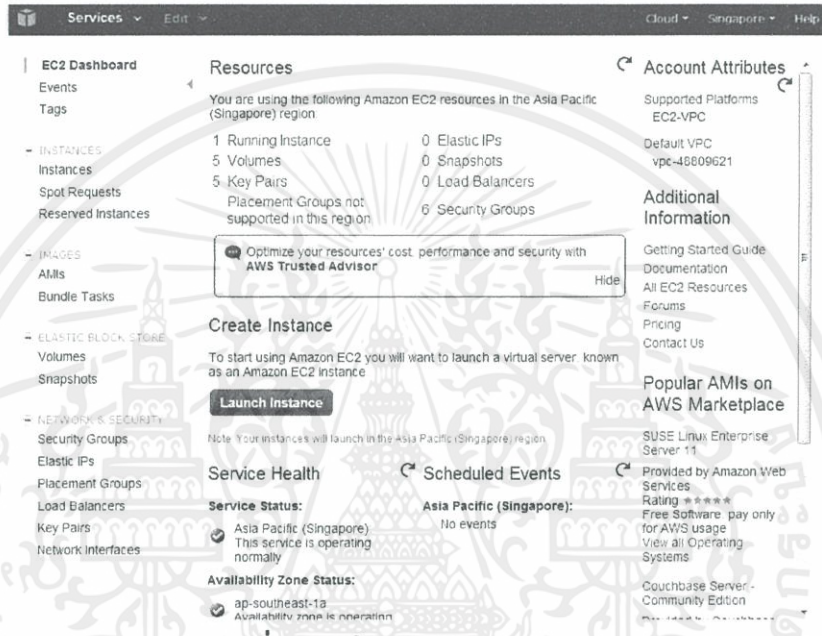
7) เมื่อยืนยันตนเรียบร้อยแล้ว ก็จะสามารถใช้บัญชีนั้นเข้าไปให้งานบริการต่าง ๆ ของ Amazon Web Services ได้

รูปที่ ก.7 บริการต่าง ๆ ของ Amazon Web Services

ก2 การสร้างเซิฟเวอร์จำลองบน Amazon EC2

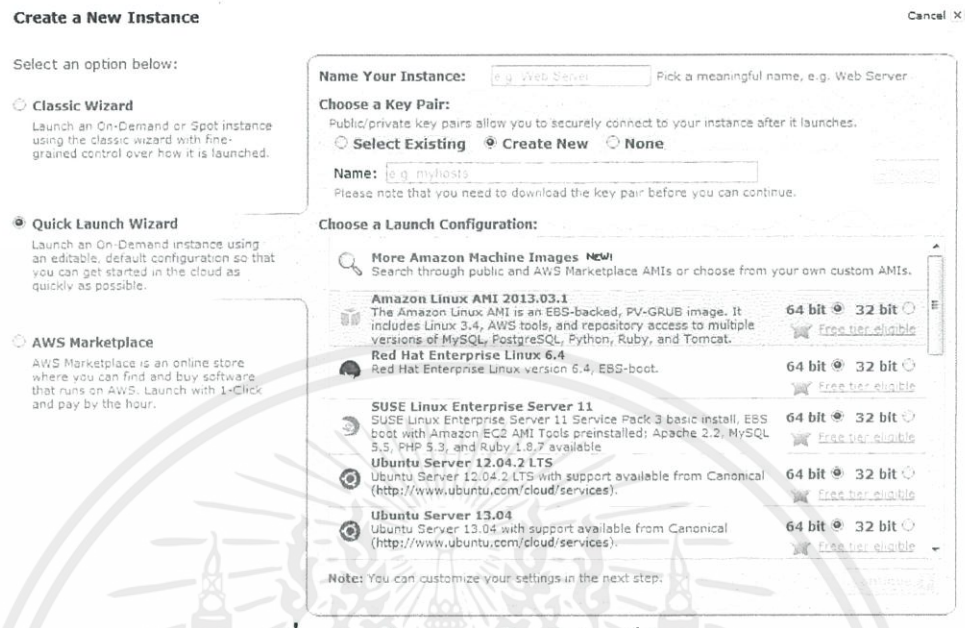
มีขั้นตอนดังนี้

1) จากหน้าจอแสดงบริการต่าง ๆ ของ Amazon Web Services ให้เลือก EC2 จะประกฏ EC2 Dashboard สรุปสถานการณ์ใช้งาน



รูปที่ ก.8 หน้าจอ EC2 Dashboard

2) สร้างเซิฟเวอร์จำลองโดยการคลิก Launch Instance แล้วเลือกวิธีการสร้าง โดยมี 3 วิธีคือ Classic Wizard คือให้ผู้ใช้เลือกระบบปฏิบัติการ, หน่วยประมวลผล, หน่วยความจำ, key สำหรับเข้าใช้งาน และบริการเสริมอื่น ๆ เองทั้งหมด Quick Launch Wizard คือ ผู้ใช้เพียงแค่เลือกระบบปฏิบัติการ แล้วปรับแต่งฮาร์ดแวร์ และ AWS Marketplace ในการซื้อ AMI ซึ่งเป็นเซิฟเวอร์จำลอง พร้อมการติดตั้งซอร์ฟแวร์ที่เลือก โดยในกรณีนี้ให้ใช้วิธี Quick Launch Wizard โดยเลือกระบบปฏิบัติการ, ตั้งชื่อเซิฟเวอร์ และสร้าง key สำหรับเข้าใช้งาน

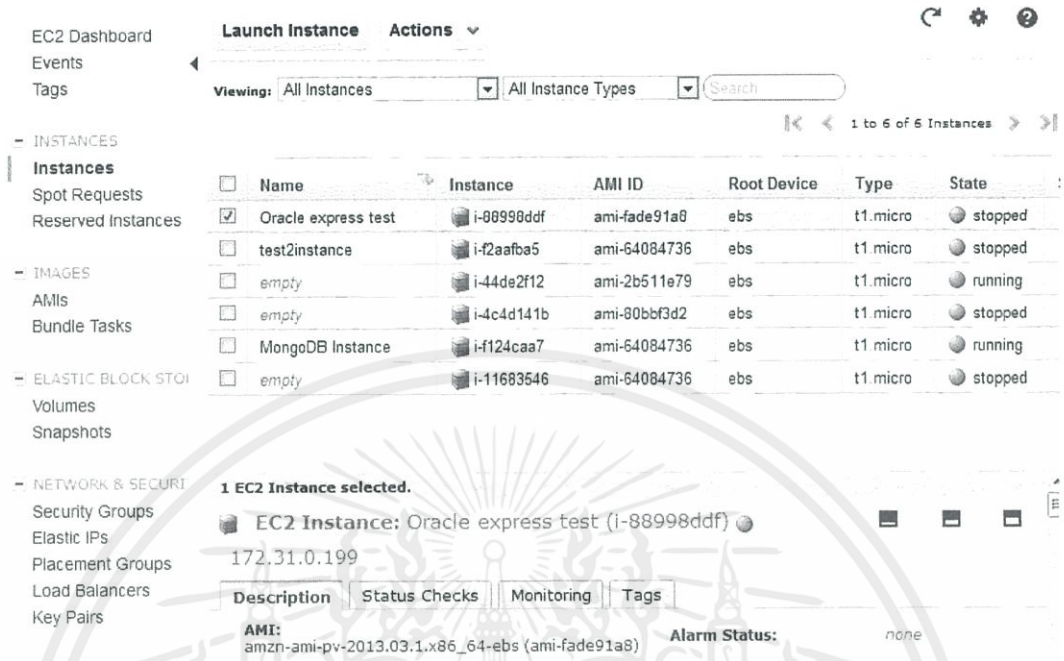


รูปที่ ก.9 หน้าจอการสร้างเซิฟเวอร์จำลอง

3) จากนั้นจะมีรายละเอียดของเซิฟเวอร์จำลองแสดงขึ้นมา ซึ่งผู้ใช้สามารถปรับแต่งเองได้ เมื่อปรับแต่งเสร็จแล้วให้คลิก Launch

รูปที่ ก.10 หน้าการการปรับแต่งเซิฟเวอร์จำลอง

4) เซิฟเวอร์จำลองทั้งหมดที่สร้างมาจะแสดงไว้ที่แท็บ Instances



รูปที่ ก.11 หน้าจอแสดงเซิฟเวอร์จำลองที่สร้างขึ้น

ก3 การติดตั้งดีบีทู (DB2) ลงเซิฟเวอร์จำลอง

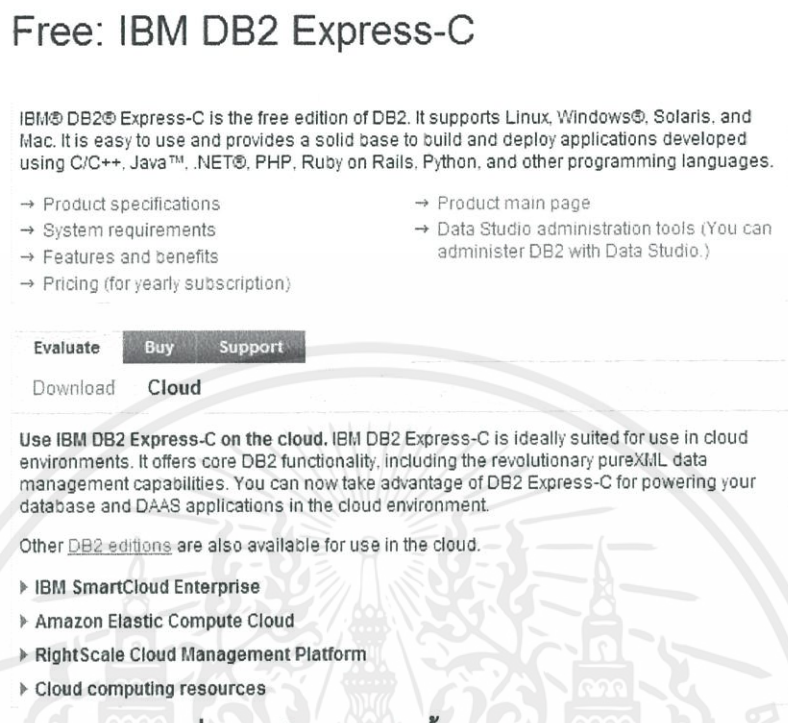
การติดตั้งดีบีทูจาก AMI

ในการติดตั้งดีบีทู จะใช้วิธีลงจาก AMI (Amazon Machine Image) ซึ่งเป็นการสร้างเซิฟเวอร์จำลอง พร้อมทั้งซอร์ฟแวร์จากผู้ให้บริการต่าง ๆ ซึ่ง IBM ก็มีเช่นกันโดยมี DB2 ในรุ่นต่าง ๆ คือ DB2 Express, DB2 Express-C และ DB2 Workgroup ซึ่งในกรณีนี้จะติดตั้ง DB2 Express-C

1) เข้าเว็บไซด์

   http://www.ibm.com/developerworks/downloads/im/db2express/cloud.html

   ซึ่งจะมีข้อมูลการติดตั้ง DB2 รุ่นต่าง ๆ บนคลาวด์ เช่น IBM SmartCloud และ Amazon Elastic Compute Cloud

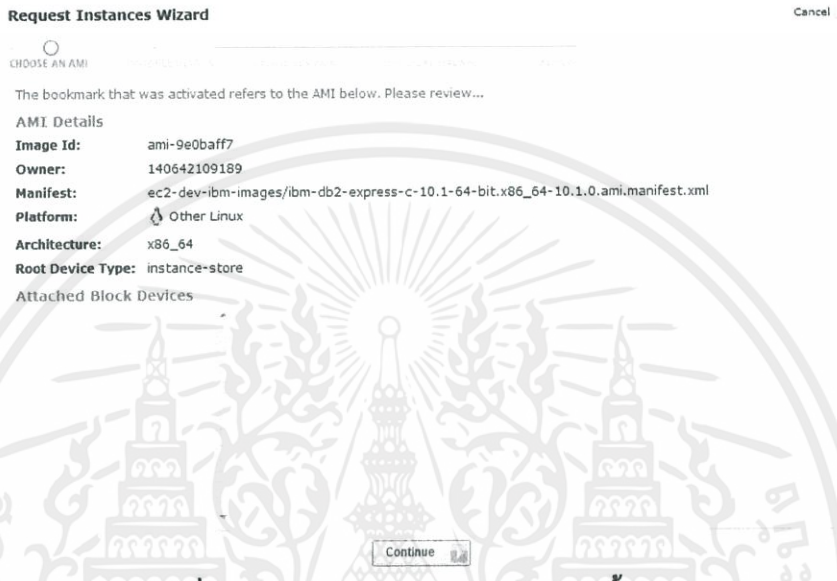


รูปที่ ก.12 ข้อมูลการติดตั้ง DB2 Express-C

2) ในเมนู Amazon Elastic Compute Cloud ผู้ใช้สามารถเลือกลิงค์ไปยังการติดตั้ง AMI ได้ โดยมีให้เลือกทั้ง DB2 Express-C แบบ 32 บิต และ 64 บิต ซึ่งในกรณีนี้เลือก 64 บิต ซึ่งจะปรากฏหน้าจอดังรูป

รูปที่ ก.13 AMI ของ DB2 Express-C

3) ก่อนที่จะติดตั้ง AMI ผู้ใช้จะต้องคลิกซื้อผลิตภัณฑ์นี้ก่อนในหัวข้อ Purchase Information
4) คลิก Launch AMI แล้วเลือกไซต์ (Site) ที่ต้องการจะติดตั้ง จากนั้นผู้ใช้จะเข้ามาที่บริการ EC2 ในการปรับแต่งเซิฟเวอร์ดังรูป



รูปที่ ก.14 การปรับแต่งเซิฟเวอร์ขณะติดตั้ง AMI

5) เมื่อปรับแต่งเสร็จแล้วก็จะได้เซิฟเวอร์จำลองพร้อมทั้ง DB2 Express-C มาโดยการเชื่อมต่อนั้น ให้ใช้โปรแกรมประเภท SSH Client เช่น PuTTY

การติดตั้งดีบีทูเองบน EC2

1) สร้าง VM บน EC2 ขึ้น โดยเลือก OS ที่ลง DB2 ได้เช่น Red Hat
2) โหลด DB2 Express-C มาติดตั้งบน VM (จะอยู่ในโฟลเดอร์ /opt/ibm/db2/V10.5)
3) สร้าง users:
   db2 instance = ec2-user (ตอนสร้าง VM จะมี user นี้ขึ้นมาให้เอง)
   db2 fence user = db2fenc1
   db2 admin = db2das1
4) สร้าง db2 administration server instance ด้วยคำสั่ง /opt/ibm/db2/V10.5 /instance/dascrt -u db2das1
5) สร้าง db2 instance ด้วยคำสั่ง /opt/ibm/db2/V10.5/instance/db2icrt -a server -u db2fenc1 ec2-user

6) เปิด port ที่ใช้ติดต่อกับ DB2 เป็น port 50001 โดยการแก้ไขไฟล์ /etc/services

```
db2c_ec2-user    50001/tcp                # DB2 connection service port
```

รูปที่ ก.15 การติดต่อกับ DB2 โดยใช้ port 50001

7) แก้ไข firewall ให้อนุญาตการติดต่อด้วย port 50001 โดยใช้คำสั่ง iptables -I INPUT -p tcp –-dport 50001 -j ACCEPT จะได้ iptable เป็นดังนี้

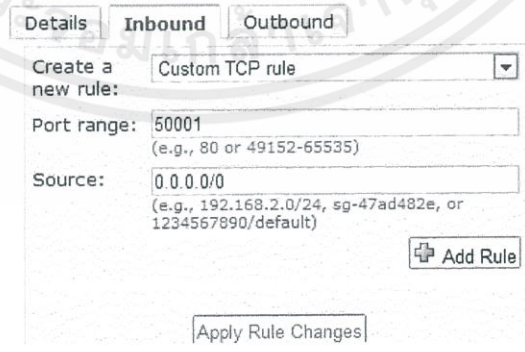
```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:db2c_ec2-us
er
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTAB
LISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere            state NEW tcp dpt:s
sh
REJECT     all  --  anywhere             anywhere            reject-with icmp-ho
st-prohibited
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:db2c_ec2-us
er

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere            reject-with icmp-ho
st-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

รูปที่ ก.16 iptable หลังแก้ไข firewall

8) เพิ่ม port 50001 ใน Security Groups ใน EC2 Dashboard ให้สามารถติดต่อกับ VM ได้



รูปที่ ก.17 การเพิ่ม port 50001 ใน Security Groups ใน EC2 Dashboard

9) เข้าไปใน DB2 Command Line Processor แล้วเพิ่มคำสั่งให้ port ที่เปิดไปใช้กับ DB2 ได้ update database manager configuration using svcename db2c_ec2-user

10) คล้ายกับที่ลงใน PC ปรกติ โดย สามารถเอา code ที่ compile แล้ว (.class) ไปวางไว้ที่ /home/ec2-user/sqllib/function แล้วใช้คำสั่ง SQL สร้าง UDF และใช้งานได้

ก4 การติดตั้งมองโกดีบี (MongoDB) ลงเซิฟเวอร์จำลอง

มีขั้นตอนดังนี้

1) เชื่อมต่อไปยังเซิฟเวอร์จำลองที่สร้างขึ้นมาโดยคลิกขวาที่เซิฟเวอร์ แล้วเลือก Connect โดยผู้ใช้สามารถเชื่อมต่อผ่านโปรแกรม SSH Client ของผู้ใช้ หรือใช้ Java SSH Client ของ Amazon ในกรณีนี้จะใช้ Java SSH Client โดยใส่ที่อยู่ของ Private key และ User name ที่จะเข้าไปยังเซิฟเวอร์

รูปที่ ก.18 การติดต่อไปยังเซิฟเวอร์จำลองผ่าน SSH Client ของ Amazon

2) ตั้งค่าระบบ Package Management System (APT) แล้วติดตั้งมองโกดีบี โดยใช้คำสั่งดังต่อไปนี้

โปรแกรมที่ ก.1 คำสั่งการตั้งค่าระบบ Package Management System

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart
dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
```

```
sudo apt-get update
sudo apt-get install mongodb-10gen
```

3) จากนั้นเชื่อมต่อไปยัง Mongo Shell เมื่อใช้การในการติดต่อกับมองโกดีบี โดยใช้คำสั่ง mongo จะได้ผลลัพธ์ดังรูป

```
ubuntu@ip-172-31-4-246:~$ mongo
MongoDB shell version: 2.4.6
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
>
```

รูปที่ ก.19 การเข้าใช้งาน Mongo Shell

ภาคผนวก ข

การใช้งาน

ข1 การใช้งาน AWS Elastic Beanstalk

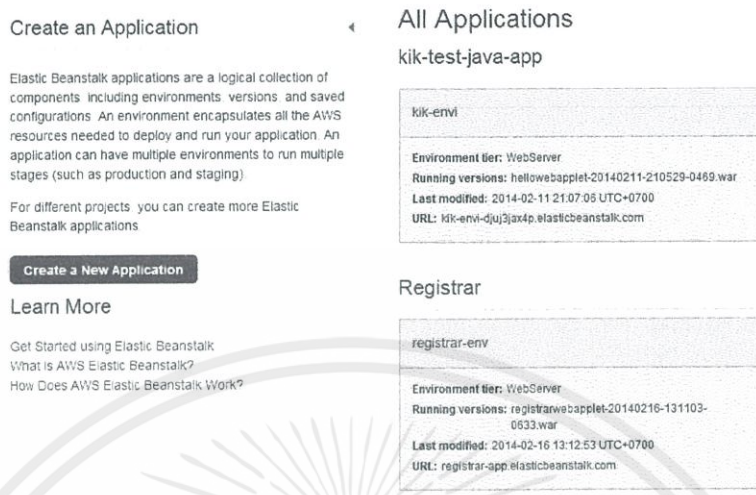
การสร้าง application เพื่อวิ่งบน AWS Elastic Beanstalk มีขั้นตอนดังนี้

1) เลือกใช้บริการ Elastic Beanstalk จากหน้า dashboard ของ AWS

รูปที่ ข.1 หน้า dashboard ของ AWS

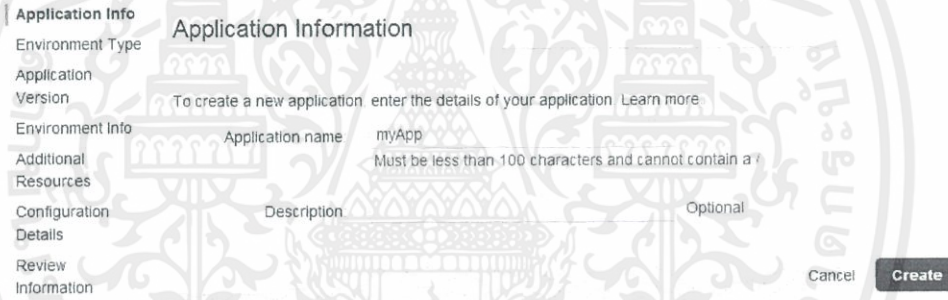
2) เลือก create a new application จากหน้าจอแสดง application ซึ่งมีการแสดงรายละเอียดดังนี้

- ชื่อ
- ประเภทของ environment
- version ปัจจุบัน
- วัน เวลาที่แก้ไขครั้งล่าสุด



รูปที่ ข.2 หน้าจอแสดง application ของ Elastic Beanstalk

3) ใส่ชื่อ app และคำอธิบาย แล้วเลือก create



รูปที่ ข.3 การสร้าง application ใหม่ใน Elastic Beanstalk

4) เลือก environment type ที่ต้องการ

environment tier มีตัวเลือกคือ

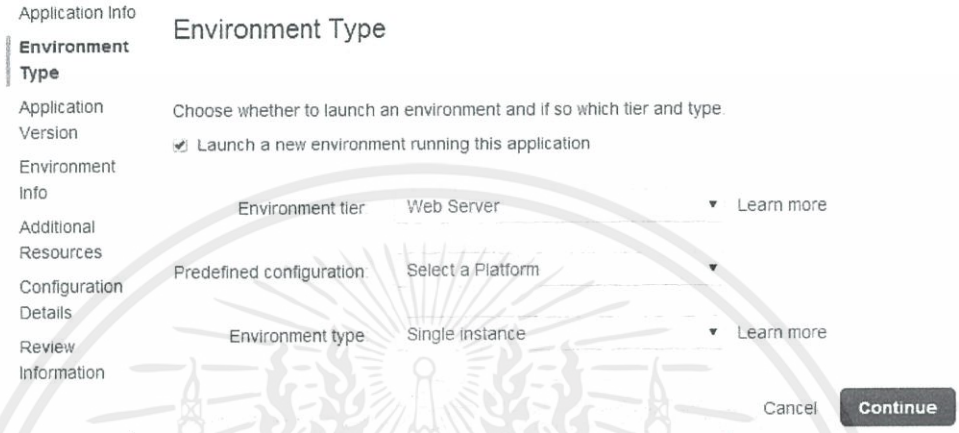
- Web server: รองรับ HTTP(S) requests
- Worker: รองรับ background job ของ app

predefined configuration มีตัวเลือกคือเลือก platform ที่ใช้กับ environment ที่สร้างขึ้น

- web server environment tier รองรับ application ที่พัฒนาใน Java, PHP, .NET, Node.js, Python และ Ruby
- worker environment tier รองรับ Node.js, PHP, Python, Ruby และ Java

environment type มีตัวเลือกคือ

- Single instance
- Load balancing, autoscaling: สามารถกระจาย load ที่เข้ามาไปยัง instance ต่างๆ ได้ และเพิ่ม/ลด instance ให้เหมาะสมกับปริมาณ load ในปัจจุบันด้วย

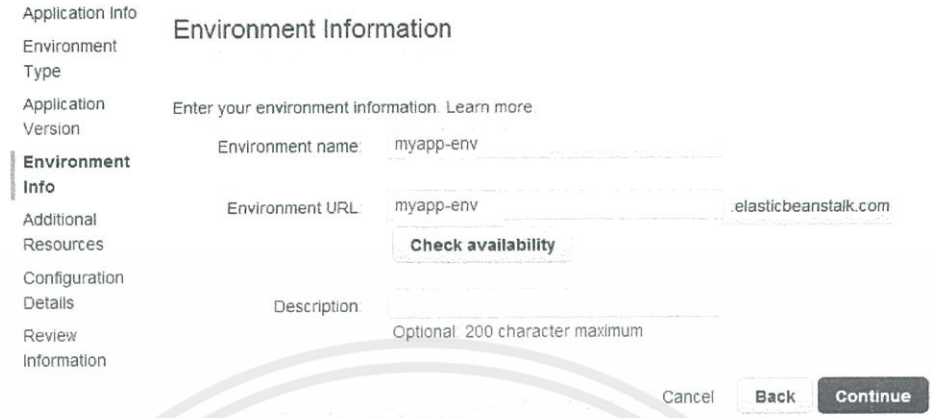


รูปที่ ข.4 การสร้าง environment ให้กับ application ที่สร้างใหม่

5) เลือกไฟล์ของ applicaton ที่จะขึ้นไปวิ่งบน cloud หรือจะเลือก sample application ก็ได้

รูปที่ ข.5 การเลือก application ที่จะทำการ deploy

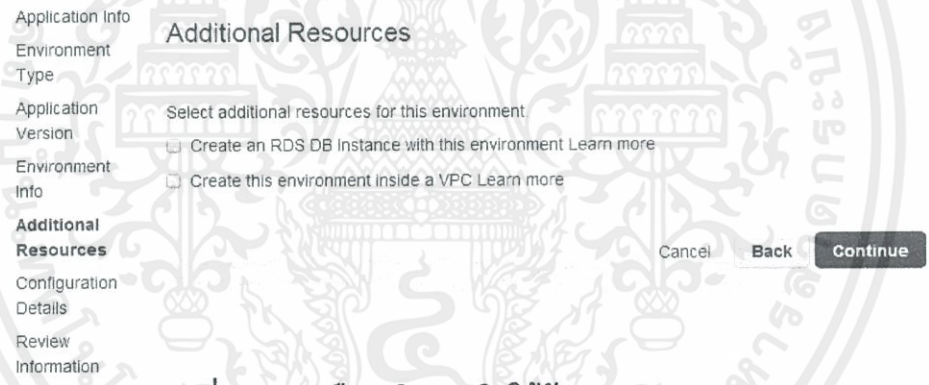
6) ตั้งชื่อ environment และ URL



รูปที่ ข.6 การตั้งชื่อ URL ให้กับ application

7) เลือกบริการเสริม

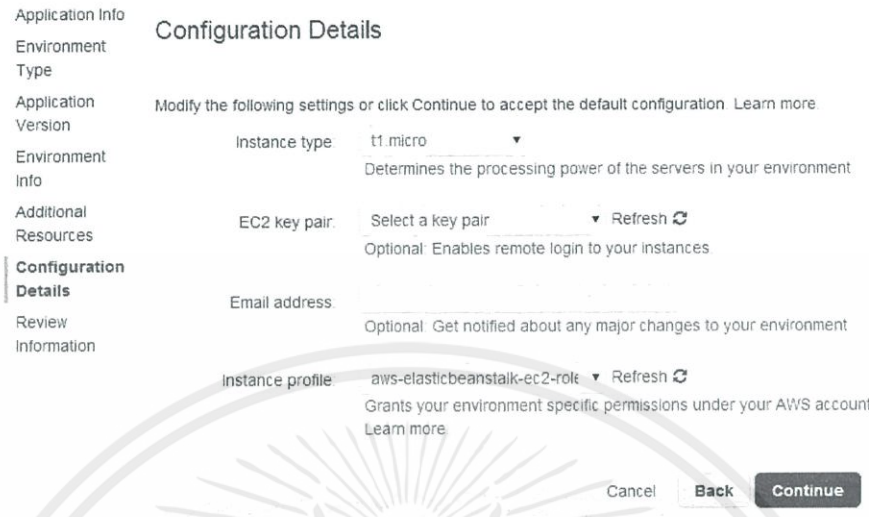
- Amazon Relational Database Service (Amazon RDS)
- Amazon Virtual Private Cloud (Amazon VPC)



รูปที่ ข.7 การเลือกบริการเสริมให้กับ application

8) เลือกรายละเอียด configuration ให้กับ application

- เลือก instance type
- EC2 key pair สำหรับ ติดต่อกับ instance
- Email ที่แจ้งเตือนเมื่อ environment เปลี่ยนแปลง
- Instance profile



รูปที่ ข.8 การเลือกรายละเอียด configuration ให้กับ application

9) จะได้หน้าแสดงแสดงผลการสร้าง application

รูปที่ ข.9 หน้าจอแสดงรายละเอียด application ที่สร้างขึ้น

นอกจากนี้ AWS Elastic Beanstalk ยังมีความสามารถในการจัดการกับ application ดังนี้

- Moniter

  สามารถตรวจสอบการใช้งาน network, CPU, latency, sum request ได้

รูปที่ ข.10 หน้าจอ monitor แสดงผลการใช้งานทรัพยากรของ application

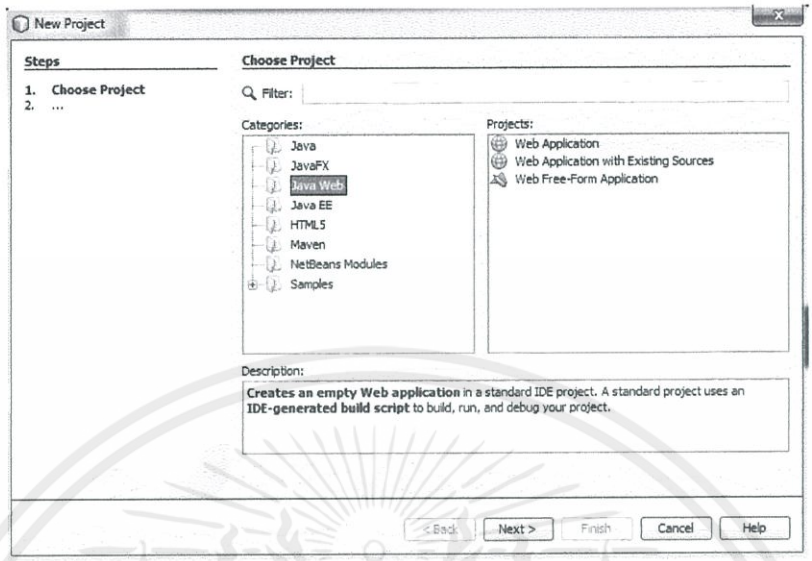
- การเก็บประวัติ version ของ application

รูปที่ ข.11 หน้าจอเก็บประวัติ version ของ application

ข2 การใช้งาน NetBeans IDE เพื่อติดตั้งโปรแกรมประยุกต์บน AWS Elastic Beanstalk

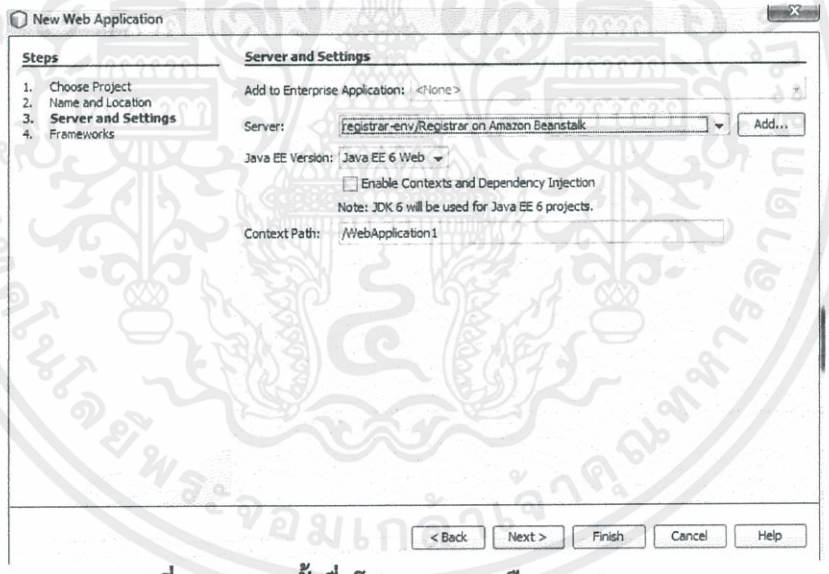
NetBeans IDE เป็น software ที่ใช้ในการพัฒนาโปรแกรมภาษา Java มีความสามารถในการเชื่อมต่อกับ AWS Elastic Beanstalk เพื่อ deploy application ที่สร้างขึ้นมาได้ โดยมีวิธีการดังนี้

1) สร้างโปรเจคใหม่ ใน NetBeans IDE เลือกประเภท Web Application



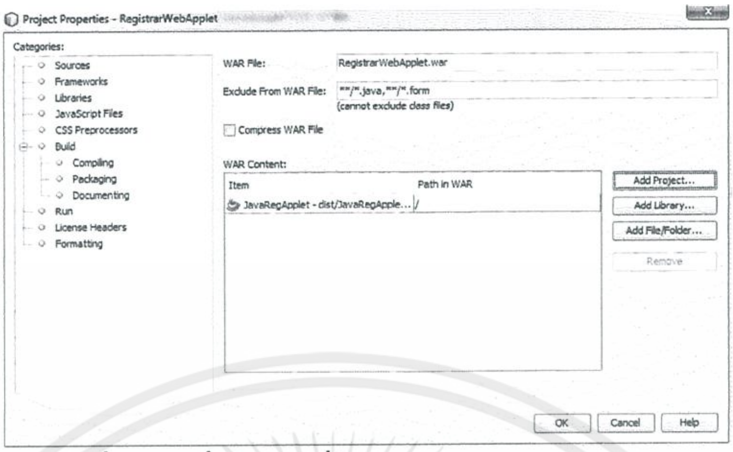
รูปที่ ข.12 การสร้างโปรเจค Web Application ใน NetBeans IDE

2) ตั้งชื่อโปรเจค, เลือก server เป็น web server ที่อยู่บน AWS Elastic Beanstalk



รูปที่ ข.13 การตั้งชื่อโปรเจค และเลือก web server

3) คลิกขวาที่โปรเจค เลือก properties จะแสดงหน้าต่าง Project Properties จากนั้นเลือก Packaging แล้วเพิ่มโปรเจคที่ web application จะใช้



รูปที่ ข.14 เพิ่มโปรเจคที่ต้องการบน web application

4) นำ library ที่แอพเพล็ต ใช้ ไปไว้ในโฟลเดอร์ของโปรเจค แก้ไข ไฟล์ index.jsp ในโปรเจค ให้แอพเพล็ต ทำงานได้

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registrar Application Demo</title>
    </head>
    <body>
        <h3>Registrar Application</h3>
        <applet
            codebase="http://registrar-app.elasticbeanstalk.com/"
            archive="JavaRegApplet.jar, lib/db2jcc.jar, lib/gson-2.2.4.jar, lib/itextpdf-5.4.5.jar, lib/mongo-java-driver-2.11.2.jar"
            code="my.app.RegApplet"
            width=1200 height=800>
        </applet>
    </body>
</html>
```

รูปที่ ข.15 แก้ไข code ในไฟล์ index.jsp ให้แอพเพล็ต สามารถทำงานได้

5) สั่ง run project ที่สร้างขึ้น เพื่อ deploy web application ขึ้น web server ที่เลือกไว้บน AWS Elastic Beanstalk สามารถใช้งาน app ได้จาก web browser ด้วย URL ที่สร้างไว้ตอนที่สร้าง app บน AWS Elastic Beanstalk

นอกจากนี้ปรกติการใช้งานแอพเพล็ต ผ่าน web จะมีข้อจำกัดอยู่เพื่อความปลอดภัยของผู้ใช้แอพเพล็ต เช่นการป้องกันไม่ให้อ่าน/เขียนไฟล์ เป็นต้น แต่หาก application มีการใช้งานเหล่านี้ ผู้ใช้ต้องไปให้สิทธิต่างๆในไฟล์ java.policy ซึ่งอยู่ในโฟลเดอร์ C:\Program Files\Java\jre7\lib\security แล้วเพิ่มการให้สิทธิกับ URL ที่ต้องการ

```
grant codeBase "http://registrar-app.elasticbeanstalk.com/lib/*" {
        permission java.security.AllPermission;
};

grant codeBase "http://registrar-app.elasticbeanstalk.com/*" {
        permission java.security.AllPermission;
};
```

รูปที่ ข.16 เพิ่มสิทธิให้กับการวิ่ง application ใน java.policy