การประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอล

TRANSACTION PROCESSING ON CLOUD SQL DATABASE SYSTEM

กัญญพร ฐิตะโสหะกุล
รัษฐ์ รัษฐปานะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

การประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอล

TRANSACTION PROCESSING ON CLOUD SQL DATABASE SYSTEM

กัญญพร ฐิตะโลหะกุล
รัษฐ์ รัษฐปานะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

ปริญญานิพนธ์ปีการศึกษา 2555

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอล

TRANSACTION PROCESSING ON CLOUD SQL DATABASE SYSTEM

ผู้จัดทำ

1. นางสาว กัญญพร ฐิตะโลหะกุล รหัสนักศึกษา 52010043
2. นาย รัษฐ์ รัษฐปานะ รหัสนักศึกษา 52010998

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. ศุภมิตร จิตตะยโศธร)

การประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอล

นางสาว กัญญพร	ฐิตะโลหะกุล	52010043
นาย รัษฐ์	รัษฐปานะ	52010998
รศ.ดร. ศุภมิตร	จิตตะยโศธร	อาจารย์ที่ปรึกษา

ปีการศึกษา 2555

บทคัดย่อ

ระบบคอมพิวเตอร์ชนิดคลาวด์เป็นเทคโนโลยีที่น่าสนใจและมีอนาคต มีระบบจัดการฐานข้อมูลจำนวนหนึ่ง ปฏิบัติงานบนระบบคอมพิวเตอร์ชนิดคลาวด์ได้แล้ว โครงงานนี้เป็นการศึกษาขีดความสามารถ ในการประมวลผลทรานแซกชั่นของระบบจัดการฐานข้อมูล Google Cloud SQL บนระบบคลาวด์ โดยทำงานศึกษาเกี่ยวกับการเลื่อนการบังคับกฎบังคับความถูกต้องของฐานข้อมูล สถานะของทรานแซกชั่น การจัดลำดับการปฏิบัติคำสั่งในทรานแซกชั่น การปฏิบัติงานร่วมกันในช่วงเวลาเดียวกัน การปรับระดับของความถูกต้องตามความต้องการของโปรแกรมประยุกต์ การปฏิบัติหลายงานเป็นกิจกรรมเดียว การกู้คืนทรานแซกชั่นจากกรณีความล้มเหลวของระบบ Google Cloud SQLจะถูกประเมินในหัวข้อต่างๆเหล่านี้ พร้อมทั้งมีการพัฒนาโปรแกรมประยุกต์ต้นแบบ บน Google Cloud SQL ในระบบคลาวด์

TRANSACTION PROCESSING ON CLOUD SQL DATABASE SYSTEM

Ms. Kanyaporn	Thitalohakul	52010043
Mr. Rast	Rastapana	52010998
Assoc.Prof.Dr.Suphamit	Chittayasothorn	Advisor

Academic Year 2012

ABSTRACT

Cloud computing is a promising technology. There are database management systems that operate in the cloud computing environment. This project studies transaction processing capabilities of the Google Cloud SQL system in the cloud. Students have to first understand the principles of transaction processing including deferred integrity constraints enforcement, transaction states, transaction scheduling and concurrency control, levels of consistency, transaction atomicity and recovery. The Google Cloud SQL DBMS in the cloud will then be evaluated based on these aspects. Prototype applications that work on Google Cloud SQL in the cloud will be constructed.

กิตติกรรมประกาศ

โครงงาน การประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอลสามารถสำเร็จสมบูรณ์และลุล่วงได้ด้วยดีจากการให้คำปรึกษาของ รศ.ดร. ศุภมิตร จิตตะยโศธร ซึ่งคอยให้คำแนะนำ และมีการติดตามความคืบหน้าของโครงงานอยู่อย่างสม่ำเสมอ

ขอขอบพระคุณอาจารย์ทุกท่านที่อยู่ในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ประสิทธิ์ประสาทวิชาความรู้ให้ นำมาซึ่งผลลัพธ์ที่ช่วยให้โครงงานดำเนินก้าวหน้าไปได้ด้วยดี

ขอขอบคุณรุ่นพี่สำหรับการช่วยเหลือ และคำแนะนำในการปฏิบัติตัวให้เป็นนักศึกษาชั้นปีที่สี่ที่มีรับผิดชอบกับการทำโครงงาน รวมไปถึงขอบคุณสำหรับปริญญานิพนธ์ที่รุ่นพี่ทิ้งไว้ให้ได้ศึกษาแนวทางในการเขียนและรูปแบบการเขียน

ขอขอบคุณบิดามารดาที่ให้การเลี้ยงดูและให้ความสนับสนุนในทุกด้านเพื่อให้การทำงานเป็นไปได้ด้วยดีและทำให้มีกำลังใจในการทำงานอยู่เสมอ

ขอขอบคุณเพื่อนที่ช่วยให้คำปรึกษาและแก้ปัญหาที่เกิดขึ้นในระหว่างการทำโครงงานรวมทั้งการให้ความช่วยเหลือในเรื่องอื่น ๆ ที่เป็นประโยชน์ต่อการเรียนและใช้ชีวิตในฐานะนักศึกษาคณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ขอขอบคุณห้องวิจัย Hardware Laboratory ที่ให้สถานที่ในการทำโครงงาน จนกระทั่งสามารถสำเร็จลุล่วงไปได้ด้วยดี

นางสาว กัญญพร ฐิตะโลหะกุล

นาย รัษฐ์ รัษฐปานะ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

เนื่องจาก Cloud computing เป็นแนวคิดที่ปัจจุบันได้รับความนิยมในหลายองค์กร และ มีแนวโน้มว่าจะได้รับความนิยมมากขึ้นต่อไปในอนาคต ทำให้นักศึกษาเกิดความคิดที่จะนำเอา Cloud computing มาประยุกต์ใช้กับระบบฐานข้อมูล (Database system) จึงมีแนวคิดที่จะพัฒนา ระบบ Cloud ไปประยุกต์ใช้ โดย ระบบฐานข้อมูล มีพื้นฐานในการดำเนินกับระบบฐานข้อมูล จะต้องมีคุณสมบัติสำคัญหลักๆคือ Atomicity อีกทั้งยังมี เรื่องของ Consistency ซึ่งเป็นเรื่อง ของการรักษาความถูกต้องของข้อมูล นอกจากนี้ยังมีเรื่องของ Isolation ซึ่งสามารถรองรับการทำงาน แบบ multi User ได้โดยไม่รบกวนกัน และมีเรื่องของ Durability และ การQuery processing การทำโครงงานครั้งนี้ จึงต้องเริ่มจากการศึกษา พื้นฐาน Transaction processing บน Database system รวมไปถึงการพัฒนาแอพพลิเคชั่น ว่าแอพพลิเคชั่นที่พัฒนาขึ้นนั้นรองรับคุณสมบัติพื้นฐานของทรานแซกชั่น

1.2 วัตถุประสงค์ของโครงงาน

1) การใช้ Cloud database ช่วยลดค่าใช้จ่าย ในการพัฒนาระบบ เนื่องจาก สามารถประหยัด Resource ได้ โดยไม่ต้องเสียเวลาในการบำรุงรักษาเอง และ ใช้จ่ายตามปริมาณการใช้งาน
2) สามารถใช้ Resource ได้อย่างคุ้มค่า เนื่องจากการ ทำ Cloud คือการสร้างการจำลองอุปกรณ์ หรือทรัพยากร ต่างๆ ซึ่งสามารถเพิ่ม ลด ได้ โดยไม่ต้องคำนึงถึงว่า ทรัพยากรนั้นอยู่ที่ใด (Resource on demand)
3) เพื่อสามารถนำApplication ไปใช้งานได้จริง บนระบบฐานข้อมูล
4) เพื่อเพิ่มประสิทธิภาพการใช้งานเทคโนโลยีใหม่ๆ ต่อไปในอนาคต
5) เพื่อตรวจสอบประสิทธิภาพ การทำ Transaction processing ว่าใน Cloud database มีคุณสมบัติเหมือนการทำ ในระบบ Database system หรือไม่

1.3 ขอบเขตของโครงงาน

การพัฒนาโครงงานการประมวลผลทรานแซกชั่นบนระบบฐานข้อมูลแบบคลาวด์เอสคิวแอล จะเป็นการศึกษาเรื่องของทรานแซกชั่น เพื่อเปรียบเทียบความเหมือนและความแตกต่าง ถึงขีดจำกัดของระบบฐานข้อมูลบนคลาวด์ เทียบกับระบบฐานข้อมูลทั่วไป รวมไปถึงการพัฒนาเว็บแอพพลิเคชั่นโปรแกรม เพื่อทดลองการใช้งานของทรานแซกชั่นให้ชัดเจนยิ่งขึ้น

1.4 วิธีการดำเนินการ

วางแผนการทำงานช่วงระยะเวลาเทอมหนึ่งเน้นศึกษาทรานแซกชั่นและวิเคราะห์ระบบงาน

ตารางที่ 1.1 วิธีการดำเนินการในภาคเรียนที่ 1

แผนการทำงานแต่ละสัปดาห์	มีนาคม				เมษายน				พฤษภาคม				มิถุนายน				กรกฎาคม				สิงหาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1) เสนอหัวข้อโครงงาน																								
2) ศึกษาเกี่ยวกับคุณสมบัติการทำ Transaction Processing																								
3) วิเคราะห์ระบบงาน																								
3.1 Project Planning																								
3.1.1 Requirement Specification																								
3.2 Project Analysis																								
3.3 Project Design																								
3.4 Project Implementation																								
3.4.1 Project Development																								
3.4.2 Project Verification																								
3.5 Project Analysis																								

ตารางที่ 1.2 วิธีการดำเนินการในภาคเรียนที่ 2

แผนการทำงานแต่ละสัปดาห์	ตุลาคม				พฤศจิกายน				ธันวาคม				มกราคม				กุมภาพันธ์				มีนาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1) ศึกษารูปแบบการพัฒนาแอพพลิเคชั่น(Model)																								
2) ศึกษาภาษาที่ใช้ในการพัฒนาเว็บแอพพลิเคชั่นบนคลาวด์																								
3) วางแผนและพัฒนาเว็บแอพพลิเคชั่นที่หนึ่ง																								
3.1 ออกแบบ ER Diagram แอพพลิเคชั่นที่หนึ่ง																								
3.2 พัฒนาเว็บแอพพลิเคชั่นด้วยจาวาและทรานแซกชั่น																								
4)วางแผนและพัฒนาเว็บแอพพลิเคชั่นที่สอง																								
4.1 ออกแบบ ER Diagram แอพพลิเคชั่นที่สอง																								
4.2 พัฒนาเว็บแอพพลิเคชั่นที่สอง																								
5)ตรวจสอบความสมบูรณ์ของชิ้นงาน																								
6)จัดทำรูปเล่มรายงาน																								

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้เรียนรู้การทำงานของระบบคลาวด์ที่ได้รับความนิยมในปัจจุบัน
2) ได้รับความรู้และความเข้าใจเกี่ยวกับ ระบบจัดการฐานข้อมูลชัดเจนมากยิ่งขึ้น
3) ได้เรียนรู้ขั้นตอนในการพัฒนาโปรแกรมด้วยภาษาต่าง ๆ
4) ได้สร้างเว็บแอพพลิเคชั่นที่สามารถทำงานอย่างมีระบบ และ ปลอดภัย

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐานของระบบคลาวด์ ทฤษฎีที่เกี่ยวข้องกับระบบจัดการฐานข้อมูล ซึ่งได้แก่ทรานแซกชั่น การรักษาความปลอดภัยในระบบคลาวด์ การกู้คืนทรานแซกชั่นในกรณีที่ระบบเกิดความล้มเหลว การทำงานแบบขนาน และทฤษฎีอื่นๆ

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงการออกแบบโครงสร้างของแอพพลิเคชั่น การออกแบบการทดลองต่าง ๆ โดยแบ่งเป็นการออกแบบการทดลอง และ การออกแบบการพัฒนาเว็บแอพพลิเคชั่น

บทที่ 4 การทดลองและผลการทดลอง จะแบ่งออกเป็นสองส่วนใหญ่ คือ การทดลองทั่วไปเพื่อทดสอบทรานแซกชั่นบนระบบคลาวด์ อาทิเช่น การอิมพอร์ตข้อมูลลงคลาวด์ การอัพโหลดแอพพลิเคชั่น การให้สิทธิแก่ผู้ใช้ การทดสอบปัญหาที่เกี่ยวข้องของทรานแซกชั่น การทดลองเกี่ยวกับการตั้งค่า SAVEPOINT และ ROLLBACK TO SAVEPOINT และ เครื่องมือที่ใช้สถานะการทำงาน อีกส่วนหนึ่ง คือ การพัฒนา Web application ซึ่งเป็นการพัฒนาทำเว็บแอพพลิเคชั่นต้นแบบ

บทที่ 5 บทสรุป กล่าวถึงสรุปผลที่ได้จากการทดลอง รวมถึงปัญหาอุปสรรคต่างๆ ในระหว่างการทำโครงงาน และทำการสรุปสิ่งที่ได้ทำตลอดโครงงาน และข้อเสนอแนวทางในการพัฒนาต่อ

ภาคผนวก ก แสดงการติดตั้ง Google cloud SQL, Google App Engine และ Google cloud storage พร้อมการใช้งานเบื้องต้น

ภาคผนวก ข เป็นการแสดงรายละเอียดการพัฒนา Web application และ เครื่องมือที่ใช้ในการพัฒนา

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 การประมวลผลแบบกลุ่มเมฆ (Cloud computing)

Cloud computing คือ รูปแบบการใช้งานทรัพยากรต่างๆ ซึ่งได้แก่ ฮาร์ดแวร์ และซอฟต์แวร์ โดยรูปแบบการบริการต่างๆเหล่านี้ ผู้ใช้งานจะเรียกใช้ผ่านอินเตอร์เน็ตในคอมพิวเตอร์หรือ อุปกรณ์ที่มีสัญญาณเครือข่ายเชื่อมต่อถึง โดยผู้ใช้สามารถเลือกใช้ได้จากผู้ให้บริการ ซึ่งจะแบ่งสรรทรัพยากรให้กับผู้ใช้งานนั้นๆ โดย เรียกการใช้งานผ่านอินเตอร์เน็ตอีกอย่างหนึ่งว่า คลาวด์ และเรียกการประมวลผลของคลาว์ว่า Cloud computing ซึ่งการทำงานในลักษณะดังกล่าว ผู้ใช้งานไม่จำเป็นต้องมีความรู้เกี่ยวกับ โครงสร้างพื้นฐานของเครือข่ายในการเชื่อมต่อ หรือ อุปกรณ์ที่ใช้ในการติดตั้ง รวมไปถึงไม่จำเป็นต้องรู้ตำแหน่งของเครื่องปลายทางที่แท้จริงในการสื่อสาร ซึ่งหมายถึงการทำงานในลักษณะดังกล่าวของการประมวลผลแบบกลุ่มเมฆ ผู้ให้บริการจะดำเนินการให้บริการจากระยะไกลซึ่งให้บริการทั้งส่วนของข้อมูลของผู้ใช้ ซอฟต์แวร์ของผู้ใช้ รวมไปถึงระบบการประมวลผลต่างๆ โดยในการใช้ระบบคลาวด์ดังกล่าว จะช่วยให้ผู้ใช้สามารถลดการจัดการงานด้านสารสนเทศได้อย่างมีประสิทธิภาพมากขึ้น

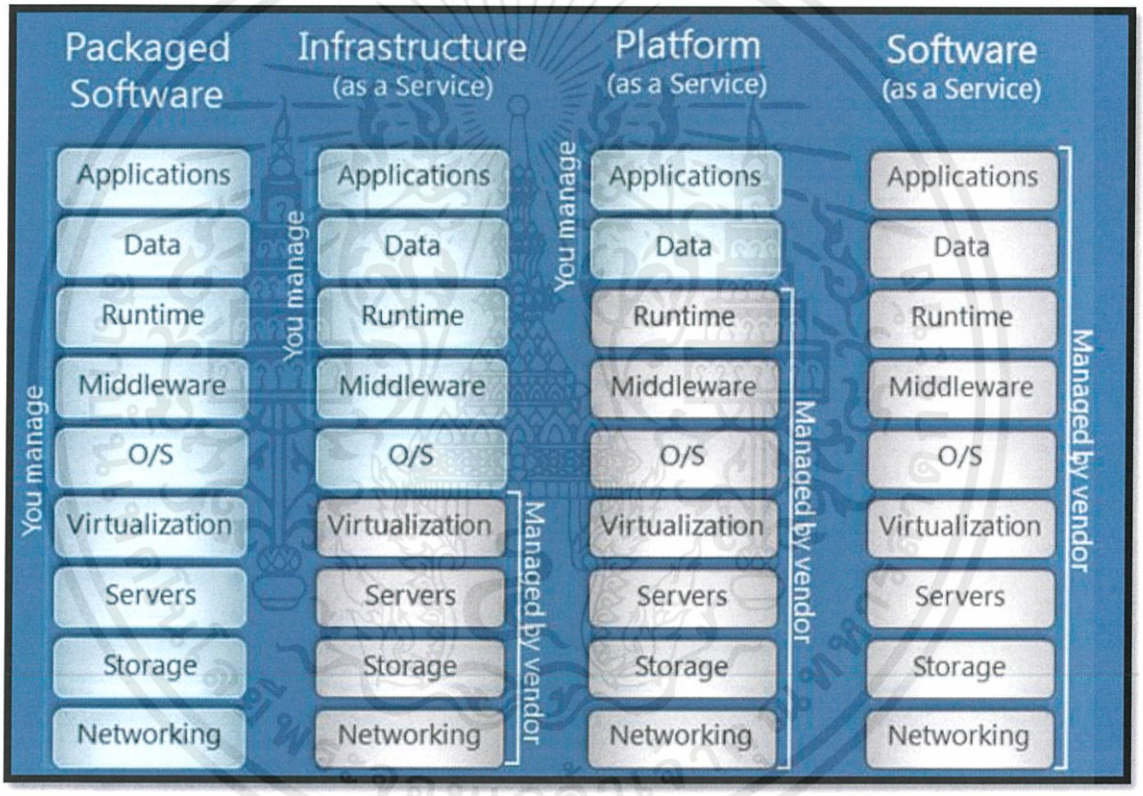
การบริการบนระบบการประมวลผลแบบกลุ่มเมฆ สามารถแบ่งรูปแบบได้หลายระดับขึ้นอยู่กับความต้องการในการเลือกใช้งาน ซึ่งสามารถแบ่งได้ดังนี้

2.1.1 Infrastructure as a service (IaaS) เป็นระบบที่เน้นการให้บริการเฉพาะเกี่ยวกับโครงสร้างพื้นฐาน ซึ่งเกี่ยวข้องกับ ฮาร์ดแวร์ รวมไปถึงการให้บริการที่เก็บข้อมูลต่างๆ เช่น Virtual servers , Disk storage , Networking และ Security ต่าง ๆ ในรูปแบบ เวอร์ชวลไลเซชั่น ซึ่งทำให้เราสามารถจัดสรรทรัพยากรได้แบบไดนามิก เช่น การเพิ่มหรือลดขนาดของซีพียู หรือ แรมของเครื่องเซิร์ฟเวอร์ เป็นต้น โดยในที่นี้ผลิตภัณฑ์ของกูเกิลที่ใช้ในการทำโครงงานครั้งนี้ ได้แก่ Google Cloud storage ซึ่งต้องสอดคล้องกับ ระบบปฏิบัติการของระบบฐานข้อมูล (Database system)

2.1.2 Platform as a service (PaaS) เป็นระบบที่เน้นการให้บริการเกี่ยวกับแพลตฟอร์มที่รองรับการสร้างและจัดการแอพพลิเคชั่น โดยผู้ใช้บริการสามารถปรับใช้ และจัดการได้เอง ซึ่งการใช้บริการที่ระดับนี้ จะไม่ต้องคำนึงถึง ฮาร์ดแวร์ที่อยู่ข้างใต้ แต่พิจารณาที่การควบคุมแอพพลิเคชั่นให้ตรงกับความต้องการของผู้พัฒนาเป็นสำคัญ ซึ่งในผลิตภัณฑ์ของกูเกิลที่ใช้ในโครงงาน

ที่เป็น PaaS ได้แก่ Google App Engine ซึ่งเป็น Web service ที่สามารถพัฒนาได้จาก ภาษา Java , Python และ Go นอกจากนี้ Google Cloud SQL ซึ่งเป็นฐานข้อมูลที่เราใช้ในการวิเคราะห์ Cloud database ก็ถูกจัดเป็น PaaS เช่นกัน

2.1.3 Software as a service (SaaS) เป็นระบบที่เน้นการให้บริการซอฟต์แวร์ซึ่งจะให้บริการการประมวลผลแอพพลิเคชั่นที่แม่ข่ายของผู้ให้บริการและเปิดให้บริการทางด้านซอฟต์แวร์ต่างๆ ซึ่งถูกยกตัวอย่างที่เห็นได้ง่ายที่สุดคือ อีเมล เช่น Gmail หรือ Google docs เป็นต้น



รูปที่ 2.1 รูปแบบต่าง ๆ ของ Cloud computing

2.2 ส่วนประกอบอื่นในการประมวลผลแบบกลุ่มเมฆ

Google cloud platform เป็นรูปแบบของคลาวด์ที่สร้างขึ้นมาเพื่อให้ผู้ใช้งานเน้นการสร้างแอพพลิเคชั่น รวมถึงการเก็บและวิเคราะห์ข้อมูลบน Google's infrastructure โดยเน้นจุดเด่นไปที่เรื่องของ ความเร็วและการเพิ่มขนาดของแอพพลิเคชั่นได้อย่างไม่จำกัด เน้นการสร้างและการจัดการ รวมถึงค่าใช้จ่ายที่ผู้ใช้งานสามารถเลือกให้เหมาะสมกับงานของตนเองได้

รูปที่ 2.2 คุณสมบัติเด่นของ Google App engine ที่ใช้สร้างแอพพลิเคชั่น

ส่วนประกอบอื่นๆที่ Google cloud environment สร้างขึ้นเพื่อให้การใช้งานบนคลาวด์นั้นมีการให้บริการที่หลากหลาย โดยออกแบบหลากหลายผลิตภัณฑ์หรือหลายเซอร์วิส ซึ่งสามารถร่วมกันทำงานได้และประกอบรวมไปเป็นแอพพลิเคชั่นบนเว็บ (Web application) ซึ่งเซอร์วิสดังกล่าวประกอบไปด้วย เว็บเซอร์วิส โปรแกรม หรือ สคริปเฉพาะ ทั้งนี้เพื่อให้ผู้พัฒนาแอพพลิเคชั่นบนอินเตอร์เน็ตสามารถเพิ่มรูปแบบในการสร้างแอพพลิเคชั่นได้ตามความต้องการของผู้พัฒนามากขึ้น แทนการที่จะต้องสร้างแอพพลิเคชั่นที่ซับซ้อนบางโปรแกรมด้วยตัวเอง จึงมีการบริการดังกล่าวเรียกว่า Google APIs โดยในปัจจุบันนั้นเปิดให้ผู้พัฒนาเว็บแอพพลิเคชั่นทั้งหลาย ใช้งานได้แล้ว แม้ยังมีบางเซอร์วิสที่เปิดให้บริการยังเป็นเวอร์ชั่นทดลอง ซึ่งมีดังต่อไปนี้

- Ad Exchange Buyer API
- AdSense Host API
- AdSense Management API
- Analytics API
- Audit API
- BigQuery API
- Blogger API v3
- Books API
- Calendar API
- Custom Search API
- DFA Reporting API
- Drive API
- Drive SDK

- Enterprise License Manager API
- Freebase API
- Fusion Tables API
- Google Affiliate Network API
- Google Cloud Messaging for Android
- Google Cloud SQL
- Google Cloud Storage
- Google Cloud Storage JSON API
- Google Compute Engine
- Google Maps API v2
- Google Maps API v3
- Google Maps Coordinate API
- Google Play Android Developer API
- Google+ API
- Google+ Hangouts API
- Identity Toolkit API
- Latitude API
- Moderator API
- Orkut REST API
- Page Speed Online API
- PageSpeed Service
- Places API
- Prediction API
- Search API for Shopping
- Site Verification API
- Static Maps API
- Street View Image API
- Tasks API

- Translate API
- URL Shortener API
- Web Fonts Developer API
- YouTube Analytics API
- YouTube Data API

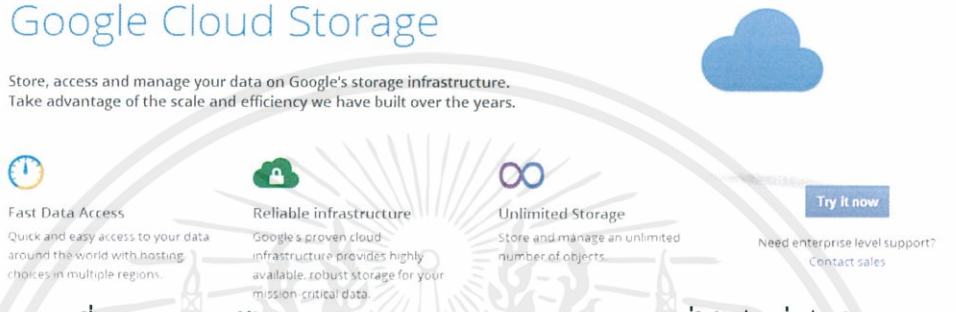
จากบริการต่างๆที่ระบุไว้ด้านบน การพัฒนาแอพพลิเคชั่นที่จำเป็นในการทดลองโครงงานนี้ประกอบไปด้วยสองเซอร์วิสด้วยกันได้แก่ Google Cloud Storage และ Google Cloud SQL และอีกหนึ่งบริการซึ่งเรียกว่า Google App engine ซึ่งเป็นตัวจัดการแอพลิเคชั่นให้สามารถอยู่ในอินเตอร์เน็ตและใช้เป็น แพลตฟอร์มการพัฒนาเว็บแอพพลิเคชั่นต่อไป ซึ่งสามารถเจาะลึกรายละเอียดของการให้บริการคลาวด์ออกเป็น 3 อย่างดังต่อไปนี้

2.2.1 Google Cloud Storage เป็นบริการสำหรับที่จัดเก็บและการเข้าถึงข้อมูลบนคลาวด์ ซึ่งบริการนี้รวมไปถึงความสามารถในการแสดงผล (Performance) และความสามารถในการรองรับการ ขยายขนาดของที่เก็บข้อมูล (Scalable storage) ในคลาวด์ของกูเกิลด้วยความปลอดภัยและมีประสิทธิภาพในการใช้งานคลาวด์ในระบบร่วมกัน จุดเด่นของการบริการ Google Cloud Storage ที่ระบุไว้เพิ่มเติม คือ ข้อมูลทุกข้อมูลจะถูกเก็บไว้ซ้ำกันมากกว่าหนึ่งที่ ซึ่งหมายถึงข้อมูลจะถูกกระจายไปยังหลายศูนย์ที่เก็บข้อมูล (Data center) เพื่อป้องกันข้อมูลที่สูญหาย นอกจากนี้ยังรับประกันการอ่านและเขียนข้อมูลด้วยความถูกต้อง (Data consistency) และข้อมูลที่อัพโหลดหรือดาวน์โหลดสามารถมีขนาดใหญ่ได้ถึงระดับ เทราไบต์ ดังนั้น ผู้ใช้งานสามารถอัพโหลดไฟล์ประเภทต่างๆ ขึ้นไปเก็บไว้ในบริการประเภทนี้ได้ โดยในปัจจุบันยังสามารถใช้งานได้ฟรี โดยใช้งานได้ 5 กิกะไบต์ โดยในโครงงานนี้เราสนใจการอัพโหลดข้อมูลบางประเภทที่ไม่รองรับด้วยคำสั่งพื้นฐานของเอสคิวแอล ซึ่งได้แก่ ไฟล์นามสกุล .CSV เป็นต้น ซึ่งในการเริ่มขอเปิดการใช้งานเบื้องต้น ผู้ใช้งานจำเป็นต้องมีสิ่งต่อไปนี้

1) Google account ผู้ที่ต้องการขอเปิดการใช้งานจำเป็นต้องมี Google account ก่อน ซึ่งถ้าหากมีบัญชีอยู่เรียบร้อยแล้วก็ไม่จำเป็นที่จะต้องเปิดบัญชีใหม่

2) Google APIs console project โดยที่ Google APIs console เป็นหน้าเว็บเพจแบบกราฟิก ซึ่งใช้ในการจัดการบริการต่างๆของ Google APIs ที่ได้กล่าวไปแล้ว ภายในจะมีส่วนประกอบของ บริการต่างๆ ให้ผู้ใช้ทำการเลือกใช้

ทั้งนี้การนำเข้าไฟล์ข้อมูลในครั้งแรก อาจพบปัญหาการนำไฟล์เข้าสู่ Google Cloud SQL ที่ไม่สำเร็จ เนื่องจากในตอนแรก หากไม่สร้างฐานข้อมูลรอไว้ก่อนและไม่ทำการประกาศการ use database นั้นในนามสกุล .SQL จะทำให้ ตารางที่จะใช้ในการสร้างไม่รู้ว่าจะสร้างตารางดังกล่าวไว้ที่ฐานข้อมูลตัวใด ดังนั้นจึงต้องประกาศไว้ในไฟล์ก่อนอัพโหลดขึ้นไปยัง Google Cloud Storage

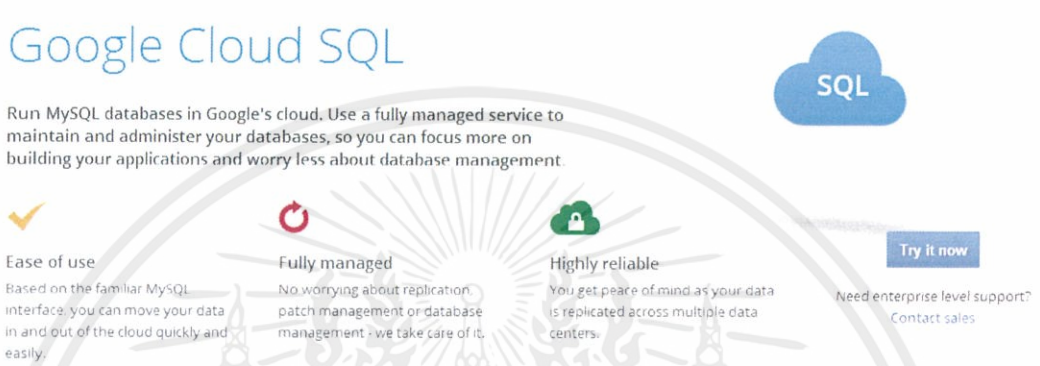


รูปที่ 2.3 คุณสมบัติเด่นของ Google Cloud Storage ที่ใช้เป็นที่เก็บข้อมูล

2.2.2 Google Cloud SQL เป็นระบบเว็บเซอร์วิสที่อนุญาตให้ผู้ใช้ สามารถสร้าง และตั้งค่าการใช้ Relational database ได้บนคลาวด์ โดยการบริการดังกล่าวมีการจัดการอย่างเต็มประสิทธิภาพ ทั้งในแง่ของการดูแล การจัดการฐานข้อมูล โดยความสามารถของบริการนี้ คล้ายคลึงกับต้ว MySQL database โดยที่บนคลาวด์นั้น อาจมีข้อแตกต่างในด้านคุณสมบัติบางประการที่เพิ่มขึ้นและลดลงในบางส่วน ในส่วนที่เพิ่มขึ้น คือการจัดการของ MySQL database ที่กระทำบนคลาวด์ นอกจากนี้ยังมีการเก็บข้อมูลซ้ำพร้อมกันในระดับภูมิศาสตร์ (Synchronous geographic replication) เพื่อป้องกันการสูญหายของข้อมูล หรือ ในกรณีของการดึงข้อมูลและการส่งออกฐานข้อมูลสามารถทำได้โดยใช้ mysqldump โดยที่ Google Cloud SQL รองรับการใช้ภาษา Java และ Python ซึ่งการเรียกใช้งานฐานข้อมูล สามารถทำงานได้ผ่านเว็บเซอร์วิสใน Google APIs Console ซึ่งสามารถเรียกใช้ผ่าน SQL Prompt หรือ สามารถโหลดเครื่องมือ Command line tool ซึ่งเป็นเครื่องมือที่สามารถเชื่อมต่อกับ Google Cloud SQL ในการแก้ไขได้โดยตรง หรือปรับปรุงฐานข้อมูลต่างๆได้ โดยมีลักษณะคล้ายกับ MySQL Command line เพียงแต่เครื่องมือนี้ต้องเชื่อมต่อเข้าไปยังระบบฐานข้อมูลก่อนทำการใช้งาน และ ส่วนที่เป็นข้อจำกัดของฐานข้อมูลระบบคลาวด์ ได้แก่ จำกัดขนาดของ instance ไม่เกิน 10 กิกะไบต์ ไม่รองรับ Defined function ไม่รองรับการทำ MySQL replication และไม่รองรับคำสั่ง MySQL ดังต่อไปนี้

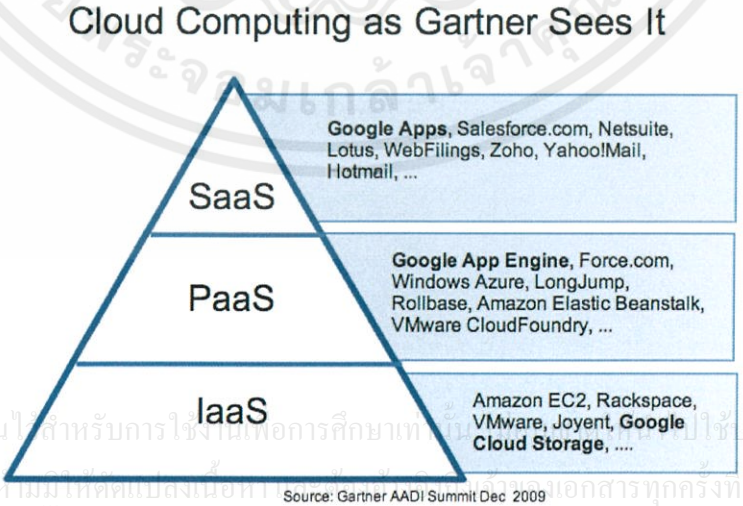
คำสั่งที่ 2.1 ข้อจำกัดของคำสั่งที่ไม่รองรับใน MySQL ในระบบคลาวด์

```
- LOAD DATA INFILE
- SELECT … INTO OUTFILE/DUMPFILE
- INSTLL/UNINSTALL PLUGIN …
- CREATE FUNCTION…
- LOAD_FILE ( )
```



รูปที่ 2.4 คุณสมบัติเด่นของ MySQL–Google cloud SQL ที่ใช้เป็นระบบฐานข้อมูล

2.2.3 Google App Engine คือ แพลตฟอร์มของการพัฒนาและการให้บริการพื้นที่ Application ของกูเกิลที่เปิดให้ผู้พัฒนาเว็บ Application สามารถเขียนโปรแกรมเข้าไปเชื่อมต่อกับโครงสร้างข้อมูลของกูเกิลได้มากขึ้น โดยผู้พัฒนานั้นไม่ต้องยุ่งยากกับการติดตั้งและตั้งค่าเว็บเซอร์วิส เพียงสมัครเข้าใช้งานแล้วทำการอัพโหลด Source code ไปที่ App Engine ของกูเกิล ก็จะได้ Web application ที่จะอิงกับสถาปัตยกรรมของกูเกิล ไม่ว่าจะเป็นระบบฐานข้อมูลหรือโครงสร้างพื้นฐานอื่นๆ โดยในปัจจุบัน Google App Engine รองรับได้ 3 ภาษาได้แก่ Java , Python และ Go



รูปที่ 2.5 ตัวอย่าง Cloud computing services ของ Google

2.3 ทรานแซกชั่น (Transaction)

แต่เดิมในยุคแรกๆ ทรานแซกชั่นหมายถึง ไฟล์ที่ใช้เก็บรายการเปลี่ยนแปลงจากแฟ้มข้อมูลหลัก ซึ่งเป็นระบบไฟล์แบบมีลำดับและมีการจัดเรียงอย่างเรียบร้อย โดยอดีตข้อมูลจะถูกเก็บในม้วนกระดาษ หรือเทป เนื่องจากในสมัยก่อนยังไม่มีฮาร์ดดิสก์

ในปัจจุบันทรานแซกชั่น คือ กลุ่มคำสั่งในภาษาฐานข้อมูลในระดับลอจิคอล (Logical level) ซึ่งรวมหลาย ๆ การดำเนินการ หรือเป็นกลุ่มของการดำเนินการบนฐานข้อมูล (Database operation) เช่น กลุ่มของคำสั่งภาษา SQL (Standard Query Language) หรืออาจเป็นกลุ่มของการเรียกเมธอด (Method) ที่ยอมให้มีการละเมิดกฎที่บังคับความถูกต้องของข้อมูล (Integrity constraints) เป็นการภายในได้แต่ก่อนและหลังทรานแซกชั่นนั้น ฐานข้อมูลต้องมีความถูกต้อง ซึ่งในที่นี้กฎที่บังคับความถูกต้องของข้อมูลนั้นรวมถึงกฎทางธุรกิจ (Business rule) โดยกฎเหล่านี้จะเก็บอยู่บนฐานข้อมูลและบังคับใช้โดยระบบจัดการฐานข้อมูล (Database Management System)

2.3.1 คำสั่งพื้นฐานที่ใช้ในการดำเนินการทรานแซกชั่น (Transaction statement)

ในส่วนของ การพัฒนาบน Google Cloud SQL นั้น (รูปที่ 2.6) ในทางปฏิบัติมีการใช้งานคำสั่งแบบ MySQL ในการดำเนินทรานแซกชั่นต่างๆ ดังต่อไปนี้

1) คำสั่ง BEGIN หรือ START TRANSACTION เป็นคำสั่งที่ใช้ในการเริ่มต้นทรานแซกชั่น
2) คำสั่ง COMMIT เป็นจุดที่ใช้ในการจบการดำเนินการทรานแซกชั่น ซึ่งยืนยันการเปลี่ยนแปลงอย่างถาวรลงในฐานข้อมูล
3) คำสั่ง ROLLBACK เป็นจุดที่ใช้ในการจบการดำเนินการทรานแซกชั่น ซึ่งยกเลิกการเปลี่ยนแปลงอย่างถาวรลงในฐานข้อมูล
4) SET AUTOCOMMIT เป็นการใช้ในการกำหนดโหมดของ COMMIT แบ่งได้เป็น

4.1) SET AUTOCOMMIT ON คือ ทุกคำสั่งที่ดำเนินการจะเปรียบเสมือน COMMIT ทุกคำสั่งทันทีที่ปฏิบัติเสร็จ

4.2) SET AUTOCOMMIT OFF คือ การดำเนินการกระทำไปเรื่อยๆ จนกว่าจะมีคำสั่ง COMMIT โดยผู้ใช้งาน ซึ่งในการใช้ BEGIN หรือ START TRANSACTION จะเป็นการ SET AUTOCOMMIT OFF จนกว่าจะปิดทรานแซกชั่น ด้วยคำส่ง COMMIT หรือ ROLLBACK

```
START TRANSACTION [WITH CONSISTENT SNAPSHOT]
BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}
```

รูปที่ 2.6 รูปแบบการประกาศทรานแซกชั่น

ดังนั้นเมื่อคำสั่งพื้นฐานต่างๆ ดำเนินการอยู่ในทรานแซกชั่น จะก่อให้เกิดสถานะของทรานแซกชั่นที่เกิดขึ้นได้ 5 สถานะดังต่อไปนี้

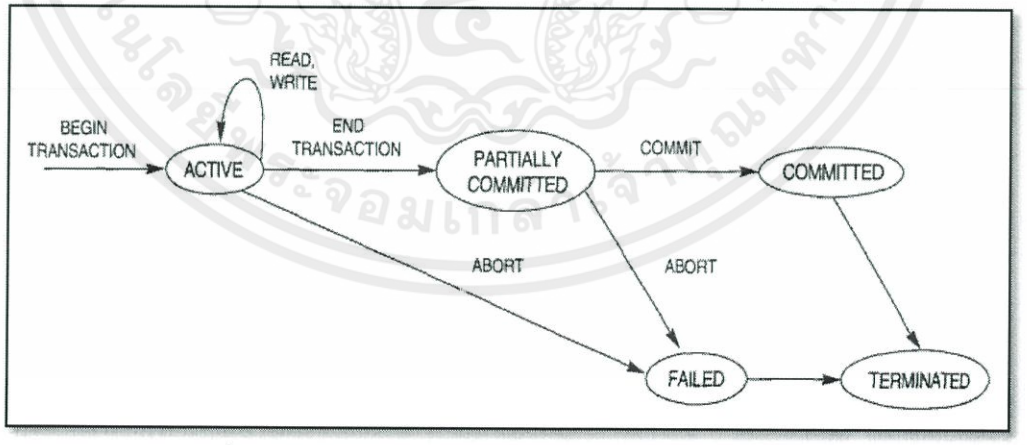
1) Active เป็นสถานะที่ต่อจากการเริ่มต้นทรานแซกชั่น ซึ่งอยู่ในระหว่างการปฏิบัติคำสั่งทรานแซกชั่น โดยในสถานะนี้ จะมีการดำเนินการอ่านหรือเขียนค่าลงฐานข้อมูล

2) Partially committed เป็นสถานะเมื่อทรานแซกชั่นปฏิบัติคำสั่งสุดท้ายเสร็จ

3) Failed เป็นสถานะที่ทรานแซกชั่นไม่สามารถปฏิบัติงานต่อไปได้ เนื่องจากปัญหาการล่มของเซิฟเวอร์ เป็นต้น

4) Abort เป็นสถานะของทรานแซกชั่นที่ถูก ROLLBACK ซึ่งคือการถูกยกเลิกการเปลี่ยนแปลงตั้งแต่ sync point (จุดเริ่มต้นและลงท้ายทรานแซกชั่น) ล่าสุดจนถึงจุด ROLLBACK

5) Committed เป็นสถานะที่ยืนยันการเปลี่ยนแปลง ซึ่งค่าที่เปลี่ยนไปจะคงอยู่อย่างถาวร แม้ว่าหลังจากนั้นอาจมีความล้มเหลวของระบบเกิดขึ้น



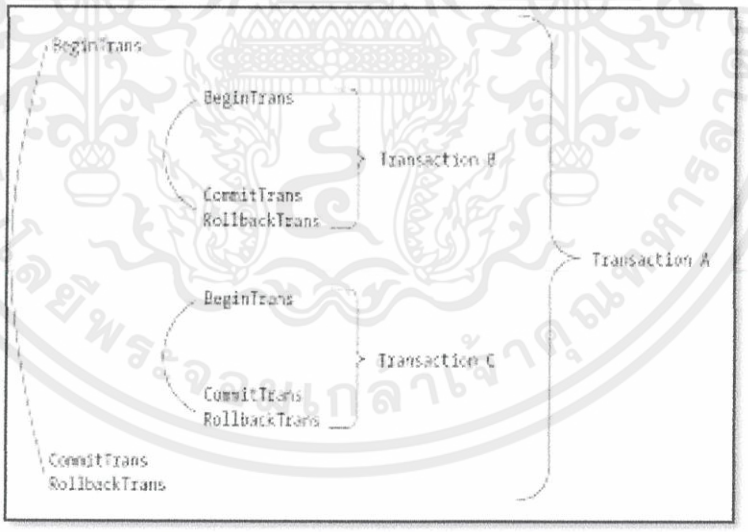
รูปที่ 2.7 แผนภาพสถานะของการดำเนินการทรานแซกชั่น

2.3.2 คุณสมบัติของทรานแซกชั่น (ACID properties)

จากที่กล่าวว่า ทรานแซกชั่น คือหน่วยของการจัดการของโปรแกรม ซึ่งจะมีการเข้าถึง และ การเปลี่ยนแปลงข้อมูลในแถวของตารางภายในฐานข้อมูล ในกระบวนการ การ

ดำเนินการของทรานแซกชั่นเพื่อรักษาความถูกต้องของข้อมูลนั้น ระบบการจัดการฐานข้อมูลที่ดีจึงจำเป็นที่จะต้องมีคุณสมบัติเฉพาะที่เรียกว่า ACID ซึ่งเป็นตัวอักษรย่อของแต่ละคุณสมบัติ ซึ่งนำมาใช้ในการควบคุมทรานแซกชั่นที่มีการดำเนินการพร้อมกัน (Concurrency control) และเป็นวิธีการในการกู้คืนทรานแซกชั่นของระบบจัดการฐานข้อมูล

1) Atomicity กล่าวคือ ทุกๆการดำเนินการย่อยบนทรานแซกชั่น จะถูกมองเป็นหน่วยเดียวกันแบ่งย่อยไม่ได้ หากต้องสำเร็จทั้งหมด หรือ ล้มเหลวทั้งหมด โดยทั่วไปทรานแซกชั่นจะเป็นไปตาม Atomicity แต่ยกเว้น กรณีของทรานแซกชั่นที่มีมนุษย์เข้าไปเกี่ยวข้อง (Human interaction) ซึ่งถ้าเกิดความล้มเหลวไม่ควรจะ rollback ทั้งทรานแซกชั่น ดังนั้นจึงมีการนำกลไกที่จะช่วยให้การดำเนินการให้ทรานแซกชั่นไม่ถูกยกเลิกทั้งหมด (Nested transaction) ซึ่งหากระบบเกิดความล้มเหลวจะถูกยกเลิกแค่บางส่วนของการดำเนินการ ซึ่งกลไกภายในจะแบ่งทรานแซกชั่น ออกเป็นทรานแซกชั่นย่อยๆ โดยมีจุดเซฟพอยท์ (Save point) เป็นขอบเขตการแบ่งของทรานแซกชั่นย่อย ซึ่งหากระบบล้มเหลว กลไกจะเกิดการ rollback ถึงแค่จุดเซฟพอยท์เดิม โดยทั้งหมดถูกครอบด้วยคำสั่งพื้นฐานในการเริ่มทรานแซกชั่น และจุดจบของทรานแซกชั่น ซึ่งจะเรียกจุดเริ่มต้นและลงท้ายของทรานแซกชั่นว่า จุดซิงค์พอยท์ (Sync point)



รูปที่ 2.8 Nested transaction

โดยในคลาวด์เมื่อทำการศึกษาแล้ว เราจะพบว่าเราสามารถทำการตั้งค่าต่างๆ เหล่านี้ได้ โดยมีรูปแบบการใช้คำสั่งดังนี้

คำสั่งที่ 2.2 การประกาศ Nested transaction

```
SAVEPOINT identifier
.....
.....
ROLLBACK TO SAVEPOINT identifier
```

ซึ่งเมื่อการดำเนินการมาถึงจุดที่เกิด Rollback จะทำการย้อนกลับไปถึงเพียงแค่ จุดเซฟพอยท์ที่กำหนดไว้ ซึ่งขึ้นอยู่ว่าจะให้ย้อนกลับไปที่ identifier ใด ซึ่งเป็นชื่อที่ใช้เพื่อคั่นขอบเขตของ ทรานแซกชั่น ทั้งนี้โดยทั่วไปถ้าไม่ระบุว่า ถ้าเกิดการ Rollback แล้วให้กลับไปยังจุดเซฟพอยท์ใด ทรานแซกชั่นนั้นจะถูกยกเลิกทั้งทรานแซกชั่น เช่นกรณีที่ระบบล้มเหลว แต่กรณีของการยกเลิกที่เกิดจากการ timeout ของบางคำสั่ง อันเนื่องมาจากการติดล็อคของทรานแซกชั่น การ Rollback นั้นจะเป็นแค่คำสั่งนั้นๆ ที่ทำให้เกิดการล็อค แต่ไม่ได้ Rollback ทั้งหมดเหมือนกรณีของปัญหาจากระบบหรือโปรแกรม

2) Consistency คือการรักษาความถูกต้องของฐานข้อมูลตามกฎบังคับความถูกต้อง ซึ่งในบางผลิตภัณฑ์อาจยอมให้มีการละเมิดกฎเป็นการภายในได้ชั่วคราว ซึ่งตรงตามนิยามของทรานแซกชั่น แต่เมื่อจบทรานแซกชั่นแล้ว ฐานข้อมูลต้องมีความถูกต้องและสอดคล้องกับกฎดังกล่าว ซึ่งการเลื่อนการบังคับกฎบังคับความถูกต้องของฐานข้อมูล (Deferred integrity constraints enforcement) ในระบบคลาวด์ของ Google Cloud SQL ที่ทำการทดลองใช้งานนั้นไม่ได้สนับสนุนการเลื่อนการบังคับกฎดังกล่าว นอกจากนี้กฎที่ใช้บังคับความถูกต้อง สามารถตั้งค่ากฎต่าง ๆ ได้เมื่อทำการสร้างตารางใหม่ หรืออาจทำการตรวจสอบหลังจากการสร้างตาราง ซึ่งมีการรูปแบบการเขียนได้สองแบบดังนี้

คำสั่งที่ 2.3 การ check constraint กำหนดตอนสร้างตารางแบบที่ 1

```
CREATE TABLE table_name
(
      Column_name column_typs CHECK (condition)
);
```

คำสั่งที่ 2.4 การcheck constraint กำหนดตอนสร้างตารางแบบที่ 2

```
CREATE TABLE table_name
(
column_name column_type,
CONSTRAINT constraint_name CHECK (condition)
);
```

คำสั่งที่ 2.5 การ check constraint หลังสร้างตาราง

```
CREATE TABLE table_name
(
column_name column_type
);
ALTER TABLE table_name
ADD CONSTAINT constraint_name CHECK (condition);
```

หมายเหตุ แต่ในทางปฏิบัติแล้วคำสั่งในการตั้งค่า integrity rules (การใช้คำสั่ง check ตามคำสั่งด้านบน) ของตัว MySQL นั้น ไม่สามารถนำไปใช้งานจริงได้ แม้ว่าจะทำผ่านระบบคลาวด์หรือไม่ก็ตาม กล่าวคือ มีการกล่าวถึงคำสั่ง CHECK เหล่านี้ในคู่มือของ MySQL แต่ไม่ถูกพัฒนาในระบบจริงที่ใช้งานกับระบบจัดการฐานข้อมูล จึงเป็นเหตุผลว่า เมื่อเราสร้างตารางที่กำหนดสิทธิ์ต่างๆแล้ว ค่าที่ไม่ถูกต้องจึงสามารถเข้าไปเก็บในฐานข้อมูลได้ โดยปัญหานี้ถูกเก็บไว้เป็น Reported bug ของ MySQL เป็นระยะเวลา 8 ปีแล้ว ดังนั้นหากไม่สามารถบังคับกฎความถูกต้องได้ ทำให้ค่าที่ inputเข้าไปนั้นไม่ถูกต้อง ซึ่งปัญหาดังกล่าว ต้องใช้ Trigger เข้ามาช่วย โดยที่เป็นวัตถุของฐานข้อมูลตัวหนึ่งที่ใช้ทำงานร่วมกับตาราง

Trigger เป็นวัตถุที่ช่วยในการจัดการกับตารางในฐานข้อมูล โดยการใช้งานขึ้นอยู่กับวัตถุประสงค์ เช่น เพื่อตรวจสอบค่าที่จะ insert เข้าไปในตาราง หรือใช้เพื่อการคำนวณค่าที่เกี่ยวข้องในการ update โดยที่การตั้งค่าต่างๆ จะถูกเก็บไว้ที่ เซิฟเวอร์ในลักษณะของ Stored program โดยมีผลที่แถวหรือคอลัมน์ที่ต้องการตั้งค่า integrity rules หรือ business rules และบังคับใช้งานคำสั่งโดย DBMS

ยกตัวอย่างเพื่อแสดงรูปแบบของการใช้งานคำสั่ง ในการพัฒนาแอพพลิเคชั่นบน Google App engine นั้น กฎที่สำคัญอย่างหนึ่งของแอพพลิเคชั่นร้านขายหนังสือออนไลน์คือ หนังสือที่จะสั่งซื้อได้นั้น จะต้องมีค่าไม่น้อยกว่าศูนย์ ซึ่งถือเป็นกฎที่ต้องดำเนินการผ่านเว็บแอพพลิเคชั่น และให้ DBMS เป็นตัวจัดการ และ นำผลลัพธ์ของโปรแกรมมาแสดงที่หน้าเว็บแอพพลิเคชั่นว่าสั่งซื้อได้ หรือไม่ เป็นต้น

```
mysql> delimiter //
mysql> CREATE TRIGGER upcheck BEFORE UPDATE ON books
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.amount < 0 THEN
    ->         SIGNAL SQLSTATE '80000'
    ->          SET MESSAGE_TEXT='ERROR';
    ->     END IF;
    -> END;//
```

รูปที่ 2.9 ตัวอย่างคำสั่งที่ใช้ในการประกาศ Trigger

<u>คำอธิบายเกี่ยวกับคำสั่งข้างต้น</u>

1) Delimiter ใช้ในการระบุจุดตำแหน่งที่จะจบของ โปรแกรมให้ชัดเจน เช่น // และลดปัญหาการเกิดข้อผิดพลาดต่างๆ โดยให้สามารถใช้ ; ภายใน Stored program ได้
2) CREATE TRIGGER มีรูปแบบการประกาศดังนี้

คำสั่งที่ 2.6 รูปแบบของการประกาศการใช้ integrity rules ใน MySQL

```
CREATE [DEFINER= {user | CURRENT_USER}] TRIGGER trigger_name
{BEFORE|AFTER}
{UPDATE|INSERT|DELETE}
ON table_name
FOR EACH ROW
trigger_statements
```

โดยที่จะเริ่มต้นทำการสร้างชื่อของTrigger หลังจากนั้นกำหนดว่าจะให้ทำ Trigger ก่อน หรือ หลัง การ UPDATE, INSERT หรือ DELETE และกระทำบนตารางใด หลังจากนั้นให้ใส่คำสั่ง BEGIN ... ENG ครอบเอาไว้ และภายในเป็นเงื่อนไขที่ใช้ในการตรวจสอบ เช่น ในที่นี้ ก่อนการอัพเดทค่าลงไปที่ตาราง books ถ้าจำนวนที่ใส่เข้ามาทำให้มีค่า amount น้อยกว่า ศูนย์ จะไม่ทำการอัพเดทคำสั่งที่รับมาจากแอพพลิเคชั่น ซึ่งทำให้เกิดการ Rollback และแสดงผลลัพธ์ออกมาเป็นข้อความ Exception (ในที่นี้คือ คำว่า ERROR) แต่ถ้าไม่น้อยกว่าศูนย์ ก็จะการอัพเดทค่าใหม่ในตารางที่ระบุ

คำสั่งที่ 2.7 การกำหนดเงื่อนไขภายใน Trigger ด้วยข้อความ Exception

```
IF NEW.amount < 0 THEN
SIGNAL SQLSTATE '80000'
SET MESSAGE_TEXT='ERROR';
```

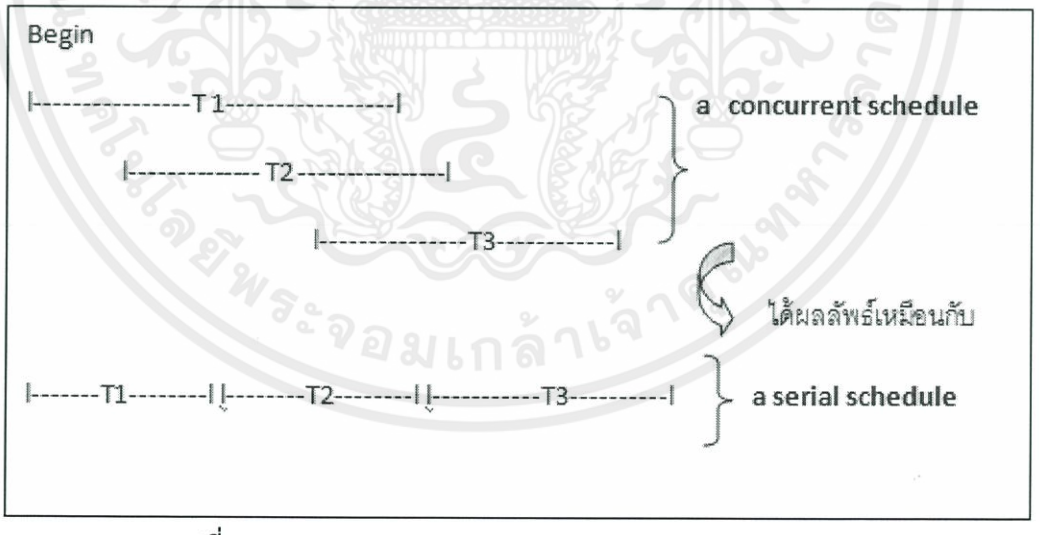
หมายเหตุ: การใช้ SIGNAL SQLSTATE ไม่สามารถใช้ได้ สำหรับ MySQL เวอร์ชั่นที่ต่ำกว่า MySQL 5.2 ทั้งนี้ระบบฐานข้อมูลบนคลาวด์ใช้เวอร์ชั่น MySQL 5.5 จึงสามารถใช้งานในส่วนนี้ได้

คำสั่งที่ 2.8 การทดลองอัพเดทค่าที่ทำให้เกิด Exception ใน interactive SQL

```
sql> update books set StockNum=StockNum-8 where ISBN=2;
ERROR 1644 (SQLSTATE HY000): ERROR
```

3) Isolation คือ เมื่อมีหลายทรานแซกชั่นถูกปฏิบัติอยู่ในช่วงเวลาเดียวกัน ต้องไม่มีการรบกวนกันของทรานแซกชั่น แม้จะใช้ในฐานข้อมูลเดียวกัน โดย การที่มีหลายทรานแซกชั่นกระทำพร้อมๆกันนั้น ระบบจัดการฐานข้อมูลจะต้องมีการจัดลำดับ การปฏิบัติคำสั่งในทรานแซกชั่น (Schedule) ที่มีความถูกต้องและเหมาะสม

3.1) การปฏิบัติคำสั่งแบบขนาน (Concurrent Execution) โดยทั่วไปกระบวนของทรานแซกชั่นควรอนุญาติให้มีการทำงานหลายงานพร้อมกันได้ ทั้งนี้เพื่อการใช้ทรัพยากรภายในระบบให้มีประสิทธิภาพ และ ช่วยลดเวลาที่เกิดจากการรอทรานแซกชั่นอื่น ทั้งนี้เมื่อมีการทำงานหลายทรานแซกชั่นวิ่งร่วมกันจึงต้องพิจารณาถึงความถูกต้องของข้อมูลเป็นหลักและต้องไม่เห็นการทำงานของกัน



รูปที่ 2.10 Concurrent schedule และ Serial schedule

จากรูปที่ 2.10 แสดงถึงการจัดลำดับ การจัดการคำสั่งในทรานแซกชั่น โดยทั่วไปการดำเนินการในช่วงระยะเวลาปกติ จะมีการจัดลำดับคำสั่งแบบอนุกรม (Serial schedule) ซึ่งคำสั่งของทรานแซกชั่นเดียวกันจะถูกปฏิบัติต่อเนื่องตั้งแต่ ต้นจนจบ โดยไม่มีคำสั่งของทรานแซกชั่นอื่นมาขัดขวางในช่วงการทำงาน แต่การทำงานจริงสิ่งที่ระบบอยากได้คือการทำงาน

ที่ได้ผลลัพธ์เหมือนการทำงานแบบอนุกรม แต่สามารถทำงานพร้อมๆกันขนานกันไปได้ (Concurrent schedule) เพราะช่วงเวลาเดียวกันทรานแซกชั่นอื่นที่เข้ามา อาจมีดำเนินการที่ต่างแถว ต่างตารางกันซึ่งสามารถทำงานขนานกันได้ และ ไม่ทำให้ฐานข้อมูลเกิดความไม่ถูกต้อง ซึ่งทำให้การทำงานมีความรวดเร็วมากขึ้นกว่าการทำงานแบบอนุกรม โดยเรียก การจัดลำดับคำสั่งแบบขนานว่า Concurrent schedule ซึ่งหากการดำเนินการดังกล่าวได้ผลลัพธ์เหมือน Serial schedule จะเรียกว่า Concurrent serializable schedule

สมมติให้ ในช่วงระยะเวลาหนึ่งมีการดำเนินการสองทรานแซกชั่น ซึ่งเกี่ยวข้องกับการโอนเงินจากบัญชีหนึ่งไปอีกบัญชีหนึ่ง โดยเริ่มให้ ลำดับของทรานแซกชั่นที่เข้ามาเป็น T_1 และ T_2 ตามลำดับ โดยกำหนดให้เป็น T_1 ทำการอ่านค่าของบัญชี A มาแล้ว โอนเงิน 50 บาท จากบัญชี A ไปให้ บัญชี B ในขณะที่ทรานแซกชั่น T_2 เริ่มต้นจากการอ่านค่าของบัญชี A และทำการโอนเงิน 10 เปอร์เซ็นต์ของ A ให้บัญชี B โดยให้ A เริ่มต้นมีเงินในบัญชี 1,000 บาทและบัญชี B มีเงิน 2,000 บาท

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	

รูปที่ 2.11 Serial schedule

จากรูปที่ 2.11 เป็นการแสดง ลำดับการทำงานระหว่างทรานแซกชั่น ซึ่งเราจะเห็นว่า การทำงานดังกล่าวนั้นต้องให้ทรานแซกชั่น T_1 ทำงานให้เสร็จก่อน T_2 จึงจะเริ่มต้นทรานแซกชั่นได้ ซึ่งเป็นการจัดลำดับแบบอนุกรม ในขณะที่ ถ้าต้องการให้การทำงานเกิดขึ้นพร้อมกันได้นั้น ผลลัพธ์ที่ควรได้ต้องเท่ากับ แบบที่ทำทีละทรานแซกชั่นเพื่อให้เป็นแบบ concurrent serializable schedule โดยที่ข้อพึงระวังในการ ทำหลายทรานแซกชั่นพร้อมกันคือ การดำเนินการดังกล่าวอาจทำให้ผลลัพธ์สุดท้ายไม่เหมือนกับ การทำแบบทีละ ทรานแซกชั่นได้ ซึ่งอาจแสดงได้ดังนี้

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

รูปที่ 2.12 Concurrent schedule ที่ได้ผลลัพธ์ตรงกับแบบ Serial schedule

จากรูปที่ 2.12 แสดงให้เห็นการปฏิบัติหลายทรานแซกชั่นร่วมกันในช่วงเวลาหนึ่ง ซึ่งเมื่อพิจารณาผลลัพธ์ดังกล่าว แล้วจะได้ค่าเหมือนกันกับแบบที่ปฏิบัติทีละทรานแซกชั่น เนื่องจากสามารถสลับคำสั่งที่ไม่ Conflict กันได้ เพราะดำเนินการคนละ Data item กัน (เป็น Conflict equivalent schedule) จึงเป็น Conflict serializable schedule แต่หากพิจารณาอีกกรณีซึ่งปฏิบัติทรานแซกชั่นร่วมกัน แต่ได้ผลลัพธ์สุดท้ายที่ไม่เหมือนกัน ดังรูปที่ 2.13

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

รูปที่ 2.13 Concurrent schedule ที่ได้ผลลัพธ์สุดท้ายไม่ตรงกับแบบ Serial schedule

ในการตรวจสอบว่าผลลัพธ์สุดท้ายเหมือนกับ การดำเนินการแบบอนุกรมหรือไม่นั้น สามารถทำได้ 2 วิธี คือใช้การแทนค่าตัวแปร โดยทั่วไปจะใช้การแทนค่าเพื่อตรวจสอบผลลัพธ์ที่ถูกต้องโดยต้องพิจารณาเข้าไปดูถึงระดับซอสโค้ด (Source code) หรือหากเป็น

ระบบจัดการฐานข้อมูลจะ พิจารณา จากคำสั่ง Read หรือ Write เนื่องจากเป็นคำสั่งที่มีผลกับความถูกต้องของการทำงาน

3.2) การจัดลำดับการสลับกันอย่างสมดุล (Conflict Serializability) เมื่อพิจารณาถึงการจัดลำดับการปฏิบัติคำสั่งในทรานแซกชั่น จากตัวอย่างที่ผ่านมา ถ้าทั้งสองทรานแซกชั่นปฏิบัติบนคนละแถว (Data item) เราสามารถทำการสลับตำแหน่งคำสั่งได้ซึ่งไม่กระทบกับผลลัพธ์ เราจะเรียกการจัดลำดับคำสั่งทรานแซกชั่นทั้งสองว่า Conflict equivalent แต่ในทางกลับกันหากทรานแซกชั่นต่างดำเนินการปฏิบัติบนแถวเดียวกัน จะทำให้ลำดับของการวางตำแหน่งมีผลต่อผลลัพธ์ โดยเมื่อทำการ ใช้คำสั่งอ่าน หรือ เขียน สามารถพิจารณาการเกิดกรณีได้ 4 กรณี

3.2.1) กรณีที่เป็นการอ่านทั้งคู่ ทรานแซกชั่น T_1=READ (A) และ T_2 =READ (A) กรณีนี้ลำดับของทั้งสองไม่มีผล ดังนั้นสามารถสลับตำแหน่งของคำสั่งได้

3.2.2) กรณีที่เป็น การอ่านกระทำก่อนการเขียน ทรานแซกชั่น T_1=READ (A) และ T_2=WRITE (A) ดังนั้น T_1 จะไม่อ่านข้อมูลที่เขียนโดย T_2 และ T_2 จะต้องเขียนข้อมูลหลังจากที่ T_1 อ่านค่าแล้ว

3.2.3) กรณีที่การเขียนกระทำก่อนการอ่าน ทรานแซกชั่น T_1=WRITE (A) และ T_2=READ (A) ดังนั้น T_2 จะต้องอ่านข้อมูลหลังจาก T_1 เขียนค่าแล้ว และ T_1 จะต้องเขียนข้อมูลก่อน T_2 จะอ่านค่า

3.2.4) กรณีเป็นการเขียนทั้งคู่ ทรานแซกชั่น T_1=WRITE (A) และ T_2=WRITE (A) โดยการกระทำทั้งสองไม่มีผลกระทบต่อกันโดยตรง แต่จะมีผลต่อการอ่านที่ตามมา ดังนั้นจึงไม่สามารถสลับค่าได้

การจะระบุว่าสองคำสั่งใดๆ จะ conflict กันก็ต่อเมื่อสองคำสั่งนั้นมาจากคนละทรานแซกชั่นกัน ปฏิบัติบนแถว (Data item) เดียวกันและหนึ่งในคำสั่งนั้นเป็นการเขียน และจะไม่คอนฟลิกกันก็ต่อเมื่อทั้ง สองคำสั่งมาจากทรานแซกชั่นเดียวกัน หรือ ถ้าต่างทรานแซกชั่นแต่ดำเนินการบนคนละแถวข้อมูล หรือ ดำเนินการบนแถวข้อมูลเดียวกันแต่เป็นการอ่านทั้งคู่

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

รูปที่ 2.14 Serial schedule

จากรูปที่ 2.14 คือการจัดลำดับคำสั่งแบบอนุกรม เนื่องจากเป็นการกระทำของทรานแซกชั่นเกิดขึ้นเพียงทรานแซกชั่นเดียวในช่วงเวลาหนึ่ง และ เมื่อพิจารณาจากรูปที่ 2.15 เปรียบเทียบกับรูปที่ 2.15 ซึ่ง Conflict equivalent กับการจัดลำดับคำสั่งแบบอนุกรม เพราะ สามารถแปลงรูปไปเป็นแบบการจัดลำดับคำสั่งแบบอนุกรมได้ โดยทำการสลับตำแหน่งคำสั่งที่ไม่คอนฟลิกกัน จึงเรียกว่าเป็น Conflict serializable schedule ซึ่งเป็นการจัดลำดับแบบอนุกรมที่มีผลลัพธ์คำตอบที่เหมือนกัน แต่ดำเนินการแบบขนาน

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

รูปที่ 2.15 Conflict serializable schedule

3.3) การกู้คืน (Recoverability) จากที่กล่าวถึงการจัดลำดับการปฏิบัติคำสั่งในทรานแซกชั่น ที่สามารถดำเนินการได้จากมุมมองของความถูกต้องของฐานข้อมูลที่ผ่านมา เราถือ

ว่าไม่มีความล้มเหลวของทรานแซกชั่น แต่ในความเป็นจริงปัญหาเหล่านี้เกิดขึ้นได้เสมอ ซึ่งจะกล่าวถึงผลกระทบของความล้มเหลวของทรานแซกชั่น ระหว่างการปฏิบัติคำสั่งแบบขนาน

เมื่อทรานแซกชั่นแรกที่ทำงานขนานกันไปเกิดความล้มเหลว จึงจำเป็นที่จะต้องทำการยกเลิกการทำงานที่มีผลกระทบ ทั้งนี้เนื่องจากคุณสมบัติของ Atomicity ดังนั้นหากทรานแซกชั่นแรกล้มเหลว ทรานแซกชั่นอื่นที่ตามมาต้องถูกยกเลิกด้วย ดังนั้นจึงต้องมีการจำกัดประเภทของการจัดลำดับการปฏิบัติคำสั่งในทรานแซกชั่นของระบบ

3.3.1) การจัดลำดับการปฏิบัติคำสั่งที่สามารถกู้คืนได้ (Recoverable Schedules) พิจารณาจากรูปที่ 2.16 ซึ่ง T_9 เป็นทรานแซกชั่นที่ทำการทำคำสั่งอ่านเพียงอย่างเดียว และสมมติว่า T_9 จะทำการ commit ทรานแซกชั่นทันทีหลังจากทำการอ่านค่าเรียบร้อยแล้ว ดังนั้นจะเห็นได้ว่า ทรานแซกชั่นที่มาที่หลังจะทำการ commit ค่าก่อนทรานแซกชั่นแรก หากสมมติว่า T_8 ล้มเหลว และ T_9 ได้ทำการอ่านค่า A ที่เขียนโดย T_8 เราควรทำการยกเลิก T_9 เพื่อความถูกต้องของข้อมูลที่ T_9 อ่านค่าผิดไป แต่ในสถานการณ์นี้จะไม่สามารถกู้คืนความผิดพลาดที่เกิดจาก T_8 ได้ (Nonrecoverable schedule) ซึ่งในระบบจัดการฐานข้อมูลส่วนใหญ่จะไม่อนุญาตให้เกิดกรณีดังกล่าวขึ้น โดยการจัดลำดับคำสั่งต้องสามารถกู้คืนได้ เรียกว่า Recoverable schedule ซึ่งกำหนดไว้ว่าทั้งสองทรานแซกชั่น เมื่อทรานแซกชั่นที่สองทำการอ่านค่าที่ถูกเขียนโดยทรานแซกชั่นแรก ทรานแซกชั่นแรกต้อง commit ก่อนที่ทรานแซกชั่นที่สองจะดำเนินการ commit

T_8	T_9
read(A)	
write(A)	
	read(A)
read(B)	

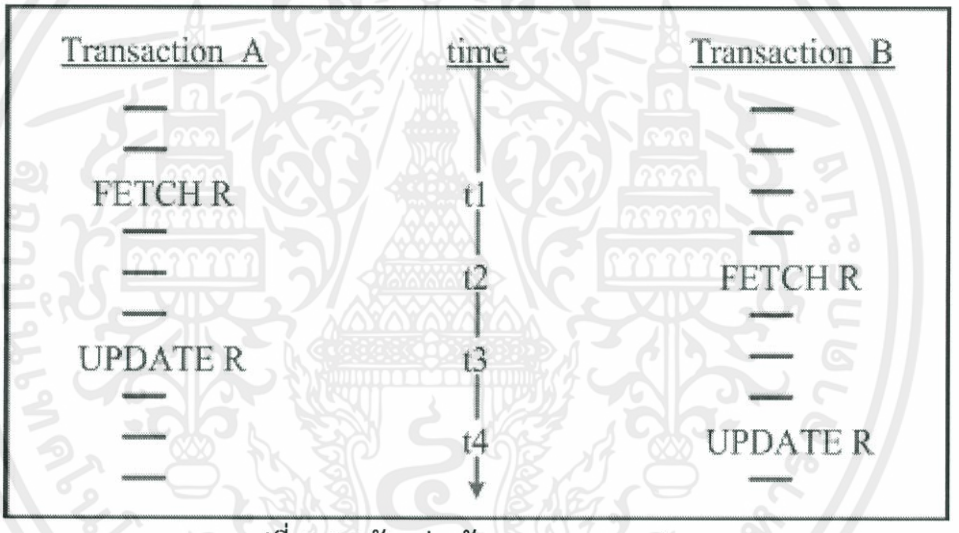
รูปที่ 2.16 Recoverable schedule

3.3.2) การจัดลำดับการปฏิบัติคำสั่งที่ไม่เชื่อมโยง (Cascadeless Schedule) การกู้คืนจากความล้มเหลวของทรานแซกชั่นอาจเกิดการยกเลิกในหลายทรานแซกชั่น สมมติให้ทรานแซกชั่น T_{10} ทำการเขียนค่าของ A ซึ่งถูกอ่านค่าต่อโดย T_{11} และ ทำการเขียนค่า A ซึ่งถูกอ่านต่อโดย T_{12} และหากที่จุด T_{10} เกิดความล้มเหลวทำให้ต้องถูก rollback ดังนั้น T_{11} ที่ขึ้นตรงต่อ T_{10} และ T_{12} ที่ขึ้นตรงต่อ T_{11} ต้องถูก rollback โดยเหตุการณ์ที่เกิดจากทรานแซกชั่นเดียวทำให้เกิดการ rollback ทั้งชุดเรียกว่า Cascading rollback โดยการจัดลำดับการปฏิบัติคำสั่งที่ไม่

เชื่อมโยงกัน หมายถึง แต่ละสองทรานแซกชั่น T_1 และ T_2 ใดๆ T_2 จะทำการอ่านค่าได้ T_1 จะต้องทำการดำเนินการ commit ก่อน ซึ่งการจัดลำดับการปฏิบัติคำสั่งในรูปแบบนี้ สามารถลดปัญหาที่จะต้อง rollback หลายครั้งได้ และถ้าเป็นการจัดลำดับการปฏิบัติคำสั่งที่ไม่เชื่อมโยง ย่อมเป็นการจัดลำดับการปฏิบัติคำสั่งที่สามารถกู้คืนได้

3.4) ปัญหา 4 ข้อของการประมวลผลทรานแซกชั่นแบบขนาน (The 4 concurrency control problems)

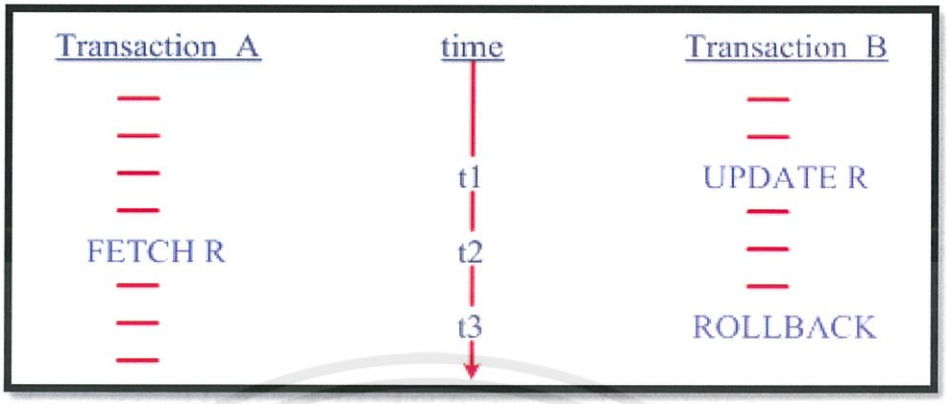
3.4.1) ปัญหา Lost update ปัญหาเกิดจากเมื่อทรานแซกชั่นทั้งสองทำการอ่านค่าของข้อมูลชุดเดียวกัน ทำให้ใครที่ทำการเขียนที่หลังอาจทำการทับค่าของทรานแซกชั่นก่อนหน้า



รูปที่ 2.17 ตัวอย่างปัญหา Lost update

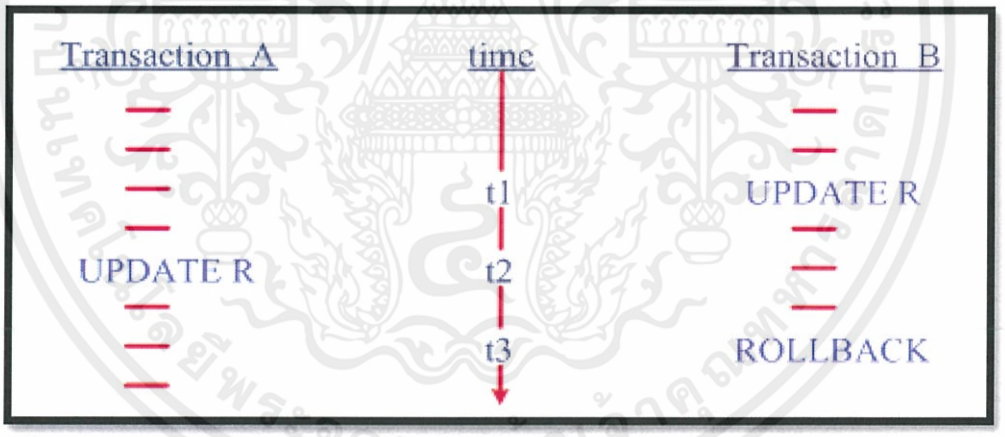
ยกตัวอย่างปัญหาเช่น หากค่าเริ่มต้นของ R=30 เมื่อ เวลา t_1 ทรานแซกชั่น A อ่านค่าได้เท่ากับ 30 เวลา t_2 ทรานแซกชั่น B อ่านค่าได้ 30 เท่ากัน เมื่อเวลา t_3 ทรานแซกชั่น A ทำการอัพเดทค่า A ให้เป็น 31 และเมื่อเวลา t_4นั้น B ทำการอัพเดทค่าเพิ่มเป็นสองเท่า และทำให้ B ได้ค่าเป็น 60 ซึ่งในความเป็นจริงแล้ว ถ้าการดำเนินการพร้อมกันของทรานแซกชั่นซึ่งเป็นแบบ Serial schedule นั้นค่าที่ได้ต้องเป็น 62 แต่เป็นเพราะ B นำค่าเดิมที่อ่านได้ไปทำการอัพเดท ทั้งนี้ทั้งสองทรานแซกชั่นล้วนไม่ Conflict serializable เพราะไม่สามารถทำการสลับคำสั่งแล้วเป็น Serial schedule ได้

3.4.2) The uncommitted dependency problem ปัญหาดังกล่าวเกิดขึ้นเนื่องจากการนำค่าข้อมูลไปใช้ซึ่งเป็นค่าที่ยังไม่ได้ commit แล้วถูก rollback



รูปที่ 2.18 ตัวอย่างปัญหา The uncommitted dependency problem **กรณีอ่านค่า**

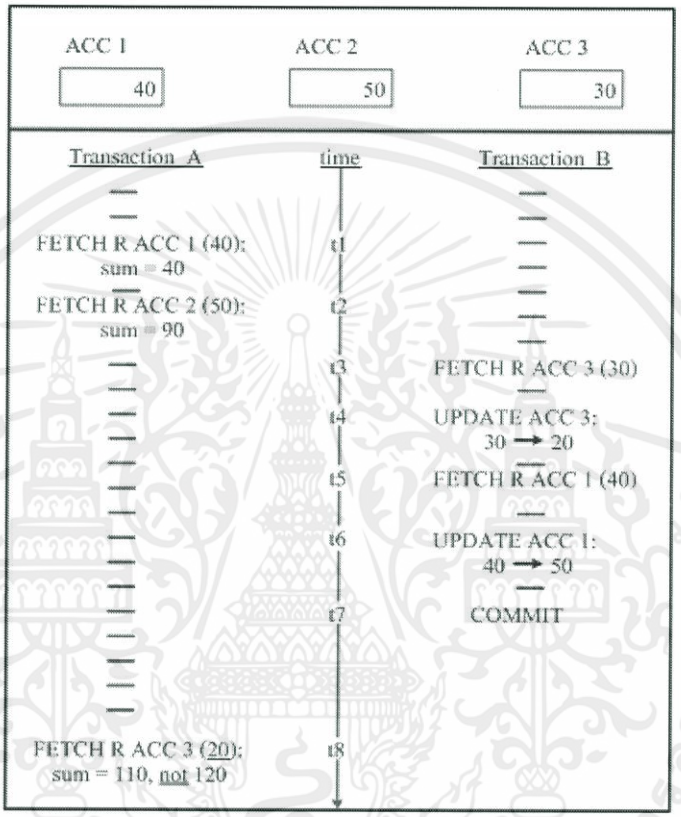
ยกตัวอย่างปัญหาเช่น ที่เวลา t_2 ทรานแซกชั่น A ทำการอ่านค่า R ซึ่งเป็นการนำค่าข้อมูลที่ยังไม่ได้ commit มาจากทรานแซกชั่น B แต่เมื่อเวลา t_3 ทรานแซกชั่น B เกิดการ rollback ทำให้ A ทำการอ่านค่าที่ผิดไป และถ้า ทรานแซกชั่น A ทำการ commit ก็จะทำให้เกิดปัญหาสถานการณ์ที่ไม่สามารถกู้คืนความผิดพลาดได้ (Nonrecoverable schedule)



รูปที่ 2.19 ตัวอย่างปัญหา The uncommitted dependency problem **กรณีอัพเดทค่า**

3.4.3) The inconsistent analysis problem ปัญหาดังกล่าวเกิดขึ้นเมื่อทรานแซกชั่นหนึ่งกำลังวิเคราะห์ข้อมูลที่เป็นการอ่านค่า (Read only) ระหว่างนั้นมีอีก ทรานแซกชั่นเข้ามาทำการอัพเดทค่าที่กำลังทำการวิเคราะห์อยู่ ยกตัวอย่างเช่น ทรานแซกชั่น A ทำการรวมนับค่าของทุกบัญชี และในระหว่างนั้น ทรานแซกชั่น B มีการอัพเดทค่าบางค่าของผู้ใช้งาน ซึ่งอาจทำให้ ทรานแซกชั่นแรกที่ทำการรวมค่าของทุกบัญชี เกิดการรวมค่าของผลลัพธ์ที่มีความคลาดเคลื่อนหรือผิดพลาดได้ ซึ่งในที่นี้ทำการเปลี่ยนแปลงจากบัญชีที่สาม จาก 30 เป็น 20 และบัญชีที่หนึ่ง ทำการอัพเดทค่าแล้วทำการ commit ซึ่งจากรูปปัญหาเกิดขึ้นเมื่อทรานแซกชั่นแรกทำการอ่านค่า ของ

บัญชีที่พึ่งทำการบวก และลบค่าใหม่ ทำให้ได้ผลรวมของผลลัพธ์ที่ไม่ถูกต้อง จากที่ควรจะเป็น 120 แต่ได้ผลรวมแค่ 110 เพราะ ก่อนที่บัญชี 3 จะถูก รวมค่า บัญชี 1 ได้ถูกรวมค่าไปแล้ว เลยทำให้ดูเสมือนเงินหายไป



รูปที่ 2.20 ตัวอย่างปัญหา The inconsistent analysis problem

3.4.4) The phantom phenomenon ปัญหาดังกล่าวคล้ายคลึงกับปัญหา 3.4.3 แต่ เปลี่ยนจากการอัพเดทเป็นการนำข้อมูลลงสู่ตาราง โดยเป็นปรากฏการณ์ที่มีการใช้คำสั่ง Insert เข้าไป และผลลัพธ์ของการนำข้อมูลลงสู่ตาราง (Insert) เห็นโดยทรานแซกชั่นที่ปฏิบัติคำสั่งก่อนหน้า

3.5) ระดับไอโซเลชั่นของทรานแซกชั่น (Transaction Isolation Levels) หรือเรียกว่าเป็นระดับของความถูกต้องในระบบจัดการฐานข้อมูล ซึ่งมีทั้งหมด 4 ระดับ

3.5.1) Serializable เป็นระดับที่มีการรับประกันว่าเป็น Conflict serializable schedule และรับประกัน Cascadeless schedule รวมถึงการแก้ปัญหา Phantom phenomenon ซึ่งเป็นปรากฏการณ์ที่มีการใช้คำสั่ง Insert เข้าไป และผลลัพธ์ของการนำข้อมูลลงสู่ตาราง (Insert) เห็นโดยทรานแซกชั่นที่ปฏิบัติคำสั่งก่อนหน้า

3.5.2) Repeatable read เป็นระดับที่มีการรับประกันความถูกต้องที่เกือบเท่ากับ Serializable ยกเว้นไม่ได้แก้ปัญหา Phantom phenomenon ซึ่งในระบบคลาวด์เอสคิวแอล ของกูเกิลใช้ Repeatable read เป็นระดับมาตรฐาน

3.5.3) Read committed เป็นระดับที่การรับประกันเฉพาะ Cascadeless schedule ซึ่งก็คือ ทรานแซกชั่นแรกต้อง commit ก่อนทรานแซกชั่นอื่นจะอ่านค่า

3.5.4) Read uncommitted เป็นระดับการจัดลำดับการปฏิบัติคำสั่งซึ่งไม่รับประกัน Cascadeless โดยทั้งสี่ระดับที่กล่าวมาป้องกันไม่ให้เกิดปัญหา เขียนข้อมูลที่ถูกแก้ไขไว้ก่อนแล้วยังไม่ Commit (Dirty write)

คำสั่งที่ 2.9 การตั้งค่าระดับของไอโซเลชั่น

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
  {
      REPEATABLE READ
    | READ COMMITTED
    | READ UNCOMMITTED
    | SERIALIZABLE
  }
```

การตั้งค่าระดับไอโซเลชั่นนั้น ผู้ตั้งค่าจะต้องมีสิทธิในระดับซุปเปอร์ ซึ่งได้แก่ ผู้ดูแลระบบหรือคนที่สิทธิเทียบเท่าเท่านั้น โดยที่การตั้งค่า ระหว่าง GLOBAL และ SESSION มีความแตกต่างกัน คำว่า SESSION หมายถึงการเข้าระบบ กระทำทรานแซกชั่นต่าง ๆ จนกระทั่งออกจากระบบไป ซึ่งการตั้งค่าลักษณะนี้หมายถึง หลังจากการตั้งค่าแล้ว จะมีผลกับทุก ๆ ทรานแซกชั่นรวมถึงทรานแซกชั่นปัจจุบัน จนกว่าจะทำการออกจากระบบโดยมีผลเฉพาะกับผู้ทำการตั้งค่าเท่านั้น ไม่มีผลกระทบต่อผู้ใช้งานคนอื่นในระบบ ส่วนการไม่ระบุการตั้งค่าเป็น SESSION หรือ GLOBAL จะมีผลกับทรานแซกชั่นถัดไปหลังจากตั้งค่าเพียงทรานแซกชั่นเดียว

ถ้าระบุการตั้งค่าเป็นแบบ GLOBAL จะเป็นการตั้งค่าระดับไอโซเลชั่นนี้ให้กับผู้ใช้ทุกคนที่อยู่ในระบบฐานข้อมูล โดยมีผลกับทุกเซสชั่นย่อย ๆ หลังการตั้งค่า โดยที่เซสชั่นที่ยังคงอยู่ในระบบจะไม่มีผลกระทบด้วย แต่หากทำการเข้าระบบใหม่ระดับไอโซเลชั่นจึงจะเปลี่ยน

จากปัญหา The inconsistent anylysis problem และ The phantom phenomenon นั้นจะเห็นได้ว่า การค้นหาข้อมูลในตาราง เกิดจากคำสั่ง SELECT เพียงคำสั่งเดียว แต่อาศัยการใช้ CURSOR ช่วยในการเลื่อนดูการเปลี่ยนแปลงของข้อมูลในแถวระหว่างการดำเนินการในทรานแซกชั่น เช่น การ SUM ค่าของทุกแถว ดังนั้นจะอธิบายเรื่องเกี่ยวกับ CURSOR ในภาษาจาวา และ CURSOR ภายใน MySQL พอสังเขปดังต่อไปนี้

3.6) CURSOR ในภาษาจาวา โดยจากปัญหา The 4 concurrency control problems เราพบว่า ปัญหาบางข้อ ต้องอาศัยการใช้งานของ CURSOR เพื่อใช้ในการตรวจสอบการแก้ปัญหาของ Isolation level ที่ระดับต่างๆ โดยการใช้งานของชนิดของ CURSOR ในภาษาจาวานั้น จะเป็นการเรียกใช้กับ ResultSet ซึ่งเป็นค่าผลลัพธ์ของคำตอบที่ได้จากการ ค้นหาข้อมูลในฐานข้อมูล โดยชนิดของ CURSOR มีดังต่อไปนี้

1) TYPE FORWARD ONLY ค่าคงที่ใช้กำหนดให้ cursor เลื่อนไปข้างหน้าเท่านั้น
2) TYPE SCROLL INSENSITIVE ค่าคงที่ใช้กำหนด cursor สามารถเลื่อนไปมาได้ แต่ไม่เห็นการเปลี่ยนแปลงที่เกิดจากทรานแซกชั่นอื่น
3) TYPE SCROLL SENSITIVE ค่าคงที่ใช้กำหนด cursor สามารถเลื่อนไปมาได้ และสามารถเห็นการเปลี่ยนแปลงโดบทรานแซกชั่นอื่นได้

ในการเรียกใช้ ResultSet ร่วมกับชนิดของ cursor แล้ว ยังมีปัจจัยหนึ่งที่ใช้ในการควบคุม Isolation level เพื่อตรวจสอบการแก้ปัญหาดังกล่าวซึ่งเป็น โหมดของการรักษาความถูกต้อง (Concurrency mode) ให้กับ ผลลัพธ์ของคำตอบเหล่านั้น แบ่งเป็น 2 ชนิด คือ

1) CONCUR READ ONLY ค่าคงที่ ที่ใช้แสดงโหมดของการรักษาความถูกต้องของผลลัพธ์ ที่ยอมให้มีการอ่านค่าเท่านั้น ไม่อนุญาติให้ทำการอัพเดท
2) CONCUR UPDATABLE ค่าคงที่ ที่ใช้แสดงโหมดของการรักษาความถูกต้องของผลลัพธ์ ที่ยอมให้ทำการอัพเดทได้

ตัวอย่างการประกาศชนิด CURSOR และ การระบุระดับความถูกต้องของผลลัพธ์ของคำตอบ

คำสั่งที่ 2.10 การ Query หาข้อมูลโดยใช้ method ของ JAVA

```
Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                        ResultSet.CONCUR_UPDATABLE);
ResultSet rs = stmt.executeQuery("SELECT a, b FROM TABLE2")
```

จากภาพเป็นการประกาศให้ ResultSet มี CURSOR แบบ SCROLL INSENSITIVE และมีระดับความถูกต้องของผลลัพธ์เป็น CONCUR UPDATABLE การทดลองทรานแซกชั่นด้วยCURSOR จาก 4 ปัญหาหลักดังกล่าวนั้น จะพบว่าปัญหาที่จำเป็นต้องใช้ CURSOR เลื่อนดูการเปลี่ยนแปลงของข้อมูล คือปัญหา The inconsistent analysis problem และ ปัญหา Phantom phenomenon เนื่องจาก ใช้การ SELECT เพียงแค่หนึ่งครั้ง แต่ปัญหาเกิดจากช่วงระยะเวลาดังกล่าว มีอีกทราน

แซกชั่น เข้ามาแก้ไขข้อมูล หรือ เพิ่มแถวของข้อมูลใหม่ ทำให้การตรวจสอบดังกล่าวต้องนำ CURSOR มาใช้ตรวจหาการเปลี่ยนแปลงของทรานแซกชั่น

3.7) CURSOR ในระบบฐานข้อมูล CURSOR ถูกใช้งานในคำสั่ง SQL SELECT โดยที่สนับสนุนการทำงานภายใน Stored procedure หรือ ฟังก์ชั่น โดยที่ CURSOR ต้องมีการประกาศและระบุคำสั่งที่ใช้ในการ ค้นหาข้อมูล โดยที่ CURSOR ต้องประกาศในส่วนของ DECLARE ของโปรแกรม และต้องเปิด CURSOR ก่อนการประมวลผลและปิดหลังการประมวลผล คุณสมบัติของ CURSOR มีดังนี้

- Asensitive เซิฟเวอร์อาจทำสำเนาของตารางผลลัพธ์ หรือ ไม่ก็ได้
- Read only อ่านอย่างเดียว อัพเดทไม่ได้
- Nonscrollable อนุญาตให้มีการเดินของ CURSOR ในทิศทางเดียว และไม่สามารถกระโดดข้ามแถวได้

คำสั่งที่ 2.11 ไวยากรณ์ที่ใช้ในการประกาศการใช้งาน CURSOR

```
DECLARE <cursor_name> CURSOR FOR <select_statement>;
```

คำสั่งที่ 2.12 ไวยากรณ์เพื่อใช้เปิด CURSOR

```
OPEN <cursor_name>;
```

คำสั่งที่ 2.13 ไวยากรณ์ที่ใช้เก็บข้อมูลใน CURSOR

```
FETCH <cursor_name> INTO <var1>,<var2>;
```

คำสั่งที่ 2.14 ไวยากรณ์ที่ใช้ปิด CURSOR

```
CLOSE <cursor_name>;
```

จากระดับไอโซเลชั่นทั้ง 4 ระดับเมื่อวิเคราะห์กับปัญหา จะพบว่าแต่ละระดับสามารถแก้ไขปัญหาได้ไม่เท่ากัน ระดับที่มีไอโซเลชั่นสูงที่สุด ระบบจัดการฐานข้อมูลแก้ปัญหาได้มากกว่า ระดับล่างที่ยอมให้ ปัญหาบางตัวยังมีอยู่ Serializable สามารถแก้ปัญหาทั้ง 4 ข้อ, Repeatable read สามารถแก้ปัญหา Lost update, The uncommitted dependency problem และ The inconsistent analysis problem ได้ ,Read committed สามารถแก้ปัญหา The uncommitted dependency problem ได้ แต่ Read uncommitted ไม่สามารถแก้ปัญหาใดๆ ได้

4) Durability ถ้าทรานแซกชั่น commit แล้ว จะต้องอยู่อย่างถาวรแม้ว่าระบบจะล้มเหลวต่อจากช่วงเวลานั้นก็ตาม ซึ่งในซอฟต์แวร์เรียกส่วนการทำงานนี้ว่า ส่วนประกอบการจัดการการกู้คืน (Recovery-management component)

2.3.3 ตัวจัดการชนิดของตาราง (MySQL storage engine)

ตัวจัดการชนิดของตารางต่าง ๆ ซึ่งจัดการทั้งตารางที่เกี่ยวกับทรานแซกชั่นและไม่เกี่ยวข้อง โดยขึ้นอยู่กับรูปแบบการใช้งาน ซึ่งได้แก่ MyISAM, Memory, InnoDB และ Archive โดยมีรูปแบบการใช้งานที่แตกต่างกัน ทั้งนี้โครงงานพิจารณาเน้นเกี่ยวกับ ทรานแซกชั่น จึงต้องทำการเลือก InnoDB engine เป็นตัวจัดการการปฏิบัติคำสั่งบนทรานแซกชั่น โดยในระบบคลาวด์ถูกตั้งค่าไว้เป็นมาตรฐาน

Feature	MyISAM	Memory	InnoDB	Archive
Storage limits	256TB	RAM	64TB	None
Transactions	No	No	Yes	No
Locking granularity	Table	Table	Row	Table
MVCC	No	No	Yes	No
Geospatial data type support	Yes	No	Yes	Yes
Geospatial indexing support	Yes	No	No	No
B-tree indexes	Yes	Yes	Yes	No
Hash indexes	No	Yes	No	No
Full-text search indexes	Yes	No	Yes	No
Clustered indexes	No	No	Yes	No
Data caches	No	N/A	Yes	No
Index caches	Yes	N/A	Yes	No
Compressed data	Yes	No	Yes	Yes
Encrypted data	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No
Replication support	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No
Backup / point-in-time recovery	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes

รูปที่ 2.21 คุณสมบัติของแต่ละ Storage engine

2.4 กลไกควบคุมสภาวะการใช้งานพร้อมกันตามทฤษฎี (Concurrency control)

จากคุณสมบัติพื้นฐานของทรานแซกชั่น เพื่อทำให้เกิดการโต้ตอบระหว่างทรานแซกชั่นที่ทำงานพร้อมกันได้ ระบบต้องมีกลไกที่ใช้ในการควบคุมสภาวะดังกล่าว ซึ่งกลไกที่ใช้ในการควบคุมเรียกว่า Concurrency-control mechanism

กลไกภายใต้ของระบบจัดการฐานข้อมูล จะมีวิธีสร้างความสามารถในการทำงานพร้อมกัน เพื่อให้สามารถดักจับการทำงานที่เกิดขึ้นพร้อมกันได้อย่างถูกต้อง จึงเรียกว่า กลไกพื้นฐานสงวนการเข้าถึงของข้อมูล (Lock-Based Protocols) โดยประกอบไปด้วย 3 ส่วนหลัก

1) คำสั่งที่ใช้ในการล็อค (Lock primitive) ประกอบด้วยหลายโหมดที่ใช้จัดการล็อคบนแถว ซึ่งคำสั่งที่ใช้ในการล็อค จะบ่งชี้ว่าทรานแซกชั่นเราจะทำอะไรได้บ้าง

1.1) Shared-mode lock (lock-S) คือถ้าทรานแซกชั่น สั่ง slock บนแถวใดๆ ทรานแซกชั่นนั้นจะไปอ่านค่าข้อมูลได้

1.2) Exclusive-mode lock (lock-X) คือถ้าทรานแซกชั่นสั่ง xlock บนแถวใดๆ ทรานแซกชั่นจะสามารถอ่านและเขียนค่าในแถวนั้นได้

2) ความเข้ากันได้ของคำสั่งที่ใช้ในการล็อค (Lock compatibility matrix) ในช่วงเวลาที่แต่ละทรานแซกชั่นกำลังปฏิบัติคำสั่งอยู่นั้น ถ้าพิจารณาแถวที่เรากำลังปฏิบัติคำสั่งอยู่ ย่อมมีการจัดการล็อคบนแถวนั้น ซึ่งในทำนองเดียวกันทรานแซกชั่นอื่นย่อมใช้คำสั่งล็อคในการดำเนินการในแถวของตนเอง ซึ่งในทีนี้จะพิจารณาที่แถวเดียวกัน โดยใช้แมทริกซ์ที่ระบุว่า คำสั่งที่ใช้ในการล็อคใดปฏิบัติร่วมกันได้ ถ้าเป็น True หมายถึงสามารถทำงานร่วมกันในแถวนั้นได้ แต่ถ้าเป็น False หมายถึงผู้ที่มาที่หลังต้องรอการปฏิบัติคำสั่งทรานแซกชั่นก่อนหน้า โดยจากการสังเกตจะพบว่า ถ้ามี Xlock ตัวใดตัวหนึ่งย่อมทำให้ตัวที่ตามมาต้องรอเสมอ

ตารางที่ 2.1 Compatibility matrix

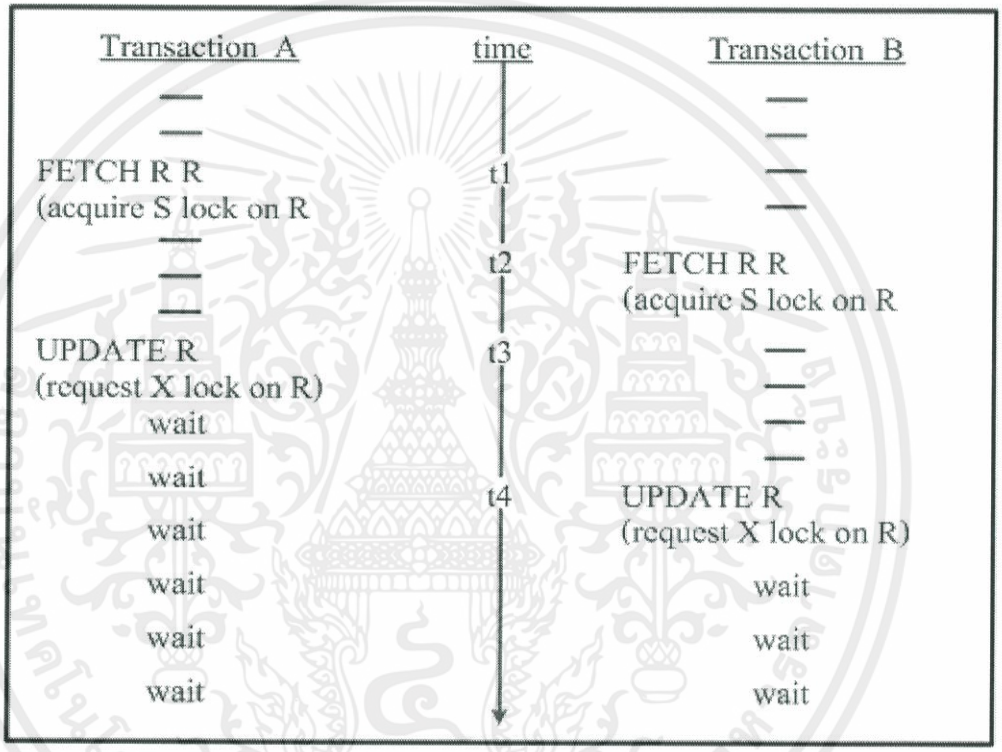
	Slock	Xlock
Slock	True	False
Xlock	False	False

3) โปรโตคอลในการล็อค (Lock protocol) จากสองส่วนที่กล่าวไป ยังไม่เพียงพอต่อการทำให้ผลลัพธ์ของข้อมูลมีความถูกต้อง เนื่องจาก เป็นการล็อคและปลดล็อคโดยไม่มีหลักการที่ถูกต้อง จึงต้องมีโปรโตคอลซึ่งระบุขั้นตอนในการใช้คำสั่ง Lock และ Unlock ซึ่งมีอยู่ในระบบจัดการฐานข้อมูลทุกผลิตภัณฑ์

3.1) The Two-phase locking protocol (2PL) เป็นโปรโตคอลในการล็อคที่รับประกัน Conflict serializable schedule โดยภายในมีหลักการสำคัญภายใน

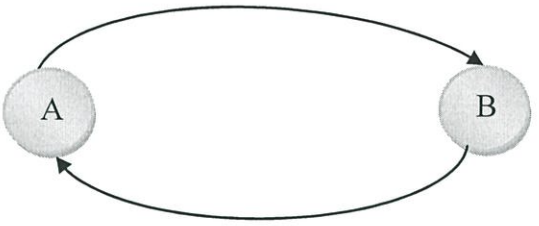
1) Phase 1 (Growing phase) คือ Lock ได้เท่านั้น Unlock ไม่ได้

2) Phase 2 (Shrinking phase) คือ Unlock แล้วจะ Lock อีกไม่ได้



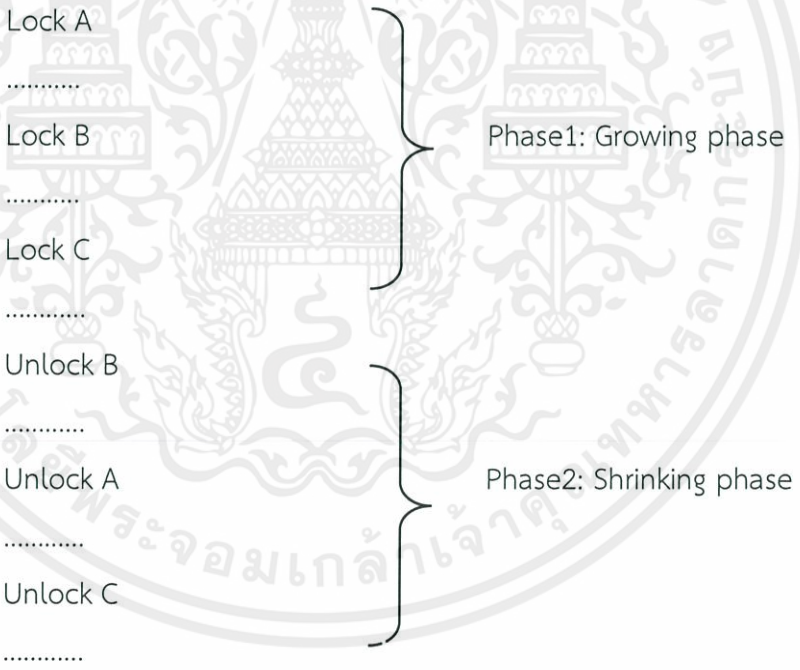
รูปที่ 2.22 ตัวอย่างการแก้ปัญหาด้วย Lock protocol

จากภาพพิจารณาการปฏิบัติคำสั่งทั้งสองทรานแซกชั่น จะพบว่า ที่เวลา t_1 ทรานแซกชั่นแรกทำการอ่านข้อมูล R ทำให้เกิด Slock ที่ R ต่อมาที่ t_2 ทรานแซกชั่นสองทำการอ่านค่า R ทำให้ต้องเกิด Slock โดยเมื่อพิจารณาจาก แมทริกซ์จะพบว่า Slock เจอกับ Slock สามารถทำงานในพร้อมกันได้ โดยไม่เกิดการรอ แต่เมื่อ ทรานแซกชั่น A พยายามจะทำการอัพเดทค่าของ R โดยทำการขอ Xlock ที่R แต่ติด Slock จากทรานแซกชั่นที่สอง ที่ทำการอ่านค่าไว้ ทำให้ทรานแซกชั่น A เกิดการคอย และเมื่อ ทรานแซกชั่นที่สองจะทำการอัพเดทค่า ก็จะติด Slock ของทรานแซกชั่น A โดยการที่สองทรานแซกชั่นใดๆ เกิดการคอยซึ่งกันและกันจะเรียกว่า การเกิดภาวะติดตาย หรือวงจรอับ (Deadlock) ซึ่งช่วยป้องกัน ไม่ให้เกิดค่าที่ผิดพลาดแสดงผลลัพธ์ออกมา ซึ่งก็คือพวกค่าที่ไม่ Conflict serializable จะถูกดักด้วยวงจรอับด้วยวิธีของ The Two-phase locking protocol



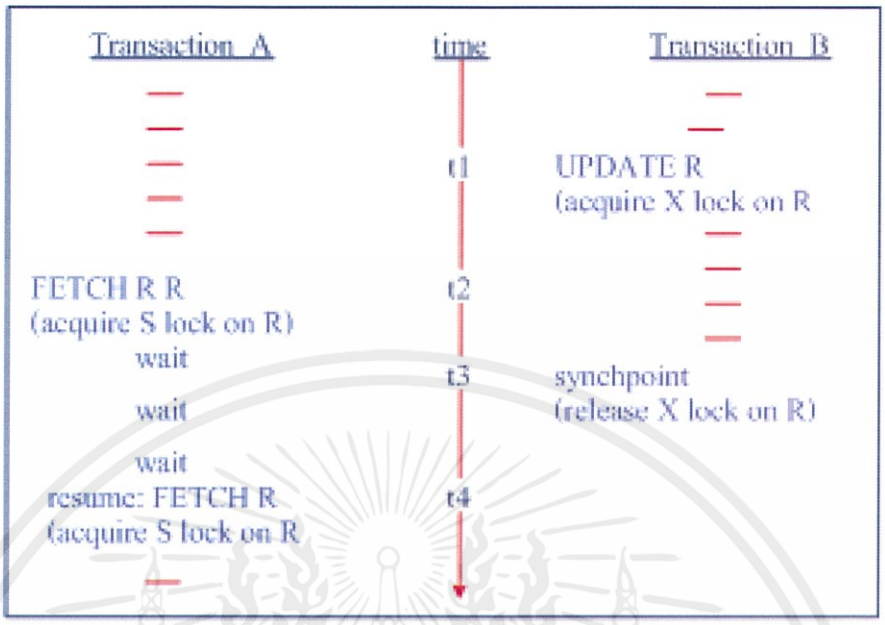
รูปที่ 2.23 Wait for graph

หลักการคือ เริ่มแรกทำการล็อคคำสั่งในแต่ละแถว เมื่อเสร็จการทำงานทุกอย่างแล้วและ ไม่มีการล็อคเพิ่มเติม จึงทำค่อยทำการปลดล็อค โดยที่เมื่อทำการปลดล็อคเพียงครั้งเดียว จะทำการล็อคอีกไม่ได้ แต่อย่างไรก็ตาม ปัญหาหลักทั้งสี่ข้อนั้น Two-phase locking ยังไม่สามารถแก้ไขปัญหา The uncommiited dependency ได้ เพราะปัญหา ไม่สามารถแก้ไขได้ด้วย Conflict serializable schedule ซึ่งปัญหาดังกล่าวเกี่ยวข้องกับ Cascadeless schedule



รูปที่ 2.24 ลำดับการทำโปรโตคอลในการล็อค

เพื่อแก้ปัญหาข้อสอง Xlock จะต้องถูกปลดล็อคที่จุดซิงค์พอยท์ (Strict 2 - phase locking) เนื่องจากของเดิมทำการปลดล็อคเร็วเกินไป ดังนั้นระบบจัดการฐานข้อมูลจะต้องพิจารณาว่า เป็น Slock หรือ Xlock ดังนั้นในระบบจัดการฐานข้อมูลบางผลิตภัณฑ์จะทำการปลดล็อคทั้ง Slock และ Xlock ที่จุดซิงค์พอยท์ เรียกว่า Rigorous 2-phase locking



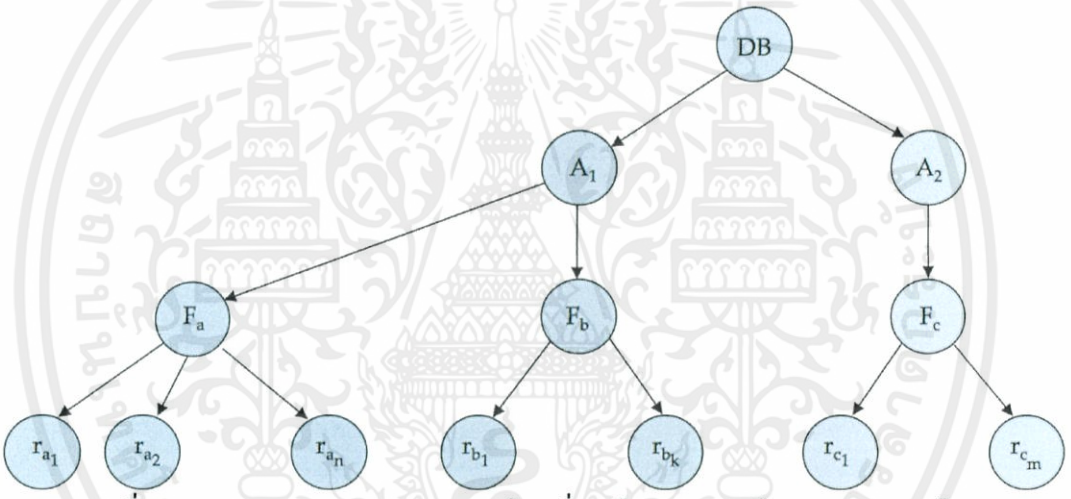
รูปที่ 2.25 การปลดล็อคที่จุดซิงค์พอยท์

ในบางกรณี การล็อคแถวข้อมูล หรือการล็อคตารางที่ละเอียดมากเกินความจำเป็น ทำให้เกิดการรอของทรานแซกชั่นอื่นที่เพิ่มขึ้น ซึ่งทำให้ระบบจัดการฐานข้อมูลบางผลิตภัณฑ์ใช้การล็อคทั้งแบบขนาดเล็กหรือขนาดใหญ่ขึ้นอยู่กับรูปแบบของการทำงาน ซึ่งช่วยในการล็อคให้มีประสิทธิภาพมากขึ้น เรียกว่า Multiple granularity

3.2) Multiple-granularity locking protocol คือความสามารถในการล็อคตามขนาดที่พอสมควร ซึ่งช่วยแก้ปัญหา Phantom phenomenon ที่เกิดจากเพิ่มค่าข้อมูลลงฐานข้อมูล และแก้ปัญหา Inconsistent analysis โดยไม่ก่อให้เกิดวงจรอับ นอกจากนี้ยังสามารถลด overhead ขณะที่ระบบไม่ได้ทำงานหนักได้ โดยปัญหาต่อมาอาจเกิดขึ้นได้ว่า ผู้ใช้จะทราบได้อย่างไรว่า granularity ที่ขนาดเล็กกว่า เช่นระดับแถว ไม่ได้มีทรานแซกชั่นอื่นใช้งานอยู่ ถ้าตัวเองต้องการล็อคระดับตาราง ทั้งนี้ระบบจัดการฐานข้อมูลมีการใช้ Intention lock เข้ามาช่วยในการตรวจสอบความเข้ากันได้ของการประมวลผลระหว่างทรานแซกชั่น Intention-shared (IS) คือ โหนดไหนที่ถูก IS lock Sub-tree ใต้โหนดนั้นจะถูก Slock ถ้าเป็นกรณี Intention-exclusive (IX) คือ โหนดไหนที่ถูก IX lock Sub-tree ใต้โหนดนั้นจะถูก Xlock หรือ Slock และถ้าเป็น Shared and intention-exclusive (SIX) คือ โหนดไหนที่ถูก SIX lock โหนดนั้นรวมถึง Sub-tree ใต้โหนดนั้นจะถูก Xlock

	IS	IX	S	SIX	X
IS	true	true	true	true	false
IX	true	true	false	false	false
S	true	false	true	false	false
SIX	true	false	false	false	false
X	false	false	false	false	false

รูปที่ 2.26 Compatibility matrix



รูปที่ 2.27 Granularity hierarchy โดยที่ A คือ Area F คือ File และ r คือ row

ความแตกต่างของของ MySQL กับทฤษฎีที่ระบุไว้ คือแมทริกซ์ความเข้ากันได้ ซึ่งมีความแตกต่างจากทฤษฎีบางส่วน ซึ่งไม่มีการระบุถึง Shared and intention-exclusive ดังรูปที่ 2.28

	X	*IX*	*S*	*IS*
X	Conflict	Conflict	Conflict	Conflict
IX	Conflict	Compatible	Conflict	Compatible
S	Conflict	Conflict	Compatible	Compatible
IS	Conflict	Compatible	Compatible	Compatible

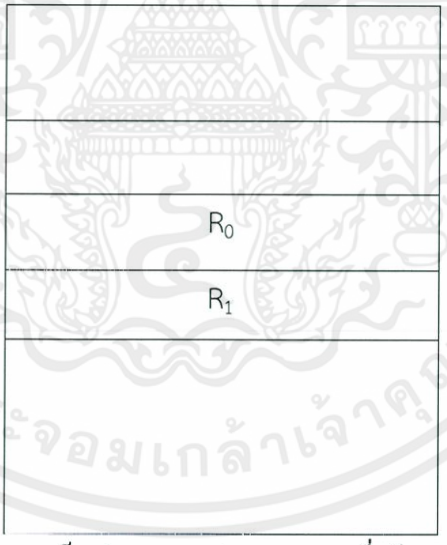
รูปที่ 2.28 Compatibility matrix ของ MySQL

ภายในระบบจัดการฐานข้อมูล นอกจาก พื้นฐานสงวนการเข้าถึงของข้อมูล (Lock-Based Protocols) แล้ว ยังมีกลไกการจัดการอีกแบบหนึ่ง ซึ่งไม่ใช้การล็อคข้อมูล แต่ใช้เรื่องของเวลามาช่วยในการจัดการ เรียกโปรโตคอลแบบนี้ว่า Timestamp-based protocol

2.4.1 Multiversion schemes

จากสภาวะที่เกิดการปฏิบัติทรานแซกชั่นพร้อม ๆ กันนั้น อาจทำให้การทำงานขนาน เกิดความล่าช้าเนื่องจากการรอ หรือ อาจถูกยกเลิกทรานแซกชั่น ยกตัวอย่างเช่น เมื่อมีคำสั่งให้อ่านค่าข้อมูลจากแถวใด ๆ อาจเกิดการรอขึ้นได้ เนื่องจากค่าที่เหมาะสมนั้น ยังไม่ถูกการดำเนินการเขียนให้สำเร็จ หรืออาจเกิดการยกเลิกทรานแซกชั่น เนื่องจากค่าที่ต้องการอ่าน ถูกเขียนทับไปเรียบร้อยแล้ว

กลไกควบคุมการทำ Multiversion คือแต่ละคำสั่งที่ทำการเขียนจะทำการสร้างเวอร์ชั่นของแถวนั้นขึ้นมา และเมื่อมีคำสั่งการอ่านค่าในแถว ภายในระบบจัดการฐานข้อมูลจะทำการเลือกเวอร์ชั่นหนึ่งในหลายเวอร์ชั่นของแถวนั้นเพื่อทำการอ่านค่า โดยที่เวอร์ชั่นที่ทำการอ่านมานั้น ต้องมีความถูกต้อง และสามารถที่จะทำคำสั่งอื่นต่อไปได้



รูปที่ 2.29 การเก็บค่าแบบMultiversion ซึ่งมีหลายเวอร์ชั่น

จากเดิมหากไม่มีการทำ Multiversion แล้วเกิดกรณีการเขียนซ้ำทับค่าเดิมจะทำให้บางทรานแซกชั่นที่ต้องการจะอ่านค่าเดิมไม่สามารถทำงานต่อได้ เช่น ให้ทรานแซกชั่นแรก เป็น T_1 ที่ต้องการอ่านค่า r_0 แต่ทรานแซกชั่น T_2 ทำการอัพเดทค่า r_0เป็น r_1 ซึ่งพอ T_1 ต้องการจะอ่านค่า จึงไม่สามารถอ่านค่าได้ ซึ่งเมื่อเป็น Multiversion T_1 ก็สามารถอ่านค่าจาก r_0 T_2 ทำการอ่านค่าจาก r_1

โดยการอ่านก็จะต้องอ่านให้ถูกเวอร์ชั่น และเมื่อทำการเขียน ก็ให้เขียนเวอร์ชั่นใหม่เสมอ ไม่ทับค่าเดิม ดังรูปที่ 2.24 ซึ่งทำให้ไม่เกิดการรอเกิดขึ้น โดย Multiversion มีสองประเภทคือ Multiversion Timestamp Ordering กับ Multiversion Two-Phase Locking โดยจากการศึกษาเราพบว่า Isolation level ของ **Repeatable read** ของ MySQL นั้นมีการใช้งานของ Multiversion concurrency control ด้วยเช่นกัน

2.5 หลักการควบคุมสภาวะการใช้งานพร้อมกันภายใน MySQL ของระบบคลาวด์

จาก InnoDB engine ซึ่งรองรับการทำงานของทรานแซกชั่นในระบบฐานข้อมูล ซึ่งรองรับ Isolation level โดยแต่ละระดับจะใช้หลักการล็อคที่แตกต่างกัน โดยที่เราสามารถบังคับความถูกต้องได้โดยดีฟอลต์ของ MySQL คือ ระดับ Repeatable read ซึ่งใช้ในการดำเนินการข้อมูลสำคัญต่างๆ ที่เน้นการใช้งานคุณสมบัติของทรานแซกชั่น (ACID properties) โดยเราสามารถลดหย่อนกฎของความถูกต้องได้ด้วยการใช้ระดับที่ต่ำลง เช่น Read committed และ Read uncommitted ในขณะที่ SERIALIZABLE จะเป็นระดับที่มีความเข้มงวดมากกว่า Repeatable read

การดำเนินการภายในระดับต่างๆ ของ Isolation level ใน MySQL

1) Repeatable read เป็นดีฟอลต์ของ InnoDB Engine ซึ่งภายใน innoDB จะเพิ่ม กลไกที่มีประสิทธิภาพมากขึ้น เรียกว่า Multiversion concurrency control (MVCC) ซึ่งมีการใช้งานใน Oracle , PostgreSQL ด้วย โดยเบื้องต้นในการอ่านข้อมูล MVCC จะทำการเก็บ snapshot ของข้อมูลไว้ ซึ่งทรานแซกชั่นจะเห็น Consistent view ของข้อมูล โดยในที่นี้ จะใช้การอ่านแบบ Consistent read ซึ่งจะอธิบายต่อด้านล่าง ซึ่งแตกต่างกับการอ่าน แบบ Locking read ที่ไม่ใช้ Multiversion

    - **Consistent read*** (เช่น SELECT ... FROM...) ในการอ่านในลักษณะดังกล่าวจะมีความแตกต่างกับการอ่านในระดับ Read committed คือ ทุกๆConsistent read ในทรานแซกชั่นเดียวกัน จะอ่านจากsnapshot (ภาพนิ่ง) ที่เกิดจากการ Select ครั้งแรก แม้ในทรานแซกชั่นจะมีหลาย Select ต่อไป ก็จะอ่านที่ค่า snapshot ตัวเดิมเสมอ (ค่า Select ครั้งแรก) หากต้องการได้ค่าใหม่จากการค้นหา ต้องทำการ commit ทรานแซกชั่นปัจจุบันก่อน โดยการอ่านลักษณะนี้ แต่ละทรานแซกชั่นสามารถเห็นข้อมูลที่แตกต่างกันได้ ในตารางที่เหมือนกัน ในเวลาเดียวกัน

*Consistent read หรือ Consistent nonlocking reads เป็นการใช้ Multiversion ในการแสดง Query snapshot (ผลของการค้นหาที่เป็น snapshot ที่อ่านจากฐานข้อมูลในจุดเวลาหนึ่ง) ณ เวลาหนึ่งในการค้นหาข้อมูล จะเห็นข้อมูลที่เกิดจากการเปลี่ยนแปลง ที่เกิดจากทรานแซกชั่นที่ commit ก่อนช่วงเวลาที่ทำการค้นหาแต่จะไม่เห็นการเปลี่ยนแปลงที่เกิดจากทรานแซกชั่นภายหลัง หรือ ทรานแซกชั่นที่ยังไม่ commit ในกรณีคนละทรานแซกชั่น แต่สามารถเห็นการเปลี่ยนแปลงได้ จากคำสั่งก่อนหน้าในทรานแซกชั่นเดียวกัน

- **หากใช้ในส่วนของคำสั่ง การ Locking read (SELECT...for UPDATE, SELECT .... LOCK IN SHARE MODE), UPDATE หรือ DELETE** การล็อคจะขึ้นอยู่กับคำสั่งว่า เป็นคำสั่ง Unique index ด้วยเงื่อนไขแบบ unique search หรือ เป็นคำสั่งที่ใช้เป็นเงื่อนไขแบบ range type search โดยการทำงานจะเป็นตามลำดับดังนี้
  - Unique index with unique search condition InnoDB จะล็อคเฉพาะ index record ที่ตรงตามเงื่อนไข ไม่มีการล็อคช่องว่างที่บนหรือล่าง record แบบนี้สามารถเกิด Phantom ได้
  - Range type search คือลักษณะการค้นหาเป็นช่วง InnoDB จะใช้การล็อคเป็นแบบ next-key locks (รวม index record กับ gap lock)เพื่อที่จะป้องกันการ insert จาก session อื่น ลงในช่องว่างภายในช่วง range นั้นๆ ลักษณะนี้ไม่เกิด Phantom read

2) Read committed ในส่วนของ consistent read ในการ SELECT แต่ละครั้ง จะเป็นอ่านค่า snapshot ใหม่ทุกครั้ง โดยแต่ละครั้งอาจจะได้ค่าแตกต่างกัน

ในส่วนของ คำสั่ง Locking read InnoDB จะล็อคเฉพาะ index record ทั้งหมด ไม่ว่าจะเป็น specific search หรือ range type search ก็ตาม และอนุญาติให้เกิดการ insert ค่าได้ระหว่าง index record แต่ในส่วน UPDATE กับ DELETE จะเหมือนกับ Repeatable read ที่จะขึ้นกับเงื่อนไขว่าเป็น เป็นการใช้ unique index ด้วย unique search (เช่น where id = 100) หรือการค้นหาแบบ range-type (เช่น where id > 100) โดยถ้าหากเป็นแบบ unique search InnoDB จะล็อคเฉพาะ recoed ที่ตรงตามเงื่อนไข ไม่มีการล็อคช่องว่างระหว่างแต่ละ record และในทางกลับกัน การค้นหาแบบ range-type นั้น InnoDB จะใช้ next-key lock ที่จะล็อคทั้ง record และช่องว่างระหว่าง record ด้วย เพื่อป้องกันการเกิด Phantom row

3) Read Uncommitted ในการใช้ Select Statement จะดำเนินการในรูปแบบ nonlocking และ มีโอกาสจะอ่านค่าที่ถูกเปลี่ยนแปลงโดยทรานแซกชั่นอื่น ที่แม้จะยังไม่ commit ก็ตาม ซึ่งเรียกว่า dirty read

4) Serializable ลักษณะการทำงานเหมือนกับ Repeatable Read แต่ InnoDB จะเปลี่ยนการทำคำสั่งของ SELECT...FROM ให้มีการทำงานแบบ SELECT ... LOCK IN SHARE MODE (Slock) ในระดับ Repeatable read ให้โดยอัตโนมัติ

2.5.1) Transaction model and locking

1) InnoDB lock mode มีการใช้งานล็อคโดยหลัก 2 แบบ ในระดับแถว คือ

1.1) Shared lock (S-lock) อนุญาตให้ทรานแซกชั่นอ่านข้อมูลในแถว

1.2) Exclusive lock (X-lock) อนุญาตให้ทรานแซกชั่นทำการupdateหรือdelete แถว

ในกรณีที่ ทรานแซกชั่น T_1 กำลัง Slock อยู่บนแถว R ดังนั้น จะส่งผลให้ T_2 ที่เข้ามาทีหลังซึ่งจะเข้ามา lock แถว r ถูกจัดการ ดังต่อไปนี้

- ถ้า T_2 จะ Slock บนแถว R จะสามารถล็อคได้ทันที ไม่ต้องรอ โดยที่แถว R จะมีทั้ง Slock ของ T_1 และ T_2 ล็อคอยู่
- ถ้า T_2 จะมา Xlock บนแถว R จะไม่สามารถล็อคได้ทันที

ถ้าทรานแซกชั่น T_1 Xlock บนแถว R การร้องขอการล็อคจาก T_2 จะไม่สามารถล็อคได้ ไม่ว่าจะเป็น Slock หรือ Xlock จะไม่สามารถปฏิบัติได้ทันที จำเป็นต้องรอให้ T1 ปล่อยล็อคบนแถว R ก่อน

นอกจากนี้ InnoDB รองรับการล็อคแบบ Multiple granularity locking ซึ่งอนุญาตให้มีการดำเนินการร่วมกันของล็อคที่ระดับแถว หรือ การล็อคทั้งตาราง โดยการล็อคในระดับดังกล่าวในทางปฏิบัติ จะนำ intention lock มาช่วย โดยที่ intention lock คือการล็อคตารางสำหรับทรานแซกชั่น ซึ่งแสดงชนิดของการล็อค(Slock,Xlock) ที่ต้องการต่อไปสำหรับแถวในตารางนั้น โดยแบ่งได้ 2 ชนิด สำหรับ intention lock ใน InnoDB

ถ้า Intention shared (IS) ที่ตาราง Subtree จะเซต Slock แถวที่SELECT มาจากตาราง

ถ้า Intension exclusive (IX) ที่ตาราง Subtreeจะเซต Xlockแถวที่SELECT มาจากตาราง

โดยที่คำสั่งในการทำ IS lock คือ SELECT .... LOCK IN SHARE MODE และ IX lock คือ SELECT .... FOR UPDATE โดยที่เมื่อทรานแซกชั่นต้องการ Slock บนแถวในตาราง T จะต้อง IS lock ที่ ตารางก่อน ในขณะที่เมื่อทรานแซกชั่นต้องการ Xlock จะต้อง IX lock ที่ตารางก่อนเช่นกัน สามารถดูได้ตาราง Compatibility matrix ได้ที่รูป 2.28

2) SELECT ... FOR UPDATE และ SELECT ... LOCK IN SHARED MODE

ในบางสถานการณ์ การอ่านข้อมูลโดยไม่ทำการล็อค อาจทำให้ไม่ได้คุณสมบัติที่ต้องการ จึงมีการใช้วิธีการอ่านข้อมูลแบบล็อคขึ้นมาแทน ซึ่งใน InnoDB engine สนับสนุนการอ่านแบบล็อค 2 แบบ คือ

- **SELECT ... LOCK IN SHARED MODE** เป็นการเซ็ต shared mode lock ลงบนแถวที่อ่านขึ้นมา ซึ่งจะอนุญาติให้ session อื่น สามารถอ่านแถวดังกล่าวขึ้นมาได้ แต่ไม่อนุญาติให้ทำการแก้ไขของแถวดังกล่าวได้ โดยที่แถวดังกล่าวที่อ่านขึ้นมานั้น ต้องเป็นข้อมูลล่าสุดที่ commit แล้ว เพราะฉะนั้นถ้าหาก แถวเหล่านั้นกำลังถูก ทรานแซกชั่นอื่นดำเนินการแก้ไขอยู่ และยังไม่ commit แถวเหล่านั้นจะไม่สามารถอ่านค่าดังกล่าวได้ จนกว่าทรานแซกชั่นนั้นจะสิ้นสุดลง
- **SELECT.... FOR UPDATE** เป็นการป้องกันการที่ทรานแซกชั่นอื่น ใช้คำสั่ง SELECT...LOCK IN SHARED MODE ที่ข้อมูลเดียวกัน ณ ช่วงเวลาเดียวกัน ซึ่งเปรียบเสมือนว่า เป็นการ SELECT ข้อมูลออกมา เพื่อทำการแก้ไขข้อมูลนั้นภายหลัง ดังนั้นทรานแซกชั่นอื่น จึงไม่สามารถปฏิบัติกับแถวดังกล่าวด้วยการ slock หรือXlock ได้

ดังนั้น หากไม่ต้องการล็อคให้ใช้ SELECT...FROM ในลักษณะเดิมซึ่งเป็น consistent read ก็จะสามารถอ่านค่าได้โดยไม่มีการติดล็อค ยกตัวอย่างสถานการณ์ต่อไปนี้ ที่แสดงประโยชน์ของการ locking read (SELECT ..FOR UPDATE / SELECT ... LOCK IN SHARE MODE) ดังนี้

2.1) สมมติให้มีตาราง 2 ตาราง ที่มีความสัมพันธ์กันระหว่างตาราง parent กับ child โดยที่ child ทุกคนจะต้องมี parent โดยที่สมมติให้ กรณีที่เราต้องการทำการ insert แถวลง child table เพื่อกฎบังคับความถูกต้องของฐานข้อมูล เราจึงควรต้องทำการ SELECT ...LOCK IN SHARE MODE ตาราง parent เพื่อทำการตรวจสอบ ว่ามี parent อยู่จริง ถ้าเราทำการ select โดยไม่มีการล็อคแถวนั้น แล้วเราจะสามารถ insert child row ได้อย่างปลอดภัยโดยมั่นใจว่ามี parent อยู่หรือไม่? คำตอบ คือไม่ เนื่องจากก่อนที่จะทำการ insert อาจเป็นไปได้ ที่จะมี session อื่น เข้ามาทำการลบ row ใน parent table ที่เราต้องการ ออกไป ซึ่งจะสถานการณ์ดังกล่าว สามารถแก้ไขปัญหาเหล่านี้ได้โดย ทำการ SELECT แบบ LOCK IN SHARE MODE เพื่อป้องกันไม่ให้เกิดการเปลี่ยนแปลงภายใน row ของ table นั้นๆ เช่น

คำสั่งที่ 2.15 การ Query ข้อมูลโดยมีการ Lock ข้อมูลด้วย

```
SELECT * FROM PARENT WHERE NAME ='JOHN' LOCK IN SHARE MODE
```

ผลจาก share mode lock จะป้องกันไม่ให้ผู้อื่นมาทำการ UPDATE หรือ DELETE แถวดังกล่าวนี้ได้ ซึ่งหมายความว่า เราจะ Insert แถวของ child table ได้อย่างปลอดภัย หากมีการ SELECT ด้วย locking read

2.2) ตัวอย่างถัดไป สมมติให้มีตาราง child_codes ซึ่งมี counter_field เป็น column ที่ใช้ในการระบุ identifier ให้กับแต่ละ child ที่ถูกเพิ่มลงไปยังตาราง child ซึ่ง หากทำการใช้การอ่าน แบบไม่มีการล็อค หรือ ใช้การล็อคแบบ Slock อาจทำให้ผู้ใช้งานสองคนเห็นค่า identifier ล่าสุดที่เหมือนกัน และอาจทำให้เกิดข้อผิดพลาดเนื่องจากเกิด duplicate key ซึ่งในกรณีการใช้คำสั่ง SELECT ... LOCK IN SHARE MODE ก็ยังไม่ดีเท่าที่ควรเช่นกัน ถ้าผู้ใช้ 2 คน อ่านค่า identifier ออกมาพร้อมกัน และทั้ง 2 คนทำการอัพเดทจะส่งผลให้เกิด deadlock ซึ่งใน MySQL ทรานแซกชั่นที่เล็กกว่าจะถูก rollback แต่อีกคนจะสามารถอัพเดทได้ จึงควรใช้ SELECT... FOR UPDATE เพื่อที่จะได้ทำการ ล็อคทั้งตาราง และล็อคค่าที่แถวที่นั้นๆ ตั้งแต่เบื้องต้นเพื่อป้องกัน deadlock

3) InnoDB Record, Gap, และ next-key lock

3.1) Record lock: คือ การล็อคบน index record

3.2) Gap lock คือการล็อคช่องหว่างระหว่าง index record หรือล็อคระหว่างบนสุดหรือล่างสุดของ index record

3.3) Next-key lock คือ เป็นการรวมกันของ record lock และ gap lock

ปัญหา Phantom เกิดขึ้นในทรานแซกชั่นที่ เมื่อ SELECT ข้อมูลที่เหมือนกัน แต่ได้ผลลัพธ์ของแถวแตกต่างกันในช่วงเวลาที่แตกต่างกัน เช่นถ้า ทำการค้นหาข้อมูล 2 ครั้ง จะแสดงแถวในช่วงเวลาที่สอง ที่ไม่ปรากฏในช่วงเวลาแรก และเรียก แถวดังกล่าวว่า phantom row ซึ่งปัญหาดังกล่าวเกิดขึ้นจากการ Insert ข้อมูลใหม่ในช่วงที่ปฏิบัติระหว่าง 2 ทรานแซกชั่นดังกล่าว

โดยหลักการในการป้องกันการเกิด Phantom row จะใช้ next-key lock โดยเป็นการล็อคระดับแถว (row-level locking) เมื่อทำการค้นหา หรือ แสกนแบบ table index โดย InnoDB ภายในจะทำการเซ็ต Slock หรือ Xlock ที่ index record ที่เกี่ยวข้อง และทำการใช้ gap lock ที่ index record นั้น ทำให้หาก session หนึ่งทำการเซ็ต slock หรือ Xlock บน record r แล้ว

session อื่นจะไม่สามารถ insert ค่า index record ระหว่างช่องว่างได้จนกว่าจะเสร็จสิ้นทรานแซกชั่น

ยกตัวอย่าง เพื่อป้องกันการ insert ค่าเข้าไปยังในตารางโดยที่ where id มากกว่า 100 กลไกการล็อคภายใน จะเกิดการล็อคขึ้นโดยใช้ gap lock ซึ่งก็คือ การล็อคทุกแถวที่มีค่า id มากกว่า 100 ถ้าสมมติเดิมมีค่า 90 กับ 102 ดังนั้น InnoDB จะทำการล็อคช่องหว่าง ก่อน 102 และหลังค่า 102 เป็นต้น

2.6 ระบบการกู้คืน (Recovery system)

จากที่กล่าวถึงคุณสมบัติของทรานแซกชั่น ซึ่งได้แก่ Atomicity, Consistency, Isolation และ Durability โดยที่ Durability คือเมื่อทรานแซกชั่นใดทำการปฏิบัติคำสั่งเสร็จเรียบร้อยแล้ว (commit แล้ว) การเปลี่ยนแปลงข้อมูลนั้นจะต้องอยู่อย่างถาวร แม้ว่าในเวลาต่อมาอาจเกิดปัญหาระบบล่ม หรือสาเหตุอื่น ๆ โดยที่ระบบจัดการฐานข้อมูลจะเป็นรับรองการเปลี่ยนแปลงดังกล่าว ทั้งนี้การกู้คืนทรานแซกชั่นกลับคืนมาได้นั้น เป็นเรื่องของคุณสมบัติ Durability กับ Atomicity โดยปัญหาความล้มเหลวที่เกิดขึ้น มีหลายสาเหตุและหลายประเภท

2.6.1 ประเภทของความล้มเหลว (Failure classification)

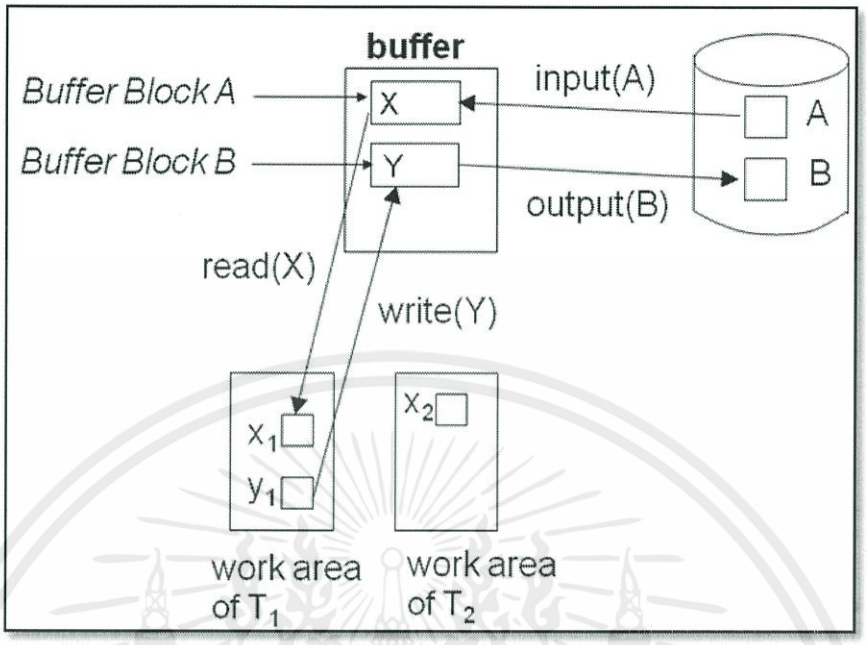
1) Transaction failure คือ เมื่อทรานแซกชั่นหนึ่งล้มเหลว จะไม่กระทบกับทรานแซกชั่นอื่น และ ไม่กระทบเซิฟเวอร์หรือดิสก์

1.1 Logical error ปัญหาที่เกิดจากภายในเช่น โค้ดที่ใช้พัฒนาในแอพพลิเคมีความผิดพลาด เช่น ติด infinite loop

1.2 System error ระบบที่เกิดจากปัญหาภายในระบบฐาน ข้อมูล เช่นการเกิดวงจรอับ หรือไฟล์ในระบบเต็ม เป็นต้น

2) System crash ปัญหาดังกล่าว เช่นการเกิดระบบเซิฟเวอร์ล่ม เมนบอร์ดเสียหาย หรือไฟดับ ทำให้เกิดความล้มเหลวทั้งระบบ โดยสมมติฐานว่า ปัญหาดังกล่าวนี้ที่ไม่กระทบกับดิสก์ เช่น DB Space

3) Disk failure ดิสก์เกิดความเสียหายซึ่งเป็นได้ทั้งกรณี Volatile storage และ Nonvolatile storage กรณีนี้ เฉพาะกรณีที่สามนี้ ระบบจัดการฐานข้อมูลไม่สามารถจัดการเองได้ ต้องมีผู้ดูแลระบบเข้ามาเกี่ยวข้องในการจัดการปัญหา



รูปที่ 2.31 ตัวอย่าง Data Access

ในการเข้าถึงข้อมูลต่าง ๆ นั้นมีกลไกภายในอยู่เบื้องหลัง กล่าวคือ สมมติให้ผู้ใช้งานบนแอพพลิเคชั่น ต้องการอ่านค่า X ซึ่งจะส่งคำสั่งไปให้ระบบจัดการฐานข้อมูลดำเนินการหาผลลัพธ์ให้ โดยก่อนที่ระบบจะทำการอ่านค่า X มาได้จะทำการตรวจสอบดูว่า ค่า X อยู่ที่บัฟเฟอร์หรือไม่ ถ้าไม่มีจึงต้องมีการดึงข้อมูล (Input) จากดิสก์เข้ามายังบัฟเฟอร์ก่อนที่จะแสดงผลที่แอพพลิเคชั่น โดยบัฟเฟอร์ดังกล่าวเป็นบัฟเฟอร์กลางที่ทุกทรานแซกชั่นมีการใช้งานร่วมกันโดยเรียกว่าดาต้าเบสบัฟเฟอร์ (DB buffer) และเรียกดิสก์ดังกล่าวว่าดาต้าเบสสเปซ (DB space)

กรณีต้องการเขียนค่า Y ลงในระบบฐานข้อมูล ระบบจัดการฐานข้อมูลจะทำการตรวจสอบว่าค่าดังกล่าวอยู่ในหน่วยความจำหลักหรือไม่ ถ้าไม่อยู่ ระบบจัดการฐานข้อมูลจึงต้องไปดึงข้อมูลมาจากดิสก์ก่อนเช่นกัน

จะสังเกตเห็นว่า การใช้คำสั่งอ่านหรือเขียนระหว่างโปรแกรมกับบัฟเฟอร์นั้น โปรแกรมจะไม่ได้พิจารณาถึงการโหลดข้อมูลจากบัฟเฟอร์ลงในดิสก์ (Output) เนื่องจากคำสั่งดังกล่าว จะถูกสั่งจากระบบจัดการฐานข้อมูล เมื่อระบบปฏิบัติการและระบบจัดการฐานข้อมูลเห็นสมควร และโปรแกรมจะทำงานกับบัฟเฟอร์โดยไม่เกี่ยวข้องกับดิสก์

2.6.2 Transaction recovery problem statement

เป็นปัญหาที่จำเป็นต้องพิจารณา เนื่องจากหากทำการเข้าถึงข้อมูล ดังรูปData access ไม่เพียงพอต่อการกู้คืนของคำสั่งทรานแซกชั่น ซึ่งก่อให้เกิดปัญหาต่าง ๆ ดังนี้

1) การที่ทรานแซกชั่น commit ไปก่อน แต่การเปลี่ยนแปลงยังไม่ถูกโหลดข้อมูลจากดาต้าเบสบัฟเฟอร์ลงดิสก์

2) การที่ทรานแซกชั่นยังไม่ทำการ commit แต่ได้โหลดข้อมูลลงไปในดิสก์บางส่วนแล้ว โดยมักเกิดจากการดำเนินการทรานแซกชั่น กับข้อมูลที่มีขนาดใหญ่ ที่ต้องมีการ input และ output ข้อมูลตลอดเวลา เนื่องจากหน่วยความจำหลักมีเนื้อที่ไม่เพียงพอ

2.6.3 Log-based recovery

เป็นมาตรฐานหนึ่งที่ ระบบจัดการฐานข้อมูลในปัจจุบันใช้ในการช่วยการกู้คืนของทรานแซกชั่น โดยอาศัย Log files ซึ่งเป็นไฟล์ชั่วคราวซึ่งช่วยในการเก็บรายการเปลี่ยนแปลงที่ปฏิบัติโดยทรานแซกชั่นเอาไว้ โดยรูปแบบการเก็บรายการเก็บคำสั่งใน Log file

```
<Tx, Start>
<Tx, Q, ov, nv>
<Tx, Commit>
```

รูปที่ 2.32 รูปแบบการเก็บ Log file

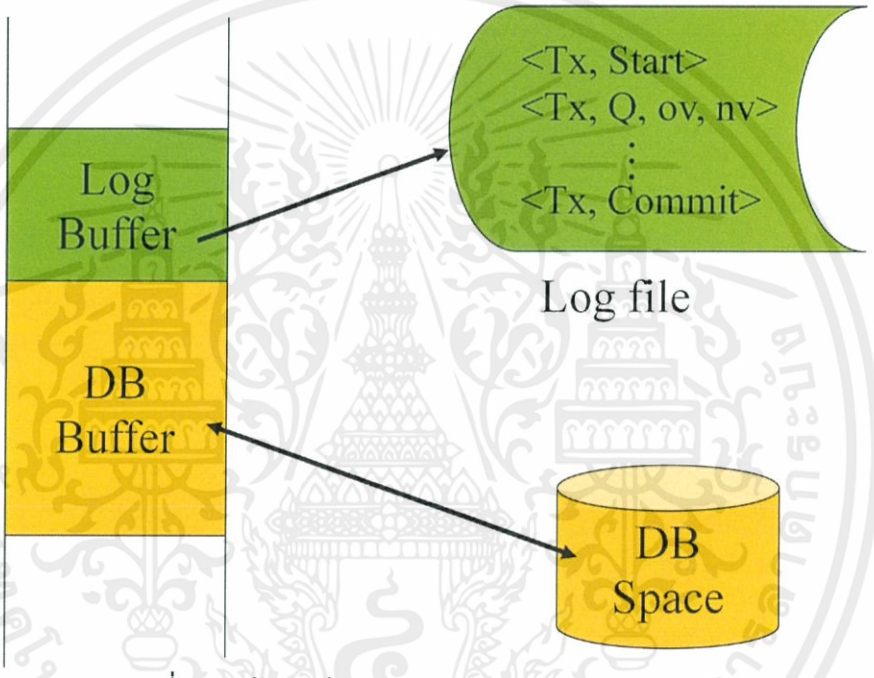
จากรูปที่ 2.32 นั้นแสดงให้เห็นถึง จุดเริ่มต้นของทรานแซกชั่น และประกอบไปด้วยค่า TxID, Q คือ แถวที่ถูกปฏิบัติคำสั่ง, Ov (Old value) คือค่าเดิมของแถวนั้นก่อนที่จะถูกปฏิบัติคำสั่ง และ Nv(New value) คือค่าใหม่ของแถวนั้นหลังถูกปฏิบัติคำสั่งในทรานแซกชั่น และจบคำสั่งทรานแซกชั่นแล้ว

การทำงานของ Log-based recovery มีหลักการทำงานเริ่มต้นเมื่อ เกิดการ commit ของทรานแซกชั่น โดย Log buffer จะทำการเก็บคำสั่ง Log record ซึ่งคือ <Tx , Q , ov , nv> เอาไว้ จนกว่าจะมีการ commit ทรานแซกชั่น , ตอนกำลังจะทำการปิดเครื่อง หรือ บางกรณีจะเก็บจนกว่า Database blocks จะเต็ม จึงค่อยโหลดข้อมูลลงไปใน Log file โดยการเปลี่ยนแปลงอาจอยู่ในบัฟเฟอร์ และยังไม่ทำการลงในดิสก์ โดยหากเกิดการความเสียหายของระบบ เมื่อทำการ รีสตาร์ทระบบปฏิบัติการและระบบจัดการฐานข้อมูลแล้ว ส่วนหนึ่งของระบบจัดการฐานข้อมูล เรียกว่า DBMS recovery manager จะดำเนินการตามขั้นตอนดังต่อไปนี้

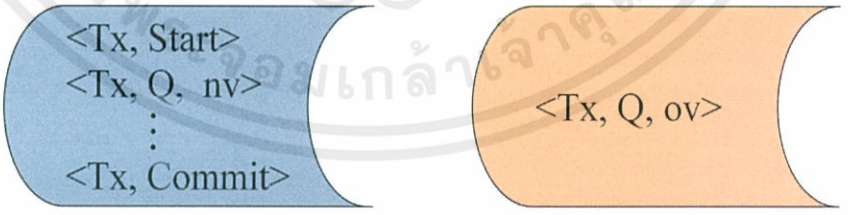
1) ระบบจัดการฐานข้อมูลจะทำการแสกน Log file จากท้ายไล่ขึ้นมาเพื่อตรวจสอบว่าทรานแซกชั่นใด commit และทรานแซกชั่นใดถูกยกเลิก โดยทรานแซกชั่นใดที่ถูก commit แล้วจะนำไปเก็บไว้ใน Redo list แต่ทรานแซกชั่นที่ถูกยกเลิก จะถูกเก็บไว้ใน Undo list ซึ่งในบางผลิตภัณฑ์มีการแยก Log file ออกเป็น Redo log file และ Undo log file

2) สำหรับแต่ละทรานแซกชั่นใน Undo list (ทรานแซกชั่นที่ถูกยกเลิก) ระบบจัดการฐานข้อมูลจะนำ ค่าเก่ามา Undo ดาต้าเบสสเปซ เป็นการแก้ปัญหาทรานแซกชั่นยังไม่ทำการ commit แต่ได้โหลดข้อมูลลงไปในดิสก์บางส่วนแล้ว

3) สำหรับแต่ละทรานแซกชั่นใน Redo list แล้ว (ทรานแซกชั่นที่ commit แล้ว) ระบบจัดการฐานข้อมูล จะนำค่าใหม่ มาทำซ้ำในดาต้าเบสสเปซ เป็นการแก้ปัญหาทรานแซกชั่น commit ไปก่อน แต่การเปลี่ยนแปลงยังไม่ถูกโหลดข้อมูลจากดาต้าเบสบัฟเฟอร์ลงดิสก์



รูปที่ 2.33 โครงสร้างรวมของ Log-based recovery



After Image Journal
(AIJ)
(Redo log file)

Before Image Journal
(BIJ)
(Undo log file)
(Rollback segment)

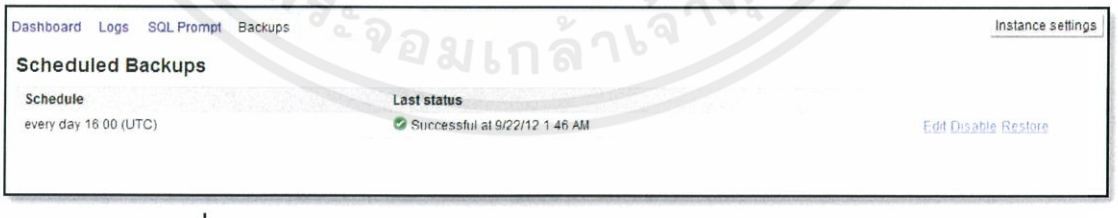
รูปที่ 2.34 Log file ที่แยกเป็น Redo log file และ Undo log file

จากภาพ Log file สามารถแยกไฟล์ระหว่างค่าเก่าและค่าใหม่ได้ ทั้งนี้เพื่อช่วยประหยัดพื้นที่ เนื่องจากถ้าทรานแซกชั่น commit แล้วจะไม่เกิดปัญหาที่ต้องนำเอาค่าเก่ามา Undo ดาต้าเบสสเปซ ทำให้ ไม่จำเป็นต้องเก็บค่าเก่าเอาไว้ ทำให้เมื่อทรานแซกชั่นใด commit แล้วก็สามารถนำ พื้นที่ที่เก็บค่าเก่ามาใช้ต่อได้ โดยการแยกไฟล์ทั้งสอง ขึ้นอยู่กับผลิตภัณฑ์ที่ใช้งาน

2.7 การสำรองข้อมูล (Database Backup)

ในการสำรองข้อมูลที่อยู่ในระบบนั้น โดยทั่วไปมีการทำอยู่สองลักษณะใหญ่ ๆ คือ Volume backup ซึ่งเป็นการสำรองข้อมูลไฟล์ทั้งหมด และ Incremental backup ซึ่งจะเป็นการสำรองข้อมูลเฉพาะที่มีการเปลี่ยนแปลงจากการสำรองข้อมูลครั้งก่อนหน้า โดยในผลิตภัณฑ์ของ MySQL นั้นมีลักษณะการทำงานดังกล่าว โดยมีการสำรองข้อมูลแบบ Incremental backup โดยใช้ Binary Log บนเซิฟเวอร์หลัก และทำการส่ง record ของการเปลี่ยนแปลงของข้อมูลไปยัง เซิฟเวอร์สำรอง เช่น การดำเนินการสร้างตาราง การเปลี่ยนแปลงของข้อมูลในตาราง เพื่อทำให้การเปลี่ยนแปลงถูกทำการเหมือนกับที่เซิฟเวอร์หลัก และใน MySQL enterprise backup สามารถทำ Full backup ได้

ส่วนในระบบคลาวด์ของกูเกิลนั้นได้มีการบริการเกี่ยวกับการสำรองข้อมูลเพื่อให้งานต่อผู้ใช้ โดยมีการสำรองข้อมูลจากศูนย์ข้อมูล (Data center) ไว้มากมายหลายแห่ง กูเกิลได้มีการรับประกันว่าข้อมูลจะไม่เกิดการสูญหาย โดยมีการให้บริการ การสำรองข้อมูลที่เรียกว่า Scheduling backup ซึ่งผู้ใช้สามารถเข้าไปตั้งเวลาเพื่อสำรองข้อมูลทุกวันได้ ว่าต้องการสำรองข้อมูลในทุก ๆ วัน ที่ช่วงเวลาใด ซึ่งทำให้ผู้ใช้งานใน Google Cloud SQL สามารถมั่นใจได้ว่า เมื่อเกิดปัญหาของข้อมูล ผู้ใช้สามารถดึงข้อมูลเก่าที่ทำการสำรองข้อมูลไว้กลับมาใช้ได้ ทำให้สะดวกต่อผู้ใช้งานในระบบคลาวด์มากยิ่งขึ้น



รูปที่ 2.35 ตารางการสำรองข้อมูลแบบScheduing backup ในระบบคลาวด์

2.8 ระบบความปลอดภัยภัยของฐานข้อมูล (Database security)

ในเรื่องความปลอดภัยของระบบฐานข้อมูลนั้น โดยพื้นฐานทั่วไปมีหลายเรื่องด้วยกัน ได้แก่ การสร้างระบบความปลอดภัยของผู้ใช้ , การให้สิทธิต่าง ๆ กับผู้ใช้ตามความเหมาะสม, การกำหนดการมองเห็นตามประเภทของผู้ใช้ นอกจากนี้ เมื่อมีการทำงานเพิ่มขึ้นมาในระบบคลาวด์นั้น

สิ่งหนึ่งที่มีความสำคัญมากอย่างหนึ่ง คือ ความปลอดภัยของข้อมูลของบุคคล โดยกูเกิล มีการดูแลความปลอดภัยในตัวข้อมูลที่ไม่มีการสูญหาย เนื่องจากศูนย์ข้อมูลของกูเกิลมีหลายแห่งทั่วโลก นอกจากนี้กูเกิลยังใช้มาตรฐานความปลอดภัยต่าง ๆ รวมไปถึงการมี Confidentiality หรือความถูกต้องของข้อมูล (integrity) ต่าง ๆ เพื่อความปลอดภัยของข้อมูลผู้ใช้งาน

เมื่อพิจารณาที่ ระบบฐานข้อมูล เราสามารถทำการเพิ่มความปลอดภัยต่าง ๆ ได้ดังนี้

2.8.1 การยืนยันตัวตนผู้เข้าใช้งาน (Authentication)

2.8.1.1 การยืนยันตัวตนผู้เข้าใช้งานด้วยระบบฐานข้อมูล

ในการเข้าใช้งานในระบบนั้น ผู้ใช้งานจะต้องทำการล็อกอินเข้าสู่ระบบ โดยทำให้การเข้าใช้งานเป็นไปด้วยความถูกต้องโดยต้องผ่านการตรวจสอบจากระบบจัดการฐานข้อมูล ไม่ใช่การเข้าสู่ระบบผ่านรหัสของ root ซึ่งฝังในโค้ดของโปรแกรม ดังนั้นเมื่อผู้ใช้ทำการกรอกชื่อและรหัสผ่าน ระบบจัดการฐานข้อมูลจะทำการเข้าไปที่ meta table ของระบบที่มีอยู่ในฐานข้อมูล Mysql ซึ่งภายในจะประกอบด้วย ตารางที่เกี่ยวข้องกับการเข้าถึงระบบฐานข้อมูล ซึ่งถูกให้สิทธิจากผู้ดูแลระบบ เพื่อเช็คสิทธิในการเข้าถึงของผู้ใช้งานนั้นๆ และเรียกวิธีนี้ว่า Native authentication โดยที่ตารางต่างๆ เหล่านี้ได้แก่ User table , db table , host table , tables_priv table , column_priv table, procs_priv table และ proxies_priv table โดยที่แต่ละตารางจะมีสิทธิในการทำคำสั่งและเข้าถึงระบบฐานข้อมูลได้มากน้อยแตกต่างกัน ดังนี้

1) Administrative privileges คือระดับที่อนุญาตให้สามารถทำการดำเนินการกับเซิฟเวอร์ของฐานข้อมูลได้ และสามารถเข้าถึงระบบได้อย่างกว้างขวาง ไม่มีการระบุสิทธิจำกัด ซึ่งได้แก่ผู้ใช้งานที่มีชื่อ ใน user table ที่มีสิทธิในระดับนี้ได้ ซึ่งได้แก่พวก System admin

2) Database privileges คือระดับที่สามารถใช้งานกับฐานข้อมูล และทุก ๆ อ็อบเจ็กต์ใด ๆ ภายในฐานข้อมูลนั้นได้ โดยสิทธิระดับนี้จะถูกกำหนดสิทธิเฉพาะบางฐานข้อมูล หรือ ใช้กับฐานข้อมูลทั้งหมด ขึ้นอยู่กับผู้ดูแลระบบ ซึ่งผู้ใช้งานสิทธินี้ได้แก่ db table

3) Privileges for database objects คือระดับที่สามารถใช้งานในตาราง, คอลัมน์, อินเดกซ์ หรือ วิว ซึ่งจะถูกกำหนดสิทธิให้เฉพาะ บางอ็อบเจ็กต์ในฐานข้อมูล ได้แก่ tables_priv table, column_priv table, proc_priv table เป็นต้น

2.8.1.2 การยืนยันตัวตนผู้เข้าใช้งานด้วยวิธีการอื่น

จากที่ได้กล่าวแล้วว่า โดยปกติแล้วเมื่อผู้ใช้ทำการเชื่อมต่อเข้ามาที่เซิฟเวอร์ เซิฟเวอร์จะใช้ชื่อของผู้ใช้งาน และนำไปตรวจสอบเข้ากับระบบฐานข้อมูลชื่อ mysql ผ่านตาราง user ซึ่งตาราง

ดังกล่าวเป็น metatable ซึ่งถือว่าเป็น native authentication หากไม่มีปลั๊กอิน อื่นช่วยในการยืนยัน

วิธีการดังกล่าวในปัจจุบันที่สามารถช่วยให้ระบบฐานข้อมูลไม่จำเป็นต้องทำการตรวจสอบเอง โดยมีการยืนยันตัวผ่านปลั๊กอินหรือเซิฟเวอร์ ที่รู้จักกันดีในนามของ Authentication server เช่น RADIUS server เป็นต้น แต่ใน MySQL เองนั้น ตั้งแต่เวอร์ชั่น 5.5.7 ได้มีหลายปลั๊กอิน ที่เข้ามาช่วยในการยืนยันตัวตนของผู้ใช้งาน ซึ่งหลักการคือ เมื่อผู้ใช้ทำการล็อกอิน ดาต้าเบสเซิฟเวอร์จะทำการเชื่อมต่อกับ Authentication server และจะทำการรีเทิร์นสถานะสิทธิ์ในการเข้าถึงระบบฐานข้อมูลผู้ใช้งาน โดยการยืนยันตัวตนลักษณะนี้ทำให้เกิดสามารถที่สำคัญ 2 อย่างได้แก่

- External authentication การยืนยันตัวตนด้วยปลั๊กอิน จะทำให้ผู้ใช้งานสามารถเชื่อมต่อ MySQL server อย่างมีหลักฐานรับรอง ซึ่งเป็นวิธีการที่เหมาะสมในการยืนยันตัวตนมากกว่า แบบ Native authentication ที่ใช้ค่าของพาสเวิร์ดที่ถูกเก็บไว้ในตาราง mysql.user โดยปลั๊กอินถูกสร้างเพื่อใช้กับวิธีการยืนตัวตนจากภายนอก ยกตัวอย่าง เช่น PAM (Pluggable Authentication Modules), Windows login IDs, LDAP, หรือ Kerberos
- Proxy users เมื่อผู้ใช้งานได้รับสิทธิ์ในการเชื่อมต่อ ตัว Authentication plugin จะคืนค่าชื่อของผู้ใช้ไปยังเซิฟเวอร์เป็นอีกชื่อของผู้ใช้ที่กำลังเชื่อมต่ออยู่ เพื่อเรื่องของความปลอดภัย จึงเสมือนทำเป็นผู้ใช้งานคนที่สอง ซึ่งสามารถเลียนแบบผู้ใช้งานคนอื่นได้ หรือ กลายเป็นที่รู้จักในฐานะผู้ใช้รายอื่น ซึ่งสามารถตั้งค่าระดับของ Proxy ได้ เพื่อความเหมาะสม

2.8.2 การกำหนดสิทธิในการเข้าถึงระบบฐานข้อมูล (Grant privileges)

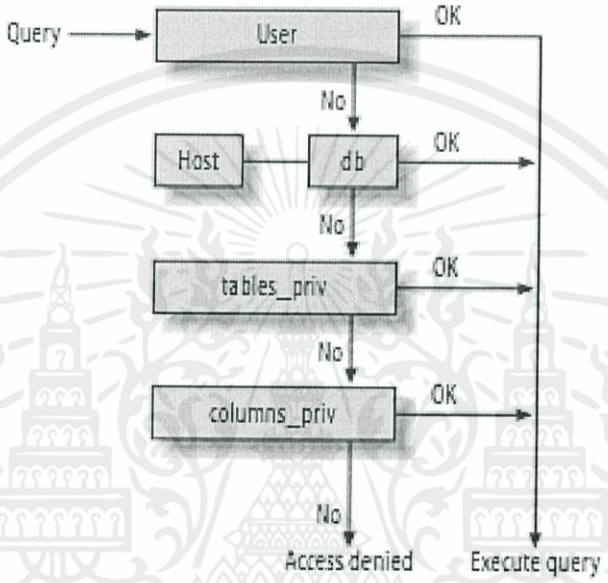
ภาระและหน้าที่ในการกำหนดสิทธิให้กับผู้ใช้แต่ละคนนั้น เป็นเรื่องของผู้ดูแลระบบ ที่ต้องมีการระบุสิทธิของผู้ใช้แต่ละคนตามความเหมาะสม ไม่ควรกำหนดสิทธิที่มากเกินไป หรือน้อยจนเกินไป ทั้งนี้เพื่อความปลอดภัยของระบบฐานข้อมูลเอง โดยการกำหนดสิทธินั้น มีรูปแบบคำสั่งในการให้สิทธิดังต่อไปนี้

คำสั่งที่ 2.17 การให้สิทธิแก่ผู้ใช้งาน

```
GRANT priv_type [(column_list)]
    ON [object_type] priv_level
    TO user_specification [, user_specification] ...
    [REQUIRE {NONE | ssl_option [[AND] ssl_option] ...}]
    [WITH with_option ...]
```

โดยที่ object_type ได้แก่ DB_name,table_name,function และ priv_level ได้แก่ * , *.* , db_name.* , db_name.tbl_name , tbl_name , db_name.routine_name เป็นต้น

ส่วน priv_type นี้เป็นได้หลายสิทธิพร้อมกัน เช่น ALL (กำหนดให้ดำเนินการได้ทุกอย่าง ยกเว้นไม่สามารถให้สิทธิกับคนอื่นได้) , SELECT , UPDATE , DELETE , ALTER , CREATE เป็นต้น



รูปที่ 2.36 MySQL checks privileges

.ในการตรวจสอบว่าผู้ใช้งานที่เข้ามาในระบบมีสิทธิระดับใดนั้น ระบบจัดการฐานข้อมูลจะทำการ ตรวจสอบจาก user table ลงมาเรื่อยๆ จนถึงระดับล่างสุด หากไม่พบจึงจะทำการ Access denied กับ ผู้ใช้งานบุคคลนั้นๆ

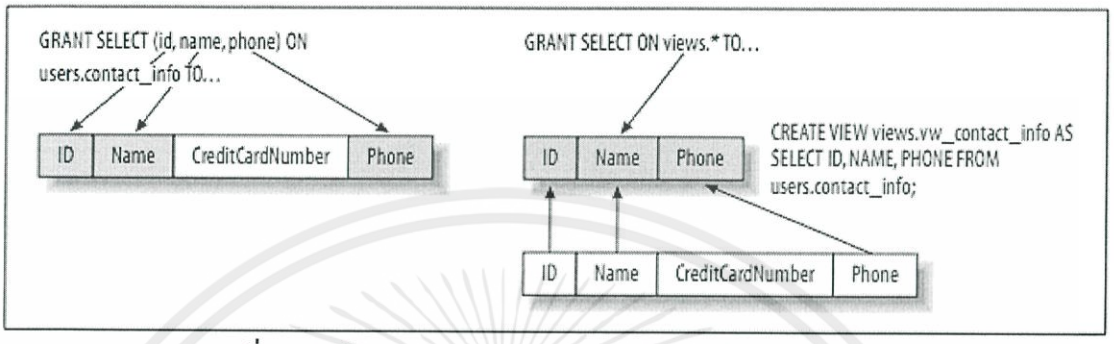
2.8.3 การกำหนดการมองเห็นตามประเภทผู้ใช้งาน (Views)

วิวเป็นตารางเสมือนที่ถูกสร้างขึ้นมาเพื่อป้องกันข้อมูลบางส่วนของตาราง เพื่อให้เห็นส่วนที่จำเป็นของตารางนั้นโดยเป็นไปตามที่ผู้ดูแลระบบเป็นผู้กำหนด ทั้งนี้เพื่อจำกัดการเข้าถึงของข้อมูลผู้ใช้ และให้ผู้ใช้เข้าถึงข้อมูลได้บางส่วนโดยการปรับปรุงแก้ไขในวิวจะมีผลกระทบต่อค่าในฐานข้อมูลวิว

คำสั่งที่ 2.18 การสร้าง view

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

ดังนั้น ประโยชน์ของการใช้งานของตารางเสมือน คือ สามารถนำผลลัพธ์ที่ได้ มาช่วยในการกำหนดสิทธิของผู้ใช้ในระดับ วิว ได้อย่างเหมาะสมมากขึ้น โดยสามารถทำได้ ดังรูปที่ 2.37



รูปที่ 2.37 ตัวอย่างการ GRANT privileges ในระดับ View

วิวนั้นสามารถทำการเปลี่ยนแปลงรายละเอียดของวิวได้ หรือ ทำการลบวิวได้ โดยใช้คำสั่ง ALTER VIEW หรือ DROP VIEW ได้ นอกจากนี้ ในการสร้างวิว สามารถอ้างถึงตารางหรือวิวอื่นได้ และใช้คำสั่งการ JOIN, UNION, SUBQUERIES เช่นดังตัวอย่างที่จะแสดงเกี่ยวกับ การสร้างวิวจากการ JOIN ตารางสองตาราง สมมติให้มีตาราง employee กับ job แสดงดังรูป

คำสั่งที่ 2.19 การสร้างวิวที่เกิดจากการ JOIN สองตาราง

```
CREATE VIEW myView AS
SELECT employee.first_name,job.title
FROM employee join job using (id) ;
```

นอกจากนี้วิวส่วนใหญ่ สามารถทำการอัพเดทได้ และInsert ค่าในตารางได้ ด้วยคำสั่ง UPDATE, DELETE, INSERT โดยวิวที่สามารถอัพเดทได้ จะต้องมีความสัมพันธ์กันแบบ หนึ่งต่อหนึ่ง (One-to-one relationship) ระหว่าง แถวในวิว กับ แถวของตารางนั้น ยกตัวอย่างดังต่อไปนี้

คำสั่งที่ 2.20 ตัวอย่างการอพเดทข้อมูลของวิว

```
UPDATE Oceania SET Population = Population * 1.1 WHERE Name =
'Australia';
```

ส่วนวิวที่ไม่สามารถอัพเดทได้ คือ วิวที่ใช้คำสั่งได้แก่ GROUP BY, UNION, AGGREGATE FUNCTION (SUM( ), MIN( ),MAX( ),COUNT( )) หรือ ข้อจำกัดอื่นๆ บางประการ โดยการค้นหาดังกล่าวซึ่งเปลี่ยนแปลงข้อมูล อาจรวมถึงการ JOIN แต่คอลัมน์ที่จะถูกเปลี่ยนแปลงต้องอยู่ในตารางเดียวกัน

ในส่วนของการ INSERT นั้น วิวที่สามารถทำการอัพเดท จะสามารถ INSERT ได้ ถ้าเป็นไปได้เงื่อนไขสำหรับคอลัมน์วิว ดังต่อไปนี้

- ภายในวิว ต้องไม่มีชื่อคอลัมน์ของวิวที่ซ้ำกัน
- วิวต้องรวมทุกคอลัมน์ของตารางพื้นฐานที่ไม่มีค่า Default
- คอลัมน์ของวิว ต้องเป็นคอลัมน์ที่อ้างอิงง่าย และไม่ใช่ผลลัพธ์ของการคำนวณที่เกิดจากการ Query Expression (Derived column) เช่น Col1+3, Col3/Col4, 3.14159 เป็นต้น

ข้อจำกัดของวิว

1) วิวไม่สามารถทำการสร้าง Index บนวิวได้
2) SUBQUERY ไม่สามารถใช้ได้ ในส่วน FROM ของวิว

บทที่ 3
การออกแบบและพัฒนา

ในบทนี้จะกล่าวถึงโครงสร้างของแอพพลิเคชั่นที่ใช้งานเชื่อมต่อกับระบบจัดการฐานข้อมูลบนคลาวด์ว่ามีลักษณะการทำงานของ Model ต่างๆอย่างไร รวมไปถึงหน้าที่การทำงานของแต่ละหน่วยบนโครงสร้างดังกล่าวทำหน้าที่อย่างไรบ้าง โดยทั้งนี้จะไม่กล่าวถึงรายละเอียดของโค้ดที่อยู่ในแอพพลิเคชั่นแต่จะเน้นในส่วนของรูปแบบการทำงาน รวมทั้งยังมีส่วนของการออกแบบการทดลองตามทฤษฎีที่ได้ศึกษาในหัวข้อเรื่อง การปฏิบัติทรานแซกชั่นแบบขนาน การใช้งานทริกเกอร์

3.1 การออกแบบการทดลอง

ในการออกแบบการทดลองจะอ้างอิงตามทฤษฎีที่ได้ศึกษาเป็นหลักว่าระบบฐานข้อมูลบนคลาวด์มีความสามารถและการทำงานต่างๆตรงตามทฤษฎีที่ได้ศึกษาหรือไม่หรือมีการทำงานที่เหมือนและแตกต่างจากทฤษฎีอย่างไร โดยเน้นพิจารณาที่การประมวลผลการทำงานของทรานแซคชั่นแบบขนานของระบบจัดการฐานข้อมูลที่อยู่บนคลาวด์ และนำมาเปรียบเทียบกับการประมวลผลการทำงานของทรานแซคชั่นแบบขนานของระบบจัดการฐานข้อมูลแบบปกติ รวมไปถึงการการดูแลในส่วนของความปลอดภัยในการเข้าถึงระบบฐานข้อมูลยนคลาวด์

โดยในการออกแบบการทดลองจะแบ่งการทดลองออกเป็นหัวข้อดังต่อไปนี้

1) การนำข้อมูลลงสู่ระบบฐานข้อมูลบนคลาวด์

2) การนำแอพพลิเคชั่นขึ้นบนระบบคลาวด์ของกูเกิล

3) การกำหนดสิทธิการเข้าใช้งานระบบฐานข้อมูลบนคลาวด์

4) การทดสอบปัญหา 4 ข้อของการประมวลผลทรานแซคชั่นแบบขนาน

การทดสอบตามปัญหา 4 ข้อของการทำงานกันแบบขนาน เพื่อทดลองหาข้อแตกต่างของ Isolation level โดยที่ใช้ใช้รูปแบบคำสั่งเช่นเดียวกับบทที่ 2 ซึ่งประกอบด้วย ทรานแซกชั่นที่ดำเนินการในช่วงเวลาเดียวกัน จำนวน 2 ทรานแซกชั่น ซึ่งตามหลักทฤษฎีที่ได้กล่าวข้างต้นนั้น ระดับไอโซเลชั่นยิ่งต่ำ อาจทำให้การดำเนินการทรานแซกชั่น เห็นค่าของการเปลี่ยนแปลงที่เกิดจากอีกทรานแซกชั่นหนึ่งได้ เช่น ระดับ Read uncommitted

5) การทดสอบการตรวจสถานการณ์ทำงานต่างๆของระบบ

6) การทดสอบการใช้งานคำสั่ง SAVEPOINT และ ROLLBACK TO SAVEPOINT

7) การทดลองเพื่อทดสอบเรื่อง CURSOR ใน Java การทดลองเรื่องของการ FETCH CURSOR นี้ เพื่อทดลองปัญหาข้อที่ 3 The inconsistent analysis problem และ ปัญหาข้อที่ 4 The phantom phenomenon ให้ถูกต้องตามหลักทฤษฎี เนื่องจากว่า ปัญหาทั้งสองข้อนี้เป็นปัญหาที่เกิดจากการ SELECT เพียงครั้งเดียวแต่ใช้การเลื่อน CURSOR เพื่อดูค่าแต่ละแถว ซึ่งอาจทำให้เห็นการเปลี่ยนแปลงที่เกิดจากผู้ใช้รายอื่นที่ ทำทรานแซกชั่นในช่วงเวลาเดียวกัน โดยปัจจัยที่เกี่ยวข้องกับการเปลี่ยนแปลง ได้แก่

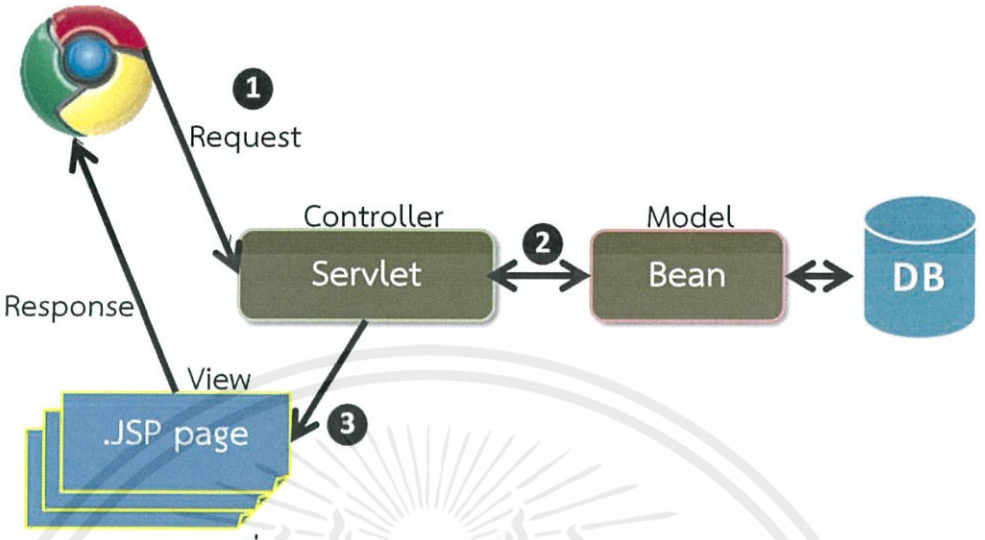
- การตั้งค่า Concurrency mode ซึ่งในภาษาจาวา มีสองโหมด คือ Read only และ Updatable
- การตั้งค่า Isolation level ในการดำเนินการทรานแซกชั่น
- การ Refresh cursor จุดประสงค์เพื่อทำการอัพเดทค่าของแถวปัจจุบันด้วยค่าล่าสุดในระบบฐานข้อมูล

8) การทดลองเรื่องการใช้ TRIGGER จุดประสงค์ในการทำทริกเกอร์ คือ เนื่องจากการกำหนด Integrity constraint ด้วย CHECK condition ยังไม่ได้ implement จริงใน MySQL ทำให้ต้องหาวิธีการกำหนดกฎของโปรแกรมเพื่อความถูกต้องของการใช้งาน เช่น หนังสือทุกเล่ม จะซื้อได้ต้องมีจำนวนไม่น้อยกว่า หรือเท่ากับศูนย์เล่ม หากน้อยกว่าต้องทำการแจ้งออกมาทางโปรแกรม เพื่อให้ผู้ใช้งานทราบ โดยในการทดลองเรื่อง TRIGGER จะเริ่มทำการทดลองที่ MySQL ที่ติดตั้งอยู่ภายในเครื่อง PC ก่อน เพื่อตรวจสอบการเช็คค่าต่างๆได้อย่างถูกต้อง โดยรูปแบบและคำสั่งในการประกาศ เหมือนที่กล่าวไว้แล้วในบทที่ 2

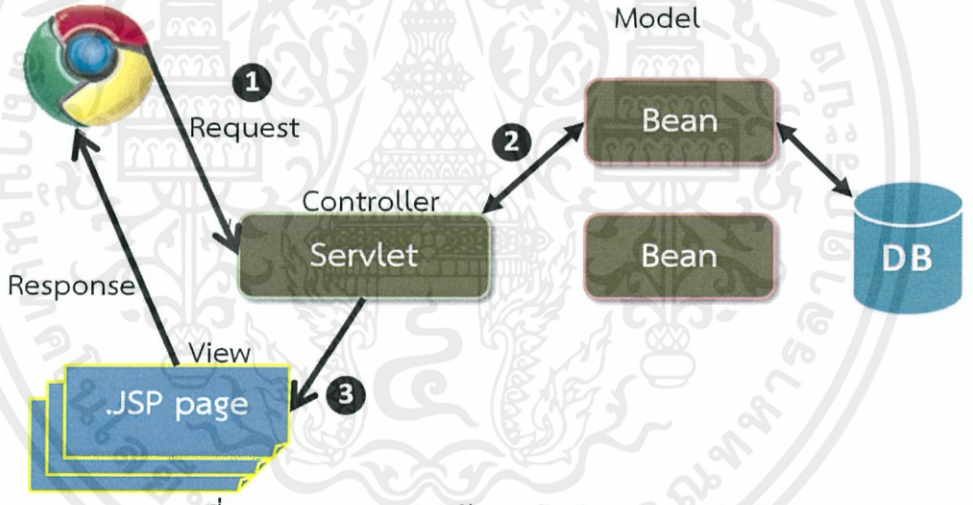
3.2 การออกแบบการพัฒนา Web application

3.2.1 รูปแบบโครงสร้างของแอพพลิเคชั่นที่ใช้ในโครงงาน

โครงสร้างที่ใช้สำหรับการพัฒนาแอพพลิเคชั่นคือลักษณะของ Model-View-Controller (MVC) ซึ่งเป็นสถาปัตยกรรมซอฟต์แวร์ชนิดหนึ่ง ซึ่งในขณะนี้ถือว่าเป็นแบบแผนสถาปัตยกรรม (architectural pattern) ที่ใช้ในสาขาวิศวกรรมซอร์ฟแวร์ รูปแบบ MVC ใช้เพื่อแยกส่วนซอฟต์แวร์ในส่วน ตรรกะเนื้อหา (domain logic) ได้แก่ความเข้าใจในระบบของผู้ใช้ และส่วนการป้อนข้อมูลและแสดงผล (GUI) ซึ่งช่วยให้การพัฒนา การทดสอบ และการดูแลรักษาซอฟต์แวร์ แยกออกจากกัน



รูปที่ 3.1 MVC model กับการติดต่อ MySQL



รูปที่ 3.2 MVC model กับการติดต่อ MySQL (ต่อ)

3.2.2 องค์ประกอบของ MVC Model

องค์ประกอบต่างๆภายใน MVC Model แบ่งออกเป็นสามส่วนคือ Model , View และ Controller โดยหน้าที่การทำงานของแต่ละส่วนภายในแอพพลิเคชั่นจะอธิบายดังต่อไปนี้

1) Model

เป็นส่วนที่ใช้ในการติดต่อกับระบบฐานข้อมูลบนคลาวด์ ยกตัวอย่างเช่น การดูข้อมูลจากฐานข้อมูล , การแก้ไขข้อมูลภายในฐานข้อมูล เป็นต้น

2) View

เป็นส่วนที่เป็นการแสดงผลออกมาให้กับผู้ใช้งานได้เห็น โดยใน MVC Model สามารถมี View ได้หลายแบบขึ้นอยู่กับความเหมาะสมในการติดต่อกับผู้ใช้งาน โดยจะสามารถทำงานทั้งรับข้อมูลจากผู้ใช้งาน และแสดงข้อมูลออกไปให้ผู้ใช้งาน

3) Controller

เป็นส่วนหลักของการทำงานแบบ MVC Model คือเป็นตัวควบคุมการทำงานต่างๆ โดยจะรับข้อมูลมาและคำนวณว่าควรจะไปเรียกใช้ Model ใดและควรจะแสดงออกมาให้ผู้ใช้งานได้รับรู้ด้วย View แบบใด ยกตัวอย่างเช่น ผู้ใช้งานต้องการดึงข้อมูลออกจากตารางในฐานข้อมูล โดยการระบุข้อมูลที่ต้องการลงในเว็บฟอร์ม (Web Form) ซึ่งถือเป็น View แบบหนึ่ง ข้อมูลดังกล่าวจะถูกส่งต่อจาก View ไปสู่ Controller และ Controller จะทำการคำนวณว่าลักษณะข้อมูลที่เข้ามาควรจะเรียกการทำงานของ Model ใดเพื่อที่จะได้ผลลัพธ์ตามที่ต้องการ เป็นต้น

3.2.3 การใช้ทรานแซกชั่น บน Web application ที่ 1 (online Bookstore)

ISOLATION LEVEL : ระดับไอโซเลชั่นที่ใช้ในเว็บแอพพลิเคชั่นร้านขายหนังสือ จะใช้เป็น Repeatable read ซึ่งเป็นดีฟอลต์ของระบบจัดการฐานข้อมูลบนคลาวด์อยู่แล้ว เนื่องจาก ที่ได้กล่าวไปแล้วว่า Repeatable read ของ MySQL มีลักษณะของการใช้งานแบบ Multiversion ที่ไม่รบกวนการทำงานซึ่งกันและกัน ในขณะที่ Serializable จะทำการล็อคข้อมูลที่ค้นหาขึ้นมาเป็นส่วนใหญ่ ซึ่งอาจทำให้เกิดการรอที่ยาวนานได้ และข้อดีของ Multiversion อีกอย่างคือ ทรานแซกชั่นอื่นสามารถ insert แถวข้อมูลใหม่ได้ โดยไม่เป็น phantom row (ทรานแซกชั่นอื่น insert ลง แต่ทรานแซกชั่นแรกจะไม่เห็นจนกว่าทรานแซกชั่นแรกจะ COMMIT) ดังนั้นจะใช้ในการ SELECT ลูกค้าสามารถดูข้อมูลของหนังสือได้จริง โดยไม่มีการล็อคระหว่างที่กำลังทำการเลือกดูหนังสือ

การใช้ทรานแซกชั่นใน Web application ร้านหนังสือโดยหลัก มีดังนี้

1) การเพิ่มหนังสือลงในระบบ

เพื่อป้องกันการ INSERT ISBN เดียวกัน จากผู้ใช้งานในระบบ 2 คนในช่วงระยะเวลาเดียวกัน จะใช้การ SELECT .. FOR UPDATE ตารางก่อน ทำให้ตารางทั้งหมดใน ช่วงเวลานั้นถูก IX lock และ ทุกแถวของตารางถูก Xlock ซึ่งจะป้องกันไม่ให้เกิดการ insert ที่ซ้อนทับกัน ซึ่งอาจก็ให้เกิดรอ หรือ deadlock ได้

2) การแก้ไขข้อมูลต่างๆ ของหนังสือที่มีอยู่

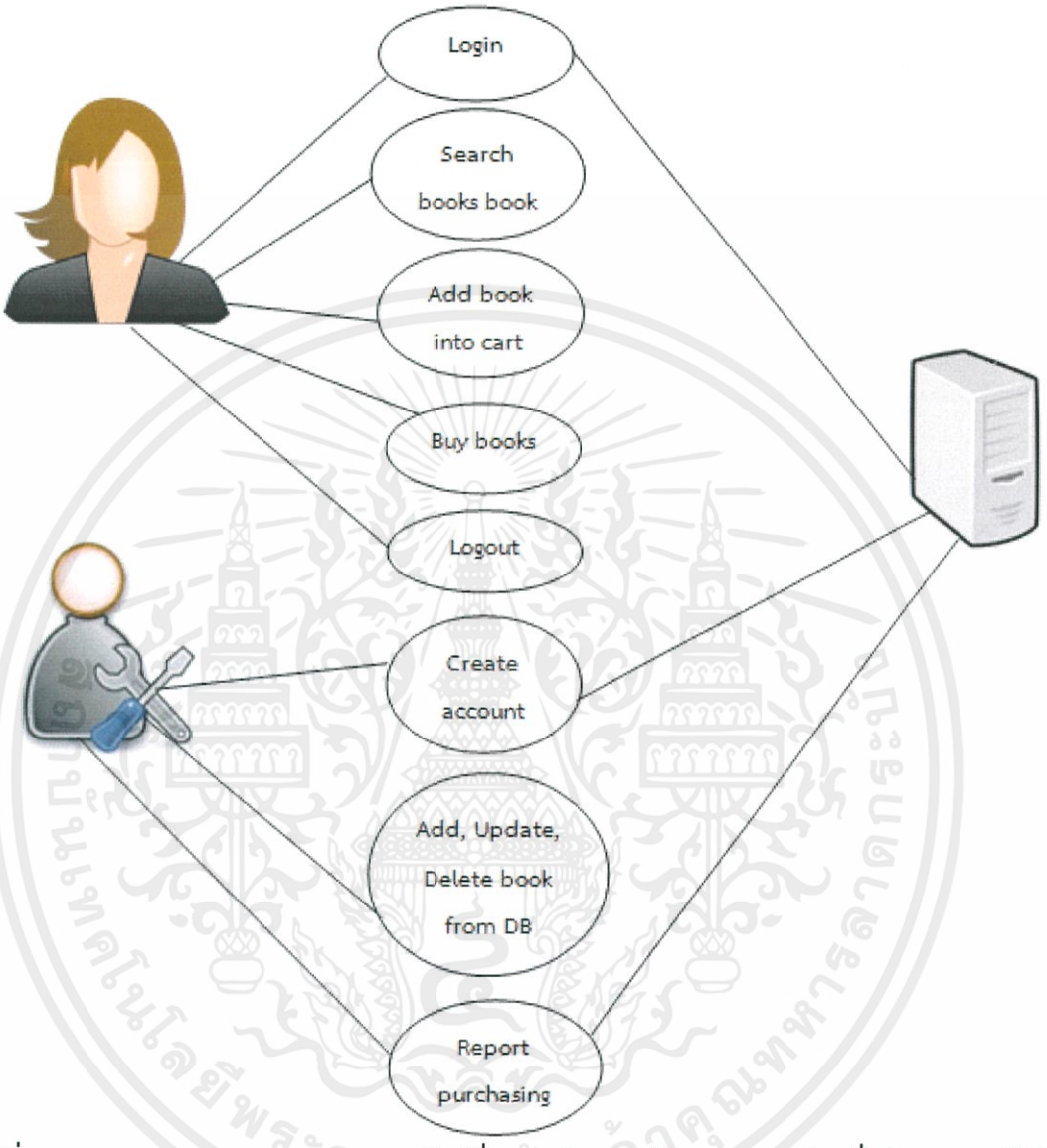
เพื่อทำการ UPDATE รายละเอียดของข้อมูลในบางแถวหรือคอลัมน์ภายในตาราง จะใช้การ IS lock ที่ตารางเหล่านั้นขึ้นมาก่อน เพื่อทำการดูว่า ISBN ของหนังสือเล่มที่ต้องการแก้ไขมีอยู่จริง และยังไม่ได้แก้ไขข้อมูลดังกล่าว ดังนั้นจะใช้ SELECT .. LOCK IN SHARE MODE ก่อน

หลังจากนั้น จะทำการ IX lock ที่ตาราง (SELECT...FOR UPDATE) เพื่อเริ่มทำการแก้ไขข้อมูล ในแถวที่ต้องการ ทั้งนี้ ที่ทำการ IS lock ก่อน ไม่ IX เลย เพื่อป้องกันกรณีหาก ผู้ใช้งานคนอื่นเข้ามาในขณะ IX lock จะได้ติด wait อีกด้วยเช่นกัน ในจุดเวลานั้น

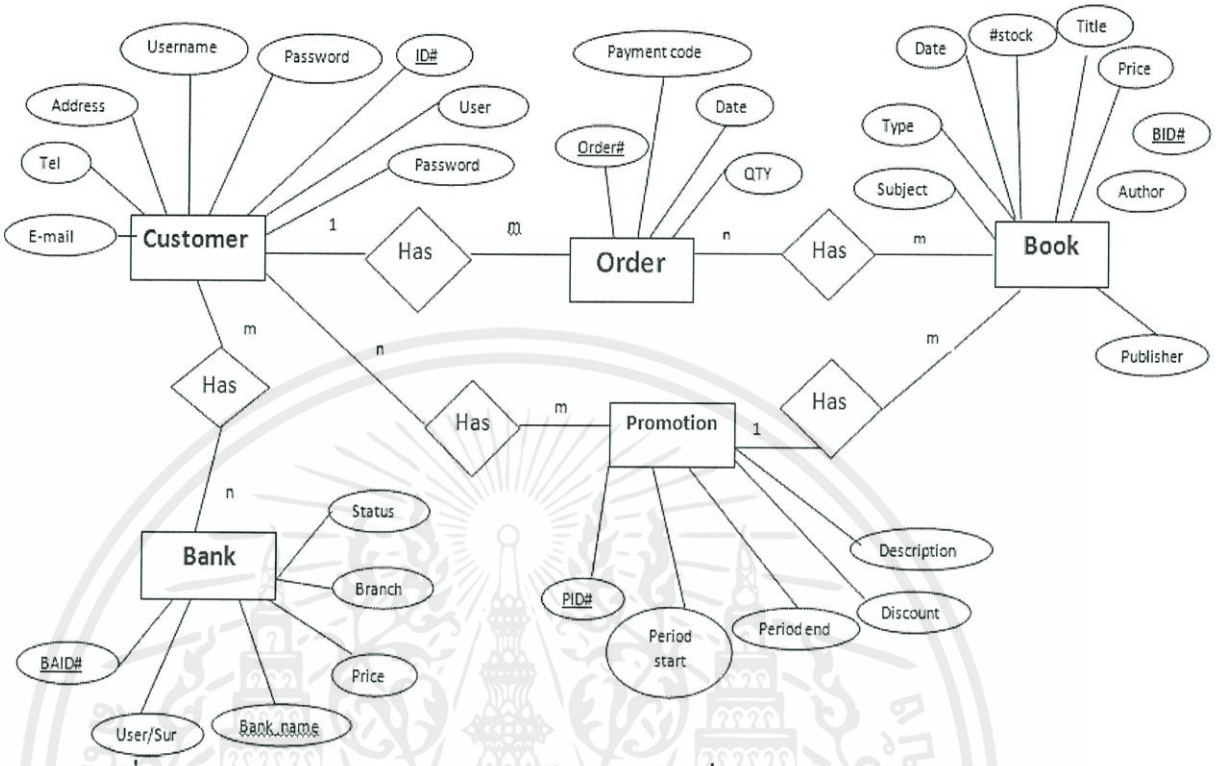
3) ความถูกต้องขณะการแก้ไขจ่ายเงินหลังเลือกซื้อหนังสือ

ในขณะการเลือกซื้อสินค้าเข้า CART จะอนุญาติให้ลูกค้าทุกคน สามารถหยิบของที่มีอยู่ใน catalog ได้ แม้ว่าจะมีอยู่เพียงชิ้นเดียวก็ตาม แต่เมื่อเข้าสู่หน้าของการจ่ายเงิน (มีการเช็คว่าล็อคอินแล้วหรือไม่ จาก session) จะมีหน้าสรุปรายการสินค้า ภายในส่วนนี้จะมีการทำ SELECT .... FOR UPDATE โดยภายใน ระบบฐานข้อมูลได้มีการตั้งค่าที่คอลัมน์ StockNum เพื่อตรวจสอบจำนวนของหนังสือก่อนการอัพเดทค่า ผ่าน Trigger เพื่อนำจำนวนหนังสือที่มีในตาราง กับ จำนวนที่ลูกค้าต้องการว่าเพียงพอ และสามารถซื้อได้ ถ้าไม่สามารถสั่งซื้อได้จะทำการ แสดง ERROR MESSAGE ให้ลูกค้า แต่ถ้ามีจำนวนพอที่จะสั่งซื้อได้หลังจาก กดชำระสินค้า จะทำอัพเดทข้อมูล ซึ่งก็คือการ IX lock ที่ตาราง เพื่อ Xlock ที่แถวที่ต้องการเพื่อ UPDATE จำนวนที่คงเหลือล่าสุด (หักจำนวนซื้อล่าสุด) หลังจากนั้นจะทำการ insert ข้อมูลใหม่ ที่ลูกค้าสั่งซื้อลงในตาราง order เพื่อใช้ในการเก็บรายละเอียด ตรวจสอบ และจัดส่งต่อไป และ insert ลงไปที่ตารางความสัมพันธ์ระหว่าง Order และ เมื่อทำรายการเสร็จสิ้น จึงแสดงหน้ารายละเอียดของใบเสร็จรับเงิน ต่อไป โดยจะทำการ Join สองตารางที่ insert ค่าเข้าไป และ สร้าง VIEW แสดงรายการซื้อของลูกค้า โดยในการกำหนดสิทธิ์ต่างๆ ของสมาชิก มีดังต่อไปนี้

- สิทธิ์ในการ SELECT ข้อมูลทั้งหมดในตาราง books
- สิทธิ์ในการ INSERT ข้อมูลในตาราง orders ได้
- สิทธิ์ในการ UPDATE StockNum ในตาราง books ได้
- สิทธิ์ในการ SELECT ข้อมูลใน VIEW ของสมาชิกคนนั้น ได้



รูปที่ 3.3 USE CASE DIAGRAM แสดงฟังก์ชั่นการทำงานหลักของแอพพลิเคชั่นร้านขายหนังสือ



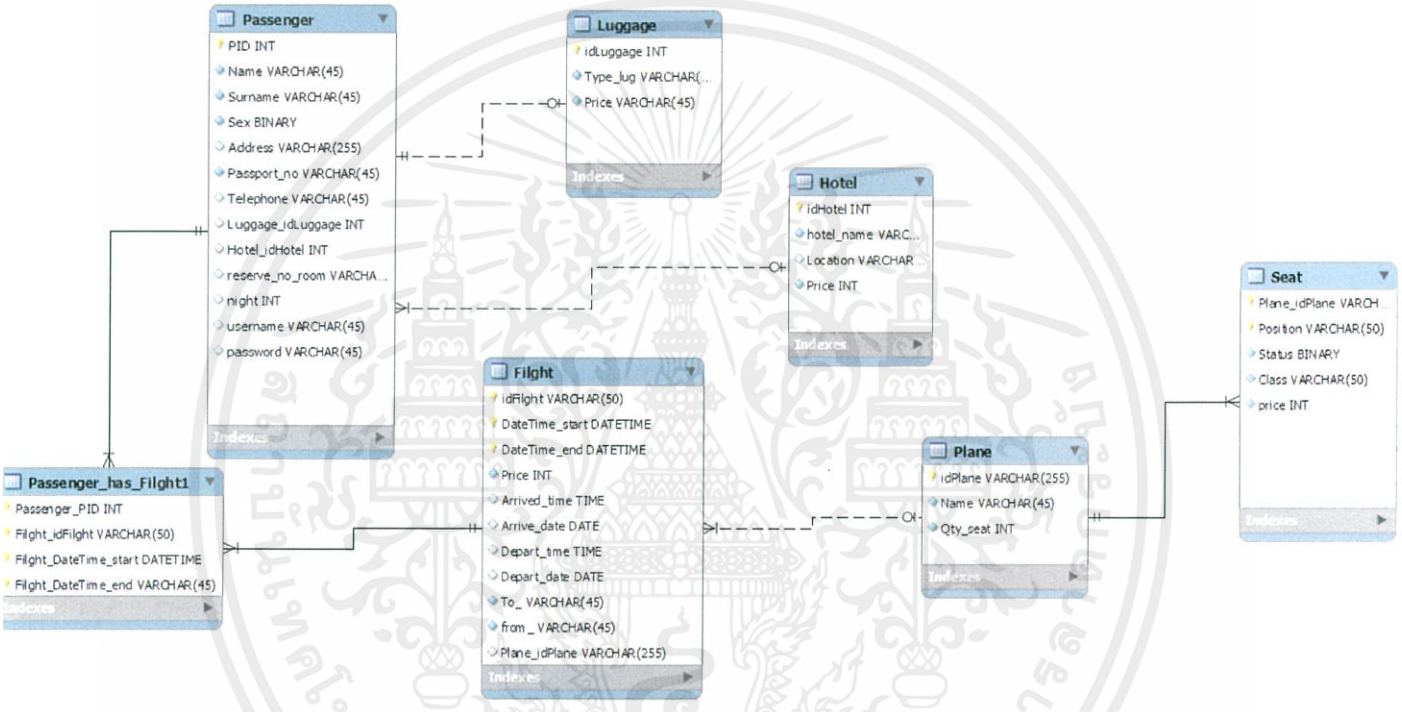
รูปที่ 3.4 ภาพแสดง ER Diagram ของเว็บแอพพลิเคชั่นร้านขายหนังสือออนไลน์

3.2.4 การใช้ทรานแซกชั่น บน Web application ที่ 2 (Airplane reservation)

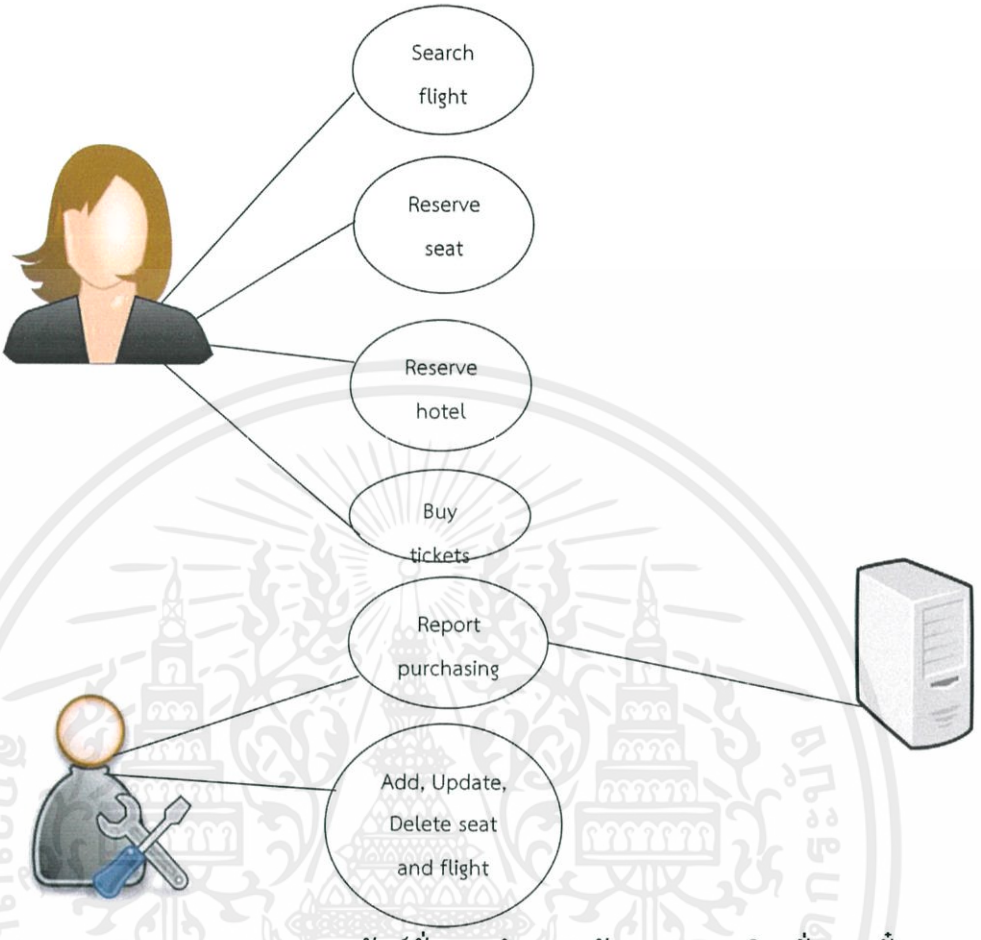
ISOLATION LEVEL : ระดับไอโซเลชั่นที่ใช้ในเว็บแอพพลิเคชั่นการจองตั๋วเครื่องบินออนไลน์ โดยระดับของไอโซเลชั่นที่จะใช้กับระบบฐานข้อมูลกูเกิลคลาวด์ จะใช้เป็น Repeatable read เหมือนแอพพลิเคชั่นแรก เบื้องต้นแล้วจะเห็นว่าแอพพลิเคชั่นที่สองนี้ ต้องเน้นความถูกต้องมากกว่าแอพพลิเคชั่นแรก เพราะที่นั่งแต่ละที่จะมีเพียงแค่หนึ่งเดียว ถ้าหากการกระทำทรานแซกชั่นเกิดความผิดพลาด จะให้เกิดความเสียหายเป็นอย่างมาก แต่สาเหตุที่ไม่ใช้ Serializable เพราะ Repeatable read สามารถแก้ปัญหาได้ทั้ง 4 ข้อ โดยไม่ทำการ Lock resources ที่ตนเองจับจองอยู่ ณ ช่วงเวลานั้น ซึ่งทำให้คนอื่นต้องเกิดการรอเกิดขึ้นทั้งที่อาจไม่ได้เป็นการทำงานที่ตำแหน่งแถวเดียวกัน ทำให้อาจเสียเรื่อง Performance ได้มากกว่า ดังนั้นจะใช้การ Consistent read เพื่อแสดงตำแหน่งที่นั่ง ที่ว่างและไม่ว่างก่อน และนำมาแสดงผลที่ Web application นอกจากนี้ยังมีการแสดงเที่ยวบินของแต่ละวัน ซึ่งเป็นเที่ยวบินไปและเที่ยวบินขากลับที่ผู้ใช้งานทำการเลือก มีอัตราค่าโดยสารที่แตกต่างกันขึ้นกับช่วงเวลาการจอง และค่าต่างๆ ที่จำเป็นในการทำทรานแซกชั่นต่อไป ที่ไม่จำเป็นต้องทำทรานแซกชั่น ซึ่งไม่มีผลกระทบหากใช้งานพร้อมๆกันจากผู้ใช้หลายคน จนกระทั่งเมื่อได้ค่าครบจึงเริ่มทำทรานแซกชั่น โดยใช้ระดับไอโซโลชั่นเป็น Repeatable read

การใช้ทรานแซกชั่นใน Web application การจองตั๋วเครื่องบินออนไลน์หลัก มีดังนี้

การอัพเดทสถานะของที่นั่ง จุดประสงค์หลักในการทำทรานแซกชั่นนี้ คือการล็อกที่นั่งที่ผู้จองทำการเลือกเอาไว้ โดยจะเปลี่ยนค่า Status จากศูนย์เป็นหนึ่ง เท่ากับจำนวนที่นั่งที่จองไว้ และภายในทรานแซกชั่นจะทำการ Insert ค่าลงใน Table อื่นๆ เพื่อเพิ่มเรื่อง กระเป๋าเดินทางของผู้จองแต่ละคน ข้อมูลของผู้จอง และตารางความสัมพันธ์ระหว่าง ผู้จองกับเที่ยวบิน เมื่อทำการ Insert และอัพเดทแล้ว จะทำการ COMMIT ทรานแซกชั่น ซึ่งเป็นอันจบกระบวนการของทรานแซกชั่น



รูปที่ 3.5 ภาพแสดง ER Diagram ของเว็บแอพพลิเคชั่นการจองตั๋วเครื่องบินออนไลน์



รูปที่ 3.6 USE CASE DIAGRAM แสดงฟังก์ชั่นการทำงานหลักของแอพพลิเคชั่นจองตั๋วสายการบิน

บทที่ 4
การทดลองและผลการทดลอง

การทดลองและผลการทดลองจะถูกนำมาใช้ในการพิจารณาความแตกต่างระหว่าง MySQL ที่ติดตั้งและทำงานแบบแสตนด์อโลน (Stand alone) และ MySQL ที่ทำงานอยู่บนกูเกิลคลาวด์อินฟราสตรัคเจอร์ (Google Cloud Infrastructure) โดยจากที่กล่าวไปแล้วในบทที่ 2 (ทฤษฎีที่เกี่ยวข้อง) พบว่า Google Cloud SQL มีรูปแบบการทำงานและลักษณะเฉพาะคล้ายคลึงกับ MySQL แบบแสตนด์อโลน แต่มีบางรูปแบบที่ไม่สามารถใช้งานได้ เช่น การทำงานของคำสั่ง LOAD DATA INFILE, SELECT ... INTO OUTFILE/DUMPFILE, INSTALL/UNINSTALL PLUGIN, CREATE FUNCTION, LOAD_FILE( ) เป็นต้น

ทั้งนี้ รวมไปถึงการทดลองที่เกี่ยวข้องกับโครงงานโดยเริ่มที่ส่วนของการนำข้อมูลลงสู่ระบบฐานข้อมูลบนคลาวด์ การทดลองการอัพโหลดแอพพลิเคชั่นที่พัฒนาบนเครื่องของผู้พัฒนาลงสู่กูเกิลแอพเอนจิน (Google App Engine) ที่ทำงานอยู่บนคลาวด์ ซึ่งมีหลักการของการ Export และ import file ในส่วนการทดลองเกี่ยวกับตัวผลิตภัณฑ์ Google Cloud SQL จะเจาะจงหัวข้อของการทดลองเช่น การกำหนดสิทธิในการเข้าใช้งานระบบฐานข้อมูลของผู้ใช้งาน, ปัญหา 4 ข้อของการประมวลผลทรานแซคชั่นแบบขนาน, การเปรียบ เทียบความสามารถในการตั้งค่าของผลิตภัณฑ์แบบแสตนด์อโลนและผลิตภัณฑ์บนคลาวด์ และเครื่องมือที่ใช้ในการตรวจสอบสถานะการทำงานของผลิตภัณฑ์

ทั้งนี้การทดลองตามขอบเขตดังกล่าวเป็นเชิงทดสอบความสามารถการทำงานของระบบจัดการฐานข้อมูลบนคลาวด์ว่ามีความสามารถรวมไปถึงรูปแบบในการทำงานต่าง ๆ เหมือนการทำงานของระบบจัดการฐานข้อมูลที่ติดตั้งอยู่บนเครื่องของผู้ใช้งาน (ในที่นี้จะเรียกว่าระบบจัดการฐานข้อมูลแบบปกติ) หรือไม่อย่างไร เพราะฉะนั้นการทดลองในส่วนใหญ่จะเป็นการทดลองการทำงานที่ระบบจัดการฐานข้อมูลแบบปกติสามารถทำงานได้อยู่แล้วเพื่อตรวจสอบว่าระบบฐานข้อมูลบนคลาวด์สามารถทำงานได้หรือไม่ หรือมีข้อจำกัดการทำงานหรือไม่อย่างไร โดยการทดลองจะระบุหัวข้อที่ทำการทดลองทั้งหมด 6 หัวข้อดังต่อไปนี้

4.1 การทดลองการจัดการฐานข้อมูลบนคลาวด์เพื่อให้สอดคล้องกับทฤษฎี

4.1.1 การนำข้อมูลลงสู่ระบบฐานข้อมูลบนคลาวด์

โดยข้อมูลที่จะนำลงสู่ระบบฐานข้อมูลบนคลาวด์นั้นเป็นข้อมูลที่มีนามสกุล CSV ที่ย่อมาจาก Comma-Separated Value (Variable) ซึ่งคือไฟล์สำหรับเก็บข้อมูลแบบตาราง โดยใช้จุลภาค (เครื่องหมายลูกน้ำ) แบ่งข้อมูลในแต่ละคอลัมน์ (แนวตั้ง) และเว้นบรรทัดแทนแถว (แนวนอน) ยกตัวอย่างไฟล์ที่มีนามสกุล CSV ที่ใช้ในการทดสอบการนำข้อมูลลงสู่ระบบฐานข้อมูลบนคลาวด์จริง ดังรูป

```
PRES_NAME,BIRTH_YR,YRS_SERV,DEATH_AGE,PARTY,STATE_BORN
Washington G,1732,7,67,Federalist,Virginia
Adams J,1735,4,90,Federalist,Massachusetts
Jefferson T,1743,8,83,Demo-Rep,Virginia
Madison J,1751,8,85,Demo-Rep,Virginia
Monroe J,1758,8,73,Demo-Rep,Virginia
Adams J Q,1767,4,80,Demo-Rep,Massachusetts
Jackson A,1767,8,78,Democratic,South Carolina
Van Buren M,1782,4,79,Democratic,New York
Harrison W H,1773,0,68,Whig,Virginia
Tyler J,1790,3,71,Whig,Virginia
Polk J K,1795,4,53,Democratic,North Carolina
Taylor Z,1784,1,65,Whig,Virginia
Fillmore M,1800,2,74,Whig,New York
Pierce F,1804,4,64,Democratic,New Hampshire
Buchanan J,1791,4,77,Democratic,Pennsylvania
Lincoln A,1809,4,56,Republican,Kentucky
Johnson A,1808,3,66,Democratic,North Carolina
Grant U S,1822,8,63,Republican,Ohio
Hayes R B,1822,4,70,Republican,Ohio
Garfield J A,1831,0,49,Republican,Ohio
Arthur C A,1830,3,56,Republican,Vermont
Cleveland G,1837,8,71,Democratic,New Jersey
```

รูปที่ 4.1 ไฟล์สกุล CSV ที่ใช้ในการทดลอง

โดยไฟล์สกุล CSV สามารถสร้างได้จากโปรแกรมสเปรดชีต (Spreadsheet) ต่าง ๆ เช่น Misrosoft Excel, Numbers ฯลฯ นอกจากนี้ยังเป็นไฟล์นามสกุลมาตรฐานในการนำเข้า (Import) และการนำออก (Export) จากระบบฐานข้อมูลมายเอสคิวแอล

ในส่วนของระบบฐานข้อมูลแบบปกติ การนำไฟล์สกุล CSV เข้าสู่ระบบสามารถทำได้โดยใช้งานคำสั่งเอสคิวแอลยกตัวอย่างดังนี้

คำสั่งที่ 4.1 การนำเข้า (Import)

```
LOAD DATA INFILE 'D:/TableName.csv'
INTO TABLE TableName
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

แต่เนื่องจากระบบจัดการฐานข้อมูลบนคลาวด์ Google Cloud SQL ไม่สนับสนุนการทำงานของคำสั่ง LOAD DATA INFILE ส่งผลให้ไม่สามารถนำไฟล์ CSV ลงสู่ระบบฐานข้อมูลผ่านการทำงานของคำสั่งดังกล่าวได้ แต่ทั้งนี้ Google Cloud SQL สนับสนุนการนำไฟล์ที่มีนามสกุล SQL ลงสู่ระบบฐานข้อมูลบนคลาวด์โดยที่ไม่จำเป็นต้องเรียกใช้งานคำสั่ง LOAD DATA INFILE

ไฟล์นามสกุล SQL คือชนิดข้อมูลชนิดหนึ่งซึ่งภายในไฟล์จะประกอบด้วยคำสั่งภาษาเอสคิวแอล โดยเริ่มตั้งแต่การสร้างตาราง (Table) จนถึงการนำข้อมูลลงสู่ตาราง (INSERT INTO) ซึ่งสามารถสร้างได้จากการนำข้อมูลออก (Export) จากระบบฐานข้อมูล โดยจุดประสงค์หลักของการเก็บไฟล์ที่มีนามสกุล SQL คือเพื่อเป็นการสำรองข้อมูลในระบบฐานข้อมูล

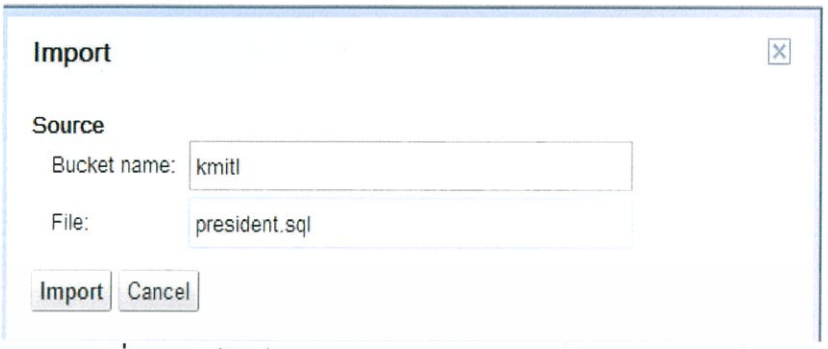
ดังนั้นการทดลองนำข้อมูลลงสู่ระบบฐานข้อมูลบนคลาวด์จะมีขั้นตอนดังต่อไปนี้

1) นำไฟล์นามสกุล CSV ลงสู่ระบบฐานข้อมูลแบบปกติผ่านการทำงานของคำสั่งLOAD DATA INFILE
2) สร้างไฟล์นามสกุล SQL จากฐานข้อมูลที่ต้องการโดยการใช้ mysqldump ยกตัวอย่างคำสั่งดังนี้

คำสั่งที่ 4.2 คำสั่งการสร้างไฟล์สกุล SQL (Export file)

```
mysqldump -uUsername -pPassword Database > Database.sql
```

3) อัพโหลดไฟล์สกุล SQL ลงสู่ Google Cloud Storage โดยใช้ Python และ gsutil
4) นำไฟล์สกุล SQL จาก Google Cloud Storage ลงสู่ Google Cloud SQL ดังรูป



รูปที่ 4.2 นำไฟล์สกุล SQL ลงสู่ระบบฐานข้อมูลบนคลาวด์

4.1.2 การนำแอพพลิเคชั่นขึ้นบนระบบคลาวด์ของกูเกิล

Google Cloud SQL สนับสนุนการทำงานร่วมกับ App Engine Java SDK เพื่อพัฒนาเว็บแอพพลิเคชั่นที่สามารถเชื่อมต่อกับระบบฐานข้อมูลที่อยู่บนคลาวด์ได้โดยเพียงอัพโหลดแอพพลิเคชั่นลงสู่ Google Cloud Environment ซึ่งผู้ใช้งานไม่จำเป็นต้องหมั่นตรวจสอบดูแลและรักษาการทำงานของเซิฟเวอร์ และแอพพลิเคชั่นต่าง ๆ จะทำงานอยู่บนกูเกิลคลาวด์อินฟราสตรัคเจอร์ผ่าน Google App Engine

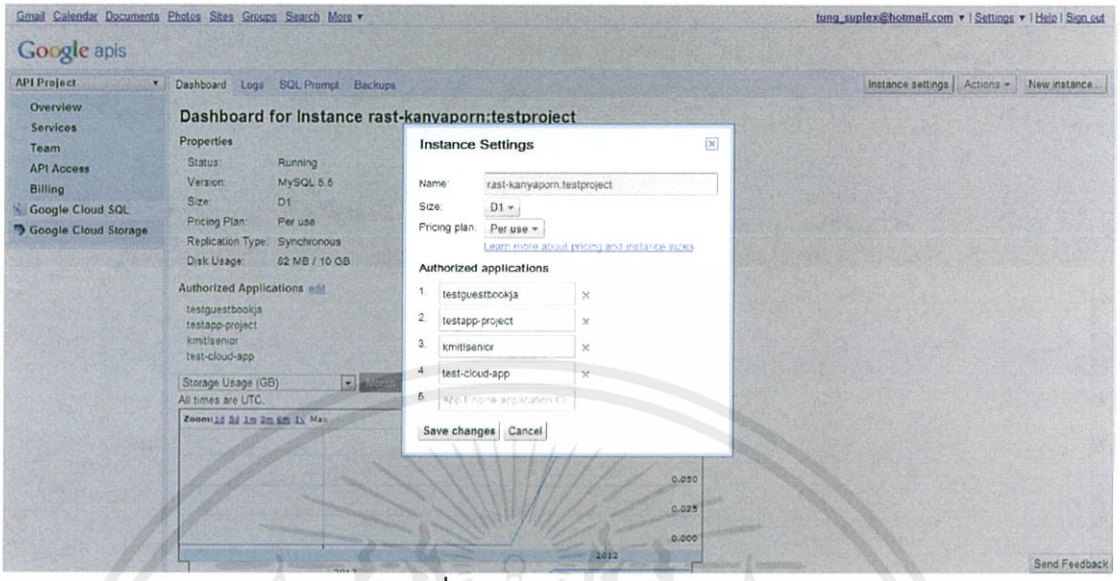
โดยในการทดลองการอัพโหลดแอพพลิเคชั่นจะแบ่งออกเป็นขั้นตอนดังต่อไปนี้

4.1.2.1 สร้างแอพพลิเคชั่นและฐานข้อมูลที่จะนำมาใช้งานสำหรับการทดสอบ

การสร้างและพัฒนาแอพพลิเคชั่นจะใช้ Java software development kit (SDK)โดย Java ที่ App Engine สนับสนุนคือ Java5 และ Java6 การพัฒนาแอพพลิเคชั่นจะทำบนโปรแกรม eclipse และใช้เครื่องมือ Google Plugin for Eclipse ในการช่วยอัพโหลดแอพพลิเคชั่น (เวอร์ชั่นของ eclipse ที่สนับสนุนคือ 3.3 , 3.4 , 3.5 , 3.6 และ 3.7) โดยขั้นตอนในการสร้างแอพพลิเคชั่นมีดังต่อไปนี้

1) สร้างและพัฒนาแอพพลิเคชั่น
2) กำหนดให้แอพพลิเคชั่นมีสิทธิในการติดต่อกับ Google Cloud SQL Instance โดยที่มีขั้นตอนดังต่อไปนี้

2.1) ที่หน้า Google Cloud SQL เลือกเมนู Instance setting



รูปที่ 4.3 Instance setting

2.2) ระบุ Google App Engine application ID ในกรอบ Authorized application

2.3) เลือก Save changes

3) สร้างฐานข้อมูลและตารางที่จะใช้ในการทดสอบยกตัวอย่างเช่น

คำสั่งที่ 4.3 สร้างฐานข้อมูล

```
CREATE DATABASE guestbook;
```

4) เชื่อมต่อกับฐานข้อมูลที่สร้างขึ้น

โดยก่อนที่จะสามารถเชื่อมต่อไปยังฐานข้อมูลได้จำเป็นต้องมีการลงทะเบียนไดรฟ์เวอร์ (Driver) ของ JDBC ก่อน ทั้งนี้ Google Plugin for Wclipse จะอณุญาติให้ใช้ไดรฟ์เวอร์ com.google.appengine.api.rdbms.AppEngineDriver โดยจำเป็นต้องนำมาใส่ในคลาสจาวาที่สร้างขึ้นดังนี้

คำสั่งที่ 4.4 อิมพอร์ทไดรฟ์เวอร์ของ Google Cloud SQL

```
Import com.google.appengine.api.rdbms.AppEngineDriver;
```

โดยในส่วนของโค้ดจาวาจะใช้เมธอด getConnection() ในการเชื่อมต่อกับฐานข้อมูล โดยหากการเชื่อมต่อจำเป็นต้องใช้ยูสเซอร์เนมและพาสเวิร์ดในการเชื่อมต่อ ส่วนของเมธอดจะมีพารามิเตอร์ที่รองรับดังตัวอย่างเช่น

คำสั่งที่ 4.5 เชื่อมต่อระบบฐานข้อมูลโดยใช้ยูสเซอร์เนมและพาสเวิร์ด

```
String url = "jdbc:google:rdbms://instance_name/database";
String user = "username";
String pass = "password";
Connection c = DriverManager.getConnection(url , user , pass);
```

และในส่วนสุดท้ายที่จะทำให้สามารถอัพโหลดแอพพลิเคชั่นได้สำเร็จ คือการระบุเซิร์ฟเล็ทคลาสกับยูอาร์แอลที่ใช้ในเว็บแอพพลิเคชั่นดังตัวอย่าง

คำสั่งที่ 4.6 ระบุเซิร์ฟเล็ทคลาสกับยูอาร์แอลที่ใช้

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns=http://java.sun.com/xml/ns/javaee
xmlns:web=http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">
<servlet>
     <servlet-name>sign</servlet-name>
     <servlet-class>guestbook.GuestServlet</servlet-class>
</servlet>
<servlet-mapping>
     <servlet-name>sign</servlet-name>
     <url-pattern>/sign</url-pattern>
</servlet-mapping>
<welcome-file-list>
     <welcome-file>guestbook.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

5) Deploy เว็บแอพพลิเคชั่นลงสู่ Google Cloud Environment และทดสอบความถูกต้องโดยเข้าไปที่ยูอาร์แอลที่อยู่บนโดเมน .appspot.com

4.1.3 การกำหนดสิทธิการเข้าใช้งานระบบฐานข้อมูลบนคลาวด์

เป็นการทดลองการกำหนดสิทธิในการใช้งานระบบฐานข้อมูลให้กับผู้ใช้งาน (GRANT) โดยจะทดสอบการกำหนดสิทธิในระดับดังต่อไปนี้

1) Administrative privileges

2) Database privileges

3) Privileges for database objects

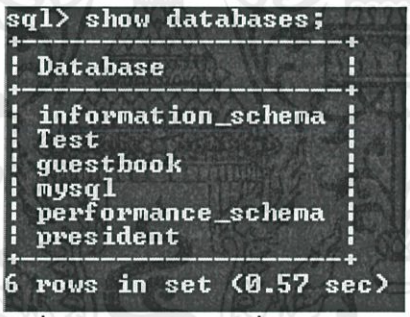
คำสั่งที่ใช้ในการกำหนดสิทธิของผู้ใช้งานมีรูปแบบดังตัวอย่างต่อไปนี้

คำสั่งที่ 4.7 กำหนดสิทธิในระดับ Database privileges

```
GRANT ALL ON db1.* TO 'jeffrey'@'localhost';
```

4.1.3.1 ทดลองสิทธิระดับ Administrative privileges

กำหนดให้ผู้ใช้งานที่มียูสเซอร์เนมเป็น 'test1'@'localhost' แล้วทดลองใช้คำสั่ง show databases; ในการแสดงฐานข้อมูลทั้งหมดในระบบฐานข้อมูลซึ่งการใช้คำสั่งนี้ได้จำเป็นต้องเป็นผู้ใช้งานที่มสิทธิในระดับ Administrative privileges เท่านั้นจึงจะสามารถใช้งานคำสั่งได้ดังตัวอย่างการทดลอง



รูปที่ 4.4 คำสั่งแสดงฐานข้อมูลที่อยู่ในระบบฐานข้อมูล

ทดลองคำสั่งที่สามารถใช้งานได้ในระดับ Administrative privileges เช่นคำสั่งในการสร้างยูสเซอร์เนมและพาสเวิร์ดของผู้ใช้งาน และ กำหนดสิทธิในการใช้งานฐานข้อมูลให้กับยูสเซอร์เนม

รูปที่ 4.5 สร้างยูสเซอร์เนม

รูปที่ 4.6 กำหนดสิทธิการใช้งาน

จากการทดลองจะเห็นว่าการสร้างยูสเซอร์เนมสามารถทำได้แต่ไม่สามารถกำหนดสิทธิการใช้งานฐานข้อมูลให้กับยูสเซอร์เนมได้เนื่องจากในการกำหนดสิทธินั้นจะต้องมี privileges "GRANT OPTION" ซึ่งเป็น privileges ในระดับ Administrative privileges โดยสามารถกำหนดให้กับ 'test1'@'localhost' ได้ดังนี้

คำสั่งที่ 4.8 กำหนด privileges "GRANT OPTION"

```
GRANT GRANT OPTION ON *.* TO 'test1'@'localhost';
```

ซึ่งเมื่อมีการกำหนดสิทธิดังกล่าวแล้ว 'test1'@'localhost' จะสามารถกำหนดสิทธิการใช้งานให้กับยูสเซอร์เนมอื่น ๆ ได้ รวมไปถึงสามารถจัดการกับสิทธิการใช้งานของยูสเซอร์เนมที่ตนเองดูแลได้ดังรูป

```
sql> GRANT ALL ON Test.* TO 'test3'@'localhost';
0 row(s) affected.
```

รูปที่ 4.7 กำหนดสิทธิการใช้งานให้กับ 'test3'@'localhost'

4.1.3.2 ทดลองสิทธิระดับ Database privileges

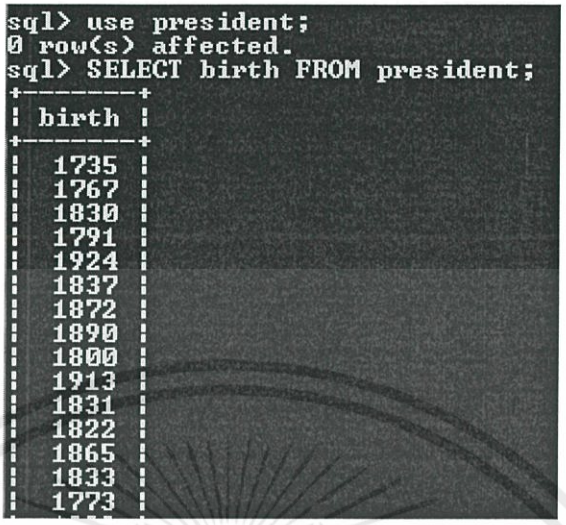
จากการทดลองในหัวข้อ 4.1.3.1 ได้สร้างยูสเซอร์เนมที่มีสิทธิในระดับ Database privileges คือยูสเซอร์เนม 'test2'@'localhost' โดยฐานข้อมูลที่มีสิทธิในการเข้าถึงคือฐานข้อมูล president ในการทดลองจะทดลองให้ยูสเซอร์เนม 'test2'@'localhost' ดึงข้อมูลจากฐานข้อมูล president รวมไปถึงการทดลองในการช้างานฐานข้อมูลอื่น ๆ เช่น guestbook และทดสอบการใช้งานคำสั่งในระดับ Administrative privileges ว่าสามารถทำงานได้หรือไม่อย่างไร

```
sql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| president          |
+--------------------+
2 rows in set (0.57 sec)
```

รูปที่ 4.8 ผลลัพธ์ของคำสั่งแสดงฐานข้อมูล

```
sql> use Test;
ERROR 1044 (SQLSTATE 42000): Access denied for user 'test2'@'localhost' to datab
ase 'Test'
```

รูปที่ 4.9 ผลลัพธ์การเข้าถึงฐานข้อมูล Test



รูปที่ 4.10 ผลลัพธ์การดึงข้อมูลจากตาราง president ในฐานข้อมูล president

จากผลการทดลองจะเห็นว่า 'test2'@'localhost' สามารถใช้งานคำสั่ง SHOW DATABASES ได้แต่ผลลัพธ์ที่ได้จะไม่เหมือนกับยูสเซอร์เนมในระดับ Administrative privileges ใช้งานคำสั่งดังกล่าว และยูสเซอร์เนม 'test2'@'localhost' ไม่สามารถเข้าถึงฐานข้อมูล Test ได้ เนื่องจากสิทธิในระดับ Database privileges ที่ได้รับมาสามารถเข้าถึงได้เพียงแค่ฐานข้อมูล president จึงสามารถเข้าถึงและดึงข้อมูลจากฐานข้อมูล president ได้ดังรูปที่ 4.10

4.1.3.3 ทดลองสิทธิระดับ Privileges for database objects

ในสิทธิระดับ Privileges for database objects จะเป็นระดับของหน่วยย่อยภายในฐานข้อมูล เช่น ตาราง (Table) , แถว (Row) และ หลัก (Column) เป็นต้น โดยในส่วนการทดลองจะมีการทดลองใช้งานคำสั่งในระดับ Table โดยดึงข้อมูลจากตารางหนึ่งในฐานข้อมูลที่ได้รับสิทธิในการเข้าถึงตารางนั้น ๆ เปรียบเทียบกับการดึงข้อมูลจากตารางที่ไม่ได้รับสิทธิในการเข้าถึงข้อมูลได้ รวมไปถึงการทดลองดึงข้อมูลเฉพาะหลัก (Column) ในตารางโดยที่กำหนดให้ยูสเซอร์เนมสามารถดึงข้อมูลได้เฉพาะหลัก ๆ เดียวในตารางดังคำสั่งตัวอย่างต่อไปนี้

คำสั่งที่ 4.9 กำหนดสิทธิการเข้าถึงในระดับ Privileges for database objects

```
GRANT ALL ON president.president TO 'test4'@'localhost';
GRANT SELECT (ID) ON president.book TO 'test4'@'localhost';
```


โดยการทดลองจะทดลองโดยการดึงข้อมูลจากตารางที่อยู่ในฐานข้อมูล president สองตารางคือ ตาราง president และตาราง theatre โดยจะได้ผลการทดลองดังรูป

```
sql> SELECT pres_name FROM president;
+-----------------+
| pres_name       |
+-----------------+
| Adams J         |
| Adams J Q       |
| Arthur C A      |
| Buchanan J      |
| Carter J E      |
| Cleveland G     |
| Coolidge C      |
| Eisenhower D D  |
| Fillmore M      |
| Ford G R        |
| Garfield J A    |
| Grant U S       |
| Harding W G     |
| Harrison B      |
| Harrison W H    |
| Hayes R B       |
| Hoover H C      |
```

รูปที่ 4.11 คำสั่งดึงข้อมูลจากตาราง president

```
sql> SELECT * FROM theatre;
ERROR 1142 (SQLSTATE 42000): SELECT command denied to user 'test4'@'localhost' f
or table 'theatre'
```

รูปที่ 4.12 คำสั่งดึงข้อมูลจากตาราง theatre

จากผลการทดลองจะเห็นได้ว่ายูสเซอร์เนม 'test4'@'localhost' สามารถดึงข้อมูลจากตาราง president ซึ่งเป็นตารางที่ได้รับสิทธิการเข้าถึงได้ แต่ตาราง theatre ไม่ได้รับสิทธิในการเข้าถึงข้อมูลจึงไม่สามารถเข้าถึงข้อมูลในตาราง theatre ได้

ในส่วนของการทดลองการดึงข้อมูลในระดับหลักของตาราง (Column) จะทดลองโดยการดึงข้อมูล ID ในตาราง book และดึงข้อมูลทุกหลักในตาราง book ได้ผลการทดลองดังรูป

```
sql> SELECT ID FROM book;
+-------------+
| ID          |
+-------------+
|           9 |
|          11 |
|          14 |
+-------------+
3 rows in set (1.83 sec)
```

รูปที่ 4.13 คำสั่งดึงข้อมูล ID จากตาราง book

```
sql> SELECT * FROM book;
ERROR 1142 (SQLSTATE 42000): SELECT command denied to user 'test4'@'localhost' f
or table 'book'
```

รูปที่ 4.14 คำสั่งดึงข้อมูลทุกหลักจากตาราง book

จากผลการทดลองจะเห็นได้ว่าไม่สามารถดึงข้อมูลจากทุกหลักในตาราง book ได้เนื่องจากสิทธิในการเข้าถึงข้อมูลในตาราง book ของ 'test4'@'localhost' อนุญาตให้เข้าถึงข้อมูลในหลัก ID เท่านั้น หลักอื่น ๆ ของตารางไม่สามารถเข้าถึงได้

4.1.4 การทดสอบปัญหา 4 ข้อของการประมวลผลทรานแซคชั่นแบบขนาน

เป็นการทดสอบการประมวลผลทรานเซคชั่นของระบบจัดการฐานข้อมูลบนคลาวด์ในแต่ละ Isolation Level ว่ามีผลลัพธ์ตรงตามทฤษฎีหรือไม่ โดยในการทดลองจะแยกหัวข้อตามลักษณะของปัญหาทั้งสี่คือ

1) Lost update

2) The uncommitted dependency problem

3) The inconsistency analysis problem

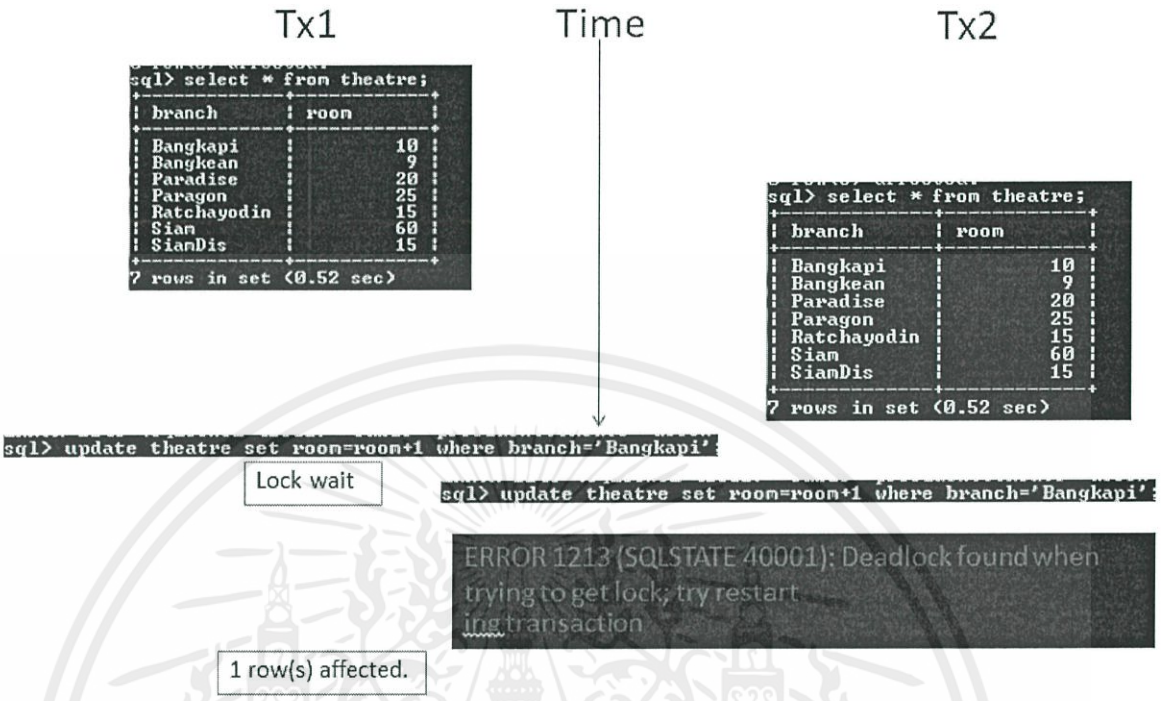
4) The phantom phenomenon

โดยการทดลองจะทดลองที่ตาราง theatre ในฐานข้อมูล president และอ้างอิงผลการทดลองตามทฤษฎีที่ได้ศึกษา

4.1.4.1 Lost update

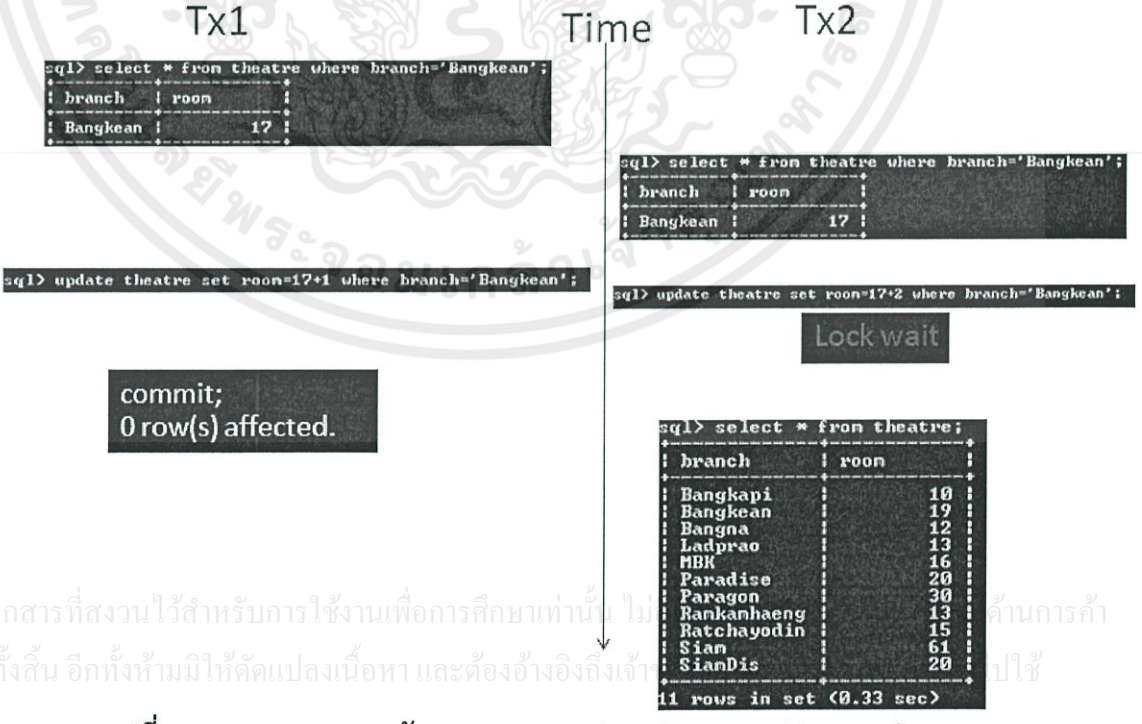
ปัญหา Lost updateแยกการทดลองตาม Isolation Level คือ serializable,Repeatable read , read committed และ read uncommitted

1) serializable จากผลการทดลองจะเห็นได้ว่าระบบจัดการฐานข้อมูลมีการป้องกันการเกิดปัญหา Lost update ด้วยการใช้ dead lock จะได้ผลการทดลองดังภาพด้านล่าง



รูปที่ 4.15 ผลการทดลองปัญหา Lost update (Serializable)

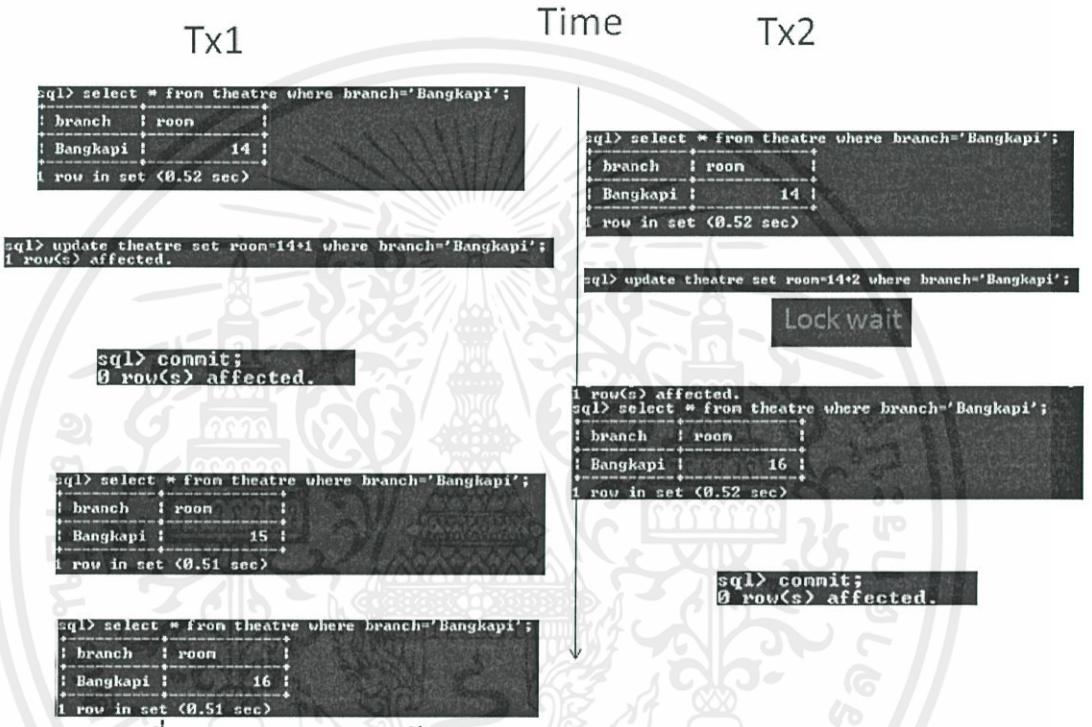
2) repeatable read ได้ผลการทดลองดังรูป



รูปที่ 4.16 ผลการทดลองปัญหา Lost update (Repeatable read)

จากผลการทดลองจะเห็นว่าปัญหา Lost update ยังคงมีโอกาสเกิดอยู่เนื่องจากกลไกข้างในการอ่านข้อมูลจะไม่มีการล็อคแถวที่อ่านเพราะใช้ Multiversion

3) read committed ได้ผลการทดลองดังรูป



รูปที่ 4.17 ผลการทดลองปัญหา Lost update (Read committed)

จากผลการทดลองจะเห็นว่าปัญหา Lost update ยังคงเกิดอยู่เนื่องจากเหตุผลในข้อที่แล้ว

4) read uncommitted ได้ผลการทดลองดังรูป

รูปที่ 4.18 ผลการทดลองปัญหา Lost update (Read uncommitted)

จากผลการทดลองจะเห็นว่าปัญหา Lost update ไม่ได้รับการแก้ไขซึ่งตรงตามกับทฤษฎีที่ได้ศึกษามา

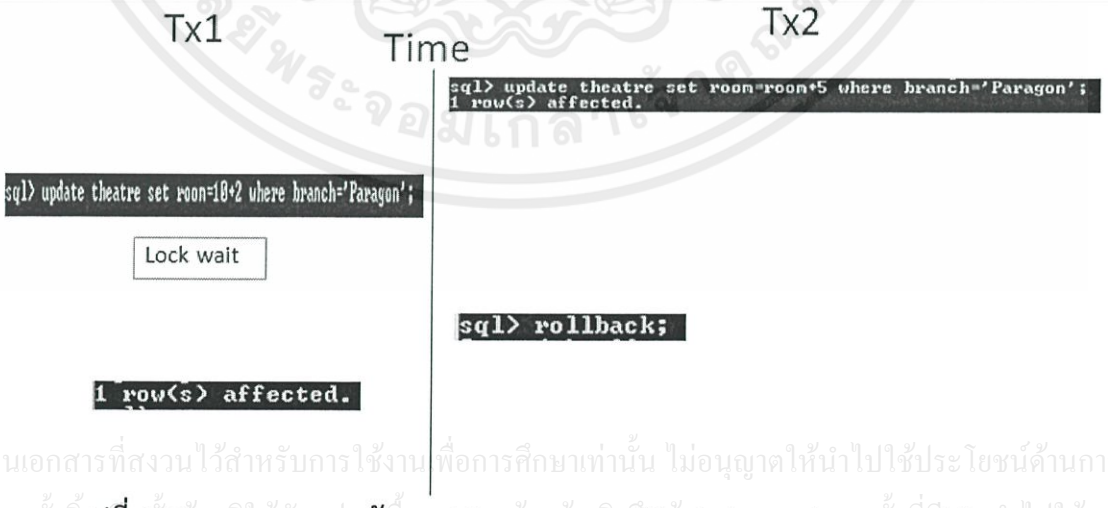
4.1.4.2 The uncommitted dependency problem

ปัญหา The uncommitted dependency problem แยกการทดลองตาม Isolation Level คือ serializable , repeatable read , read committed และ read uncommitted โดยปัญหา The uncommitted dependency problem จะเกิดขึ้นในสองกรณีคือ กรณีที่ทรานแซคชั่นแรกดึงข้อมูลออกมาอ่าน (SELECT) และกรณีที่ทรานแซคชั่นแรกแก้ไขข้อมูล (UPDATE)

1) serializable ได้ผลการทดลองดังรูป

รูปที่ 4.19 ผลการทดลองปัญหา The uncommitted dependency problem (serializable) กรณีการ SELECT

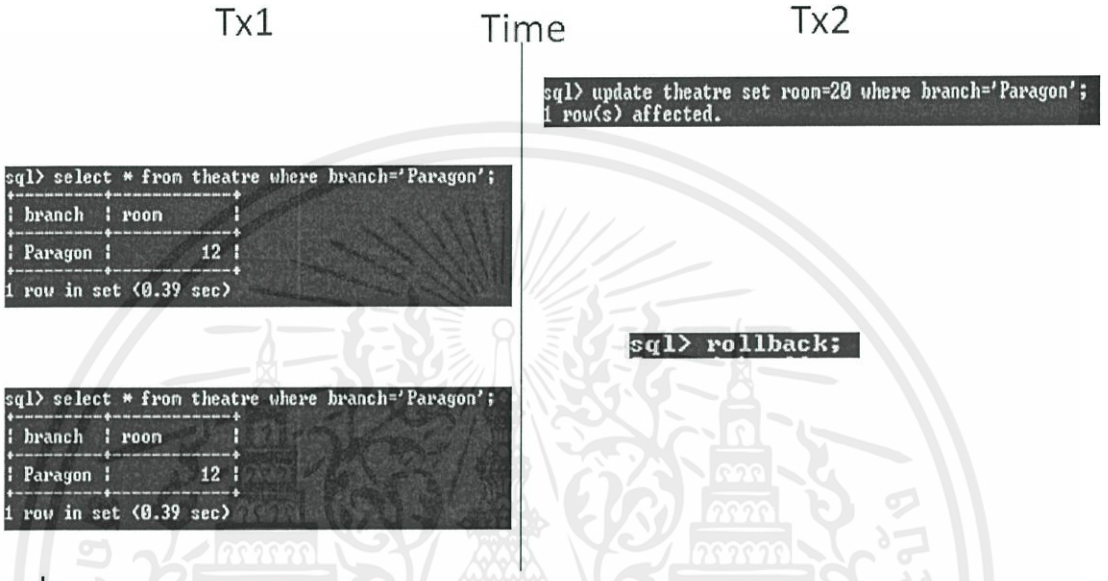
จากผลการทดลองจะเห็นได้ว่าทรานแซคชั่นที่หนึ่ง ติด Lock wait เนื่องจากข้อมูลที่ทรานแซคชั่นที่สองได้แก้ไขยังไม่ถูกยืนยัน (commit) ซึ่งเมื่อทรานแซคชั่นที่สองเกิดยกเลิกการแก้ไข (rollback) Lock ที่ทรานแซคชั่นที่หนึ่งติดอยู่จะถูกปลดปล่อย และสามารถดึงข้อมูลดังกล่าวออกมาดูได้ ซึ่งถือว่าเป็นป้องกันการเกิด The uncommitted dependency problem



รูปที่ 4.20 ผลการทดลองปัญหา The uncommitted dependency problem (serializable) กรณีการ UPDATE

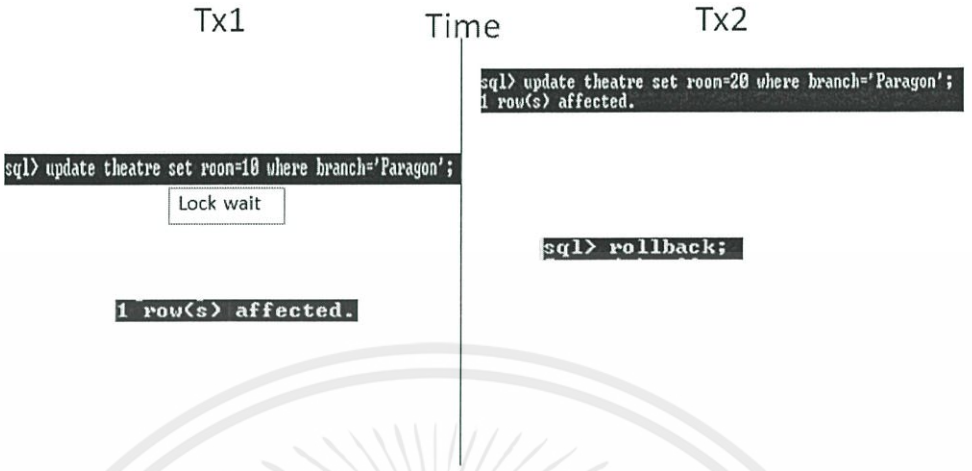
ซึ่งผลการทดลองที่ได้เหมือนกับกรณีของการ SELECT คือป้องกันไม่ให้เกิดปัญหา The uncommitted dependency problem

2) repeatable read ได้ผลการทดลองดังรูป



รูปที่ 4.21 ผลการทดลองปัญหา The uncommitted dependency problem (repeatable read) กรณีการ SELECT

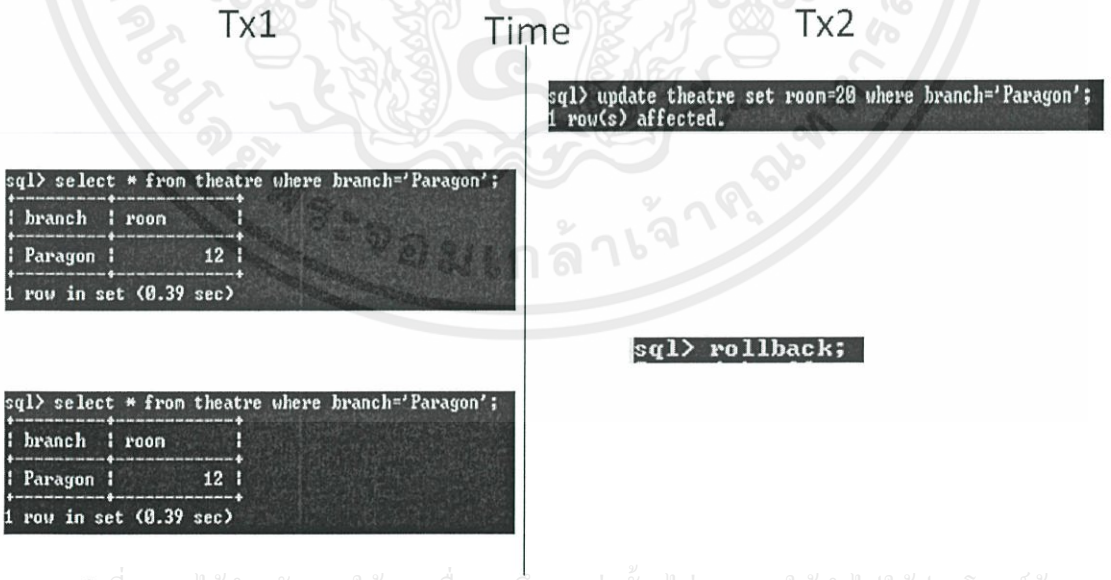
จากการทดลองจะเห็นได้ว่าทรานแซคชั่นที่หนึ่งสามารถอ่านข้อมูลออกมาได้ถึงแม้ว่าข้อมูลที่ทรานแซคชั่นที่สองแก้ไขยังไม่ถูกยืนยันก็ตามแต่ข้อมูลที่อ่านมาได้นั้นเป็นข้อมูลเก่าที่ยังไม่ถูกแก้ไขโดยทรานแซคชั่นที่สอง เพราะฉะนั้นเมื่อทรานแซคชั่นที่สองยกเลิกการเปลี่ยนแปลง จึงไม่เกิดผลกระทบต่อข้อมูลที่ทรานแซคชั่นที่หนึ่งอ่านมาได้ ซึ่งถือเป็นวิธีการหนึ่งในการป้องกันปัญหา The uncommitted dependency problem



รูปที่ 4.22 ผลการทดลองปัญหา The uncommitted dependency problem (repeatable read) กรณีการ UPDATE

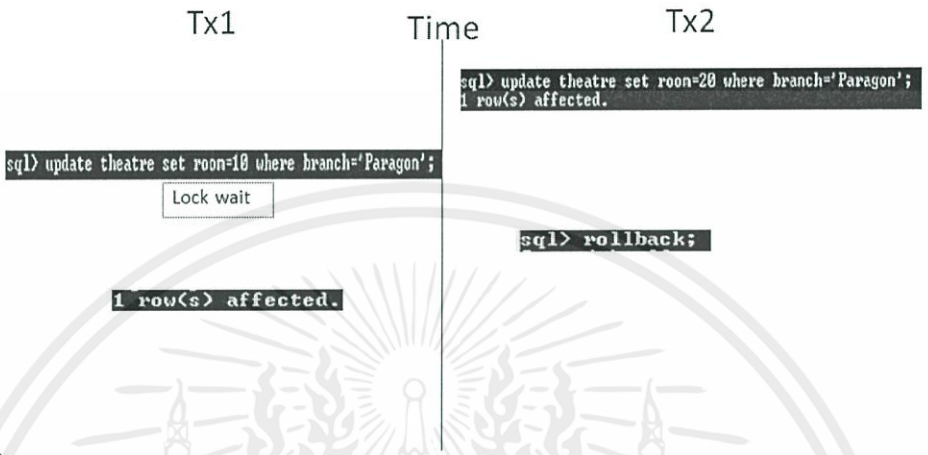
จากผลการทดลองจะเห็นว่าเมื่อทรานแซคชั่นที่หนึ่งพยายามจะแก้ไขข้อมูลที่ทรานแซคชั่นที่สองได้แก้ไขไว้แล้วแต่ยังไม่ยืนยัน ทรานแซคชั่นที่หนึ่งจะต้องรอให้ทรานแซคชั่นที่สองยืนยันหรือยกเลิกการเปลี่ยนแปลงของตนเองเสียก่อน ซึ่งถือเป็นอีกวิธีหนึ่งในการป้องกันปัญหา The uncommitted dependency problem

3) read committed ได้ผลการทดลองดังรูป



รูปที่ 4.23 ผลการทดลองปัญหา The uncommitted dependency problem (read committed) กรณีการ SELECT

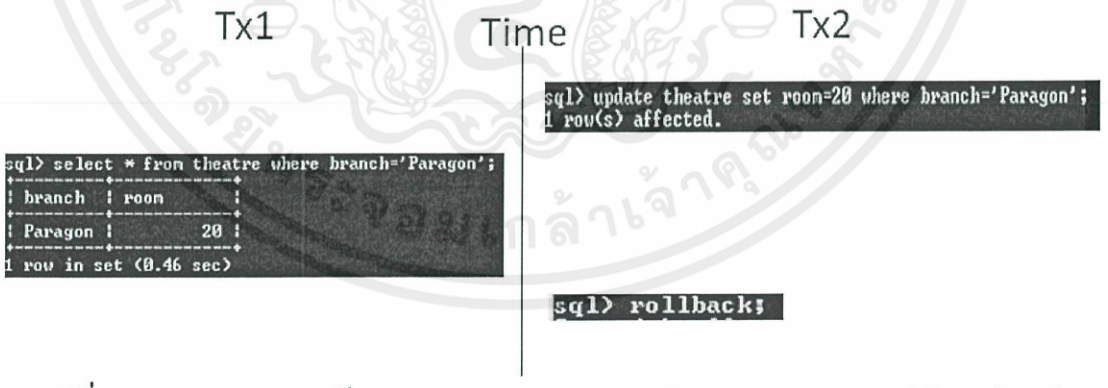
จะเห็นได้ว่าผลการทดลองเหมือนกับในระดับ repeatable read ซึ่งสามารถแก้ปัญหา The uncommitted dependency problem ได้



รูปที่ 4.24 ผลการทดลองปัญหา The uncommitted dependency problem (read committed) กรณีการ UPDATE

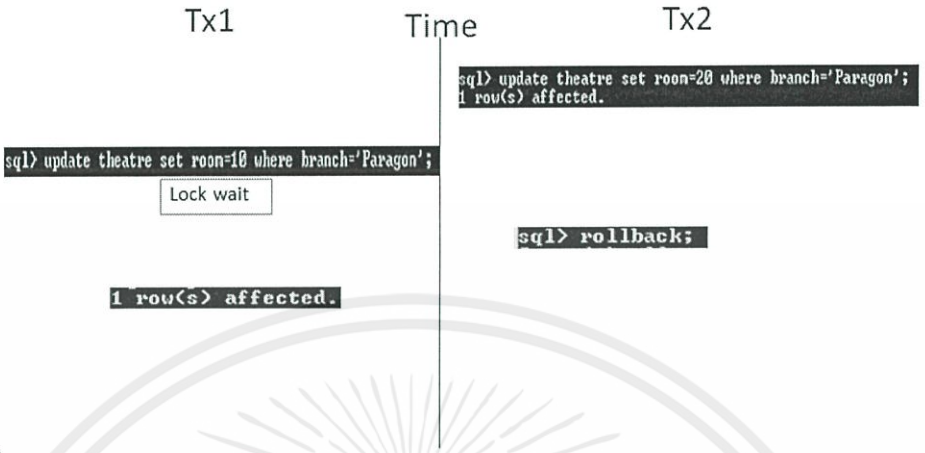
จากผลการทดลองจะเห็นได้ว่าผลลัพธ์เหมือนกับ repeatable ด้วยเหตุผลเดียวกันจึงสามารถป้องกันปัญหา The uncommitted dependency problem

4) read uncommitted ได้ผลการทดลองดังรูป



รูปที่ 4.25 ผลการทดลองปัญหา The uncommitted dependency problem (read uncommitted) กรณีการ SELECT

จากผลการทดลองจะเห็นได้ว่าไม่สามารถป้องกันปัญหา The uncommitted dependency problem ได้เนื่องจากทรานแซคชั่นที่หนึ่งอ่านข้อมูลที่ทรานแซคชั่นที่สองแก้ไขโดยที่ยังไม่ได้ยืนยันได้สำเร็จ



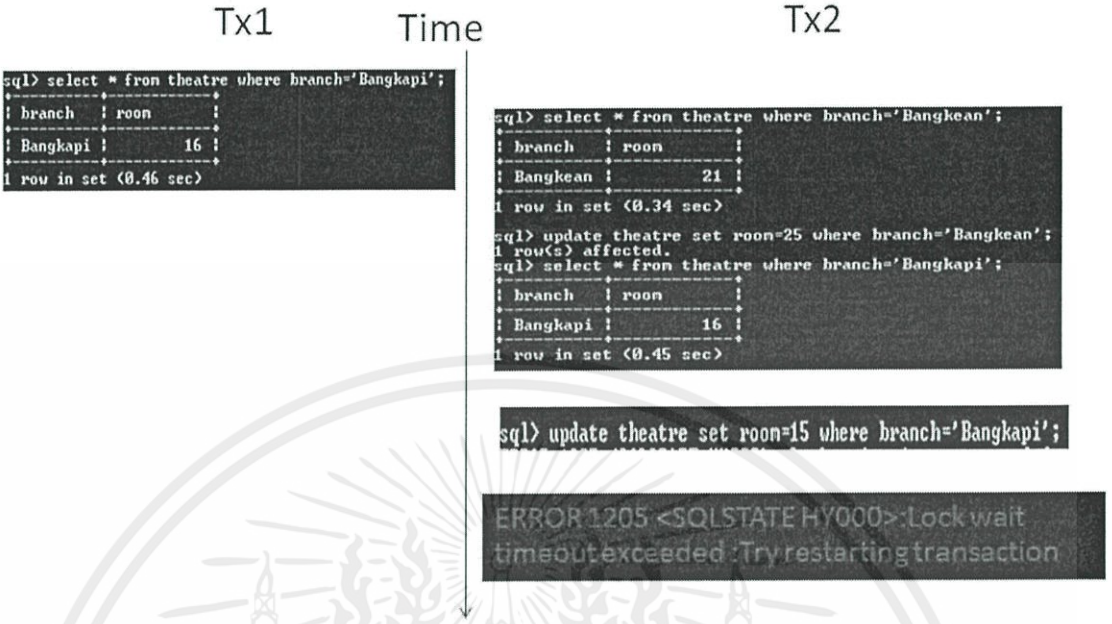
รูปที่ 4.26 ผลการทดลองปัญหา The uncommitted dependency problem (read uncommitted) กรณีการ UPDATE

จากผลการทดลองจะเห็นว่ายังมีการป้องกันในกรณีการ UPDATE ของทรานแซคชั่นที่หนึ่งจำเป็นจะต้องรอให้ทรานแซคชั้นที่สองยืนยันหรือยกเลิกการเปลี่ยนเปลงก่อนจึงจะสามารถเปลี่ยนแปลงข้อมูลในแถวนั้นๆได้

4.1.4.3 The inconsistent analysis problem

ปัญหา The uncommitted dependency problem แยกการทดลองตาม Isolation Level คือ serializable , repeatable read , read committed และ read uncommitted

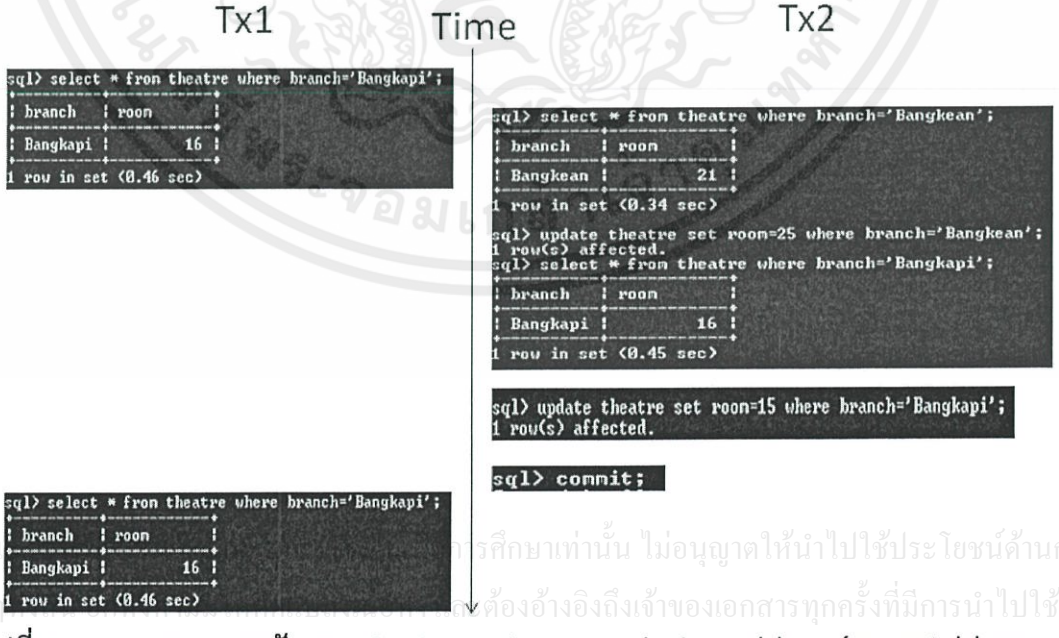
1) serializable ได้ผลการทดลองดังรูป



รูปที่ 4.27 ผลการทดลองปัญหา The inconsistent analysis problem (serializable)

จากผลการทดลองจะเห็นได้ว่าจะมีการป้องกันปัญหาเนื่องจากทรานแซคชั้นที่สองจะต้องรอ Lock ของทรานแซคชั่นที่หนึ่งเนื่องจากแถวของข้อมูลที่ทรานแซคชั่นที่หนึ่งอ่านอยู่จะติด shared lock ซึ่งทรานแซคชั่นที่สองจะไม่สามารถเข้าไปแก้ไขข้อมูลในแถวนั้นได้

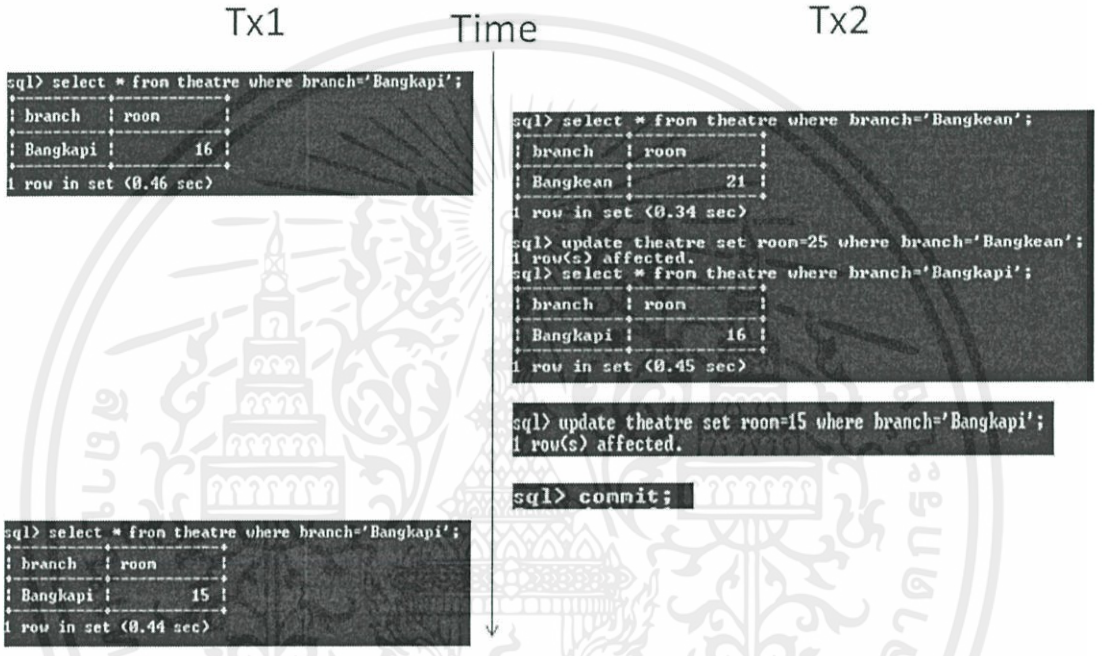
2) repeatable read ได้ผลการทดลองดังรูป



รูปที่ 4.28 ผลการทดลองปัญหา The inconsistent analysis problem (repeatable read)

จากการทดลองพบว่าไม่เกิดปัญหาของ The inconsistent analysis problem เนื่องจากทรานแซคชั่นที่หนึ่งอ่านข้อมูลจากค่าเก่าซึ่งยังไม่ถูกแก้ไขโดยทรานแซคชั่นที่สองถึงแม้ว่าทรานแซคชั่นที่สองจะยืนยันการเปลี่ยนแปลงไปแล้วก็ตามส่งผลให้การทำงานของทรานแซคชั่นที่หนึ่งและสองส่งผลเหมือนกระทำแบบอนุกรมกัน

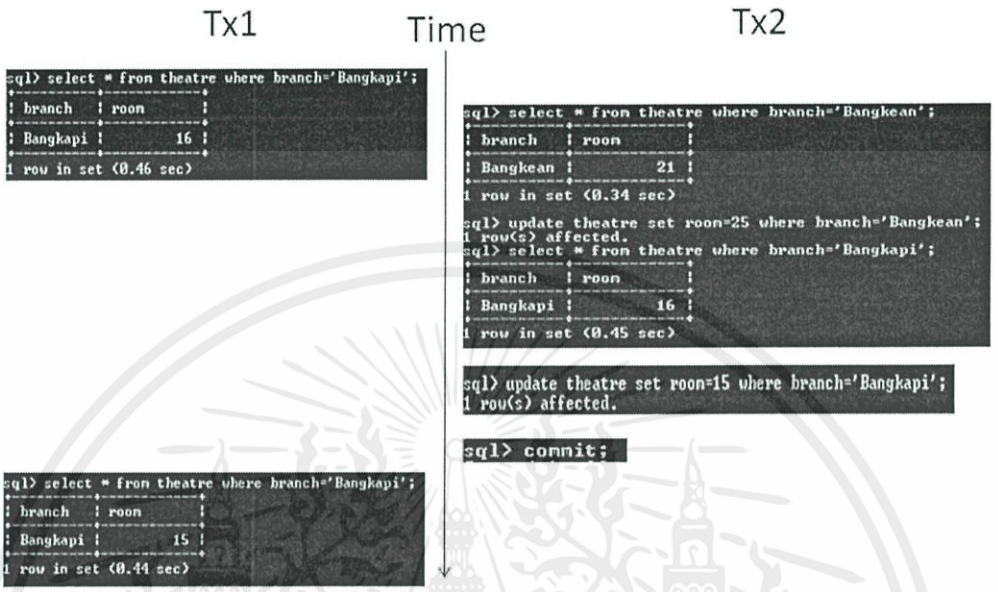
3) read committed ได้ผลการทดลองดังรูป



รูปที่ 4.29 ผลการทดลองปัญหา The inconsistent analysis problem (read committed)

จากผลการทดลองจะเห็นว่าจะเกิดปัญหา The inconsistent analysis problem เนื่องจากทรานแซคชั่นที่หนึ่งอ่านข้อมูลที่ทรานแซคชั่นที่สองแก้ไขและยืนยันแล้วทำให้จะเห็นได้ว่าการอ่านข้อมูลที่แถวเดียวกันสองครั้งมีผลลัพธ์ที่ต่างกันซึ่งถือว่าเป็นปัญหา

4) read uncommitted ได้ผลการทดลองดังรูป



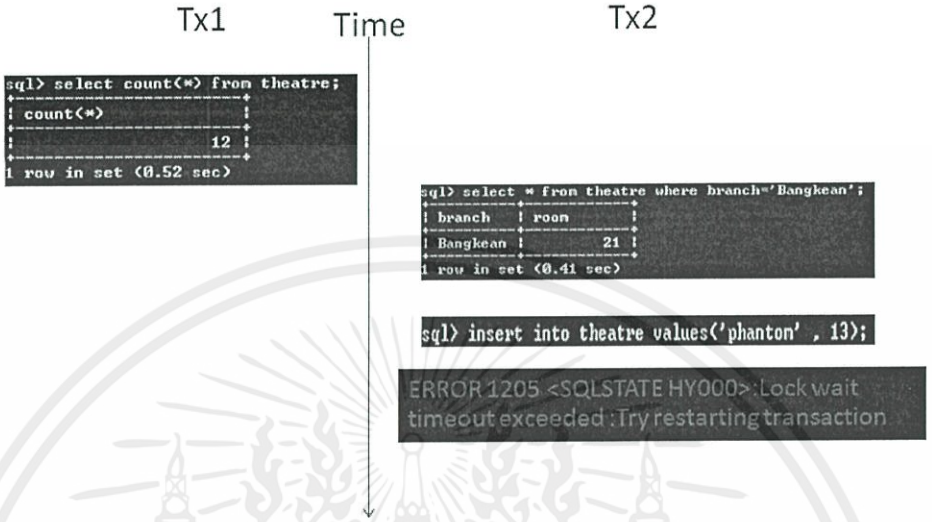
รูปที่ 4.30 ผลการทดลองปัญหา The inconsistent analysis problem (read uncommitted)

จากผลการทดลองจะเห็นว่าได้ผลลัพธ์เหมือน read committed ซึ่งยังคงไม่สามารถแก้ปัญหา The inconsistent analysis problem ได้

4.1.4.4 The phantom phenomenon

ปัญหา The uncommitted dependency problem แยกการทดลองตาม Isolation Level คือ serializable , repeatable read , read committed และ read uncommitted

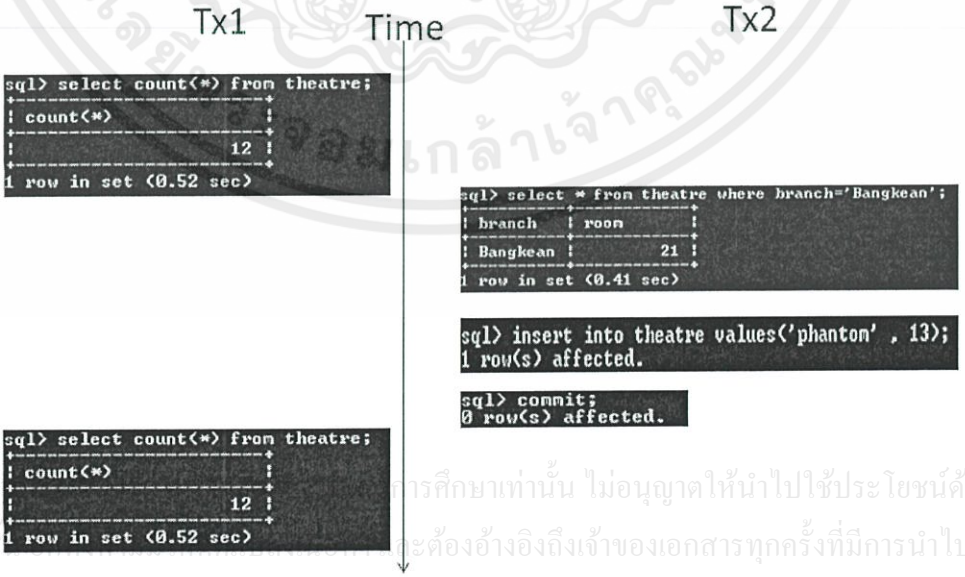
1) serializable ได้ผลการทดลองดังรูป



รูปที่ 4.31 ผลการทดลองปัญหา The phantom phenomenon (serializable)

จากการทดลองจะเห็นว่า serializable มีการป้องกันปัญหา The phantom phenomenon โดยการที่เมื่อทรานแซคชั่นที่หนึ่งอ่านค่าจากทุกแถวในตารางจะทำการล็อคทั้งตารางแบบ Shared lock ทำให้ทรานแซคชั่นที่สองไม่สามารถเพิ่มแถวใหม่ลงในตารางได้จนกว่าล็อคที่ทราดแซคชั่นที่หนึ่งรับผิดชอบอยู่จะถูกปลดปล่อย

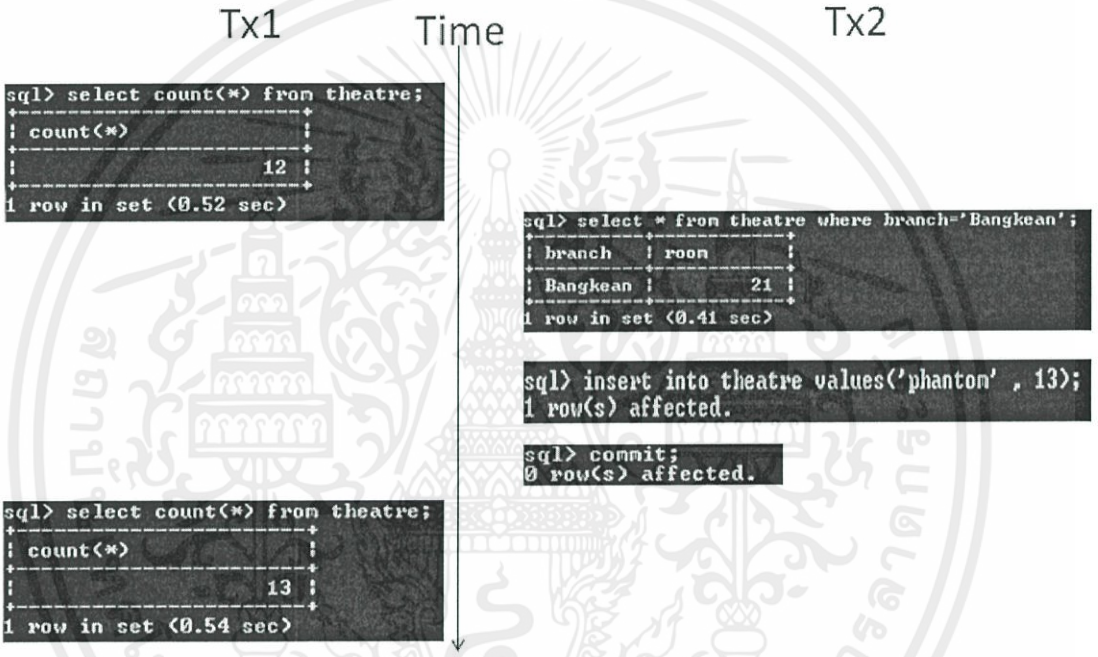
2) repeatable read ได้ผลการทดลองดังรูป



รูปที่ 4.32 ผลการทดลองปัญหา The phantom phenomenon (repeatable read)

จากผลการทดลองจะเห็นได้ว่ามีการป้องกันปัญหา The phantom phenomenon โดยจำนวนแถวของตารางที่ทรานแซคชั่นที่หนึ่งอ่านมาได้จะเป็นจำนวนแถวเก่า ซึ่งจะเห็นได้ว่าถึงแม้ทรานแซคชั่นที่สองทำการเพิ่มแถวใหม่ลงไปในตารางและยืนยันแล้ว ก็จะยังไม่ส่งผลต่อการทำงานของทรานแซคชั่นที่หนึ่งเนื่องจากทรานแซคชั่นที่หนึ่งจะไม่รับรู้ถึงแถวใหม่ที่ทรานแซคชั่นที่สองได้เพิ่มเข้ามา ส่งผลให้เปรียบเสมือนเป็นการทำงานแบบขนานกันของทั้งสองทรานแซคชั่น

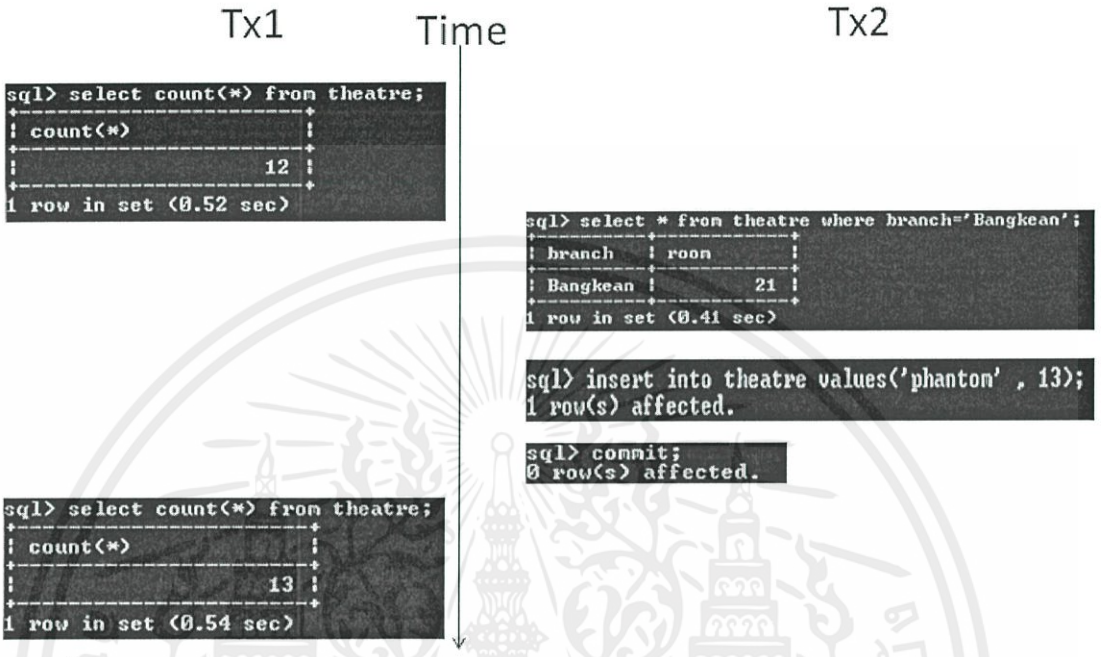
3) read committed ได้ผลการทดลองดังรูป



รูปที่ 4.33 ผลการทดลองปัญหา The phantom phenomenon (read committed)

จากการทดลองจะเห็นได้ว่าเกิดปัญหา The phantom phenomenon ขึ้นเนื่องจากทรานแซคชั่นที่หนึ่งอ่านจำนวนแถวในตารางสองครั้งได้ผลลัพธ์ไม่เหมือนกัน สาเหตุเพราะผลลัพธ์ในครั้งหลังกระทำหลังเกิดหลังจากที่ทรานแซคชั่นที่สองเพิ่มแถวใหม่ลงในตารางและยืนยันเรียบร้อยแล้ว

4) read uncommitted ได้ผลการทดลองดังรูป



รูปที่ 4.34 ผลการทดลองปัญหา The phantom phenomenon (read uncommitted)

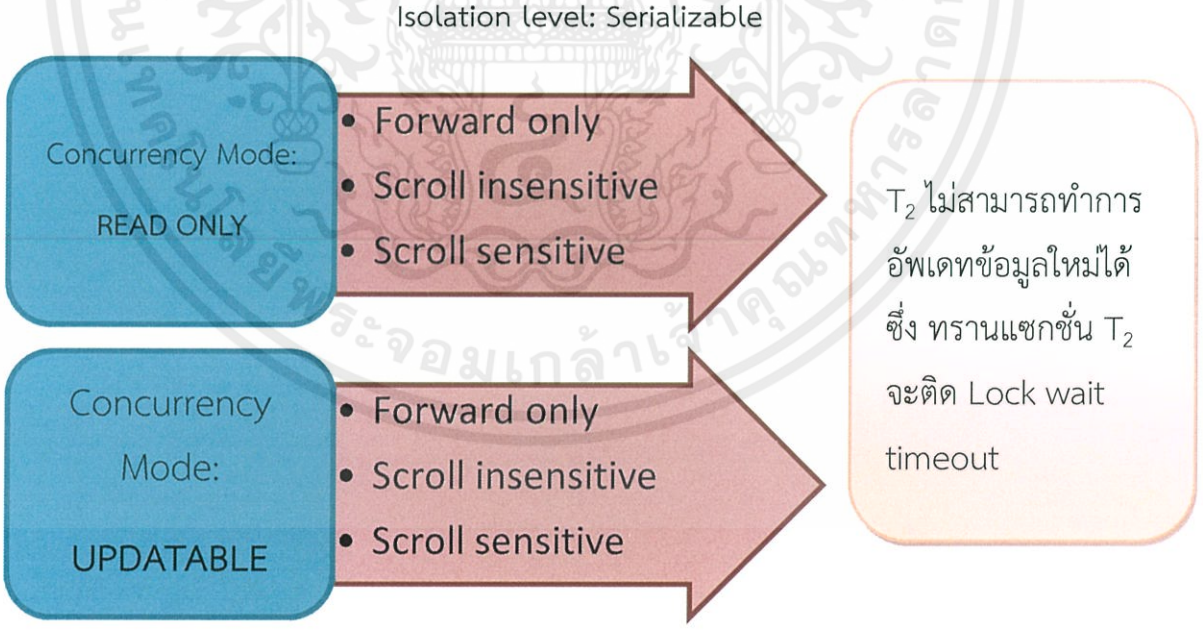
จากการทดลองจะเห็นได้ว่ามีผลลัพธ์เหมือนกับ read committed คือการอ่านจำนวนแถวของทรานแซคชั่นที่หนึ่งทั้งสองครั้งนั้นได้คำตอบไม่เหมือนกันซึ่งเป็นผลจากปัญหา The phantom phenomenon

4.1.4.5 การทดลองปัญหา The 4 concurrency control problem ด้วย CURSOR ในภาษาจาวา (เฉพาะ The inconsistent analysis problem และ The Phantom phenomenon)

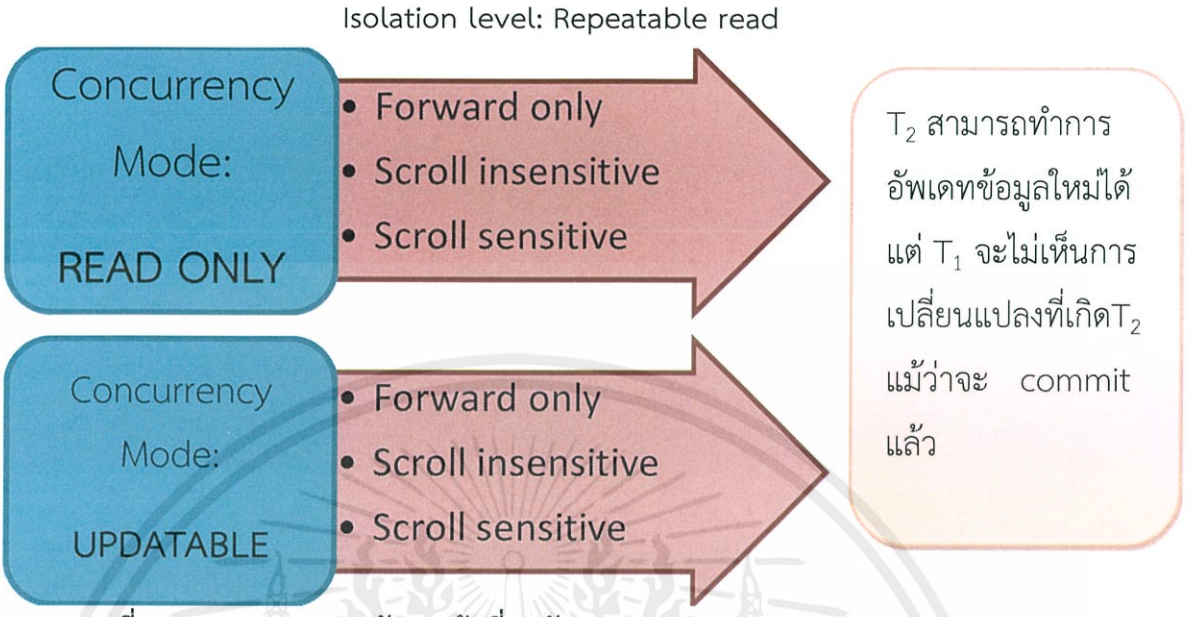
รูปที่ 4.35 การใช้ CURSOR ทดลองปัญหา The inconsistent analysis problem

4.1.5 การทดลองปัญหาข้อที่ 3 The inconsistent analysis problem ด้วย CURSOR

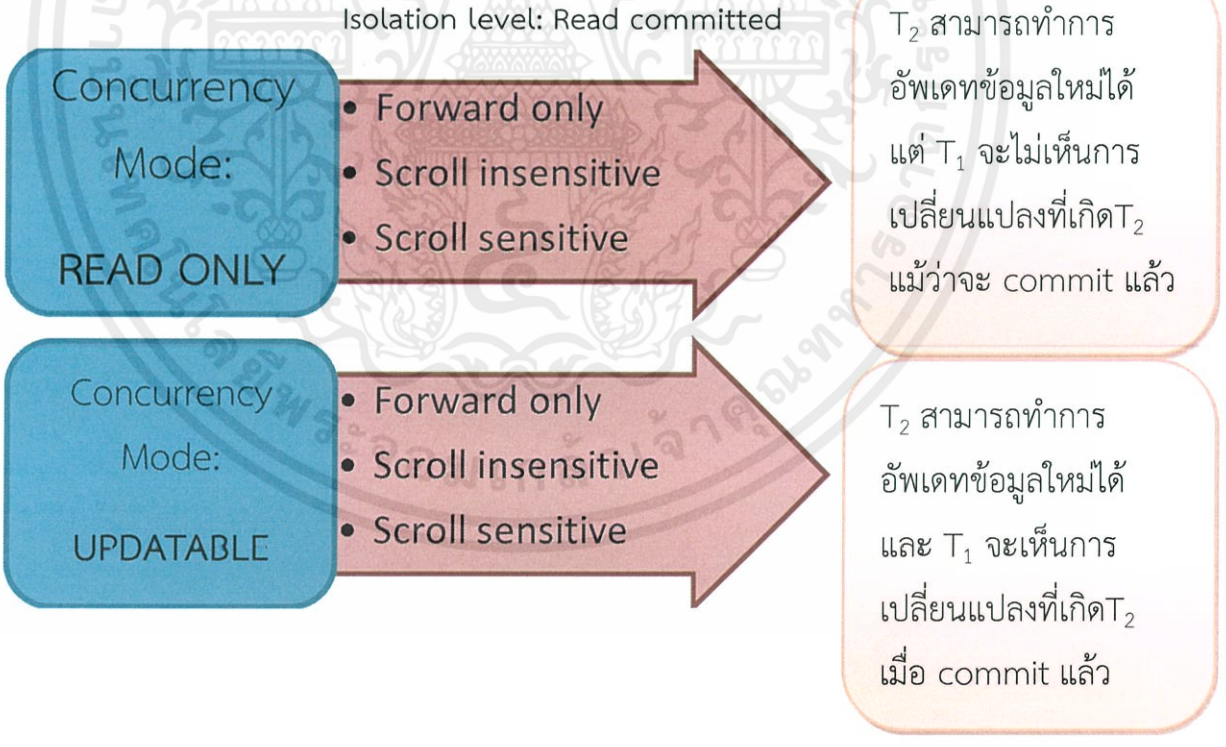
สมมติให้ในการทดลอง มีทรานแซกชั่น 2 ทรานแซกชั่น โดยทรานแซกชั่นแรก เป็น T_1 และทรานแซกชั่นที่ 2 เป็น T_2 ตามลำดับ



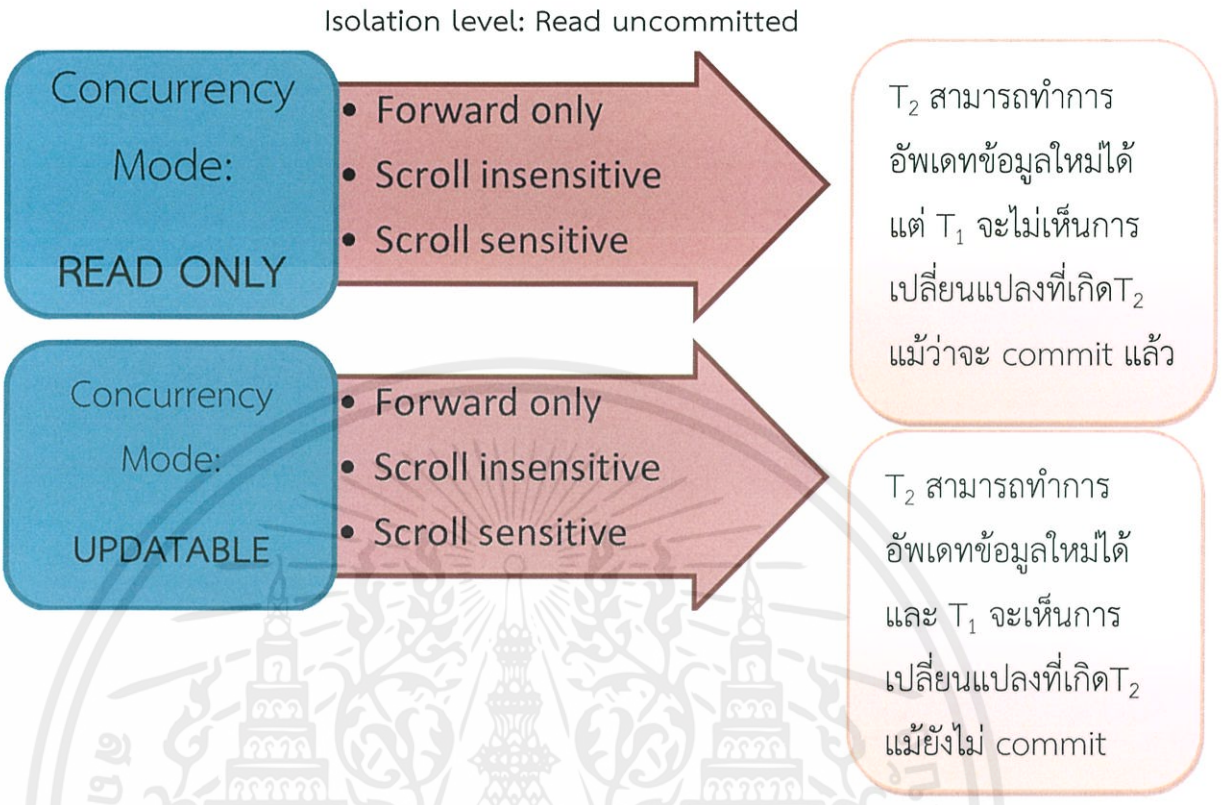
รูปที่ 4.36 ผลการทดลองปัญหาข้อที่ 3 กับ Isolation level : Serializable



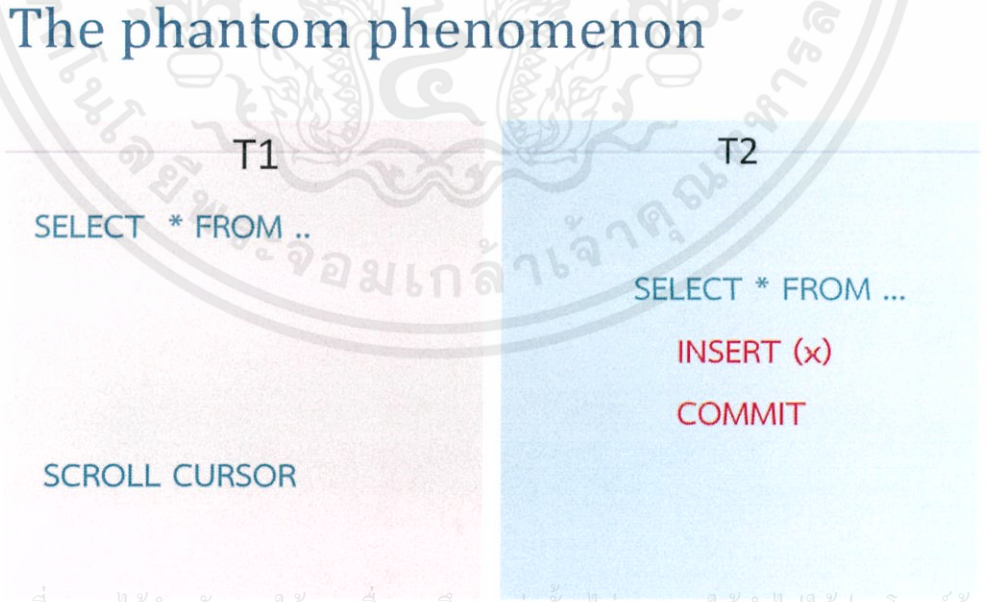
รูปที่ 4.37 ผลการทดลองปัญหาข้อที่ 3 กับ Isolation level : Repeatable read



รูปที่ 4.38 ผลการทดลองปัญหาข้อที่ 3 กับ Isolation level : Read committed



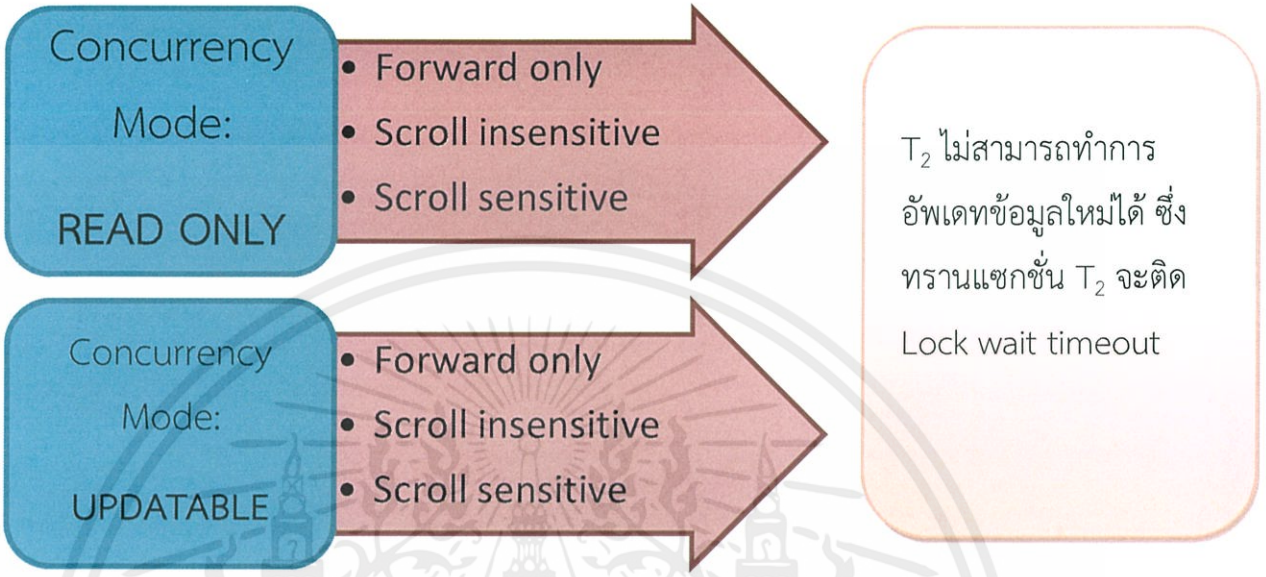
รูปที่ 4.39 ผลการทดลองปัญหาข้อที่ 3 กับ Isolation level : Read uncommitted



รูปที่ 4.40 การใช้ CURSOR ทดลองปัญหา The Phantom phenomenon problem

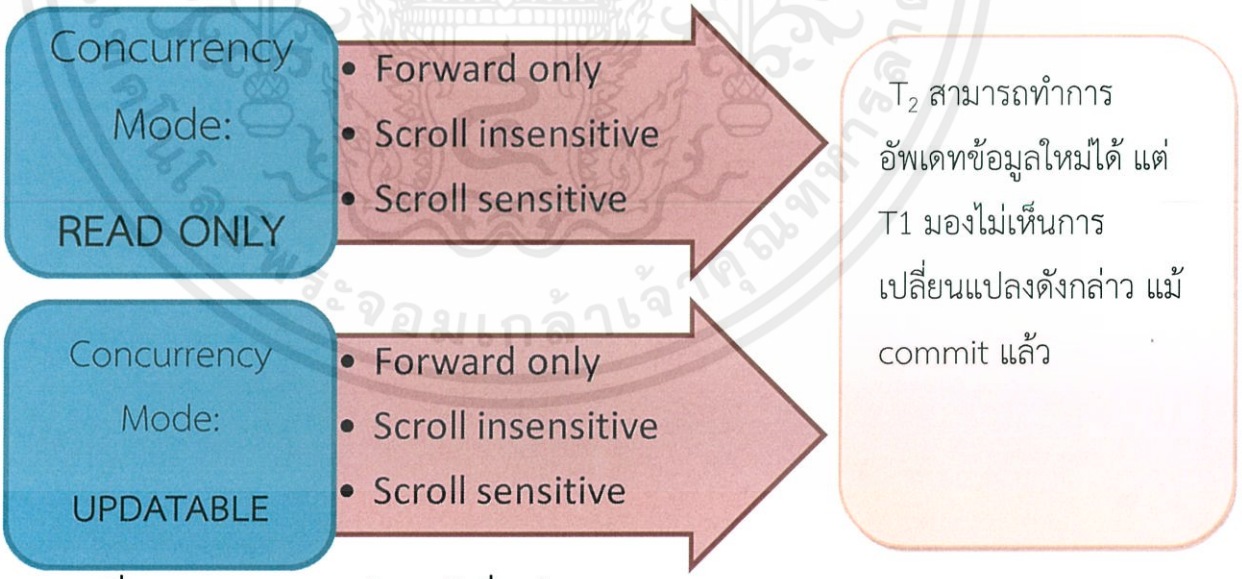
4.1.6 การทดลองปัญหาข้อที่ 4 The phantom phenomenon ด้วย CURSOR

Isolation level: Serializable

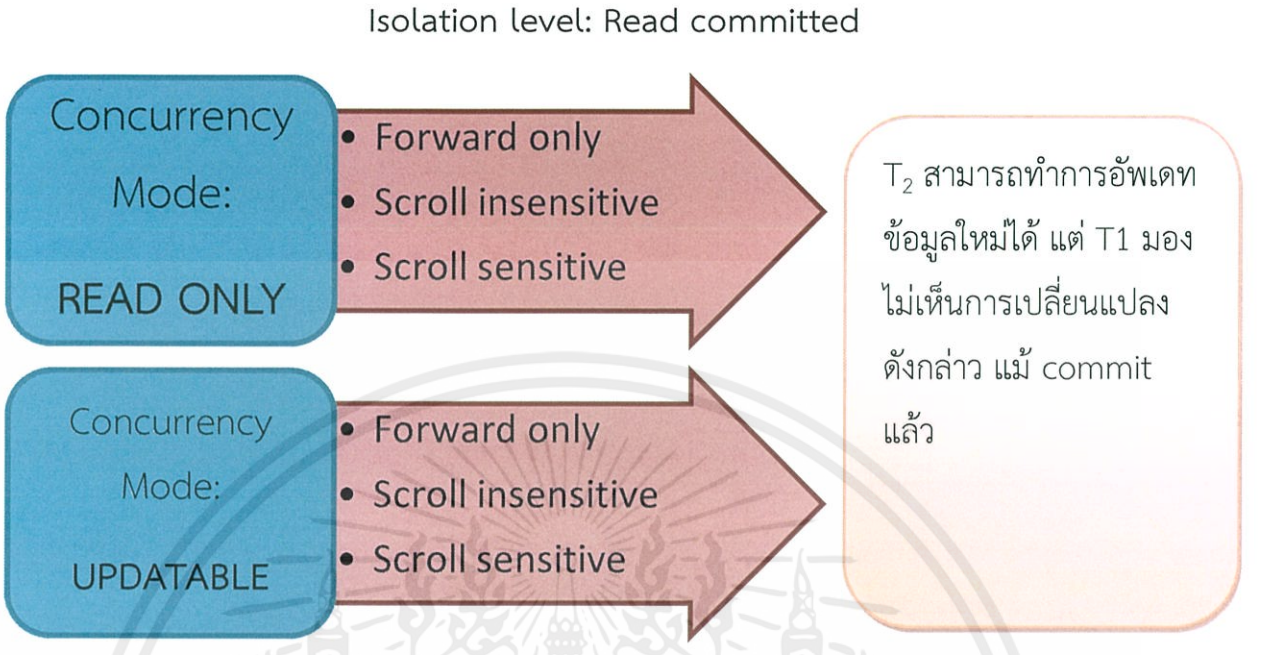


รูปที่ 4.41 ผลการทดลองปัญหาข้อที่ 4 กับ Isolation level : Serializable

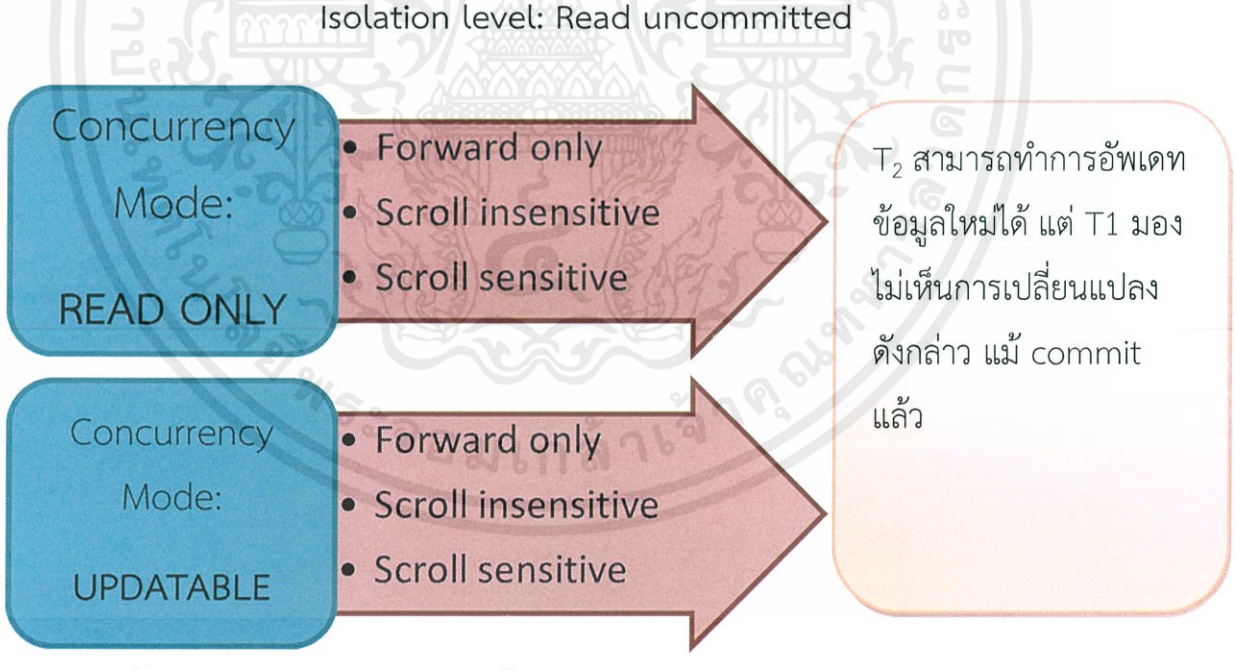
Isolation level: Repeatable read



รูปที่ 4.42 ผลการทดลองปัญหาข้อที่ 4 กับ Isolation level : Repeatable read



รูปที่ 4.43 ผลการทดลองปัญหาข้อที่ 4 กับ Isolation level : Read committed



รูปที่ 4.44 ผลการทดลองปัญหาข้อที่ 4 กับ Isolation level : Read uncommitted

การที่ปัญหาที่ 3 เมื่อ T_2 ทำการอัพเดทค่าและ T_1 ทำการเห็นค่าดังกล่าว ทั้งที่เห็นหลังจาก Commit หรือ ก่อน Commit เพราะใช้การ Refresh cursor ที่แถวนั้นๆใหม่อีกครั้งหนึ่ง ซึ่งเป็นการล็อคเอาค่าใหม่ของแถวนั้น ที่ฝั่ง Server ในขณะที่ ปัญหาที่ 4 เกิดจากการ insert แถวเข้าไปใหม่ ดังนั้นแม้ทำการ Refresh ใหม่ ส่วนใหญ่ก็ยังไม่เห็นค่าใหม่ดังกล่าว เพราะ การ insert ไม่ได้มีการล็อคเกิดขึ้น โดยจะเห็นได้ว่า ถ้าเซ็ตระดับไอโซเลชั่นไว้สูง เช่น Serializable นั้น ทรานแซกชั่นอื่นจะไม่สามารถทำการ insert ได้ เนื่องจากติด Xlock ที่แถวนั้นๆ แต่ระดับอื่นๆ สามารถinsert ได้แต่ก็ไม่สามารถเห็นค่าใหม่ได้ จนกว่าจะมีการ SELECTใหม่อีกครั้งหนึ่ง ซึ่งการที่ insert ไม่เข้า หรือ เข้าแต่ไม่เห็นการเปลี่ยนแปลงที่เกิดจากทรานแซกชั่นอื่น ล้วนเรียกว่า ไม่มี phantom row หรือไม่เกิดปรากฏการณ์ Phantom phenomenon

หากพิจารณาจากผลการทดลองเราจะพบว่า ผลลัพธ์ที่ได้ของ ResultSet นั้น จะขึ้นอยู่กับ Concurrency mode และ isolation level ที่ตั้งค่าไว้เป็นหลัก เนื่องจากผลการทดลองแสดงให้เห็นว่า ไม่ว่า CURSOR TYPE จะเซ็ตค่าไว้เป็นอะไร ผลลัพธ์นั้นจะขึ้นอยู่กับสองปัจจัยที่กล่าวไปข้างต้นเท่านั้น ดังนั้นเราอาจมองได้ว่า CURSOR TYPE เป็น Don't care

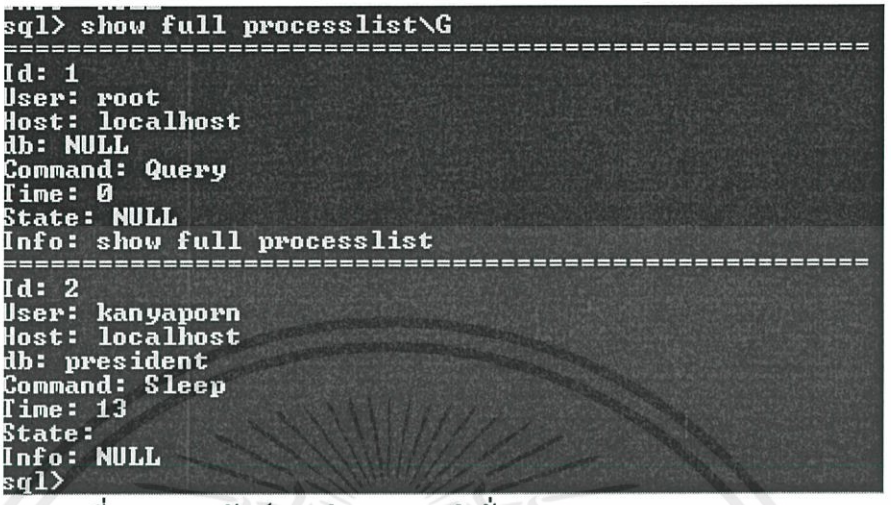
4.1.7 ทดสอบการตรวจสอบสถานะการทำงานต่างๆของระบบ

ในส่วนของการใช้งานระบบจัดการฐานข้อมูลบนคลาวด์จะมีความแตกต่างกับระบบจัดการฐานข้อมูลแบบปกติอยู่เล็กน้อยคือ ในส่วนของระบบฐานข้อมูลแบบปกติ เมื่อหลังจากการติดตั้งแล้วจะมีเครื่องมือที่ช่วยในการตรวจสอบสถานะการทำงานรวมไปถึงการตรวจสอบความผิดปกติต่างๆ ภายในระบบ ซึ่งในส่วนของระบบจัดการฐานข้อมูลบนคลาวด์ ผู้ใช้จะไม่สามารถไปเรียกใช้งานได้โดยตรง ดังนั้นจึงจำเป็นต้องมีการใช้งานคำสั่งบางคำสั่งหรือการใช้งาน meta table บางอย่างภายในระบบฐานข้อมูล เพื่อตรวจสอบสถานะการทำงานต่างๆของฐานข้อมูลแทนการใช้เครื่องมือช่วยเสริมเหมือนระบบจัดการฐานข้อมูลแบบปกติ

โดยการทดลองจะยกตัวอย่างการใช้งานคำสั่งบางอย่างที่จะไปอ่านข้อมูลจากตารางภายใน information schema ของระบบจัดการฐานข้อมูลเพื่อตรวจสอบการทำงานของฐานข้อมูลในเวลาปัจจุบัน ยกตัวอย่างดังรูป

```
sql> show processlist;
+----+----------+-----------+-----------+---------+------+-------+------+
| Id | User     | Host      | db        | Command | Time | State | Info |
+----+----------+-----------+-----------+---------+------+-------+------+
|  1 | root     | localhost | NULL      | Query   |    0 | NULL  | show processlist |
|  2 | kanyaporn | localhost | president | Sleep   |    2 |       | NULL |
+----+----------+-----------+-----------+---------+------+-------+------+
2 rows in set (0.83 sec)
```

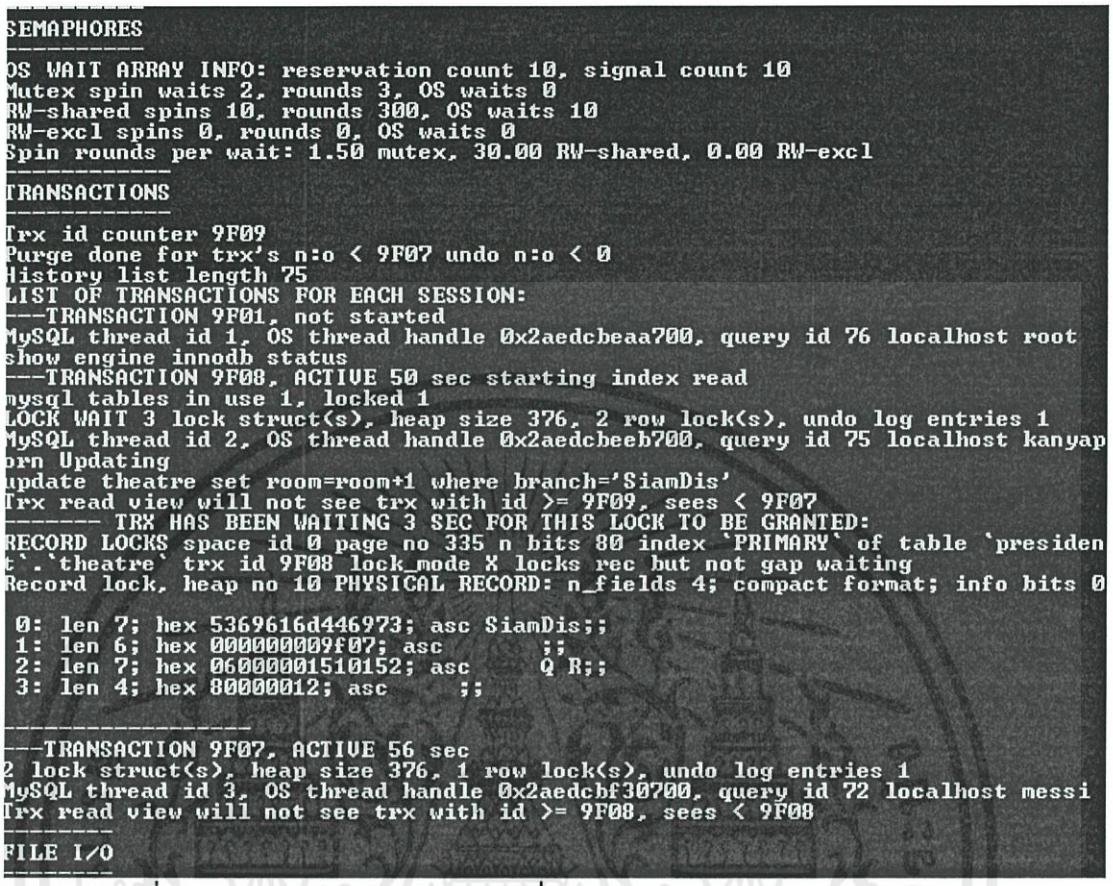
รูปที่ 4.45 ผลลัพธ์การทำงานของคำสั่ง SHOW PROCESSLIST



รูปที่ 4.46 ผลลัพธ์การทำงานของคำสั่ง SHOW PROCESSLIST\G

จากรูปผลการทดลองของคำสั่ง SHOW PROCESSLIST และ SHOW PROCESSLIST\G จะได้ผลลัพธ์การทำงานเหมือนกันเพียงแต่รูปแบบในการนำเสนอของผลลัพธ์จะไม่เหมือนกัน โดยเป็นการอ่านข้อมูลจาก PROCESSLIST ซึ่งเป็นตารางที่อยู่ภายในฐานข้อมูล information schema ของระบบฐานข้อมูลที่มีไว้เพื่อช่วยสนับสนุนในส่วนของการดูสถานะการทำงานในปัจจุบันของผู้ใช้งานที่เข้ามาใช้งานระบบฐานข้อมูล ณ เวลานั้นๆ

นอกจากนี้ยังมีการตรวจสอบการทำงานของ InnoDB Engine ซึ่งเป็น Engine หนึ่งของระบบจัดการฐานข้อมูล MySQL ที่สนับสนุนการทำงานของทรานแซคชั่น โดยผู้ใช้งานสามารถเรียกดูการทำงานของ InnoDB Engine ได้โดยการใช้งานคำสั่ง SHOW ENGINE INNODB STATUS ซึ่งจะได้ผลลัพธ์ดังภาพ

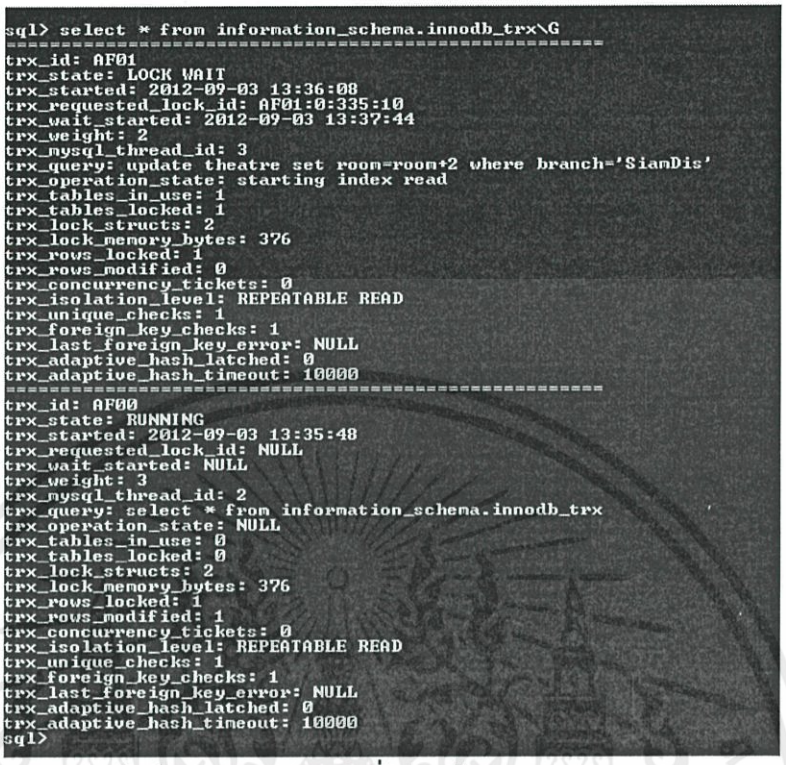


```
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 10, signal count 10
Mutex spin waits 2, rounds 3, OS waits 0
RW-shared spins 10, rounds 300, OS waits 10
RW-excl spins 0, rounds 0, OS waits 0
Spin rounds per wait: 1.50 mutex, 30.00 RW-shared, 0.00 RW-excl
------------
TRANSACTIONS
------------
Trx id counter 9F09
Purge done for trx's n:o < 9F07 undo n:o < 0
History list length 75
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 9F01, not started
MySQL thread id 1, OS thread handle 0x2aedcbeaa700, query id 76 localhost root
show engine innodb status
---TRANSACTION 9F08, ACTIVE 50 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 3 lock struct(s), heap size 376, 2 row lock(s), undo log entries 1
MySQL thread id 2, OS thread handle 0x2aedcbeeb700, query id 75 localhost kanyap
orn Updating
update theatre set room=room+1 where branch='SiamDis'
Trx read view will not see trx with id >= 9F09, sees < 9F07
------- TRX HAS BEEN WAITING 3 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 0 page no 335 n bits 80 index `PRIMARY` of table `presiden
t`.`theatre` trx id 9F08 lock_mode X locks rec but not gap waiting
Record lock, heap no 10 PHYSICAL RECORD: n_fields 4; compact format; info bits 0

 0: len 7; hex 5369616d446973; asc SiamDis;;
 1: len 6; hex 000000009f07; asc       ;;
 2: len 7; hex 06000001510152; asc     Q R;;
 3: len 4; hex 80000012; asc     ;;

------------------
---TRANSACTION 9F07, ACTIVE 56 sec
2 lock struct(s), heap size 376, 1 row lock(s), undo log entries 1
MySQL thread id 3, OS thread handle 0x2aedcbf30700, query id 72 localhost messi
Trx read view will not see trx with id >= 9F08, sees < 9F08
---------
FILE I/O
---------
```

รูปที่ 4.47 ผลลัพธ์การทำงานของคำสั่ง SHOW ENGINE INNODB STATUS

หมายเหตุ การทำงานของคำสั่ง SHOW PROCESSLIST และ SHOW ENGINE INNODB STATUS สามารถทำงานได้บนระบบจัดการฐานข้อมูลแบบปกติเช่นกัน

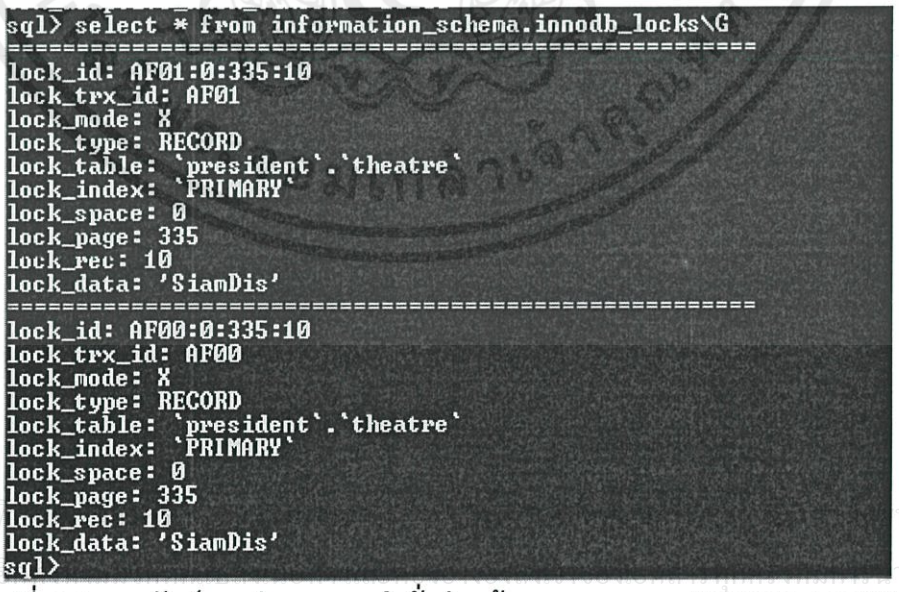
นอกจากนี้ตารางอื่นๆที่สามารถใช้ในการตรวจสอบการทำงานของระบบฐานข้อมูลในฐานข้อมูล information schema ยังมีอีกหลายตารางยกตัวอย่างเช่น ตาราง INNODB_TRX , ตาราง INNODB_LOCKS และตาราง INNODB_LOCK_WAITS โดยมีผลลัพธ์การแสดงข้อมูลในตารางดังรูป



```
sql> select * from information_schema.innodb_trx\G
*************************** 1. row ***************************
trx_id: AF01
trx_state: LOCK WAIT
trx_started: 2012-09-03 13:36:08
trx_requested_lock_id: AF01:0:335:10
trx_wait_started: 2012-09-03 13:37:44
trx_weight: 2
trx_mysql_thread_id: 3
trx_query: update theatre set room=room+2 where branch='SiamDis'
trx_operation_state: starting index read
trx_tables_in_use: 1
trx_tables_locked: 1
trx_lock_structs: 2
trx_lock_memory_bytes: 376
trx_rows_locked: 1
trx_rows_modified: 0
trx_concurrency_tickets: 0
trx_isolation_level: REPEATABLE READ
trx_unique_checks: 1
trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
trx_adaptive_hash_latched: 0
trx_adaptive_hash_timeout: 10000
*************************** 2. row ***************************
trx_id: AF00
trx_state: RUNNING
trx_started: 2012-09-03 13:35:48
trx_requested_lock_id: NULL
trx_wait_started: NULL
trx_weight: 3
trx_mysql_thread_id: 2
trx_query: select * from information_schema.innodb_trx
trx_operation_state: NULL
trx_tables_in_use: 0
trx_tables_locked: 0
trx_lock_structs: 2
trx_lock_memory_bytes: 376
trx_rows_locked: 1
trx_rows_modified: 1
trx_concurrency_tickets: 0
trx_isolation_level: REPEATABLE READ
trx_unique_checks: 1
trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
trx_adaptive_hash_latched: 0
trx_adaptive_hash_timeout: 10000
sql>
```

รูปที่ 4.48 ผลลัพธ์การทำงานของคำสั่งอ่านข้อมูลจากตาราง INNODB_TRX

จากรูปจะเห็นได้ว่าตาราง INNODB_TRX จะเก็บข้อมูลในส่วนของทรานแซคชั่นที่กำลังประมวลผลอยู่ในปัจจุบันซึ่งจะมีข้อมูลต่างๆของการใช้ทรัพยากรณ์ต่างๆรวมถึงเรื่องของ Lock wait ของแต่ละทรานแซคชั่นด้วย



```
sql> select * from information_schema.innodb_locks\G
*************************** 1. row ***************************
lock_id: AF01:0:335:10
lock_trx_id: AF01
lock_mode: X
lock_type: RECORD
lock_table: `president`.`theatre`
lock_index: `PRIMARY`
lock_space: 0
lock_page: 335
lock_rec: 10
lock_data: 'SiamDis'
*************************** 2. row ***************************
lock_id: AF00:0:335:10
lock_trx_id: AF00
lock_mode: X
lock_type: RECORD
lock_table: `president`.`theatre`
lock_index: `PRIMARY`
lock_space: 0
lock_page: 335
lock_rec: 10
lock_data: 'SiamDis'
sql>
```


รูปที่ 4.49 ผลลัพธ์การทำงานของคำสั่งอ่านข้อมูลจากตาราง INNODB_LOCKS

จากรูปแสดงให้เห็นว่าตาราง INNODB_LOCKS เก็บข้อมูลของทรานแซคชั่นที่ใช้งานคำสั่งเอสคิวแอลแล้วทำให้เกิด Lock กับตารางที่อยู่ภายในฐานข้อมูล

```
sql> select * from information_schema.innodb_lock_waits;
+-------------------+-------------------+-------------------+-------------------+
| requesting_trx_id | requested_lock_id | blocking_trx_id   | blocking_lock_id  |
+-------------------+-------------------+-------------------+-------------------+
| AF01              | AF01:0:335:10     | AF00              | AF00:0:335:10     |
+-------------------+-------------------+-------------------+-------------------+
1 row in set (0.44 sec)
```

รูปที่ 4.50 ผลลัพธ์การทำงานของคำสั่งอ่านข้อมูลจากตาราง INNODB_LOCK_WAITS

จากรูปแสดงให้เห็นว่าตาราง INNODB_LOCK_WAITS เก็บข้อมูลของทรานแซคชั่นที่กำลังติด Lock ของทรานแซคชั่นอื่น รวมทั้งทรานแซคชั่นที่เป็นต้นเหตุทำให้เกิด Lock ด้วย

4.1.8 ทดสอบการใช้งานคำสั่ง SAVEPOINT และ ROLLBACK TO SAVEPOINT

จากที่กล่าวในส่วนของทฤษฎีการใช้งาน SAVEPOINT และ ROLEBABK TO SAVEPOINT มีส่วนช่วยในเรื่องของการทำงานที่เกี่ยวข้องกับทรานแซคชั่นในส่วนของ Atomicity ดังนั้นจึงจำเป็นต้องทดลองกับระบบจัดการฐานข้อมูลที่อยู่บนคลาวด์ว่าสามารถใช้งานได้จริงหรือไม่ โดยการทดลองจะแสดงดังนี้

1) ดึงข้อมูลทั้งหมดจากตาราง theatre ได้ผลลัพธ์ดังรูป

```
sql> select * from theatre;
+-----------+-----------+
| branch    | room      |
+-----------+-----------+
| Bangkapi  |        16 |
| Bangkean  |        21 |
| Bangna    |        21 |
| Ladprao   |        13 |
| MBK       |        16 |
| Paradise  |        20 |
| Paragon   |        15 |
| Ramkamhaeng |      13 |
| Ratchayodin |      15 |
| Siam      |        10 |
| SiamDis   |        10 |
+-----------+-----------+
11 rows in set (0.72 sec)
```

รูปที่ 4.51 ดึงข้อมูลจากตาราง theatre

2) เริ่มทรานแซคชั่นและทำการแก้ไขข้อมูลในตาราง theatre และทำการใช้งานคำสั่ง SAVEPOINT ดังรูป


```
sql> begin;
0 row(s) affected.
sql> update theatre set room=20 where branch='Bangkapi';
1 row(s) affected.
sql> SAVEPOINT sve_point;
0 row(s) affected.
```

รูปที่ 4.52 เริ่มทรานแซคชั่น , แก้ไขข้อมูล และใช้งาน SAVEPOINT

3) เพิ่มแถวลงไปในตารางและทำการใช้งานคำสั่ง ROLEBACK TO SAVEPOINT โดยกลับไปที่ SAVEPOINT ชื่อ sve_point

```
sql> insert into theatre values('Svpoint1' , 10);
1 row(s) affected.
sql> ROLLBACK TO SAVEPOINT sve_point;
0 row(s) affected.
```

รูปที่ 4.53 เพิ่มแถวลงในตารางและ rollback กลับไปยัง sve_point

4) เพิ่มแถวใหม่ลงในตารางและยืนยันการเปลี่ยนแปลง (COMMIT)

```
sql> insert into theatre values('Svpoint2' , 10);
1 row(s) affected.
sql> commit;
0 row(s) affected.
```

รูปที่ 4.54 เพิ่มแถวลงในตารางและ rollback และยืนยันการเปลี่ยนแปลง

5) ตรวจสอบโดยการใช้คำสั่ง SELECT * FROM theatre

```
sql> select * from theatre;
+-------------+-------------+
| branch      | room        |
+-------------+-------------+
| Bangkapi    |          20 |
| Bangkean    |          21 |
| Bangna      |          21 |
| Ladprao     |          13 |
| MBK         |          16 |
| Paradise    |          20 |
| Paragon     |          15 |
| Ramkamhaeng |          13 |
| Ratchayodin |          15 |
| Siam        |          10 |
| SiamDis     |          10 |
| Svpoint2    |          10 |
+-------------+-------------+
12 rows in set (0.34 sec)
```

รูปที่ 4.55 ดูข้อมูลในตาราง theatre

จะเห็นได้ว่าการใช้งานคำสั่ง SAVEPOINT และ ROLEBACK TO SAVEPOINT สามารถใช้งานได้ปกติในระบบจัดการฐานข้อมูลบนคลาวด์

4.1.9 การใช้ Session และ Cookies ในเว็บแอพพลิเคชั่น

4.1.9.1 Session ของ google app engine

Session ในการพัฒนา web application เป็นการเรียกใช้งานวัตถุ (Object) HttpSession โดยสามารถนำมาใช้ประโยชน์ได้หลายแบบ แต่หลักๆที่นำมาใช้มากคือการใช้เป็นเหมือนกระดาษทดในการเก็บวัตถุ (Object) เพื่อที่หน้าเพจอื่นๆสามารถอ้างถึงวัตถุนั้นๆได้โดยที่ไม่จำเป็นต้องสร้างวัตถุขึ้นมาใหม่ หรือเป็นการอ้างอิงตัวบุคคลที่กำลังทำการติดต่อกับเซอเวอร์ในขณะนั้นอยู่ โดยในการเก็บ Object ลงไปใน session และการนำค่า Object นั้นออกมาจะใช้ method setAttribute() และ getAttribute() ตามลำดับ

Session ใน google app engine โดยมาตรฐานจะถูกตั้งเป็น Disable คือ ไม่สามารถใช้งาน session ได้ ผู้พัฒนา web application สามารถตั้งค่าการใช้งาน session โดยการเพิ่มคำสั่งดังกล่าวที่ไฟล์ appengine-web.xml

คำสั่งที่ 4.10 เปิดการใช้งาน Object Session

```
<sessions-enabled>true</sessions-enabled>
```

และสิ่งที่สำคัญในการใช้งาน session ให้เปรียบเสมือนกระดาษทดเพื่อเก็บ Object ที่จะสามารถเรียกใช้ Object นั้นๆได้ในหน้าอื่นคือ Object ที่จะเก็บใน Session นั้นจะต้อง Implement java.io.Serializable เนื่องจาก App engine จะเก็บข้อมูลของ session ลงใน data store และ memcache ดังนั้นจึงมีความเป็นไปได้ว่าในการพัฒนา web application บน local จะสามารถใช้งาน session ในการเก็บข้อมูล Object บางตัวได้ แต่บน Cloud ของ google App Engine การใช้งาน session ในการเก็บข้อมูล Object ดังกล่าวไม่สามารถทำได้เนื่องจาก Object นั้นไม่ได้ implement Serializable ดังที่จะกล่าวในส่วนต่อไป

ปัญหาการใช้งาน Session ในโปรเจค BookShop

จากที่ได้กล่าวมาแล้วในข้างต้นว่าการเก็บ Object ลงใน HttpSession นั้น Object ดังกล่าวต้องเป็น Object ที่ Implement Serializable ยกตัวอย่าง Object ที่ implement Serializable เช่น String Integer Date ฯลฯ เป็นต้น ทั้งนี้ Object ที่แสดงถึงการเชื่อต่อยัง Database หรือ Connection เป็น Object ที่ไม่ได้ Implement Serializable จึงไม่สามารถเก็บ Object ดังกล่าวลงใน HttpSession ได้ ซึ่งหมายความว่าหากการเชื่อต่อไปยัง Database ครั้งหนึ่ง (1 Connection) เกิดขึ้นที่ Servlet ตัวที่หนึ่ง Servlet ตัวที่สองจะไม่สามารถอ้างถึงการเชื่อมต่อ (Connection) นั้นได้ เนื่องจาก วัตถุ Connection ไม่สามารถเก็บไว้ใน วัตถุ HttpSession ได้ จึงใช้วิธีการติดต่อและดึง

ข้อมูลหรือแก้ไขข้อมูลในดาต้าเบสให้สำเร็จภายใน Servlet ตัวเดียวโดยที่ไม่ต้องเก็บวัตถุ Connection ไว้ใน HttpSession ซึ่งจะมีความหมายว่า Transaction ที่เกิดขึ้นนั้นจะเริ่มและสิ้นสุด Transaction ภายใน Servlet ตัวเดียวเช่นกัน นอกจากนี้ในส่วนของการยืนยันตัวตนจะใช้วิธีการเก็บ Username ของผู้ใช้ลงในวัตถุ HttpSession เนื่องจาก วัตถุ String สามารถเก็บในวัตถุ HttpSession ได้

4.1.9.2 การใช้ Cookie

การใช้งาน Cookie จะมีความคล้ายคลึงกับการใช้งาน HttpSession โดยจะแตกต่างกันที่ข้อมูลที่ถูกเก็บอยู่ใน Cookie นั้นจะถูกเก็บไว้ที่ตัว client (Browser ที่ติดต่อไปยัง Server) และข้อมูลที่จะเก็บได้ใน Cookie นั้นจะต้องเป็นข้อมูลชนิด String เท่านั้น จากที่กล่าวมานี้ทำให้เรานำ Cookie มาใช้ในส่วนของการทำตะกร้าสินค้า หรือ Cart โดยที่จะให้ Cookie แต่ละตัวเก็บค่าของ ISBN ของหนังสือ และจำนวนของหนังสือเล่มนั้นๆ โดยที่ Servlet ทุกตัวสามารถรับข้อมูลที่เก็บอยู่ใน Cookie นั้นมาใช้อ้างอิงในการดึงข้อมูลจากดาต้าเบสได้ โดยเหตุผลที่ไม่ใช้ Cookie ในการเก็บข้อมูลของลูกค้าเช่น Username และ Password เนื่องจากเหตุผลที่ว่า Cookie ถูกเก็บอยู่ใน Browser ของเครื่อง ดังนั้นจึงไม่ปลอดภัยเนื่องจากมี Browser บางตัวที่สามารถแก้ไขข้อมูลใน Cookie ได้ ทำให้ไม่ปลอดภัย

4.2 การพัฒนา Web Application

เป็นหัวข้อการพัฒนา Web Application ตัวอย่างที่สามารถประมวลผลการปฏิบัติทรานแซคชั่นแบบขนานบนระบบฐานข้อมูลบนคลาวด์ของ Google โดยที่จะมีหัวข้อย่อยดังต่อไปนี้

- โครงสร้างของระบบคลาวด์ของ Google ในการพัฒนา Web Application
- สิ่งที่จำเป็นต้องมีในการพัฒนา Web Application เพื่อติดต่อกับระบบฐานข้อมูลบนคลาวด์
- หลักการและการใช้งาน Web Application บนคลาวด์
- ข้อจำกัดของ Application ที่ปฏิบัติบนคลาวด์เมื่อเปรียบเทียบกับ Application ปกติ
- ภาพตัวอย่างประกอบของ Web Application

4.2.1 โครงสร้างของระบบคลาวด์ของ Google ในการพัฒนา Web Application

ก่อนที่จะพัฒนา Web Application ที่ติดต่อกับฐานข้อมูลบนคลาวด์นั้น จำเป็นที่จะต้องรู้และเข้าใจถึงโครงร้างการทำงานต่างของ Web Application บนคลาวด์เสียก่อน ซึ่ง Google มี

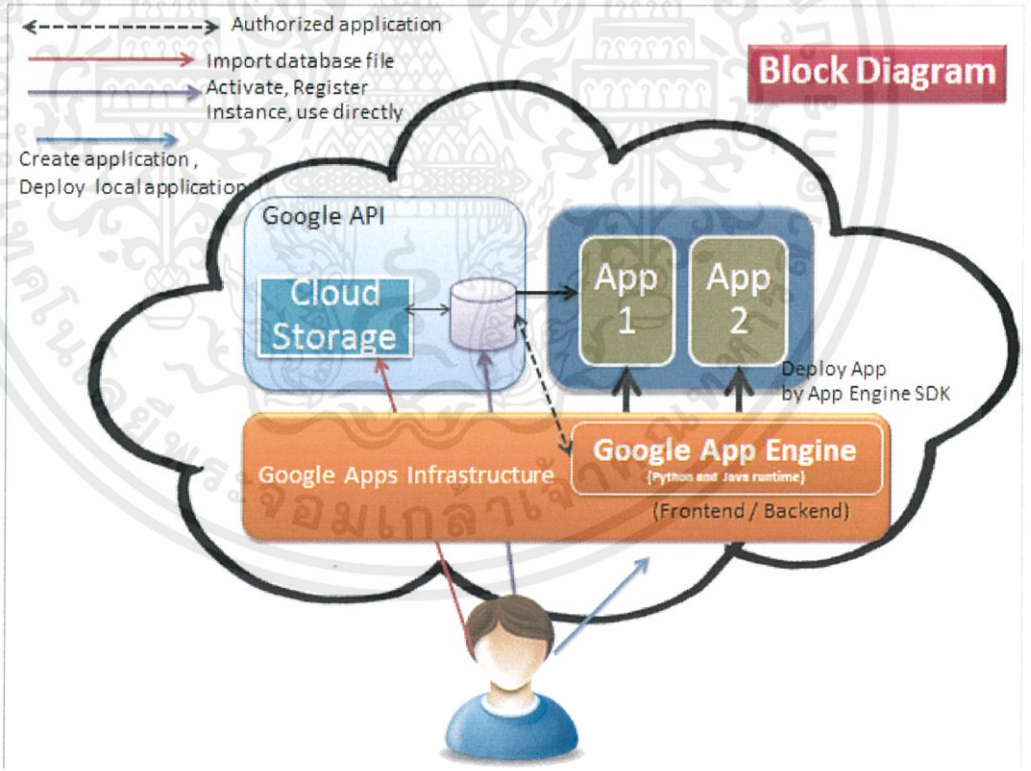
Service ต่างๆที่จำเป็นในการทำ Web Application ที่สามารถเชื่อมต่อกับฐานข้อมมูล Google Cloud SQL ได้ดังนี้

1. Google Cloud SQL เป็นส่วนที่สำคัญที่สุดคือเป็นระบบฐานข้อมูลบนคลาวด์ โดยที่ Google Cloud SQL จะใช้ระบบจัดการฐานข้อมูลของ MySQL version 5.5

2. Google Cloud Storage เป็น Service ที่ผู้ใช้งานสามารถฝากไฟล์ไว้บนคลาวด์ของ Google ได้ โดยที่ไฟล์ที่ฝากไว้สามารถติดต่อกับฐานข้อมูลได้ อ้างอิงจากที่เคยกล่าวไว้แล้วในบทก่อน

3. Google App Engine เป็น Service ที่ใช้ฝาก Web Application ที่พัฒนาแล้วลงไปปฏิบัติบนคลาวด์ได้ และเนื่องจากเป็น Platform as a Service จึงส่งผลให้ผู้ใช้บริการไม่จำเป็นต้องสร้าง Cloud Infrastructure ขึ้นมาเพื่อนำ Application ไปปฏิบัติการ แต่เพียงแค่ Deploy Web Application ลงไปบนคลาวด์เท่านั้น โดยที่ Service ดังกล่าวสามารถติดต่อกับฐานข้อมูลของ Google ได้เช่นกัน

โดยที่การติดต่อกันระหว่าง Service ต่างๆบน Google Cloud จะแสดงดังรูป



รูปที่ 4.56 ภาพการติดต่อกันระหว่างแต่ละ Service ของ Google

4.2.2 สิ่งที่จำเป็นต้องมีในการพัฒนา Web Application เพื่อติดต่อกับระบบฐานข้อมูลบนคลาวด์

หลังจากที่ได้เรียนรู้และเข้าใจถึงการทำงานและการติดต่อสื่อสารกันระหว่างแต่ละ Service ของ Google Cloud SQL แล้ว สิ่งที่จำเป็นต่อไปคือผลิตภัณฑ์ที่สนับสนุนกับการทำงานต่างๆของ Service ของ Google โดยที่ผู้ใช้สามารถพัฒนา Web Application โดยภาษา JAVA และ Python ซึ่งทางผู้ทดลอง พัฒนาด้วยภาษา JAVA โดยใช้โปรแกรม eclipse เนื่องจาก Google App Engine มี Plugin ที่สนับสนุนโปรแกรม eclipse ซึ่ง Plugin ดังกล่าวช่วยให้ผู้พัฒนาสามารถสร้าง , จัดการ รวมไปถึงการ Deploy Project Web Application ของตนเองขึ้นไปสู่ Google App Engine ได้โดยง่าย

4.2.3 หลักการและการใช้งาน Web Application บนคลาวด์

ในหัวข้อนี้จะเป็นการอธิบายรายละเอียดต่างๆที่จำเป็นในการพัฒนา Web Application เพื่อให้สามารถติดต่อกับระบบฐานข้อมูลบนคลาวด์ได้โดยในเนื้อหาจะรวมไปถึง ส่วนของโค้ดที่จำเป็นในการติดต่อกับฐานข้อมูล และ Model ที่ผู้พัฒนาใช้ในการพัฒนาซึ่งเคยได้กล่าวไปแล้วบางส่วนในหัวข้อก่อนหน้า

ในการพัฒนา Web Application ให้ติดต่อกับระบบฐานข้อมูลบนคลาวด์ได้นั้น จำเป็นจะต้องมี Driver ในการเชื่อมต่อกับฐานข้อมูล โดยโค้ดในการ register Driver ดังกล่าวให้ติดต่อกับ Google Cloud SQL เป็นดังนี้

คำสั่งที่ 4.11 การ Register Driver สำหรับเชื่อมต่อกับระบบฐานข้อมูลบนคลาวด์

```
DriverManager.registerDriver(new AppEngineDriver());
```

ซึ่งในการจะใช้งานโค้ดดังกล่าว ผู้ใช้จำเป็นต้อง Import library ของ AppEngine ก่อนโดยการพิมพ์คำสั่ง

คำสั่งที่ 4.12 Import Driver ที่จำเป็นในการเชื่อมต่อ

```
Import com.google.appengine.api.rdbms.AppEngineDriver;
```

และในการติดต่อกับฐานข้อมูล จำเป็นจะต้องระบุ Directory ที่อยู่ของระบบฐานข้อมูลบนคลาวด์ ซึ่งจำเป็นต้องระบุ Instance name และ ชื่อฐานข้อมูลลงไปด้วย การสร้าง Instance name และสร้างฐานข้อมูลได้ระบุไปแล้วในบทก่อนหน้า

คำสั่งที่ 4.13 ตัวอย่างการเชื่อมต่อกับระบบฐานข้อมูลบนคลาวด์

```
Connection c =
DriverManager.getConnection("jdbc:google:rdbms://rast-
kanyaporn:testproject/mydb");
```

ซึ่ง Object Connection ที่สร้างขึ้นดังโค้ดตัวอย่างนี้สามารถนำมาเรียกใช้ในการ Query , Update , Insert หรือ Delete ข้อมูลที่อยู่บนระบบฐานข้อมูลได้ นอกจากนี้ยังสามารถเรียกใช้ method ในการปฏิบัติคำสั่งทรานแซคชั่นได้ดังตัวอย่างต่อไปนี้

คำสั่งที่ 4.14 ตัวอย่างการ Query ข้อมูล

```
Statement St= connection.createStatement();
ResultSet rs = St.executeQuery(SQL);
```

คำสั่งที่ 4.15 ตัวอย่างการ Update ข้อมูล

```
Statement St= connection.createStatement();
ResultSet rs = St.executeUpdate(SQL);
```

คำสั่งที่ 4.16 การเริ่มต้นทรานแซคชั่น

```
connection.setAutoCommit(false);
```

คำสั่งที่ 4.17 การยืนยันการเปลี่ยนแปลงที่เกิดขึ้นโดยทรานแซคชั่น

```
connection.commit();
```

คำสั่งที่ 4.18 การยกเลิกการเปลี่ยนแปลงที่เกิดขึ้นโดยทรานแซคชั่น

```
connection.rollback();
```

ซึ่งจากตัวอย่างเป็นเพียงแค่ตัวอย่างการใช้งาน method ง่ายๆในการติดต่อกับฐานข้อมูลเท่านั้นซึ่งในความเป็นจริงแล้วยังมี method ให้ใช้งานได้อีกมากมาย ขึ้นอยู่กับความต้องการของผู้ใช้งานว่าจะต้องการให้การปฏิบัติคำสั่งบนฐานข้อมูลมีระดับความถูกต้องเพียงใด

และจากการศึกษา Web Application ที่พัฒนาเพื่อให้ติดต่อกับระบบฐานข้อมูลส่วนใหญ่นั้นจะเห็นได้ว่ามีช่องโหว่คือมีการเก็บ Username และ Password ที่สามารถเชื่อมต่อกับระบบฐานข้อมูลได้ไว้ในตัวโค้ด ซึ่งถือว่าไม่ใช่ยุทธวิธีที่ดีที่สุดในการใช้งาน ทางผู้พัฒนาจึงปรับการใช้งานดังกล่าวโดยการสร้าง Username ใหม่และ Grant สิทธิ์ต่างๆในการเข้าถึงระบบฐานข้อมูลได้ แล้วใช้ Username Password ดังกล่าวเป็น Authentication การเชื่อมต่อฐานข้อมูล ซึ่งจะสามารถช่วยลด

ช่องโหว่ในการฝัง Username และ Password ลงในโค้ดโปรแกรมไปได้ระดับหนึ่ง และยังช่วยให้ใช้งาน DBMS ได้อย่างมีประสิทธิภาพเพิ่มมากขึ้นอีกด้วย

นอกจากนี้ในการพัฒนา Web Application นั้นผู้ใช้งานเลือกใช้ Model ในการพัฒนาคือ MVC Model ที่จะแยกการทำงานของ Web Application ออกเป็นส่วนๆที่แตกต่างกัน เช่นส่วนของการติดต่อระบบฐานข้อมูล , ส่วนของการแสดงผล Output และส่วนของการควบคุมการทำงานของการติดต่อและการแสดงผล ดังที่ได้เคยอธิบายไว้แล้วในบทก่อนหน้า

4.2.4 ข้อจำกัดของ Application ที่ปฏิบัติบนคลาวด์เมื่อเปรียบเทียบกับ Application ปกติ

จากการทดลองของผู้พัฒนา Web Application ทำให้ได้รับรู้ว่า Web Application แบบปกติกับ Web Application ที่ปฏิบัติบน Cloud Platform จะมีความแตกต่างกันอยู่บางประการซึ่งผู้ที่ต้องการจะพัฒนา Web Application บน Local Site ของตนเองแล้วค่อยนำไปลงบน Cloud ของ Google จำเป็นต้องทราบและทำความเข้าใจให้ดีเสียก่อน โดยความแตกต่างและข้อจำกัดต่างๆของ Cloud Application คือ

1) มาตรฐานเวลาที่ใช้บน Application เนื่องจาก Server ของ Google Cloud นั้นกระจัดกระจายอยู่ทั่วโลก ทั้งนี้แต่ละ Server ก้จะมีมาตรฐานเวลาที่แตกต่างกันตามแต่ละพื้นที่ ดังนั้นผู้ใช้ที่ต้องการทราบเวลาปัจจุบัน (โดยอาจจะใช้ method ต่างๆในการหา) จำเป็นตั้งปรับ format ให้ตรงกับมาตรฐานเวลาตามประเทศของตนเอง
2) การใช้งาน Object Session ของ Google App Engine ดังที่ได้อธิบายไปในบทก่อนหน้าที่ว่า Google App Engine นั้นเก็บข้อมูลบน data store และ memcache ส่งผลให้การใช้งาน Object Session ในการเก็บ Object ต่างๆนั้น Object ที่จะเก็บต้อง Implement java.io.Serializable เนื่องจากต้องจัด format ของการเก็บไฟล์ ดังนั้น Object ของ Connection จึงไม่สามารถเก็บลงใน Object Session ได้ ส่งผลให้เกิดปัญหาบางประการในการปฏิบัติทรานแซคชั่นเป็นเวลายาวนานและในทรานแซคชั่นนั้นจำเป็นต้องปฏิบัติข้ามหน้าเพจ (ใช้ Servlet หลายตัวในการปฏิบัติหนึ่งทรานแซคชั่น) ทางผู้พัฒนาจึงได้คิดวิธีแก้ไขโดยไม่ทำการส่ง Objection Connection ข้ามหน้าเพจ แต่ใช้การทำทรานแซคชั่นภายใน Servlet ตัวเดียว และให้ Servlet แต่ละตัวทำการเชื่อมต่อกับฐานข้อมูลใหม่ทุกครั้งที่ต้องการข้อมูลจากระบบฐานข้อมูลโดยในการเชื่อมต่อจะกำหนด Username ที่มีสิทธิในการเข้าถึงข้อมูลส่วนที่จำเป็นเท่านั้น โดยที่การปฏิบัติด้วยยุทธวิธีดังกล่าวมีข้อดีตรงที่ในการปฏิบัติทรานแซคชั่นหนึ่งๆ ถ้าหากทำอยู่ภายใน Servlet เดียวนั้น จะมีโอกาสที่ อีกทรานแซคชั่นจะเข้าถึง Data Item เดียวกันค่อนข้าง

น้อย เนื่องจากทรานแซคชั่นจะเริ่มและจบลงอย่างรวดเร็ว (Commit เร็ว) และจะไม่เกิดกรณีที่เป็น Long Duration Transaction

โดยถ้าหากผู้ที่จะพัฒนา Web Application บน Local Site ก่อนแล้วเมื่อสำเร็จจึงนำไป Deploy ลงบนคลาวด์ของ Google นั้น ทางผู้จัดทำรายงานมีข้อแนะนำว่าควรใช้ DBMS ของผลิตภัณฑ์เดียวกันและควรเป็นรุ่นเดียวกันด้วย เนื่องจากการใช้ผลิตภัณฑ์คนละผลิตภัณฑ์ หรือแม้แต่กระทั่งผลิตภัณฑ์เดียวกันแต่คนละรุ่น ก็จะสามารถทำให้เกิดปัญหาขึ้นได้หลังจากที่ Deploy ขึ้นไปบน Google App Engine เรียบร้อยแล้ว

บทที่ 5
บทสรุปและข้อเสนอแนะ

การศึกษาการประมวลผลทรานแซกชั่นบนระบบคลาวด์นั้น ได้มีการประยุกต์มาจากความรู้พื้นฐานของระบบฐานข้อมูล และ ระบบจัดการฐานข้อมูล โดยนำความรู้เกี่ยวกับทรานแซกชั่นของระบบจัดการฐานข้อมูลมาใช้ ให้เป็นไปตามลักษณะ และประเภทของงาน ซึ่งปัจจุบันเป็นกลไกสำคัญอันหนึ่ง ที่จัดการโดยระบบจัดการฐานข้อมูลทุกผลิตภัณฑ์

ในการดำเนินการทรานแซกชั่นดังกล่าว จากการศึกษา จะพบว่ารูปแบบของการใช้งาน เช่น การตั้งระดับไอโซเลชั่น มีผลต่อผลลัพธ์ และ ความถูกต้องของคำตอบ ขึ้นกับว่าเน้นความละเอียดและเข้มงวดในงานแต่ละงานมากน้อยเพียงใด โดยการใช้งานทรานแซกชั่นดังกล่าว สามารถใช้งานบนระบบฐานข้อมูลปกติได้เช่นกัน โดยวัตถุประสงค์ของโครงงานนี้เพื่อทดลองการใช้ระบบฐานข้อมูลบนระบบคลาวด์ และ ศึกษาความแตกต่างระหว่างทั้งสองระบบ

5.1 บทสรุป

การทำงานของระบบฐานข้อมูลบนคลาวด์จะมีการทำงานรวมถึงหน้าที่ต่างๆคล้ายคลึงกับการทำงานของระบบฐานข้อมูลแบบปกติเป็นส่วนใหญ่ เกือบทั้งหมด โดยใน Google Cloud SQL จะมีการทำงานบางอย่างที่ระบบจัดการฐานข้อมูลบนคลาวด์ไม่สนับสนุน หรือ ไม่รองรับ รวมถึงเรื่องของเครื่องมือที่ช่วยเหลือในการทำงานแบบต่างๆของระบบคลาวด์ยังมีได้ไม่เท่ากับระบบฐานข้อมูลแบบปกติ แต่จะทดแทนด้วยความสะดวกในการใช้งานรวมไปถึงลดเวลาในการดูแลรักษาระบบอยู่เสมอ เนื่องจากระบบฐานข้อมูลบนคลาวด์จะรับประกันในส่วนความปลอดภัยไปจนถึงการสำรองข้อมูลต่างๆ อีกทั้งการขยายตัวของระบบ (Scalability) ยังสามารถทำได้ง่ายกว่าระบบฐานข้อมูลแบบธรรมดา นอกจากนี้ยังรองรับการทำงานควบคู่กับแอพพลิเคชั่นต่างๆเพื่อให้สามารถใช้งานระบบจัดการฐานข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น ข้อแตกต่างที่เพิ่มเติมและความสามารถพิเศษที่เพิ่มขึ้นมา บน Google cloud SQL สามารถสรุปได้ดังต่อไปนี้

ความสามารถพิเศษที่เพิ่มเติม

- ✓ ความสามารถในการเป็นเจ้าของระบบฐานข้อมูล MySQL บนระบบคลาวด์
- ✓ ค่าใช้จ่ายของการใช้งานมีหลายรูปแบบ ขึ้นกับการใช้งาน Per use จะเป็นการจ่ายเฉพาะเวลาที่เข้าถึงข้อมูลในระบบคลาวด์

- ✓ ในการใช้งานเบื้องต้นของระบบคลาวด์บางฟังก์ชั่นสามารถเริ่มการใช้งานได้โดยไม่ต้องเสียค่าใช้จ่าย
- ✓ Google Cloud SQL instances สามารถขยายขนานของแรม ได้ถึง 16 GB และ 100 GB สำหรับการจัดเก็บข้อมูล
- ✓ SQL prompt เป็นลักษณะพิเศษภายใน Google APIs Console ที่อนุญาตให้มีการติดต่อระบบฐานข้อมูลได้โดยตรงผ่านเว็บ
- ✓ ข้อมูลที่อยู่ในระบบคลาวด์ ถูกเก็บไว้ที่ Data center หลากหลายพื้นที่ทั่วโลก ป้องกันข้อมูลที่อาจสูญหายได้
- ✓ ,มีการสำรองข้อมูลแบบทั้ง Synchronous และ Asynchronous ตามภูมิภาค
- ✓ การ Import หรือ Export ฐานข้อมูล จะใช้คำสั่ง mysqldump
- ✓ รองรับการใช้ภาษา Java และ Python
- ✓ สามารถติดตั้ง Command line tool ซึ่งช่วยให้ ผู้พัฒนาหรือผู้ดูแลฐานข้อมูล สามารถติดต่อกับฐานข้อมูลได้โดยตรง ไม่ต้องผ่านเว็บไซต์ โดยลักษณะคล้ายคลึงกับ Command prompts บน Windows

ข้อจำกัด

- ✗ การจัดเก็บข้อมูลมีขนาด instances ขยายได้สูงสุด 100 GB
- ✗ ไม่รองรับ User defined function
- ✗ คำสั่ง SQL ต่อไปนี้ ไม่รองรับใน Google cloud SQL ได้แก่
  - LOAD DATA INFILE
  - SELECT ... INTO OUTFILE/DUMPFILE
  - INSTALL/UNINSTALL PLUGIN ...
  - CREATE FUNCTION ...
  - LOAD_FILE()
- ✗ ฟังก์ชั่น MySQL ที่ไม่รองรับใน Google cloud SQL ได้แก่
  - SHA2()

ในทางปฏิบัติจริง แม้ข้อจำกัดที่เกิดขึ้นจากระบบฐานข้อมูลบนคลาวด์จะทำให้ผู้พัฒนาไม่สามารถ ใช้คำสั่งตามปกติได้ แต่ในทางปฏิบัติคำสั่งเหล่านั้น สามารถใช้ฟังก์ชั่น หรือ คำสั่งอื่นๆ เข้ามาเพื่อทดแทนได้ด้วยวิธีที่แตกต่างจากวิธีเดิม เช่น คำสั่ง LOAD DATA INFILE แม้ไม่สามารถ ทำ

การ IMPORT ตาราง หรือข้อมูลได้ตรงๆ ผ่าน Google cloud SQL แต่ ใน Interactive website ของ Google รองรับ การ IMPORT ไฟล์ที่มาจาก Google cloud storage ทำให้วิธีการอาจเปลี่ยนไปได้ แต่ได้ผลลัพธ์สุดท้ายที่เหมือนกัน

5.1.1) สรุปการทดลองตลอดโครงงาน 1 และ 2

- ทดลองการใช้งานของ กูเกิลคลาวด์โดยรวม และเฉพาะผลิตภัณฑ์
- ทดลองใช้คำสั่งพื้นฐาน ในการค้นข้อมูล และดำเนินการกับฐานข้อมูลอื่นๆ เช่น INSERT, SELECT, DELETE, UPDATE, ALTER, MODIFY หรือ ใช้คำสั่งการ SELECT ร่วมกับ BUILD FUNCTION , GROUP BY,ORDER BY ,HAVING ได้อย่างปกติ รวมถึงการใช้ SUBQUERY, JOIN, หรือการสร้าง VIEW เพื่อเรื่องการกำหนดสิทธิในการมองเห็น ก็สามารถทำได้เหมือนระบบฐานข้อมูลปกติ
- ทดลองการ IMPORT ข้อมูลจาก MySQL บนเครื่องธรรมดา ลงสู่คลาวด์
- ศึกษาผลิตภัณฑ์ 3 ตัวของกูเกิล ได้แก่ Google cloud SQL, Google cloud Storage และ Google App Engine
- ศึกษา Storage engine ของ MySQL และ เลือกใช้ InnoDB engine ในการทำทรานแซกชั่น
- การออกแบบการทดลองที่สอดคล้องกับทฤษฎี ในการทดลองและแอพพลิเคชั่น
- ทดสอบการประกาศทรานแซกชั่น และ การใช้งานทรานแซกชั่นบนคลาวด์
- การศึกษาและทดสอบ ACID Properties บนกูเกิลคลาวด์
- การออกแบบและทดลองปัญหา The 4 concurrency control problems บน Isolation level ที่แตกต่างกัน
- ตรวจสอบการทำงานภายในของ ระบบจัดการฐานข้อมูล
- การใช้งาน Nested Transaction กรณี Long duration transaction
- การประกาศและบังคับใช้ Integrity Constraint
- พัฒนาแอพพลิเคชั่นตัวอย่างที่สนับสนุนการทำงานของ Transaction
- ศึกษากลไก การจัดการ concurrency control และการตรวจสอบการ Lock ของ DBMS ด้วย Performance tool ของ MySQL
- การประกาศสิทธิในการเข้าถึง และตรวจสอบความสามารถของผู้ใช้งาน

การตั้งค่า System variables ต่างๆ บน Google Cloud SQL

- ศึกษาระดับไอโซลชั่นที่แตกต่างกัน เกี่ยวกับกลไกการล็อคภายในที่แตกต่างกัน เปรียบเทียบกันทฤษฎี
- ศึกษา กลไก Mutiversion และ 2 phase locking protocol ซึ่งเป็นกลไกหลักในการบังคับผลลัพธ์ให้มีความถูกต้อง
- ศึกษาการใช้ CURSOR ใน Java และทดสอบปัญหาข้อ 3 , 4 ด้วยการเลื่อน CURSOR เพื่อดูการเปลี่ยนแปลงของตาราง
- ศึกษาการใช้เครื่องมือช่วยในการออกแบบ ER (MySQL workbench)
- ศึกษาและประยุกต์ใช้งาน TRIGGER เพื่อใช้ในการกำหนด กฎของความถูกต้องในการพัฒนาเว็บแอพพลิเคชั่น แทนการใช้คำสั่ง CHECK
- พัฒนาเว็บแอพพลิเคชั่นต้นแบบ จำนวน 2 เว็บ สามารถแบ่งได้เป็น
    - Online bookstore web application เป็นเว็บไซต์ร้านหนังสือออนไลน์ ใช้ระดับไอโซเลชั่นเป็น Repeatable read เป็นตัวดำเนินการทรานแซกชั่น
    - Airplane reservation web application เป็นเว็บไซต์จองตั๋วเครื่องบินออนไลน์ ใช้ระดับไอโซเลชั่นเป็น Repeataable read เช่นกัน

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

ปัญหาและอุปสรรคที่พบจากการทำงานมีอยู่หลายประเด็นโดยสามารถแสดงให้เห็นถึงปัญหาและแนวทางแก้ไขในแต่ละข้อดังต่อไปนี้

1) ปัญหาเรื่องการพัฒนาเว็บแอพพลิเคชั่น

เนื่องด้วยนักศึกษายังมีความรู้ในส่วนของการพัฒนาเว็บแอพพลิเคชั่นไม่เชี่ยวชาญนัก ส่งผลให้การพัฒนาเว็บแอพพลิเคชั่นเป็นไปได้ค่อนข้างช้า โดยแนวทางแก้ไขคือ สอบถามข้อมูลรวมไปถึงเทคนิคการพัฒนาเว็บแอพพลิเคชั่นจากผู้เชี่ยวชาญเพื่อนำมาปรับใช้กับโครงงาน

2) ความไม่สอดคล้องกันของการทำงานตามทฤษฎีและผลิตภัณฑ์

การทำงานบางอย่างของระบบจัดการฐานข้อมูลที่อยู่ในทฤษฎีที่ศึกษากับการทำงานจริงของตัวผลิตภัณฑ์ที่ทดสอบไม่สอดคล้องกัน ซึ่งมักจะมีบางส่วนแตกต่างกันอยู่ทำให้ผลลัพธ์ของ

การทดสอบบางอย่างจะไม่ตรงกับผลลัพธ์ในทางทฤษฎี แนวทางแก้ไขคือบันทึกทุกผลการทดลองเปรียบเทียบกับผลทางทฤษฎีอย่างละเอียดเพื่อใช้เป็นข้อมูลในการอ้างอิงการทำงานของระบบจัดการฐานข้อมูล

3) การทดสอบทฤษฎีบางข้อ

ในการทดสอบทฤษฎีบางข้อ ยกตัวอย่างเช่นปัญหา The phantom phenomenon นั้นไม่สามารถทดลองได้โดยการทดสอบแบบทั่วไปทำให้การทดสอบเป็นไปด้วยความล่าช้า ส่งผลให้จำเป็นต้องมีการประยุกต์การใช้งานของ CURSOR เพื่อช่วยให้ได้ผลลัพธ์การทดลองที่แม่นยำ

4) การวางแผนการทดลอง

ในการออกแบบและวางแผนการทดลองบางหัวข้อยังไม่สอดคล้องกับการทำงานตามทฤษฎี ส่งผลให้การทดลองดังกล่าวผลลัพธ์ไม่ตรงประเด็นกับผลการทดลองที่คาดหวังซึ่งมีส่วนให้เป็นผลการทดลองที่ผิดพลาด เช่น การพัฒนาแอพพลิเคชั่นแรก เพื่อทดลองในส่วนการทำ Authentication พบว่า ในการเข้าระบบเพื่อทำเว็บแอพพลิเคชั่น เราควรเน้นการยืนยันตัวตนผ่าน DBMS แทนการฝังโค้ดของ Admin ไว้ที่แอพพลิเคชั่น แนวทางแก้ไขคือ หลังจากทดลองแล้วต้องตรวจสอบความถูกต้องหรือประเด็นของการการทดลองครั้งนั้นๆอยู่เสมอโดยอาจใช้วิธีในการสอบถามอาจารย์ที่ปรึกษา เพื่อน และศึกษา best practice เพื่อให้ได้ผลการทดลองที่ถูกต้องและแม่นยำที่สุด

5) การใช้ Integrity constraint

ด้วยคำสั่ง CHECK condition ไม่สามารถตรวจสอบค่าอินพุทที่ได้รับจากผู้ใช้งานจริง การแก้ไขปัญหา จะเปลี่ยนการใช้งานจาก CHECK condition เป็นการใช้ Trigger ซึ่งเป็นวัตถุภายในฐานข้อมูล ที่เก็บไว้ที่ฝั่งเซิฟเวอร์ ใช้ดัก DBMS ได้

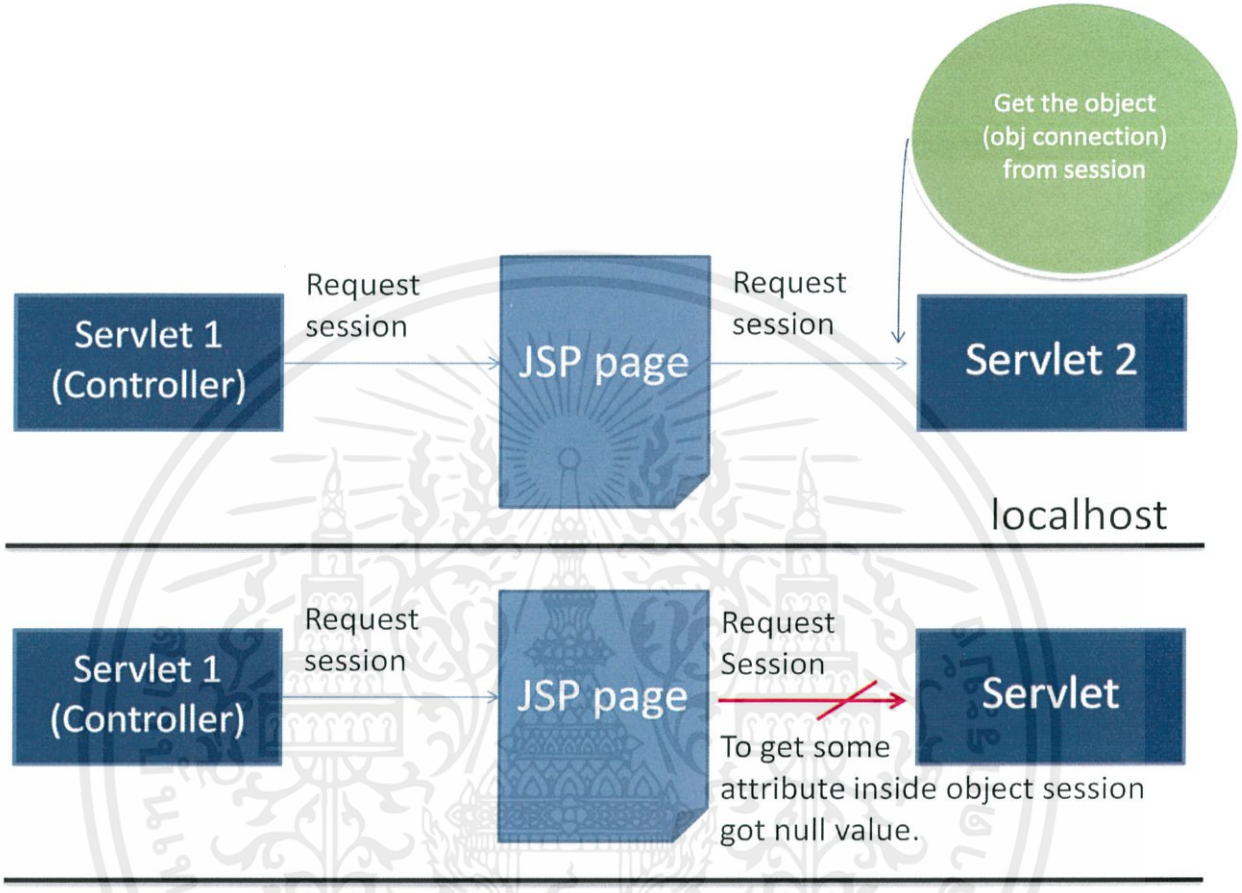
6) การใช้ Cookie

Browser เมื่อทำงานร่วมกับ Cookies แล้ว พบว่า จะมีค่า Cookies ซึ่งเป็นหนังสือที่ซื้อคราวก่อน ค้างอยู่ใน Browser ทำให้ข้อมูลในตะกร้าไม่ถูกต้อง ดังนั้นสามารถแก้ไขปัญหาโดยเนื่องจากเป็นข้อจำกัดของ Browser ที่เก็บค่า Cookies ค้างไว้ ดังนั้น จึงต้องทำการปิด Browser ก่อนการเริ่มซื้อสินค้าใหม่ทุกครั้ง หรือทำการเคลียร์ค่าในคุ้กกี้ หลังปิดบราวน์เซอร์

7) ปัญหาจากการใช้ Google App Engine

การพัฒนาโปรแกรมด้วย Localhost กับ Cloud มีข้อแตกต่างกัน ตรงที่หากเป็นที่ Server ใน Localhost จะมี Cookies เริ่มต้นมาให้หนึ่งค่า แต่ที่ Cloud ไม่มี Cookies เริ่มต้นมาให้ ทางแก้ไขคือ enable session หรือต้องมีการปรับ Code ที่ใช้พัฒนาเล็กน้อย เพื่อให้ดึงค่า Cookies ที่ถูกต้องและไม่เกิดปัญหา NumberFormatException

8) ปัญหาจากการใช้ Google App Engine เรื่อง session



รูปที่ 5.1 ภาพปัญหาการใช้งาน Object Session บนคลาวด์

Session ในคลาวด์ไม่สามารถเก็บค่า object ที่ไม่ implement serializable ได้ ในขณะที่ Localhost สามารถทำได้ ทำให้เป็นปัญหาในการทำ Web application เพราะ session สามารถช่วยในการเก็บ Object connection ซึ่งใช้ในการ ยืนยันได้ว่าคือผู้ใช้งานท่านใด ในแนวทางการแก้ปัญหา จากเรื่องจาวา ต้องทำการหาวิธีการ implement serializable ตัว object connection หรือ Object ที่ไม่ Serializable เช่น Connection

ดังนั้นในการพัฒนาแอพพลิเคชั่นเพื่อทำทรานแซกชั่น การติดต่อฐานข้อมูลจะดำเนินการให้เสร็จสิ้นภายใน Servlet ตัวเดียว และไม่ส่งข้ามไปที่หน้าอื่น เพื่อดำเนินทรานแซกชั่นต่อ นั่นคือการแก้ปัญหาให้กับการทำทรานแซกชั่นของผู้ใช้งานคนหนึ่ง ในช่วงเวลาหนึ่ง ส่วน การสร้าง Cart ของผู้

ซื้อ เพื่อระบุว่าเป็นผู้ใช้งานคนใดให้ใช้ Cookies ซึ่งถูกจัดเก็บไว้บนเครื่องคอมพิวเตอร์ด้านฝั่งของผู้ใช้บริการ โดยเป็นข้อมูลที่เป็นชื่อของสินค้า แต่จะไม่ทำการใส่ราคา หรือข้อมูลสำคัญลงใน cookies เพราะเนื่องจาก ในทาง security ค่าในคุ้กกี้สามารถทำการแก้ไขได้โดยผู้ใช้งาน

5.3) ข้อเสนอแนะเพิ่มเติม: วิธีการย้าย Web application จาก Localhost ลง Google App engine (Portability)

1) Copy ไฟล์หลักที่สำคัญจาก Localhost มาวางยัง Workplace ใน Java eclipse
2) เซ็ต Application ID ของ Project ที่สร้างขึ้น
3) Import **com.google.appengine.api.rdbms.AppEngineDriver;**

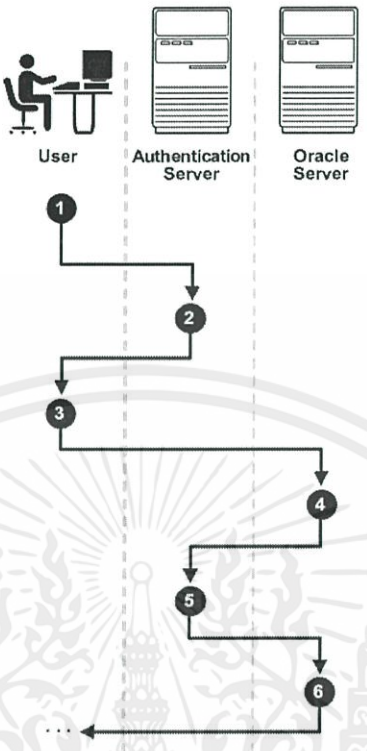
คำสั่งที่ 5.1 คำสั่งที่ใช้ในการเชื่อมต่อระบบฐานข้อมูลบนคลาวด์

```
DriverManager.registerDriver(new AppEngineDriver());
Connection c  =
DriverManager.getConnection("jdbc:google:rdbms://rast-
kanyaporn:testproject/president",user,pass);
```

4) จากปัญหาในข้อที่สาม ทำให้มีการเปลี่ยนแปลงการรับค่าของ Cookies เริ่มแรกให้เหมาะสมกับการใช้งานบนคลาวด์

5.4 แนวทางในการพัฒนาต่อ

ในการพัฒนาโครงงานต่อจะศึกษาพิ่มเติมในส่วนของแอพพลิเคชั่นที่ต้องประมวลผลทรานแซคชั่นที่แสดงให้เห็นผลลัพธ์ที่ถูกต้องและชัดเจนยิ่งขึ้น รวมไปถึงศึกษาทฤษฎีของระบบจัดการฐานข้อมูลเพิ่มเติมเพื่อนำมาทดสอบและตรวจสอบกับระบบจัดการฐานข้อมูลบนคลาวด์ โดยแบ่งหัวข้อของแนวทางในการพัฒนาดังต่อไปนี้



รูปที่ 5.2 ภาพตัวอย่าง หลักการของ ระบบ Authentication server ใน Oracle

1. พัฒนาเว็บแอพพลิเคชั่นให้มีความสมบูรณ์เพิ่มเติม เพื่อเพิ่มเรื่อง Security เช่นการพัฒนาระบบ Authentication อาจใช้ authentication server แทนการใช้แบบ native authentication เพื่อตระหนักเรื่องความปลอดภัยที่มากขึ้น และป้องกันภัย ซึ่งอาจพยายามเจาะเข้าระบบคลาวด์ ผ่านเครือข่ายเน็ตเวิร์ก โดยขั้นตอนภาพรวม จะเริ่มจากให้ผู้ใช้งานทำการระบุตัวตนโดยการส่งรหัสผ่าน หลังจากนั้นตัว Authentication server จะทำการตรวจสอบและส่ง Ticket หรือหนังสือรับรองกลับมาให้ผู้ใช้งาน เมื่อผู้รับ จะทำการแนบหนังสือรับรองไปกับรีเควสอื่นๆ เช่น การเชื่อมต่อไปยัง Database โดยที่เซิฟเวอร์จะทำการส่งหนังสือรับรองนั้นกลับไปยัง Authentication server โดยถ้า Server รับหนังสือรับรองนั้น เซิฟเวอร์จะทำการแจ้งบอก Database server และ รับรองผู้ใช้งาน เป็นต้น
2. ศึกษาเพิ่มเติม เรื่องของ Database security บนระบบคลาวด์ให้ละเอียดมากยิ่งขึ้น
3. ศึกษาการนำ ระบบ Cloud computing ไปใช้ต่อในกรณี ทำเป็น Distributed database หรือ Backup site ในส่วนของระบบฐานข้อมูล
4. ศึกษาความแตกต่างระหว่างเทคโนโลยีบนคลาวด์ ที่มีการใช้งานแบบ Database system เช่น Google cloud SQL กับ ระบบที่ไม่ใช่ Database เช่น BigQuery ซึ่งไม่รองรับคุณสมบัติการจัดการฐานข้อมูล ไม่รองรับ update หรือ delete แต่มีประโยชน์สำหรับการ

วิเคราะห์ข้อมูลจำนวนมาก ซึ่งไม่ได้เน้นการเปลี่ยนแปลงค่าแบบการทำทรานแซกชั่น ที่อาจมีการเปลี่ยนแปลงข้อมูลได้ตลอดเวลา ซึ่งเป็นที่นิยมศึกษากันในปัจจุบัน ทั้งนี้เพื่อนำไปสู่การประยุกต์ใช้ในงานแต่ละงานให้เหมาะสม

บรรณานุกรม

[1] Abraham Silberschatz, Henry F. Korth, S. Sudarshan. **Database System Concept Fifth Edition.** Singapore : McGraw-Hill Education, 2006.

[2] วรเศรษฐ สุวรรณิก, ทศพล ธนะทิพานนท์. **JSP สำหรับงาน E-Commerce.** กรุงเทพมหานคร : สำนักพิมพ์วรรณิก.กุมภาพันธ์ 2552.

[3] ชาญชัย ศุภอรรถกร. **จัดการฐานข้อมูลด้วย MySQL.** กรุงเทพมหานคร : สำนักพิมพ์ ซิมพลิฟาย. ธันวาคม 2552.

[4] สาธิต ชัยวิวัฒน์ตระกูล, วิทยา ต่อศรีเจริญ. **เก่ง JSP ให้ครบสูตร.** กรุงเทพมหานคร : บริษัท วิตตี้ กรุ๊ป จำกัด. ธันวาคม 2545.

[5] ตวงพร ขอเจริญพร. **เขียนโปรแกรม Java บน Web ด้วย Servlets และ JSP.** กรุงเทพมหานคร : สำนักพิมพ์ เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด

[6] Ian Sommerville. **SOFTWARE ENGINEERING.** United states of America : Pearson Education Inc.,2011

[7] MySQL Developer zone. **"MySQL 5.5 Reference Manual".** [Online]. Available : http://dev.mysql.com/doc/refman/5.5/en/index.html

ภาคผนวก ก

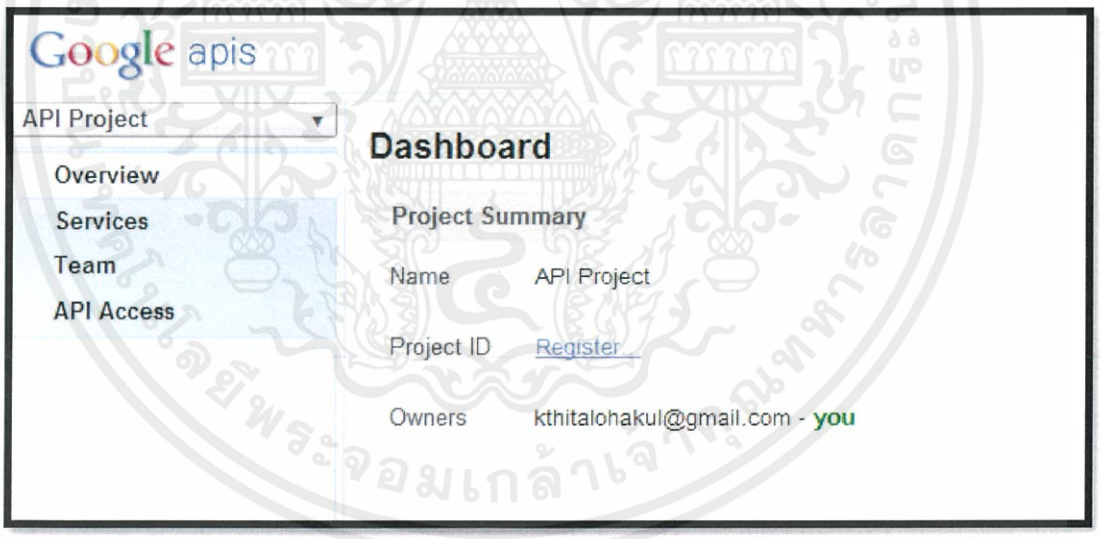
คู่มือการติดตั้ง

1. การติดตั้งบริการ Google Cloud Storage และการใช้งาน

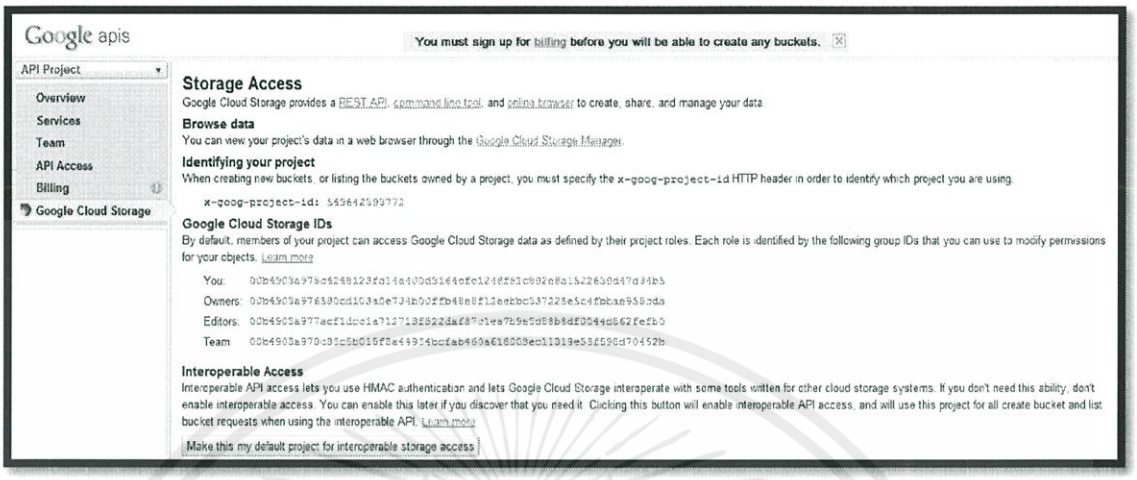
การเริ่มขอเปิดการใช้งานสามารถเบื้องต้นผู้ใช้งานจำเป็นต้องมี

1) Google account ผู้ที่ต้องการขอเปิดการใช้งานจำเป็นต้องมี Google account ก่อน ซึ่งถ้าหากผู้ต้องการใช้งานมีบัญชีอยู่เรียบร้อยแล้วก็ไม่จำเป็นที่จะต้องเปิดบัญชีใหม่

2) Google APIs console project โดยที่ Google APIs console เป็นหน้าเว็บเพจแบบกราฟิก ซึ่งใช้ในการจัดการบริการต่างๆของ Google APIs ที่ได้กล่าวไปแล้ว ภายในจะมีส่วนประกอบของ บริการต่างๆ ให้ผู้ใช้ทำการเลือกใช้ ดังภาพ และเพื่อทำการขอใช้งาน Google cloud storage ให้ทำการเลือกที่หัวข้อ Services และ ภายในจะประกอบไปด้วยหลายบริการให้เลือก On ที่ Status ของบริการ Google cloud storage



รูปที่ ก.1 Google APIs console



รูปที่ ก.2 หน้าจอหลักของ Google Cloud Storage

เมื่อทำการสมัครเรียบร้อย หน้าจอของ Google APIs console จะปรากฏดังภาพ ก.2 ซึ่งปรากฏบริการใหม่ของ Google APIs ที่ทำการเพิ่มเข้าไป

จากเนื้อความที่ได้กล่าวไปก่อนหน้านี้ ว่า Google Cloud Storage สามารถทำการอัพโหลดไฟล์เข้าไปยังคลาวด์ได้นั้น แต่วัตถุประสงค์ของการใช้บริการนี้ เนื่องจากว่า คำสั่งที่ใช้ในการอัพโหลดไฟล์เข้าไปยังดาต้าเบสไม่สามารถเรียกใช้ได้โดยตรง ซึ่งคำสั่งที่ว่านั้น คือ LOAD DATA INFILE ซึ่งไม่ถูกรองรับการให้บริการจาก Google Cloud SQL ทำให้ หากเราต้องการอัพโหลดไฟล์ฐานข้อมูลต่างๆ ที่เราเคยมีอยู่แล้ว ไปยัง Google Cloud SQL Instance ต้องกระทำผ่าน Google Cloud Storage account และใช้ คำสั่ง mysqldump ช่วยในสร้างไฟล์ให้เป็นไฟล์นามสกุล .SQL ก่อน

ใช้คำสั่ง mysqldump เพื่อใช้ในสำรองข้อมูล เป็นหลักซึ่งสามารถใช้ในการสร้างไฟล์ ให้เป็นนามสกุล .SQL ขึ้นมา โดยทำการติดตั้ง MySQL Command Line Client ไว้ที่ตัวเครื่องและทำการเรียกผ่าน Command Prompt ของระบบปฏิบัติการ และทำการพิมพ์คำสั่งดังต่อไปนี้

คำสั่งที่ ก.1 การสำรองข้อมูลจากตาราง

```
mysqldump -uUser -pPasword db_name table_name> newtable.sql
```

โดยไฟล์ใหม่ที่ทำการสำรองข้อมูลนี้ เราสามารถอัพโหลดขึ้นไปที่ Google Cloud Storage ได้ โดยก่อนการอัพโหลดไฟล์นั้น จำเป็นจะต้องทำการดาวน์โหลดโปรแกรม Python , GSUtil tool ซึ่งใช้ในการอัพโหลดข้อมูลเข้าไปยัง Google Cloud Storage โดยในการเริ่มใช้งานนั้น Google Cloud Storage จะมีการยืนยันก่อนการเข้าใช้งานผ่านระบบ ทั้งหมดดำเนินการดังต่อไปนี้

เข้าไปยังหน้าต่าง Command Prompt ของ Window และทำการเข้าไปใน c:\gsutil พิมพ์คำสั่ง python gsutil config ซึ่งหลังจากนั้น gsutil จะทำการแสดง URL ของหน้าเว็บที่ใช้ในการแสดงตัวตน (Authorization page) ซึ่งเพื่อให้เราแสดงตัวเราในการเข้าถึง Google Cloud Storage โดยหน้าต่างจะรอรหัส Authentication code และเมื่อเราทำการเข้าไปที่หน้าดังกล่าว เราจะได้ Authentication code โดยให้นำรหัสดังกล่าวใส่ลงไป หลังจากนั้นจะได้รับการตอบรับของ gsutil prompt โดยคำสั่งจะถามถึง project-id ซึ่งเราสามารถเข้าไปดูได้ใน Google APIs console และเมื่อป้อนรหัสลงไป เราสามารถเริ่มทำการอัพโหลดข้อมูลใน Google Cloud Storage หลังจากนั้นทำการสร้าง bucket ซึ่งเป็นตัวที่เก็บ ไฟล์ฐานข้อมูลต่างๆ หรืออาจมองเป็น Folder ของGoogle Cloud Storage สามารถพิมพ์ได้ด้วยคำสั่ง ก.2

คำสั่งที ก.2 การสร้างโฟลเดอร์ใน Google Cloud Storage

```
python gsutil mb gs://bucketname
```

คำสั่งที่ ก.3 การคัดลอกไฟล์ที่ทำการสำรองข้อมูล

```
Python gsutil cp tablefile.sql gs://bucketname
```

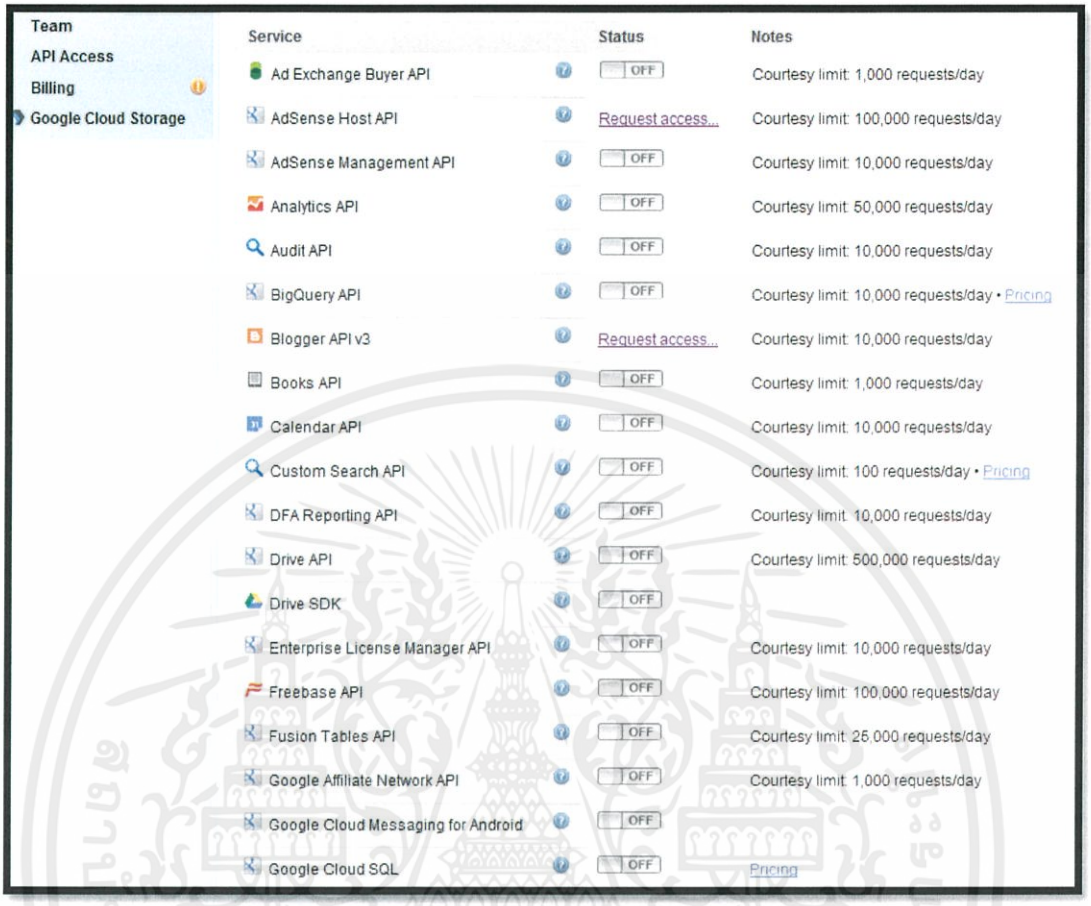
คำสั่งที่ ก.4 การตรวจสอบการอัพโหลดไฟล์

```
Python gsutil ls -l gs://bucketname
```

ซึ่งไฟล์ที่คัดลอกขึ้นไปเรียบร้อยนั้นจะปรากฏว่าอยู่ภายในโฟลเดอร์ gs://bucketname/newtable.sql และระบุว่ามีขนาดเท่าใด แสดงว่าไฟล์ดังกล่าวถูกอัพโหลดเข้าไปที่ Google Cloud Storage เรียบร้อยแล้ว ซึ่งขั้นตอนต่อไปคือการนำไฟล์ข้อมูลดังกล่าว มาใช้ต่อใน Google Cloud SQL ต่อไป

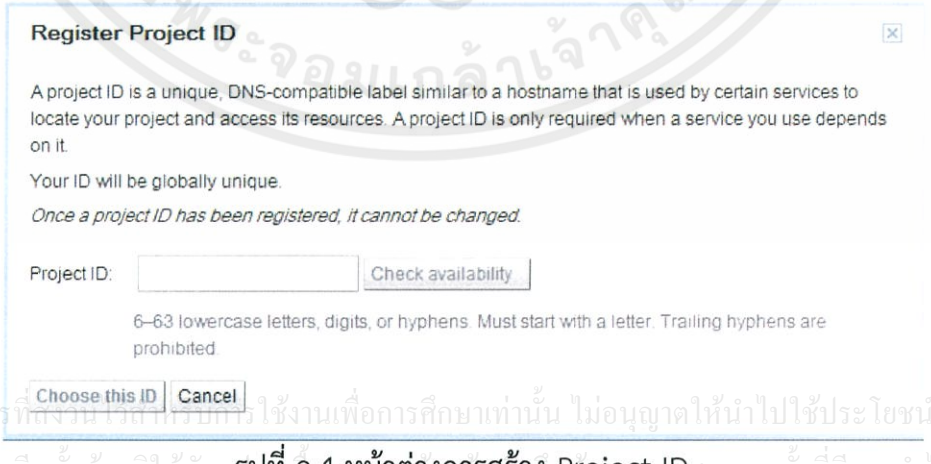
2. การติดตั้งบริการ Google Cloud SQL

เริ่มขอเปิดใช้งาน โดยอาศัย Google account และทำการเข้าไปที่หน้า Google APIs Console เมื่อเข้าไปแล้วให้ดำเนินการคล้ายกับ Google Cloud Storage เลือกที่ Services แล้วทำการ เปลี่ยน Status ของบริการ Google Cloud SQL ให้เป็น On เพื่อขอเปิดใช้งาน



รูปที่ ก.3 หน้าบริการหลักต่าง ๆ ภายในระบบคลาวด์

หลังจากนั้นที่หน้าจอจะปรากฏหน้าต่าง เพื่อให้เราทำการตั้งค่า Project ID ซึ่งใช้ในการระบุโปรเจคที่เราจะดำเนินการใช้งานต่อไป



รูปที่ ก.4 หน้าต่างการสร้าง Project ID

เมื่อทำการตั้งค่า Project ID เรียบร้อยแล้ว จะต้องทำการสร้าง Instance ซึ่งเปรียบเสมือนเป็น ระบบจัดการฐานข้อมูล โดยเมื่อทำการสร้าง Instance จะมีหน้าต่างให้กรอกชื่อ และกำหนดรูปแบบการจ่ายเงิน ซึ่งในปัจจุบัน (หลังวันที่ 12 มิถุนายน พ.ศ.2555) การใช้งานบนคลาวด์เอสคิวแอล จำเป็นต้องมีการจ่ายเงินในระหว่างทดสอบ ซึ่งแบ่งออกเป็น

1) จ่ายแบบ Packages Billing เหมาะกับผู้ใช้ที่ต้องการใช้ขนาดของ RAM หรือพื้นที่ที่เก็บข้อมูลเป็นจำนวนมาก และเน้น การใช้งานตลอดทั้งวัน โดยราคาในปัจจุบันสามารถสรุปได้ดังตาราง ก.1 ดังนี้

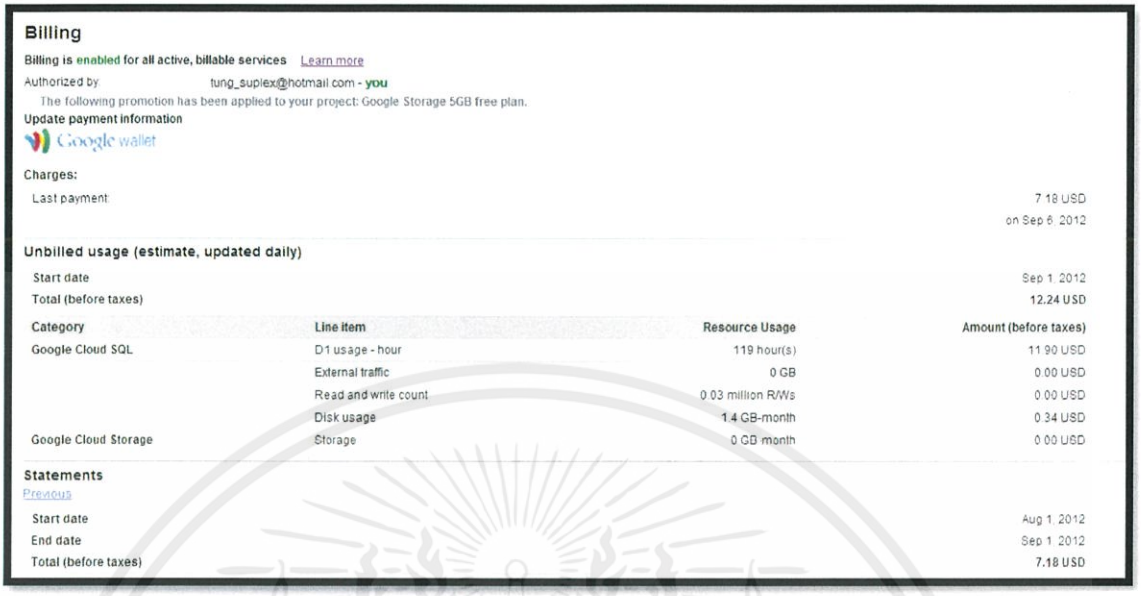
ตารางที่ ก.1 รายละเอียดการคิดค่าบริการแบบ Packing Billing

Tier	RAM	Included Storage	Included I/O per Day	Charge per Day
D1	0.5GB	1GB	850K	$ 1.46
D2	1GB	2GB	1.7M	$ 2.93
D4	2GB	5GB	4 M	$ 5.86
D8	4GB	10GB	8 M	$11.71

2) จ่ายแบบ Per Use Billing เหมาะกับผู้ใช้งานที่เน้นการใช้งานของฮาร์ดแวร์ที่ไม่มาก และมีการทำงานแต่ละวัน มากหรือน้อยไม่สม่ำเสมอ ทำให้โครงงานนี้ เลือกใช้การจ่ายตามเท่าที่ใช้ โดยการค่าใช้จ่ายคิดเป็นจำนวนชั่วโมงที่ใช้กับขนาดของ RAM ที่ต้องการใช้งาน โดยในโครงงาน ใช้พื้นที่ใช้งานเป็น D1 Database Instance (0.5GB RAM) ซึ่งคิดเป็นการใช้งานชั่วโมงละ 0.10 $ โดยตารางในปัจจุบันสามารถสรุปได้ดังตาราง ก.2 ดังนี้

ตารางที่ ก.2 รายละเอียดการคิดค่าบริการแบบ Per Use Billing

Resource	Charge
D1 Database Instance (0.5GB RAM)	$ 0.10 per hour
D2 Database Instance (1.0GB RAM)	$ 0.19 per hour
D4 Database Instance (2.0GB RAM)	$ 0.38 per hour
D8 Database Instance (4.0GB RAM)	$ 0.77 per hour
1GB Storage	$ 0.24 per month
I/O	$ 0.10 per Million



Billing

Billing is enabled for all active, billable services Learn more

Authorized by: tung_suplex@hotmail.com - **you**

The following promotion has been applied to your project: Google Storage 5GB free plan.

Update payment information

Google wallet

Charges:

Last payment:	7.18 USD on Sep 6, 2012

Unbilled usage (estimate, updated daily)

Start date			Sep 1, 2012
Total (before taxes)			12.24 USD
Category	**Line item**	**Resource Usage**	**Amount (before taxes)**
Google Cloud SQL	D1 usage - hour	119 hour(s)	11.90 USD
	External traffic	0 GB	0.00 USD
	Read and write count	0.03 million R/Ws	0.00 USD
	Disk usage	1.4 GB-month	0.34 USD
Google Cloud Storage	Storage	0 GB month	0.00 USD

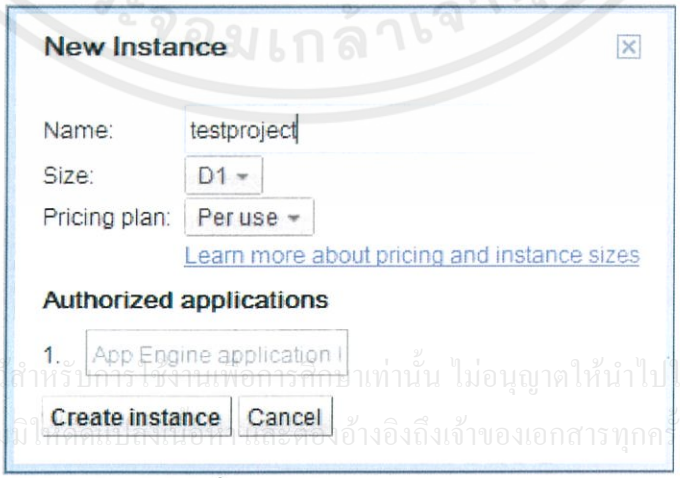
Statements

Previous

Start date	Aug 1, 2012
End date	Sep 1, 2012
Total (before taxes)	7.18 USD

รูปที่ ก.5 หน้าต่าง Billing

หลังจากทำการตัดสินใจที่จะเลือกการใช้จ่ายในรูปแบบ D1 ของ Per Use Billing แล้วให้ทำการสร้าง Instance และทำการตั้งค่าต่างๆ โดยที่ในช่อง Authorized applications คือช่องที่ใช้ในการระบุว่า เราต้องการให้ระบบจัดการฐานข้อมูล หรือ Instance ตัวนี้ ถูกใช้งานเข้ากับ Web Application ตัวไหนบ้าง ซึ่งจะอธิบายต่อไปเมื่อถึงบริการ Google App Engine ต่อไป โดยค่าต่างๆ เมื่อทำการตั้งค่าแล้วหน้าจอหลักของ Google APIs Console จะมีบริการเพิ่มขึ้นอีก 1 รายการ ซึ่งทำให้ในการใช้งานจนถึงปัจจุบันนี้ ที่หน้าหลักจะแสดง ค่าของ Project ID และ แสดงอีเมลของผู้เป็นเจ้าของโปรเจค และเมื่อทำกดเลือกไปที่ Google Cloud SQL จะพบว่า Instance: testproject กำลังอยู่ในสถานะใช้งานอยู่



New Instance

Name: testproject

Size: D1

Pricing plan: Per use

Learn more about pricing and instance sizes

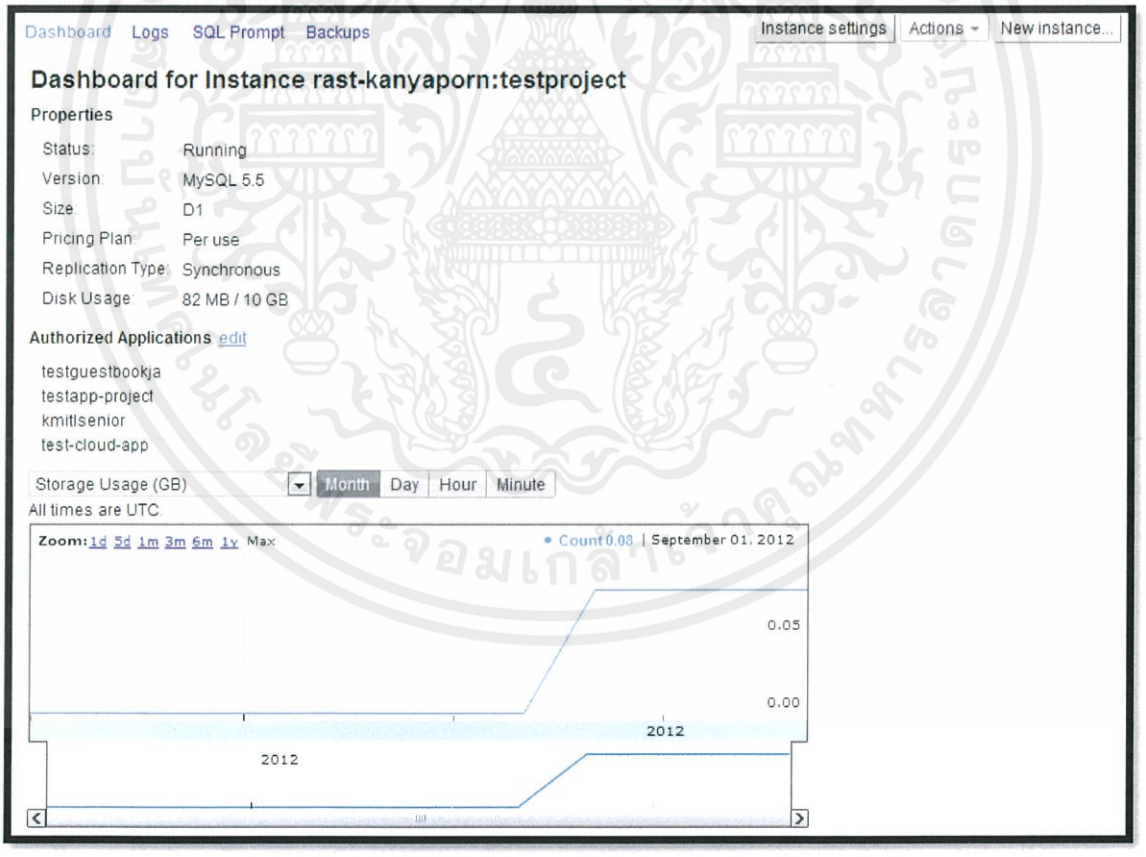
Authorized applications

1. App Engine application I

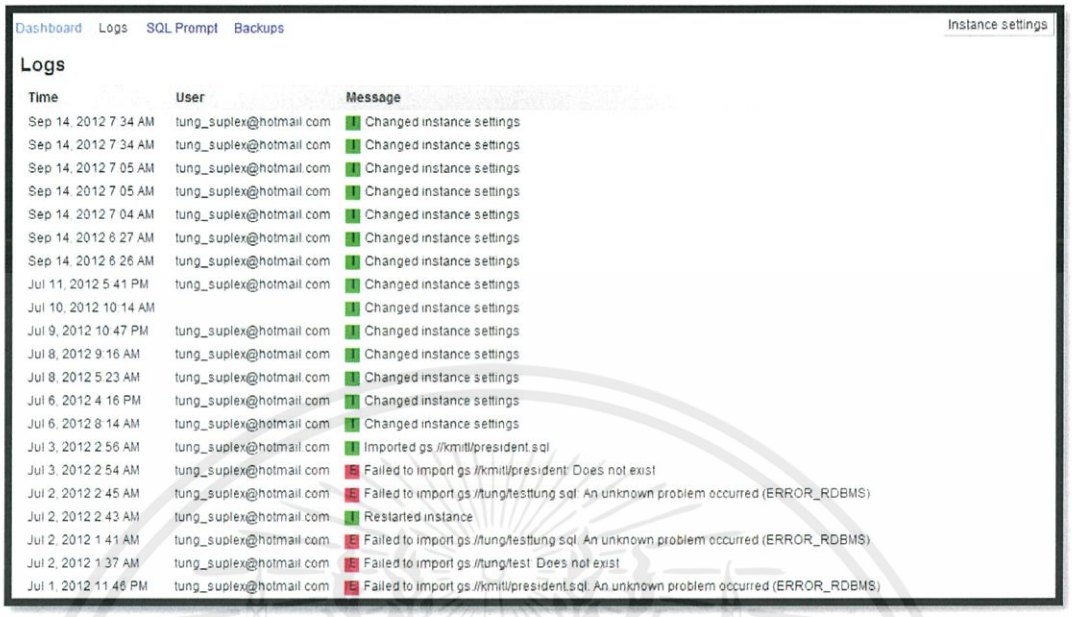
Create instance | Cancel

รูปที่ ก.6 หน้าต่าง Instance

เมื่อทำการสร้าง Instance เสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปคือการเข้าไปทดลองใช้งานในระบบคลาวด์ โดยศึกษาถึงคุณสมบัติเบื้องต้นบนเว็บที่สามารถใช้งานได้ พบว่าบนหน้าหลักของเว็บจะประกอบไปด้วย Dashboard ซึ่งเป็นเหมือนกระดานที่แสดงคุณสมบัติและรายละเอียดของ Instance นั้นๆ นอกจากนี้จะแสดง Logs ที่เกิดจากการดำเนินการเปลี่ยนแปลงค่าของ Instance ได้แก่ การดึงค่าข้อมูลเข้า การนำค่าข้อมูลออกจากคลาวด์เอสคิวแอล หรือการรีเซตค่า Instance ใหม่เป็นต้น นอกจากนี้กูเกิลได้ทำการสร้างคุณสมบัติหนึ่งเพิ่มเติมขึ้นมา ชื่อว่า SQL Prompt เพื่อให้ผู้ใช้งานในระบบคลาวด์เกิดความง่ายต่อการใช้งาน ซึ่งสามารถทำงานโต้ตอบกับผู้ใช้งานบนระบบฐานข้อมูลได้ผ่านเว็บเซอร์วิส และสามารถเชื่อมต่อการใช้งานกับฐานข้อมูลได้โดยตรง และใช้คำสั่งเอสคิวแอลในการดำเนินการกับแต่ละฐานข้อมูลหรือแต่ละตาราง ซึ่งคำสั่งเอสคิวแอลบนคลาวด์ มีโครงสร้างของภาษาเช่นเดียวกัน MySQL ทั่วไป



รูปที่ ก.7 หน้าต่าง Dashboard



รูปที่ ก.8 แสดงหน้าต่าง Logs

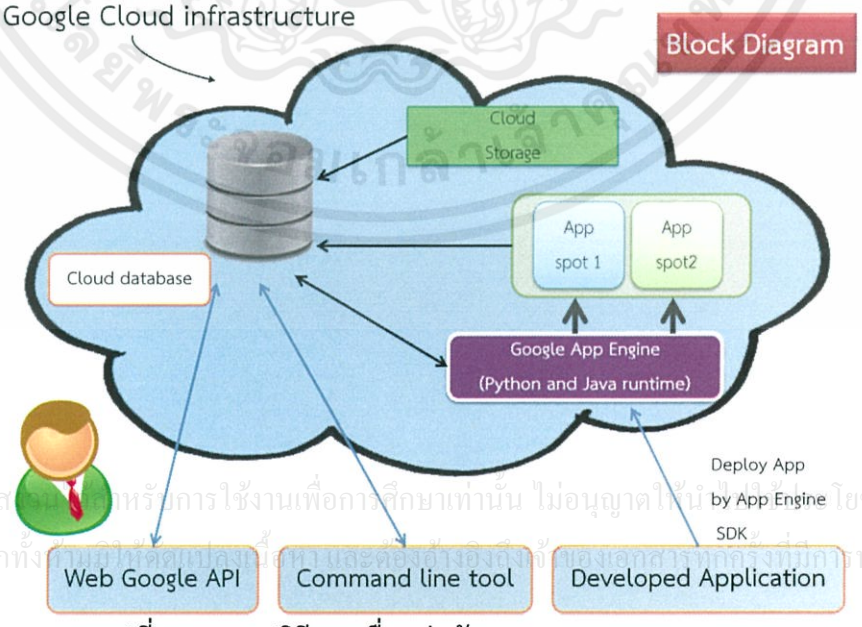
รูปที่ ก.9 หน้าต่าง SQL Prompt

รูปที่ ก.10 หน้าต่าง Backups

เครื่องมืออีกชนิดหนึ่ง ที่สามารถใช้ในการเข้าถึงฐานข้อมูลบนคลาวด์ได้โดยไม่จำเป็นต้องทำการเข้าผ่านเว็บ โดยเครื่องมือชนิดนี้เรียกว่า Command line tool ซึ่งเป็นเครื่องมือที่ถูกพัฒนามาจากทีมพัฒนาของกูเกิลโดยลักษณะของเครื่องมือ ทำให้การใช้งานบนคลาวด์ง่ายมากยิ่งขึ้น ลักษณะคล้ายกับ Command Prompt ของ Windows โดยในการเชื่อมต่อ สามารถพิมพ์โดยใช้คำสั่งในการเข้าระบบ

รูปที่ ก.11 แสดงการใช้งาน Command line tool

กล่าวโดยสรุป ผู้ใช้งานสามารถติดต่อกับ Google Cloud SQL ด้วย Google service ได้หลายวิธีด้วยกันเช่นผ่าน Web google APIs (https://code.google.com/apis/console/) และสามารถเข้าผ่านตัว Command line tool ซึ่งอธิบายไปในข้างต้นได้ โดยทำการระบุ Instance ที่ต้องการเข้าใช้งาน และ อีกทางหนึ่ง คือการติดต่อผ่านเว็บแอพพลิเคชั่นได้ โดยทำการ Deploy เว็บแอพพลิเคชั่น ที่เชื่อมต่อกับ ฐานข้อมูลบนคลาวด์



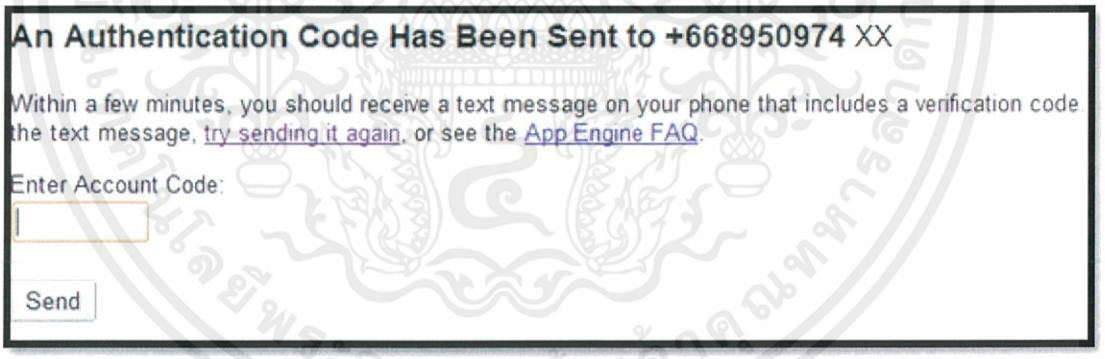
รูปที่ ก.12 สรุปวิธีการเชื่อมต่อกับ Google cloud SQL

3. การติดตั้งบริการ Google App Engine

การเริ่มใช้งานสามารถทำได้โดยการ เข้าไปยังหน้าหลักของ Google App Engine

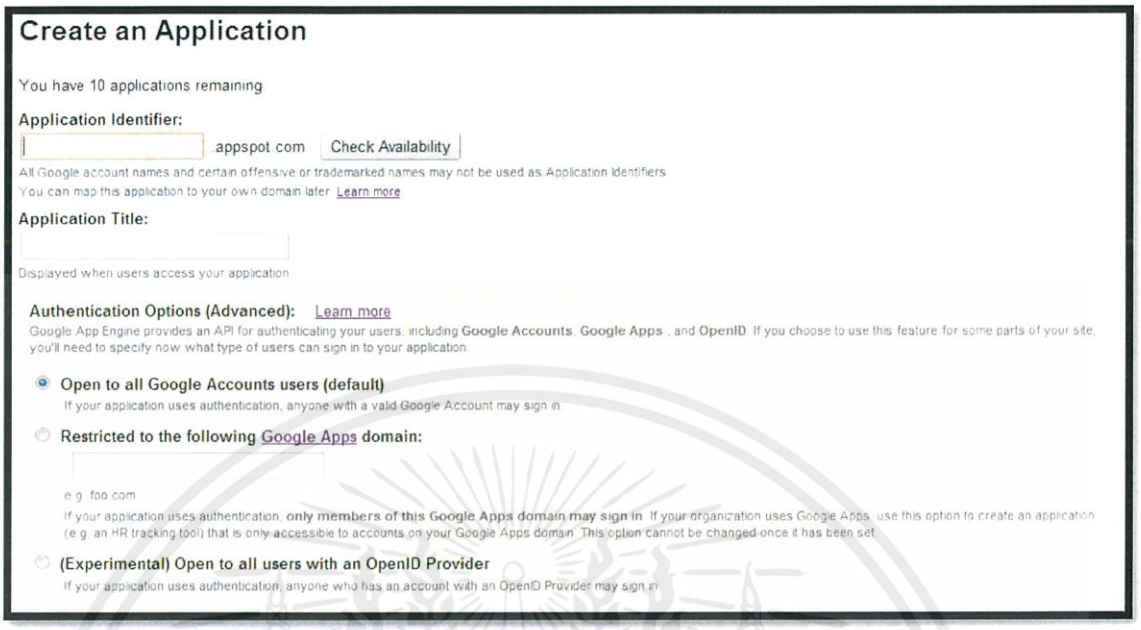
รูปที่ ก.13 แสดงหน้าต่างแรกเพื่อสมัครเข้าใช้งาน

โดยเมื่อทำการเข้าไปสร้าง Application เว็บไซต์จะกำหนดให้เรา ป้อนเบอร์โทรศัพท์มือถือที่เราใช้ เพื่อทำการรับ SMS เป็นรหัสยืนยัน (Verification code) ในการขอเข้าใช้งาน โดยในระยะแรกยังไม่มีการคิดค่าบริการ



รูปที่ ก.14 การยืนยันตัวผู้ใช้ก่อนเริ่มใช้งาน

หลังจากนั้นหน้าต่างจะให้ ผู้ใช้ทำการสร้างชื่อแอพพลิเคชั่น และกำหนดความเข้มงวดการเข้าถึงเบื้องต้น



รูปที่ ก.15 การสร้างแอพพลิเคชั่นบน Google App engine

รูปที่ ก.16 หน้าจอหลักของ Google App Engine

ภาคผนวก ข.

รายละเอียดการพัฒนา Web Application

1. เครื่องมือที่ใช้ในการพัฒนา Web Application

หลังจากที่ได้ติดตั้ง Google App Engine แล้ว จะสามารถพัฒนา Web Application ได้โดยใช้ภาษา Java หรือ Python โดยในการพัฒนาให้สามารถติดต่อกับระบบฐานข้อมูลบนคลาวด์นั้น ผู้ใช้จำเป็นต้องสร้าง Google Cloud Instance ตามที่ได้กล่าวในบทภาคผนวก ก. และสร้าง Application ของ Google App Engine ที่จะติดต่อกับระบบฐานข้อมูล โดยในการพัฒนาจะมีหัวข้อดังนี้

1) ติดตั้ง Jaca SDK (Java Software Development Kit) โดยที่ SDK นั้นรวม software สำหรับ Web server ที่ผู้ใช้สามารถใช้เพื่อทดสอบ Web Application ของตนเอง

2) ติดตั้ง Java เนื่องจาก Google App Engine จะสนับสนุน Java 5 และ Java 6 โดยปกติแล้วเมื่อ Application ถูกปฏิบัติบน Google App Engine จะใช้ Java 6 Virtual Machine และ libraly พื้นฐาน ทั้งนี้การพัฒนาจึงแนะนำว่าควรใช้ Java 6 ถึงจะดีที่สุด

3) ติดตั้งและใช้งาน Eclipse และ Google Plugin for Eclipse เนื่องจาก Google App Engine จะสนับสนุนการใช้งาน Eclipse คู่กับ Google Plugin for Eclipse ที่จะช่วยในการ Deploy Web Application ขึ้นไปสู่ Google App Engine ได้โดยง่าย ทั้งนี้ Plugin ดังกล่าวสามารถใช้งานได้กับ Eclipse Version 3.3, 3.4, 3.5, 3.6, 3.7 และ 4.2 โดยในการติดตั้งสามารถทำได้ดังต่อไปนี้

3.1) เลือกเมนู Help > Install New Software

3.2) ระบุ directoey เป็น https://dl.google.com/eclipse/plugin/เวอร์ชั่นของ Eclipse

3.3) ติดตั้ง Plugin

หลังจากที่ติดตั้งโปรแกรมต่างๆเรียบร้อยแล้วผู้ใช้สามารถพัฒนา Web Application และ Deploy ขึ้นไปสู่ Google App Engine โดยใช้ Google Plugin for Eclipse ได้

2. การพัฒนา Web Application

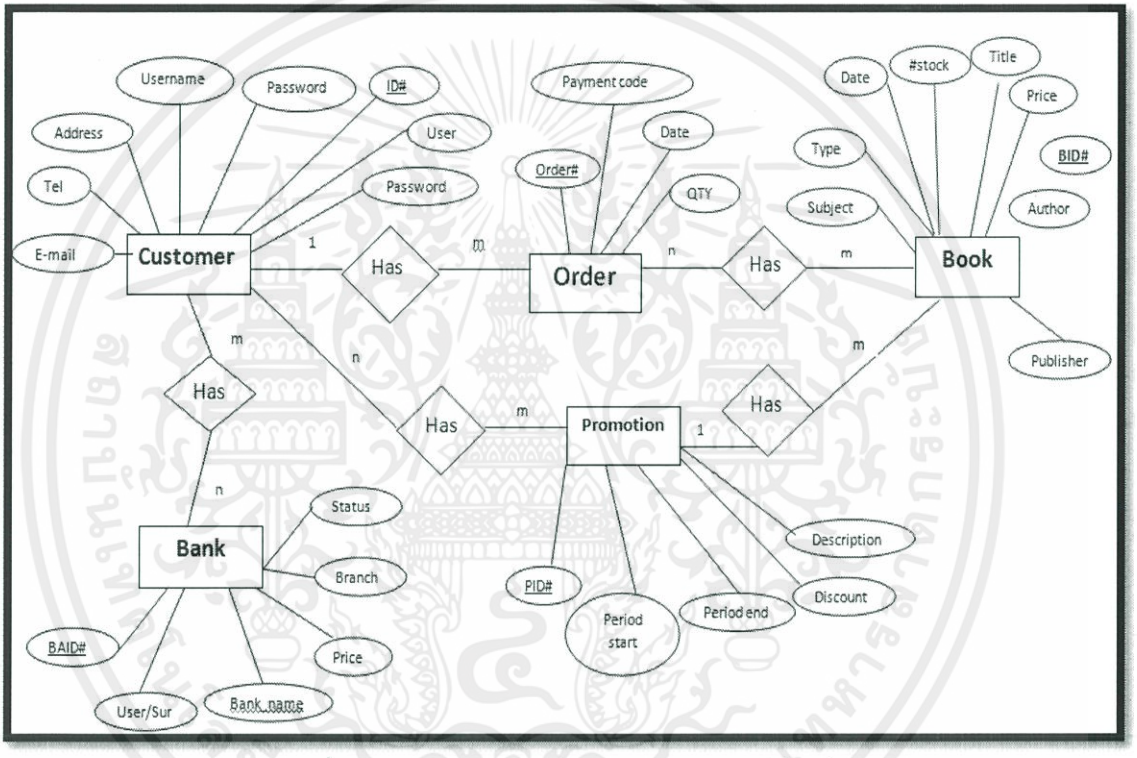
ในการพัฒนา Web Application จะแบ่งออกเป็น 2 Application คือ ส่วนของร้านขายหนังสือออนไลน์ และ เว็บที่ใช้ในการจองตั๋วเครื่องบินออนไลน์ โดยที่ทั้งสอง Application เป็นเพียงตัวอย่างของ Web Application ที่มุ่งเน้นที่ส่วนของการทำทรานแซคชั่นในการซื้อขาย การแก้ไข

ข้อมูล การเพิ่มข้อมูล และ การจองที่นั่ง ของผู้ใช้งานมากกว่าหนึ่งคนในช่วงเวลาเดียวกัน ทั้งนี้ทั้งสอง Application จะใช้ Model MVC ในการพัฒนาขึ้นมา และมีการออกแบบ ER Diagram ที่แตกต่างกันตามความต้องการของผู้ใช้งาน

1) Web Application ร้านขายหนังสือออนไลน์

(เว็บไซต์ : http://testguestbookja.appspot.com/)

ร้านขายหนังสือออนไลน์จะมีการออกแบบ ER Diagram ดังนี้



รูปที่ ข.1 ER Diagram ร้านขายหนังสือออนไลน์

ซึ่งการพัฒนาตาม MVC Model จะแบ่งส่วนออกเป็น 3 ส่วนหลักๆคือ ส่วนของคลาสในการติดต่อกับระบบฐานข้อมูลบนคลาวด์,ส่วนของคลาสที่ควมคุมการทำงานต่างๆของ Web Application และส่วนของการแสดงผลซึ่งจะแสดงเป็นหน้าเพจ .jsp ซึ่งในส่วนของการคลาสที่ติดต่อกับฐานข้อมูลจะมี method หลักๆดังต่อไปนี้

คำสั่งที่ ข.1 method ติดต่อระบบฐานข้อมูล


```
public boolean connecToDatabase(String user, String pass)
throws Exception {
      try {
      Class.forName("com.mysql.jdbc.Driver");
```

```
      Connection c  =
DriverManager.getConnection("jdbc:mysql://localhost/president"
,user,pass);
      this.connection = c;
      return true;
      }
      catch (SQLException ex){
      return false;
      }
}
```

เป็น method ที่ทำหน้าที่ในการติดต่อกับระบบฐานข้อมูลโดยใช้ Username และ Password ในการ Login เข้าสู่ระบบฐานข้อมูล ทั้งนี้ส่วนที่ควบคุมการ Login นั้นจะเป็น DBMS ส่งผลให้ผู้ใช้ไม่ต้องฝัง Username และ Password ลงในโค้ดของโปรแกรม

คำสั่งที่ ข.2 method แสดงหนังสือทั้งหมดในฐานข้อมูล

```
public List<book> showallBook(){
        String SQL = "SELECT * FROM books";
        List<book> list = new ArrayList<book>();
        try {
            Statement state = connection.createStatement();
            ResultSet rs = state.executeQuery(SQL);
            while(rs.next()) {
                book book = new book(rs.getInt(1),
                                     rs.getString(2),
                                     rs.getString(3),
                                     rs.getDate(4),
                                     rs.getInt(5),
                                     rs.getString(6),
                                     rs.getString(7),
                                     rs.getString(8),
                                     rs.getString(9));
                list.add(book);
            }
            return list;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return null;
        }

    }
```


ใช้ในการแสดงรายชื่อรวมทั้งรายละเอียดทั้งหมดของหนังสือแต่ละเล่มที่อยู่ในฐานข้อมูล

คำสั่งที่ ข.3 method ในการหาหนังสือจาก keyword

```
public List<book> findByKeyword(String keyword, String Type){
        String sql = "SELECT * FROM books WHERE " + Type + "
LIKE " + "'%" + keyword + "%'";
        List<book> list = new ArrayList<book>();
        try {
            Statement state = connection.createStatement();
            ResultSet rs = state.executeQuery(sql);
            while(rs.next()) {
                book book = new book(rs.getInt(1),
                                     rs.getString(2),
                                     rs.getString(3),
                                     rs.getDate(4),
                                     rs.getInt(5),
                                     rs.getString(6),
                                     rs.getString(7),
                                     rs.getString(8),
                                     rs.getString(9));
                list.add(book);
            }
            return list;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return null;
        }


    }
```

ใช้ในการค้นหาหนังสือที่อยู่ในระบบฐานข้อมูลโดยผู้ใช้จะกำหนด keyword ที่ใช้ในการช่วยค้นหา และผลลัพธ์ที่ออกมาจะแสดงรายละเอียดของหนังสือแต่ละเล่มที่มีส่วนที่ตรงกับ keyword

คำสั่งที่ ข.4 method ในการแก้ไขข้อมูลหนังสือ

```
public boolean updateBook(String amount, String isbn){
     String sql = "UPDATE books SET StockNum = StockNum - " +
amount + " WHERE ISBN = " + isbn;
try {
     Statement state = connection.createStatement();
     state.executeUpdate(sql);
     return true;

     } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
     return false;
     }
}
```

ใช้สำหรับพนักงานที่ต้องการแก้ไขข้อมูลของหนังสือที่อยู่ในระบบฐานข้อมูล

คำสั่งที่ ข.5 method ในการค้นหา ID ของลูกค้าที่อยู่ในฐานข้อมูล

```
public int findUser(String user, String password){
         String sql = "SELECT UID FROM customers WHERE
username = '" + user + "' AND password = '" + password + "'";
        try {
            Statement state = connection.createStatement();
            ResultSet rs = state.executeQuery(sql);
            rs.next();
            int uid = rs.getInt(1);
            return uid;



        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return 0;
        }
    }
```

ใช้หา ID ของลูกค้าที่อยู่ในระบบฐานข้อมูลเพื่อนำ ID ดังกล่าวไปใช้งานต่อไปในส่วนของทรานแซคชั่นการซื้อขายหนังสือ เพื่อเก็บข้อมูลของลูกค้าที่ซื้อหนังสือแต่ละเล่ม

คำสั่งที่ ข.6 method ในการเก็บข้อมูลการสั่งซื้อ

```
public void InsertOrderTable(int uid){
        java.sql.Date sqlDate = new java.sql.Date(new
java.util.Date().getTime());
        String sql="INSERT INTO OrderTable (Dates,UID)
values(?,?)";
        try {
            PreparedStatement pstmt =
connection.prepareStatement(sql);
            pstmt.setDate(1, sqlDate);
            pstmt.setInt(2, uid);
            pstmt.executeUpdate();
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }
```

ใช้ในการเก็บข้อมูลการสั่งซื้อหนังสือของลูกค้าแต่ละท่านลงในระบบฐานข้อมูล

คำสั่งที่ ข.7 method ในการค้าหา ID ของการสั่งซื้อ

```
public int getOrderNum(){
         String sql = "SELECT MAX(orderNum) FROM OrderTable";
        try {
            Statement state = connection.createStatement();
            ResultSet rs = state.executeQuery(sql);
            rs.next();
            int ordernum = rs.getInt(1);
            return ordernum;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return 0;
        }

    }
```

ค้นหาหมายเลขการสั่งซื้อหนังสือในครั้งล่าสุดเพื่อนำมาใช้ในการทำทรานแซคชั่นในส่วนของการซื้อหนังสือ

คำสั่งที่ ข.8 method เก็บข้อมูลความสัมพันธ์ของหนังสือและรายการสั่งซื้อ

```
public void InsertbookOrderShip(int ordernum, int ISBN, int
amount){
          String sql="INSERT INTO bookOrderShip
(orderNum,ISBN,amount) values(?,?,?)";
        try {
            PreparedStatement pstmt =
connection.prepareStatement(sql);
            pstmt.setInt(1, ordernum);
            pstmt.setInt(2, ISBN);
            pstmt.setInt(3, amount);
            pstmt.executeUpdate();
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

เก็บข้อมูลของการความสัมพันธ์ระหว่างรายการการสั่งซื้อกับหนังสือและจำนวนที่สั่งซื้อลงในระบบฐานข้อมูล

คำสั่งที่ ข.9 method แก้ไขข้อมูลของหนังสือ

```
public boolean editBook(String Subject, String Type, int
StockNum, String Title, String Prize, String Author, String
Publisher, int ISBN){
         String sql = "UPDATE books SET Subject=?, Type=?,
StockNum=?, Title=?, Price=?, Author=?, Publisher=? WHERE
ISBN=?";
        try {
            PreparedStatement pstmt =
connection.prepareStatement(sql);
            pstmt.setString(1, Subject);
            pstmt.setString(2, Type);
            pstmt.setInt(3, StockNum);
            pstmt.setString(4, Title);
            pstmt.setString(5, Prize);
            pstmt.setString(6, Author);
            pstmt.setString(7, Publisher);
            pstmt.setInt(8, ISBN);
            pstmt.executeUpdate();
            return true;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return false;
        }
    }
```

ใช้ในการที่พนักงานของร้านต้องการที่จะแก้ไขข้อมูลต่างๆของหนังสือ เช่น ชื่อหนังสือ ราคา รวมไปถึงจำนวนหนังสือที่ร้านมีอยู่ ลงในระบบฐานข้อมูล

คำสั่งที่ ข.10 method ลบหนังสือ

```
public boolean deleteBook(int ISSBN){
         String sql = "DELETE FROM books WHERE ISBN=?";
        try {
            PreparedStatement pstmt =
connection.prepareStatement(sql);
            pstmt.setInt(1, ISSBN);
            pstmt.executeUpdate();
            return true;
        } catch (SQLException ex) {
```

```
Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return false;
        }

    }
```

ใช้ในการลบหนังสือออกจากระบบฐานข้อมูล

คำสั่งที่ ข.11 method แสดงรายการสั่งซื้อ

```
public List<Order> listOrder() {
        String sql = "SELECT * FROM OrderTable";
        List<Order> list = new ArrayList<Order>();
        try {
            Statement state = connection.createStatement();
            ResultSet rs = state.executeQuery(sql);
            while(rs.next()) {
                Order order = new Order(rs.getInt(1),
                                        rs.getDate(2).toString(),
                                        rs.getInt(3));
                list.add(order);
            }
            return list;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return null;
        }
    }
```

ใช้แสดงรายการการสั่งซื้อทั้งหมดของลูกค้าทุกคนที่สั่งซื้อหนังสือโดยมีรายละเอียดของหมายเลขของรายการและลูกค้าที่ถือครองรายการนั้นๆ

คำสั่งที่ ข.12 แสดงรายละเอียดของหนังสือที่ถูกซื้อโดยลูกค้า

```
public List<OrderDetail> showOrderbyID(int OrderID){
         String sql = "select books.ISBN, Title, Price,
amount"+
                      " from books inner join bookOrderShip"
+
                      " on books.ISBN = bookOrderShip.ISBN" +
                      " inner join OrderTable" +
                      " on bookOrderShip.orderNum =
OrderTable.orderNum" +
                      " where OrderTable.orderNum = ?";
```

```
         List<OrderDetail> list = new
ArrayList<OrderDetail>();
        try {
            PreparedStatement pstmt =
connection.prepareStatement(sql);
            pstmt.setInt(1, OrderID);
            ResultSet rs = pstmt.executeQuery();

            while (rs.next()){
                OrderDetail orderdetail = new
OrderDetail(rs.getInt(1),

rs.getString(2),

rs.getInt(3),

rs.getInt(4));
                    list.add(orderdetail);
            }
            return list;
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
            return null;
        }
    }
```

แสดงรายละเอียดของหนังสือแต่ละเล่มที่ถูกซื้อโดยลูกค้าแต่ละคนที่มีหมายเลขรายการการสั่งซื้อหมายเลขใดหมายเลขหนึ่งเพื่อสรุปรายการการสั่งซื้อนั้นๆ

คำสั่งที่ ข.13 method เปิดทรานแซคชั่น

```
public void beginTx() {
        try {
            connection.setAutoCommit(false);
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }
        }
```

เป็น method ที่ใช้ในการเริ่มต้นทรานแซคชั่นมีค่าเทียบเท่าใช้งานคำสั่ง begin บน Interactive SQL

คำสั่งที่ ข.14 method ในการ commit

```
public void commitResult() {
        try {
            connection.commit();
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }
        }
```

ใช้เพื่อยืนยันการเปลี่ยนแปลงที่เกิดจากทรานแซคชั่น

คำสั่งที่ ข.15 method rollback ทรานแซคชั่น

```
public void rollback(){
        try {
            connection.rollback();
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

ใช้ในการยกเลิกการเปลี่ยนแปลงที่เกิดจากทรานแซคชั่นในกรณีที่ตรวจสอบพบความผิดปกติในส่วนของข้อมูลที่อยู่ในระบบฐานข้อมูลกับส่วนบังคับใช้ความถูกต้องของระบบฐานข้อมูล

คำสั่งที่ ข.16 method ล็อคข้อมูล

```
public void findIDLock(String id){
        String sql = "SELECT * FROM books WHERE ISBN = " + id
+ " FOR UPDATE";
        try {
            Statement state = connection.createStatement();
            state.executeQuery(sql);
        } catch (SQLException ex) {
  Logger.getLogger(Database.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
```

เป็น method ที่ใช้ในการค้นหาหนังสือในระบบฐานข้อมูลที่มีหมายเลขของหนังสือตรงกับที่ต้องการและทำการล็อคข้อมูลของหนังสือหมายเลขนั้นๆเพื่อป้องกันการเปลี่ยนแปลงที่อาจเกิดขึ้นโดยทรานแซคชั่นอื่นที่อยู่ในระบบ

ในส่วนต่อไปเป็นส่วนของคลาสที่ใช้ควบคุมการทำงานของ Web Application ในส่วนของการเลือกติดต่อกับระบบฐานข้อมูลและแสดงผลออกมาให้ผู้ใช้งานเห็น ซึ่งคลาสเหล่านี้จะเป็น Servlet Class ที่ใช้จัดการกับ Request ต่างๆที่เข้ามาใน Server โดยจะยกตัวอย่าง Class ที่สำคัญๆ บางส่วนดังต่อไปนี้

คำสั่งที่ ข.17 คลาสที่ดูแลการเชื่อมต่อฐานข้อมูล

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, SQLException
{
        Database db = new Database();
        if(db.connectionToDatabaseGuest()==true) {
            List<book> result = db.showallBook();
            List<book> result2 = db.showallBookHot();
            List<book> result3 = db.showallBookNew();
            int num = result.size();
            int num2 = result2.size();
            int num3 = result3.size();
            int numm;
            if(num%4==0){
                numm = num/4;
            } else {
                numm = (num/4) + 1;
            }
            db.closeDB();
            request.setAttribute("result", result);
            request.setAttribute("result2", result2);
            request.setAttribute("result3", result3);
            request.setAttribute("num", num);
            request.setAttribute("numm", numm);
            request.setAttribute("num2", num2);
            request.setAttribute("num3", num3);
            RequestDispatcher rd =
request.getRequestDispatcher("SlideBook.jsp");
            rd.forward(request, response);
        }
        else {
                    response.sendRedirect("failed.jsp");
                }
    }
```

เป็นคลาสเริ่มต้นของผู้ใช้งานในการเข้ามาเชื่อมต่อกับระบบฐานข้อมูลโดยถ้าหากเชื่อมต่อสำเร็จจะทำการค้นหาหนังสือทั้งหมดที่อยู่ในระบบฐานข้อมูลแล้วส่งผลลัพธ์ที่ได้ให้หน้า JSP ในการแสดงผลให้ผู้ใช้งานเห็นต่อไป ส่วนในกรณีที่ไม่สามารถเชื่อมต่อกับระบบฐานข้อมูลได้จะแสดงหน้า

เพจ JSP ที่บ่งบอกว่าไม่สามารถเชื่อมต่อกับระบบฐานข้อมูลได้ออกมา ทั้งนี้การเชื่อมต่อในคลาสนี้จะเป็นการเชื่อมต่อในฐานะของ Guest

คำสั่งที่ ข.18 คลาสการลบหนังสือจากฐานข้อมูล

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, Exception {
        PrintWriter out = response.getWriter();

       int id = Integer.parseInt(request.getParameter("id"));
       String user = (String)
request.getSession().getAttribute("user");
       String password = (String)
request.getSession().getAttribute("password");
       Database db = new Database();
       db.connecToDatabase(user, password);
       db.beginTx();
       boolean check = db.deleteBook(id);
       if(check) {
           out.println("success");
           db.commitResult();
           db.closeDB();
       } else {
           out.println("failed");
           db.rollback();
           db.closeDB();
       }
    }
```

เป็นคลาสในการลบหนังสือออกจากฐานข้อมูลโดยในการลบข้อมูลนั้นจะปฏิบัติอยู่ในทรานแซคชั่น

คำสั่งที่ ข.19 คลาสการเพิ่มหนังสือลงยังฐานข้อมูล

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, Exception {
        PrintWriter out = response.getWriter();

        String ISBN = (String) request.getParameter("ISBN");
        int issbn = Integer.parseInt(ISBN);
        String Subject = (String)
request.getParameter("Subject");
        String Type = (String) request.getParameter("Type");
        String Stock = (String)
request.getParameter("StockNum");
        int StockNum = Integer.parseInt(Stock);
```

```
        String Title = (String) request.getParameter("Title");
        String Prize = (String) request.getParameter("Prize");
        String Author = (String)
request.getParameter("Author");
        String Publisher = (String)
request.getParameter("Publisher");
        String user = (String)
request.getSession().getAttribute("user");
        String password = (String)
request.getSession().getAttribute("password");
        Database db = new Database();
        db.connecToDatabase(user, password);
        db.beginTx();

        boolean check = db.addBook(issbn, Subject, Type,
StockNum, Title, Prize, Author, Publisher);
        if (check == true){
            out.println("success");
            db.commitResult();
            db.closeDB();
        } else{
            out.println("failed");
            db.rollback();
            db.closeDB();
        }

    }
```

เป็นคลาสที่ใช้ในการเพิ่มหนังสือลงในระบบฐานข้อมูลโดยที่จะถูกบังคับความถูกต้องของฐานข้อมูลด้วย Primary key Constraint และทำทรานแซคชั่นในกรณีที่ปฏิบัติคำสั่งแบบขนาน

คำสั่งที่ ข.20 คลาสที่ดูแลการทำทรานแซคชั่นการซื้อหนังสือ

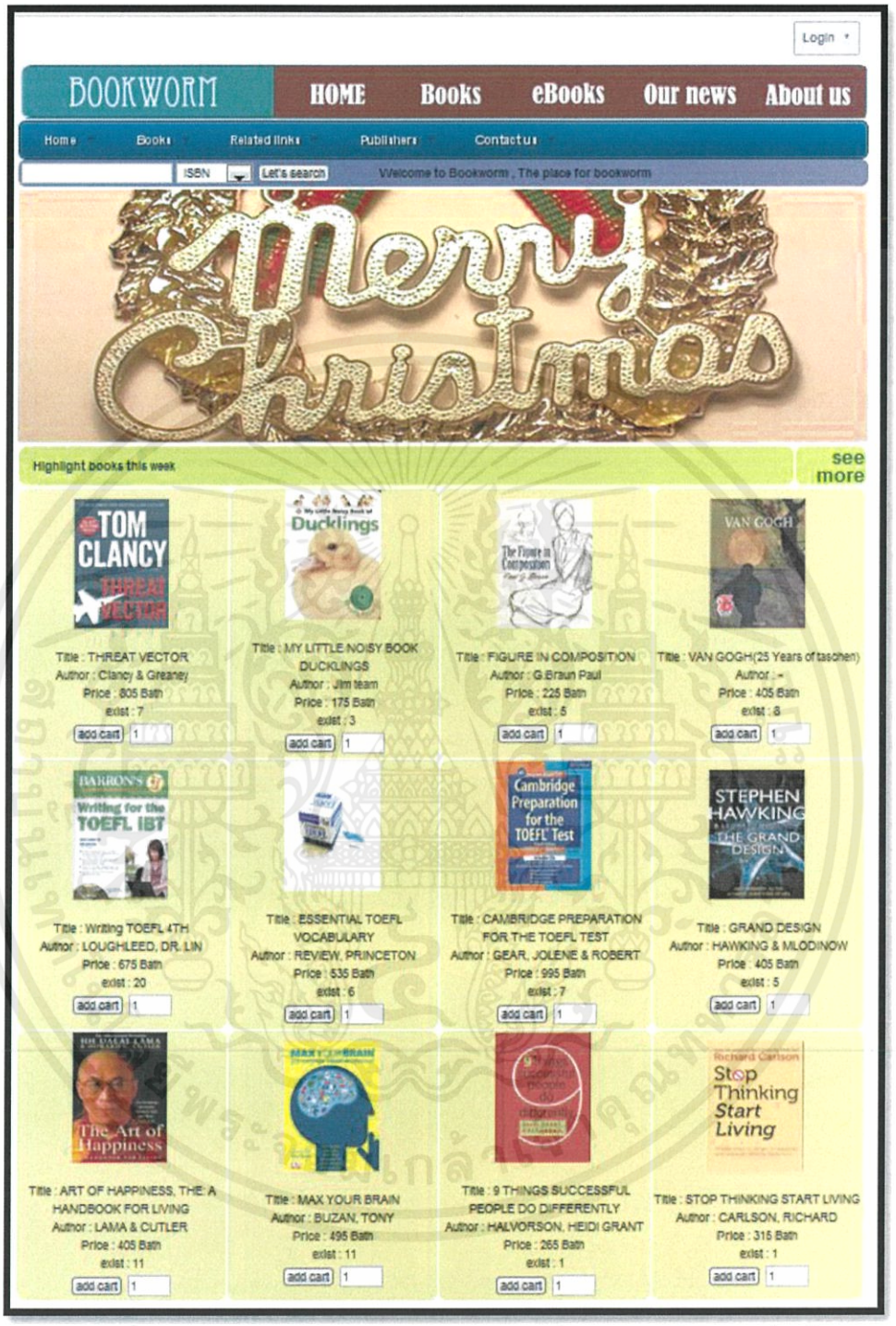
```
public class BuynowServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, SQLException
{
        PrintWriter out = response.getWriter();
        if (request.getSession().getAttribute("user")==null &&
request.getSession().getAttribute("password")==null){
            RequestDispatcher rd =
request.getRequestDispatcher("PleaseLogIn.jsp");
            rd.forward(request, response);
        } else {
        Cookie [] cookie = request.getCookies();
        Database db = new Database();
        db.connectionToDatabaseGuest();
        ShoppingCart cart = new ShoppingCart();
```

```
        Cookie mycookie = null;
        book book = null;
        Item item = null;
        List<String> checkname = new ArrayList<String>();

        db.beginTx();
        for (int i=1;i<cookie.length;i++){

            mycookie = cookie[i];
            checkname.add(mycookie.getName());
            book = db.findByID(mycookie.getName());
            item = new Item(book,
Integer.parseInt(mycookie.getValue()));
            cart.add(item);

        }
        int itemincart = cart.getLenght();
        boolean [] check = new boolean[cart.getLenght()];

        for (int a=0; a<itemincart ;a++){

db.findIDLock(Integer.toString(cart.getIndex(a).getISSBN()));


        }

        for (int x=0; x<itemincart;x++){

            check[x] =
db.updateBook(Integer.toString(cart.getIndex(x).getAmount()),
Integer.toString(cart.getIndex(x).getISSBN()));
        }

        int count = 0;
        for (int b =0 ; b<check.length;b++){
            if (check[b]){
                count = count+1;

            }
            else count = count;
        }
        if (count == check.length){
            //insert here ordernumber paymentcode date qty uid
            String user = (String)
request.getSession().getAttribute("user");
            String password = (String)
request.getSession().getAttribute("password");
            int uid = db.findUser(user, password);
            out.println(uid);
            db.InsertOrderTable(uid);
            int ordernum = db.getOrderNum();
            int numitem = cookie.length-1;
```

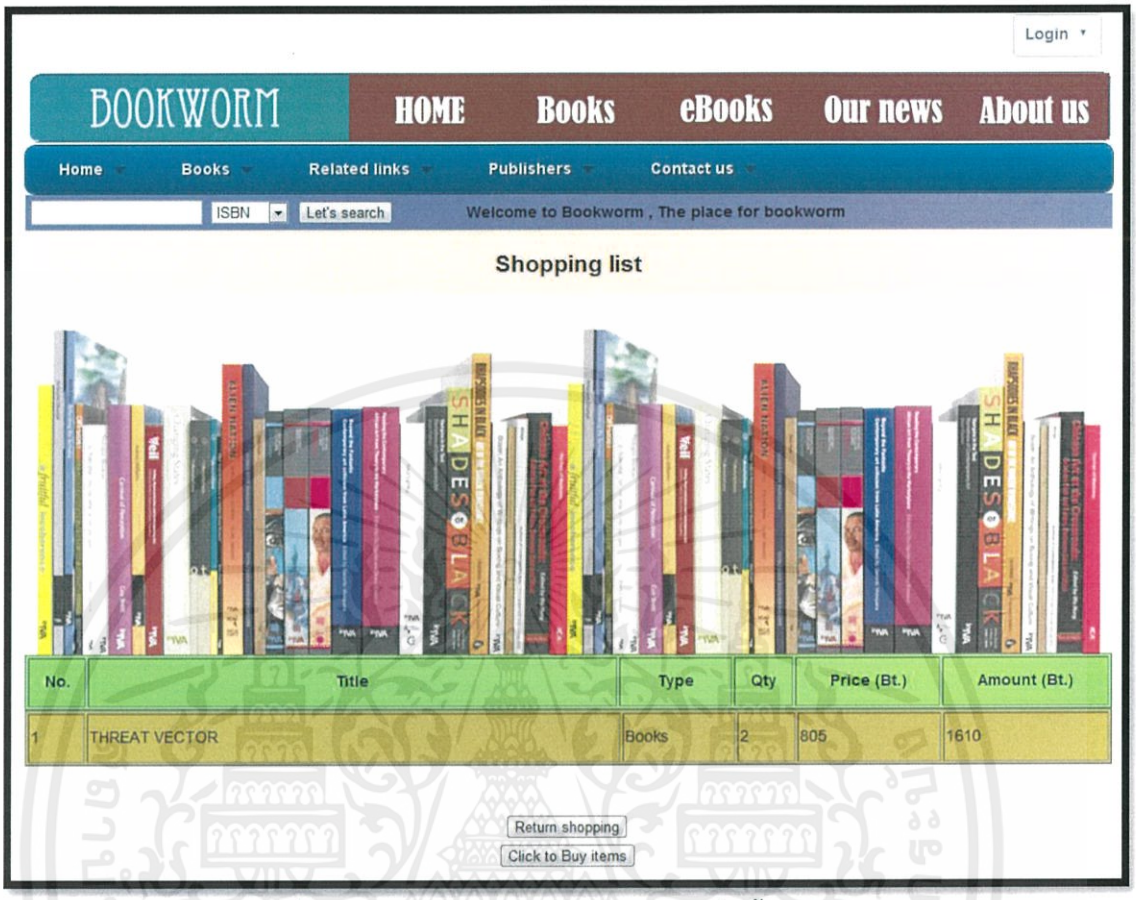
```
            for (int i =0; i< numitem ; i++){
                int ISBN = cart.getIndex(i).getISSBN();
                db.InsertbookOrderShip(ordernum, ISBN ,
cart.getIndex(i).getAmount());
                out.println(cart.getIndex(i).getTitle());
            }
            //
            db.commitResult();
            String Pycode = "abc"+Integer.toString(ordernum);
            String [] Title = new String[cart.getLenght()];
            int [] amount = new int[cart.getLenght()];
            int [] prize = new int[cart.getLenght()];
            for (int i = 0; i<numitem ; i++){
                Title[i] = cart.getIndex(i).getTitle();
                amount[i] =
Integer.parseInt(cookie[i+1].getValue());
                prize[i] =
Integer.parseInt(cart.getIndex(i).getPrice());
            }

            Receipt rt = new Receipt(Pycode, new
java.sql.Date(new java.util.Date().getTime()), uid, "test",
Title, amount ,prize);
            out.println("success");
            request.setAttribute("rt", rt);
            request.setAttribute("cart", cart);
            RequestDispatcher rd =
request.getRequestDispatcher("Receipt.jsp");
            rd.forward(request, response);
        }else{
            db.rollback();
            RequestDispatcher rd =
request.getRequestDispatcher("Txfail.jsp");
            rd.forward(request, response);
        }

    }
    }
```

เป็นคลาสหลักที่สำคัญที่สุดเนื่องจากเป็นคลาสที่ดูแลการสั่งซื้อหนังสือของลูกค้าหลายๆคน ในช่วงเวลาเดียวกัน ซึ่งคลาสนี้จะช่วยดูแลความถูกต้องของฐานข้อมูล และดูแลปัญหา The 4 concurrency control problems ด้วยเช่นกัน

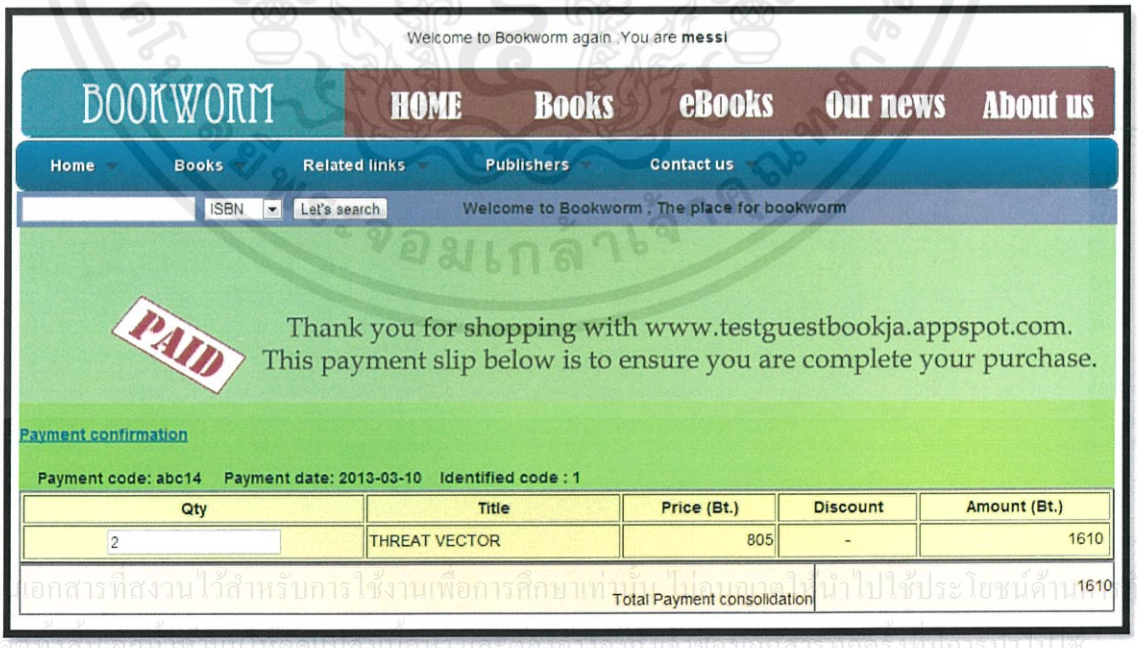
ในส่วนของการแสดงผลให้ผู้ใช้งานจะแสดงดังรูปต่อไปนี้



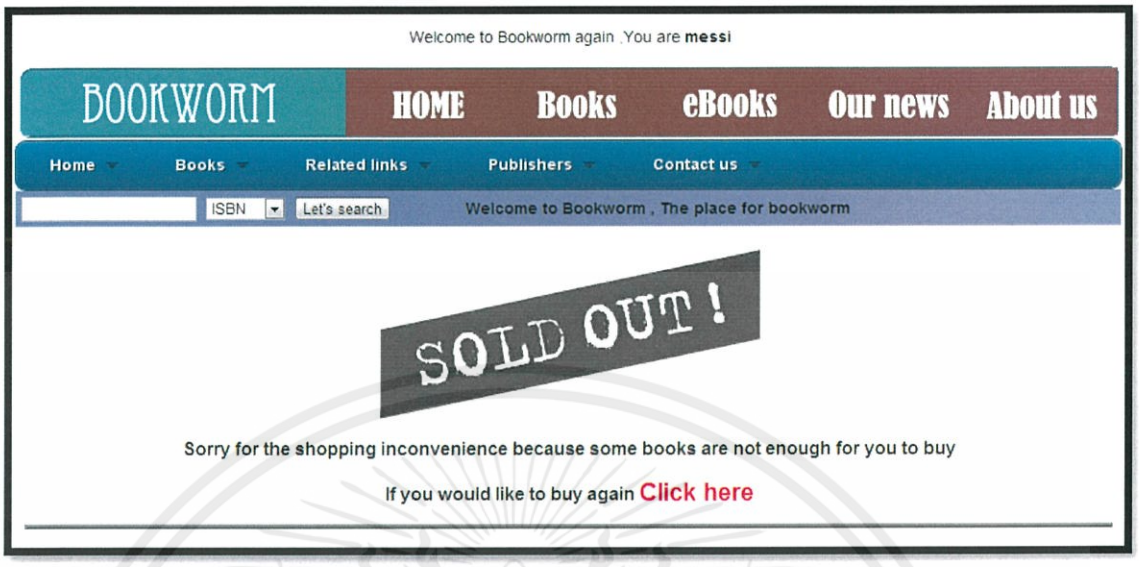
รูปที่ ข.2 ตัวอย่างหน้า Web Application ร้านขายหนังสือออนไลน์



รูปที่ ข.3 หน้า Web ก่อนเริ่มทรานแซกชั่นซื้อหนังสือ



รูปที่ ข.4 หน้า Web หลังจากทำทรานแซกชั่นสำเร็จ

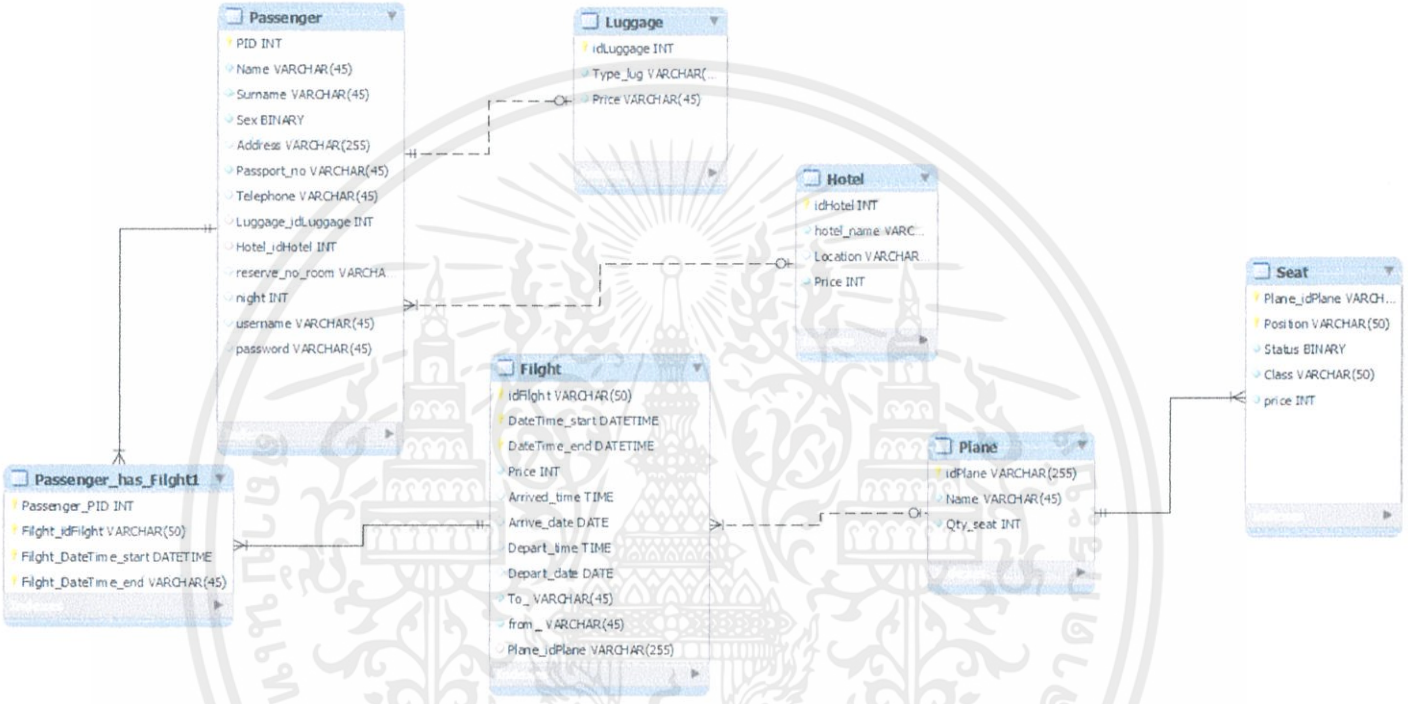


รูปที่ ข.5 หน้า Web หลังจากทำทรานแซกชั่นไม่สำเร็จ

2) Web Application ร้านจองตั๋วเครื่องบินออนไลน์

(เว็บไซต์ : http://airplaneonlineproject.appspot.com/)

ร้านขายหนังสือออนไลน์จะมีการออกแบบ ER Diagram ดังนี้



รูปที่ ข.6 ER Diagram ของ Web Application จองตั๋วเครื่องบินออนไลน์

ในส่วน Web Application จองตั๋วเครื่องบินออนไลน์นั้นจะเน้นไปที่การเลือกที่นั่งของผู้ใช้งานแต่ละคนนั้น ในเบื้องต้นแต่ละคนจะเห็นที่นั่งที่ว่างของเครื่องบินเป็นจำนวนเท่ากัน แต่ในการจองนั้นผู้ใช้งานที่เลือกยืนยันการจองที่นั่งก่อนจะได้รับสิทธิในการจองที่นั่งนั้นๆ โดยที่ผู้ใช้งานอีกคนหนึ่งจะไม่สามารถจองที่นั่งนั้นๆได้ โดยการพัฒนาจะใช้ MVC Model ในการพัฒนาเช่นเดียวกันกับ Web Application ซึ่งในส่วนของ Model ที่ติดต่อกับระบบฐานข้อมูลนั้นจะมี method ต่างๆที่คล้ายคลึงกับ method ของ Web Application ที่หนึ่งดังนี้

คำสั่งที่ ข.21 method ในการเชื่อมต่อระบบฐานข้อมูล

```
      public boolean connectDB() throws SQLException{
              try {
                  Class.forName("com.mysql.jdbc.Driver");
                  Connection c=
DriverManager.getConnection("jdbc:mysql://localhost/mydb",
"root","1234");
                  this.connection = c;
              return true;
              } catch (ClassNotFoundException ex) {
              return false;
              }
          }
```

คำสั่งที่ ข.22 method แสดงรายการสายการบินทั้งหมด

```
public List<fitstPageFlight> showFlight(){
        List<fitstPageFlight> flight = new
ArrayList<fitstPageFlight>();
        String SQL = "SELECT * from Filght";
        try {
            Statement St= connection.createStatement();
            ResultSet rs = St.executeQuery(SQL);
            while(rs.next()){
              fitstPageFlight FS= new
fitstPageFlight(rs.getString(1),

rs.getTimestamp(2),

rs.getTimestamp(3),

rs.getInt(4),

rs.getTime(5),

rs.getDate(6),

rs.getTime(7),

rs.getDate(8),

rs.getString(9),

rs.getString(10),
rs.getString(11));
              flight.add(FS);
            }
            return flight;
```

```
        } catch (SQLException ex) {

Logger.getLogger(getFirstInformation.class.getName()).log(Leve
l.SEVERE, null, ex);
          return null;

        }
    }
```

คำสั่งที่ ข.23 method เปิดทรานแซคชั่น

```
public void beginTx() {
        try {
            connection.setAutoCommit(false);
        } catch (SQLException ex) {

Logger.getLogger(getFirstInformation.class.getName()).log(Leve
l.SEVERE, null, ex);
        }
    }
```

คำสั่งที่ ข.24 method คำสั่งยืนยันการเปลี่ยนแปลงของทรานแซคชั่น

```
public void commitTx(){
        try {
            connection.commit();
        } catch (SQLException ex) {

Logger.getLogger(getFirstInformation.class.getName()).log(Leve
l.SEVERE, null, ex);
        }
    }
```

คำสั่งที่ ข.25 method ยกเลิกการเปลี่ยนแปลงของทรานแซคชั่น

```
public void rollbackTx() {
        try {
            connection.rollback();
        } catch (SQLException ex) {

Logger.getLogger(getFirstInformation.class.getName()).log(Leve
l.SEVERE, null, ex);
        }
    }
```

คำสั่งที่ ข.26 method เพิ่มข้อมูลการจองที่นั่งของแต่ละเที่ยวบิน

```
public String insertFilghtHasSeat(String idFlight, String
DateTime_start, String DateTime_end, String idPlane, String
Position, int PID){
        String SQL = "INSERT INTO filght_has_seat"
                + " (Filght_idFilght, Filght_DateTime_start,
Filght_DateTime_end, Seat_Plane_idPlane, Seat_Position,
Passenger_PID)"
                + " VALUES(?,?,?,?,?,?)";
        try {
            PreparedStatement pstmt =
connection.prepareStatement(SQL);
            pstmt.setString(1, idFlight);
            pstmt.setString(2, DateTime_start);
            pstmt.setString(3, DateTime_end);
            pstmt.setString(4, idPlane);
            pstmt.setString(5, Position);
            pstmt.setInt(6, PID);
            pstmt.executeUpdate();
            return "eiei";
        } catch (SQLException ex) {

Logger.getLogger(getFirstInformation.class.getName()).log(Leve
l.SEVERE, null, ex);
            return ex.toString();
        }
    }
```

ในส่วนของ Controler ในการควบคุมการการทำงานของ Request ที่เข้ามายัง Server จะใช้ Servlet Class ในการดูแลและควบคุมกลไลการทำงานต่างๆโดยที่จะแสดงส่วนของการทำงานที่เป็นส่วนสำคัญๆดังต่อไปนี้

คำสั่งที่ ข.27 คลาสในการค้นหาตารางเที่ยวบินที่ผู้ใช้ต้องการ

```
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
                throws ServletException, IOException,
SQLException {
            String getFrom= request.getParameter("getFrom");
            String getTo=request.getParameter("getTo");
            String
getDateDepart=request.getParameter("date");
            String getDateBack=request.getParameter("date2");
            String name2 = request.getParameter("name2");
            String amount = request.getParameter("amount");
            request.getSession().setAttribute("amount",
amount);
            PrintWriter out = response.getWriter();
```

```
                DateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                java.util.Date currentDate = new Date();
                String a = df.format(currentDate);
                String rest = a.substring(4);
                String years = a.substring(0, 4);
                int year = (Integer.parseInt(years))-543;
                String years2 = Integer.toString(year);
                String resultYear = years2.concat(rest);
                request.getSession().setAttribute("resultYear",
resultYear);

                    getFirstInformation GF = new
getFirstInformation();
                    if(name2.equals("R")){


              if(GF.connectDB()==true){
                    List<fitstPageFlight> flight=
GF.showFlight(getFrom, getTo,getDateDepart,resultYear,
amount);
                    List<fitstPageFlight>
flight2=GF.showFlight(getTo, getFrom, getDateBack,resultYear,
amount);
                    if(flight==null){
                        out.println("failed");
                        out.println(getDateDepart);
                    } else{
                            int total = flight.size();
                            int total2 = flight2.size();
                        request.setAttribute("Flight", flight);
                        request.setAttribute("Flight2", flight2);
                    request.setAttribute("total", total);
                    request.setAttribute("total2",total2);


                    RequestDispatcher rd =
request.getRequestDispatcher("resultFlight.jsp");

                    rd.forward(request, response);

                    GF.closeDB();
                     }
                }
                     } else if(name2.equals("O")){
                         if(GF.connectDB()==true){
                    List<fitstPageFlight> flight=
GF.showFlight(getFrom, getTo,getDateDepart,resultYear,
amount);

                     if(flight==null){
```

```
                    out.println("failed");
                    out.println(getDateDepart);
                } else{
                        int total = flight.size();

                    request.setAttribute("Flight", flight);

                request.setAttribute("total", total);

                RequestDispatcher rd =
request.getRequestDispatcher("returnFlightOneway.jsp");

                rd.forward(request, response);

                GF.closeDB();
                 }
            }
                }

        }
```

เป็นคลาสในการตรวจสอบเที่ยวบินที่ตรงตามความต้องการของผู้ใช้งาน โดยมีการแยกแยะว่าผู้ใช้เดินทางแบบเที่ยวเดียวหรือไปกลับ ซึ่งจะแสดงผลหน้าเพจแตกต่างกันออกไป

คำสั่งที่ ข.28 คลาสในการส่งต่อข้อมูลของผู้ใช้

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, SQLException
{

        PrintWriter out = response.getWriter();

        String departureFlightID =
request.getParameter("departureFlightID");
        String returnFlightID =
request.getParameter("returnFlightID");
        request.getSession().setAttribute("departureFlightID",
departureFlightID);
        request.getSession().setAttribute("returnFlightID",
returnFlightID);
        String currentTime = (String)
request.getSession().getAttribute("resultYear");
        if(departureFlightID==null || returnFlightID==null){
            response.setContentType("text/html");
            out.println("<script type=\"text/javascript\">");
            out.println("alert('please select both Dearting
and Returning Flight');");
            out.println("history.back();");
            out.println("</script>");
```

```
        }else {
        getFirstInformation GF = new getFirstInformation();
        if(GF.connectDB()==true){
            List<fitstPageFlight>
flight=GF.checkFlight(departureFlightID, currentTime);
            List<fitstPageFlight>
flight2=GF.checkFlight(returnFlightID, currentTime);
            List<Hotel> HT =
GF.checkHotel(flight.get(0).getTo());
            int noHotel = HT.size();

            request.setAttribute("flight", flight);
            request.setAttribute("flight2", flight2);
            request.setAttribute("Hotel", HT);
            request.setAttribute("noHotel", noHotel);

            RequestDispatcher rd =
request.getRequestDispatcher("Information.jsp");
            rd.forward(request, response);
            GF.closeDB();
        } else{
            out.println("failed");
        }

        }
    }
```

เป็นคลาสที่ใช้ส่งต่อข้อมูลการจองตั๋วของผู้ใช้ต่อไปยัง Servlet ตัวอื่นเพื่อประมวลผลทรานแซคชั่นต่อไป

คำสั่งที่ ข.29 method ทรานแซคชั่นการจองตั๋ว

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException, SQLException
{
        int x=1;
        PrintWriter out = response.getWriter();
        if (x==0){
            out.println(request.getParameter("inputret"+1));
        } else {
        String amount = (String)
request.getSession().getAttribute("amount");
        int amoun = Integer.parseInt(amount);
        String departureFlightID = (String)
request.getSession().getAttribute("departureFlightID");
        String returnFlightID = (String)
request.getSession().getAttribute("returnFlightID");
```

```
        String currentTime = (String)
request.getSession().getAttribute("resultYear");
        getFirstInformation GF = new getFirstInformation();
        GF.connectDB();
        List<fitstPageFlight>
flight=GF.checkFlight(departureFlightID, currentTime);
        List<fitstPageFlight>
flight2=GF.checkFlight(returnFlightID, currentTime);
        List<Integer> lugPrice = new ArrayList<Integer>();
        List<Integer> lugPriceret = new ArrayList<Integer>();
        String idhotelsend = (String)
request.getParameter("selectedHotel");
        String resevenumroomsend = (String)
request.getParameter("numroom");
        String nightsend = (String)
request.getParameter("night");
        int pricehotelsend = 0;
        List<Integer> positionprice = new
ArrayList<Integer>();
        List<Integer> positionpriceret = new
ArrayList<Integer>();
        GF.beginTx();
        if(idhotelsend.equals("0")){
            pricehotelsend = 0;
            resevenumroomsend = "0";
            nightsend = "0";
        } else {
            pricehotelsend = GF.getPriceHotel(idhotelsend);
        }
        int check = 0;
        int checkPID = 0;
        String checkLug = "check";
        String testLog = "long";
        for(int i=0; i<amoun; i++){

            String luggagePrice =
request.getParameter("Luggage"+(i+1));
            String luggageType;
            if(luggagePrice.equals("175")){
                luggageType = "+5kg";
            } else if(luggagePrice.equals("350")) {
                luggageType = "+10kg";
            } else if(luggagePrice.equals("525")){
                luggageType = "+15kg";
            } else if(luggagePrice.equals("700")){
                luggageType = "+20kg";
            } else if(luggagePrice.equals("875")) {
                luggageType = "+25kg";
            } else {
                luggageType = "No baggage";
            }
```

```
            boolean testLug = GF.insertLuggage(luggageType,
luggagePrice);
            if (testLug == false){
                check = check+1;
                checkLug = "Lug mai long";
            } else {
                check = check;
                int maxLug = GF.maxLuggage();
                int lugprice = GF.getPriceLuggage(maxLug);
                lugPrice.add(lugprice);
            }
            int maxLug = GF.maxLuggage();
            String lugid = Integer.toString(maxLug);
            String name = (String)
request.getParameter("Name"+(i+1));
            String surname = (String)
request.getParameter("Surname"+(i+1));
            String sex = (String)
request.getParameter("Sex"+(i+1));
            String sexx;
            if(sex.equals("true")){
                sexx="0";
            } else {
                sexx = "1";
            }
            String address = (String)
request.getParameter("Address"+(i+1));
            String passportnum = (String)
request.getParameter("PassportNum"+(i+1));
            String tel = (String)
request.getParameter("Tel"+(i+1));
            String idhotel = (String)
request.getParameter("selectedHotel");
            String resevenumroom = (String)
request.getParameter("numroom");
            String night = (String)
request.getParameter("night");
            if(i>0){
                resevenumroom = "with other";
                night = "0";
            } else {
                resevenumroom = resevenumroom;
                night = night;
            }
            boolean testpass;
            if(idhotel.equals("0")){
            testpass = GF.insertPassengerNoHotel(name,
surname, sexx, address, passportnum, tel, lugid);

            } else {
```

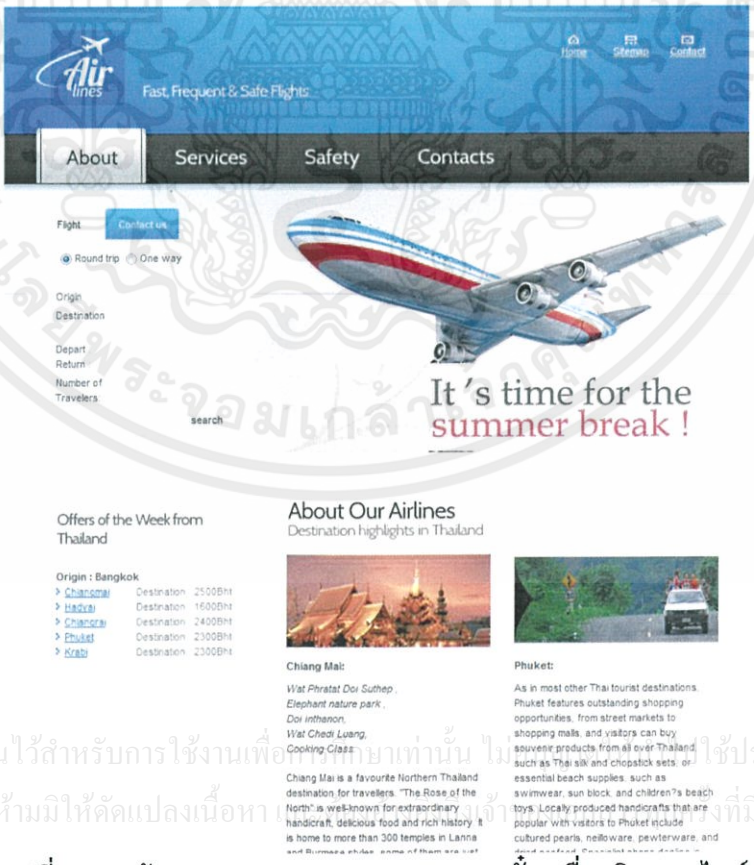
```
            testpass = GF.insertPassenger(name, surname, sexx,
address, passportnum, tel, lugid, idhotel, resevenumroom,
night);
            }
            if (testpass == false){
                check = check+1;
            } else {
                check = check;
            }
            int PID = GF.maxPID();

            String idflight = flight.get(0).getId();
            String datetimestart =
flight.get(0).getDatetime_s().toString().substring(0, 19);
            String datetimeend =
flight.get(0).getDatetime_e().toString().substring(0, 19);
            String idplane = flight.get(0).getPlaneid();
            String position = (String)
request.getParameter("input"+(i+1));
            String test = GF.insertFilghtHasSeat(idflight,
datetimestart, datetimeend, idplane, position, PID);
            if (test.equals("eiei")){
                check = check;
                checkPID = PID;
                int PSTprice = GF.getPositionPrice(idplane,
position);
                positionprice.add(PSTprice);

            } else {
                check = check+1;
                checkPID = PID;
                testLog = test;
            }
        }
        for(int i=0; i<amoun; i++){
            String luggagePrice =
request.getParameter("Luggageret"+(i+1));
            String luggageType;
            if(luggagePrice.equals("175")){
                luggageType = "+5kg";
            } else if(luggagePrice.equals("350")) {
                luggageType = "+10kg";
            } else if(luggagePrice.equals("525")){
                luggageType = "+15kg";
            } else if(luggagePrice.equals("700")){
                luggageType = "+20kg";
            } else if(luggagePrice.equals("875")) {
                luggageType = "+25kg";
            } else {
                luggageType = "No baggage";
            }
```

```
            boolean testLug = GF.insertLuggage(luggageType,
luggagePrice);
            if (testLug == false){
                check = check+1;
                checkLug = "Lug mai long";
            } else {
                check = check;
                int maxLug = GF.maxLuggage();
                int lugprice = GF.getPriceLuggage(maxLug);
                lugPriceret.add(lugprice);
            }
            int maxLug = GF.maxLuggage();
            String lugid = Integer.toString(maxLug);
            String name = (String)
request.getParameter("Name"+(i+1));
            String surname = (String)
request.getParameter("Surname"+(i+1));
            String sex = (String)
request.getParameter("Sex"+(i+1));
            String sexx;
            if(sex.equals("true")){
                sexx="0";
            } else {
                sexx = "1";
            }
            String address = (String)
request.getParameter("Address"+(i+1));
            String passportnum = (String)
request.getParameter("PassportNum"+(i+1));
            String tel = (String)
request.getParameter("Tel"+(i+1));

            boolean testpass = GF.insertPassengerNoHotel(name,
surname, sexx, address, passportnum, tel, lugid);
            if (testpass == false){
                check = check+1;
            } else {
                check = check;
            }
            int PID = GF.maxPID();

            String idflight = flight2.get(0).getId();
            String datetimestart =
flight2.get(0).getDatetime_s().toString().substring(0, 19);
            String datetimeend =
flight2.get(0).getDatetime_e().toString().substring(0, 19);
            String idplane = flight2.get(0).getPlaneid();
            String position = (String)
request.getParameter("inputret"+(i+1));
            String test = GF.insertFilghtHasSeat(idflight,
datetimestart, datetimeend, idplane, position, PID);
            if (test.equals("eiei")){
```

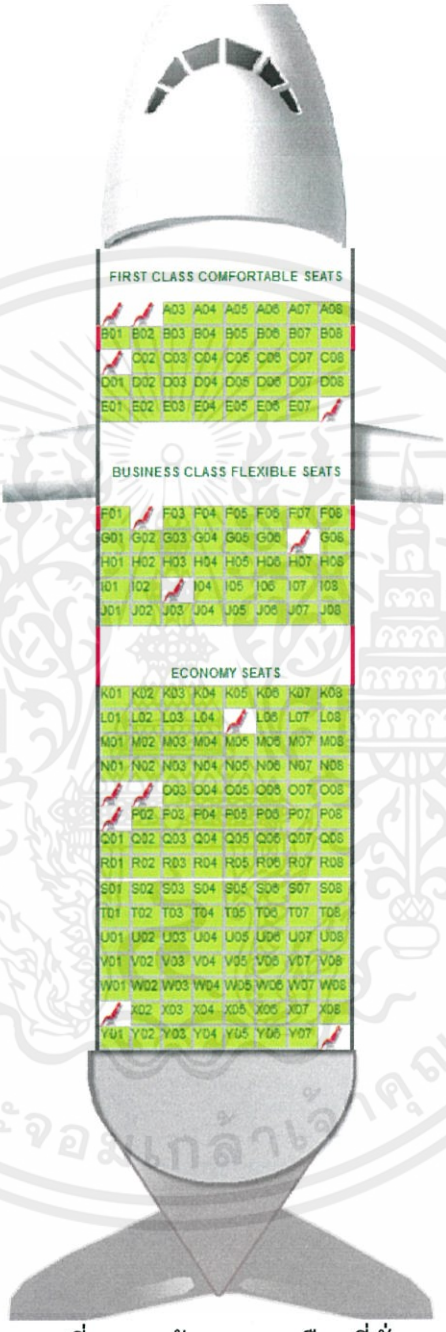
```
                check = check;
                checkPID = PID;
                int PSTprice = GF.getPositionPrice(idplane,
position);
                positionpriceret.add(PSTprice);

            } else {
                check = check+1;
                checkPID = PID;
                testLog = test;
            }


        }

        if(check == 0 ){
            int num = 0;
            for(int i=0 ; i<lugPrice.size(); i++ ){
                int numm = lugPrice.get(i);

                int numm2 = lugPriceret.get(i);
                num = num + numm + numm2;
            }

            GF.commitTx();
            request.setAttribute("lugPrice", lugPrice);
            request.setAttribute("lugPriceret", lugPriceret);

request.setAttribute("pricehotelsend",pricehotelsend);
            request.setAttribute("resevenumroomsend",
resevenumroomsend);
            request.setAttribute("nightsend", nightsend);
            request.setAttribute("positionprice",
positionprice);
            request.setAttribute("positionpriceret",
positionpriceret);
            request.setAttribute("flight", flight);
            request.setAttribute("flight2", flight2);
            String namesend = (String)
request.getParameter("Name"+(1));
            String surnamesend = (String)
request.getParameter("Surname"+(1));
            request.setAttribute("namesend", namesend);
            request.setAttribute("surnamesend", surnamesend);
            RequestDispatcher rd =
request.getRequestDispatcher("BillingPage.jsp");
            rd.forward(request, response);
        } else {
            out.println("failed");
            GF.rollbackTx();
            out.println(check);
            out.println(checkPID);
```

```
            out.println(checkLug);
            out.println(flight.get(0).getId());
out.println(flight.get(0).getDatetime_s().toString().substring
(0, 19));

out.println(flight.get(0).getDatetime_e().toString().substring
(0, 19));
            out.println(flight.get(0).getPlaneid());
            out.println(testLog);

        }
        }
    }
```

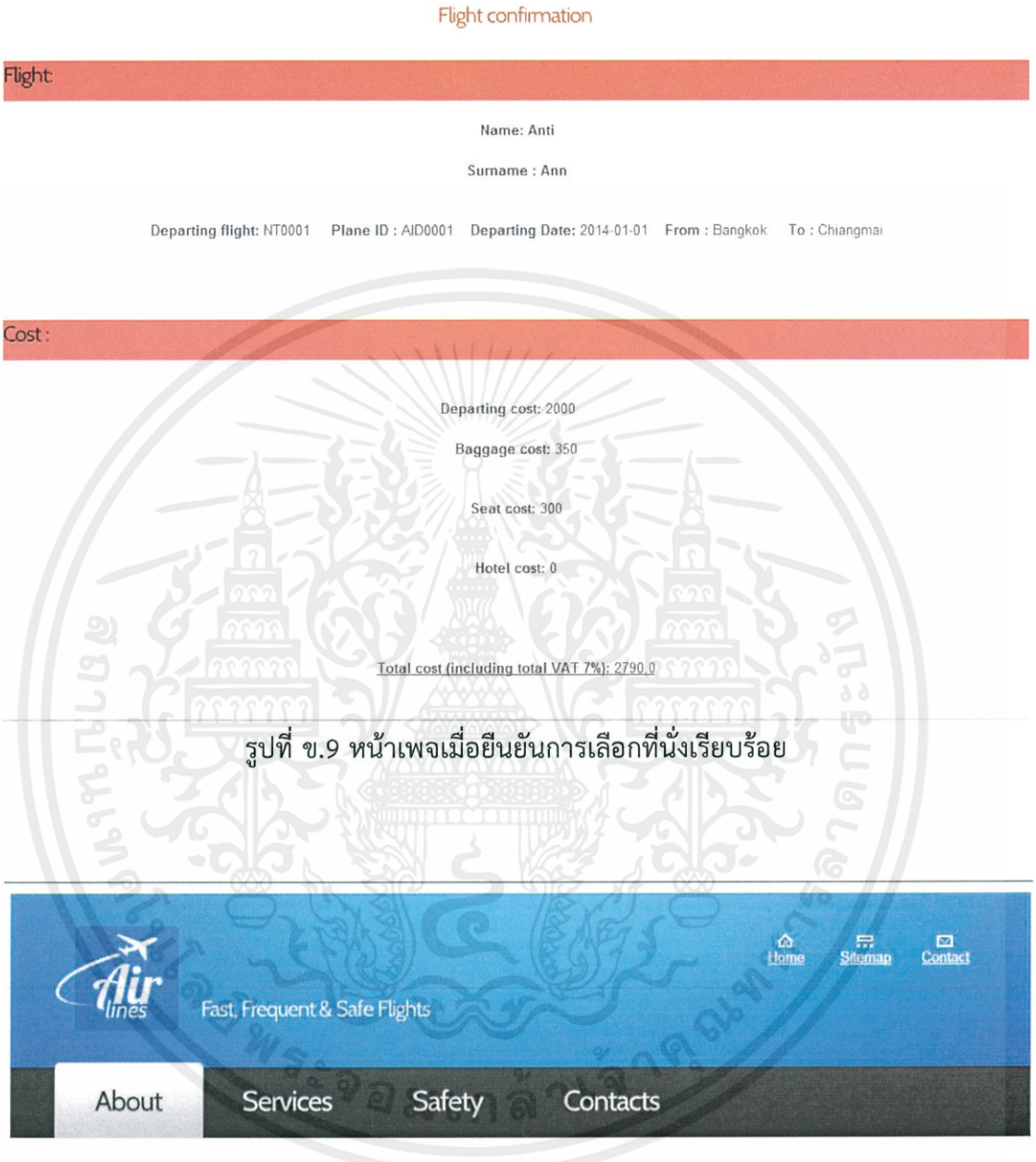
เป็นคลาสส่วนที่สำคัญที่สุดในการจองตั๋วเครื่องบินโดยจะมีการตรวจสอบเที่ยวบินและที่นั่งที่ผู้ใช้ต้องการก่อนที่จะเก็บข้อมูลลงในฐานข้อมูล ซึ่งตรงจุดนี้ Web Application จะใช้ Isolation Level ระดับ Repeatable Read เพื่อให้ผู้ใช้งานทั้งสองคนมองเห็นที่นั่งที่ว่างเป็นจำนวนและตำแหน่งเดียวกัน ซึ่งส่งผลให้ผู้ที่ยืนยันการจองก่อน (Request Servlet ด้านบน) จะได้รับสิทธิ์ในที่นั่งนั้นแต่เพียงผู้เดียว ในส่วนของการแสดงผลหน้าเว็บเพจมีตัวอย่างดังต่อไปนี้



รูปที่ ข.7 หน้าเพจ Web Application จองตั๋วเครื่องบินออนไลน์



รูปที่ ข.8 หน้าเพจการเลือกที่นั่ง



รูปที่ ข.9 หน้าเพจเมื่อยืนยันการเลือกที่นั่งเรียบร้อย

รูปที่ ข.10 หน้าเพจหลังยกเลิกทรานแซกชั่นได้สำเร็จ