บทที่ 9

การจัดการทรานแซคชัน และการควบคุมภาวะพร้อมกัน

(Transaction Management and Concurrency Control)

เป็นที่ทราบกันดีว่า การจัดเก็บข้อมูลรวมเป็นฐานข้อมูลไว้ด้วยกัน ก่อให้เกิดประโยชน์หลายอย่าง เช่น ลดความซ้ำซ้อนของข้อมูล หลีกเลี่ยงความขัดแย้งของข้อมูล รักษาความถูกต้องเชื่อถือได้ของข้อมูล กำหนดความเป็นมาตรฐานเดียวกัน กำหนดระบบความปลอดภัยของข้อมูล และที่สำคัญคือ หากผู้ใช้ต้องการใช้ข้อมูลในฐานข้อมูลที่มาจากแฟ้มข้อมูลต่าง ๆ ก็สามารถที่จะทำได้ง่าย

การประมวลผลที่เกิดขึ้นภายในฐานข้อมูล ได้แก่การอ่าน และ การเขียน (Read & Write) ข้อมูล มีผลโดยตรงต่อความคงสภาพของข้อมูล ผลลัพธ์จากการทำงานบางอย่างของฐานข้อมูลจะทำให้เกิดความเสียหายกับข้อมูล หรืออาจจะทำให้ระบบล้มเหลว ซึ่งจะมีผลทำให้ข้อมูลในฐานข้อมูลอยู่ในสถานะที่ไม่มั่นคง (Unconsistency) จำเป็นอย่างยิ่งที่ระบบจัดการฐานข้อมูลจะต้องรักษากฎเกณฑ์เกี่ยวกับการทำงานในฐานข้อมูลเอาไว้ เพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้น

การใช้งานฐานข้อมูลหากมีผู้ใช้ข้อมูลเพียงคนเดียว ย่อมจะไม่มีปัญหา แต่วัตถุประสงค์ของการมีฐานข้อมูลก็เพื่อที่จะให้มีการใช้งานข้อมูลร่วมกัน (Shared Data) ด้วยการรองรับการทำงานในลักษณะแบบผู้ใช้หลายคน จึงมักจะพบเสมอว่าการที่มีผู้เรียกใช้ข้อมูลพร้อมกัน ในช่วงเวลาเดียวกัน อาจจะทำให้การประมวลผลเกิดการสอดแทรกระหว่างผู้ใช้แต่ละคนบนชุดข้อมูลเดียวกัน และเมื่อต้องการปรับปรุงข้อมูลให้ถูกต้อง จะต้องดำเนินการให้เสร็จก่อนที่ข้อมูลนั้นจะถูกเรียกใช้โดยผู้ใช้คนอื่น ดังนั้นถ้าไม่มีการควบคุมการเข้าถึงข้อมูลที่ดี อาจจะก่อให้เกิดความเสียหายกับข้อมูลที่เกิดขึ้นในระหว่างการประมวลผล ผลที่ตามมาคือข้อมูลจะไม่สอดคล้องกับความเป็นจริง ทำให้ผู้ใช้คนอื่นได้ข้อมูลที่มีค่าไม่ถูกต้อง

ข้อมูลเป็นทรัพยากรสำคัญขององค์กร การป้องกันการสูญหายข้อมูลและวิธีการในการกู้ข้อมูลกลับคืนมาใช้ จึงเป็นปัจจัยหลักอีกอย่างหนึ่งที่ผู้ออกแบบจะต้องคำนึงถึงและให้ความสำคัญเมื่อทำการออกแบบฐานข้อมูล ถ้าเทคนิคและวิธีการกู้คืนข้อมูลมีประสิทธิภาพ จะทำให้ข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลมีความถูกต้องอยู่เสมอและสูญหายได้ยาก สิ่งสำคัญจึงอยู่ที่เรื่องของข้อมูล ไม่ว่าจะเป็นข้อมูลในหน่วยความจำหลัก หรือในหน่วยความจำสำรอง หากระบบคอมพิวเตอร์เกิดขัดข้องก็

อาจจะมีผลกระทบถึงข้อมูลได้ จำเป็นที่จะต้องทำซ้ำใหม่ หรืออาจต้องทำการกู้คืนข้อมูล (Recovery) เพื่อทำให้ข้อมูลอยู่ในสภาพที่ถูกต้องใช้งานได้ ในบทนี้ผู้เรียบเรียงจะได้กล่าวถึงเนื้อหาเกี่ยวกับแนวคิดของทรานแซคชัน (Transaction) ซึ่งเป็นพื้นฐานของการควบคุมภาวะการทำงานพร้อมกัน (Concurrency Control) และการกู้คืนข้อมูลจากความล้มเหลวของระบบที่จัดการโดยระบบฐานข้อมูล

1. ทรานแซคชัน หรือ รายการการดำเนินการ (Transaction)

1.1 นิยามหรือความหมายทรานแซคชัน

ทรานแซคชัน หมายถึง*คำสั่ง* อาจจะเป็นคำสั่งเดียวหรือหลายคำสั่งก็ได้ หรือเป็น*หน่วย*ของการทำงานในเชิงตรรกะ ที่ใช้จัดการกับข้อมูลในฐานข้อมูลในลักษณะต่าง ๆ เช่นการเพิ่ม การลบ และการแก้ไขข้อมูล เพื่อเปลี่ยนแปลงสถานะของขอมูลจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง ชุดของคำสั่งหรือหน่วยของการทำงานดังกล่าวเมื่อมีการประมวลผลกลุ่มหน่วยการทำงานหรือเมื่อทำงานสำเร็จเรียบร้อยแล้ว ข้อมูลในฐานข้อมูลจะต้องอยู่ในสภาพที่สมบูรณ์ถูกต้อง

ทรานแซคชัน อาจจะหมายถึงงานที่ต้องจัดการกับข้อมูลให้เสร็จสิ้น ไปในคราวเดียวกัน ดังนั้นในทรานแซคชันหนึ่ง ๆ จะประกอบไปด้วย กลุ่มคำสั่งหลาย ๆ คำสั่งที่สร้างขึ้นเพื่อประมวลผลข้อมูลเรื่องใดเรื่องหนึ่ง การสร้างทรานแซคชัน จะสามารถช่วยควบคุมความถูกต้องสมบูรณ์ของข้อมูลเมื่อมีการเข้าถึงข้อมูลพร้อม ๆ กัน โดยเฉพาะงานที่เกี่ยวข้องกับการแก้ไขข้อมูลในหลาย ๆ ตาราง

1.2 คุณสมบัติทรานแซคชัน (Transaction Properties)

ภายในทรานแซคชันประกอบด้วยชุดการกระทำ (Operation) ต่างๆ เรียงกันเป็นลำดับ ทรานแซคชันหนึ่งอาจจะประกอบไปด้วยงานย่อย ๆ หลายงาน และถ้างานย่อย ๆ บางงานประมวลผลไม่สำเร็จ จะก่อให้เกิดความผิดพลาดกับงานในระดับเชิงตรรกะได้ ดังนั้นเพื่อให้เกิดความมั่นใจว่าการทำงานใด ๆ กับข้อมูลในฐานข้อมูลเป็นไปอย่างถูกต้องสมบูรณ์ จึงมี การกำหนดคุณสมบัติของทรานแซคชัน ที่นิยมเรียกกันย่อ ๆ ว่า **ACID** ขึ้นมา ดังนี้

1.2.1 ความเป็นหนึ่งเดียว (Atomicity: **A**) หมายถึงทรานแซคชันจะให้ผลการทำงานเป็นหนึ่งเดียว หรือการทำงานเสร็จอย่างสมบูรณ์ ไม่ว่าทรานแซคชันจะทำงานปกติ หรือมีปัญหาก็ตาม หมายความว่า “ทุกขั้นตอนของทรานแซคชันจะต้องมีการทำงานที่เสร็จสมบูรณ์ หรือไม่ เช่นนั้นก็ต้องยกเลิกทุกขั้นตอนในทรานแซคชันนั้น”

นั่นคือทุก ๆ คำสั่งในทรานแซคชันจะต้องมีการประมวลผล หรือทำงานที่เสร็จสมบูรณ์ตั้งแต่เริ่มต้นจนจบกระบวนการทรานแซคชันนั้น ๆ หากมีงานบางส่วนในทรานแซคชันไม่สมบูรณ์จะต้องยกเลิก (Aborted) ทุกการกระทำในทรานแซคชันนั้น คือจะต้องย้อนกลับ (Rolled Back) ไปยังจุดเริ่มต้นใหม่

1.2.2 ความถูกต้อง (Consistency : **C**) หมายถึง ระบบจะรักษาข้อมูลที่เกิดจากการทำงานของทรานแซคชันให้ถูกต้องและสอดคล้องกันเสมอ นั่นคือหลังจากการทำงานของทรานแซคชันเสร็จสมบูรณ์ ทุกทรานแซคชันจะต้องก่อให้เกิดข้อมูลในฐานข้อมูลที่มีความถูกต้องอยู่เสมอ

เนื่องจากการประมวลผลคำสั่งของทรานแซคชันจะทำให้เกิดการเปลี่ยนแปลงสถานะของฐานข้อมูล ดังนั้นทุก ๆ ทรานแซคชันจะต้องรักษาความถูกต้องของข้อมูลในฐานข้อมูลไว้ (Integrity Constraints) นั่นคือไม่ว่าการประมวลผลคำสั่งในทรานแซคชันจะสมบูรณ์หรือไม่สมบูรณ์ก็ตาม ข้อมูลในฐานข้อมูลต้องมีความถูกต้องตรงกันอยู่เสมอ ถ้าการประมวลผลในทรานแซคชันไม่สมบูรณ์ จะต้องคืนสภาพกลับไปยังจุดเริ่มต้นก่อนที่จะทำทรานแซคชัน

1.2.3 ความเป็นอิสระ (Isolation : **I**) เป็นคุณสมบัติที่เกี่ยวข้องกับผลกระทบในการกระทำพร้อม ๆ กันของทรานแซคชัน ทรานแซคชันที่ทำงานพร้อมกันในเวลาเดียวกันโดยผู้ใช้หลายคนจะต้องไม่รบกวนกัน นั่นคือแต่ละทรานแซคชันจะต้องมีความเป็นอิสระต่อกัน คือแยกออกจากกันอย่างสิ้นเชิง ซึ่งจะทำให้มั่นใจได้ว่าการประมวลผลคำสั่งในทรานแซคชันหนึ่ง จะไม่มีผลกระทบต่อการประมวลผลในอีกทรานแซคชันหนึ่ง ทำให้แต่ละทรานแซคชันไม่สามารถเปลี่ยนแปลงข้อมูลได้พร้อมกัน จะต้องรอให้ทรานแซคชันก่อนหน้าที่เรียกใช้ข้อมูลเดียวกันเปลี่ยน แปลงค่าของข้อมูลให้เสร็จสิ้นก่อนเพื่อที่จะเป็นไปตามลำดับของทรานแซคชันที่ถูกต้อง และไม่ส่งผลกระทบต่อข้อมูลที่ไม่ถูกต้องในภายหลังได้

1.2.4 ความคงทน (Durability : **D**) หมายถึงเมื่อทรานแซคชันทำงานเสร็จสิ้นระบบจะมีการบันทึกข้อมูลลงในฐานข้อมูล นั่นคือเมื่อทรานแซคชันถูกประมวลผลจนสมบูรณ์แล้วสามารถมั่นใจได้ว่าข้อมูลจะถูกปรับปรุง และบันทึกลงในฐาน ข้อมูลอย่างถูกต้องเช่นกัน

ความคงทนเป็นคุณสมบัติที่เมื่อทรานแซคชันได้ทำงานหรือทำให้เกิดการเปลี่ยนแปลงของขอมูลในฐานขอมูลเสร็จสมบูรณ ระบบจะต้องสามารถคงข้อมูลที่เปลี่ยนแปลงนั้นไว้ได้ ถึงแม้ว่าภายหลังอาจจะเกิดเหตุการณ์ที่ก่อให้เกิดความเสียหาย

และหากเกิดเหตุการณ์ที่ไม่คาดคิดขึ้นอันจะส่งผลให้ระบบล่มในระหว่างที่ทรานแซคชันนั้นกำลังประมวลผลอยู่ ระบบจะต้องสามารถย้อนกลับด้วยการยกเลิกการทำงานที่ผ่านมาทั้งหมดได้ (เปรียบเสมือนว่ายังไม่มีเหตุการณ์ใด ๆ เกิดขึ้น) เมื่อทรานแซคชันถูกประมวลผลจนสมบูรณ์แล้ว สามารถมั่นใจได้ว่าข้อมูลจะถูกปรับปรุง และบันทึกลงในฐานข้อมูลอย่างถูกต้องเช่นกัน

จะเห็นได้ว่าการทำงานของระบบฐานข้อมูล อาจจะถูกสอดแทรกการกระทำต่าง ๆ จากหลาย ๆ ทรานแซคชัน เพื่อไม่ให้เกิดผลกระทบดังกล่าว ระบบฐานข้อมูลจะต้องมีวิธีการจัดการการทำงานของทรานแซคชันที่ก่อให้เกิดสภาวะการทำงานพร้อมกัน ซึ่งถือเป็นสิ่งสำคัญต่อการจัดการทรานแซคชัน ดังนั้นจึงต้องมีการควบคุมสภาวะการทำงานพร้อมกัน และ การกู้คืนฐานข้อมูลด้วย

1.3 สถานะของทรานแซคชัน (Transaction State)

ในระหว่างที่ทรานแซคชันกำลังทำงาน จะมีสถานะต่าง ๆ ได้ดังนี้

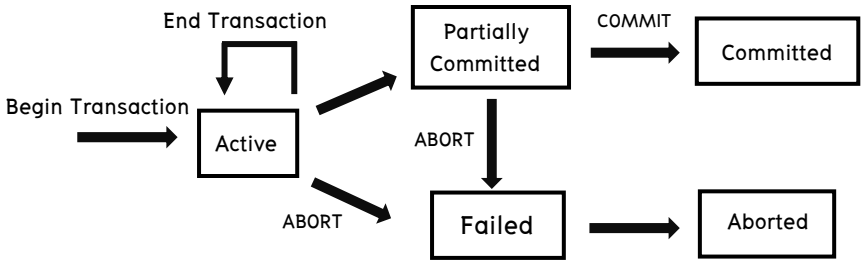
1.3.1 *Active* คือ สถานะ*เริ่มต้น*การทำงานของทรานแซคชัน

1.3.2 *Partially Committed* คือ สถานะที่แต่ละการกระทำของทรานแซคชันเป็นไปอย่าง*ถูกต้อง*

1.3.3 *Failed* คือ สถานะที่มี*ข้อผิดพลาด*เกิดขึ้นกับทรานแซคชันนั้น

1.3.4 *Committed* คือ สถานะที่ทรานแซคชันทำงาน*เสร็จสมบูรณ์*

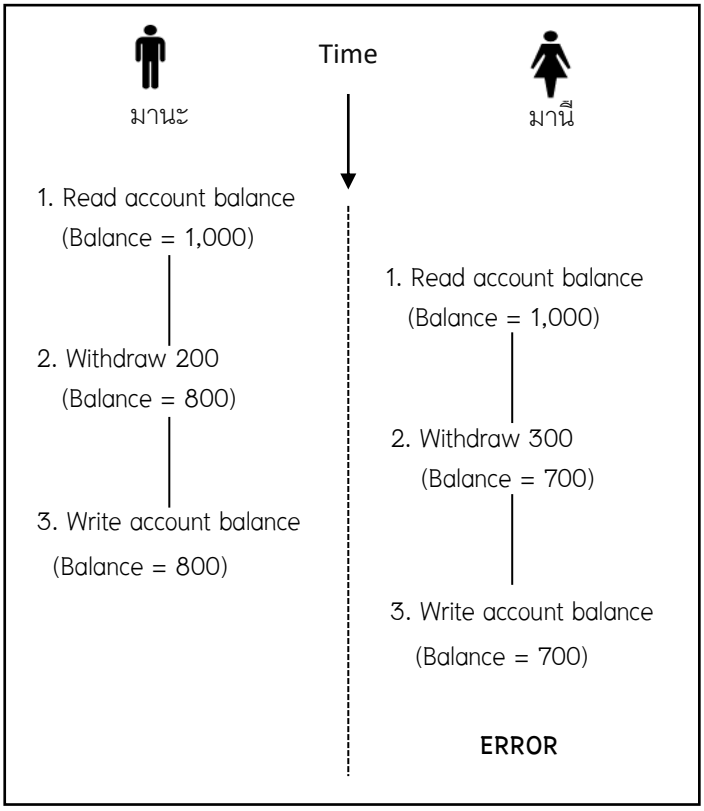
1.3.5 *Aborted* คือ สถานะที่ทรานแซคชัน*หยุดทำงาน* จะเกิดขึ้นหลังจากที่เกิดสถานะ *Failed* เพื่อ*หยุดการทำงาน*ของทรานแซคชัน และจะมีการ Rolled Back เพื่อทำให้ข้อมูลในฐานข้อมูลคืนสู่สภาพเดิมก่อนเริ่มของทรานแซคชัน



ภาพที่ 9.1 แสดงสถานะต่าง ๆ ของทรานแซคชันระหว่างการประมวลผล

ภาพที่ 9.1 อธิบายสถานะของทรานแซคชันระหว่างการประมวลผลได้ ดังนี้

- ทรานแซคชันจะเข้าสู่ สถานะ Active ทันทีที่เริ่มทำงาน
- เมื่อทรานแซคชันทำงานครบทุกคำสั่ง จะเข้าสู่สถานะ Partially Committed ณ จุดนี้จะมีการตรวจสอบความสมบูรณ์ของทรานแซคชัน
- ถ้าการทำงานของทรานแซคชันไม่มีปัญหา และเสร็จสมบูรณ์ จะเข้าสู่สถานะ Committed นั่นคือการเปลี่ยนแปลง หรือการแก้ไขต่าง ๆ กับข้อมูลในฐานข้อมูลจะคงอยู่อย่างถาวร
- ถ้าพบว่าการทำงานของทรานแซคชันที่อยู่ในระหว่างสถานะ Active เกิดมีปัญหา เช่นทรานแซคชันล้มเหลว หรือถูกยกเลิก ทรานแซคชันจะเข้าสู่สถานะ Failed ได้ หลังจากนั้นจะเข้าสู่สถานะ Aborted เพื่อยกเลิก การทำงานทั้งหมดของทรานแซคชัน



ภาพที่ 9.2 ภาวะการเกิดพร้อมกันระหว่างทรานแซคชันมานะและมานี

จากภาพ จะเห็นว่า มานะ และ มานี ต่างก็มีรายการทรานแซคชันที่ต้องการเข้าถึงข้อมูลบัญชีเดียวกันในการถอนเงิน ลำดับเหตุการณ์เป็นขั้นตอนดังนี้

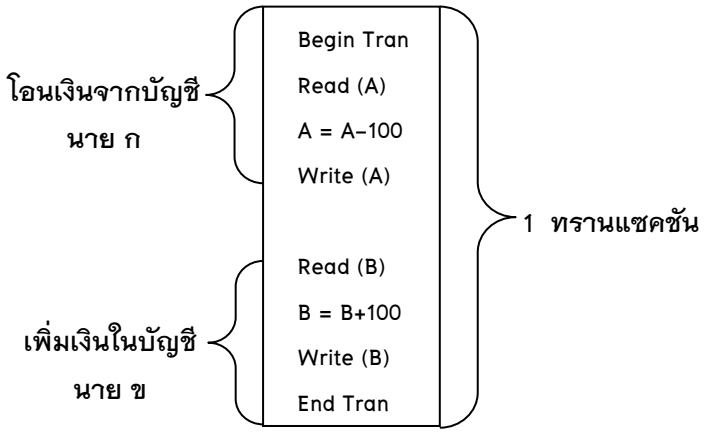
ขั้นตอนที่ 1 ทรานแซคชันของ มานะ อ่านยอดคงเหลือในบัญชีลูกค้าจำนวน 1,000 บาท หลังจากช่วงเวลานั้นไม่นาน ทรานแซคชันของ มานี ก็ได้อ่านยอดคงเหลือในบัญชีจำนวน 1,000 บาทจากบัญชีของลูกค้าคนเดียวกันกับที่ มานะ กำลังดำเนินการอยู่

ขั้นตอนที่ 2 มานะ ได้ถอนเงินจากบัญชีจำนวน 200 บาท ทำให้ยอดเงินคงเหลือสุทธิ 800 บาท ในขณะที่ มานี ก็ได้ถอนเงินจำนวน 300 บาท ทำให้ยอดเงินคงเหลือสุทธิ 700 บาท

ขั้นตอนที่ 3 มานะ บันทึกยอดคงเหลือสุทธิ 800 ลงในบัญชี ในขณะที่มานี ก็ทำการบันทึกยอดคงเหลือในลำดับหลังสุด จึงทำให้ยอดคงเหลือสุทธิในบัญชีมีค่าเท่ากับ 700 บาท ซึ่งถือว่าเป็นข้อมูลที่ไม่ถูกต้อง

ตัวอย่างที่ 1 : การโอนเงินที่ตู้ธนาคารแห่งหนึ่ง ระบบมีคำสั่งทำงานเป็น ขั้นตอนดังนี้

ขั้นตอนที่ 1 เริ่มต้นทรานแซคชัน
ขั้นตอนที่ 2 อ่านบัตร ATM
ขั้นตอนที่ 3 ตรวจสอบรหัสผ่าน
ขั้นตอนที่ 4 รับและกำหนดค่ายอดเงินโอน
ขั้นตอนที่ 5 หักยอดเงินต้นทาง
ขั้นตอนที่ 6 ตรวจสอบบัญชีปลายทาง
ขั้นตอนที่ 7 เพิ่มยอดเงินให้บัญชีปลายทาง
ขั้นตอนที่ 8 สิ้นสุดทรานแซคชัน



ภาพที่ 9.3 ตัวอย่างทรานแซคชันการโอนเงิน

จากตัวอย่างข้างบนจะเห็นว่าทรานแซคชันการโอนเงินหนึ่งทรานแซคชัน มีขั้นตอนที่ต้องทำให้สำเร็จ 8 ขั้นตอนด้วยกัน สมมุติว่าถ้าเกิดกระแสไฟฟ้าขัดข้องขึ้นในขั้นตอนที่ 5 คือหลังจากหักยอดเงินจากบัญชีนาย ก สำเร็จแล้ว แต่ยังไม่ได้นำกลับไปใส่ให้กับบัญชีนาย ข จึงเป็นการทำงานที่ไม่ถูกต้อง เพื่อให้เกิดความมั่นใจว่า การทำงานใด ๆ กับข้อมูลในฐานข้อมูลให้เป็นไปอย่างถูกต้อง จึงต้องมีการกำหนดคุณสมบัติของทรานแซคชัน 4 ประการ หรือ ACID เอาไว้ดังที่กล่าวมาแล้วข้างบน ดังนั้นหากมีการเขียนทรานแซคชันเอาไว้แล้ว ถ้าคำสั่งใดคำสั่งหนึ่งในทรานแซคชันทำงานไม่สำเร็จ ระบบจะย้อนกลับไปยังจุดเริ่มต้นของทรานแซคชันเสมือนว่าเหตุการณ์ทั้งหมดไม่ได้เกิดขึ้น หรือยังไม่ได้โอนเงินนั่นเอง เพิ่มเงินในบัญชีนาย ข

1.4 ทรานแซคชัน โอเพอเรชัน (Transaction Operations)

จากความรู้เบื้องต้น และจากนิยามของทรานแซคชันที่กล่าวมาแล้ว สามารถสรุป ทรานแซคชันโอเพอเรชัน ได้ดังนี้

1.4.1 *Begin Transaction* “คำสั่งเริ่มประมวลผลของทรานแซคชัน” เป็นคำสั่งแรกสุดของแต่ละทรานแซคชัน เพื่อกำหนดจุดเริ่มต้นของทรานแซคชัน

1.4.2 Read หรือ Write Transaction: “คำสั่งที่ใช้ในการอ่าน การเขียนหรือการบันทึกข้อมูลบนฐานข้อมูล” ซึ่งถือว่าเป็นส่วนหนึ่งของการ Execute Transaction

1.4.3 End Transaction : “คำสั่งที่จบการอ่านหรือการเขียนบันทึกข้อมูลใน ฐานข้อมูล” ตรงจุดนี้จะมีการตรวจสอบการเปลี่ยนแปลงของทรานแซคชันว่าการปรับปรุงข้อมูลสำเร็จสมบูรณ์ หรือถูกยกเลิกเพื่อเตรียมย้อนกลับการทำงาน

1.4.4 *Commit Transaction* : “คำสั่งสิ้นสุดทรานแซคชัน” ทำหน้าที่ในการนำเอาข้อมูลที่แก้ไขแล้วบันทึกกลับลงฐานข้อมูล ใช้ในการยืนยันผลการเปลี่ยนแปลงที่เกิดขึ้นกับฐาน ข้อมูล เพื่อให้ผลลัพธ์นั้นคงอยู่ตลอดไปจนกว่าจะมีคำสั่งในการเปลี่ยนแปลงข้อมูล นั่นคือถ้าการทำงานของทรานแซคชันเสร็จสมบูรณ์ ข้อมูลที่ทำการปรับปรุงในฐานข้อมูลจะไม่สามารถย้อนการทำงานกลับ (Undone) ได้

1.4.5 *Rollback Transaction* : เป็นคำสั่งในการยกเลิกทรานแซคชัน เป็นการส่งสัญญาณถึงความไม่สมบูรณ์ของทรานแซคชัน ดังนั้นการเปลี่ยนแปลงที่กระทำกับฐานข้อมูลระหว่างการทำงานของทรานแซคชันใด ๆ ก็ตาม จะถูกยกเลิกทั้งหมดและฐานข้อมูลจะย้อนกลับไปสู่สถานะก่อนการเริ่มประมวลผลทรานแซคชัน

สถานะของข้อมูลหลังจาก Rollback มี 2 สถานะคือ*การเปลี่ยนแปลงข้อมูลสามารถยกเลิกได้* และ*ข้อมูลก่อนหน้าสามารถเรียกคืนกลับมาได้*

ทรานแซคชันจะประกอบไปด้วยคำสั่งกี่คำสั่งก็ได้ และโดยทั่วไปทรานแซคชัน*จะเกิดขึ้นเมื่อ* มีการเริ่มใช้คำสั่ง SQL ใด ๆ และ*ทรานแซคชันจะจบ* เมื่อพบคำสั่ง Commit หรือ Rollback หรือ พบคำสั่ง DDL หรือ ระบบเกิดความล้มเหลว

1.5 ประเภททรานแซคชัน (Transaction Types)

1.5.1 เอ็กพลิซิท ทรานแซคชัน (Explicit Transactions) สามารถทำได้โดยการใช้คำสั่งจาก Transact-SQL โดยอาศัยคำสำคัญ (key words) เช่น :

- BEGIN TRANSACTION
- ROLLBACK TRANSACTION
- COMMIT TRANSACTION

- ROLLBACK WORK
- COMMIT WORK
- SAVE TRANSACTION
- COMMIT

1.5.2 อิมพลิซิท ทรานแซคชัน (Implicit Transaction) เป็นทรานแซคชันที่ทำได้โดยกำหนดโหมดของการทำงานด้วยคำสั่ง SET IMPLICIT_TRANSACTIONS ON หลังจากนั้นระบบจะทำการสร้างทรานแซคชันโดยอัตโนมัติถ้าพบคำสั่งต่อไปนี้

ALTER TABLE	**FETCH**	**REVOKE**	**CREATE**
SELECT	**DELETE**	**INSERT**	**TRUNCATE TABLE**
DROP	**OPEN**	**UPDATE**	**GRANT**

เราสามารถทำทรานแซคชันใด ๆ ได้โดยไม่ต้องเริ่มต้นด้วยคำสั่ง BEGIN TRAN แต่ยังคงต้องใช้คำสั่ง COMMIT TRAN และ ROLLBACK TRAN

1.5.3 ออโตเมติค ทรานแซคชัน (Automatic Transaction) หมายความว่า แต่ละคำสั่งจะมีทรานแซคชันเป็นของตัวเองไม่ได้ขึ้นกับทรานแซคชันอื่นแต่อย่างใด

2. ภาวะความพร้อมกัน (Concurrency)

จุดมุ่งหมายหลักประการหนึ่งของฐานข้อมูลคือ การที่ผู้ใช้จากหน่วยงานย่อยสามารถจะเรียกใช้ข้อมูลในฐานข้อมูลได้พร้อม ๆ กัน โดยเฉพาะฐานข้อมูลที่ใช้งานในระดับองค์กรซึ่งเป็นฐานข้อมูลในระบบที่ซับซ้อน เมื่อจะต้องมีการปรับค่าข้อมูลใด ๆ ให้ถูกต้อง และดำเนิน การให้เสร็จสมบูรณ์ก่อนที่ผู้ใช้คนอื่นจะเรียกไปใช้ ถ้าเกิดภาวะเรียกใช้ข้อมูลพร้อมกัน จะเกิดปัญหาข้อมูลไม่ได้ถูกแก้ไขให้ถูกต้อง ทำให้ได้ข้อมูลที่มีค่าไม่ถูกต้อง เป็นปัญหาสำคัญอย่างหนึ่งในการใช้ระบบฐานข้อมูล ระบบการจัดการฐานข้อมูลจะต้องมีกลไกในการจัดการเพื่อให้ผู้ใช้จากหน่วยงานย่อย ต่างก็สามารถเรียกใช้ข้อมูลในฐานข้อมูลเพื่อทำงานได้พร้อมกัน โดยที่ผลลัพธ์ที่ได้จากการทำงานจะต้องมีค่าถูกต้องเสมอ

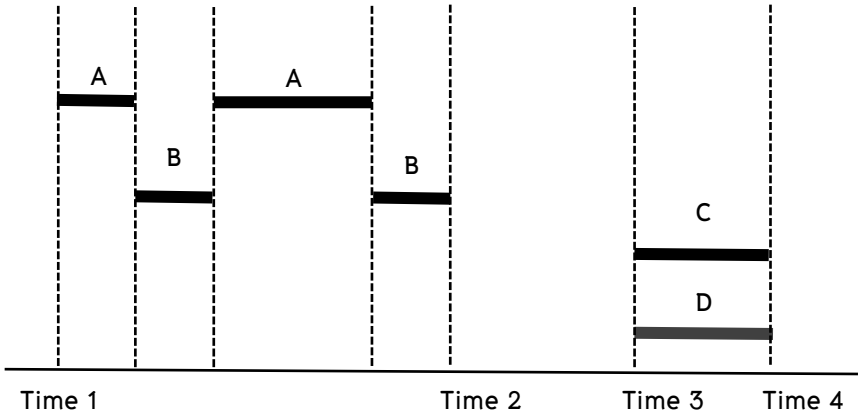
2.1 ความหมายภาวะความพร้อมกัน

“ภาวะความพร้อมกัน” หมายถึงการที่ทรานแซคชันหลาย ๆ ทรานแซคชันต้องการเรียก*ใช้ข้อมูลเดียวกัน* และ*ในเวลาเดียวกัน*จากฐานข้อมูล อาจจะกล่าวให้เข้าใจได้ง่าย ๆ ก็คือ “ภาวะความพร้อมกัน” ก็คือกระบวนการสำหรับการควบคุมข้อมูลที่ถูกเรียกใช้โดยแต่ละงานให้มีความถูกต้องอยู่เสมอนั่นเอง

การทำงานในระบบหลายโปรแกรม เป็นการทำงานของระบบคอมพิวเตอร์ที่ออกแบบเพื่อให้ CPU หรือหน่วยประมวลผลกลาง ทำงานได้หลายๆ งานในขณะเดียวกัน เพื่อให้การใช้งาน CPU เป็นไปอย่างคุ้มค่าโดยไม่ต้องอยู่ว่าง เนื่องจาก CPU จะมีประสิทธิภาพและความเร็วในการทำงานสูง ถ้าไม่มีระบบการทำงานแบบระบบหลายโปรแกรม CPU จะต้องทำงานใดงานหนึ่งให้แล้วเสร็จก่อน จึงจะสามารถทำงานต่อ ๆ ไปได้ ดังนั้นจึงเกิดแนวคิดการประมวลผลแบบหลายโปรแกรม โดยมีการสลับช่วงการทำงานระหว่างโปรแกรม เพื่อสลับให้ CPU ไปทำงานของทรานแซคชันอื่น โดยใช้หลักการที่ว่า “ระบบจัดการฐานข้อมูลจะควบคุมภาวะความพร้อมกันเพื่อให้ทรานแซคชันมีการสลับการทำงานกันในขณะใดขณะหนึ่ง และผลลัพธ์ที่ได้จะต้องมีความถูกต้องเสมือนว่า แต่ละทรานแซคชันทำงานเรียงกันลำดับกันทีละทรานแซคชันจนสิ้นสุดงานของแต่ละทรานแซคชันนั้น” หลักการนี้เรียกว่า *อินเทอลีฟ* (interleaved)

การทำงานในระบบคอมพิวเตอร์ที่มี CPU หลายตัว เพื่อรองรับการทำงานของทรานแซคชันใดทรานแซคชันหนึ่ง โดยไม่ต้องสลับการทำงานระหว่างทรานแซคชัน

หลาย ๆ ทรานแซคชัน CPU แต่ละตัวก็จะทำงานของแต่ละทรานแซคชันแยกกันไปจนกว่างานนั้นจะแล้วเสร็จ



ภาพที่ 9.4 (ซ้าย) การประมวลผลในระบบหลายโปรแกรม (Multiprogramming)
(ขวา) การประมวลผลในเวลาเดียวกัน (Simultaneous Processing)

การประมวลผลระบบหลายโปรแกรม ทรานแซคชัน A และ B ทำงานพร้อมกันในช่วงเวลา Time 1 และ 2 ระบบมีการควบคุมภาวะความพร้อมกัน โดยมีการสลับการทำงานระหว่าง ทรานแซคชัน A และ B ส่วนการทำงานในระบบที่มี CPU หลายตัว เป็นการประมวล ผลในเวลาเดียวกันของทรานแซคชัน C และ D ในช่วงเวลา Time 3 และ 4 แต่ละทรานแซคชันจะมี CPU ประมวลผลแยกกัน

2.2 ปัญหาที่เกิดจากภาวะความพร้อมกัน (Concurrency Problems)

การควบคุมภาวะความพร้อมกันในการใช้งานฐานข้อมูลเป็นสิ่งสำคัญอย่างยิ่ง เพราะหากระบบจัดการฐานข้อมูลไม่มีกลไกดังกล่าวย่อมจะก่อให้เกิดปัญหาในการทำงาน ปัญหาที่เกิดจากสภาวะการเข้าถึงพร้อมกัน หรือจากการที่ทรานแซคชันหลายทรานแซคชันถูกประมวลผลพร้อมกันโดยไม่ได้มีการควบคุม มีดังนี้

2.2.1 ปัญหาการสูญหายข้อมูลที่เกิดจากการปรับปรุงข้อมูล (The Lost Update Problem) เป็นปัญหาที่เกิดจากทรานแซคชันมากกว่าหนึ่งทรานแซคชันเรียกใช้ข้อมูลชุดเดียวกัน และแต่ละทรานแซคชันต่างก็ต้องการปรับปรุงข้อมูลในช่วงเวลาไกล้ ๆ กัน โดยไม่ มีการป้องกันการเปลี่ยนแปลงจากรายการอื่น ทำให้ได้ผลลัพธ์ไม่ถูกต้อง เพราะข้อมูลที่ถูกแก้ไขโดยทรานแซคชันก่อนหน้านั้นจะหายไปหมด จะปรากฏเฉพาะผลลัพธ์ที่เกิดจากการปรับปรุงแก้ไขของทรานแซคชันหลังสุดเท่านั้น ดังตารางที่ 9.1

ตารางที่ 9.1 แสดงปัญหาการสูญหายข้อมูลที่เกิดจากการปรับปรุงข้อมูล

Transaction	Time				
	Time 1	Time 2	Time 3	Time 4	Time 5
T1	Read A, 100 A : = A –50		Write A, 50 Read B ,300		B : = B+50 Write B,350
T2		Read A, 100 A : = A + 50		Write A,200	

2.2.2 ปัญหาการเรียกใช้ข้อมูลที่ยังไม่ได้รับการยืนยันความสมบูรณ์ (The Uncommitted Dependency Problem) สาเหตุเกิดจากทรานแซคชันตั้งแต่สองทรานแซคชันขึ้นไป เรียกใช้ข้อมูลชุดเดียวกันโดยที่ทรานแซคชันแรกยังอยู่ในระหว่างการประมวลผลแต่ยังไม่มีการยืนยันความเสร็จสมบูรณ์ของข้อมูล ในขณะเดียวกันทรานแซคชันที่สองก็เรียกใช้ข้อมูลที่แก้ไขโดยทรานแซคชันแรก หลังจากนั้นปรากฏว่าทรานแซคชันแรกมีปัญหาจะต้องถูกยกเลิก และกลับมาเริ่มต้นประมวลผลใหม่ทั้งหมด (Rollback) ทำให้ข้อมูลที่ทรานแซคชันที่สองเรียกใช้กลับไปมีค่าเดิม ทรานแซคชันอื่นที่เรียกใช้ข้อมูลเดียวกันก็จะได้รับข้อมูลที่ไม่ถูกต้อง

ตารางที่ 9.2 แสดงปัญหาการเรียกใช้ข้อมูลที่ยังไม่ได้รับการยืนยันความสมบูรณ์

Transaction	Time		
	Time 1	Time 2	Time 3
T1	Read A, 100 A : = A –50 Write A, 50		Read B, 300 Rollback (A = 100)
T2		Read A, 50 A : = A + 100 Write A, 150	(ควรจะได้ค่า A เป็น 200)

(Uncommitted Dependency)

2.2.3 ปัญหาจากการวิเคราะห์ข้อมูลที่ขัดแย้งกัน (The Inconsistent Analysis Problem) หรือความไม่สอดคล้องกันของข้อมูล เกิดจากกรณีที่ทรานแซคชันตั้งแต่สองทรานแซคชันขึ้นไปเรียกใช้ข้อมูลชุดเดียวกัน แต่มีอีกทรานแซคชันหนึ่งใช้ข้อมูลนั้นเพื่อประมวลผล ในขณะที่มีอีกทรานแซคชันหนึ่งได้มีการแก้ไขข้อมูลชุดเดียวกัน ทำให้ผลลัพธ์ของทรานแซคชันแรกไม่ถูกต้องเพราะได้รับข้อมูลที่อยู่ในสถานะไม่ถูกต้องไปใช้งาน ส่งผลให้การประมวลผลของทรานแซคชันนั้นได้ผลลัพธ์ที่ผิดพลาด

ตารางที่ 9.3 แสดงปัญหาจากการวิเคราะห์ข้อมูลที่ขัดแย้งกัน

Time / Transaction	Time 1	Time 2	Time 3	Time 4
T1	Sum = 0 Read A, 100			Read B,400 Sum=sum+A+B
T2	–	Read A, 100 A : = A – 50 Write A, 50	Read B, 300 B=B+100 Write B,400	

(Uncommitted Dependency)

3. ภาวะติดตาย (Dead Locks)

ระบบคอมพิวเตอร์ทั่ว ๆ ไป จะประกอบไปด้วยทรัพยากรต่าง ๆ เช่น พื้นที่ว่างในหน่วยความจำหลัก (Memory Space) เวลาในการประมวลผล (CPUTime) แฟ้มข้อมูล (File) และอุปกรณ์รับส่งข้อมูล (I/O Device) เชื่อมโยงกันอยู่มากมาย ทรัพยากรเหล่านี้จะมีจำนวนจำกัด แต่ละชนิดอาจจะมีจำนวนมากกว่า 1 ตัวก็ได้ ซึ่งจะต้องแบ่งปันให้ทรานแซคชันต่าง ๆ ในระบบใช้ทำงาน

การทำงานในระบบโปรแกรมเดียว หรือทรานแซคชันเดียวจะไม่มีปัญหาเกิดขึ้น เนื่องจากมีเพียงทรานแซคชันเดียวเท่านั้นครอบครองทรัพยากรไม่มีทรานแซคชันอื่นมาใช้งานร่วมด้วย แต่ระบบคอมพิวเตอร์ที่มีการทำงานแบบหลายทรานแซคชัน อาจจะมีการแข่งกันเข้าใช้ทรัพยากรซึ่งมีอยู่อย่างจำกัดในการใช้งานในระบบ เมื่อ ทรานแซคชันหนึ่งต้องการเข้าใช้ทรัพยากรดังกล่าวจะต้องร้องขอเข้าใช้งานทรัพยากรนั้น ถ้าใน

ขณะนั้นทรัพยากรนั้นว่าง ทรานแซคชันสามารถเข้าใช้งานได้เลย และจะต้องออกจากการใช้ทรัพยากรนั้นหลังจากเสร็จสิ้นการทำงาน แต่ถ้าในขณะนั้นทรัพยากรนั้นไม่ว่าง ทรานแซคชันจะต้องรอจนกว่าทรัพยากรจะว่างจึงจะเข้าทำงานได้

ดังนั้นจึงอาจเจอเหตุการณ์ที่ทรานแซคชันถูกให้อยู่ในสถานะการรออย่างไม่มีที่สิ้นสุด เนื่องจากทรัพยากรนั้นก็ถูกร้องขอจากทรานแซคชันอื่นที่รออยู่เช่นกัน ทำให้ไม่มีทรานแซคชันใดสามารถทำงานต่อได้ เพราะว่าแต่ละทรานแซคชันต่างไม่ยอมปลดปล่อยทรัพยากรของตน จนกว่าอีกทรานแซคชันหนึ่งจะปลดปล่อยทรัพยากรที่ตนต้องการให้ก่อน จึงเกิด*ภาวะติดตาย*ขึ้นในระบบ กลุ่มของทรานแซคชันจะตกอยู่ในภาวะติดตายเมื่อทุก ๆ ทรานแซคชันที่อยู่ในกลุ่มนั้นต่างก็กำลังรอคอยที่จะใช้ทรัพยากรที่กำลังถูกใช้โดยทรานแซคชันอื่นในกลุ่มนั้น

ภาวะติดตาย ส่วนมากจะเกี่ยวข้องกับความขัดแย้งในการใช้ทรัพยากรร่วมกันของระบบโดยทรานแซคชันสองทรานแซคชันหรือมากกว่า อาจนิยาม *ภาวะติดตาย* ได้ดังนี้ “เกิดจากการที่ทรานแซคชันตั้งแต่สองทรานแซคชัน หรือมากกว่าทำการปิดกั้นข้อมูลที่ใช้งาน โดยแต่ละทรานแซคชันต่างรอใช้งานข้อมูลของทรานแซคชันอื่นที่ใช้งานอยู่อย่างไม่รู้จบในลักษณะเป็นลูกโซ่ จึงทำให้ไม่สามารถทำการประมวลผลต่อได้ เรียกว่าเกิด *ภาวะติดตาย* ” เมื่อเกิดภาวะติดตายขึ้นในระบบ จะทำให้ไม่มีทรานแซคชันใดที่สามารถทำงานจนเสร็จสมบูรณ์ได้ และทรัพยากรจะถูกครอบครองจนหมด

ภาพที่ 9.5 ตัวอย่างภาวะติดตายของการจราจร

(ที่มา : https://sunisa1.wordpress.com/page/5/)

การทำงานของทรานแซคชันในการใช้ทรัพยากรของระบบ ทรานแซคชันจะต้องร้องขอใช้ทรัพยากรก่อนที่จะได้ใช้ทรัพยากรนั้น และจะต้องคืนทรัพยากรนั้นกลับสู่ระบบเมื่อใช้เสร็จ

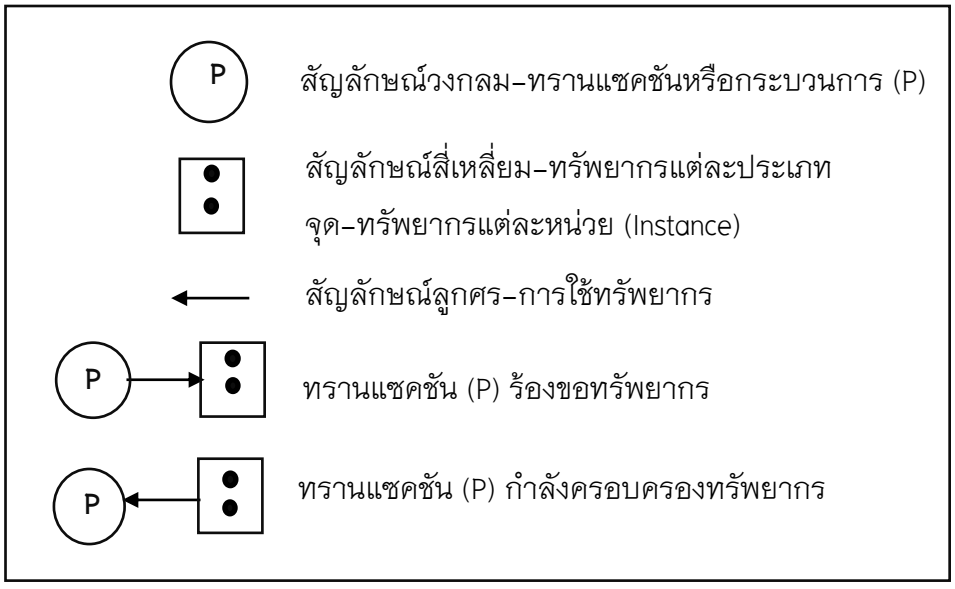
3.1 กราฟการจัดสรรทรัพยากร (Resource-Allocation Graph)

การทำงานของทรานแซคชันในการใช้ทรัพยากร จะเขียนเป็นกราฟ และใน การเขียนกราฟจะใช้*วงกลม*แทนกระบวนการ หรือ ทรานแซคชัน *ใช้สี่เหลี่ยม* แทนทรัพยากร และใช้*จุด* แทนจำนวนทรัพยากรแต่ละตัว ซึ่งจะอยู่ภายในสี่เหลี่ยม โดยที่*เส้นร้องขอ*จะต้องชี้ไปยังกรอบสี่เหลี่ยม (ทรัพยากร) เท่านั้น แต่*เส้นถือครอง* จะต้องชี้ไปยังจุดใดจุดหนึ่งในสี่เหลี่ยม และมีลำดับขั้นตอนดังนี้

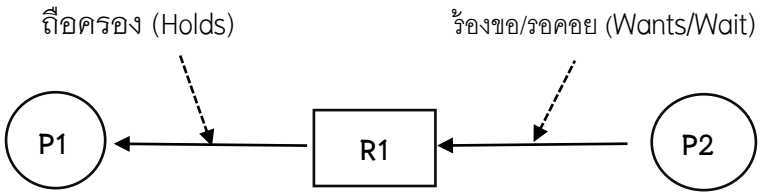
ลำดับแรก คือ*การร้องขอ* (Request) ในกรณีการร้องขอไม่ได้รับการอนุมัติจากระบบในทันที อาจจะเนื่องจากทรัพยากรที่ต้องการนั้นกำลังถูกทรานแซคชันอื่นใช้งาน ดังนั้น กระบวนการที่ร้องขอจะต้องรอจนกว่าจะได้รับทรัพยากรที่ต้องการ

ลำดับที่สอง คือ การ*ใช้งาน* (Use) เป็นลำดับที่ทรานแซคชันสามารถใช้งานทรัพยากรที่ได้รับการจัดสรรจากระบบ เช่นถ้าทรัพยากรที่ร้องขอคือเครื่องพิมพ์ เมื่อได้รับการจัดสรรจากระบบแล้ว ทรานแซคชันนั้นสามารถพิมพ์ข้อมูลออกทางเครื่องพิมพ์ได้

ลำดับที่สาม คือ*การคืน* (Release) เป็นลำดับที่ทรานแซคชัน ต้องออกจากการใช้ทรัพยากร หรือต้องคืนทรัพยากรที่ใช้เสร็จแล้วกลับสู่ระบบนั่นเอง

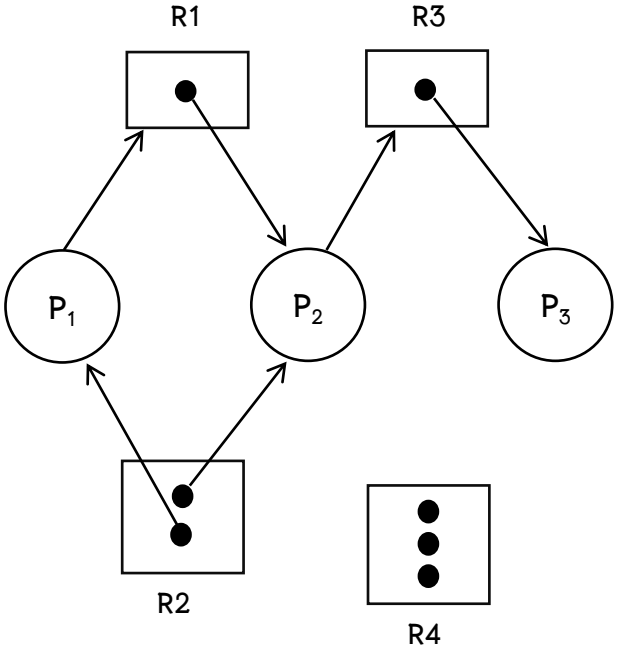


ภาพที่ 9.6 สัญลักษณ์และการทำงานทรานแซคชันในการใช้ทรัพยากรของระบบ



ภาพที่ 9.7 การทำงานทรานแซคชันในการใช้ทรัพยากรของระบบ

- ทรานแซคชัน P1 กำลังถือครองหรือใช้งานทรัพยากร R1
- ทรานแซคชัน P2 รอคอย หรือรอใช้ทรัพยากร R1



ภาพที่ 9.8 กราฟการจัดสรรทรัพยากรในนระบบ

อธิบาย

เซต P = { P1 , P2 , P3 } หมายถึง *ทรานแซคชัน*

เซต R = { R1 , R2 , R3 , R4 } หมายถึง*ทรัพยากร*

เซต E = { P1→ R1, P2→ R3 , R1→ P2 , R2→ P2 ,R2→ P1 , R3→ P3 }

ทรัพยากรในระบบ

ทรัพยากร R1 มี 1 ตัว ทรัพยากร R2 มี 2 ตัว

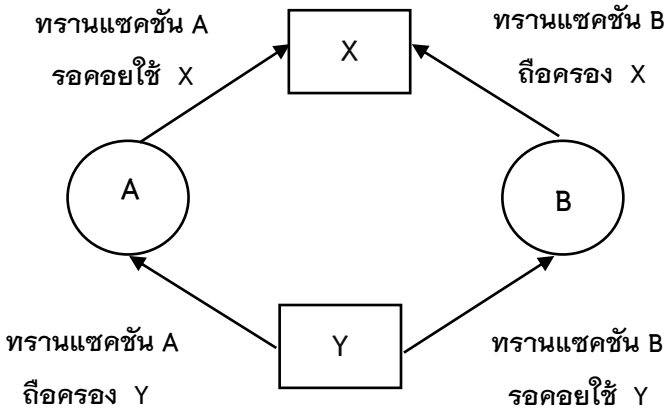
ทรัพยากร R3 มี 1 ตัว ทรัพยากร R4 มี 3 ตัว

สถานะของทรานแซคชัน

- ทรานแซคชัน P1 กำลัง*ถือครอง*ทรัพยากร R2 และกำลัง*รอคอย*ที่จะใช้ทรัพยากร R1
- ทรานแซคชัน P2 กำลัง*ถือครอง*ทรัพยากร R1 และ R2 และกำลังรอคอย ที่จะใช้ทรัพยากร R3
- ทรานแซคชัน P3 กำลัง*ถือครอง*ทรัพยากร R3

จากกราฟจะเห็นว่าถ้าไม่มีวงจรกราฟ หรือ ไม่มีวงกลม ในกราฟ ระบบจะไม่อยู่ในสถานะภาวะติดตายแต่ถ้ามีวงจรกราฟ หรือ มีวงกลม อาจจะเกิดภาวะติดตายขึ้นก็ได้ โดยที่

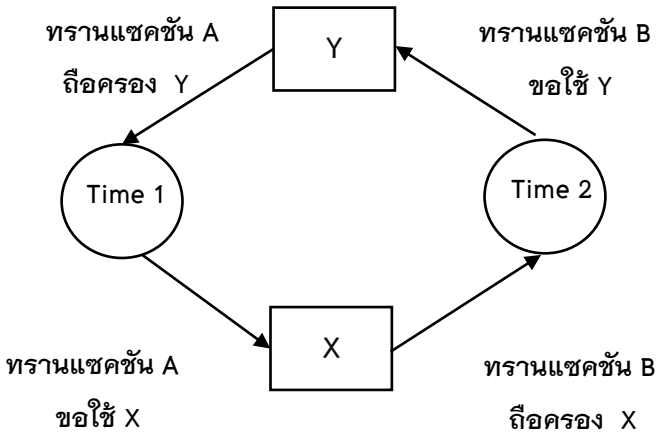
- ถ้าเป็นระบบที่ทรัพยากรแต่ละชนิดมีเพียง 1 ตัว เมื่อมีวงจรในกราฟก็จะเกิดภาวะติดตายในระบบอย่างแน่นอน
- ถ้าเป็นระบบที่ทรัพยากรแต่ละชนิดมีจำนวนมากกว่า 1 ตัว วงจรที่เกิดขึ้นในกราฟ ไม่สามารถบอกได้ว่า จะเกิดภาวะติดตายในระบบหรือไม่ (อาจจะเกิดหรือไม่เกิดภาวะติดตายในระบบก็ได้)



ภาพที่ 9.9 แสดงการจัดสรรทรัพยากรที่ยังไม่เกิดภาวะติดตาย

จากภาพที่ 9.9 ทรานแซคชัน **A** กำลัง*ถือครอง*ทรัพยากร **Y** และ*รอคอย*การใช้งานทรัพยากร **X** แต่ทรัพยากร **X** กำลังถูก*ถือครอง*โดยทรานแซคชัน **B** ในขณะเดียวกัน ทรานแซคชัน **B** ก็ต้องการใช้งานทรัพยากร **Y** ซึ่งกำลังถูก*ถือครอง*โดยทรานแซคชัน A การทำงานในขณะนี้ยังไม่เกิดภาวะติดตาย เพราะว่าแต่ละทรานแซคชันต่างก็ถือครองทรัพยากร และมีการรอการใช้งานทรัพยากรอยู่

จากภาพที่ 9.10 ภาวะติดตาย เกิดขึ้นจาก ณ เวลา Time 1 ทรานแซคชัน A กำลังใช้ทรัพยากร Y และในช่วงเวลาเดียวกันทรานแซคชัน B ก็กำลังใช้ทรัพยากร X อยู่ในช่วงเวลา Time 2 ทรานแซคชัน B ต้องการใช้ทรัพยากร Y ซึ่งทรานแซคชัน A ยังคงใช้อยู่ เช่นเดียว กับทรานแซคชัน B ก็ยังคงใช้ทรัพยากร X ที่ทรานแซคชัน A ต้องการใช้อยู่ ลักษณะนี้จะทำให้เกิดภาวะติดตายขึ้น



ภาพที่ 9.10 แสดงการจัดสรรทรัพยากรที่เกิดภาวะติดตาย

3.2 เงื่อนไขที่ทำให้เกิดภาวะติดตาย ที่สำคัญ คือ

3.2.1 เงื่อนไขที่ 1 *การไม่เกิดร่วม* (Mutual Exclusion Condition) หมายถึงกรณีที่ทรัพยากรอย่างน้อย 1 ตัวที่ไม่ได้รับอนุญาตให้ใช้ร่วมกันได้ คือในเวลาหนึ่งจะมีเพียง ทรานแซคชันเดียวเท่านั้นที่ใช้งานทรัพยากรได้ ทรานแซคชันอื่นไม่สามารถใช้งานได้จะต้องรอจนกว่าทรานแซคชันดังกล่าวได้ใช้งานเสร็จและปล่อยทรัพยากรนั้นว่าง นั่นก็คือ *ห้ามใช้ทรัพยากรร่วมกัน*นั่นเอง

3.2.2 เงื่อนไขที่ 2 *การถือครองแล้วรอคอย* (Hold and Wait Condition) หมายถึง การที่ทรานแซคชันหนึ่งใช้ทรัพยากรอยู่ แต่ต้องการใช้ทรัพยากรอื่นเพิ่มเติม โดยที่กำลังรอคอยทรัพยากรอีกตัวหนึ่ง ซึ่งทรัพยากรตัวนั้นก็มีสถานะไม่ว่าง เพราะถูกครอบครองโดยอีกทรานแซคชันหนึ่ง ทำให้ทรานแซคชันนั้นต้องรอ

3.2.3 เงื่อนไขที่ 3 *ไม่มีการแทรกกลางคัน* (No Preemption Condition) หมายถึงทรานแซคชันที่รอคอยใช้ทรัพยากรต่อจากทรานแซคชันที่กำลังใช้ทรัพยากรนั้น ๆ อยู่ จะต้องรอคอยจนกว่าทรานแซคชันนั้น ๆ ทำงานเสร็จและปลดปล่อยทรัพยากร ทรานแซคชันไม่สามารถแย่งชิงทรัพยากรจากทรานแซคชันอื่นได้ ระบบไม่สามารถบังคับให้ทรานแซคชันปลดปล่อยทรัพยากรให้เป็นอิสระได้ ทรัพยากรจะเป็นอิสระได้ก็ต่อเมื่อ ทรานแซคชันยกเลิกการถือครองหรือต้องการคืนเท่านั้น ซึ่งก็หมายถึง*ห้ามแซงนั่นเอง*

3.2.4 เงื่อนไขที่ 4 *การรอเป็นวงกลม* (Circular Wait Condition) เป็นการรอคอยแบบงูกินหางหรือแบบลูกโซ่ หมายถึง ถ้าเกิดวงจรรอคอยของทรานแซคชันสองทรานแซคชัน หรือ มากกว่า ที่ต่างก็รอคอยทรัพยากรที่ถือครองโดยทรานแซคชันที่อยู่ในวงจรลูกโซ่นั้น ก็จะเกิดภาวะติดตายขึ้นในระบบ

จะเห็นได้ว่าเงื่อนไขที่ 1 ที่ 2 และที่ 3 ที่กล่าวมานั้นเป็นสิ่งที่ระบบปฏิบัติการทั่ว ๆ ไปต้องการและใช้อยู่เพื่อความถูกต้อง ตรงกันในการใช้งานของฐานข้อมูล อย่างไรก็ตามเงื่อนไขทั้งสามเงื่อนไขดังกล่าวอาจจะทำให้เกิดหรือไม่ทำให้เกิดภาวะติดตายก็ได้ ภาวะติดตายจะเกิด ขึ้นจริงก็ต่อเมื่อมี *เงื่อนไขที่ 4 Circuit wait condition* หรือ “การรอเป็นวงกลม” เกิดขึ้น

3.3 การแก้ไขภาวะติดตาย

การแก้ไขภาวะติดตายมี 3 วิธี คือ *การป้องกันไม่ให้เกิดภาวะติดตาย* (Deadlock Prevention) *การตรวจจับหรือตรวจสอบการเกิดภาวะติดตาย* (Deadlock Detection) และ*การหลีกเลี่ยงภาวะติดตาย* (Deadlock Avoidance) ดังนี้

3.3.1 การป้องกันไม่ให้เกิดภาวะติดตาย เป็นการป้องกันไม่ให้เกิดภาวะติดตาย โดยที่ระบบการจัดการฐานข้อมูล จะต้องป้องกันไม่ให้เกิดเงื่อนไขที่ทำให้เกิดภาวะติดตายซึ่งมี 4 เงื่อนไขด้วยกัน คือ *การไม่เกิดร่วม การถือครองแล้วรอคอย ไม่มีการแทรกแซงกลางคัน* และ *การรอเป็นวงกลม* ดังที่กล่าวมาแล้ว หรือระบบจะต้องกำหนดเอาไว้ว่าทรานแซคชันจะต้องปิดกั้นทุกข้อมูลที่ต้องการเรียกใช้ไว้

ล่วงหน้าก่อนที่จะมีการใช้งาน ถ้าหาก ทรานแซคชันไม่สามารถจะปิดกั้นข้อมูลไว้ล่วงหน้าได้ จะต้องรอจนกว่าจะปิดกั้นได้ครบก่อน จึงจะเริ่มต้นทำงานได้

3.3.2 การตรวจสอบการเกิดภาวะติดตาย เป็นวิธีแก้ปัญหาเมื่อเกิดภาวะติดตายแล้ว ระบบจะทำการตรวจสอบว่าทรานแซคชันใดทำให้เกิดภาวะติดตาย โดยตรวจหาการรอแบบ “เป็นวงกลม” ระหว่างทรานแซคชันต่าง ๆ โดยเครื่องมือที่สำคัญคือ Wait–For Graph หรือ “ กราฟการจัดสรรทรัพยากรของระบบ ” เป็นกราฟที่มี ทิศทางเดียว (Directed Graph) ที่แสดงถึงความสัมพันธ์ระหว่างทรานแซคชันกับทรัพยากรในประเด็นการร้องขอและการครองทรัพยากร ถ้าพบว่าทรานแซคชันใดมีสถานะการรอแบบ“เป็นวงกลม” ระบบจะยกเลิกการทำงานของทรานแซคชันนั้น หรือปิดระบบทั้งหมดแล้วเริ่มต้นทำงานใหม่

3.3.3 การหลีกเลี่ยงปัญหาการเกิดภาวะติดตาย วิธีหลีกเลี่ยงภาวะติดตายนี้ ระบบจะทำการตรวจสอบสถานะการจัดสรรทรัพยากรอยู่ตลอดเวลาเพื่อที่จะทำให้เชื่อมั่นได้ว่าการจัดสรรทรัพยากรนั้นจะไม่นำไปสู่เงื่อนไข “การรอเป็นวงกลม” ซึ่งจะทำให้เกิดปัญหาภาวะติดตาย อาจจะสรุปได้ว่าการเกิดภาวะติดตายจะทำให้ระบบปฏิบัติการเสียเสถียรภาพไป ดังนั้นควรจะมีการป้องกันไว้ก่อน หรือระบบควรที่จะมีการตรวจสอบอยู่อย่างสม่ำเสมอ

4. เทคนิคการควบคุมภาวะความพร้อมกัน (Concurrency Control Technique)

สาเหตุหนึ่งที่ทำให้ระบบฐานข้อมูลเกิดปัญหาความไม่สอดคล้องกันของข้อมูล คือ การเกิดความขัดแย้ง (Conflict) ของทรานแซคชันต่างกันที่กำลังเรียกใช้ข้อมูลตัวเดียวกัน หรือการแย่งชิงข้อมูลที่ยังประมวลผลไม่เสร็จนำไปใช้งานโดยทรานแซคชันอื่น เพื่อแก้ปัญหาการเกิดความขัดแย้งดังกล่าว ระบบฐานข้อมูลจะต้องมีกระบวนการที่ทำการควบคุมหรือจัดการไม่ให้ ทรานแซคชันหลาย ๆ ทรานแซคชันเรียกใช้ข้อมูลชุดเดียวกันพร้อม ๆ กัน การควบคุมดังกล่าวเรียกว่า การควบคุมภาวะความพร้อมกัน

เมื่อเกิดภาวะความพร้อมกันในระบบ ระบบการจัดการฐานข้อมูลจะต้องมีเทคนิคหรือวิธีในการจัดเรียงลำดับการทำงานของแต่ละทรานแซคชัน เทคนิคหรือวิธีในการควบคุมภาวะพร้อมกันจะมีอยู่หลายเทคนิคด้วยกัน เช่น

4.1 การควบคุมภาวะพร้อมกันโดยใช้วิธีการปิดกั้น (Locking Technique)

ระบบฐานข้อมูลมีวิธีควบคุมแบบนี้เพื่อใช้ในการควบคุมสภาวะการเข้าถึงข้อมูลพร้อมกัน วิธีนี้เป็นการกำหนดสถานการณ์ปิดกั้นให้กับข้อมูล เพื่อบอกถึงสิทธิ์

ในการใช้งานข้อมูล และเพื่อป้องกันไม่ให้ทรานแซคชันอื่นเข้ามาใช้งานข้อมูลนั้นในฐานข้อมูล จนกว่า ทรานแซคชันที่เรียกใช้ข้อมูลนั้น จะทำงานเสร็จสิ้นสมบูรณ์ หรือยกเลิกการแก้ไขข้อมูล จึงจะยกเลิกการปิดกั้นข้อมูลนั้น ทรานแซคชันอื่นจึงจะเข้ามานำข้อมูลชุดนั้นไปใช้ทำงานได้พื้นฐานของเทคนิคการปิดกั้น มี 2 แบบ คือ "Read Lock" หมายถึงถ้าทรานแซคชันใดมีการ "Read Lock" ข้อมูล จะอ่านข้อมูลได้ แต่จะไม่สามารถปรับปรุงข้อมูล และ "Write lock" หมายถึงถ้า ทรานแซคชันใดมีการ "Write lock" ข้อมูล จะสามารถทำได้ทั้งการอ่านและการปรับปรุงข้อมูล

4.1.1 การปิดกั้นข้อมูลแบ่งเป็น 2 ประเภท คือ *แชร์ล็อค* หรือ *การปิดกั้นแบบแบ่งใช้ร่วม* (Shared Locks / Read Lock หรือ S Lock) และอีกประเภทคือ *เอ็กคลูซีฟล็อค* หรือ การปิดกั้นแบบผูกขาด (Exclusive Locks / Write Lock / X Lock)

1) แชร์ล็อค เป็นการป้องกันข้อมูล โดยกำหนดสถานะของข้อมูลที่อนุญาตให้ทรานแซคชันอื่นสามารถเข้ามาอ่านข้อมูลต่าง ๆ ร่วมกันได้ การอ่านข้อมูลไม่มีผลต่อการเปลี่ยนแปลงข้อมูล เราสามารถทำการปิดกั้นแบบนี้เพื่อใช้งานร่วมกันระหว่างหลาย ๆ ทรานแซคชันได้

2) เอ็กคลูซีฟล็อค เป็นการปิดกั้นแบบผูกขาดจะไม่อนุญาตให้ทรานแซคชัน อื่นเข้ามาใช้ข้อมูลที่ถูกปิดกั้นอยู่ ไม่ว่าจะเป็นการอ่านหรือการเขียน เพื่อเป็น การป้องกันการแทรกการทำงานจากทรานแซคชันอื่น ระบบจึงอนุญาตให้เพียงทรานแซคชันเดียวเท่านั้นที่สามารถปิดกั้นข้อมูลที่ต้องการใช้งานแบบนี้ได้ ดังนั้นข้อมูลจะใช้งานได้เฉพาะผู้ที่ใช้วิธีปิดกั้นแบบนี้เท่านั้น ในขณะนั้นผู้ใช้คนอื่นที่ต้องการใช้งานข้อมูลนั้นจะไม่สามารถอ่าน และแก้ไขข้อมูลได้จนกว่าทรานแซคชันนั้นจะทำการปลดปล่อยการปิดกั้นนั้นก่อน

4.1.2 ระดับการปิดกั้นข้อมูล (Locking Level) การควบคุมสภาวะการเข้าถึงข้อมูลพร้อมกันโดยการใช้การปิดกั้น สามารถที่จะเลือกระดับการปิดกั้นข้อมูลได้หลายระดับตามลักษณะของงาน เช่นปิดกั้นฐานข้อมูลทั้งฐาน (Database Locking) ปิดกั้นเฉพาะบางตาราง (Table Locking) ปิดกั้นเฉพาะบางแถว (Record Locking) หรือปิดกั้นเฉพาะบางฟิลด์ (Field Locking) เป็นต้น

1) การปิดกั้นฐานข้อมูลทั้งฐาน เป็นการปิดกั้นทั้งฐานข้อมูล ขณะที่ ทรานแซคชันใดทรานแซคชันหนึ่งกำลังประมวลผล ฐานข้อมูลทั้งหมดจะถูกปิดกั้น ดังนั้นผู้ใช้คนอื่น ๆ จะไม่สามารถใช้ฐานข้อมูลนั้นได้เลย จนกว่าจะมีการปล่อย

การปิดกั้น การปิดกั้นแบบนี้เหมาะสมกับการปฏิบัติงานที่มีการประมวลผลแบบชุดคำสั่งหรือแบตช์ (Batch Processing)

2) การปิดกั้นเฉพาะบางตารางในฐานข้อมูล โดยจะทำการปิดกั้นตารางใดตารางหนึ่งที่ต้องการ ทำให้ผู้ใช้คนอื่น ๆไม่สามารถอ่านเรคอร์ดของตารางได้จนกว่าจะมีการปลดล็อกตาราง การปิดกั้นแบบตารางนี้จะเหมาะสมกับงานที่ต้องการปรับปรุงหรือแก้ไขข้อมูลทั้งหมดในตารางนั้น

3) การปิดกั้นเฉพาะบางแถว เป็นการปิดกั้นข้อมูลบางแถวหรือบางระเบียนในตาราง ทำให้ผู้ใช้คนอื่น ๆไม่สามารถเข้าดู แก้ไข หรือปรับปรุงข้อมูลแถวที่ถูกปิดกั้นนี้ได้จนกว่าจะมีการปล่อยปิดกั้น การปิดกั้นแบบนี้จะถูกใช้มากในการเขียนโปรแกรม ที่ใช้ในการปรับปรุงข้อมูลแถวใดแถวหนึ่งที่ต้องการในตาราง เป็นการห้ามไม่ให้ผู้อื่นเข้าถึงข้อมูลที่ถูกปิดกั้นนี้ได้ ไม่ว่าจะเป็นการดู หรือการแก้ไขข้อมูล

4) การปิดกั้นเฉพาะบางฟิลด์ เป็นการปิดกั้นบางคอลัมน์ของข้อมูล โดยเฉพาะคอลัมน์ที่มักจะถูกใช้งานหรือแก้ไขข้อมูลบ่อย ๆ โดยที่ผู้ใช้คนอื่นยังสามารถเข้าใช้งานข้อมูลต่างๆ ได้ยกเว้นคอลัมน์ที่ถูกปิดกั้นอยู่ การปิดกั้นบางฟิลด์นี้เหมาะสำหรับกับ การปิดกั้นคอลัมน์ที่มักจะมีการเปลี่ยนแปลงค่าหรือแก้ไขข้อมูลอยู่เสมอ แต่ส่วนใหญ่แล้วการปิดกั้นในระดับนี้จะใช้กันน้อยมาก

4.2 การควบคุมภาวะพร้อมกันโดยไม่ใช้วิธีการปิดกั้น (Without Locking)

การควบคุมภาวะพร้อมกันโดยเทคนิคการปิดกั้นทำให้เสียค่าใช้จ่ายในระบบค่อนข้าง มาก และอาจส่งผลทำให้เกิดภาวะการติดตายได้ ดังนั้นจึงมีการควบคุมภาวะพร้อมกันโดยใช้วิธี “ไม่ใช้วิธีการปิดกั้น” หรือ “ไม่ใช้เทคนิคการปิดกั้น” และยังสามารถแก้ไขปัญหาการเกิดภาวะติดตายได้ด้วย การควบคุมภาวะพร้อมกันโดยไม่ใช้วิธีการปิดกั้นมี 2 วิธีที่สำคัญคือ วิธีออฟติมิสติค (Optimistic Concurrence Control) และวิธีไทม์แสตมป์ (Timestamp–Based Concurrency Control)

4.2.1 วิธีควบคุมภาวะพร้อมกันแบบออฟติมิสติค การควบคุมภาวะพร้อมกันโดยวิธีนี้ อยู่บนแนวคิดที่อนุญาตให้ทรานแซคชันทำงานใช้ข้อมูลตามปกติ ไม่มีการตรวจสอบล่วงหน้า แต่ละทรานแซคชันจะมีพื้นที่เป็นของตนเอง (Private Workspace) ข้อมูลที่มีการแก้ไขโดยแต่ละทรานแซคชันจะไม่บันทึกไว้ในฐานข้อมูลโดยตรง แต่จะบันทึกไว้ในพื้นที่อื่น ที่กำหนดไว้ชั่วคราวจนกว่าทรานแซคชันนั้นจะทำงานเสร็จสมบูรณ์ จึงจะมีการตรวจสอบก่อนที่จะบันทึกลงในฐานข้อมูลจริง

การตรวจสอบจะพิจารณาจาก “ทรานแซคชันนั้นมีการทำงานที่ไม่เป็นไปตามลำดับ และทำให้เกิดการขัดกันกับทรานแซคชันอื่นหรือไม่”

1) ถ้ามีการขัดแย้งกัน ระบบจะยกเลิกผลลัพธ์ทรานแซคชันนั้น และมีการ Roll Back เพื่อเริ่มต้นทำงานทรานแซคชันนั้นใหม่ต่อไป

2) ถ้าไม่มีการขัดแย้งกัน ระบบจะนำข้อมูลที่บันทึกไว้ในพื้นที่ที่กำหนดไว้ชั่วคราวนี้บันทึกลงในฐานข้อมูลจริงต่อไป

การทำงานของออฟติมิสติค แบ่งออกเป็น 3 ขั้นตอน คือ

- ขั้นตอนการอ่าน (Read Phase) ทรานแซคชันจะอ่านข้อมูลจากฐานข้อมูล และทำการปรับปรุงแก้ไขข้อมูลที่ต้องการ หลังจากนั้นจะทำการบันทึกไว้ในพื้นที่ชั่วคราวที่กำหนดไว้ (Private Workspace) สำหรับทรานแซคชันนั้น ๆ ซึ่งทรานแซคชันอื่นจะไม่สามารถเข้ามาเกี่ยวข้องได้
- ขั้นตอนการตรวจสอบ (Validation phase) ระบบจะทำการตรวจสอบลำดับการทำงานของแต่ละทรานแซคชัน โดยพิจารณาจากทรานแซคชันที่มีการยืนยันความสำเร็จในการทำงานด้วยคำสั่ง commit แล้ว ถ้ามีปัญหาก็จะยกเลิกการทำงานและผลลัพธ์ของทรานแซคชันนั้น และเริ่มต้นการทำงานใหม่
- ขั้นตอนการบันทึก (Write phase) ในกรณีที่ไม่เกิดปัญหาการขัดแย้งกันระหว่างทรานแซคชัน ข้อมูลที่เก็บไว้ในพื้นที่ชั่วคราว จะถูกนำมาบันทึกถาวรลงในฐานข้อมูล แต่ถ้าหากมีปัญหาจากขั้นตรวจสอบก็จะไม่มีการบันทึกข้อมูลลงในฐานข้อมูล และทรานแซคชันนั้นจะต้องทำงานใหม่

4.2.2 วิธีควบคุมภาวะพร้อมกันแบบไทม์แสตมป์ วิธีการควบคุมแบบนี้ทำงานโดยระบบจัดการฐานข้อมูลจะเป็นตัวกำหนดลำดับของแต่ละทรานแซคชันในการเข้าทำงาน โดยจะทำการเรียงตามลำดับการเกิดก่อน–หลังของแต่ละทรานแซคชันไปเรื่อย ๆ ดังนั้น การควบคุมภาวะพร้อมกันโดยวิธีนี้จะไม่มีปัญหาการเกิดภาวะติดตาย ในการกำหนดค่าไทม์แสตมป์ให้กับแต่ละทรานแซคชันกระทำได้ดังนี้

1) กำหนดตัวแปรเพื่อเก็บค่าไทม์แสตมป์ โดยนำค่าของเวลาในระบบเครื่องคอมพิวเตอร์ (System Clock) มาใช้กำหนดลำดับของแต่ละทรานแซคชัน โดยทำงานเรียงลำดับการ*เข้าตามเวลา*ก่อน–หลังของแต่ละทรานแซคชัน

2) กำหนดตัวแปรโดยสร้างตัวนับ (Counter) ขึ้นเอง โดยตัวแปรตัวนับจะเป็นตัวนับเพิ่มทีละ 1 เพื่อนำมาเป็นตัวกำหนดลำดับที่ ให้กับแต่ละทรานแซคชัน ดังนั้นเมื่อมี ทรานแซคชันต้องการเรียกใช้ข้อมูล ก็จะกำหนดตัวนับให้เพิ่มทีละหนึ่งโดยเรียงลำดับ 1, 2, 3,...ไปเรื่อย ๆ เพื่อจัดลำดับการทำงานของทรานแซคชัน หากค่าตัวนับเพิ่มค่าไปจนถึงค่าสูงสุด แล้วจะต้องมีการเซต (Reset) ตัวนับให้เป็นศูนย์

โดยสรุปจะเห็นว่าวิธีการควบคุมภาวะพร้อมกันโดยเทคนิคไทม์แสตมป์จะไม่ทำให้เกิดเหตุการณ์ภาวะติดตาย เพราะเป็นการเรียงลำดับตามการเกิดของทรานแซคชันก่อนหลังไปเรื่อย ๆ

T_{11}	T_{12}
Read (Q)	
	Read (Q)
	Q : = Q–200
	Read (R)
	R : = R+200
Read (R)	
<Validate>	
Display (Q + R)	
	<Validate>
	Write (Q)
	Write (R)
	Display (Q + R)

ภาพที่ 9.11 การควบคุมภาวะพร้อมกันแบบวิธีออฟติมิสติค

T_9	T_{10}
Read (Q)	
	Write (Q)
	Commit
Commit	

ภาพที่ 9.12 การควบคุมภาวะพร้อมกันแบบไทม์แสตมป์

5. การกู้คืนหรือการฟื้นสภาพฐานข้อมูล (Database Recovery)

เป็นที่ทราบกันดีแล้วว่าหน้าที่พื้นฐานของคอมพิวเตอร์ คือ การประมวลผลโปรแกรม ที่ประกอบด้วยข้อมูล และคำสั่งจำนวนหนึ่ง ซีพียูจะทำหน้าที่ในการประมวลผลแต่ละคำสั่งที่อยู่ในโปรแกรมโดยอ่านคำสั่งจากหน่วยความจำเข้ามาทีละคำสั่ง จากนั้นจะทำการประมวลผล คำสั่งนั้น และการประมวลผลที่เกิดขึ้นภายในฐานข้อมูล เช่น การอ่าน การเขียน การบันทึก ข้อมูล มีผลโดยตรงต่อความคงสภาพของข้อมูล (Data Integrity) ผลลัพธ์จากการทำงานบางอย่างของฐานข้อมูลจะทำให้เกิดความเสียหายกับข้อมูลได้ ซึ่งจะมีผลให้ฐานข้อมูลอยู่ในสถานะที่ไม่มั่นคง ดังนั้นระบบปฏิบัติการฐานข้อมูลจำเป็นอย่างยิ่งที่จะต้องรักษากฎเกณฑ์ที่เกี่ยวกับการทำงานในฐานข้อมูลเอาไว้เพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้น

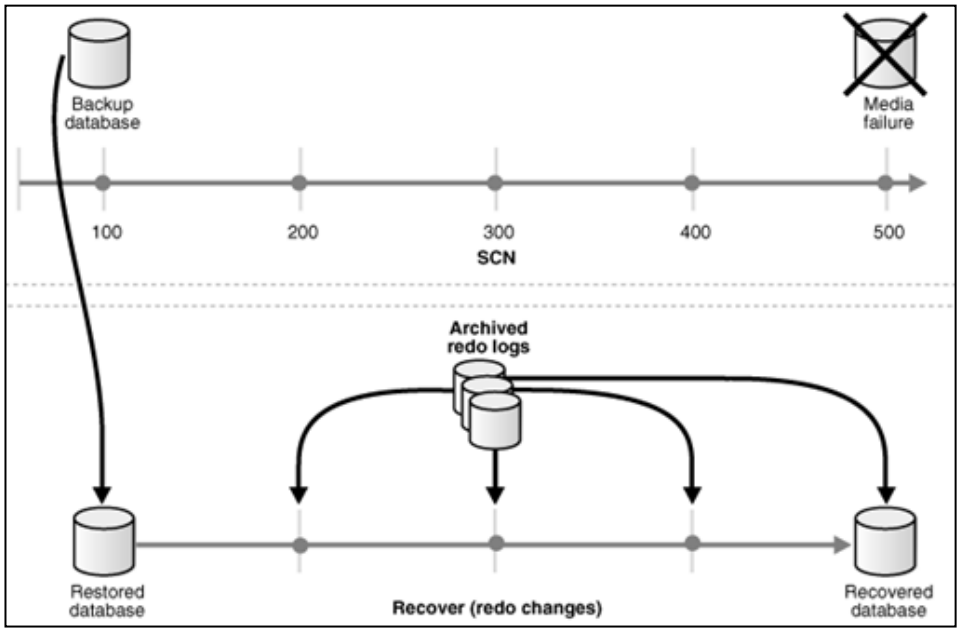
การจัดเก็บข้อมูลชั่วคราวของหน่วยความจำสำรองในคอมพิวเตอร์จะต้องใช้อุปกรณ์ต่าง ๆ หลายประเภท ทั้งประเภทที่จะเก็บข้อมูลได้ก็ต่อเมื่อมีกระแสไฟฟ้า (Volatile Storage) จ่ายให้กับวงจรหน่วยความจำเพื่อรักษาข้อมูลไว้ได้ หากไฟฟ้าดับคือไม่มีไฟฟ้า หรือปิด เครื่องคอมพิวเตอร์ ข้อมูลที่เก็บอยู่ในหน่วยความจำประเภทนี้จะหายไปหมด เช่น RAM และ Cache Memory และประเภทที่สามารถจัดเก็บข้อมูลได้โดยไม่ขึ้นกับไฟฟ้าที่เลี้ยงวงจร (Nonvolatile Storage) เช่นดิสก์ และ ประเภทที่ไม่ต้องอาศัยกระแสไฟฟ้าในการจัดเก็บข้อมูล เช่น เทปแม่เหล็ก (Magnetic Tape) ซึ่งเป็นอุปกรณ์ที่มีการใช้กันมานานแล้วสำหรับเก็บข้อมูลจำนวนมาก และมีการจัดเก็บและเรียกค้นข้อมูลแบบเป็นลำดับ ในขณะที่ข้อมูลผ่านอุปกรณ์เหล่านี้ อาจจะมีข้อผิดพลาดเกิดขึ้นได้ ดังนั้นจำเป็นที่จะต้องคำนึงถึงประเภทของอุปกรณ์ที่ใช้ในการจัดเก็บข้อมูลขณะที่เกิดข้อผิดพลาดหรือขัดข้องขึ้นด้วย

5.1 ความหมาย

การกู้คืนฐานข้อมูล หรือบางครั้งอาจจะเรียกว่า *การฟื้นฟูสภาพ* เป็นกระบวนการที่ระบบจัดการฐานข้อมูลจัดการกับข้อมูลให้ย้อนกลับไปอยู่ในสภาพเดิมที่ถูกต้อง หรือเป็นการกู้ข้อมูลที่เกิดความล้มเหลวจากการใช้งาน ให้กลับมาอยู่ในสภาพที่ถูกต้องสามารถใช้งานได้

การกู้คืนฐานข้อมูล เป็นกระบวนการเรียกคืนฐานข้อมูลให้กลับสู่ภาวะเดิมที่สามารถใช้งานได้ ในกรกณีที่เกิดเหตุขัดข้องบางประการในระหว่างการประมวลผล ซึ่งเป็นเรื่องที่มีความจำเป็นอย่างยิ่ง เนื่องจากการใช้งานอาจเกิดความล้มเหลวได้ตลอดเวลา ไม่ว่าจะเป็นความผิดพลาดจากสาเหตุใด ๆ ก็ตามที่อาจไปทำลายข้อมูล

บางส่วน หรือทำให้ข้อมูลนั้นไม่ถูกต้อง ไม่น่าเชื่อถือ ดังนั้นจึงต้องมีวิธีการเพื่อจะนำข้อมูลที่ถูกทำลายไป กลับคืนมาและอยู่ในสภาพที่ถูกต้องน่าเชื่อถือดังเดิม



ภาพที่ 9.13 แสดงการสำรองและการกู้คืนข้อมูล

(ที่มา : https://docs.oracle.com/cd/B19306_01/server.102/b14220/backrec.htm#i1007108)

5.2 ประเภทของการเกิดความขัดข้อง (Types of Failures)

สาเหตุของความขัดข้องที่เกิดกับฐานข้อมูลอาจจะมาจากหลายสาเหตุด้วยกัน เช่นความผิดพลาดจากโปรแกรมแอปพลิเคชัน ความขัดข้องที่เกิดจากภัยธรรมชาติ การไม่ดูแลเอาใจใส่ หรือเกิดจากการก่อวินาศกรรมก็ได้ แต่การขัดข้องหลัก ๆ ที่เกิดกับฐานข้อมูลมี 3 สาเหตุ ได้แก่

5.2.1 ความขัดข้องของระบบ (System Failures / System Error) หรือ Soft Crash เป็นความขัดข้องที่เกิดจากสาเหตุของการทำงานของระบบที่ไม่ถูกต้องหรือไม่ครบถ้วนพอ เช่น ปัญหาที่เกิดจากการจัดลำดับการทำงานของทรานแซคชันต่าง ๆ ที่กำลังดำเนินการอยู่ไม่ดีพอส่งผลให้เกิดปัญหาภาวะติดตายของข้อมูลขึ้น ทำให้ระบบไม่สามารถทำงานต่อไปได้จำเป็นต้องปิดระบบคอมพิวเตอร์และเปิดเครื่องใหม่ (Restart) ซึ่งจะมีผลกับทุกทรานแซคชันที่กำลังดำเนินอยู่ โดยข้อมูลที่บันทึกในฐานข้อมูลจะไม่สูญหาย แต่ข้อมูลที่อยู่ในหน่วยความจำหลักซึ่งอยู่ระหว่างการทำงานและยังไม่ได้บันทึกจะสูญหายไปได้

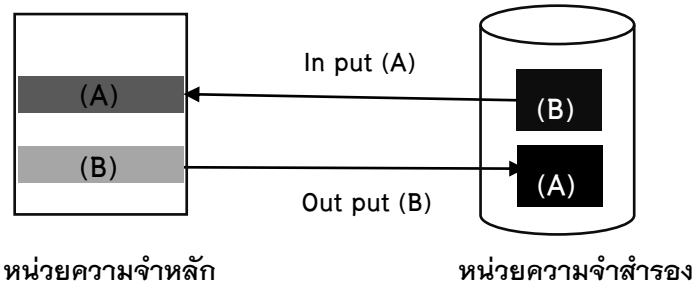
5.2.2 ความขัดข้องของสื่อบันทึกข้อมูล (Media Failures) หรือ *Hard Crash* เป็นความผิดพลาดต่าง ๆ ที่เกิดขึ้นอุปกรณ์ที่ใช้จัดเก็บข้อมูล เช่น ปัญหาฮาร์ดดิสก์พัง แฟ้ม ถูกลบทำให้ข้อมูลที่จัดเก็บอยู่ในดิสก์บางส่วนสูญหายไป ความขัดข้องแบบนี้เป็นความขัดข้องที่จะเกิดผลกระทบต่อทรานแซคชันทั้งหมดหรือเกือบทั้งหมดที่กำลังดำเนินการอยู่

5.2.3 ความขัดข้องของทรานแซคชัน (Transaction Failures) ความขัดข้องแบบนี้มีผลกระทบเฉพาะทรานแซคชันที่มีความขัดข้องเกิดขึ้นเท่านั้น ความขัดข้องของทรานแซคชันมี 2 ลักษณะ คือ ความขัดข้องที่สามารถป้องกันได้ และความขัดข้องที่ไม่สามารถป้องกันได้

5.3 วิธีการพื้นฐานในการกู้คืนฐานข้อมูล (Basic Recovery)

จะเห็นว่าวิธีการใหม่ ๆ ในการเก็บรักษาหรือถ่ายโอนข้อมูลดิจิทัลมีแนวโน้มที่จะทำให้ข้อมูลสูญหาย ด้วยสาเหตุต่าง ๆ ดังนั้นการสำรองข้อมูลไว้ และมีเครื่องมือกู้คืนข้อมูลที่ผ่านการทดสอบอย่างดี จึงเป็นสิ่งสำคัญมาก

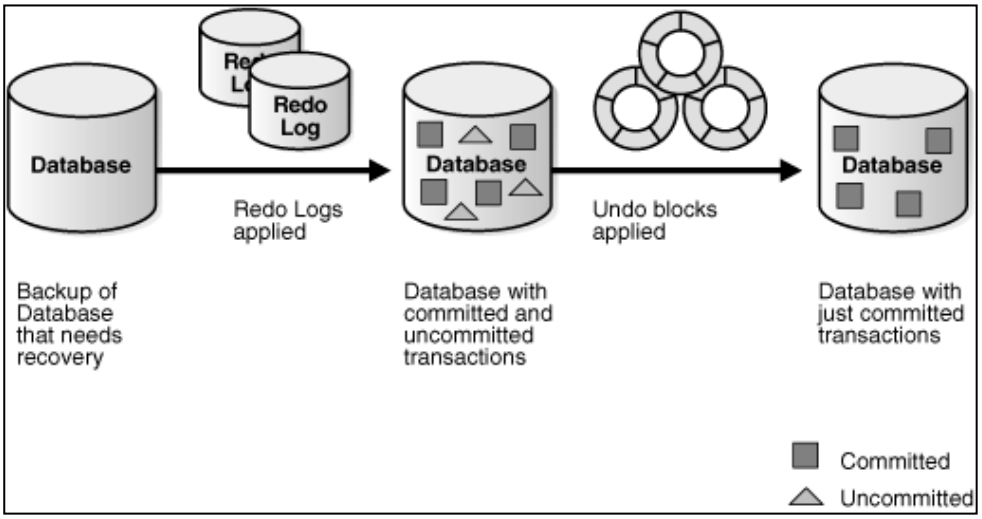
ในระบบคอมพิวเตอร์ การจัดเก็บข้อมูลจะเก็บในหน่วยความจำสำรอง ซึ่งส่วนใหญ่จะเป็นดิสก์ โดยที่พื้นที่ของดิสก์จะแบ่งออกเป็นส่วนๆ (Block) หรือเป็นหน่วย ที่ใช้ในการเคลื่อนย้ายข้อมูลระหว่างดิสก์กับหน่วยความจำหลัก (Memory) พื้นที่ที่ใช้ในการพักข้อมูลในหน่วยความจำหลักเรียกว่า บัฟเฟอร์ (Buffer) หรือ ที่พักข้อมูลชั่วคราว



ภาพที่ 9.14 การเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำหลักและหน่วยความจำสำรอง

In put (A) เคลื่อนย้ายข้อมูลจากหน่วยความจำหลักมายังหน่วยความจำสำรอง และ Out put (B) เคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราว ในหน่วยความจำหลักมายังหน่วยความจำสำรอง

วิธีการพื้นฐานในการกู้คืนฐานข้อมูลมี 3 วิธี ด้วยกัน คือ การยกเลิก (Undo) การทำซ้ำ (Redo) และจุดตรวจสอบ (Checkpoint) ความขัดข้องบางกรณีอาจจะใช้เพียงวิธีใดวิธีหนึ่งหรืออาจจะใช้ทั้ง 2 วิธี หรือทั้ง 3 วิธีร่วมกันก็ได้



ภาพที่ 9.15 แสดงวิธีการกู้คืนพื้นฐาน : Rolling Forward and Rolling Back

(ที่มา : https://docs.oracle.com/cd/B19306_01/server.102/b14220/backrec.htm#i1007108)

5.3.1 การยกเลิก เป็นการยกเลิกสิ่งที่ทำมาแล้วทั้งหมด สำหรับทรานแซคชันที่กำลังทำงานอยู่ และเกิดความขัดข้องขึ้นระหว่างการดำเนินงานหรือก่อนที่จะทำงานเสร็จสมบูรณ์ หรือ Commit การยกเลิกนี้จะยกเลิกการกระทำที่ได้ทำมาแล้วทั้งหมด

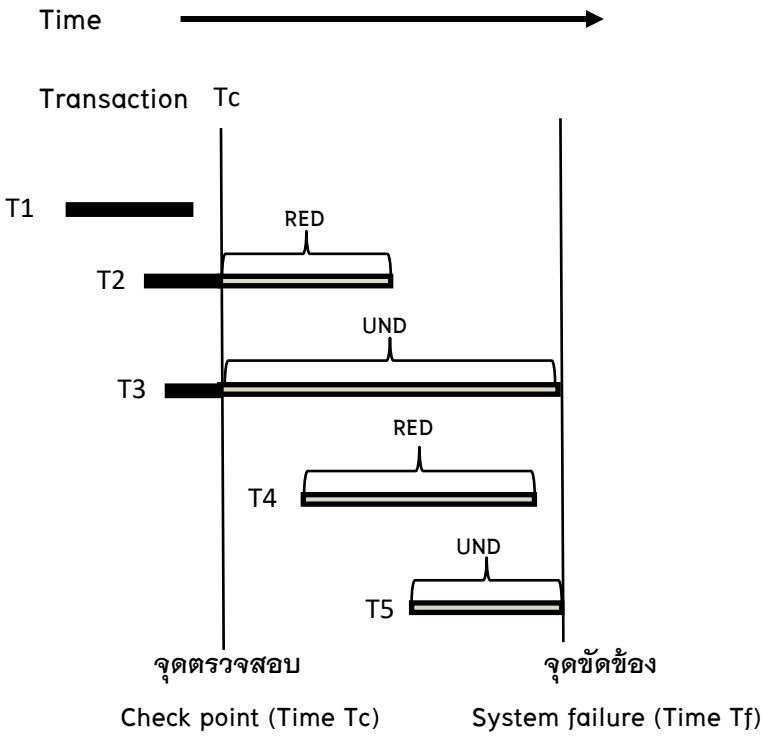
5.3.2 การทำซ้ำ หมายถึงการทำซ้ำสำหรับทรานแซคชันที่ทำงานเสร็จแล้ว แต่ยังไม่มีการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราวลงสู่ฐานข้อมูลจริงแต่เกิดปัญหาขัดข้องเสียก่อน ซึ่งเสมือนว่าทรานแซคชันยังไม่ทำงาน จึงต้อง*กระทำซ้ำ*ทรานแซคชันนั้น

5.2.3 จุดตรวจสอบ หมายถึง เวลา ณ ขณะใดขณะหนึ่งที่ระบบปฏิบัติการและระบบจัดการฐานข้อมูลใช้ทำการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราวในหน่วยความจำหลักลงฐานข้อมูลในดิสก์ โดย*จุดตรวจสอบ*จะถูกบันทึกในไฟล์ประวัติ (Log File) เป็นระยะ ๆ เพื่อประโยชน์ในการกู้คืนฐานข้อมูล

การกำหนดจุดตรวจสอบ (Tc) บ่อย และ ห่างมีทั้งข้อดีและข้อเสีย คือ

1) ถ้ากำหนดจุดตรวจสอบถี่ ข้อดีคือการกู้คืนจะเร็วขึ้น เพราะสามารถยกเลิกและทำซ้ำทรานแซคชันที่เกิดการขัดข้องได้ในช่วงเวลาของจุดตรวจสอบนั้น โดยไม่ต้องกู้คืนฐานข้อมูล ณ จุดเริ่มต้นของทรานแซคชัน และไม่เปลืองเนื้อที่พักข้อมูลชั่วคราวในหน่วยความจำหลัก แต่ข้อเสียคือต้องใช้เวลาในการเคลื่อนย้ายข้อมูลจากหน่วยความจำหลักมายังฐานข้อมูลในดิสก์มากกว่าปกติ อาจทำให้เกิดปัญหาคอขวด (Bottle neck) ขึ้นได้ และอาจจะลดประสิทธิภาพการทำงานของระบบเพราะต้องติดต่อกับส่วนรับเข้า/ส่งออก (I/O) บ่อย ๆ

2) ถ้ากำหนดจุดตรวจสอบห่าง จะมีข้อเสียคือ อาจทำให้เปลืองเนื้อที่ในหน่วยความจำหลักเพราะต้องใช้ที่พักข้อมูลชั่วคราวขนาดใหญ่ในการเก็บข้อมูลจำนวนมากที่ยังไม่ได้เคลื่อนย้ายลงฐานข้อมูลในดิสก์ ดังนั้นช่วงเวลาของการทำจุดตรวจสอบว่าจะเลือกให้ถี่แค่ไหนขึ้นกับลักษณะและปริมาณของทรานแซคชัน เช่น ทรานแซคชันขนาดใหญ่ ๆ ก็ไม่ควรกำหนดจุดตรวจสอบถี่ เพราะจะเสียเวลาในการเคลื่อนย้ายข้อมูลมาก



ภาพที่ 9.16 แสดงกำหนดจุดตรวจสอบและจุดขัดข้อง

จากภาพจะเห็นว่า เมื่อเกิดความขัดข้องขึ้น ณ เวลา Tf ใดๆ จะมีเฉพาะทรานแซคชัน T1 เท่านั้นที่ทำงานเสร็จสมบูรณ์ และปลอดภัย เพราะ Commit หรือมีการยืนยันไปก่อนแล้วก่อนจุดตรวจสอบครั้งสุดท้าย ณ เวลา Tc และข้อมูลได้ถูกเก็บไว้ในฐานข้อมูลจริงแล้ว

T1 : เป็นทรานแซคชันที่เกิดขึ้นและทำงานเสร็จสมบูรณ์และปลอดภัย ก่อนที่จะถึงจุดตรวจสอบ เมื่อระบบเกิดขัดข้อง ระบบ*ไม่ต้อง*ทำรายการนี้ใหม่

T2 : เป็นรายการที่เริ่มทำงานก่อนจะถึงเวลาจุดตรวจสอบ แต่ทำงานเสร็จหลังจากผ่านจุดตรวจสอบไปแล้ว เมื่อระบบเกิดความผิดพลาดและมีการกู้คืน ระบบจะต้องทำรายการนี้ใหม่ (REDO)เฉพาะส่วนที่อยู่หลังจุดตรวจสอบ

T3 : เป็นรายการที่เริ่มทำงานหลังจุดตรวจสอบ และทำงานยังไม่เสร็จก็เกิดความผิดพลาดของระบบขึ้นก่อนที่จะทำงานเสร็จ ระบบจะยกเลิก (UNDO) รายการนี้ทั้งหมดจนถึงก่อนจุดตรวจสอบ

T4 : เป็นรายการที่เริ่มหลังจุดตรวจสอบและทำงานเสร็จก่อนถึงจุดที่ระบบเกิดความผิดพลาด ระบบจะต้องนำรายการนี้กลับมาทำใหม่ทั้งหมด

T5 : เป็นรายการที่เริ่มทำงานหลังจดตรวจสอบ แต่ทำยังไม่เสร็จระบบก็เกิดความผิดพลาดก่อน ระบบจะต้องยกเลิก (UNDO) รายการนี้ทั้งหมด

เมื่อสิ้นสุดการประมวลผล จะพบว่าใน Undo List ประกอบด้วยทรานแซคชัน T3 และ T5 ให้*ยกเลิกการทำงานของทรานแซคชันนั้น* โดยการย้อนกลับการทำงานกลับสู่ค่าเริ่มต้น ส่วนใน Redo List ประกอบด้วย ทรานแซคชัน T2 และ ทรานแซคชัน T4 ให้นำทรานแซคชันที่อยู่ใน Redo List มา*ประมวลผลใหม่*

5.4 การกู้คืนฐานข้อมูลที่เกิดจากความขัดข้องของระบบ หรือ Soft crash

ความขัดข้องของระบบ หรือ Soft crash เป็นความขัดข้องที่เกิดจากหน่วยความจำ หรือสาเหตุจากความขัดข้องของทรานแซคชัน (transaction failures) หรือความผิดพลาดที่เกิดจากการทำงานตัวระบบที่ไม่ถูกต้อง ไม่ครบถ้วนพอ เช่น การจัดลำดับการทำงานของโปรแกรมต่าง ๆ ไม่ดีพอ ส่งผลให้เกิดปัญหาภาวะติดตาย ของข้อมูลเป็นต้น

การกู้คืนฐานข้อมูลแบบนี้ กระทำโดยจะกำหนดให้มีจุดตรวจสอบเพื่อใช้ตรวจสอบการทำงานของ ทรานแซคชัน ซึ่งจะใช้หลักการของ Transaction Log หรือ Journal (ตารางข้อมูลที่มีการจัดเก็บประวัติการเปลี่ยนแปลงข้อมูลในฐานข้อมูล) ในการกู้คืนข้อมูลของฐานข้อมูล โดยการกำหนดจุดเริ่มต้นของทรานแซคชันที่เริ่มบันทึกรายการลงในตารางหรือ Log File ที่เรียกว่าจุดตรวจสอบไว้ หากมีทรานแซคชันใดทำงานไม่สมบูรณ์ ให้นำข้อมูลใน Log File ตั้งแต่จุดตรวจสอบที่มาประมวลผลใหม่ โดยไม่จำเป็นต้องเอาทุกรายการในตารางข้อมูลมาประมวลผล

5.4.1 การกู้ข้อมูลแบบ Log-Based เป็นการกู้ข้อมูลที่อาศัยข้อมูลจาก ไฟล์ประวัติ (Log File) ซึ่งเป็นแฟ้มข้อมูลที่ใช้ในการบันทึกการกระทำต่าง ๆ ในทรานแซคชัน เพื่อบอกสถานะของข้อมูล และสถานะของทรานแซคชัน

Transaction Log หรือ Journal คือ ตารางข้อมูลที่จัดเก็บประวัติการเปลี่ยนแปลงข้อมูลในฐานข้อมูล การกู้คืนข้อมูลของฐานข้อมูลตัวจัดการกู้คืน (Recovery Manager) ที่อยู่ในระบบการจัดการฐานข้อมูลจะใช้ข้อมูลใน Transaction Log ช่วยในการกู้คืนข้อมูล

การกู้คืนข้อมูลไม่ว่าจะเกิดจากความขัดข้องสาเหตุใดก็ตาม จะใช้ไฟล์ประวัติเข้ามาช่วยในการนำข้อมูลกลับคืนมาให้อยู่ในสภาวะที่ถูกต้องเสมอ

การจัดเก็บข้อมูลในตารางไฟล์ประวัติ จะประกอบไปด้วยรายละเอียดดังนี้

- ชื่อทรานแซคชัน (Transaction name) ชื่อทรานแซคชันที่กระทำการบันทึก (Write Operation)
- ลำดับหมายเลขรายการ (Log Sequence Number: LSN)
- หมายเลขรายการเปลี่ยนแปลง (Transaction Identifier)
- การกระทำ (Action)
- เวลา (Time)
- ตารางข้อมูลที่มีการเปลี่ยนแปลง (Table)
- รายการข้อมูลที่มีการเปลี่ยนแปลง (Record)
- ชื่อฟิลด์ (column)
- ค่าข้อมูลเก่า ก่อนการบันทึก หรือ Before Image
- ค่าข้อมูลใหม่ หลังการบันทึก หรือ After Image

รูปแบบรายการที่บันทึกในไฟล์ประวัติ แบ่งออกได้เป็น 3 ประเภทคือ

- Transaction Start เป็นรายการที่จะถูกบันทึกลงใน Log File เมื่อ Transaction เริ่มต้นทำงาน
- Transaction Commit เป็นรายการที่จะถูกบันทึกลงใน Log File เมื่อ Transaction ทำงานเสร็จ
- Update Activity เป็นรายการที่จะถูกบันทึกลงใน Log File ทุกครั้ง เมื่อการกระทำนั้นก่อให้เกิดการเปลี่ยนแปลงค่าของข้อมูล

Tid	Time	Operation	Object	Before image	After image	pPtr	nPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

ภาพที่ 9.17 ตัวอย่างตารางไฟล์ประวัติ

ตัวอย่างที่ 2 การโอนเงินจากบัญชีมานะไปบัญชีมานี :

สมมติบัญชีมานะ มียอดเงิน 1,000 บาท บัญชีมานีมียอดเงิน 800 บาท ต้องการโอนเงิน 300 บาท จากบัญชีมานะ ไปยังบัญชีมานี มีขั้นตอนการทำงานดังนี้

ขั้นตอนที่ 1 อ่านยอดเงินบัญชีมานะจากฐานข้อมูลในดิสก์ไปเก็บไว้ในที่พักชั่วคราว ในหน่วยความจำหลัก

ขั้นตอนที่ 2 คำนวณยอดเงินบัญชีมานะ โดยหักเงินโอน 300 บาท ดังนั้นยอดเงินในบัญชีมานะใหม่ คือ 700 บาท ซึ่งทำให้ค่าของข้อมูลในที่พักข้อมูลชั่วคราวเปลี่ยนแปลงไป

ขั้นตอนที่ 3 อ่านยอดเงินจากบัญชีมานีจากฐานข้อมูลในดิสก์ มาเก็บไว้ในที่พักข้อมูลชั่วคราว

ขั้นตอนที่ 4 คำนวณยอดเงินบัญชีมานี ได้ยอดเงินใหม่ คือ 1100 บาท ซึ่งจะเก็บไว้ในที่พักข้อมูลชั่วคราว

ขั้นตอนที่ 5 เมื่อที่พักข้อมูลชั่วคราวเต็ม จะมีการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราวไปเก็บในฐานข้อมูลในดิสก์ ถ้ายอดเงินบัญชีมานะถูกบันทึกลงฐานข้อมูลในดิสก์ก่อน แต่ยอดเงินบัญชีของมานียังไม่ถูกบันทึกลงฐานข้อมูลในดิสก์ และเกิดปัญหาการขัดข้องขึ้นในระหว่างนี้ จะทำให้ข้อมูลในที่พักข้อมูลชั่วคราวสูญหายไป จะมีผลทำให้บัญชีมานี ในฐานข้อมูลในดิสก์ไม่ถูกต้อง

ดังนั้นเพื่อความมั่นใจว่าข้อมูลในฐานข้อมูลถูกต้องทั้งหมด จึงต้องทำซ้ำข้อมูลตั้งแต่ต้นของไฟล์ประวัติซึ่งกรณีนี้เป็นวิธีที่ไม่มีประสิทธิภาพ เพราะจะต้องทำซ้ำใหม่ตั้งแต่ต้น ดังนั้นจึงมีจุดตรวจสอบขึ้นเพื่อให้ทราบว่าจะทำการกู้คืนข้อมูลย้อนหลังถึงจุดไหน คือทำให้การกู้คืนข้อมูลทำได้เร็วขึ้น

การกู้คืนข้อมูลแบบ Log-Based มี 3 วิธี คือ การยกเลิก ทำซ้ำ และจุดตรวจสอบ ซึ่งความขัดข้องบางกรณีอาจใช้เพียงวิธีใดวิธีหนึ่ง หรืออาจจะใช้ทั้ง 2 วิธี หรือใช้ทั้ง 3 วิธีร่วมกันได้

5.5 การกู้คืนฐานข้อมูลที่เกิดขึ้นจากความขัดข้องของสื่อบันทึกข้อมูล หรือ Hard Crash

ความขัดข้องที่เกิดกับสื่อข้อมูล เช่นบางส่วนของหน่วยความจำสำรองถูกทำลาย อาจมีผลไปทำลายข้อมูลในสื่อข้อมูลได้ การกู้คืนฐานข้อมูลในกรณีนี้ คือ

5.5.1 อาศัยข้อมูลสำรองครั้งล่าสุด ซึ่งควรจะมีการทำสำรองข้อมูลไว้อย่างสม่ำเสมอ

5.5.2 ใช้ข้อมูลในไฟล์ประวัติ ทำการทำซ้ำสำหรับทรานแซคชันที่ได้กระทำเสร็จสมบูรณ์แล้ว

จะเห็นว่าการสำรองข้อมูลมีความจำเป็นอย่างมาก หากมีการสำรองข้อมูลบ่อย ๆ ก็ยิ่งทำให้การกู้คืนฐานข้อมูลรวดเร็วมากขึ้น เพราะข้อมูลสำรองมีความทันสมัยใกล้เคียงข้อมูลที่ ถูกทำลาย อย่างไรก็ตามการสำรองข้อมูลมักต้องใช้

เวลานาน หากเป็นระบบใหญ่ ๆ ทำให้ไม่สามารถใช้ระบบในงานด้านอื่น ๆ ได้ จึงควรกำหนดช่วงเวลาการสำรองข้อมูลให้เหมาะสม

5.6 การกู้คืนฐานข้อมูลแบบ Shadow Paging

เป็นการกู้คืนข้อมูลที่จัดเก็บไว้ในอุปกรณ์ชั่วคราวที่จะจัดเก็บได้ก็ต่อเมื่อมีกระแสไฟฟ้า (Volatile Storage) ในกรณีที่ไม่มีกระแสไฟฟ้าข้อมูลที่จัดเก็บอยู่ในอุปกรณ์จะหายไป เช่น RAM, Cache Memory Volatile Storage

การกู้คืนฐานข้อมูลแบบนี้ เป็นการจัดเก็บข้อมูลเพื่อนำมาใช้ในการกู้คืนข้อมูลโดยการกำหนดเนื้อที่ในดิสก์ไว้ส่วนหนึ่ง เพื่อใช้ในการทำสำเนาข้อมูลที่จะถูกเรียกใช้ในทรานแซคชันไว้ก่อนที่ทรานแซคชันนั้นจะเริ่มต้นทำงาน พื้นที่ที่เตรียมไว้สำหรับสำเนาข้อมูลเรียกว่า Shadow Page

*ข้อมูล*ที่อยู่ใน Shadow Page จะไม่สามารถเปลี่ยนแปลงแก้ไขได้ และจะถูกนำมาใช้เมื่อ ทรานแซคชันที่ทำงานนั้นมีข้อผิดพลาดเกิดขึ้น โดยการนำข้อมูลใน Shadow Page มาบันทึกทับลงในฐานข้อมูลเดิม และในกรณีที่ในทรานแซคชันทำงานสมบูรณ์ ข้อมูลใน Shadow Page จะถูกยกเลิกไป

5.7 การกู้คืนข้อมูลแบบ จุดตรวจสอบ

เป็นการกู้คืนข้อมูลโดยการใช้ไฟล์ประวัติ แต่ในการบันทึกข้อมูลของไฟล์ประวัติจะกระทำเป็นช่วง ๆ แทน โดยมีการกำหนดจุดเริ่มต้นของทรานแซคชันที่เริ่มบันทึกรายการลงใน ไฟล์ประวัติที่เรียกว่าจุดตรวจสอบไว้ เมื่อเกิดมีทรานแซคชันที่ทำงานไม่สมบูรณ์ ก็จะนำข้อมูลในไฟล์ประวัติตั้งแต่จุดตรวจสอบมาประมวลผลใหม่

การป้องกันข้อมูลจากความผิดพลาดของการประมวลผลที่มีผู้ใช้เพียงคนเดียวของระบบจะไม่ยุ่งยากมากนัก แต่การใช้ระบบฐานข้อมูล มักจะพบปัญหาที่เกิดจากภาวการณ์ที่ผู้ใช้เรียกใช้ข้อมูลพร้อม ๆ กัน ไม่ว่าจะเป็นข้อมูลในหน่วยความจำหลักซึ่งกำลังประมวลผลอยู่ หรือเป็นข้อมูลในหน่วยความจำสำรองก็ตาม เพราะผู้ใช้แต่ละคนสามารถเข้าถึงข้อมูล และจัดการข้อมูลพร้อมกันในเวลาเดียวกันได้ การป้องกันข้อมูลจากความผิดพลาดของระบบจะมีความยุ่งยาก และซับซ้อนมาก เมื่อต้องการปรับปรุงข้อมูลค่าใดค่าหนึ่งให้ถูกต้อง จะต้องดำเนินการให้เสร็จสิ้นก่อนที่ข้อมูลค่านั้น จะถูกเรียกใช้โดยผู้ใช้คนอื่น ๆ หรือหากระบบคอมพิวเตอร์เกิดความขัดข้องขึ้น ก็อาจจะมีผลกระทบถึงข้อมูลได้ จำเป็นที่ต้องทำซ้ำใหม่ หรืออาจต้องทำการกู้คืน เพื่อให้ข้อมูลอยู่ในสภาพถูกต้องใช้งานได้

เมื่อเกิดภาวะพร้อมกันในระบบ ระบบจัดการฐานข้อมูล หรือ DBMS จะต้องมีเทคนิค หรือกระบวนการในการจัดเรียงลำดับการทำงานของแต่ละทรานแซคชันก่อนและหลังป้องกันไม่ให้เกิดการแย่งชิงข้อมูลที่ยังประมวลผลไม่เสร็จและนำไปใช้งานโดยทรานแซคชันอื่น จึงต้องมีตัวจัดตารางการทำงาน หรือจัดเรียงลำดับของแต่ละทรานแซคชันก่อนหลัง ซึ่งเทคนิคการควบคุมภาวะพร้อมกันจะมีอยู่หลายเทคนิคด้วยกัน เช่นวิธีการล็อก (Locking) และไม่ใช้วิธีการล็อก (Without Locking) ดังที่กล่าวมาแล้วข้างบน

****