แผนการสอนประจำบทเรียน

รายชื่ออาจารย์ผู้จัดทำ ผู้ช่วยศาสตราจารย์ ณัฏฐพร พิมพายน

รายละเอียดของเนื้อหา

ตอนที่11.1 **การฟื้นสภาพ** (Recovery)

เรื่องที่ 11.1.1 ประเภทของการเกิดความขัดข้อง

เรื่องที่ 11.1.2 การกู้ข้อมูลจากความขัดข้องที่เกิดจากระบบคอมพิวเตอร์

เรื่องที่ 11.1.3 การกู้ข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของรายการและสื่อบันทึกข้อมูล

ตอนที่ 11.2 **การควบคุมภาวะความพร้อมกัน**

เรื่องที่ 11. 2.1 แนวคิดเกี่ยวกับการใช้ข้อมูลพร้อมกัน

เรื่องที่ 11.2.2 ประเภทและระดับของการล็อค

เรื่องที่ 11.2.3 ปัญหาและวิธีการแก้ไข deadlock

เรื่องที่ 11.2.4 วิธีการควบคุมภาวะความพร้อมกัน

แนวคิด

1. การประมวลผลข้อมูลที่มีผู้ใช้เพียงคนเดียว การป้องกันข้อมูลจากความผิดพลาดของระบบจะไม่เกิดความยุ่งยากมากนัก แต่การประมวลผลที่มีผู้ใช้หลายคนในระบบพร้อมกันและมีการใช้ฐานข้อมูลร่วมกันในการการป้องกันข้อมูลจากความผิดพลาดของระบบ(recovery)จะมีความยุ่งยากและซับซ้อนมาก ซึ่งจะต้องมีการการฟื้นสภาพ (Recovery)โดยการกู้ข้อมูล
2. ฐานข้อมูลเป็นระบบที่ให้ผู้ใช้หลาย ๆคนสามารถใช้ข้อมูลร่วมกัน ผู้ใช้แต่ละคนสามารถเข้าถึงข้อมูลและจัดการข้อมูลพร้อมกันในเวลาเดียวกัน ดังนั้น DBMS จะต้องมีการควบคุมภาวะการเข้าถึงข้อมูลพร้อมกัน(Concurrency Control)

วัตถุประสงค์

หลังจากศึกษาบทเรียนที่ 11 แล้ว นักศึกษาสามารถ

1. บอกลักษณะและความหมายของการการฟื้นสภาพได้
2. บอกการควบคุมภาวะการเข้าถึงข้อมูลพร้อมกันได้

กิจกรรมการเรียนการสอน

กิจกรรมที่นักศึกษาต้องทำสำหรับการเรียนการสอน ได้แก่

1. ศึกษาเอกสารการสอน
2. ปฏิบัติกิจกรรมตามที่ได้รับมอบหมายในเอกสารการสอนแต่ละตอน

สื่อการสอน

1. เอกสารการสอนของชุดวิชา
2. แบบฝึกปฏิบัติ
3. บทความ/ข้อมูลทางคอมพิวเตอร์
4. การให้คำปรึกษาทางโทรศัพท์
5. CD-ROM
6. Homepage ของชุดวิชาผ่านทางอินเตอร์เน็ต

เอกสารประกอบการสอน

1. Fundamentals of Database Systems, by Ramez Elmasri, Shamkant B. Navathe, The Second Edition, 1994
2. Database System Concepts, by Abraham Siberschaty, Henry F.Korth, S.Sudarshan, The Third Edition, 1991

ประเมินผล

1. ประเมินผลจากแบบฝึกหัด/ทดสอบ ในแต่ละบท
2. ประเมินผลจากการสอนประจำภาคการศึกษา

ตอนที่11.1 การฟื้นสภาพ

หัวเรื่อง

เรื่องที่ 11.1.1 ประเภทของการเกิดความขัดข้อง

เรื่องที่ 11.1.2 การกู้ข้อมูลจากความขัดข้องที่เกิดจากระบบคอมพิวเตอร์

เรื่องที่ 11.1.3 การกู้ข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของรายการและสื่อบันทึกข้อมูล

แนวคิด

1. ข้อมูลจัดเป็นทรัพยากรที่สำคัญต่อองค์การ การป้องกันการสูญหายของข้อมูลและวิธีการในการกู้ข้อมูลกลับมาใช้จึงเป็นปัจจัยสำคัญที่ผู้ออกแบบฐานข้อมูลต้องคำนึงถึงเมื่อทำการออกแบบฐานข้อมูลขึ้นใช้งาน
2. เทคนิคและวิธีการกู้ข้อมูลที่มีประสิทธิภาพ ทำให้ข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลสูญหายได้ยากมาก และยังสามารถรักษาข้อมูลในฐานข้อมูลให้มีความถูกต้องอยู่เสมอได้อีกด้วย การกู้ข้อมูลจากความขัดข้องที่เกิดจากระบบ(System failures) เป็นการทำให้ฐานข้อมูลมีความคงสภาพหรือไม่มีความขัดแย้งของข้อมูลเกิดขึ้น
3. การกู้ข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของรายการ (Transaction failures)เป็นการใช้คำสั่งการควบคุมทรานแซกชัน คือคำสั่งแสดงจุดเริ่มต้นของทรานแซกชัน(BEGIN WORK) คำสั่งแสดงถึงการเสร็จสิ้นอย่างสมบูรณ์ของทรานแซกชัน(COMMIT WORK) และคำสั่งแสดงการยกเลิกทรานแซกชัน(ROLLBACK WORK) การกู้ข้อมูลจากความขัดข้องที่เกิดจากสื่อบันทึกข้อมูล (media failures) สามารถทำได้ด้วยการนำข้อมูลที่สำรองไว้ในเทปมาลงในดิสค์ใหม่แทนแล้วใช้รายการที่เกิดขึ้นของวันนั้นทำการกู้ข้อมูลคืนกลับมา

วัตถุประสงค์

หลังจากที่ศึกษาตอนที่ 11.1 แล้ว นักศึกษาสามารถ

1. อธิบายและระบุประเภทของการเกิดความขัดข้องได้
2. บอกลักษณะการกู้ข้อมูลจากความขัดข้องที่เกิดจากระบบคอมพิวเตอร์ ได้
3. บอกลักษณะการกู้ข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของรายการได้
4. บอกลักษณะการกู้ข้อมูลจากความขัดข้องที่เกิดจากสื่อบันทึกข้อมูล (media failures)ได้

เรื่องที่ 11.1.1 **ประเภทของการเกิดความข้อข้อง** (Failare)

1.ความหมายของการฟื้นสภาพ

การฟื้นสภาพ (recovery) คือ การที่ระบบจัดการฐานข้อมูลจัดการกับข้อมูลให้ย้อนไปอยู่ในสภาพเดิมที่ถูกต้อง ในการจัดการฐานข้อมูลการเกิดความขัดข้องหรือความเสียหายกับระบบไม่ว่ากรณีใด ๆ ที่อาจไปทำลายข้อมูลบางส่วน หรือทำให้ข้อมูลนั้นไม่ถูกต้อง ไม่น่าเชื่อถือได้ ดังนั้นจึงต้องมีวิธีการเพื่อจะนำข้อมูลที่ถูกทำลายไปกลับคืนมาและอยู่ในสภาพที่ถูกต้องน่าเชื่อถือดังเดิม ซึ่งถือเป็นหน้าที่หนึ่งที่สำคัญของระบบจัดการฐานข้อมูล การฟื้นสภาพเป็นการทำให้มั่นใจว่าทรานแซกชันที่ถูกยกเลิกหรือที่ทำผิดพลาดอันเกิดจากความขัดข้องต่างๆ เช่น ความผิดพลาดของโปรแกรม ดิสก์เสีย ระบบไฟฟ้าขัดข้อง เป็นต้น จะไม่ก่อให้เกิดความเสียหายกับฐานข้อมูล หรือทรานแซกชันอื่นที่ทำงานร่วมกัน

2.ประเภทของการเกิดความขัดข้อง

ความขัดข้องที่เกิดขึ้นกับระบบนั้นมีหลายประการ ประเภทของความขัดข้องจำแนก (type of failures) ได้เป็น 3 ลักษณะ คือ ความขัดข้องของทรานแซกชัน ความขัดข้องของระบบ และความขัดข้องของสื่อข้อมูล

2.1 ความขัดข้องของระบบ (system failures) เป็นความขัดข้องที่จะเกิดผลกระทบต่อทรานแซกชันทั้งหมด หรือเกือบทั้งหมดที่กำลังดำเนินการอยู่ ความผิดพลาดของระบบปฏิบัติการ ทำให้ระบบไม่สามารถทำงานต่อไปได้ จำเป็นต้องปิดระบบคอมพิวเตอร์และเปิดเครื่องใหม่อีกครั้ง(restart) ซึ่งจะมีผลกับทุกทรานแซกชันที่กำลังดำเนินอยู่ แต่ไม่ทำลายฐานข้อมูลที่อยู่ในดิสก์ เรียกความขัดข้องลักษณะนี้ว่า ***ซอฟต์แครช (Soft crash)*** โดยข้อมูลที่บันทึกในฐานข้อมูลจะไม่สูญหาย แต่ข้อมูลที่อยู่ในหน่วยความจำหลักซึ่งอยู่ระหว่างการทำงานและยังไม่ได้บันทึกจะสูญหายไปได้

2.2 ความขัดข้องของทรานแซกชัน (transaction failures) ความขัดข้องลักษณะนี้จะมีผลกระทบเฉพาะทรานแซกชันที่มีความขัดข้องเกิดขึ้นเท่านั้น ความขัดข้องของทรานแซกชันมี 2 ลักษณะ คือ

1) ความขัดข้องที่สามารถป้องกันได้โดยการตรวจสอบภายในโปรแกรม เช่น ทรานแซกชันการโอนเงิน ถ้ายอดเงินโอนมีค่ามากกว่ายอดเงินที่มีอยู่ในบัญชี เมื่อโปรแกรมตรวจพบจะแสดงข้อความผิดพลาด อาทิ “ยอดเงินไม่เพียงพอ (insufficient fund)” ซึ่งลักษณะนี้ต้องทำการยกเลิกการเปลี่ยนแปลงต่าง ๆ โดยปรับข้อมูลให้เป็นค่าเดิมเมื่อเริ่มต้น หรือ ผู้เขียนโปรแกรมกำหนดขอบเขตของทรานแซกชันไม่ถูกต้อง ความขัดข้องดังกล่าวสามารถป้องกันได้โดยผู้เขียนโปรแกรมต้องตรวจสอบความถูกต้องภายในโปรแกรมให้ครอบคลุมทุกกรณีที่มีโอกาสจะละเมิดกฎควบคุมความถูกต้อง และต้องกำหนดขอบเขตของทรานแซกชันให้ถูกต้อง

2) ความขัดข้องที่ไม่สามารถป้องกันได้โดยการตรวจสอบภายในโปรแกรม เช่น การล้น (overflow) เกินพื้นที่ในการทำงาน ซึ่งทำให้มีเนื้อที่ไม่เพียงพอต่อการปฏิบัติการของทรานแซกชัน ซึ่งอาจก่อให้เกิดความผิดพลาดแก่ทรานแซกชันได้

2.3 ความขัดข้องของสื่อข้อมูล (media failures) เป็นความขัดข้องที่จะเกิดผลกระทบต่อทรานแซกชันทั้งหมด หรือเกือบทั้งหมดที่กำลังดำเนินการอยู่ เช่น หัวอ่านกระทบกับแผ่นดิสก์ หรือตัวควบคุมการทำงานของดิสก์ (disk controller) พัง ความขัดข้องของสื่อข้อมูลจะมีผลทำลายฐานข้อมูล หรือบางส่วนของฐานข้อมูล และมีผลกระทบต่อทรานแซกชันที่กำลังใช้งานฐานข้อมูลนั้นอยู่ เรียกความขัดข้องลักษณะนี้ว่าว่า ฮาร์ดแครช (hard crash)

3.วิธีการฟื้นสภาพ

การแก้ปัญหาความขัดข้องต่างๆ จะใช้ ไฟล์ประวัติ (log file) เข้ามาช่วยในการฟื้นสภาพ ซึ่งเรียกวิธีนี้ว่า การฟื้นสภาพแบบล็อกเบส (log-based recovery) มีหลักการ คือ การปฏิบัติการของทรานแซกชันเกี่ยวกับการบันทึก (write operation) จะต้องบันทึก (write) ไว้ในไฟล์ประวัติเสมอ เพื่อบอกถึงสถานะของข้อมูลและสถานะของทรานแซกชัน

3.1 รายละเอียดของไฟล์ประวัติ (logfile) จะประกอบด้วยสิ่งต่าง ๆ ดังนี้

- ชื่อทรานแซกชัน (tranasaction name) คือ ชื่อของทรานแซกชันที่กระทำการบันทึก (write operation)
- ชื่อเดตาไอเท็ม (data item name) คือ ชื่อของรายการข้อมูลที่ถูกบันทึก
- ค่าเก่า (old value) คือ ค่าเก่าของเดตาไอเท็มก่อนการบันทึก
- ค่าใหม่ (new value) คือ ค่าใหม่ของเดตาไอเท็มหลังการบันทึก

3.2 วิธีการฟื้นสภาพ เนื่องจากการฟื้นสภาพแบบล็อกเบส จะต้องปฏิบัติตามกฎเกณฑ์การบันทึก (write-ahead protocal) โดยเรคอร์ดประวัติจะต้องบันทึก (write) ลงไฟล์ประวัติก่อนเสมอ จึงจะยอมให้มีการบันทึกข้อมูลจากที่พักข้อมูลชั่วคราว (buffer) ลงฐานข้อมูลได้ ด้วยกฎเกณฑ์นี้ทำให้เราสามารถที่จะยกเลิกหรือทำซ้ำการเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นกับฐานข้อมูลได้โดยใช้ค่าเก่าหรือค่าใหม่ที่เก็บไว้ไฟล์ประวัติ (log file) การฟื้นสภาพมี 3 วิธี ดังนี้ คือ การยกเลิก การทำซ้ำ และจุดตรวจสอบ ซึ่งความขัดข้องบางกรณีอาจใช้เพียงวิธีใดวิธีหนึ่ง หรืออาจจะใช้ทั้ง 2 วิธี หรือใช้ทั้ง 3 วิธีร่วมกันได้

1) การยกเลิก (UNDO) คือ การยกเลิกสิ่งที่ทำมาแล้วสำหรับทรานแซกชันที่กำลังทำงานอยู่แล้วเกิดความขัดข้องขึ้นระหว่างการดำเนินงานก่อนที่จะคอมมิต (COMMIT) ลักษณะนี้จะยกเลิกการกระทำที่ได้ทำมาแล้วทั้งหมด

2) การทำซ้ำ (REDO) คือ การทำซ้ำสำหรับทรานแซกชันที่ทำงานเสร็จสมบูรณ์แล้ว (COMMIT) แล้วแต่ยังไม่มีการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราว (buffer) ในหน่วยความจำหลักลงสู่ฐานข้อมูลจริงก็เกิดปัญหาขัดข้องเสียก่อน ซึ่งเสมือนว่าทรานแซกชันยังไม่ทำงาน ดังนั้นจึงต้องกระทำซ้ำทรานแซกชันนั้น

3) จุดตรวจสอบ (CHECKPOINT) คือ เวลา ณ ขณะใดขณะหนึ่งที่ระบบปฏิบัติการ (OS) และระบบจัดการฐานข้อมูล (DBMS) ใช้ทำการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราว (buffer) ในหน่วยความจำหลักลงฐานข้อมูลในดิสก์ โดยจุดตรวจสอบจะถูกบันทึกในไฟล์ประวัติ (log file) เป็นระยะๆ เพื่อประโยชน์ในการฟื้นสภาพ การทำจุดตรวจสอบบ่อยหรือถี่ ๆ ย่อมทำให้การฟื้นสภาพเร็วขึ้นแต่อาจจะลดประสิทธิภาพการ

ทำงานของระบบเพราะต้องติดต่อกับส่วนรับเข้า/ส่งออก (I/O) บ่อย ๆ ดังนั้นช่วงเวลาของการทำจุดตรวจสอบว่าจะเลือกให้ถี่แค่ไหนขึ้นกับลักษณะและปริมาณของทรานแซกชัน เช่น ถ้าเป็นทรานแซกชันขนาดใหญ่ ๆ ก็ไม่ควรให้ถี่นักเพราะต้องเสียเวลาเคลื่อนย้ายข้อมูล การกำหนดจุดตรวจสอบว่าควรจะบ่อยครั้งนั้นอาจมีผลเสียคือถ้าจุดตรวจสอบถี่ (checkpoint time หรือ Tc) จะทำให้การฟื้นสภาพเร็วขึ้น เพราะสามารถยกเลิก (UNDO) ทำซ้ำ (REDO) ทรานแซกชันที่เกิดข้อขัดข้องได้ในช่วงเวลาของจุดตรวจสอบนั้น โดยไม่ต้องฟื้นสภาพ ณ จุดเริ่มต้นของทรานแซกชัน แต่อาจทำให้เกิดปัญหาคอขวดขึ้น (bottle neck) ขึ้นได้ คือ อาจใช้เวลาในการเคลื่อนย้ายข้อมูลจากหน่วยความจำหลักลงฐานข้อมูลในดิสก์มากกว่าปกติ แต่ถ้าจุดตรวจสอบห่าง (checkpoint time หรือ Tc) อาจทำให้เปลืองเนื้อที่ในหน่วยความจำหลักเพราะต้องใช้ที่พักข้อมูลชั่วคราว (buffer) ขนาดใหญ่ในการเก็บข้อมูลจำนวนมากที่ยังไม่ได้เคลื่อนย้ายลงฐานข้อมูลในดิสก์

การฟื้นสภาพไม่ว่าจะเกิดจากความขัดข้องประการใดๆ จะใช้ไฟล์ประวัติ (log file) เข้ามาช่วยในการนำข้อมูลกลับคืนมาให้อยู่ในสภาวะที่ถูกต้อง

เรื่องที่ 11.1.2 การกู้ข้อมูลจากความขัดข้องที่เกิดจากระบบคอมพิวเตอร์

ความขัดข้องของระบบมีผลกระทบโดยตรงต่อข้อมูล และค่าตัวแปรต่าง ๆ ในหน่วยความจำหลัก (main memory) โดยทั่วไปข้อมูลต่าง ๆ ที่ถูกอ่านหรือเขียน (read/write) จะไม่กระทำบนฐานข้อมูลโดยตรง แต่จะอ่านหรือเขียนข้อมูลไปพักในที่พักข้อมูลชั่วคราว (buffer) ซึ่งเป็นส่วนหนึ่งของหน่วยความจำหลัก ซึ่งจะช่วยลดการติดต่อกับส่วนรับเข้า/ส่งออก (input/output) ทำให้การทำงานเร็วขึ้น เมื่อเนื้อที่ของที่พักข้อมูลชั่วคราวเต็มหรือมีงานอื่นๆ ที่ต้องการใช้เนื้อที่ส่วนนี้ ข้อมูลในที่พักข้อมูลชั่วคราวก็จะถูกเคลื่อนย้ายไปเก็บไว้ในหน่วยความจำสำรอง คือ ฐานข้อมูลในดิสก์ การเกิดความขัดข้องจะทำให้ข้อมูลที่อยู่ในที่พักข้อมูลรับเข้า/ส่งออก (I/O buffer) สูญหาย แต่จะไม่มีผลกระทบกับข้อมูลในหน่วยความจำสำรอง คือข้อมูลในฐานข้อมูลจะไม่สูญหายไป แต่ข้อมูลอาจไม่ถูกต้องสมบูรณ์ เนื่องจากระหว่างที่ทรานแซกชันทำงานอยู่นั้น อาจมีข้อมูลบางส่วนถูกบันทึกในฐานข้อมูลแล้ว แต่บางส่วนยังคงอยู่ในที่พักข้อมูลชั่วคราวและเมื่อเกิดความขัดข้องขึ้นก็จะทำให้ข้อมูลที่อยู่ในที่พักข้อมูลชั่วคราวเกิดการสูยหายขึ้น ทำให้ข้อมูลในฐานข้อมูลอาจไม่ถูกต้องได้ วิธีการฟื้นสภาพที่เกิดจากความขัดข้องของระบบคอมพิวเตอร์ อาจใช้ ยกเลิก (UNDO) ทำซ้ำ (REDO) และ จุดตรวจสอบ (CHECKPOINT) ซึ่งจะได้กล่าวต่อไป

1.ตัวอย่างความขัดข้องของระบบ

พิจารณาตัวอย่าง *การโอนเงินจากบัญชี ก. ไปบัญชี ข.* สมมติ บัญชี ก. มียอดเงิน 1,000 บาท บัญชี ข. มียอดเงิน 800 บาท และต้องการโอนเงิน 300 บาท จากบัญชี ก. ไปยังบัญชี ข.

ขั้นตอนการทำงานเป็นดังนี้

1. อ่านยอดเงินบัญชี ก. จากฐานข้อมูลในดิสก์ไปเก็บไว้ในที่พักข้อมูลชั่วคราว (buffer) ในหน่วยความจำหลัก

2. คำนวณยอดเงินบัญชี ก. โดยหักเงินโอน 300 บาท ดังนั้นยอดเงินในบัญชี ก. ใหม่ คือ 700 บาท ซึ่งทำให้ค่าของข้อมูลในที่พักข้อมูลชั่วคราว (buffer) ในหน่วยความจำหลักเปลี่ยนแปลงไป

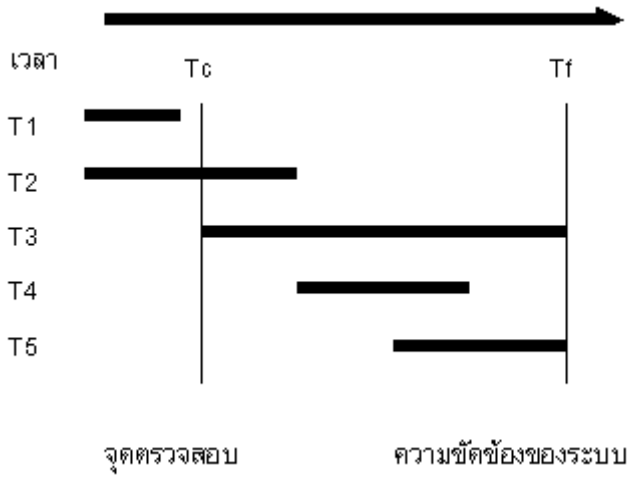
3. อ่านยอดเงินจากบัญชี ข. จากฐานข้อมูลในดิสก์ มาเก็บไว้ในที่พักข้อมูลชั่วคราวในหน่วยความจำหลัก

4. คำนวณยอดเงินบัญชี ข. ได้ยอดเงินใหม่ คือ 1100 บาท ซึ่งจะเก็บไว้ในที่พักข้อมูลชั่วคราวในหน่วยความจำหลัก

5. เมื่อที่พักข้อมูลชั่วคราวเต็ม จะมีการเคลื่อนย้ายข้อมูลจากที่พักข้อมูลชั่วคราวไปเก็บในฐานข้อมูลในดิสก์ ถ้ายอดเงินบัญชี ก. ถูกบันทึกลงฐานข้อมูลในดิสก์ก่อน แต่ยอดเงินบัญชี ข. ยังไม่ถูกบันทึกลงฐานข้อมูลในดิสก์และเกิดปัญหาขัดข้องระหว่างนี้ก็จะทำให้ข้อมูลในที่พักข้อมูลชั่วคราวสูญหายไป และทำให้ยอดเงินบัญชี ข. ในฐานข้อมูลในดิสก์ไม่ถูกต้อง

2.ขั้นตอนการฟื้นสภาพ

จากตัวอย่างข้างต้นจะเห็นว่า แม้ว่าทรานแซกชันจะทำงานจนเสร็จสิ้นถึงคอมมิต (COMMIT) ก่อนเกิดปัญหาขัดข้องแล้วก็ตาม แต่ค่ายอดเงินของบัญชี ข. ยังอยู่ในที่พักข้อมูลชั่วคราวเป็น 1100 บาท ยังมิทันได้เก็บบันทึกลงฐานข้อมูลจริงเลย คือค่าในฐานข้อมูลยังเป็น 800 บาท ซึ่งไม่ถูกต้องเพราะค่ายอดเงินในบัญชี ก. ของฐานข้อมูลเปลี่ยนแปลงเป็นค่าใหม่แล้ว แต่ค่ายอดเงินในบัญชี ข. ยังคงเป็นค่าเดิม ทำให้เราต้องทำการฟื้นสภาพ แต่มีคำถามที่เกิดขึ้นคือเราต้องทำการฟื้นสภาพย้อนหลังถึงเมื่อไร ซึ่งโดยทั่วไปแล้วเราไม่มีโอกาสทราบได้เลยว่าเมื่อระบบเกิดข้อขัดข้องขึ้น แล้วข้อมูลส่วนไหนที่เขียนลงฐานข้อมูลไปบ้าง ดังนั้นโดยหลักการแล้วเพื่อความมั่นใจว่าข้อมูลในฐานข้อมูลถูกต้องทั้งหมด จึงต้องทำซ้ำ (REDO) ข้อมูลตั้งแต่ต้นของไฟล์ประวัติ (log file) ซึ่งกรณีนี้เป็นวิธีที่ไม่มีประสิทธิภาพ เพราะจะต้องทำซ้ำใหม่ตั้งแต่ต้น ดังนั้นจึงมี***จุดตรวจสอบ (CHECK POINT)*** ขึ้นเพื่อให้ทราบว่าจะทำการฟื้นสภาพ (recover) ย้อนหลังถึงจุดไหนนั่นเอง ซึ่งประโยชน์ของจุดตรวจสอบ คือทำให้การฟื้นสภาพทำได้เร็วขึ้น



Tc คือ จุดตรวจสอบครั้งสุดท้าย ณ เวลา Tc

Tf คือ ความขัดข้องของระบบ ณ เวลา Tf

ภาพที่ 11.1 แสดงลักษณะของทรานแซกชัน

จากภาพที่ 11.1 นั้น ลักษณะของทรานแซกชันต่าง ๆ ที่เกิดขึ้นจัดได้ 5 ลักษณะ เมื่อกำหนดระบบเกิดความขัดข้อง ณ เวลา Tf และจุดตรวจสอบครั้งสุดท้าย กระทำขึ้น ณ เวลา Tc

ทรานแซกชัน T1 กระทำเสร็จสมบูรณ์ก่อนเวลา Tc
ทรานแซกชัน T2 เกิดขึ้นก่อนเวลา Tc และเสร็จสิ้นก่อน Tf
ทรานแซกชัน T3 เกิดขึ้นก่อนเวลา Tc และณ เวลา Tf ยังทำงานไม่เสร็จ
ทรานแซกชัน T4 เกิดขึ้นหลังเวลา Tc และเสร็จสิ้นก่อน Tf
ทรานแซกชัน T5 เกิดขึ้นหลังเวลา Tc และ ณ เวลา Tf ยังทำงานไม่เสร็จ

จะเห็นได้ว่า เมื่อเกิดความขัดข้องขึ้น ณ เวลา Tf ใดๆนั้น

- ทรานแซกชัน T1 เท่านั้น ซึ่งเสร็จสิ้นสมบูรณ์และปลอดภัย เพราะคอมมิต (COMMIT) แล้วก่อนจุดตรวจสอบครั้งสุดท้าย ณ เวลา Tc และข้อมูลได้ถูกเก็บไว้ในฐานข้อมูลจริงแล้ว
- ทรานแซกชัน T3,T5 ยังไม่ได้คอมมิตจึงต้องยกเลิก (UNDO) การทำงานทั้งหมด
- ทรานแซกชัน T2,T4 คอมมิตแล้ว ถึงแม้ว่าทรานแซกชัน T2 และ T4 จะเสร็จสิ้นสมบูรณ์แล้ว แต่ยังไม่ทันที่ทรานแซกชันทำงานเสร็จสิ้นก่อนจุดตรวจสอบเพื่อเคลื่อนย้ายข้อมูลไปสู่ฐานข้อมูลจริง ก็เกิดปัญหาขัดข้องเสียก่อน ดังนั้นจึงต้องทำซ้ำ (REDO) ทรานแซกชันนั้นใหม่อีกครั้ง

ระบบจัดการฐานข้อมูลจะมีส่วนจัดการการฟื้นสภาพ (Recovery Manager; RM) เป็นตัวจัดการการฟื้นสภาพทั้งหมด โดยเริ่มต้นพิจารณาว่าทรานแซกชันใดต้องยกเลิก (UNDO) หรือ ทำซ้ำ (REDO) ซึ่งจะอาศัยข้อมูลจากไฟล์ประวัติ (log file) ดังขั้นตอนต่อไปนี้

ขั้นที่ 1 เริ่มต้นด้วยการสร้างลิสต์ของทรานแซกชัน 2 กลุ่ม คือ

- ลิสต์ทรานแซกชันของกลุ่มที่ต้องยกเลิก (UNDO-LIST)
- ลิสต์ทรานแซกชันของกลุ่มที่ต้องทำซ้ำ (REDO-LIST)

ณ จุดตรวจสอบครั้งสุดท้าย ณ เวลาที่ Tc พิจารณาว่ามีทรานแซกชันใดทำงานผ่านจุดตรวจสอบนั้นให้นำทรานแซกชันทั้งหมดไปอยู่ในกลุ่ม UNDO-LIST ในขั้นตอนนี้จะได้ UNDO-LIST ประกอบด้วยทรานแซกชัน T2 และ T3 และในเบื้องต้นกำหนดให้ REDO-LIST เป็นลิสต์ว่าง

ขั้นที่ 2 พิจารณาข้อมูลในไฟล์ประวัติ (log file) โดยเริ่มตรวจสอบ*หลังจาก*จุดตรวจสอบครั้งสุดท้าย ณ เวลา Tc

- ถ้าพบทรานแซกชันใดเริ่มต้นทำงาน (start) ให้เพิ่มทรานแซกชันนั้นเข้าไปใน UNDO-LIST

(ในที่นี้จะได้ UNDO-LIST ประกอบด้วยทรานแซกชัน T2,T3,T4 และ T5)

- ถ้าพบทรานแซกชันใดคอมมิต (COMMIT) ให้ย้าย ทรานแซกชันนั้นจาก UNDO-LIST ไปยัง REDO-LIST

(ในที่นี้ REDO-LIST ประกอบด้วยทรานแซกชัน T2 และ T4 และ UNDO-LIST ประกอบด้วยทรานแซกชัน T3 และ T5

โดยสรุปจะได้ลิสต์ทรานแซกชันทั้ง 2 กลุ่มมาดำเนินการ UNDO หรือ REDO ต่อไป โดยส่วนจัดการการฟื้นสภาพจะพิจารณาข้อมูลในไฟล์ประวัติ กรณีที่ยกเลิก (UNDO) ก็จะดึงข้อมูลเดิม (old value) ก่อนการเปลี่ยนแปลงกลับมา และเริ่มทำงานตามทรานแซกชันนั้นอีกครั้ง สำหรับกรณีทำซ้ำ (REDO) ก็จะไปดึงข้อมูลหลังการเปลี่ยนแปลง (new value) มาแทนที่ จนกระทั่งทรานแซกชันนั้นสมบูรณ์

มีข้อสังเกตคือทรานแซกชัน T2 จะทำซ้ำ (REDO) เฉพาะการเปลี่ยนแปลงที่เกิดขึ้นหลังจุดตรวจสอบ ณ เวลา Tc เท่านั้น เนื่องจากการเปลี่ยนแปลงของทรานแซกชันก่อนจุดตรวจสอบ ณ เวลา Tc ได้ถูกบันทึกในฐานข้อมูลจริงแล้ว และในทำนองเดียวกันทรานแซกชัน T3 ก็จะยกเลิก (UNDO) เฉพาะการเปลี่ยนแปลงที่เกิดขึ้นก่อนจุดตรวจสอบ ณ เวลา Tc เท่านั้น เนื่องจากการเปลี่ยนแปลงที่เกิดขึ้นหลังจากจุดตรวจสอบ ณ เวลา Tc กระทำอยู่ในที่พักข้อมูลชั่วคราว (buffer) เท่านั้นยังไม่ได้บันทึกลงฐานข้อมูลจริง ส่วนการเปลี่ยนแปลงที่เกิดก่อนจุดตรวจสอบ ณ เวลา Tc ได้ถูกเก็บไว้ในฐานข้อมูลแล้ว ดังนั้นจึงจำเป็นต้องยกเลิกการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นก่อนจุดตรวจสอบ ณ เวลา Tc เพื่อให้ข้อมูลในฐานข้อมูลกลับสู่สภาพเดิม

เรื่องที่ 11.1.3 การกู้ข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของรายการและสื่อบันทึกข้อมูล

ปัญหาของความขัดข้องของระบบจะทำให้เกิดสถานะของทรานแซกชันได้ 2 สถานะคือ ทรานแซกชันยังทำไม่เสร็จสมบูรณ์ หรือ ทรานแซกชันทำเสร็จแล้วแต่ยังไม่ได้บันทึกลงฐานข้อมูล การกู้ข้อมูลหรือการฟื้นสภาพไม่ว่าจะเกิดความขัดข้องประการใดๆจะใช้วิธีการฟื้นสภาพแบบล็อกเบส(logbase recovery) โดยไฟล์ประวัติ(log file) จะเข้ามาช่วยในการนำข้อมูลกลับคืนมาให้อยู่ในภาวะที่ถูกต้อง

1.ลักษณะความขัดข้องที่เกิดจากข้อผิดพลาดของรายการ

ความขัดข้องที่เกิดจากข้อผิดพลาดของรายการ ความขัดข้องของทรานแซกชันมี 2 ลักษณะ คือ

1.1 ความขัดข้องที่สามารถป้องกันได้โดยการตรวจสอบภายในโปรแกรม เช่น ทรานแซกชันการโอนเงิน ถ้ายอดเงินโอนมีค่ามากกว่ายอดเงินที่มีอยู่ในบัญชี เมื่อโปรแกรมตรวจพบจะแสดงข้อความผิดพลาด อาทิ “ยอดเงินไม่เพียงพอ (insufficient fund)” ซึ่งลักษณะนี้ต้องทำการยกเลิกการเปลี่ยนแปลงต่าง ๆ โดยปรับข้อมูลให้เป็นค่าเดิมเมื่อเริ่มต้น หรือ ผู้เขียนโปรแกรมกำหนดขอบเขตของทรานแซกชันไม่ถูกต้อง ดังตัวอย่างที่ ความขัดข้องดังกล่าวสามารถป้องกันได้โดยผู้เขียนโปรแกรมต้องตรวจสอบความถูกต้องภายในโปรแกรมให้ครอบคลุมทุกกรณีที่มีโอกาสจะละเมิดกฎควบคุมความถูกต้อง และต้องกำหนดขอบเขตของทรานแซกชันให้ถูกต้อง ตัวอย่างที่ 2 ทรานแซกชันการโอนเงินจากบัญชี A ไปยังบัญชี B โดยกฎควบคุมความถูกต้อง (integrity rule) ในการโอนเงิน คือ ยอดเงินต้องไม่หายไปจากระบบ โดยกำหนดขอบเขตของทรานแซกชันดังนี้.

Begin Tx

```
    UPDATE account  SET AMOUNT = AMOUNT – x
    WHERE  ACCT# = 'ACCT_A';
End Tx

Begin Tx
    UPDATE account  SET AMOUNT = AMOUNT + x
    WHERE ACCT# = 'ACCT_B';
End Tx
```

จะเห็นว่าทรานแซกชันถอนเงิน กับ ทรานแซกชันฝากเงิน เป็นคนละทรานแซกชันกัน ซึ่งเป็นการกำหนดขอบเขตของทรานแซกชันที่***ผิด*** ซึ่งที่ถูกต้องแล้วควรกำหนดทรานแซกชันถอน และ ทรานแซกชันฝาก อยู่รวมกันเป็นทรานแซกชันเดียว เพราะถ้าอยู่แยกกัน ขณะที่ทรานแซกชันถอนเงินจากบัญชี A ทำเสร็จเรียบร้อยแล้ว แต่ถ้าทรานแซกชันฝากเงินเข้าบัญชี B ผิดพลาด จะทำให้ไปยกเลิกส่วนทรานแซกชันถอนเงินไม่ได้ เนื่องจากการทำงานของทรานแซกชันชุดนี้ทำแยกกัน ดังนั้นจึงมีส่วนของทรานแซกชันถอนเงินสำเร็จเกิดขึ้น และมีส่วนของทรานแซกชันฝากเงินผิดพลาด (failed) เกิดขึ้น ทรานแซกชันทั้งชุดนี้ไม่ได้ถูกกระทำเป็นหน่วยเดียว ลักษณะนี้ถือว่าทรานแซกชันนี้ไม่มีคุณสมบัติอะตอมมิกซิตี (atomicity) เพราะว่าผู้เขียนโปรแกรมกำหนดขอบเขตของทรานแซกชันไม่ถูกต้อง นอกจากนี้ผลที่ตามมาทำให้ข้อมูลในฐานข้อมูลไม่มีความถูกต้องเกิดขึ้น ซึ่งทำให้ทรานแซกชันขาดคุณสมบัติของความถูกต้อง (correctness) ซึ่งคุณสมบัติความถูกต้อง (correctness) เป็นหน้าที่ของโปรแกรมเมอร์ที่ต้องรับผิดชอบกำหนดขอบเขตของ ทรานแซกชันให้ถูกต้องโดยทำให้เป็นอะตอมมิกดังนี้

```
Begin Tx
    UPDATE account  SET AMOUNT = AMOUNT – x
            WHERE  ACCT# = 'ACCT_A';
    UPDATE account  SET AMOUNT = AMOUNT + x
            WHERE ACCT# = 'ACCT_B';
End Tx
```

1.2 ความขัดข้องที่ไม่สามารถป้องกันได้โดยการตรวจสอบภายในโปรแกรม เช่น การล้น (overflow) เกินพื้นที่ในการทำงาน ซึ่งทำให้มีเนื้อที่ไม่เพียงพอต่อการปฏิบัติการของทรานแซกชัน ซึ่งอาจก่อให้เกิดความผิดพลาดแก่ทรานแซกชันได้

2.การกู้ข้อมูลจากความขัดข้องที่เกิดจากรายการ

การกู้ข้อมูลหรือการฟื้นสภาพจากความขัดข้องของทรานแซกชัน(transaction failures) จะพิจารณาในกรณีที่โปรแกรมไม่สามารถจัดการหรือควบคุมได้ เช่น การล้น (overflow) เกินพื้นที่ในการทำงาน ซึ่งทำให้

มีเนื้อที่ไม่เพียงพอต่อการทำงานของทรานแซกชัน วิธีการฟื้นสภาพในกรณีนี้ คือ ยกเลิก (UNDO) การกระทำต่าง ๆ ที่เกิดขึ้นกับทรานแซกชันทั้งหมด และปรับค่าข้อมูลต่าง ๆ ในฐานข้อมูลให้กลับคืนมาเหมือนว่าทรานแซกชันนั้นยังไม่เริ่มต้นทำงาน ระบบจัดการฐานข้อมูลจะมีส่วนจัดการการฟื้นสภาพ (Recovery Manager; RM) ทำหน้าที่ดูแลและจัดการฟื้นสภาพ โดยจะตรวจสอบข้อมูลและดำเนินการย้อนหลัง (backward trace) ตามรายละเอียดที่บันทึกในไฟล์ประวัติ (log file) ซึ่งบันทึกค่าตัวแปรต่าง ๆ ทั้งก่อน และหลังการเปลี่ยนแปลงไว้ ทำให้ส่วนจัดการการฟื้นสภาพ สามารถปรับค่าข้อมูลต่าง ๆ กลับคืนมาเป็นค่าเดิมเมื่อเริ่มต้นทรานแซกชันได้

3.การกู้ข้อมูลจากความขัดข้องที่เกิดจากสื่อบันทึกข้อมูล

การฟื้นสภาพจากความขัดข้องของสื่อข้อมูล (Media Failures) ความขัดข้องที่เกิดกับสื่อข้อมูล เช่น บางส่วนของหน่วยความจำสำรองถูกทำลาย ซึ่งอาจเกิดจากหัวอ่านกระทบกับแผ่นดิสก์ ซึ่งอาจมีผลไปทำลายข้อมูลในสื่อข้อมูลได้ การฟื้นสภาพในกรณีนี้ คือ

- จำเป็นต้องอาศัยข้อมูลสำรองครั้งล่าสุด ซึ่งควรจะมีการทำสำรองข้อมูลไว้อย่างสม่ำเสมอ
- ใช้ข้อมูลในไฟล์ประวัติทำการทำซ้ำ (REDO) สำหรับทรานแซกชันที่ได้กระทำเสร็จสมบูรณ์แล้ว คือทรานแซกชันนั้นคอมมิต (COMMIT) แล้ว

สำหรับความขัดข้องในกรณีนี้ ทำให้เห็นว่าการสำรองข้อมูลมีความจำเป็นอย่างมาก หากมีการสำรองข้อมูลบ่อยๆ ก็ยิ่งทำให้การฟื้นสภาพรวดเร็วมากขึ้น เพราะข้อมูลสำรองมีความทันสมัยใกล้เคียงข้อมูลที่ถูกทำลาย อย่างไรก็ตามการสำรองข้อมูล มักต้องใช้เวลานาน บางครั้งอาจใช้เวลาเป็นวัน ๆ หากเป็นระบบใหญ่ ๆ ทำให้ไม่สามารถใช้ระบบในงานด้านอื่นๆ ได้ จึงควรกำหนดช่วงเวลาการสำรองข้อมูลให้เหมาะสม เช่นหากมีการเปลี่ยนแปลงโครงสร้างข้อมูลระบบใหม่ก็ควรจะทำการสำรองข้อมูลทันที เป็นต้น

ตอนที่ 11.2 การควบคุมภาวะความพร้อมกัน (Concurrency Control)

หัวเรื่อง

เรื่องที่ 11. 2.1 แนวคิดเกี่ยวกับการใช้ข้อมูลพร้อมกัน

เรื่องที่ 11.2.2 ประเภทและระดับของการล็อค

เรื่องที่ 11.2.3 ปัญหาและวธีการแก้ไข deadlock

เรื่องที่ 11.2.4 วิธีการควบคุมภาวะความพร้อมกัน

แนวคิด

1. เพื่อให้สามารถใช้ข้อมูลพร้อมกันได้อย่างถูกต้อง และมีประสิทธิภาพ ผู้ใช้จำเป็นต้องเข้าใจแนวคิดในการใช้ข้อมูลพร้อมกันและผู้ใช้ควรที่จะทราบวิธีการควบคุมและป้องกันปัญหาต่าง ๆ ที่อาจเกิดขึ้นได้
2. การควบคุมการเกิดภาวะพร้อมกันวิธีหนึ่ง คือการล็อก ซึ่งเป็นวิธีการเรียกใช้หรือการปรับปรุงแก้ไขข้อมูลจากตารางหนึ่งๆในขณะที่มีคนอื่นใช้อยู่ ในการล็อกข้อมูลจะมีประเภทและระดับการล็อกได้หลายแบบ
3. การล็อกเป็นการป้องกันความผิดพลาดของข้อมูลที่อาจจะเกิดขึ้นจากการที่ผู้ใช้ข้อมูลที่ยังทำงานไม่เสร็จและมีผู้อื่นๆเรียกข้อมูลที่ไม่ถูกต้องนี้ไปใช้ ปัญหาของการล็อกข้อมูล ได้แก่ ปัญหา Deadlock ซึ่งเป็นเหตุการณ์ที่ทรานแซกชัน 2 รายการได้ทำการล็อกข้อมูลเดียวกันและต่างก็รอซึ่งกันและกัน ในการแก้Deadlock มี 2 แนวทางคือ การป้องกันการเกิดDeadlock และการแก้ปัญหาด้วย DBMS DBMSสามารถตรวจสอบและแก้ไขการเกิด Deadlockได้
4. ในการควบคุมภาวะพร้อมกัน ระบบจัดการฐานข้อมูลจะต้องมีกลไกสำหรับจัดเรียงลำดับก่อนหลังการทำงานของแต่ละทรานแซกชัน (scheduler) ด้วยวิธีการ ล็อกกิง (locking), ไทม์แสตมป์ (time stamp) และออปทิมิสทิก (optimistic)

วัตถุประสงค์

หลังจากศึกษาตอนที่ 11.2 แล้ว นักศึกษาสามารถ

1. อธิบายแนวคิดการใช้ข้อมูลพร้อมกันได้
2. อธิบายวิธีการควบคุมการใช้ข้อมูลพร้อมกันได้
3. อธิบายปัญหาที่เกิดจากการเข้าถึงข้อมูลพร้อมกันได้
4. อธิบายวิธีการแก้ไขการเข้าถึงข้อมูลพร้อมกันได้

เรื่องที่ 11.2.1 แนวคิดเกี่ยวกับการใช้ข้อมูลพร้อมกัน

คุณสมบัติที่สำคัญอย่างหนึ่งของฐานข้อมูลก็คือ การที่ผู้ใช้จากส่วนต่างๆ สามารถจะเรียกใช้ข้อมูลในฐานข้อมูลได้พร้อมๆ กัน ในกรณีที่ฐานข้อมูลที่จัดทำขึ้นในเครื่องไมโครคอมพิวเตอร์เพื่อใช้งานส่วนบุคคลนั้น

ก็ย่อมไม่จำเป็นต้องคำนึงถึงการควบคุมการทำงานที่เกิดจากผู้ใช้หลายๆ คนที่ต้องการใช้งานฐานข้อมูลนั้นในขณะเดียวกัน เนื่องจากการใช้งานฐานข้อมูลอาจจะเป็นการใช้งานโดยผู้ใช้คนใดคนหนึ่ง ณ เวลาใดเวลาหนึ่งเท่านั้น แต่สำหรับฐานข้อมูลที่ใช้งานในระดับองค์กรที่เป็นฐานข้อมูลในระบบที่ซับซ้อนมากขึ้นนั้น ระบบจัดการฐานข้อมูลจะต้องมีกลไกในการจัดการเพื่อให้ผู้ใช้จากหน่วยงานต่างๆ สามารถเรียกใช้ข้อมูลในฐานข้อมูลเพื่อทำงานได้พร้อมๆ กัน โดยที่ผลลัพธ์ที่ได้จากการทำงานจะต้องถูกต้องเสมอ

1. **ความหมายของภาวะพร้อมกัน**

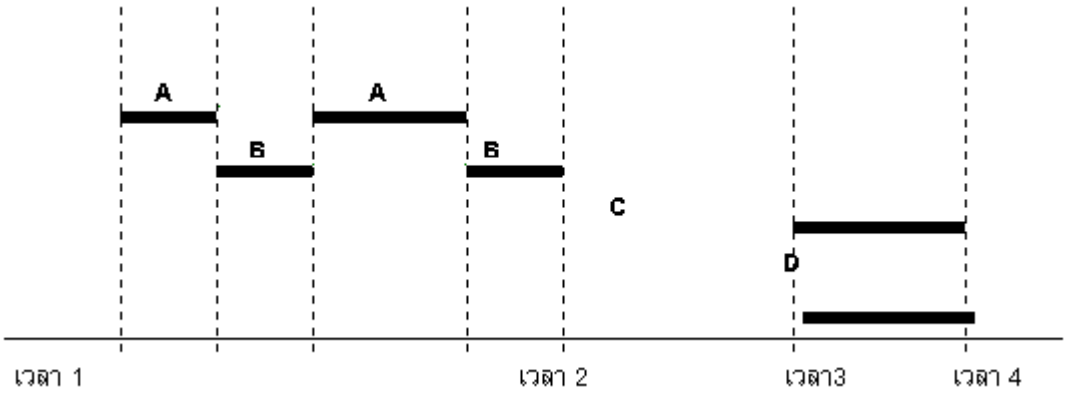
คำว่า “ภาวะพร้อมกัน (concurrency)” หมายความว่า การที่มีทรานแซกชันหลายๆ ทรานแซกชันต้องการเรียกใช้ข้อมูลเดียวกันในเวลาเดียวกันจากฐานข้อมูลเพื่อทำงานของแต่ละทรานแซกชัน ภาวะการทำงานพร้อมกันเกิดจากระบบการทำงานได้ 2 ระบบ

การทำงานในระบบหลายโปรแกรม

การทำงานในระบบการประมวลผลในเวลาเดียวกัน

1.1 การทำงานในระบบหลายโปรแกรม หรือการทำงานแบบมัลติโปรแกรมมิง (multiprogramming) เป็นการทำงานของระบบคอมพิวเตอร์ที่ออกแบบเพื่อให้หน่วยประมวลผลผลกลางหรือซีพียู (Central Processing Unit; CPU) ทำงานหลายๆ งานในขณะเดียวกันได้ ทั้งนี้ด้วยเหตุผลเพื่อให้การใช้งานซีพียูเป็นไปอย่างคุ้มค่าโดยไม่ต้องอยู่ว่าง (idle) เนื่องจากซีพียูจะมีความเร็วในการทำงานสูงกว่าอุปกรณ์อื่นๆ ถ้าหากไม่มีระบบการทำงานแบบมัลติโปรแกรมมิง ซีพียูต้องทำงานใดงานหนึ่งจนเสร็จจึงจะสามารถทำงานที่ 2, 3ต่อไปได้ ซึ่งหมายความขณะที่งานนั้นกำลังใช้เครื่องพิมพ์หรืออุปกรณ์อื่นๆ ที่ไม่ใช่ซีพียู ซีพียูก็ต้องเสียเวลารอ ดังนั้นจึงเกิดแนวคิดการประมวลผลแบบให้หลายโปรแกรมทำงานพร้อมๆ กัน โดยมีการสลับช่วงการทำงานระหว่างโปรแกรมเพื่อสลับให้ซีพียูไปทำงานของทรานแซกชันอื่นๆ ซึ่งแต่ละทรานแซกชันที่ต้องการให้ซีพียูทำงานนั้นอาจจะเป็นโปรแกรมเดียวกันหรือต่างโปรแกรมก็ได้ โดยใช้หลักการอินเทอลีฟ มาใช้ในการควบคุมภาวะพร้อมกัน

อินเทอลีฟ (interleaved) คือ การที่ทรานแซกชันมากกว่าหนึ่งทรานแซกชัน มีการสลับการทำงานกันในขณะใดขณะหนึ่ง โดยที่ระบบจัดการฐานข้อมูลจะต้องควบคุมภาวะพร้อมกัน (concurrency control) เพื่อให้แต่ละทรานแซกชันมีการทำงานสลับกันไปมา ทั้งนี้ผลลัพธ์ที่ได้จะต้องมีความถูกต้องเสมือนว่าแต่ละทรานแซกชันทำงานเรียงลำดับทีละทรานแซกชันจนสิ้นสุดงานของแต่ละทรานแซกชันนั้น ตัวอย่างเช่น ภาพที่ 11.2 มีทรานแซกชัน A และ B ทำงานพร้อมกันในช่วงเวลา time1 และ time2 โดยระบบจัดการฐานข้อมูลต้องควบคุมภาวะพร้อมกัน โดยมีการสลับการทำงานระหว่างทรานแซกชัน A และ B



ภาพที่ 11.2 การทำงานแบบอินเทอลีฟ เทียบกับ การประมวลผลในเวลาเดียวกัน

1.2 การประมวลผลในเวลาเดียวกัน (simultaneous processing) เป็นการทำงานในระบบคอมพิวเตอร์ที่มีซีพียูมากกว่า 1 ซีพียู เพื่อรองรับการทำงานของโปรแกรมใดโปรแกรมหนึ่งได้โดยไม่ต้องสลับทำงานระหว่างหลายทรานแซกชัน ดังนั้นซีพียูแต่ละตัวก็จะทำงานของโปรแกรมใดโปรแกรมหนึ่งแยกกันไปแต่ละซีพียูจนเสร็จงาน เช่น ภาพที่ 11.2 ทรานแซกชัน C และ D ทำงานในช่วงเวลา3 และ เวลา4 โดยแต่ละทรานแซกชันมีซีพียูประมวลผลแยกไปคนละซีพียู

ในการกล่าวถึงการควบคุมภาวะการทำงานพร้อมกันในที่นี้จะเป็นในสภานการณ์แบบอินเทอลีฟ โดยมีซีพียูที่จะทำงานเพียงซีพียูเดียวเท่านั้น

2. ปัญหาที่ทำให้มีการควบคุมภาวะพร้อมกัน

การควบคุมภาวะพร้อมกันในการใช้งานฐานข้อมูลเป็นสิ่งสำคัญอย่างยิ่ง เพราะหากระบบจัดการฐานข้อมูลไม่มีกลไกดังกล่าวย่อมจะก่อให้เกิดปัญหาในการทำงานดังนี้

2.1 ปัญหาการสูญหายของข้อมูลที่มีการปรับปรุงแก้ไข (the lost update problem) เป็นปัญหาที่เกิดจากทรานแซกชันมากกว่าหนึ่งทรานแซกชันต้องการปรับปรุงแก้ไขข้อมูล เดียวกันในเวลาไล่เรี่ยกัน ทำให้ผลลัพธ์ที่ได้ไม่ถูกต้อง เพราะข้อมูลที่ถูกแก้ไขโดยทรานแซกชันก่อนหน้าหายไปหมด จะปรากฏแต่ผลลัพธ์ที่เกิดจากการปรับปรุงแก้ไขของทรานแซกชันหลังสุดเท่านั้น

ตัวอย่างเช่น การฝากและถอนเงิน สมมติว่า จำนวนเงินที่มีในบัญชี ณ ปัจจุบันเท่ากับ 350 บาท โดยมีทรานแซกชันที่ 1 และ มีทรานแซกชันที่ 2 ต้องการฝากและถอนเงิน ซึ่งถ้าหากมีการทำงานตามลำดับก่อน-หลัง โดยให้ทรานแซกชันที่ 1 ทำการฝากเงินเท่ากับ 350 เข้าไปในฐานข้อมูลเสียก่อน หลังจากนั้นทรานแซกชันที่ 2 จะทำการถอนเงินในฐานข้อมูลก็ย่อมไม่มีปัญหาอะไร ดังตารางที่ 11.1 แสดงทำงานของทรานแซกชันที่ 1 และ 2 ในภาวะพร้อมกัน แบบเรียงลำดับก่อน-หลัง

ตารางที่ 11.1 การทำงานของทรานแซกชันในภาวะพร้อมกันแต่เป็นการเรียงลำดับก่อน-หลัง

ทรานแซกชันที่ 1	เวลา	ทรานแซกชันที่ 2	ค่าของข้อมูลในฐานข้อมูล
อ่านจำนวนเงินในบัญชี	Time 1		350

ฝากเงิน 1000 (นั่นคือจำนวนเงินทั้งหมดในบัญชี 1000+350=1350)	Time 2		350
บันทึกจำนวนเงินทั้งหมด	Time 3		1350
	Time 4	อ่านจำนวนเงินในบัญชี	1350
	Time 5	ถอนเงินออกจากบัญชี 300 (นั่นคือ 1350-300=1050)	1350
	Time 6	บันทึกจำนวนเงินที่เหลือค้า	1050

ถ้าระบบจัดการฐานข้อมูลไม่มีกลไกในการควบคุมภาวะพร้อมกันจะเกิดปัญหาขึ้นได้ โดยหากทรานแซกชันที่ต้องการเรียกใช้ข้อมูลในการทำงานพร้อมกันโดยไม่เป็นลำดับก่อน-หลังทีละทรานแซกชัน ดังแสดงในตารางที่ 11.2 นั่นคือ สมมติว่าทรานแซกชันที่ 1 และทรานแซกชันที่ 2 เผอิญต้องการเรียกใช้ข้อมูลเดียวกันในเวลาไล่เลี่ยกัน ดังนี้

ตารางที่11.2 ปัญหาการสูญหายของข้อมูลที่มีการปรับปรุงแก้ไข

ทรานแซกชันที่ 1	**เวลา**	**ทรานแซกชันที่ 2**	**ค่าของข้อมูลในฐานข้อมูล**
อ่านจำนวนเงินในบัญชี	Time 1		350
	Time 2	อ่านจำนวนเงินในบัญชี	350
ฝากเงิน 1000(นั้นคือจำนวนเงินทั้งหมดในบัญชี 1000+350=1350)	Time 3		350
	Time 4	ถอนเงินออกจากบัญชี 300(นั่นคือ350-300=50)	350
บันทึกจำนวนเงินทั้งหมด	Time 5		1350
	Time 6	บันทึกจำนวนเงินทั้งหมด	50

- ช่วงเวลา Time1 ทรานแซกชันที่ 1 อ่านจำนวนเงินในบัญชีมีค่าเท่ากับ 350 บาท

- ช่วงเวลา Time 2 ทรานแซกชันที่ 2 ก็อ่านอ่านจำนวนเงินในบัญชีมีค่าเท่ากับ 350 บาท เช่นกัน

- ช่วงเวลา Time 3 ทรานแซกชันที่ 1 ทำการฝากเงิน 1000 อีก 1000 บาท นั่นคือ 1000+350=1350 แต่ยังไม่ได้มีการบันทึกในฐานข้อมูล จึงทำให้ค่าข้อมูลในฐานข้อมูลเท่ากับ 350 บาทอยู่เช่นเดิม

- ช่วงเวลา Time 4 ทรานแซกชันที่ 2 ถอนเงินออกจากบัญชี 300 บาท (นั่นคือ 350-300=50) แต่ยังไม่ได้มีการบันทึกในฐานข้อมูล จึงทำให้ค่าในฐานข้อมูลเท่ากับ 350 บาทยอยู่เช่นเดิม

- ช่วงเวลา Time 5 ทรานแซกชันที่ 1 บันทึกค่าที่คำนวณได้เท่ากับ 1350 ลงในฐานข้อมูล

- ช่วงเวลา Time 6 ทรานแซกชันที่ 2 บันทึกค่าที่คำนวณเท่ากับ 50 ลงในฐานข้อมูล ซึ่งบันทึกทับข้อมูลเดิมที่มีการปรับปรุงแก้ไขโดยทรานแซกชันแรก ทำให้ข้อมูลสูญหายไปจากฐานข้อมูล โดยปรากฏแต่ข้อมูลที่มีการปรับปรุงแก้ไขโดยทรานแซกชันที่ 2 เท่านั้น

2.2ปัญหาจากการเรียกใช้ข้อมูลชุดเดียวกันของทรานแซกชันที่ยังไม่คอมมิต (uncommitted dependency problem) เป็นปัญหาที่เกิดจากทรานแซกชันมากกว่าหนึ่งทรานแซกชันต้องการเรียกใช้ข้อมูลชุดเดียวกัน โดยทรานแซกชันที่ 1 ยังอยู่ระหว่างกลางในการทำงาน ขณะเดียวกันทรานแซกชันที่ 2 เรียกใช้ข้อมูลที่แก้ไขโดยทรานแซกชันที่ 1 หลังจากนั้นปรากฏว่าทรานแซกชันที่ 1 มีปัญหา จะต้องถูกยกเลิกและโรลแบ็ก (rollback) เพื่อเริ่มต้นทำงานใหม่ทั้งหมด ดังนั้นข้อมูลที่ทรานแซกชันที่ 2 เรียกไปใช้งานไปแล้วจึงเป็นข้อมูลไม่ถูกต้อง ทำให้ผลลัพธ์ที่ได้ไม่ถูกต้องด้วย เพราะมีการยกเลิกไปแล้วจากทรานแซกชันที่ 1

ตัวอย่างเช่น ถ้าทรานแซกชันที่ 1 ต้องการฝากเงินอีก 1000 หน่วย ขณะที่ทรานแซกชันที่ 2 ต้องการถอนเงินออกไป 300 บาท ดังนั้น ถ้าหากทรานแซกชันที่ 1 และ 2 ทำงานเรียงลำดับโดยทรานแซกชันที่ 2 รอให้ทรานแซกชันที่ 1 ทำงานเสร็จเสียก่อน แล้วทรานแซกชันที่ 2 จึงเรียกใช้ข้อมูลเดียวกัน ปัญหาก็จะไม่เกิด และผลลัพธ์ของข้อมูลที่บันทึกในฐานข้อมูลก็จะถูกต้องด้วย ดังตารางที่ 11.3

ตารางที่ 11.3 การเรียกใช้ข้อมูลภายหลังจากมีการโรลแบ็ก

ทรานแซกชันที่ 1	เวลา	ทรานแซกชันที่ 2	ค่าของข้อมูลในฐานข้อมูล
อ่านจำนวนเงินในบัญชี	Time 1		350
ฝากเงิน อีก 1000	Time 2		350
บันทึกจำนวนเงินทั้งหมด	Time 3		1350
มีปัญหาและการโรลแบ็ก	Time 4		350
	Time 5	อ่านจำนวนเงินในบัญชี	350
	Time 6	ถอนเงิน 30	350
	Time 7	บันทึกจำนวนเงินทั้งหมด	50

แต่ถ้าหากทรานแซกชันที่ 2 มีการเรียกใช้ข้อมูลที่ถูกปรับแก้โดยทรานแซกชันที่ 1 ซึ่งมีปัญหา และจะต้องถูกยกเลิกเพื่อเริ่มต้นทำงานนั้นใหม่ ก็จะทำให้ข้อมูลในฐานข้อมูลผิดพลาดไป ดังตารางที่ 11.4

ตารางที่ 11.4 การเรียกใช้ข้อมูลของทรานแซกชันที่ยังไม่คอมมิต

ทรานแซกชันที่ 1	เวลา	ทรานแซกชันที่ 2	ค่าของข้อมูลในฐานข้อมูล

อ่านจำนวนเงินในบัญชี	Time 1		350
ฝากเงินอีก 1000	Time 2		350
บันทึกจำนวนเงินทั้งหมด	Time 3		1350
	Time 4	อ่านจำนวนเงินในบัญชี	1350
	Time 5	ถอนเงิน 300 (นั่นคือ 1350-300)	1350
มีปัญหาและต้องโรลแบ็ก	Time 6		350
	Time 7	บันทึกจำนวนเงินทั้งหมด	1050

2.3 ปัญหาการเรียกใช้ข้อมูลที่ไม่สอดคล้องกัน (inconsistent retrieval problem) เป็นปัญหาที่เกิดจากทรานแซกชันมากกว่าหนึ่งทรานแซกชัน มีการใช้งานชุดข้อมูลเดียวกัน โดยทรานแซกชันหนึ่งใช้ข้อมูลนั้นเพื่อประมวลผลใดๆ ในขณะเดียวกันก็มีทรานแซกชันอื่นได้มีการปรับปรุงแก้ไขข้อมูลชุดเดียวกัน ทำให้ผลลัพธ์ของทรานแซกชันแรกไม่ถูกต้อง

ตัวอย่าง ตารางที่ 11.5 แสดงคำสั่งภาษาเอสคิวแอลเพื่อสั่งให้ทรานแซกชันที่ 1 และที่ 2 ทำงาน ดังนี้

- ทรานแซกชันที่ 1 ต้องการทราบปริมาณสินค้าทั้งหมดโดยนำค่าปริมาณสินค้า (QPROD) แต่ละชนิดจากตารางสินค้า (product) มารวมกัน

- ทรานแซกชันที่ 2 ต้องการปรับปรุงแก้ไขยอดปริมาณสินค้าของสินค้า 2 ชนิด คือ รหัสสินค้า A3 เพิ่มอีก 30 หน่วย และรหัสสินค้า A4 หักออก 30 หน่วย

ตารางที่ 11.5 การทำงานของทั้งสองทรานแซกชันด้วยภาษาเอสคิวแอล

ทรานแซกชันที่ 1	**ทรานแซกชันที่** 2
SELECT SUM (QPROD) FROM PRODUCT;	UPDATE PRODUCT SET QPROD = QPROD+30 WHERE PROD_ID = 'A3';
	UPDATE PRODUCT SET QPROD = QPROD-30 WHERE PROD_ID = 'A4';
	COMMIT;

ส่วนตารางที่ 11.6 แสดงข้อมูลในตารางสินค้าทั้งก่อนและหลังการทำงานของทรานแซกชันที่ 2 จากตารางที่ 11.5 ดังนี้

ตารางที่ 11.6 ข้อมูลในตารางสินค้า

	ปริมาณสินค้า (QPROD)	
รหัสสินค้า (PROD_ID)	ก่อนการทำงาน	หลังการทำงาน
A1	100	100
A2	120	120
A3	70	100 (เกิดจาก 70+30)
A4	35	5 (เกิดจาก 35-30)
A5	100	100
A6	30	30
Total	455	455

ถึงแม้ว่าผลลัพธ์ที่แสดงในตารางสินค้า (ตารางที่11.6) จะได้ผลลัพธ์ถูกต้อง แต่ถ้าหากดูการทำงานของแต่ละทรานแซกชัน ในแต่ละช่วงเวลาจะพบว่าผลรวมปริมาณสินค้าไม่ถูกต้อง ผลรวมที่ถูกต้องควรจะเป็น 455 แต่ผลรวมที่ได้จากการคำนวณ แสดงในตารางที่ 11.7 ได้เท่ากับ 485 ซึ่งยังไม่ถูกต้อง

ตารางที่ 11.7 ปัญหาการใช้ข้อมูลที่ไม่สอดคล้องกัน

ทรานแซกชันที่ 1	เวลา	ทรานแซกชันที่ 2	ค่าข้อมูลในฐานข้อมูล	ผลรวมสินค้า
อ่านปริมาณสินค้าของ PROD_ID = “A1”	Time1		100	100
อ่านปริมาณสินค้าของ PROD_ID = “A2”	Time2		120	220
	Time 3	อ่านปริมาณสินค้าของ PROD_ID = “A3”	70	
	Time 4	ปริมาณสินค้า = 70+30		
	Time 5	บันทึกปริมาณสินค้าของ PROD_ID = “A3”	100	
อ่านปริมาณสินค้าของ PROD_ID = “A3”	Time 6		100	320
อ่านปริมาณสินค้าของ PROD_ID = “A4”	Time 7		35	355
	Time 8	อ่านปริมาณสินค้าของ PROD_ID = “A4”	35	

	Time 9	ปริมาณสินค้า = 35-30		
	Time 10	บันทึกปริมาณสินค้าของ PROD_ID = “A4”	5	
	Time 11	COMMIT		
อ่านปริมาณสินค้าของ PROD_ID = “A5”	Time 12		100	455
อ่านปริมาณสินค้า ของ PROD_ID = “A6”	Time 13		30	485

จากปัญหาที่เกิดขึ้นทั้ง 3 สถานการณ์ของการทำงานในภาวะพร้อมกัน ระบบจัดการฐานข้อมูลจึงต้องมีกลไกในการควบคุมภาวะพร้อมกัน เพื่อไม่ให้เกิดปัญหาที่กล่าวข้างต้น

เรื่องที่ 11.2.2 ประเภทและระดับของการล็อค

1.ประเภทของการล็อก

ประเภทของการล็อก (Type of Locking) จะมีอยู่ 2 ประเภท คือ การล็อกแบบแบ่งส่วน (Shared Locks) และการล็อกแบบผูกขาด (Exclusive Locks)

1.1เอ็กคลูซีฟล็อก (exclusive lock) คือการกำหนดสถานะของข้อมูลให้เป็นล็อกโดยไม่ให้ทรานแซกชันอื่นใช้ข้อมูลที่ถูกล็อกนั้น เป็นการล็อกที่ผู้ใช้ล็อกแบบนี้จะเป็นผู้เดียวที่สามารถเปลี่ยนแปลงข้อมูลที่ล็อกไว้ได้ผู้เดียว ผู้ใช้คนอื่น ๆ จะไม่สามารถอ่านหรือแก้ไขระเบียนนั้นได้เลย จนกว่าจะมีการปล่อยล็อก การกำหนดให้ข้อมูลนั้นเป็นเอ็กคลูซีฟล็อก (exclusive lock) ได้จะเป็นกรณีที่ทรานแซกชันนั้นต้องการเปลี่ยนแปลงแก้ไขหรือบันทึกข้อมูล ซึ่งสามารถกำหนดในภาษาเอสคิวแอล ด้วยคำสั่ง “WRITE_LOCK” เพื่อไม่ให้ทรานแซกชันอื่นใช้ข้อมูลนั้นจนกว่าจะบันทึกเสร็จ ดังนั้นเมื่อทรานแซกชันใช้งานข้อมูลนั้นเสร็จแล้ว ควรจะใช้คำสั่ง “UNCLOCK” เพื่อให้ทรานแซกชันอื่นเรียกใช้ข้อมูลนั้นได้

1.2 แชร์ล็อก (share lock) คือ การกำหนดสถานะของข้อมูลให้ใช้งานร่วมกันกับทรานแซกชันอื่นได้ เช่น คำสั่ง “READ” จะมีการกำหนดสถานะของข้อมูลให้เป็น “share_lock” โดยอัตโนมัติ เนื่องจากการอ่านข้อมูลไม่มีผลทำให้เกิดการเปลี่ยนแปลงข้อมูล ซึ่งไม่ก่อให้เกิดปัญหาตามมาได้ จึงสามารถจะแชร์ล็อกเพื่อใช้งานร่วมกันระหว่างหลายๆ ทรานแซกชันได้

2.ข้อกำหนดของการล็อก

ข้อกำหนดของการล็อกทั้ง 2 ประเภท คือ

2.1ถ้าทรานแซกชัน ก. ทำการล็อกระเบียนแบบ S อยู่ดังนั้น

ถ้ามีทรานแซกชัน A ทำการล็อกระเบียนแบบแบ่งส่วน (Shared Bocks) จะสามารถล็อกระเบียนร่วมกันกับทรานแซกชัน A ได้เพื่อทำการอ่านข้อมูลนั้นอย่างเดียว

แต่ถ้ากับทรานแซกชัน B จะขอล็อกระเบียนนั้นแบบผูกขาด (Exclusive Locks) เพื่อปรับปรุงข้อมูลทรานแซกชัน B. จะทำการล็อกไม่ได้ จนกว่าทรานแซกชัน A. จะปล่อยล็อก

2.2 ถ้าทรานแซกชัน A ทำการล็อกระเบียนแบบผูกขาดอยู่ ดังนั้นทรานแซกชัน B จะไม่สามารถทำการล็อกระเบียนนั้นได้เลย ไม่ว่าจะเป็นการขอล็อกแบบแบ่งส่วน (Shared Locks) หรือแบบผูกขาดก็ตาม จะต้องรอให้ทรานแซกชันA ปล่อยล็อกอย่างเดียว

ตารางที่ 11.8 แสดงสถานะภาพของการล็อก

สถานะภาพของการล็อกของทรานแซกชัน B

สถานะภาพของการล็อกที่ **ทรานแซกชัน Aต้องการ**	Unlocked	Shared	Exclusive
Unlock		ได้	ได้
Shared	ได้	ได้	ไม่ได้
Exclusive	ได้	ไม่ได้	ไม่ได้

ตัวอย่าง การทำงานร่วมกันระหว่าง เอ็กคลูซีฟล็อก (exclusive lock) และแชร์ล็อก (sharelock) การล็อกนั้นจะสามารถใช้แก้ปัญหาของการสูญเสียผลของการแก้ไขได้ แต่ การล็อกก็สามารถนำไปสู่ปัญหาอื่นได้อีก ได้แก่ ล็อกค้าง (deadlock)

3.การแบ่งระดับการล็อก

การแบ่งระดับการล็อก (Locking Level) ในการควบคุมภาวะการเข้าถึงข้อมูลพร้อมกัน โดยการใช้การล็อกนี้จะสามารถเลือกรูปแบบของการล็อกข้อมูลได้หลายระดับ ตามลักษณะของงานนั้น ซึ่งสามารถแบ่งระดับของการล็อกนี้เป็นระดับต่าง ๆ ดังนี้

3.1การล็อกฐานข้อมูลทั้งฐาน (entire database) เป็นการล็อกทั้งฐานข้อมูลขณะที่ทรานแซกชันใดทรานแซกชันหนึ่งกำลังใช้งานมักจะเป็นกรณีที่มีการประมวลผลแบบแบตช์ (batch processing) แต่ไม่เหมาะสำหรับการประมวลผลแบบออนไลน์ ฐานข้อมูลทั้งหมดจะถูกล็อก โดยผู้บริหารฐานข้อมูล (DBA) ดังนั้นผู้ใช้คนอื่น ๆ จะไม่สามารถใช้ฐานข้อมูลนั้นได้เลย จนกว่าจะมีการปล่อยล็อก การล็อกแบบนี้จะล็อกในระหว่างที่มีการสำรอง (Backup) ฐานข้อมูลทั้งหมดนั้นขึ้นเทป

3.2การล็อกเฉพาะตารางใดตารางหนึ่งในฐานข้อมูล (a relation) การล็อกตารางเป็นการล็อกตารางข้อมูลที่ต้องการซึ่งระเบียนต่าง ๆที่อยู่ในตารางนั้นจะถูกล็อกด้วยโดยอัตโนมัติ ผู้ใช้คนอื่นจะไม่สามารถใช้ข้อมูลใดที่อยู่ในตารางที่ถูกล็อกนี้ได้เลย จนกว่าจะมีการปล่อยล็อก การล็อกตารางจะถูกทำในกรณีที่จะมีการปรับปรุงหรือแก้ไขข้อมูลทั้งหมดในตารางนั้น ๆ

3.3การล็อกเฉพาะแถวบางแถว (a tuple) ในฐานข้อมูลการล็อกแถวข้อมูลเป็นการล็อกเฉพาะระเบียนหรือแถวข้อมูลใด ๆ ในตาราง ซึ่งผู้ใช้คนอื่นจะไม่สามารถแก้ไขหรือปรับปรุงระเบียนที่ถูกล็อกนี้ได้จน

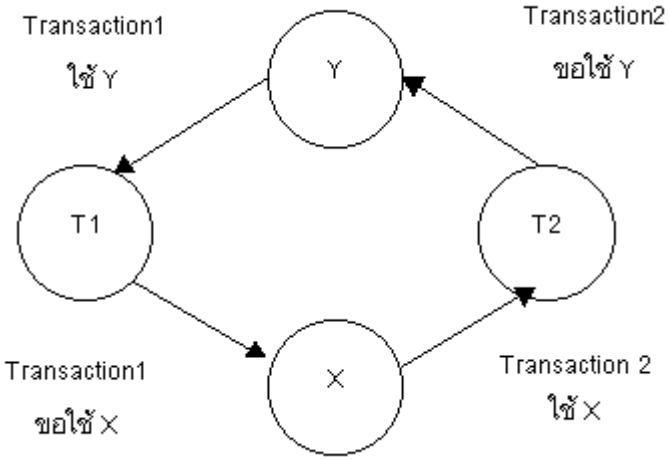
กว่าจะมีการปล่อนล็อก การล็อกระเบียนจะถูกใช้มากในการเขียนโปรแกรมที่ใช้ในการปรับปรุงระเบียนใดระเบียนหนึ่งในตาราง การล็อกระเบียนเป็นการห้ามไม่ให้ผู้อื่นเข้าถึงข้อมูลที่ถูกล็อกนี้ได้ ไม่ว่าจะเป็นการดูข้อมูลหรือการแก้ไขข้อมูล

3.4การล็อกเฉพาะบางฟิลด์ (field) หรือบางแอตทริบิวต์ (attribute) ในฐานข้อมูลเท่านั้น การล็อกเฉพาะบางฟิลด์เป็นการล็อกเฉพาะเขตข้อมูลใดๆในระเบียนที่ต้องการของตาราง การล็อกในระดับนี้จะใช้ในกรณีที่มีการแก้ไขข้อมูลในเขตข้อมูลที่ทำการล็อกนั้นบ่อย ๆ ตัวอย่างเช่น ตารางสินค้าคงคลัง มีเขตข้อมูลปริมาณสินค้าที่อยู่ในมือที่จะถูกเปลี่ยนแปลงค่าข้อมูลบ่อยมาก ในขณะที่ข้อมูลในเขตข้อมูลอื่น เช่น รหัสสินค้า ชื่อสินค้า แทบจะไม่มีการเปลี่ยนแปลงเลย ดังนั้น ในการทำงานก็อาจมีการล็อกเฉพาะเขตข้อมูลปริมาณสินค้าที่มีอยู่ในมือเท่านั้น แต่ส่วนใหญ่แล้วการล็อกในระดับนี้จะใช้น้อยมาก

เรื่องที่ 11.2.3 ปัญหาและการแก้ไขการเกิดเดดล็อก

1.การเกิดเดดล็อก

การเกิดเดดล็อก (dead lock) คือ เหตุการณ์ที่ทรานแซกชันรอการใช้ข้อมูลที่ถูกล็อกโดยการทรานแซกชันอื่นอย่างไม่รู้จบในลักษณะเป็นลูกโซ่ เหมือนงูกินหาง เช่น การที่ทรานแซกชันที่ 2 รอคอยเพื่อใช้ทรัพยากร ได้แก่ ข้อมูลในฐานข้อมูลหรืออุปกรณ์อื่นๆ ที่กำลังถูกล็อกไว้และใช้งานโดยทรานแซกชันที่ 1 จึงทำให้ทรานแซกชันที่ 2 ไม่สามารถจะเรียกใช้ทรัพยากรนั้นในขณะเดียวกันได้ ดังนั้นทรานแซกชันที่ 2 จะต้องรอจนกว่าทรานแซกชันที่ 1 จะทำงานเสร็จและคลายล็อก (unlock) ทรัพยากรนั้นเสียก่อน อย่างไรก็ตามทรานแซกชันที่ 1 ก็ยังไม่สามารถทำงานใดๆ ต่อได้ เพราะทรานแซกชันที่ 1 ต้องการเรียกใช้ทรัพยากรที่กำลังถูกใช้งานและถูกล็อกไว้โดยทรานแซกชันที่ 2 จึงทำให้ทรานแซกชันที่ 1 ต้องรอทรานแซกชันที่ 2 ทำงานนั้นให้เสร็จและคลายล็อกทรัพยากรนั้นเสียก่อน โดยที่ทั้ง 2 ทรานแซกชันต่างไม่รู้ว่าตนเองรอทรัพยากรของอีกฝ่ายซึ่งต่างฝ่ายต่างล็อกไว้และกำลังใช้งานอยู่ ดังนั้นจึงไม่มีทรานแซกชันใดสามารถจะดำเนินการใดต่อไปได้ จะต้องอยู่ในสภาพหยุดนิ่งและรอไปเรื่อยไม่รู้จบ เหตุการณ์ลักษณะนี้เรียกว่า เดดล็อก (dead lock) ดังภาพที่ 11.3 แสดงเหตุการณ์ที่เกิดเดดล็อก



ภาพที่ 11.3 แสดงเหตุการณ์ของการเกิดเดดล็อก

จากภาพเกิด deadlock ขึ้นได้เนื่องจาก ณ เวลา t1 transaction 1 กำลังใช้ข้อมูล Y อยู่และในขณะเดียวกัน transaction 2 ก็กำลังใช้ข้อมูล X อยู่ ในช่วงเวลา t2 transaction 2 ต้องการใช้ข้อมูล Y ซึ่ง transaction 1 ยังใช้อยู่ ส่วน transaction 2 ก็ยังใช้ข้อมูล X ที่ transaction 1 ต้องการใช้อยู่ ลักษณะนี้จะทำให้เกิด deadlock ขึ้น

2.การแก้ปัญหาการเกิดเดดล็อก

วิธีการแก้ปัญหาการเกิดเดดล็อก มีดังนี้

2.1 การป้องกันก่อนเกิดเดดล็อก (deadlock prevention) วิธีนี้ระบบจัดการฐานข้อมูลในส่วนของการควบคุมภาวะพร้อมกันจะกำหนดว่า ทรานแซกชันที่ต้องการเรียกใช้ข้อมูลใดก็ตามจะต้องล็อกข้อมูลทุกอันที่ต้องการเรียกใช้ทั้งหมดไว้ก่อนการใช้งาน ถ้าหากทรานแซกชันไม่สามารถจะล็อกข้อมูลไว้ล่วงหน้าได้ จะต้องรอจนกว่าจะล็อกได้ครบเสียก่อนจึงจะเริ่มต้นทำงานได้วิธีการป้องกันการเกิดเดดล็อก (dead lock prevention)

วิธีนี้คือให้แน่ใจว่าการเกิดเดดล็อกจะไม่มีวันเกิดขึ้น ซึ่งวิธีการป้องกันนี้ง่ายต่อการใช้งานเที่ยงตรงแน่นอน วิธีการป้องกันการเกิดเดดล็อกมีอยู่หลายวิธีด้วยกันอย่างง่าย คือ transaction ต้องขอใช้ทรัพยากรทุกชิ้นที่ต้องการพร้อม ๆ กันในคราวเดียว คือ หมายถึงว่า transaction นั้นถ้าได้ใช้ทรัพยากรก็จะได้หมดทุกชิ้น หรือไม่เช่นนั้นก็ไม่ได้เลย หรืออีกวิธีหนึ่งคือให้ transaction ที่ของทรัพยากรแล้วไม่ได้ทันที ให้ปลดปล่อยทรัพยากรที่ครอบครองไว้ทั้งหมด แล้วขอใหม่ร่วมกับทรัพยากรที่ขอไม่ได้นั้น

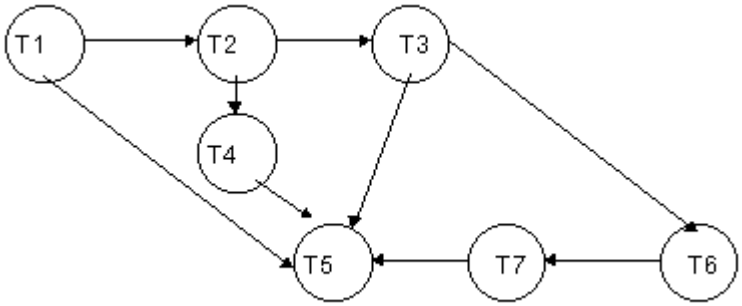
2.2 การตรวจจับการเกิดเดดล็อก (deadlock detection) วิธีนี้เป็นวิธีแก้ปัญหาเมื่อเกิดเดดล็อกแล้ว โดยระบบจัดการฐานข้อมูลจะตรวจจับว่าทรานแซกชันใดทำให้เกิดเดดล็อกบ้าง โดยจะตรวจจับจากกราฟที่เรียกว่า “wait-for graph” การตรวจจับการเกิดเดดล็อก

วิธีนี้จะย่อมให้เกิดเดดล็อกได้และพยายามตรวจตราระบบเพื่อหาการเกิดเดดล็อกแล้วทำการแก้ไข หลักการโดยทั่วไปคือ ตรวจหาการรอเป็นวัฏจักร (Circular wait) ระหว่าง transaction ต่าง ๆ โดยเครื่องมือที่สำคัญ คือ resource allocation graph หรือ wait-for graph ซึ่งเป็นกราฟแสดงความสัมพันธ์ระหว่างทรัพยากรและกระบวนการต่าง ๆ ในแง่ของการจับจองใช้และการขอใช้ วิธีการตรวจจับการเกิดเดดล็อกมีตั้งแต่แบบง่าย ๆ โดยอาศัยประสบการณ์และความชำนาญของพนักงานควบคุมเครื่องที่จะเข้าใจถึงระบบทั้งหมดโดยสามารถทราบถึงจุดที่เกิดการหยุดชะงักไม่ดำเนินไปตามที่เคยเป็น ปัญหาที่ตามมาคือ การแก้ไขการเกิดเดดล็อกที่เกิดขึ้นแล้ว ซึ่งก็มีตั้งแต่ปิดระบบทั้งหมดแล้วเริ่มกันใหม่ หรือวิธีที่ละเอียดอ่อนกว่านั้น คือ ตัด

T1 รอข้อมูลที่ถูกล็อกโดย T2 และT5
T2 รอข้อมูลที่ถูกล็อกโดย T3และ T4
T3 รอข้อมูลที่ถูกล็อกโดย T5 และT6
T4 รอข้อมูลที่ถูกล็อกโดย T5
T6 รอข้อมูลที่ถูกล็อกโดยT7
T7 รอข้อมูลที่ถูกล็อกโดยT5

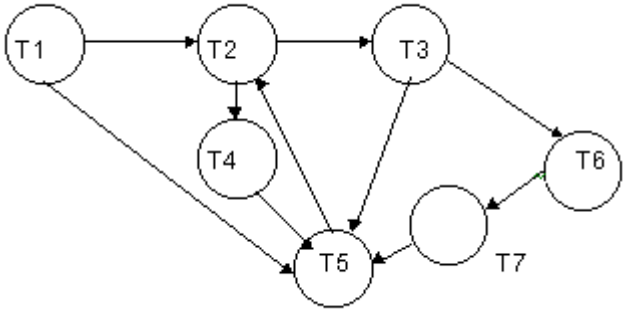
transaction ออกจากวัฏจักรการรอคอยที่ละ transaction จนระบบดำเนินต่อไปได้ ดังตัวอย่างเช่น

ซึ่งระบบจัดการฐานข้อมูลจะใช้ข้อมูลจากตัวอย่างข้างต้นสร้างเป็นกราฟเพื่อตรวจจับเดดล็อก ดังแสดงในภาพที่ 11. ดังนี้



ภาพที่ 11. 4 แสดงทรานแซกชันใน wait-for-graph ที่ไม่เกิดเดดล็อก

จากภาพที่ 11.4 จะเห็นว่า ถึงแม้ว่าใน wait for graph มีการรอข้อมูลที่ถูกล็อกโดยทรานแซกชันต่างๆ แต่จะไม่มีทรานแซกชันใดที่ทำให้เกิดเดดล็อก แต่ถ้าหากในภาพที่ 11.4 มีการเพิ่มเติมโดยให้ทรานแซกชันที่ 5 รอข้อมูลที่ถูกล็อกโดยทรานแซกชันที่ 2 จะทำให้เกิดเดดล็อกดังแสดงในภาพที่ 11.5



ภาพที่ 11. 5 แสดงทรานแซกชันใน wait-for-graph ที่ทำให้เกิดเดดล็อกโดย T2, T4, T5

ถ้าหากพบว่ามีทรานแซกชันใน wait for graph นั่นคือ มีการรอในลักษณะวนเป็นลูกโซ่ ซึ่งเป็น wait for graph ที่เกิดเดดล็อก ระบบจัดการฐานข้อมูลจะตรวจสอบและยกเลิกการทำงานของทรานแซกชันที่ทำให้เกิดเดดล็อก ซึ่งอาจจะมีการโรลแบ็กเพื่อให้ทรานแซกชันเริ่มต้นทำงานใหม่ทั้งหมด วิธีนี้สิ้นเปลือง (overhead) มาก เพราะระบบจัดการฐานข้อมูลต้องคอยเพิ่มเติมข้อมูลเข้าใน wait for graph เพื่อใช้ในการตรวจจับเดดล็อกตลอดเวลา

เรื่องที่ 11.2.4 วิธีการควบคุมภาวะความพร้อมกัน

ในการควบคุมภาวะพร้อมกัน ระบบจัดการฐานข้อมูลจะต้องมีกลไกสำหรับจัดเรียงลำดับก่อนหลังการทำงานของแต่ละทรานแซกชัน (scheduler) ซึ่งกลไกดังกล่าวอาจจะใช้วิธีวิธีใดวิธีหนึ่งต่อไปนี้ในการจัด

เรียงลำดับทรานแซกชัน ได้แก่ ล็อกกิง (locking), ไทม์แสตมป์ (time stamp) และออปทิมิสทิก (optimistic) เป็นต้น

1.Locking Technique

Locking Technique เป็นวิธีการกำหนดสถานะของข้อมูลเพื่อให้แต่ละทรานแซกชันที่ต้องการใช้ข้อมูลได้ล็อกข้อมูลนั้นในฐานข้อมูลไม่ให้ทรานแซกชันอื่นเรียกใช้ได้จนกว่าทรานแซกชันที่เรียกใช้ข้อมูลนั้นจะทำงานเสร็จสิ้นสมบูรณ์เสียก่อนแล้วจึงคลายล็อกข้อมูลนั้น ทรานแซกชันอื่นจึงจะใช้ข้อมูลชุดนั้นทำงานได้) ข้อเสียของวิธีล็อกกิงก็คือ การเกิดการล็อกโดยไม่คลายหรือเดดล็อก (deadlock)

การล็อก (Looking) การล็อกเป็นกระบวนการอย่างหนึ่งซึ่ง DBMS มีไว้ให้เพื่อใช้ในการควบคุมภาวะการเข้าถึงข้อมูลพร้อมกัน โดยสามารถเขียนโปรแกรมประยุกต์ควบคุมการล็อกขึ้นเอง ถ้าผู้ใช้คนใดต้องการแก้ไข (Update) ข้อมูลนั้นจะถูกล็อกเอาไว้ ผู้ใช้คนอื่นจะถูกปฏิเสธไม่ให้สามารถแก้ไขข้อมูลที่ถูกล็อกไว้แล้วนั้นได้ จนกว่าคนที่ทำการล็อกข้อมูลนั้นจะทำการแก้ไขข้อมูลเสร็จ (หรือยกเลิกการแก้ไข) ก็จะปล่อยล็อกให้ผู้อื่นสามารถแก้ไขข้อมูลนั้นได้ ตัวอย่างในตารางที่11.9 แสดงถึงการล็อกระเบียนบัญชีเงินฝาก A โดยเมื่อ Transaction 1 ทำการถอนเงินออกจากบัญชีนั้นคือต้องการทำการแก้ไขระเบียนบัญชีเงินฝาก A ของ Transaction 1 ดังนั้น โปรแกรมที่เขียนก็จะทำการล็อกระเบียนบัญชีเงินฝาก A ก่อนที่จะมีการอ่านระเบียนนั้นขึ้นมาไว้ในหน่วยความจำหลัก หลังจากนั้น Transaction 1 ก็จะทำการถอนเงิน 200 บาท ยอดเงินคงเหลือก็จะถูกปรับปรุงเป็น 800 บาท ในระหว่างนั้น ถ้า Transaction 2 ทำรายการกับบัญชีเงินฝาก A จะไม่สามารถทำรายการได้ จนกว่ารายการของ Transaction 1 จะทำการปรับปรุงระเบียนบัญชีเงินฝาก A ในฐานข้อมูลจนเสร็จเรียบร้อยและทำการปล่อยล็อก

ตารางที่ 11.9 แสดงการล็อกระเบียน

Transaction 1		Transaction 2
ล็อกระเบียนบัญชีเงินฝาก A	Time 1	
อ่านระเบียนบัญชีเงินฝาก A	Time 2	ถอนเงินจากบัญชีเงินฝาก A (ไม่สามารถทำรายการได้)
ทำรายการถอนเงิน 200 บาท	Time 3	50
ปรับปรุงยอดบัญชีเงินฝาก A	Time 4	
ปล่อยล็อกระเบียนบัญชีเงินฝาก A	Time 5	
	Time 6	ล็อกระเบียนบัญชีเงินฝาก A
	Time 7	อ่านระเบียนบัญชีเงินฝาก A
	Time 8	ทำรายการถอนเงิน
	Time 9	ปรับปรุงยอดบัญชีเงินฝาก A
	Time 10	ปล่อยล็อกระเบียนบัญชีเงินฝาก A

ณ เวลา Time 1 Transaction 1 ต้องการทำรายการสอบถามยอดเงินคงเหลือจากบัญชีเงินฝาก A ดังนั้นโปรแกรมจะทำการล็อกระเบียนบัญชีเงินฝาก A แบบ S ให้เนื่องจากเป็นการอ่านข้อมูลอย่างเดียว และในเวลาไล่เรี่ยกันคือ เวลา Time 2 Transaction 2 ได้ทำรายการสอบถามยอดเงินคงเหลือจากบัญชีเงินฝาก A เช่นกัน ดังนั้น โปรแกรมจะทำการล็อกแบบ S ให้ด้วยที่เวลา Time 3 Transaction 1 ได้ทราบยอดเงินคงเหลือของตน และที่เวลา Time 4 Transaction 1 ต้องการทำรายการถอนเงินต่อไป ซึ่งการถอนเงินนี้จะต้องมีการปรับปรุงยอดเงินคงเหลือ ดังนั้นโปรแกรมจะต้องทำการขอล็อกแบบ X แต่เนื่องจากระเบียนบัญชีเงินฝาก A นี้ได้ถูก Transaction 2 ทำการล็อกแบบ S ไว้อยู่ด้วย ดังนั้น รายการของ Transaction 1 จะต้องรอให้ Transaction 2 ปล่อยล็อกแบบ S นี้ก่อน และเนื่องจากเวลา Time 6 Transaction 2 แสดงความประสงค์จะทำรายการถอนเงินบ้าง ซึ่งก็จะต้องขอล็อกแบบ X ก็จะทำไม่ได้เช่นกัน เนื่องจาก Transaction 1 ได้ล็อกแบบ S อยู่ ดังนั้น แต่ละ Transaction ก็จะทำการรอให้แต่ละ Transaction ทำรายการเสร็จซึ่งจะเป็นการรอแบบไม่รู้จบ ลักษณะเช่นนี้จะทำให้เกิดล็อกค้าง (deadlock) ขึ้น ดังนั้นทั้งสอง Transaction จึงไม่สามารถทำรายการถอนนี้ได้ทั้ง ๆ ที่เงินในบัญชีมีเพียงพอที่จะถอนได้

ตารางที่ 11.10 การควบคุมภาวะพร้อมกันโดยวิธีล็อกกิง

Transaction 1		Transaction 2
ทำการสอบถามยอดเงินคงเหลือ	Time 1	
	Time 2	ทำรายการสอบถามยอดเงินคงเหลือ (ทำการ Lock แบบ S)
อ่านระเบียนบัญชีเงินฝาก A	Time 3	
	Time 4	อ่านระเบียนบัญชีเงินฝาก A
ทำการถอนเงินจากบัญชี A (ขอล็อกแบบ X ถูกปฏิเสธ)	Time 5	
500	Time 6	ทำการถอนเงินจากบัญชี A (ขอล็อกแบบ X ถูกปฏิเสธ)

2. Time Stamp Technique

Time Stamp Technique คือวิธีการควบคุมภาวะความพร้อมกัน โดย ระบบจัดการฐานข้อมูลจะระบุลำดับของแต่ละทรานแซกชันในการเข้าทำงาน ดังนั้นการควบคุมภาวะพร้อมกันโดยวิธีไทม์แสตมป์จะไม่มีปัญหาการเกิดเดดล็อกเหมือนวิธีล็อกกิงการกำหนดค่าไทม์แสตมป์อาจจะทำได้ดังนี้

●การกำหนดตัวแปรเพื่อเก็บค่าไทม์แสตมป์โดยนำค่าของเวลาในระบบเครื่องคอมพิวเตอร์ (system clock) มาเป็นลำดับที่ของทรานแซกชัน โดยทำงานเรียงลำดับการเข้าตามเวลาก่อนหลังของแต่ละทรานแซกชัน

● การกำหนดตัวแปรโดยสร้างตัวนับ (counter) ขึ้นเอง ดังนั้นเมื่อมีทรานแซกชันต้องการเรียกใช้ข้อมูลก็จะกำหนดตัวนับให้เพิ่มทีละหนึ่งโดยเรียงลำดับ 1, 2, 3 ... เพื่อจัดลำดับการทำงานของทรานแซกชัน วิธีนี้จะต้องมีการเซต (reset) ตัวนับให้เป็นศูนย์ หากค่าตัวนับเพิ่มค่าไปจนถึงค่าสูงสุดแล้ว

โดยสรุปจะเห็นว่าวิธีการควบคุมภาวะพร้อมกันโดยเทคนิคไทม์แสตมป์จะไม่ทำให้เกิดเหตุการณ์เดดล็อก เพราะเป็นการเรียงลำดับตามการเกิดของทรานแซกชันก่อนหลังไปเรื่อยๆ

3. Optimistic Technique

Optimistic Technique เป็นวิธีหรือเทคนิคในการควบคุมภาวะพร้อมกันที่กล่าวแล้ว 2 วิธีแรกนั่นคือ วิธีล็อกกิงและไทม์แสตมป์จะเป็นการตรวจสอบก่อนทรานแซกชันก่อนการทำงาน ซึ่งทำให้เกิดความสิ้นเปลือง (overhead) ในระบบจัดการฐานข้อมูลมากขึ้น แต่การควบคุมภาวะพร้อมกันโดย Optimistic Technique จะไม่มีการตรวจสอบล่วงหน้าก่อนทรานแซกชันทำงานหรือขณะทำงาน แต่จะอยู่บนแนวคิดที่ว่าให้ทรานแซกชันทำงานใช้ข้อมูลตามปกติ ข้อมูลที่มีการแก้ไขหรือเปลี่ยนแปลงโดยแต่ละทรานแซกชันจะไม่บันทึกไว้ในฐานข้อมูลโดยตรง แต่จะบันทึกไว้ในพื้นที่อื่นๆ ที่กำหนดไว้ชั่วคราวของแต่ละทรานแซกชัน จนกว่าทรานแซกชันนั้นจะทำงานเสร็จสมบูรณ์เสียก่อน ทั้งนี้ก่อนที่นำข้อมูลที่เก็บไว้ในพื้นที่ชั่วคราวมาบันทึกถาวรในฐานข้อมูล ระบบจัดการฐานข้อมูลจะตรวจสอบเสียก่อนว่า ทรานแซกชันนั้นมีการทำงานแบบขั้นตอนไม่เป็นลำดับ ซึ่งจะทำให้เกิดข้อมูลขัดแย้งกับข้อมูลของทรานแซกชันอื่นหรือไม่ หากตรวจแล้วไม่มีความขัดแย้ง ก็จะนำข้อมูลที่เก็บไว้ในพื้นที่ชั่วคราวบันทึกในตัวฐานข้อมูลอย่างถาวรต่อไป แต่ถ้าหากตรวจสอบแล้วมีปัญหาระบบจัดการฐานข้อมูลก็จะยกเลิกผลลัพธ์ของทรานแซกชันนั้น และมีการโรลแบ็ก (roll back) เพื่อเริ่มต้นทำงานทรานแซกชันนั้นใหม่ต่อไป

การควบคุมภาวะพร้อมกันโดยวิธี Optimistic Technique แบ่งขั้นตอนตามการทำงานเป็น 3 ขั้นคือ

● ขั้นการอ่าน (read phase) เป็นขั้นที่ทรานแซกชันอ่านข้อมูลจากฐานข้อมูล และทำการปรับปรุงแก้ไขข้อมูลที่ต้องการ หลังจากนั้นจะบันทึกไว้ในพื้นที่ชั่วคราวที่กำหนดไว้สำหรับทรานแซกชันนั้น ซึ่งทรานแซกชันอื่นไม่สามารถใช้ข้อมูลของทรานแซกชันในพื้นที่ชั่วคราวได้

● ขั้นการตรวจสอบ (validation phase) เป็นขั้นที่ระบบจัดการฐานข้อมูลจะตรวจสอบลำดับการทำงานของทรานแซกชัน ถ้าหากไม่มีปัญหาก็จะไปทำขั้นการบันทึกต่อไป ถ้าหากมีปัญหาก็จะยกเลิกการทำงานและผลลัพธ์ของทรานแซกชันนั้น

- ขั้นการบันทึก (write phase) จะนำข้อมูลที่เก็บไว้ชั่วคราวในพื้นที่ทำงานของทรานแซกชันนั้นมาบันทึกถาวรในฐานข้อมูล ถ้าหากมีปัญหาจากขั้นตรวจสอบก็จะไม่มีการบันทึกข้อมูลในฐานข้อมูล และทรานแซกชันนั้นจะต้องทำงานใหม่

นอกจากนี้ยังมีวิธีการควบคุมภาวะพร้อมกันวิธีอื่นอีก เช่น มัลติเวอร์ชัน (multiversion) ซึ่งเป็นวิธีหนึ่งที่ได้รับความนิยมเช่นกัน