ระบบฐานข้อมูลที่มีคุณสมบัติของเวลาและมีระดับความปลอดภัยหลายระดับ

A BITEMPORAL MULTILEVEL SECURE DATABASE SYSTEM

วรวิทย์ หมื่นราช

WORAWIT MEANRACH

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2550

ระบบฐานข้อมูลที่มีคุณสมบัติของเวลาและมีระดับความปลอดภัยหลายระดับ

A BITEMPORAL MULTILEVEL SECURE DATABASE SYSTEM

วรวิทย์ หมื่นราช

WORAWIT MEANRACH

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2550

A BITEMPORAL MULTILEVEL SECURE DATABASE SYSTEM

WORAWIT MEANRACH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อวิทยานิพนธ์	ระบบฐานข้อมูลที่มีคุณสมบัติของเวลาและมีระดับความปลอดภัยหลายระดับ
นักศึกษา	นายวรวิทย์ หมื่นราช
รหัสนักศึกษา	45061053
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.ศุภมิตร จิตตะยโศธร

บทคัดย่อ

วิทยานิพนธ์ได้กล่าวถึงฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับซึ่งได้กำหนดระดับชั้นให้กับผู้ใช้ และเช่นเดียวกันกับข้อมูลมีการกำหนดระดับความปลอดภัยด้วย นอกจากนี้มีการใช้คุณสมบัติของกฎพื้นฐานจาก เบล และลาพาดูลา ซึ่งเป็นวิธีการของการจัดการข้อมูลที่มีระดับความปลอดภัยหลายระดับมาใช้กับฐานข้อมูลเชิงสัมพันธ์และได้แก้ไขปรับปรุงกฎพื้นฐานดังกล่าวให้เหมาะสมกับการใช้งานมากยิ่งขึ้น นอกจากคุณสมบัติการแบ่งข้อมูลเป็นระดับชั้นยังเพิ่มคุณสมบัติของฐานข้อมูลที่แปรเปลี่ยนตามเวลาเข้าไป โดยการเพิ่มคุณสมบัติของเวลาสองส่วนคือ เวลาที่ทำให้ข้อมูลเป็นจริง และเวลาขณะบันทึก ให้กับแอททริบิวของข้อมูลที่มีระดับความปลอดภัยหลายระดับทำให้แทนที่จะเก็บข้อมูลเฉพาะที่เป็นปัจจุบันเท่านั้น ยังสามารถเก็บข้อมูลในอดีตและวางแผนข้อมูลในอนาคตได้อีกด้วย การทำเช่นนี้ทำให้เกิดความซับซ้อนของข้อมูลมากขึ้น เราได้ออกแบบโมเดลให้สามารถรองรับคุณสมบัตินี้ได้เพื่อแสดงให้เห็นว่าทำอย่างไรให้สามารถพัฒนาได้ สิ่งที่ตามมาคือการตีความหมายซึ่งข้อมูลมีหลายระดับความปลอดภัยและมีเวลามาเกี่ยวข้อง นอกจากนี้ยังมีการกำหนดกฎข้อบังคับเพื่อไม่ให้เกิดความกำกวมแก่ฐานข้อมูล ต้องใช้กฎข้อบังคับของทั้งโมเดลที่มีระดับความปลอดภัยหลายระดับ และฐานข้อมูลที่แปรเปลี่ยนตามเวลามาปรับปรุงเพื่อให้ใช้งานร่วมกัน ในด้านของการใช้งานเราได้ปรับปรุงการแก้ไขข้อมูลนั้นเพื่อความสะดวกจึงทำให้มีลักษณะคล้ายภาษาเอสคิวแอลดั้งเดิม และเพิ่มคุณสมบัติในการไควรีให้มีความสามารถเพียงพอให้ใช้กับข้อมูลที่มีระดับความปลอดภัยหลายระดับ และมีข้อมูลที่แปรเปลี่ยนตามเวลาในส่วนของการแก้ไขข้อมูลซึ่งได้แก่การคำสั่ง INSERT, UPDATE, DELETE และคำสั่งเพิ่มเติมเข้ามา UPLEVEL ก็ได้รวมอยู่ในวิทยานิพนธ์ฉบับนี้

Thesis Title	A Bitemporal Multilevel Secure Database System
Student	Mr.Worawit Meanrach
Student ID.	45061053
Degree	Master of Engineering
Program	Computer Engineering
Year	2007
Thesis Advisor	Assoc. Prof. Dr. Suphamit Chittayasothorn

ABSTRACT

Multilevel database security is a security management which provides clearance levels for users. It also provides classification labels for the data. Most multilevel secure database systems employ the mandatory access control as proposed by Bell and La Padula. Most research works apply the multilevel secure technique to conventional relational databases. This research project's proposes an extension to the mandatory access control so that it suits our application requirements. Moreover, the support of bitemporal database features is also included. Both valid time and transaction time are added to the already complex multilevel secure relational database. Integrity rules for this new model are introduced and finally, an extension to the SQL language is proposed to be used on the data model.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดีด้วยคำแนะนำ และคำปรึกษาจาก รศ.ดร.ศุภมิตร จิตตะ-ยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์ และขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือ ในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

วรวิทย์ หมื่นราช

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ระบบฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับ หรือระบบฐานข้อมูลเอ็มแอลเอส (Multilevel Security Database System) เป็นระบบฐานข้อมูลที่มีความสามารถในการปกปิดข้อมูลได้ในบางส่วนหรือทั้งหมด การปกปิดข้อมูลนี้ทำให้การมองข้อมูลเดียวกันจากผู้ใช้ต่างระดับกันแตกต่างกันได้ โดยได้มีการแบ่งระดับชั้นในระดับของข้อมูลและระดับของผู้ใช้ ระดับความปลอดภัยที่แตกต่างกันนี่เองเป็นตัวกำหนดมุมมองที่มีต่อข้อมูล

ในอดีตระบบเช่นนี้ได้คิดค้นสำหรับงานการทหารที่มีข้อมูลข่าวกรองซึ่งเป็นข้อมูลที่อยู่บนพื้นฐานความเชื่อและข้อสันนิษฐาน มีความเป็นได้ว่าข้อมูลเป็นจริง หรือเป็นเท็จก็ได้ ทุกข้อมูลพร้อมที่จะถูกเปลี่ยนแปลงตลอดเวลา ดังนั้นการบันทึกการเปลี่ยนแปลงของข้อมูลก็เป็นเรื่องจำเป็น เราได้เพิ่มเติมฐานข้อมูลที่มีคุณสมบัติของเวลา (Temporal Database) เข้าไปด้วย เวลาที่กำกับแต่ละข้อมูลนั้นยังแบ่งออกเป็นสองส่วน (Bitemporal Database) คือเวลาที่ใช้กำกับในความเป็นจริง และเวลาที่ใช้กำกับขณะบันทึกข้อมูล เมื่อเรานำคุณสมบัติที่กล่าวมาทั้งหมดรวมกันแล้วเราเรียกโมเดลใหม่ว่าเอ็มเอสบีดี (Multilevel Secure Bitemporal Data Model)

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อศึกษาและออกแบบโครงสร้างฐานข้อมูล เพื่อรองรับการใช้งานฐานข้อมูลที่มีความระดับความปลอดภัยหลายระดับ และมีคุณสมบัติของเวลา โดยกำหนดคุณสมบัติการใช้งานให้เหมาะสม และสอดคล้องกับการใช้งานรวมถึงกำหนดโครงสร้างพื้นฐานที่ประยุกต์ใช้กับฐานข้อมูลเชิงสัมพันธ์ ด้วยการสร้างกฎข้อบังคับรวมถึงวิธีการใช้งาน เพื่อให้สามารถแบ่งผู้ใช้ออกเป็นหลายระดับความปลอดภัย รวมถึงการให้มีเวลาเข้ามาเกี่ยวข้องที่ฐานข้อมูลซึ่งฐานข้อมูลเชิงสัมพันธ์แบบปกติไม่ได้มีการรองรับการใช้งานเหล่านี้ไว้

1.3 สมมติฐานของการศึกษา

การรักษากฎเกณฑ์พื้นฐานของฐานข้อมูลเชิงสัมพันธ์เป็นแนวทางในการประยุกต์โมเดลใหม่ใช้เป็นเครื่องยืนยันได้ว่าสมมติฐานสำหรับโมเดลใหม่ใช้งานได้จริง นอกจากนั้นยังต้องลดความซ้ำซ้อนให้มากที่สุด และไม่ให้เกิดความกำกวมของข้อมูล สำหรับการทำให้ฐานข้อมูลที่รองรับการ

ใช้งานในลักษณะความปลอดภัยหลายระดับและมีคุณสมบัติของเวลานั้น เราต้องการโครงสร้างข้อมูลที่นำโครงสร้างของฐานข้อมูลปกติมาปรับเปลี่ยน และกฎข้อบังคับมาปรับเปลี่ยนเพื่อความเหมาะสม ซึ่งมีหลายสิ่งหลายอย่างที่ต้องกำหนดขึ้น และอาจไม่เป็นไปตามข้อบังคับในแบบปกติ เราจำเป็นต้องทำข้อบังคับให้สอดคล้อง เพื่อไม่ให้เกิดความกำกวมของข้อมูล และไม่ขัดกับกฎเกณฑ์พื้นฐานที่พึงจะเป็น นั่นหมายความว่าสามารถรักษากฎเกณฑ์ของฐานข้อมูลเชิงสัมพันธ์ให้เป็นเช่นเดิม ซึ่งเปรียบเสมือนมีตารางจากฐานข้อมูลเชิงสัมพันธ์ซ้อนทับกันอยู่ในมุมมองของความแตกต่างของระดับความปลอดภัยและเวลา หรือพูดให้เข้าใจง่ายขึ้นถ้าเรามองเพียงระดับความปลอดภัยเดียว และมองที่เวลาเพียงจุดเดียว กับข้อมูลในตารางเอ็มเอสบีดีแล้ว เราสามารถมองเป็นข้อมูลจากตารางฐานข้อมูลเชิงสัมพันธ์แบบปกติได้ทุกประการ

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

งานวิจัยนี้ได้ยึดหลักการดั่งเดิมมาปรับปรุงตั้งแต่มีผู้เริ่มวิจัยเกี่ยวกับความปลอดภัยหลายระดับ คือกฎของเบล และพาลาดูลา [2] เหล่านี้ได้พัฒนาให้มีความชัดเจนยิ่งขึ้นเรื่อยมารวมถึงงานวิจัยนี้ก็ได้ทำการขยายความออกไปอีก และฐานข้อมูลเชิงเวลาที่ได้มีการตีพิมพ์ไปแล้วนับพันบทความ เราได้เลือกบทความที่มีความเหมาะสมและเป็นที่ยอมรับ นำมาปรับปรุงและใช้กับงานวิจัยของเรา แนวความคิดเหล่านี้ โดยเราได้ยึดการใช้ของผู้ใช้เป็นหลัก ดังนั้นจึงต้องมีความสะดวก และความชัดเจนสูง

1.5 การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีการแบบพื้นฐาน

ก่อนหน้านี้ได้มีบทความหนึ่งได้กล่าวถึงการรวมกันระหว่างคุณสมบัติความปลอดภัยหลายระดับ และคุณสมบัติการแปรผันตามเวลา ด้วยการกำหนดความสามารถแปรผันตามเวลาในลักษณะเวลาวาลิด (Valid Time) เป็นคาบเวลาที่กำกับว่าข้อมูลนั้นเป็นจริงในช่วงเวลาใดบ้าง ช่วงเวลาเหล่านี้กำหนดให้ทุกแอทริบิว สำหรับคาบเวลาที่ได้กล่าวถึงในตารางนี้ได้มีการสมมติให้เวลาอยู่ในรูปแบบของคาบเวลาที่อยู่ในช่วง $[0,\infty] = \{0, 1, 2, \ldots, now\}$ เป็นจุดของเวลาที่เพิ่มมากขึ้นตามแกนของเวลา ได้มีการกำหนดให้ "∞" แทนด้วยเวลาปัจจุบันโดยอ้างอิงกับเวลาของระบบ เพราะฉะนั้นในโมเดลของนิกินี้จะไม่กล่าวถึงเวลาในอนาคต และได้ทำการปรับเพื่อไปใช้กับฐานข้อมูลเชิงสัมพันธ์ ได้มีข้อกำหนดดังต่อไปนี้

1. ทุกแอททริบิวมีคาบเวลาของตนเอง
2. แต่ละแอททริบิวมีระดับความปลอดภัย
3. แต่ละแถวมี "tuple class" ในการกำหนดระดับความปลอดภัยระดับแถว

ตัวอย่างที่อธิบายในระบบฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับมักจะใช้ข้อมูลสำหรับการอธิบายจากภาพยนตร์เรื่องสตาร์เทรค ในข้อมูลนี้ประกอบด้วยยานอวกาศที่ได้รับมอบหมายงานไปยังดวงดาวต่างๆ ในจักรวาล การปฏิบัติการของยานแต่ละลำบางครั้งอาจมีจุดประสงค์อื่นแอบแฝง หรืออาจต้องการให้เป็นความลับ จึงสอดคล้องกับคุณสมบัติของเอ็มแอลเอสที่สามารถรองรับความต้องการนี้ได้ โดยตารางนี้มีชื่อว่าเอสโอดี (SOD) ซึ่งเป็นตัวย่อจากสามแอททริบิวคือ Starship, Objective และ Destination

ตารางที่ 1.1 ตารางเอสโอดีในโมเดลของนิกิ

Starship	*Starship Class*	*Objective*	*Objective Class*	*Destination*	*Destination Class*	*Tuple Class*
Enterprise	Unclassified	< [80,90] shipping >	Unclassified	< [85,89] Mars >	Unclassified	Unclassified
Enterprise	Unclassified	< [80,∞] shipping >	Unclassified	< [85,89] Mars >	Unclassified	Unclassified
Voyager	Classified	< [86,93] spying >	Classified	< [86,93] Pluto >	Classified	Classified

การพัฒนาในโมเดลของนิกิ ได้กล่าวถึงการจัดการข้อมูลที่แบ่งส่วนของเวลาเป็นสองส่วน ส่วนหนึ่งจัดการเก็บข้อมูลในปัจจุบัน อีกส่วนหนึ่งจัดการข้อมูลในอนาคต โดยใช้วิธีการของ "Cellular Chaining" ที่มีการแบ่งแต่ระดับความปลอดภัยแยกออกจากกัน และได้มีการกำหนดบีสตาร์ทรีเพื่อเป็นตัวชี้ส่วนของเวลาปัจจุบัน และเวลาในอดีต

จากโมเดลของนิกิเป็นการพัฒนาในลักษณะกายภาพคือการจัดการการเก็บข้อมูล ซึ่งแตกต่างจากโมเดลของเราที่นำเสนอในด้านโครงสร้างของข้อมูล ที่จะได้กล่าวถึงในบทที่ 4 เป็นการนำฐานข้อมูลเชิงสัมพันธ์ที่มีใช้อยู่ในปัจจุบันปรับให้สามารถรองรับของคุณสมบัติของความปลอดภัยหลายระดับและคุณสมบัติของเวลาได้

ในโมเดลของเราได้กำหนดข้อบังคับความถูกต้องของข้อมูล (Integrity Constraints) ไว้อย่างชัดเจน เนื่องจากส่วนสำคัญของงานวิจัยนี้คือต้องการไม่ให้เกิดความกำกวม และข้อมูลซ้ำซ้อน นั่นหมายความว่า การที่มีระดับความปลอดภัยหลายระดับต้องสามารถบ่งบอกได้ว่า ข้อมูลในแต่ละระดับเป็นเช่นไร โดยไม่มีข้อมูลที่อ้างถึงข้อมูลเดียวกันแต่มีสองความหมาย เช่นในตารางเอสโอดี ผู้ใช้ในแต่ละระดับต้องสามารถตีความได้เพียงหนึ่งเดียวว่ายานแต่ละลำได้เดินทางไปทำอะไร เมื่อไร หากเราสามารถระบุถึงข้อมูลแต่ละระดับแต่ละเวลาให้มีข้อมูลที่แน่ชัดไม่เกิดความกำกวมและข้อมูลซ้ำซ้อนแล้ว จะเปรียบเสมือนกับฐานข้อมูลเชิงสัมพันธ์ปกติที่มีเพียงระดับเดียว และเป็นเพียงข้อมูล ณ เวลาหนึ่งๆ เท่านั้น ดังนั้นการที่มีระดับความปลอดภัยหลายระดับ และคุณสมบัติของเวลานั้น เหมือนกับมีตารางของฐานข้อมูลเชิงสัมพันธ์เรียงรายอยู่มากมายที่สามารถจำแนกออกเป็นตารางเพียงตารางเดียวได้ด้วยการบ่งชี้ถึงระดับความปลอดภัยหนึ่ง และเวลาหนึ่ง

อย่างไรก็ตามโมเดลของนิกิได้กล่าวถึงเวลาวาลิดเพียงอย่างเดียวร่วมกับเอ็มแอลเอสและเป็นการกล่าวอย่างคราวๆ ไม่ได้ทำกฎข้อบังคับหรืออธิบายการใช้งานที่ชัดเจนแต่อย่างใด และแนวทางการพัฒนาเป็นในแนวทางการปรับปรุงเชิงกายภาพของระบบฐานข้อมูลคือการใช้อินเดกซ์ทรี ซึ่ง

แตกต่างกับแนวทางของงานวิจัยนี้ที่ได้เลือกฐานข้อมูลเชิงสัมพันธ์ปกติมาปรับปรุงโดยใช้ประโยชน์จากคุณสมบัติที่มีอยู่ปรับปรุงให้สามารถรองรับคุณสมบัติที่ต้องการได้

สำหรับในงานวิจัยของนี้ ได้ยึดพื้นฐานสำหรับคุณสมบัติความปลอดภัยหลายระดับจากโมเดลเอ็มแอลอาร์ (Multilevel Relation Model) [5] ได้มีการนำเสนอกฎที่ควบคุมการให้ข้อมูลจากระดับต่ำกว่าให้เป็นส่วนหนึ่งในข้อมูลที่ยอมรับโดยระดับที่สูงกว่า แต่พื้นฐานของฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับนั้น ไม่สามารถส่งข้อมูลใดๆ ให้แก่ระดับต่ำกว่าได้ เนื่องด้วยเหตุผลการป้องกันข้อมูลจากระดับบนรั่วไหลแต่ในความเป็นจริงแล้วมีหลายแอพพลิเคชันที่จำเป็นต้องส่งข้อมูลให้ระดับล่าง ที่มีลักษณะเป็นการออกคำสั่งจากระดับสูงกว่าให้แก่ระดับต่ำกว่า เราได้ทำการปรับปรุงให้สามารถทำในส่วนนี้ได้ ด้วยการกำหนดแอทริบิวใหม่ขึ้นมาคือ CC (Creator Class) นอกเหนือจากแอทริบิวเดิม TC (Tuple Class) ซึ่งเป็นตัวบ่งบอกถึงระดับที่สร้างแถวนั้น เรามีการแบ่งแยกข้อมูลออกเป็นสองส่วนคือ คำสั่ง และรายงานที่แบ่งแยกโดย ถ้า t[TC] = t[CC] หมายความว่าแถว t เป็นระดับที่ได้รับการยอมรับโดยระดับของตนเอง และเป็นผู้ที่สร้างแถวนี้ขึ้นเอง ถ้าระดับสูงกว่าสร้างแถวให้ยอมรับในระดับต่ำกว่า หรือ t[TC] < t[CC] แล้วถือเป็นคำสั่ง

ในตารางที่ 1.2 มีรูปแบบโครงสร้างที่ใช้ในงานวิจัยของเราที่มีทั้งคุณสมบัติของเวลาและมีหลายระดับความปลอดภัยโดยตารางนี้มีพื้นฐานโครงสร้างข้อมูลจากตารางเอสโอดี ระดับ S ได้ส่งคำสั่งให้ระดับ U โดยให้ทำการเพิ่มคำสั่งนี้ไปเมื่อ วันที่ 6 กันยายน ให้ปฏิบัติการวันที่ 10 ข้อมูลนี้สามารถมองเห็นได้ในระดับของและได้มีผู้ที่ออกคำสั่งคือ S ในคอลัมน์ CC และบอกว่าเป็นข้อมูลให้แก U ในคอลัมน์ TC นอกเหนือจากนั้นยังสามารถกำหนดขอบเขตของเวลาที่ออกคำสั่งรวมการรายงานกลับด้วย

ตารางที่ 1.2 ข้อมูลในตารางเอสโอดีที่ใช้โครงสร้างรูปแบบของโมเดลเอ็มเอสบีดี

SHIP	*OBJ*	*DEST*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
Enterprise U	Mining U	Talos U	2006-09-10	2006-09-15	2006-09-06	9999-12-31	U	S
Enterprise U	Mining U	Talos U	2006-09-11	2006-09-15	2006-09-16	9999-12-31	U	U

รายงานและคำสั่งแยกกันออกอย่างชัดเจนระดับสูงกว่าไม่สามารถแก้ไขรายงานของระดับต่ำกว่าได้ ดังเราได้เพิ่มประสิทธิภาพการใช้งาน โดยไม่ได้ละเมิดกฎเกณฑ์เดิม

ฐานข้อมูลเอ็มแอลเอสเต็มไปด้วยความซับซ้อนของข้อมูล ที่มีการปกปิด และข้อมูลลวง แต่ละระดับก็มีความเชื่อที่มีข้อมูลเป็นของตัวเอง ดังนั้นความคิดที่เชื่อต่อข้อมูลเดิมที่มีอยู่เปลี่ยนแปลงได้เสมอ จึงควรมีเก็บข้อมูลการแก้ไขทั้งหมด เพื่อสามารถติดตามและตรวจสอบข้อมูลที่ผ่านมาได้ นอกจากนั้นความสามารถในการเข้าถึงข้อมูลในอดีตหรือวางแผนงาน หรือการแก้ไขข้อมูลในแต่ละครั้งไม่ให้เกิดการศูนย์หาย เราได้ใช้ความสามารถในเชิงเวลาแบบไบเทมโปรัล (Bitemporal) [12] คือมีทั้งเวลาที่เป็นความเชื่อ และเวลาที่ทำการบันทึกไว้ในระบบ เป็นระบบที่มีสองคาบเวลามา

เกี่ยวเนื่อง ทำให้ข้อมูลทุกข้อมูลที่เคยบันทึกจะถูกเก็บไว้ทั้งสิ้นไม่มีการลบข้อมูลไปจริงๆ และยังกำหนดคาบเวลาที่เคยเชื่อว่าเป็นจริงเอาไว้ด้วย การรวมไบเทมโปรัลเข้ากับฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับ ทำให้สามารถเพิ่มขอบเขตความสามารถที่ในเอ็มแอลเอสไม่มี ฐานข้อมูลที่เป็นเอ็มแอลเอสเพียงอย่างเดียว สำหรับการกำหนดคีย์หลักต้องเป็นข้อมูลที่รู้จักกันดีในแต่ละระดับ ในตารางที่ 1.2 คีย์หลักในที่นี้คือ ชื่อยานอวกาศ Enterprise เป็นที่รู้จักกันดีในทุกระดับ ไม่ได้เป็นการกำหนดไว้ใช้เป็นส่วนตัวในระดับใดระดับหนึ่ง เนื่องด้วยไม่สามารถมองเห็นข้อมูลของระดับอื่นได้ทุกระดับ จึงไม่สามารถกำหนดเป็นคีย์หลักที่สร้างโดยอัตโนมัติดังเช่นลำดับของเลขจำนวนเต็มได้

1.6 ขอบเขตการวิจัย

งานวิจัยนี้ได้กำหนดโครงสร้างของฐานข้อมูล รวมถึงกฎข้อบังคับ และปรับให้ฐานข้อมูลปกติสามารถรองรับความสามารถของความปลอดภัยที่มีหลายระดับ และมีคุณสมบัติของเวลา นอกเหนือจากนั้นยังกำหนดฟังก์ชันการใช้งาน และได้นำเสนอการนำมาใช้งานให้สามารถนำมาใช้ได้จริง พร้อมทั้งนำเสนอออกเป็นตัวอย่าง และนำมาใช้จริงบนระบบฐานข้อมูลของออราเคิล เพื่อทดสอบและทำให้เข้าใจถึงความสามารถและการใช้งานของการวิจัยนี้

บทที่ 2
ฐานข้อมูลที่มีความปลอดภัยหลายระดับ

ถึงแม้ว่าระบบฐานข้อมูลที่มีการใช้งานในปัจจุบันสามารถรองรับรูปแบบของฐานข้อมูลในลักษณะที่เหมาะสมกับความต้องการของงานได้หลายประเภท แต่ข้อมูลที่มีความต้องการปิดเป็นความลับที่เป็นข้อมูลที่มีความอ่อนไหวต่อสถานภาพขององค์กร ข้อมูลประเภทนี้มีข้อจำกัดในการเผยแพร่แก่บุคคลอื่น หรือแม้แต่บางหน่วยงานในองค์กรเอง ฉะนั้นในช่วงสิบปีที่ผ่านมาได้มีงานวิจัยหลายงาน ได้พูดถึงการออกแบบฐานข้อมูลเพื่อให้เหมาะสมกับข้อมูลในลักษณะนี้ โดยฐานข้อมูลที่มีความปลอดภัยหลายระดับ (Multilevel Security) หรือเอ็มแอลเอส เป็นการจัดระบบความปลอดภัยที่ได้นำมาใช้ แต่เดิมเอ็มแอลเอสได้ออกแบบมาใช้กับงานทางทหาร เนื่องจากระบบทางการทหารมีข้อมูลที่เป็นความลับ และมีความละเอียดอ่อนของการเผยแพร่ข้อมูลสูง ข้อมูลทางการทหารบางอย่างจำเป็นต้องเก็บเป็นความลับเนื่องจากอาจส่งผลแก่ความมั่นคงของประเทศชาติ และมีความต้องการของการจัดเก็บของข้อมูลที่ต้องการความปลอดภัยจากผู้ที่ไม่ได้รับอนุญาตให้สามารถเข้าถึงข้อมูลส่วนนี้ได้ การจัดการสำหรับสิทธิ์การเข้าถึงข้อมูลนั้นต้องการฐานข้อมูลที่มีการแบ่งระดับของข้อมูลและผู้ใช้ เพื่อให้ข้อมูลในแต่ระดับสามารถเข้าถึงโดยผู้ใช้ที่มีสิทธ์เท่านั้น เช่นมีการแบ่งระดับชั้นของข้อมูลที่สามารถเปิดเผยได้ในเฉพาะหน่วยงาน หรือบุคคลบางกลุ่ม ซึ่งอาจแบ่งได้ตามชั้นยศ แต่แทนที่จะแยกข้อมูลเป็นความลับที่ไม่เปิดเผยแก่ระดับอื่นเลยเท่านั้น เราต้องการให้ข้อมูลนั้นกระจายออกไปได้ในบางส่วน หรือปกปิดในบางส่วน ทั้งยังทำให้การปกปิดนั้นสามารถให้ข้อมูลที่แตกต่างตามความเป็นจริงได้เป็นความต้องการของงานฐานข้อมูลในหลายๆ หน่วยงานไม่เฉพาะเพียงการทหารเท่านั้น รูปแบบของฐานข้อมูลประเภทนี้ยังมีการใช้งานกับองค์กรอื่นๆ ด้วยไม่ว่าจะเป็นในองค์กรธุรกิจถ้าหากต้องการปกปิดยอดขายที่แท้จริง หรือในทางการแพทย์ในงานตรวจรักษาของโรงพยาบาล ในกรณีที่ผู้ป่วย ป่วยเป็นโรคที่สังคมรังเกียจหรือไม่เป็นความประสงค์ของผู้ป่วยที่ต้องการให้ ผู้อื่นล่วงรู้อาการป่วยของตน หรือในกรณีที่มีผลต่อสถาบันความมั่นคง ดังนั้นในกรณีนี้ข้อมูลต่างๆ ของผู้ป่วยโรงพยาบาลอาจต้องเก็บข้อมูลไว้เป็นความลับ

2.1 กฎเกณฑ์พื้นฐานของฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับ

สิ่งที่แตกต่างจากฐานข้อมูลปกติคือแทนที่จะมีระดับของข้อมูลเพียงระดับเดียว ผู้ใช้ที่ได้รับอนุญาตเข้าถึงข้อมูลสามารถเข้าถึงข้อมูลที่เหมือนกันหมดคือมีระดับเดียวกันทั้งหมด แต่รูปแบบฐานข้อมูลเอ็มแอลเอสมีระดับของความปลอดภัย (security level) ต่างๆ กันในมากกว่าหนึ่งระดับ

การกำหนดระดับความปลอดภัยมีการให้ระดับแก่ข้อมูล (object) ตัวอย่างเช่น แถว, คอลัมน์ และการแบ่งระดับให้แก่ผู้ใช้ (subject) ระดับความปลอดภัยของข้อมูล (classification level) เช่น ลับสุดยอด (top secret), เป็นความลับ (secret), ปกปิด (classified), ไม่เป็นความลับ (unclassified) เช่นเดียวกันกับการกำหนดระดับของผู้ใช้ (clearance level) ก็มีลักษณะเดียวกัน ทุกระดับมีการกำหนดระดับเอาไว้ด้วย เช่น top secret > secret > classified > unclassified ซึ่งหมายถึง top secret นั้นมีระดับสูงสุดและ unclassified อยู่ในระดับต่ำสุด

ทั้งหมดได้ยึดเกณฑ์การเข้าถึงข้อมูลพื้นฐานของเบล และพาลาดูลา ซึ่งได้วางไว้ดังนี้

1. คุณสมบัติความปลอดภัยพื้นฐาน (Simple Security Property) โดยผู้ใช้สามารถอ่านข้อมูลได้ถ้าหากระดับความปลอดภัยนั้นตรงกันหรือสูงกว่า

2. คุณสมบัติสตาร์ (*-Property) โดยผู้ใช้สามารถทำการเข้าถึงข้อมูลแบบเขียนได้ก็เมื่อระดับความปลอดภัยนั้นตรงกันหรือต่ำกว่า

ในกฎข้อแรกมีจุดมุ่งหมายในความสามารถการเข้าถึงข้อมูลได้เมื่อผู้ใช้นั้นมีระดับความปลอดภัยที่เท่ากันหรือสูงกว่าระดับความปลอดภัยของข้อมูล กล่าวได้ว่าผู้ใช้นั้นมีสิทธิ์ในการอ่านครอบคุมเหนือข้อมูล ส่วนกฎข้อที่สองนั้นข้อมูลสามารถถูกแก้ไขได้ก็ต่อเมื่อระดับความปลอดภัยของผู้ใช้เท่ากันหรือต่ำกว่าระดับความปลอดภัยของข้อมูล คือนอกจากสามารถแก้ไขข้อมูลที่มีระดับตรงกันแล้ว ข้อมูลที่แก้ไขนี้ส่งผลถึงข้อมูลที่ระดับสูงกว่าสามารถอ่านได้ด้วย

2.2 ความหมายของข้อมูลในฐานข้อมูลเอ็มแอลเอส

เราสามารถแบ่งระดับชั้นของฐานข้อมูลได้อย่างง่ายด้วยการเพิ่มคอลัมน์หนึ่งเข้าไป เป็นตัวกำหนดความสามารถการเข้าถึงตามกฎของเบล และลาพาดูลา คอลัมน์ที่เพิ่มมานั้นเราเรียกว่า ทัปเปิลคลาส (tuple class) หรือ TC เป็นระดับความปลอดภัยในระดับของทั้งแถวข้อมูล ทำให้เราสามารถแบ่งแยกข้อมูลของแต่ละระดับชั้นได้

การนำเสนอเราขอยกตัวอย่างที่เป็นที่นิยมกันในงานวิจัยฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับ นั่นคือข้อมูลของยานอวกาศจากภาพยนตร์เรื่องสตาร์เทรก เป็นตัวอย่างที่มีมานานแล้ว เนื่องจากมีความเหมาะสมกับการอธิบายถึงฐานข้อมูลเอ็มแอลเอสได้เป็นอย่างดี ยานอวกาศที่เดินทางไปปฏิบัติภาระกิจ ยังสถานที่ๆ ถูกกำหนดไว้ จึงประกอบด้วยข้อมูลสามคอลัมน์คือ ชื่อยานอวกาศ (SHIP), ภาระกิจ (OBJ) และ สถานที่ (DEST)

ฐานข้อมูลของตัวอย่างตาราง SOD ซึ่งย่อมาจาก "Starship Name" , "Objective" และ "Destination" ตามลำดับ แสดงปฏิบัติการของยานอวกาศ

ตารางที่ 2.1 ตารางเอสโอดีในแบบจำลองเอ็มแอลเอสพื้นฐาน

SHIP	*OBJ*	*DEST*	*TC*
Enterprise	Spying	Mars	TS
Enterprise	Spying	Pluto	C
Enterprise	Shipping	Pluto	U

ถ้าเป็นข้อมูลในฐานข้อมูลแบบปกติแล้ว "SHIIP" เป็นคีย์หลัก เมื่อเราพิจารณาที่ระดับเดียวแล้วจากในรูปทั้งสามแถวอยู่คนละระดับ การมองเพียงระดับเดียวเห็นได้ว่ายังคงคุณสมบัติเดิมของคีย์หลักไว้ได้ แต่เมื่อมีข้อมูลรวมอยู่ด้วยกันแล้วเราไม่ได้ทิ้งกฎเกณฑ์เดิมไปเสียทีเดียว เรียกคีย์หลักใหม่ว่าคีย์หลักแอพพาเรนท์ (Apparent Primary Key) เราสามารถเขียนได้ว่า A_k, TC $\longrightarrow$ A_i โดย A_k คือคีย์หลักแอพพาเรนท์ A_i คือข้อมูลแต่ละแอททริบิว

ตัวอย่างการกำหนดระดับความปลอดภัยโดยการกำหนดเป็นสัญลักษณ์ย่อระดับความปลอดภัยดังตัวอย่างที่ผ่านมาคือ (TS = top secret, S = secret, C = classified, U = unclassified) นำมาใช้กับฐานข้อมูลภารกิจของยานอวกาศโดยการใส่ระดับความปลอดภัยให้กับทุกแถวดังตารางที่ 2.1

TC (tuple class) ใช้แสดงระดับความปลอดภัยของแต่ละแถว ผู้ใช้ในระดับ *TS* สามารถมองเห็นข้อมูลทั้งหมดเนื่องจากมีระดับผู้ใช้สูงที่สุด และมีสิทธิ์ในการอ่านข้อมูลได้ทั้งหมด ผู้ใช้ในระดับ *C* มองเห็นเฉพาะสองแถวล่าง และ ผู้ใช้ในระดับ *U* เห็นเฉพาะแถวสุดท้าย ถ้าหากผู้ใช้ระดับ *U* นั้นเป็นผู้ใช้ระดับต่ำสุดแล้วควรมองเห็นในลักษณะที่ไม่มีคอลัมน์ *TC* คือมองเห็นข้อมูลที่มีระดับเดียวดังตารางที่ 2.2 การที่มองข้อมูลเดียวกัน จากผู้ใช้ในระดับต่างกัน ที่อาจมองเห็นได้ไม่เหมือนกันนั้นเรียกว่าโพลีอินแสตนติเอชัน (Polyinstantiation)

จากตารางที่ 2.1 เรามองภาพรวมของข้อมูล หรือเป็นการมองจากระดับสูงสุดของตาราง คือ TS หรือระดับที่สูงกว่านี้ก็สามารถมองเห็นข้อมูลได้ทั้งหมด ข้อมูลในตารางฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับอาจไม่เป็นความจริงในโลกความเป็นจริง ซึ่งเหมือนกับว่าอาจมีการปกปิดหลอกลวง หรือมีการทำให้เข้าใจผิดกับข้อมูลโดยเจตนาได้

ตารางที่ 2.2 มุมมองของผู้ใช้ระดับ U ต่อตารางที่ 2.1

SHIP	*OBJ*	*DEST*
Enterprise	Shipping	Pluto

นอกเหนือจากการกำหนดระดับความปลอดภัยให้กับทุกแถวแล้ว ยังสามารถกำหนดระดับความปลอดภัยให้กับแต่ละแอททริบิวอีกด้วย ทำให้มีความสามารถในการสร้างความสอดคล้องของข้อมูลในแต่ละระดับสำหรับแอททริบิว การกำหนดระดับความปลอดภัยนั้นก็ทำในลักษณะเดียวกันกับที่ทำกับแต่ละแถวคือเพิ่มสัญลักษณ์ย่อของระดับให้กับแต่ละแอททริบิวจากตัวอย่างข้อมูลจากตารางเอสโอดีมีลักษณะดังรูปที่ 2.3 โดยกำหนดสัญญลักษณ์ย่อกับ *SC*, *OC*, *DC* เป็นระดับความ

ปลอดภัยของ "SHIP", "OBJ", และ "DEST" ตามลำดับ แสดงให้เห็นว่ายานเอนเตอร์ไพรส์ที่เป็นข้อมูลเห็นโดยผู้ใช้ระดับ *TS* ได้มีการปกปิดข้อมูลของภารกิจและดาวเป้าหมายกับระดับอื่น ผู้ใช้ระดับ C ไม่ทราบว่ายานเอนเตอร์ไพรส์ในความคิดของ TS เดินทางไปดาวอังคารไม่ใช่ดาวพลูโต และผู้ใช้ระดับ U ก็มีความคิดที่มีต่อยานเอนเตอร์ว่าไปทำการขนส่งซึ่งแตกต่างจาก C และ TS-subject ในลักษณะเช่นได้เรียกว่าการปิดบังข้อมูล (cover story) ส่วนในระดับความปลอดภัยในระดับแอททริบิวนั้นบ่งบอกความเชื่อว่าเป็นความเชื่อที่มีพื้นฐานมาจากของผู้ใช้ในระดับใด หากเป็นระดับที่ต่ำกว่าคือเชื่อว่าข้อมูลนี้เป็นจริงที่ระดับล่าง และยืนยันว่าเป็นข้อมูลที่เชื่อถือได้ที่ระดับตน ในตารางที่ 2.3 ภาระกิจของเอนเตอร์ไพรส์ในความคิดของผู้ใช้ระดับ TS มีความเชื่อว่าข้อมูลจากระดับ C นั้นเชื่อถือได้ และไว้วางใจให้เป็นข้อมูลที่ตนยอมรับในระดับของตนเอง

ตารางที่ 2.3 ตารางเอสโอดีที่มีการมีระดับความปลอดภัยในระดับแอททริบิวและแถว

SHIP	*SC*	*OBJ*	*OC*	*DEST*	*DC*	*TC*
Enterprise	U	Spying	C	Mars	TS	TS
Enterprise	U	Spying	C	Pluto	U	C
Enterprise	U	Shipping	U	Pluto	U	U

เห็นได้ว่าหากฐานข้อมูลที่ระดับเดียว โดยไม่มีระดับความปลอดภัยเข้ามาร่วมด้วย จะมี "SHIP" เป็นคีย์หลัก แต่ในเอ็มแอลเอสแล้วไม่เพียงพอที่เป็นคีย์หลักได้แต่เรียกว่าคีย์หลักแอพพาเรนท์ โดยใน เอ็มแอลเอสนั้นจะมีคุณสมบัติดังนี้คือ

- $A_1, C_1, C_i \longrightarrow A_i$
- $A_1, C_1, TC \longrightarrow A_i, C_i$

โดย A_1 เป็นคีย์หลักแอพพาเรนท์, C_1 เป็นระดับความปลอดภัยของคีย์หลักแอพพาเรนท์ และ C_i เป็นระดับความปลอดภัยของ A_i ซึ่งเป็นแอททริบิวอื่นที่ไม่ได้เป็นคีย์หลักแอพพาเรนท์ ในส่วนข้อมูลที่ได้รับการยืนยันจากผู้ใช้ TS คือข้อมูลที่อยู่ในระดับเดียวกันดูได้จาก TC ที่มีระดับความปลอดภัยเป็น TS และสามารถอ่านได้โดยผู้ใช้ในระดับ TS เท่านั้นหรือผู้ใช้ที่สูงกว่าถ้าหากมี ได้มีงานวิจัยมากมายกล่าวถึงการกำหนดกฎเกณฑ์สำหรับการทำความเข้าใจกับข้อมูลในระดับตนและระดับต่ำกว่า เพื่อลดความสับสนเนื่องจากคุณสมบัติของโพลีอินสแตนติเอชันยอมให้มีคีย์หลักแอพพาเรนท์ตรงกันได้ ในแต่ละข้อมูลภายในแถวนั้น สามารถมามองเห็นได้ต่อเมื่อเป็นข้อมูลที่ได้รับอนุญาตให้มองเห็นได้ ในส่วนถัดไปนั้นจะกล่าวถึงโมเดลเอ็มแอลเอสโมเดลต่างๆ

2.3 คุณสมบัติ และข้อดีข้อเสียของเอ็มเอลเอสในโมเดลอื่น

ในช่วงสิบปีที่ผ่านมาได้มีการคิดโมเดลที่สามารถ รองรับกฎพื้นฐานของ เบล และลาพาดูลา นำมาใช้ร่วมกับฐานข้อมูลเชิงสัมพันธ์หลายรูปแบบด้วยกัน ปัญหาหลักคือความกำกวมของข้อมูลในแต่ละระดับชั้นความปลอดภัย ได้เลือกมาเฉพาะงานวิจัยที่น่าสนใจดังต่อไปนี้

2.3.1 โมเดลของจาโจเดีย-ซานดู

จาโจเดีย-ซานดู (Jajodia-Sandhu) [4] ได้มีข้อกำหนดไว้สองส่วน ในส่วนแรกคือ โครงสร้างของตาราง $R(A_1, C_1, A_2, C_2,\ldots, A_n, C_n, TC)$ เมื่อ Ai เป็นแอททริบิวข้อมูลที่อยู่บนที่อยู่บนโดเมน D_i, และมี C_i เป็นระดับความปลอดภัยของแต่ละ A_i และ TC เป็นแอททริบิวทัปเปิลคลาส ในส่วนที่สองเป็นกลุ่มข้อมูลของตาราง $Rc(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$ ในแต่ละระดับความปลอดภัยก็จะมีเซทของแถวข้อมูลที่ต่างกันออกไปในรูปแบบของ $(a_1, c_1, a_2, c_2,\ldots, a_n, c_n, tc)$ เมื่อ tc ต้องมีระดับเป็นขอบเขตของระดับบนสุดของทุกแอททริบิวทั้งแถว และได้มีการกำหนดให้ระดับความปลอดภัยในแต่ละแถวไม่สามารถเป็นนัล (null) ได้

ในโมเดลนี้ไม่ได้เป็นไปตาม เบล และลาพาดูลา เสียทีเดียว เพราะได้มีการเพิ่มตัวป้องกันข้อมูลที่เรียกว่าโคเวิร์ทแชนเนล (covert channel) ซึ่งเป็นการทำให้ข้อมูลของผู้ใช้ที่ระดับสูงนั้นสามารถส่งลงมายังระดับต่ำกว่าได้ด้วย ถ้าหากเรามีข้อมูลดังตารางที่ 2.4 เป็นข้อมูลของผู้ป่วยในโรงพยาบาลที่ประกอบด้วยชื่อของผู้ป่วย (Patient_Name), กลุ่มโรค (Diagnosis), อายุ (Age) และเลขที่ห้อง (Room) ในโมเดลจาโจเดีย-ซานดูนี้ไม่เพียงแต่ผู้ใช้ระดับ U เท่านั้นที่สามารถเห็นเฉพาะข้อมูลในแถวที่สองเท่านั้น ยังสามารถมองเห็นข้อมูลบางส่วนในระดับ C ด้วย ดังในรูปที่ 2.5 ซึ่งเป็นมุมมองจาก U-subject ซึ่งข้อมูลของ Jones ที่ insert ลงไปโดย C-subject นั้นสามารถแสดงข้อมูลบางส่วนที่มี classification attribute ที่สามารถมองเห็นได้ให้กับ subject ในระดับต่ำกว่าได้ด้วย แต่ยังมีความกำกวมอยู่คือถ้าเป็นมุมมองของ TS-subject จะทราบได้อย่างไรว่าต้องใช้ข้อมูลไหนควรจะใช้ข้อมูลของ C tuple หรือไม่

ตารางที่ 2.4 ข้อมูลผู้ป่วยในรูปแบบโมเดลของจาโจเดีย-ซานดู

Patient_Name	*PC*	*Diagnosis*	*DC*	*Age*	*AC*	*Room*	*RC*	*TC*
Jones	U	Leukemia	C	12	C	202	C	C
Adams	U	Diarrhea	U	21	U	201	U	U

ตารางที่ 2.5 มุมมองของผู้ใช้ในระดับ U ของโมเดลของจาโจเดีย-ซานดู

Patient Name	*PC*	*Diagnosis*	*DC*	*Age*	*AC*	*Room*	*RC*	*TC*
Jones	U	Null	U	Null	U	Null	U	U
Adams	U	Diarrhea	U	21	U	201	U	U

2.3.2 โมเดลเอ็มแอลอาร์

โมเดลเอ็มแอลอาร์ (Multilevel Relation) [5] เป็นเมเดลที่ได้รวมแนวความคิดจากโมเดลซีวิว (Sea-View) ซึ่งเป็นโมเดลแรกที่นำมาใช้ในเชิงสัมพันธ์ และโมเดลแอลดีวี (LDV) เข้าด้วยกันและเพิ่มกฎข้อบังคับการยืมข้อมูลจากระดับต่ำกว่า (data-borrow integrity) ลงไปด้วย ซึ่งเป็นการทำให้ข้อมูลนั้นส่งขึ้นไปสู่ระดับที่สูงกว่าได้โดยอัตโนมัติโดยเป็นข้อมูลที่ระดับสูงนั้นได้ยืมมาจากระดับต่ำกว่าเมื่อมีการแก้ไขข้อมูลนั้นก็จะส่งผลถึงระดับสูงด้วย ในแต่ละข้อมูลนั้น สามารถมองเห็นได้ต่อเมื่อเป็นข้อมูลที่ได้รับอนุญาตให้มองเห็นได้ โดยแนวความคิดในเรื่องมีความคิดเกี่ยวกับการตีความของแต่ละแถวเป็นดังนี้

1. ข้อมูลที่ยอมรับโดยผู้ใช้ที่ระดับหนึ่งนั้น ประกอบด้วยสองส่วนคือ ข้อมูลที่เป็นเจ้าของโดยผู้ใช้เอง และข้อมูลที่ยืมมาจากผู้ใช้อื่นในระดับต่ำกว่า ซึ่งข้อมูลที่ยืมมาจากระดับต่ำกว่านี้สามารถที่จะเปลี่ยนแปลงได้ขึ้นกับผู้ใช้ในระดับต่ำกว่าซึ่งเป็นเจ้าของข้อมูลนั้น
2. ข้อมูลที่ผู้ใช้สามารถเห็นได้นั้นเป็นข้อมูลในระดับของผู้ใช้เอง และในระดับที่ต่ำกว่า
3. แถว c ประกอบไปด้วยข้อมูลที่ได้รับการยอมรับ (ทั้งที่เป็นเจ้าของและยืม) โดยผู้ใช้ระดับ c ซึ่งถ้ามองไม่เห็นข้อมูลของแถว c หมายถึงเอนติตีที่มีอยู่นั้นไม่ได้รับการยอมรับโดยผู้ใช้ในระดับ c

แตกต่างจากโมเดลจาโจเดีย-ซานดู คือไม่มีการถ่ายทอดข้อมูลของตนลงในระดับต่ำกว่า และข้อมูลในระดับตนเองหากมีข้อมูลที่เป็นของระดับต่ำกว่า ต้องเป็นข้อมูลที่มีอยู่ในระดับนั้นจริงๆ อยู่แล้วจึงจะทำการยืมข้อมูลนั้นขึ้นมาใช้กับระดับของตนได้

ตารางที่ 2.6 ข้อมูลผู้ป่วยในรูปแบบโมเดลเอ็มแอลอาร์

Patient_Name	*PC*	*Diagnosis*	*DC*	*Age*	*AC*	*Room*	*RC*	*TC*
Jones	C	Leukemia	C	12	C	209	S	S
Jones	C	Leukemia	C	12	C	202	C	C
Adams	U	Diarrhea	U	21	U	201	U	U

ในโมเดลนี้มีคำสั่งที่ใช้ในการดึงข้อมูลจากระดับล่างขึ้นมาเป็นระดับของตน เรียกว่าคำสั่ง "UPLEVEL" จากตารางที่ 2.6 เห็นได้ว่าผู้ใช้ในระดับ S ได้ทำการใช้คำสั่ง "UPLEVEL" ทำให้เกิดผลในแถวแรกขึ้นมาโดยข้อมูลของผู้ป่วยชื่อ Jones นี้ ได้รับการยืนยันว่าเป็นความจริงโดยผู้ใช้ในระดับ S ในส่วนของ ชื่อผู้ป่วย, กลุ่มโรค และอายุ ยกเว้นหมายเลขห้องนั้นยังไม่ได้รับการยืนยันว่าเป็นความจริงในระดับ S เป็นห้อง 209 ไม่ใช่ 202 นอกจากข้อมูลในส่วนอื่นที่ไม่ใช่หมายเลขห้องและคีย์หลักแอพพาเรนท์แล้ว เมื่อผู้ใช้ในระดับ C ทำการแก้ไขข้อมูลก็จะส่งผลให้เกิดการแก้ไขข้อมูลตามในระดับ S ด้วย อย่างไรก็ตามยังมีจุดด้วยที่ไม่สามารถปกปิดข้อมูลในส่วนที่เป็นคีย์หลักแอพพาเรนท์ได้เพราะเนื่องจากได้ให้คีย์หลักแอพพาเรนท์เป็นตัวระบุเอนติตี

2.3.3 โมเดลจูดิก

โมเดลนี้มีพื้นฐานกับการสรุปความเชื่อ [7] ให้สามารถบ่งบอกได้ทันทีว่ามีความคิดเช่นไรกับข้อมูลจุดเด่นของโมเดลนี้คือมีการนำเซทของริชเชอร์มาใช้เป็นสัญลักษณ์ย่อระดับความปลอดภัย ซึ่งทำให้ลดความซ้ำซ้อนของการเก็บข้อมูลหากต้องการกระจายข้อมูลในหลายระดับ และทั้งยังสามารถตีความหมายของข้อมูลกับข้อมูลที่ระดับต่ำกว่าได้อย่างง่ายดายอีกด้วยเครื่องหมายบอกระดับข้อมูลนั้นเป็นชุดของระดับความปลอดภัย ตัวอย่างเช่น UC-S ซึ่งมีความหมายว่าเป็นจริงยอดรับโดยผู้ใช้ระดับ U และ C และเป็นเท็จที่ผู้ใช้ระดับ S

ได้มีการแยกออกเป็นสองส่วนคือระดับความปลอดภัยหลัก (primary level) และระดับความปลอดภัยรอง (second level)โดย ระดับความปลอดภัยหลักเป็นระดับที่เป็นผู้ทำการเพิ่มข้อมูลลงในฐานข้อมูลในส่วนที่สองหรือในส่วนที่เหลือเป็นระดับความปลอดภัยรองเป็นความเชื่อที่มีต่อแถวนั้นในระดับที่สูงกว่า ถ้ามีเครื่องหมายลบอยู่หมายถึงความเชื่อที่ระดับนั้นเป็นเท็จ ถ้าไม่มีคือเป็นจริง ตัวอย่างเช่น UC-S หมายถึง U และ C เชื่อว่าเป็นจริงและ S เชื่อว่าเป็นเท็จ ส่วน U-CS หมายถึง U เชื่อว่าเป็นจริง C และ S เชื่อว่าเป็นเท็จในตารางที่ 2.7 เป็นตัวอย่างของโมเดลนี้

ตารางที่ 2.7 ข้อมูลผู้ป่วยในรูปแบบโมเดลจูดิก

Patient_Name	***PC***	***Diagnosis***	***DC***	***Age***	***AC***	***Room***	***RC***	***TC***
Smith	CS	Sars	S	56	CS	901	CS	S
Smith	CS	Cold	C-S	56	CS	901	CS	C-S
Jones	C	Leukemia	C	12	C	202	C	C
Adams	U	Diarrhea	U	21	U	201	U	U

แถวในระดับที่ต่ำกว่านั้นได้ถูกตีความโดยระดับที่สูงกว่าดังนี้คือ

- เป็นจริง
- เป็นเท็จ (ได้แก่ข้อมูลปิดบัง หรือไม่เป็นจริง)
- ยังไม่ได้รับการยืนยัน

ที่มุมมองของผู้ใช้ในระดับ S ผู้ป่วยชื่อ Smith ในแถวที่สองนั้นเป็นเท็จซึ่งมีลักษณะของการปิดบังเพียงบางคอลัมน์จึงมีลักษณะของปิดบัง ส่วนแถวที่ไม่เป็นจริงนั้นเป็นกรณีของแถวที่ได้รับการยืนยันว่าไม่เป็นจริง และไม่มีส่วนเกี่ยวข้องด้วยเลยกับเอนติตีในโลกความเป็นจริงเลย ในความเชื่อของ ผู้ใช้ในระดับสูง ถึงอย่างไรก็ตามแม้ว่าจะมีการกำหนดลักษณะของความเชื่อเกี่ยวกับแถวในระดับที่ต่ำกว่าแต่ยังไม่สามารถบอกได้ว่าอันไหนเป็นข้อเท็จ หรือในกรณีของผู้ใช้ระดับสูงนั้นจะทราบได้อย่างไรว่าข้อมูลไหนควรได้รับการยืนยันว่าเป็นความจริง และยังไม่สามารถปกปิดในส่วนที่เป็นคีย์หลักแอพพาเรนท์ได้

2.4 โมเดลเอ็มเอสบีดีในส่วนของเอ็มแอลเอส

ในส่วนนี้กล่าวถึงเฉพาะส่วนข้อมูลที่มีความปลอดภัยหลายระดับชั้นเท่านั้นในโมเดลเอ็มเอสบีดี โดยยังไม่กล่าวถึงคุณสมบัติแปรเปลี่ยนตามเวลาในส่วนนี้

2.4.1 โครงสร้าง

ในโมเดลได้ได้ยึดพื้นฐานจากโมเดลเอ็มแอลอาร์ และได้เพิ่มคุณสมบัติบางอย่างเพื่อเข้าไปโดยมีโครงสร้างพื้นฐานดังต่อไปนี้

$R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC, CC)$

A_i คือ แอททริบิวข้อมูล (Data Attribute) เช่นเดียวกับข้อมูลในฐานข้อมูลเชิงสัมพันธ์แบบปกติ

C_i คือ ระดับของข้อมูล (Classification Label) สำหรับ A_i

TC คือ ระดับความปลอดภัยในระดับแถว (Tuple Class)

สามแอททริบิวที่กล่าวมาเป็นสามแอททริบิวที่ได้ใช้ในเอ็มแอลอาร์สำหรับ CC (Creator Class) ได้กำหนดขึ้นมาใหม่ ข้อจำกัดของโมเดลเอ็มแอลอาร์คือไม่สามารถดึงข้อมูลจากระดับต่ำกว่าได้พร้อมกันมากกว่าหนึ่งหากเอนติตีนั้นไม่ได้มีระดับพื้นฐานเดียวกัน เราขอยกตัวอย่างประกอบกำหนดงานติดตามบุคคลของสำนักงานนักสืบแห่งหนึ่งที่มีการเก็บข้อมูลของผู้ถูกติดตามประกอบด้วยการกระทำและสถานที่ มีโครงสร้างอย่างคร่าวดังนี้ TRACKING_REPORTS(TNAME, ACTION, LOCAT) โครงสร้างตารางนี้ ประกอบด้วยชื่อของผู้ถูกติดตาม TNAME ซึ่งย่อมากจาก "Target Name" และเป็นคีย์หลัก, การกระทำในขณะนั้น ACTION และ LOCAT ซึ่งย่อจาก LOCATION เป็นสถานที่ที่ผู้ถูกติดตามอยู่ในขณะนั้น ตารางนี้เป็นตารางที่อ้างอิงถึงข้อมูลในปัจจุบัน และมีการเปลี่ยนแปลงข้อมูลเสมอเมื่อข้อมูลของเป้าหมายในปัจจุบันเปลี่ยนแปลงไป เมื่อนำมาปรับปรุงให้เป็นเอ็มแอลเอสแล้วได้โครงสร้างดังต่อไปนี้

TRACKING_REPORTS(TNAME, TN_C, ACTION, AT_C, LOCAT, LC_C, TC, CC)

แอททริบิว TN_C, AT_C, LC_C เป็นการกำหนดระดับให้กับแอททริบิว TNAME, ACTION และ LOCAT ตามลำดับ แอททริบิว TNAME ถ้าอยู่ในตารางเชิงสัมพันธ์แบบปกติเป็นคีย์หลัก แต่เมื่ออยู่ปรับมาใช้ในเอ็มแอลเอสจถูกเรียกว่าคีย์หลักแอพพาเรนท์ (Apparent Primary Key) สำหรับคีย์หลักจริงๆ ของโมเดลนี้ กล่าวได้ว่า A_k, TC, CC $\rightarrow$ A_i, C_i ตัวอย่างข้อมูลในตารางที่ 2.8 เป้าหมายมีเพียงเป้าหมายเดียวที่ต้องการติดตามคือ JOHN CONNER ประกอบด้วยกันสองแถวแต่ต่างระดับกัน นักสืบระดับ C1 และ C2 มีความเชื่อต่อข้อมูลแวดล้อมของ JOHN CONNER แตกต่างกัน

ตารางที่ 2.8 ตัวอย่างข้อมูลในตาราง TRACKING_REPORTS

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*TC*	*CC*
JOHN CONNER	C1	phone	C1	Airport	C1	C1	C1
JOHN CONNER	C2	meeting someone	C2	S&P Restaurant	C2	C2	C2

2.4.2 การปรับปรุงข้อมูล

การปรับปรุงข้อมูลภายภายใต้โมเดลเอ็มแอลเอส เราได้ใช้คำสั่งเอสคิวแอลมาตรฐานมาดัดแปลงดังนั้นในหนึ่งคำสั่งปรับปรุงข้อมูลอาจประกอบด้วยคำสั่งเอสคิวแอลมาตรฐานหลายคำสั่ง ปกติแล้วการข้อมูลอื่นนอกเหนือจากแอททริบิวข้อมูลจะถูกกำหนดโดยระบบไม่สามารถกำหนดได้เอง แต่ในอนุประโยค WHERE สามารถมีชุดข้อมูลเหล่านี้รวมอยู่ด้วยได้

2.4.2.1 คำสั่ง INSERT

กำหนดให้แถว $t \in r$ การเพิ่มข้อมูลลงในตารางเอ็มแอลเอสจะต้องมี t[CC] เท่ากับระดับของตนเองนั่นหมายความถึงระดับผู้ที่ใช้คำสั่ง INSERT และสามารถกำหนดได้ว่าต้องการเพิ่มแถวให้กับระดับใดก็ได้เฉพาะในระดับตนเองและระดับต่ำกว่า ถ้าหากไม่ระบุข้อมูลนั้นเป็นรายงานในระดับของตนคือ t[TC] เท่ากับระดับของตน

ในกรณีที่มีการกำหนดแถวนั้นมองเห็นได้ในระดับต่ำกว่า ระดับของแอททริบิวข้อมูลและ t[TC] จะเท่ากับระดับข้อผู้ออกคำสั่ง INSERT เพื่อให้เป็นข้อมูลของระดับนั้นและสามารถมองเห็นได้ แต่ยังคงระบุ t[CC] ว่าระดับใดเป็นผู้ส่งคำสั่งนี้ลงมาให้ สำหรับตัวอย่างติดตามบุคคล นักสืบในระดับ C2 ทำการติดตาม JOHN CONNER ที่นัดพบกับใครสักคนที่ร้านเอสแอนท์พี นักสืบทำการออกคำสั่ง

```
INSERT INTO TRACKING_REPORTS
VALUES ( 'JOHN CONNER', 'meeting someone', 'S&P Restaurant')
LEVELS 'ME'
```

เนื่องการจากเพิ่มข้อมูลเป็นในลักษณะรายงาน จึงมีการกำหนดข้อมูลที่เพิ่มเข้าไปให้เป็นข้อมูลนระดับของตนเอง แต่แทนที่จะให้คำสั่ง INSERT ดังข้างตนให้ระบบทำการเรียกคำสั่งต่อไปนี้มาแทน

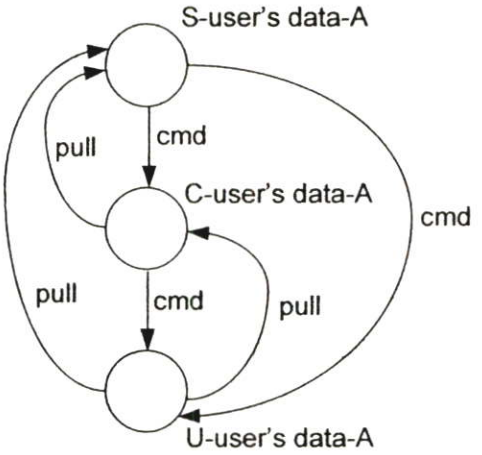
```
INSERT INTO TRACKING_REPORTS
VALUES ( 'JOHN CONNER', 'C2', 'meeting someone', 'C2', 'S&P Restaurant', 'C2', 'C2', 'C2')
```

ระดับของแอททริบิว , t[TC] และ t[CC] จะเท่ากับระดับของนักสืบระดับ C2 ผู้ซึ่งทำการเพิ่มแถวเข้าไป ผลลัพท์ของคำสั่ง INSERT นี้แสดงในตารางที่ 2.8 ในแถวที่สอง

2.4.2.2 คำสั่ง UPLEVEL

ผู้ใช้ในระดับที่สูงกว่าเมื่อมีความเชื่อต่อข้อมูลในระดับล่างว่าเป็นจริง มีแนวทางการนำข้อมูลนั้นเป็นข้อมูลในระดับตนเองได้สองแนวทาง วิธีการแรกคือ ใช้คำสั่ง INSERT หรือคำสั่ง UPDATE ทำการปรับข้อมูลในระดับของตน วิธีที่สองใช้คำสั่ง UPLEVEL เป็นให้การยอมรับความเชื่อถือต่อผู้ใช้ในระดับต่ำว่านั่นหมายความว่าข้อมูลส่วนใดที่เรายอมรับสามารถเปลี่ยนแปลงได้ตามระดับล่างที่เราให้ความเชื่อถือ เป็นการนำข้อมูลในระดับต่ำกว่ามาเป็นข้อมูลในระดับตน หรือหมายถึงนำมายืนยันว่าเป็นจริงในระดับของตน ในรูปที่ 2.1 แสดงถึงคำสั่ง INSERT สามารถออกคำสั่งลงในระดับต่ำกว่าได้โดยมี "cmd" กำกับลูกศร และคำสั่ง UPLEVEL สามารถดึงข้อมูลจากระดับล่างเป็นข้อมูลในระดับของตนโดยมี "pull" กำกับลูกศร

การให้คำสั่งรายงานแก่ระดับล่าง ปกติแล้วตามกฎเกณฑ์ของเบล และพาลาดูลาที่จะกล่าวในบทที่ 2 ข้อสำคัญคือความสามารถทางการเข้าถึงข้อมูลแบบเขียนได้ต้องเป็นระดับตนเองหรือสูงกว่าเท่านั้น ในระดับสูงกว่าในที่นี้ ไม่ได้เป็นการเขียนโดยตรง และจะไม่มีการรับรู้จากในระดับล่างเลยว่าส่งผลกระทบต่อระดับบน หากแต่ระดับบนยินยอมให้ข้อมูลดังกล่าวแก้ไขโดยอัตโนมัติด้วยระดับล่างเองเท่านั้น ในรูปที่ 1.1 เป็นการอธิบายถึงระดับผู้ใช้ที่มีความแตกต่างของแต่ละระดับเป็น S > C > U เปรียบเสมือนระดับ S เป็นเจ้านายสูงสุดจากทั้งสองระดับ สามมารถออกคำสั่ง (cmd) ได้ทั้งสองระดับ หรือเลือกที่จะดึงข้อมูล (pull) จากระดับ C และ U ได้ การออกคำสั่งนี้เองเป็นการเพิ่มความสามารถในการเขียนลงในระดับต่ำกว่า โดยใช้แนวความคิดแยกกลุ่มของข้อมูลมาอีกหนึ่งชุดคือ คำสั่ง เป็นชุดข้อมูลที่ส่งให้ระดับล่างเพื่อสื่อสารบอกข้อมูลแก่ระดับล่างได้ แต่ไม่สามารถแก้ไขข้อมูลใดๆ ที่เป็นรายงานของระดับต่ำกว่าได้ ในตารางที่ 1.6 มีรูปแบบโครงสร้างที่ใช้ในงานวิจัยของเราที่มีทั้งคุณสมบัติของเวลาและมีหลายระดับความปลอดภัยโดยตารางนี้มีพื้นฐานโครงสร้างข้อมูลจากตารางเอสโอดี ระดับ S ได้ส่งคำสั่งให้ระดับ U โดยให้ทำการเพิ่มคำสั่งนี้ไปเมื่อ วันที่ 6 กันยายน ให้ปฏิบัติการวันที่ 10 ข้อมูลนี้สามารถมองเห็นได้ในระดับของและได้มีผู้ที่ออกคำสั่งคือ S ในคอลัมน์ CC และบอกว่าเป็นข้อมูลให้แก U ในคอลัมน์ TC นอกเหนือจากนั้นยังสามารถกำหนดขอบเขตของเวลาที่ออกคำสั่งรวมการรายงานกลับด้วย



รูปที่ 2.1 ความสามารถในการจัดการออกคำสั่ง และรายงาน ระหว่างระดับความปลอดภัย

นักสืบในระดับ S1 มองเห็นข้อมูลในตาราง 2.8 ได้ทั้งหมด และพิจารณาข้อมูลของนักสืบระดับ C1 และ C2 ที่มีต่อข้อมูลเป้าหมาย JOHN CONNER นักสืบในระดับ S1 ให้ความซื่อถือเรื่องการกระทำจากระดับ C1 และเชื่อข้อมูลสถานที่จาก C2 ความพิเศษของโมเดลเอ็มเอสบีดีคือการให้ความเชื่อกับเอนติตีจากระดับสองระดับคือ JOHN CONNER ได้ โดยกำกับระดับที่ทำการดึงข้อมูลไว้ในเซทของระดับของคีย์หลักแอพพาเรนท์ในที่นี้คือ t[TN_C] นักสืบระดับ S1 ออกคำสั่งเป็นเอสคิวแอลดังนี้

```
UPLEVEL  TRACKING_REPORTS
GET ACTION FROM C1, LOCAT FROM C2
WHERE TNAME = 'JOHN CONNER'
```

ผลของคำสั่ง UPLEVEL เป็นดังตารางที่ 2.9

ตารางที่ 2.9 ตัวอย่างข้อมูลในตาราง TRACKING_REPORTS หลังคำสั่ง UPLEVEL

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*TC*	*CC*
JOHN CONNER	C1, C2	phone	C1	S&P Restaurant	C2	S1	S1
JOHN CONNER	C1	phone	C1	Airport	C1	C1	C1
JOHN CONNER	C2	meeting someone	C2	S&P Restaurant	C2	C2	C2

แถวที่เกิดขึ้นจากคำสั่ง UPLEVEL คือแถวบนโดยมีเป้าหมาย JOHN CONNER จากสองระดับ ดังนั้น t[TN_C] จึงประกอบด้วยระดับสองระดับ คือ C1 และ C2 เป็นการลดความกำกวมจากโมเดลเอ็มแอลอาร์ที่เดิมทีหากมีการดึงจากสองระดับ C1, C2 อาจมี JOHN CONNER ได้สองแถวที่

ต่าง t[TN_C] กัน ทำให้เกิดความสับสนสำหรับข้อมูลของ JOHN CONNER ได้การปรับลดทอนของเอ็มเอสบีดีเป็น Ak, TC → Ai, Ci ซึ่งได้ตัด C1 ออกจากกลุ่มแอททริบิวทางด้านซ้าย เป็นการกำหนดให้เอนติตีหนึ่งสามารถมีได้เพียงหนึ่งเดียวในระดับหนึ่งๆ เท่านั้น และเพิ่มให้ระดับความปลอดภัยของคีย์ในฐานข้อมูลเอ็มแอลเอส เป็นเซทของระดับความปลอดภัยที่อ้างอิงถึง พิจารณาตารางที่ 2.9 เมื่อระดับ S ต้องการดึงข้อมูลจากระดับ C1 และ C2 ในสองแถวล่าง ผลที่ได้ออกมาในแถวบน การทำแบบนี้เพื่อยังรักษาคุณสมบัติเดิมของความต้องการการแพร่กระจายข้อมูลอัตโนมัติจากระดับต่ำกว่า คือเมื่อข้อมูลของ JOHN CONNER ในระดับ C1 และ C2 ถูกลบส่งผลให้แถวในระดับ S1 ถูกลบตามไปด้วยเนื่องจากเอนติตีนี้ไม่มีต่อไปแล้ว

ในการใช้งานคำสั่ง UPLEVEL กับฐานข้อมูลเชิงสัมพันธ์ปกติ เราสามารถเรียกใช้ผ่านโพรซีเจอร์ สำหรับคำสั่ง

```
UPLEVEL  TRACKING_REPORTS
GET ACION  FROM C1
WHERE TNAME = 'JOHN CONNER'
```

ประกอบด้วยชุดคำสั่งเอสคิวแอลดังต่อไปนี้

```
UPDATE TRACKING_REPORTS
SET TN_C = ADD_LV( TN_C, 'C1'),  ACTION =
  (SELECT ACTION
    FROM TRACKING_REPORTS
    WHERE TNAME = 'JOHN CONNER' AND AT_C = 'C1' AND TC = 'C1' AND CC = 'C1')
WHERE TNAME = 'JOHN CONNER'  AND TC = 'S1' AND CC = 'S1'

INSERT INTO TRACKING_REPORTS
SELECT TNAME, TN_C, ACTION, AT_C, null, 'S1', 'S1', 'S1'
FROM TRACKING_REPORTS
WHERE TNAME = 'JOHN CONNER' AND AT_C = 'C1'
  AND TC = 'C1' AND CC = 'C1'
  AND NOT EXISTS( SELECT TNAME
                    FROM TRACKING_REPORTS
                    WHERE TNAME = 'JOHN CONNER'  AND TC = 'S1' AND CC = 'S1');
```

ในคำสั่ง UPDATE เป็นชุดคำสั่งสำหรับกรณีมีข้อมูลของ JOHN CONNER อยู่แล้วในระดับ S1 ให้ทำการปรับข้อมูลเดิมที่มีอยู่ ด้วยการเพิ่มระดับ C1 ซึ่งเป็นระดับที่ต้องการดึงขึ้นมาลงใน t[TN_C] โดยการใช้ฟังก์ชัน ADD_LV เป็นฟังก์ชันทำการรวมระดับความปลอดภัยเข้าไป ตัวอย่างเช่น ADD_LV('C1', 'C2') ผลลัพธ์รีเทิร์นประโยค "C1,C2" คำสั่ง INSERT เป็นชุดคำสั่งสำหรับกรณีไม่มีข้อมูลของ JOHN CONNER อยู่ในระดับ S1 ให้ทำการเพิ่งแถวใหม่เข้าไป สำหรับแอททริบิวอื่นที่ไม่ได้อยู่ในอนุประโยค GET ให้เป็น null โดยมีระดับเท่ากับ S1 ซึ่งเป็นผู้ออกคำสั่ง

2.4.2.3 คำสั่ง UPDATE

ผู้ใช้สามารถออกคำสั่ง UPDATE ในส่วนเฉพาะแถวที่ผู้ใช้เป็นผู้สร้างขึ้นมาเองเท่านั้นบ่งบอกโดย t[CC] นั่นหมายความว่าสามารถแก้ไขได้ทั้งคำสั่ง และรายงานที่เป็นผู้สร้างขึ้น ถ้านักสืบต้องการแก้ไขข้อมูลในตาราง TRACKING_REPORTS อาจมีข้อมูลบางส่วนในระดับสูงกว่าโดนอัพเดทตามไปด้วยโดยที่ผู้ออกคำสั่งไม่สามารถรู้ได้ จากข้อมูลในตารางที่ 2.9 เมื่อนักสืบระดับ C2 ทราบมาว่าข้อมูลของตนผิดในส่วนของสถานที่จาก "S&P Restaurant" ต้องการเปลี่ยนเป็น "Siam Paragon" สามารถทำได้โดยออกคำสั่งดังต่อไปนี้

```
UPDATE TRACKING_REPORTS
SET LOCAT = 'Siam Paragon'
WHERE TNAME = 'JOHN CONNER'
```

ตารางที่ 2.10 ตัวอย่างข้อมูลในตาราง TRACKING_REPORTS หลังคำสั่ง UPDATE

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*TC*	*CC*
JOHN CONNER	C1, C2	phone	C1	Siam Paragon	C2	S1	S1
JOHN CONNER	C1	phone	C1	Airport	C1	C1	C1
JOHN CONNER	C2	meeting someone	C2	Siam Paragon	C2	C2	C2

การเปลี่ยนแปลงสถานที่ของผู้ถูกติดตามในระดับ C2 นอกเหนือจากเปลี่ยนแปลงในมุมมองของผู้ใช้ C2 แล้วยังส่งผลต่อระดับ S1 ด้วยเนื่องจากผู้ใช้ระดับ S1 ได้ให้ความไว้วางใจสำหรับข้อมูลสถานที่สำหรับ JOHN CONNER ว่าผู้ใช้ระดับ C2 นั้นเชื่อถือได้ในการพัฒนาด้วยฐานข้อมูลแบบปกตินั้นต้องใช้สองคำสั่ง UPDATE ดังต่อไปนี้

```
UPDATE TRACKING_REPORTS
SET LOCAT = 'Siam Paragon', LT_C = 'C2', TN_C = ADD_LV(TN_C, 'C2')
WHERE TNAME = 'JOHN CONNER' AND TC = 'C2' AND CC = 'C2'

UPDATE CRIMINALS
SET LOCAT = 'Siam Paragon'
WHERE TNAME = 'JOHN CONNER' AND TO_NUM(TC) > TO_NUM('S')
  AND LT_C = 'C2'
```

ในคำสั่ง UPDATE แรกเป็นการเปลี่ยนแปลงข้อมูลสถานที่สำหรับข้อมูลในระดับ C2 เอง และเนื่องจากข้อมูลนี้เป็นข้อมูลที่ถูกกำหนดขึ้นในระดับของตนเองข้อมูลของ JOHN CONNER ต้องเป็นข้อมูลที่เกิดจากระดับ C2 เองด้วยดังนั้น ต้องทำการเพิ่มระดับ C2 ลงใน t[TN_C] หากยังไม่ได้กำหนดไว้ สำหรับคำสั่ง UPDATE ที่สองนั้นเป็นการแก้ไขในระดับที่สูงกว่าในที่นี้คือระดับ S1 ซึ่งได้มีการดึงข้อมูลสถานที่จากระดับ C2 ไป มีการใช้ฟังก์ชัน TO_NUM ซึ่งเป็นการเปลี่ยนจากอักษร

ระดับไปเป็นตัวเลขเพื่อให้ใช้การเปรียบเทียบได้ สำหรับผลจากการแก้ไขข้อมูลในครั้งนี้เป็นดังตารางที่ 2.10

ถ้าหากมีการเปลี่ยนแปลงชื่อของ JOHN CONNER ซึ่งเป็นคีย์หลักแอพพาเรนท์แล้วเปรียบเหมือนว่าข้อมูลของ JOHN CONNER ไม่ได้มีอีกต่อไปแล้ว ดังนั้นหากมีระดับที่สูงกว่ามีการดึงข้อมูลนี้ไป มีสองกรณีที่สามารถเกิดขึ้นได้ จากตารางที่ 2.9 ถ้าหากนักสืบในระดับ C2 เปลี่ยนชื่อ JOHN CONNER เป็น JOHN LENNON ด้วยคำสั่งเอสคิวแอลของโมเดลเอ็มเอสบีดีดังนี้

```
UPDATE TRACKING_REPORTS
SET TNAME = 'JOHN LENNON'
WHERE TNAME = 'JOHN CONNER'
```

ผลลัพธ์จากการออกคำสั่งนี้แสดงในตาราง 2.11 ในแถวสุดท้าย ข้อมูลของชื่อเป้าหมายของระดับ C2 นั้นเป็นไปเป็น JOHN LENNON แต่เนื่องจากเดิมนักสืบระดับ S1 ได้ดึงข้อมูลสถานที่ของ JOHN CONNER มาจาก C2 เมื่อข้อมูลของ JOHN CONNER ไม่มีอยู่อีกต่อไปแล้ว ข้อมูลสถานที่ต้องถูกระบุเป็น "null" และระดับที่กำหนดใน t[TN_C] ต้องลบระดับ C2 ออกไปด้วย เนื่องจาก JOHN CONNER ไม่ได้มากจาก C2 อีกแต่ไปแล้วมีเพียง C1 เท่านั้น ถ้าหากข้อมูล JOHN CONNER ในระดับ C1 ได้หายไปเนื่องจากการลบหรือการแก้ไขอีก ข้อมูลของ JOHN CONNER ในระดับ S1 ต้องถูกลบเนื่องจากไม่มีข้อมูลนี้อยู่จริงในระดับล่างแล้ว

ตารางที่ 2.11 ตัวอย่างข้อมูลในตาราง TRACKING_REPORTS หลังคำสั่ง UPDATE

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*TC*	*CC*
JOHN CONNER	C1	phone	C1	null	C2	S1	S1
JOHN CONNER	C1	phone	C1	Airport	C1	C1	C1
JOHN LENNON	C2	meeting someone	C2	S&P Restaurant	C2	C2	C2

คำสั่งเอสคิวแอลของโมเดลเอ็มเอสบีดีด้านบนนั้นนำไปพัฒนาใช้กับเอสคิวแอลมาตรฐานประกอบด้วยหนึ่งคำสั่ง DELETE และสี่คำสั่ง UPDATE ฟังก์ชันที่ถูกสร้างขึ้นเพื่อใช้รองรับคำสั่งเหล่านี้คือ RM_LV เป็นฟังก์ชันใช้ลบระดับความปลอดภัยออกจากชุดระดับความปลอดภัยใน t[TN_C] ตัวอย่างเช่น RM_LV('C2,S1', 'C2') หมายถึงทำการลบระดับความปลอดภัยจากอินพุททางขวาออกจากอินพุดทางซ้าย จะรีเทิร์นประโยคผลลัพธ์ออกมาเป็น 'S1' และอีกฟังก์ชันหนึ่งคือ INLB ตรวจสอบว่ามีระดับความปลอดภัยอยู่ในชุดความปลอดภัยหรือไม่ ตัวอย่างเช่น INLB('C2,S1', 'C2') ตรวจว่ามีระดับความปลอดภัยจากอินพุททางขวาอยู่ในอินพุททางซ้ายหรือไม่ถ้ามีรีเทิร์นผลลัพธ์ 1

```
UPDATE TRACKING_REPORTS
SET TNAME = 'JOHN LENNON', TN_C = 'C2'
WHERE TNAME = 'JOHN CONNER' AND TC = 'C2' AND CC = 'C2'

DELETE FROM TRACKING_REPORTS
WHERE TNAME = 'JOHN CONNER' AND
 AND TO_NUM( TC) > TO_NUM( 'C2') and TN_C = 'C2'

UPDATE TRACKING_REPORTS
SET TN_C = RM_LV(TN_C, 'C2')
WHERE TNAME = 'JOHN CONNER'
 AND LNUM( TC) > TO_NUM( 'C2') and INLB( TN_C, 'C2') = 1

UPDATE TRACKING_REPORTS
SET ACION = null
WHERE TNAME = 'JOHN_CONNER'
  AND TO_NUM( TC) > TO_NUM( 'C2')
  AND AT_C = 'C2'

UPDATE TRACKING_REPORTS
SET LOCAT  = null
WHERE TNAME = 'JOHN_CONNER'
  AND TO_NUM( TC) > TO_NUM( 'C2')
  AND LC_C = 'C2'
```

ในชุดคำสั่งทั้งหมดนี้สืบเนื่องจากมีการแก้ไขข้อมูลในส่วนของชื่อเป้าหมายซึ่งเป็นคีย์หลังแอพพาเรนท์ ในอัพเดทแรกเป็นการแก้ไขชื่อเป็น JOHN LENNON ในระดับของตอนเองสำหรับ คำสั่ง DELETE และอีกสามคำสั่ง UPDATE ถัดมาเป็นการลบความเชื่อที่มีต่อ JOHN CONNER สำหรับระดับ C2 ออกทั้งหมดจากระดับที่สูงกว่า ในคำสั่ง DELETE อยู่ในกรณีที่ระดับที่สูงกว่าสร้างแถวที่เกิดจากการดึงข้อมูลเฉพาะ JOHN CONNER จาก C2 เพียงระดับเดียวนั่นคือมี t[TN_C] = C2 ดังนั้นเมื่อ JOHN CONNER ในระดับ C2 ไม่มีอีกต่อไปแล้วจึงต้องทำการลบแถวในระดับสูงกว่าเสีย คำสั่ง UPDATE ถัดมาเป็นการลบความเชื่อเดิมของจากชุดระดับความปลอดภัย t[TN_C] ด้วยการใช้ฟังก์ชัน RM_LV เพื่อลบระดับ C2 ออกไป ในสองคำสั่ง UPDATE สุดท้ายสำหรับแอททริบิวอื่นๆ นองกจากคีย์หลักแอพพาเรนท์ต้องถูกเปลี่ยนเป็น null

2.4.2.4 คำสั่ง DELETE

คำสั่ง DELETE มีส่วนคล้ายคำสั่ง UPDATE คือนอกจากลบความเชื่อในระดับของผู้ออกคำสั่งแล้วยังส่งผลกระทบต่อระดับสูงกว่าที่มีการดึงข้อมูลจากระดับนั้นไปด้วย สำหรับตัวอย่างจากข้อมูลในตารางที่ 2.9 ถ้าหากนักสืบในระดับ C2 ทำการลบข้อมูล JOHN CONNER ออกหรือเขียนด้วยเอสคิวแอลของโมเดลเอ็มเอสบีดีดังนี้

```
DELETE FROM TRACKING_REPORTS
WHERE TNAME = 'JOHN CONNER'
```

ข้อมูลของเป้าหมาย JOHN CONNER ในระดับ C2 ถูกลบออกไป ทำให้แถวบนสุดในตาราง 2.9 ที่มีการดึงข้อมูลจากระดับ C2 ต้องมีผลกระทบไปด้วยผลลัพธ์หลังการใช้คำสั่ง DELETE นี้ดังตารางที่ 2.12

ตารางที่ 2.12 ตัวอย่างข้อมูลในตาราง TRACKING_REPORTS หลังคำสั่ง UPDATE

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*TC*	*CC*
JOHN CONNER	C1	phone	C1	null	C2	S1	S1
JOHN CONNER	C1	phone	C1	Airport	C1	C1	C1

t[TN_C] ถูกตัด C2 ออกเนื่องจากแถวในระดับ S1 นี้ไม่มีข้อมูลที่มาจากระดับ C2 ร่วมอยู่ด้วยแล้วเป็นผลมาจากคำสั่ง DELETE เช่นเดียวกันข้อมูลสถานที่ที่ดึงมาจาก C2 ต้องถูกเปลี่ยนเป็น null คำสั่งเอสคิวแอลมาตรฐานจากคำสั่ง DELETE ด้านบนซึ่งคล้ายกับคำสั่ง UPDATE ที่ผ่านมาเพียงแต่ในคำสั่งแรกเป็นการลบข้อมูลในระดับ C2 แทนที่จะเป็นการแก้ไข

```
DELETE FROM TRACKING_REPORTS
 WHERE TNAME = 'JOHN CONNER'  AND TC = 'C2' AND CC = 'C2'

 DELETE FROM TRACKING_REPORTS
 WHERE TNAME = 'JOHN CONNER'
  AND TO_NUM( TC) > TO_NUM( 'C2') AND TN_C = 'C2'

 UPDATE TRACKING_REPORTS
 SET TN_C = RM_LV(TN_C, 'C2')
 WHERE TNAME = 'JOHN CONNER'
  AND LNUM( TC) > TO_NUM( 'C2') and INLB( TN_C, 'C2') = 1

 UPDATE TRACKING_REPORTS
 SET ACION = null
 WHERE TNAME = 'JOHN_CONNER'
   AND TO_NUM( TC) > TO_NUM( 'C2')
   AND AT_C = 'C2'

 UPDATE TRACKING_REPORTS
 SET LOCAT  = null
 WHERE TNAME = 'JOHN_CONNER'
   AND TO_NUM( TC) > TO_NUM( 'C2')
   AND LC_C = 'C2'
```

สิ่งคำสั่งล่างเป็นการจัดการกับข้อมูลที่มีการดึงข้อมูลจาก C2 ได้แก่ในระดับ S1 ในกรณีที่เป็นแถวในระดับ S1 เป็นข้อมูลที่สร้างขึ้นจากข้อมูลเป้าหมาย JOHN CONNER ในระดับ C2 เพียงอย่างเดียวต้องถูกลบไปแต่สำหรับตัวอย่างในตารางที่ 2.9 นี้ยังมีข้อมูลจาก C1 อีกหนึ่งระดับดังนั้น คำสั่ง

นี้ไม่มีผลอะไรเกิดขึ้น แต่จะมีผลต่อคำสั่ง UPDATE ถัดมาคือให้ลบความเชื่อของ JOHN CONNER ในระดับ C2 จาก t[TN_C] ออก และสองคำสั่ง UPDATE สุดท้ายสำหรับแอททริบิวข้อมูลอื่นที่มีการดึงข้อมูลมาจาก C2 นี้ต้องถูกปรับเป็น null

บทที่ 3
ฐานข้อมูลที่แปรเปลี่ยนตามเวลา

ข้อมูลเชิงเวลาคือการจัดการในเอสคิวแอลมีทั้งในทางความหมายของข้อมูลเวลาที่อ้างอิงในเวลาปัจจุบัน (Current) หรือเป็นลำดับเหตุการณ์ (Sequence) หรือไม่สนใจเวลา (Nonsequence) ทั้งหมดยังแสดงการอ้างถึงเวลาได้ด้วยเวลาวาลิด (Valid Time) และ เวลาทรานเซคชัน (Transaction Time) ความท้าท้ายสำหรับข้อมูลเช่นนี้คือการแก้ไขปรับปรุงข้อมูลเชิงเวลาและการป้องกันการซ้ำซ้อนของข้อมูลเชิงเวลาให้สามารถใช้ได้โดยไม่เกิดความกำกวมของข้อมูล ในบทนี้เราได้อธิบายถึงข้อมูลที่แปรเปลี่ยนตามเวลาด้วยฐานข้อมูลเชิงสัมพันธ์แบบปกติ

3.1 การป้องกันการเกิดการซ้ำซ้อนของข้อมูล

ตัวอย่างที่ใช้ในการประกอบการอธิบายเป็นข้อมูลของโรงพยาบาลเด็กที่ประกอบด้วยตารางบันทึกสถานะของผู้ป่วยเด็กมีชื่อตารางว่า "NICUStatus" ได้ตัดทอนแล้วดังตารางที่ 3.1 ในตารางเชิงเวลานี้ และได้มีการต่อเติมคอลัมน์ในแต่ละแถว ที่แสดงถึงเวลาของสถานะของผู้ป่วยเด็กเวลาเอาไว้ด้วย ได้แก่ คอลัมน์ "from_date" แสดงถึงวันที่เด็กเริ่มมีสถานะนั้นเป็นครั้งแรก ส่วนในคอลัมน์ "to_date" แสดงถึงวันที่เด็กได้มีสถานะเปลี่ยนไป ลักษณะของการการกำหนดคอลัมน์เช่นนี้เป็นลักษณะของ "คาบเวลา" ภายใต้สถานะที่ยังคงเป็นจริงอยู่ (Valid) แสดงถึงคาบเวลาในรูปแบบวันเวลาเริ่ม และวันเวลาสิ้นสุด กำหนดให้วันเวลาเริ่มรวมอยู่ในคาบเวลาด้วย ในขณะที่วันเวลาสิ้นสุดนั้นไม่ได้รวมอยู่ในคาบเวลาด้วย คาบเวลายังสามารถกำหนดในรูปแบบอื่นได้อีก การกำหนดแถวข้อมูลเป็นจริงอยู่เรื่อยไปจนถึงปัจจุบันทำได้โดยการกำหนด "to_date" ให้เวลาที่ไม่สิ้นสุด ซึ่งในเอสคิวแอล-92 สามารถกำหนดได้มากที่สุดคือ 9999-12-31 อีกวิธีหนึ่งสามามารถทำได้กำหนดค่านัลให้ถือว่าเป็นปัจจุบัน หรืออาจจะกำหนดให้เป็นเวลาที่ผ่านมาแล้ว ลักษณะเหล่านี้เป็นการกำหนดคาบเวลาสำหรับตารางเวลาวาลิดที่การบันทึกเวลาในรูปแบบของคาบเวลาที่ทำให้ข้อมูลเป็นจริง (period of validity)

ตารางที่ 3.1 ข้อมูลบางส่วนจากตารางผู้ป่วยเด็ก

Name	*Status*	*from_date*	*to_date*
Kenneth Robert	serious	1997-11-19	1997-11-21
Alexis May	serious	1997-11-19	1997-11-27
Natalie Sue	serious	1997-11-19	1997-11-25
Kelsey Ann	serious	1997-11-19	1997-11-26
Brandon James	serious	1997-11-19	1997-11-26
Nathan Roy	serious	1997-11-19	1997-11-28
Joel Steven	critical	1997-11-19	1997-11-20
Joel Steven	serious	1997-11-20	1997-11-26
Kenneth Robert	fair	1997-11-21	1998-01-03
Alexis May	fair	1997-11-27	1998-01-11
Alexis May	fair	1997-12-02	9999-12-31
Alexis May	fair	1997-12-02	9999-12-31

ตารางที่ 3.2 ข้อมูลเวลาปัจจุบันของตารางผู้ป่วยเด็ก

Name	*Status*
Alexis May	fair
Alexis May	fair
Alexis May	fair

ความซ้ำซ้อนที่เกิดจากข้อมูลตรงกันคอลัมน์ต่อคอลัมน์ กับแถวอื่นๆ การซ้ำซ้อนแบบนี้เป็นลักษณะไม่คำนึงถึงเวลา (Nonsequenced Duplicate) จากตารางที่ 3.1 สองแถวสุดท้ายของตารางเป็นการซ้ำซ้อนแบบไม่คำนึงถึงเวลาคือทุกคอลัมน์มีข้อมูลที่ตรงกันทั้งหมด และยังมีสามลักษณะของการซ้ำซ้อนของข้อมูลอยู่ในตารางนี้ด้วย

ในสามแถวสุดท้ายเป็นมีข้อมูลตรงกัน (Value-Equivalent) ยกเว้นในส่วนของการบันทึกเวลา ข้อมูลเหล่านี้ตรงกันแบบไม่คำถึงการแปรเปลี่ยนตามเวลา ซึ่งก็คือสองคอลัมน์ "Name" และ "Status"

สามแถวสุดท้ายยังเป็นการซ้ำซ้อนในเวลาปัจจุบัน ที่อ้างอิงด้วยเวลาปัจจุบันของตาราง กำหนดให้เวลาปัจจุบันเป็นวันที่ 6 มกราคม 1998 จะเกิดการซ้ำซ้อนในเวลาปัจจุบันขึ้นแสดงในรูปที่ 3.2 ซึ่งข้อมูลของเวลาปัจจุบันนี้สามารถเปลี่ยนแปลงได้ตลอดเวลา ถ้าหากไม่มีการแก้ไขใดกับตารางนี้ภายในสัปดาห์นี้ การซ้ำซ้อนในเวลาปัจจุบันจะหายไปหนึ่งแถว

การซ้ำซ้อนของข้อมูลที่มีความสำคัญมากก็คือการซ้ำซ้อนแบบลำดับเหตุการณ์ คำว่าลำดับเหตุการณ์หมายถึงเป็นข้อกำหนดที่บังคับในตลอดช่วงเวลาอย่างอิสระ ในสามแถวสุดท้ายเป็นการซ้ำซ้อนแบบลำดับเหตุการณ์โดยมีข้อมูลของผู้ป่วยชื่อ "Alexis" มีสถานะอาการเป็น "fair" ตลอดทั้งเดือนธันวาคม ปี 1997 รวมถึง 11 วันแรกของของปี 1998

ในตารางที่ 3.3 แสดงถึงความสัมพันธ์ของการซ้ำซ้อนต่อกันในแต่ละรูปแบบ โดยแต่ละการซ้ำซ้อนของข้อมูลกำหนดให้การซ้ำซ้อนในคอลัมน์แรกตัดกับการซ้ำซ้อนด้านบน ถ้าหากมีเครื่องหมายถูกที่ตรงกันก็แสดงถึงว่ามีลักษณะเช่นนั้นด้วย ส่วนที่เว้นว่างไว้คือไม่จำเป็นต้องเกิด

ความซ้ำซ้อนทั้งสองรูปแบบ สำหรับข้อมูลในตารางนี้ถ้าสองแถวเป็นการซ้ำซ้อนไม่สนใจเวลา (Nonsequenced) ก็ยังสามารถเป็นการซ้ำซ้อนแบบลำดับเหตุการณ์ (Sequenced) ได้ด้วย

ตารางที่ 3.3 ความสัมพันธ์รูปแบบการซ้ำซ้อนกันของข้อมูลภายในตารางเชิงเวลา

	Sequenced	Current	Value-equivalent	Nonsequenced
Sequenced	√		√	
Current	√	√	√	
Value-equivalent			√	
Nosequenced	√		√	√

รูปแบบข้อบังคับที่มีผลต่อข้อบังคับอื่นน้อยที่สุดก็คือข้อมูลตรงกัน (Value-equivalent) และการซ้ำซ้อนที่มีผลข้อบังคับอื่นมากที่สุดคือการซ้ำซ้อนที่ไม่สนใจเวลาคือมีความเป็นไปได้ทุกรูปแบบหมดยกเว้นกรณีการซ้ำซ้อนที่อ้างอิงเวลาปัจจุบัน

3.2 ฐานข้อมูลข้อมูลที่แปรเปลี่ยนตามเวลา

ในฐานข้อมูลทั่วไปมีการเก็บข้อมูลเฉพาะข้อมูลปัจจุบัน (เช่น ยอดคงคลัง, เงินเดือนปัจจุบัน) ฐานข้อมูลเช่นนี้เมื่อมีการเปลี่ยนแปลงข้อมูลในปัจจุบันข้อมูลเดิมจะหายไปทันที สำหรับฐานข้อมูลประวัติ (Historical Database) เป็นฐานข้อมูลที่เก็บเฉพาะข้อมูลที่เป็นอดีต แต่ฐานข้อมูลข้อมูลที่แปรเปลี่ยนตามเวลา (Temporal Database) อาจจะเก็บข้อมูลเกี่ยวกับอนาคตได้ด้วย เช่น การวางแผน หรือหมายกำหนดการ

สำหรับความหมายอย่างเป็นทางการของฐานข้อมูลข้อมูลที่แปรเปลี่ยนตามเวลาคือ "ฐานข้อมูลที่สนับสนุนลักษณะบางประการของเวลา และไม่รวมถึงเวลาที่ได้ถูกกำหนดโดยผู้ใช้" คือการกำหนด "แอททริบิวที่ไม่ได้อยู่ในโดเมนของวันและเวลา ได้แก่เวลาที่ถูกกำหนดโดยผู้ใช้ ตัวอย่างเช่น เงิน, ตัวเลขจำนวนเต็ม หรือเป็น แอททริบิว อย่างเช่น วันเกิด และวันที่ทำการจ้างงาน" การเพิ่มคอลัมน์วันเกิดให้กับตารางของพนักงานนั้นไม่ได้มีหมายถึงการเปลี่ยนแปลงตามเวลา เนื่องจากข้อมูลวันเกิดนั้นถูกกำหนดขึ้นอย่างตายตัวให้กับพนักงานตลอดไป ดังนั้นคอลัมน์วันเวลาไม่ได้หมายความว่าเป็นฐานข้อมูลที่เป็นฐานข้อมูลข้อมูลที่แปรเปลี่ยนตามเวลาเสมอไป

3.3 การไควรีข้อมูลในตารางที่แปรเปลี่ยนตามเวลาวาลิด

ตัวอย่างที่ใช้พิจารณาในส่วนนี้เป็นข้อมูลของฝูงวัวที่ต้องการทำการย้ายคอกและทุ่งหญ้า เพื่อตรวจหาประวัติของการแพร่เชื้อโรควัวบ้า พวกฝูงสัตว์เหล่านี้ถูกจัดเป็นกลุ่ม (lot) ถูกย้ายจากคอกหนึ่งไปสู่อีกคอกหนึ่ง โดยมีการกำหนดตาราง "LOT_LOC" สำหรับใช้บันทึกจำนวนฝูงสัตว์ในแต่

ละกลุ่มที่ถูกเคลื่อนย้ายไปในแต่ละคอกของแต่ละทุ่งเลี้ยงสัตว์ ในโครงสร้างตารางแบบสมบูรณ์มีทั้งหมดเก้าคอลัมน์ดังนี้

LOT_LOC(FDYD_ID, LOT_ID_NUM, PEN_ID, HD_CNT, FROM_DATE,
FROM_MOVE_ORDER, TO_DATE, TO_MOVE ORDER,_RECORD DATE)

ในตารางนี้เป็นตารางที่แปรเปลี่ยนตามเวลาวาลิด คือบันทึกข้อมูลที่เป็นจริง (valid) ในแต่ละช่วงเวลา และเก็บสถานะของมันไว้ด้วย สำหรับคอลัมน์ "FROM" และ "TO" เป็นตัวกำหนดขอบเขตของคาบเวลาที่เชื่อว่าเป็นจริงอยู่ (Period of Validity) ของข้อมูลในแต่ละแถว หน่วยย่อยของเวลาในตารางนี้มีความละเอียดมากกว่าวัน "วัน" นั่นคือสามารถมีการเคลื่อนย้ายฝูงสัตว์ได้มากกว่าหนึ่งครั้งต่อวัน และมีการกำหนดเวลา "RECORD_DATE" เป็นตัวระบุว่าข้อมูลนี้ได้ถูกบันทึกเมื่อไร ในการอธิบายนี้เรายังไม่ให้ความสนใจกับคอลัมน์ "FROM_MOVE_ORDER", "TO_MOVE_ORDER" และ "RECORD_DATE" และทำการไควรีในโครงสร้างตารางอย่างง่ายที่ทำการลดทอนแล้ว เฉพาะสี่คอลัมน์แรกเป็นคอลัมน์ที่เป็นข้อมูลประเภทเลขจำนวนเต็ม ในสองคอลัมน์สุดท้ายเป็นข้อมูลประเภทประเภทวันที่

ตารางที่ 3.4 ข้อมูลในตาราง "LOT_LOC"

FDYD_ID	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	137	1	17	1998-02-07	1998-02-18
1	219	1	43	1998-02-25	1998-03-01
1	219	1	20	1998-03-01	1998-03-14
1	219	2	23	1998-03-01	1998-03-14
1	219	2	43	1998-03-14	9999-12-31
1	374	1	14	1998-02-20	9999-12-31

จากตารางที่ 3.4 ฝูงวัว 17 ตัวอยู่ในคอกที่ 1 เป็นเวลา 11 วัน แล้วได้ทำการเคลื่อนย้ายตัวที่มีอาการไม่ดีออกจากทุ่งในวันที่ 18 กุมภาพันธ์ และฝูงวัว 14 ตัวจากกลุ่ม 374 ยังอยู่ในคอกที่ 1 (คำว่า "ตลอดไป" ในการระบุให้เป็นจริงเรื่อยไปจนถึงปัจจุบัน) ฝูงวัวจำนวน 23 ตัวจากกลุ่ม 219 ได้ถูกย้ายออกจากคอก 1 ไปยังคอก 2 ในวันที่ 1 มีนาคม ทำให้เหลือฝูงวัว 20 ตัวในกลุ่ม และได้ถูกย้ายไปยังคอก 2 ในวันที่ 14 มีนาคม และอยู่ที่นั่นจนถึงปัจจุบัน

ในส่วนที่ผ่านมาได้อธิบายถึงสามลักษณะพื้นฐานของข้อบังคับการเกิดข้อมูลซ้ำซ้อนได้แก่ ปัจจุบัน, ลำดับเหตุการณ์, และไม่สนใจเวลา ในข้อบังคับการเกิดข้อมูลซ้ำซ้อนในปัจจุบัน(ของชื่อผู้ป่วย และสถานะ สำหรับบันทึกอาการของผู้ป่วยในหน่วยดูแลผู้ป่วยเด็ก) ได้บอกถึง "แต่ละผู้ป่วยสามารถมีสถานะได้มากที่สุดหนึ่งสถานะ" สำหรับลำดับเหตุการณ์คือ "ไม่มีเวลาใดๆ ที่ผู้ป่วยสามารถมีสองสถานะตรงกันได้" และไม่สนใจเวลาคือ "ผู้ป่วยไม่สามารถมีสถานะตรงกัน โดยมีคาบเวลาตรงกัน" เราเห็นว่า ลำดับเหตุการณ์เป็นเหมือนเงื่อนไขปกติบนข้อบังคับปกติบน

ฐานข้อมูลปกติ และเช่นเดียวกับตาราง "LOT_LOC" ซึ่งควรมีข้อบังคับการเกิดข้อมูลซ้ำซ้อนให้กับ "FDYD_ID", "LOT_ID_NUM", "PEN_ID" เป็นหนึ่งเดียวตลอดเวลา ซึ่งก็คือแบบลำดับเหตุการณ์นั่นเอง

3.3.1 การไควรีข้อมูลจากตารางที่แปรเปลี่ยนตามเวลา

ในส่วนนี้สร้างข้อกำหนดของการไควรีในรูปแบบต่างๆ ตัวอย่างเช่น "ฝูงสัตว์จำนวนเท่าไรที่อยู่ในแต่ละคอกของกลุ่ม 219 ในทุ่งเลี้ยงสัตว์ 1" ถ้าตารางเป็นตารางที่ไม่ได้แปรผันตามเวลามีโครงสร้างตารางเป็น LOT_LOC_SNAPSHOT(FDY_ID, LOT_ID_NUM, PEN_ID, HD_CNT) การไควรีแบบนี้สามารถเขียนเป็นเอสคิวแอลได้อย่างไม่ยาก

```
SELECT PEN_ID, HD CNT
FROM LOT_LOC
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
```

เช่นเดียวกันกรณีเป็นแปรเปลี่ยนตามเวลาปัจจุบันภายใต้ตาราง "LOT_LOC" ที่เป็นเวลาวาลิด คือ "มีจำนวนฝูงสัตว์เท่าไรจากกลุ่ม 219 ในทุ่งเลี้ยงสัตว์ 1 (ในเวลาปัจจุบัน) อยู่ในแต่ละคอก" สำหรับการไควรีเช่นนี้ เราจะสนใจเฉพาะในส่วนที่เป็นจริงในปัจจุบันเท่านั้น โดยการเพิ่มเงื่อนไขในอนุประโยค "WHERE" เพื่อหาข้อมูลตามเงื่อนไข

```
SELECT PEN_ID, HD CNT
FROM LOT_LOC
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
    AND TO_DATE = DATE '9999-12-31'
```

ผลจากการไควรี ได้เป็นผลลัพธ์ซึ่งเป็นสถานะของสัตว์ทุกตัวในกลุ่ม 219 ซึ่งในในปัจจุบันมีอยู่คอกเดียวดังแสดงในตารางที่ 3.5

ตารางที่ 3.5 ผลลัพท์จากการไควรีเวลาปัจจุบัน

PEN_ID	HD_CNT
2	43

สำหรับการไควรีในลักษณะของลำดับเหตุการณ์คือ "ให้แสดงข้อมูลจากการบันทึกจำนวนฝูงสัตว์จากกลุ่ม 219 ในทุ่งเลี้ยงสัตว์ 1 ในแต่ละคอก" ในการไควรีแบบลำดับเหตุการณ์ทำได้โดยให้มีส่วนของคอลัมน์บันทึกเวลาต่อท้ายอยู่ด้วย

```
SELECT PEN_ID, HD_CNT, FROM_DATE, TO_DATE
FROM LOT_LOC
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
```

ผลลัพท์เพื่อดูการบันทึกในตารางที่ 3.6 เราเห็นได้ว่ากลุ่ม 219 มีการเคลื่อนย้ายไป

ตารางที่ 3.6 ผลลัพท์จากการไควรีแบบลำดับเหตุการณ์

PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	43	1998-02-25	1998-03-01
1	20	1998-03-01	1998-03-14
2	23	1998-03-01	1998-03-14
2	43	1998-03-14	9999-12-31

การไควรีแบบไม่สนใจเวลา คือ "จำนวนฝูงสัตว์เท่าไรจากกลุ่ม 219 ในทุ่งเลี้ยงสัตว์ที่ 1 ที่เคยได้อยู่ในคอกใดบ้าง" ไม่ต้องสนใจว่าข้อมูลเป็นจริงเมื่อไรบ้าง สนใจเฉพาะเมื่อไรก็ตามที่มีกลุ่มที่ต้องการอยู่ในคอกเท่านั้น การไควรีนี้เหมือนการไควรีอย่างง่ายในเอสคิวแอลโดยไม่ต้องสนใจคอลัมน์บันทึกเวลาแต่อย่างใด

```
SELECT PEN_ID, HD_CNT
FROM LOT_LOC
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
```

ตารางที่ 3.7 ผลลัพท์จากการไควรีแบบไม่สนใจเวลา

PEN_ID	HD_CNT
1	43
1	20
2	23
2	43

3.4 การปรับปรุงข้อมูลในตารางเวลาวาลิด

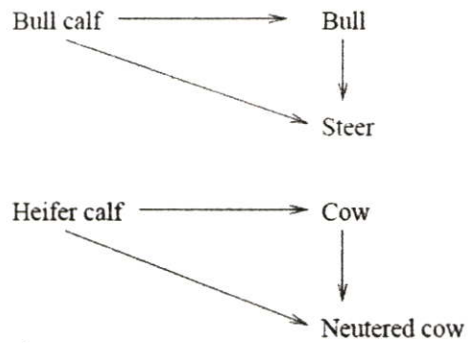
ในหัวข้อที่ 3.3 ผ่านมาได้อธิบายตัวอย่างของข้อมูลของวัวที่ได้เคลื่อนย้ายจากคอกหนึ่งไปสู่อีกคอกหนึ่งในทุ่งเลี้ยงสัตว์ ซึ่งเป็นวิธีการติดตามตรวจสอบการระบาดของโรคที่ถ่ายทอดกันภายในฝูงวัว เราขอกล่าวถึงการแบ่งลักษณะนามของวัวประกอบด้วยการเปลี่ยนแปลงทางร่างกายของวัว ซึ่งก็เป็นตัวแปรสำคัญของการระบาดของโรคเช่นกัน เพื่อเป็นตัวอย่างประกอบการอธิบาย

3.4.1 คำอธิบายตัวอย่าง

วัวเพศผู้เรียกว่า "bull" วัวเพศเมียคือ "cow" ลูกวัวคือ "calf" วัวตัวเมียที่ยังไม่มีลูกคือ "heifer" และฝูงวัวคือ "cattle" วัวที่ทำการตอนแล้วเรียกว่า "steer" การตอนของ "cow" และ "heifer" จะเรียกว่า "neutered cow" รูปที่ 3.1 ได้อธิบายถึงการเปลี่ยนแปลงที่เป็นไปได้ทั้งหมด เมื่อเปลี่ยนแล้วไม่สามารถเปลี่ยนกลับได้

การแบ่งลักษณะนามของฝูงวัวมีความสำคัญมากในการศึกษาการระบาดของโรค เนื่องการแบ่งลักษณะนามนี้มีผลในการระบาดของโรคด้วย โครงสร้างตาราง "LOT" มีการใช้เวลาวาลิดเข้ามาร่วมอยู่ด้วย แสดงดังตารางที่ 3.8 ซึ่งเป็นตารางที่ได้ลดทอนข้อมูลบางส่วนออกไป

"GNDR_CODE" เป็นอักษรย่อตัวเดียวได้แก่ c = bull calf, h = heifer และ s = steer ใช้ "FROM_DATE" และ "TO_DATE" ในการกำหนดคาบเวลาให้ทุกคอลัมน์ของแถวเป็นจริง ข้อมูลในตารางที่ 3.8 เมื่อวันที่ 23 มีนาคม 1998 ได้เกิดเหตุการณ์กับวัวในกลุ่ม 101 ถูกตอน และกลุ่ม 234 ประกอบไปด้วยกลุ่มของลูกวัวที่มี "TO_DATE" เป็นตลอดไปหมายถึงยังเป็นจริงอยู่จนถึงปัจจุบัน กลุ่ม 234 เข้ามาในทุ่งเลี้ยงสัตว์วันที่ 17 กุมภาพันธ์ และฝูง 799 เข้ามา



รูปที่ 3.1 การเปลี่ยนแปลงลักษณะนามของวัว

ตารางที่ 3.8 ข้อมูลบางส่วนที่ตัดทอนมาจากตาราง "LOT"

LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	c	1998-01-01	1998-03-23
101	s	1998-03-23	9999-12-31
234	c	1998-02-17	9999-12-31
799	s	1998-03-12	9999-12-31

ในวันที่ 12 มีนาคมได้สะสมข้อมูลจากทุ่งเลี้ยงสัตว์เพื่อหาสถิติจากฐานข้อมูล และได้ทำวิธีการแก้ไขเพื่อปรังปรุงข้อมูลในตาราง LOT (การปรับปรุงข้อมูลประกอบด้วย คำสั่ง "INSERT", "DELETE" และ "UPDATE") ดังที่ได้อธิบายผ่านมาในข้อบังคับการเกิดการข้อมูลซ้ำซ้อนของ เวลาปัจจุบัน, ลำดับเหตุการณ์, และไม่สนใจเวลา การไควรีในส่วนต่อไปจะพูดถึงการแก้ไข และปรับปรุงข้อมูลของ เวลาในปัจจุบัน, ลำดับเหตุการณ์, และไม่สนใจเวลา

3.4.2 การปรับปรุงข้อมูลในเวลาปัจจุบัน

ถ้าหากมี "heifer" กลุ่มใหม่มาถึงในวันนี้ การเพิ่มข้อมูลเข้าไปแบบเวลาปัจจุบันมีคำสั่งในเอสคิวแอลดังนี้

```
INSERT INTO LOT
VALUES (433, 'h', CURRENT_DATE, DATE '9999-12-31')
```

จากคำสั่งจะระบุให้คาบเวลาเริ่มตั้งแต่เวลา ณ ปัจจุบัน และกำหนดให้เรื่อยไปในจุดสิ้นสุดของคาบเวลา ถ้าหากกลุ่มที่ 101 ออกจากทุ่งเลี้ยงสัตว์ก็ควรมีคำสั่ง "DELETE" เป็น

```
DELETE FROM LOT
WHERE LOT_ID_NUM = 101
```

แต่การทำการลบในเวลาปัจจุบันในลักษณะของลอจิคอลที่เป็นตารางที่แปรผันตามเวลานั้น จะใช้คำสั่ง "UPDATE" แทนคำสั่ง DELETE การลบแบบปัจจุบันจะทำตั้งแต่ ปัจจุบันถึงเรื่อยไป

```
UPDATE LOT
SET TO_DATE = CURRENT_DATE
WHERE LOT_ID_NUM = 101
   AND TO_DATE = DATE '9999-12-31'
```

ตารางที่ 3.9 ข้อมูลในตาราง "LOT" ก่อนทำการลบ

LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	c	1998-01-01	1998-03-23
101	s	1998-03-23	9999-12-31
234	c	1998-02-17	1998-10-17
234	s	1998-10-17	9999-12-31
799	c	1998-03-12	9999-12-31

ตารางที่ 3.10 ผลจากการลบแบบปัจจุบันจากตารางที่ 3.9

LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	c	1998-01-01	1998-03-23
101	s	1998-03-23	9999-12-31
234	c	1998-02-17	1998-07-29
799	c	1998-03-12	9999-12-31

พิจารณาตาราง LOT ที่แสดงในตารางที่ 3.9 สมมติวันนี้เป็นวันที่ 29 กรกฎาคม ตารางนี้แสดงให้เห็นว่ากลุ่ม 234 ถูกกำหนดให้มีการตอนวัว ในวันที่ 17 ตุลาคม หากต้องการลบกลุ่มที่ 234 ออกในวันนี้จะประกอบด้วยคำสั่งในลักษณะฟิสิกคอล "UPDATE" และคำสั่ง "DELETE"

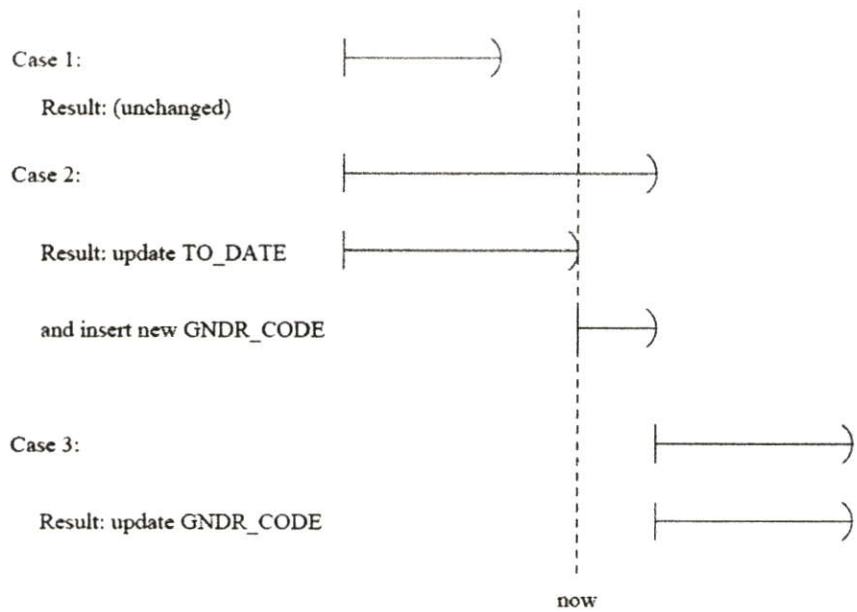
```
UPDATE LOT
SET TO_DATE = CURRENT_DATE
WHERE LOT_ID_NUM = 234
   AND TO_DATE >= CURRENT_DATE
   AND FROM_DATE < CURRENT_DATE

DELETE FROM LOT
WHERE LOT_ID_NUM = 234
   AND FROM_DATE > CURRENT_DATE
```

ต้องใช้สองคำสั่งการลบจึงเป็นไปอย่างสมบูรณ์ ผลการลบข้อมูลจากตารางที่ 3.9 แสดงในตารางที่3.10 ข้อมูลที่เหลือของกลุ่ม 234 ที่อยู่หลังจากเวลาปัจจุบันก็ต้องถูกลบออกไปด้วย

ถ้ามีการตอนวัวในกลุ่มที่ 799 คำสั่งของตารางที่ไม่ได้เป็นข้อมูลเชิงเวลาแล้วควรมีคำสั่งเป็นดังนี้

```
UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799
```

แต่การปรับปรุงข้อมูลในปัจจุบันนี้จะประกอบด้วยคำสั่ง "UPDAET" พร้อมทั้งมีคำสั่ง "INSERT" ร่วมด้วย ดังนั้นการแก้ไขข้อมูลบนตารางเวลาวาลิดเป็นดังต่อไปนี้



รูปที่ 3.2 กรณีต่างๆ ของการปรับปรุงข้อมูลแบบเวลาปัจจุบัน

```
INSERT INTO LOT
SELECT DISTINCT 799, 's', CURRENT_DATE, DATE '9999-12-31'
FROM LOT
WHERE EXISTS (SELECT *
    FROM LOT
    WHERE LOT_ID_NUM = 799
      AND TO_DATE = DATE '9999-12-31')

UPDATE LOT
SET TO_DATE = CURRENT_DATE
WHERE LOT_ID_NUM = 799
  AND GNDR_CODE <> 's'
  AND TO_DATE = DATE '9999-12-31'
```

การปรับปรุงข้อมูลนี้หยุดที่ค่าเวลาปัจจุบัน และเพิ่มแถวและข้อมูลใหม่ การปรับปรุงเกิดหลังจากการ "INSERT" ในส่วนที่เวลามากกว่าปัจจุบันขึ้นไป และ "UPDATE" เพื่อเปลี่ยน GNDR_CODE ไปเป็น 's' และเปลี่ยน FROM_DATE ไปเป็นเวลาปัจจุบัน

รูปที่ 3.2 แสดงถึงสามกรณีที่เป็นไปได้ทั้งหมดของคาบเวลาจากการปรับปรุงข้อมูลในตารางในแต่ละแถว จากรูปมีการเวลาที่เคลื่อนจากซ้ายไปขวา และ "now" แสดงด้วยเส้นประในแนวตั้ง ในกรณีที่ 1 ถ้าคาบเวลาสิ้นสุดไปแล้วในอดีตการปรับปรุงข้อมูลจะไม่มีผลอะไรกับแถวนั้นเลย (หรือพูดได้ว่าการ "UPDATE" ทำจาก "now" ถึง "forever") ถ้าแถวนั้นเป็นจริงในปัจจุบัน (กรณีที่ 2) และส่วนที่อยู่ก่อน "now" ต้องถูกกำหนดให้มันสิ้นสุดว่าเป็นเช่นนั้นจริงจนถึง "now" และสร้างแถวใหม่ด้วยเพศของวัวที่แก้ไขแล้ว การ "INSERT" เข้าไปนี้จะเริ่มเวลาที่ "now" และสิ้นสุดที่เวลาเดิมของแถวเดิม ถ้าแถวเริ่มในอนาคตก็ต้องถูก "UPDATE" ไปด้วย ทั้งหมดนี้ใช้สองคำสั่ง "UPDATE" กับอีกหนึ่งคำสั่ง "INSERT"

```
INSERT INTO LOT
SELECT LOT_ID_NUM, 's', CURRENT_DATE, TO_DATE
FROM LOT
WHERE LOT_ID_NUM = 799
   AND FROM_DATE <= CURRENT_DATE
   AND TO_DATE > CURRENT_DATE

UPDATE LOT
SET TO_DATE = CURRENT_DATE
WHERE LOT_ID_NUM = 799
   AND GNDR_CODE <> 's'
   AND FROM_DATE < CURRENT_DATE
   AND TO_DATE > CURRENT_DATE

UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799
   AND FROM_DATE >= CURRENT_DATE
```

ในคำสั่ง "UPDATE" ลำดับที่สองนั้นทำการปรับปรุงข้อมูลทั้งหมดที่เริ่มตั้งแต่ปัจจุบันเรื่อยไปจนถึงอนาคต

3.4.3 การปรับปรุงข้อมูลแบบลำดับเหตุการณ์

การปรับปรุงข้อมูลแบบเวลาปัจจุบันกระทำจาก "now" ถึง "forever" ส่วนการปรับปรุงแบบลำดับเหตุการณ์ ทั่วไปคือกระทำเฉพาะส่วนคาบเวลาที่กำหนดได้ไว้ ในคาบเวลานี้ยังสามารถอยู่ในอดีต ในอนาคต หรือซ้อนทับกับ "now" ก็ได้

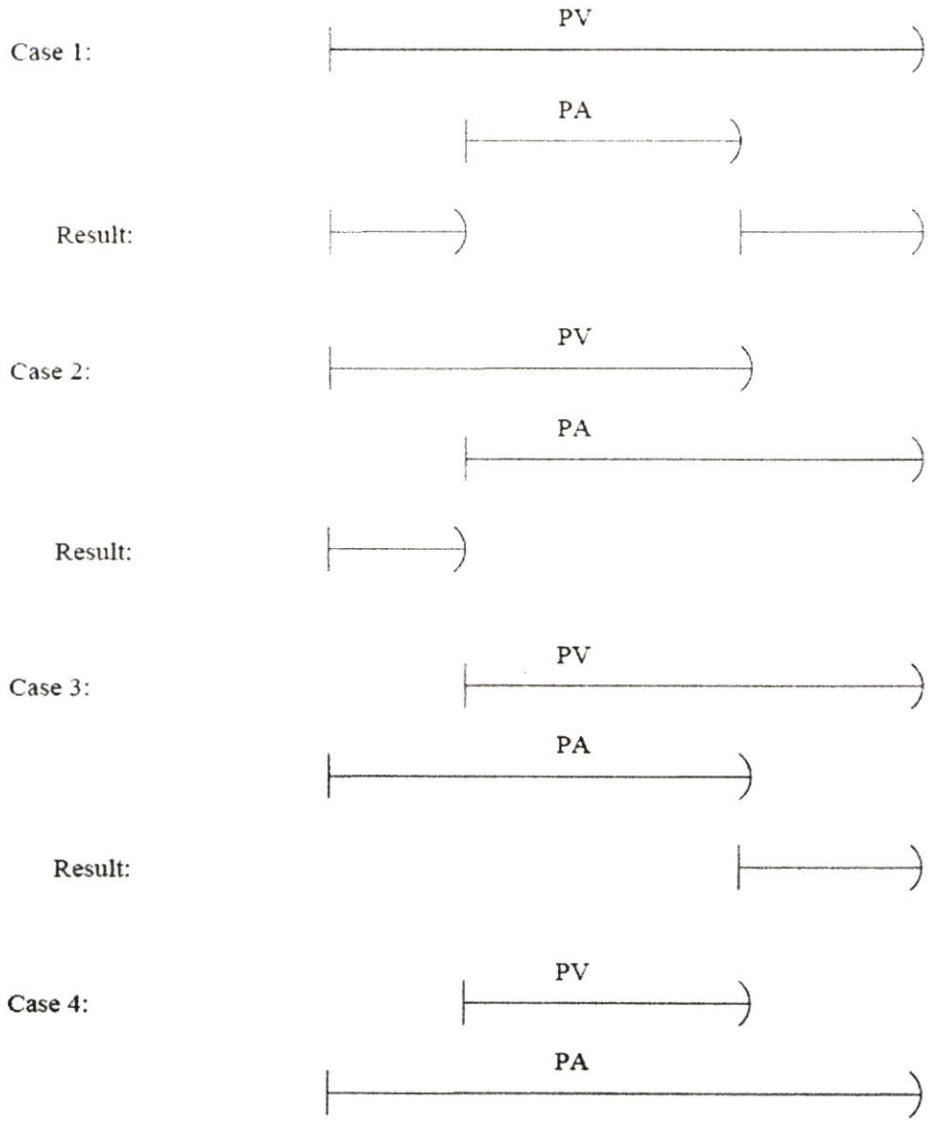
ที่ได้อธิบายมาแล้วนั้นสามารถนำมาใช้กับการปรับปรุงข้อมูลแบบลำดับเหตุการณ์ได้ ก็คือแทน CURRENT_DATE กับจุดเริ่มของคาบเวลาที่ต้องการปรับปรุง และ DATE '9999-12-31' แทนด้วยจุดสิ้นสุดของคาบเวลาที่ต้องการกระทำ

ในการเพิ่มข้อมูลแบบลำดับเหตุการณ์ขอยกตัวอย่างการเพิ่มกลุ่ม 426 เป็นฝูงของ "heifer" อยู่บนทุ่งเลี้ยงสัตว์ตั้งแต่วันที่วันที่ 26 มีนาคมถึง 14 กุมภาพันธ์

```
INSERT INTO LOT
VALUES (426, 'h', DATE '1998-03-26', DATE '1998-04-14')
```

การลบแบบเวลาปัจจุบันการสร้างนั้นใช้คำสั่ง "UPDATE" กับส่วนที่เป็นจริงในปัจจุบัน และใช้คำสั่ง "DELETE" กับส่วนที่เริ่มขึ้นในอนาคต การ "DELETE" แบบ ลำดับเหตุการณ์ เป็นไปได้ด้วยกันทั้งหมด 4 กรณี ดังที่ได้แสดงในรูปที่ 3.3 ในแต่ละกรณีมีคาบเวลาเดิม (Period of Validity หรือ PV) อยู่ด้านบนของคาบเวลาที่เข้ามากระทำ (Period of Applicability หรือ PA) เป็นคาบเวลาที่ต้องการลบ ในกรณีที่หนึ่ง แถวเดิมคลุมส่วนคาบกระทำทั้งหมด ดังนั้นส่วนหัวและท้ายของคาบเดิมจึงยังคงอยู่ ในส่วนหัวของคาบปรับ TO_DATE ไปเป็นจุดเริ่มของคาบเวลาที่เป็นตัวกระทำการลบ ส่วนคาบเวลาตอนปลายให้ทำการ "INSERT" เข้าไปใหม่ ในกรณีที่สอง มีเพียงส่วนต้นของคาบเวลาของแถวเดิมเท่านั้นที่ยังคงอยู่ เช่นเดียวกันในกรณีที่สาม มีเพียงส่วนปลายเท่านั้นของคาบเวลาที่ยังคงเดิม และในกรณีที่สี่ทั้งแถวต้องถูกลบ ซึ่งคาบเวลากระทำการลบครอบคลุมทั้งหมด

การลบแบบ ลำดับเหตุการณ์ ใช้การปรับปรุงข้อมูลที่เป็นฟิสิกคอลทั้งหมดสี่ตัว เราต้องการให้กลุ่ม 234 ออกจากทุ่งเลี้ยงสัตว์ในสามสัปดาห์แรกของเดือนตุลาคม เมื่อมีการตอนเกิดขึ้น (จากข้อมูลในตารางที่ 3.10) ดังนั้นคาบเวลาที่กระทำการลบก็คือ DATE '1998-10-01' ถึง DATE '1998-10-22' (ได้กำหนดให้คาบเวลา TO_DATE อยู่หลังคาบเวลาสิ้นสุด)



รูปที่ 3.3 กรณีทั้งหมดของการลบแบบ ลำดับเหตุการณ์

```
INSERT INTO LOT
SELECT LOT_ID_NUM, GNDR_CODE, DATE '1998-10-22', TO_DATE
FROM LOT
WHERE LOT_ID_NUM = 234
   AND FROM_DATE <= DATE '1998-10-01'
   AND TO_DATE > DATE '1998-10-22'

UPDATE LOT
SET TO_DATE = DATE '1998-10-01'
WHERE LOT_ID_NUM = 234
   AND FROM_DATE < DATE '1998-10-01'
   AND TO_DATE >= DATE '1998-10-01'

UPDATE LOT
SET FROM_DATE = DATE '1998-10-22'
WHERE LOT_ID_NUM = 234
```

```
  AND FROM_DATE < DATE '1998-10-22'
  AND TO_DATE >= DATE '1998-10-22'

DELETE FROM LOT
WHERE LOT_ID_NUM = 234
  AND FROM_DATE >= DATE '1998-10-01'
  AND TO_DATE <= DATE '1998-10-22'
```

ในกรณีที่หนึ่งจะมีผลกับสองคำสั่งแรก ในคำสั่งที่สองยังครอบคลุมถึงกรณีที่สองด้วย ในคำสั่งที่สามจัดการกับกรณีที่สาม และคำสั่งที่สี่จัดการกับกรณีที่สี่ ทั้งสี่คำสั่งต้องถูกทำตามลำดับ โดยจะครอบคลุมทั้งหมดทุกกรณีที่เป็นไปได้

การ "UPDATE" แบบ ลำดับเหตุการณ์เหมือนกับการ "UPDATE" ในตารางที่ไม่ได้แปรเปลี่ยนตามเวลาหากแต่เพียงมีการกำหนดคาบเวลาที่กระทำเข้าไปด้วยเช่นหากมีการตอนวัวกลุ่มที่ 799

```
UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799
```

เราจะเปลี่ยนคำสั่งนี้ไปเป็นการ "UPDATE" แบบ ลำดับเหตุการณ์ เหมือนกับการลบแบบลำดับเหตุการณ์ มีหลายกรณีที่สามารถเกิดขึ้นได้ในการ "UPDATE" แบบลำดับเหตุการณ์ ซึ่งมีอยู่ 4 กรณีด้วยกันในรูปที่ 3.4 เป็นการจัดทุกรูปแบบของการ "UPDATE" ในกรณีที่หนึ่งจากรูป ส่วนต้นและปลายของคาบเวลายังคงเดิมอยู่ (โดยใช้สองคำสั่ง "INSERT") และคาบเวลาเดิมก็ทำการ "UPDATE" กรณีที่สองมีเพียงส่วนต้นของคาบเวลาเท่านั้นที่ยังคงเดิม ในกรณีที่สามมีเพียงส่วนปลายเท่านั้นที่ยังคงเดิม ในกรณีที่สี่คาบเวลาทั้งหมดถูก "UPDATE" เนื่องจากคาบเวลาที่มากกระทำคลุมทั้งหมดของคาบเวลาที่จะถูก "UPDATE"

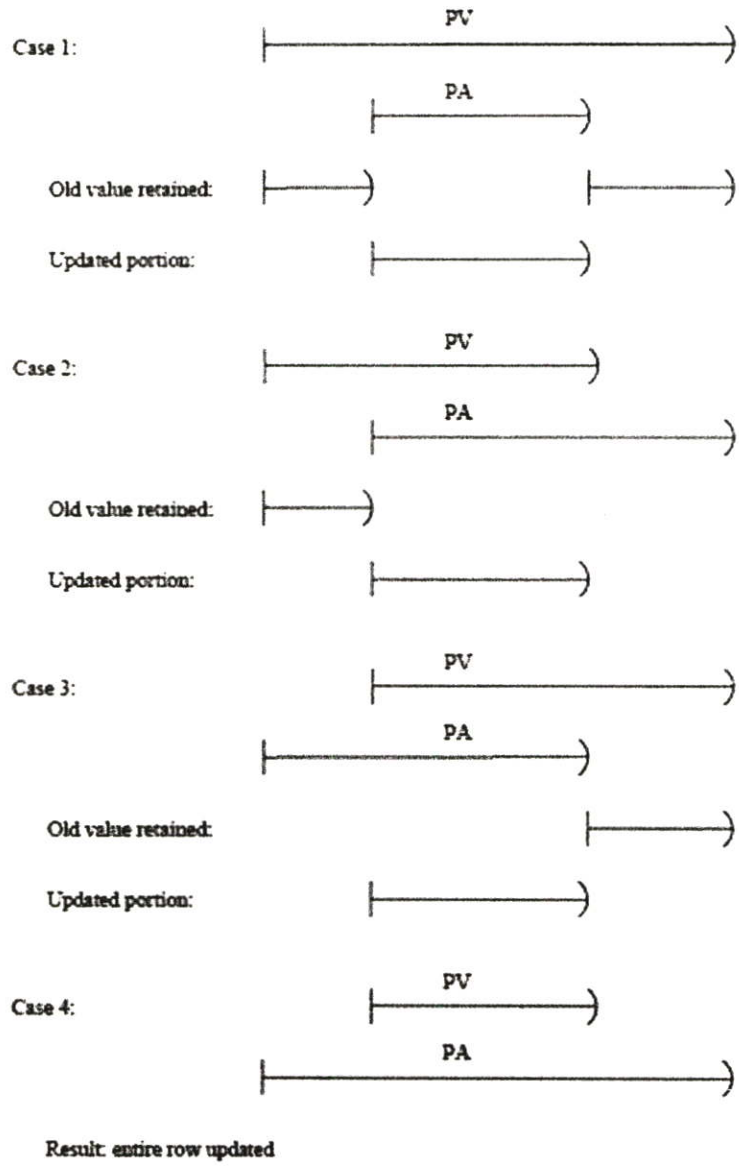
สิ่งที่เกิดขึ้นคือ (1) ทำคำสั่ง "INSERT" เปลี่ยนค่าเดิมของ FROM_DATE ไปเป็นจุดเริ่มของคาบเวลาที่เข้ามาทำการลบ (2) ทำคำสั่ง "INSERT" จากค่าเดิมของจุดสิ้นสุดของคาบเวลาที่กระทำการลบให้เป็น TO_DATE (3) ทำการ "UPDATE" คอลัมน์ของแถวที่มีการซ้อนทับของคาบเวลา (4) ทำการ "UPDATE" FROM_DATE ให้เป็นจุดเริ่มของคาบเวลาที่กำหนดการลบของแถวที่ซ้อนทับกัน (5) ทำการ "UPDATE" TO_DATE ให้เป็นจุดสิ้นสุดของคาบเวลาที่กำหนดการลบ

ตัวอย่างของการ "UPDATE" แบบ ลำดับเหตุการณ์ ให้มีการตอนในกลุ่มวัวเฉพาะที่อยู่ในช่วงเดือนมีนาคมเท่านั้น (มีบางอย่างเกิดขึ้นในวันที่ 1 เมษายน เพื่อที่จะแสดงให้เห็นว่าการทำการ "UPDATE" แบบ ลำดับเหตุการณ์นี้ไม่ได้ทำการ "UPDATE" แค่ส่วนของฝูงวัวเท่านั้น) คาบเวลาที่กำหนดการลบจึงอยู่ในช่วง DATE '1998-03-01' ถึง DATE '1998-04-01'

ในคำสั่ง "INSERT" แรกเป็นตัวจัดการส่วนเริ่มต้นของกรณีที่ 1 และ 2 คำสั่ง "INSERT" ถัดมาจักการกับส่วนปลายของกรณีที่ 2 และ 3 ส่วนคำสั่ง "UPDATE" แรกนั้นทำการ "UPDATE" ให้กับ

ทั้ง 4 กรณี ส่วนการ "UPDATE" ถัดมาและสุดท้ายทำการปรับจุดเริ่มต้น (กรณีที่ 1 และ 2) และจุดสิ้นสุด (กรณีที่ 1 และ 3) ในการ "UPDATE" สามคำสั่งนั้นจะไม่มีผลต่อแถวที่ได้ "INSERT" เข้าไปดังคำสั่ง "INSERT" สองคำสั่งแรก ก็คือคาบเวลาเหล่านั้นอยู่นอกเหมือนคาบเวลาที่เป็นตัวกำหนดการ "UPDATE" คำสั่งทั้งห้าคำสั่งที่ใช้ในการ "UPDATE" แบบลำดับเหตุการณ์ เป็นดังนี้

```
INSERT INTO LOT
SELECT LOT_ID_NUM, GNDR_CODE, FROM_DATE, DATE '1998-03-01'
FROM LOT
WHERE LOT_ID_NUM = 799
   AND FROM_DATE < DATE '1998-03-01'
   AND TO_DATE > DATE '1998-03-01'

INSERT INTO LOT
SELECT LOT_ID_NUM, GNDR_CODE, DATE '1998-04-01', TO_DATE
FROM LOT
WHERE LOT_ID_NUM = 799
   AND FROM_DATE < DATE '1998-04-01'
   AND TO_DATE > DATE '1998-04-01'

UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799
   AND FROM_DATE < DATE '1998-04-01'
   AND TO_DATE > DATE '1998-03-01'

UPDATE LOT
SET FROM_DATE = DATE '1998-03-01'
WHERE LOT_ID_NUM = 799
   AND FROM_DATE < DATE '1998-03-01'
   AND TO_DATE > DATE '1998-03-01'

UPDATE LOT
SET TO_DATE = DATE '1998-04-01'
WHERE LOT_ID_NUM = 799
   AND FROM_DATE < DATE '1998-04-01'
```



รูปที่ 3.4 กรณีทั้งหมดจากการ "UPDATE" แบบ ลำดับเหตุการณ์

3.4.4 การปรับปรุงข้อมูลแบบไม่สนใจเวลา

การปรับปรุงข้อมูลแบบ ไม่สนใจเวลา ก็เหมือนกับคอนสแตรนท์ และการไควรี คือให้การบันทึกเวลาเป็นเหมือนกับคอลัมน์อื่นๆ ตัวอย่างการปรับปรุงข้อมูล ให้ "ลบกลุ่ม 234" ในการลบแบบเวลาปัจจุบันคือ "กลุ่ม 234 ได้ออกจากทุ่งเลี้ยงสัตว์" ในการลบแบบลำดับเหตุการณ์จะมีการกำหนดคาบเวลาที่กระทำด้วย คือ "กลุ่ม 234 หายไปจากทุ่งเลี้ยงสัตว์ในสามาสัปดาห์แรกของเดือนมิถุนายน" สำหรับการลบแบบ ไม่สนใจเวลา ทำการลบจากช่วงเวลาที่ได้อ้างอิงถึงเพื่อเป็นการลบ ตัวอย่างคือ "ลบกลุ่ม 234 ที่อยู่มายาวนานกว่า 3 เดือนออกจากฐานข้อมูล"

```
DELETE FROM LOT
WHERE LO_ ID_NUM = 234
AND (TO_DATE – FROM_DATE MONTH) > INTERVAL '3' MONTH
```

การลบแบบเวลาปัจจุบัน และลำดับเหตุการณ์จะอ้างถึงเหตุการณ์ที่เกิดขึ้นจริงๆ เนื่องจากมันทำให้โมเดลเปลี่ยน สำหรับการแก้ไขข้อมูลที่ไม่สนใจเวลาเป็นการอ้างถึงส่วนที่เราต้องการ (การลบแบบเจาะจงแถว) สำหรับแบบอิงเวลาปัจจุบันปัจจุบัน และลำดับเหตุการณ์จะมีความซับซ้อนมากกว่าการลบแบบไม่สนใจเวลา

การปรับปรุงข้อมูลทุกอย่างทำให้เกิดการเปลี่ยนแปลงของโมเดล (บางข้อมูลบางอย่างอาจกลายเป็นจริง หรือเป็นจริงในบางเวลาในอนาคต หรืออาจจะมีการเปลี่ยนแปลงในปัจจุบันหรืออนาคต) ไม่ว่าเป็นเวลาปัจจุบันหรือลำดับเหตุการณ์สำหรับการปรับปรุงข้อมูลแบบไม่สนใจเวลานั้นง่ายต่อการใช้งานด้วยเอสคิวแอลแต่ปกติแล้วการปรับปรุงข้อมูลในลักษณะนี้เราได้ใช้ไม่บ่อยนัก

บทที่ 4
โครงสร้างของฐานข้อมูลเอ็มเอสบีดี

การปรับปรุงฐานข้อมูลเชิงสัมพันธ์ที่มีใช้อยู่ในปัจจุบันให้มีคุณสมบัติตามแบบเอ็มเอสบีดีได้นั้น เป็นเรื่องที่ท้าทายอย่างมากเพราะเต็มไปด้วยความซับซ้อนของข้อมูลที่ต่างเวลาและต่างระดับชั้นของความปลอดภัย รวมถึงการตีความหมายของข้อมูลด้วย ในบทนี้ได้กล่าวถึงข้อกำหนดเพื่อให้ฐานข้อมูลมีคุณสมบัติของเอ็มเอสบีดีได้

4.1 โครงสร้างและความหมายของแอททริบิว

โครงสร้างพื้นฐานของเอ็มเอสบีดี เราได้ทำการนำฐานข้อมูลเชิงสัมพันธ์มาดัดแปลงเพื่อให้มีความสามารถรองรับในความต้องการของเราได้ ในบางคุณลักษณะที่ไม่มีในฐานข้อมูลเช่นลักษณะของคุณสมบัติของเวลา และระดับความปลอดภัยหลายระดับ ดังนั้นเพื่อให้ได้ความสามารถตามที่ต้องการโดยการเพิ่มคุณสมบัติเหล่านั้นเข้าไปโดยเพิ่มคอลัมน์ที่เป็นระบุถึงระดับความปลอดภัยให้แก่ระดับแถว หรือ tuple class (TC) ระบุผู้ที่สร้างแถว creator class (CC) และเพิ่มระดับความปลอดภัยให้กับแอททริบิวข้อมูลแต่ละตัว ได้เป็นโครงสร้างดังนี้

$R(A_1, C_1, A_2, C_2,, \dots , A_n, C_n, VT\_S, VT\_E, TT\_S, TT\_E, TC, CC)$

การเพิ่มเข้ามาจากคุณสมบัติที่มีสองส่วนในส่วนแรกขอกล่าวถึงระดับความปลอดภัยที่มีหลายระดับชั้นเราต้องการการเพิ่มของแอททริบิวขึ้นมาดังต่อไปนี้

A_i คือเป็นแอททริบิวข้อมูลที่จัดเก็บในฐานข้อมูลเช่นเดียวกับข้อมูลที่จัดเก็บในฐานข้อมูลเชิงสัมพันธ์แบบปกติ แต่หากเพียงข้อมูลนี้ถูกแยกออกไปอยู่ในส่วนที่แตกต่างกันของเวลาและแตกต่างกันของระดับชั้นความปลอดภัย

C_i คือระดับความปลอดภัยของข้อมูล เป็นตัวบอกว่าเป็นข้อมูลที่ได้นำมาจากระดับใด เนื่องจากในฐานข้อมูลแบบมีระดับความปลอดภัยหลายระดับนี้ สามารถนำข้อมูลจากระดับอื่นมาเป็นของระดับตนได้ จึงมีการกำกับว่าข้อมูลนี้ได้นำมาจากระดับใด และทำให้มีคุณสมบัติส่งผ่านข้อมูลได้โดยอัตโนมัติผ่านการระบุของระดับความปลอดภัยนั่นเอง

TC (Tuple-Class) หรือเป็นระดับความปลอดภัยในระดับแถวกล่าวคือเป็นตัวบ่งบอกว่าเป็นข้อมูลในระดับใด ซึ่งเป็นเจ้าของโดยระดับนั้น แตกต่างจาก C_i ที่ระบุถึงระดับของข้อมูลที่ดึงมาหรือเป็น t[TC] คือเป็นข้อมูลที่ได้รับการยอมรับในระดับต่ำกว่า

CC (Creator-Class) เป็นตัวระบุถึงผู้ที่ทำการสร้างแถวว่าเป็นระดับใดเป็นผู้สร้าง เราได้เพิ่มความสามารถนี้ขึ้นมาเพื่อให้ระดับที่สูงกว่า สามารถส่งผ่านข้อมูลให้ระดับล่างได้

การเพิ่มของแอททริบิวเพื่อรองรับคุณสมบัติของความปลอดภัยหลายระดับนั้นทำให้มีการเพิ่มขึ้นของข้อมูลเป็นผลจากการแบ่งแยกข้อมูลแต่ละระดับชั้นออกจากกันในระดับแถว แต่ละมุมมองของผู้ใช้แต่ละระดับจะมองข้อมูลเดียวกันอาจมีข้อมูลที่เหมือน หรือแตกต่างกัน และอาจมีการใช้ข้อมูลร่วมกันในบางส่วนก็ได้ ซึ่งทั้งหมดนี้เป็นตารางเดียวกัน ถ้ามองข้อมูลเพียงระดับเดียวโดยไม่นับคุณสมบัติทางด้านเวลาก็จะเหมือนกับฐานข้อมูลเชิงสัมพันธ์ปกติ หรือพูดได้ว่าข้อมูลเดียวกันที่จำแนกด้วยคีย์หลักแอพพาเรนท์สามารถมากซ้ำกันที่สุดเท่ากับจำนวนระดับชั้นความปลอดภัยที่มีอยู่ทั้งหมด แตกต่างกับความสามารถทางด้านเวลา ที่มีได้ไม่จำกัดจำนวนของแถวกับจำนวนระดับชั้นความปลอดภัย แต่ข้อมูลจะถูกจำกัดจำนวนแถวขึ้นอยู่กับหน่วยเล็กที่สุดของคาบเวลาที่สามารถบันทึกได้

VT_S (Valid Time Start) เป็นจุดเริ่มของคาบเวลาที่เชื่อว่าข้อมูลนั้นเป็นจริง

VT_E (Valid Time End) เป็นจุดสิ้นสุดของคาบเวลาที่เชื่อว่าข้อมูลนั้นเป็นจริง โดยไม่รวมจุดเวลาดังกล่าวเข้าไปด้วย

TT_S (Transaction Time Start) เป็นจุดเริ่มต้นของคาบเวลาที่ได้เริ่มทำการบันทึกข้อมูลนี้เข้าไป

TT_E (Transaction Time End) เป็นจุดสิ้นสุดของคาบเวลาปกติแล้วถ้าหากข้อมูลนั้นยังเป็นจริงอยู่ จะมีค่าเป็น "ตลอดกาล" ถ้าหากข้อมูลนั้นถูกแก้ไขจุดสิ้นสุดของเวลาจะถูกเปลี่ยนเป็นเวลาปัจจุบันขณะทำการแก้ไข

การทำเช่นนี้ทำให้เราสามารถมีข้อมูลที่แบ่งตามระดับชั้น ทั้งยังสามารถค้นหาข้อมูลในแต่ละช่วงเวลา แน่นอนว่างานของเรานั้นเป็นงานที่ประกอบด้วยกันหลายความเชื่อ และอาจเปลี่ยนแปลงได้ตลอดเวลา การบันทึกข้อมูลเดิมที่เคยบันทึกเอาไว้จึงมีความสำคัญ อีกทั้งเราได้ให้สิทธิ์ให้ผู้ใช้ทำการแก้ไขข้อมูลได้ด้วย ดังนั้นการตรวจสอบข้อมูลที่ได้เคยบันทึกไว้ในอดีต โมเดลนี้ก็ยังสามารถรองรับได้ด้วยเช่นกัน

ข้อมูลในตารางเอ็มเอสบีดีนี้ประกอบด้วยความหลากหลายข้อมูลในตารางเดียวกัน อย่างไรก็ดีสามารถจำแนกข้อมูลออกมาให้มีความหมายดังเช่นฐานข้อมูลเชิงสัมพันธ์ปกติได้ การแบ่งข้อมูลออกเป็นหลายมิติ มิติหนึ่งคือมิติของความหลายระดับของความปลอดภัย ที่มีการกำหนดให้ผู้ใช้แต่ละคนมีได้หนึ่งระดับ และสามารถมีการใช้ข้อมูลร่วมกับระดับอื่นได้ และอีกมิติหนึ่งคือมิติของเวลาเพื่อย้อนไปดูรายละเอียด ที่เต็มไปด้วยความข้อเท็จจริงที่สามารถเปลี่ยนแปลงได้เสมอ ภายใต้ข้อมูลที่แตกต่างกันในแต่ละระดับ การใช้คาบเวลาเป็นตัวกำหนดข้อเท็จจริง และความเชื่อเพื่อบ่งบอกการเปลี่ยนแปลงรายละเอียด แล้วนำมาวิเคราะห์ความน่าเชื่อถือหรือความไว้วางใจต่อข้อมูลหรือต่อบุคคล

4.2 ความหมายของข้อมูลในตารางเอ็มเอสบีดี

ข้อมูลในตารางที่เป็นเอ็มเอสบีดีมีการแบ่งข้อมูลเป็นสองส่วนสำคัญคือ ส่วนของรายงานเป็นส่วนที่ยึดถือกฎดั้งเดิมของระบบฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับก็คือกฎของ เบลและลาพาดูลา ที่ยอมให้มองเห็นได้เฉพาะข้อมูลในระดับของตนเองและระดับต่ำกว่า และอีกข้อหนึ่งคือการเปลี่ยนแปลงแก้ไขข้อมูลสามารถทำได้ที่ระดับตนเองหรือสูงกว่าเท่านั้น ถ้า t เป็นแถวของข้อมูลในตารางเอ็มเอสบีดีแล้ว ส่วนของรายงานเราได้กำหนดไว้ว่าให้ t[TC] = tc และ t[CC] = cc โดย tc และ cc เป็นระดับความปลอดภัยภายใต้โมเดลเอ็มเอสบีดี จะเป็นรายงานก็ต่อเมื่อ tc = cc นั่นคือผู้ใช้ในระดับนั้นเองเป็นผู้สร้างข้อมูล และยอมรับข้อมูลนั้นสำหรับระดับตนเอง อย่างไรก็ตามข้อมูลในส่วนรายงานนี้เปรียบเสมือนกับได้แยกออกมาอย่างชัดเจนจากข้อมูลที่เป็นคำสั่ง โดยให้ t'[TC] = tc' และ t'[CC] = cc' ในคือข้อมูลนี้เป็นคำสั่งก็ต่อเมื่อ tc' < cc' ข้อมูลคำสั่งมีจุดประสงค์ให้สร้างความสอดคล้องของข้อมูล ให้มีประสิทธิภาพการใช้งานในความแตกต่างระดับดังเช่นผู้บังคับบัญชาออกคำสั่งให้ผู้ใต้บังคับบัญชา

ตัวอย่าที่เราใช้ประกอบการอธิบายเราได้ยกตัวอย่างงานติดตามบุคคลจากสำนักงานนักสืบแห่งหนึ่งใช้ในการกำหนดงานด้วย JOB_ID ด้วยการระบุผู้ที่ให้ทำการติดตาม (TNAME ย่อมาจาก "Target Name") ทั้งสองแถวในตารางที่ 4.1 นี้เป็นข้อมูลชนิดคำสั่ง จากระดับความปลอดภัยเราได้กำหนดไว้ให้ S1 > C1 > U1 และข้อมูลภายในตารางล้วนมี t[TC] < t[CC] ซึ่งทั้งสองแถวนี้เป็นการถ่ายทอดคำสั่งจากระดับ S1 ให้นักสืบระดับ C1 และระดับ U1 ให้ทำการเฝ้าติดตาม JACK ROBIN ตั้งแต่วันที่ 10 กันยายน เรื่อยไปโดยไม่มีกำหนดยกเลิก นักสืบระดับ C1 สามารถมองเห็นได้เพียงแถวเดียวคือแถวบน และนักสืบระดับ U1 มองเห็นเพียงแถวล่างแถวเดียว แม้ว่าระดับ C1 สูงกว่าระดับ U1 แต่ไม่สามารถมองเห็นแถวล่างได้เนื่องจากเป็นคำสั่งจากระดับที่สูงกว่าให้แก่ U1 ดังนั้น C1 จึงไม่มีสิทธิ์รับรู้ถึงข้อมูลในส่วนนี้ คำสั่งในแถวบนได้ออกคำสั่งไว้ตั้งแต่วันที่ 5 กันยายน ดังที่ระบุไว้ใน t[TT_S] เป็นวันที่เริ่มนำข้อมูลเข้าสู่การบันทึก ดังนั้นข้อมูลนี้เมื่อวันที่ 5 เปรียบเสมือนแผนการที่สั่งการไว้ล่วงหน้า และที่ t[TT_E] ได้ระบุไว้เป็นตลอดกาลคือเป็นวันที่มากที่สุดเท่าที่เป็นไปได้ ซึ่งก็คือ 9999-12-31 มีความหมายข้อมูลที่ได้รับการบันทึกนี้ยังเป็นจริงอยู่คนถึงปัจจุบัน

ตารางที่ 4.1 ข้อมูลการออกคำสั่งภายใต้โมเดลเอ็มเอสบีดี

JOB_ID	*JI_C*	*T NAME*	*TFN_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
337	C1	JACK ROBIN	C1	2006-09-10	9999-12-31	2006-09-05	9999-12-31	C1	S1
337	U1	JACK ROBIN	U1	2006-09-10	9999-12-31	2006-09-10	9999-12-31	U1	S1

ในแต่ละข้อมูลในตารางความสามารถในการมองถูกจำกัดไว้ขึ้นอยู่กับระดับความปลอดภัย ข้อมูลที่ได้รับอนุญาตเท่านั้นให้มองเห็นได้ โดยแนวความคิดในเรื่องมีความคิดเกี่ยวกับการตีความของแต่ละแถวเป็นดังนี้

1. ข้อมูลที่ยอมรับโดยผู้ใช้ที่ระดับหนึ่งนั้น ประกอบด้วยกันสองส่วนคือ ข้อมูลที่เป็นเจ้าของโดยผู้ใช้เอง และข้อมูลที่ยืมมาจากผู้ใช้ในระดับต่ำกว่า ซึ่งข้อมูลที่ยืมมาจากระดับต่ำกว่านี้สามารถที่จะเปลี่ยนแปลงได้ขึ้นกับผู้ใช้ในระดับต่ำกว่าซึ่งเป็นเจ้าของข้อมูลทำการเปลี่ยนแปลงข้อมูล การยืมข้อมูลเหมือนกับเป็นการไว้วางใจให้กับข้อมูลในระดับต่ำกว่าไม่ว่าจะมีการเปลี่ยนแปลงแปลงไปอย่างไรก็ทำการแก้ไขตามไปด้วยอย่างอัตโนมัติ

2. ข้อมูลที่ผู้ใช้สามารถเห็นได้นั้นเป็นข้อมูลในระดับของผู้ใช้เอง และในระดับที่ต่ำกว่า แต่ผู้ที่สร้างแถวนั้นจะต้องมีระดับของผู้ใช้เองหรือต่ำกว่าด้วย ตัวอย่างเช่นการออกคำสั่งจากระดับสูงกว่า จากตารางที่ 4.1 แม้ว่าในแถวล่างเป็นข้อมูลของระดับ U_1 ที่ระบุโดย t[TC] และ U_1 สามารถมองเห็นได้ แต่แถวล่างนี้ไม่สามารถมองเห็นได้ในระดับ C_1 เนื่องเป็นข้อมูลที่ถูกสร้างขึ้นโดยระดับ S_1 ซึ่งเป็นระดับสูงกว่าและไม่ได้เป็นข้อมูลในระดับของตน กรณีเช่นนี้มีเฉพาะในข้อมูลลักษณะที่เป็นคำสั่งถ่ายทอดลงมาในระดับล่างเท่านั้น

3. ข้อมูลในแถวของระดับหนึ่งประกอบไปด้วยข้อมูลที่ได้รับการยอมรับ (ทั้งที่เป็นเจ้าของข้อมูลเองและยืมมาจากระดับต่ำกว่า) โดยผู้ใช้ระดับนั้นที่ไม่มีข้อมูลของแถวอื่นอยู่ในระดับของตนเองหมายถึงข้อมูลนั้นไม่ได้รับการยอมรับโดยผู้ใช้ในระดับนั้น

ในตารางที่ 4.2 เป็นตารางที่ใช้รายงานการเฝ้าติดตามบุคคลที่มีจุดประสงค์เพื่อเก็บข้อมูลของผู้ที่ถูกให้ติดตาม ได้แก่ ชื่อผู้ถูกให้ติดตาม (TNAME), การกระทำในขณะนั้น (ACTION), และสถานที่ (LOCAT) ตารางนี้เป็นตารางเฉพาะรายงานดังนั้นไม่จำเป็นต้องมีคอลัมน์ CC ก็ได้ พิจารณาถึงข้อมูลการติดตามของ JACK ROBINได้มีข้อมูลของบุคคลดังกล่าวอยู่ในสามระดับ ได้แก่ S1, C1 และ U1 ในแถวแรกบอกได้ว่าเมื่อวันที่ 15 กันยายน 2006 ระดับ S1 ได้รับการบันทึกให้เชื่อว่าตั้งแต่วันที่ 10 กันยายน เป็นต้นมาจนถึงปัจจุบันว่า JACK BOBIN ได้พักผ่อนอยู่ที่สนามกอล์ฟอัลไพน์ โดยข้อมูลทั้งหมดได้ดึงข้อมูลจากระดับ U1 ขึ้นมาดูได้จาก t[AT_C] และ t[LC_C] เป็นตัวระบุถึงระดับของข้อมูลว่าเป็นข้อมูลมาจากระดับ U1 แต่ในวันที่ 16 ได้มีการแก้ไขข้อมูลให้ข้อมูลนี้สิ้นสุดลง ถ้าเทียบกับฐานข้อมูลแบบปกติคือถูกลบทิ้งในวันที่ 16 นั่นเอง และมีความเชื่อใหม่แสดงแถวที่สองเป็นคาบเวลาการบันทึกที่ต่อเนื่องกันโดยมี t[TT_E] ของแถวแรกตรงกับ t[TT_S] ในแถวที่สอง คาบเวลาที่มีความเชื่อใหม่นี้อยู่ในช่วงเวลาตั้งแต่วันที่ 10 ถึง 13 กันยายน

เท่านั้นที่ยังเป็นความเชื่อเดิมสำหรับสถานที่ติดตาม "Alpine course" แถวที่สามได้มีการเปลี่ยนความเชื่อจากที่เคยเชื่อในระดับ U1 เป็นระดับ C1 ในส่วนของสถานที่ติดตามไปเป็น "Amari Hotel" ตั้งแต่วันที่ 13 เป็นต้นไป ดังนั้นในตอนนี้มีข้อมูลที่ยังถือว่าได้รับการบันทึกว่าเป็นข้อมูลที่เป็นจริงสำหรับ S1 อยู่สองแถวคือแถวที่สองและแถวที่สาม โดยดูได้จาก t[TT_E] ยังมีคุณสมบัติเป็นจริงจนถึงปัจจุบัน เห็นได้ว่าเปรียบเสมือนว่าแอททริบิว LOCAT มีการเปลี่ยนแปลงข้อมูลที่เวลาๆ หนึ่งก็ทำการสร้างแถวใหม่ขึ้นมา เป็นสิ่งที่แสดงให้เห็นว่าเราสามารถให้คุณสมบัติของเวลาแก่แต่ละแอททริบิวได้อย่างอิสระ สิ่งที่ทำให้ข้อมูลในตารางนี้เป็นเช่นนี้เกิดจากคำสั่ง "UPLEVEL" ที่จะได้อธิบายการใช้งานในบทถัดไป เป็นการยืนยันความเชื่อจากระดับ C1 ในวันที่ 13 ทำให้ข้อมูลในระดับ S1 เปลี่ยนและเพิ่มแถวใหม่มาถึงสองแถวด้วยกัน และจะไม่มีการลบข้อมูลเดิมออก ดังเช่นในแถวแรกนั่นเอง

ตารางที่ 4.2 ข้อมูลติดตามบุคคลในโมเดลเอ็มเอสบีดี

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
JACK ROBIN	U1	relax	U1	Alpine course	U1	2006-09-10	9999-12-31	2006-09-15	2006-09-16	S1
JACK ROBIN	U1	relax	U1	Alpine course	U1	2006-09-10	2006-09-13	2006-09-16	9999-12-31	S1
JACK ROBIN	C1, U1	relax	U1	Amari Hotel	C1	2006-09-13	9999-12-31	2006-09-16	9999-12-31	S1
JACK ROBIN	C1	relax	C1	Amari Hotel	C1	2006-09-10	9999-12-31	2006-09-11	9999-12-31	C1
JACK ROBIN	U1	relax	U1	Alpine course	U1	2006-09-10	9999-12-31	2006-09-11	9999-12-31	U1

4.3 กฎข้อบังคับในเอ็มเอสบีดี

กฎข้อบังคับ (Integrity Constraints) ในเอ็มเอสบีดีเป็นส่วนควบคุมไม่ให้เกิดความกำกวมของข้อมูลภายในตาราง แต่ที่พิเศษนอกเหนือจากการควบคุมการเกิดข้อมูลซ้ำซ้อนจากการมีระดับความปลอดภัยหลายระดับ และคุณสมบัติของเวลาร่วมด้วยแล้ว กฎที่เกิดจากคำสั่งพิเศษที่ใช้ในการดึงข้อมูลมาในระดับความปลอดภัยของตนเพื่อสร้างความสอดคล้องของข้อมูลภายในตารางก็เป็นเรื่องที่จะได้กล่าวต่อไปนี้ด้วย

4.3.1 กฎข้อบังคับที่จำเป็นในเอ็มเอสบีดี

มีคุณสมบัติของอินทีกริตตีสำหรับเอ็มเอสบีดีอยู่ด้วยกันห้าส่วน โดยเดิมที่ได้มีการกล่าวไว้ในส่วนของความปลอดภัยหลายระดับของ [5] เราได้ทำการแก้ไขให้มีความเหมาะสมกับโมเดลของเราได้แก่

- กฎข้อบังคับเอนติตี (Entity Integrity)
- กฎข้อบังคับการเกิดข้อมูลซ้ำซ้อน (Polyinstantiation Integrity)
- กฎข้อบังคับของความเชื่อที่มีต่อข้อมูลระดับต่ำกว่า (Belief-based Integrity)

- กฎข้อบังคับฟอเรจ์นคีย์ (Foreign Key Integrity)
- กฎข้อบังคับการอ้างถึงข้อมูลต่างตาราง (Referential Integrity)

เราได้ทำการปรับปรุงมากจากโมเดลเอ็มแอลอาร์ นอกจากการปรับให้กระชับเข้าใจง่ายยังเพิ่มคุณสมบัติของการป้องกันการซ้อนทับกันของเวลาเข้าไปด้วย

4.3.2 กฎข้อบังคับเอนติตี (Entity Integrity)

ให้ A_k เป็นคีย์หลักแอพพาเรนท์ของตาราง R โดยมีข้อมูลเป็น r เป็นไปตามกฎข้อบังคับเอนติตี ก็ต่อเมื่อทุกๆ แถว $t \in r$

1. $A_i \in A_k \Rightarrow t[A_i] \neq \text{null}$
2. $A_i, A_j \in A_k \Rightarrow t[C_i] = t[C_j]$ (นั่นคือ A_k มีระดับความปลอดภัยเดียวกัน)
3. t[VT_S] $\leq$ t[VT_E] และ t[TT_S] $\leq$ t[TT_E]

ในข้อกำหนดแรกนั้นมาจากโมเดลเชิงสัมพันธ์มาตรฐานซึ่งทำให้แน่ใจได้ว่าไม่มีแถวใดในตารางเป็นค่านัลในทุกแอททริบิวของคีย์หลักแอพพาเรนท์ A_k

ข้อกำหนดที่สองคือทุกแอททริบิวใน A_k มีระดับความปลอดภัยตรงกันทั้งหมดในแถวหนึ่งๆ ทำให้แน่ใจได้ว่า A_k เกิดจากเอนติตีที่มีพื้นฐานเดียวกัน

ในกฎข้อที่สามเป็นการกำหนดให้จุดสิ้นสุดคาบเวลาต้องเกิดหลังหรือพร้อมกับจุดเริ่มต้น การที่จุดเริ่มและสิ้นสุดตรงกันเปรียบเสมือนข้อมูลนั้นไม่มีช่วงเวลาที่เป็นจริงเลย จึงเป็นเหมือนขยะส่วนเกินที่ไม่มีความหมายใดๆ ในตาราง R

4.3.3 กฎข้อบังคับการเกิดข้อมูลซ้ำซ้อน (Polyinstantiation Integrity)

ข้อมูลในตาราง R เป็นไปตามกฎข้อบังคับการเกิดข้อมูลซ้ำซ้อนก็ต่อเมื่อ $1 \leq i \leq n$ โดยมี A_1 คีย์หลักแอพพาเรนท์ ในกรณีถ้าไม่มีเวลามาเกี่ยวข้องเราสามารถกำหนดได้ว่า A1, TC, CC $\rightarrow$ Ai, Ci เมื่อได้รวมกับคุณสมบัติของไบเทมโปรัลแล้ว เป็นไปตามกฎข้อบังคับการเกิดข้อมูลซ้ำซ้อนก็ต่อเมื่อต้องไม่มีสองแถว t และ t' โดย $t \neq t'$ นั่นคือไม่ได้เป็นแถวเดียวกัน

t[A1, TC, CC] = t'[A1, TC, CC] $\wedge$ t[VT_S, VT_E] overlap t'[VT_S, VT_E]

$\wedge$ t[TT_S, TT_E] overlap t'[TT_S, TT_E]

ในกฎข้อบังคับนี้ในประโยคแรกเป็นการกำหนดให้ที่ระดับหนึ่ง และเป็นข้อมูลประเภทหนึ่งไม่ว่าจะเป็นคำสั่งหรือว่ารายงาน สามารถมีคีย์หลักแอพพาเรนท์ได้เพียงตัวเดียว และในสองประโยคถัดมาเป็นการกำหนดให้ข้อบังคับในประโยคแรกเกิดได้เพียงหนึ่ง ณ เวลาหนึ่งเท่านั้นคือต้องไม่มีแถวอื่นใดที่ซ้อนทับกันโดยมีคีย์หลักแอพพาเรนท์ตรงกันระดับเดียวกัน และเป็นคำสั่งหรือรายงานเหมือนกันได้

ตารางที่ 4.3 ข้อมูลบางส่วนในตาราง DETECTIVES ที่ละเมิดกฎข้อบังคับ

CODE NAME	*CN _C*	*FULLNAME*	*FN _C*	*DEPARTMENT*	*D _C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
SHERRY	U2	TINA STONE	U2	WATCH SECTION	U2	2006-10-05	2006-11-10	2006-10-01	9999-12-31	U2	U2
SHERRY	U2	TINA STONE	U2	BOMB SQUAD	U2	2006-11-01	9999-12-31	2006-10-01	9999-12-31	U2	U2

ตารางที่ 4.3 เป็นตารางที่เก็บข้อมูลของนักสืบในที่นี้คือเป็นผู้ใช้ระบบฐานข้อมูลเอ็มเอสบีดีประกอบไปด้วยชื่อเรียกสายลับ (CODNAME), ชื่อจริง (FULLNAME) และหน่วยงาน (DEPARTMENT) จากกฎข้อบังคับเราไม่ต้องการให้เกิดความซ้ำซ้อนและกำกวมของข้อมูล จากข้อมูลภายในตารางมีข้อมูลของนักสืบ SHERRY อยู่ด้วยกันสองแถวซึ่งทั้งสองแถวได้บ่งบอกว่าข้อมูลได้มีการบันทึกพร้อมกันภายใต้เวลาทรานเซคชันในวันที่ 1 ตุลาคม 2006 และยังคงเป็นเชื่อว่าการบันทึกนี้ถูกต้องจนถึงปัจจุบัน แต่ข้อมูลของนักสืบ SHERRY นี้มีคาบเวลาวาลิดที่อ้างว่าได้อยู่ที่หน่วยงาน WATCH SECTION ตั้งแต่วันที่ 5 ตุลาคม 2006 จนถึง 10 พฤศจิการยน 2006 ในแถวแรกซึ่งซ้อนทับกับข้อมูลนักสืบ SHERRY ในช่วงต้นเดือนพฤศจิการยนที่ยังทำงานในหน่วยงาน BOMB SQUAD ดังแสดงในแถวที่สองอีกด้วย การมีคาบเวลาทับซ้อนทำให้เกิดข้อมูลซ้ำซ้อนกันสองแถวในส่วนของคอลัมน์ CODENAME และ FULLNAME และเกิดความกำกวมของข้อมูลที่ไม่ตรงกันของหน่วยงานที่ทำในช่วงต้นเดือนพฤศจิการยนอีกด้วยทำให้ไม่สามารถสรุปได้ว่านักสืบ SHERRY ได้ทำงานในหน่วยงาน "WATCH SECTION" หรือ "BOMB SQUAD" ดังนั้นข้อมูลในตาราง 4.3 นี้ไม่ละเมิดกฎข้อบังคับการเกิดข้อมูลซ้ำซ้อน

4.3.4 กฎข้อบังคับของความเชื่อที่มีให้ต่อข้อมูลระดับล่าง (Belief-based Integrity)

การป้องกันความสัมพันธ์ของข้อมูลระหว่างระดับชั้นให้มีความสอดคล้องกัน เป็นส่วนหนึ่งที่ทำให้ข้อมูลไม่เกิดความกำกวมอันเนื่องมากจากข้อมูลที่ส่งผลกระทบต่อกันระหว่างระดับชั้น เนื่องจากในโมเดลเอ็มเอสบีดีมีการยอมให้ข้อมูลที่ระดับต่ำกว่าสามารถกระจายข้อมูลขึ้นสู่ระดับสูงกว่าได้โดยอัตโนมัติ มีกฎข้อบังคับเป็นให้ r เป็นข้อมูลในตารางและเป็นไปตามกฎข้อบังคับของความเชื่อที่มต่อขอ้มูลระดับล่างก็ต่อเมื่อทุกๆ แถว $t \in r$ และ $1 \leq i \leq n$, ถ้า $t[A_i] \neq \text{null} \wedge t[C_i] < t[TC]$ และมี $t' \in r$ ที่ $t'[A_i] = t[A_i] \wedge t'[\text{TC}] \in t[\text{C}_1] \wedge t'[TC] = t'[C_i] = t[C_i] \wedge t'[CC] = t[CC]$

$\wedge$ $t'[A_i] = t[A_i]$ $\wedge$ t'[VT_S, VT_E] cover t[VT_S, VT_E] $\wedge$ t'[TT_S, TT_E] cover t[TT_S, TT_E]

ทุกข้อมูลที่มีการยืมมาจากระดับต่ำกว่าต้องสอดคล้องกันคือ ต้องมีข้อมูลอยู่ในระดับล่างจริงตลอดช่วงเวลาที่มีการยืมข้อมูลนั้น

ตารางที่ 4.4 ข้อมูลบางส่วนในตาราง DETECTIVES ที่ละเมิดกฎข้อบังคับ

CODE NAME	*CN _C*	*FULLNAME*	*FN _C*	*DEPARTMENT*	*D _C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
KORN	U2,C2	JANE	U2	WATCH SECTION	C2	2006-11-01	9999-12-31	2006-10-05	9999-12-31	S1	S1
KIR	C1	NULL	U2	WATCH SECTION	C1	2006-11-01	9999-12-31	2006-10-05	9999-12-31	C1	C1
KORN	C2	JOAN	C2	WATCH SECTION	C2	2006-11-01	9999-12-31	2006-10-05	2006-11-05	C2	C2
TEQUILA	U1	JACK	U1	POLICE OFFICE	U1	2006-11-01	9999-12-31	2006-11-01	9999-12-31	U1	U1

เราขออธิบายด้วยตัวอย่างจากตางราง DETECTIVES ในตารางที่ 4.4 เนื่องจากการกำหนดให้มีระดับความปลอดภัยในระดับแอททริบิวนอกเหนือจากระดับแถว ทำให้เราสามารถสร้างความสอดคล้องระหว่างเอนติตีที่อยู่ต่างระดับความปลอดภัยกับแอททริบิวอื่นนอกเหนือจากคีย์หลักแอพพาเรนท์ได้ ความสอดคล้องที่หมายถึงนี้คือการมีข้อมูลในระดับต่ำกว่าอยู่แถวที่ตนเองยอมรับ จากข้อมูลในตารางเป็นตารางที่มีข้อมูลของนักสืบด้วยกันสี่ระดับ ในแถวแรกข้อมูลของนักสืบ KORN ในคอลัมน์ FULLNAME และ DEPARTMENT เป็นข้อมูลที่ดึงมาจาก U2 และ C2 ตามลำดับ มีหน่วยงาน WATCH SECTION ที่ได้มีการดึงข้อมูลมาจากระดับ C2 ซึ่งสอดคล้องกับกฎข้อบังคับคือมีข้อมูลนี้อยู่จริงในระดับ C2 และมีข้อมูลตรงกัน แต่อย่างไรก็ดีไม่มีอยู่จริงตลอดช่วงเวลาที่มีการดึงข้อมูลคือหลังจากวันที่ 5 พฤศจิกายนไปแล้วข้อมูลของ KORN ในระดับ C2 จะไม่มีอยู่จริงดังที่ระบุในคอลัมน์ TT_E ดังนั้นจึงไม่สอดคล้องกับเวลาของการบันทึก และในคอลัมน์ FULLNAME นั้นได้มีการระบุว่ามีชื่อ JANE ซึ่งเป็นข้อมูลที่ดึงมาจากระดับ U2 แต่กลับไม่มีแถวข้อมูลของ KORN ในระดับ U2 อยู่จริงดังนั้นในส่วนนี้ก็เป็นการละเมิดข้อบังคับ

สำหรับแถวที่สองนักสืบ KIR มีการดึงข้อมูล FULLNAME มาจาก U2 มีค่าเป็น NULL และไม่มีข้อมูล KIR ในระดับ U2 อยู่เลยก็ไม่ได้เป็นการละเมิดกฎข้อบังคับแต่อย่างใดเพราะการระบุเป็น NULL ถือว่าไม่มีข้อมูลในระดับล่าง หรือถ้ามีอยู่ก็ต้องเป็น NULL เช่นเดียวกัน

4.3.5 กฎข้อบังคับฟอเรจ์นคีย์ (Foreign Key Integrity)

การกำหนดข้อบังคับฟอเรจ์นคีย์มีกฎที่คล้ายคลึงกฎข้อบังคับเอนติตีดังนี้

1. ฟอเรจ์นคีย์เป็น NULL หรือไม่เป็นก็ได้
2. ทุก $A_i, A_j \in FK$ ต้องเป็นระดับความปลอดภัยเดียวกันทั้งหมด

ในข้อกำหนดแรก เหมือนกับในฐานข้อมูลเชิงสัมพันธ์ปกติ ส่วนในข้อที่สองเป็นเหมือนข้อบังคับเอนติตี คือกำหนดให้สมาชิกของคีย์หลักแอพพาเรนท์มีระดับความปลอดภัยเท่ากันทั้งหมด

4.3.6 กฎข้อบังคับการอ้างถึงข้อมูลต่างตาราง (Referential Integrity)

ในกรณีที่ฟอเรจ์นคีย์เป็นนัลกำหนดให้ r_1 เป็นข้อมูลในตารางที่ทำการอ้างและ r_2 เป็นข้อมูลในตารางที่ถูกอ้างถึงโดย $t_1 \in r_1$ และมี $t_2 \in r_2$ นั้นจะเป็นไปตามกฎข้อบังคับก็ต่อเมื่อ สำหรับทุก $t_1[FK] \neq null$ มี

1. $t_1[FK] = t_2[AK]$
2. $t_1[TC] = t_2[TC] \wedge t_1[CC] = t_2[CC]$
3. t_2[VT_S, VT_E] cover t_1[VT_S, VT_E] $\wedge$ t_2[TT_S, TT_E] cover t_1[TT_S, TT_E]

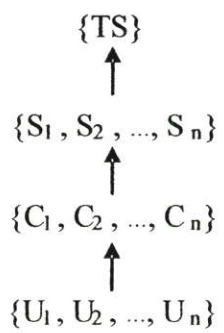
ให้ TC และ CC เท่ากันนั้นหมายความว่าให้ยอมรับ และเป็นผู้ที่สร้างเฉพาะข้อมูลของระดับเดียวกันเท่านั้น ในข้อสุดท้ายตัวที่อ้างถึงข้อมูลที่ถูกอ้างต้องอยู่ในช่วงเวลาของตัวถูกอ้าง เนื่องจากต้องมีข้อมูลที่อ้างถึงอยู่จริง

บทที่ 5
ภาษาเอสคิวแอลสำหรับโมเดลเอ็มเอสบีดี

ในบทนี้กล่าวถึงการใช้งานของฐานข้อมูลในงานวิจัยของเรา เพื่อให้แสดงถึงความสามารถของฐานข้อมูลนี้ เราได้ยกตัวอย่างประกอบในเรื่องราวเกี่ยวกับสำนักงานนักสืบเพื่อให้ต่อเนื่องจากบทก่อนๆ และทำให้ง่ายแก่การอธิบายและเข้าใจ หากเราต้องการฐานข้อมูลที่ข้อมูลสามารถเปลี่ยนแปลงได้บ่อยครั้ง และมีข้อมูลที่ต้องการปิดเป็นความลับที่แตกต่างจากความเข้าใจของบุคคลอื่นๆ เอ็มเอสบีดี เป็นระบบที่เหมาะสมเป็นอย่างยิ่ง

5.1 การกำหนดระดับชั้นความปลอดภัย

สำนักงานนักสืบ ที่ทำงานการสืบสวนข้อมูลสำคัญมีสายข่าวอยู่มากมาย และหลายหน่วยงานบ่อยครั้งที่มีผู้ไม่ประสงค์ดีต่อหน่วยงาน ทำให้ได้รับข้อมูลที่คลาดเคลื่อนหรือนอกเหนือจากนั้น ยังมีผู้ไม่ประสงค์ดีต้องการนำข้อมูลไปใช้ในทางมิชอบ ดังนั้นทางสำนักงานนักสืบ จำเป็นต้องคอยควบคุมดูแล ข้อมูลที่ได้รับมาจากหน่วยงานต่างๆ ที่รับมาจากสายข่าวอีกทีหนึ่ง เพื่อใช้ในการอธิบายในฐานข้อมูลที่เกี่ยวข้องกับเวลา และมีหลายระดับชั้นนั้นเราพิจารณาถึงการติดตามบุคคลการทำงานในหน่วยงานของสำนักงานนักสืบ ซึ่งแบ่งเป็นหน่วยงานย่อยๆ ดังนี้ หน่วยงานพิเศษ (Special Cell), หน่วยงานป้องกันและต่อต้านผู้ก่อการร้าย (Anti-Terrorist Cell), หน่วยงานเฝ้าระวัง (Watch Section) และหน่วยงานตำรวจ (Police Office) ทั้งหมดแบ่งเป็นกลุ่มย่อยๆ ภายในหน่วยงาน ตั้งแต่ 1 ถึง n กลุ่ม ยกเว้นในระดับ TS มีเพียงกลุ่มเดียวเพื่อให้สามารถควบคุมข้อมูลได้ทั้งหมด เนื่องจากแม้ว่าจะอยู่ในระดับเดียวกัน แต่หากเป็นคนละกลุ่มจะไม่สามารถมองเห็นกันได้ แสดงเป็นระดับชั้นดังรูปที่ 1



รูปที่ 5.1 ระดับชั้นภายในสำนักงานนักสืบ

การแบ่งหน่วยงานตามระดับความปลอดภัยที่ได้กำหนดไว้เพื่อให้สามารถรักษาความลับภายในองค์กร และด้วยหลักการของฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับทำให้นักสืบสามารถมองเห็นข้อมูลในระดับของตนเองหรือระดับต่ำกว่า แต่มีข้อยกเว้นคือ แม้ว่าอยู่ในระดับเดียวกันก็ไม่สามารถมองเห็นกันได้หากเครื่องหมายบอกระดับต่างกัน นั้นกล่าวได้ว่าผู้ใช้ในระดับ Secret (S) มีระดับย่อยเป็น S1 และ S2 ทั้งสองอยู่ในระดับเดียวกันก็ไม่สามารถมองเห็นกันได้ ทำให้ไม่สามารถมองเห็นข้อมูลเกี่ยวข้องกับนักสืบของทั้งสองฝ่าย รวมทั้งข้อมูลการทำงานอีกด้วย

สำนักงานนักสืบมีวิธีการได้มาของข้อมูลด้วยกันหลายทาง คือทำการสืบด้วยตัวเอง หรือนำข้อมูลในระดับล่างมาช่วยในการพิจารณา ในข้อมูลระดับล่างนี้อาจหมายถึงข้อมูลของผู้ใต้บังคับบัญชา ดังนั้นฐานข้อมูลต้องสามารถรองรับการถ่ายทอดคำสั่งไปสู่ระดับต่ำกว่า เพื่อให้ผู้ใต้บังคับบัญชาทำการสืบข้อมูลและรายงานข้อมูลตามคำสั่งได้ แต่เนื่องจากข้อมูลในทุกระดับชั้นต้องไม่มีการเปลี่ยนแปลงข้อมูลโดยบุคคลอื่นที่มีเครื่องหมายบอกระดับต่างกัน ไม่ว่าอยู่ในระดับชั้นเดียวกันหรือต่ำกว่า รวมถึงสูงกว่าก็ไม่สามารถเปลี่ยนแปลงข้อมูลได้ กล่าวได้ว่าทุกระดับชั้นมีสิทธิในข้อมูลของตนเองเต็มที่และไม่มีการก้าวก่ายข้ามระดับชั้นกัน เพื่อไม่ให้เกิดความสับสน

การที่นักสืบในระดับหนึ่งทำการเพิ่มข้อมูลเข้าไปในระดับตนเองในที่นี้เราขอเรียกว่ารายงาน รายงานดังกล่าวเพื่อให้รายงานสามารถบอกช่วงเวลาของการติดตามเป้าหมายได้ด้วย จึงมีคุณสมบัติของเวลาเข้ามาเกี่ยวข้องในที่นี้ด้วย และเนื่องจากความเชื่อที่มีต่อรายงานสามารถเปลี่ยนแปลงได้เสมอ ซึ่งการเปลี่ยนแปลงแต่ละครั้งอาจมีผลกระทบต่อข้อมูลในหลายๆ ระดับ

มีข้อมูลอีกประเภทหนึ่งที่ได้มีการถ่ายทอดไปสู่ระดับล่าง และทำให้ระดับล่างมองเห็นได้ เป็นข้อแตกต่างจากแนวคิดพื้นฐานเดิมของฐานข้อมูลที่มีระดับความปลอดภัยหลายระดับ การถ่ายทอดข้อมูลสู่ในระดับล่างเป็นสิ่งที่ผิด และไม่ควรทำ เราได้รักษากฎนี้ไว้ส่วนหนึ่ง และแยกข้อมูลที่สามารถส่งผ่านได้มาอีกส่วนหนึ่ง เพื่อให้เพิ่มขีดความสามารถในการใช้งานของฐานข้อมูลที่มีหลายระดับ คล้ายกับเราสามารถสื่อสารและประสานงานได้ด้วยตัวข้อมูลในฐานข้อมูล แทนที่เหมือนกับมุมมองที่มีต่อข้อมูลเพียงด้านเดียวคือ ระดับบนมองเห็นระดับล่าง แต่ไม่สามารถทำอะไรได้เลย เพียงแต่พิจารณาข้อมูลในระดับล่างเท่านั้นว่ามีความเชื่อถือมาประกอบว่าน่าเชื่อถือได้มากเพียงใด ตัวอย่างเช่น

ความสามารถในการมองเห็นในเอ็มเอสบีดีนั้น ความแตกต่างจากฐานข้อมูลปกติแล้ว เปรียบเหมือนว่าเรามองเห็นได้เพียงข้อมูลที่เป็นปัจจุบันเท่านั้น และไม่ได้มีการแบ่งระดับหรือมีระดับเดียวเท่านั้น แต่ในเอ็มเอสบีดีเราสามารถกำหนดความสามารถในการมองเห็นที่บ่งบอกถึงเวลาที่ต้องการของข้อมูลที่อ้างอิงกับเวลาในโลกความเป็นจริงหรือเวลาที่บันทึกหรือเคยได้บันทึกเอาไว้ นอกจากนั้นยังสามารถออกเป็นข้อมูลที่เป็นอนาคตคือวางหมายกำหนดการ นอกจากนั้น ข้อมูลยังแบ่งออกเป็นหลายระดับ เราสามารถมองเห็นได้ในระดับของตนเองและระดับต่ำกว่า

ส่วนหนึ่งที่ผู้ใช้ควรจะรู้ถึงความสามารถของระบบฐานข้อมูลนี้และเข้าใจกฎเกณฑ์พื้นฐาน เพื่อให้ผู้ใช้สามารถนำไปใช้ได้ ในฐานะของผู้ใช้หรือผู้ออกแบบฐานข้อมูล เราได้สรุปความสามารถของเอ็มเอสบีดี และขอยกตัวอย่างประกอบคำอธิบายสำหรับงานของตารางนักสืบมีโครงสร้างตารางดังต่อไปนี้

- DETECTIVES (CODENAME, CN_C, FULLNAME, FN_C, DEPARTMENT, D_C, VT_S, VT_E, TT_S, TT_E, TC, CC) เป็นตารางที่บอกรายะละเอียดข้อมูลของนักสืบ ประกอบด้วยชื่อและหน่วยงาน

- TARGETS (TNAME, TN_C, BIRTHDAY, B_C, ADDRESS, A_C, VT_S, VT_E, TT_S, TT_E, TC, CC) เป็นตารางเก็บรายละเอียดของเป้าหมายที่ถูกติดตามประกอบด้วยวันเกิดและที่อยู่

- JOBS(JOB_ID, JI_C, OBJECTIVE, O_C, JOB_TYPE, JT_C, VT_S, VT_E, TT_S, TT_E, TC, CC) มีหน้าที่ กำหนดงาน และข้อมูลที่ต้องการสั่งให้ทำ

- JOB_DET(JOB_ID, JI_C, CODENAME, CN_C, VT_S, VT_E, TT_S, TT_E, TC, CC) กำหนดนักสืบที่รับทำงานงานติดตามนี้อาจมีผู้ร่วมงานหนึ่งๆ หลายคน

- JOB_TAR(JOB_ID, JI_C, TNAME, TN_C) กำหนดเป้าหมายในงานของผู้ถูกติดตามอาจมีผู้ถูกติดตามหลายคนในงานหนึ่งๆ

- TRACKING_REPORTS(TNAME, TN_C, ACTION, AT_C, LOCAT, LC_C, VT_S, VT_E, TT_S, TT_E, TC) เป็นตารางบอกรายละเอียดของเป้าหมายที่ทำการติดตามไม่ว่าจะเป็น การกระทำและสถานที่

ทุกตารางเราได้ใส่คุณสมบัติของเอ็มเอสบีดีเข้าไปด้วย โดยการออกแบบเริ่มจากการออกแบบโดยไม่ต้องคำนึงถึงแอททริบิวเพิ่มเติมพิเศษของเอ็มเอสบีดี ให้ออกแบบในรูปแบบเชิงสัมพันธ์มาตรฐาน หลังจากนั้นก็ทำการใส่แอททริบิวระดับความปลอดภัยในระดับแถว (TC) ผู้สร้างแถว (CC) กำหนดแอทริบิวระดับความปลอดภัยในระดับแอทริบิวข้อมูลในที่นี้เราใช้ตัวย่อของแอททริบิวตามด้วย "_C" เช่นให้ t เป็นแถวในตาราง DETECTIVES แล้ว t[CODENAME] มีระดับความปลอดภัยระดับแอททริบิวเป็น t[CN_C]

ให้มีสำนักงานนักสืบ ที่ประกอบด้วยกลุ่มของนักสืบ ที่อาจจะทำงานกันเป็นกลุ่มการทำงานที่อาจเป็นทีมหรือคณะที่ประกอบด้วยนักสืบหลายคน หรืออาจจะเป็นคนเดียว เราได้กำหนดให้แต่ละกลุ่มของการทำงานนี้ แบ่งออกให้มีระดับความปลอดภัยของตนเอง ตัวอย่างการแบ่งระดับชั้นของหน่วยงานเป็นดังนี้ เรากำหนดกลุ่มของนักสืบทำงานร่วมกัน หมายถึงข้อมูลที่ปฏิบัติงานกันภายในกลุ่มเปิดเผยกันเองและใช้ข้อมูลร่วมกันทั้งหมด เปรียบเสมือนมีเพียงตารางเดียวที่นักสืบทุกคนเข้ามาเพิ่มหรือแก้ไขข้อมูลเดียวกัน

ตารางที่ 5.1 ข้อมูลในตาราง DETECTIVES

CODE NAME	*CN _C*	*FULLNAME*	*FN _C*	*DEPARTMENT*	*D _C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
VODKA	TS	JONATHAN	TS	SPECIAL CELL	TS	2006-11-01	9999-12-31	2006-10-05	9999-12-31	TS	TS
GIN	S1	JANE	S1	WATCH SECTION	S1	2005-02-01	9999-12-31	2005-01-05	2006-10-05	S1	S1
GIN	S1	JANE	S1	WATCH SECTION	S1	2005-02-01	2006-11-01	2006-10-05	9999-12-31	S1	S1
GIN	S1	JANE	S1	ANTI-TERRORIST	S1	2006-11-01	9999-12-31	2006-10-05	9999-12-31	S1	S1
SHERRY	S2	JOANNE	S2	WATCH SECTION	S2	2006-11-01	9999-12-31	2006-10-05	9999-12-31	S2	S2
KIR	C1	JOHN	C1	WATCH SECTION	C1	2006-11-01	9999-12-31	2006-10-05	9999-12-31	C1	C1
KORN	C2	JOAN	C2	WATCH SECTION	C2	2006-11-01	9999-12-31	2006-10-05	9999-12-31	C2	C2
TEQUILA	U1	JACK	U1	POLICE OFFICE	U1	2006-11-01	9999-12-31	2006-11-01	9999-12-31	U1	U1

จากตารางที่ 5.1 เราลำดับเหตุการณ์ของนักสืบ GIN ได้ดังนี้ ในแถวที่สองนักสืบ GIN เริ่มนำข้อมูลเข้าสู่ระบบฐานข้อมูลเมื่อวันที่ 1 กุมภาพันธ์ 2005 โดยเข้าทำงานกับหน่วยงาน WATCH SECTION ในวันที่ 1 กุมภาพันธ์ 2005 เรื่อยไป แต่ในวันที่ 5 ตุลาคม 2006 ได้มีเหตุการณ์ที่ต้องการปรับนักสืบ GIN ให้ย้ายไปอยู่ที่หน่วยงาน ANTI-TERRORIST ในวันที่ 1 พฤศจิกายน 2006 ข้อมูลในแถวที่สองเกิดการเปลี่ยนแปลงข้อมูลที่เคยบันทึกไว้ ดังนั้น t[TT_E] ในแถวที่สองจึงลงเวลาการแก้ไขข้อมูลดังกล่าวไว้เพื่อระบุวันที่สิ้นสุดความเชื่อ ดังนั้นในแถวที่สามเป็นการลงบันทึกประวัติเดิมที่เคยอยู่กับหน่วยงาน WATCH SECTION ซึ่งเป็นข้อมูลที่ยังเป็นจริงอยู่เพราะ GIN ได้ทำงานใน WATCH SECTION จริงจนถึงวันที่ 1 พศจิการยน 2006 แถวที่สี่เป็นข้อมูลที่นักสืบ GIN เปลี่ยนเข้าทำงานใน ANTI-TERRORIST ในวันที่ 1 พฤศจิกายน 2006 และเป็นเช่นนั้นเรื่อยมาจนถึงปัจจุบัน โดยยังไม่มีกำหนดเปลี่ยน เห็นได้ว่าเราสามารถกำหนดเวลาได้อย่างอิสระกับทุกข้อมูลของนักสืบไม่ว่าจะเป็นชื่อจริงหรือว่าเป็นหน่วยงาน เมื่อมีการเปลี่ยนแปลงในแต่ละครั้งจะถูกบันทึกไว้อย่างละเอียด แม้แต่เวลาได้ทำการบันทึก ตลอดจนข้อมูลเดิมที่ถูกเปลี่ยนไปแล้วก็จะถูกเก็บไว้ด้วย ทั้งหมดแสดงในสามแถวจากแถวที่สองถึงแถวที่สี่

5.2 คำสั่ง SELECT

การใช้คำสั่งมีรูปแบบดังต่อไปนี้

[NONSEQUENCED VALIDTIME AND TRANSACTIONTIME]

SELECT A_1[, A_2...]

FROM R

[WHERE p]

[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

คำสั่ง SELECT ใหม่นี้ส่วนที่แตกต่างจากคำสั่งเอสคิวแอลมาตรฐานคือได้เพิ่มอนุประโยคของรูปแบบเวลาไว้ ประกอบด้วย "VALIDTIME", "NOSEQUENCED VALIDTIME", "TRANSACTIONTIME", "NOSEQUENCED TRANSACTIONTIME" เป็นการกำหนดรูปแบบของเวลาที่กำหนดให้ใช้ในการไควรี อนุประโยคเหล่านี้อยู่ภายใต้ "[" และ "]" หมายถึงสามารถระบุหรือไม่ก็ได้ ตัวอย่างเช่น ในงานติดตามบุคคลนักสืบ "GIN" ซึ่งอยู่ในระดับ S1 ต้องการข้อมูลที่ได้รับการยอมรับในระดับตนเองจากตาราง TRACKING_REPORTS แสดงตารางที่ 5.6 ซึ่งเป็นข้อมูลรายงานการติดตามบุคคล เพื่อดูข้อมูลในช่วงเวลาหนึ่งเท่านั้นคือตั้งแต่วันที่ 10 ถึง 15 กันยายน 2006 มีการกำหนดคาบเวลาที่อ้างอิงถึงเวลาการติดตามดังนั้นจึงเป็นการไควรีแบบเวลาวาลิดด้วยการใส่อนุประโยค "VALIDTIME" กำหนดไว้ และอาศัยฟังก์ชันในการตรวจสอบการซ้อนทับของคาบเวลากับคาบเวลาที่ต้องการ

```
VALIDTIME
SELECT *
FROM TRACKING_REPORTS
WHERE VALIDTIME( TRACKING_REPORTS) OVERLAPS PERIOD [ DATE '2006-09-10',
DATE '2006-09-15')
LEVELS me
```

ผลลัพท์จากการไควรีทั้งหมดมีอยู่ด้วยกันสองแถวเราขอแสดงด้วยผลจาก แอพพลิเคชันที่เราต้องการเห็นควรเป็นดังตารางที่ 5.2 ที่ต้องอาศัยการไควรีเพื่อนำข้อมูลที่แยกออกจากกัน ระหว่างแถวที่มีความต่อเนื่องของข้อมูลที่มีเวลาต่อเนื่องกันเช่นใน t[LOCAT] ถูกแยกออกเป็นสองแถวเนื่องจาก t[ACTION] ขาดความต่อเนื่องของเวลา แต่เมื่อนำมาแสดงผลแล้วเป็นดังตารางที่ 5.2 หรือแสดงในลักษณะแอพพลิเคชันในภาคผนวก ก.

ตารางที่ 5.2 ผลลัพท์หลังการไควรีที่เป็นมุมมองของผู้ใช้

TNAME	*ACTION*	*LOCAT*
JACK ROBIN	PLAY GOLF [2006-09-10, 2006-09-13)	ALPINE COURSE [2005-02-01,2006-11-01)
	RELAX [2006-09-13, 9999-12-31)	

5.3 คำสั่ง INSERT

การใช้คำสั่งมีรูปแบบดังต่อไปนี้

[VALIDTIME PERIODS (A_i[,A_j],...) VALUES ([vt_s_i,vt_e_i)[,[vt_s_j,vt_e_j)],...)]
INSERT INTO R(A_i[,A_j]...)
VALUES (a_i[, a_j], ...)
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

คำสั่ง INSERT สามารถกำหนดเวลาวาลิดได้เพื่อใช้ในการรายงานแบบเจาะจงเวลาที่ต้องการได้ ในแต่ละข้อมูลในแอททริบิวว่าให้มีคาบเวลาที่เป็นจริงนั้นเริ่มต้นและสิ้นสุดลงเมื่อไร โดยการออกคำสั่งในประโยค "VALIDTIME" ความแตกต่างระหว่างการออกคำสั่ง และการรายงานคือหากกำหนดในประโยค "LEVELS" มีระดับของผู้ทำการออกคำสั่งรวมอยู่ด้วยคือเป็นรายงาน และถ้ามีระดับอื่นนอกเหนือที่ต่ำกว่าระดับของผู้ใช้ก็จะกลายเป็นคำสั่งไป ดังนั้นการออกคำสั่ง INSERT แต่ละครั้งเป็นได้ทั้งรายงานหรือคำสั่งขึ้นอยู่กับระดับที่ระบุในอนุประโยค LEVELS ให้ R เป็นชื่อตารางโดยมี A_i, A_j เป็นแอททริบิวของข้อมูลที่มีข้อมูล เป็น a_i, a_j และ l_i, l_j เป็นระดับความปลอดภัยที่เท่ากับหรือต่ำกว่าระดับของผู้ใช้คำสั่งเอง

การใช้คำสั่ง INSERT จะทำการเพิ่มแถว t ตามค่าของข้อมูลแอททริบิวที่ได้กำหนดไว้โดยมีระดับความปลอดภัยของแต่ละระดับที่ระบุไว้ในประโยค "LEVELS" แถว t ที่เกิดขึ้นมาใหม่ตามระดับที่กำหนดมีระดับที่ระบุอยู่ใน t[TC] คือให้เป็นข้อมูลที่ยอมรับในระดับนั้น และให้ทุก t ที่ทำการเพิ่มมี $t[CC]$ = ระดับของผู้ที่ใช้คำสั่ง INSERT คือเป็นผู้สร้างแถวขึ้นมา

เมื่อต้องการเพิ่มข้อมูลนักสืบเข้าไปข้อมูลรายชื่อนักสืบ โดยสมมติให้เวลาปัจจุบันนี้คือวันที่ 1 พฤศจิการยน 2006 เมื่อผู้ใช้ในระดับ U1 ทำการเพิ่มข้อมูลของนักสืบ TEQUILA เข้าไป โดยมีชื่อจริงว่า JACK และเข้าบรรจุอยู่ในสำนักงานตำรวจในวันเดียวกันกับที่ได้ออกคำสั่งโดยไม่ได้มีการกำหนดออกจากหน่วยงาน สามารถเขียนเป็นคำสั่งเอสคิวแอลภายใต้โมเดลเอ็มเอสบีดีได้เป็น

```
INSERT INTO DETECTIVES( CODENAME, FULLNAME, DEPARTMENT)
VALUES( 'TEQUILA', 'JACK', 'POLICE OFFICE')
```

เราไม่ได้กำหนดในส่วนของประโยค "VALIDTIME" และ "LEVELS" นั่นหมายถึงเวลาที่เชื่อว่าเป็นจริงนั้นคือเวลาปัจจุบัน และให้เป็นจริงเรื่อยไป สำหรับการไม่ระดับนั้นจะทำการเพิ่มเฉพาะในระดับของตนเอง คำสั่งนี้ไม่แตกต่างคำสั่งในเอสคิวแอลมาตรฐานแต่อย่างใด และข้อมูลนี้เมื่อเรา

มองเพียงระเดียวนะเวลาปัจจุบันก็จะมีข้อมูลที่เหมือนกันด้วยกับการมองจากฐานข้อมูลเชิงสัมพันธ์ ปกติผลจากการออกคำสั่งแสดงในแถวสุดท้ายของตารางที่ 5.1

5.4 คำสั่ง UPDATE

การใช้คำสั่งมีรูปแบบดังต่อไปนี้

[VALIDTIME '[vt_s, vt_e)']
UPDATE *R*
SET $A_i = s_i$[, $A_j = s_j$] …
WHERE *p*
[LEVELS l_i[,l_j]… EXCEPT l_n[,l_m]…]

การใช้คำสั่ง UPDATE ทำโดยผู้ใช้สามารถใช้ได้กับข้อมูลในระดับของตนเองหรือรายงาน และกับระดับต่ำกว่าในลักษณะคำสั่ง คือสามารถแก้ไขข้อมูลได้ที่ตนได้เป็นผู้สร้างเท่านั้นคือดูจาก t[CC] ตรงกันกับระดับความปอลดภัยของผู้ใช้ การทำการอัพเดทข้อมูลโดยระบุชื่อตาราง *R* และแอททริวบิวข้อมูลที่ต้องการอัพเดท A_i, A_j โดยมีข้อมูลใหม่ที่ต้องการแก้ไขเป็น s_i, s_j ตามลำดับ

เมื่อใช้คำสั่งอัพเดทผลของการใช้คำสั่งเป็นดังนี้คือ ในประโยค "LEVELS" หากมีระดับของผู้ที่ออกคำสั่งอยู่คือแก้ไขข้อมูลรายงานในระดับของตน

- แถวที่อยู่ในการพิจารณานอกเหนือเงื่อนไขในประโยค "WHERE" คือต้องเป็นแถวที่มี t[TC] = t[CC] ซึ่งตรงกับระดับความปลอดภัยของผู้ที่ทำการออกคำสั่ง ทำการเปลี่ยนแปลงข้อมูลใหม่ตามข้อมูลใหม่ที่อยู่ในประโยค "SET" และกำหนดให้ระดับความปลอดภัยของแอททริบิวนั้นเป็นระดับของผู้ออกคำสั่ง

หากประโยค "LEVELS" ประกอบด้วยระดับความปลอดภัยอื่นที่ไม่ใช่ระดับตนเองคือแก้ไขคำสั่งในระดับต่ำกว่า

- แถวที่อยู่ในการพิจารณานอกเหนือเงื่อนไขในประโยค "WHERE" คือต้องเป็นแถวที่มี t[TC] ตรงกับระดับความปลอดภัยที่ระบุในประโยค "LEVELS" และมี t[CC] ตรงกับระดับความปลอดภัยของผู้ที่ทำการออกคำสั่ง ทำการเปลี่ยนแปลงข้อมูลใหม่ตามข้อมูลใหม่ที่อยู่ในประโยค "SET" และกำหนดให้ระดับความปลอดภัยของแอททริบิวนั้นเท่ากับ t[TC] ของแถวที่ถูกแก้ไข

หลังจากนั้นหากเป็นเป็นข้อมูลในระดับตนเองไม่ได้ยืมมาจากระดับต่ำกว่า

- ให้ t เป็นแถวระดับบนที่การดึงข้อมูลมาจากแถวที่ถูกอัพเดทและมีคีย์หลักแอพพาเรนท์รวมอยู่ในการอัพเดทด้วย ต้องทำการลบระดับความปลอดภัยของระดับผู้ออกคำสั่งออกจากเซท t[Ck] ถ้าหลังจากลบออกแล้ว t[Ck] = $\varnothing$ ต้องทำการลบแถวนั้นออก อย่างไรก็ตามข้อมูลที่ถูกลบยังคงอยู่ คือถูกบันทึกในช่วงเวลาหนึ่งของเวลาทรานเซคชัน
- ถ้าอัพเดทเฉพาะแอททริบิวอื่น ที่ไม่อยู่ในคีย์หลักแอพพาเรนท์ หากมีระดับสูงกว่าดึงข้อมูลของแถวนี้นี้ไปต้องแก้ไขตามข้อมูลตาม s_t ด้วยเพื่อให้สอดคล้องกับกฎข้อบังคับความเชื่อที่มีต่อข้อมูลในระดับล่าง

การอัพเดทนี้ต้องเป็นไปตามข้อบังคับเอนติตี ความซ้ำซ้อนของข้อมูล และการอ้างถึงข้อมูลต่างตารางด้วย จากตาราง DETECTIVES ซึ่งเป็นข้อมูลของนักสืบในตารางที่ 5.1 มีเหตุการณ์ของนักสืบ GIN อย่างหนึ่งคือการย้ายจากหน่วยงาน WATCH SECTION ไปยังหน่วยงาน ANTI-TERRORIST ในวันที่ 1 พฤศจิกายน 2006 และยังไม่มีกำหนดเปลี่ยนแปลงหลังจากนี้ เป็นการทำการแก้ไขข้อมูลแบบอ้างถึงเวลาวาลิดซึ่งเป็นวันที่ของการย้ายหน่วยงาน ดังนั้นการย้ายหน่วยงานมายังหน่วยงานใหม่จึงเริ่มต้นตั้งแต่วันที่ 1 พฤศจิกายน 2006 จนถึงปัจจุบันเรื่อยไป โดยสมมติให้วันที่ทำการออกคำสั่งแก้ไขเป็นวันที่ 5 ตุลาคม 2006 สามารถเขียนเป็นเอสคิวแอลในโมเดลเอ็มเอสบีดีได้ดังนี้

```
VALIDTIME [ DATE '2006-11-01',  DATE '9999-12-31')
UPDATE DETECTIVES
SET DEPARTMENT = 'ANTI-TERRORIST'
WHERE CODENAME = 'GIN'
```

คำสั่งนี้ต้องสั่งมาจากระดับเดียวกันกับข้อมูลของนักสืบ GIN นั่นคือในระดับ S1 ผลหลังการออกคำสั่งในมุมมองของผู้ใช้ โดยใช้คำสั่ง

```
VALIDTIME SELECT *
FROM DETECTIVES
WHERE CODENAME = 'GIN'
```

มีผลลัพท์ดังในตารางที่ 5.3 เป็นมุมของผู้ใช้ที่มีต่อแอพพลิเคชันนี้ในชุดคำสั่งไลวรีได้มีการกำหนด VALIDTIME เพื่อให้แสดงคาบเวลาที่ข้อมูลเป็นจริงอยู่ในปัจจุบันออกมา เนื่องจากไม่ได้กำหนดเวลาต้องการแสดงเอาไว้ เวลาปัจจุบันจริงเป็นค่าปริยาย เนื่องจากไม่ได้กำหนดเวลาทรานเซคชันเช่นกันคือยังเป็นข้อมูลที่ได้รับการยอมรับอยู่ด้วยสำหรับจีนมีอยู่ด้วยกันสองแถวที่มี t[TT_E] เป็นวันที่ 9999-12-31 เป็นข้อมูลที่ยังได้รับการยอมรับอยู่จนถึงปัจจุบัน

ตารางที่ 5.3 ผลลัพท์จากใช้คำสั่ง "SELECT"

CODENAME	*FULLNAME*	*DEPARTMENT*
GIN	JANE [2005-02-01,9999-12-31)	WATCH SECTION [2005-02-01,2006-11-01) ANTI-TERRORIST [2006-11-01,9999-12-31)

ความสามารถในการส่งผ่านคำสั่งไปยังลูกน้องใต้บังคับบัญชา ทำให้มีการติดต่อกันระหว่างระดับได้ แต่การทำเช่นนั้นทำได้เฉพาะออกคำสั่งในระดับที่ต่ำกว่าเท่านั้น เรามีกลไกในการกำหนดให้เป็นส่วนที่แบ่งแยกความต้องการการใช้งานของแต่ละเพื่อบ่งชี้แบ่งแยกข้อมูล ในที่นี้เราแบ่งข้อมูลในฐานข้อมูลออกเป็นสองลักษณะได้แก่ รายงาน ในที่นี้คือข้อมูลข้อเท็จจริงที่ผู้ใช้ได้เพิ่มหรือแก้ไขในฐานข้อมูล และอีกส่วนหนึ่งคือข้อมูลที่เป็นคำสั่ง เหล่านี้จะสอดคล้องกับความต้องการกับระดับผู้บังคับบัญชาผู้มีระดับสูงกว่าสามารถกำหนดงานให้ระดับต่ำกว่าทำงานได้ และข้อมูลในส่วนรายงานไม่ให้มีความก้าวก่ายกันได้ เพื่อความปลอดภัยของข้อมูลในระดับนั้นๆ เองอย่างไรก็ดีไม่สามารถมีการแก้ไขข้อมูลชนิดเติมแต่งได้โดยตรงโดยไม่รู้ที่มาที่ไป เนื่องจากการแก้ไขข้อมูลทุกครั้งจะมีเวลาระบุไว้ด้วยว่าได้รับการบันทึกตั้งแต่เมื่อไหร่ และข้อมูลก่อนหน้านั้นก็ยังอยู่เช่นกัน

กลับมาที่ตัวอย่างสำนักงานนักสืบ นักสืบแต่ละคนสามารถออกคำสั่งให้แก่ผู้ใต้บังคัญบัญชาได้ ในที่นี้คือผู้ที่อยู่ระดับต่ำกว่าทั้งหมด เช่นนักสืบ GIN ในระดับ S1 ต้องการข้อมูลจากติดตามบุคคลหนึ่ง โดยได้ออกคำสั่งให้กับนักสืบ KIR และนักสืบ TEQUILA ซึ่งอยู่ในระดับ C1 และ U1 ตามลำดับ และทั้งสองอยู่คนละหน่วยงานกัน โดยมีเป้าหมายให้เฝ้าติดตาม JACK ROBIN โดยออกคำสั่งเมื่อวันที่ 5 ตุลาคม 2006 และนักสืบ GIN ยังได้มีการกำหนดเวลาของการปฏิบัติงานด้วย คืองานนี้ให้เริ่มเมื่อ 10 กันยายน 2006 เรื่อยไปโดยยังไม่มีกำหนดสิ้นสุด ดั้งนั้นนักสืบ GIN ต้องส่งคำสั่งให้แก่ทั้งนักสืบทั้งสองคนที่อยู่ต่างระดับกัน นั่นคือต้องทำการออกคำสั่งเหมือนแต่แตกต่างนักสืบและแตกต่างระดับเพื่อรับคำสั่ง เราจะพิจารณาวิธีการออกคำสั่งโดยงานที่สร้างโดยนักสืบ GIN มีหมายเลขงานเป็น 337 การออกทำสั่งด้วยเอสคิวแอลนั้นทำได้โดยใช้คำสั่ง "INSERT" และระบุระดับที่ต่ำกว่าและกำหนดชื่อนักสืบที่ KIR และ TEQUILA ดังนี้คือ

```
VALIDTIME [ DATE '2006-09-10', DATE '9999-12-31')
INSERT INTO JOB_DET( JOB_ID, CODENAME)
VALUES( 337, 'KIR')
LEVELS C1

VALIDTIME [ DATE '2006-09-10', DATE '9999-12-31')
INSERT INTO JOB_DET( JOB_ID, CODENAME)
VALUES( 337, 'TEQUILA')
LEVELS U1
```

ในชุดคำสั่งแรกที่สั่งโดยระบุให้นักสืบ KIR ซึ่งอยู่ในระดับ C1 ให้รับงาน ในคำสั่งชุดที่สองให้ TEQUILA ในระดับ U1 เป็นผู้รับงานผลของการเพิ่มสองคำสั่งนี้ดังตารางที่ 5.4 แถวที่หนึ่งและแถวที่สามนับจากบนตามลำดับ

จากการนักสืบสองคน KIR และ TEQUILA ติดตาม JACK ROBIN ผู้รับคำสั่งนี้มีสามารถในการมองเห็นคำสั่งนี้ได้รวมถึงผู้ออกคำสั่ง หรือสูงกว่ากว่าผู้ที่ออกคำสั่งเท่านั้น นั่นคือข้อมูลของหน่วยงานอื่นที่ไม่ได้มีระดับความปลอดภัยเดียวกันไม่สามารถเห็นได้แม้จะมีระดับความปลอดภัยที่เท่ากันก็ตาม หลังจากที่มีการออกคำสั่งออกไป เมื่อนักสืบ KIR ได้รับทราบว่ามีคำสั่งมาถึงตน และได้ทำการยอมรับการปฏิบัติงานนี้ แอพพลิเคชันทำการดึงคำสั่งนั้นมาเป็นข้อมูลที่ยอมรับในระดับ C1 ซึ่งเป็นระดับของนักสืบ KIR การที่ต้องนำข้อมูลมาเป็นข้อมูลในระดับของตนเองจากคำสั่งนี้เนื่องจากผู้ใช้ระดับ C1 ไม่สามารถแก้ไขข้อมูลอะไรที่ไม่เป็นเป็นผู้ที่สร้างขึ้นมาเองได้ การทำเช่นนี้ผู้ใช้อาจมีการแก้ไขข้อมูลจากคำสั่งที่สั่งมาได้เองคือไม่ได้ทำตามคำสั่ง หรือเลือกปฏิบัติตามคำสั่งอื่น และเพื่อไม่ให้มีผลกระทบต่อข้อมูลที่เป็นข้อเท็จจริงในผู้ใช้ระดับ C1 ผู้ใช้ระดับสูงกว่าไม่สามารถแก้ไขข้อมูลในระดับต่ำกว่าหรือเป็นรายงานได้ และเมื่อนักสืบ KIR ตกลงในการรับคำสั่งในวันที่ 6 กันยายน 2006 ทำการคัดลอกรายละเอียดของคำสั่งมาไว้ในระดับของตนแสดงในแถวที่สองของตารางที่ 5.4

ตารางที่ 5.4 ข้อมูลในตาราง JOB_DET

JOB_ID	*JI_C*	*CODENAME*	*CN_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
337	C1	KIR	S1	2006-09-10	9999-12-31	2006-09-05	9999-12-31	C1	S1
337	C1	KIR	S2	2006-09-10	9999-12-31	2006-09-06	9999-12-31	C1	C1
337	U1	TEQUILA	U1	2006-09-10	9999-12-31	2006-09-05	9999-12-31	U1	S1
337	U1	TEQUILA	U1	2006-09-10	9999-12-31	2006-09-10	9999-12-31	U1	U1

ตารางที่ 5.5 ข้อมูลในตาราง JOB_TAR

JOB_ID	*JI_C*	*TNAME*	*TFN_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*	*CC*
337	C1	JACK ROBIN	C1	2006-09-10	9999-12-31	2006-09-05	9999-12-31	C1	S1
337	C1	JACK ROBIN	C1	2006-09-10	9999-12-31	2006-09-06	9999-12-31	C1	C1
337	U1	JACK ROBIN	U1	2006-09-10	9999-12-31	2006-09-05	9999-12-31	U1	S1
337	U1	JACK ROBIN	U1	2006-09-10	9999-12-31	2006-09-10	9999-12-31	U1	U1

ตารางที่ 5.6 ข้อมูลในตาราง TRACKING_REPORTS

TNAME	*TFN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2006-09-10	2006-09-13	2006-09-15	9999-12-31	S1
JACK ROBIN	C1,U1	RELAX	C1	ALPINE COURSE	U1	2006-09-13	9999-12-31	2006-09-15	9999-12-31	S1
JACK ROBIN	C1	RELAX	C1	AMARI HOTEL	C1	2006-09-10	9999-12-31	2006-09-11	9999-12-31	C1
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2006-09-10	9999-12-31	2006-09-11	9999-12-31	U1

เมื่อนักสืบ KIR และ TEQUILA ทำการติดตาม JACK ROBIN และได้รับทราบข้อมูลก็ทำการรายงานข้อมูลจากการติดตามลงในในตาราง "TRACKING_REPORTS" ในที่นี้ไม่จำเป็นต้องมีคอลัมน์ CC ก็ได้หากต้องการให้มีเฉพาะรายงานเท่านั้น นักสืบ KIR สามารถที่ให้ความเชื่อถือข้อมูลจากนักสืบ TEQUILA หรือไม่ก็ได้ขึ้นกับนักสืบ KIR เองผู้ซึ่งอยู่ในระดับสูงกว่าสามารถพิจารณาความน่าเชื่อถือได้เอง นักสืบทั้งสองเชื่อว่า JACK ROBIN อยู่กันคนละสถานที่ ในวันที่ 10 ตุลาคม 2006 นั่นคือนักสืบ KIR เชื่อว่า JACK ROBIN อยู่ที่ AMARI HOTEL และกำลังพักผ่อนอยู่ในขณะที่นักสืบ TEQUILA เชื่อว่าอยู่ที่ ALPINE COURSE เพื่อเล่นกอล์ฟและรายงานลงในฐานข้อมูลดังแสดงในตารางที่ 5.6 สองแถวล่าง

5.5 คำสั่ง UPLEVEL

การใช้คำสั่งมีรูปแบบดังต่อไปนี้

[VALIDTIME '[vt_s, vt_e)']
UPLEVEL *R*
GET A_i FROM c_i[, A_j FROM c_j] ...
[WHERE *p*]

คำสั่งนี้สามารถดึงข้อมูลที่ผู้ใช้ต้องการยืนยันโดยรับรู้ด้วยตนเองว่า ข้อมูลนั้นเป็นความจริงและต้องการมาเป็นข้อมูลในระดับของตน การดึงข้อมูลขึ้นมาต้องเป็นไปตามกฎข้อบังคับของความเชื่อที่มีต่อข้อมูลระดับล่าง เป็นข้อจำกัดของข้อมูลที่แพร่ขึ้นมาสู่ระดับที่สูงกว่าเพื่อป้องกันความกำกวมของข้อมูล จากรูปแบบคำสั่ง UPLEVEL ทางด้านบน A_i, A_j,... เป็นแอททริบิวข้อมูลของตาราง R และมี c_i, c_j... เป็นระดับความปลอดภัยสำหรับในอนุประโยค GET โดยไม่มีคีย์หลักแอพพาเรนท์เป็นส่วนประกอบ ในทุกคีย์หลักแอพพาเรนท์ของแถวที่ดึงขึ้นมาตรงกันกับแถวที่ถูกดึงรวมถึงต้องเพิ่มระดับความปลอดภัยของระดับที่ดึงมาในเซทระดับความปอลดภัยของคีย์หลักแอพพาเรนท์ที่ดึงมาด้วย ในอนุประโยค WHERE แถวที่นำมาใช้ได้ต้องเป็นแถวที่ผู้ใช้สามารถมองเห็นได้เท่านั้น คือระดับเดียวกันหรือเป็นแถวที่มีระดับต่ำกว่า สามารถกำหนดช่วงเวลาของข้อมูลได้ในอนุประโยค VALIDTIME มีคาบเวลาเป็น [*vt_s*,*vt_e*)

จากข้อมูลของการติดตาม JACK ROBIN ที่ได้มีการรายงานข้อมูลจากนักสืบ KIR และ TEQUILA แล้วนั้น นักสืบ GIN ในระดับ S1 ได้ทำการตรวจสอบข้อมูลจากนักสืบทั้งสองคน แล้วตัดสินใจเชื่อในข้อมูลของ TEQUILA ทั้งข้อมูลการกระทำและสถานที่ตั้งแต่วันที่ 10 ถึง 13 กันยายน 2006 และได้เชื่อถือข้อมูลจาก KIR ว่า JACK ROBIN ได้พักผ่อนอยู่ ในวันที่ 13 เป็นต้นไป

และยังคงเชื่อข้อมูลเดิมของสถานที่จาก TEQUILA ดังนั้นตั้งแต่วันที่ 13 กันยายน 2006 นักสืบ GIN ได้เชื่อข้อมูลจากทั้ง KIR และ TEQUILA ในบางส่วน สามารถออกคำสั่งเป็นเอสคิวแอลในโมเดลเอ็มเอสบีดีได้ดังนี้

```
VALIDTIME [ DATE '2006-09-10', DATE '2006-09-13')
UPLEVEL TRACKING_REPORTS
GET ACTION FROM U1, LOCAT FROM U1
WHERE TNAME = 'JACK ROBIN'

VALIDTIME [ DATE '2006-09-13', DATE '9999-12-31')
UPLEVEL TRACKING_REPORTS
GET ACTION FROM C1, LOCAT FROM U1
WHERE TNAME = 'JACK ROBIN'
```

คำสั่ง UPLEVEL แรกเป็นการดึงข้อมูลของนักสืบ TEQUILA ตั้งแต่วันที่ 10 ถึง 13 กันยายน 2006 และในคำสั่ง UPLEVEL ถัดมาเป็นการดึงข้อมูลจากสองระดับคือ C1 และ U1 ตั้งแต่วันที่ 13 กันยายน 2006 เป็นต้นไป ผลลัพธ์ของสองคำสั่งนี้แสดงในตารางที่ 5.6 สองแถวบนตามลำดับ

5.6 คำสั่ง DELETE

การใช้คำสั่งมีรูปแบบดังต่อไปนี้

[VALIDTIME '[vt_s, vt_e)']
DELETE
FROM R
[WHERE p]
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

ถ้านักสืบ TEQUILA ต้องการลบข้อมูลติดตาม JACK ROBIN ที่ได้รายงานไว้ในตารางที่ 5.6 นอกเหนือจากจะมีผลในระดับของตนเองคือ U1 แล้วยังส่งผลต่อระดับบน S1 ซึ่งได้ดึงข้อมูลจากระดับ U1 ไปใช้ด้วย นักสืบ TEQUILA ทำการลบข้อมูลของการติดตาม JACK ROBIN ตั้งแต่วันที่ 13 กันยายน 2006 เป็นต้นไปสามารถออกคำสั่งเป็นเอสคิวแอลในโมเดลเอ็มเอสบีดีได้ดังนี้

```
VALIDTIME [ DATE '2006-09-13', DATE '9999-12-31')
DELETE FROM TRACKING_REPORTS
WHERE TNAME = 'JACK ROBIN'
```

การลบครั้งนี้ส่งผลต่อระดับบนที่ทำการยืมข้อมูลนี้ไปคือ S1 ประกอบด้วยสองแถวบนใน ตารางที่ 5.6 เนื่องจากคาบเวลาของทั้งสองแถวมีส่วนซ้อนทับกันกับส่วนที่ถูกลบผลเป็นดังตารางที่ 5.7

ตารางที่ 5.7 ข้อมูลในตาราง TRACKING_REPORTS หลังถูกลบข้อมูล

TNAME	*TFN_C*	*ACTION*	*AT_C*	*LOCALE*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2006-09-10	2006-09-13	2006-09-15	2006-09-16	S1
JACK ROBIN	C1,U1	RELAX	C1	ALPINE COURSE	U1	2006-09-13	9999-12-31	2006-09-15	2006-09-16	S1
JACK ROBIN	C1	RELAX	C1	NULL	U1	2006-09-13	9999-12-31	2006-09-16	9999-12-31	S1
JACK ROBIN	C1	RELAX	C1	AMARI HOTEL	C1	2006-09-10	9999-12-31	2006-09-11	9999-12-31	C1
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2006-09-10	9999-12-31	2006-09-11	2006-09-16	U1

ผลจากการลบข้อมูลของ JACK ROBIN จากนักสืบ TEQUILA ในระดับ U1 ส่งผลกระทบต่อข้อมูลในระดับ S1 จากตารางที่ 5.7 แถวแรกถูกลบแทนที่ทำการลบแถวจริงๆ ก็ทำการหยุดความเชื่อที่วันที่ 16 เนื่องจากแถวมีความเชื่อเดิมทั้งหมดจาก U1 คือดูได้จาก t[TFN_C] ส่วนในช่วงหลังวันที่ 13 มีข้อมูลจากสองระดับคือ C1 และ U1 เมื่อ U1 ถูกลบไป ความเชื่อของแถวที่สองได้สิ้นสุดลงในวันที่ 16 ข้อมูลก็จะถูก แก้ไขโดยการเพิ่มแถวใหม่กำหนดข้อมูลส่วนของ U1 ให้เป็น NULL แต่แถวไม่ถูกลบไปเนื่องจาก U1 ไม่ได้เป็นส่วนประกอบทั้งหมดใน t[TFN_C] ดังในแถวที่สาม

บทที่ 6
การพัฒนาเอ็มเอสบีดี

การประยุกต์เพื่อให้ฐานข้อมูลเชิงสัมพันธ์แบบปกติ สามารถรองรับคุณสมบัติของโมเดลของเรา ซึ่งต้องประกอบไปด้วยสองส่วนหลักๆ คือทั้งต้องให้มีคุณสมบัติของเวลา รวมถึงมีคุณสมบัติแบ่งออกเป็นระดับชั้น และยังมีส่วนของข้อบังคับอีกด้วย จากบทที่ 5 ได้กล่าวถึงตัวอย่างของฐานข้อมูลเอ็มเอสบีดี และการนำไปใช้งานในบทนี้จะได้แสดงให้เห็นว่าการนำฐานข้อมูลเชิงสัมพันธ์ที่มีใช้อยู่กันนั้นสามารถนำไปประยุกต์ใช้ให้สามารถมีคุณสมบัติของเอ็มเอสบีดีได้

ในบทนี้กล่าวถึงการใช้เอสคิวแอลมาตรฐานมาปรับใช้กับฐานข้อมูลที่มีโครงสร้างแบบโมเดลเอ็มเอสบี เพื่อให้รองรับคำสั่ง SELECT, INSERT, UPDATE, DELETE และ UPLEVEL ได้ ในการนำไปใช้นั้นเราสามารถสร้างคำสั่งทดแทนการใช้งานในชุดคำสั่งของโปรซีเยอร์ได้ด้วย วิธีนี้สามารถนำไปใช้งานด้วยอาศัยการใช้งานร่วมกับทริกเกอร์ ใช้ในการตรวจสอบความถูกต้องและสอดคล้องของข้อมูลเมื่อมีการใช้คำสั่งในการปรับปรุงข้อมูลทุกครั้ง

ในความเป็นจริงการใช้งานเอ็มเอสบีดีสามารถแบ่งออกได้หลายรูปแบบ แต่เราได้เสนอการนำไปใช้สำหรับการแก้ไขข้อมูลแบบอ้างถึงลำดับเหตุการณ์ คือครอบคลุมการปรับปรุงข้อมูลได้ตลอดช่วงเวลาทั้งหมด

6.1 การพัฒนาคำสั่ง SELECT

เพื่อให้สอดคล้องกับกฎเกณฑ์พื้นฐานการอ่านของข้อมูลในระดับความปลอดภัยหลายระดับนอกเหนือจากการกำหนดให้แต่ละผู้ใช้มีระดับของตนเองแล้ว วิธีการกำหนดขอบเขตความสามารถของการอ่านของแต่ละผู้ใช้ สามารถทำได้ด้วยการสร้างฟังก์ชันขึ้นมาในที่นี้ให้ชื่อว่า TO_NUM(*label*) โดยกำหนดระดับความปลอดภัย แล้วคืนค่าระดับความปลอดภัยเป็นตัวเลขโดยอ้างอิงจากตารางที่ 6.1 ซึ่งเก็บระดับความปลอดภัยในฐานข้อมูลเอ็มเอสบีดี

ตารางที่ 6.1 ข้อมูลบางส่วนจากตาราง Security_Labels

LABEL	*CLASS_NAME*	*NUMLABEL*
TS	Top Secret Level	9
S1	Secret Level 1	6
S2	Secret Level 2	6
C1	Classified Level 1	3
C2	Classified Level 2	3
U1	Unclassified Level 1	0

การกำหนดตัวเลขให้กับระดับความปลอดภัยแต่ละระดับโดยในระดับเดียวกันแต่ต่างชื่อระดับ เช่น ระดับ S1 และระดับ S2 มีระดับความปลอดภัยเท่ากันกำหนดให้มีค่าเป็นตัวเลขคือ 6 เช่นเดียวกัน ระดับที่สูงกว่ามีค่าตัวเลขสูงกว่า การเปลี่ยนระดับความปลอดภัยให้เป็นตัวเลขด้วยฟังก์ชัน TO_NUM(*label*) นี้ทำให้เราสามารถเปรียบเทียบระดับความปลอดภัยได้ด้วยเครื่องหมายเปรียบเทียบทางคณิตศาตร์ แต่ละผู้ใช้เมื่อต้องการดูข้อมูลมีขอบเขตให้เพียงมองผ่านวิวที่มีคำสั่งดังต่อไปนี้

SELECT *

FROM *R*

WHERE *TC* = *user_level* OR *CC* = *user_level* OR

(TO_NUM(*TC*) < TO_NUM(*user_level*) AND TO_NUM(CC) < TO_NUM(*user_level*))

จากเงื่อนไขในอนุประโยค WHERE กำหนดให้ *user_level* เป็นระดับความปลอดภัยของผู้ใช้ ผลจากคำสั่งนี้ทำให้มองเห็นได้เฉพาะที่เป็นข้อมูลในระดับของตนเอง หรือเป็นแถวที่ตนเองสร้างขึ้น หรือแถวอื่นที่มีระดับอื่นที่ต่ำว่า และต้องถูกสร้างโดยระดับตนเองหรือต่ำกว่าเท่านั้น

ในตารางเอ็มเอสบีดี ข้อมูลในแต่ละแอททริบิวมีคาบเวลาเป็นของตนเอง จากบทที่ห้าเราได้อธิบายว่าอาจมีการแยกข้อมูลที่ต่อเนื่องของเวลาออกเป็นหลายแถวได้ ไม่ว่าจากการเพิ่มข้อมูลที่ไม่ต่อเนื่องกันของบางแอททริบิวทำให้แอททริบิวที่มีข้อมูลต่อเนื่องกันตลอดเวลาถูกแยกแถวไปด้วย ในตารางที่ 6.2 เป็นข้อมูลจากตารางหนึ่งในฐานข้อมูลของสำนักงานนักสืบ ที่เฝ้าติดตามเป้าหมาย (TNAME) โดยให้คอยเฝ้ามองว่าทำอะไรอยู่ (ACTION) และสถานที่ใด (LOCAT) ข้อมูลทั้งหมดสามารถบ่งบอกเวลาได้ด้วยคุณสมบัติจากเอ็มเอสบีดี คือสามารถกำหนดเวลาได้ในระดับแอททริบิว เนื่องจากเป้าหมายสามารถทำอะไรได้มากว่าสองอย่างในสถานที่เดียวกันในข้อมูลตัวอย่างจากตารางที่ 6.2 เป้าหมาย "JACK ROBIN" ที่ถูกติดตามโดยนักสืบสามระดับ คือระดับ S1, C1 และ U1 นักสืบจากทั้งสามระดับมีข้อมูลที่แตกต่างกัน ในระดับ S1 ประกอบด้วยสองแถวข้อมูลที่ต่อเนื่องกันนั่นคือได้รายงานการติดตามของ "JACK ROBIN" ตั้งแต่วันที่ 16 กุมภาพันธ์ 2007 จนถึงปัจจุบันดูได้จากเวลาวาลิด และได้รับการบันทึกข้อมูลนี้เมื่อวันนี้ 17 กุมภาพันธ์ 2007 และข้อมูลนี้ยังคงได้รับการบันทึกโดยยังไม่มีการเปลี่ยนแปลงจนถึงปัจจุบันดูได้จากเวลาทรานเซคชัน

ตารางที่ 6.2 ข้อมูลบางส่วนจากตาราง TRACKING_REPORTS

TNAME	*TFN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2007-02-16	2007-02-27	2007-02-17	9999-12-31	S1
JACK ROBIN	C1,U1	RELAX	C1	ALPINE COURSE	U1	2007-02-27	9999-12-31	2007-02-17	9999-12-31	S1
JACK ROBIN	C1	RELAX	C1	AMARI HOTEL	C1	2007-02-16	9999-12-31	2007-02-16	9999-12-31	C1
JACK ROBIN	U1	PLAY GOLF	U1	ALPINE COURSE	U1	2007-02-16	9999-12-31	2007-02-16	9999-12-31	U1

ตลอดช่วงเวลาของข้อมูลในระดับ S1 นี้มีข้อมูลที่ต่อเนื่องกันของ t[LOCAT] แต่ใน t[ACTION] เป็นข้อมูลที่ไม่ต่อเนื่องเนื่องจากได้มีการเปลี่ยนจากการเล่นกอล์ฟมาเป็นพักผ่อนในแถวที่หนึ่งและแถวที่สองนับจากบนตามลำดับ ดังนั้นการนำไปใช้ในแอพพลิเคชันควรจะมีการรวมข้อมูลที่ต่อเนื่องกันเป็นข้อมูลเดียวโดยกำกับด้วยเวลาที่ต่อเนื่องยาวที่สุด เช่นใน t[LOCAT] ควรแสดงให้เห็นว่าอยู่ที่ "ALPINE COURSE" ตั้งแต่วันที่ 16 ถึง 17 กุมภาพันธ์ 2007 การหาคาบเวลายาวสุดแบ่งเป็นสี่กรณีด้วยกันกรณีแรกเป็นการหาข้อมูลที่เป็นข้อมูลเดียวกันและต่อเนื่องกันพอดีในส่วนปลายคาบเวลาและเริ่มต้นอีกคาบเวลาดังรูปที่ 6.1 เวลาที่ต้องการหาก็คือ t[R1.VT_S] คือเวลาเริ่มต้นของข้อมูลต่อเนื่องที่ยาวที่สุด ด้วยการกำหนดเงื่อนไขต้องไม่ให้มีคาบเวลา R3 ที่เป็นข้อมูลเดียวกันและต่อเนื่องกันกับ t[R1.VT_S]

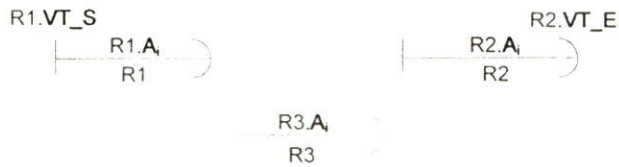
รูปที่ 6.1 การตรวจสอบคาบเวลาเริ่มต้นที่ยาวที่สุดของข้อมูลที่ต่อเนื่องกัน

เช่นเดียวกันต้องมีการหาเวลาสิ้นสุดที่เป็นข้อมูลเดียวกันและต่อเนื่องกันมากที่สุดในที่นี้คือ t[R2.VT_E] ดังนั้นรูปที่ 6.2 ต้องไม่มี R3 ที่เป็นข้อมูลเดียวกันและต่อเนื่องกันกับ t[R2.VT_E] ผลลัพธ์สุดท้ายที่เราต้องการก็คือคาบเวลา [t[R1.VTS], t[R2.VT_E]) ที่ระบุอยู่ในรูปที่ 6.1 ถึง 6.4

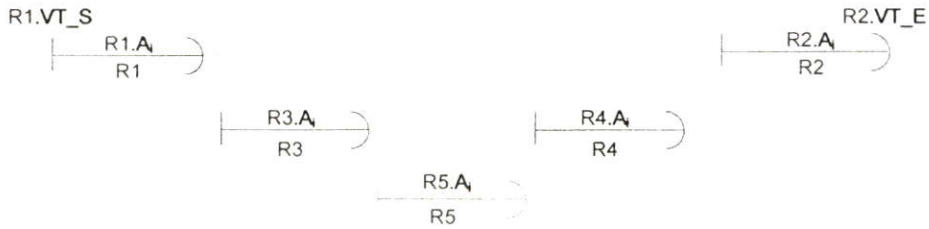
รูปที่ 6.2 การตรวจสอบคาบเวลาสิ้นสุดที่ยาวที่สุดของข้อมูลที่ต่อเนื่องกัน

สิ่งที่ทำให้ข้อมูลไม่เป็นข้อมูลเดียวกันและต่อเนื่องกันคือ คาบเวลาไม่ต่อเนื่องบรรจบกันพอดีหรือข้อมูลไม่ตรงกันดังนั้นจึงต้องมีการตรวจสอบไม่ให้มีคาบเวลา R3 ซึ่งมี $t[R3.A_i] \neq t[R1.A_i]$ ดังในรูปที่ 6.3 และในการตรวจสอบสุดท้ายในรูปที่ 6.4 เพื่อตรวจสอบความต่อเนื่องว่าไม่มีการขาด

ระยะกันของคาบเวลาต้องมีคาบเวลา R5 อยู่เสมอระหว่าง R3 และ R4 ที่อยู่ภายใต้คาบเวลา [t[R1.VTS], t[R2.VT_E])



รูปที่ 6.3 การตรวจสอบความต่อเนื่องเป็นข้อมูลเดียวกันยาวที่สุด



รูปที่ 6.4 การตราวสอบให้เป็นข้อมูลเดียวกันและต่อเนื่องกันพอดี

6.2 การพัฒนาคำสั่ง INSERT

จากตารางที่กล่าวในบทที่แล้วถึงการติดตามข้อมูลบุคคลจากสำนักงานนักสืบ เมื่อต้องการรายงานผลการติดตามสามารถใช้คำสั่ง INSERT ในการเพิ่มข้อมูล กำหนดให้ A_1 เป็นคีย์หลักแอพพาเรนท์และ $1 < i \leq n$ เนื่องจากทุกแอททริบิวต้องการการอิงเวลาที่สอดคล้องกับโลกความเป็นจริงที่เกาะอยู่กับ A_1 เพียงอย่างเดียวดังนั้น หากมี A_i ที่มีคาบเวลาต่างออกไปจากคาบเวลาของ A_1 แต่ต้องอยู่ในขอบเขตของ $[VT_s, VT_e)$ ซึ่งเป็นตัวกำหนดคาบเวลาวาลิดของแถว ตามข้อบังคับของกฎบังคับการเกิดข้อมูลซ้ำซ้อน แม้ว่าในแต่ละเวลาที่ติดกัน (Adjacent) และมี $t[A_i]$ เหมือนกันก็ทำการแยกออกจากกันก็ต่อเมื่อข้อมูลใน $t[A_i]$ แตกต่างกัน ตัวอย่างจากตาราง DETECTIVES เราต้องการเพิ่มนักสืบใหม่เข้าสู่ระบบ โดยต้องการเพิ่มนักสืบ SHERRY เข้าสู่ระบบวันที่ 5 ตุลาคม 2006 โดยมีชื่อจริงว่า TINA STONE แต่ต้องการให้เธอทำงานในหน่วยงาน BOMB SQUAD ตั้งแต่วันที่ 1 พฤศจิกายน 2006 เป็นต้นไปซึ่งเป็นเวลาหลังจากได้เพิ่มข้อมูลของเธอไปแล้วเกือบหนึ่งเดือน ในโมเดลเอ็มเอสบีดีสามารถทำการเพิ่มข้อมูลในลักษณะนี้ได้ เนื่องจากสามารถกำหนดคาบเวลาที่ต้องการให้กับแต่ละแอททริบิวได้ จากความต้องการเพิ่มข้อมูลนักสืบ SHERRY สามารถเขียนเป็นเอสคิวแอลสำหรับโมเดลเอ็มเอสบีดีได้ดังนี้

```
VALIDTIME( CODENAME, DEPARTMENT)
PERIODS([DATE '2006-10-5', DATE '9999-12-31') ,[DATE '2006-11-1', DATE '9999-12-31'))
INSERT INTO DETECTIVES( CODENAME, FULLNAME, DEPARTMENT)
VALUES( 'SHERRY', 'TINA STONE', 'BOMB SQUAD')
LEVELS U2
```

จากคำสั่งเราไม่ได้ให้ความครอบคลุมการเปลี่ยนแปลงของแอททริบิว FULLNAME ขณะ INSERT จะถูกกำหนดโดยอัตโนมัติว่ามีคาบเวลาเท่ากับ CODENAME แต่ในแอททริบิวแผนกเมื่อนำมาใช้งานจริงเราก็ควรใช้ผ่านโปรซีเจอร์แทน ผลเมื่อใช้คำสั่ง INSERT กับตาราง DETECTIVES เป็นเฉพาะส่วนที่เพิ่มไปใหม่ดังตารางที่ 6.3

ตารางที่ 6.3 ข้อมูลที่เพิ่มขึ้นจากคำสั่ง INSERT กับตาราง DETECTIVES

CODE NAME	CN _C	FULLNAME	FN _C	DEPARTMENT	D _C	VT_S	VT_E	TT_S	TT_E	TC	CC
SHERRY	U2	TINA STONE	U2	NULL	U2	2006-10-05	2006-11-01	2006-10-01	9999-12-31	U2	U2
SHERRY	U2	TINA STONE	U2	BOMB SQUAD	U2	2006-11-01	9999-12-31	2006-10-01	9999-12-31	U2	U2

เราทำการ INSERT เพียงคำสั่งเดียวผลลัพธ์มีด้วยกันสองแถว เนื่องจากความต่อเนื่องของเวลาในแต่ข้อมูล ไม่สอดคล้องกันเองภายในแถวจากรูปเห็นได้ว่า นักสืบ SHERRY มีคาบเวลาวาลิดเริ่มต้นของโคดเนม เริ่มตั้งแต่วันที่ 5 ตุลาคม 2006 แต่ยังไม่มีการระบุหน่วยงาน ต่อมาในวันที่ 1 พฤศจิกายน 2006 ให้บรรจุในหน่วยงาน BOMB SQUAD ทำให้ข้อมูลต้องแยกออกไปเป็นอีกแถวหนึ่ง และทั้งหมดบันทึกข้อมูลในวันที่ 1 ตุลาคม 2006 เป็นการวางแผนล่วงหน้าให้มีโคดเนมก่อนแล้วบรรจุลงในหน่วยงานภายหลัง

คำสั่งของการ INSERT สามารถเขียนได้ในรูปแบบของโปรซีเจอร์ที่มีอินพุทเป็นข้อมูลที่ประกอบด้วย ข้อมูลในลักษณะฐานข้อมูลปกติ คาบเวลาของข้อเท็จจริงในแต่ละข้อมูล และความต้องการให้ข้อมูลนั้นอยู่ในระดับใด เป็นการเจาะจงว่าเป็นคำสั่งหรือรายงาน จากคำสั่ง INSERT ข้างต้นภายในโปรซีเจอร์ของเรามีการทำงานด้วยคำสั่งดังต่อไปนี้

```
INSERT INTO DETECTIVES
VALUES( 'SHERRY', 'TINA STONE', 'U2', NULL, 'U2', DATE '2006-10-05', DATE '9999-12-31',
  CURRENT_DATE, DATE '9999-12-31', 'U2', 'U2', 'U2')

INSERT INTO DETECTIVES
SELECT CODENAME, CODENAME_C, FULLNAME, FULLNAME_C,
  DEPARTMENT, DEPARTMENT_C VT_S, DATE '200-11-01', TT_S, TT_E, TC, CC
FROM DETECTIVES
WHERE CODENAME = 'SHERRY' AND TC = 'U2' AND CC = 'U2'
  AND VT_S < DATE '200-11-01' AND VT_E > DATE '200-11-01'
  AND TT_E = DATE '9999-12-31'

INSERT INTO DETECTIVES( CODENAME, CODENAME_C, FULLNAME, FULLNAME_C,
  DEPARTMENT, DEPARTMENT_C, VT_S, VT_E, TT_S, TT_E TC, CC)
SELECT CODENAME, CODENAME_C, FULLNAME, FULLNAME_C,
```

```
  DEPARTMENT, DEPARTMENT_C, DATE '9999-12-31', VT_E, TT_S, TT_E, TC, CC
FROM DETECTIVES
WHERE CODENAME = 'SHERRY' AND TC = 'U2' AND CC = 'U2'
  AND VT_S < DATE '9999-12-31'    AND VT_E > DATE '9999-12-31'
  AND TT_E = DATE '9999-12-31'

UPDATE DETECTIVES
SET DEPARTMENT = 'BOMB SQUAD'
WHERE CODENAME = 'SHERRY' AND TC = 'U2' AND CC = 'U2'
  AND VT_S < DATE '9999-12-31'
  AND VT_E > DATE '200-11-01'
  AND TT_E = DATE '9999-12-31'

UPDATE DETECTIVES
SET VT_S = DATE '200-11-01'
WHERE CODENAME = 'SHERRY' AND TC = 'U2' AND CC = 'U2'
  AND VT_S < DATE '200-11-01'
  AND VT_E > DATE '200-11-01'
  AND TT_E = DATE '9999-12-31'

UPDATE DETECTIVES
SET VT_E = DATE '9999-12-31'
WHERE CODENAME = 'SHERRY' AND TC = 'U2' AND CC = 'U2'
  AND VT_S < DATE '9999-12-31'
  AND VT_E > DATE '9999-12-31'
  AND TT_E = DATE '9999-12-31'
```

จากตัวอย่างเป็นการเพิ่มข้อมูลของนักสืบ SHERRY เข้าไปในตาราง DETECTIVES ข้อมูลอย่างย่อที่เราได้นำเสนอนั้น ประกอบด้วย CODENAME, FULLNAME และ DEPARTMENT การเพิ่มข้อมูลในมิติของเวลานั้น เราสามารถกำหนดให้ได้ในคำสั่ง INSERT นี้ อย่างไรก็ตามสำหรับชื่อจริงในตัวอย่างไม่มีความต้องการใน การแปรเปลี่ยนตามเวลา มีเฉพาะโดดเนมที่จำเป็นต้องมีเนื่องจาก เป็นคีย์หลักแอพพาเรนท์ และหน่วยงานเท่านั้น

ในคำสั่ง INSERT แรกเป็นการกำหนดคาบเวลาให้กับคีย์หลักแอพพาเรนท์ซึ่งก็คือ SHERRY เนื่องจากคาบเวลานี้ครอบคลุมคาบเวลาของแอททริบิวอื่นๆ ทั้งหมด และในการเพิ่มขึ้นของข้อมูลต้องไม่เกิดการซ้ำซ้อนกันของเหตุการณ์เชิงเวลาอีกด้วยอีกด้วย กำหนดให้มีการสร้างคาบเวลาของคีย์หลักแอพพาเรนท์ขึ้นมา ลักษณะเช่นนี้เหมือนกับการกับหนดให้เป็นคาบเวลาในระดับของแถวทุกประการ เพียงแต่ไม่จำเป็นต้องกำหนดค่าของแอททริวบิวอื่นๆ ให้ละเป็น NULL ดังตารางที่ 6.3

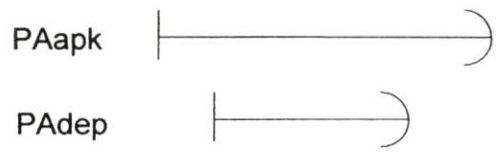
รูปที่ 6.5 คาบเวลาที่เพิ่มขึ้นสำหรับคำสั่ง INSERT

ประโยชน์อีกอย่างหนึ่งคือ เราสามารถรู้ได้ทันทีว่าการเพิ่มข้อมูลนี้ถูกต้องตามกฎข้อบังคับในส่วนของการป้องกันการละเมิดต่างๆ โดยเฉพาะอย่างยิ่งสามารถตรวจการเกิดข้อมูลซ้ำซ้อนในมิติของเวลาได้ทันที หากคาบเวลาที่ต้องการไปซ้อนทับกับคาบเวลาเดิมอื่นที่มีอยู่แล้วก็สามารถยกเลิกทรานเซกชันการเพิ่มข้อมูลนี้ได้ตั้งแต่ คำสั่งแรก ดังในรูปที่ 6.6 คาบเวลา PV (Period of Validity) เป็นคาบเวลาของข้อมูลเดิมที่มีอยู่แล้ว ซ้อนทับกับส่วนที่เราต้องการเพิ่มขึ้นมาใหม่ การมีข้อมูลเดิมอยู่แล้วการเพิ่มข้อมูลใหม่เข้าไปถือว่าไม่ยอมรับ

รูปที่ 6.6 คาบเวลาที่มีอยู่เดิมก่อนทำการเพิ่มข้อมูลใหม่

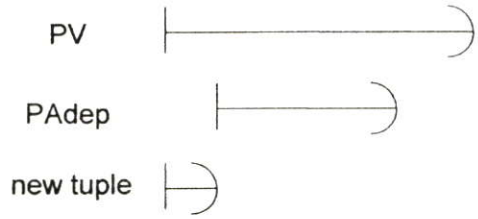
ในการเพิ่มข้อมูลเรายอมให้ทำการเพิ่มข้อมูลได้เฉพาะในระดับของตน หรือระดับต่ำกว่า นั่นคือสามารถสร้างแถวใหม่ที่มี t[CC] ซึ่งเป็นสิ่งระบุถึงระดับที่สร้างแถวเป็นระดับของตน ถ้าหาก PV และ PAapk (Period of Applicability) เป็นคาบเวลาที่ต้องการกระทำต่างระดับกันที่จำแนกโดย t[TC] แล้ว ก็ไม่มีความเกี่ยวในข้อบังคับของซ้ำซ้อนของข้อมูลเชิงเวลาแต่อย่างใด สำหรับการเพิ่มข้อมูลในระดับที่สูงกว่าเราไม่สามารถทำได้ เช่นเดียวกับความสามารถในการมองเห็นข้อมูล

คำสั่ง INSERT ที่สองและสามเป็นการตัดคาบเวลา หากมีคาบเวลาที่มีจุดเริ่มต้นหรือจุดสิ้นสุดไม่พอดีกับเวลาวาลิดของคีย์หลักแอพพาเรนท์



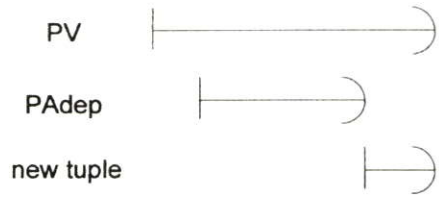
รูปที่ 6.7 คาบเวลาของหน่วยงานที่ถูกกำหนดเทียบกับคาบเวลาของคีย์หลักแอพพาเรนท์

PAdep รวมถึงคาบเวลาอื่นที่ไม่เป็นคาบเวลาของคีย์หลักแอพพาเรนท์จำเป็นต้องซ้อนทับ หรืออยู่ภายใต้ PAapk ทั้งหมด เพื่อสร้างความสอดคล้องภายในแถวตามกฎข้อบังคับเอนติตี เพื่อรักษาส่วนเดิมที่ไม่ซ้อนทับโดย PAdep ด้วยการเพิ่มแถวใหม่ดังนี้



รูปที่ 6.8 การเพิ่มคาบเวลาทางซ้ายจากคำสั่ง INSERT

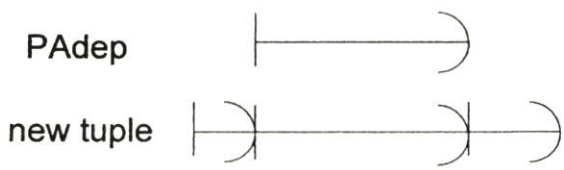
และใช้อีกคำสั่ง INSERT เพื่อทำส่วนทางขวา



รูปที่ 6.9 การเพิ่มคาบเวลาทางขวาจากคำสั่ง INSERT

เราจะได้สามแถวที่ยังซ้อนทับกันอยู่แต่มีเพียงแถวเดียวเท่านั้นที่ซ้อนทับกับ PAdep ซึ่งเป็นผลจากการ INSERT ทั้งสามครั้งที่ผ่านมา

จากนั้นใช้คำสั่ง UPDATE ปรับส่วนที่ซ้อนทับกับ PAdep ให้มีข้อมูลของหน่วยงานเป็นไปตามที่ต้องการ ในที่นี้คือคาบเวลาแถวกลาง เห็นได้ว่าความเป็นได้เมื่อทำการ INSERT เสร็จสิ้นจะมีคาบเวลาที่ติดกันโดยมีคีย์หลักแอพพาเรนท์ตัวเดียวกัน ได้สูงสุดสามคาบ ในกรณีที่ทั้งจุดเริ่มของ PAdep และ สิ้นสุดไม่ได้เท่ากันพอดีกับ จุดเริ่มต้น และจุดสิ้นสุดของ PAapk ตามลำดับ ผลลัพท์จากการ INSERT ครั้งนี้ควรเป็นดังนี้



รูปที่ 6.11 ผลลัพท์ของคาบเวลาหลังทำการเพิ่มข้อมูล

ผลลัพท์ดังสามคาบล่าง ในส่วนแรกและส่วนทางขวา เป็นส่วนที่เราได้ทำการ INSERT เพิ่มขึ้นมาใหม่ ส่วนตรงกลางเป็นผลมาจาก การแก้ไขคาบเวลาผลจาก PAapk ใช้สองคำสั่ง UPDATE ในการปรับทั้งหัว และท้าย และสุดท้ายคาบเวลาที่ตรงกับ PAdep หรือส่วนตรงกลางนั้นให้ทำการแก้ไขข้อมูลไปตามข้อมูลที่ต้องการเพิ่มนั้นๆ นี่เป็นหนึ่งขั้นตอนสำหรับกรณีให้มีคาบเวลาในแต่ละ

แอททริบิว รวมถึงคีย์หลักแอพพาเรนท์ด้วย ที่กล่าวมาเป็นการกระทำกับหนึ่งแอททริบิวที่ไม่เป็นคีย์หลักแอพพาเรนท์ หากมีแอททริบิวอื่นนอกเหนือจากนี้ให้ทำการทำซ้ำใหม่โดยยึดผลสุดท้ายทำต่อไป ซึ่งไม่จำเป็นต้องมีลำดับก่อนหลังในการใช้คำสั่ง INSERT ที่เกี่ยวพันธ์กับคาบเวลาของแต่ละแอททริวบิว เนื่องจากเปรียบเสมือนการหั่นเวลาออกเป็นชิ้นๆ ไม่ว่าจะหั่นตรงรอยไหนก่อนสุดท้ายก็ได้ผลลัพท์ออกมาเหมือนกัน เมื่อทำการ INSERT แล้ว t[CC] ต้องเป็นตัวระบุถึงผู้สร้างแถวและผู้ที่มองเห็น และต้องให้ความสนใจกับข้อมูลนี้คือที่ t[TC] สิ่งบ่งบอกว่าเป็นข้อมูลในระดับตน อาจเป็นส่วนที่ตนเองสร้างขึ้นหรือ ระดับล่างสร้างขึ้นมาที่เราให้นิยามว่าเป็นคำสั่งทุกคำสั่งที่ผ่านมานั้น เมื่อมีการสร้างแถวใหม่เรากำหนดให้มี จุดเริ่มเวลาของทรานเซคชันเป็นเวลาในปัจจุบัน และสิ้นสุดที่วันที่ 31 ธันวาคม 9999 หรือในที่นี้คือยังเชื่ออยู่เช่นนั้นเรื่อยไป

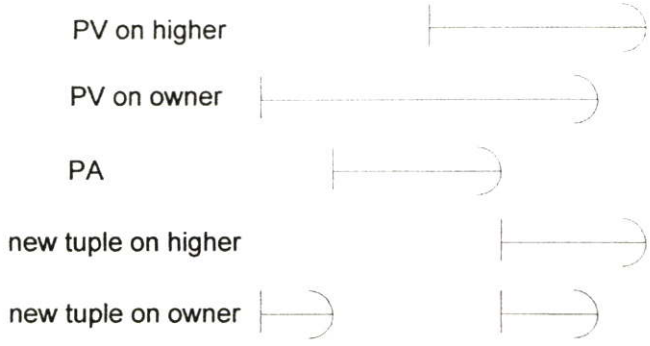
6.3 การพัฒนาคำสั่ง UPDATE

คำสั่งอัพเดทในเอ็มเอสบีดีไม่มีการทำการแก้ไขข้อมูลเดิมที่ต้องการกับแถวที่มีอยู่เดิมจริงๆ หากแต่ต้องทำการเพิ่มแถวเข้าไปใหม่แล้วระบุว่าแถวเดิมนั้นเป็นอดีตไปเสีย นั่นเป็นข้อดีที่ทำให้เราสามารถรับรู้ว่าช่วงเวลาหนึ่งเราได้เคยเชื่อข้อมูลนั้นว่าเป็นอย่างไร และยังมีข้อมูลของเวลาที่สามารถบอกความคงอยู่ของข้อมูลอีกด้วย นั่นคือเวลาวาลิดไม่เพียงแค่นั้นใน การ ปรับปรุงข้อมูลแม้ว่าจะกระทำได้ในระดับของตนหรือต่ำกว่า คือเราเป็นผู้สร้างโดยระบุใน t[CC] เท่านั้นแล้ว นั่นคือไม่มีสิทธิแก้ไขข้อมูลของระดับต่ำกว่าได้โดยเด็ดขาด เช่นเดียวกันเราไม่สามารถไปแก้ไขข้อมูลระดับสูงกว่าได้โดยตรง แต่ระดับบนเท่านั้นที่จะเป็นผู้ตัดสินว่ายินยอมให้ทำการแก้ไขหรือไม่ ซึ่งไม่ได้อยู่ในส่วนที่ผู้ใช้จะสามารถรับรู้ได้เลยว่าระดับสูงกว่าคิดอย่างไร ยังมีส่วนของการปรับปรุงข้อมูลที่ผู้ใช้ผู้ทำการไม่สามารถมองเห็นได้คือ ข้อมูลที่ส่งผลถึงระดับบน กล่าวคือหากระดับให้ความไว้วางใจกับข้อมูลในระดับล่างไว้แล้วนั้น หากข้อมูลในระดับล่างเปลี่ยนแปลง จะส่งผลกระทบถึงระดับบนด้วย

จากข้อมูลของนักสืบ SHERRY ในตารางที่ 6.3 เดิมทีนักสืบ SHERRY อยู่ในหน่วยงาน BOMB SQUAD ตั้งแต่วันที่ 1 พฤศจิกายน 2006 จนถึงปัจจุบัน ถ้าวันนี้เป็นที่ 1 ธันวาคม 2006 ต้องการปรับให้นักสืบ SHERRY ย้ายไปอยู่ในหน่วยงาน ANTI-TERRIST ตั้งแต่เดือนธันวาคม 2006 ตลอดจนถึงเดือนเมษายนในปีถัดไป ให้นักสืบ SHERRY ย้ายไปอยู่เป็นการชั่วคราว และยังคงข้อมูลเดิมให้กลับมาอยู่ที่ BOMB SQUAD ภายหลังจากนั้นสามารถออกคำสั่งเอสคิวแอลภายใต้โมเดลเอ็มเอสบีดีได้ดังนี้

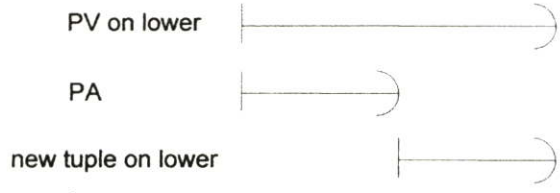
```
VALIDTIME PERIODS( [ DATE '2006-12-01', DATE '2007-05-01')
UPDATE DETECTIVES
SET DEPARTMENT = 'ANTI-TERRIST"
WHERE CODENAME = 'SHRRY'
```

ภายใต้คำสั่งอัพเดทของเอ็มเอสบีดีนั้นมีการกระทำต่อข้อมูลได้ทุกระดับชั้น ไม่ว่าระดับ ตนเองต่ำกว่าหรือสูงกว่า ถ้าหากมีระดับที่สูงกว่าทำการดึงข้อมูลของนักสืบ SHERRY ไปก็ต้องมีการเปลี่ยนแปลงข้อมูลตามไปด้วย ในรูปที่ 6.11 และ 6.12 เป็นการรักษาข้อมูลเดิมที่ไม่ได้ถูกกระทำโดย PA แล้วนั้น จึงไม่จำเป็นต้องไม่มีการเปลี่ยนแปลงใดๆ ความเป็นไปได้ของส่วนเดิมที่เหลืออยู่คือหัวและท้าย เราจำเป็นต้องเพิ่มแถวมาใหม่อีกสองแถวด้วยสองคำสั่ง INSERT โดยทำข้อมูลยังคงเดิมเช่นใน PV



รูปที่ 6.11 ข้อมูลภายใต้คาบเวลาที่ถูกปรับปรุงข้อมูล

สำหรับการปรับปรุงข้อมูลในระดับที่ต่ำกว่าผู้ใช้สามารถกำหนดได้เลยโดยตรง ข้อมูลนี้เป็นข้อมูลประเภทคำสั่งที่สร้างขึ้นโดยผู้ที่ออกคำสั่งเอง จึงไม่ส่งผลกระทบถึงระดับบนแต่อย่างไร

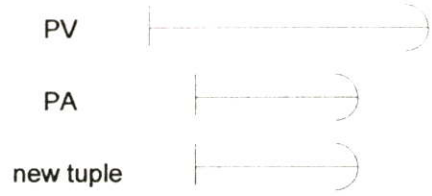


รูปที่ 6.12 บางส่วนของคาบเวลาที่ไม่ซ้อนทับกับ PA

ในสองการ INSERT เพื่อคงข้อมูลเดิมที่ไม่เกี่ยวข้องกับการแก้ไขนี้เราสามารถกำหนดเงื่อนไขได้ไม่เย็นนัก แต่หากจำเป็นต้องกระทำต่อระดับที่สูงกว่าด้วย เนื่องจากการปรับปรุงข้อมูลนี้ก็ส่งผลกระทบต่อระดับบนด้วยเช่นกัน เมื่อเราสามารถรักษาข้อมูลเดิมไว้ได้แล้ว เราจึงเริ่มทำการสร้างแถวใหม่เปรียบเหมือนข้อมูลที่ถูกทำการปรับปรุงข้อมูลแล้วลงไปด้วย ประกอบด้วยสามส่วน ส่วนแรกคำสั่ง INSERT ที่เป็นข้อมูลที่ได้อัพเดทแล้วที่สร้างโดยระดับผู้ที่สั่ง ดูได้จากแถวเดิมมี t[CC] เป็นของผู้ทำการปรับปรุงข้อมูล ส่วนที่สองทำการเพิ่มแถวใหม่ที่มีข้อมูลใหม่ที่อัพเดทในระดับสูงกว่าที่มีการยืมข้อมูลระดับนี้ไป ส่วนสุดท้ายทำการเพิ่มแถวใหม่ที่มีข้อมูลใหม่ ที่ในระดับต่ำกว่ามีการ

กำหนดคีย์หลักแอพพาเรนท์เปลี่ยนไป ดังนั้นข้อมูลในระดับสูงกว่าต้องถูกเปลี่ยนให้เป็น NULL เนื่องจากไม่มีข้อมูลในระดับล่างอยู่อีกแล้ว

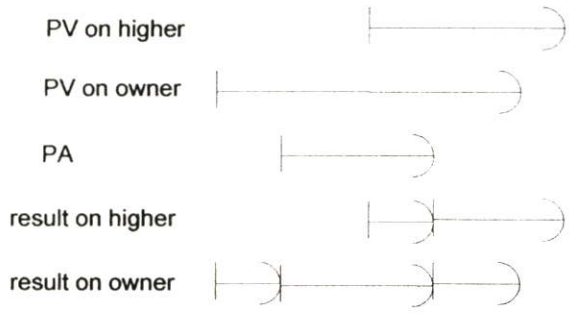
การปรับปรุงข้อมูลในระดับที่ระบุ คือระดับตนในส่วนรายงานทีมี t[TC] = t[CC] หรือคำสั่งในระดับล่างใดๆ ที่มี t[CC] เป็นระดับของผู้ใช้นั่นคือเป็นผู้ที่สร้างแถวนั้นขึ้นมา แล้วต้องเป็นส่วนที่ซ้อนทับกับข้อมูลเดิมที่มีอยู่ด้วยแสดงในรูปที่ 6.13



รูปที่ 6.13 ผลเพิ่มแถวของการปรับปรุงข้อมูลที่มีส่วนซ้อนทับกับ PA

ต่อมาทำการในระดับสูงกว่าในกรณีที่ไม่มีการเปลี่ยนแปลง APK ทำให้ระดับบนที่ให้ความไว้ใจในระดับล่างสามารถเปลี่ยนแปลงข้อมูลระดับบนได้ นั่นคือมีระดับความปลอดภัยเป็นระดับที่อยู่ระดับต่ำกว่าซึ่งต้องเป็นไปตามข้อบังคับของกฎข้อบังคับของความเชื่อต่อข้อมูลในระดับล่างต้องสอดคล้องกับข้อมูลในระดับล่าง หากระดับล่างทำการแก้ไขแล้วระดับบนต้องเปลี่ยนแปลงไปด้วยการปรับปรุงข้อมูลเฉพาะส่วนที่ซ้อนทับเท่านั้น โดยใช้ฟังก์ชัน "CASE" ช่วย นอกจากนั้นเรายังมีคำสั่งปรับปรุงข้อมูลอีกอย่างหนึ่งในกรณีที่มีการปรับปรุงข้อมูลของคีย์หลักแอพพาเรนท์ส่งผลให้เอนติตีในระดับนั้นเปลี่ยนแปลงไป ความเชื่อในระดับสูงกว่าต้องหมดไปทันที นอกจากนั้นระดับความปลอดภัยที่มีต่อความเชื่อเดิมต้องถูกลบออกไปด้วย นั่นคือคีย์หลักแอพพาเรนท์นี้ไม่มีการอ้างอิงกับระดับล่างอีกแล้ว ดังนั้นระดับความปลอดภัยสามารถบอกได้ว่าหากมีบางแอททริบิวอื่นที่เป็นนัลสามารถบอกได้ทันทีได้ว่าที่เป็นนัลเนื่องจากไม่มีเอนติตีนี้ในระดับล่างหรือว่าระดับล่างมีอยู่แต่มีแค่เป็นนัลด้วยเท่านั้น

สุดท้ายเป็นการออกคำสั่งให้แถวเดิมที่ก่อนการปรับปรุงข้อมูลหรือซ้อนทับกับ PA และยังเป็นข้อมูลเก่าอยู่ให้สิ้นสุดความเชื่อของเวลาทรานเซคชันไว้ ณ จุดนี้ ทำให้ข้อมูลก่อนหน้านี้เป็นข้อมูลเก่าที่เคยเชื่อในอดีตไป

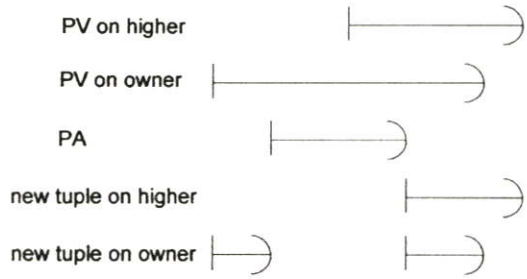


รูปที่ 6.14 ผลการปรับปรุงข้อมูลทำให้เกิดแถวใหม่ทั้งระดับตนเองและระดับสูงกว่าได้

คำสั่ง DELETE มีลักษณะคล้ายกับคำสั่ง UPDATE เพียงแต่ไม่ต้องสนใจในการสร้างแถวใหม่ในส่วนของข้อมูลที่มีการแก้ไขเท่านั้น

6.4 การพัฒนาคำสั่ง DELETE

คำสั่งนี้มีผลได้ทั้งในระดับตนเองและระดับสูงกว่าที่มีการยืมข้อมูลจากระดับนี้ไป แต่เป็นคำสั่งที่ง่ายที่สุดเพียงแค่การเพิ่มแถวใหม่ที่ซ้อนทับอยู่ในกรณีกับคาบเวลา PA ดังรูปที่ 6.15 ประกอบด้วยสามคำสั่ง INSERT และอีกหนึ่งคำสั่ง UPDATE สองคำสั่ง INSERT แรกเป็นการปรับส่วนที่ไม่ไม่อยู่ตรงกับคาบเวลา PV แต่เหลื่อมล้ำกันอยู่ทางซ้ายและขวาตามลำดับ และมีผลกระทบต่อระดับบนด้วยคำสั่ง INSERT ที่สาม และทำการหยุดเวลาความเชื่อของเวลาทรานเซคชันกับแถวที่ก่อนทำการลบด้วยคำสั่ง UPDATE



รูปที่ 6.15 คาบเวลาจากการลบที่ส่งผลกระทบระดับต่างๆ

6.5 การพัฒนาคำสั่ง UPLEVEL

คำสั่ง UPLEVEL เป็นคำสั่งที่นำมาใช้เพื่อให้สามารถดึงข้อมูล และเป็นการยืนยันข้อมูลในระดับต่ำกว่ามาเป็นข้อมูลของในระดับตนได้ นอกเหนือจากนั้นเมื่อระดับล่างทำการอัพเดทข้อมูลก็ส่งผลมาสู่ระดับบนให้อัพเดทตามด้วย คำสั่ง UPLEVEL นี้ออกแบบมาให้มีลักษณะเป็น INSERT หรือ UPDATE คือถ้าหากไม่มีอยู่ให้ทำการ INSERT หรือถ้าหากมีแถวนั้นอยู่แล้วให้ทำการ UPDATE เป็นเรื่องน่าสนใจเมื่อต้องใช้ INSERT หรือ UPDATE ในคาบเวลา เนื่องจากถ้าเรามองในฐานข้อมูลปกติแล้ว สามารถทำได้ไม่ยาก แต่ในคำสั่ง UPLEVEL ของเอ็มเอสบีดีนั้นมีอยู่ด้วยกันถึง 4 กรณี และตัวอย่างต่อไปนี้เป็นกรณีที่ทำให้เกิดทั้ง 4 กรณีเป็นผลให้เกิดแถวใหม่ขึ้นมา 4 แถว จากข้อมูลตัวอย่างของตาราง TRACKING_REPORS ในตารางที่ 6.4 นักสืบระดับ S1 มีข้อมูลของ ROBERT SIMSON ที่ระบุว่าเขาได้เล่นกอล์ฟอยู่ที่ ALPINE COURSE ดังแสดงในแถวแรกและข้อมูลในระดับ S1 นี้เป็นข้อมูลที่สร้างขึ้นโดยนักสืบระดับ S1 ทั้งหมด แต่ในวันที่ 21 กันยายน

2006 นักสืบในระดับ S1 ต้องการแก้ไขข้อมูลโดยให้ความเชื่อกับข้อมูลจากระดับ U1 ที่เชื่อว่ากำลังพักผ่อนอยู่โดยให้ทำการดึงข้อมูลส่วนนี้ขึ้นมาตั้งแต่วันที่ 5 ถึง 15 กันยายน 2006 สามารถเขียนเป็นคำสัสส่งเอสคิวแอลในเอ็มเอสบีดีโมเดลได้ดังนี้

```
VALIDTIME '[2006-09-02, 2006-09-20)'
UPLEVEL TRACKING_REPORTS
GET ACTION FROM U1
WHERE TNAME = 'ROBERT SIMSON'
```

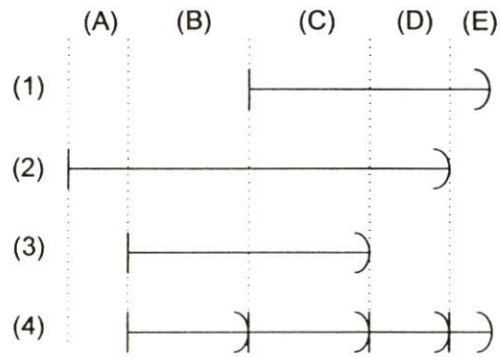
ตารางที่ 6.4 ข้อมูลตัวอย่างก่อนใช้คำสั่ง UPLEVEL

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
ROBERT SIMSON	S1	PLAY GOLF	S1	ALPINE COURSE	S1	2006-09-10	2006-09-25	2006-09-15	9999-12-31	S1
ROBERT SIMSON	U1	RELAX	U1	ALPINE COURSE	U1	2006-09-05	2006-09-15	2006-09-20	9999-12-31	U1

ตารางที่ 6.5 ข้อมูลตัวอย่างหลังใช้คำสั่ง UPLEVEL

TNAME	*TN_C*	*ACTION*	*AT_C*	*LOCAT*	*LC_C*	*VT_S*	*VT_E*	*TT_S*	*TT_E*	*TC*
ROBERT SIMSON	S1	PLAY GOLF	S1	ALPINE COURSE	S1	2006-09-10	2006-09-15	2006-09-15	2006-09-21	S1
ROBERT SIMSON	U1	RELAX	U1	NULL	S1	2006-09-05	2006-09-10	2006-09-21	9999-12-31	S1
ROBERT SIMSON	U1,S1	RELAX	U1	ALPINE COURSE	S1	2006-09-10	2006-09-15	2006-09-21	9999-12-31	S1
ROBERT SIMSON	S1	NULL	U1	ALPINE COURSE	S1	2006-09-15	2006-09-20	2006-09-21	9999-12-31	S1
ROBERT SIMSON	S1	PLAY GOLF	S1	ALPINE COURSE	S1	2006-09-20	2006-09-25	2006-09-21	9999-12-31	S1
ROBERT SIMSON	U1	RELAX	U1	ALPINE COURSE	U1	2006-09-05	2006-09-20	2006-09-20	9999-12-31	U1

ผลลัพธ์จากการใช้คำสั่ง UPLEVEL แสดงในตารางที่ 6.5 เนื่องจากประกอบด้วยหลายกรณีการซ้อนทับดังที่ได้แสดงในรูปที่ 6.12 ทำให้เกิดแถวเพิ่มขึ้นถึงสี่แถว



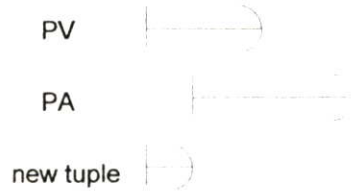
รูปที่ 6.16 คาบเวลาที่เกิดจากคำสั่ง UPLEVEL

การดึงข้อมูลระดับต่ำกว่ามาเป็นระดับของตน และเป็นคำสั่งที่เป็นทั้งการ UPDATE แถวเดิม (ในเอ็มเอสบีดีคือการสร้างใหม่แล้วให้ของเก่าหมดความเชื่อไป) รวมทั้งการ INSERT tuple ใหม่ ด้วย จากรูปที่ 6.16 (1) เป็นข้อมูลเดิมที่มีอยู่ (2) เป็นคาบเวลาที่เข้ามากระทำจากคำสั่ง UPLEVEL (3) เป็นคาบเวลาจากระดับล่างที่เราต้องการดึงข้อมูล (4) เป็นผลลัพธ์จากคำสั่ง UPLEVEL คาบเวลาจะถูกแบ่งออกไป ในแนวนอนด้านบน เราแสดงให้เห็นกรณีต่างๆ ที่เป็นไปได้ระหว่างข้อมูล ระดับผู้ใช้กับระดับล่าง (แต่ยังไม่รวมถึงกรณีการซ้อนทับกัน, เหลื่อมซ้าย เหลื่อมขวา หรือซ้อนทับทั้งหมด) ใน (A) เป็นส่วนที่ไม่มีข้อมูลในระดับล่าง และในระดับของเราก็ไม่มีข้อมูลใดๆ ด้วยในเวลานั้น จึงไม่เกิดผลอะไรใน (4) ส่วน (B) เป็นส่วนที่ในระดับผู้ใช้ไม่มีข้อมูลในช่วงเวลานั้น แต่ในระดับล่างมี และอยู่คาบเวลามีผลของ UPLEVEL ดังนั้น จึงต้องสร้างแถวที่มีคาบเวลาดังใน (4) ส่วนทางซ้าย ในส่วน (C) ทั้งระดับผู้ใช้ และระดับล่างมีข้อมูลอยู่แล้ว ข้อมูลในระดับบนต้องทำการ UPDATE ตามระดับล่าง ส่วน (D) มีข้อมูลอยู่ในระดับผู้ใช้แต่ไม่มีข้อมูลสำหรับดึงในระดับล่าง ให้ทำการ UPDATE ข้อมูลส่วนนั้นเป็นนัลในส่วนสุดท้าย (E) อยู่นอกคาบเวลากระทำของคำสั่ง UPLEVEL ข้อมูลคงเดิม แต่ต้องถูก INSERT ใหม่ แล้วให้ข้อมูลเดิม (1) เป็นอดีตไป คำสั่ง UPLEVEL ต้องอาศัยถึง 9 INSERT เนื่องจากมีกรณีดังรูปที่ 6.16 ที่มีส่วนเหลื่อมล้ำในส่วนกลางการขาดช่วงของคาบเวลาร่วมด้วย เห็นว่าจากข้อมูลเดิมในหนึ่งคาบเวลาต่อเนื่อง (1) เราได้ข้อมูลใน (4) เพิ่มถึงห้าคาบเวลาติดกัน

เห็นได้ว่าเรามีถึง สี่รูปแบบ ในแต่ละรูปแบบก็ต้องทำการตรวจตามลักษณะของการซ้อนทับกันดด้วยเช่นเหลื่อมซ้ายและขวา หรือครอบคลุมทั้งหมด เมื่อเรานำไปใช้กับคำสั่งเอสคิวแอลมาตรฐานในฐานข้อมูลแบบปกติ ประกอบด้วยชุดคำสั่งต่อไปนี้

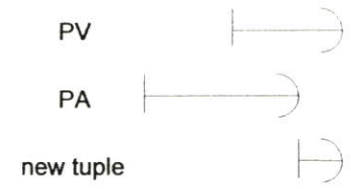
```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
   LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT TNAME, TNAME_C, ACTION, ACTION_C, LOCAT, LOCAT_C,
   VT_S, DATE '2006-09-02', DATE '2006-09-21', TT_E, TC, CC
FROM TRACKING_REPORTS
WHERE TNAME = 'ROBERT SIMSON'
    AND TC = 'S1' AND CC = TC
    AND VT_S < DATE '2006-09-02' AND VT_E > DATE '2006-09-02'
    AND TT_E = DATE '9999-12-31'

INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
   LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT TNAME, TNAME_C, ACTION, ACTION_C, LOCAT, LOCAT_C,
   DATE '2006-09-20', VT_E, DATE '2006-09-21', TT_E, TC, CC
FROM TRACKING_REPORTS
WHERE TNAME = 'ROBERT SIMSON'
    AND TC = 'S1' AND CC = TC
    AND VT_S < DATE '2006-09-20' AND VT_E > DATE '2006-09-20'
    AND TT_E = DATE '9999-12-31'
```

ในคำสั่ง INSERT แรกทำการตัดข้อมูลทางด้านซ้ายออก ด้วยการเพิ่มข้อมูลเข้าไปใหม่ดังรูปที่ 6.17



รูปที่ 6.17 เพิ่มคาบเวลาทางซ้ายจากคำสั่ง UPLEVEL

ข้อมูลในตาราง 6.4 ตั้งแต่วันที่ 20 ถึง 25 กันยายน 2006 มีอยู่ในระดับ S1 และไม่ครอบคลุมถึงคาบเวลาของคำสั่ง UPLEVEL ดังนั้นจึงต้องทำการรักษาข้อมูลเดิมในระดับ S1 นี้ไว้ว่า ROBERT SIMSON ได้เล่นกอล์ฟที่ ALPINE CORSE ตั้งแต่วันที่ 20 ถึง 25 กันยายน 2006 คำสั่ง INSERT ที่สองทำการตัดข้อมูลทางด้านขวาออก ด้วยการเพิ่มข้อมูลเข้าไปใหม่ดังในรูปที่ 6.18 และผลลัพธ์แสดงในตารางที่ 6.5 แถวที่ 5



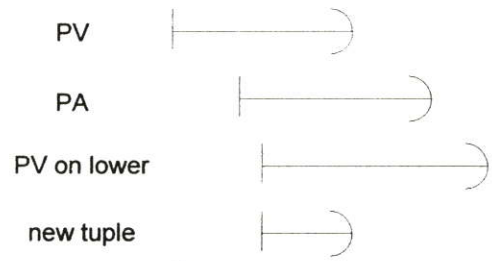
รูปที่ 6.18 เพิ่มคาบเวลาทางขวาจากคำสั่ง UPLEVEL

ถ้าหากมีข้อมูลเดิมอยู่ในระดับ S1 แล้ว ช่วงเวลาที่อยู่ในแถวเดิมมีบางส่วนที่อยู่ในคาบเวลาการกระทำการของคำสั่ง UPLEVEL แล้ว ยังคงต้องรักษาข้อมูลเดิมเอาไว้ ด้วยการ INSERT แถวใหม่ให้มีข้อมูลเช่นเดิมในคาบเวลาที่อยู่นอกเหนือการกระทำของคำสั่ง UPLEVEL ประกอบด้วยสองคำสั่ง INSERT ในสองกรณีคือ เหลื่อมล้ำกันทางซ้ายและขวาดังรูปที่ 6.17 และ 6.18 ในช่วงคาบเวลาตั้งแต่วันที่ 10 ถึง 15 กันยายน 2006 จากตารางที่ 6.4 มีข้อมูลเดิมอยู่ในระดับ S1 และมีข้อมูลที่ต้องการดึงขึ้นมาในระดับ U1 และอยู่ภายใต้คาบกระทำของคำสั่ง UPLEVEL ดังนั้นต้องทำการดึงข้อมูลว่า ROBERT SIMSON กำลังพักผ่อนอยู่ซึ่งเป็นข้อมูลจาก U1 และข้อมูลสถานที่ที่ไม่รวมอยู่ในอนุประโยค GET ให้รักษาข้อมูลเดิมคือ ALPINE COURSE เอาไว้ผลลัพธ์แสดงในแถวที่สามของตารางที่ 6.5 โดยใช้ชุดคำสั่ง INSERT ในการดึงข้อมูลในส่วนที่ซ้อนทับกัน และอยู่ภายใต้ PA ดังนี้

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
   LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R1.TNAME, ADD_LV( R1.TNAME_C, 'U1'), R2.ACTION, 'U1',
   R1.LOCAT, R1.LOCAT_C,
   CASE WHEN R1.VT_S >= R2.VT_S AND R1.VT_S > DATE '2006-09-02'
      THEN R1.VT_S
```

```
    WHEN R2.VT_S > R1.VT_S AND R2.VT_S > DATE '2006-09-02'
      THEN R2.VT_S
    ELSE DATE '2006-09-02' END,
  CASE WHEN R1.VT_E <= R2.VT_E AND R1.VT_E < DATE '2006-09-20'
      THEN R1.VT_E
    WHEN R2.VT_E < R1.VT_E AND R2.VT_E < DATE '2006-09-20'
      THEN R2.VT_E
    ELSE DATE '2006-09-20' END, DATE '2006-09-21', DATE '9999-12-31', R1.TC, R1.CC
FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2
WHERE R1.TNAME = 'ROBERT SIMSON' AND R1.TNAME = R2.TNAME
  AND R1.TC = 'S1' AND R1.CC = R1.TC
  AND R1.ACTION_C <> 'U1'
  AND R2.TC = 'U1' AND R2.CC = R2.TC AND R2.ACTION_C = R2.TC
  AND R1.VT_S < R2.VT_E AND R2.VT_S < R1.VT_E
  AND R1.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R1.VT_E
  AND R2.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R2.VT_E
  AND R1.TT_E = DATE '9999-12-31' AND R1.TT_E = R2.TT_E;
```

ทำการดึงข้อมูลขึ้นมาในส่วนที่มีข้อมูลอยู่ในระดับล่างจริง ในที่นี้ได้ใช้ฟังก์ชัน ADD_LV เป็นฟังก์สำหรับเพิ่มระดับความปลอดภัยของระดับล่างที่ทำการดึงข้อมูล ไปเป็นส่วนหนึ่งของเซทระดับความปลอดภัยของคีย์หลักแอพพาเรนท์ คาบเวลาจากทำสั่ง INSERT นี้แสดงในรูปที่ 6.19

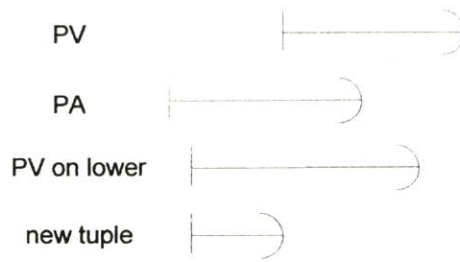


รูปที่ 6.19 คาบใหม่ที่เกิดขึ้นจากการดึงข้อมูลที่มีอยู่ในระดับล่าง

ในชุดคำสั่งถัดมาทำการดึงข้อมูลในส่วนที่ซ้อนทับกันของ PV ของทั้งสองระดับเท่านั้นแบ่งออกเป็นสามส่วน ส่วนแรกทำการดึงข้อมูลจากระดับล่างมาสู่ข้อมูลในระดับของตนที่ยังไม่มีข้อมูลอะไรอยู่ เฉพาะทางซ้าย ดังรูปที่ 6.20

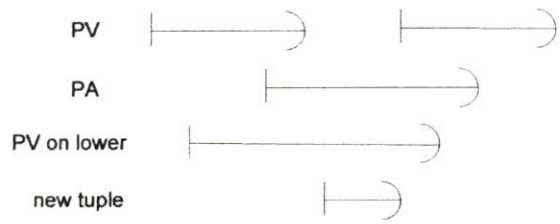
```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
  LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R1.TNAME, 'U1', R2.ACTION, 'U1', null, 'S1',
  CASE WHEN R2.VT_S > DATE '2006-09-02'
    THEN R2.VT_S
  ELSE DATE '2006-09-02' END,
  CASE WHEN R1.VT_S > R2.VT_E
    THEN R2.VT_E
  ELSE R1.VT_S END, DATE '2006-09-21', DATE '9999-12-31', R1.TC, R1.CC
FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2
WHERE R1.TNAME = 'ROBERT SIMSON'
  AND R2.TNAME = R1.TNAME
  AND R1.ACTION_C <> 'U1'
  AND R2.TC = 'U1' AND R2.CC = R2.TC AND R2.ACTION_C = R2.TC
```

```
AND R2.VT_S < R1.VT_S AND DATE '2006-09-02' < R2.VT_E
AND R1.TC = 'S1' AND R1.CC = R1.TC
AND R1.VT_S > DATE '2006-09-02' AND R1.VT_S < DATE '2006-09-20'
AND NOT EXISTS(
  SELECT R3.TNAME
  FROM TRACKING_REPORTS R3
  WHERE R1.TNAME = R3.TNAME
    AND R3.ACTION_C <> 'U1'
    AND R3.TC = R1.TC AND R3.TC = R3.CC
    AND R3.VT_S < R1.VT_S AND R3.VT_E > DATE '2006-09-02'
    AND R3.TT_E = DATE '9999-12-31'
  )
AND R1.TT_E = DATE '9999-12-31'
AND R2.TT_E = DATE '9999-12-31';
```



รูปที่ 6.20 คาบเวลาเฉพาะส่วนทางซ้ายเป็นข้อมูลคาบเวลาว่างทางซ้าย

คำสั่ง INSERT ถัดทำการดึงข้อมูลจากระดับล่างมาสู่ข้อมูลในระดับตนที่ยังไม่มีข้อมูลอะไรอยู่เฉพาะอยู่ระหว่างกลางดังแสดงในรูปที่ 6.21



รูปที่ 6.21 คาบเวลาเฉพาะส่วนซ้อนกับส่วนที่ไม่มีข้อมูลเดิมอยู่ตรงกลาง

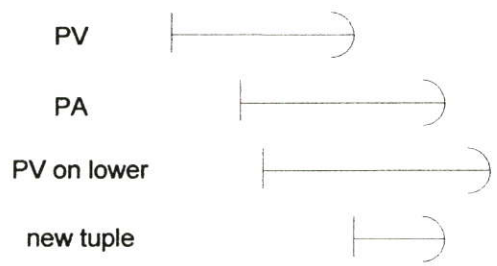
มีการใช้คำสั่ง INSERT ดังนี้

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
  LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R1.TNAME, 'U1', R4.ACTION, 'U1', null, 'S1',
   CASE WHEN R4.VT_S > R1.VT_E
     THEN R4.VT_S
   ELSE R1.VT_E END,
   CASE WHEN R2.VT_S > R4.VT_E
     THEN R4.VT_E
   ELSE R2.VT_S END,
   DATE '2006-09-21', DATE '9999-12-31', R1.TC, R1.CC
FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2, TRACKING_REPORTS R4
WHERE R1.TNAME = 'ROBERT SIMSON' AND R1.TNAME = R2.TNAME
```

```
  AND R4.TNAME = R1.TNAME
  AND R1.TC = 'S1' AND R1.CC = R1.TC
  AND R2.TC = 'S1' AND R2.CC = R2.TC
  AND R4.TC = 'U1' AND R4.CC = R4.TC AND R4.ACTION_C = R4.TC
  AND R1.VT_E < R2.VT_S
  AND R1.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R1.VT_E
  AND R2.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R2.VT_E
  AND R1.ACTION_C <> 'U1'
  AND R2.ACTION_C <> 'U1'
  AND R4.VT_S < R2.VT_S AND R1.VT_E < R4.VT_E
  AND NOT EXISTS(
     SELECT R3.TNAME
     FROM TRACKING_REPORTS R3
     WHERE R3.TNAME = R1.TNAME
      AND R3.ACTION_C <> 'U1'
      AND R3.TC = R1.TC AND R3.TC = R3.CC
      AND R1.VT_E < R3.VT_E AND R3.VT_S < R2.VT_S
      AND R3.TT_E = DATE '9999-12-31'
     )
 AND R1.TT_E = DATE '9999-12-31' AND R1.TT_E = R2.TT_E AND R4.TT_E = R1.TT_E;
```

ในการหาคาบว่างส่วนกลางนี้ใช้การจอยระหว่างสองตารางเพื่อหาคาบเวลาที่ว่างอยู่ในข้อมูลเดิม และใช้การจอยอีกตารางเพื่อดึงข้อมูลจากระดับล่าง

คำสั่ง INSERT ถัดมาคล้ายกับคำสั่งก่อนหน้าที่ตรวจสอบหาการเหลื่อมล้ำที่ยังไม่มีข้อมูลในระดับเดิมอยู่ และข้อมูลอยู่ในระดับต่ำกว่าที่ต้องการดึงข้อมูลแต่ในส่วนนี้ทำกับการเหลื่อมทางขวา แสดงในรูปที่ 6.22



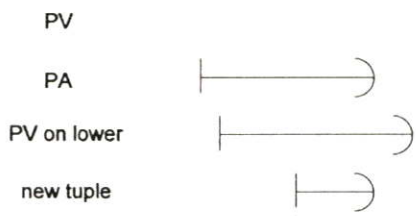
รูปที่ 6.22 คาบเวลาเฉพาะส่วนทางซ้ายเป็นข้อมูลคาบเวลาว่างทางขวา

มีการใช้คำสั่ง INSERT ดังนี้

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
   LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R1.TNAME, 'U1', R2.ACTION, 'U1', null, 'S1',
    CASE WHEN R2.VT_S > R1.VT_E
      THEN R2.VT_S
    ELSE R1.VT_E END,
    CASE WHEN DATE '2006-09-20' > R2.VT_E
     THEN R2.VT_E
    ELSE DATE '2006-09-20' END, DATE '2006-09-21', DATE '9999-12-31', R1.TC, R1.CC
FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2
WHERE R1.TNAME = 'ROBERT SIMSON'
    AND R1.ACTION_C <> 'U1'
    AND R2.TNAME = R1.TNAME
```

```
AND R2.TC = 'U1' AND R2.CC = R2.TC AND R2.ACTION_C = R2.TC
AND R2.VT_S < DATE '2006-09-20' AND R1.VT_E < R2.VT_E
AND R1.TC = 'S1' AND R1.CC = R1.TC
AND R1.VT_E < DATE '2006-09-20' AND R1.VT_E > DATE '2006-09-02'
AND NOT EXISTS(
  SELECT R3.TNAME
  FROM TRACKING_REPORTS R3
  WHERE R1.TNAME = R3.TNAME
    AND R3.ACTION_C <> 'U1'
    AND R3.TC = R1.TC AND R3.TC = R3.CC
    AND R3.VT_S < DATE '2006-09-20' AND R3.VT_E > R1.VT_E
    AND R3.TT_E = DATE '9999-12-31'
  )
AND R1.TT_E = DATE '9999-12-31'
AND R2.TT_E = DATE '9999-12-31';
```

จากข้อมูลในตารางที่ 6.4 ช่วงเวลาระหว่างวันที่ 5 ถึงวันที่ 10 กันยายน 2006 ไม่มีข้อมูลของ ROBERT SIMSON ในระดับ S1 แต่มีข้อมูลนี้อยู่ในระดับ U1 และในช่วงเวลานี้อยู่ในคาบเวลากระทำของคำสั่ง UPLEVEL ดังนั้นต้องมีการดึงข้อมูลจากระดับ U1 ขึ้นมาโดยสร้างแถวใหม่โดยคำสั่ง INSERT ทำการดึงข้อมูลทั้งหมดในกรณีที่ด้านบนไม่ซ้อนทับอะไรเลย และให้แอททริบิวอื่นเป็นนัล ดังรูปที่ 6.23 ผลจากคำสั่ง INSERT นี้อยู่ในแถวที่สองจากตารางที่ 6.5 ในแอททริบิว LOCAT ถูกกำหนดเป็น NULL เนื่องจากไม่ได้อยู่ในอนุประโยค GET



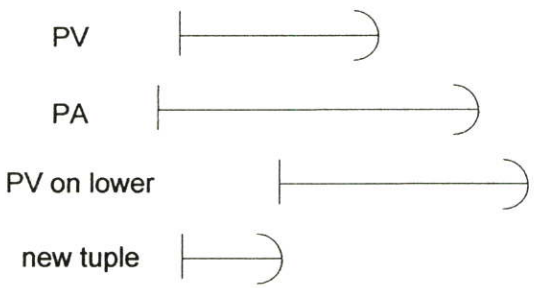
รูปที่ 6.23 คาบเวลาใหม่ที่มีการเหลื่อมล้ำของเวลากับข้อมูลเดิม

มีการใช้คำสั่ง INSERT ดังนี้

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
  LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R2.TNAME, 'U1', R2.ACTION, 'U1', null, 'S1',
  CASE WHEN R2.VT_S > DATE '2006-09-02'
    THEN R2.VT_S
  ELSE DATE '2006-09-02' END,
  CASE WHEN DATE '2006-09-20' > R2.VT_E
    THEN R2.VT_E
  ELSE DATE '2006-09-20' END,DATE '2006-09-21', DATE '9999-12-31', 'S1', 'S1'
FROM TRACKING_REPORTS R2
WHERE R2.TNAME = 'ROBERT SIMSON'
  AND R2.TC = 'U1' AND R2.CC = R2.TC AND R2.ACTION_C = R2.TC
  AND R2.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R2.VT_E
  AND NOT EXISTS(
    SELECT R1.TNAME
```

```
      FROM TRACKING_REPORTS R1
      WHERE R1.TNAME = 'ROBERT SIMSON'
       AND R1.TC = 'S1' AND R1.CC = R1.TC
       AND R1.VT_S < DATE '2006-09-20' AND R1.VT_E > DATE '2006-09-02'
       AND R1.TT_E = DATE '9999-12-31'
     )
  AND R2.TT_E = DATE '9999-12-31';
```

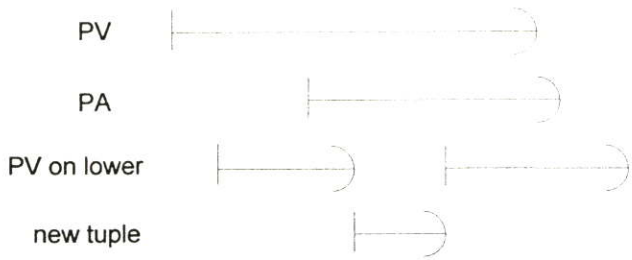
ในกรณีที่ไม่มีข้อมูลอยู่ในระดับล่างที่ต้องการดึงข้อมูลให้ทำการกำหนดให้เป็น NULL ถ้าหากมีข้อมูลเดิมอยู่แล้วในระดับตนเอง ในคำสั่ง INSERT ต่อไปนี้เป็นกรณีที่มีการเหลื่อมล้ำกันทางซ้าย ดังรูปที่ 6.24



รูปที่ 6.24 ผลจากเพิ่มแถวใหม่มีค่าเป็นนัลกรณีมีข้อมูลเดิมอยู่แต่ไม่มีในระดับล่างทางซ้าย

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
   LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
SELECT R1.TNAME, R2.TNAME_C, null, 'U1', R2.LOCAT, R2.LOCAT_C,
    CASE WHEN R2.VT_S > DATE '2006-09-02'
      THEN R2.VT_S
     ELSE DATE '2006-09-02' END,
     CASE WHEN R1.VT_S > R2.VT_E
      THEN R2.VT_E
     ELSE R1.VT_S END,
     DATE '2006-09-21', DATE '9999-12-31', R2.TC, R2.CC
FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2
WHERE R1.TNAME = 'ROBERT SIMSON'
    AND R2.TNAME = R1.TNAME
    AND R2.ACTION_C <> 'U1'
    AND R2.TC = 'S1' AND R2.CC = R2.TC
    AND R2.VT_S < R1.VT_S AND DATE '2006-09-02' < R2.VT_E
    AND R1.TC = 'U1' AND R1.CC = R1.TC AND R1.ACTION_C = R1.TC
    AND R1.VT_S > DATE '2006-09-02' AND R1.VT_S < DATE '2006-09-20'
    AND NOT EXISTS(
       SELECT R3.TNAME
       FROM TRACKING_REPORTS R3
       WHERE R1.TNAME = R3.TNAME
        AND R3.TC = R1.TC AND R3.TC = R3.CC AND R3.ACTION_C = R3.TC
        AND R3.VT_S < R1.VT_S AND R3.VT_E > DATE '2006-09-02'
        AND R3.TT_E = DATE '9999-12-31'
       )
   AND R1.TT_E = DATE '9999-12-31'
   AND R2.TT_E = DATE '9999-12-31';
```

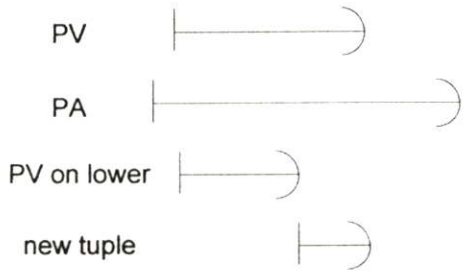
คำสั่ง INSERT ถัดมาทำเพื่อแก้ไขข้อมูลที่มีอยู่เดิมให้เป็นนัลกรณีไม่มีคาบเวลาในระดับต่ำกว่า แต่มีในระดับสูงกว่า ระหว่างกลางดังรูปที่ 6.25



รูปที่ 6.25 ผลจากเพิ่มแถวใหม่มีค่าเป็นนัลกรณีมีข้อมูลเดิมอยู่แต่ไม่มีในระดับล่างระหว่างกลาง

```
  INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C, LOCAT,
LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
  SELECT R1.TNAME, null, 'U1', R4.LOCAT, R4.LOCAT_C,
   CASE WHEN R4.VT_S > R1.VT_E
    THEN R4.VT_S
    ELSE R1.VT_E END,
   CASE WHEN R2.VT_S > R4.VT_E
    THEN R4.VT_E
    ELSE R2.VT_S END,
   DATE '2006-09-21', DATE '9999-12-31', R4.TNAME_C, R4.TC, R2.CC
  FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2, TRACKING_REPORTS R4
  WHERE R1.TNAME = 'ROBERT SIMSON' AND R1.TNAME = R2.TNAME AND R4.TNAME =
R1.TNAME
   AND R4.ACTION_C <> 'U1'
   AND R1.TC = 'U1' AND R1.CC = R1.TC AND R1.ACTION_C = R1.TC
   AND R2.TC = 'U1' AND R2.CC = R2.TC AND R2.ACTION_C = R2.TC
   AND R4.TC = 'S1' AND R4.CC = R4.TC
   AND R1.VT_E < R2.VT_S
   AND R1.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R1.VT_E
   AND R2.VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < R2.VT_E
   AND R4.VT_S < R2.VT_S AND R1.VT_E < R2.VT_E
   AND NOT EXISTS(
     SELECT R3.TNAME
     FROM TRACKING_REPORTS R3
     WHERE R3.TNAME = R1.TNAME
      AND R3.TC = R1.TC AND R3.TC = R3.CC AND R3.ACTION_C = R3.TC
      AND R1.VT_E < R3.VT_E AND R3.VT_S < R2.VT_S
      AND R3.TT_E = DATE '9999-12-31'
     )
   AND R1.TT_E = DATE '9999-12-31' AND R1.TT_E = R2.TT_E AND R4.TT_E = R1.TT_E;
```

จากข้อมูลของเป้าหมายในตารางที่ 6.4 ในช่วงเวลาระหว่างวันที่ 15 ถึง 20 กันยายน 2006 มีข้อมูลเดิมของ ROBERT SIMSON อยู่ในระดับ S1 แต่ไม่มีข้อมูลอยู่ในระดับ U1 และอยู่ในคาบเวลากระทำของคำสั่ง UPLEVEL ใช้คำสั่ง INSERT ในการแก้ไขข้อมูลที่มีอยู่เดิมให้เป็น NULL กรณีไม่มีคาบเวลาในระดับต่ำกว่า แต่มีในระดับสูงกว่า ทางขวาดังรูปที่ 6.26



รูปที่ 6.26 ผลจากเพิ่มแถวใหม่มีค่าเป็นนัลกรณีมีข้อมูลเดิมอยู่แต่ไม่มีในระดับล่างทางขวา

มีการใช้คำสั่ง INSERT ดังนี้

```
INSERT INTO TRACKING_REPORTS( TNAME, TNAME_C, ACTION, ACTION_C,
  LOCAT, LOCAT_C, VT_S, VT_E, TT_S, TT_E, TC, CC)
 SELECT R1.TNAME, R2.TNAME_C, null, 'U1', R2.LOCAT, R2.LOCAT_C,
   CASE WHEN R2.VT_S > R1.VT_E
    THEN R2.VT_S
   ELSE R1.VT_E END,
   CASE WHEN DATE '2006-09-20' > R2.VT_E
    THEN R2.VT_E
   ELSE DATE '2006-09-20' END,
    DATE '2006-09-21', DATE '9999-12-31', R2.TC, R2.CC
  FROM TRACKING_REPORTS R1, TRACKING_REPORTS R2
  WHERE R1.TNAME = 'ROBERT SIMSON'
   AND R2.TNAME = R1.TNAME
   AND R2.ACTION_C <> 'U1'
   AND R2.TC = 'S1' AND R2.CC = R2.TC
   AND R2.VT_S < DATE '2006-09-20' AND R1.VT_E < R2.VT_E
   AND R1.TC = 'U1' AND R1.CC = R1.TC AND R1.ACTION_C = R1.TC
   AND R1.VT_E < DATE '2006-09-20' AND R1.VT_E > DATE '2006-09-02'
   AND NOT EXISTS(
      SELECT R3.TNAME
      FROM TRACKING_REPORTS R3
      WHERE R1.TNAME = R3.TNAME
       AND R3.TC = 'U1' AND R3.TC = R3.CC AND R3.ACTION_C = R3.TC
       AND R3.VT_S < DATE '2006-09-20' AND R3.VT_E > R1.VT_E
       AND R3.TT_E = DATE '9999-12-31'
      )
   AND R1.TT_E = DATE '9999-12-31'
   AND R2.TT_E = DATE '9999-12-31';
```

ผลจากคำสั่ง INSERT นี้มีผลลัพธ์แสดงในตารางที่ 6.5 ในแถวที่สี่ สำหรับคำสั่ง UPDATE สุดท้ายเป็นการทำข้อมูลเดิมให้เป็นอดีตด้วยการ ทำให้ t[TT_E] มีค่าเป็นเวลาปัจจุบัน ด้วยการตรวจสอบว่าแถวใดที่มีอยู่ในช่วงเวลาของการ UPLEVEL ยังมีระดับความปลอดภัยไม่เป็นไปตามระดับล่างที่เราต้องการข้อมูลนั้นผลของคำสั่ง UPDATE นี้แสดงในตารางที่ 6.5 ในแถวแรกมีการใช้คำสั่ง UPDATE ดังต่อไปนี้

```
UPDATE TRACKING_REPORTS
SET TT_E = DATE '2006-09-21'
WHERE TNAME = 'ROBERT SIMSON'
 AND ACTION_C <> 'U1'
 AND TC = 'S1' AND TC = CC
 AND VT_S < DATE '2006-09-20' AND DATE '2006-09-02' < VT_E
 AND TT_E = DATE '9999-12-31';
```

6.6 การพัฒนาข้อบังคับการเกิดข้อมูลซ้ำซ้อน

กฎข้อบังคับในการเกิดข้อมูลซ้ำซ้อนของคีย์หลักที่ในฐานข้อมูลแบบเชิงสัมพันธ์แบบปกติไม่ยอมให้เกิด แต่ในโมเดลเอ็มเอสบีดีสามารถเกิดข้อมูลซ้ำของคีย์หลักแอพพาเรนท์ได้ในหลายรูปแบบทั้งความแตกต่างของเวลาข้อมูล รวมถึงเวลาที่เชื่อข้อมูล แล้วยังเกิดการซ้ำซ้อนของข้อมูลได้อีกในกรณีที่มีข้อมูลนั้นอยู่ในหลายระดับความปลอดภัย ดังนั้นเราต้องมีข้อกำหนดให้ อยู่ในขอบเขตที่ไม่มีการละเมิดหรือทำให้เกิดความกำกวมของข้อมูล ทั้งในความแตกต่างของระดับชั้นและเวลา เช่นเดียวกับป้องกันของคีย์หลักในฐานข้อมูลเชิงสัมพันธ์แบบปกติ สามารถป้องกันด้วยการใช้ทริกเกอร์สำหรับฐานข้อมูลของออราเคิลเป็นดังต่อไปนี้

```
CREATE OR REPLACE TRIGGER DETECTIVES_SEQ_DUPLICATE
AFTER INSERT OR UPDATE ON DETECTIVES
DECLARE
VALID INTEGER;
BEGIN
 SELECT 1 INTO VALID
 FROM DUAL
 WHERE NOT EXISTS(
  SELECT D1.CODENAME
  FROM DETECTIVES D1, DETECTIVES D2
  WHERE D1.CODENAME = D2.CODENAME AND D1.TC = D2.TC AND D1.CC = D2.CC
   AND D1.TT_S < D2.TT_E AND D2.TT_S < D1.TT_E
   AND D1.VT_S < D2.VT_E AND D2.VT_S < D1.VT_E
   AND D1.ROWID != D2.ROWID
  );
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  RAISE_APPLICATION_ERROR( -20001, 'CODENAME MUST BE SEQUENCED UNIQUE');
END DETECTIVES_SEQ_DUPLICATE;
```

ข้อแตกต่างกับ [12] ในส่วนนี้คือให้คีย์หลักแอพพาเรนท์เท่านั้นที่มีอยู่ได้ ณ เวลาหนึ่งๆ เพียงตัวเดียว ในที่นี้หมายความว่าเปรียบเสมือนหนึ่งแถวที่จำแนกด้วยคีย์หลักปกติ ในเวลาหนึ่งมีเพียงตัวเดียว ซึ่งจะรองรับข้อบังคับเช่นปกติเหมือนกับ ฐานข้อมูลเชิงสัมพันธ์มาตรฐาน

ไม่มีเอกสารจากต้นฉบับ หน้า 84

บทที่ 7
สรุปผลการวิจัย และข้อเสนอแนะ

ความต้องการของความสามารถของฐานข้อมูลในรูปแบบอื่นที่ไม่รองรับด้วยฐานข้อมูลแบบปกติ เมื่อมีบางแอพพลิเคชันต้องการความสามารถในการจัดการข้อมูลที่มากขึ้น การปรับฐานข้อมูลแบบปกติให้รองรับความสามารถที่ต้องการเป็นการง่ายต่อความเข้าใจ และง่ายต่อการนำไปใช้งาน ความสามารถเพิ่มเติมที่เราได้นำเสนอในวิทยานิพนธ์นี้คือ การให้ข้อมูลสามารถแปรเปลี่ยนตามเวลาในสองรูปแบบเวลา ทำให้สามารถดูข้อมูลภายในฐานข้อมูลโดยสามารถกำหนดเวลาของข้อมูล นั่นคือข้อมูลที่บันทึกในฐานข้อมูลสามารถเปลี่ยนแปลงได้เสมอดังนั้นการบอกเวลาว่าในอดีต หรือปัจจุบันข้อมูลนั้นเป็นอย่างไร และในการแก้ไขข้อมูลแต่ละครั้งรวมถึงการลบข้อมูล ก็ยังสามารถเรียกดูด้วยการกำหนดเวลาของการบันทึกอีกตัวหนึ่งด้วย ดังนั้นข้อมูลที่อย่างในฐานข้อมูลนี้ จะไม่มีทางศูนย์หายจากการแก้ไขข้อมูล สามารถเรียกตรวจสอบการเปลี่ยนแปลงได้ นอกเหนือจากนั้นยังมีการแบ่งระดับชั้นของข้อมูลออกเป็นหลายระดับด้วย โดยการแบ่งข้อมูลนี้เป็นการใช้ฐานข้อมูลร่วมกัน ตารางเดียวกัน แต่มุมมองที่ผู้ใช้ต่างระดับมองข้อมูลเดียวกันไม่เหมือนกัน

งานวิจัยนี้สร้างความเข้าใจต่อฐานข้อมูลที่มีความซับซ้อนในเรื่องของการมีหลายระดับความปลอดภัยบนตารางเดียวกัน รวมทั้งข้อมูลมีการแปรเปลี่ยนตามเวลาด้วย ได้ทำการปรับปรุงจากแนวความคิดของฐานข้อมูลเชิงสัมพันธ์ โดยการรวมทั้งสองคุณสมบัติเข้าด้วยกันให้มีความสอดคล้องของข้อมูล ด้วยการวางกฎเกณฑ์ไม่ให้เกิดความกำกวมของข้อมูล พร้อมทั้งได้นิยามความหมายของข้อมูลภายในฐานข้อมูลเพื่อให้เป็นรูปแบบของการกำหนดโครงสร้างภายในฐานข้อมูลสำหรับรูปแบบข้อมูลที่มีคุณสมบัติพิเศษนี้ การกำหนดรูปแบบของคำสั่งการใช้งาน วิธีการไควรี และวิธีการปรับปรุงข้อมูล นอกเหนือจากนั้นเราได้ทำการพัฒนาทดสอบด้วยฐานข้อมูลเชิงสัมพันธ์ ให้สามารถทำงานในรูปแบบความต้องการของงานวิจัยนี้ได้

สำหรับแอพพลิเคชันที่ต้องการคุณสมบัติของฐานข้อมูลประเภทนี้ค่อนข้างเป็นลักษณะงานแบบเฉพาะเจาะจง คือต้องมีความต้องการการแบ่งข้อมูลเป็นหลายระดับชั้น โดยผู้ใช้หลายระดับซึ่งไม่เหมือนกับคุณสมบัติของเวลาเพียงอย่างเดียวที่สามารถรวมกับแอพพลิเคชันอื่นได้ง่ายกว่าเนื่องจากข้อมูลมีการแปรเปลี่ยนตามเวลาอยู่แล้ว การกำกับเวลาเข้าไปด้วยทำให้ดูข้อมูลในช่วงเวลาต่างๆ ได้ก็เป็นข้อดีอย่างหนึ่ง ดังนั้นแอพพลิเคชันที่เหมาะสมกับงานวิจัยนี้ควรเป็นงานที่ต้องการคุณสมบัติของการมีระดับความปลอดภัยหลายระดับเป็นหลัก เนื่องจากถ้าปรับให้ความสามารถนี้ไปใช้งานกับกับแอพพลิเคชันทั่วไปที่มีเพียงระดับเดียวก็จะไม่เกิดประโยชน์อะไรเลย แต่ในคุณสมบัติของเวลาแม้ว่าไม่มีความต้องการอย่างเจาะจง เมื่อนำมาใช้อย่างน้อยก็สามารถเรียกดูข้อมูลก่อนหน้าที่แก้ไขไปแล้วได้ ดังนั้นการใช้งานจึงเหมาะสมกับข้อมูลที่ต้องการจำกัด

การเผยแพร่ข้อมูล หรือปิดบังข้อมูลที่แท้จริง เช่นข้อมูลของโรงพยาบาลที่ต้องการปกปิดข้อมูลของผู้ป่วยอาจเป็นชื่อโรคที่สังคมไม่ยอมรับหรือชื่อผู้ที่เข้ารับบริการทำศัลยกรรม หรือเวบไซท์ที่สามารถแสดงข้อมูลให้เหมาะสมตามอายุของผู้เข้าชมได้

สิ่งที่น่าท้าทายสำหรับงานวิจัยนี้คือการลดความซ้ำซ้อนของข้อมูล ในงานวิจัยนี้ได้เลือกใช้โมเดลเชิงสัมพันธ์เป็นหลักในการสร้างโมเดลข้อมูลใหม่ขึ้นมา แต่เนื่องจากข้อจำกัดบางประการของฐานข้อมูลประเภทนี้ทำให้ต้องยอมให้เกิดความซ้ำซ้อนของข้อมูลที่อยู่ร่วมกันในระดับความปลอดภัยหลายระดับในตารางเดียวกัน และเก็บข้อมูลในทุกลำดับเหตุการณ์ของเวลาทำให้ไม่สามารถหลีกเลี่ยงการซ้ำซ้อนของข้อมูลได้มากนัก อย่างไรก็ดีหากเปลี่ยนแนวความคิดนำไปใช้กับฐานข้อมูลประเภทอื่นเช่น ฐานข้อมูลวัตถุเชิงสัมพันธ์ หรือฐานข้อมูลเชิงวัตถุ ก็อาจเป็นทางออกที่ดีก็ได้ โดยการปรับโครงสร้าง และรูปแบบการเก็บข้อมูล เพื่อลดความซ้ำซ้อนของข้อมูลที่เกิดขึ้น เพื่อเพิ่มประสิทธิภาพความเร็วในการไควรี และลดพื้นที่จัดเก็บข้อมูลให้น้อยลง อย่างไรก็ได้การปรับปรุงกับรูปแบบของฐานข้อมูลอื่น เราหวังว่าสามารถใช้งานวิจัยของเราเป็นแนวทางการปรับปรุงต่อไปได้

บรรณานุกรม

[1] Pissinou, Niki, Kia Makki, and E. K. Park. "**Towards a framework for integrating secure models and temporal databases**". Proc. of Third International Conference on Information and Knowledge Management, 1994, pp 280-287.

[2] Bell, D.E. and LaPadula, L.J. "**Secure Computer Systems: Unified Exposition and Multics Interpretation.**" MTR-2997, MITRE (1975).

[3] Denning D. E. (1988), "**The Sea View Security Model**", Proceedings: IEEE Symposium on Security and Privacy, Oakland, Ca, April, 218-233.

[4] Jajodia S. and Sandhu R. (1991), "**Toward a Multilevel Secure Relational Data Model**", Proceedings: ACM SIGMOD, Denver, Colorado, May, ACM, New York, 50-59.

[5] Sandhu R., and Chen F., (1995), "**The Semantics and Expressive Power of the MLR Data Model**", Proceedings: IEEE Symposium on Security and Privacy, Oakland, Ca, May, 128-142.

[6] Smith K. and M. Winslett. "**Entity modeling in the MLS relational model**". Proceedings of Eighteenth VLDB, pp 199-210, 1992.

[7] Jukic N, Vrbsky SV. "**Asserting Beliefs in MLS Relational Models**". SIGMOD Record, Vol. 26,No.3,pp.30-35,1997.

[8] Jukic N., Vrbsky S., Parrish A., Dixon B, Jukic B. "**A Belief-Consistent Multilevel Secure Relational Data Model**". Information Systems, Vol. 24, No. 5, pp. 377-402, 1999.

[9] Pranjic M., Jukic N., Fertalj K. "**Implementing Belief-Consistent Multilevel Secure Relational Data Model**: Issues and Solutions". 25th Int. Conf. Information Technology Interfaces, 2003

[10] Pranjic M., Fertalj K., Jukic N. "**Importance of Semantics in MLS Database Models**". 24th Inf. Conf. Inoformation Technology Interfaces, 2002

[12] R. Snodgrass. "**Managing Temporal Data A Five-Part Series**". A Timecenter Technical Report", 1998

[13] Worawit M., Suphamit C., "**A Multilevel Secure Bitemporal Database System**". International Conference on Advances in Intelligent Systems, 2004

ภาคผนวก

ภาคผนวก ก.

การใช้งานผ่านแอพพลิเคชัน

ในส่วนนี้แสดงถึงการใช้งานโมเดลเอ็มเอสบีดีที่แสดงด้วยตัวอย่างการติดตามข้อมูลบุคคลจากสำนักงานนักสืบ โดยการทำงานของแอพพลิเคชันทั้งหมดเป็นตัวอย่างจากคำสั่งเอสคิวแอลในบทที่ 5 เพื่อให้เห็นความสามารถของโมเดลเอ็มเอสบีดีเมื่อเป็นมุมมองของผู้ใช้งานแอพพลิเคชัน

- **เครื่องมือในการพัฒนาแอพพลิเคชัน**

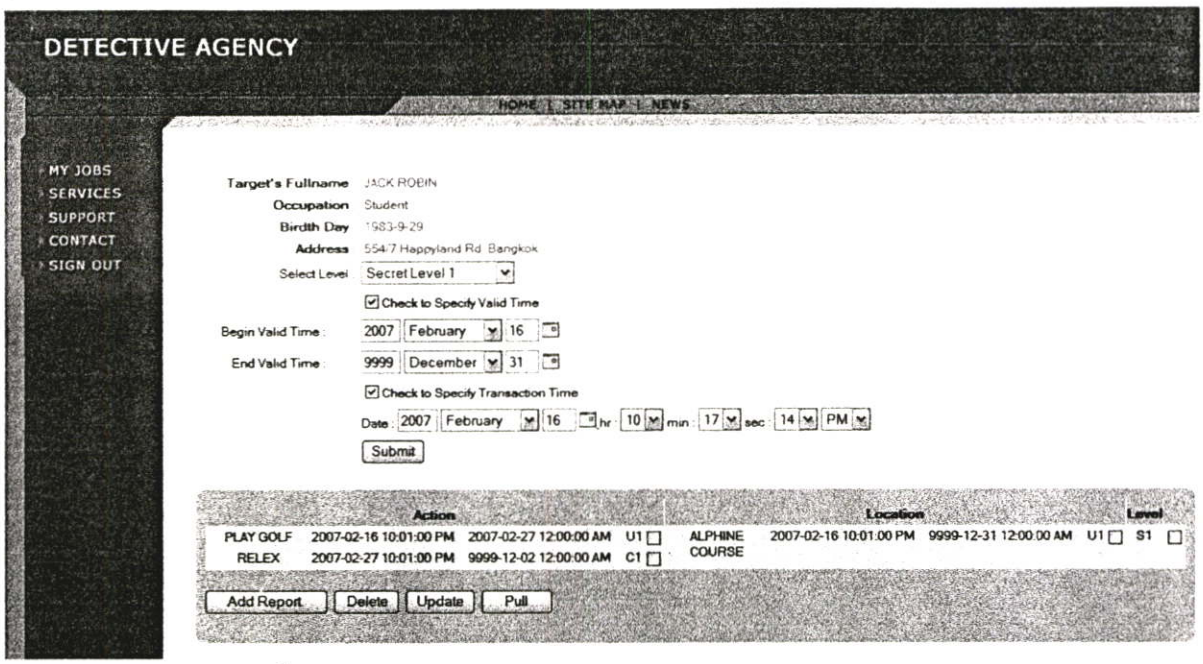
เราได้ใช้แนวความคิดในการพัฒนาในบทนี้ทดลองสร้างเป็นแอพพลิเคชันสำนักงานนักสืบขึ้นมาโดยใช้ฐานข้อมูลออราเคิลเวอร์ชัน 9i และใช้งานร่วมกันเป็นเวบแอพพลิเคชันกับ ASP.NET

- **ตัวอย่างการทำงานผ่านแอพพลิเคชัน**

การทำงานผ่านแอพพลิเคชันที่มีพื้นฐานจากการสั่งผ่านคำสั่งเอสคิวแอลที่มีความซับซ้อนของการสั่งงานนอกเหนือจากรูปแบบเอสคิวแอลมาตรฐาน ยังมีการควบคุมในส่วนของระดับความปลอดภัยที่เข้าถึงข้อมูล และเวลาที่กำกับข้อมูลไม่ว่าเป็นเวลาวาลิด หรือเป็นเวลาทรานเซคชัน การใช้งานผ่านแอพพลิเคชันเป็นเรื่องที่ง่ายกว่ามาก ด้วยรูปแบบที่มีเครื่องมือการเรียกดูรายงาน หรือการปรับปรุงข้อมูลที่เข้าใจง่าย

- **การเรียกดูข้อมูล**

การเรียกดูข้อมูลผ่านเวบแอพพลิเคชันถูกจำกัดความสามารถของการมองข้อมูลด้วยระดับชั้นของนักสืบ รูปที่ ก.1 เป็นภาพตัวอย่างการเรียกดูข้อมูลจากนักสืบในระดับ S1 เพื่อดูข้อมูลการติดตามเป้าหมายชื่อ JACK ROBIN ซึ่งสามารถกำหนดช่วงเวลาที่เป้าหมายทำอะไรอยู่ที่ไหนได้ด้วยการกำหนด คาบเวลาให้เริ่มตรวจสอบข้อมูลตั้งแต่วันที่ 16 กุมภาพันธ์ 2007 ถึงปัจจุบัน และกำหนดเวลาที่ทำการบันทึกได้ด้วยในที่นี้กำหนดเป็นข้อมูลที่ยังคงบันทึกอยู่ในวันที่ 16 กุมภาพันธ์ 2007 ความสามารถในการเลือกเวลาการบันทึกหมายความว่าเราสามารถดูข้อมูลในอดีตที่เคยบันทึกแม้ว่าในปัจจุบันข้อมูลอาจเปลี่ยนแปลงไปแล้ว



รูปที่ ก.1 ภาพตัวอย่างการเรียกดูข้อมูลจากแอพพลิเคชัน

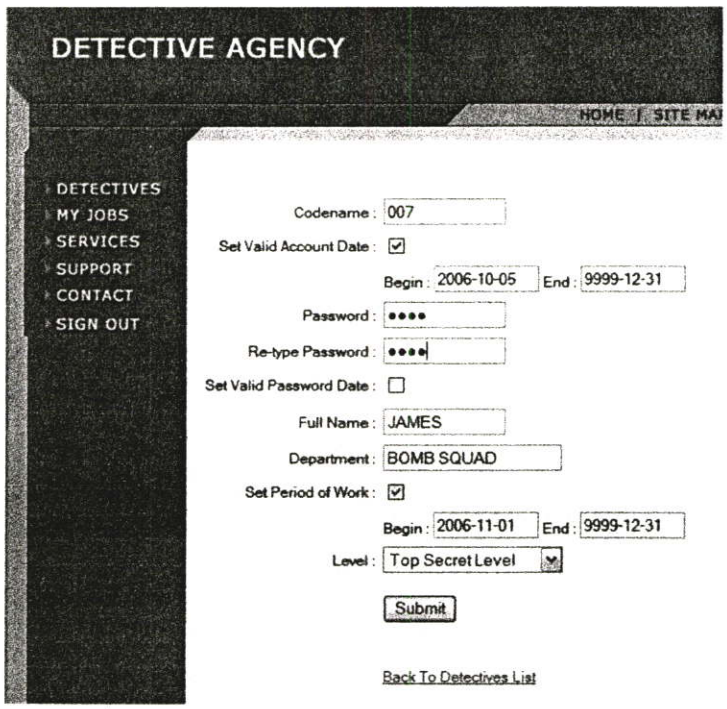
การแสดงผลในแอพพลิเคชันทำให้เข้าใจง่ายกว่าการมองจากตารางในฐานข้อมูล เนื่องจากบางข้อมูลในตารางจากฐานข้อมูลที่มีคาบเวลาต่อเนื่องกันอาจถูกแบ่งออกเป็นหลายแถว ดังในภาพสถานที่ที่ JACK ROBIN อยู่คือ ALPHINE COURSE ในฐานข้อมูลจริงแล้วเก็บเป็นสองแถว เนื่องจากข้อมูลการกระทำใน นั่นคือขณะที่ JACK ROBIN อยู่ที่ ALPHINE COURSE นั้นได้ทำกิจกรรมสองอย่างคือ PLAY GOLF และ RELEX ในแอพพลิเคชันสามารถรวมข้อมูลสถานที่นี้เข้าด้วยกันและแสดงเป็นคาบเวลาที่ต่อเนื่องกันได้

- **การปรับปรุงข้อมูล**

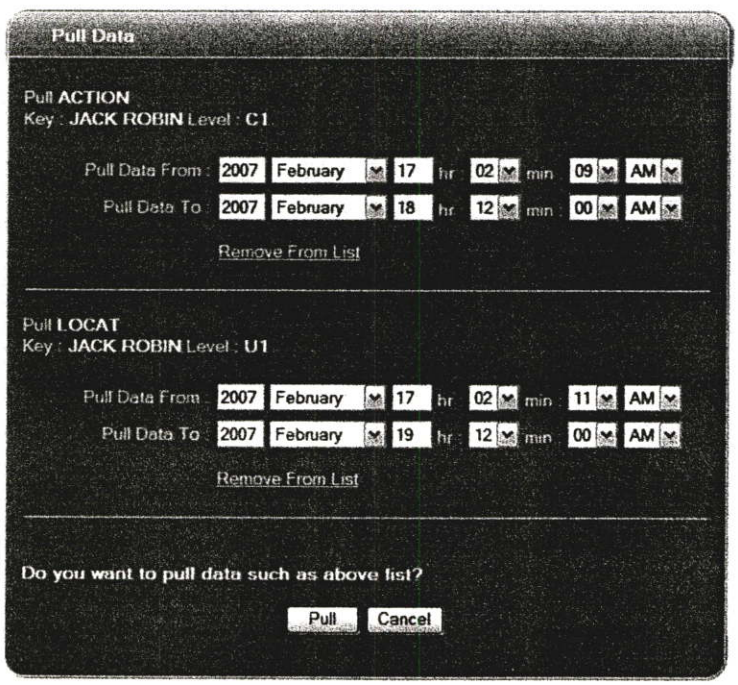
การปรับปรุงข้อมูลแอพพลิเคชันสำนักงานนักสืบ อยู่ภายใต้โมเดลเอ็มเอสบีดี ดังนั้นนอกเหนือจากคำสั่งเอสคิวแอลที่ใช้ในการแก้ไขข้อมูลได้แก่ INSERT, UPDATE และ DELETE แล้ว ยังมีคำสั่ง UPLEVEL สำหรับดึงข้อมูลในระดับล่างอีกด้วย ในรูปที่ ก.2 เป็นภาพตัวอย่างการเพิ่มข้อมูลของนักสืบผ่านแอพพลิเคชัน ซึ่งนอกจากกำหนดรายละเอียดชื่อและหน่วยงานของนักสืบแล้ว ยังสามารถกำหนดได้ว่าต้องการให้ข้อมูลแต่ละข้อมูลที่เพิ่มเข้าไปมีเวลากำกับด้วย ตัวอย่างเช่นได้เพิ่มนักสืบ 007 ลงไปในสำนักงานนักสืบโดยให้มีผลตั้งแต่วันที่ 5 ตุลาคม 2006 เป็นต้นไป นอกเหนือจากนั้นยังกำหนดวันที่จะเข้าไปทำงานในหน่วยงาน BOMB SQUAD ในวันที่ 1 พฤศจิกายน 2006 และกำหนดระดับชั้นของนักสืบให้อยู่ในระดับ TOP SECRETE อีกด้วย

ในตัวอย่างแอพพลิชันสำนักงานนักสืบได้กำหนดให้นักสืบติดตามบุคคล และให้รายงานการกระทำ และสถานที่ของผู้ถูกเฝ้าติดตาม การหาข้อมูลมานั้นอาจสั่งงานต่อไปให้ในผู้ใต้บังคับบัญชา และเมื่อผู้ใต้บังคับบัญชารายงานข้อมูลแล้ว ก็สามารถดึงข้อมูลมาเป็นข้อมูลรายงานในระดับชั้นของตนเองได้ ในเอสคิวแอลของโมเดลเอ็มอสบีดีเรียกคำสั่งนี้ว่า UPLEVEL

ในภาพที่ ก.3 แสดงการดึงข้อมูลจากระดับ C1 และ U2 ซึ่งเป็นระดับที่ต่ำกว่าผู้ที่ใช้งานแอพพลิเคชันโดยการดึงข้อมูลของเป้าหมาย JACK ROBIN และสามารถกำหนดได้ว่าต้องการดึงข้อมูลใดบ้างไม่ว่าจะเป็นการกระทำของเป้าหมาย (ACTION) หรือสถานที่ที่เป้าหมายอยู่ (LOCAT) และยังสามารถกำหนดเวลาที่ข้อมูลเป็นจริงได้ด้วย



รูปที่ ก.2 ภาพตัวอย่างการเพิ่มข้อมูลผ่านแอพพลิเคชัน



รูปที่ ก.3 ภาพตัวอย่างการดึงข้อมูลจากระดับอื่นผ่านแอพพลิเคชัน

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. Worawit Meanrach, Suphamit Chittayasothorn, "A Multilevel Secure Bitemporal Database System".International Conference on Advances in Intelligent Systems, 2004

AISTA 2004

International Conference

Advances in Intelligent Systems - Theory and Applications

In cooperation with the IEEE Computer Society

Conference Program | Abstract of Accepted Papers

Conference organised by:

SITEC

in collaboration with:

SES ASTRA

uni.lu

HENRI TUDOR

and with the support of:

fonds national de la

Luxembourg, November 15-18, 2004

ISBN 2-9599776-8-8

MO2.1: Security - Authentication & Services

A Secure Management for Service Gateways in Home Service Environment
Jong-Hoon Lee , Ho Jin Park
CDMA: Coordinated Distributed Multiple Denial of Service Attack Detection
Srinivas Mukkamala and Andrew H. Sung
Searching for the best nonlinearity of homogeneous Boolean functions by using simple genetic algorithm
Mohannad NAJJAR

MO2.2: Advanced Applications & Techniques

Pattern Recognition for Conductive Buried Tag Identification
Adel ZITOUNI, Larbi BEHEIM, and Fabien BELLOIR
Fuzzy linguistic rules identification system for wood quality
E. SCHMITT, C. MAZAUD, V. BOMBARDIER, P. LHOSTE, P. CHARPENTIER and R. VOGRIG
Simulation and testing of RF Transceiver suitable for wireless short range applications
ARSHAK, E. JAFER, and D. MCDONAGH

MO2.3: Security - Authentication & Services

A Multilevel Secure Bitemporal Database System
Worawit MEANRACH, and Supbamit CHITTAYASOTHORN
Securing Information on Compact Discs The Forbidden Access Approach
Jayesh SESHADRI and Nirgi P. BOTHRA,
An Image Watermarking Technique using Overlap-block
Somkiat WANGSIRIPITAK, Sorakit MORASILPIN, and Kritawan SIRIBOON
Secure and Intelligent Approach for Web applications
Mamoun Khadraoui and Beat Hirsbrunner

MO2.4: Markov Models

Sensitivity Properties of Markovian Models
Theodore Charitos and Linda C. van der Gaag
Cooperation through communication in decentralized Markov games
Raghav Aras, Alain Dutech, François Charpillet

A Multilevel Secure Bitemporal Database System

Worawit MEANRACH, and Suphamit CHITTAYASOTHORN

***Abstract*—Multilevel database security is a security management which provides clearance levels for users. It also provides classification labels for the data. Most multilevel secure database systems employ the mandatory access control as proposed by Bell and La Padula. Most research works apply the multilevel secure technique to conventional relational databases. This research project proposes an extension to the mandatory access control so that it suits our application requirements. Moreover, the support of bitemporal database features is also included. Both valid time and transaction time are added to the already complex multilevel secure relational database. Integrity rules for this new model are introduced and finally, an extension to the SQL language is proposed to be used on the data model.**

***Index Terms*—Databases, Dimensional Databases, Multilevel Security, Temporal Databases**

I. Introduction

Multilevel secure (MLS) database systems are database systems that provide several clearance security levels for users. The clearance levels are attached to each attribute and also attached to tuples. According to current literatures [3-11], users whose clearance level is equal or higher than the clearance level of the data items can read them. Users may write facts and the facts can then be accessed by users whose clearance level is higher than or equal to the one who writes. These read and write properties are according to the classic simple security properties proposed by Bell and La Padula [2]. At present, most MLS research works are based on the relational database model.

A temporal database is a database that supports some aspects of time, not counting user-defined time (an uninterpreted attribute domain of date and time). There are three major categories based on the relational model namely the valid time state table, transaction time state table and bitemporal tables. Conventional database applications nowadays assume that all the facts which are kept in the database are "'true"'. This assumption follows the 2-value logic: "'If found then true, if not found then false."' If a fact is updated, the new fact replaces the old one. For example, John Doe decided to change his first name by the end of 2002 and became Peter Doe. After the update took place, there is no John Doe in the database. Any new reports and queries that refer to the person, even the ones that refer back to the data before the end of 2002, refer to him as Peter Doe.

A kind of temporal database table called "'valid time state table"' has been introduced. Valid time start and valid time end are attached to the end of each tuple of a relation. The insertion of each tuple to such a relation requires a timestamp of the valid time of the facts on the tuple. This insertion activity is trivial. The hard parts are delete and update operations.

A temporal delete operation is not the same as a conventional delete one. It may involve marking the old tuple to be valid up to the deletion date, insert a new tuple and delete the existing tuple. The actual operations required depend on the relationship between period of validity of the fact to be deleted and the period of applicability of the deletion.

Temporal updates face similar challenges. One cannot replace an old fact by a new fact. The tuple that contains the new fact may be marked invalid and new tuples may need to be inserted. The actual operations required depend on the relationship between period of validity of the fact to be updated and the period of applicability of the update. Once the temporal delete and update operations are properly performed, queries on facts which are true over periods of time are possible.

A transaction time state table is intended to be used for facts whose state is not supposed to be changed according to time. For example, the height of a mountain is assumed not to be changed with respect to human life. However, with better measurement technology, different values of the height are recorded for each measurement time. The transaction time state table keeps facts and the time those facts are recorded (or believed to be true).

Bitemporal database tables keep facts whose states can be changed in time (valid time) and the time that they are recorded. [12-14]

This research paper presents an original approach to incorporate the concept of multilevel secure database on bitemporal databases. Only one previous work on MLS and temporal database is found [1] based only on valid time state table. A motivation of this work is our intention to create an MLS database system for an intelligence organization. There are many user levels range from Top to Public. Top officers not only would like to see facts entered by lower-level officers but also would like to enter facts to users with different clearance levels under them. Officers with different clearance levels may see different facts of the same real situation.

Manuscript received October 30, 2004

Worawit MEANRACH and Suphamit CHITTAYASOTHORN are with the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520 Thailand (e-mail: {s5061053, suphamit}@kmitl.ac.th).

Lower-level officers may enter facts which are valid over period of time, according to their belief, as reports to higher level officers. A higher level officer may choose to accept ("'up-level"') a fact from one of the facts reported to him on the same subject matter.

According to the application requirements, the MLS simple security properties have to be modified in order to allow facts entered by higher level users to be seen by lower level ones. Also, both valid time and transaction time are required in this MLS environment. In summary, this paper describes the logical data structure for an MLS bitemporal database. Integrity constraints and an SQL-like data manipulation language for the MLS bitemporal database tables are also presented.

II. Multilevel Secure Database

The major difference between a conventional database and a multilevel secure database is that the conventional one has only one level of data. Permitted users of the conventional database are allowed to have access to the same data. MLS database, on the other hand, has several security clearance levels for the users and security labels for data objects such as files, records or fields. Data classification levels can range from Top Secret (*TS*), Secret (*S*), Classified (*C*) and Unclassified (*U*). User's clearance levels are also classified this way. The hierarchy is $TS > S > C > U$.

Existing MLS database systems all follow two following basic rules [2]:

1) The Simple Security Property: A subject can read an object only when it has the same or higher clearance level than the classification level of the object.

*2) The *-Property*: A subject can write an object only when its clearance level is the same or lower than the classification level of the object.

SHIP	OBJ	DEST	TC
Enterprise	Spying	Mars	TS
Enterprise	Spying	Pluto	C
Enterprise	Shipping	Pluto	U

Fig. 1. A SOD classic MLS instance

SHIP	OBJ	DEST
Enterprise	Shipping	Pluto

Fig. 2. A SOD classic MLS instance at level *U*

In brief, a subject can read an object which is written by someone with a lower security level and can write/overwrite and object so that someone higher can read it as well. As an example, Fig. 1 shows a classic MLS table SOD (Starship, Objective, Destination). *TC* is the tuple class which represents the classification level of each tuple. A *TS*-subject can see all data since it has the highest clearance level. A *C*-subject can see only the last two tuples and a *U*-subject can see only the last tuple. In the case that the *U*-subject is the lowest level subject, it should see the tuple without the *TC* attribute as shown in Fig. 2. The MLS characteristic that allows different users to have different views on the same database is called polyinstantiation.

SHIP	SC	OBJ	OC	DEST	DC	TC
Enterprise	U	Spying	C	Mars	TS	TS
Enterprise	U	Spying	C	Pluto	U	C
Enterprise	U	Shipping	U	Pluto	U	U

Fig. 3. A SOD MLS instance

Classification levels can also be assigned to each attribute as well. As an example, Fig.3 shows security labels *SC*, *OC*, *DC* which are used to keep classification levels of *SHIP*, *OBJ* and *DEST* respectively. The tuples show objectives and destinations of the Enterprise as seen by users from different levels. The *TS* users see that the ship goes to Mars for a spying mission but the *U* users see it as going to Pluto for shipping and the *C* users see a spying mission at Pluto. Higher level users also see lower level tuples. Classification levels at the attributes show the origin of the facts. For example, the fact that the Enterprise goes spying is originated at the *C* level and believed by *TS*. Similarly, the fact that its destination is Pluto is originated by U and believed by *C*. The fact that it actually goes to Mars is both originated and believed by *TS*.

III. Multilevel Secure Bitemporal Data Model (MSBD)

A. Data Structure

The proposed MSBD model incorporates more labels to the MLS database. A tuple creator classification CC is introduced to each tuple to denote the originator of the tuple. Valid time and transaction time are also introduced to each attribute. The structure is as follows:

$$R(A_1, TT_1, VT_1, C_1, A_2, TT_2, VT_2, C_2, \ldots, A_n, TT_n, VT_n, C_n, TC, CC)$$

The motivation behind the introduction of the time labels is because we would like to see the fact which is believed to be true at present, facts that were believed to be true in the past and even those which are expected to be true in the future. The creator class is introduced because we would like to know the clearance level of the creator so that we will know if the tuple is generated from a higher level as a command or propagate up from a lower level as a report.

The MSBD model allows both commands and reports. Unlike other MLS systems that allow only reports up to the higher level, we introduce the concept that a higher-level subject should be able to write and allow lower-level ones to see. The newly introduced creator level *CC* identifies the creator of a tuple. For each tuple t $\in$ relation instance r,

if $t[TC] = t[CC]$ the tuple is a report one;

if $t[TC] < t[CC]$ the tuple is a command one.

Time granularity can be specified at the creation of the table. It could be month, week, day, minute or second or more detailed ones. The open-close format of the time interval is used through out the paper. The beginning symbol '[' denotes the beginning of the period and ')' denotes the end of the

SHIP	STT	SVT	SC	OBJ	OTT	OVT	OC	DEST	DTT	DVT	DC	TC	CC
Enterprise	[3,∞)	[4,∞)	U_1	Shipping	[3,∞)	[7,11)	U_1	Mars	[3,∞)	[7,11)	C	C	C
Enterprise	[3,∞)	[4,∞)	U_1	Shipping	[3,∞)	[7,11)	U_1	Mars	[3,∞)	[7,11)	U_1	U_1	U_1
Enterprise	[3,∞)	[4,∞)	U_2	Spying	[3,∞)	[8,11)	U_2	Uranus	[3,∞)	[9,12)	U_2	U_2	U_2
Voyager	[3,∞)	[4,∞)	U_1	Spying	[3,∞)	[7,11)	U_1	Pluto	[3,∞)	[7,12)	U_1	U_1	U_1

Fig. 4. A SOD MSBD instance

period but excluding the specified time. For example the period [1Jan04,1Jan05) means the period from and including 1Jan04 to 31Dec04. This kind of period format is used by the temporal database community because it is easy to check time period continuity and overlapping. In the case that the end time is '∞', it means the fact is a current fact.

B. User Hierarchy

In the original MLS database, there are different subject levels as previously described. In the MSBD model, we introduce multiple subject levels within the same clearance level. For example, instead of having only a U level, we allow several U levels U_1, U_2,...,U_n. This is a useful feature since a higher level subject may choose to write a fact to only some of the U-level subjects and may also want to see facts entered by only some of the U-level subjects. This is a discretionary access control since U_1 cannot see U_2's facts and vise versa. Fig. 5 show the user hierarchy of the MSBD model.

$\{TS\}$
↑
$\{S\}$
↑
$\{C\}$
↑
$\{U_1, U_2, \ldots U_n\}$

Fig. 5. A user hierarchy for the MSBD model

IV. Integrity Rules

The MSDB model has six integrity rules namely:

- Entity Integrity
- Polyinstantiation Integrity
- Belief-based Integrity
- Command and Report Integrity
- Foreign Key Integrity
- Referential Integrity

A. Entity Integrity

Let A_k be the apparent primary key of R such that instance r of R satisfies the entity integrity when each and every $t \in r$:

1) $A_i \in A_k \Rightarrow t[A_i] \neq \text{null}$
2) $A_i, A_j \in A_k \Rightarrow t[C_i] = t[C_j]$ (That is A_k belong to the same level)
3) $A_i, A_j \in A_k \Rightarrow t[TT_i] = t[TT_j] \wedge t[VT_i] = t[VT_j]$
4) $A_i \notin A_k \Rightarrow t[C_i] \geq t[CA_k]$ (when CA_k is the classification of the apparent primary key)
5) $A_i \notin A_k \Rightarrow t[TT_i] \subseteq t[TT_k] \wedge t[VT_i] \subseteq t[VT_k]$

B. Polyinstantiation Integrity

Instance r of R satisfies polyinstantiation integrity when $1 \leq i \leq n$, where A_1 is the apparent primary key,
1) $A_1, TT_1, TT_2, \ldots, TT_n, TC, CC \rightarrow C_i$
2) $A_1, C_1, C_i, TT_1, TT_2, \ldots, TT_n, CC \rightarrow A_i$

The actual primary key of the relation is $A_1, C_1, C_2, \ldots, C_n, TC, CC$ can be obtained as follow:

$A_1, TT_1, TT_2, \ldots, TT_n, TC, CC \rightarrow C_1$;
$A_1, C_1, TT_1, TT_2, \ldots, TT_n, TC, CC \rightarrow A_i, C_i$

C. Belief-based Integrity

Belief-based integrity is a key property in MLS database. It allows consistent propagation of facts from lover levels to higher levels. The belief-based integrity is defined as follow:

Instance r of a multilevel relation R satisfy belief-based integrity when $t \in r$ and $1 \leq i \leq n$, if $t[A_i] \neq \text{null} \wedge t[C_i] < t[TC]$ and there exist $t' \in r$ such that $t'[A_1, C_1] = t[A_1, C_1] \wedge t'[TC] = t'[C_i] = t[C_i] \wedge t'[CC] = t[CC] \wedge t'[A_i] = t[A_i] \wedge t[TT_1] \subseteq t'[TT_1] \wedge t[TT_i] \subseteq t'[TT_i]$

This integrity rule is based on the concept that a c-tuple comprises all facts which are accepted by a c-subject which may not be the creator of the facts.

D. Command and Report Integrity

A relation R satisfies Command and Report integrity if a tuple $t \in$ relation instance r has $t[VT_i] = [t_{vstarti}, t_{vstopi})$, $t[TT_i] = [t_{tstarti}, t_{tstopi})$ that follows the conditions:
1) $\forall t_{vstopi} < t_{tstarti} \Rightarrow t[TC] = t[CC]$
2) $\forall t_{vstopi} \geq t_{tstarti} \Rightarrow t[TC] = t[CC] \vee t[TC] < t[CC]$
3) There is no other periods apart from the ones specified by the above two rules.

The first rule allows the valid time period of a fact can be the time in the past only when the fact is a reported one. The second one allows reports and commands to be in the present time and the future. The last one enforces that the command cannot be given to the time in the past.

E. Foreign Key Integrity

This one enforces that all foreign key value must have the same level:

1) All $A_i, A_j \in FK$ must have the same level.
2) $A_i, A_j \in FK \Rightarrow t[TT_i] = t[TT_j] \wedge t[VT_i] = t[VT_j]$

F. Referential Integrity

In the case that a foreign key value is not null, let r_1 be an instance of a referencing relation and r_2 be an instance of a referenced relation and $t_1 \in r_1$ and $t_2 \in r_2$ The referential integrity is satisfied if
1) $t_1[FK] = t_2[AK]$
2) $t_1[TC] = t_2[TC]$
3) $t_1[C_{FK}] \geq t_2[C_{AK}]$
4) $t_1[TT_{FK}] \subseteq t_2[TT_{AK}]$

The second rule enforces that both the primary and secondary key must belong to the same tuple level. The third one $t_1[C_{FK}] \geq t_2[C_{AK}]$ allows the value to be propagated from a lower level and the last one allows a value to be borrowed up from a lower level.

V. AN MSBD SQL LANGUAGE

An extension to the SQL language is proposed. Several keywords are introduced to support the new concepts. The temporal part is obtained from the TSQL language [12]. MLS keywords on levels are added as follow:

- *me* is the level of the current user.
- *anyone* is every levels that can see the data.
- *Below(l)* is all level which is lower than *l* where *l* is the level of the current user.
- *Above(l)* is all level which is higher than *l* where *l* is the level of the current user.

A. INSERT command:

[VALIDTIME PERIODS (A_i[,A_j],...) VALUES (a_i[,a_j],...)
INSERT INTO $R(A_i[,A_j]...)$
VALUES (a_i[, a_j], ...)
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

The INSERT command may have a valid time specified to enforce that the facts which are to be inserted are true during the specified period. Levels play important roles here since they determine the intension of the insert. If the level in the LEVEL clause is the current level, then the insert is a report activity. Some higher level users are expected to read and use the data. In the case that the levels are lower than the current user's level, then the insert is a command to some lower level ones.

B. UPLEVEL command:

[VALIDTIME PERIODS (A_i[,A_j],...) VALUES (a_i[,a_j],...)
UPLEVEL *R*
GET A_i FROM c_i[, A_j FROM c_j] ...
[WHERE *p*]

When higher-level subjects would like to use facts entered by lower-level users by bringing the facts to be their own facts, an UPLEVEL statement can be used. This command brings the required data from a specified lower level up to the current user's level. The uplevel activity must follow the belief-based integrity rule.

After the UPLEVEL statement is applied, the system may insert new rows or replace existing rows with new values. As an example, an S-subject has seen and verified that the facts entered by U1 are true. It then uplevels the facts to be its own facts on the 22nd of a month. This can be done by using the command:

```
VALIDTIME PERIODS (SHIP) VALUES ([14,20))
UPLEVEL SOD
GET OBJ FROM U1, DEST FROM U1
WHERE SHIP = 'Enterprise'
```

A new tuple will be created. The new S-level tuple will have the same apparent primary key and level as the U1 subject between the day 14 to 20.

C. UPDATE command:

[VALIDTIME PERIODS (A_i[,A_j],...) VALUES (a_i[,a_j],...)
UPDATE R
SET $A_i = s_i$[, $A_j = s_j$] ...
WHERE *p*
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

The UPDATE command can be applied by a current-level subject as a report purpose. It can be used as a command on lower levels. The command can be applied to data which has the same classification level as the clearance level of the user. As an example, suppose a U1 subject knows that during the day 17 and 18 the Enterprise will come to Earth, the corresponding UPDATE statement will be as follows:

```
VALIDTIME PERIODS (DEST) VALUES ([17,18))
UPDATE SOD
SET DEST = 'Earth'
WHERE SHIP = 'Enterprise'
```

D. DELETE Command:

[VALIDTIME PERIODS (A_i[,A_j],...) VALUES (a_i[,a_j],...)
DELETE
FROM *R*
[WHERE *p*]
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

The subject who would like to delete tuples must have a clearance level equal to the classification level of the tuples. Of course, tuples which satisfy the WHERE condition will be deleted. Other tuples at higher level that use values from the deleted tuples will also be deleted or have attribute values set to null.

Note that in all of the above cases, transaction time will be automatically recorded after the operations are completed. Also, INSERT, UPDATE and DELETE activities are temporal activities. UPDATE and DELETE, in particular, are not always physically deleted. Some of the them may physically have the Valid time marked as no longer valid only.

SHIP	STT	SVT	SC	OBJ	OTT	OVT	OC	DEST	DTT	DVT	DC	TC	CC
Enterprise	[22,∞)	[14,∞)	U_1	Exploration	[22,∞)	[15,17)	U_1	Uranus	[22,23)	[16,18)	U_1	S	S
Enterprise	[22,∞)	[14,∞)	U_1	Exploration	[22,∞)	[15,17)	U_1	Uranus	[23,∞)	[16,17)	U_1	S	S
Enterprise	[22,∞)	[14,∞)	U_1	Exploration	[22,∞)	[15,17)	U_1	Earth	[23,25)	[17,18)	U_1	S	S
Enterprise	[20,∞)	[14,∞)	U_1	Exploration	[20,∞)	[15,17)	U_1	Uranus	[20,23)	[16,18)	U_1	U_1	U_1
Enterprise	[20,∞)	[14,∞)	U_1	Exploration	[20,∞)	[15,17)	U_1	Uranus	[23,∞)	[16,17)	U_1	U_1	U_1
Enterprise	[20,∞)	[14,∞)	U_1	Exploration	[20,∞)	[15,17)	U_1	Earth	[23,∞)	[17,18)	U_1	U_1	U_1

Fig. 6. The query result of a temporal query

Examples of these temporal operations can be found in [12].

E. SELECT statement:

[NONSEQUENCED VALIDTIME AND
TRANSACTIONTIME]
SELECT A_1[, A_2...]
FROM R
[WHERE p]
[LEVELS l_i[,l_j]... EXCEPT l_n[,l_m]...]

The SELECT statement has both the temporal and multilevel secure extension. The detail of its functionality would require a separate paper. However, we can give an example here. Suppose an S-subject would like to find a mission record on the day 22. This is a "sequenced" valid time and "nonsequenced" transaction time. The SELECT statement is as follow:

VALIDTIME AND NONSEQUENCED
TRANSACTIONTIME
SELECT *
FROM SOD
WHERE TRANSACTIONTIME(SOD) OVERLAPS TIME
'22'
LEVEL anyone

The query result is as shown in Fig 6.

VI. Conclusion

This paper presents the combination of Multilevel Secure (MLS) and Temporal Database concepts. This new concept enables users and data to be classified. Data are introduced to the system both for report purpose and for command purpose. Valid time and transaction time are also introduced so that one could request information which are true in the past, present and future together with the record time. Integrity rules and an extension to the SQL language is also described.

References

[1] Pissinou, Niki, Kia Makki, and E. K. Park. "Towards a framework for integrating secure models and temporal databases, " *Third International Conference on Information and Knowledge Management*, 1994, pp 280-287.

[2] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation," *MTR-2997*, MITRE, 1975.

[3] Denning D. E. (1988), "The Sea View Security Model," *IEEE Symposium on Security and Privacy*, Oakland, Ca, April, 218-233.

[4] Jajodia S. and Sandhu R. (1991), "Toward a Multilevel Secure Relational Data Model,"*ACM SIGMOD*, Denver, Colorado, May, ACM, New York, pp. 50-59.

[5] Sandhu R., and Chen F., (1995), "The Semantics and Expressive Power of the MLR Data Model," *IEEE Symposium on Security and Privacy*, Oakland, Ca, May, pp. 128-142.

[6] Smith K. and M. Winslett. "Entity modeling in the MLS relational model, " *Eighteenth VLDB*, 1992, pp 199-210.

[7] Jukic N, Vrbsky SV. "Asserting Beliefs in MLS Relational Models," *SIGMOD Record*, Vol. 26,No.3, 1997, pp.30-35.

[8] Jukic N., Vrbsky S., Parrish A., Dixon B, Jukic B. "A Belief-Consistent Multilevel Secure Relational Data Model," *Information Systems*, Vol. 24, No. 5, 1999, pp. 377-402.

[9] Pranjic M., Jukic N., Fertalj K. "Implementing Belief-Consistent Multilevel Secure Relational Data Model: Issues and Solutions," *25th Int. Conf. Information Technology Interfaces*, 2003.

[10] Pranjic M., Fertalj K., Jukic N. "Importance of Semantics in MLS Database Models," *24th Inf. Conf. Inoformation Technology Interfaces*, 2002

[11] Jukic N, Nestorov S, Vrbsky S. "Closing the Key Loophole in MLS Database," *SIGMOD Record*, Vol.32, No2, 2003.

[12] R. T. Snodgrass. "Managing Temporal Data A Five-Part Series. TimeCenter Technical Report TR-28," University of Arizona, 1998.

[13] Gadia, Shashi K. "Applicability of Temporal Data Models to Query Multilevel Security Databases: A Case Study," *Temporal Databases – Research and Practice LNCS 1399*, 1998, pp. 238-256.

[14] Gadia, Shashi K. "A homogenous relational model and query languages for temporal databases," *ACM Transactions on Database Systems*, December 1988, pp. 13(4):418-448.

[15] Tsz Shing Cheng and Shashi K. Gadia. "An algebra for belief persistence in multi- level security" An unpublished manuscript, available as Tech. Report TR97-18, Computer Science Department, Iowa State University, Ames, 1997.

ประวัติผู้เขียน

นายวรวิทย์ หมื่นราช เกิดเมื่อวันที่ 16 พฤศจิกายน พ.ศ.2523 ที่จังหวัดสุราษฎร์ธานี สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้าคอมพิวเตอร์ จากภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าฯ พระนครเหนือ ในปีการศึกษา 2544 และเข้าศึกษาต่อในระดับปริญญาโท หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2545 และมีความสนใจด้านระบบฐานข้อมูล