เครื่องมือพัฒนาแอพพลิเคชันฐานข้อมูลเชิงเวลา

Temporal Database Application Development Tool

โดย

ศุภฤกษ์ ศิวะมาศ

รหัส 44067001

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2547

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	เครื่องมือพัฒนาแอพพลิเคชันฐานข้อมูลเชิงเวลา
นักศึกษา	นายศุภฤกษ์ ศิวะมาศ
อาจารย์ที่ปรึกษา	รศ. ดร. ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

แนวความคิดในเรื่องฐานข้อมูลเชิงเวลาได้เกิดขึ้นมาเป็นเวลานาน แต่ยังไม่มีการนำมาใช้งานโดยแพร่หลาย แม้ว่า จะมีประโยชน์มาก สาเหตุหนึ่งคือ แนวความคิดในเรื่องฐานข้อมูลเชิงเวลานั้นยังยากต่อความเข้าใจ ทำให้โครงการที่พยายามใช้การออกแบบฐานข้อมูลเชิงเวลาต้องประสพปัญหาหลายประการเช่น สมาชิกในทีมพัฒนาไม่สามารถปรับตัวเข้ากับแนวความคิดใหม่ ต้องใช้เวลาเพิ่มกับการฝึกอบรม สูญเสียผลิตภาพ ผู้ใช้ไม่สามารถสืบค้นฐานข้อมูลได้โดยตรง และอื่นๆ

งานวิจัยฉบับนี้ มีจุดประสงค์ที่จะหาวิธีการให้ฐานข้อมูลเชิงเวลาสามารถอยู่ร่วมกับฐานข้อมูลปกติได้ สามารถนำไปใช้โดยผู้ใช้งาน ผู้ดูแลระบบ และนักพัฒนา โดยไม่จำเป็นต้องเข้าใจหลักการฐานข้อมูลเชิงเวลาอย่างละเอียด โดยการสร้างเครื่องมือช่วยในการต่อขยายฐานข้อมูลปกติ ให้กลายเป็นฐานข้อมูลเชิงเวลาโดยกระทบต่อการทำงานของแอพพลิเคชันที่ใช้ฐานข้อมูลนั้นน้อยที่สุดหรือไม่กระทบเลย

Title	Temporal Database Application Development Tool
Student	Mr. Suparurk Sivamard
Advisor	Assoc. Prof. Dr. Suphamit Chittayasothorn
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2004

ABSTRACT

Temporal database concept has been around for a long time. However, the implementation of temporal database in database application is limited. One reason is that the temporal database concept is difficult to understand. Therefore, the projects that try to implement it face many problems such as team members can not tune in with the new concept, delays due to training, loss of productivity, power users can not directly query database, and so on.

The objective of this research is to find a method to make temporal database concept able to coexist with normal database. Use of temporal-enabled database should not require total understanding of temporal database concept; therefore, users, administrators, or team members can utilize the database without extensive training. A tool is developed so that to extend temporal functionalities to normal database is not difficult and causes least effects to the existing applications using the normal database.

กิตติกรรมประกาศ

โครงการนี้ไม่อาจจะเริ่มต้นและสำเร็จลงได้ หากขาด อ.ศุภมิตร จิตตะยโศธร ผู้จุดประกายแนวความคิดเรื่องฐานข้อมูลเชิงเวลา ผู้เป็นอาจารย์ที่ปรึกษา ให้คำแนะนำด้านวิชาการ ช่วยให้การสนับสนุน และ ช่วยนำทางเวลาที่ข้าพเจ้าพบทางตัน ข้าพเจ้าขอขอบคุณ อ.ศุภมิตร มา ณ ที่นี้

ขอขอบคุณ คณาจารย์คณะเทคโนโลยีสารสนเทศ ที่ได้ให้ความรู้แก่ข้าพเจ้าตลอด 4 ปีที่ศึกษาอยู่ที่คณะ และสามารถนำความรู้เหล่านั้นมาใช้ในการพัฒนาโครงการนี้

ข้าพเจ้าหวังว่า งานวิจัยฉบับนี้ จะมีส่วนช่วยให้ผู้สนใจฐานข้อมูลเชิงเวลา สามารถนำความรู้ที่นำเสนอไปใช้ได้จริง หากงานวิจัยนี้มีข้อผิดพลาดประการใด คือความผิดของข้าพเจ้าแต่เพียงผู้เดียว

ศุภฤกษ์ ศิวะมาศ

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่

สารบัญภาพ

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมา

แนวความคิดเรื่องฐานข้อมูลเชิงเวลาได้มีการศึกษาและนำเสนอมาเป็นเวลานาน ถึงแม้ปกติเราจะสามารถบันทึกข้อมูลเวลาไว้ในฐานข้อมูลอยู่แล้ว แต่การใช้ข้อมูลเวลาเหล่านั้นยังได้ไม่มากเท่าที่ต้องการ เช่น หากเราต้องการสืบค้นข้อมูลในอดีต สืบค้นแบบตามรอยข้อมูลในอดีต หรือ หาความสัมพันธ์ของข้อมูลในเชิงเวลา เรามักพบว่า ฐานข้อมูลปกติที่เราเก็บไว้ไม่เพียงพอต่อการสืบค้นลักษณะนี้

การศึกษาฐานข้อมูลเชิงเวลาได้นำเสนอแนวทางการนำฐานข้อมูลเชิงเวลามาใช้ แต่กลับไม่แพร่หลายเท่าที่ควร เพราะถึงแม้การใช้ฐานข้อมูลเชิงเวลาจะเกิดประโยชน์ แต่การนำมาใช้กลับทำให้ผู้นำมาใช้ประสบปัญหาหลายประการ

1.2 ปัญหาในการนำฐานข้อมูลเชิงเวลามาใช้

ในการนำแนวความคิดเรื่องฐานข้อมูลเชิงเวลามาใช้กับแอพพลิเคชันปกติ ประสบปัญหาใหญ่ๆ ดังต่อไปนี้

1.2.1 ยากต่อความเข้าใจ

เวลา เป็นเรื่องที่ทุกคนทราบและคุ้นเคยกันอย่างดี แต่หากต้องอธิบายความสัมพันธ์ระหว่างข้อมูลในเชิงเวลา จะพบว่า เป็นเรื่องที่เข้าใจได้ยากมาก ประกอบกับผู้เกี่ยวข้องในโครงการพัฒนาระบบสารสนเทศในปัจจุบันมีความคุ้ยเคยกับฐานข้อมูลปกติ (ที่มักบันทึกข้อมูล ณ ปัจจุบัน) ดังนั้นในการนำฐานข้อมูลเชิงเวลาไปใช้จริง ผู้เกี่ยวข้องจะพบว่า ลักษณะของฐานข้อมูลเชิงเวลา แตกต่างกับฐานข้อมูลปกติมาก และไม่เข้าใจหากไม่ได้รับคำอธิบายที่ละเอียดและเพียงพอ

ตารางที่ 1.1 ผู้เกี่ยวข้องกับการพัฒนาระบบสารสนเทศ

ผู้เกี่ยวข้อง	ความเกี่ยวข้อง
นักออกแบบฐานข้อมูล	ต้องเข้าใจแนวความคิดฐานข้อมูลเชิงเวลาอย่างถ่องแท้จึงจะสามารถออกแบบฐานข้อมูลเชิงเวลาได้
นักพัฒนาซอฟต์แวร์	ต้องเข้าใจแนวความคิดฐานข้อมูลเชิงเวลาอย่างถ่องแท้ จึงสามารถเขียนคำสั่งในการสืบค้นข้อมูล หรือ ปรับปรุงข้อมูลในฐานข้อมูลเชิงเวลาได้
ผู้ใช้ที่มีความรู้ในการสืบค้นข้อมูล	ปกติ ผู้ใช้ที่มีความรู้จะเข้าไปสืบค้นสารสนเทศจากฐานข้อมูลโดยตรง แต่หากเป็นฐานข้อมูลเชิงเวลา ผู้ใช้ต้องเข้าใจแนวความคิดฐานข้อมูลเชิงเวลาก่อน จึงจะสามารถเข้าไปใช้งานได้ และการสืบค้นฐานข้อมูลเชิงเวลาโดยใช้ภาษา SQL ปกตินั้น ซับซ้อนพอสมควร อาจเกินระดับของผู้ใช้

1.2.2 เปลี่ยนฐานข้อมูลปกติที่มีอยู่แล้วได้ยาก

ระบบสารสนเทศที่ใช้ในปัจจุบัน หากต้องการเปลี่ยนฐานข้อมูลปกติให้เป็นฐานข้อมูลเชิงเวลา จำเป็นต้องมีการออกแบบฐานข้อมูลใหม่ ปรับปรุงโค้ดในโปรแกรม อาจต้องถึงขนาดเขียนแอพพลิเคชันใหม่ก็เป็นไปได้

1.2.3 ขาดแคลนคำสั่งที่ใช้ในการสืบค้นเชิงเวลา

ภาษา SQL มาตรฐานมีคำสั่งที่ใช้ในการประมวลผลข้อมูลเชิงเวลาน้อยมาก และเป็นเพียงคำสั่งสำหรับ User-defined Time (อธิบายในบทที่ 2) เท่านั้น

นักพัฒนาจำเป็นต้องมีความเข้าใจฐานข้อมูลเชิงเวลา และเขียนคำสั่งภาษา SQL ที่ถูกต้องเพื่อสามารถดึงข้อมูลที่ต้องการจากฐานข้อมูลเชิงเวลา ซึ่งตามปกติจะพบว่า คำสั่งที่ใช้ในการดึงข้อมูลจากฐานข้อมูลเชิงเวลามักจะซับซ้อนและยาว

นอกจากนี้ ยังมีผู้ใช้งานระดับสูงที่สามารถดึงข้อมูลจากฐานข้อมูลโดยตรงเพื่อไปทำรายงานหรือทำการวิเคราะห์ ผู้ใช้งานเหล่านี้มักไม่ใช้คนที่มีความรู้ด้านฐานข้อมูลโดยตรง เพียงมีความเข้าใจขั้นพื้นฐานเพียงพอที่จะใช้ดึงข้อมูลได้เท่านั้น เป็นเรื่องยากที่ต้องให้ผู้ใช้เหล่านี้เรียนรู้ฐานข้อมูลเชิงเวลา หรือ วิธีการเขียนคำสั่ง SQL ที่ซับซ้อนเพื่อดึงข้อมูล

1.2.4 สมรรถนะในการทำงาน

ระบบจัดการฐานข้อมูลปัจจุบัน ไม่ได้ออกแบบมาเพื่อประมวลผลคำสั่งเชิงเวลา สมรรถนะในการสืบค้นข้อมูลจะลดลงแม้ว่าเป็นเพียงการสืบค้นปกติ

1.2.5 ไม่มี Object ในการจัดการ Constraint เชิงเวลา

การควบคุม Constraint ต่างๆ เช่น Uniqueness หรือ Foreign Key Constraint ผู้พัฒนาจำเป็นต้องเขียนโปรแกรมควบคุมในแอพพลิเคชัน หรือ เขียนคำสั่ง SQL ในรูปแบบ Stored Procedure หรือ Trigger ซึ่งการควบคุม Constraint เหล่านี้จำเป็นต้องใช้ชุดคำสั่งที่ซับซ้อน

1.2.6 ขอบเขตของระบบไม่ได้ต้องการฐานข้อมูลเชิงเวลาทั้งหมด

ระบบสารสนเทศอาจต้องการข้อมูลเชิงเวลาบางส่วนเท่านั้น ในการออกแบบครั้งแรกสามารถออกแบบรองรับได้ แต่หากมีความต้องการใหม่เกิดเพิ่มในภายหลัง ต้องมีการปรับเปลี่ยนฐานข้อมูลและแอพพลิเคชันเพื่อรองรับ

การปรับเปลี่ยนเพื่อความต้องการข้อมูลเชิงเวลาทำได้ไม่ง่ายนักเพราะต้องเปลี่ยนโครงสร้างตารางข้อมูลนั้น

1.2.7 ฐานข้อมูลเชิงเวลาเหมาะกับฐานข้อมูลเชิงวัตถุ

งานวิจัยหลายชิ้นสรุปว่า ระบบฐานข้อมูลที่เหมาะสมที่สุดที่จะพัฒนาฐานข้อมูลเชิงเวลาคือระบบฐานข้อมูลเชิงวัตถุ (Steiner. 1998 : 4)

ในปัจจุบัน เริ่มมีระบบจัดการฐานข้อมูลเชิงวัตถุแบบการค้าออกมา เช่น Cache DBMS หรือ Oracle DBMS (Oracle รองรับฐานข้อมูลเชิงวัตถุได้ จึงเรียกตนเองเป็น Object-Relational DBMS) แต่แอพพลิเคชันส่วนใหญ่ยังใช้ฐานข้อมูลเชิงสัมพันธ์อยู่ ดังนั้น หากต้องการใช้ฐานข้อมูลเชิงเวลาให้มีประสิทธิภาพโดยการเปลี่ยนไปใช้ฐานข้อมูลเชิงวัตถุ ก็เป็นแนวทางที่ต้องการความเปลี่ยนแปลงมาก ยากต่อการนำไปปฏิบัติโดยทั่วไป

1.3 วัตถุประสงค์

งานวิจัยฉบับนี้มีวัตถุประสงค์ในการหาแนวทางประยุกต์ใช้ฐานข้อมูลเชิงเวลาโดยเน้นการแก้หรือลดปัญหาดังกล่าว เพื่อให้สามารถประยุกต์ใช้ฐานข้อมูลเชิงเวลาได้

เมื่อได้แนวทางแล้ว ได้ทำการสร้างเครื่องมือเพื่อใช้ในการประยุกต์ฐานข้อมูลเชิงเวลาเข้าในฐานข้อมูลปกติ

1.4 แนวทางแก้ปัญหา

โดยทั่วไป การใช้งานฐานข้อมูลเชิงเวลาจะทำตั้งแต่เริ่มต้น คือ ต้องเป็นฐานข้อมูลเชิงเวลาอย่างเดียวเท่านั้น ทฤษฎีและงานวิจัยเกือบทั้งหมดจะมุ่งประเด็นไปว่า ทำฐานข้อมูลเชิงเวลาอย่างไรจึงจะสมบูรณ์และครบถ้วน

เนื่องจากปัญหาในทางปฏิบัติดังกล่าวข้างต้น จึงมีความคิดว่า น่าจะทำการประยุกต์ใช้ฐานข้อมูลเชิงเวลา โดยการสร้างตารางเชิงเวลาเพิ่มออกจากตารางข้อมูลปกติอยู่ในฐานข้อมูลเดิม โดยผู้ใช้ฐานข้อมูลปกติ และ แอพพลิเคชันที่มีอยู่เดิมก็ยังสามารถทำงานได้เหมือนเดิม ส่วนความสามารถทางเชิงเวลาที่เพิ่มขึ้นนั้น ให้เขียนเพิ่มในแอพพลิเคชันเชิงเวลา โดยวิธีนี้ ตารางข้อมูลเดิมจะกลายเป็น Current Version ของตารางข้อมูลเชิงเวลา น่าจะสามารถแก้ปัญหาในทางปฏิบัตินั้นได้

อย่างไรก็ตาม การที่มีตารางข้อมูล 2 เวอร์ชันอยู่ในฐานข้อมูล คือ Temporal Version และ Current Version ก่อให้เกิดปัญหาเรื่อง Consistency เพราะทั้งสองเวอร์ชันต้องไม่ขัดแย้งกัน ซึ่งปัญหาในข้อนี้ อาศัยเทคโนโลยีแบบกระจายในปัจจุบัน สามารถป้องกันและแก้ไขได้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- ได้วิธีการประยุกต์ใช้ฐานข้อมูลเชิงเวลาที่แก้ปัญหาการนำมาใช้ข้างต้น 7 ประการ
- ได้เครื่องมือช่วยในการประยุกต์ใช้ฐานข้อมูลเชิงเวลา
- สามารถสร้างฐานข้อมูลเชิงเวลาจากฐานข้อมูลปกติที่มีอยู่แล้ว โดยมีผลกระทบกับแอพพลิเคชันที่มีอยู่แล้วให้น้อยที่สุด

1.6 ขั้นตอนในการพัฒนา

1. ศึกษาแนวความคิดของฐานข้อมูลเชิงเวลา
2. วิเคราะห์ปัญหาการนำฐานข้อมูลเชิงเวลามาใช้ในแอพพลิเคชันฐานข้อมูล
3. ค้นหาแนวทางประยุกต์ใช้โดยคำนึงถึงปัญหาการนำมาใช้เป็นหลัก
4. สร้างเครื่องมือช่วยในการประยุกต์ใช้ฐานข้อมูลเชิงเวลาเพื่อรองรับผลการวิจัย
5. ทดสอบเครื่องมือกับแอพพลิเคชันฐานข้อมูลปกติ

1.7 รายละเอียดของแต่ละบท

บทที่ 2 อธิบายถึงทฤษฎีพื้นฐานของฐานข้อมูลเชิงเวลาในขอบเขตของงานวิจัย ในบทที่ 3 ทำการวิเคราะห์ปัญหา 7 ประการดังที่กล่าวไว้ในหัวข้อ 1.2 และนำเสนอแนวทางประยุกต์ใช้ฐานข้อมูลเชิงเวลา

การวิเคราะห์ความต้องการของซอฟต์แวร์เครื่องมือช่วยในการประยุกต์ใช้ฐานข้อมูลเชิงเวลาจะนำเสนอในบทที่ 4 และการออกแบบซอฟต์แวร์เครื่องมือในบทที่ 5

บทที่ 6 แสดงหน้าตาของซอฟต์แวร์เครื่องมือและการใช้งาน และสรุปผลงานวิจัยและแนวทางพัฒนาในอนาคตในบทที่ 7

1.8 เครื่องมือและซอฟต์แวร์ที่ใช้ในการพัฒนา

เครื่องมือและซอฟต์แวร์ต่างๆที่ใช้ในการพัฒนามีดังนี้

Microsoft SQL Server 2000 เป็นระบบจัดการฐานข้อมูล ทำงานบนระบบปฏิบัติการ Microsoft Windows XP Professional ระหว่างการพัฒนา และบน Microsoft Windows Server 2003 ในการทดสอบ

Transact-SQL เป็นภาษา SQL ที่ใช้ทั้งหมดในงานวิจัยนี้ เนื่องจากเป็นภาษาที่ใช้ใน Microsoft SQL Server 2000

Microsoft Visual Studio .NET 2003 เป็นเครื่องมือใช้ในการพัฒนาแอพพลิเคชัน การพัฒนาซอฟต์แวร์คอมโปแนนท์ใช้ภาษา C# ซึ่งเป็นภาษาหลักใน .NET Framework

Borland Together .NET เป็นเครื่องมือที่ใช้ในการออกแบบ UML

Microsoft Visio for Enterprise Architects เป็นเครื่องมือสำหรับออกแบบฐานข้อมูล และทำ Forward Engineering โดยการสร้างสคริปต์ภาษา SQL เพื่อสร้างตารางข้อมูลและออปเจกต์ต่างๆในฐานข้อมูล Microsoft SQL Server

บทที่ 2
ฐานข้อมูลเชิงเวลา

2.1 ภาพรวมของฐานข้อมูลเชิงเวลา

ฐานข้อมูลเชิงเวลาประกอบด้วยแนวความคิด 3 แนวซึ่งพบได้ในทุกแอพพลิเคชันที่เกี่ยวข้องกับเวลา คือ ประเภทข้อมูลเชิงเวลา ชนิดของเวลา และ คำสั่งเชิงเวลา (Snodgrasss. 2000 : 3 ; Snodgrass. 1998)

2.1.1 ประเภทข้อมูลเชิงเวลา

ข้อมูลเชิงเวลาแบ่งออกได้เป็น 3 ประเภทคือ

- จุดเวลา (Instant) คือ ณ เวลาหนึ่ง เช่น วันที่ 1 มกราคม 2548 เวลา 12:00 น.
- ช่วงห่างเวลา (Interval) หรือเรียกว่าเป็นความยาว ความนานของเวลา เช่น นาน 3 วัน หรือ ใช้เวลา 10 เดือน
- ช่วงเวลา (Period) เป็นระยะเวลาที่มีการกำหนดจุดเริ่มต้นและจุดสิ้นสุด เช่น ระหว่างวันที่ 1 มกราคม 2548 ถึงวันที่ 15 มกราคม 2548

SQL-92 สนับสนุนข้อมูลเชิงเวลาประเภทจุดเวลาและช่วงห่างเวลา แต่ยังไม่สนับสนุนข้อมูลประเภทช่วงเวลา

2.1.2 ชนิดของเวลา

ข้อมูลเวลาในฐานข้อมูล สามารถแบ่งออกได้เป็น 3 ชนิดคือ

- เวลากำหนด (User-defined Time) เป็นข้อมูลเวลาที่ถูกนิยามความหมายไว้เอง โดยถือเป็นข้อมูลหนึ่งๆที่เทียบเท่าข้อมูลอื่นทั่วไป ไม่ได้นำมาใช้ในฐานข้อมูลเชิงเวลา เช่น วันเกิด วันที่เอกสาร
- เวลาเป็นจริง (Valid Time) คือการประทับตราเวลา (Timestamp) ไปบนข้อมูลว่า นี่คือช่วงเวลาที่ข้อมูลนั้น เป็นจริง
- เวลารับรู้ (Transaction Time) คือการประทับตราเวลาไปบนข้อมูลว่า นี่คือช่วงเวลาที่ข้อมูลนั้นถูกรับรู้ หรืออีกแง่หนึ่งคือ ถูกเก็บไว้ในฐานข้อมูล

ในตารางเวลาหนึ่ง อาจมีข้อมูลเวลาทั้ง 3 ชนิดเก็บอยู่ หรือ อาจไม่มีเลยก็ได้ เพราะข้อมูลเวลาทั้ง 3 ชนิดนั้น ไม่มีความสัมพันธ์กัน SQL-92 สนับสนุนเพียง User-defined Time เท่านั้น

2.1.3 คำสั่งเชิงเวลา

ในการใช้งานฐานข้อมูลเชิงเวลา สามารถสั่งได้ 3 ลักษณะ คือ

- คำสั่งปัจจุบัน (Current Query) คือการสั่งให้กระทำกับข้อมูล ณ ปัจจุบัน หรือ ขณะนี้
- คำสั่ง Sequenced (Sequenced Query) คือคำสั่งที่กระทำโดยคำนึงถึงตราเวลา (Timestamp) ที่ประทับไว้กับข้อมูล เช่น หากเรามีตารางข้อมูลเชิงเวลาชนิด Valid Time 2 ตารางคือ ข้อมูลตำแหน่งพนักงาน และ ข้อมูลเงินเดือนพนักงาน คำสั่งสืบค้นว่า พนักงานคนหนึ่งเคยมีตำแหน่งอะไร และได้รับเงินเดือนเท่าไร ในลักษณะ Sequenced คือจะนำช่วงเวลาที่เป็นจริงมาพิจารณาด้วยว่า ณ ช่วงเวลาใด พนักงานคนนี้ มีตำแหน่งอะไร และ รับเงินเดือนเท่าไร เนื่องจากการเปลี่ยนตำแหน่งและการเปลี่ยนเงินเดือน ไม่จำเป็นที่ต้องทำพร้อมๆกัน
- คำสั่ง Nonsequenced (Nonsequenced Query) คือคำสั่งที่มองว่า ตราเวลาที่ประทับไว้ ก็เป็นข้อมูลเช่นกัน แตกต่างกับ Sequenced Query ที่มองตราเวลาเป็นอีกส่วนหนึ่งต่างหาก ดังนั้น สามารถนำข้อมูลตราเวลามาใช้ประมวลได้

2.1.4 ช่วงเวลา

เป็นระยะเวลาที่มีกำหนดเริ่มต้นและกำหนดสิ้นสุด การนำเสนอช่วงเวลาสามารถทำได้แบบปลายปิด ซึ่งแสดงด้วยสัญลักษณ์ ‘[ ]’ หมายความว่า ช่วงเวลานั้น รวมจุดเวลาเริ่มต้นหรือสิ้นสุดที่ระบุไว้ด้วย หรือแบบปลายเปิด แสดงด้วยสัญลักษณ์ ‘( )’ หมายความว่า ช่วงเวลาไม่รวมจุดเวลาเริ่มต้นหรือสิ้นสุด (Snodgrasss. 2000 : 90)

การนำเสนอ 2 แบบสามารถผสมกันเป็น ช่วงเวลา 4 ลักษณะ หากให้ a และ b เป็นจุดเวลา การนำเสนอช่วงเวลามีได้ดังนี้คือ

- PERIOD [a, b] เป็นช่วงเวลาแบบปลายปิด-ปิด (Closed-Closed Representation)
- PERIOD [a, b) เป็นช่วงเวลาแบบปลายปิด-เปิด (Closed-Open Representation)
- PERIOD (a, b] เป็นช่วงเวลาแบบปลายเปิด-ปิด (Open-Closed Representation)
- PERIOD (a, b) เป็นช่วงเวลาแบบปลายเปิด-เปิด (Open-Open Representation)

ในฐานข้อมูลเชิงเวลา การเปรียบเทียบช่วงเวลาควรเปรียบเทียบช่วงเวลาที่นำเสนอแบบเดียวกัน การเปรียบเทียบช่วงเวลาที่นำเสนอต่างกัน ต้องนำความหยาบละเอียดของจุดเวลาเข้ามาเกี่ยวข้องด้วยว่ามีหน่วยเป็นวัน เป็นชั่วโมง หรือ เป็นนาที มิฉะนั้นความหมายจะเปลี่ยนแปลงช่วงเวลาที่เหมาะสม และเมื่อใช้ ต้องคำนึงถึงความหยาบละเอียดของจุดเวลาน้อยคือช่วงเวลาแบบปลายปิด-เปิด ดังนั้น งานวิจัยฉบับนี้จะยึดช่วงเวลาแบบปลายปิด-เปิดเป็นหลัก

ช่วงเวลาแบบปลายปิด-เปิด PERIOD [a , b) พบว่า a ต้องน้อยกว่า b เสมอจึงจะเป็นช่วงเวลาที่เป็นจริง

ความสัมพันธ์ของช่วงเวลา

หาก A = PERIOD [a1, a2), B = PERIOD [b1, b2) (Snodgrasss. 2000 : 91)

ตารางที่ 2.1 ความสัมพันธ์ของช่วงเวลา

ความสัมพันธ์	ความหมาย	ความสัมพันธ์เทียบเท่า
A equals B	a1 = b1 AND a2 = b2	B equals A
A before B	a2 < b1	B before $^{-1}$ A
A meets B	a2 = b1	B meet $^{-1}$ A
A overlaps B	a1 < b1 AND a2 < b2 AND b1 < a2	B overlaps $^{-1}$ A
A during B	b1 < a1 AND a2 < b2	B during $^{-1}$ A
A starts B	a1 = b1 AND a2 < b2	B starts $^{-1}$ A
A finishes B	b1 < a1 AND b2 = a2	B finishes $^{-1}$ A

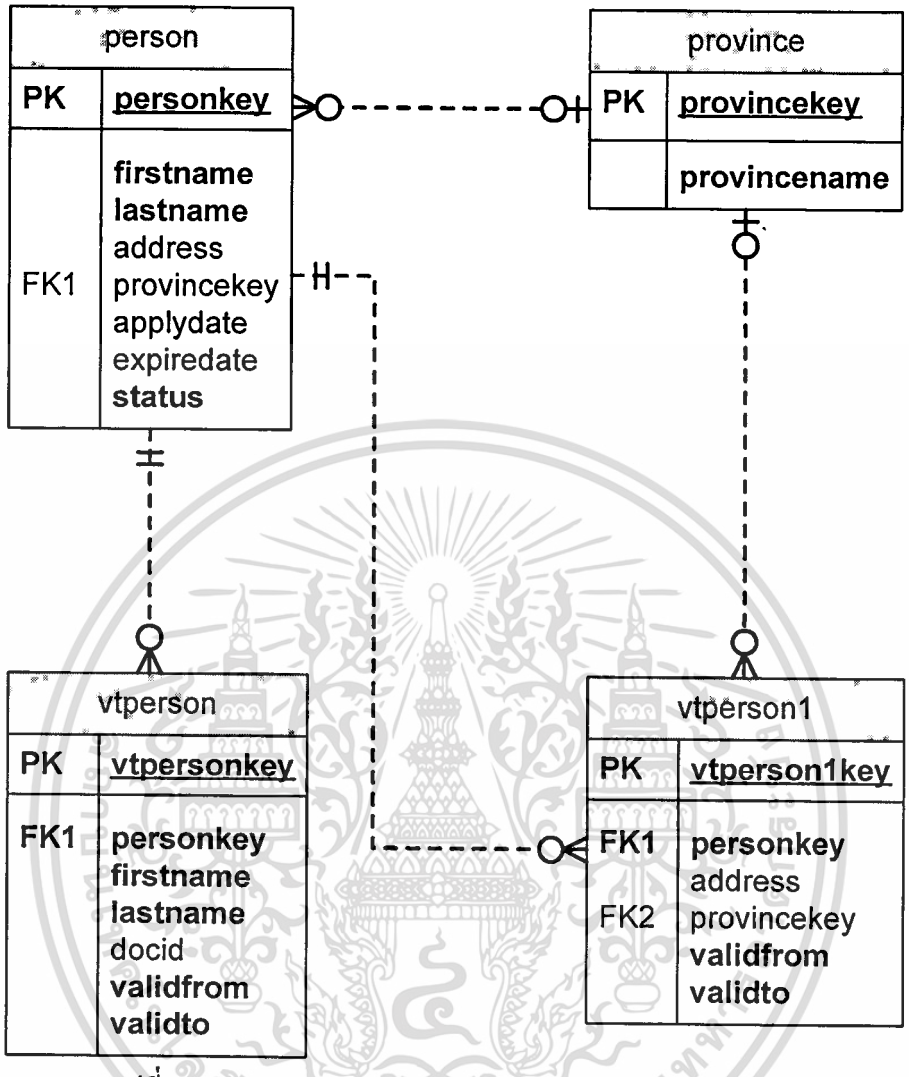
2.1.5 ฐานข้อมูลตัวอย่าง temporaldb

ในงานวิจัยฉบับนี้ จะใช้ฐานข้อมูล temporaldb เป็นตัวอย่างในการอธิบายหลักการต่างๆ ของฐานข้อมูลเชิงเวลา โดยฐานข้อมูล temporaldb มีตารางข้อมูลดังรูป 2.1

ตารางข้อมูล person และ province เป็นส่วนหนึ่งของฐานข้อมูลปกติ โดยมี vtperson กับ vtperson1 เป็นตารางข้อมูลเชิงเวลา ในบทนี้ ขอให้สนใจเพียงตารางข้อมูล vtperson และ vtperson1 เท่านั้น เพราะต้องการใช้ในการอธิบายหลักการฐานข้อมูลเชิงเวลา

นอกจากนี้ vtpersonkey และ vtperson1key เป็น Primary Key ของตารางข้อมูล vtperson และ vtperson1 แต่เป็น Primary Key ในความหมายของฐานข้อมูลเชิงสัมพันธ์ ไม่ใช่ Primary Key ในความหมายของฐานข้อมูลเชิงเวลา

ตารางข้อมูล person และ province จะใช้ในบทถัดไปเพื่ออธิบายแนวทางประยุกต์ใช้ฐานข้อมูลเชิงเวลาร่วมกับฐานข้อมูลปกติ



รูปที่ 2.1 Entity-Relationship Diagram ของฐานข้อมูลตัวอย่าง

2.2 ประโยชน์ของฐานข้อมูลเชิงเวลา

การใช้งานฐานข้อมูลเชิงเวลาในระดับ Valid Time Period ให้ประโยชน์กับผู้ใช้ดังนี้

2.2.1 การสืบค้นลักษณะ Time-Slice ในตารางเดียว

เป็นการสืบค้นว่า ข้อมูล ณ ช่วงเวลาที่กำหนดนั้นเป็นอย่างไร (Snodgrasss. 2000 : 145) หากไม่ได้นำ Valid Time Period มาใช้ เช่น ตารางข้อมูลที่มีเพียงคอลัมน์ที่เป็น User-Defined Time การสืบค้นลักษณะนี้

หากไม่มีการเก็บประวัติไว้ในรูปแบบใดๆ ไม่สามารถสืบค้นได้

หากเก็บประวัติในลักษณะจุดเวลา (วันที่เปลี่ยนแปลง) จะต้องตั้งสมมุติฐานก่อนว่า จะให้ข้อมูลเป็นช่วงเวลาอย่างไร ทำให้ไม่สามารถนำเสนอข้อมูลที่เป็นจริง

ตัวอย่าง

ณ วันที่ 1/1/2546 พนักงานรหัส 10000 ชื่อ นามสกุลอะไร

ณ ช่วงเวลาตั้งแต่ ปี 2510 ถึง ปี 2548 บุคคล รหัส 10000 ใช้ชื่อ นามสกุลอะไร

2.2.2 การสืบค้นข้อมูลในอดีตเชื่อมโยงหลายตาราง

ข้อมูลที่เราต้องการอาจถูกแยกออกเก็บไว้ในหลายตาราง แต่ละตาราง หากเราต้องการสืบค้นหาความสัมพันธ์เชิงเวลาของข้อมูลเหล่านั้น

หากไม่มีการเก็บประวัติไว้ในรูปแบบใดๆ ไม่สามารถสืบค้นได้

หากเก็บประวัติไว้ในลักษณะจุดเวลา การเชื่อมตารางโดยใช้ช่วงเวลาเพื่อหาข้อมูลที่สัมพันธ์กัน ไม่น่าเป็นไปได้ (เพราะไม่มีช่วงเวลา) การเชื่อมโดยอาศัยจุดเวลาก็ไม่ให้ผลลัพธ์ตามต้องการ

การสืบค้นเชิงเวลาต้องอาศัย Sequnced Join เพื่อให้ได้ผลลัพธ์ตามต้องการ (Snodgrasss. 2000 : 146)

ตัวอย่าง

ดูประวัติการย้ายสาขา การเลื่อนตำแหน่ง และเงินเดือนของพนักงาน (หากข้อมูลการย้ายสาขา การเลื่อนตำแหน่ง และ ข้อมูลเงินเดือนอยู่ต่างตารางข้อมูลเชิงเวลา)

2.2.3 การสืบค้นหาความสัมพันธ์เชิงเวลาระหว่างแถวข้อมูล

ติดตามความสัมพันธ์ของแถวข้อมูล โดยกำหนดว่า ต้องการความสัมพันธ์บนคอลัมน์ใด (ที่ไม่ใช่ Primary Key) และ ช่วงเวลาใด

เช่นเดียวกับกรณีข้อ 2.2.1 และ 2.2.2 หากไม่มีการเก็บ Valid Time Period ไว้ ก็ไม่สามารถสืบค้นได้ เพราะกรณีนี้จำเป็นต้องใช้ Sequenced Join หรือการหาช่วงเวลาที่เหลื่อมซ้อนกัน (Snodgrasss. 2000 : 146)

ตัวอย่าง

นาย ก เคยมีโอกาสรู้จักนาย ข บ้างไหน เช่น เรียนอยู่ที่เดียวกัน ในช่วงเวลาเดียวกัน หรืออยู่ตำบลเดียวกัน ในช่วงเวลาเดียวกัน

กำหนดช่วงเวลา นาย ก เคยมีโอกาสรู้จักนาย ข บ้างไหน เช่น เรียนอยู่ที่เดียวกัน ในช่วงระหว่างปี 2000 และปี 2004 หรือ อยู่ตำบลเดียวกัน ในช่วงเวลาที่กำหนด

2.3 ตารางข้อมูลเชิงเวลา

ตารางข้อมูลเชิงเวลาชนิด Valid Time คือตารางข้อมูลที่มีตราเวลา ประทับอยู่เพื่อบอกให้ทราบว่าแถวข้อมูลหนึ่งๆ เป็นจริงเฉพาะในช่วงเวลาใด

เราใช้คอลัมน์ ชื่อว่า validfrom และ validto เป็นการนำเสนอช่วงเวลาแบบปลายปิด-เปิด โดยให้ช่วงเวลา Valid Time นั้นเป็นคุณสมบัติของแต่ละแถวข้อมูล

Steiner (1998 : 24) เสนอว่า การสร้างตราเวลาในระดับแถวข้อมูลก่อให้เกิด Redundancy ขึ้น เนื่องจากแต่ละคอลัมน์ในแถวข้อมูลต่างมีช่วงเวลาของตนเอง ดังนั้น ตราเวลาควรผูกติดอยู่กับระดับคอลัมน์ ซึ่งนำไปสู่แนวความคิดว่า ฐานข้อมูลเชิงวัตถุน่าจะเหมาะสมกับการนำฐานข้อมูลเชิงเวลาไปใช้มากกว่าฐานข้อมูลเชิงสัมพันธ์

ในงานวิจัยฉบับนี้ เนื่องจากต้องการนำฐานข้อมูลเชิงเวลาไปใช้กับฐานข้อมูลที่มีอยู่ในปัจจุบันซึ่งเป็นฐานข้อมูลเชิงสัมพันธ์จึงเลือกใช้ตราเวลาในระดับแถวข้อมูล อย่างไรก็ตาม ในบทที่ 3 ซึ่งอธิบายถึงแนวทางประยุกต์ จะพบว่า สามารถสร้างตารางข้อมูลเชิงเวลาแบบนำตราเวลาไปไว้ในระดับคอลัมน์ก็ได้ หากในฐานข้อมูลเชิงเวลานั้นมีคอลัมน์ข้อมูลเพียงคอลัมน์เดียว

2.4 การปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลา

การปรับปรุงข้อมูลมี 3 ลักษณะ คือ Current Sequenced และ Nonsequenced อย่างไรก็ตาม งานวิจัยมุ่งประเด็นไปที่การปรับปรุงข้อมูลแบบ Sequenced เป็นหลัก คำอธิบายต่อไปนี้เป็นการทำงานแบบ Sequenced เท่านั้น

การปรับปรุงข้อมูลมีอยู่ 3 คำสั่งคือ INSERT DELETE และ UPDATE

ในชุดคำสั่งต่างๆ ที่มีในหัวข้อนี้ ชื่อที่นำหน้าด้วย "@" หมายความถึงพารามิเตอร์ที่บรรจุค่าตามคำสั่งต่างๆ

2.4.1 INSERT

การเพิ่มข้อมูลเชิงเวลา เพื่อรักษา Temporal Uniqueness ต้องไม่สามารถเพิ่มข้อมูลที่มีช่วงเวลาเหลื่อมกับช่วงเวลาที่มีอยู่แล้วในตารางข้อมูลเชิงเวลา (Snodgrasss. 2000 : 188)

ตัวอย่างคำสั่งสำหรับเพิ่มข้อมูลเชิงเวลาเข้าไปในตารางข้อมูล vtperson เป็นดังนี้

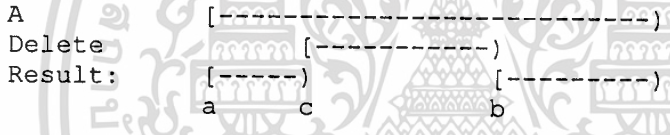
```
INSERT vtperson (
      personkey, firstname, lastname, docid,
      validfrom, validto)
VALUES (
      @personkey, @firstname, @lastname, @docid,
      @validfrom, @validto)
```

คำสั่งที่ 2.1 Temporal Insert

2.4.2 DELETE

การลบข้อมูลเชิงเวลา จะซับซ้อนกว่าการเพิ่มข้อมูล โดยระหว่างช่วงเวลาที่มีอยู่ (A) กับช่วงเวลาที่ต้องการลบ (Delete) มีความสัมพันธ์กัน 4 กรณี ดังต่อไปนี้ โดย a และ b เป็นการ Update หรือ Insert ในขณะที่ c เป็นการ Delete (Snodgrasss. 2000 : 190)

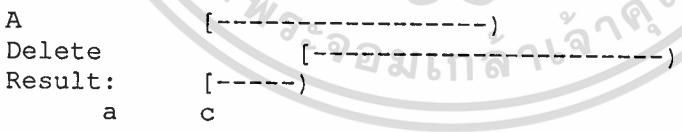
กรณี 1



เงื่อนไขของกรณีนี้ สามารถเขียนเป็น Selection Clause ได้ดังนี้

```
WHERE A.validfrom < @validfrom AND A.validto > @validto
```

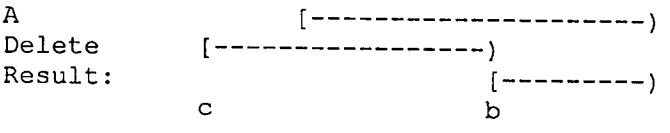
กรณี 2



เงื่อนไขของกรณีนี้ สามารถเขียนเป็น Selection Clause ได้ดังนี้

```
WHERE A.validto > @validfrom AND A.validto < @validto
```

กรณี 3



เงื่อนไขของกรณีนี้ สามารถเขียนเป็น Selection Clause ได้ดังนี้

```
WHERE A.validfrom < @validto AND A.validto > @validto
```

กรณี 4

```
A                   [------------------)
Delete       [--------------------------------------)
Result: entire row deleted = c
```

เงื่อนไขของกรณีนี้ สามารถเขียนเป็น Selection Clause ได้ดังนี้

```
WHERE A.validfrom >= @validfrom AND A.validto <= @validto
```

คำสั่ง

ตัวอย่างคำสั่งของการลบข้อมูลเชิงเวลาจากตารางข้อมูล vtperson ดังนี้

```
--1
INSERT dbo.vtperson (
      personkey, firstname, lastname, docid,
      validfrom, validto)
SELECT personkey, firstname, lastname, docid,
      @validto, validto
FROM dbo.vtperson
WHERE personkey = @personkey
  AND validfrom < @validfrom
  AND validto > @validto

--2
UPDATE dbo.vtperson
SET validto = @validfrom
WHERE personkey = @personkey
  AND validfrom < @validfrom
  AND validto > @validfrom

--3
UPDATE dbo.vtperson
SET validfrom = @validto
WHERE personkey = @personkey
  AND validfrom < @validto
  AND validto > @validto

--4
DELETE FROM dbo.vtperson
WHERE personkey = @personkey
  AND validfrom >= @validfrom
  AND validto <= @validto
```

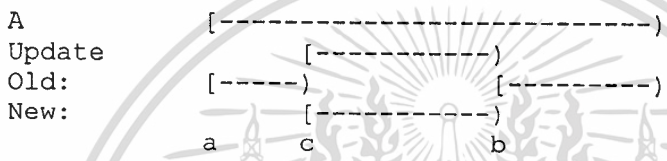
คำสั่งที่ 2.2 Temporal Delete

ประโยค 1 ในคำสั่ง ทำให้เกิดส่วน b ในกรณี 1 โดยประโยค 2 ทำให้เกิดส่วน a ในกรณี 1 และ 2 ส่วนประโยค 3 ทำให้เกิดส่วน b ในกรณี 3 และประโยค 4 ทำให้เป็นไปในกรณี 4

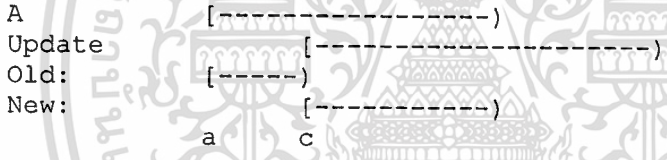
2.4.3 UPDATE

ในการ Update ทั่วไป มักสามารถใช้ Delete ข้อมูลเก่า และ Insert ข้อมูลใหม่เข้าไปได้ ในในการ Update เชิงเวลา ไม่สามารถทำเช่นนั้น ดังนั้น การใช้คำสั่งกับ 4 กรณีที่เกิดขึ้นจึงไม่เหมือนกับการ Delete ดังนี้ (Snodgrasss. 2000 : 194)

กรณี 1

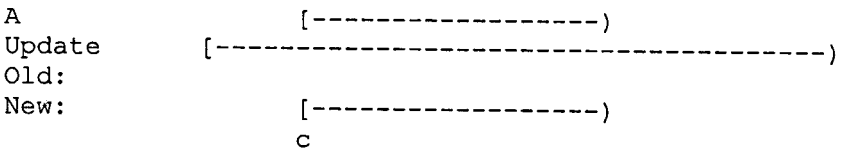


กรณี 2



กรณี 3

กรณี 4



คำสั่ง

ตัวอย่างคำสั่งของการแก้ไขข้อมูลเชิงเวลาจากตารางข้อมูล vtperson ดังนี้

```
--1
INSERT dbo.vtperson (personkey, firstname, lastname,
docid, validfrom, validto)
SELECT personkey, firstname, lastname, docid, validfrom,
@validfrom
FROM dbo.vtperson
WHERE personkey = @personkey
  AND validfrom < @validfrom
  AND validto > @validfrom

--2
INSERT dbo.vtperson (personkey, firstname, lastname,
docid, validfrom, validto)
SELECT personkey, firstname, lastname, docid, @validto,
validto
FROM dbo.vtperson
WHERE personkey = @personkey
  AND validfrom < @validto
  AND validto > @validto

--3
UPDATE dbo.vtperson
SET
  firstname= @firstname,
  lastname = @lastname,
  docid = @docid
WHERE personkey = @personkey
  AND validfrom < @validto
  AND validto > @validfrom

--4
UPDATE dbo.vtperson
SET validfrom = @validfrom
WHERE personkey = @personkey
  AND validfrom < @validfrom
  AND validto > @validfrom

--5
UPDATE dbo.vtperson
SET validto = @validto
WHERE personkey = @personkey
  AND validfrom < @validto
  AND validto > @validto
```

คำสั่งที่ 2.3 Temporal Update

ประโยค 1 ก่อให้เกิด a ในกรณี 1 และ 2 ในประโยค 2 ก่อให้เกิด b ในกรณี 1 และ 3 ส่วนประโยค 3 เป็นการปรับค่าในแถวข้อมูล c ให้เป็นค่าใหม่ในทั้ง 4 กรณี ประโยค 4 เป็นการปรับค่า

ในคอลัมน์ validfrom ของแถวข้อมูล c ให้เป็นค่าที่ถูกต้องในกรณี 1 และ 2 และ ประโยค 5 เป็นการปรับค่าคอลัมน์ validto ให้ถูกต้องในกรณี 1 และ 3

โดยสรุป กรณี 1 ต้องการประโยค 1(a), 2(b), 3(c-value), 4(c-validfrom), 5(c-validto) ส่วนกรณี 2 ต้องการประโยค 1(a), 3(c-value), 4(c-validfrom) กรณี 3 ต้องการประโยค 2(b), 3(c-value), 5(c-validto) และกรณี 4 ต้องการประโยค 3(c-value)

2.5 Temporal Constraint

2.5.1 Temporal Uniqueness Constraint

การพิจารณา Uniqueness ในฐานข้อมูลเชิงเวลา สามารถแบ่งความซ้ำซ้อนออกมาได้เป็น 4 แบบ (Snodgrasss. 2000 : 121) คือ

- Value-equivalent คือการที่ ค่าคีย์หนึ่งมีแถวข้อมูล 2 แถวหรือมากกว่า ที่มีข้อมูลที่ไม่ใช่ตราเวลา เหมือนกัน
- Sequenced Duplicates คือ ณ จุดเวลาหนึ่ง ค่าคีย์หนึ่งมีแถวข้อมูล 2 แถวหรือมากกว่า
- Current Duplicates ณ ปัจจุบัน ค่าคีย์หนึ่งมีแถวข้อมูล 2 แถวหรือมากกว่า
- Nonsequenced Duplicates คือ ค่าคีย์หนึ่งมีแถวข้อมูล 2 แถวหรือมากกว่า ที่มีข้อมูลเหมือนกันทั้งหมด (รวมข้อมูลตราเวลาด้วย)

ในที่นี้ พิจารณาเฉพาะ Sequenced Uniqueness นั่นคือการป้องกันไม่ให้เกิด Sequenced Duplicates นั่นเอง

การรักษาความเป็น Sequenced Uniqueness ของคอลัมน์เชิงเวลาคือ ณ เวลาใดๆ จะมีข้อมูลได้เพียงแถวข้อมูลเดียวสำหรับค่าคีย์ค่าหนึ่ง ดังนั้นต้องไม่มีเหตุการณ์ Overlaps, During, หรือ Equals ระหว่างแถวข้อมูลที่เป็นของคีย์นั้นๆ

ตารางข้อมูล vtperson และ vtperson1 มี personkey เป็นคีย์ การตรวจสอบแบ่งได้เป็น 4 กรณีเช่นเดียวกัน

กรณี 1

```
A          [----------------------)
B                     [----------------------)
```

หากมองจาก A

กรณีที่ผิด temporal uniqueness constraint คือ

```
A.validto > B.validfrom AND
A.validto <= B.validto
```

หากมองจาก B

กรณีที่ผิด temporal uniqueness constraint คือ

```
B.validfrom >= A.validfrom AND
B.validfrom < A.validto
```

กรณี 2

```
A                        [----------------------)
B            [----------------------)
```

หากมองจาก A

กรณีที่ผิด temporal uniqueness constraint คือ

```
A.validfrom >= B.validfrom AND
A.validfrom < B.validto
```

หากมองจาก B

กรณีที่ผิด temporal uniqueness constraint คือ

```
B.validto > A.validfrom AND
B.validto <= A.validto
```

กรณี 3

```
A                        [-------------)
B            [-----------------------------------)
```

กรณีนี้ มองจาก A หรือ B จะให้ผลเหมือนกัน

กรณีทิผิด temporal uniqueness constraint คือ

```
A.validfrom >= B.validfrom AND
A.validto <= B.validto
```

อย่างไรก็ตาม หากตรวจสอบแค่ผิดหรือไม่ผิด Uniqueness ในกรณี 3 สามารถใช้ Clause เหมือนกรณี 2 ได้

กรณี 4

```
A              [----------------------------------)
B                          [------------)
```

กรณีนี้ มองจาก A หรือ B จะให้ผลเหมือนกัน

กรณีทิผิด temporal uniqueness constraint คือ

```
A.validfrom <= B.validfrom AND
A.validto >= B.validto
```

อย่างไรก็ตาม หากตรวจสอบแค่ผิดหรือไม่ผิด Uniqueness ในกรณี 3 สามารถใช้ Clause เหมือนกรณี 1 ได้

ขั้นตอนการตรวจสอบมีดังนี้

1. ทราบว่าค่าคีย์ที่ต้องการตรวจคืออะไร โดยทำการตรวจสอบทีละ 1 ค่าคีย์เท่านั้น
2. ดึงกลุ่มแถวข้อมูลของค่าคีย์นี้จากตารางข้อมูลเชิงเวลา
3. ทำการหาว่า เกิด overlaps, during, equals ขึ้นในกลุ่มแถวข้อมูลเหล่านั้นหรือไม่ โดยใช้ Self Join การเปรียบเทียบอาจใช้กรณี 1 เพียงกรณีเดียวก็พอแล้ว เพราะกรณี 1 กับ 2 นั้น เหมือนภาพกระจกซึ่งกันและกัน และกรณี 1 สามารถใช้ตรวจสอบกรณี 4 ส่วนกรณี 2 สามารถใช้ตรวจสอบกรณี 3

```
A.validfrom >= B.validfrom AND
A.validfrom < B.validto
```

หากทำ Query ตรวจสอบแบบ Self Join โดยยอมให้มีการ JOIN สลับข้าง (แต่ไม่ JOIN ตนเอง) ยกเพียงกรณีเดียวใน 2 แบบ ก็สามารถตรวจสอบได้ทั้งหมด (A สามารถเป็น B ได้ในประโยคถัดไป)

หากให้ Self Join ไม่สลับข้าง ต้องตรวจสอบทั้ง 2 กรณีในประโยคเดียว

2.5.2 Temporal Foreign Key Constraint

มี 4 กรณี (Snodgrasss. 2000 : 126)

ตารางที่ 2.2 กรณีของ Temporal Foreign Key Constraint

กรณี	วิธีการทำ Foreign Key Constraint
คอลัมน์ในตารางข้อมูลปกติอ้างถึงคอลัมน์ในตารางข้อมูลปกติ	ใช้ Foreign Key Constraint ปกติ
คอลัมน์ในตารางข้อมูลเชิงเวลาอ้างถึงคอลัมน์ในตารางข้อมูลปกติ	ใช้ Foreign Key Constraint ปกติ
คอลัมน์ในตารางข้อมูลเชิงเวลาอ้างถึงคอลัมน์ในตารางข้อมูลเชิงเวลา	การตรวจสอบคือ ในแต่ละแถวข้อมูล r ในตารางข้อมูลที่อ้างถึง: จะต้องมี 1 แถวข้อมูลที่มีค่าคีย์ตรงกับ r ในตารางข้อมูลที่ถูกอ้างเมื่อ r เริ่มต้น Valid Time จะต้องมี 1 แถวข้อมูลที่มีค่าคีย์ตรงกับ r ในตารางข้อมูลที่ถูกอ้างเมื่อง r จบ Valid Time จะต้องไม่มีช่องว่างเวลาในตารางข้อมูลที่ถูกอ้างในช่วงเวลา Valid Time ของค่าคีย์นั้นๆ
คอลัมน์ในตารางข้อมูลปกติอ้างถึงคอลัมน์ในตารางข้อมูลเชิงเวลา	ต้องนิยามว่าข้อมูลปกติคือช่วงเวลาอะไร มักมี 2 แบบคือ เป็น เวลาปัจจุบัน หรือ เป็นข้อมูลนิรันดร์ (ไม่เปลี่ยนแปลงตามเวลา) หากเป็นเวลาปัจจุบัน สามารถใช้ Current Foreign Key Constraint (Snodgrass. 2000 : 127) หากเป็นข้อมูลนิรันดร์ สามารถใช้ Foreign Key Constraint ปกติ

2.6 การสืบค้นข้อมูลเชิงเวลา

2.6.1 Temporal Query

การสืบค้นข้อมูลจากตารางข้อมูลเชิงเวลาแบ่งออกได้เป็น 3 ลักษณะคือ Current Sequenced และ Nonsequenced ดังนี้

Current Query

เป็นการสืบค้นโดยใช้คำถามแบบเดียวกับการถามจากตารางข้อมูลปกติ (Snodgrasss. 2000 : 143) เช่น

พนักงานรหัสนี้ ชื่ออะไร

คำถามลักษณะนี้ มักจะหมายความถึง ข้อมูล ณ ปัจจุบัน แต่ละไว้ การสืบค้นในตารางข้อมูลเชิงเวลา ต้องระบุว่า เป็นการสืบค้น ณ ปัจจุบัน

```
SELECT firstname, lastname
FROM vtperson
WHERE vtpersonkey = @vtpersonkey
  AND validfrom <= @now AND validto > @now
```

คำสั่งที่ 2.4 การสืบค้นปัจจุบัน

Nonsequenced Query

เป็นคำถามที่ใช้ข้อมูลตราเวลามาประมวลในคำถามด้วย และผลที่ได้รับไม่จำเป็นต้องกลายเป็นตารางข้อมูลเชิงเวลา ตัวอย่างที่เห็นได้ชัดคือ กรณี Time-Slice Query เป็นคำสั่งสอบถามข้อมูลในอดีต เช่น ณ เวลานั้น หรือ ณ ช่วงเวลานั้น ดังตัวอย่าง

ณ วันที่ 1 มกราคม 2000 พนักงานรหัสนี้ ชื่ออะไร

```
SELECT firstname, lastname
FROM vtperson
WHERE vtpersonkey = @vtpersonkey
  AND validfrom <= '1/1/2000' AND validto > '1/1/2000'
```

คำสั่งที่ 2.5 การสืบค้นแบบ Nonsequenced

Sequenced Query

เป็นคำถามที่ไม่ได้ใช้ตราเวลา ในการดึงข้อมูลจากตารางข้อมูลเชิงเวลามาประมวลผลต่อ (Snodgrasss. 2000 : 145) ผลของ Sequenced Query จะได้ออกมาเป็นตารางข้อมูลเชิงเวลา ตัวอย่างเช่น

พนักงานคนใดชื่อ Somchai

```
SELECT *
FROM vtperson
WHERE firstname = 'Somchai'
```

คำสั่งที่ 2.6 การสืบค้นแบบ Sequenced

การทำ Sequenced Query จากหลายตารางข้อมูล จำเป็นต้องมีการทำ Sequenced Join ซึ่งจะกล่าวในหัวข้อถัดไป

2.6.2 Sequenced Join

การทำ Sequenced Join ในฐานข้อมูลเชิงเวลา ต้องพิจารณา 4 กรณี (Snodgrasss. 2000 : 148) เช่นกันดังนี้

กรณี 1

```
A            [---------------------)
B                  [---------------------)
Result             [---------------)
```

เงื่อนไขที่เกิดกรณีนี้คือ

```
A.validto > B.validfrom AND
A.validto <= B.validto
```

คำสั่งที่ 3 คือ ผลของการ Join ในกรณีนี้ แต่ผลที่ได้รับอาจซ้ำกับ กรณี 4 ดังนั้นจึงต้องเพิ่มเงื่อนไขเข้าไปอีก 1 ประโยคคือ

```
A.validfrom < B.validfrom
```

กรณี 2

```
A                  [---------------------)
B            [---------------------)
Result             [---------------)
```

เงื่อนไขที่เกิดกรณีนี้คือ

```
A.validfrom >= B.validfrom AND
A.validfrom < B.validto
```

คำสั่งที่ 2 คือ ผลของการ Join ในกรณีนี้ แต่ผลที่ได้รับอาจซ้ำกับ กรณี 3 ดังนั้นจึงต้องเพิ่มเงื่อนไขเข้าไปอีก 1 ประโยคคือ

```
A.validto > B.validto
```

กรณี 3

```
A                  [---------------------)
B            [-----------------------------------)
Result             [---------------------)
```

เงื่อนไขที่เกิดกรณีนี้คือ

```
A.validfrom >= B.validfrom AND
A.validto <= B.validto
```

คำสั่งที่ 1 เป็นของกรณีนี้ ก่อให้เกิดช่วงเวลาที่ Intersect กันคือมาจากแถวข้อมูล A ทั้งหมด

กรณี 4

```
A           [----------------------------------)
B                 [----------------------)
Result            [----------------------)
```

เงื่อนไขที่เกิดกรณีนี้คือ

```
A.validfrom < B.validfrom AND
A.validto > B.validto
```

คำสั่งที่ 4 เป็นของกรณีนี้ ก่อให้เกิดช่วงเวลาที่ Intersect กันคือมาจากแถวข้อมูล B ทั้งหมด

คำสั่ง

```
--1. during, equals
SELECT A.<primarykey>, A.<attr_a>, B.<attri_b>,
A.validfrom, A.validto
FROM A JOIN B ON A.<primarykey> = B.<primarykey>
WHERE A.validfrom >= B.validfrom
  AND A.validto <= B.validto

UNION ALL

--2. overlaps-1
SELECT A.<primarykey>, A.<attr_a>, B.<attri_b>,
A.validfrom, B.validto
FROM A JOIN B ON A.<primarykey> = B.<primarykey>
WHERE A.validfrom >= B.validfrom
  AND A.validfrom < B.validto
  AND A.validto > B.validto

UNION ALL

--3. overlaps
SELECT A.<primarykey>, A.<attr_a>, B.<attri_b>,
B.validfrom, A.validto
FROM A JOIN B ON A.<primarykey> = B.<primarykey>
WHERE  A.validto > B.validfrom
  AND A.validto <= B.validto
  AND A.validfrom < B.validfrom


UNION ALL

--4. during-1
SELECT A.<primarykey>, A.<attr_a>, B.<attri_b>,
B.validfrom, B.validto
FROM A JOIN B ON A.<primarykey> = B.<primarykey>
WHERE A.validfrom < B.validfrom
  AND A.validto > B.validto
```

คำสั่งที่ 2.7 Sequenced Join

2.7 คำสั่ง SQL เชิงเวลา

TimeCenter เป็นศูนย์วิจัยซึ่งให้การสนับสนุนการวิจัยและนำฐานข้อมูลเชิงเวลามาใช้บน DBMS ที่มีอยู่ในปัจจุบันและที่กำลังพัฒนาอยู่ มีสำนักงานอยู่ที่ Aalborg University ในประเทศเดนมาร์ก และ University of Arizona ในประเทศอเมริกา มีเวปไซต์อยู่ที่ http://www.cs.auc.dk/research/DP/tdb/TimeCenter นักวิจัยด้านฐานข้อมูลเชิงเวลาได้มีส่วนร่วมและนำเสนอมาตรฐานเกี่ยวกับฐานข้อมูลเชิงเวลาเพื่อให้เข้าไปอยู่ในมาตรฐาน SQL3 (ปัจจุบันเรียกว่า SQL-99) แต่ยังไม่ได้รับการบรรจุให้เป็นมาตรฐานอยู่ใน SQL-99 นี้ (Snodgrass. et al. 1997)

อย่างไรก็ตาม คีย์เวิร์ดบางตัวของ SQL เชิงเวลา (Snodgrass. 2000 : 417) มีส่วนช่วยให้การเขียนคำสั่งเชิงเวลาได้ง่ายขึ้น งานวิจัยฉบับนี้จึงยกบางคีย์เวิร์ดมาใช้ดังนี้

2.7.1 การปรับปรุงข้อมูลเชิงเวลา

INSERT

```
VALIDTIME PERIOD ( <valid_from> , <valid_to> )
INSERT [ INTO ] <table_name> [ ( <column_list> ) ]
VALUES ( { DEFAULT | NULL | <expression> } [ , … n ] )
```

คำสั่งที่ 2.8 SQL3 Insert

โดยมี VALIDTIME เป็นคีย์เวิร์ดที่แสดงว่า คำสั่งนี้เป็น คำสั่งเชิงเวลา เพื่อเพิ่มข้อมูลเข้าไปในฐานข้อมูลเชิงเวลา ในช่วงเวลาที่กำหนดไว้ใน PERIOD (<validfrom>, <validto>)

คำสั่งนี้เวลาทำงานจะถูกแปลเป็น คำสั่ง SQL ปกติ ตาม คำสั่งที่ 2.1

DELETE

```
VALIDTIME PERIOD ( <valid_from> , <valid_to> )
DELETE [ FROM ] <table_name>
WHERE <search_condition>
```

คำสั่งที่ 2.9 SQL3 Delete

VALIDTIME เป็นคีย์เวิร์ดที่แสดงว่า คำสั่งนี้เป็นคำสั่งเชิงเวลา เพื่อลบข้อมูลที่กำหนดใน <search condition> เฉพาะในช่วงเวลาที่กำหนดใน PERIOD (<validfrom>, <validto>)

เวลาทำงาน คำสั่งจะถูกแปลเป็น SQL ปกติ ตามคำสั่งที่ 2.2 รวม 4 ประโยค

UPDATE

```
VALIDTIME PERIOD ( <valid_from> , <valid_to> )
UPDATE <table_name>
SET
      { <column_name> = { DEFAULT | NULL | <expression> }
      [ , … n ]
WHERE
      <search_condition>
```

คำสั่งที่ 2.10 SQL3 Update

VALIDTIME เป็นคีย์เวิร์ดที่แสดงว่า คำสั่งนี้เป็นคำสั่งเชิงเวลา เพื่อแก้ไขข้อมูลที่กำหนดใน <search condition> เฉพาะในช่วงเวลาที่กำหนดใน PERIOD (<validfrom>, <validto>)

เวลาทำงาน คำสั่งจะถูกแปลเป็น SQL ปกติ ตามคำสั่งที่ 2.3 รวม 5 ประโยค

2.7.2 การสืบค้นข้อมูลเชิงเวลา

```
[ NONSEQUENCED ] VALIDTIME
[ PERIOD ( <valid_from> , <valid_to> ) ]
SELECT { <select_list> | VALIDTIME( ) } [ , … n ]
FROM <table_list>
WHERE <search_condition>
GROUP BY <group_by_expression>
HAVING <having_search_condition>
ORDER BY <order_by_expression>
```

คำสั่งที่ 2.11 SQL3 Select

VALIDTIME เป็นคีย์เวิร์ดที่แสดงว่า คำสั่งนี้เป็นคำสั่งเชิงเวลา NONSEQUENCED เป็นคีย์เวิร์ดเพื่อบอกว่า การสืบค้นนี้ ให้ทำแบบ Nonsequenced (Snodgrass. 2000 : 415)

หากไม่มีคีย์เวิร์ด Nonsequenced และใน <table_list> และข้อมูลมาจากตารางข้อมูลมากกว่า 1 ตาราง เป็นการทำ Sequenced Join แต่หากใช้คีย์เวิร์ด Nonsequenced การ Join เป็นการ Join ปกติตามเงื่อนไขที่กำหนด

2.8 ขอบเขตของงานวิจัย

2.8.1 ภาพรวม

ประเภทข้อมูลเชิงเวลาที่ให้ความสนใจคือ ช่วงเวลา และใช้งานนำเสนอช่วงเวลาแบบปลาย-เปิด (Closed-Open Representation)

ส่วนชนิดของเวลา งานวิจัยจะพิจารณาเฉพาะ Valid Time เท่านั้น จะไม่ลงไปที่ Transaction Time

คำสั่งเชิงเวลา หากเป็นคำสั่งปรับปรุงข้อมูลจะใช้คำสั่ง Sequenced เป็นหลัก แต่สำหรับการสืบค้น อาจมีทั้ง Sequenced และ Nonsequenced

2.8.2 Temporal Constraint

มีการควบคุม Temporal Uniqueness Constraint ในงานวิจัย เพื่อรักษา Uniqueness ของข้อมูลเชิงเวลาเมื่อมีการปรับปรุงข้อมูล

เนื่องจากขอบเขตงานเป็นการทำข้อมูลเชิงเวลาบนแอพพลิเคชันปกติ ดังนั้น การใช้ Foreign Key Constraint จะเป็นในลักษณะที่ 2 เสมอ เพราะอิงไปยังตารางข้อมูลปกติ ดังนั้นในงานวิจัยจะไม่อธิบายถึงกรณีที่ 3 และ 4

2.8.3 คำสั่ง SQL เชิงเวลา

คีย์เวิร์ดที่มีอยู่ในงานวิจัย มีดังนี้

Period Constructor

ตารางที่ 2.3 Period Constructor

คีย์เวิร์ด	ความหมาย
VALIDTIME(table)	ดึงข้อมูล Valid From และ Valid To ออกจากแถวข้อมูลเชิงเวลา มีอยู่ใน Select List และ Search Condition
VALIDTIME( )	แสดง Valid time period ของผลลัพธ์ที่ได้จากการทำ operation ทั้งหมดแล้ว ระบุไม่ได้ว่า จะออกมาจากตารางข้อมูลใด ในกรณีมีการ Join เกิดขึ้น มีได้ใน Select List เท่านั้น
PERIOD (date , date)	สร้างช่วงเวลาจาก ข้อมูลประเภท datetime 2 ตัว มีอยู่ใน Select List และ Search Condition

Predicates

ตารางที่ 2.4 Predicates

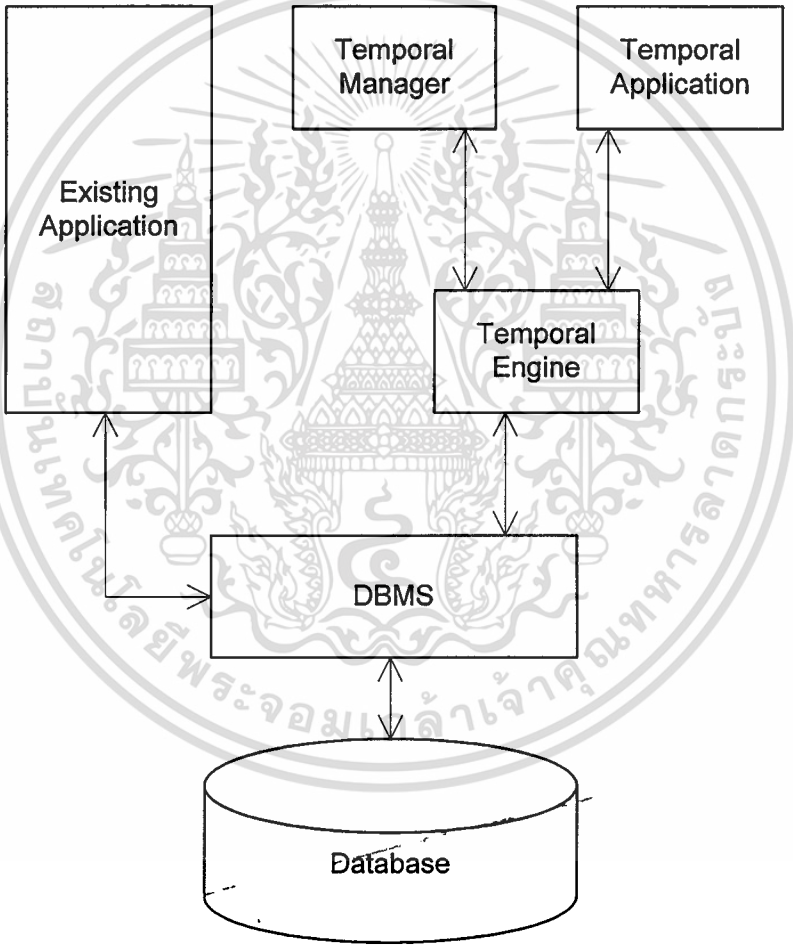
คีย์เวิร์ด	ความหมาย
a PRECEDES b	a.validto <= b.validfrom มีอยู่ใน Search Condition ของ การสืบค้นแบบ Nonsequenced
a SUCCEEDS b	a.validfrom >= b.validto มีอยู่ใน Search Condition ของ การสืบค้นแบบ Nonsequenced
a OVERLAPS b	(a.validto > b.validfrom AND a.validto <= b.validto) OR (a.validfrom >= b.validfrom AND a.validfrom < b.validto) OR (a.validfrom <= b.validfrom AND a.validto >= b.validto) มีอยู่ใน Search Condition ของ การสืบค้นแบบ Nonsequenced

บทที่ 3

แนวทางประยุกต์ใช้ฐานข้อมูลเชิงเวลาร่วมกับฐานข้อมูลปกติ

3.1 แนวทางแก้ไขปัญหา

ตามรูป 3.1 เป็นโมเดลการประยุกต์ใช้ฐานข้อมูลเชิงเวลาร่วมกับฐานข้อมูลปกติ



รูปที่ 3.1 Application Model

Existing Application แอพพลิเคชันปกติที่ใช้ฐานข้อมูลปกติอยู่ ติดต่อกับฐานข้อมูลโดยผ่าน DBMS (ในที่นี้คือ Microsoft SQL Server 2000)

Temporal Manager โปรแกรมเครื่องมือ ช่วยในการสร้างฐานข้อมูลเชิงเวลาบนฐานข้อมูลปกติ

Temporal Application	แอพพลิเคชันฐานข้อมูลเชิงเวลา ที่เข้าถึงข้อมูลเชิงเวลา โดยอาศัย Temporal Engine เพื่อสามารถสืบค้น ปรับปรุงข้อมูลเชิงเวลาได้โดยง่าย
Temporal Engine	ซอฟต์แวร์คอมโปแนนท์ที่ทำหน้าที่ให้บริการฐานข้อมูลเชิงเวลาให้กับโปรแกรมต่างๆ รวมทั้ง Temporal Manager ซึ่งเป็นโปรแกรมจัดการ และ แอพพลิเคชันอื่นๆที่สร้างขึ้นเพื่อให้ผู้ใช้ได้ใช้ประโยชน์จากฐานข้อมูลเชิงเวลา
DBMS	ระบบจัดการฐานข้อมูล
Database	ฐานข้อมูลปกติ ที่ได้ถูกทำให้รองรับคุณสมบัติฐานข้อมูลเชิงเวลาได้

เครื่องมือจะประกอบด้วย Temporal Manager ทำหน้าที่เป็นแอพพลิเคชันจัดการฐานข้อมูลเชิงเวลา ช่วยนักพัฒนาในการสร้างตารางข้อมูลเชิงเวลาจากตารางข้อมูลปกติ และ Temporal Engine เป็นซอฟต์แวร์คอมโปแนนท์ทำหน้าที่ประมวลคำสั่งเชิงเวลา แอพพลิเคชันเชิงเวลาที่นักพัฒนาสร้างขึ้น สามารถเรียกใช้บริการจาก Temporal Engine เพื่อติดต่อกับตารางข้อมูลเชิงเวลาและส่งคำสั่งเชิงเวลาไปประมวลผลได้

เห็นว่า ตามโมเดลนี้ แอพพลิเคชันเดิมที่ใช้งานฐานข้อมูลปกติอยู่ จะไม่ได้รับผลกระทบจากการนำฐานข้อมูลเชิงเวลามาใช้ เพราะ ถึงแม้จะเป็นฐานข้อมูลเดียวกัน แต่ตารางข้อมูลเดิมไม่มีการเปลี่ยนแปลง

โดยเครื่องมือและแนวทางที่สร้างขึ้นสามารถแก้ไขประเด็นปัญหาการนำฐานข้อมูลเชิงเวลาไปใช้ทั้ง 7 ประการดังนี้

3.1.1 ยากต่อความเข้าใจ

เครื่องมือนี้ ต้องสามารถแยกฐานข้อมูลธรรมดากับฐานข้อมูลเชิงเวลาออกจากกันในแง่มุมมองของผู้ที่เกี่ยวข้องกับระบบสารสนเทศได้ ผู้ที่เกี่ยวข้องกับฐานข้อมูลธรรมดาสามารถใช้งานฐานข้อมูลในแง่ของข้อมูลปัจจุบันตามปกติ มีแต่ผู้ที่เกี่ยวข้องกับข้อมูลเชิงเวลาเท่านั้นที่จำเป็นต้องเรียนรู้การใช้งานข้อมูลเชิงเวลา

ปัญหาข้อนี้ ทำให้ต้องรักษาตารางข้อมูลเดิมซึ่งเป็นตารางข้อมูลปกติไว้ การทำงานกับข้อมูลเชิงเวลา ให้ไปกระทำที่ตารางข้อมูลเชิงเวลาที่สร้างขึ้นมาให้

จะเห็นว่า ข้อมูลที่ตารางข้อมูลปกติจะเป็นเสมือนมุมมองปัจจุบันของตารางข้อมูลเชิงเวลา ดังนั้น ระบบต้องสามารถรักษาความถูกต้องของข้อมูลทั้งสองตารางไว้ได้ด้วย

3.1.2 เปลี่ยนฐานข้อมูลปกติที่มีอยู่แล้วได้ยาก

การเปลี่ยนฐานข้อมูลธรรมดาให้เป็นฐานข้อมูลเชิงเวลา ควรเป็นในกรณีที่แอพพลิเคชันก็ยังเห็นฐานข้อมูลเหมือนเดิม เฉพาะการทำงานใหม่ๆที่เพิ่มขึ้นในส่วนของข้อมูลเชิงเวลาเท่านั้นที่เห็นฐานข้อมูลเชิงเวลา

เมื่อไม่ต้องทิ้งตารางข้อมูลเดิม แต่เป็นการเพิ่มตารางข้อมูลใหม่ที่เป็นข้อมูลเชิงเวลาเข้าไป แอพพลิเคชันต่างๆที่ทำงานบนฐานข้อมูลเดิมก็สามารถทำงานได้ตามปกติ ส่วนแอพพลิเคชันฐานข้อมูลเชิงเวลาที่เพิ่มขึ้น ก็ใช้งานในส่วนฐานข้อมูลเชิงเวลาไป .

3.1.3 ขาดแคลนคำสั่งที่ใช้ในการสืบค้นเชิงเวลา

เครื่องมือต้องสามารถให้ผู้ใช้งานสืบค้นข้อมูลในรูปแบบคำสั่งเชิงเวลาตาม Proposed SQL-3 Standard ได้ ส่วนผู้ใช้งานสืบค้นข้อมูลปัจจุบันต้องสามารถใช้คำสั่งแบบเดิมได้โดยไม่ต้องเปลี่ยนแปลงใดๆ

ระบบสร้าง Temporal Engine ขึ้นมาเพื่อทำหน้าที่แปลคำสั่ง SQL เชิงเวลา ให้เป็นคำสั่ง Transact-SQL เพื่อส่งไปประมวลผลที่ Microsoft SQL Server ต่อไป

3.1.4 สมรรถนะในการทำงาน

เครื่องมือต้องทำให้การสืบค้นปกติมีสมรรถนะไม่ด้อยกว่าการสืบค้นในฐานข้อมูลธรรมดา

ไม่มีการเปลี่ยนแปลงตารางข้อมูลในฐานข้อมูลธรรมดา อาจมีการเพิ่ม Trigger เข้าไปแต่ไม่มีผลกับการสืบค้นข้อมูล การสืบค้นข้อมูลปกติกระทำบนฐานข้อมูลเดิม เพราะฉะนั้น สมรรถนะไม่เปลี่ยนแปลงจากเดิม

3.1.5 ไม่มี Object ในการจัดการ Constraint เชิงเวลา

การจัดการ Constraint ให้สามารถกระทำผ่านเครื่องมือได้ โดยนักพัฒนาไม่จำเป็นต้องเขียนชุดคำสั่งควบคุมเอง

สร้าง Temporal Uniqueness Constraint ให้นักพัฒนาใช้ได้เลย ไม่จำเป็นต้องเขียนชุดคำสั่งส่วนนี้เอง

3.1.6 ขอบเขตของระบบไม่ได้ต้องการฐานข้อมูลเชิงเวลาทั้งหมด

เครื่องมือต้องช่วยให้การปรับเปลี่ยนฐานข้อมูลธรรมดาให้เป็นฐานข้อมูลเชิงเวลาได้ โดยให้มีผลกระทบกับแอพพลิเคชันน้อยที่สุด โดยเฉพาะโปรแกรมส่วนที่ไม่จำเป็นต้องเกี่ยวข้องกับข้อมูลเชิงเวลาไม่ควรต้องมีผลกระทบ

การนำฐานข้อมูลเชิงเวลามาใช้ในลักษณะสร้างเพิ่มจากฐานข้อมูลปกติ ทำให้สามารถเลือกใช้ฐานข้อมูลเชิงเวลาได้เฉพาะส่วนที่ต้องการ และขยายเพิ่มเติมภายหลังได้

3.1.7 ฐานข้อมูลเชิงเวลาเหมาะกับฐานข้อมูลเชิงวัตถุ

เครื่องมือควรพัฒนาบน Relational Database System เพื่อสามารถนำไปใช้ได้ ณ ปัจจุบัน

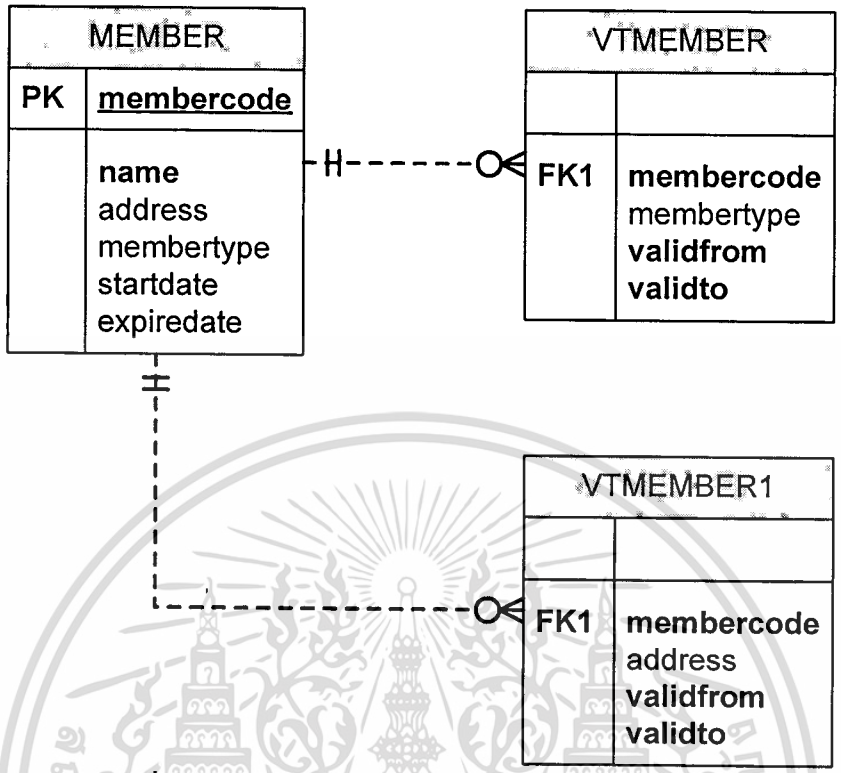
3.2 การพัฒนาเครื่องมือช่วยในการพัฒนาแอพพลิเคชันฐานข้อมูลเชิงเวลา

จากผลสรุปในหัวข้อ 3.1 พบว่าเครื่องมือช่วยในการพัฒนาแอพพลิเคชันเชิงเวลานี้ มีประเด็นต้องพิจารณาคือ

1. ในตารางข้อมูลปกติ 1 ค่าคีย์สามารถมีแถวข้อมูลได้เพียง 1 แถว ในขณะที่ในตารางข้อมูลเชิงเวลาสามารถมีได้มากกว่า 1 แถว ดังนั้น ค่าที่อยู่ในแถวข้อมูลปกติควรเป็นค่าอะไร
2. การปรับปรุงข้อมูลในตารางข้อมูลปกติมีผลอย่างไรต่อตารางข้อมูลเชิงเวลา
3. การปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลามีผลอย่างไรต่อตารางข้อมูลปกติ

3.3 ช่วงชีวิตของแถวข้อมูล (Record Lifespan)

ในตารางข้อมูลปกติ ค่าคีย์หนึ่งมีได้ 1 แถว ซึ่งในขณะที่ตารางข้อมูลเชิงเวลาสามารถมีได้หลายแถว (ช่วงเวลาต่างกัน) หากในตารางข้อมูลเชิงเวลา ค่าคีย์หนึ่งมีชีวิตอยู่และมีค่าเปลี่ยนแปลงไปตามช่วงเวลา (Valid Time Period) ที่ต่างกัน ดังนั้น เราสามารถบอกได้ว่า ข้อมูล 1 แถวในตารางข้อมูลปกตินั้นคือข้อมูล ณ ช่วงชีวิต (Lifespan) หนึ่งของค่าคีย์นั้น



รูปที่ 3.2 ตารางข้อมูลสมาชิกแบบปกติและแบบเชิงเวลา

ในตารางข้อมูลปกติ บางครั้งมีคอลัมน์ที่เป็น User-defined Time ด้วย ถึงแม้คอลัมน์เหล่านั้นจะไม่สามารถเป็น Valid Time Period แต่บางครั้ง คอลัมน์ในตารางข้อมูลปกติ อาจนำเสนอเป็นช่วงชีวิต (Lifespan Period) ของค่าคีย์นั้นๆ ได้ หากดูตารางข้อมูล MEMBER จะเห็นว่า คอลัมน์ startdate และ expiredate สามารถนำเสนอช่วงชีวิตของแต่ละแถวข้อมูลได้

เมื่อสมาชิกสมัครเข้ามาต้องกรอกใบสมัคร เลือกประเภทสมาชิก มีอายุ 1 ปี นับตั้งแต่ startdate จนถึง expiredate

ในปีถัดไป สมาชิกมาต่ออายุ อาจมีการเปลี่ยนประเภทสมาชิก ในตารางข้อมูลปกติแก้ไขข้อมูล expiredate และ membertype แต่ไม่แก้ไข startdate ในตารางข้อมูลเชิงเวลา VTMEMBER เพิ่มแถวข้อมูลใหม่สำหรับปีถัดไปโดยมี membertype ใหม่

จะเห็นว่าช่วงชีวิตของแถวข้อมูลปกติจะยืดยาวไปเรื่อยๆ ตราบเท่าที่สมาชิกคนนี้ต่ออายุไปทุกปี แต่ในตารางข้อมูลเชิงเวลา จำนวนแถวข้อมูลจะเพิ่มขึ้นตามแต่ละช่วงเวลาต่ออายุสมาชิก ดังนั้น ช่วงชีวิตของแถวข้อมูลในตารางข้อมูลปกติไม่ใช่ช่วงเวลาของแถวข้อมูลในตารางข้อมูลเชิงเวลา แต่ในกรณีนี้มีความสัมพันธ์กัน หากไม่ต่ออายุ ช่วงชีวิตของแถวข้อมูลปกติ จะเท่ากับระยะเวลารวมของช่วงเวลาในตารางข้อมูลเชิงเวลาสำหรับสมาชิกคนเดียวกัน

หากกรณีสมาชิกเปลี่ยนที่อยู่ เป็นการใช้ตารางข้อมูลเชิงเวลา VTMEMBER1 แต่ในกรณีนี้เห็นว่า ช่วงชีวิตของแถวข้อมูลใน MEMBER ไม่เกี่ยวข้องอะไรกับช่วงเวลา Valid Time ในตารางข้อมูล VTMEMBER1 เลย

การพิจารณาช่วงชีวิตของแถวข้อมูลปกติมีส่วนในการพิจารณาว่า หากมีการปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลา จะเกิดผลอย่างไรกับตารางข้อมูลปกติ ซึ่งก็คือจะเกิดผลอย่างไรกับช่วงชีวิตของแถวข้อมูลปกตินั่นเอง

บางครั้งในตารางข้อมูลปกติไม่มีการกำหนดคอลัมน์เวลาไว้เลย แต่ใช้วิธีกำหนดคอลัมน์สถานะว่า แถวข้อมูลนี้ใช้ได้ หรือ เลิกใช้แล้ว

การตีความหมายแถวข้อมูลในตารางข้อมูลปกติที่ไม่มีช่วงชีวิต อาจตีความหมายว่า เป็นปัจจุบัน หรือ เป็นค่านิรันดร์

3.3.1 ประเภทช่วงชีวิต

ตารางที่ 3.1 ประเภทช่วงชีวิต

ประเภท	ความหมาย
ค่านิรันดร์	ไม่มีคอลัมน์ที่เกี่ยวกับเวลาที่แสดงถึงช่วงชีวิต สามารถตีความหมายได้ คือ • Time Zero – Infinity ในความหมายทั่วๆ ไป พอเกิดขึ้นแล้ว ใครจะใช้ก็ได้ ใช้ในอดีต ใช้ในอนาคต หรือใช้ในปัจจุบัน • Now – Infinity เมื่อคิดว่าเป็นค่าปัจจุบัน ใช้ในปัจจุบันเท่านั้น (โปรแกรมมีการเพิ่มเฉพาะข้อมูลใหม่ ไม่มีการแก้ไขข้อมูลที่ผ่านไปแล้ว) หากใช้กับคอลัมน์สถานะ ทำให้มีความหมายเป็น 2 ทางแต่ช่วงชีวิตก็ยังคิดใน 2 ความหมายนั้น
ช่วงชีวิตเริ่มต้น ณ (Lifespan From)	มี คอลัมน์เวลา และแสดงความหมายว่า ช่วงชีวิตของแถวข้อมูลเริ่มต้นเมื่อไร เช่น วันที่เริ่มต้นสมาชิก, วันที่สมัคร, วันที่กรอกเอกสาร แสดงความหมายช่วงชีวิตคือ • Lifespan From – Infinity โดยเวลาเริ่มต้น สามารถเป็นเวลาในอดีต ณ ปัจจุบัน หรือ ในอนาคต
ช่วงชีวิตสิ้นสุด ณ (Lifespan To)	มีคอลัมน์เวลาและแสดงความหมายว่า ช่วงชีวิตของแถวข้อมูลนี้สิ้นสุดเมื่อไร เช่น วันที่หมดอายุสมาชิก, วันที่สิ้นสุดสภาพ แสดงความหมายช่วงชีวิตคือ • Time Zero – Lifespan To โดยเวลาสิ้นสุดสามารถเป็นอดีต ปัจจุบัน หรือ อนาคต
ช่วงชีวิต (Lifespan Period)	มีคอลัมน์เวลา 2 คอลัมน์ และสามารถสื่อความหมายเป็นช่วงชีวิตของแถวข้อมูลปกติได้ Lifespan From – Lifespan To โดยเวลาสามารถเป็นอดีต ปัจจุบัน หรือ อนาคต และ ช่วงชีวิตเริ่มต้น ต้องน้อยกว่า ช่วงชีวิตสิ้นสุดเสมอ

3.4 การปรับปรุงข้อมูลในตารางข้อมูลปกติ

ตารางข้อมูลปกติ และ ตารางข้อมูลเชิงเวลา มักมีคอลัมน์ข้อมูลร่วมกันเป็นบางคอลัมน์ แต่ไม่ใช่ทุกคอลัมน์ ดังนั้น การปรับปรุงข้อมูลจากตารางข้อมูลปกติมีผลไปยังตารางข้อมูลเชิงเวลาต้องระวังไม่ให้ปรับปรุงไม่ได้ ลองพิจารณาคำสั่งที่ใช้ในการปรับปรุงดังนี้

3.4.1 INSERT

การเพิ่มข้อมูลเข้าไปในตารางข้อมูลปกติ เป็นจุดเริ่มที่บอกว่า ค่าคีย์นี้เกิดขึ้นแล้วในระบบจะต้องไปเพิ่มข้อมูลในตารางข้อมูลเชิงเวลาด้วย

ช่วงเวลาของข้อมูลในตารางข้อมูลเชิงเวลาให้กำหนดจากช่วงชีวิตของแถวข้อมูลนั้นๆ ดังนั้น ความหมายของช่วงชีวิตของแถวข้อมูลในตารางข้อมูลปกติต้องนิยามให้ชัดเจน

หากไม่สามารถเพิ่มข้อมูลเข้าไปในตารางข้อมูลเชิงเวลาได้ แสดงว่า ค่าคีย์นั้นมีอยู่แล้ว ต้องไม่สามารถเพิ่มข้อมูลที่ตารางข้อมูลปกติได้เช่นกัน นั่นคือ หากในตารางข้อมูลเชิงเวลามีข้อมูลคีย์นี้อยู่ ในฐานข้อมูลปกติก็ต้องมีแถวข้อมูลของคีย์นี้อยู่เช่นกัน

3.4.2 DELETE

การลบในแอพพลิเคชันปกติคือ แถวข้อมูลนั้น ไม่มีอีกต่อไป เพราะแอพพลิเคชันจะไม่มีการอ้างอิงใช้แถวข้อมูลนั้นอีก เหมือนไม่มีตัวตนอีกแล้ว คีย์ที่แสดงความเป็นตัวตนของแถวข้อมูลนั้นก็หายไปด้วย ระบบสามารถกำหนดผลกระทบได้ 2 ลักษณะคือ

อนุญาตให้ลบได้ หากใช้ลักษณะนี้ ในตารางข้อมูลเชิงเวลาก็ไม่มีประโยชน์ที่รักษาข้อมูลเชิงเวลาของคีย์นี้ไว้เพราะ ไม่สามารถอ้างอิงใช้กลับมายังตารางข้อมูลปกติได้อีก ดังนั้นควรให้ลบข้อมูลคีย์นี้ออกจากตารางข้อมูลเชิงเวลาทั้งหมด

อีกลักษณะหนึ่งคือไม่อนุญาตให้ลบ หากเป็นลักษณะนี้ ก็ไม่ต้องทำอะไร เพราะ Foreign Key Constraint ที่กำหนดไว้แล้ว จะห้ามไม่ให้แอพพลิเคชันปกติลบข้อมูลปกติที่มีข้อมูลเชิงเวลาผูกอยู่ด้วย

3.4.3 UPDATE

การแก้ไขค่าในคอลัมน์ในตารางข้อมูลปกติ หากคอลัมน์นั้นได้ถูกนิยามให้เป็นคอลัมน์เชิงเวลาแล้ว การแก้ไขจำเป็นต้องกระทำจากแอพพลิเคชันเชิงเวลา หรือ กระทำที่ตารางข้อมูลเชิงเวลาและให้ผลการแก้ไขกลับมายังแถวข้อมูลปกติ

การที่เราให้คอลัมน์เป็นคอลัมน์เชิงเวลาเพราะต้องการเก็บประวัติการเปลี่ยนแปลง การแก้ไขในตารางข้อมูลปกติ ทำให้ไม่สามารถเก็บค่าความเปลี่ยนแปลงได้ จึงต้องห้ามไว้

ข้อควรระวังคือ ต้องห้ามแก้ไขเฉพาะคอลัมน์ที่นิยามเป็นคอลัมน์เชิงเวลาเท่านั้น คอลัมน์อื่นๆ ต้องสามารถแก้ไขได้ และระวังว่า หากมีการแก้ไขมาจากแอพพลิเคชันเชิงเวลา (ปรับปรุงค่าหลังแก้ไขไปในตารางข้อมูลเชิงเวลาแล้ว) ต้องอนุญาตให้แก้ไขได้ด้วย

3.5 การปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลา

เมื่อมีการปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลา ผลของการแก้ไขนั้น ต้องมีการพิจารณาว่าจะส่งกลับมายังตารางข้อมูลปกติหรือไม่ หลักในการพิจารณาคืออาศัยนิยามช่วงชีวิตของแถวข้อมูลปกตินั่นเอง

แอพพลิเคชันสั่งปรับปรุงข้อมูลเชิงเวลาได้ โดยการส่งคำสั่งเป็น SQL เชิงเวลาผ่าน Temporal Engine เพื่อให้ Temporal Engine แปลคำสั่งให้เป็น Transact-SQL ตามหัวข้อ 2.4 และ 2.6

หลังจากแก้ไขข้อมูลแล้ว โดยปกติ ระบบทำการตรวจสอบช่วงชีวิตของข้อมูลดังนี้

1. ช่วงชีวิตที่ครอบคลุมปัจจุบันถือเป็น ช่วงชีวิต ปัจจุบัน (เกิดขึ้นแล้วในอดีต และ จบลงในอนาคต)
2. ช่วงชีวิตในอดีต (เคยเกิดขึ้นแล้ว และจบไปแล้ว)
3. ช่วงชีวิตในอนาคต (ยังไม่เกิดขึ้น แต่มีอยู่ในตารางข้อมูลเชิงเวลา)

หากพบข้อมูลที่ช่วงชีวิตใด จะใช้ค่าของช่วงชีวิตนั้นไปแก้ไขแถวข้อมูลของคีย์นั้นในตารางข้อมูลปกติ

อย่างไรก็ตาม หากตารางข้อมูลปกติมีคอลัมน์สถานะ และต้องการผูกสถานะกับช่วงชีวิต เช่นมีการยกเลิกในอนาคต เมื่อถึงเวลานั้น คอลัมน์สถานะก็ต้องถูกแก้ไขเป็นยกเลิก หรือ ค่าคีย์ที่กำหนดมีช่วงชีวิตมากกว่า 1 ช่วง พอช่วงชีวิตในอนาคตมาถึงก็ต้องมีการปรับปรุงค่าในแถวข้อมูลปกติให้เป็นไปตามช่วงชีวิตอนาคต การปรับข้อมูลในอนาคตจะกระทำโดย Temporal Agent ซึ่งจะอธิบายในหัวข้อถัดไป

3.6 เอเจนท์เชิงเวลา (Temporal Agent)

Temporal Agent เป็นโปรแกรมที่ทำงานโดยอัตโนมัติอยู่บน Microsoft SQL Server เพื่อทำหน้าที่ปรับปรุงค่าในแถวข้อมูลปกติให้เป็นไปตามช่วงชีวิตที่ถูกต้องตามค่าในตารางข้อมูลเชิงเวลา

หลักการในการทำงานคือ

เมื่อมีการปรับปรุงข้อมูลในตารางข้อมูลเชิงเวลา ระบบจะบันทึกค่าคีย์ของแถวข้อมูลที่มีการปรับปรุง

ระบบไปตรวจในตารางข้อมูลเชิงเวลาว่า ค่าคีย์นี้มีช่วงชีวิตที่จะใช้ในแถวข้อมูลปกติเป็นอย่างไร หากเป็นปัจจุบัน ให้ดูว่า ช่วงชีวิตจบ ณ จุดเวลาใด ซึ่งต้องเป็นจุดเวลาในอนาคต ให้บันทึกจุดเวลานี้ไว้ หากเป็นอนาคต ให้ดูว่า ช่วงชีวิตเริ่มต้น ณ จุดเวลาใด ให้บันทึกจุดเวลานั้นไว้ แต่หากเป็นช่วงชีวิตอดีต ไม่ต้องบันทึกอะไร

สร้างตารางการทำงานของเอเจนท์ใน Microsoft SQL Server ให้ทำงาน ณ จุดเวลาที่บันทึกไว้

เมื่อถึงเวลานั้น เอเจนท์จะทำงาน ทำการตรวจสอบช่วงชีวิตของค่าคีย์นี้ใหม่ และปรับปรุงข้อมูลของค่าคีย์นี้ในแถวข้อมูลปกติ

3.7 เอนจิ้นเชิงเวลา (Temporal Engine) และ Temporal Manager

เป็นซอฟต์แวร์คอมโปแนนท์พัฒนาโดยใช้ Microsoft .NET Framework เอนจิ้นเชิงเวลาเป็นส่วนประมวลผลคำสั่งเชิงเวลาทั้งหมดและควบคุมความถูกต้องของข้อมูลเชิงเวลาในฐานข้อมูล ส่วน Temporal Manager เป็นโปรแกรมที่ให้นักพัฒนาใช้ในการจัดการฐานข้อมูลเชิงเวลา Temporal Manager เปรียบเสมือนส่วน Presentation Layer ของโปรแกรม ส่วนที่ทำงานคือ Temporal Engine

นอกจากนี้ นักพัฒนาสามารถพัฒนาแอพพลิเคชันเชิงเวลาขึ้น ซึ่งสามารถใช้งาน Temporal Engine เพื่อใช้งานฐานข้อมูลเชิงเวลาได้

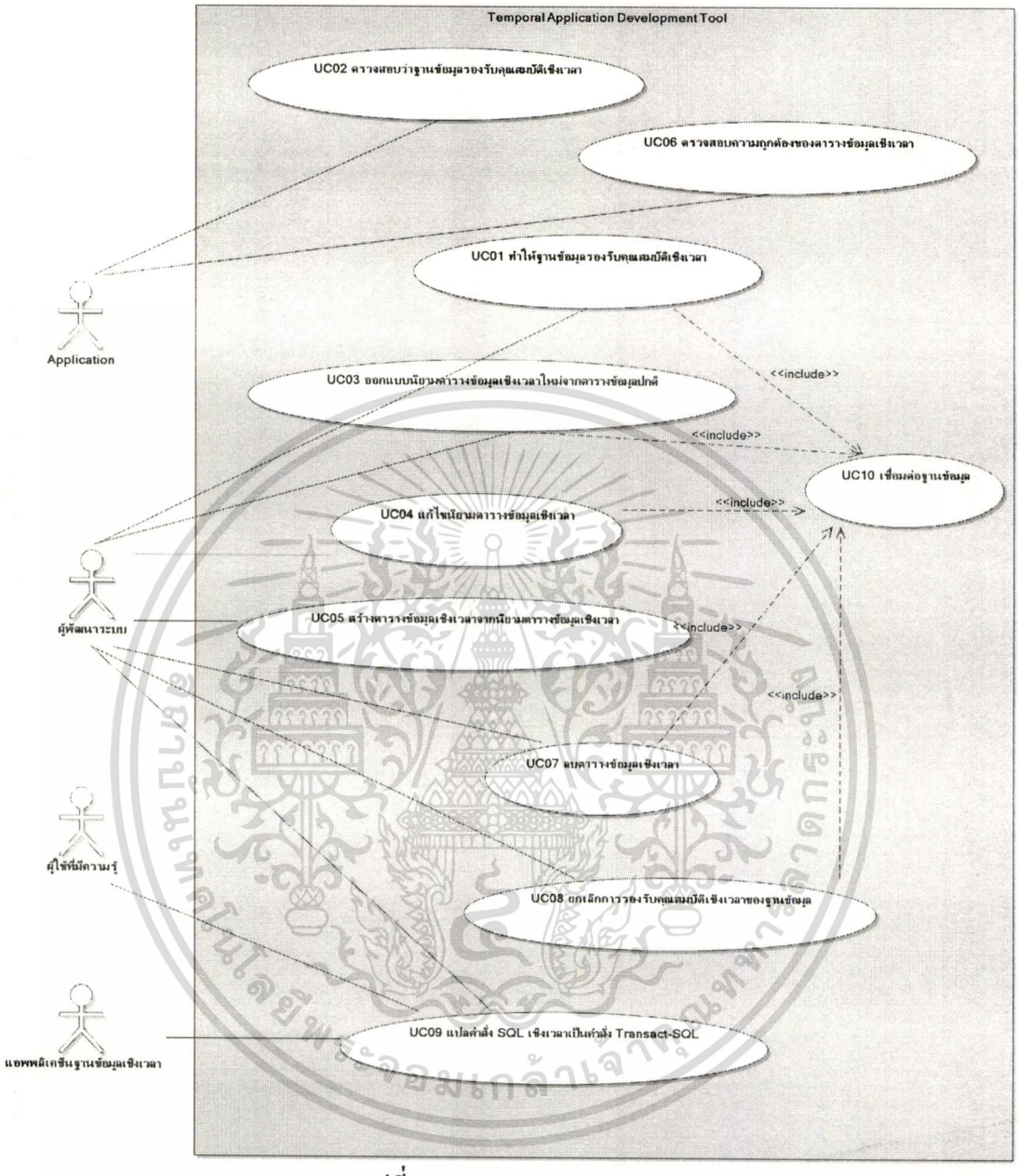
บทที่ 4
การวิเคราะห์ระบบ

4.1 Use Case

จำนวน Use Case ของซอฟต์แวร์เครื่องมือมีดังนี้

ตารางที่ 4.1 รายชื่อ Use Case

เลขที่	ชื่อ Use Case
UC01	ทำให้ฐานข้อมูลรองรับคุณสมบัติเชิงเวลา
UC02	ตรวจสอบว่าฐานข้อมูลรองรับคุณสมบัติเชิงเวลา
UC03	ออกแบบนิยามตารางข้อมูลเชิงเวลาใหม่จากตารางข้อมูลปกติ
UC04	แก้ไขนิยามตารางข้อมูลเชิงเวลา
UC05	สร้างตารางข้อมูลเชิงเวลาจากนิยามตารางข้อมูลเชิงเวลา
UC06	ตรวจสอบความถูกต้องของตารางข้อมูลเชิงเวลา
UC07	ลบตารางข้อมูลเชิงเวลา
UC08	ยกเลิกการรองรับคุณสมบัติเชิงเวลาของฐานข้อมูล
UC09	แปลคำสั่ง SQL เชิงเวลาเป็นคำสั่ง Transact-SQL



รูปที่ 4.1 Use Case Diagram

4.1.1 UC01 ทำให้ฐานข้อมูลรองรับคุณสมบัติเชิงเวลา

Actor

ผู้พัฒนาระบบ

รายละเอียด

ปรับฐานข้อมูลปกติให้สามารถรองรับคุณสมบัติเชิงเวลาได้

เงื่อนไขก่อน

ฐานข้อมูลที่ปรับต้องยังไม่เคยปรับมาก่อน

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ Uses UC10 เชื่อมต่อฐานข้อมูล
2. ผู้พัฒนาเลือกว่า ต้องการใช้ฐานข้อมูลใด
3. ระบบตรวจว่า ฐานข้อมูลนี้พร้อมที่จะทำการปรับให้รองรับคุณสมบัติเชิงเวลา
4. ผู้พัฒนายืนยันการปรับ
5. ระบบทำการสร้างออปเจกต์ที่จำเป็นเพื่อให้ฐานข้อมูลรองรับคุณสมบัติเชิงเวลาได้
6. ระบบแจ้งให้ผู้พัฒนาระบบทราบ

เงื่อนไขหลัง

ฐานข้อมูลมีออปเจกต์เพื่อรองรับคุณสมบัติเชิงเวลาครบ

ขั้นตอนเหตุการณ์สำรอง

ข้อ 4 หากไม่พร้อม เช่น ถูกปรับไปแล้ว หรือ ถูกปรับแล้ว แต่มีออปเจกต์ที่จำเป็นไม่ครบ ระบบจะไม่ยอมให้ทำการปรับต่อ

4.1.2 UC02 ตรวจสอบว่าฐานข้อมูลรองรับคุณสมบัติเชิงเวลา

Actor

แอพพลิเคชัน

รายละเอียด

ตรวจสอบจำนวนออปเจกต์ในฐานข้อมูลว่ามีครบถ้วนหรือไม่

เงื่อนไขก่อน

มีฐานข้อมูล

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ แอพพลิเคชันทำการเชื่อมต่อไปยังเซิร์ฟเวอร์ฐานข้อมูล โดยใช้ข้อมูล: Server Name, Authentication Type, (Login Name), (Password)
2. เมื่อเชื่อมต่อเซิร์ฟเวอร์ฐานข้อมูลได้แล้ว ระบบดึงชื่อฐานข้อมูลออกมาจากเซิร์ฟเวอร์
3. แอพพลิเคชันขอให้ระบบตรวจสอบว่าฐานข้อมูลนั้นรองรับการใช้งานฐานข้อมูลเชิงเวลาหรือไม่
4. ระบบตรวจสอบว่า ฐานข้อมูลนั้นรองรับการใช้งานฐานข้อมูลเชิงเวลาหรือไม่
5. ระบบแจ้งให้แอพพลิเคชันทราบ

เงื่อนไขหลัง

-

ขั้นตอนเหตุการณ์สำรอง

-

4.1.3 UC03 ออกแบบนิยามตารางข้อมูลเชิงเวลาใหม่จากตารางข้อมูลปกติ

Actor

ผู้พัฒนาระบบ

รายละเอียด

เพื่อให้ผู้พัฒนาระบบสามารถออกแบบตารางข้อมูลเชิงเวลาได้

เงื่อนไขก่อน

ฐานข้อมูลต้องรองรับคุณสมบัติเชิงเวลาแล้ว

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ Uses UC10 เชื่อมต่อฐานข้อมูล
2. ผู้พัฒนาเลือกว่า ต้องการใช้ฐานข้อมูลใด
3. ระบบดึงชื่อของตารางที่อยู่ในฐานข้อมูลนั้นๆออกมา พร้อมตรวจสอบว่า ฐานข้อมูลนั้นรองรับการใช้งานฐานข้อมูลเชิงเวลา
4. ผู้พัฒนาเลือกว่า ต้องการสร้างตารางข้อมูลเชิงเวลาบนตารางข้อมูลใด
5. ระบบดึงชื่อคอลัมน์, ประเภทข้อมูล, ความยาวข้อมูล, คุณสมบัติ Null/Not Null ของคอลัมน์ในตารางนั้นออกมาแสดงให้ผู้พัฒนาเห็น รวมทั้ง Primary Key, Foreign Key, และ Unique Constraints ทั้งหมดที่ติดอยู่กับตารางนี้ออกมาด้วย
6. ผู้พัฒนาสามารถเลือกคอลัมน์ใดๆในตารางเพื่อสร้างให้เป็นคอลัมน์เชิงเวลา
7. ผู้พัฒนาเลือกว่า คอลัมน์ใด เป็นช่วงชีวิต คอลัมน์สถานะ และค่าของคอลัมน์สถานะ
8. ผู้พัฒนาสามารถเพิ่มคอลัมน์เข้าไปในตารางข้อมูลเชิงเวลาได้
9. ผู้พัฒนาสั่งให้ระบบบันทึกนิยาม
10. ระบบตรวจสอบว่านิยามถูกต้องตามกฎ
11. ระบบบันทึกนิยามตารางข้อมูลเชิงเวลาไปยังตารางข้อมูลระบบและแจ้งผลการบันทึกให้ผู้พัฒนาระบบ

เงื่อนไขหลัง

นิยามตารางข้อมูลเชิงเวลาใหม่ถูกบันทึกลงฐานข้อมูล

ขั้นตอนเหตุการณ์สำรอง

ข้อ 4 หากฐานข้อมูลนั้นไม่รองรับการใช้งานฐานข้อมูลเชิงเวลา ระบบไม่อนุญาตให้สร้างนิยาม

ข้อ 8 อาจไม่จำเป็นต้องเลือกก็ได้ หากตารางข้อมูลนั้นไม่มีข้อมูลช่วงชีวิต หรือคอลัมน์สถานะ

4.1.4 UC04 แก้ไขนิยามตารางข้อมูลเชิงเวลา

Actor

ผู้พัฒนาระบบ

รายละเอียด

แก้ไขนิยามตารางข้อมูลเชิงเวลาที่ได้สร้างไว้แล้ว

เงื่อนไขก่อน

นิยามตารางข้อมูลเชิงเวลาต้องยังไม่ถูกสร้างเป็นตารางข้อมูลเชิงเวลาในฐานข้อมูล

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ Uses UC10 เชื่อมต่อฐานข้อมูล
2. ผู้พัฒนาเลือกว่า ต้องการใช้ฐานข้อมูลใด
3. ระบบดึงชื่อของตารางที่อยู่ในฐานข้อมูลนั้นๆออกมา พร้อมตรวจสอบว่า ฐานข้อมูลนั้นรองรับการใช้งานฐานข้อมูลเชิงเวลา และแสดงข้อมูลนิยามตารางข้อมูลเชิงเวลา
4. ผู้พัฒนาเลือกนิยามตารางข้อมูลเชิงเวลาที่ต้องการแก้ไข
5. ระบบตรวจสอบว่า นิยามนั้นถูกสร้างเป็นตารางจริงหรือยัง
6. ระบบดึงข้อมูลนิยามตารางข้อมูลเชิงเวลาออกมา
7. ผู้พัฒนาปรับปรุงแก้ไขนิยาม
8. ผู้พัฒนาสั่งให้ระบบบันทึกนิยาม
9. ระบบตรวจสอบว่านิยามถูกต้องตามกฎ
10. ระบบบันทึกนิยามตารางข้อมูลเชิงเวลาไปยังตารางข้อมูลระบบและแจ้งผลการบันทึกให้ผู้พัฒนาระบบ

เงื่อนไขหลัง

นิยามตารางข้อมูลเชิงเวลาถูกแก้ไข

ขั้นตอนเหตุการณ์สำรอง

ข้อ 6 หากถูกสร้างแล้ว ระบบไม่สามารถสร้างได้

4.1.5 UC05 สร้างตารางข้อมูลเชิงเวลาจากนิยามตารางข้อมูลเชิงเวลา

Actor

ผู้พัฒนาระบบ

รายละเอียด

สร้างตารางข้อมูลเชิงเวลาจากนิยามที่สร้างไว้แล้ว

เงื่อนไขก่อน

ฐานข้อมูลต้องรองรับคุณสมบัติเชิงเวลา และนิยามถูกสร้างไว้แล้ว

ขั้นตอนเหตุการณ์

1. Use Case เริ่มต้นเมื่อ ผู้พัฒนาเลือกนิยามตารางข้อมูลเชิงเวลาที่ต้องการสร้าง
2. ผู้พัฒนาสั่งให้ระบบสร้างตารางข้อมูลเชิงเวลา
3. ระบบตรวจสอบว่าตารางข้อมูลเชิงเวลายังไม่มีในฐานข้อมูล
4. ระบบอ่านข้อมูลมาจากตารางข้อมูลระบบ
5. ระบบสร้างตารางข้อมูลเชิงเวลา
6. ระบบสร้าง Trigger บนตารางข้อมูลปกติ
7. ระบบสร้างและปรับปรุง Stored Procedure ที่เกี่ยวข้องกับตารางข้อมูลเชิงเวลา
8. ระบบแจ้งผลการสร้างให้ผู้พัฒนาทราบ

เงื่อนไขหลัง

ตารางข้อมูลเชิงเวลาถูกสร้างขึ้นในฐานข้อมูลพร้อมทั้งออปเจกต์อื่นๆที่เกี่ยวข้อง

ขั้นตอนเหตุการณ์สำรอง

ข้อ 3 หากระบบพบว่า มีตารางข้อมูลเชิงเวลานั้นในฐานข้อมูลแล้ว ยกเลิกการทำงาน

4.1.6 UC06 ตรวจสอบความถูกต้องของตารางข้อมูลเชิงเวลา

Actor

แอพพลิเคชัน

รายละเอียด

เพื่อตรวจสอบว่า ตารางข้อมูลเชิงเวลามีออปเจกต์ที่จำเป็นครบถ้วน

เงื่อนไขก่อน

มีตารางข้อมูลเชิงเวลาอยู่ในระบบแล้ว

ขั้นตอนเหตุการณ์

1. Use Case เริ่มต้นเมื่อ แอพพลิเคชันสั่งให้ระบบตรวจสอบตารางข้อมูลเชิงเวลา
2. ระบบทำการตรวจสอบจำนวนออปเจกต์
3. ระบบแจ้งผลการตรวจสอบให้ผู้พัฒนาทราบ

เงื่อนไขหลัง

-

ขั้นตอนเหตุการณ์สำรอง

-

4.1.7 UC07 ลบตารางข้อมูลเชิงเวลา

Actor

ผู้พัฒนาระบบ

รายละเอียด

เพื่อเลิกใช้ตารางข้อมูลเชิงเวลาที่มีอยู่แล้ว

เงื่อนไขก่อน

มีตารางข้อมูลเชิงเวลาอยู่ในระบบแล้ว

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ Uses UC10 เชื่อมต่อฐานข้อมูล
2. ผู้พัฒนาเลือกตารางข้อมูลเชิงเวลาที่ต้องการลบออกจากระบบ
3. ผู้พัฒนาสั่งลบตารางข้อมูลเชิงเวลา
4. ระบบขอคำยืนยันจากผู้พัฒนา โดยแจ้งว่า การลบตารางข้อมูลเชิงเวลามีผลให้ข้อมูลเชิงเวลาทั้งหมดของตารางนั้นหายไปและไม่สามารถกู้กลับมาได้อีก
5. ผู้พัฒนายืนยันการลบ
6. ระบบทำการลบตารางข้อมูล
7. ระบบแจ้งผลการลบให้ผู้พัฒนาทราบ

เงื่อนไขหลัง

ตารางข้อมูลเชิงเวลาถูกลบออกไปจากฐานข้อมูล

ขั้นตอนเหตุการณ์สำรอง

ข้อ 4 หากผู้พัฒนาไม่ยืนยัน ระบบยกเลิกการทำงาน

4.1.8 UC08 ยกเลิกการรองรับคุณสมบัติเชิงเวลาของฐานข้อมูล

Actor

ผู้พัฒนาระบบ

รายละเอียด

เพื่อยกเลิกคุณสมบัติเชิงเวลาของฐานข้อมูล

เงื่อนไขก่อน

-

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ Uses UC10 เชื่อมต่อฐานข้อมูล
2. ผู้พัฒนาเลือกว่า ต้องการยกเลิกคุณสมบัติเชิงเวลาของฐานข้อมูลใด
3. ระบบขอคำยืนยันจากผู้พัฒนา โดยแจ้งว่า ข้อมูลเชิงเวลาทั้งหมดของฐานข้อมูลจะสูญไปและไม่สามารถกู้กลับมาได้
4. ผู้พัฒนายืนยัน
5. ระบบทำการลบออปเจกต์เชิงเวลาทั้งหมดในฐานข้อมูล
6. ระบบแจ้งผลการทำงานให้ผู้พัฒนาทราบ

เงื่อนไขหลัง

ฐานข้อมูลไม่มีออปเจกต์เชิงเวลา และไม่รองรับคุณสมบัติเชิงเวลา

ขั้นตอนเหตุการณ์สำรอง

ข้อ 4 หากผู้พัฒนาไม่ยืนยัน ระบบยกเลิกการทำงาน

4.1.9 UC09 แปลคำสั่ง SQL เชิงเวลาเป็นคำสั่ง Transact-SQL

Actor

ผู้พัฒนาระบบ ผู้ใช้ที่มีความรู้ แอพพลิเคชันฐานข้อมูลเชิงเวลา

รายละเอียด

รับคำสั่ง SQL เชิงเวลาเพื่อเปลี่ยนเป็น Transact-SQL และส่งไปยัง DBMS เพื่อประมวลผล

เงื่อนไขก่อน

ผู้พัฒนาให้ข้อมูลการเชื่อมต่อกับ DBMS กับระบบเพื่อระบบสามารถเชื่อมต่อ DBMS เพื่อส่งคำสั่งไปประมวลผลได้

ขั้นตอนเหตุการณ์

1. Use Case เริ่มต้นเมื่อ Actor ส่งคำสั่ง SQL มายังระบบ
2. ระบบตรวจว่าคำสั่ง SQL นั้นเป็นคำสั่งเชิงเวลา
3. ระบบแปลคำสั่งเชิงเวลาให้เป็น Transact-SQL ตามกฎที่สร้างไว้
4. ส่งคำสั่ง Transact-SQL ที่แปลแล้วไปทำงานที่ DBMS
5. ระบบผลจาก DBMS และส่งกลับให้กับ Actor

เงื่อนไขหลัง

Actor ได้รับผลจากคำสั่ง SQL เชิงเวลา

ขั้นตอนเหตุการณ์สำรอง

ข้อ 2 หากระบบพบว่าเป็นคำสั่ง Transact-SQL จะข้ามขั้นตอนการแปลและส่งคำสั่งไปยัง DBMS เลย

4.1.10 UC10 เชื่อมต่อฐานข้อมูล

Actor

ผู้พัฒนาระบบ

รายละเอียด

เพื่อติดต่อฐานข้อมูล

เงื่อนไขก่อน

-

ขั้นตอนเหตุการณ์

1. Use Case นี้เริ่มต้นเมื่อ ผู้พัฒนาระบบทำการเชื่อมต่อไปยังเซิร์ฟเวอร์ฐานข้อมูล โดยใช้ข้อมูล: Server Name, Authentication Type, (Login Name), (Password)
2. เมื่อเชื่อมต่อเซิร์ฟเวอร์ฐานข้อมูลได้แล้ว ระบบดึงชื่อฐานข้อมูลออกมาจากเซิร์ฟเวอร์

เงื่อนไขหลัง

เชื่อมต่อฐานข้อมูล

ขั้นตอนเหตุการณ์สำรอง

-

4.2 ความต้องการอื่นๆ

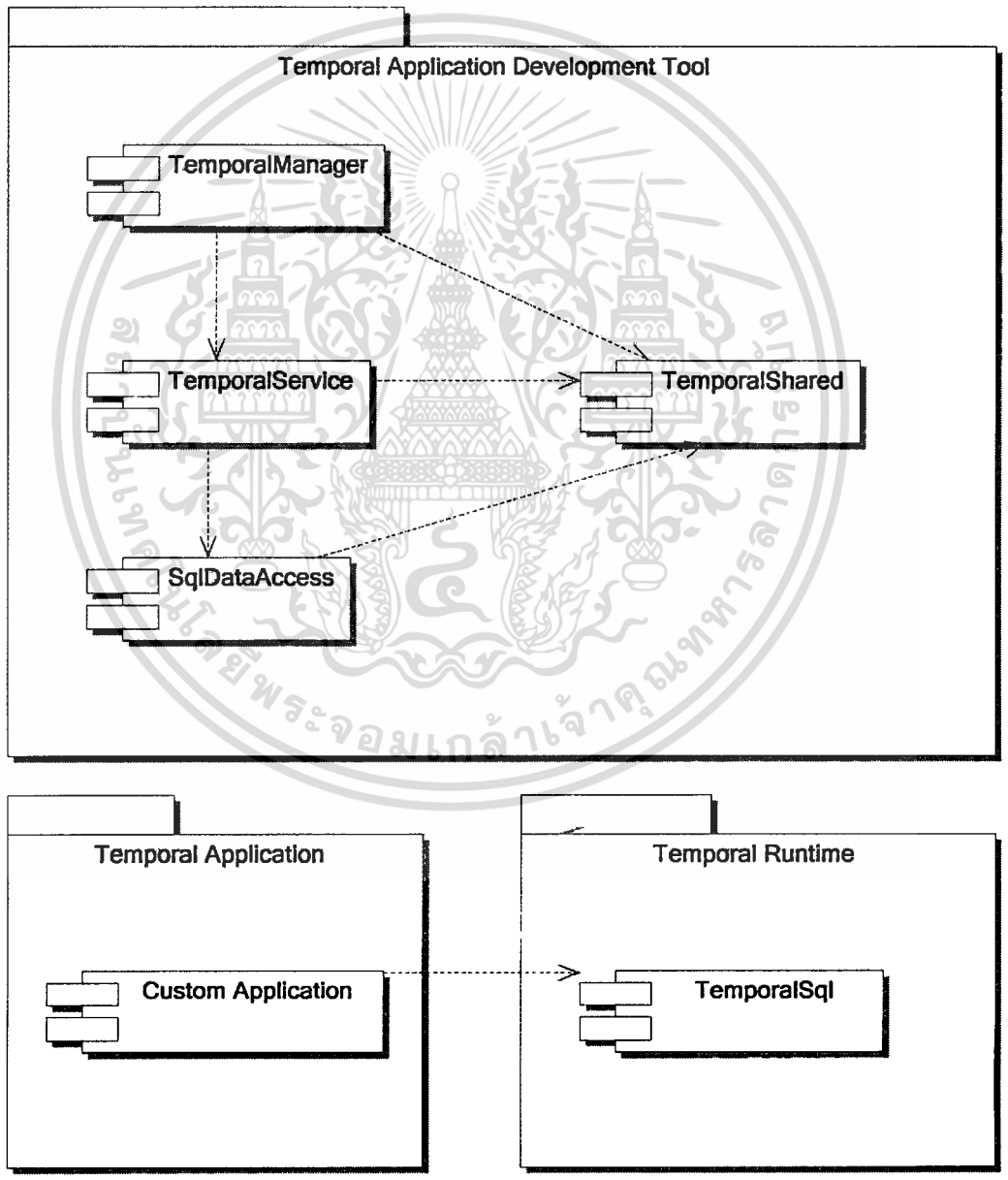
หนึ่งตารางข้อมูลปกติสามารถสร้างเป็นตารางข้อมูลเชิงเวลาได้มากกว่าหนึ่ง ขึ้นอยู่กับว่าชุดของคอลัมน์ที่ต้องการเปลี่ยนเป็นคอลัมน์เชิงเวลามีเท่าไร แต่คอลัมน์หนึ่งๆสามารถอยู่ในตารางข้อมูลเชิงเวลาได้เพียงตารางเดียวเท่านั้น

การเก็บข้อมูลเชิงเวลา บางครั้งการเปลี่ยนแปลงใดๆ จำเป็นต้องมีข้อมูลประกอบเพิ่ม เช่น เลขที่เอกสาร ดังนั้น ตารางข้อมูลเชิงเวลาควรสามารถเพิ่มคอลัมน์ที่ไม่มีอยู่ในตารางข้อมูลปกติได้

บทที่ 5

การออกแบบระบบ

5.1 Component Model

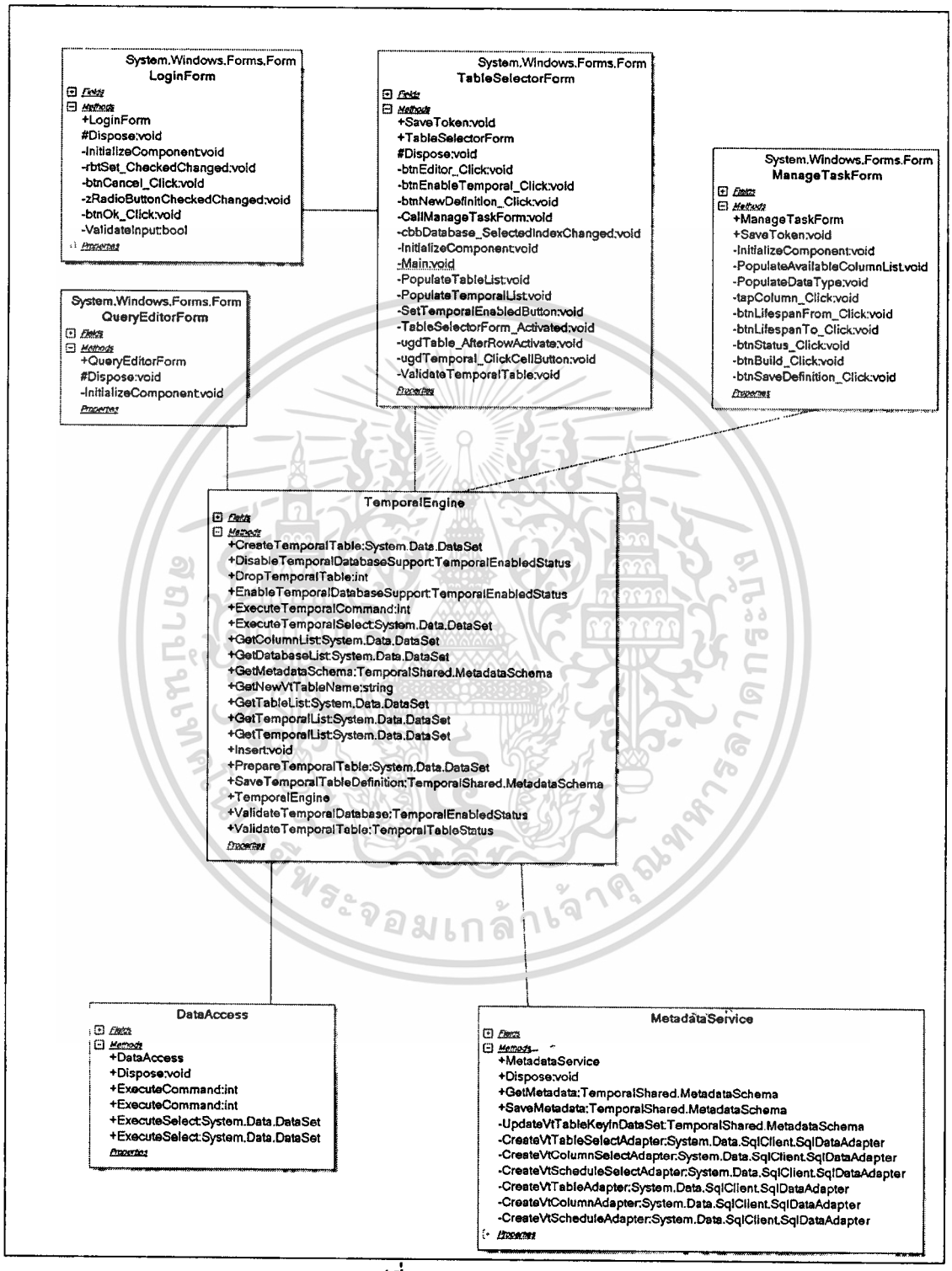


รูปที่ 5.1 Component Diagram

Temporal Application Development Tool ประกอบด้วย

TemporalManager Application	Presentation Layer เป็นวินโดว์สแอพพลิเคชันที่เขียนโดยใช้ภาษา C#
TemporalService Component	Business Layer ทำหน้าที่ประมวลคำสั่งในการจัดการฐานข้อมูลเชิงเวลา และแปลคำสั่งภาษา SQL เชิงเวลาให้เป็น Transact-SQL ก่อนส่งไปให้ Microsof t SQL Server ประมวลผลต่อ เป็น In-memory DLL Component เขียนโดยใช้ภาษา C# ทำงานบน Microsoft .NET Framework
TemporalSql Component	เป็น Runtime version ของ TemporalService ทำหน้าที่แปลคำสั่ง tSQL ให้เป็น Transact-SQL
SqlDataAccess Component	Data Access Layer ทำหน้าที่ติดต่อกับ Microsoft SQL Server ส่งคำสั่ง และรับผลที่ได้ส่งให้กับ Business Layer
TemporalShared Component	เป็น Private Component ที่เก็บโครงสร้างข้อมูลและค่าคงที่ที่ใช้ในระบบ
SQL Server Stored Procedures และ Triggers	ระบบจำเป็นต้องใช้เซิร์ฟเวอร์ออปเจกต์ใน Microsof t SQL Server

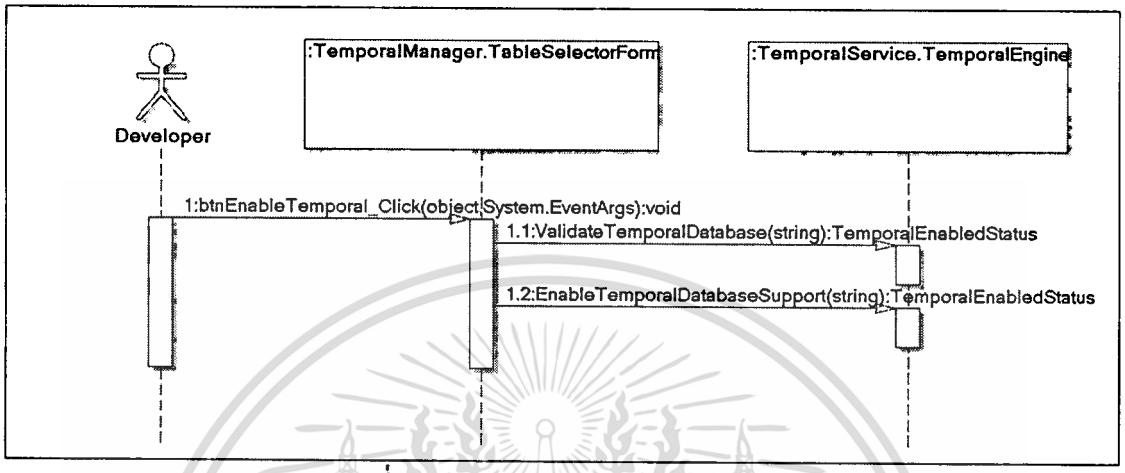
5.2 Class Diagram



รูปที่ 5.2 Class Diagram

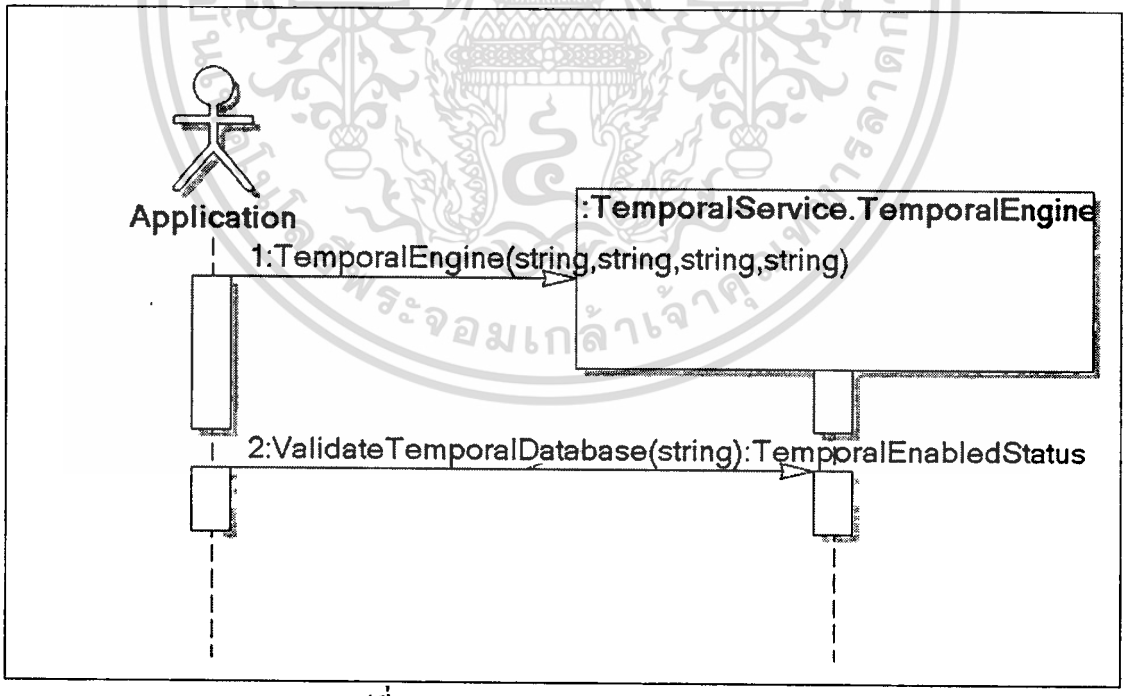
5.3 Sequence Diagram

5.3.1 UC01 ทำให้ฐานข้อมูลรองรับคุณสมบัติเชิงเวลา



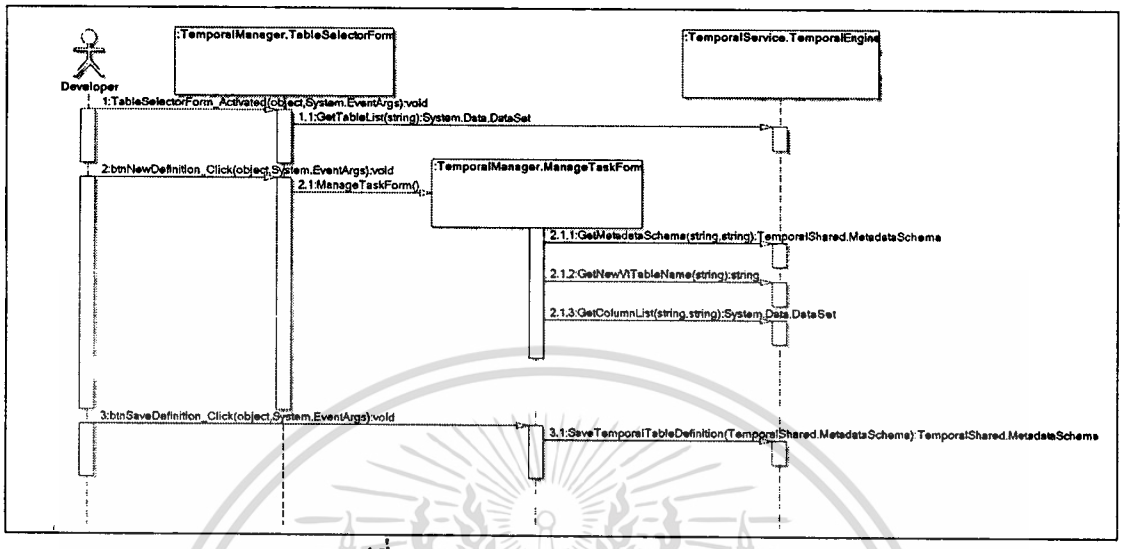
รูปที่ 5.3 Sequence Diagram ของ UC01

5.3.2 UC02 ตรวจสอบว่าฐานข้อมูลรองรับคุณสมบัติเชิงเวลา



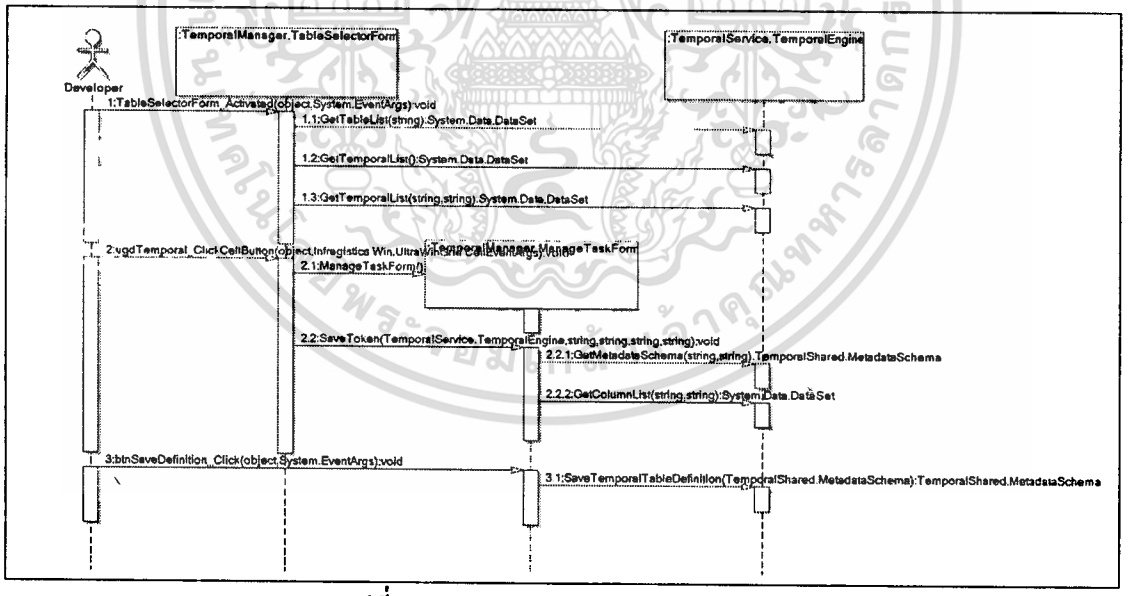
รูปที่ 5.4 Sequence Diagram ของ UC02

5.3.3 UC03 ออกแบบนิยามตารางข้อมูลเชิงเวลาใหม่จากตารางข้อมูลปกติ



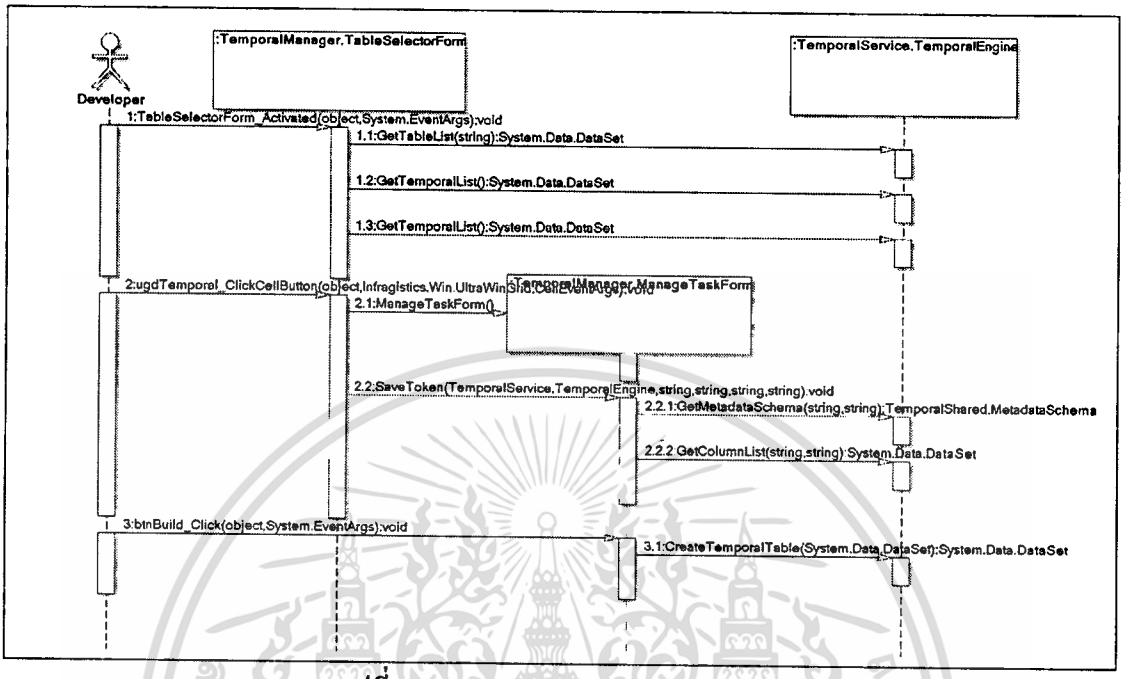
รูปที่ 5.5 Sequence Diagram ของ UC03

5.3.4 UC04 แก้ไขนิยามตารางข้อมูลเชิงเวลา



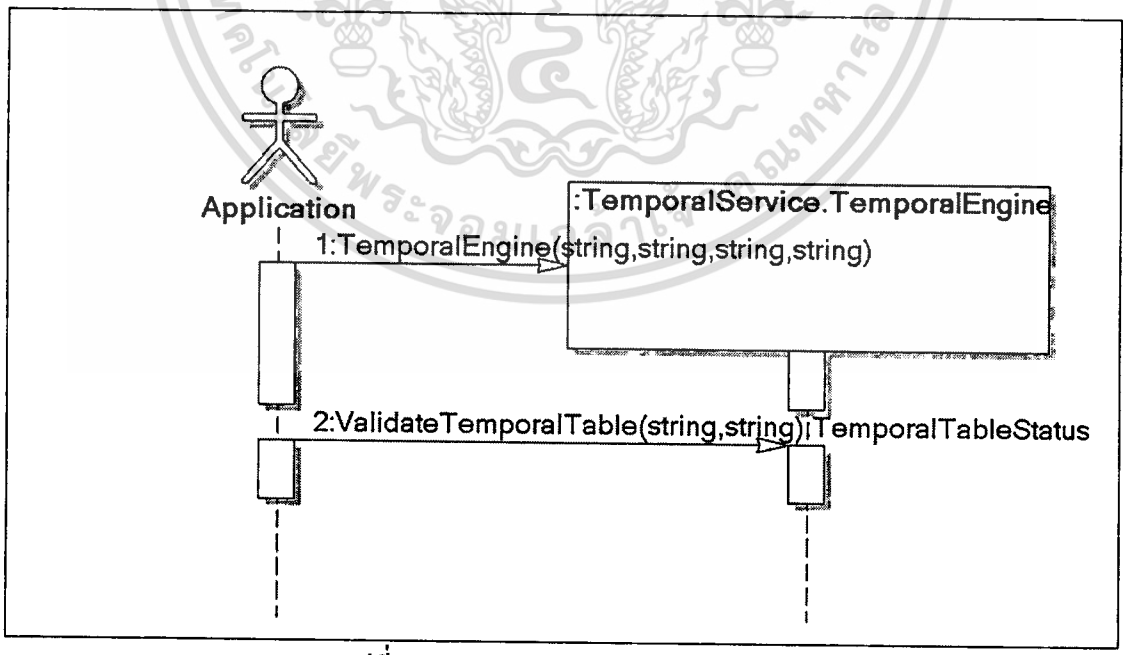
รูปที่ 5.6 Sequence Diagram ของ UC04

5.3.5 UC05 สร้างตารางข้อมูลเชิงเวลาจากนิยามตารางข้อมูลเชิงเวลา



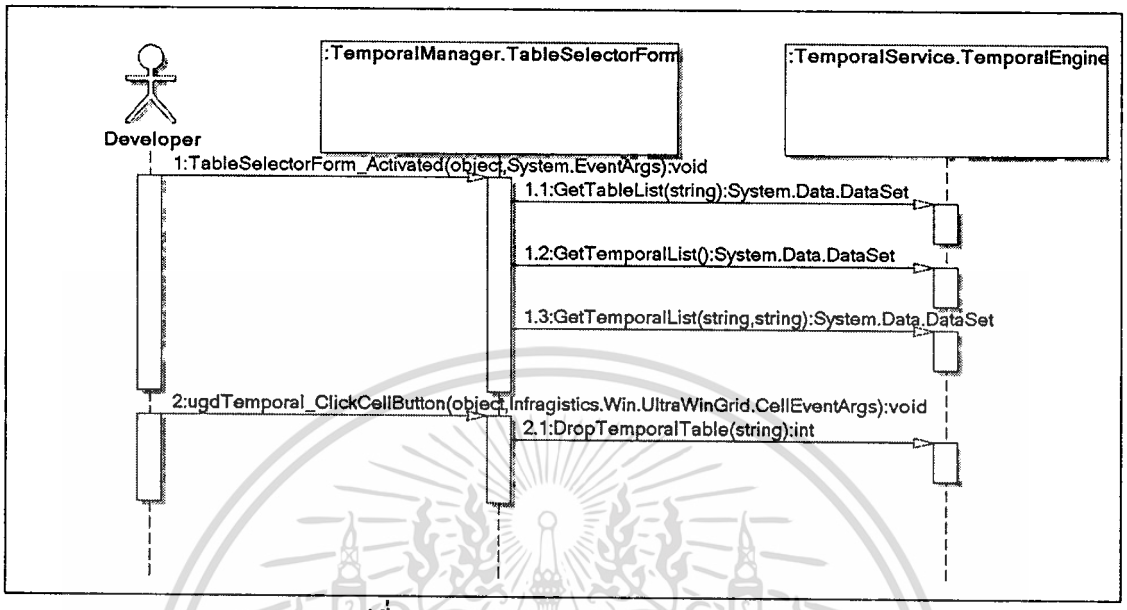
รูปที่ 5.7 Sequence Diagram ของ UC05

5.3.6 UC06 ตรวจสอบความถูกต้องของตารางข้อมูลเชิงเวลา



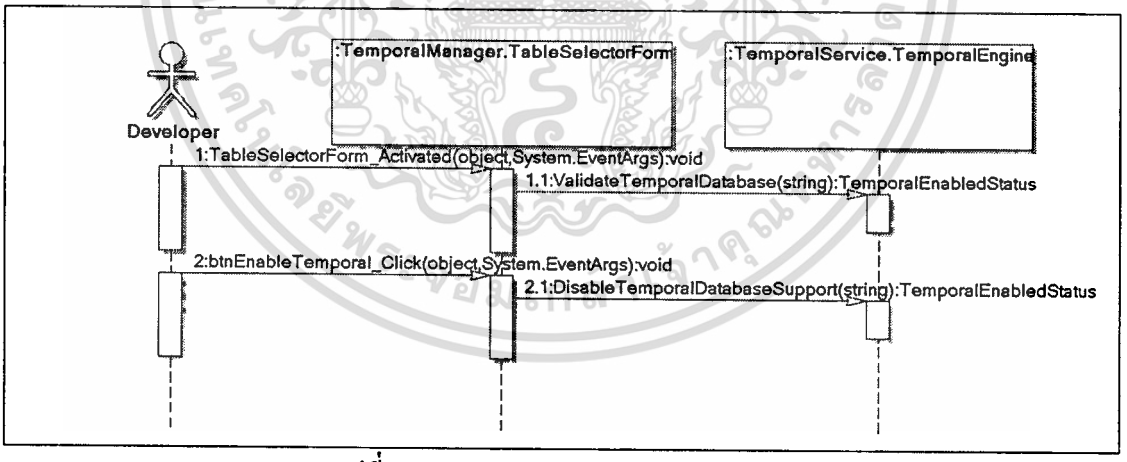
รูปที่ 5.8 Sequence Diagram ของ UC06

5.3.7 UC07 ลบตารางข้อมูลเชิงเวลา



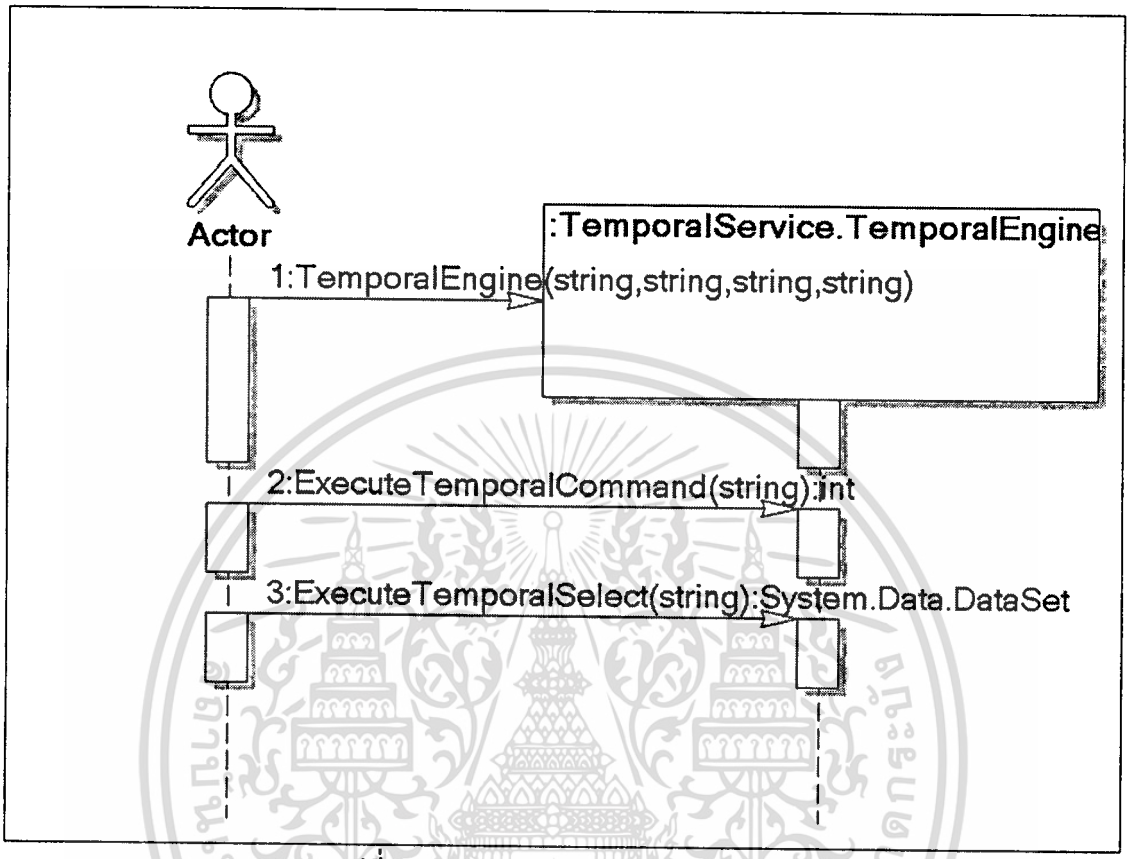
รูปที่ 5.9 Sequence Diagram ของ UC07

5.3.8 UC08 ยกเลิกการรองรับคุณสมบัติเชิงเวลาของฐานข้อมูล



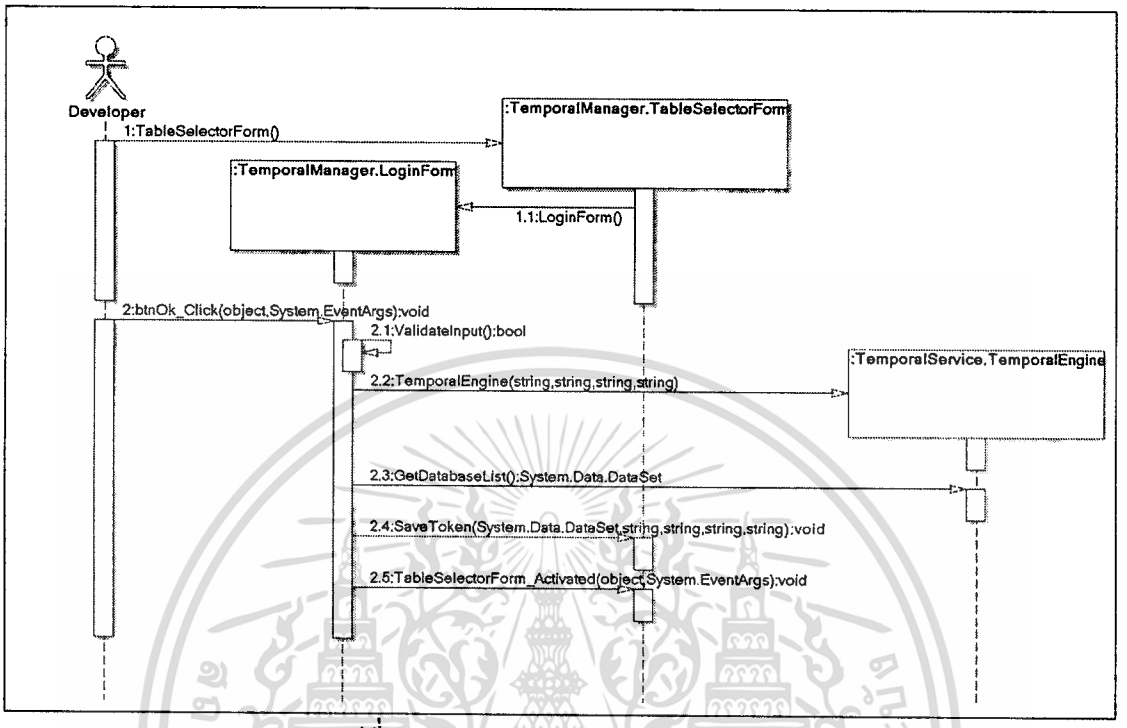
รูปที่ 5.10 Sequence Diagram ของ UC08

5.3.9 UC09 แปลคำสั่ง SQL เชิงเวลาเป็นคำสั่ง Transact-SQL



รูปที่ 5.11 Sequence Diagram ของ UC09

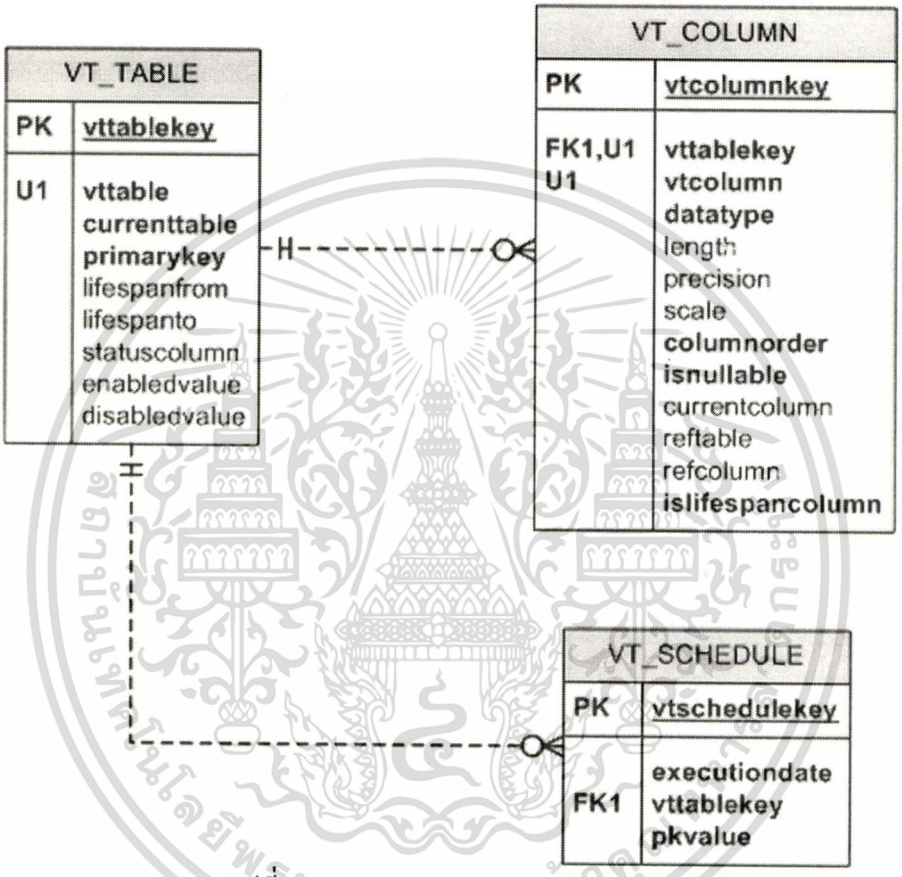
5.3.10 UC10 เชื่อมต่อฐานข้อมูล



รูปที่ 5.12 Sequence Diagram ของ UC10

5.4 Data Model

5.4.1 Entity-Relationship Diagram



รูปที่ 5.13 Entity-Relationship Diagram

5.4.2 Data Dictionary

VT_COLUMN

คำอธิบาย

เป็นตารางข้อมูลที่เก็บข้อมูลคอลัมน์เชิงเวลาของฐานข้อมูล

คอลัมน์

ชื่อคอลัมน์	รายละเอียด	ประเภทข้อมูล	Allow Nulls	คีย์
vtcolumnkey	คีย์ของตาราง	int identity	N	PK
vttablekey	คีย์ของตารางข้อมูล VT_TABLE	int	N	FK1, U1
vtcolumn	ชื่อคอลัมน์เชิงเวลา	varchar(128)	N	U1
datatype	ประเภทข้อมูล	varchar(20)	N	
length	ความยาวข้อมูล	smallint	Y	
precision	ความยาวตัวเลข	tinyint	Y	
scale	ความยาวทศนิยม	tinyint	Y	
columnorder	ลำดับคอลัมน์	tinyint	N	
reftable	อ้างอิงถึงตารางข้อมูล	varchar(128)	Y	
refcolumn	อ้างอิงถึงคอลัมน์	varchar(128)	Y	
islifespancolumn	คอลัมน์นี้เป็นส่วนหนึ่งของ Lifespan หรือไม่	bit	N	DEFAULT(0)

VT_SCHEDULE

คำอธิบาย

เป็นตารางข้อมูลที่เก็บข้อมูลกำหนดการทำงานของ Temporal Agent

คอลัมน์

ชื่อคอลัมน์	รายละเอียด	ประเภทข้อมูล	Allow Nulls	คีย์
vtschedulekey	คีย์ของตาราง	int identity	N	PK
executiondate	วันที่ เวลา ในการทำงานของ Agent	datetime	N	
vttablekey	คีย์ของตารางข้อมูล VT_TABLE	int	N	FK1
pkvalue	ค่าคีย์ของตารางข้อมูลปกติที่ต้องมีการตรวจสอบช่วงชีวิตโดย Agent	int	N	

VT_TABLE

คำอธิบาย

เป็นตารางข้อมูลที่เก็บข้อมูลตารางข้อมูลเชิงเวลา

คอลัมน์

ชื่อคอลัมน์	รายละเอียด	ประเภทข้อมูล	Allow Nulls	คีย์
vttablekey	คีย์ของตาราง	int identity	N	PK
vttable	ชื่อตารางข้อมูลเชิงเวลา	varchar(128)	N	U1
currenttable	ชื่อตารางข้อมูลปกติ	varchar(128)	N	
primarykey	คีย์ของตารางข้อมูลปกติ	varchar(128)	N	
lifespanfrom	คอลัมน์ของตารางข้อมูลปกติที่แสดง ช่วงชีวิตเริ่มต้น	varchar(128)	Y	
lifespanto	คอลัมน์ของตารางข้อมูลปกติที่แสดง ช่วงชีวิตสิ้นสุด	varchar(128)	Y	
statuscolumn	คอลัมน์ของตารางข้อมูลปกติที่แสดง สถานะ	varchar(128)	Y	
enabledvalue	ค่าที่หมายความเป็น Enabled	int	Y	
disabledvalue	ค่าที่หมายความเป็น Disabled	int	Y	

5.5 SQL Server Objects

5.5.1 Procedures สำหรับการทำงานของ Agent

NAME	DESCRIPTION
p_synchronize_database	อยู่ในฐานข้อมูลเชิงเวลา มีฐานข้อมูลละ 1 ออปเจกต์ ทำหน้าที่อ่าน VT_SCHEDULE เพื่อดึงว่า มี @key อะไรบ้างที่ต้องการปรับปรุงช่วงชีวิต ดึงข้อมูลเฉพาะแถวข้อมูล ที่ executiondate <= GETDATE() เท่านั้น เก็บข้อมูลที่ได้ไว้ใน CURSOR ทำการ fetch เพื่อ execute คำสั่ง modify ทีละ record อีกครั้ง หลังจาก execute เป็นผลสำเร็จ (ทีละแถว) ก็ทำการลบทิ้งจาก VT_SCHEDULE ได้เลย สร้างไว้เสมอเมื่อมีการตั้งให้ฐานข้อมูลรองรับคุณสมบัติเชิงเวลา
p_synchronize_<vttable>	อยู่ในฐานข้อมูลเชิงเวลา จำนวน Procedure เท่ากับจำนวนตารางข้อมูลเชิงเวลาในฐานข้อมูลนั้นๆ ทุกครั้งที่สร้าง vttable ต้องสร้าง Procedure นี้ขึ้นมาด้วย ทำหน้าที่แก้ไขช่วงชีวิตของตารางข้อมูลปกติตามข้อมูลในอนาคตของตารางข้อมูลเชิงเวลา
p_create_temporal_agent	Stored Procedure นี้อยู่ในฐานข้อมูลเชิงเวลา ทำหน้าที่สร้าง Agent Task ขึ้นมา พร้อมกำหนด log file location ต่างๆ โดยยังไม่กำหนด task schedule Task มี 1 step คือเรียกใช้ Stord procedure p_activate_lifespan_modification

p_set_temporal_job_schedule	อยู่ในฐานข้อมูลเชิงเวลา ทำหน้าที่ตั้ง Execution Time ครั้งต่อไปของ Temporal Agent ให้เป็นการรอ Execution Time ครั้งต่อไป เวลาตั้ง ที่โปรแกรมต้องระวัง และตรวจด้วยว่า Execution Time ครั้งต่อไปต้องไม่น้อยกว่า Now เสมอ มิฉะนั้น จะไม่ได้ Execute Task นี้

5.5.2 Procedure สำหรับ Uniqueness Constraint

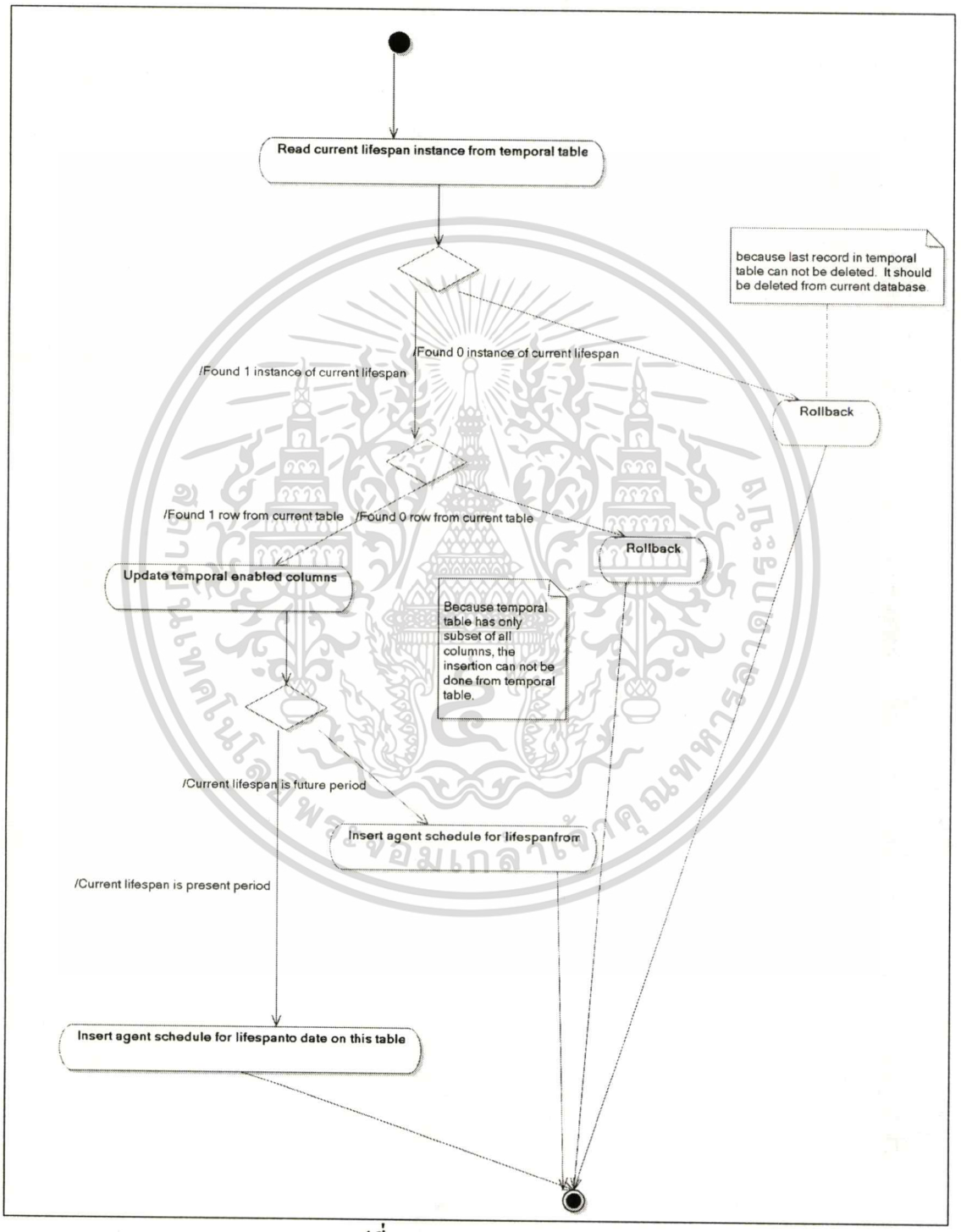
ชื่อ	คำอธิบาย
p_check_tuc_<vttable>	อยู่ในฐานข้อมูลเชิงเวลา จำนวน Procedure เท่ากับจำนวนตารางข้อมูลเชิงเวลาในฐานข้อมูลนั้น แต่ละตารางข้อมูลเชิงเวลา ต้องมี procedure นี้เอง ทุกครั้งที่สร้าง ตารางข้อมูลเชิงเวลา ต้องสร้าง Procedure นี้ขึ้นมาด้วย ทำหน้าที่ตรวจสอบ Sequence Uniqueness Constraint ของตารางข้อมูลเชิงเวลา

5.5.3 Trigger สำหรับตารางข้อมูลปกติ

ชื่อ	คำอธิบาย
<currenttable>_insert	อยู่ในฐานข้อมูลเชิงเวลา จำนวน Procedure เท่ากับจำนวนตารางข้อมูลปกติที่มีตารางข้อมูลเชิงเวลาในฐานข้อมูลนั้น พอสร้าง ตารางข้อมูลเชิงเวลา สำหรับ ตารางข้อมูลปกติ ต้องมีการ Regenerate Trigger ตัวนี้เสมอ ทำหน้าที่ควบคุมการเพิ่มข้อมูลไปยังตารางข้อมูลเชิงเวลา
<currenttable>_update	อยู่ในฐานข้อมูลเชิงเวลา จำนวน Procedure เท่ากับจำนวนตารางข้อมูลปกติที่มีตารางข้อมูลเชิงเวลาในฐานข้อมูลนั้น พอสร้าง ตารางข้อมูลเชิงเวลา สำหรับ ตารางข้อมูลปกติ ต้องมีการ Regenerate Trigger ตัวนี้เสมอ ทำหน้าที่ควบคุมไม่ให้มีการแก้ไขค่าในคอลัมน์เชิงเวลาจากแอพพลิเคชันปกติ

5.6 Activity Diagram

5.6.1 การแก้ไขข้อมูลตารางข้อมูลปกติโดย Temporal Agent



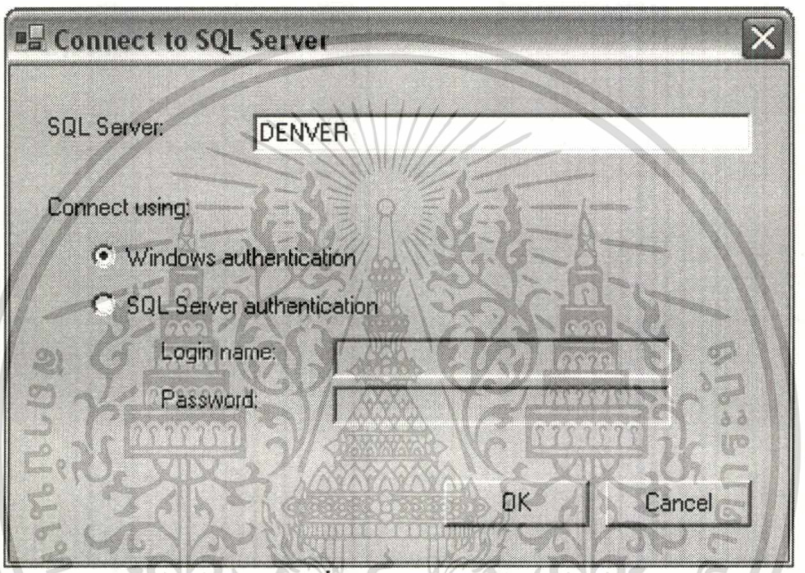
รูปที่ 5.14 Activity Diagram

5.7 ข้อจำกัดของระบบตามแบบ

1. Primary Key ของตารางข้อมูลปกติที่เปลี่ยนเป็นตารางข้อมูลเชิงเวลาได้ ต้องเป็น Simple Key เท่านั้น
2. คอลัมน์ที่อ้างถึงตารางข้อมูลปกติอื่นหากเปลี่ยนเป็นคอลัมน์เชิงเวลาต้องอ้างไปยังตารางข้อมูลปกติ ที่มี Primary Key เป็น Simple Key เท่านั้น (หากต้องการรักษา Foreign Key Constraint)
3. ประเภทข้อมูลของ คอลัมน์ Primary Key ต้องเป็น Integer เท่านั้น
4. คอลัมน์อื่นๆที่เพิ่มเข้าไปในตารางข้อมูลเชิงเวลา ต้องมีคุณสมบัติเป็นอนุญาตให้มีค่า Null เท่านั้น (เพราะการเพิ่มแถวข้อมูลแรกของคีย์จะมาจากตารางข้อมูลปกติซึ่งไม่มีคอลัมน์เหล่านี้)
5. ตามขอบเขตข้อมูลของประเภทข้อมูล datetime ของ Microsoft SQL Server ให้ Time Zero มีค่าเท่ากับ 1/1/1753
6. ตามขอบเขตข้อมูลของประเภทข้อมูล datetime ของ Microsoft SQL Server ให้ Infinity มีค่าเท่ากับ 12/31/9999
7. คอลัมน์สถานะต้องเป็นหมวดประเภท Integer เท่านั้น เป็น bit, byte, int ได้
8. การปรับปรุงแถวข้อมูลในตารางข้อมูลเชิงเวลา สามารถเกิดได้ทีละ 1 แถวข้อมูลเท่านั้น
9. ในแอพพลิเคชันปกติ ต้องไม่มีการแก้ไขคอลัมน์เชิงเวลา ใน ตารางข้อมูลผ่าน Trigger ในฐานข้อมูล เนื่องจากในการแก้ไขย้อนกลับมาจากตารางข้อมูลเชิงเวลาเป็นการกระทำผ่าน Trigger
10. การส่งคำสั่ง SQL เชิงเวลา ผ่านทาง Temporal Engine สามารถส่งได้ทีละ 1 ประโยคเท่านั้น ไม่สามารถส่งเป็น Script ได้
11. แถวข้อมูลของค่าคีย์หนึ่งที่พบในตารางข้อมูลเชิงเวลาต้องปรากฏในตารางข้อมูลปกติ หากใน ตารางข้อมูลปกติไม่มี หมายความว่า ในตารางข้อมูลเชิงเวลาก็ต้องไม่มีด้วย
12. ผู้ทำการใช้ Temporal Manager ต้องมีสิทธิในการสร้าง Stored Procedure ใน master Database (sp_create_temporal_agent) และ สิทธิในการสร้าง Agent และ กำหนด Agent Schedule บน SQL Server Agent
13. SQL Server Agent ต้องทำงานในขณะใช้งาน
14. การตั้งเวลา ระวังไม่ให้ตกหลัง 23:00 น. เนื่องจากใน Microsoft SQL Server 2000 การตั้งเวลาทำงานนั้นแยกวันที่และเวลาออกจากกัน การทำงานจะเกิดข้อผิดพลาดในกรณีเริ่มต้นทำงานในวันหนึ่งเมื่อบวกเวลาทำงานและ Retry ปรากฏว่าเปลี่ยนวันที่ไปตกวันใหม่

บทที่ 6
การใช้ระบบ

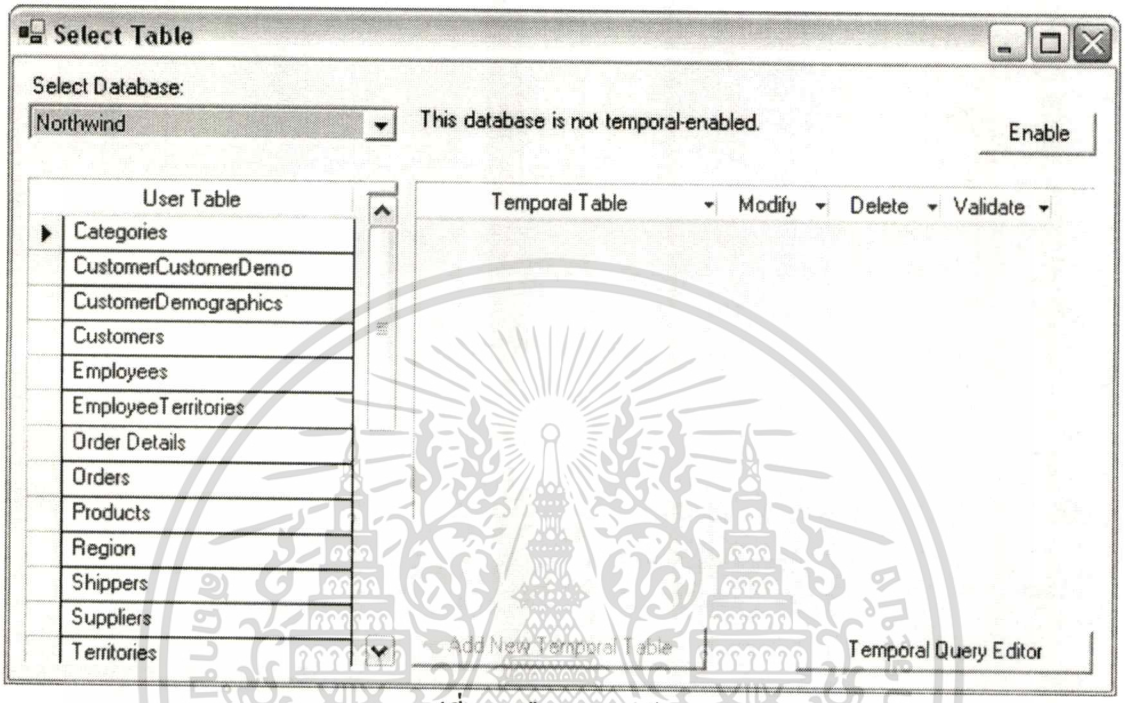
6.1 การ Login เข้าเซิร์ฟเวอร์ฐานข้อมูล



รูปที่ 6.1 Login Form

เมื่อเรียกโปรแกรม Temporal Manager เพื่อทำการจัดการฐานข้อมูลเชิงเวลา ผู้ใช้ต้องทำการ Login เข้าสู่เซิร์ฟเวอร์ฐานข้อมูลก่อน โดยเมื่อ Login แล้ว โปรแกรมจะทำงานติดต่อกับเซิร์ฟเวอร์ที่ Login เท่านั้น

6.2 เลือกตารางข้อมูล



รูปที่ 6.2 เลือกตารางข้อมูล

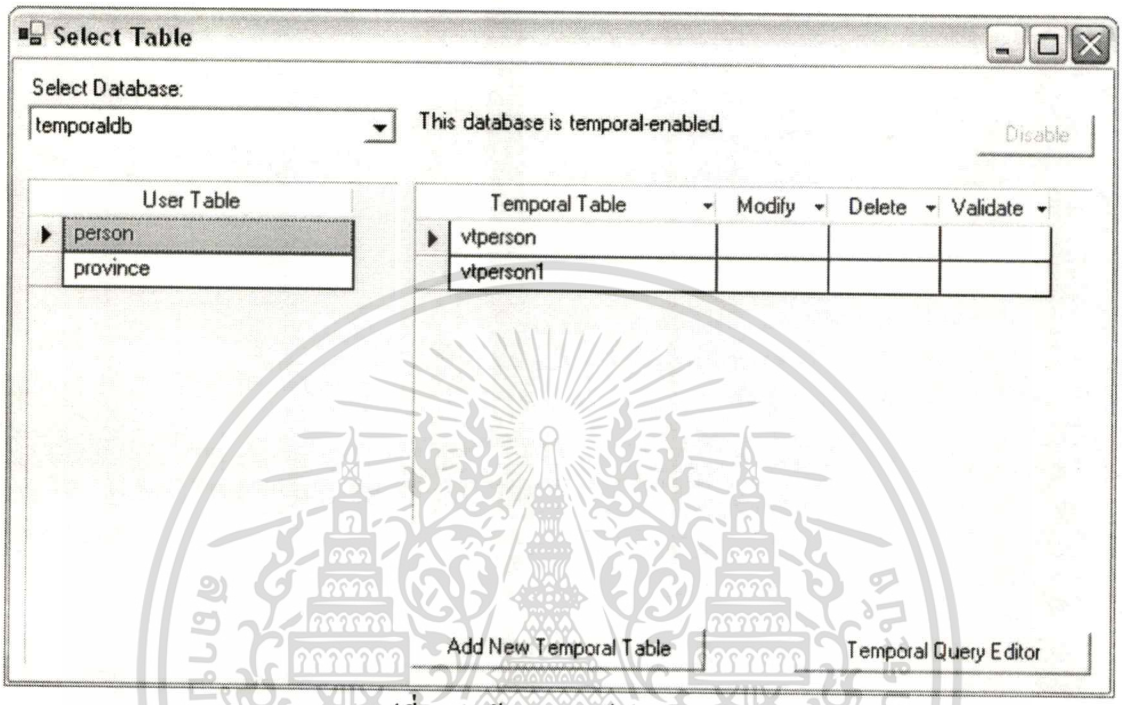
ณ หน้าจอนี้ สามารถทำงานได้หลายอย่าง ดังนี้

6.2.1 ปรับฐานข้อมูลให้รองรับคุณสมบัติฐานข้อมูลเชิงเวลา

เนื่องจากการสร้างตารางข้อมูลเชิงเวลาจำเป็นต้องมีการเก็บ Metadata ไว้ในฐานข้อมูลด้วย รวมทั้งต้องมีออปเจกต์ฐานข้อมูลบางตัวอยู่ในฐานข้อมูล master ของเซิร์ฟเวอร์ ดังนั้น ฐานข้อมูลใดที่ต้องการใช้คุณสมบัติเชิงเวลา ต้องมีการปรับก่อน โดยโปรแกรมจะไปสร้างตารางข้อมูลและออปเจกต์ต่างๆที่จำเป็นไว้ในฐานข้อมูลเพื่อรองรับตารางข้อมูลเชิงเวลาต่อไป

เลือกฐานข้อมูลที่ต้องการ กดปุ่ม Enable เพื่อปรับฐานข้อมูล

6.2.2 สร้าง / แก้ไข นิยามตารางข้อมูลเชิงเวลา

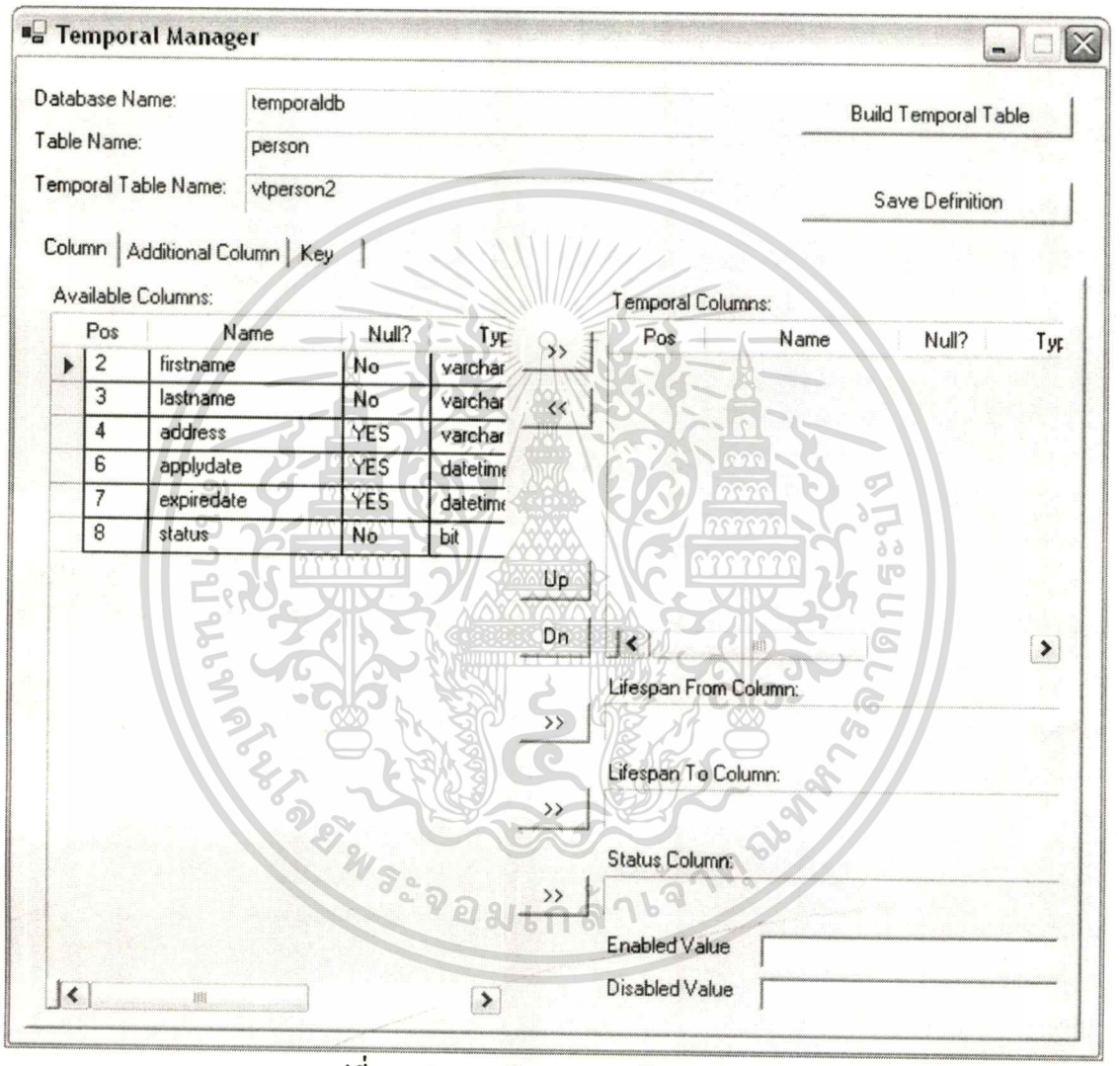


รูปที่ 6.3 สร้างตารางข้อมูลเชิงเวลา

หากฐานข้อมูลรองรับคุณสมบัติเชิงเวลา เราต้องการสร้างนิยามตารางข้อมูลเชิงเวลาใหม่ ทำการเลือกตารางข้อมูลปกติ และกดปุ่ม Add New Temporal Table

6.3 กำหนดนิยามตารางข้อมูลเชิงเวลา

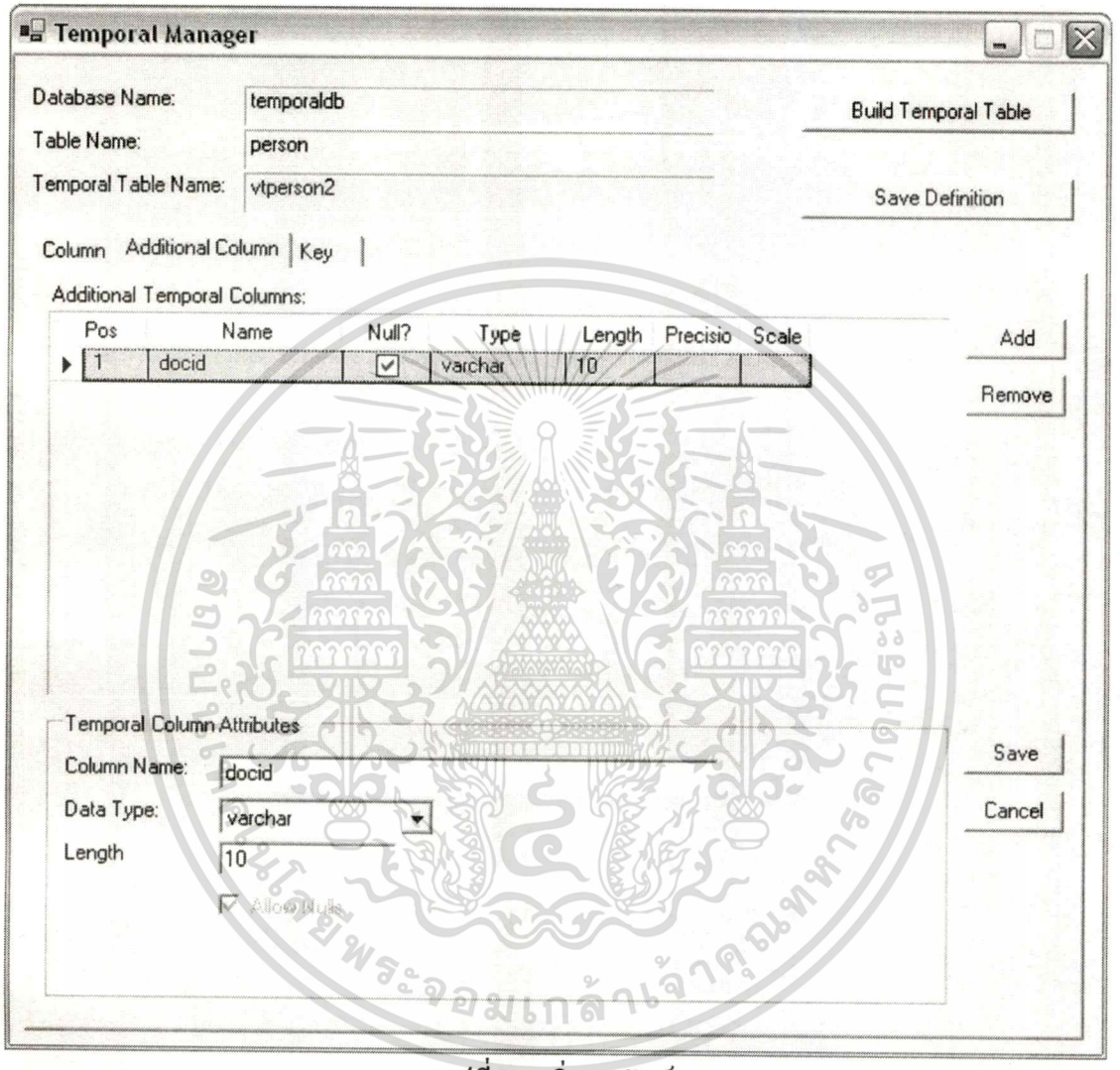
6.3.1 เลือกคอลัมน์



รูปที่ 6.4 กำหนดนิยามตารางข้อมูลเชิงเวลา

ในหน้าต่างนี้ เราสามารถเลือกคอลัมน์ของตารางข้อมูลปกติ ว่า คอลัมน์ใดที่เป็นคอลัมน์เชิงเวลา คอลัมน์ใดแสดงถึงจุดเริ่มต้นช่วงชีวิต จุดสิ้นสุดช่วงชีวิต และคอลัมน์สถานะ

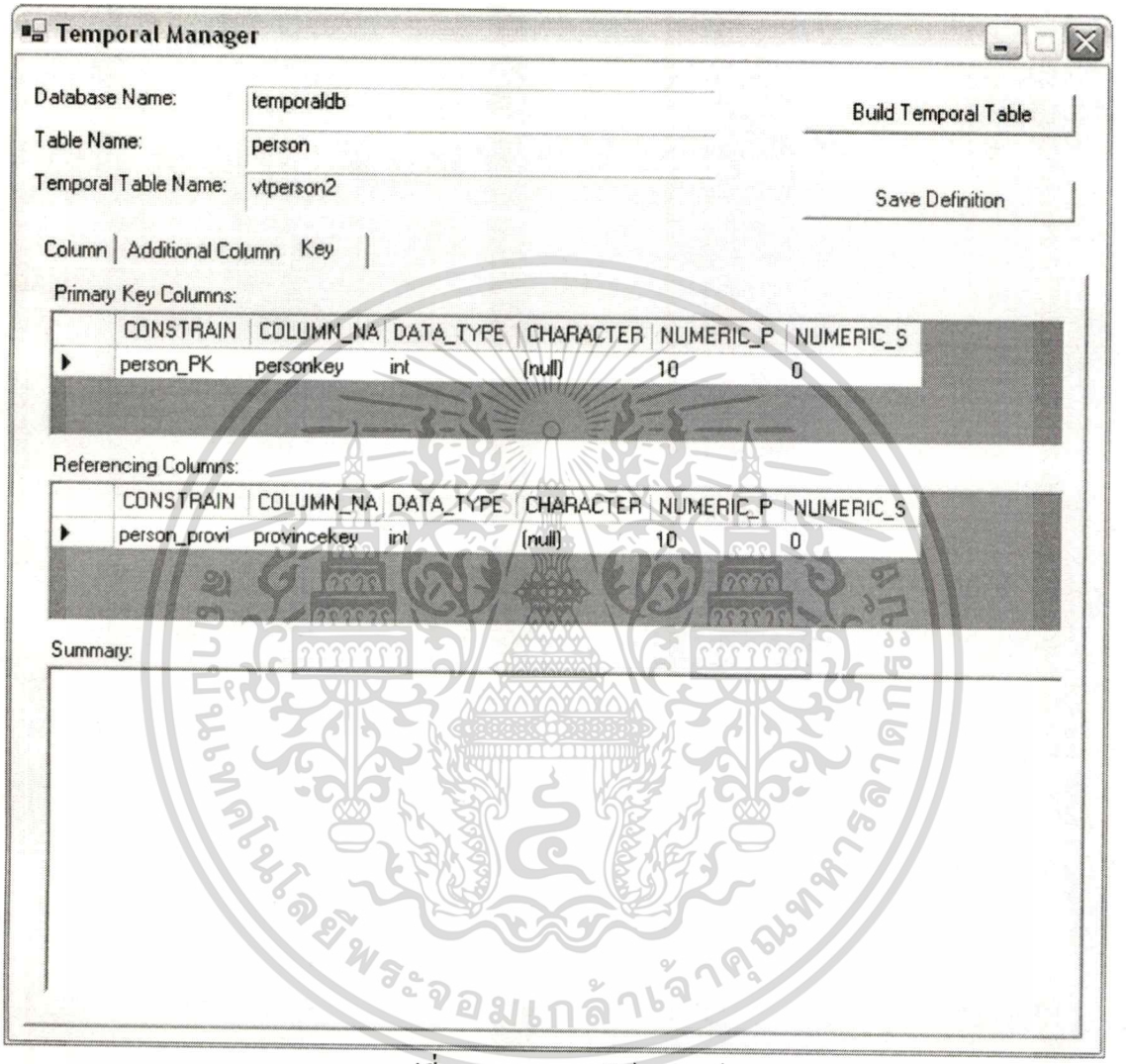
6.3.2 เพิ่มคอลัมน์



รูปที่ 6.5 เพิ่มคอลัมน์

เราสามารถเพิ่มคอลัมน์ที่ไม่มีในตารางข้อมูลปกติเข้าไปในตารางข้อมูลเชิงเวลาได้ แต่คอลัมน์นั้นต้องอนุญาตให้มีค่า Null ได้

6.3.4 ตรวจสอบคีย์คอลัมน์ของตาราง



รูปที่ 6.6 ตรวจสอบคีย์คอลัมน์

ในแทปนี้ เราสามารถตรวจสอบได้ว่า ตารางข้อมูลปกติมี Primary Key และ Foreign Key อะไร

หลังจากที่ปรับแต่งนิยามของตารางข้อมูลเชิงเวลาเรียบร้อยแล้ว กดปุ่ม Save Definition เพื่อเก็บนิยามนี้ไว้ในฐานข้อมูล

6.4 การสร้างตารางข้อมูลเชิงเวลา

เมื่อต้องการสร้างตารางข้อมูลเชิงเวลา ให้เลือกนิยาม (ดังรูป 6.3) และกดปุ่ม Modify โปรแกรมจะดึงนิยามนั้นออกมาแสดงดังรูป 6.4 หากเราตรวจสอบว่าถูกต้อง กดปุ่ม Build Temporal Table จะเป็นการสร้างตารางข้อมูลเชิงเวลา จากนิยามที่กำหนดไว้

6.5 การลบตารางข้อมูลเชิงเวลา

ใช้ปุ่ม Delete บนหน้าจอดังรูป 6.3 หากนิยามนั้นมีตารางข้อมูลอยู่ด้วย ผู้ใช้จะเลือกได้ว่าจะลบตารางข้อมูลเพียงอย่างเดียว หรือ ลบทั้งตารางข้อมูลและนิยาม

หากลบตารางข้อมูลแล้ว ไม่สามารถกู้ขึ้นมาได้อีก

บทที่ 7
สรุปและแนวทางพัฒนาต่อในอนาคต

7.1 สรุป

แนวทางประยุกต์ที่งานวิจัยนำเสนอในการนำฐานข้อมูลเชิงเวลามาใช้ร่วมกับฐานข้อมูลเดิมซึ่งไม่ใช่ฐานข้อมูลเชิงเวลา ถึงแม้จะสามารถแก้ปัญหาการนำฐานข้อมูลเชิงเวลามาใช้ตรงๆ 7 ประการที่เกิดขึ้น แต่ตัวแนวทางที่นำเสนอก็ต้องประสบกับปัญหาใหม่ ซึ่งเป็นปัญหาเนื่องจากเป็นการสร้างตารางข้อมูล 2 ชุด คือ การจัดการให้ข้อมูลในตารางข้อมูล 2 ชุดนั้นสอดคล้องกันอยู่เสมอ

เราจำเป็นต้องใช้ออปเจกต์ของฐานข้อมูล เช่น Stored Procedure และ Trigger เพื่อควบคุมความสอดคล้องของข้อมูล ซึ่งอาจสร้างความยุ่งยากในระดับหนึ่งให้กับแอพพลิเคชันเดิมที่มีอยู่แล้ว จึงทำให้ไม่สามารถประกันได้ว่า วิธีการนี้จะไม่กวนกับแอพพลิเคชันเดิม 100 เปอร์เซ็นต์

นอกจากนี้ ยังมีข้อจำกัดของระบบ ตามหัวข้อ 5.7 ซึ่งอาจเป็นเหตุให้ไม่สามารถนำไปใช้กับแอพพลิเคชันทั่วไปได้ทั้งหมด

แต่อย่างไรก็ตาม ในขณะนี้ เราไม่มีระบบจัดการฐานข้อมูลซึ่งรวมคุณสมบัติฐานข้อมูลเชิงเวลาไว้ในตัว สามารถใช้งานได้ง่าย รองรับภาษาสืบค้นเชิงเวลา ที่พร้อมใช้ในตลาด สำหรับการนำฐานข้อมูลเชิงเวลามาใช้เพื่อรองรับความต้องการสืบค้นเชิงเวลา แนวทางที่งานวิจัยนี้นำเสนอน่าจะเป็นวิธีที่ช่วยให้สามารถเริ่มต้นสร้างแอพพลิเคชันเชิงเวลาขึ้นมาได้ โดยไม่จำเป็นต้องละทิ้งโครงสร้างฐานข้อมูลเดิมทั้งหมด เป็นการผสมผสานความสามารถของฐานข้อมูลเชิงเวลาเข้ากับฐานข้อมูลปกติ โดยที่ผู้พัฒนาระบบไม่จำเป็นต้องเลือกว่า จะต้องไปทางใดทางหนึ่งเท่านั้น

7.2 แนวทางพัฒนาต่อในอนาคต

เครื่องมือที่ได้สร้างขึ้นมาเพื่อพิสูจน์แนวทางประยุกต์ยังไม่สมบูรณ์ในหลายประการ ดังเช่น อธิบายในหัวข้อ 5.7

การพัฒนาเครื่องมือนี้ต่อไป ควรที่จะให้สามารถเลือกได้ว่า ต้องการสร้างฐานข้อมูลเชิงเวลาบนระบบจัดการฐานข้อมูลอื่นๆ ได้ นอกเหนือจาก Microsoft SQL Server และควรสนับสนุนมาตรฐานภาษา SQL เชิงเวลาที่สมบูรณ์มากกว่าปัจจุบัน

วิธีในการควบคุมความสอดคล้องของข้อมูลอาจต้องมีมากกว่า แบบที่นำเสนอ 1 แบบ เพื่อสามารถรองรับแอพพลิเคชันปัจจุบันได้มากกว่าเดิม

นอกเหนือจากนี้ การพัฒนาในอนาคตสามารถเพิ่มคุณสมบัติให้รองรับ เวลาชนิด Transaction Time ได้ รวมทั้ง Bitemporal (คือมีทั้ง Valid Time และ Transaction Time)

บรรณานุกรม

Microsoft. 2002. **Programming with the Microsoft .NET Framework (Microsoft Visual C# .NET Delivery Guide.** Redmond: Microsoft Corporation.

Rumbaugh, J. et al. 1999. **The Unified Modeling Language Reference Manual.** Massachusetts: Addison Wesley Longman.

Schneider, Geri and Winters, Jason P. 2001. **Applying Use Cases, Second Edition A Practical Guide.** New Jersey: Addison Wesley Longman.

Snodgrass T., Richard. et al. 1997. **Transitioning Temporal Support in TSQL2 to SQL3.** [Online]. Available: http://www.cs.auc.dk/research/DP/tdb/TimeCenter/TimeCenterPublications/TR-8.pdf.

Snodgrass T., Richard. 1998. **Managing Temporal Data A Five-Part Series.** [Online]. Available: http://www.cs.auc.dk/research/DP/tdb/TimeCenter/TimeCenterPublications/TR-28.pdf.

Snodgrass T., Richard. 2000. **Developing Time-Oriented Database Application in SQL.** San Francisco: Morgan Kaufmann.

Steiner, Andreas. 1998. **A Generalisation Approach to Temporal Data Models and their Implementations.** [Online]. Available: http://citeseer.ist.psu.edu/steiner98generalisation.html

ภาคผนวก ก

Test Case สำหรับการปรับปรุงข้อมูลเชิงเวลา

INSERT

A: Period ของเรคอร์ดที่มีอยู่ในตาราง

B: Period ของเรคอร์ดที่เพิ่มเข้าไป

เลขที่	TEST CASE	ผลการทดสอบ
A01	A [----------) B [----------)	เพิ่มข้อมูลไม่ได้
A02	A [----------) B [----------)	เพิ่มข้อมูลไม่ได้
A03	A [----------) B [----------)	เพิ่มข้อมูลไม่ได้
A04	A [----------) B [----------)	เพิ่มข้อมูลไม่ได้
A05	A [----------) B [----------)	เพิ่มข้อมูลไม่ได้
A06	A [----------) B [----------)	เพิ่มข้อมูลได้
A07	A [----------) B [----------)	เพิ่มข้อมูลได้
A08	A [----------)[----------) B [----------)	เพิ่มข้อมูลไม่ได้
A09	A [----------) B [----------)	เพิ่มข้อมูลได้
A10	A [----------) B [----------)	เพิ่มข้อมูลได้

DELETE

A: Period ของเรคอร์ดที่มีอยู่ในตาราง

B: Period ของเรคอร์ดที่ต้องการลบออก

เลขที่	TEST CASE	ผลการทดสอบ
B01	A [----------) B [----------) Result	ผลตาม Result
B02	A [----------) B [----------) Result [--------)	ผลตาม Result
B03	A [----------) B [----------) Result [-------)	ผลตาม Result
B04	A [----------) B [----------) Result	ผลตาม Result
B05	A [----------) B [----------) Result [--------) [----------)	ผลตาม Result
B06	A [----------) B [----------) Result [----------)	ผลตาม Result
B07	A [----------) B [----------) Result [----------)	ผลตาม Result
B08	A [----------)[----------) B [----------) Result [--------) [-------)	ผลตาม Result

B09	A [----------) B [----------) Result [----------)	ผลตาม Result
B10	A [----------) B [----------) Result [----------)	ผลตาม Result

UPDATE

A: Period ของเรคอร์ดที่มีอยู่ในตาราง

B: Period ของเรคอร์ดที่ต้องการแก้ไขข้อมูล

เลขที่	CASE	ผลการทดสอบ
C01	A [----------) B [----------) Result Update [----------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C02	A [----------) B [----------) Result [-------) Update [-------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C03	A [----------) B [----------) Result [-------) Update [-------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C04	A [----------) B [----------) Result Update [----------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update

C05	A [----------) B [----------) Result [--------) [----------) Update [----------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C06	A [----------) B [----------) Result [----------) Update	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C07	A [----------) B [----------) Result [----------) Update	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C08	A [----------)[----------) B [----------) Result [--------) [-------) Update [-------)[--------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C09	A [----------) B [----------) Result [----------) update	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C10	A [----------) B [----------) Result [----------) Update	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update
C11	A [----------)[----------) B [----------) Result Update [----------)[----------)	ผลตาม Result และ Period ที่แก้ไขอยู่ใน Update

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย ศุภฤกษ์ ศิวะมาศ
วันเดือนปีเกิด	14 ตุลาคม 2508
สถานที่เกิด	จ. ชลบุรี
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต สาขาไฟฟ้า
สถานที่สำเร็จการศึกษา	จุฬาลงกรณ์มหาวิทยาลัย
ปีที่สำเร็จการศึกษา	2529
ปัจจุบันทำงานที่	บริษัท ซอฟต์แวร์คอนสตรัคเตอร์ จำกัด