การศึกษาพัฒนา Interactive TSQL ของ Valid time และ Transaction time ด้วย SQL-92

Development Interactive TSQL with Valid time and Transaction time by SQL-92

โดย

นาย ณัฐวุฒิ รัตนจรัสกุล

รหัส 41067123

อาจารย์ที่ปรึกษา

รศ. ดร. ศุภมิตร จิตตะยโศธร

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2543
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การศึกษาพัฒนา Interactive TSQL ของ Valid time และ Transaction time ด้วย SQL-92
นักศึกษา	นายณัฐวุฒิ รัตนจรัสกุล
อาจารย์ที่ปรึกษา	รศ. ดร. ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

มีโปรแกรมทางฐานข้อมูลจำนวนมากที่ใช้ข้อมูลด้านเวลา เช่น โปรแกรมด้านบัญชี บุคคล บริหารเวลา และ datawarehoursing ในเวลาเดียวกันเราก็จะทราบว่าการ query และ การ update ที่ถูกต้องของข้อมูลที่มีการแปรผันด้านเวลาให้ถูกต้องและยากเมื่อใช้ SQL ที่เป็นมาตราฐาน เราจึงแสดงการทำงานให้เห็นชัดเจนบน SQL มาตรฐาน เมื่อมีการบริหารข้อมูลด้านเวลา

ในรายงานฉบับนี้จึงเป็นการศึกษาโครงสร้างของรูปแบบคำสั่งในการเรียกค้นข้อมูลของทั้ง สองภาษา คือ Temporal SQL ที่เป็นคำสั่งบน Temporal Database และ SQL-92 ซึ่งเป็นคำสั่งบน Conventional Database ซึ่งจะเปรียบเทียบภายใต้ข้อมูลที่มีลักษณะเป็น time-varying data เพื่อให้เห็นลักษณะการทำงานในโครงสร้างของภาษาทั้งสอง ในส่วนที่จัดการเกี่ยวกับข้อมูลที่มีความเกี่ยวข้องกับเวลา ในด้านการทำงานของ SQL Queries, โครงสร้างของภาษา, และการใช้ procedural code เพื่อให้เห็นถึงลักษณะการทำงาน ประสิทธิภาพ และความแตกต่างกันของโครงสร้างภาษา

Title	Development Interactive TSQL with Valid time and Transaction time by SQL-92
Student	Mr. Nattavuth Rattanajaruskul
Advisor	Dr. Suphamit Jittajasothorn
Level of Study	Master of Science in Information Technology
Major	Information Scince
Academic Year	2000

ABSTRACT

A wide range of database applications manage time-varying data. Examples include, e.g., accounting, personnel, schedule, and data warehousing applications. At the same time, it is well-known that querying and correctly updating time-varying data is difficult when using standard SQL. We will show clearly process over standard SQL when manage time-vary

For this paper is the study of query language in Temporal SQL that were statement on Temporal Database and SQL-92 on Conventional Database. We will comparison in time-varying data state for reveal structured in SQL that in data with related by time on structure language, procedure code and query processing for represent about performance and indifferenciate

กิตติกรรมประกาศ

ขอกราบขอบพระคุณบิดามารดาที่ได้อบรมสั่งสอนให้ ผมสนใจในการเรียนตั้งแต่ยังเด็ก ตลอดจนให้กำลังใจตลอดเวลา

ขอขอบพระคุณ อาจารย์ ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงการนี้ ที่ให้คำแนะนำข้อคิดเห็น ชี้แนวทาง และความรู้ที่เป็นประโยชน์มาโดยตลอด

รวมถึงการสนับสนุนจาก การสอนของคณาจารย์ และเพื่อน ๆ ทุกคน จึงใครขอขอบพระคุณทุกท่านที่ได้มีส่วนร่วมในการทำให้โครงพัฒนาระบบงานฉบับนี้สำเร็จขึ้นมาได้

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

หน้า

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมา

ในระบบฐานข้อมูลทั่วไป ที่ได้ออกแบบมาให้เก็บข้อมูลที่ทันสมัย นั่นคือเป็นข้อมูลในปัจจุบัน ซึ่งในลักษณะนี้คือฐานข้อมูลแบบ conventional databases แต่ปัญหาที่เกิดขึ้นคือไม่ได้ถูกออกแบบมาเพื่อรองรับ ข้อมูลที่มีความแปรผันตามเวลา ซึ่งข้อมูลในโลกของความเป็นจริงนั้น ก็มีการใช้งานอยู่ ดังนั้น จึงอาจเกิดปัญหาตามมาดังนี้คือ ข้อมูลใหม่มีการ update ทับข้อมูลเดิม หรืออาจจะถูกลบทิ้งจากฐานข้อมูลในทันที เมื่อมีการเพิ่ม หรือแก้ไขข้อมูล หรืออาจจะเกิดความซ้ำซ้อน หรือ การใช้ภาษาที่ทำการเรียกค้นมีความยากลำบากขึ้นในการบริหารจัดการข้อมูลเกี่ยวกับเวลา จึงเกิดการคิดมาตรฐานให้รองรับกับข้อมูลดังกล่าวซึ่งนั่นคือ Temporal Database

ปัจจุบันความหลากหลายของ database applications ที่มีการจัดการข้อมูลด้าน time-varying ที่มีลักษณะดังที่กล่าวมาแล้วคือ temporal database และมีการใช้งานจริง ตัวอย่างเช่น ด้านการศึกษา การแพทย์ การธนาคาร การประกันภัย และ application ด้าน data warehousing เป็นต้น

ในเวลาเดียวกัน SQL-92 เป็นที่รู้จักอย่างกว้างขวาง และการบริหารข้อมูลเชิง temporal ใน SQL-92 คือข้อเสนอที่สับสนและเกิดข้อผิดพลาด updates และ queries บนข้อมูลเชิง temporal มีความซับซ้อนและยากที่จะเปลี่ยนเป็นสูตรได้ถูกต้อง และ เข้าใจอย่างเป็นขั้นตอน สิ่งนี้จึงเป็นสิ่งที่ลึกซึ้งและต้องใช้ความเข้าใจอย่างมาก

ที่สามารถทำได้ในทางปฏิบัติ ภาษาเชิง temporal ต้องพบกับความท้าทายของโครงสร้างภาษาเดิม โดยเฉพาะอย่างยิ่ง temporal query language ควรจะ compatible อย่างมากกับ SQL-92 หมายความว่าการทำงานของโครงสร้างภาษาแบบเดิม จะไม่กระทบเมื่อ temporal สนับสนุน คือนำมาใช้ด้วยกันได้ ยิ่งไปกว่านั้น ภาษามันควรที่จะอนุญาตสำหรับเพิ่มการทำประโยชน์ของ temporal support เมื่อ ภาษา temporal นำมาใช้ครั้งแรก ไม่มี application code ที่นำเอาคุณสมบัติของ temporal features เลย เพียงเมื่อแปลง application เก่า และพัฒนาใหม่ก็จะเกิดประโยชน์ของ temporal support จะมีความสามารถเมื่อนำมันมาผ่าน temporal support temporal และแบบเก่า นั้นคือ non-temporal application จะนำมาใช้ร่วมกันได้อย่างราบเรียบและเข้ากันได้

ในโปรเจคนี้จึงนำเสนอ การพัฒนาระบบ interactive TSQL ของ valid time และ transaction time คือการแปลงภาษาที่มีลักษณะเป็นภาษา temporal ในรูปแบบที่เป็น valid time และ

transaction time ให้เป็นภาษา SQL-92 เพื่อให้คนที่จะพัฒนารูปแบบภาษาของ temporal ได้เข้าใจว่าการทำงานหรือลักษณะโครงสร้างภาษาของ SQL-92 ที่มีคนใช้และรู้จักกันอย่างกว้างขวาง เพื่อให้อยู่ในรูปแบบที่เข้าใจได้ง่าย และสามารถนำไปพัฒนาในระบบเดิมได้อย่างดี และสามารถจะปรับไปใช้ในฐานข้อมูลที่ไม่มีการสนับสนุนภาษา temporal ได้

1.2 วัตถุประสงค์

1. เพื่อให้เห็นแนวความคิด และเหตุผลในการเกิดการขยายของ SQL มาตราฐาน จนเกิดเป็น temporal database
2. เพื่อศึกษาและความเข้าใจในระบบฐานข้อมูลที่กำหนดเป็นมาตราฐานใหม่ นั้นคือ temporal database
3. เข้าใจการทำงาน โครงสร้าง ความแตกต่าง ของ time-varying data และการบริหารข้อมูลด้านเวลาชนิด valid time และ transaction time
4. เพื่อให้เข้าใจประโยชน์หรือความจำเป็นในการใช้งาน หรือ พัฒนาระบบด้วยฐานข้อมูลแบบ temporal
5. ทำให้เข้าใจในรูปแบบของภาษาหรือลักษณะการทำงานของ temporal จากการเปรียบเทียบให้อยู่ในรูปแบบของ SQL-92 ที่เป็นภาษาที่เป็นที่นิยม

1.3 ขอบเขตของระบบ

ศึกษาโครงสร้างของภาษา temporal ที่เป็นชนิดของฐานข้อมูลที่อยู่ในลักษณะของ valid time และ transaction time โดยอิงโครงสร้างของภาษา temporal จาก มาตรฐานภาษา TSQL2 และโครงสร้างภาษาของ SQL-92

1.4 ขั้นตอนการศึกษา

- ศึกษาการทำงานและลักษณะการใช้งานของภาษา temporal
- ศึกษาฐานข้อมูลชนิดต่าง ๆ ของ temporal และ ความแตกต่างของลักษณะภาษาและโครงสร้างของ SQL-92 และ temporal SQL
- ศึกษา semaintics ของภาษาของ SQL-92 และ TSQL เพื่อนำมาจับความสัมพันธ์และผลลัพธ์ที่เกิดขึ้น
- จับ syntax และแปลภาษาเพื่อทำการ Maping ภาษาทั้งสอง

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- เพื่อแสดงให้เห็นถึงความจำเป็นในการพัฒนาระบบงานที่มีลักษณะเป็น temporal database และ เห็นถึงความสำคัญของการนำช่วงเวลามาไว้ในฐานข้อมูล
- ข้อดีและข้อเสีย ประโยชน์เมื่อมีการนำ temporal database มาใช้ในระบบงาน
- แสดงคุณสมบัติที่ดีและเพิ่มประสิทธิภาพในการพัฒนา application
- แสดงถึงความแตกต่างกันของโครงสร้างภาษาของ SQL ที่ใช้ในปัจจุบัน และโครงสร้างภาษาของ temporal SQL
- เข้าใจในโครงสร้างและความแตกต่างของภาษา SQL-92 และ temporal SQL ในด้านการ design และการใช้งาน รวมถึงเข้าใจในความเหมาะสมของภาษาต่อระบบงานที่มีลักษณะแตกต่างกัน และจะทำให้เห็นความสำคัญในการใช้และออกแบบระบบงานที่มีการใช้ฐานข้อมูลเชิง temporal และงานด้านอื่น ๆ

บทที่ 2
วรรณกรรมและทฤษฎีที่เกี่ยวข้อง

2.1 แนวคิดในการเกิด temporal database

ฐานข้อมูลที่มีความนิยม และใช้มากที่สุดคือ relational data model เนื่องจากมีพื้นฐานที่ง่าย และ มีรูปหลักการทางคณิตศาสตร์รองรับ แต่มันก็ไม่ได้เตรียมไว้สำหรับข้อมูลที่มีการมิติของเวลา และมันก็ไม่เพียงพอที่สำหรับ applications ที่ต้องการข้อมูลเวลาที่เป็น อดีต ปัจจุบัน หรืออนาคต แม้ว่า ในความเป็นจริงแล้วเราจะใช้ relational database ที่ยอมให้มีการซ้ำซ้อนของข้อมูลได้มาก แต่ก็ยังทำให้การเขียนภาษา SQL นั้นยุ่งยาก ซับซ้อน และเข้าใจได้ยาก

จึงมีแนวทางที่จะชนะข้อจำกัดนั้น คือมีการขยายความสามารถของ Relational database ให้เป็น Temporal Database เกิดขึ้น

ในทศวรรษนี้มีจึงมีงานวิจัยใน temporal databases ที่ก่อให้เกิดการพัฒนาของ temporal query language อย่างแพร่หลายโดยมีพื้นฐานบน relational languages เช่น TQUEL หรือ temporal extensions ของ SQL และตัวที่มีชื่อเสียงที่สุดคือ TSQL2 และมีอีกมากเช่น ATSQL2 และในปัจจุบันได้มีการเสนอ temporal extension ของ SQL3 ที่ได้ ISO และ ANSI standardization committees-SQL/Temporal

2.2 ข้อเสนอในปัจจุบัน

ทุกข้อเสนอในปัจจุบันจะได้รับการยอมรับที่การใช้ timestamping ใส่ลงไปใน tuples ทั่วไป ๆ ในเวลา ณ ขณะนั้น ที่เป็นตัวนำที่สนับสนุนความต้องการต่อการใช้ข้อมูลจำนวนมากใน tuple นั้น และ ควรมี *repeated* (ให้มีการซ้ำซ้อนได้) สำหรับทุก ๆ ช่วงเวลาที่ซึ่งจะถูกแสดงตามความจริงใน tuple ที่ถูกจองไว้ แทนที่เป็นการเข้ารหัสของชุดของช่วงเวลานั้น (โดยปกติเรียกว่า *periods* ของ Validity)

2.3 ตัวอย่างของ Application

- ด้านการศึกษา เช่น การออกหนังสือรับรองการศึกษาในเทอมก่อนและเทอมปัจจุบัน หรือออกเกรดสำหรับเทอมก่อน
- ด้านบัญชี เช่น บิลอะไรที่มีการออกและออกให้เมื่อไร, บัญชีที่ค้างชำระและเงินหมุนเวียนหมดอายุเมื่อไร
- งบประมาณ เช่น งบประมาณก่อนและหลังโปรเจค ว่างบประมาณในปีต่าง ๆ หรือเดือนต่าง ๆ เป็นเช่นไร
- ด้าน data warehouse เช่น การวิเคราะห์แนวโน้มของประวัติต่าง ๆ สำหรับการทำระบบช่วยตัดสิน
- และอื่น ๆ

โดยเราจะยกตัวอย่างข้อมูลของตารางที่มีความสัมพันธ์ที่ใช้เวลาดังนี้

Employee1:

Name	Title	Start	Stop
Bob	Programmer	1993-01-01	1993-06-01
Bob	Senior-prog.	1993-06-01	1995-01-01
Bob	Junior-SA	1995-01-01	1996-01-01

ตารางที่ 2.1 ข้อมูลของพนักงาน

ในตัวอย่างของตาราง Employee1 ซึ่งเราจะเห็นว่าเป็นการเก็บข้อมูลในลักษณะที่เป็น history และเราสามารถทำการเรียกค้นข้อมูลในแต่ละช่วงเวลาออกมาใช้ได้

ทั้งหมดของตัวอย่างจะเป็นการยากที่จะทำเป็น application ถ้าไม่ใช้การบริหารข้อมูลด้วย temporal

Application ดังกล่าวนั้นจะได้ประโยชน์จากการใช้ temporal ที่สนับสนุนใน DBMS ที่จะทำให้การพัฒนา application นั้นมีประสิทธิภาพมากขึ้น และเพิ่มศักยภาพของการทำงาน

2.4 ข้อดีของการใช้ Temporal

เมื่อมีการพัฒนาให้เป็น Temporal แล้วจึงทำให้มีข้อดีในการจัดการข้อมูลเชิงเวลา เมื่อเทียบกับการใช้ฐานข้อมูลแบบเดิมดังนี้

- โครงสร้างของ semantics จะง่ายขึ้นมากเมื่อนำเสนอด้วย temporal query language
- ผู้ใช้สามารถเขียน temporal queries ในการประกาศตัวแปรที่เข้าใจได้ง่าย
- มันสนับสนุนการให้ความหมายของการ duplicate semantices และ aggregation
- Queries ที่ใช้จะเป็นภาษาที่สามารถทำให้ประสิทธิภาพบนการใช้ temporal relations ดีขึ้น และทำให้มีการเขียน code ที่สั้นขึ้น ประโยชน์ที่ตามมาก็คือ ง่ายต่อการเข้าใจ, ง่ายต่อการตรวจสอบความถูกต้อง, ง่ายต่อการบำรุงรักษา

2.5 ภาษาของ SQL/TP (SQL/temporal)

ในส่วนนี้เราจะกำหนด syntax และ semantics ของ SQL/TP ที่รวมถึง data definition, data query และ data manipulation ที่เป็นส่วนประกอบของภาษา ใน 3 สิ่งนี้เราจะแสดง SQL/TP ที่เป็นธรรมชาติของโครงสร้างภาษาที่เป็นส่วนที่เพิ่มขึ้นจาก SQL ที่อยู่บน temporal database ยิ่งกว่านั้น semantics ของ SQL/TP จำเป็นในการระบุของ semantics ของ SQL ในการเพิ่มศักยภาพของตารางที่ไม่มีจุดสิ้นสุดของเวลา

Data Definition Language

เราจะเริ่มที่ Data Definition Language มันจำเป็นในการระบุมาตราฐานของ SQL-92

Create table <rid> (<signal>)

Create view <rid> (<query>)

เมื่อ <rid> คือตารางที่ใช้และ <signal> คือสัญลักษณ์ที่แทนด้วยตารางใหม่

สำหรับ views เป็นสัญลักษณ์ที่ถ่ายทอดมาจาก <query> สิ่งที่เป็นความแตกต่างของ temporal attributes ที่เราประกาศคือ data type ใหม่ซึ่งคือ time ที่เป็นส่วนที่เพิ่มเติมขึ้นมาเพื่อใช้กำหนดว่า จะทำอย่างไรให้เก็บเวลาลงใน temporal table ได้จริง ๆ

ใช้ points : เป็นค่าคงที่ของเวลาที่เก็บจะเป็นค่าที่ atomic คล้ายกับค่าอื่น ๆ ทั้งหมดใน data types ทางเลือกนี้เป็นชุดที่สามารถแทนค่าเหตุการณ์ที่เป็น atomic คือเหตุการณ์เกิดขึ้นอย่างเฉพาะเจาะจง เช่น เมื่อบางส่วนของ tuple ถูก insert ใน database

ใช้ [bounded | unbounded] interval (เป็นช่วงเวลา) : เป็นเซ็ตของเวลาที่ต่อเนื่องด้วยส่วนของข้อมูลใน tuple ที่มีการ encoded โดยใช้ช่วงเวลา การ encoding นี้เป็นชุดสำหรับแทนช่วงเวลาของเหตุการณ์ เช่น valid time ของ fact (ที่ซึ่งเป็นจริงเมื่ออยู่ในช่วงเวลานี้) bounded และ

unbounded จะมีคำที่เฉพาะคือ ถ้าเป็น -α และ α บางทีอาจใช้เป็นจุดสิ้นสุดของช่องเวลา โดยมีลักษณะดังตารางที่ 2.2 ดังนี้

มันสำคัญที่เข้าใจว่าผลกระทบจากการปรับปรุง จะมีผลกระทบเพียงตารางที่เก็บลงไปเท่านั้น แต่ไม่มีผลต่อ semantices ของ queries (คล้ายกับสิ่งที่เฉพาะเจาะจง เช่น การเรียงลำดับ key ของ table)

ในอนาคตสิ่งนี้เป็นรายการที่อนุญาตในการทำให้เหมาะสมกับการ encodings ที่แตกต่างกัน การปรับปรุงเราจะทำเพียงแค่ใน SQL/TP ขณะที่ผลกระทบของ syntax จะเป็นตัวเลือกในการ encoding โดย default ของการปรับปรุง unbounded time คือสมมุติฐานสำหรับทุก ๆ temporal attributes ที่นอกจากจะมีสถานะที่ชัดเจน

Name	Yearmin...Yearmax
Czech Kingdom	1198 ... 1620
Czechoslovakia	1918 ... 1938
Czechoslovakia	1945 ... 1992
	...
Slovakia	1940 ... 1944
Slovakia	1993 ... α
Poland	1918 ... 1938
Poland	1945 ... α

ตารางที่ 2.2 ตาราง indep

สามารถสร้างเป็น SQL/TP ได้ดังนี้

```
Create table indep
(name char(20), year time using
unbounded intervals)
```

ซึ่ง semantic เหล่านี้ เช่น interval ของเวลาได้มีการกำหนดไว้ในมาตรฐานของ SQL3 ที่เป็น data type ชุดใหม่คือ PERIOD(TIME) และ PERIOD(TIME STAMP) ซึ่งเป็นค่าที่แสดงถึง

ขอบเขตของเวลา ดังตารางที่ 2 ที่เป็น field ของ yearmin-yearmax เพื่อให้สามารถจัดการบริหารข้อมูลด้านเวลาได้สะดวกขึ้น

2.6 มิติของเวลา

ใน temporal databases เวลาสามารถมีความสัมพันธ์กับความจริงที่แตกต่างกัน Valid time ของความจริง คือเวลาเมื่อความจริงคือความถูกต้องในโมเดลที่เป็นจริง ขณะที่ transaction time ของฐานข้อมูลที่เป็นจริงคือเวลาเมื่อความจริงคือฐานข้อมูลที่เป็นจริงในเวลาปัจจุบันและบางทีอาจถูกเรียกคืนได้

Snapshot database ไม่สนับสนุนข้อมูลแบบ temporal (conventional database) ที่ซึ่งไม่มีประโยชน์ในการใช้ฐานข้อมูลนี้ เมื่อจำเป็นที่จะใช้ข้อมูลเชิง temporal เพราะสิ่งนี้คือฐานข้อมูลทั่ว ๆ ไป ที่ไม่มีเวลามาเกี่ยวข้อง

มีคุณสมบัติดังนี้

- สามารถปรับปรุงฐานข้อมูลได้
- ใช้ static query

Valid-time database สนับสนุนเพียง valid-time เมื่อเวลาคือสิ่งจำเป็นต้องมีความสัมพันธ์กับข้อมูล แต่ไม่สนับสนุนข้อมูลที่เป็นประวัติ จึงมีเพียงข้อมูลปัจจุบันเท่านั้นที่จะสนับสนุน ไม่มีการปรับปรุงข้อมูลที่สามารถเข้าถึงได้ valid-time เป็นการเตรียมโดย user และ ข้อมูลเกี่ยวกับอนาคตที่สามารถจะรวมกลุ่มเข้าด้วยกันได้ โดยมี ลักษณะคือมีความต่อเนื่อง คือมีการเปลี่ยนแปลงตามสถานะที่เป็นความจริง

มีคุณสมบัติดังนี้

- บางทีอาจแก้ไขฐานข้อมูลได้
- สนับสนุน valid time
- สนับสนุน historical query

เช่น ตัวอย่างของข้อมูลพนักงาน ดังที่ผ่านมาแล้วในตารางที่ 2.1

Transaction database สนับสนุนเพียง transaction-time มันสนับสนุนข้อมูลที่เป็นประวัติ ดังนั้นมันจึงเข้าถึงข้อมูลที่เป็นอดีตได้ มันไม่สามารถแทนข้อมูลที่เหลื่อมกันในอนาคตได้ มีเพียงข้อมูลของฐานข้อมูลที่ทำงานอยู่จะถูกบันทึก

มีคุณสมบัติดังนี้

- Append ได้เท่านั้น : สิ่งที่ถูกต้องใน snapshot states ก่อนหน้านั้นไม่อนุญาตให้ทำได้
- สนับสนุน transaction time

เช่น ข้อมูลทางดาราศาสตร์ จาก Smithsonian Astrophysical Observatory J2000 Catalogue

RA_ Hour	RA_ min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discovery	Mag_ First	Trans_Start	Trans_Stop
00	00	00	75	30	'A 1248'	12.0	1989-03-12	1992-11-15
00	00	09	75	30	'A 1248'	12.0	1992-11-15	1994-05-18
00	00	09	75	30	'A 1248'	10.5	1994-05-18	1995-07-23
00	00	08	75	30	'A 1248'	10.5	1995-07-23	9999-12-31
05	57	40	00	02	'BU 1190'	6.5	1988-11-08	9999-12-31
04	13	20	50	32	'CHR 15'	15.5	1989-03-12	9999-12-31
01	23	70	-09	55	'HJ 3433'	10.5	1991-03-25	9999-12-31
02	33	10	-09	25	'LDS3402'	10.6	1993-12-19	1996-07-09

ตารางที่ 2.3 ตาราง ข้อมูลทางดาราศาตร์

โดยที่ RA คือตำแหน่ง, Dec คือแสงที่ลดลง, Discovery คือ ชื่อดาว, Mag คือ ความส่องสว่าง

นั่นคือในแต่ละ transaction time สถานะของแต่ละช่วงจะมีการเปลี่ยนแปลง ในแต่ละช่วงเวลาก็เป็นความจริง

Bitemporal database ที่ซึ่งสนับสนุนทั้งคู่ของมิติเวลา (ทั้ง transaction time และ valid time) สิ่งนี้จะให้ฐานข้อมูลยืดหยุ่นและมีประสิทธิภาพ

มีคุณสมบัติดังนี้

- Append ได้เท่านั้น
- สนับสนุน valid และ transaction time
- สนับสนุน rollback ด้วย historical query

ตารางจะประกอบด้วยทั้ง transaction time และ valid time

2.7 โครงสร้างของ TSQL2

โครงสร้างของ TSQL2 ใช้ linear time structure โดยใช้ ขอบเขตของเวลาของ TSQL2 time line ใช้ขอบเขตของจุดสิ้นสุด และจุดเริ่มต้นของเวลา โดยเป็นชนิดของ table ดังนี้

- Snapshot table : จะไม่มีอะไรหลังจาก attributes
- Valid-time state table : AS VALID [STATE] <granularity>
- Valid-time event table : AS VALID EVENT <granularity>
- Transaction-time table : AS TRANSACTION
- Bitemporal state table : AS VALID [STATE] <granularity> AND TRANSACTION
- Bitemporal event table : AS VALID EVENT <granularity> AND TRANSACTION

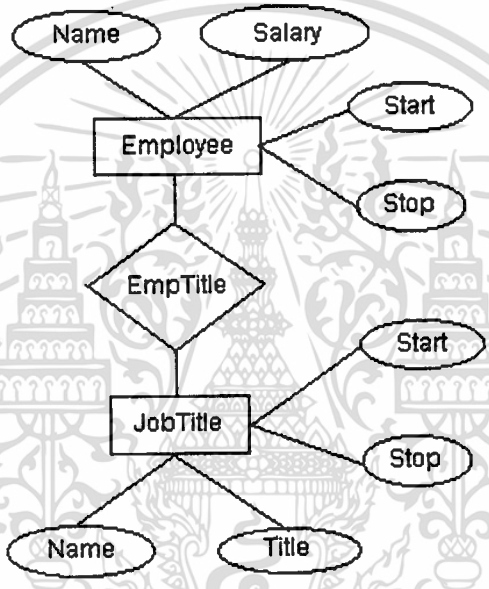
2.8 ข้อแตกต่างในด้านภาษาของ Temporal SQL กับ SQL-92

ในฐานข้อมูลชนิดของ temporal ที่ในปัจจุบันเราจะพบว่ามีการกำหนดออกมาเป็นมาตรฐานโดย ANSI และ ISO นั่นคือ SQL3 แต่ยังไม่มีผลิตภัณฑ์ด้าน DBMS ตัวใดสนับสนุน SQL3 หรือ กล่าวคือยังไม่สามารถนำคุณสมบัติของทางด้านภาษาของ temporal มาใช้ได้อย่างสมบูรณ์ เราจึงจะนำโครงสร้างภาษาของ SQL-92 มาเปรียบเทียบกับ temporal SQL ถึงลักษณะการทำงานดังนี้

ในตารางของ temporal ในทุก ๆ row เราจะกำหนดสถานะแรกเริ่มคือ timestamp ที่เป็นคู่ลำดับของ SQL DATEs นั่นคือ column ของ*เวลาเริ่มต้น* ที่แสดงถึงวันที่เริ่มมีการเพิ่มสถานะเข้าไป และ column ของ*เวลาที่สิ้นสุด* คือการแสดงสถานะของเวลาที่มีการเปลี่ยนแปลง ใน column ที่แสดงถึงช่วงเวลานั้น ๆ คือตัวแสดงสถานะว่า valid นั่นคือ ข้อมูลที่เราสนใจในช่วงของขอบเขต

ของเวลานั้น ความแตกต่างของในตารางเราสามารถใช้ timestamp ให้เป็นค่าที่ใช้เป็นช่วงของเวลาได้ แต่เราจะใช้ได้เฉพาะกับ*เวลาเริ่มต้น*เท่านั้น เพราะเวลาสิ้นสุด อาจจะไม่มีจุดสิ้นสุดของเวลาจริง ๆ ดังนั้นในช่วงของเวลาเราอาจจะใช้วิธีอื่นในการแทนค่าที่ไม่มีจุดสิ้นสุดได้ นั่นคือ ใน SQL-92 ใช้ 9999-12-31 หรืออีกทางเลือกหนึ่งคือแสดงถึงเวลาปัจจุบัน (now) ให้เป็นค่า NULL

เราจะทดสอบลักษณะของภาษาทั้งสองโดยใช้ ER-schema ดังในรูปที่ 2.1 ดังนี้ ซึ่งจะใช้เป็นตารางตัวอย่างในหัวข้ออื่น ๆ ถัดไป



รูปที่ 2.1 ER Diagram ของ Database Design

Create Table

เราจะยกตัวอย่างในการสร้างตารางของ Employee ด้วย ภาษา SQL-92 จะมีลักษณะดังนี้

A
```
CREATE TABLE EMPLOYEE (
    Name CHAR(15),
    Salary DECIMAL(8,2),
    start DATE,
    stop DATE,
    UNIQUE(Name,Salary,start,stop)
)
```

ด้วย Temporal SQL จะมีลักษณะดังนี้

B
```
SET SCALE AS INTERVAL '1' DAY
CREATE TABLE EMPLOYEE(
   Name CHAR(15),
   PRIMARY KEY (Name),
   Salary DECIMAL(8,2),  )
AS VALID STATE
```

Current Modifications

ในการเพิ่มหรือลบข้อมูลในฐานข้อมูลเชิง temporal นั้น ไม่ใช่แต่เพียงเพิ่มหรือลบฐานข้อมูลเดิมแต่จะเป็นการเปลี่ยนแปลงสถานะของเวลา ของช่วงเวลา เพื่อให้ตรงกับเหตุการณ์ที่เกิดขึ้นในช่วงเวลานั้น ๆ ดังในตารางที่ 2.4 เมื่อมีการเพิ่มข้อมูล
ใน Temporal SQL จะมีลักษณะดังนี้

A.
```
INSERT INTO EMPLOYEE
VALUE ('Bob',80000)
VALID PERIOD '[1/1/1994 - now]'
```

Name	Salary	Start	Stop
Bob	60000	1993-01-01	1993-06-01
Bob	70000	1993-06-01	9999-12-31

ตารางที่ 2.4 ตาราง Employee

Name	Salary	Start	Stop
Bob	60000	1993-01-01	1993-06-01
Bob	70000	1993-06-01	1994-01-01
Bob	80000	1994-01-01	9999-12-31

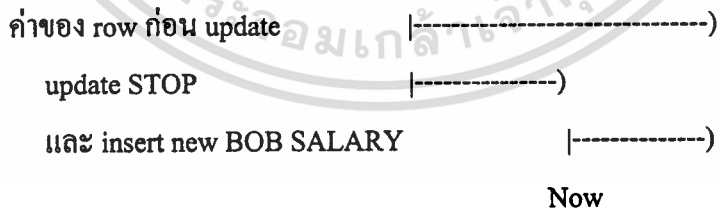
ตารางที่ 2.5 ตาราง Employee ที่มีการ insert ข้อมูลแล้ว

หลังจากที่ได้มีการ insert ข้อมูลแล้วจะมีลักษณะดังตารางที่ 2.5 โดยที่ถ้าเป็น SQL-92 จะมีการทำงาน 3 ขั้นตอน ดังนี้

ให้ CURRENT_DATE คือเวลาปัจจุบันที่มีการเพิ่มข้อมูล หรือเป็นข้อมูลที่เรากำหนดให้

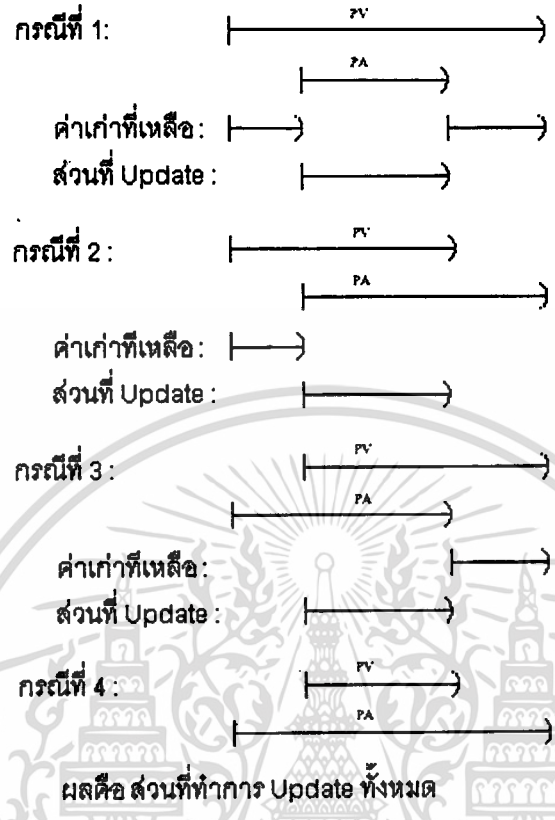
```
UPDATE EMPLOYEE
SET STOP = CURRENT_DATE
WHERE Name = Bob
   AND STOP >= CURRENT_DATE
   AND START < CURRENT_DATE


DELETE FROM EMPLOYEE
WHERE Name = Bob
   AND START > CURRENT_DATE

INSERT INTO EMPLOYEE
VALUE ('Bob',80000,CURRENT_DATE,DATE'9999-12-31')
```

ลักษณะการทำงานจะเป็นดังรูปที่ 2.2 ดังนี้



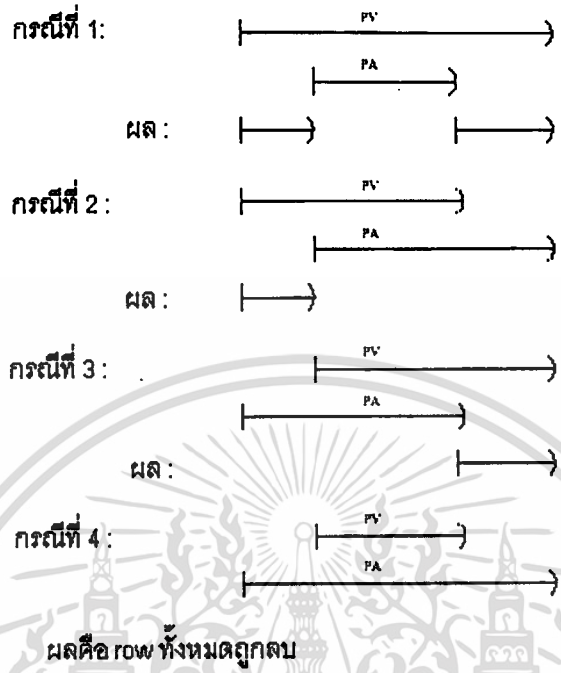
รูปที่ 2.2 การ insert ข้อมูล

จากการปรับปรุงข้อมูลในลักษณะนี้เราสามารถแยกเป็นกรณีการ insert เป็น 4 กรณีดังรูปที่ 3 โดยให้ช่วงเวลาที่เรายอมรับ หรือ period of validity (PV) คือเป็น tuple เดิม และ period of applicability (PA) คือ tupple ที่จะทำการ insert โดยจะมีลักษณะการ UPDATE ข้อมูล และการ INSERT ตามรูปดังกล่าว



รูปที่ 2.3 กรณีของการ Update

จากการ Update เราจะสร้างลักษณะการ deleted ได้อีก 4 กรณี ดังรูปที่ 4 คือ ในกรณีที่ (1) row เดิมจะคลุมทับ PA ดังนั้น ทั้งเวลาเริ่มต้นและสิ้นสุดจะเหลืออยู่ โดยที่เวลาเริ่มต้นจะถูกเปลี่ยนเวลา STOP ให้เป็นเวลา START ของ PA และส่วนท้ายก็จะถูกเปลี่ยนเวลาเริ่มต้นเป็นเวลา STOP ของ PA (2) มีเพียงส่วนต้นของ PV ที่เหลืออยู่เท่านั้น (3) มีเพียงส่วนท้ายของ PV ที่เหลืออยู่ กรณีที่ (4) ทั้งหมดของ PV ถูก PA delete จนหมด



รูปที่ 2.4 กรณีของการ Delete

Temporal Projection และ Selection

ตามความเป็นจริงแล้วการ query ของ conventional (non-temporal) นั้นคือการสืบค้นค่าปัจจุบันอยู่แล้วโดยไม่มีสถานะของเวลามาเกี่ยวข้อง ซึ่งใน temporal จะมีเพิ่มเวลามาช่วยในการสืบค้นข้อมูลด้วยดังตัวอย่างเช่น

ใน Temporal SQL มีลักษณะดังนี้

A
```
SELECT *
VALID PERIOD '[1/3/1993-now]'
FROM EMPLOYEE
WHERE Name = 'Bob'
```

และใน SQL-02 จะมีลักษณะดังนี้

B
```
SELECT *
FROM EMPLOYEE
WHERE Name = 'Bob'
AND START >= '1993-03-01'
AND STOP < '9999-12-31'
```

บทที่3

โครงการศึกษาและพัฒนา Interactive TSQL ของ Valid time และ Transaction time ด้วย SQL-92

3.1 Temporal SQL และการออกแบบ

การนำ temporal SQL มาแปลงเป็น SQL-92 นี้ เราจะแนะนำการทำงาน สถาปัตยกรรมและการออกแบบ เพื่อให้ได้ผลลัพธ์สำหรับการทำงานดังนี้

3.1.1 การทำงานของ Temporal

โดยทั่ว ๆ ไปของ temporal มี 2 ชนิด ตามลักษณะของความจริงในฐานข้อมูล นั่นคือ *transaction time* ของ records ที่เป็นจริง เมื่อความจริงคือความเป็นปัจจุบันในฐานข้อมูล และถูกจัดการโดย temporal DBMS และ *valid time* ของ records ที่เป็นจริง เมื่อความจริงคือความถูกต้องใน ความจริงที่ถูกกำหนดไว้ และถูกจัดการโดย user หรือ ค่า default ที่จัดเตรียมไว้โดย temporal DBMS และ data model หรือ DBMS ที่มี build-in support สำหรับเวลาทั้ง 2 เรียกว่า bitemporal และนอกเหนือจากนั้นที่ไม่มีการจัดการด้านเวลานั่นคือ non-temporal

ตามตัวอย่าง ถ้า relation **Employee** ใน ตารางที่ 3.1 คือ bitemporal reletion ที่เป็นข้อมูลแบบ snapshot ซึ่งจะมี relation records คือข้อมูล department ของ employee และด้วย 4 columns สุดท้ายคือ transaction-tme และ valid-time

Tida	Toy	8-8-1996	19-8-1996	10-8-1996	*NOW*
Anuwat	Sports	12-8-1996	*NOW*	23-8-1996	31-8-1996
Tida	Toy	19-8-1996	*NOW*	10-8-1996	21-8-1996
Tida	Sports	19-8-1996	*NOW*	21-8-1996	*NOW*

ตารางที่ 3.1 Bitemporal Relation ของตาราง Employee

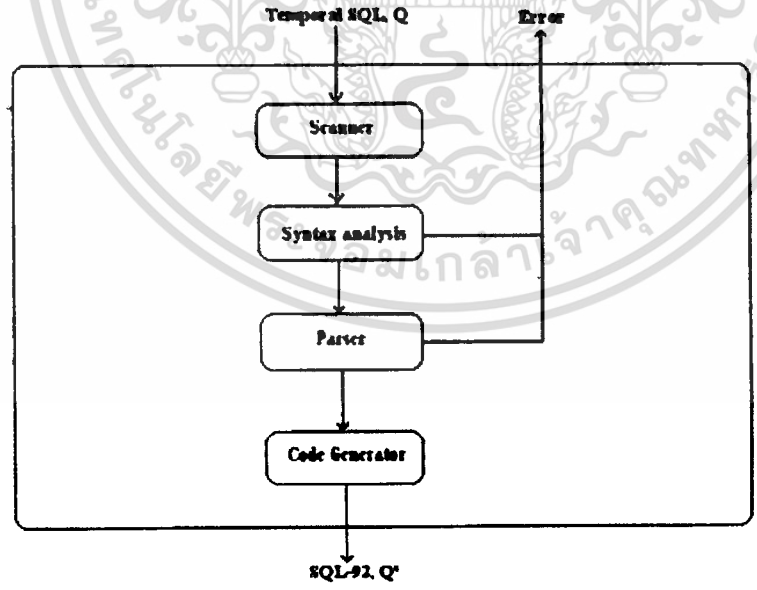
ช่วงวันที่ 8 สิงหาคม ตาราง (Tida , Toy, 8-8-1996, *NOW* , 10-8-4996, *NOW*) ถูก insert หมายถึง Tida เข้ามาทำงานในแผนก Toy ตั้งแต่ 10 สิงหาคม จนถึงปัจจุบัน เหมือนกับวันที่ 12

สิงหาคม เราบันทึกข้อมูลที่ Anuwat เข้าทำงานในแผนก Sports จาก 23 สิงหาคม จนถึง 31 สิงหาคม หลังจาก 19 สิงหาคม เรารู้ว่า Tida เริ่มทำงานในแผนก Sports วันที่ 21 สิงหาคม ความสัมพันธ์ที่เกิดขึ้นในภายหลังจะ update record ที่มีความเชื่อใหม่ แต่เราไม่เชื่อว่า Tida จะทำงานในแผนก Toy จาก 10 สิงหาคม จนกระทั่ง เวลาปัจจุบัน แต่ความเชื่อถูกแทนที่ว่า Tida ควรจะอยู่ในแผนก Toy จนถึง 21 สิงหาคม และควรจะอยู่ในแผนก Sports จาก 21 สิงหาคม จนถึงเวลาปัจจุบัน ดังนั้น *NOW* ใน tuple แรกถูกเปลี่ยนเป็น 19 สิงหาคม และผลของ 2 tuple ใหม่กับความเชื่อใหม่ที่ถูก insert

Relation ในตารางที่ 1 แสดงว่า data model สำหรับ temporal SQL คือสิ่งที่แตกต่างจาก data model ของ SQL-92 และ attribute ทั้ง 4 ถูกเพิ่มใน data structure (relation)

3.2 สถาปัตยกรรมของการแปลภาษา

ในส่วนนี้เราจะแสดงความแตกต่างของกระบวนการในการทำงาน ในความคิดหลักคือเพื่อแตกการทำงาน ลดเวลา และทำให้การทำงานสะดวกขึ้น โดยเป็นการ implements ให้ temporal query สะดวกใน SQL-92 โดยมีโครงสร้างเป็นดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างการแปลภาษา

เราจะเริ่มจากรูป ผู้ใช้ป้อน query Q ที่จะมีการแปล errors ที่ค้นพบระหว่างการทำ parsing จะถูกรายงานออกมา ถ้าไม่มี errors ซึ่งก็จะได้คำสั่งที่เทียบเท่ากันของ SQL-92 ที่จะถูกเรียกว่า Q'

สถาปัตยกรรมนี้ทำให้มันเป็นไปได้ในการแปลงค่าที่ทำให้ได้ผลด้วยความพยายามที่น้อยลงโดยให้ผลมากในการนำกลับมาใช้อีก และทำให้มีประสิทธิภาพในการแปลงข้อมูลได้ดียิ่งขึ้น

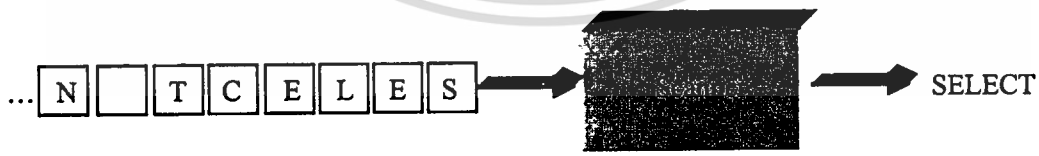
3.2.1 Scanner

ในส่วนของ Scanner นี้จะเป็นการทำงานขั้นแรก ที่ทำการวิเคราะห์โครงสร้างของ input นั่นคือ Q และทำการจัดกลุ่มของ characters เข้ามาเป็นรูปแบบพื้นฐาน

โดยที่การทำงานจะทำการอ่าน characters จาก Q ทีละตัว และทำการจัดโครงสร้างของภาษาให้เป็นกลุ่มของโครงสร้างใหม่ที่ทำให้ขั้นตอนต่อไปมีความเข้าใจและลดความซับซ้อนลง ที่จะการทำงานดังรูปที่ 3.2 ดังนี้ และการตีความก็จะเปรียบเทียบกับ definitions ซึ่งเป็นส่วนที่แบ่งชนิดของ ข้อความต่าง ๆ

ในส่วนของ definitions นี้จะ predefine strings และจัดไว้เป็นส่วน ๆ ดังนี้

Whitespace	[]
Uc_letter	[A-Z]
Lc_letter	[a-z]
Digit	[0-9]
Letter	{lc_letter} | {uc_letter}
Operator	[+ , - , * , / , =]
Parenthesis	[(,)]



รูปที่ 3.2 การทำงานของ Scanner

ตัวอย่างการทำงานใน Scanner จาก source(Q) ที่นำเข้ามาดังนี้

VALIDTIME INSERT INTO EMPLOYEE
VALUE (ANUWAT, SPORTS)

เมื่อผ่านการทำงานในส่วน Scanner ก็จะทำการอ่านข้อมูล และแยกส่วนข้อมูลต่าง ๆ โดยแบ่งเป็น object เป็นส่วน ๆ ดังนี้และตีความดังนี้

VALIDTIME	*Validtime statement*
INSERT INTO	*insert statement*
EMPLOYEE	*table*
VALUE	*statement*
(	*parenthesis*
ANUWAT	*value*
SPORTS	*value*
)	*parenthesis*

ในขั้นตอนนี้จะทำการแยกแยะชนิดของ statement ที่เป็น transactiontime และ validtime รวมถึง SQL statement อื่น ๆ ให้ออกจากกัน

3.2.2 Syntax analysis

Syntax analysis เป็นส่วนที่รับค่าต่อมาจาก Scanner ที่เป็นรูปแบบที่ง่าย ๆ และเป็นกลุ่มที่นำมารวมกันเป็นโครงสร้างที่มีรูปแบบ เพื่อให้เป็นรูปประโยคที่ถูกต้องที่มีความหมายใน temporal SQL และความแตกต่างทั้งหลายก็จะออกแสดงออกมาเป็น error

โดยการทำงานจะอ่านจากค่าที่ได้มาจากการ Scan แล้วตีความหมายของข้อความที่ได้รับมาในขั้นแรก เมื่อรับข้อมูลเข้ามาแล้วก็เปรียบเทียบจากตารางที่เก็บไว้ว่า เป็นชุดข้อมูลที่เราต้องการหรือไม่ เช่น ถ้าเป็น SELECT, INSERT, UPDATE หรือ DELETE ก็จะเป็น SQL statement

ซึ่งในขั้นต้นถ้าไม่ใช่ SQL statement ก็จะแสดง error ออกมาว่า Statement ที่รับเข้ามาไม่ใช่ ประโยคที่เป็น SQL

ถัดมาก็จะตรวจสอบ grammar ของภาษาว่าถูกต้องหรือไม่ ซึ่งคือการพิจารณา syntax ที่เป็นของ statement ต่าง ๆ นั่นเอง

Select

โดยมีรูปแบบพื้นฐานที่ประกอบด้วย 3 ประโยคย่อยดังนี้ **SELECT, FROM** และ **WHERE** และมีตัวอย่างของรูปแบบ SQL query ดังนี้

SELECT A1, A2, …, An
FROM r1, r2, … rm
WHERE P

โดยที่ Ai แทน attribute และ ri แทน relation และ P แทน predicate ถ้ามี WHERE แล้ว P จะต้องมีด้วย

ถ้าผิดรูปแบบไปจากนี้นั่นคือ statement นั้นอาจจะไม่สมบูรณ์ ซึ่งเป็นข้อผิดพลาด และจะทำให้ไม่สามารถแปลงออกเป็น statement ที่ถูกต้องได้ ซึ่งเราจะมีการส่ง messages ออกไปแจ้งให้ทราบถึงข้อผิดพลาด

และในการทำงานใน statement อื่น ๆ ก็จะมีการทำงานเช่นเดียวกัน

Modification

โดยจะมี insert update และ delete ในส่วนนี้

Insert

คือการเพิ่มข้อมูลลงใน relation ครั้งละ 1 tuple ซึ่งจะต้องมี attribute ใน domain โดยจะมีรูปแบบพื้นฐานดังนี้

INSERT INTO Employee
VALUE ('anuwat', Sport)

ในส่วนที่เป็นสีดำเข้มคือส่วนที่เป็น syntax ที่เราต้องใส่

Update

จะทำการ update ข้อมูลทีละ 1 field โดยมีรูปแบบดังนี้

UPDATE Employee

SET Name = 'anuwat'

WHERE Department = 'sport'

ในส่วนการ update ก็เช่นเดียวกันในส่วนที่เป็นสีดำเข้มคือ เป็น syntax ที่จะต้องมี ส่วนตัวเอียงอาจจะมีหรือไม่ก็ได้แล้วแต่กรณี

Delete

เราสามารถ delete ไดใน tuples ทั้งหมด แต่เราไม่สามารถ delete ค่าเพียงบางส่วน ใน attribute ได้ นั่นคือเราจะมี relation เดียวที่ปรากฏในคำสั่ง

โดยจะมีรูปแบบพื้นฐานดังนี้

DELETE FROM r

WHERE P

จากการจัดเรียงโครงสร้างต่าง ๆ ตามรูปแบบนี้ ก็จะทำให้เราแยกชนิดของตัวแปลต่าง ๆ ได้ และสามารถนำไปจำแนดชนิดและออกเป็นผลลัพธ์ได้

ในส่วนโครงสร้างของ ทุก statement ทาง logical คล้ายกับ non-temporal แต่ทาง physical จะมีกระบวนการที่เพิ่มเติมขึ้นมา ซึ่งการกระทำดังกล่าว จะแสดงในรายละเอียดในส่วน parser ต่อไป

3.2.3 Parser generators

เป็นส่วนที่ทำการตีความหมายหลังจากที่ผ่านมา syntax analysis แล้วไม่มีข้อผิดพลาด โดยจะทำการแยกชนิดของ statement และตีความหมายในแต่ละชนิดดังนี้

Update Processing

Relations ที่สนับสนุน transaction time จะเป็นการ append เท่านั้น สิ่งนี้หมายถึง เช่น logical deletion สำหรับ tuple จาก relations นั้น คือการแปลงเป็น updates และ insertions บน snapshot tables ที่เราแสดงการทำงานนี้ในเรื่อง การศึกษาพัฒนา Interactive TSQL ของ Valid time และ Transaction time ด้วย SQL-92 ในส่วนนี้เราจะอธิบายถึงทำให้ temporal นั้น insert, delete, และ update statements บน bitemporal relations ถูกเปลี่ยนเป็น conventional insert และ update statements บน SQL-92 reletaions ในสิ่งที่จะทำนี้ เราจะแสดงความแตกต่างระหว่าง sequenced, และ non-sequenced modifications

ข้างล่างนี้ เราจะแสดง concepts โดยแสดงความหมายตามตัวอย่างดังนี้

Insertion

Temporal query language คือ สมมุติฐานที่รักษาช่วงเวลาของ validtime ของ tuples ที่จัดการโดยผู้ใช้เมื่อ tuple ทำการ insert สิ่งนี้หมายถึง tuples ที่ทำการ insert นี้จะรวมค่าที่เหมือนกันมาเชื่อมต่อกัน

เราจะแสดงการ insertion โดยใช้ statement ในรูปที่ 3.3A ด้วย. **Employee** ที่เป็น relation ในรูปที่ 3.3 ให้นึกถึง Temporal compatable statements คือโครงสร้างของ SQL-92 ที่ถูกต้อง แต่เป็นการประยุกต์สู่ temporal relations โดยที่ Temporal compatable insertion ต้องมี timestamp inserted tuple ดังนั้นจะมีค่าใน current state (ทั้ง valid time และ transaction time) เริ่มต้นที่ เวลาที่ทำการ insert ในวิธีนี้ Temporal compatable statement อื่น เช่น query และ constraints บางทีก็ง่ายในการ implement โดย timeslicing ของ argument temporal tables ทั้งหมดของ current time (ในทั้ง valid time และ transaction time) และเมื่อประเมิน (SQL-92) query บน snapshots ตามตัวอย่างนี้

A INSERT INTO Employee
VALUE ('Kim','Shoes')

B INSERT INTO Employee VALUES
('Kim','Shoes', CURRENT_TIME, 31-12-999
CURRENT_TIME, 31-12-99)

รูปที่ 3.3 Temporal compatable Insertion

ในตัวอย่าง modification inserts ของ employee Kim ผู้ซึ่งอยู่ใน แผนกรองเท้า temporal insert statement ถูกเปลี่ยนเป็น SQL-92 insert statement ใน รูปที่ 3.1B มันกำหนดเป็นเวลาปัจจุบันที่ transaction-time start และ valid-time begin และ 31-12-9999 (ค่าสูงสุดของวันที่ ที่ให้เป็นขอบเขตที่ไม่มีอันสิ้นสุดของเวลาปัจจุบัน) ที่ transaction-timestop และ valid-time end วิธีนี้ tuple ที่ถูก insert จะมีคุณสมบัติสำหรับ query บน current state และดังนั้นค่าที่ถูกต้องของ Temporal compatable จะถูก insert ลงใน tuple

Sequenced และ non-sequenced จะมีการ insertions ที่มีการจัดการที่คล้ายคลึงกัน (จะมีอธิบายในส่วนถัดไปที่เกี่ยวกับการ deletion และ update) อย่างไรก็ตาม การ insert รวมถึงลักษณะเฉพาะเจาะจงของช่วงเวลาของ valid-time ที่มีความสัมพันธ์กับการ insert ของ tuple transaction-time timestamps จะถูกจัดการโดยตัวอย่างที่แสดงข้างบน

Deletion

Temporal deletion statement รวมถึงการยืนยันลักษณะของ tuples ที่จะทำการ delete ด้วย การ delete ของ Temporal compatable จะ delete ที่ tuples ดังนั้นจะไม่มีอะไรหายไป ในสถานะของ current transaction-time และ valid-time states มันเพียงแต่เปลี่ยนแปลงค่าในช่วงเวลาที่เป็น current เท่านั้น

การ deletions แบบ Sequenced และ non-sequenced ที่รวมถึงลักษณะเฉพาะของช่วงเวลาสำหรับ valid-time period ที่ซึ่ง delete คุณสมบัติของ tuples (จาก current transaction-time state) เช่น deletion จะถูกเปลี่ยนเป็นค่าที่เป็น update ของ SQL-92 ค่า attribute ของ transaction-time stop ไปเป็น current time สิ่งนี้จะทำให้เกิดการ insertions ตั้งแต่ 0-2 ครั้ง ที่จะสะท้อนให้เห็นการเปลี่ยนของ time attributes ความจำนวนที่แตกต่างกันของการ insert ซึ่งจะมีกรณีที่แตกต่างกัน 3 กรณีดังนี้

- valid-time period ของ tuple ที่ deleted คือค่าทั้งหมดที่บรรจุในช่วงเวลาที่กำหนดใน temporal delete statement ในกรณีนี้ ไม่จำเป็นต้อง insert ซึ่งตรงกับ กรณีที่ 4 ในบทที่2 ที่เป็นกรณีของการ delete
- valid-time period ของ tuple ที่ deleted มี overlaps ด้วยช่วงเวลาที่กำหนดใน delete statement และในส่วนที่ไม่ overlap โดย ช่วงเวลาที่กำหนดมีช่วงเวลาเดียว ในกรณีนี้ จะมีการ insert 1 ครั้ง ซึ่งตรงกับ กรณีที่ 2 และ 3 ในบทที่2 ที่เป็นกรณีของการ delete
- valid-time period ของ tuple ที่ deleted มี overlaps ด้วยช่วงเวลาที่กำหนดใน delete statement และในส่วนที่ไม่ overlap โดย ช่วงเวลาที่กำหนดประกอบด้วยเวลา 2 ช่วง ในกรณีนี้จะมีการ insert 2 ครั้ง ซึ่งตรงกับ กรณีที่ 1 ในบทที่2 ที่เป็นกรณีของการ delete

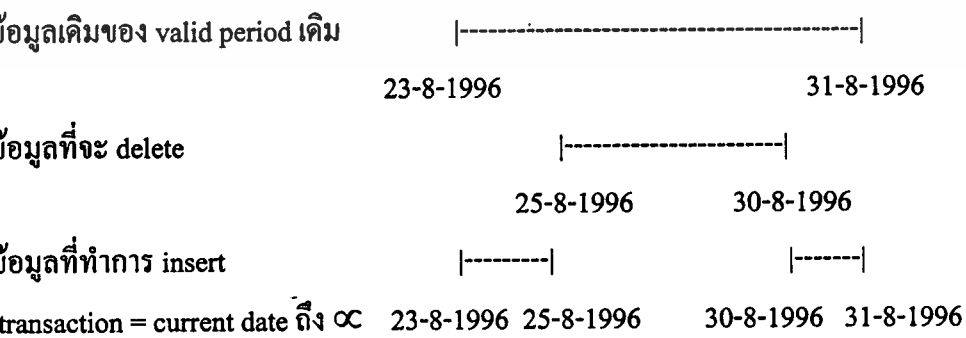
ตัวอย่างที่แสดงในกรณีสุดท้าย พิจารณา sequenced deletion ในรูปที่ 3.4 ที่ซึ่ง deleteds Anuwat จาก Sports department ในช่วงเวลา [25-8-1996, 30-8-1996) (รูปที่ 1 แสดงถึง Anuwat บันทึกข้อมูลปัจจุบันใน Sports จาก 23-8-1996, 31-8-1996)

A
```
DELETE FROM Employee
WHERE Name = 'Anuwat' AND Dept = 'Sports'
VALID PERIOD '25-8-1996 - 30-8-1996'
```

B
```
UPDATE Employee SET T-Stop = CURRENT_TIME
WHERE Name = 'Anuwat' AND Dept = 'Sports' AND T-Stop = 31-12-9999
INSERT INTO Employee VALUES ('Anuwat','Sports', CURRENT_TIME,
                                        31-12-9999, 23-8-1996, 25-8-1996)
INSERT INTO Employee VALUES ('Anuwat','Sports', CURRENT_TIME,
                                        31-12-9999, 30-8-1996, 31-8-1996)
```

รูป 3.4 Sequenced Deletion

Statement ถูกเปลี่ยนเป็น SQL-92 update และ inserts ในรูปที่ 3.4B ครั้งแรก tuple จะ deleted จาก logically deleted โดยการเซ็ตค่า ของ transaction-time stop เป็น current time เพราะช่วงเวลาที่ถูก delete [25-8-1996, 30-8-1996) โดยแยกค่าเก่าของ valid-time period, [23-8-1996, 31-8-1996) เข้าเป็น2 ช่วงเวลา 2 tuples ต้อง insert ใน record ของข้อมูลนี้ การ insert ครั้งแรกต้องครอบคลุม valid time period [23-8-1996, 25-8-1996) และในการ insert ครั้งที่สองจะคลุมช่วงเวลา [30-8-1996, 31-8-1996) transaction-time เริ่มต้นทั้ง insert ที่เป็นเวลาปัจจุบัน และ transaction-time คือ 31-12-9999 ซึ่งมีลักษณะการทำงานดังรูปที่ 3.5 นี้



รูป 3.5 ลักษณะการทำ Sequenced Deletion

Non-sequenced deleteions คือการจัดการ sequenced deletions ที่คล้ายคลึงกัน และ Temporal compatable deletions จะจัดการโดยการเซ็ตค่าของ transaction-time stop เป็น current time

Update

ใน snapshot relation การ update สามารถมีโครงสร้างที่ deletion จากการ insertion ซึ่งเป็นความคิดเดียวกับการประยุกต์ใน bitemporal relation ความหมายนี้คือ tuples ที่ updates โดยครั้งแรกจะทำการ delete โดยการเซ็ตค่าของ transaction-time stop ที่ current time ดังนั้นจะมีการ insert ได้ตั้งแต่ 1 ถึง 3 ครั้งได้ดังนี้ โดยที่จะสะท้อนให้เห็นถึงค่าใหม่และเปลี่ยนแปลง time attributes (เหมือนกับ 3 กรณีของการ deletion ที่จะต้องพิจารณา)

ตามตัวอย่าง พิจารณา nen-sequenced update ในรูป 3.6A ที่ซึ่ง updates ค่า EMPLOYEE relation (ในรูปที่ 1) ที่ record ที่ Anuwat คือใน Shoe department ระหว่าง วันที่ 30 สิงหาคม และ 10 กันยายน 1996

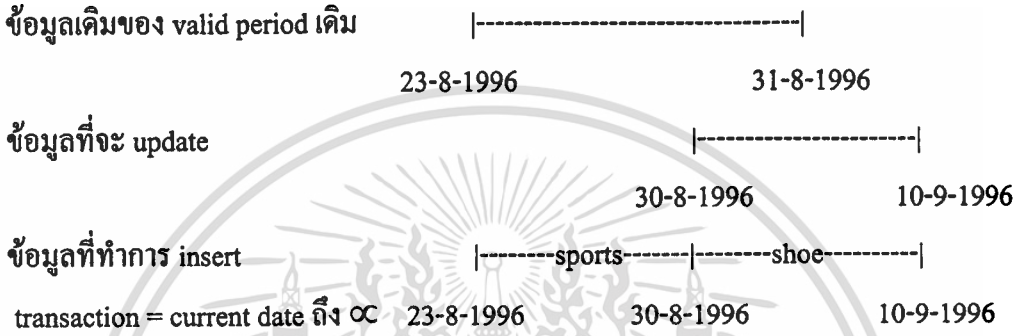
A
```
UPDATE Employee SET Department = 'Shoe'
WHERE Name = 'Anuwat'
VALID PERIOD '30-8-1996 - 10-9-1996'
```

B
```
UPDATE Employee SET T-Stop = CURRENT_TIME
WHERE Name = 'Anuwat' AND T-Stop = 31-12-9999
INSERT INTO Employee VALUES ('Anuwat','Sports', CURRENT_TIME,
                                    31-12-9999, 23-8-1996, 30-8-1996)
INSERT INTO Employee VALUES ('Anuwat','Shoe', CURRENT_TIME,
                                    31-12-9999, 30-8-1996, 10-9-1996)
```

รูปที่ 3.6 Non-sequenced Update

การ update จะถูกเปลี่ยนเป็น SQL-92 update และ insertions ในรูปที่ 3.6B ครั้งแรก เราจะทำการ logical delete tuple ที่ update โดยการเซ็ตค่า attribute ของ transaction-time stop ที่ **current_time** เพราะว่า ลักษณะเฉพาะของการ update คือช่วงเวลาของการ valid-time ที่มีการ

เหลื่อมกันของช่วงเวลา valid-time ของ tuple ในฐานข้อมูลครั้งแรกเราจะ insert tuple ที่มีผลกระทบ คือ Anuwat ไม่อยู่ใน Sports department นานพอ ๆ กับ record แรกที่บันทึก เช่น จาก 23 สิงหาคม ถึง 30 สิงหาคม เมื่อนั้นเราจะ insert tuple ด้วย department และ valid-time period ใหม่ ซึ่งจะมีลักษณะการทำงานดังรูปที่ 3.7



รูป 3.7 ลักษณะการทำ Sequenced Update

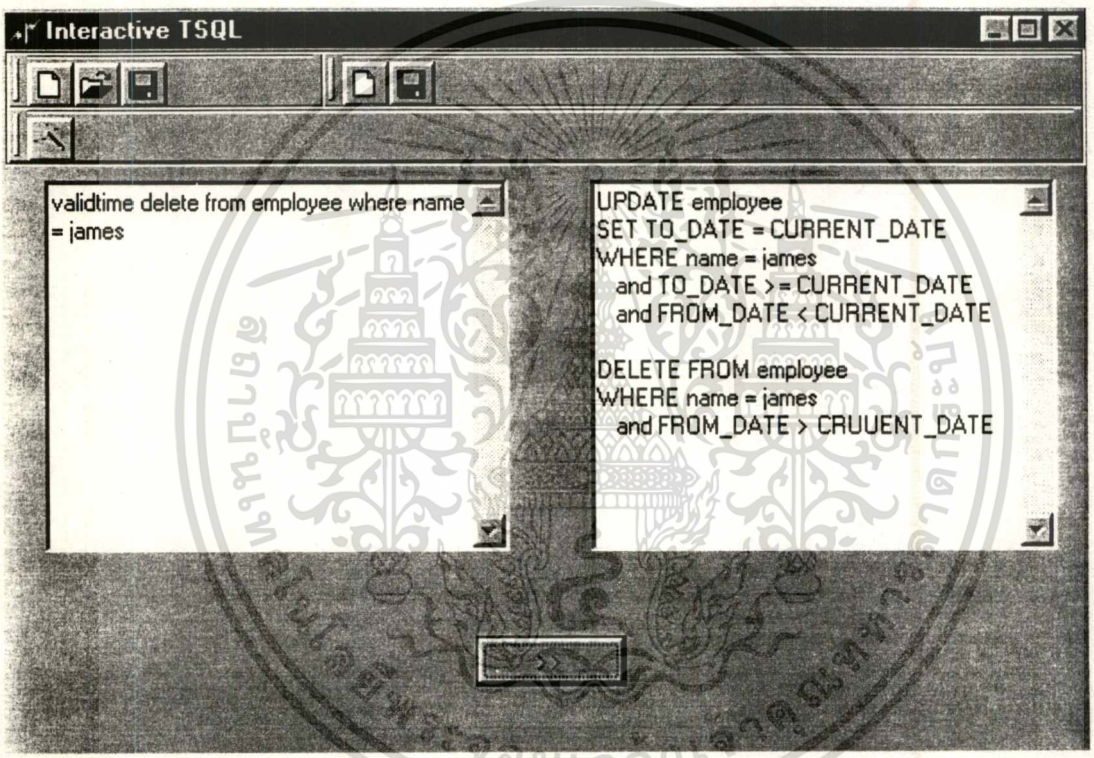
Sequenced updates จะถูกจัดการคล้ายกับ non-sequenced updates ด้วย นอกจาก valid-time periods ที่ติดไปกับ ผลลัพธ์ของ tuples จาก update ที่ถ่ายทอดมาจาก information ในฐานข้อมูล ไม่เพียงแต่ valid-time period ที่ถูกระบุโดย user ยิ่งไปกว่านั้น ผลลัพธ์ของเวลาของ tuples จะพบที่ intersections ระหว่าง tuples ที่มีอยู่ และ time period ที่กำหนด ใน query

Temporal compativle update คือ รูปแบบของ Temporal compativle deletion ตามด้วย Temporal compativle insertion และผลคือ 1 tuple ถูก insert

3.2.4 Code Generation

หลังจากที่ผ่านกระบวนการ parser แล้ว นั่นคือนำเอา syntax ที่ถูกต้องแล้วนำมา mapping กับความสัมพันธ์แล้ว โดยอาศัยตรวจสอบจากตารางที่มีการเทียบกันแล้ว ใน code generation ก็จะนำเอา wording ที่ตรงกันนำมา รวมกันจนได้ผลลัพธ์ที่ได้มาแสดงผล

โดยตัวอย่างของ output ที่ได้จะคล้ายดังรูปที่ 3.8 ดังนี้



รูปที่ 3.8 ผลลัพธ์ที่ได้จากการทำงาน

โดยที่เป็นการ delete ในภาษา temporal ที่เป็น logical current deletion บน valid-time state จึงเป็น current deletions จาก “now” ถึง “forever” และจะทำให้เกิด Current modification ที่จะเป็นการ update

ในกระบวนการนี้ก็จะเป็นการแสดงผลลัพธ์ที่ได้จากการ parsing และนำมาสร้างผลให้ออกเป็น output ที่เป็นการนำเอา temporal SQL เปลี่ยนมาเป็น SQL-92 ที่เท่าเทียมกัน

บทที่4

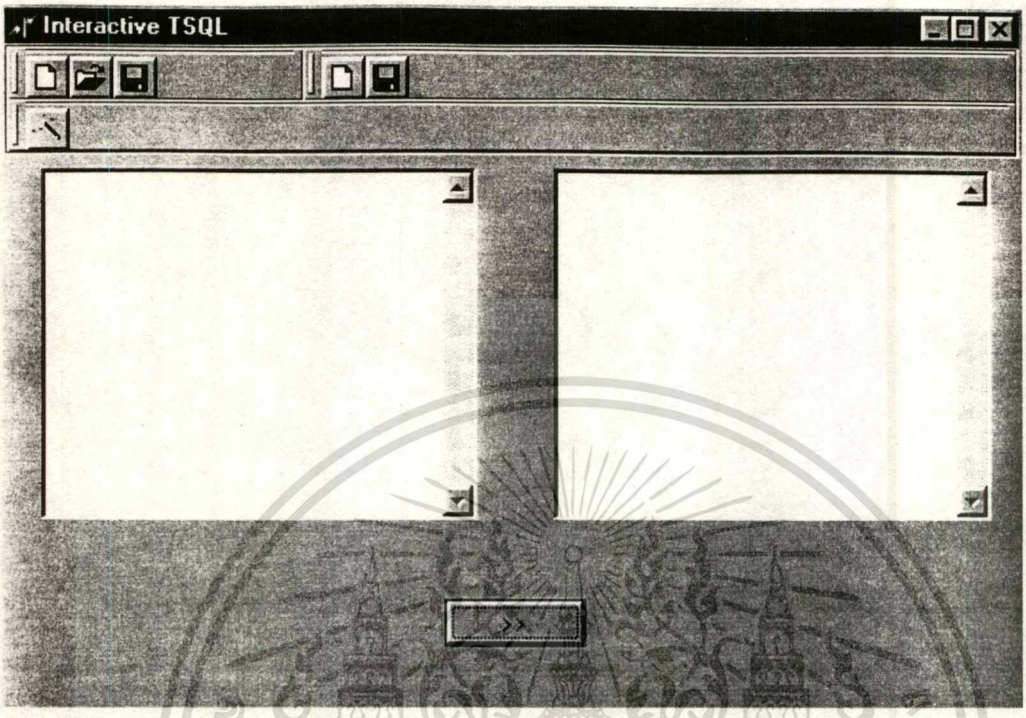
การทำงานของระบบ Interactive TSQL ของ Validtime และ Transactiontime

ระบบ Interactive TSQL ของ Validtime และ Transactiontime เป็นระบบที่ทำการแปลงภาษาของ temporal SQL ให้เป็นรูปแบบภาษามาตราฐานของ SQL-92 ที่มีลักษณะการทำงานเท่ากัน โดยจะมีการทำงานที่สำคัญดังนี้

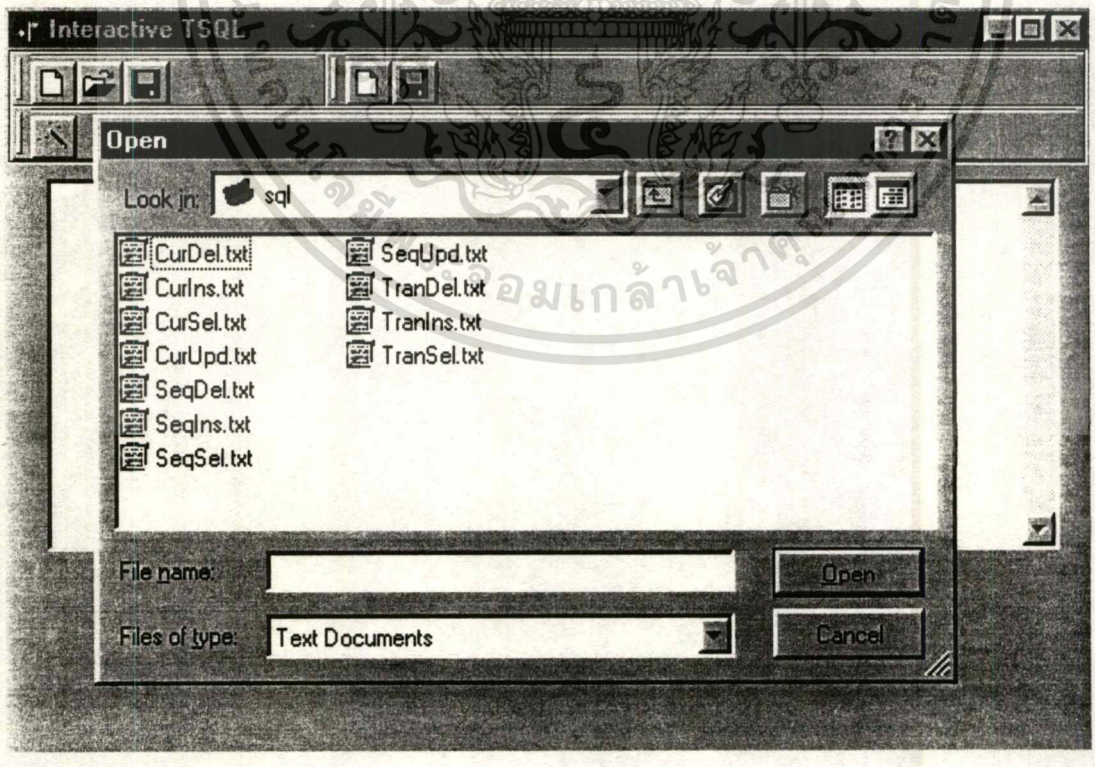
ป้อนคำสั่งที่อยู่ในรูปแบบของ Temporal SQL ที่มีลักษณะของเวลาเป็น Validtime และ transactiontime ที่ถูกต้องลงในช่องด้านซ้ายมือ

โดยที่เราสามารถทำการ save หรือ load ได้จากการ click เมนู icon ที่อยู่ทางด้านบน โดยจะมีการทำงานดังนี้

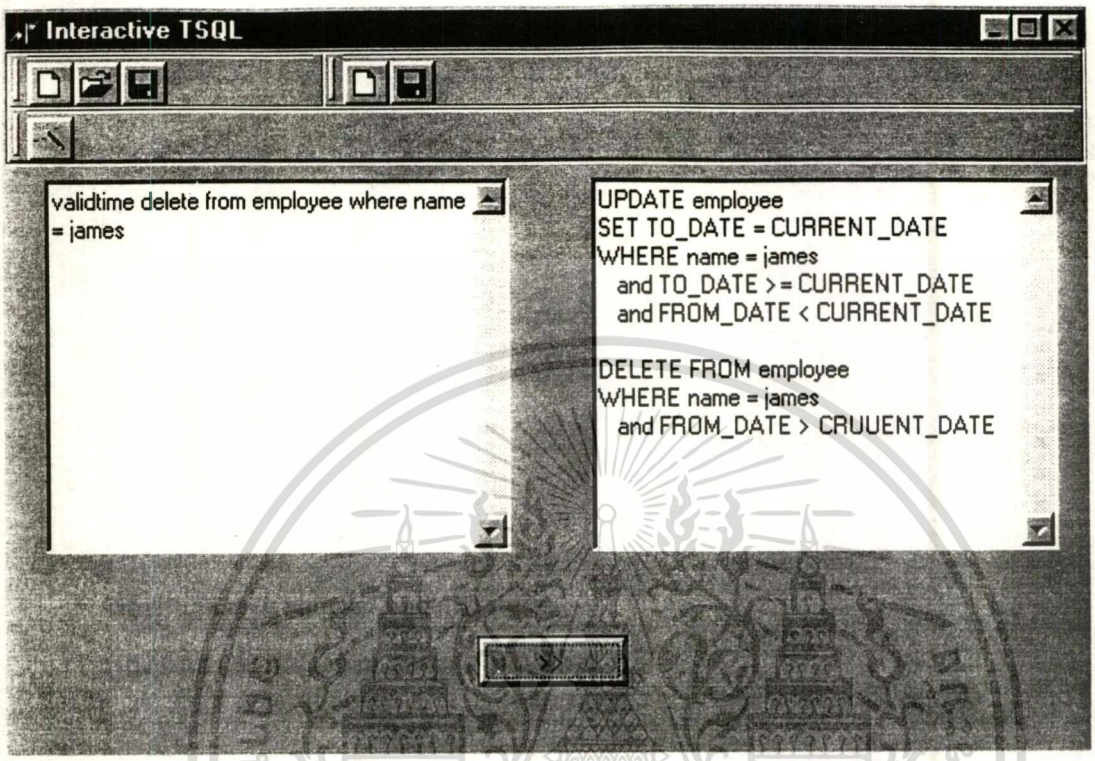
-  เมื่อต้องการจะ clear ค่าให้เป็นหน้าจอที่ว่าง พร้อมที่จะป้อนคำสั่ง SQL ลงไป
-  เมื่อต้องการที่จะเปิดข้อความที่ได้รับการบันทึกไว้
-  เมื่อต้องการที่จะบันทึกข้อความที่ต้องการ
-  เป็น icon wizard



รูปที่ 4.1 แสดงหน้าจอการทำงาน

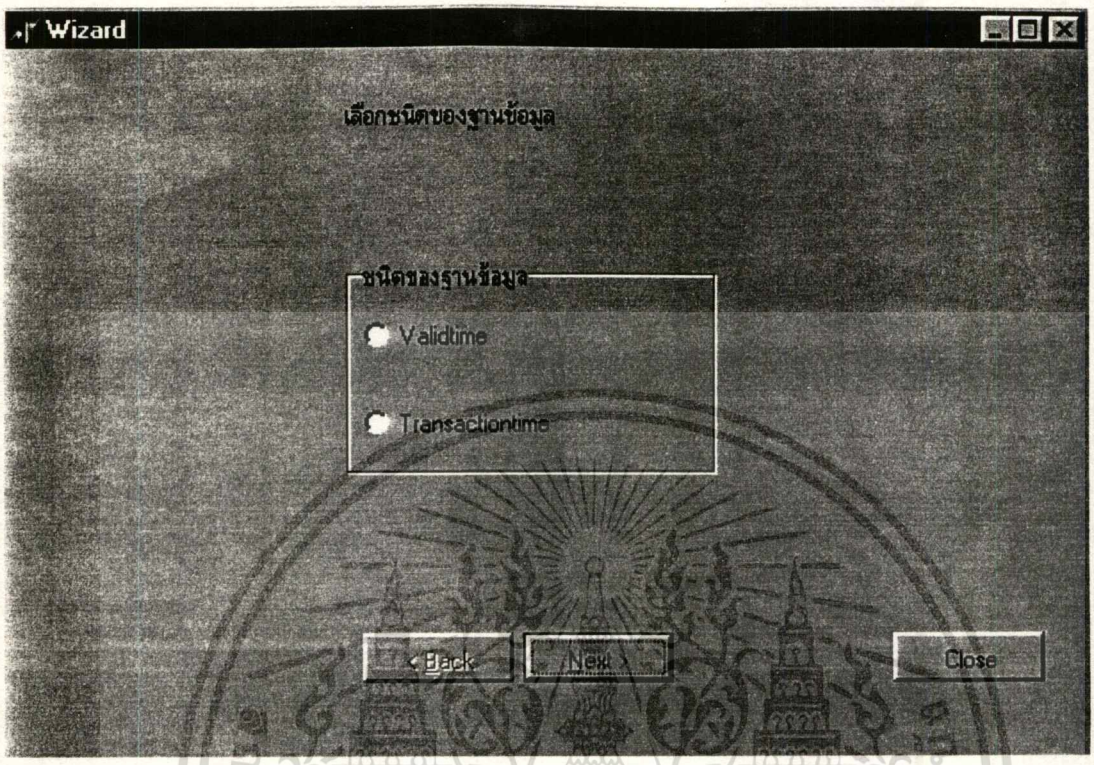


รูปที่ 4.2 แสดงหน้าจอการ load



รูปที่ 4.3 ผลจากการทำงาน

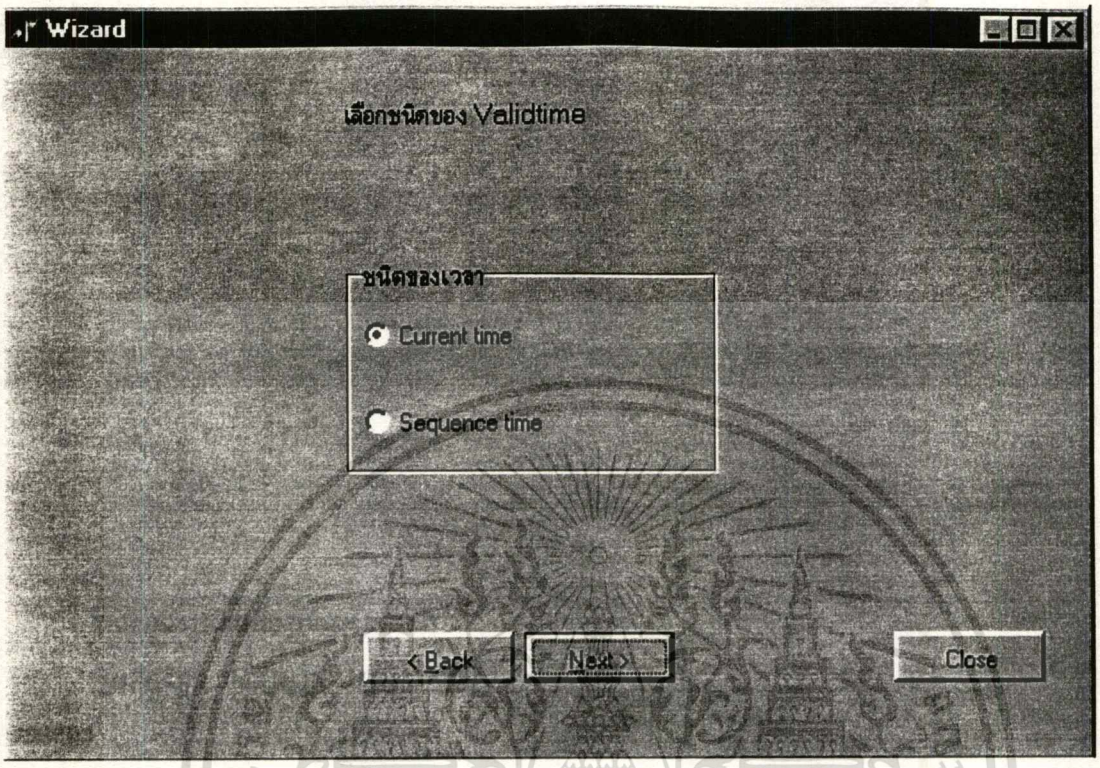
การทำงานของ wizard จากการกดปุ่ม  ข้างบน โดยจะมีลักษณะการทำงานดังนี้ โดยให้เลือกชนิดของข้อมูลด้านเวลาที่เป็น validtime และ transaction



รูปที่ 4.4 แสดงการเลือกชนิดของเวลา

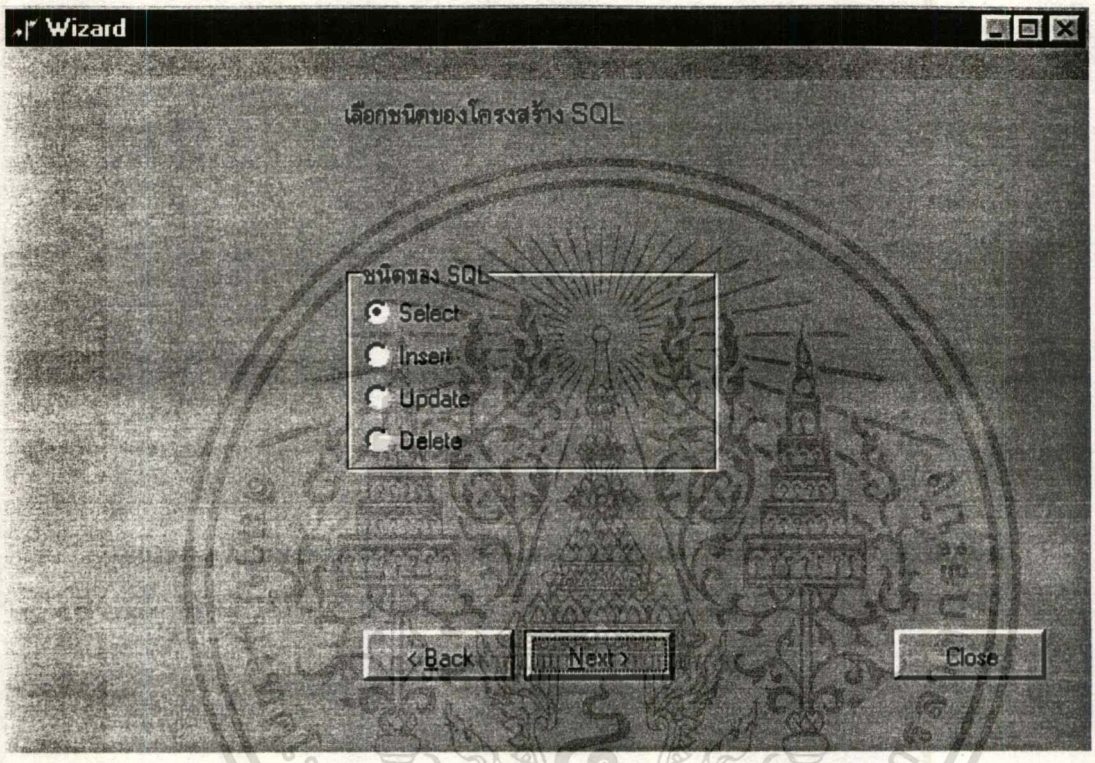
ถ้าเลือกข้อมูลที่เป็น validtime จะสามารถเลือกชนิดของเวลาได้อีก 2 ชนิดคือ current time และ sequence time

โดยที่ current time จะมีไม่แสดงขอบเขตของเวลา



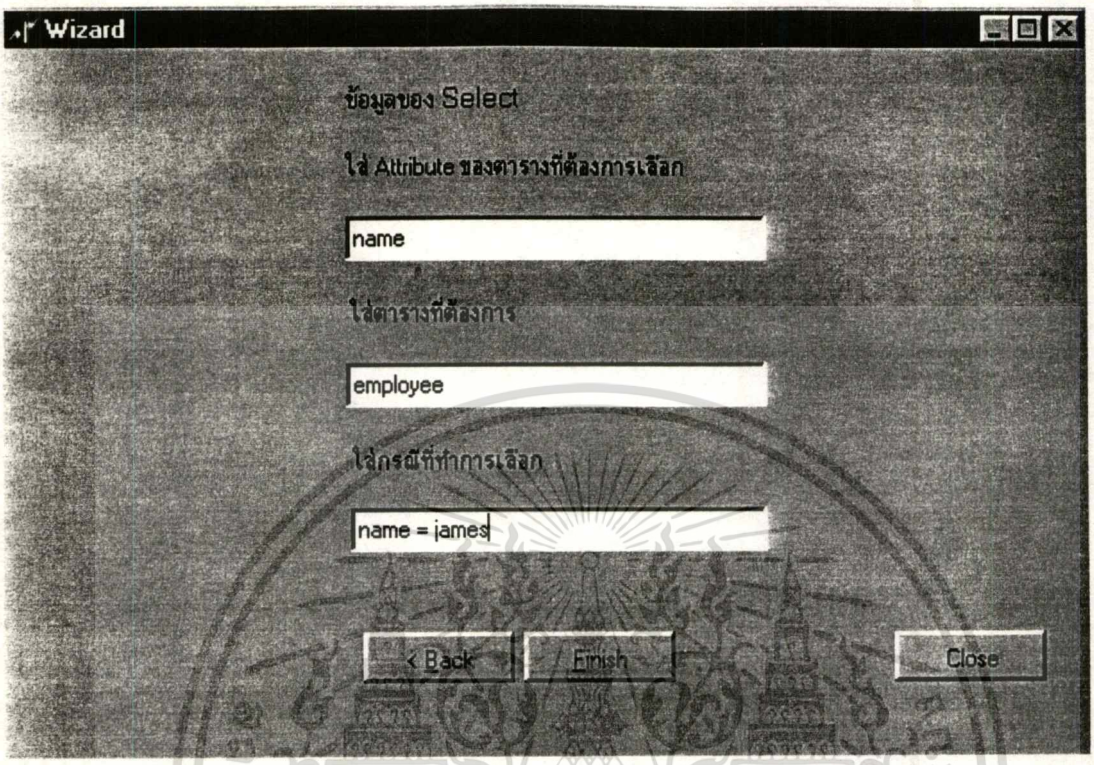
รูป 4.5 แสดงการเลือกชนิดของเวลา

หลักจากเลือกแล้วก็จะเป็นการเข้าสู่หน้าจอของการเลือก SQL statement



รูปที่ 4.6 เลือกชนิดโครงสร้างของ SQL

โดยจะมี 4 ชนิดดังนี้คือ select insert update และ delete โดยจะมีความแตกต่างกันดังนี้



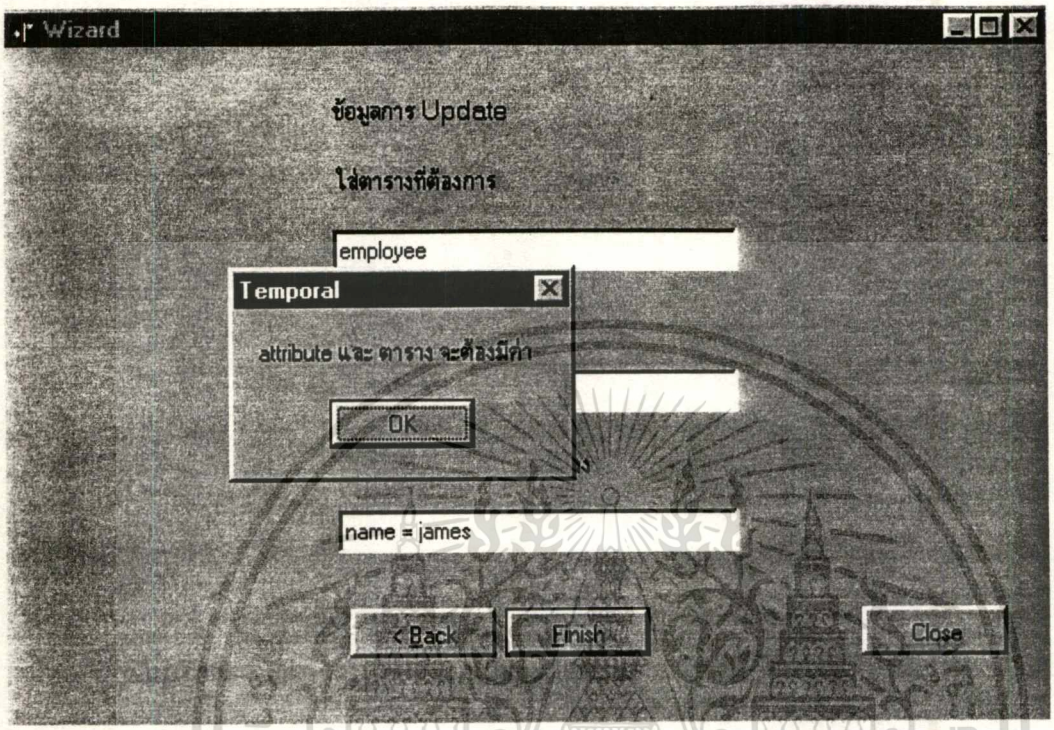
รูปที่ 4.7 หน้าจอการ Select

ในช่องแรกจะให้ใส่ attribute ของตารางที่จะทำการ select

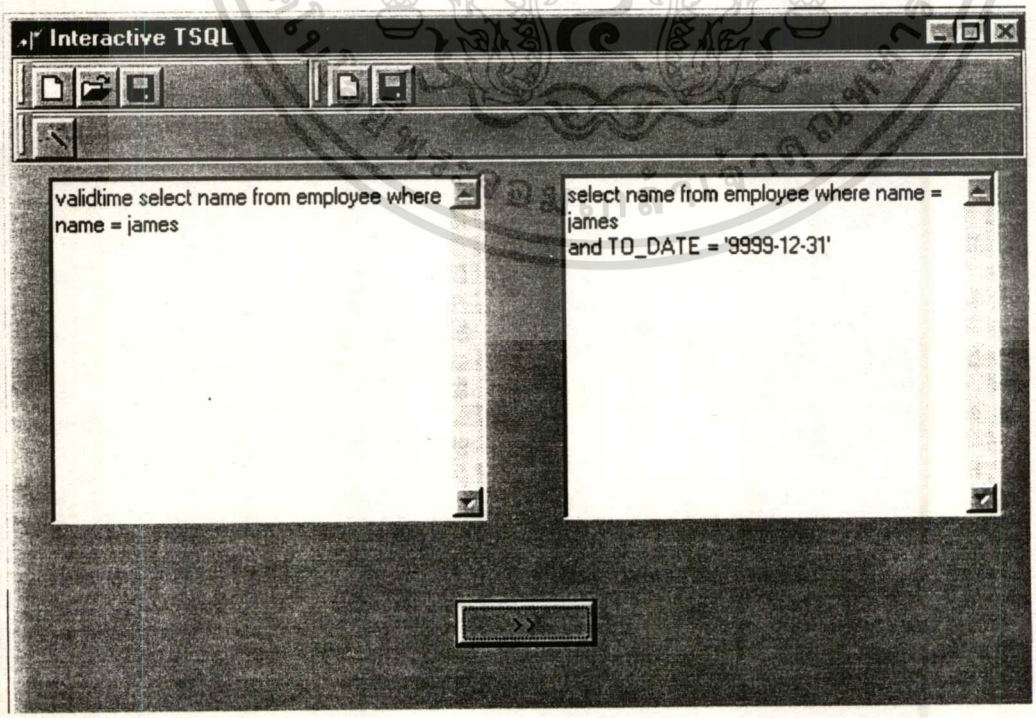
ในช่องที่สองจะให้ใสตารางที่ต้องการ ใส่ได้เพียงตารางเดียวโดยโปรแกรมจะทำการตรวจสอบเอง

ในช่องที่สามจะให้ใส่กรณีที่จะเลือก

และต้องใส่ข้อมูลในช่องให้ครบ

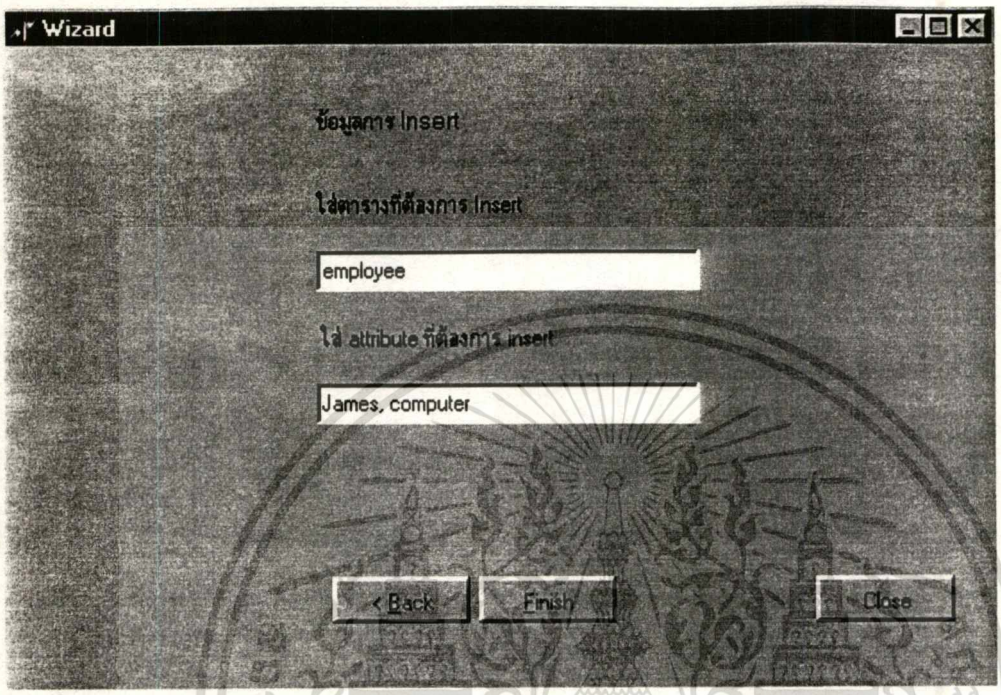


รูปที่ 4.8 แสดงข้อผิดพลาดในการใช้งาน



รูปที่ 4.9 ผลจากการทำงานของ validtime select

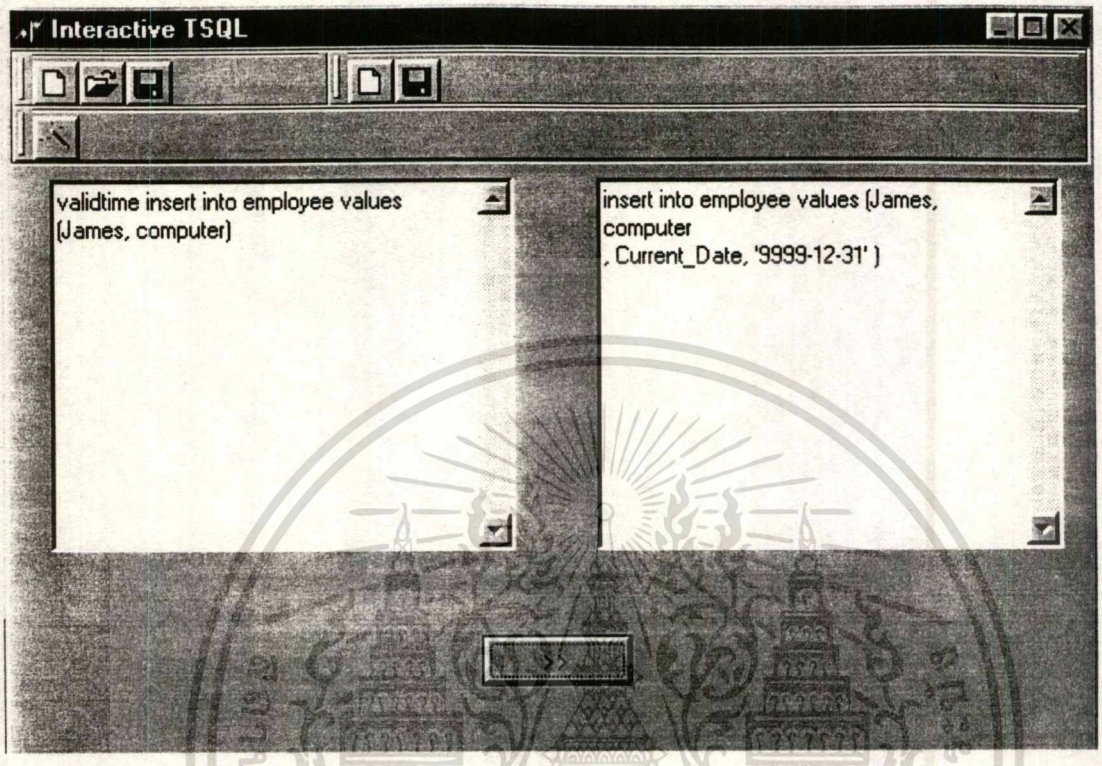
การทำงานของการ insert



รูป 4.10 การจัดการข้อมูลการ insert

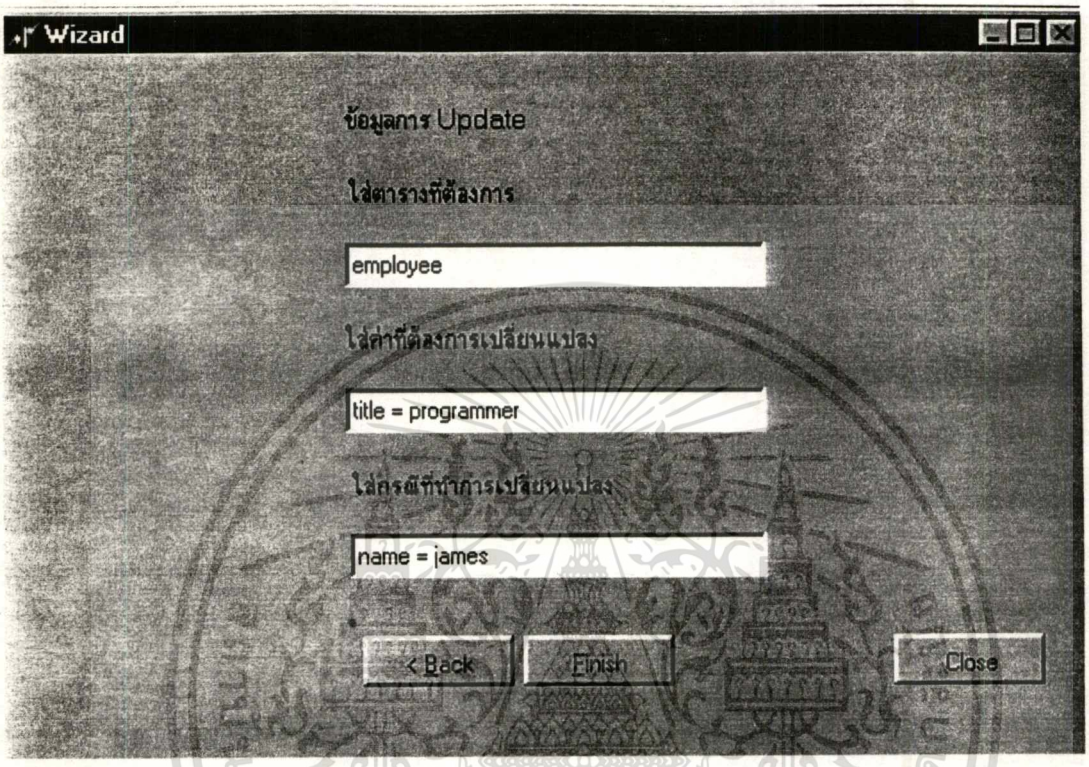
ในช่องแรกจะใส่ค่าของตารางที่ต้องการ insert

ช่องที่สองจะใส่ attribute ที่ต้องการ insert



รูปที่ 4.11 ผลของ validtime insert

การทำงานของการ update



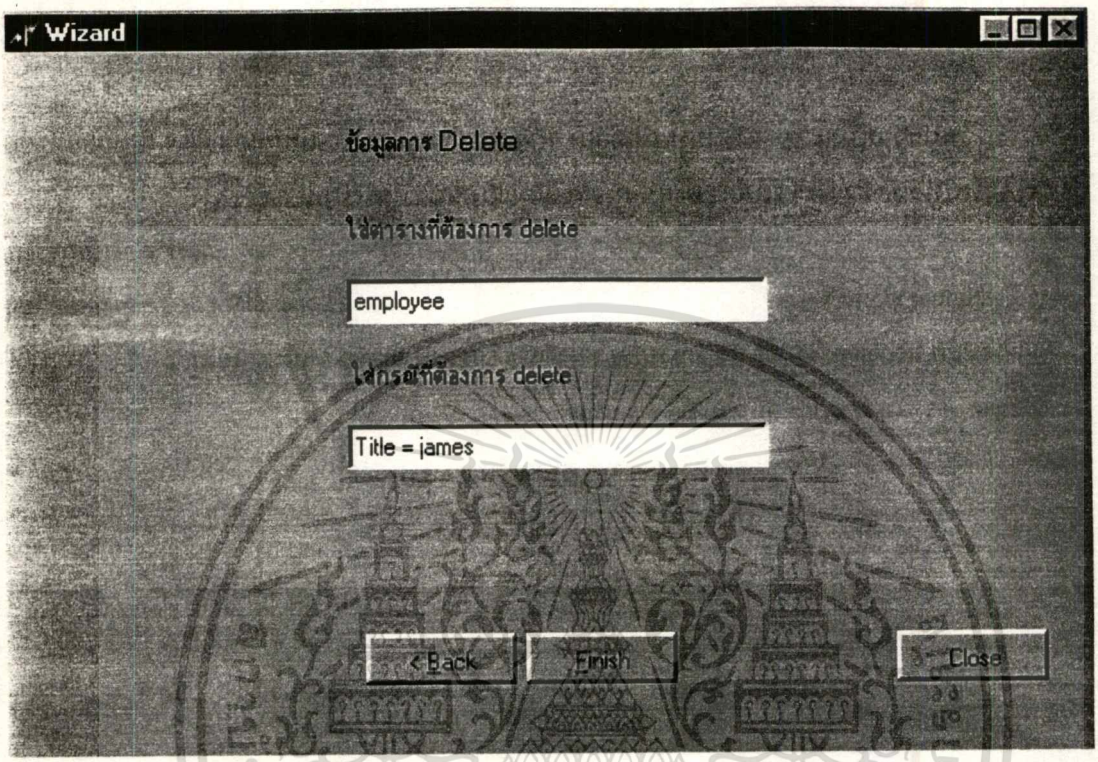
รูปที่ 4.12 หน้าจอป้อนข้อมูลการ update

ในช่องแรกจะให้ใส่ attribute ของตารางที่จะทำการ update

ในช่องที่สองจะให้ใสตารางที่ต้องการ ใส่ได้เพียงตารางเดียวโดยโปรแกรมจะทำการตรวจสอบเอง

ในช่องที่สามจะให้ใส่กรณีที่จะ update

การทำงานของการ Delete

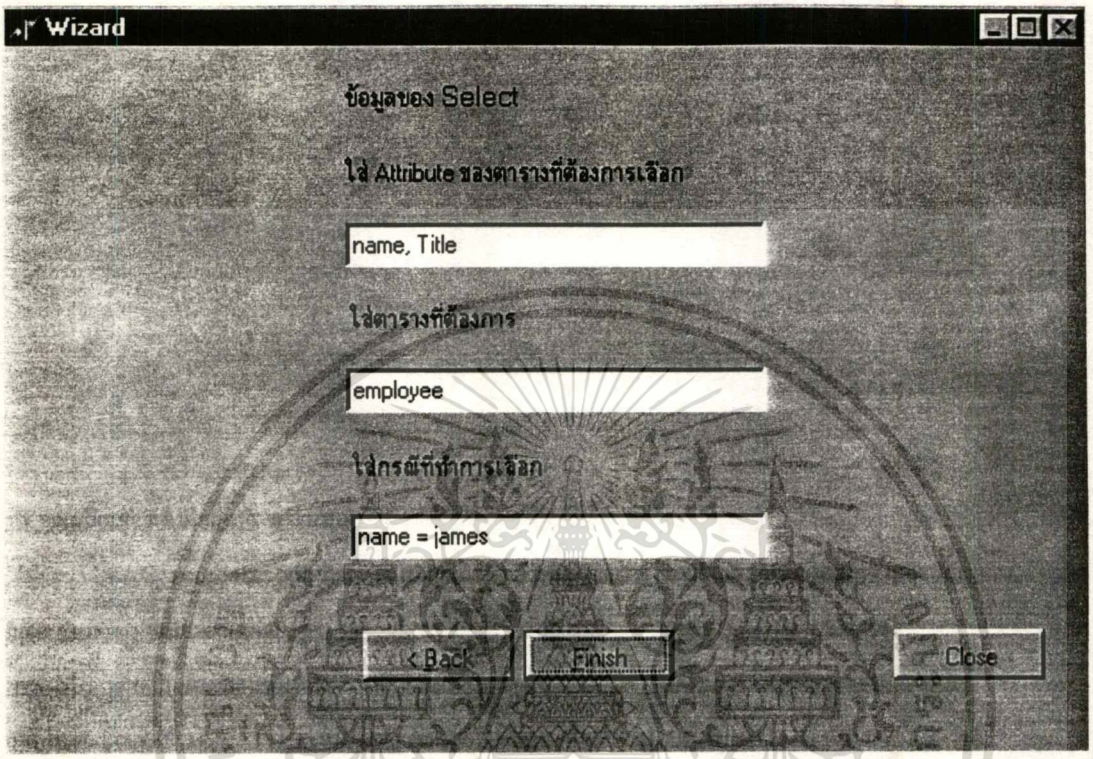


รูปที่ 4.13 หน้าจอป้อนข้อมูลการ delete

ในช่องแรกจะใส่ค่าของตารางที่ต้องการ delete

ช่องที่สองจะใส่ attribute ที่ต้องการ delete

การทำงานของการ insert



รูปที่ 4.14 หน้าจอป้อนข้อมูลการ insert

ในช่องแรกจะให้ใส่ attribute ของตารางที่จะทำการเลือก

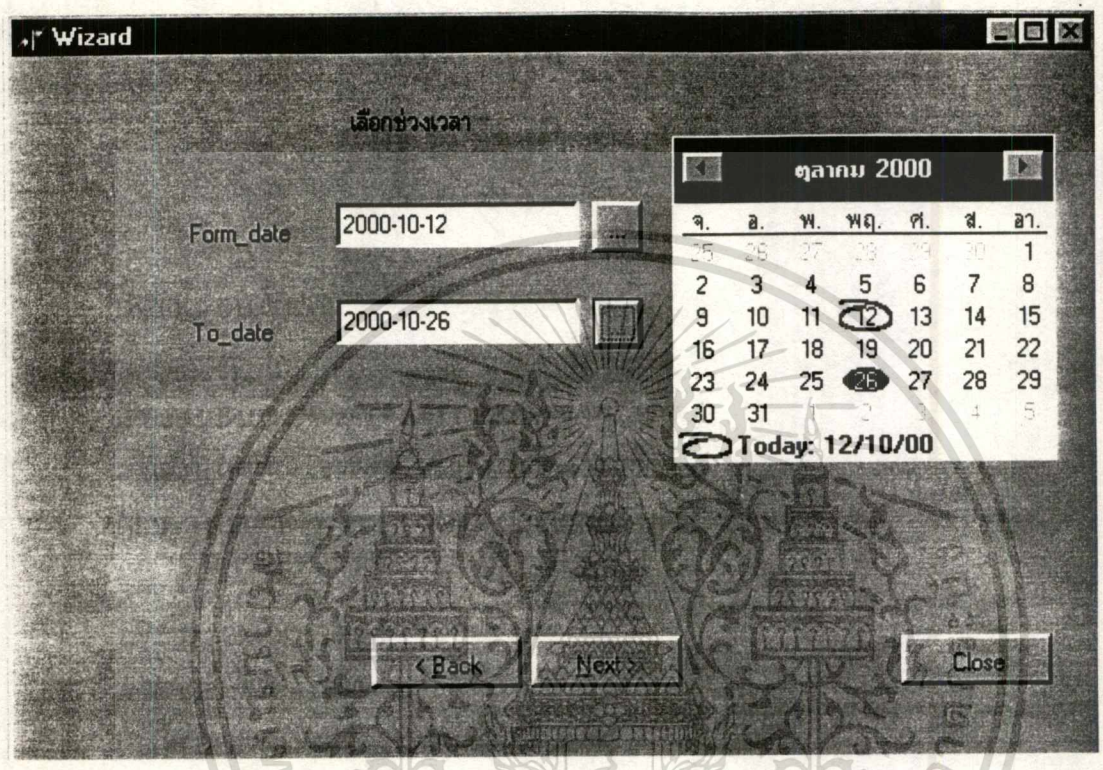
ในช่องที่สองจะให้ใสตารางที่ต้องการ ใส่ได้เพียงตารางเดียวโดยโปรแกรมจะทำการตรวจสอบเอง

ในช่องที่สามจะให้ใส่กรณีที่จะเลือก

การทำงานของ validtime แบบ sequence จะมีช่วงของเวลามาเกี่ยวข้องด้วย โดยจะมี form_date และ To_date นั่นคือจากจุดเริ่มต้นของเวลาจนถึงจุดสิ้นสุด

โดยในส่วนนี้จะมีการตรวจสอบว่าถ้า To_date น้อยกว่า Form_date จะเกิด error

และถัดมาก็จะทำงานในขั้นตอนการเลือกชนิดของ SQL นั่นคือ select update insert delete ที่จะมีการทำงานเหมือนกันรูปที่ 4.6

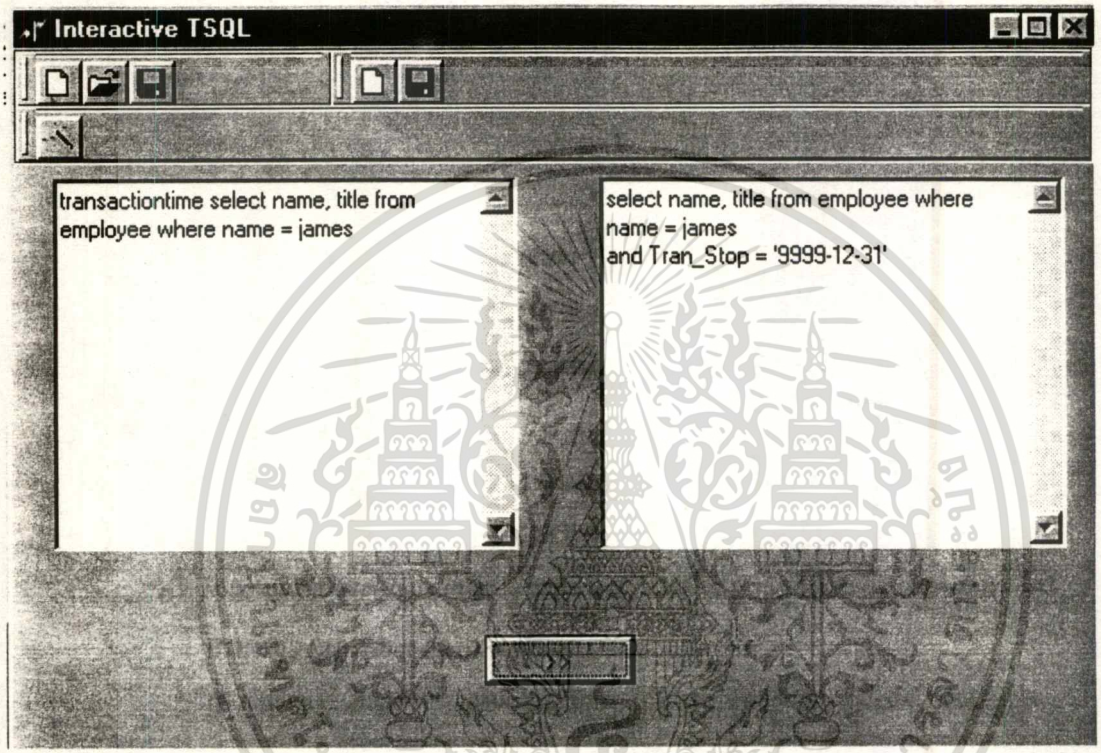


รูปที่ 4.15 แสดงการทำงานของ validtime แบบ sequence

หลังจากทำจนครบขั้นตอนแล้วผลที่ได้ก็จะเป็นดังนี้

รูปที่ 4.16 ผลของ validtime sequence

และในส่วนสุดท้ายคือการทำงานของ transactiontime โดยจะเหมือนกับรูปที่ ปป แล้วก็จะทำตามขั้นตอนการเลือกชนิดของ SQL จนเสร็จจะได้รูปของผลทำงานดังนี้



รูปที่ 4.17 ผลของการทำ transactiotime

บทที่ 5
สรุปผลการศึกษา

เราได้พยายามที่จะทำการทดลองในความคิดและเทคนิคสำหรับการ ศึกษาพัฒนา Interactive TSQL ของ Valid time และ Transaction time ด้วย SQL-92 จากกรรมวิธีที่แสดงในรายงานฉบับนี้ ซึ่งเป็นการพัฒนาให้เห็นมุมมองที่กว้างขึ้นของคุณสมบัติ ประโยชน์ ความจำเป็น และวิธีการทำงานของ temporal database

และทำให้เราได้ทราบถึงว่า temporal database จะมีกระบวนการทำงานที่สั้น และเข้าใจได้ง่ายต่อการนำไปใช้ แต่เมื่อมองกระบวนการทำงานที่เป็น physical ในลักษณะของ SQL-92 แล้วก็จะทำให้เราเข้าใจในกระบวนการทำงานของมันมากขึ้น และรู้ว่ากระบวนการทำงานจริง ๆ และมุมมองที่ถูกต้องต่อกระบวนการทำงานเป็นเช่นไร และได้เข้าใจถึงลักษณะการขยายคุณสมบัติของ SQL-92 เพื่อให้สนองต่องานในด้านต่าง ๆ ที่ต้องมี time-varying data

และในด้านนี้ ก็จะทำให้ผู้ที่สนใจ และเห็นประโยชน์ในการพัฒนา และนำ temporal database มาประยุกต์ใช้ในระบบงานจริง ๆ ได้มากขึ้น

บรรณานุกรม

Bennett J.P. 1960. **Introduction to compiling technques.** England : McGRAW-HILL.

David Gries, **Compiler Construction for digital computers.** Lewis P. M. II, Rosenkrantz D. J. and Stearns R. E. 1978. **Compiler Design Theory.** London : Addison-Wesley publishing company.

David Toman. **A point-based Temporal Extension of SQL.** Department of computer science, University of Toronto, Ontario M5S 1A4, Canada.

Kristian Torp, Christian S. Jensen and Michael Bohlen. 1997. **Layered Implementation of Temporal DBMSs – Concepts and Techniques.** A TIMECENTER Technical Report.

Richard T. Snodgrass. 1995. **Managing Temporal Data A Five-Part Series.** A TIMECENTER Technical Report.

Snodgrass, R.T. 1995. **An Evaluation of TSQL2.** TSQL2 Commentary. [Online]. Available : ftp://ftp.cs.arizona.edu/tsql/tsql2/eval.pdf

Tansel , A. J. et. al. 1993. **Temporal Databases:Theory, Design, and Implementation,** Database Systems and Applications Series. Redwood City, CA : Benjamin/Cummings Pub. Co.

Zaniolo, C. et. al. 1995 **Advanced Database Systems.** San Francisco, CA : Morgan Kaufmann Publishers, Inc.

ประวัติผู้เขียน

นาย ณัฐวุฒิ รัตนจรัสกุล จบการศึกษาระดับปริญญาตรี จาก ม. ธรรมศาสตร์ สาขาศาสตร์คอมพิวเตอร์ ปัจจุบันทำงานอยู่ฝ่ายพัฒนาระบบข้อมูล ที่ ธนาคารเอเชียจำกัดมหาชน