การพัฒนาโปรแกรมประยุกต์บนฐานข้อมูลเชิงเวลา

Temporal Database Application Development

จัดทำโดย

นายนิติศักดิ์ มูลตรีศรี

นางสาวกัญฐิพัศภ์ ภิรมย์ธรรม

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมประยุกต์บนฐานข้อมูลเชิงเวลา

Temporal Database Application Development

ผู้จัดทำ

1. นายนิติศักดิ์ มูลตรีศรี รหัสนักศึกษา 47015326

2. นางสาวกัญฐิพัศภ์ ภิรมย์ธรรม รหัสนักศึกษา 47015670

อาจารย์ที่ปรึกษา

(รศ.ดร.ศุภมิตร จิตตะยโศธร)

การพัฒนาโปรแกรมประยุกต์บนฐานข้อมูลเชิงเวลา

นายนิติศักดิ์ มูลตรีศรี	รหัสนักศึกษา 47015326
นางสาวกัญฐิพัศภ์ ภิรมย์ธรรม	รหัสนักศึกษา 47015670
รศ.ดร.ศุภมิตร จิตตะยโศธร	อาจารย์ที่ปรึกษา
ปีการศึกษา 2549	

บทคัดย่อ

ในโลกปัจจุบันข้อมูลนั้นถือว่าเป็นสิ่งสำคัญต่อการพัฒนาองค์กรต่างๆ จึงมีโปรแกรมประยุกต์จำนวนมากที่ต้องการเก็บข้อมูลอ้างอิงกับเวลา (ยกตัวอย่างเช่น ระบบสนับสนุนการตัดสินใจของผู้บริหาร, ระบบคลังข้อมูล ,ระบบ Spatio-Temporal Database เป็นต้น) โดยที่คำว่าข้อมูลอ้างอิงกับเวลานั้นหมายถึง ฐานข้อมูลนั้นสามารถเก็บข้อมูลได้ทั้งที่เป็น อดีต ปัจจุบัน และ อนาคต เช่น ข้อมูลกำหนดการ หรือข้อมูลที่มีการวางแผนไว้ล่วงหน้า

ปริญญานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อนำเสนอแนวคิดเกี่ยวกับฐานข้อมูลเชิงเวลา และได้นำเสนอขั้นตอนในการพัฒนาโปรแกรมประยุกต์บนฐานข้อมูลเชิงเวลา โดยได้ยกตัวอย่าง Spatio Temporal Database ซึ่งเป็นการนำคุณสมบัติของฐานข้อมูลเชิงเวลามาช่วยในการเก็บข้อมูลเชิงภูมิศาสตร์ โดยได้นำความสามารถในการสนับสนุนฐานข้อมูลเชิงเวลาของระบบจัดการฐานข้อมูล Oracle 10g มาช่วยในการจัดการฐานข้อมูลเชิงเวลา

Temporal Database Application Development

Nitisak Mooltreesri	47015326
Gantipaj Piromthum	47015670
ASSOC.PROF.DR. Suphamit Chittayasothorn	Advisor
Academic Year 2006	

ABSTRACT

Nowadays, data to have importance for organization development, most application to need record time-varying information (e.g., Decision Support System, Data Warehouse, Spatio-Temporal Database System). The term "time-varying" database is awkward, because even if only the current state is kept in the database (e.g., the current stock, the current salary and job title of employees), this database will change as reality changes, and so could perhaps be considered a time-varying database. The term "historical database" implies that the database only stores "historical" information, that is, information about the past; a temporal database may store information about the future, e.g., schedules or plans.

This thesis intend to present Temporal Database concept, step of Temporal Database application development. We are included Spatio-Temporal Database case study and develop application with Oracle 10g.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับคำแนะนำ และความรู้จากท่าน รศ.ดร.ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาทางคณะผู้จัดทำขอขอบพระคุณอย่างยิ่งสำหรับทุกสิ่งทุกอย่างที่ได้รับจากอาจารย์ นอกจากนี้ต้องขอขอบพระคุณอาจารย์ทุกท่านที่ได้สอนสั่งคณะผู้จัดทำจนมีความรู้ความสามารถจนถึงทุกวันนี้

ขอขอบคุณบริษัท Oracle ที่ได้สนับสนุน Software Oracle 10g Enterprise Edition ให้แก่คณะผู้จัดทำได้นำมาใช้ในโครงงาน

ขอขอบคุณเพื่อนๆ พี่ๆ รวมถึงน้องๆ ทุกคนที่ให้ความช่วยเหลือมาตลอด

สุดท้ายขอกราบขอบพระคุณ บิดา มารดา ที่เป็นผู้อุปการเลี้ยงดู และให้กำลังใจตลอดมา

คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ คณะผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

นายนิติศักดิ์ มูลตรีศรี รหัสนักศึกษา รหัสนักศึกษา 47015326

นางสาวกัญฐิพัศภ์ ภิรมย์ธรรม รหัสนักศึกษา 47015670

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในฐานข้อมูลที่แบบดังเดิม (non-temporal) จะเก็บเฉพาะข้อมูล ณ. เวลาใดเวลาหนึ่งเท่านั้น ซึ่งปกตินั้นจะเป็นข้อมูลล่าสุด การเปลี่ยนแปลงข้อมูลนั้นข้อมูลเก่าจะถูกเขียนทับโดยข้อมูลใหม่ การเรียกค้นข้อมูลว่าถูกต้องหรือไม่เป็นการเรียกค้นข้อมูลที่มีจริงขณะที่เรียกค้นเท่านั้นการเรียกค้นเมื่อต้องการประวัติของข้อมูลจะไม่สามารถทำได้ ยกตัวอย่างเช่น เมื่อปี 2545 นายเกริกพล ได้เข้าศึกษาในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง หลังจากนั้นเมื่อ 2546 นายเกริกพลได้เปลี่ยนชื่อเป็น นายปองพล ดังนั้นในปี 2548 หากพนักงานของสถาบันจะหาข้อมูลของนายเกริกพล ก็จะหาไม่เจอเนื่องจากข้อมูลนั้นถูกเปลี่ยนแปลงไปแล้ว ซึ้งถ้าหากเป็นฐานข้อมูลเชิงเวลาซึ้งจะมีการเก็บข้อมูลเดิมไว้ด้วยโดยอ้างอิงกับช่วงเวลา ทำให้สามารถสืบค้นข้อมูลในอดีตได้

ถึงแม้ฐานข้อมูลเชิงเวลาช่วยให้สามารถรู้ข้อมูลในอดีต ปัจจุบัน หรืออนาคต ได้ การทำเช่นนี้จะต้องอาศัยใช้ช่วงเวลาเพื่อนำมาประกอบกับข้อมูล จึงทำให้เกิดความยุ่งยากในการจัดการฐานข้อมูลเป็นอย่างมาก เช่น การป้องกันความซ้ำซ้อนของข้อมูล การนำข้อมูลมาแสดง เป็นต้น

ปริญญานิพนธ์ฉบับนี้ได้เสนอ ขั้นตอนต่างๆ เพื่อจะช่วยจัดการความยุ่งยากในการพัฒนาโปรแกรม โดยใช้ฐานข้อมูลเชิงเวลา โดยจะใช้ระบบจัดการฐานข้อมูลที่ช่วยสนับสนุนฐานข้อมูลเชิงเวลา คือ Oracle 10g รวมไปถึงการนำฐานข้อมูลเชิงเวลามาใช้ร่วมกับฐานข้อมูลเชิงพื้นที่

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อเสนอแนวทางในการจัดการฐานข้อมูลเชิงเวลา

1.2.2 เพื่อเสนอแนวทางในพัฒนาโปรแกรมประยุกต์ โดยใช้ฐานข้อมูลเชิงเวลา

1.2.3 เพื่อศึกษาการใช้ Oracle 10g ในการพัฒนาโปรแกรมประยุกต์ โดยใช้ฐานข้อมูลเชิงเวลา

1.2.4 เพื่อนำเสนอตัวอย่างการนำคุณสมบัติของฐานข้อมูลเชิงเวลามาใช้กับข้อมูลเชิงพื้นที่

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ ความเข้าใจเกี่ยวกับแนวทางในการจัดการฐานข้อมูลเชิงเวลา

1.3.2 สามารถพัฒนาโปรแกรมประยุกต์ โดยใช้ฐานข้อมูลเชิงเวลา

1.4 ส่วนประกอบของปริญญานิพนธ์

บทที่ 1 กล่าวถึงความเป็นมาของปัญหา วัตถุประสงค์ของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ ขอบเขตของโครงงาน และส่วนประกอบของรายงานฉบับนี้

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ซึ่งประกอบด้วยทฤษฎีของฐานข้อมูลเชิงเวลา ทฤษฎีของฐานข้อมูลเชิงภูมิศาสตร์ และSpatio Temporal Database

บทที่ 3 กล่าวถึงการใช้ Oracle 10g ช่วยจัดการฐานข้อมูลเชิงเวลา ,ฐานข้อมูลเชิงภูมิศาสตร์ บน Oracle Spatial และ Spatio Temporal Database บน Oracle 10g

บทที่ 4 การทดลองและผลการทดลอง ในการใช้ Oracle 10g ช่วยในการจัดการฐานข้อมูลเชิงเวลา และ Spatio Temporal Database

บทที่ 5 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน ปัญหาอุปสรรคและแนวทางแก้ไข และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

1.5 ขอบเขตของโครงงาน

ศึกษาทฤษฎีเกี่ยวกับฐานข้อมูลเชิงเวลา ทฤษฎีเกี่ยวกับฐานข้อมูลภูมิศาสตร์ และการนำคุณสมบัติของฐานข้อมูลทั้งสองชนิด เพื่อนำมาเป็นแนวทางในการพัฒนาโปรแกรมประยุกต์โดยใช้ฐานข้อมูลเชิงเวลาและ Spatio-Temporal Database รวมถึงการศึกษาระบบจัดการฐานข้อมูล Oracle 10g เพื่อช่วยในการจัดการฐานข้อมูลเชิงเวลา และ Spatio-Temporal Database

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ฐานข้อมูลเชิงเวลา

ฐานข้อมูลเชิงเวลาได้ถูกคิดค้นขึ้นเพื่อปรับปรุงระบบฐานข้อมูลแบบเดิมซึ่งฐานข้อมูลแบบเดิมไม่มีเวลามาเกี่ยวข้องจะเกิดปัญหาขึ้นคือ

1. หากมีการเปลี่ยนแปลงข้อมูลกับฐานข้อมูลแล้วข้อมูลใหม่จะเขียนทับข้อมูลเดิม
2. ไม่สามารถค้นหาประวัติของข้อมูลที่ต้องการได้

โดยนิยามของฐานข้อมูลเชิงเวลา คือ "ฐานข้อมูลที่อ้างอิงข้อมูลกับเวลาซึ่งเวลาที่เราสนใจนั้นไม่รวมถึงเวลาที่ผู้ใช้กำหนดขึ้น (User Define Time) "[1] โดยประเภทและความหมายของเวลาจะได้กล่าวในหัวข้อถัดไป

2.2 ประเภทและความหมายของเวลา

ในฐานข้อมูลเชิงเวลานั้นจะมีการระบุเวลาเข้ากับฐานข้อมูลซึ่งประเภทของเวลาที่ระบุในฐานข้อมูลมีอยู่ 3 ชนิด [2] คือ

2.2.1 Valid Time

ช่วงเวลาที่แสดงว่า Fact นั้นเป็นจริง หรือช่วงเวลาที่ข้อมูลนั้นเป็นจริงในฐานข้อมูล ยกตัวอย่างเช่น มีพนักงาน ก. เข้าทำงานเมื่อวันที่ 1 มกราคม 2543 มีเงินเดือนเดือนละ 15000 เมื่อวันที่ 1 มกราคม 2544 บริษัทได้ปรับเงินเดือนให้เป็นเดือนละ 18000 ต่อเดือน ดังนั้นช่วงเวลา Valid Time ของข้อมูลของพนักงาน ก. ที่มีเงินเดือน 15000 บาท คือ 1 มกราคม 2543 ถึง 1 มกราคม 2544 เป็นต้น

2.2.2 Transaction Time

เวลาที่ข้อมูลนั้นถูกบันทึกลงในฐานข้อมูล Transaction Time ส่วนใหญ่จะได้จาก DBMS และจะไม่สามารถแก้ไขข้อมูล Transaction Time ในอดีตได้เนื่องจากจะผิดกฎของเวลาที่ว่าเราไม่สามารถเปลี่ยนแปลงอดีตได้ [2] ยกตัวอย่างเช่น ข้อมูลเงินเดือนของพนักงาน ก. เดือนละ 15000 บาทต่อเดือนได้ ถูกบันทึกเมื่อวันที่ 1 มกราคม 2543 ได้มีการวางแผนว่าจะปรับเงินเดือนให้เป็น 18000 บาทต่อเดือน ในวันที่ 1 มกราคม 2544 แต่ฝ่ายการเงินได้ บันทึกข้อมูลเมื่อวันที่ 30 ธันวาคม 2543 ดังนั้น Transaction Time ของข้อมูลของพนักงาน ก. ที่มีเงินเดือน 15000 บาทต่อเดือน คือ 1 มกราคม 2543 ถึงวันที่ 30 ธันวาคม 2543 เป็นต้น เราสามารถนำ Transaction Time มาใช้เพื่อกำหนดช่วง

Transaction Time ของแต่ละข้อมูลได้ เช่น ช่วงเวลา ของ Transaction Time ของพนักงาน ก. ที่มีเงินเดือน 15000 บาทคือ วันที่ 1 มกราคม 2543 ถึง 30 ธันวาคม 2543

2.2.3 User Define Time

เป็นเวลาที่อยู่ในรูปข้อมูลที่เป็นส่วนหนึ่งของมูล คือ ระบบจะมองเป็นแค่ข้อมูลธรรมดา ถึงแม้จะมีประเภทข้อมูลเป็นเวลาก็ตาม ส่วนใหญ่จะเป็นข้อมูลที่มีค่าคงที่ ยกตัวอย่าง วันเกิด วันแต่งงาน วันเข้าทำงาน เป็นต้น

2.3 ความซ้ำซ้อนของข้อมูลในฐานข้อมูลเชิงเวลา

ปัญหาความซ้ำซ้อนของข้อมูลในฐานข้อมูลนั้นเป็นเรื่องที่ยุ่งยาก สำหรับ DBA และโปรแกรมเมอร์ที่จะต้องคอยจัดการให้ดี เนื่องจากหากจัดการไม่ดีแล้วจะทำให้เกิดปัญหาภายหลัง ยิ่งเป็นฐานข้อมูลเชิงเวลาแล้วยิ่งมีโอกาสเกิดความซ้ำซ้อนได้มากกว่าฐานข้อมูลแบบธรรมดา เนื่องจากข้อมูลในฐานข้อมูลเชิงเวลานั้นส่วนใหญ่จะเป็นการ INSERT และ UPDATE เป็นหลัก เพื่อให้ง่ายในการอธิบายจะของยกตัวอย่างเฉพาะ Valid Time ความซ้ำซ้อนใน Transaction Time ก็มีลักษณะเดียวกัน

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

รูปที่ 2.1 ตาราง NICUStatus เพื่อเก็บข้อมูลสถานะคนไข้

Name	Status
Alexis May	fair
Alexis May	fair
Alexis May	fair

รูปที่ 2.2 แสดงการเกิด Value-equivalent duplicat ในตาราง NICUStatus

ขอยกตัวอย่างฐานข้อมูลสถานนะของคนไข้ในโรงพยาบาลแห่งหนึ่งซึ่งจะบันทึกสถานะของคนไข้และเวลาที่ใช้อ้างอิง โดยจะยอมให้มีสถานณะซ้ำได้ในวันเดียวกันดังนั้นจะเกิดความซ้ำซ้อนในรูปแบบต่างๆ ดังนี้

2.4.1 Non-Sequenced duplicate คือ ความซ้ำซ้อนที่มีแถวใดๆ มีค่าในทุกๆ ค่า เท่ากันกับแถวอื่นในตาราง จะเห็นได้จากสองแถวด้านล่างสุด ของตารางสถานะคนไข้ ในรูปที่ 2.1

2.4.2 Value-equivalent duplicat คือ ความซ้ำซ้อนของข้อมูลที่มีแถวใดๆ ที่มีข้อมูลที่ไม่ใช่ข้อมูลเชิงเวลาเท่ากันกับแถวอื่นในตาราง ดังตัวอย่างที่แสดงในรูปที่ 2.2 เป็นต้น

2.4.3 Current duplicate คือ ความซ้ำซ้อนที่ ณ. เวลาปัจจุบันมีข้อมูลในแถวใดๆ ที่มีความซ้ำซ้อนเกิดขึ้นในตาราง ยกตัวอย่างเช่น ถ้าวันนี้เป็นวันที่ January 6, 1998 3 rows สุดท้ายนั้นจะเป็น Current duplicate แต่ถ้าวันนี้เป็นวันก่อนวันที่ December 02, 1998 และหลังวันที่ January 11, 1998 จะไม่ถือว่าเป็น Current duplicate

2.4.4 Sequenced duplicate คือ ความซ้ำซ้อนที่เกิด ณ. เวลาใดๆ (ไม่สนใจว่าจะเป็น อดีต ปัจจุบัน หรือ อนาคต) ยกตัวอย่างเช่น 3 Rows สุดท้ายก็ถือว่าเป็น Sequenced duplicate

ตารางที่ 2.1 แสดงความสัมพันธ์ระหว่างประเภทของความซ้ำซ้อนชนิดต่างๆ ในแนวตั้งคือประเภทของความซ้ำซ้อนที่จะเกิดขึ้นเมื่อความซ้ำซ้อนตามแนวนอนเกิดขึ้น

	Sequenced	Current	Value-equivalent	Nonsequenced
Sequenced	X		X	
Current	X	X	X	
Value-equivalent			X	
Nonsequenced	X		X	X

2.4 ประเภทของฐานข้อมูลเชิงเวลา

ฐานข้อมูลเชิงเวลา [1] (Temporal Database) คือ คือฐานข้อมูลที่สนับสนุนเวลาแบบต่างๆ แต่ไม่รวมถึง User-Define Time ฐานข้อมูล Valid Time และ Transaction Time ทำให้เกิดฐานข้อมูลเชิงเวลาซึ่งเรียกตามชนิดการเก็บข้อมูลดังนี้คือ

2.5.1 Historical Database

เป็นฐานข้อมูลที่มีการเก็บข้อมูล โดยอ้างอิงกับ Valid Time ซึ่งสามารถเปลี่ยนแปลงข้อมูลได้ และในมุมมองของฐานข้อมูลเชิงเวลายังมองการทำแก้ไขข้อมูลในฐานข้อมูลชนิดนี้ได้เป็น 2 ลักษณะคือ

2.3.1.1 ลักษณะทั่วไป คือ ทั่วไป คือการทำการแก้ไขข้อมูลในอดีต ปัจจุบัน หรืออนาคตก็ได้

2.3.1.2 ลักษณะที่มีข้อจำกัด คือ จะยอมให้มีการแก้ไขข้อมูลแบบปัจจุบันได้เท่านั้น

2.5.2 Rollback Database

จะเก็บเฉพาะ Transaction Time ในการเก็บจะไม่เก็บประวัติของข้อมูลโดยตรงแต่จะเก็บบัน ทึกเฉพาะเวลาของการเปลี่ยนแปลงข้อมูล Tuple จะไม่เปลี่ยนสถานะแต่จะมีข้อมูลใหม่ซ้ำสถานะจะไม่ สามารถ เปลี่ยนแปลงข้อมูลได้แต่จะเพิ่มข้อมูลได้ หรือจะเป็นการเก็บพฤติกรรมข้อมูล

2.5.3 Bitemporal Database

จะทำการเก็บทั้ง Transaction Time และ Valid Time ซึ่งจะทำให้ได้ทั้งพฤติกรรมข้อมูลและประวัติของข้อมูลซึ่งจะเป็นฐานข้อมูลที่ชัดเจนที่สุดในฐานข้อมูลเชิงเวลา

2.5 กฎบังคับความถูกต้องในฐานข้อมูลเชิงสัมพันธ์ในรูปแบบฐานข้อมูลเชิงเวลา

กฎบังคับความถูกต้องในฐานข้อมูลเชิงสัมพันธ์ในรูปแบบฐานข้อมูลเชิงเวลา คือ กฎที่ใช้บังคับความถูกต้องของข้อมูล โดยจะอาศัยส่วนประกอบสองอย่าง คือ ข้อมูล และเวลา มีอยู่สองประเภท คือ Temporal Existential Integrity “ attributes ที่ใช้เป็นคีย์หลักของข้อมูลจะต้องไม่มีค่าเป็น Null ในทุกช่วงเวลาของข้อมูล ”[2] และ Temporal Referential Integrity Constraints

2.3.2 Temporal Existential Integrity

“ Attributes ที่ใช้เป็นคีย์หลักของข้อมูลจะต้องไม่มีค่าเป็น Null ในทุกช่วงเวลาของข้อมูล ”[2] ขยายความ คือ Existential Integrity ในฐานข้อมูลเชิงสัมพันธ์แบบธรรมดา (non-temporal) จะเป็นกฎที่บังคับความสมบูรณ์ของข้อมูลในด้านเดียวคือในตารางเดียวกันห้ามมีคีย์หลักซ้ำกัน หรือมีค่าเป็น Null แต่ในฐานข้อมูลเชิงเวลากฎนี้จะนำข้อมูลเชิงเวลามาอ้างอิงด้วย ยกตัวอย่างดังแสดงในตาราง โดยกำหนดให้ E# เป็นคีย์หลัก ซึ้งแม้จะมีคีย์หลักซ้ำกันในสองแถวแรกแต่อยู่คนละช่วงเวลา จึงสามารถบันทึกข้อมูลได้โดยไม่ผิดกฎบังคับความสมบูรณ์ของข้อมูล เป็นต้น

ตารางที่ 2.2 แสดงความตาราง เก็บข้อมูลของพนักงาน

E#	Start	End
121	10	12
121	14	*Now*
133	23	30
147	18	*Now*

2.6.1 Temporal Referential Integrity

คือ " กฎที่ใช้บังคับสำหรับการอ้างอิงข้อมูล โดยข้อมูลที่อ้างอิงนั้น ต้องสัมพันธ์กับข้อมูลของคีย์หลัก และต้องอยู่ในช่วงเวลาของคีย์หลักที่ใช้อ้างอิงด้วย " ซึ่งต้องอาศัยส่วนประกอบสองส่วน คือ ข้อมูล และ เวลา เพื่อใช้อ้างอิงข้อมูล จะแบบออกได้เป็น 2 กรณีคือ

2.6.1.1 กรณีที่ทั้งตารางที่เป็นคีย์หลักเป็นตารางที่สนับสนุน Valid Time แล้ว ตารางที่อ้างถึงข้อมูลต้องเป็นตารางที่สนับสนุน Valid Time แล้วเท่านั้น และข้อมูลที่อ้างถึงนั้นต้องอยู่ในช่วงเวลาของคีย์หลักเท่านั้น

2.6.I.2 กรณีที่ตารางที่เป็นคีย์หลักเป็นตารางที่ไม่ได้สนับสนุน Valid Time แล้ว จะไม่สนใจเวลาสามารถอ้างถึงข้อมูลนั้นได้เลย

2.6 คุณสมบัติที่จำเป็นในการที่ทำให้ฐานข้อมูลแบบธรรมดาสามารถสนับสนุนฐานข้อมูลเชิงเวลา

การที่จะทำให้ฐานข้อมูลแบบธรรมดาให้สามารถสนันสนุนฐานข้อมูลเชิงเวลาต้องมีปัจจัยประกอบ 3 ประการ ดังนี้

2.6.1 โครงสร้างข้อมูล จะต้องสามารถเก็บข้อมูลเกี่ยวกับเวลาได้

การเพิ่ม Attribute Time Stemp และ Tuple Time Stemping ที่เกี่ยวกับเวลาเข้าไปในโครงสร้างข้อมูลเป็นเรื่องที่ง่ายมาก เพียงเราเพิ่ม Colum ที่เกี่ยวกับ Valid Time หรือ Transaction Time เข้าไปใน Database Schema แล้วให้ชนิดของข้อมูลลเป็นเวลาหรือวัน เช่น VALIDTIME_START และ VALIDTIME_END เป็นต้น สำหรับฐานข้อมูลแบบสัมพันธ์ หรือสำหรับฐานข้อมูลแบบวัตถุเพียงแค่เราเพิ่ม Attribute ที่ใช้เก็บ Valid Time หรือ Transaction Time ให้กับคลาส Date เท่านั้น

แต่สำหรับการเพิ่มโอเปอเรชั่นเกี่ยวกับฐานข้อมูลเชิงเวลานั้นจะซับซ้อนกว่าการเพิ่ม Attribute เกี่ยวกับเวลาในฐานข้อมูล เช่น ผลต่างของสองช่วงเวลาอาจได้ผลออกมาเป็นช่วงเวลาหลายช่วง (Set of interval) ซึ่งโดยทั่วไปจะใช้ Temporal element ซึ่งเป็น มาช่วยเรื่อง Time Stemp สำหรับฐานข้อมูลเชิงสัมพันธ์ นั้นส่วนที่ซับซ้อนที่สุดคือ Time Stemp ส่วนของ data structure เช่น teple หรือ Atttribute Timestemp ส่วน Object Data Model ทำได้ 3 แนวทาง คือ

1. Time Stemp ที่ Attribute หรือ Time Stemp ในระดับ ออบเจ็ค
2. Time Stempในระดับ Type
3. Time Stemp ในระดับ Identifier

2.6.2 โอเปอร์เรชั่นต่างๆ เพื่อการค้นหาและแก้ไขข้อมูล

การเพิ่มโอเปอร์เรชั่นเกี่ยวกับ Temporal เป็นเรื่องที่มีปัญหามากจาก Temporal algebra ที่ได้มีการนิยามไว้แล้ว ซึ่งเราสามารถทำได้โดยตรงกับระบบฐานข้อมูลเชิงเวลาหรือทำเป็นเลเยอร์อยู่บน non-temporal database system ซึ่งจำเป็นต้องมีส่วนขยายของภาษาในการสืบค้นข้อมูลให้สามารถใช้ โอเปอร์เรชั่นเกี่ยวกับ Temporal Operations ร่วมกับส่วนที่เป็น non-temporal อื่นๆ ได้

2.6.3 กฎบังคับความถูกต้องของข้อมูล

ใน Temporal Relational Database จะมีกฎที่ใช้ในการบังคับความถูกต้องของข้อมูล เช่น Temporal Existential Integrity และ Temporal Referential Integrity เป็นต้นส่วน DBMS ที่เป็นแบบ non-temporal database management system จะให้โปรแกรมเมอร์เป็นผู้จัดการ Temporal Integrity Constrain ที่เพิ่มขึ้นมาเอง ซึ่งรายละเอียดได้กล่าวในหัวข้อที่ 2.4

2.7 การ Queryและนำข้อมูลจากฐานข้อมูลที่สนับสนุน Valid-Time

ในหัวข้อนี้เราขอยกตัวอย่างตารางซึ่งเก็บข้อมูล ปศุสัตว์ในอเมริกาเพื่อใช้ในการตรวจสอบหาต้นเหตุของโรคระบาทที่เกิดขึ้น

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

รูปที่ 2.3 ตาราง LOT_LOC

ตารางที่ 2.3 แสดงรายละเอียดของตาราง LOT_LOC มีดังนี้

ชื่อ Column ความหมาย

FDYD_ID	หมายเลขของฟาร์ม
LOT_ID_NUM	หมายเลขของแต่ละฝูง
PE_N_ID	หมายเลขคอก
HD_CNT	จำนวนของวัวในแต่ละคอก
FROM_DATE	เวลาเริ่มต้นของ Valid Time ของแต่ละ Row
TO_DATE	เวลาสิ้นสุดของ Valid Time ของแต่ละ Row

การ Queries ซึ่งเป็นการนำข้อมูลในฐานข้อมูลออกมาแสดงและนำมาใช้ประโยชน์ ในฐานข้อมูลเชิงเวลาจะมีการลักษณะของการ Quereis ได้แก่ current, sequenced, non-sequenced ในหัวข้อนี้จะแสดงถึงความยุ่งยากในการใช้ภาษา SQL ธรรมดา Queries ข้อมูลในฐานข้อมูลเชิงเวลา ก่อนอื่นเราจะพิจารณาการ Queries ฐานข้อมูลแบบธรรมดา โดยจะพิจารณาคำถามว่า "ฝูงที่ 219 ในฟาร์มที่ 1 แต่ละคอกมีวัวอยู่เท่าไหร่" จะมีคำสั่งดังนี้

```
SELECT PEN ID, HD CNT
FROM LOT LOC                                    ( 3.1 )
WHERE FDYD ID = 1 AND LOT ID NUM = 219
```

2.7.1 Current Queries ในฐานข้อมูลเชิงเวลาหากต้องการถามคำถามแบบอ้างอิงปัจจุบัน (Current Queries) จะมีลักษณะการใช้คำสั่งที่ต้องระบุเวลาสิ้นสุดของข้อมูลที่ต้องการให้มากกว่าเวลาปัจจุบัน แต่ถ้ากำหนดให้ตารางไม่สามารถกำหนดช่วงเวลาสิ้นสุดล่วงหน้าก็สามารถกำหนดให้เป็นค่าที่มากที่สุดของช่วงเวลาได้ โดยมีรูปแบบคำสั่งในการสอบถามว่า "วัวฝูงที่ 219 ในฟาร์มที่ 1 ปัจจุบันในแต่ละคอกมีวัวอยู่เท่าไหร่"

```
SELECT PEN_ID, HD_CNT
FROM LOT LOC
WHERE FDYD ID = 1 AND LOT ID NUM = 219                ( 2.2 )
   AND TO_DATE = DATE '9999-12-31'
```

ผลลัพธ์ของคำสั่งได้ดังนี้

PEN_ID	HD_CNT
2	43

รูปที่ 2.4 ผลลัพธ์ของคำสั่ง 2.2

2.8.2 Sequenced Queries คำถามแบบนี้จะแสดงทุกช่วงเวลาของข้อมูลที่เราต้องการ ยกตัวอย่างเช่น " ในอดีต Lot 219 ฟาร์มที่ 1 แต่ละคอกมีวัวอยู่เท่าไหร่" โดยในการ Queries ต้องแสดงเวลาด้วย ตัวอย่าง ภาษา SQL

```
SELECT PEN ID, HD CNT, FROM_DATE, TO_DATE
FROM LOT_LOC                                    ( 2.3)
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
```

จะได้ผลลัพธ์ดังนี้

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

รูปที่ 2.4 ผลลัพธ์ของคำสั่ง 2.3

2.8.3 Non-Sequenced Queries คำถามแบบนี้จะอยากรู้ข้อมูลมากกว่าเวลา ยกตัวอย่าง เช่น "Lot 219 ฟาร์มที่ 1 แต่ละคอกมีวัวอยู่เท่าไหร่" โดยในการ Queries ต้องแสดงเวลาด้วย ตัวอย่าง ภาษา SQL

```
SELECT PEN ID, HD CNT
FROM LOT LOC                                    ( 2.4)
WHERE FDYD ID = 1 AND LOT ID NUM = 219
```

PEN_ID	HD_CNT
1	43
1	20
2	23
2	43

รูปที่ 2.5 ผลลัพธ์ของคำสั่ง 2.4

2.8 Temporal Joins

ในบางคำถามการใช้เพียงตารางเดียวอาจจะไม่เพียงพอที่จะตอบคำถามนั้นได้ เทคนิคอย่างหนึ่งที่นิยมนำมาช่วยใช้ในการตอบคำถามต่างๆ คือการ Jion ตาราง ยกตัวอย่างทำถามที่ต้องการทราบว่า "วัวฝูงไหนที่เคยอยู่ในคอกเดียวกันมาก่อนบ้าง" นั้นเราต้องใช้ Selt Join เข้ามาช่วย
ตัวอย่าง คำสั่ง SQL มีดังนี้

```
SELECT L1.LOT_ID NUM, L2.LOT_ID_NUM, L1.PEN_ID
FROM LOT LOC SNAPSHOT AS L1, LOT_LOC SNAPSHOT AS L2          ( 2.5)
WHERE L1.LOT ID NUM< L2.LOT_ID_NUM
      AND L1.FDYD ID = L2.FDYD_ID
      AND L1.PEN ID = L2.PEN_ID
```

2.8.1 Current Queries ในกรณีที่ต้องการอ้างอิงถึงข้อมูลที่มีช่วงเวลาอยู่ในปัจจุบัน ตัวอย่างคำถามได้แก่ "ปัจจุบันมีวัวฝูงไหนที่อยู่ในคอกเดียวกันบ้าง" โดยจะอ้างอิง From_date ตั้งแต่เวลาปัจจุบันเป็นต้นไป ตัวอย่างคำสั่ง SQL ดังนี้

```
SELECT L1.LOT ID NUM, L2.LOT ID NUM, L1.PEN ID
FROM LOT LOC AS L1, LOT LOC AS L2
WHERE L1.LOT ID NUM<L2.LOT ID NUM                            ( 2.6)
      AND L1.FDYD ID = L2.FDYD ID
      AND L1.PEN ID = L2.PEN ID
      AND L1.TO DATE = DATE '9999-12-31'
      AND L2.TO DATE = DATE '9999-12-31'
```

ซึ่งกรณีนี้ค่าที่มีค่าปัจจุบันจะมีการกำหนด TO_DATE วันที่ '9999-12-31' ให้เป็นผลลัพธ์ของชุดคำสั่งนี้ คือไม่มีข้อมูลกับมาเนื่องจากปัจจุบันไม่มีฝูงไหนที่อยู่ในคอกเดียวกันเลย

2.8.2 non-Sequenced Queries ได้แก่ "มีวัวฝูงไหนที่อยู่ในคอกเดียวกันบ้าง ซึ่งอาจจะเป็นคนละช่วงเวลาก็ได้ " โดยคำถามในลักษณะนี้จะไม่สนใจ Column ที่เป็น Timestamp
ตัวอย่างคำสั่ง SQL การตอบคำถามแบบ non-Sequenced

```
SELECT L1.LOT ID NUM, L2.LOT ID NUM, L1.PEN ID
```

```
FROM LOT LOC AS L1, LOT LOC AS L2
WHERE L1.LOT ID NUM < L2.LOT ID NUM                    ( 2.7)
      AND L1.FDYD ID = L2.FDYD ID
      AND L1.PEN ID = L2.PEN ID
```

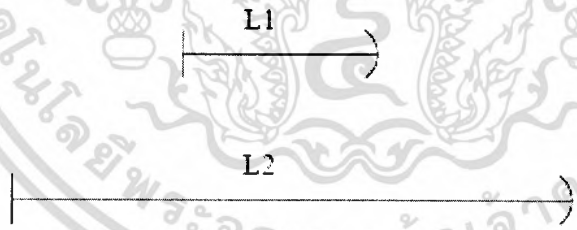
ผลลัพธ์ของคำสั่งข้างบนจะแสดงให้เห็นว่า มีวัวที่อาศัยอยู่ในคอกที่เดียวกันคือคอกที่ 1 มีอยู่ 3 ฝูง คือ 137 , 219 และ 374

L1	L2	PEN_ID
137	219	1
137	219	1
137	374	1
219	374	1
219	374	1

รูปที่ 2.6 ผลลัพธ์ของคำสั่ง 2.7

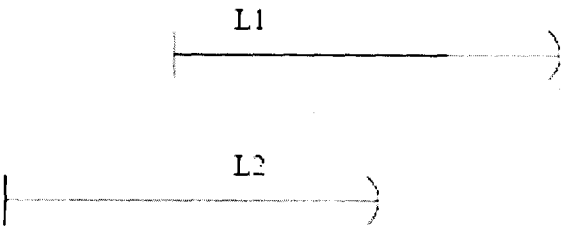
2.8.3 Sequenced Queries ลักษณะของคำถามเช่น "ในอดีตถึงปัจจุบันมีวัวฝูงไหนบ้างที่เคยอาศัยอยู่ในคอกเดียวกัน" การจะบอกคำถามในลักษณะนี้นั้นมีความยุ่งยากอยู่ที่เราต้องตรวจสอบทุกกรณีที่เป็นไปได้ที่จะทำให้ได้คำตอบโดยจะแบ่งเป็น 4 กรณี โดยให้สัญลักษณ์ L1 แทนช่วงเวลาที่ 1 และสัญลักษณ์ L2 แทนช่วงเวลาที่ 2

กรณีที่ 1 ช่วงเวลา L1 เป็นส่วนหนึ่งของ L2 (เกิดหลังและจบก่อน)



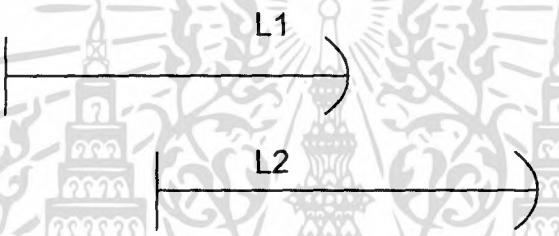
รูปที่ 2.7 Sequenced Queries กรณีที่ 1

กรณีที่ 2 ช่วงเวลาที่ L1 และ L2 มีการเหลื่อมล้ำกันโดย L1 เกิดหลัง L2 และจบหลัง L2 แต่จุดเริ่มต้นของ L1 เกิดก่อนที่ช่วงเวลา L2 จะสิ้นสุด



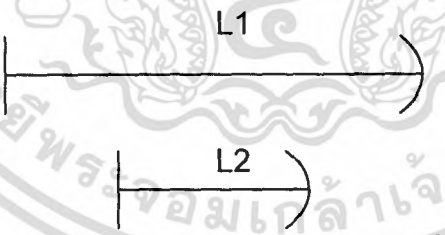
รูปที่ 2.8 Sequenced Queries กรณีที่ 2

กรณีที่ 3 ช่วงเวลาที่ L1 และ L2 มีการเหลื่อมล้ำกันโดย L2 เกิดหลัง L1 และจบหลัง L1 แต่จุดเริ่มต้นของ L2 เกิดก่อนที่ช่วงเวลา L1 จะสิ้นสุด



รูปที่ 2.9 Sequenced Queries กรณีที่ 3

กรณีที่ 4 ช่วงเวลา L2 เป็นส่วนหนึ่งของ L1 (เกิดหลังและจบก่อน)



รูปที่ 2.10 Sequenced Queries กรณีที่ 4

โดยคำสั่งนำทุกกรณีมา Union กัน ดังแสดงในตัวอย่าง

```
SELECT L1.LOT ID NUM, L2.LOT ID NUM, L1.PEN ID, L1.FROM DATE, L1.TO DATE
FROM LOT LOC AS L1, LOT LOC AS L2
```

```
WHERE L1.LOT ID NUM<L2.LOT ID NUM
                                    ( 2.8)
        AND L1.FDYD ID = L2.FDYD ID
        AND L1.PEN ID = L2.PEN ID
        AND L2.FROM DATE<= L1.FROM DATE
        AND L1.TO DATE<<= L2.TO DATE

UNION

SELECT  L1.LOT_ID_NUM,  L2.LOT_ID_NUM,  L1.PEN_ID,  L1.FROM_DATE,
L2.TO_DATE
FROM LOT_LOC AS L1, LOT_LOC AS L2
WHERE L1.LOT ID NUM< L2.LOT_ID_NUM
        AND L1.FDYD_ID = L2.FDYD_ID
        AND L1.PEN_ID = L2.PEN_ID
        AND L1.FROM_DATE>> L2.FROM_DATE
        AND L2.TO_DATE<L1.TO_DATE
        AND L1.FROM_DATE< L2.TO_DATE

UNION

SELECT L1.LOT ID NUM, L2.LOT ID NUM, L1.PEN ID, L2.FROM DATE, L1.TO DATE
FROM LOT LOC AS L1, LOT LOC AS L2
WHERE L1.LOT ID NUM<L2.LOT ID NUM
        AND L1.FDYD ID = L2.FDYD ID
        AND L1.PEN ID = L2.PEN ID
        AND L2.FROM DATE>> L1.FROM DATE
        AND L1.TO DATE<L2.TO DATE
        AND L2.FROM DATE< L1.TO DATE

UNION
```

```
SELECT L1.LOT ID NUM, L2.LOT ID NUM, L1.PEN ID, L2.FROM DATE, L2.TO DATE
FROM LOT LOC AS L1, LOT LOC AS L2
WHERE L1.LOT ID NUM< L2.LOT ID NUM
        AND L1.FDYD ID = L2.FDYD ID
        AND L1.PEN ID = L2.PEN ID
        AND L2.FROM DATE>= L1.FROM DATE
        AND L2.TO DATE<= L1.TO DATE
```

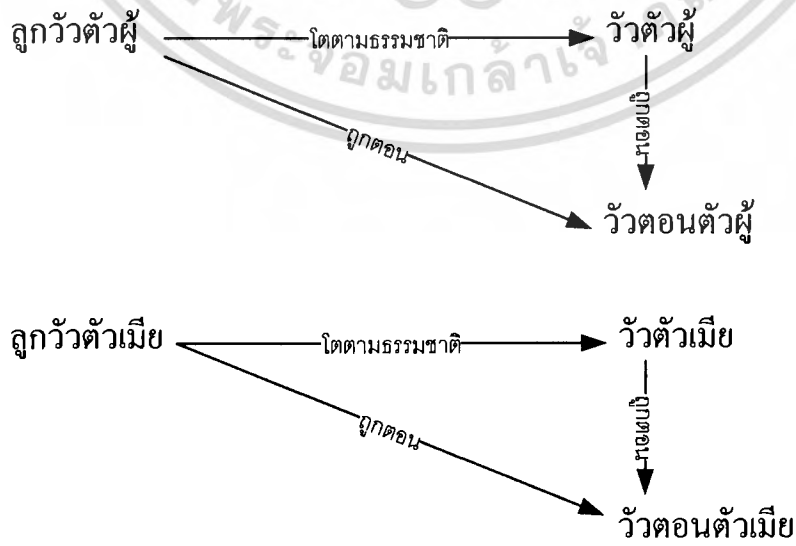
ผลลัพธ์ของคำสั่งข้างบนจะได้ข้อมูล 2 Rows

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	FROM_DATE	TO_DATE
219	374	1	1998-02-25	1998-03-01
219	374	1	1998-03-01	1998-03-14

รูปที่ 2.11 ผลลัพธ์ของคำสั่ง 2.8

2.9 การแก้ไขข้อมูล Valid-Time State Tables

ในหัวข้อนี้จะได้กล่าวถึงลักษณะการแก้ไขข้อมูล ในตารางที่สนับสนุน Valid Time จะยกตัวตารางการเปลี่ยนแปลงเพศของปศุสัตว์ เพื่อใช้ประกอบคำอธิบายโดยมีรายละเอียดดังนี้



รูปที่ 2.12 แผนภูมิแสดงการเปลี่ยนแปลงเพศของปศุสัตว์

ลูกวัว เมื่อถูกเลี้ยงปล่อยให้โตตามธรรมชาติ จะยังคงสถานะเป็นเพศเดิม แต่ถ้าถูกตอนจะกลายเป็นวัวตอน และจะไม่สามารถกลับมาเป็นสถานะเดิมได้อีก

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

รูปที่ 2.13 ตาราง LOT เพื่อเก็บความเปลี่ยนแปลงเพศของปศุสัตว์

ชื่อ Column	**ความหมาย**
LOT_ID_NUM	หมายเลขของฝูง
GNDR_CODE	สัญลักษณ์ของการแสดงเพศ
FROM_DATE	เวลาเริ่มต้นของ Valid Time ของแต่ละ Row
TO_DATE	เวลาสิ้นสุดของ Valid Time ของแต่ละ Row

โดยความหมายสัญลักษณ์ของการแสดงเพศ

c แทนด้วย วัวตัวผู้

h แทนด้วย วัวตัวเมีย

s แทนด้วย วัวที่ถูกตอนแล้ว

2.9.1 Current Modifications คือ การแก้ไขข้อมูลโดยอ้างอิงเวลาปัจจุบัน มีดังนี้

การ INSERT จะต้องใส่ช่วง Valid Time ตั้งแต่ ปัจจุบัน ถึงวันสิ้นสุด และอาจจะต้องมี Trigger เพื่อตรวจสอบความซ้ำซ้อนแบบต่างๆ

ยกตัวอย่างเช่น ปัจจุบันต้องการบันทึก ข้อมูลของฝูงที่ 433 เป็นวัวตัวเมีย มีคำสั่ง SQL ดังนี้

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

การ DELETE ในฐานข้อมูลเชิงเวลา จะเป็น Logical DELETE ก่อนอื่นต้อง Update TO_DATE ข้อมูลที่ต้องการลบด้วยเวลาปัจจุบัน

ในแง่มุมของฐานข้อมูลเชิงเวลานั้นมี อยู่ 2 เหตุการณ์ คือ

1. เหตุการณ์ทั่วไป คือการทำการแก้ไขข้อมูลในอดีต ปัจจุบัน หรือ อนาคตก็ได้ เป็นไปได้ทั้ง FROM_DATE หรือ TO_DATE โดย TO_DATE ต้องกำหนดขอบเขตที่ชัดเจน (ไม่ใช่ เครื่องหมาย Infinity)
2. เหตุการณ์ที่มีข้อจำกัด คือจะยอมให้มีการแก้ไขข้อมูลแบบปัจจุบันได้เท่านั้น คือ ถ้าข้อมูลยังเป็นจริงอยู่ให้กำหนด TO_DATE ให้เป็น Infinity

ตัวอย่าง Current Delete แบบทั่วไป คือวันนี้มีการย้ายฝูงที่ 234 ออก โดยมีคำสั่ง SQL ดังนี้

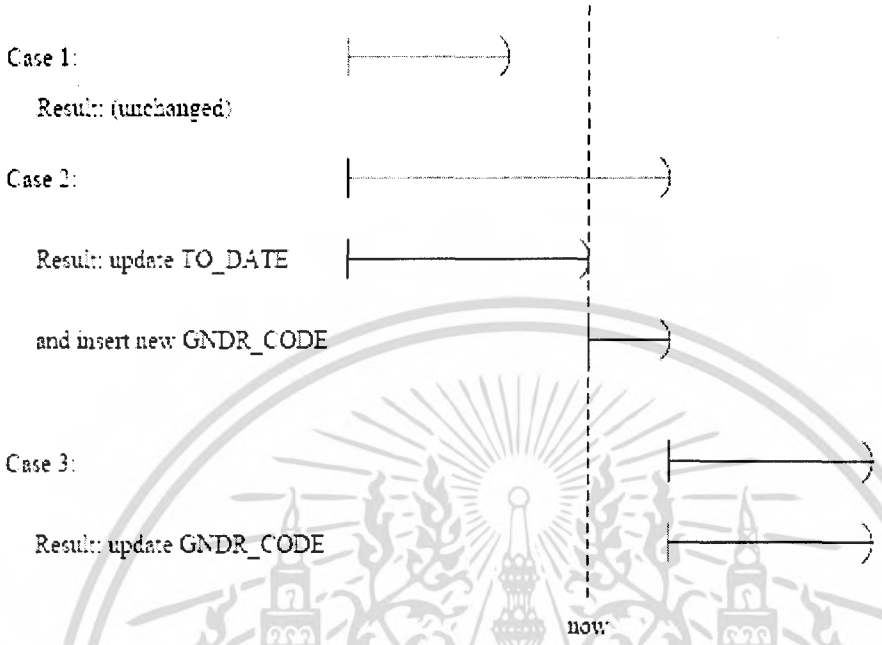
```
UPDATE LOT
SET TO DATE = CURRENT DATE
WHERE LOT ID NUM = 234
        AND TO DATE >= CURRENT DATE
        AND FROM DATE < CURRENT DATE

DELETE FROM LOT
WHERE LOT ID NUM = 234
        AND FROM DATE  <  CURRENT DATE
```

ก่อนอื่นต้อง Update ข้อมูลของฝูงที่ 234 ที่มี Valid Time อยู่ในช่วงปัจจุบันที่โดยให้ TO_DATE เป็นเวลาปัจจุบัน และลบข้อมูลของฝูงที่ 234 ที่มีการวางแผนไว้ในอนาคตออกด้วย Current Update ในการ Update มีอยู่ 3 กรณีที่เราจะต้องพิจารณาคือ

1. ข้อมูลที่ต้องการแก้ไข Valid ในอดีต ในกรณีนี้เราไม่ต้องแกไขข้อมูลใดๆ

2. กรณีที่ช่วงเวลา Valid Time อยู่ในปัจจุบัน เราต้องกำหนด TO_DATE ของข้อมูลเดิมให้เป็นเวลาปัจจุบัน และ INSERT ข้อมูลใหม่ลงไปโดยให้ FROM_DATE เป็นเวลาปัจจุบัน และ TO_DATE เป็นเวลาเดียวกันกับข้อมูลเดิม

3. กรณีที่ช่วงเวลา Valid Time อยู่ในอนาคตต้อง Update ทั้งช่วงเวลาของข้อมูลเดิม



รูปที่ 2.14 แสดงเส้นเวลาเพื่อพิจารณา Current Update

โดยมีตัวอย่างคำสั่ง SQL ดังนี้

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

```
UPDATE LOT
SET TO DATE = CURRENT DATE
WHERE LOT ID NUM = 799
AND GNDR_CODE<>'s'
AND FROM DATE<CURRENT_DATE
AND TO DATE> CURRENT_DATE

UPDATE LOT
SET GNDR CODE = 's'
WHERE LOT ID NUM = 799
AND FROM DATE>= CURRENT DATE
```

Sequenced Modifications คือ การแก้ไขข้อมูลโดยสามารถอ้างอิงเวลาได้ทั้ง อดีต ปัจจุบัน หรือ อนาคต มีรายละเอียดดังนี้

Sequenced Insert เราสามารถทำได้โดยใส่ข้อมูลที่ต้องการบันทึก พร้อมกับช่วงของ Valid Timeให้กับข้อมูลไม่ว่าจะเป็น อดีต ปัจจุบัน หรือ อนาคต

ตัวอย่าง "วันที่ 01/12/2006 มีฝูงวัวตัวเมียเข้ามาในฟาร์ม โดยมีหมายเลขฝูงเป็น 433 " คำสั่ง SQL ที่ใช้ คือ

```
INSERT INTO LOT
VALUES (433, 'h', DATE '01/12/2006', DATE '9999-12-31')
```

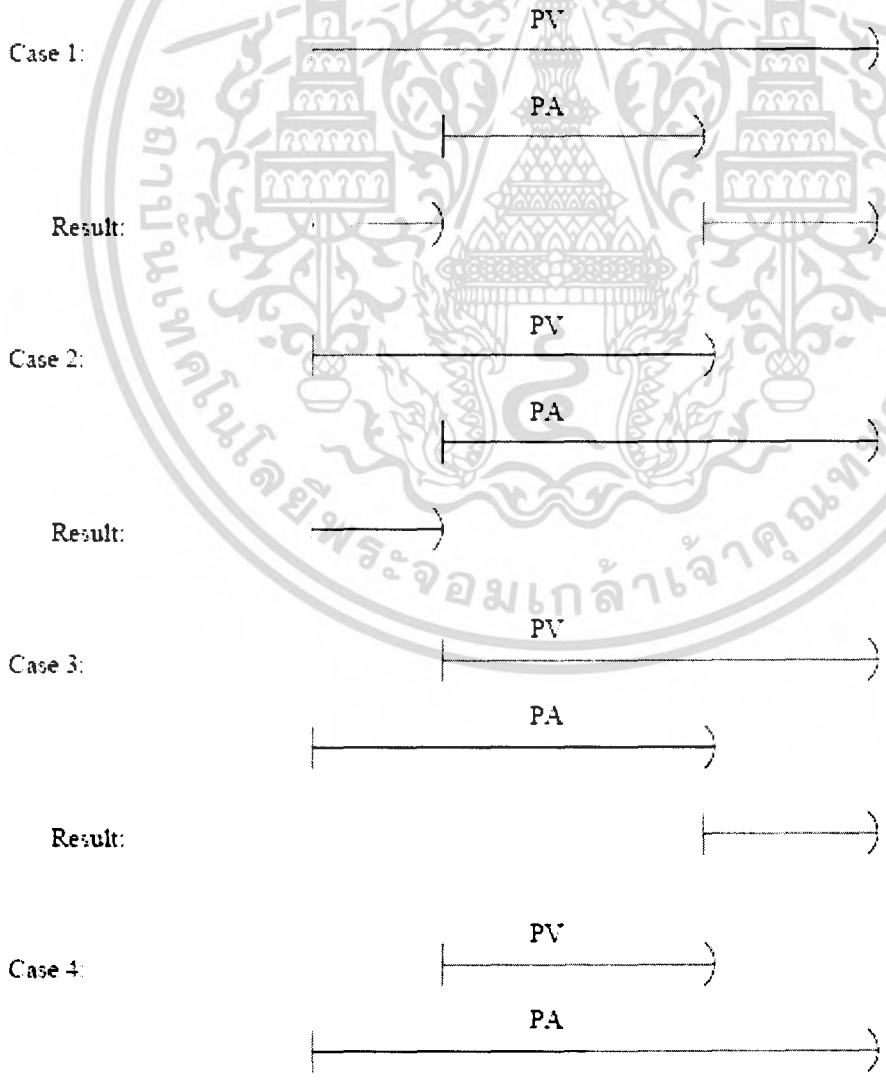
Sequenced Delete จะต้องกำหนดช่วงเวลาที่ต้องการลบ โดยจะต้องเปรียบเทียบช่วงเวลา Valid Time ของข้อมูลที่ต้องการลบ (แทนด้วย PV) กับช่วงเวลาที่จะลบ (แทนด้วย PA) ซึ่งมีได้ 4 กรณี คือ

1. PA อยู่ในช่วงเวลาของ PV ผลของการลบจะออกมาเป็นสองช่วงคือ จากจุดเริ่มต้นของ PV จนถึงจุเริ่มต้นของ PA และอีกช่วงคือ จากจุดสิ้นสุด PA จนถึงจุดสิ้นสุดของ PV โดยจะต้อง Update ให้ TO_DATE ของ PV = FROM_DATE ของ PA Insert row ใหม่โดยกำหนดให้ FROM_DATE = TO_DATE ของ PA และ TO_DATE = TO_DATE ของ PV

2. PV เกิดก่อน PA แต่จุดสิ้นสุดของ PV อยู่ในช่วงเวลาของ PA ผลจากการลบจะอยู่ในช่วง จากจุดเริ่มต้นของ PV จนถึงจุดเริ่มต้นของ PA โดยจะต้อง Update ให้ TO_DATE ของ PV = FROM_DATE ของ PA

3. PA เกิดก่อน PV แต่จุดสิ้นสุดของ PA อยู่ในช่วงเวลาของ PV ผลจากการลบจะอยู่ในช่วง จากจุดสิ้นสุด PA จนถึงจุดสิ้นสุดของ PV โดยจะต้อง Update ให้ FROM_DATE ของ PV = TO_DATE ของ PA

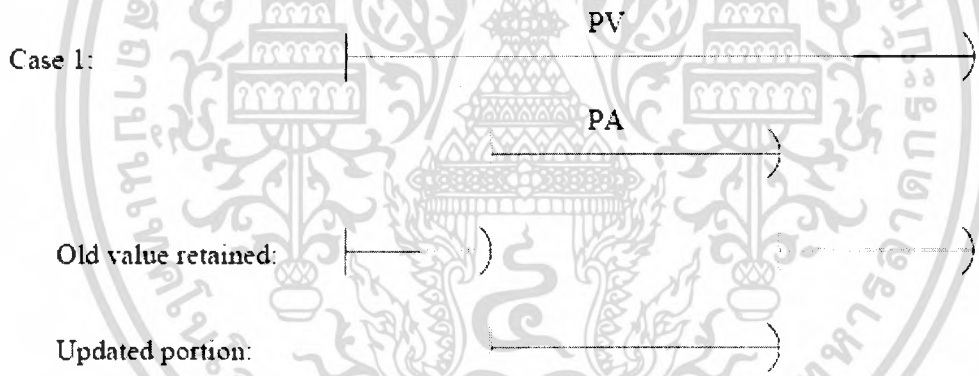
4. PV อยู่ในช่วงเวลาของ PA จะลบข้อมูลของ PV ออกทั้งช่วงโดยจะต้อง Delete ข้อมูลที่มี TO_DATE >= FROM_DATE ของ PA และ TO_DATE <= TO_DATE ของ PV



รูปที่ 2.15 แสดงเส้นเวลาการตรวจสอบ Sequence Delete

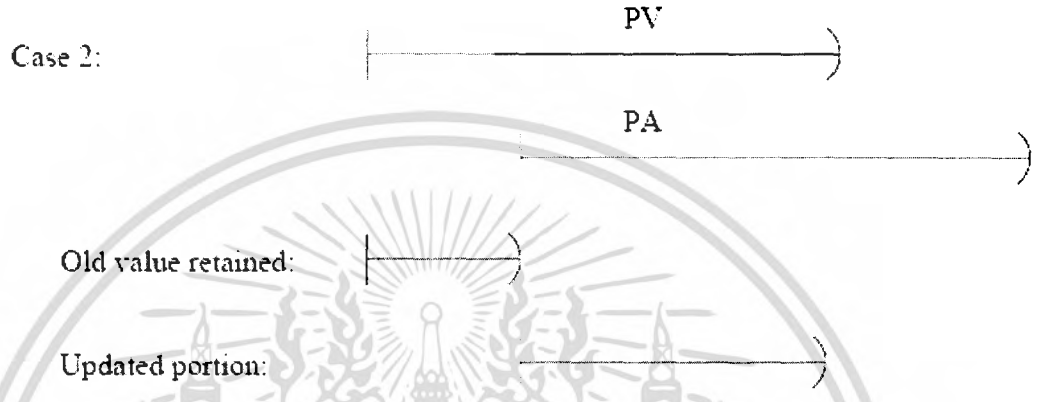
Sequenced Update จะต้องกำหนดช่วงเวลาที่ต้องการแก้ไข โดยจะต้องเปรียบเทียบช่วงเวลา Valid Time ของข้อมูลที่ต้องการแก้ไข (แทนด้วย PV) กับช่วงเวลาที่จะแก้ไข (แทนด้วย PA) ซึ่งมีได้ 4 กรณี คือ

1. PA อยู่ในช่วงเวลาของ PV ผลของการแก้ไขจะได้มาเป็นสามช่วงคือ จากจุดเริ่มต้นของ PV จนถึงจุเริ่มต้นของ PA อีกช่วงคือ จากจุดสิ้นสุด PA จนถึงจุดสิ้นสุดของ PV และข้อมูลที่จะแก้ไข และช่วง Valid Time ของข้อมูลนั้น
   a) โดยจะต้อง Update ให้ TO_DATE ของ PV = FROM_DATE ของ PA
   b) Insert ข้อมูลเดิม โดยกำหนดให้ FROM_DATE = TO_DATE ของ PA และ TO_DATE = TO_DATE ของ PV
   c) Insert row ที่มีช่วง Valid Time และ ข้อมูลใหม่ลงไปในฐานข้อมูล



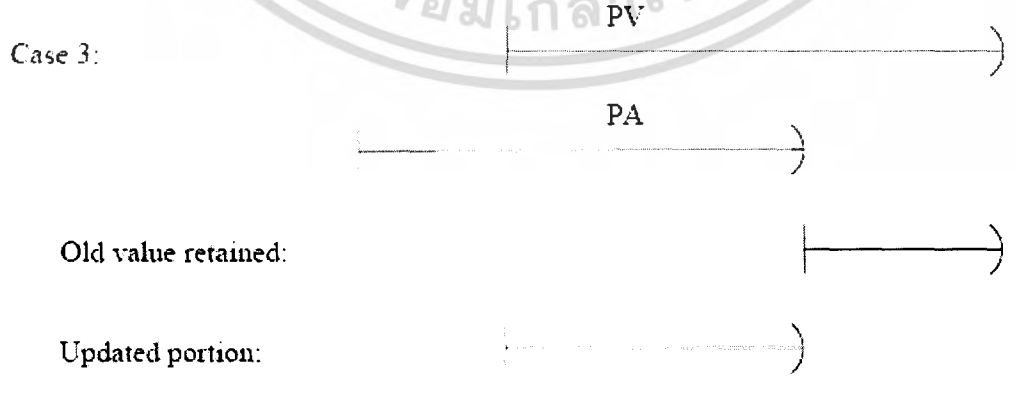
รูปที่ 2.16 แสดงเส้นเวลาการตรวจสอบ Sequence Update กรณีที่ 1

2. PV เกิดก่อน PA แต่จุดสิ้นสุดของ PV อยู่ในช่วงเวลาของ PA แก้ไขโดย
   a) Update TO_DATE ของ PV = FROM_DATE ของ PA
   b) Insert ข้อมูลใหม่ โดยกำหนดให้ FROM_DATE = FROM_DATE ของ PA และ TO_DATE = TO_DATE ของ PV



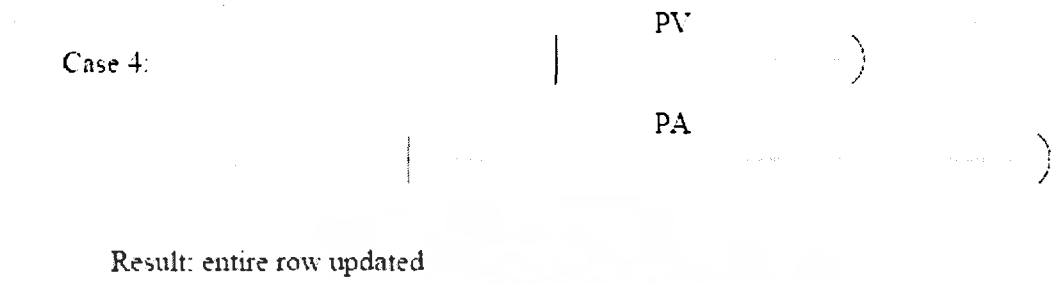
รูปที่ 2.17 แสดงเส้นเวลาการตรวจสอบ Sequence Update กรณีที่ 2

3. PA เกิดก่อน PV แต่จุดสิ้นสุดของ PA อยู่ในช่วงเวลาของ PV แก้ไขโดย
   a) Update FROM_DATE ของ PV = TO_DATE ของ PA
   b) Insert ข้อมูลใหม่ โดยกำหนดให้ FROM_DATE = FROM_DATE ของ PV และ TO_DATE = TO_DATE ของ PA



รูปที่ 2.18 แสดงเส้นเวลาการตรวจสอบ Sequence Update กรณีที่ 3

4. PV อยู่ในช่วงเวลาของ PA Update ข้อมูลเดิมด้วยข้อมูลใหม่ และช่วงเวลาใหม่



รูปที่ 2.19 แสดงเส้นเวลาการตรวจสอบ Sequence Update กรณีที่ 4

Nonsequenced Modifications เป็นการแก้ไขข้อมูลโดยที่จะอ้างถึงช่วงเวลาที่ต้องการแก้ไข ตัวอย่าง การแก้ไขแบบนี้คือ "ต้องการลบข้อมูลของวัวฝูงที่ 234 ที่มีอยู่ตั้งแต่สามเดือนขึ้นไป" โดยมีคำสั่ง SQL ดังนี้

```
DELETE FROM LOT
WHERE LOT ID NUM = 234
        AND (TO_DATE – FROM_DATE_MONTH)>INTERVAL '3' MONTH
```

ในการแก้ไข แบบนี้จะแก้ไขเหมือนฐานข้อมูลธรรมดา คือถ้าเป็นการลบข้อมูลก็คือจะลบออกจากฐานข้อมูลเลย

ข้อควรระวัง คือ ผู้ใช้ต้องมั่นใจว่าการแก้ไขข้อมูลที่ทำนั้นจะมีผลกระทบกับแถวที่ต้องการแก้ไขเท่านั้น เนื่องจากอาจจะทำให้เกิดการแก้ไขข้อมูลที่ขัดกับกฎบังคับความถูกต้องของข้อมูล หรือไปมีผลกระทบกับแถวที่ไม่เกี่ยวข้อง และถ้าเกิดในกรณีนี้ขึ้นการแก้ไขข้อมูลจะไม่สมบูรณ์ตามที่ต้องการได้

2.10 transaction-Time

Transacetion Time : เวลาที่ข้อมูลนั้นถูกบันทึกลงในฐานข้อมูล Transaction Time ส่วนใหญ่จะได้จาก DBMS และจะไม่สามารถแก้ไขข้อมูล Transaction Time ในอดีตได้เนื่องจากจะ

ผิดกฎของเวลาที่ว่าเราไม่สามารถเปลี่ยนแปลงอดีตได้ [2] ยกตัวอย่างเช่น ข้อมูลเงินเดือนของพนักงาน ก. เดือนละ 15000 บาทต่อเดือนได้ ถูกบันทึกเมื่อวันที่ 1 มกราคม 2543 ได้มีการวางแผนว่าจะปรับเงินเดือนให้เป็น 18000 บาทต่อเดือน ในวันที่ 1 มกราคม 2544 แต่ฝ่ายการเงินได้ บันทึกข้อมูลเมื่อวันที่ 30 ธันวาคม 2543 ดังนั้น Transaction Time ของข้อมูลของพนักงาน ก. ที่มีเงินเดือน 15000 บาทต่อเดือน คือ 1 มกราคม 2543 ถึงวันที่ 30 ธันวาคม 2543 เป็นต้น เราสามารถนำ Transaction Time มาใช้เพื่อกำหนดช่วง Transaction Time ของแต่ละข้อมูลได้ เช่น ช่วงเวลา ของ Transaction Time ของพนักงาน ก. ที่มีเงินเดือน 15000 บาทคือ วันที่ 1 มกราคม 2543 ถึง 30 ธันวาคม 2543

2.11 Transaction-Time State Table

คุณสมบัติพื้นฐานของการเปลี่ยนแปลงตามช่วงเวลา(Time-varying) ของ Temporal data นั้นคือ Valid Time กับ Transaction Time ซึ่งในเนื้อหาส่วนนี้จะกล่าวถึง Transaction Time

Transaction-Time State Table คือ ตารางที่เพิ่มการบันทึกข้อมูล Transaction-Time ซึ่งคือ เวลาเริ่มต้น เมื่อมีการเปลี่ยนแปลงFactของข้อมูลในแต่ละ Row และเวลาสิ้นสุดเมื่อFactของข้อมูลในแต่ละTupleนั้นมีการเปลี่ยนแปลงไป ซึ่งจะต่างจาก Valid Timeอย่างชัดเจน โดย Valid Time จะบันทึกช่วงเวลาที่Factของข้อมูลในTupleนั้นเป็นจริง

โดยทั่วไปแล้ว Transaction-Time State Table เหมาะสมกับการใช้บันทึกเพื่อทำความเข้าใจในสิ่งที่มีวิวัฒนาการ การเปลี่ยนแปลงอย่างค่อยๆเป็นค่อยๆไป เพื่อทำความเข้าใจในระบบนั้นๆ

ตัวอย่างการใช้งาน Transaction Time State Table เช่น ระบบแจ้งบันทึกรวบรวมดวงดาว ดวงดาวของ Washington Double Star (WDS) โดยในบัญชีของ WDS จริงๆ Schema จะประกอบไปด้วย 20 colum แต่ในที่นี้ เราจะอ้างถึงแค่บางส่วนของ colum ดังตาราง โดย

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	Mag_ First
00	00	08	75	30	'A 1248'	10.5
05	57	40	00	02	'BU 1190'	6.5
04	13	20	50	32	'CHR 15'	15.5
01	23	70	-09	55	'HJ 3433'	10.5

รูปที่ 2.20 ตาราง WDS

- 5 Colum แรกจะเป็นFactที่บอกถึงการวางตำแหน่งของดวงดาวบนท้องฟ้า Right Ascension และ Declination
- Colum Discoverer จะเป็นรหัสอักษร 1-3 ตัว ตามด้วยหมายเลขรหัสของผู้ค้นพบ โดย Colum นี้เป็น Primary Key
- Colum Mag_First เป็นcolumที่บอกขนาด(Magnitude)ของดวงดาว

เราสร้างตารางใหม่ชื่อ WDS_TT ซึ่งจะเป็น Audit log บันทึกการเปลี่ยนแปลงFactข้อมูลของตาราง WDS และที่สำคัญคือ เพิ่มColum Trans_Start และ Trans_Stop เพื่อเก็บเวลาในการเปลี่ยนแปลงทันทีทันใด ซึ่งต่างจากตารางเดิมที่ไม่มี timestamps

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	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	1990-02-09	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

รูปที่ 2.21 ตาราง WDS_TT

Trans_Start จะระบุถึงเวลาเมื่อมีการ insert tuple ใหม่ลงในตารางเดิม (WDS) หรือ เมื่อมีการ update ด้วยข้อมูลใหม่ Trans_Stop จะถูกระบุถึงเมื่อมีการ delete ข้อมูลTupleนั้นๆออกจากตารางWDS เมื่อมีการ update ด้วยข้อมูลใหม่ ในตัวอย่างนี้ใช้ timestamps เป็น DATEs แต่หากงานที่ต้องการความละเอียดของเวลามากขึ้น เช่น งานที่มีหลายๆ Transactions เกิดขึ้นในเวลาหนึ่งวัน ก็ควรใช้ timestamps เป็น TIMESTAMP แทนซึ่งจะได้ความละเอียดในหน่วยของวินาที

หากTrans_Stopแสดงเป็น (9999-12-31) จะหมายถึง ตลอดไป ซึ่งแปลว่า สถานะ Fact ของ tuple นั้นในตาราง WDS ยังคงเป็นจริงอยู่ในปัจจุบัน ซึ่งหากดูจากตาราง WDS_TT มี Trans_Stop เป็น (9999-12-31) 4 tuple หมายความว่า บัจจุบันทั้ง 4 tuple นี้ยังเป็นFactที่จริงอยู่ในตาราง WDS

ข้อมูลของ LDS3402 ถูกinsert ครั้งแรกตอนปลายปี1993 และเมื่อถูกตรวจสอบว่าFactนี้ในความเป็นจริงเกิดการเปลี่ยนแปลงไปก็จะทำการ delete tuple นี้ในเดือนกรกฎาคมปี1996 ซึ่งเวลา Trans_Stop ก็จะเป็นเวลาที่ทำการ delete

ข้อมูลของ A 1248 ถูกinsertครั้งแรกในปี1989 และถูกแก้ไปเปลี่ยนแปลงหลายครั้งโดย ในเดือนพฤศจิกายน ปี 1992 ทำการแก้ไขตำแหน่งของ RA_Sec, ในเดือนพฤษภาคม ปี1994 ทำการแก้ไขขนาดMag_First, และในเดือนกรกฎาคม ปี1995 ทำการแก้ไข RA_Sec อีกเล็กน้อย การเปลี่ยนแปลงแก้ไขทั้งหมดนี้ไม่ได้หมายถึง ว่าในเวลานั้นๆดวงดาวมีการเปลี่ยนแปลง แต่หมายถึงณ.เวลานั้นๆ ได้มีคนตรวจสอบค้นพบการเปลี่ยนแปลงและทำการบันทึกข้อมูลเหล่านั้นลงในตาราง WDS ของฐานข้อมูลและเมื่อมีการแก้ไขเปลียนแปลงลง ตาราง WDS แล้วFact tuple ข้อมูลในcolum Trans_Stop ซึ่งเป็น(9999-12-31) ก็จะเป็นเท็จ จึงใส่เวลาบัจจุบันที่มีการแก้ไขลงไป ซึ่งแสดงถึง Fact ของ tupleนี้เป็นจริงณ.ช่วงเวลาดังกล่าวเท่านั้น

Maintaining the Audit Log

การสร้าง Audit Log (ตาราง WDS_TT) นั้นสามารถสร้างมีการทำงานอย่างอัตโนมัติโดยใช้ trigger ซึ่งประกาศตอนสร้างตารางWDS โดย Audit Log จะมี side-effect ก็ต่อเมื่อมีการเปลี่ยนแปลงตารางWDS โดย Primary Key ของ Transaction-Time State Table เพียงแค่เพิ่ม Trans_Start ต่อท้ายPrimary Key เดิม เช่นPrimary Key ของตาราง WDS_TT จะใช้ (Discoverer, Trans_Start)

การใช้งาน Trigger ทำให้ Audit Log จะทำการบันทึกทุกๆ การเปลี่ยนแปลงของตาราง WDS โดย เมื่อมีการ Insert tuple ใหม่ลงในตารางWDS ก็จะมีการinsert tuple นั้นๆลงใน Audit Log ด้วยเช่นกัน โดยจะให้ Trans_Start เป็นตำแหน่งเวลาบัจจุบัน(CURRENT_DATE) ในส่วนของ Trans_Stop นั้นจะให้เป็น Forever คือ (9999-12-31) และเมื่อเกิด logical delete tuple นี้ Trans_Stop ก็จะเปลี่ยนเป็นค่าเวลาบัจจุบัน now ใน Audit_Log แต่จะไม่มีtupleใดถูกdelete ใน Audit Log นะ

ในส่วนของ update จะทำ delete ก่อนแล้วจึงตามด้วย insert

CREATE TRIGGER **INSERT WDS**

AFTER INSERT ON WDS

REFERENCING NEW AS N

FOR EACH ROW

```
        INSERT INTO WDS TT (RA Hour, RA Min, RA Sec, Dec Degree, Dec Minute,
        Discoverer, Mag First, Trans Start, Trans Stop)
        VALUES (N.RA Hour, N.RA Min, N.RA Sec, N.Dec Degree, N.Dec Minute,
        N.Discoverer, N.Mag First, CURRENT DATE, DATE '9999-12-31')


CREATE TRIGGER DELETE WDS
AFTER DELETE ON WDS
REFERENCING OLD AS O
FOR EACH ROW
        UPDATE WDS TT
        SET STOP TIME = CURRENT DATE
        WHERE WDS TT.Discoverer = O.Discoverer
        AND WDS TT.Trans Stop = DATE '9999-12-31'
CREATE TRIGGER UPDATE WDS
AFTER UPDATE ON WDS
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
        BEGIN ATOMIC

                UPDATE WDS TT
                SET Trans Stop = CURRENT DATE
                WHERE WDS TT.Discoverer = O.Discoverer
                AND WDS TT.Trans Stop = DATE '9999-12-31';

                INSERT INTO WDS TT (RA Hour, RA Min, RA Sec,Dec Degree, Dec Minute,
                Discoverer, Mag First,Trans Start, Trans Stop)
                VALUES (N.RA Hour, N.RA Min, N.RA Sec,N.Dec Degree, N.Dec Minute,
                N.Discoverer, N.Mag First,CURRENT DATE, DATE '9999-12-31');
        END
```

2.11.2 uerying the Audit Log ใน Transaction-time state tables จะพิจรณาการquery 3 ประเภท คือ

1. Current State Query ของ WDS table เราทำการqueryธรรมดาโดยตรงจากตารางWDS หรือจาก Audit Log ก็ได้

```
SELECT RA_Hour, RA_Min, RA_Sec, Dec_Degree, Dec_Minute, Discoverer, Mag_First
FROM WDS_TT
WHERE Trans_Stop = DATE '9999-12-31'
```

ประโยชน์ของ Audit Log ที่เห็นได้อย่างชัดเจนคือเวลาที่นำมาใช้ในการ rollback ตาราง WDS ไปยังสถานะก่อนหน้า เช่น เราต้องการตาราง WDS ในขณะที่อยู่ในวันที่ 1 เมษายน 1994

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	Mag_ First
00	00	08	75	30	'A 1248'	10.5
05	57	40	00	02	'BU 1190'	6.5
04	13	20	50	32	'CHR 15'	15.5
01	23	70	-09	55	'HJ 3433'	10.5

รูปที่ 2.22 ตาราง WDS

```
CREATE VIEW WDS_April_1 AS
SELECT RA_Hour, RA_Min, RA_Sec, Dec_Degree, Dec_Minute, Discoverer, Mag_First
FROM WDS_TT
WHERE Trans_Start <= DATE '1994-04-01' AND DATE '1994-04-01' < Trans_Stop
```

ผลลัพธ์

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	Mag_ First
00	00	09	75	30	'A 1248'	12.0
05	57	40	00	02	'BU 1190'	6.5
04	13	20	50	32	'CHR 15'	15.5
01	23	70	-09	55	'HJ 3433'	10.5
02	33	10	-09	25	'LDS3402'	10.6

รูปที่ 2.23 ตาราง WDS_April_1

ผลลัพธ์จากการQueryจากตารางWDS_TT ซึ่งเป็น Audit Log แสดง Fact ของข้อมูล LDS3402 (ซึ่งในเวลาบัจจุบันถือว่าเท็จ) และ Fact ของข้อมูล A 1248 (ซึ่งในเวลาบัจจุบันถือว่าเท็จ เพระ Magnitude และ ตำแหน่ง) ซึ่งการquery ได้ผลลัพธ์เป็น WDS_April_1 นั้นจะตรงกับข้อมูลที่ได้จากการquery ข้อมูลจากตาราง WDS ของDBMS ในวันที่ 1 เมษายน 1994 จริงๆ

ดังนั้น หากเราถามว่าดาวดวงไหนที่มีขนาด Mag_First น้อยกว่าหรือเท่ากับ 11 กับ ตาราง WDS_April_1

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	Mag_ First
00	00	09	75	30	'A 1248'	12.0
05	57	40	00	02	'BU 1190'	6.5
04	13	20	50	32	'CHR 15'	15.5
01	23	70	-09	55	'HJ 3433'	10.5
02	33	10	-09	25	'LDS3402'	10.6

รูปที่ 2.24 ตาราง WDS_April_1 (จากAudit Log)

```
SELECT Discoverer
FROM WDS_April_1
WHERE Mag_First <= 11.0
```

ผลลัพธ์จะออกมาเป็น

Discoverer
'BU 1190'
'HJ 3433'
'LDS3402'

รูปที่ 2.25 ผลลัพธ์

และ หากเราถามว่าดาวดวงไหนที่มีขนาด Mag_First น้อยกว่าหรือเท่ากับ 11 กับ ตารางWDS

RA_ Hour	RA_ Min	RA_ Sec	Dec_ Degree	Dec_ Minute	Discoverer	Mag_ First
00	00	08	75	30	'A 1248'	10.5
05	57	40	00	02	'BU 1190'	6.5
04	13	20	50	32	'CHR 15'	15.5
01	23	70	-09	55	'HJ 3433'	10.5

รูปที่ 2.26 ตาราง WDS (บัจจุบัน)

```
SELECT Discoverer
FROM WDS
WHERE Mag_First <= 11.0
```

ผลลัพธ์จะออกมาเป็น

Discoverer
'A 1248'
'BU 1190'
'HJ 3433'

รูปที่ 2.27 ผลลัพธ์

2. Sequence Query และ 3. Non-Sequence Query จะต้องqueryจากตาราง Transaction-time state tables

2.11.2.1 Sequence Query

พิจรณาคำถาม " When was it recorded that A 1248 had a magnitude other than 10.5 ? "

เราแบ่งความหมายของคำถามออกมาเป็น 2 ส่วน โดยส่วนแรก "When was it recorded" เป็นการถามที่ต้องอ้างอิงถึงเวลาเราจึงควรนึกถึง Transaction time ดังนั้นจึงใช้ตารางWDS_TT และแสดงว่าค่าที่จะ return กลับออกมาจะต้องเป็นค่าของช่วงเวลา จากความสัมพันธ์ข้างต้นที่กล่าวมาจึงสามารถสรุปได้ว่า เราควรส่งคำถามประเภท Sequence Query เข้าไป

```
SELECT Mag_First, Trans_Start, Trans_Stop
FROM WDS_TT
WHERE Discoverer = 'A 1248'
  AND Mag_First <> 10.5
```

ผลลัพธ์

Mag_ First	Trans_ Start	Trans_ Stop
12.0	1989-03-12	1992-11-15
12.0	1992-11-15	1994-05-18

รูปที่ 2.28 Sequence Query

พิจรณาคำถาม " When was it recorded that a star had a magnitude equal to that of A 1248 ? "

เหมือนตัวอย่างที่แล้ว คือ เราแบ่งความหมายของคำถามออกมาเป็น 2 ส่วน โดยส่วนแรก "When was it recorded" เป็นการถามที่ต้องอ้างอิงถึงเวลาเราจึงควรนึกถึง Transaction time ดังนั้น

จึงใช้ตารางWDS_TT และแสดงว่าค่าที่จะreturnกลับออกมาจะต้องเป็นค่าของช่วงเวลา จากความสัมพันธ์ข้างต้นที่กล่าวมาจึงสามารถสรุปได้ว่า เราควรส่งคำถามประเภท Sequence Query เข้าไป โดยอีกส่วนหนึ่งของคำถาม " that a star had a magnitude equal to that of A 1248 " จะหมายถึงการทำ self-join ซึ่งจะแสดงใน Oracle เป็น

```
SELECT   W1.Discoverer,
         GREATEST( W1.Trans Start, W2.Trans Start ),
         LEAST( W1.Trans Stop, W2.Trans Stop )

FROM     WDS_TT W1 ,
         WDS_TT W2

WHERE    W1.Discoverer  =  'A 1248'          AND
         W2.Discoverer <>  W1.Discoverer     AND
         W1.Mag First   =  W2.Mag First      AND
         GREATEST( W1.Trans Start, W2.Trans Start ) < LEAST( W1.Trans Stop,
W2.Trans Stop )
```

ผลลัพธ์

Discoverer	Trans_ Start	Trans_ Stop
'HJ 3433'	1994-05-18	1995-07-23
'HJ 3433'	1995-07-23	9999-12-31

รูปที่ 2.29 จากผลลัพธ์จะสรุปได้ว่า มีการบันทึกไว้ในเดือนพฤษภาคม ปี1994 ว่า HJ3433 นั้นมีขนาดเท่ากับ A 1248 และตอนนี้ก็ยังคงมีขนาดเท่านั้นอยู่

2.11.2.2 Non-Sequence Query ใช้บน Transaction-time tables แล้วจะได้ผลอย่างแท้จริงในกรณีเพื่อระบุความเปลี่ยนแปลงที่ผ่านมา

พิจรณาคำถาม " When was the RA_Sec position of a double star corrected ? "(ณ.เวลาใดในอดีตที่ตำแหน่ง RA_Sec ถูกแก้ไข)

```
SELECT   W1.Discoverer,
         W1.RA_Sec     AS  Old_Value,
```

```
        W2.RA_Sec      AS  New_Value,
        W1.Trans_Stop  AS  When_Changed

FROM    WDS_TT AS W1,
        WDS_TT AS W2

WHERE       W1.Discoverer   =   W2.Discoverer
  AND   W1.Trans_Stop   =   W2.Trans_Start
  AND   W1.RA_Sec       <>  W2.RA_Sec
```

ผลลัพธ์

Discoverer	Old_ Value	New_ Value	When_ Changed
'A 1248'	00	09	1992-11-15
'A 1248'	09	08	1995-07-23

รูปที่ 2.30 จากผลลัพธ์แสดงให้เห็นว่า ตำแหน่งของ A 1248 ถูกเปลี่ยนแปลงแก้ไข 2 ครั้ง โดยในครั้งแรกจาก 00 เป็น 09 และครั้งที่สองจาก 09ไปเป็น 08

2.11.3 Modifying the Audit Log

การแก้ไขเปลี่ยนแปลงแบบ Sequence และ non-Sequence นั้นสามารถเปลี่ยนแปลงสถานะ Factของข้อมูลในตาราง Valid-time Table ได้แต่ไม่สามารถทำอย่างนั้นได้กับAudit Logเพราะเป็นการฝ่าฝืน ละเมิด semantic ของ Audit Log

ในทางปฏิบัติวันนี้เราอาจทำการ insert ข้อมูลลงไปตรงๆใส่ในตาราง WDS_TT(Audit Log)ด้วยข้อมูล Trans_Start มีค่าเป็น(1994-04-01)ได้ แต่นั่นหมายถึง ในตาราง WDS ในวันนี้จะต้องมี tuple นั้นๆอยู่แล้ว ไม่สามารถแก้ไขอดีตได้ เนื่องจากเหตุผลที่ว่า อาจเกิดการแก้ไข Audit Log ในอดีตโดยไม่ได้รับอนุญาตได้ ดังนั้น วิธีที่สมควรคือควรจะมีการให้การเปลี่ยนแปลงAudit Log เป็นหน้าที่ของ trigger เพียงผู้เดียว

2.12 ฐานข้อมูลเชิงพื้นที่

Spatial Database คือ ฐานข้อมูลที่เก็บจัดการspatial data โดย spatial data จะหมายถึงข้อมูลซึ่งมีชนิดเป็น Spatial Data Type (SDTs) ในData Modelของข้อมูลชนิดนั้นๆ โดยมีการจัดการฐานข้อมูลได้โดยใช้query language โดยประเภทของข้อมูลในSpatial Databaseแบ่งออกเป็น 2 ส่วนที่สำคัญคือ

2.12.2 Non-Spatial Data คือ ข้อมูลที่ใช้ในการบรรยาย เป็นข้อมูลที่เกี่ยวกับคุณลักษณะต่าง ๆ ที่เกี่ยวข้องกับพื้นที่นั้น ๆ (Associated Attributes) ข้อมูล เหล่านี้ได้แก่ ข้อมูลประชากร และลักษณะของพื้นที่ เช่นเขตลาดกระบัง มี 8 ตำบล มีพื้นที่ทั้งหมด 65,535.5 ตารางกิโลเมตร

2.12.3 Spatial Data คือ ข้อมูลเชิงพื้นที่เป็นข้อมูลที่ระบุตำแหน่งพิกัดที่ตั้ง ข้อมูลประเภทนี้เป็นสิ่งที่จำเป็นอย่างยิ่งเพราะ ระบบสารสนเทศข้อมูลแผนที่ทางภูมิศาสตร์เป็นระบบข้อมูลที่ต้องอ้างอิงภูมิศาสตร์ (Geo-Referenced) ข้อมูลเหล่านี้ได้แก่แผนที่ต่างๆเป็นต้น ข้อมูลที่มีการเก็บเป็นรูปทรงต่างๆ โดยในกรณีที่เป็น 2-D จะมีรูปร่างพื้นฐานที่ใช้ในการนำเสนอ 2 ประเภท คือ **Object:**

2.12.3.1 สำหรับแสดงขอบเขตของพื้นที่ เช่น แสดงพื้นที่ของเมือง, แสดงพื้นที่ขอบเขตของพื้นที่ป่าไม้ หรือ แสดงบริเวณแม่น้ำ เป็นต้น เพื่อใช้แบ่งแยกให้เห็นความแตกต่างโดยจะใช้ Single objects ช่วยนำเสนอ

2.12.3.2 Space: ใช้ในการอธิบายพื้นที่ของตนเองนำเสนอโดยจุดทุกๆจุดในพื้นที่

เราอาจประเภทของรูปทรงได้เป็น 2 ประเภท

1. **Single objects**: เป็น Fundamental abstractions ของ modeling คือ Point, Line, Region

โดยจะเป็นการเก็บข้อมูลแบบ Vector Model คือข้อมูลเชิงพื้นที่ถูกจัดเก็บในลักษณะเชิงเส้นที่มีโครงสร้างในการกำกับ ก่อน-หลัง หรือ ซ้าย-ขวา โดยการใช้เส้น และจุดเป็นองค์ประกอบพื้นฐานในการจัดเก็บข้อมูลเชิงพื้นที่ โครงสร้างของแฟ้มข้อมูลระบบสารสนเทศแผนที่ทางภูมิศาสตร์ในระบบนี้จะประกอบด้วย ตำแหน่งพิกัด X, Y และ ตัวชี้ลำดับก่อน-หลัง หรือ ซ้าย-ขวา ของข้อมูล ข้างเคียงโครงสร้างของข้อมูลระบบนี้จะใช้เนื้อที่ในการจัดเก็บน้อย แต่การปรับปรุงแก้ไขจะทำได้ยาก และไม่สะดวกเท่าที่ควร ข้อมูลเชิงพื้นที่ในรูปเวคเตอร์จะเก็บอยู่ใน 4 รูปแบบดังต่อไปนี้

รูปที่ 2.31 Single objects:

รูปที่ 2.32 Point

- **Point** จุดเป็นค่าพิกัดในตำแหน่งแนวนอน (X axis) และแนวตั้ง (Y axis) ไม่มีความยาว หรือพื้นที่จะใช้แสดงข้อมูลบนพื้นโลกที่เป็นลักษณะของตำแหน่งที่ตั้งเช่น ที่ตั้งของมหาวิทยาลัย เป็นต้น

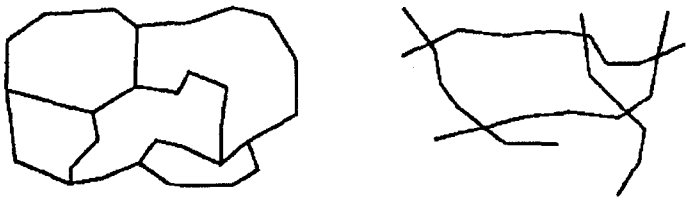
รูปที่ 2.33 Line

- **Line** เส้นเป็นชุดของค่าพิกัดตำแหน่งที่ต่อเนื่องกันโดยมีจุดเริ่มต้นและจุดปลายมีเฉพาะความยาวไม่มีพื้นที่ จะใช้แสดงข้อมูลบนพื้นโลกที่เป็นลักษณะของเส้นเช่น เส้นทางคมนาคม , ขอบเขต
การปกครอง

รูปที่ 2.34 Polygon

- **Polygon** พื้นที่รูปปิดเป็นชุดของค่าพิกัดตำแหน่งที่ต่อเนื่องกัน โดยมีจุดเริ่มต้นและจุดปลายอยู่ที่ตำแหน่งเดียวกันมีทั้งความยาวหรือเส้นรอบรูป และพื้นที่จะใช้แสดงข้อมูลที่เป็นลักษณะของพื้นที่
เช่น พื้นที่ของบริเวณป่าไม้ เป็นต้น

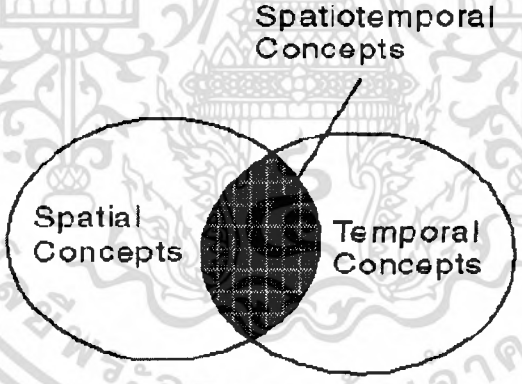
2. Spatially related collection of objects: Partition, Networks



รูปที่ 2.35 Partition, Networks

ซึ่งSpatial Databaseจะต้องทำการเชื่อมต่อข้อมูลทั้ง non-Spatialกับ Spatial สองอย่างนี้เข้าด้วยกัน เพื่อทำให้ข้อมูลnon-Spatialนั้นสามารถอธิบายข้อมูลในรูปแบบSpatialได้

2.13 ฐานข้อมูล Spatio –Temporal



รูปที่ 2.36 ฐานข้อมูล Spatio –Temporal

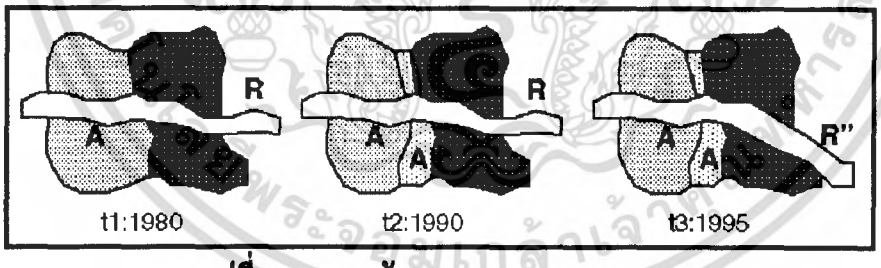
Spatio-Temporal Database เป็นการผสมผสานคุณสมบัติระหว่างฐานข้อมูลเชิงภูมิศาสตร์ ฐานข้อมูลเชิงเวลา เนื่องจากมีกรณีการคิดว่าตลอดเวลาข้อมูลเชิงภูมิศาสตร์ต่างๆ สามารถเคลื่อนที่และเปลี่ยนรูปร่างได้ตลอดเวลา

Spatio-Temporal Database นั้นจะจัดเก็บ ข้อมูลเชิงภูมิศาสตร์โดยอ้างอิงกับข้อมูลเชิงเวลา วัตถุประสงค์เพื่อให้ฐานข้อมูลเชิงภูมิศาสตร์แบบสามารถตอบคำถามได้มากขึ้นเนื่องจากจะมีการ

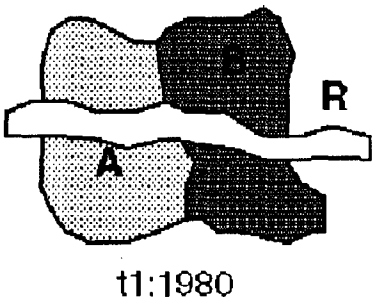
เก็บข้อมูลเชิงภูมิศาสตร์ที่มีการเปลี่ยนแปลงเก็บไว้ด้วยโดยอาศัยคุณสมบัติของฐานข้อมูลเชิงเวลาเข้ามาช่วย โดย Spatio-Temporal Database จะแบ่งออกเป็น 3 ประเภทตาม คือ

1. โปแกรมประยุกต์ ที่จัดการกับข้อมูลเชิงภูมิศาสตร์ที่มีการเคลื่อนที่แบบต่อเนื่องและไม่เปลี่ยนรูปร่างเมื่อเวลาต่างกัน เช่น ระบบ Navigational systems ที่จัดการกับวัตถุที่เคลื่อนที่
2. โปแกรมประยุกต์ ที่จัดการกับวัตถุที่มีการเคลื่อนที่แบบต่อเนื่อง และมีการเปลี่ยนรูปร่างเมื่อเวลาต่างกัน เช่น การเก็บข้อมูลของ พายุ ซึ่งมาการเคลื่อนที่ มีการเปลี่ยนคุณสมบัติ (ความรุนแรง) และมีการเปลี่ยนรูปร่าง
3. โปแกรมประยุกต์ ที่จัดการกับวัตถุที่มีการไม่เคลื่อนที่และเปลี่ยนรูปร่างเมื่อเวลาต่างกัน เช่น การเจริญเติบโตของพืช
4. โปแกรมประยุกต์ ที่จัดการกับวัตถุที่มีการไม่เคลื่อนที่และไม่เปลี่ยนรูปร่างเมื่อเวลาต่างกัน แต่คุณสมบัติของพิกัดนั้นเปลี่ยนไป

ตัวอย่างของโปแกรมประยุกต์ Spatio Temporal แสดงสภาวะการเปลี่ยนแปลงของภูมิประเทศในช่วงระยะเวลาต่างๆ กัน

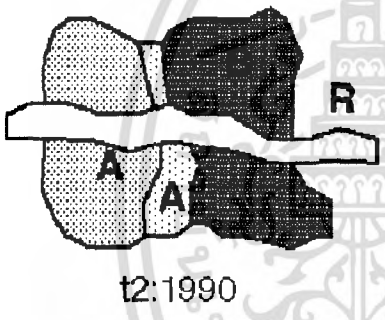


รูปที่ 2.37 ฐานข้อมูล Spatio –Temporal



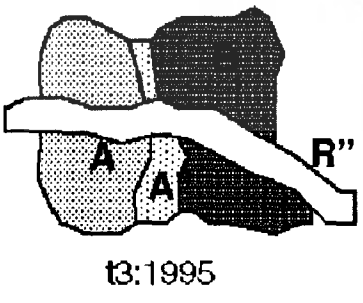
รูปที่ 2.38 ในปี1980 : ภูมิประเทศA

- ในปี1980 : ภูมิประเทศA จะเป็นขอบเขตแดนติดต่อกันกับภูมิประเทศB โดยจะมีแม่น้ำR ไหลผ่านตรงกลางระหว่างAกับBโดยAจะมีพื้นที่ชนิดดินเหนียวไม่สามารถปลูกป่าไม้ได้ ส่วนBจะเป็นเขตพื้นที่ดินที่สามารถปลูกป่าไม้ได้



รูปที่ 2.39 ในปี1990 : ภูมิประเทศA

- ในปี1990: ในปีนี้พบว่าพื้นที่Aถูกแบ่งแยกออกเป็น2 ส่วนคือ AและA' และพื้นที่ขอบเขตพื้นที่ของBมีพื้นที่ป่าไม้หนาแน่นมากขึ้น ส่วน A'เป็นเขตที่ดินที่มีป่าไม้ขึ้นเล็กน้อย



รูปที่ 2.40 ในปี 1995: แม่น้ำ R ได้เปลี่ยนแปลงตำแหน่งและกลายเป็นแม่น้ำR'

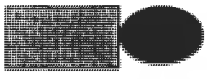
Spatial Data Types และ Operations

Spatial Relationships

- **Topological relationships:**

Disjoint =>

Touch =>



Overlap =>

In =>

Cover =>

Equal =>

รูปที่ 2.41 Topological relationships

Direction relationships

: Above, below, north_of, southwest_of,...

Metric relationships

: Distance

Querying

การQueryingสามารถทำได้ 2 ทาง

เชื่อมต่อผ่านทางoperations ของspatial algebraไปยังfacilities ของDBMSโดยใช้DBMS query language

Graphical presentation ของspatial dataเพื่อแสดงผลลัพธ์ของการqueries และ รับinput graphicalของ STD (spatial data types) เพื่อใช้ในการqueries

Fundamental spatial algebra operations

Spatial selection: จะreturn Objectซึ่งตรงกับ spatial predicate ของqueryที่ส่งเข้าไป

- ยกตัวอย่างเช่น ในทั้งหมดมีเมืองไหนบ้างที่ไม่ใหญ่เกิน 300km. จากเมือง Lausanne
- SELECT cname
  FROM cities c
  WHERE dist(c.center, Lausanne.center) < 300 AND c.pop > 500K

Spatial join: jointจะcompares ตามpredicateทุกค่า spatial attribute ระหว่าง 2 objects ที่ joined กันบนพื้น

- สำหรับแม่น้ำแต่ละสายที่ไหลผ่านกลางเมืองSwitzerland, ให้หาทุกๆเมืองที่อยู่ห่างเป็นระยะไม่น้อยกว่า50เมตร
- SELECT c.cname
  FROM river r, cities c
  WHERE r.route intersects Switzerland.area ADN dist(r.route, c.area) < 50KM

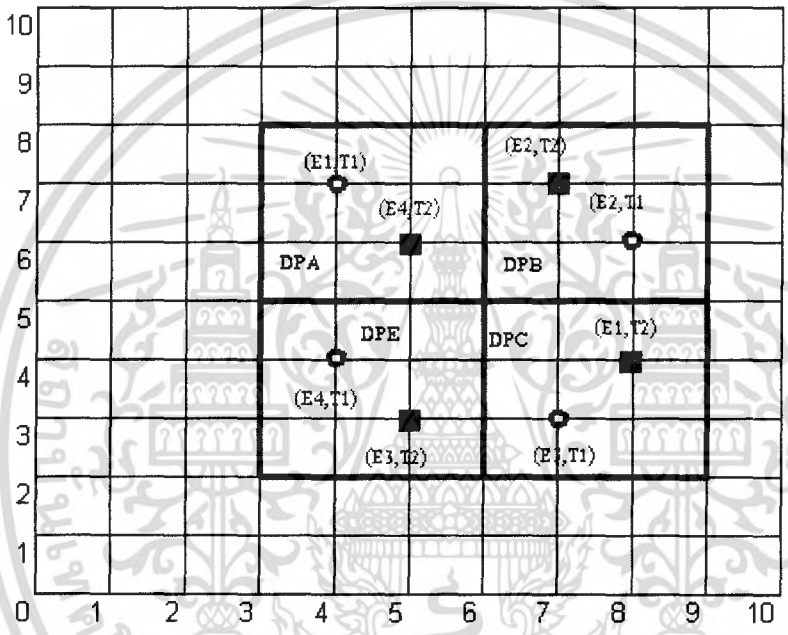
รูปแบบของคำถามใน Spatio-Temporal Database
โดยหัวข้อนี้จะได้ยกตัวอย่างความสัมพันธ์ระหว่างพิกัดของพนักงานและตำแหน่งของแผนก

ตารางที่ 2.4 แสดงข้อมูลของพนักงาน

รหัสพนักงาน	ชื่อ	รายได้	ตำแหน่งที่พักอาศัย PIONT(X,Y)	VALID_FROM (MM-YYYY)	VALID_TO (MM-YYYY)
0001	E1	150000	POINT(4,7)	(01-2005)	(01-2006)
0002	E2	160000	POINT(8,6)	(01-2005)	(01-2006)
0003	E3	170000	POINT(7,3)	(01-2005)	(01-2006)
0004	E4	180000	POINT(4,4)	(01-2005)	(01-2006)
0001	E1	170000	POINT(9,4)	(01-2006)	(01-2007)
0002	E2	180000	POINT(7,7)	(01-2006)	(01-2007)
0003	E3	190000	POINT(5,3)	(01-2006)	(01-2007)
0004	E4	200000	POINT(5,6)	(01-2006)	(01-2007)

ตารางที่ 2.5 แสดงข้อมูลตำแหน่งแผนกต่างๆ

รหัสตึก	ชื่อ	ตำแหน่งที่โต๊ะทำงาน Rectangle (Xll,Yll,...Xru,Yru)
0001	DPA	Rectangle (3,5,6,8)
0002	DPB	Rectangle (6,5,9,8)
0003	DPC	Rectangle(6,2 ,9,5)
0004	DPE	Rectangle(3,2 ,6,5)



รูปที่ 2.42 แสดงจุดพิกัดโต๊ะของพนักงานและตำแหน่งของแผนก

โดยได้พิจารณาจาก ปัจจัยหลักสองประการคือ

a. History คือ ข้อมูลหรือเหตุการณ์ที่เราสนใจ โดยข้อมูลนั้นสามารถเป็นได้ทั้งข้อมูลทั่วไปหรือเป็นข้อมูลเชิงพื้นที่ก็ได้ ยกตัวอย่างเช่น หากผู้ใช้ต้องการทราบถึงข้อมูลการอัตราการขึ้นเงินเดือนของพนักงานคนหนึ่ง ข้อมูลนั้นคือ history นั่นเอง

b. Timestamp หมายถึง ช่วงเวลาที่ใช้อ้างอิงกับ history นั้น

โดยสามารถแบ่งรูปแบบของคำถามใน Spatio-temporal Database ออกได้เป็น 4 รูปแบบดังนี้

1. Intra History Cross Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยสนใจเพียง 1 domain ในช่วงเวลาของ Timestamp ที่ต่างกัน เช่น ในช่วงปี 2005-2007 เงินรายได้เฉลี่ยของพนักงาน EMP1 เป็นกี่เปอร์เซนต์ เป็นต้น จะสังเกตุได้ว่าสิ่ง

ที่เราต้องการนั้นคือ ข้อมูลของพนักงาน EMP1 แต่ความสัมพันธ์ข้อมูลที่ต้องการนั้นผ่าน 2 ช่วงเวลาคือ t1= [01-2005,01-2006) และ t2=[01-2006,01-2007)

2. Cross History Cross Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจตั้งแต่ 2 domain ขึ้นไป ในช่วงเวลาของ Timestamp ที่ต่างกัน เช่น ในช่วงปี 2005-2007 พนักงานคนไหนที่ย้ายที่อยู่มาในแผนกABUILDING มากขึ้น เป็นต้น คำตอบของคำถามนี้เราจะต้องนำ ข้อมูลสองส่วนมาเปรียบเทียบกันคือ ข้อมูลที่อยู่ของพนักงานและข้อมูลตำแหน่งของแผนก ส่วนช่วงเวลาที่อ้างอิงข้อมูลนั้นผ่าน 2 ช่วงเวลาคือ t1= [01-2005,01-2006) และ t2=[01-2006,01-2007)
3. Cross History Intra Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจตั้งแต่ 2 domain ขึ้นไป ในช่วงเวลาของ Timestamp เดียวกัน เช่น ปีไหนที่ที่อยู่ของพนักงานEMP1 ใกล้แผนก DPA มากที่สุด เป็นต้น คำตอบของคำถามนี้เราจะต้องนำ ข้อมูลสองส่วนมาเปรียบเทียบกันคือ ข้อมูลที่อยู่ของพนักงาน EMP1 และข้อมูลตำแหน่งของแผนก ส่วนช่วงเวลามีช่วงเดียว คือ [01-2005,01-2006)
4. Intra History Intra Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจเพียง 1 domain และคำตอบที่ได้อยู่ในช่วงเวลาเดียวกัน เช่น ปีไหนพนักงานEMP1 ได้เงินเดือนการเงินเดือนสูงที่สุด เป็นต้น ข้อมูลที่เราสนใจนั้นมีเพียงอย่างเดียว คือ เงินเดือนของพนักงานEMP1 และช่วงเวลามีช่วงเดียว คือ [01-2006,01-2007)

บทที่ 3
การออกแบบและพัฒนา

3.1 การสนับสนุน Valid-Time ด้วย Oracle Workspace Manager

ปัจจุบันมีหลายโปรแกรมประยุกต์ (Application) ที่ต้องการเก็บข้อมูลโดยอ้างอิงกับช่วงเวลาที่ข้อมูลนั้นเป็นจริง ซึ่งต้องการให้ข้อมูลที่นั้นมูลนั้นเป็นจริงในช่วงเวลาที่กำหนดเท่านั้น Workspace Manager ทำจะช่วยอำนวยความสะดวกทำให้กับนักพัฒนาสามารถสร้างตารางที่สนับสนุน Valid Time ได้ง่าย (และยังสามารถเพิ่มให้ตารางเดิมที่เรามีอยู่แล้วสามารถสนับสนุน Valid Time ได้ ซึ่งจะกล่าวในหัวข้อต่อ ไป) ถ้าเราทำให้ตารางของเราสนับสนุน Valid Time แต่ละแถว (Tuple) จะมีคอลัมน์ (Column) ที่เก็บ ช่วง Valid Time เพิ่มขึ้นมา

เราสามารถเจาะจงช่วงเวลาในแต่ละส่วนได้ และ Workspace Manager ยังสามารถให้ความหมั่นใจในการสอบถาม (Queries) นำข้อมูลมาแสดง (Projection) รวมถึงการแก้ไขข้อมูลต่างๆ (INSERT ,UPDATE , DELETE) ได้ว่าสามารถทำได้อย่างถูกต้องและเหมาะสมกับช่วงเวลาของ Valid Time ที่เราต้องการ โดยสามารถกำหนดช่วงเวลาของ Valid Time ได้ทั้งอดีต ปัจจุบัน หรืออนาคต

3.1.1 Workspace Manager คืออะไร

Workspace คือ พื้นที่จำลอง ซึ่งทำให้ผู้ใช้ตั้งแต่หนึ่งคนขึ้นไปสามารถแลกเปลี่ยนหรือใช้ข้อมูลร่วมกันในฐานข้อมูลได้

Workspace Manager เป็นส่วนที่คอยจัดการ Workspace ให้สามารถทำงานได้หลาย Version คือ ฐานข้อมูลสามารถมีหลายเวอร์ชั่นใน Record เดียวกันใน Workspace เดียวหรือหลาย WorkSpac

3.1.2 ข้อมูลชนิด WM_PERIOD

ชนิดข้อมูล WM_PERIOD ใช้เพื่อในการระบุช่วงเวลา Valid Time ให้กับ Session , Workspace และสำหรับแต่ละแถว (Tuple) ในตารางที่ได้ทำ Version-enable ไว้แล้ว เราสามารถประกาศชนิดข้อมูล WM_PERIOD ได้ดังนี้

CREATE TYPE WM_PERIOD AS OBJECT (validFrom TIMESTAMP WITH TIME ZONE,

validTill TIMESTAMP WITH TIME ZONE);

validFrom คือ เวลาเริ่มต้นของช่วงเวลาที่เรากำหนดโดยเวลานี้จะอยู่ในช่วงเวลานั้นด้วย

validTill คือ เวลาสิ้นสุดของช่วงเวลาที่เรากำหนดโดยเวลานี้จะไม่อยู่ในช่วงเวลานั้นด้วย

ดังนั้นช่วงของเวลาที่ข้อมูลเป็นจริง คือ ตั้งแต่ validFrom ไปจนถึงก่อน validTill

ตัวอย่างการกำหนดช่วงเวลาของฐานข้อมูล ให้อยู่ในช่วงเดือน มกราคม 2549

EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-2006', 'MM-DD-YYYY'), TO_ DATE('02-01-2006', 'MM-DD-YYYY'));

INSERT INTO employees VALUES('Baxter',40000,
WMSYS.WM_PERIOD(TO_DATE('01-01-2000', 'MM-DD-YYYY'),
DBMS_WM.UNTIL_CHANGED));

3.1.3 ค่าคงที่ที่ช่วยในการสนับสนุน Valid Time

ตาราง 3.1 Constants for Valid Time Support

Constant Explanation

ค่าคงที่	ความหมาย
DBMS_WM.DEFAULT_VALID_FROM	ค่าเวลาอัตโนมัติ สำหรับส่วนของ validFrom
DBMS_WM.DEFAULT_VALID_TILL	ค่าเวลาอัตโนมัติ สำหรับส่วนของ validTill
DBMS_WM.MIN_TIME	ค่าเวลาที่น้อยที่สุดเท่าที่ Workspace Manager จะสนับสนุนได้ซึ่งขณะนี้คือเริ่มจาก วันที่ 01 มกราคม ค.ศ. 4713 (4713 BCE).
DBMS_WM.MAX_TIME	ค่าเวลาที่มากที่สุดเท่าที่ Workspace Manager จะสนับสนุนได้ซึ่งในขณะนี้วันสุดท้ายคือ วันที่ 31 - ธันวาคม-9999.
DBMS_WM.UNTIL_CHANGED	คือ การคงค่าของเวลา ไว้ที่ DBMS_WM.MAX_TIME จนกว่าจะมีการเปลี่ยนแปลงค่า ของข้อมูล

3.1.4 API ที่ช่วยสนับสนุน Valid Time

ตารางที่ 3.2 แสดงรายการของโปรแกรมย่อย ที่ช่วยสนับสนุน Valid Time รายละเอียดของ Parameter ต่างๆ

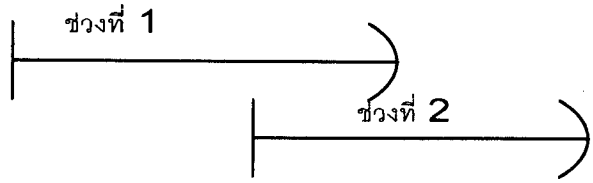
โปรแกรมย่อย	สนับสนุน Valid Time
EnableVersioning	ถ้าค่าของ validTime parameter เป็น TRUE,จะแสดงว่าตารางนั้นเป็น version-enabled ที่ทำให้สามารถสนับสนุน Valid Time ได้ column ชื่อ WM_VALID ซึ่งข้อมูลเป็นชนิด WM_PERIOD จะถูกเพิ่มเข้าไปในตาราง สำหรับข้อมูลที่มีอยู่ในฐานข้อมูลแต่ก่อนแล้ว column WM_VALID จะถูกกำหนดให้ validFrom timestamp เป็น SYSTIMESTAMP และ validTill timestamp เป็น DBMS_WM.UNTIL_CHANGED.
DisableVersioning	เป็นกำหนดให้เป็น version-disabled โดยจะเป็นการสิ้นสุดการเก็บค่า ตัวแปล keepWMValid หรือเป็นการลบ column WM_VALID
SetValidTime	เป็นการกำหนดช่วงเวลาให้กับ session ของ Valid Time
GetValidFrom	ใช้ดูค่า validFrom timestamp จากช่วงเวลา session ของ valid time
GetValidTill	ใช้ดูค่า validTill timestamp จากช่วงเวลา session ของ valid time

3.1.5 Operators ที่ช่วยในการสนับสนุน Valid Time

Workspace Manager ได้จัดเตรียมส่วนที่ช่วยกระทำการตรวจสอบความสัมพันธ์ระหว่างสองช่วงเวลา (relationship checking operators) และ ผู้ใช้สามารถนำไปประยุกต์เพื่อทำการค้นหาหรือสอบถาม (query) ได้ ตัวกระทำการ (Operator) ที่ได้จะเตรียมให้มีอยู่สองลักษณะ คือ

1. ตรวจสอบความสำพันธ์ระหว่างสองช่วงเวลา ที่มีผลลัพธ์เป็นค่าเท่ากับ 1 เมื่อมีความสำพันธ์ที่ได้ตรวจสอบ และจะได้ผลลัพธ์เป็น 0 เมื่อไม่มีความสัมพันธ์นั้นอยู่ ได้แก่

WM_OVERLAPS ตรวจสอบความสัมพันธ์ว่าสองช่วงเวลามีการทับซ้อนกันอยู่หรือไม่



รูปที่ 3.1 ตรวจสอบความสัมพันธ์ว่าสองช่วงเวลามีการทับซ้อนกันอยู่หรือไม่

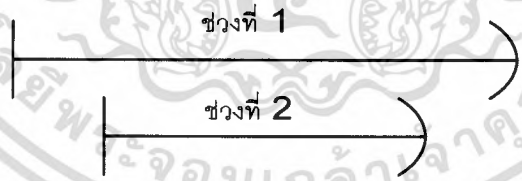
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_OVERLAPS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในช่วง ปี 1990 - 1999 มีรูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_OVERLAPS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2000', 'MM-DD-YYYY'))) = 1;
```

WM_CONTAINS ตรวจสอบความสัมพันธ์ว่า ช่วงเวลาที่ 2 เป็นส่วนหนึ่งของช่วงเวลาที่ 1 หรือไม่



รูปที่ 3.2 ตรวจสอบความสัมพันธ์ว่า ช่วงเวลาที่ 2 เป็นส่วนหนึ่งของช่วงเวลาที่ 1 หรือไม่

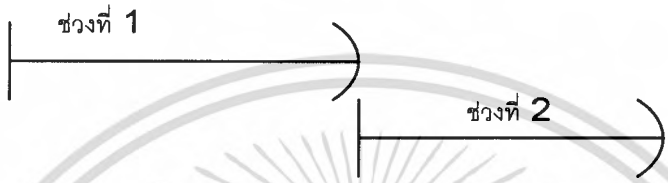
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_COMTAINS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_CONTAINS(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY'))) = 1;
```

WM_MEETS ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรก เป็นเวลาเริ่มต้นของช่วงเวลาที่สอง หรือไม่



รูปที่ 3.3 WM_MEETS

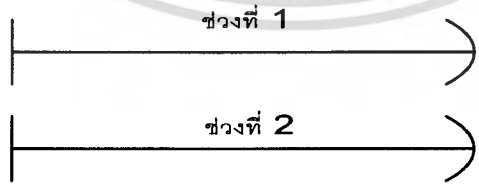
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_MEETS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_MEETS(e.wm_valid,
        wm_period(TO_DATE('01-01-2005', 'MM-DD-YYYY'),
        TO_DATE('01-01-2006', 'MM-DD-YYYY'))) = 1;
```

WM_EQUALS ตรวจสอบว่าช่วงเวลาแรก เท่ากับช่วงเวลาที่สอง หรือไม่



รูปที่ 3.4 WM_EQUALS

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_EQUALS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1990 ถึงวันที่ 1 มกราคม 2005 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_EQUALS(e.wm_valid,
        wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
        TO_DATE('01-01-2005', 'MM-DD-YYYY'))) = 1;
```

WM_LESSTHAN ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรกน้อยกว่า (ก่อน)ช่วงเวลาที่สองหรือไม่

รูปที่ 3.5 WM_LESSTHAN

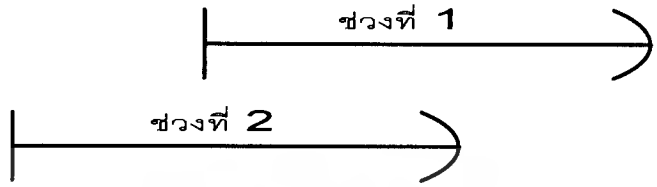
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_LESSTHAN(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
WM_LESSTHAN(TO_DATE('01-01-1980', 'MM-DD-YYYY'),TO_DATE('01-01-1990', 'MM-DD-YYYY'),TO_DATE('01-01-1991', 'MM-DD-YYYY'),TO_DATE('01-01-1992', 'MM-DD-YYYY') = 1

SELECT * FROM employees e
WHERE WM_LESSTHAN(e.wm_valid,
        wm_period(TO_DATE('01-01-2010', 'MM-DD-YYYY'),
        TO_DATE('01-02-2010', 'MM-DD-YYYY'))) = 1;
```

WM_GREATERTHAN ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรกมากกว่า (หลัง)ช่วงเวลาที่สอง หรือไม่



รูปที่ 3.6 WM_GREATERTHAN

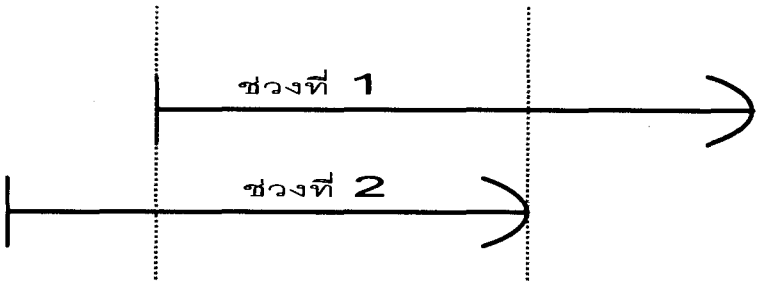
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_GREATERTHAN (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

```
WM_GREATERTHAN(TO_DATE('01-01-1980', 'MM-DD-YYYY'),
        TO_DATE('01-01-1990', 'MM-DD-YYYY')
        TO_DATE('01-01-1970', 'MM-DD-YYYY'),
        TO_DATE('01-01-1980', 'MM-DD-YYYY') = 1

SELECT * FROM employees e
WHERE WM_GREATERTHAN(e.wm_valid,
        wm_period(TO_DATE('01-01-2001', 'MM-DD-YYYY'),
        TO_DATE('01-02-2001', 'MM-DD-YYYY'))) = 1;
```

2. ตรวจสอบความสำพันธ์ระหว่างสองช่วงเวลา ที่มีผลลัพธ์เป็นค่าตามสัมพันธ์ที่ตรวจสอบ หากมีความสัมพันธ์นั้นอยู่ และจะได้ผลลัพธ์เป็น Null เมื่อไม่มีความสัมพันธ์นั้นอยู่ ได้แก่

WM_INTERSECTION จะเปรียบเทียบช่วงเวลาสองช่วงเวลา จะได้ผลลัพธ์เป็นช่วงเวลาที่เป็นจุดตัดของทั้งสองช่วงเวลา และจะได้ผลลัพธ์เป็น Null เมื่อไม่มีความสัมพันธ์นั้นอยู่

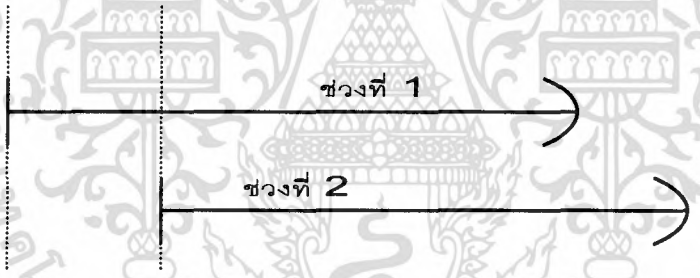


รูปที่ 3.7 WM_INTERSECTION

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_GREATERTHAN (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

WM_LDIFF จะเปรียบเทียบช่วงเวลาสองช่วงเวลาว่ามีการทับซ้อนกันอยู่หรือไม่ จะได้ผลลัพธ์เป็นช่วงเวลาตั้งแต่เวลาเริ่มต้นของช่วงเวลาแรกจนถึงเวลาเริ่มต้นของช่วงเวลาที่สอง หากช่วงเวลาแรกเกิดก่อนช่วงเวลาที่สอง และจะได้ผลลัพธ์เป็น Null เมื่อช่วงเวลาที่สองเกิดก่อนช่วงเวลาแรกหรือไม่มีความสัมพันธ์นั้นอยู่

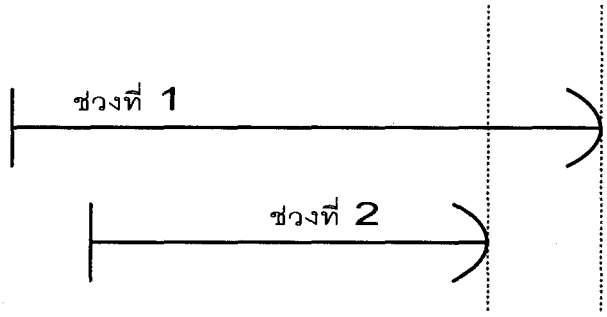


รูปที่ 3.8 WM_LDIFF

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_LDIFF (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

WM_RDIFF จะเปรียบเทียบช่วงเวลาสองช่วงเวลาว่ามีการทับซ้อนกันอยู่หรือไม่ จะได้ผลลัพธ์เป็นช่วงเวลาที่แตกต่างระหว่างสองช่วงเวลาในส่วน ตั้งแต่เวลาสิ้นสุดของช่วงเวลาที่สองจนถึงเวลาสิ้นสุดของช่วงเวลาแรก และจะได้ผลลัพธ์เป็น Null เมื่อช่วงเวลาแรกสิ้นสุดก่อนช่วงเวลาที่สองหรือไม่มีความสัมพันธ์นั้นอยู่



รูปที่ 3.9 WM_RDIFF

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_RDIFF (ช่วงเวลาที่ 1, ช่วงเวลาที่ 2)

3.1.6 การสืบค้นข้อมูลและการแก้ไขข้อมูล

ในหัวข้อนี้เราจะกล่าวถึงลักษณะ และการพิจารณาในการสอบถามต่างๆ (queries) และการใช่ภาษาในการจัดการข้อมูล (insert, update, delete) ที่เกี่ยวกับ Valid Time

3.1.6.1 การสืบค้นข้อมูล (Queries)

การสอบถามต่างๆ เราจะเห็นเฉพาะข้อมูลที่อยู่ในช่วงเวลาที่เรากำหนดให้กับ Session ของตารางเท่านั้น (ตั้งค่าโดยการใช้โพรซิเดอร์ SetValidTime) โดยเราสามารถ Operators ที่ได้กล่าวมาในส่วนที่ มาช่วยในการสอบถามร่วมกับภาษา SQL ได้

3.1.6.2 การแก้ไขข้อมูลด้วย Data Manipulation Language (DML)

ในการแก้ไขข้อมูลด้วย DML (INSERT,UPDATE,DELETE) ในตารางที่ได้ทำ version-enabled ให้สนับสนุน valid time แล้วนั้น เราสามารถกำหนดระยะเวลาให้กับ session ของผู้ใช้แต่ละคนว่าสามารถแก้ไขข้อมูลได้ในช่วงเวลาไหน หรือผู้ใช้สามารถกำหนดเองได้โดยใช้ API SetValidTime (ซึ่งกล่าวไว้ในภาคผนวก ข.). โดยคำสั่งจะมีผลเฉพาะช่วงเวลาที่กำหนดเท่านั้น

ซึ่งโดยปรกติแล้วจะอยู่ในช่วงเวลาปัจจุบันจนกว่าผู้ใช้ทำการกำหนดช่วงเวลาให้กับ Session ใหม่ โดยการแก้ไข้ข้อมูลยังสามารถใช้ร่วมกับ Operators ที่ได้กล่าวไว้ในหัวข้อ 4.6 นอกเหนือจากนี้ยังมีส่วนที่ช่วยจัดการให้เป็นที่สนับสนุน Valid Time มีดังนี้

3.1.6.3 กฎบังคับความถูกต้องของข้อมูล

ในหัวข้อนี้จะเสนอถึงกฎบังคับความสมบูรณ์ของข้อมูลโดย Workspace Manager จะเป็นผู้จัดการกับกฎเหล่านี้แทนผู้ใช้

Referential Integrity Constraints ในที่นี้คือ กฎที่ใช้บังคับสำหรับการอ้างอิงข้อมูล ซึ่งสำหรับการทำ version-enabled ให้สนับสนุน Valid Time นั้นจะแบบออกได้เป็น 2 กรณีคือ

2 กรณีที่ทั้งตารางที่เป็นคีย์หลักเป็นตารางที่ถูกทำ version-enabled ให้สนับสนุน Valid Time แล้ว ตารางที่อ้างถึงข้อมูลต้องเป็นตารางที่ถูกทำ version-enabled ให้สนับสนุน Valid Time แล้วเท่านั้น และข้อมูลที่อ้างถึงนั้นต้องอยู่ในช่วงเวลาของคีย์หลักเท่านั้น

3 กรณีที่ตารางที่เป็นคีย์หลักเป็นตารางที่ไม่ได้ทำ version-enabled ให้สนับสนุน Valid Time แล้ว จะไม่สนใจเวลาสามารถอ้างถึงข้อมูลนั้นได้เลย

Unique Constraints ในที่นี้คือกฎที่บังคับว่าห้ามคีย์หลักซ้ำกันในช่วงเวลาหรือจุดเวลาเดียวกัน ตารางที่ทำให้สามารถสนับสนุน Valid Time ด้วย Workspace Manager นั้นต้องจะกำหนดคีย์หลักหรือกำหนดให้ห้ามมีค่าซ้ำ (Unique) ใน Column ใด Column หนึ่งเสมอ จะไม่ยอมให้มีแถวใดๆ มีคีย์หลักมีค่าซ้ำกันในช่วงเวลาเดียวกัน เช่น หากกำหนดให้รหัสพนักงานเป็นคีย์หลัก ในการบันทึกข้อมูลพนักงานให้รหัสพนักงานมีค่าซ้ำกับข้อมูลที่มีอยู่แล้วและช่วงเวลามีการเหลื่อมล้ำกับข้อมูลเดิม จะไม่สามารถทำได้เนื่องจากมีข้อมูลซ้ำในช่วงเวลาเดียวกัน แต่หากข้อมูลนั้นอยู่คนละช่วงเวลาจะสามารถบันทึกหรือแก้ไขข้อมูลได้ เป็นต้น

3.1.6.4 การแก้ไขตารางที่มีอยู่แล้วให้สามารถสนับสนุน Valid Time

ผู้ใช้สามารถทำให้ตารางที่มีอยู่แล้วนั้นสนับสนุน Valid Time ได้โดยใช้ โพรซิเดอร์ AlterVersionedTable (โดยมีรูปแบบของคำสั่งและรายละเอียดจะแสดงใน ภาคผนวก ข.) ผู้ใช้สามารถกำหนดช่วงเวลาให้กับข้อมูลที่อยู่ในตารางใน Column WM_VALID หรือจะให้ใส่ให้โดยอัตโนมัติช่วงเวลาจะเป็นเวลาปัจจุบันจนถึงจนกว่าจะเปลี่ยนแปลง

3.2 FlashBack Features ช่วยในการสนับสนุน Transaction Time

Oracle Database มีกลุ่มของfeatureที่รู้จักกันในชื่อว่า flashback ซึ่งมีเพื่อ ดู-ขอข้อมูล สถานะของข้อมูลในอดีตจากdatabase object โดยไม่ใช้ อุปกรณ์ที่ช่วยในการสำรองข้อมูล

เราสามารถใช้งาน Flashback ได้ดังนี้

เรียกใช้คำสั่ง query เพื่อข้อมูลในอดีต

เรียกใช้คำสั่ง query เพื่อข้อมูล metadata ซึ่งจะแสดงรายละเอียด การเปลี่ยนแปลง สถานะขอฐานข้อมูลในอดีต

กู้คืนตารางหรือ rows จากข้อมูลในอดีต

Flashback ใช้ ระบบ Automatic Undo Management เพื่อเอา metadata และข้อมูลในอดีตของ Transactions ซึ่งจะอาศัย Undo data (คือ records ที่เปลี่ยนแปลงไปจากการทำงานของ transactionsหนึ่งๆ)

ยกตัวอย่าง เช่น ถ้า เรา Executes คำสั่ง update เพื่อเปลียนแปลงเงินเดือนจาก 1,000 บาท ไปเป็น 1,100 บาท ในส่วน Oracle Database จะทำการเก็บข้อมูล 1,000 ซึ่งเป็นข้อมูลเก่า ไว้เป็น undo data

Undo data เป็น persistent data ดังนั้นข้อมูลเหล่านี้จะไม่หายไปเมื่อมีการปิดเครื่อง โดยปรกติเราจะเก็บ Persistent data แยกจากDatabase เช่น เก็บแยก Drive ดังนั้นหาก Drive ที่เก็บฐานข้อมูลเสียหายเราจึงสามารถกู้คืนข้อมูลได้จาก เป็น persistent data

โดยในด้านการใช้งานกับ Flashback feature นั้น เราสามารถ queryข้อมูลหรือกู้คืนข้อมูลในอดีตได้จาก undo data

นอกจากนี้ Oracles Database ยังใช้ flashback operation การทำงานภายในเพื่อใช้ undo data กรณี Rollback transactions เช่น ถ้า Transaction บาง Transaction ที่ยังไม่ commit ทำงานเกี่ยวข้องกับข้อมูลจำนวนมาก มีการถ่ายข้อมูลจาก database buffer ลง database space ไปแล้วบ้าง โดยที่ยังไม่ commit หาก transaction นี้เกิด fail ขึ้นมา Oracle Database จะนำ undo data ที่เก็บไว้ มายกเลิกข้อมูลที่มีเปลี่ยนแปลงและลง database space ไปแล้ว ให้ย้อนกลับคืนเหมือนเก่า เพราะ transaction
นั้นๆ ยังไม่ได้ Commit

คุณสมบัติที่ช่วยในการพัฒนาโปรแกรมประยุกต์

Oracle Flashback Query

เราสามารถใช้ feature นี้ในการ retrieve ข้อมูลในอดีตที่ต้องการโดย ระบุ AS OF เป็น ส่วนclause ให้กับ คำสั่ง SELECT

DBMS_FLASHBACK package

เราสามารถใช้ featureนี้ในการ set เวลาภายในของOracleของการเก็บข้อมูลในอดีต

Oracle Flashback Version Query

เราสามารถใช้ feature นี้ในการ retrieve metadata และข้อมูลในอดีตโดยสามารถระบุช่วงเวลาได้ เช่น เราสามารถดูทุกๆ rows ใน Table ที่เคยอยู่ในช่วงระยะเวลาที่ระบุ Metadata ของ row เดียวกันนั้นจะแตกต่างกันตาม version โดยข้อมูลที่ถือเป็น metadata จะเป็น start time, end time, type of cahnge operation และ identity ของ transaction ซึ่งจะถูกสร้างขึ้นมาตอน สร้าง row version ใหม่ให้

เราระบุ VERSION BETWEEN เป็นส่วนclause ให้กับ คำสั่ง SELECT ในการสร้าง Flashback Transaction Query

Oracle Flashback Transaction Query

ความสามารถเพื่ออำนวยความสะดวกสำหรับผู้ดูแลฐานข้อมูล (Database Administration)

Administration สามารถใช้งานความสามารถเหล่า นี้ได้โดย

- Database user หรือ database administrator
  - **DBMS_FLASHBACK package**
  - **Flashback Query**
  - **Flashback Version Query**
  - **Flashback Transaction Query**
- ใช้ได้เฉพาะ Database administrator

โดยซึ่ง 3 คุณลักษณะนี้เป็นเทคนิค data recovery ดั่งเดิม

- **Oracle Flashback Table**เราสามารถใช้ Fetureนี้เพื่อกู้คืนข้อมูลใน table ให้เป็นข้อมูลในอดีตที่เราระบุได้ และเราสามารถ restore ข้อมูลใน table ในขณะที่ database ยังทำงานอยู่ได้ด้วย โดยการ undoing นั้นจะกระทำกับ Table ที่เราระบุเท่านั้น
- **Oracle Flashback Drop**เราสามารถใช้ feture นี้เพื่อกู้คืนtableที่ Droped ไปแล้วได้ (การทำงานจะได้ผลตรงข้ามกับการใช้คำสั่ง DROP TABLE)
- **Oracle Flashback Database**

เราสามารถใช้ feture นี้เพื่อขอ database ในอดีตที่เราระบุ โดยการ undoing ยกเลิกทุกๆ การเปลี่ยนแปลงตั้งแต่นั้นเป็นต้นมา ซึ่งจะเร็วกว่าการ restore จาก database backups

3.3 การใช้โดยใช้ ORACLE 10g Spatial เพื่อสนันสนุน Spatial Database

Oracle Spatial ออกแบบมาเพื่อความง่ายในการจัดการฐานข้อมูล เช่น manipulated, retrieved, และ relate การหาความสัมพันธ์กันของ data store ในฐานข้อมูล ที่ต้องอ้างอิงตำแหน่งพิกัดหรืองาน ประเภท ระบบสารสนเทศเชิงภูมิศาสตร์ Geographic Information System (GIS) ยกตัวอย่าง เช่น Spatial data เราจะเห็น road map โดย road map ที่เห็นนั้นจะเป็น 2-Dimensions objects ซึ่งภายใน road map จะประกอบด้วย Points, Line, และ Polygon ซึ่งจะสามารถนำเสนอ cities, roads, และ political boundaries เช่น states หรือ provinces

บริษัทORACLEได้พัฒนาOracle Spatialโดยอ้างอิงจาก Spatial Database Concept โดยทาง ORACLEได้จัดเตรียม SQL schema และ function ไว้สำหรับอำนวยความสะดวกในการจัดการ StorageและRetrieval ข้อมูลรวมถึง Update และ Query ข้อมูลที่เป็นspatial ขึ้นมาในรูปแบบต่างๆ โดยSpatial ประกอบด้วย

1. Schema (MDSYS): เป็นการกำหนดstorage, syntax และ semanticsเพื่อรองรับ geometric data type
2. Spatial indexing mechanism
3. Operators, Function และวิธีการในการ query พื้นที่ที่สนใจ, การquery โดยspatial join และการoperationในการวิเคราะห์ Spatial อื่นๆ
4. Function และ Procedures สำหรับUtilityและ tuning operations
5. Topology Data model ใช้สำหรับการทำงานกับข้อมูลที่เกี่ยวกับ Node, Edges และ faces ใน Topology
6. Network Data model ใช้สำหรับนำเสนอObject ที่ทำการModelแล้วในลักษณะNodes และ link ในNetwork
7. GeoRaster เป็นfeatureซึ่งสามารถใช้ในการเก็บข้อมูล,index,query,analyze และ deliver GeoRaster Data (ซึ่งก็คือ raster image และ gridded data) และ ข้อมูล Metadata ที่เกี่ยวข้องต่างๆ

โดย spatial component ของ spatial feature จะเป็น geometric ซึ่งนำเสนอเป็นshape โดย พิกัดcoordinate

Object-Relational Model

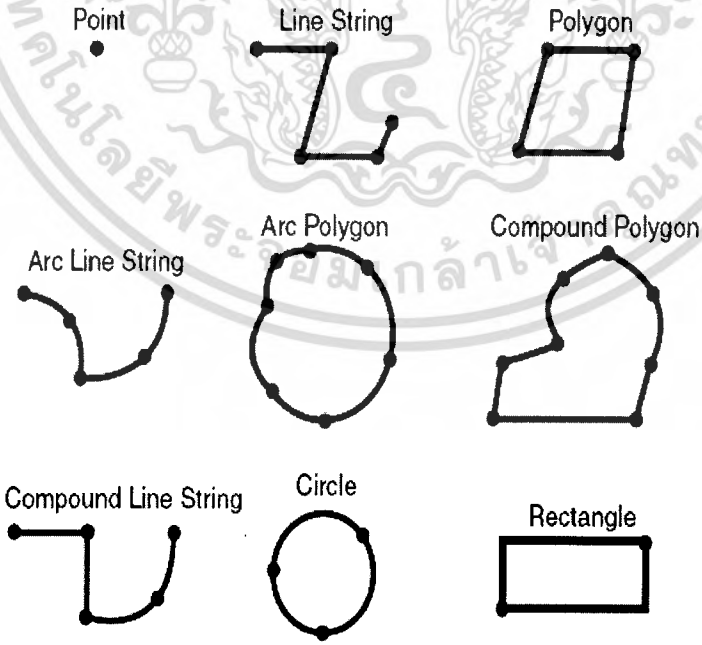
Spatial นั้นใช้ Object-relational Model สำหรับนำเสนอ geometries จึงเก็บค่าentry geometryได้ โดยข้อมูลSpatialที่จะเก็บจะต้องเป็น Oracle Spatial data type ซึ่งจะเป็นVector Data เรียกว่า SDO_GEOMETRY

โดย Oracle table สามารถเก็บข้อมูลประเภทนี้ได้ตั้งแต่ 1 columns ขึ้นไป โดยObject Relational modelในที่นี้คือการใช้ SQL กับ Geometry Types เพื่อ implement Spatial feature table บน Open GIS ODBC/SQL ในรูปแบบเฉพาะสำหรับ geospatial features และเนื่องจาก Geometry มีหลาย geometry type เช่น arcs, circles, compound polygons, compound line strings, และ optimized rectangles จะทำให้เหมาะสมกับการใช้งานModel แบบ object-relational

Geometry Types

Geometry เป็น ordered sequence ของvertices ซึ่งจะconnected โดยตรงกับline segment กับ circular arcs โดยความหมายของ Geometry จะถูกตีความตามtypeที่กำหนด spatial สนับสนุน Primitive types และ geometries composed ของcollections ของtypeนั้นๆ

สำหรับ 2-Dimentional ประกอบไปด้วย:



รูปที่ 3.10 Geometry Types

- Point และ Point Clusters เป็นelementsซึ่งประกอบด้วย จุดordinates 2 จุด คือ (x, y) โดยทั่วไปมักจะเรียกว่า longitude และ latitude เช่น แสดงตำแหน่งของlocation

- Line strings จะประกอบไปด้วยหนึ่งหรือมากกว่าหนึ่งคู่ของจุดขึ้นไป ซึ่งจะเป็นการประกาศ line segments เช่น การนำเสนอเส้นทางของถนน, เส้นทางการบิน และ Polygon อาจจะแสดง state, city, zoning district หรือ city block

- n-Point polygons จะประกอบไปด้วยการเชื่อมต่อของline string เป็นรูปทรงปิด ทำให้เกิดพื้นที่ของpolygon

อย่างไรก็ตาม ไม่สามารถทำSelf-crossing polygonsได้แม้ว่า Self-crossing line strings จะสามารถทำได้ก็ตามแต่หาก line string crossingกับ line string นั่นจะไม่เกิดทรง polygon และก็จะไม่เกิดพื้นที่ด้วยเช่นกัน

- Arc line strings (All arcs are generated as circular arcs)
- Compound polygons
- Compound line string
- Circles
- Optimized rectangles

นอกจากนี้Spatial ยังสนับสนุนstorageและindexของ 3-Dimentional และ 4-Dimentional geometric types โดยการประกาศจุดcoordinatesทั้ง 3หรือ4จุด เพื่อเป็นแต่ละvertexของobjectที่จะสร้าง
อย่างไรก็ตามSpatial Functions (ยกเว้นสำหรับ LRS functions และ MBR-related functions) สามารถทำงานได้กับ 2-Dimentions อันแรกเท่านั้น, และทุกๆspatial operators' ยกเว้น SDO_FILTER จะไม่สามารถใช้งานได้หากยังไม่ได้มีการสร้าง spatial index บนDimensionsที่มากกว่า2-Dimentionsขึ้นไป

Data Model

Spatial data model จะมีโครงสร้างเป็นแบบ hierarchical structure ซึ่งประกอบไปด้วย elements, geometries และ layer โดยที่Layerแต่ละLayerจะประกอบไปด้วยgeometriesซึ่งข้างในก็จะประกอบขึ้นมาจากelements

1. Element:

Elementsจะเป็นโครงสร้างbuilding blockพื้นฐานของgeometries โดยสิ่งที่เป็นspatial element type คือ point, line strings และ polygons ยกตัวอย่างเช่น elementsที่อาจจะเป็นmodel ของกลุ่มดาว (point clusters), ถนน(line strings), ขอบเขตอาณาบริเวณของเมือง(polygons) เป็นต้น โดยแต่ละCoordinateในelement จะเก็บตำแหน่งเป็นคู่ x, y ในส่วนของภายนอกและภายในของComplex polygon จะพิจารณาจาก single element ข้อมูลของPointจะประกอบด้วย 1 coordinate ข้อมูลของLineจะประกอบไปด้วย 2 coordinates เพื่อใช้ในการนำเสนอline segmentของelement ข้อมูลของPolygonจะประกอบไปด้วยคู่ของcoordinates หนึ่งคู่vertexจะเป็นแต่ละline segmentที่จะนำมาประกอบเป็นPolygon โดยที่จุดcoordinateที่ประกาศจะเรียงตามลำดับจะเรียงอยู่รอบๆ polygon (หากหมุนตามเข็มนาฬิกาจะเป็นภายในpolygon หากหมุนทวนเข็มนาฬิกาจะเป็นภายนอก polygon)

2. Geometry

Geometry object จะนำเสนอ Spatial feature โดยจัดให้เป็น Modeled set ของprimitive elements Geometry สามารถประกอบด้วยเพียงelementเดียวได้ ซึ่งก็หมายถึงว่า instanceหนึ่งก็สามารถ supported primitive element types หรือ homogeneous collection หรือ heterogeneous collection ของelements ได้

Multipolygon เช่น หน่วยหนึ่งใช้ในการนำเสนอsetของislandsซึ่งจะเป็นhomogeneous collection ในส่วนของ heterogeneous collection คือ หนึ่งหน่วยจะประกอบไปด้วยelements หลากหลายชนิดเช่น ประกอบด้วย point และ polygon

ตัวอย่างของ geometry เช่น buildable landในเมือง ซึ่งจะสามารถนำเสนอเป็น แม่น้ำด้วย polygonที่มีรอง หรือ แสดงขอบเขตสงวนที่ปลอดการก่อสร้าง

3. Layer

เป็น collectionของgeometriesซึ่งจะมีattribute setเดียวกัน ยกตัวอย่าง เช่น ในหนึ่งlayerของGIS อาจจะรวมลักษณะทางภูมิประเทศ topographical feature, ขณะเดียวกันก็มีคำอธิบายเกี่ยวกับความหนาแน่นของประชากร, และในชั้นที่3ก็อธิบายโครงข่ายของถนน สะพานในพื้นที่(lineและpoint) geometries และ associated spatial index สำหรับแต่ละlayerนั้นจะเก็บลงในฐานข้อมูล(ในTable มาตรฐาน)

4. Coordinate System

อาจเรียกได้อีกแบบว่า spatial reference system โดยที่spatial data ทุกประเภทจะต้องมีcoordinate systemของตัวมันเอง โดยที่ coordinate system จะเป็น georeference (แสดงพิกัดบนโลก) หรือ ไม่เป็น georeference(เช่น Cartesian)

ก่อนOracle Spatial Release 8.1.6. , geometries(object ชนิด SDO_GEOMETRY) จะเก็บเป็นstringของจุดcoordinate โดยไม่มี reference ไปยังcoordinate system

Spatial function และ operators จะสมมุติcoordinate system ซึ่งมีคุณสมบัติOrthogonal กับ Cartesian System ทำให้บางครั้งอาจให้คำตอบที่ผิดพลาดได้เช่น ถ้าEarth-base geometriesซึ่งจะต้องเก็บพิกัดที่เป็น longitude และ latitude coordinate systems

ใน Oracle Spatial Release 8.1.6. นั้นSpatial ได้สนับสนุนระบบCoordinate System ที่หลากหลายมากขึ้น และในระบบพิกัดที่ต่างกันสามารถconvert data ได้

Spatial data สามารถแสดงความสัมพันธ์กันด้วย Cartesian, Geodetic (geographical), Projected, และ local coordinate system มีรายละเอียด ดังนี้

- Cartesian Coordinates: เป็นระบบพิกัดที่มีมาตั้งแต่versionก่อนOracle Spatial Release 8.1.6. ใช้ในการนำเสนอระบบ 2-Dimension และ 3-Dimension

- Geodetic Coordinates: บางทีอาจเรียกว่า geographic coordinates เป็นระบบพิกัดแบบ angular coordinate คือประกอบด้วย longitude และ latitude ซึ่งสัมพันธ์กับ Spherical polar coordinates

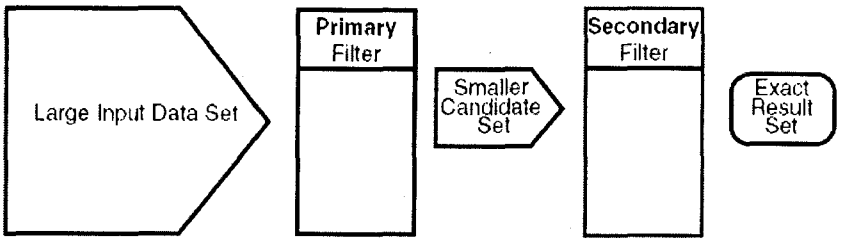
- Project Coordinates: เป็น planar Cartesian coordinate แบบแบน ซึ่งจะให้คำตอบมาจากการ mathematical mapping จากตำแหน่งต่างๆบนพื้นผิวโลกไปเป็นพื้นผิวแนวราบ มักนิยมใช้ในบริเวณที่สนใจเฉพาะเจาะจงเป็นพิเศษ

- Local Coordinates: เป็นCartesian coordinate ที่อยู่บน non-Earth (non-georeferenced) โดยทั่วไป ระบบนี้มักจะใช้บน CAD applications และงานทางด้าน local serveys

หากต้องการให้มีประสิทธิภาพในการใช้งานoperations ของ Geometries Spatial จะใช้ทั้งการModelคำนวณแบบ Cartesian และ Curvilinear เพื่อให้เหมาะสมกับการแสดงความสัมพันธ์กันในระบบ Coordinate System ของ Spatial data

Query Model

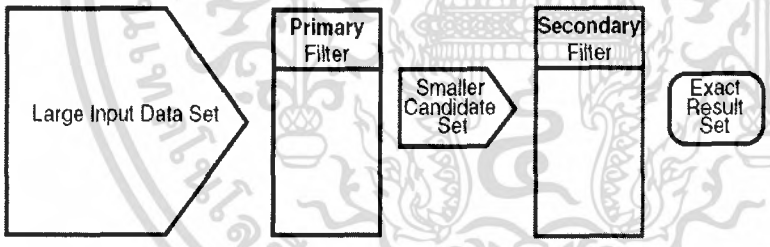
Spatial ใช้ two-tier query model ในการหาคำตอบให้กับ spatial queries และ spatial joins ซึ่งแบ่งออกเป็น 2 ขั้น คือ



รูปที่ 3.11 Query Model

Primary Filter: เป็นการหาcandidate record อย่างคราวๆก่อนที่จะส่งเข้าไปยัง Primary Filter โดย Primary Filter จะทำการcompare geometry อย่างคราวๆ(approximations) เพื่อลดการคำนวณที่ซับซ้อนโดยอาศัยการตัดสินใจที่cost ในการfilter ถูก และเนื่องจากPrimary Filter เป็นการfilter โดยอาศัยการcompare geometric อย่างคราวๆ จึงทำให้ได้ผลลัพธ์ออกมาเป็นsupersetของคำตอบดังนั้นจึงต้องนำไปกรองออกอย่างละเอียดโดยใช้การคำนวณอย่างซับซ้อนต่อไปโดยใช้ Secondary Filter

Secondary Filter: เป็นการหาคำตอบโดยอาศัยการคำนวณที่ซับซ้อน ดังนั้นจึงต้องใช้costแพงมาก



รูปที่ 3.12 WM_MEETS

รูปนี้เป็นตัวอย่างการทำงานของFilterทั้งสอง โดยPrimary Filter operation จะทำการกรองข้อมูลที่เข้ามาซึ่งมีขนาดใหญ่มากให้ออกมาเป็นcandidate setของคำตอบก่อนทำให้ข้อมูลที่จะเข้าไปในSecondary Filter มีขนาดน้อยลง หลังจากนั้นจะส่งไปให้Secondary Filter operation หาคำตอบที่ถูกต้องซึ่งจากกระบวนการทั้งหมดเป็นการลดภาระให้กับSecondary Filter ซึ่งต้องใช้costแพงมากเมื่อมี Primary Filter เข้ามาช่วยลดsetของcandidate ที่จะเข้ามาจึงทำให้มีประสิทธิภาพในการทำงานมากขึ้น

Spatial ใช้ spatial indexในการimplement Primary Filter ในบางกรณีSpatial ไม่จำเป็นต้องใช้ทั้งPrimaryและSecondary Filter เช่น บางกรณี เพียงแค่ใช้Primary Filterก็เพียงพอแล้ว

ยกตัวอย่างเช่น feature ZOOM ที่ใช้ในการmapping application queries สำหรับข้อมูลที่เป็นของ rectangle โดยที่Primary Filter นั้นจะมีความเร็วในการให้supersetของผลลัพธ์เร็วมาก mapping application สามารถdisplayการเปลี่ยนแปลงได้ทันทีบน target area

จุดประสงค์ของการมีPrimary Filterเพื่อความเร็วในการสร้างsubsetของข้อมูลและลด processing burden บนSecondary Filter

ดังที่กล่าวมา Primary Filter จึงมีประสิทธิภาพ (มองในแง่ของการselectได้เร็ว) จึงได้มีการตัดสินใจจากคุณสมบัตินี้นำมาใช้กับ spatial index บนข้อมูลแบบspatial

Index ของ Spatial Data

Spatial Indexจะต้องมีคุณสมบัติดังนี้ คือ

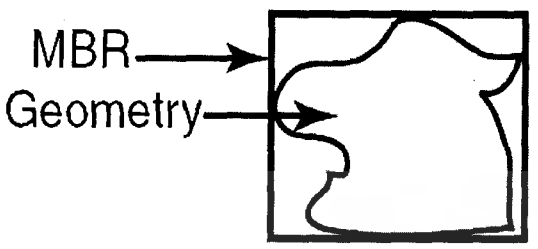
- ค้นหาObjectsข้างในindex data space ซึ่งจะกระทำกับpointหรือareaที่สนใจที่ได้รับมา (อยู่ใน window query)
- ค้นหาpairsของobjects จากข้างใน2 indexed data spaces ซึ่งจะกระทำกันระหว่างspatialกับอีก spatial (spatial joint)

Spatial index จะพิจารณา Logical index โดยที่แต่ละentriesในspatial indexนั้นจะขึ้นอยู่กับ logical ของgeometriesในcoordinate space แต่ค่าของindex valueนั้นจะอยู่คนละdomain

Index entries อาจจะเรียงโดยใช้linearly ordered, และ coordinates สำหรับgeometryอาจจะเป็นคู่ของ integer, floating-point, หรือ double-precision number

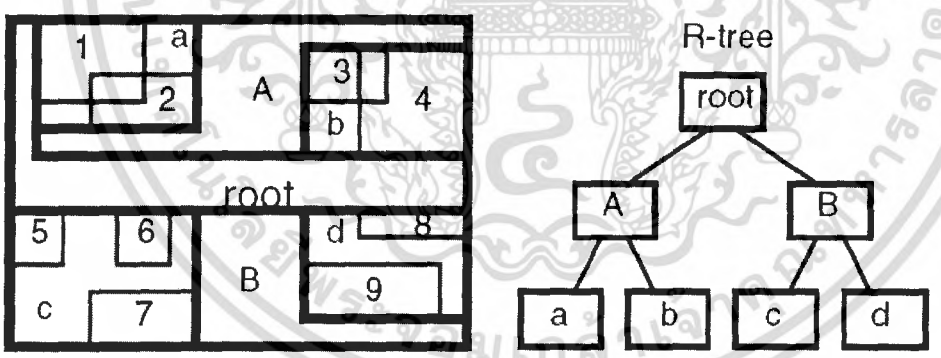
Oracle Spatial โดยdefaultและจะใช้R-tree index หรือ Quadtree indexing หรือ ใช้ทั้งคู่ อย่างไรก็ตามการใช้งานQuadtree indexesนั้นยุ่งยากเกินไป ยากกว่าการใช้R-tree indexing ทำให้แนวทางการพัฒนาประสิทธิภาพการทำงานของindexในreleaseนี้มุ่งไปที่การทำงานของ Spatial R-tree indexesเท่านั้น โดยไม่สนใจปรับปรุง Quad tree index รวมถึงข้อมูลต่างๆที่เคยเป็นของ Quadtree indexก็ได้ถูกตัดออกจากGuide แต่ก็ยังมีการใช้งานอยู่ในเรื่องของOracle Technology Network ดังนั้นหากอยากจะหาข้อมูลเกี่ยวกับQuadtree index จะหาได้เอกสารซึ่งแยกเฉพาะชื่อ Oracle Spatial Quadtree Indexing ดังนั้นจึงขอนำเสนอเนื้อหาเฉพาะconceptและoption ของR-tree indexing

R-Tree Indexing



รูปที่ 3.13 R-Tree Indexing

Spatial R-tree index สามารถใช้เป็นindexของspatial dataได้จนถึง4-Dimensions โดยที่R-tree indexนั้นจะทำการประมาณแต่ละgeometry โดยใช้รูปทรงrectangleทรงปิดที่minimally encloses 1 อัน เรียกว่า MBR (Minimum Bounding Rectangle) แสดงดังรูปข้างบน



รูปที่ 3.14 WM_MEETS

สำหรับLayerของgeometries นั้น R-tree index ประกอบไปด้วย hierarchical index บน MBRs ของgeometries ในlayer ดังรูปข้างบน

- 1ถึง9 นั้นจะเป็นgeometriesในlayer

- a, b, c, และd จะเป็นleaf node ของ R-tree index และจะบรรจุด้วย minimum bounding rectangle ของ geometries โดยที่จะมี pointerชี้ไปยัง geometries ยกตัวอย่างเช่น a ประกอบไปด้วยMBRของ geometries1และgeometries2 , b ประกอบไปด้วยMBR ของ geometries3 และ geometries4 เป็นต้น

- A จะบรรจุ MBRของ a และ b , ส่วน B จะบรรจุ MBR ของ c และ d

- Root จะบรรจุ MBR ของAและB

R-tree index นั้นเก็บอยู่ใน spatial index table (tableชื่อ SDO_INDEX_TABLE ซึ่งอยู่ในview ชื่อUSER_SDO_INDEX_METADATA)

R-tree index จะmaintain sequence object (SDO_RTREE_SEQ_NAME บนviewที่ชื่อว่า USER_SDO_INDEX_METADATA) และจะมีการupdateไปพร้อมๆกันทันทีที่userทำการสร้าง

R-Tree Quality

การทำงานจริงนั้นการที่เราทำ operation insert, delete หลายๆครั้งอาจส่งผลกระทบกับโครงสร้างของR-Tree index ซึ่งอาจส่งผลให้ประสิทธิภาพquery performanceต่ำลงได้

R-Tree index นั้นจะมีโครงสร้างเป็นแบบ Hierarchical Tree structure ด้วยความสูงที่แตกต่างกัน performanceของ R-Tree index structure สำหรับการqueries นั้นจะแปรผันตามพื้นที่และเส้นรอบวงของ index nodes ของ R-tree

โดยถ้าPerformanceของ SDO_FILTER operations ลดต่ำลง และมีการทำoperations insert, update และ delete จำนวนมาก จะส่งผลกระทบกับ geometries ประสิทธิภาพที่ลดต่ำจะนั้นอาจจะเป็น ความสัมพันธ์ของR-Tree index

เราสามารถตรวจสอบประสิทธิภาพได้โดยใช้ function SDO_TUNE.QUALITY_DEGRADATION โดยหากค่าที่คืนกลับมาเป็นค่าที่มากกว่า2 แสดงว่าคุณควรจะrebuilding index ใหม่ โดยใช้คำสั่ง ALTER INDEX REBUILD แต่อย่างไรก็ตามสิ่งที่มีผลกระทบกับประสิทธิภาพนอกจากจะพิจารณาจาก R-Tree index quality degradation number แล้วยั้งต้องดูที่ปัจจัยquery performanceอื่นในภาพรวมอื่นด้วย เช่น Oracle caching strategies, Table piningซึ่งเป็นปัจจัยที่สำคัญในการremove I/O overhead จากR-Tree index queries เป็นต้น

Spatial Relationships และ Filtering

Spatial จะใช้ secondary filter ในการพิจารณาSpatial Relationship ระหว่างentitiesในฐานข้อมูล Spatial Relationship นั้นเป็นพื้นฐานของgeometry location ซึ่ง spatial relationships โดยทั่วไปจะอยู่บนพื้นฐานของ topology และ distance ยกตัวอย่าง Boundary ของพื้นที่ประกอบขึ้นมาจาก setของcurvesซึ่งแบ่งพื้นที่ออกจากพื้นที่อื่นๆในcoordinate space Interior ของพื้นที่ประกอบขึ้นมาจากทุกๆจุดในพื้นที่นั้นซึ่งไม่ใช่อยู่บนboundaryของgeometryนั้นๆยกตัวอย่างเช่น ให้พื้นที่มา 2 พื้นที่จะกล่าวได้ว่า adjacentกัน ได้ก็ต่อเมื่อ ทั้ง 2 พื้นที่ share part ของ boundary ร่วมกัน แต่ไม่ได้share pointsของinterior กันนั่นเองDistance ระหว่าง 2 spatial object คือระยะทางระหว่างจุด2จุดที่น้อยที่สุด ในการพิจารณาSpatial Relationships นั้นSpatial จะมีSecondary filter methods หลาย method SDO_RELATE operator ใช้ในการหาค่า topological criteria

SDO_WITHIN_DISTANCE operator ใช้ในการพิจารณา เช่น ถ้ามี 2 spatial object และ มี spatial object หนึ่งอยู่ใน specified distance ของอีกอันหนึ่ง

SDO_NN operator ใช้ในการระบุ nearest neighbors สำหรับ Spatial Object

ชนิดข้อมูลแบบ SDO_GOMETRY

ใน Spatialนั้น การอธิบายgeometricของSpatial object จะเก็บอยู่ใน 1 row ใน1 column ของ object type ที่ชื่อว่า SDO_GEOMETRY ซึ่งจะอยู่ในตารางที่เราเองเป็นคนสร้างขึ้น(user-defined table) ในตารางจะมีcolumnของข้อมูลประเภท SDO_GEOMETRY ซึ่งจะต้องมีcolumnอื่นด้วย หรือ set ของ columns ที่จะต้องมีการประกาศ unique primary key สำหรับตารางนั้นๆ

ORACEL Spatial ประกาศobject type SDO_GEOMETRY ดังนี้ :

```
CREATE TYPE  sdo_geometry  AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY
);
```

ORACEL Spatial ประกาศ SDO_POINT_TYPE, SDO_ELEM_INFO, และ SDO_ORDINATE_ARRAY typeโดยที่เหล่านี้จะใช้ใน SDO_GEOMETRY type ซึ่งประกาศดังต่อไปนี้

```
CREATE TYPE   sdo_point_type          AS OBJECT (
```

```
X NUMBER,
                                        Y NUMBER,
                                        Z NUMBER  );
CREATE TYPE  sdo_elem_info_array  AS VARRAY (1048576) of  NUMBER;
CREATE TYPE  sdo_ordinate_array   AS VARRAY (1048576) of  NUMBER;
```

เนื่องจากว่า SDO_ORDINATE_ARRAY มีขนาดมากที่สุดได้ 1,048,576 numbers โดยมีค่ามากที่สุดในSDO_GEOMETRY object ซึ่งขึ้นอยู่กับจำนวนของ dimensions โดยสำหรับ two dimensions จะมีได้ 524,288 หน่วย , สำหรับ three dimensions จะมีได้ 349,525 หน่วย ,และสำหรับ four dimensions จะมีได้ 262,144 หน่วย

3.4 ใช้ ORACLE 10g enterpriseในการ สนับสนุน Spatio Temporal Database

การ Implement Spatio Database โดยใช้ ORACLE 10g enterprise ใช้ความสามารถร่วมกันระหว่าง Oracle spatial และ Workspace Manager ซึ่งจะทำให้ Oracle Spatial Database สามารถเก็บข้อมูลอ้างอิงกับ Valid Time ได้ โดยมีขั้นตอนดังนี้

1. สร้างตารางที่เก็บ ข้อมูล Spatial โดยกำหนดให้คอลัมน์ ที่ใช้เก็บข้อมูลเชิงภูมิศาตร์มีชนิดข้อมูลเป็น SDO_GEOMETRY
2. กำหนด Primary Key ให้กับตารางที่เราสร้างขึ้นมา โดยเราจะไม่สามารถกำหนดคอลัมน์ที่เราใช้เก็บข้อมูลเชิงภูมิศาตร์ให้เป็น Primary Key ได้
3. ทำการ Enable Versioning ซึ่งวิธีการได้กล่าวไปแล้วใน ส่วนที่ 3.1
4. หลังจากขั้นตอนนี้ไปแล้วตารางที่เราสร้างขึ้นก็สามารถเก็บข้อมูลเป็น Spatio-Temporal Database แล้ว
5. Updates USER_SDO_GEOM_METADATA ซึ่งเป็น view เพื่อส่งผลไปยังข้อมูลในdimensional สำหรับพื้นที่ โดยจะต้องเปลี่ยนชื่อตารางเป็น “ชื่อตาราง_LT”
6. หากผู้ใช้ต้องการสร้าง Spatial Index ขอให้ทำหลังจาก Insert ข้อมูลไปแล้ว เนื่องจาก Spatial Index นั้นจำเป็นจะต้องมีข้อมูลเชิงภูมิศาสตร์ก่อนเพื่อใช้ในการกำหนด Index ซึ่งอธิบายในส่วนที่ 3.1 โดยจะต้องเปลี่ยนชื่อตารางเป็น “ชื่อตาราง_LT”

1. สร้างตาราง ชื่อ *employees(EMPID,NAME,EARNINGS,LOCATION)* และ *buildings(ENO,BNAME,GEOM)* เพื่อเก็บข้อมูล Spatial โดยกำหนดให้คอลัมน์ที่ใช้เก็บข้อมูลเชิงภูมิศาตร์ชื่อ LOCATION โดยกำหนดให้มีชนิดข้อมูลเป็น SDO_GEOMETRY และ CUS_ID เป็น Primary Key

```
CREATE TABLE DPEMPLOYEES (
        "EMPID" VARCHAR2(20 BYTE),
        "NAME" VARCHAR2(64 CHAR),
        "SALARY" NUMBER,
        "LOCATION" SDO_GEOMETRY ,
        PRIMARY KEY ("EMPID")) ;
```

```
CREATE TABLE BUILDINGS (
        "BNO" VARCHAR2(20 BYTE),
        "BNAME" VARCHAR2(64 CHAR),
        "GEOM"  SDO_GEOMETRY ,
        PRIMARY KEY ("BNO")) ;
```

2. ทำการ Enable Versioning ให้กับตาราง employees

```
EXECUTE DBMS_WM.ENABLEVERSIONING('EMPLOYEES', 'VIEW_WO_OVERWRITE',
FALSE,TRUE);
```

3. Insert ข้อมูลลงในตารางที่สร้างขึ้น

```
-- Set Session Time t1 2005-2006
EXECUTE DBMS_WM.SETVALIDTIME(TO_TIMESTAMP('01-01-2005','DD-MM-
YYYY'),TO_TIMESTAMP('01-01-2006','DD-MM-YYYY'));

INSERT INTO CUSTOMERS VALUES('0001' ,'EMP1' ,150000,
SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(3, 4, NULL),NULL,NULL),
NULL   -- ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session
```

);

INSERT INTO CUSTOMERS VALUES('0002' ,'EMP2' ,180000,

SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(4, 5, NULL),NULL,NULL),

NULL – ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session

);

INSERT INTO CUSTOMERS VALUES('0003' ,'EMP3' ,195000,

SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(6, 7, NULL),NULL,NULL),

NULL – ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session

);

– Set Session Time t2 2006-2007

EXECUTE DBMS_WM.SETVALIDTIME(TO_TIMESTAMP('01-01-2006','DD-MM-YYYY'),TO_TIMESTAMP('01-01-2007','DD-MM-YYYY'));

INSERT INTO CUSTOMERS VALUES('0001' ,'EMP1' ,170000,

SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(4, 4, NULL),NULL,NULL),

NULL – ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session

);

INSERT INTO CUSTOMERS VALUES('0002' ,'EMP2' ,185000,

SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(4, 5, NULL),NULL,NULL),

NULL – ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session

);

INSERT INTO CUSTOMERS VALUES('0003' ,'EMP3' ,210000,

SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(6, 5, NULL),NULL,NULL),

NULL – ใส่ค่า ValidTime เป็น Null เพื่อใช้ค่าของเวลาของ Session

);

4. Updates USER_SDO_GEOM_METADATA ซึ่งเป็น view เพื่อส่งผลไปยังข้อมูลในdimensional สำหรับพื้นที่ โดยจะต้องเปลี่ยนชื่อตารางเป็น “ชื่อตาราง_LT”

```
INSERT INTO user_sdo_geom_metadata
(TABLE_NAME,COLUMN_NAME,DIMINFO,SRID)
VALUES ('EMPLOYEES_LT','LOCATION',
SDO_DIM_ARRAY( -- 20X20 grid
SDO_DIM_ELEMENT('X', 0, 10, 0.005),
SDO_DIM_ELEMENT('Y', 0, 10, 0.005)),
NULL – SRID
);
```

5. สร้าง Spatial Index โดยจะต้องเปลี่ยนชื่อตารางเป็น “ชื่อตาราง_LT” โดยจะต้องเปลี่ยนชื่อตารางเป็น “ชื่อตาราง_LT”

```
CREATE INDEX SIDX_EMPLOYEES
ON EMPLOYEES_LT(LOCATION)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

บทที่ 4
การทดลองและวิเคราะห์ผล

4.1 ขั้นตอนการทดลอง

4.1.1 วางแผนในการทดลองโดยมีวัตถุประสงค์เพื่อ

4.1.1.1 ทดสอบความสามารถในการสนับสนุน Valid Time

4.1.1.2 ทดสอบความสามารถในการสนับสนุน Transaction Time

4.1.1.3 ทดสอบความสามารถในการสนับสนุน Spatio-temporal

4.1.2 จัดเตรียมสภาพแวดล้อมสำหรับการทดลอง คือ

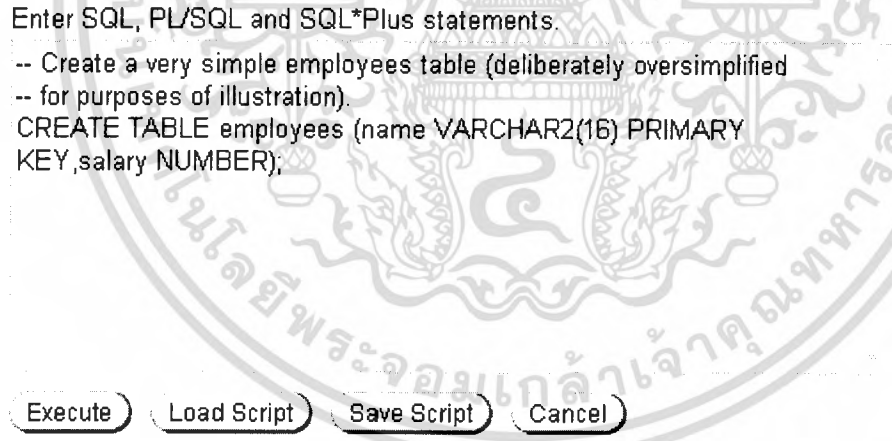
4.1.2.1 Install DBMS Oracle 10g บน Windows XP Service pack 2

4.1.2.2 เตรียมโปรแกรม SpatioTemporalPlus ซึ่งเป็นโปรแกรมที่ได้พัฒนาขึ้นมาเพื่อทำการทดลองนี้

4.2 การทดลองใช้โปรแกรม SQL/PLUS ทดสอบความสามารถในการสนับสนุน Valid Time

ตัวอย่างง่ายๆ สำหรับการสร้างตารางให้สนับสนุนน Valid Time โดยมีขั้นตอนดังนี้

4.2.1 สร้างตาราง โดยตัวตารางนั้นจะต้องกำหนด คีย์หลัก (Primary key) ไว้ด้วย ยกตัวอย่างเช่นตารางที่ใช้เก็บข้อมูลขอชื่องพนักงานและเงินเดือนของแต่ละคน โดยมีตัวอย่างคำสั่งและผลดังรูป ดังนี้



รูปที่ 4.1 แสดงการสร้างตารางเพื่อใช้เก็บข้อมูล

4.2.2 ทำ Version-enables (โดยมีรูปแบบของคำสั่งดังตัวอย่าง) ให้ตารางสามารถสนับสนุน Valid Time โดยจะเพิ่ม column ชื่อว่า WM_VALID ให้โดยอัตโนมัติ โดย คอลัมน์นี้จะมีชนิดข้อมูล WM_PERIOD ซึ่ง รายละเอียดจะได้อธิบายในหัวข้อถัดไป รูปแบบของคำสั่งที่จะทำ Version-enables จะแสดงดังรูป

Enter SQL, PL/SQL and SQL*Plus statements.

```
EXECUTE DBMS_WM.EnableVersioning
('EXPEMPLOYEES', 'VIEW_WO_OVERWRITE', FALSE,TRUE);
```

Execute Load Script Save Script Cancel

โปรซีเจอร์ PL/SQL เสร็จสมบูรณ์

รูปที่ 4.2 แสดงการ EnableVersioning

4.2.3 Insert ข้อมูลลงในตาราง โดยต้องกำหนดชื่อ เงินเดือน และ ช่วงเวลาของข้อมูล

```
INSERT INTO EXPEMPLOYEES
VALUES('Adams',30000,WMSYS.WM_PERIOD(TO_DATE('01-01-1990', 'MM-DD-
YYYY'),TO_DATE('01-01-2005', 'MM-DD-YYYY')));

INSERT INTO EXPEMPLOYEES
VALUES('Baxter',40000,WMSYS.WM_PERIOD(TO_DATE('01-01-2000', 'MM-DD-
YYYY'), DBMS_WM.UNTIL_CHANGED));

INSERT INTO EXPEMPLOYEES
VALUES('Coleman',50000,WMSYS.WM_PERIOD(TO_DATE('01-01-2003', 'MM-DD-
YYYY'),TO_DATE('12-31-9999', 'MM-DD-YYYY')));
COMMIT;
-- Set valid time period to virtually all time.
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-1900', 'MM-DD-
YYYY'),TO_DATE('01-01-9999', 'MM-DD-YYYY'));
```

ผล

สร้างแถวแล้ว 1 แถว

สร้างแถวแล้ว 1 แถว

สร้างแถวแล้ว 1 แถว

การคอมมิตเสร็จสมบูรณ์

รูปที่ 4.3 แสดงการผลลัพธ์จากการ Insert ข้อมูล

4.2.4 กำหนดช่วงเวลาให้กับฐานข้อมูลว่าจะให้แสดงข้อมูลในช่วงเวลาไหน

```
-- Operators for Workspace Manager Valid Time Support
-- WM_CONTAINS
SELECT * FROM EXPEMPLOYEES e
WHERE WM_CONTAINS(e.wm_valid,wm_period(TO_DATE('01-01-1995', 'MM-DD-
YYYY'),TO_DATE('01-02-1995', 'MM-DD-YYYY'))) = 1;
```

NAME	SALARY	WM_VALID(VALIDFROM, VALIDTILL)
Adams	30000	WM_PERIOD('01 Á.¤. 1990 00.00.00.000000 +07:00', '01 Á.¤. 2005 00.00.00.000000 +07:00')

รูปที่ 4.4 แสดงการผลลัพธ์จากการใช้ WM_CONTAINS

```
-- WM_OVERLAPS
SELECT * FROM EXPEMPLOYEES e
WHERE WM_OVERLAPS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2000', 'MM-DD-YYYY'))) = 1;
```

NAME	SALARY	WM_VALID(VALIDFROM, VALIDTILL)
Adams	30000	WM_PERIOD('01 Á.¤. 1990 00.00.00.000000 +07:00', '01 Á.¤. 2005 00.00.00.000000 +07:00')

รูปที่ 4.5 แสดงการผลลัพธ์จากการใช้ WM_OVERLAPS

```
-- WM_MEETS
SELECT * FROM EXPEMPLOYEES e
WHERE WM_MEETS(e.wm_valid,
wm_period(TO_DATE('01-01-2005', 'MM-DD-YYYY'),
TO_DATE('01-01-2006', 'MM-DD-YYYY'))) = 1;
```

NAME	SALARY	WM_VALID(VALIDFROM, VALIDTILL)
Adams	30000	WM_PERIOD('01 Á.¤. 1990 00.00.00.000000 +07:00', '01 Á.¤. 2005 00.00.00.000000 +07:00')

รูปที่ 4.6 แสดงการผลลัพธ์จากการใช้ WM_MEETS

```
-- WM_EQUALS
SELECT * FROM EXPEMPLOYEES e
WHERE WM_EQUALS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2005', 'MM-DD-YYYY'))) = 1;
```

NAME	SALARY	WM_VALID(VALIDFROM, VALIDTILL)
Adams	30000	WM_PERIOD('01 Á¤. 1990 00.00.00.000000 +07:00', '01 Á¤. 2005 00.00.00.000000 +07:00')

รูปที่ 4.7 แสดงการผลลัพธ์จากการใช้ WM_EQUALS

```
-- WM_LESSTHAN
SELECT * FROM EXPEMPLOYEES e
WHERE WM_LESSTHAN(e.wm_valid,
wm_period(TO_DATE('01-01-2010', 'MM-DD-YYYY'),
TO_DATE('01-02-2010', 'MM-DD-YYYY'))) = 1;
```

NAME	SALARY	WM_VALID(VALIDFROM, VALIDTILL)
Adams	30000	WM_PERIOD('01 Á¤. 1990 00.00.00.000000 +07:00', '01 Á¤. 2005 00.00.00.000000 +07:00')

รูปที่ 4.8 แสดงการผลลัพธ์จากการใช้ WM_LESSTHAN

```
-- WM_INTERSECTION
SELECT e.name, WM_INTERSECTION(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM EXPEMPLOYEES e;
```

NAME	WM_INTERSECTION(E.WM_VALID,WM_PERIOD(TO_DATE('01-01-1995','MM-DD-YYYY'),TO_DATE('01-02-1995','MM-DD-YYYY'))) (VALIDFROM, VALIDTILL)
Adams	WM_PERIOD('01 Á¤. 1995 00.00.00.000000 +07:00', '02 Á¤. 1995 00.00.00.000000 +07:00')
Baxter	
Coleman	

รูปที่ 4.9 แสดงการผลลัพธ์จากการใช้ WM_INTERSECTION

```
-- WM_LDIFF
SELECT e.name, WM_LDIFF(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM EXPEMPLOYEES e;
```

NAME	WM_LDIFF(E.WM_VALID,WM_PERIOD(TO_DATE('01-01-1995','MM-DD-YYYY'),TO_DATE('01-02-1995','MM-DD-YYYY')))(VALIDFROM, VALIDTILL)
Adams	WM_PERIOD('01 Á¤. 1990 00.00.00.000000 +07:00', '01 Á¤. 1995 00.00.00.000000 +07:00')
Baxter	
Coleman	

รูปที่ 4.10 แสดงการผลลัพธ์จากการใช้ WM_LDIFF

```
-- WM_RDIFF
SELECT e.name, WM_RDIFF(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM EXPEMPLOYEES e;
```

NAME	WM_RDIFF(E.WM_VALID,WM_PERIOD(TO_DATE('01-01-1995','MM-DD-YYYY'),TO_DATE('01-02-1995','MM-DD-YYYY')))(VALIDFROM, VALIDTILL)
Adams	WM_PERIOD('02 Á¤. 1995 00.00.00.000000 +07:00', '01 Á¤. 2005 00.00.00.000000 +07:00')
Baxter	
Coleman	WM_PERIOD('01 Á¤. 2003 00.00.00.000000 +07:00', '31 ¿¬. 9999 00.00.00.000000 +07:00')

รูปที่ 4.11 แสดงการผลลัพธ์จากการใช้ WM_RDIFF

4.2.5 แก้ไขข้อมูลเงินเดือนและช่วงเวลาของข้อมูล

```
-- Update the salary for an existing employee. Perform "sequenced" update, so
-- that existing time-related information is preserved. This results in two rows
-- for Baxter.
-- First, set valid time to the intended range for Baxter's raise.
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-2003', 'MM-DD-
YYYY'),DBMS_WM.UNTIL_CHANGED);

-- Give Baxter a raise, effective 01-Jan-2003 until changed.
UPDATE EXPEMPLOYEES SET salary = 45000 WHERE name = 'Baxter';
COMMIT;
```

โปรซีเจอร์ PL/SQL เสร็จสมบูรณ์

อัปเดตแถวแล้ว 1 แถว

การคอมมิตเสร็จสมบูรณ์

รูปที่ 4.12 แสดงการผลลัพธ์จากการ แก้ไขข้อมูลเงินเดือนและช่วงเวลาของข้อมูล

4.2.6 ทำ Disables versioning (โดยมีรูปแบบของคำสั่งดังตัวอย่าง) ให้กับตาราง ในขั้นตอนนี้เราจะทำหรือไม่ทำก็ได้ หากไม่ทำการ Disables versioning ผู้ใช้จะไม่สามารถ ลบตาราง นี้ได้

```
-- Disable versioning. By default (keepWMValid parameter value of TRUE),
-- the WM_VALID column is kept, with all its data.
EXECUTE DBMS_WM.DisableVersioning ('EXPEMPLOYEES');
```

โปรซีเจอร์ PL/SQL เสร็จสมบูรณ์

4.3 การทดลองการใช้งาน Flashback

4.3.1 Flashback Query (SELECT .... AS OF)

เราปฏิบัติ Flashback Query ได้โดยใช้คำสั่ง SELECT โดยมี AS OF เป็นclause โดยเราใช้ Flashback Query ในการค้นคืนข้อมูลจากอดีต โดยคำสั่งที่ใช้ในการquery จะแสดงออกมาอย่างชัดเจนว่า อ้างอิงถึงช่วงเวลาใดในอดีต ซึ่งก็คือ timestamp หรือ SCN นั่นเอง ผลจากการใช้คำสั่งนี้จะreturn committed data ของช่วงเวลานั้นออกมา

ความสามารถในการใช้งาน **Flashback Query**

- กู้คืนข้อมูลที่หายไป หรือ เกิดจากการกระทำที่ไม่ถูกต้อง โดยข้อมูลนั้นได้มีการcommited ไปแล้ว เช่น กระทำการdelete หรือ update tupleและcommit ไปแล้วแต่ตอนหลังมาพบว่าเป็นการกระทำที่ผิดพลาด สามารถแก้ไขได้undo ข้อมูลที่ได้ทำผิดพลาดไปได้โดยทันที
- เปรียบเทียบข้อมูลปัจจุบันกับข้อมูลในอดีต เช่น ทำรายงานเปรียบเทียบข้อมูลในแต่ละวัน

- เช็คข้อมูลของTransaction ในแต่ละเวลา เช่น ตรวจสอบaccount balance ในแต่ละวันเพื่อให้แน่ใจ
- ช่วยให้การออกแบบDesignระบบหรือapplication ทำได้ง่านขึ้นในส่วนของ Temporal data โดยมีFlashback Query มาช่วยจัดการแทน ทำให้เราสามารถ retrieve ข้อมูลในอดีตได้โดยตรงจากฐานข้อมูล

ตัวอย่างนี้ใช้ Flashback Query ในการตรวจพิจรณาสถานะของtableในอดีต สมุตติ เวลาเที่ยงคืน 12.30 PM DBAพบว่า row ของ สมชาย บุญประกอบ ได้ถูกdeleteออกไปจาก table employee

DBAทราบว่าตอนเก้าโมงเช้า 9.30 AM. ข้อมูลของ สมชาย ยังถูกตรวจ-แก้ไขอยู่ใน database

ดังนั้น DBA จึงสามารถใช้ Flashback Query ในการตรวจพิจรณา เนื้อหาของ Table ตั้งแต่ตอน เก้าโมงเช้า 9.30 AM. เป็นต้นมาว่า data row นี้หายไปตอนไหน

ตัวอย่าง การค้นคืน row ของ สมชาย บุญประกอบ จาก Flashback Query ในเวลา 9:30AM, April 4, 2004:

```
SELECT   *
FROM     employees
AS OF    TIMESTAMP
TO_TIMESTAMP ('2004-04-04  09:30:00', 'YYYY-MM-DD HH:MI:SS')
WHERE    last_name = 'บุญประกอบ';
```

ถ้าข้อมูลของ สมชาย บุญประกอบ นั้นถูกลบไปโดยไม่เหมาะสม DBAก็สามารถ re-insert ข้อมูล rowนี้ลงไปใน table employees

```
INSERT INTO employees
  (
    SELECT  *
    FROM    employees
    AS OF   TIMESTAMP
```

```
    TO_TIMESTAMP ('2004-04-04  09:30:00', 'YYYY-MM-DD HH:MI:SS')
    WHERE     last_name = 'บุญประกอบ'
);
```

Tips for Using Flashback Query

- เราสามารถระบุหรือละเว้นการใส่ AS OF clause ในแต่ละtableได้และระบุเวลาสำหรับtableที่ต่างกัน เราใช้ AS OF clause ในการquery เพื่อปฏิบัติ DDL (data definition language operation เช่น การ create) หรือ DML (data manipulate language เช่น การ insert delete)

- ในการใช้งาน ผลลัพธ์จาก การทำ Flashback Query โดย คำสั่งชนิดDDL หรือ DML จะส่งผลกระทบกับสถานะบัจจุบันของdatabase โดยใช้ AS OF เป็นclause ข้างในคำสั่ง INSERT หรือ CREATE TABLE AS SELECT

- เวลาที่เราจะลือกใช้ timestamp หรือ SCN ในการทำ Flashback Query นั้น Oracle database ใช้ SCN ภายในและ ทำการmaps ไปยัง timestamp

- เราสามารถสร้าง view และอ้างอิงถึงข้อมูลในอดีตโดยใช้คำสั่ง AS OF เป็นclauseในคำสั่ง SELECT ซึ่งจะนำมาเป็นนิยามให้กับ view

```
CREATE VIEW hour_ago AS
SELECT *
FROM employees AS OF
TIMESTAMP (SYSTIMESTAMP - INTERVAL '60' MINUTE);
```

SYSTIMESTAMP อ้างถึง time zone ของ database host environment

- เราสามารถใช้ AS OF เป็น clauseในการทำ self-join หรือ ทำคำสั่งใน set operation เช่น INTERSECT ,MINUS และสามารถเก็บผลลัพธ์ที่ได้จากกระบวนการทำ Flashback Query ด้วยคำ CREATE TABLE AS SELECT หรือ INSERT INTO TABLE SELECT ดังตัวอย่างนี้จะทำการ reinsert ลงไปใน table employees ทำให้ rows นี้อยู่ในตาราง employees เมื่อชั่วโมงที่แล้ว

```
INSERT INTO employees
( SELECT *
```

```
FROM employees AS OF
TIMESTAMP (SYSTIMESTAMP - INTERVAL '60' MINUTE) )

-- SYSTIMESTAMP อ้างถึง time zone ของ database host environment MINUS SELECT *
FROM employees);
```

4.3.2 การใช้งาน DBMS_FLASHBACK package

โดยทั่วไปแล้ว DBMS_FLASHBACK package จะมีหน้าที่การทำงานคล้ายกับ Flashback Query แต่ การใช้งาน Flashback Query นั้นจะสามารถใช้งานได้สะดวกกว่าในบางเวลา

DBMS_FLASHBACK package จะทำงานคล้ายกับ Time machine คือ เราสามารถย้อนเวลากลับไปได้ โดยเราใช้คำสั้งquery ธรรมดาเหมือนเรากำลังกระทำการค้นคืนกับข้อมูลในอดีต

สาเหตุที่เราทำเช่นนี้ได้ เพราะว่าคุณสามารถใช้ DBMS_FLASHBACK package ในการปฏิบัติคำสั่งquery บนข้อมูลในอดีตได้โดย ไม่ต้องเพิ่มเติมคำสั่งพิเศษลงไป (clause เช่น AS OF หรือ VERSION BETWEEN) เราสามารถใช้คำสั่งใน PL/SQL ในการ query databaseในอดีตได้เลย

ในการใช้เราต้อง Set ให้สามารถใช้งาน DBMS_FLASHBACK package ได้ด้วย วิธีการใช้งาน DBMS_FLASHBACK package สำหรับทำคำสั่ง PL/SQL ได้ มีดังนี้

1. ใช้คำสั่ง **DBMS_FLASHBACK.**ENABLE_AT_SYSTEM_CHANGE_NUMBER เพื่อย้อนเวลากลับไปในเวลาที่เราระบุในอดีต หลังจากนั้นทุกๆคำสั่ง query ที่เรากระทำจะเป็นการปฏิบัติคำสั่งกับข้อมูลที่อยู่ในเวลาขณะนั้นๆ ของอดีตที่เราระบุไป

2. สร้างคำสั่งในการ query ธรรมดาโดยไม่ต้องใช้ special flashback-feature syntax เช่น as of นะ Databaseจะทำการqueryข้อมูลมาให้ตามเวลาในอดีตที่เราระบุให้เลย ที่สำคัญคือ เราจะทำได้เพียงแค่การ query ข้อมูลเท่านั้นนะ ไม่สามารถจะทำกับคำสั่งประเภท DDL (data definition language เช่น CREATE) หรือ DML (Data manipulate language เช่น update insert delete) ได้นะ

3. ใช้คำสั่ง DBMS_FLASHBACK.DISABLE

ในการที่เราจะกลับมาสู่บัจจุบันได้นั้นต้องใช้ไปเรียก DISABLE โดยหลังจากนั้นเราอาจจะเรียก ENABLE เวลาอื่นๆ อีกก็ได้ แต่ไม่สามารถจะทำการซ้อน(nested) คำสั่ง ENABLE/ DISABLE เป็นคู่ได้นะ

เราก็สามารถใช้งาน cursor เพื่อเก็บผลลัพธ์จากการทำ query ที่ผ่านมาได้โดยต้องเปิด cursor ก่อนคำสั่ง DISABLE และหลังจาก store ข้อมูลแล้ว เราก็ค่อยสั่ง DISABLE เมื่อเก็บผลลัพธ์แล้วและ DISABLE แล้ว เราสามารถใช้ความสามารถนี้เพื่อทำ

- ปฏิบัติคำสั่ง INSERT หรือ UPDATE เพื่อแก้ไขข้อมูลบัจจุบันที่ได้มาจากการเก็บผลลัพธ์
- เปรียบเทียบข้อมูลในบัจจุบันกับในอดีต โดย หลังจาก DISABLE ก็ให้เปิด cursor ขึ้นมาอีกอันนึง โดย cursor ตัวแรก retrieve ข้อมูลจากอดีตขึ้นมาแล้ว และ cursor ที่เปิดขึ้นมาตัวที่สองนั่นให้ retrieve ข้อมูลบัจจุบันขึ้นมา โดยสามารถเก็บข้อมูลในอดีตไว้ในอีกตารางนึงชั่วคราวไว้ก่อนได้ และ ใช้ set operation เช่น MINUS, UNION เพื่อเปรียบเทียบความแตกต่างหรือรวมกันระหว่างข้อมูลในอดีตกับข้อมูลในบัจจุบันได้

เราสามารถใช้คำสั่ง **DBMS_FLASHBACK.**GET_SYSTEM_CHANGE_NUMBER ได้เมื่อต้องการ SCN (System Change Number) บัจจุบัน สังเกตว่า จะคืนค่า SCN บัจจุบันกลับมา; this takes no account of previous calls to DBMS_FLASHBACK.ENABLE*.

4.3.3 การใช้งาน ORA_ROWSCN ORA_ROWSCN คือ เป็นpseudocolumnColumซึ่งสร้างขึ้นเพื่อเก็บ SCN ยกตัวอย่าง เช่น

```
SELECT  ora_rowscn, last_name, salary
FROM    employees
WHERE   employee_id = 7788;

ORA_ROWSCN   NAME   SALARY
----------           ----       ------
202553               Fudd       3000
```

คำสั่ง COMMIT สุดท้ายของ row นี้จะแปะ SCN 202553 ลงไป โดยเราสามารถใช้ฟังชั่นก์
SCN_TO_TIMESTAMP เพื่อแปลงจาก SCN(ORA_ROWSCN)ไปเป็นค่า TIMESTAMP ที่ตรงกัน

ORA_SCN คือ ค่าเวลาที่ commit ล่าสุด โดยเป็นค่าที่ค่อนข้างแม่นยำ สำหรับ row-dependent table(สามารถสร้างได้โดย ใช้คำสั่ง CREATE TABLE โดยใช้ส่วน clause เป็น ROWDEPENDENCIES)

สิ่งที่ต้องคำนึงถึงในการใช้งาน ORA_ROWSCN กับ การพัฒนา application นั้นจะต้องมี concurrency control และ client cache invalidation ดังตัวอย่าง

ถ้า application เราจะตรวจวิเคราะห์ ข้อมูลในrow และ record ที่ตรงกับค่า ORA_ROWSCN = 202553 โดยหลังจากนั้นต้องการupdate row นั้น แต่ record ของ data นั้นจะต้องยังคงถูกต้องอยู่ เป็นการปฏิบัติคำสั่ง update เฉพาะบางส่วนโดยที่rowต้องไม่เปลี่ยน ดังนั้น เราจึงต้องให้สร้างเงื่อนไข โดยระบุให้ ORA_ROWSCN = 202553

```
UPDATE employees
SET    salary = salary + 100
WHERE  employee_id = 7788
AND    ora_rowscn = 202553;
```

0 rows updated.

จะเห็นการใช้เงื่อนไขดังกล่าว ได้ผลการ update fails เพราะ ORA_ROWSCN นั้นตอนนี้ไม่ได้เป็น 202553 แล้ว นั่นหมายความว่า มีบางคนหรือบางโปรแกรม ได้เปลี่ยนแปลง row หรือ ปฏิบัติคำสั่งที่ได้ COMMIT ไปแล้ว ล่าสุดเมื่อเร็วๆนี้

เราจึงทดลองอีกครั้งโดยqueriesข้อมูลในrowและORA_SCNใหม่ ซึ่งสมมุติได้ค่า ORA_ROWSCN ในขณะนี้เป็น 415639 และทำการพยายาม updateอีกทีนึงโดยใช้ค่า ORA_ROWSCNใหม่ดังข้างล่าง

```
SQL> UPDATE  employees
    SET     salary = salary + 100
    WHERE   empno  = 7788
    AND     ora_rowscn = 415639;
1 row updated.

SQL> COMMIT;
Commit complete.
```

ซึ่งผลการupdate succeeds และ committed เรียบร้อยดี เราทดลอง query ดูใหม่

```
SQL> SELECT  ora_rowscn, name, salary
    FROM    employees
    WHERE   empno = 7788;

ORA_ROWSCN   NAME   SALARY
----------   ----   ------
465461       Fudd   3100
```

ปรากฏว่า ค่า ORA_ROWSCN เปลี่ยนไปเป็น 465461 ซึ่งก็คือ ค่า SCN ของการ COMMIT ครั้งใหม่เป็น 465461

นอกจากเราจะสามารถใช้ ORA_ROWSCN กับการทำคำสั่ง UPDATE โดยใส่ในส่วน clauseของ WHERE แล้ว ยังสามารถใช้กับ การทำคำสั่ง DELETE โดยใส่ในส่วนclauseของ WHERE หรือ ในส่วนclauseของ AS OF ของการทำ Flashback Query

4.3.4 การใช้งาน Flashback Version Query

เราสามารถใช้งาน Flashback Version Query ในการ retrieve version ต่างๆ ของ rows ที่ระบุและอยู่ในช่วงเวลาที่กำหนด

row version ใหม่จะเกิดเมื่อใดก็ตาม ที่ คำสั่งที่ปฏิบัตินั้นได้รับการ COMMIT

การระบุว่าเราจะทำ FLASH_BACK_QUERY นั้นสามารถระบุได้โดยใช้ คำสั่ง VERSIONS_BETWEEN เป็นส่วน clause ของคำสั่ง SELECT

VERSIONS { BETWEEN {SCN | TIMESTAMP} start AND end }

ตำแหน่ง start และ end นั้น จะเป็นช่วงเวลาเริ่มต้นและสิ้นสุด ที่จะ query

การทำ Flashback Version Query จะ คืนค่ากลับออกมาเป็นtableของ row แต่ละversion ของ row นั้นๆ ที่อยู่ในช่วงเวลาดังกล่าวที่เราระบุ

โดยในแต่ละ rowจะประกอบไปด้วยสิ่งที่เพิ่มเติมเข้ามาคือ pseudoColumns ของ metadata เกี่ยวกับ row version นั้นๆ ซึ่ง metadata จะประกอบไปด้วยข้อมูลต่างๆ ดังตารางด้านล่าง โดยข้อมูลนั้นจะแสดงให้เห็นถึงเวลา และ สิ่งที่เราทำการเปลี่ยนแปลงกับdatabase

Pseudocolumn Name	Description
VERSIONS_STARTSCN VERSIONS_STARTTIME	Starting System Change Number (SCN) or TIMESTAMP when the row version was created. This identifies the time when the data first took on the values reflected in the row version. You can use this to identify the past target time for a Flashback Table or Flashback Query operation. If this is NULL, then the row version was created before the lower time bound of the query BETWEEN clause.
VERSIONS_ENDSCN VERSIONS_ENDTIME	SCN or TIMESTAMP when the row version expired. This identifies the row expiration time. If this is NULL, then either the row version was still current at the time of the query or the row corresponds to a DELETE operation.
VERSIONS_XID	Identifier of the transaction that created the row version.
VERSIONS_OPERATION	Operation performed by the transaction: I for insertion, D for deletion, or U for update. The version is that of the row that was inserted, deleted, or updated; that is, the row *after* an INSERT operation, the row *before* a DELETE operation, or the row affected by an UPDATE operation. *Note:* For user updates of an **index key**, a Flashback Version Query may treat an UPDATE operation as two operations, DELETE plus INSERT, represented as two version rows with a D followed by an I VERSIONS_OPERATION.

รูปที่ 4.13 PseudoColumns ของ Metadata

ยกตัวอย่าง เช่น

VERSIONS_START_TIME	VERSIONS_END_TIME	SALARY
09-SEP-2002	25-NOV-2003	10243

ข้อมูลที่แสดงให้เห็นว่า SALARY เคยเป็น 10243 เมื่อวันที่09-SEP-2002 จนถึงวันก่อนวันที่ 25-NOV-2003(ไม่รวมวันที่ 25)

นี่คือตัวอย่างการใช้คำสั่งทำ Flashback Version Query

```
SELECT   versions_startscn,
      versions_starttime,
      versions_endscn,
      versions_endtime,
      versions_xid,
      versions_operation,
      name,
      salary

FROM employees

VERSIONS BETWEEN TIMESTAMP
      TO_TIMESTAMP('2003-07-18 14:00:00', 'YYYY-MM-DD HH24:MI:SS')
      AND
      TO_TIMESTAMP('2003-07-18 17:00:00', 'YYYY-MM-DD HH24:MI:SS')

WHERE name = 'JOE';
```

Pseudocolumn versions_xid นั้นจะ uniqueสำหรับแต่ละTransaction ซึ่งจะใช้ระบุถึง Transaction ที่ใส่ข้อมูลลงในstateนั้นๆ เราสามารถใช้ค่านี้ในการ ติดต่อกับ Flashback Transaction Query เพื่อหา metadataของ transactionใน FLASHBACK_TRANSACTION_QUERY view ซึ่งจะใช้ในกรณีที่ SQL ต้องการ undo ค่าที่เปลี่ยนแปลงของ row นั้นๆ และต้องการทราบว่าผู้ใช้คนไหนเป็นคนทำการเปลี่ยนแปลง row นั้นๆ

4.3.6 การใช้งาน Flashback Transaction Query

Flashback Transaction Query นั้นเป็นการ query จากบนview Flashback Transaction Query

เราใช้งาน Flashback Transaction Query เพื่อให้ได้มาซึ่งข้อมูลของ transactionรวมถึง SQL code ที่เราสามารถทำundoการเปลี่ยนแปลงบางอย่างที่ได้กระทำไปแล้วโดยTransactionนั้นๆ

ยกตัวอย่าง การqueryบน FLASHBACK_TRANSACTION_QUERY view เพื่อเอาค่า transactionID, operation ที่ทำ, operation startและend SCNs, userที่เป็นคนรับผิดชอบในการทำ operationนั้นๆ, และ SQL code ในการทำ undo

```
SELECT   xid,
         operation,
         start_scn,
         commit_scn,
         logon_user,
         undo_sql

FROM     flashback_transaction_query

WHERE    xid = HEXTORAW('000200030000002D');
```

ตัวอย่างนี้ เคยกล่าวถึงแล้วในหัวข้อการทำ Flashback Version Query ซึ่งตอนนี้จะแสดงขั้นตอนจริงในการ query โดยทำ Flashback Version Query เป็น subquery ซึ่งจะใช้หาว่าใครเป็นคนมาแก้ไขเปลี่ยนแปลงข้อมูลrowนี้ โดยใช้ LOGON_USER เป็นผู้รับผิดชอบในการทำการเปลี่ยนปลงนั้นๆ

```
SELECT   xid, logon_user
FROM     flashback_transaction_query
WHERE    xid IN (
         SELECT   versions_xid
```

```
    FROM employees
    VERSIONS  BETWEEN TIMESTAMP
    TO_TIMESTAMP ('2003-07-18 14:00:00', 'YYYY-MM-DD HH24:MI:SS')
    AND
    TO_TIMESTAMP ('2003-07-18 17:00:00', 'YYYY-MM-DD HH24:MI:SS')
);
```

ตัวอย่างต่อไปนี้เป็นตัวอย่างการใช้งาน Flashback Transaction Query และ Flashback Version Query ร่วมกัน โดย DBA กระทำการสร้าง table 2 table ชื่อ emp กับ dept ตามลำดับดังนี้:

```
connect hr/hr

CREATE TABLE emp (
        empno   NUMBER PRIMARY KEY,
        empname VARCHAR2(16),
        salary  NUMBER
);

INSERT INTO emp
VALUES (111, 'Mike', 555);
COMMIT;

CREATE TABLE dept (
        deptno   NUMBER,
        deptname VARCHAR2(32)
);
INSERT INTO dept
VALUES (10, 'Accounting');
COMMIT;
```

ในขณะนี้ table 2 table จะมี row 1 row ถ้าพูดถึง version จะหมายถึง แต่ละตาราง ตอนนี้มี 1 version 1 row สมมุติ ต่อมาเกิดความผิดพลาดของtransactionในการ delete ลูกจ้าง ที่มีหมายเลข id = 111 จากตาราง emp:

```
UPDATE emp
SET    salary = salary + 100
WHERE  empno = 111;
INSERT INTO dept
VALUES (20, 'Finance');
DELETE
FROM emp
WHERE empno = 111;
COMMIT;
```

สมมุติว่าตอนนี้ DBA ตรวจสอบพบว่าapplicationเกิดข้อผิดพลาดและต้องการจะวิเคราะห์ว่าเกิดจากปัญหาเกิดจากอะไร DBAจึงทำการquery เพื่อค้นคืน version ของ rows จาก emp table ซึ่งมีข้อมูลตรงกับ empno=111 โดยการ queryนั้น DBA ใช้ Flashback Version Query pseudocolums

```
Connect  dba_name/password
SELECT  versions_xid XID ,
      versions_startscn START_SCN ,
      versions_endscn END_SCN ,
      versions_operation OPERATION ,
      empname ,
      salary
FROM    hr.emp
VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
Where   empno = 111;
```

ผลลัพธ์จะออกมาเป็นtable โดยจะเรียงตามวันเดือนปีจากข้างล่างขึ้นข้างบน โดย rowที่3 จะเป็นversionของ row ในตารางempที่เก่าที่สุด ซึ่งจะเป็นการ insert row นี้ครั้งแรกตอนสร้างตาราง

rowที่2 เป็นvesionที่rowนี้ถูกdeleteเมื่อเกิดความผิดพลาดของtransaction row1 เป็นversionที่เกิดจากrowนี้ถูก reinsert ใหม่ด้วยชื่อลูกจ้างคนใหม่ ที่ชื่อว่าTom

XID	START_SCN	END_SCN	OPERATION	EMPNAME	SALARY
1 0004000700000058	113855		I	Tom	927
2 000200030000002D	113564		D	Mike	555
3 000200030000002E	112670	113564	I	Mike	555

3 rows selected

DBA พบว่าTransactionหมายเลข 000200030000002D เกิดความผิดพลาด...

DBAจึงทำ Flashback Transaction Query เพื่อตรวจสอบทุกๆอย่างที่ทำโดย transactionนี้

```
SELECT   xid, start_scn START, commit_scn COMMIT,
         operation OP, logon_user USER, undo_sql
FROM     flashback_transaction_query
WHERE    xid = HEXTORAW('000200030000002D');
```

XID	START	COMMIT	OP	USER	UNDO_SQL
000200030000002D	195243	195244	DELETE	HR	**insert** into "HR"."EMP" ("EMPNO", "EMPNAME", "SALARY") values ('111','Mike','655');
000200030000002D	195243	195244	INSERT	HR	**delete** from "HR"."DEPT" where ROWID = 'AAAKD4AABAAAJ3BAAB';
000200030000002D	195243	195244	UPDATE	HR	**update** "HR"."EMP" set "SALARY" = '555' where ROWID = 'AAAKD2AABAAAJ29AAA';

0002000300000002D 195243 113565 BEGIN HR

4 rows selected

ทางด้านขวาสุดจะเป็น colum undo_sql จะเก็บข้อมูลซึ่งเป็น คำสั่งSQL ซึ่งเป็นคำสั่งทำให้เกิดการเปลี่ยนแปลง และ DBAสามารถปฏิบัติคำสั่งเหล่านี้เพื่อundoการเปลี่ยนแปลงดังกล่าวที่ทำโดย Transaction และ USER column(logon_user) จะแสดงuserที่เป็นคนกระทำtransactionนั้น

DBA อาจจะสนใจอยากรู้ทุกๆ การเปลี่ยนแปลงที่เกิดจากเวลานั้น ในสถานการณ์ตัวอย่างนี้ DBA ปฏิบัติคำสั่งqueryเพื่อดูรายละเอียดของทุกๆtransactionที่executeตั้งแต่เกิดความผิดพลาดของ transactionที่เราระบุได้ว่าผิดพลาดในก่อนหน้านี้(ดูรายละเอียดของตัวที่ผิดพลาดด้วย)

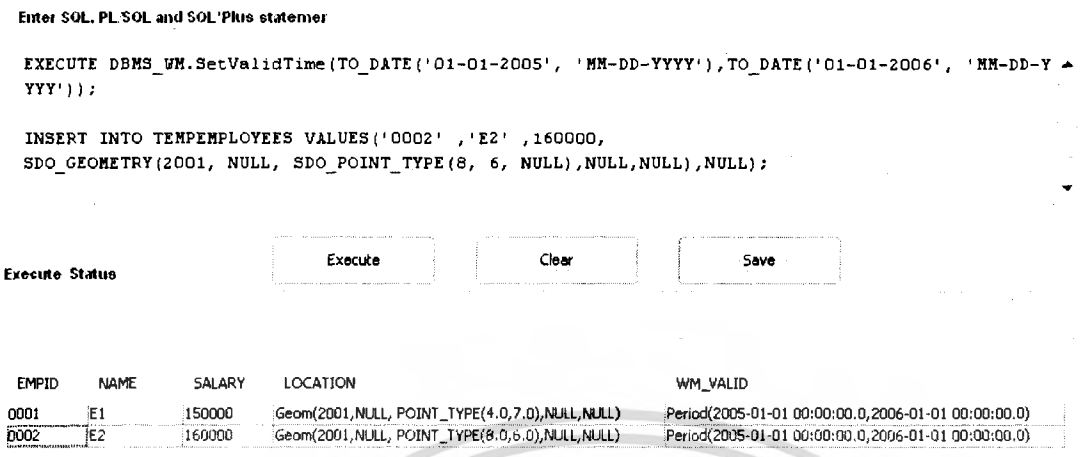
```
SELECT    xid ,start_scn ,commit_scn , operation , table_name ,table_owner
FROM      flashback_transaction_query
WHERE     table_owner = 'HR'
      AND
      start_timestamp >= TO_TIMESTAMP
      ('2002-04-16 11:00:00','YYYY-MM-DD  HH:MI:SS');
```

XID	START_SCN	COMMIT_SCN	OPERATION	TABLE_NAME	TABLE_OWNER
0004000700000058	195245	195246	UPDATE	EMP	HR
0004000700000058	195245	195246	UPDATE	EMP	HR
0004000700000058	195245	195246	INSERT	EMP	HR
0002000300000002D	195243	195244	DELETE	EMP	HR
0002000300000002D	195243	195244	INSERT	DEPT	HR
0002000300000002D	195243	195244	UPDATE	EMP	HR

6 rows selected

4.4 การทดลอง INSERT UPDATE DELETE บนฐานข้อมูล Spatio Temporal Database

4.4.1 INSERT บนฐานข้อมูล Spatio Temporal Database

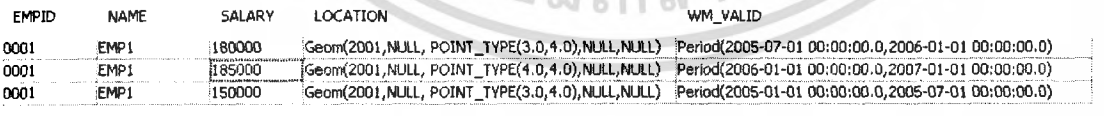


รูปที่ 4.14 INSERT บนฐานข้อมูล Spatio Temporal Database

4.4.2 UPDATE บนฐานข้อมูล Spatio Temporal Database

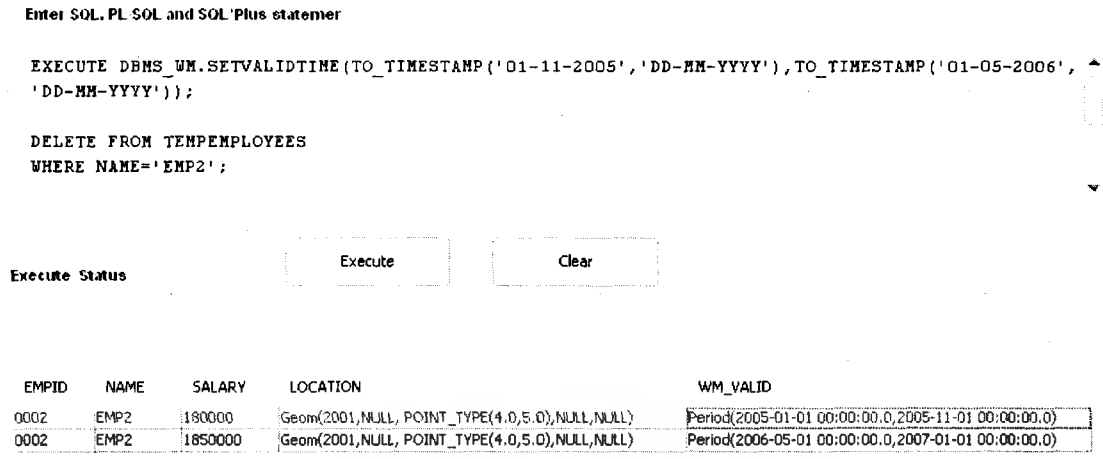
รูปที่ 4.15 Update บนฐานข้อมูล Spatio Temporal Database

ต้องการ UPDATE ข้อมูลของพนักงาน EMP1 ตั้งแต่วันที่ 01 กรกฎาคม 2005 จนถึง 01 มกราคม 2006 รูปที่ แสดงให้เห็นถึงความเปลี่ยนแปลงของข้อมูล



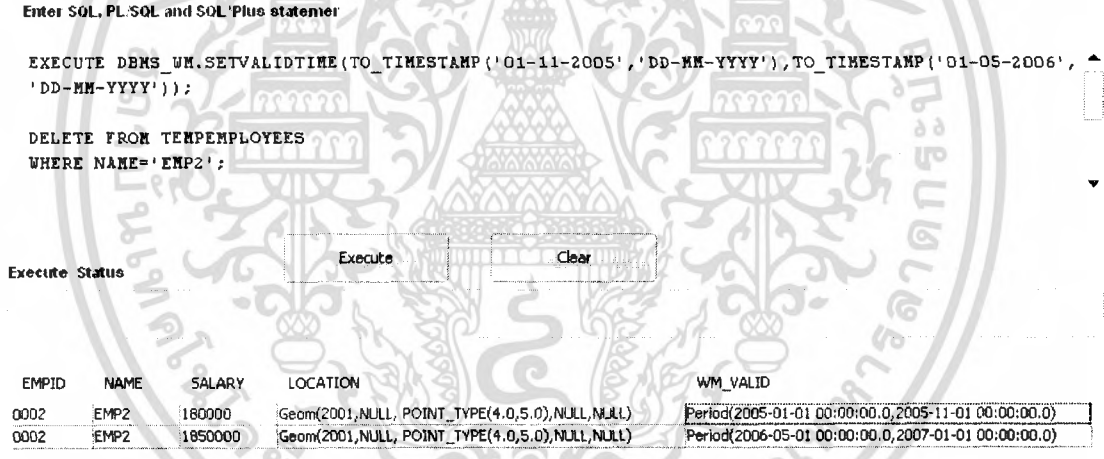
รูปที่ 4.16 ผลลัพท์จาก Update บนฐานข้อมูล Spatio Temporal Database

4.4.3 Sequence Delete บนฐานข้อมูล Spatio Temporal Database



รูปที่ 4.16 DELETE บนฐานข้อมูล Spatio Temporal Database

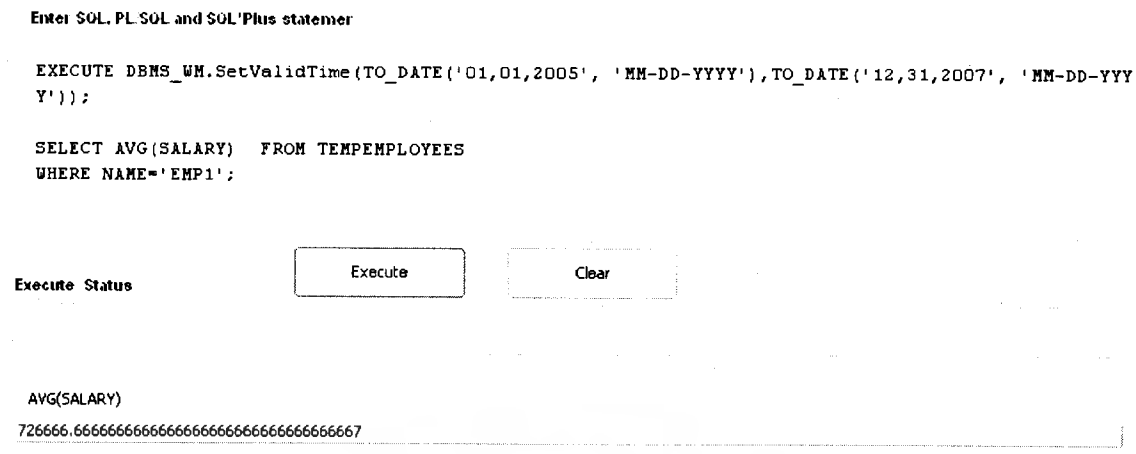
ต้องการลบข้อมูลของพนักงาน EMP2 ตั้งแต่วันที่ 01 พฤษจิกายน 2005 จนถึง 30 เมษายน 2006 รูปที่ แสดงให้เห็นถึงความเปลี่ยนแปลงของข้อมูล



รูปที่ 4.17 ผลลัพธ์ DELETE บนฐานข้อมูล Spatio Temporal Database

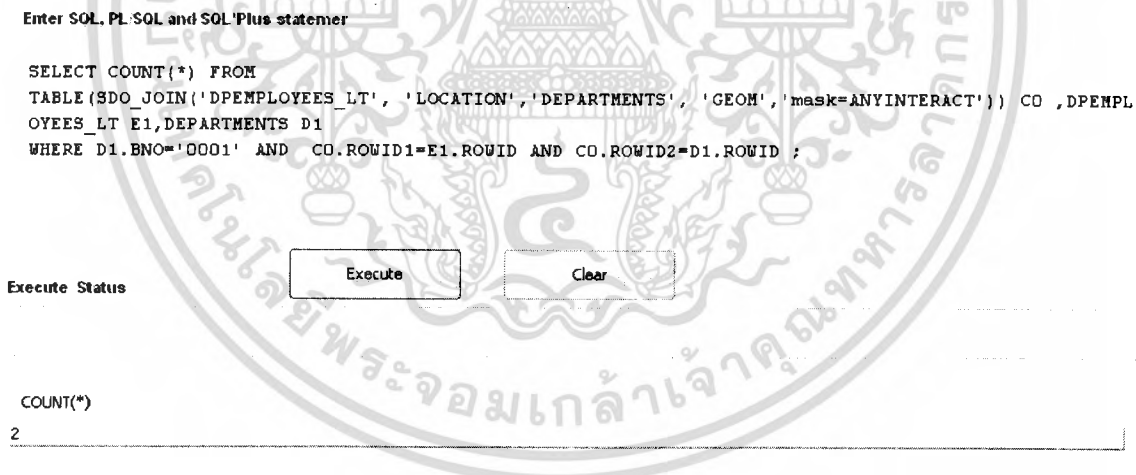
4.5 การทดลอง Query ฐานข้อมูล Spatio Temporal Database

4.5.1 Intra History Cross Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยสนใจเพียง 1 domain ในช่วงเวลาของ Timestamp ที่ต่างกัน เช่น ในช่วงปี 2005-2007 เงินรายได้เฉลี่ยของพนักงาน EMP1 เป็นเท่าไหร่ เป็นต้น จะสังเกตุได้ว่าสิ่งที่เราต้องการนั้นคือ ข้อมูลของพนักงาน EMP1 แต่ความสัมพันธ์ข้อมูลที่ต้องการนั้นผ่าน 2 ช่วงเวลาคือ t1= [01-2005,01-2006) และ t2=[01-2006,01-2007) ผลการทดลองได้ดังนี้



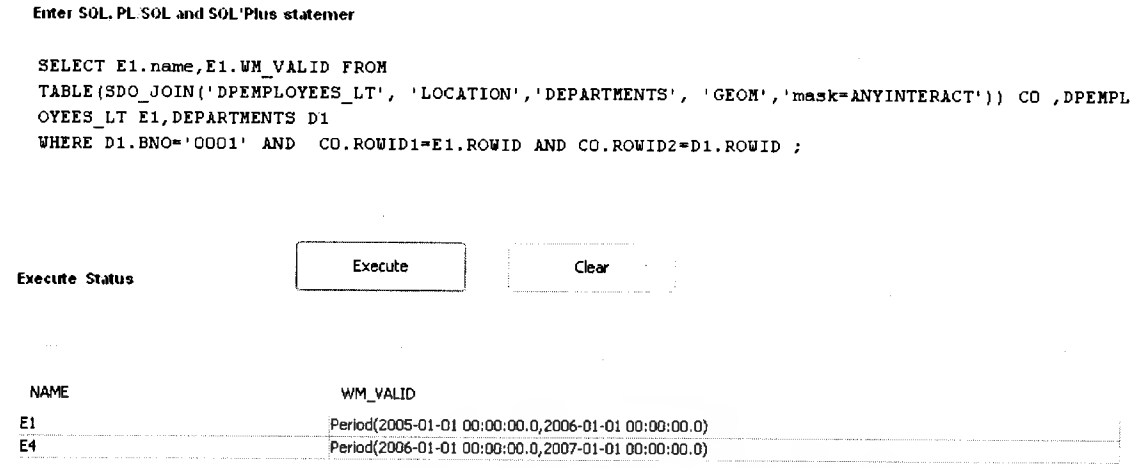
รูปที่ 4.18 การทดลอง Query แบบ Intra History Cross Timestamp

4.5.2 Cross History Cross Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจตั้งแต่ 2 domain ขึ้นไป ในช่วงเวลาของ Timestamp ที่ต่างกัน เช่น แผนก DPA เคยมีจำนวนพนักงานกี่คน เป็นต้น ซึ่งต้องอาศัย Spatial joint เพื่อหาความสัมพันธ์ของ ตำแหน่งของ แผนก DPA และตำแหน่งพนักงาน ผลการทดลองได้ดังนี้



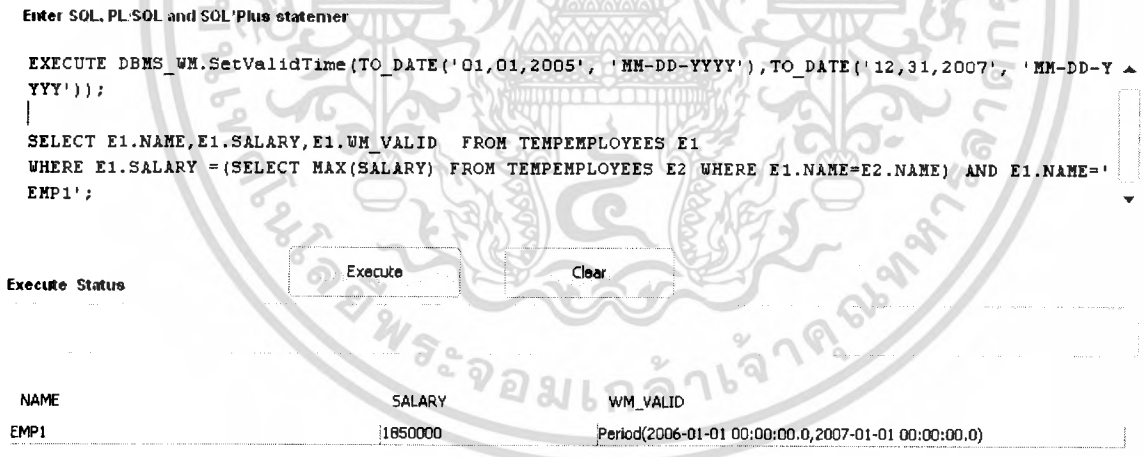
รูปที่ 4.19 การทดลอง Query แบบ Cross History Cross Timestamp

4.5.3 Cross History Intra Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจตั้งแต่ 2 domain ขึ้นไป ในช่วงเวลาของ Timestamp เดียวกัน เช่น ใครเคยอยู่ในแผนก DPA บ้าง เป็นต้น



รูปที่ 4.20 การทดลอง Query แบบ Cross History Intra Timestamp

4.5.4 Intra History Intra Timestamp หมายถึง คำถามที่ระบุการหาคำตอบโดยพิจารณาความสัมพันธ์ของข้อมูลที่เราสนใจเพียง 1 domain และคำตอบที่ได้อยู่ในช่วงเวลาเดียวกัน เช่น ปีไหนพนักงานEMP1 ได้เงินเดือนการเงินเดือนสูงที่สุด เป็นต้น ข้อมูลที่เราสนใจนั้นมีเพียงอย่างเดียว คือ เงินเดือนของพนักงานEMP1 และช่วงเวลามีช่วงเดียว คือ [01-2006,01-2007)



รูปที่ 4.21 การทดลอง Query แบบ Intra History Intra Timestamp

บทที่ 5
บทวิจารณ์และสรุป

5.1. บทสรุป

จากการทดลองได้สรุปความสามารถในการพัฒนาโปรแกรมประยุกต์ บนฐานข้อมูลเชิงเวลาโดยใช้ Oracle 10g ได้ดังนี้

5.1.1 ในส่วนของการใช้ Work Space Manager ของ Oracle 10g สนับสนุน ValidTime นั้น Oracle 10g มีความสามารถที่จะสนับสนุน ฐานข้อมูล Valid Time ได้ตาม[1]ทุกประการ

5.1.2 ในส่วนของการใช้ Flashback ของ Oracle 10g สนับสนุน Transaction Time นั้น Oracle 10g มีความสามารถที่จะสนับสนุน ฐานข้อมูล Transaction Time ได้ตาม[1] ทุกประการ และยังช่วยป้องกันการแก้ไข Log file ได้อีกด้วย

5.1.3 ในส่วนของการใช้ Work Space Manager ร่วมกับ Flashback สนับสนุน เพื่อสนับสนุน Bitemporal นั้นหลังจากที่ทำ EnableVersioning แล้วการ Query จากฐานข้อมูลต้องเปลี่ยนชื่อของตารางเดิมเป็น “ชื่อตารางเดิม_LT” เนื่องจากการทำ EnableVersioning DBMS จะทำการเปลี่ยนชื่อตารางเดิมของเราให้เป็น “ชื่อตารางเดิม_LT” เพื่อใช้เก็บข้อมูลจริง และสร้าง Database View ขึ้นมามีชื่อตรงกับชื่อตารางเดิมของเรา เพื่อใช้ในการจัดการข้อมูล

5.1.4 ในส่วนของการใช้ Work Space Manager ร่วมกับ Oracle Spatial เพื่อให้สนับสนุนฐานข้อมูลแบบ Spatio-temporal Database ในส่วนของการ INSERT UPDATE DELETE นั้นสามารถทำได้ตามได้[1]ทุกประการ ในส่วนของการ Query จะมีข้อจำกัดในเรื่องของการทำ Spatial Index คือการทำเมื่อ EnableVersioning DBMSจะไม่อณุญาติให้ผู้ใช้ ทำการ เปลี่ยนแปลงโครงสร้างของตารางและ Index ได้จนกว่าจะทำการ DisableVersion ก่อน ซึ่งส่งผลให้ Spatial Index ไม่สามารถ update ได้อาจจะทำให้ความเร็วในการ Query ลดลง จำไม่ไชเหมาะกับ Application ที่มีการเปลี่ยนแปลงตำแหน่งบ่อยๆ เพราะการสร้าง Spatial Index นั้นจะต้องมีข้อมูลในตารางก่อน Spatial Index จึงจะสามารถ Map มาเป็น R-Tree ได้

5.2 วิจารณ์สิ่งที่ได้จากโครงงาน

5.2.1 โปรแกรมที่สามารถแสดงข้อมูล SDO_GEOMETRY และ WM_PERIOD ในรูปแบบของตัวอักษรได้

5.2.2 วิธีและขั้นตอนในการใช้ Oracle 10g ในสนับสนุน ฐานข้อมูลเชิงเวลาแบบต่างๆ เพื่อใช้ในการพัฒนา Application บนฐานข้อมูลเชิงเวลาต่อไป

5.3 ปัญหาอุปสรรคและแนวทางแก้ไข

5.3.1 ในส่วนของการใช้ Work Space Manager ของ Oracle 10g สนับสนุน ValidTime

5.3.1.1 การทำใช้ Work Space Manager หลังจากทำการ EnbleVersioning แล้ว DBMS จะไม่อนุญาติ ให้ผู้ใชทำการเปลี่ยนแปลงโครสร้างของตาราง แนวทางการแก้ปัณหา คือ ให้ทำการ SetValidTime ของ Session ให้ครอบคลุมการช่วงเวลาของข้อมูลในตารางทั้งหมดเพื่อไม่ให้ข้อมูลถูกลบหลังจากการทำ DisableVersion ทำการ DisableVersion โดยไม่ต้องลบคอลัมน์ WM_VALID หลังจากนั้นจำทำการเปลี่ยนแปลงโครสร้างของตารางแล้วจึงทำการ EnbleVersioning กลับ

5.3.2 ในส่วนของการใช้ Flashback ของ Oracle 10g สนับสนุน Transaction Time

5.3.2.1 ขนาดของ Flashback Area น้อย การ Flashback สนับสนุน Transaction Time ต้องมั้นใจว่ามีขนาดของ Flashback ใหญ่เพียงพอมิเช่นนั้นจะสามารถ Query ย้อนหลังได้ไม่นาน เนื่องจาก ข้อมูลจะถูกเก็บไว้ Flashback Area ซึ่งอยู่ในส่วนของ Redo log

5.3.2.2 หากตารางของเรามีการเปลี่ยนแปลงโครสร้างของตาราง จะไม่สามารถใช้ Flashback ได้โดยจะได้รับข้อความแจ้ง Error

```
ORA-01466: unable to read data - table definition has changed
```

ให้ทำการ Export ข้อมูลออกจารตารางเดิมก่อนแล้วทำการ Drop ตารางนั้นทิ้ง สร้างตารางใหม่ขึ้นมา และจึง Import ข้อมูลกลับ แต่ข้อมูลของ Transaction Time ก่อนหน้านี้ก็จะไม่สามารถดูได้

5.3.3 ในส่วนของการใช้ Work Space Manager ร่วมกับ Flashback สนับสนุน เพื่อสนับสนุน Bitemporal

5.3.3.1 นั้นหลังจากที่ทำ EnableVersioning และเมื่อใช้ Flashback อาจได้รับข้อความแจ้ง Error

```
ORA-01466: unable to read data - table definition has changed
```

เนื่องจากการทำ EnableVersioning นั้นต้องมีการเปลี่ยนแปลงโครงสร้างของตารางดังนี้
เมื่อจะใช้ Bitemporal สร้างตารางแล้วต้องทำ EnableVersioning ก่อนนำข้อมูลลงเสมอ
มิฉะนั้นจะไม่สามารถใช้ Flashback ได้
ไม่สามารถการทำ Rebuild หรือ Drop Spatial Index หลังจาก EnableVersioning
เนื่องจากเมื่อ EnableVersioning DBMSจะไม่อณุญาติให้ผู้ใช้ ทำการ เปลี่ยนแปลง
โครงสร้างของตารางและ Index ได้จนกว่าจะทำการ DisableVersion ก่อน ซึ่งส่งผลให้
Spatial Index ไม่สามารถ update ได้อาจจะทำให้ความเร็วในการ Query ลดลง

5.4 แนวทางการพัฒนาต่อ

โปรเจ็คนี้ได้นำเสนอข้อมูลเกี่ยวพัฒนาโปรแกรมประยุกต์ บนฐานข้อมูลเชิงเวลาโดยได้ยกตัวอย่าง Spatio Temporal Database ซึ่งเป็นการนำคุณสมบัติของฐานข้อมูลเชิงเวลามาใช้ร่วมกับฐานข้อมูลเชิงพื้นที่ หากต้องการนำโปรเจ็คนี้ไปพัฒนาต่อควรศึกษาเพิ่มเติม การออกแบบ ฐานข้อมูลเชิงพื้นที่และการนำข้อมูลที่ได้จาก ฐานข้อมูลมาแสดงในรูปแบบของภาพ

บรรณานุกรม

[1] Richard T. Snodgrass " Managing Temporal Data A Five-Part Series "
Available : http://www.cs.auc.dk/research/DBS/tdb/TimeCenter/TR-28.pdf .1998

[2] Abdulla Uz Tansel and Group "Temporal Database" The Bejamin/Cummings Publishing Company, Inc. 1993

[3] Abdullah Uz Tansel, "Integrity Constraints in Temporal Relational Databases," itcc, p.460, International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2, 2004

[4] Oracle. "Workspace Manager Valid-Time Support " [Online].
Available : http://web.deu.edu.tr/oracle/B19306_01/appdev.102/b14224.pdf 2005.

[5] Oracle. "Flashback Feature." [Online].
Available : http://web.deu.edu.tr/oracle/B19306_01/appdev.102/b10851.pdf 2005.

[6] Nassima Djafri, Alvaro A.A. Fernandes, Norman W. Paton, Tony Griffiths,"Spatio-Temporal Evolution:Querying Patterns of Change in Databases"

[7] Oracle."Oracle Spatial"
Available : http://download-east.oracle.com/docs/html/A85337_01/sdo_intr.htm 2005.

ภาคผนวก ก.

Oracle Database 10g Release 2 (10.2.0.1) Installation Guide

1. Minimum Hardware Requirements

Hardware Items	Minimum Value
Physical Memory (RAM)	อย่างต่ำ 256 MB แนะนำให้มี 512 MB
Virtual Memory	ขนาดเป็น 2 เท่าของ RAM
Disk space	Basic Installation Type : 2.04 GB Advanced Installation Type : 1.94 GB ดูรายละเอียดในตารางที่ 2
Video Adapter	256 สี
Processor	550 MHz

1.1 **Harddisk Space Requirements** เป็นดังตารางซึ่งเป็นค่าสำหรับ NTFS (ถ้าเป็น FAT32 จะมากกว่านี้เล็กน้อย)

Installation Type	TEMP Space	C:\Program Files\Oracle	Oracle Home	Datafiles *	Total
Basic Installation	125 MB	3.1 MB	905 MB	1.03 GB	2.04 GB
Advanced Installation	125 MB	3.1 MB	905 MB **	950 MB **	1.94 GB **

* เป็นเนื้อที่สำหรับ admin, flash_recovery_area, และ oradata directories ใน *ORACLE_BASE* directory.

** อาจจะใช้พื้นที่มากกว่านี้โดยขึ้นอยู่กับ options ที่เลือกขณะ install ถ้า install Oracle Database โดยเลือกให้มี Automated backups ด้วยจะต้องใช้พื้นที่เพิ่มอีกอย่างน้อย 2GB

2. Minimum Software Requirement

Requirement	Value
System Architecture	Processor: Intel (x86), AMD64, and Intel EM64T
Operating System	Windows 2000 service pack 1 หรือใหม่กว่านั้น Windows Server 2003 ทุก editions Window XP Professional ไม่รองรับ Windows NT ได้
Compiler	ACUCOBOL-GT version 6.2 Micro Focus Net Express 4.0 ไม่รองรับ Object Oriented COBOL (OOCOBOL)
Network Protocol	TCP/IP TCP/IP with SSL Named Pipes

3. วิธีการติดตั้ง Oracle Database Server 10g release 2 แบบ Basic

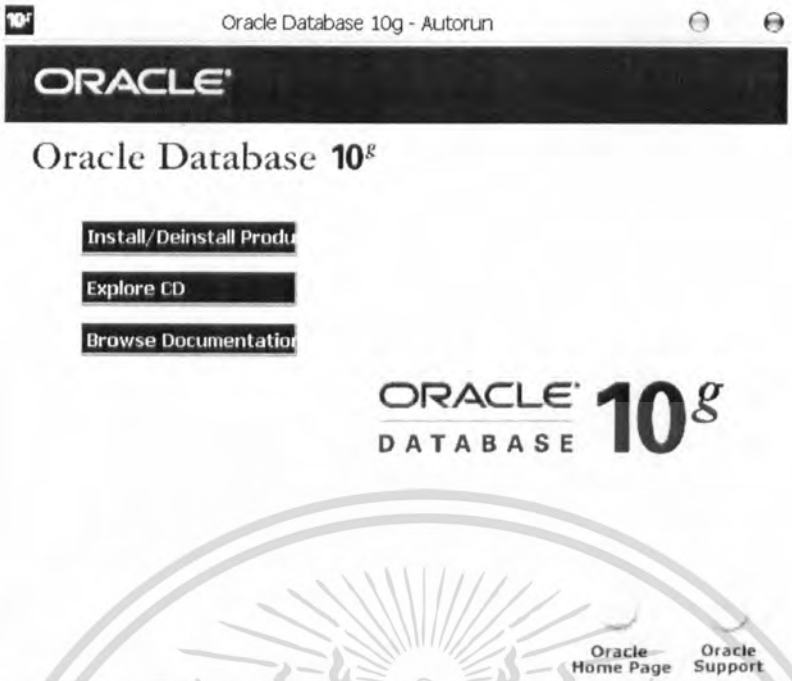
3.1 double-click file "setup.exe" มีสัญลักษณ์ดังรูป 

เพื่อ start Oracle Universal In-staller หรือรอให้ Autorun ทำงานซึ่งจะแสดงหน้าต่างดังภาพ ให้เลือก Install/Deinstall Product

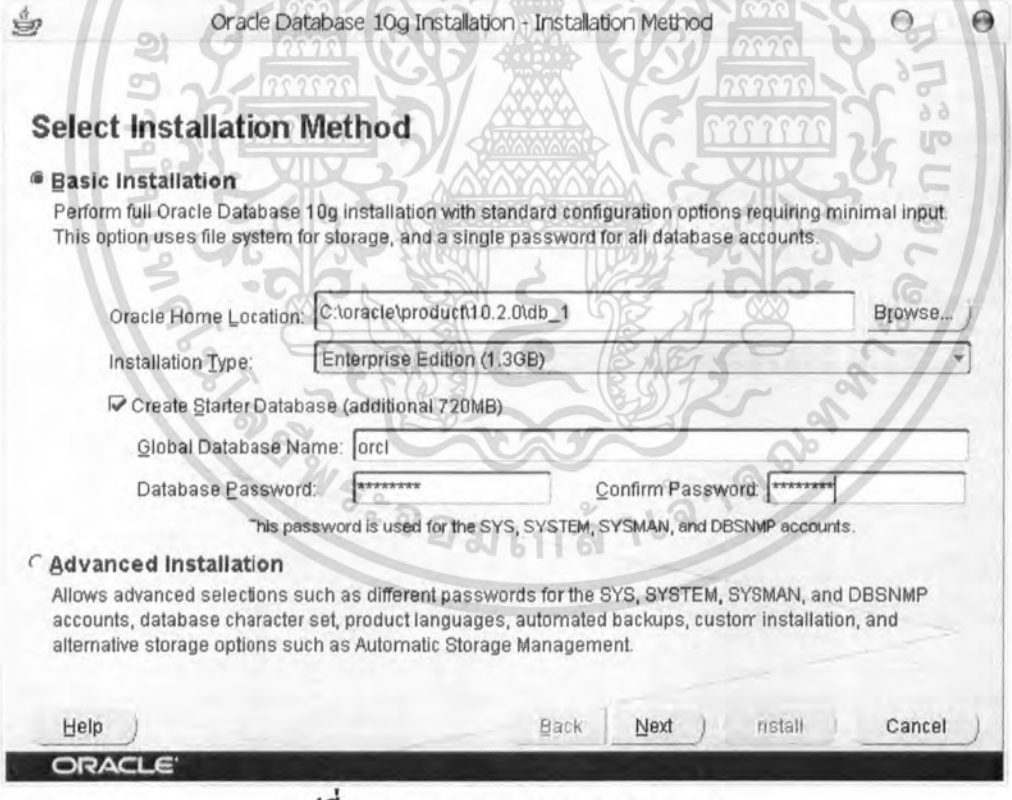
3.2 เข้าสู่ขั้นตอนการเลือก mode ของการ install ให้เลือก Basic Installation และในช่องของ Database Password และ Confirm Password ให้ใส่ Password ที่มีความยาวระหว่าง 4 ถึง 30 ตัวอักษร โดยสามารถมีสัญลักษณ์ต่อไปนี้ได้ คือ underscore(_), dollar($), pound sign(#) และ

- ห้ามขึ้นต้นด้วยตัวเลข
- ห้ามใส่ password ที่เหมือนกับ user name
- ห้ามใช้คำสงวนของ Oracle
- ห้ามใช้คำว่า change_on_install, sysman, dbsnmp เป็น password

หมายเหตุ ในที่นี้ให้ใส่คำว่า password ทั้งในช่อง Database Password และ Confirm Password) แล้ว click ปุ่ม Next



รูปที่ ก-1 Oracle Database 10g Autorun

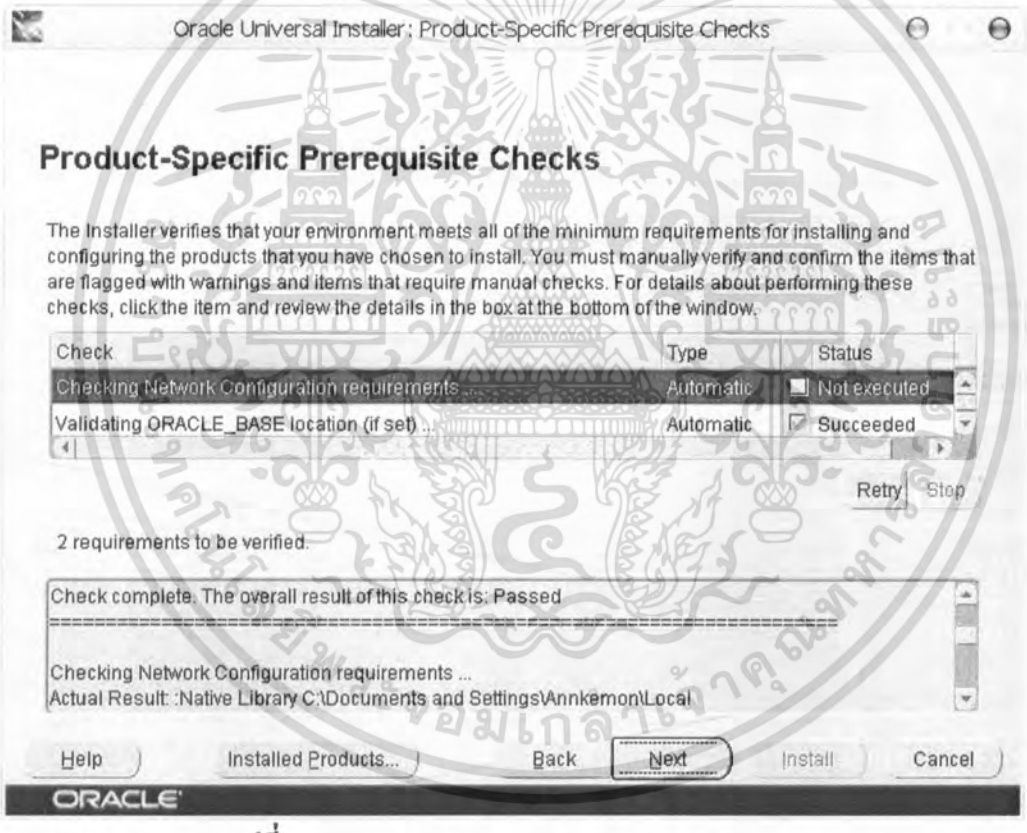


รูปที่ ก-2 Install/Deinstall Product

หลังจากนั้นระบบจะเตรียมการ Install ดังรูป

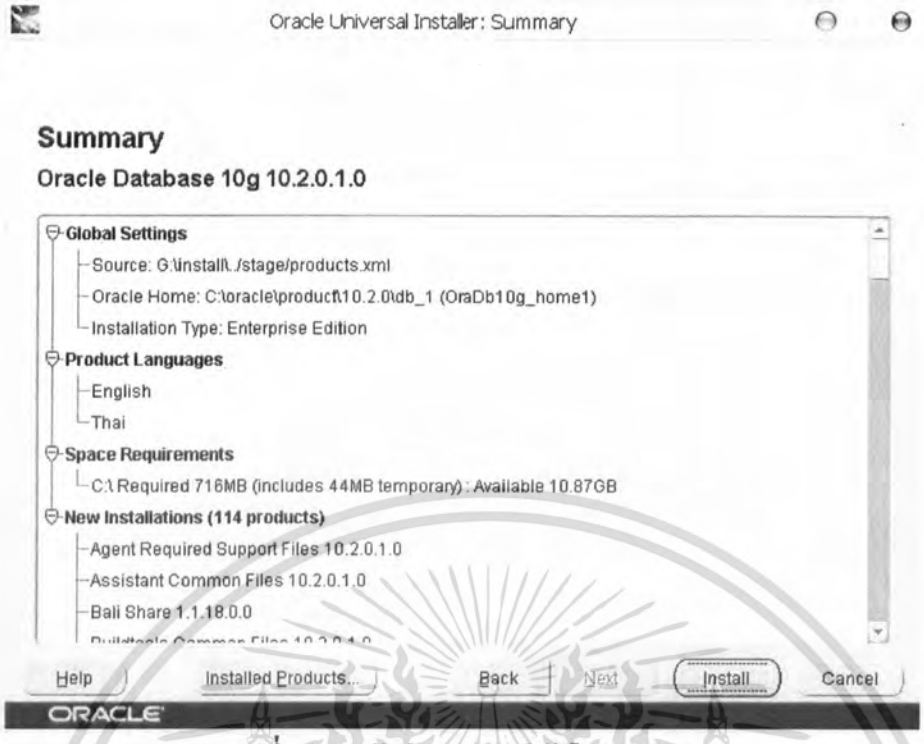
รูปที่ ก-3 ระบบจะเตรียมการ Install

3.3 เมื่อแสดงหน้าต่าง Product-Specific Prerequisite Checks ให้ click ปุ่ม next



รูปที่ ก-4 Product-Specific Prerequisite Checks

3.4 จากหน้าต่างสรุปผลให้เลือก Install



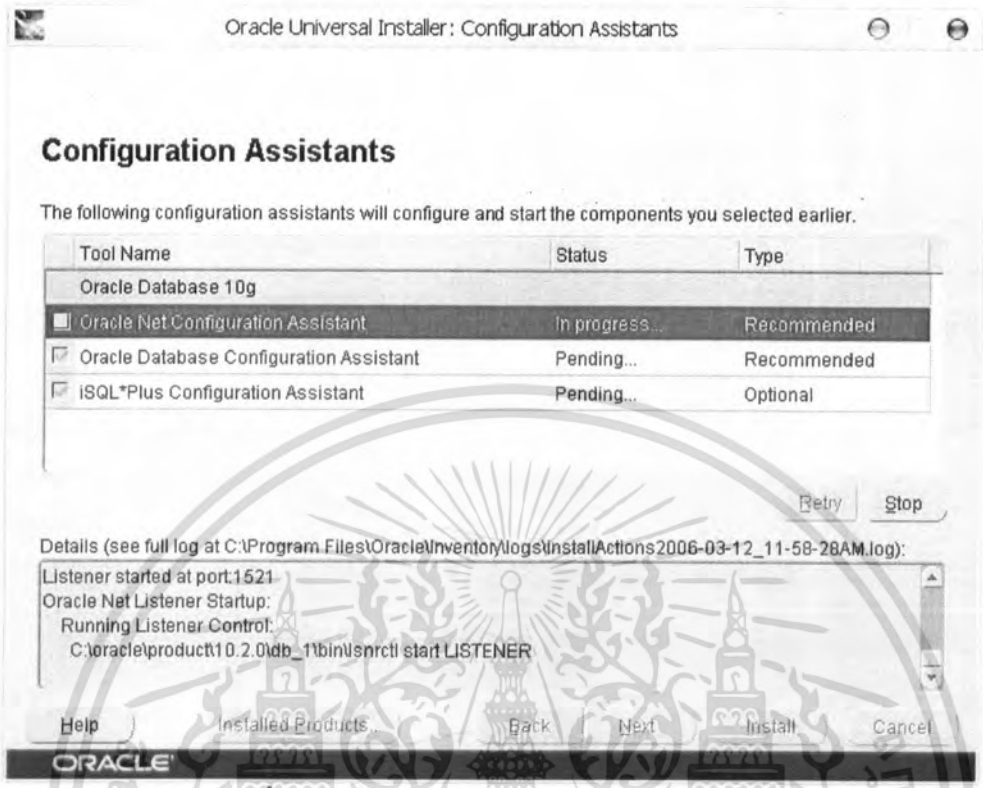
รูปที่ ก-5 หน้าต่างสรุปผลให้เลือก Install

ขณะ install จะแสดงหน้าต่าง Install ดังรูป

รูปที่ ก-6 หน้าต่าง Install

หลังจากนั้นจะแสดงหน้าต่าง Oracle Universal Installer : Configuration Assistants (ซึ่ง ผู้

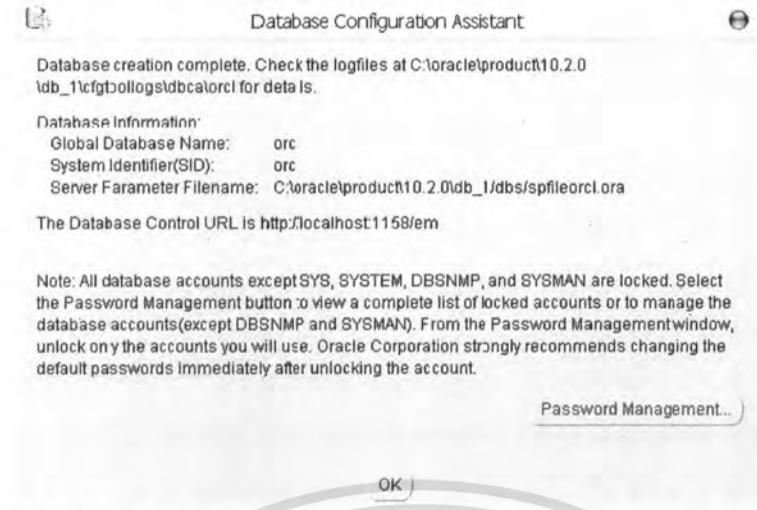
ติดตั้งไม่ต้องทำอะไร) ดังรูป



รูปที่ ก-7 หน้าต่าง Oracle Universal Installer

ต่อมาจะแสดงหน้าต่าง Database Configuration Assistant (ซึ่งผู้ติดตั้งไม่ต้องทำอะไร) ดังรูป 3.5 ในขั้นตอนต่อมาเป็นการแสดงรายละเอียดผลของการ install ให้ click ปุ่ม OK

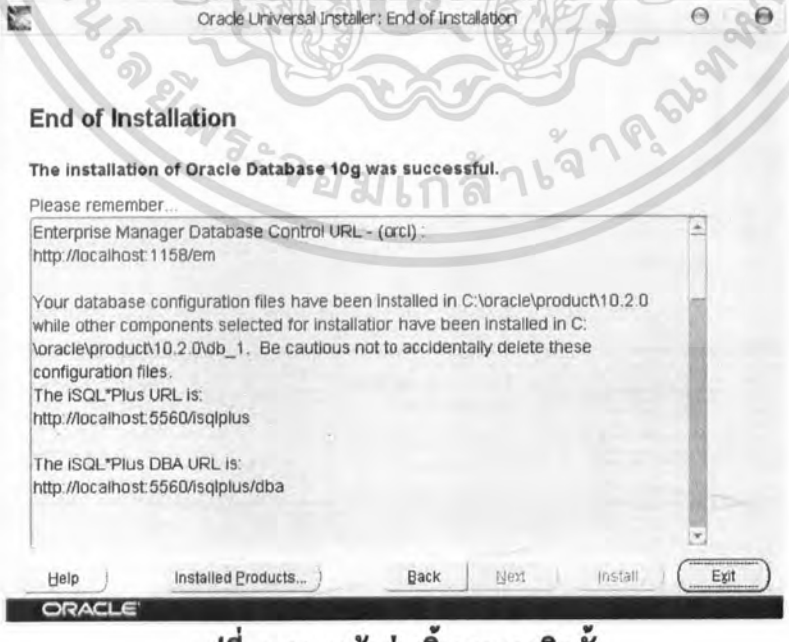
รูปที่ ก-8 หน้าต่าง Database Configuration Assistant



รูปที่ ก-9 หน้าต่าง Database Configuration Assistant Password Management

3.6 ในขั้นตอนสุดท้ายของการ install ให้ click ปุ่ม Exit

รูปที่ ก-10 หน้าต่างยืนยันการออกจากการติดตั้ง

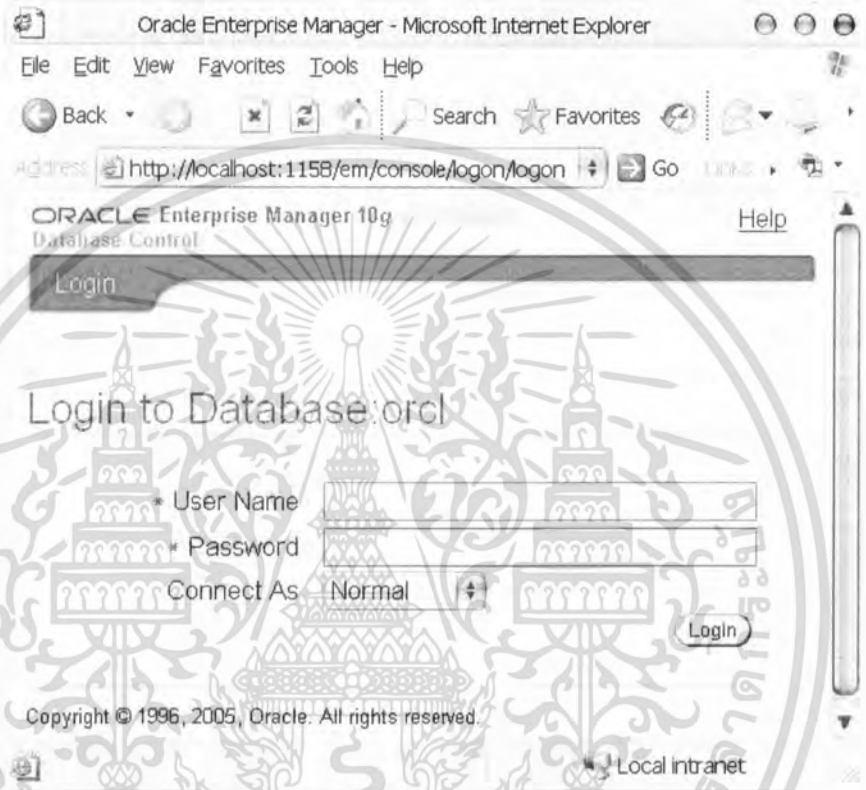


รูปที่ ก-11 หน้าต่างสิ้นสุดการติดตั้ง

4 ขั้นตอนหลังการติดตั้ง

4.1 หลังจากการติดตั้ง

ให้เปิด Web browser เช่น Internet Explorer หรือ Firefox แล้วเข้าสู่หน้าแรกของ Oracle Database ที่ http://localhost:1158/em/



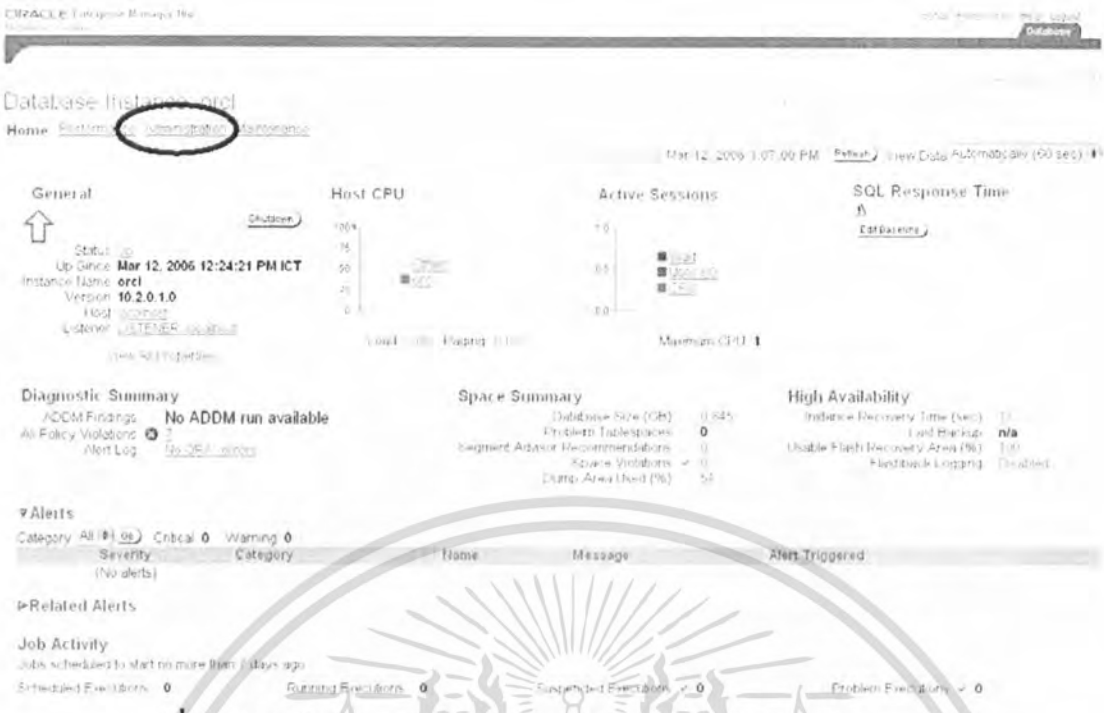
รูปที่ ก-12 หน้าต่างการ Login Oracle 10g Enterpris Manager

ช่อง Password ใส่ว่า password แล้ว click ปุ่ม Login

หน้าที่แสดง Oracle D

ปุ่ม

4.4 หลังจากนั้น browser จะแสดงดังรูป ให้เลือก link ที่ Administration



รูปที่ ก-13 หน้าต่างการหลังจาก Login Oracle 10g Enterpris Manager

4.5 ในการใช้งานจะต้องสร้าง Table space เพื่อเป็นพื้นที่ในการเก็บข้อมูล จากภาพเป็นผล จาก link ของ Administration ให้เลือก link ที่ Tablespace จากหมวดหมู่ Storage

รูปที่ ก-14 หน้าต่างจาก link ของ Administration

4.6 ในหน้า Tablespaces ให้ click ปุ่ม Create เพื่อสร้าง Tablespace

รูปที่ ก-15 หน้าต่าง Tablespaces

4.7 ตั้งชื่อ Tablespace ใส่ในช่อง Name ในที่นี้ใช้ชื่อว่า CARDB ในส่วนของ Datafiles ให้ click ที่ Add

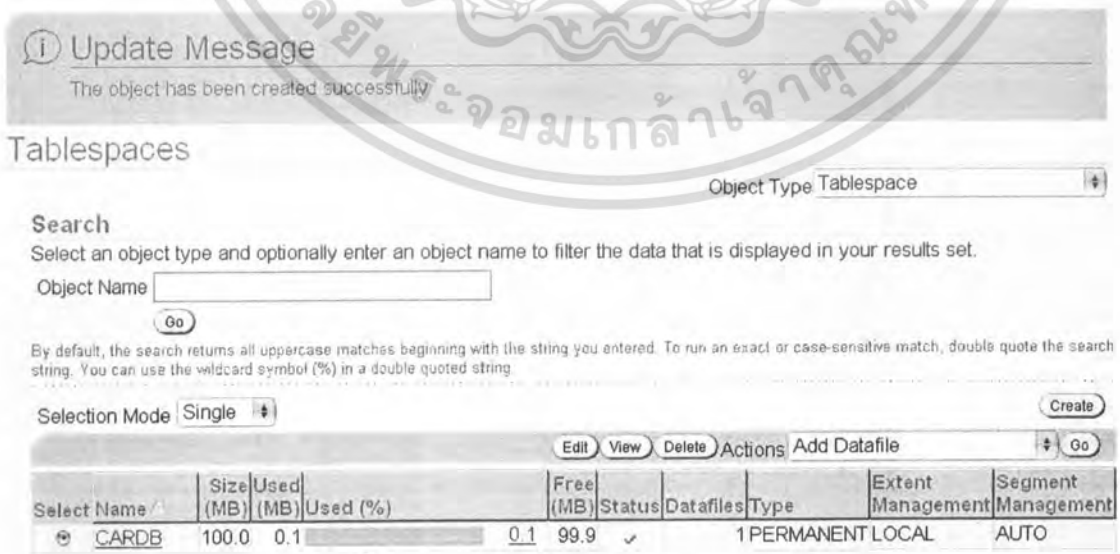
รูปที่ ก-16 หน้าต่าง สร้าง Tablespaces

4.8 ตั้งชื่อ file ของฐานข้อมูลใส่ในช่อง File Name ในที่นี้ใส่ชื่อ CRDB แล้ว click ปุ่ม Continue

ขึ้นมา ให้ click ปุ่ม OK

รูปที่ ก-17 หน้าต่างเพิ่ม Datafile

4.10 หากสร้าง Tablespace สำเร็จจะมีแถวของ Tablespace ที่เพิ่งสร้างขึ้นดังรูป



รูปที่ ก-18 แสดงเมื่อสร้าง Tablespace เสร็จ

4.11 เลือก link Database Instance: orcl เพื่อกลับไปยังหน้าของ Administration ดังรูปใน ข้อ 4.5

4.12 ขั้นตอนต่อไปจะเป็นการเพิ่มผู้ใช้งานเข้าไปในระบบโดยจากหน้า Administration ให้ เลือก link ที่ Users ในหมวดหมู่ของ Users & Privileges

4.13 ในหน้าของ Users จะแสดง user ที่มีอยู่ในปัจจุบัน ให้ click ปุ่ม Create

รูปที่ ก-19 หน้าแสดง user

4.14 ช่อง Name : ใส่ชื่อผู้ใช้ที่ต้องการเพิ่ม ในที่นี้ให้ใส่เป็น Admin

ช่อง Enter Password และ Confirm Password เป็น Password ที่จะใช้ในการเข้าถึง ฐานข้อมูลในที่นี้ให้ใส่เป็น password ทั้ง 2 ช่อง

รูปที่ ก-20 หน้าสร้าง User

4.15 ช่อง Default Tablespace ให้ click ที่  เลือก CARDB แล้ว click ที่ปุ่ม Selectช่อง Temporary Tablespace ให้ทำเช่นเดียวกันแต่เลือก TEMP แทน

รูปที่ ก-21 หน้าเลือก Default Tablespace

4.16 ผลลัพธ์ควรเป็นดังรูป ปัจจุบันหน้า Create User อยู่ที่ Tab General ให้ click link ไปที่ Roles

รูปที่ ก-22 หน้า Create User

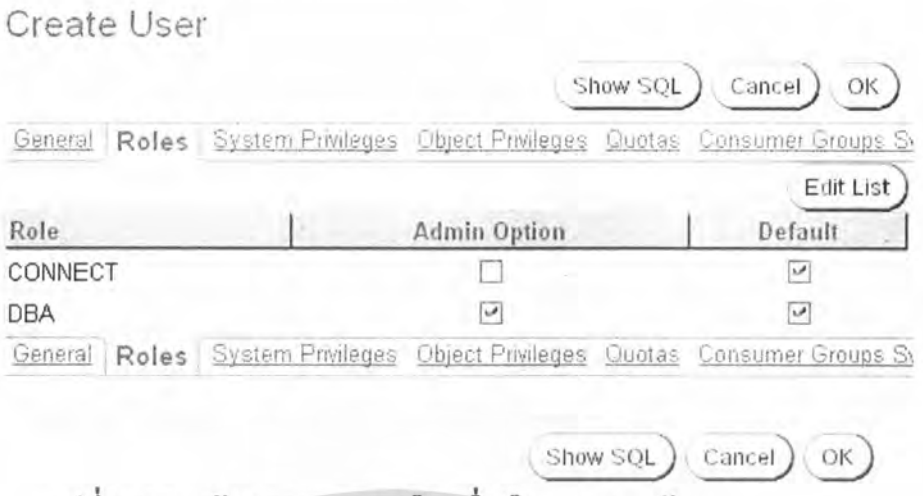
4.17 click ที่ปุ่ม 

รูปที่ ก-23 หน้า Create User เลือก Roles

4.18 หน้า Modify Roles เลือก DBA ในช่อง Available Roles แล้ว click ที่ Move แล้ว Click ที่ปุ่ม OK

รูปที่ ก-24 หน้า Modify Roles

4.19 ให้เลือก options ดังรูปแล้ว click ปุ่ม OK



รูปที่ ก-25 หน้า Create User เลือกเมื่อเลือก Roles แล้ว

4.20 หากเพิ่ม user ได้สำเร็จจะมีข้อมูลของผู้ใช้ที่เพิ่งเพิ่มเข้าไป ดังรูป

รูปที่ ก-26 หน้าแสดง User เลือกเมื่อสร้าง User เสร็จแล้ว

4.21 เลือก link Logout เพื่อจบการทำงาน หากต้องการทดสอบว่าเพิ่มผู้ใช้งานในระบบ ได้ สำเร็จหรือไม่ให้ Login โดยใส่ User name และ Password เป็นของผู้ใช้ที่เพิ่ง สร้างขึ้น ในที่นี้ให้ใส่เป็นของ Admin

ภาคผนวก ข.

การสนับสนุน Valid-Time ด้วย Oracle Workspace Manager

1 Workspace Manager คืออะไร

Workspace คือ พื้นที่จำลอง ซึ่งทำให้ผู้ใช้ตั้งแต่หนึ่งคนขึ้นไปสามารถแลกเปลี่ยนหรือใช้ข้อมูลร่วมกันในฐานข้อมูลได้

Workspace Manager เป็นส่วนที่คอยจัดการ Workspace ให้สามารถทำงานได้หลาย Version คือ ฐานข้อมูลสามารถมีหลายเวอร์ชั่นใน Record เดียวกันใน Workspace เดียวหรือหลาย WorkSpace

2 ข้อมูลชนิด WM_PERIOD

ชนิดข้อมูล WM_PERIOD ใช้เพื่อในการระบุช่วงเวลา Valid Time ให้กับ Session , Workspace และสำหรับแต่ละแถว (Tuple) ในตารางที่ได้ทำ Version-enable ไว้แล้ว เราสามารถประกาศชนิดข้อมูล WM_PERIOD ได้ดังนี้

```
CREATE TYPE WM_PERIOD AS OBJECT (     validFrom TIMESTAMP WITH TIME ZONE,
                                      validTill TIMESTAMP WITH TIME ZONE );
```

validFrom คือ เวลาเริ่มต้นของช่วงเวลาที่เรากำหนดโดยเวลานี้จะอยู่ในช่วงเวลานั้นด้วย

validTill คือ เวลาสิ้นสุดของช่วงเวลาที่เรากำหนดโดยเวลานี้จะไม่อยู่ในช่วงเวลานั้นด้วย

ดังนั้นช่วงของเวลาที่ข้อมูลเป็นจริง คือ ตั้งแต่ validFrom ไปจนถึงก่อน validTill

ตัวอย่างการกำหนดช่วงเวลาของฐานข้อมูล ให้อยู่ในช่วงเดือน มกราคม 2549

```
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-2006', 'MM-DD-YYYY'), TO_
DATE('02-01-2006', 'MM-DD-YYYY'));

INSERT INTO employees VALUES('Baxter',40000,
WMSYS.WM_PERIOD(TO_DATE('01-01-2000', 'MM-DD-YYYY'),
DBMS_WM.UNTIL_CHANGED));
```

3 ค่าคงที่ที่ช่วยในการสนับสนุน Valid Time

ตาราง ข-1 แสดงค่าคงที่ที่ช่วยในการสนับสนุน Valid Time

ค่าคงที่	ความหมาย
DBMS_WM.DEFAULT_VALID_FROM	ค่าเวลาอัตโนมัติ สำหรับส่วนของ validFrom
DBMS_WM.DEFAULT_VALID_TILL	ค่าเวลาอัตโนมัติ สำหรับส่วนของ validTill
DBMS_WM.MIN_TIME	ค่าเวลาที่น้อยที่สุดเท่าที่ Workspace Manager จะสนับสนุนได้ซึ่งขณะนี้คือเริ่มจาก วันที่ 01 มกราคม ค.ศ. 4713 (4713 BCE).
DBMS_WM.MAX_TIME	ค่าเวลาที่มากที่สุดเท่าที่ Workspace Manager จะสนับสนุนได้ซึ่งในขณะนี้วันสุดท้ายคือ วันที่ 31 - ธันวาคม-9999.
DBMS_WM.UNTIL_CHANGED	คือ การคงค่าของเวลา ไว้ที่ DBMS_WM.MAX_TIME จนกว่าจะมีการเปลี่ยนแปลงค่า ของข้อมูล

4 API ที่ช่วยสนับสนุน Valid Time

ตารางที่ ข-2 แสดงรายการของโปรแกรมย่อยที่ช่วยสนับสนุน Valid Time รายละเอียดของ Parameter

โปรแกรมย่อย	สนับสนุน Valid Time
EnableVersioning	ถ้าค่าของ validTime parameter เป็น TRUE,จะแสดงว่าตารางนั้นเป็น version-enabled ที่ทำให้สามารถสนับสนุน Valid Time ได้ column ชื่อ WM_VALID ซึ่งข้อมูลเป็นชนิด WM_PERIOD จะถูกเพิ่มเข้าไปในตาราง สำหรับข้อมูลที่มีอยู่ในฐานข้อมูลแต่ก่อนแล้ว column WM_VALID จะถูกกำหนดให้ validFrom timestamp เป็น SYSTIMESTAMP และ validTill timestamp เป็น DBMS_WM.UNTIL_CHANGED.

DisableVersioning	เป็นกำหนดให้เป็น version-disabled โดยจะเป็นการสิ้นสุดการเก็บค่า ตัวแปล keepWMValid หรือเป็นการลบ column WM_VALID
SetValidTime	เป็นการกำหนดช่วงเวลาให้กับ session ของ Valid Time
GetValidFrom	ใช้ดูค่า validFrom timestamp จากช่วงเวลา session ของ valid time
GetValidTill	ใช้ดูค่า validTill timestamp จากช่วงเวลา session ของ valid time

EnableVersioning

เป็นการทำให้ตารางที่ใช้เก็บข้อมูลแบบธรรมดา ให้สามารถสนับสนุน Valid Time โดยจะเพิ่ม Column ชื่อWM_VALID ให้โดยอัตโนมัติ ซึ่งมีรูปแบบคำสั่งดังนี้

```
DBMS_WM.EnableVersioning(
        table_name IN VARCHAR2,
        hist IN VARCHAR2 DEFAULT 'NONE',
        isTopology IN BOOLEAN DEFAULT FALSE,
        validTime IN BOOLEAN DEFAULT FALSE,
        undo_space IN VARCHAR2 DEFAULT NULL);
```

ตัวอย่างเช่น

EXECUTE DBMS_WM.EnableVersioning ('ชื่อตาราง', 'VIEW_WO_OVERWRITE', FALSE, TRUE);

DisableVersioning

เป็นการทำให้ตารางที่ได้ทำการ EnableVersioning แล้วให้กลับเป็นตารางที่ใช้เก็บข้อมูลแบบธรรมดา โดยผู้ใช้สามารถเลือกว่าจะทำการลบ Column ชื่อ WM_VALID หรือไม่ หากต้องการลบให้กำหนดค่าของ keepWMValid ให้เป็น FALSE ซึ่งมีรูปแบบคำสั่งดังนี้

```
DBMS_WM.DisableVersioning(
table_name IN VARCHAR2,
```

force IN BOOLEAN DEFAULT FALSE,

ignore_last_error IN BOOLEAN DEFAULT FALSE,

isTopology IN BOOLEAN DEFAULT FALSE,

keepWMValid IN BOOLEAN DEFAULT TRUE);

ตัวอย่างเช่น

EXECUTE DBMS_WM.DisableVersioning ('ชื่อตาราง');

SetValidTime

เป็นการกำหนดช่วงเวลาของข้อมูลให้กับฐานข้อมูลที่สนับสนุน Valid Time ที่เราต้องการ ซึ่งมีรูปแบบคำสั่งดังนี้

DBMS_WM.SetValidTime(

validFrom IN VARCHAR2 DEFAULT DBMS_WM.CURRENT_TIME,

validTill IN VARCHAR2 DEFAULT DBMS_WM.UNTIL_CHANGED,

validFromFormat IN VARCHAR2 DEFAULT DEFAULT_DATE_FMT,

validTillFormat IN VARCHAR2 DEFAULT DEFAULT_DATE_FMT);

ตัวอย่างเช่น

EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-1998', 'MM-DD-YYYY'), TO_

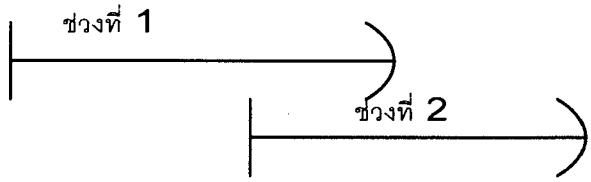
DATE('01-01-1999', 'MM-DD-YYYY'));

5 Operators ที่ช่วยในการสนับสนุน Valid Time

Workspace Manager ได้จัดเตรียมส่วนที่ช่วยกระทำการตรวจสอบความสัมพันธ์ระหว่างสองช่วงเวลา (relationship checking operators) และ ผู้ใช้สามารถนำไปประยุกต์เพื่อทำการค้นหาหรือสอบถาม (query) ได้ ตัวกระทำการ (Operator) ที่ได้จะเตรียมให้มีอยู่สองลักษณะ คือ

1. ตรวจสอบความสำพันธ์ระหว่างสองช่วงเวลา ที่มีผลลัพธ์เป็นค่าเท่ากับ 1 เมื่อมีความสำพันธ์ที่ได้ตรวจสอบ และจะได้ผลลัพธ์เป็น 0 เมื่อไม่มีความสัมพันธ์นั้นอยู่ ได้แก่

   WM_OVERLAPS ตรวจสอบความสัมพันธ์ว่าสองช่วงเวลามีการทับซ้อนกันอยู่หรือไม่



รูปที่ ข-1 ตรวจสอบความสัมพันธ์ว่าสองช่วงเวลามีการทับซ้อนกันอยู่หรือไม่

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_OVERLAPS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในช่วง ปี 1990 - 1999 มีรูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_OVERLAPS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2000', 'MM-DD-YYYY'))) = 1;
```

WM_CONTAINS ตรวจสอบความสัมพันธ์ว่า ช่วงเวลาที่ 2 เป็นส่วนหนึ่งของช่วงเวลาที่ 1 หรือไม่

รูปที่ ข-2 แสดงถึงการตรวจสอบความสัมพันธ์ว่า ช่วงเวลาที่ 2 เป็นส่วนหนึ่งของช่วงเวลาที่ 1 หรือไม่

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_COMTAINS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_CONTAINS(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY'))) = 1;
```

WM_MEETS ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรก เป็นเวลาเริ่มต้นของช่วงเวลาที่สอง หรือไม่

รูปที่ ข-3 แสดง WM_MEETS

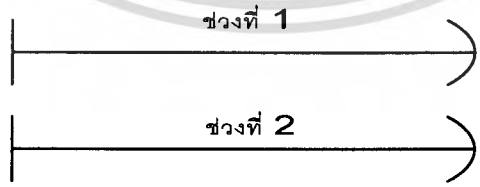
โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_MEETS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_MEETS(e.wm_valid,
        wm_period(TO_DATE('01-01-2005', 'MM-DD-YYYY'),
        TO_DATE('01-01-2006', 'MM-DD-YYYY'))) = 1;
```

WM_EQUALS ตรวจสอบว่าช่วงเวลาแรก เท่ากับช่วงเวลาที่สอง หรือไม่



รูปที่ ข-4 WM_EQUALS

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_EQUALS(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1990 ถึงวันที่ 1 มกราคม 2005 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
SELECT * FROM employees e
WHERE WM_EQUALS(e.wm_valid,
        wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
        TO_DATE('01-01-2005', 'MM-DD-YYYY'))) = 1;
```

WM_LESSTHAN ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรกน้อยกว่า (ก่อน)ช่วงเวลาที่สอง หรือไม่

รูปที่ ข-5 WM_LESSTHAN

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_LESSTHAN(ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

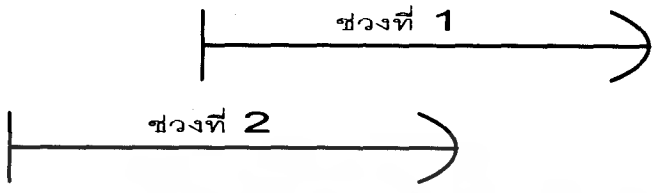
ตัวอย่างการใช้คำสั่ง เช่น ต้องการทราบว่าพนักงานคนไหนที่ทำงานในวันที่ 1 มกราคม 1995 รูปแบบการใช้คำสั่ง SQL ดังนี้

```
WM_LESSTHAN(TO_DATE('01-01-1980', 'MM-DD-YYYY'),TO_DATE('01-01-1990', 'MM-DD-
YYYY'),TO_DATE('01-01-1991', 'MM-DD-YYYY'),TO_DATE('01-01-1992', 'MM-DD-YYYY') =
1
```

ตัวอย่างการใช้คำสั่ง

```
SELECT * FROM employees e
WHERE WM_LESSTHAN(e.wm_valid,
        wm_period(TO_DATE('01-01-2010', 'MM-DD-YYYY'),
        TO_DATE('01-02-2010', 'MM-DD-YYYY'))) = 1;
```

WM_GREATERTHAN ตรวจสอบว่าเวลาสิ้นสุดของช่วงเวลาแรกมากกว่า (หลัง)ช่วงเวลาที่สอง หรือไม่



รูปที่ ข-6 WM_GREATERTHAN

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_GREATERTHAN (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

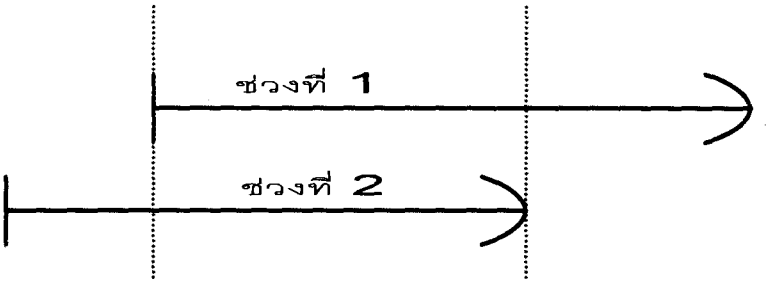
```
WM_GREATERTHAN(TO_DATE('01-01-1980', 'MM-DD-YYYY'),
        TO_DATE('01-01-1990', 'MM-DD-YYYY')
        TO_DATE('01-01-1970', 'MM-DD-YYYY'),
        TO_DATE('01-01-1980', 'MM-DD-YYYY') = 1
```

ตัวอย่างการใช้คำสั่ง

```
SELECT * FROM employees e
WHERE WM_GREATERTHAN(e.wm_valid,
        wm_period(TO_DATE('01-01-2001', 'MM-DD-YYYY'),
        TO_DATE('01-02-2001', 'MM-DD-YYYY'))) = 1;
```

2. ตรวจสอบความสำพันธ์ระหว่างสองช่วงเวลา ที่มีผลลัพธ์เป็นค่าตามสัมพันธ์ที่ตรวจสอบหากมีความสัมพันธ์นั้นอยู่ และจะได้ผลลัพธ์เป็น Null เมื่อไม่มีความสัมพันธ์นั้นอยู่ ได้แก่

WM_INTERSECTION จะเปรียบเทียบช่วงเวลาสองช่วงเวลา จะได้ผลลัพธ์เป็นช่วงเวลาที่เป็นจุดตัดของทั้งสองช่วงเวลา และจะได้ผลลัพธ์เป็น Null เมื่อไม่มีความสัมพันธ์นั้นอยู่



รูปที่ ข-7 WM_INTERSECTION

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_GREATERTHAN (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

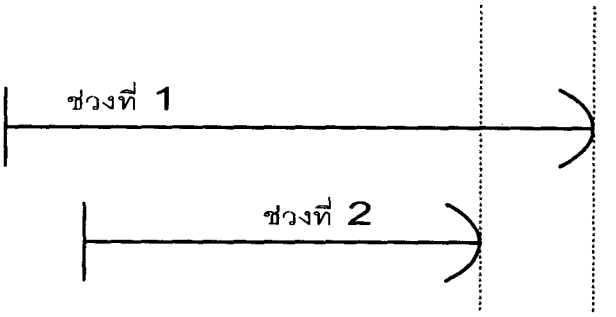
WM_LDIFF จะเปรียบเทียบช่วงเวลาสองช่วงเวลาว่ามีการทับซ้อนกันอยู่หรือไม่ จะได้ผลลัพธ์เป็นช่วงเวลาตั้งแต่เวลาเริ่มต้นของช่วงเวลาแรกจนถึงเวลาเริ่มต้นของช่วงเวลาที่สองหากช่วงเวลาแรกเกิดก่อนช่วงเวลาที่สอง และจะได้ผลลัพธ์เป็น Null เมื่อช่วงเวลาที่สองเกิดก่อนช่วงเวลาแรกหรือไม่มีความสัมพันธ์นั้นอยู่

รูปที่ ข-8 WM_LDIFF

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_LDIFF (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

WM_RDIFF จะเปรียบเทียบช่วงเวลาสองช่วงเวลาว่ามีการทับซ้อนกันอยู่หรือไม่ จะได้ผลลัพธ์เป็นช่วงเวลาที่แตกต่างระหว่างสองช่วงเวลาในส่วน ตั้งแต่เวลาสิ้นสุดของช่วงเวลาที่สองจนถึงเวลาสิ้นสุดของช่วงเวลาแรก และจะได้ผลลัพธ์เป็น Null เมื่อช่วงเวลาแรกสิ้นสุดก่อนช่วงเวลาที่สองหรือไม่มีความสัมพันธ์นั้นอยู่



รูปที่ ข-9 WM_RDIFF

โดยมีรูปแบบของการใช้คำสั่งดังนี้

WM_RDIFF (ช่วงเวลาที่ 1,ช่วงเวลาที่ 2)

การสืบค้นและการแก้ไขข้อมูล

ในหัวข้อนี้เราจะกล่าวถึงลักษณะ และการพิจารณาในการสอบถามต่างๆ (queries) และการใช้ภาษาในการจัดการข้อมูล (insert, update, delete) ที่เกี่ยวกับ Valid Time

6 การคืบค้น ข้อมูล (Query)

การสอบถามต่างๆ เราจะเห็นเฉพาะข้อมูลที่อยู่ในช่วงเวลาที่เรากำหนดให้กับ Session ของตารางเท่านั้น (ตั้งค่าโดยการใช้โพรซิเดอร์ SetValidTime) โดยเราสามารถ Operators ที่ได้กล่าวมาในส่วนที่ มาช่วยในการสอบถามร่วมกับภาษา SQL ได้

7 การแก้ไขข้อมูลด้วย Data Manipulation Language (DML)

ในการแก้ไขข้อมูลด้วย DML (INSERT,UPDATE,DELETE) ในตารางที่ได้ทำ version-enabled ให้สนับสนุน valid time แล้วนั้น เราสามารถกำหนดระยะเวลาให้กับ session ของผู้ใช้แต่ละคนว่าสามารถแก้ไขข้อมูลได้ในช่วงเวลาไหน หรือผู้ใช้สามารถกำหนดเองได้โดยใช้ API SetValidTime โดยคำสั่งจะมีผลเฉพาะช่วงเวลาที่กำหนดเท่านั้น ซึ่งโดยปรกติแล้วจะอยู่ในช่วงเวลาปัจจุบันจนกว่าผู้ใช้ทำการกำหนดช่วงเวลาให้กับ Session ใหม่ โดยการแก้ไข้ข้อมูลยังสามารถใช้

ร่วมกับ Operators ที่ได้กล่าวไว้ในหัวข้อที่ 4นอกเหนือจากนี้ยังมีส่วนที่ช่วยจัดการให้เป็นที่สนับสนุน Valid Time มีดังนี้

8 การสนับสนุนในส่วนของการกฎการบังคับความถูกต้องของข้อมูลใน Valid Time

ในหัวข้อนี้จะเสนอถึงกฎบังคับความสมบูรณ์ของข้อมูลโดย Workspace Manager จะเป็นผู้จัดการกับกฎเหล่านี้แทนผู้ใช้

Referential Integrity Constraints ในที่นี้คือ กฎที่ใช้บังคับสำหรับการอ้างอิงข้อมูล ซึ่งสำหรับการทำ version-enabled ให้สนับสนุน Valid Time นั้นจะแบบออกได้เป็น 2 กรณีคือ

2 กรณีที่ทั้งตารางที่เป็นคีย์หลักเป็นตารางที่ถูกทำ version-enabled ให้สนับสนุน Valid Time แล้ว ตารางที่อ้างถึงข้อมูลต้องเป็นตารางที่ถูกทำ version-enabled ให้สนับสนุน Valid Time แล้วเท่านั้น และข้อมูลที่อ้างถึงนั้นต้องอยู่ในช่วงเวลาของคีย์หลักเท่านั้น

3 กรณีที่ตารางที่เป็นคีย์หลักเป็นตารางที่ไม่ได้ทำ version-enabled ให้สนับสนุน Valid Time แล้ว จะไม่สนใจเวลาสามารถอ้างถึงข้อมูลนั้นได้เลย

Unique Constraints ในที่นี้คือกฎที่บังคับว่าห้ามคีย์หลักซ้ำกันในช่วงเวลาหรือจุดเวลาเดียวกัน ตารางที่ทำให้สามารถสนับสนุน Valid Time ด้วย Workspace Manager นั้นต้องจะกำหนดคีย์หลักหรือกำหนดให้ห้ามมีค่าซ้ำ (Unique) ใน Column ใด Column หนึ่งเสมอ จะไม่ยอมให้มีแถวใดๆ มีคีย์หลักมีค่าซ้ำกันในช่วงเวลาเดียวกัน เช่น หากกำหนดให้รหัสพนักงานเป็นคีย์หลัก ในการบันทึกข้อมูลพนักงานให้รหัสพนักงานมีค่าซ้ำกับข้อมูลที่มีอยู่แล้วและช่วงเวลามีการเหลื่อมล้ำกับข้อมูลเดิม จะไม่สามารถทำได้เนื่องจากมีข้อมูลซ้ำในช่วงเวลาเดียวกัน แต่หากข้อมูลนั้นอยู่คนละช่วงเวลาจะสามารถบันทึกหรือแก้ไขข้อมูลได้ เป็นต้น

ตัวอย่างง่ายๆ สำหรับการสร้างตารางให้สนับสนุนน Valid Time โดยมีขั้นตอนดังนี้

สร้างตาราง โดยตัวตารางนั้นจะต้องกำหนด คีย์หลัก (Primary key) ไว้ด้วย ยกตัวอย่างเช่น ตารางที่ใช้เก็บข้อมูลขอชื่องพนักงานและเงินเดือนของแต่ละคน

ทำ Version-enables (โดยมีรูปแบบของคำสั่งดังตัวอย่าง) ให้ตารางสามารถสนับสนุน Valid Time โดยจะเพิ่ม column ชื่อว่า WM_VALID ให้โดยอัตโนมัติ โดย คอลัมน์นี้จะมีชนิดข้อมูล WM_PERIOD ซึ่ง รายละเอียดจะได้อธิบายในหัวข้อถัดไป รูปแบบของคำสั่งที่จะทำ Version-enables จะแสดงดังรูป

Insert ข้อมูลลงในตาราง โดยต้องกำหนดชื่อ เงินเดือน และ ช่วงเวลาของข้อมูล

กำหนดช่วงเวลาให้กับฐานข้อมูลว่าจะให้แสดงข้อมูลในช่วงเวลาไหน

แก้ไขข้อมูลเงินเดือนและช่วงเวลาของข้อมูล

ทำ Disables versioning (โดยมีรูปแบบของคำสั่งดังตัวอย่าง) ให้กับตาราง ในขั้นตอนนี้เราจะทำหรือไม่ทำก็ได้ หากไม่ทำการ Disables versioning ผู้ใช้จะไม่สามารถ ลบตาราง นี้ได้

ตัวอย่าง ข-2 การสนับสนุน Valid Time

```
-- Create a very simple employees table (deliberately oversimplified
-- for purposes of illustration).
CREATE TABLE employees (name VARCHAR2(16) PRIMARY KEY,salary NUMBER);

-- Version-enable the table. Specify TRUE for valid time support.
EXECUTE DBMS_WM.EnableVersioning ('employees', 'VIEW_WO_OVERWRITE', FALSE,
TRUE);

INSERT INTO employees VALUES('Adams',30000,WMSYS.WM_PERIOD(TO_DATE('01-01-
1990', 'MM-DD-YYYY'),TO_DATE('01-01-2005', 'MM-DD-YYYY')));

INSERT INTO employees VALUES('Baxter',40000,WMSYS.WM_PERIOD(TO_DATE('01-01-
2000', 'MM-DD-YYYY'), DBMS_WM.UNTIL_CHANGED));

INSERT INTO employees VALUES('Coleman',50000,WMSYS.WM_PERIOD(TO_DATE('01-01-
2003', 'MM-DD-YYYY'),TO_DATE('12-31-9999', 'MM-DD-YYYY')));
COMMIT;
-- Set valid time period to virtually all time.
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-1900', 'MM-DD-YYYY'),TO_DATE('01-
01-9999', 'MM-DD-YYYY'));

-- Operators for Workspace Manager Valid Time Support
-- WM_CONTAINS
SELECT * FROM employees e
```

```
WHERE WM_CONTAINS(e.wm_valid,wm_period(TO_DATE('01-01-1995', 'MM-DD-
YYYY'),TO_DATE('01-02-1995', 'MM-DD-YYYY'))) = 1;

-- WM_OVERLAPS
SELECT * FROM employees e
WHERE WM_OVERLAPS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2000', 'MM-DD-YYYY'))) = 1;

-- WM_MEETS
SELECT * FROM employees e
WHERE WM_MEETS(e.wm_valid,
wm_period(TO_DATE('01-01-2005', 'MM-DD-YYYY'),
TO_DATE('01-01-2006', 'MM-DD-YYYY'))) = 1;

-- WM_EQUALS
SELECT * FROM employees e
WHERE WM_EQUALS(e.wm_valid,
wm_period(TO_DATE('01-01-1990', 'MM-DD-YYYY'),
TO_DATE('01-01-2005', 'MM-DD-YYYY'))) = 1;

-- WM_LESSTHAN
SELECT * FROM employees e
WHERE WM_LESSTHAN(e.wm_valid,
wm_period(TO_DATE('01-01-2010', 'MM-DD-YYYY'),
TO_DATE('01-02-2010', 'MM-DD-YYYY'))) = 1;

-- WM_GREATERTHAN
SELECT * FROM employees e
WHERE WM_GREATERTHAN(e.wm_valid,
wm_period(TO_DATE('01-01-2001', 'MM-DD-YYYY'),
```

```
O_DATE('01-02-2001', 'MM-DD-YYYY'))) = 1;

-- WM_INTERSECTION
SELECT e.name, WM_INTERSECTION(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM employees e;

-- WM_LDIFF
SELECT e.name, WM_LDIFF(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM employees e;

-- WM_RDIFF
SELECT e.name, WM_RDIFF(e.wm_valid,
wm_period(TO_DATE('01-01-1995', 'MM-DD-YYYY'),
TO_DATE('01-02-1995', 'MM-DD-YYYY')))
FROM employees e;

-- Update the salary for an existing employee. Perform "sequenced" update, so
-- that existing time-related information is preserved. This results in two rows
-- for Baxter.
-- First, set valid time to the intended range for Baxter's raise.
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-2003', 'MM-DD-
YYYY'),DBMS_WM.UNTIL_CHANGED);

-- Give Baxter a raise, effective 01-Jan-2003 until changed.
UPDATE employees SET salary = 45000 WHERE name = 'Baxter';
COMMIT;
-- Disable versioning. By default (keepWMValid parameter value of TRUE),
-- the WM_VALID column is kept, with all its data.
```

```
EXECUTE DBMS_WM.DisableVersioning ('employees');
```

Update Operations

การ Update ในตารางที่ทำ version-enabled เพื่อสนับสนุน valid time มีสองลักษณะคือ sequenced Update และ nonsequenced Update

Sequenced update เป็นการแก้ไขข้อมูลโดยไม่ได้เจาะจองแก้ไขช่วงเวลาของข้อมูลใน column WM_VALID ในคำสั่ง UPDATE ใน sequenced update ค่าของ WM_VALID.ValidTill จะถูกเปลี่ยนเป็นเวลาที่ถูกเปลี่ยน ส่วน WM_VALID.ValidFrom จะไม่ถูกเปลี่ยนแปลง และใส่ข้อมูลใหม่และช่วงเวลาใหม่ลงไปอีกหนึ่งแถวโดยช่วงเวลาจะต่างจากช่วงเวลลาเดิม แต่การ updates แบบ Sequenced updates นั้นต้องแน่ใจว่าจะไม่เกิดความซ้ำซ้อนของข้อมูลเนื่องจากช่วงเวลาจะเป็นช่วงเวลาใหม่หากข้อมูลเก่ากับข้อมูลใหม่มีค่าเท่ากันก็จะสามารถบันทึกข้อมูลนั้นลงในตารางได้ แต่จะทำให้เกิดความซ้ำซ้อนของข้อมูล

ตัวอย่าง ข-2 การทำ Sequenced Update

```
-- Update the salary for an existing employee. Perform "sequenced" update, so
-- that existing time-related information is preserved. This results in two rows
-- for Baxter.
-- First, set valid time to the intended range for Baxter's raise.
EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-2003', 'MM-DD-YYYY'),
DBMS_WM.UNTIL_CHANGED);

-- Give Baxter a raise, effective 01-Jan-2003 until changed.
UPDATE employees SET salary = 45000 WHERE name = 'Baxter';

-- Set valid time to encompass virtually all time.

EXECUTE DBMS_WM.SetValidTime(TO_DATE('01-01-1900', 'MM-DD-YYYY'), TO_
DATE('01-02-9999', 'MM-DD-YYYY'));

-- See what data exists for Baxter.
SELECT * FROM employees WHERE name = 'Baxter';
```

NAME SALARY

---------- ----------

WM_VALID(VALIDFROM, VALIDTILL)

Baxter 45000

WM_PERIOD('01-JAN-2003 12:00:00 -04:00', NULL)

Baxter 40000

WM_PERIOD('01-JAN-2000 12:00:00 -04:00', '01-JAN-2003 12:00:00 -04:00')

Nonsequenced update เป็นการแก้ไขข้อมูลโดยเจาะจงเวลาของข้อมูลใน column WM_VALID ในคำสั่ง UPDATE ในการ Update แบบนี้จะไม่มีการสร้างแถวใหม่ให้สำหรับข้อมูลที่ต้องการ Update ข้อควรระวังคือ ผู้ใช้ต้องมั่นใจว่าการ Update จะมีผลกระทบกับแถวที่มีคีย์หลักเดียวกันแค่แถวเดียวเท่านั้น เนื่องจากอาจจะทำให้เกิดการแก้ไขข้อมูลที่ขัดกับกฎบังคับความถูกต้องของข้อมูล และถ้าเกิดในกรณีนี้ขึ้นการแก้ไขข้อมูลจะไม่สมบูรณ์ตามที่ต้องการได้

Insert Operations เมื่อผู้ใช้ Insert ข้อมูลลงในตารางที่ทำ version-enabled ผู้ใช้สามารถกำหนดช่วงเวลา Valid Time ได้ทั้งที่เป็น อดีต ปัจจุบัน และ อนาคต หรือหากถ้าผู้ใช้ใส่เป็น Null ช่วงของ Valid Time จะถูกกำหนดให้เป็นช่วงเดียวกันกับช่วงเวลาที่เรากำหนดให้กับ Session Database Workspace Manager จะตรวจสอบว่าการ Insert นั้นได้ละเมิดกฎข้อบังคับความถูกต้องของข้อมูลหรือไม่

ตัวอย่าง ข-2 แสดงการ Insert ข้อมูลผิดพลาดเนื่องจากมีช่วงเวลาเลื่อมล้ำกัน

-- Insert. Should violate primary key constraint, because of overlapping times:

-- existing Coleman row is valid from 01-Jan-2003 until 31-Dec-9999.

INSERT INTO employees VALUES('Coleman',55000,

WMSYS.WM_PERIOD(TO_DATE('01-01-2004', 'MM-DD-YYYY'),

TO_DATE('12-31-9999', 'MM-DD-YYYY'))

);)

*

ERROR at line 6:

ORA-20010: unique key violation

ORA-06512: at "WM_DEVELOPER.OVM_INSERT_10", line 1

ORA-04088: error during execution of trigger 'WM_DEVELOPER.OVM_INSERT_10'

To make the statement in Example 3–14 succeed, first change the WM_VALID.ValidTill attribute for the Coleman row to a timestamp reflecting 01-Jan-2004

or an earlier date.

ภาคผนวก ค.

Oracle JDeveloper 10*g* Release 3 (10.1.3) Installation Guide

1. Hardware Specification ที่แนะนำ

- Operating System Windows 2000-Service Pack 4

  Windows NT-Service Pack 6a

  Windows XP-Service Pack 2

- CPU Type and Speed Pentium IV 2 GHz ขึ้นไป
- Memory 1 GB RAM
- Display 65536 colors, set ขั้นต่ำ 1024 X 768 resolutions
- Hard Drive Space Base Installation: 375 MB

  Complete Installation: 500 MB
- Java SDK Sun J2SE 1.5.0_05 สำหรับ Windows

2. วิธีการติดตั้ง Oracle JDeveloper บน Windows

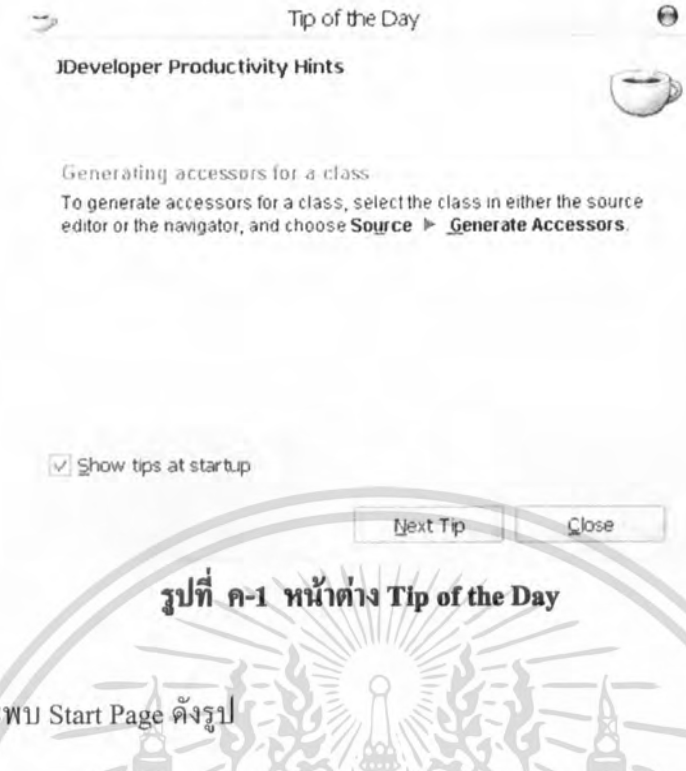
2.1 Copy folder ชื่อ Oracle JDeveloper 10g Release 3 (หรือทำการ download ได้จาก web ของ oracle) ไปไว้ที่ Drive C:

2.2 run file jdevW.exe ดังรูป 

ที่อยู่ในโฟลเดอร์ C:\Oracle JDeveloper 10g Release 3\jdev\bin\ เพื่อเข้าใช้งานโปรแกรม

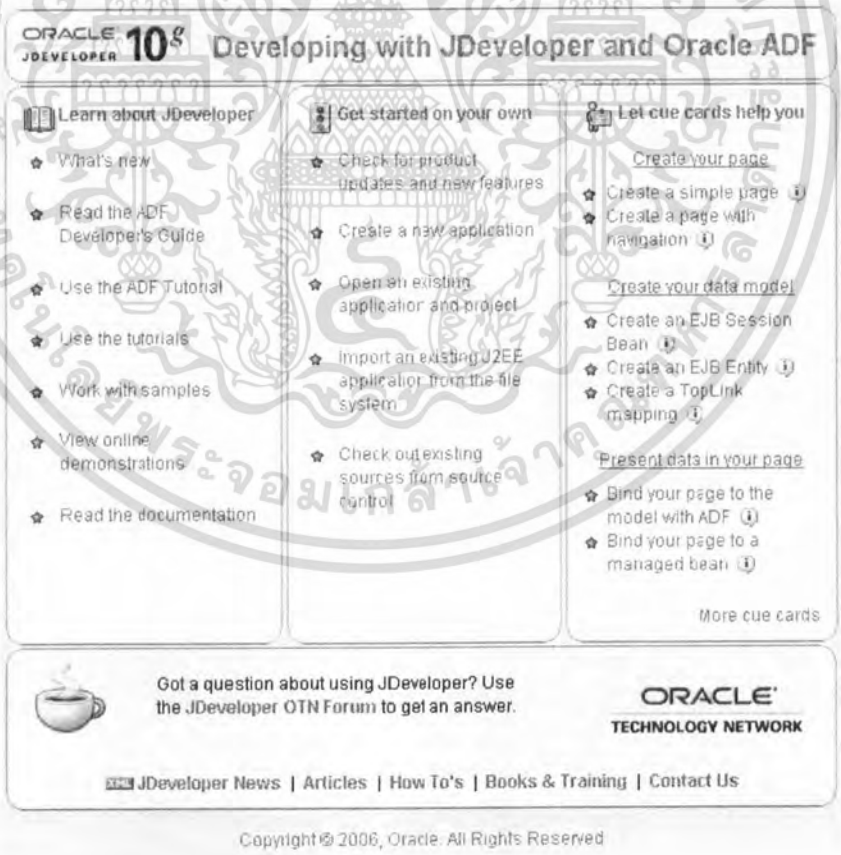
หมายเหตุ: เพื่อความสะดวกในการใช้โปรแกรมครั้งต่อๆไปอาจทำเป็น shortcut ของ file ไว้ที่ desktop

2.3 หากเข้า program สำเร็จจะแสดงหน้าต่าง Tip of the Day



รูปที่ ค-1 หน้าต่าง Tip of the Day

และจะพบ Start Page ดังรูป



รูปที่ ค-2 Start Page

ภาคผนวก ง.

โปรแกรม SpatioTemporalplus

1 System Requirement

Hardware Requirement ที่แนะนำ

CPU Type and Speed Pentium IV 2 GHz ขึ้นไป

Memory 1 GB RAM

Display 65536 colors, set ขั้นต่ำ 1024 X 768 resolutions

Hard Drive Space Base Installation: 50 MB

Complete Installation: 50 MB

Software Requirement

Operating System

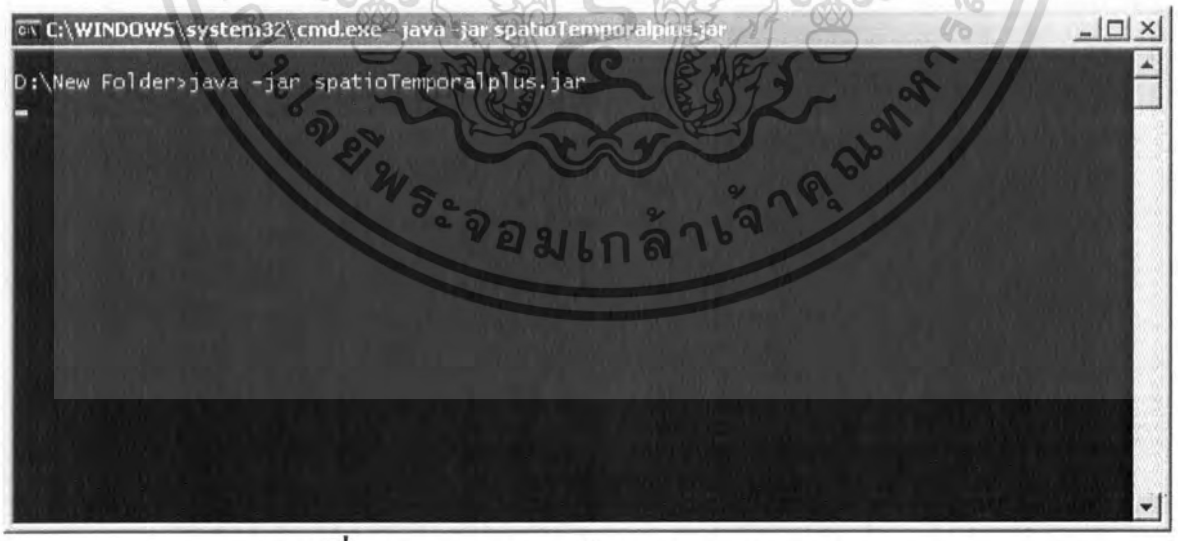
Windows 2000-Service Pack 4

Windows NT-Service Pack 6a Windows XP-Service Pack 2

Java SDK Sun J2SE 1.5.0_05 สำหรับ Windows

2 วิธีการเรียกใช้ SpatioTemporalplus บน Windows

2.1 SpatioTemporalplus เป็น Java Application ที่ไม่ต้อง ติดตั้งสามารถเรียกใช้งานได้โดยการ run command "java -jar SpatioTemporalplus.jar" ใน Windows Command ดังรูป



รูปที่ ง-1 run command ใน Windows Command

จะปรากฏโปรแกรมดังรูป

รูปที่ ง-2 SpatioTemporalPlus

3 ส่วนของการใช้งาน

3.1 การติต่อกับฐานข้อมูล

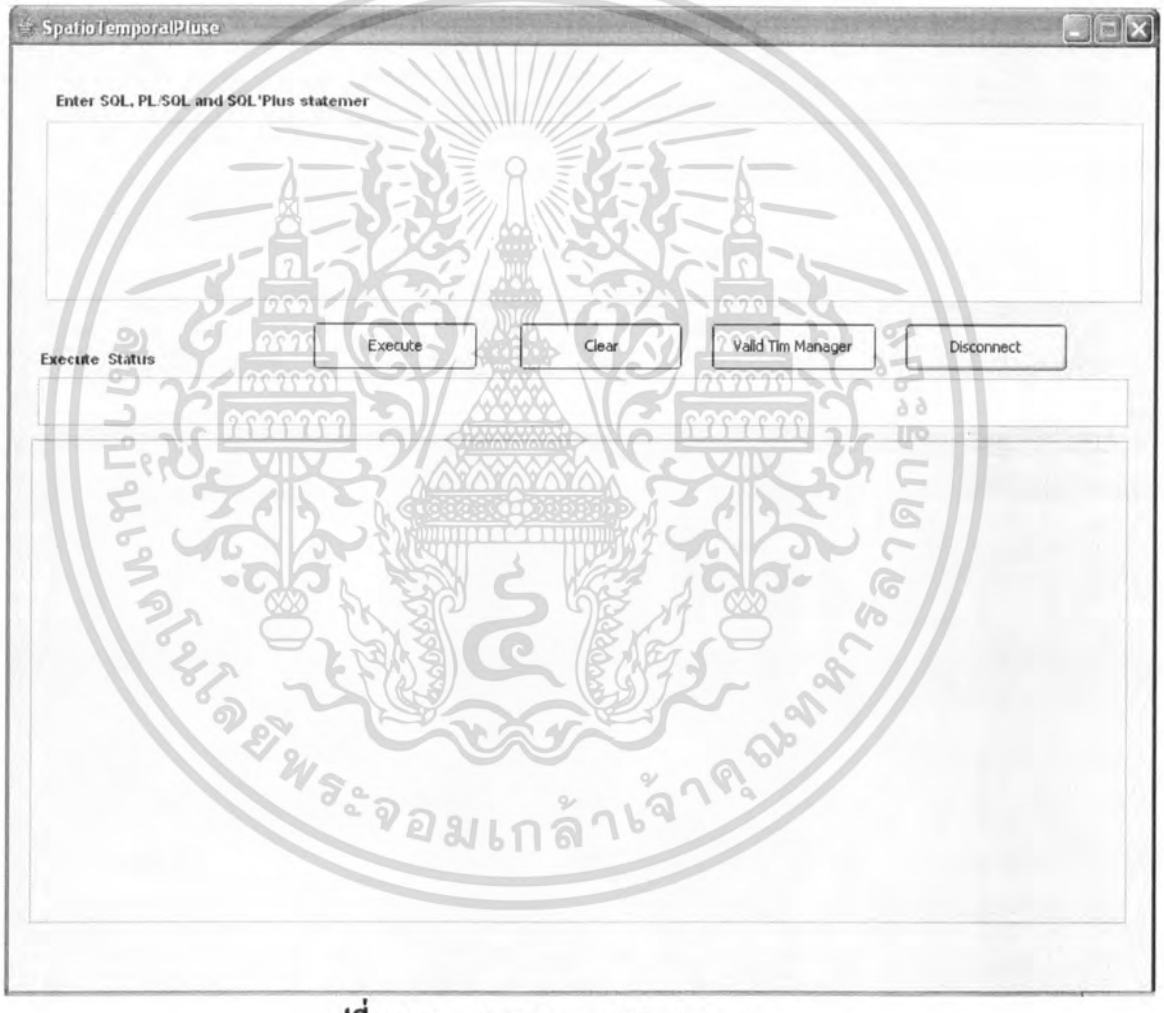
1.1.1 ผู้ใช้ต้อง Click ที่ปุ่ม Connect ในรูป ง-2 จะปรากฏหน้าต่างดังรูป ง-3

รูป ง-3 Connect

1.1.2 ให้กรอกข้อมูลให้ครบโดยมีข้อมูลดังนี้

- Hostname คือชื่อของเครื่องที่ต้องการติดต่อ
- Port คือหมายเลขพอร์ตที่ใช้ในการติดต่อ
- SID คือชื่อของ Database Instant ที่ต้องการติดต่อ
- Username คือชื่อผู้ใช้ในการติดต่อกับฐานข้อมูล
- Password คือรหัสผ่านในการติดต่อฐานข้อมูล

1.1.3 เมื่อกรอกข้อมูลครบแล้ว Click Connect จะปรากฏหน้าต่างดังรูปที่ ง-4

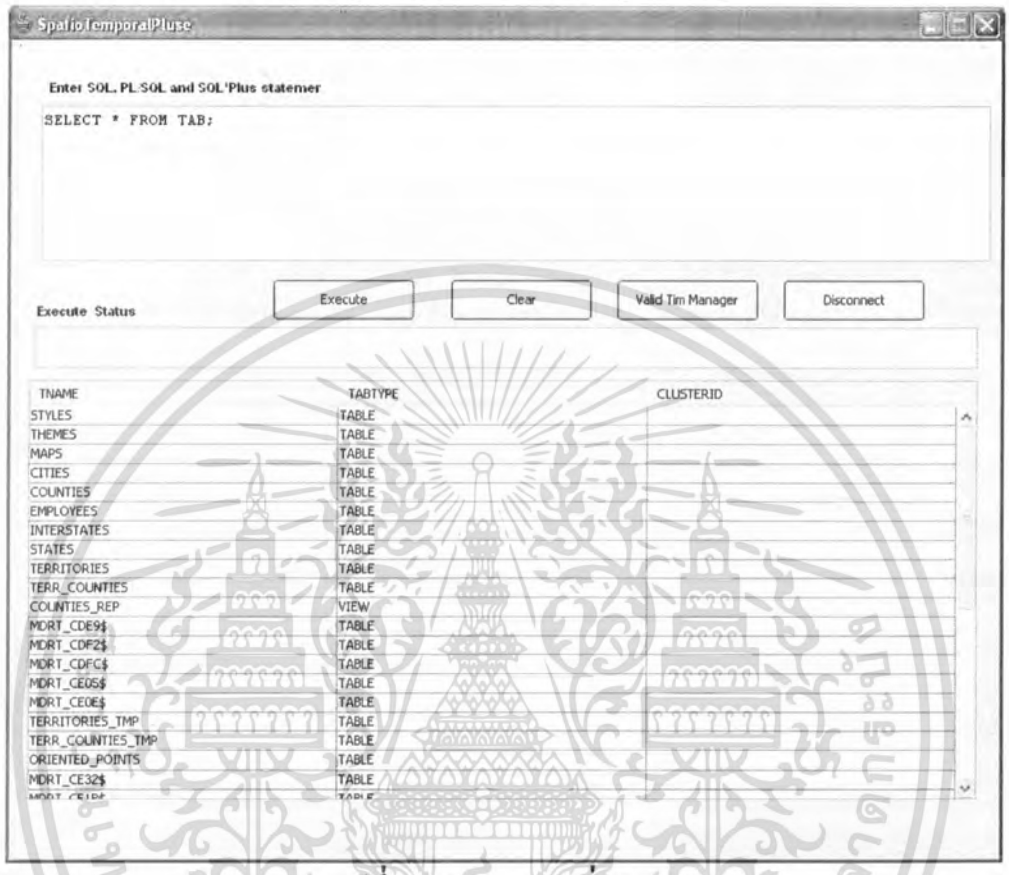


รูปที่ ง-4 SpatioTemporalPlus Home

1.2 การ Execute คำสั่ง SQL

1.2.1 ผู้ใช้สามารถส่งคำสั่งไปยังฐานข้อมูลโดยพิมคำสั่งลงใน ส่วนรับคำสั่ง SQL

1.2.2 แล้ว Click ปุ่ม Execute จะได้ผลดังรูปที่ ง-5



รูปที่ ง-5 Execute คำสั่ง SQL

1.3 การจัดการให้ตารางสนับสนุน Valid Time

1.3.1 Click ที่ปุ่ม Valid Time Manager จะปรากฏหน้าต่างเพื่อช่วยในการทำตารางให้สนับสนุน Valid Time ดังรูปที่ ง-6

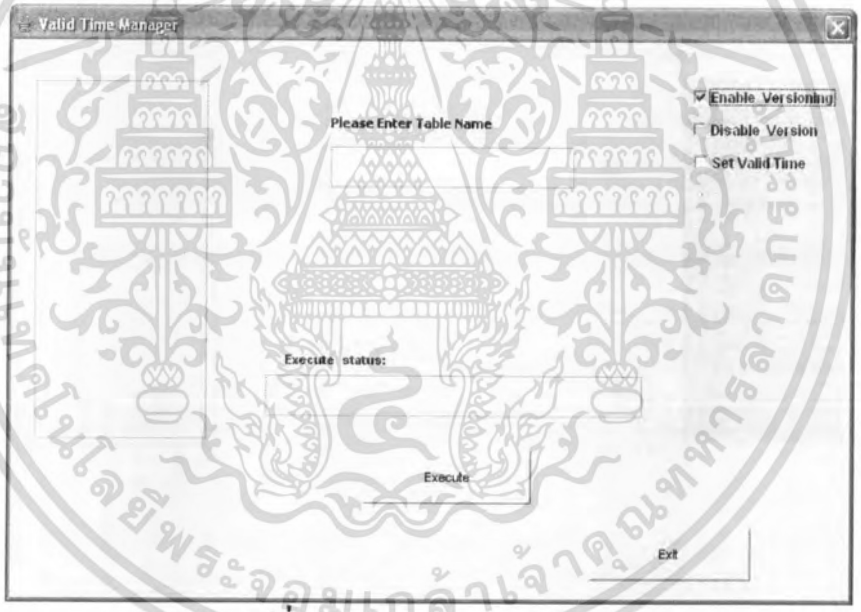
รูปที่ ง-6 Valid Time Manager

1.3.2 การ Enable Versioning

1.3.2.1 Click Enable Versioning

1.3.2.2 แล้วใส่ชื่อตารางที่เราต้องการให้สนับสนุน Valid Time

1.3.2.3 Click ที่ปุ่ม Execute ดัง รูปที่ ง-7



รูปที่ ง-7 Enable Versioning

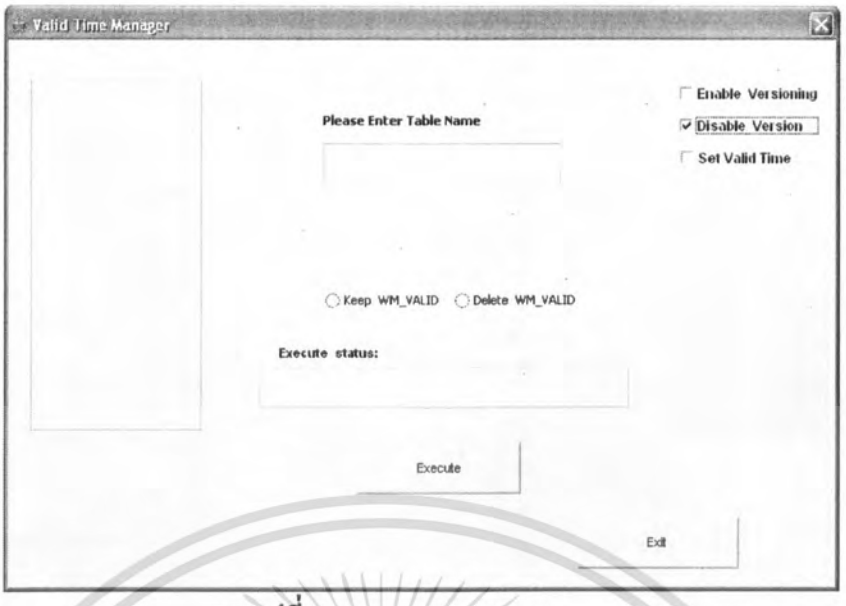
1.3.3 การ Disable Version

1.3.3.1 Click Disable Version

1.3.3.2 แล้วใส่ชื่อตารางที่เราต้องการ Disable Version

1.3.3.3 สามารถเลือกว่าจะทำการลบ column WM_VALID หรือไม่โดยเลือกที่เมนู

1.3.3.4 Click ที่ปุ่ม Execute ดัง รูปที่ ง-8



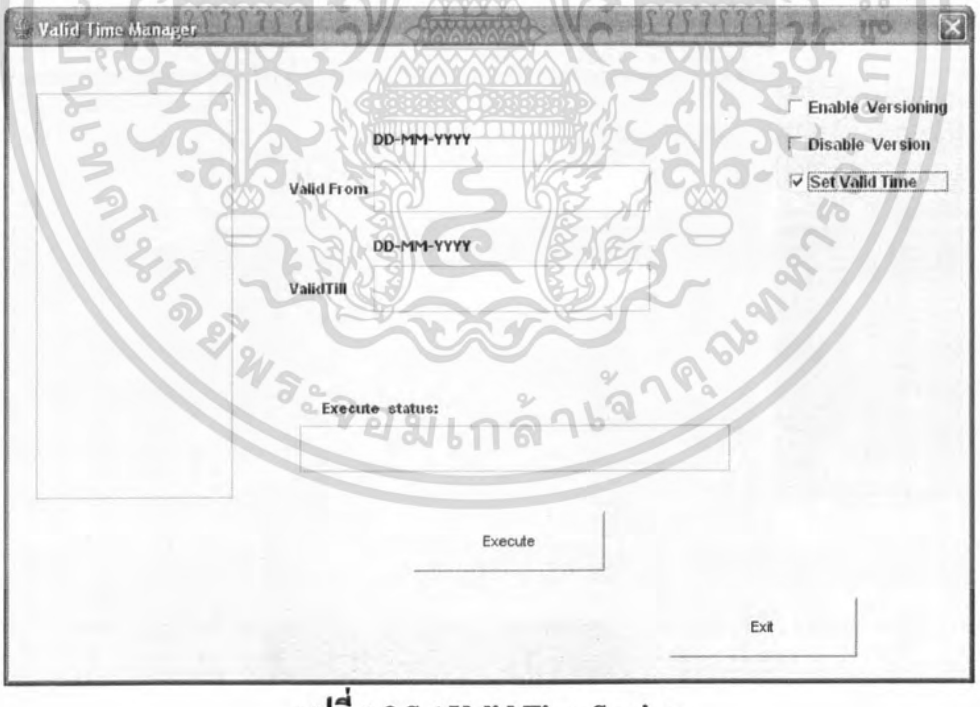
รูปที่ ง-8 Disable Version

1.3.4 การกำหนด Valid Time Session

1.3.4.1 Click Set Valid Time Session

1.3.4.2 ใส่วันเริ่มต้นและสิ้นสุด

1.3.4.3 Click ที่ปุ่ม Execute ดัง รูปที่ ง-9



รูปที่ ง-9 Set Valid Time Session