# ต้นแบบระบบการจัดการฐานข้อมูลเชิงเวลา โดยใช้แนวคิดระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ในออราเคิล

# A Temporal DBMS Prototype using ORACLE ORDBMS

โดย

**นายอิฏฐา เสขะนันทน์**

รหัส 45066023



อาจารย์ที่ปรึกษา

**รศ.ดร. ศุภมิตร จิตตะยโศธร**



รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2547
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

| | |
|---|---|
| **ชื่อหัวข้อ** | ต้นแบบระบบการจัดการฐานข้อมูลเชิงเวลา<br>โดยใช้แนวคิดระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ในออราเคิล |
| **นักศึกษา** | นายอิฏฐา เสขะนันทน์ |
| **อาจารย์ที่ปรึกษา** | รศ.ดร. ศุภมิตร จิตตะยโศธร |
| **ระดับการศึกษา** | วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ |
| **แขนงวิชา** | วิทยาการสารสนเทศ |
| **ปีการศึกษา** | 2547 |

## บทคัดย่อ


ระบบงานต่างๆ มีความจำเป็นในการเก็บข้อมูลด้านเวลาเพิ่มขึ้น โดยจะเก็บข้อมูลที่เป็นอดีต ปัจจุบัน และอนาคต โดยการจัดการข้อมูลเชิงเวลานั้นมีภาษาที่รองรับแนวคิดฐานข้อมูลเชิงเวลา คือ ภาษา TSQL ซึ่งในปัจจุบันระบบการจัดการฐานข้อมูลยังไม่มีการสนับสนุนภาษานี้ ถ้าหากจัดการแนวคิดเชิงเวลาโดยใช้ภาษา Relational SQL นั้น จะมีความซับซ้อนในการเขียนอย่างมาก ดังนั้นจึงได้พัฒนาต้นแบบระบบการจัดการฐานข้อมูลเชิงเวลาที่สนับสนุนภาษา TSQL ซึ่ง Implement ตามแนวคิดเชิงวัตถุสัมพันธ์ (ORDBMS) โดยใช้ฐานข้อมูล ORACLE

ในรายงานฉบับนี้ทำการศึกษาแนวคิดฐานข้อมูลเชิงเวลา โครงสร้างและรูปแบบคำสั่งภาษา TSQL และนำมาแปลงเป็นภาษา ORSQL โดยจะทำการ Implement ภาษา TSQL ในส่วนของ Valid time state Table เท่านั้น

| | |
|---|---|
| **Title** | A Temporal DBMS Prototype using ORACLE ORDBMS |
| **Student** | Mr. Ittha Sekhanan |
| **Advisor** | Assoc. Prof. Dr. Suphamit Jittajasothorn |
| **Level of Study** | Master of Science in Information Technology |
| **Major** | Information Science |
| **Academic Year** | 2004 |


## ABSTRACT

Nowadays the business systems require for the data collection that have more variety and complex moreover some systems need to store the data in a period of time in the past, present and future times that Temporal data management. The TSQL language is suitable for managing temporal data but currently DBMS are not support. The using of Relational SQL is very complex and difficult to manage temporal data, moreover the user must have the knowledge and ability to use SQL. But it is not suitable in the real world. So this paper would like to present the notion of ORDBMS for managing temporal database by creating a prototype to manage temporal database concept. This approach will change TSQL to ORDBMS order by using Oracle Database for implementation.

This paper studies the temporal database concept , syntax of TSQL and ORSQL in order to convert TSQL to ORSQL which implement TSQL only part of Valid time state table.

# กิตติกรรมประกาศ

โครงการพัฒนาระบบงานนี้สำเร็จลุล่วงได้เนื่องจากปัจจัยหลายประการด้วยกัน ซึ่งสิ่งแรกที่ขาดไม่ได้ คือ การได้รับความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.ศุภมิตร จิตตะยโสธร ที่ให้คำปรึกษา ข้อเสนอแนะ และให้ความรู้แก่ผู้เขียนมาโดยตลอด ผู้เขียนขอกราบขอบพระคุณท่านอาจารย์เป็นอย่างสูง

โครงการพัฒนาระบบนี้ ได้รับความช่วยเหลือด้วยดีมาโดยตลอดจากคุณวรสุดา ตันตสุรฤกษ์ และคุณจักรินทร์ วิบูลย์ศิลป์ จึงขอขอบคุณมา ณ โอกาสนี้

ขอขอบคุณ เพื่อนร่วมรุ่นที่คอยให้ความช่วยเหลือ สนับสนุน เป็นกำลังใจ และเสนอแนะในการศึกษาพัฒนาระบบงานนี้

ท้ายสุดผู้เขียนขอกราบขอบพระคุณบุคคลที่สำคัญที่สุดก็คือ บิดา มารดา ที่เลี้ยงดูและให้โอกาสได้ศึกษาเล่าเรียนอย่างเต็มที่ รวมถึงญาติๆ ทุกท่านที่ได้สนับสนุนและเป็นกำลังใจให้โดยตลอดจนกระทั่งได้พัฒนาระบบงานนี้จนเสร็จสมบูรณ์

นายอิฏฐา เสขะนันทน์

กันยายน 2547

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

หน้า

ตารางที่

# สารบัญรูป

หน้า

รูปที่

# สารบัญรูป (ต่อ)

หน้า

รูปที่

# บทที่ 1

# บทนำ

## 1.1 ความเป็นมาของปัญหา

ข้อมูลเชิงเวลามีความสัมพันธ์กับข้อมูลชนิดต่างๆ ที่เก็บในฐานข้อมูล ซึ่งระบบงานในปัจจุบันต้องอาศัยข้อมูลชนิดนี้เพิ่มมากขึ้น ข้อมูลเชิงเวลานั้นจะต้องถูกจัดเก็บลงในฐานข้อมูลตามรูปแบบของแนวคิดเชิงเวลาซึ่งจะจัดเก็บข้อมูลที่เปลี่ยนแปลงตามเวลาทั้งข้อมูลในอดีต ปัจจุบัน และอนาคต ถ้าใช้ภาษา SQL จัดการข้อมูลเชิงเวลานั้นจะมีความยุ่งยากมาก ซึ่งผู้ใช้จำเป็นที่ต้องมีความรู้และความสามารถในการใช้ภาษา SQL อย่างลึกซึ้ง จึงจำเป็นที่จะต้องมีภาษาที่ใช้ในการจัดการกับข้อมูลชนิดนี้รองรับ เพื่อสามารถใช้งานได้ง่ายขึ้น นั่นคือ ภาษา TSQL (Temporal SQL) ซึ่งได้มีการกำหนดมาตรฐานและรูปแบบไว้แล้ว แต่ยังไม่มีผู้ผลิตระบบการจัดการฐานข้อมูลรายใดนำความสามารถนี้เพิ่มเข้าไปในระบบการจัดการฐานข้อมูลของตน

การจัดการแนวคิดเชิงเวลาควรใช้ภาษาฐานข้อมูลเชิงเวลา (TSQL) (Snodgrass. 1998) ซึ่งเป็นภาษาที่ใช้จัดการ แนวคิดเชิงเวลา นั่นคือ ความสามารถที่จัดการข้อมูลประเภท Valid time , Transaction time และข้อมูลแบบ Bitemporal ซึ่งจะทำให้ง่ายในการจัดการข้อมูลแบบ Sequenced , Current และ Nonsequenced ซึ่งเป็นลักษณะพื้นฐานของข้อมูลเชิงเวลา ดังนั้นภาษา TSQL จะเพิ่ม Data type ที่ชื่อว่า Period เพื่อเก็บ ข้อมูลเวลา ซึ่งข้อมูลความละเอียดของเวลานั้นจะละเอียดมากแค่ไหนขึ้นอยู่กับความสามารถของ DBMS อย่างเช่น Period ( DATE) , Period (Second) เป็นต้น

จากปัญหาดังกล่าว คือ ยังไม่มีการนำภาษา TSQL มาใช้จริง ทำให้เกิดความยุ่งยากที่จะจัดการแนวคิดเชิงเวลาโดยใช้ภาษา SQL จึงขอนำเสนอแนวคิดการจัดการระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational DBMS) มาใช้ในการจัดการแนวคิดเชิงเวลาโดยการแปลงคำสั่ง TSQL ให้มาอยู่ในรูปแบบของแนวคิดเชิงวัตถุสัมพันธ์

## 1.2 วัตถุประสงค์ในการพัฒนาระบบงาน

1. เพื่อนำประโยชน์ที่ได้จากการใช้ฐานข้อมูลเชิงเวลามาพัฒนาระบบงานต่างๆ
2. เพื่อจัดการข้อมูลเชิงเวลาโดยใช้ภาษาฐานข้อมูลเชิงเวลาได้อย่างสมบูรณ์

3. เพื่อขจัดปัญหาของระบบการจัดการฐานข้อมูลในปัจจุบันซึ่งไม่สนับสนุนภาษาฐานข้อมูลเชิงเวลา

4. เพื่อเป็นแนวทางในการพัฒนาระบบฐานข้อมูลเชิงเวลาต่อไปในอนาคต

**1.3 ขอบเขตของการพัฒนาระบบ**

การพัฒนาต้นแบบระบบการจัดการฐานข้อมูลเชิงเวลาโดยใช้แนวคิด ORDBMS เพื่อจัดการข้อมูลเชิงเวลาโดยจำเป็นต้องกำหนด Syntax ขึ้นมาใหม่เพื่อรองรับแนวคิดการ Implement ในระดับ Attribute ซึ่งแตกต่างจากแนวคิดเดิมที่ Implement ในระดับ Record ซึ่งภาษาที่ใช้ในการจัดการนี้ได้ตั้งชื่อว่า KTSQL โดยที่ภาษานี้จะถูกแปลงเป็นภาษา ORSQL โดยใช้ฐานข้อมูลของ ORACLE ในการ Implement ระบบงานนี้ โดยจะทำการพัฒนาในส่วน Valid time เท่านั้น และจะไม่สนับสนุนการทำงานในส่วน Transaction Time

**1.4 ประโยชน์ที่คาดว่าจะได้รับ**

1. ผู้ใช้สามารถนำประโยชน์ของแนวคิดเชิงเวลามาจัดการข้อมูลได้อย่างง่ายดาย

2. ผู้ใช้สามารถใช้คำสั่ง KTSQL ที่ถูกกำหนดขึ้นใหม่ได้อย่างง่ายดายโดยไม่ต้องเขียนคำสั่ง Relational SQL หรือ คำสั่งภาษา Object Relational SQL ที่ซับซ้อน

3. การเรียกดูข้อมูลสามารถทำได้อย่างรวดเร็วเนื่องจากไม่จำเป็นต้องทำการ Join ตารางหากใช้ภาษา Relational SQL

4. เป็นแนวทางที่จะพัฒนาระบบฐานข้อมูลเชิงเวลาได้สมบูรณ์ขึ้น

**1.5 ขั้นตอนการศึกษา**

1. ศึกษาแนวคิดฐานข้อมูลเชิงเวลา
2. ศึกษาแนวคิดฐานข้อมูลเชิงวัตถุสัมพันธ์
3. ศึกษาการจัดการข้อมูลเชิงเวลาด้วยภาษา TSQL
4. ศึกษาการจัดการข้อมูลเชิงเวลาด้วยภาษา ORSQL

**1.6 แนวทางและเครื่องมือในการพัฒนาระบบ**

1. พัฒนาแอพพลิเคชันด้วย ภาษา JAVA โดยใช้ Jbuilder 9

2. ระบบจัดการฐานข้อมูล Oracle 9i Release 2
3. ระบบปฏิบัติการ Windows XP Professional

# บทที่ 2

# วรรณกรรมและทฤษฎีที่เกี่ยวข้อง

## 2.1 ฐานข้อมูลเชิงเวลา (Temporal Database) (Snodgrass. 1998)

"ฐานข้อมูลเชิงเวลา" เป็นฐานข้อมูลที่ใช้จัดเก็บข้อมูลที่เปลี่ยนแปลงตามเวลา เช่น ข้อมูลในอดีต ปัจจุบัน อนาคต ซึ่งฐานข้อมูลเชิงเวลาจะเก็บข้อมูลเวลาอยู่ 3 แบบ (วีรวดี ขุขันธิน. 2542) คือ

1. วาลิดไทม์ (Valid-time) คือ เวลาที่ข้อมูลเป็นจริง ซึ่งจะแบ่งแยกย่อยออกเป็น
   - Valid Time Start คือ เวลาเริ่มต้นที่ข้อมูลนั้นเป็นจริง
   - Valid Time End คือ เวลาสุดท้ายที่ข้อมูลนั้นเป็นจริง
2. ทรานแซคชั่นไทม์ (Transaction time) คือ เวลาที่ข้อมูลถูกเก็บใน Database โดยสามารถแบ่งออกเป็น
   - Transaction Time Start คือ เวลาที่เริ่มจัดเก็บข้อมูลลงในฐานข้อมูล
   - Transaction Time End คือ เวลาที่แก้ไขหรือลบข้อมูล
3. เวลาที่ผู้ใช้กำหนด (User-define time) คือ เวลาที่ผู้ใช้งานสร้างขึ้นเพื่อใช้งาน ซึ่งผู้ใช้งานต้องจัดการและเรียกค้นเอง ระบบฐานข้อมูลจะไม่จัดการให้อย่างอัตโนมัติ หรือกล่าวอีกนัยหนึ่งได้ว่าเวลาชนิดนี้จะ เป็นเวลาที่เป็นจริงเสมอ เช่น วันเกิด วันเข้าทำงาน เป็นต้น

**ตารางที่ 2.1** ตารางNICUStatus

| Name | Status | from_date | to_date |
|---|---|---|---|
| Alexis May | สาหัส | 19-11-2540 | 27-11-2540 |
| Kelsey Ann | สาหัส | 19-11-2540 | 26-11-2540 |
| Brandon James | สาหัส | 19-11-2540 | 26-11-2540 |
| Nathan Roy | สาหัส | 19-11-2540 | 28-11-2540 |
| Joel Steven | ฉุกเฉิน | 19-11-2540 | 20-11-2540 |

**ตารางที่ 2.1** ตารางNICUStatus (ต่อ)

| Name | Status | from_date | to_date |
|---|---|---|---|
| Joel Steven | สาหัส | 20-11-2540 | 26-11-2540 |
| Kenneth Robert | ดี | 21-11-2540 | 03-01-2541 |
| Alexis May | ดี | 27-11-2540 | 11-01-2541 |
| Alexis May | ดี | 02-12-2540 | 31-12-9999 |
| Alexis May | ดี | 02-12-2540 | 31-12-9999 |

ข้อมูลในตารางที่ 2.1 เป็นตารางวาลิดไทม์ (Valid-Time Table) ที่เก็บค่าและชื่อคนไข้ สถานะของคนไข้ วันที่เริ่มมีสถานะ และวันสิ้นสุดสถานะ จะเห็นได้ว่า วันที่ใน 2 แถวสุดท้ายในAttribute to date คือ 31-12-9999 ซึ่งใช้ใน SQL-92 แทนค่าปัจจุบัน(NOW) คือ เวลาที่ยังไม่สามารถระบุวันสิ้นสุดได้

การกำหนดและจัดการข้อมูลเกี่ยวกับเวลาที่เปลี่ยนแปลงอยู่เสมอนั้น สิ่งที่ต้องพิจารณาจะมีดังต่อไปนี้

ประเภทของความซ้ำซ้อน พิจารณารูปที่ 1 จะมีความซ้ำซ้อนประเภทต่างๆ อยู่ 4 ประเภท ดังนี้

1. *Nonsequenced Duplicate* คือ ความซ้ำซ้อนที่ค่าของทุกAttributeเหมือนกัน เช่น 2 แถวสุดท้ายของตาราง NICUStatus

2. *Value - Equivalent Duplicate* คือ ความซ้ำซ้อนที่ค่าของทุก Attribute เหมือนกันยกเว้น Attribute เวลา (Timestamp) เช่น 3 แถวสุดท้ายของตาราง NICUStatus

3. *Current Duplicate* คือ ความซ้ำซ้อนที่ค่าของทุกAttribute ยกเว้น Attribute เวลาจะมีค่าที่เหมือนกันในปัจจุบัน เช่น วันนี้คือวันที่ 6 มกราคม พ.ศ.2541 3 แถวสุดท้ายของตาราง NICUStatus จะมีความซ้ำซ้อนแบบปัจจุบัน ซึ่งจะมีค่าของ Attribute ซ้ำกันในวันนี้

4. *Sequenced Duplicate* คือ ความซ้ำซ้อนที่มีค่าของทุก Attribute ยกเว้น Attribute เวลาเหมือนกันอยู่ในช่วงระยะเวลาหนึ่งที่ซ้อนทับกัน เช่น 3 แถวสุดท้ายของตาราง NICUStatus

เราสามารถป้องกันการซ้ำซ้อนประเภทต่างๆ ที่เกิดขึ้นนี้ในภาษา Relational SQL ได้โดยขอยกตัวอย่างเพียงกรณีการป้องกันความซ้ำซ้อนแบบ Sequenced เท่านั้น

```
CREATE TABLE NICUStatus (
    Name CHAR(15),
    Status CHAR(8),
    FROM_DATE DATE,
    TO_DATE DATE
    CHECK (NOT EXISTS (SELECT L1.Name
            FROM NICUStatus AS L1
            WHERE 1 < (SELECT COUNT(Name)
            FROM NICUStatus AS L2
        WHERE L1.Name = L2.Name AND L1.Status = L2.Status
            AND L1.FROM_DATE <L2.TO_DATE
            AND L2.FROM_DATE < L1.TO_DATE)))
)
```

**การสร้างตาราง Valid – Time table**

การสร้างตาราง Valid – Time table นั้นจะเหมือนกับการสร้างตารางธรรมดาทั่วไป โดยจะแตกต่างเพียงในส่วนที่จะต้องระบุว่าตารางนั้นเป็นประเภท Valid - time ซึ่งเป็นสิ่งที่ง่ายมาก เพียงแค่ผู้ใช้มีความรู้พื้นฐานทาง SQL ก็สามารถเขียนภาษา TSQL ได้ ดังตารางที่ 2.2

**ตารางที่ 2.2 ตารางLOT_LOC**

| FDYD_ID | LOT_ID_NUM | PEN_ID | HD_CNT | FROM_DATE | END_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 |

**การสอบถามสถานะของตารางวาลิดไทม์ (Querying Valid – Time State Table)**

ข้อมูลเชิงเวลาและสถานะของข้อมูลจะถูกเก็บบันทึกไว้ในตาราง ณ เวลาใดเวลาหนึ่ง ซึ่งข้อมูลนั้นมีข้อเท็จจริงเป็นจริงในช่วงเวลานั้นเรียกว่า "Valid –Time State Table"

ลักษณะพื้นฐานของข้อมูลทางเวลามี 3 ชนิด ดังนี้

- Current : สถานะของข้อมูลในปัจจุบันซึ่งแต่ละข้อมูลจะมีได้เพียง 1 สถานะ
- Sequenced : ไม่มีเวลาใดเลยที่ข้อมูลจะมี 2 สถานะ
- Nonsequenced : ข้อมูลหนึ่งๆ จะไม่สามารถมี 2 สถานะพร้อมๆ กันได้ในช่วงเวลาเดียวกัน

จากตารางที่ 2.2 LOT_LOC นี้ จะประกอบด้วย Attribute ดังต่อไปนี้ **FDYD_ID** หมายถึง ลานเลี้ยงสัตว์ **LOT_ID_NUM** หมายถึง หมายเลขล็อท **PEN_ID** หมายถึง หมายเลขคอก **HD_CNT** หมายถึง จำนวนของวัว **FROM_DATE** หมายถึง วันที่เริ่มต้นที่เข้ามาอยู่ในคอก **TO_DATE** หมายถึง วันที่ออกจากคอก

**Temporal Projection and Selection**

การสอบถามเชิงเวลาจากตารางสามารถทำได้ 3 แบบ คือ

- Current : สอบถามข้อมูลที่เป็นจริงในปัจจุบัน
- Sequence : สอบถามข้อมูลที่มีเวลาเข้ามาเกี่ยวข้อง
- Nonsequenced : สอบถามข้อมูลทั้งหมดทั้งในอดีตและปัจจุบัน โดยไม่ให้ความสำคัญกับช่วงของเวลา

**Temporal Joins**

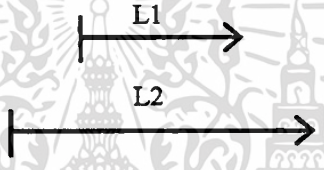
จะแสดงข้อมูลที่เป็นไปได้ทั้งหมด ซึ่งเกิดจากการเชื่อมโยงข้อมูลของ 2 Relations ที่มี Attributes ของ Timestamp บน Domain เดียวกัน Temporal Join จึงถูกพิจารณาถึงความสัมพันธ์ที่มากกว่าการ Join แบบ Non–Temporal ซึ่งนำไปสู่การสอบถามข้อมูล (Query) จากตารางข้อมูลในลักษณะของสมมุติฐาน ซึ่งการสอบถามนี้จะพิจารณาถึงการเชื่อมโยงกับตารางของตัวเอง (Self-Join) ร่วมด้วยการ Projection และ Selection โดยจะแบ่งเป็น 3 แบบ คือ

- Current : ข้อมูลที่ได้จะสนใจเฉพาะข้อมูลที่อยู่ในเวลาปัจจุบัน

- Nonsequenced : ข้อมูลที่ได้จะไม่ให้ความสำคัญกับเวลาของความสัมพันธ์
- Sequence : ข้อมูลที่ได้มีความสัมพันธ์กับเวลาซึ่งจะเป็นข้อมูลที่อยู่ในช่วงเวลาเดียวกัน

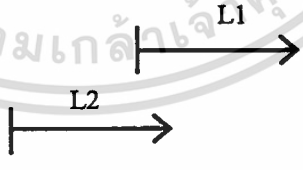
สามารถวิเคราะห์ถึงช่วงเวลาที่มีความสัมพันธ์กันของแถว L1 จากตาราง LOT_LOC ที่คาบเกี่ยวกับช่วงเวลาของแถว L2 จากตาราง LOT_LOC ได้ 4 กรณีดังต่อไปนี้

กรณีที่ 1 ระยะเวลาที่มีความสัมพันธ์ของแถว L1 ทั้งหมด จะอยู่ในช่วงของระยะเวลาที่มีความสัมพันธ์ของแถว L2 ซึ่งช่วงเวลาที่ยอมรับได้ (Period of validity) คือช่วงเวลาที่เกิดจากการ Intersection ของเวลาจากแถว L1 และ L2 ดังรูป



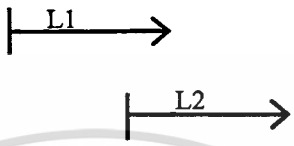

**รูปที่ 2.1** แสดงความสัมพันธ์ช่วงเวลาของแถว L1 ทั้งหมดอยู่ใน L2 ในกรณีที่ 1

กรณีที่ 2 ช่วงเวลาที่เรายอมรับได้ คือ ช่วงเวลาของ L1 Intersection กับ ช่วงเวลาของ L2 ซึ่งมีบางส่วนของช่วงเวลา L1 ที่สัมพันธ์กับช่วงเวลา L2 ซึ่งช่วงเวลาของ L2 เกิดก่อนช่วงเวลาของ L1 และสิ้นสุดก่อนช่วงเวลาของ L1 ดังรูป



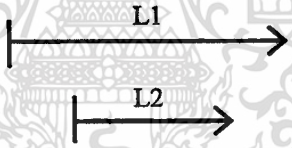

**รูปที่ 2.2** แสดงความสัมพันธ์ช่วงเวลาของแถว L1 สัมพันธ์กับใน L2 ในกรณีที่ 2

กรณีที่ 3 ช่วงเวลาที่เรายอมรับได้ คือ ช่วงเวลาของ L1 Intersection กับ ช่วงเวลาของ L2 ซึ่งมีบางส่วนของช่วงเวลา L1 ที่สัมพันธ์กับช่วงเวลาของ L2 ในกรณีที่ช่วงเวลาของ L1 เกิดก่อนช่วงเวลาของ L2 และสิ้นสุดก่อนช่วงเวลาของ L2 จะสิ้นสุดดังรูป




รูปที่ 2.3 แสดงความสัมพันธ์ช่วงเวลาของแถว L1 สัมพันธ์กับใน L2 ในกรณีที่ 3

กรณีที่ 4 ช่วงเวลาที่เรายอมรับคือช่วงเวลาของ L1 Intersection กับช่วงเวลาของ L2 โดยช่วงเวลาของ L2 ทั้งหมดอยู่ในช่วงเวลาของ L1




รูปที่ 2.4 แสดงความสัมพันธ์ช่วงเวลาของ L2 ทั้งหมดอยู่ใน L1 ในกรณีที่ 4

**การเปลี่ยนแปลงแก้ไขสถานะของตารางวาลิดไทม์ (Modifying Valid –Time State Tables)**

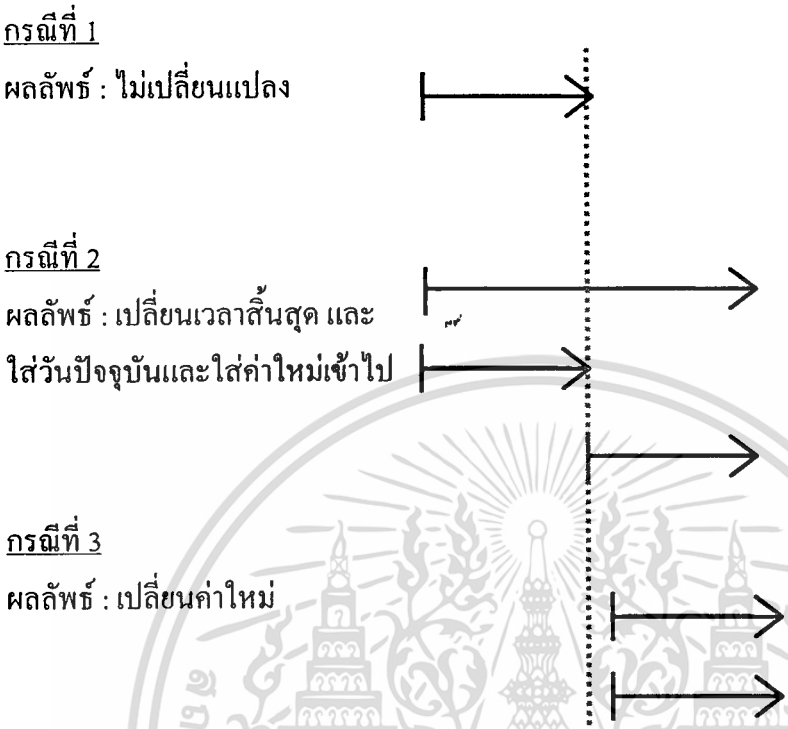
การเพิ่ม (Insert) การลบ (Delete) และการปรับปรุงแก้ไข (Update) ข้อมูลในฐานข้อมูลเชิงเวลาจะแตกต่างจากการกระทำกับข้อมูลทั่วไป เนื่องจากต้องคำนึงถึงช่วงเวลาของข้อมูลที่จะกระทำด้วย ซึ่งการพิจารณาการเปลี่ยนแปลงแก้ไขข้อมูล (Modify) จะกระทำต่อข้อมูลทั้ง 3 ลักษณะ คือ ข้อมูลปัจจุบัน (Current) ข้อมูลลำดับ (Sequenced) และข้อมูลไม่เรียงลำดับ (Nonsequenced)

Current Modification การเปลี่ยนแปลงสถานะเวลาของข้อมูลปัจจุบัน ประกอบด้วย การเพิ่ม (Insert) การลบ (Delete) และการปรับปรุงแก้ไข (Update) ข้อมูลในกรณีต่างๆ

- Current Insert การเพิ่มข้อมูลลงในฐานข้อมูล จะต้องกำหนดช่วงเวลาเริ่มต้นของข้อมูล (From_date) ที่จะเพิ่มเข้าไปให้เป็นเวลา ณ ปัจจุบัน(Now) และเวลาสิ้นสุด (To_date) เป็นเวลาในอนาคตที่ยังดำเนินต่อไปเรื่อยๆ (Forever)

- Current Delete การลบข้อมูลออกจากฐานข้อมูลซึ่งไม่นิยมกระทำ เนื่องจากการลบโดยไม่คำนึงถึงสถานะทางด้านเวลาของข้อมูล จะทำให้ข้อมูลที่เป็นปัจจุบัน และข้อมูลประวัติถูกลบออกจากฐานข้อมูลทั้งหมดเพราะฉะนั้นจึงใช้วิธีการ Update แทน

- Current Update การปรับปรุงหรือแก้ไขข้อมูลในฐานข้อมูล ซึ่งข้อมูลมีช่วงของเวลาที่ยังคงดำเนินต่อไปในอนาคต (Forever) จะต้องกำหนดเวลาที่สิ้นสุด(To_date)ให้กับข้อมูลใหม่เป็น ณ เวลาปัจจุบัน (Now) ดังนั้นการ Update จึงทำให้เกิดข้อมูลที่เป็นอดีตซึ่งสามารถค้นคืนกลับมาใช้งานได้ รูปแบบการ Update ที่ใช้กับข้อมูลที่เป็น Current มักกระทำในลักษณะที่ช่วงเวลาของข้อมูลยังดำเนินอยู่ การ Update ที่แถวของข้อมูลมีช่วงเวลาเริ่มต้น (From_date) อยู่ในอดีต (Past) และช่วงเวลาที่สิ้นสุด (To_date) ยังดำเนินอยู่ (Forever) เมื่อต้องการเปลี่ยนแปลงสถานะของข้อมูลจะต้องทำการ Insert แถวสถานะใหม่ของข้อมูลลงไป และกำหนดให้ช่วงเวลาเริ่มต้น(From_date) ณ เวลาปัจจุบัน(Now) เวลาสิ้นสุด (To_date) เป็นเวลาที่ดำเนินต่อไป (Forever) แล้วทำการ Update ข้อมูลที่ยังดำเนินต่อไปในอนาคตของข้อมูลเดิมออก โดยให้ช่วงเวลาสิ้นสุดของข้อมูล (To_date) เป็น ณ เวลาปัจจุบัน(Now) ซึ่งเป็นการทำให้ช่วงเวลาของข้อมูลสิ้นสุดในปัจจุบัน ดังรูป




**รูปที่ 2.5 แสดงการเปลี่ยนแปลงสถานะของข้อมูล**

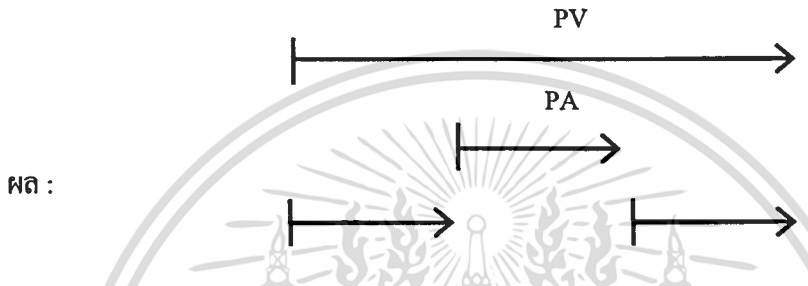
**Sequenced Modification**

การเปลี่ยนแปลงแก้ไขข้อมูลแบบ Sequenced โดยทั่วไปจะกระทำกับช่วงเวลาใดเวลาหนึ่งโดยเฉพาะซึ่งจะอยู่ในช่วงเวลาที่เหมาะสม (Period of Applicability : PA) ซึ่งประกอบไปด้วย ช่วงเวลาในอดีต ในอนาคต และมีส่วนที่คาบเกี่ยวกับเวลาในปัจจุบัน ดังนั้นช่วงเวลาเริ่มต้น (From_date) ที่เดิมเป็นเวลาในปัจจุบัน(Current) จะถูกแทนที่ด้วยช่วงเวลาเริ่มต้นของ PA และช่วงเวลาสิ้นสุด (To_date) ที่เดิมเป็นเวลาที่ยังคงดำเนินต่อไปในอนาคต (Date '9999 – 12 – 31') จะถูกแทนที่ด้วยช่วงเวลาที่สิ้นสุดของ PA สำหรับการเปลี่ยนแปลงแก้ไขข้อมูลแบบลำดับจะทำใน 3 ลักษณะ คือ การเพิ่มข้อมูล การลบข้อมูล การปรับปรุงแก้ไขข้อมูล

- Sequenced Insert เป็นการเพิ่มข้อมูล โดยมีการกำหนดช่วงเวลาของเวลาเริ่มต้นและเวลาสิ้นสุดให้อยู่ในช่วงที่เหมาะสมที่ต้องการพิจารณา ซึ่งสามารถ Insert แถวของข้อมูลนั้นลงในตาราง

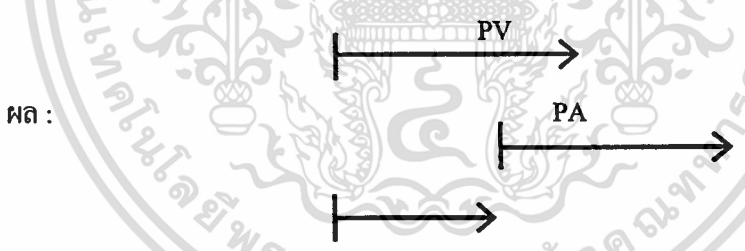
- Sequenced Delete เป็นการลบข้อมูลออกจากฐานข้อมูลซึ่งสามารถกระทำได้ 4 กรณี ซึ่งแต่ละกรณีจะกล่าวถึงแถวของข้อมูลเดิม ซึ่งก็คือช่วงเวลาที่ยอมรับได้ (Period of Validity : PV) พิจารณาร่วมกับ PA ที่ต้องการจะลบข้อมูล ซึ่งจะได้ผลลัพธ์โดยแสดงเป็นกรณีได้ ดังนี้

กรณีที่ 1



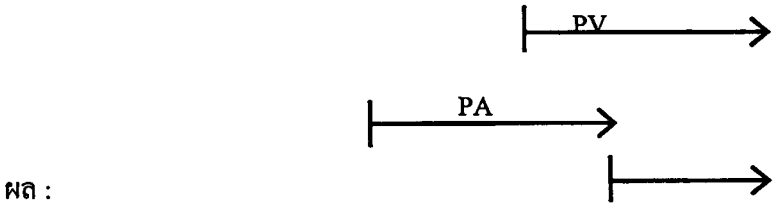

**รูปที่ 2.6** แสดงการลบข้อมูลกรณีที่ 1

กรณีที่ 2



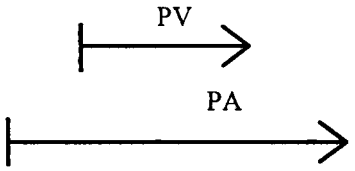

**รูปที่ 2.7** แสดงการลบข้อมูลกรณีที่ 2

กรณีที่ 3




**รูปที่ 2.8** แสดงการลบข้อมูลกรณีที่ 3

กรณีที่ 4



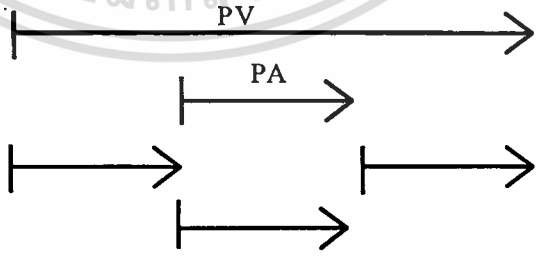

ผล : ทุก row จะถูกลบทั้งหมด

**รูปที่ 2.9** แสดงการลบข้อมูลกรณีที่ 4

ในแต่ละกรณีจะมีช่วงเวลาที่ยอมรับได้ (PV)และช่วงเวลาที่เหมาะสม (PA) สำหรับการลบข้อมูลของ PV ที่มีระยะเวลาตรงกับ PA จะถูกลบสำหรับในกรณีที่ 1 ผลลัพธ์ คือ ระยะเวลาเริ่มต้นและระยะเวลาสุดท้าย ในกรณีที่ 2 ผลลัพธ์ คือ ระยะเวลาช่วงแรกเท่านั้น ในกรณีที่ 3 ผลลัพธ์ คือ ระยะเวลาสุดท้ายเท่านั้น ในกรณีที่ 4 ผลลัพธ์ คือ ข้อมูลทั้งหมดจะถูกลบเนื่องจากระยะเวลาของ PA ครอบคลุมระยะเวลาของ PV ทั้งหมด

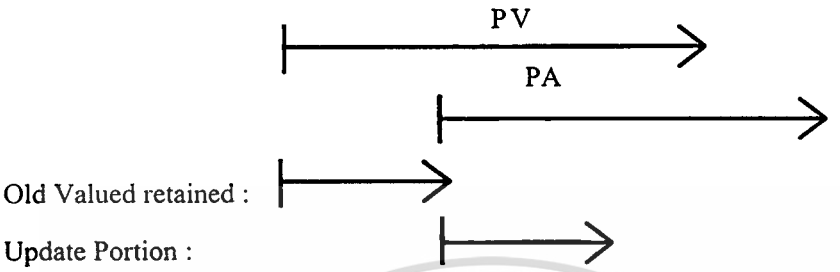
- Sequenced Update การปรับปรุงหรือแก้ไขข้อมูลในฐานข้อมูลจะต้องกำหนดเวลาเริ่มต้น (From_date) และเวลาสิ้นสุด (To_date ) ให้กับข้อมูลใหม่ ซึ่งสามารถแบ่งการกระทำได้ 4 กรณี ดังต่อไปนี้

กรณีที่ 1 การ Update จะทำที่ส่วนกลางของข้อมูล



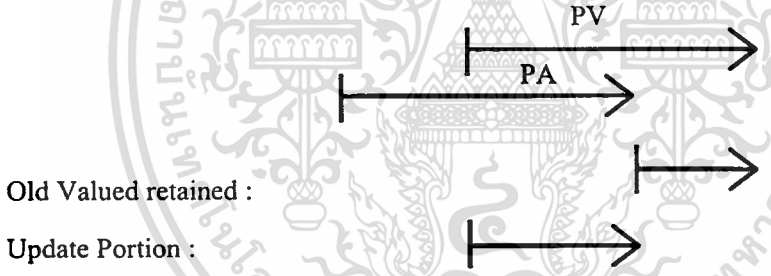

**รูปที่ 2.10** แสดงการ Update ข้อมูลกรณีที่ 1

กรณีที่ 2 การ Update จะทำที่ส่วนท้ายของข้อมูล



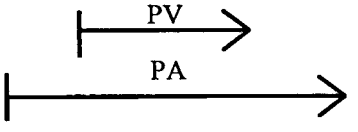

**รูปที่ 2.11** แสดงการ Update ข้อมูลกรณีที่ 2

กรณีที่ 3 การ Update จะกระทำที่ระยะแรกของข้อมูล




**รูปที่ 2.12** แสดงการ Update ข้อมูลกรณีที่ 3

กรณีที่ 4 การ Update จะทำกับทุกส่วนของข้อมูล




ผล : ทุก row จะถูก Update ทั้งหมด

**รูปที่ 2.13** แสดงการ Update ข้อมูลกรณีที่ 4

**Nonsequenced Modification**

การเปลี่ยนแปลงแก้ไขข้อมูลของข้อมูลแบบ Nonsequenced โดยทั่วไปจะไม่นิยมกระทำกับข้อมูลในฐานข้อมูลเชิงเวลา เพราะการที่จะทำการเพิ่ม การลบ และการปรับปรุงข้อมูล โดยไม่คำนึงถึงช่วงเวลาของข้อมูล จะทำให้เกิดความผิดพลาดของข้อมูลในฐานข้อมูลเชิงเวลาได้

**ภาษา TSQL ที่เสนอนี้จะมี keyword อยู่ด้วยกัน 3 คำด้วยกันคือ**

1. Sequenced หมายถึง การกระทำใดๆกับข้อมูลด้วยแนวคิดแบบ Sequenced จะมีการระบุ Keyword ไว้ด้านหน้าประโยค ดังนี้

- การ Insert จะระบุ Keyword ด้านหน้าประโยคว่า VALIDTIME PERIOD และตามด้วยช่วงเวลาที่ต้องการกำหนดเช่น '[1998-02-07 - 1998-02-18)' แล้วตามด้วยคำสั่ง SQL ในการ Insert
- การ Update จะระบุ Keyword ด้านหน้าประโยคว่า VALIDTIME PERIOD และตามด้วยช่วงเวลาที่ต้องการกำหนดเช่น '[1998-02-07 - 1998-02-18)' แล้วตามด้วยคำสั่ง SQL ในการ Update
- การ Delete จะระบุ Keyword ด้านหน้าประโยคว่า VALIDTIME PERIOD และตามด้วยช่วงเวลาที่ต้องการกำหนดเช่น '[1998-02-07 - 1998-02-18)' แล้วตามด้วยคำสั่ง SQL ในการ Delete

2. Current หมายถึง การกระทำใดๆกับข้อมูลด้วยแนวคิดแบบ Current ซึ่งในภาษา TSQL จะไม่มีการระบุ keyword นี้ จะเขียนเหมือนกับภาษา SQL -92

3. Nonsequenced Validtime หมายถึง การกระทำใดๆ กับข้อมูลโดยไม่คำนึงถึงเรื่องเวลาเข้ามาเกี่ยวข้องโดยระบุ Keyword ไว้หน้าประโยคว่า NONSEQUENCED VALIDTIME

**ตัวอย่างการใช้ภาษา TSQL (Snodgrass. 1998) จัดการกับฐานข้อมูลเชิงเวลา**

การสร้างตาราง Valid Time Table

เพื่อจัดเก็บข้อมูลเชิงเวลาจะใช้คำสั่งเหมือนกับภาษา SQL เพียงแต่ระบุคำว่า VALIDTIME และระบุความละเอียดที่ต้องการจัดเก็บโดยมีรูปแบบคำสั่งดังต่อไปนี้

```
Create Table LOT_LOC (
        FDYD_ID NUMBER,
        LOT_ID_NUM NUMBER,
        PEN_ID NUMBER,
```

```
        HD_CNT NUMBER
    ) As VALIDTIME  PERIOD (DATE)
```

คำสั่งนี้จะสร้างตารางที่ชื่อว่า LOT_LOC ขึ้นมา และตารางนี้จะมีคุณสมบัติของ Valid – time Table ตามที่ได้กำหนดไว้ ถ้าหากเราใช้คำสั่ง Relational SQL จะต้องเขียนคำสั่งหลายบรรทัด เพื่อป้องกันความซ้ำซ้อนหลายบรรทัด

การเพิ่มค่าในตาราง Valid Time Table แบบ Sequenced

```
VALIDTIME PERIOD '[1998-02-07 - 1998-02-18)'
INSERT INTO LOT_LOC (FDYD_ID, LOT_ID_ UM, PEN_ID, HD_CNT)
VALUES (1, 137, 1, 17)
```

การเพิ่มค่าในตาราง Valid Time Table แบบ Current

```
INSERT INTO LOT
VALUES (433, 'h')
```

การสอบถามตาราง Valid Time Table

- ปัจจุบันมีวัวอยู่กี่ตัวที่อยู่ใน Lot 219 และอยู่ในลานเลี้ยงสัตว์ที่ 1 อยู่ในคอกไหนบ้าง

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

- บอกประวัติว่ามีวัวจำนวนกี่ตัวมาจาก Lot 219 และลานเลี้ยงสัตว์ที่ 1 อยู่ในคอกไหนบ้างในช่วงเวลาที่เป็นไปได้

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

- มีจำนวนวัวกี่ตัวที่เคยมาจาก Lot 219 ในลานเลี้ยงสัตว์ที่ 1 อยู่ในคอกไหนบ้าง

```
NONSEQUENCED VALIDTIME SELECT PEN_ID, HD_CNT
FROM LOT_LOC
```

```
WHERE FDYD_ID = 1 AND LOT_ID_NUM = 219
```

- ปัจจุบัน Lot ไหนที่อยู่ในคอกเดียวกันในช่วงเวลาที่เป็นไปได้

```
VALIDTIME 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
AND L1.FDYD_ID = L2.FDYD_ID
AND L1.PEN_ID = L2.PEN_ID
```

- มี Lot ไหนที่อยู่ในคอกเดียวกันบ้าง

```
NONSEQUENCED VALIDTIME 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
AND L1.FDYD_ID = L2.FDYD_ID
AND L1.PEN_ID = L2.PEN_ID
```

- บอกประวัติว่ามี Lot ไหนบ้างที่อยู่ในคอกเดียวกันในช่วงเวลาที่เป็นไปได้

```
VALIDTIME SELECT L1.LOT, L2.LOT, L1.PEN_ID
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
```

การลบค่าในตาราง Valid Time Table แบบ Current

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

การลบค่าในตาราง Valid Time Table แบบ Sequenced

```
VALIDTIME PERIOD '[1998-10-01 - 1998-10-22)'
DELETE FROM LOT
```

```
WHERE LOT_ID_NUM = 234
```

การเปลี่ยนแปลงค่าในตาราง Valid Time Table แบบ Current

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

การเปลี่ยนแปลงค่าในตาราง Valid Time Table แบบ Sequenced

```
VALIDTIME PERIOD '[1998-03-01 - 1998-04-01)'
UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799
```

**2.2 ฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database)**

ฐานข้อมูลเชิงวัตถุสัมพันธ์ใน Oracle มีการนำเอาแนวคิดเชิงวัตถุมาใช้หลายอย่างด้วยกัน ซึ่งสามารถดูได้จาก (Kevin Loney and George Koch. 2000) ในเอกสารฉบับนี้จะขอกล่าวเพียงคุณสมบัติที่นำมาพัฒนาดังนี้

Nested Table เป็นการสร้าง Object ขึ้นมา เพื่อใช้เป็นต้นแบบและนำ Object นั้นมาสร้างเป็นตาราง และนำตารางนั้นมาใช้เป็น Attribute ฝังไว้ในตารางอีกตารางหนึ่ง โดยจะกลายเป็นตารางย่อยอยู่ภายในตารางที่สร้าง โดยสามารถเพิ่มข้อมูลเข้าไปได้โดยไม่มีข้อจำกัด

**ตัวอย่าง** ที่ใช้ในการสร้างความสัมพันธ์เชิงกลุ่ม แบบ Nested Table โดยการสร้าง Object ANIMAL_TY เป็นต้นแบบใช้เก็บข้อมูลของสัตว์ที่ผสมพันธุ์ ในตารางของผู้ทำการผสมพันธุ์ เริ่มจากสร้าง Object ANIMAL_TY ขึ้นมาโดยมี Attribute Breed หมายถึง ชนิดสัตว์เลี้ยง Name หมายถึง ชื่อสัตว์เลี้ยง เป็นข้อมูลที่เก็บตัวอักษรได้ไม่เกิน 25 ตัวอักษร และ Birthdate หมายถึง วันเกิดของสัตว์เลี้ยง เป็นชนิดข้อมูลประเภทวันเดือนปีโดยมีคำสั่งดังต่อไปนี้

```
Create or replace type ANIMAL_TY as object(
Breed   VARCHAR2(25),
```

```
Name    VARCHAR2(25),
Birthdate        DATE);

/
```

ลำดับต่อมานำ Object ANIMAL_TY มาสร้างเป็นตารางใช้ชื่อว่า ANIMALS_NT โดยที่คำสั่งนี้ทำให้ตารางที่สร้างขึ้นนี้มีโครงสร้างเหมือนกับโครงสร้างของ Object ซึ่งมีคำสั่งดังนี้

```
Create type ANIMALS_NT as Table of ANIMAL_TY;

/
```

สร้างตาราง BREEDER โดยมี Attribute Breeder Name หมายถึง ชื่อเจ้าของสัตว์เลี้ยง เป็นข้อมูลที่เก็บตัวอักษรได้ไม่เกิน 25 ตัวอักษร และ Animals เป็นชนิดข้อมูลประเภท ANIMALS_NT โดยจะเก็บข้อมูลไว้ในตาราง Animal_NT_Tab โดยใช้คำสั่ง

```
Create table BREEDER(
BreederName    VARCHAR2(25),
Animals          ANIMALS_NT)  nested table Animals store as Animal_NT_Tab;
```

การจัดการเกี่ยวกับข้อมูลใน Nested Table สามารถจัดการได้ดังนี้ คือ

- **Insert** เป็นการเพิ่มข้อมูล โดยจะใช้คำสั่งดังต่อไปนี้ในการใส่ข้อมูลลงในตาราง BREEDER คำสั่งนี้จะใส่ข้อมูลในตาราง BREEDER โดยใส่ชื่อผู้ผสมพันธุ์ และใส่ชื่อสัตว์ที่ต้องการเก็บข้อมูล ซึ่งจะต้องระบุชื่อตารางที่ใช้เก็บ และจะต้องระบุชื่อ Object ที่ใช้สร้างตารางนั้นเพื่อกำหนดค่าลงไป ซึ่งจะได้ตารางดังรูปที่ 4

```
Insert into BREEDER values(
'Jane James', ANIMALS_NT(
            ANIMAL_TY ('dog' , 'Butch' ,  '31-MAR- 97'),
            ANIMAL_TY ('dog' , 'Rover' , 'Rover' ,  '05-JUU'),
            ANIMAL_TY ('dog' , 'Julio' ,  '10-JAN- 97')
));
```

**ตารางที่ 2.3** ตาราง Breeder

| BreederName | Animals | | |
|---|---|---|---|
| | Breed | Name | Birtdate |
| **Jane James** | **dog** | **Butch** | **05-JUN- 97** |
| | **dog** | **Rover** | **05-JUN- 97** |
| | **dog** | **Julio** | **10-JAN- 97** |

- **Querying** การเรียกดูข้อมูลจาก Nested Table สามารถเรียกดูโดยใช้คำสั่งได้ 2 รูปแบบดังต่อไปนี้

**แบบที่ 1** ใช้ Function The() เพื่อทำการเข้าถึงข้อมูลโดยมีคำสั่งดังต่อไปนี้

```
Select NT.Birthdate from the( Select Animals from Breeder
where BreederName = 'Jane James') NT
where NT.Name = 'Julio' ;
```

**แบบที่ 2** ใช้ Function Table() เพื่อทำการเข้าถึงข้อมูลโดยมีคำสั่งดังต่อไปนี้

```
Select NT.birthdate from breeder B , Table(B.animals)NT
where breedername = 'Jane James' and NT.name= 'Julio';
```

- **Update** การปรับปรุงหรือแก้ไขข้อมูลโดยใช้คำสั่งดังต่อไปนี้

```
Update The(Select animals from Breeder where Breedername = 'Jane James')
Set Name = 'Julia' where Name = 'Julio' ;
```

คำสั่งนี้จะแก้ไขข้อมูลในตาราง Breeder เพื่อเปลี่ยนชื่อสัตว์เลี้ยงจากชื่อ Julio เป็นชื่อ Julia โดยเจ้าของสัตว์เลี้ยง คือ Jane James

- **Delete** เป็นการลบข้อมูลโดยใช้คำสั่งดังต่อไปนี้

Delete from the(Select animals from Breeder where Breedername = 'Jane James') where Name = 'Butch';

คำสั่งดังกล่าวจะทำการลบสัตว์เลี้ยงที่ชื่อว่า Butch ซึ่งเป็นสัตว์เลี้ยงของ Jane James ออกจากตาราง Breeder

# บทที่ 3

## การออกแบบ KTSQL บน ORDBMS

**แนวคิดในการอิมพลีเมนต์ TSQL โดยใช้ ORSQL**

การนำ TSQL มาประยุกต์ใช้งานบนฐานข้อมูลเชิงวัตถุสัมพันธ์จะมีปัญหาที่ต้องจัดการอยู่หลายประการด้วยกัน คือ รูปแบบของภาษา TSQL จะมีลักษณะคล้ายคลึงกับภาษา SQL ทั่วไป โดยสามารถดูได้จากตัวอย่างภาษา TSQL(Snodgrass. 1998) ที่จะต้องมี Keyword เพื่อระบุว่าต้องการจะทำงานกับข้อมูลอย่างไร เช่น Sequenced Current หรือ Nonsequenced ซึ่งภาษา TSQL จะทำการ Implement อยู่ในระดับ Record หรือ Row เท่านั้น รายงานฉบับนี้จึงได้นำเสนอแนวคิดใหม่เพื่อใช้จัดการข้อมูลเชิงเวลา โดยกำหนดภาษาขึ้นมาชื่อ KTSQL เพื่อประยุกต์ใช้กับแนวคิด ORDBMS นี้จะต้องเปลี่ยนการอิมพลีเมนต์จากระดับ Record ให้เป็นระดับ Attribute ซึ่งจะเก็บสถานะของการเปลี่ยนแปลงค่าของ Attribute และการ Implement นี้ จะมีข้อจำกัด คือ จะไม่สามารถนำเอา Primary key มากำหนดเป็น Temporal Attribute เนื่องจาก Primary key ต้องไม่สามารถเปลี่ยนแปลงค่าได้

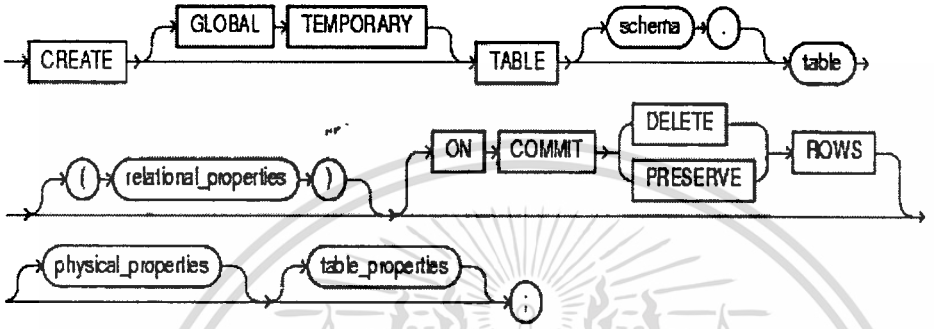
คำสั่ง KTSQL จะมีการเปลี่ยนแปลงจากเดิมเพียงบางส่วน เช่น อาจมี Keyword ระบุคำสั่งที่เปลี่ยนแปลงจากเดิม ซึ่งในรายงานฉบับนี้จะนำคำสั่งTSQL มาสร้างขึ้นใหม่เพื่อจัดการกับปัญหาข้างต้นที่กล่าวไว้แล้วโดยสร้างภาษา KTSQL ขึ้นมาและได้กำหนด Syntax ใหม่อีกด้วย ในเอกสารนี้จะทำการ Implement 5 คำสั่งหลัก คือ คำสั่ง Create , Insert , Update , Delete และ Select

### 3.1 การพัฒนาและออกแบบ Syntax ของ KTSQL

การพัฒนาและออกแบบ KTSQL บน ORDBMS ในเอกสารฉบับนี้ได้นำแนวคิดการจัดการฐานข้อมูลเชิงเวลาในส่วนของ Valid Time State Table มาพัฒนาทั้ง Sequenced และ Current เท่านั้น

**Create Statement ::=**

ผิดพลาด!




**รูปที่ 3.1** Create Statement

ในคำสั่ง Create นี้จะมีลักษณะเหมือนกับคำสั่งในภาษา SQL ทั่วไปซึ่งสามารถดูได้จาก (Oracle Corporation. 2004) ในเอกสารฉบับนี้จะขออธิบายในส่วนที่แตกต่างจากเดิมเท่านั้น คำสั่งนี้จะแตกต่างกัน คือ ส่วนของ relational_properties จะมี option ให้เลือกเพิ่มขึ้นมา คือ temporal_keyword

relational_propertie ::=

{ column datatype [temporal_keyword] [**DEFAULT** expr]

[column_ref_constraint] [column_constraint [column_constraint]...]

| { table_or_view_constraint | table_ref_constraint | supplemental_logging_props }

}

[, { column datatype [**DEFAULT** expr]

[column_ref_constraint] [column_constraint [column_constraint]...]

| { table_or_view_constraint | table_ref_constraint | supplemental_logging_props }

}

]...

temporal_keyword ::=

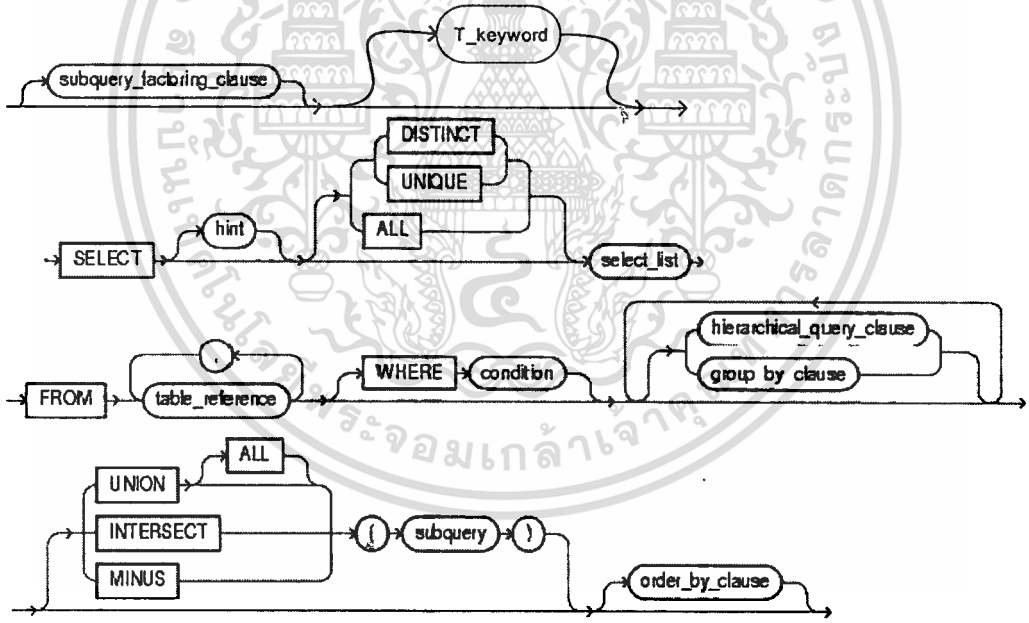
"validtime" | "curr"

คำสั่งที่ใช้สร้างตารางจะมีลักษณะคล้ายกับคำสั่งใน SQL-92 ทั่วไป แต่จะเพิ่ม Temporal_keyword ตามหลัง Attribute และ DataType ว่า Validtime หรือ Curr

ตัวอย่างคำสั่ง Create

```
CREATE TABLE LOT (
 LOT_ID_NUM INT,
 GNDR_CODE CHAR(1)  VALIDTIME
) ;
```

**Select Statement ::=**




**รูปที่ 3.2** Select Statement

ในคำสั่ง Select นี้จะแตกต่างกับคำสั่ง Select Statement ปกติใน Oracle 9i (Oracle Corporation. 2004) โดยเพิ่ม keyword ด้านหน้า คือ T_keyword ซึ่งเป็น Option ให้เลือกใส่ด้านหน้าคำสั่ง Select ได้ ซึ่งจะมีลักษณะเหมือนกันกับ TSQL (Snodgrass. 1998)

T_keyword ::=

"Validtime" [{period_t}] | "Nonsequenced ValidTime"

period_t ::=

"Period" " ' " "[" yyyy "-" mm "-" dd "-" yyyy "-" mm "-" dd ")" " ' "

คำสั่ง Select Statement ใน KTSQL นี้ จะสามารถเรียกดูข้อมูลได้อยู่ 3 ประเภทด้วยกัน คือ Sequenced Current และ Nonsequenced

1. ถ้าใช้ keyword Nonsequenced ValidTime เพื่อต้องการเรียกดูข้อมูลจากตารางโดยไม่สนใจช่วงเวลา
2. ถ้าใช้ keyword Validtime หมายถึง keyword ที่ต้องการเรียกดูข้อมูลที่เป็นแบบ Sequenced
3. ถ้าไม่ใส่ keyword ใดๆ เลย จะเป็นการเรียกดูข้อมูลแบบ Current

ตัวอย่างคำสั่ง Select แบบ Sequenced

Validtime Select GNDR_CODE From LOT where LOT_ID_NUM = 3 ;

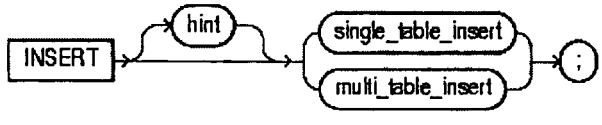
ตัวอย่างคำสั่ง Select แบบ Current

Select GNDR_CODE From LOT where LOT_ID_NUM = 3 ;

ตัวอย่างคำสั่ง Select แบบ Nonsequenced

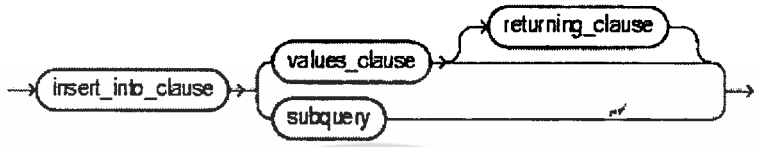
Nonsequenced ValidTime Select GNDR_CODE From LOT where LOT_ID_NUM = 3 ;

**Insert Statement (Oracle Corporat on. 2004)**




**รูปที่ 3.3** Insert Statement

Single_table_insert ::=




**รูปที่ 3.4** Single_table_insert

insert_into_clause ::=


**รูปที่ 3.5** insert_into_clause

ในคำสั่ง Insert นี้จะมีลักษณะเหมือนกับคำสั่งในภาษา SQL ทั่วไป ซึ่งสามารถดูได้จาก (Oracle Corporation. 2004) คำสั่งนี้จะแตกต่างกันในส่วนของ value_clause จะมีoption ให้เลือกเพิ่มขึ้นมา คือ period_t

values_clause ::=

VALUES ( { expr | DEFAULT }[{period_t}] [, { expr | DEFAULT } [period] ]... )

period_t ::=

“ ‘ ” “[” yyyy “-” mm “-” dd “-” yyyy “-” mm “-” dd “)” “ ‘ ”

ตัวอย่างเช่น '[2002-02-10 - 2002-03-02)'

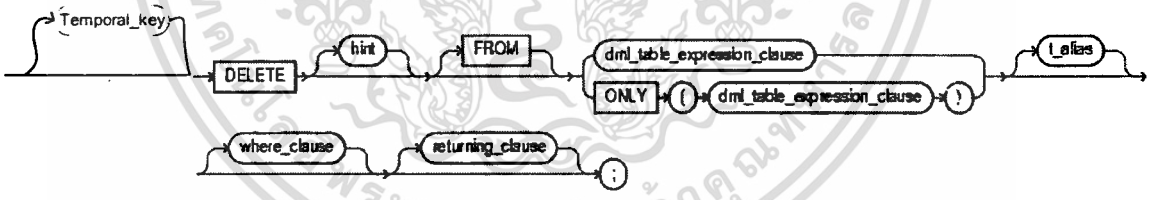
คำสั่งเพิ่มข้อมูลเข้าไปในตาราง Valid time state table จะมีลักษณะที่แตกต่างกับคำสั่ง TSQL เดิม โดยการระบุ period_t ไว้ด้านหลังค่าที่ต้องการ insert เข้าไปหลัง Temporal Attribute ถ้าเป็น Temporal Attribute แบบ Current จะไม่สามารถระบุช่วงเวลาเริ่มต้นและสิ้นสุดได้ ส่วนการ Insert Temporal Attribute ที่เป็นแบบ Sequenced สามารถทำได้ 2 แบบด้วยกัน คือ

1. สามารถระบุ period_t ด้านหลัง value ที่ต้องการ insert
2. ไม่ต้องระบุ period_t ด้านหลังเพื่อระบุว่าเป็นช่วงเวลาเริ่มต้นเป็น CURRENT_DATE และสิ้นสุด เป็น '9999-12-31'

ตัวอย่างคำสั่ง insert

```
INSERT INTO Patient
values(02, 'k.b' , 'boon' , 3 ,2 '[2002-02-10 - 2002-03-02)'
      ,2 '[2002-02-02 - 2002-03-02)' ) ;
```

**Delete Statement ::=**




**รูปที่ 3.6** Delete Statement

คำสั่งลบข้อมูลจากตาราง Valid time state table จะมีลักษณะเหมือนกับคำสั่ง TSQL (Snodgrass. 1998) เดิม โดยมี Temporal_key อยู่ด้านหน้าคำสั่ง Delete

Temporal_key ::=

"ValidTime" {period_t} | "Nonsequenced ValidTime"

period_t ::=

"Period" " ' " "[" yyyy "-" mm "-" dd "-" yyyy "-" mm "-" dd ")" " ' "

ตัวอย่างเช่น '[2002-02-10 - 2002-03-02)'

คำสั่ง Delete นี้ จะทำการลบข้อมูลออกจากตาราง Valid time state table อยู่ 3 ประเภทด้วยกัน คือ

1. ถ้าต้องการ Delete ข้อมูลออกแบบ Sequenced จะต้องระบุ keyword ว่า VALIDTIME PERIOD '[yyyy- mm- dd - yyyy-mm-dd)' อยู่ด้านหน้าคำสั่ง Delete

2. ถ้าต้องการ Delete ข้อมูลออกแบบ Current ไม่ต้องระบุช่วงเวลาซึ่งจะเหมือนกับคำสั่ง SQL ทั่วไป

3. ถ้าต้องการ Delete ข้อมูลออกแบบ Nonsequenced ซึ่งไม่สนใจช่วงเวลาจะระบุ keyword ว่า Nonsequenced ValidTime อยู่ด้านหน้าคำสั่ง Delete

การ Delete Temporal Attribute จะต้องระบุเงื่อนไข key Attribute ที่ไม่ใช่ Temporal Attribute และ Temporal Attribute ที่ต้องการลบด้วย เช่น ให้ key Attribute คือ id ระบุเงื่อนไขว่าต้องมี id หมายเลข 3 และ Temporal Attribute คือ doctor ระบุเงื่อนไขว่า doctor หมายเลข 3 เท่านั้น

ตัวอย่างคำสั่ง Delete แบบ Sequenced

```
VALIDTIME PERIOD '[1998-10-01 - 1998-10-22)'
Delete from patient where doctor=3 and id =3;
```

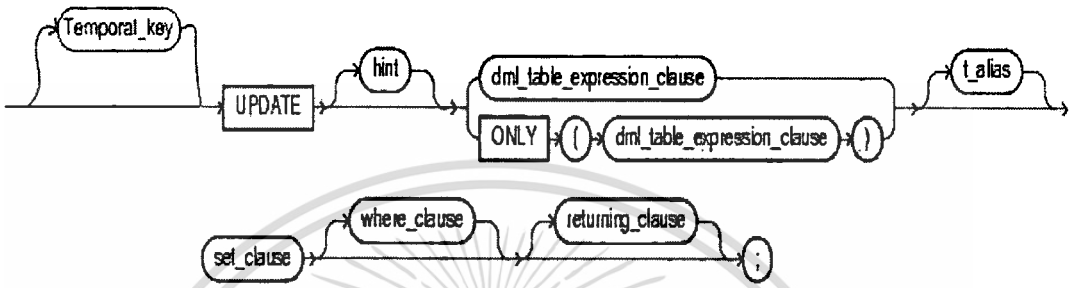
ตัวอย่างคำสั่ง Delete แบบ Current

```
Delete from patient where doctor=3 and id =3;
```

ตัวอย่างคำสั่ง Delete แบบ Nonsequenced

```
Nonsequenced ValidTime Delete from patient where doctor=3 and id =3;
```

**Update Statement ::=**




**รูปที่ 3.7** Update Statement

คำสั่งแก้ไขข้อมูลในตาราง Valid time state table จะมีลักษณะคล้ายกับคำสั่ง TSQL เดิม แต่จะแตกต่างกันในส่วนของ set_clause สามารถที่จะลบข้อมูลเชิงเวลาในระดับ Attribute ได้ทั้งแบบ Sequenced และ Current

set_clause ::=
SET {
{ ( column [, column]... ) = ( subquery ) | column = { expr | ( subquery ) | DEFAULT }
| column { period_t } ={ expr | ( subquery ) | DEFAULT } }
[, { ( column [, column]... ) = ( subquery ) | column = { expr | ( subquery ) | DEFAULT } } ]...
| VALUE ( t_alias ) = { expr | ( subquery ) }
}

Temporal_key ::=
"ValidTime" {period_t} | "Nonsequenced ValidTime"

period_t ::=

“Period” “ ‘ ” “[” yyyy “-” mm “-” dd “-” yyyy “-” mm “-” dd “)” “ ‘ ”

ตัวอย่างเช่น '[2002-02-10 - 2002-03-02)'

ตัวอย่างคำสั่ง Update แบบ Sequenced สำหรับ 1 Temporal Attribute

```
VALIDTIME    PERIOD '[1998-03-01 - 1998-04-01)'
UPDATE LOT
SET GNDR_CODE = 's'
WHERE LOT_ID_NUM = 799;
```

ตัวอย่างคำสั่ง Update แบบ Sequenced สำหรับหลาย Temporal Attribute ซึ่ง Temporal Attribute ชื่อ doctor เราสามารถระบุค่าช่วงเวลาใหม่ได้ และ Temporal Attribute Room จะมีค่าช่วงเวลาตามที่ระบุก่อนหน้า keyword UPDATE

```
VALIDTIME PERIOD '[1998-03-01 - 1998-04-01)' UPDATE patient
SET doctor PERIOD '[1998-02-07 - 1998-02-18)'= 2
,sname='sekhanan' ,room = 200
WHERE  id=01 and room <02 and doctor =1;
```

ตัวอย่างคำสั่ง Update แบบ Nonsequenced จะไม่สนใจช่วงเวลาใดๆใน Temporal Attribute

```
NONSEQUENCED  ValidTime  UPDATE patient
SET doctor= 2
WHERE id=01 and room <02 and doctor =1;
```

ตัวอย่างคำสั่ง Update แบบ Nonsequenced จะไม่สนใจช่วงเวลาใดๆ ใน Temporal Attribute และสามารถระบุช่วงเวลาที่ต้องการแก้ไข โดยไม่สนใจช่วงเวลาที่มีผลกระทบในเรื่อง Duplicate ที่กล่าวแล้วข้างต้น

```
NONSEQUENCED ValidTime UPDATE patient
SET doctor PERIOD '[1998-02-07 - 1998-02-18)'= 2
WHERE id=01 and  room  <  02  and  doctor  = 1 ;
```

ตัวอย่างคำสั่ง Update แบบ Current สำหรับ Temporal Attribute ที่ต้องการ Update แบบ Current Attribute เดียว

```
UPDATE patient set room = 2 ;
```

ตัวอย่างคำสั่ง Update แบบ Current จะทำการเช็คว่า Temporal Attribute เป็นแบบ Sequenced หรือเป็นแบบ Current เพื่อทำการ Update Current ให้เหมาะสมกับข้อมูล

```
UPDATE patient
 SET doctor = 2 ,
sname='sekhanan' ,
room = 200
WHERE id=01 and room < 02 and doctor =1 ;
```

## 3.2 ความต้องการของระบบ

### 3.2.1 Functional Requirement

- รองรับ Syntax ภาษา KTSQL เท่านั้น
- สามารถใช้คำสั่งภาษา KTSQL ได้ 5 คำสั่ง คือ Select , insert , delete , update , create
- สามารถแสดงผลเมื่อส่งคำสั่งสืบค้นข้อมูลได้
- สามารถ Clear Screen ได้โดยใช้คำสั่ง CLR

### 3.2.2 Non Functional Requirement

- CPU มีความเร็วตั้งแต่ 500 MKz ขึ้นไป
- หน่วยความจำที่ใช้จะต้องมากกว่า 256 MB ขึ้นไป
- พื้นที่ใน Hard Disk มากกว่า 6 MB
- ก่อนเริ่มใช้ระบบจำเป็นต้องติดตั้งระบบการจัดการฐานข้อมูล Oracle version 8i ขึ้นไป

### 3.2.3 ข้อจำกัดของระบบ

ในระบบ KTSQL version 1.0 นี้ มีข้อจำกัดในคำสั่ง KTSQL อยู่หลายส่วนดังนี้

- ในคำสั่ง Query ในกรณีที่ตารางนั้นมี Temporal Attribue มากกว่า 1 Temporal Attribute ถ้าระบุเงื่อนไขใน Where Condition แค่ 1 temporal Attribute จะเรียกดูข้อมูลของ Temporal Attribute

นั้น ส่วน Temporal Attribute ที่ไม่ได้ระบุเงื่อนไขถ้ามีการ Select Attribute นั้นด้วย ก็จะทำการสืบค้นข้อมูลขึ้นมาทั้งหมดโดยไม่มีเงื่อนไข เช่น ตาราง patient มี Temporal Attribute อยู่ 3 Temporal Attribute คือ doctor , room , bed ถ้าระบุเงื่อนไขที่ doctor = 2 แล้วไม่ได้ระบุ เงื่อนไขใน Temporal Attribute อื่นเลย การเรียกดูข้อมูลจะทำการแยกเรียกดูในแต่ละ Temporal Attribute ดังนั้นผลลัพธ์จะต้องแยกดูทีละ Temporal Attribute

- ในคำสั่ง Query ยังไม่สามารถทำการดึงข้อมูลโดยใช้ คำสั่ง group by , order by , built in function และ Sub query ได้

- ในคำสั่ง Insert ยังไม่สามารถ Insert ข้อมูลเข้าในตารางโดยระบุชื่อ Attribute ได้ และยังไม่สามารถทำการ Insert โดยการใช้ Select Statement ได้

**3.3 Use Case Specification**

**1. LoginDB**

**1.1 Brief Description**

Use case ทำหน้าที่รับ Username Password และ Hosting String จาก user เพื่อส่งไปประมวลผล

**1.2 Flow of Events**

*1.2.1 Basic Flow*

Use case นี้จะเริ่มทำงานเมื่อ user ต้องการติดต่อกับฐานข้อมูล

- Actor User กรอกข้อมูล Username Password และ Hosting String
- ระบบจะแสดงข้อมูลว่าสามารถเข้าใช้ระบบฐานข้อมูลได้

*1.2.2 Alternative Flows*

- ไม่มี

*1.2.3 Exceptional Flows*

- Display Error :ถ้าหาก User ใส่ข้อมูลการเข้าใช้ระบบฐานข้อมูลไม่ถูกต้องระบบจะส่งข้อความว่า "invalid username/password; logon denied "

**1.3 Special Requirement**

- ไม่มี

**1.4 Pre-Condition**

- ต้องมีระบบการจัดการฐานข้อมูลของ Oracle ตั้งแต่ version 8 ขึ้นไป
- จะต้องมีชื่อ Username และ Password อยู่ในระบบฐานข้อมูล

**1.5 Post- Condition**

- ไม่มี

**1.6 Extension Points**

- ไม่มี

**2. ConnectDB**

**2.1 Brief Description**

Use case นี้ ทำหน้าที่ติดต่อกับระบบการจัดการฐานข้อมูลของ Oracle Use case นี้จะทำการสร้าง Connection กับ DBMS เพื่อส่งข้อมูลเกี่ยวกับ Username Password และ Hosting String ให้กับ DBMS

**2.2 Flow of Events**

*2.2.1 Basic Flow*

Use case นี้จะเริ่มทำงานโดยการเรียกใช้จาก Use case LoginDB

- LoginDB เรียกใช้ Connect DB
- ConnectDB รับ Username password Hosting String จาก LoginDB
- ConnectDB ทำการติดต่อกับ DBMS โดยส่ง Username password Hosting String ให้ DBMS

*2.2.2 Alternative Flows*

- ไม่มี

*2.2.3 Exceptional Flows*

- Display Error : ถ้าหากข้อมูลที่ส่งไปไม่ถูกต้องจะส่งข้อความแจ้งว่า "invalid username/password; logon denied "

**2.3 Special Requirement**

- ไม่มี

**2.4 Pre-Condition**

- ต้องมีระบบการจัดการฐานข้อมูลของ Oracle ตั้งแต่ version 8 ขึ้นไป
- จะต้องมีชื่อ Username และ Password อยู่ในระบบฐานข้อมูล

**2.5 Post-Condition**

- ไม่มี

**2.6 Extension Points**

- AuthenticateUser

**3. AuthenticateUser**

**3.1 Brief Description**

Use case นี้ ทำหน้าที่ตรวจสอบว่าชื่อ Username password และ Hosting String ที่ได้จาก LoginDB มีความถูกต้องหรือไม่

**3.2 Flow of Events**

*3.2.1 Basic Flow*

Use case นี้จะทำงานเมื่อมี User เรียกใช้ เพื่อ ConnectDB ถูกเรียกใช้

- Authenticate User จะส่งชื่อ Username password และ Hosting String ให้กับ DBMS
- DBMS จะทำการตรวจสอบว่าข้อมูลที่ส่งถูกต้องหรือไม่
- ถ้าข้อมูลที่ส่งไปถูกต้องจะแสดงข้อความว่าสามารถ Login ได้แล้ว

*3.2.2 Alternative Flows*

- ไม่มี

*3.2.3 Exceptinonal Flows*

- Display Error : ถ้าหากข้อมูลที่ส่งไปไม่ถูกต้อง จะส่งข้อความแจ้งว่า "invalid username/password; logon denied "

**3.3 Special Requirement**

- ไม่มี

**3.4 Pre-Condition**

- ไม่มี

**3.5 Post-Condition**

- ไม่มี

**3.6 Extension Points**

- ไม่มี

**4. ConvertStmt**

**4.1 Brief Description**

Use case นี้ ทำหน้าที่รับข้อความภาษา KTSQL จาก User และแปลงภาษา KTSQL เป็น ORSQL

**4.2 Flow of Events**

*4.2.1 Basic Flow*

Use case นี้จะเริ่มทำงานโดยการ User พิมพ์คำสั่งเข้ามาทางหน้าจอ Input

- ConvertStmt จะรับข้อความจากหน้าจอ
- ConvertStmt ทำการตรวจสอบ Syntax ของภาษา KTSQLว่าถูกต้องหรือไม่
- ถ้าภาษา KTSQL มี Syntax ที่ถูกต้อง ConvertStmt จะทำการแปลงคำสั่ง KTSQL เป็นภาษา ORSQL ที่ Implement แนวคิด TSQL

*4.2.2 Alternative Flows*

- ไม่มี

*4.2.3 Exceptinonal Flows*

- Display Error : ถ้าหาก Syntax ที่ส่งไปไม่ถูกต้องจะส่งข้อความแจ้งว่า "Statement is invalid . "

**4.3 Special Requirement**

- ไม่มี

**4.4 Pre-Condition**

- User ต้องมีความรู้เกี่ยวกับ Syntax ภาษา KTSQL ที่ถูกกำหนดขึ้นมาใหม่

**4.5 Post-Condition**

- ไม่มี

**4.6 Extension Points**

- ไม่มี

**5. ManageDB**

**5.1 Brief Description**

use case นี้ ทำหน้าที่จัดการส่งคำสั่งภาษา ORSQL ที่ได้รับการแปลงแล้วไปให้ DBMS ทำการประมวลผลและแสดงผลลัพธ์ที่ตอบกลับจาก DBMS

**5.2 Flow of Events**

*5.2.1 Basic Flow*

Use case นี้จะเริ่มทำงานโดยการเรียกใช้จาก Use case ConvertStmt

- ManageDB รับคำสั่งที่ส่งมาจาก ConvertStmt
- ManageDB ทำการส่งคำสั่งที่ได้รับไปให้ DBMS เพื่อประมวลผล
- ManageDB รับผลลัพธ์ที่ตอบกลับมาจาก DBMS และทำการแสดงผล

*5.2.2 Alternative Flows*

- ไม่มี

*5.2.3 Exceptinonal Flows*

- ไม่มี

**5.3 Special Requirement**

- ไม่มี

**5.4 Pre-Condition**

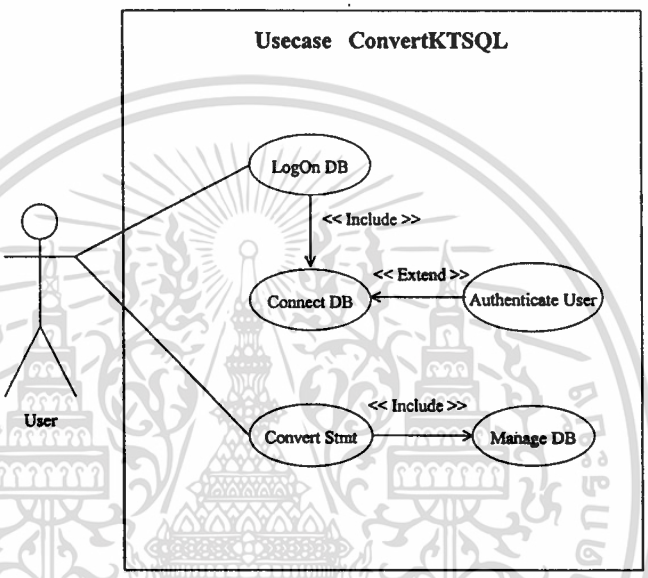
- ไม่มี

**5.5 Post-Condition**

- ไม่มี

**5.6 Extension Points**

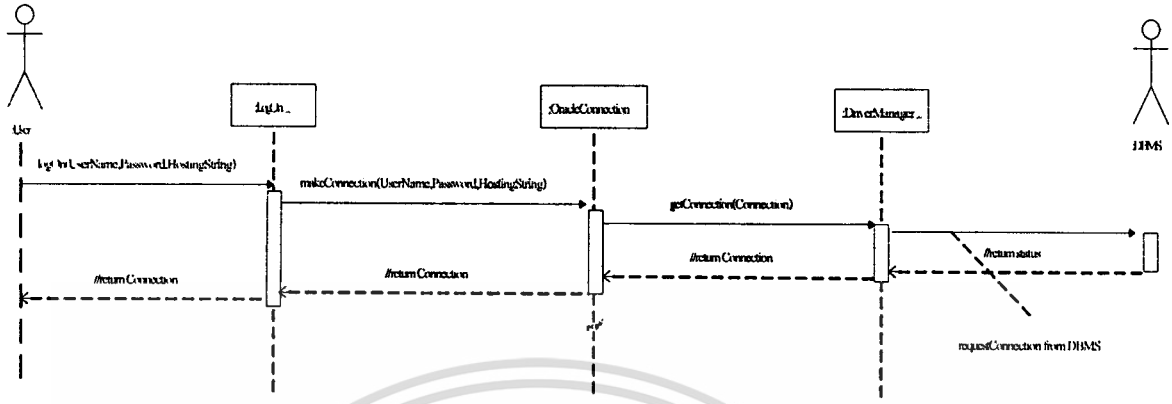
- ไม่มี




**รูปที่ 3.8** Usecase Diagram ConvertKTSQL

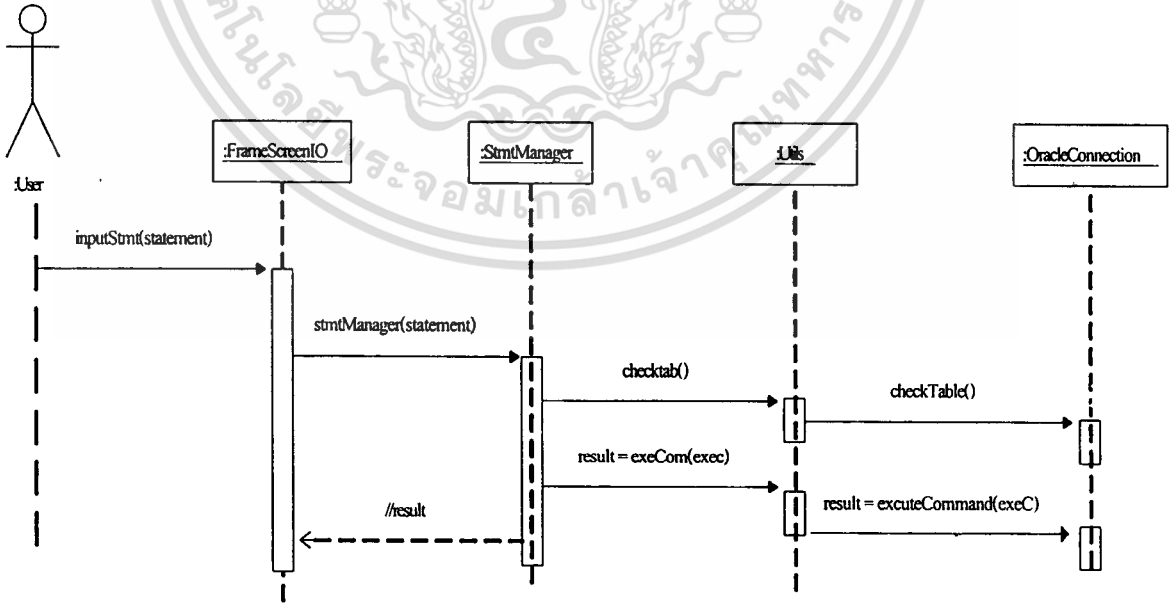
### 3.4 Sequenced Diagramของ Use case ConvertKTSQL

Sequenced Diagram ของ Use case ConvertKTSQL จะแสดงลำดับการทำงานต่างๆ โดยเริ่มต้นจาก Sequenced Diagram ของ Use case LogOn DB , Use case Connect DB และ Use case Authenticate User เนื่องจากสามารถมองการทำงานร่วมกันจึงแสดงลำดับการทำงานได้ดังนี้



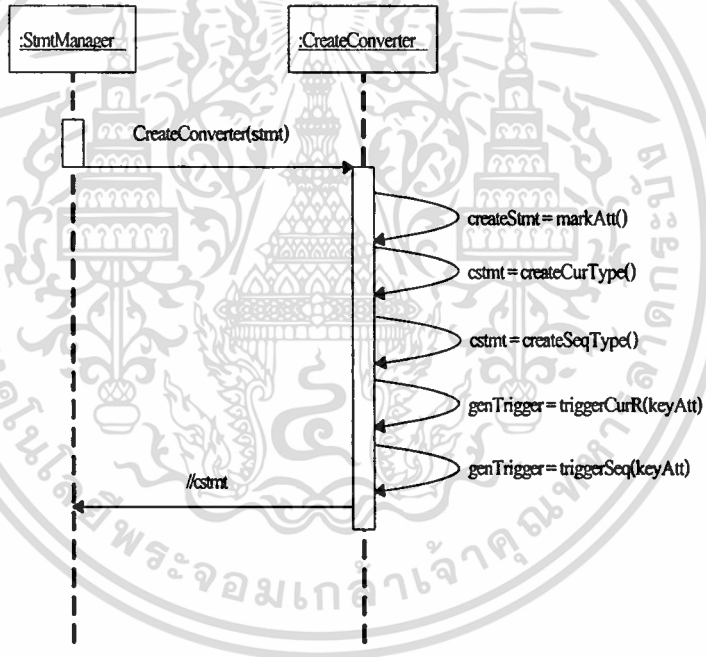

**รูปที่ 3.9** Sequenced Diagram ของ Use case LogOn DB
, Use case Connect DB และ Use case Authenticate User

จาก รูปที่ 3.9 User ทำการป้อนชื่อ User name , Password และ Hosting String ให้กับ Class Log On หลังจากนั้นระบบจะทำการ Log On ไปยังฐานข้อมูลโดยเรียก method makeConnection ( UserName , Password , HostingString ) จาก Class OracleConnection ซึ่งจะทำการเรียก method getConnection(Connection) จาก Class DriverManager เพื่อทำหน้าที่ติดต่อ DBMS ขอ Connection



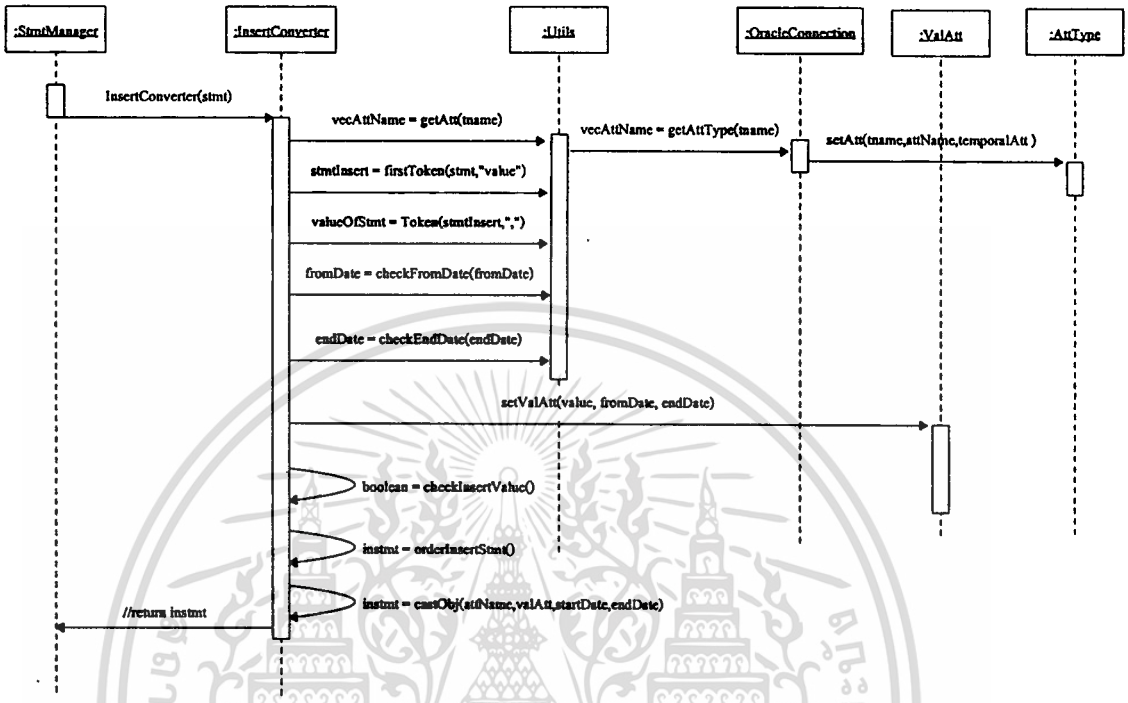

**รูปที่ 3.10** Sequenced Diagram ของ Use case Convert Stmt

จาก รูปที่ 3.10 Sequenced Diagram ของ Use case Convert Stmt จะเริ่มต้นจาก user ป้อนคำสั่งไปยัง Class FrameScreenIO หลังจากนั้นจะไปเรียก method stmtManager(statement) ของ Class StmtManager ซึ่งจะเริ่มต้นทำงาน โดยการเรียก Method checktab() ใน Class Utils เพื่อตรวจสอบว่าในฐานข้อมูลมีตารางที่ใช้เก็บข้อมูลของ KTSQL หรือไม่ ถ้ายังไม่มีก็จะทำการสร้างขึ้นมาใหม่ หลังจากนั้น StmtManager ก็จะเรียกใช้งาน Class ต่างๆ ตามคำสั่งที่ได้รับจาก User ซึ่งสามารถดูได้จาก Sequenced Diagram ของ Use case Manage DB ซึ่งหลังจากที่ได้รับคำสั่งที่แปลงส่งกลับมาแล้ว StmtManager จะทำการส่งคำสั่งนี้ไปยัง Class Util เพื่อเรียกใช้ method exeCom(exec)ทำงานหลังจากนั้นจะทำการแสดงผลที่ Class FrameScreenIO




**รูปที่ 3.11** Sequenced Diagram ของ Use case Manage DB

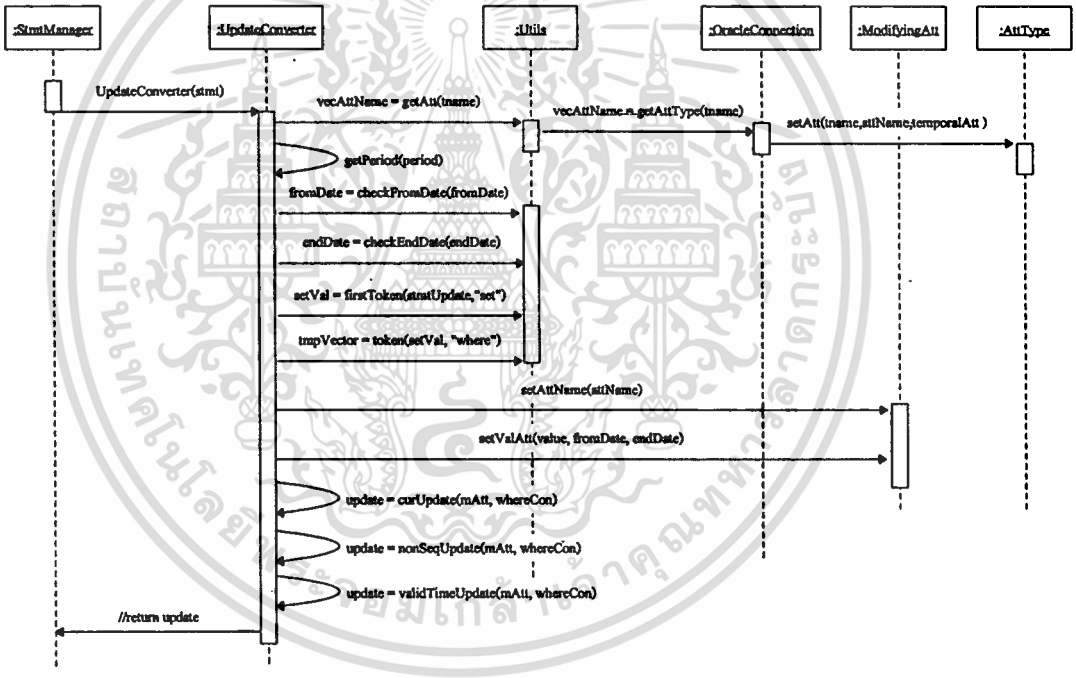
จากรูปที่ 3.11 เริ่มต้นที่ Class StmtManager ซึ่งจะทำหน้าที่ในการจัดการส่ง message เรียกไปยัง Class CreateConverter โดยผ่าน method CreateConverter(stmt) ซึ่งภายใน Classจะทำการแปลงคำสั่งให้อยู่ในรูปแบบที่เหมาะสมโดยจะเรียกใช้ method markAtt(), createCurType() , createSeqType() , triggerCurR(keyAtt) ,triggerSeq(keyAtt) และ Return คำสั่งกลับไปยัง Class StmtManager เพื่อส่งไปประมวลผลต่อไป




**รูปที่ 3.12** Sequenced Diagram ของ Use case Manage DB

จากรูปที่ 3.12 เริ่มต้นที่ User ส่งคำสั่ง Insert Statement ไปยัง Class StmtManager ซึ่งจะทำหน้าที่ในการจัดการส่ง message เรียกไปยัง Class InsertConverter เพื่อทำการแปลงคำสั่งให้อยู่ในรูปแบบที่เหมาะสมและ Return คำสั่งกลับมาเพื่อส่งไปประมวลผลต่อไป โดย Class InsertConverter จะทำการเรียกใช้ method getAtt (tname)โดยส่ง parameter tname ไปยัง Class Utils ซึ่งเก็บชื่อ Table เพื่อตรวจสอบว่า table นี้มีชื่อ Attribute อะไรบ้าง และเป็น Temporal Attribute หรือไม่ โดย Class Utils จะทำการเรียก Class OracleConnection โดยเรียกใช้ผ่าน method getAttType( tname) ซึ่ง Class OracleConnection จะทำการเรียก Class AttType เพื่อใช้ method setAtt( tname , attName , temporalAtt) แล้วเก็บใน Vetor ชื่อ vetAttName หลังจากนั้น Class InsertConverter จะเรียกใช้ Class Utils โดยมี method firstToken(stmt,value) ซึ่ง return ค่า มาเก็บใน stmtInsert และ เรียกใช้ method token( stmtInsert , "." ) ซึ่ง return ค่า มาเก็บใน valueOfStmt ต่อมาเรียกใช้ method checkFromDate (fromDate)ซึ่ง return ค่า มาเก็บใน fromDate และ checkEndDate(endDate) ซึ่ง return ค่ามาเก็บใน

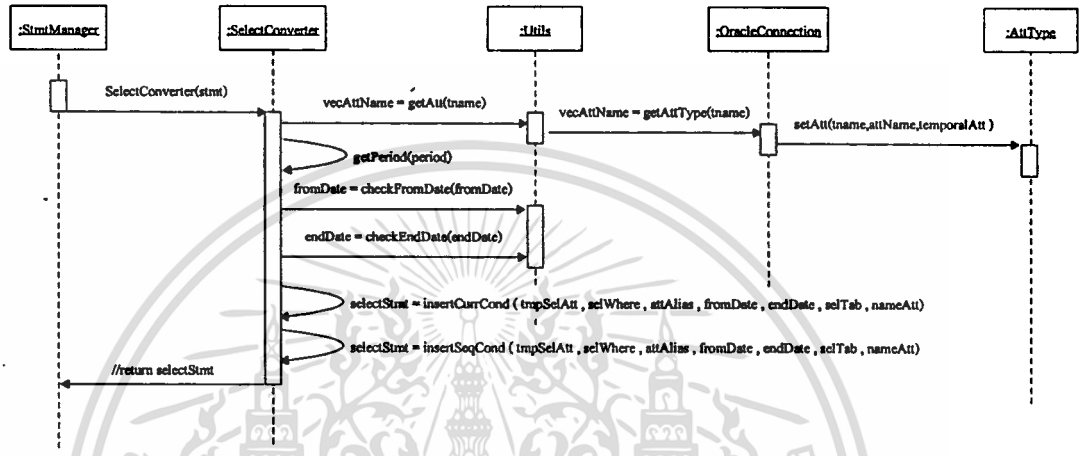
endDate ต่อมาเรียกใช้ Class ValAtt ผ่าน method setValueAtt (value , fromDate , endDate) หลังจากนั้นจะทำการเรียก method ภายใน Class ของตัวเอง คือ checkInsertValue() ซึ่งจะ return boolean , orderInsertStmt() ซึ่ง return ค่ามาเก็บใน inStmt และ castObj ( attName , valAtt , startDate , endDate ) ซึ่ง return ค่ามาเก็บใน inStmt




**รูปที่ 3.13** Sequenced Diagram ของ Use case Manage DB

จากรูปที่ 3.13 เริ่มต้นที่ User ส่งคำสั่ง Upcate Statement ไปยัง Class StmtManager ซึ่งจะทำหน้าที่ในการจัดการส่ง message โดยเรียกผ่าน method UpdateConverter(stmt) ของ Class UpdateConverter เพื่อทำการแปลงคำสั่งให้อยู่ในรูปแบบที่เหมาะสมและ Return คำสั่งกลับมาเพื่อส่งไปประมวลผลต่อไป โดย Class UpdateConverter จะทำการเรียกใช้ method getAtt (tname)โดยส่ง

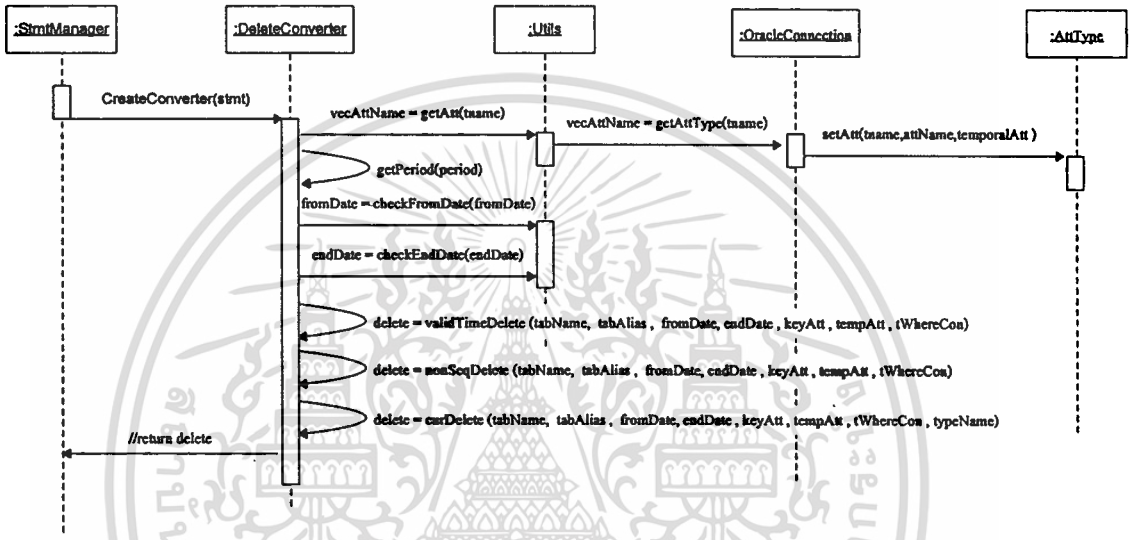
parameter tname ไปยัง Class Utils ซึ่งเก็บชื่อ Table เพื่อตรวจสอบว่า table นี้มีชื่อ Attribute อะไรบ้าง และเป็น Temporal Attribute หรือไม่ โดย Class Utils จะทำการเรียก Class OracleConnection โดยเรียกใช้ผ่าน method getAttType( tname) ซึ่ง Class OracleConnection จะทำการเรียก Class AttType เพื่อใช้ method setAtt( tname , attName , temporalAtt) แล้วเก็บใน Vetor ชื่อ vetAttName หลังจากนั้นจะเรียกใช้ method getPeriod(period) ของ Class UpdateConverter และจะเรียกใช้ Class Utils โดยมี method checkFromDate (fromDate) ซึ่ง return ค่า มาเก็บใน fromDate และ checkEndDate(endDate) ซึ่ง return ค่า มาเก็บใน endDate ต่อมาทำการเรียก method firstToken ( stmtUpdate , "set" ) ซึ่ง return ค่า มาเก็บใน setValue และเรียก method token( setValue , "where" ) ซึ่ง return ค่า มาเก็บใน tmpVector ต่อมาทำการเรียก Class ModifyingAtt โดยเรียกใช้ method setAttName(attName) และ setValueAtt( value , fromDate , endDate ) และต่อมาทำการเรียก method ภายใน Class ของตัวเองคือ currUpdat( mAtt , whereCon) ,nonSeqUpdate( mAtt , whereCon) ,validTimeUpdate(mAtt , whereCon) และนำค่าที่ถูก return มาเก็บใน update




**รูปที่ 3.14** Sequenced Diagram ของ Use case Manage DB

จากรูปที่ 3.14 เริ่มต้นที่ User ส่งคำสั่ง Select Statement ไปยัง Class StmtManager ซึ่งจะทำหน้าที่ในการจัดการส่ง message โดยเรียกผ่าน method SelectConverter(stmt) ของ Class SelectConverter เพื่อทำการแปลงคำสั่งให้อยู่ในรูปแบบที่เหมาะสมและ Return คำสั่งกลับมาเพื่อส่งไปประมวลผลต่อไป โดย Class SelectConverter จะทำการเรียกใช้ method getAtt (tname)โดยส่ง parameter tname ไปยัง Class Utils ซึ่งเก็บชื่อ Table เพื่อตรวจสอบว่า table นี้มีชื่อ Attribute อะไรบ้าง และเป็น Temporal Attribute หรือไม่ โดย Class Utils จะทำการเรียก Class OracleConnection โดยเรียกใช้ผ่าน method getAttType( tname) ซึ่ง Class OracleConnection จะทำการเรียก Class AttType เพื่อใช้ method setAtt( tname , attName , temporalAtt) แล้วเก็บใน Vetor ชื่อ vetAttName หลังจากนั้นจะเรียกใช้ method getPeriod(period) ของ Class SelectConverter และจะเรียกใช้ Class Utils โดยมี method checkFromDate (fromDate) ซึ่ง return ค่า มาเก็บใน fromDate และ checkEndDate(endDate) ซึ่ง return ค่า มาเก็บใน endDate ต่อมาทำการเรียก method ภายใน Class SelectConverter คือ insertCurrCond ( tmpSelAtt , selWhere , attAlias , fromDate , endDate , selTab , nameAtt ) ซึ่ง return

ค่า มาเก็บใน selectStmt และเรียก method insertSeqCond ( tmpSelAtt , selWhere , attAlias , fromDate , endDate , selTab , nameAtt ) ซึ่ง return ค่า มาเก็บใน selectStmt




**รูปที่ 3.15** Sequenced Diagram ของ Use case Manage DB

จากรูปที่ 3.15 เริ่มต้นที่ User ส่งคำสั่ง Delete Statement ไปยัง Class StmtManager ซึ่งจะทำหน้าที่ในการจัดการส่ง message โดยเรียกผ่าน method DeleteConverter(stmt) ของ Class DeleteConverter เพื่อทำการแปลงคำสั่งให้อยู่ในรูปแบบที่เหมาะสมและ Return คำสั่งกลับมาเพื่อส่งไปประมวลผลต่อไป โดย Class DeleteConverter จะทำการเรียกใช้ method getAtt (tname)โดยส่ง parameter tname ไปยัง Class Utils ซึ่งเก็บชื่อ Table เพื่อตรวจสอบว่า table นี้มีชื่อ Attribute อะไรบ้าง และเป็น Temporal Attribute หรือไม่ โดย Class Utils จะทำการเรียก Class OracleConnection โดยเรียกใช้ผ่าน method getAttType( tname) ซึ่ง Class OracleConnection จะทำการเรียก Class AttType เพื่อใช้ method setAtt( tname , attName , temporalAtt) แล้วเก็บใน Vetor ชื่อ vetAttName หลังจากนั้นจะเรียกใช้ method getPeriod(period) ของ Class DeleteConverter และจะเรียกใช้ Class Utils โดยมี method checkFromDate (fromDate) ซึ่ง return ค่า มาเก็บใน fromDate และ checkEndDate(endDate)

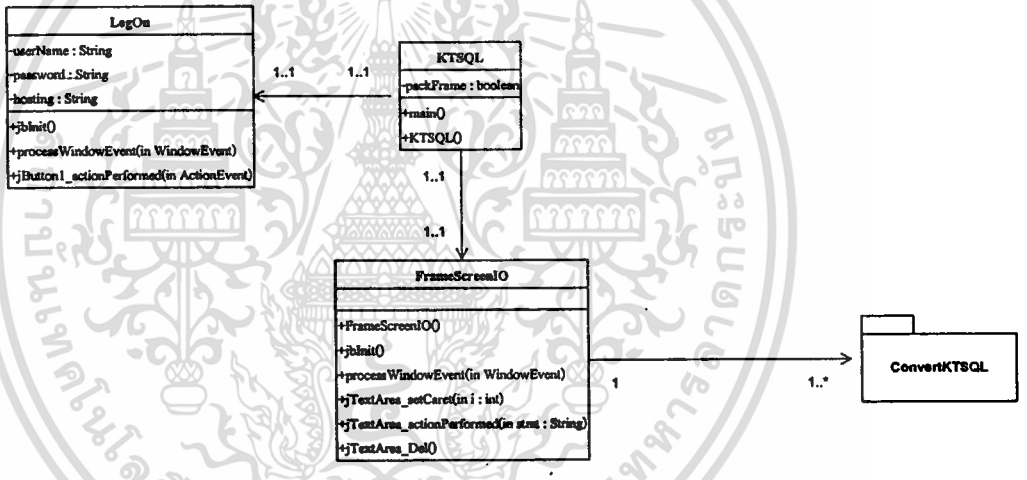
ซึ่ง return ค่า มาเก็บใน endDate ต่อมาทำการเรียก method ภายใน Class DeleteConverter คือ validTimeDelete (tabName , tabAlias , fromDate , endDate , keyAtt , tempAtt , tWhereCon) ซึ่ง return ค่า มาเก็บใน delete ต่อมาจึงเรียก nonSeqDelete (tabName , tabAlias , fromDate , endDate , keyAtt , tempAtt , tWhereCon) ซึ่ง return ค่า มาเก็บใน delete และ curDelete (tabName , tabAlias , fromDate , endDate , keyAtt , tempAtt , tWhereCon , typeName ) ซึ่ง return ค่า มาเก็บใน delete

### 3.5 Class Diagram ของระบบ ConvertKTSQL

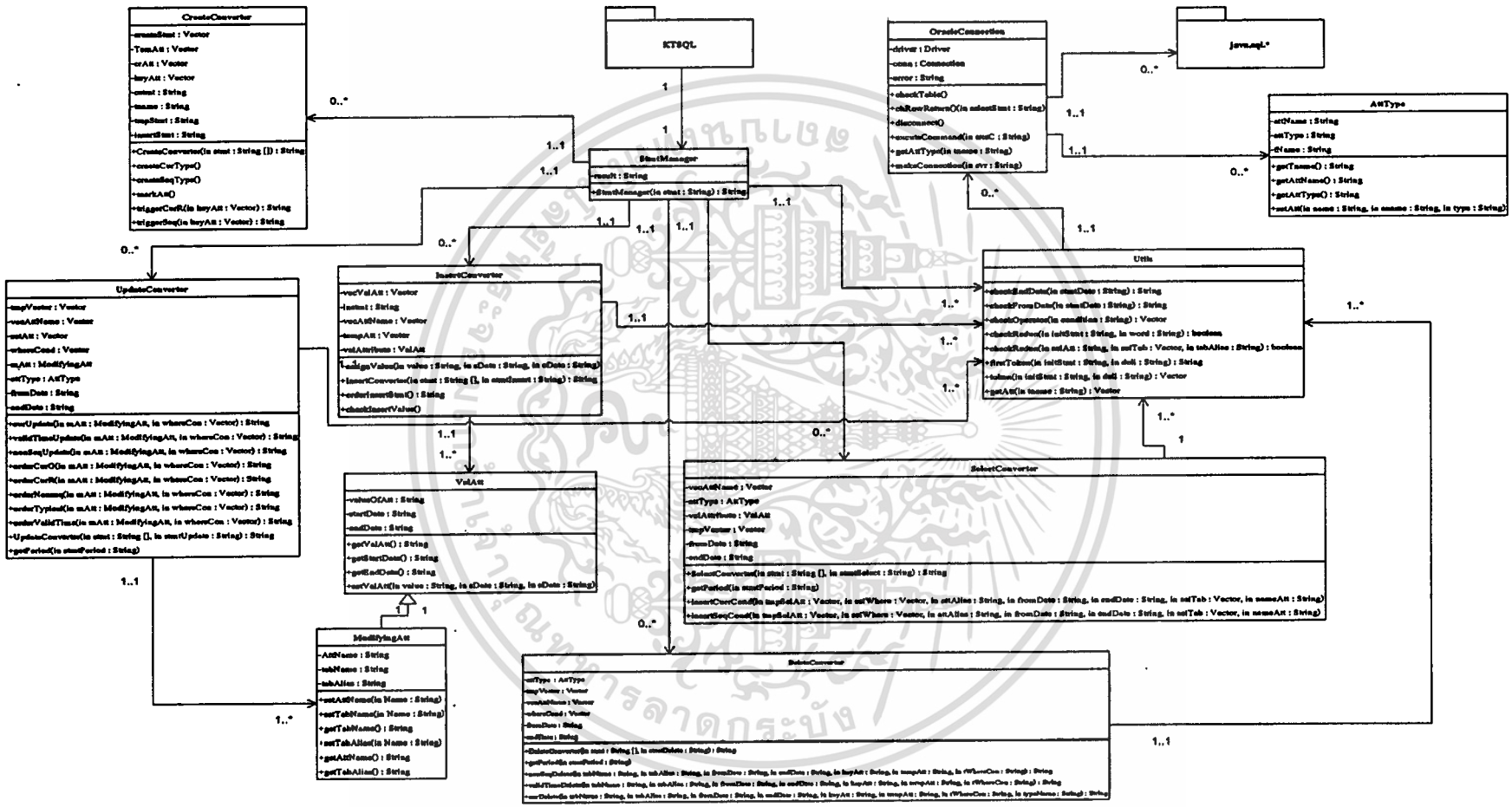
ระบบงาน ConvertKTSQL จะประกอบไปด้วย Package 2 ส่วน ด้วยกันคือ

1. Package KTSQL สามารถดูได้จากรูปที่3.16 ซึ่งจะประกอบไปด้วยClass ดังต่อไปนี้
   - Class KTSQL เป็นClass เริ่มต้นของระบบ KTSQLซึ่งจะเรียก Class LogOn และ Class FrameScreenIO
   - Class LogOn ทำหน้าที่ในการรับ User name , Password , Hosting String จากผู้ใช้
   - Class FrameScreenIO ทำหน้าที่รับคำสั่งจาก User และติดต่อกับ Package ConvertKTSQL
2. Package ConvertKTSQL สามารถดูได้จากรูปที่3.17 ซึ่งจะประกอบไปด้วยClass ดังต่อไปนี้
   - Class StmtManager จะเป็น Class เริ่มต้นของ Package นี้ซึ่งจะทำหน้าที่เรียกใช้ Class อื่นๆให้เหมาะสมกับคำสั่งที่ส่งเข้ามา
   - Class OracleConnection ทำหน้าที่ติดต่อกับฐานข้อมูลของ Oracle
   - Class Utils เป็น Class ที่รวบรวม Method ที่ถูกเรียกใช้งานบ่อยๆจาก Class อื่นๆ
   - Class CreateConverter ทำหน้าที่จัดการคำสั่ง Create table
   - Class UpdateConverter ทำหน้าที่จัดการคำสั่ง Update table
   - Class SelectConverter ทำหน้าที่จัดการคำสั่ง Query ข้อมูล
   - Class InsertConverter ทำหน้าที่จัดการคำสั่ง Insert table
   - Class DeleteConverter ทำหน้าที่จัดการคำสั่ง Delete ข้อมูล
   - Class ValAtt ทำหน้าที่เก็บข้อมูลที่ต้องการ Insert

- Class ModifyingAtt ทำหน้าที่เก็บข้อมูลที่ต้องการ Update โดย inherit มาจาก
- Class AttType ทำหน้าที่เก็บข้อมูลเกี่ยวกับ Attribute




**รูปที่ 3.16** Class Diagram ที่อยู่ในpackage KTSQL ซึ่งเป็น Interface สำหรับติดต่อกับผู้ใช้




**รูปที่ 3.17** Class Diagram ที่อยู่ในpackage ConvertKTSQL

# บทที่ 4

# การพัฒนา KTSQL บน ORDBMS

## 4.1 การพัฒนา KTSQL ด้วย ORDBMS

ในการพัฒนาและออกแบบ KTSQL นำแนวคิด Object Relational Database (ORDB) มาใช้จัดการฐานข้อมูลโดยใช้ Nested Table ในการ Implement แนวคิดเชิงเวลา เริ่มจากการสร้างตารางขึ้นโดยใช้คำสั่ง KTSQL ที่กำหนดขึ้นมาใหม่ ชื่อตาราง Patient เพื่อเก็บ หมายเลข ชื่อ นามสกุล ของคนไข้ แพทย์ที่ทำการรักษา ห้องที่เข้าพัก และหมายเลขเตียง เป็นต้น

```
CREATE TABLE Patient(
  ID int ,
  Name varchar(20) ,
  sname varchar(20) ,
  Doctor int  validtime ,
  Room int curr ,
  Bed  int curr );
```

จากคำสั่งที่ใช้สร้างตารางข้างต้นจะทำการ map คำสั่งดังกล่าว เพื่อสร้างตารางโดยใช้แนวคิด ORDB โดยมีการสร้าง Data Type ขึ้นมาสำหรับ Attribute ที่มี temporal_keyword อยู่ด้านหลัง เพื่อทำการเก็บข้อมูลแบบ Nested Table มีอยู่ 3 Attribute ด้วยกัน คือ

1. Attribute Doctor เป็น Attribute ที่ใช้เก็บข้อมูลเชิงเวลาแบบ Current Restricted โดยใช้ชื่อ Data Type ว่า doctor#patient#ty
2. Attribute Room เป็น Attribute ที่ใช้เก็บข้อมูลเชิงเวลาแบบ Sequenced โดยใช้ชื่อ Data Type ว่า room#patient#ty
3. Attribute Bed เป็น Attribute ที่ใช้เก็บข้อมูลเชิงเวลาแบบ Sequenced โดยใช้ชื่อ Data Type ว่า bed#patient#ty

ขั้นแรกทำการสร้าง Data Type ขึ้นมา 3 Data Type โดยมีคำสั่งดังต่อไปนี้

```
Create or replace type doctor#patient#ty as object (
```

```
doctor int , FROM_DATE  Date, END_DATE   Date)

Create or replace type room#patient#ty as object (
room int , FROM_DATE  Date, END_DATE   Date)

Create or replace type bed#patient#ty as object (
bed int , FROM_DATE  Date, END_DATE   Date)
```

ขั้นต่อมาจะนำ Data Type ที่ถูกสร้างขึ้นนำมาสร้าง Type ที่เป็นเหมือนกับตารางเพื่อใช้เป็น Nested Table ของ Attribute

```
Create Type room#patient#NT as Table of room#patient#ty

Create Type bed#patient#NT as Table of bed#patient#ty

Create Type doctor#patient#NT as Table of doctor#patient#ty
```

ต่อมาจึงทำการสร้างตาราง patient ซึ่งมี Attribute ตามที่กำหนดในคำสั่ง Create แต่จะแตกต่างกัน คือ Attribute ที่มี temporal_keyword จะมีการนิดข้อมูลเป็น Type ที่สร้างขึ้นจากคำสั่งก่อนหน้านี้ ซึ่งจะเก็บข้อมูลเชิงเวลาและเปรียบเสมือนเป็นตารางที่ซ้อนอยู่ในตาราง patient อีกที

```
create  table  patient (
    id int  ,  name  varchar(20)  ,  sname  varchar(20)  ,
    doctor doctor#patient#NT ,
    room room#patient#NT,
    bed bed#patient#NT)
Nested table doctor store as doctor#NT_tab
Nested table room store as room#Ntab
Nested table bed store as bed#Ntab
```

จากคำสั่งข้างต้นจะสร้างตาราง patient ขึ้นมาโดยจะมีโครงสร้างของตารางที่สร้างเป็นแบบ Nested Table ซึ่งจะมีโครงสร้างดังรูปต่อไปนี้

**ตารางที่ 4.1** ตาราง patient แบบ Nested Table

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_ DATE | END_ DATE | ROOM | FROM_ DATE | END_ DATE | BED | FROM_ DATE | END_ DATE |
| | | | | | | | | | | | |

จากคำสั่งในการสร้างตารางนี้ยังไม่สามารถที่จะป้องกันความซ้ำซ้อนของข้อมูลในตาราง patient ได้ จึงจำเป็นต้องสร้าง Trigger ขึ้นมาเพื่อคอยตรวจสอบค่าที่ทำการเพิ่มหรือแก้ไขข้อมูลว่าการ Insert หรือการ Update ข้อมูลว่ามีข้อมูลที่ไม่เหมาะสมอยู่ในตารางหรือไม่ ซึ่งทำการตรวจสอบความซ้ำซ้อนตาม temporal_keyword ที่ระบุอยู่ท้าย Type ของ Attribute ซึ่งมีอยู่ 2 ประเภทด้วยกัน ถ้ามี Keyword Validtime จะทำการป้องกันความซ้ำซ้อนแบบ Sequenced อยู่ 1 Trigger ด้วยกัน คือ seq#doctor#patient ถ้ามี Keyword Curr จะทำการป้องกันความซ้ำซ้อนแบบ Current 1 trigger คือ cr#room#patient และ cr#bed#patient

**1. Trigger ป้องกันความซ้ำซ้อนแบบ Sequenced**

```
CREATE OR REPLACE TRIGGER seq#doctor#patient
AFTER INSERT OR UPDATE ON patient
DECLARE valid INTEGER ;
BEGIN
    SELECT 1 INTO valid FROM DUAL WHERE NOT EXISTS
    (select * FROM patient L1,patient L2,
            table(L1.doctor)VT1,table(L2.doctor)VT2
            WHERE L1.id = L2.id and L1.name = L2.name and L1.sname = L2.sname
            AND  VT1.FROM_DATE < VT2.END_DATE
            AND VT2.FROM_DATE < VT1.END_DATE AND VT1.rowid != VT2.rowid);
EXCEPTION
 WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR
( -20001 , 'Data must be sequenced unique' );
END;
```

คำสั่งดังกล่าวจะทำการสร้าง Trigger ขึ้นมา เพื่อคอยตรวจสอบค่าที่เปลี่ยนแปลงในตารางไม่ว่าจะเป็นการ Insert หรือ การ Update ข้อมูลว่าจะมีความซ้ำซ้อนแบบ Sequence หรือไม่ ถ้ามีก็จะทำการแจ้งให้ทราบ

**2. Trigger ป้องกันความซ้ำซ้อนแบบ Current**

```
CREATE OR REPLACE TRIGGER cr#room#patient
AFTER INSERT OR UPDATE ON patient
DECLARE valid INTEGER ;
```

```
BEGIN
    SELECT 1 INTO valid FROM DUAL WHERE NOT EXISTS
    (select * FROM patient L1,patient L2,
            table(L1.room)VT1,table(L2.room)VT2
            WHERE L1.id = L2.id and L1.name  = L2.name  and L1.sname  = L2:sname
            AND VT1.FROM_DATE <= CURRENT_DATE
            AND CURRENT_DATE < VT1.END_DATE
            AND VT2.FROM_DATE <= CURRENT_DATE
            AND CURRENT_DATE < VT2.FROM_DATE AND VT1.rowid != VT2.rowid);
EXCEPTION
 WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR
( -20001 , 'Data must be Current unique' );
END;

CREATE OR REPLACE TRIGGER cr#bed#patient
AFTER INSERT OR UPDATE ON patient
DECLARE valid INTEGER ;
BEGIN
    SELECT 1 INTO valid FROM DUAL WHERE NOT EXISTS
    (select * FROM patient L1,patient L2,
            table(L1.bed)VT1,table(L2.bed)VT2
            WHERE L1.id = L2.id and L1.name  = L2.name  and L1.sname  = L2.sname
            AND VT1.FROM_DATE <= CURRENT_DATE
            AND CURRENT_DATE < VT1.END_DATE
            AND VT2.FROM_DATE <= CURRENT_DATE
            AND CURRENT_DATE < VT2.FROM_DATE   AND VT1.rowid != VT2.rowid);
 EXCEPTION
  WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR
  ( -20001 , 'Data must be Current unique' );
  END;
```

คำสั่งดังกล่าวจะทำการสร้าง Trigger เพื่อคอยตรวจสอบค่าที่เปลี่ยนแปลงในตารางไม่ว่าจะเป็นการ Insert หรือ การ Update ข้อมูลว่าจะมีความซ้ำซ้อนแบบ Current หรือไม่ ถ้ามีก็จะทำการแจ้งให้ทราบ

**การเปลี่ยนแปลงแก้ไขสถานะของตารางวาลิดไทม์ (Modifying Valid –Time State Tables)**

ฐานข้อมูลเชิงเวลาจะมีการทำงานกับข้อมูลอยู่ด้วยกันดังนี้ คือ การเพิ่มข้อมูล (Insert) การลบข้อมูล (Delete) การแก้ไขปรับปรุงข้อมูล (Update) การกระทำกับข้อมูลทั้ง 3 ลักษณะจะต้องคำนึงถึง คือ ข้อมูลแบบ Current , Sequenced และ Nonsequenced

**การ Insert ข้อมูลโดยใช้คำสั่ง KTSQL ที่กำหนดขึ้นมา**

จากที่กล่าวไว้ในบทที่ 3 ส่วนของการ Insert ข้อมูลเข้าไปยังฐานข้อมูลโดยมีคำสั่งดังนี้

```
INSERT INTO Patient
values(02, 'k.b' , 'boon' , 3 '[2002-02-10 - 2002-03-02)' ,2  ,2 ) ;
```

ซึ่งคำสั่งข้างต้นสามารถนำมา Map เป็นภาษา ORSQL โดยจะทำการตรวจสอบว่า Attribute ที่ทำการ Insert ข้อมูลเป็น Temporal Attribute แบบ Sequenced หรือ Current-Restricted ในกรณีเป็นแบบ Sequenced จะสามารถ Insert ข้อมูลได้ทั้งแบบใส่ช่วงเวลาหรือไม่ระบุช่วงเวลาก็ได้แต่ถ้าเป็น Current Restricted จะไม่สามารถ Insert ข้อมูลแบบเป็นช่วงเวลาได้จะสามารถ Insert ได้แค่เวลาเริ่มต้นเป็น CURRENT_DATE และเวลาสิ้นสุดเป็น '9999-12-31' เท่านั้น ซึ่งไม่จำเป็นจะต้องใส่ Parameter period_t ด้านหลังค่าที่ต้องการ Insert หลังจากที่ได้ทำการ Insert ข้อมูลลงในฐานข้อมูลแล้วจะมีข้อมูลในตาราง Patient ดังตารางที่ 4.2 โดยสมมุติข้อมูลขึ้นมาจากคำสั่งดังกล่าวสามารถสร้างเป็นคำสั่ง ORSQL ได้ดังนี้

```
insert into patient Values (
02,'k.b','boon',
doctor#patient#nt(
        doctor#patient#ty  (3,  to_date('2002-02-10','YYYY-MM-DD'),to_date
        ('2002-03-02','YYYY-MM-DD'))) ,
room#patient#nt (
```

room#patient#ty( 2 ,to_date(CURRENT_DATE,'YYYY-MM-DD'),to_date('9999-12-31','YYYY-MM-DD'))) ,

bed#patient#nt (

bed#patient#ty(2,to_date(CURRENT_DATE,'YYYY-MM-DD'),to_date('9999-12-31','YYYY-MM-DD'))) )

ตารางที่ 4.2 ตาราง patient ที่ได้ทำการ Insert ข้อมูลแล้ว

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Boon | 1 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB- 98' | '22-MAR-98' | 1 | '27-FEB- 98' | '12-MAR-98' |
| | | | 1 | '23-MAR-98' | '31-DEC-9999' | 2 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '01-APR- 98' | '10-MAY-98' | 1 | '09-APR- 98' | '21-MAY-98' |
| | | | | | | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | 2 | '29-JUN- 98' | '17-OCT-98' | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '17-OCT- 98' | '19-OCT-98' | | | |
| | | | | | | 2 | '19-OCT- 98' | '31-DEC-9999' | | | |

ตารางที่ 4.2 ตาราง patient ที่ได้ทำการ Insert ข้อมูลแล้ว (ต่อ)

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 2 | k. Alongorn | Siriwan | 2 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB- 98' | '22-MAR-98' | 2 | '17-FEB- 98' | '22-MAR-98' |
| | | | 2 | '23-MAR-98' | '31-DEC-9999' | 3 | '22-MAR-98' | '01-APR-98' | 1 | '22-MAR-98' | '01-APR-98' |
| | | | 4 | '01-JAN-98' | '31-DEC-9999' | | | | | | |
| 3 | k. Chat | Mariwan | 1 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB- 98' | '22-MAR-98' | 3 | '17-FEB- 98' | '22-MAR-98' |
| | | | 2 | '23-MAR-98' | '31-DEC-9999' | 2 | '22-MAR-98' | '01-APR-98' | 2 | '22-MAR-98' | '01-APR-98' |
| | | | 4 | '01-JAN-98' | '31-DEC-9999' | | | | | | |

**การ Update ข้อมูลโดยใช้คำสั่ง KTSQL ที่กำหนดขึ้นมา**

การเปลี่ยนแปลงแก้ไขข้อมูลในฐานข้อมูลจะแบ่งเป็น 3 ประเภทด้วยกันคือข้อมูลแบบ Sequenced , Current Restricted และ Nonsequenced ซึ่งจะอธิบายรายละเอียดดังต่อไปนี้

1. การ Update แบบ Sequenced จะต้องมี Keyword ด้านหน้าคำสั่ง UPDATE ว่า VALIDTIME สามารถ Update โดยให้ Period ตามหลังคำว่า VALIDTIME หรือตามชื่อ Attribute ที่ต้องการเปลี่ยนแปลงแก้ไขข้อมูล
2. การ Update แบบ Current จะแบ่ง Scenarios เป็น 2 แบบด้วยกันคือ Restricted Scenario และ General Scenario ซึ่ง
   - การ Update ข้อมูลแบบ Restricted Scenario จะสามารถทำการ Update ข้อมูลใน Temporal Attribute ที่เป็นแบบ Current Restricted เท่านั้น
   - การ Update ข้อมูลแบบ General Scenario จะสามารถทำการ Update ข้อมูลใน Temporal Attribute ที่เป็นแบบ Sequenced เท่านั้น
3. การ Update แบบ Nonsequenced จะไม่สนใจช่วงเวลาในการเปลี่ยนแปลงข้อมูลในตารางสามารถใส่ Period ตามหลังชื่อ Temporal Attrittribute ที่รเปลี่ยนแปลงข้อมูลได้เลยหรือจะเปลี่ยนแปลงค่าที่ต้องการโดยไม่สนใจช่วงเวลาทั้ง Current และ Sequenced ก็ได้

ตัวอย่างการ Update แบบ Sequenced

```
VALIDTIME PERIOD '[1998-03-01 - 1998-04-01)' UPDATE patient
        SET doctor PERIOD '[1998-02-07 - 1998-02-18)'= 6
        ,sname='sonthaya' ,room = 200
        WHERE  id=01 and room = 02 and doctor =01;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. alter table patient disable all triggers
2. insert into the(select doctor from patient patient where patient.id =01 ) select vt.doctor,vt.FROM_DATE,to_date('1998-02-07','YYYY-MM-DD') from patient patient1 ,table (patient1.doctor)vt where vt.doctor = 1 and vt.FROM_DATE < to_date('1998-02-07','YYYY-MM-DD')and vt.END_DATE > to_date('1998-02-07','YYYY-MM-DD') and patient1.id =01
3. insert into the(select doctor from patient patient where patient.id =01 ) select vt.doctor,to_date ('1998-02-18','YYYY-MM-DD'),vt.END_DATE from patient patient1 ,table(patient1.doctor)vt

where vt.doctor = 1 and vt.FROM_DATE < to_date('1998-02-18','YYYY-MM-DD')and vt.END_DATE > to_date('1998-02-18','YYYY-MM-DD') and patient1.id =01

4. update the(select doctor from patient patient where patient.id =01 ) set doctor=6 where FROM_DATE < to_date('1998-02-18','YYYY-MM-DD') and END_DATE > to_date('1998-02-07','YYYY-MM-DD') and doctor = 1

5. update the(select doctor from patient patient where patient.id =01 ) set FROM_DATE=to_date ('1998-02-07','YYYY-MM-DD') where FROM_DATE < to_date('1998-02-07','YYYY-MM-DD') and END_DATE > to_date('1998-02-07','YYYY-MM-DD') and doctor = 1

6. update the(select doctor from patient patient where patient.id =01 ) set END_DATE=to_date ('1998-02-18','YYYY-MM-DD') where FROM_DATE < to_date('1998-02-18','YYYY-MM-DD') and END_DATE > to_date('1998-02-18','YYYY-MM-DD') and doctor = 1

7. update patient patient set patient.sname ='sekhanan' where patient.id =01

8. insert into the(select room from patient patient where patient.id =01 ) select vt.room,vt.FROM_DATE,to_date('1998-03-01','YYYY-MM-DD') from patient patient1 ,table (patient1.room)vt where vt.room = 2 and vt.FROM_DATE < to_date('1998-03-01','YYYY-MM-DD')and vt.END_DATE > to_date('1998-03-01','YYYY-MM-DD') and patient1.id =01

9. insert into the(select room from patient patient where patient.id =01 ) select vt.room,to_date ('1998-04-01','YYYY-MM-DD'),vt.END_DATE from patient patient1 ,table(patient1.room)vt where vt.room = 2 and vt.FROM_DATE < to_date('1998-04-01','YYYY-MM-DD')and vt.END_DATE > to_date('1998-04-01','YYYY-MM-DD') and patient1.id =01

10. update the(select room from patient patient where patient.id =01 ) set room=200 where FROM_DATE < to_date('1998-04-01','YYYY-MM-DD') and END_DATE > to_date('1998-03-01','YYYY-MM-DD') and room = 2

11. update the(select room from patient patient where patient.id =01 ) set FROM_DATE=to_date ('1998-03-01','YYYY-MM-DD') where FROM_DATE < to_date('1998-03-01','YYYY-MM-DD') and END_DATE > to_date('1998-03-01','YYYY-MM-DD') and room = 2

12. update the(select room from patient patient where patient.id =01 ) set END_DATE=to_date ('1998-04-01','YYYY-MM-DD') where FROM_DATE < to_date('1998-04-01','YYYY-MM-DD') and END_DATE > to_date('1998-04-01','YYYY-MM-DD') and room = 2

13. alter table patient enable all triggers

หลังจากทำ Operation ต่างๆ ข้างต้นแล้ว ข้อมูลจะถูกเปลี่ยนแปลงดังตารางที่ 4.3

**ตารางที่ 4.3** ตาราง patient หลังจากทำการ Update แบบ Sequenced ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Son thaya | 1 | '01-JAN-98' | '7 - FEB- 98' | 200 | '17-FEB-98' | '22-MAR-98' | 1 | '27-FEB- 98' | '12-MAR-98' |
| | | | 1 | '18-FEB- 98' | '23-MAR-98' | 200 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 1 | '23-MAR-98' | '31-DEC-9999' | 2 | '17-FEB- 98' | '1-MAR-98' | 1 | '09-APR- 98' | '21-MAY-98' |
| | | | 2 | '01-JAN-98' | '31-DEC-9999' | 2 | '01-APR- 98' | '10-MAY-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | 6 | '01-JAN-98' | '23-MAR-98' | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '29-JUN-98' | '17-OCT-98' | | | |
| | | | | | | 2 | '17-OCT- 98' | '19-OCT-98' | | | |

**ตารางที่ 4.3** ตาราง patient หลังจากทำการ Update แบบ Sequenced ในส่วนที่เปลี่ยนแปลง (ต่อ)

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| | | | | | | 2 | '19-OCT- 98' | '31-DEC-9999' | | | |

ตัวอย่างการ Update แบบ Current

Update patient set

doctor =6 ,

sname='sekhanan' ,

room = 200 = 2 where id=01 and room = 2 and doctor ='

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. alter table patient disable all triggers

2. insert into the(select doctor from patient patient where patient.id =01 ) select 6,CURRENT_DATE, vt.END_DATE from patient patient1 ,table(patient1.doctor)vt where vt.doctor = 1 and patient1.id =01 and vt.FROM_DATE <= CURRENT_DATE and vt.END_DATE > CURRENT_DATE

3. update the(select doctor from patient patient where patient.id =01 ) set END_DATE = CURRENT_DATE where doctor <> 6 and doctor = 1 and FROM_DATE < CURRENT_DATE and END_DATE > CURRENT_DATE

4. update the(select doctor from patient patient where patient.id =01 ) set doctor = 6 where FROM_DATE >= CURRENT_DATE and doctor = 1

5. update patient patient set patient.sname ='sekhanan' where patient.id =01

6. insert into the(select room from patient patient where patient.id =01 ) select distinct 200,CURRENT_DATE,to_date('9999-12-31','YYYY-MM-DD') from patient patient1 ,table (patient1.room)vt where exists( select * from patient patient2 ,table(patient2.room)vt2 where vt2.room = 2 and patient2.id =01 and vt2.END_DATE = to_date('9999-12-31','YYYY-MM-DD') )

7. update the(select room from patient patient where patient.id =01 ) set END_DATE = CURRENT_DATE where room = 2 and room <> 200 and END_DATE = to_date('9999-12-31','YYYY-MM-DD')

8. alter table patient enable all triggers

**ตารางที่ 4.4** ตาราง patient หลังจากทำการ Update แบบ Current ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Son thaya | 1 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB-98' | '22-MAR-98' | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | 1 | '23-MAR-98' | '27-JUL-2004' | 2 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '01-APR-98' | '10-MAY-98' | 1 | '09-APR-98' | '21-MAY-98' |
| | | | 6 | '27- JUL-2004' | ' 31-DEC-9999' | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | 2 | '29-JUN-98' | '17-OCT-98' | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '17-OCT-98' | '19-OCT-98' | | | |

ตารางที่ 4.4 ตาราง patient หลังจากทำการ Update แบบ Current ในส่วนที่เปลี่ยนแปลง (ต่อ)

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| | | | | | | 2 | '19-OCT-98' | '27-JUL-2004' | | | |
| | | | | | | 200 | '27-JUL-2004' | '31-DEC-9999' | | | |

ตัวอย่างการ Update แบบ Nonsequenced

```
NONSEQUENCED ValidTime UPDATE patient
        SET doctor= 2
        WHERE id=01 and doctor =1;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

```
update the(select doctor from patient patient where patient.id =01 )
set doctor =2 where doctor = 1
```

**ตารางที่ 4.5** ตาราง patient หลังจากทำการ Update แบบ Nonsequenced ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Boon | 2 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB- 98' | '22-MAR-98' | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | 2 | '23-MAR-98' | '31-DEC-9999' | 2 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '01-APR- 98' | '10-MAY-98' | 1 | '09-APR- 98' | '21-MAY-98' |
| | | | | | | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | 2 | '29-JUN-98' | '17-OCT-98' | 1 | '13 -SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '17-OCT-98' | '19-OCT-98' | | | |
| | | | | | | 2 | '19-OCT-98' | '31-DEC-9999' | | | |

```
NONSEQUENCED ValidTime  UPDATE patient
        SET doctor PERIOD '[1998-02-07 - 1998-02-18)'= 2
        WHERE id=01 and  doctor  = 1 ;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. alter table patient disable all triggers
2. update the(select doctor from patient patient where patient.id =01 )
   set doctor =2, FROM_DATE = to_date('1998-02-07','YYYY-MM-DD') ,END_DATE =
   to_date('1998-02-18','YYYY-MM-DD')
   where doctor = 1;
3. alter table patient enable all triggers

**ตารางที่ 4.6** ตาราง patient หลังจากทำการ Update แบบ Nonsequenced ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Boon | 2 | '07-FEB-98' | '18-FEB-98' | 2 | '17-FEB-98' | '22-MAR-98' | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | 2 | '07-FEB-98' | '18-FEB-98' | 2 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '01-APR-98' | '10-MAY-98' | 1 | '09-APR-98' | '21-MAY-98' |
| | | | | | | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | 2 | '29-JUN-98' | '17-OCT-98' | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '17-OCT-98' | '19-OCT-98' | | | |
| | | | | | | 2 | '19-OCT-98' | '31-DEC-9999' | | | |

**การ Delete ข้อมูลโดยใช้คำสั่ง KTSQL ที่กำหนดขึ้นมา**

การลบข้อมูลในฐานข้อมูลจะแบ่งเป็น 3 ประเภทด้วยกันคือข้อมูลแบบ Sequenced , Current Restricted และ Nonsequenced ซึ่งจะอธิบายรายละเอียดดังต่อไปนี้

1. การ Delete แบบ Sequenced จะต้องมี Keyword ด้านหน้าคำสั่ง Delete ว่า VALIDTIME และมีคำว่า Period ตามหลังคำว่า VALIDTIME และระบุเงื่อนไข Attribute ที่ต้องการเปลี่ยนแปลงแก้ไขข้อมูล
2. การ Delete แบบ Current จะแบ่ง Scenarios เป็น 2 แบบด้วยกันคือ Restricted Scenario และ General Scenario ซึ่ง
   - การ Delete ข้อมูลแบบ Restricted Scenario จะสามารถทำการ Delete ข้อมูลใน Temporal Attribute ที่เป็นแบบ Current Restricted เท่านั้น
   - การ Delete ข้อมูลแบบ General Scenario จะสามารถทำการ Delete ข้อมูลใน Temporal Attribute ที่เป็นแบบ Sequenced เท่านั้น
3. การ Delete แบบ Nonsequenced จะไม่สนใจช่วงเวลาในการเปลี่ยนแปลงข้อมูลในตารางสามารถใส่ Period ตามหลังชื่อ Temporal Attribute ที่ต้องการเปลี่ยนแปลงข้อมูลได้เลยหรือจะเปลี่ยนแปลงค่าที่ต้องการโดยไม่สนใจช่วงเวลาทั้ง Current และ Sequenced ก็ได้

ตัวอย่างการ Delete แบบ Sequenced

```
validtime period '[1998-02-25 - 1998-03-10)'
Delete from patient
where doctor=3 or doctor= 1 or doctor =2 and id =3 and room = 02;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. insert into the(select doctor from patient patient where patient.id = 3 ) nt select vt.doctor ,to_date('1998-03-10','YYYY-MM-DD'),vt.END_DATE from patient patient1,table (patient1.doctor ) vt where patient1.id = 3 and vt.doctor =3 or vt.doctor = 1 or vt.doctor = 2 and vt.FROM_DATE <= to_date('1998-02-25','YYYY-MM-DD') and vt.END_DATE > to_date('1998-03-10','YYYY-MM-DD')
2. update the(select doctor from patient patient where patient.id = 3 ) nt set END_DATE = to_date('1998-02-25','YYYY-MM-DD') where FROM_DATE < to_date('1998-02-

25','YYYY-MM-DD') and END_DATE >= to_date('1998-02-25','YYYY-MM-DD') and doctor =3 or doctor = 1 or doctor = 2

3. update the(select doctor from patient patient where patient.id = 3 ) nt set FROM_DATE = to_date('1998-03-10','YYYY-MM-DD') where FROM_DATE < to_date('1998-03-10','YYYY-MM-DD') and END_DATE >= to_date('1998-03-10','YYYY-MM-DD') and doctor =3 or doctor = 1 or doctor = 2
4. delete the(select doctor from patient patient where patient.id = 3 ) nt where FROM_DATE >= to_date('1998-02-25','YYYY-MM-DD') and END_DATE <= to_date ('1998-03-10','YYYY-MM-DD') and doctor =3 or doctor = 1 or doctor = 2
5. insert into the(select room from patient patient where patient.id = 3 ) nt select vt.room ,to_date('1998-03-10','YYYY-MM-DD'),vt.END_DATE from patient patient1,table (patient1.room ) vt where patient1.id = 3 and vt.room = 02 and vt.FROM_DATE <= to_date('1998-02-25','YYYY-MM-DD') and vt.END_DATE > to_date('1998-03-10','YYYY-MM-DD')
6. update the(select room from patient patient where patient.id = 3 ) nt set END_DATE = to_date('1998-02-25','YYYY-MM-DD') where FROM_DATE < to_date('1998-02-25','YYYY-MM-DD') and END_DATE >= to_date('1998-02-25','YYYY-MM-DD') and room = 02
7. update the(select room from patient patient where patient.id = 3 ) nt set FROM_DATE = to_date('1998-03-10','YYYY-MM-DD') where FROM_DATE < to_date('1998-03-10','YYYY-MM-DD') and END_DATE >= to_date('1998-03-10','YYYY-MM-DD') and room = 02
8. delete the(select room from patient patient where patient.id = 3 ) nt where FROM_DATE >= to_date('1998-02-25','YYYY-MM-DD') and END_DATE <= to_date('1998-03-10','YYYY-MM-DD') and room = 02

ตารางที่ 4.7 ตาราง patient หลังจากทำการ Delete แบบ Sequenced ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 3 | k. Chat | Mariwan | 4 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '17-FEB- 98', | '25-FEB-98' | 3 | '17-FEB- 98' | '22-MAR-98' |
| | | | | | | 2 | '22-MAR-98' | '01-APR-98' | 1 | '22-MAR-98' | '01-APR-98' |
| | | | | | | 2 | '10-MAY-98' | '22-MAR-98' | | | |

ตัวอย่างการ Delete แบบ Current

Delete from patient where id =1 and doctor =1 and room = 02;

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. update the(select doctor from patient patient where patient.id = 1 ) nt set END_DATE = CURRENT_DATE where END_DATE >= CURRENT_DATE and FROM_DATE < CURRENT_DATE and doctor = 1
2. delete the(select doctor from patient patient where patient.id = 1 ) nt where FROM_DATE > CURRENT_DATE and doctor = 1
3. update the(select room from patient patient where patient.id = 1 ) nt set END_DATE = CURRENT_DATE where END_DATE = to_date('9999-12-31' ,'YYYY-MM-DD') and room = 02

**ตารางที่ 4.8** ตาราง patient หลังจากทำการ Delete แบบ Current ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Boon | 1 | '01-JAN-98' | '23-MAR-98' | 2 | '17-FEB-98' | '22-MAR-98' | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | 1 | '23-MAR-98' | '05– SEP - 04' | 2 | '22-MAR-98' | '01-APR-98' | 1 | '12-MAR-98' | '09-APR-98' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | 2 | '01-APR-98' | '10-MAY-98' | 1 | '09-APR-98' | '21-MAY-98' |
| | | | | | | 2 | '10-MAY-98' | '29-JUN-98' | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | 2 | '29-JUN-98' | '17-OCT-98' | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | | | | 2 | '17-OCT-98' | '19-OCT-98' | | | |
| | | | | | | 2 | '19-OCT-98' | '05-SEP-04' | | | |

ตัวอย่างการ Delete แบบ Nonsequenced

Nonsequenced ValidTime Delete from patient where id =1 and doctor=3 and room =02;

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆได้ดังต่อไปนี้

1. delete the(select doctor from patient patient where patient.id = 1 ) nt where doctor =3
2. delete the(select room from patient patient where patient.id = 1 ) nt where room = 02

ตารางที่ 4.9 ตาราง patient หลังจากทำการ Delete แบบ Nonsequenced ในส่วนที่เปลี่ยนแปลง

| ID | Name | SNAME | DOCTOR | | | ROOM | | | BED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | DOCTOR | FROM_DATE | END_DATE | ROOM | FROM_DATE | END_DATE | BED | FROM_DATE | END_DATE |
| 1 | K.B | Boon | 2 | '01-JAN-98' | ' 31-DEC-9999' | | | | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | | | | | | | 1 | '12-MAR-98' | '09-APR-98' |
| | | | | | | | | | 1 | '09-APR- 98' | '21-MAY-98' |
| | | | | | | | | | 1 | '21-MAY-98' | '13-SEP-98' |
| | | | | | | | | | 1 | '13-SEP-98' | '31-DEC-9999' |
| | | | 2 | '01-JAN-98' | ' 31-DEC-9999' | | | | 1 | '27-FEB-98' | '12-MAR-98' |
| | | | | | | | | | 1 | '12-MAR-98' | '09-APR-98' |

**การ เรียกดูข้อมูลโดยใช้คำสั่ง KTSQL ที่กำหนดขึ้นมา**

การเรียกดูข้อมูลในฐานข้อมูลจะแบ่งเป็น 3 แบบด้วยกันคือข้อมูลแบบ Sequenced , Current และ Nonsequenced ซึ่งจะอธิบายรายละเอียดดังต่อไปนี้

1. การเรียกดูข้อมูลแบบ Sequenced จะต้องมี Keyword ด้านหน้าคำสั่ง Select ว่า VALIDTIME จะเป็นการเรียกดูข้อมูลที่ต้องการดูเวลาที่เป็น Temporal Attribute และสามารถระบุช่วงเวลาที่ต้องการดูด้วยโดยระบุ Keyword Period และตามด้วยช่วงเวลา
2. การเรียกดูข้อมูลแบบ Current จะแสดงข้อมูลที่อยู่ในช่วงเวลาปัจจุบันใน Temporal Attribute
3. การเรียกดูข้อมูลแบบ Nonsequenced จะไม่แสดงข้อมูลที่เกี่ยวข้องกับช่วงเวลาใน Temporal Attribute

ซึ่งการเรียกดูข้อมูลนี้จะขอแสดงผลในบทต่อไป

ตัวอย่างการเรียกดูข้อมูลแบบ Sequenced ระบุเงื่อนไขในส่วนที่ไม่ใช่ Temporal Attribute โดยเรียกดู ข้อมูลของทั้งตาราง

```
validtime select * from patient where id = 3;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select patient.id , patient.name , patient.sname , patientdoctor_nt.doctor , patientdoctor_nt.FROM_DATE , patientdoctor_nt.END_DATE from patient patient , table(patient.doctor) patientdoctor_nt where patient.id = 3
2. select patient.id , patient.name , patient.sname , patientroom_nt.room , patientroom_nt.FROM_DATE , patientroom_nt.END_DATE from patient patient , table(patient.room) patientroom_nt where patient.id = 3
3. select patient.id , patient.name , patient.sname , patientbed_nt.bed , patientbed_nt.FROM_DATE , patientbed_nt.END_DATE from patient patient , table(patient.bed) patientbed_nt where patient.id = 3

ตัวอย่างการเรียกดูข้อมูลแบบ Sequenced โดยเรียกดูข้อมูลของตาราง patient ในช่วงเวลตั้งแต่วันที่ 23 เมษายน ปี ค.ศ. 1998 ถึงวันที่ 23 เมษายน ปี ค.ศ. 1998

```
validtime period '[1998-04-23 - 1998-04-24)' select * from patient p ;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select p.id , p.name , p.sname , pdoctor_nt.doctor , pdoctor_nt.FROM_DATE , pdoctor_nt.END_DATE from patient p , table(p.doctor) pdoctor_nt where pdoctor_nt.END_DATE >= to_date('1998-04-24','YYYY-MM-DD') and pdoctor_nt.FROM_DATE <= to_date('1998-04-23','YYYY-MM-DD')
2. select p.id , p.name , p.sname , proom_nt.room , proom_nt.FROM_DATE , proom_nt.END_DATE from patient p , table(p.room) proom_nt where proom_nt.END_DATE >= to_date('1998-04-24','YYYY-MM-DD') and proom_nt.FROM_DATE <= to_date('1998-04-23','YYYY-MM-DD')
3. select p.id , p.name , p.sname , pbed_nt.bed , pbed_nt.FROM_DATE , pbed_nt.END_DATE from patient p , table(p.bed) pbed_nt where pbed_nt.END_DATE >= to_date('1998-04-24','YYYY-MM-DD') and pbed_nt.FROM_DATE <= to_date('1998-04-23','YYYY-MM-DD')

ตัวอย่างการเรียกดูข้อมูลแบบ Sequenced เป็นการเรียกดูข้อมูลของตาราง patient ในช่วงเวลตั้งแต่วันที่ 27 มีนาคมวันที่ 27 มีนาคม ปี ค.ศ. 1998 ถึงมษายน ปี ค.ศ. 1998 โดยใช้การ Join

```
validtime period '[1998-03-27 - 1998-04-01)'
select * from patient p , patient p1 where p.id= p1.id;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select p.id , p.name , p.sname , pdoctor_nt.doctor , GREATEST(pdoctor_nt.FROM_DATE,p1doctor_nt.FROM_DATE) , LEAST( pdoctor_nt.END_DATE , p1doctor_nt.END_DATE) , p1.id , p1.name , p1.sname , p1doctor_nt.doctor , GREATEST( pdoctor_nt.FROM_DATE, p1doctor_nt.FROM_DATE ) , LEAST( pdoctor_nt.END_DATE , p1doctor_nt.END_DATE ) from patient p , patient p1 , table(p.doctor) pdoctor_nt , table (p1.doctor) p1doctor_nt where p.id = p1.id and pdoctor_nt.doctor=p1doctor_nt.doctor and GREATEST( pdoctor_nt.FROM_DATE ,p1doctor_nt.FROM_DATE)< LEAST (pdoctor_nt.END_DATE,p1doctor_nt.END_DATE) and pdoctor_nt.END_DATE >= to_date('1998-04-01','YYYY-MM-DD') and pdoctor_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD') and p1doctor_nt.END_DATE >= to_date('1998-04-

01','YYYY-MM-DD') and p1doctor_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD')

2. select p.id , p.name , p.sname , proom_nt.room , GREATEST(proom_nt.FROM_DATE,p1room_nt.FROM_DATE) , LEAST (proom_nt.END_DATE,p1room_nt.END_DATE) , p1.id , p1.name , p1.sname , p1room_nt.room , GREATEST(proom_nt.FROM_DATE,p1room_nt.FROM_DATE) , LEAST(proom_nt.END_DATE,p1room_nt.END_DATE) from patient p , patient p1 , table(p.room) proom_nt , table(p1.room) p1room_nt where p.id = p1.id and proom_nt.room = p1room_nt.room and GREATEST(proom_nt.FROM_DATE,p1room_nt.FROM_DATE)< LEAST (proom_nt.END_DATE,p1room_nt.END_DATE) and proom_nt.END_DATE >= to_date('1998-04-01','YYYY-MM-DD') and proom_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD') and p1room_nt.END_DATE >= to_date('1998-04-01','YYYY-MM-DD') and p1room_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD')

3. select p.id , p.name , p.sname , pbed_nt.bed , GREATEST(pbed_nt.FROM_D_DAT1bed_nt.FROM_DATE) , LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE) , p1.id , p1.name , p1.sname , p1bed_nt.bed , GREATEST(pbed_nt.FROM_DATE,p1bed_nt.FROM_DATE) , LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE) from patient p , patient p1 , table(p.bed) pbed_nt , table(p1.bed) p1bed_nt where p.id = p1.id and pbed_nt.bed=p1bed_nt.bed and GREATEST(pbed_nt.FROM_DATE,p1bed_nt.FROM_DATE)< LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE) and pbed_nt.END_DATE >= to_date ('1998-04-01','YYYY-MM-DD') and pbed_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD') and p1bed_nt.END_DATE >= to_date('1998-04-01','YYYY-MM-DD') and p1bed_nt.FROM_DATE <= to_date('1998-03-27','YYYY-MM-DD')

**ตัวอย่างการเรียกดูข้อมูลแบบ Sequenced เป็นการเรียกดูข้อมูลของตาราง patient โดยไม่ระบุช่วงเวลาและใช้การ Join**

validtime select * from patient p ,patient p1 where p1.id < p.id;

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select p.id , p.name , p.sname , pdoctor_nt.doctor , GREATEST(pdoctor_nt.FROM_DATE,p1doctor_nt.FROM_DATE) , LEAST( pdoctor_nt.END_DATE , p1doctor_nt.END_DATE) , p1.id , p1.name , p1.sname , p1doctor_nt.doctor , GREATEST(pdoctor_nt.FROM_DATE,p1doctor_nt.FROM_DATE) , LEAST (pdoctor_nt.END_DATE,p1doctor_nt.END_DATE) from patient p , patient p1 , table (p.doctor) pdoctor_nt , table(p1.doctor) p1doctor_nt where p1.id < p.id and pdoctor_nt.doctor=p1doctor_nt.doctor and GREATEST( pdoctor_nt.FROM_DATE , p1doctor_nt.FROM_DATE)< LEAST(pdoctor_nt.END_DATE,p1doctor_nt.END_DATE)
2. select p.id , p.name , p.sname , proom_nt.room , GREATEST(proom_nt.FROM_DATE,p1room_nt.FROM_DATE) , LEAST (proom_nt.END_DATE,p1room_nt.END_DATE) , p1.id , p1.name , p1.sname , p1room_nt.room , GREATEST(proom_nt.FROM_DATE,p1room_nt.FROM_DATE) , LEAST(proom_nt.END_DATE,p1room_nt.END_DATE) from patient p , patient p1 , table(p.room) proom_nt , table(p1.room) p1room_nt where p1.id < p.id and proom_nt.room=p1room_nt.room and GREATEST( proom_nt.FROM_DATE , p1room_nt.FROM_DATE)< LEAST(proom_nt.END_DATE,p1room_nt.END_DATE)
3. select p.id , p.name , p.sname , pbed_nt.bed , GREATEST(pbed_nt.FROM_DATE,p1bed_nt.FROM_DATE) , LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE) , p1.id , p1.name , p1.sname , p1bed_nt.bed , GREATEST(pbed_nt.FROM_DATEOM_DATE,p1bed_nt.FRO , LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE) from patient p , patient p1 , table(p.bed) pbed_nt , table(p1.bed) p1bed_nt where p1.id < p.id and pbed_nt.bed=p1bed_nt.bed and GREATEST( pbed_nt.FROM_DATE , p1bed_nt.FROM_DATE ) < LEAST (pbed_nt.END_DATE,p1bed_nt.END_DATE)

ตัวอย่างการเรียกดูข้อมูลแบบ Current โดยเรียกดูข้อมูลโดยใช้การ Join

select * from patient p ,patient p1 where p.doctor=2 and p1.id < p.id;

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select p.id , p.name , p.sname , pdoctor_nt.doctor , pdoctor_nt.FROM_DATE , pdoctor_nt.END_DATE , p1.id , p1.name , p1.sname , p1doctor_nt.doctor , p1doctor_nt.FROM_DATE , p1doctor_nt.END_DATE from patient p , patient p1 , table (p.doctor) pdoctor_nt , table(p1.doctor) p1doctor_nt where pdoctor_nt.doctor =2 and p1.id < p.id and pdoctor_nt.END_DATE=to_date('9999-12-31','YYYY-MM-DD') and p1doctor_nt.END_DATE=to_date('9999-12-31','YYYY-MM-DD')
2. select p.id , p.name , p.sname , proom_nt.room , proom_nt.FROM_DATE , proom_nt.END_DATE , p1.id , p1.name , p1.sname , p1room_nt.room , p1room_nt.FROM_DATE , p1room_nt.END_DATE from patient p , patient p1 , table (p.room) proom_nt , table(p1.room) p1.room_nt where p1.id < p.id and proom_nt.END_DATE = to_date( '9999-12-31' , 'YYYY-MM-DD' ) and p1room_nt.END_DATE = to_date( '9999-12-31' , 'YYYY-MM-DD' )
3. select p.id , p.name , p.sname , pbed_nt.bed , pbed_nt.FROM_DATE , pbed_nt.END_DATE , p1.id , p1.name , p1.sname , p1bed_nt.bed , p1bed_nt.FROM_DATE , p1bed_nt.END_DATE from patient p , patient p1 , table (p.bed) pbed_nt , table(p1.table(p1.bed) where p1.id < p.id and pbed_nt.END_DATE = to_date('9999-12-31','YYYY-MM-DD') and p1bed_nt.END_DATE=to_date('9999-12-31','YYYY-MM-DD')

ตัวอย่างการเรียกดูข้อมูลแบบ Current โดยเรียกดูข้อมูลทั้งตาราง

```
select * from patient ;
```

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

1. select patient.id , patient.name , patient.sname , patientdoctor_nt.doctor , patientdoctor_nt.FROM_DATE , patientdoctor_nt.END_DATE from patient patient , table(patient.doctor) patientdoctor_nt where patientdoctor_nt.END_DATE=to_date ('9999-12-31','YYYY-MM-DD')

2. select patient.id , patient.name , patient.sname , patientroom_nt.room , patientroom_nt.FROM_DATE , patientroom_nt.END_DATE from patient patient , table (patient.room) patientroom_nt where patientroom_nt.END_DATE=to_date('9999-12-31','YYYY-MM-DD')
3. select patient.id , patient.name , patient.sname , patientbed_nt.bed , patientbed_nt.FROM_DATE , patientbed_nt.END_DATE from patient patient , table (patient.bed) patientbed_nt where patientbed_nt.END_DATE=to_date('9999-12-31','YYYY-MM-DD')

ตัวอย่างการเรียกดูข้อมูลแบบ Nonsequenced

Nonsequenced ValidTime select id , name , sname , doctor from patient where doctor=4;

จากคำสั่งข้างต้นสามารถ Map เป็น Operation ต่างๆ ได้ดังต่อไปนี้

select patient.id , patient.name , patient.sname , patientdoctor_nt.doctor from patient patient , table(patient.doctor) patientdoctor_nt where patientdoctor_nt.doctor =4

# บทที่ 5

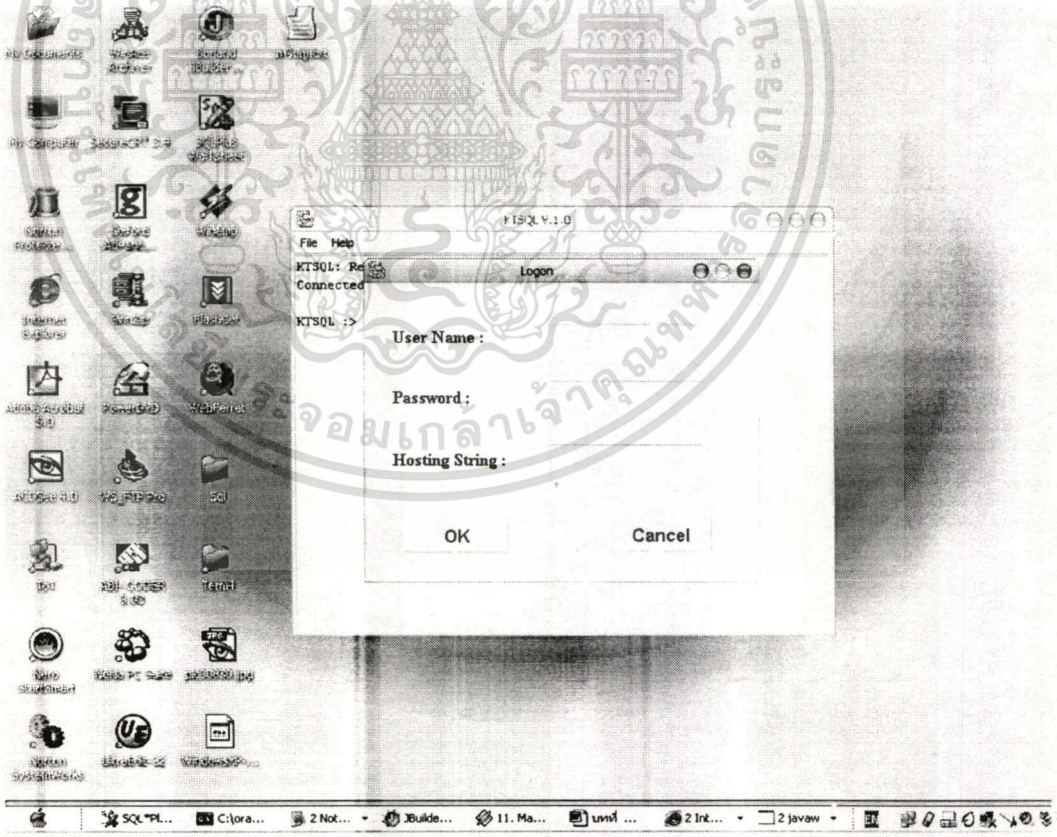
## การพัฒนาระบบงาน KTSQL

### 5.1 การออกแบบหน้าจอ

การออกแบบหน้าจอนั้นจะแบ่งเป็น 2 ส่วนด้วยกันคือ

1. หน้าจอในการ LogOn จะทำหน้าที่รับข้อมูลของ User ที่ต้องการเข้าใช้ระบบงาน KTSQL
2. หน้าจอในการรับคำสั่ง KTSQL และแสดงผลจะเป็นหน้าจอเดียวกัน ซึ่งคำสั่งที่รับนั้นสามารถดูได้จากบทที่ 3 และมีคำสั่งที่ใช้ Clear Screen โดยใช้คำสั่ง clr

โดยแต่ละส่วนจะมีรายละเอียดดังต่อไปนี้

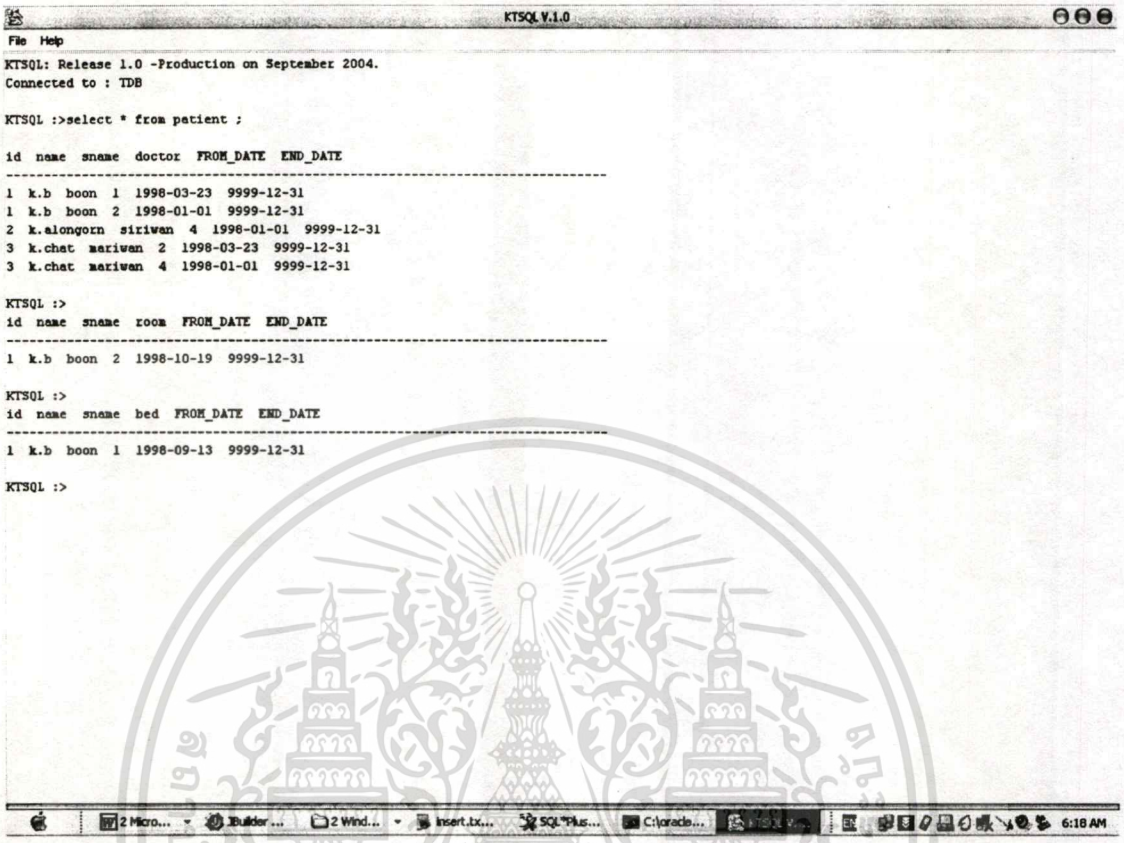



**รูปที่ 5.1** หน้าจอในการ LogOn เข้าใช้ระบบฐานข้อมูลKTSQL

จากรูปที่ 5.1 จะเป็นหน้าจอที่ใช้ในการ LogOn เข้าเพื่อใช้ระบบโดยผู้ใช้จะต้องทำการใส่ Username password และ Hosting String

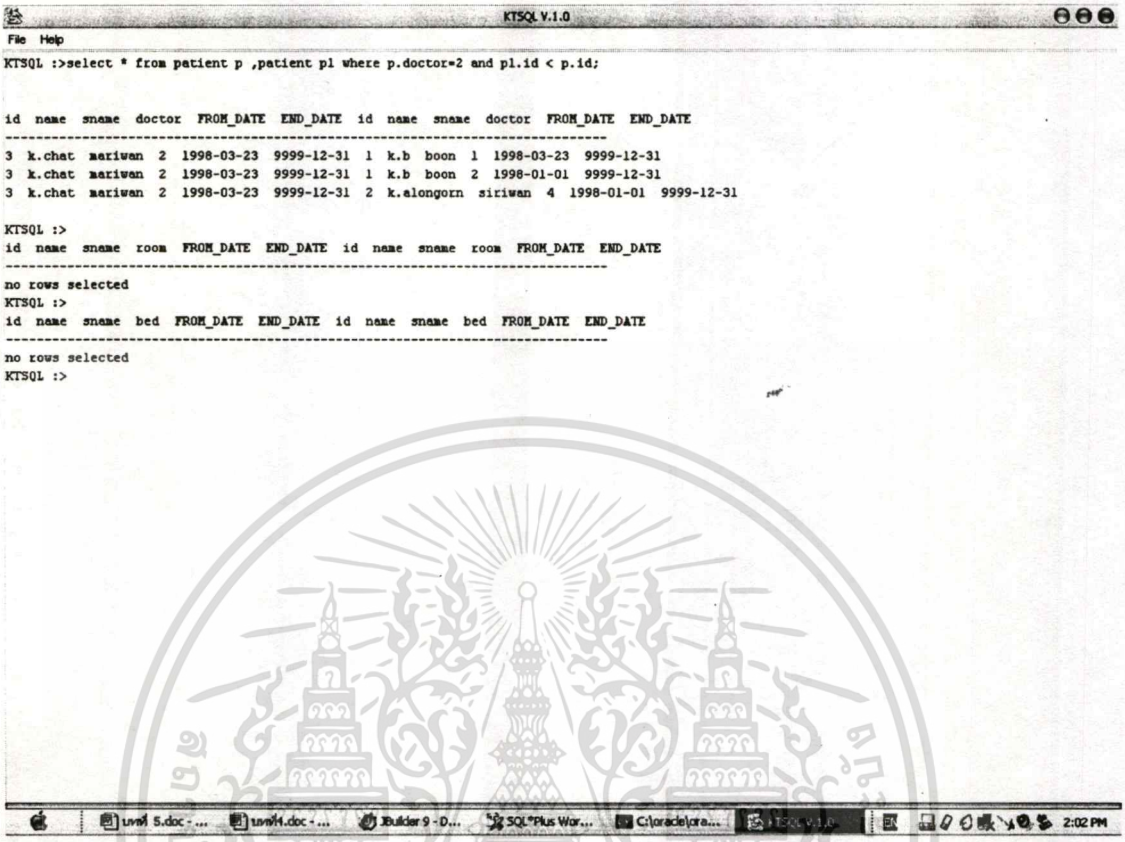

**รูปที่ 5.2 หน้าจอรับคำสั่ง KTSQLและแสดงผล**

จากรูปที่ 5.2 จะเป็นหน้าจอที่ใช้ในการรับคำสั่ง KTSQL จาก User เพื่อส่งไปประมวลผล



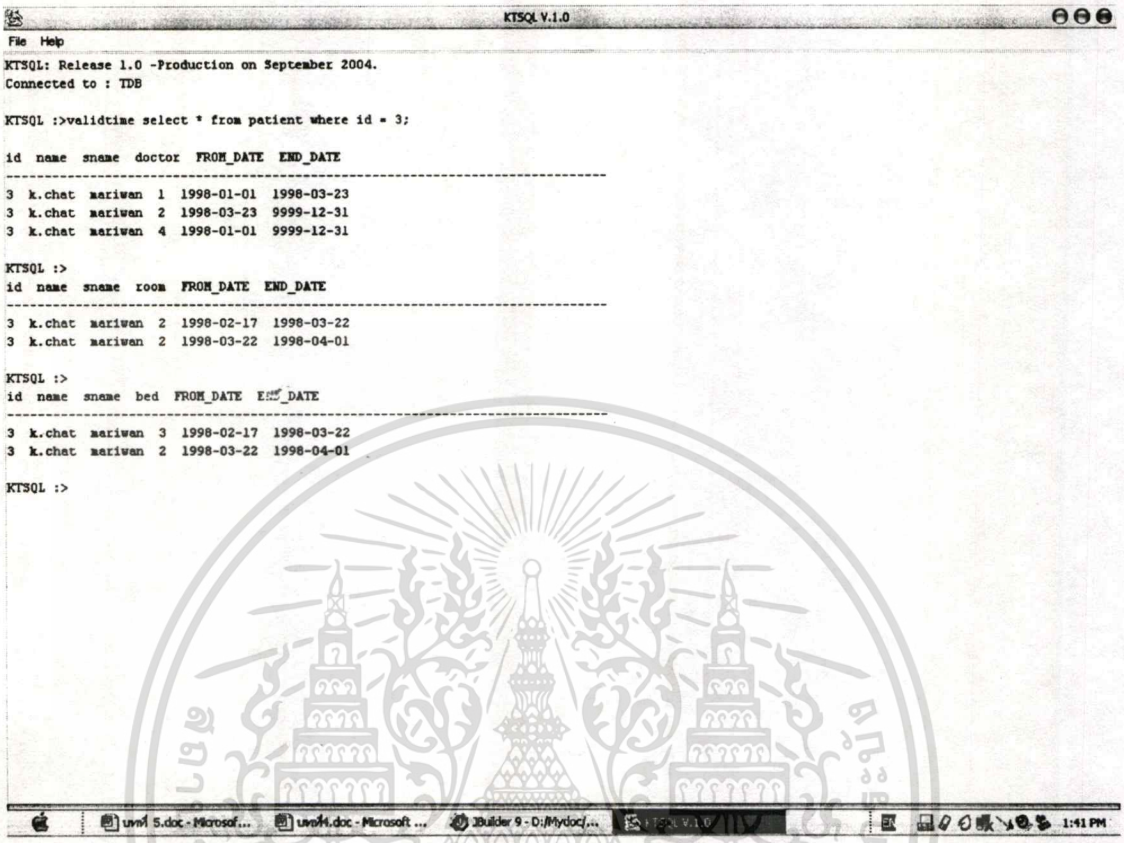

**รูปที่ 5.3** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Current

จากรูปที่ 5.3 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Current เพื่อเรียกดูข้อมูลในตาราง patient โดยมี Temporal Attribute อยู่ 3 Temporal Attribute ด้วยกัน ซึ่งจะแสดงถึงข้อมูลที่มี fact อยู่ในช่วงเวลาปัจจุบัน



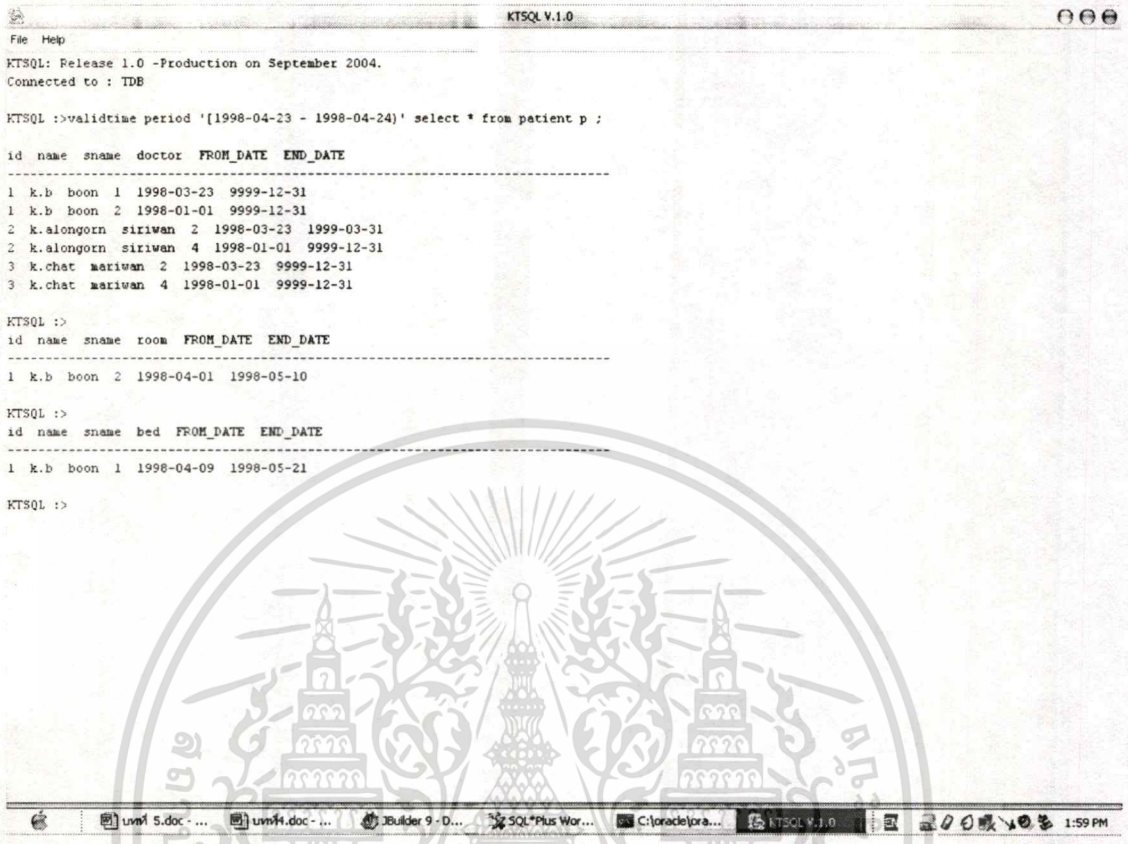

**รูปที่ 5.4** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Current

จากรูปที่ 5.4 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Current ระบุเงื่อนไขโดยมี id เป็น Key Attribute มี doctor เป็น Temporal Attribute และทำการ Self Join ซึ่งจะแสดงผลลัพธ์ตามที่ Select Attribute ทั้งหมดของตาราง โดยจะตรวจสอบเงื่อนไขเฉพาะ Temporal Attribute ที่ถูกระบุไว้ใน Where Condition ส่วน Temporal Attribute อื่นที่ไม่ถูกระบุเงื่อนไขจะถูกแสดงออกตาม Temporal Keyword



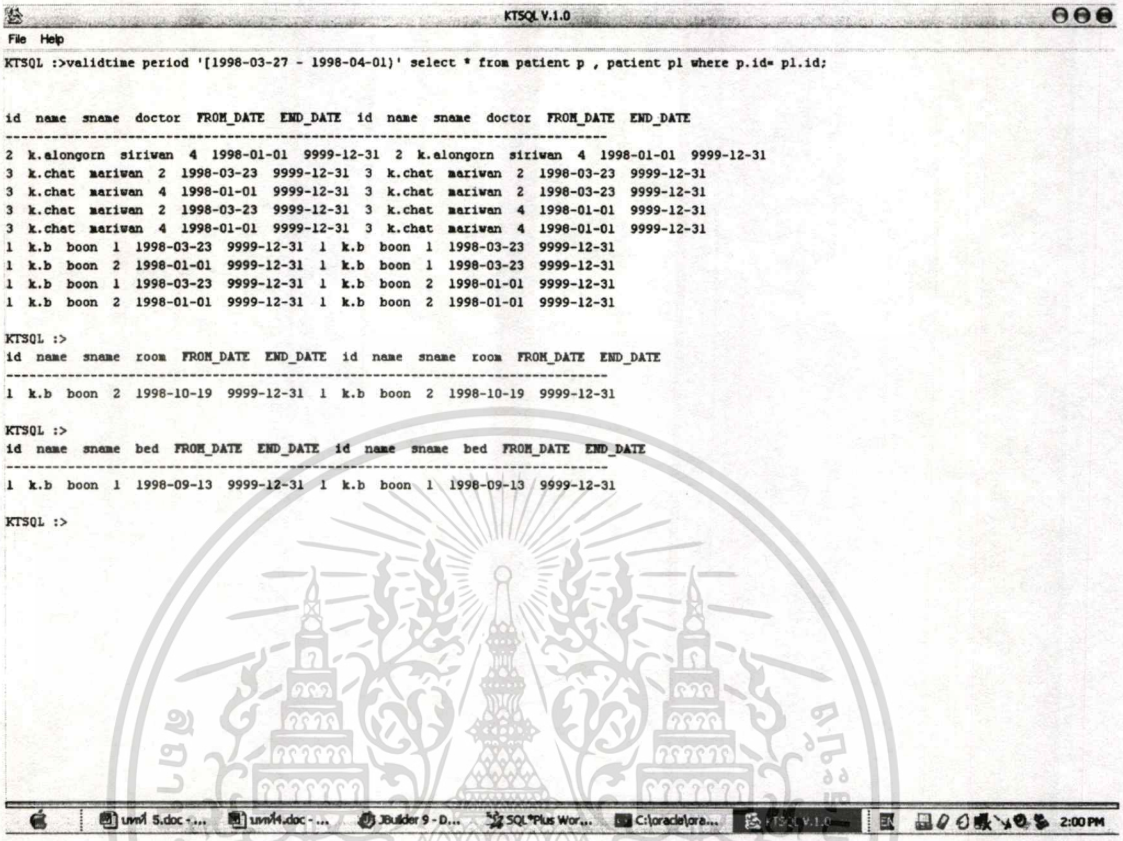

**รูปที่ 5.5 หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Sequenced**

จากรูปที่ 5.5 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Sequenced ระบุเงื่อนไข โดยมี id เป็น Key Attribute ซึ่งจะแสดงผลลัพธ์ตามที่ Select Attribute ทั้งหมดของตาราง โดยจะไม่ตรวจสอบเงื่อนไขใน Temporal Attribute ดังนั้น Temporal Attribute ที่ไม่ถูกระบุเงื่อนไขจะถูกแสดงออกตาม Temporal Keyword เท่านั้น



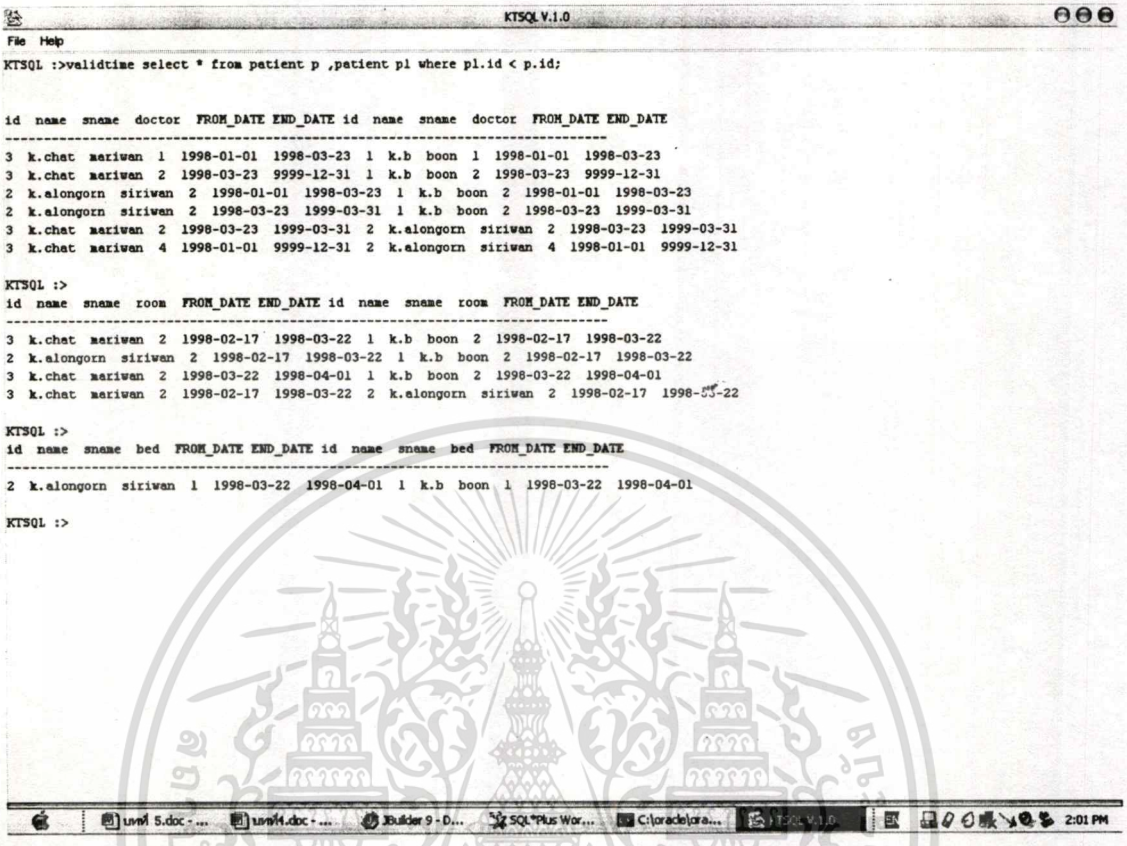

**รูปที่ 5.6** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Sequenced

จากรูปที่ 5.6 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Sequenced โดยทำการระบุช่วงเวลาที่ต้องการค้นหา ซึ่งจะแสดงผลลัพ ตามที่ Select Attribute ทั้งหมดของตาราง โดยจะไม่ตรวจสอบเงื่อนไขใน Temporal Attribute ที่ไม่ถูกระบุเงื่อนไขจะถูกแสดงออกตาม Temporal Ke word ทั้งหมด



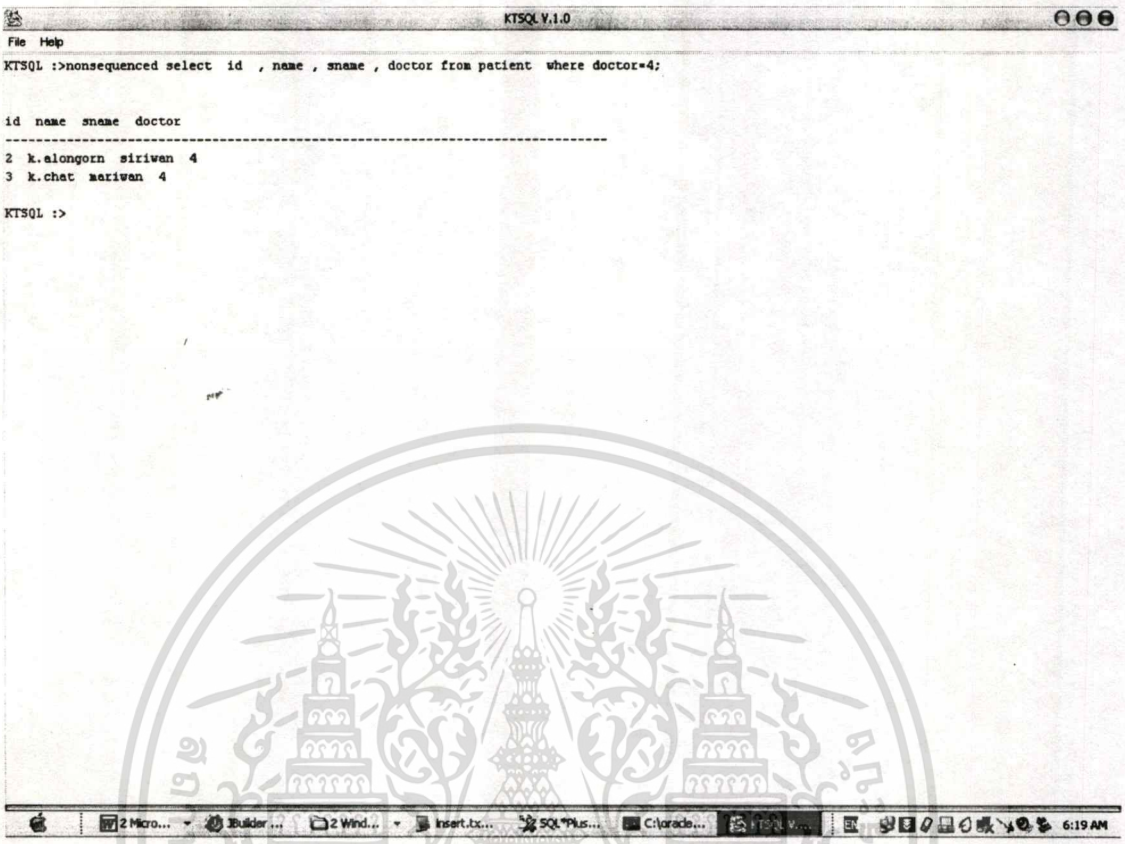

**รูปที่ 5.7** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Sequenced

จากรูปที่ 5.7 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Sequenced โดยทำการระบุช่วงเวลาที่ต้องการค้นหา ระบุเงื่อนไขโดยมี id เป็น Key Attribute และทำ Self Join ซึ่งจะแสดงผลลัพธ์ตามที่ Select Attribute ทั้งหมดของตาราง โดยจะไม่ตรวจสอบเงื่อนไขใน Temporal Attribute ที่ไม่ถูกระบุเงื่อนไขจะถูกแสดงออกตาม Temporal Keyword เท่านั้น




**รูปที่ 5.8** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Sequenced

จากรูปที่ 5.8 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Sequenced โดยระบุเงื่อนไข ซึ่งมี id เป็น Key Attribute และทำ Self Join ซึ่งจะแสดงผลลัพธ์ตามที่ Select Attribute ทั้งหมดของตาราง โดยจะไม่ตรวจสอบเงื่อนไขใน Temporal Attribute ที่ไม่ถูกระบุเงื่อนไขจะถูกแสดงออกตาม Temporal Keyword เท่านั้น




**รูปที่ 5.9** หน้าจอรับคำสั่ง KTSQL โดยส่งคำสั่ง Select แบบ Nonsequenced

จากรูปที่ 5.9 เป็นหน้าจอที่แสดงผลจากคำสั่ง Select แบบ Nonsequenced โดยระบุเงื่อนไขเฉพาะ Temporal Attribute doctor = 4

# บทที่ 6

## บทสรุปและข้อเสนอแนะ

### 6.1 บทสรุป

การพัฒนาระบบงานนี้เป็นความพยายามเพื่อจะให้เกิดประโยชน์ในการจัดการข้อมูลเชิงเวลา เนื่องจากภาษาที่ใช้ในการจัดการฐานข้อมูลเชิงเวลายังไม่มีใช้จริงในระบบการจัดการฐานข้อมูล ดังนั้นจึงได้ทำการพัฒนาระบบงาน KTSQL นี้ เพื่อให้ผู้ใช้งานทำงานกับข้อมูลเชิงเวลาได้ง่ายดายขึ้นกว่าการใช้ภาษา Relational SQL หรือ การใช้ภาษา Object Relational SQL ในการจัดการข้อมูลเชิงเวลา

ผู้ใช้สามารถศึกษา Syntax ของภาษา KTSQL ที่ถูกกำหนดขึ้นมาใหม่เพื่อรองรับแนวคิดการจัดการข้อมูลเชิงเวลารูปแบบใหม่ ซึ่งมีรูปแบบคำสั่งคล้ายกับภาษา SQL ทั่วไป โดยที่ผู้ใช้ต้องมีการปรับเปลี่ยนแนวคิดจากเดิมที่ใช้ภาษา SQL เล็กน้อยเท่านั้น โดยที่การพัฒนาระบบนี้เป็นการ Implement ในระดับ Attribute ซึ่งแตกต่างจากเดิมที่ Implement ในระดับ record และการพัฒนาระบบนี้ทำให้มีความสามารถที่เพิ่มขึ้นจากเดิมมากขึ้น ในประเด็นแรกการออกแบบฐานข้อมูลทั่วไปจะออกแบบเพื่อไม่ให้มีความซ้ำซ้อนของข้อมูลเกิดขึ้นดังนั้นถ้ามีช่วงเวลาเข้ามาเกี่ยวข้องในแต่ละ Fact จะต้องทำการแยกตารางออกจากกันเพื่อลดความซ้ำซ้อน ซึ่งการใช้แนวคิดใหม่นี้ไม่จำเป็นต้องแยกตารางออกเนื่องจากทำการ Implement ในระดับ Attribute โดยใช้แนวคิดของ Nested Table ซึ่งทำการแยก Fact ของเวลาออกมาฝังอยู่ในระดับ Attribute ประเด็นที่ 2 การแก้ไขข้อมูลทั้งการ Update และ Delete จะทำการเปลี่ยนแปลงได้ทั้งในระดับ Attribute และ ระดับ Record อย่างเช่นคำสั่ง delete จะสามารถลบข้อมูลได้ทั้งตาราง หรือจะลบข้อมูลในระดับ Attribute หรือจะลบข้อมูลในระดับ Record ประเด็นที่ 3 การที่ไม่ต้องทำการแยกตารางออก เพื่อลดความซ้ำซ้อนนั้นสามารถให้เข้าถึงข้อมูลได้เร็วยิ่งขึ้น เนื่องจากไม่ต้องใช้คำสั่งในการ Join ตาราง และไม่ต้องเขียนคำสั่งในการ Join ซึ่งจะยุ่งยากกว่า

## 6.2 ปัญหาและอุปสรรค

จากการที่นำแนวคิดการจัดการเชิงวัตถุสัมพันธ์มาใช้ในการพัฒนาฐานข้อมูลเชิงเวลา ได้พบปัญหา ซึ่งไม่สามารถใช้ Syntax ของภาษา TSQL ได้เนื่องจากการ Implement ของแนวคิดเดิมเป็นการจัดการในระดับ Record ซึ่งจำเป็นที่จะต้องเปลี่ยนแปลงให้เหมาะสมกับแนวคิดการจัดการเชิงวัตถุสัมพันธ์โดยใช้แนวคิด Nested Table จะ Implement ในระดับ Attribute จึงจำเป็นต้องกำหนด Syntax ขึ้นมาใหม่ให้เหมาะสมในการพัฒนาระบบนี้ยังมีส่วนที่ยังไม่สมบูรณ์ในส่วนของการ Query ข้อมูลโดยที่ต้องการดูข้อมูลทั้งตารางโดยใช้เครื่องหมาย ' * ' ซึ่งในตารางมีข้อมูลที่เป็น Temporal Attribute อยู่หลาย Attribute จะทำการสืบค้นข้อมูลขึ้นมาตามเงื่อนไขที่ระบุตาม Temporal Attribute และจะทำการแสดงผลที่เป็น Fact ในเงื่อนไขที่ระบุไว้ ซึ่งควรจะแก้ไขให้แสดงผลให้สัมพันธ์กันทุกๆ Fact

## 6.3 ข้อเสนอแนะ

1. รูปแบบในการแสดงผลควรจะทำให้ผู้ใช้ดูง่ายขึ้น
2. ข้อจำกัดในการแสดงผลของ KTSQL version 1.0 ในเรื่องของคำสั่ง Query นั้นจำเป็นต้องมีการปรับปรุงการแสดงผล เนื่องจากถ้าต้องการดูข้อมูลทั้งตาราง ซึ่งในตารางนั้นมีหลาย Temporal Attribute จะทำการสืบค้นข้อมูลแยกกันออกเป็นแต่ละ fact ของ Temporal Attribute เลย ซึ่งควรจะ fact ทั้งหมดสอดคล้องกันตามเงื่อนไขที่ระบุทั้งหมด

# บรรณานุกรม


วิทวัส พันธุมจินดา. 2543. **ระบบฐานข้อมูลเชิงวัตถุ.** [Online] .
เข้าถึงได้จาก : http://www.uni.net.th/~09_2543/lesson14/ms1t2.htm.

วีรวดี ขุขันธิน,ร.อ.หญิง. 2542. "การนำ Temporal Database มาใช้กับงานเก็บประวัติการรักษาพยาบาล." รายงานวิชาสัมมนา1., คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ.

ศีพัฒน์ นามวัฒน์,ร.ต. 2542. **การศึกษาระบบฐานข้อมูลเชิงวัตถุสัมพันธ์.** [Online] .
Available:http://www.rtafa.ac.th/article/ORDB.htm

Gangopadhyay Aryya.2000. **OODBMS vs ORDBMS.** [Online] .
Available:http://research.umbc.edu/~gangopad/620/oodb/tsld001.htm.

Jaymee et.al. 1997. **Object-Relational and Object-Oriented Data Base Management Systems.** [Online].
Available:http://www.sims.berkeley.edu/courses/is206/f97/GroupE/OOORDBMS/sld001.htm

Kevin Loney and George Koch. 2000. **Oracle 8i The Complete Reference**.McGraw-Hill 2000

Oracle Corporation. 2004. **Oracle9i SQL Reference Release 1 (9.0.1) Part Number A90125-01** [Online].
Available: http://download-west.oracle.com/docs/cd/A91202_01/901_doc/server.901/a90125/index.htm

Snodgrass, T. Richard. 1998. **Managing Temporal Data A Five Part-Series**. [Online].
Available:www.cs.arizona.edu/~rts/DBPD/collection.pdf.

# ประวัติผู้เขียน

| | |
|---|---|
| **ชื่อผู้เขียน** | นายอิฏฐา เสขะนันทน์ |
| **วันเดือนปีเกิด** | 26 กันยายน 2517 |
| **สถานที่เกิด** | กรุงเทพมหานคร |
| **ปริญญาตรี** | นิเทศศาสตร์บัณฑิต<br>ภาควิชาการโฆษณา<br>คณะนิเทศศาสตร์<br>มหาวิทยาลัยกรุงเทพ |
| **ปีที่สำเร็จการศึกษา** | ปีการศึกษา 2540 |