คลังข้อมูลจากฐานข้อมูลเชิงเวลา

TEMPORAL DATA WAREHOUSE

ชิดชนก โพธิภัทร

ธิดารัตน์ พูนเพชรพันธ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

คลังข้อมูลจากฐานข้อมูลเชิงเวลา

TEMPORAL DATA WAREHOUSE

ชิดชนก โพธิภัทร

ธิดารัตน์ พูนเพชรพันธ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

ปริญญานิพนธ์ปีการศึกษา 2558

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง คลังข้อมูลจากฐานข้อมูลเชิงเวลา

TEMPORAL DATA WAREHOUSE

ผู้จัดทำ

1. นางสาวชิดชนก โพธิภัทร รหัสนักศึกษา 55010283
2. นางสาวธิดารัตน์ พูนเพชรพันธ์ รหัสนักศึกษา 55010570

อาจารย์ที่ปรึกษา

(รศ. ดร. ศุภมิตร จิตตะยโศธร)

คลังข้อมูลจากฐานข้อมูลเชิงเวลา

นางสาวชิดชนก โพธิภัทร 55010283
นางสาวธิดารัตน์ พูนเพชรพันธ์ 55010570
รศ. ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2558

บทคัดย่อ

โดยทั่วไปของการสร้างคลังข้อมูล (Data warehouse) จะเป็นการรวบรวมข้อมูลเข้าจากฐานข้อมูลแบบไม่เชิงเวลา (Non-temporal database) โดยเอามาจากแหล่งข้อมูลที่หลากหลาย เช่น แฟ้มข้อมูลทรานแซกชั่นหรือฐานข้อมูลระดับปฏิบัติการ (Operational database) รูปแบบต่างๆ อย่างไรก็ตามหากแหล่งข้อมูลเป็นฐานข้อมูลเชิงเวลา (Temporal Database) ยังไม่ปรากฏว่ามีการศึกษาการแปลงข้อมูลของฐานข้อมูลเชิงเวลามาเป็น ข้อมูลสำหรับใช้กับคลังข้อมูลที่มีมิติเวลา ซึ่งปัจจุบันมีการนำฐานข้อมูลเชิงเวลามาใช้งานในเชิงพาณิชย์อย่างแพร่หลาย การที่ใช้คลังข้อมูลที่สามารถเก็บข้อมูลเชิงเวลาได้นั้น เนื่องจากเป็นฐานข้อมูลเชิงเวลาทำให้มีความสามารถในการสืบค้นข้อมูลในอดีต ปัจจุบันหรืออนาคตได้ตามต้องการ

โครงงานนี้ศึกษาการสร้างคลังข้อมูลที่ข้อมูลทั้งหมดเกิดจากการใช้จากฐานข้อมูลเชิงเวลาในฐานะฐานข้อมูลระดับปฏิบัติการ (Temporal Operational Database) ให้เป็นแหล่งข้อมูลของระบบคลังข้อมูล จึงไม่ต้องจำเป็นต้องทำการนำเข้าข้อมูลใหม่ ซึ่ง โดยปกติในคลังข้อมูลแบบทั่วไปจะเป็นข้อมูลที่ไม่ได้เป็นข้อมูล ณ เวลาปัจจุบัน (Real Time) ซึ่งข้อมูลจะเป็นล่าสุดก็ต่อเมื่อมีการนำข้อมูลเข้ามาจากแหล่งข้อมูลต่างๆ โครงงานนี้ยังได้ศึกษาการนำคุณสมบัติของออราเคิลเวิคสเปสเมเนเจอร์ (Oracle Workspace Manager) ของผลิตภัณฑ์ฐานข้อมูลออราเคิลดาต้าเบส (Oracle Database) พร้อมทั้งนำเสนอสถาปัตยกรรมระบบคลังข้อมูลจากฐานข้อมูลเชิงเวลาดังกล่าวเป็นการเพิ่มความสามารถให้กับคลังข้อมูล โดยเฉพาะการจัดการมิติที่มีการเปลี่ยนแปลงเชิงเวลา (Slow Changing Dimension)

Temporal Data Warehouse

Ms. Chitchanok Potipat 55010283

Ms. Thidarat Poonpetchphan 55010570

Assoc. Prof. Dr. Suphamit Chittayasothorn Advisor

Academic Year 2015

ABSTRACT

In general, a data warehouse is a collection of data which is collected from non-temporal databases. These input data sources include transaction files and other form of operational databases. However, if the source is a temporal database, there appears to be no previous literatures on the subject matter. Nowadays, there are the temporal databases available commercially. Owing to the fact that the source is a temporal database, it has the ability to query a data in the past, present or future whenever we want. Since the data warehouse has the dimension time which is extracted from the temporal database, a transformation is required.

This project studies about the data warehouse which uses the temporal databases as sources. Commonly, the data which is collected from the general data warehouse is not real time data. Therefore, the data will be updated when importing the recent data from other sources. Moreover, this project has been studied about the feature of Oracle Workspace Manager of Oracle database. Besides, the project is presented about the architecture of the temporal data warehouse which can increase the ability to the data warehouse especially in the handling of slow changing dimensions.

กิตติกรรมประกาศ

โครงงานคลังข้อมูลจากฐานข้อมูลเชิงเวลา (Temporal Data Warehouse) ฉบับนี้ ไม่อาจสำเร็จลุล่วงไปได้ด้วยดี หากปราศจากความช่วยเหลือ คำแนะนำ และความอนุเคราะห์จาก รศ.ดร. ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานนี้ ข้าพเจ้ารู้สึกซาบซึ้งใจที่อาจารย์คอยติดตามความก้าวหน้า และปรับปรุงแก้ไขการทำโครงงานมาตั้งแต่เริ่มต้นจนจบ

ขอขอบคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนข้าพเจ้า ทำให้ข้าพเจ้าสามารถนำความรู้ที่เรียนมาประยุกต์ใช้กับโครงงานได้

ขอขอบคุณรุ่นพี่ ตลอดจนเพื่อนๆ ทุกคนที่มีส่วนในการช่วยเหลือ ให้กำลังใจ และแลกเปลี่ยนความรู้ ความคิด ช่วยเหลือ และแนะนำในการทำโครงงานนี้

ขอขอบคุณครอบครัวของข้าพเจ้าที่อบรม สั่งสอน ให้โอกาสทางการศึกษา ให้กำลังใจ และสนับสนุนข้าพเจ้าในทุกๆ ด้านมาโดยตลอด

สุดท้ายนี้ขอขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ให้โอกาสข้าพเจ้าได้เข้ามาศึกษาหาความรู้ ข้าพเจ้ารู้สึกเป็นเกียรติอย่างยิ่ง คุณความดีใดๆ ที่ปรากฏในโครงงานนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่านมา ณ ที่นี้ด้วย

ชิดชนก โพธิภัทร
ธิดารัตน์ พูนเพชรพันธ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ข้อมูลเป็นสิ่งสำคัญในการดำเนินงานต่างๆ เปรียบเสมือนเป็นเครื่องมือที่ช่วยในการตัดสินใจขององค์กร ไม่ว่าจะเป็นแวดวงธุรกิจ ภาครัฐ หรือว่าเอกชน ต่างก็มีข้อมูลที่ต้องนำมาใช้วิเคราะห์หรือใช้ในการวางแผนงาน การมีข้อมูลที่มากมายนั้น หากขาดระบบการจัดการที่ดี ก็อาจทำให้เข้าถึงข้อมูลนั้นได้ลำบาก หรือหากข้อมูลที่ใช้วิเคราะห์มีความผิดพลาดอาจก่อให้เกิดความเสียหายต่อองค์กรได้ ทำให้ต้องทำการบริหารจัดการคลังข้อมูล จึงเป็นที่มาของของการทำคลังข้อมูล จากฐานข้อมูลเชิงเวลาดังนั้นการนำคลังข้อมูล (Data Warehouse) เข้ามาใช้จัดเก็บข้อมูล จึงเป็นประโยชน์อย่างมาก เนื่องจากข้อมูลที่นำเข้ามาจัดเก็บนั้น ล้วนผ่านการคัดกรองแล้วทั้งสิ้น อีกทั้งยังสามารถจัดเก็บข้อมูลเฉพาะส่วนที่องค์กรสนใจแยกไว้ได้ (Data Mart) จึงทำให้ง่ายต่อการนำข้อมูลมาใช้งาน และยังสะดวกต่อการนำข้อมูลมาใช้วิเคราะห์เพื่อการตัดสินใจของผู้บริหารอีกด้วย แต่ข้อเสียของการทำคลังข้อมูลแบบทั่วไปคือ ไม่สามารถที่จะนำเข้า (Extract) ข้อมูลโดยมีคาบเวลาได้ กล่าวคือคาบเวลาในการนำเข้าข้อมูลนั้นอาจถูกจำกัดจากผู้ที่ทำการนำเข้าว่าจะมีความสามารถในการนำเข้าข้อมูลในช่วงเวลาที่ถี่ได้มากน้อยแค่ไหน อีกทั้งการรวมข้อมูลเหล่านั้นมาสร้างเป็นคลังข้อมูลยังมีความยากลำบาก เนื่องจากเมื่อเวลาผ่านไป รูปแบบในการจัดเก็บข้อมูลอาจจะมีการเปลี่ยนแปลง ผู้ทำคลังข้อมูลจึงจำเป็นต้องทำการคัดกรองข้อมูลและกำหนดให้ข้อมูลที่ดึง (Load) มาจากทุกช่วงเวลาอยู่ในรูปแบบเดียวกัน อีกทั้งในคลังข้อมูลแบบทั่วไปจะจัดเก็บข้อมูลที่ไม่ใช่ข้อมูล ณ เวลาปัจจุบัน (ข้อมูลไม่ Real time) หรือกล่าวได้ว่าข้อมูลไม่ได้มีการอัพเดท (Update) อยู่ตลอดเวลา ข้อมูลจะอัพเดท (Update) ล่าสุดก็ต่อเมื่อมีการนำข้อมูลเข้ามาจากแหล่งข้อมูลต่างๆ

โครงงานนี้จึงจัดทำขึ้นเพื่อศึกษาทดลอง และนำความรู้ที่ได้จากการค้นคว้าเกี่ยวกับเทคโนโลยีที่มีอยู่ในฐานข้อมูลออราเคิล (Oracle Database) นั่นคือออราเคิลเวิคสเปสเมเนเจอร์ (OWM: Oracle Workspace Manager) มาเป็นเครื่องมือใช้ในการสร้างคลังข้อมูล เนื่องจากสามารถนำข้อมูลเชิงเวลา (Temporal Data) มาใช้งานในรูปแบบของคลังข้อมูลได้ ซึ่งในโครงงานนี้จะศึกษาการทำคลังข้อมูลที่มีแหล่งข้อมูล (Source) เป็นฐานข้อมูลเชิงเวลา (Temporal Database) หรือกล่าวคือ เป็นคลังข้อมูลที่ครอบอยู่บนฐานข้อมูลเชิงเวลา ซึ่งสามารถแก้ปัญหาที่ไม่ต้องนำเข้าข้อมูล เพราะแหล่งข้อมูลทั้งหมดอยู่ในคลังข้อมูลแล้ว อีกทั้งยังเป็นข้อมูล ณ เวลาปัจจุบันอีกด้วย ดังนั้นจะสามารถเรียกดูข้อมูลในอดีต ปัจจุบันหรืออนาคตได้ตามต้องการ เนื่องจากคลังข้อมูลมีแกนเวลา (Dimension Time) ที่ดึงมาจากฐานข้อมูลเชิงเวลาได้นั่นเอง

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อศึกษาขีดความสามารถของ OWM
2) เพื่อศึกษาการประยุกต์ใช้ OWM กับคลังข้อมูล
3) เพื่อเพิ่มความสามารถให้แก่การทำ คลังข้อมูล รูปแบบทั่วไป ในการใช้ข้อมูลเชิงเวลามาประยุกต์ใช้กับคลังข้อมูล
4) เพื่อให้ผู้ใช้งานสามารถวิเคราะห์ข้อมูลโดยสรุปเป็นช่วงเวลาต่างๆ (Time Granularity) ตามที่ผู้ใช้ต้องการได้โดยใช้แกนเวลาได้เสมือนกับคลังข้อมูลทั่วไป
5) เพื่อให้สามารถนำข้อมูลในอดีต ปัจจุบัน อนาคต (ข้อมูลที่เป็นการวางแผนล่วงหน้า) มาประกอบการวิเคราะห์ตามเรื่องที่องค์กรสนใจ โดยไม่ได้มีเพียงข้อมูลที่เป็นค่าปัจจุบันเพียงอย่างเดียว

1.3 ขอบเขตของโครงงาน

ในขั้นแรกจะศึกษาความสามารถของ OWM ในการที่จะทำเวอร์ชั่น (Version) ของข้อมูลในช่วงเวลาต่างๆ จากนั้นนำความรู้ที่ได้จากการศึกษามาพัฒนาเพื่อทำคลังข้อมูลที่มีแหล่งข้อมูล และตัวคลังข้อมูลอยู่บนในระบบเดียวกัน โดยจะไม่มีการโหลดแหล่งข้อมูลมาจากแหล่งเก็บข้อมูลอื่น ซึ่งจะใช้ข้อมูลเชิงเวลามาประยุกต์ใช้กับคลังข้อมูลจากนั้นนำแหล่งข้อมูลมาสรุปเป็นคลังข้อมูลที่ประกอบไปด้วยตารางความจริง (Fact Table) และตารางมุมมอง (Dimension Table) ตามหลักเกณฑ์ในการสร้าง คลังข้อมูลทั่วไป

1.4 วิธีการดำเนินงาน

1) ศึกษาทฤษฎีเรื่องฐานข้อมูลเชิงเวลา
2) ศึกษาการใช้งาน OWM
3) ศึกษาทฤษฎีการออกแบบและการใช้งานคลังข้อมูล
4) ศึกษาคิดค้นอัลกอริทึมในการสรุปแหล่งข้อมูลที่นำมาสร้างเป็นคลังข้อมูล
5) วิเคราะห์และออกแบบโปรแกรมประยุกต์ที่จะใช้สร้างคลังข้อมูล
6) พัฒนาโปรแกรมประยุกต์ในการใช้ข้อมูลเชิงเวลาในการสร้างคลังข้อมูล
7) ค้นคว้าเอกสารงานวิจัยที่ได้รับการตรวจสอบแล้ว (Literature Review) ควบคู่ไปกับการทำโครงงานเพื่อที่จะได้ทราบว่ามีผู้ที่คิดโจทย์ในการแก้ปัญหาเช่นเดียวกับโครงงานของเราหรือไม่ และมีวิธีแก้ไขปัญหานั้นอย่างไร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้ ความเข้าใจเกี่ยวกับฐานข้อมูลเชิงเวลา
2) ได้รับความรู้ ความเข้าใจเกี่ยวกับขีดความสามารถในการนำข้อมูลจากฐานข้อมูลเชิงเวลามาใช้งานใน OWM ได้
3) ได้รับความรู้ ความเข้าใจเกี่ยวกับทฤษฎีคลังข้อมูล
4) สามารถนำความรู้ที่ได้ไปออกแบบ และพัฒนาคลังข้อมูลที่มีขีดความสามารถมากกว่าคลังข้อมูลแบบทั่วไปได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ฐานข้อมูลเชิงเวลา (Temporal Database)

ฐานข้อมูลเชิงเวลาคือ ฐานข้อมูลที่มีความสามารถในการจัดเก็บข้อมูลที่เปลี่ยนแปลงไปตามเวลา เช่น ชื่อ ที่อยู่ สถานที่ทำงาน เป็นต้น ซึ่งในเวลาต่อมาได้มีการถกเถียงกันว่า ข้อมูลบางอย่างไม่ได้เป็นข้อมูลที่เปลี่ยนแปลงไปตามเวลาแต่เปลี่ยนแปลงไปตามความสามารถหรือเทคโนโลยีในการประเมินค่าของมนุษย์ เช่น ความสูงของยอดเขา ความสว่างของดวงดาวเป็นต้น ดังนั้นความหมายที่อธิบายคำว่าระบบฐานข้อมูลเชิงเวลาได้ดีที่สุดคือ ฐานข้อมูลที่นำเสนอแง่มุมทางเวลาโดยไม่ถือรวมเวลาที่ประกอบเป็นความจริงส่วนหนึ่งของข้อมูล

ในยุคเริ่มแรกได้มีการถกเถียงกันเกี่ยวกับคุณสมบัติทางเวลา (Timing Attribute) ว่าสมควรจะมองเสมือนคุณสมบัติธรรมดา หรือควรมองแยกออกมาเป็นอีกมิติและยกภาระทั้งหมดให้กับระบบการจัดการฐานข้อมูลเป็นของ (DBMS :Database Management System) ซึ่งแนวคิดอันหลังนี้ค่อนข้างจะเป็นที่นิยมกว่าเนื่องจากการบริหารจัดการข้อมูลเกี่ยวกับช่วงเวลานั้นมีความยุ่งยากเป็นอย่างมาก ซึ่งจะนำเสนอให้เห็นภาพดังต่อไปนี้

ในประวัติศาสตร์ประมาณปี 1996 ประเทศอเมริกาได้พบว่าเนื้อวัวที่ส่งขายจำนวนหนึ่งได้ติดเชื้อโรคและเชื้อโรคได้ระบาดในเนื้อวัวเหล่านั้น ซึ่งผู้จำหน่ายเนื้อวัวเหล่านั้นไม่สามารถทราบได้เลยว่าเนื้อจำนวนนั้น มาจากโรงฆ่าสัตว์ไหน และมีวัวตัวไหนที่อยู่คอกเดียวกันในเวลาเดียวกันบ้าง โดยตั้งสมมติฐานที่ว่า วัวที่อยู่คอกเดียวกัน น่าจะเป็นวัวที่ติดเชื้อชนิดเดียวกัน ซึ่งผู้จำหน่ายเนื้อวัวเหล่านี้ไม่สามารถตอบคำถามเหล่านี้ได้เลย เป็นผลให้ต้องเรียกคืนเนื้อวัวทั้งหมดที่จำหน่ายโดยบริษัทนี้ในประเทศอเมริกาทำให้บริษัทนี้ สูญเสียรายได้เป็นอย่างมาก จึงเกิดแนวคิดขึ้นว่าหากเราสามารถสอบถามข้อมูลย้อนกลับไปได้ว่าเนื้อแต่ละชิ้นมาจากวัวตัวไหน คอกไหน หรือแม้แต่กระทั้งวัวตัวใดเคยอยู่คอกเดียวกันกับวัวตัวที่ติดเชื้อเหล่านี้บ้าง จะทำให้ลดการสูญเสียรายได้มหาศาลของบริษัทนี้ลงได้

จากเหตุการณ์ดังกล่าวเราสามารถนำฐานข้อมูลเชิงเวลาเข้ามาแก้ได้ปัญหาที่เกิดจากเหตุการณ์นี้ได้ ดังจะกล่าวถึงต่อไป แต่เริ่มแรกต้องเข้าใจรูปแบบของฐานข้อมูลเชิงเวลาก่อน ดังนี้

2.1.1 ประเภทและความหมายของเวลา

ในการจัดการกับฐานข้อมูลเชิงเวลานั้น จะต้องมีการระบุเวลาในฐานข้อมูล ซึ่งประเภทของเวลาที่ระบุในฐานข้อมูลมี 3 ประเภท ดังนี้

1) Valid Time คือ เวลาที่แสดงความสัมพันธ์ที่เป็นจริงของข้อมูล (เวลาที่ค่าความจริงเป็นจริง) หรือเวลาที่ข้อมูลนั้นเป็นจริงในฐานข้อมูล ซึ่งต้องมีการกำหนดค่าความเป็นจริงตั้งแต่เมื่อไหร่ถึงเมื่อไหร่
2) Transaction Time คือ เวลาที่ข้อมูลนั้นถูกบันทึกลงในฐานข้อมูล ซึ่งจัดการโดย DBMS ทำหน้าที่ในการบันทึกข้อมูลอัตโนมัติ
3) User-defined Time คือ เวลาที่เป็นส่วนหนึ่งของค่าความจริง (Fact) ถึงแม้จะเป็นข้อมูลประเภทเวลา แต่ระบบจะมองเป็นแค่ข้อมูลธรรมดา เช่น วันเกิด วันจบการศึกษา เป็นต้น

2.1.2 ความซ้ำซ้อนของข้อมูลเชิงเวลา

ระบบฐานข้อมูลเชิงเวลาที่ดี ไม่ควรมีความซ้ำซ้อนของข้อมูลเกิดขึ้น เพราะจะส่งผลให้เกิดปัญหาในการเพิ่มข้อมูล (Insert) การปรับปรุงข้อมูล (Update) และการลบข้อมูล (Delete) เพื่อแสดงให้เห็นความซ้ำซ้อนของข้อมูลในรูปแบบต่างๆ จึงยกตัวอย่างตารางที่แสดงสถานะของคนไข้ของโรงพยาบาลเด็กแห่งหนึ่ง ดังตาราง 2.1

ตาราง 2.1 สถานะของคนไข้ในโรงพยาบาลแห่งหนึ่ง[1]

Name	Status	From_date	To_date
Kenneth Robert	Serious	1997-11-19	1997-11-21
Alexis May	Serious	1997-11-19	1997-11-27
Natalie Sue	Serious	1997-11-19	1997-11-25
Kelsey Ann	Serious	1997-11-19	1997-11-26
Brandon James	Serious	1997-11-19	1997-11-26
Nathan Roy	Serious	1997-11-19	1997-11-28
Joel Steven	Critical	1997-11-19	1997-11-20

[1] Managing Temporal Data. A Five-Part Series

ตาราง 2.1 สถานะของคนไข้ในโรงพยาบาลแห่งหนึ่ง (ต่อ)[1]

Name	Status	From_date	To_date
Joel Steven	Serious	1997-11-20	1997-11-26
Kenneth Robert	fair	1997-11-21	1998-01-03
Alexis May	Fair	1997-11-27	1998-01-11
Alexis May	Fair	1997-12-02	9999-12-31
Alexis May	Fair	1997-12-02	9999-12-31

จากตารางข้างต้นเป็นตารางเชิงเวลาเพื่อบันทึกสถานะการรักษาผู้ป่วยเด็กเล็ก โดยประกอบด้วย Name แทนชื่อผู้ป่วย Status แทนสถานะอาการ From_date แทนวันที่เริ่มต้นที่เกิดอาการ และ To_date แทนวันที่สิ้นสุดของอาการ ซึ่งอาการของผู้ป่วย มีดังนี้ อาการสาหัส (Serious) อาการวิกฤต (Critical) และอาการปกติ (Fair)

จากตาราง 2.1 จะเห็นได้ว่า Kenneth Robert มีสถานะ Serious ตั้งแต่วันที่ 1997-11-19 ถึง 1997-11-21 และต่อมา Kenneth Robert มีสถานะ Fair ตั้งแต่วันที่ 1997-11-21 ถึง 1998-01-03 จะเห็นได้ว่าวันสิ้นสุดของสถานะ Serious จะเป็นการบันทึกวันเริ่มต้นของสถานะปกติ (Fair) ดังนั้นช่วงเวลาที่ข้อมูลของสถานะ Serious เป็นจริงคือ ตั้งแต่วันที่ 1997-11-19 ถึง 1997-11-20 และตารางตัวอย่างข้างต้นได้แสดงความซ้ำซ้อนของข้อมูลทั้งหมด 4 รูปแบบ คือ ความซ้ำซ้อนแบบนอนซีเควนซ์ (Non Sequence Duplicate) ความซ้ำซ้อนแบบค่าเท่ากัน (Value-Equivalent Duplicate) ความซ้ำซ้อนแบบปัจจุบัน (Current Duplicate) และความซ้ำซ้อนแบบซีเควนซ์ (Sequenced Duplicate) รายละเอียดของความซ้ำซ้อนมีดังต่อไปนี้

1) ความซ้ำซ้อนแบบนอนซีเควนซ์ (Non Sequence Duplicate) คือ ความซ้ำซ้อนของข้อมูลที่เหมือนกันทุกประการ กล่าวคือเหมือนกันทุกคอลัมน์ โดยไม่สนใจคอลัมน์เวลา หรือถือคอลัมน์เวลาเป็นคอลัมน์พิเศษ สามารถแก้ไขปัญหาความซ้ำซ้อนนี้ได้โดยกำหนดคอลัมน์ Name Status From_date และ To_date เป็นคีย์หลัก (Primary Key) ร่วมกัน ซึ่งจะทำให้คนไข้ไม่สามารถมีหลายสถานะเกิดขึ้นในเวลาเดียวกัน แต่การแก้ไขปัญหาโดยวิธีนี้ยังไม่เพียงพอ

[1]Managing Temporal Data. A Five-Part Series

ตาราง 2.2 ตัวอย่างความซ้ำซ้อนแบบ Non Sequenced Duplicates[1]

Name	Status	From_date	To_date
Alexis May	Fair	1997-12-02	9999-12-31
Alexis May	Fair	1997-12-02	9999-12-31

2) ความซ้ำซ้อนแบบค่าเท่ากัน (Value Equivalent Duplicate) คือ ความซ้ำซ้อนของข้อมูลที่ทุกคอลัมน์มีค่าเหมือนกัน ยกเว้นคอลัมน์ช่วงเวลา สามารถแก้ไขปัญหาความซ้ำซ้อนนี้ได้โดยกำหนดคอลัมน์ Name และ Status เป็น Primary Key ร่วมกัน ซึ่งจะทำให้ผู้ป่วยมีได้สถานะเดียวเท่านั้น

ตาราง 2.3 ตัวอย่างความซ้ำซ้อนแบบค่าเท่ากัน (Value Equivalent Duplicates)[1]

Name	Status	From_date	To_date
Alexis May	Fair	1997-11-27	1998-01-11
Alexis May	Fair	1997-12-02	9999-12-31
Alexis May	Fair	1997-12-02	9999-12-31

3) ความซ้ำซ้อนแบบปัจจุบัน (Current Duplicate) คือ ความซ้ำซ้อนของข้อมูลที่ทุกคอลัมน์มีค่าเหมือนกัน ในช่วงเวลา ในเวลาปัจจุบัน หรือกล่าวได้ว่ามี Fact ซ้ำกันในเวลาปัจจุบัน สมมติให้วันนี้เป็นวันที่ 10 มกราคม 1998 สามารถแก้ไขปัญหาความซ้ำซ้อนนี้ได้โดยตรวจสอบการบันทึกข้อมูลให้ถูกต้องก่อน

[1] Managing Temporal Data. A Five-Part Series

ตาราง 2.4 ตัวอย่างความซ้ำซ้อนแบบ Current Duplicates[1]

Name	Status	From_date	To_date
Alexis May	Fair	1997-11-27	1998-01-11
Alexis May	Fair	1997-12-02	9999-12-31
Alexis May	Fair	1997-12-02	9999-12-31

4) ความซ้ำซ้อนแบบซีเควนซ์ (Sequenced Duplicate) คือ ความซ้ำซ้อนของข้อมูลในแต่ละแถวที่ทุกคอลัมน์มีค่าเหมือนกัน ยกเว้นคอลัมน์ช่วงเวลา โดยจะมีค่าที่ซ้ำซ้อนกันในช่วงเวลาใดๆ สามารถแก้ไขปัญหาความซ้ำซ้อนนี้ได้โดยกำหนดเงื่อนไขในการบันทึกข้อมูล เพื่อไม่ให้มีการบันทึกสถานะผู้ป่วยที่เหมือนกันในช่วงเวลาใดๆ

ตาราง 2.5 ตัวอย่างความซ้ำซ้อนแบบ Sequenced Duplicates[1]

Name	Status	From_date	To_date
Alexis May	Fair	1997-11-27	1998-01-11
Alexis May	Fair	1997-12-02	9999-12-31
Alexis May	Fair	1997-12-02	9999-12-31

2.1.3 Valid Time State Table

Valid Time State Table คือตารางที่เพิ่ม Valid Time เข้ามาเพื่อบอกว่าข้อมูลใดที่ยังเป็นจริงอยู่ในฐานข้อมูลเชิงเวลาบ้าง ดังตัวอย่างต่อไปนี้

ตาราง 2.6 Valid Time State Table แสดงรายละเอียดของฝูงวัว[1]

FDYD_ID	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	137	1	17	1998-02-07	1998-02-18
1	219	1	43	1998-02-25	1998-03-01
1	219	1	20	1998-03-01	1998-03-14

[1] Managing Temporal Data. A Five-Part Series

ตาราง 2.6 Valid Time State Table แสดงรายละเอียดของฝูงวัว (ต่อ)[1]

FDYD_ID	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
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

ตารางนี้นำเสนอรายละเอียดของวัวแต่ละฝูงโดยที่ FDYD_ID แทน หมายเลขลานให้อาหารของวัว LOT_ID_NUM แทน หมายเลขฝูงของวัว PEN_ID แทน หมายเลขคอกย่อยของวัว HD_CNT แทน จำนวนวัวในฝูงนั้น และ FROM_DATE TO_DATE แทนวันเวลาที่ข้อมูลนั้นยังเป็นจริง ซึ่งหาก TO_DATE เป็น 9999-12-31 ซึ่งหมายความ ข้อมูลนั้นยังเป็นจริงมาจนถึงปัจจุบัน

จากตางรางข้างต้นแสดงตัวอย่างให้เห็นว่า วัวฝูง 219 นั้นเคยอยู่คอกที่ 1 เมื่อวันที่ 1998-02-25 จนถึงวันที่ 1998-02-28 จำนวนทั้งสิ้น 43 ตัว จากนั้นวันที่ 1998-03-01 ได้แบ่งวัวฝูงนี้เป็น 2 คอก คือคอกที่ 1 จำนวน 20 ตัวและคอกที่ 2 จำนวน 23 ตัวจนถึงวันที่ 1998-03-13 จากนั้นได้จับวัวฝูงนี้มาอยู่รวมที่คอกที่ 2 ด้วยกันอีกครั้งจนถึงปัจจุบัน

2.1.3.1 ตัวอย่างคำถามเชิงเวลาสำหรับข้อมูลเชิงเวลา

1) Temporal Projection and Selection เป็นการสอบถามข้อมูลเชิงเวลาโดยไม่มีการรวม (Join) ระหว่าง 2 ตาราง (ข้อมูลทุกอย่างปรากฏครบถ้วนในตารางเดียว) Current Query ตัวอย่างคำถาม เช่น "ในแต่ละคอกย่อยของลานให้อาหารที่ 1 มีวัวฝูง 219 อยู่คอกใดและมีจำนวนกี่ตัวบ้าง" จะเห็นได้ว่าจากคำถามข้างต้นสามารถใช้ภาษา SQL (Structured Query Language) ธรรมดาในการสอบถามได้ดังโปรแกรมที่ 2.1 แต่หากเป็นฐานข้อมูลเชิงเวลา (เป็น Valid Time State Table) จะต้องระบุช่วงเวลาที่ไปด้วยดังโปรแกรม 2.2 ซึ่งผลลัพธ์ของการ Select ทั้งสองแบบจะได้ผลดังตาราง 2.7 ส่วนการ Sequence Query คำถามประเภทนี้จะเป็นการถามโดยการระบุช่วงเวลาในอดีตจนถึงปัจจุบันเพื่อต้องการทราบประวัติการเปลี่ยนแปลงของข้อมูลเช่น "ในเวลาที่ผ่านมา วัวฝูง 219 เคยอยู่ในคอกย่อยใดบ้างและคอกล่ะกี่ตัว" สามารถเขียน SQL เพื่อตอบคำถามดังกล่าวได้ตามโปรแกรม 2.3 จะเห็นว่าหากเราไม่ระบุเวลาในฐานข้อมูลเชิงเวลาจะมีความหมายเทียบเท่าช่วงเวลาทั้งหมดจากอดีตจนถึงปัจจุบัน ซึ่งคำถามลักษณะนี้ หากเป็นการเก็บลงฐานข้อมูลปกติ จะไม่มีทางตอบคำถามได้

[1] Managing Temporal Data. A Five-Part Series

อย่างแน่นอน โดยจะให้ผลลัพธ์ดังตาราง 2.8 และการทำ Non Sequenced Query ตัวอย่างคำถาม เช่น "มีจำนวนวัวฝูง 219 กี่ตัวที่อยู่ในในลาน 1" สามารถเขียน SQL เพื่อตอบคำถามดังกล่าวได้ดังโปรแกรม 2.4 ซึ่งจะได้ผลลัพธ์ดังตาราง 2.9

โปรแกรม 2.1 การ Query แบบ Current Query โดยไม่ระบุช่วงเวลา

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

โปรแกรม 2.2 การ Query แบบ Current Query โดยระบุช่วงเวลา

```
SELECT PEN ID, HD CNT
FROM LOT LOC
WHERE FDYD ID = 1 AND LOT ID NUM = 219 AND TO DATE =
DATE '9999-12-31'
```

ตาราง 2.7 ผลลัพธ์ของการ Select ทั้งสองแบบข้างต้น[1]

PEN_ID	HD_CNT
2	43

โปรแกรม 2.3 การ Query แบบ Sequence Query

```
SELECT PEN ID, HD CNT, FROM DATE, TO DATE FROM LOT LOC
WHERE FDYD ID = 1 AND LOT ID NUM = 219
```

ตาราง 2.8 ผลลัพธ์จากการทำ Sequence Query[1]

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

[1]Managing Temporal Data. A Five-Part Series

โปรแกรม 2.4 การ Query แบบ Non Sequenced Query

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

ตาราง 2.9 ผลลัพธ์จากการทำ Non Sequence Query[1]

PEN_ID	HD_CNT
1	43
1	20
2	23
2	43

2) Temporal Joins ในบางครั้งตารางเพียง 1 ตารางไม่เพียงพอจะให้ข้อมูลในการตอบคำถามทั้งหมด จึงจำเป็นต้องมีการ Join เกิดขึ้นซึ่งจะแสดงให้เห็นดังตัวอย่างคำถามต่อไปนี้ "มีวัวฝูงใดบ้างที่อยู่คอกเดียวกัน" พิจารณาจากคำถามนี้หากใช้ฐานข้อมูลทั่วไป ไม่มีทางเลยที่จะตอบได้แต่หากเป็นฐานข้อมูลเชิงเวลาจะตอบคำถามดังกล่าวได้ จะต้องคิดพิจารณาถึงเหตุการณ์การอยู่ร่วมกันของวัว 2 ฝูงดังนี้

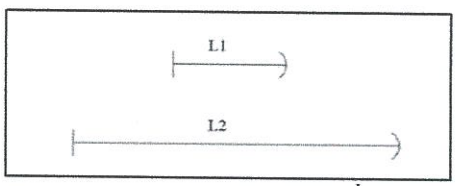
2.1) วัว 2 ฝูงเคยอยู่ร่วมกันจริงในอดีต (Sequence Query)

2.2) วัว 2 ฝูงอยู่ร่วมกันจริงในปัจจุบัน (Current Query)

2.3) วัว 2 ฝูงเคยอยู่คอกเดียวกัน แต่คนละเวลา (Non Sequence Query)

<u>คำตอบที่ 1</u> ฝูงวัวที่เคยอยู่ร่วมกันจริงในอดีต หากพิจารณาเป็นเส้นเวลาจะสามารถเขียนได้ดังนี้

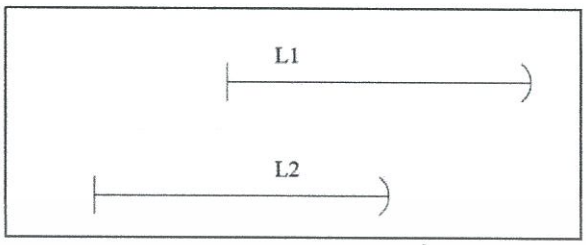
กรณีที่ 1 วัวฝูงที่ 1 มาอยู่คอกเดียวกันในช่วงเวลาที่ฝูงที่ 2 ยังคงอยู่คอกเดิม และออกจากคอกนี้ไปก่อนฝูง 2 จะออกดังนี้



รูป 2.1 ลักษณะของกรณีที่ 1[1]

[1]Managing Temporal Data. A Five-Part Series

กรณีที่ 2 วัวฝูงที่ 1 มาอยู่ร่วมคอกเดียวกับ วัวฝูงที่ 2 โดยที่วัวฝูงที่ 2 นั้นมาอยู่ก่อน วัวฝูงที่ 1 มาอยู่ตามหลังและวัวฝูงที่ 2 ได้ออกไปจากคอกก่อนที่วัวฝูงที่ 1 จะออกตามไปดังแสดงให้เห็นในแผนภาพดังนี้



รูป 2.2 ลักษณะของกรณีที่ 2[1]

กรณีที่ 3 คล้ายกรณีที่ 2 แต่ต่างกันที่วัวฝูงที่ 2 มาอยู่ร่วมคอกเดียวกับ วัวฝูงที่ 1 โดยที่วัวฝูงที่ 1 นั้นมาอยู่ก่อน วัวฝูงที่ 2 มาอยู่ตามหลังและวัวฝูงที่ 1 ได้ออกไปจากคอกก่อนที่วัวฝูงที่ 2 จะออกตามไป

กรณีที่ 4 คล้ายกรณีที่ 1 แต่ต่างกันที่วัวฝูงที่ 2 มาอยู่คอกเดียวกันในช่วงเวลาที่ฝูงที่ 1 ยังคงอยู่คอกเดิม และออกจากคอกนี้ไปก่อนฝูง 1 จะออก เมื่อพิจารณากรณีที่เป็นไปได้ทั้งหมดแล้วต้องเขียนคำตอบสำหรับคำตอบนี้คำตอบเดียวถึง 4 กรณี ดังแสดงให้เห็นด้วยภาษา SQL-92 ดังโปรแกรม 2.5 ซึ่งจะเห็นได้ว่าแค่คำตอบเดียว จำเป็นต้องเขียนภาษา SQL มากมายเพื่อหาคำตอบให้ครอบคลุมถึง 4 กรณีด้วยกัน ซึ่งมันอาจจะไม่ใช่เรื่องยาก แต่มันเป็นเรื่องยุ่งที่จะต้องเขียนโดยไม่ให้มีข้อผิดพลาด

โปรแกรม 2.5 การ Query เพื่อหาคำตอบว่ามีฝูงวัวใดบ้างที่เคยอยู่ร่วมกันจริงในอดีต

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


[1] Managing Temporal Data. A Five-Part Series

โปรแกรม 2.5 การ Query เพื่อหาคำตอบว่ามีฝูงวัวใดบ้างที่เคยอยู่ร่วมกันจริงในอดีต (ต่อ)

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

<u>คำตอบที่ 2</u> วัว 2 ฝูงอยู่ร่วมกันจริงในปัจจุบัน

โปรแกรม 2.6 การ Query เพื่อหาคำตอบว่าวัว 2 ฝูงใดอยู่ร่วมกันจริงในปัจจุบัน

```
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
AND L1.TO DATE = DATE '9999-12-31' AND L2.TO DATE =
DATE '9999-12-31'
```

<u>คำตอบที่ 3</u> วัว 2 ฝูงเคยอยู่คอกเดียวกันแต่คนละเวลา ดังโปรแกรม 2.7 ซึ่งสุดท้ายแล้ว จะต้องนำคำตอบทั้งหมดของแต่ละคำถามมารวมกัน (Union) เพื่อตอบคำถามที่ว่า "วัวตัวใดเคยอยู่คอกเดียวกัน" จากความยุ่งยากดังกล่าวอาจารย์ Richard Snodgrass ผู้เขียนบทความได้พยายามจะผลักดันให้เกิดเป็นคุณสมบัติ (Feature) ใหม่ในภาษา SQL เพื่อลดความยุ่งยากในการพัฒนาโปรแกรมประยุกต์เชิงเวลา

คำถามเดิม "วัวคอกไหนเคยอยู่คอกเดียวกันบ้าง" สำหรับคำตอบที่ว่า วัวต้องเคยอยู่คอกเดียวกันจริงไม่ว่าจะช่วงเวลาใดในอดีต สามารถเขียนได้โดยเพิ่มคำว่า "VALIDTIME" ดังโปรแกรม 2.8 โดย DBMS จะจัดการให้ครอบคลุมทั้ง 4 กรณีที่เกิดขึ้นให้เองโดยที่ผู้พัฒนาโปรแกรมไม่จำเป็นต้องรู้ว่าระบบฐานข้อมูล มีคอลัมน์เวลาอยู่ด้วยซ้ำ และหากต้องการคำตอบที่ว่า วัวเคยอยู่คอกเดียวกันแต่คนละช่วงเวลา ก็ทำได้เพียงแค่ระบุ "NONSEQUENCED" ก่อนหน้า "VALIDTIME" เท่านั้น

โปรแกรม 2.7 การ Query เพื่อหาคำตอบว่าวัว 2 ฝูงใดเคยอยู่คอกเดียวกันแต่คนละเวลา

```
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
```

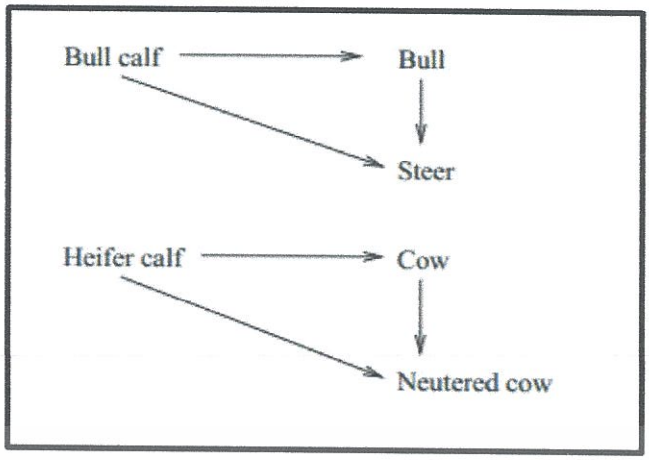
โปรแกรม 2.8 การ Query โดยใช้ VALIDTIME

```
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
```

2.1.3.2 Current Modification

จะเห็นว่าการ Select ข้อมูลเชิงเวลานั้นมีความยุ่งยากมาพอสมควร ในการปรับปรุงข้อมูลเชิงเวลาก็มีความยุ่งยากมากไม่แพ้กัน โดยการแก้ไขข้อมูลในปัจจุบันนั้น มีการแบ่งการแก้ไขออกเป็น

1) General Scenario เป็นการอนุญาตให้แทรกข้อมูล ปรับปรุงข้อมูล ลบข้อมูลในช่วงเวลาไหนก็ได้
2) Restricted Scenario เป็นการอนุญาตให้แก้ไขข้อมูลตารางเฉพาะช่วงเวลาในปัจจุบันของข้อมูลเท่านั้น โดยมีตัวอย่างตารางแสดงสถานะของฝูงวัว ซึ่งมีแผนภาพแสดงสถานะของวัวดังตาราง 2.10 ซึ่งจากตารางจะเห็นได้ว่าฝูง 101 เป็น Calf ตั้งแต่วันที่ 1998-01-01 ถึง 1998-03-23 จากนั้นก็ถูกตอนตั้งแต่วันที่ 1998-03-23 จนถึงปัจจุบัน



รูป 2.3 สถานะของวัวก่อนถูกตอน แยกตามเพศ[1]

ตาราง 2.10 สถานะของฝูงวัว[1]

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

Current Insert ถ้าต้องการ Insert Fact ที่เป็นจริงตั้งแต่ปัจจุบัน สามารถทำได้ ตัวอย่างเช่น

โปรแกรม 2.9 การ Insert ข้อมูลแบบ Current

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

Current Delete ถ้าต้องการ Delete LOT101 ในปัจจุบัน จะไม่สามารถ Delete ปกติได้ ดังโปรแกรม 2.10 ซึ่งจะต้องพิจารณาเวลาที่คาบเกี่ยวกันด้วย เพื่อให้ลบไปเฉพาะเวลาในปัจจุบัน โดยใช้ตัวอย่างของตารางแสดงสถานะของฝูงวัว ดังตาราง 2.11 และจากตารางข้างต้น ฝูง 234 เป็น Calf ตั้งแต่วันที่ 1998-02-17 ถึง 1998-10-17 และวางแผนว่าจะตอนในวันที่ 1998-10-17 โดยสมมติว่าวันนี้เป็นวันที่ 1998-07-29 ถ้าต้องการ delete ฝูง 234 จะต้องแก้ไขให้ข้อมูลสิ้นสุดที่

[1]Managing Temporal Data. A Five-Part Series

วันนี้ และอนาคตต้องไม่มีข้อมูลนั้นอยู่แล้ว ดังโปรแกรม 2.11 โดยคำสั่งดังกล่าวเป็นการแก้ไขฝูง 234 c 1998-02-17 1998-10-17 ซึ่งเป็นแถวที่มีเวลาคาบเกี่ยวกับเวลาปัจจุบัน แก้ไขโดยปรับ to_date เป็น CURRENT DATE และลบ 234 S 1998-10-17 9999-12-31 ที่เป็นเวลาในอนาคตทิ้งไป สุดท้ายจะได้ผลลัพธ์จากการทำ Current Delete ดังตาราง 2.12

โปรแกรม 2.10 การ Delete ข้อมูลแบบ Current

```
DELETE FROM LOT
WHERE LOT ID NUM = 101
```

ตาราง 2.11 สถานะของฝูงวัวใช้สำหรับตัวอย่าง Current Delete[1]

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

โปรแกรม 2.11 การ Delete ตามช่วงเวลาที่ต้องการ

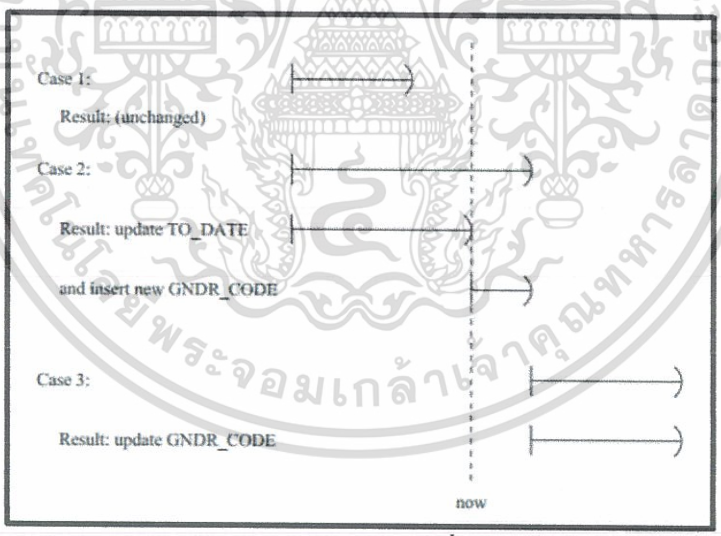
```
UPDATE LOT
SET TO DATE = CURRENT DATE
WHERE LOT ID NUM = 234
AND TO DATE >= CURRENT DATE
AND FROM DATE < CURRENT DATE
DELETE FROM LOT
WHERE LOT ID NUM = 234
AND FROM DATE > CURRENT DATE
```


[1]Managing Temporal Data. A Five-Part Series

ตาราง 2.12 ผลลัพธ์สถานะของฝูงวัว หลังจากการทำ Current Delete[1]

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

Current Update ในการปรับปรุงข้อมูลก็ต้องพิจารณาช่วงเวลาที่คาบเกี่ยวกันด้วยเช่นกัน ไม่สามารปรับปรุงโดยวิธีปกติได้ โดยต้องตรวจสอบช่วงเวลาระหว่างเวลาเดิมก่อนการปรับปรุงกับเวลาที่ต้องการให้เป็นว่ามันคาบเกี่ยวกับแบบไหน ซึ่งสามารถพิจาณาได้ 3 กรณีดังนี้



รูป 2.4 กรณีของการปรับปรุงเวลาของการเปลี่ยนแปลงในช่วงเวลาปัจจุบัน[1]

กรณีที่ 1 ปรับปรุงข้อมูลในอดีตในเวลาปัจจุบัน โดยแก้ไข to_date เป็นเวลาในปัจจุบัน

กรณีที่ 2 ปรับปรุงข้อมูลที่ยังเป็นจริงอยู่ในเวลาปัจจุบัน โดยการปรับปรุง to_date ของแถวเดิมเป็นเวลาปัจจุบัน และแทรกแถวใหม่ที่มี from_date เป็นเวลาปัจจุบัน

[1] Managing Temporal Data. A Five-Part Series

กรณีที่ 3 ปรับปรุงข้อมูลในอนาคตในเวลาปัจจุบัน

2.1.3.3 Sequenced Modification

1) Sequence Insert ถ้าจะ Insert Sequence ต้องบอกด้วยว่า Fact นั้น Valid ตั้งแต่เมื่อไหร่ถึงเมื่อไหร่ ตัวอย่างเช่น

โปรแกรม 2.12 การ Query โดยใช้ VALIDTIME

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

2) Sequence Delete จากตัวอย่างได้มีการให้ฝูงวัว 234 ย้ายออกจากฟาร์ม 3 สัปดาห์แรกของเดือนตุลาคม ซึ่งเป็นช่วงเวลาที่มีการวางแผนจะเปลี่ยนฝูงวัว 234 ให้เป็นวัวตอน โดยช่วงเวลาที่ต้องการเปลี่ยนแปลงข้อมูล คือวันที่ '1998-10-01' ถึงวันที่ '1998-10-22' โดยการเปลี่ยนแปลงลักษณะนี้ทำให้ต้องพิจารณาการแปลี่ยนแปลงข้อมูล ซึ่งจะเกิดกรณีของการเปลี่ยนแปลงข้อมูลได้ 4 กรณี ดังรูป 2.5

2.1) PV คือ ช่วงเวลาของข้อมูลเดิม กล่าวคือเป็นช่วงเวลาที่ Fact เป็นจริงอยู่ในฐานข้อมูล

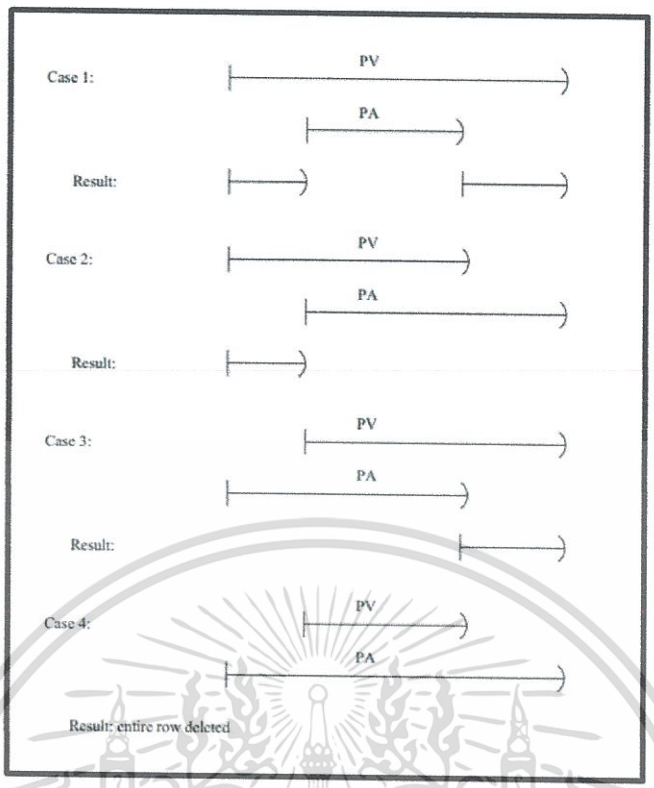
2.2) PA คือ ช่วงเวลาที่ต้องการเปลี่ยนแปลงข้อมูล กล่าวคือเป็นช่วงเวลาที่ถูกปฏิบัติ

กรณีที่ 1 PA เกิดทีหลังและสิ้นสุดก่อน PV ผลของการลบข้อมูลจะเกิดการปรับปรุง 1 แถว โดย from_date เท่ากับ from_date ของ PV และเปลี่ยน to_date เป็น from_date ของ PA รวมทั้งแทรกอีก 1 แถว โดยให้ from_date เท่ากับ to_date ของ PA และ from_date เท่ากับ from_date ของ PV

กรณีที่ 2 PA เกิดขึ้นหลัง PV และสิ้นสุดหลัง PV ผลของการลบข้อมูลจะเป็นการปรับปรุง 1 แถว โดย from_date เท่ากับ from_date ของ PV และ to_date เท่ากับ from_date ของ PA

กรณีที่ 3 PA เกิดก่อน PV และสิ้นสุดก่อน PV ผลของการลบข้อมูลจะเป็นการปรับปรุง 1 แถว โดย from_date เท่ากับ to_date ของ PA และ to_date เท่ากับ to_date ของ PV

กรณีที่ 4 PA เกิดขึ้นก่อนและสิ้นสุดหลัง PV ผลของการลบข้อมูลจะเป็นการลบแถวข้อมูลของ PA ออกจากฐานข้อมูล



รูป 2.5 กรณีของลบช่วงเวลาแบบ Sequence (Sequence Delete) ทั้ง 4 กรณี[1]

จากตัวอย่าง ถ้าต้องการลบฝูง 234 ในช่วงเวลาตั้งแต่วันที่ 01-10-1998ถึง 10-1998- 22จะต้องพิจารณาให้ครบ 4 กรณี โดยใช้คำสั่งเอสคิวแอล ดังโปรแกรม 2.13 จะเห็นว่าการทำ Sequence Delete จะต้องเขียนคำสั่งเอสคิวแอลมาตรฐานถึง 4 คำสั่ง เพื่อให้เกิดการลบที่ถูกต้อง โดยในแต่ละคำสั่งผู้พัฒนาโปรแกรมจะต้องตรวจสอบเงื่อนไขของช่วงเวลาให้ถูกต้องทั้ง from_date และ to_date ซึ่งเป็นเรื่องที่ซับซ้อนและยุ่งยากเป็นอย่างมาก ถ้าเขียนผิดก็อาจส่งผลให้เกิดการลบที่ไม่ถูกต้องได้ ซึ่งก็ยังมีคำสั่งเอสคิวแอลที่เขียนง่ายกว่า โดยระบุช่วงเวลาที่ต้องการลบ จากนั้น DBMS จะตรวจสอบเงื่อนไขทั้ง 4 ให้โดยอัตโนมัติ คำสั่งจะเป็นดังโปรแกรม 2.17

โปรแกรม 2.13 กรณีที่ 1

```
INSERT INTO LOT
SELECT LOT ID NUM, GNDR CODE, DATE '1998-10-22',TO DATE
FROM LOT
WHERE LOT ID NUM = 234
AND FROM DATE <= DATE '1998-10-01'
AND TO DATE > DATE '1998-10-22'
```


[1] Managing Temporal Data. A Five-Part Series

โปรแกรม 2.14 กรณีที่ 2

```
UPDATE LOT
SET TO DATE = DATE '1998-10-01'
WHERE LOT ID NUM = 234
AND FROM DATE < DATE '1998-10-01'
AND TO DATE >= DATE '1998-10-01'
```

โปรแกรม 2.15 กรณีที่ 3

```
UPDATE LOT
SET FROM DATE = DATE '1998-10-22'
WHERE LOT ID NUM = 234
AND FROM DATE < DATE '1998-10-22'
AND TO DATE >= DATE '1998-10-22'
```

โปรแกรม 2.16 กรณีที่ 4

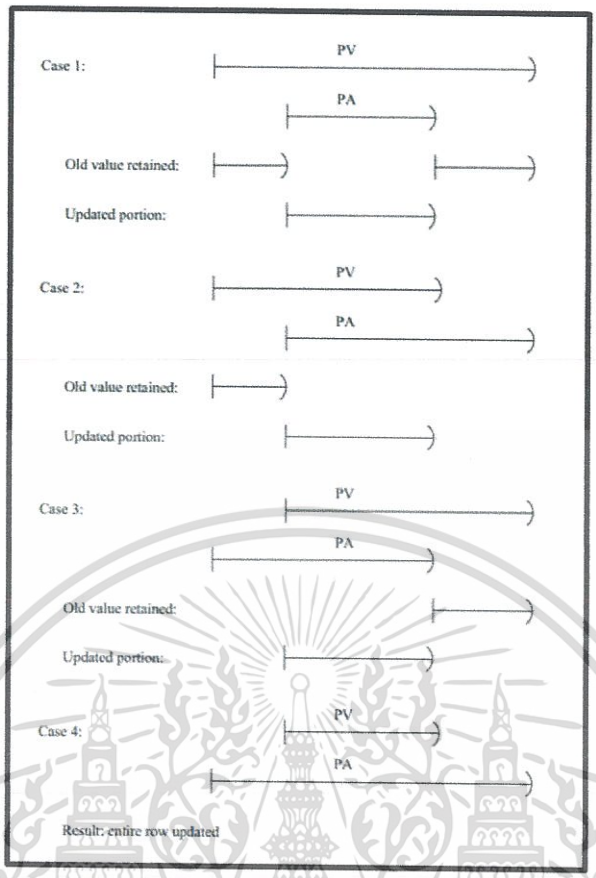
```
DELETE FROM LOT
WHERE LOT ID NUM = 234
AND FROM DATE >= DATE '1998-10-01'
AND TO DATE <= DATE '1998-10-22'
```

โปรแกรม 2.17 การ Delete แบบ Sequence โดยการระบุช่วงเวลา

```
VALIDTIME PERIOD '[1998-10-01 - 1998-10-22)'
DELETE FROM LOT
WHERE LOT ID NUM = 234
```

3) Sequence Update

ตัวอย่าง ถ้าต้องการเปลี่ยนวัว LOT 799 เป็นวัวตอนตัวผู้ เฉพาะในเดือนมีนาคม การเปลี่ยนแปลงนี้จะต้องพิจารณา 4 กรณี ดังนี้



รูป 2.6 กรณีของปรับปรุงช่วงเวลาแบบ Sequence (Sequence Update) ทั้ง 4 กรณี[1]

กรณีที่ 1 PA เกิดขึ้นหลังและสิ้นสุดก่อน PVผลของการปรับปรุงจะเป็นการเพิ่มแถว 2 แถวและ ปรับปรุง 1 แถว ดังนี้ เพิ่มแถวแรกโดย from_date มีค่าเท่ากับ from_date ของ PV และ to_date มีค่าเท่ากับ from_date ของ PA เพิ่มแถวที่สองโดย from_date มีค่าเท่ากับ to_date ของ PA และ from_date มีค่าเท่ากับ from_date ของ PV ปรับปรุงแถวข้อมูลหนึ่งแถวโดยปรับปรุง from_date ของ PV ให้เป็น form_date ของ PA และ to_date ของPV ให้มีค่าเป็น to_date ของ PA และค่าของข้อมูลจะเท่ากับค่าของ PA

กรณีที่ 2 PA เกิดขึ้นและสิ้นสุดหลัง PV ผลของการปรับปรุงจะเป็นการเพิ่มแถว 1 แถว และปรับปรุง 1 แถว เพิ่มแถวโดย from_date มีค่าเท่ากับ from_date ของ PV และ to_date มีค่าเท่ากับ from_date ของ PA ปรับปรุงแถวโดยปรับปรุง from_date ของ PV ให้เป็น form_date ของ PA และ to_date ของ PV มีค่าเท่ากับ to_date ของ PA และค่าของข้อมูลจะเท่ากับค่าของ PA

กรณีที่ 3 PA เกิดขึ้นและสิ้นสุดก่อน PV ผลของการปรับปรุงจะเป็นการเพิ่ม 1 แถวและปรับปรุง 1 แถว เพิ่มแถวโดย from_date มีค่าเท่ากับ to_date ของ PA

[1] Managing Temporal Data. A Five-Part Series

และ to_date มีค่าเท่ากับ to_data ของ PV ปรับปรุงแถวโดยปรับปรุง from_date ของ PV ให้เป็น form_date ของ PA และ to_date ของ PV มีค่าเท่ากับ to_date ของ PA และค่าของข้อมูลจะเท่ากับค่าของ PA

กรณีที่ 4 PA เกิดขึ้นก่อนและสิ้นสุดหลัง ผลของการปรับปรุงจะเป็นการปรับปรุง 1 แถว คือ ปรับปรุงเฉพาะค่า PA โดย from_date และ to_date ยังมีค่าเท่าเดิม ซึ่งสามารถใช้คำสั่งเอสคิวแอลในการปรับปรุงข้อมูลแบบ Sequence ได้ดังนี้

โปรแกรม 2.18 การ Update แบบ Sequence

```
INSERT INTO LOT
SELECT LOT ID NUM, GNDR CODE, FROM DATE, DATE '1998-03-
01'
FROM LOT
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-03-01'
AND TO DATE > DATE '1998-03-01'
INSERT INTO LOT
SELECT LOT ID NUM, GNDR CODE, DATE '1998-04-01', TO
DATE
FROM LOT
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-04-01'
AND TO DATE > DATE '1998-04-01'
INSERT INTO LOT
SELECT LOT ID NUM, GNDR CODE, DATE '1998-04-01', TO
DATE
FROM LOT
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-04-01'
AND TO DATE > DATE '1998-04-01'
UPDATE LOT
SET GNDR CODE = 's'
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-04-01'
AND TO DATE > DATE '1998-03-01'
UPDATE LOT
SET FROM DATE = DATE '1998-03-01'
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-03-01'
AND TO DATE > DATE '1998-03-01'
UPDATE LOT
SET TO DATE = DATE '1998-04-01'
WHERE LOT ID NUM = 799
AND FROM DATE < DATE '1998-04-01'
AND TO DATE > DATE '1998-04-01'
```

จะเห็นได้ว่าการทำ Sequence Update ต้องเขียนคำสั่งเอสคิวแอลมาตรฐานถึง 5 คำสั่ง เพื่อให้เกิดการปรับปรุงข้อมูลที่ถูกต้อง โดยในแต่ละคำสั่งผู้พัฒนาโปรแกรมจะต้องตรวจสอบเงื่อนไขของช่วงเวลาให้ถูกต้องทั้ง from_date และ to_date ซึ่งเป็นเรื่องที่ซับซ้อนและยุ่งยากเป็นอย่างมาก ถ้าเขียนผิดก็อาจส่งผลให้เกิดการปรับปรุงที่ไม่ถูกต้องได้ ซึ่งก็ยังมีคำสั่งเอสคิวแอลที่เขียนง่ายกว่า โดยระบุช่วงเวลาที่ต้องการปรับปรุง จากนั้น DBMS จะตรวจสอบเงื่อนไขทั้ง 4 ให้โดยอัตโนมัติ คำสั่งจะเป็นดังนี้

โปรแกรม 2.19 การ Update แบบ Sequence แบบระบุช่วงเวลา

```
VALIDTIME PERIOD '[1998-03-01 - 1998-04-01)'
UPDATE LOT
SET GNDR CODE = 's'
WHERE LOT ID NUM = 799
```

2.2 เวิคสเปสเมเนเจอร์ (Workspace Manager)

ถ้ากล่าวถึง OWM มักหมายถึงเวิคสเปสเมเนเจอร์ (Workspace Manager) ซึ่งเป็นเครื่องมือที่ช่วยให้แอพพลิเคชั่นต่างๆ สามารถสร้างพื้นที่การทำงาน (Workspace) และเวอร์ชั่น (Version) ที่แตกต่างกันของค่าในตารางได้ในหลายเวิคสเปส ผู้ใช้งานสามารถสร้างเวอร์ชั่นใหม่ของข้อมูลเพื่อทำการปรับปรุงได้ ในขณะเดียวกันข้อมูลเก่าก็ถูกเก็บรักษาไว้ ซึ่งค่าผลลัพธ์ต่างๆ จะถูกเก็บไว้อย่างถาวร เพื่อทำให้มั่นใจได้ว่าค่าเหล่านั้นมีความสอดคล้องกันในฐานข้อมูล โดยทั่วไปแล้วแอพพลิเคชั่นที่ใช้ OWM ในการทำงาน มักเป็นแอพพลิเคชั่นที่ต้องการทำงานอย่างใดอย่างหนึ่งดังต่อไปนี้

1) ต้องการจัดการกับชุดข้อมูลการปรับปรุงและข้อมูลที่ถูกนำเข้ามาใหม่ ก่อนที่ข้อมูลเหล่านั้นจะผ่านกระบวนการต่างๆ ออกเป็นข้อมูลที่ใช้งานจริง (Production Data) ผู้ใช้งานสามารถเรียกดูข้อมูลที่มีการเปลี่ยนแปลง และย้อนการกระทำก่อนหน้า (Roll Rack) ได้ก่อนจะมีการทำเชจพับบลิก (Change Public) ซึ่งก่อนที่จะมีการทำเชจพับบลิกนั้น ผู้ใช้งานคนอื่นจะไม่สามารถมองเห็นข้อมูลในฐานข้อมูลได้ จะเข้าถึงข้อมูลได้แค่ข้อมูลที่ใช้งานจริง ซึ่งผู้ใช้งานสามารถสร้าง Workspace แบบธรรมดา หรือแบบซับซ้อนได้ ในรูปแบบของลำดับชั้น (Workspace Hierarchy) ตัวอย่างเช่น แอพพลิเคชั่นด้านงานวิทยาศาสตร์ ที่ OWM ช่วยในการค้นพบและการประกันคุณภาพ (QA) ของกระบวนการโดยการจัดการกับชุดข้อมูลการปรับปรุงก่อนจะกลายเป็นข้อมูลที่ใช้งานจริง
2) ต้องการการพัฒนาร่วมกัน ทีมนักพัฒนาสามารถแบ่งปันหรือเข้าถึงข้อมูลการปรับปรุงและข้อมูลที่ถูกนำเข้ามาใหม่ได้ Workspace สามารถควบคุมการเข้าถึงและการดำเนินงานกับมันได้ และยังสามารถจำกัดสิทธิ์การเข้าถึง Workspace แบบเขียนได้คนเดียว (Single-Writer) อ่านได้อย่างเดียว (Read-Only) หรือไม่อนุญาตให้เข้าถึงเลย ซึ่ง OWM จะป้องกันการปรับปรุงข้อมูลระหว่างโปรเจคที่อยู่บน Workspace ต่างกันไม่ให้ขัดแย้งกัน ตัวอย่างเช่น แอพพลิเคชั่นที่เป็นโปรเจคทางวิศวกรรม จะมีโปรเจคย่อยๆ ที่ต้องพัฒนาควบคู่กันไปในต่าง Workspace มีการใช้ชุดข้อมูลร่วมกันสำหรับการสร้างสถานการณ์ที่หลากหลาย เพื่อสนับสนุนการวิเคราะห์แบบวอทอีฟ (What-If) หรือชุดข้อมูลเหล่านั้นมีการแก้ไขบ่อยครั้ง
3) ผู้ใช้งานสามารถจัดการการเปลี่ยนแปลงข้อมูลใน Workspace เพื่อดูข้อมูลเหล่านั้นส่วนในของฐานข้อมูลทั้งหมดได้ ซึ่งผู้ใช้งานแต่ละคนสามารถเปลี่ยนแปลงข้อมูลเดียวกัน (ใน Row เดียวกัน) พร้อมกันได้ ผู้ใช้สามารถตรวจสอบและแก้ปัญหาความขัดแย้งกันของข้อมูลได้ ตัวอย่างเช่น แอพพลิเคชั่นเกี่ยวกับการสื่อสารโทรคมนาคมที่ OWM สนับสนุนการสร้างข้อมูลของโทรศัพท์มือถือหลายๆ เครื่อง ให้ครอบคลุมสถานการณ์ต่างๆ เพื่อหาการออกแบบที่ดีที่สุด

4) ต้องการเก็บประวัติของข้อมูลที่มีการเปลี่ยนแปลง ผู้ใช้สามารถกลับไปหาข้อมูลในเวิคสเปสในเวอร์ชั่นเก่าได้ตามไมล์สโตน (Milestone) ที่มีอยู่ โดยจะย้อนการกระทำของข้อมูลในตารางใน Workspace อันปัจจุบันกลับไปเป็นตาม Workspace ที่อยู่ ณ ไมล์สโตนนั้นๆ ตัวอย่างเช่น แอพพลิเคชั่นเกี่ยวกับการจัดการข้อมูลที่ดิน ที่ OWM สนับสนุน Regulatory Requirements โดยจะเก็บประวัติของการเปลี่ยนแปลงที่ดินทั้งหมดไว้

OWM เป็นผลิตภัณฑ์หนึ่งของออราเคิลดาต้าเบส (Oracle Database) ที่อนุญาตให้นักพัฒนาแอพพลิเคชั่น และผู้ดูแลระบบฐานข้อมูล (Database Administrator) สามารถจัดการกับข้อมูลที่มีหลายเวอร์ชั่นของฐานข้อมูลเดิมได้ โดย Workspace เป็นเสมือนสิ่งแวดล้อมจำลองที่แยกชุดข้อมูลที่มีการเปลี่ยนแปลง เก็บประวัติของการเปลี่ยนแปลงข้อมูล และยังสามารถสร้างสถานการณ์ของข้อมูลที่หลากหลายได้ เพื่อใช้ในการวิเคราะห์ข้อมูล ซึ่งสามารถประหยัดค่าใช้จ่าย เวลา และลดความยากลำบากในการใช้คำสั่งในแบบเดิม

OWM สนับสนุนภาษา PL/SQL ซึ่งนักพัฒนาสามารถเพิ่มแอพพลิเคชั่นใหม่หรือแอพพลิเคชั่นที่มีอยู่แล้วให้เป็นตารางเวอร์ชั่นอินาเบิลได้ (Version-enable Tables) ได้ สามารถทำงานใน Workspace ได้ โดยใช้เวิคสเปสเซฟพอยท์ (Workspace Savepoint) ประวัติข้อมูล (History) ได้ เป็นต้น

2.2.1 ตารางเวอร์ชั่นอินาเบิลได้ (Version-enable Tables)

OWM อนุญาตให้มีการใช้เวอร์ชั่นอินาเบิลได้มากกว่าหนึ่งตารางในฐานข้อมูล ซึ่งหน่วยของมันคือแถว (Row) เมื่อผู้ใช้ตั้งค่าตารางให้เป็น Version-enable แล้ว ทุกแถวในตารางจะสนับสนุนการทำงานแบบมัลติเวอร์ชั่น (Multiple Versions) กับข้อมูลได้ แถวข้อมูลที่เป็นเวอร์ชั่นใหม่ (Versioned Row) จะถูกเก็บอยู่ในตารางเดิม และทุกๆ การเพิ่มข้อมูล การปรับปรุงข้อมูล และการลบข้อมูลของแถวข้อมูลที่เป็นเวอร์ชั่นใหม่ จะถูกดำเนินการปกติตามมาตรฐานของ Oracle กล่าวคือทำให้มั่นใจว่าข้อมูลมีความถูกต้องตามกฎ (Integrity) ของข้อมูลเวอร์ชั่น โดยผู้ใช้งานจะไม่เห็นโครงสร้างของมัน

เมื่อเปิดใช้ Version-enable แล้ว OWM จะเปลี่ยนชื่อตารางเดิมเป็น *table name*_LT และเพิ่มบางคอลัมน์ที่ตาราง เพื่อเก็บข้อมูลที่เป็นเวอร์ชั่นและสร้างวิว (View) โดยยังคงใช้ชื่อเดิมและกำหนด INSTEAD OF Triggers บนวิวเพื่อให้สามารถใช้งานกับภาษา SQL และตัวดำเนินการ (Operation) พื้นฐานได้ และถ้าผู้ใช้ไม่ต้องการใช้ Version-enable อีกต่อไป ก็สามารถสั่ง Disable Versioning กับตารางได้

2.2.2 เวิคสเปส (Workspace)

Workspace เป็นสภาวะจำลอง ไม่ใช่แหล่งเก็บข้อมูลจริงๆ สามารถดำเนินการทำงานพร้อมกัน มีการจับเก็บและแยกชุดของข้อมูลที่มีการเปลี่ยนแปลง (New row versions) ซึ่งเป็นการ

เก็บข้อมูลแบบ Long Transaction กล่าวคือเก็บอย่างไม่มีระยะเวลากำหนด ผู้ใช้งานจะเห็นเวอร์ชั่นของแถวที่ถูกต้องอยู่แล้วโดยอัตโนมัติ

ค่าเริ่มต้นของเวิคสเปสเรียกว่า LIVE มันคือที่ที่ผู้ใช้งานเห็นเวอร์ชั่นปัจจุบัน (Current Version) ของข้อมูล เราสามารถสร้าง Workspace เป็นลำดับชั้นได้ โดยที่ LIVE จะเป็น Workspace ที่อยู่ข้างบนสุดเสมอ โดยพื้นฐานแล้ว เมื่อมีการสร้างเวิคสเปสเพิ่ม มันจะเป็นลูกของ LIVE โดยอัตโนมัติ ซึ่งถ้าผู้ใช้ต้องการจะเปลี่ยนการตั้งค่านี้สามารถดำเนินการได้โดยใช้คำสั่งใน GotoWorkspace และ Workspace ยังสามารถใช้งานได้มากกว่าหนึ่งผู้ใช้งาน

เวอร์ชั่นใหม่ของของแถวจะถูกสร้างขึ้นเมื่อแถวใน Workspace ที่เป็นลูก (Child Workspace) มีการเปลี่ยนแปลงเกิดขึ้นครั้งแรก การเปลี่ยนแปลงต่างๆ ในภายหลังที่เกิดขึ้นจะถูกเก็บอยู่ในเวอร์ชั่นใหม่นี้จนกว่าจะทำการ Savepoint ซึ่งแถวของเวอร์ชั่นใหม่ที่เกิดขึ้นนี้ก็ยังคงอยู่ในตารางเดิมกับที่แถวของเวอร์ชั่นเก่าอยู่ การเปลี่ยนแปลงใน Workspace ไม่สามารถถูกมองเห็นจากภายนอกได้ จนกว่าจะทำการรวมแบบ Explicitly กับข้อมูลที่ใช้งานจริงก่อน ตัวอย่าง Operation ที่ใช้ใน Workspace เช่น CREATE, GOTO, REFRESH, MERGE, ROLLBACK, COMPRESS, REMOVE, MULTI-PARENT และ ALTER DESCRIPTION เป็นต้น

2.2.3 เซฟพอยท์ (Savepoint)

Savepoint เป็นกลไกในการสร้างเวอร์ชั่นใหม่ เป้าหมายของการทำ Savepoint ก็เพื่อให้ย้อนการกระทำกับข้อมูลกลับมาได้ และยังทำให้ผู้ใช้งานสามารถกลับมาดูสถานะของคลังข้อมูลแต่ละช่วงตามไมล์สโตนที่มีได้

Savepoint สามารถทำได้โดยสองวิธีการ หนึ่งคือแบบ Implicit เป็นการ Savepoint บน Parent Workspace โดยอัตโนมัติ เกิดขึ้นเมื่อมีการสร้าง Child Workspace วิธีที่สองคือแบบ Explicit เป็นการ Savepoint โดยผู้ใช้งาน ซึ่ง Savepoint เป็นไมล์สโตนตามช่วงเวลาหรือเหตุการณ์ต่างๆ ที่ต้องการ

2.2.4 ประวัติการเปลี่ยนแปลง (History of Changes)

เมื่อตารางเปิดใช้งาน Version-enable มันจะมีตัวดำเนินการกับตัวเก็บประวัติ (Option History) ให้เลือกใช้งาน ถ้าเปิดใช้งาน Option นี้ OWM จะเพิ่ม Transaction Time Timestamp ในทุกๆ ครั้งที่แถวของข้อมูลในตารางเกิดการเปลี่ยนแปลง ซึ่ง Option นี้ อนุญาตให้ประวัติการเปลี่ยนแปลงสามารถสร้างเวอร์ชั่นได้ใหม่ โดยการ Savepoint ทำให้ผู้ใช้งานกลับไปยังจุดต่างๆ ของเวลาได้

2.2.5 แวลิดไทม์ (Valid Time)

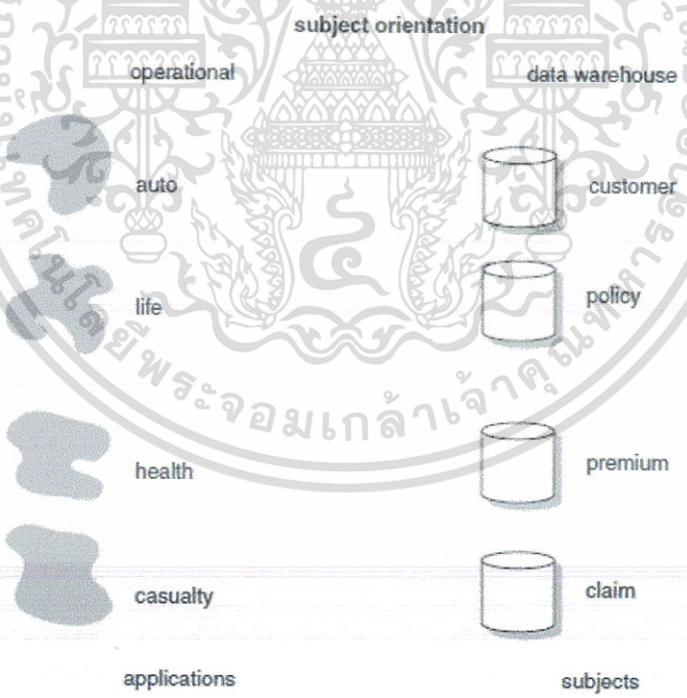
บางแอพพลิเคชั่นต้องการการจัดเก็บข้อมูลที่เกี่ยวข้องกับเวลา เพื่อใช้ระบุว่าข้อมูลนั้นจะเป็นจริง ณ เวลาหนึ่ง ถ้าตารางมีการเปิดใช้งาน Valid Time ตารางนั้นจะมีคอลัมน์เพิ่มขึ้นมาเพื่อเก็บค่า Valid Timeโดยสามารถเก็บเวลาที่เป็นอดีต ปัจจุบัน หรืออนาคตได้ ซึ่งถ้าผู้ใช้งานต้องการ

สอบถาม (Query) ข้อมูลในตาราง ผู้ใช้งานต้องระบุ Valid Time ของข้อมูลที่ต้องการสอบถามด้วย โดยใช้ SetValidTime

2.3 คลังข้อมูล (Data Warehouse)

คลังข้อมูล คือกระบวนการนำข้อมูลที่มีการเก็บรวมรวมเป็นช่วงที่สม่ำเสมอจากหลายๆ แหล่งข้อมูล มารวบรวมและจัดเก็บในรูปแบบของ Dimensional Model หรือรูปแบบ Relational Model ก็ได้ เพื่อที่จะนำข้อมูลเหล่านั้นมาวิเคราะห์ในเชิง Business Intelligence หรือทำการวิเคราะห์ข้อมูลในด้านอื่นๆ ซึ่งประโยชน์หลักที่ผู้ใช้งานต้องการจากคลังข้อมูลนั้นก็คือการวิเคราะห์ความเปลี่ยนแปลงที่เชื่อมโยงกันของข้อมูลเพื่อนำประโยชน์ส่วนนี้ไปวางแผนข้อมูลทางธุรกิจสำหรับผู้บริหาร โดยส่วนใหญ่การนำเข้าข้อมูล จาก Operational Database จะทำเป็นชุดข้อมูลโดยมีคาบเวลา (Period) ที่แน่นอน โดยจะไม่ทำทุกๆ ครั้งที่เกิดการเปลี่ยนแปลงที่เป็นผลมาจาก Transaction ที่แหล่งข้อมูล

2.3.1 Data Warehouse Environment



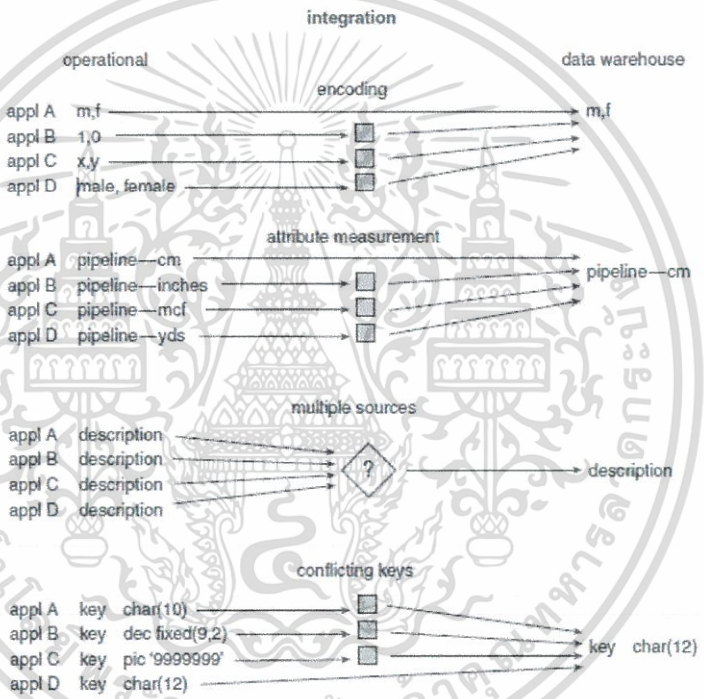
รูป 2.7 ตัวอย่าง Application และ Subjects ใน Data Warehouse[2]

จากรูปจะแสดงตัวอย่างของ Operational และคลังข้อมูล โดยที่ Operational นั้นจะตามแอพพลิเคชั่น ซึ่งจากตัวอย่างจะเห็นได้ว่ามี แอพพลิเคชั่นที่เกี่ยวกับประกันสุขภาพ ประกันอุบัติเหตุ เป็นต้น ส่วนทางขวานั่นจะเป็นเรื่องของคลังข้อมูล ซึ่งจะแสดงตาม Subject (หัวข้อ) ที่สนใจ

[2] W.H Inmon Buiding the Data Warehouse Third Edition

เนื่องจากในการวิเคราะห์ข้อมูลนั้นส่วนใหญ่มักจะไม่ถามตามแอพพลิเคชั่น แต่จะถามตาม Subject มากกว่า ตัวอย่าง เช่น ลูกค้า (รวมลูกค้าทุกประกัน) Claim กรมธรรม์ต่างๆ เป็นต้น ซึ่งจะเห็นได้ว่าการออกแบบข้อมูลเพื่อใช้งานในแอพพลิเคชั่นกับการแยกประเภทข้อมูลตาม Subject นั้นมีความแตกต่างกันจะเห็นได้ว่าตาราง Customer ของคลังข้อมูลจะต้องถูกโหลดมาจากทุกตารางที่มีข้อมูลเกี่ยวกับลูกค้าในแอพพลิเคชั่น ดังนั้นข้อมูลที่จะโหลดมารวมกันจะต้องมีรูปแบบที่เหมือนกันถึงจะรวมกันได้

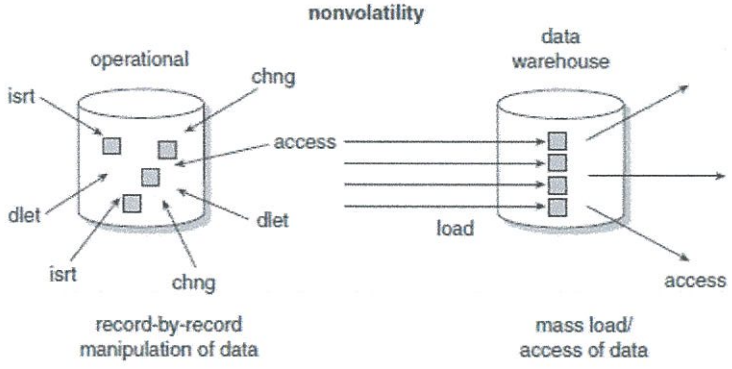
ปัญหาที่เกิดขึ้นคือแต่ละแอพพลิเคชั่นอาจจะถูกพัฒนาโดยผู้พัฒนาโปรแกรมหลายๆคน ซึ่งเป็นไปได้ยากที่ข้อมูลจากแอพพลิเคชั่นนั้นจะมีรูปแบบเหมือนกันทุกประการ ดังตัวอย่าง



รูป 2.8 การรวมข้อมูลจากแต่แอพพลิเคชั่นไปเป็น Subject ในคลังข้อมูล[2]

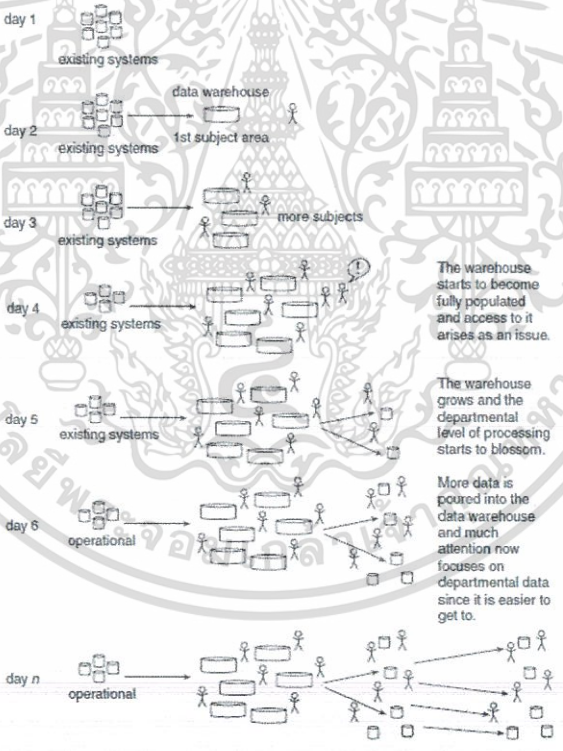
จะเห็นได้ว่าจาก Operational แต่ละแอพพลิเคชั่นนั้นมีรูปแบบ ของ namming, value, data type ของ Attribute ที่ไม่เหมือนกัน ดังนั้นเมื่อนำข้อมูลจาก Attribute เหล่านั้นไปรวมกันในคลังข้อมูลจะต้องตกลงกันไว้ก่อนว่าเมื่อไปเป็นข้อมูลในคลังข้อมูลแล้วจะให้เป็น Attribute ตัวกลางอย่างไร ซึ่งจะต้องชัดเจนในเป็นอย่างมากในขั้นตอนนี้ ดังนั้นขั้นตอนแรกในการออกแบบคลังข้อมูลนั้นจะต้องกำหนดโค้ด (Code) ส่วนกลาง และปลายทางให้เรียบร้อยเสียก่อน ก่อนที่จะจัดซื้อ Hardware หรือพัฒนา Software ในส่วนอื่น

[2] W.H Inmon Buiding the Data Warehouse Third Edition



รูป 2.9 ความแตกต่างระหว่างฐานข้อมูลและคลังข้อมูล[2]

ความแตกต่างระหว่างฐานข้อมูล และคลังข้อมูลก็คือ ฐานข้อมูลนั้นจะต้องมีการ Insert, Update, Delete แต่คลังข้อมูลนั้นไม่จำเป็นต้องมี มีแค่เพียงการนำเข้าและเข้าถึงข้อมูลก็เพียงพอแล้ว



รูป 2.10 Day-1 to Day-n Phenomenon[2]

จากรูปเป็นการแสดง Day-1 to Day-n Phenomenon นั่นคือคลังข้อมูลที่ว่าจะเริ่มมาจาก Day-1 คือมี Existing System โดยไม่จำเป็นต้องมี 1 Operational Database อาจจะมีหลาย Operational Database ก็เป็นได้ แล้วนำเอา Operation เหล่านั้นมาสร้างเป็น 1st Subject Area ใน Day-2 จากนั้นที่ Day-3 เมื่อมีคนเริ่มใช้คลังข้อมูลก็จะเกิด More Subject หรือ Subject อื่นๆ ที่คน

[2] W H Inmon Buiding the Data Warehouse Third Edition

สนใจขึ้นมา ทำให้คนใช้คลังข้อมูลเริ่มเกิดการสับสนว่าควรไปหยิบข้อมูลมาจาก Subject ไหนกันแน่ที่ตัวเองต้องการ จึงต้องมีการนำเข้าข้อมูลจากคลังข้อมูลไปเป็น Mart ซึ่งจะมีขนาดเล็กกว่าคลังข้อมูล แต่บางครั้ง Mart เพียงชั้นเดียวอาจจะยังใหญ่ไปสำหรับหน่วยงานที่มาใช้งานคลังข้อมูล ทำให้ต้องสร้าง Mart ขึ้นมาหลายๆ ชั้น จนกระทั่งเกิดเป็น Day-N ดังรูป ถึงเวลาผู้ใช้งานก็จะไปหยิบ Mart Level ก่อนหน้าตนมาวิเคราะห์ จะเห็นได้ว่าแผนภาพนี้แสดงให้เห็นอย่างชัดเจนว่าคลังข้อมูลนั้นไม่ได้ Online กันแบบ Realtime

แนวคิดในปัจจุบันคลังข้อมูลนั้นยังคงเป็นการนำเข้าข้อมูลจากหลายๆ แหล่งข้อมูล มารวมกันอยู่ แต่ไม่แน่ในอนาคตข้างหน้าทุกๆ Operation ทั้งหมดจะอยู่บนเครื่องเดียวกัน และ Mart หรือคลังข้อมูล จะเป็น View ที่เรียก Operation เหล่านั้นมาแสดง เนื่องจากแนวคิดเก่านั้น Operation คือ ฐานข้อมูล Current Version แล้วเมื่อเวลาผ่านไปถึง Life Span ก็จะนำเข้าฐานข้อมูล Current Version นั้นเก็บไว้ เมื่อต้องการจะทำคลังข้อมูล ก็เพียงแต่โหลดเวอร์ชั่นเก่าเหล่านั้นตามที่ต้องการมารวมไว้ในที่เดียวกันแล้วทำการวิเคราะห์ แต่หากเราวิเคราะห์ขั้นตอนการทำคลังข้อมูลทั้งหมด เราจะพบว่า หากฐานข้อมูลที่ใช้เป็นฐานข้อมูลเชิงเวลาแล้ว Operation นั้นเป็น Current Version หากเราต้องการจะทำคลังข้อมูลก็เพียงแค่ค้นหา ย้อนหลังตราบเท่าที่เราต้องการแล้วใช้ Vertual Machine ดูเฉพาะที่เราต้องการจะดู ก็จะสามารถทำคลังข้อมูลได้แล้ว อีกทั้งยังไม่ต้องทำการแปลงชนิดของข้อมูลอีกด้วย เนื่องจากทั้งหมดนั้นอยู่บนฐานข้อมูลเดียวกัน

2.3.2 คุณลักษณะของคลังข้อมูล

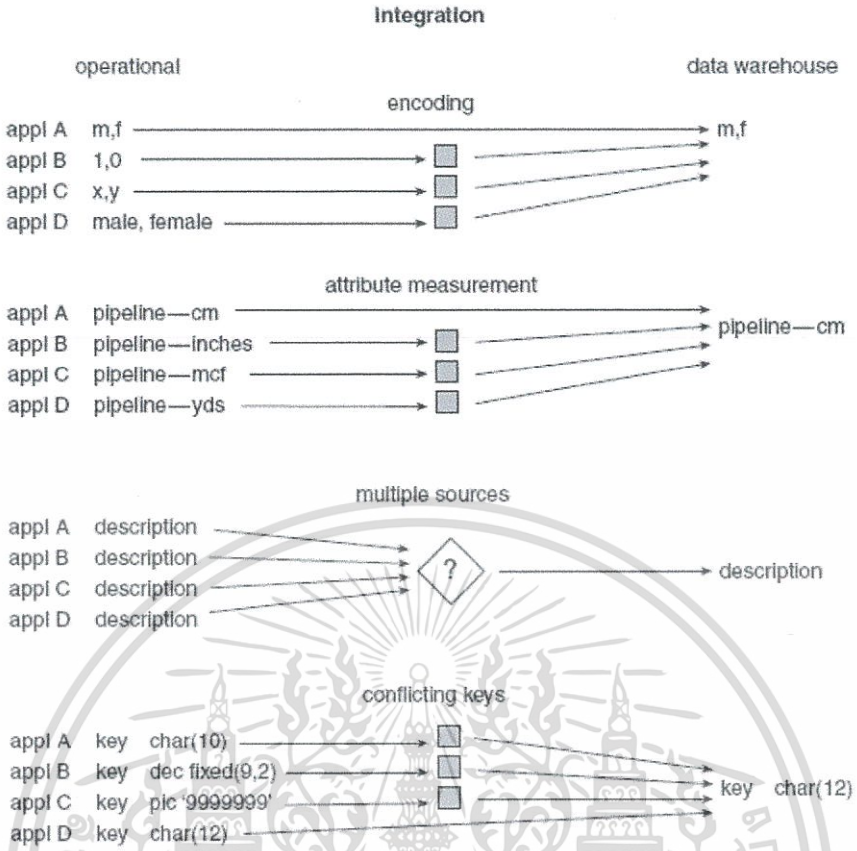
คลังข้อมูลมีคุณลักษณะที่สำคัญ 4 ประการดังนี้

2.3.2.1 Subject-Oriented

คือการจัดกลุ่มข้อมูลตามประเด็นหลักขององค์กรที่สนใจจะนำมาสร้างเป็นคลังข้อมูล ซึ่งข้อมูลจะถูกสร้างขึ้นมาจากหัวข้อธุรกิจที่สนใจ เช่น ข้อมูลลูกค้า ข้อมูลสินค้า หรือข้อมูลยอดขาย ซึ่งข้อมูลที่สร้างขึ้นจะประกอบด้วยส่วนข้อมูลที่เกี่ยวข้องกับหัวข้อนั้นๆ เท่านั้น เพื่อที่จะประกอบกันเป็นข้อมูลที่ใช้ในการวิเคราะห์

2.3.2.2 Integrated

คือการจัดข้อมูลต่างรูปแบบให้อยู่ในรูปแบบเดียวกัน ซึ่งจะต้องมีความสอดคล้องไม่ขัดแย้งกันในคลังข้อมูลเดียวกัน เนื่องจากข้อมูลถูกรวบรวมมาจากแหล่งข้อมูลต่างๆ อีกทั้งยังมีรูปแบบข้อมูลที่หลากหลาย ดังนั้นก่อนจะทำการสร้างคลังข้อมูล สิ่งที่ต้องทำเป็นอันดับแรกคือการกำหนดค่าตัวแปรของฐานข้อมูลให้เหมือนกันเป็นหนึ่งเดียวก่อน

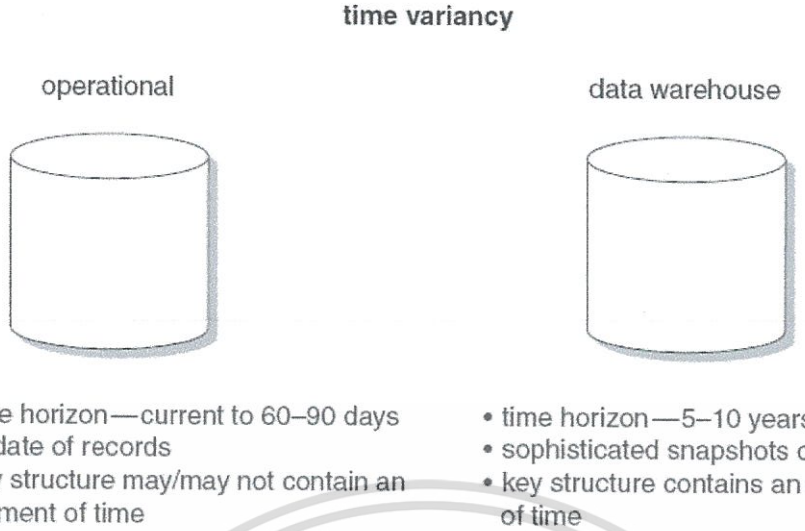


รูป 2.11 การรวบรวมข้อมูลก่อนที่จะนำเข้าสู่คลังข้อมูล[2]

2.3.2.3 Time-Variant

คลังข้อมูลเป็นการนำข้อมูลที่เก็บไว้เป็นระยะเวลานาน เช่น ข้อมูลในช่วง 5 ปีที่แล้วเพื่อนำมาทำนายแนวโน้มความสัมพันธ์ของข้อมูลในแต่ละช่วงเวลา (Time Granularity) ตามที่ผู้ใช้งานกำหนด ซึ่งตัวกำหนดช่วงเวลานั้นโดยส่วนมากในทางคลังข้อมูลมักจะใช้ Time Dimension เป็นตัวระบุช่วงเวลา ซึ่งต่างจากการเก็บข้อมูลใน Operational Database คือ Operational Database จะเก็บแค่ข้อมูลที่เป็นจริงเฉพาะในปัจจุบันเท่านั้น (Current Value) ในขณะที่โครงสร้างของคลังข้อมูลจะต้องมีองค์ประกอบของเวลาเข้ามารวมอยู่ด้วย

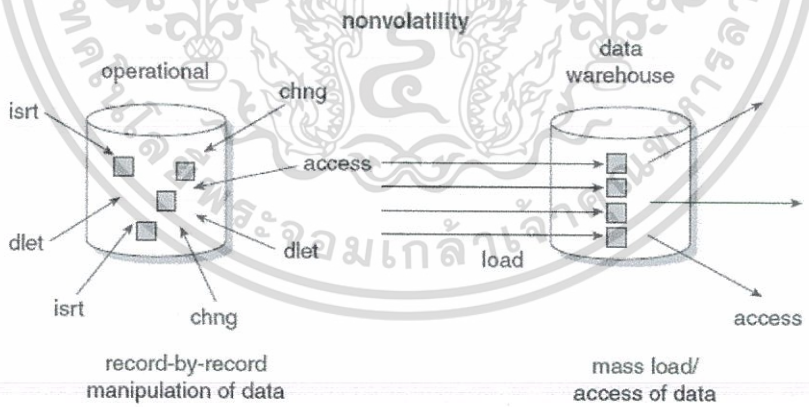
[2] W.H Inmon Buiding the Data Warehouse Third Edition



รูป 2.12 การจัดเก็บข้อมูลแบบระยะสั้นในฐานข้อมูลประจำวัน และระยะยาวใน Data Warehouse[2]

2.3.2.4 Nonvolatile

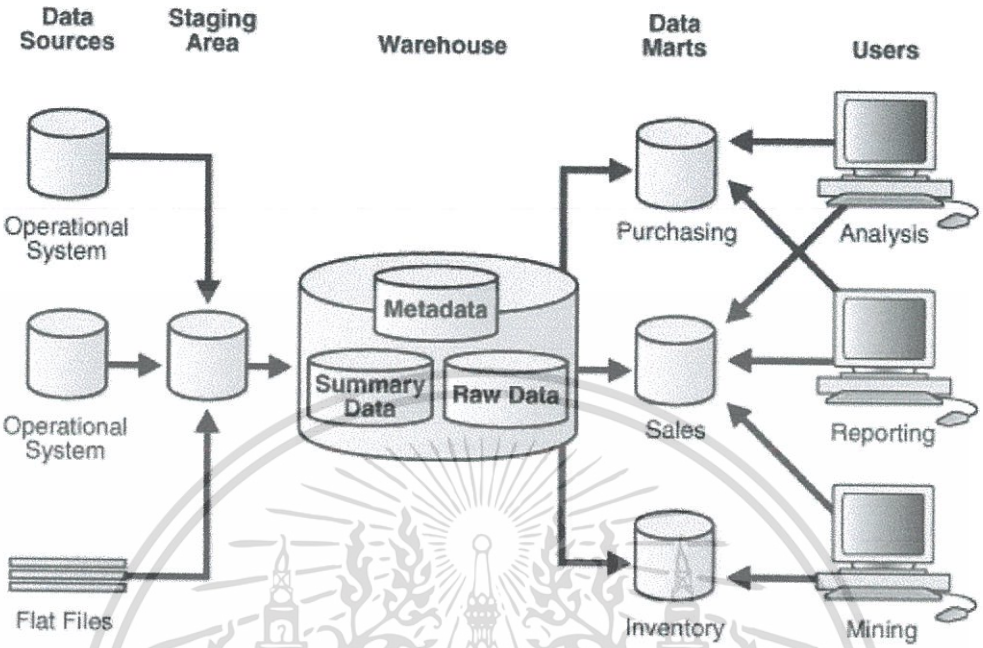
เมื่อมีการเปลี่ยนแปลงข้อมูลที่มีการโหลดข้อมูลจาก Operational Database ไปยังคลังข้อมูล ข้อมูลในคลังข้อมูลนั้นจะไม่ได้มีการเปลี่ยนแปลงอีกหลังจากที่ถูกโหลด โดยกระบวนการโหลดข้อมูลจาก Operational Database นั้นจะเรียกกระบวนการนี้ว่า การทำ Extract Transfer Load (ETL)



รูป 2.13 ลักษณะของการจัดเก็บข้อมูลแบบไม่มีการเปลี่ยนแปลง[2]

[2] W.H Inmon Buiding the Data Warehouse Third Edition

2.3.3 องค์ประกอบของ Data Warehouse



รูป 2.14 องค์ประกอบของคลังข้อมูล[2]

2.3.3.1 Date Staging Area

เป็นส่วนที่ให้ข้อมูลมาพักไว้ก่อนที่จะเข้าสู่คลังข้อมูล ซึ่งในส่วนนี้จะมีกระบวนการหลักคือการคัดเลือกข้อมูล การรวบรวมข้อมูล การทำข้อมูลให้เป็นมาตรฐาน ซึ่งส่วนพักข้อมูลนั้นจะเป็นทั้งส่วนที่เก็บข้อมูล และเป็นส่วนของการบวนการ ETL

Extract: การดึงข้อมูล คือการพยายามเข้าไปอ่านและพยายามเข้าถึงแหล่งข้อมูล

Transformation: จะเป็นกระบวนการที่มีการคัดเลือก จัดข้อมูลให้ถูกต้องและมีการรวมข้อมูลที่มาจากหลายแหล่งเข้าด้วยกัน

Load: เป็นกระบวนการที่จะโหลดข้อมูลเข้าไปยังคลังข้อมูล ซึ่งจะโหลดเข้าไปจัดเก็บไว้ยัง Dimension Table และ Fact Table

2.3.3.2 Data Presentation Area

เป็นส่วนที่ต้องออกแบบว่าจะจัดเก็บข้อมูลอย่างไร โดยจะมี Data Mart เป็นเสมือนวิวของคลังข้อมูลอีกทีหนึ่ง ซึ่งจุดประสงค์ของ Data Mart นั้นมีไว้เพื่อช่วยคัดกรองข้อมูลของแต่ละแผนกในองค์กรที่ไม่เกี่ยวข้องกัน อีกทั้งยังสามารถใช้ในการกำหนดสิทธ์ในการที่จะเข้าถึงข้อมูลของแต่ละแผนกในองค์กรได้อีกด้วย โดยที่ Data Mart แต่ละ Mart นั้นอาจจะมีการใช้ Dimension Table ร่วมกันได้ โดยในส่วนการเก็บข้อมูลนั้นจะมีหลักในการออกแบบที่เกี่ยวข้องสองอย่างคือ Fact table และ Dimension table

[2] W.H Inmon Buiding the Data Warehouse Third Edition

2.3.3.3 Data Mart

เป็นคลังข้อมูลขนาดเล็กที่คัดกรองเฉพาะข้อมูลที่เกี่ยวข้องกับสิ่งที่สนใจเท่านั้น โดยข้อมูลนั้นจะแยกย่อยจาก Data Warehouse อีกทีหนึ่ง เพื่อนำไปใช้สนับสนุนการทำงานของแต่ละแผนก ซึ่งเป็นข้อมูลสำหรับการดำเนินการทางธุรกิจเพียงธุรกิจเดียว (Single Business Process) เช่นดูเฉพาะเรื่องการขายของบริษัทเท่านั้น (โดยทั่วไปจะให้ 1 Mart ต่อ 1 Business Process) ซึ่งการทำ Mart นั้นยังช่วยในเรื่องของความรวดเร็วในการสืบค้นข้อมูล (Performance) อีกด้วย เนื่องจากในงานคลังข้อมูลนั้น จะต้องใช้ข้อมูลปริมาณมากในการวิเคราะห์การเปลี่ยนแปลง ดังนั้นการแยกข้อมูลสำหรับแต่ละบิสิเนสโปรเซสจึงเป็นเรื่องที่สำคัญ

2.3.3.4 Meta Data

เนื่องจากคลังข้อมูลนั้นเป็นพื้นที่สำหรับเก็บข้อมูลขนาดใหญ่ ดังนั้นย่อมมีความยุ่งยากกว่าการบริหารข้อมูลที่มีขนาดเล็กหรือปานกลาง ดังนั้นจึงจำเป็นต้องสร้าง Meta Data ขึ้นมาเพื่ออธิบายข้อมูลที่อยู่ในคลังข้อมูลอีกทีหนึ่ง ซึ่งใน Meta Data จะเก็บข้อเท็จจริงต่างๆ ที่เกี่ยวข้องกับข้อมูลในทุกแง่มุมเอาไว้ ทั้งนี้เพื่อสองวัตถุประสงค์คือ

1) เพื่ออธิบายความหมายหรือคำจำกัดความของข้อมูล
2) เพื่อใช้เป็นข้อมูลสำหรับการดำเนินงานต่างๆกับข้อมูลตามกระบวนการของคลังข้อมูล

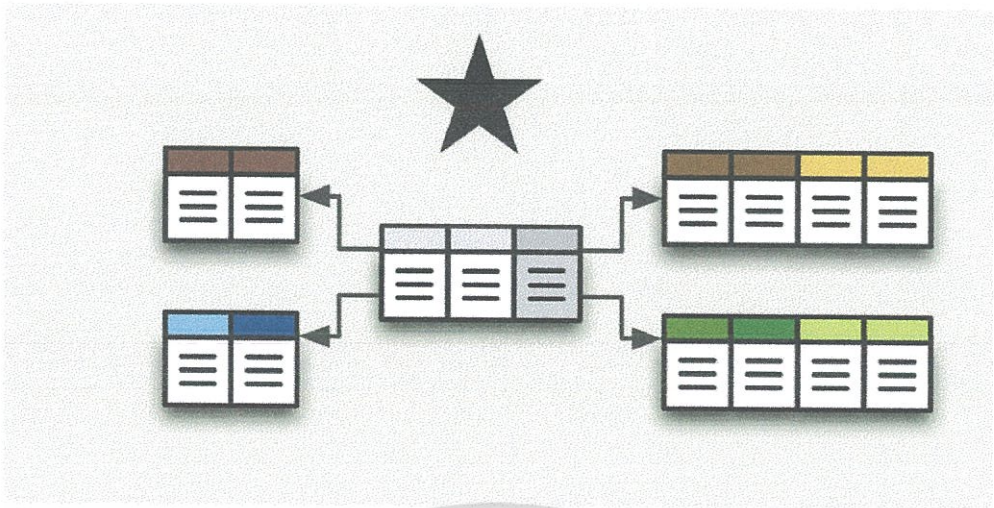
2.3.4 แบบจำลองข้อมูลสำหรับ Data Warehouse (Data Model for Data Warehouse)

2.3.4.1 Dimensional Data Model

1) **Measures** หมายถึง ข้อมูลที่ต้องการวัด ทั้งในเชิงปริมาณ และเชิงคุณภาพของสิ่งใดสิ่งหนึ่งเช่น ยอดขายรวม กำไร ค่าธรรมเนียม เป็นต้น ซึ่ง Measure จะมีชนิดของข้อมูลเป็นตัวเลขเสมอ
2) **Dimension** หมายถึง ข้อมูลที่เป็นมุมมองให้แก่ Measure เพื่อประโยชน์ในการวิเคราะห์ข้อมูล เช่น เวลา จังหวัด อำเภอ เป็นต้น
3) **Facts** หมายถึง ชุดของข้อมูลที่เกิดจากการจับคู่กันของ Dimension และ Measure ที่ทำให้เกิดค่าหนึ่งที่มีความหมายสามารถวัดได้ และบอกเล่าข้อเท็จจริงอย่างใดอย่างหนึ่ง

2.3.5 Star Schema

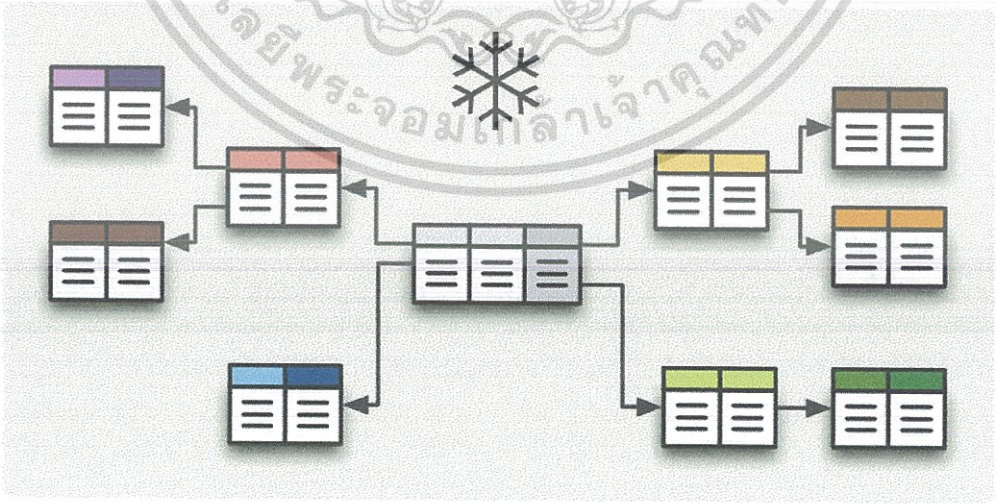
Star Schema หมายถึง Dimension Data Model ที่มี Fact Table ขนาดใหญ่เพียงหนึ่งเดียวอยู่ตรงกลางและมี Dimension Table จำนวนหนึ่งอยู่รอบข้างเพื่อกำหนดมุมมองที่จะมีต่อ Measure ใน Fact Table นั้น โดยจำนวนมุมมองที่มองได้จะเท่ากับจำนวนของ Dimension Table ที่รายรอบอยู่ และเท่ากับจำนวน Dimension ที่เชื่อมต่อโดยตรงจาก Fact Table



รูป 2.15 รูปแบบของ Star schema

2.3.6 Snowflake Schema

Snowflake Schema หมายถึง Dimension Data Model ที่มี Fact Table ขนาดใหญ่เพียงหนึ่งเดียวอยู่ตรงกลาง และมี Dimensional Table จำนวนหนึ่งรายรอบอยู่เพื่อที่จะกำหนดมุมมองที่จะมีต่อ Measure ใน Fact Table นั้น โดยจำนวนมุมมองที่มองได้จะเท่ากับจำนวน Dimension Table ที่รายรอบอยู่ และจะมากกว่าจำนวน Dimension ที่เชื่อมต่อโดยตรงกับ Fact Table โดยที่ Dimension ที่ไม่ได้เชื่อมต่อโดยตรงกับ Fact Table จะมีความสัมพันธ์กับ Dimension ตัวอื่นๆ จึงสามารถสรุปได้ว่า Snowflake Schema คือรูปแบบการเขียน Star Schema ที่จัด Dimension ให้อยู่ในรูป Normal Form



รูป 2.16 รูปแบบของ Snowflake Schema

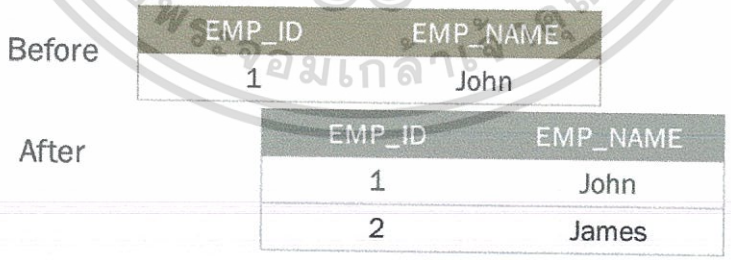
2.3.7 Slowly Changing Dimension

Slowly Changing Dimension นั้นเป็นปัญหาในการทำคลังข้อมูลที่เกิดขึ้นเมื่อ Dimension Table มีการปรับปรุง ดังนั้นจึงมีรูปแบบในการแก้ไขปัญหาที่เกิดขึ้น 3 รูปแบบด้วยกัน ซึ่งแต่ละแบบนั้นมีข้อดีข้อเสียแตกต่างกันไป ขึ้นอยู่กับผู้ออกแบบระบบว่าจะแก้ไขกับปัญหานี้ได้อย่างไร

1) Overwriting the Old Value เป็นการจัดการกับข้อมูลเมื่อมีการปรับปรุงข้อมูลใน Dimension Table จะทำการเขียนทับข้อมูล Version ก่อนหน้า ซึ่งหากเลือกวิธีนี้ในการจัดการกับข้อมูลที่มีการปรับปรุงจะไม่สามารถติดตามข้อมูลในอดีตได้ แต่มีข้อดีคือไม่เปลืองพื้นที่เหมาะสำหรับงานที่ไม่ต้องการความแม่นยำในการวิเคราะห์ข้อมูลและไม่ต้องการที่จะดูการเปลี่ยนแปลงของข้อมูลในอดีต

รูป 2.17 การจัดการปัญหา Slowly Changing Dimension รูปแบบที่ 1

2) Creating a New Key วิธีนี้จะเพิ่ม Primary Key ของข้อมูลมาใหม่ เมื่อข้อมูลมีการปรับปรุง ก็จะเพิ่มแถว เพื่อจัดเก็บข้อมูลใหม่ลงไป ข้อเสียคือ เจะไม่สามารถรู้ได้เลยว่า คนสองคนนี้เป็นคนเดียวกัน



รูป 2.18 การจัดการปัญหา Slowly Changing Dimension รูปแบบที่ 2

3) Adding a New Column เพิ่มคอลัมน์สำหรับเก็บค่า Current และ Previous ของข้อมูลก่อนที่จะมีการ ปรับปรุงเอาไว้เพื่อให้มีการติดตามเวอร์ชั่นก่อนการปรับปรุงได้ 1 ครั้ง ข้อดีคือ ไม่เปลืองเนื้อที่ในการเก็บข้อมูลที่มีการเปลี่ยนแปลงและสามารถติดตามการเปลี่ยนแปลงของข้อมูลได้บ้างส่วน แต่ข้อเสียคือ หากข้อมูลมีการเปลี่ยนแปลงมาก ก็ไม่สามารถติดตามได้ทุกเวอร์ชั่นของข้อมูลนั้น

Before

EMP_ID	EMP_NAME
1	John

After

EMP_ID	EMP_NAME	Status
1	John	Old
1	James	New

รูป 2.19 การจัดการปัญหา Slowly Changing Dimension รูปแบบที่ 3

บทที่ 3

การออกแบบและพัฒนาซอฟต์แวร์

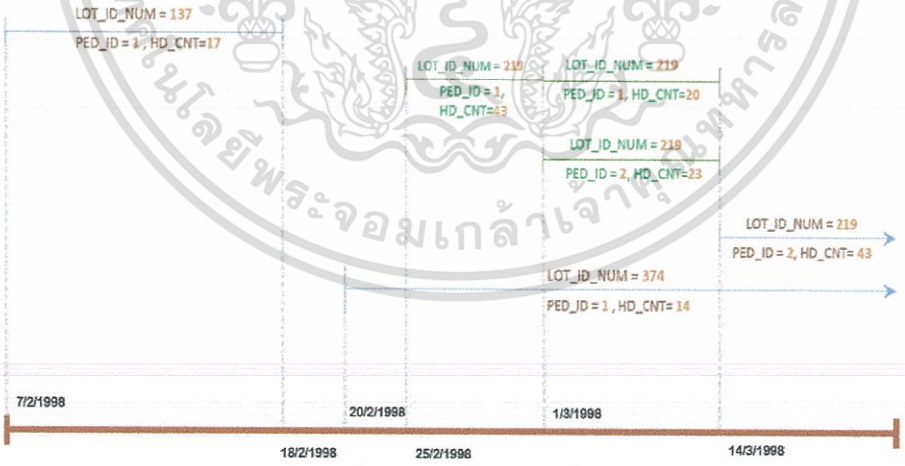
3.1 การทดสอบตัวดำเนินการทางเวลา (Operators for Valid Time)

เนื่องจากบางโปรแกรมต้องมีการจัดเก็บข้อมูลที่เกี่ยวข้องช่วงเวลาเพื่อความถูกต้องของข้อมูล ดังนั้นเมื่อทำการบันทึกข้อมูลจึงต้องมีการบันทึกช่วงเวลาเข้าไปด้วย และการสอบถามข้อมูลก็ต้องมีการระบุช่วงเวลา

ฐานข้อมูลต่อไปนี้เป็นฐานข้อมูลที่แสดงรายละเอียดของฝูงวัวที่อยู่ในช่วงเวลาต่างๆ ซึ่งจะใช้ในการทดสอบกับตัวดำเนินการที่มีอยู่ใน OWM

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

รูป 3.1 รายละเอียดของวัวแต่ละฝูงว่าเคยอยู่คอกใดในช่วงเวลาใดบ้าง



รูป 3.2 ผลอยู่ในรูปของเส้นเวลา

คำถาม: วัวฝูงใดบ้างอยู่คอกเดียวกัน?

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	HD_CNT	HD_CNT
137	219	1	17	43
137	219	1	17	20
137	374	1	17	14
219	374	1	20	14
219	374	1	43	14

รูป 3.3 ข้อมูลฝูงวัวทั้งหมด

3.1.1 WM_OVERLAPS

เป็นการตรวจสอบ 2 Periods ที่ซ้อนทับกัน

โปรแกรม 3.1 การใช้ Operation WM_OVERLAPS

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME);
SELECT e1.lot_id_num,
e2.lot_id_num,e1.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num < e2.lot_id_num
AND e1.pen_id = e2.pen_id AND
WM_OVERLAPS(e1.WM_VALID,e2.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	HD_CNT	HD_CNT	FROM_DATE	TO_DATE
219	374	1	20	14	1998-03-01	1998-03-14
219	374	1	43	14	1998-02-25	1998-03-01

รูป 3.4 ผลลัพธ์เมื่อใช้ Operation WM_OVERLAPS

3.1.2 WM_CONTAINS

เป็นการตรวจสอบว่า Period แรก อยู่ใน Period สอง

โปรแกรม 3.2 การใช้ Operation WM_CONTAINS

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e2.lot_id_num,
e1.lot_id_num,e1.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num < e2.lot_id_num
AND e1.pen_id = e2.pen_id AND
WM_CONTAINS(e2.WM_VALID,e1.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	HD_CNT	HD_CNT	FROM_DATE	TO_DATE
374	219	1	20	14	1998-03-01	1998-03-14
374	219	1	43	14	1998-02-25	1998-03-01

รูป 3.5 ผลลัพธ์เมื่อใช้ Operation WM_CONTAINS

3.1.3 WM_MEETS

เป็นการตรวจสอบว่าจุดสิ้นสุด Period แรก เป็นจุดเริ่มต้นของ Period สอง

โปรแกรม 3.3 การใช้ Operation WM_MEETS

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e2.lot_id_num,
e1.lot_id_num,e1.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num <= e2.lot_id_num
AND WM_MEETS(e1.WM_VALID,e2.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	PEN_ID	HD_CNT	HD_CNT	FROM_DATE	TO_DATE
219	219	1	1	43	20	1998-02-25	1998-03-01
219	219	1	2	43	23	1998-02-25	1998-03-01
219	219	1	2	20	43	1998-03-01	1998-03-14
219	219	2	2	23	43	1998-03-01	1998-03-14

รูป 3.6 ผลลัพธ์เมื่อใช้ Operation WM_MEETS

3.1.4 WM_EQUALS

เป็นการตรวจสอบว่าทั้งสอง Periods มีค่าเท่ากัน (จุดเริ่มต้นและสุดสิ้นสุดเป็นจุดเดียวกัน)

โปรแกรม 3.4 การใช้ Operation WM_EQUALS

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e2.lot_id_num,
e1.lot_id_num,e1.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num <= e2.lot_id_num
AND WM_MEETS(e1.WM_VALID,e2.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	PEN_ID	HD_CNT	HD_CNT	FROM_DATE	TO_DATE
219	219	1	2	20	23	1998-03-01	1998-03-14

รูป 3.7 ผลลัพธ์เมื่อใช้ Operation WM_EQUALS

3.1.5 WM_LESSTHAN

เป็นการตรวจสอบว่าจุดสิ้นสุดของ Period แรกน้อยกว่าจุดเริ่มต้นของ Period สอง (Period แรกสิ้นสุดก่อนที่ Period สองจะเกิด)

โปรแกรม 3.5 การใช้ Operation WM_LESSTHAN

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e1.lot_id_num, e2.lot_id_num,
e1.pen_id,e2.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num < e2.lot_id_num
AND WM_LESSTHAN(e1.WM_VALID,e2.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	PEN_ID	HD_CNT	HD_CNT	FROM_DATE	TO_DATE
137	219	1	1	17	43	1998-02-07	1998-02-18
137	219	1	1	17	20	1998-02-07	1998-02-18
137	219	1	2	17	23	1998-02-07	1998-02-18
137	219	1	2	17	43	1998-02-07	1998-02-18
137	374	1	1	17	14	1998-02-07	1998-02-18

รูป 3.8 ผลลัพธ์เมื่อใช้ Operation WM_LESSTHAN

3.1.6 WM_GREATERTHAN

เป็นการตรวจสอบว่าจุดเริ่มของ Period แรกมากกว่าจุดสิ้นสุดของ Period สอง (Period แรกเกิดหลังจากที่ Period สองสิ้นสุดไปแล้ว)

โปรแกรม 3.6 การใช้ Operation WM_GREATERTHAN

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e1.lot_id_num, e2.lot_id_num,
e1.pen_id,e2.pen_id,e1.HD_CNT,e2.HD_CNT,
TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
FROM lotcow e1, lotcow e2
WHERE e1.lot_id_num > e2.lot_id_num
AND WM_LESSTHAN(e1.WM_VALID,e2.WM_VALID)=1;
```

ผลลัพธ์ที่ได้

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

รูป 3.9 ผลลัพธ์เมื่อใช้ Operation WM_GREATERTHAN

Operation ที่จะ return เป็น period

3.1.7 WM_INTERSECTION

จะคืนค่า Period ที่ทั้งสอง Period ซ้อนทับกัน

LOT_ID_NUM	LOT_ID_NUM	PEN_ID	HD_CNT	HD_CNT	WM_INTERSECTION(E1.WM_VALID,WM_PERIOD(TO_DATE('02-25-1998','MM-DD
137	219	1	17	43	
137	219	1	17	20	
137	374	1	17	14	
219	374	1	20	14	WMSYS.WM_PERIOD('1998-03-01 00:00:00.0','1998-03-14 00:00:00.0')
219	374	1	43	14	WMSYS.WM_PERIOD('1998-02-25 00:00:00.0','1998-03-01 00:00:00.0')

รูป 3.10 แสดงผลลัพธ์เมื่อใช้ Operation WM_INTERSECTION

3.1.8 WM_LDIFF

จะคืนค่า Period ที่ต่างกันระหว่างสอง Period ซึ่งค่า Period ที่คืนค่ามานั้นเป็นค่าที่อยู่ทางฝั่งซ้าย

โปรแกรม 3.7 การใช้ Operation WM_LDIFF

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e1.lot_id_num, e1.pen_id,e1.HD_CNT,
WM_LDIFF(e1.WM_VALID, wm_period(TO_DATE('02-25-1998',
'MM-DD-YYYY'),TO_DATE('03-14-1998', 'MM-DD-YYYY')))
FROM lotcow e1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	PEN_ID	HD_CNT	WM_LDIFF(E1.WM_VALID,WM_PERIOD(TO_DATE('02-25-1998','MM-DD-YYYY'),TO_D
137	1	17	WMSYS.WM_PERIOD('1998-02-07 00:00:00.0','1998-02-18 00:00:00.0')
219	1	43	
219	1	20	
219	2	23	
219	2	43	
374	1	14	WMSYS.WM_PERIOD('1998-02-20 00:00:00.0','1998-02-25 00:00:00.0')

รูป 3.11 ผลลัพธ์เมื่อใช้ Operation WM_LDIFF

3.1.9 WM_RDIFF

จะคืนค่า Period ที่ต่างกันระหว่างสอง Period ซึ่งค่า Period ที่คืนค่ามานั้นเป็นค่าที่อยู่ทางฝั่งขวา

โปรแกรม 3.8 การใช้ Operation WM_RDIFF

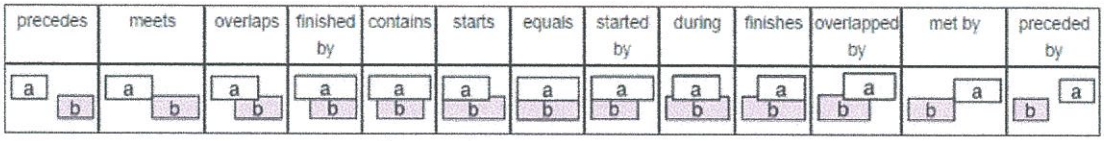
```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
SELECT e1.lot_id_num, e1.pen_id,e1.HD_CNT,
WM_RDIFF(e1.WM_VALID, wm_period(TO_DATE('02-25-1998',
'MM-DD-YYYY'),TO_DATE('03-14-1998', 'MM-DD-YYYY')))
FROM lotcow e1;
```

ผลลัพธ์ที่ได้

LOT_ID_NUM	PEN_ID	HD_CNT	WM_RDIFF(E1.WM_VALID,WM_PERIOD(TO_DATE('02-25-1998','MM-DD-YYYY'),TO_DATE('
137	1	17	
219	1	43	
219	1	20	
219	2	23	
219	2	43	WMSYS.WM_PERIOD('1998-03-14 00:00:00.0','9999-12-31 23:59:59.999999')
374	1	14	WMSYS.WM_PERIOD('1998-03-14 00:00:00.0','9999-12-31 23:59:59.999999')

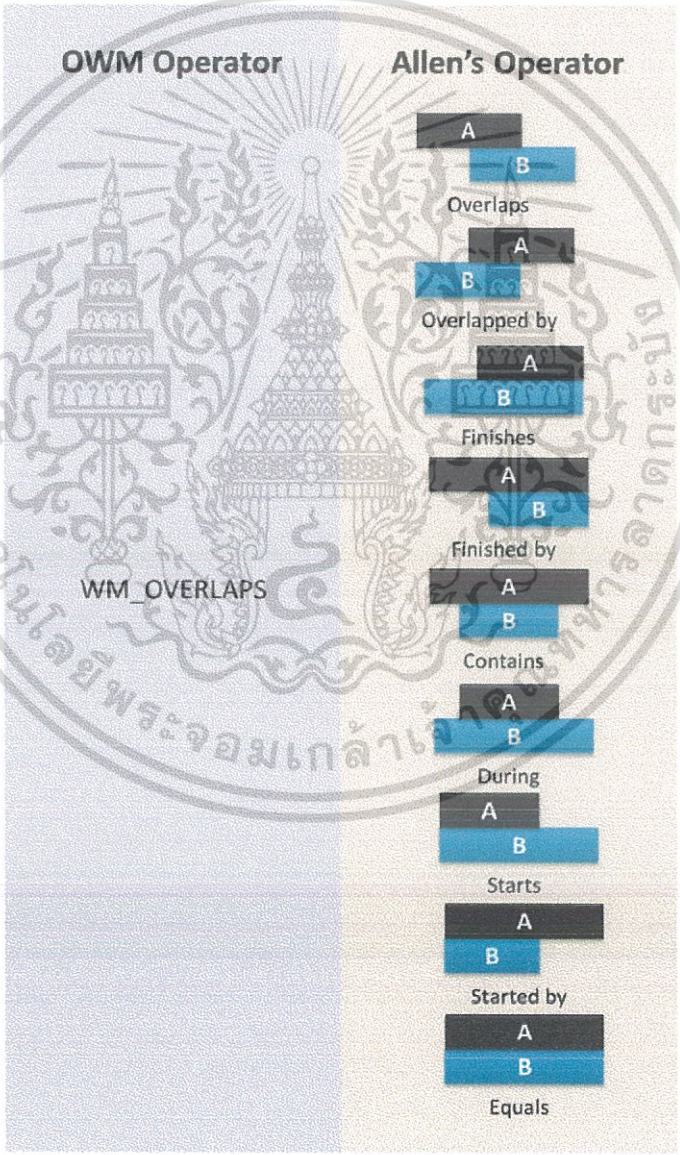
รูป 3.12 ผลลัพธ์เมื่อใช้ Operation WM_RDIFF

จากตัวอย่างการใช้งานตัวดำเนินการทางเวลาข้างต้นนั้นมีรูปแบบการนำเสนอเวลาคล้ายคลึงกับ Allen's Operators ซึ่ง Operators ของ Allen นั้นสามารถแสดงเป็นแผนภาพได้ดังนี้



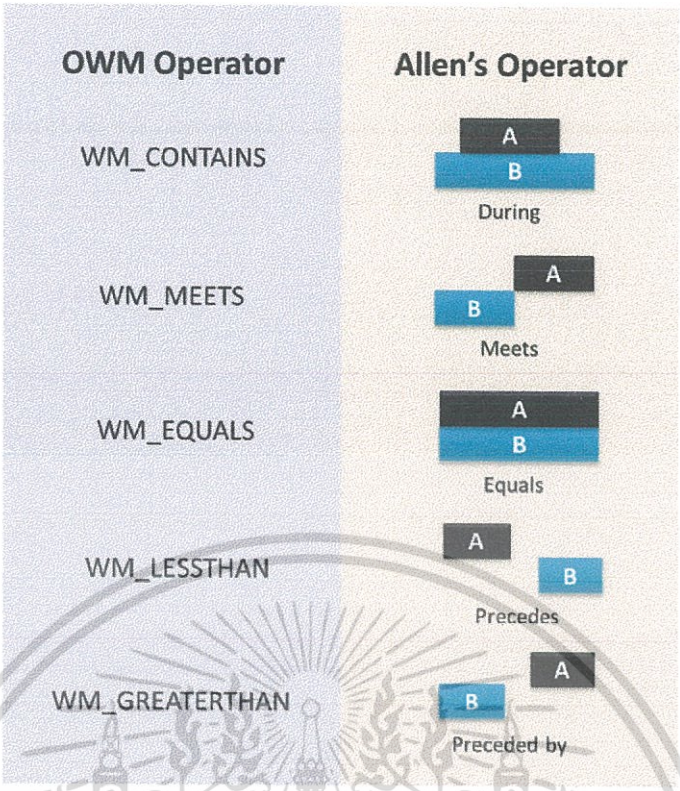
รูป 3.13 Allen's Interval Algebra[3]

เราสามารถเปรียบเทียบนิยามของตัวดำเนินการทางเวลาของ Allen กับตัวดำเนินการทางเวลาของ OWM ได้ดังนี้



รูป 3.14 การเปรียบเทียบ WM operators กับ Allen's Operators

[3] SQL for date ranges, gaps and overlaps



รูป 3.14 การเปรียบเทียบ WM operators กับ Allen's Operators (ต่อ)

3.2 System Architecture

โครงสร้างของระบบจะเป็นออกเป็น 3 Module ใหญ่ๆ คือ

3.2.1 Data Warehouse is Non-Temporal Data

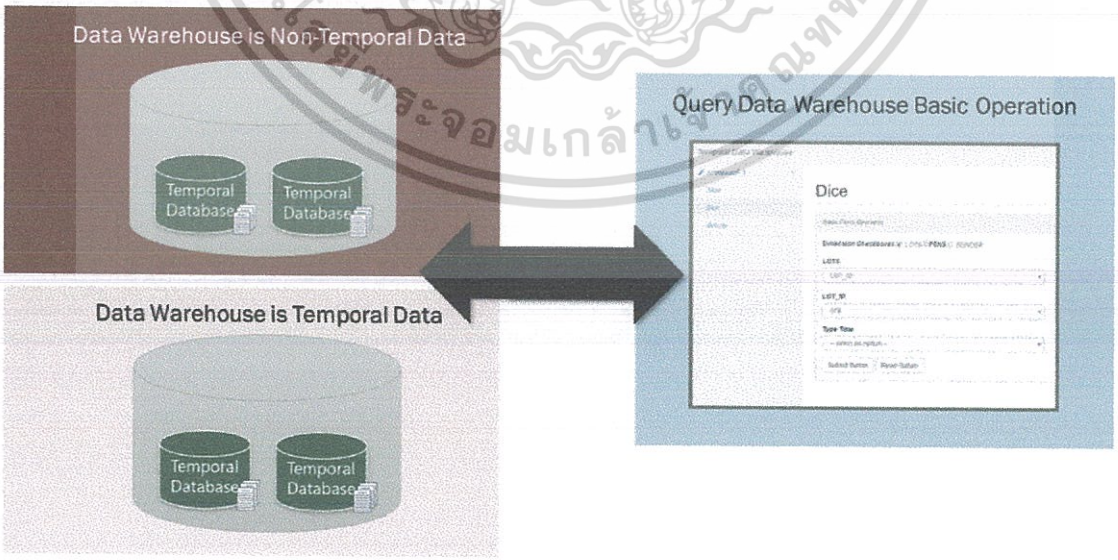
เป็นกรณีศึกษาที่ 1 ที่ Data Warehouse มี Source เป็น Temporal Database แต่ตัว Data Warehouse เองไม่ได้เป็น Temporal Data โดยใน Module นี้ เราจะทำการแปลงตัว Source ที่เป็น Temporal Database เข้าสู่ Data Warehouse โดย Extract Valid Time ของตัว Source เทียบกับ Timekey ใน Time Dimension แล้วจึงโหลดเข้า Fact Table

3.2.2 Data Warehouse is Temporal Data

เป็นกรณีศึกษาที่ 2 ที่ Data Warehouse มี Source เป็น Temporal Database และตัว Data Warehouse เองก็ยังเป็น Temporal Data ด้วย เราจะทำการนำ Source ที่เป็น Temporal Database เข้าสู่ Data Warehouse แต่เนื่องจากตัว Data Warehouse เองก็เป็น Temporal Data ดังนั้นเราสามารถใช้ Valid Time ที่ตัว Source เป็น Valid Time ของ Dimension ที่เป็น Temporal ได้ แต่เรายังคง Extract Valid Time เทียบกับ Timekey ใน Time Dimension เพื่อโหลดเข้า Fact Table และใน Fact Table ยังมี Valid Time ของ Timekey อีกด้วย (Valid Time ของ 1 วัน)

3.2.3 Data Warehouse Basic Operation

เป็น Module ส่วนที่ใช้ติดต่อกับ Data Warehouse โดยในโปรเจคนี้ จะใช้ Website เป็น User Interface เพื่อให้ User สามารถใช้งานในการตอบคำถามต่างๆ ได้ง่ายขึ้น ไม่ว่าจะเป็นการใช้งานฟังก์ชั่นพื้นฐานของ Data Warehouse (Slice, Dice, Rollup) หรือการตอบคำถามอื่นๆ แต่เบื้องหลังของ Module นี้เป็น OWM เป็นตัวติดต่อกับ Data Warehouse เอง

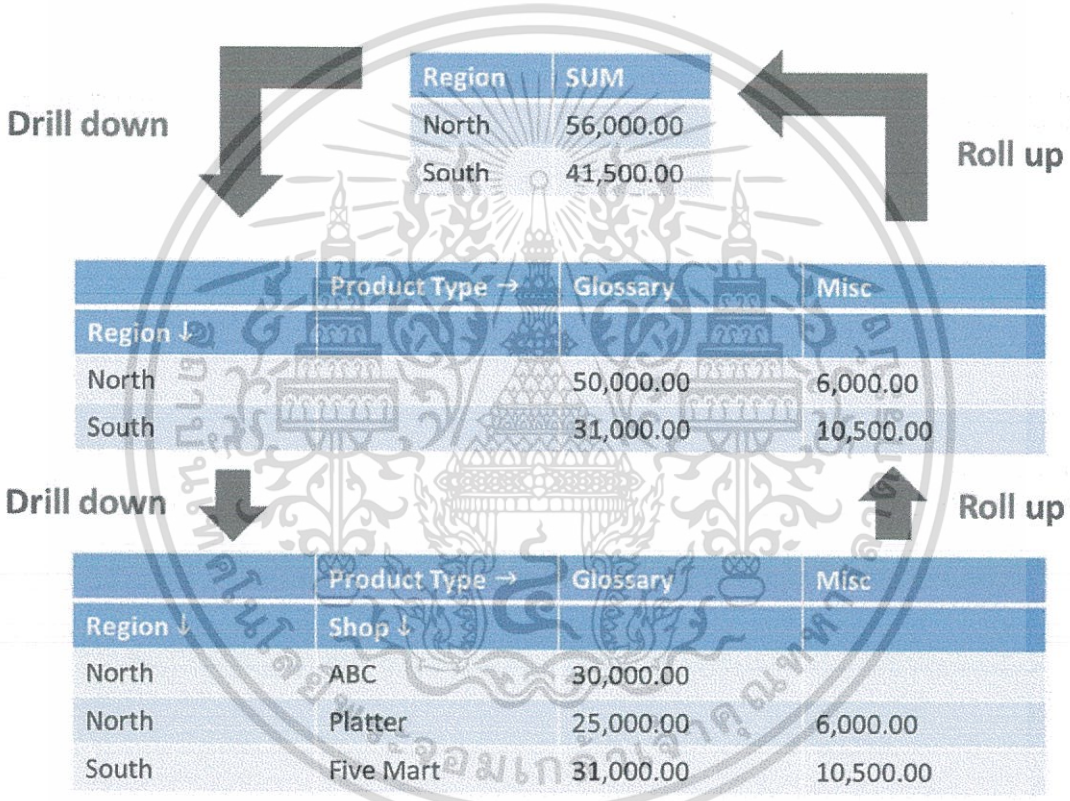


รูป 3.15 System Architecture

3.3 Operation พื้นฐานที่ใช้งานกับ Data Warehouse

3.3.1 Roll up และ Drill down

Roll up และ Drill down เป็นการเปลี่ยนแปลงระดับความละเอียดของการพิจารณาข้อมูล โดยที่ Roll up จะเป็นการเปลี่ยนแปลงระดับความละเอียดของการพิจารณาข้อมูลจากระดับที่ละเอียดไปสู่ระดับที่หยาบมากขึ้น ในขณะที่ Drill down ทำตรงกันข้ามกันคือ เปลี่ยนแปลงระดับความละเอียดของการพิจารณาข้อมูลจากระดับที่หยาบไปสู่ระดับที่ละเอียดมากขึ้น พิจารณาดังรูป 3.14 จะเป็นกว่า การ Roll up จะเป็นการเปลี่ยนแปลงมุมมองของข้อมูล โดยครั้งแรกที่ Roll up จะได้เป็นผมรวมของยอดขายของแต่ละภูมิภาค (Region) ของแต่ละประเภทสินค้า เมื่อ Roll up อีกครั้งจะได้ผลรวม



Region	SUM
North	56,000.00
South	41,500.00

	Product Type →	Glossary	Misc
Region ↓			
North		50,000.00	6,000.00
South		31,000.00	10,500.00

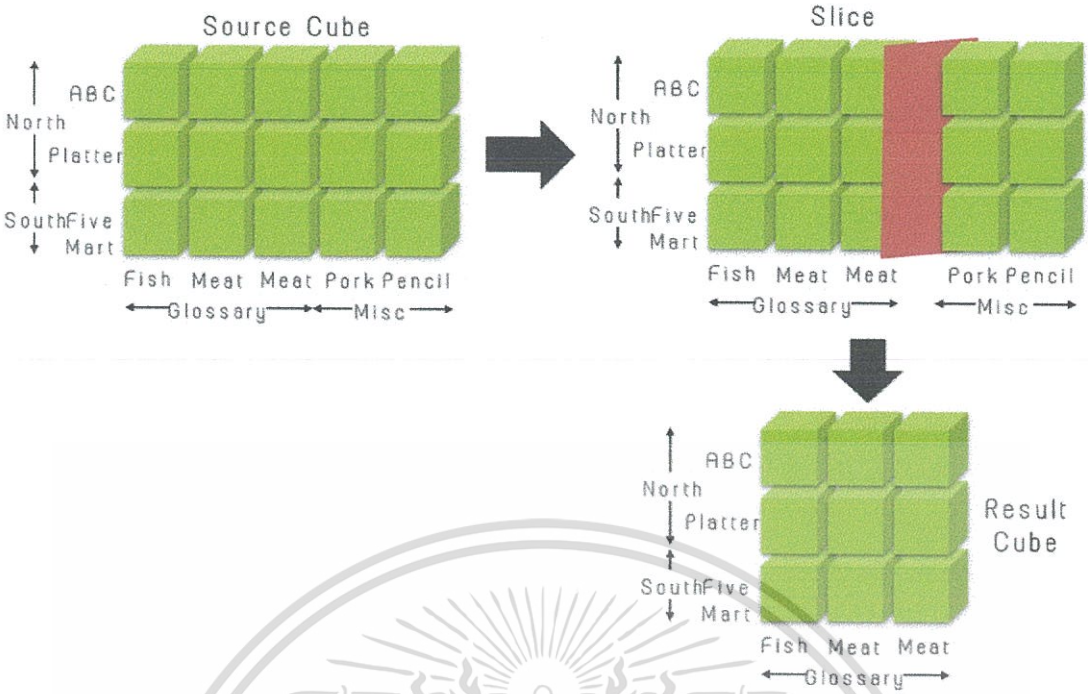
	Product Type →	Glossary	Misc
Region ↓	Shop ↓		
North	ABC	30,000.00	
North	Platter	25,000.00	6,000.00
South	Five Mart	31,000.00	10,500.00

รูป 3.16 การ Roll up และ Drill down

3.3.2 Slice and Dice

3.3.2.1 Slice

Slice คือ การเลือกพิจารณาผลลัพธ์บางส่วนที่เราสนใจ โดยจะตัดส่วนที่สนใจตาม Dimension 1 Dimension พิจารณารูป 3.15 เรา Slice เฉพาะบางประเภทสินค้าที่สนใจ นั่นคือ Glossary



รูป 3.17 การ Slice

3.3.2.2 Dice

Dice คือ การเลือกพิจารณาผลลัพธ์ส่วนที่เราสนใจ โดยมีความซับซ้อนมากขึ้น ใช้ตั้งแต่ 2 Dimension ขึ้นไปในการเลือกพิจารณา พิจารณารูป 3.16 จะเลือกส่วนเฉพาะบาง Shop และ Product Type ที่สนใจ

รูป 3.18 การ Dice

3.4 การออกแบบ Data Warehouse

ในการสรุปข้อมูลจาก Temporal Data นั้นจะแบ่ง Data Warehouse เป็น 2 ลักษณะคือ

3.4.1 กรณี 1 Data Warehouse ที่มี source เป็น Temporal Data และ Dimension table เป็น Non-temporal data

LOT_ID	PEN_ID	AMOUNT	FROM_DATE	TO_DATE
137	1	17	2014-01-07	2014-03-29
219	1	43	2014-02-25	2014-03-01
219	1	20	2014-03-01	2014-03-14
219	2	23	2014-03-01	2014-03-14
219	2	43	2014-03-14	9999-12-31
374	1	14	2014-02-20	9999-12-31

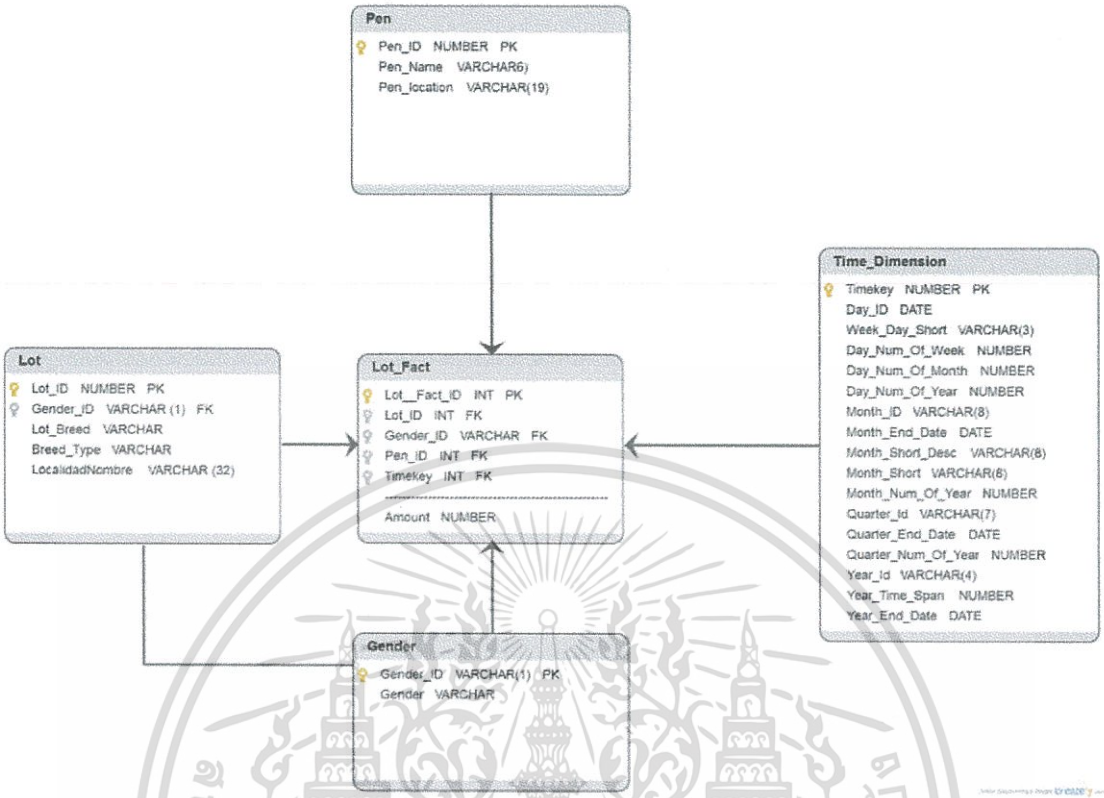
LOT_ID	PEN_ID	AMOUNT	DATE
137	1	17	2014-01-07
137	1	17	2014-01-08
137	1	17	2014-01-09
.	.	.	.
.	.	.	.
.	.	.	.
137	1	17	2014-03-28

LOT_ID	PEN_ID	GENDER_ID	TIMEKEY	AMOUNT
137	1	M	7	17
137	1	M	8	17
137	1	M	9	17
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
137	1	M	88	17

Timekey	Date_ID	Month_ID	Year_ID
1	2014-01-01	2014-01	2014
2	2014-01-02	2014-01	2014
3	2014-01-03	2014-01	2014
.	.	.	.
.	.	.	.
.	.	.	.

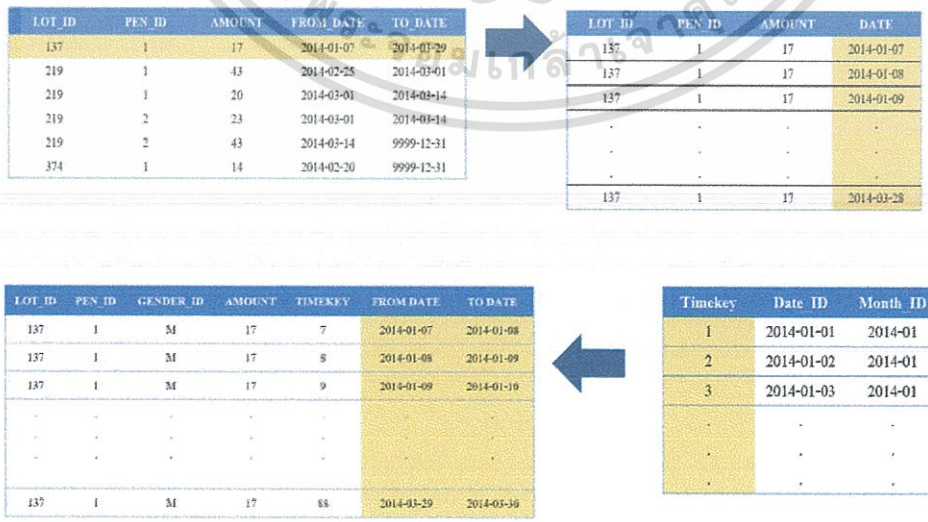
รูป 3.19 การแปลง Valid Time จาก Temporal Database เป็น Data Warehouse ของกรณีที่ 1

เนื่องจาก Source มีลักษณะเป็น Temporal Data มีการบันทึกช่วงเวลาที่เป็นจริงของข้อมูล ดังนั้นเมื่อทำการแปลงข้อมูล จะเทียบช่วงเวลาที่ข้อมูลภายใน Source เป็นจริง กับ Time Dimension โดยขั้นตอนแรกผู้ใช้งานต้องพิจารณาก่อนว่าระดับความละเอียดการเปลี่ยนแปลงเป็นเท่าใด แล้วกำหนด Timekey ภายใน Time Dimension ซึ่งใน Data Warehouse จะมี Surrogate Key เพื่อใช้เป็น Timekey จากตัวอย่างข้างต้นเป็นความละเอียดระดับวัน เมื่อได้ระดับความละเอียดแล้ว ซึ่งเป็นระดับวัน ก็ทำการแปลงข้อมูลที่อยู่ในรูปแบบช่วงเวลาเป็นให้เป็นวันที่ต่างๆ จากนั้นก็เทียบค่ากับ Time Dimension เพื่อให้ทราบ Timekey ทำให้เป็น Dimension table ที่อยู่ในรูปแบบ Non-temporal Data ได้



รูป 3.20 star-schema สำหรับ Data Warehouse ที่มี source เป็น Temporal Data และ Dimension table เป็น Non-temporal data

3.4.2 กรณี 2 Data Warehouse ที่มี source เป็น Temporal Data และ Dimension table เป็น Temporal data



LOT_ID	PEN_ID	AMOUNT	FROM_DATE	TO_DATE
137	1	17	2014-01-07	2014-03-29
219	1	43	2014-02-25	2014-03-01
219	1	20	2014-03-01	2014-03-14
219	2	23	2014-03-01	2014-03-14
219	2	43	2014-03-14	9999-12-31
374	1	14	2014-02-20	9999-12-31

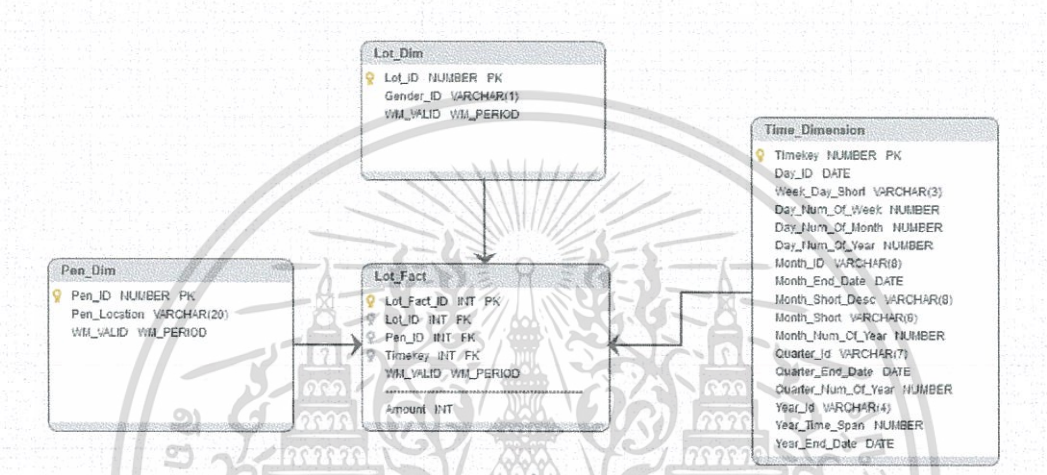
LOT_ID	PEN_ID	AMOUNT	DATE
137	1	17	2014-01-07
137	1	17	2014-01-08
137	1	17	2014-01-09
.	.	.	.
.	.	.	.
.	.	.	.
137	1	17	2014-03-28

LOT_ID	PEN_ID	GENDER_ID	AMOUNT	TIMEKEY	FROM DATE	TO DATE
137	1	M	17	7	2014-01-07	2014-01-08
137	1	M	17	8	2014-01-08	2014-01-09
137	1	M	17	9	2014-01-09	2014-01-10
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.	.	.
137	1	M	17	88	2014-03-29	2014-03-30

Timekey	Date ID	Month ID	Year ID
1	2014-01-01	2014-01	2014
2	2014-01-02	2014-01	2014
3	2014-01-03	2014-01	2014
.	.	.	.
.	.	.	.
.	.	.	.

รูป 3.21 การแปลง Valid Time จาก Temporal Database เป็น Data Warehouse ของกรณีที่ 2

เหมือนกรณีที่ 1 แต่เพิ่มเติมคือ เมื่อทำการเทียบค่ากับ Time Dimension เพื่อให้ทราบ Timekey แล้ว ซึ่งเลข Timekey จะแทนวันหนึ่งวัน เมื่อระหว่างที่ทำการเทียบค่า Timekey กับวันที่ของ Source เมื่อรู้ Timekey ก็ต้องมีการบันทึกด้วยว่า From Date ,To Date เป็นเวลาเท่าใด เช่น เมื่อ Timekey เป็น 7 แสดงว่า From Date มีค่าเป็น 7/01/2014 และ To Date มีค่าเป็น 8/01/2014 เนื่องจากจะคิดวันที่ตั้งแต่ From date ถึงวันที่ก่อน To Date ทำให้เป็น Dimension table ที่อยู่ในรูปแบบ Temporal Data ได้



รูป 3.22 star-schema สำหรับ Data Warehouse ที่มี source เป็น Temporal Data และ Dimension table เป็น Temporal data

3.5 การแก้ปัญหา Slowly Changing Dimensions ใน Temporal Data Warehouse

จากปัญหาที่ได้กล่าวไว้ที่บทที่ 2 เรื่อง Slowly Changing Dimension คือ การที่ Dimension table มีการเปลี่ยนแปลง สามารถใช้ Temporal Data Warehouse แก้ปัญหาได้ดังต่อไปนี้

ในขั้นตอนแรกต้องทำการกำหนดระดับความละเอียดการเปลี่ยนแปลงข้อมูลภายในData Warehouse ซึ่งระดับความละเอียดนั้นขึ้นอยู่กับว่า ใน Dimension table มีการปลี่ยนแปลงบ่อยเพียงใด โดยในOracle Workspace Manager สามารถดึงค่าข้อมูลจาก source ที่มีการเปลี่ยนแปลงได้ถึงระดับวินาที

ตาราง 3.1 Pen

PEN_ID	PEN_NAME	PEN_LOCATION
1	Brig	A1
2	Naters	A2

จากตาราง 3.1 เป็น Dimension Table ของ Pen ซึ่งแสดงรายละเอียด ชื่อคอก และ ตำแหน่งของคอก ซึ่งเดิมหาก pen_id=2 มีการเปลี่ยน pen_location เป็น A3 จะทำให้เกิด slowly changing dimension

ดังนั้นหากมีการประยุกต์ใช้ Temporal Database ที่ Dimension Data เมื่อ Dimension เกิดการเปลี่ยนแปลงคือ pen_location ของ pen_id = 2 เป็น A3 ก็จะมีการบันทึกช่วงเวลาที่ pen_location มีค่าเป็น A2 ซึ่งมีค่าความจริงเป็นจริงในอดีต ดังตารางที่ 3.2

ตาราง 3.2 Pen เมื่อข้อมูลมีการเปลี่ยนแปลง

PEN_ID	PEN_NAME	PEN_LOCATION	From_Date	To_Date
1	Brig	A1	2014-01-07	9999-12-31
2	Naters	A2	2014-01-07	2015-06-06
2	Naters	A3	2015-06-06	9999-12-31

อีกตัวอย่างหนึ่งคือ ตาราง 3.3 เป็นตาราง Gender ที่บอกว่าวัวฝูงไหนเป็นเพศอะไร เช่น ฝูงที่ 137 เป็นเพศเมียที่ยังไม่ถูกตอน (C) ตั้งแต่ 07-01-1998 จนถึงปัจจุบัน แต่หลังจากนั้นไม่นาน ณ วันที่ 10-02-1998 ฝูง 137 ได้ถูกตอน (N) Temporal Data Warehouse จะทำการเพิ่ม Fact ขึ้นอีกหนึ่งแถว เพื่อบันทึกการเปลี่ยนแปลงเพศของฝูง 137 ในครั้งนี้ ซึ่งเมื่อฝูง 137 ถูกตอนแล้ว Valid Time ของมันจะเริ่มตั้งแต่วันที่ถูกตอน จนถึงตลอดไป ดังตาราง 3.4

ตาราง 3.3 Gender เมื่อยังไม่มีการเปลี่ยนแปลงข้อมูล

LOT_ID	Gender_ID	FROM_DATE	TO_DATE
137	C	1998-01-07	9999-12-31
219	C	1998-02-25	9999-12-31
374	B	1998-02-20	9999-12-31

ตาราง 3.4 Gender เมื่อข้อมูลมีการเปลี่ยนแปลง

LOT_ID	Gender_ID	FROM_DATE	TO_DATE
137	C	1998-01-07	1998-02-10
137	N	1998-02-10	9999-12-31
219	C	1998-02-25	9999-12-31
374	B	1998-02-20	9999-12-31

ผลที่ได้หลังจาก Dimension Data เป็น Temporal Data คือ สามารถติดตามข้อมูลในอดีตได้ทุกเวอร์ชั่นของข้อมูล ซึ่งผลลัพธ์ที่ได้คือข้อมูลที่เกิดขึ้นมีความแม่นยำขึ้นเพื่อนำมาใช้ในการวิเคราะห์ข้อมูล ซึ่งสามารถแก้ไขปัญหา Slowly Changing Dimension ได้

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองการสรุปข้อมูลใน Data Warehouse ที่ได้รวบรวมข้อมูลมาจาก Source

4.1.1 กรณีที่ 1 Source ที่มี Source เป็น Temporal Data แต่ Dimension Table เป็น Non-Temporal Data

ต่อไปนี้เป็นตารางข้อมูลที่สร้างขึ้นซึ่งจัดเก็บอยู่ใน Database เดียวกัน โดยเป็นข้อมูลเกี่ยวกับการดูแลคอกวัว โดยมีโครงสร้างตารางดังนี้

1) Lot เป็นตารางของฝูงวัวที่บอกชื่อ LOT สายพันธุ์ของวัว และเพศของฝูงวัว

ตาราง 4.1 Lot

LOT_ID	LOT_BREED	BREED_TYPE	GENDER_ID
137	Amerifax	Both	M
219	Danish Red	Both	F
374	Florida Cracker	Meat	F
400	Icelandic	Dairy	M
526	Siri	Dairy	F
678	Sanga	Meat	M

2) Pen เป็นตารางหมายเลขคอก แสดงชื่อคอกและตำแหน่งที่ตั้งเพื่อบอกที่อยู่ของวัวแต่ละ LOT

ตาราง 4.2 Pen

PEN_ID	PEN_NAME	PEN_LOCATION
1	Brig	46.317276, 8.010390
2	Naters	46.317647, 8.012633

3) Gender เป็นตารางแสดงรายละเอียดเพศของวัวแต่ละฝูง

ตาราง 4.3 Gender

GENDER_ID	GENDER
M	Male
F	Female

4) Lot_Tran เป็นตารางแสดงการดูแลจัดสรรของฝูงวัว อยู่ในคอกไหน เมื่อวันที่เท่าไหร่ และจำนวนเท่าใด

ตาราง 4.4 Lot_Tran

LOT_ID	PEN_ID	AMOUNT	FROM_DATE	TO_DATE
137	1	17	1998-01-07	1998-03-29
219	1	43	1998-02-25	1998-03-01
219	1	20	1998-03-01	1998-03-14
219	2	23	1998-03-01	1998-03-14
219	2	43	1998-03-14	9999-12-31
374	1	14	1998-02-20	9999-12-31
400	1	50	1998-04-11	1998-04-30
400	1	25	1998-04-30	9999-12-31
400	2	25	1998-04-30	9999-12-31
526	2	35	1998-04-14	1998-05-07
526	1	15	1998-05-07	1998-05-26
526	2	20	1998-05-07	9999-12-31
678	2	28	1998-03-13	1998-04-02
678	1	10	1998-04-02	1998-05-01
678	2	18	1998-04-02	1998-05-01
678	2	28	1998-05-01	9999-12-31

ทั้ง 4 ตารางข้อมูลข้างต้นจะอยู่ในฐานข้อมูลเดียวกันโดยมี Lot, Pen, Gender เป็น Non Temporal Database ส่วน Lot_Tran เป็น Temporal database แล้วทำการโหลดตารางทั้งหมดเข้ามายังคลังข้อมูลสำหรับใช้วิเคราะห์ข้อมูลตาม Star Schema

4.1.1.1 ตัวอย่างของ Fact table

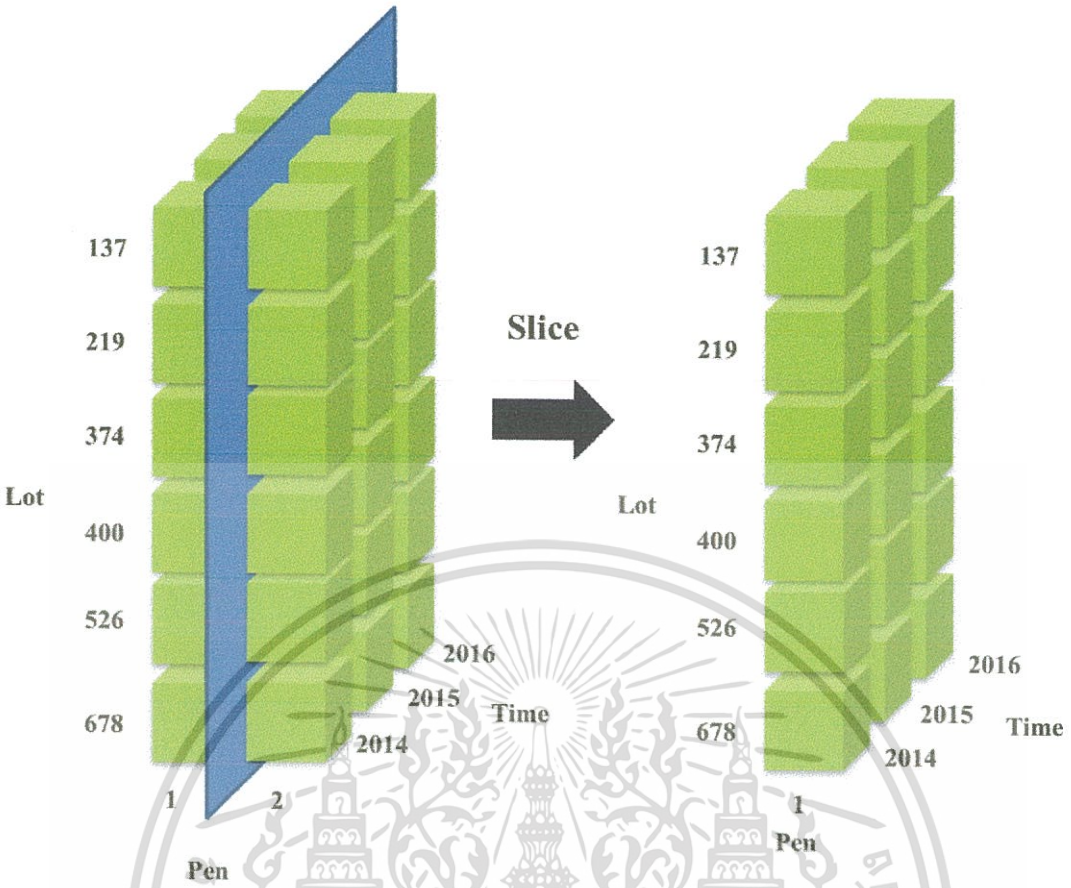
LOT_FACT เป็นตารางFact table ซึ่งแสดงการดูแลจัดสรรของฝูงวัว อยู่ในคอกไหน แต่ละฝูงเพศอะไร เมื่อวันที่เท่าไหร่และจำนวนเท่าใด โดยเกิดจาก primary key ของ Dimension ที่สนใจ

ตาราง 4.5 Fact table

LOT_ID	PEN_ID	GENDER_ID	TIMEKEY	AMOUNT
137	1	M	7	17
137	1	M	8	17
137	1	M	9	17
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
678	2	M	837	28

4.1.1.2 ตัวอย่างคำถาม

1) การทดสอบการใช้ตัวดำเนินการของคลังข้อมูลซึ่งคือ Slice กับคำถามที่ว่า "ในแต่ละปี มีวัวกี่ตัวในแต่ละฝูงที่เคยอยู่คอกที่ 1" ซึ่งฟังก์ชั่นจะทำการตัดเฉพาะส่วนที่สนใจ นั่นคือ Slice ที่ Dimension Pen เลือกเฉพาะฝูงวัวในคอกที่ 1 ของแต่ละปี ดังรูป 4.1 และผลลัพธ์ของคำตอบจะได้ตามรูป 4.2 ซึ่งจะแสดงจำนวนวัวของแต่ละฝูงที่อยู่ในคอกที่ 1 ของแต่ละปี



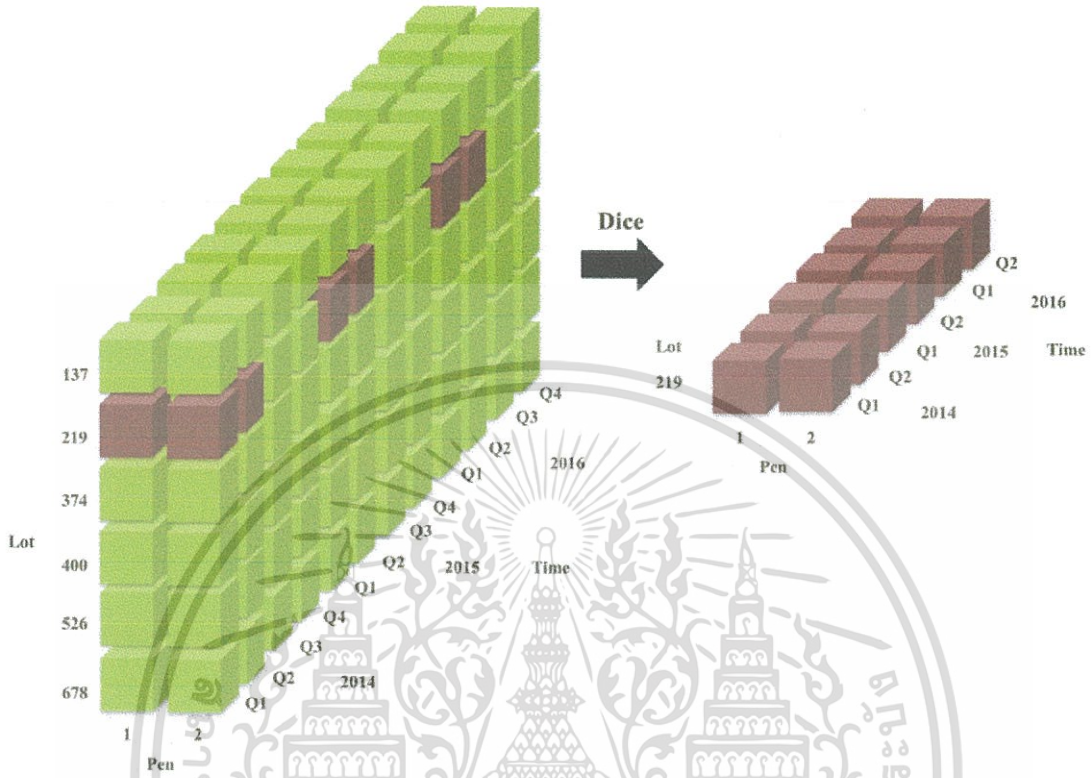
รูป 4.1 การ Slice ของคำถาม มีวัวกี่ตัวในแต่ละฝูงที่เคยอยู่คอกที่ 1

LOT_ID	PEN_ID	TYPETIME	MAX(FACT_TABLE.AMOUNT)
137	1	2014	17
219	1	2014	43
374	1	2014	14
374	1	2015	14
374	1	2016	14
400	1	2014	50
400	1	2015	25
400	1	2016	25
526	1	2014	15
678	1	2014	10

รูป 4.2 คำตอบของคำถาม มีวัวกี่ตัวในแต่ละฝูงที่เคยอยู่คอกที่ 1

2) การทดสอบการใช้ตัวดำเนินการของคลังข้อมูลซึ่งคือ Dice กับคำถามที่ว่า "ใน Q1 และ Q2 มีฝูง 219 อยู่ในแต่ละคอกเป็นจำนวนเท่าไหร่" ซึ่งฟังก์ชั่นจะทำการตัดเฉพาะส่วนที่สนใจ โดยเลือกส่วนที่สนใจมากกว่า 2 Dimension ขึ้นไป ทำให้มีความเฉพาะเจาะจงมากว่า โดยเลือกตัดที่ Dimension Lot และ Time เลือกเฉพาะวัวฝูง 219 ของ Q1 และ Q2 ดังรูป 4.3 และผลลัพธ์ของคำตอบจะ

ได้ตามรูป 4.4 ซึ่งจะแสดงจำนวนวัวของฝูง 219 ทั้ง 2 คอก ในช่วงเวลา Q1 และ Q2 ของแต่ละปี



รูป 4.3 การ Dice ของคำถาม ใน Q1 และ Q2 มีฝูง 219 อยู่ในแต่ละคอกเป็นจำนวนเท่าไหร่

LOT_ID	PEN_ID	YEAR	Q	AMOUNT
219	1	2014	1	43
219	2	2014	1	43
219	2	2014	2	43
219	2	2015	1	43
219	2	2015	2	43
219	2	2016	1	43
219	2	2016	2	43

รูป 4.4 คำตอบของคำถาม ใน Q1 และ Q2 มีฝูง 219 อยู่ในแต่ละคอกเป็นจำนวนเท่าไหร่

3) การทดสอบการใช้ตัวดำเนินการของคลังข้อมูลซึ่งคือ Roll up กับคำถามที่ว่า "จำนวนของฝูงวัว 219 ในแต่ละปี" ซึ่งฟังก์ชั่นจะทำการรวมกลุ่มข้อมูลระดับของเวลาตั้งแต่วันไปเป็นปี ผลลัพธ์ของคำตอบจะได้ตามรูป 4.5 ซึ่งจะแสดงจำนวนวัวของฝูง 219 ในระดับปี

NOV-2015	1	39
NOV-2015	2	116
NOV-2015		155
OCT-2014	1	39
OCT-2014	2	116
OCT-2014		155
OCT-2015	1	39
OCT-2015	2	116
OCT-2015		155
SEP-2014	1	39

MONTH_ID	PEN_ID	AMOUNT
SEP-2014	2	116
SEP-2014		155
SEP-2015	1	39
SEP-2015	2	116
SEP-2015		155
		4098

รูป 4.5 คำตอบของคำถาม จำนวนของฝูงวัว ในแต่ละปี 219

4.1.1.3 สรุปกรณีที่ 1 Source เป็น temporal data แต่ Dimension table เป็น Non-temporal data

ตาราง Lot_Fact เป็นตาราง Fact Table ที่มี attribute เป็น Foreign Key ที่มาจาก Primary Key ของ Dimension table ต่างๆ และ Measure ที่ผู้ใช้งานสนใจ ซึ่งผู้ใช้งานเลือกได้ว่าต้องการสรุปข้อมูล Measure จากการวิเคราะห์ Dimensionใด และระดับของเวลาแบบใด ซึ่งกำหนดระดับของเวลาที่เล็กที่สุดคือเป็นระดับวันหรือเลือกเป็นการเปลี่ยนแปลงข้อมูลช่วงเวลาเท่าใด นอกจากนี้ ยังสามารถใช้ตัวดำเนินการพื้นฐานของคลังข้อมูลในการตอบคำถามการคิวรี่ได้

4.1.2 กรณีที่ 2 Source ที่มี Source เป็น Temporal Data แต่ Dimension table เป็น Temporal Data

ข้อมูลตัวอย่างเป็นข้อมูลที่จัดเก็บอยู่ใน Database เดียวกัน ซึ่งเป็นข้อมูลเกี่ยวกับการดูแลคอกวัวโดยมีโครงสร้างตารางดังนี้

1) Gender เป็นตารางแสดงรายละเอียดการเปลี่ยนแปลงเพศของวัวแต่ละฝูง

ตาราง 4.6 Gender

LOT_ID	Gender_ID	FROM_DATE	TO_DATE
137	C	1998-01-07	1998-02-10
137	N	1998-02-10	9999-12-31
219	C	1998-02-25	9999-12-31
374	B	1998-02-20	9999-12-31
400	C	1998-04-11	9999-12-31
526	C	1998-04-14	1998-05-07
526	N	1998-05-07	9999-12-31
678	B	1998-03-13	1998-05-01
678	S	1998-05-01	9999-12-31

2) Pen เป็นตารางแสดงรายละเอียดการเปลี่ยนแปลงตำแหน่งของคอกวัว

ตาราง 4.7 Pen

PEN_ID	PEN_NAME	PEN_LOCATION	FROM_DATE	TO_DATE
1	Brig	A1	2014-01-07	9999-12-31
2	Naters	A2	2014-01-07	2014-05-01
2	Naters	A3	2014-05-01	9999-12-31

3) Lot_Tran เป็นตารางแสดงการดูแลจัดสรรของฝูงวัว อยู่ในคอกไหน เมื่อวันที่เท่าไหร่และจำนวนเท่าใด

ตาราง 4.8 Lot_Tran

LOT_ID	PEN_ID	AMOUNT	FROM_DATE	TO_DATE
137	1	17	2014-01-07	2014-03-29
219	1	43	2014-02-25	2014-03-01
219	1	20	2014-03-01	2014-03-14
219	2	23	2014-03-01	2014-03-14
219	2	43	2014-03-14	9999-12-31
374	1	14	2014-02-20	9999-12-31
400	1	50	2014-04-11	2014-04-30
400	1	25	2014-04-30	9999-12-31
400	2	25	2014-04-30	9999-12-31
526	2	35	2014-04-14	2014-05-07
526	1	15	2014-05-07	2014-05-26
526	2	20	2014-05-07	9999-12-31
678	2	28	2014-03-13	2014-04-02
678	1	10	2014-04-02	2014-05-01
678	2	18	2014-04-02	2014-05-01
678	2	28	2014-05-01	9999-12-31

ทั้ง 3 ตารางข้อมูลข้างต้นจะอยู่ในฐานข้อมูลเดียวกันโดยมี Lot_Tran, Pen, Gender เป็น Temporal Database แล้วทำการโหลดตารางทั้งหมดเข้ามายังคลังข้อมูลสำหรับใช้วิเคราะห์ข้อมูลตาม Star Schema

4.1.2.1 ตัวอย่างของ Fact table

LOT_FACT เป็นตาราง Fact table ซึ่งแสดงการดูแลจัดสรรของฝูงวัว อยู่ในคอกไหน แต่ละฝูงเพศอะไร เมื่อวันที่เท่าไหร่และจำนวนเท่าใด โดยเกิดจาก primary key ของ Dimension ที่สนใจ

ตาราง 4.9 Fact table

LOT_ID	PEN_ID	AMOUNT	TIMEKEY	WM_VALID	
				From Date	To Date
137	1	17	7	2014-01-07	2014-01-08
137	1	17	8	2014-01-08	2014-01-09
137	1	17	9	2014-01-09	2014-01-10
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
678	2	28	837	2014-04-16	2014-04-17

4.1.2.2 ตัวอย่างคำถาม

1) **จำนวนเพศวัวในแต่ละคอก** จากคำถาม เพื่อต้องการทราบจำนวนเพศวัวในแต่ละคอก โดยจะทำการใช้ฟังก์ชั่น WM_OVERLAPS เพื่อเลือกช่วงเวลาระหว่าง Fact Table และ Lot Dimension เพื่อให้ทราบว่า จำนวนเพศวัวในแต่ละคอกมีเท่าไหร่ โดยผลลัพธ์จำนวนเพศวัวในแต่ละคอกมีเท่าไหร่ ณ วันที่ 06/05/2014 จะแสดงในรูป 4.6 และจำนวนเพศวัวในแต่ละคอกมีเท่าไหร่ ณ วันที่ 07/05/2014 จะแสดงในรูป 4.7 ซึ่งจะเห็นว่าจำนวนของเพศมีการเปลี่ยนแปลงไปในวันที่ 7 (มีเพศ N เพิ่มขึ้นมา) เนื่องจากวัวบางฝูงได้ถูกตอนแล้วในวันที่ 7

GEN	PEN	SUM(A.AMOUNT)
B	1	14
B		14
C	1	25
C	2	103
C		128
S	2	28
S		28
		170

รูป 4.6 คำตอบของคำถาม จำนวนเพศวัวในแต่ละคอก ณ วันที่ 06/05/2014

GEN	PEN	SUM(A.AMOUNT)
B	1	14
B		14
C	1	25
C	2	68
C		93
N	1	15
N	2	20
N		35
S	2	28
S		28
		170

รูป 4.7 คำตอบของคำถาม จำนวนเพศวัวในแต่ละคอก ณ วันที่ 07/05/2014

2) จำนวนวัว และตำแหน่งของวัวในแต่ละฝูง จากคำถาม เพื่อต้องการทราบจำนวนวัว และตำแหน่งของวัวในแต่ละฝูง โดยจะทำการใช้ฟังก์ชั่น WM_OVERLAPS เพื่อเลือกช่วงเวลา ระหว่าง Fact Table และ Pen Dimension โดยผลลัพธ์ของจำนวนวัว และตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 01/05/2015 จะแสดงในรูป 4.8 และผลลัพธ์ของจำนวนวัว และตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 30/04/2015 จะแสดงในรูป 4.9 ซึ่งจะเห็นว่ามีบางส่วนของวัวฝูง 678 ได้มีการย้ายคอกไป

LOT	PEN	LOCA	AMOUNT	FROM_DATE	TO_DATE
219	2	A3	43	2014-05-01	2014-05-02
374	1	A1	14	2014-05-01	2014-05-02
400	1	A1	25	2014-05-01	2014-05-02
400	2	A3	25	2014-05-01	2014-05-02
526	2	A3	35	2014-05-01	2014-05-02
678	2	A3	28	2014-05-01	2014-05-02

รูป 4.8 คำตอบของคำถาม จำนวนวัวและตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 01/05/2015

LOT	PEN	LOCA	AMOUNT	FROM_DATE	TO_DATE
219	2	A3	43	2014-04-30	2014-05-01
374	1	A1	14	2014-04-30	2014-05-01
400	1	A1	25	2014-04-30	2014-05-01
400	2	A3	25	2014-04-30	2014-05-01
526	2	A3	35	2014-04-30	2014-05-01
678	1	A1	10	2014-04-30	2014-05-01
678	2	A3	18	2014-04-30	2014-05-01

รูป 4.9 คำตอบของคำถาม จำนวนวัวและตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 30/04/2015

4.1.2.3 สรุปกรณีที่ 2 Source เป็น temporal data แต่ Dimension table เป็น temporal data

ตาราง Lot_Fact เป็นตาราง Fact Table ที่มี attribute เป็น Foreign Key ที่มาจาก Primary Key ของ Dimension table ต่างๆ และ Measure ที่ผู้ใช้งานสนใจ ซึ่งผู้ใช้งานเลือกได้ว่าต้องการสรุปข้อมูล Measure จากการวิเคราะห์ Dimension ใด เปลี่ยนแปลงข้อมูลช่วงเวลาเท่าใด และเนื่องจากกำหนดให้คลังข้อมูลอยู่ในรูปแบบของ Temporal data ดังนั้น Fact Table และ Dimension Table จะมีคอลัมน์ของช่วงเวลา เก็บช่วงเวลาที่ Fact เป็นจริง ซึ่งเมื่อต้องการคิวรี่ข้อมูลต่างๆ จะมีตัวดำเนินการของ OWM ช่วยสำหรับการตอบคำถามของการคิวรี่ข้อมูลเชิงเวลา

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุปของโครงงาน

Oracle Workspace Manager รองรับการทำงานของข้อมูลเชิงเวลาเพื่อมีการเก็บบันทึกข้อมูลที่มีค่าความจริงเป็นเท็จ ทำให้สามารถทราบข้อมูลที่มีค่าความจริงในอดีต ดังนั้นหากมีการประยุกต์ใช้งานกับคลังข้อมูลจะทำให้คลังข้อมูลสามารถนำข้อมูลเชิงเวลาที่มีค่าความจริงในอดีตมาใช้ในการวิเคราะห์ตอบคำถามได้ ทำให้ผลลัพธ์ที่ได้มีความถูกต้อง เนื่องจากในขั้นตอนการวิเคราะห์จะใช้ข้อมูลประกอบกับเวลาที่ข้อมูลนั้นเป็นจริง และจากการออกแบบคลังข้อมูลให้อยู่บนระบบเดียวกันกับฐานข้อมูลเชิงเวลา ซึ่งในอดีตเดิมต้องมีกระบวนการ ETL ก็เป็นการลดความซับซ้อนของกระบวนการนี้ได้ เพราะในขั้นตอนวิเคราะห์จะเป็นการสร้างแบบจำลองของคลังข้อมูลขึ้นมาจากฐานข้อมูลเชิงเวลา และนอกจากนี้ในอดีตเมื่อมีการสร้างคลังข้อมูลจะเกิดปัญหาทั่วไปของข้อมูลที่พบได้ในคลังข้อมูลนั่นคือ slowly changing dimensions แต่หากทำเป็น คลังข้อมูลเชิงเวลาจะเป็นการแก้ไขปัญหานี้ เนื่องจากหากมีการเปลี่ยนแปลงข้อมูลภายใน Dimension จะมีการบันทึกช่วงเวลาที่เป็นจริงของข้อมูลนั้น ซึ่งทำให้ผลลัพธ์ที่ได้เป็นการแก้ไขปัญหาที่พบขึ้นอยู่บ่อยครั้งในคลังข้อมูลทั่วไป

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข

1) เนื่องจาก Oracle Workspace Manager สนับสนุนฐานข้อมูลเชิงเวลาและมีหลากหลายฟังก์ชั่นที่รองรับการใช้งาน เพื่อจะให้คลังข้อมูลสามารถใช้งานฟังก์ชั่นและตอบคำถามได้อย่างถูกต้อง จึงต้องทำการศึกษาว่าฟังก์ชั่นของ Oracle Workspace Manager มีรูปแบบการทำงานอย่างไร และการทำงานของฟังก์ชั่นของฐานข้อมูลเชิงเวลาเป็นไปตามทฤษฎีของ Richard T. Snodgrass หรือไม่

2) การสร้างฐานข้อมูลและสภาพแวดล้อมของฐานข้อมูลสำหรับการศึกษาและทดลองโครงงานนี้ค่อนข้างยาก ในตอนต้น จึงมีแนวทางป้องกันข้อผิดพลาดโดยจำลองสภาพแวดล้อมทั้งหมดไว้ในโปรแกรม VM Ware Workstation 11

3) ข้อมูลที่ไม่เป็นจริงในปัจจุบัน (Valid Time เป็นอดีต) ไม่แสดงผล ดังนั้นต้อง Execute ฟังก์ชั่นของ Oracle ที่รองรับ Valid time ทุกครั้ง เมื่อคิวรี่ข้อมูล

4) Library ของ php สำหรับส่งค่าไปยังฟังก์ชั่นใน Oracle ส่งค่าได้ ไม่เกิน 5 Parameter ดังนั้นเราต้องมีการประยุกต์ฟังก์ชั่นของเราใหม่ก่อนส่งค่าเข้าไป

5) ฟังก์ชั่นการใช้งานของ Oracle Workspace Manager ไม่สามารถเรียกใช้งานโดยตรงผ่านทาง php ได้ เนื่องจาก php ไม่สนับสนับฟังก์ชั่นเชิงเวลาของ Oracle ดังนั้นเราจึงจำเป็นต้องสร้างฟังก์ชั่นฝังไว้ภายในใน OWM ก่อน

5.3 แนวทางในการพัฒนาต่อ

มีแนวทางในการพัฒนาและศึกษาต่อดังนี้

1) ศึกษาคลังข้อมูลและออกแบบฟังก์ชั่นการใช้งานสำหรับการคิวรี่ข้อมูล เพื่อเพิ่มความสามารถของการถามคำถามสำหรับผู้ใช้งาน ให้ใช้งานได้ง่ายและตอบคำถามได้อย่างครอบคลุมมากยิ่งขึ้น
2) พัฒนาการความสามารถการตอบคำถามของข้อมูลที่ต้องมีการคิดช่วงเวลาใช้ในการคำนวณ ซึ่งอาจมีกระบวนการคิดที่ซับซ้อน
3) ศึกษาการประยุกต์ใช้คลังข้อมูลเชิงเวลากับรูปแบบอื่นๆของคลังข้อมูลเพิ่มเติมนอกจากรูปแบบที่ใช้ในการทดลองข้างต้น

เอกสารอ้างอิง

[1] Richard T. Snodgrass. September 3, 1998. **Managing Temporal Data. A Five-Part Series.** TR-28. A TIMECENTER Technical Report

[2] Kulkarni, G. K. and Michels, J. E. 2012. **Temporal features in SQL: 2011.** ACM SIGMOD Record. 41(3): 34-43 (2012)

[3] Allen, James F. **Maintaining knowledge about temporal intervals.** Communications of the ACM 26(11) pp.832-843, Nov. 1983.

[4] Bill Beauregard, Chuck Murray and Ben Speckhard. 2009. **Oracle Database Workspace Manager Developer's Guide 11g Release 1 . U.S.A: Oracle Corporation.** [Online]. Available : http://docs.oracle.com/cd/B28359_01/appdev.111/b28396/long_intro.htm

[5] Oracle Help Center. 2014. **Oracle Database Online Documentation 12c Release 1 (12.1). Data Warehousing and Business Intelligence. 1 Data Warehousing Concepts. .** Available : https://docs.oracle.com/cd/E11882_01/server.112/e25554/concept.htm#DWHSG8063

[6] กิตติพงษ์ กลมกล่อม. 2552. **การออกแบบและพัฒนาคลังข้อมูล (Data Warehouse).** กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์

[7] Christian S. Jensen, TorbenBachPedersen, Christian Thomsen. 2010. **MultidimensionalDatabasesand DataWarehousing.** Morgan & Claypool Publishers series.

ภาคผนวก

โค้ดโปรแกรมที่สำคัญ

1 โค้ดของการทดลองชุดที่ 1

การทดลองชุดที่ 1 คือ Data Warehouse มี Dimension Table เป็น Non-Temporal Data

1.1 การสร้าง Time Dimension

โค้ด SQL ที่ใช้ในการสร้าง Time Dimension นี้จะใช้ทั้ง 2 การทดลอง โดย Time Dimension ที่ใช้ใน Temporal Data Warehouse จะมีคอลัมน์ดังต่อไปนี้ TIMEKEY, DAY_ID, WEEK_DAY_SHORT, MONTH_ID, DAY_NUM_OF_WEEK, DAY_NUM_OF_MONTH, DAY_NUM_OF_YEAR, MONTH_END_DATE, MONTH_SHORT_DESC, MONTH_SHORT, MONTH_NUM_OF_YEAR, QUARTER_ID, YEAR_ID, QUARTER_END_DATE, QUARTER_NUM_OF_YEAR, YEAR_TIME_SPAN และ YEAR_END_DATE ซึ่ง TIMEKEY จะเป็น Surrogate Key ที่เรียงเป็น Sequence แทนวันในหนึ่งวัน กล่าวคือ แต่ละวันก็จะมี Time Key ที่ต่างกัน โดยโค้ดที่ใช้คือ โปรแกรม 1

โปรแกรม 1 การสร้าง Time Dimension

```
CREATE TABLE time_calendar_dim AS
WITH base_calendar AS
  (SELECT CurrDate              AS Day_ID,
    TO_CHAR(CurrDate,'DY')      AS Week_Day_Short,
    TO_NUMBER(TRIM(leading '0'
  FROM TO_CHAR(CurrDate,'D')))  AS Day_Num_of_Week,
    TO_NUMBER(TRIM(leading '0'
  FROM TO_CHAR(CurrDate,'DD')))  AS Day_Num_of_Month,
    TO_NUMBER(TRIM(leading '0'
  FROM TO_CHAR(CurrDate,'DDD')))  AS Day_Num_of_Year,
    UPPER(TO_CHAR(CurrDate,'Mon')
    || '-'
    || TO_CHAR(CurrDate,'YYYY'))  AS Month_ID,
    TO_CHAR(CurrDate,'Mon')
    || ' '
    || TO_CHAR(CurrDate,'YYYY')  AS Month_Short_Desc,

    TO_CHAR(CurrDate,'Mon')      AS Month_Short,
       TO_NUMBER(TRIM(leading '0'
  FROM TO_CHAR(CurrDate,'MM')))  AS Month_Num_of_Year,
```

โปรแกรม 1 การสร้าง Time Dimension (ต่อ)

```
    'Q'
    || UPPER(TO_CHAR(CurrDate,'Q')
    || '-'
    || TO_CHAR(CurrDate,'YYYY'))      AS Quarter_ID,
    TO_NUMBER(TO_CHAR(CurrDate,'Q')) AS
Quarter_Num_of_Year,


    TO_CHAR(CurrDate,'YYYY') AS Year_ID
  FROM
    (SELECT level n,
      -- Calendar starts at the day after this date.
      TO_DATE('31/12/2010','DD/MM/YYYY') +
NUMTODSINTERVAL(level,'DAY') CurrDate
    FROM dual
      -- Change for the number of days to be added to
the table.
      CONNECT BY level <= 1100
    )
  )
SELECT day_id,
  week_day_short,
  day_num_of_week,
  day_num_of_month,
  day_num_of_year,
  month_id,
  MAX(day_id) OVER (PARTITION BY month_id) AS
Month_End_Date,
  month_short_desc,
  month_short,
  month_num_of_year,
  quarter_id,
  MAX(day_id) OVER (PARTITION BY quarter_id) AS
Quarter_End_Date,
  quarter_num_of_year,
  year_id,
  COUNT(*) OVER (PARTITION BY year_id)    AS
Year_Time_Span,
  MAX(day_id) OVER (PARTITION BY year_id) AS
Year_End_Date
FROM base_calendar
ORDER BY day_id;
--
COMMIT;
```

1.2 การสร้าง และการนำข้อมูลจาก Source ไปยัง Gender Dimension

เราสามารถเลือกเฉพาะคอลัมน์ที่เราสนใจเข้าสู่ Data Warehouse ได้ แต่เนื่องจาก Source ในส่วนนี้ มีคอลัมน์เหมือนกับ Gender Dimension ดังนั้นเราจะเลือกทุกคอลัมน์ของ Source มาใส่ใน Gender Dimension

โปรแกรม 2 Gender Dimension

```
CREATE TABLE Genders
  ( gender_id number(3) not null,
  gender VARCHAR2(6 BYTE) not null,
  CONSTRAINT genders PRIMARY KEY (gender_id)
  )

INSERT
  INTO Genders
    (gender_id, gender)
  SELECT g_ID, g_gender FROM gender ;
END;
```

1.3 การสร้าง และการนำข้อมูลจาก Source ไปยัง Lot Dimension

Source ในส่วนนี้ ก็มีหน้าตาคอลัมน์เหมือนกับ Lot Dimension ดังนั้นเราจะเลือกทุกคอลัมน์ของ Source มาใส่ใน Lot Dimension

โปรแกรม 3 Lot Dimension

```
CREATE TABLE LOTS
  (LOT_ID number(3) not null,
  LOT_BREED VARCHAR2(15 BYTE),
  BREED_TYPE VARCHAR2(5 BYTE),
  GENDER_ID VARCHAR2(1 BYTE)
  CONSTRAINT LOTS PRIMARY KEY (LOT_ID)
  )

INSERT
  INTO LOTS
    (LOT_ID, LOT_BREED,BREED_TYPE, GENDER_ID)
  SELECT l_ID, l_BREED,l_TYPE, LG_ID FROM LOT ;
END;
```

1.4 การสร้าง และการนำข้อมูลจาก Source ไปยัง Pen Dimension

Source ในส่วนนี้ ก็มีหน้าตาคอลัมน์เหมือนกับ Pen Dimension ดังนั้นเราจะเลือกทุกคอลัมน์ของ Source มาใส่ใน Pen Dimension

โปรแกรม 4 Pen Dimension

```
CREATE TABLE PENS
  (PEN_ID number(3) not null,
  PEN_NAME VARCHAR2(6 BYTE),
  PEN_LOCATION VARCHAR2(19 BYTE),
  CONSTRAINT PENS PRIMARY KEY (PEN_ID)
  )

INSERT
  INTO PENS
    (PEN_ID,PEN_NAME,PEN_LOCATION)
  SELECT P_ID,P_NAME,P_LOCATION FROM PEN ;
END;
```

1.5 การสร้าง Fact Table

จริงๆ แล้วการสร้าง Table ต่างๆ สามารถใช้ GUI ของโปรแกรม Oracle SQL Developer สร้างได้ ในการสร้าง Fact Table ซึ่งจะประกอบไปด้วย Primary Key ของแต่ละ Dimension

โปรแกรม 5 การสร้าง Fact Table

```
CREATE TABLE fact_tb
  ( lot_id number(3) not null,
  pen_id number(3) not null,
  timekey number(4) not null,
  time_date date,
  amount number(3) not null,
  CONSTRAINT fact_pk PRIMARY KEY (
lot_id,pen_id,timekey)
  )
```

1.6 การแปลงข้อมูลจาก Source ไปยัง Fact Table

ในการแปลงข้อมูลจาก Source เข้าสู่ Data Warehouse ในส่วนนี้ คือ Source จาก ตาราง pen_transections ที่บันทึกการเปลี่ยนแปลงของวัวแต่ละฝูงว่าอยู่ที่คอกไหน เมื่อไหร่ เนื่องจากเป็น Temporal Data Warehouse การจะ Insert ข้อมูลที่เป็น Temporal Data กล่าวคือ เป็นข้อมูลที่มี Valid Time เข้าสู่ Fact Table นั้น มีรูปแบบที่เฉพาะตัว โดย Valid Time ใน Fact Table นั้น ได้ Extract มาจาก Valid Time ของ Source ซึ่งเป็นการ Extract จาก Period Time ไปเป็น Point of Time กล่าวคือ Valid Time ใน Source เป็นช่วงเวลา แต่ Valid Time ใน Fact Table จะเป็นแค่เวลาวันเดียว โดยแทนวันดังกล่าวด้วย Time Key ซึ่งโค้ดที่ใช้ในการ Insert Data เข้า Fact Table ได้ ดังโปรแกรม 6

โปรแกรม 6 การแปลงข้อมูลจาก Source ไปยัง Fact Table

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
DECLARE
  ncount NUMBER:=0;
  F      DATE;
  T      DATE;
  D      DATE;
BEGIN
  FOR tran_rec IN
  (SELECT * FROM pen_transitions
  )
  LOOP
    BEGIN
      F
:=to_date(TO_CHAR(tran_rec.wm_valid.validfrom,
'DD/MM/YYYY'),'DD/MM/YYYY');
      T   :=
to_date(TO_CHAR(tran_rec.wm_valid.validtill,
'DD/MM/YYYY'),'DD/MM/YYYY');
      D   :=F;
      IF T IS NULL THEN
        T :=sysdate;
      END IF;
     -- DBMS_OUTPUT.PUT_LINE('T'||T);
      --Loop for insert
      WHILE D<T
      LOOP
        INSERT INTO fact_tb
          ( lot_id,pen_id,timekey,time_date,amount
          )
        SELECT tran_rec.lot_id,
          tran_rec.pen_id,
          time_dimension.timekey,
          time_dimension.day_id,
          tran_rec.amount
        FROM time_dimension
        WHERE day_id = D;
        D           :=D+1;
      END LOOP;
      COMMIT;
    END;
  END LOOP;
END;
```

2 โค้ดของการทดลองชุดที่ 2

การทดลองชุดที่ 2 คือ Data Warehouse มี Dimension Table เป็น Temporal Data

2.1 การสร้าง และการนำข้อมูลจาก Source ไปยัง Lot Dimension

Dimension ส่วนนี้ ได้มีการนำเข้ามาจาก Source Gender_trans ที่บันทึกว่าวัวฝูงไหนถูกตอนเมื่อไหร่บ้าง ซึ่งในการทดลองที่ 2 นี้ Dimension จะเป็น Temporal Data ดังนั้นจะมีคอลัมน์ Valid Time โดยสามารถนำ Valid Time ที่อยู่ใน Source มาได้เลย

โปรแกรม 7 Lot Dimension

```
CREATE TABLE LOTS
  (LOT_ID number(3) not null,
  GENDER_ID VARCHAR2(5 BYTE),
  CONSTRAINT LOTS PRIMARY KEY (LOT_ID)
  )

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

EXECUTE DBMS_WM.SetValidTime ( DBMS_WM.MIN_TIME,
DBMS_WM.MAX_TIME );

INSERT
  INTO LOTS
    (LOT_ID, GENDER_ID, wm_valid)
  SELECT l_ID, LG_ID, wm_valid FROM GENDER_TRANS ;
END;
```

2.2 การสร้าง และการนำข้อมูลจาก Source ไปยัง Pen Dimension

Dimension ส่วนนี้ ก็ได้มีการนำเข้ามาจาก Source Pen ที่บันทึกว่าคอกนี้เคยตั้งอยู่ที่ใดบ้าง ซึ่งในการทดลองที่ 2 นี้ Dimension จะเป็น Temporal Data ดังนั้นจะมีคอลัมน์ Valid Time โดยสามารถนำ Valid Time ที่อยู่ใน Source มาได้เลย

โปรแกรม 8 Pen Dimension

```
CREATE TABLE PENS
  (PEN_ID number(3) not null,
  PEN_LOCATION VARCHAR2(20 BYTE),
  CONSTRAINT PENS PRIMARY KEY (PEN_ID)
  )

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

EXECUTE DBMS_WM.SetValidTime ( DBMS_WM.MIN_TIME,
DBMS_WM.MAX_TIME );

INSERT
  INTO PENS
    (PEN_ID,PEN_LOCATION, wm_valid))
  SELECT P_ID,P_LOCATION, wm_valid) FROM PEN ;
END;
```

2.3 การสร้าง Fact Table

จริงๆ แล้วการสร้าง Table ต่างๆ สามารถใช้ GUI ของโปรแกรม Oracle SQL Developer สร้างได้ ในการสร้าง Fact Table ซึ่งจะประกอบไปด้วย Primary Key ของแต่ละ Dimension

โปรแกรม 9 การสร้าง Fact Table

```
CREATE TABLE fact_table
  (
    lot_id  NUMBER(3) NOT NULL,
    pen_id  NUMBER(3) NOT NULL,
    amount  NUMBER(3) NOT NULL,
    timekey NUMBER(4) NOT NULL,
    CONSTRAINT fact_pk PRIMARY KEY (
lot_id,pen_id,timekey)
  )

EXECUTE
DBMS_WM.EnableVersioning('fact_table','VIEW_WO_OVERWRIT
E',FALSE, TRUE);
```

2.4 การแปลงข้อมูลจาก Source ไปยัง Fact Table

Fact Table ของการทดลองที่ 2 คล้ายกับการทดลองที่ 1 แต่ต่างกันตรงที่การทดลองที่ 2 จะเพิ่มคอลัมน์ของ Point of Time ในรูปของ Period กล่าวคือ จะแสดงเป็น Valid Time แต่เป็น Valid Time ของวันหนึ่งวัน ซึ่งโค้ดที่ใช้ในการ Insert Data เข้า Fact Table ได้ ดังโปรแกรม 10

โปรแกรม 10 การแปลงข้อมูลจาก Source ไปยัง Fact Table

```
  EXECUTE DBMS_WM.SetValidTime
  (
    DBMS_WM.MIN_TIME,
    DBMS_WM.MAX_TIME
  );
DECLARE
  ncount NUMBER:=0;
  F      DATE;
  T      DATE;
  D      DATE;
BEGIN
  FOR tran_rec IN
  (SELECT * FROM USERTEST.amount_transition
  )
  LOOP
    BEGIN
      F
:=to_date(TO_CHAR(tran_rec.wm_valid.validfrom,
'DD/MM/YYYY'),'DD/MM/YYYY');
      T      :=
to_date(TO_CHAR(tran_rec.wm_valid.validtill,
'DD/MM/YYYY'),'DD/MM/YYYY');
      D    :=F;
      IF T IS NULL THEN
        T  :=sysdate;
      END IF;
      -- DBMS_OUTPUT.PUT_LINE('T'||T);
      --Loop for insert
      WHILE D<T
      LOOP
        INSERT INTO fact_table
          ( lot_id,pen_id,amount,timekey,wm_valid
          )
        SELECT tran_rec.lot_id,
          tran_rec.pen_id,
          tran_rec.amount,
          time_dim.timekey,
```

โปรแกรม 10 การแปลงข้อมูลจาก Source ไปยัง Fact Table (ต่อ)

```
    WMSYS.WM_PERIOD(time_dim.day_id,time_dim.day_id+1)
          FROM time_dim
          WHERE day_id = D;
          D            :=D+1;
       END LOOP;
       COMMIT;
     END;
  END LOOP;
END;
```

3 โค้ดการใช้ในการตอบคำถามจาก Data Warehouse

3.1 การ Roll up

จากคำถามที่ว่า "จำนวนของฝูงวัว 219 ในแต่ละปี"

โปรแกรม 11 การ Roll up

```
SELECT month_id,
  pen_id,
  SUM(COUNT)AS amount
FROM
  (SELECT f.LOT_ID,
    f.PEN_ID,
    T.MONTH_ID,
    MIN(f.amount) AS COUNT
  FROM fact_table f,
    TIME_DIMENSION t
  WHERE f.TIMEKEY=t.TIMEKEY
  GROUP BY (f.LOT_ID,f.PEN_ID,T.MONTH_ID)
  )
GROUP BY rollup(month_id,pen_id);
```

3.2 การ Slice

จากคำถามที่ว่า "ในแต่ละปี มีวัวกี่ตัวในแต่ละฝูงที่เคยอยู่คอกที่ 1"

โปรแกรม 12 การ Slice

```
select  fact_table.lot_id, fact_table.pen_id,year_ID,
Max(fact_table.amount)
from time_dimension, fact_table
where fact_table.timekey = time_dimension.timekey and
fact_table.PEN_ID = 1
group by fact_table.lot_id, fact_table.pen_id, year_ID
order by fact_table.lot_id, fact_table.pen_id, year_ID;
```

3.3 การ Dice

จากคำถามที่ว่า "ใน Q1 และ Q2 มีฝุ่ง 219 อยู่ในแต่ละคอกเป็นจำนวนเท่าไหร่"

โปรแกรม 13 การ Dice

```
select fact_table.lot_id, fact_table.pen_id,year_id,
QUARTER_NUM_OF_YEAR as q, max(fact_table.amount) as
Amount
from time_dimension, fact_table
where (fact_table.timekey = time_dimension.timekey) and
(QUARTER_NUM_OF_YEAR=1 or QUARTER_NUM_OF_YEAR=2) and
fact_table.LOT_ID = 219
group by fact_table.lot_id, fact_table.pen_id, year_ID,
QUARTER_NUM_OF_YEAR
order by fact_table.lot_id, fact_table.pen_id, year_ID,
QUARTER_NUM_OF_YEAR;
```

3.4 คำถามที่เกี่ยวข้องกับ Slowly Changing Dimension

1) จากการทดลองที่ 2 ในบทที่ 4 ที่ Dimension เป็น Temporal Data จากคำถามที่ว่า "จำนวนเพศวัวในแต่ละคอก ณ วันที่ 06/05/2014" สามารถ Query ได้ตามโปรแกรม ก.14 โดยการตรวจเวลาระหว่าง Fact Table และ Dimension Table นั้น จะใช้ฟังก์ชั่น WM_OVERLAPS ซึ่งเป็นฟังก์ชั่นที่ตรวจสอบเวลาสองช่วงเวลาที่ซ้อนทับกัน ทำให้ทราบว่า จำนวนเพศวัวในแต่ละคอกมีเท่าไหร่ ณ วันที่นั้นๆ ซึ่งถ้าอยากรู้จำนวนเพศของวัน ณ วันไหน ก็เปลี่ยนวันที่ได้ในโค้ด

โปรแกรม 14 หาจำนวนเพศวัวในแต่ละคอก ณ วันที่ 06/05/2014

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
    SELECT a.gen, a.pen, SUM(a.amount)
    FROM
      (SELECT e1.lot_id AS lot,
        e1.pen_id       AS pen,
        e2.gender_id    AS gen ,
        e1.amount       AS amount,
        TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,
        TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
      FROM fact_table e1, lot_dim e2 ,time_dim e3
      WHERE e1.lot_id = e2.lot_id
      AND e1.timekey  = e3.timekey
      AND e3.day_id   = TO_DATE ('06/05/2014',
'dd/mm/yyyy')
      AND WM_OVERLAPS(e1.WM_VALID,e2.WM_VALID)=1
      ORDER BY e1.lot_id, e1.timekey, e1.pen_id
      ) a
    GROUP BY  rollup(a.gen, a.pen);
```

2) จากคำถามที่ว่า "จำนวนวัวและตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 01/05/2015" ซึ่งจะสามารถ Query ได้ตามโปรแกรม ก.15 โดยการตรวจเวลาระหว่าง Fact Table และ Dimension Table นั้น ทำการใช้ฟังก์ชั่น WM_OVERLAPS เพื่อเลือกช่วงเวลา ระหว่าง Fact Table และ Pen Dimension ทำให้ทราบได้ว่าตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 01/05/2015 มีจำนวนเท่าไหร่ ซึ่งถ้าอยากทราบวันอื่น ก็แก้ไขวันที่ที่อยากทราบได้ในโค้ด

โปรแกรม 15 หาจำนวนวัวและตำแหน่งของวัวในแต่ละฝูง ณ วันที่ 01/05/2015

```
EXECUTE
DBMS_WM.SetValidTime(DBMS_WM.MIN_TIME,DBMS_WM.MAX_TIME)
;
      SELECT e1.lot_id AS lot,
        e1.pen_id        AS pen,
        e2.pen_location     AS Loca ,
        e1.amount        AS amount,
        TO_CHAR(e1.WM_VALID.VALIDFROM,'YYYY-MM-DD')
FROM_DATE,
        TO_CHAR(e1.WM_VALID.VALIDTILL,'YYYY-MM-DD')
TO_DATE
      FROM fact_table e1, pen_dim e2 , time_dim e3
      WHERE e1.pen_id = e2.pen_id
      AND e1.timekey  = e3.timekey
      AND e3.day_id   = TO_DATE ('01/05/2014',
'dd/mm/yyyy')
      AND WM_OVERLAPS(e1.WM_VALID,e2.WM_VALID)=1
      ORDER BY e1.lot_id, e1.pen_id;
```