การประยุกต์ฐานข้อมูลเชิงเวลาโดยใช้ภาษาเอสคิวแอลเชิงเวลา 2011

A TEMPORAL DATABASE APPLICATION USING TEMPORAL SQL 2011

ธนัตถ์ พีระไพศาลทรัพย์

พิสิฐ ศรีพรสวัสดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

การประยุกต์ฐานข้อมูลเชิงเวลาโดยใช้ภาษาเอสคิวแอลเชิงเวลา 2011

A TEMPORAL DATABASE APPLICATION USING TEMPORAL SQL 2011

ธนัตถ์ พีระไพศาลทรัพย์
พิสิฐ ศรีพรสวัสดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

ปริญญานิพนธ์ปีการศึกษา 2559

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ฐานข้อมูลเชิงเวลาโดยใช้ภาษาเอสคิวแอลเชิงเวลา 2011

A TEMPORAL DATABASE APPLICATION USING TEMPORAL SQL 2011

ผู้จัดทำ

1. ธนัตถ์ พีระไพศาลทรัพย์ รหัสนักศึกษา 56010564
2. พิสิฐ ศรีพรสวัสดิ์ รหัสนักศึกษา 56010873

อาจารย์ที่ปรึกษา

(รศ. ดร. ศุภมิตร จิตตะยโศธร)

การประยุกต์ฐานข้อมูลเชิงเวลา
โดยใช้ภาษาเอสคิวแอลเชิงเวลา 2011

นายธนัตถ์ พีระไพศาลทรัพย์ 56010564
นายพิสิฐ ศรีพรสวัสดิ์ 56010873
รศ. ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2559

บทคัดย่อ

โครงงานนี้มีจุดมุ่งหมายในการศึกษาขีดความสามารถในการจัดการฐานข้อมูลเชิงเวลาของระบบจัดการฐานข้อมูลเชิงพาณิชย์ที่มีภาษาเอสคิวแอลสอดคล้องกับมาตรฐานเอสคิวแอล๒๐๑๑ นักศึกษาจะศึกษาแนวคิดและหลักการของฐานข้อมูลเชิงเวลา ทำการทดสอบฐานข้อมูลเชิงเวลาโดยใช้ระบบจัดการฐานข้อมูลเชิงพานิชย์และพัฒนาโปรแกรมประยุกต์เชิงเวลาที่มีนัยสำคัญทางเทคนิค

A Temporal Database Application

using Temporal SQL 2011

Mr. Tanat Perapaisarnsub 56010564

Mr. Phisit Siphonsawat 56010873

Assoc. Prof. Dr. Suphamit Chittayasothorn Advisor

Academic Year 2016

ABSTRACT

This project aims at the studies of temporal database features of a major commercial database management system that follows the SQL 2011 standard. The students first study the concepts and principles of temporal databases. A particular commercial DBMS will be chosen and its temporal database features will be tested. A substantially challenged temporal application will be developed.

กิตติกรรมประกาศ

โครงงานการประยุกต์ฐานข้อมูลเชิงเวลาโดยใช้มาตรฐานเอสคิวแอล ๒๐๑๑ ฉบับนี้ ไม่อาจสำเร็จลุล่วงไปได้ด้วยดี หากปราศจากความช่วยเหลือ คำแนะนำ และความอนุเคราะห์จาก รศ. ดร. ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานนี้ ท่านได้ให้ความเมตตาและให้โอกาสแก่ข้าพเจ้าตลอดในการทำงาน ข้าพเจ้ารู้สึกซาบซึ้งใจที่อาจารย์คอยติดตามความก้าวหน้า คอยเป็นห่วง และให้กำลังใจเสมอมา พร้อมทั้งช่วยแนะนำการทำโครงงานชิ้นนี้ตั้งแต่ต้นจนจบ

ขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนวิชาความรู้ต่างๆ แก่ข้าพเจ้ามาโดยตลอด ทำให้ข้าพเจ้าสามารถนำความรู้เหล่านั้น มาใช้พัฒนาโครงงานนี้จนสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณรุ่นพี่รวมทั้งเพื่อนๆ ทุกคนที่มีส่วนร่วมประติดประต่อความรู้ความสามารถแลกเปลี่ยนความคิดเห็นและแนะนำการทำโครงงานในด้านต่างๆตลอดมา

ขอขอบคุณบิดา มารดาและครอบครัว ที่ให้การอบรมสั่งสอน เลี้ยงดูให้โอกาสทางการศึกษาและให้การสนับสนุนในทุกๆด้านมาโดยตลอด

สุดท้ายนี้ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้ข้าพเจ้าได้เข้ามาศึกษาหาความรู้ที่นี่ข้าพเจ้ารู้สึกเป็นเกียรติอย่างยิ่งคุณความดีใดๆที่ปรากฏในโครงงานนี้ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่านมา ณ ที่นี้

ธนัตถ์ พีระไพศาลทรัพย์
พิสิฐ ศรีพรสวัสดิ์

สารบัญ

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

รูป **หน้า**

สารบัญรูป (ต่อ)

รูป **หน้า**

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูป **หน้า**

สารบัญโปรแกรม

โปรแกรม **หน้า**

สารบัญโปรแกรม (ต่อ)

สารบัญโปรแกรม (ต่อ)

สารบัญโปรแกรม (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันฐานข้อมูลที่ใช้กันอยู่ล้วนเป็นฐานข้อมูลที่เก็บเฉพาะข้อเท็จจริงที่เป็นจริงในปัจจุบัน แต่ยังมีคนอีกส่วนมากยังไม่ทราบแท้ที่จริงแล้วฐานข้อมมูลที่เราใช้กันอยู่นี้เป็นฐานข้อมูลที่เก็บข้อเท็จจริงที่เป็นจริงในอดีตด้วย เพียงแต่ฐานข้อมูลได้นำเสนอให้เราเห็นข้อมูลข้อเท็จจริงที่เป็นจริงในปัจจุบันให้เราเห็นเท่านั้น

เมื่อเราทราบว่าฐานข้อมูลแท้ที่จริงล้วมีการเก็บข้อมูลในอดีตไว้ด้วย ซึ่งตอนนี้อาจจะยังไม่ค่อยได้เห็นการใช้งานที่แพร่หลายมากนักที่จะแสดงฐานข้อมูลทั้งหมดตั้งแต่ในอดีตจนถึงปัจจุบัน ก็ไม่แน่ว่าในอนาคตข้างอันใกล้นี้จะมีการใช้ฐานข้อมูลแบบนี้กันมากขึ้น เนื่องจากข้อจำกัดในเรื่องของหน่วยความจำ และความเร็วในเรื่องของการประมวลผลจะลดลงไปเรื่อยๆ ซึ่งจะทำให้เราได้เห็นฐานข้อมูลที่แท้จริง โดยที่กล่าวมานี้เราจะเรียกฐานข้อมูลที่แสดงข้อมูลที่ข้อเท็จจริงเชิงเวลาว่าฐานข้อมูลเชิงเวลา (Temporal Database)

ฐานข้อมูลเชิงเวลานั้นได้มีผู้ที่นำเสนอแนวคิดของเรื่องนี้มาได้ระยะหนึ่งแล้ว โดยผู้นำเสนอนั้นได้แสดงให้เห็นถึงความสำคัญของมันที่จะทำให้เราค้นหาย้อนไปในอดีต การค้นหาข้อมูลในช่วงเวลาที่เราต้องการ รวมถึงการจัดการกับข้อมูลเชิงเวลาไม่ว่าจะเป็นการเพิ่มข้อมูล การลบ การปรังปรุง ซึ่งต้องมีความระมัดระวังและมีประเด็นที่จะต้องพิจารณาที่เกี่ยวกับเวลา โดยจะได้อธิบายให้ทราบในบทถัดไป

หลังจากที่ผู้นำเสนอฐานข้อมูลเชิงเวลาได้เสนอถึงประเด็นต่างๆที่มีอยู่ฐานข้อมูลเชิงเวลาแล้ว เขาได้นำเสนอถึงวิธีการแก้ปัญหาเหล่านั้น หรือทำให้ปัญหา ประเด็นเหล่านั้นมันลดลงไป โดยได้นำเสนอเป็นเครื่องมือที่เรียกว่าคุณสมบัติเชิงเวลาบนมาตรฐานเอสคิวแอล ผู้นำเสนอได้เสนอมานานแล้วแต่เพิ่งได้ถูกนำไปบรรจุเป็นส่วนในมาตรฐานเอสคิวแอล 2011 นี้เอง โดยมาตรฐานเอสคิวแอลจะเพิ่มคุณสมบัติเชิงเวลาเข้ามาทำให้เราสามารถจัดการกับฐานข้อมูลเชิงเวลาได้สะดวกมากขึ้นนั่นเอง

โครงงานชิ้นนี้จะศึกษาและนำเสนอเครื่องมือเชิงเวลาที่อยู่ในมาตรฐานเอสคิวแอล 2011 พร้อมทั้งศึกษาระบบจัดการฐานข้อมูลที่เป็นที่นิยมกันในปัจจุบันเป็นอยางมากคือระบบจัดการฐานข้อมูล Oracle โดยจะศึกษาว่าคำสั่งเอสคิวแอลที่ใช้บนระบบจัดการฐานข้อมูลนี้มีคุณสมบัติใดบ้างที่เกี่ยวข้องกับมาตรฐานเอสคิวแอล 2011 พร้อมทั้งนำความรู้จากการศึกษาไปทดลองและสรุปผล

เมื่อเราได้ความรู้ทั้งเรื่องฐานข้อมูลเชิงเวลาและเครื่องมือที่จะใช้นั่นคือคุณสมบัติเชิงเวลาที่ระบบจัดการฐานข้อมูล Oracle สามารถที่จะทำได้ เรานำองค์ความรู้เหล่านี้ไปประยุกต์ใช้กับ

คลังข้อมูล (Data Warehouse) ซึ่งคลังข้อมูลนั้นจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องและเราจะใช้เครื่องมือเหล่านี้แก้ปัญที่เกิดขึ้นบนคลังข้อมูลได้

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อศึกษาขีดความสามารถเชิงเวลาของมาตรฐานเอสคิวแอล 2011
2) เพื่อศึกษาขีดความสามารถเชิงเวลาของออราเคิลเอสคิวแอล
3) เพื่อศึกษาและสร้างคลังข้อมูลเชิงเวลาที่มีแหล่งข้อมูลเป็นฐานข้อมูลเชิงเวลาและไม่เป็นฐานข้อมูลเชิงเวลา

1.3 ขอบเขตของโครงงาน

ในขั้นแรกนี้จะศึกษาขีดความสามารถเชิงเวลาของมาตรฐานเอสคิวแอล 2011 และความสามารถเชิงเวลาของออราเคิลเอสคิวแอล พร้อมเปรียบเทียบการทำงานของระบบจัดการฐานข้อมูล Oracle Database 12c กับมาตรฐานเอสคิวแอล 2011 เกี่ยวกับการสร้างฐานข้อมูลเชิงเวลา พร้อมทั้งศึกษาการนำฐานข้อมูลเชิงเวลาไปประยุกต์ใช้กับคลังข้อมูล โดยการใช้ระบบจัดการฐานข้อมูล Oracle Database 12c ในการสร้างฐานข้อมูลเชิงเวลาและคลังข้อมูลเชิงเวลา รวมถึงศึกษาปัญหาและหาวิธีแก้ปัญาเกี่ยวกับการใช้งานคลังข้อมูลเชิงเวลา

1.4 วิธีการดำเนินงาน

1) ศึกษาแนวคิดเรื่องฐานข้อมูลเชิงเวลาและวิธีการจัดการปัญหาต่างบนฐานข้อมูลเชิงเวลา
2) ศึกษาคุณสมบัติเชิงเวลาของมาตรฐานเอสคิวแอล 2011
3) ศึกษาการททำงานของระบบจัดการฐานข้อมูล Oracle Database 12c
4) ทดสอบคุณสมบัติเชิงเวลาของออราเคิลเอสคิวแอลบนระบบจัดการฐานข้อมูล Oracle Database 12c
5) เปรียบเทียบคุณสมบัติเชิงเวลาของออราเคิลเอสคิวแอลบนระบบจัดการฐานข้อมูล Oracle Database 12c กับมาตรฐานเอสคิวแอล 2011
6) ศึกษาแนวคิดและการออกแบบคลังข้อมูล
7) ทดสอบสร้างคลังข้อมูลบนระบบจัดการฐานข้อมูล Oracle Database 12c
8) ศึกษาและทดลองการทำ Slowly Changing Dimensions
9) ศึกษาและทดลองการทำ ETL จากฐานข้อมูลเชิงเวลา
10) ค้นคว้าเอกสารงานวิจัยที่ได้รับการตรวจสอบแล้ว (Literature Review) ควบคู่ไปกับการทำโครงงานเพื่อที่จะได้ทราบว่ามีผู้ที่คิดโจทย์ในการแก้ปัญหาเช่นเดียวกับโครงงานของเราหรือไม่ ถ้ามีเขามีวิธีการแก้ไขปัญหาเหล่านั้นอย่างไร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้เรื่องการจัดการปัญหาบนฐานข้อมูลเชิงเวลา
2) ได้รับความรู้เกี่ยวกับคุณสมบัติเชิงเวลาของมาตรฐานเอสคิวแอล 2011
3) ได้ทดลองติดตั้งระบบจัดการฐานข้อมูล Oracle Database 12c พร้อมทั้งได้ทดลองใช้งาน
4) ได้รับความรู้เกี่ยวกับการออกแบบและสร้างคลังข้อมูลทั่วไปและคลังข้อมูลเชิงเวลา
5) ได้รับรู้ถึงปัญหาในการทำ Slowly Changing Dimensions
6) ได้รับรู้ถึงปัญหาในการทำ ETL จากแหล่งข้อมูลต่างๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ฐานข้อมูลเชิงเวลา (Temporal Database)[1]

ข้อมูลเชิงเวลา เป็นสิ่งที่แพร่หลาย และ ท้าทายสำหรับการจัดการในภาษา SQL ในช่วงเดือน มิถุนายน จนถึง ตุลาคม บนวารสาร Database Programming and Design (เล่มที่ 11, ฉบับที่ 6-10) มีหัวข้อพิเศษ เรื่องฐานข้อมูลเชิงเวลา จะถูกยกมา ห้าบทความ และ สามกรณีศึกษา คือ บนหออภิบาลทารกแรกเกิด, ฟาร์มเลี้ยงวัวในเชิงพาณิชย์ และ แคตตาล็อกดาวดาราศาสตร์ ถูกนำมาใช้เพื่อแสดงให้เห็นวิธีการใช้งานตัวดำเนินการในภาษา SQL แนวคิดของ Valid Time เมื่อเทียบกับ Transaction Time และเวลาในปัจจุบัน, ความเป็นลำดับและไม่เป็นลำดับ ของ integrity constraints, เป้าหมาย และและการปรับเปลี่ยนเน้นย้ำ

ระบบฐานข้อมูลที่รองรับ การบันทึก ข้อมูลเวลาที่แตกต่างกัน (time-varying information) คือ ฐานข้อมูลเชิงเวลา (Temporal Database) บนการตีความของบริบท "เวลาที่แตกต่างกัน" ในเชิงของการเก็บข้อมูลลงบนฐานข้อมูลนั้นเป็นเรื่องที่มีการถกเถียงกัน เพราะถ้าเก็บข้อมูลเวลาเฉพาะสถานะปัจจุบัน(Current State) ฐานข้อมูลนี้จะต้องมีการเปลี่ยนแปลงตามความเป็นจริงที่เปลี่ยนไป เพื่อให้พิจารณาข้อมูลเวลาที่แตกต่างของฐานข้อมูลได้ ในบริบท "ฐานข้อมูลในอดีต" ตัวฐานข้อมูลจะต้องเก็บเฉพาะ "ข้อมูลในอดีต" หรืออาจจะเก็บข้อมูลในอนาคตก็ได้

ความหมายอย่างเป็นทางการของ ฐานข้อมูลเชิงเวลา คือ "ฐานข้อมูลที่สนับสนุนข้อมูลด้านเวลาส่วนหนึ่ง ไม่นับเวลาที่ผู้ใช้กำหนด (Userdefined time)" โดย เวลาที่ผู้ใช้กำหนด (Userdefined time) นั้นนิยามได้ว่า "ขอบเขตคุณสมบัติของวันที่และเวลาที่เป็นส่วนหนึ่งของค่าความจริง เวลาที่ผู้ใช้กำหนดขนานกับขอบเขต เช่น เงิน และ จำนวนเต็ม อาจจะอยู่ในรูปของคุณสมบัติเช่น วันเกิด และการจ้างงาน"

2.1.1 ประเภทและความหมายของเวลา

การเก็บข้อมูลเชิงเวลา นั้นจะต้องมีการระบุประเภทของเวลาในฐานข้อมูล ซึ่งจำแนกออกเป็น 3 ประเภทดังนี้

1) Valid time คือ เวลาที่มีค่าความจริงเป็นจริง แสดงความสัมพันที่เป็นจริงของข้อมูล หรือช่วงเวลาที่ข้อมูลนั้นเป็นจริงบนฐานข้อมูล ซึ่งต้องมีการกำหนดระยะเวลา
2) Transaction time คือ เวลาที่ข้อมูลถูก DBMS (Database Management System) จัดเก็บลงฐานข้อมูลโดยอัตโนมัติ

---

[1] Richard T. Snodgrass. September 3, 1998. Managing Temporal Data A Five-Part Series
Chitchanok Potipat, Thidarat Poonpetchphan. 2016. Temporal Data Warehouse

3) Userdefined time คือ ขอบเขตคุณสมบัติของวันที่และเวลาที่เป็นส่วนหนึ่งของค่าความจริง เป็นเวลาที่ผู้ใช้กำหนด แต่ระบบจะมองเป็นข้อมูลธรรมดา เช่น วันที่จ้างงาน วันเกิด เป็นต้น

2.1.2 ความซ้ำซ้อนของข้อมูลเชิงเวลา (Duplicates and Septuplets)

ในการจัดเก็บข้อมูลเชิงเวลาจะต้องพบกับความท้าทายในการนิยาม และการจัดการความหมายของเวลา เพื่อไม่ให้เกิดความซ้ำซ้อนของข้อมูลที่จะส่งผลกระทบกับการเพิ่มข้อมูล (Insert), การปรับปรุงข้อมูล (Update) และการลบข้อมูล (Delete) เพื่อให้เห็นปัญหา และกรณีที่ทำให้เกิดการซ้ำซ้อนของข้อมูลเชิงเวลา จึงได้ยกตัวอย่างตารางแสดงสถานะคนใช้จากกรณีฝาแฝด 7 คน ของ McCaughey (McCaughey septuplets) ดังตารางที่ 2.1 มาเพื่อแสดงตัวอย่างของความซ้ำซ้อนที่เกิดขึ้น

ตาราง 2.1 ตารางแสดงสถานะคนใช้จากกรณีฝาแฝด 7 คน ของ McCaughey (McCaughey septuplets)

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

จากตารางข้างต้นเป็นตารางเชิงเวลาบันทึกสถานะของฝาแฝด โดยมีข้อมูลของ ชื่อผู้ป่วย (Name), สถานะ (Status), วันที่เริ่มเกิดอาการป่วย (From_date) และวันสิ้นสุดของอาการ (To_date) โดยลักษณะของอาการจะแบ่งได้เป็น พอดีพอร้าย (fair), วิกฤต (serious) และ วิกฤตอันตราย (critical)

ในตารางที่ 2.1 ที่สามแถวสุดท้าย นั้นมีความซ้ำซ้อนเกิดขึ้นได้ 4 แบบ ได้ดังนี้

2.1.2.1 ความซ้ำซ้อนแบบเป็นลำดับขั้นตอน (Sequenced Duplicate)

ความซ้ำซ้อนแบบเป็นลำดับขั้นตอนคือ เมื่อค่าในแถวเดียวกัน มีค่าความจริง (Fact) ซ้ำกันและมีช่วงเวลา (Valid time) คาบเกี่ยวกัน ตามตารางที่ 1.2 โดยจะแก้ปัญหาความซ้ำซ้อนนี้ด้วยการสร้างเงื่อนไขเพื่อไม่ให้บันทึกค่าที่ซ้ำซ้อนของผู้ป่วยเด็กที่มีสถานะเหมือนกัน

ตาราง 2.2 ตารางแสดงตัวอย่างความซ้ำซ้อนแบบเป็นลำดับขั้นตอน (Sequenced Duplicate)

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

2.1.2.2 ความซ้ำซ้อนแบบปัจจุบัน (Current Duplicate)

ความซ้ำซ้อนแบบปัจจุบันคือ ค่าความจริง(Fact) ซ้ำซ้อนกัน มีช่วงเวลา (Valid time) คาบเกี่ยวกันแล้วมี to_date ตัวใดตัวนึงหรือทั้งสองตัวคาบเกี่ยวมาถึงปัจจุบัน แก้ปัญหาได้โดยการตรวจสอบก่อนการบันทึกข้อมูลให้ถูกต้อง

ตาราง 2.3 ตารางแสดงตัวอย่างความซ้ำซ้อนแบบปัจจุบัน (Current Duplicate)

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

2.1.2.3 ความซ้ำซ้อนแบบค่าเท่ากัน (Value-equivalent Duplicate)

ความซ้ำซ้อนแบบค่าเท่ากันคือ เมื่อมีค่าความจริง (Fact) มีค่าเหมือนกัน โดยไม่ได้มีการนำช่วงเวลา (Valid time) มาคิด โดยสามารถแก้ปัญหาได้ด้วยการสร้างกฎ (Intrigity Constrain) เพื่อให้ไม่ยินยอมถ้ามีการบันทึกค่าที่ซ้ำกันลงไป

ตาราง 2.4 ตารางแสดงตัวอย่างความซ้ำซ้อนแบบค่าเท่ากัน (Value-equivalent Duplicate)

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

2.1.2.4 ความซ้ำซ้อนแบบไม่เป็นลำดับขั้นตอน (Nonsequenced Duplicate)

ความซ้ำซ้อนแบบไม่เป็นลำดับขั้นตอนคือ การที่ค่าทุกค่าในแถว มีการซ้ำกันกับแถวอื่นโดยไม่มีการสนใจเรื่องเวลาเป็นพิเศษ แก้ไขได้โดยการกำหนด Primary Key ให้กับ Name, Status, From_date และ To_date ทำให้ไม่เกิดการซ้ำกันของสถานะ แต่วิธีการแก้ปัญหาแบบนี้ยังไม่เพียงพอ

ตาราง 2.5 ตารางแสดงตัวอย่างความซ้ำซ้อนแบบไม่เป็นลำดับขั้นตอน (Nonsequenced Duplicate)

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

2.1.3 การตอบคำถามโดยใช้ Valid-Time State Tables

Valid-time State Table คือ คอลลัมของช่วงเวลาที่เป็นในฐานข้อมูลเชิงเวลาที่เพิ่ม Valid-time เข้ามาในตารางเพื่อบอกว่าข้อมูลไหนบ้างยังเป็นจริงในเวลานั้นๆ ตัวอย่างในตารางที่ 2.6

ตาราง 2.6 Valid-time state table ของฝูงวัว

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

ตารางนี้นำเสนอข้อมูลโดยมี FDYD_ID แทนหมายเลขลานให้อาหารของวัว, LOT_ID_NUM แทนหมายเลขของฝูงวัว, PEN_ID แทนเลขคอกของฝูงวัว, HD_CNT แทนจำนวนของวัวในฝูง, FROM_DATE และ TO_DATE เป็นคาบเวลาของความถูกต้องข้องข้อมูล

ในกรณีข้างต้นจะเห็นว่า วัว 17 ตัวอยู่ในคอกที่ 1 เป็นเวลา 11 วัน แล้วจะย้ายออกจากลานให้อาหารวันที่ 18 กุมภาพันธ์ 1998 ส่วนวัวในฝูง 374 จะยังอยู่ในคอกที่ 1 ต่อไป (จะใช้คำว่า "ตลอดไป" เพื่อแสดงถึงแถวที่ถูกต้องในปัจจุบัน) และ วัว 23 ตัวจากฝูงที่ 219 ถูกย้ายจากคอกที่ 1 ไปยังคอกที่ 2 วันที่ 1 มีนาคม 1998 เหลือวัว 20 ตัวที่ไม่ได้ย้ายไปคอกที่ 2 จนถึงวันที่ 14 มีนาคม 1998 ก่อนที่จะมารวมกันในคอกที่ 2 จนถึงปัจจุบัน โดยตัวอย่างการตอบคำถามข้อมูลจะมีดังนี้

2.1.3.1 Temporal Projection and Selection

Temporal Projection คือ การที่สามารถตอบคำถามได้จากตัวตารางเองโดยไม่ต้องรวมตาราง (Join) ในตัวอย่างที่คำถามที่เป็น Current query เช่น "มีวัวกี่ตัวของฝูง 219 ที่อยู่ในลานให้อาหารที่ 1 และอยู่ในคอกใดบ้าง" จะเห็นว่าในการตอบคำถามนี้เราสามารถใช้การเขียนโปรแกรมบน SQL ธรรมดา ดังโปรแกรมที่ 2.1 แต่ในการตอบคำถามลักษณะนี้บน Valid-time state table นั้นจะต้องมีการระบบช่วงเวลาเข้าไปด้วย ดังโปรแกรมที่ 2.2 ผลลัพท์ที่ออมาเป็นดังตารางที่ 2.3 ในการทำ Sequence Query คำถามที่ถามจะมีการระบุถึงช่วงเวลาในอดีตถึงปัจจุบัน เช่น "ในเวลาที่ผ่านมา วัวฝูง 219 เคยอยู่คอกใดบ้างและ คอกละกี่ตัว" สามารถตอบคำถามเป็นโปรแกรมที่ 2.3 จะเห็นว่าการไม่ระบุเวลาในฐานข้อมูเชิงเวลาจะมีความหมายเทียบกับเวลาทั้งหมด ตั้งแต่อดีตถึงปัจจุบัน ซึ่งในฐานข้อมูลแบบปกติจะไม่สามารถตอบคำถามได้ โดยมีผลลัพธ์ดังตารางที่ 2.8 และการทำ Non Sequence Query จากคำถาม "มีจำนวนวัวของฝูง 219 จำนวนกี่ตัวที่อยู่ในลานให้อาหารที่ 1" จะสามารถเขียนได้ออกมาเป็นโปรแกรมที่ 2.4 และได้ผลลัพธ์เป็นตารางที่ 2.9

โปรแกรม 2.1 การ Query กับคำถามที่เป็น 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 กับคำถามที่เป็น 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 ผลลัพท์จากการ Query ในโปรแกรมที่ 2.1 และ 2.2

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 ผลลัพท์จากการ Query ในโปรแกรมที่ 2.3

PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	43	1998-02-25	1988-03-01
1	20	1998-03-01	1988-03-14
2	23	1998-03-01	1988-03-14
2	43	1998-03-14	9999-12-31

โปรแกรม 2.4 การ Query แบบ Non Sequence Query

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

ตาราง 2.9 ผลลัพท์จากการ Query ในโปรแกรมที่ 2.4

PEN_ID	HD_CNT
1	43
1	20
2	23
2	43

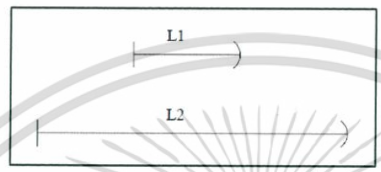
2.1.3.2 Temporal Join

Temporal join เป็นการตอบคำถามที่ไม่สามรถตอบได้โดยใช้ข้อมูลในตารางเดียว จึงต้องนำการ Join สองตารางเพื่อใช้ตอบคำถาม ในที่นี้จะใช้ตัวอย่างคำถามที่ว่า “มีวัวฝูงใดบ้างที่อาศัยอยู่ในคอกเดียวกัน” จากคำถามนี้จะพบว่าการใช้ฐานข้อมูลธรรมดาจะไม่สามารถตอบคำถามนี้ได้ แต่ถ้าเป็นฐานข้อมูลเชิงเวลาจะต้องพิจารณาถึงเหตุการณ์ที่อยู่ร่วมกันของวัวทั้งสองฝูง

1) วัวทั้งสองฝูงอยู่ด้วยกันจริงในอดีต (Sequence Query)
2) วัวทั้งสองฝูงอยู่ด้วยกันจริงในปัจจุบัน (Current Query)
3) วัวทั้งสองฝูงอยู่คอกเดียวกัน แต่คนละเวลา (Non Sequence Query)

คำตอบที่ 1 วัวทั้งสองฝูงอยู่ด้วยกันจริงในอดีต เมื่อพิจารณา จะได้เป็นเส้นเวลาที่เขียนออกมาได้ดังนี้

กรณีที่ 1 วัวฝูงที่ 1 มาอยู่กับวัวของฝูงที่ 2 ในขณะที่วัวฝูงที่ 2 ยังอยู่ในคอกเดิม และออกไปก่อนฝูงที่ 2 จะเป็นดังรูปที่ 2.1



รูป 2.1 ลักษณะของกรณีที่ 1

กรณีที่ 2 วัวฝูงที่ 1 มาอยู่กับวัวฝูงที่ 2 ในณะที่วัวฝูงที่ 2 ยังอยู่ในคอกเดิม แต่ย้ายออกจากคอกหลังจากวัวฝูงที่ 2 ย้ายออก จะเป็นดังรูปที่ 2.2

รูป 2.2 ลักษณะของกรณีที่ 2

กรณีที่ 3 คล้ายกับ กรณีที่ 2 แต่ต่างกันตรงที่เป็น วัวฝูงที่ 2 มาอยู่กับวัวฝูงที่ 1 ในขณะที่วัวฝูงที่ 1 ยังอยู่ในคอกเดิม แต่ย้ายออกหบังจากวัวฝูงที่ 1 ย้ายออก

กรณีที่ 4 คล้ายกลับกรณีที่ 1 แต่ต่างกันตรง วัวฝูงที่ 2 มาอยู่กับวัวของฝูงที่ 1 ในขณะที่วัวฝูงที่ 1 ยังอยู่ในคอกเดิม และออกไปก่อนฝูงที่ 1

จะเห็นได้ว่าคำตอบในคำถามนี้เป็นไปได้ถึง 4 คำตอบ ซึ่งหากเราใช้ภาษา SQL-92 มาช่วยหาคำตอบ จะได้เพียงคำตอบเดียว ทำให้ต้องเขียน SQL เพิ่มขึ้นอีกมากมายเพื่อให้สามารถตอบคำถามนี้ได้ครบ ซึ่งเป็นการเพิ่มภาระในส่วนของการเขียนคำสั่งที่เพิ่มขึ้นอาจจะไม่ใช่เรื่องยาก แต่เป็นการเพิ่มโอกาศความผิดพลาดที่สามารถเกิดขึ้นได้

คำตอบที่ 1 วัว 2 ฝูงอยู่ด้วยกันจริงในอดีต

โปรแกรม 2.5 การ Query เพื่อหาคำตอบว่ามีวัวฝูงใดเคยอยู่ร่วมกันในอดีต

```
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
UNION
SELECT L1.LOT_ID_NUM, L2.LOT_ID_NUM, L1.PEN_ID,
L1.FROM_DATE, L2.TO_DATE
FROM LOT_LOC AS L1, LOT_LOC AS L2
WHERE L1.LOT_ID_NUM < L2.LOT_ID_NUM
AND L1.FDYD_ID = L2.FDYD_ID AND L1.PEN_ID = L2.PEN_ID
AND L1.FROM_DATE > L2.FROM_DATE_AND L2.TO_DATE <
L1.TO_DATE
AND L1.FROM_DATE < L2.TO_DATE
UNION
SELECT L1.LOT_ID_NUM, L2.LOT_ID_NUM, L1.PEN_ID,
L2.FROM_DATE, L1.TO_DATE
FROM LOT_LOC AS L1, LOT_LOC AS L2
WHERE L1.LOT_ID_NUM < L2.LOT_ID_NUM
AND L1.FDYD_ID = L2.FDYD_ID AND L1.PEN_ID = L2.PEN_ID
AND L2.FROM_DATE > L1.FROM_DATE AND L1.TO_DATE <
L2.TO_DATE
AND L2.FROM_DATE < L1.TO_DATE
UNION
SELECT L1.LOT_ID_NUM, L2.LOT_ID_NUM, L1.PEN_ID,
L2.FROM_DATE, L2.TO_DATE
FROM LOT_LOC AS L1, LOT_LOC AS L2
WHERE L1.LOT_ID_NUM < L2.LOT_ID_NUM
AND L1.FDYD_ID = L2.FDYD_ID AND L1.PEN_ID = L2.PEN_ID
AND L2.FROM_DATE >= L1.FROM_DATE AND L2.TO_DATE <=
L1.TO_DATE
```

คำตอบที่ 2 วัว 2 ฝูงอยู่ด้วยกันจริงในปัจจุบัน

โปรแกรม 2.6 การ Query เพื่อหาคำตอบว่าวัว 2 ฝูงอยู่ร่วมกันจริงในปัจจุบัน

```
SELECT L1.LOT_ID _NUM, L2.LOT_ID _NUM, L1.PEN_ID
FROM LOT_LOC AS L2, LOT_LOC AS L2
WHERE L1.LOT_ID _NUM < L2.LOT_ID _NUM
AND L2.FDYD_ID = L2.FDYD_ID
AND L1.PEN_ID = L2.PEN_ID
AND L1.TO_DATE = DATE '9999-12-31'
```

คำตอบที่ 3 วัว 2 ฝูงเคยอยู่ในคอกเดียวกันแต่คนละเวลาดังโปรแกรมที่ 2.7 ซึ่งจะต้องนำคำตอบของทั้งหมดของแต่ละคำถามมารวมกัน (Union) เพื่อตอบคำถามที่ว่า "วัวตัวใดเคยอยู่คอกเดียวกัน" จากปัญหาความยุ่งยากดังกล่าวทำให้ อาจารย์ Richard Snodgrass ผู้เขียนบทความได้พยายามผลักดันคุณสมบัติ (Feature) ใหม่เข้าไปในภาษา SQL เพื่อลดความยุ่งยากในการพัฒนาโปรแกรมประยุกต์เชิงเวลาจากคำถามเดิมที่ว่า "วัวตัวใดเคยอยู่คอกเดียวกัน" สำหรับคำตอบที่ว่า วัวต้องเคยอยู่คอกเดียวกันจริงไม่ว่าจะช่วงเวลาใดในอดีต จะสามารถทำได้โดยเพิ่มคำว่า "VALIDTIME" เข้าไปในโปรแกรมภาษา SQL ดังโปรแกรมที่ 2.8 โดย DBMS จะจัดการกรณีของเวลาทั้ง 4 แบบให้เอง โดยผู้พัฒนาไม่จำเป็นต้องรู้ว่าระบบฐานข้อมูลมีคอลลัมเวลาอยู่ด้วยหรือไม่ ถ้าหากต้องการคำตอบคนละช่วงเวลา ทำได้โดยการระบุ "NONSEQUENCE" ก่อนหน้า "VALIDTIME"

โปรแกรม 2.7 การ Query เพื่อหาคำตอบว่าวัว 2 ฝูงอยู่คอกเดียวกันแต่คนละเวลา

```
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
```

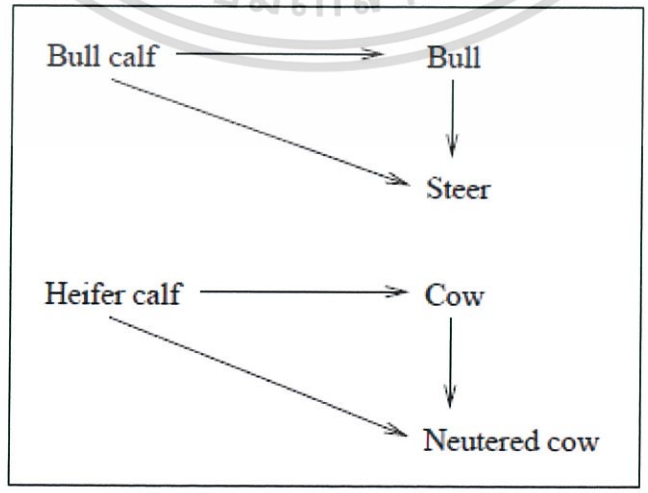
2.1.3 Modifying Valid-Time State Tables

ในส่วนก่อนหน้านี้พูดถึงการติดตามวัวที่ย้ายจากคอกหนึ่ง ไปยังอีกคอกหนึ่งในลานให้อาหารเดียวกัน ในหัวข้อต่อไปจะเดี่ยวกับธรรมชาติของการระบาดของโรคในปศุสัตว์และปัจจัยที่เกี่ยวข้อ

2.1.3.1 Current Modifications

จากหัวข้อก่อนหน้านี้จะสังเกตเห็นว่าในการ Select ข้อมูลเชิงเวลาจะมีความยุ่งยากพอสมควร ในการปรับปรุงข้อมูลก็มีความยุ่งยากเช่นเดียวกัน โดยในการแก้ไขข้อมูลที่อยู่ในปัจจุบัน (Current) นั้นจะแบ่งออกเป็น

1) General Scenario เป็นการอนุญาติแทรกข้อมูล, ปรับปรุงข้อมูล และ ลบข้อมูลในช่วงเวลาใดก็ได้
2) Restricted Scenario เป็นการอนุญาติให้แก้ไขข้อมูลตารางเฉพาะช่วงเวลาปัจจุบันของข้อมูลเท่านั้น โดยตัวอย่างตารางแสดงสถานะของฝูงวัวซึ่งแผนภาพแสดงสถานะของวัวดังตารางที่ 2.10 ซึ่งจากตารางจะเห็นว่าฝูง 101 เป็น Calf ตั้งแต่วันที่ 1998-01-01 ถึง 1998-03-23 จากนั้นก็ถูกตอนตั้งแต่วันที่ 1998-03-23 ถึงปัจจุบัน



รูป 2.3 สถานะของวัวก่อนถูกตอน แยกตามเพศ

ตาราง 2.10 สถานะของฝูงวัว

LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	C	1998-01-01	1988-03-23
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 สามารถทำได้ดังตัวอย่างในโปรแกรมที่ 2.9

โปรแกรม 2.9 การทำ Current Insert

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

Current Delete ในการลบข้อมูล LOT101 ในปัจจุบันจะไม่สามารถลบแบบปกติได้ดังโปรแกรมที่ 2.10 ซึ่งต้องมีการนำเวลาที่คาบเกี่ยวมาพิจารณาด้วย เพื่อให้ลบไปเฉพาะเวลาในปัจจุบัน โดยใช้ตัวอย่างของตารางแสดงสถานะของฝูงวัวดังตารางที่ 2.11 จากตาราง ฝูง 234 เป็น Calf ตั้งแต่ 1998-10-17 โดย สมมติให้วันนี้เป็นวันที่ 1998-07-29 ถ้าต้องการลบฝูง 234 จะต้องแก้ไขให้ความเป็นจริงของข้อมูลสิ้นสุดที่วันนี้ และในอนาคตต้องไม่มีข้อมูลนั้นอยู่แล้ว ดังโปรแกรมที่ 2.11 โดยจะเป็นคำสั่งแก้ไขโดยแก้ไขฝูง 234 c 1998-02-17 1998-10-17 ซึ่งเป็นแถวที่มีเวลาคาบเกี่ยวกับเวลาปัจจุบัน โดยจะปรับ To_date เป็น CURRENT DATE และลบ 234 S 1998-10-17 9999-12-31 ที่เป็นเวลาในอนาคตออกไป จะได้ผลลัพธ์ดังตารางที่ 2.12

โปรแกรม 2.10 การทำ Current Delete

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

ตาราง 2.11 สถานะของฝูงวัวที่ใช้เป็นตัวอย่างในการทำ Current Delete

LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	C	1998-01-01	1988-03-23
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 การลบข้อมูลตามช่วงเวลาที่ต้องการ

```
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
```

ตาราง 2.12 สถานะของฝูงวัวหลังจากการทำ Current Delete

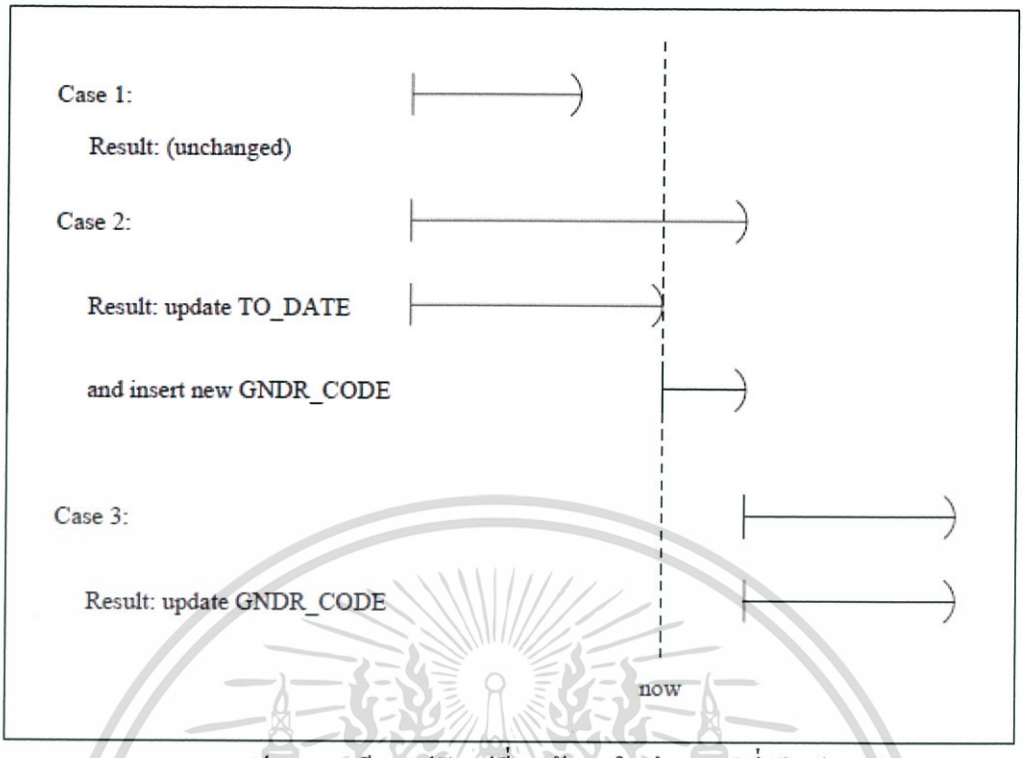
LOT_ID_NUM	GNDR_CODE	FROM_DATE	TO_DATE
101	C	1998-01-01	1988-03-23
101	S	1998-03-23	9999-12-31
234	C	1998-02-17	1998-07-29
799	C	1998-03-12	9999-12-31

Current Update การปรับปรุงข้อมูลในช่วงเวลาที่คาบเกี่ยวกัน จะไม่สามารถปรับปรุงด้วยวิธีปกติ โดยจะต้องทำการตรวจสอบช่วงเวบาเดิมก่อนทำการปรับปรุง ว่ามีการคาบเกี่ยวกันแบบไหน ซึ่งสามารถพิจารณาได้ 3 กรณี ตามรูปที่ดังนี้

กรณีที่ 1 (Case 1) ปรับปรุงข้อมูลในอดีตในเวลาปัจจุบัน โดยแก้ไข To_date เป็นเวลาในปัจจุบัน

กรณีที่ 2 (Case 2) ปรับปรุงข้อมูลที่ยังเป็นจริงในปัจจุบัน โดยการปรับปรุง To_date ของแถวเดิม เป็นเวลาในปัจจุบัน แล้วแทรกแถวใหม่ที่มี From_date เป็นเวลาปัจจุบัน

กรณีที่ 3 (Case 3) ปรับปรุงข้อมูลในอนาคตในเวลาปัจจุบัน



รูป 2.4 กรณีการปรับเปลี่ยนข้อมูลในช่วงเวลาที่เป็นปัจจุบัน

2.1.3.2 Sequenced Modifications

1) Sequence Insert ในการทำ Sequence นั้นจะต้องระบุ Validtime ที่ Fact นั้นเป็นจริงตั้งแต่เมื่อไรถึงเมื่อไร ดังตัวอย่างในโปรแกรมที่ 2.12

โปรแกรม 2.12 การทำ Sequence Insert

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

2) Sequence Delete ในกรณีการทำ Sequence Delete จะขอยกตัวอย่าง ได้มีการให้ฝูงวัว 234 ย้ายออกจากฟาร์ม 3 ในสัปดาห์แรกของเดือนตุลาคม ซึ่งเป็ นช่วงเวลาที่มีการวางแผนจะเปลี่ยนฝูงวัว 234 ให้เป็นวัวตอน โดยช่วงเวลาที่ต้องการเปลี่ยนแปลงข้อมูล คือวันที่ '1998-10-01' ถึงวันที่ '1998-10-22' โดยการเปลี่ยนแปลงลักษณะนี้ทำให้ต้องพิจารณาการแปลี่ยนแปลงข้อมูล ซึ่งจะเกิดกรณีของการเปลี่ยนแปลงข้อมูลได้4 กรณี ดังรูปที่ 2.5

2.1) PV คือช่วงเวลาของข้อมูลเดิม เป็น fact เป็นจริงอยู่ในฐานข้อมูล

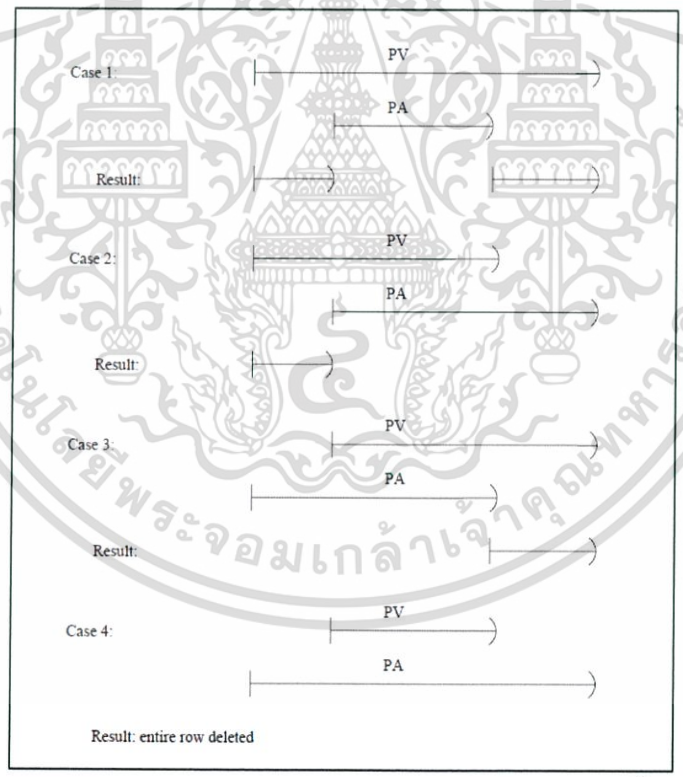
2.2) PA คือ ช่วงเวลาที่ต้องการเปลี่ยนแปลงข้อมูล

กรณีที่ 1 PA เกิดทีหลังและสิ้นสุดก่อน PV ผลจากการลบข้อมูลจะเกิดการปรับปรุง 1 แถว โดย From_date เท่ากับ From_date ของ PV แล้วเปลี่ยน To_date เป็น From_date ของ PA จากนั้นแทรกแถวอีก 1 แถวโดยให้ To_date เท่ากับ To_date ของ PA และให้ 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_dateเท่ากับ From_date ของ PA

กรณีที่ 3 PA เกิดก่อน PV และสิ้นสุดลงก่อน PV ผลของการลบจะเป็นการปรับปรุง 1 แถว โดย From_date จะเท่ากับ To_date ของ PA และ To_date เท่ากับ To_date ของ PV

กรณีที่ 4 PA เกิดขึ้นก่อนและสิ้นสุดลงหลัง PV ผลของการลบจะเป็นการลบแถวข้อมูลของ PA ออกจากฐานข้อมูล



รูป 2.5 กรณีของลบช่วงเวลาแบบ Sequence (Sequence Delete) ทั้ง 4 กรณี

จากตัวอย่าง ถ้าต้องการลบฝูง 234 ในช่วงเวลาตั้งแต่ 01-10-1998 ถึง 10-1998-22 จะต้องพิจารณาทั้ง 4 กรณีโดยใช้คำสั่ง SQL ในโปรอแกรมที่ 2.13 - 2.16 ทั้ง 4 กรณี เพื่อให้เกิดการลบที่ถูกต้อง โดยในแต่ละคำสั่งจะต้องมีการตรวจสอบ

ความถูกต้องทั้ง From_date และ To_date ซึ่งมีความซับซ้อน และผิดพลาดได้ง่าย ในที่นี้ยังมีวิธีการเขียนที่ง่ายกว่าโดยระบุช่วงเวลาที่ต้องการลบ จากนั้น DBMS จะตรวจสอบให้ทั้ง 4 กรณี ดังโปรแกรมที่ 2.17

โปรแกรม 2.13 กรณีที่ 1 ของ Sequence Delete

```
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'
```

โปรแกรม 2.14 กรณีที่ 2 ของ Sequence Delete

```
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 ของ Sequence Delete

```
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 ของ Sequence Delete

```
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 แบบระบุช่วงเวลาของ Sequence Dalete

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

3) Sequence Update

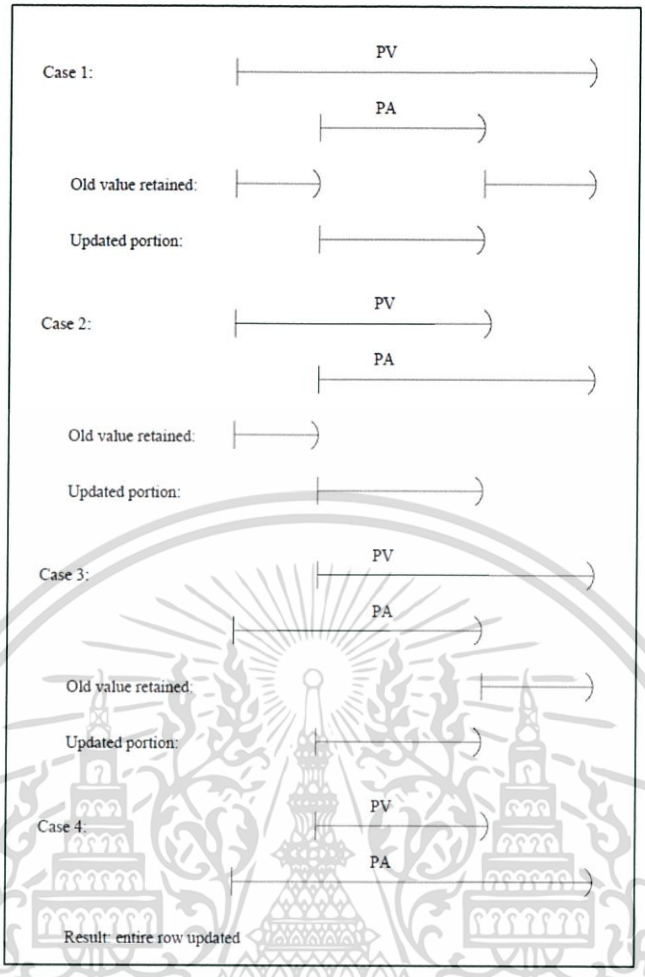
ตัวอย่าง ถ้าต้องการเปลี่ยนวัว LOT 799 เป็นวัวตอนตัวผู้ เฉพาะในเดือนมีนาคม การเปลี่ยนแปลงนี้จะต้องพิจารณา 4 กรณี ดังนี้

กรณีที่ 1 PA เกิดขึ้นหลังและสิ้นสุดก่อน PV ผลของการปรับปรุงจะเป็นการเพิ่มแถว 2 แถว และ ปรับปรุง 1 แถว โดยเพิ่มแถวแรกโดย From_date มีค่าเท่ากับ From_date ของ PV และ To_date มีค่าเท่ากับ From_date ของ PA เพิ่มแถวที่สองโดย From_date ที่ค่าเท่ากับ To_date ของ PA และ From_date มีค่าเท่ากับ From_date ของ PV ปรับปรุงแถวข้อมูลหนึ่งแถวโดยปรับปรุง From_date ของ PV 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 แถว และปรับปรุง 1 แถว เพิ่มแถวโดย From_date มีค่าเท่ากับ To_date ของ PA 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 ซึ่งสามารถใช้คาสั่งเอสคิวแอลในการปรับปรุงข้อมูลแบบ Sequence ได้ดังโปรแกรมที่ 2.18



รูป 2.6 กรณีของปรับปรุงช่วงเวลาแบบ Sequence (Sequence Update) ทั้ง 4 กรณี

โปรแกรม 2.18 การ Update แบบ Sequence ทั้ง 4 กรณี

```
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'
```

จะเห็นว่าการทำ Sequence Update ต้องใช้คำสั่ง SQL มาตรฐานที่คลอบคลุมทุกกรณี เพื่อให้การปรับปรุงข้อมูลที่ถูกต้อง จะต้องมีการตรวจสอบเวลาให้ถูกต้อง ทั้ง From_date และ To_date ซึ่งเป็นเรื่องที่ซับซ้อนและยุ่งยาก ซึ่งการเขียนคำสั่งจะง่ายขึ้นถ้ามีการระบุช่วงเวลาไปด้วยในการ Update แล้วให้ DBMS ตรวจสอบทั้ง 4 กรณี ดังตัวอย่างในโปรแกรมที่ 2.19

โปรแกรม 2.19 การ Update แบบระบุช่วงเวลาของ Sequence Update

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

2.1.3.3 Nonsequenced Modifications

การแก้ไขข้อมูลแบบ Nonsequence นั้นจะรักษาเวลาเดียวกับคอลลัมอื่นๆ โดยพิจารณาการเปลี่ยนแปลง ยกตัวอย่างการ "Delete lot 234" ตัวแปร Current คือ "Lot 234 จะออกไปจากลานให้อาหาร" ถ้าเป็นแบบ Sequence จะเป็น "Lot 234 จะไม่อยู่ในลานให้อาหารตั้งแต่ 3 อาทิตย์แรกของเดือน มิถุนายน" และถ้าเป็นการลบแบบ Nonsequence จะอ้างถึงระยะเวลาที่ต้องการลบออกไป ตัวอย่างเช่น "ลบบันทึกของ Lot 234 ที่มีช่วงเวลามากกว่า 3 เดือน" ดังโปรแกรมที่ 2.20

โปรแกรม 2.20 การ Delete แบบ Nonsequence

```
DELETE FROM LOT
WHERE LOT_ID_NUM = 234
AND (TO_DATE - FROM_DATE MONTH) > INTERVAL '3' MONTH
```

ในการลบแบบ Current กับ Sequence จะอ้างถึง ว่าเกิดอะไรขึ้นในความเป็นจริง เพราะเมื่อตัวอย่างเปลี่ยนแปลง nonsequnce จะเป็นคำสั่งที่เกี่ยวข้องกับการแสดงผลที่เฉพาะเจาะจง (ลบบันทึกที่เจาะจง) แต่กลับกันในการใช้คำสั่ง SQL ในการลบแบบ nonsequence นั้นมีความซับซ้อนเป็นอย่างมากความท้าทายของการแก้ไขปรับปรุงดังกล่าวจะต้องแม่นยำ Sequence Primary Key และ intrigrity constraints

2.2 คุณสมบัติเชิงเวลาในมาตรฐานเอสคิวแอล 2011 (Temporal Features in SQL:2011)[2]

2.2.1 ช่วงเวลา (Periods)

สิ่งสำคัญที่เป็นพื้นฐานของข้อมูลเชิงเวลาในมาตรฐานเอสคิวแอล 2011 นั้นคือความสามารถในการกำหนดและหาความต่อเนื่องกันของช่วงเวลาในแถวต่างๆของตาราง ช่วงเวลาที่ใช้จะเป็นช่วงทางคณิตศาตร์(Mathematical Interval) บนแกนของเวลา โดยจะมีการกำหนดจุดเริ่มต้นชองเวลา (Start time) และจุดจบของเวลา (End time)

[2] Krishna Kulkarni, Jan-Eike Michels. 2012. Temporal features in SQL:2011

การนำเสนอฐานข้อมูลเชิงเวลาจำนวนมากจะมีการเพิ่มชนิดของข้อมูลที่เป็นช่วงเวลา (Period data type) ที่มีการกำหนดไว้เป็นคู่ของค่าเวลา เพื่อวัตถุประสงค์ในการบอกความต่อเนื่องกันของช่วงเวลาในแถวต่างๆของตาราง มาตรฐานเอสคิวแอล 2011 ยังไม่ได้มีการทำตามแนวทางนี้ เนื่องจากการเพิ่มชนิดของข้อมูลใหม่เข้าไปในมาตรฐานเอสคิวแอลหรือเพิ่มเข้าไปในผลิตภัณฑ์ที่เป็นเอสคิวแอลนั้นมีต้นทุน เพราะว่าต้องมีการสนับสนุนข้อมูลชนิดใหม่นี้ในเครื่องมือต่างๆ ซอฟต์แวร์ต่างๆ ที่ทำงานร่วมกับภาษาเอสคิวแอล

แทนที่จะทำการเพิ่มชนิดข้อมูลแบบใหม่เข้าไปในมาตรฐานเอสคิวแอล เอสคิวแอล 2011 ได้เพิ่มการกำหนดหรือนิยามค่าของช่วงเวลา (Period definitions) เพื่อใช้กำกับหรืออธิบายข้อมูลชนิดที่เป็นช่วงเวลาในตารางต่างๆ การกำหนดนิยามค่าของช่วงเวลานั้นเป็นชื่อเรียกส่วนประกอบของตารางที่เป็นคู่ของคอลัมน์ที่เป็นเวลาเริ่มต้นและเวลาสิ้นสุดของช่วงเวลา คำสั่ง CREATE TABLE และ ALTER TABLE จะมีการปรับปรุงโครงสร้างของภาษาใหม่ (Syntax) เพื่อที่จะทำให้สามารถทำการสร้างหรือยกเลิกการกำหนดนิยามค่าของช่วงเวลาได้ คอลัมน์ของเวลาเริ่มต้นและเวลาจบของช่วงเวลานั้นเป็นคอลัมน์ธรรมดาที่มีชื่อต่างกันและไม่สามารถมีชื่อไปซ้ำกับคอลัมน์อื่นๆได้

มาตรฐานเอสคิวแอล 2011 มีการเลือกใช้ close-open period model กล่าวคือ ช่วงเวลาทั้งหมดนั้น เวลาเริ่มต้นจะเริ่มจากค่าในคอมลัมน์ของเวลาเริ่มต้น (Start time) เวลาจะดำเนินต่อเนื่องไปเรื่อยๆจนถึงเวลาจบแต่ไม่นับรวมเอาค่าในคอลัมน์เวลาสิ้นสุด (End time) สำหรับแต่ละแถวนั้นเวลาสิ้นสุดของช่วงเวลาจะต้องมีค่ามากกว่าเวลาเริ่มต้นของช่วงเวลา ในความเป็นจริงการประกาศหรือกำหนดนิยามค่าของช่วงเวลาในตารางนั้น จะปรากฏในรูปแบบของกฎที่บังคับตาราง (Table constraint) เพื่อบังคับใช้คุณสมบัติของช่วงเวลา

ในฐานข้อมูลเชิงเวลานั้นมีมิติของเวลา (Time dimensions) ที่นำเสนอข้อมมูลเชิงเวลาที่ได้รับการยอมรับอยู่ 2 ชนิด คือ

1) แวลิดไทม์ (Valid time) คือช่วงเวลาที่แถวนั้นถูกกำหนดให้เป็นจริงตามที่ผู้ใช้ฐานข้อมูลกำหนด
2) ทรานแซคชันไทม์ (Transaction time) คือช่วงเวลาที่แถวนั้นถูกบันทึกลงฐานข้อมูล

2.2.2 Application-time period tables

ตารางแบบ Application-time period นั้นมีวัตถุประสงค์เพื่อตอบสนองความต้องการของการใช้งานที่มีความสนใจในการเก็บช่วงระยะเวลาที่ข้อมูลนั้นเชื่อว่าถูกต้องหรือเป็นจริงในโลกแห่งความเป็นจริง

ความต้องการหลักของการใช้งานดังกล่าวคือ ผู้ใช้จะบันทึกค่าเวลาเริ่มต้นและเวลาสิ้นสุดของ valid time ในแต่ละแถว ผู้ใช้มีอิสระในการกำหนดค่าของเวลาเริ่มต้นและเวลาสิ้นสุด

ไม่ว่าจะเป็นการกำหนดค่าเวลาในอดีต เวลาในปัจจุบัน หรือเวลาในอนาคต ส่วนความต้องการอื่นๆ คือ ผู้ใช้ได้รับการอนุญาตให้ปรับปรุงข้อมูลของช่วงเวลาที่มีความผิดพลาดหรือเมื่อทราบข้อมูลใหม่

ตารางใดๆก็ตามที่มีการกำหนดช่วงของเวลา (period definitions) โดยใช้ชื่อที่ผู้ใช้กำหนดเอง เราจะเรียกตารางนั้นว่าตาราง Application-time period

โปรแกรม 2.21 ตัวอย่างการสร้างตารางที่เป็น Application-time period

```
CREATE TABLE EMP (
      ENO INTEGER,
      ESTART DATE,
      EEND DATE,
      EDEPT INTEGER,
      PERIOD FOR EPERIOD (ESTART, EEND)
);
```

ผู้ใช้สามารถเลือกใช้ชื่ออะไรก็ได้ที่ต้องการเป็นชื่อของช่วงเวลา เช่นเดียวกันกับชื่อของคอลัมน์เวลาเริ่มต้นและเวลาสิ้นสุดของช่วงเวลาก็สามารถใช้ชื่อที่ต้องการได้ โดยชนิดของข้อมูลในคอลัมน์เวลาเริ่มต้นและเวลาสิ้นสุดจะเป็นชนิด DATE หรือ TIMESTAMP ก็ได้ แต่ชนิดของข้อมูลของคอลัมน์ทั้งสองต้องเป็นชนิดเดียวกัน

คำสั่ง INSERT แบบธรรมดาที่เคยใช้ยังคงสามารถใช้ในการกำหนดค่าของคอลัมน์เวลาเริ่มต้นและคอลัมน์เวลาสิ้นสุดได้

โปรแกรม 2.22 ตัวอย่างการเพิ่มข้อมูลลงในตาราง Application-time period

```
INSERT INTO EMP VALUES (22217, DATE '2010-01-01', DATE
'2011-11-12', 3);
```

ตาราง 2.13 ผลลัพธ์ของการเพิ่มข้อมูลข้างต้น

ENO	ESTART	EEND	EDEPT
22217	2010-01-01	2011-11-12	3

คำสั่ง UPDATE แบบธรรมดาก็ยังสามารถนำมาใช้ปรับเปลี่ยนข้อมูลในแต่ละแถวของตารางแบบ Application-time period ได้ (การปรับเปลี่ยนข้อมูลที่ว่านี้นับรวมถึงการปรับเปลี่ยนค่าในคอลัมน์ของเวลาเริ่มต้นและเวลาสิ้นสุดด้วย) และในทำนองเดียวกันกับคำสั่ง DELETE แบบธรรมดายังคงสามารถใช้ลบแถวออกจากตารางแบบ Application-time period ได้

คุณสมบัติใหม่ในมาตรฐานเอสคิวแอล 2011 นั้นมีความสามารถในการทำการเปลี่ยนแปลงข้อมูลที่มีผลกระทบจากช่วงเวลาที่กำหนดได้ด้วย โดยคุณสมบัตินี้เป็นส่วนขยายของคำสั่ง UPDATE และคำสั่ง DELETE ที่ผู้ใช้มีการกำหนดช่วงเวลาที่สนใจ ตัวอย่างเช่น จากตารางที่ 2.13 นั้นเราจะใช้คำสั่ง UPDATE เพื่อเปลี่ยนแผนก (EDEPT) ของพนักงานหมายเลข 22217 (ENO) โดยเปลี่ยนจากแผนกที่ 3 เป็นแผนกที่ 4 ซึ่งมีช่วงเวลาตั้งแต่วันที่ 3 กุมภาพันธ์ 2011 ถึงวันที่ 10 กันยายน 2011

โปรแกรม 2.23 ตัวอย่างการใช้คำสั่ง UPDATE ตามมาตรฐานเอสคิวแอล 2011

```
UPDATE EMP FOR PORTION OF EPERIOD FROM DATE '2011-02-03'
TO DATE '2011-09-10' SET EDEPT = 4 WHERE ENO = 22217;
```

ในการปฏิบัติคำสั่งนี้ระบบจัดการฐานข้อมูล (DBMS) จะมีการนำแถวที่มีช่วงเวลาของ valid time ซ้อนทับกับช่วงเวลา P คือตั้งแต่ 3 กุมภาพันธ์ 2011 ถึง 10 กันยายน 2011 และจะมีการตีความช่วงเวลาดังกล่าวเป็นแบบ close-open ตามมาตรฐานเอสคิวแอล 2011 ดังนั้นแล้วช่วงเวลา P จะนับรวมวันที่ 3 กุมภาพันธ์ 2011 แต่ไม่นับรวมเอาวันที่ 10 กันยายน 2011 แถวใดๆก็ตามที่มีช่วงเวลา P เป็นส่วนหนึ่ง (contain) ของ valid time จะถูกปรับเปลี่ยนข้อมูล แต่ถ้าหากแถวที่มีการซ้อนทับของ valid time กับช่วงเวลา P ไม่ได้ถูกเลือกที่เวลาเริ่มต้นหรือเวลาสิ้นสุดของ valid time แล้ว แถวนั้นจะถูกแบ่งออกเป็น 2 แถวหรือ 3 แถวที่อยู่ติดกัน โดยขึ้นอยู่กับขอบเขตของช่วงเวลาที่ซ้อนทับกันและแถวเหล่านี้ที่ซ้อนทับกับช่วงเวลา P จะถูกปรับเปลี่ยนแก้ไข

ตาราง 2.14 ผลลัพธ์ของการ UPDATE ข้อมูลตามคำสั่งในโปรแกรมที่ 2.23

ENO	ESTART	EEND	EDEPT
22217	2010-01-01	2011-02-03	3
22217	2011-02-03	2011-09-10	4
22217	2011-09-10	2011-11-12	3

จากตารางที่ 2.14 เป็นผลลัพธ์การทำงานตามคำสั่งของโปรแกรมที่ 2.23 โดยกำหนดให้โปรแกรมที่ 2.23 นั้นกระทำกับตารางที่ข้อมูลตามตารางที่ 2.13 ในตัวอย่างนี้แถวที่มีค่า EDEPT ปรับเปลี่ยนเป็น 4 นั้นคือแถวเดิมซึ่งจะถูกปรับเปลี่ยน (UPDATE triggers) ส่วนสองแถวที่เหลือเป็นแถวใหม่ที่ถูกเพิ่มเข้ามา (INSERT triggers)

คำสั่ง DELETE นั้นจะใช้รูปแบบของคำสั่งในทำนองเดียวกันกับการ UPDATE คือมีการเพิ่มคำสั่ง FOR PORTION OF เพื่ออำนวยความสะดวกในการลบข้อมูลในช่วงระยะเวลาที่กำหนด

ตัวอย่างเช่น ถ้าเราต้องการลบช่วงเวลาที่อยู่ในแผนกนั้นของพนักงานหมายเลข 22217 ตั้งแต่วันที่ 3 กุมภาพันธ์ 2011 ถึง 10 กันยายน 2011

โปรแกรม 2.24 ตัวอย่างการใช้คำสั่ง DELETE ตามมาตรฐานเอสคิวแอล 2011

```
DELETE EMP FOR PORTION OF EPERIOD FROM DATE '2011-02-03'
TO DATE '2011-09-10' WHERE ENO = 22217;
```

ตัวอย่างนี้จะคล้ายกับตัวอย่างการ UPDATE ข้างต้นคือ แถวใดๆก็ตามที่ valid time มีช่วงเวลา P ตั้งแต่วันที่ 3 กุมภาพันธ์ 2011 ถึง 10 กันยายน 2011 เป็นส่วนหนึ่งใน valid time จะถูกลบทิ้ง แต่ถ้าหากแถวที่มีการซ้อนทับของ valid time กับช่วงเวลา P ไม่ได้ถูกเลือกที่เวลาเริ่มต้นหรือเวลาสิ้นสุดของ valid time แล้ว แถวนั้นจะถูกแบ่งออกเป็น 2 แถวหรือ 3 แถวที่อยู่ติดกันและแถวที่มีการซ้อนทับกับช่วงเวลา P จะถูกลบทิ้ง

ตาราง 2.15 ผลลัพธ์ของการ DELETE ข้อมูลตามคำสั่งในโปรแกรมที่ 2.4

ENO	ESTART	EEND	EDEPT
22217	2010-01-01	2011-02-03	3
22217	2011-09-10	2011-11-12	3

จากตารางที่ 2.15 นั้นเป็นผลลัพธ์การทำงานตามคำสั่งของโปรแกรมที่ 2.24 โดยกำหนดให้โปรแกรมที่ 2.24 นั้นกระทำกับตารางที่ข้อมูลตามตารางที่ 2.13 โดยแถวที่มีอยู่เดิมในตารางที่ 2.13 นั้นจะถูกลบทิ้ง (DELETE triggers) และจะมีการเพิ่มแถวใหม่เข้าไป 2 แถว (INSERT triggers)

2.2.2.1 Primary keys on application-time period tables

จากตัวอย่างที่ผ่านมาที่เป็นตารางของพนักงาน (EMP) ซึ่งเราอาจจะคิดว่า ENO เป็น Primary key อย่างไรก็ตามหลังจากที่เราได้ดูผลลัพธ์ของคำสั่ง UPDATE เราจะเห็นว่าทั้งสามแถวนั้นมี ENO เป็น 22217 ทั้งหมด ซึ่งตัวอย่างนี้แสดงให้เห็นว่า Primary key จะต้องมีการรวม valid time เข้าไปด้วย คือ รวมคอลัมน์ ESTART และ EEND เข้าไปเป็น Primary key ด้วย

การเพิ่มเพียงแค่คอลัมน์ ESTART และ EEND เข้าไปนั้นดูเหมือนว่าจะไม่เพียงพอ ให้เราลองพิจารณาตามตารางที่ 2.11

ตาราง 2.16 ตัวอย่างตารางที่มีการใช้ ENO, ESTART และ EEND ประกอบกันเป็น Primary key

ENO	ESTART	EEND	EDEPT
22217	2010-01-01	2011-09-10	3
22217	2010-02-03	2011-11-12	4

จากตารางที่ 2.16 ใน 3 คอลัมน์แรกนั้นจะเห็นว่าค่า (22217, 2010-01-01, 2011-09-10) และ (22217, 2011-02-03, 2011-11-12) ไม่ซ้ำกัน ซึ่งเป็นค่าที่ยอมรับได้ตามการกำหนดค่า Primary key ที่เป็นสามคอลัมน์ประกอบกัน (ENO, ESTART, EEND) แต่เรากลับพบว่าค่าของ valid time มีช่วงเวลาที่ซ้อนทับกัน ซึ่งถ้าหากเราตีความจากตารางดังกล่าวจะได้ว่า พนักงานหมายเลข 22217 ได้สังกัดอยู่ในแผนก 3 และ 4 ในช่วงเวลาตั้งแต่วันที่ 3 กุมภาพันธ์ 2010 ไปจนถึงวันที่ 10 กันยายน 2011 ซึ่งในบางครั้งผู้ใช้มีความต้องการที่จะให้พนักงานสามารถสังกัดแผนกได้ 2 แผนกในเวลาเดียวกัน อย่างไรก็ตามโดยทั่วไปแล้วพนักงานสามารถสังกัดได้เพียงแค่แผนกเดียวในช่วงเวลาใดช่วงเวลาหนึ่ง เพื่อให้สามารถทำตามเงื่อนไขที่ว่านี้ได้ โดยไม่มีการซ้อนทับกันของช่วงเวลา เราสามารถใช้รูปแบบคำสั่งตามโปรแกรมที่ 2.25 ได้

โปรแกรม 2.25 ตัวอย่างการใช้คำสั่งประกาศ Primary key ตามมาตรฐานเอสคิวแอล 2011

```
ALTER TABLE EMP ADD PRIMARY KEY (ENO, EPERIOD WITHOUT
OVERLAPS);
```

2.2.2.2 Referential constraints on application-time period tables

ต่อเนื่องจากตัวอย่างก่อนหน้านี้ที่เป็นตารางพนักงาน (EMP) สมมติว่าให้มีอีกตารางโดยมีการประกาศไว้ตามโปรแกรมที่ 2.26

โปรแกรม 2.26 คำสั่งประกาศตารางตัวอย่างที่เก็บข้อมูลของ Department

```
CREATE TABLE DEPT (
      DNO INTEGER,
      DSTART DATE,
      DEND DATE,
      DNAME VARCHAR(30),
      PERIOD FOR DPERIOD (DSTART, DEND),
      PRIMARY KEY (DNO, DPERIOD WITHOUT OVERLAPS)
);
```

สมมติว่าเราต้องการที่จะทำให้มั่นใจได้ว่าเวลาใดๆของทุกๆค่าใน EDEPT ของตาราง EMP สอดคล้องกับบางค่าของ DNO ในตาราง DEPT เช่น พนักงานแต่ละคนที่ถูกจ้างงานใน

แผนกใดในช่วงเวลาใดแล้วจะต้องมีแผนกนั้นอยู่ในช่วงเวลานั้นด้วย เราสามารถทำตามเงื่อนไขนี้ได้อย่างไร ลองดูข้อมูลตัวอย่างต่อไปนี้ โดยกำหนดให้ตาราง EMP มีข้อมูลตามตารางที่ 2.17

ตาราง 2.17 ตัวอย่างข้อมูลในตาราง EMP

ENO	ESTART	EEND	EDEPT
22218	2010-01-01	2011-02-03	3
22218	2011-02-03	2011-11-12	4

และกำหนดให้ตาราง DEPT มีข้อมูลตามตารางที่ 2.18

ตาราง 2.18 ตัวอย่างข้อมูลในตาราง DEPT

DNO	DSTART	DEND	DNAME
3	2009-01-01	2011-12-31	Test
4	2011-06-01	2011-12-31	QA

เมื่อเราตรวจสอบอย่างละเอียดกับค่าในคอลัมน์ EDEPT ของตาราง EMP และค่าในคอลัมน์ DNO ของตาราง DEPT เราอาจจะสรุปได้ว่านี่คือ Referential integrity constraint ของทั้งสองตาราง แต่เมื่อสังเกตดูเราจะพบว่าพนักงานหมายเลข 22218 นั้นได้ถูกบรรจุเข้ามาในแผนกหมายเลข 4 (DNO เท่ากับ 4) ตั้งแต่วันที่ 3 กุมภาพันธ์ 2011 ถึงวันที่ 12 พฤศจิกายน 2011 ซึ่งช่วงเวลาตั้งแต่วันที่ 3 กุมภาพันธ์ 2011 ถึงวันที่ 1 มิถุนายน 2011 นั้นยังไม่มีแผนกหมายเลข 4 ซึ่งนี่เป็นการละเมิดเงื่อนไขของเราอย่างชัดเจนคือ ทุกๆค่าของคอลัมน์ EDEPT ในตาราง EMP สอดคล้องกับบางค่าของคอลัมน์ DNO ในตาราง DEPT ที่ช่วงเวลาใดๆ เพื่อที่จะไม่ทำให้เกิดเหตุการณ์แบบนี้ขึ้น มันจะต้องมีความเป็นไปได้ที่จะทำให้ไม่เกิดข้อมูลของแถวในตารางลูก (Child table) ที่สัมพันธ์กับแถวในตารางแม่ (Parent table) นั้นมี valid time ที่ได้ไม่อยู่ใน valid time ของตารางแม่ ซึ่งเราสามารถใช้รูปแบบคำสั่งต่อไปนี้เพื่อจัดการกับปัญหานี้ได้

โปรแกรม 2.27 ตัวอย่างการใช้คำสั่งประกาศ Foreign key ตามมาตรฐานเอสคิวแอล 2011

```
ALTER TABLE EMP ADD FOREIGN KEY (EDEPT, PERIOD EPERIOD)
REFERENCES DEPT (DNO, PERIOD DPERIOD);
```

จากโปรแกรมที่ 2.27 จะเป็นการประกาศ Referential constraint ซึ่งข้อมูลจากตารางตัวอย่างนั้นจะถูกห้ามไม่ให้ละเมิดกฎนี้ได้

โดยทั่วไปแล้วแถวที่อยู่ในตารางลูกไม่จำเป็นว่าต้องสัมพันธ์กับเพียงแค่แถวเดียวในตารางแม่ ซึ่งมี valid time อยู่ใน valid time ของแถวในตารางลูก ตราบใดที่มีหลายๆแถวตารางแม่ที่สอดคล้องกับแถวในตารางลูกนั้นมี valid time ที่รวมกันแล้วยังอยู่ในช่วงของ valid time ในตารางลูกจะถือว่าไม่เป็นการละเมิด Rdferential constraint

2.2.2.3 Querying application-time period tables

ในมาตรฐานเอสคิวแอล 2011 ตารางแบบ Appliction-time period นั้นสามารถใช้คำสั่งสอบถาม (Query) ข้อมูลแบบปกติที่เคยใช้สอบถามข้อมูลได้เหมือนเดิม ตัวอย่างเช่น ถ้าเราต้องการสอบถามว่า พนักงานหมายเลข 22217 เคยทำงานที่แผนกใดในวันที่ 2 มกราคม 2011 เราสามารถใช้คำสั่งต่อไปนี้สอบถามข้อมูลได้

โปรแกรม 2.28 ตัวอย่างการสอบถามข้อมูลในตารางแบบ Application-time period 1

```
SELECT ENAME, EDEPT FROM EMP WHERE ENO = 22217
AND ESTART <= DATE '2011-01-02' AND EEND > DATE '2011-01-
02';
```

วิธีที่ง่ายที่จะกำหนดคำสั่งเพื่อสอบถามข้อมูลดังกล่าวข้างต้นเกี่ยวกับกับการจ้างงานในช่วงเวลาใดเวลาหนึ่งนั้น ในมาตรฐานเอสคิวแอล 2011 จะมีตัวดำเนินการที่ใช้เป็นเงื่อนไขในคำสั่งที่เกี่ยวข้องกับช่วงเวลาคือ CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES และ IMMEDIATELY SUCCEEDS สำหรับตัวอย่างข้างต้นนั้นเราสามารถใช้ตัวดำเนินการ CONTAINS เพื่อสอบถามข้อมูลดังกล่าวได้ตามโปรแกรมต่อไปนี้

โปรแกรม 2.29 ตัวอย่างการสอบถามข้อมูลในตารางแบบ Application-time period 2 (SQL:2011)

```
SELECT ENAME, EDEPT FROM EMP WHERE ENO = 22217
AND EPERIOD CONTAINS DATE '2011-01-02';
```

ถ้าหากว่าเราต้องการรู้ว่าพนักงานหมายเลข 22217 เคยทำงานในแผนกใดบ้างในช่วงเวลาตั้งแต่วันที่ 1 มกราคม 2010 ถึงวันที่ 1 มกราคม 2011 เราสามารถใช้คำสั่งตามโปรแกรมต่อไปนี้สอบถามได้

โปรแกรม 2.30 ตัวอย่างการสอบถามข้อมูลในตารางแบบ Application-time period 3

```
SELECT ENAME, EDEPT FROM EMP WHERE ENO = 22217
AND ESTART < DATE '2011-01-01' AND EEND > DATE '2010-01-
01';
```

จะสังเกตเห็นว่าเงื่อนไขช่วงเวลาในคำสั่งดังกล่าวเป็นแบบ close-open กล่าวคือ ช่วงเวลาดังกล่าวนับรวมวันที่ 1 มกราคม 2010 แต่ไม่นับรวมเอาวันที่ 1 มกราคม 2011 ซึ่งในอีกทางหนึ่งเราสามารถใช้ตัวดำเนินการ OVERLAPS สอบถามคำถามดังกล่าวได้ตามโปรแกรมต่อไปนี้

โปรแกรม 2.31 ตัวอย่างการสอบถามข้อมูลในตารางแบบ Application-time period 4 (SQL:2011)

```
SELECT ENAME, EDEPT FROM EMP WHERE ENO = 22217
AND EPERIOD OVERLAPS PERIOD(DATE '2010-01-01',DATE '2011-
01-01');
```

ตัวดำเนินการกับช่วงเวลานั้นมีลักษณะการทำงานคล้ายกับตัวดำเนินการช่วงของ Allen (Allen's interval operators) ที่เรารู้จักกันดี ซึ่งเราสามารถอธิบายความสอดคล้องกันระหว่างตัวดำเนินการช่วงเวลาในภาษาเอสคิวแอลกับตัวดำเนินการของ Allen ได้ดังนี้

1) ตัวดำเนินการ X OVERLAPS Y ในมาตรฐานเอสคิวแอล 2011 จะให้ผลลัพธ์เป็นค่าความจริง (Boolean) เทียบเท่าการใช้ตัวดำเนินการของ Allen ดังนี้ (X overlaps Y) OR (X overlapped_by Y) OR (X during Y) OR (X contains Y) OR (X starts Y) OR (X started_by Y) OR (X finishes Y) OR (X finished_by Y) OR (X equal Y) เราจะสังเกตเห็นว่าตัวดำเนินการ overlaps ของ Allen นั้นไม่ได้เป็นทดสอบการซ้อนทับกันของช่วงจริงๆ ซึ่งเราคิดว่าการซ้อนทับกันของช่วง 2 ช่วงนั้นถูกพิจารณาว่าทั้งสองช่วงนั้นมีอย่างน้อยหนึ่งจุดที่อยู่ร่วมกัน มันแสดงให้เห็นว่าไม่เป็นจริงสำหรับการตรวจสอบการซ้อนทับกันกับตัวดำเนินการ overlaps ของ Allen ในทางตรงกันข้ามตัวดำเนินการ OVERLAPS ในมาตรฐานเอสคิวแอล 2011 นั้นถูกตีความว่า ถ้า X OVERLAPS Y เป็นจริงแล้ว Y OVERLAPS X ต้องเป็นจริงด้วย

2) ตัวดำเนินการ X CONTAINS Y ในมาตรฐานเอสคิวแอล 2011 จะให้ผลลัพธ์เป็นค่าความจริง (Boolean) เทียบเท่าการใช้ตัวดำเนินการของ Allen ดังนี้ (X contains Y) OR (X starts Y) OR (X finishes Y) OR (X equal Y) เราจะสังเกตเห็นว่าตัวดำเนินการ contains ของ Allen นั้นไม่ได้เป็นทดสอบว่าเป็นส่วนประกอบกันจริง ซึ่งเราคิดว่าช่วง Y ถูกพิจารณาว่าอยู่ใน X ก็ต่อเมื่อทุกๆจุดของเวลาใน Y นั้นอยู่ใน X มันแสดงให้เห็นว่าตัวดำเนินการ contains ของ Allen ไม่ได้เป็นการทดสอบการเป็นส่วนประกอบกันจริง Allen ในทางตรงกันข้ามตัวดำเนินการ CONTAINS ในมาตรฐานเอสคิวแอล 2011 นั้นเป็นการทดสอบการเป็นส่วนประกอบกันจริง

3) ตัวดำเนินการ X PRECEDES Y ในมาตรฐานเอสคิวแอล 2011 จะให้ผลลัพธ์เป็นค่าความจริง (Boolean) เทียบเท่าการใช้ตัวดำเนินการของ Allen ดังนี้ (X before Y) OR (X meets Y)

4) ตัวดำเนินการ X SUCCEEDS Y ในมาตรฐานเอสคิวแอล 2011 จะให้ผลลัพธ์เป็นค่าความจริง (Boolean) เทียบเท่าการใช้ตัวดำเนินการของ Allen ดังนี้ (X after Y) OR (X met_by Y)

5) ตัวดำเนินการ X EQUALS Y, X IMMEDIATELY PRECEDES Y และ X IMMEDIATELY SUCCEEDS Y ในมาตรฐานเอสคิวแอล 2011 จะให้ผลลัพธ์เทียบเท่าการใช้ตัวดำเนินการของ Allen ดังนี้ X equal Y, X meets Y และ X met_by Y ตามลำดับ

2.3 คลังข้อมูล (Data Warehouse)[3]

2.3.1 การจัดการข้อมูลหลายมุมมอง (Multidimensional Data Management)

รูปแบบของข้อมูลรีเลชัน (Relational data model) นั้นได้ถูกนำเสนอโดย E.F. Codd ในปี 1970 ซึ่งเขานั้นได้รับรางวัล Turing Award ในอีกสิบปีถัดมา ได้เป็นส่วนประกอบสำคัญในอุตสาหกรรมฐานข้อมูลที่มีระดับหลายพันล้านดอลลาร์สหรัฐในทุกวันนี้ ในช่วงปี 1990 นั้นได้มีรูปแบบของข้อมูลแบบใหม่ปรากฏออกมานั่นคือ multidimensional data model ซึ่งอยู่บนพื้นฐานของรูปแบบข้อมูลแบบรีเลชันโดยมีวัตถุประสงค์เพื่อการวิเคราะห์ข้อมูลมากกว่าที่จะทำออนไลน์ทรานแซคชัน (Online Transaction) รูปแบบข้อมูลหลายมุมมอง (Multidimensional data model) นั้นถือว่าเป็นส่วนหนึ่งของอุตสาหกรรมธุรกิจระดับหลายพันล้านดอลลาร์สหรัฐเช่นกันซึ่งมีเป็นส่วนสำคัญคล้ายๆกับรูปแบบข้อมูลรีเลชัน

รูปแบบข้อมูลหลายมุมมองนั้นถูกออกแบบมาให้สนับสนุนต่อการวิเคราะห์ข้อมูล และมีรูปแบบข้อมูลอื่นๆอีกจำนวนมากที่ถูกนำเสนอโดยนักวิจัยในสถาบันทางการศึกษาและภาคอุตสาหกรรม โดยในทางภาคการศึกษานั้นได้นำเสนอรูปแบบโมเดลทางคณิตศาสตร์ที่ดูเป็นทางการ ในขณะที่ทางภาคอุตสาหกรรมนั้นได้นำเสนอในรูปแบบที่เฉพาะทางมากกว่า โดยจะออกมาในรูปแบบของซอฟแวร์หรือเครื่องมือสำหรับการวิเคราะห์

รูปแบบของข้อมูลหลายมุมมอง (Multidimensional data model) นั้นได้มีการจัดหมวดหมู่ของข้อมูลเป็นข้อเท็จจริง (Fact) ที่สัมพันธ์กับจำนวนหรือค่าบางอย่าง (Measures) หรือการนำมุมมอง (Dimesion) มาเป็นตัวบอกคุณสมบัติของข้อเท็จจริงนั้น ตัวอย่างเช่น ร้านขายหนังสือได้ขายหนังสือในเวลาไหนขายไปจำนวนเท่าไหร่และในราคาเท่าไหร่ ซึ่งถ้ามองมองจากตัวอย่างนั้นข้อเท็จจริง (Fact) ที่ว่าคือการซื้อและค่าหรือจำนวนที่สัมพันธ์กับข้อเท็จจริง (Measures) นั้นคือ

---

[3] Christian S. Jensen, Torben Bach Pedersen, Christian Thomsen. 2010. Multidimensional Databases and Data Warehousing

จำนวนของหนังสือและราคาที่ถูกซื้อ ส่วนมุมมอง (Dimension) ที่มีไว้สำหรับอธิบายข้อเท็จจริงนั้นก็เป็นสถานที่ที่มีการซื้อ การซื้อหนังสือนั้นอาจจะมีการรวมข้อมูลเกี่ยวกับประเภทของหนังสือ ผู้เขียนหนังสือและรวมถึงเวลาที่ซื้อหนังสือนั้นด้วย ซึ่งลักษณะของการสอบถามข้อมูลอาจเป็นการสอบถามว่าจำนวนของหนังสือที่ขายได้ต่อเดือนหรือจำนวนที่ขายได้ของผู้เขียนแต่ละคน เป็นต้น

รูปแบบข้อมูลหลายมุมมองนั้นได้มีส่วนสำคัญที่ถูกนำไปประยุกต์ใช้ 3 อย่างหลักๆ อย่างแรกคือการวิเคราะห์ข้อมูลซึ่งรูปแบบข้อมูลหลายมุมมองถูกนำไปใช้ในคลังข้อมูล (Data warehouse) พูดอย่างรวมๆจะได้ว่าคลังข้อมูลนั้นเป็นที่เก็บรวบรวมข้อมูลขนาดใหญ่ที่รวบรวมข้อมูลมาจากหลายๆแหล่งเพื่อวัตถุประสงค์สำหรับการวิเคราะห์ข้อมูลเฉพาะเรื่องที่สนใจ จึงทำให้รูปแบบข้อมูลหลายมุมมองมีประโยชน์ที่จะนำมาใช้สำหรับการวิเคราะห์ข้อมูลในคลังข้อมูล

อย่างที่สองที่รูปแบบของข้อมูลหลายมุมมองถูกนำไปใช้คือ ถูกนำไปเป็นส่วนสำคัญของระบบ On-Line Analytical Processing (OLAP) โดยระบบนี้มีความรวดเร็วในการหาคำตอบของการสอบข้อมูลที่มีจำนวนมากเกี่ยวกับรายละเอียดของข้อมูลและการหาภาพรวมของทิศทางข้อมูล ซึ่งในปัจจุบันรูปแบบข้อมูลหลายมุมมองกำลังเป็นที่นิยม

อย่างที่สามคือการนำมาทำเหมืองข้อมูล (Data mining) ซึ่งมีเป้าหมายสำหรับการทำให้เราค้นพบบางอย่างที่ปัจจุบันเรายังไม่รู้ในฐานข้อมูลขนาดใหญ่ ซึ่งมีการนำรูปแบบข้อมูลหลายมุมมองมาประยุกต์ใช้เพิ่มขึ้น ซึ่งที่จริงแล้วนั้นการจัดการรูปแบบข้อมูลหลายมุมมองเป็นเครื่องมือเหมาะต่อการค้นหาคำตอบสำหรับการทำเหมืองข้อมูล

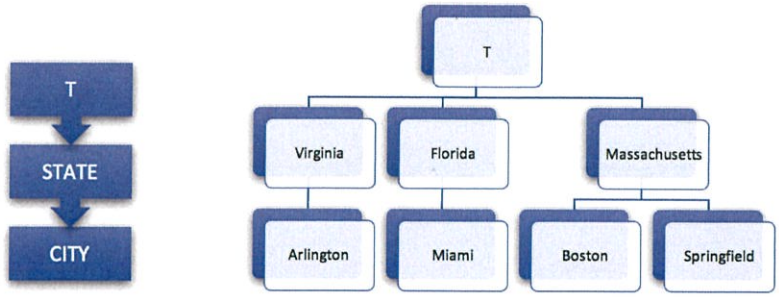
จากที่กล่าวให้เห็นแล้วนั้นเราจะเห็นได้ว่ารูปแบบของข้อมูลหลายมุมมอง (Multidimensional data model) นั้นมีความสามารถสูงสำหรับใช้ในการวิเคราะห์ข้อมูล ซึ่งถูกนำไปใช้อย่างกลว้างขวางในภาคอุตสาหกรรมและองค์ความรู้ของมันยังสามารถทำให้ผู้ที่ต้องการนำไปสำหรับการวิเคราะห์หรือทำความเข้าใจข้อมูลขนาดใหญ่ได้

2.3.2 แนวคิดพื้นฐาน (Fundamental Concepts)

2.3.2.1 มุมมอง (Dimesions)

แนวคิดของมุมมอง (Deimesions) นั้นเป็นสิ่งสำคัญที่ขาดไม่ได้สำหรับการแยกแยะหรือจำแนกสิ่งต่างๆสำหรับรูปแบบของข้อมูลหลายมุมมองโดยถูกใช้เพื่อวัตถุประสงค์หลักๆ 2 อย่างคือ ใช้สำหรับการเลือกข้อมูลและใช้สำหรับการจัดกลุ่มของข้อมูลในระดับความละเอียดของข้อมูลที่ต้องการ

มุมมอง (Dimension) เป็นจัดการส่วนประกอบระดับชั้นของสิ่งเหมือนๆกัน ซึ่งจะมีจำนวนหลายชั้นความละเอียด โดยจะแต่ละชั้นความละเอียดนั้นจะนำเสนอข้อมูลที่เราสนใจจะวิเคราะห์ โดยที่ในแต่ละค่าของมุมมองจะถูกเรียกว่า Dimension value หรือ Dimension member ซึ่งแต่ละค่าจะเป็นส่วนประกอบของแต่ละระดับชั้น



รูป 2.7 โครงสร้างและส่วนประกอบสำหรับมุมมองของสถานที่ (Location dimension)

ในรูป 2.7 นั้นได้แสดงให้เห็นถึงโครงสร้างและส่วนประกอบของมุมมองที่เกี่ยวกับสถานที่ โดยในมุมมองของสถานที่ (Location dimension) นั้นจะมีทั้งหมด 3 ระดับคือ มีระดับของเมือง (City Level) เป็นระดับที่เล็กที่สุด โดยที่ค่าทีอยู่ในระดับของเมืองนั้นเป็นกลุ่มของเมืองที่อยู่ค่าของระดับรัฐฐ (State Level) ตัวอย่างเช่น Miami นั้นเป็นเมืองที่อยู่ในรัฐ Florida เป็นต้น ซึ่งค่า T นั้นเรากำหนดให้เป็น Top Level ที่นำเสนอภาพรวมทั้งหมดของมุมมอง โดยที่ทุกๆค่ามุมมอง (Dimension value) นั้นจะต้องเป็นส่วนหนึ่งของค่า T

2.3.2.2 ข้อเท็จจริง (Facts)

ข้อเท็จจริง (Facts) มีวัตถุประสงค์เพื่อนำเสนอหัวข้อที่เราสนใจจะวิเคราะห์ เช่น สิ่งของ เหตุการณ์หรือกระบวนการอะไรบางอย่าง ซึ่งเราจะนำมันมาวิเคราะห์เพื่อให้เข้าใจพฤติกรรมของมันมากขึ้น ตัวอย่างเช่นในร้านขายหนังสือของเรา เราต้องการที่จะวิเคราะห์การขายส่วนใหญ่ในรูปแบบของข้อมูลหลายมุมมองนั้นข้อเท็จจริง (Facts) นั้นคือการอธิบายโดยใช้การรวมกัน (Combination) ของค่าในมุมมองต่างๆ ซึ่งถ้าหากว่าไม่มีส่วนใดของการรวมกันที่ว่างหรือเกิดการรวมกันที่สมบูรณ์แล้ว แปลว่าข้อเท็จจริงนั้นมีอยู่จริง และในทางกลับกันก็จะทำให้เราทราบว่าไม่มีข้อเท็จจริงนั้นอยู่ถ้าหากว่าในการรวมกันนั้นมีบางส่วนของข้อมเท็จจริงหายไป

ในส่วนต่อไปที่จะกล่าวถึงคือ โดยส่วนมากในรูปแบบข้อมูลหลายมุมมองนั้นต้องการให้หนึ่งข้อเท็จจริงนั้นมีความสัมพันธ์กับค่าในมุมมองที่อยู่ในระดับต่ำที่สุดของแต่ละมุมมอง โดยข้อเท็จจริงนั้นจะประกอบไปด้วยส่วนของจุดที่ระบุรายละเอียด โดยที่รายละเอียดเหล่านั้นสามารถหาได้จากค่าในมุมมองในระดับที่ค่านั้นอยู่ โดยที่ระดับความหยาบหรือละเอียดนั้นจะขึ้นอยู่กับจุดของรายละเอียดที่เรากำหนด

ข้อเท็จจริงนั้นเราสามารถจำแนกหรือแบ่งมันออกได้เป็น 2 ชนิดคือ ข้อเท็จจริงที่เป็นเหตุการณ์ (event facts) และข้อเท็จจริงที่เกิดขึ้นในช่วงใดช่วงหนึ่ง (snapshot facts) โดยที่ข้อเท็จจริงเชิงเหตุการณ์ (จะมีจุดของรายละเอียดที่ละเอียดที่สุด) นั้นคือเหตุการณ์ที่เกิดขึ้นจริงๆในโลกจริง (real world) ซึ่งมันจะหมายความว่าจะมีเพียงข้อเท็จจริงเดียวของแต่ละเหตุการณ์ที่มีลักษณะเฉพาะไม่เหมือนใคร (unique event) ที่เกิดขึ้นในเหตุการณ์ทั้งหมดในโลกจริงที่เราได้เก็บ

ตรวจวัดมันไว้ได้ ตัวอย่างเช่น ถ้าเป็นจุดของข้อเท็จจริงที่ละเอียดจะได้เป็น เหตุการณ์การณ์ที่เราขายหนังสือแต่ละเล่ม แต่ถ้าเป็นจุดของข้อเท็จจริงที่หยาบขึ้นมาหน่อยจะได้เป็นว่า เหตุการณ์ที่เราขายหนังสือในแต่ละวัน เป็นต้น

โดยทั่วไปแล้วข้อเท็จจริงนั้นมักจะมีส่วนเกี่ยวกับกับจำนวนบางอย่างซึ่งเราเรียกว่าค่าที่วัด (measure) ซึ่งเป็นคุณสมบัติที่เราต้องการจะวิเคราะห์ ตัวอย่างเช่น จำนวนสิ่งของหรือจำนวนเงิน เป็นต้นโดยจะอธิบายในหัวข้อถัดไป

ถ้าหากว่าข้อเท็จจริงนั้นไม่มีค่าที่เราจะวัดได้ เราจะเรียกมันว่า Factless facts อย่างไรก็ตามนี่มันคงเป็นกรณีที่แย่ที่สุดหรือเราอาจจะโชคร้ายที่มันเกิดขึ้น หรือเราจะเรียกมันว่า Measureless facts ก็ได้ โดยที่ข้อเท็จจริงเหล่านี้จะมักจะเกิดขึ้นในบางเหตุการณ์

มาถึงข้อเท็จจริงที่เกิดขึ้นในช่วงใดช่วงหนึ่ง (snapshot facts) หรืออาจจะเรียกว่า State facts โดยจะมีแนวคิดของสภาพแวดล้อม (State) ที่เปลี่ยนแปลงในเวลาใดเวลาหนึ่ง ตัวอย่างเช่น ปริมาณสินค้าที่อยู่ในร้านหรือคลังเก็บสินค้า หรือ จำนวนของผู้เข้าใช้เว็บไซต์ ซึ่งหากเราเก็บหรือวัดข้อมูลที่เวลาต่างกันอาจได้ข้อเท็จจริงที่ต่างกัน

มันสำคัญมากที่จะต้องเข้าใจเพื่อมีข้อเท็จจริงเชิงเหตุการณ์ (event facts) และข้อเท็จจริงที่เกิดขึ้นในช่วงเวลาใดเวลาหนึ่ง (snapshot facts) เกิดขึ้น หากพูดให้เข้าใจง่ายๆจะได้ว่า Event facts นั้นเป็นการอธิบายเหตุการณ์ที่เกิดขึ้น ตัวอย่าเช่นการขาย โดยที่มันเกิดขึ้นในโลกจริงด้วย ในหลักการแล้ว Event facts นั้นจะเกิดขึ้นในเวลาไหนก็ได้ซึ่งเราไม่สามารถรู้ได้ ในทางตรงกันข้าม Snapshot facts นั้นเป็นการอธิบายสภาพแวดล้อมที่เรารู้เวลาแน่นอน เช่น ปริมาณสินค้าของวันแรกในแต่ละเดือน เป็นต้น

2.3.2.3 ค่าวัด (Measures)

การวัดมีสององค์ประกอบคือคุณสมบัติเชิงตัวเลขของความเป็นจริงเช่นราคาขายหรือกำไรและสูตร (ซึ่งมักเป็นฟังก์ชันการรวบรวมข้อมูลแบบง่ายเช่น SUM) ที่สามารถนำมาใช้เพื่อรวมค่าต่างๆเข้าด้วยกัน ในฐานข้อมูลขนาดใหญ่มาตรการต่างๆโดยทั่วไปแสดงถึงคุณสมบัติของข้อเท็จจริงที่เลือกซึ่งผู้ใช้ต้องการศึกษาเช่นเพื่อวัตถุประสงค์ในการเพิ่มประสิทธิภาพ

นักออกแบบฐานข้อมูลขนาดใหญ่จึงกำหนดว่าตัวเลขเป็นตัววัดอย่างไร สำหรับความเป็นจริงของภาพรวมที่แสดงยอดขายในวันที่ระบุการวัดนี้มักจะแสดงจำนวนรายการที่ขายในระหว่างวันนั้น แต่ผู้ออกแบบอาจกำหนดมาตรการที่จะสะสม (กล่าวคือเติบโตขึ้นเท่านั้น) ซึ่งหมายความว่าการวัดนี้เป็นยอดขายรวมในวันนั้นและทุกวันก่อนหน้า

วัดค่าที่ต่างกันสำหรับชุดค่ามิติต่างๆกัน แท้จริงการวัดสามารถในแง่คณิตศาสตร์ได้รับการพิจารณาฟังก์ชันบางส่วนจากผลิตภัณฑ์คาร์ทีเซียนของมิติในก้อนไปยังชุดของตัวเลขเช่น N หรือ R คุณสมบัติและสูตรจะถูกเลือกเช่นว่าค่าของการวัดมีความหมายสำหรับทั้งหมด การ

รวมกันของระดับการรวม (รวมถึงระดับ "บนสุด" หรือไม่) สูตรถูกกำหนดไว้ในข้อมูลเมตาและดังนั้นจึงไม่ถูกจำลองแบบเช่นในตัวอย่างสเปรดชีต

แม้ว่ารูปแบบข้อมูลหลายมิติส่วนใหญ่มีมาตรการบางอย่างไม่ได้ ในมิติข้อมูลเหล่านี้ค่ามิติจะใช้สำหรับการคำนวณด้วยเช่นกันโดยไม่คำนึงถึงความจำเป็นในการวัด แต่ต้องเสียค่าใช้จ่ายในการใช้งาน

เป็นสิ่งสำคัญที่จะแยกความแตกต่างระหว่างสามชั้นของค่าวัดคือ additive, semi-additive และ non-additive มาตรการเนื่องจากเหล่านี้ทำงานค่อนข้างแตกต่างกันในการคำนวน

ค่าวัด additive สามารถรวมกันได้ตามความหมายทุกมิติ ตัวอย่างเช่นคุณควรเพิ่มยอดขายรวมทั้ง Book, Location และ Time เนื่องจากไม่ทำให้เกิดการซ้อนทับกับปรากฏการณ์ในโลกแห่งความเป็นจริงที่ทำให้เกิดค่านิยมของแต่ละบุคคล ค่าวัดเสริมที่เกิดขึ้นสำหรับชนิดของความเป็นจริงใด ๆ

ค่าการวัด Semi-additive ไม่สามารถรวมกันได้ตามมิติข้อมูลอย่างน้อยหนึ่งมิติส่วนใหญ่จะเป็นมิติข้อมูลเวลา Semi-addictive มักเกิดขึ้นเมื่อความเป็นจริงของภาพรวม ตัวอย่างเช่นการรวมหมวดพื้นที่โฆษณาในแต่ละช่วงเวลาไม่ได้ผลเนื่องจากรายการพื้นที่โฆษณาเดียวกันเช่นหนังสือเฉพาะอาจมีการนับหลายครั้ง แต่จะมีความหมายในการรวมระดับพื้นที่โฆษณาระหว่างหนังสือและร้านค้า

ค่าวัด Non-additive เป็นค่าวัดที่ไม่ตามมิติข้อมูลใด ๆ ได้เนื่องจากเป็นสูตรที่เลือกตัวอย่างเช่นกรณีนี้เกิดขึ้นเมื่อค่าเฉลี่ยสำหรับค่าระดับต่ำกว่าไม่สามารถรวมกันเป็นค่าเฉลี่ยสำหรับค่าระดับสูง ๆ ได้ ค่าวัด Non-additive ได้สามารถเกิดขึ้นได้สำหรับความจริงทุกชนิด

2.3.2.4 การนำเสนอด้วยฐานข้อมูลรีเรชันนอล (Relational Representations)

หลังจากที่ได้นำแนวคิดพื้นฐานมาใช้ในส่วนก่อนหน้าแล้วเราพร้อมที่จะพิจารณาวิธีการแสดงข้อมูลเหล่านี้ในฐานข้อมูลเชิงสัมพันธ์ โปรดทราบว่ามีการแสดงแนวคิดแบบหลายมิติมากกว่าความสัมพันธ์อื่น ๆ ด้วย รายละเอียดเพิ่มเติม relational representations จะถูกพิจารณาที่นี่เนื่องจากใช้กันอย่างแพร่หลายและง่ายต่อการเข้าใจ มีสองวิธีหลักในการแสดงมิติข้อมูลในฐานข้อมูลเชิงสัมพันธ์ เราอธิบายทั้งสองอย่าง

1) สกีมาดาว (Star Schema) สกีมาดาวมีตารางมิติข้อมูลหนึ่งมิติ ตารางนี้มีคอลัมน์หลักและหนึ่งคอลัมน์สำหรับแต่ละระดับของมิติข้อมูล (ยกเว้น T) คอลัมน์ระดับจะมีคำอธิบายเกี่ยวกับต้นฉบับของค่ามิติข้อมูลในระดับนั้น สุดท้ายตารางมิติข้อมูลประกอบด้วยคอลัมน์เดียวสำหรับคุณสมบัติแต่ละระดับในมิติข้อมูล นอกจากนี้สคีมาของดาวมีตารางความเป็นจริงซึ่งมีแถวสำหรับความเป็นจริงหลายมิติ ตารางความจริงมีหนึ่งคอลัมน์สำหรับแต่ละการวัด แถวคอลัมน์นี้มีค่าวัดสำหรับแถวที่เป็นจริง ตารางความจริงก็มีหนึ่งคอลัมน์สำหรับแต่ละมิติ แถว

คอลัมน์เหล่านี้มีค่าคีย์ต่างประเทศที่อ้างอิงค่าคีย์หลักของตารางมิติข้อมูล ตัวอย่างสกีมาดาวสำหรับคิวบ์การขายจะแสดงในรูปที่ 2.3 (Primary key จะขีดเส้นใต้ไว้) ชื่อ "สกีมาดาว" หมายถึงการสังเกตว่าถ้าตารางถูกวาดอย่างที่ตารางความเป็นจริงอยู่ตรงกลางและตารางด้านมิติรอบ ๆ รูปทรงนั้นจะมีลักษณะคล้ายกับดาว เพื่อช่วยให้เห็นภาพคำเปรียบเทียบดาวนี้จะมีการวาดเส้นระหว่างตารางความเป็นจริงและตารางแต่ละมิติในรูปที่ 2.3 แถวในตารางความจริงของสคีมาดาวที่แสดงในรูปที่ 2.3 แสดงจำนวนการขายสำหรับชุดข้อมูล "Book", "City" และ "Day" ตารางความเป็นจริงจะมีคอลัมน์คีย์ต่างประเทศสำหรับแต่ละคอลัมน์ มิติข้อมูลหนังสือ สถานที่และเวลา ตารางมิติข้อมูลมีคอลัมน์สำคัญที่เกี่ยวข้องเช่น "CityID" และหนึ่งคอลัมน์สำหรับแต่ละระดับเช่น "City" และ "State" สำหรับตารางที่แสดงมิติข้อมูลสถานที่ ไม่มีคอลัมน์ที่จำเป็นสำหรับ? ระดับเป็นคอลัมน์ที่มักจะถือค่าเดียวกัน คอลัมน์หลักในตารางมิติคือคีย์จำนวนเต็ม "โง่" โดยไม่มีความหมายใด ๆ นั่นคือคีย์ตัวแทน ข้อดีของการใช้คีย์ข้อมูลจากระบบซอร์สรวมถึงการใช้ที่เก็บข้อมูลที่ดีขึ้นการป้องกันปัญหาที่เกี่ยวกับการใช้ key ซ้ำ ช่วยสนับสนุนการอัปเดตมิติข้อมูลและการประมวลผลข้อความที่มีประสิทธิภาพมากขึ้น จะเห็นได้ว่ามีข้อมูลซ้ำซ้อนในข้อมูลระดับสูงขึ้น ตัวอย่างเช่น "มีนาคม 2009" จะปรากฏในแต่ละวันในเดือนนั้นซึ่งหมายความว่าคอลัมน์เดือนใช้ค่า "มีนาคม 2009" ใน 31 แถว (สมมติว่ามีการขายหนังสืออย่างน้อยหนึ่งเล่มในแต่ละวันของเดือนนั้น) วันของมันต้องแสดง) อย่างไรก็ตามเนื่องจากขนาดโดยทั่วไปจะใช้เวลาเพียง 1-5% ของพื้นที่จัดเก็บข้อมูลทั้งหมดที่จำเป็นสำหรับสคีสดาว แต่ความซ้ำซ้อนไม่เป็นปัญหาในแง่พื้นที่ ไม่ก่อให้เกิดปัญหาเรื่องการอัพเดทประสิทธิภาพ นอกจากนี้การอัปเดตมิติข้อมูลได้รับการจัดการโดยส่วนกลางแล้วจึงเป็นไปได้ที่จะทำให้เกิดความสม่ำเสมอ ดังนั้นจึงมักเป็นความคิดที่ดีที่จะใช้ตารางมิติข้อมูลที่ซ้ำซ้อนเหล่านี้เพราะการสนับสนุนเหล่านี้สนับสนุนการสร้างแบบสอบถามที่มีประสิทธิภาพ (และมีประสิทธิภาพดีกว่า) ที่ง่ายกว่าการทำซ้ำตามปกติ

Book (dimension table)

BookID	Book	Genre
7493	Tropical Food	Cooking
9436	Winnie the Pooh	Children's books

Sale (fact table)

BookID	CityID	DayID	Sale
9436	854	2475	20
7493	854	2475	5
7493	876	3456	2
9436	876	3456	11
9436	876	2475	18

Location (dimension table)

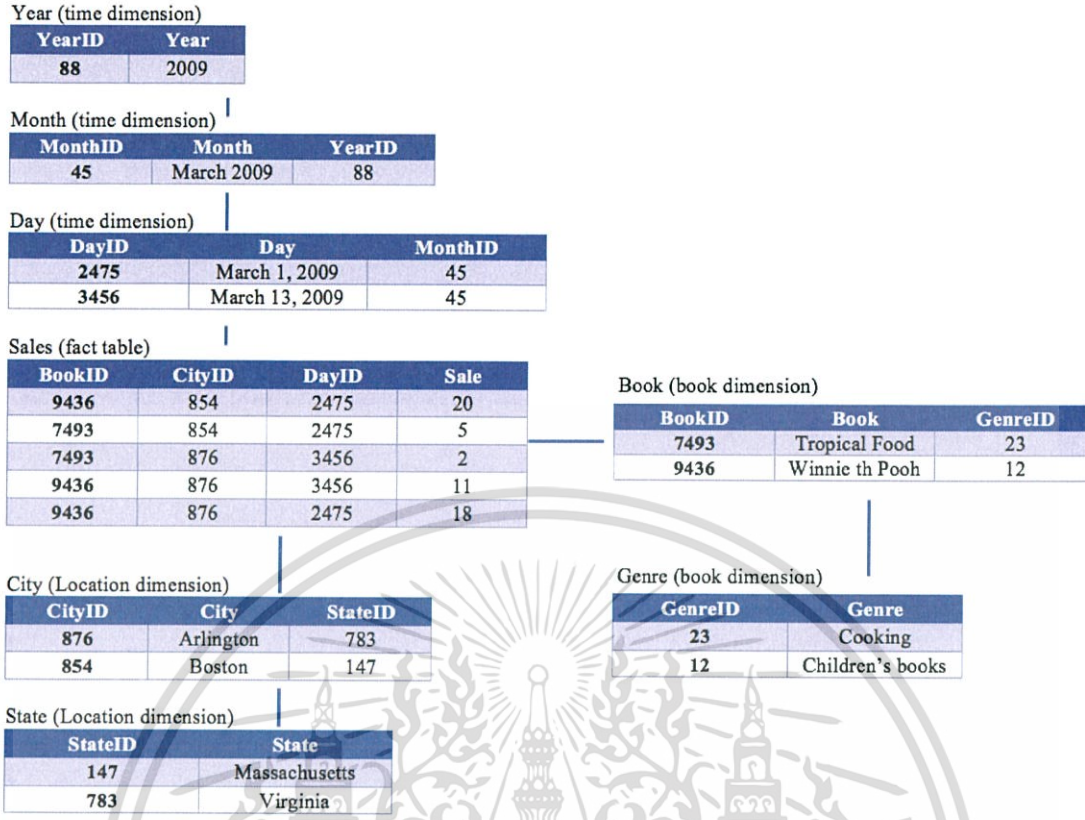
CityID	City	State
876	Arlington	Virginia
854	Boston	Massachusetts

Time (dimension table)

DayID	Day	Month	Year
2475	March 1, 2009	March 2009	2009
3456	March 13,2009	March 2009	2009

รูป 2.8 สกีมาดาว สำหรับ Sales cube

2) สกีมาเกล็ดหิมะ (Snowflake Schema) สกีมาเกล็ดหิมะมีตารางความจริงเช่นเดียวกับสกีมาดาว สกีมาสเกี่ยวกับเกล็ดหิมะมีตารางมิติข้อมูลหลายมิติสำหรับแต่ละมิติ ได้แก่ ตารางหนึ่งสำหรับแต่ละระดับ (ไม่ใช่ T) ซึ่งหมายความว่าการหลีกเลี่ยงความซ้ำซ้อนอาจเป็นประโยชน์ในบางสถานการณ์และทำให้มิติข้อมูลในมิติข้อมูลมีความชัดเจน ตารางมิติข้อมูลประกอบด้วยคีย์คอลัมน์ที่เก็บคำอธิบายต้นฉบับของค่าระดับและอาจเป็นคอลัมน์สำหรับคุณสมบัติระดับต่างๆตารางสำหรับระดับล่างจะมีคีย์ต่างประเทศที่มีอยู่ในระดับด้วย ถ้าตารางถูกวาดด้วยตารางความเป็นจริงในตารางมิติกลางและตารางที่เกี่ยวข้องซึ่งอยู่ถัดจากแต่ละรูปลักษณ์จะมีลักษณะคล้ายกับเกล็ดหิมะ ดังนั้นชื่อ "สกีมาเกล็ดหิมะ" รูป 2.8 แสดงตัวอย่างสกีมาเกล็ดหิมะที่เก็บข้อมูลเดียวกันกับสกีมาดาวในรูปที่ 2.8 ตัวอย่างเช่นตารางวันในรูป 2.9 ประกอบด้วย integer key, วันที่ และ foreign key ในตารางเดือน โปรดทราบว่าในสกีมานี้ค่าเดือนจะไม่ถูกทำซ้ำ อย่างไรก็ตามเนื่องจากยากที่จะค้นหาคีมาเนื่องจากต้องใช้การรวมกันหลายรายการ การรวมกันหลายครั้งทำให้ระบบการจัดการฐานข้อมูล (DBMS) ต้องใช้เวลาในการคำนวณผลการค้นหามากขึ้น การเลือกว่าจะใช้สคีมาดาวหรือสคีลเกล็ดหิมะขึ้นอยู่กับคุณสมบัติที่ต้องการของระบบที่กำลังพัฒนาอยู่หรือไม่ อันที่จริงแล้วคุณสามารถใช้การสโนว์เกิลได้เพียงบางส่วนกับสคีมาของดาวเท่านั้นทั้งมิติข้อมูลที่เลือกไว้ทั้งหมดหรือบางส่วนภายในมิติข้อมูล คำว่า starflake schema ถูกใช้กับสกีมา ที่เป็นผลลัพธ์ สำหรับความสั้นเราละเว้นการอธิบายแบบเต็มรูปแบบในด้านนี้



รูป 2.9 สคีมาเกล็ดหิมะ สำหรับ Sales cube

2.3.2.5 คลังข้อมูลและเดต้ามาร์ท (Data Warehouses and Data Marts)

ก่อนหน้านี้เราเรียกว่ากลุ่มก้อนข้อมูลที่เกี่ยวข้องกับคลังข้อมูล อย่างไรก็ตามมีข้อมูลมากกว่าที่จะพูดเกี่ยวกับคลังข้อมูล Bill Inmon ซึ่งเป็นหนึ่งในผู้บุกเบิกการจัดเก็บข้อมูลกำหนดคลังข้อมูลเป็นข้อมูลที่มุ่งเน้นการผสมผสานระหว่างเวลาการรวบรวมข้อมูลที่ไม่ทำให้เกิดความผันผวนของข้อมูลเพื่อสนับสนุนกระบวนการตัดสินใจของฝ่ายจัดการ ต่อไปนี้เราจะพิจารณาถึงความหมายขององค์ประกอบต่างๆของคำจำกัดความนี้ซึ่งแสดงถึงมุมมองที่สำคัญเกี่ยวกับแนวคิดของคลังข้อมูล

เริ่มต้นจากด้านหลังจะเห็นได้ว่าวัตถุประสงค์ของคลังข้อมูลคือการสนับสนุนการตัดสินใจ ระบบนี้แตกต่างจากจุดประสงค์ของระบบปฏิบัติการขององค์กรที่สนับสนุนกระบวนการทางธุรกิจในชีวิตประจำวัน (การจัดการคำสั่งซื้อการส่งมอบตั๋วเงิน ฯลฯ) แต่ไม่ใช่การวิเคราะห์เชิงกลยุทธ์และการตัดสินใจ โฟกัสที่แตกต่างกันนี้ส่งผลต่อข้อมูลที่เก็บในคลังข้อมูลและวิธีจัดเก็บข้อมูล ในคำนิยามของ Inmon คลังข้อมูลจะอยู่ภายใต้การกำหนด กล่าวอีกนัยหนึ่งคลังข้อมูลได้รับการออกแบบมาโดยรอบประเด็นสำคัญที่เกี่ยวข้องกับธุรกิจเพื่อให้สามารถวิเคราะห์ได้ง่าย สำหรับร้านค้าปลีกหนังสือซึ่งรวมถึง "การขาย" คลังข้อมูลของร้านค้าปลีกหนังสือจึงได้รับการออกแบบมาเพื่อขาย และเพื่อให้การวิเคราะห์ง่ายขึ้นการขายได้รับการอธิบายโดยใช้หน่วยงานเช่น "หนังสือ" และ "ร้านค้า" ในทางตรงกันข้ามฐานข้อมูลการดำเนินงานเป็นแบบจำลองเพื่อสนับสนุนการ

ดำเนินงานรายวันและได้รับอิทธิพลอย่างมากจากกระบวนการทำงานประจำวัน เช่นเวิร์กโฟลว์) ในธุรกิจและวิธีการประยุกต์ใช้งานที่ใช้พวกเขาทำงานร้านค้าปลีกหนังสืออาจมีการใช้งานที่แตกต่างกันสำหรับการสั่งซื้อหนังสือจากสำนักพิมพ์แจกจ่ายให้กับร้านค้าแคมเปญการตลาดและการบัญชีการวิเคราะห์ผลกำไรจากการขายในระหว่างแคมเปญนั้นจะต้องใช้นักวิเคราะห์เพื่อดึงข้อมูลจากระบบต่างๆที่จัดระเบียบข้อมูลในรูปแบบต่างๆและมีแนวโน้มที่จะให้ข้อมูลที่ไม่สอดคล้องกัน

นอกจากนี้ Inmon ยังระบุว่ามีการรวมคลังข้อมูลไว้ด้วยข้อมูลที่พบในคลังข้อมูลมักจะมาจากระบบปฏิบัติการต่างๆเช่นแคตตาล็อกหนังสือของผู้เผยแพร่หนังสือระบบการเก็บสต็อคและการลงทะเบียนเงินสดในร้านค้า แต่ละระบบอาจใช้รูปแบบที่แตกต่างกัน ตัวอย่างเช่นหนึ่งสามารถใช้รหัส "P" สำหรับปกอ่อนในขณะที่ระบบอื่นอาจใช้ "ปกอ่อน" และระบบที่สามอาจใช้รหัสตัวเลข "3. " ในคลังข้อมูลรหัสเฉพาะหรือคำอธิบายจะใช้เช่นว่า ผู้ใช้ไม่ได้รับสับสนโดยรหัสที่แตกต่างกันที่หมายถึงเดียวกันหรือตามรหัสที่คล้ายกันกับความหมายที่แตกต่างกันนอกจากนี้เรายังทราบว่าถือเป็นการปฏิบัติที่ไม่ดีมากที่จะใช้รหัสที่คลุมเครือในคลังข้อมูล ควรใช้คำอธิบายแบบข้อความที่เข้าใจง่ายไม่ย่อคำอธิบายแบบข้อความ (เช่นใช้ "หนังสือปกอ่อน" แทนคำว่า "P" หรือ "3") ระบบที่ต่างกันอาจใช้หน่วยต่าง ๆ เช่นแคตตาล็อกหนังสือของผู้จัดพิมพ์บางรายอาจถือมิติของหนังสือเป็นเซนติเมตรขณะที่อีกหนึ่งใช้นิ้ว ในคลังข้อมูลใช้หน่วยเดียวเท่านั้น เป็นกระบวนการที่เรียกว่า Extract-Transform-Load (ETL) เพื่อดึงข้อมูลจากระบบต้นทางที่แตกต่างกันทำความสะอาดข้อมูลและแปลงข้อมูลให้เป็นรูปแบบรวมก่อนที่จะโหลดข้อมูลลงในคลังข้อมูลเป็นส่วนสำคัญของโครงการคลังข้อมูลใด ๆ เพื่อสร้างกระบวนการ ETL ที่ทำงานได้

คลังข้อมูลเป็นตัวแปรเวลาซึ่งหมายความว่าคลังข้อมูลจะแสดงวิวัฒนาการตามเวลาไม่ใช่ข้อมูลล่าสุดเนื่องจากระบบปฏิบัติการมีแนวโน้มที่จะทำ กล่าวได้ว่าเป็นไปได้ที่จะมองเห็นโลกจำลองที่ดูเหมือนในเวลาใดเวลาหนึ่ง สำหรับร้านค้าปลีกหนังสือก็เป็นไปได้ที่จะเห็นทั้งจำนวนสำเนาของ "Winnie Pooh" ได้รับการขายในจำนวนรวมถึงตอนนี้และจำนวนสำเนาถูกขายโดยเวลาเดียวกันของปีที่แล้ว คลังข้อมูลจะสามารถรวบรวมการเปลี่ยนแปลงของโลกจำลองได้เช่นเมื่อร้านค้าปลีกหนังสือแห่งใดแห่งหนึ่งมีการขยายตัว เป็นไปได้ที่นักวิเคราะห์จะเห็นขนาดของร้านค้าที่จุดต่างๆในเวลาและตรวจสอบว่าการขยายตัวมีผลต่อยอดขายอย่างไรเรากล่าวเพิ่มเติมเกี่ยวกับการเปลี่ยนแปลงประเภทนี้ในบทถัดไปเราทราบว่าเกือบทุกคลังข้อมูลมี มิติเวลา

ในที่สุด Inmon ระบุว่าคลังข้อมูลไม่ระเหยซึ่งหมายความว่าจะไม่มีการลบหรืออัปเดตใด ๆ กับข้อมูลที่มีอยู่แล้วในคลังข้อมูล การเปลี่ยนแปลงเพียงอย่างเดียวเกิดจากการโหลดข้อมูลใหม่ ในทางตรงกันข้ามระบบปฏิบัติการได้รับการออกแบบมาเพื่อสนับสนุนการอัพเดต (พร้อมกัน) และต้องใช้การจัดการธุรกรรมขั้นสูงและการออกแบบที่ได้รับการรับรองมาตรฐานเพื่อให้มีการอัปเดตที่มีประสิทธิภาพและหลีกเลี่ยงการอัปเดตความผิดปกติ ในคลังข้อมูลไม่จำเป็นต้องมีการทำให้

เป็นบรรทัดฐาน (เรียกคืนสคีมาดาวจากส่วน 2.3.2.4) นอกจากนี้ระบบปฏิบัติการมักประกอบด้วยข้อมูลล่าสุดที่เป็นธรรมและไม่สนับสนุนการวิเคราะห์ข้อมูลในอดีต

"ดาต้ามาร์ท" โดยทั่วไปถือว่าเป็นเซตย่อยของคลังข้อมูลขณะที่คลังข้อมูลใช้ร่วมกันทั่วทั้งองค์กรและสามารถเก็บข้อมูลเกี่ยวกับหัวข้อต่างๆได้ข้อมูลเหล่านี้จะเก็บข้อมูลเกี่ยวกับเรื่องเดียวเช่นการขาย บ่อยครั้งที่ข้อมูลมาร์ทเป็นผู้เชี่ยวชาญเฉพาะต่อความต้องการของแผนก (ย่อย) ดังนั้นคลังข้อมูลขององค์กรสามารถมีก้อนและขนาดสำหรับทั้งยอดขายของร้านค้าการขายทางอินเทอร์เน็ตการเก็บสต็อคและการจัดเก็บข้อมูลพนักงานขณะที่มีการสร้างข้อมูลเพื่อเก็บข้อมูลเกี่ยวกับการขายซึ่งเป็นกระบวนการเดียวที่ฝ่ายขายสนใจเท่านั้น ในทำนองเดียวกันข้อมูลอื่น ๆ สามารถสร้างขึ้นเพื่อเก็บรักษาข้อมูลเกี่ยวกับพนักงานเท่านั้นซึ่งเป็นที่สนใจของแผนกทรัพยากรบุคคล อย่างไรก็ตามโปรดทราบว่าแผนกต่างๆไม่ควรมีข้อมูลส่วนตัวของตนเอง ถ้าหน่วยงานสองคนหรือมากกว่าต้องใช้ข้อมูลการขายทั้งสองควรใช้ข้อมูลเดียวกันกับศูนย์ข้อมูลการขาย คลังข้อมูลจะถูกโหลดด้วยข้อมูลจากระบบต้นทางการทำงานที่แตกต่างกันในขณะที่ข้อมูลดาต้ามาร์ทเต็มไปด้วยข้อมูล (ที่ทำความสะอาดข้อมูลแล้ว) จากคลังข้อมูลหรือแม้แต่ให้มุมมองเชิงตรรกะของข้อมูลในคลังข้อมูล

แต่น่าเสียดายที่มีความไม่ลงรอยกันในงานเขียนเกี่ยวกับความสัมพันธ์ระหว่างคลังข้อมูลและดาต้ามาร์ทที่ควรจะเป็น บุคคลบางคนสนับสนุนมุมมองที่ว่าคลังข้อมูลประกอบด้วยกลุ่มดาต้ามาร์ท มุมมองนี้แสดงให้เห็นถึงวิธีการจากล่างขึ้นบนที่คลังข้อมูลสร้างโดยการสร้างดาต้ามาร์ทจากนั้นจึงรวมข้อมูลเหล่านี้เพื่อรับคลังข้อมูล นี่เป็นคำตรงกันข้ามกับคำจำกัดความที่กล่าวมาข้างต้นซึ่งดาต้ามาร์ทจะได้รับมาจากคลังข้อมูลตามลักษณะจากบนลงล่าง Inmon สนับสนุนแนวทางด้านบนลงล่างอย่างมาก ท่ามกลางปัญหาอื่น ๆ เกี่ยวกับวิธีการจากล่างขึ้นบนเขาชี้ให้เห็นถึงการขาดการรวมข้อมูลที่สำคัญเข้าด้วยกัน

ปัญหาสำคัญที่เกิดขึ้นจากวิธีการจากบนลงล่างคือความยากลำบากในกระบวนการพัฒนาอาจเป็นเรื่องยากมากที่จะสร้างคลังข้อมูลในลักษณะที่อยู่บนลงล่างสำหรับองค์กรขนาดใหญ่ เป็นผลให้กระบวนการอาจมีความยาวและผลตอบแทนจากการลงทุนช้า

ในสิ่งที่เราอาจมองว่าเป็นความพยายามในการได้รับสิ่งที่ดีที่สุดในโลกทั้งสอง Kimball ผู้บุกเบิกคลังข้อมูลมิติแนะนำแนวคิดเกี่ยวกับมิติข้อมูลและข้อเท็จจริงที่มีการกำหนดรูปแบบ ที่นี่องค์กรมาตรฐานข้อเท็จจริงและมิติที่จะใช้ร่วมกันในองค์กร ซึ่งจะช่วยให้กระบวนการมีทั้งด้านบนและด้านล่างขึ้นและทำให้มันเป็นไปได้ที่จะรวมหลาย marts หรือก้อนในรูปแบบที่สอดคล้องกัน ส่วนถัดไปครอบคลุมกระบวนการสร้างแบบจำลองในรายละเอียดเพิ่มเติม

2.3.2.6 การวิเคราะห์และการถามคำถาม (Analysis and Querying)

1) Roll-up คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้สูงขึ้นเพื่อทำการสรุปข้อมูลหรือดูข้อมูลในภาพรวม

2) Drill-down คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้ต่ำลงเพื่อเป็นการดูรายละเอียดของข้อมูล

3) Drill-out คือเพิ่มมุมองที่ต้องการวิเคราะห์เพื่อเป็นการดูรายละเอียดของข้อมูลในมุมมองอื่นประกอบการวิเคราะห์

4) Slice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองเพียงมุมมองเดียว

5) Dice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองหลายมุมมอง

6) Drill-Across เกิดจากการรวมกันของข้อมูลของ cube 2 อัน กล่าวคือ การวิเคราะห์ข้อมูลบางครั้งไม่สามารถใช้เพียง cube เดียวแล้ววิเคราะห์ได้ จึงต้องมีการใช้ cube อีกอันซึ่งมี conformed dimensions มาช่วยในการวิเคราะห์

7) Pivot table คือตารางที่แสดงข้อมูลโดยใช้มุมมอง 2 มุมมอง โดยจะเป็นการแสดงค่าผลรวมย่อยและค่าผลรวมทั้งหมด ซึ่งอนุญาตให้ทำการ roll-up และ drill-down ได้และสามารถมีหลายๆมุมมองใน 1 แกนของตารางได้

8) Ranking คือการนำข้อมูลที่ต้องการวิเคราะห์มาจัดอันดับ

2.3.3 แนวคิดขั้นสูง (Advanced Concepts)

2.3.3.1 Slowly Changing Dimensions

เช่นเดียวกับฐานข้อมูลใด ๆ ฐานข้อมูลแบบหลายมิติหรือแบบจำลองคลังข้อมูลที่เลือกลักษณะของความเป็นจริงบางประการซึ่งการจับภาพเฉพาะเจาะจงขึ้นอยู่กับการใช้งานที่ต้องการของฐานข้อมูล เนื่องจากลักษณะของความเป็นจริงความเป็นจริงและความเป็นจริงของการใช้คลังข้อมูลจะเปลี่ยนไปตามกาลเวลา นี่อนุมานได้ว่าฐานข้อมูลจะต้องสามารถพัฒนาขึ้นได้เพื่อที่จะให้บริการตามวัตถุประสงค์ต่อไปได้ ในส่วนนี้เราจะพิจารณาการจัดการการเปลี่ยนแปลงในฐานข้อมูลที่แสดงโดยใช้สคีมาดาว(Star Schema) ตามที่อธิบายไว้ก่อนหน้านี้ ฐานข้อมูลดังกล่าวมีตารางความจริงกลางและตารางมิติข้อมูล(Dimension table)

1) The Problem (ปัญหา) คลังข้อมูลได้จำลองกระบวนการในโลกแห่งความจริงบางอย่างที่เราสนใจศึกษา ตัวอย่างที่เราพิจารณาจะมุ่งเน้นไปที่ขั้นตอนการขายหนังสือ เมื่อกระบวนการวิวัฒนาการแถวใหม่จะถูกแทรกลงในตารางความเป็นจริงและในตารางมิติข้อมูล ในตัวอย่างของเราในรูป 2.3 เมื่อยอดขายเข้ามาในวันใหม่วันนั้นจะแทรกเป็นแถวในตารางมิติข้อมูลเวลาและแถวจะแทรกลงในตารางข้อเท็จจริงการขายสำหรับการรวมกันของหนังสือและเมืองที่มี มีการขายอย่างน้อยหนึ่งวันในระหว่างนั้น หากหนังสือเล่มใหม่เริ่มจำหน่ายหนังสือเล่มนั้นจะถูกแทรกลงในตารางมิติข้อมูลหนังสือและหากยอดขายเริ่มต้นขึ้นในเมืองใหม่เมืองนั้นจะถูกแทรกลงในตารางมิติข้อมูลตำแหน่ง สถานการณ์นี้แสดงถึงวิวัฒนาการที่ต้องการของคลังข้อมูล อย่างไรก็ตามในทางปฏิบัติจำเป็นที่จะต้องสามารถรับมือ

กับการเปลี่ยนแปลงประเภทอื่น ๆ ซึ่งรวมถึงมิติข้อมูลที่เรียกว่าการเปลี่ยนแปลงอย่างช้าๆซึ่งเกิดขึ้นเมื่อต้องมีการอัปเดตแถวที่มีอยู่ในตารางมิติข้อมูล ภาพจิตที่ตั้งใจโดยการตั้งชื่อนี้ก็คือถึงแม้ว่าแถวที่มีอยู่ในตารางมิติข้อมูลจะต้องได้รับการอัปเดตเป็นครั้งคราวสิ่งนี้เกิดขึ้นนาน ๆ ครั้ง พิจารณา star schema โดยใช้ตารางมิติข้อมูลหนังสือที่ปรับปรุงใหม่ซึ่งแสดงในรูปที่ 3.1 ตารางนี้มีคอลัมน์ Rating ที่ระบุว่าลูกค้าชอบหนังสือมากน้อยเพียงใด ทั้งการให้คะแนนและประเภทของหนังสืออาจเปลี่ยนแปลงไปตามช่วงเวลา ตัวอย่างเช่นการจัดอันดับของ "GoneWith theWind" อาจลดลงเป็น "3 ดาว" หรือประเภทของหนังสือ "อาหารอิตาเลียน" อาจได้รับการขัดเกลาเพื่อ "การปรุงอาหารแบบเมดิเตอร์เรเนียน" ปัญหาเบื้องต้นเกี่ยวกับการอัปเดตแถวก็คือแถวเหล่านี้ ได้รับการอ้างถึงโดยแถวตารางความจริงกับยอดขายเก่า แถวตารางข้อเท็จจริงที่มีอยู่แล้วนี้พึ่งพาแถวของตารางมิติข้อมูลที่มีค่าแอตทริบิวต์เฉพาะเมื่ออัปเดตค่าแอตทริบิวต์ดังกล่าวแถวตารางข้อมูลจริงจะอ้างถึงแถวมิติข้อมูลตารางที่มีการเปลี่ยนแปลง ตัวอย่างเช่นการซื้อ "GoneWith theWind" เมื่อได้รับการจัดอันดับว่าเป็น "4 ดาว" ตอนนี้ถือว่าเป็นการซื้อหนังสือที่มีเพียง "3 ดาว" เท่านั้น ด้วยวิธีนี้ข้อมูลที่ไม่ถูกต้องถูกสร้างขึ้นต่อไป ปัญหาเบื้องต้นเกี่ยวกับการไม่เปลี่ยนแถวเพื่อแสดงการเปลี่ยนแปลงการให้คะแนนและการจำแนกประเภทคือคลังข้อมูลล้าสมัย แถวใหม่ที่ป้อนลงในตารางจริงต้องอ้างอิงแถวตารางมิติข้อมูลหนังสือที่ปรับปรุงแล้ว สังเกตว่าการปรับปรุงประเภทของแนวอาหาร "Tropical Food" จาก "หนังสือเด็ก" เป็น "การทำอาหาร" เป็นที่ยอมรับได้โดยการเปลี่ยนแปลงนี้คือการแก้ไขข้อผิดพลาด

Book (dimension)

BookID	Book	Rating	Genre
7493	Tropical Food	4 stars	Children's books
9436	Winnie the Pooh	5 stars	Children's books
9948	Gone With the Wind	4 stars	Fiction
9967	Italian Food	4 stars	Cooking

รูป 2.10 แก้ไขตารางมิติข้อมูลหนังสือสำหรับดิวบ์ของการขาย

2) Solution (วิธีแก้ปัญหา) พิจารณาแนวทางสามวิธีในการแก้ไขการเปลี่ยนแปลงที่อาจเกิดขึ้นในแถวตารางมิติข้อมูล วิธีแรกคือเพียงแค่เขียนทับค่าแอตทริบิวต์เก่าเท่านั้น นี่เรียกว่าการอัพเดตประเภท 1 ตามที่ได้กล่าวมาแล้วแถวตารางความจริงเก่านี้อ้างถึงแถวมิติข้อมูลตารางที่มีการเปลี่ยนแปลงและถ้าสถานะเดิมของคลังข้อมูลถูกต้องคลังข้อมูลจะมีข้อมูลที่ไม่ถูกต้อง ข่าวดีก็คือแถวตารางข้อมูลใหม่จะสามารถอ้างอิงถึงแถวตารางมิติข้อมูลที่ถูกต้องได้ วิธีนี้ใช้งานง่ายและหาก

ตารางข้อมูลอัปเดตเป็นเพียงการแก้ไขข้อผิดพลาดเท่านั้นเช่นในกรณีที่ "Tropical Food" จัดอยู่ในประเภทหนังสือสำหรับเด็กวิธีนี้เหมาะอย่างยิ่ง นอกจากนี้อาจมีบางกรณีที่ความไม่ถูกต้องที่นำมาใช้โดยการอัปเดตถือเป็นความไม่สำคัญ ตัวอย่างเช่นนี่อาจเป็นกรณีสำหรับการปรับแต่งประเภท อย่างไรก็ตามบรรทัดล่างคือว่าวิธีนี้โดยทั่วไปไม่สนใจปัญหาพื้นฐาน วิธีที่สองเรียกว่าการปรับปรุงประเภทที่ 2 (type 2 update) คือการจัดเรียงแถวในตารางมิติข้อมูล การเปลี่ยนแปลงจะถูกจับโดยการแทรกแถวใหม่ที่มีค่าแอตทริบิวต์ที่อัปเดตออกจากแถวที่มีอยู่ซึ่งยังไม่ได้แก้ไข นี่มีผลต่อความคิดที่ว่าตารางมิติข้อมูลจะไปจากการบันทึกแถวไปยังการบันทึก "เวอร์ชันของแถว" คอลัมน์คีย์หลักจะต้องมีลักษณะทั่วไปเพื่อจับภาพแถวของแถวแทนที่จะเป็นแถว คำอธิบายลักษณะนี้เป็นสิ่งที่ตรงไปตรงมาเมื่อมีการใช้คีย์แทนตามวิธีที่แนะนำ ด้วยวิธีนี้ตารางแถวความเป็นจริงเก่าจะอ้างถึงแถวตารางมิติข้อมูลเดิมต่อไปในขณะที่แถวตารางความเป็นจริงใหม่อ้างถึงแถวตารางมิติข้อมูลใหม่ ในตัวอย่างของเราหากการจัดอันดับของ "GoneWith theWind" เปลี่ยนไปเราจะสร้างแถวใหม่สำหรับหนังสือเล่มนี้ซึ่งแตกต่างจากแถวที่มีอยู่เฉพาะกับค่าจัดอันดับและการขาย "GoneWith theWind" ใหม่จะอ้างอิงถึงแถวใหม่นี้ วิธีนี้ช่วยให้สามารถจับภาพข้อมูลที่ถูกต้องในคลังข้อมูล มีผลต่อขนาดตารางที่เพิ่มขึ้น โดยทั่วไปแล้วปัญหานี้มักไม่ใช่ปัญหาเนื่องจากช่องว่างที่ใช้โดยมิติข้อมูลมักมีขนาดเล็กกว่าพื้นที่ที่ใช้ในการจัดเก็บความเป็นจริงเมื่อตารางมิติข้อมูลเปลี่ยนไปตามช่วงเวลาอาจมีความเกี่ยวข้องในการจับภาพการเปลี่ยนแปลงเหล่านี้ได้อย่างถูกต้อง ในกรณีที่มีมิติเวลาอยู่ซึ่งโดยปกติจะเป็นกรณีในคลังข้อมูลครั้งแรกที่แถวตารางความเป็นจริงหมายถึงแถวในตารางมิติสามารถกำหนดได้และคราวนี้สามารถใช้เป็นขอบเขตอนุรักษ์เมื่อมิติ แถวโต๊ะเข้ามามีชีวิตอยู่ เวลาที่แน่นอนเมื่อมีการเปลี่ยนแปลงเกิดขึ้นในมิติข้อมูลสามารถบันทึกได้โดยการแทรกแถวพิเศษในตารางข้อเท็จจริง แถวนี้อ้างอิงถึงแถวมิติใหม่และแถวในตารางมิติข้อมูลเวลาที่แสดงถึงช่วงเวลาที่เกิดการเปลี่ยนแปลง อย่างไรก็ตามนี่เป็นเรื่องยุ่งยากที่จะใช้ในการสืบค้น ทางเลือกที่ง่ายที่สุดก็คือการนำคอลัมน์ที่มีมูลค่าเท่า ๆ กันสองคอลัมน์ในมิติที่อาจมีการเปลี่ยนแปลง คอลัมน์เหล่านี้จะบันทึกเวลาที่แถวนั้นมีค่าและไม่ถูกต้องตามลำดับ โปรดทราบว่าสำหรับเวอร์ชันที่ยังคงใช้งานอยู่เราสามารถให้คอลัมน์บอกเวลาที่จะกลายเป็นประเด็นที่ไม่ถูกต้องในบางวันได้ในอนาคต สามารถใช้ค่า NULL ได้ แต่จะทำให้การสอบถามยากขึ้น โซลูชันสามารถขยายด้วยคอลัมน์ที่แต่ละแถวบอกว่าแถวนั้นเป็นแถวล่าสุดหรือไม่ คอลัมน์นี้ได้รับการอัปเดตแล้วในเวอร์ชันเก่าเมื่อแทรกเวอร์ชันใหม่ คอลัมน์สามารถเพิ่ม

เพื่อบันทึกหมายเลขเวอร์ชันของแต่ละแถว โปรดทราบว่าทั้งสองคอลัมน์หลัง (การบันทึกหากเป็นเวอร์ชันล่าสุดและหมายเลขเวอร์ชันตามลำดับ) ในหลักการสามารถอนุมานได้จากคอลัมน์สองคอลัมน์ที่มีค่าเวลา (การบันทึกเมื่อเวอร์ชันถูกต้องหรือไม่ถูกต้องตามลำดับ) อย่างไรก็ตามแบบสอบถามบางเรื่องสามารถกำหนดได้ง่ายขึ้นเมื่อคอลัมน์สองคอลัมน์หลังมีอยู่ในตารางมิติข้อมูล รูปที่ 2.11 แสดงตารางมิติข้อมูลหนังสือเมื่อใช้รูปแบบนี้

Book (dimension)

BookID	Book	Rating	Genre	ValidFrom	ValidTo	Newest	Version
7493	Tropical Food	4 stars	Children's books	2006-03-01	2008-12-31	No	1
9436	Winnie the Pooh	5 stars	Children's books	2000-01-01	9999-12-31	Yes	1
9948	Gone With the Wind	4 stars	Fiction	1999-06-01	2008-10-15	No	1
9967	Italian Food	4 stars	Cooking	2003-04-05	2009-05-01	No	1
9995	Gone With the Wind	3 stars	Fiction	2008-10-16	9999-12-31	Yes	2
10100	Tropical Food	4 stars	Cooking	2009-01-01	9999-12-31	Yes	2
11319	Italian Food	4 stars	Mediterranean cooking	2009-05-02	9999-12-31	Yes	2

รูป 2.11 ตารางมิติข้อมูลหนังสือพร้อมเวอร์ชันของแถว ("Type 2 Updates")

วิธีที่สามเพื่อสนับสนุนมิติการเปลี่ยนแปลงอย่างช้าๆแตกต่างกันมาก สำหรับคอลัมน์ตารางมิติข้อมูลแต่ละคอลัมน์ที่อาจมีการเปลี่ยนแปลงจะมีการแนะนำคอลัมน์รุ่นเพิ่มเติม นี่เรียกว่าการปรับปรุงประเภท 3 (Type 3 update) มีผลทำให้เราสามารถบันทึกค่าสองค่าสำหรับคอลัมน์ที่อาจมีการเปลี่ยนแปลงได้ เราใช้ค่าหนึ่งเพื่อบันทึกค่าปัจจุบันและเราใช้ค่าอื่นเพื่อบันทึกค่าปัจจุบันก่อนหน้านี้ เมื่อรูปแบบนี้ใช้กับตัวอย่างของเราเราจะได้ตารางมิติข้อมูลดังแสดงในรูปที่ 2.12

Book (dimension)

BookID	Book	Rating	OldRating	Genre	OldGenre
7493	Tropical Food	4 stars	4 stars	Cooking	Children's books
9436	Winnie the Pooh	5 stars	5 stars	Children's books	Children's books
9948	Gone With the Wind	3 stars	4 stars	Fiction	Fiction
9967	Italian Food	4 stars	4 stars	Mediterranean cooking	Cooking

รูป 2.12 ตารางมิติข้อมูลหนังสือพร้อมคอลัมน์ที่จัดรูปแบบ ("type 3 updates")

ด้วยวิธีนี้แถวตารางความเป็นจริงก่อนและหลังการเปลี่ยนแปลงล่าสุดของแอตทริบิวต์อ้างถึงแถวตารางมิติข้อมูลเดียวกัน และเนื่องจากแถวนี้เก็บค่าปัจจุบันและค่าล่าสุดก่อนหน้าปัจจุบันตารางแถวข้อเท็จจริงจะอ้างอิงโดยตรงกับค่าสองค่าของแอตทริบิวต์เดียวกัน ข้อตกลงนี้อาจใช้เพื่อวิเคราะห์ข้อมูลในการเปลี่ยนแปลง ตัวอย่างเช่นเราสามารถศึกษาความแตกต่างในการแจกจ่ายยอดขายในการจำแนกประเภทของหนังสือทั้งสองประเภทเพื่อเตรียมพร้อมสำหรับการตัดสินใจว่าจะนำหนังสือประเภทใหม่ที่มีประโยชน์สำหรับการวิเคราะห์หรือไม่ วิธีนี้ถูก จำกัด ด้วยความสามารถในการจับภาพเพียงสองค่าสำหรับแต่ละแอตทริบิวต์ที่สามารถเปลี่ยนแปลงได้และวิธีนี้ทำให้การใช้งานมีประโยชน์สำหรับบางกรณีเท่านั้น

นอกจากนี้ควรสังเกตว่าไม่สามารถจับภาพเมื่อมีการเปลี่ยนแปลงเกิดขึ้นสำหรับแอตทริบิวต์ อย่างไรก็ตามวิธีการดังกล่าวอาจได้รับการสนับสนุนโดยทั่วไปเพื่อสนับสนุนกรณีที่มีการเปลี่ยนแปลงเกิดขึ้นสำหรับสมาชิกทุกมิติในเวลาที่กำหนด ตัวอย่างเช่นย่านขายอาจถูกกำหนดใหม่สำหรับแต่ละปี ในมิติข้อมูลร้านค้าอาจมีแอตทริบิวต์สำหรับแต่ละปีที่แสดงเขตการขายสำหรับปีนั้น เมื่อนำมาใช้แนวทางประเภท 2 หรือ 3 เพื่อให้สามารถสะท้อนถึงการเปลี่ยนแปลงในโลกแห่งความเป็นจริงได้มักรวมการใช้วิธีแก้ไขปัญหาประเภท 1 สำหรับการแก้ไขข้อผิดพลาด ในรูป 2.11 เราได้สร้างเวอร์ชันแถวใหม่สำหรับหนังสือ "Tropical Food" ซึ่งถูกกำหนดขึ้นโดยไม่สมควรให้เป็นหนังสือสำหรับเด็ก การสร้างเวอร์ชันใหม่จะแก้ไขปัญหาเฉพาะจากจุดที่เวอร์ชันใหม่มีผล; ข้อเท็จจริงเก่าที่ไม่ได้อ้างถึงเวอร์ชันใหม่ยังอ้างถึงข้อมูลที่ไม่ถูกต้อง ทางออกที่ดีกว่าสำหรับข้อผิดพลาดนี้ก็คือเปลี่ยนค่าประเภทที่ไม่ถูกต้องสำหรับหนังสือ เราพิจารณาเฉพาะสตาร์สตาร์ในส่วนนี้ ในรูปแบบเกล็ดหิมะอาจมีการเปลี่ยนแปลงขนาดช้าๆเช่นกัน วิธีที่ 1 และประเภท 3 สามารถใช้งานได้อย่างตรงไปตรงมา นอกจากนี้ยังสามารถใช้วิธีการประเภทที่ 2 ได้ด้วย แต่เพื่อหลีกเลี่ยงสถานการณ์ที่ซับซ้อนเกินไปควรวางเวอร์ชันแถวไว้ในตารางมิติข้อมูลสำหรับระดับต่ำที่สุดดังนั้นหากเวอร์ชันใหม่ถูกสร้างขึ้นในระดับที่สูงกว่าเวอร์ชันใหม่ที่สอดคล้องกัน ควรจะสร้างขึ้นสำหรับระดับล่าง เราสรุปได้โดยการตั้งข้อสังเกตว่าเทคนิคที่ยืดหยุ่นที่สุดสำหรับการจัดการมิติที่เปลี่ยนแปลงช้าคือวิธีการกำหนดแถวแถว ("type 2") และแนะนำให้ใช้โดยทั่วไป

2.3.3.2 Other Special Kinds of Dimensions

1) Minidimesions ในส่วนของ Slow Changing Dimension ถ้าเราพิจารณาการเปลี่ยนไปอย่างรวดเร็ว จะมีวิธีแก้ไขกรณีดังกล่าว อย่างไร สมมติว่าผู้ค้าปลีกหนังสือของเราสามารถเข้าถึงข้อมูลรายละเอียดเกี่ยวกับลูกค้าที่เข้าร่วมโปรแกรมความภักดีได้ข้อมูลรายละเอียดนี้ประกอบด้วยที่อยู่วันเกิด สถานภาพสมรสจำนวนบุตรการศึกษาประเภทงานเงินเดือนรายปี ฯลฯ เมื่อสมาชิกของโปรแกรมความภักดีซื้อ ผลิตภัณฑ์จากร้านค้าจะแสดงบัตรสมาชิกเพื่อรับส่วนลด ข้อตกลงนี้ทำให้ผู้ค้าปลีกหนังสือสามารถติดตามการซื้อสินค้าของสมาชิกได้ในรายละเอียดมาก ร้านค้าปลีกหนังสือสามารถรวมข้อมูลเกี่ยวกับสมาชิกรายบุคคลแต่ละรายในฐานข้อมูลแบบหลายมิติด้วยมิติข้อมูลลูกค้า ด้วยวิธีนี้ผู้จำหน่ายหนังสือสามารถวิเคราะห์ว่ายอดขายหนังสือเกี่ยวข้องกับอายุระดับการศึกษา ฯลฯ เราถือว่าร้านค้าปลีกมีคุณลักษณะที่อธิบายถึงลูกค้า 30 ราย อย่างไรก็ตามข้อมูลลูกค้าจะเปลี่ยนไปตามเวลาเช่นเมื่อลูกค้าย้ายหรือได้รับการยกระดับ

ดังนั้นขนาดของลูกค้าจึงเป็นมิติที่เปลี่ยนแปลงไปอย่างช้าๆผู้ค้าปลีกสามารถใช้การอัปเดตประเภท 2 ที่มีการสร้างเวอร์ชันใหม่เพื่อติดตามการเปลี่ยนแปลงเหล่านี้ เวอร์ชันใหม่จะถูกสร้างขึ้นเมื่อแอตทริบิวต์ใด ๆ ได้รับค่าใหม่อย่างน้อยหนึ่งค่า ดังนั้นมิติข้อมูลลูกค้าจะมีเวอร์ชันมูลค่าเป็นจำนวนมาก (หรือเวอร์ชันแถวในระบบ Relational OLAP) เมื่อเปลี่ยนมิติข้อมูลเป็นประจำและเราใช้การกำหนดเวอร์ชันแถวผลก็คือขนาดของมิติข้อมูลอาจมีขนาดใหญ่มาก ซึ่งอาจเกิดขึ้นในตัวอย่างข้างต้น ยังคงมีการจัดทำดัชนีที่เหมาะสมโดยทั่วไปแล้วจะเป็นไปได้ในการจัดการขนาดใหญ่ที่มีการเปลี่ยนแปลงอย่างรวดเร็วขนาด ในกรณีพิเศษที่การทำให้เวอร์ชันมีขนาดใหญ่เกินไปวิธีที่สามารถทำได้คือการแบ่งมิติออกเป็นสองส่วน (ใน OLAP เชิงสัมพันธ์ทั้งสองตารางมิติจะถูกอ้างอิงจากตารางข้อเท็จจริง) ความคิดคือการทำลายคุณลักษณะที่เปลี่ยนแปลงบ่อยๆและวางไว้ในมิติใหม่เรียกว่า minidimension minidimension สามารถถือค่าผสมที่เป็นไปได้ทั้งหมดของค่าของแอตทริบิวต์ที่เปลี่ยนแปลงบ่อย ที่นี่ "ชุดค่าผสมที่เป็นไปได้ทั้งหมด" อาจหมายถึงผลิตภัณฑ์ Cartesian ของค่าแอตทริบิวต์นั่นคือชุดค่าผสมทั้งหมดที่เป็นไปได้จากมุมมองทางทฤษฎีหรืออาจหมายถึงชุดค่าผสมทั้งหมดที่เกิดขึ้นจริงในชีวิตจริง ขนาดของชุดหลังเป็นเพียงเศษเสี้ยวของขนาดของผลคูณ Cartesian เท่านั้น เพื่อลดขนาดของ minidimension ค่าแอตทริบิวต์รายละเอียดอาจถูกแทนที่ด้วยช่วงของค่า ตัวอย่างเช่นหากมิติข้อมูลลูกค้ามีการบันทึกรายได้เฉลี่ยต่อปีของลูกค้ารายได้ขั้นต่ำของกลุ่มลูกค้าใหม่อาจบันทึกรายได้ต่อปีโดยใช้ตัวอย่างเช่น 5 ช่วง ซึ่งจะ จำกัด จำนวนค่าที่เป็นไปได้ใน minidimension ในตัวอย่างของเราผู้ค้าปลีกสามารถสร้าง minidimension โปรไฟล์ซึ่งอธิบายถึงโปรไฟล์ลูกค้า (ไม่ใช่ลูกค้ารายบุคคล) และมีคุณลักษณะทั้งหมดที่เปลี่ยนแปลงบ่อยๆเช่นเงินเดือนจำนวนบุตรและการจัดอันดับเครดิตในขณะที่ (เช่นการศึกษาและวันเกิด) และคำอธิบายเฉพาะลูกค้าแต่ละรายเช่นที่อยู่ยังอยู่ในมิติลูกค้า เนื่องจากการผสมผสานที่เป็นไปได้ทั้งหมดจะแสดงใน minidimension ไม่จำเป็นต้องมีการจัดการการเปลี่ยนแปลงใน minidimension แทนการเปลี่ยนแปลงในโลกจำลองจะถูกจับโดยข้อเท็จจริง ตัวอย่างเช่นถ้าลูกค้าได้รับเงินเดือนที่สูงขึ้นคุณไม่จำเป็นต้องอัปเดตอะไรใน minidimension ที่เกี่ยวข้อง แทนที่จะมีการเปลี่ยนแปลงในครั้งต่อไปที่เกิดขึ้นจริงสำหรับลูกค้าที่กำหนด ความเป็นจริงใหม่จะอ้างอิงค่ามิติข้อมูล minidimension ที่ครอบคลุมเงินเดือนแล้ว ผลกระทบเชิงลบของการใช้ minidimension คือข้อมูลจะกลายเป็นรายละเอียดน้อยลงและการสอบถามบางอย่างอาจกลายเป็นเรื่องยากที่จะกำหนดขึ้นเนื่องจากคุณลักษณะที่พวกเขาอ้าง

ถึงมีอยู่ในมิติข้อมูลมากกว่าสองมิติ ในการใช้ OLAP เชิงสัมพันธ์ตารางความเป็นจริงจำเป็นต้องมีคอลัมน์คีย์ต่างประเทศพิเศษ เนื่องจากคลังข้อมูลมักมีจำนวนจริงจำนวนแถวจึงต้องมีพื้นที่เพิ่มขึ้นไม่น้อย ผลอีกอย่างหนึ่งคือการเปลี่ยนแปลงบางอย่างสามารถจับภาพได้เฉพาะเมื่อมีข้อเท็จจริงเกิดขึ้นด้วยมิติที่เปลี่ยนแปลงไปอย่างช้าๆคุณสามารถจับภาพการเปลี่ยนแปลงเช่นข้อมูลประชากรเกี่ยวกับเงินเดือนของลูกค้าในวันที่การเปลี่ยนแปลงเกิดขึ้นได้แม้ว่าจะไม่มี ความเป็นจริงสำหรับลูกค้าในวันนั้น หากข้อมูลถูกวางไว้ใน minidimension การเปลี่ยนแปลงจะไม่สามารถบันทึกได้จนกว่าจะมีการเกิดขึ้นจริงกับลูกค้ารายนั้นในครั้งต่อไป

2) Outriggers เราเพิ่งเห็นว่าผู้จัดจำหน่ายหนังสือสามารถอธิบายโปรไฟล์ลูกค้าด้วยข้อมูลส่วนบุคคลโปรไฟล์เพื่อให้มิติลูกค้ามีขนาดเล็กลงแม้ว่าจะมีการเปลี่ยนแปลงอย่างรวดเร็วก็ตาม สมมติว่าผู้ขายหนังสือต้องการดูโปรไฟล์ปัจจุบันที่เชื่อมโยงกับลูกค้าที่กำหนดโดยไม่ต้องพึ่งพาความเป็นจริงสำหรับลูกค้าและโปรไฟล์ปัจจุบัน วิธีแก้ปัญหาที่ไร้เดียงสาคือการรวมแอตทริบิวต์ 30 รายการจากมิติข้อมูลโปรไฟล์ในมิติข้อมูลลูกค้าและใช้การอัปเดตประเภท 1 เพื่อแสดงโปรไฟล์ปัจจุบัน อย่างไรก็ตามดูเหมือนว่าคุณควรรักษาแอตทริบิวต์ Profile ไว้บางส่วนไว้ในทั้งข้อมูลส่วนบุคคล minidimension และในมิติข้อมูล Customer ทางออกที่ดีคือการอ้างอิง minidimension ของโปรไฟล์โดยตรงจากมิติข้อมูลลูกค้า (และใช้การปรับปรุงประเภทที่ 1 เพื่ออ้างอิงโปรไฟล์ปัจจุบันของลูกค้าเท่านั้น) มิติที่อ้างอิงจากอีกมิติหนึ่งเรียกว่า outrigger (outrigger) Outriggers ใช้ในสภาพแวดล้อม OLAP เชิงสัมพันธ์ซึ่งตารางมิติข้อมูลของขากรรไกรถูกอ้างอิงโดยคีย์ต่างประเทศในตารางมิติข้อมูลอื่น เมื่อต้องการใช้ outrigger (Outrigger) จะมีการรวมกันระหว่างตารางมิติและขาหนีบ โปรดสังเกตว่าการใช้ outrigger แตกต่างจากการใช้ schema เกล็ดหิมะ ตัวยึดหลักไม่จำเป็นต้องเป็นปกติ มิติสามารถใช้เป็นมิติสามัญและเป็น outrigger ในเวลาเดียวกัน ตัวอย่างเช่นกรณีเช่นนี้สำหรับมิติข้อมูลโปรไฟล์ที่อธิบายไว้ด้านบน อีกตัวอย่างหนึ่งคือมิติเวลา มิติเวลาเป็นส่วนหนึ่งของก้อนโดยทั่วไปซึ่งเป็นข้อเท็จจริงที่เกี่ยวข้องกับวันที่กำหนด แต่มิติเวลาอาจมีการอ้างอิงจากส่วนข้อมูลอื่น ๆ เช่นจากมิติข้อมูลลูกค้าเพื่อติดตามวันเกิดของลูกค้าหรือจากมิติข้อมูลหนังสือ เพื่อติดตามเมื่อมีการเผยแพร่หนังสือ

3) Degenerate Dimensions ต่อไปนี้เราสมมติว่าร้านค้าปลีกหนังสือของเราใช้รายละเอียดปลีกย่อยและติดตามรายการแต่ละครั้งที่มีการขายหนังสืออย่างน้อยหนึ่งเล่ม (ไม่เฉพาะยอดขายต่อหนึ่งเล่มต่อวันต่อสถานที่) มิติข้อมูล Book, Time, และ Location แต่ไม่เพียงพอเนื่องจากเราไม่สามารถระบุการทำธุรกรรมแต่ละ

รายการได้ ดังนั้นเราจึงต้องมีมิติข้อมูลอีกอันหนึ่งเพื่อให้สามารถระบุรายการที่ไม่ซ้ำกันได้ มิติข้อมูลนี้มีตัวระบุที่ไม่ซ้ำกันจากแต่ละธุรกรรมเช่นหมายเลขเฉพาะที่เครื่องลงทะเบียนเงินสดพิมพ์ในใบเสร็จรับเงินหรือเพียงแค่จำนวนเต็ม "dumb" ที่เพิ่มขึ้นสำหรับแต่ละธุรกรรมเท่านั้น อย่างไรก็ตามไม่มีแอตทริบิวต์คำอธิบายอื่น ๆ และระดับเฉพาะของมิติข้อมูลเป็นระดับต่ำสุดสำหรับตัวระบุที่ไม่ซ้ำและเป็นพิเศษหรือไม่ T ระดับสำหรับการทำธุรกรรมทั้งหมด มิติดังกล่าวประกอบด้วยเพียงตัวระบุเดียวเท่านั้นที่เป็นมิติที่เลวร้าย ในการแทนความสัมพันธ์ตามที่กล่าวไว้ในข้อ 2.14 ไม่จำเป็นต้องสร้างตาราง separate dimension หากสำหรับมิติที่เสื่อมลง ถ้าเราทำตารางความเป็นจริงจะมีคีย์ต่างประเทศที่อ้างถึงตารางมิติข้อมูลซึ่งจะมีแอตทริบิวต์เดียว (เป็นร่วมกันสามารถหลีกเลี่ยง) เพียงแค่วางแอตทริบิวต์ตัวบ่งชี้ของมิติที่เสื่อมลงโดยตรงในตารางความเป็นจริงจึงหลีกเลี่ยงมิติที่แยกต่างหากและคีย์ต่างประเทศทราบว่ามันเป็นสิ่งสำคัญที่จะ ใช้จำนวนเต็มเป็นตัวระบุเพื่อให้ขนาดของตารางความเป็นจริงลง รูปที่ 2.13 แสดงตารางความเป็นจริงสำหรับร้านค้าปลีกหนังสือของเราเมื่อเราได้เพิ่ม transaction ของ regenerate dimension (ตารางความเป็นจริงคือสถานที่เดียวที่ระบุตัวตนของธุรกรรม TransactionID) ทราบว่าเราได้เพิ่มการวัดเป็นราคาปลีกแล้ว ความละเอียดช่วยให้เราสามารถติดตามราคาหนังสือที่ขายในธุรกรรมที่กำหนด (ซึ่งอาจแตกต่างไปจากธุรกรรมต่อธุรกรรมและจากร้านค้าไปยังร้านค้าเนื่องจากข้อเสนอพิเศษแคมเปญฯลฯ)

Sales (fact table)

BookID	CityID	DayID	TransactionID	Sale	Price
7493	854	2475	102	1	10
7493	854	2475	123	1	10
7493	854	2475	232	2	20
7493	854	2475	244	1	10
7493	876	3456	400	1	15
7493	876	3456	523	1	15
9436	876	3456	523	1	12

รูป 2.13 แก้ไขตารางความเป็นจริงสำหรับ Sales cube

4) Junk Dimensions ด้วยการแนะนำและใช้ข้อมูล Transaction-level ในส่วนก่อนหน้านี้เป็นเรื่องที่เกี่ยวข้องกับผู้จัดจำหน่ายหนังสือของเราเพื่อให้มีบริบทมากขึ้นเกี่ยวกับหนังสือแต่ละเล่มที่ขายใน transaction ขณะนี้เรายังต้องการ ติดตามหนังสือเล่มนี้ "แสดงในตำแหน่งที่สำคัญ" "แสดงในตำแหน่งรอง" และ "ไม่

ปรากฏ") หากหนังสือมีการลดราคา (มีสองตัวเลือกคือ "ส่วนลด" "ไม่ลดราคา") และสุดท้าย (มีสี่ตัวเลือกคือ "ข่าว" "การลงนามหนังสือ" "เหตุการณ์อื่น ๆ " "ไม่มีกิจกรรม") ในการติดตามผลตัวเลือกเหล่านี้เราสามารถเพิ่มมิติข้อมูลการแสดงผลส่วนลดและกิจกรรม 3 มิติได้ มิติข้อมูลแต่ละมิติมีค่ามิติน้อย (ระหว่างสองและสี่!) อย่างไรก็ตามเนื่องจากแต่ละมิติข้อมูลมีค่ามิติน้อยมากจึงเป็นไปได้ที่จะ "ผสาน (merge)" ให้เป็นมิติเดียวโปรโมชัน มิติข้อมูลนี้มีชุดค่าผสมที่เป็นไปได้ทั้งหมดในตัวเลือกรวม $3 \cdot 2 \cdot 4 = 24$ ค่ามิติเราเรียกมิติดังกล่าวโดยใช้ชุดค่าที่ไม่เกี่ยวข้องซึ่งเป็นมิติข้อมูลขยะ มิติข้อมูลขยะควรใช้เฉพาะกับธงกลุ่มและคำอธิบายที่มีความละเอียดต่ำเนื่องจากอาจมีชุดค่าผสมที่เป็นไปได้มากเกินไป ในขณะที่การสืบค้นข้อมูลอาจกลายเป็นเรื่องยากขึ้นเล็กน้อยกับ มิติขยะ (junk dimension) แทนที่จะเป็นมิติข้อมูลที่เป็นอิสระหลายประการการจัดเรียงนี้จะลดขนาดของลูกบาศก์ ในการเป็นตัวแทนเชิงสัมพัทธ์นี้นำไปสู่การประหยัดพื้นที่ขนาดใหญ่เนื่องจากตารางความจริงจำเป็นต้องใช้คอลัมน์พิเศษเพียงอย่างเดียวเท่านั้นเพื่อรองรับข้อมูลตามบริบทใหม่แทนที่จะเป็นคอลัมน์พิเศษสามคอลัมน์ (โปรดจำไว้ว่าตารางความเป็นจริงน่าจะมีแถวหลายล้านแถว) คิมบอลล์ & รอสส์ ได้ยกตัวอย่างให้เห็นของมิติขยะโดยการเปรียบเทียบกับลิ้นชักขยะในห้องครัวที่กรรไกรแถบยางเทป ฯลฯ จะถูกเก็บไว้ น่าจะมีลิ้นชักสำหรับแต่ละประเภท แต่ก็มีพื้นที่ว่างไม่เพียงพอ ดังนั้นเราจึงนำเสนอวิธีการแก้ปัญหาอย่างจริงจังและนำสิ่งของที่ไม่เกี่ยวข้องทั้งหมดเหล่านี้ไปไว้ในลิ้นชักหนึ่งลิ้นชัก

5) Time Dimensions กระบวนการสร้างฐานข้อมูลแบบหลายมิติในแบบเรียลไทม์หรือสถานะที่มีวิวัฒนาการไปเรื่อย ๆ (ในตัวอย่างของเราผู้จำหน่ายหนังสือขายหนังสือทุกวัน) ในการสร้างแบบจำลองนี้จะใช้มิติข้อมูลเวลา มิติเวลาจึงพบได้ในเกือบทุกฐานข้อมูลหลายมิติ จำได้ว่า Inmon มองว่ามีมิติเวลาเป็นลักษณะเฉพาะของคลังข้อมูล มิติข้อมูล Adate หมายถึงวันที่ (เช่น "11 พฤศจิกายน 2009") ในขณะที่มิติข้อมูลแบบเวลาเป็นวันแสดงเวลานาฬิกา (เช่น "11:47 น.") ในระหว่างวันที่ไม่ระบุรายชื่อ มิติข้อมูลทั้งสองแบบมีอยู่ในฐานข้อมูลแบบหลายมิติหรือไม่นั้นขึ้นอยู่กับการใช้ฐานข้อมูลที่ต้องการ บางครั้งก็เพียงพอที่จะจับภาพวันที่และไม่ใช่เวลาที่ละเอียดขึ้นในแต่ละวันความละเอียดของมิติข้อมูลเหล่านี้อาจแตกต่างกันขึ้นอยู่กับการใช้งานที่ต้องการ ตัวอย่างเช่นอาจมีการรวบรวมข้อมูลในระดับรายเดือนแทนที่จะเป็นระดับรายวัน (ซึ่งหมายความว่ามิติข้อมูลวันที่จะเป็นเดือนที่เฉพาะเจาะจงเช่น "พฤศจิกายน 2009" แทนวันที่ระบุ) อาจเป็นไปได้ที่จะรวมทั้งวันที่และเวลาของวันในมิติข้อมูลเดียว อย่างไรก็ตามมิตินี้จะเติบโตขึ้น

อย่างรวดเร็ว สมมติว่ามิติดังกล่าวแสดงแต่ละนาทีในแต่ละวัน สำหรับวันเดียวจะมีค่ามิติเท่ากับ 60404 = 1,440 สำหรับปีเดียวจะมีค่ามิติ 1,440 · 365 = 525,600 ในการเปรียบเทียบมีเพียง 1,440 ค่ามิติในมิติเวลา (ไม่ขึ้นกับจำนวนปีที่แสดงในมิติข้อมูลวันที่) และค่ามิติข้อมูล 365 ต่อปีในมิติข้อมูลวันที่ นอกจากนี้นักวิเคราะห์ยังอาจพิจารณาเวลาของวันและวันที่อย่างเป็นธรรมชาติด้วย ตัวอย่างเช่นจะทำให้สามารถสร้างตาราง Pivot ที่มีวันที่ที่ระดับเดือนหนึ่งแกนและช่วงเวลาของวันที่ระดับชั่วโมงตามแกนอื่น ๆ ซึ่งจะช่วยให้สามารถวิเคราะห์ว่ายอดขายในช่วงกลางวันมีผลต่อฤดูกาลอย่างไรแม้ว่า DBMS จะใช้สำหรับการเก็บข้อมูลข้อมูลมีประเภทแอตทริบิวต์ DATE แนะนำให้ใช้มิติข้อมูลเวลาแทนที่จะใช้แอตทริบิวต์ DATE แบบง่ายในตารางข้อเท็จจริง เมื่อมีมิติข้อมูลเวลาความรู้เฉพาะโดเมนก็สามารถแสดงและทำให้นักวิเคราะห์ได้ง่าย ตัวอย่างเช่นสามารถจับได้อย่างง่ายดายไม่ว่าจะเป็นวันที่เป็นส่วนหนึ่งของเทศกาลวันหยุดหรือไม่ว่าจะมีการจัดกิจกรรมพิเศษอย่าง SoccerWorld Cup ในระหว่างวันใดวันหนึ่ง ไม่สามารถแยกข้อมูลประเภทนี้ออกจากประเภท DATE แบบธรรมดา นอกจากนี้มิติข้อมูลเวลาจะใช้งานได้ง่ายขึ้นในการสืบค้นข้อมูลเนื่องจากไม่มีตรรกะของปฏิทินต้องถูกดำเนินการในแบบสอบถามเมื่อมีการแสดงมิติข้อมูลแบบลำดับชั้นการทำงานแบบเจาะลึกและแบบโรลอัพที่เกี่ยวข้องกับการใช้งานจะง่ายขึ้นในการดำเนินการเพื่อให้มิติข้อมูลเวลามีประโยชน์มากที่สุดเท่าที่จะเป็นไปได้สิ่งสำคัญคือต้องรวมแอตทริบิวต์ที่อธิบายถึงไว้ด้วยเช่นกันถ้าแอตทริบิวต์เหล่านี้เป็นตัวเดียวกัน ตัวอย่างเช่นมีความสัมพันธ์แบบหนึ่งต่อหนึ่งระหว่าง DayNumberInWeek (มีค่า 1, 2, ... , 7) และWeekDay (มีค่าวันจันทร์อังคาร ... วันอาทิตย์) แต่ก็ยังมีประโยชน์ที่จะรวมทั้ง แอตทริบิวต์เพื่อให้การกำหนดการสืบค้นและเรียกดูค่ามิติง่ายและใช้งานง่าย อีกตัวอย่างหนึ่งคือจำนวนวันในเดือนหนึ่งซึ่งโดยหลักการอาจมาจากเดือน (และปีในกรณีของเดือนกุมภาพันธ์) แต่แทนที่จะใช้ภาระกับผู้ใช้รายนี้แอตทริบิวต์เช่น NumberOfDaysInMonth ควร รวมอยู่ในมิติข้อมูลวันที่ มิติข้อมูลวันที่มักมีมากกว่าหนึ่งลำดับชั้นเนื่องจากทั้งปีปฏิทินและปีงบประมาณมักได้ผลกลับไปยังลำดับหลายชั้น

6) Data Quality Dimensions บางครั้งก็เป็นไปได้ที่จะประเมินคุณภาพของข้อมูลในฐานข้อมูล ตัวอย่างเช่นสมมติว่าร้านหนังสือของร้านค้าปลีกแห่งหนึ่งมักขายหนังสือ 3-5 เล่มในแต่ละวัน แต่ดูเหมือนว่าวันหนึ่งขายหนังสือเล่มนี้ได้ 1,000 เล่ม นี่อาจเป็นข้อผิดพลาด แต่อาจเป็นความจริงเนื่องจากมีส่วนลดหรือเป็นภาพยนตร์ใหม่ที่อิงตามหนังสือ ดังนั้นอาจเป็นอันตรายต่อการลบข้อมูล สามารถใช้มิติพิเศษ

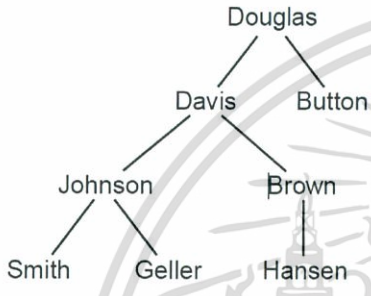
เพื่ออธิบายถึงความเป็นจริงแต่ละประการค่ามิติที่จะรวมไว้ในมิติข้อมูลคุณภาพดังกล่าวขึ้นอยู่กับความต้องการทางธุรกิจ ค่าทั่วไปคือ "ค่าปกติ", "ค่าที่อยู่นอกขอบเขต", "ค่าที่ไม่น่าเชื่อถือ", "ค่าที่ได้รับการยืนยันแล้ว", "ค่าที่ไม่ได้ตรวจสอบ" และ "ค่าที่ไม่แน่นอน" การรวมข้อมูลทั้งหมดแม้ว่าคุณภาพที่เกี่ยวข้องจะเป็นปัญหานักวิเคราะห์จะต้องให้ภาพเต็มรูปแบบ หากต้องการใช้เฉพาะข้อมูลที่มีคุณภาพสูงเท่านั้นพวกเขาสามารถ จำกัด มิติข้อมูลคุณภาพเป็น "ค่าปกติ" หรือ "มูลค่าที่ยืนยันแล้ว" หากต้องการดูข้อมูลทั้งหมด (รวมถึงข้อมูลที่ไม่ถูกต้อง) พวกเขาสามารถละเลยมิติข้อมูลคุณภาพข้อมูลได้ ทั้งหมด

2.3.3.3 Advanced Hierarchies

การจัดลำดับชั้นในเบื้องต้น เป็นลำดับชั้นที่เรียบง่าย เราสมมติว่าพวกเขามีความสมดุล (เช่นในลำดับชั้นใด ๆ ก็ตาม Leaf ทั้งหมดอยู่ในระดับต่ำสุดของสคีมา) ครอบคลุม (เช่นในแต่ละกรณีเส้นทางเริ่มต้นที่รากและจากนั้นไปที่ระดับด้านล่างทันที ในสคีมาแล้วไปที่ระดับถัดไปด้านล่างและอื่น ๆ โดยไม่ข้ามระดับใดก็ได้) และเข้มงวด (เช่นในกรณีที่ระบุไม่มีค่ามิติใดที่มีผู้ถือมากกว่าหนึ่งราย) อย่างสังหรณ์ใจ นั่นหมายความว่ากรณีที่เป็นรูปแปปของ balanced tree ในส่วนนี้เราพิจารณาลำดับชั้นทั่วไปที่ไม่เป็นไปตามข้อกำหนดเหล่านี้และเราจะหารือถึงวิธีการสนับสนุนลำดับชั้นดังกล่าว นอกจากนี้เรายังอธิบายว่ามิติข้อมูลสามารถมีลำดับชั้นได้อย่างไรบ้างแทนที่จะเป็นแบบเดียวกับที่สมมติในการลับดับชั้นแบบพื้นฐาน

1) Parent-Child Hierarchies ในการแบ่งลำดับชั้นเมื่อเราได้ทำการจัดแบ่งกลุ่มลำดับขั้นแล้วกับค่าระดับที่มีอยู่ ตัวอย่างเช่นเมืองถูกจัดกลุ่มเป็นรัฐ ในบางกรณีประเภทของระดับผู้ปกครอง(Parent)จะไม่แตกต่างไปจากประเภทของระดับเด็ก(Child) ดังนั้นผู้ปกครองสามารถเป็นลูกของผู้ปกครองคนอื่น ๆ ที่มีประเภทเดียวกันได้เราแสดงให้เห็นถึงสิ่งนี้โดยลำดับชั้นของพ่อแม่และลูก ตัวอย่างคือขนาดพนักงาน พนักงานมีผู้จัดการที่เป็นพนักงานและมีผู้จัดการคนอื่น ๆ Johnson มี Davis เป็นผู้จัดการและ Davis มี Douglas เป็นผู้จัดการ ฯลฯ ไม่เป็นประโยชน์ในการใช้จำนวนระดับที่กำหนดในมิติ Employee เพื่อแสดงลำดับชั้นดังกล่าว สำหรับร้านค้าปลีกหนังสือร้านเล็ก ๆ อาจมีเพียงผู้จัดการร้านและผู้ช่วยน้อย (เช่นสองระดับ) ในขณะที่ร้านใหญ่ ๆ ก็มีผู้จัดการระดับกลาง (เช่นสามระดับขึ้นไป) แทนที่จะให้จำนวนระดับที่แน่นอนเราอนุญาตให้สคีมาของลำดับชั้นแม่และลูกมีระดับเพียงอย่างเดียว อย่างไรก็ตามในกรณีของลำดับชั้นเราอนุญาตให้ค่ามิติข้อมูล (จากระดับเดียว) มีค่ามิติข้อมูลอื่น (จากระดับเดียวกัน) เป็น parent ด้วยเหตุนี้เราจึงสามารถมีได้ไม่ จำกัด จำนวนระดับในกรณีเช่นนี้ การวัดค่าของเด็กของผู้ปกครองจะรวมกันเป็นค่าเดียวสำหรับผู้ปกครองโดยใช้สูตรในการวัด วิธีการข้างต้น

สามารถจับลำดับชั้นแม่และเด็กบางระบบ relational OLAP สามารถใช้การแทนดังกล่าวได้ อย่างไรก็ตามการแสดงนี้ไม่สะดวกในการวิเคราะห์โดยใช้ SQL มาตรฐาน ถ้าเราต้องการหายอดขายทั้งหมดที่สร้างโดยผู้จัดการร้านและพนักงานทุกคนที่อยู่ต่ำกว่าพวกเขาการแสดงในรูปที่ ? เป็นเรื่องยากที่จะใช้เนื่องจากไม่มีวิธีที่ง่ายในการสำรวจลำดับชั้นและ roll up จากผู้ช่วย ไปยังผู้จัดการร้าน เพื่อแก้ปัญหานี้จะใช้การแทนตารางอื่นที่เรียกตารางบริดจ์ได้ ตารางบริดจ์มีแถวสำหรับแต่ละเส้นทางระหว่างบรรพบุรุษและลูกหลานในลำดับชั้นรวมถึงเส้นทางที่มีความยาวน้อย 0 ตารางบริดจ์มีคอลัมน์ต่อไปนี้



Employee dimension table

EmployeeID	Employee	ManagerID
1	Douglas	NULL
2	Davis	1
3	Johnson	2
4	Smith	3
5	Geller	3
6	Brown	2
7	Hansen	6
8	Button	1

รูป 2.14 Parent-child dimension และ relational representation[3]

- บรรพบุรุษ (Ancestor): คอลัมน์ foreign key ที่อ้างถึงคอลัมน์ primary key ของตารางมิติเพื่อรวบตัวกับบรรพบุรุษของแถว
- ลูกหลาน (Descendant): คอลัมน์ foreign key ที่อ้างถึงคอลัมน์ primary key ของตารางมิติเพื่อรวบตัวกับลูกหลานของแถว
- ระยะทาง (Distance): จำนวนเต็มที่จับความยาวของเส้นทางระหว่างบรรพบุรุษและลูกหลาน
- ค่าสถานะล่างสุด (Bottom Flag): ค่าความจริงที่ระบุว่า Descendant อยู่ในระดับต่ำสุด (เช่น ไม่มีลูกหลาน)
- ค่าสถานะสูงสุด (Top Flag): ค่าบูลความจริงที่ระบุว่าบรรพบุรุษอยู่ที่ระดับสูงสุด (ไม่ได้มีบรรพบุรุษ)

ลำดับชั้น parent-child แสดงในรูปที่ 3.5 แสดงโดยตารางบริดจ์ที่แสดงในรูปที่ 3.6 (สมมติว่า EmployeeID เหมือนกันก่อน) สามารถมองเห็นได้จากแถวแรกที่ Douglas กับ EmployeeID 1 เป็นผู้จัดการตนเอง จากแถวที่สองก็จะเห็นได้ว่า ดักลาสยังเป็นผู้จัดการของ Davis ที่มี EmployeeID 2 fact table ยังคงเหมือนเดิม และคุณสามารถละเลยตารางบริดจ์และเชื่อมต่อ fact table กับ dimension table ข้อมูลได้โดยตรง ในตัวอย่างนี้หมายความว่า fact table มีคอลัมน์ foreign key ที่

อ้างอิง dimension table ข้อมูลพนักงาน fact table หมายถึงสิ่งที่พนักงานแต่ละคนขายได้ ตารางบริดจ์เข้ามาเข้ามาในส่วนนี้หากเราต้องการข้ามลำดับชั้น parent และ child เช่นการเพิ่มยอดขายให้กับผู้จัดการร้านหากต้องการทำเช่นนี้เราจะเข้ารวม dimension, bridge และ fact table โดยมีเงื่อนไขว่า EmployeeID (primary key) ของ dimension table เท่ากับแอตทริบิวต์บรรพบุรุษ (Ancestor) ในตารางบริดจ์ และคอลัมน์ foreign key ใน fact table จะเท่ากับคอลัมน์ลูกหลาน (Descendant) ในตารางบริดจ์และเราจะวางข้อจำกัด ที่เกี่ยวข้องกับ dimension ข้อมูล Employee ตามที่แสดงต่อไป

โปรแกรม 2.32 แสดงข้อมูลพนักงานและสิ่งที่พนักงานขายได้[3]

```
SELECT E.EmployeeID, SUM(F.Sales_Amount)
FROM Employee E, Bridge B, Sales F
WHERE E.EmployeeID = B.Ancestor AND B.Descendant =
F.EmployeeID
AND E.Title = 'Shop Manager'
GROUP BY E.EmployeeID
```

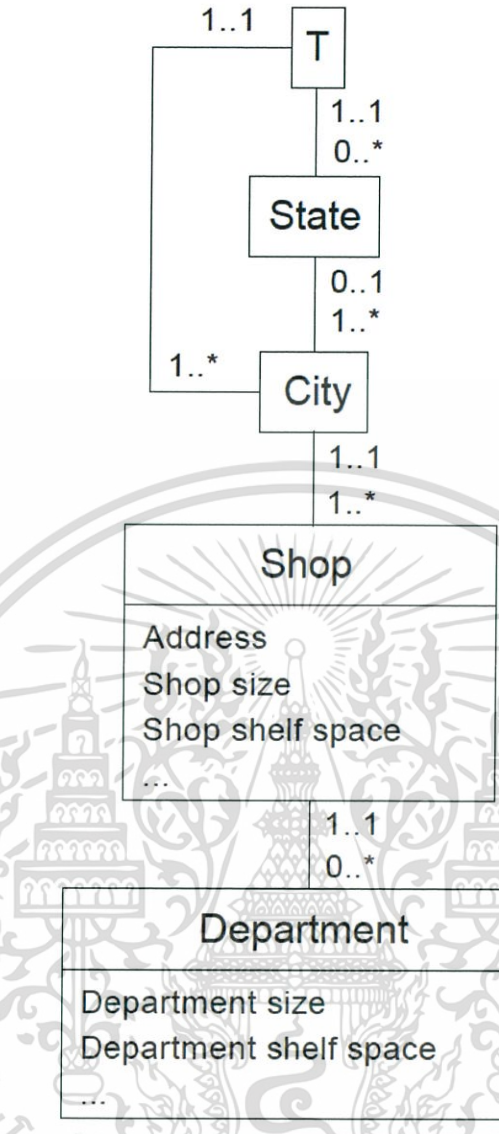
Ancestor	Descendant	Distance	Bottom Flag	Top Flag
1	1	0	False	True
1	2	1	False	True
1	8	1	True	False
1	3	2	False	False
1	6	2	False	False
1	4	3	True	False
1	5	3	True	False
1	7	3	True	False
2	2	0	False	False
2	3	1	False	False
2	6	1	False	False
2	4	2	True	False
2	5	2	True	False
2	7	2	True	False
8	8	0	True	False
3	3	0	False	False
3	4	1	True	False
3	5	1	True	False
6	6	0	Flase	False
6	7	1	True	False
4	4	0	True	Flase
5	5	0	True	Flase
7	7	0	True	Flase

รูป 2.15 ตารางบริดจ์ที่อ้างถึง ลำดับความสำคัญแบบ parent-child

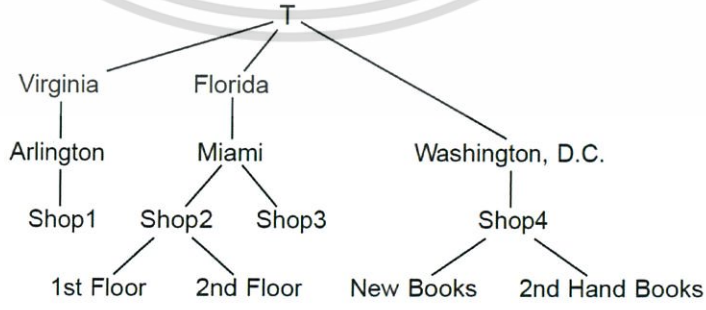
เพื่อทำความเข้าใจวิธีการทำงานนี้โปรดจำไว้ว่าตารางบริดจ์เก็บแถวสำหรับแต่ละเส้นทางระหว่างบรรพบุรุษและลูกหลานในลำดับชั้นเมื่อเราเข้าร่วมตารางมิติข้อมูลพนักงานกับตารางบริดจ์เราจะได้แถวสำหรับการรวมกันของผู้จัดการร้านและพนักงานใต้ผู้จัดการ (รวมถึงผู้จัดการ) แถวเหล่านี้จะถูกรวมเข้าด้วยกันแล้วด้วยแถวความเป็นจริงที่เราได้รับยอดขายที่พนักงานทำไว้ สุดท้าย GROUP BY ช่วยให้มั่นใจได้ว่าเราสรุปยอดขายทั้งหมดสำหรับพนักงานภายใต้ผู้จัดการที่ระบุ เมื่อใช้แอตทริบิวต์ระยะทางในตารางบริดจ์เราสามารถวิเคราะห์ขั้นสูงได้มากขึ้น ถ้าเราเพิ่มคำจำกัดความ B.Distance = 1 ลงในแบบสอบถามข้างต้นเราจะพบยอดขายโดยพนักงานเหล่านั้นโดยตรงภายใต้ผู้จัดการ ถ้าเราใช้ B.Distance> 0 แทนเราพบยอดขายที่ทำโดยพนักงานของผู้จัดการ แต่เราไม่รวมยอดขายที่

ผู้จัดการร้านทำเองด้วย ในทำนองเดียวกันธงด้านบนและด้านล่างสามารถใช้เพื่อแสดงข้อสงสัยบางอย่างได้สะดวก

2) Unbalanced Hierarchies ในตัวอย่างของลำดับชั้นที่ไม่สมดุลค่ามิติข้อมูลจะอยู่ในระดับที่แตกต่างจากระดับต่ำสุดในสคีมาของลำดับชั้นและไม่มีลูกในอินสแตนซ์ กล่าวอีกนัยหนึ่งค่ามิติสำหรับระดับต่ำสุดจะหายไปซึ่งอินสแตนซ์จะสร้างต้นไม้ที่ใบมีระยะทางที่แตกต่างกันไปในระดับ T ตัวอย่างหนึ่งของลำดับชั้นที่ไม่สมดุลคือลำดับชั้นระดับ parent-child ในรูปที่ 2.14 ตัวอย่างอื่น ๆ (ซึ่งไม่ใช่ลำดับชั้นระดับ parent-child) เกิดขึ้นถ้าร้านค้าปลีกขนาดใหญ่ของร้านค้าปลีกหนังสือกลายแบ่งออกเป็นแผนกต่างๆเช่นแผนกต่างๆอยู่ในร้านค้า ร้านค้าขนาดเล็กไม่แบ่งย่อย หมายความว่าร้านค้าที่ระบุมีศูนย์หรือมากกว่าแผนก schema สำหรับ dimension Shop ที่ได้รับการแก้ไขจะแสดงในรูปที่ 2.16 ซึ่งเราได้ขยายสัญกรณ์ที่ใช้เพื่อให้สามารถแสดง schema ขั้นสูงเพิ่มเติมได้ ดังนั้นระดับจะแสดงด้วยกล่องที่ชื่อระดับจะแสดงเป็นตัวหนาในบรรทัดด้านบนมากที่สุด ด้านล่างชื่อระดับและภายในช่องคุณสมบัติระดับจะแสดงขึ้นถ้ามี เส้นแบ่งระหว่างสองระดับ A และ B (ที่ A ถูกวาดเหนือ B) ยังคงหมายความว่ามีลำดับชั้นที่ระดับ A อยู่เหนือระดับ B นั่นคือค่า B จะจัดกลุ่มเป็นค่า A บรรทัดดังกล่าวอาจมีความชัดเจนที่แสดงว่าเชื่อมต่อกับกล่องต่างๆ ในภาพ 2.16 เราจะเห็นได้ว่าร้านค้ามีตั้งแต่ 0 ถึงหลายร้าน (* หมายถึงจำนวนเต็มบวกที่ไม่ จำกัด) ในขณะที่แผนกเป็นของตั้งแต่ 1 ถึง 1 (กล่าวคือ 1) ร้านค้า ถ้าไม่มี cardinalities แสดงเราสมมุติสมมติว่าถ้า A สามารถ roll-up ไปที่ B แล้วแต่ละค่าเป็นของค่า B เดียวกันและที่อย่างน้อยหนึ่งและอาจเป็นจำนวนมากค่า B เป็นค่าที่กำหนด ในรูปที่ 2.17 ซึ่งแสดงตัวอย่างของมิติร้านค้าที่แก้ไขแล้วเราจะเห็นได้ว่าร้าน Shop2 มีแผนกในขณะที่ร้าน Shop1 ไม่มี ในระดับการดำเนินงานเป็นการยากที่จะสนับสนุนลำดับชั้นที่ไม่สมดุล ถ้าระดับเป็นประเภทเดียวกันอาจใช้ลำดับชั้นระดับ parent-child หากระดับของแต่ละประเภทมีความแตกต่างกันแต่ก็ยากขึ้นเนื่องจากข้อเท็จจริงที่แตกต่างกันมีผลต่อระดับที่แตกต่างกันตามลำดับชั้น ทางออกหนึ่งคือทำให้ลำดับชั้นไม่สมดุลสมดุลโดยการขยายช่องด้วยค่าตัวยึดตำแหน่งที่ระดับล่าง การดำเนินการนี้สามารถดำเนินการได้อย่างโปร่งใสโดยระบบ



รูปที่ 2.16 schema ที่ได้รับการแก้ไขสำหรับมิติข้อมูลร้านค้า[3]

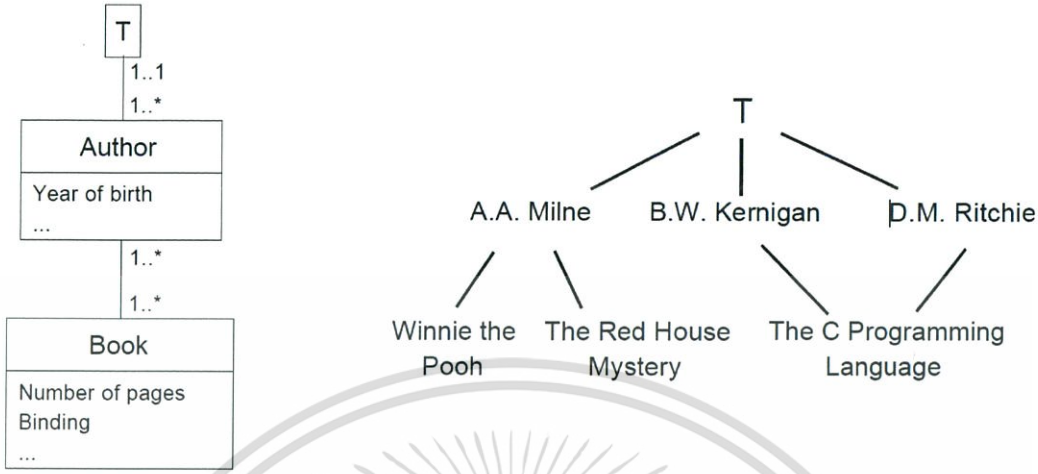


รูป 2.17 ตัวอย่างของมิติข้อมูลร้านค้าที่แก้ไขแล้ว[3]

3) Non-covering Hierarchies ลำดับชั้นที่ไม่ครอบคลุมช่วยให้อินสแตนซ์สามารถข้ามระดับระหว่าง leaf และ root ได้ พิจารณามิติข้อมูลร้านค้าในรูป 3.7 ที่ค่า City ถูกจัดกลุ่มเป็นค่าสถานะ ที่นี่เมือง Washington, DC ไม่ได้เป็นของรัฐ ซึ่งหมายความว่าเราข้ามระดับรัฐเมื่อเป็นตัวแทน Washington, DC ปรากฏการณ์นี้จะเกิดขึ้นหากผู้ค้าปลีกหนังสือขยายไปยังประเทศอื่น ๆ โดยไม่มีรัฐต่างๆเช่นประเทศในยุโรปหลายแห่งโปรดสังเกตว่าในสัญกรณ์ใหม่ของเรา cardinality 0..1 จากเมือง ไปยังรัฐหมายความว่ามูลค่าเมืองอาจไม่ใช่ของรัฐหรือมีมูลค่ารัฐเดียว หากเป็นของรัฐไม่มีค่าก็ยังคงเป็นของค่าจากระดับเหนือรัฐในกรณีนี้ วิธีการสนับสนุนลำดับชั้นที่ไม่ครอบคลุมคือการแทรกค่าตัวยึดตำแหน่งเมื่อค่าใด ๆ ขาดค่าที่มีอยู่ในระดับความเป็นเจ้าของ (ทันที) ใน schema ในตัวอย่างเช่นค่าตัวยึดตำแหน่งจะถูกแทรกลงในระดับรัฐ ค่าตัวยึดตำแหน่งนี้มีค่าระดับเมือง Washington, D.C. เป็น child เดียว ค่าคุณสมบัติชื่อและระดับที่เหมาะสมสำหรับตัวยึดตำแหน่งนี้จะพิจารณาจากความต้องการทางธุรกิจ อาจทำให้รู้สึกรวมถึงรัฐโดยปลอมค่า "Washington, D.C" ซึ่งนักวิเคราะห์สามารถดำเนินการได้เช่นเดียวกับ Washington, D.C. เป็นรัฐหรืออาจเป็นการดีกว่าที่จะปล่อยให้ชื่อว่างเปล่าเพื่อแสดงให้เห็นว่านี่เป็นเพียงตัวยึดตำแหน่งเท่านั้น ในบางระบบเช่น Microsoft Analysis Services (ซึ่งลำดับชั้นที่ไม่ครอบคลุมเรียกว่าลำดับชั้นที่ขาดหายไป) สามารถซ่อนค่าตัวยึดตำแหน่งเมื่อนักวิเคราะห์ใช้ลำดับชั้นได้

4) Non-Strict Hierarchies เราได้พิจารณาลำดับชั้นที่แต่ละค่า dimension ของข้อมูล (นอกเหนือจากค่า T พิเศษ) เป็นของค่าสัมบูรณ์เดียวกัน ลำดับชั้นดังกล่าวเรียกว่าลำดับขั้นที่เข้มงวด ในกรณีที่มีลำดับชั้นที่ไม่เข้มงวด child อาจมี parent มากกว่าหนึ่งรายได้เช่นกันซึ่งความสัมพันธ์ระหว่าง many-to-many สามารถอยู่ระหว่างค่ามิติข้อมูลในระดับต่างๆได้ ลำดับชั้นที่ไม่เข้มงวดเกิดขึ้นบ่อยๆในโลกแห่งความเป็นจริง ในตัวอย่างร้านค้าปลีกหนังสือของเรามีความสัมพันธ์ระหว่างหนังสือและนักเขียนเป็นจำนวนมาก (ผู้แต่งสามารถเขียนหนังสือจำนวนมากหนังสือเล่มหนึ่งอาจมีผู้เขียนจำนวนมาก) นี่แสดงในรูปที่ 3.9 ความสัมพันธ์ many-to-many ก่อให้เกิดปัญหาเกี่ยวกับการรวมตัวสมมติว่าหนังสือ B เขียนโดยผู้แต่งสองคน ถ้าเราเลื่อนขึ้นจากระดับ Book ไปที่ระดับ Author และจากระดับ Author T ไปที่ B เรานับยอดขายของ B สองครั้ง (หนึ่งครั้งสำหรับผู้แต่งแต่ละคน) เว้นแต่เราจะสนใจเหตุการณ์นี้เป็นพิเศษ เพื่อเป็นตัวแทนความสัมพันธ์แบบ many-to-many ใน ROLAP ไม่เพียงพอที่จะใช้คอลัมน์เดียวเพื่อแสดงระดับในกรณีของ star-schema หรือ foreign key ในกรณีที่เป็น snowflake schema เราใส่ตารางสำหรับแต่ละระดับ

ที่มีส่วนร่วมในความสัมพันธ์ many-to-many และรวมถึงตารางบริดจ์ที่อยู่ระหว่างทั้งสองภาพนี้เป็นภาพประกอบสำหรับมิติข้อมูลหนังสือในรูปที่ 3.10



รูป 2.18 Schema และอินสแตนซ์ตัวอย่างสำหรับมิติข้อมูล Book ที่มีความสัมพันธ์ระหว่างผู้เขียนและหนังสือเป็นจำนวนมาก[3]

Book level table

BookID	Title	...
1	Winnie the Pooh	...
2	The C Programming Language	...
3	The Red House Mystery	...

Author level table

AuthorID	Name	...
1	A.A. Milne	...
2	B.W. Kernigan	...
3	D.M. Ritchie	...

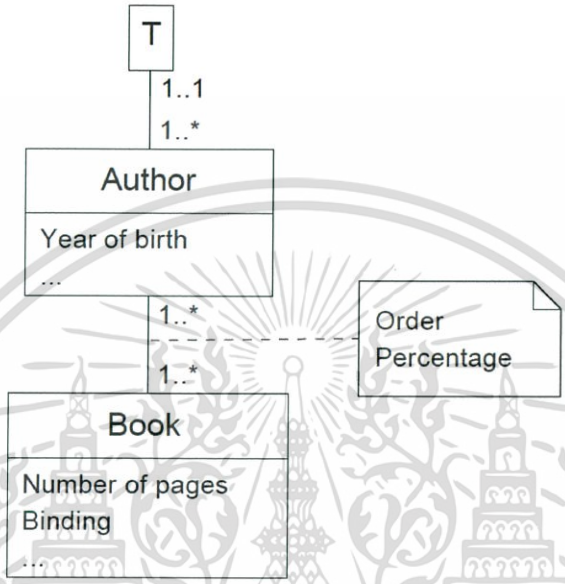
Bridge table

BookID	AuthorID
1	1
2	2
2	3
3	1

รูป 2.19 ตารางที่แสดง dimention ของ Book ใน ROLAP[3]

ด้วยวิธีที่แนะนำไม่สามารถแสดงการสั่งซื้อของผู้เขียนหนังสือและเราไม่สามารถสมมติว่าผู้แต่งได้รับการเรียงลำดับตามตัวอักษรตามลำดับตัวอักษรเสมอ ดังนั้นเราจึงอนุญาตให้ความสัมพันธ์แบบ many-to-many มีคุณสมบัติตามที่แสดงในรูปที่ 2.20 เมื่อแสดงใน ROLAP คุณสมบัติของความสัมพันธ์แบบ many-to-many จะทำให้เกิดคอลัมน์ในตารางบริดจ์ ในตัวอย่างนี้ลำดับเป็นเพียงจำนวนเต็มบวก นอกจากนี้เรายังรวมเปอร์เซ็นต์ที่แสดงว่าส่วนใหญ่เป็นของ "ผู้ถือครอง" เป็นจำนวนเท่าใด หากเรายกตัวอย่างการ roll-up ยอดขายเป็นดอลลาร์ลงในระดับผู้แต่งส่วนนี้จะคูณด้วยค่าวัดเช่นว่าเรานับจำนวนจากการขายเพียงครั้งเดียว ในรูปที่ 2.21 เราจะแสดงตัวอย่างหนังสือที่มีผู้แต่งสองคนซึ่งมีเปอร์เซ็นต์เป็น 0.5 สำหรับ

แต่ละคน หากสำเนาหนังสือเล่มเดียวจำหน่ายได้ในราคา 40 ดอลลาร์และ roll-up ยอดขายดอลลาร์ในระดับผู้เขียนเราจึงพบว่าผู้แต่งแต่ละรายทั้งสองรายสร้างยอดขายได้ 0.5*40 ดอลลาร์ = 20 ดอลลาร์ การขายครั้งเดียวเมื่อโหลดข้อมูลลงในฐานข้อมูลเป็นสิ่งสำคัญที่ค่าเหล่านี้ได้รับการกำหนดอย่างถูกต้องเช่นร้อยละที่เพิ่มขึ้นเป็น 1.0 หรือว่ามีสำหรับหนังสือที่ระบุเป็นเพียงผู้เขียนคนแรก ฯลฯ



รูปที่ 2.20 Schema สำหรับ Dimension ของหนังสือที่มีความสัมพันธ์ระหว่างผู้เขียนกับหนังสือและลำดับที่มา[3]

Book level table

BookID	Title	...
1	Winnie the Pooh	...
2	The C Programming Language	...
3	The Red House Mystery	...

Author level table

AuthorID	Name	...
1	A.A. Milne	...
2	B.W. Kernigan	...
3	D.M. Ritchie	...

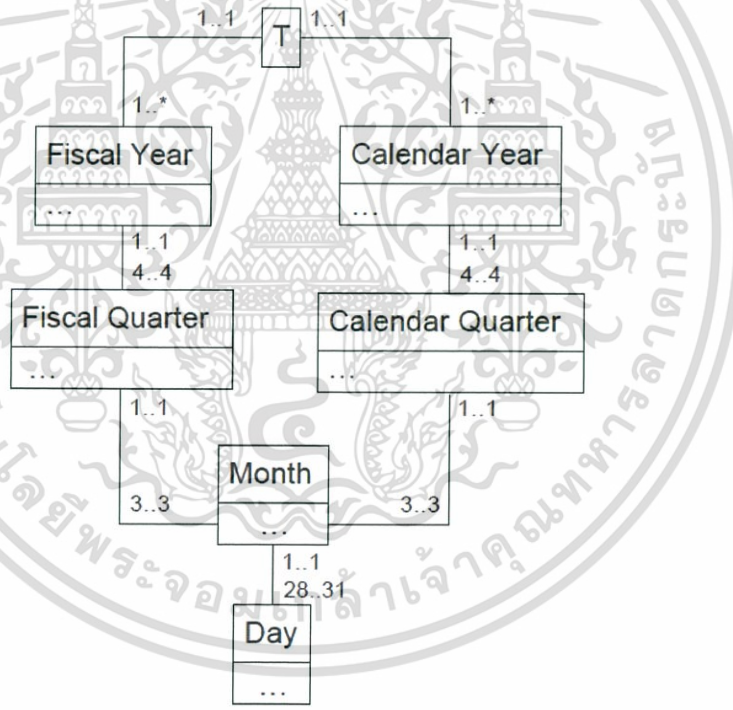
Bridge table

BookID	AuthorID	Order	Percentage
1	1	1	1.0
2	2	1	0.5
2	3	2	0.5
3	1	1	1.0

รูป 2.21 ตารางที่แสดงมิติข้อมูลหนังสือรวมทั้งคุณสมบัติในตารางบริดจ์

5) Multiple Hierarchies and Parallel Hierarchies ในหลาย ๆ กรณีความสะดวกหรือจำเป็นต้องมีมากกว่าหนึ่งลำดับชั้นใน dimension ของข้อมูล พิจารณา dimension ของเวลาที่ปรับปรุงใหม่ดังแสดงในรูปที่ 2.22 ซึ่งมีสองลำดับชั้น (ไม่แสดงคุณสมบัติของระดับ) ทั้งสองกลุ่มเป็นเดือน แต่แบ่งกลุ่มกันเป็นเดือนๆต่างกัน: ใน

หนึ่งลำดับชั้นหมายถึงปฏิทินตามปกติในขณะที่อีกกลุ่มหนึ่งหมายถึงปฏิทินการคลัง เมื่อเทียบกับระดับชั้นแบบธรรมดา เราจะขยายความคิดของเราเกี่ยวกับ dimension ของข้อมูลเพื่อให้ dimension ของข้อมูลถูกจัดเป็นลำดับชั้นเหมือนกันอย่างน้อยหนึ่งรายการที่อาจมีหรือไม่มีระดับกลาง (ระดับ T จะถูกแบ่งปันเสมอ) เมื่อมี dimension มากกว่าหนึ่งลำดับชั้นเรากล่าวว่า dimension ของข้อมูลมีลำดับชั้นหลายครั้งบ่อยครั้งไม่เหมาะที่จะใช้ลำดับชั้นหลายตัวร่วมกันในการวิเคราะห์เพียงครั้งเดียว ในการวิเคราะห์การขายหนังสือของผู้ค้าปลีกไม่ควรพิจารณา pivot table ที่แกน x แสดงไตรมาสทางการเงินและแกน Y แสดงตารางปฏิทิน ไม่น่าสนใจเนื่องจากลำดับชั้นสองมีวัตถุประสงค์ในการวิเคราะห์เดียวกัน: การจัดกลุ่มวันให้เป็นหน่วยเวลาที่หยาบกว่า ลำดับชั้นจัดกลุ่มวันแตกต่างกัน นอกจากนี้การจัดกลุ่มมีความสัมพันธ์กันมากและ pivot table ด้วยการจัดกลุ่มทั้งสองนี้บนแกนจะมีเฉพาะเซลล์ที่มีตัวเลขขวางกันเท่านั้น



รูป 2.22 Schema สำหรับข้อมูล Time Dimension ที่มีลำดับชั้นหลายแบบ[3]

6) Summarizability พิจารณาการวัดที่เป็นการเติมแต่งตามมิติข้อมูลเมื่อ aggregates ระดับสูงสามารถคำนวณจาก aggregates ระดับต่ำกว่าในลำดับชั้นในมิติเรากล่าวว่าลำดับชั้นเป็น summarizable เกี่ยวกับการวัด ตัวอย่างเช่นเราสามารถคำนวณยอดขายหนังสือทั่วประเทศของผู้จำหน่ายหนังสือทั่วประเทศได้จากยอดขายหนังสือในรัฐต่างๆและยอดขายในรัฐสามารถคำนวณได้จากยอดขายในเมืองที่

สามารถคำนวณได้จากยอดขายในแต่ละร้านค้า ดังนั้นลำดับชั้นใน Shop Dimension จึงสรุปได้โดยใช้มาตรการการขาย โดยทั่วไปแล้วลำดับชั้นควรมีความสมดุลครอบคลุมและเข้มงวดเพื่อให้มั่นใจได้ว่าสรุปได้ ความสามารถในการนำข้อมูลที่รวบรวมไว้แล้วในข้อความค้นหาใหม่ ๆ มาใช้ให้ มีศักยภาพในการเสนอข้อมูลทางคณิตศาสตร์อย่างมาก สรุปข้อมูลเป็นตัวเลขได้เร็วขึ้นในการรวม (ข้อมูลที่คำนวณแล้ว) ตัวเลขที่แสดงยอดขายในแต่ละรัฐมากกว่าที่จะต้องพิจารณาแต่ละความเป็นจริงเมื่อประมวลผลการค้นหาสำหรับจำนวนหนังสือขายทั้งหมด ระบบ OLAP สมัยใหม่ใช้ประโยชน์จากทางลัดดังกล่าวอย่างมากเพื่อให้คำตอบที่รวดเร็วโดยอิงจากมวลรวมที่ได้รับการประมวลผลล่วงหน้า

2.3.4 Data Staging Area[4]

Kimball กล่าวว่า data staging area นั้นเป็นที่สำหรับสร้างคลังข้อมูล ซึ่งจะมีหน้าที่หลักสำหรับทำการแปลงข้อมูล (data transformation) และจัดการกับข้อมูลที่ไม่ต้องการ (data cleansing) เนื่องจากเรามีแหล่งข้อมูลหลายแหล่งทำให้เราต้องใช้ data staging area เพื่อทำให้ข้อมูลเหล่านั้นอยู่ในรูปแบบเดียวกันและได้ข้อมูลถูกต้องตามที่เราต้องการ

Data Staging Area อาจมีหลายส่วนตามหน้าที่ที่ต้องทำซึ่งจะอยู่ระหว่างการ Extract และการ Load

2.3.5 Extract-Transform-Load (ETL)[4]

2.3.4.1 Extract

การดึงข้อมูลจากต้นทางจากแหล่งต่างๆนั้นมีข้อควรพิจารณาตามประเภทของแหล่งข้อมูลดังนี้

1) Source Type

- Production operational Systems คือระบบที่ทำการบันทึกรายการปัจจุบันอยู่ หรือระบบที่กำลังทำงานหลักอยู่ จะต้องมีการคำนึงถึงเรื่องระบบที่แตกต่างกัน เช่น Operating system Platforms, Hardware Platforms, File systems, Database systems
- Archives (Historical data) โดยส่วนใหญ่จะหมายข้อมูลที่ถูก back up เก็บไว้ เช่นเก็บไว้ใน tape เป็นต้น โดยเราสามารถดึงข้อมูลได้ตามระยะเวลาที่เก็บไว้ ทำให้เราได้ข้อมูลที่มีช่วงเวลาที่มาก และเมื่อทำการโหลดก็ทำการโหลดเพียงครั้งเดียว แต่อาจจะต้องคำนึงถึงวิธีการอ่านไฟล์จากข้อมูลที่ถูกเก็บเป็นแบบ Sequencial File

[4] Paulraj Ponniah. 2010. Data warehousing Fundamentals for IT Professionals 2nd Edition

- Internal file คือไฟล์เอกสารต่างๆที่บริษัทเก็บไว้ เช่น ไฟล์ Excel ไฟล์ Spreadsheets ต่างๆ รวมไปถึงเอกสารที่เป็นกระดาษ
- External data คือข้อมูลมาจากนอกองค์กรหรือนอกบริษัทของเรา บางครั้งเราจำเป็นต้องการที่จะดึงข้อมูลจากแหล่งอื่นที่ไม่ได้อยู่ภายในองค์กร

2) Mapping Data กล่าวคือ จะต้องมีการกำหนด Attributes ของ Operational database ที่ต้องการใช้ หลังจากก็ต้องมีการกำหนด Attributes ของ Data warehouse ที่ต้องการ และต้องกำหนดว่าเราต้องการแปลง Attributes ไหนใน Operational database ไปเป็น Attributes ไหนใน Data warehouse อย่างไร เช่น Map period ใน Temporal database เป็น Time dimension ใน Data warehouse รวมไปถึงมีการพิจารณารูปแบบ (Format) เช่นว่าจะ Map รูปแบบเวลาที่เป็น dd/mm/yy เป็น dd-mm-yyyy

3) Analysis เราจะต้องมีการวิเคราะห์ในเรื่องของเทคโนโลยีของแหล่งข้อมูลต้นทางว่ามีการใช้เทคโนโลยีไหนอย่างไร มีการพิจารณาว่าข้อมูลที่แหล่งข้อมูลนั้นเป็นข้อมูลปัจจุบันหรือข้อมูลในอดีตที่ถูกเก็บไว้ รวมถึงการวิเคราะห์ว่าข้อมูลจากแหล่งนั้นๆมีความน่าเชื่อถือแค่ไหน คุณภาพของข้อมูลเป็นอย่างไรและใครเป็นเจ้าของ และต้องพิจารณาด้วยว่าขนาดของข้อมูลใหญ่มากน้อยแค่ไหน และสุดท้ายคือการพิจารณาว่าแหล่งข้อมูลนั้นๆมีช่วงเวลาใดบ้างที่เราสามารถเข้าถึงและดึงข้อมูลโดยที่ไม่รบกวนการทำงานของระบบ

2.3.4.2 Transform

1)Transformation Types

- Cleaning จะทำการ Map ค่า Null เป็นค่าอื่นๆที่ถูกต้อง เช่น ถ้าเป็น Period time แล้วมี Null ที่ Start time เราจะต้องบอกให้ได้ว่า Null นั้นควรเป็นวันที่เท่าไหร่
- Deduplication เราต้องหาให้ได้ว่ามีข้อมูลซ้ำกันหรือไม่ ถ้าเกิดมีข้อมูลซ้ำให้เราเลือกข้อมูลที่ซ้ำเพียงอันเดียว หรือถูกต้องเพียงอันเดียว ถ้าเป็น Temporal database จะต้องมีการตรวจสอบ Duplicates ทั้ง 4 แบบ (Sequenced, Current, Value-equivalent, Nonsequenced)
- Data/Time conversion จะต้องมีการปรับ Format ของเวลาให้อยู่ในรูปแบบเดียวกันเช่นแปลงจาก Format 11/10/2008 เป็น 11 OCT 2008
- Conversion of Units of Measurements คือจะต้องมีการแปลงหน่วยของสิ่งต่างๆให้อยู่ในหน่วยเดียวกันหรือปรับค่าให้อยู่ในระดับของการวัดที่

เท่ากัน เช่น ถ้าเป็นเวลาที่เป็น Timestamp หากเราต้องการเป็นวันที่ (Date)ก็ต้องมีการปรับเป็นวันที่

- Key restructuring เนื่องจากเรามีข้อมูลมาจากหลายแหล่ง เป็นไปได้ว่าอาจจะมี Key ที่ไม่เหมือนกันในเรื่องเดียวกัน เพื่อให้ง่ายเราจะต้องมีการ Map key เหล่านั้นแล้วนำมาสร้างเป็น key ใหม่

2.3.4.3 Load

1) Load คือ ถ้าตารางที่ต้องการ (Fact tables, Dimension tables) ที่เราต้องการโหลดข้อมูลไปใส่นั้นมีข้อมูลอยู่แล้ว เมื่อเราทำการโหลด ข้อมูลเดิมจะหายไปและถูกแทนที่ด้วยข้อมูลที่ถูกโหลดเข้าไปใหม่ หรือถ้าตารางที่ต้องการไม่มีข้อมูลอยู่เลย เมื่อทำการโหลดข้อมูลทั้งหมดจะถูกนำไปใส่ในตาราง

2) Append คือ ถ้าตารางที่ต้องการ (Fact tables, Dimension tables) ที่เราต้องการโหลดข้อมูลไปใส่นั้นมีข้อมูลอยู่แล้วเมื่อทำการโหลด ข้อมูลใหม่ที่โหลดเข้าไปจะถูกนำไปเพิ่มในตารางเดิม เมื่อมีข้อมูลที่ซ้ำกัน (Duplicate) เราจะต้องกำหนดว่าจะยอม(Accept)หรือไม่ยอม (Reject)ให้ข้อมูลที่ซ้ำกันนั้นถูก Load เข้าไปหรือไม่

3) Destructive Marge คือถ้าข้อมูลที่จะโหลดเข้าไปใหม่ปรากฏว่ามี Primary key ซ้ำกันกับข้อมูลที่มีอยู่เดิม ก็จะทำการ Update ข้อมูลที่มีอยู่เดิมให้เป็นข้อมูลอันใหม่ ถ้าข้อมูลที่โหลดเข้าไปใหม่ไม่ปรากฏว่ามี Primary key อยู่ในข้อมูลเดิมให้ทำการเพิ่มข้อมูลอันใหม่เข้าไป

4) Constructive Marge คือ ถ้าข้อมูลที่จะโหลดเข้าไปใหม่ปรากฏว่ามี Primary key ซ้ำกันกับข้อมูลที่มีอยู่เดิมจะทำการนำข้อมูลเดิมออกไป และแทนที่ด้วยข้อมูลใหม่ พร้อมทั้งมีการบอกว่าอันที่เข้าไปใหม่นี้เป็น Version ใหม่ ถ้าข้อมูลที่โหลดเข้าไปใหม่ไม่ปรากฏว่ามี Primary key อยู่ในข้อมูลเดิมให้ทำการเพิ่มข้อมูลอันใหม่เข้าไป

บทที่ 3
การออกแบบและพัฒนาซอฟต์แวร์

3.1 การทดสอบตัวดำเนินการทางเวลาใน Oracle Database 12c[5]

3.1.1 การทดสอบตัวดำเนินการเกี่ยวกับการกำหนดนิยามของตาราง

3.1.1.1 การทดสอบสร้างตารางที่มี Valid-time

ในฐานข้อมูลเชิงเวลานั้นจะมีการเก็บข้อมูลที่เกี่ยวข้องกับเวลาและจะมีการเก็บค่าของช่วงเวลา (Period) ดังนั้นการดำเนินการสอบถามข้อมูลในฐานข้อมูลเชิงเวลานั้นจะต้องมีส่วนระบุค่าของเวลาหรือช่วงเวลาที่ต้องการ สำหรับตัวอย่างฐานข้อมูลที่จะใช้เป็นตัวอย่างเพื่อทดสอบการสอบถามข้อมูลเชิงเวลานั้นจะเป็นฐานข้อมูลเกี่ยวฟาร์มปศุสัตว์ที่แสดงรายละเอียดของฝูงวัวที่ละฝูงว่าอยู่ในลานให้อาหารใด ฝูงใด คอกย่อยใด มีจำนวนเท่าใด และช่วงเวลาที่อาศัยอยู่ ซึ่งจะใช้ตัวดำเนินการที่อยู่ใน Oracle Database 12c SQL

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

รูป 3.1 แสดงรายละเอียดของฝูงวัวแต่ละฝูงในช่วงเวลาต่างๆ

จากรูป 3.1 นั้นคอลัมน์ FDYD คือ หมายเลขของลานให้อาหาร คอลัมน์ LOT_ID_NUM คือ หมายเลขของฝูง คอลัมน์ PEN_ID คือ หมายเลขของคอกย่อย คอลัมน์ HD_CNT คือจำนวนวัว และคอลัมน์ FROM_DATE TO_DATE เป็นเวลาเริ่มต้นและเวลาสิ้นสุดที่ฝูงวัวอยู่ในฝูงนั้น โดยจะมีการกำหนดให้ FROM_DATE และ TO_DATE เป็นช่วงเวลาตามมาตรฐานเอสคิวแอล 2011

[5] Mary Beth Roeser. January 2016. Oracle Database SQL Language Reference, 12c Release 1 (12.1)

โปรแกรม 3.1 การทดสอบสร้างตารางที่มี Valid time

```
CREATE TABLE LOT_LOC (
      FDYD INTEGER,
      LOT_ID_NUM INTEGER,
      PEN_ID INTEGER,
      HD_CNT INTEGER,
      FROM_DATE DATE,
      TO_DATE DATE,
      PERIOD FOR VALID_TIME (FROM_DATE, TO_DATE)
);
```

ผลลัพธ์ที่ได้

รูป 3.2 แสดงผลลัพธ์เมื่อใช้คำสั่งสร้างตารางที่มีแวร์ลิดไทม์

จากการทดสอบพบว่าโปรแกรม 3.1 สามารถสร้างตารางที่มี Valid-time ได้ ซึ่งมีการใช้คำสั่ง PERIOD FOR เพื่อกำหนดช่วงเวลา (Valid-time) มีการตั้งชื่อช่วงเวลา เวลาเริ่มต้นและเวลาสิ้นสุด

3.1.1.2 การประกาศ Primary key ในคำสั่งสร้างตาราง

ในหัวข้อนี้จะทำการทดสอบการประกาศ Primary key และทดสอบการทำงานว่าการประกาศ Primary key นั้นสามารถทำงานที่เกี่ยวข้องกับช่วงเวลาได้หรือไม่

โปรแกรม 3.2 การทดสอบประกาศ Primary key ในคำสั่งสร้างตาราง แบบที่ 1

```
CREATE TABLE LOT_LOC (
      FDYD INTEGER,
      LOT_ID_NUM INTEGER,
      PEN_ID INTEGER,
      HD_CNT INTEGER,
      FROM_DATE DATE,
      TO_DATE DATE,
      PERIOD FOR VALID_TIME (FROM_DATE, TO_DATE),
      PRIMARY KEY (FDYD, LOT_ID_NUM, PEN_ID, VALID_TIME
WITHOUT OVERLAPS)
);
```

ผลลัพธ์ที่ได้

```
ORA-00907: missing right parenthesis
```

รูป 3.3 แสดงผลลัพธ์เมื่อใช้คำสั่งสร้างตารางที่มีการประกาศ Primary key แบบที่ 1

จากการทดสอบพบว่าคำสั่งดังกล่าวนี้ไม่สามารถใช้กับ Oracle database 12c ได้ เนื่องจากระบบจัดการฐานข้อมูลตีความคำว่า OVERLAPS ในคำสั่งสร้างตารางเป็น binary operator ซึ่งต้องการตัวถูกดำเนินการ 2 ตัว (ผิด Syntax ของ Oracle)

โปรแกรม 3.3 การทดสอบประกาศ Primary key ในคำสั่งสร้างตาราง แบบที่ 2

```
CREATE TABLE LOT_LOC (
      FDYD INTEGER,
      LOT_ID_NUM INTEGER,
      PEN_ID INTEGER,
      HD_CNT INTEGER,
      FROM_DATE DATE,
      TO_DATE DATE,
      PERIOD FOR VALID_TIME (FROM_DATE, TO_DATE),
      PRIMARY KEY (FDYD, LOT_ID, PEN_ID, VALID_TIME)
);
```

ผลลัพธ์ที่ได้

Results Explain Describe Saved SQL History

```
Table created.
```

รูป 3.4 แสดงผลลัพธ์เมื่อใช้คำสั่งสร้างตารางที่มีการประกาศ Primary key แบบที่ 2

จากการทดสอบพบว่าคำสั่งดังกล่าวนี้สามารถใช้ได้กับ Oracle database 12c โดยจะมีการเพิ่ม VALID_TIME ซึ่งเป็น period definition เข้าไปเป็น primary key

เมื่อทดสอบกับข้อมูลตัวอย่างทำให้แน่ใจได้ว่าการกำหนด primary key แบบนี้ระบบจัดการฐานข้อมูลไม่ได้มีการตรวจสอบการซ้อนทับกันของช่วงเวลาให้ และผลการทดลองยังยืนยันว่าการเพิ่ม period definition เข้าไปประกอบเป็น primary key นั้นไม่มีผลใดๆ

โปรแกรม 3.4 ทดสอบเพิ่มข้อมูลเพื่อตรวจสอบการซ้อนทับกันของแวร์ลิดไทม์ที่ประกาศไว้

```
INSERT INTO LOT_LOC VALUES (1, 137, 1, 17, DATE '1998-02-
07', DATE '1998-02-18');
```

โปรแกรม 3.5 ทดสอบเพิ่มข้อมูลเพื่อตรวจสอบการซ้อนทับกันของแวร์ลิดไทม์ที่ประกาศไว้

```
INSERT INTO LOT_LOC VALUES (1, 137, 1, 17, DATE '1998-02-
10', DATE '1998-02-18');
```

โปรแกรม 3.6 ทดสอบเพิ่มข้อมูลเพื่อตรวจสอบการซ้อนทับกันของแวร์ลิดไทม์ที่ประกาศไว้

```
INSERT INTO LOT_LOC VALUES (1, 137, 1, 17, DATE '1999-02-
07', DATE '1999-02-18');
```

ผลลัพธ์ที่ได้จากโปรแกรม 3.5 และ 3.6

```
ORA-00001: unique constraint (MARK.SYS_C0021850) violated
```

รูป 3.5 แสดงผลลัพธ์เมื่อใช้คำสั่งเพิ่มข้อมูลเพื่อทดสอบการซ้อนทับกันของแวร์ลิดไทม์

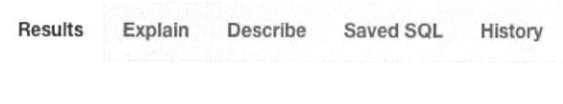
3.1.1.3 การทดสอบประกาศ Foreign key ในคำสั่งสร้างตาราง

การทดสอบในหัวข้อนี้จะใช้ตัวอย่างตารางพนักงานและตารางแผนกที่สังกัดของพนักงาน โดยเริ่มต้นจากการประกาศสร้างตารางแผนกที่สังกัดของพนักงานและตามด้วยการสร้างตารางพนักงาน

โปรแกรม 3.7 การทดสอบประกาศ Foreign key ในคำสั่งสร้างตาราง (ตารางแผนก)

```
CREATE TABLE DEPT(
      DNO INTEGER,
      DSTART DATE,
      DEND DATE,
      DNAME VARCHAR (30),
      PERIOD FOR DPERIOD (DSTART, DEND),
      PRIMARY KEY (DNO, DPERIOD)
);
```

ผลลัพธ์ที่ได้



รูป 3.6 ผลลัพธ์เมื่อใช้คำสั่งสร้างตาราง (ตารางแผนก)

โปรแกรม 3.8 การทดสอบประกาศ Foreign key ในคำสั่งสร้างตาราง (ตารางพนักงาน) แบบที่ 1

```
CREATE TABLE EMP(
      ENo INTEGER,
      ES DATE,
      EE DATE,
      EDept INTEGER,
      PERIOD FOR EP (ES, EE),
      PRIMARY KEY (ENO, EP),
      FOREIGN KEY (Edept, PERIOD EP)
            REFERENCES DEPT (DNO, PERIOD DPERIOD)
);
```

ผลลัพธ์ที่ได้

รูป 3.7 ผลลัพธ์เมื่อใช้คำสั่งสร้างตารางที่มีการประกาศ Foreign key แบบที่ 1

จากการทดลองตามโปรแกรม 3.7 และ 3.8 พบว่าไม่สามารถสร้างตารางที่กำหนด Foreign key แบบนี้ได้ โดยระบบจัดการฐานข้อมูลตีความคำว่า PERIOD เป็น binary operator ซึ่งผิด syntax ของ Oracle database 12c

โปรแกรม 3.9 การทดสอบประกาศ Foreign key ในคำสั่งสร้างตาราง (ตารางพนักงาน) แบบที่ 2

```
CREATE TABLE EMP(
      ENo INTEGER,
      ES DATE,
      EE DATE,
      EDept INTEGER,
      PERIOD FOR EP (ES, EE),
      PRIMARY KEY (ENO, EP),
      FOREIGN KEY (Edept, EP)
           REFERENCES DEPT (DNO, DPERIOD)
);
```

ผลลัพธ์ที่ได้

รูป 3.8 ผลลัพธ์เมื่อใช้คำสั่งสร้างตารางที่มีการประกาศ Foreign key แบบที่ 2

เมื่อลบคำว่า PERIOD ใน โปรแกรม 3.8 ที่มีปัญหาออกไป ทำให้เราสามารถสร้างตารางที่มี Foreign key ตามรูปแบบนี้ได้

แต่การทดลองจากหัวข้อ 3.1.1.2 เรื่องการประกาศ Primary key นั้นได้ยืนยันว่าการเพิ่ม Period definition เข้าไปประกอบเป็น Primary key นั้นไม่มีผลใดๆ ดังนั้นจึงสรุปได้ว่าการเพิ่ม Period definition เข้าไปในการประกาศ Foreign key ไม่มีผลเช่นกันกับการประกาศ Primary key

3.1.1.4 การทดสอบใช้คำสั่งแก้ไขตาราง

ในกรณีที่มีตารางเดิมอยู่แล้วซึ่งตารางนั้นไม่ได้มีการกำหนดช่วงเวลาไว้ (period definitions) เราสามารถทำการกำหนดช่วงเวลาได้หลังจากสร้างตารางแล้วผ่านประโยคคำสั่ง ALTER TABLE

โปรแกรม 3.10 การกำหนดช่วงเวลาในกรณีที่ไม่ได้กำหนดไว้ตอนสร้างตาราง

```
ALTER TABLE table_name ADD PERIOD FOR user_valid_time;
```

จากโปรแกรมที่ 3.10 นั้น table_name คือ ชื่อตารางที่เราต้องการเพิ่มช่วงเวลาที่เป็นแวลิดไทม์และ user_valid_time คือ ชื่อของช่วงเวลาที่เราต้องการ เมื่อที่รันคำสั่งในโปรแกรมที่ 3.3 แล้วจะมีการสร้างคอลัมน์ในตารางที่เราต้องการ 2 คอลัมน์คือ USER_VALID_TIME_START

และ USER_VALID_TIME_END แต่ทั้งสองคอลัมน์นั้นจะถูกซ่อนไว้ ซึ่งสามารถเพิ่มข้อมูลลงตารางโดยค่าค่าของคอลัมน์ทั้งสองได้ แต่สองคอลัมน์นั้นจะไม่ปรากฏในการแสดงผลลัพธ์ (Output) ผ่านคำสั่ง SELECT นอกจากจะมีการกำหนดชื่อของคอลัมน์ทั้งสองลงไป

3.1.2 การทดสอบตัวดำเนินการสำหรับเพิ่มข้อมูลลงตาราง

สำหรับฐานข้อมูลเชิงเวลาที่สร้างใน Oracle Database เราสามารถเพิ่มข้อมูลเชิงเวลาที่มีแวร์ลิดไทม์ ประกอบไปด้วยเวลาเริ่มต้นและเวลาสิ้นสุดเข้าไปในตารางได้ ดังโปรแกรม 3.11

โปรแกรม 3.11 การเพิ่มข้อมูลลงตารางที่มี Valid time

```
INSERT INTO LOT_LOC VALUES (1, 137, 1, 17, DATE '1998-02-
07', DATE '1998-02-18');
INSERT INTO LOT_LOC VALUES (1, 219, 1, 43, DATE '1998-02-
25', DATE '1998-03-01');
INSERT INTO LOT_LOC VALUES (1, 219, 1, 20, DATE '1998-03-
01', DATE '1998-03-14');
INSERT INTO LOT_LOC VALUES (1, 219, 2, 23, DATE '1998-03-
01', DATE '1998-03-14');
INSERT INTO LOT_LOC VALUES (1, 219, 2, 43, DATE '1998-03-
14', DATE '9999-12-31');
INSERT INTO LOT_LOC VALUES (1, 374, 1, 14, DATE '1998-02-
20', DATE '9999-12-31');
```

3.1.3 การทดสอบตัวดำเนินการสำหรับปรับปรุงข้อมูลในตาราง

คำสั่ง UPDATE แบบธรรมดายังสามารถนำมาใช้ปรับปรุงข้อมูลในแต่ละแถวของตารางแบบ valid-time period ได้ (การปรับเปลี่ยนข้อมูลที่ว่านี้นับรวมถึงการปรับเปลี่ยนค่าในคอลัมน์ของเวลาเริ่มต้นและเวลาสิ้นสุดด้วย)

โปรแกรม 3.12 ทดสอบคำสั่งปรับปรุงข้อมูลในตาราง แบบที่ 1

```
UPDATE LOT_LOC SET HD_CNT = 18 ,FROM_DATE = DATE '1990-01-
01' WHERE FDYD = 1 AND LOT_ID_NUM = 137 AND PEN_ID = 1;
```

ผลลัพธ์ที่ได้

รูป 3.9 ผลลัพธ์เมื่อใช้คำสั่งปรับปรุงข้อมูลในตาราง แบบที่ 1

จากโปรแกรมที่ 3.12 เป็นการปรับปรุงข้อมูลในตารางที่มี Valid time ซึ่งเราสามารถใช้คำสั่งปรับปรุงข้อมูล (UPDATE) แบบเดิมเพื่อแก้ไขข้อมูลในคอลัมน์ต่างๆ ได้รวมถึงคอลัมน์ที่เป็นเวลาเริ่มต้นและเวลาสิ้นสุด

โปรแกรม 3.13 ทดสอบคำสั่งปรับปรุงข้อมูลในตาราง แบบที่ 2

```
UPDATE LOT_LOC FOR PORTION OF VALID_TIME FROM DATE '1990-
02-03' TO DATE '1997-09-10' SET PEN_ID = 2 WHERE FDYD = 1
AND LOT_ID_NUM = 137 AND PEN_ID = 1;
```

ผลลัพธ์ที่ได้

รูป 3.10 ผลลัพธ์เมื่อใช้คำสั่งปรับปรุงข้อมูลในตาราง แบบที่ 2

จากโปรแกรม 3.13 นั้นเป็นการทดสอบคำสั่งที่มีรูปแบบเหมือนมาตรฐาน SQL:2011 พบว่าระบบจัดการฐานข้อมูล Oracle Database 12c ไม่สามารถปรับปรุงข้อมูลให้เราได้ เนื่องไม่รู้จักคำว่า PORTION ซึ่งเป็นคำสั่งการปรับปรุงข้อมูลในช่วงเวลาใดๆ

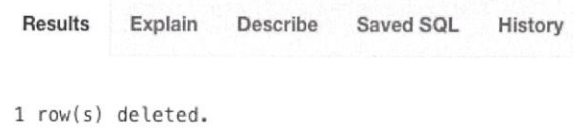
3.1.4 การทดสอบตัวดำเนินการสำหรับลบข้อมูลในตาราง

คำสั่ง DELETE แบบธรรมดายังสามารถนำมาใช้ลบข้อมูลในแต่ละแถวของตารางแบบ valid-time period ได้

โปรแกรม 3.14 ทดสอบคำสั่งลบข้อมูลในตาราง แบบที่ 1

```
DELETE LOT_LOC WHERE FDYD = 1 AND LOT_ID_NUM = 137 AND
PEN_ID = 1;
```

ผลลัพธ์ที่ได้

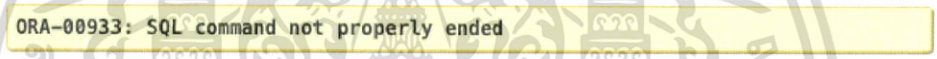


รูป 3.11 ผลลัพธ์เมื่อใช้คำสั่งลบข้อมูลในตาราง แบบที่ 1

โปรแกรม 3.15 ทดสอบคำสั่งลบข้อมูลในตาราง แบบที่ 2

```
DELETE LOT_LOC FOR PORTION OF VALID_TIME FROM DATE '1990-
02-03' TO DATE '1997-09-10' WHERE FDYD = 1 AND LOT_ID_NUM
= 137 AND PEN_ID = 1;
```

ผลลัพธ์ที่ได้



รูป 3.12 ผลลัพธ์เมื่อใช้คำสั่งลบข้อมูลในตาราง แบบที่ 2

จากการทดสอบในโปรแกรม 3.15 ซึ่งเป็นรูปแบบคำสั่งเดียวกันกับมาตรฐาน SQL:2011 พบว่าระบบจัดการฐานข้อมูล Oracle Database 12c ไม่สามารถทำการลบข้อมูลที่ต้องการได้ เนื่องจากมีการใช้รูปแบบของคำสั่งที่ไม่เหาะสม ไม่ถูกต้อง

เมื่อวิเคราะห์แล้วน่าจะมาจากการใช้คำว่า FOR ในประโยคคำสั่ง DELELE ซึ่งจริงๆแล้วสามารถใช้ได้หากตามหลังคำว่า PARTITION หรือคำว่า SUBPARTITION

3.1.5 การทดสอบตัวดำเนินการสำหรับสอบถามข้อมูลในตาราง

3.1.5.1 ตัวดำเนินการ OVERLAPS

เป็นคำสั่งแบบ Binary operator เพื่อช่วยในการตรวจสอบการซ้อนทับกันของช่วงเวลา 2 ช่วงเวลา โดยจะต้องมีการกำหนดค่าของเวลาเริ่มต้นและเวลาสิ้นสุดของช่วงเวลาทั้งสองด้วย

โปรแกรม 3.16 การใช้ Operation OVERLAPS

```
SELECT * FROM LOT_LOC WHERE (FROM_DATE, TO_DATE) OVERLAPS
(DATE '1998-03-01', DATE '1998-03-05');
```

ผลลัพธ์ที่ได้

FDYD	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	374	1	14	02/20/1998	12/31/9999
1	219	1	20	03/01/1998	03/14/1998
1	219	2	23	03/01/1998	03/14/1998

รูป 3.13 แสดงผลลัพธ์เมื่อใช้ตัวดำเนินการ OVERLAPS

3.1.5.2 Flashback Query (AS OF PERIOD)

คุณสมบัตินี้ใช้เพื่อดึงข้อมูลในอดีตหรือในปัจจุบันก็ได้ โดยจะคืนค่าของแถวที่มีเวลาที่กำหนดอยู่ในช่วงเวลาของแถวนั้นๆ ซึ่งการกำหนดค่าของเวลาที่ต้องการจะกำหนดผ่านคำสั่ง AS OF ในประโยคคำสั่ง SELECT ซึ่งสามารถใช้ทดแทนตัวดำเนินการ CONTAINS ได้

โปรแกรม 3.17 การใช้คำสั่ง AS OF เพื่อสอบถามข้อมูลตามวันที่ต้องการ

```
SELECT * FROM LOT_LOC AS OF PERIOD FOR VALID_TIME DATE
'1998-02-08';
```

ผลลัพธ์ที่ได้

FDYD	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1	137	1	17	02/07/1998	02/18/1998

รูป 3.14 แสดงผลลัพธ์เมื่อใช้คำสั่ง AS OF กับ Valid time ในวันที่ต้องการ

โปรแกรม 3.18 การใช้คำสั่ง AS OF เพื่อสอบถามข้อมูลตามวันปัจจุบัน

```
SELECT * FROM LOT_LOC AS OF PERIOD FOR VALID_TIME
SYSDATE;
```

ผลลัพธ์ที่ได้

	FDYD	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1		374	1	14	02/20/1998	12/31/9999
1		219	2	43	03/14/1998	12/31/9999

รูป 3.15 แสดงผลลัพธ์เมื่อใช้คำสั่ง AS OF กับ Valid time ในวันปัจจุบัน

3.1.5.3 Flashback Version Query (VERSIONS PERIOD)

คุณสมบัตินี้ใช้เพื่อดึงข้อมูลในช่วงเวลาต่างๆ โดยมีการกำหนดค่าของช่วงเวลาที่ต้องการ เช่น ต้องการดูแถวทั้งหมดที่มีช่วงของเวลาที่กำหนดอยู่ในแวลิดไทม์ เป็นต้น ซึ่งสามารถใช้งานผ่านคำสั่ง VERSIONS ในประโยคคำสั่ง SELECT ซึ่งสามารถใช้ทดแทนตัวดำเนินการ OVERLAPS ได้

โปรแกรม 3.19 การใช้คำสั่ง VERSIONS เพื่อสอบถามข้อมูลในช่วงเวลาที่กำหนด

```
SELECT * FROM LOT_LOC VERSIONS PERIOD FOR VALID_TIME
BETWEEN DATE '1998-03-01' AND DATE '1998-03-05';
```

ผลลัพธ์ที่ได้

	FDYD	LOT_ID_NUM	PEN_ID	HD_CNT	FROM_DATE	TO_DATE
1		374	1	14	02/20/1998	12/31/9999
1		219	1	20	03/01/1998	03/14/1998
1		219	2	23	03/01/1998	03/14/1998

รูป 3.16 แสดงผลลัพธ์เมื่อใช้คำสั่ง VERSIONS กับ Valid time ในช่วงเวลาที่กำหนด

3.2 เปรียบเทียบการดำเนินการทางเวลาใน Oracle Database 12c กับมาตรฐานเอสคิวแอล 2011[5]

เนื่องจากระบบจัดการฐานข้อมูล Oracle Database 12c นั้นยังไม่สนับสนุนการทำงานที่สอดคล้องตามมาตรฐานเอสคิวแอล 2011 ทั้งหมดจึงต้องทำการสรุปข้อเปรียบเทียบความสามารถในการจัดข้อมูลเชิงเวลาของระบบจัดการฐานข้อมูล Oracle Database 12c ว่ามีคุณสมบัติใดบ้างที่ทำงานสอดคล้องตามมาตรฐานเอสคิวแอล 2011

3.2.1 การประกาศนิยามตารางของข้อมูลเชิงเวลา

ในหัวข้อนี้จะแสดงคำสั่งที่ระบบจัดการฐานข้อมูล Oracle Database 12c สามารถสนับสนุนการทำงานที่สอดคล้องตามมาตรฐานเอสคิวแอล 2011 ในเรื่องของการสร้างตารางที่มี valid time สำหรับข้อมูลเชิงเวลา

โปรแกรม 3.20 การสร้างตารางที่มี Valid time ตามมาตรฐานเอสคิวแอล 2011

```
CREATE TABLE LOT_LOC (
      FDYD INTEGER,
      LOT_ID_NUM INTEGER,
      PEN_ID INTEGER,
      HD_CNT INTEGER,
      FROM_DATE DATE,
      TO_DATE DATE,
      PERIOD FOR VALID_TIME (FROM_DATE, TO_DATE)
);
```

จากโปรแกรมที่ 3.20 นั้นคำสั่งจะเหมือนกันกับโปรแกรมที่ 3.1 โดยคำสั่งในการสร้างตารางนี้เราได้เพิ่มคำสั่ง PERIOD FOR เข้าไปเพื่อสร้างคอลัมน์ของช่วงเวลาที่เป็น valid time มีการตั้งชื่อของช่วงเวลา และมีการตั้งชื่อของเวลาเริ่มต้นกับเวลาสิ้นสุดไว้ด้วย

จากการทดลองสร้างตารางของฐานข้อมูลเชิงเวลาตามการกำหนดนิยามของตารางในโปรแกรม 3.1 พบว่าระบบจัดการฐานข้อมูลออราเคิลสามารถประกาศนิยามของช่วงเวลา (Time Period) ในคำสั่งสร้างตาราง (CREATE TABLE) ได้เหมือนกับมาตรฐานเอสคิวแอล 2011 สามารถตั้งชื่อช่วงเวลาได้ตามต้องการ รวมถึงสามารถตั้งชื่อเวลาเวลาเริ่มต้นและเวลาสิ้นสุดได้ตามต้องการ ทั้งหมดนี้สามารถทำได้เหมือนกันในรูปแบบของโครงสร้างภาษาเลย (Syntax เหมือนกัน)

ระบบจัดการฐานข้อมูลออราเคิลจะมีการตรวจสอบว่าเวลาสิ้นสุดต้องมีค่ามากกว่าหรือต้องมาหลังจากเวลาเริ่มต้น ถ้าหากเวลาสิ้นสุดมาหลังจากเวลาเริ่มต้นแล้ว ระบบจัดการฐานข้อมูลจะตีความช่วงเวลาดังกล่าวเป็นรูปแบบ close-open

3.2.1.1 การประกาศ Primary keys

จากการทดลองพบว่าระบบจัดการฐานข้อมูลไม่ได้มีการตรวจสอบการซ้อนทับกันของช่วงเวลาให้ตามมาตรฐาน SQL:2011 และผลการทดลองยังยืนยันว่าการเพิ่ม Period definition เข้าไปประกอบเป็น Primary key นั้นไม่มีผลใดๆ

หลังจากศึกษาคู่มือของ Oracle ยังไม่พบรูปแบบคำสั่งกำหนด Primary key ที่สามารถตรวจสอบการซ้อนทับกันของช่วงเวลาได้สะดวกเหมือนมาตรฐาน SQL:2011

3.2.1.2 การประกาศ Foriegn keys

จากการทดสอบในหัวข้อ 3.1.1.3 เรื่องการประกาศ Foreign keys ทั้งสองแบบนั้น พบว่าระบบจัดการฐานข้อมูล Oracle ยังไม่รองรับการประกาศ Foreign keys ที่มีการตรวจสอบความสอดคล้องกันของช่วงเวลาระหว่างตารางหลักและตารางรอง เมื่อทดลองต่อไปพบว่าถึงจะแก้ปัญหาเรื่องรูปแบบของคำสั่งแล้ว การทำงานของระบบจัดการฐานข้อมูล Oracle ยังไม่มีการทำงานที่สอดคล้องตามมาตรฐาน SQL:2011

หลังจากศึกษาคู่มือของ Oracle ยังไม่พบรูปแบบคำสั่งประกาศ Foreign key ที่สามารถตรวจสอบการซ้อนทับกันของช่วงเวลาได้สะดวกเหมือนมาตรฐาน SQL:2011

3.2.1.3 คำสั่งแก้ไขตาราง (ALTER TABLE)

จากการทดสอบในหัวข้อ 3.1.1.4 เรื่องการใช้คำสั่งแก้ไขตาราง ALTER TABLE พบว่าระบบจัดการฐานข้อมูล Oracle สามารถทำได้เพียงการแก้ไขในเรื่องของการเพิ่มช่วงเวลา (Period definition) ได้เท่านั้น กล่าวคือยังไม่รองรับการแก้ไขที่เกี่ยวข้องกับ Primary key และ Foreign key

3.2.2 การเพิ่มข้อมูลเชิงเวลา (Insert temporal data)

ในส่วนของการเพิ่มข้อมูลลงฐานข้อมูลตามมาตรฐานเอสคิวแอล 2011 นั้นไม่ได้การกำหนดคุณสมบัติเพิ่มเติม ซึ่งสามารถใช้คำสั่งเพิ่มข้อมูลแบบเดิมได้

จากการทดลองในหัวที่ 3.1.2 พบว่าสามารถเพิ่มข้อมูล (Insert) ลงฐานข้อมูลเชิงเวลาที่ได้มีการกำหนดนิยามของตารางตามโปรแกรม 3.1 โดยใช้คำสั่งเพิ่มข้อมูลแบบเดิม และจากการค้นคว้ายังไม่พบคุณสมบัติเพิ่มเติมเกี่ยวกับเพิ่มข้อมูลเชิงเวลาใน Oracle database ที่จะทำให้ใช้งานได้สะดวกขึ้นหรือเทียบเท่าคุณสมบัติตามมาตรฐาน SQL:2011

โปรแกรม 3.21 ตัวอย่างการเพิ่มข้อมูลลงฐานข้อมูลเชิงเวลา

```
INSERT INTO LOT_LOC VALUES (1, 137, 1, 17, DATE '1998-02-
07', DATE '1998-02-18');
```

3.2.3 การปรับปรุงข้อมูลเชิงเวลา (Update temporal data)

คุณสมบัติใหม่ในมาตรฐานเอสคิวแอล 2011 นั้นมีความสามารถในการปรับปรุงข้อมูลที่มีผลกระทบจากช่วงเวลาที่กำหนดได้ด้วย โดยคุณสมบัตินี้เป็นส่วนขยายของคำสั่ง UPDATE

จากการทดสอบใช้คำสั่งการปรับปรุงแก้ไขข้อมูลเชิงเวลาในระบบจัดการฐานข้อมูล Oracle Database 12c นั้นยังไม่สนับสนุนการทำงานที่สอดคล้องตามมาตรฐานเอสคิวแอล 2011 ในส่วนที่จะมีการต้องมีการทำ UPDATE triggers และ INSERT triggers

จากการค้นคว้าหาข้อมูลยังไม่พบคุณสมบัติอื่นๆใน Oracle Database 12c ที่เพิ่มความสะดวกในการปรังปรุงข้อมูลเชิงเวลาหรือมีความสามารถเทียบเท่ามาตรฐาน SQL:2011

3.2.4 การลบข้อมูลเชิงเวลา (Delete temporal data)

คุณสมบัติใหม่ในมาตรฐานเอสคิวแอล 2011 นั้นมีความสามารถในการลบข้อมูลที่มีผลกระทบจากช่วงเวลาที่กำหนดได้ด้วย โดยคุณสมบัตินี้เป็นส่วนขยายของคำสั่ง DELETE ที่ผู้ใช้มีการกำหนดช่วงเวลาที่สนใจ

จากการทดสอบใช้คำสั่งลบข้อมูลเชิงเวลาในระบบจัดการฐานข้อมูล Oracle Database 12c นั้นยังไม่สนับสนุนการทำงานที่สอดคล้องตามมาตรฐานเอสคิวแอล 2011 ในส่วนที่จะต้องมีการทำ DELETE triggers และ INSERT triggers และจากการค้นคว้าหาข้อมูลยังไม่พบคุณสมบัติอื่นๆใน Oracle database ที่เพิ่มความสะดวกในการลบข้อมูลเชิงเวลาหรือมีความสามารถเทียบเท่ามาตรฐาน SQL:2011

3.2.5 การสอบถามข้อมูลเชิงเวลา (Query temporal data)

ในส่วนของการสอบถามข้อมูลเชิงเวลานั้นจะแบ่งออกเป็นการสอบถามข้อมูลที่คำตอบมาจากตารางเดียวโดยไม่มีการรวมกันหลายตารางหรือตารางเดียวกันเพื่อตอบคำถาม (Temporal Projection Querying) และการสอบถามข้อมูลเชิงเวลาที่คำตอบที่มาจากการรวมกันหลายๆตาราง (Temporal joins Querying) โดยระบบจัดการฐานข้อมูล Oracle Database 12c สามารถสนับสนุนการทำงานของตัวดำเนินการกับช่วงเวลาเพียงตัวดำเนินการเดียวคือ OVERLAPS และจากการค้นคว้ายังไม่พบตัวดำเนินการอื่นๆนอกจาก OVERLAPS ที่สามารถทำงานตอบคำถามได้สอดคล้องตามตัวดำเนินการของมาตรฐาน SQL:2011

3.2.5.1 Temporal Projection and Selection Querying

สำหรับการสอบถามที่เป็นแบบ Temporal projection นั้นระบบฐานข้อมูล Oracle Database 12c สามารถตอบคำถามข้อมูลในอดีตและปัจจุบันได้ผลลัพธ์ที่สอดคล้องกับคำสั่งตามมาตรฐานเอสคิวแอล 2011

โปรแกรม 3.22 ตัวอย่างการสอบถามข้อมูลเชิงเวลาตามวันที่กำหนดใน Oracle Database 12c

```
SELECT * FROM LOT_LOC AS OF PERIOD FOR VALID_TIME DATE
'1998-02-08';
```

โปรแกรม 3.23 ตัวอย่างการสอบถามข้อมูลเชิงเวลาตามวันที่กำหนดตามมาตรฐานเอสคิวแอล 2011

```
SELECT * FROM LOT_LOC WHERE VALID_TIME CONTAINS DATE
'1998-02-08';
```

จากโปแกรมที่ 3.22 นั้นระบบจัดการฐานข้อมูล Oracle Database 12c สามารถทำงานได้สอดคล้องกับคำสั่งในโปรแกรมที่ 3.23 ที่เป็นรูปแบบคำสั่งตามมาตรฐานเอสคิวแอล 2011 ซึ่งระบบจัดการฐานข้อมูล Oracle Database 12c นั้นยังไม่รองการทำงานของตัวดำเนินการ CONTAINS ตามมาตรฐานเอสคิวแอล 2011

โปรแกรม 3.24 ตัวอย่างการสอบถามข้อมูลเชิงเวลาตามช่วงเวลาที่กำหนดใน Oracle Database 12c

```
SELECT * FROM LOT_LOC VERSIONS PERIOD FOR VALID_TIME
BETWEEN DATE '1998-03-01' AND DATE '1998-03-05';
```

โปรแกรม 3.25 ตัวอย่างการสอบถามข้อมูลเชิงเวลาตามช่วงเวลาที่กำหนดตามมาตรฐานเอสคิวแอล 2011

```
SELECT * FROM LOT_LOC WHERE VALID_TIME OVERLAPS PERIOD
(DATE '1998-03-01',DATE '1998-03-05');
```

จากโปรแกรมที่ 3.24 นั้นเป็นรูปแบบคำสั่งที่ระบบจัดการฐานข้อมูล Oracle Database 12c สามารถทำงานได้สอดคล้องตามคำสั่งในโปรแกรมที่ 3.25 ที่เป็นตัวอย่างคำสั่งตามมาตรฐานเอสคิวแอล 2011

โปรแกรม 3.26 ตัวอย่างการสอบถามข้อมูลเชิงเวลาใน Oracle Database 12c โดยใช้ OVERLAPS

```
SELECT * FROM LOT_LOC WHERE (FROM_DATE, TO_DATE) OVERLAPS
(DATE '1998-03-01', DATE '1998-03-05');
```

โปรแกรมที่ 3.26 ได้แสดงให้เห็นถึงการใช้ตัวดำเนินการ OVERLAPS ในระบบจัดการฐานข้อมูล Oracle Database 12c ซึ่งมีการทำงานที่สอดคล้องกับโปรแกรมที่ 3.24 และ 3.25

3.2.5.2 Temporal joins Querying

ในบางครั้งคำถามที่ต้องการสอบถามอาจจะต้องการการรวมตาราง (join) เพื่อตอบคำถามเหล่านั้นโดยคำถามส่วนใหญ่ที่มีการรวมตารางเพื่อตอบนั้นจำเป็นจะต้องมีการตรวจสอบการซ้อนทับของเวลาระหว่างการรวม ซึ่งระบบฐานข้อมูล Oracle Database 12c นั้นรองรับการตรวจสอบการซ้อนทับกันของช่วงเวลาผ่านตัวดำเนินการ OVERLAPS ดังโปรแกรมที่ 3.27

โปรแกรม 3.27 ตัวอย่างการสอบถามข้อมูลเชิงเวลาแบบ Temporal join ใน Oracle Database 12c

```
SELECT L1.LOT_ID_NUM "LOT1", L2.LOT_ID_NUM
"LOT2",L1.PEN_ID,L1.FROM_DATE,L1.TO_DATE
FROM LOT_LOC L1,LOT_LOC L2
WHERE L1.LOT_ID_NUM < L2.LOT_ID_NUM AND L1.FDYD = L2.FDYD
AND
L1.PEN_ID = L2.PEN_ID AND (L1.FROM_DATE, L1.TO_DATE)
OVERLAPS (L2.FROM_DATE, L2.TO_DATE);
```

3.3 ทดสอบการสร้างและวิเคราะห์คลังข้อมูลโดยใช้ระบบจัดการฐานข้อมูล Oracle Database 12c[6]

3.3.1 การทดสอบสร้างฐานข้อมูลต้นทาง

เริ่มต้นเราได้ทำการจำลองสร้างฐานข้อมูลต้นทางที่เป็นฐานข้อมูลสำหรับเก็บรายงานขายทั้งหมด (Operational Databaser) ซึ่งในฐานข้อมูลจะมีข้อมูลเกี่ยวกับข้อมูลการจำนวนขายหนังสือที่ขายได้ โดยจะมีข้อมูลทั้งเมืองของร้านค้าปลีกที่ขายได้ ข้อมูลของหนังสือเล่มนั้นๆที่ขายได้ รวมถึงวันที่ขายได้ การสร้างตัวอย่างฐานข้อมูลต้นทางจะใช้คำสั่งดังนี้

[6] Paul Lane, Padmaja Potineni. November 2014. Oracle Database, Data Warehousing Guide 12c Release 1 (12.1)

โปรแกรม 3.28 คำสั่งสร้างฐานข้อมูลต้นทางที่มีข้อมูลจำนวนหนังสือที่ขายได้ในร้านค้าปลีก

```
CREATE TABLE BOOK_DB(
      BOOK_ID INTEGER,
      BOOK_NAME VARCHAR(30),
      BOOK_GENRE VARCHAR(30),
      SELL_CITY VARCHAR(30),
      SELL_STATE VARCHAR(30),
      SELL_DATE DATE,
      SELL INTEGER
);
INSERT INTO BOOK_DB VALUES (9436,'Winnie the
Pooh','Childrens books','Boston','Massachusetts',DATE
'2016-03-02',20);
INSERT INTO BOOK_DB VALUES (7493,'Tropical
Food','Cooking','Boston','Massachusetts',DATE '2016-03-
02',5);
INSERT INTO BOOK_DB VALUES (7493,'Tropical
Food','Cooking','Arlington','Virginia',DATE '2016-03-
03',2);
INSERT INTO BOOK_DB VALUES (9436,'Winnie the
Pooh','Childrens books','Arlington','Virginia',DATE
'2016-03-03',11);
INSERT INTO BOOK_DB VALUES (9436,'Winnie the
Pooh','Childrens books','Arlington','Virginia',DATE
'2016-03-02',18);
INSERT INTO BOOK_DB VALUES (7493,'Tropical
Food','Cooking','Boston','Massachusetts',DATE '2016-03-
03',2);
```

ผลลัพธ์ที่ได้

BOOK_ID	BOOK_NAME	BOOK_GENRE	SELL_CITY	SELL_STATE	SELL_DATE	SELL
7493	Tropical Food	Cooking	Arlington	Virginia	03/03/2016	2
7493	Tropical Food	Cooking	Boston	Massachusetts	03/03/2016	2
7493	Tropical Food	Cooking	Boston	Massachusetts	03/02/2016	5
9436	Winnie the Pooh	Childrens books	Arlington	Virginia	03/02/2016	18
9436	Winnie the Pooh	Childrens books	Arlington	Virginia	03/03/2016	11
9436	Winnie the Pooh	Childrens books	Boston	Massachusetts	03/02/2016	20

รูป 3.17 แสดงผลลัพธ์เมื่อใช้คำสั่งสร้างฐานข้อมูลการขายหนังสือของร้านค้าปลีก

ต่อไปจะเป็นการสร้างฐานข้อมูลการขายหนังสือของร้านค้าบนอินเทอร์เน็ต ซึ่งจะมีข้อมูลสถานที่การขายที่ระบุเป็น IP Address และชื่อเว็ปไซต์ โดยจะใช้คำสั่งดังต่อไปนี้ในการสร้าง

โปรแกรม 3.29 คำสั่งสร้างฐานข้อมูลต้นทางที่มีข้อมูลจำนวนหนังสือขายได้ในร้านค้าอินเทอร์เน็ต

```
CREATE TABLE BOOK_INTERNET_DB(
    BOOK_ID INTEGER,
    BOOK_NAME VARCHAR(30),
    BOOK_GENRE VARCHAR(30),
    SELL_WEBSITE VARCHAR(30),
    SELL_IP_ADD VARCHAR(30),
    SELL_DATE DATE,
    SELL INTEGER
);
INSERT INTO BOOK_INTERNET_DB VALUES (9436,'Winnie the
Pooh','Childrens books','Amazon.com','100.1.1.10',DATE
'2016-03-02',6);
INSERT INTO BOOK_INTERNET_DB VALUES (7493,'Tropical
Food','Cooking','Textbooks.com','102.2.2.20',DATE
'2016-03-03',9);
INSERT INTO BOOK_INTERNET_DB VALUES (9436,'Winnie the
Pooh','Childrens
books','Textbooks.com','102.2.2.20',DATE '2016-03-
03',2);
INSERT INTO BOOK_INTERNET_DB VALUES (9436,'Winnie the
Pooh','Childrens
books','Textbooks.com','102.2.2.20',DATE '2016-03-
02',4);
INSERT INTO BOOK_INTERNET_DB VALUES (7493,'Tropical
Food','Cooking','Amazon.com','100.1.1.10',DATE '2016-
03-03',3);
```

ผลลัพธ์ที่ได้

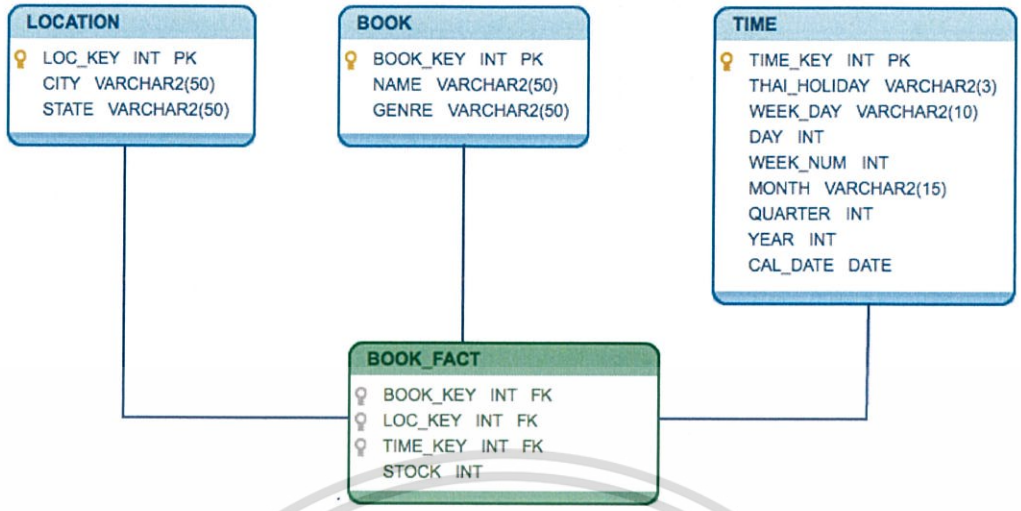
BOOK_ID	BOOK_NAME	BOOK_GENRE	SELL_WEBSITE	SELL_IP_ADD	SELL_DATE	SELL
9436	Winnie the Pooh	Childrens books	Amazon.com	100.1.1.10	03/02/2016	6
7493	Tropical Food	Cooking	Textbooks.com	102.2.2.20	03/03/2016	9
9436	Winnie the Pooh	Childrens books	Textbooks.com	102.2.2.20	03/03/2016	2
9436	Winnie the Pooh	Childrens books	Textbooks.com	102.2.2.20	03/02/2016	4
7493	Tropical Food	Cooking	Amazon.com	100.1.1.10	03/03/2016	3

รูป 3.18 แสดงผลลัพธ์เมื่อใช้คำสั่งสร้างฐานข้อมูลการขายหนังสือของร้านค้าบนอินเทอร์เน็ต

3.3.2 การทดสอบสร้างคลังข้อมูล

3.3.2.1 การทดสอบสร้างคลังข้อมูลแบบ Star Schema

ในขั้นตอนแรกเราต้องระบุวัตถุประสงค์ของข้อมูลที่ต้องการวิเคราะห์ก่อน โดยในที่นี้ต้องการวิเคราะห์ข้อมูลการขายหนังสือ ในขั้นต่อไปเราจะต้องระบุมุมมองข้อมูลที่ต้องการวิเคราะห์และรายละเอียดของแต่ละมุมมอง โดยสามารถสรุปเป็นแผนภาพได้ดังนี้



รูป 3.19 แสดงแผนภาพการออกแบบคลังข้อมูลแบบ Star Schema

หลังจากที่เราได้ออกแบบคลังข้อมูลแล้วเราก็จะเริ่มทำการสร้างคลังข้อมูลจากแหล่งข้อมูลที่เราได้สร้างไว้ก่อนหน้านี้ โดยสมมติให้เรามีแหล่งข้อมูลเพียงแหล่งเดียว ซึ่งเราจะเริ่มทำการสร้างจากมุมมอง (Dimensions) แต่ละมุมมองตามแผนภาพ ตั้งแต่การสร้าง Book dimension, Location dimension และ Time dimension และหลังจากนั้นเราจะทำการสร้าง Fact table (BOOK_FACT) โดยคำสั่งที่ใช้จะมีดังนี้

โปรแกรม 3.30 คำสั่งสร้าง Book dimension แบบ Star Schema

```
CREATE TABLE BOOK_DIM(
      BOOK_ID INTEGER,
      NAME VARCHAR(30),
      GENRE VARCHAR(30),
      PRIMARY KEY(BOOK_ID)
);
INSERT INTO BOOK_DIM
      SELECT MAX(ROWNUM),BOOK_NAME,BOOK_GENRE
      FROM BOOK_DB
      GROUP BY BOOK_NAME,BOOK_GENRE;
```

ผลลัพธ์ที่ได้

BOOK_ID	NAME	GENRE
5	Winnie the Pooh	Childrens books
6	Tropical Food	Cooking

รูป 3.20 แสดงผลลัพธ์การสร้าง Book dimension แบบ Star Schema

โปรแกรม 3.31 คำสั่งสร้าง Location dimension ของร้านค้าปลีกแบบ Star Schema

```
CREATE TABLE BOOK_LOCATION_DIM(
      LOC_ID INTEGER,
      CITY VARCHAR(30),
      STATE VARCHAR(30),
      PRIMARY KEY(LOC_ID)
);
INSERT INTO BOOK_LOCATION_DIM
      SELECT MAX(ROWNUM),SELL_CITY,SELL_STATE
      FROM BOOK_DB
      GROUP BY SELL_CITY,SELL_STATE ;
```

ผลลัพธ์ที่ได้

	LOC_ID	CITY	STATE
5		Arlington	Virginia
6		Boston	Massachusetts

รูป 3.21 แสดงผลลัพธ์การสร้าง Location dimension (Shop sell) แบบ Star Schema

โปรแกรม 3.32 ตัวอย่างคำสั่งสร้าง Time dimension แบบ Star Schema

```
CREATE TABLE TIME_DIM(
      TIME_ID INTEGER,
      THAI_HOLIDAY VARCHAR(3),
      WEEK_DAY VARCHAR(10),
      DAY INTEGER,
      WEEK_NUM INTEGER,
      MONTH VARCHAR(15),
      QUARTER INTEGER,
      YEAR INTEGER,
      CAL_DATE DATE,
      PRIMARY KEY(TIME_ID)
);
INSERT INTO TIME_DIM VALUES
(1,'NO','Wednesday',2,10,'March',1,2016,DATE '2016-3-
2');
INSERT INTO TIME_DIM VALUES
(2,'NO','Thursday',3,10,'March',1,2016,DATE '2016-3-
3');
INSERT INTO TIME_DIM VALUES
(3,'NO','Friday',4,10,'March',1,2016,DATE '2016-3-4');
INSERT INTO TIME_DIM VALUES
(4,'NO','Saturday',5,10,'March',1,2016,DATE '2016-3-
5');
INSERT INTO TIME_DIM VALUES
(5,'NO','Sunday',6,11,'March',1,2016,DATE '2016-3-6');
INSERT INTO TIME_DIM VALUES
(6,'NO','Monday',7,11,'March',1,2016,DATE '2016-3-7');
...
```

ผลลัพธ์ที่ได้

	TIME_ID	THAI_HOLIDAY	WEEK_DAY	DAY	WEEK_NUM	MONTH	QUARTER	YEAR	CAL_DATE
1		NO	Wednesday	2	10	March	1	2016	03/02/2016
2		NO	Thursday	3	10	March	1	2016	03/03/2016
3		NO	Friday	4	10	March	1	2016	03/04/2016
4		NO	Saturday	5	10	March	1	2016	03/05/2016
5		NO	Sunday	6	11	March	1	2016	03/06/2016
6		NO	Monday	7	11	March	1	2016	03/07/2016
7		NO	Tuesday	8	11	March	1	2016	03/08/2016
8		NO	Wednesday	9	11	March	1	2016	03/09/2016
9		NO	Thursday	10	11	March	1	2016	03/10/2016
10		NO	Friday	11	11	March	1	2016	03/11/2016

รูป 3.22 แสดงผลลัพธ์การสร้าง Time dimension แบบ Star Schema

โปรแกรม 3.33 คำสั่งสร้าง Fact table ของร้านค้าปลีก (Star Schema)

```
CREATE TABLE BOOK_FACT(
     BOOK_ID INTEGER,
     LOC_ID INTEGER,
     TIME_ID INTEGER,
     SELL INTEGER,
     FOREIGN KEY (BOOK_ID) REFERENCES
BOOK_DIM(BOOK_ID),
     FOREIGN KEY (LOC_ID) REFERENCES
BOOK_LOCATION_DIM(LOC_ID),
     FOREIGN KEY (TIME_ID) REFERENCES TIME_DIM(TIME_ID)
);
INSERT INTO BOOK_FACT
     SELECT
BOOK_DIM.BOOK_ID,BOOK_LOCATION_DIM.LOC_ID,TIME_DIM.TIME
_ID,SELL
     FROM BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM,BOOK_DB
     WHERE BOOK_DB.BOOK_NAME = BOOK_DIM.NAME AND
               BOOK_DB.BOOK_GENRE = BOOK_DIM.GENRE AND
     BOOK_DB.SELL_CITY = BOOK_LOCATION_DIM.CITY AND
     BOOK_DB.SELL_STATE = BOOK_LOCATION_DIM.STATE AND
     BOOK_DB.SELL_DATE = TIME_DIM.CAL_DATE;
```

ผลลัพธ์ที่ได้

BOOK_ID	LOC_ID	TIME_ID	SELL
5	6	1	20
6	6	1	5
5	5	1	18
6	6	2	2
6	5	2	2
5	5	2	11

รูป 3.23 แสดงผลลัพธ์การสร้าง Fact table ของร้านค้าปลีก (Star Schema)

หลังจากที่เราสร้างคลังข้อมูลของร้านค้าปลีกเสร็จแล้ว เราจะมาสร้างคลังข้อมูลสำหรับร้านค้าบนอินเทอร์เน็ต เนื่องจากข้อมูลของหนังสือและเวลานั้นเหมือนกันจึงใช้ Book dimension และ Time dimension ร่วมกันได้ ดังนั้นสิ่งที่เราจะต้องสร้างเพิ่มจึงมีแค่ Location Dimension และ Fact table สำหรับร้านค้าบนอินเทอร์เน็ต โดยคำสั่งที่ใช้จะมีดังนี้

โปรแกรม 3.34 คำสั่งสร้าง Location dimension ของร้านค้าบนอินเทอร์เน็ต (Star Schema)

```
CREATE TABLE BOOK_WEB_LOCATION_DIM(
    WEB_LOC_ID INTEGER,
    WEBSITE VARCHAR(30),
    IP_ADDRESS VARCHAR(30),
    PRIMARY KEY(WEB_LOC_ID)
);
INSERT INTO BOOK_WEB_LOCATION_DIM SELECT
MAX(ROWNUM),SELL_WEBSITE,SELL_IP_ADD FROM
BOOK_INTERNET_DB GROUP BY SELL_WEBSITE,SELL_IP_ADD;
```

ผลลัพธ์ที่ได้

WEB_LOC_ID	WEBSITE	IP_ADDRESS
4	Textbooks.com	102.2.2.20
5	Amazon.com	100.1.1.10

รูป 3.24 แสดงผลลัพธ์การสร้าง Location dimension (Internet Sell) แบบ Star Schema

โปรแกรม 3.35 คำสั่งสร้าง Fact table ของร้านค้าบนอินเทอร์เน็ต (Star Schema)

```
CREATE TABLE BOOK_WEB_LOCATION_DIM(
    WEB_LOC_ID INTEGER,
    WEBSITE VARCHAR(30),
    IP_ADDRESS VARCHAR(30),
    PRIMARY KEY(WEB_LOC_ID)
);
INSERT INTO BOOK_WEB_LOCATION_DIM SELECT
MAX(ROWNUM),SELL_WEBSITE,SELL_IP_ADD FROM
BOOK_INTERNET_DB GROUP BY SELL_WEBSITE,SELL_IP_ADD;
```

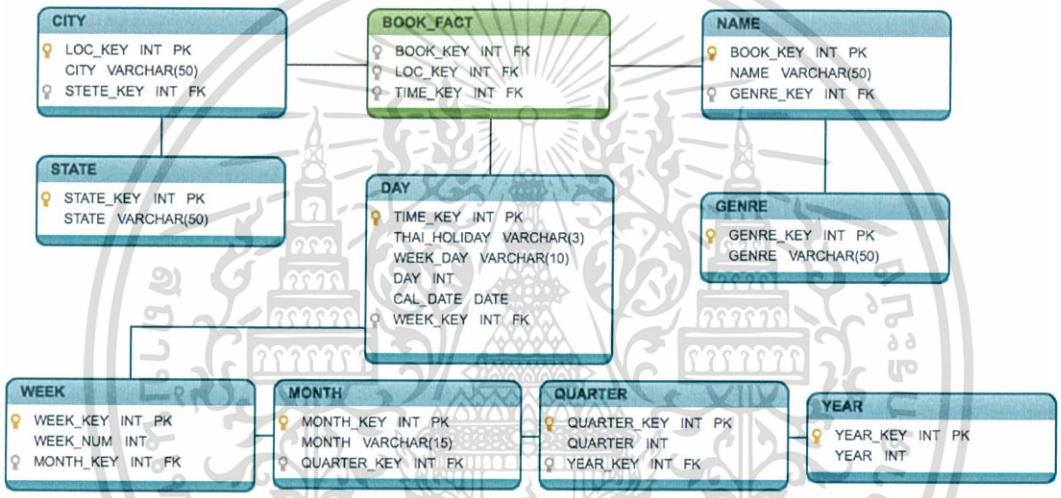
ผลลัพธ์ที่ได้

BOOK_ID	WEB_LOC_ID	TIME_ID	SELL
5	5	1	6
5	4	1	4
6	5	2	3
6	4	2	9
5	4	2	2

รูป 3.25 แสดงผลลัพธ์การสร้าง Fact table ของร้านค้าบนอินเทอร์เน็ต (Star Schema)

3.3.2.2 การทดสอบสร้างคลังข้อมูลแบบ Snowflake Schema

ในขั้นตอนแรกเราต้องระบุวัตถุประสงค์ของข้อมูลที่ต้องการวิเคราะห์ก่อน โดยในที่นี้ต้องการวิเคราะห์ข้อมูลการขายหนังสือ ในขั้นต่อไปเราจะต้องระบุมุมมองข้อมูลที่ต้องการวิเคราะห์และรายละเอียดของแต่ละมุมมอง แต่เนื่องจากการออกแบบคลังข้อมูลแบบ Snowflake Schema นั้นจะแตกต่างกับแบบ Star Schema ตรงที่ว่าในแต่ละ Dimension นั้นสามารถมีได้มากกว่า 1 dimension table และแต่ละ dimension tables นั้นจะมี foreign key ของ dimension table อีกตารางหนึ่งยกเว้น dimension table ที่เป็นระดับที่สูงสุด (ไม่นับ Top Level) โดยสามารถสรุปเป็นแผนภาพได้ดังนี้



รูป 3.26 แสดงแผนภาพการออกแบบคลังข้อมูลแบบ Snowflake Schema

หลังจากที่เราได้ออกแบบคลังข้อมูลแล้วเราก็จะเริ่มทำการสร้างคลังข้อมูลจากแหล่งข้อมูลที่เราได้สร้างไว้ก่อนหน้านี้ โดยสมมติให้เรามีแหล่งข้อมูลเพียงแหล่งเดียว ซึ่งเราจะเริ่มทำการสร้างจากมุมมอง (Dimensions) แต่ละมุมมองตามแผนภาพ ตั้งแต่การสร้าง Book dimension, Location dimension และ Time dimension และหลังจากนั้นเราจะทำการสร้าง Fact table (BOOK_FACT) โดยคำสั่งที่ใช้จะมีดังนี้

โปรแกรม 3.36 คำสั่งสร้าง Book dimension แบบ Snowflake Schema

```
CREATE TABLE B_GENRE(
      GENRE_KEY INTEGER,
      GENRE VARCHAR(50) NOT NULL,
      PRIMARY KEY(GENRE_KEY)
);
INSERT INTO B_GENRE
      SELECT MAX(ROWNUM),BOOK_GENRE
      FROM BOOK_DB
      GROUP BY BOOK_GENRE;
CREATE TABLE B_NAME(
      BOOK_KEY INTEGER,
      NAME VARCHAR(50) NOT NULL,
      GENRE_KEY INTEGER,
      PRIMARY KEY(BOOK_KEY),
      FOREIGN KEY(GENRE_KEY) REFERENCES
B_GENRE(GENRE_KEY)
);
INSERT INTO B_NAME
      SELECT MAX(ROWNUM),BOOK_NAME,GENRE_KEY
      FROM BOOK_DB,B_GENRE
      WHERE BOOK_GENRE = GENRE
      GROUP BY BOOK_NAME,GENRE_KEY;
```

ผลลัพธ์ที่ได้

	GENRE_KEY	GENRE
5		Childrens books
6		Cooking

รูป 3.27 แสดงผลลัพธ์การสร้าง Book dimension (Genre Level) แบบ Snowflake Schema

	BOOK_KEY	NAME	GENRE_KEY
6		Tropical Food	6
5		Winnie the Pooh	5

รูป 3.28 แสดงผลลัพธ์การสร้าง Book dimension (Name Level) แบบ Snowflake Schema

โปรแกรม 3.37 คำสั่งสร้าง Location dimension (Shop sell) แบบ Snowflake Schema

```
CREATE TABLE L_STATE(
      STATE_KEY INTEGER,
      STATE VARCHAR(50) NOT NULL,
      PRIMARY KEY(STATE_KEY)
);
INSERT INTO L_STATE
      SELECT MAX(ROWNUM),SELL_STATE
      FROM BOOK_DB
      GROUP BY SELL_STATE;
CREATE TABLE L_CITY(
      LOC_KEY INTEGER,
      CITY VARCHAR(50) NOT NULL,
      STATE_KEY INTEGER,
      PRIMARY KEY(LOC_KEY),
      FOREIGN KEY(STATE_KEY) REFERENCES
L_STATE(STATE_KEY)
);
INSERT INTO L_CITY
      SELECT MAX(ROWNUM),SELL_CITY,STATE_KEY
      FROM BOOK_DB,L_STATE
      WHERE SELL_STATE = STATE
      GROUP BY SELL_CITY,STATE_KEY;
```

ผลลัพธ์ที่ได้

STATE_KEY	STATE
6	Massachusetts
5	Virginia

รูป 3.29 แสดงผลลัพธ์การสร้าง Location dimension (State Level) ของร้านค้าปลีกแบบ Snowflake Schema

LOC_KEY	CITY	STATE_KEY
6	Boston	6
5	Arlington	5

รูป 3.30 แสดงผลลัพธ์การสร้าง Location dimension (City Level) ของร้านค้าปลีกแบบ Snowflake Schema

โปรแกรม 3.38 คำสั่งสร้าง Time dimension แบบ Snowflake Schema

```
CREATE TABLE T_YEAR(
      YEAR_KEY INTEGER,
      YEAR INTEGER NOT NULL,
      PRIMARY KEY(YEAR_KEY)
);
INSERT INTO T_YEAR
      SELECT MAX(ROWNUM),YEAR
      FROM TIME_DIM
      GROUP BY YEAR;
CREATE TABLE T_QUARTER(
      QUARTER_KEY INTEGER,
      QUARTER INTEGER NOT NULL,
      YEAR_KEY INTEGER,
      PRIMARY KEY(QUARTER_KEY),
      FOREIGN KEY(YEAR_KEY) REFERENCES T_YEAR(YEAR_KEY)
);
INSERT INTO T_QUARTER
      SELECT MAX(ROWNUM),QUARTER,YEAR_KEY
      FROM TIME_DIM,T_YEAR
      WHERE TIME_DIM.YEAR = T_YEAR.YEAR
      GROUP BY QUARTER,YEAR_KEY;
CREATE TABLE T_MONTH(
      MONTH_KEY INTEGER,
      MONTH VARCHAR(15) NOT NULL,
      QUARTER_KEY INTEGER,
      PRIMARY KEY(MONTH_KEY),
      FOREIGN KEY(QUARTER_KEY) REFERENCES
T_QUARTER(QUARTER_KEY)
);
INSERT INTO T_MONTH
      SELECT MAX(ROWNUM),MONTH,QUARTER_KEY
      FROM TIME_DIM,T_QUARTER,T_YEAR
      WHERE      TIME_DIM.QUARTER = T_QUARTER.QUARTER AND
                 TIME_DIM.YEAR = T_YEAR.YEAR AND
                 T_QUARTER.YEAR_KEY = T_YEAR.YEAR_KEY
      GROUP BY MONTH,QUARTER_KEY;
CREATE TABLE T_WEEK(
      WEEK_KEY INTEGER,
      WEEK_NUM INTEGER NOT NULL,
      MONTH_KEY INTEGER,
      PRIMARY KEY(WEEK_KEY),
      FOREIGN KEY(MONTH_KEY) REFERENCES
T_MONTH(MONTH_KEY)
);
```

```
INSERT INTO T_WEEK
     SELECT MAX(ROWNUM),WEEK_NUM,MONTH_KEY
     FROM TIME_DIM,T_MONTH,T_QUARTER,T_YEAR
     WHERE     TIME_DIM.MONTH = T_MONTH.MONTH AND
               TIME_DIM.QUARTER = T_QUARTER.QUARTER AND
               TIME_DIM.YEAR = T_YEAR.YEAR AND
               T_MONTH.QUARTER_KEY =
T_QUARTER.QUARTER_KEY AND
               T_QUARTER.YEAR_KEY = T_YEAR.YEAR_KEY
     GROUP BY WEEK_NUM,MONTH_KEY;
CREATE TABLE T_DAY(
     TIME_KEY INTEGER,
     THAI_HOLIDAY VARCHAR(3) NOT NULL,
     WEEK_DAY VARCHAR(10) NOT NULL,
     DAY INTEGER NOT NULL,
     CAL_DATE DATE NOT NULL,
     WEEK_KEY INTEGER,
     PRIMARY KEY(TIME_KEY),
     FOREIGN KEY(WEEK_KEY) REFERENCES T_WEEK(WEEK_KEY)
);
INSERT INTO T_DAY
     SELECT
MAX(ROWNUM),THAI_HOLIDAY,WEEK_DAY,DAY,CAL_DATE,WEEK_KEY
     FROM TIME_DIM,T_WEEK,T_MONTH,T_QUARTER,T_YEAR
     WHERE     TIME_DIM.WEEK_NUM = T_WEEK.WEEK_NUM AND
               TIME_DIM.MONTH = T_MONTH.MONTH AND
               TIME_DIM.QUARTER = T_QUARTER.QUARTER AND
               TIME_DIM.YEAR = T_YEAR.YEAR AND
               T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
               T_MONTH.QUARTER_KEY =
T_QUARTER.QUARTER_KEY AND
               T_QUARTER.YEAR_KEY = T_YEAR.YEAR_KEY
     GROUP BY
CAL_DATE,DAY,WEEK_DAY,THAI_HOLIDAY,WEEK_KEY;
```

ผลลัพธ์ที่ได้

YEAR_KEY	YEAR
105	2016

รูป 3.31 แสดงผลลัพธ์การสร้าง Time dimension (Year Level) แบบ Snowflake Schema

QUARTER_KEY	QUARTER	YEAR_KEY
30	1	105
105	2	105

รูป 3.32 แสดงผลลัพธ์การสร้าง Time dimension (Quarter Level) แบบ Snowflake Schema

MONTH_KEY	MONTH	QUARTER_KEY
105	June	105
30	March	30
60	April	105
91	May	105

รูป 3.33 แสดงผลลัพธ์การสร้าง Time dimension (Month Level) แบบ Snowflake Schema

WEEK_KEY	WEEK_NUM	MONTH_KEY
30	14	30
25	13	30
74	20	91
102	24	105
91	23	91
32	14	60
60	18	60
81	21	91
105	25	105
46	16	60
67	19	91
95	23	105

รูป 3.34 แสดงผลลัพธ์การสร้าง Time dimension (Week Level) แบบ Snowflake Schema

TIME_KEY	THAI_HOLIDAY	WEEK_DAY	DAY	CAL_DATE	WEEK_KEY
11	NO	Saturday	12	03/12/2016	11
23	NO	Thursday	24	03/24/2016	25
36	YES	Wednesday	6	04/06/2016	39
61	YES	Sunday	1	05/01/2016	67
68	NO	Sunday	8	05/08/2016	74
77	NO	Tuesday	17	05/17/2016	81
79	NO	Thursday	19	05/19/2016	81
80	YES	Friday	20	05/20/2016	81
102	NO	Saturday	11	06/11/2016	102
105	NO	Tuesday	14	06/14/2016	105
2	NO	Thursday	3	03/03/2016	4
3	NO	Friday	4	03/04/2016	4

รูป 3.35 แสดงผลลัพธ์การสร้าง Time dimension (Day Level) แบบ Snowflake Schema

โปรแกรม 3.39 คำสั่งสร้าง Fact table ของร้านค้าปลีก (Snowflake Schema)

```
CREATE TABLE BOOK_FACT_SF(
     BOOK_KEY INTEGER,
     LOC_KEY INTEGER,
     TIME_KEY INTEGER,
     SELL INTEGER NOT NULL,
     FOREIGN KEY (BOOK_KEY) REFERENCES
B_NAME(BOOK_KEY),
     FOREIGN KEY (LOC_KEY) REFERENCES L_CITY(LOC_KEY),

     FOREIGN KEY (TIME_KEY) REFERENCES T_DAY(TIME_KEY)
);
INSERT INTO BOOK_FACT_SF
SELECT
B_NAME.BOOK_KEY,L_CITY.LOC_KEY,T_DAY.TIME_KEY,SELL
FROM B_NAME,B_GENRE,L_CITY,L_STATE,T_DAY,BOOK_DB
WHERE      BOOK_DB.BOOK_NAME = B_NAME.NAME AND
           BOOK_DB.BOOK_GENRE = B_GENRE.GENRE AND
           BOOK_DB.SELL_CITY = L_CITY.CITY AND
           BOOK_DB.SELL_STATE = L_STATE.STATE AND
           BOOK_DB.SELL_DATE = T_DAY.CAL_DATE AND
           B_NAME.GENRE_KEY = B_GENRE.GENRE_KEY AND
           L_CITY.STATE_KEY = L_STATE.STATE_KEY;
```

ผลลัพธ์ที่ได้

BOOK_KEY	LOC_KEY	TIME_KEY	SELL
5	5	2	11
6	5	2	2
6	6	2	2
5	5	1	18
6	6	1	5
5	6	1	20

รูป 3.36 แสดงผลลัพธ์การสร้าง Fact table ของร้านค้าปลีก (Snowflake Schema)

หลังจากที่เราสร้างคลังข้อมูลของร้านค้าปลีกเสร็จแล้ว เราจะมาสร้างคลังข้อมูลสำหรับร้านค้าบนอินเทอร์เน็ต เนื่องจากข้อมูลของหนังสือและเวลานั้นเหมือนกันจึงใช้ Book dimension และ Time dimension ร่วมกันได้ ดังนั้นสิ่งที่เราจะต้องสร้างเพิ่มจึงมีแค่ Location Dimension และ Fact table สำหรับร้านค้าบนอินเทอร์เน็ต โดยคำสั่งที่ใช้จะมีดังนี้

โปรแกรม 3.40 คำสั่งสร้าง Location dimension ของร้านค้าบนอินเทอร์เน็ต (Snowflake Schema)

```
CREATE TABLE L_WEBSITE(
   WEB_LOC_KEY INTEGER,
   WEBSITE VARCHAR(30),
   IP_ADDRESS VARCHAR(30),
   PRIMARY KEY(WEB_LOC_KEY)
);
INSERT INTO L_WEBSITE SELECT
MAX(ROWNUM),SELL_WEBSITE,SELL_IP_ADD FROM
BOOK_INTERNET_DB GROUP BY SELL_WEBSITE,SELL_IP_ADD;
```

ผลลัพธ์ที่ได้

	WEB_LOC_KEY	WEBSITE	IP_ADDRESS
4		Textbooks.com	102.2.2.20
5		Amazon.com	100.1.1.10

รูป 3.37 แสดงผลลัพธ์การสร้าง Location dimension (Internet sell) แบบ Snowflake Schema

โปรแกรม 3.41 คำสั่งสร้าง Fact table ของร้านค้าบนอินเทอร์เน็ต (Snowflake Schema)

```
CREATE TABLE BOOK_INT_FACT_SF(
   BOOK_KEY INTEGER,
   WEB_LOC_KEY INTEGER,
   TIME_KEY INTEGER,
   SELL INTEGER,
   FOREIGN KEY (BOOK_KEY) REFERENCES B_NAME(BOOK_KEY),
   FOREIGN KEY (WEB_LOC_KEY) REFERENCES
L_WEBSITE(WEB_LOC_KEY),
   FOREIGN KEY (TIME_KEY) REFERENCES T_DAY(TIME_KEY)
);
INSERT INTO BOOK_INT_FACT_SF
SELECT
B_NAME.BOOK_KEY,L_WEBSITE.WEB_LOC_KEY,T_DAY.TIME_KEY,SE
LL
FROM B_NAME,B_GENRE,L_WEBSITE,T_DAY,BOOK_INTERNET_DB
WHERE BOOK_INTERNET_DB.BOOK_NAME = B_NAME.NAME AND
      BOOK_INTERNET_DB.BOOK_GENRE = B_GENRE.GENRE AND
      BOOK_INTERNET_DB.SELL_WEBSITE = L_WEBSITE.WEBSITE
AND
BOOK_INTERNET_DB.SELL_IP_ADD = L_WEBSITE.IP_ADDRESS AND
BOOK_INTERNET_DB.SELL_DATE = T_DAY.CAL_DATE AND
      B_NAME.GENRE_KEY = B_GENRE.GENRE_KEY;
```

ผลลัพธ์ที่ได้

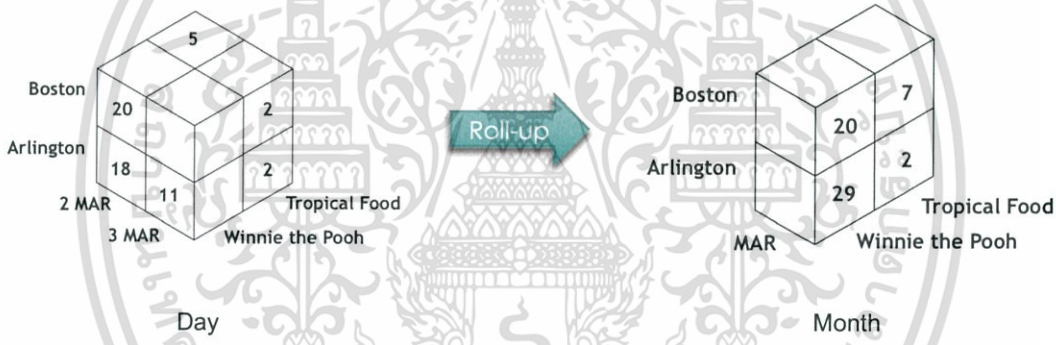
BOOK_KEY	WEB_LOC_KEY	TIME_KEY	SELL
6	5	2	3
6	4	2	9
5	4	2	2
5	5	1	6
5	4	1	4

รูป 3.38 แสดงผลลัพธ์การสร้าง Fact table ของร้านค้าบนอินเทอร์เน็ต (Snowflake Schema)

3.3.3 การวิเคราะห์และตอบคำถามบนคลังข้อมูล

3.3.3.1 การวิเคราะห์และตอบคำถามบนคลังข้อมูลแบบ Star Schema

1) Roll-up คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้สูงขึ้นเพื่อทำการสรุปข้อมูล โดยการตอบคำถามแบบ roll-up สามารถนำเสนอได้ดังนี้



รูป 3.39 แสดงการ Roll-up ใน Time dimension จากวันเป็นเดือน (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

MONTH	NAME	CITY	SUM(SELL)
March	Winnie the Pooh	Arlington	29
March	Tropical Food	Arlington	2
March	Winnie the Pooh	Boston	20
March	Tropical Food	Boston	7

Roll-up

DAY	MONTH	NAME	CITY	SELL
2	March	Winnie the Pooh	Arlington	18
2	March	Winnie the Pooh	Boston	20
2	March	Tropical Food	Boston	5
3	March	Tropical Food	Arlington	2
3	March	Tropical Food	Boston	2
3	March	Winnie the Pooh	Arlington	11

รูป 3.40 แสดงการ Roll-up ใน Time dimension จากวันเป็นเดือน (Relation)

เพื่อให้เราสามารถตอบคำถาม Roll-up ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.42 คำสั่ง SQL Roll-up (Star Schema)

```
SELECT     MIN(TIME_DIM.MONTH) AS "MONTH",
      BOOK_DIM.NAME,
      BOOK_LOCATION_DIM.CITY,
      SUM(SELL)
FROM       BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE      BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
      BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
      BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
      TIME_DIM.MONTH = 'March'
GROUP BY BOOK_DIM.NAME,BOOK_LOCATION_DIM.CITY
ORDER BY 1;
```

2) Drill-down คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้ต่ำลง เพื่อเป็นการดูรายละเอียดของข้อมูล โดยการตอบคำถามแบบ Drill-down สามารถนำเสนอได้ดังนี้

รูป 3.41 แสดงการ Drill-down ใน Time dimension จากเดือนเป็นวัน (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

MONTH	NAME	CITY	SUM(SELL)
March	Winnie the Pooh	Arlington	29
March	Tropical Food	Arlington	2
March	Winnie the Pooh	Boston	20
March	Tropical Food	Boston	7

DAY	MONTH	NAME	CITY	SELL
2	March	Winnie the Pooh	Arlington	18
2	March	Winnie the Pooh	Boston	20
2	March	Tropical Food	Boston	5
3	March	Tropical Food	Arlington	2
3	March	Tropical Food	Boston	2
3	March	Winnie the Pooh	Arlington	11

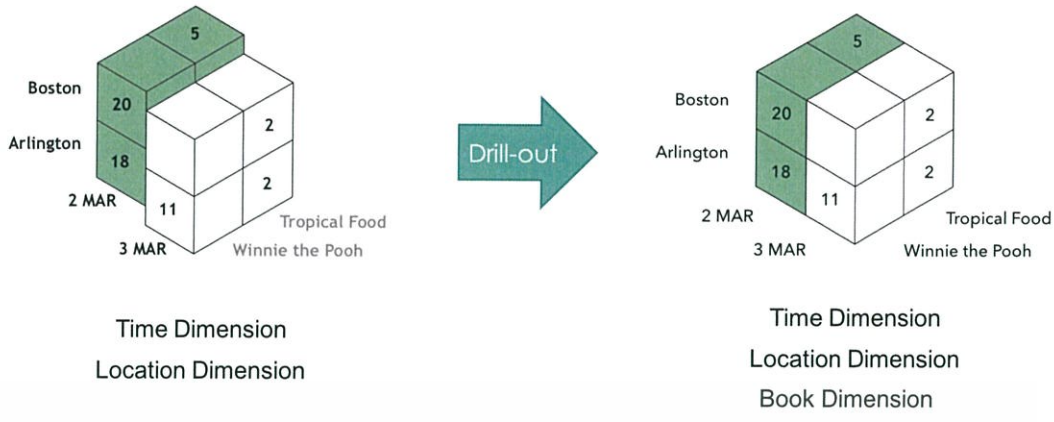
รูป 3.42 แสดงการ Drill-down ใน Time dimension จากเดือนเป็นวัน (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-down ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.43 คำสั่ง SQL Drill-down (Star Schema)

```
SELECT    TIME_DIM.DAY AS "DAY",
     TIME_DIM.MONTH AS "MONTH",
     BOOK_DIM.NAME,
     BOOK_LOCATION_DIM.CITY,
     SELL
FROM      BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE     BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
     BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
     BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID
ORDER BY 1;
```

3) Drill-out คือเพิ่มมุมองที่ต้องการวิเคราะห์เพื่อเป็นการดูรายละเอียดของข้อมูลในมุมมองอื่นประกอบ โดยการตอบคำถามแบบ Drill-out สามารถนำเสนอได้ดังนี้



รูป 3.43 แสดงการ Drill-out โดยการเพิ่ม Book dimension (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

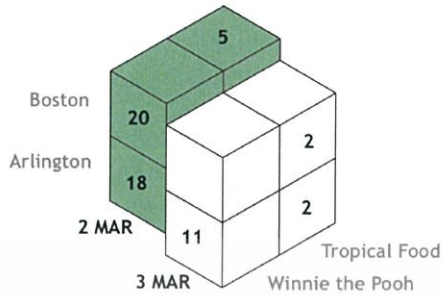
รูป 3.44 แสดงการ Drill-out โดยการเพิ่ม Book dimension (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-out ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.44 คำสั่ง SQL Drill-out (Star Schema)

```
SELECT    MIN(TIME_DIM.DAY) AS "DAY",
     MIN(TIME_DIM.MONTH) AS "MONTH",
     BOOK_LOCATION_DIM.CITY,
     BOOK_DIM.NAME,
     SUM(SELL)
FROM      BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE     BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
     BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
     BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
     TIME_DIM.CAL_DATE = DATE '2016-03-02'
GROUP BY BOOK_LOCATION_DIM.CITY,BOOK_DIM.NAME;
```

4) Slice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองเพียงมุมมองเดียว โดยการตอบคำถามแบบ Slice สามารถนำเสนอได้ดังนี้



Slice(2 MARCH) = SUM(2 MARCH)
SUM(2 MARCH) = 20+5+18+0 = 43

รูป 3.45 แสดงการ Slice บน Time dimension ของวันที่ 2 มีนาคม (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

DAY	MONTH	SELL
2	March	43

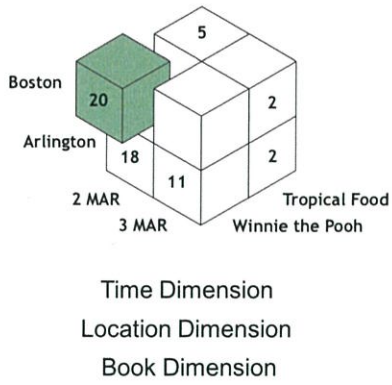
รูป 3.46 แสดงการ Slice บน Time dimension ของวันที่ 2 มีนาคม (Relation)

เพื่อให้เราสามารถตอบคำถาม Slice ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.45 คำสั่ง SQL Slice (Star Schema)

```
SELECT MIN(TIME_DIM.DAY) AS "DAY",MIN(TIME_DIM.MONTH)
AS "MONTH",SUM(SELL) AS "SELL"
FROM BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE     BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
     BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
     BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
     TIME_DIM.CAL_DATE = DATE '2016-03-2'
GROUP BY TIME_DIM.CAL_DATE;
```

5) Dice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองหลายมุมมอง โดยการตอบคำถามแบบ Slice สามารถนำเสนอได้ดังนี้



รูป 3.47 แสดงการ Dice โดยใช้ 3 dimensions (Cube)

จากรูป 3.47 นั้นเป็นการ Dice โดยใช้ 3 dimensions ประกอบไปด้วย Time dimension, Location dimension และ Book dimension เพื่อเป็นการสรุปดูยอดขายหนังสือ Winnie the Pooh ในวันที่ 2 มีนาคม 2016 ที่ถูกขายใน Boston โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

DAY	MONTH	CITY	NAME	SELL
2	March	Boston	Winnie the Pooh	20

รูป 3.48 แสดงการ Dice โดยใช้ 3 dimensions (Relation)

เพื่อให้เราสามารถตอบคำถาม Dice ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.46 คำสั่ง SQL Dice (Star Schema)

```
SELECT    MIN(TIME_DIM.DAY) AS
"DAY",MIN(TIME_DIM.MONTH) AS "MONTH",
     BOOK_LOCATION_DIM.CITY AS "CITY",
     BOOK_DIM.NAME AS "NAME",
     SUM(SELL) AS "SELL"
FROM BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE     BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
     BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
     BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
     TIME_DIM.CAL_DATE = DATE '2016-03-2' AND
     BOOK_LOCATION_DIM.CITY = 'Boston' AND
     BOOK_DIM.NAME = 'Winnie the Pooh'
GROUP BY
TIME_DIM.CAL_DATE,BOOK_LOCATION_DIM.CITY,BOOK_DIM.NAME;
```

6) Drill-Across เกิดจากการรวมกันของข้อมูลของ cube 2 อัน กล่าวคือ การวิเคราะห์ข้อมูลบางครั้งไม่สามารถใช้เพียง cube เดียวแล้ววิเคราะห์ได้ จึงต้องมีการใช้ cube อีกอันซึ่งมี conformed dimensions มาช่วยในการวิเคราะห์โดยการตอบคำถามแบบ Drill-Across สามารถนำเสนอได้ดังนี้

รูป 3.49 แสดงการ Drill-Across โดยใช้ Shop sell และ Internet sell (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

รูป 3.50 แสดงการ Drill-Across โดยใช้ Shop sell และ Internet sell (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-Across ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.47 คำสั่ง SQL Drill-Across (Star Schema)

```
SELECT MIN(DATE1) AS "DATE",SUM(SELL1) AS "SELL"
FROM (
SELECT TIME_DIM.CAL_DATE AS "DATE1",SUM(SELL) AS
"SELL1"
FROM
BOOK_INT_FACT,BOOK_DIM,BOOK_WEB_LOCATION_DIM,TIME_DIM
     WHERE BOOK_INT_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
BOOK_INT_FACT.WEB_LOC_ID =
BOOK_WEB_LOCATION_DIM.WEB_LOC_ID AND
BOOK_INT_FACT.TIME_ID = TIME_DIM.TIME_ID AND
          TIME_DIM.CAL_DATE = DATE '2016-03-2' GROUP BY
TIME_DIM.CAL_DATE
UNION
SELECT TIME_DIM.CAL_DATE AS "DATE",SUM(SELL) AS "SELL"
FROM BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
      BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
      BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
      TIME_DIM.CAL_DATE = DATE '2016-03-2'
GROUP BY TIME_DIM.CAL_DATE);
```

7) Pivot table คือตารางที่แสดงข้อมูลโดยใช้มุมมอง 2 มุมมอง โดยจะเป็นการแสดงค่าผลรวมย่อยและค่าผลรวมทั้งหมด ซึ่งอนุญาตให้ทำการ roll-up และ drill-down ได้และสามารถมีหลายๆมุมมองใน 1 แกนของตารางได้ ซึ่งการตอบคำถามในรูปแบบของ Relation นั้นไม่สามารถตอบได้โดยตรง โดยสามารถตอบคำถามให้ใกล้เคียงในรูปแบบของ Relation ได้ดังนี้

GENRE	CITY	SUBTOTAL
Cooking	Boston	7
Childrens books	Boston	20
Cooking	Arlington	2
Childrens books	Arlington	29

รูป 3.51 แสดงตัวอย่างการตอบคำถามที่ให้ข้อมูลใกล้เคียงกับ Pivot table (Relation)

เพื่อให้เราสามารถตอบคำถามตามรูป 3.51 ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.48 คำสั่ง SQL ที่ให้คำตอบใกล้เคียง Pivot table (Star Schema)

```
SELECT     BOOK_DIM.GENRE AS "GENRE",
      BOOK_LOCATION_DIM.CITY AS "CITY",
      SUM(SELL) AS "SUBTOTAL"
FROM       BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE      BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
      BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
      BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID
GROUP BY BOOK_DIM.GENRE,BOOK_LOCATION_DIM.CITY
```

8) Ranking คือการนำข้อมูลที่ต้องการวิเคราะห์มาจัดอันดับ โดยการตอบคำถามแบบ Ranking สามารถนำเสนอได้ดังนี้

รูป 3.52 แสดงการทำ Ranking บน Time dimension และ Location dimension (Cube)

โดยผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

RANK	DAY	MONTH	CITY	SUM(SELL)
1	2	March	Boston	25
2	2	March	Arlington	18

รูป 3.53 แสดงการทำ Ranking บน Time dimension และ Location dimension (Relation)

เพื่อให้เราสามารถตอบคำถาม Ranking ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

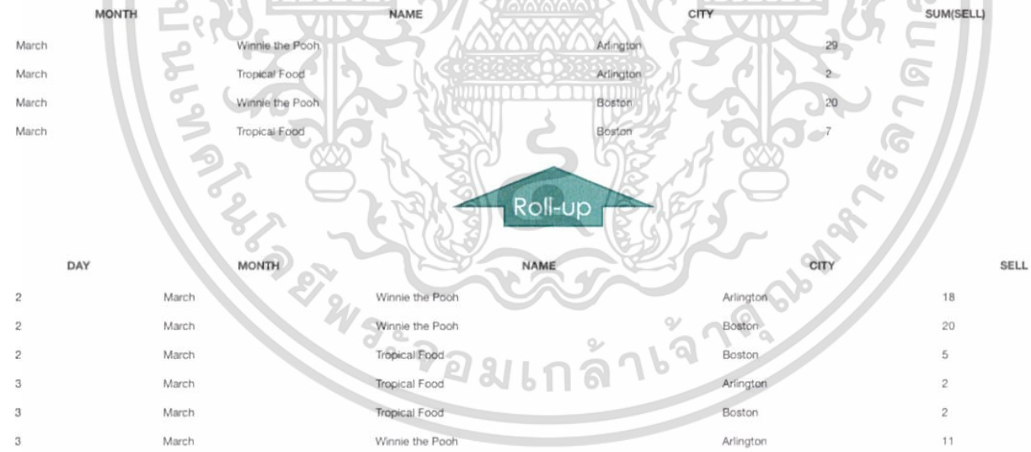
โปรแกรม 3.49 คำสั่ง SQL Ranking (Star Schema)

```
SELECT RANK() OVER (ORDER BY SUM(SELL) DESC) AS "RANK",
     MIN(TIME_DIM.DAY) AS "DAY",
     MIN(TIME_DIM.MONTH) AS "MONTH",
     BOOK_LOCATION_DIM.CITY,
     SUM(SELL)
FROM BOOK_FACT,BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM
WHERE BOOK_FACT.BOOK_ID = BOOK_DIM.BOOK_ID AND
     BOOK_FACT.LOC_ID = BOOK_LOCATION_DIM.LOC_ID AND
     BOOK_FACT.TIME_ID = TIME_DIM.TIME_ID AND
     TIME_DIM.CAL_DATE = DATE '2016-03-02'
GROUP BY BOOK_LOCATION_DIM.CITY;
```

3.3.3.2 การวิเคราะห์และตอบคำถามบนคลังข้อมูลแบบ Snowflake Schema

เนื่องจากในหัวข้อหัวนี้มีการใช้ตัวอย่างเดียวกันกับในหัวข้อที่แล้ว ดังนั้นจะมีการตัดภาพสำหรับการอธิบายบางส่วนออกไป

1) Roll-up คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้สูงขึ้นเพื่อทำการสรุปข้อมูล โดยการตอบคำถามแบบ roll-up สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้



รูป 3.54 แสดงการ Roll-up ใน Time dimension จากวันเป็นเดือน (Relation)

เพื่อให้เราสามารถตอบคำถาม Roll-up ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.50 คำสั่ง SQL Roll-up (Snowflake Schema)

```
SELECT     MIN(T_MONTH.MONTH) AS "MONTH",
           B_NAME.NAME,
           L_CITY.CITY,
           SUM(SELL)
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
           T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
           T_MONTH.MONTH = 'MARCH'
GROUP BY B_NAME.NAME,L_CITY.CITY
ORDER BY 1;
```

2) Drill-down คือการปรับระดับของความละเอียดของมุมมองที่ต้องการให้ต่ำลง เพื่อเป็นการดูรายละเอียดของข้อมูล โดยการตอบคำถามแบบ Drill-down สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้

MONTH	NAME	CITY	SUM(SELL)
March	Winnie the Pooh	Arlington	29
March	Tropical Food	Arlington	2
March	Winnie the Pooh	Boston	20
March	Tropical Food	Boston	7

Drill-down

DAY	MONTH	NAME	CITY	SELL
2	March	Winnie the Pooh	Arlington	18
2	March	Winnie the Pooh	Boston	20
2	March	Tropical Food	Boston	5
3	March	Tropical Food	Arlington	2
3	March	Tropical Food	Boston	2
3	March	Winnie the Pooh	Arlington	11

รูป 3.55 แสดงการ Drill-down ใน Time dimension จากเดือนเป็นวัน (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-down ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.51 คำสั่ง SQL Drill-down (Snowflake Schema)

```
SELECT     T_DAY.DAY AS "DAY",
           T_MONTH.MONTH AS "MONTH",
           B_NAME.NAME,
           L_CITY.CITY,
           SELL
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND

           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND

           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
           T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY
ORDER BY 1;
```

3) Drill-out คือเพิ่มมุมองที่ต้องการวิเคราะห์เพื่อเป็นการดูรายละเอียดของข้อมูลในมุมมองอื่นประกอบ โดยการตอบคำถามแบบ Drill-out สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้

รูป 3.56 แสดงการ Drill-out โดยการเพิ่ม Book dimension (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-out ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.52 คำสั่ง SQL Drill-out (Snowflake Schema)

```
SELECT     MIN(T_DAY.DAY) AS "DAY",
           MIN(T_MONTH.MONTH) AS "MONTH",
           L_CITY.CITY,
           B_NAME.NAME,
           SUM(SELL)
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
           T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
           T_DAY.DAY = 2 AND
           T_MONTH.MONTH = 'MARCH'
GROUP BY L_CITY.CITY,B_NAME.NAME;
```

4) Slice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองเพียงมุมมองเดียว โดยการตอบคำถามแบบ Slice สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้

DAY	MONTH	SELL
2	March	43

รูป 3.57 แสดงการ Slice บน Time dimension ของวันที่ 2 มีนาคม (Relation)

เพื่อให้เราสามารถตอบคำถาม Slice ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.53 คำสั่ง SQL Slice (Snowflake Schema)

```
SELECT     T_DAY.DAY AS "DAY",
           T_MONTH.MONTH AS "MONTH",
           SUM(SELL) AS "SELL"
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
           T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
           T_DAY.DAY = 2 AND
           T_MONTH.MONTH = 'MARCH'
GROUP BY T_DAY.DAY,T_MONTH.MONTH;
```

5) Dice คือการเลือกพิจารณาหรือวิเคราะห์ข้อมูลโดยใช้มุมมองหลายมุมมอง โดยการตอบคำถามแบบ Slice สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้

DAY	MONTH	CITY	NAME	SELL
2	March	Boston	Winnie the Pooh	20

รูป 3.58 แสดงการ Dice โดยใช้ 3 dimensions (Relation)

เพื่อให้เราสามารถตอบคำถาม Dice ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.54 คำสั่ง SQL Dice (Snowflake Schema)

```
SELECT      T_DAY.DAY AS "DAY",
            T_MONTH.MONTH AS "MONTH",
            L_CITY.CITY AS "CITY",
            B_NAME.NAME AS "NAME",
            SUM(SELL) AS "SELL"
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE       BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
            BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
            BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
            T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
            T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
            T_DAY.DAY = 2 AND
            T_MONTH.MONTH = 'MARCH' AND
            L_CITY.CITY = 'BOSTON' AND
            B_NAME.NAME = 'WINNIE THE POOH'
GROUP BY
T_DAY.DAY,T_MONTH.MONTH,L_CITY.CITY,B_NAME.NAME;
```

6) Drill-Across เกิดจากการรวมกันของข้อมูลของ cube 2 อัน กล่าวคือ การวิเคราะห์ข้อมูลบางครั้งไม่สามารถใช้เพียง cube เดียวแล้ววิเคราะห์ได้ จึงต้องมีการใช้ cube อีกอันซึ่งมี conformed dimensions มาช่วยในการวิเคราะห์โดยการตอบคำถามแบบ Drill-Across สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation จะสามารถแสดงได้ดังนี้

DATE	SELL
03/02/2016	53

รูป 3.59 แสดงการ Drill-Across โดยใช้ Shop sell และ Internet sell (Relation)

เพื่อให้เราสามารถตอบคำถาม Drill-Across ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.55 คำสั่ง SQL Drill-Across (Snowflake Schema)

```
SELECT    MIN(DATE1) AS "DATE",SUM(SELL1) AS "SELL"
FROM (
SELECT    T_DAY.CAL_DATE AS "DATE1",SUM(SELL) AS
"SELL1"
FROM      BOOK_INT_FACT_SF,B_NAME,L_WEBSITE,T_DAY
WHERE     BOOK_INT_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY
          AND BOOK_INT_FACT_SF.WEB_LOC_KEY =
          L_WEBSITE.WEB_LOC_KEY AND
          BOOK_INT_FACT_SF.TIME_KEY = T_DAY.TIME_KEY
          AND T_DAY.CAL_DATE = DATE '2016-03-2'
GROUP BY  T_DAY.CAL_DATE
UNION
SELECT    T_DAY.CAL_DATE AS "DATE2",SUM(SELL) AS
"SELL2"
FROM      BOOK_FACT_SF,B_NAME,L_CITY,T_DAY
WHERE     BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
          BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
          BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
```

7) Pivot table คือตารางที่แสดงข้อมูลโดยใช้มุมมอง 2 มุมมอง โดยจะเป็นการแสดงค่าผลรวมย่อยและค่าผลรวมทั้งหมด ซึ่งอนุญาตให้ทำการ roll-up และ drill-down ได้และสามารถมีหลายๆมุมมองใน 1 แกนของตารางได้ ซึ่งการตอบคำถามในรูปแบบของ Relation นั้นไม่สามารถตอบได้โดยตรง โดยสามารถตอบคำถามให้ใกล้เคียงในรูปแบบของ Relation ได้ดังนี้

GENRE	CITY	SUBTOTAL
Cooking	Boston	7
Childrens books	Boston	20
Cooking	Arlington	2
Childrens books	Arlington	29

รูป 3.60 แสดงตัวอย่างการตอบคำถามที่ให้ข้อมูลใกล้เคียงกับ Pivot table (Relation)

เพื่อให้เราสามารถตอบคำถามตามรูป 3.60 ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.56 คำสั่ง SQL ที่ให้คำตอบใกล้เคียง Pivot table (Snowflake Schema)

```
SELECT     B_GENRE.GENRE AS "GENRE",
           L_CITY.CITY AS "CITY",
           SUM(SELL) AS "SUBTOTAL"
FROM BOOK_FACT_SF,B_NAME,B_GENRE,L_CITY,T_DAY
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND
           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND
           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           B_NAME.GENRE_KEY = B_GENRE.GENRE_KEY
GROUP BY B_GENRE.GENRE,L_CITY.CITY
```

8) Ranking คือการนำข้อมูลที่ต้องการวิเคราะห์มาจัดอันดับ โดยการตอบคำถามแบบ Ranking สามารถแสดงผลลัพธ์ที่เราต้องการในรูปแบบของ Relation ได้ดังนี้

RANK	DAY	MONTH	CITY	SUM(SELL)
1	2	March	Boston	25
2	2	March	Arlington	18

รูป 3.61 แสดงการทำ Ranking บน Time dimension และ Location dimension (Relation)

จากรูป 3.61 เป็นการทำ Ranking เพื่อหาคำตอบว่ายอดขายหนังสือในวันที่ 2 มีนาคม 2560 นั้นมียอดขายในแต่ละเมืองเป็นเท่าไหร่และทำการจัดอันดับออกมา และเพื่อให้เราสามารถตอบคำถาม Ranking ที่เป็นรูปแบบ Relation ดังกล่าวได้ เราสามารถใช้คำสั่งได้ดังนี้

โปรแกรม 3.57 คำสั่ง SQL Ranking (Snowflake Schema)

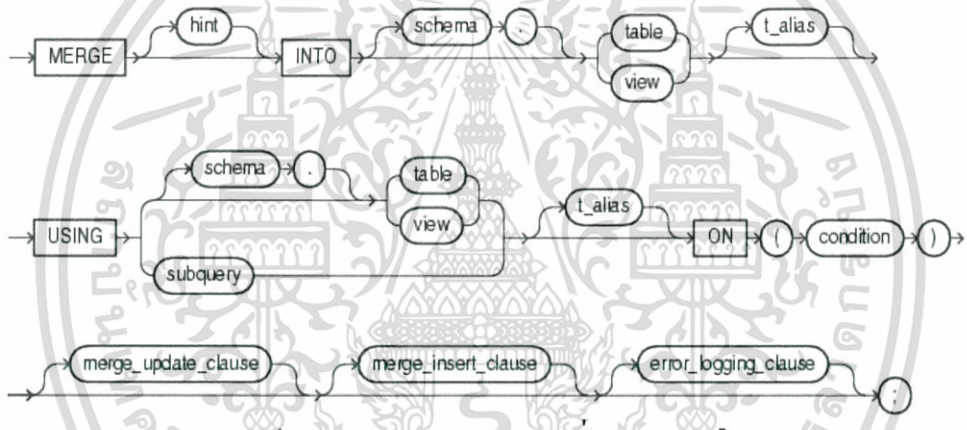
```
SELECT     RANK() OVER (ORDER BY SUM(SELL) DESC) AS
"RANK",
           MIN(T_DAY.DAY) AS "DAY",
           MIN(T_MONTH.MONTH) AS "MONTH",
           L_CITY.CITY,
           SUM(SELL)
FROM BOOK_FACT_SF,B_NAME,L_CITY,T_DAY,T_WEEK,T_MONTH
WHERE      BOOK_FACT_SF.BOOK_KEY = B_NAME.BOOK_KEY AND

           BOOK_FACT_SF.LOC_KEY = L_CITY.LOC_KEY AND

           BOOK_FACT_SF.TIME_KEY = T_DAY.TIME_KEY AND
           T_DAY.WEEK_KEY = T_WEEK.WEEK_KEY AND
           T_WEEK.MONTH_KEY = T_MONTH.MONTH_KEY AND
           T_DAY.DAY = 2 AND
           T_MONTH.MONTH = 'MARCH'
GROUP BY L_CITY.CITY;
```

บทที่ 4

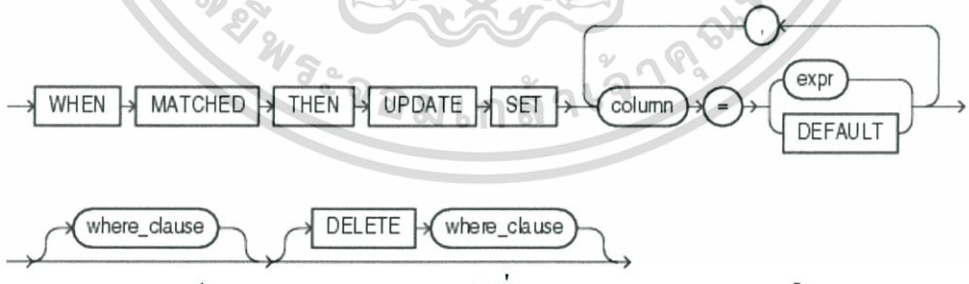
การทดลองและผลการทดลอง

4.1 การทดลองและสรุปผลการแก้ปัญหา Slowly Changing Dimensions ด้วยออราเคิลเอสคิวแอล[7]

การแก้ปัญหา Slowly Changing Dimensions นั้นในโครงงานนี้จะใช้คำสั่ง SQL ในการแก้ปัญหา โดยจะใช้คำสั่ง MERGE ในการตรวจสอบข้อมูลที่มีอยู่เดิมใน Dimension table และข้อมูลใหม่ที่จะโหลดเข้ามา หากข้อมูลเหมือนกันตามเงื่อนไขที่กำหนดเราสามารถทำการ UPDATE หรือ DELETE ข้อมูลที่มีอยู่เดิมได้ และในกรณีที่เป็นข้อมูลใหม่ที่ยังไม่ปรากฏใน Dimension table ก็จะทำการ INSERT ข้อมูลใหม่นั้นลงไป

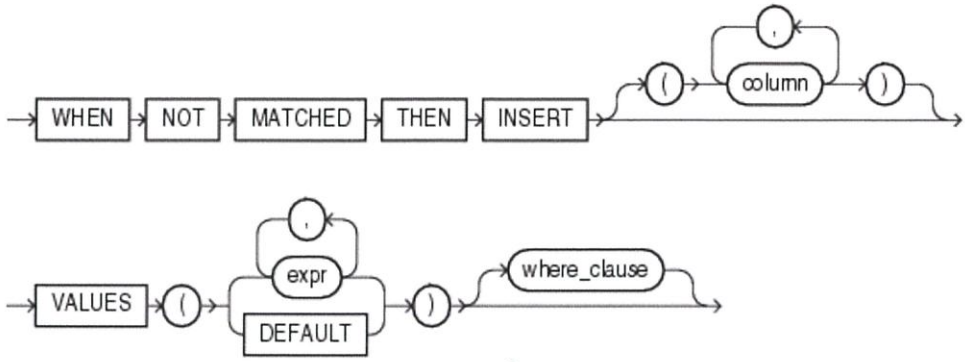


รูป 4.1 แสดง Syntax ของคำสั่ง MERGE[7]

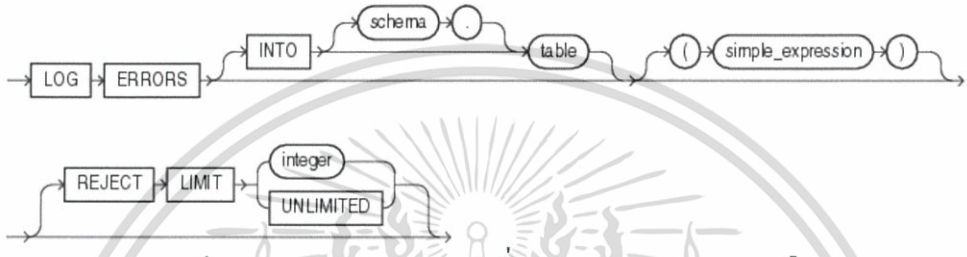


รูป 4.2 แสดง Syntax ของคำสั่ง merge_update_clause[7]

[7] Mary Beth Roeser. January 2016. Oracle Database SQL Language Reference, 12c Release 1 (12.1).



รูป 4.3 แสดง Syntax ของคำสั่ง merge_insert_clause[7]

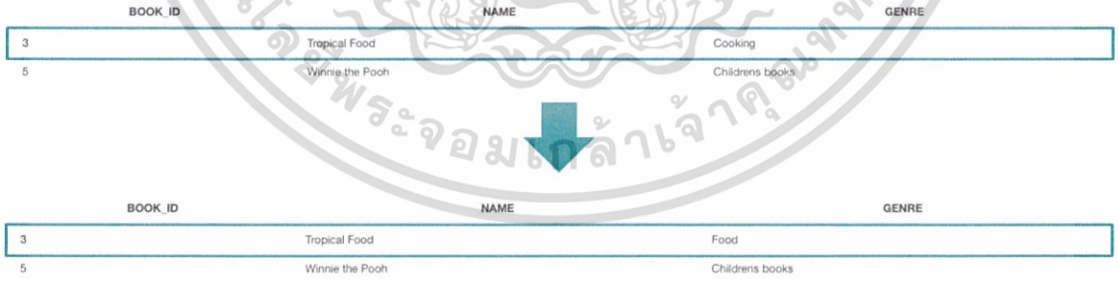


รูป 4.4 แสดง Syntax ของคำสั่ง error_logging_clause[7]

จากรูป 4.1 ถึง 4.4 นั้นเป็นการแสดงการอธิบายรูปแบบโครงสร้างภาษาของคำสั่ง MERGE

4.1.1 การทดลองและสรุปผลการทำ Slowly Changing Dimensions แบบที่ 1

วิธีนี้เป็นการแก้ปัญหาด้วยการอัปเดตค่าใหม่ทับค่าเก่าลงไปเลย ทำให้ข้อมูลเก่าที่อยู่ใน Dimension หายไป



รูป 4.5 แสดงการแก้ปัญหา Slowly Changing Dimension แบบที่ 1

จากรูป 4.5 นั้นเริ่มต้นข้อมูลที่มีอยู่ใน Book dimension นั้นปรากฏว่าหนังสือ Tropical Food เป็นหนังสือประเภท Cooking ต่อมาภายหลังได้มีการเปลี่ยนประเภทไปเป็น Food เมื่อเกิดการโหลดข้อมูลเข้าไปใน Book dimension ใหม่จะได้ผลลัพธ์ดังรูป โดยเราสามารถใช้คำสั่งเอสคิวแอลในการโหลดข้อมูลดังกล่าวได้ดังนี้

โปรแกรม 4.1 คำสั่ง Merge เพื่อทำ Slowly Changing Dimension แบบที่ 1

```
MERGE INTO BOOK_DIM_S1 BD
   USING (SELECT ROWNUM SKEY,BOOK_NAME, BOOK_GENRE
          FROM BOOK_TDB
          GROUP BY BOOK_NAME,BOOK_GENRE) N
   ON (BD.NAME = N.BOOK_NAME)
   WHEN MATCHED THEN
       UPDATE SET BD.GENRE = N.BOOK_GENRE
       WHERE BD.GENRE <> N.BOOK_GENRE
   WHEN NOT MATCHED THEN
       INSERT (BD.BOOK_ID,BD.NAME,BD.GENRE)
       VALUES (N.SKEY,N.BOOK_NAME,N.BOOK_GENRE);
```

4.1.2 การทดลองและสรุปผลการทำ Slowly Changing Dimensions แบบที่ 2

วิธีนี้เป็นการแก้ปัญหาด้วยการเพิ่ม (Insert) ข้อมูลใหม่ลงไป โดยที่ Dimension table จะมีคอลัมน์ที่บอกว่า Valid time ของ dimension value แต่ละอันและมีคอลัมน์บอกว่า version ของแต่ละ dimension value ด้วย แต่การแก้ปัญหาด้วยวิธีใหม่นี้จะทำให้เราได้ Surrogate key ใหม่ทุกครั้ง

BOOK_ID	NAME	GENRE	VALID_FROM	VALID_TO	NEWEST	VERS
3	Tropical Food	Cooking	03/02/2016	12/31/9999	YES	1
5	Winnie the Pooh	Childrens books	03/02/2016	12/31/9999	YES	1

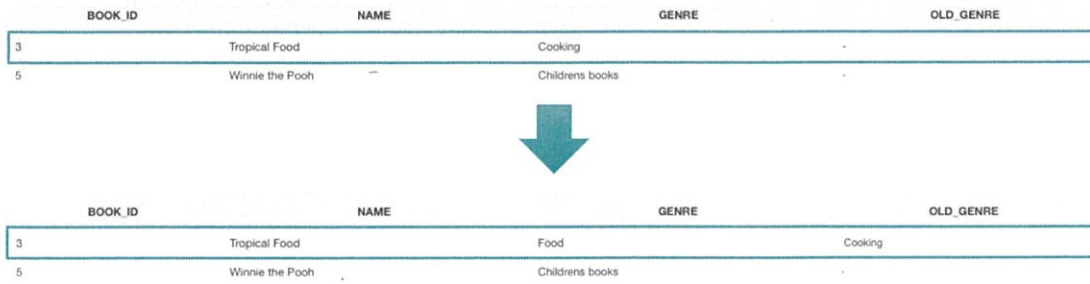
BOOK_ID	NAME	GENRE	VALID_FROM	VALID_TO	NEWEST	VERS
3	Tropical Food	Cooking	03/02/2016	04/01/2016	NO	1
6	Tropical Food	Food	04/01/2016	12/31/9999	YES	2
5	Winnie the Pooh	Childrens books	03/02/2016	12/31/9999	YES	1

รูป 4.6 แสดงการแก้ปัญหา Slowly Changing Dimension แบบที่ 2

เนื่องจากการทำ Slowly Changing Dimension แบบที่ 2 นั้นจะมีการอัปเดตค่าในคอลัมน์ Valid_to เป็นวันที่ถูกอัปเดตและอัปเดตค่าในคอลัมน์ Newest เป็น NO และมีการ Insert ข้อมูลใหม่เข้าไปซึ่งเป็นข้อมูลล่าสุดตามรูป 4.6 เนื่องจากคำสั่ง Merge นั้นไม่สามารถใช้คำสั่ง UPDATE พร้อมกับการใช้คำสั่ง INSERT ในกรณีที่หนังสือเล่มนั้นได้มีการเปลี่ยนประเภทไปจากเดิม ทำให้เราไม่สามารถใช้คำสั่ง Merge วิธีเดิมได้อาจจะต้องมีการใช้คำสั่งอื่นๆมาช่วยในการทำ Slowly Changing Dimension แบบที่ 2 นี้

4.1.3 การทดลองและสรุปผลการทำ Slowly Changing Dimensions แบบที่ 3

วิธีนี้เป็นการแก้ปัญหาโดยการเพิ่มคอลัมน์เข้าไปใน dimension table เพื่อไว้สำหรับเก็บค่าเก่าของ dimension value ของ attribute ที่ถูกเปลี่ยนแปลง



BOOK_ID	NAME	GENRE	OLD_GENRE
3	Tropical Food	Cooking	-
5	Winnie the Pooh	Childrens books	-

BOOK_ID	NAME	GENRE	OLD_GENRE
3	Tropical Food	Food	Cooking
5	Winnie the Pooh	Childrens books	-

รูป 4.7 แสดงการแก้ปัญหา Slowly Changing Dimension แบบที่ 3

จากรูป 4.7 เราสามารถใช้คำสั่ง MERGE เพื่อทำ Slowly Changing Dimension แบบที่ 3 ได้ดังนี้

โปรแกรม 4.2 คำสั่ง Merge เพื่อทำ Slowly Changing Dimension แบบที่ 3

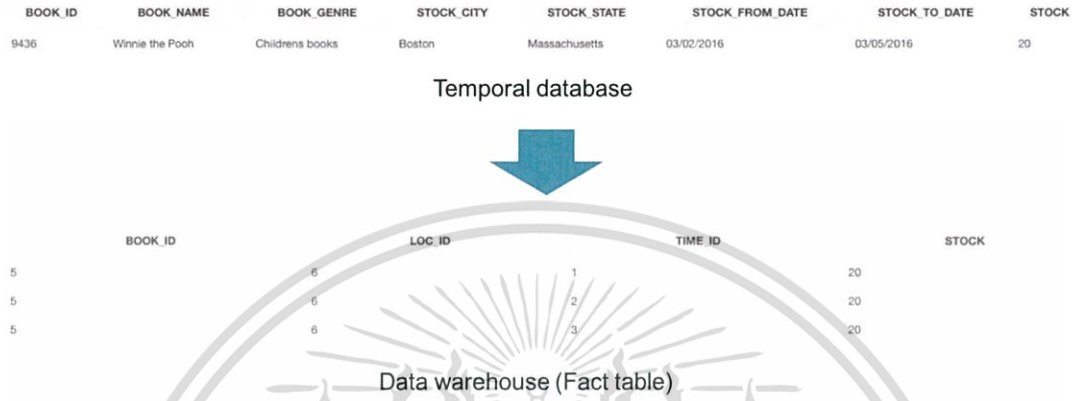
```
MERGE INTO BOOK_DIM_S3 BD
   USING (SELECT ROWNUM SKEY,BOOK_NAME, BOOK_GENRE
          FROM BOOK_TDB
          GROUP BY BOOK_NAME,BOOK_GENRE) N
   ON (BD.NAME = N.BOOK_NAME)
   WHEN MATCHED THEN
      UPDATE SET BD.GENRE = N.BOOK_GENRE,
                 BD.OLD_GENRE = BD.GENRE
      WHERE BD.GENRE <> N.BOOK_GENRE
   WHEN NOT MATCHED THEN
      INSERT (BD.BOOK_ID,BD.NAME,
              BD.GENRE,BD.OLD_GENRE)
      VALUES (N.SKEY,N.BOOK_NAME,N.BOOK_GENRE,NULL);
```

4.2 การทดลองและสรุปผลการทำ ETL (Extract-Transform-Load) ด้วยภาษาเอสคิวแอล 2011[6]

การทำ ETL ในหัวข้อนี้จะเป็นการโหลดข้อมูลจากฐานข้อมูลเชิงเวลาซึ่งมีช่วงเวลา (Period) เป็นแบบ Valid time เข้าสู่คลังข้อมูล (Data warehouse) ซึ่งมีเวลาเป็นจุดของเวลา ทำให้เราต้องแตกช่วงเวลาจากแหล่งข้อมูลที่เป็นฐานข้อมูลเชิงเวลาไปเป็นจุดของเวลาในคลังข้อมูล โดยเราจะต้องทำการนำเสนอจุดของเวลาให้ครบทุกจุดที่มีอยู่ในช่วงเวลานั้น โดยกำหนดให้มีแหล่งข้อมูลที่เป็นฐานข้อมูลเชิงเวลาดังนี้

BOOK_ID	BOOK_NAME	BOOK_GENRE	STOCK_CITY	STOCK_STATE	STOCK_FROM_DATE	STOCK_TO_DATE	STOCK
9436	Winnie the Pooh	Childrens books	Boston	Massachusetts	03/02/2016	03/05/2016	20
7493	Tropical Food	Cooking	Boston	Massachusetts	03/02/2016	03/05/2016	5
7493	Tropical Food	Cooking	Arlington	Virginia	03/05/2016	03/10/2016	2
9436	Winnie the Pooh	Childrens books	Arlington	Virginia	03/05/2016	03/10/2016	11
9436	Winnie the Pooh	Childrens books	Arlington	Virginia	03/02/2016	03/05/2016	18

รูป 4.8 ตัวอย่างแหล่งข้อมูลที่เป็นฐานข้อมูลเชิงเวลา



รูป 4.9 แสดงตัวอย่างการโหลดข้อมูลจากฐานข้อมูลเชิงเวลาเข้าสู่คลังข้อมูล

จากรูป 4.9 นั้นจะเห็นได้ว่าที่แหล่งข้อมูล (Temporal database) นั้นมีช่วงเวลาตั้งแต่ 2 มีนาคม 2016 ถึง 4 มีนาคม 2016 เมื่อทำการโหลดข้อมูลดังกล่าวเข้าสู่คลังข้อมูลลงไปที่ Fact table จะได้ดังรูปคือจะมีจุดเวลาทั้งหมด 3 จุดของวันที่ 2, 3 และ 4 มีนาคม 2016 จำนวน 20 เล่ม เนื่องจากข้อมูลของการจัดเก็บสินค้าจึงทำให้มีค่า 20 เท่ากันทุกจุด โดยเราสามารถใช้คำสั่ง SQL2011 ดังต่อไปนี้ในการโหลดข้อมูลตัวอย่างได้

โปรแกรม 4.3 คำสั่งโหลดข้อมูลจากฐานข้อมูลเชิงเวลาเข้าสู่คลังข้อมูล

```
CREATE TABLE BOOK_FACT(
     BOOK_ID INTEGER,
     LOC_ID INTEGER,
      TIME_ID INTEGER,
     STOCK INTEGER,
     PRIMARY KEY (BOOK_ID,LOC_ID,TIME_ID),
     FOREIGN KEY (BOOK_ID) REFERENCES
BOOK_DIM(BOOK_ID),
     FOREIGN KEY (LOC_ID) REFERENCES
BOOK_LOCATION_DIM(LOC_ID),
     FOREIGN KEY (TIME_ID) REFERENCES TIME_DIM(TIME_ID)
);
```

```
INSERT INTO BOOK_FACT
SELECT BOOK_DIM.BOOK_ID,BOOK_LOCATION_DIM.LOC_ID,
       TIME_DIM.TIME_ID,STOCK
FROM BOOK_DIM,BOOK_LOCATION_DIM,TIME_DIM,BOOK_TDB
WHERE BOOK_TDB.BOOK_NAME = BOOK_DIM.NAME AND
     BOOK_TDB.BOOK_GENRE = BOOK_DIM.GENRE AND
     BOOK_TDB.STOCK_CITY = BOOK_LOCATION_DIM.CITY AND
     BOOK_TDB.STOCK_STATE = BOOK_LOCATION_DIM.STATE AND
     (BOOK_TDB.STOCK_FROM_DATE, BOOK_TDB.STOCK_TO_DATE)
     OVERLAPS (TIME_DIM.CAL_DATE,TIME_DIM.CAL_DATE)
GROUP BY BOOK_DIM.BOOK_ID,BOOK_LOCATION_DIM.LOC_ID,
      TIME_DIM.TIME_ID,STOCK;
```

จากโปรแกรม 4.3 นั้นเราสามารถใช้คำสั่ง OVERLAP ที่อยู่ในมาตรฐาน SQL 2011 ในการเลือกช่วงเวลาที่เราต้องการได้ โดยในคำสั่งนั้นจะมีการ join ข้อมูลจาก Dimension tables ต่างๆเข้าด้วยกันและเลือกช่วงเวลาที่ปรากฏในแหล่งข้อมูลโดยใช้คำสั่ง OVERLAP เพื่อตรวจสอบว่าจุดของเวลาใดบางที่เกิดจากการ join นั้นอยู่ในช่วงเวลาของแหล่งข้อมูล ถ้าหากเป็นจุดของเวลาที่เกิดจากการ join นั้นอยู่ในช่วงเวลาที่ต้องการเราก็จะทำการโหลดข้อมูลนั้นเข้าสู่คลังข้อมูล

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุปของโครงงาน

เนื่องจากฐานข้อมูลเชิงเวลานั้นจะมีการเก็บช่วงเวลาของ Fact ว่าเป็นจริงตั้งแต่เมื่อไหร่ถึงเมื่อไหร่ ทำให้เราสามารถย้อนดูข้อมูลที่เป็นจริงในช่วงเวลาต่างๆได้ โดยฐานข้อมูลเชิงเวลาดังกล่าวนั้นสามารถใช้คุณสมบัติเชิงเวลาที่มีอยู่ในมาตรฐาน SQL 2011 ในการช่วงตรวจสอบความถูกต้องของข้อมูลและสามารถช่วยตอบคำถามเชิงเวลาได้สะดวกมากขึ้น ซึ่งโครงงานชิ้นนี้ก็ได้ทำการทดสอบคุณสมบัติเชิงเวลาต่างๆที่มีอยู่ในมาตรฐาน SQL 2011 บนฐานข้อมูล Oracle database 12c ซึ่งปรากฏว่า Oracle database 12c นั้นมีการสนับสนุนคุณสมบัติเชิงเวลาเพียงไม่กี่คุณสมบัติเท่านั้นตามที่ได้ทำการทดสอบไปในบทที่ 3 หลังจากนั้น โครงงานชิ้นนี้ก็ได้ศึกษาเรื่องของคลังข้อมูล เรื่องของการทำ ETL และการวิเคราะห์คลังข้อมูล หลังจากนั้นก็ได้ศึกษาการและทดลองการโหลดข้อมูลจากฐานข้อมูลเชิงเวลาโดยใช้เครื่องมือที่เป็นมาตรฐาน SQL 2011 ช่วยในการโหลดข้อมูล

การโหลดข้อมูลจากแหล่งข้อมูลที่เป็นฐานข้อมูลเชิงเวลานั้นทำให้เราสามารถนำข้อมูลต่างๆในอดีตมาวิเคราะห์ได้ ซึ่งเป็นข้อดีของฐานข้อมุลเชิงเวลาที่เก็บข้อมูลในอดีตไว้ด้วย ซึ่งการวิเคราะห์ต่างๆนั้นถ้าหากเรามีข้อมูลต่างๆในอดีตมาช่วยในการวิเคราะห์เพิ่มก็จะช่วยให้การวิเคราะห์คลังข้อมูลนั้นให้ผลลัพธ์ที่ละเอียดมากยิ่งขึ้น และการศึกษาการทำ ETL จากแหล่งข้อมูลที่เป็นฐานข้อมูลเชิงเวลาโดยใช้ภาษา SQL 2011 นั้นจะช่วยให้เราสามารถทำการโหลดข้อมูลเชิงเวลาได้สะดวกมากยิ่งขึ้น

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไขปัญหา

1) การติดตั้งฐานข้อมูล Oracle database 12c ช่วงแรกใช้เวลานานสำหรับการติดตั้งครั้งแรก แต่สามารถแก้ไขได้ด้วยการศึกษาคู่มืออย่างละเอียดเพื่อติดตั้งให้รวดเร็วขึ้นได้
2) Oracle database 12c นั้นไม่สนับสนุนคุณสมบัติเชิงเวลาได้ครบถ้วน ทำให้การแก้ปัญหาบางอย่างไม่สามารถเรียกใช้คุณสมบัติที่ต้องการได้โดยตรง อาจจะต้องใช้วิธีการอื่นๆในการแก้ปัญหา
3) โครงสร้างภาษา (Syntax) ของภาษาเอสคิวแอลใน Oracle database 12c นั้น ในส่วนของคำสั่ง MERGE ยังไม่สามารถรองรับการ INSERT ในกรณีที่ข้อมูลใน Dimesnion table กับแหล่งข้อมูลนั้นเหมือนกัน
4) การตอบคำถามที่เป็น Pivot table นั้น Oracle database 12c ยังไม่สามารถให้ผลลัพธ์ที่เป็น Pivot table แบบซับซ้อนได้ แต่สามารถตอบคำถามอย่างง่ายได้

5.3 แนวทางในการพัฒนาต่อ

1) ศึกษาการทำ ETL จากแหล่งข้อมูลเชิงเวลาในกรณีอื่นๆให้ละเอียดมากขึ้น โดยเฉพาะเรื่องของ Identifier ของข้อมูลที่จะทำการโหลดลงคลังข้อมูล
2) ศึกษาการแก้ปัญหา Slowly Changing Dimensions แบบอื่นๆ เช่น การใช้คุณสมบัติเด่นของฐานข้อมูลเชิงเวลามาช่วยแก้ปัญหาและอาจจะศึกษาต่อถึงการนำมาตรฐาน SQL 2011 มาใช้แก้ปัญหาเพื่อเพิ่มความสะดวก
3) ถ้าหากเราสามารถทำ ETL จากแหล่งข้อมูลเชิงเวลาได้ดีแล้ว อาจจะมีการศึกษาต่อถึงความเป็นไปได้ที่จะนำคลังข้อมูลมาอยู่บนฐานข้อมูล กล่าวคือเราสามารถตอบคำถามต่างๆเพื่อวิเคราะห์ข้อมูลจากฐานข้อมูลได้โดยตรง ซึ่งจะมีการรบกวนระบบฐานข้อมูลให้น้อยที่สุดเพื่อตอบคำถามดังกล่าว
4) ศึกษาการการสร้าง Dimension ที่มี Hierarchies แบบแปลกๆ เช่น Parent-Child Hierarchies, Unbalanced Hierarchies, Non-Covering Hierarchies, Non-Strict Hierarchies, Multiple Hierarchies และ Parallel Hierarchies
5) ศึกษาเรื่อง Temporal Data warehouse เกี่ยวกับเรื่องของ Period time, Granularity time และ Time dimension

เอกสารอ้างอิง

[1] Christian S. Jensen, Torben Bach Pedersen, Christian Thomsen. 2010. **Multidimensional Databases and Data Warehousing.** M. Tamer Ozsu, University of Waterloo.

[2] Paulraj Ponniah. 2010. **Data warehousing Fundamentals for IT Professionals 2nd Edition.** A John Wiley & Sons, INC., Publication

[3] Chitchanok Potipat, Thidarat Poonpetchphan. 2016. **Temporal Data Warehouse.** Faculty of Engineering, KMITL.

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

[5] Krishna Kulkarni, Jan-Eike Michels. 2012. **Temporal features in SQL:2011.** IBM Corporation.

[6] Mary Beth Roeser. January 2016. **Oracle Database SQL Language Reference, 12c Release 1 (12.1).** E41329-20. Oracle Corporation.

[7] Paul Lane, Padmaja Potineni. November 2014. **Oracle Database, Data Warehousing Guide 12c Release 1 (12.1).** E41670-08. Oracle Corporation

[8] Oracle Corporation. 2016. **Temporal Validity Support**. [Online]. Available : https://docs.oracle.com/database/121/ADFNS/adfns_design.htm#ADFNS967

[9] Oracle Corporation. 2016. **PL/SQL Reserved Words and Keywords**. [Online]. Available : https://docs.oracle.com/database/121/LNPLS/reservewords.htm#LNPLS019