การพัฒนาระบบคลังข้อมูลการประกันวินาศภัย

Data Warehouse for Insurance

โดย

นายชินพัฒน์ เคลือวัลย์

รหัส 45066068

H002169

อาจารย์ที่ปรึกษา

ผศ.ดร.วรพจน์ กรีสุระเดช

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2546

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบคลังข้อมูลการประกันวินาศภัย
นักศึกษา	นายชินพัฒน์ เคลือวัลย์
อาจารย์ที่ปรึกษา	ผศ.ดร. วรพจน์ กรีสุระเดช
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขวงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2546

บทคัดย่อ

การวิเคราะห์ข้อมูลให้อยู่ในรูปแบบสารสนเทศ เป็นสิ่งที่มีบทบาทมากในธุรกิจปัจจุบัน ซึ่งข้อมูลที่มีอยู่ในฐานข้อมูลนั้นส่วนใหญ่จะอยู่ในรูปของรายการที่เกิดจากกิจกรรมทางธุรกิจ และการวิเคราะห์ข้อมูลในมุมมองต่างๆ ไม่สามารถทำได้อย่างมีประสิทธิภาพ สำหรับโครงงานนี้เป็นการพัฒนาระบบวิเคราะห์ข้อมูลการประกันวินาศภัย โดยจะทำการเปลี่ยนรูปแบบข้อมูลให้เป็นข้อมูลสรุปซึ่งจัดเก็บในดาต้าแวร์เฮาส์ และจะทำการวิเคราะห์ข้อมูลเพื่อนำเสนอในรูปแบบรายงาน

Title	Data Warehouse for Insurance
Student	Mr. Chinnapat Kluewan
Advisor	Asst. Prof. Dr. Worapoj Kreesuradej
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2003

ABSTRACT

Information analysis is most important for business today. While data in database is business transaction and information analysis for decision support is non-efficiency. This project is providing data warehouse for insurance data and information analysis for present into excusive report.

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมา

การประกันวินาศภัยรถยนต์ภาคสมัครใจ จัดอยู่ในหมวดหนึ่งของประเภทการประกันวินาศภัย ซึ่งเป็นธุรกิจที่มีการเติบโตค่อนข้างสูง ทำให้มีการแข่งขันในกลุ่มของบริษัทในธุรกิจนี้สูงด้วยเช่นกัน ดังนั้นจึงมีความต้องการที่จะทำวิธีที่มีประสิทธิภาพเข้ามาใช้ในการวิเคราะห์ข้อมูลต่างๆ ที่เป็นรายการที่เกิดขึ้นจากอดีต ถึงปัจจุบัน เพื่อช่วยในการตัดสินใจ และเป็นแนวทางในการดำเนินธุรกิจในอนาคต ดังนั้นกระบวนการดังกล่าว จึงเป็นเรื่องที่น่าสนใจในการศึกษา เพื่อเป็นการสนับสนุนสารสนเทศให้แก่กลุ่มบริษัทประกันภัย

การจัดเตรียมข้อมูลให้อยู่ในรูปแบบที่เหมาะสมสำหรับประมวลผล เป็นเรื่องที่มีความสำคัญสำหรับการทำให้การประมวลผลมีประสิทธิภาพ เนื่องจากข้อมูลที่อยู่ในระบบงานแหล่งต่างๆ นั้นเป็นข้อมูลที่มีรูปแบบที่ดีหรือและไม่มีรูปแบบที่ไม่เหมาะสมในการประมวลผล ดังนั้นจึงมีความจำเป็นที่จะต้องทำการเลือกและจัดข้อมูลให้อยู่ในรูปแบบที่ง่ายและมีประสิทธิภาพในการประมวลผล สำหรับข้อมูลที่จะนำมาใช้ในการประมวลผลได้แก่ ข้อมูลการทำประกันวินาศภัย ข้อมูลการจ่ายค่าสินไหมทดแทน และข้อมูลเชิงสถิติต่างๆ ซึ่งข้อมูลเหล่านี้จะต้องเปลี่ยนรูปแบบรายการที่เป็นทรานเซคชั่น (Transaction) ให้อยู่ในรูปแบบของข้อมูลอนุกรมเวลาในรูปแบบต่างๆ ให้เหมาะสมกับความต้องการในการวิเคราะห์ ดังเช่น การวิเคราะห์อัตราการเติมโตของธุรกิจ สถิติตามช่วงเวลา แนวโน้มสำหรับโอกาสทางธุรกิจ และการวิเคราะห์ความเสี่ยง เป็นต้น

ดาต้าแวร์เฮาส์ (Data Warehouse) เป็นหมวดหมู่ของเทคโนโลยีที่สนับสนุนการตัดสินใจ ซึ่งมีวัตถุประสงค์ให้ผู้ทำงานสามารถใช้ในการบริหาร จัดการ และวิเคราะห์ได้ดีขึ้นและรวดเร็วกว่าเดิม ในปัจจุบันดาต้าแวร์เฮาส์เป็นจุดที่สร้างความสนใจในธุรกิจประเภทต่างๆ เนื่องจากข้อมูลที่มีจำนวนมาก เช่นรายการขายจากระบบประกันวินาศภัย มีจำนวนมากในระดับกิกะไบท์ ซึ่งในการประมวลผลของดาต้าเบสที่มีอยู่ในปัจจุบัน ส่วนใหญ่เป็น RDBMS นั้นได้รับความนิยมมาก แต่ความสามารถในการประมวลผลรายการเพื่อสนับสนุนความต้องการสารสนเทศแบบปรับเปลี่ยน (Ad Hoc Report or Query) นั้นไม่สามารถทำงานได้อย่างมีประสิทธิภาพ เนื่องจากข้อมูลนั้นมีข้อจำกัดในเรื่องของปริมาณข้อมูล ความถูกต้องของข้อมูล และความซับซ้อนของประเภทรายการเหล่านี้เป็นปัจจัยทำให้ความต้องการรายงานหรือการค้นหานั้นไม่มีประสิทธิภาพ จึงจำเป็นที่หาวิธีที่

จะช่วยแก้ปัญหาดังกล่าว ซึ่งเป็นการลดรูปของข้อมูลและจัดรูปแบบให้เหมาะสมอย่างมีความสัมพันธ์กัน ให้อยู่ในดาต้าแวร์เฮาส์ ซึ่งดาต้าแวร์เฮาส์นี้เองจะมีการแยกออกจากระบบจัดการฐานข้อมูลทั่วไป เนื่องจากเหตุผลหลาย อย่างเช่น การสนับสนุนการทำงานประมวลผลเชิงวิเคราะห์แบบออนไลน์ หน้าที่การทำงาน และความต้องการความสามารถสูงในการประมวลผล

สำหรับทฤษฎีในการสร้างดาต้าแวร์เฮาส์นั้นมีหลักการหลายอย่างที่จะต้องคำนึงถึง ทั้งในเรื่องของประเภทข้อมูล ช่วงความเหมาะสม ต่างๆ เพื่อให้ได้ข้อมูลที่ถูกนั้น ยังพบว่าปัญหาของการสร้างดาต้าแวร์เฮาส์ยังมีอีกหลายอย่างซึ่งในโครงการนี้จะกล่าวถึงปัญหาที่อาจจะเกิดและแนวทางแก้ไข เพื่อเป็นแนวทางในการพัฒนาระบบ และในทฤษฎีของการสร้างฐานข้อมูลสำหรับดาต้าแวร์เฮาส์จะใช้โมเดลของฐานข้อมูลแบบหลายมิติ (Modeling Multidimensional Database) ซึ่งจะสามารถนำมาพัฒนาระบบประกันวินาศภัยรถยนต์ภาคสมัครใจโดยใช้ดาต้าแวร์เฮาส์ ได้

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาหลักการในการจัดทำดาต้าแวร์เฮาส์ ปัญหาและการแก้ไขที่เหมาะสม

1.2.2 เพื่อศึกษาทฤษฎีของโมเดลของฐานข้อมูลแบบหลายมิติ ถึงหลักการในการสร้างฐานข้อมูลอย่างมีประสิทธิภาพ

1.2.3 เพื่อนำทฤษฎีมาทำการพัฒนาระบบประกันวินาศภัยรถยนต์ภาคสมัครใจ โดยใช้ดาต้าแวร์เฮาส์

1.3 ขอบเขตการดำเนินงาน

โครงการนี้เป็นการศึกษาเพื่อพัฒนาระบบประกันวินาศภัยรถยนต์ภาคสมัครใจโดยใช้ดาต้าแวร์เฮาส์ โดยมีขอบเขตของการศึกษาดังต่อไปนี้

1.3.1 เป็นการพัฒนาระบบโดยใช้โมเดลของฐานข้อมูลแบบหลายมิติ ในการสร้างดาต้าแวร์เฮาส์

1.3.2 ข้อมูลที่ใช้ในการวิเคราะห์เป็นข้อมูลการทำประกันวินาศภัยรถยนต์ภาคสมัครใจที่ได้จากกรมการประกันภัย สำหรับข้อมูลระหว่างปีพ.ศ. 2544 ถึง พ.ศ. 2546

1.3.3 ระบบวิเคราะห์ข้อมูลเพื่อทำการสร้างดาต้าแวร์เฮาส์ พัฒนาด้วย VBScript ซึ่งจะมีขั้นตอนการเลือกและการตรวจสอบการเปลี่ยนแปลงของข้อมูล แล้วทำการปรับปรุงดาต้าแวร์เฮาส์

1.4 ขั้นตอนและวิธีการดำเนินงาน

ในการสร้างระบบประกันวินาศภัยรถยนต์ภาคสมัครใจโดยใช้ดาต้าแวร์เฮาส์ ของโครงการพัฒนาระบบงานที่จัดทำขึ้นนี้มีขั้นตอนการดำเนินงานดังต่อไปนี้

1.4.1 ศึกษาเทคโนโลยีของดาต้าแวร์เฮาส์ เพื่อเป็นแนวทางในการดำเนินงาน

1.4.2 ศึกษาหลักการการสร้างดาต้าแวร์เฮาส์ ปัญหาและวิธีการแก้ไข

1.4.3 ศึกษาโมเดลของฐานข้อมูลแบบหลายมิติ เพื่อดำเนินการสร้าง Schema ของดาต้าแวร์เฮาส์

1.4.4 ดำเนินการรวบรวมข้อมูลการทำประกันวินาศภัยรถยนต์ภาคสมัครใจจากระบบฐานข้อมูลที่เกี่ยวข้อง

1.4.5 ดำเนินการพัฒนาระบบงาน โดยแบ่งส่วนการทำงานเป็น 4 ขั้นตอนดังนี้คือ

1.4.5.1 การวิเคราะห์รูปแบบความต้องการของข้อมูลประเภทต่างๆ เพื่อเป็นขอบเขตในการเลือกข้อมูล

1.4.5.2 การล้างข้อมูลและการนำข้อมูลเข้าดาต้าแวร์เฮาส์

1.4.5.3 การสร้างแบบสอบถามหรือรายงาน เพื่อเรียกใช้ข้อมูลจากดาต้าแวร์เฮาส์

1.4.5.4 การแสดงผลหรือการนำเสนอข้อมูล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถเรียนรูปและเข้าใจทฤษฎีของดาต้าแวร์เฮาส์ ในเรื่องของความแตกต่างจากระบบฐานข้อมูลทั่วไป

1.5.2 สามารถสร้างดาต้าแวร์เฮาส์ได้ตามโมเดลของฐานข้อมูลแบบหลายมิติ และการสร้างข้อมูลในดาต้าแวร์เฮาส์

1.5.3 สามารถพัฒนาระบบที่สนับสนุนระบบประกันวินาศภัยรถยนต์ภาคสมัครใจโดยใช้ดาต้าแวร์เฮาส์

บทที่ 2
เทคโนโลยีดาต้าแวร์เฮาส์

ด้าตาแวร์เฮาส์เป็นระบบความสำคัญอย่างมากที่ช่วยในเรื่องของการสนับสนุนการตัดสินใจ (Decision Support) ซึ่งในปัจจุบันอุตสาหกรรมดาต้าเบสได้พยายามมุ่งเน้นมากขึ้น ดังจะพบว่าหลายผลิตภัณฑ์ ต่างมุ่งเน้นสำหรับสินค้าและบริการที่เป็นไปได้ที่เป็นเชิงการค้าให้มีการทำดาต้าแวร์เฮาส์เพื่อสนับสนุนเรื่องของการประมวลผลเพื่อช่วยในระบบการตัดสินใจ เช่นเทคโนโลยีการประมวลผลเชิงวิเคราะห์แบบออนไลน์ (On-Line Analytical Processing : OLAP) ซึ่งมีการเสนอเครื่องมือต่างๆ ในการช่วยสร้าง แต่ทั้งนี้ ในเรื่องของหลักการสร้างที่เหมาะสมยังเป็นศิลป์ที่ผู้ทำการสร้างจะต้องเข้าใจและออกแบบให้เหมาะสม

2.1 เบื้องต้นของดาต้าแวร์เฮาส์

ดาต้าแวร์เฮาส์ เป็นเทคโนโลยีที่ช่วยสนับสนุนการตัดสินใจ โดยมีเป้าหมายเพื่อให้คนทำงาน (ผู้บริหาร, ผู้จัดการ, นักวิเคราะห์) สามารถทราบว่าจะทำอย่างไรให้ดี และรวดเร็ว จากอดีตที่ผ่านมากจะพบว่าสินค้าและบริการมีการเพิ่มจำนวนยอดขายอย่างรวดเร็วจึงทำให้ดาต้าแวร์เฮาส์มีการพัฒนาอย่างมากเช่นกัน และเหตุผลหลักที่สำคัญคือ ดาต้าแวร์เฮาส์สนับสนุนการประมวลผลเชิงวิเคราะห์แบบออนไลน์ ฟังก์ชั่น และประสิทธิภาพสูง ซึ่งแตกต่างกับระบบการประมวลผลรายการ (On-Line Transaction Processing : OLTP) ที่จะประมวลผลกับระบบฐานข้อมูลสำหรับงานทั่วไป

ในความแตกต่างของดาต้าแวร์เฮาส์ คือ เป้าของของระบบช่วยสนับสนุนการตัดสินใจ ด้านประวัติ ข้อมูลสรุป และข้อมูลในการตัดสินใจจะมีความสำคัญมากกว่ารายละเอียด ซึ่งจะมีความสามารถมากในเรื่องของช่วงเวลา ซึ่งทำงานได้ดีกว่าระบบฐานข้อมูลทั่วไป โดยเฉพาะโครงการที่มีขนาดข้อมูลเป็นร้อยกิกะไบท์ การทำงานจะมีประสิทธิภาพมากถ้าเป็นคิวรีแบบ Ad Hoc เพราะการที่ทำงานของคิวรีที่มีความซับซ้อน จะสามารถเข้าถึงข้อมูลในจำนวนล้านเรคคอร์ดด้วยฟังก์ชั่นการทำงานของ Join และ Aggregates แล้วให้ผลลัพธ์ของคิวรีตามเวลาที่ไม่นาน ซึ่งเป็นเรื่องที่มีความสำคัญมากกว่า ผลของ รายการทรานเซคชั่น

หลักการทั่วไปของโมเดลในดาต้าแวร์เฮาส์ ก็คือ Multidimensional หรือ มุมมองหลายมิติจะช่วยให้สามารถทำการวิเคราะห์ข้อมูลที่ซับซ้อน และเห็นภาพได้ง่าย ตัวอย่างเช่น ยอดขาย

กรมธรรม์ในดาต้าแวร์เฮาส์ จะมีรายละเอียดเกี่ยวกับประเภทกรมธรรม์ ประเภทรถยนต์ ระยะเวลาความคุ้มครอง อัตราเบี้ยประกันภัย และค่าสินไหมทดแทน การที่จะสรุปข้อมูลนั้นขึ้นอยู่กับมุมมองที่เราสนใจว่าต้องการเลือกข้อมูลอะไร ช่วงเวลาแบบใด เช่น วัน เดือน ไตรมาส ปี ในส่วนประเภทกรมธรรม์ เช่น ประกันภัยประเภท 1, 2, 3 เป็นต้น

ในการทำงานของฐานข้อมูลที่มีการออกแบบอย่างดีเพื่อให้สามารถทำงานแบบ OLTP (On-Line Transaction Processing) จะมีการประมวลผลที่ซับซ้อนกว่า OLAP (On-Line Analytical Processing) และประสิทธิภาพในการแสดงผลลัพธ์นั้นยอมรับไม่ได้สำหรับใช้ในระบบสนับสนุนการตัดสินใจ ซึ่งอาจจะเกิดจากการทำงานผิดพลาดของฐานข้อมูลเอง หรือการทำคิวรีซ้ำของข้อมูลเก่าที่ไม่ถูกต้อง เพราะว่าระบบฐานข้อมูลจะทำการบันทึกเพียงข้อมูลปัจจุบันเท่านั้น ระบบสนับสนุนการตัดสินใจโดยปกติจะใช้ข้อมูลจากแหล่งที่มีความแตกต่างกัน ซึ่งอาจจะรวมถึงฐานข้อมูลภายนอกในระบบอื่นๆ ซึ่งแหล่งข้อมูลที่แตกร่างกันอาจจะมีคุณภาพของข้อมูลที่แตกต่างกัน หรือไม่สามารถนำเสนอได้ รหัส หรือรูปแบบ ซึ่งจะมีความขัดแย้งกันได้ สรุปได้ว่าการสนับสนุนของโมเดลข้อมูลแบบหลายมิติ และการทำงาน OLAP มีความต้องการข้อมูลที่มีการจัดโครงสร้างโดยเฉพาะ วิธีการเข้าถึงข้อมูล และวิธีการนำมาใช้งาน ซึ่งไม่ได้เตรียมไว้ในระบบฐานข้อมูลในเชิงธุรกิจทั่วไป จึงเป็นเหตุผลที่ต้องแยกดาต้าแวร์เฮาส์ออกจากระบบการทำงานของฐานข้อมูล

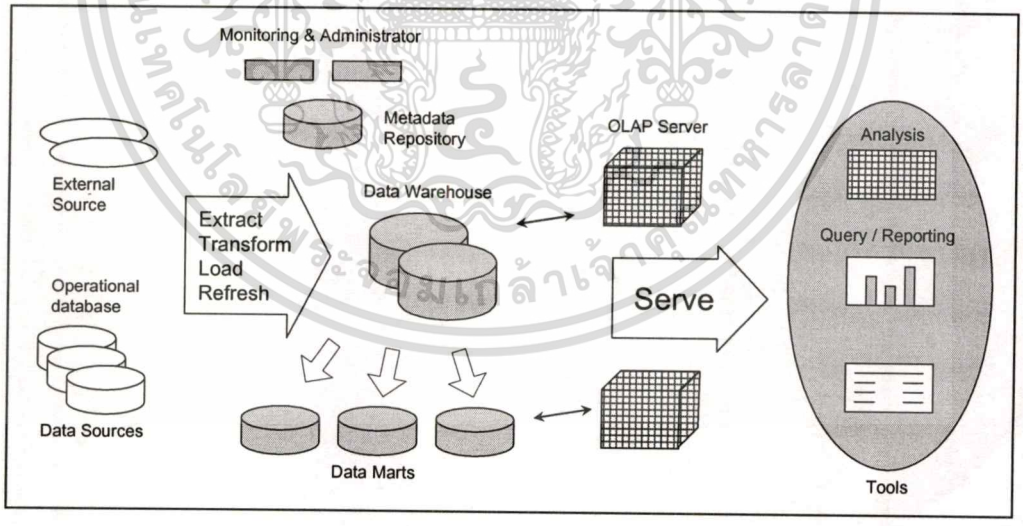
ดาต้าแวร์เฮาส์อาจจะมีนำมาใช้บนมาตราฐาน หรือเพิ่มเติมใน RDBMSs ที่เรียกว่า รีเลชั่นนอล OLAP (ROLAP) ซึ่งเครื่องที่ทำการติดตั้งจะทำการบันทึกข้อมูลเสมือนบันทึกในฐานข้อมูลแบบรีเลชั่นนอล และจะสนับสนุนคำสั่งเพิ่มเติมของ SQL และการเข้าถึงข้อมูลและวิธีการที่มีประสิทธิภาพที่ใช้กับโมเดลข้อมูลแบบหลายมิติ ซึ่งจะแตกต่างจากการใช้ มัลติไดเมนชั่น OLAP ที่เครื่องจะทำการบันทึกข้อมูลแบบมุมมองหลายมิติ และเป็นโครงสร้างพิเศษ (เช่น อาเรย์) และการใช้งาน OLAP จะต้องอยู่ภายใต้โครงสร้างข้อมูลเฉพาะนี้

มีวิธีการสร้างและปรับปรุงดาต้าแวร์เฮาส์ที่มากกว่าติดตั้ง OLAP การออกแบบ schema และการทำคิวรีที่ซับซ้อน เนื่องจากความแตกต่างของปัจจุบันสถาปัตยกรรมที่มีอยู่นี้สามารถเลือกได้ และหลายองค์กรต้องการที่จะรวมระบบให้เป็นเอนเตอร์ไพรส์แวร์เฮาส์ (Enterprise Warehouse) ที่จะรวมทั้งหมดเข้าด้วยกัน เช่น ลูกค้า สินค้า ทรัพย์สิน เป็นต้น ซึ่งอยู่ในระบบต่างๆ อย่างไรก็ดีการสร้างเอนเตอร์ไพรส์แวร์เฮาส์จะต้องใช้เวลานานและมีความซับซ้อนในการประมวลผล และใช้ค่าใช้จ่ายสูง ซึ่งขึ้นกับนโยบายทางธุรกิจของระดับบริหาร และใช้เวลาหลายปีกว่าจะประสบความสำเร็จ บางองค์กรอาจจะแยกข้อมูลออกเป็นดาต้ามาร์ท (Data Mart) ซึ่งจะทำแยกแผนกโดยเลือกเฉพาะส่วนที่สนใจในการสร้าง ดาต้ามาร์ทเหล่านี้สามารถทำงานได้เร็วและไม่ต้องการความ

สอดคล้องกันในระหว่างดาต้ามาร์ท ขององค์กรที่มีขนาดใหญ่ แต่อาจจะเกิดปัญหาในการรวมเข้าด้วยกัน และใช้เวลานานในการทำงาน ถ้าธุรกิจที่ต้องการความสมบูรณ์ของระบบ โมเดลนี้ไม่เหมาะสม

2.2 สถาปัตยกรรมและการประมวลผลแบบ End-to-End

จากรูป 2-1 เป็นการรวมเครื่องมือสำหรับดึงข้อมูลจากฐานข้อมูลหลายๆ แหล่ง และแหล่งข้อมูลภายนอก สำหรับการปรับแก้ข้อมูลให้อยู่ถูกต้อง (Clearing) การเปลี่ยนรูปข้อมูล และการรวมข้อมูลเข้าไว้ในดาต้าแวร์เฮาส์ โดยจะมีการรีเฟรชเป็นระยะซึ่งเป็นผลที่เกิดจากการปรับปรุงของแหล่งข้อมูล และจากการลบข้อมูลออกจากดาต้าแวร์เฮาส์ ในส่วนที่เพิ่มเติมเข้าดาต้าแวร์เฮาส์หลักอาจจะมีหลายดาต้ามาร์ท ข้อมูลที่อยู่ในแวร์เฮาส์และดาต้ามาร์ทจะถูกบันทึกและจัดการโดยหนึ่งหรือหลายเซิฟเวอร์แวร์เฮาส์ ซึ่งจะอยู่ในมุมมองหลายมิติของข้อมูลที่มีความแตกต่างของเครื่องมือที่สร้าง (front end tools) ซึ่งได้แก่ query tools, report writers, analysis tools และ data mining tools ซึ่งระบบเหล่านี้จะมีที่จัดเก็บและจัดการเมตะดาต้า และเครื่องมือสำหรับจัดการแวร์เฮาส์ในการบริหารและติดตาม



รูปที่ 2-1 แสดงสถาปัตยกรรมทั่วไปของการทำดาต้าแวร์เฮาส์

แวร์เฮาส์อาจจะเป็นรูปแบบกระจายเพื่อให้เกิดความสมดุลในการทำงาน (Load balancing) มากที่สุดเท่าที่จะทำได้ ดังตัวอย่างเช่นสถาปัตยกรรมแบบกระจาย การสร้างฐานข้อมูลเพื่อเก็บเมตะดาต้าของแต่ละส่วนที่กระจายอยู่ (Fragment) ของแวร์เฮาส์ เพื่อที่จะสามารถนำมารวมกันได้ ซึ่งการ

เลือกสถาปัตยกรรมแบบนี้อาจจะต้องมีค่าใช้จ่ายที่สูงขึ้นเช่นกัน สำหรับมุมมองที่กลับกัน การรวมหรือการแยกแต่ละแวร์เฮาส์เป็นดาต้ามาร์ท ก็จต้องมีแหล่งเก็บแยกจากกัน และต้องมีระบบจัดการแบบกระจายด้วย

การออกแบบและการปรับให้ดาต้าแวร์เฮาส์ลดความซับซ้อนของการประมวลผลจะต้องคำนึงกิจกรรมดังนี้

- กำหนดรูปแบบสถาปัตยกรรม วางแผนความจุของข้อมูล และเลือกเซิฟเวอร์ที่จะใช้บันทึกเป็นฐานข้อมูลกับ OLAP เซิฟเวอร์ และเลือก tools ที่จะใช้งาน
- การปรับให้เซิฟเวอร์ แหล่งเก็บข้อมูลและเครื่องใช้งานให้เข้ากัน
- ออกแบบ schema และวิวของแวร์เฮาส์
- กำหนดโครงสร้างของแวร์เฮาส์ที่ต้องการ การปรับแก้ข้อมูล การแบ่งกลุ่มข้อมูล และวิธีการเข้าถึงข้อมูล
- เชื่อมต่อแหล่งข้อมูลโดยใช้เกตเวย์ ผ่าน ODBC หรือใช้ตัวกลางอื่นๆ (Wrappers)
- ออกแบบและประยุกต์สคลิปต์สำหรับการแยกข้อมูล การปรับแก้ข้อมูล การเปลี่ยนรูปข้อมูล การโหลดและการรีเฟรชแวร์เฮาส์
- ทำการข้อมูลจาก schema และวิวที่กำหนดไว้, สคลิปต์
- ออกแบบและพัฒนา end-user แอพพลิเคชั่น
- ตรวจสอบแวร์เฮาส์และแอพพลิเคชั่น

2.3 เครื่องมือ Back End และยูทิลิตี้

ระบบการทำดาต้าแวร์เฮาส์จะใช้ความหลากหลายของการแยกข้อมูล และเครื่องมือในการปรับแก้ข้อมูล และโปรแกรมช่วยการโหลดและรีเฟรช สำหรับทำการย้ายแวร์เฮาส์ การแยกข้อมูลจากแหล่งข้อมูลที่ต่างกัน โดยส่วนมากจะทำโดยผ่านเกตเวย์และอินเตอร์เฟสมาตราฐาน เช่น Information Builders EDA/SQL, ODBC, Oracle Open Connect, Sybase Enterprise Connect, Informix Enterprise Gateway เป็นต้น

2.3.1 การปรับแก้ข้อมูล (Data Cleaning)

เนื่องจากดาต้าแวร์เฮาส์เป็นการใช้งานสำหรับการสร้างการตัดสินใจ ดังนั้นจึงมีความสำคัญที่ว่าข้อมูลในแวร์เฮาส์จะต้องถูกต้อง แต่เนื่องจากขนาดของข้อมูลจากแหล่งต่างๆ ที่เกี่ยวข้องมีโอกาสที่จะเกิดความผิดพลาดและผิดปกติในตัวของข้อมูล ซึ่งเครื่องมือที่ช่วยในการค้นหาข้อมูลที่ผิดปกติไม่สามารถทำการตรวจหาได้ จึงจำเป็นจะต้องล้างเพิ่มเริ่มทำงานใหม่

ตัวอย่างเช่น ขนาดของฟิลด์ที่ไม่คงที่ คำอธิบายที่ไม่ถูกต้อง ค่าของสัญลักษณ์ไม่ถูกต้อง มีการป้อนค่าผิด และมีการฝ่าฝืนความถูกต้องของข้อมูล (Integrity constraints) สำหรับสิ่งที่เกิดโดยไม่คาดคิดมาก่อน เช่นการเลือกฟิลด์จากแหล่งข้อมูลที่ไม่เหมาะสม ข้อมูลมี 3 ความสัมพันธ์ แต่มีอะไรที่แตกต่างกัน

การแบ่งประเภทของเครื่องมือที่ทำการปรับแก้ข้อมูล ได้แก่ data migration tools เป็นการย้ายข้อมูลตามกฎที่กำหนด เช่นการแทนที่คำว่า "gender" ด้วย "sex" ซึ่งระบบจัดการแวร์เฮาส์ของ Prism เป็นตัวอย่างของกรณีนี้, data scrubbing tools ใช้ความรู้เฉพาะขอบเขตที่กำหนด เช่น รหัสไปรษณีย์ ซึ่งจะใช้หลักการในการแยกและใช้เทคนิคฟัซซี่แม็ทชิ่ง เครื่องมือบางชนิดอาจจะทำการหาความสัมพันธ์ของแหล่งข้อมูล สำหรับ data auditing tools อาจจะค้นหากฎของความสัมพันธ์ของข้อมูลโดยการสแกนข้อมูล ซึ่งเครื่องมือดังกล่าวอาจจะพิจารณาจากความแตกต่างของ data mining tools

2.3.2 การโหลดข้อมูล

หลักจากทำการแยกข้อมูล การแก้ไขและการเปลี่ยนรูปของข้อมูลแล้ว จะต้องทำการโหลดข้อมูลเข้าดาต้าแวร์เฮาส์ สิ่งที่จะต้องทำก่อนการประมวลผลก็คือ การตรวจสอบกฎความถูกต้อง (Integrities constraints) การเรียงลำดับ การคำนวณผลรวม การสรุป และการคำนวณอื่นที่จะต้องบันทึกในตารางที่สร้างไว้ในแวร์เฮาส์ การสร้าง index สำหรับการเข้าถึงข้อมูล การแบ่งพื้นที่ในการจัดเก็บ โดยทั่วไปจะใช้วิธีการโหลดแบบ batch สำหรับยูทิลิตี้ที่ช่วยในการโหลดข้อมูลเข้าแวร์เฮาส์จะต้องสามารถให้ผู้ดูแลระบบสามารถทำการดูสถานะ เพื่อที่จะยกเลิก สั่งให้หยุดหรือทำงานต่อ และการเริ่มต้นทำงานใหม่ถ้าเกิดการผิดผลาดเนื่องจากข้อมูลสูญเสียความถูกต้อง

ยูทิลิตี้ที่ช่วยในการโหลดข้อมูลสำหรับดาต้าแวร์เฮาส์ ต้องใช้เวลาในการโหลดในเวลาจำกัด (ปกติจะทำช่วงกลางคืน) เมื่อแวร์เฮาส์ offline เพื่อทำการรีเฟรชข้อมูล การโหลดจะใช้เวลานานมาก ถ้าข้อมูลเป็นเทราไบท์ อาจจะใช้เวลาเป็นสัปดาห์หรือเป็นเดือน ดังนั้นการทำ pipeline และการแบ่งงานทำแบบคู่ขนาน ก็จะช่วยลดระยะเวลาการทำงานได้มาก โดยการทำเป็นลักษณะ batch transaction ขณะที่ทำการประมวลผลรายการปัจจุบันเสร็จก็จะทำการ commits และระบบจะเริ่มประมวลผลใหม่ถ้าเกิดความผิดพลาดที่จุด checkpoint ล่าสุด

อย่างไรก็ดี การทำงานแบบคู่ขนานก็ยังคงใช้เวลานาน โดยส่วนมากยูทิลิตี้ที่ใช้ในงานด้านธุรกิจจะใช้วิธีการโหลดข้อมูลเมื่อมีการเพิ่มหรือลดขนาดจำนวนข้อมูลพร้อมกัน ซึ่งเป็นเพียงการ

update ค่าเมื่อมีการ insert ถึงแม้วิธีการนี้ก็จะเป็นการเพิ่มการทำงานของการคิวรี แต่การทำงานของระบบฐานข้อมูลในการเข้าถึงข้อมูลนั้นมีกระบวนการที่ดีและการใช้อินเด็กซ์

2.3.3 รีเฟรชแวร์เฮาส์

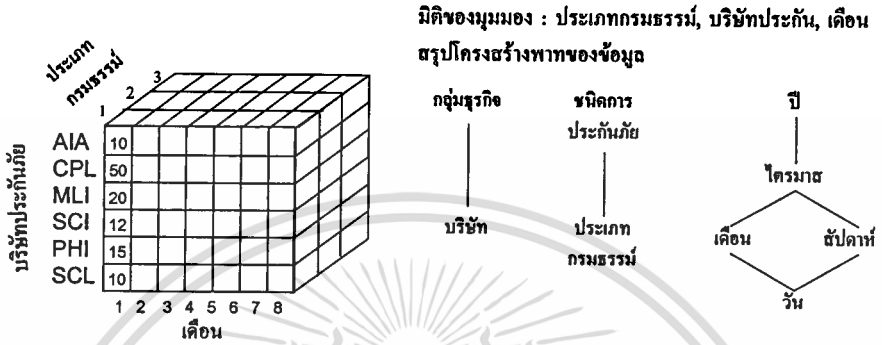
การรีเฟรชแวร์เฮาส์ หรือการปรับปรุงข้อมูลในแวร์เฮาส์ให้ทันสมัยจากแหล่งข้อมูลที่ทำงานร่วมกันของข้อมูลหลักและบันทึกข้อมูลเหล่านั้นในแวร์เฮาส์ มีสองสิ่งที่ต้องพิจารณา คือ เมื่อไรที่จะทำการรีเฟรช และจะทำการรีเฟรชอย่างไร โดยปกติ แวร์เฮาส์จะมีการรีเฟรชเป็นระยะเวลา เช่น ทุกวัน หรือทุกสัปดาห์ ในบางระบบอาจจะมีความจำเป็นที่ต้องมีการปรับปรุงเสมอเพื่อให้เป็นข้อมูลปัจจุบัน จึงจำเป็นที่จะต้องทำการปรับปรุงตามทุกครั้งที่มีการเปลี่ยนแปลง นโยบายการรีเฟรสจะถูกกำหนดโดยผู้บริหารระบบ โดยพิจารณาจากความต้องการของผู้ใช้งานและความคับคั้งของเครือข่าย และอาจจะแตกต่างกันระหว่างแหล่งข้อมูลที่ต่างกัน

เทคนิคของการรีเฟรชอาจจะขึ้นอยู่กับลักษณะของแหล่งข้อมูลและความจุของฐานข้อมูล ประเภทของแหล่งข้อมูลอาจจะเป็นระบบไฟล์ หรือ เป็นฐานข้อมูลที่มีราคาแพงก็ได้ ซึ่งในการต้องทำงานกับระบบงานเก่า เราจะต้องทราบว่าระบบงานนั้นสามารถสนับสนุนกับการทำเรพพลิเคท (Replicate) หรือไม่เมื่อแหล่งข้อมูลมีการเปลี่ยนแปลง ตัวอย่างของเทคนิคเรพพลิเคท เช่น การทำ log ไฟล์ ซึ่งอาจจะเป็น table ในฐานข้อมูลแล้วสร้าง trigger ทำเมื่อมีการเปลี่ยนแปลงรายการข้อมูลก็จะไปเพิ่มบันทึก log และเมื่อระบบเรพพลิเคททราบว่าเกิดการเปลี่ยนแปลงโดยดูจาก log ก็จะทำการปรับปรุงข้อมูลทันที ซึ่งวิธีจะเป็นการเพิ่มการทำงานให้กับฐานข้อมูลมากขึ้น เนื่องจากไม่สามารถหาวิธีอื่นที่จะทำการตรวจสอบการเปลี่ยนแปลงข้ามระบบฐานข้อมูลที่เป็น DBMSs ที่ต่าง Vender เพราะยังไม่มาตราฐาน APIs สำหรับการเข้าไปดึงข้อมูลใน log ของทรานเซคชั่นได้ ดังนั้นการทำเรพพลิเคทเซิฟเวอร์จะถูกใช้ในการรีเฟรชดาต้าแวร์เฮาส์ แต่อย่างไรก็ดีในแต่ละรอบที่ทำการรีเฟรชจะขึ้นกับจำนวนข้อมูลที่ต้องทำของยูทิลิตี้ที่ทำการโหลดข้อมูลด้วย

2.3.4 Conceptual Model และเครื่องมือ Front End

โมเดลในการย้ายข้อมูลนั้นจะหมายถึง เครื่องมือ front end การออกแบบฐานข้อมูล และคิวรีเอนจิ้นสำหรับ OLAP ที่เป็น วิวข้อมูลในแวร์เฮาส์แบบหลายมิติ ในดาต้าโมเดลแบบหลายมิติ จะมีการตัวเลขในการวัดค่าของการวิเคราะห์ ตัวอย่างเช่น การวัดจำนวนยอดขาย รายได้ ค่าใช้จ่าย และผลตอบแทนการลงทุน ตัวเลขแต่ละค่าที่ใช้วัดจะขึ้นอยู่กับชุดของไดเมนชั่น (dimension) ซึ่งเตรียมข้อมูลสำหรับการวัด เช่น การวัดค่าความเกี่ยวข้องของยอดขายในเมือง ชื่อสินค้า และวันที่เมื่อมีการขาย สมมติว่ามีไดเมนชั่นเดียวที่จะทำการวัด ซึ่งค่าในมิติเป็นที่ว่าง และแต่ละมิติ จะถูกอธิบายด้วยเซ็ตของแอตทริบิวส์ ตัวอย่างเช่น ประเภทของการประกันภัย แบบของการทำประกัน, บริษัท

ประกันภัย, เดือนที่ทำการเริ่มคุ้มครอง โดยสมมติว่า บริษัท A ได้ทำการขายกรมธรรม์ประเภท 1 ในเดือนเมษายน ได้จำนวน 5,000,000 บาท สามารถทำการวัดได้ว่าเป็นสัดส่วนของข้อมูลในทั้งกลุ่มโดยจะวัดได้จากความสัมพันธ์ระหว่างข้อมูลประเภทเดียวกันจากมิติที่มอง



รูปที่ 2-2 แสดงข้อมูลแบบมุมมองหลายมิติ

สำหรับความสามารถอื่นๆ ของคอนเซ็พชัลโมเดลตามตัวอย่างนี้ เป็นการสรุปโดย aggregate ในการวัดหนึ่งหรือหลายมิติ ที่ใช้เป็นตัวหลักในการทำงาน เช่น การคำนวณและแบ่งผลรวมยอดขาย ตามบริษัท ในแต่ละเดือน สำหรับการคำนวณโดยใช้สองตัววัด อย่างเช่น จำนวนกรมธรรม์ และเบี้ยประกัน ซึ่งจะมีการวัดค่าโดยใช้ลักษณะมิติได้เหมือนกัน

มิติแบบแบ่งช่วงเวลา (Time Dimension) เป็นรูปแบบที่สนับสนุนการตัดสินใจ เช่นการวิเคราะห์แนวโน้ม ซึ่งจะถูกใช้บ่อยมาก โดยจะมีการเพิ่มความสามารถลงไปในปฏิทินและฟังก์ชั่นต่างๆที่เกี่ยวกับมุมมองมิติแบบแบ่งช่วงเวลา

เครื่องมือ Front End

เครื่องมือ front end เป็นส่วนที่ใช้ในการนำเสนอข้อมูลจากโมเดลข้อมูลหลายมิติ โดยส่วนมากนิยมใช้โปรแกรม spreadsheet ในการใช้สำหรับการวิเคราะห์ทางธุรกิจ เนื่องจากกระดาษทำการ (Spreadsheet) จะสามารถมีพื้นที่ว่างในการใช้ประโยชน์เพื่อคำนวณอื่นๆ เพิ่มเติมได้ และเป็นความท้าทายที่จะทำให้ OLAP สามารถสนับสนุนการทำงานบนกระดาษทำการได้อย่างมีประสิทธิภาพบนฐานข้อมูลที่มีขนาดใหญ่หลายกิกะไบท์ได้ ซึ่ง Microsoft Excel ก็เป็น front end ถูกพัฒนาเพื่อรองรับการทำงานสำหรับมุมมองหลายมิตินี้

อย่างที่ทราบว่าความนิยมที่จะนำกระดาษทำการมาใช้งานเนื่องจากเป็นโปรแกรมที่สนับสนุนมุมมองหลายมิติ อีกงานหนึ่งที่เห็นได้ก็คือการทำ pivoting หรือการสรุปสาระสำคัญโดยการเปลี่ยนระดับการมอง เมื่อพิจารณาจากรูปที่ 2 ซึ่งสามารถแทนลงในกระดาษทำการ เมื่อแต่ละ

แถวอธิบายถึงบริษัทที่เกี่ยวกับยอดขาย โดยมีหนึ่งคอลัมภ์สำหรับแต่ละมุมมอง และคอลัมภ์พิเศษที่แทนค่ายอดขาย สำหรับตัวอย่างมุมมองแบบสรุปสาระสำคัญ (Pivot) จะมีการเลือกสองมิติที่ต้องการวัดค่าตำแหน่ง (x, y) ที่จะทำการสรุปค่าที่จะวัดจากมิติแรกโดยมีค่า x และมิติที่สองเป็นค่า y จากตัวอย่าง ถ้าเลือกมิติเป็นประเภทกรมธรรม์ และ เดือน ดังนั้น แกน x อาจจะแทนค่าทั้งหมดของประเภทกรมธรรม์ และแกน y อาจจะแทนค่าทั้งหมดของเดือน ดังนั้นจุด (x, y) จะแทนค่าสรุป (aggregate) ยอดขาย สำหรับสำหรับกรมธรรม์ประเภท x ในเดือน y และค่าอะไรก็ตามในแถวหรือคอลัมภ์ที่อยู่ในตารางข้อมูลเดิม ก็จะไปปรากฏเป็นตัวเลือกในตารางสรุปสาระสำคัญ

การทำงานอื่นๆ ที่เกี่ยวข้องกับการสรุปสาระสำคัญก็คือการ rollup หรือ drill-down, rollup เป็นการทำงานในลักษณะการทำข้อมูลปัจจุบันและทำการจัดกลุ่ม (group by) ด้วยมิติหนึ่งของทั้งหมด เช่นการ rollup ข้อมูลยอดขาย แล้วใช้ฟังก์ชั่นสรุปค่าของประเภทกรมธรรม์ และชื่อบริษัท ส่วน drill-down เป็นการทำงานย้อนกลับของ rollup

Slice_and_dice เป็นการทำงานที่ลดย่อมิติของข้อมูล เช่นการทำ projection ของข้อมูลให้เป็นซับเซ็ตของมิติทั้งหมด ตัวอย่างเช่น การทำ slice_and_dice ข้อมูลยอดขาย สำหรับการเลือกเฉพาะบริษัทประกันภัย แล้วสร้างเป็นตารางที่เป็นมิติสำหรับประเภทกรมธรรม์ และเดือนของยอดขาย นอกจากนี้ยังมีวิธีการอื่นๆ อีก เช่นการจัดลำดับ (ranking) และการคำนวณฟิลด์ข้อมูล

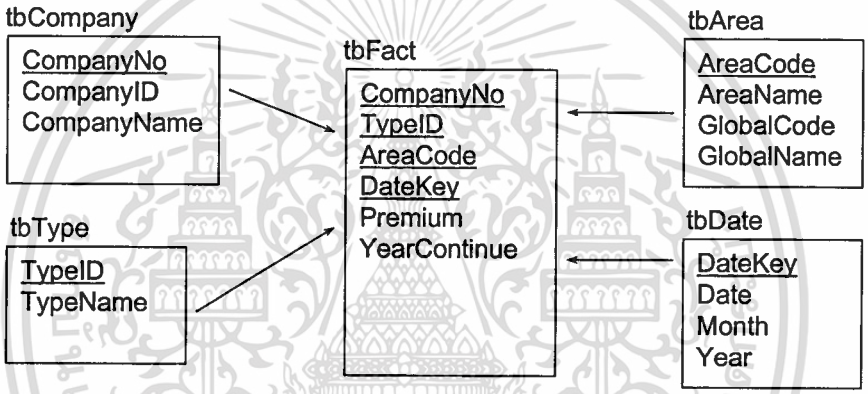
ทว่าสิ่งที่ให้มากับมุมมองหลายมิติของกระดาษทำการนั้นมีความน่าสนใจอย่างยิ่ง เนื่องจากเป็นการเพิ่มความสามารถอย่างมากให้ผู้ใช้งานสามารถทำการวิเคราะห์ข้อมูลธุรกิจ แต่สิ่งนี้ไม่ได้เป็นการเข้ามาแทนที่ระบบที่ทำการวิเคราะห์อยู่เดิม เช่นการค้นหา หรือการทำคิวรีที่ซับซ้อนสำหรับใช้งานจากระบบฐานข้อมูลเดิม ดังนั้น เครื่องมือที่สามารถให้ผู้ใช้งานระบบสามารถทำการคิวรีเพื่อดูเฉพาะข้อมูลที่ต้องการ ก็จะต้องเป็นโปรแกรมที่เข้าถึงข้อมูลดิบได้อย่างมีประสิทธิภาพตามการควบคุมของระบบฐานข้อมูลหลัก แต่ใช้กระบวนการของเครื่องมือ เช่น Microsoft Access จะสามารถสร้างคิวรีแบบ ad hoc SQL ที่เรียกว่า “pointing-and-clicking” ในความหลากหลายของเครื่องมือสำหรับดาต้าไมนิ่งเหล่านี้จะมีการใช้เป็นเครื่องมือ front end สำหรับดาต้าแวร์เฮาส์

2.4 กระบวนการออกแบบฐานข้อมูล

โมเดลของข้อมูลหลายมิติมุมมอง จากรายละเอียดก่อนหน้าจะสามารถนำมาใช้งานโดยตรงกับเซิฟเวอร์แบบ MOLAP ได้ แต่อย่างไรก็ดี ถ้ามีการใช้แบบ ROLAP เราจะต้องหาความสัมพันธ์ระหว่างโมเดลหลายมิติมุมมองกับการปฏิบัติงาน ซึ่งการออกแบบฐานข้อมูลแบบเชิงสัมพันธ์นี้จะสะท้อนภาพมุมมองหลายมิติของข้อมูลได้

ปกติ ER diagram (Entity Relationship) และเทคนิคการทำ Normalization เป็นวิธีการที่นิยมใช้ในการออกแบบฐานข้อมูลใน OLAP ต่างอย่างไรก็ดี การออกแบบฐานข้อมูลโดยใช้ ER diagram เป็นวิธีที่ไม่เหมาะสมสำหรับระบบสนับสนุนการตัดสินใจ เพราะมีส่วนที่ต้องคำนึงถึงประสิทธิภาพในการคิวรีและการโหลดข้อมูลซึ่งเป็นสิ่งสำคัญมาก

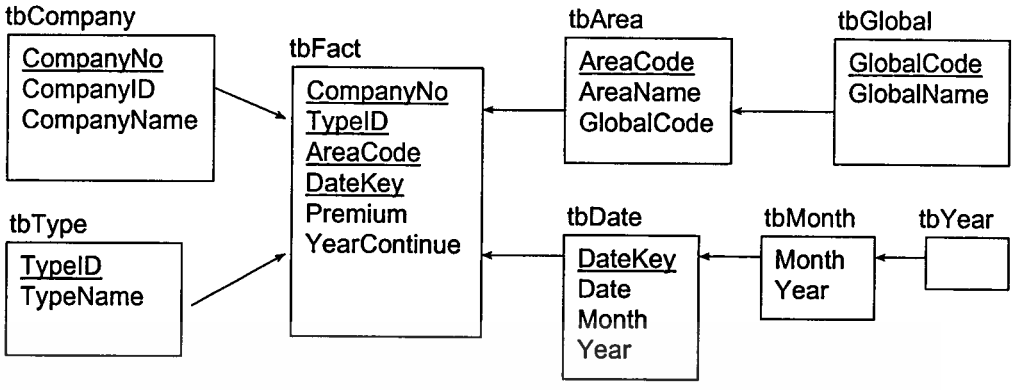
โดยส่วนมาก คาต้าแวร์เฮาส์จะใช้ star schema ในการแทนความหมายของโมเดลข้อมูลหลายมิติ ซึ่งฐานข้อมูลที่เป็นข้อมูลจริงจะเป็น ตารางเดี่ยว สำหรับแต่ละมิติ แต่ละแถวในตารางจะมีจุดเชื่อม (foreign key) ไปยังแต่ละมิติที่ทำงานร่วมกันและสิ่งที่บันทึกเป็นตัวเลขที่ใช้ในการทำงานร่วมกัน ดังแสดงตัวอย่าง star schema ตามรูปที่ 3



รูปที่ 2-3 แสดง Star Schema

Star Schema ไม่มีความชัดเจนในการเตรียมให้สนับสนุนสำหรับแอทริบิวส์แบบหลายชั้น แต่Snowflake schema เป็นการปรับปรุงให้มีโครงสร้างลำดับชั้นของมิติมีความชัดเจนโดยการทำนอมอลไลซ์ ตารางมิติดังแสดงตามรูปที่ 2-4 ซึ่งวิธีการนี้เป็นประโยชน์ในการปรับปรุงตารางมิติได้ อย่างไรก็ดีถ้าโครงสร้างของตารางมิติไม่มีการทำนอมอลไลซ์โครงสร้างก็อาจจะเหมาะสมมากกว่าสำหรับการใช้แค่ค้นหาข้อมูล

Fact constellations เป็นตัวอย่างของโครงสร้างตารางมิติที่แยกตารางมิติค่าจริง (fact table) ออกเป็นหลายตาราง สำหรับตัวอย่างอาจจะมีการแยกตารางออก และตารางที่แยกออกก็อาจจะเป็นตารางจริงได้



รูปที่ 2-4 แสดง Snowflake Schema

2.5 เซิฟเวอร์สำหรับแวร์เฮาส์

ดาต้าแวร์เฮาส์ อาจจะบรรจุข้อมูลเป็นจำนวนมาก เพื่อตอบคิวรีได้อย่างมีประสิทธิภาพ ซึ่งเมื่อเราต้องการความประสิทธิภาพสูงในวิธีการเข้าถึงข้อมูลและเทคนิคการประมวลผลคิวรี ดังนั้นจึงมีหลายเรื่องที่ต้องดำเนินการ สิ่งแรกก็คือ ดาต้าแวร์เฮาส์ต้องใช้โครงสร้างที่ซ้ำซ้อนกัน เช่นการใช้อินเด็กซ์และการใช้วิว การเลือกอินเด็กซ์จะใช้สำหรับสร้าง และวิวจะใช้ในการแก้ไขปัญหาการออกแบบ ในแนวความคิดถัดไปเพื่อให้เกิดประสิทธิภาพในการใช้อินเด็กซ์และวิวที่มีอยู่เพื่อใช้ตอบคิวรี การลดความซับซ้อนของคิวรีเป็นปัญหาที่มีความสำคัญต่อมา ดังนั้น ขณะที่จะคิวรีข้อมูลการใช้อินเด็กซ์ในการสแกนหา อาจจะมีประสิทธิภาพมาก ส่วนข้อมูลที่คงที่และไม่มีมาก คิวรีอาจจะใช้วิธีสแกนแบบลำดับ จึงเป็นข้อสรุปเพื่อประสิทธิภาพในการสแกนหาข้อมูล ความจำเป็นในการทำงานแบบขนานเพื่อเป็นการลดเวลาในการประมวลผล คงจะเป็นไปไม่ได้ เพราะจำเป็นอย่างยิ่งที่จะต้องทำการวิเคราะห์ในแต่ละส่วน

2.5.1 โครงสร้างอินเด็กซ์และการใช้งาน

จำนวนของเทคนิคในการประมวลผลคิวรีนั้นควรจะใช้ประโยชน์จากอินเด็กซ์ สำหรับตัวอย่าง เช่น การเลือกจากหลายเงื่อนไข สามารถใช้อินเด็กซ์ทำการอินเตอร์เซค (Intersection) นอกจากนี้ยังสามารถใช้อินเด็กซ์ทำการยูเนี่ยน (Union) ซึ่งอินเด็กซ์เหล่านี้เป็นตัวที่ใช้ในการเข้าถึงข้อมูลหลัก

เซิฟเวอร์สำหรับแวร์เฮาส์ สามารถใช้อินเด็กซ์แบบ bit map ซึ่งจะช่วยให้การทำยูเนี่ยนและอินเตอร์เซ็ค มีประสิทธิภาพเพิ่มขึ้น

เซิฟเวอร์สำหรับดาต้าแวร์เฮาส์สามารถใช้บิทแมปอินเด็กซ์ ซึ่งสนับสนุนการทำงานประเภทยูเนี่ยน หรืออินเตอร์เซ็คชั่นของอินเด็กซ์ได้อย่างมีประสิทธิภาพ การพิจารณาแต่ละหน้าในโครงสร้างอินเด็กซ์ที่เป็นขอบเขตของค่า d ซึ่งหน้านั้นจะบรรจุรายการของรหัสเรคคอร์ด (RIDs) ที่เป็นเรคคอร์ดที่บรรจุค่า d แต่ว่าบิทแมปอินเด็กซ์จะถูกใช้เป็นทางเลือกในการนำเสนอค่าที่สัมพันธ์กับรายการ RID ที่เป็นบิทเวคเตอร์ ซึ่งแต่ละบิทของเรคคอร์ด ซึ่งเป็นการกำหนดไว้ในขอบเขตของค่าที่เรคคอร์ดเป็น d ในแนวคิดนี้ บิทแมปอินเด็กซ์จะไม่ใช่โครงสร้างอินเด็กซ์ใหม่ แต่เป็นทางเลือกในการนำมาใช้งานของรายการ RID การที่นิยมใช้บิทแมปอินเด็กซ์เพื่อให้ได้ค่าจริงในการแทนค่าของรายการ RID สามารถเพิ่มความเร็วของการทำงาน Intersection, Union, Join และ Aggregation สำหรับตัวอย่าง ถ้า เรามีคิวรีของเงื่อนไข column1 = d และ column2 = d' ดังนั้นเราจะเจาะจงรายการโดยใช้ AND ของสองบิทเวคเตอร์ ขณะที่เราแทนค่านั้นจะมีประสิทธิภาพสำหรับลดปัญหาการขัดแย้งของขอบเขต เช่น "gender" ซึ่งจะมีประสิทธิภาพสูงสำหรับขอบเขตของข้อมูลที่มีการบีบอัดไว้ (เพราะเวลารันระบบจะต้องถอดรหัสเสมอ) บิทแมบอินเด็กซ์นั้นแรกๆ จะถูกใช้ในโมเดล 204 แต่ผลิตภัณฑ์เกี่ยวกับฐานข้อมูลจำนวนมากพึ่งสนับสนุนการทำงานเร็วๆ นี้ เช่น Sybase IQ คำถามที่น่าสนใจคือ จะเลือกแอตทริบิวท์ใดมาทำอินเด็กซ์ เพราะโดยทั่วไปการจะเลือกแอตทริบิวท์ใดต้องขึ้นอยู่กับการออกแบบฐานข้อมูล

ในการเพิ่มอินเด็กซ์ในตาราง ที่ทำสำหรับ star schema เพื่อทำการจอยน์ อินเด็กซ์โดยเฉพาะสำหรับระบบสนับสนุนการตัดสินใจ ขณะที่การแมปอินเด็กซ์ค่าในแถวนั้น จะต้องทำการจอยน์อินเด็กซ์ระหว่างฟอร์เรนจ์คีย์กับไพร์มารีย์คีย์ ซึ่งใน star schema จะมีการจอยน์อินเด็กซ์ที่สามารถกำหนดความสัมพันธ์ของค่าหนึ่งหรือหลายแอตทริบิวท์ของตารางมิติที่ทำการแมทซ์แถวกับตารางค่าจริง ยกตัวอย่างจากรูปที่ 3 การทำการจอยน์อินเด็กซ์ "Area" สำหรับแต่ละ area ที่อยู่ใน RIDs ของแถวในตารางค่าจริงนั้นจะสอดคล้องกับยอดขายของ area ดังนั้นการจอยน์อินเด็กซ์จะมีการคำนวณล่วงหน้าแบบ binary join สำหรับการจอยน์อินเด็กซ์ที่เป็นมัลติคีย์ (Mulitkey) สามารถทำการคำนวณล่วงหน้าโดยการจอยน์แบบ n-way ตัวอย่างเช่น ยอดขายทั้งหมดในฐานข้อมูลสามารถทำการจอยน์อินเด็กซ์จาก (ประเภทกรมธรรม์, รหัสบริษัท) ในตารางค่าจริง เช่นการกำหนดค่า ('1', 'AIA') จุดที่ RIDs ของแถวทั้งหมดในตารางยอดขาย จะถูกรวมกันก่อน การใช้วิธีการดังกล่าว การจอยน์อินเด็กซ์ของมุมมองหลายมิตินี้สามารถทำการเตรียมไว้ในบางเวลาก่อนได้ เพราะจะเป็นการประหยัดเวลาในการทำ intersection ในการแยกอินเด็กซ์ของประเภทกรมธรรม์และรหัสบริษัทประกันภัย ได้ ดังนั้นการจอยน์อินเด็กซ์สามารถใช้สำหรับรายการ RID เพื่อเพิ่มประสิทธิภาพในการประมวลผลของการจอยน์ได้

ฐานข้อมูลของระบบสนับสนุนการตัดสินใจควรจะมีสิ่งที่สามารถสรุปรายละเอียดเกี่ยวกับตัวอักษรและอินเด็กซ์นี้เองจะสนับสนุนในการค้นหาข้อความได้อย่างดี

2.5.2 Materialized View และการใช้งาน

คิวรีจำนวนมากบนดาต้าแวร์เฮาส์ต้องการข้อมูลแบบสรุป และ การใช้ aggregate ซึ่งการทำอินเด็กซ์เพิ่มก็จะช่วยลดเวลาในการสรุปข้อมูลของคิวรีทั่วไปได้ ตัวอย่าง เช่นเรื่องของ สินไหมค่าทดแทน จะมีการทำคิวรีที่ต้องใช้ทรัพยากรมากในการทำงานเพื่อจะให้ได้คำตอบสำหรับรายไตรมาสหรือรายเดือน ดังนั้นการที่ทำสรุปข้อมูลไว้จะเป็นการช่วยเพิ่มความเร็วในการประมวลผลได้

ในความคิดเกี่ยวกับการใช้งาน materialized view นี้จะไม่เหมือนกับการใช้งานอินเด็กซ์ คือ (1) การระบุไปว่าวิวเป็น materialize (2) เป็นการใช้งาน materialized view เพื่อเป็นคำตอบของคิวรี (3) เกิด ประสิทธิภาพในระหว่างการโหลดและรีเฟรชก็จะทำการปรับปรุง materialized view สำหรับปัจจุบันการนำแนวคิดนี้มาใช้งานยังมีปัญหา เมื่อพิจารณาความสัมพันธ์ของโครงสร้าง ดังเช่น view ของการจอยน์ในตารางค่าจริง ซึ่งเป็นซับเซ็ตของตารางมิติ (อาจจะเป็นไปได้หลังจากที่มีการเลือกมิติเหล่านี้) กับฟังก์ชั่นในการวัดค่าโดยการ group by ค่าของแอตทริบิวท์ จากตารางมิติ เนื่องจากโครงสร้างของวิวเหล่านี้จะมีขนาดที่เล็กที่ซับซ้อนมาก (ดูจากโครงสร้างแบบ snowflake)

แม้ว่าจะมีการรูปแบบวิวที่เป็น materialize จะเป็นทางเลือกในการใช้งาน แต่โดยลักษณะของการทำงานจะเป็นการเพิ่มการทำงานด้าน update และต้องใช้พื้นที่ในการจัดเก็บมากขึ้น จากข้อสมมติฐานอย่างง่ายด้วย greedy algorithm แสดงให้เห็นว่าประสิทธิภาพดี จากปัญหาที่เกี่ยวข้องภายใต้การปรับปรุงประสิทธิภาพก็อาจจะเลือก materialized view เพื่อมาใช้งานสำหรับการหา aggregation ตอนสร้างความสัมพันธ์

ถึงแม้การใช้งานจะง่าย แต่ประโยชน์ในการใช้ materialized view ก็มีมากในการ select ข้อมูล หรือการ rollup โดยการ group และการเพิ่มฟังก์ชั่น ตัวอย่างเช่น สมมติว่า materialized view ประกอบด้วยผลรวมยอดขายสำหรับแต่ละไตรมาส ในแต่ละรหัสบริษัทประกันภัย ซึ่ง materialized view นี้สามารถใช้ตอบคิวรีที่ถามเกี่ยวกับผลรวมยอดขายของ “AIA” สำหรับปีโดย ถ้าจะแสดงเป็นปีจะต้อง rolling up จากยอดรวมไตรมาสเป็นยอดรวมปี

โดยทั่วไป อาจจะมีหลายทางวิธีในการใช้งาน materialized view เพื่อเป็นคำตอบคิวรี ตัวอย่างเช่น วิว V เป็นเซ็ตของมุมมองมิติของ Q ถ้ามีการเลือกบางส่วนใน Q แล้วปรากฏว่าส่วนที่เลือกนั้นอยู่ใน V และถ้าทำการ group-by คอลัมภ์ใน V เพื่อเป็นซับเซ็ตของ group-by คอลัมภ์ใน Q ก็จะเห็นได้ว่าวิว V สามารถทำงานแทน Q ที่เกิดใหม่ได้ โดยกำหนดให้เป็นเซ็ต materialized view

M และคิวรี Q เราสามารถกำหนดได้ว่า M' เป็นเซ็ตที่น้อยสุดสำหรับ Q ตัวอย่างการใช้งานเช่น การถามผลรวมยอดขายกรมธรรม์ประเภท 3 โดยมีวิวที่ใช้งาน 2 วิวในการสร้าง คือ (1) ผลรวมยอดขายแต่ละจังหวัดสำหรับแต่ละประเภทกรมธรรม์ และ (2) ผลรวมที่ขายแต่ละภาคสำหรับแต่ละบริษัทประกันภัย ด้วยวิธีการแบบนี้จะเป็นการทำให้การค้นหาแคบลงและ materialized view เหมาะสมในการใช้งาน ขณะที่เราทำให้เป็นคำตอบที่น้อยที่สุดของคิวรี แต่ปัญหาโดยทั่วไปคือการทำ multi materialized view ก็จะมีความซับซ้อนมากขึ้น ในกรณีที่ทำ Select-Project-Join

2.5.3 การประมวลผลแบบขนาน

การประมวลผลแบบขนานสำหรับฐานข้อมูลที่มีขนาดใหญ่กับข้อมูล Teradata เป็นเรื่องหนึ่งของเทคโนโลยี โดยผู้จำหน่ายระบบจัดการฐานข้อมูลทั้งหมด ขณะนี้ได้เสนอเทคโนโลยีในการจัดการข้อมูลแบบแบ่งส่วน (partitioning) และการประมวลผลแบบขนาด (parallel) โดยเทคนิคที่เกี่ยวข้องตัวหนึ่งที่น่าสนใจคือ การใช้ทรัพยากรของระบบสนับสนุนการตัดสินใจแบบอ่านอย่างเดียว (read-only) ที่เป็นลักษณะของสแกนแบบ piggybacking โดยจะทำงานลักษณะนี้เมื่อถูกสั่งงานแบบมัลติเพิลคิวรี (ใช้ใน Redbrick)

การสแกนแบบ piggybacking จะลดเวลาในการทำงานทั้งหมดได้เป็นอย่างดี โดยจะทำงานสำหรับคำสั่งที่มาพร้อมๆ กันแบบ overlapping

2.5.4 สถาปัตยกรรมของเซิฟเวอร์สำหรับการประมวลผลคิวรี

เซิฟเวอร์แบบรีเรชั่นนอลแบบเดิมไม่ค่อยมีความฉลาดในการใช้งานอินเด็กซ์และความต้องการอื่นๆ ที่สนับสนุนวิวและข้อมูลแบบหลายมิติ โดยผู้จำหน่าย RDBMS ทั้งหมดต่างก็พยายามให้ระบบสามารถสนับสนุนได้อย่างรวดเร็วโดยการเพิ่มตามความต้องการ ดังนั้นสำหรับผู้พัฒนาระบบต้องพิจารณาสามเรื่องที่ต้องมีในระบบสนับสนุนการตัดสินใจ

- Specialized SQL Servers : Redbrick เป็นตัวอย่างของเซิฟเวอร์ที่มีวัตถุประสงค์เตรียมไว้สำหรับภาษาคิวรีชั้นสูงและสนับสนุนการประมวลผลคิวรีบน star และ snowflake schema ภายใต้การทำงานแบบ read-only
- ROLAP Servers : มีการจัดเก็บที่อยู่ของข้อมูลระหว่าง relational back end server (จะบันทึกที่อยู่ของข้อมูลที่อยู่ในแวร์เฮาส์) กับ client front end tools สำหรับ Microstrategy เป็นตัวอย่างของเซิฟเวอร์แบบนี้ ซึ่งจะไปเพิ่มความสามารถของระบบจัดการฐานข้อมูลรุ่นเก่า โดยระบบจะทำงานเป็นตัวกลาง (Middleware) ที่จะสนับสนุนให้คิวรีแบบ OLAP ทำงานได้อย่างมีประสิทธิภาพ และการทำงานเหมาะสมกับ back

end relational servers โดยจะทำการสร้างวิวให้เป็น materialized และทำการเปลี่ยนรูปแบบของการใช้คิวรีให้เหมาะสมกับ materialize view แล้วสร้าง multi-statement SQL สำหรับ back end server ดังนั้นการเตรียมเซอร์วิส ที่เพิ่มนั้นจะต้องทำการจัดตารางของคิวรีและการกำหนดให้ใช้ทรัพยากรให้เหมาะสม ซึ่งแนวโน้มจะมีการเปลี่ยนมาใช้ ROLAP มาก และด้วยจุดเด่นของเซิฟเวอร์ ROLAP นี้จะสามารถขยายระบบและคุณสมบัติการทำทรานเซคชั่นของระบบรีเลชั่นนอลได้ อย่างไรก็ดี การทำงานระหว่างคิวรีในรูปแบบ OLAP และ SQL นั้นก็ยังไม่ดี เช่น ลำดับในการประมวลผลยังเป็นแบบลำดับ (sequential) หรือการใช้ aggregate ยังคงทำกับคอลัมภ์ จึงเป็นสาเหตุทำให้เกิดบัททอลเน็คของเซิฟเวอร์ OLAP

- MOLAP Servers : เซิฟเวอร์ประเภทนี้จะสนับสนุนข้อมูลและวิวแบบมุมมองหลายมิติโดยตรง เนื่องจากมีเอนจิ้นสำหรับเก็บข้อมูลแบบหลายมิติ ความเห็นไปได้ในการนำมาใช้คิวรีด้วย front-end ในมุมมองหลายมิติบนเลเยอร์ของที่เก็บข้อมูลจะสามารถทำการแมปได้เลย ตัวอย่างของเซิฟเวอร์นี้คือ Essbase ดังเช่นการใช้ประโยชน์จากคุณสมบัติของอินเด็กซ์จะทำได้ดีมาก แต่การเตรียมพื้นที่จัดเก็บข้อมูลจะค่อนข้างไม่มีประสิทธิภาพ โดยเฉพาะอย่างยิ่งเมื่อชุดข้อมูลอยู่กระจาย สำหรับเซิฟเวอร์ MOLAP หลายผลิตภัณฑ์สามารถนำมาใช้งานเป็น 2-Level โดยจะทำการปรับการกระจายของข้อมูลให้อยู่รวมกันอย่างต่อเนื่อง ซึ่งในพื้นที่เก็บข้อมูลของ 2-Level นี้จะถูกทำให้เป็นซับอาเรย์ หนึ่งหรือสองมิติ ซึ่งจะทำให้ข้อมูลอยู่รวมกันต่อเนื่อง แต่ว่าการใช้งานของเครื่องมือออกแบบหรือการที่ผู้ใช้งานกำหนดเอง ให้อยู่ในรูปแบบอาเรย์ ซึ่งจะทำอินเด็กซ์ให้เอง มีหลายเทคนิคที่เป็นกลไกของระบบฐานข้อมูลทั่วไปได้นำมาใช้งานในเซิฟเวอร์ MOLAP

2.5.5 คำสั่ง SQL เพิ่มเติม

มีหลายคำสั่งของ SQL ที่เพิ่มเข้ามาซึ่งง่ายต่อการใช้งานและมีวัตถุประสงค์เพื่อประมวลผลคิวรีของ OLAP หรือการนำมาใช้เพิ่มเติมในเซิฟเวอร์แบบรีเลชั่นนอล อธิบายบางส่วนของคำสั่งที่เพิ่มดังนี้

- กลุ่มฟังก์ชั่น aggregate กลุ่มนี้จะรวมความสามารถในการทำการเรียงลำดับ (rank) และการแจกแจงแบบ percentile เช่น all Product in Top 10 Percentile หรือ Top 10 Products by total Sale และยังสนับสนุนฟังก์ชั่นที่รู้จักกันดีและใช้งานเกี่ยวกับการเงิน เช่น mean, mode, mediam

- คุณสมบัติการทำรายงาน การสร้างรายงานสำหรับการวิเคราะห์ข้อมูลเชิงธุรกิจ ความความต้องการคุณสมบัติ aggregate ที่นำมาใช้เป็นลักษณะ time window เช่น แสดงการเคลื่อนที่ของค่าเฉลี่ย ซึ่งคุณสมบัติที่เพิ่มนี้มีความสำคัญที่จะกำหนดจุดหยุด (breakpoint) และทำการหาผลรวมที่เกิดขึ้น โดยคุณสมบัติ SQL นี้มีใน Redbrick
- Multiple Group-By เครื่องมือที่เป็น front end เช่น กระดาษทำการมุมมองหลายมิติต้องการที่จะทำการ group by ค่าของแอตทริบิวท์ที่แตกต่างกัน ด้วยเงื่อนไขนี้จะสามารถทำการทดลองกำหนดให้คำสั่งของ SQL ทำการสแกนว่ามีข้อมูลใดบ้างที่สามารถอยู่ในรูปแบบของ multiple times แต่วิธีการนี้ไม่มีประสิทธิภาพ ในปัจจุบันมี 2 คำสั่งใหม่ที่จะทำ ก็คือ Rollup และ Cube ซึ่งมีวัตถุประสงค์เพื่อทำการเชื่อม SQL การทำ Rollup ของรายการแอตทริบิวท์ (รหัสบริษัท, เดือน, ประเภทกรมธรรม์) แล้วข้อมูลที่เป็นผลลัพธ์ก็จะแสดงคำตอบตามรูปแบบ Group by คือ (a) group by (รหัสบริษัท, เดือน, ประเภทกรมธรรม์), (b) group by (รหัสบริษัท, เดือน), (c) group by (รหัสบริษัท) จากรายการจำนวนทั้งหมดจะแสดงจำนวนเป็น k คอลัมภ์ ส่วนการทำ Cube เป็นการเตรียม group-by สำหรับแต่ตัวที่เป็นตัวประกอบ (Combination) การทำงาน 2K ของคอลัมภ์ ดังนั้นคำสั่ง group-by แบบหลายจำนวนสามารถประมวลผลได้มีประสิทธิภาพมากกว่า โดยการตรวจสอบการทำงานระหว่างค่าของข้อมูลด้วยกันเอง ซึ่ง Microsoft SQL Server สนับสนุนการทำงานของทั้ง Cube และ Rollup
- การเปรียบเทียบ เนื่องจากความไม่สมบูรณ์ของ SQL จึงทำให้เกิดการเปรียบเทียบ ซึ่งเป็นเรื่องปกติในทางธรุกิจ ตัวอย่างเช่นการเปรียบเทียบระหว่างผลรวมยอดขายทั้งหมดและผลรวมยอดขายแต่ละไตรมาส ซึ่งปัจจุบันยังต้องทำการวิจัยเพื่อให้สนับสนุนการเปรียบเทียบ

2.5.6 เมตะดาต้าและการจัดการแวร์เฮาส์

เนื่องจากดาต้าแวร์เฮาส์เป็นการพัฒนาให้เป็นโมเดลทางธุรกิจของกิจการ โดยปัจจัยที่มีความสำคัญสำหรับสถาปัตยกรรมการทำแวร์เฮาส์ก็คือ การจัดการเมตะดาต้า มีหลายวิธีที่แตกต่างกันในการจัดการเมตะดาต้า

- Administrative Metadata ประกอบด้วย สารสนเทศที่จำเป็นสำหรับการตั้งค่าและการใช้แวร์เฮาส์ ซึ่งประกอบด้วย รายละเอียดของที่มาของฐานข้อมูล เครื่องมือที่เป็น front-end, back-end การให้คำจำกัดความของ warehouse schema มิติและลำดับชั้นของ

ข้อมูล การกำหนดคิวรีและรายงาน ที่จัดเก็บดาต้ามาร์ท นโยบายการรีเฟรชและการล้างดาต้าแวร์เฮาส์ และสิทธิ์การใช้งานระบบต่างๆ รวมทั้ง Log ของผู้ใช้ระบบ

- Business Metadata ประกอบด้วย คำจำกัดความและความหมายเกี่ยวกับธุรกิจ ผู้เป็นเจ้าของข้อมูล และนโยบายการเปลี่ยนแปลง
- Operational Metadata ประกอบด้วยสารสนเทศที่เกิดขึ้นระหว่างการทำงานของดาต้าแวร์เฮาส์ ได้แก่ การย้ายข้อมูล และการเปลี่ยนรูปของข้อมูล ข้อมูลปัจจุบันในแวร์เฮาส์ (สถานะ คงอยู่ หรือทำการลบ) การข้อมูลสำหรับการเฝ้าติดตาม เช่นข้อมูลสถิติ รายงานความผิดพลาด และการตรวจสอบ

การสร้างและการจัดการระบบแวร์เฮาส์เป็นเรื่องที่ยาก เนื่องด้วยความแตกต่างของเครื่องมือที่ใช้งานมีมาก ในการพัฒนาเครื่องมือที่จะใช้ออกแบบและแก้ไขสคีมา วิว สคริปต์ กฎ คิวรี และรายงาน จะต้องมีการวางแผนและวิเคราะห์เครื่องมือที่จะใช้ว่าสำหรับเรื่องต่างๆ ว่าอะไรใช้ทำการแก้ไขสคีมา หรืออัตราการรีเฟรช การวางแผนความจุ เครื่องมือใจการจัดการแวร์เฮาส์จะถูกใช้ในการเฝ้าติดตามแวร์เฮาส์ การทำรายงานสถิติ และให้คำแนะนำผู้ดูแลระบบ โดยใช้แยกและตารางสรุป เวลาในการประมวลผลคิวรี ชนิดความถี่ในการ drill downs หรือ rollup เพื่อรู้ว่าผู้ใช้งานต้องการอะไร จุดสูงสุดและค่าเฉลี่ยของเวลาการทำงาน การทำรายงานเพิ่มเติม ค้นหาคิวรีที่กำลังทำงานอยู่ และตัววัดคุณภาพของบริการ สำหรับเครื่องมือในการจัดการระบบและเครือข่าย จะถูกใช้วัดความคับคลั่งระหว่างไคล์แอนท์ และเซิฟเวอร์ ระหว่างเซิฟเวอร์แวร์เฮาส์และระบบฐานข้อมูล เพื่อใช้ในการพิจารณาสำหรับการจัดการเรื่องของการประมวลผลเช่น โหลด รีเฟรช ในขั้นตอนของการประมวล ก็จะทำให้เกิดความเหมาะสม และสามารถทำงานในลักษณะกำหนดเวลาทำงานหรือทำงานเมื่อต้องการ ซึ่งระบบจะแสดงให้เห็นลำดับของการทำงานและสถานะของงานว่าทำสำเร็จหรือ เกิดความผิดพลาด ก็จะสามารถกู้สิ่งที่ดำเนินการด้วยการ roll back

บทที่ 3
โมเดลสำหรับฐานข้อมูลแบบมุมมองหลายมิติ

ระบบฐานข้อมูลแบบมุมมองหลายมิติ เป็นระบบที่สามารถทำการหมุนเพื่อดูมุมมองข้อมูลแบบต่างๆ ซึ่งใช้สร้างแอพพลิเคชั่นสำหรับระบบสนับสนุนการตัดสินใจ โดยความต้องการของข้อมูลจะเป็นลักษณะเชิงสรุป เป็นการรวมรวมข้อมูล การสร้างมุมมอง หรือการประยุกต์ใช้สูตร กับข้อมูลที่มีหลายมิติ ซึ่งประกอบด้วย ดาต้าโมเดลที่เป็น hypercube ซึ่งเป็นชุดคำสั่งพื้นฐานในการออกแบบวิธีที่ใช้กันอยู่ในปัจจุบัน และฟังก์ชั่นที่เพิ่มเติม โดยวัตถุประสงค์ของโมเดลที่ทำให้เกิดความสมดุลสำหรับทุกมิติของข้อมูลรวมถึงการวัดค่า ด้วยโมเดลที่ช่วยให้ยืดหยุ่นในการเตรียมให้สามารถทำงานได้ในแต่ละมิติและสนับสนุน ad hoc aggregate

3.1 โมเดลข้อมูล (Data Model)

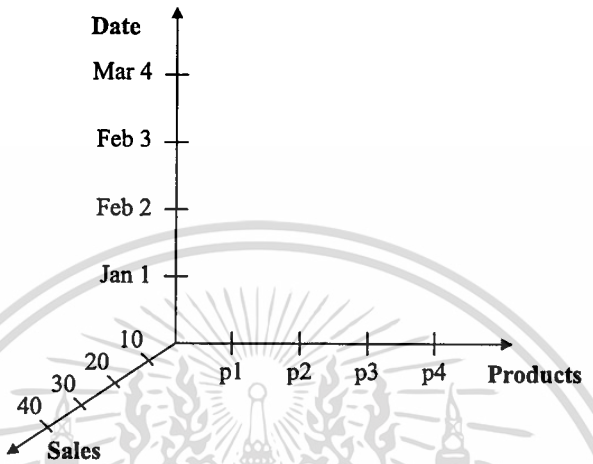
โมเดลข้อมูลนี้ เป็นรูปแบบที่สนับสนุนสำหรับข้อมูลที่มีมุมมองหลายมิติ และคำสั่งที่มีใช้อยู่ในระบบฐานข้อมูลแบบมุมมองหลายมิติทั่วไป สำหรับหลักการของโมเดลนี้จะอธิบายวิธีการทำมิติและการวัดค่าที่ได้แต่ละสัดส่วน และทำให้การทำงานมีความยืดหยุ่น ทั้งแบบหลายลำดับชั้น การทำ ad hoc aggregate โดย logical model ของข้อมูลจะมีโครงสร้างเป็นหนึ่งหรือหลายคูป โดยแต่ละคูปจะประกอบด้วย

- k มิติ และแต่ละมิติจะมีชื่อ $\mathcal{D}_i$ อยู่ในโดเมน dom_i แล้วจะมีค่าปรากฏ
- ค่าที่กำหนดให้เป็นคู่ $\mathcal{E}(C)$ จาก $dom_1 \times \ldots \times dom_k$ ไปจนถึง n-tuple (n แถว) ถ้า $\mathcal{E}(C)(d_1, \ldots, d_k)$ จะเป็นการอ้างถึงค่า ณ ตำแหน่งที่ $d_1, \ldots, d_k$ ของ cube C
- ส่วนของเมตะดาต้าที่เป็น n-tuple ของชื่อ โดยในแต่ละชื่อจะอธิบายเป็นหนึ่งในสมาชิกของค่า n-tuple ที่อยู่ใน cube

แต่ละค่าของคูปสามารถเป็น 0, 1 หรือ n-tuple $<X_1,\ldots, X_n>$ ถ้าค่าตรงกันกับ $\mathcal{E}(C)(d_1,\ldots,d_k)$ จะเป็น 0 แสดงว่าค่านั้นไม่มีอยู่ในฐานข้อมูล ถ้าเป็น 1 แสดงว่าค่านั้นตรงกัน และการแสดง n-tuple นั้นเกิดจากการเพิ่มค่าที่เป็นไปได้สำหรับค่าที่ตรงกันในแต่ละมิติ ถ้าบางค่าของคูปเป็น 1 แล้วจะสามารถมีค่าที่จะเป็น n-tuple เราจะแทนค่าเหล่านั้นให้ครบมิติของคูป ซึ่งจะมีอย่างน้อยหนึ่งค่าของคูปที่ไม่ใช้ 0 แต่ถ้าค่าทั้งหมดของ cube เป็น 0 แสดงว่าเป็นคูปว่าง หรือในกรณีถ้าโดเมน dom_i ของมิติ $\mathcal{D}_i$ ไม่มีค่าเลยคูปก็ว่างเช่นกัน

3.2 โอเปอร์เรเตอร์ (Operators)

ในการแสดงมุมมองข้อมูลจะแสดงเป็น 3 มิติ เพื่อให้เห็นมุมมองของมิติ จากรูปที่ 3-1 จะแสดง สินค้า, วันที่, ยอดขาย ซึ่ง ยอดขายไม่ได้ใช้วัดค่า แต่เป็นมิติสำหรับโมเดลนี้



รูปที่ 3-1 แสดง Logical cube ที่มียอดขายเป็นมิติ

โอเปอร์เรเตอร์บน logical cube มิติของยอดขายอาจจะไม่แสดง ซึ่งอยู่ในคูปดังนั้นจำนวนของยอดขายสามารถวัดค่าได้โดยมิติของสินค้า และวันที่ ซึ่งจะอธิบายภายหลังว่าดึงมาแสดงได้อย่างไร สำหรับตอนนี้จะใช้คูปด้วยจำนวนยอดขายที่มีสมาชิกค่าของ cube ดังนั้นค่า <15> สำหรับ "Date = Mar 4" และ "Product = p1" ในรูปที่ 3-2 ที่แสดงนั้นใน logical cube รูปที่ 2 จะเป็นค่าที่ตรงกันกับ "Date = Mar 4", "Product = p1" และ "Sales = 15" ซึ่งมีค่าเป็น "1" ซึ่งจะแสดงเมตะดาต้าของแต่ละค่าที่เป็นคำอธิบายในคูป ดังนั้น <Sales> ในรูปที่ 3-2 จะแสดงแต่ละค่าในคูปที่เป็นค่าของยอดขาย

หมายเหตุ สิ่งที่กำหนดเป็นโอเปอร์เรเตอร์ในการใช้ cube C ด้วย k มิติ โดยอ้างถึงมิติที่เป็น $\mathcal{D}_1,\ldots,\mathcal{D}_k$ ในการใช้ $\mathcal{D}_i$ เพื่ออ้างถึงโดเมนของมิติ $\mathcal{D}_i$ และจะใช้ตัวอักษรเป็นตัวเล็ก a,b,c เพื่ออ้างถึงค่าคงที่

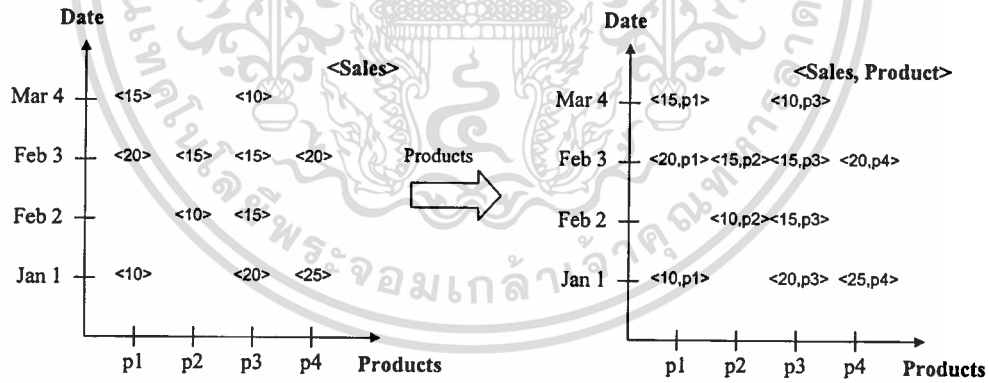
ค่าของมิติในฟังก์ชั่นโมเดลข้อมูลนี้จะมีอ้างค่าของคูป เป็นผลของโปรแกรมที่เกิดจากโอเปอร์เรชั่น อาจจะมีการแมปค่าแบบบหนึ่งต่อหนึ่งหรือหลายค่า ที่เป็นคำตอบของคูป ค่าเหล่านี้จะเป็นการรวมเข้าด้วยกันเป็นหนึ่งค่าด้วยฟังก์ชั่นใด ก็จะเรียกว่า element combining function โดยแทนค่าเป็น f_{elem}

บางครั้งการเมิร์จ (merge) ของค่าตลอดชุดมิติ จะเรียกว่า dimension merging function โดยแทนด้วย f_{merge}

3.2.1 Push

Push เป็นโอเปอร์เรเตอร์ (ดูจากรูป 3-2) เป็นการคอนเวิรท์ค่าในมิติโดยการสลับตำแหน่ง โดยใช้ฟังก์ชั่น f_{elem} ซึ่งโอเปอร์เรเตอร์นี้จะยอมให้มิติและการวัดค่าเป็นรูปแบบเฉพาะ

- Input : C, $\mathcal{D}_i$
- Output : C ด้วยค่าที่ไม่เป็น 0 และถูกขยายโดยสมาชิกของค่าที่เพิ่ม ซึ่งเป็นค่ามิติของ $\mathcal{D}_i$ สำหรับแต่ละค่า ถ้าค่าไม่ใช่ n-tuple แต่เป็น "1" เมื่อทำการคอนเวิรท์ไปเป็น 1-tuple ก็จะบรรจุค่าที่ตรงกันของมิติ
- Mathematically : push(C, $\mathcal{D}_i$) = C_{ans}.
  $\mathcal{E}(C_{ans})(d_1,\ldots,d_k) = g \oplus \langle d_i \rangle$ เมื่อ $g = \mathcal{E}(C)(d_1,\ldots,d_k)$ โดยเครื่องหมาย $\oplus$ กำหนดให้เป็น 0 ถ้า $g = 0$ และจะมีค่าเป็น $\langle d_i \rangle$ ถ้า $g = 1$ และทุกกรณีจะทำต่อเนื่องกัน 2 แถว



รูปที่ 3-2 แสดงโอเปอร์เรชั่น push ในมิติของสินค้า

3.2.2 Pull

Pull เป็นโอเปอร์เรเตอร์ที่ใช้ย้อนค่ากลับจากผลที่เกิดจากการทำโอเปอร์เรเตอร์ push โอเปอร์เรเตอร์ Pull จะทำการสร้างมิติใหม่สำหรับแยกสมาชิกในแต่ละค่า โดยการทำงานจะคอนเวิรท์ค่าที่อยู่ในมิติ ดังนั้นค่าสามารถทำการเมิร์จหรือทำการจอยน์ โอเปอร์เรเตอร์นี้จะใช้ได้ก็ต่อเมื่อมิติมีขนาดเท่ากัน (symmetric)

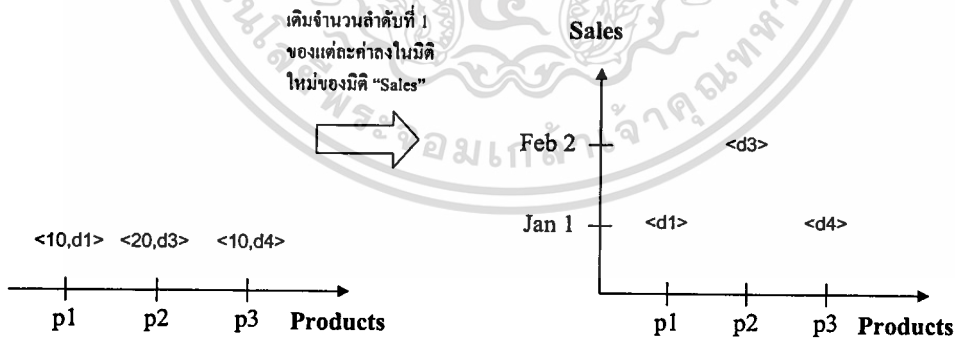
- Input : C, ชื่อใหม่ของมิติ $\mathcal{D}$, Integer i
- Output : C_{ans} จะเป็นมิติ $\mathcal{D}$ ที่ถูกเพิ่ม โดยจะดึงค่าในตำแหน่ง i ของแต่ละของเมทริกซ์
- Constraint : ค่าทั้งหมดที่ไม่เป็น 0 ของ C จะเป็น n-tuples เพราะแต่ละค่าที่ไม่เป็น 0 ก็จะมีสมาชิกอย่างน้อยหนึ่งค่าที่สามารถสร้างมิติใหม่ได้
- Mathematically : pull(C, $\mathcal{D}$, i) = C_{ans}, $1 \leq i \geq n$

  $\mathcal{D}$ จะกลายเป็นมิติที่ $k+1$ ของคูป

  $dom_{k+1}(C_{ans}) = \{ e \mid e$ เป็นสมาชิก i ของบางค่าใน $\mathcal{E}(C)(d_1,\ldots,d_k) \}$

  $\mathcal{E}(C_{ans})(d_1,\ldots,d_k) = < e_1, \ldots, e_{i-1}, e_{i+1},\ldots,e_n >$ ถ้า $\mathcal{E}(C)(d_1,\ldots,d_k) = < e_1, \ldots, e_i,\ldots,e_n >$,

  มิเช่นนั้น $\mathcal{E}(C_{ans})(d_1,\ldots,d_k) = 0$ หมายเหตุ : ถ้าค่าของผลลัพธ์ไม่มีจำนวนสมาชิกจะถูกแทนค่าด้วย 1



รูปที่ 3-3 การดึง (Pull) ค่าสมาชิกตัวแรกของแต่ละค่าให้เป็นมิติของยอดขาย

3.2.3 Destroy Dimension

บ่อยครั้งที่มีความจำเป็นต้องลดขนาดจำนวนมิติในคูป ซึ่งโอเปอร์เรเตอร์นี้จะทำการลบมิติ $\mathcal{D}$ นั้นออก ซึ่งอยู่ในโดเมนที่เป็นค่าเดี่ยว ความหมายของค่าเดี่ยวนั้นให้หมายถึงค่าที่คงเหลือจาก

มิติ k-1 ซึ่งมิติของคูป k-1 จะไม่ซ้ำกัน ถ้ามิติ $\mathcal{D}$ ถูกลบออกแล้ว ผลของมิติ k-1 ก็จะเกิดช่องว่าง และจะถูกลดขนาด

- Input : C, ชื่อใหม่ของมิติ $\mathcal{D}_i$
- Output : C_{ans} ด้วยมิติ $\mathcal{D}_i$ ที่ถูกทำลาย
- Constraint : $\mathcal{D}_i$ มีเพียงหนึ่งค่า
- Mathematically : destroy(C, $\mathcal{D}_i$)

  C_{ans} มีจำนวนมิติ k-1

ถ้ามิตินั้นมีหลายค่าจะไม่สามารถทำการลบโดยตรงได้เนื่องจากมีค่าที่เกี่ยวข้องกับฟังก์ชั่นที่ให้ค่านั้น ในกรณีที่มีหลายค่าและต้องการ destroy จต้องทำการเมิร์จโอเปอร์เรชั่น และทำคำสั่งนี้อีกครั้ง

3.2.4 Restriction

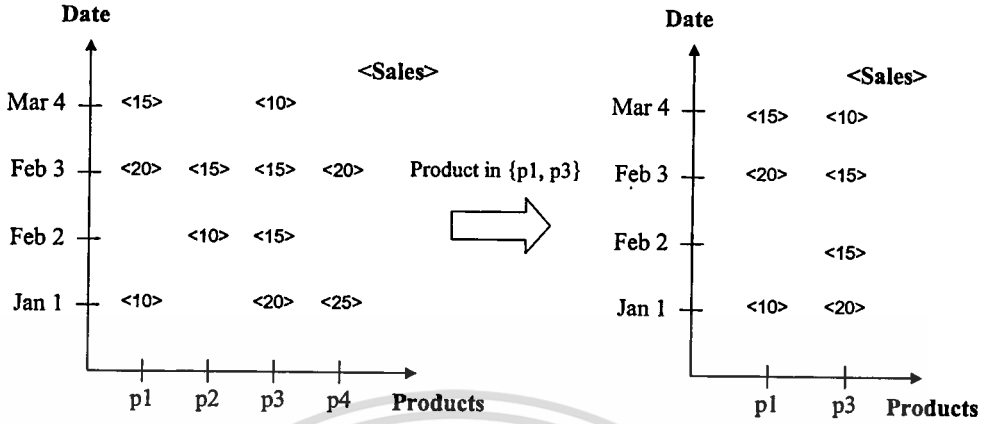
โอเปอร์เรเตอร์ Restrict เป็นการทำงานบนมิติของคูปและลบค่าในมิติของคูปที่เงื่อนไขไม่ถูกต้องออก จากรูปที่ 3-4 เป็นผลที่เกิดจากการทำ restrict หมายเหตุ : คำสั่งนี้ก็คือ slicing/dicing ในการจัดการคูปของฐานข้อมูลมุมมองหลายมิติ

- Input : คูป C และค่า $\mathcal{P}$ ที่อ้างอยู่บน $\mathcal{D}_i$
- Output : คูปใหม่ C_{ans} ที่ทำการย้ายค่าที่มีการอ้างถึง $\mathcal{P}$ ในมิติ $\mathcal{D}_i$ ออกจาก C

  หมายเหตุ $\mathcal{P}$ เป็นสิ่งที่ประกอบเป็นชุดข้อมูล และไม่ได้เป็นค่าเดี่ยว แต่ว่าถ้า $\mathcal{P}$ เป็นสิ่งที่ถูกกำหนดโดเมนของ $\mathcal{D}_i$ เช่น “top 5 values” ถ้าไม่มีค่าของมิติ $\mathcal{D}_i$ แล้ว C_{ans} ก็จะเป็นคูปว่าง
- Mathematically : restrict(C, $\mathcal{D}_i$, $\mathcal{P}$) = C_{ans}

  $dom_j(C_{ans}) = dom_j(C)$ ถ้า $1 \leq j \leq k$ & $j \neq i$ ไม่เช่นนั้น $dom_j(C_{ans}) = \mathcal{P}(dom_j(C))$

  $\mathcal{E}(C_{ans})(d_1,\ldots,d_k) = \mathcal{E}(C)(d_1,\ldots,d_k)$



รูปที่ 3-4 แสดงผลของโอเปอร์เรชั่น Restrict

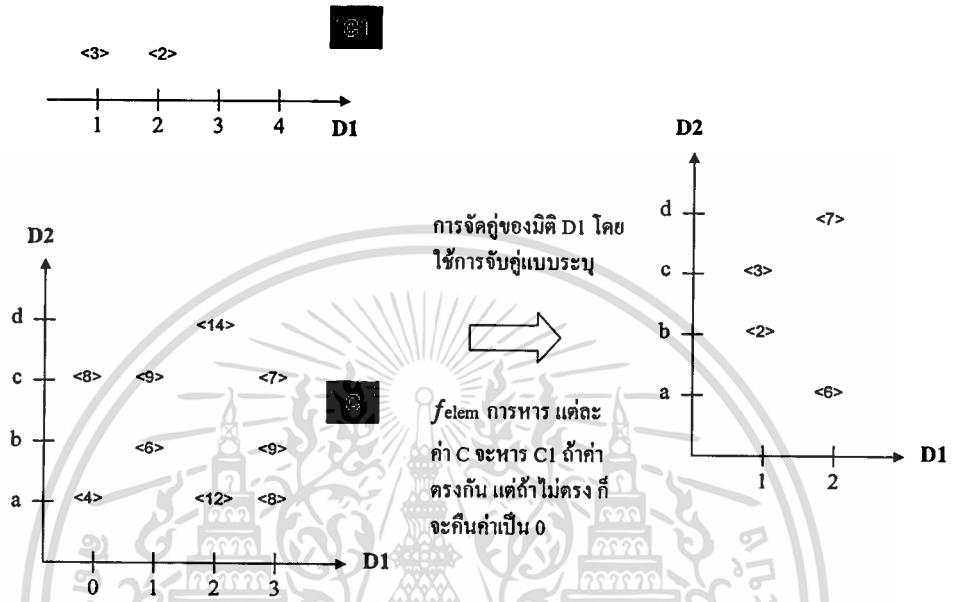
3.2.5 Join

โอเปอร์เรชั่นการจอยน์เป็นการใช้สร้างความสัมพันธ์ของสารสนเทศในสองคูป โดยผลของการจอยน์จะเป็น m มิติ เมื่อคูป C มีจำนวน n มิติ คูป $C1$ บน k มิติ จะเรียกการจอยน์มิติว่าเป็นคูป C_{ans} ด้วย $m+n-k$ มิติ ในการจอยน์แต่ละ $\mathcal{D}_i$ มิติของ C จะรวมเป็นหนึ่งมิติ $\mathcal{D}_x$ ของ $C1$ ผลลัพธ์ของมิติที่ตรงกันจะมีค่าเป็นยูเนี่ยนของค่าใน $\mathcal{D}_i$ และ $\mathcal{D}_{x_1}$ สำหรับการทำ Transformation สามารถนำมาใช้เพื่อเปลี่ยนค่าของมิติ $\mathcal{D}_i$ และ $\mathcal{D}_{x_1}$ ก่อนที่จะทำการแมปเพื่อให้ผลลัพธ์ สำหรับค่าเป็นผลลัพธ์ของคูป C_{ans} จะถูกรวมเข้าด้วยกันโดยใช้ฟังก์ชั่น f_{elem} ทุกค่าของ C และ $C1$ นั้นจะทำการแมปไปยังค่าของคูป C_{ans}

รูปที่ 3-5 ภาพแสดงการจอยน์คูป C กับคูป C_1 บนมิติของ $\mathcal{D}_1$ (ไม่มีการใช้ Transformation) มิติ $\mathcal{D}_1$ ของผลลัพธ์คูปจะมีเพียงสองค่า ฟังก์ชั่น f_{elem} ทำการหารทุกค่าจากคูป C ด้วยค่าจาก $C1$ ถ้าแต่ละค่าเป็น 0 ก็จะทำให้ผลลัพธ์ได้ 0 ด้วย ค่าของผลลัพธ์ในมิตินั้นจะมีเพียงค่า 0 เท่านั้นก็จะไม่แสดงใน C_{ans} (เหมือนค่า 0 และ 3 ในมิติ $\mathcal{D}_1$) ในการขจัดค่าจะทำตามลำดับที่อยู่ในคูปโดยตลอดแนว มิตินั้นจะต้องมีอย่างน้อยหนึ่งค่าที่ไม่ใช่ค่า 0

- Input : C กับมิติ $\mathcal{D}_1 \ldots \mathcal{D}_m$ และ $C1$ กับมิติ $\mathcal{D}_{m-k} \ldots \mathcal{D}_n$ โดยมิติ $\mathcal{D}_{m-k} \ldots \mathcal{D}_m$ เป็นมิติที่จอยน์ ด้วยฟังก์ชั่นการแมป 2k ฟังก์ชั่น $f_{m-k_1}, \ldots, f_m$ นี้ถูกประกาศค่าของมิติ $\mathcal{D}_{m-k} \ldots \mathcal{D}_m$ ของ C และ ฟังก์ชั่น $f'_{m-k_1}, \ldots, f'_m$ เป็นการประกาศบนมิติ $\mathcal{D}_{m-k} \ldots \mathcal{D}_m$ ของ $C1$ การแมป f_i จะถูกใช้กับค่า $v \in dom_i(C)$ เพื่อสร้างค่าสำหรับมิติ $\mathcal{D}_i$ ใน C_{ans} ในลักษณะเดียวกัน f'_i จะถูกใช้กับค่า $v' \in dom_i(C1)$ เพื่อสร้างค่าสำหรับ

มิติ $\mathcal{D}_i$ ใน C_{ans} ดังนั้นสิ่งที่ต้องมีสำหรับการใช้ฟังก์ชั่น $f$elem เป็นการรวมค่าที่มาจาก Cและ $C1$ เพื่อให้ได้ผลลัพธ์เป็นค่าของ C_{ans}



รูปที่ 3-5 แสดงการจอยน์ค่าสองคูป

- Output : C_{ans} กับ n มิติ โดย $\mathcal{D}_1,\ldots, \mathcal{D}_n$ ค่าต่างๆ ใน Cและ $C1$ สามารถที่จะทำการแมปกับ C_{ans} ที่เป็นผลรวมของฟังก์ชั่น $f$elem ที่สร้างค่าผลลัพธ์ของ C_{ans} ถ้ามีค่า v บางค่าของมิติ $\mathcal{D}_i$ ค่า $\mathcal{E}(C_{ans})(x_1,\ldots, v, x_{i+1} \ldots, x_n)$ เป็น 0 สำหรับทุกค่าของมิติอื่นๆด้วย ดังนั้น v จึงไม่ควรรวมอยู่ในมิติ $\mathcal{D}_i$ ของ C_{ans}
- Mathematically : join(C, $C1$, [$f_{m-k},\ldots, f_m, f'_{m-k},\ldots, f'_m$], $f$elem) = C_{ans}

$dom_i(C_{ans}) = dom_i(C)$ ถ้า $1 \leq i \leq m-k-1$

$dom_i(C_{ans}) = dom_i(C1)$ ถ้า $m \leq i \leq n$

$dom_i(C_{ans}) = \{ d^a \mid d^a \in dom_i(C)$ หรือ $d^a \in f'_i(d'), d' \in dom_i(C1) \}$

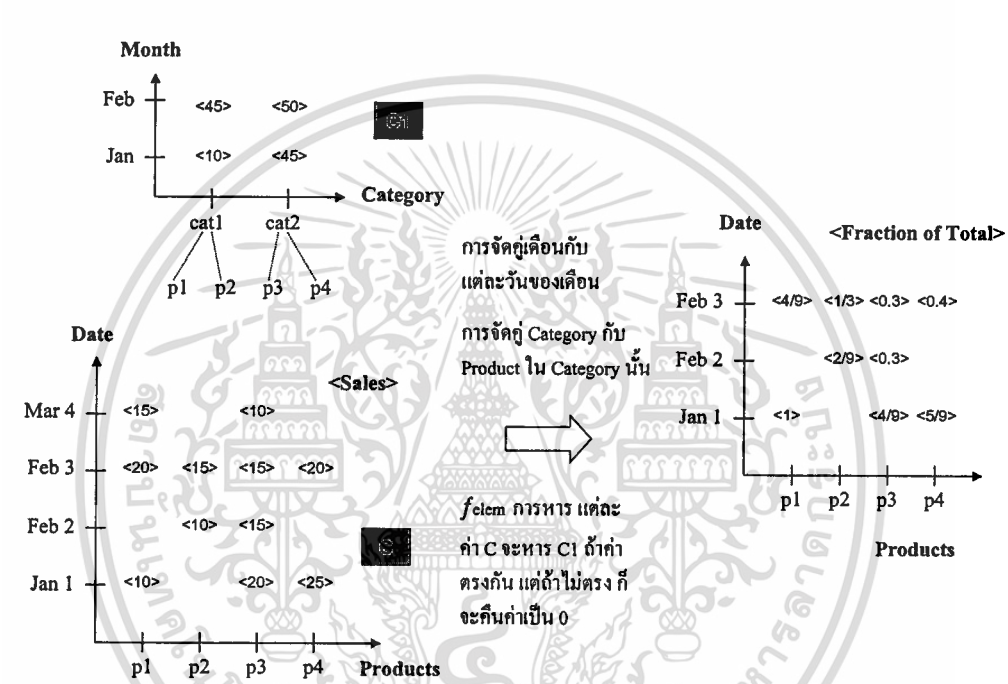
$\mathcal{E}(C_{ans})(d_1,\ldots,d^a_{m-k},\ldots, d^a_m,\ldots, d_n) = f$elem $(\{t1\}, \{t2\})$ ดังนั้น

$t1 = \mathcal{E}(C)(d_1,\ldots,d_{m-k},\ldots, d_m)$, $t2 = \mathcal{E}(C1)(d'_{m-k},\ldots, d'_m,\ldots, d_{m+1},\ldots, d_n)$

และ

$d^a_i \in f_i(d_i)$ หรือ $d^a_i \in f'_i(d'_i)$ สำหรับค่า $m-k \leq i \leq m$

สำหรับโอเปอร์เรเตอร์จอยน์ มี 2 กรณีที่ไม่สามารถทำได้ คือ cartesian product และ associate ในกรณีของ cartesian product คือ มีสองคูปที่ไม่มีมิติปกติที่ทำการจอยน์กันได้ ส่วน associate จะมีใช้ในระบบของ OLAP สำหรับการคำนวณที่มีลักษณะ "ทำด่วน สำหรับหายอดขายของแต่ละเดือน คิดเป็นเปอร์เซ็นต์จากยอดขายรายไตรมาส" โดย associate เป็น asymmetric และต้องการแต่ละมิติของ *C1* มาทำการจอยน์กับบางมิติของ *C* รูปที่ 3-6 เป็นการแสดงการทำ associate สำหรับคูป *C1* กับ *C*



รูปที่ 3-6 แสดงการทำ associate ของสองคูป

ในแต่ละเดือนในมิติ *C1* และแต่ละวันในมิติ *C* ทำการจอยน์โดยแมปที่ product ของ C_{ans} สำหรับมิติของเดือน จะมีการแมปไปยังวันที่ทั้งหมดของเดือนนั้น สำหรับมิติของ categories จะมีค่าเป็น cat1 ที่จะแมปไปยัง product p1, p2, และ cat2 จะแมปไปยัง p3, p4 สำหรับมิติวันที่และสินค้าจะมีการระบุวิธีการแมบโดยใช้ฟังก์ชั่น $f$elem การหารค่าจากคูป *C* ด้วยค่าในคูป *C1* ถ้าแต่ละค่าเป็น 0 ผลลัพธ์ที่ได้ก็จะเป็น 0 ดังจะเห็นว่าค่าใน Mar4 ที่อยู่ใน C_{ans} ไม่มีเนื่องจากค่าทั้งหมดไม่ตรงกันเลยจึงได้ค่าเป็น 0

3.2.6 Merge

โอเปอร์เรชั่น Merge เป็นการทำสรุป ดังรูปที่ 3-7 ซึ่งแสดงให้เห็นว่าลำดับชั้นในฐานข้อมูลมุมมองหลายมิติเป็นอย่างไร เพื่อที่จะทำการประยุกต์ใช้สำหรับโอเปอร์เรชั่นเมิร์จ

ฟังก์ชั่นในการเมิร์จเป็นการใช้การแมปชื่อ product หลายๆ ค่าที่อยู่ในหนึ่งหรือหลาย categories และฟังก์ชั่นอื่นๆ ที่ถูกใช้เพื่อแมปข้อมูลที่มีค่าตรงกันของเดือน ดังนั้นหลายๆค่าบนแต่และมิติที่ถูกเมิร์จจะถูกสร้างเป็นมิติใหม่ซึ่งเป็นไปได้ว่าจะมีขนาดเล็กกว่าโดเมนเดิม ผลของการเมิร์จแต่ละมิติ จะมีการรวมค่าและสามารถใส่ฟังก์ชั่นเฉพาะเข้าไปเพื่อหาผลสรุปได้ จากตัวอย่างของฟังก์ชั่น aggregate ที่ใช้ f_{elem} คือ SUM

โดยทั่วไปการใช้ฟังก์ชั่นเมิร์จมิติอาจจะเป็นความสัมพันธ์แบบ หนึ่งต่อหลาย ทำให้ค่าในระบบล่างจะรวมอยู่ในค่าของระดับที่สูงกว่าของลำดับชั้น

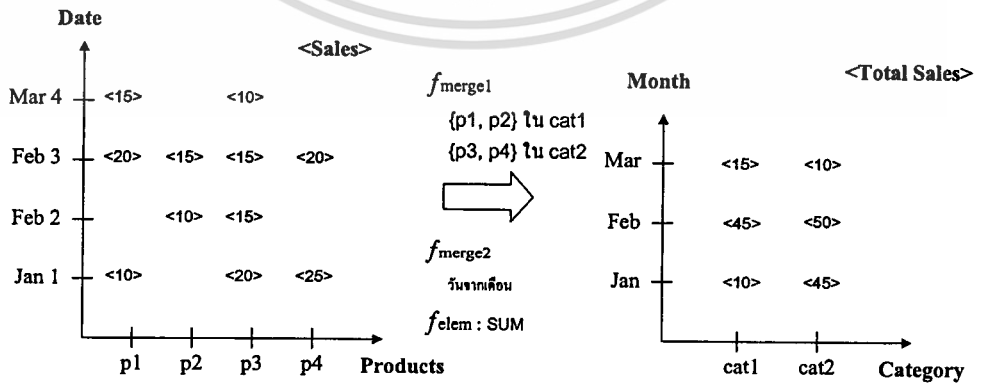
- Input : C ฟังก์ชั่น f_{elem} สำหรับการเมิร์จค่าและคู่ m (มิติ, ฟังก์ชั่น)
  สมมติว่าคู่ m เป็น $[\mathcal{D}_1, f_{\text{merge}_1}], \ldots, [\mathcal{D}_m, f_{\text{merge}_m}]$
- Output : คูป C_{ans} ของบางส่วนที่มีลักษณะใกล้เคียง C จะมีการเมิร์จมิติ $\mathcal{D}_i$ โดยการใช้ฟังก์ชั่น f_{merge_i} โดยค่าที่ตรงกันจะถูกรวมให้เป็นค่าของ f_{elem}
- Mathematically : $\text{merge}(C, \{[\mathcal{D}_1, f_{\text{merge}_1}], \ldots, [\mathcal{D}_m, f_{\text{merge}_m}]\}, f_{\text{elem}}) = C_{ans}$

  $dom_i(C_{ans}) = \{ f_{\text{merge}_i}(e) \mid e \in dom_i(C) \}$ ถ้า $1 \le i \le m$

  ไม่เช่นนั้น $dom_i(C_{ans}) = dom_i(C)$

  $\mathcal{E}(C_{ans})(d_1, \ldots, d_k) = f_{\text{elem}}(\{ t \mid t = \mathcal{E}(C)(d'_1, \ldots, d'_k)$

  เมื่อ $f_{\text{merge}_i}(d'_i) = d_i$ ถ้า $1 \le i \le m$ ไม่เช่นนั้น $d'_i = d_i$ $\})$



รูปที่ 3-7 แสดงการเมิร์จมิติ Date และ Product โดยใช้ f_{elem} = SUM

3.3 Multidimensional Query Language

ภาษาสำหรับการค้นหาข้อมูลหลายมิติ เป็นภาษาที่คิดค้นเพิ่มเติมจาก SQL พื้นฐาน โดยรูปแบบของภาษาจะมีความแตกต่างกันไป ขึ้นกับผลิตภัณฑ์ที่ใช้งาน ในรายงานนี้ได้นำเสนอ Multidimensional Expression (MDX) ซึ่งสนับสนุน OLE DB สำหรับ OLAP ซึ่งลักษณะของการทำงานของ MDX Query จะเป็นการดึงข้อมูลจากดาต้าคูป เพื่อได้ดาต้าเซ็ตสำหรับนำมาใช้งาน ดังรูปที่ 3-8

รูปที่ 3-8 แสดงภาพเบื้องต้นของการทำงาน MDX Query

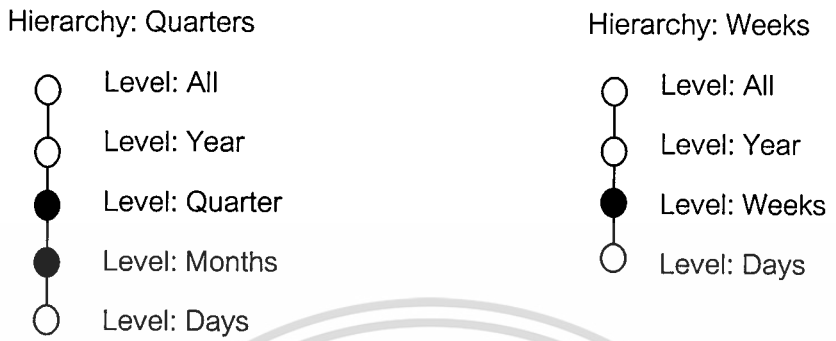
คำจำกัดความของ MDX Data Cube

Dimension	หมายถึง เซ็ตของกลุ่มข้อมูลที่มีชนิดเดียวกัน
Member	หมายถึง สมาชิกของ Dimension
Hierarchy	หมายถึง กลุ่มสมาชิกของ Dimension
Member-Properties	หมายถึง รายละเอียดเพิ่มเติมของแต่ละ Member

ตัวอย่าง MDX Data Cube

4 Dimensions	SalesPerson, Time, Measures, Product
SalesPerson-dimension	Komol, Wanchai, …,Sitthisak
Time-dimension	2001, Jan 2002, Qtr2 2003, …
Measures-dimension	Sales, Profit-Loss
Product-dimension	Computer, Printer, …
Member-Properties of SalesPerson-dimension	Name, Phone, …

รูปแบบ Hierarchy-Schemata บน Time-dimension



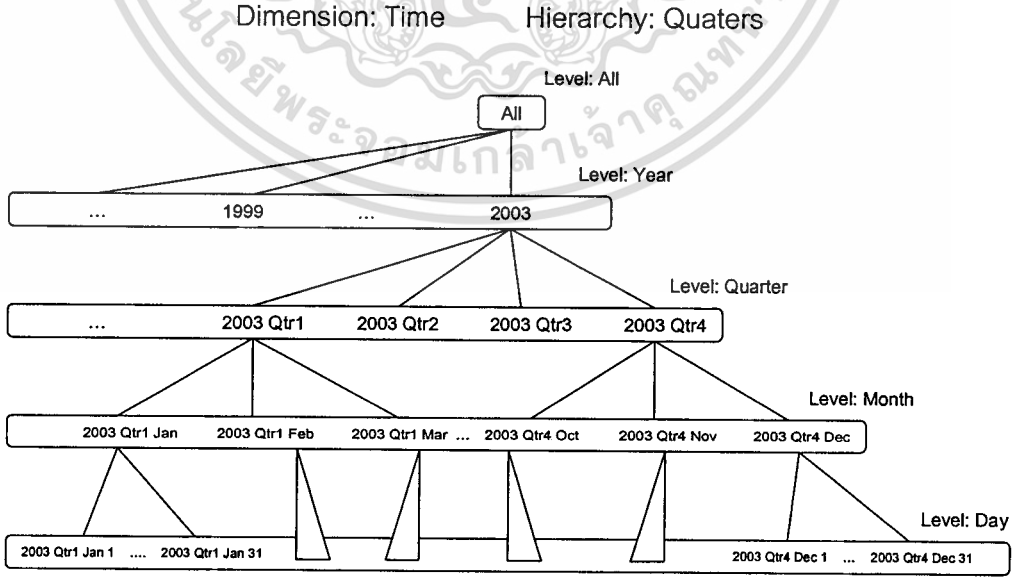
รูปที่ 3-9 แสดงภาพโครงสร้างลำดับชั้นที่กำหนดเป็นรูปแบบของ Time-dimension

จากรูปที่ 3-9 แสดงถึงลำดับของลำดับชั้นของเวลา โดยแบ่งประเภทของแต่ละลำดับดังนี้

○ หมายถึง สมาชิกของไดเมนชั่นที่บรรจุค่าลำดับชั้นที่แตกต่างกัน

● หมายถึง สมาชิกของไดเมนชั่นที่บรรจุค่าเพียงหนึ่งลำดับชั้น

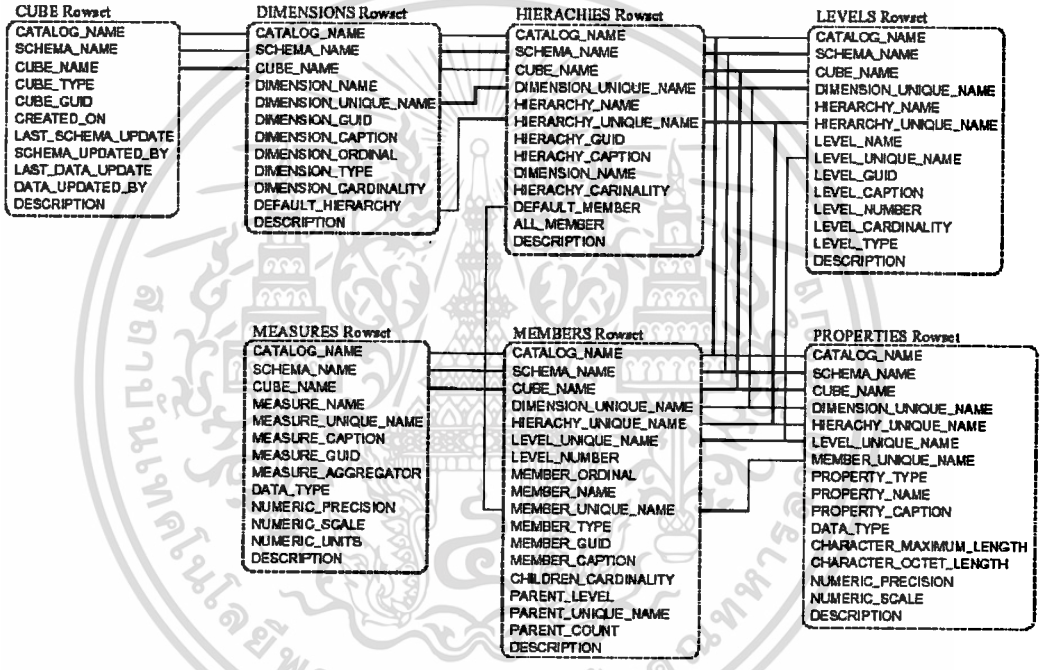
ตัวอย่างโครงสร้างข้อมูลลำดับชั้นบน Time-dimension



รูปที่ 3-10 แสดงภาพโครงสร้างข้อมูลลำดับชั้นบน Time-dimension

จากรูปที่ 3-10 เป็นภาพแสดงถึงความสัมพันธ์ในการจัดกลุ่มของลำดับชั้นของไดเมนชั่น โดยแบ่งลำดับชั้นตามรูปแบบ Quarter ที่แสดงดังรูปที่ 3-9 แล้วนั้น ซึ่งลำดับของข้อมูลที่เห็นได้แบ่งเป็นลำดับของความสัมพันธ์คือชั้นบนสุดจะแสดงความสัมพันธ์ทั้งหมดของปี และแต่ละปี จะมีลำดับชั้นความสัมพันธ์ของไตรมาสของปีนั้น และแต่ละไตรมาสจะสัมพันธ์กับเดือนและแต่ละเดือนจะสัมพันธ์กับช่วงวันของเดือนนั้นด้วย

โดยลำดับความสัมพันธ์ของ Dimension จะมีโครงสร้างของการจัดเก็บเมตะดาต้าสำหรับ Rowset ของ CUBE ดังรูปที่ 3-11



รูปที่ 3-11 แสดงภาพโครงสร้างของการจัดเก็บเมตะดาต้าในCUBE Schema

โครงสร้างภาษาสำหรับ MDX

รูปแบบของภาษาของคิวรีที่ใช้ในการสอบถามข้อมูลใน Cube มีโครงสร้างดังนี้

[WITH <formula_spec>]

SELECT <axis_spec> [, <axis_spec>, ...]

FROM <cube_spec> [, <cube_spec>, ...]

WHERE <slicer_spec> [cell_properties]

คำอธิบาย Syntax ใน MDX

WITH : ใช้เพื่อสร้างไดเมนชั่นใหม่ ด้วยสิ่งที่อยู่ในประโยคของ Expression เช่น

WITH MEMBER Measures.[% Change from Last Year's PolicyCount] AS

PolicyCount / (PolicyCount, Year.PREVMEMBER)

<axis_spec> : การกำหนดไดเมนชันให้อยู่แกนมิติที่กำหนด เช่น

2543, 2544 ON COLUMNS

<cube_spec> : การระบุข้อมูลที่อยู่ในคูบ (ส่วนมากจะระบุเพียงตัวเดียว)

<slicer_spec> : จำกัดไดเมนชั่นไม่ให้อยู่ <axis_spec> ที่กำหนดไว้

ตัวอย่าง MDX Query

```
SELECT CROSSJOIN({PolicyType.CHILDREN},
        {VehicleType.CHILDREN,
            110, 120})
                ON COLUMNS
            {BrandModel.CHILDREN}
                ON ROWS
FROM Policy_Profile
WHERE (UYr, [2543], PolicyCount.All)
```

บทที่ 4
การพัฒนาระบบคลังข้อมูลการประกันวินาศภัย

จากทฤษฎีเกี่ยวกับเทคโนโลยีดาต้าแวร์เฮาส์ และโมเดลในการพัฒนาฐานข้อมูลหลายมิติ ดังที่กล่าว ดาต้าแวร์เฮาส์ ที่ได้กล่าวไปแล้ว สามารถนำมาใช้ในการพัฒนาระบบคลังข้อมูลการประกันวินาศภัยได้ ซึ่งในโครงการพัฒนาระบบนี้ จะประกอบด้วยกระบวนการเตรียมข้อมูล การออกแบบ การประมวลผลข้อมูล และการนำเสนอสารสนเทศมุมมองหลายมิติ เมื่อได้สารสนเทศที่ผ่านการประมวลผลแล้ว จะเป็นขั้นตอนในการเรียกใช้สารสนเทศในรูปแบบ Ac Hoc Query

4.1 วัตถุประสงค์ในการจัดทำดาต้าแวร์เฮาส์

ในการจัดทำดาต้าแวร์เฮาส์สำหรับระบบงานประกันวินาศภัยรถยนต์ภาคสมัครใจ มีวัตถุประสงค์ในการลดเวลาของการประมวลผลข้อมูล ช่วยให้ง่ายต่อการสอบถาม ซึ่งสารสนเทศจากการวิเคราะห์จะช่วยให้ผู้ใช้งานข้อมูลสามารถนำไปใช้ร่วมสำหรับการตัดสินใจ สำหรับแนวโน้มและทิศทางทางการตลาดที่เป็นไปได้ในอนาคต โดยข้อมูลดังกล่าวจะเป็นภาพรวมของตลาด ทั้งนี้ผู้ใช้ข้อมูลสามารถนำข้อมูลที่มีอยู่ขององค์กรตนเองมาใช้เปรียบเทียบได้ง่ายขึ้น

4.2 การเตรียมข้อมูลและการทำดาต้าคลีนนิ่ง

เนื่องด้วยข้อมูลจากฐานข้อมูลในระบบงานประกันวินาศภัยเดิมจะเป็นข้อมูลในลักษณะทรานเซคชั่น และจะต้องมีการประมวลผลเชิงสถิติเพื่อออกรายงาน ซึ่งข้อมูลดังกล่าวยังไม่มีความเหมาะสมในเรื่องของข้อมูล เช่น ไม่มีการระบุรุ่นของรถยนต์ หรือค่าของข้อมูลบางค่าอยู่นอกโดเมนของข้อมูลที่ต้องการวัดค่าทำให้เกิดปัญหาในการประมวลผลที่นานมาก จึงมีความจำเป็นที่จะต้องเตรียมรายการทรานเซคชั่น เหล่านี้ให้อยู่ใน View หรือเป็นข้อมูลที่พร้อมในการประมวลผล ดังมีขั้นตอนดังนี้

4.2.1 การกำหนดโครงสร้างฐานข้อมูลสำหรับดาต้าแวร์เฮาส์

เป็นขั้นตอนที่สำคัญ เพราะความถูกต้องของระบบจะต้องมาจากพื้นฐานการได้มาของข้อมูลที่ถูกต้อง ซึ่งข้อมูลที่จะต้องสร้างได้แก่ ข้อมูลการรับประกันภัย และข้อมูลสินไหมทดแทน เป็นส่วนสำคัญในการวิเคราะห์ระบบงาน

POLICY_PROFILE
UYr
InsNo
FleetNo
Fleet_Count
PolicyNo
AcctYr
IssueDate
PolType
BeginDate
ExpireDate
ExposureUnit
TotPremPerExpUnit
::
VehType
VehGroup
VehBrand
VehModel
VehSubModel
VehYear
VehAge
VehSize
AddEquip
ODFT_SI_Band
ODFT_SI
PD_LIMIT
BI_PersonLimit
BI_EventLimit
BasePrem
::
FleetDiscRate
FleetDisc_PolicyCount
FleetDiscAmt
::
PolStatus

CLAIM_PROFILE
UYr
InsNo
PolicyNo
AcctYr
IssueDate
PolType
BeginDate
ExpireDate
ExposureUnit
TotPremPerExpUnit
MainCoverPremPerExpUnit
::
VehType
VehGroup
VehBrand
VehModel
VehSubModel
VehYear
VehAge
VehSize
::
OD_ClaimCount
OD_LI_Total
OD_Paid_Total
Fire_TotalLossCount
Fire_ClaimCount
Theft_ClaimCount
THEFT_LI
THEFT_Paid
THEFT_OS
::
Recovery

รูปที่ 4-1 แสดงภาพ Data warehouse schema ของระบบคลังข้อมูลการประกันวินาศภัย

- ข้อมูลการรับประกันภัย

เป็นข้อมูลเกี่ยวกับการปรับประกันภัยรถยนต์ สำหรับข้อมูลที่ใช้ในการวัดค่า ได้แก่ เบี้ยรับประกันภัยและเบี้ยส่วนเพิ่มสำหรับความคุ้มครองประเภทต่างๆ, เบี้ยประกันภัยรวม, วงเงินความคุ้มครอง และ

ข้อมูลที่ใช้ในการกำหนดโดเมนของข้อมูลได้แก่ ประเภทของกรมธรรม์, บริษัทรับประกันภัย, ปีรับประกันภัย, ประเภทรถยนต์, ยี่ห้อ-รุ่นของรถยนต์, อายุรถยนต์, ระดับความเสี่ยง

- ข้อมูลสินไหมค่าทดแทน

เป็นข้อมูลเกี่ยวกับการจ่ายค่าสินไหมทดแทน เมื่อเกิดเหตุตามปีรับประกันภัยและเลขที่กรมธรรม์ที่ตรงกับการข้อมูลการรับประกันภัย สำหรับข้อมูลที่ใช้ในการวัดค่า ได้แก่ วงเงินคุ้มครอง, ความเสียหายต่อทรัพย์สิน, ความเสียหายต่อร่างกาย, ความเสียหายที่เกิดจากการโจรกรรมหรือไฟไหม้, สินไหมรวม

ข้อมูลที่ใช้ในการกำหนดโดเมนของข้อมูลได้แก่ ประเภทของกรมธรรม์, บริษัทรับประกันภัย, ปีรับประกันภัย, ประเภทรถยนต์, ยี่ห้อ-รุ่นของรถยนต์, อายุรถยนต์, ระดับความเสี่ยง ในการเตรียมข้อมูลสำหรับข้อมูลการรับประกันภัยและสินไหมค่าทดแทน จะทำการเตรียมด้วยกระบวนการ Data Transformation และ Store Procedure

4.2.2 การทำดาต้าคลีนนิ่ง (Data Cleaning)

กระบวนการทำดาต้าคลีนนิ่ง หมายถึง การลดความผิดเพี้ยน ในตัวข้อมูลให้สามารถประมวลผลได้ถูกต้อง หรือใกล้เคียงมากที่สุด สำหรับปัญหาของการเกิดความผิดเพี้ยนของข้อมูลในระบบมาเนื่องจากที่มาของข้อมูล ทั้งข้อมูลการรับประกันภัยและข้อมูลสินไหม มาจากบริษัทประกันภัย ซึ่งมีการป้อนข้อมูลด้วยวิธีที่แตกต่างกัน เมื่อรวมรวมข้อมูลเหล่านั้นไว้ด้วยกัน ย่อมทำให้เกิดความหลากหลายของข้อมูล อย่างเล่น ยี่ห้อรถยนต์ จะไม่มีมาตรฐานของการบันทึกชื่อ เป็นต้น ดังนั้นในขั้นตอนการเตรียมทำ Data Cleaning มีดังต่อไปนี้

- พิจารณาข้อมูลที่จะใช้ในการสอบถาม หรือใช้เป็นมุมมองของรายงานที่ต้องการ ซึ่งเหล่านี้ถือเป็นไดเมนชั่น ซึ่งทั้งข้อมูลการรับประกันภัย และข้อมูลสินไหมทดแทน ซึ่งได้แก่ ปีรับประกันภัย, ประเภทกรมธรรม์, ประเภทรถยนต์, ยี่ห้อ-รุ่นของรถยนต์ เป็นต้น
- ตรวจสอบค่าของแต่ละมุมมองที่สนใจ แล้วพิจารณาค่าของข้อมูลเหล่านั้นว่าอยู่นอกโดเมนที่กำหนดหรือไม่

ตารางที่ 4-1 แสดงชื่อของมุมมองข้อมูลของกลุ่มข้อมูลจากดาต้าแวร์เฮาส์ และโดเมนข้อมูลที่จะทำคลีนนิ่ง

ไดเมนชั่น	โดเมนที่กำหนด	ค่าที่อยู่นอกโดเมน
ปีรับประกันภัย	…, 2540, 2541, …	
ประเภทกรมธรรม์	1, 2, 3	
ประเภทรถยนต์	1, 2, 3, 4, 5, {ไม่ระบุ}	<Null> จะเปลี่ยนเป็นไม่ระบุ
ยี่ห้อ-รุ่นรถยนต์	Toyota, Honda, Nissan, Mazda, Ford, BMW, Volvo, Benz, {Other}	ทุกค่าที่อยู่นอกจาก List ที่สนใจ จะถือว่าเป็น Other
อายุรถ	1, 2, …, 10, {99}	ค่าอื่นๆ ที่เกิน 10 จะเปลี่ยนเป็น 99
จังหวัดทะเบียนรถ	กท, อย, … {NA}	<Null> หรือ ไม่อยู่ใน List จะถือว่าเป็นค่า NA
หน่วยความเสี่ยง	-1, {0}, 1, …, 10	<Null> จะตั้งค่าเป็น 0 ถ้าค่า > 10 จะให้ค่าเป็น 10 ถ้าค่า < -1 จะให้ค่าเป็น -1
กรมธรรม์ระบุชื่อคนขับ	{0}, 1, 2	<Null> จะตั้งค่าเป็น 0 คือไม่ระบุ

จากตารางที่ 4-1 แสดงให้ทราบชื่อของมุมมองข้อมูลที่ต้องเตรียมความพร้อมสำหรับการประมวลผล ซึ่งค่าของกลุ่มโดเมนจะมีทั้งที่เป็นช่วง หรือเป็นค่าที่กำหนด และค่าที่เป็นวงเล็บ {} จะถือว่าเป็นค่า Default สำหรับใช้ในการทำ Data Cleaning ถ้าค่าข้อมูลที่ตรวจสอบอยู่นอกโดเมนดังกล่าว

จากตัวอย่างเช่น ยี่ห้อ-รุ่นรถยนต์ ที่มีอยู่ในปัจจุบันมีหลายยี่ห้อและหลายรุ่นมาก แต่ในการสรุปข้อมูล ผู้สอบถามต้องการรู้เพียงไม่กี่ยี่ห้อ ซึ่งต้องการเฉพาะยี่ห้อตลาดเท่านั้น เช่น Toyota, Honda, Nissan, Mitsubishi, Benz, BMW, Volvo เป็นต้น ส่วนยี่ห้ออื่นๆ เช่น Daihatsu มีจำนวนน้อยมากเมื่อเทียบกับตลาดรถยนต์ ยี่ห้ออื่น ดังนั้น ค่าข้อมูลที่อยู่นอกรายการที่ระบุ จะเปลี่ยนเป็น Other ทำให้สามารถประมวลผลข้อมูลได้ครบทุกรายการ

4.3 การออกแบบและทำการประมวลผลข้อมูล OLAP

เนื่องจากข้อมูลที่อยู่ในดาต้าแวร์เฮาส์ เราไม่สามารถใช้งานข้อมูลเหล่านั้นโดยตรงได้อย่างมีประสิทธิภาพ จึงจำเป็นที่จะต้องนำข้อมูลเหล่านั้นมาผ่านกระบวนการประมวลผลที่เหมาะสม จึงจะทำให้การเรียกใช้ข้อมูลทำงานได้มีประสิทธิภาพสูง

โดยกระบวนการในการออกแบบเป็นการกำหนดเกี่ยวกับเมตะดาต้าของโครงสร้างไดเมนชั่น, คูป แล้วจึงทำการประมวลผลข้อมูล

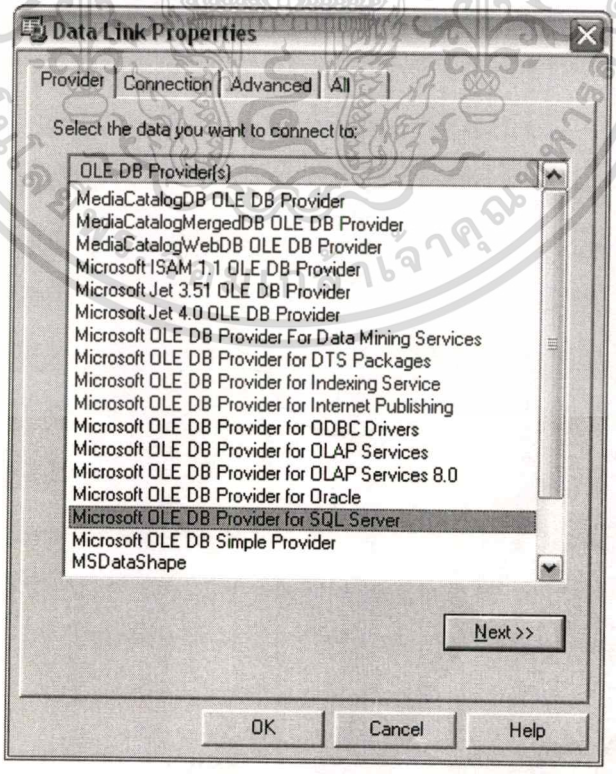
4.3.1 การระบุแหล่งที่มาของข้อมูล (Data Source)

แหล่งที่มาของข้อมูล เป็นข้อมูลที่จะใช้ในการประมวลผล ซึ่งอาจจะเป็น Text File, Excel, ระบบฐานข้อมูล ซึ่งจะสามารถสร้างตัวเชื่อมโยงแหล่งข้อมูลได้ตามที่มีอยู่ใน Provider ของระบบ เช่น การใช้ ODBC เพื่อกำหนด Data Source เชื่อมโยงฐานข้อมูล Oracle เป็นต้น สำหรับระบบนี้ จะทำการเชื่อมโยงฐานข้อมูล Microsoft SQL

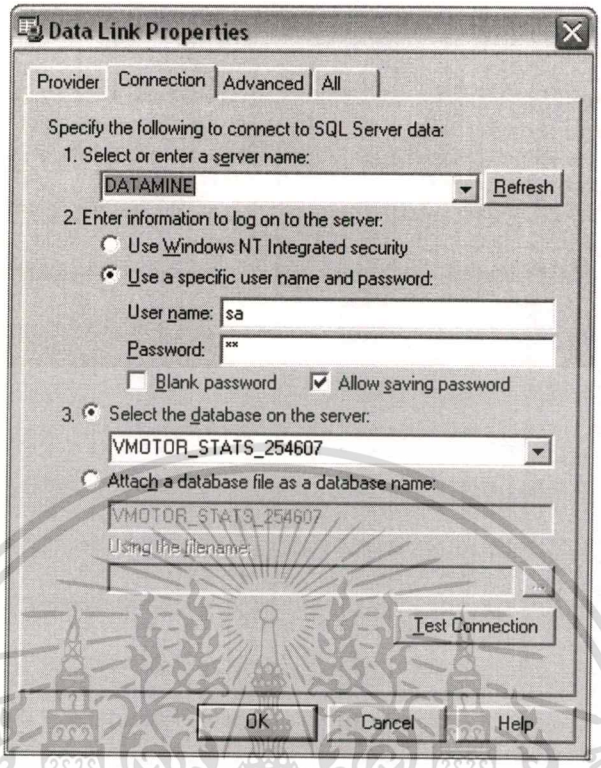
ขั้นตอนที่ 1: เลือกการเชื่อมโดยโดยเลือก “Microsoft OLE DB Provider for SQL”

ขั้นตอนที่ 2: ระบุชื่อ Server, User Name, Password, ชื่อฐานข้อมูล

ขั้นตอนที่ 3: ตั้งชื่อ Data Source เพื่อใช้ในการอ้างอิงการเชื่อมโยงข้อมูล



รูปที่ 4-2 แสดงการสร้าง Data Source (โดยเลือก Provider Connection)



รูปที่ 4-3 แสดงการสร้าง Data Source (ในการระบุ Parameter สำหรับ Connection)

4.3.2 การสร้างไดเมนชั่น (Dimensions)

สำหรับระบบงานที่พัฒนา จะเลือกการสร้างไดเมนชั่นที่บันทึกค่าไว้เป็นแบบ Shared Dimension เพราะค่าข้อมูลเหล่านี้ จะมีการถูกใช้ร่วมกันในคูปต่างๆ ที่สร้าง

ขั้นตอนที่ 1: เลือกประเภทของ Dimension ซึ่งแต่ละประเภทจะมีวิธีการกำหนดค่าที่แตกต่างกัน ในที่นี้เลือกแบบ Star Schema

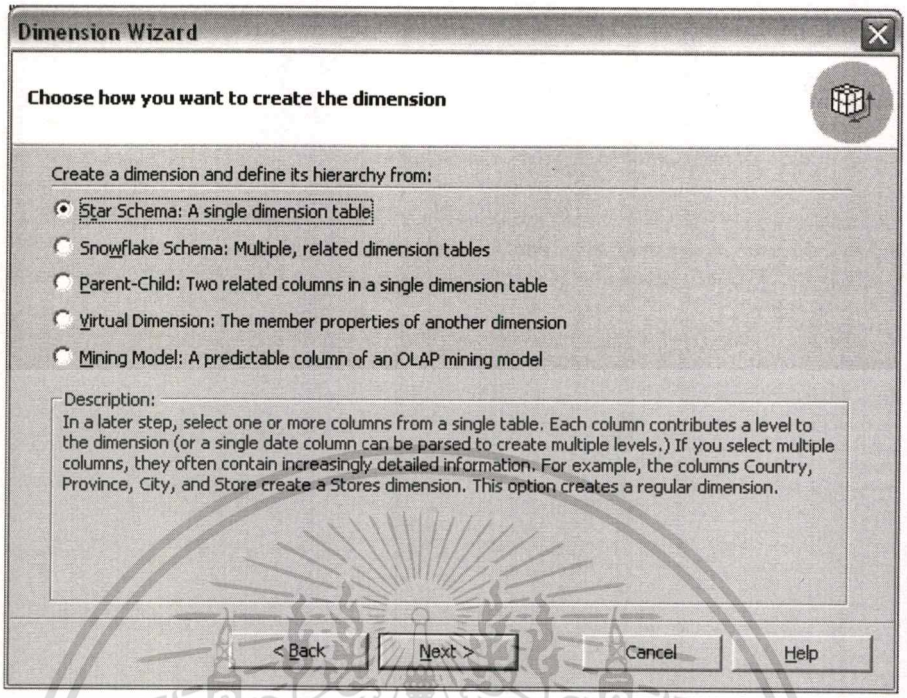
ขั้นตอนที่ 2: เลือก Table ของจาก Data Source ที่ใช้เป็น Dimension

ขั้นตอนที่ 3: กำหนด Level จากรายการ Field ที่อยู่ใน Table ที่เลือก โดย Field นี้จะใช้สำหรับแสดงค่าข้อมูลในมิตินี้

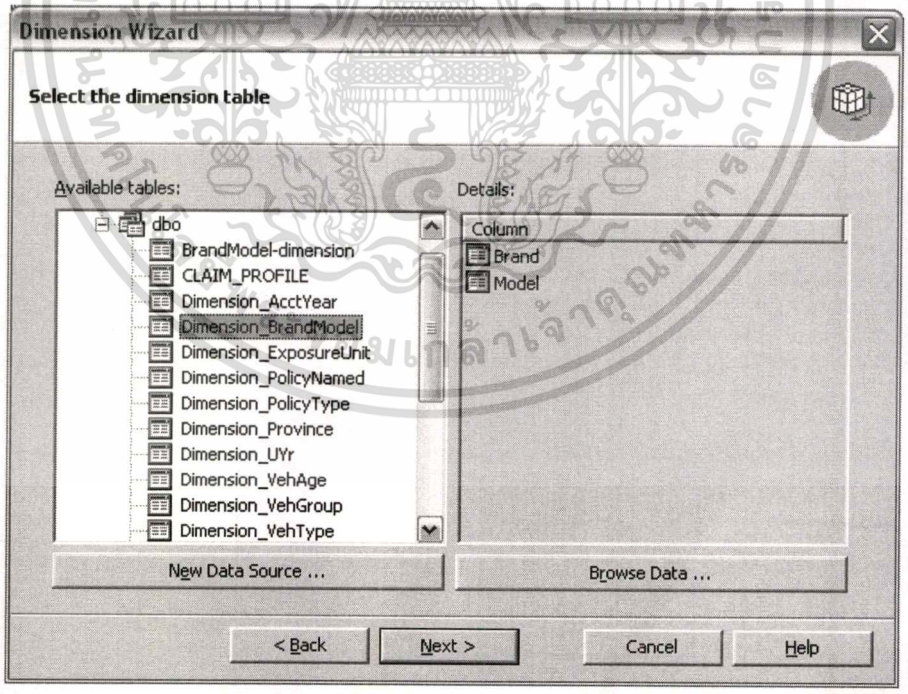
ขั้นตอนที่ 4: กำหนดฟิลด์ข้อมูลที่ใช้เป็นคีย์ ซึ่งคีย์อาจจะไม่ใช่ชื่อเดียวกับที่เลือกเป็น Level

ขั้นตอนที่ 5: บันทึกชื่อไดเมนชั่น

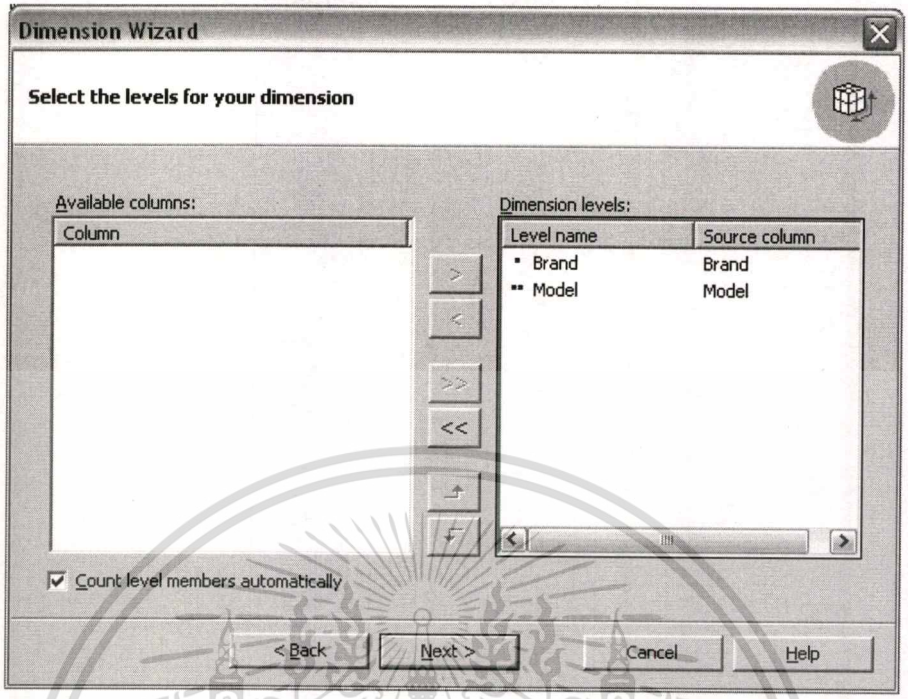
ขั้นตอนที่ 6: ปรับแต่ค่าของ Dimension มุมมองการออกแบบ



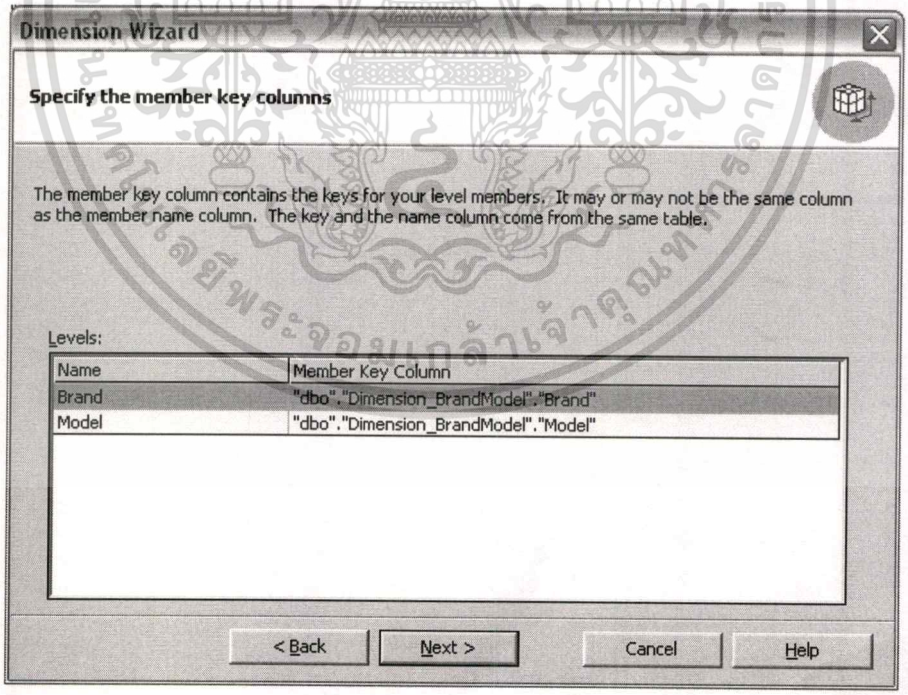
รูปที่ 4-4 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 1)



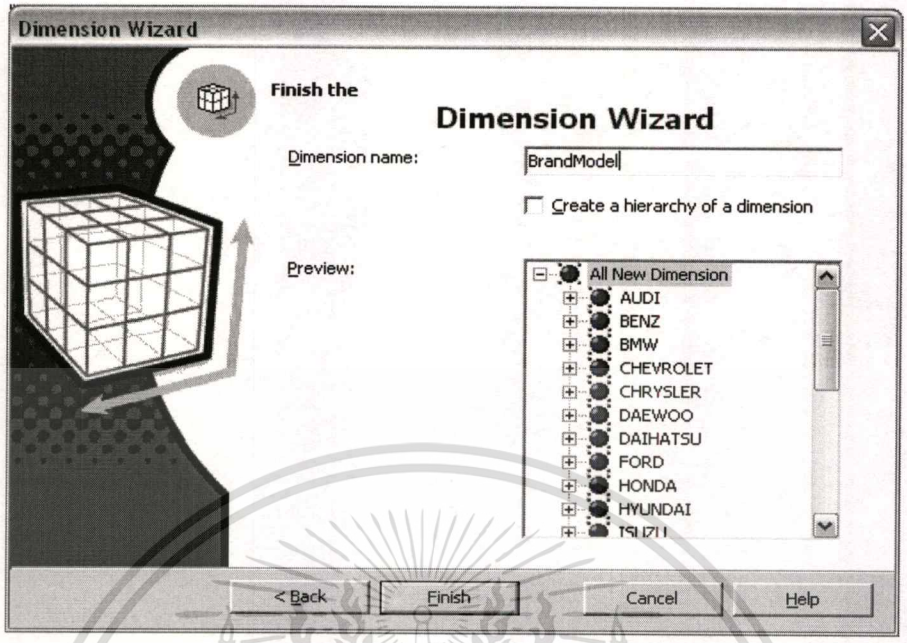
รูปที่ 4-5 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 2)



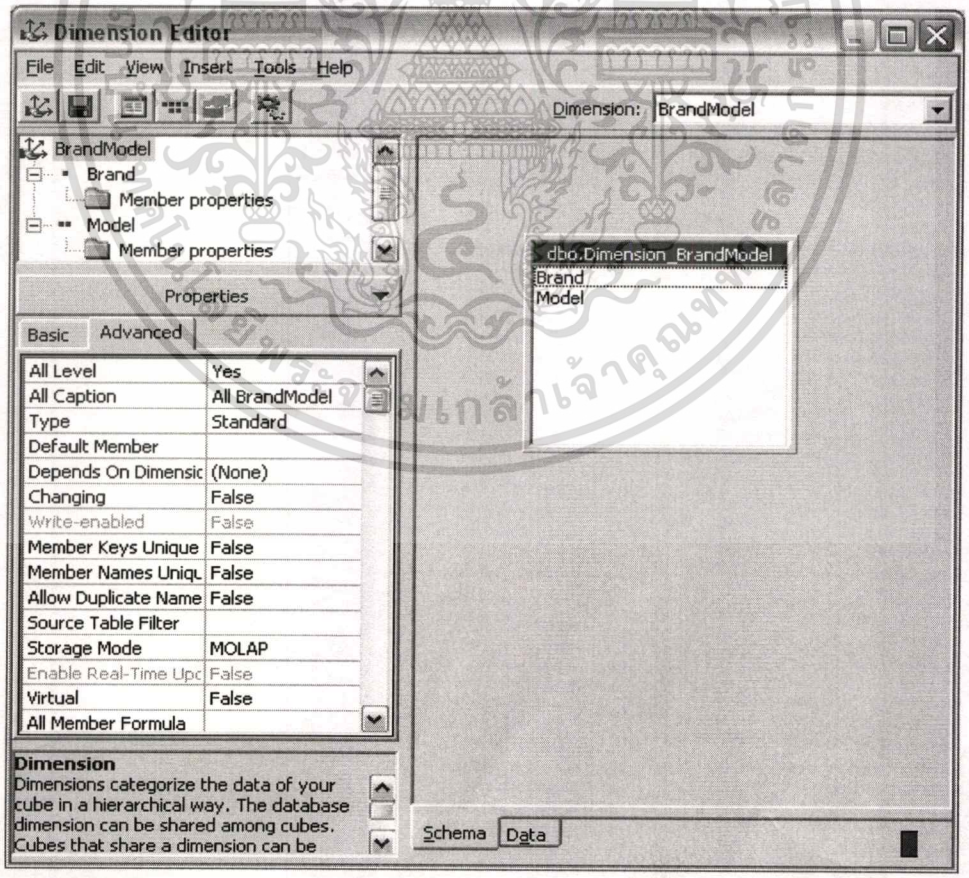
รูปที่ 4-6 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 3)



รูปที่ 4-7 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 4)



รูปที่ 4-8 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 5)



รูปที่ 4-9 แสดงการสร้างไดเมนชั่น (ขั้นตอนที่ 6)

ตารางที่ 4-2 แสดงรายการ Dimension และ Level ที่ทำสำหรับการประมวลผลข้อมูล

Dimension Name	Level Name	Description
UnderwritingYear	UYR	ปีรับประกันภัย
BrandModel	Brand Model	ยี่ห้อ – รุ่น รถยนต์
DriverNamed	Policy Named	การระบุชื่อผู้ขับขี่
ExposureUnit	Exposure Unit	หน่วยความเสี่ยง
PolicyType	Policy Type	ประเภทกรมธรรม์
Province	Province	จังหวัดที่จดทะเบียนรถยนต์
VehicleType	Veh Type	ประเภทรถยนต์
VehicleGroup	Veh Group	กลุ่มรถยนต์ (ตามลักษณะรถและการใช้งาน)
VehicleAge	Veh Age	อายุรถยนต์

4.3.3 การสร้างคูป (Cube)

ตามทฤษฎีของการสร้างคูปซึ่งจะทำการเก็บข้อมูลได้หลายมิติ ซึ่งสามารถสร้างได้ทั้งแบบบันทึกข้อมูลที่ประมวลผลจริง และการสร้างคูปในลักษณะวิว (มุมมองย่อยในคูป) สำหรับการสร้างคูปทั่วไปมีขั้นตอนดังนี้

ขั้นตอนที่ 1: เลือก Table ที่ต้องการสร้างคูปจาก Data Source

ขั้นตอนที่ 2: เลือก Fields สำหรับที่ใช้ทำ Measures ซึ่งได้แก่ Fields เป็นข้อมูลที่ต้องการนำเสนอและสามารถคำนวณวัดค่าได้

ขั้นตอนที่ 3: กำหนด Dimension ที่ใช้ในการสอบถาม / ประมวลผล

ขั้นตอนที่ 4: บันทึกชื่อ Cube

ขั้นตอนที่ 5: ปรับแต่ค่าของการ Join สำหรับ Dimensions, กำหนด Aggregate สำหรับแต่ละ Measure ที่เลือก ได้แก่ Max, Min, Sum, Average เป็นต้น

ขั้นตอนที่ 6: ทำการประมวลผลข้อมูล

ในการสร้างคูป สามารถทำแบบ Virtual Cube (วิวของคูป) ได้หลังจากที่ทำการประมวลผลข้อมูลเสร็จแล้ว โดยขั้นตอนสำหรับการสร้าง Virtual Cube มีดังนี้

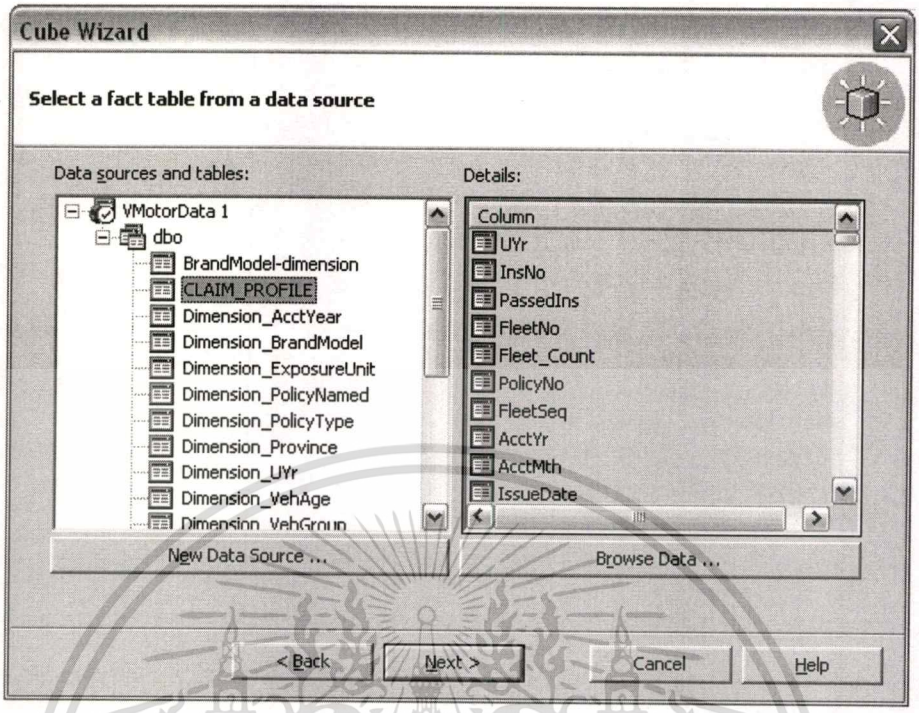
ขั้นตอนที่ 1: เลือก Cube ที่ต้องการสร้าง View ซ้อนใน Cube นั้น

ขั้นตอนที่ 2: เลือก Measure ที่ต้องการแสดง

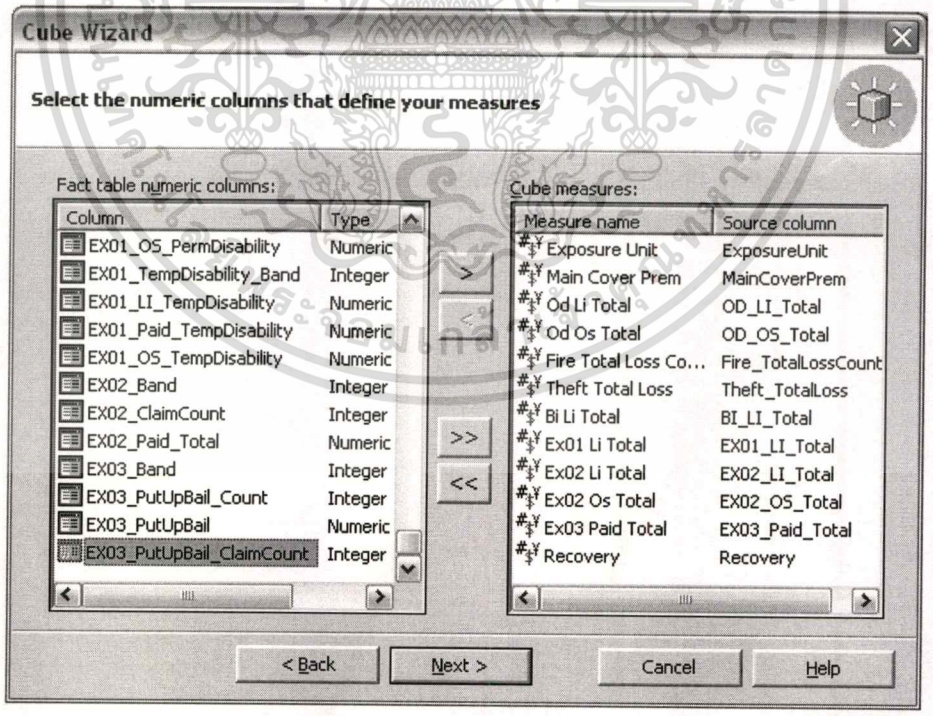
ขั้นตอนที่ 3: เลือก Dimension ที่ต้องการให้เป็นเงื่อนไขการแสดงข้อมูล

ขั้นตอนที่ 4: บันทึกชื่อ Virtual Cube

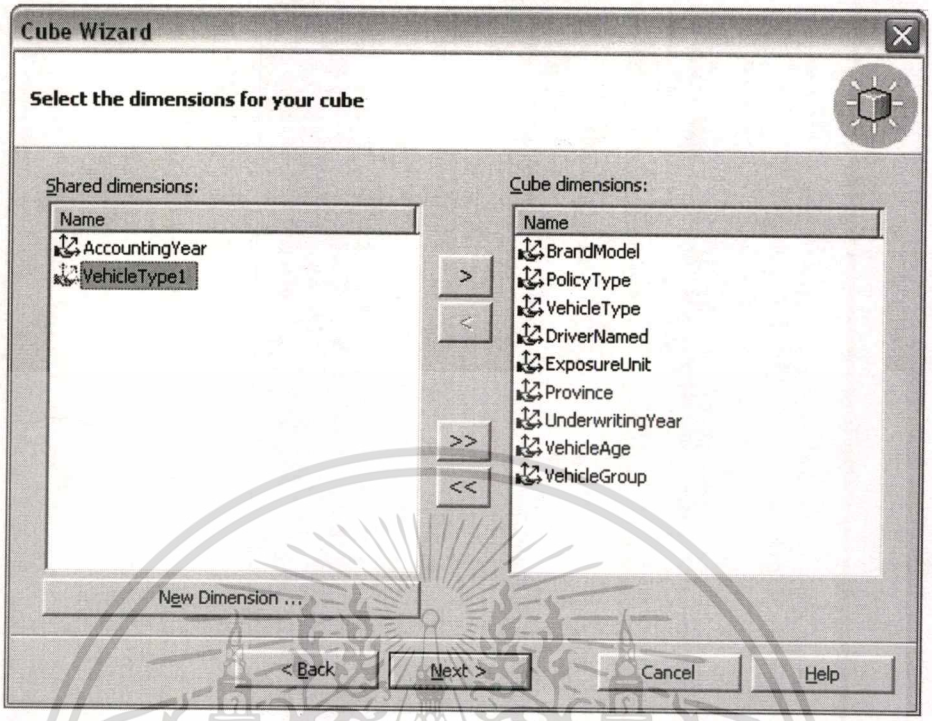
ขั้นตอนที่ 5: ปรับแก้ไขมุมมองการออกแบบ



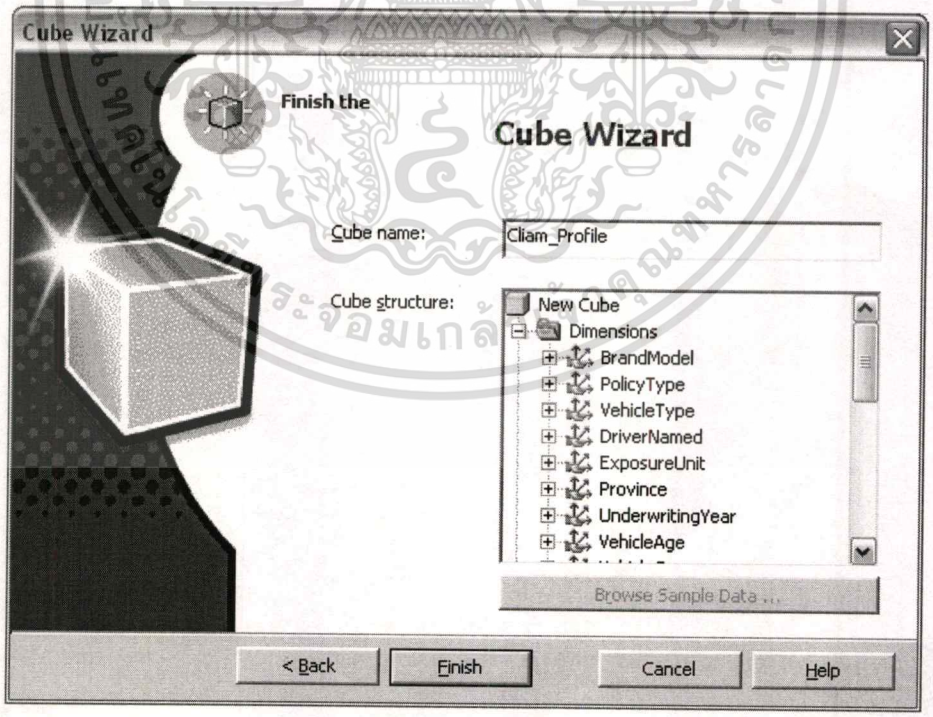
รูปที่ 4-10 แสดงการสร้างคูบ (ขั้นตอนที่ 1)



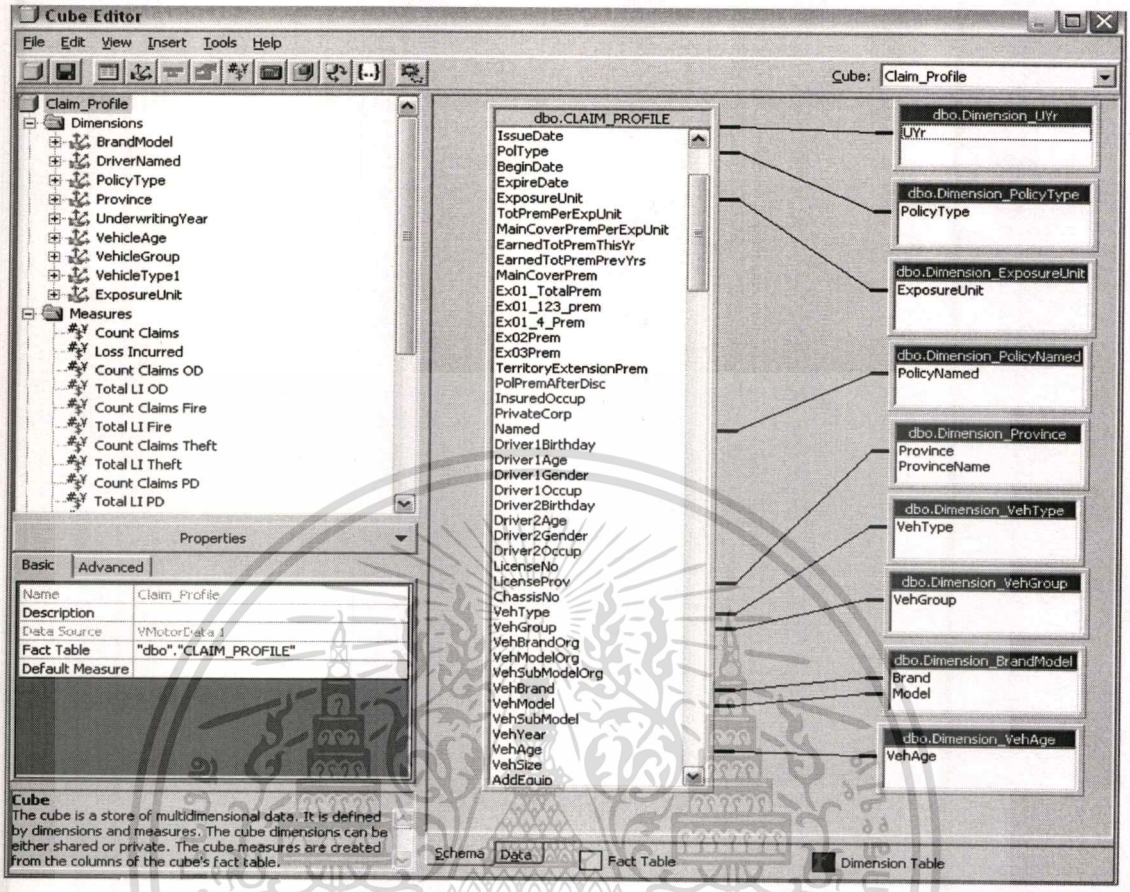
รูปที่ 4-11 แสดงการสร้างคูบ (ขั้นตอนที่ 2)



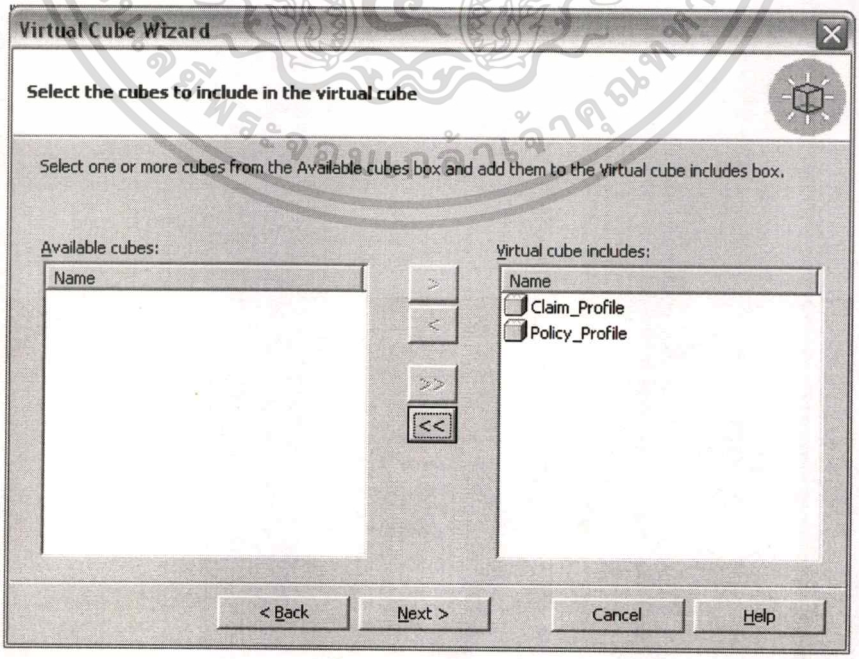
รูปที่ 4-12 แสดงการสร้างคูป (ขั้นตอนที่ 3)



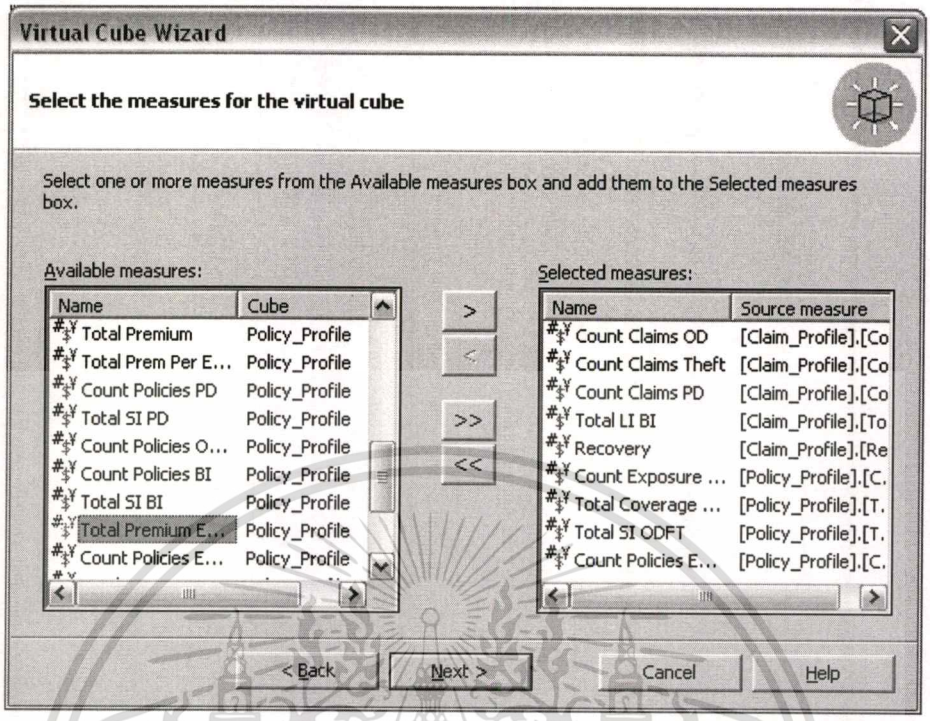
รูปที่ 4-13 แสดงการสร้างคูป (ขั้นตอนที่ 4)



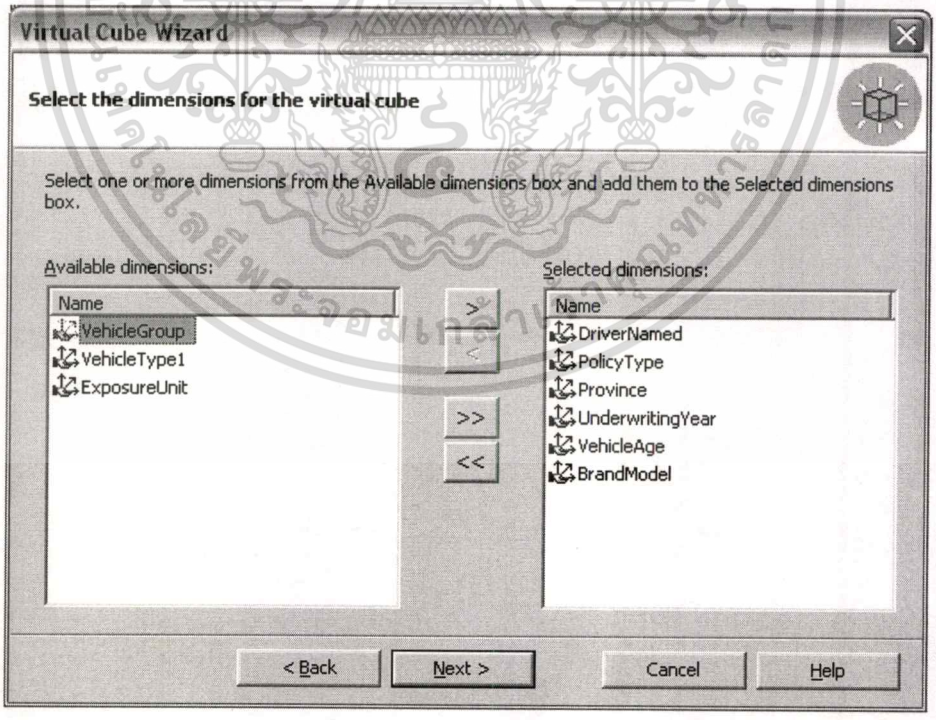
รูปที่ 4-14 แสดงการสร้างคูป (ขั้นตอนที่ 5)



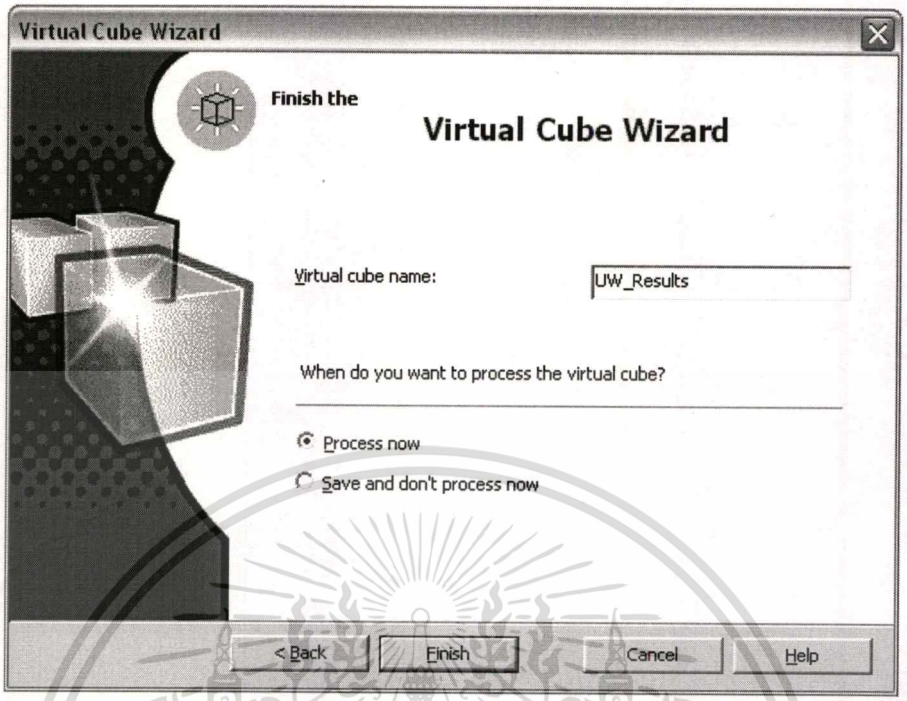
รูปที่ 4-15 แสดงการสร้าง Virtual Cube (ขั้นตอนที่ 1)



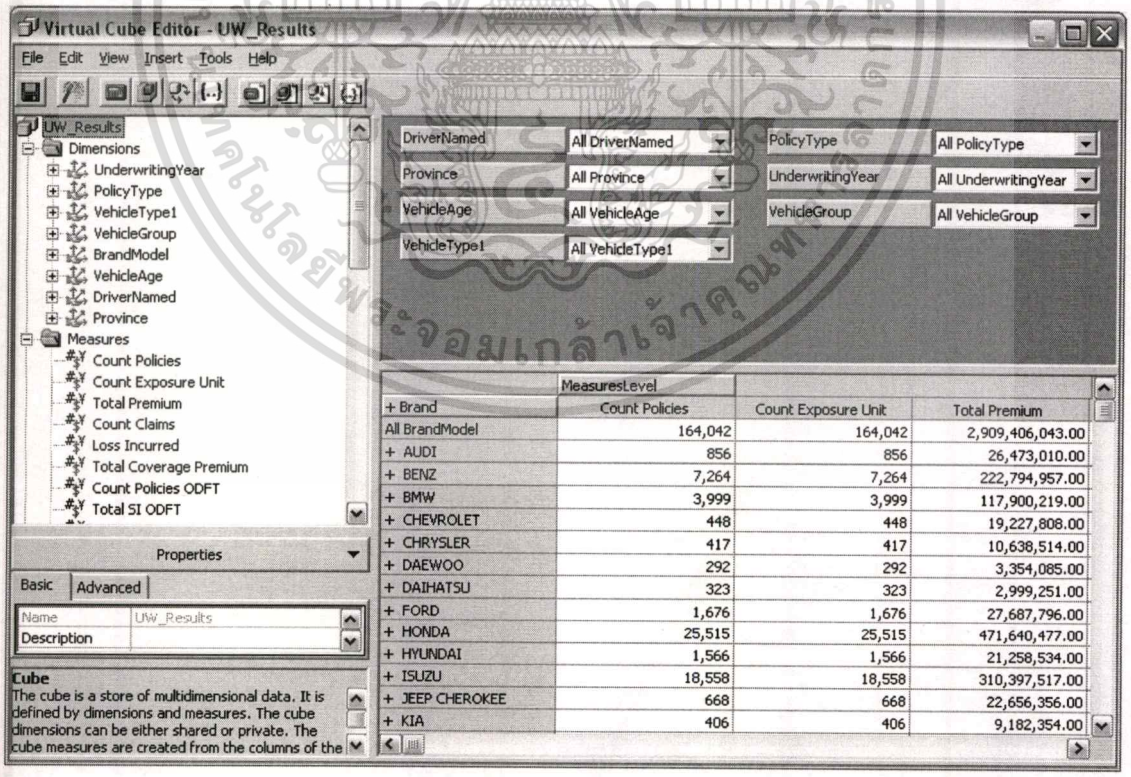
รูปที่ 4-16 แสดงการสร้าง Virtual Cube (ขั้นตอนที่ 2)



รูปที่ 4-17 แสดงการสร้าง Virtual Cube (ขั้นตอนที่ 3)



รูปที่ 4-18 แสดงการสร้าง Virtual Cube (ขั้นตอนที่ 4)



รูปที่ 4-19 แสดงการสร้าง Virtual Cube (ขั้นตอนที่ 5)

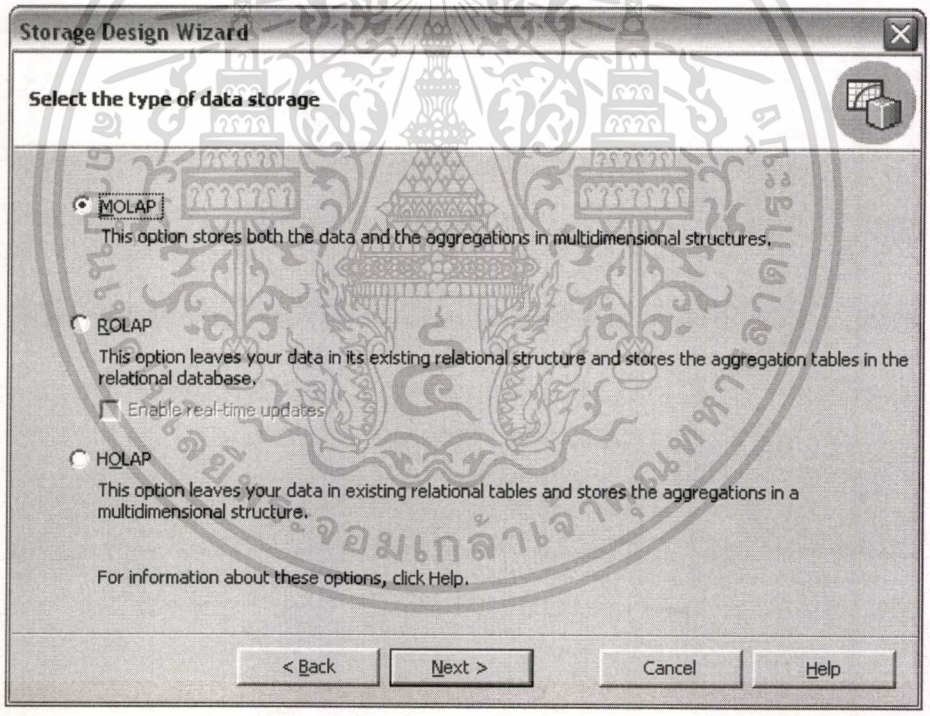
4.3.4 การออกแบบที่จัดเก็บข้อมูล (Storage Design)

โดยการออกแบบวิธีการจัดเก็บข้อมูล จะเป็นขั้นตอนที่ทำให้ผู้ออกแบบระบบสามารถกำหนดปัจจัยที่ช่วยในการเพิ่มประสิทธิภาพ โดยจะทำการเปรียบเทียบระหว่างประสิทธิภาพความเร็วในการเข้าถึงข้อมูลและพื้นที่จัดเก็บข้อมูลที่ มีขั้นตอนดังนี้

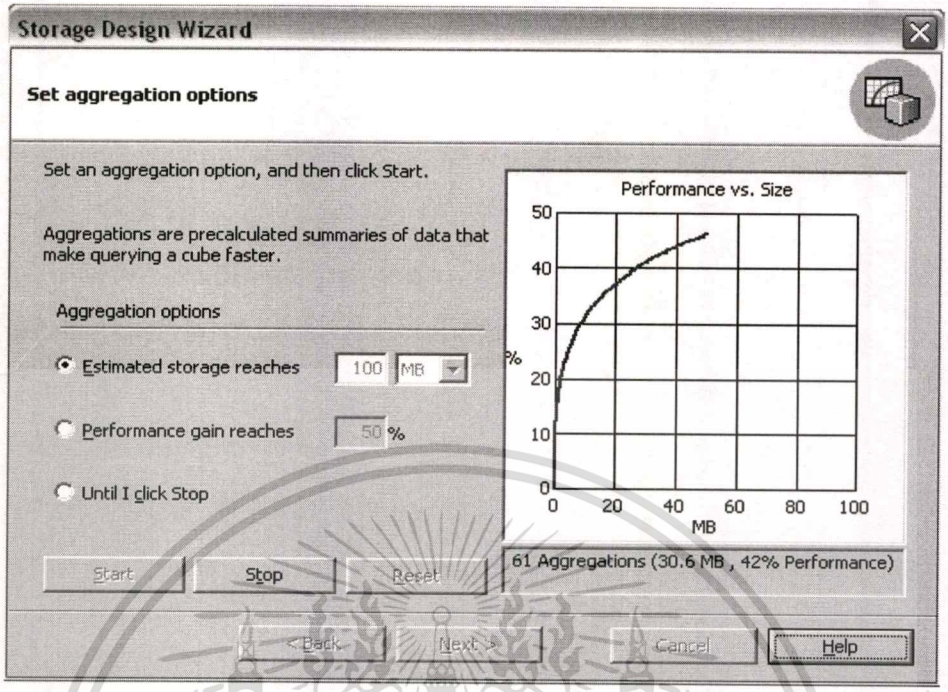
ขั้นตอนที่ 1: เลือกประเภทของวิธีการบันทึกข้อมูลได้แก่ MOLAP, ROLAP, HOLAP ซึ่งแต่ละวิธีการจัดเก็บจะแตกต่างกันในเรื่องของแหล่งที่จัดเก็บข้อมูล และการรวมข้อมูลกับ Aggregates

ขั้นตอนที่ 2: กำหนดความพื้นที่สำหรับบันทึกข้อมูล หรือกำหนดระดับความเร็วใจการเข้าถึงข้อมูล เพื่อเปรียบเทียบความเร็จและพื้นที่จัดเก็บข้อมูลที่เหมาะสม

ขั้นตอนที่ 3: ทำการ Reprocess เพิ่มเริ่มดำเนินการประมวลผลข้อมูลใหม่ทั้งหมด



รูปที่ 4-20 แสดงขั้นตอนการออกแบบพื้นที่จัดเก็บข้อมูล (ขั้นตอนที่ 1)



รูปที่ 4-21 แสดงขั้นตอนการออกแบบพื้นที่จัดเก็บข้อมูล (ขั้นตอนที่ 2)

4.3.5 การประมวลผลข้อมูล (Process or Reprocess)

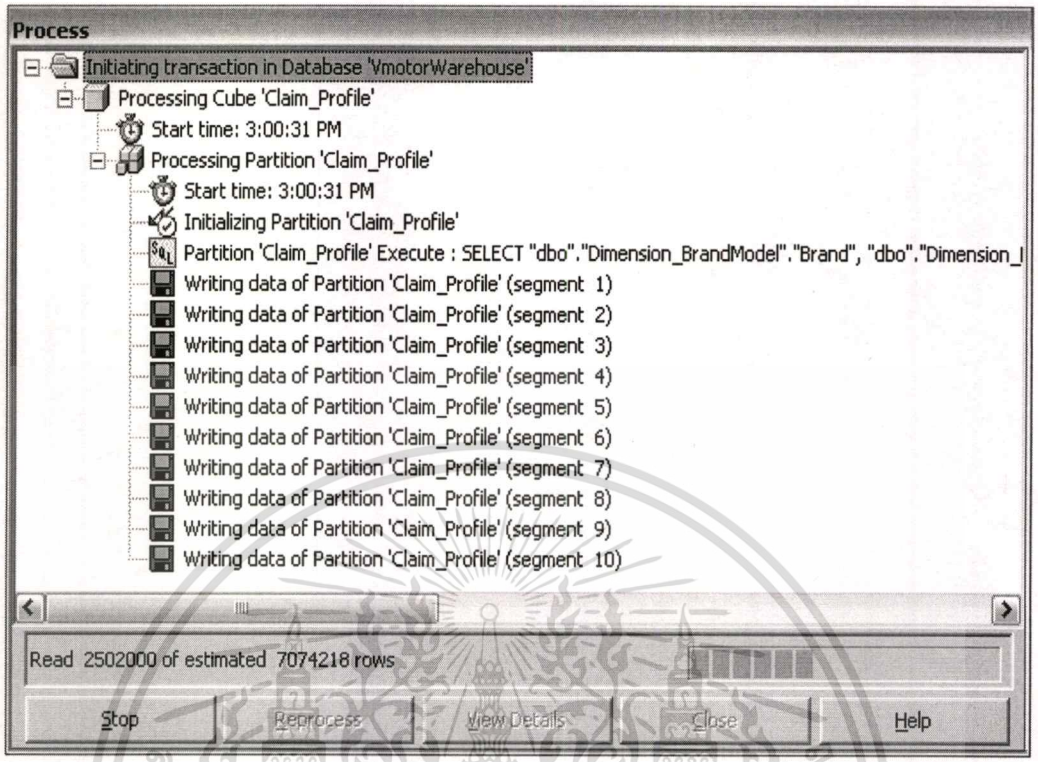
สำหรับคูป, ไดเมนชั่นที่ได้ออกแบบไปนั้น จะยังไม่สามารถแสดงข้อมูลได้จนกว่าจะทำการประมวลผลข้อมูล(Process / Reprocess) เพื่อทำการบันทึกข้อมูล โดยสามารถเลือกประมวลผลเฉพาะ คูป หรือเลือกประมวลผลทั้งหมดได้ตามรูปที่ 4-22

ในระบบประกันวินาศภัยรถยนต์ภาคสมัครใจได้สร้างไว้ 3 กลุ่ม คือ

1) Policy Profile (Policy_Profile) เป็นข้อมูลของการรับประกันภัย ซึ่งจะประกอบด้วยข้อมูลเกี่ยวกับประเภทกรมธรรม์, ประเภทรถยนต์ ซึ่งมีตัววัดค่า เช่น เบี้ยประกันภัย, ความคุ้มครอง เป็นต้น

2) Claim Profile (Claim_Profile) เป็นข้อมูลของการจ่ายสินไหมทดแทน ซึ่งจะประกอบด้วยข้อมูลเกี่ยวกับประเภทกรมธรรม์, ประเภทรถยนต์ ซึ่งมีตัววัดค่า เช่น วงเงินคุ้มครอง, จำนวนสินไหมต่อตัวทรัพย์, ความเสียหายต่อร่างกาย, ความเสียหายจากการลักทรัพย์และไฟไหม้, จำนวนสินไหมค้างจ่าย เป็นต้น

3) Under Writing เป็นข้อมูลผลการรับประกันภัย ซึ่งจะเป็นการสรุปข้อมูลที่เกิดของ Policy_Profile และข้อมูลใน Claim_Profile ซึ่งจะเป็นการสร้างคูปประเภท Virtual Cube



รูปที่ 4-22 แสดงหน้าจอการประมวลผลข้อมูลของคูปทั้งหมด

ตารางที่ 4-3 แสดงตัวอย่างของการออกแบบ "Claim_Profile" Cube Schema

Measure	Measure	Dimension Name	Description
UYr		UnderwritingYear	ปีรับประกันภัย
InsNo			รหัสบริษัทประกันภัย
PassedIns			สถานะของบริษัทประกันภัยผ่านเกณฑ์
FleetNo			เลขที่กลุ่ม
Fleet_Count			จำนวนในกลุ่ม
PolicyNo	Count		เลขที่กรมธรรม์
FleetSeq			ลำดับกลุ่ม
AcctYr			ปีบัญชี
AcctMth			เดือนบัญชี
IssueDate			วันที่เริ่มมีผลบังคับ
PolType		PolicyType	ประเภทกรมธรรม์
BeginDate			วันที่เริ่มต้นคุ้มครอง
ExpireDate			วันที่สิ้นสุดคุ้มครอง
ExposureUnit	Sum		หน่วยความเสี่ยง
TotPremPerExpUnit	Sum		เบี้ยรวมต่อหน่วยความเสี่ยง
MainCoverPremPerExpUnit	Sum		เบี้ยความคุ้มครองหลักต่อหน่วยความเสี่ยง
EarnedTotPremThisYr			รายได้รวมของปีนี้
MainCoverPrem			เบี้ยความคุ้มครองหลัก
::			
VehBrand		BrandModel.Brand	ยี่ห้อรถยนต์
VehModel		BrandModel.Model	รุ่นรถยนต์
VehAge		VehicleYear	อายุรถยนต์
Recovery	Sum		จำนวนเงินคืน

4.4 การพัฒนาระบบส่วนการสอบถามและนำเสนอสารสนเทศ

ระบบสอบถามและนำเสนอสารสนเทศนี้เป็นการพัฒนาด้วย Active Server Page (ASP) และให้บริการผ่านเครือข่ายอินทราเน็ต ในการนำเสนอข้อมูลแบบสำหรับโครงสร้างฐานข้อมูลแบบหลายมิติ สามารถที่จะแสดงข้อมูลให้อยู่ได้ทั้งในรูปแบบตาราง (2 มิติ) และการแสดงรายงานลักษณะ Pivot Table ซึ่งมีความสามารถในลักษณะ Dynamic คือการกรองข้อมูล การเปลี่ยนแกนข้อมูล ซึ่งจะนำมาใช้ในโครงการนี้

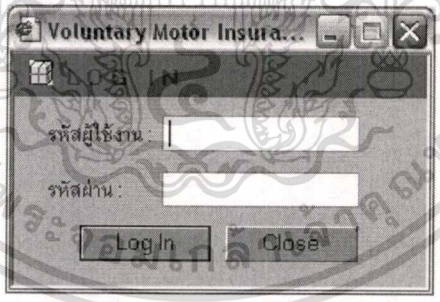
การพัฒนาการสอบถามและการแสดงรายงานสารสนเทศ

ขั้นตอนที่ 1: การล็อกอินเข้าระบบเพื่อระบุสิทธิ์ของผู้ใช้งาน (รูปที่ 2-23)

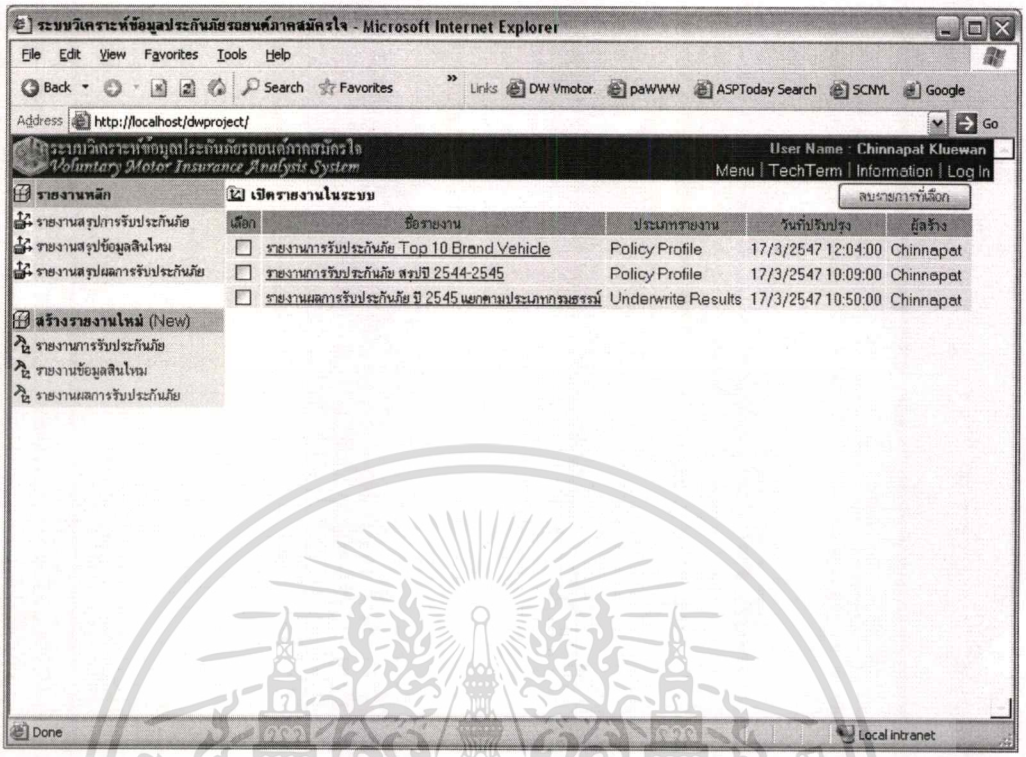
ขั้นตอนที่ 2: หน้าจอแสดงเมนูรายงานที่เป็นลักษณะรายงานมาตรฐาน และรายงานของผู้ใช้งานระบบ (รูปที่ 2-24)

ขั้นตอนที่ 3: เลือกรายงานที่ต้องการ เพื่อทำการออกแบบและทำการบันทึกรายการ

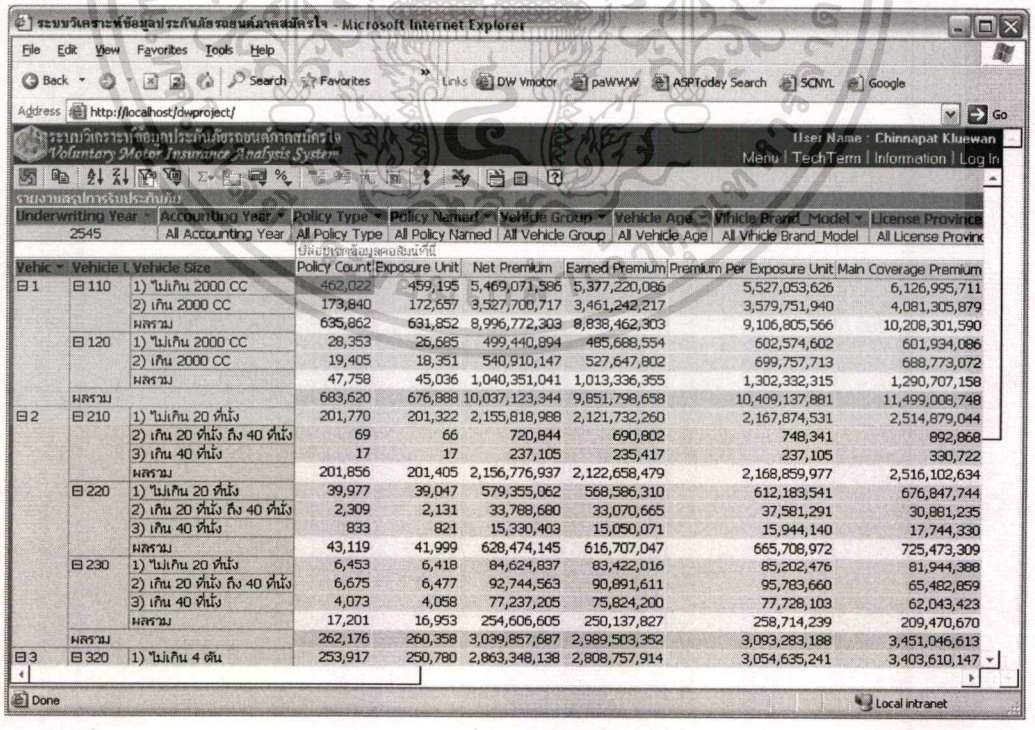
- เลือกรายงานใหม่ หรือเลือกรายงานมาตรฐาน
- ออกแบบหรือปรับแก้ไขรายงานให้อยู่ในรูปแบบที่ต้องการ (รูปที่ 2-25)
- ทำการบันทึกรายงาน โดยตั้งชื่อรายงานใหม่ และกำหนดสิทธิ์ในการเข้าถึงข้อมูล (รูปที่ 2-26)



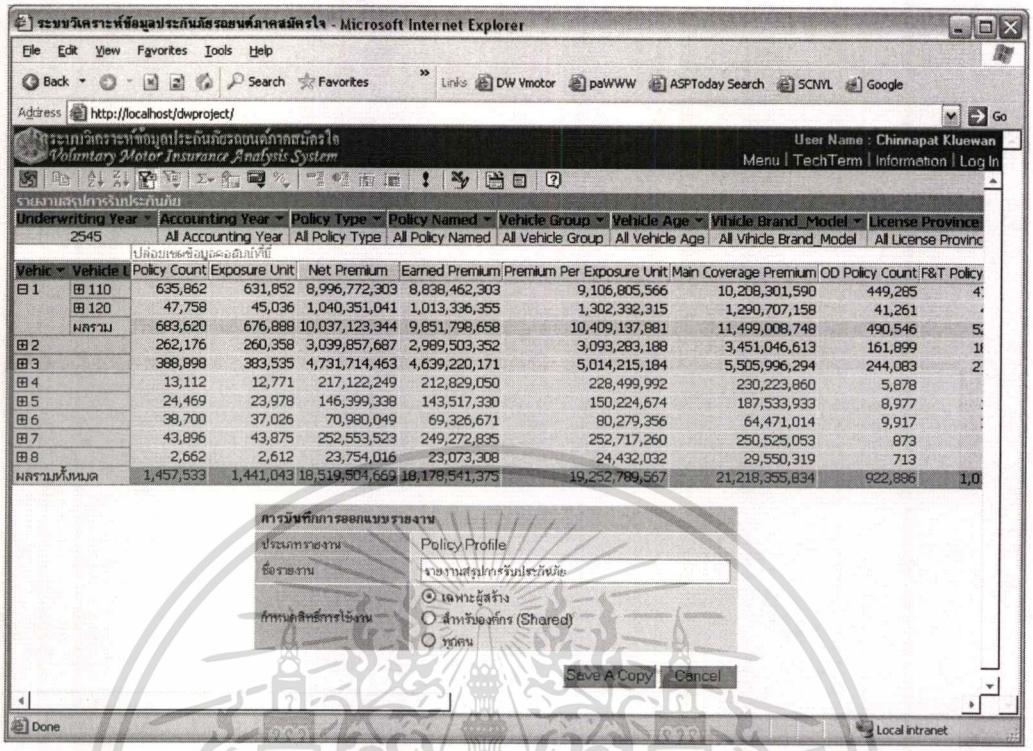
รูปที่ 4-23 แสดงหน้าจอการล็อกอินเข้าใช้งานระบบ



รูปที่ 4-24 แสดงหน้าจอเมนูหลักรายงาน

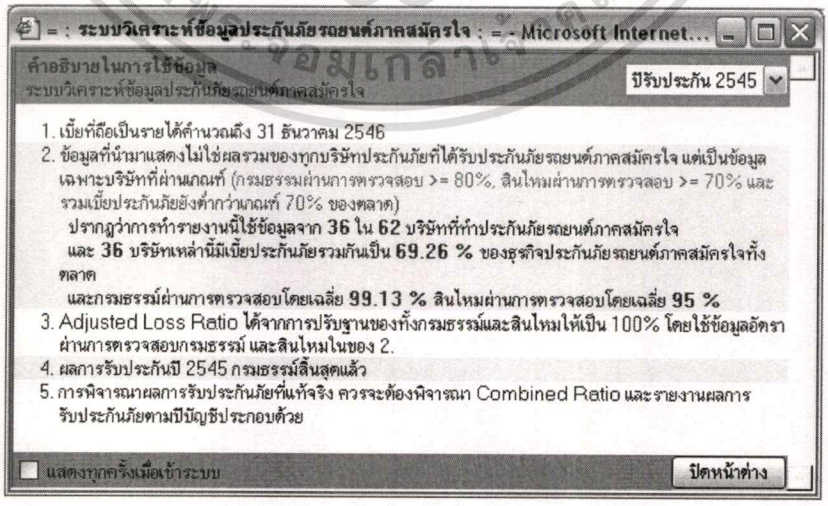


รูปที่ 4-25 แสดงรายงาน “สรุปผลการรับประกันภัย” ที่ออกแบบ

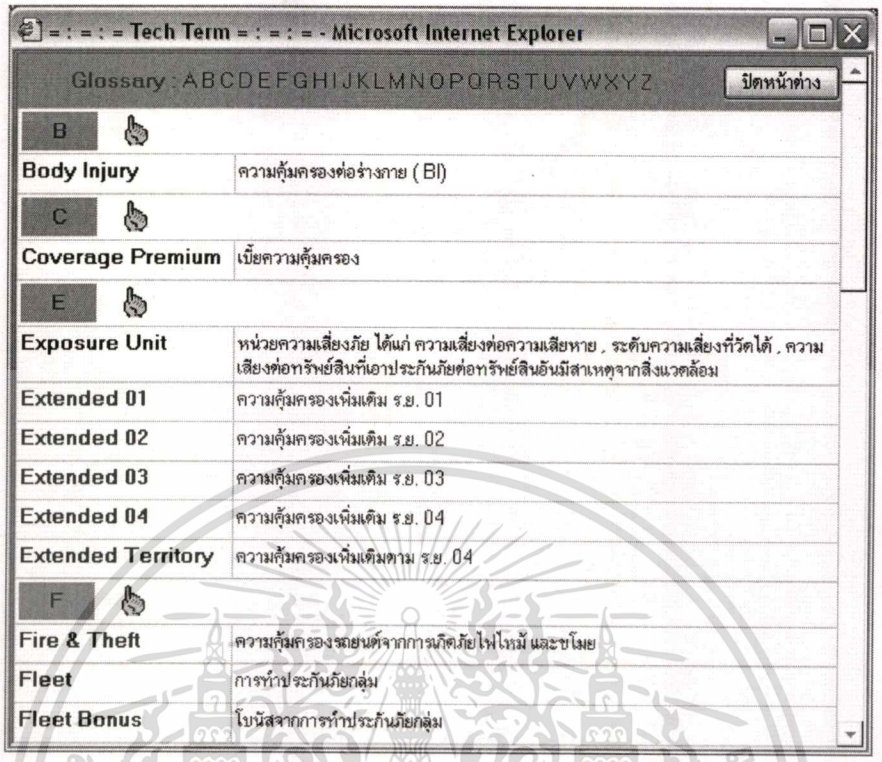


รูปที่ 4-26 แสดงการบันทึกรายงานที่ออกแบบไว้

นอกจากส่วนของระบบที่เป็นรายงานแล้ว จำเป็นจะต้องมีหมายเหตุและคำอธิบายของข้อมูล เพื่อให้ผู้ใช้งานสามารถทราบที่มาของข้อมูล และคำอธิบายสำหรับตัวข้อมูล ดังแสดงรูปที่ 4-27 อธิบายที่มาของข้อมูลและ 4-28 อธิบายคำศัพท์ธุรกิจ



รูปที่ 4-27 แสดงหน้าจอข้อมูลที่ใช้อธิบายที่มาของข้อมูล



รูปที่ 4-28 แสดงหน้าจอคำศัพท์ธุรกิจสำหรับระบบประกันวินาศภัย

บทที่ 5
การทดสอบระบบและสรุปผล

5.1 การทดสอบระบบ

หลังจากการพัฒนาระบบคลังข้อมูลการประกันวินาศภัย ตามขั้นตอนการทำงานในบทที่ 4 แล้ว สามารถทำการศึกษาและทดสอบระบบ เพื่อวัดประสิทธิภาพและความถูกต้องโดยมีรายละเอียดในการทดสอบระบบดังนี้

ที่มาของข้อมูล : ฐานข้อมูลสำหรับดาต้าแวร์เฮาส์ ซึ่งเกิดจากการการเตรียมข้อมูลมาจากฐานข้อมูลที่เป็นทรานเซคชั่น

CUBE : POLICY_PROFILE

จำนวนข้อมูลการรับประกันภัย :	6,099,815	รายการ
การกำหนดค่าของการออกแบบพื้นที่จัดเก็บข้อมูล		
ประสิทธิภาพในการทำงาน	60	%
ความจุของดิสก์ที่ใช้	198	MB
ผลการประมวลผลข้อมูล		
จำนวน Segment ในการแบ่งประมวลผล	14	Segments
เวลาที่ใช้ในการประมวลผล	16 นาที 12 วินาที	

CUBE : CLAIM_PROFILE

จำนวนข้อมูลสินไหมค่าทดแทน :	7,074,218	รายการ
การกำหนดค่าของการออกแบบพื้นที่จัดเก็บข้อมูล		
ประสิทธิภาพในการทำงาน	60	%
ความจุของดิสก์ที่ใช้	220	MB
ผลการประมวลผลข้อมูล		
จำนวน Segment ในการแบ่งประมวลผล	25	Segments
เวลาที่ใช้ในการประมวลผล	19 นาที 18 วินาที	

เปรียบเทียบความเร็วในการการประมวลผลจาก RDBMS กับ OLAP

คำสั่งในการทดสอบเพื่อเปรียบเทียบ

```
-- // หาผลรวมของเบี้ยประกันภัยรวม //--
SELECT    SUM(MainCoverPrem) AS TotalAllPremium
FROM      POLICY_PROFILE
```

การประมวลผลจาก RDBMS

เวลาที่ใช้ในการประมวลผล	1 นาที 25 วินาที
จำนวน CPU ที่ใช้	14,912 CPU ms.

การประมวลผลจาก OLAP

อ้างถึงเวลาในการประมวลผลข้อมูลคูป "POLICY_PROFILE" ซึ่งการทำงานจะเป็นลักษณะ Preprocessing โดยจะทำการคำนวณ Aggregates ต่างๆ ซึ่งซับซ้อนมากกว่าคำสั่งในการหาค่าเฉพาะผลรวมเบี้ยประกันภัย ตามตัวอย่างด้านบน ซึ่งใช้เวลา 16 นาที 12 วินาที ปรากฏว่า เวลาในการคำนวณค่าข้อมูลสามารถแสดงได้ทันที ซึ่งอาจจะใช้เวลาเพียงเล็กน้อยในการหาผลรวมเฉพาะค่าที่อยู่ใน Dimensions ที่กำหนด/เงื่อนไขที่กรองค่า (เวลาที่ใช้เพียง 1-2 วินาที)

หมายเหตุ : ในการทดสอบวัดประสิทธิภาพการประมวลผลได้ใช้ SQL Profiler เป็น Tool ในการทดสอบการประมวลผลคำสั่ง SQL

5.2 การสรุปผล

จากผลการทดสอบการประมวลผลข้อมูลประกันวินาศภัยรถยนต์ภาคสมัครใจ พบว่าประสิทธิภาพในการประมวลผลและความเร็วในการทำงานของคูปขึ้นอยู่กับการออกแบบพื้นที่จัดเก็บข้อมูลลงดิสก์ ซึ่งเป็นปัจจัยสำคัญ อีกทั้งในเรื่องของความถูกต้องของข้อมูลในการนำเสนอเป็นเป้าหมายหลักของการพัฒนาระบบดาต้าแวร์เฮาส์ ดังนั้นการลดความข้อผิดพลาดในการประมวลผลจะต้องกำหนดกระบวนการคลีนนิ่งข้อมูลให้เหมาะสม เพื่อป้องกันการสูญหายของข้อมูลดังกล่าวด้วย และผลที่ได้จากการพัฒนาระบบจะเป็นแนวทางในการพัฒนาระบบดาต้าแวร์เฮาส์สำหรับระบบงานประกันวินาศภัยด้านอื่นๆ ให้มีประสิทธิภาพต่อไป

บรรณานุกรม

Rakesh Agrawal. 2002. **Modeling Multidimensional Databases** [Online]. Available: http://infolab.usc.edu/csci599/Fall2002/paper/I3_agrawal95modeling.pdf

Robert Fenk. 1999. **MDX – Multidimensional Expressions**. [Online]. Available: http://www3.informatik.tu-muenchen.de/lehre/SS2001/DAWA-bayer/DWH-Ch10-1.pdf

David Willson. 2001. **The Microsoft Data Warehousing Framework**. [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnolap/html/intromdx.asp

Carl Nolan. 1999. **Introduction to Multidimensional Expressions (MDX)**. [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnolap/html/intromdx.asp

ประวัติผู้เขียน

ชื่อผู้เขียน นายชินพัฒน์ เคลือวัลย์ นักศึกษาปริญญาโท สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วุฒิการศึกษาระดับปริญญาตรีจาก มหาวิทยาลัยมหิดล คณะวิทยาศาสตร์ ภาควิชาวิทยาการคอมพิวเตอร์ ปี พ.ศ.2545 และมหาวิทยาลัยเกษมบัณฑิต คณะบริหารธุรกิจ ภาควิชาคอมพิวเตอร์ธุรกิจ ปี พ.ศ. 2539

ปัจจุบันมีตำแหน่งเป็นนักวิเคราะห์ระบบ บริษัท ไทยอินชัวเรอส์ ดาต้าเน็ต จำกัด ซึ่งออกแบบและพัฒนาโครงการประกันภัยเอื้ออาทร และระบบประมวลผลข้อมูลประกันภัยรถยนต์ภาคสมัครใจ