การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

DATA WAREHOUSE

นายตุวุติ เวงไธสง

นายธวัช พรมฮวด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

DATA WAREHOUSE

โดย

นายจตุวุฒิ เวงไชสง รหัสประจำตัว 44015319

นายธวัช พรมอวด รหัสประจำตัว 44015328

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

DATA WAREHOUSE

คณะผู้จัดทำ นายจตุวุฒิ เวงไธสง รหัสประจำตัว 44015319

นายธวัช พรมฮวด รหัสประจำตัว 44015328

.....อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

นายจตุวุฒิ เวงไธสง 44015319
นายธวัช พรมฮวด 44015328
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

โครงงานนี้เป็นการศึกษาระบบฐานข้อมูลดาต้าแวร์เฮาส์ (Data Warehouse) ซึ่งเป็นระบบฐานข้อมูลอีกรูปแบบหนึ่งที่แตกต่างไปจากฐานข้อมูลทั่วไป (Relational Database) ระบบฐานข้อมูลดาต้าแวร์เฮาส์มีการจัดเก็บข้อมูลในลักษณะมัลติไดเมนชันแนล (Multidimensional) โครงงานนี้ใช้ Microsoft SQL Server 2000 Analysis Service มาทดลองสร้างระบบดาต้าแวร์เฮาส์ เมื่อได้ระบบดาต้าแวร์เฮาส์แล้วก็นำมาใช้งานกับ MDX (Multidimensional Expression) และเปรียบเทียบการใช้งาน นอกจากนี้ยังพัฒนาโปรแกรมประยุกต์ไคลเอ็นต์ขึ้นมา เพื่อให้ผู้ใช้เข้าถึงฐานข้อมูลดาต้าแวร์เฮาส์ได้สะดวกและยืดหยุ่นมากยิ่งขึ้น

Data Warehouse

Jatuwut Wengthaisong 44015319
Tawat Promhoud 44015328
Dr. Worrawat Limpoka Advisor
Academic Year 2003

Abstract

This paper has been developed the Data Warehouse which it is a newest database system, which it is more different from Relational Database. It is sterns data in Multidimensional data models. This paper has been used Microsoft SQL Server 2000 Analysis Server to create Data Warehouse System, and than brig it to work with MDX (Multidimensional Expression), finally compare the result. Moreover we have developed client application that user can be more convenient and flexible to access the Data Warehouse database system.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีด้วยความช่วยเหลือ ร่วมมือและสนับสนุนจากหลายฝ่าย เป็นอย่างดี จึงใคร่ขอขอบพระคุณ บิดา มารดาและพี่น้องทุกๆ คน ผู้ส่งเสริมด้านการศึกษา เป็นกำลังใจ และคอยให้ความสนับสนุนในทุกๆ ด้านมาโดยตลอด ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาในการทำปริญญานิพนธ์ที่กรุณาให้คำแนะนำและชี้แนะแนวทางในการดำเนินงาน ให้ความเอาใจใส่และช่วยเหลือเสมอมา อาจารย์ทุกๆ ท่านที่ประสิทธิ์ประสาทวิชาความรู้ต่างๆ

สุดท้ายขอมอบคุณความดีจากปริญญานิพนธ์ฉบับนี้ให้แด่ คุณบิดา มารดาและครูอาจารย์ผู้มีพระคุณทุกท่าน

จตุวุฒิ เวงไธสง

ธวัช พรมยวด

สารบัญ

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

สารบัญภาพ

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

สารบัญตาราง

หน้าที่

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันนี้การใช้ข้อมูลเป็นสิ่งจำเป็นอย่างยิ่งผู้ที่สามารถนำข้อมูลมาใช้ได้อย่างมีประสิทธิภาพมากกว่าและรวดเร็วกว่าก็จะส่งผลให้ก่อให้เกิดข้อได้เปรียบเหนือผู้อื่น สิ่งที่จำเป็นต้องทำก่อนนำข้อมูลมาใช้นั่นคือการเก็บข้อมูล การเก็บข้อมูลจะมีการนำระบบฐานข้อมูลเข้ามาใช้ โดยระบบฐานข้อมูลที่ใช้นี้จะมีลักษณะเป็นระบบฐานข้อมูลสำหรับงานประจำวัน (OLTP) ลักษณะที่สำคัญของระบบฐานข้อมูลดังกล่าวคือ สนับสนุนให้ผู้ใช้หลายคนให้สามารถเข้ามาทำงานในเวลาเดียวกันได้ การเก็บข้อมูลไม่ได้เก็บสัมพันธ์กับเวลาโดยจะเก็บข้อมูลล่าสุดเสมอ มีโครงสร้างที่ซับซ้อน มักถูกนำไปใช้กับงานทรานแซคชัน (transaction) ถูกนำไปใช้กับงานที่ทำในลักษณะวันต่อวัน

การใช้ข้อมูลที่สำคัญอย่างหนึ่งสำหรับการดำเนินธุรกิจในทุกวันนี้คือ การเรียกใช้ข้อมูลเพื่อการตัดสินใจ โดยจำเป็นต้องทำได้อย่างรวดเร็วและมีประสิทธิภาพ ดังนั้นฐานข้อมูลประจำวันจึงไม่เหมาะสมกับการเรียกใช้ข้อมูลในลักษณะนี้ด้วยสาเหตุจากการคิวรี (query) เพื่อการตัดสินใจมักจะเป็นการคิวรีแบบเฉพาะกิจ (ad hoc query) ผู้เรียกใช้จำเป็นต้องมีความรู้ทางด้านเทคนิค การคิวรีต้องใช้คำสั่งที่ซับซ้อนซึ่งอาจจะทำให้ประสิทธิภาพของระบบต่ำลงมากหรือไม่อาจจะคาดการณ์ได้ ทำให้ไม่เหมาะกับการทำการวิเคราะห์แบบออนไลน์ (online analytical) การเก็บข้อมูลในระบบฐานข้อมูลประจำวันไม่ได้มีการเก็บข้อมูลย้อนหลัง (historical Data) แต่บางครั้งการตัดสินใจจำเป็นต้องใช้ข้อมูลที่ผ่านมาเพื่อช่วยในการคาดคะเนแนวโน้มที่จะเป็นไปได้ในอนาคต

ดังนั้นจึงได้ทำการศึกษาระบบฐานข้อมูลในรูปแบบอื่น ที่สามารถนำมาใช้สนับสนุนการตัดสินใจได้ ซึ่งก็คือระบบฐานข้อมูลคลังข้อมูลโดยที่ระบบฐานข้อมูลคลังข้อมูลนั้นได้ออกแบบมามีจุดประสงค์เพื่อจัดการกับข้อมูลและจัดสรรให้เกิดข้อมูลที่รองรับสำหรับการวิเคราะห์และตัดสินใจ โดยมีลักษณะดังนี้ คือ รวมข้อมูลจากหลายๆ แหล่งข้อมูล (Heterogeneous data source) ให้อยู่ในรูปแบบเดียวกัน มีการจัดการโครงสร้างของข้อมูลให้มีประสิทธิภาพต่อการคิวรีเพื่อการวิเคราะห์มากกว่าระบบฐานข้อมูลประจำวัน มีการเก็บข้อมูลย้อนหลังเพื่อช่วยให้การตัดสินใจมีความแม่นยำมากขึ้น

1.2 วัตถุประสงค์ของการทำโครงงาน

1. ศึกษาระบบฐานข้อมูลคลังข้อมูล คืออะไร มีลักษณะอย่างไร นำมาใช้ประโยชน์ได้อย่างไร
2. ศึกษาวิธีการออกแบบระบบฐานข้อมูลคลังข้อมูล โดยใช้โมเดลสกีมาแบบดาว (Star schema) และสโนว์เฟลกสกีมา (Snowflake schema)
3. ศึกษาระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 Analysis Service และนำมาสร้างฐานข้อมูลคลังข้อมูลโดยใช้ฟังก์ชันการทำงานมีมาให้ใน Microsoft SQL Server 2000 Analysis Service

4. ศึกษาโครงสร้างการทำงานและการใช้งาน MDX (Multidimensional Expression)
5. เปรียบเทียบการทำงานของ Microsoft SQL Server 2000 Analysis Service กับMDX
6. พัฒนาทูลส์ที่ใช้สำหรับคิวรีระบบฐานข้อมูลคาต้าแวร์เฮาส์ในลักษณะ Client - Server

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะมุ่งเน้นไปที่การศึกษาระบบฐานข้อมูลคาต้าแวร์เฮาส์ว่าคืออะไร มีประโยชน์อย่างไร และสามารถนำมาใช้ได้อย่างไร จากนั้นก็ทดลองสร้างระบบฐานข้อมูลคาต้าแวร์เฮาส์โดยใช้ฟังก์ชันการทำงานที่มีมาให้ใน Microsoft SQL Server 2000 Analysis Service

เมื่อได้สร้างฐานข้อมูลคาต้าแวร์เฮาส์ขึ้นมาแล้วก็จะนำไปใช้กับ MDX เพื่อทำการคิวรีและทำการเปรียบเทียบการใช้งานกับการทำงานของ Microsoft SQL Server 2000 Analysis Service หลังจากนั้นได้ทำการพัฒนาโปรแกรมประยุกต์ทางฝั่งไคลเอ็นต์ขึ้นมา เพื่อให้ผู้ใช้สามารถเรียกคิวรีระบบฐานข้อมูลคาต้าแวร์เฮาส์ได้จาก Client

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาและทำความเข้าใจเกี่ยวกับระบบฐานข้อมูลคาต้าแวร์เฮาส์ ประโยชน์ของฐานข้อมูลคาต้าแวร์เฮาส์ที่เหนือกว่าฐานข้อมูลประจำวัน และความเหมาะสมกับการนำไปใช้
2. ศึกษาทฤษฎีที่เกี่ยวข้องกับระบบฐานข้อมูลคาต้าแวร์เฮาส์ โครงสร้างของข้อมูล รูปแบบการเก็บข้อมูล รวมไปถึงโมเดลการออกแบบระบบคาต้าแวร์เฮาส์
3. ศึกษาวิธีการใช้งาน วิธีการติดตั้ง ของซอฟต์แวร์ Microsoft SQL Server 2000 Analysis Service และศึกษาฟังก์ชันการทำงานที่ใช้สำหรับทำคาต้าแวร์เฮาส์บนโปรดักส์นี้
4. ศึกษาโครงสร้างการทำงานและการใช้งาน MDX
5. เปรียบเทียบการทำงานของ Microsoft SQL Server 2000 Analysis Service กับ MDX
6. พัฒนาโปรแกรมประยุกต์ทางฝั่งไคลเอ็นต์ที่ทำงานบนแลนขึ้นมา โดยได้เลือกใช้ Microsoft Visual Basic 6.0

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ปัจจุบันนี้จะเห็นได้ว่าหลายๆ องค์กรมักจะมีการทำระบบคาต้าแวร์เฮาส์ของแต่ละองค์กร เนื่องจากว่าการตัดสินใจในทางธุรกิจเป็นที่สำคัญและจำเป็นต้องทำให้ได้อย่างรวดเร็วและถูกต้อง การที่จะทำเช่นนั้นได้จำเป็นต้องมีข้อมูลสนับสนุน ซึ่งระบบฐานข้อมูลคาต้าแวร์เฮาส์สามารถนำมาใช้ในส่วนของการเตรียมข้อมูลสนับสนุนการตัดสินใจได้ และในโครงงานนี้ได้อธิบายถึงทฤษฎีและลักษณะของระบบฐานข้อมูลคาต้าแวร์เฮาส์ซึ่งคาดว่าจะสามารถนำไปใช้ให้เป็นประโยชน์ นอกจากนี้โครงงานนี้ยังได้มีการพัฒนาโปรแกรมประยุกต์ทางฝั่งไคลเอ็นต์ขึ้นมา เพื่อที่ว่าจะได้สามารถนำไปใช้ได้กับระบบคาต้าแวร์เฮาส์อื่นๆ ที่สร้างบน Microsoft SQL Server 2000 Analysis Service ได้อย่างสะดวก

บทที่ 2
ทฤษฎีเบื้องต้น

2.1 นิยามของดาต้าแวร์เฮาส์

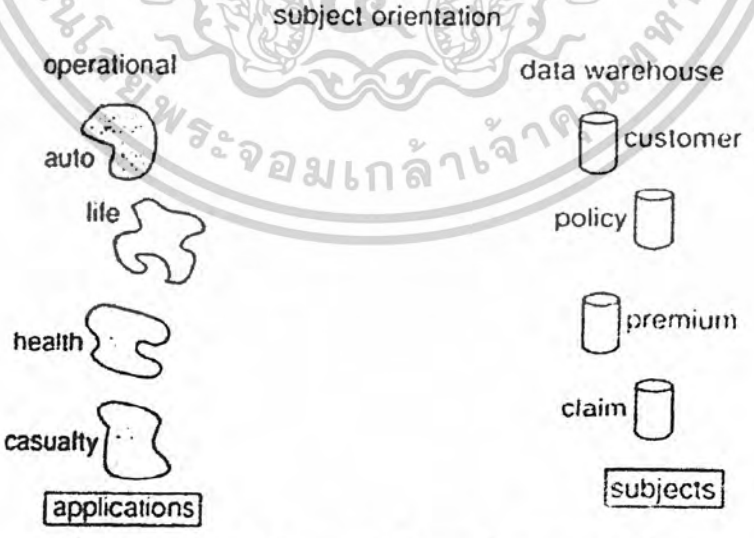
ดาต้าแวร์เฮาส์คือ การรวบรวมข้อมูลจากฐานข้อมูลของส่วนปฏิบัติงาน (Operational Database) หลายๆ รูปแบบหรืออาจจะมาจากแหล่งข้อมูลที่สำคัญและจำเป็นอื่นๆ มาทำการแปลงหรือสรุปให้อยู่ในรูปของฐานข้อมูลที่มีรูปแบบเหมาะสมต่อการใช้ในการวิเคราะห์ การเก็บรวบรวมและการนำกลับมาใช้ทำให้ได้เป็นแหล่งรวมของข้อมูลที่อยู่ในความสนใจของผู้ใช้ เพื่อใช้ประกอบในการตัดสินใจ ใช้เป็นข้อมูลทางธุรกิจ การวางแผน หรือเป็นข้อมูลสำหรับผู้บริหาร ซึ่งจะช่วยให้สามารถทำการตัดสินใจได้อย่างถูกต้องโดยง่าย และรองรับข้อมูลจำนวนมากได้ เป็นการนำเสนอแนวทางในการเข้าถึงข้อมูลในองค์กรได้อย่างมีประสิทธิภาพ

2.2 ลักษณะที่สำคัญของดาต้าแวร์เฮาส์

ดาต้าแวร์เฮาส์เป็นพื้นฐานของระบบสนับสนุนการตัดสินใจ นั่นคือดาต้าแวร์เฮาส์เป็นฐานข้อมูลที่ช่วยในกระบวนการตัดสินใจ โดยฐานข้อมูลดังกล่าวจะต้องมีลักษณะดังต่อไปนี้

2.2.1 เก็บข้อมูลตามหัวข้อ (Subject Oriented)

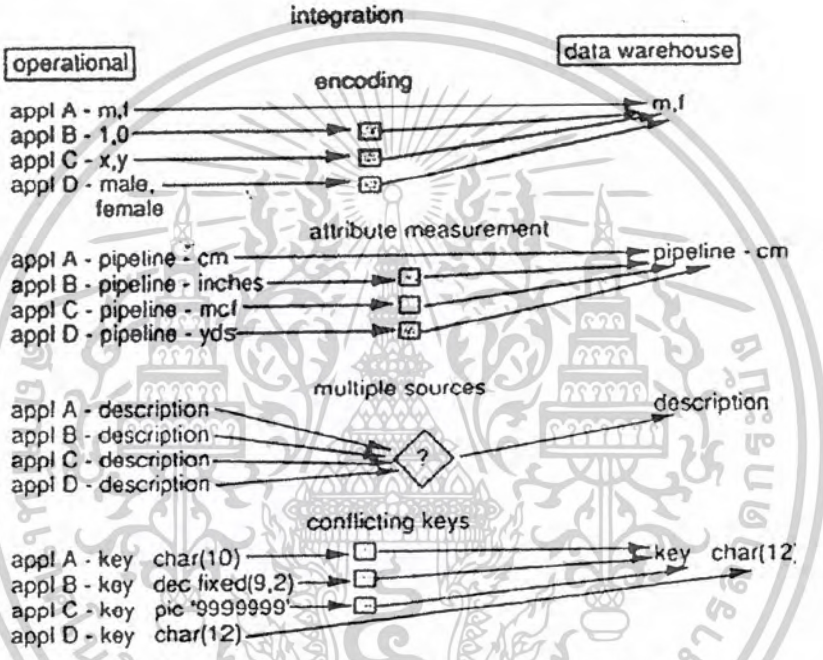
ตามปกติถ้าเป็นงานในระดับโอเปอเรชัน (Operational) ข้อมูลจะถูกจัดเก็บมาจากแต่ละแอพพลิเคชันขององค์กร เช่น ถ้าเป็นข้อมูลของบริษัทประกันภัย ข้อมูลที่ได้จากแอพพลิเคชันจะเก็บตามประวัติ ข้อมูลสุขภาพ การดำเนินชีวิต ฯลฯ ในขณะที่หัวข้อหลักในการเก็บข้อมูลขององค์กรประกอบด้วย ข้อมูลลูกค้านโยบายสิทธิพิเศษและการเอาประกันเป็นต้น แสดงความแตกต่างนี้ได้ดังรูป



รูปที่ 2-1 แสดงการเก็บข้อมูลตามหัวข้อและเก็บตามแอพพลิเคชัน

2.2.2 มีการรวบรวมข้อมูลเข้าด้วยกัน (Integrated)

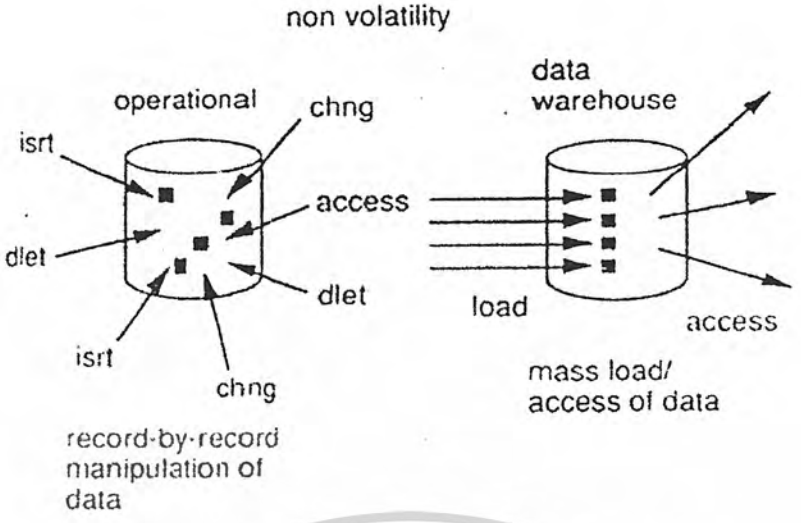
การรวบรวมข้อมูลเข้าด้วยกันเป็นลักษณะหนึ่งที่สำคัญมากของคาต้าแวร์เฮาส์ แสดงได้ดังรูปที่ 2-2 โดยการรวบรวมข้อมูลนี้เกิดขึ้นในขณะที่มีการส่งข้อมูลจากงานโอเปอเรชัน (ซึ่งการจัดเก็บข้อมูลทำตามแต่ละแอพพลิเคชัน) มายังส่วนของคาต้าแวร์เฮาส์ เนื่องจากการออกแบบงานในระดับแอพพลิเคชันได้มีขึ้นอยู่ตลอดเวลา ทำให้ขาดการปรับเปลี่ยนรหัสข้อมูล ชื่อข้อมูลที่ใช้ แอททริบิวต์ (Attribute) ที่ใช้งาน หน่วยการวัดและอื่นๆ ของข้อมูลในแต่ละแอพพลิเคชันให้สอดคล้องกัน ทำให้ข้อมูลที่เข้ามายังคาต้าแวร์เฮาส์มีความสับสนจากแต่ละแอพพลิเคชันที่เข้ามา ตัวอย่างเช่น รหัสของเพศที่เข้ามายังคาต้าแวร์เฮาส์อาจเข้ามาในรูปแบบของ m/f หรือ 1/0 หรือ x/y และอื่นๆ ได้อีกมากมาย รหัสดังกล่าวจึงจำเป็นต้องถูกปรับเปลี่ยนให้อยู่ในรูปแบบเดียวกันก่อนเก็บในคาต้าแวร์เฮาส์



รูปที่ 2-2 แสดงการรวมข้อมูลก่อนเข้าสู่คาต้าแวร์เฮาส์

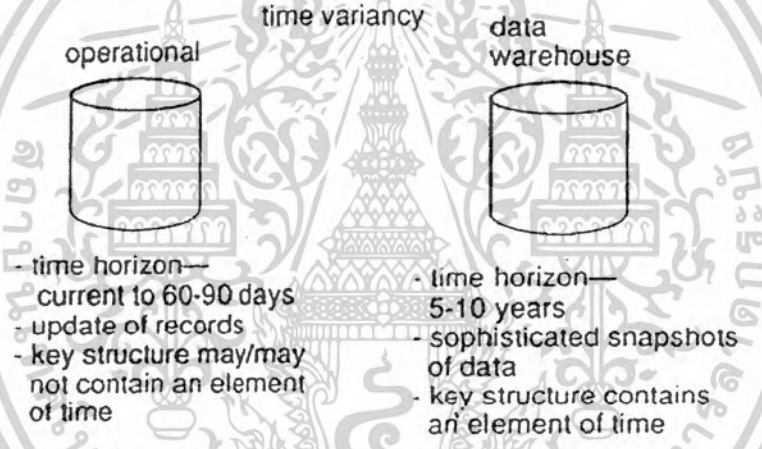
2.2.3 ข้อมูลจะไม่เปลี่ยนแปลงได้ง่ายๆ (Nonvolatile)

ลักษณะของข้อมูลในคาต้าแวร์เฮาส์จะไม่มีการเปลี่ยนแปลงบ่อยๆ ในขณะที่ข้อมูลในส่วนของโอเปอเรชันจะมีการเข้าถึงข้อมูลและจัดการกับข้อมูลนั้นได้ครั้งละ 1 เรคอร์ด การอัพเดตจะเกิดขึ้นที่ตัวข้อมูลแต่ในคาต้าแวร์เฮาส์จะมีรูปแบบการจัดเก็บข้อมูลที่แตกต่างออกไป เนื่องจากข้อมูลจะถูกโหลดเข้ามาและถูกเข้ามาใช้งานได้ แต่การอัพเดตข้อมูล (การเปลี่ยนแปลงที่ตัวเนื้อข้อมูลจริงๆ) จะไม่เกิดขึ้นภายในคาต้าแวร์เฮาส์



รูปที่ 2-3 แสดง non volatility

2.2.4 มีเวลาเป็นองค์ประกอบ (Time variant)



รูปที่ 2-4 แสดงลักษณะของคาต้าแวร์เฮาส์ซึ่งเก็บข้อมูลโดยมีเวลาเป็นองค์ประกอบ

ตามรูปที่ 2-4 แสดงให้เห็นว่าเวลาที่ใช้ในการเก็บข้อมูลของคาต้าแวร์เฮาส์ จะนานกว่าที่เก็บในส่วนของงานโอเปอเรชัน ซึ่งโดยทั่วไปแล้วภายในคาต้าแวร์เฮาส์จะเก็บข้อมูลอยู่ในช่วง 5-10 ปี ในขณะที่ฐานข้อมูลของงานโอเปอเรชันจะเก็บค่าข้อมูลในขณะนั้น (current value) ในทางกลับกันคาต้าแวร์เฮาส์จะเก็บข้อมูลในหลายๆ ช่วงเวลา

โครงสร้างของข้อมูลในงานโอเปอเรชันไม่จำเป็นจะต้องประกอบด้วยค่าที่เกี่ยวกับเวลา เช่น ปี เดือน วัน ในขณะที่โครงสร้างของคาต้าแวร์เฮาส์จะต้องประกอบด้วยเวลาเป็นแกนหนึ่งเสมอ คือจะต้องมีองค์ประกอบของเวลาเข้ามารวมอยู่ด้วย

2.3 คุณสมบัติของดาต้าแวร์เฮาส์

คาต้าแวร์เฮาส์ได้รับการออกแบบให้เหมาะกับงานวิเคราะห์ข้อมูลในลักษณะ ข้อมูลรายปี รายสัปดาห์ หรือรายวัน ตัวอย่างเช่น ระบบวิเคราะห์ยอดขาย ยอดเงินเอาประกัน ซึ่งต้องอาศัยข้อมูลในการวิเคราะห์และวางแผนในการบริหารนั้น โดยดาต้าแวร์เฮาส์จะต้องมีคุณสมบัติดังนี้

- โครงสร้างข้อมูลต้องสื่อให้ผู้ใช้เข้าใจ ซึ่งต้องทำการรวบรวมข้อมูลจากตารางต่างๆ ในหลายๆ ที่หรืออาจต้องทำการสรุปข้อมูลเตรียมไว้ก่อน
- ข้อมูลในฐานข้อมูลค่อนข้างจะคงที่ การเปลี่ยนแปลงของฐานข้อมูลจะเกิดขึ้นเฉพาะในขณะที่ทำการถ่ายโอนข้อมูลลงในดาต้าแวร์เฮาส์
- ระบบจะต้องสามารถรองรับข้อมูลจำนวนมากได้
- จะต้องมีการสำรองข้อมูล (Back Up) โดยจะทำทั้งหมดหรือบางส่วนก็ได้

โดยกิจกรรมที่เกิดในคลังข้อมูลสามารถแบ่งได้เป็น 2 ส่วนหลักคือ ส่วนแรกคือระบบสนับสนุนการตัดสินใจ (Decision Support Systems :DSS) ที่มีการเข้าถึงข้อมูลจากผู้ใช้ (end user) และส่วนที่สองคือการทำงานของระบบที่ต้องทำการถ่ายเทและจัดการข้อมูลภายในดาต้าแวร์เฮาส์เอง

2.4 ความแตกต่างของดาต้าแวร์เฮาส์กับระบบฐานข้อมูลประจำวัน

คาต้าแวร์เฮาส์และระบบฐานข้อมูลประจำวันนั้นมีความแตกต่างกันมากมาย ดังต่อไปนี้

2.4.1 ความถูกต้อง (Consistency)

ทั้งงานโอเปอเรชันและดาต้าแวร์เฮาส์ ต่างก็ให้ความสำคัญในเรื่องความถูกต้องของข้อมูล (data consistency) สำหรับงานโอเปอเรชันที่มีการทำทรานแซคชันจำนวนมากๆนั้น สิ่งที่ต้องการคือ การทำทรานแซคชันให้ครบไม่มีสูญหาย ดังนั้นจึงมีความจำเป็นที่ผู้ส่งและผู้รับจะต้องรับรู้และตรวจสอบอยู่ตลอดเวลาว่าขณะนี้มีการทำทรานแซคชันเกิดขึ้นหรือไม่

สำหรับดาต้าแวร์เฮาส์จะไม่สนใจการทำทรานแซคชันแต่ละครั้ง แต่จะสนใจว่า การโหลดข้อมูลใหม่เข้ามานั้นทำเสร็จแล้วหรือยัง นั่นคือสนใจว่าโหลดข้อมูลเข้ามาทั้งหมดและข้อมูลถูกต้องหรือไม่

2.4.2 ทรานแซคชัน (Transaction)

สำหรับระบบงานโอเปอเรชันนั้นในแต่ละวันอาจมีการทำทรานแซคชันเป็นหมื่นเป็นแสนครั้ง ซึ่งการทำทรานแซคชันแต่ละครั้งจะใช้ข้อมูลเพียงเล็กน้อยเท่านั้น อาจเป็นการใช้ข้อมูลเพียง 1 เรคอร์ด

สำหรับดาต้าแวร์เฮาส์ แต่ละวันจะทำเพียงแค่ 1 ทรานแซคชัน ซึ่งการทำทรานแซคชันนี้อาจต้องใช้ข้อมูลเป็นล้านเรคอร์ดเลยก็ได้ ดังนั้นเราจึงเรียกกระบวนการนี้ว่าเป็น production data load แทนสิ่งที่เราสนใจในกระบวนการนี้มีเพียงแค่ production data load เท่านั้น ถ้าการทำ production data load ถูกทำให้หยุดกลางคัน ระบบจะทำการเอาข้อมูลที่เคยมีมาก่อนที่จะทำ production data load มาเขียนทับลงทันที

2.4.3 ระดับของผู้ใช้งาน

สำหรับงานออนไลน์ทรานแซคชัน (OLTP) นั้น ผู้ใช้คือ ผู้ที่ทำงานอยู่ในองค์กรนั้น อาจจะเป็นเจ้าหน้าที่ที่ทำหน้าที่รับออร์เดอร์, เจ้าหน้าที่ที่ทำหน้าที่ตรวจสอบและดูแลเงิน, เจ้าหน้าที่ที่ทำหน้าที่คอยบริการลูกค้าใหม่, เจ้าหน้าที่ที่ทำหน้าที่ป้อนข้อมูลเข้าไป ซึ่งส่วนใหญ่ผู้ใช้เหล่านี้ในช่วงเวลาหนึ่งจะ

ทำงานกับ 1 แอคเคาท์ (account) เท่านั้น ผู้ใช้งานออนไลน์ทรานแซคชันนี้มักจะทำงานที่มีลักษณะเป็นงานซ้ำ ๆ เดิม ๆ รายงานส่วนใหญ่ที่ได้จากการทำงานบนระบบออนไลน์ทรานแซคชันนี้มักจะมีลักษณะเป็นรายลิสต์ของทั้งตารางเลย

สำหรับคาต้าแวร์เฮาส์ผู้ใช้คือผู้ที่ทำหน้าที่คอยดูแลการทำงานของพนักงานในองค์กรเท่านั้น ลักษณะการทำงานเช่น คอยนับจำนวนออร์เดอร์ใหม่ ๆ , หาเหตุว่าทำไมลูกค้าจึงไม่พอใจ, คอยตรวจดูว่ามีข้อมูลอะไรใหม่ ๆ เข้ามาบ้าง, คอยตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูล ผู้ใช้ของคาต้าแวร์เฮาส์จะไม่ทำงานทีละ 1 แอคเคาท์ แต่จะพิจารณาจากแอคเคาท์ทั้งหมดแล้วหาคำตอบที่ต้องการออกมา (ชุดคำตอบ (answer set) ขนาดเล็ก ๆ) และคำถามที่ให้กับคาต้าแวร์เฮาส์ก็สามารถเปลี่ยนแปลงได้เรื่อยๆ ซึ่งอาจไม่ใช่คำถามเดิมที่เคยถามก็ได้

2.4.4 ความเกี่ยวพันกับเวลา (Time Dimension)

ระบบโอเปอเรชันจะทำงานอย่างรวดเร็วและทำทรานแซคชันอย่างสม่ำเสมอ การวัดเวลาใช้หน่วยเป็นนาทีและวินาที สถานะของข้อมูลต่าง ๆ มีการเปลี่ยนแปลงอยู่ตลอดเวลา และความสัมพันธ์ระหว่างเอนติตี้ (entity) ต่าง ๆ ก็เปลี่ยนแปลงไปด้วย

ฐานข้อมูลในงานออนไลน์ทรานแซคชันจะขาดการสนับสนุนหรือการอ้างอิงจากข้อมูลในอดีต แต่ทว่าในคาต้าแวร์เฮาส์มักจะมีคำถามที่ถามถึงการวิเคราะห์ข้อมูลในอดีตด้วย และแม้ว่าเราสามารถเก็บข้อมูลอดีตไว้ในงานออนไลน์ทรานแซคชันได้ แต่ก็เป็นภาระหนักของระบบในการทำให้มองเห็นภาพในอดีต และจะทำให้ยากในการทำทรานแซคชัน (ข้อมูลมีมากขึ้น ก็ต้องใช้เวลามากขึ้นด้วย)

2.5 การสร้างข้อมูล (Data Model)

ความแตกต่างที่สำคัญที่สุดของระบบโอเปอเรชันและคาต้าแวร์เฮาส์ คือ การสร้างข้อมูลหรือคาต้าโมเดล (Data Model) นั่นเอง

งานโอเปอเรชันจะใช้ Entity Relation Modeling คือ การกำจัดความซ้ำซ้อนให้หมดไป เพื่อการทำทรานแซคชันจะได้สามารถทำได้ง่ายและรวดเร็วยิ่งขึ้น สามารถทำทรานแซคชันได้โดยเข้าถึงข้อมูลเพียงตัวเดียวเท่านั้น การทำ entity relation modeling นั้นจะใช้วิธีแยกข้อมูล (Normalized) ออกเป็นตารางเล็ก ๆ และแต่ละตารางสามารถเชื่อมต่อไปยังตารางอื่นได้

ข้อสังเกตเกี่ยวกับคาต้าโมเดลของงานโอเปอเรชันซึ่งจะใช้ entity relation diagram มีดังนี้

- ไดอะแกรมนี้มีลักษณะสมมาตร (symmetric) หมายถึง ทุกๆ ตารางเหมือนกันหมด คือ ไดอะแกรมนี้ไม่สามารถบอกได้ว่าตารางไหนสำคัญกว่าหรือใหญ่กว่า และไม่สามารถบอกได้ว่า ตารางใดบรรจุตัววัดที่เป็นชนิดตัวเลข (numerical measurement) ทางธุรกิจ ซึ่งไดอะแกรมรูปแบบนี้ ผู้ใช้ทำความเข้าใจและจดจำได้ยาก
- ถ้ามี 2 ตารางในไดอะแกรมที่ต้องการที่จะรวมกันก็มีหลายวิธีที่จะทำได้และไม่ว่าจะวิธีไหนก็จะได้คำตอบเดียวกัน ซึ่งเป็นเรื่องยากในการเลือกว่าจะใช้เส้นทาง (path) ใดในการเชื่อม (link) ระหว่างตารางที่ต่างกันเพราะถึงแม้ท้ายสุดจะได้คำตอบเดียวกัน แต่ในระหว่างการทำอินเนอร์จอยน์ (Inner join) นั้นจะมีการใช้คาต้าอิลิเมนท์ (data element) ไม่เหมือนกัน

สำหรับดาต้าโมเดลของดาต้าแวร์เฮาส์จะใช้ไดเมนแนลโมเดล (Dimensional Model) หรือ สตาร์จอยน์สกีมา (Star Join Schema) ซึ่งเป็นชื่อที่นักออกแบบฐานข้อมูลใช้กันมานานแล้ว เนื่องจากสกีมามีรูปร่างคล้ายดาว ซึ่งมีตารางหลัก 1 ตาราง อยู่ตรงกลางและมีตารางเล็ก ๆ ที่มีความสัมพันธ์กับตารางหลักอยู่รอบ ๆ ซึ่งตารางหลักนี้เป็นตารางเดียวที่ใช้การเชื่อมต่อหลายจุด (Multiple join) เพื่อเชื่อมต่อกับตารางอื่น ๆ แต่ตารางอื่น ๆ ที่อยู่รอบ ๆ นั้นจะมีแค่การเชื่อมต่อจุดเดียว (Single join) เพื่อเชื่อมเข้ากับตารางหลักเท่านั้น โดยตารางหลักเรียกว่าตารางแฟ็ค (Fact table) ส่วนตารางอื่น ๆ จะเรียกว่าตารางไดเมนชัน (Dimension table)

2.6 ความแตกต่างของลักษณะของข้อมูลที่ใช้ในงานโอเปอเรชันและดาต้าแวร์เฮาส์

ความแตกต่างของลักษณะของข้อมูลที่ใช้ในงานโอเปอเรชันและดาต้าแวร์เฮาส์ สามารถแสดงเปรียบเทียบได้ดังตารางที่ 2-1

ข้อมูลในงานโอเปอเรชัน	ข้อมูลในดาต้าแวร์เฮาส์
- เก็บรายละเอียดของข้อมูลตามแต่ละเรคอร์ดที่นำข้อมูลเข้า - สามารถทำการอัพเดตข้อมูลได้ - เข้าถึงข้อมูลครั้งละ 1 เรคอร์ด - ข้อมูลส่วนใหญ่ขึ้นกับงานของทรานแซคชันที่ทำ - ข้อมูลไม่มีความซ้ำซ้อน การเปลี่ยนแปลงข้อมูลแต่ละครั้งจะเขียนทับข้อมูลเดิม - โครงสร้างข้อมูลไม่มีการเปลี่ยนแปลง - ใช้งานกับฐานข้อมูลขนาดเล็กกว่า - สนับสนุนการปฏิบัติงานแบบวันต่อวัน - โปรแกรมที่ใช้งานขึ้นอยู่กับตัวแอพพลิเคชันที่ทำ - การใช้งานข้อมูลเกิดขึ้นตลอดเวลา	- เก็บเฉพาะข้อมูลสรุป - ไม่สามารถอัพเดตข้อมูลได้ ข้อมูลใหม่ที่เข้ามาจะเขียนต่อจากข้อมูลเดิม - เข้าถึงข้อมูลครั้งละกลุ่มข้อมูลหรือเป็นเซต - ข้อมูลจะถูกเก็บตามงานการวิเคราะห์ที่ต้องการใช้ - ข้อมูลมีความซ้ำซ้อนเพราะต้องเก็บข้อมูลเดียวกันในหลายช่วงเวลา - โครงสร้างข้อมูลมีความยืดหยุ่นตามการใช้งาน - ใช้งานกับฐานข้อมูลขนาดใหญ่ - จัดการข้อมูลได้ตามความต้องการใช้งาน - โปรแกรมที่ใช้งานขึ้นอยู่กับการวิเคราะห์ที่ต้องการ - การใช้งานข้อมูลเกิดขึ้นเป็นช่วงเวลา

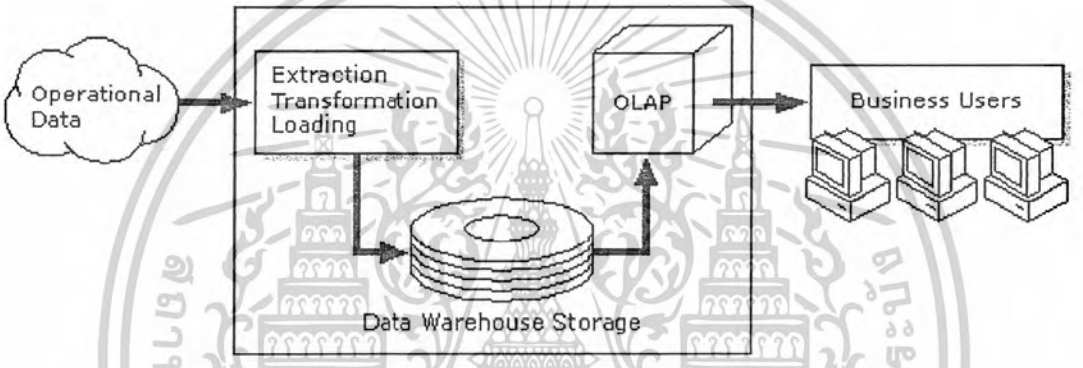
ตารางที่ 2-1 แสดงความแตกต่างของข้อมูลในโอเปอเรชันกับข้อมูลในดาต้าแวร์เฮาส์

2.7 ดาต้าแวร์เฮาส์ และ OLAP

ดาต้าแวร์เฮาส์ คือ ระบบฐานข้อมูลที่บรรจุข้อมูลที่แสดงถึงประวัติของธุรกิจขององค์กรนั้นๆ ข้อมูลประวัติข้อมูลถูกใช้สำหรับวิเคราะห์และสนับสนุนการตัดสินใจใจหลายๆระดับเพื่อวางแผนและพัฒนาประสิทธิภาพขององค์กร ข้อมูลในดาต้าแวร์เฮาส์ถูกจัดการในการวิเคราะห์มากกว่าการทำแบบระบบ OnLine Transaction Processing (OLTP)

โดยปกติแล้วดาต้าแวร์เฮาส์จะถูกใช้เป็นพื้นฐานสำหรับระบบสนับสนุนการตัดสินใจ (Decision Support System) เนื่องจากดาต้าแวร์เฮาส์ได้ถูกออกแบบมาเพื่อแก้ปัญหาที่เกิดขึ้นเมื่อเราใช้ฐานข้อมูลของงานโอเปอเรชันมาทำการวิเคราะห์

เทคโนโลยี OLAP (OnLine Analytical Processing) ทำให้เราสามารถนำข้อมูลไปใช้ได้อย่างมีประสิทธิภาพมากยิ่งขึ้นเหมาะสำหรับการวิเคราะห์แบบออนไลน์ (online analysis) ทำให้สามารถตอบคำถามที่มีความซับซ้อนได้อย่างรวดเร็ว เทคนิคมัลติไดเมนชันแนลโมเดล(Multidimensional data model) ของ OLAP และเทคนิคการรวมค่าข้อมูล (data aggregation) จะเข้ามาช่วยจัดการและสรุปค่าของข้อมูลที่มีปริมาณมากให้มีประสิทธิภาพยิ่งขึ้น ทำให้เราสามารถประเมินได้เร็วขึ้นโดยใช้การวิเคราะห์ออนไลน์ และเครื่องมือที่เป็นกราฟฟิคที่สนับสนุนระบบ OLAP ทำให้เกิดความเร็วและความยืดหยุ่นที่สนับสนุนการวิเคราะห์แบบเรียลไทม์ (Real Time)

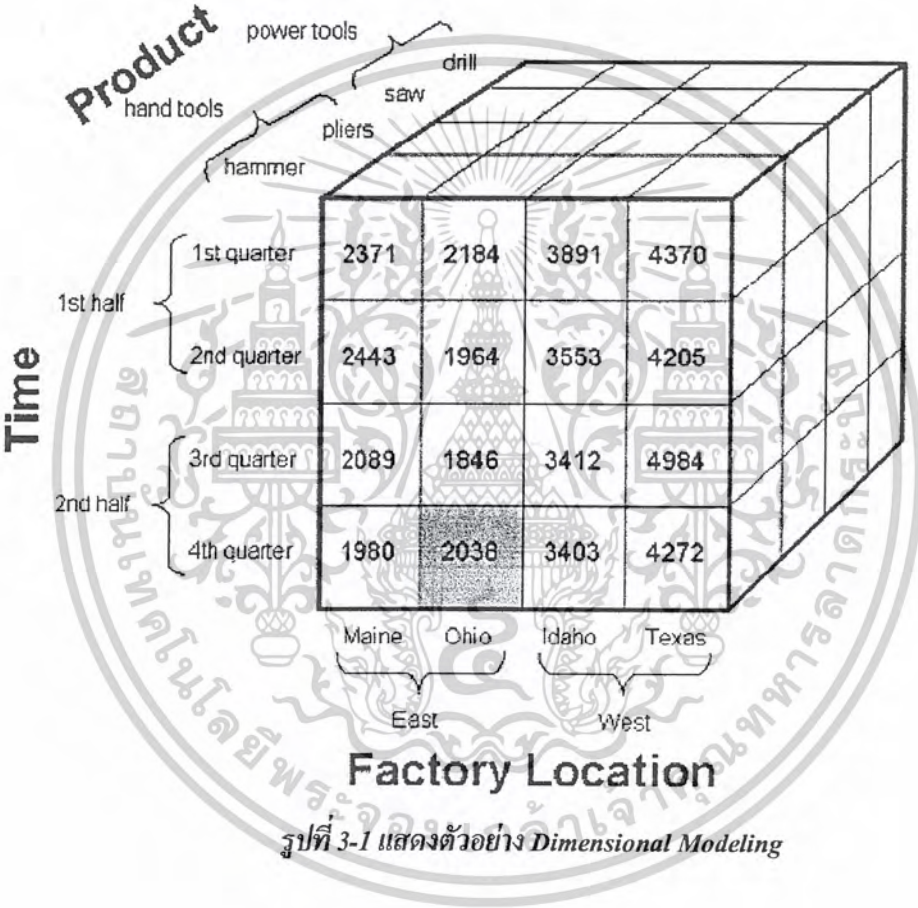


รูปที่ 2-5 แสดงความสัมพันธ์ระหว่างดาต้าแวร์เฮาส์และ OLAP

บทที่ 3
โครงสร้างและการออกแบบของดาต้าแวร์เฮาส์

3.1 Dimensional Modeling

ในการออกแบบดาต้าแวร์เฮาส์นั้น เราจะใช้โมเดลการมองหลายมิติ (Dimensional Model) เพื่อเอาไปออกแบบดาต้าแวร์เฮาส์ โดยจะใช้โมเดลที่ทำให้เห็นง่ายๆจึงเลือกใช้โมเดล 3 มิติ (dimensions) เราจะมองภาพโมเดลนี้ได้เป็นลูกบาศก์หรือคิวบ์ (Cube) ซึ่งจะมองเป็นกี่มิติก็ได้



รูปที่ 3-1 แสดงตัวอย่าง Dimensional Modeling

เราสามารถใช้เทคนิคที่เรียกว่า Slice และ Dice ในการตัดข้อมูลมาวิเคราะห์ได้ เช่น ถ้าต้องการจะรู้ว่า hammer ที่ขายที่ Ohio ในช่วงเวลา 4th quarter ขายได้ทั้งหมดเป็นเงินเท่าไรก็สามารถ Slice และ Dice เข้าไปดูข้อมูลได้ ข้อมูลที่จะได้ คือ 2038 ดอลลาร์ เป็นต้น

โดยที่ Dimensional Modeling จะประกอบด้วย 3 ส่วน คือ ตารางแฟ็ค (Fact table) ,ไดเมนชัน (Dimension) และก็ตัววัด (Measure)

- **แฟ็ค (Fact)**

แต่ละคาต้าแวร์เฮาส์จะมีตารางเดียวหรือหลายตารางของตารางแฟ็คก็ได้ แต่ละตารางแฟ็คจะมีตัววัดซึ่งไว้วัดค่าในการดำเนินการที่ธุรกิจนั้นๆต้องการ ตารางแฟ็คจะเก็บเรคอร์ดที่มีขนาดใหญ่บางครั้งเป็น 100 ล้านเรคอร์ด

ลักษณะของตารางแฟ็คนั้นจะบรรจุข้อมูลที่เป็นตัวเลขซึ่งสามารถนำมารวมกันได้ แต่ละตารางแฟ็คจะมีหลายอินเด็กซ์ที่จะเป็น foreign key ซึ่งเป็น primary key ของตารางไดเมนชันที่เชื่อมอยู่กับตารางแฟ็ค

Column	Description
Product_id	Foreign key for dimension table **product**.
Time_id	Foreign key for dimension table **time_by_day.**
Customer_id	Foreign key for dimension table **customer.**
Promotion_id	Foreign key for dimension table **promotion**.
Store_id	Foreign key for dimension table **store.**
Store_sales	Currency column containing the value of the sale.
Store_cost	Currency column containing the cost to the store of the sale.
Unit_sales	Numeric column containing the quantity sold.

ตารางที่ 3-1 แสดงตัวอย่าง ตารางแฟ็ค (fact table) เป็นตัวอย่างของตารางแฟ็คของ Food Mart 2000

ตารางแฟ็คจะทำหน้าที่เก็บข้อมูลที่มีความสัมพันธ์กัน จะประกอบไปด้วยตัววัดและตัวข้อมูลซึ่งแสดงถึงข้อมูลทางธุรกิจและการดำเนินการทางธุรกิจหรือเหตุการณ์ที่สามารถใช้ในการวิเคราะห์หรือใช้ประมวลผลทางธุรกิจ ใน OLAP Services ส่วนของแฟ็คจะอยู่ในรูปของตารางหลักของฐานข้อมูล OLAP ซึ่งจะเก็บข้อมูลที่เป็นตัวเลข ตารางเหล่านี้จะหมายถึงตารางแฟ็คและมักจะได้รับมาจากข้อมูลดิบในฐานข้อมูลเชิงปฏิบัติการ

ข้อมูลในตารางแฟ็ค ควรจะมีลักษณะ คือ เป็นตัวเลข (Numeric), มีค่าต่อเนื่อง (Continuously valued) และมาจากทุก ๆ ตารางไดเมนชัน (Additive)

- **Numeric** : เหตุผลที่ต้องเป็นตัวเลข ก็คือ ในความเป็นจริงคำถามทุกคำถามที่ทำกับตารางแฟ็คนั้น ระบบจัดการฐานข้อมูล (DBMS) จะใช้เรคอร์ดเป็นพัน ๆ หมื่นๆ หรือเป็นล้านๆ เรคอร์ด เพื่อสร้างเซตคำตอบขึ้นมา จำนวนเรคอร์ดที่มากมายนี้จะถูกบีบอัดเป็นจำนวนแถวเพียงไม่กี่แถว ในเซตคำตอบของผู้ใช้นั่นก็คือ การทำคำสั่ง Select list หรือ Built-in function : SUM ของ SQL นั่นเอง วิธีเดียวที่จะบีบอัดเรคอร์ดเหล่านี้ให้

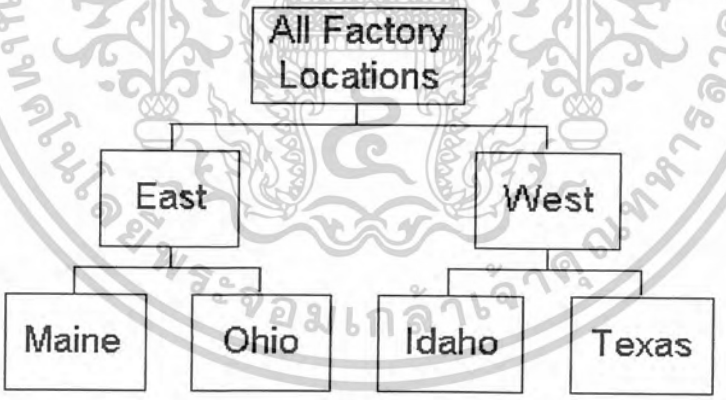
กลายเป็นเซตคำตอบได้ ก็คือ การบวกค่าของเรคอร์ดต่าง ๆ เข้าด้วยกัน ดังนั้นเรคอร์ดที่มีลักษณะเป็นตัวเลขและ Additive จะทำให้สร้างเซตคำตอบได้ง่าย

- **Continuously valued** : ค่าในตารางแฟ็คควรจะเป็นค่าที่มีความต่อเนื่อง เพื่อที่จะสามารถนำทางให้กับผู้ออกแบบฐานข้อมูล ให้สามารถแยกได้ว่าอะไรคือแฟ็ค อะไรคือแอททริบิวต์ (Attribute) ของตารางไดเมนชัน
- **Additive** : ค่าของตัววัดในตารางแฟ็ค จะเป็นค่า Additive แต่ก็จะมีค่าแฟ็คที่ไม่เป็น Additive ด้วย เป็นแฟ็คที่เป็น Semiadditive หรือ Nonadditive ซึ่ง Semiadditive คือ เกรน (grain) ที่ไม่ได้มาจากทุกตารางไดเมนชัน ซึ่งจะสามารถหาผลรวมตามไดเมนชันบางไดเมนชันเท่านั้น ส่วน Nonadditive คือ เกรนที่ไม่สามารถหาผลรวมตามไดเมนชันใด ๆ ได้

- **ไดเมนชัน (Dimension)**

ไดเมนชันคือ โครงสร้างของแอททริบิวท์ของลูกบาศก์ ไดเมนชันจะมีข้อมูลของลำดับชั้นที่ถูกจัดเก็บเพื่อเอาไว้อธิบายข้อมูลที่อยู่ในตารางแฟ็ค

เก็บคำอธิบายของแต่ละไดเมนชันของธุรกิจเอาไว้ ซึ่งคำอธิบายเหล่านี้จะช่วยในการอธิบายถึงสมาชิกในทุก ๆ ไดเมนชันและในตารางไดเมนชันจะประกอบด้วยหลาย ๆ แอททริบิวท์ ซึ่งแอททริบิวท์ที่ดีจะต้องเป็นตัวอักษร และแต่ละแอททริบิวท์ต้องแยกออกจากกัน แอททริบิวท์หล่านี้ถูกใช้เป็นแหล่งที่มาของข้อบังคับ (Constrains) และ row header ในเซตคำตอบของผู้ใช้



รูปที่ 3-2 ลำดับขั้นในการออกแบบไดเมนชัน

จากรูปแสดงลำดับขั้นของไดเมนชัน Location โดยไดเมนชันนี้ถูกกำหนดมาจากการเลือกRegion และ State จากตารางข้างล่าง

State_ID	Region	State
1	East	Maine
2	East	Ohio
3	West	Idaho
4	West	Texas

รูปที่ 3-3 แสดงตารางของไดเมนชัน

จากตารางนี้ State_ID จะถูกนำไปเป็น Foreign Key ของตารางแฟ็คที่เกี่ยวข้องต่อไป

- **ตัววัดค่า (measure)**

เป็นแอททริบิวท์ที่เป็นตัวเลขของแฟ็คแสดงประสิทธิภาพหรือพฤติกรรมของธุรกิจที่เกี่ยวข้องกับไดเมนชัน ซึ่งสมาชิกที่แท้จริง จะเรียกว่า ตัวแปร (Variables)

3.2 การออกแบบข้อมูล (Data Modeling)

การออกแบบข้อมูล เป็นขั้นตอนที่สำคัญในการสร้างดาต้าแวร์เฮาส์ เพราะการออกแบบจะทำให้สามารถมองระบบที่กำลังจะสร้างได้ชัดเจนยิ่งขึ้น

3.2.1 Data Cubes

ในดาต้าแวร์เฮาส์นั้น ข้อมูลจะถูกสร้างให้อยู่ในรูปแบบที่ง่ายต่อการมองและวิเคราะห์ โดยจะสร้างเป็นแบบลูกบาศก์หรือคิวบ์โดยส่วนใหญ่แล้วมักจะเป็น 3 มิติ หรือมากกว่า 3 มิติ (มากกว่า 3 มิติ จะเรียกว่า Hypercube)

3.2.2 โอเปอเรชัน (Operation) ที่ใช้ในการวิเคราะห์ข้อมูล

ผู้ออกแบบข้อมูลสามารถใช้ Dimensional Modeling ที่รองรับ OLAP และรองรับการตัดสินใจ โดยการวิเคราะห์ข้อมูลใน OLAP Services จะมีอยู่ 4 โอเปอเรชันด้วยกัน คือ Drill-down, Drill-up, Slicing และ Dicing

- Drill-down และ Drill-up

Drill-down และ Drill-up เป็นโอเปอเรชันที่ใช้สำหรับการมองข้อมูลที่เป็นระดับๆ ในไดเมนชัน โดยการ Drill-down จะทำให้เห็นรายละเอียดของข้อมูลในระดับที่ลึกลงไป ในขณะที่การ Drill-up นั้นจะเป็นการดูรายละเอียดของข้อมูลในระดับที่สูงขึ้น ดังแสดงในตารางที่ 3-2

	Quarter	Ford	Mercury
A B C-Autos	Q1 – 1998	120	89
	Q2 – 1998	123	91
	Q3 – 1998	101	75
	Q4 – 1998	99	73
Drill-Up ↑ Drill-Down ↓			
	Month	Ford	Mercury
A B C-Autos	Oct – 1998	40	31
	Nov – 1998	35	29
	Dec – 1998	24	13

ตารางที่ 3-2 แสดงการ Drill-down และ Drill-up

จากตารางที่ 3-2 เป็นข้อมูลแสดงการขายรถยนต์ของตัวแทนจำหน่าย ซึ่งจะแบ่งเป็น 4 ควอเตอร์ (Quarter) ด้วยกัน เมื่อทำการ Drill-down ในควอเตอร์ที่ 4 (Q4 – 1998) แล้วจะเห็นว่าจะเป็นการแสดงข้อมูลในระดับของเดือนใน Q4 (เดือนตุลาคม ถึง เดือนธันวาคม) ส่วนการ Drill-up จะแสดงผลตรงกันข้ามกับการ Drill-down นั่นเอง เช่น ถ้า Drill-up ข้อมูลในระดับเดือน ก็จะเห็นข้อมูลแสดงในระดับของควอเตอร์

- Slicing และ Dicing

การตัดข้อมูล (Slicing) เวลาที่ทำการวิเคราะห์ข้อมูลสามารถที่จะวิเคราะห์ข้อมูลได้เพียงบางช่วงหรือเฉพาะส่วนที่สนใจเท่านั้นส่วนการหมุนข้อมูล (Dicing) ก็สามารถหมุนหรือสับเปลี่ยนตำแหน่งของไดเมนชันได้ ดังตารางที่ 3-3, 3-4 และ 3-5

1998 Sales		Ford		Mercury	
	Quarter	Escort	Taurus	Tracer	Sable
Midwest	Detroit	21	39	13	33
	Cleveland	19	40	9	35
Northeast	Philadelphia	8	26	3	19
	New Jersey	9	19	5	12

ตารางที่ 3-3 แสดงการตัดข้อมูล (Slicing)

1998 Sales		1998 Ford and Mercury			
	Quarter	Q1	Q2	Q3	Q4
Midwest	Detroit	119	134	128	110
	Cleveland	104	124	120	101
Northeast	Philadelphia	99	110	109	89
	New Jersey	65	79	78	69

ตารางที่ 3-4 แสดงการหมุนข้อมูล (Dicing)

1998 Sales		1998 Ford Only			
	Quarter	Q1	Q2	Q3	Q4
Midwest	Detroit	73	80	73	70
	Cleveland	65	79	79	59
Northeast	Philadelphia	33	59	49	38
	New Jersey	36	45	49	38

ตารางที่ 3-5 แสดงการตัดข้อมูลจากข้อมูลในตารางที่ 3-4

จากตารางข้างต้น ประกอบไปด้วย 3 ไดเมนชัน คือ Geographic location, Time และ Product (รถยนต์) ในตารางที่ 3-3 เป็นการตัดข้อมูลการวิเคราะห์เฉพาะรถยนต์ยี่ห้อ Ford และ Mercury ใน 4 เมืองทางเขต Midwest และ Northeast ในปี 1998 ในตารางที่ 3-4 แสดงการหมุนข้อมูล โดย Time จะกลายเป็นไดเมนชันหลัก ส่วนในตารางที่ 3-5 เป็นการตัดข้อมูลจากตารางที่ 3-4 ที่เลือกเฉพาะรถยนต์ยี่ห้อ Ford ขึ้นมาวิเคราะห์เท่านั้น

3.3 โครงสร้างฐานข้อมูล (Database Schema)

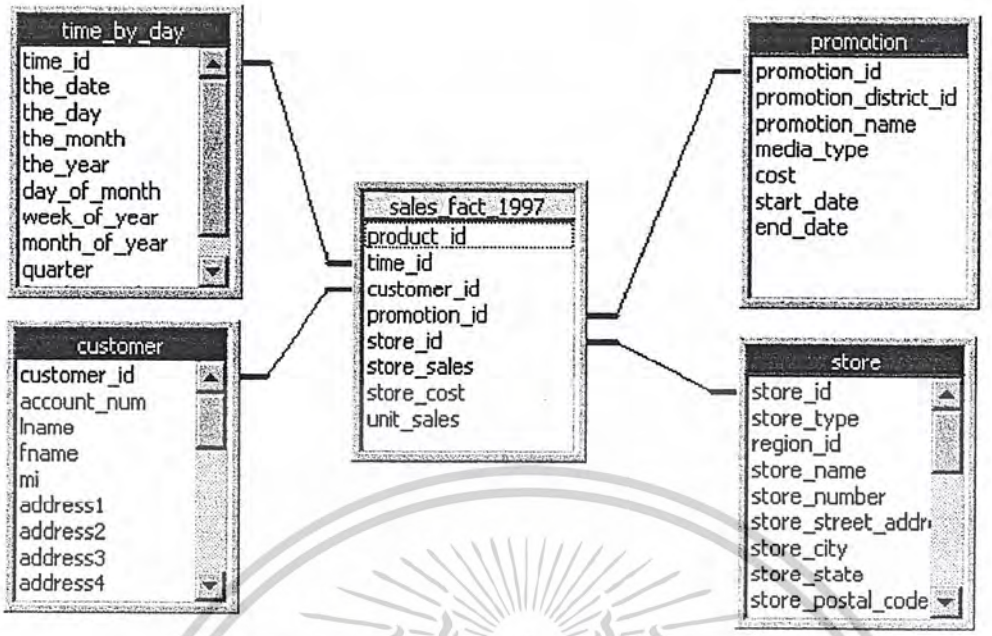
โครงสร้างฐานข้อมูลหลัก ๆ จะมีอยู่ 2 รูปแบบ คือ สกีมาแบบดาว (Star Schema) และสกีมาแบบสโนว์เฟล่ก (Snowflake schema)

3.3.1 สกีมาแบบดาว

สกีมาแบบดาวเป็นวิธีหนึ่งของไดเมนชันแนลโมเดล ซึ่งเป็นชื่อที่ใช้กันมานานแล้ว เนื่องจากมีรูปร่างคล้ายกับดาว ซึ่งประกอบไปด้วยตารางใหญ่ 1 ตารางอยู่ตรงกลาง และมีตารางเล็ก ๆ ที่มีความสัมพันธ์กับตารางหลักอยู่รอบๆ ตารางหลักนี้จะเป็นตารางเดียวที่ใช้การเชื่อมต่อแบบหลายจุด (Multiple join) เพื่อเชื่อมต่อกับตารางอื่น ๆ แต่ตารางอื่น ๆ ที่อยู่รอบ ๆ จะมีการเชื่อมต่อเพียงแค่จุดเดียว (Single join) เพื่อเชื่อมเข้ากับตารางหลักเท่านั้น

ตารางหลัก จะเรียกว่า ตารางแฟ็ค (Fact Table)

ตารางอื่น ๆ จะเรียกว่า ตารางไดเมนชัน (Dimension Table)

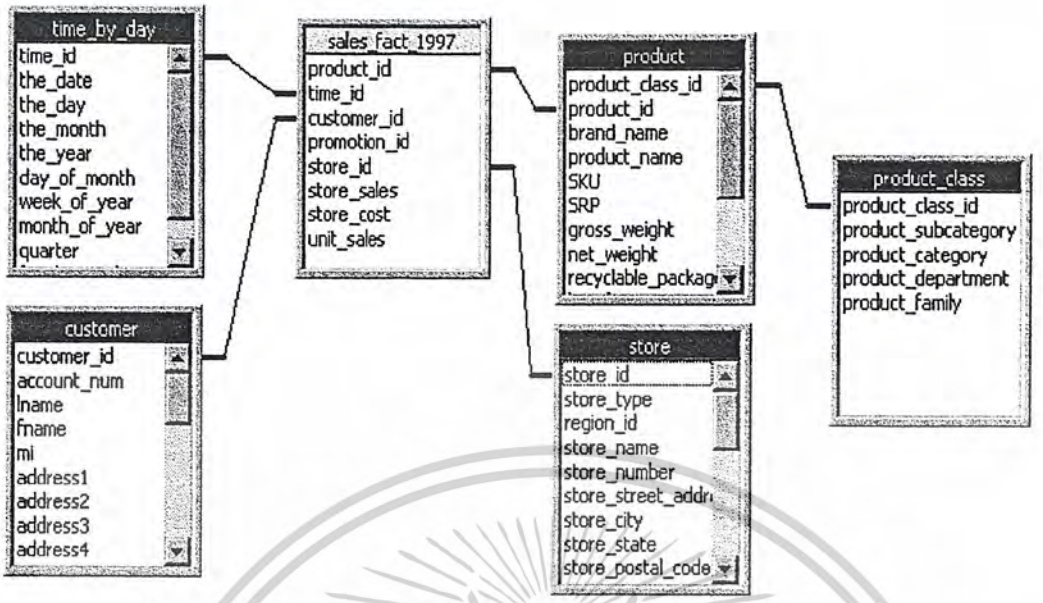


รูปที่ 3-4 แสดงโครงสร้างฐานข้อมูลที่เป็นสกีมาแบบดาว (Star Schema)

3.3.2 สกีมาแบบสโนว์เฟลก (Snowfake Schema)

ตารางไดเมนชันที่มีข้อมูลแบบเป็นลำดับขั้น (Hierarchy) อาจจะมีการแตกข้อมูลออกมาเป็นตารางย่อย ๆ เพื่อให้ผู้ออกแบบสามารถเข้าใจได้ง่ายยิ่งขึ้น ซึ่งเรียกโครงสร้างนี้ว่า สโนว์เฟลก(Snowflake) ความสัมพันธ์ที่เป็นแบบ many – to – one แต่ละอันจะถูกแยกออกเป็นตารางย่อย ๆ โดย Primary key ของตารางย่อย จะต้องมี Foreign key ของตารางที่ตัวเองไปเกาะอยู่ ดังแสดงในรูปที่ 3-5

การทำโครงสร้างแบบสโนว์เฟลก อาจทำให้ประสิทธิภาพในการขับเคลื่อนลดต่ำลง เนื่องจากต้องมีการรวมกันระหว่างตารางย่อยกับตารางไดเมนชันอีกทีหนึ่ง รวมทั้งต้องสร้าง SQL ที่มากขึ้น แต่อย่างไรก็ตาม สโนว์เฟลกก็เหมาะสำหรับฐานข้อมูลที่มีขนาดใหญ่ เช่น มีจำนวนหลายแสนเรคอร์ด เพราะมีส่วนช่วยลดความซ้ำซ้อนลงได้ รวมทั้งโครงสร้างแบบสโนว์เฟลกมีข้อดี เช่น เมื่อต้องการเรียกดูเฉพาะฟิลด์ (field) ในตารางไดเมนชัน ที่เป็นสโนว์เฟลก ก็ทำเพียงแค่ดึงข้อมูลจากตารางไดเมนชันนั้นเพียงตารางเดียวและทำการเชื่อมต่อกับตารางแฟ็คเท่านั้น ไม่ต้องทำการเชื่อมต่อระหว่างตารางไดเมนชันนั้นกับตารางย่อยของตัวมันเองอีกและจะไม่ทำให้เกิดปัญหาการอัพเดทโดยไม่ต้องทำการอัพเดทหลายที่



รูปที่ 3-5 แสดงโครงสร้างฐานข้อมูลที่เป็นสกีมาแบบสโนว์เฟลก (Snowflake Schema)

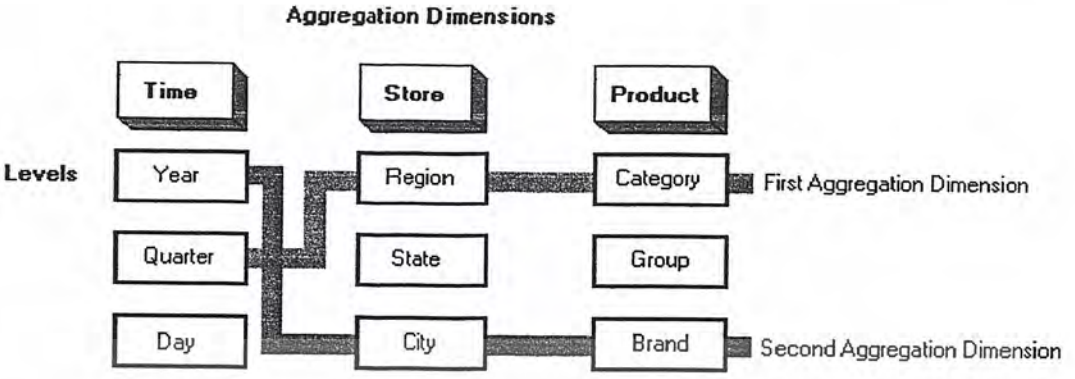
3.4 การรวมค่าข้อมูล (Aggregation)

เนื่องจากข้อมูลพื้นฐานของ ไดเมนชันแนลดาต้าแวร์เฮาส์ (Dimensional Data Warehouse) จะประกอบด้วยเรคอร์ดจำนวนมาก ถ้านักวิเคราะห์ต้องการทำคิวรีโดยไม่มีการกำหนดข้อบังคับให้กับบางไดเมนชัน เนื่องจากในดาต้าแวร์เฮาส์จะมีเฉพาะแต่ข้อมูลพื้นฐานเท่านั้น (Based-level data) จึงทำให้เวลาคิวรีจะต้องทำการรวมข้อมูลภายในเรคอร์ดจำนวนมากมาย ซึ่งถ้าเป็นเช่นนี้จะทำให้การทำคิวรีเกิดความสิ้นเปลืองสูงมาก ดังนั้นจึงได้มีการทำการรวมค่าข้อมูลเอาไว้ล่วงหน้า เพื่อเร่งให้การทำคิวรีสามารถทำได้เร็วขึ้น มีประสิทธิภาพที่ดียิ่งขึ้น

การรวมค่าต้องมี fact table record ที่แสดงถึงข้อสรุปที่ได้จากตารางแฟ็ค ระดับพื้นฐาน (Based-level fact table) เป็นข้อมูลตัววัดมีขนาดเล็กที่สุด

การรวมค่ามีหลายชนิด และแต่ละชนิดจะมีตารางแฟ็คเป็นของตัวเอง ซึ่งตารางแฟ็คเหล่านี้จะถูกเรียกว่าเป็น Derivative fact table เพราะข้อมูลเหล่านั้นได้รับมาจากตารางแฟ็คระดับพื้นฐาน

เราสามารถสร้างการรวมค่าได้มากมายตามที่เราต้องการ ซึ่งในความเป็นจริงเราจะสร้างการรวมค่าไว้เฉพาะส่วนที่เราต้องการเท่านั้น ไม่จำเป็นต้องสร้างการรวบรวมทุก ๆ ฟิลด์ในแต่ละไดเมนชันและไม่จำเป็นต้องทำการรวมค่าในทุก ๆ ไดเมนชันด้วย



รูปที่ 3-6 แสดงการรวมค่า (Aggregation)

3.5 Crosstabulation หรือ Crosstab

CrossTabs เป็นวิธีในการแสดงกลุ่มของข้อมูล เพื่อให้มีความสัมพันธ์กันและสามารถมองเห็นแนวโน้มได้ง่ายซึ่งฟิลด์ของตารางไดเมนชันจะอยู่ในรูปของ CrossTabs และค่าในแต่ละฟิลด์จะถูกแบ่งกลุ่มและถูกสรุปภายในไดแมนชัน ทำให้สามารถมองได้ง่ายกว่าแสดงแบบตารางปกติ

	Store Cost	Store Sales	Store Profit
Breakfast Foods	2,756.80	$6,941.46	151.79%
Carousel	595.97	$1,500.11	151.71%
Canned Products	1,317.13	$3,314.52	151.65%
Baking Goods	15,370.61	$38,670.41	151.59%
Alcoholic Beverages	5,576.79	$14,029.08	151.56%
Health and Hygiene	12,972.99	$32,571.86	151.07%
Snack Foods	26,963.34	$67,609.82	150.75%
Baked Goods	6,564.09	$16,455.43	150.69%
Beverages	11,069.53	$27,748.53	150.67%
Frozen Foods	22,030.66	$55,207.50	150.59%

รูปที่ 3-7 แสดงข้อมูลในตารางแบบปกติ

		MeasuresLevel	
- Country	+ State Province	Unit Sales	Store Cost
All Customers	All Customers Total	266,773.00	225,627.23
- Canada	Canada Total		
	+ BC		
+ Mexico	Mexico Total		
+ USA	USA Total	266,773.00	225,627.23

รูปที่ 3-8 แสดงข้อมูลในตารางแบบ CrossTabs

3.6 ขั้นตอนการออกแบบดาต้าแวร์เฮาส์

ในการออกแบบคาต้าแวร์เฮาส์ จะประกอบไปด้วย 4 ขั้นตอน คือ

3.6.1 เลือกกระบวนการทางธุรกิจ (Business Process) ที่ต้องการสร้าง

กระบวนการทางธุรกิจ เป็นกระบวนการหลักที่ต้องการทำในองค์กร ซึ่งกระบวนการนั้นมีระบบเดิมสนับสนุนอยู่ โดยข้อมูลในระบบนั้นสามารถนำมารวบรวม เพื่อทำเป็นคาต้าแวร์เฮาส์ได้ เช่น ใบสั่งของ (Order), ใบส่งของ (Invoice), สินค้าที่ส่งไป (Shipments), รายการสินค้า (Inventory) ฯลฯ

3.6.2 เลือกเกรน (Grain) ของกระบวนการทางธุรกิจ

เกรน : เป็นข้อมูลพื้นฐาน ข้อมูลที่เป็นอะตอมมิก (Atomic) ซึ่งถูกแสดงอยู่ในตารางแฟ็ค สำหรับกระบวนการนี้ เกรนที่มีอยู่ทั่วไป เช่น ข้อมูลการทำทรานแซคชันในแต่ละ, ข้อมูลของการทำงานในแต่ละวันหรือสรุปในแต่ละวัน, ข้อมูลสรุปการทำงานในแต่ละเดือน

3.6.3 เลือกไดเมนชันที่จะถูกนำมาใช้กับแต่ละเรคอร์ดของตารางแฟ็ค

ไดเมนชันที่มีอยู่โดยทั่วไป เช่น Product, Customer, Time ซึ่งแต่ละไดเมนชันจะถูกอธิบายแยกกันในลักษณะของแอททริบิวท์และจะถูกอธิบายเป็นตัวหนังสือ

3.6.4 เลือกแฟ็คที่มีการวัดหรือมีการประมวลผลหรือมีการคำนวณไว้แล้ว

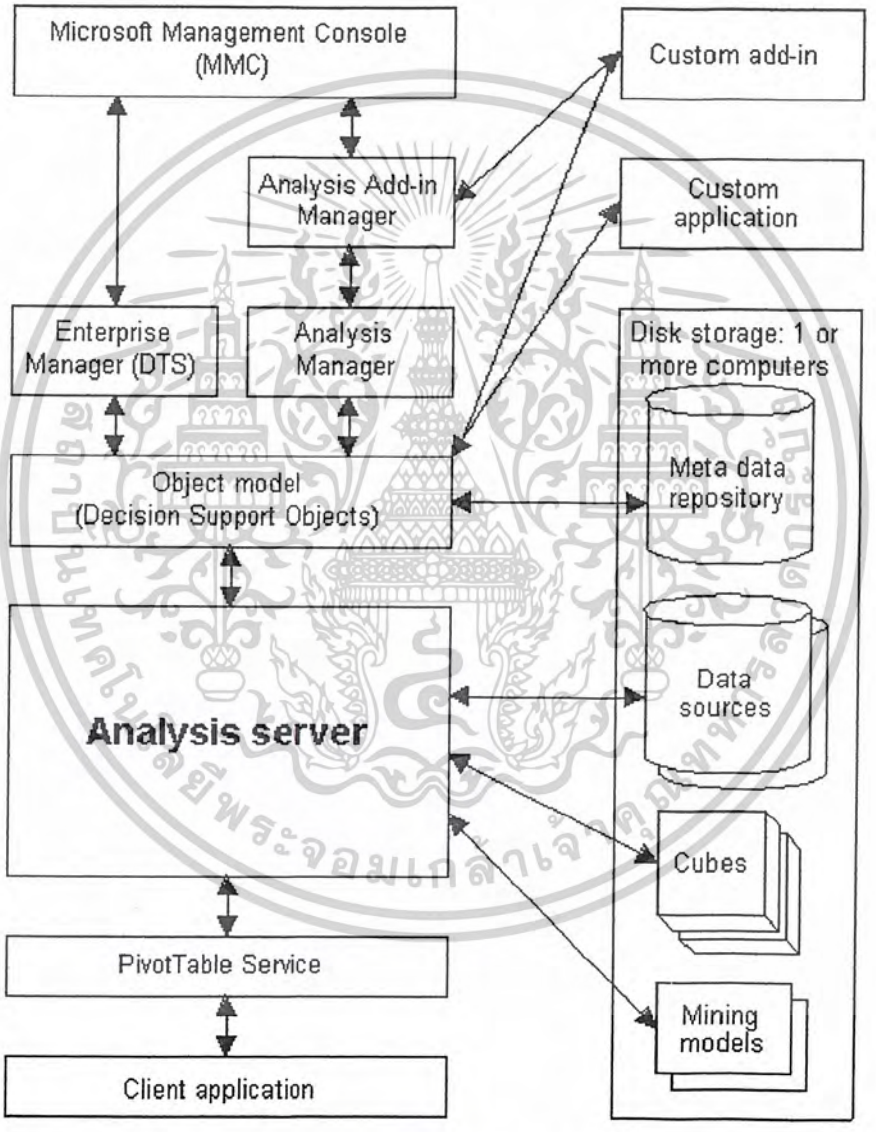
ที่จะเก็บอยู่ในแต่ละ เรคอร์ดของตารางแฟ็ค ตัววัดที่มีอยู่โดยทั่วไป เช่น ปริมาณต่าง ๆ ซึ่งจะมีลักษณะเป็นตัวเลข ได้แก่ ยอดขาย, จำนวนสินค้าที่ ขายได้, ค่าใช้จ่ายทั้งหมด

บทที่ 4

การใช้งาน Microsoft SQL Server 2000 Analysis Service

4.1 Microsoft Analysis Service

ทูลส์ที่เราเลือกนำมาสร้างระบบดาต้าแวร์เฮาส์คือ Microsoft SQL Server 2000 Analysis Service ซึ่งในตัวโปรดักส์นี้จะมีส่วนของ Analysis Service ซึ่งเป็นส่วนสำคัญที่ใช้สำหรับสร้างระบบดาต้าแวร์เฮาส์ โดยใน Microsoft SQL Server 2000 Analysis Service จะประกอบไปด้วยคอมโพเนนท์มากมายและมีสถาปัตยกรรมดังรูปที่ 4-1



รูปที่ 4-1 แสดงสถาปัตยกรรมภายในของ Analysis Service

4.2 การสร้างมัลติไดเมนชันแนลโมเดล (Multidimensional Modal) บน Analysis Service

เราสามารถสร้างระบบคาต้าแวร์เฮาส์ได้โดยใช้โมเดลของมัลติไดเมนชันแนลโมเดล โครงสร้างหลักของข้อมูลที่เก็บในลักษณะมัลติไดเมนชัน คือ คิวบ์ การสร้างคิวบ์ใน Analysis Service จะมี วิซาร์ด (Wizard) มาช่วยทำให้เพิ่มความสะดวกสบายให้ กับผู้ออกแบบมายิ่งขึ้น สามารถทำได้ตามขั้นตอนดังนี้

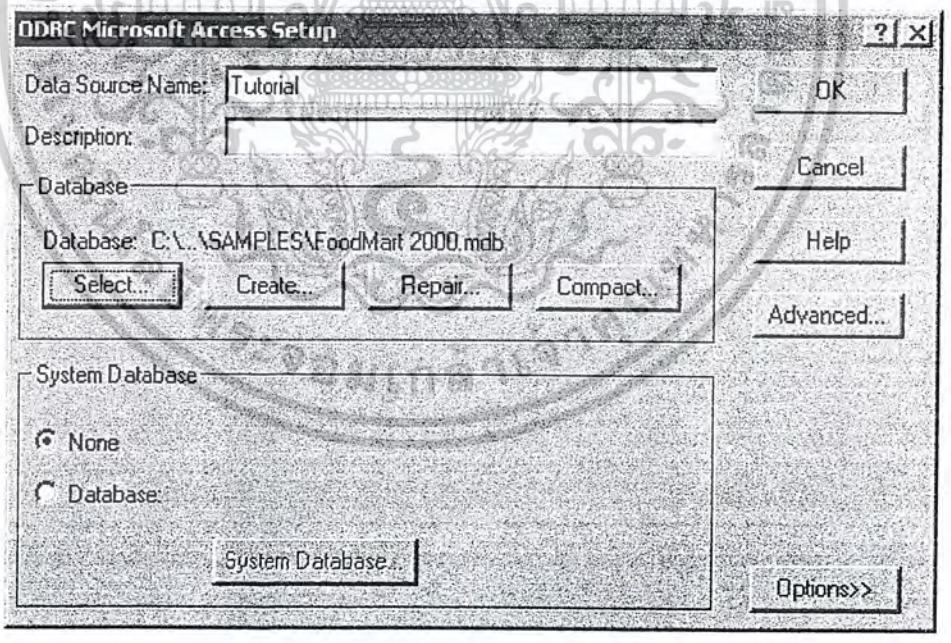
4.2.1 การติดต่อแหล่งข้อมูล (Data Source)

ก่อนเริ่มทำงานกับ Analysis Service อย่างแรกต้องตั้งค่าการเชื่อมต่อกับแหล่งข้อมูลใน ODBC Data Source Administrator

4.2.1.1 วิธีการติดตั้ง Data Source Name (DSN)

1. สำหรับ Window NT :คลิกที่ปุ่ม **Start** ชี้ไปที่ **Setting**, คลิกที่ **Control Panel** และ ดับเบิลคลิกที่ Data Sources (ODBC)

   สำหรับ Window 2000: คลิกที่ **Start**, ชี้ไปที่ **Setting**, คลิกที่ **Control Panel**, ดับเบิลคลิกที่ **Administrative Tools** และ ดับเบิลคลิกที่ **Data Source (ODBC)**
2. ที่แท็ป **System DSN** คลิก **Add**
3. เลือก Driver ของ Data Source นั้นๆ ในที่นี้เลือกของ **Microsoft Access Driver (*.mdb)** และ ก็คลิก **Finish**
4. ในช่อง **Data Source Name** ใส่ชื่อ Data Source Name ที่ต้องการ ในที่นี้จะใส่ **Tutorial** และได้ **Database** คลิก **Select**



รูปที่ 4-2 การติดตั้ง Data Source Name

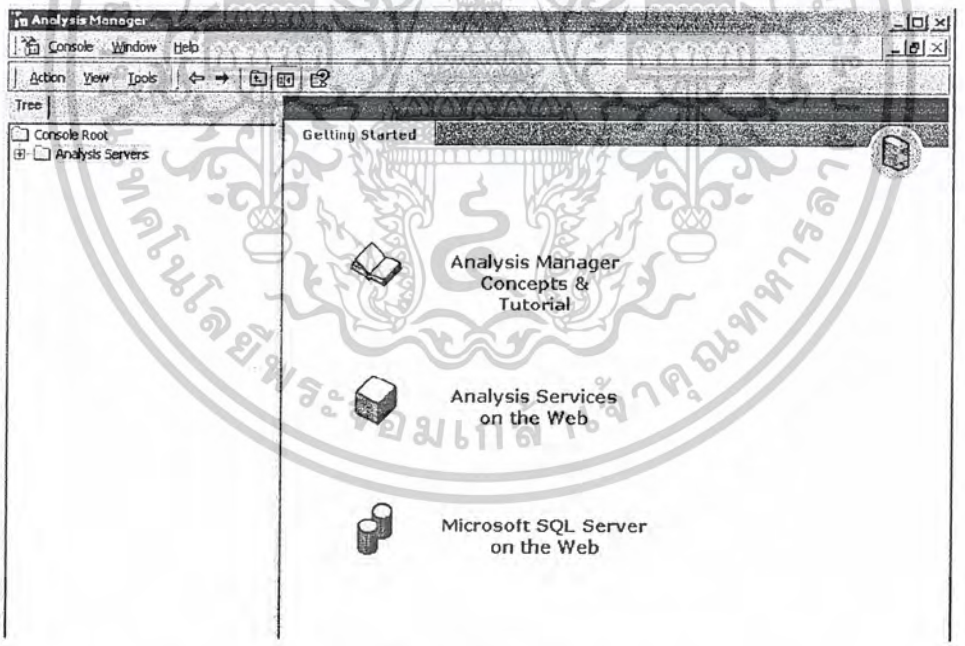
5. ในช่อง **Database** ให้คลิกปุ่มที่ **Select** ซึ่งจะปรากฏไดอะล็อกบ๊อกซ์ขึ้นมาให้เลือกตำแหน่งที่ Database นั้นอยู่ ในที่นี้จะเลือก **C:\Program Files\Microsoft Analysis Services\Samples** และคลิก **FoodMart 2000.mdb** คลิก **Ok**
6. ใน **ODBC Microsoft Access Setup dialog box** คลิก **Ok**
7. ใน **ODBC Data Source Administrator dialog box** คลิก **Ok**

4.2.1.2 ติดตั้ง Database และ Datasource

ก่อนที่จะออกแบบคิวบ์ จะต้องติดตั้ง Database Structure ให้ติดต่อไปยัง Data source ที่ได้ติดตั้งไปแล้วก่อนหน้านี้ใน ODBC Data Source Administrator

วิธีการติดตั้ง Database Structure

1. เปิดโปรแกรม Analysis Manager ขึ้นมาและใน Analysis Manger Tree ให้ กดปุ่มขยาย **Analysis Servers**
2. คลิกที่ชื่อ Server ของที่เราต้องการจะทำงานด้วย
3. คลิกขวาและคลิก **New Database**
4. ใน **Database** Dialog Box ใน **Database name** box ให้ใส่ Database name ที่ต้องการ ในที่นี้จะใส่เป็น **Tutorial** และคลิก **Ok**
5. ใน Analysis Manager Tree pane, ให้ขยาย **server** และขยาย **Tutorial** database ที่เพิ่งสร้างไป



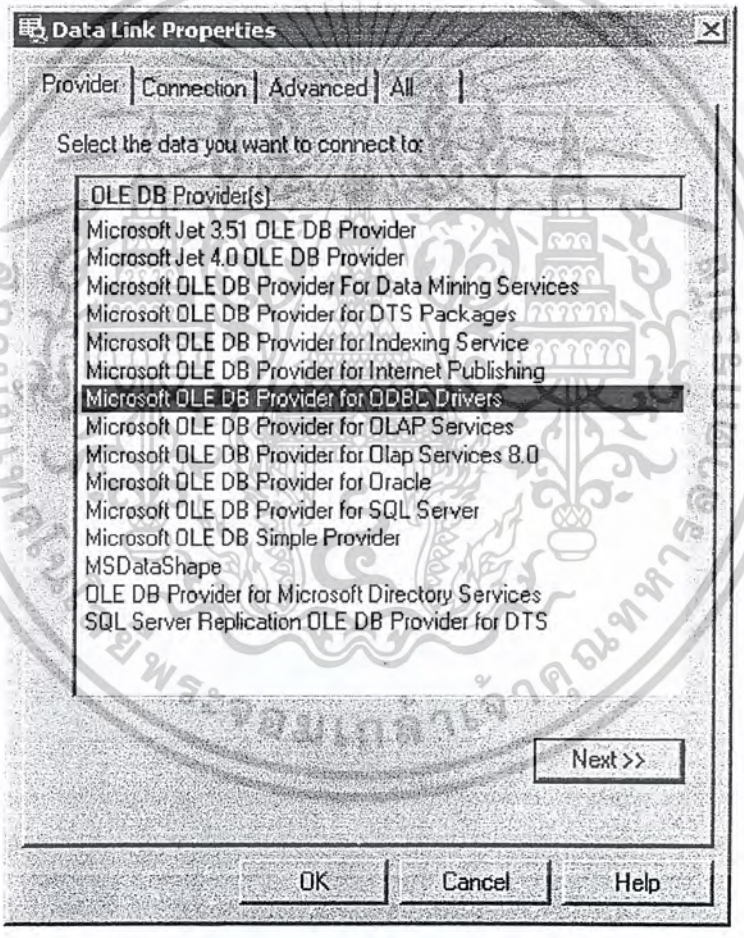
รูปที่ 4-3 รูปแสดงหน้าจอของโปรแกรม Analysis Manager

ต่อไปจะเป็นการสร้างการเชื่อมต่อไปยังข้อมูลตัวอย่าง ใน **Tutorial** data source การติดตั้ง data source ใน Analysis Manager ที่ทำไปนี้ ถูกเชื่อมต่อไปยังฐานข้อมูล ที่ system data source name (DSN)

ที่ได้ทำการติดตั้งใน ODBC Data Source Administrator ไว้ล่วงหน้าก่อนแล้ว ข้อมูลทั้งหมดที่มาจากแหล่งข้อมูลนี้จะถูกนำมาสร้างคิวบ์

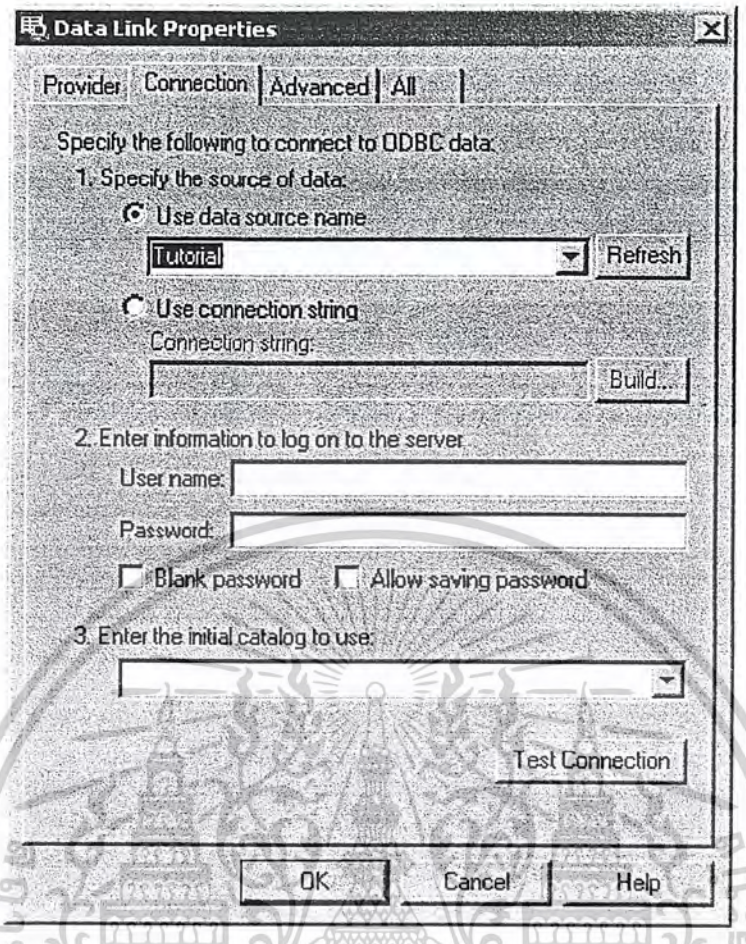
4.2.1.3 วิธีการติดตั้ง Data Source ให้กับ Analysis Manger

1. ใน **Analysis Manager** tree pane, คลิก **Data source** ใต้ **Tutorial database** และคลิกขวาเลือก **New Data Source**
2. ในหน้าต่าง **Data Link Properties** คลิกแท็ป **Provider** และคลิก **Microsoft OLE DB Provider for ODBC Drivers** ซึ่งถ้าเราใช้ฐานข้อมูลตัวอื่นที่ไม่ใช่ Microsoft Access ให้เลือกตัวอื่นตามแต่ละชนิดของฐานข้อมูล เช่น ถ้าฐานข้อมูลของเราเป็น Microsoft SQL Server ก็ให้เลือกที่ Microsoft OLE DB Provider for SQL Server



รูปที่ 4-4 การเลือก Driver เพื่อใช้ในการติดต่อกับแหล่งข้อมูล

3. คลิกแท็ป **Connection** และที่ **Use data source name** ให้คลิกเลือก **Tutorial**
4. คลิก **Test Connection** เพื่อตรวจสอบการเชื่อมต่อว่าสมบูรณ์หรือไม่



รูปที่ 4-5 การเลือก Data Source Name

5. คลิก **OK** เพื่อปิดหน้า **Data Link Properties**

4.2.2 เริ่มสร้างคิวบ์ (Building Cube)

ในขั้นตอนนี้จะใช้ **Cube Wizard** เข้ามาช่วยสร้างคิวบ์ ซึ่งคิวบ์ที่จะสร้างนี้จะใช้เป็นคาต้าแวร์เฮาส์ของแผนกการตลาด เพื่อนำมาใช้ในการวิเคราะห์ค่ายอดขาย (Sales)

- เปิด **Cube Wizard** ใน Analysis Manager tree pane, ใต้ **Tutorial** database คลิกขวาที่โฟลเดอร์ Cube, คลิกที่ **New Cube** และก็คลิกที่ **Wizard**

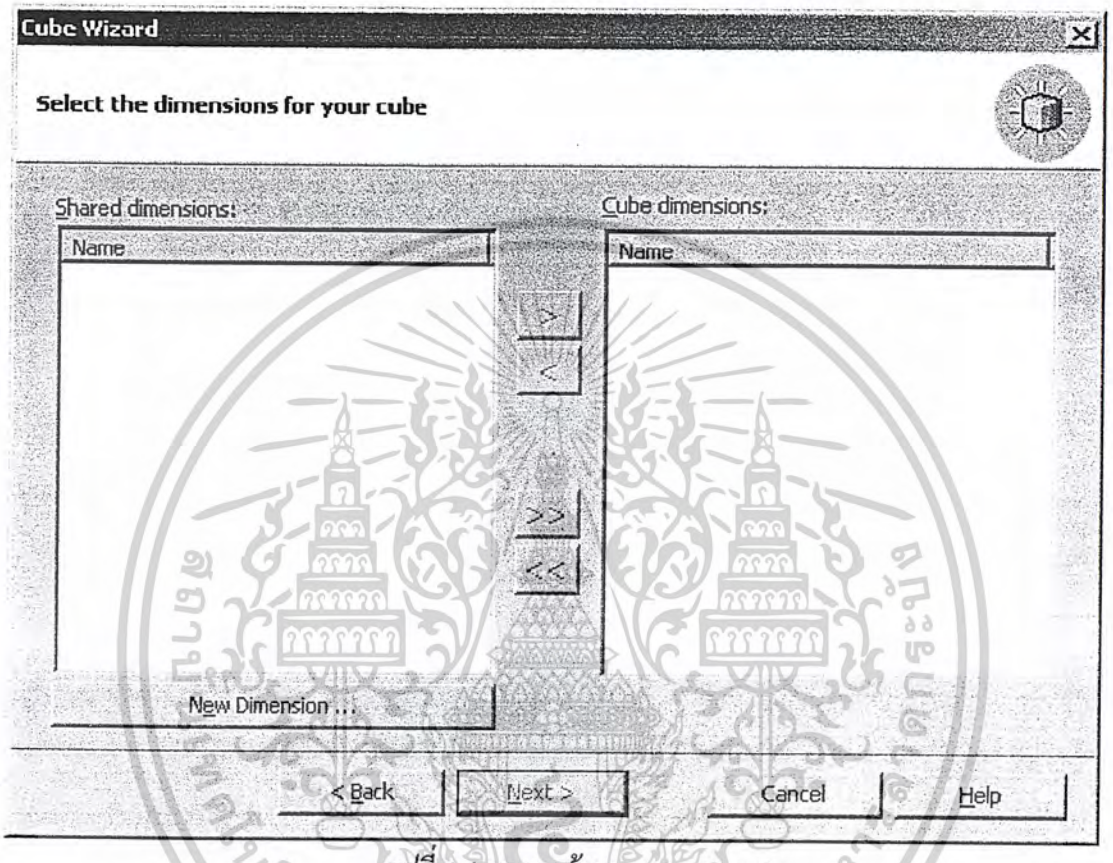
4.2.2.1 วิธีการสร้างตัววัด (Measures)

1. ในหน้าต่าง **Welcome** ของ Cube Wizard ให้คลิก **Next**
2. ในหน้าต่าง **Select a fact table from a data source** ให้ขยายที่ฐานข้อมูล **Tutorial**, และคลิกขวาที่ Cube folder คลิก **sale_fact_1998** (เลือกเป็น ตารางแฟ็ค)
3. สามารถ view ข้อมูลในตาราง **Sales_fact_1998** โดยการคลิก **Browse Data** หลังจากปิด Browse Data แล้วให้ คลิก **Next**

4. กำหนดตัววัดสำหรับคิวบ์ ใต้ **Fact Table numeric columns,** คับเบิลคลิกเลือก **Store_sales, store_cost, unit sales** และ คลิก **Next.**

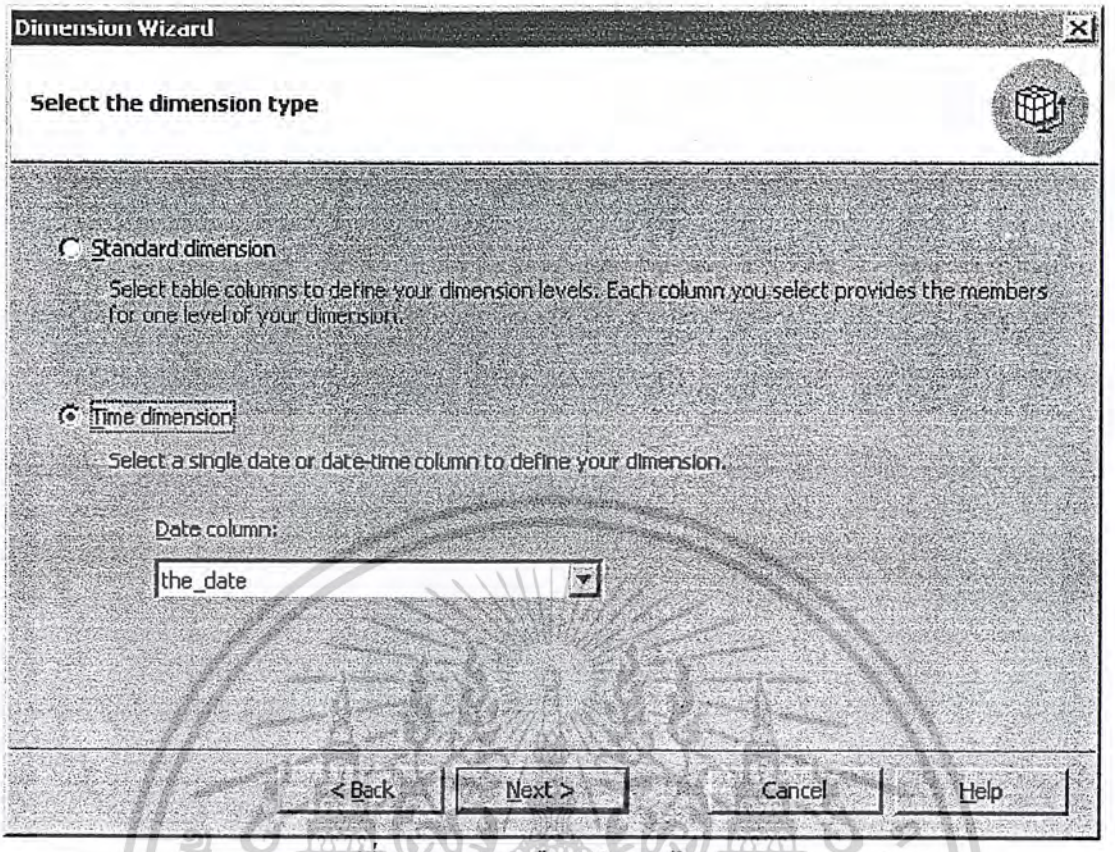
4.2.2.2 วิธีสร้างไดเมนชัน Time

1. ในหน้าต่าง **Select the dimensions for your cube** ของ Cube Wizard , คลิก **New Dimension** ซึ่งมันจะไปเรียก **Dimension Wizard**



รูปที่ 4-6 แสดงหน้าจอ Cube Wizard

2. ที่ขั้นตอน Welcome คลิก **Next**
3. ที่ขั้นตอน **Choose how you want to create dimension** เลือก **Star Schema: A single dimension table,** และ คลิก **Next**
4. ที่ขั้นตอน **Select dimension table** เลือก **time_by_day** คลิก **Next**
5. ที่ขั้นตอน **Select the dimension type,** เลือก **Time dimension,** และ คลิก **Next**

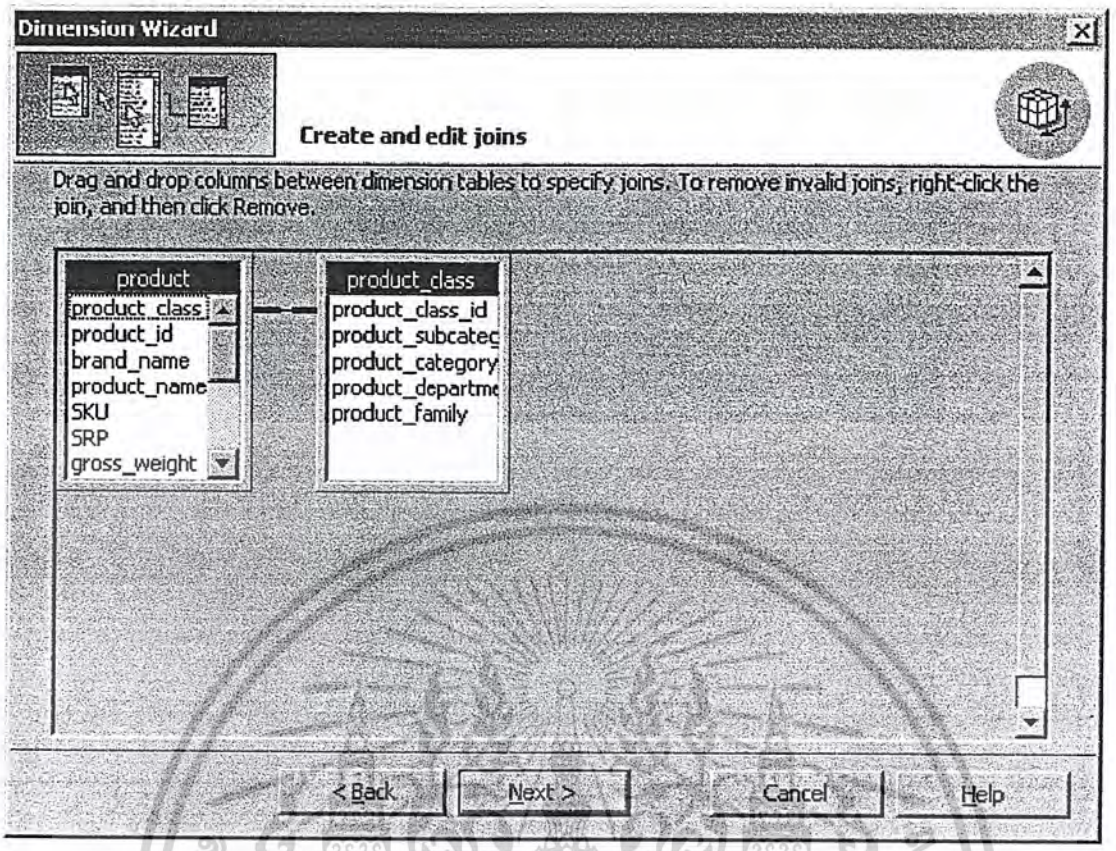


รูปที่ 4-7 แสดงการเลือกชนิดของไดเมนชัน

5. ถัดไป ทำการกำหนด Level สำหรับไดเมนชัน ที่ขั้นตอน **Create the time dimension levels** คลิก **select time levels**, เลือก **Year, Quarter, Month** และคลิก Next
6. ที่ขั้นตอน **Select advanced option** คลิก **Next**
7. ที่ Wizard สุดท้ายใส่ ช่องของ New dimension ให้ใส่ Time คลิก **finish** เพื่อกลับที่ **Cube Wizard**

4.2.2.3 วิธีสร้างไดเมนชัน Product

1. คลิกเลือกที่ **New Dimension** อีกครั้ง ที่ขั้นตอน **Welcome to the Dimension** คลิก **Next**
2. ที่ขั้นตอน **Choose how you want to create the dimension** เลือก **Snowflake schema:Multiple, related dimension tables** หลังจากนั้น คลิก **Next**
3. ที่ขั้นตอน **Select the dimension tables** คับเบิลคลิก **product** และ **product_class** ย้ายไปยัง **Selected tables** หลังจากนั้นคลิก **Next**
4. สองตารางที่ได้เลือกไว้ที่ขั้นตอนก่อนหน้านี้ จะได้ถูกเชื่อต่อระหว่างกันและจะถูกแสดงไว้ที่ขั้นตอน **Create and edit joins ของ Dimension Wizard** หลังจากนั้นคลิก **Next**
5. กำหนด Level สำหรับ **Product Dimension** โดยให้ดับเบิลคลิกไปยังที่ **Product_category, Product_subcategory, Brand_name** ตามลำดับหลังจากนั้นคลิก **Next**



รูปที่ 4-8 แสดงการเชื่อมต่อของ 2 ตารางเมื่อทำ Snowflake schema

6. ที่ขั้นตอน **Specify the member key columns** คลิก **Next**
7. ที่ขั้นตอน **Select advanced options** คลิก **Next**
8. ที่ขั้นตอนสุดท้ายใส่ชื่อ **Product** ที่ **Dimension name** แล้วก็ให้เลือกช่อง **Share this dimension with other cubes** ออกไปจากนั้นคลิก **Finish**
9. คุณสามารถเห็น **Product** Dimension ที่ **Cube dimension** list

4.2.2.4 สร้างไดเมนชัน Store

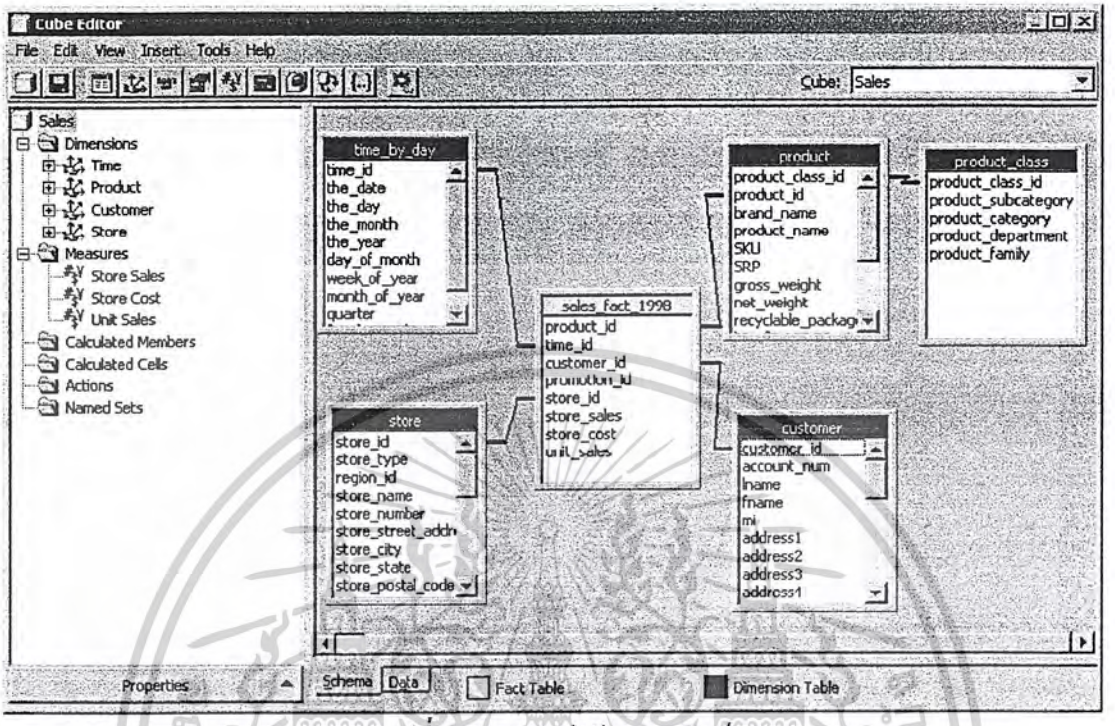
ทำการสร้างไดเมนชัน Store เข้าไปที่ **New dimension** เลือก **A single Dimension table (flat of star schema)** จากนั้นเลือก **Store** ส่วน **Levels** ให้เลือกเป็น Store_country, store_state, store_city, และ **store_name columns** ตามลำดับ แล้วตั้งชื่อไดเมนชันนี้ว่า "Store" แล้วก็ให้เลือกช่อง Share this dimension with other cubes ออกไป เช่นกัน

4.2.2.5 สร้างไดเมนชัน Promotion

ทำการสร้างไดเมนชัน Promotion เข้าไปที่ **New dimension** เลือก **A Single Dimension table (flat of star schema)** จากนั้นให้ทำการเลือก **Promotion** ส่วน **Levels** ให้เลือกเป็น **Media_type** และ **promotion_name columns** ตามลำดับ แล้วตั้งชื่อไดเมนชันนี้ว่า **"Promotion"** แล้วก็เลือกช่อง **Share this dimension with other cubes** ออกไป เช่นกัน

4.2.2.6 สร้างไดเมนชัน Customer

ทำเหมือนๆ ไดเมนชันอื่นๆ ก่อนหน้านี้ โดยเรียง Level ดังนี้ **Country, State province, city, name**



รูปที่ 4-9 แสดงรูปสกีมาของคิวบ์ Sales

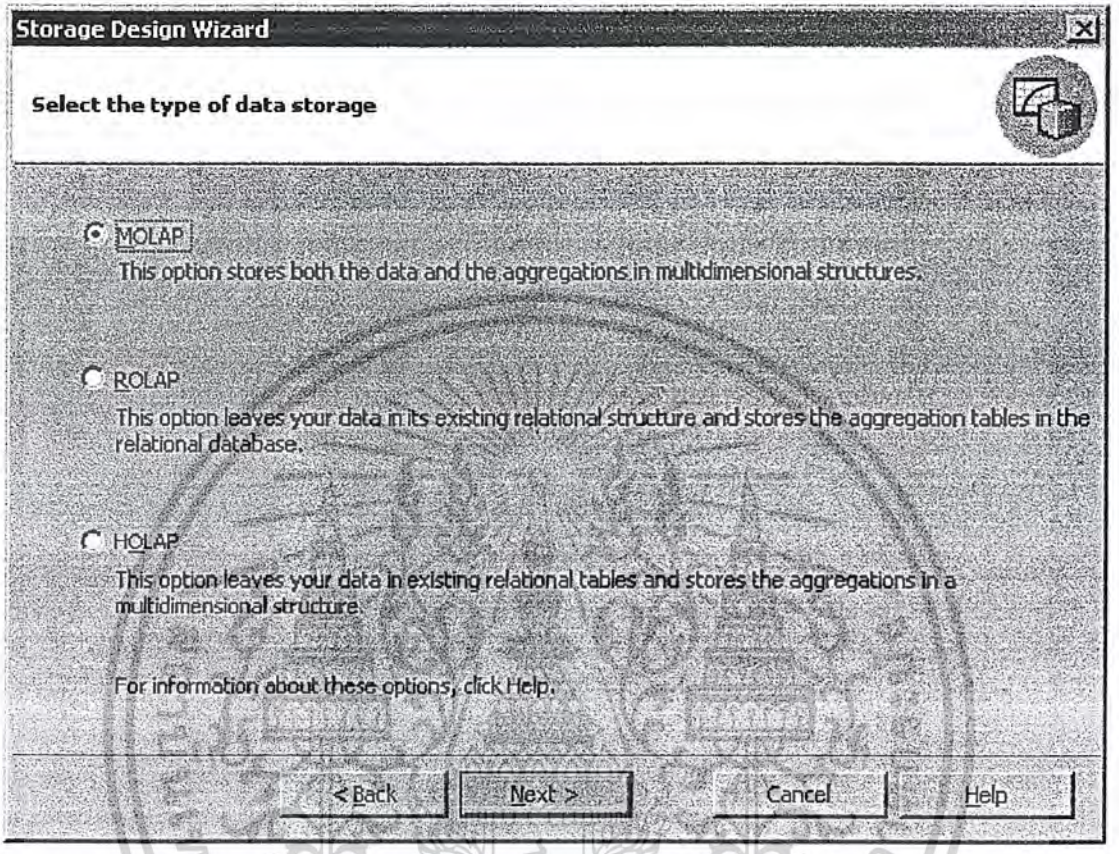
ทั้ง **Time, Product, Store** และ **Promotion Dimension** ที่เลือกมาทั้งหมดนั้น จะถูกเก็บอยู่ในสิสต์ของ Cube Dimension เป็นอันเสร็จสิ้นขั้นตอนของการสร้างคิวบ์ ให้ตั้งชื่อคิวบ์ นี้ว่า "**Sales**"

- การแก้ไขคิวบ์ โดยใช้ Cube editor wizard ในหน้าต่างที่แสดง Schema ของ Cube editor เราจะเห็นตารางแฟ็ค เป็นสีเหลือง ส่วนไดเมนชันที่มาเชื่อมต่อ จะเป็นสีน้ำเงิน และใน Cube editor tree view จะแสดงโครงสร้างของคิวบ์ในแบบลำดับขั้นต้นไม้ (Hierarchical Tree)

ใน Cube editor นี้ สามารถที่จะสร้างไดเมนชันที่ต้องการใช้เพิ่มได้ แต่ไดเมนชันที่สร้างเพิ่มนี้จะเป็น Private dimension คือ ใช้ได้เฉพาะในคิวบ์ที่กำลังใช้งานอยู่เท่านั้น ไม่สามารถใช้ร่วมกับคิวบ์ อันอื่นได้

4.2.3 การออกแบบการจัดเก็บข้อมูลและการประมวลผลคิวบ์ (Designing Storage and Processing Cube)

Analysis Services นั้น เราสามารถปรับปรุงประสิทธิภาพและเวลาที่ใช้ในการคิวรีได้ โดยใช้ **Storage Design Wizard** ระบุวิธีการจัดเก็บได้ 3 ชนิด คือ **MOLAP, ROLAP หรือ HOLAP**



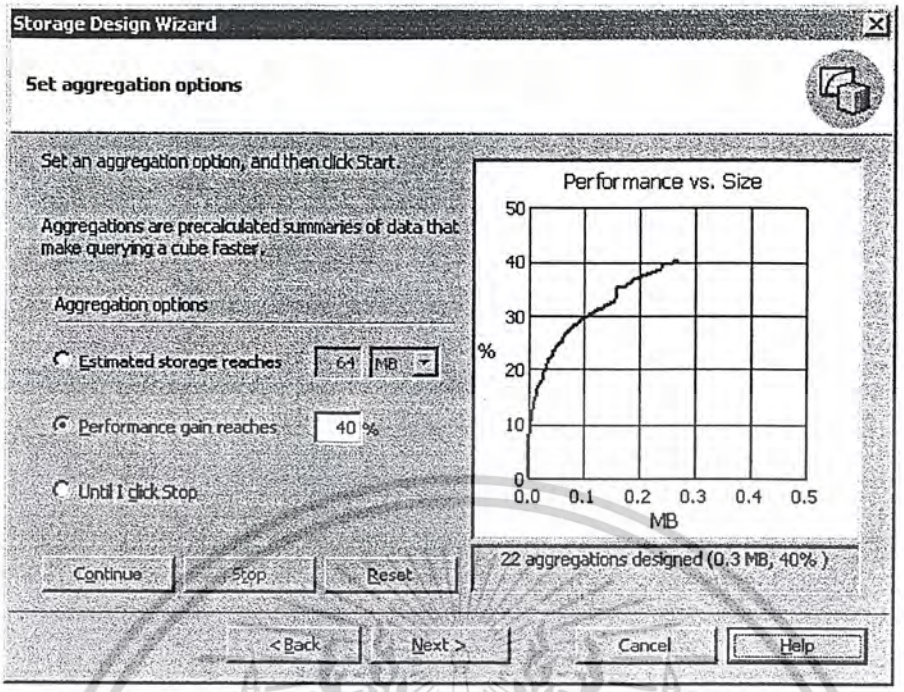
รูปที่ 4-10 เลือกวิธีการจัดเก็บข้อมูล

MOLAP เป็นโหมดที่มีการทำการรวมค่าข้อมูลและก๊อปปี้ข้อมูลไปเก็บไว้ในโครงสร้างแบบมัลติไดเมนชันแนลบนเครื่อง Analysis Server

ROLAP ใช้วิธีการเก็บข้อมูลที่เป็นการรวมค่าไว้ในตารางของรีเลชันแนลดาต้าเบส (Relational Database)

HOLAP เป็นการรวมวิธีของ MOLAP และ ROLAP เข้าด้วยกัน ส่วนที่เหมือน MOLAP คือจะเก็บข้อมูลที่เป็นการรวมค่าไว้ในโครงสร้างแบบมัลติไดเมนชันแนลบนเครื่อง Analysis Server แต่ข้อมูลจริงจะไม่ได้ถูกเก็บไว้ด้วย

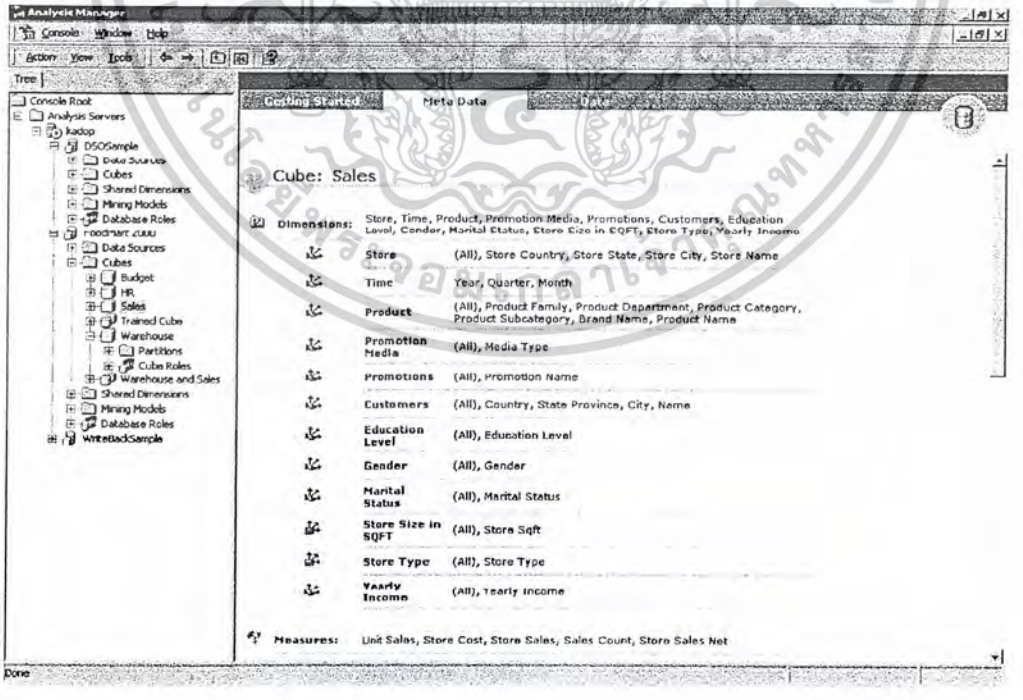
หลังจากที่เลือกชนิดของการจัดเก็บข้อมูลแล้ว สามารถดูกราฟที่แสดงถึงประสิทธิภาพ (Performance) และขนาด (Size) ของคิวบ์ แสดงได้ดังรูปที่ 4-11



รูปที่ 4-11 แสดงการวัดประสิทธิภาพและขนาดของ Cube ที่สร้างขึ้นมา

4.2.4 การดูข้อมูลรายละเอียดของคิวบ์ (Viewing Metadata for Cube)

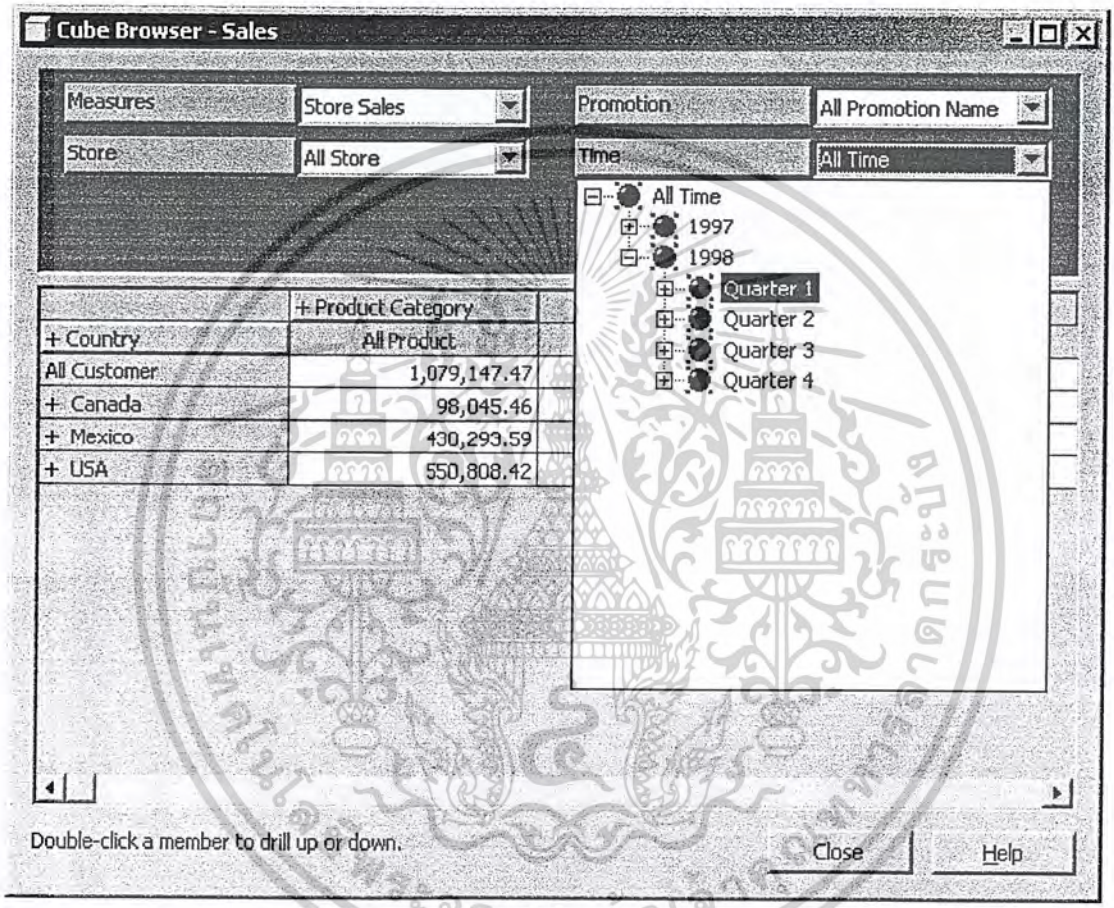
Analysis Services สามารถดูรายละเอียดข้อมูลที่มีอยู่ในคิวบ์ หรือเมต้าดาต้า (Metadata) ได้แก่ Dimensions, Measures, Calculated Members, Source Tables, Processed, Type, Size และ Data Source



รูปที่ 4-12 แสดงรายละเอียดของข้อมูล (Meta data)

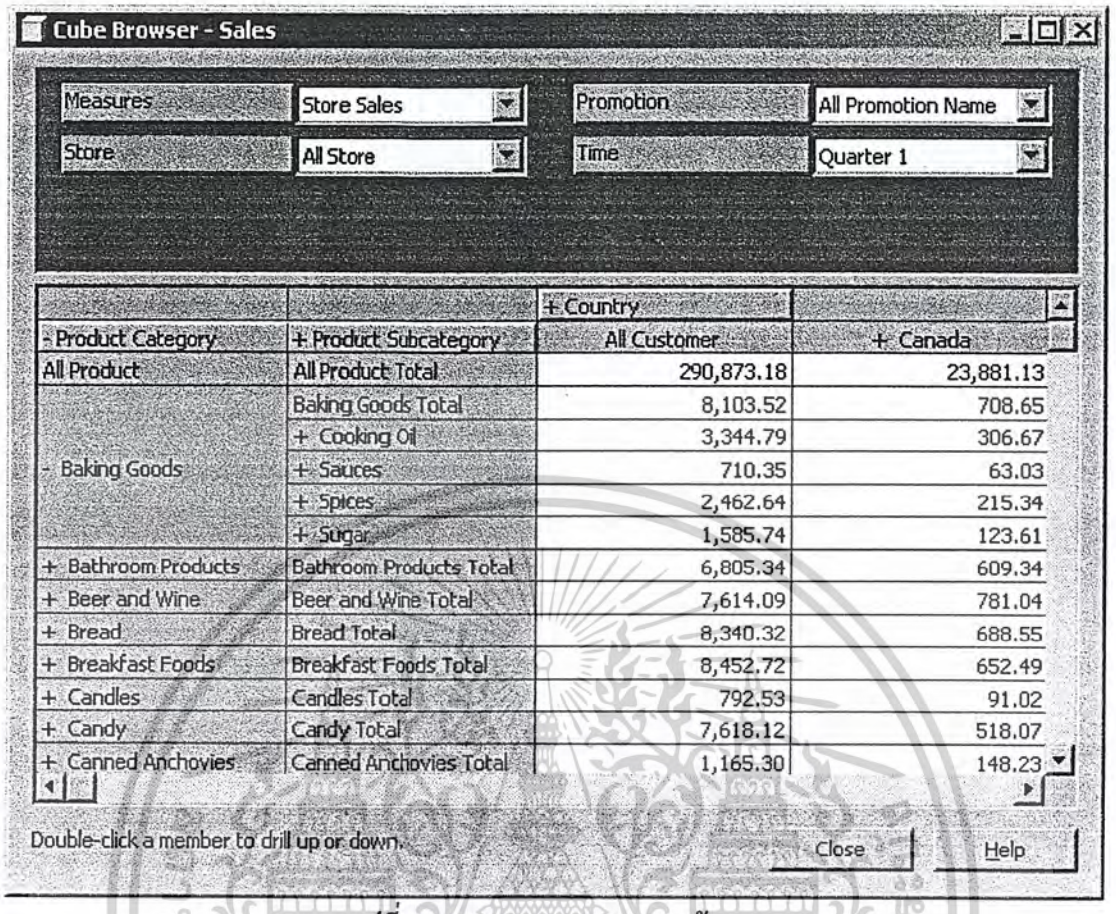
4.2.5 การดูข้อมูลในคิวบ์ (Browse Cube)

สามารถใช้ Cube Browser เพื่อดูข้อมูลในลักษณะต่าง ๆ กันได้ สามารถใส่จำนวนไดเมนชันที่จะดูข้อมูลและสามารถที่จะ Drill down ในรายละเอียดที่มากขึ้นได้ด้วยที่หน้าต่าง Cube Browser และยังสามารถเพิ่มไดเมนชันที่ต้องการดู เข้าไปแทนที่ไดเมนชันที่แสดงอยู่ในกริด (Grid) ได้ เพียงแค่ลากไดเมนชันจากส่วนของพาเลต (Palette) ด้านบน ลงมาวางแทนไดเมนชันในกริด ส่วนไดเมนชันที่ถูกแทนที่ก็จะถูกนำขึ้นไปเก็บไว้ที่ Palette แทน ในส่วนของ Time Dimension เราสามารถเลือกดูลักษณะของเวลาได้ทั้งแบบ Year, Quarter หรือ Month



รูปที่ 4-13 แสดงการเลือก Level เพื่อ Slice และ Dice ข้อมูล

ส่วนการทำ Drill down ข้อมูล ก็สามารถทำได้ง่ายเช่นกัน เพียงแค่ดับเบิลคลิกที่เซลล์ ในกริดเพียงเท่านี้เราก็จะเห็นรายละเอียดของข้อมูลในระดับที่ลึกลงไป แสดงได้ดังรูปที่ 4-14 ส่วนการทำ Drill Up ข้อมูล ก็ทำตรงกันข้ามกับการ Drill down คือ ให้ดับเบิลคลิกที่เซลล์ที่มีเครื่องหมาย – อยู่ข้างหน้า เราก็จะเห็นข้อมูลในระดับที่สูงขึ้นมา 1 ระดับ



รูปที่ 4-14 แสดงการ Drill down ข้อมูล

บทที่ 5

Multidimensional Expression (MDX)

5.1 แนะนำ MDX

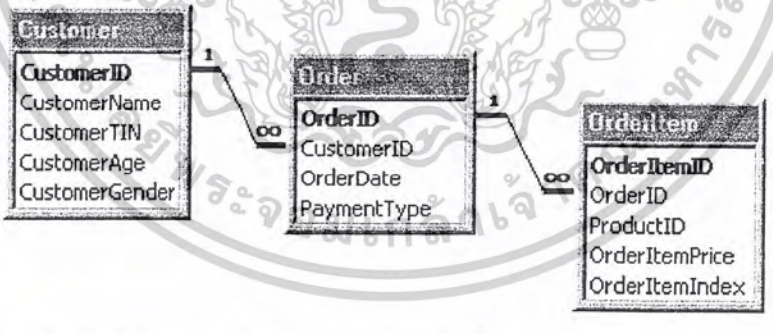
MDX หรือ Multidimensional Expression เป็นภาษาที่ใช้ในการกำหนดโครงสร้างและปฏิบัติการกับข้อมูลที่มีลักษณะเป็นมัลติไดเมนชันแนล MDX นี้จะมีหลายส่วนที่คล้ายคลึงกับภาษา SQL แต่ MDX ไม่ใช่ส่วนขยายของ SQL แต่อย่างใด บางฟังก์ชันที่ทำใน MDX ได้ก็อาจจะสามารถทำใน SQL ได้เช่นเดียวกันแต่อาจจะทำได้อย่างไม่มีประสิทธิภาพเท่าใดนัก

เช่นเดียวกันกับภาษา SQL ,. ภาษา MDX ก็ต้องใช้คีย์เวิร์ด SELECT , FROM ,WHERE ด้วยเช่นกัน โดยคีย์เวิร์ดต่างๆ ใน MDX นี้จะใช้ในการดึงข้อมูลจากคิวบ์เพื่อใช้ทำการวิเคราะห์ MDX ก็จะมีรูปแบบของ Data Definition Language (DDL) สำหรับจัดการ โครงสร้างของข้อมูล ซึ่งก็จะมีคำสั่งที่ใช้ในการสร้างหรือลบคิวบ์, ไดเมนชัน, ตัววัด, และออบเจ็กต์ อื่นๆ

5.2 คำที่ควรรู้จักใน MDX

5.2.1 Dimensions, Levels, Members, and Measures

เกือบจะทุกภาษาที่ใช้เกี่ยวกับฐานข้อมูลสำหรับกำหนดโครงสร้างและจัดการข้อมูล เช่น SQL นั้น ได้ถูกออกแบบมาเพื่อดึงข้อมูลออกมาในลักษณะสองไดเมนชัน นั่นคือ เป็น ไดเมนชันของคอลัมน์ และ ไดเมนชันของแถว (row) รูปข้างล่างนี้จะแสดงลักษณะการใช้งานของรีเลชันแนลดาต้าเบสทั่วๆ ไป

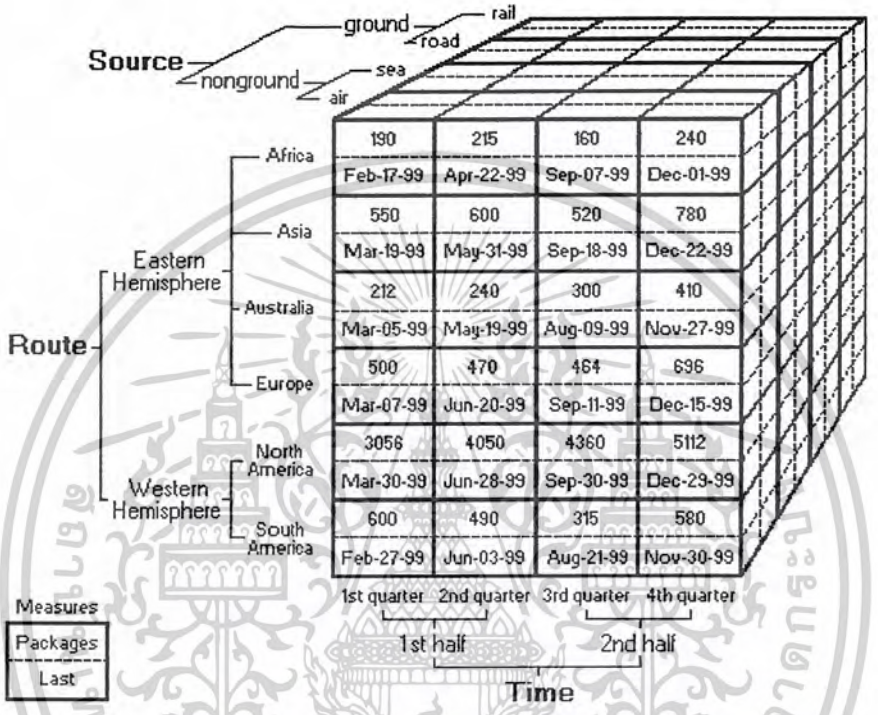


รูปที่ 5-1 แสดงโครงสร้างของรีเลชันแนลดาต้าเบส (relational database)

จากรูปแต่ละตารางจะแสดงข้อมูลในลักษณะสองไดเมนชัน โดยใน SQL นั้นเราสามารถที่จะระบุคอลัมน์ที่ต้องการได้โดยใช้คีย์เวิร์ด SELECT ในขณะที่เราสามารถกำหนดจำนวนแถวที่ต้องการให้ส่งกลับออกมาได้โดยใส่เงื่อนไขไว้ใน WHERE

ในทางตรงกันข้ามกับข้อมูลที่มีแบบมัลติไดเมนชันแนล (Multidimensional data) ซึ่งข้อมูลสามารถจัดโครงสร้างได้มากกว่าสองไดเมนชัน โครงสร้างในลักษณะนี้จะถูกเรียกว่าคิวบ์ที่จุดอินเตอร์เซค

(intersect) กันของไดเมนชันทั้งหลายภายในคิวบ์ อาจจะมีข้อมูลมากกว่า 1 อิลิเมนท์ (element) ซึ่งจะเรียกว่าตัววัด (measure) รูปข้างที่ ก-2 จะแสดงคิวบ์ซึ่งประกอบไปด้วย 3 ไดเมนชัน คือ Route, Service และ Time และประกอบไปด้วย 2 ตัววัด คือ Packages และ Last ซึ่งแต่ละไดเมนชันสามารถแตกย่อยลงไปได้หลายเลเวล โดยที่แต่ละเลเวลก็ยังสามารถแตกย่อยลงไปได้ถึงระดับสมาชิก (members) ได้เช่นกัน ตัวอย่างเช่น ไดเมนชัน Source เราสามารถเลือกสนใจในระดับเลเวลที่เป็น Eastern Hemisphere ซึ่งยังสามารถแตกย่อยได้เป็น 4 สมาชิก คือ Africa, Asia, Australia และ Europe



รูปที่ 5-2 แสดงโครงสร้างของคิวบ์

การจะเข้าใจ MDX ได้ จำเป็นอย่างยิ่งที่จะต้องรู้และเข้าใจคอนเซปต์ของคิวบ์ (Cube), ไดเมนชัน (Dimension), เลเวล (Level), สมาชิก (Members) และตัววัด (Measures)

5.2.2 Cell, Tuples, and Sets

ใน SQL จะรีเทิร์นซับเซ็ตของข้อมูลจากตารางในลักษณะ 2 ไดเมนชัน ในขณะที่ MDX จะรีเทิร์นซับเซ็ตของข้อมูลจากคิวบ์ในลักษณะมัลติไดเมนชันแนล (multidimensional data)

ในรูปที่ 5-2 ได้แสดงให้เห็นว่าที่จุดอินเตอร์เซ็คของ multidimensional members จะทำให้เกิด cells ซึ่งจะมีข้อมูลอยู่ภายใน ในการที่จะระบุและดึงข้อมูลนั้น ใน MDX จะใช้สิ่งที่เรียกว่า tuples, Tuples จะแสดง dimensions และ members ในการที่จะระบุ cell ใดเซลล์หนึ่ง และด้วยสาเหตุที่ว่า cell คือจุดตัดของไดเมนชันต่างๆ ภายในคิวบ์ดังนั้นจึงทำให้ tuple สามารถระบุเซลล์ทุกเซลล์ภายในคิวบ์ได้อย่างเป็นเอกลักษณ์ (unique) ส่วน measures จะถูกจัดการเป็นลักษณะของ private dimension ที่มีชื่อว่า

Measures ดังนั้นในการอ้างถึง measures เราก็จะไปอ้างข้อมูลจากไดเมนชัน Measures ตัวอย่างการใช้งาน เช่น

(Source.[EasternHemisphere].Africa,Time.[2ndhalf].[4thquarter],Route.Air,
Measures.Packages)

tuple ไม่จำเป็นที่ต้องเกิดจากการระบุรวมทุกไดเมนชันภายใน cube ก็ได้ เช่นตัวอย่างต่อไปนี้

(Source.[Eastern Hemisphere])
(Time.[2nd half], Source.[Western Hemisphere])

การรวม tuple เข้าด้วยกัน เราจะเรียกว่าเซต (set) ในภาษา MDX . ไดเมนชันที่เป็น axis และ slicer จะประกอบไปด้วยเซตของ tuple ตัวอย่างของเซตของ tuple เช่น

{ (Time.[1st half].[1st quarter]), Time.[2nd half].[3rd quarter]) }

นอกจากนี้ เรายังสามารถสร้าง named set ได้ด้วย ซึ่ง named set ก็คือเซตที่ได้มีการตั้งชื่อเอาไว้ เพื่อทำให้ MDX query สามารถเข้าใจได้ง่ายขึ้น

5.2.3 Axis and Slicer Dimensions

ในภาษา SQL เรามักจะต้องกำหนดจำนวนข้อมูลที่รีเทิร์นกลับมาจากตารางเสมอ โดยการใช้คีย์เวิร์ด SELECT ในการกำหนดจำนวนคอลัมน์ที่รีเทิร์น และใช้คีย์เวิร์ด WHERE ในการระบุจำนวน row ที่รีเทิร์น

ในภาษา MDX ก็จะใช้หลักการในลักษณะเดียวกันนี้ นั่นคือ SELECT statement ใช้สำหรับเลือก dimensions และ members ที่จะรีเทิร์นกลับมา เราจะเรียกนี้ว่า **axis dimensions** ในขณะที่ WHERE statement จะใช้ในการกำหนดจำนวนข้อมูลที่รีเทิร์นกลับมา โดยจะให้รีเทิร์นกลับมาเฉพาะ dimension และ member ที่ตรงตามเงื่อนไขของเรา ซึ่งเราจะเรียกนี้ว่า **slicer dimension**
Axis dimension ถูกคาดการว่าจะรีเทิร์นข้อมูลออกมาในลักษณะของหลาย members ในขณะที่ slicer dimension ถูกคาดการว่าจะรีเทิร์นข้อมูลออกมาในลักษณะของ member เดียว

5.2.4 Calculated Members

Calculaed members คือ members ที่ไม่ใช่ตัวข้อมูลที่เก็บอยู่ แต่เป็นสมการการคำนวณโดย ใช้ฟังก์ชันต่างๆ ที่มีมาพร้อมกับ MDX

5.3 เปรียบเทียบ SQL กับ MDX

ถ้ามองโดยผิวเผินแล้ว MDX จะมีหลายๆ อย่างซึ่งคล้ายกับ SQL หลายๆ ฟังก์ชันที่มีปรากฏใน MDX ก็จะมีลักษณะคล้ายๆ กับฟังก์ชันที่ปรากฏใน SQL เช่นกัน อย่างไรก็ตาม ข้อแตกต่างระหว่าง MDX และ SQL ก็มีอยู่มาก ดังนั้นจำเป็นที่ต้องคำนึงถึงความแตกต่างในระดับคอนเซปต์นี้ด้วย สิ่งหนึ่งที่จะแสดงให้เห็นถึงความไม่เหมือนกันระหว่าง SQL กับ MDX ก็คือ ความสามารถในการอ้างถึงข้อมูล โดย MDX จะอ้างถึงข้อมูลที่เป็นที่มีลักษณะหลายๆ ไดเมนชัน ถึงแม้ว่าเราจะสามารถใช้ SQL ในการคิวรีคิวบ์บน Microsoft SQL Server2000 Analysis Services ได้ก็ตาม แต่เราจะสามารถทำคิวรีได้อย่างมีประสิทธิภาพกว่าเมื่อ เราใช้ MDX เนื่องจากว่า MDX ได้ถูกออกแบบมาเป็นพิเศษสำหรับการดึงข้อมูลที่มีลักษณะเป็นแบบมัลติไดเมนชันแนล ซึ่งจะประกอบไปด้วยหลายไดเมนชัน

เมื่อเราทำการคิวรี ใน SQL จะอ้างถึงข้อมูลเพียงแค่สองไดเมนชัน คือ row และ column ในขณะที่ MDX สามารถทำได้ทั้ง หนึ่ง, สอง หรือมากกว่า โดยคำว่า row และ column จะเปรียบได้กับชื่อของ axis dimension สองอันแรกใน MDX ส่วนไดเมนชันอื่นๆ ก็จะมีชื่อต่างๆ กันไป ในปัจจุบันนี้ทูลหลายๆ ตัวที่ใช้สำหรับ OLAP จะมีความสามารถแสดงผลได้เพียงแค่สองไดเมนชันเท่านั้น

ในภาษา SQL , เราจะใช้ SELECT ในการกำหนด column ที่ต้องการให้รีเทิร์นกลับออกมา และใช้ WHERE เพื่อระบุ row ที่ต้องการ เมื่อมาเปรียบกับ MDX , SELECT จะเอาไว้ใช้กำหนดจำนวน axis dimensions ในขณะที่ WHERE เอาไว้สำหรับกำหนดเงื่อนไขเลือก dimension หรือ member ที่เราต้องการ

ในภาษา SQL , เราจะใช้ WHERE clause ในการกรองข้อมูลที่ต้องการให้รีเทิร์นออกมา ในขณะที่เราใช้ WHERE clause เพื่อให้เกิดการตัดบางส่วนของข้อมูล (slice) ที่จะรีเทิร์นออกมาใน MDX เราจะเห็นได้ว่าทั้งสองอย่างนี้จะมีความคล้ายคลึงกันในระดับคอนเซปต์ แต่จำไว้ว่ามันไม่เหมือนกัน

5.4 เริ่มต้นใช้งาน MDX

โครงสร้างพื้นฐานของภาษา MDX ที่เราจะใช้ในการคิวรีเป็นดังนี้

```
SELECT [<axis_specification>
        [, <axis_specification>... ]]
FROM   [<cube_specificaion>]
[WHERE [<slicer_specification>] ]
```

5.4.1 โครงสร้างของ SELECT statement ใน MDX

ในภาษา MDX จะใช้ประโยคคำสั่ง SELECT ในการระบุคาต้าเซต (data set) ซึ่งจะเก็บซับเซตของข้อมูลมัลติไดเมนชันแนล ในการระบุคาต้าเซต เราจำเป็นต้องกำหนดข้อมูลต่อไปนี้

- จำนวนของ axis ซึ่ง MDX ได้กำหนดไว้ว่าสามารถใส่ได้สูงสุดถึง 128 axis
- Members จากแต่ละไดเมนชัน ที่เราต้องการจะรวมเข้าไปไว้ใน axis
- ชื่อคิวบ์ที่ต้องการจะทำการคิวรี
- Members ใน slicer dimension ซึ่งเอาไว้สำหรับตัดข้อมูลเฉพาะส่วนที่ต้องการ

ตัวอย่าง เช่น

```
SELECT
  { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,
  { [Time].[1997], [Time].[1998] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

SELECT - ใช้เพื่อกำหนด axis dimension ตัวอย่างข้างต้นนี้ได้กำหนดไว้ 2 axis dimension

FROM - ใช้เพื่อเลือกแหล่งข้อมูลที่เราต้องการ จะดึงข้อมูลมาใช้ โดยแหล่งข้อมูลนี้จะเก็บข้อมูลในลักษณะของมัลติไดเมนชันแนล

WHERE - อาจจะใส่หรือไม่ใส่ก็ได้ โดยเอาไว้สำหรับกำหนด slicer dimension ในตัวอย่างข้างต้นเราได้เลือก member ของไดเมนชัน Store มาทำเป็น slicer dimension

นอกจากนี้ใน SELECT statement เราอาจใช้คีย์เวิร์ด WITH รวมทั้งฟังก์ชันต่างๆ ที่มีมาใน MDX ด้วยก็ได้ Syntax ของ MDX ดูๆ แล้วจะคล้ายกับใน SQL แต่อย่างไรก็ตาม ยังมีส่วนที่ไม่เหมือนกันดังนี้

- MDX จะแยกเซตของ tuples หรือ members ออกจากกันโดยใช้ เครื่องหมาย {}
- MDX สามารถกำหนด axis dimension ได้สูงสุดถึง 128 แกน แต่เฉพาะ 5 axis แรกเท่านั้นที่ได้กำหนดชื่อเรียกไว้แล้ว เราสามารถที่จะอ้างถึง axis ได้โดยอาจจะใช้ชื่อเรียกที่กำหนดเอาไว้แล้ว หรืออีกวิธีหนึ่งคือใช้ลำดับในการระบุ ตัวอย่างเช่น เราสามารถที่จะใช้คิวรีนี้แทนคิวรีข้างบนได้ โดยที่ผลลัพธ์เหมือนกัน

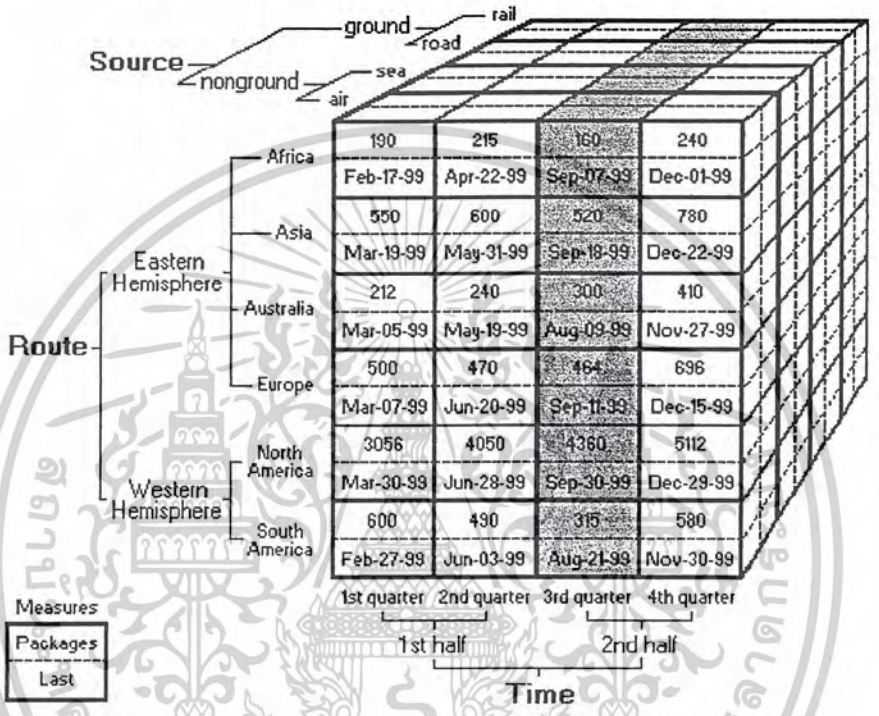
  ```
  SELECT
    { [Measures].[Unit Sales], [Measures].[Store Sales] } ON AXIS(0),
    { [Time].[1997], [Time].[1998] } ON AXIS(1)
  FROM Sales
  WHERE ( [Store].[USA].[CA] )
  ```

- ใน MDX, FROM จะเอาไว้อ้างถึงแหล่งข้อมูลเช่นเดียวกันกับใน SQL แต่อย่างไรก็ตาม มันจะไม่เหมือนกับ SQL ตรงที่ FROM clause ใน MDX ถูกกำหนดให้ใช้กับเพียงคิวบ์เดียว แต่เราก็ยังสามารถที่จะดึงข้อมูลจากคิวบ์อื่นได้โดยใช้ฟังก์ชัน LookupCube
- WHERE clause ใช้ระบุ slicer dimension ถ้าเราไม่ได้มีการระบุไดเมนชันไว้ใน WHERE clause , Microsoft SQL Server2000 Analysis Services จะทำงานโดยนำทุกไดเมนชันที่เราไม่ได้ใช้เป็น axis dimension มาใช้เป็น slicer dimension และ member ที่จะกำหนดให้กับแต่ละไดเมนชันนั้นก็จะกำหนดโดยใช้ค่าดีฟอลต์(default)

5.4.2 Members, Tuples, and Sets

ก่อนที่เราจะทำคิวรีโดยใช้ MDX จำเป็นที่จะต้องเข้าใจความหมายของ members, tuples, และเซตเสียก่อน เนื่องจากจะมีการอ้างถึง element เหล่านี้

Member คือ ส่วนหนึ่งในไดเมนชันที่แสดงถึงข้อมูล โดยอาจจะแสดงข้อมูลเดียวหรือมากกว่า โดยอาจคิดได้ว่า member ในไดเมนชันเปรียบเสมือน เรคอร์ดหนึ่ง (หรือมากกว่า) ที่อยู่ภายในฐานข้อมูล ทั่วๆไป member จะเป็นระดับล่างสุดเมื่อเรากำลังอ้างถึงเซลล์ของข้อมูลภายในคิวบ์ ตัวอย่างข้างล่างนี้ ในส่วนที่แรเงาจะแสดงถึง Time. [2nd half]. [3rd quarter] member



รูปที่ 5-3 แสดงส่วนของ member

เราจะใช้เครื่องหมาย [ ] คร่อมเมื่อชื่อของ member มีช่องว่างหรือตัวเลขรวมอยู่ด้วย ถึงแม้ Time จะเป็นคำธรรมดาที่ไม่ตัวเลขหรือช่องว่างรวมอยู่ แต่เราก็สามารถใส่ [ ] คร่อมได้เช่นกัน ดังเช่น [Time] . [2nd half]. [4th quarter]

5.4.3 Member Names และ Member Keys

เราสามารถอ้างถึง member ได้โดยใช้ member name หรือ member key ตัวอย่างข้างต้นที่ได้แสดงไปจะอ้างถึง member ในไดเมนชัน Time โดยใช้ member name (4th quarter) อีกวิธีที่จะใช้ในการอ้างถึง member คือการใช้ member key โดยใช้เครื่องหมาย & เพื่อเป็นการบ่งบอกว่าเป็น member key ดังตัวอย่างข้างต่อไปนี้

[Time] . [2nd half] . &[Q4]

โดยการใช้ member key ทำให้เราได้ member ที่ถูกต้องเนื่องจาก member name อาจจะมีซ้ำกัน หรือไดเมนชันอาจจะมีการเปลี่ยนแปลงไป

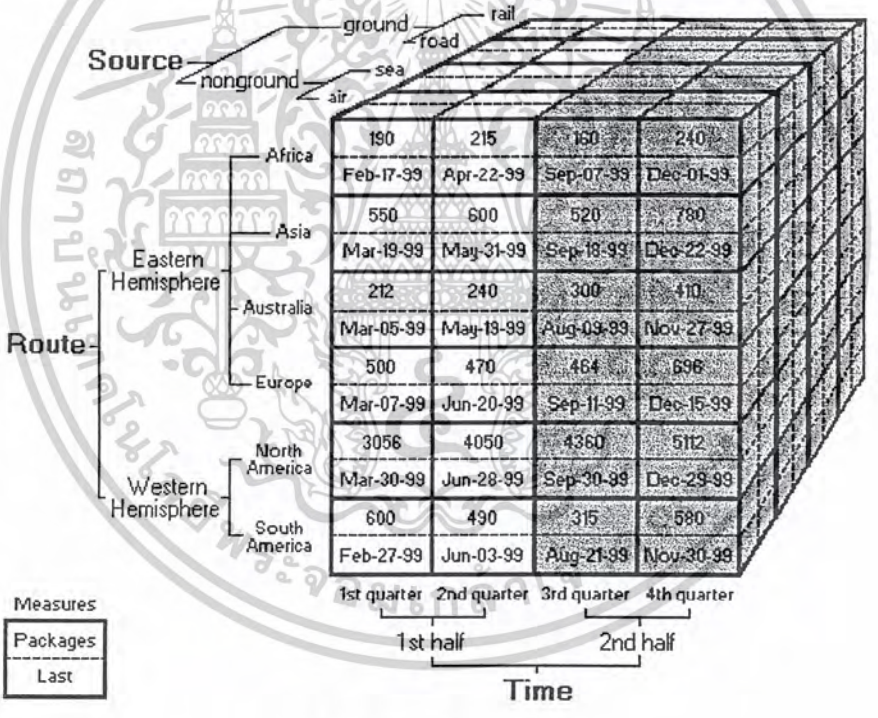
5.4.4 Calculated Members

เราสามารถสร้าง members ใน MDX query โดยให้รีเทิร์นข้อมูลที่เป็นผลมาจากการคำนวณแทนที่จะรีเทิร์นข้อมูลที่เก็บอยู่ในคิวบ์ได้ โดยเราจะเรียก member นี้ว่า calculated members ซึ่งทำให้เกิดความยืดหยุ่นในการใช้ MDX ในการที่จะสร้าง calculated member เราต้องใช้คีย์เวิร์ด WITH เข้ามาร่วมด้วย ตัวอย่างของการสร้าง calculated member แสดงได้ดังข้างล่างนี้

WITH MEMBER [Measures] . [PackagesForecast] As ' [Measures] . [Packages] * 1.1 '

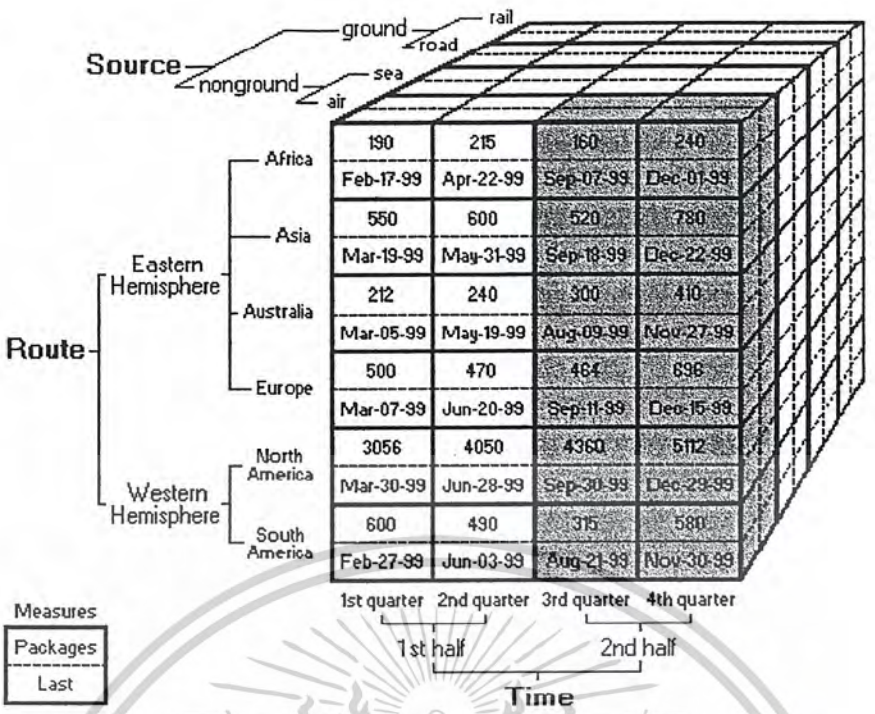
5.4.5 Tuples

ในการเลือกบางส่วนของข้อมูลจากคิวบ์เราต้องใช้ tuple ซึ่ง tuple ก็คือการนำ member จากหนึ่งไดเมนชันหรือมากกว่ามาประกอบเข้ากัน ตัวอย่างข้างล่างแสดงส่วนของ (Time . [2nd half]) tuple



รูปที่ 5-4 แสดงส่วนของ tuple

รูปข้างล่างนี้จะแสดงส่วนของ (Time . [2nd half] , Route.nonground.air) tuple



รูปที่ 5-5 แสดงส่วนของ tuple

ถ้า tuple มีเพียงหนึ่ง member ที่มาจากไดเมนชันเพียงไดเมนชันเดียว เราจะเรียกว่า simple tuple และเราสามารถเขียนในรูปแบบข้างล่างนี้ได้

Time . [2nd half]

แต่ถ้า tuple ประกอบมาด้วยมากกว่าหนึ่ง member และมาจากหลายไดเมนชันแล้ว tuple นี้จะต้องใส่วงเล็บ ( ) คร่อม member แสดงได้ดังนี้

(Time . [2nd half] , Route . nonground . air)

5.4.6 เซต (Sets)

เซต คือ การรวม tuple เข้าไว้ด้วยกัน อาจจะมีหนึ่ง tuple หรือมากกว่าหรือไม่มีเลยก็ได้เซตจะถูกใช้เพื่อกำหนด axis และ slicer dimension ในคิวรี MDX ตัวอย่างข้างล่างนี้แสดงเซตของ 2 tuple

{ (Time . [1st half] , Route . nonground . air) , (Time . [2nd half] , Route . nonground . sea) }

นอกจากนี้เซตยังสามารถประกอบไปด้วย tuple เดียวกันมากกว่าหนึ่ง tuple ก็ได้ ดังเช่น

{ Time . [2nd half] , Time . [2nd half] }

5.4.7 Set Functions

MDX ได้มีฟังก์ชันมาให้หลายฟังก์ชันในการรีเทิร์นเซต เช่น ใช้ colon (:) ในการแสดงแทนลำดับของ member ในการสร้างเซต จะได้

{ [1st quarter] : [4th quarter] }

ซึ่งจะให้ผลเหมือนกับ

{ [1st quarter] , [2nd quarter], [3rd quarter], [4th quarter] }

colon operator นั้นเป็นฟังก์ชันที่ทำงานในลักษณะนับรวมทั้งต้นและท้าย โดยที่ member ที่เป็นหัวและท้ายของฟังก์ชันก็จะถูกนับรวมเข้ามาเป็นเซตด้วย

5.4.8 Named Sets

Named set คือ เซตที่มีการตั้งชื่อเรียกไว้เรียบร้อยแล้ว named set จะใช้มากใน MDX เพื่อที่จะทำให้ง่ายต่อความเข้าใจและบำรุงรักษาง่าย นั่นคือ กลับมาดูอีกทีก็เข้าใจได้อย่างง่ายและถูกต้อง

5.4.9 Axis และ Slicer Dimensions

เมื่อเราทำการคิวรีโดยใช้ MDX , ลักษณะการทำงานของโปรแกรมคือ แบ่งเซตของไดเมนชันออกเป็นสองซับเซต คือ Axis dimension และ Slicer dimension

ด้วยการใช้ axis dimension และ slicer dimension ทำให้เราสามารถเลือกส่วนของข้อมูลที่ต้องการให้รีเทิร์นออกมาได้ ตัวอย่างเช่น สมมุติมีคิวบ์ชื่อ TestCube ซึ่งประกอบไปด้วยสองไดเมนชันคือ Route และ Time และ measures ของคิวบ์จะอยู่ใน ไดเมนชันที่ชื่อ Measures ดังนั้น cube นี้จึงมีทั้งหมดสามไดเมนชัน เมื่อเราต้องการ จะคิวรีเพื่อเปรียบเทียบ Packages measures ระหว่าง routes และ times ก็จะสามารถทำได้ดังนี้

```
SELECT
        { Route . nonground . Members } ON COLUMNS,
        { Time . [1st half] . Members } ON ROWS
FROM TestCube
WHERE ( [Measures] . [Packages] )
```

ในที่นี้เราใช้ฟังก์ชัน Members เข้ามาช่วย ทำให้เราไม่จำเป็นต้องเขียนทุก member ที่จะใช้ในการสร้าง set การทำงานของโปรแกรมคือ MDX จะเริ่มจากคำนวณหา axis และ slicer dimension ก่อนเป็นลำดับแรก แล้วจึงมาทำโครงสร้างของผลลัพธ์ที่ต้องรีเทิร์นไป จากนั้นค่อยดึงข้อมูลออกมาจากคิวบ์ที่เราทำการคิวรี

- **การกำหนด Axis dimension**

Axis dimension จะใช้เพื่อระบุเซ็ตของคำตอบ โดย MDX ใช้ SELECT clause ในการกำหนด axis dimension มีรูปแบบดังนี้

<axis_specification> :: = <set> ON <axis_name>

<axis_name> :: = COLUNMS | ROWS | PAGES | SECTIONS | CHAPTERS | AXIS(<index>)

โดย MDX อนุญาตให้เราสามารถกำหนด axis dimension ได้มากที่สุดถึง 128 axis แต่ในความเป็นจริงแล้วจะมีน้อยมากที่คิวรีโดยใช้มากกว่า 5 axis การระบุ axis ก็สามารถทำได้โดยใช้ axis name หรือใช้ axis index นั่นคือ AXIS(0), AXIS(1), AXIS(2), AXIS(3) และ AXIS(4) ก็เปรียบได้กับ COLUMNS, ROWS, PAGES, SECTIONS และ CHAPTERS ตามลำดับ ใน MDX ได้กำหนดไว้ว่าเราไม่สามารถข้าม axis ได้ นั่นคือ เราจะกำหนด COLUMNS และ PAGES โดยที่เราไม่กำหนด ROWS ไม่ได้

- **การกำหนด Slicer Dimension**

Slicer dimension ใช้สำหรับกรองเอาเฉพาะข้อมูลที่เราต้องการให้รีเทิร์นกลับมาเป็นผลลัพธ์ โดยเราจะระบุ slicer dimension ใน WHERE clause ของ MDX

โดยไดเมนชันที่ไม่ได้กำหนดเป็น axis dimension จะถูกนำมาใช้เป็น slicer dimension โดยจะใช้ค่าดีฟอล์ตในการระบุ member ซึ่งสามารถกำหนดได้ผ่านทางพร็อพเพอร์ตี้ของ Default Member ใน Analysis Manager (ปกติแล้ว ค่าดีฟอล์ตจะถูกตั้งไว้เป็น All member) รูปแบบของการกำหนด slicer dimension เป็นดังนี้

[WHERE [<slicer_specificaion>]]

slicer dimension จะรับเฉพาะรูปแบบของ expression ที่สามารถหาค่าเป็น tuple เดียวได้ ไม่ได้หมายความว่าเฉพาะ single tuple เท่านั้นที่จะสามารถใช้เป็น slicer dimension ได้ โดย single tuple แสดงได้ดังนี้

WHERE ([Time]. [1st half], [Route]. [nonground])

เมื่อเรากำหนด slicer dimension เป็นเซ็ตของ tuple , MDX ก็จะพยายามหาค่าของเซตโดยการรวมค่าเซลล์ผลลัพธ์ในแต่ละ tuple เข้าด้วยกันซึ่งทำได้โดยใช้ฟังก์ชัน Aggregate ดังนั้น where clause ข้างล่างนี้ก็จะสามารถให้ผลลัพธ์ได้อย่างถูกต้อง

WHERE { ([Time].[1st half], [Route].[nonground]), ([Time].[1st half], [Route].[ground]) }

- **การระบุคิวบ์**

เราสามารถระบุคิวบ์ที่เราต้องการคิวรีได้โดยใช้ FROM clause ตัวอย่างจะเป็นดังนี้

FROM SalesCube

โดยที่การระบุชื่อคิวบ์เราสามารถใส่ได้เพียงชื่อเดียวเท่านั้น แต่ก็ไม่ได้หมายความว่าจะเป็นข้อจำกัดในการใช้ข้อมูล เนื่องจากว่าเราสามารถใช้ฟังก์ชัน LookupCube จัดการได้เมื่อเราต้องการข้อมูลจาก คิวบ์อื่นด้วย ดังนั้นตัวอย่างการระบุคิวบ์ต่อไปนี้จึงเป็นตัวอย่างที่ผิดเนื่องจาก MDX ไม่ได้อนุญาตให้มีการ join กันดังเช่นใน SQL

FROM SalesCube, OtherCube

5.4.10 CrossJoin

Crossjoin เป็นฟังก์ชันที่มีประโยชน์อย่างมาก ช่วยให้สามารถจัดการแสดงผลออกมาตามที่เราต้องการได้ ซึ่งรูปแบบของการใช้ฟังก์ชันนี้จะเป็นดังนี้

Crossjoin(«Set1», «Set2»)

หรือเราสามารถเขียนอีกแบบได้ดังนี้

«Set1» * «Set2»

ลำดับของ tuple ในเซตของผลลัพธ์จะขึ้นกับลำดับของ <<Set1>> และ <<Set2>> รวมทั้งลำดับของ member ภายในเซตนั้นดังนั้น ถ้า «Set1» = {*x1*, *x2*,...,*xn*} and «Set2» = {*y1*, *y2*, ..., *yn*} แล้ว ผลลัพธ์ของ **Crossjoin**(*Set1*, *Set2*) จะเป็นดังนี้

{(*x1*, *y1*), (*x1*, *y2*),...,(*x1*, *yn*), (*x2*, *y1*), (*x2*, *y2*),..., (*x2*, *yn*),..., (*xn*, *y1*), (*xn*, *y2*),..., (*xn*, *yn*)}

ตัวอย่างการใช้ Crossjoin เป็นดังนี้

```
SELECT
   Crossjoin(  [Product].[Product Family].members ,[Store].[Store City].members) on rows,
        {[Measures].[MeasuresLevel].members} on columns
FROM sales
```

โดยจะได้ผลลัพธ์ออกมาเป็นดังรูป

		Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Drink	Alameda					
	Beverly Hills	1,945.00	1,547.71	$3,940.54	618	2,392.83
	Los Angeles	2,422.00	1,953.55	$4,823.88	767	2,870.33
	San Diego	2,560.00	2,014.67	$5,065.10	797	3,050.43
	San Francisco	175.00	146.34	$373.72	109	227.38
	Portland	2,371.00	1,880.59	$4,743.04	751	2,862.45
	Salem	3,735.00	2,955.76	$7,394.25	1202	4,438.49
	Bellingham	208.00	167.10	$413.08	130	245.98
	Bremerton	2,288.00	1,860.05	$4,621.14	730	2,761.09
	Seattle	2,213.00	1,746.74	$4,370.65	709	2,623.91
	Spokane	2,238.00	1,709.79	$4,293.30	694	2,583.51
	Tacoma	3,092.00	2,412.35	$6,092.91	904	3,680.56
	Walla Walla	191.00	143.28	$355.81	121	212.53
	Yakima	1,159.00	939.29	$2,348.79	366	1,409.50
Food	Vancouver					
	Victoria					
	Mexico City					

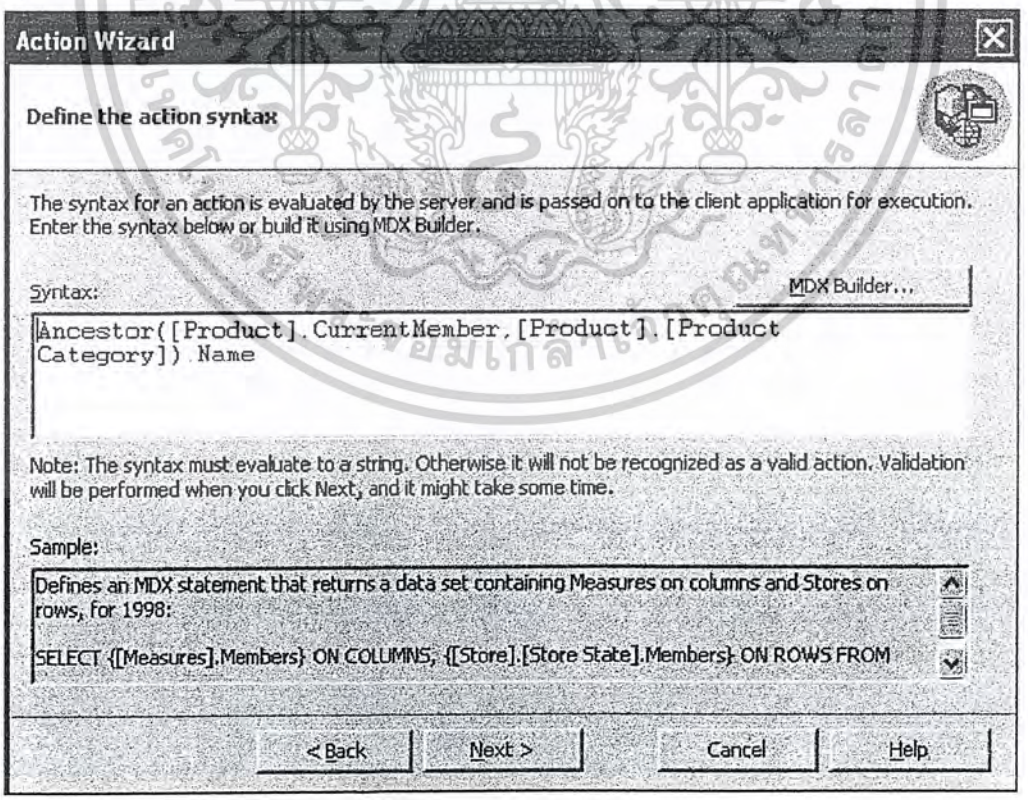
รูปที่ 5-6 แสดงผลลัพธ์ของคำสั่ง Crossjoin

5.5 Using MDX with Analysis Services

MDX สามารถใช้สร้าง items ใน Analysis Services ดังต่อไปนี้

- Action

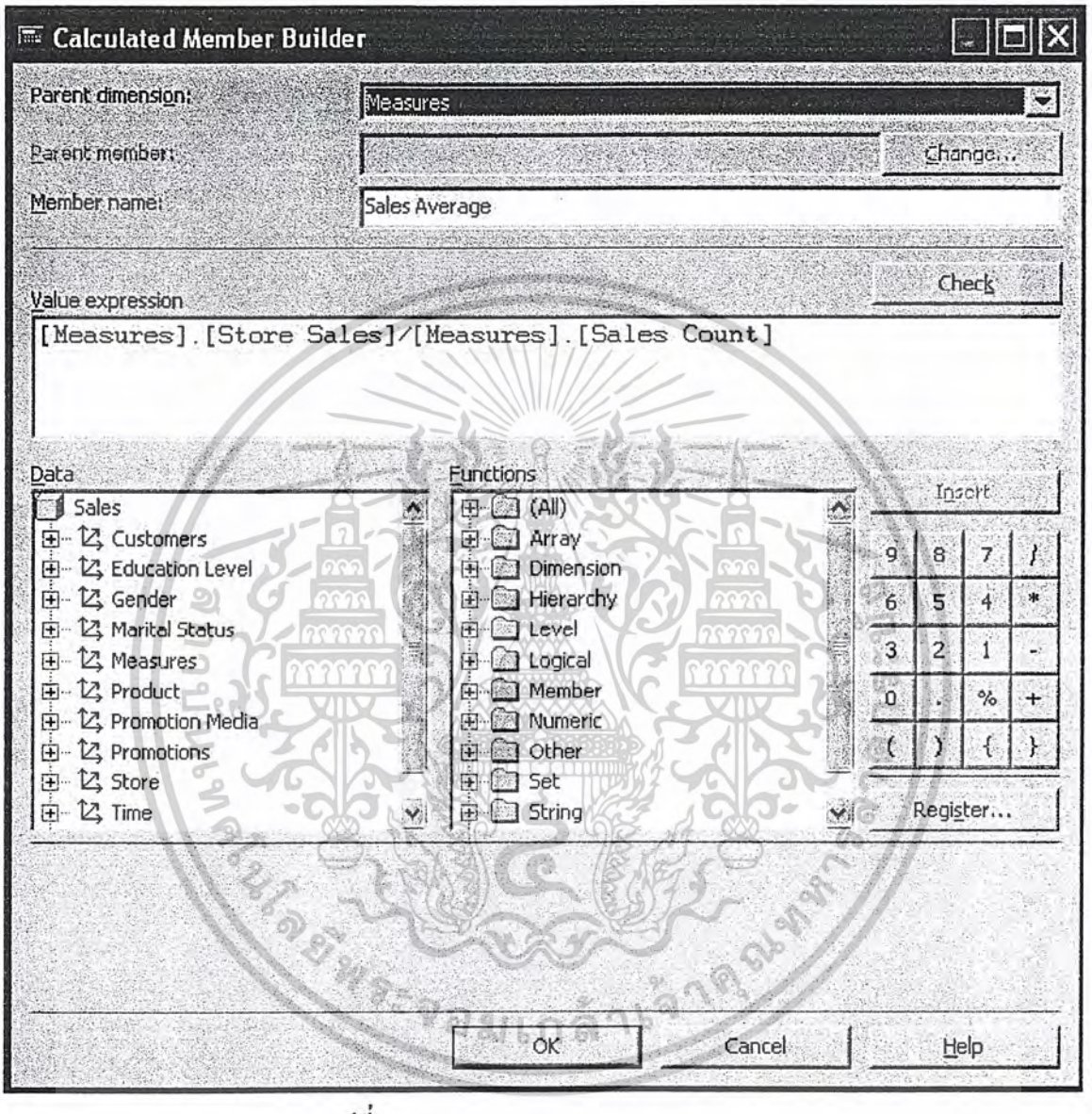
Action คือ สิ่งที่ user สามารถกำหนดการกระทำที่เกิดกับ cube ได้ เช่น เมื่อ user พบว่า current stock ของ สินค้าชนิดหนึ่ง มีปริมาณที่น้อยเกินไป ก็จะสั่งให้กระทำ Action new order ไปยัง order entry system



รูปที่ 5-7 แสดง Action Wizard

- Calculated member

Calculated member เป็น member ที่ไม่ใช่ตัวข้อมูลที่เก็บอยู่ แต่เป็นสมการการคำนวณโดยใช้ฟังก์ชันต่างๆ ของ MDX



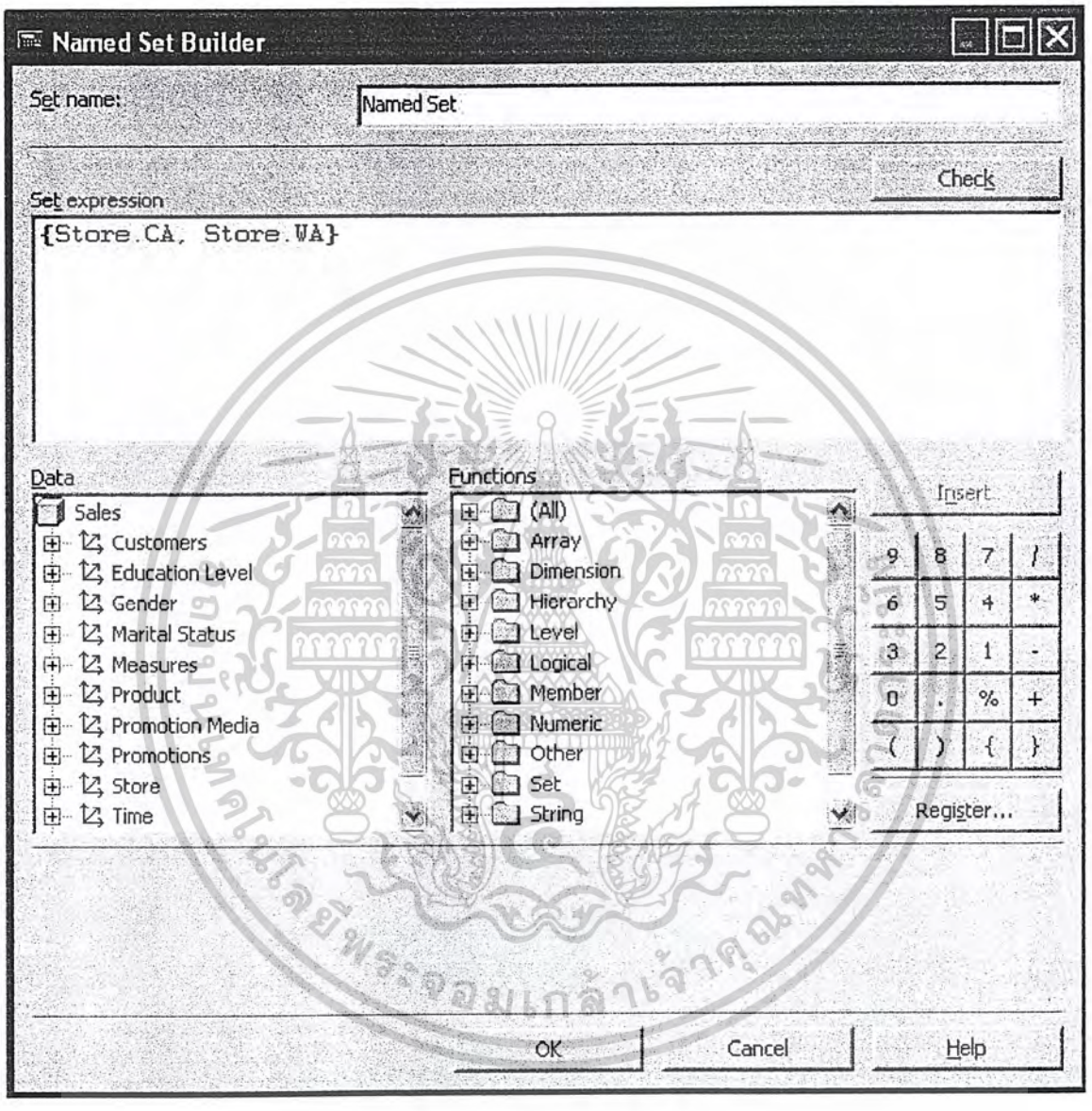
รูปที่ 5-8 แสดง Calculated member Builder

- Calculated cells

  Calculated cells เป็น cellsที่จะเกิดค่าขึ้นในขณะ run timeโดยใช้ MDX กำหนดรายละเอียดต่าง ๆ

- Named set

  Named Set คือ set ของ members หรือ set ของ expression ซึ่งต้องใช้ MDX ในการสร้างขึ้น



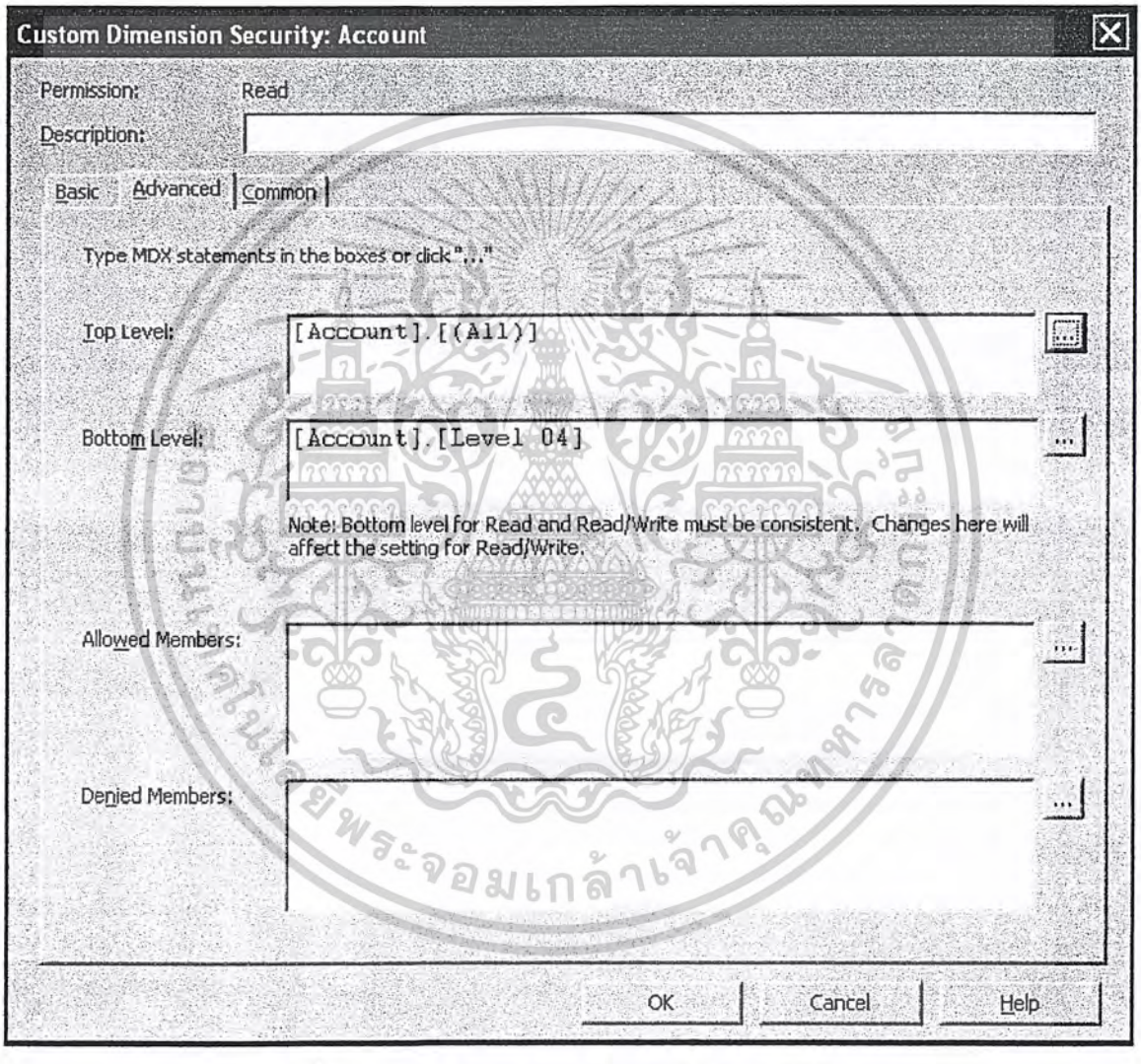
รูปที่ 5-9 แสดง Named set Builder

- Custom rollup formula & Custom member formula

เป็นการกำหนด cell values ใน cube ว่าจะใช้ค่า member ตัวใดบ้าง โดย custom rollup formula จะมีผลกับทุก ๆ members รวมทั้ง calculated member ด้วย ส่วน custom member formula จะไม่มีผลกับ calculated member

- Custom rule in dimension security & Custom rule in cell security

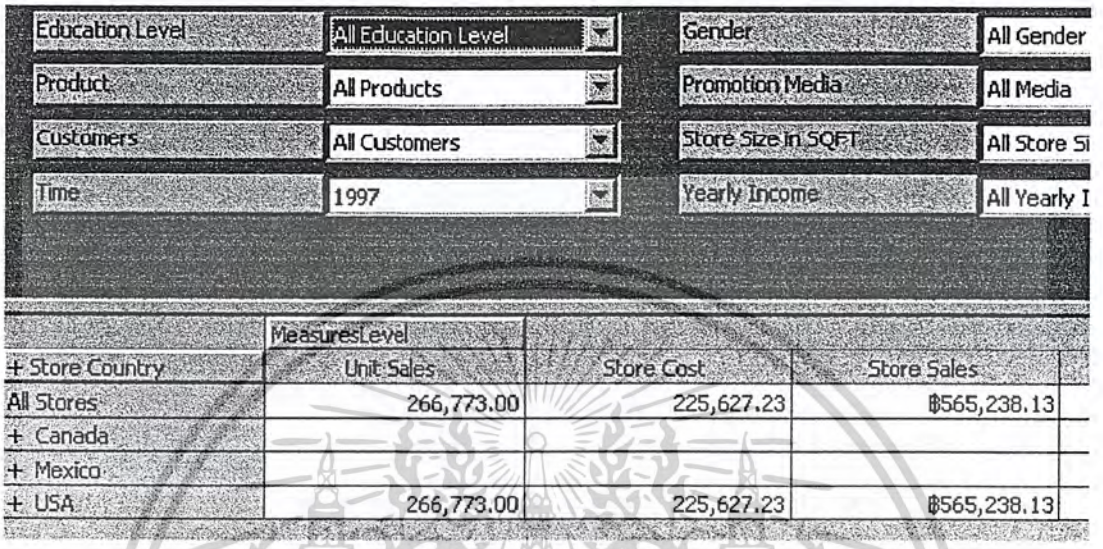
ในแต่ละ Dimension และแต่ละ Cell ภายใน Cube สามารถกำหนด rule ในการใช้งานได้ โดยสามารถนำ MDX เข้ามากำหนด rule ได้



รูปที่ 5-10 แสดง Custom rule in dimension security

5.6 สิ่งที่ Analysis ทำไม่สามารถทำได้แต่ MDX สามารถทำได้

- MDX สามารถกำหนด Axis Dimensions และ Slicer Dimensions ที่เฉพาะเจาะจงกว่า Analysis Service โดยการใช้ Function ต่าง ๆ ตัวอย่างเช่น ต้องการดูข้อมูลจำนวนสินค้าใน Store ของประเทศ Canada และ USA ผลที่ได้จากการใช้ Analysis Service จะได้ดังรูปที่ 5-11



Education Level	All Education Level	Gender	All Gender
Product	All Products	Promotion Media	All Media
Customers	All Customers	Store Size in SQFT	All Store Si
Time	1997	Yearly Income	All Yearly I

	MeasuresLevel		
+ Store Country	Unit Sales	Store Cost	Store Sales
All Stores	266,773.00	225,627.23	฿565,238.13
+ Canada			
+ Mexico			
+ USA	266,773.00	225,627.23	฿565,238.13

รูปที่ 5-11 แสดงเมื่อใช้ Analysis Service ดูข้อมูลของ Store

จะเห็นได้ว่าจะปรากฏข้อมูลของ Store ใน Mexico ที่ไม่ต้องการดูด้วยแต่ถ้าใช้ MDX จะได้คำตอบดังรูปที่ 5-12

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Canada					
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90

รูปที่ 5-12 แสดงเมื่อใช้ MDX ดูข้อมูลของ Store

- การใช้ Visual Basic for Application Function และ Excel Function (ในการกำหนด Axis Dimensions) ตัวอย่างเช่น ต้องการดูข้อมูลของ Unit Sales เฉพาะ Product ที่มีชื่อ fruit ที่ได้จากการใช้ Analysis Service จะได้ดังรูปที่ 5-13

	MeasuresLevel
Product Name	Unit Sales
	266,773.00
	24,597.00
	6,838.00
	6,838.00
	1,683.00
Good Total	269.00
Good Imported Beer	154.00
Good Light Beer	115.00
Pearl Total	385.00
Pearl Imported Beer	175.00
Pearl Light Beer	210.00
Portsmouth Total	362.00
Portsmouth Imported Beer	187.00
Portsmouth Light Beer	175.00
Top Measure Total	306.00
Top Measure Imported Bee	145.00
Top Measure Light Beer	161.00
Walrus Total	361.00

รูปที่ 5-13 แสดงเมื่อใช้ Analysis Service ดูข้อมูลของ Product

จะเห็นได้ว่าเราไม่สามารถแยกดูเฉพาะข้อมูลที่เราต้องการได้ แต่หากเป็น MDX จะสามารถกระทำได้ (โดยใช้ Visual Basic for Application Function) ดังรูปที่ 5-14

	Unit Sales
Applause Canned Mixed Fruit	205.00
Big City Canned Mixed Fruit	204.00
Green Ribbon Canned Mixed Fruit	142.00
Swell Canned Mixed Fruit	204.00
Toucan Canned Mixed Fruit	187.00
Best Choice Apple Fruit Roll	174.00
Best Choice Grape Fruit Roll	114.00
Best Choice Raspberry Fruit Roll	110.00
Best Choice Strawberry Fruit Roll	150.00
Fast Apple Fruit Roll	149.00
Fast Grape Fruit Roll	173.00
Fast Raspberry Fruit Roll	163.00
Fast Strawberry Fruit Roll	154.00
Fort West Apple Fruit Roll	181.00
Fort West Grape Fruit Roll	178.00
Fort West Raspberry Fruit Roll	210.00
Fort West Strawberry Fruit Roll	189.00
Horatio Apple Fruit Roll	177.00
Horatio Grape Fruit Roll	191.00
Horatio Raspberry Fruit Roll	149.00
Horatio Strawberry Fruit Roll	169.00
Nationeel Apple Fruit Roll	185.00
Nationeel Grape Fruit Roll	216.00
Nationeel Raspberry Fruit Roll	167.00
Nationeel Strawberry Fruit Roll	138.00

รูปที่ 5-14 แสดงเมื่อใช้ MDX ดูข้อมูลของ Product

- Analysis จะใช้ Level ใด Level หนึ่งใน Dimension หนึ่งที่เหลือจากการเป็น Axis Dimensions เพื่อเป็น Slice Dimension แต่ใน MDX สามารถเลือกได้มากกว่า 1 Level (โดยการใช้ Nameset ช่วยในการกำหนด where)

```
WITH MEMBER [Product].[Food OR Drink] AS
                    '([Product].[Food], Measures.[Unit Sales]) +
                    ([Product].[Drink],Measures.[Unit Sales])'
SELECT  {Measures.[Unit Sales]}  ON COLUMNS,
        DESCENDANTS(Time.[1997], [Quarter], SELF_AND_BEFORE) ON ROWS
FROM Sales
WHERE [Product].[Food OR Drink]
```

จากตัวอย่าง code ข้างต้นจะเห็นเราสามารถดูข้อมูลของ Product ที่เป็นทั้ง Food และ Drink พร้อม ๆ กันได้เลย

	Unit Sales
1997	216,537.00
Q1	53,785.00
Q2	50,720.00
Q3	53,505.00
Q4	58,527.00

รูปที่ 5-15 แสดงข้อมูลที่ได้จาก code MDX(ดูข้อมูลFood และ Drink พร้อมกัน)

- ใน MDX สามารถใช้ Nameset ช่วยในการกำหนด Axis Dimensions ได้ แต่ใน Analysis จะใช้ Level ของ Dimensions ในการกำหนด

```
WITH MEMBER [Time].[First Half 97] AS  '[Time].[1997].[Q1] + [Time].[1997].[Q2]'
        MEMBER [Time].[Second Half 97] AS '[Time].[1997].[Q3] + [Time].[1997].[Q4]'
SELECT {[Time].[First Half 97], [Time].[Second Half 97]} ON COLUMNS,
    NON EMPTY {[Store].[Store Name].MEMBERS} ON ROWS
FROM [Sales]
```

จากตัวอย่าง code ข้างต้นจะเห็นได้ว่าเราสามารถกำหนด Axis Dimensions ที่มีความหลากหลายได้จากการใช้ Nameset

	First Half 97	Second Half 97
Store 6	9,659.00	11,674.00
Store 7	11,927.00	13,736.00
Store 24	12,381.00	13,254.00
Store 14	975.00	1,142.00
Store 11	13,561.00	12,518.00
Store 13	20,805.00	20,775.00
Store 2	1,028.00	1,209.00
Store 3	11,906.00	12,670.00
Store 15	12,020.00	12,991.00
Store 16	11,098.00	12,493.00
Store 17	16,543.00	18,714.00
Store 22	1,042.00	1,161.00
Store 23	5,956.00	5,535.00

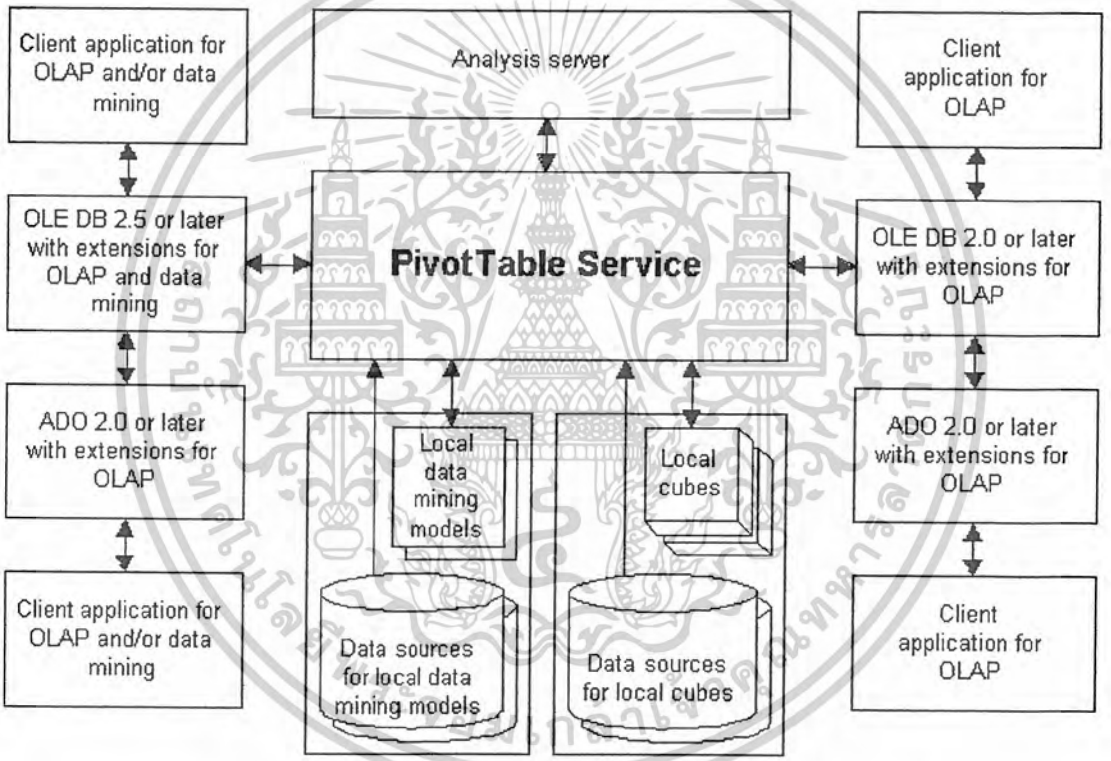
รูปที่ 5-16 แสดงข้อมูลที่ได้จาก code MDX

บทที่ 6

Microsoft OLAP Client Architecture

6.1 สถาปัตยกรรมไคลเอ็นต์ภายใน Analysis Service

หัวใจของสถาปัตยกรรมไคลเอ็นต์ คือ PivotTable Service โดยจะทำหน้าที่เป็นตัวติดต่อกับ Analysis server และจะสร้างอินเตอร์เฟสซึ่งจะเรียกใช้โดยโปรแกรมประยุกต์ไคลเอ็นต์ ในการเข้าถึงข้อมูล OLAP โปรแกรมประยุกต์ไคลเอ็นต์สามารถเรียกใช้ PivotTable Service ได้โดยเรียกผ่านอินเตอร์เฟสของ OLE DB เมื่อใช้โปรแกรมภาษา C++ หรือใช้ Microsoft ActiveX Data Object (ADO) เมื่อใช้โปรแกรมพวก Visual Basic ดังรูปที่ 6-1



รูปที่ 6-1 แสดงโครงสร้างของไคลเอ็นต์ใน Analysis Service

6.2 PivotTable Service

PivotTable Service คือ OLE DB Provider สำหรับข้อมูลมัลติไดเมนชันแนล นั่นคือ PivotTable Service จะสร้างฟังก์ชันของ OLE DB ให้แอพพลิเคชันเรียกใช้ในการเข้าถึงข้อมูลมัลติไดเมนชันแนลและนอกจากนี้ PivotTable Service ยังสามารถดึงข้อมูลที่เป็นตารางหรือข้อมูลที่เป็นมัลติไดเมนชันแนลเพื่อให้รองรับกับการใช้ SQL หรือ MDX (MultiDimensional eXpression)

PivotTable Sevice สามารถสร้างไฟล์โลคอลคิวบ์ (local cube) ที่เก็บข้อมูลจากคิวบ์บนเครื่องเซิร์ฟเวอร์หรือข้อมูลจากรีเลชันแนลดาต้าเบส OLEDB, โลคอลคิวบ์สามารถเก็บข้อมูลที่เป็นแฟ้มข้อมูล

คิวบ์แบบมัลติไดเมนชันแนลบนเครื่องไคลเอ็นต์และยังสามารถใช้การวิเคราะห์แบบออฟไลน์ (Off-line) ได้ นั่นคือถ้าโลคอลมีการเก็บข้อมูลแบบ Multidimensional OLAP (MOLAP) ก็จะสามารถทำได้โดยไม่ต้องทำการเชื่อมต่อไปยัง Analysis Server

PivotTable Service ถูกออกแบบสำหรับการวิเคราะห์ข้อมูลทั้งแบบออนไลน์และออฟไลน์ เมื่อมีการร้องขอมาจากเครื่องไคลเอ็นต์ทาง Analysis Server ก็จะทำการประมวลผลการร้องขอเหล่านั้นและสร้างเซตข้อมูลจากส่วนของข้อมูลที่มีการรวมค่าไว้ก่อนหรือจากการประมวลผลการรวมค่าใหม่ เมื่อทำเสร็จ Analysis Server ก็จะส่งเซตข้อมูลกลับไปให้ยังเครื่องไคลเอ็นต์ที่ร้องขอมา

เราสามารถพัฒนาโปรแกรมประยุกต์ไคล์เอ็นต์ซึ่งเรียกใช้ PivotTable Service ได้หลายวิธี นั่นคือสามารถใช้ Microsoft ActiveX Data Objects Multidimensional (ADO MD) สำหรับการพัฒนาโปรแกรมด้วยภาษาที่มีลักษณะเป็น Component Object Model (COM) Automation เช่น Visual Basic หรือ ASP หรืออีกวิธีก็คือการใช้ OLE DB สำหรับการพัฒนาโปรแกรมด้วยภาษา C++

6.2.1 ลักษณะเด่นของ PivotTable Service

- ความสามารถในการปรับปรุงประสิทธิภาพของโปรแกรมประยุกต์ไคลเอ็นต์ในการเข้าถึงข้อมูลของ OLAP
- PivotTable Service จะถูกรวมเข้าไปใน Microsoft Excel 2000 ซึ่งจะทำให้ Excel กลายเป็นทูลส์ที่มีประสิทธิภาพในการนำเสนอข้อมูล OLAP
- สามารถสร้างคิวบ์ของฝั่งไคลเอ็นต์ได้ ซึ่งยอมให้นำไปวิเคราะห์บนเครื่อง คอมพิวเตอร์เครื่องใดก็ได้ ในขณะที่ไม่ได้เชื่อมต่อกับระบบเครือข่ายและแหล่งข้อมูล (Data Source) หลัก, คิวบ์ของฝั่งไคลเอ็นต์มักจะเก็บไว้ในไฟล์ คิวบ์บนเครื่องไคลเอ็นต์ไฟล์เหล่านี้สามารถเปิดและอ่านได้โดย OLAP
- เนื่องจาก PivotTable Service ถูกติดตั้งบนเครื่องไคลเอ็นต์ เช่น เมื่อทำการติดตั้ง Microsoft Excel ผู้พัฒนาโปรแกรมก็สามารถสร้างแอพพลิเคชัน โดยการเขียนโปรแกรมด้วยภาษาใด ๆ ก็ตามที่ถนัด เพื่อให้เข้าใช้ข้อมูล OLAP หรือแหล่งข้อมูลรีเลชันแนล โดยใช้เทคโนโลยีของ OLE DB
- PivotTable Service ทำงานกับ Analysis Service เหมือนกับเป็นส่วนหนึ่งของสถาปัตยกรรมของเซอร์วิสเหล่านี้ดังนั้น PivotTable Service จึงยอมให้หลาย ๆ ไคลเอ็นต์สามารถเข้าใช้คิวบ์อันเดียวกันได้ โดยแบ่งการประมวลผลตามผู้ใช้
- สนับสนุน Multidimensional Expressions (MDX) ซึ่งเป็นภาษาที่ใช้ในการคิวรีและอนุญาตให้ทำการเข้าใช้คิวบ์ที่เก็บข้อมูลไดเมนชันและส่วนอื่น ๆ ได้โดยใช้ MDX Statement
- PivotTable Service อนุญาตให้ทำการดาวน์โหลดข้อมูลจากแหล่งข้อมูลและเก็บข้อมูลในโครงสร้างแบบมัลติไดเมนชันแนลบนโลคอลคอมพิวเตอร์เพื่อใช้สำหรับการวิเคราะห์แบบออฟไลน์

6.2.2 คอมโพเนนท์ที่ใช้ใน PivotTable Service

ใน PivotTable Service ได้มีการรวม Dynamic-Link Libraries (DLLs) เอาไว้ดังนั้นที่เครื่องไคลเอ็นต์ก็จะต้องมีไฟล์เหล่านี้ซึ่งจะต้องมีไฟล์ใดบ้างก็ขึ้นอยู่กับแอพพลิเคชันที่ทำต้องการใช้คุณสมบัติใดของ PivotTable Service

File set	Component files
1	Msolap80.dll, Msolui80.dll, Msolap80.rll, Olapuir.rll, and Microsoft® Data Access Components (MDAC)
2	File Set 1 plus Msmdcb80.dll, Msmdgd80.dll, and an appropriate OLE DB tabular data provider
3	File Set 1 plus Msdmine.dll, Msmdun80.dll, Msdmine.rll, and Msdmeng.dll

ตารางที่ 6-1 แสดง file set ของ PivotTable Service

Task	File set
Communicate with the Analysis server using TCP/IP or HTTP and read local cube files	1
Create and refresh local cubes	2
Read OLAP and relational data mining models	3

ตารางที่ 6-2 แสดงไฟล์ที่จำเป็นสำหรับแต่ละงาน

6.2.3 การติดตั้งและลงทะเบียนคอมโพเนนท์

เราสามารถติดตั้งและลงทะเบียนคอมโพเนนท์ได้อย่างสะดวกโดยใช้โปรแกรมติดตั้งที่มีมาให้ในแผ่น CD ของ SQL Server 2000 โดยจะอยู่ใน ไดเรคทอรี \Msolap\Install\Pts ซึ่งจะมีอยู่ 2 โปรแกรม คือ Ptslite.exe และ Ptsfull.exe โดย Ptslite.exe จะติดตั้งเฉพาะไฟล์ของ PivotTable Service แต่ถ้าเลือก Ptsfull.exe ก็จะติดตั้ง Microsoft Data Access Component (MDAC) ให้ด้วย

Ptslite.exe เมื่อเรารัน Ptslite.exe โปรแกรมก็จะติดตั้งไฟล์เหล่านี้ให้

atl.dll	msdmeng.dll	msdmine.dll
msmdcb80.dll	mdmdgd80.dll	msolap80.dll
msolui80.dll	msmdcube.dll	msmdgdrv.dll
msolap.dll	msolapui.dll	msdmine.rll
msolap80.rll	olapuir.rll	Msvbvm60.dll
msmdun80.dll	msolapr.dll	

ตารางที่ 6-3 แสดงไฟล์ที่จะติดตั้งโดย Ptslite.exe

ถ้า Ptsfull.exe โปรแกรมจะติดตั้งไฟล์ที่ได้แสดงไว้ข้างต้น และจะติดตั้ง MDAC เพิ่มให้ด้วย

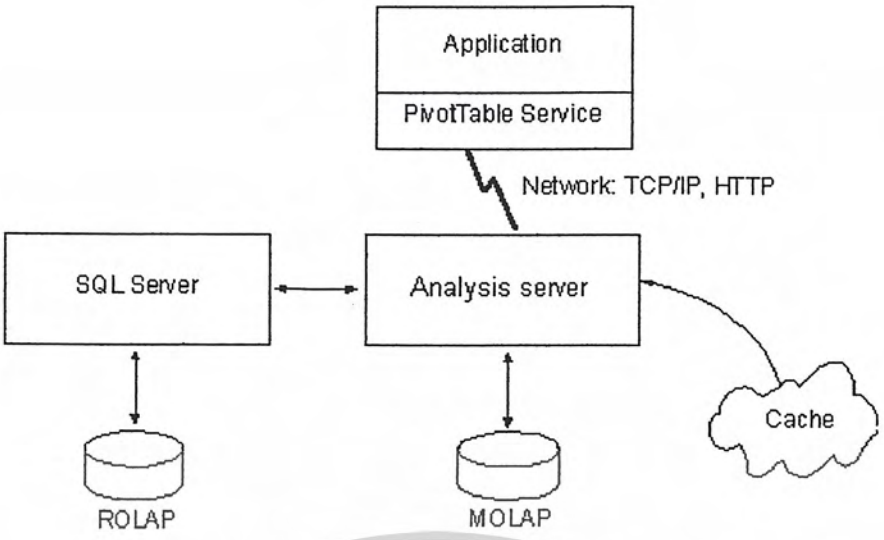
6.2.4 ลักษณะการเชื่อมต่อของ PivotTable Service

เราสามารถใช้ PivotTable Service เชื่อมต่อได้ 3 รูปแบบคือ

- การเชื่อมต่อกับ Microsoft SQL Server2000 Analysis Services
- เชื่อมต่อกับ OLE DB Provider
- เชื่อมต่อกับโลคอลคิวบ์

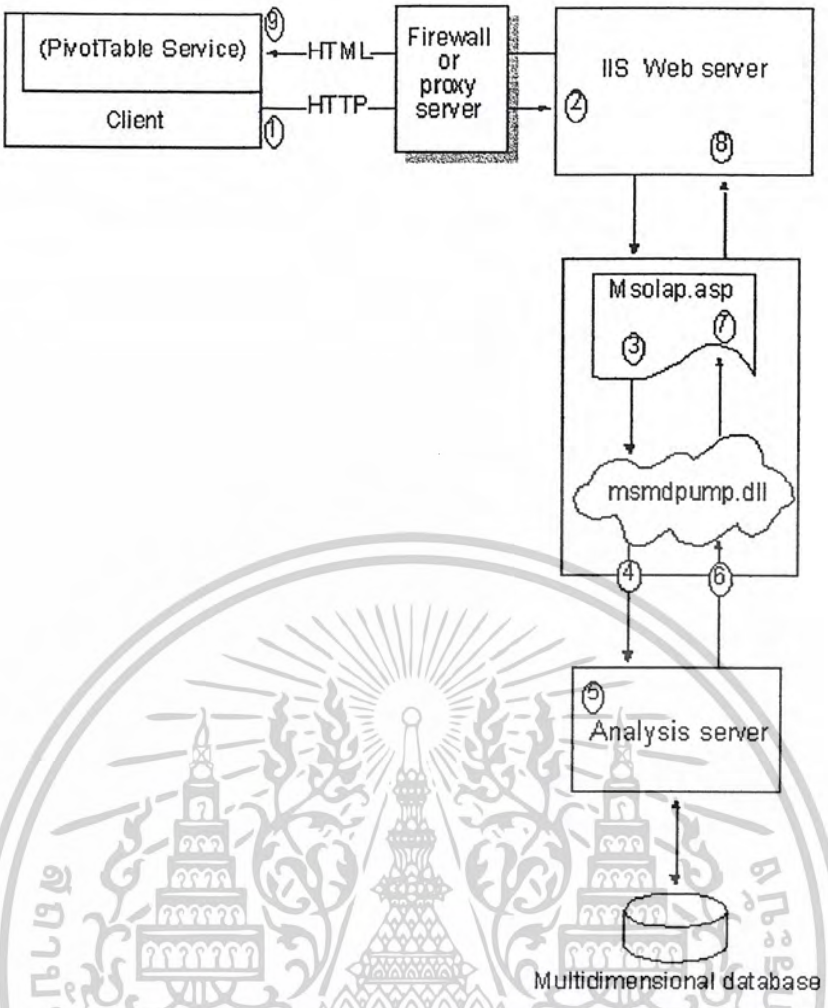
6.2.4.1 การเชื่อมต่อกับ SQL Server2000 Analysis Service

เราสามารถเชื่อมต่อกับ Analysis Server ได้หลายวิธี เช่น ใช้โพรโตคอล HTTP ในการเชื่อมต่อผ่านอินเตอร์เน็ต เมื่อเราใช้ PivotTable Service ในการแสดงข้อมูลจาก Analysis Service , PivotTable Service จะทำการติดต่อกับ Remote Analysis Server ผ่านทางเน็ตเวิร์คโดยใช้โพรโตคอล TCP/IP หรือ HTTP ดังรูป



รูปที่ 6-2 แสดงการเชื่อมต่อกับ Analysis server

การเชื่อมต่อโดยใช้โพรโตคอล HTTP ทำให้ผู้ใช้งานทั่วไปสามารถติดต่อกับ Analysis server โดยผ่าน Microsoft Internet Information Service (IIS) ได้ โดยการกำหนดคุณสมบัติของแหล่งข้อมูลใน connection string ให้เป็น HTTP หรือ HTTPS URL (ซึ่งจะมีระดับความปลอดภัยที่สูงกว่า) ลักษณะการเชื่อมต่อจะเป็นดังรูป



รูปที่ 6-3 แสดงการเชื่อมต่อโดยใช้ HTTP

ตัวอย่างของโปรแกรมที่ทำการติดต่อในรูปแบบนี้ แสดงได้ดัง

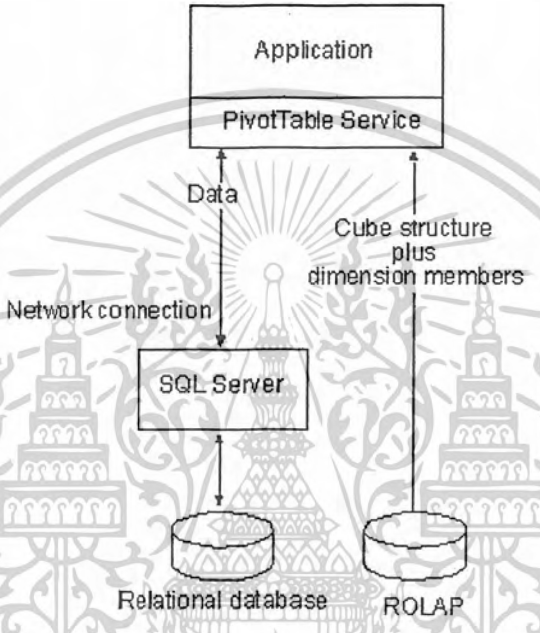
```
Dim cat as new ADOMD.Catalog
cat.ActiveConnection = "Provider = msolap;" & _
    " Datasource =" _ &
    " http://<URL>/;" & _
    " Initial Catalog = FoodMart 2000"
```

ถ้าต้องการให้มีการใช้ SSL (Secure Socket Layer) ด้วยก็สามารถทำได้ดังนี้

```
Dim cat as new ADOMD.Catalog
cat.ActiveConnection = "Provider = msolap; Datasource =" & _
  " https://<URL>/;" & _
  " Initial Catalog = FoodMart 2000"
```

6.2.4.2 การเชื่อมต่อกับ OLE DB Provider

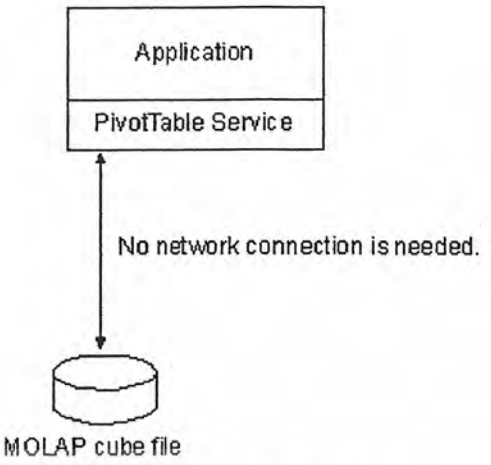
เมื่อต้องการทำงานกับโลคอลคิวบ์ที่เป็น Relational OLAP (ROLAP) เราต้องทำการเชื่อมต่อกับ Relational data provider โดยโลคอลคิวบ์ในลักษณะนี้จะเก็บนิยามโครงสร้างของคิวบ์แต่จะไม่ได้เก็บข้อมูลจริงๆ หรือข้อมูลที่ทำการคำนวณไว้ล่วงหน้า ดังนั้นเมื่อเราต้องการดึงข้อมูลจึงจำเป็นต้องใช้การติดต่อกับ tabular data provider โดยขบวนการเหล่านี้ไคลเอ็นต์แอพพลิชันไม่จำเป็นต้องรับรู้ ผลจากการทำงานในลักษณะนี้ทำให้เราจะได้คิวบ์ที่มีขนาดเล็กกว่าคิวบ์ซึ่งสร้างโดย MOLAP แต่ก็การทำงานก็จะมีประสิทธิภาพน้อยกว่า local cube แบบ MOLAP ด้วยเช่นกัน ลักษณะการเชื่อมต่อจะเป็นดังรูป



รูปที่ 6-4 แสดงการเชื่อมต่อกับ OLE DB Provider

6.2.4.3 การเชื่อมต่อกับ โลคอลคิวบ์ไฟล์

ในการแสดงข้อมูลจากคิวบ์ MOLAP, PivotTable Service จะเชื่อมต่อกับโลคอลคิวบ์ในแบบเดียวกับที่เชื่อมต่อกับแหล่งข้อมูลอื่น ๆ โดย PivotTable Service จะประมวลผลจากการคิวรี แล้วส่งค่าที่ได้คืนกลับไปยังโปรแกรมประยุกต์ ซึ่งโปรแกรมประยุกต์ทางฝั่งไคลเอ็นต์สามารถเข้าถึงไดเมนชัน, เลเวล (level), พร็อพเพอร์ตี้ ได้โดยที่ไม่ต้องทำการเชื่อมต่อกับรีโมทเซิร์ฟเวอร์ (Remote Server)



รูปที่ 6-5 แสดงการเชื่อมต่อกับ ไฟล์โลคอลคิวบ์

6.2.5 การทำงานของไคลเอ็นต์ใน PivotTable Service

- การเชื่อมต่อกับแหล่งข้อมูล

วิธีการเบื้องต้นในการเชื่อมต่อแหล่งข้อมูลคือการใช้ออปเจ็กต์ (Object) **Connection** หรือใช้คุณสมบัติ **ActiveConnection** ของออปเจ็กต์ **Catalog** โดยเราสามารถกำหนดพารามิเตอร์ของการเชื่อมต่อได้โดยกำหนดใน **Connection String**

ADO Connection Object

เมธอด (Method) **Open** ของออปเจ็กต์ **Connection** ได้รวมเอาพารามิเตอร์สำหรับใช้ในการเชื่อมต่อเข้าไปใน **connectionstring Property** เมื่อเมธอดนี้ถูกสั่งให้ทำงานก็จะทำการเชื่อมต่อกับแหล่งข้อมูลที่กำหนดในเมธอด Open : connect. Open connectionstring, UserId, Password , OpenOptions

ในการเชื่อมต่อกับ SQL Server 2000 Analysis Service นั้น **Datasource** ต้องตั้งชื่อหรือ IP address ของ Analysis server ที่ต้องการจะเชื่อมต่อ ส่วน **Provider** ต้องกำหนดเป็น "MSOLAP" ส่วน **Initial Catalog** ให้ระบุฐานข้อมูลบนเซิร์ฟเวอร์ที่ต้องการจะติดต่อ

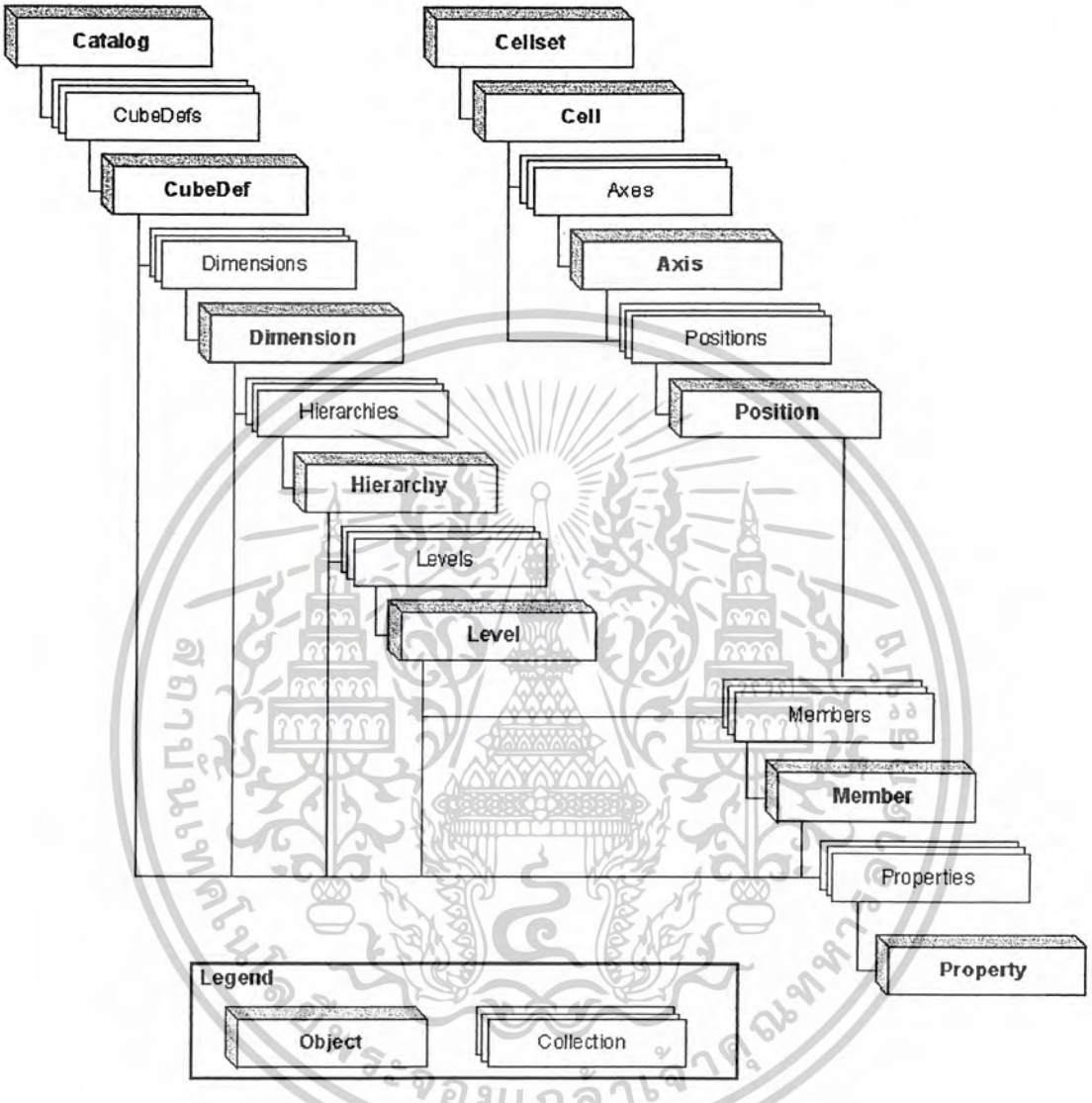
- การแสดงโครงสร้างของข้อมูล (Retrieving Schema Information)

สามารถใช้ Microsoft ActiveX Data Objects (Multidimensional)(ADO MD), ADO หรือ OLE ในการแสดงโครงสร้างของสกีมา (schema rowsets) โดยใช้ PivotTable Service

ในการแสดงโครงสร้างของข้อมูลของคิวบ์จะใช้ออปเจ็กต์ **Cubedef** ใน ADO MD หรือใช้เมธอด **OpenSchema** ใน ADO , ออปเจกต์ **CubeDef** จะมีโครงสร้างคิวบ์เป็นแบบลำดับขั้น (hierarchy) จะมีบางข้อมูลเกี่ยวกับคิวบ์ที่ไม่ได้เก็บอยู่ในออปเจกต์ **CubeDef** เช่นการกำหนดแอคชัน (actions) และการจัดรูปแบบเซลล์ (cell formulas) ซึ่งจะต้องใช้เมธอด **OpenSchema** ในการแสดงข้อมูลเหล่านี้

ในการแสดงโครงสร้างของสกีมาจะใช้ ADO หรือ OLE DB สำหรับใน ADO จะใช้เมธอด **OpenSchema** ของออปเจ็กต์ **Connection** ส่วนใน OLE DB จะใช้อินเทอร์เฟส IDBchemaRowset COM

ออปเจ็กต์ **CubeDef** จะเก็บข้อมูลไดเมนชันของโลคอลคิวบ์โดยใช้คอลเล็คชัน (collection) **Dimensions** ของตัวมันเอง ซึ่ง **Dimensions** มีรูปแบบเป็นคอลเล็คชันของ **Hierarchies** แสดงดังรูปที่ 6-6



รูปที่ 6-6 แสดงโครงสร้างลำดับขั้น

- การแสดงข้อมูล (Retrieving Data)

เราสามารถแสดงข้อมูลทำได้โดยใช้ออปเจ็กต์ **Cellset** ของ ADOMD หรือใช้ **Recordset** ของ ADODB เราสามารถใช้ ADOMD ในการแสดงผลลัพธ์ของการคิวรีด้วย MDX จากโลคอลคิวบ์โดยใช้ออปเจ็กต์ **Cellset** ได้ ส่วนการแสดงผลลัพธ์ที่มีลักษณะเป็นตารางจะใช้ ออปเจ็กต์ **Command** และ **Recordset**

<u>ตัวอย่างการใช้ **Cellset**</u>

จะใช้ออปเจ็กต์ **Connection** ในการเชื่อมต่อกับ Analysis Server, ค่าของพร็อพเพอร์ตี้ **Source** ของ **Cellset** จะอยู่ในรูปแบบของ MDX query, พร็อพเพอร์ตี้ **ActiveConnection** ของ **Cellset** จะถูก

กำหนดเหมือนกับพร็อพเพอร์ตี้ **ActiveConnection** ของออปเจ็กต์ **Connection** และมีการใช้เมธอด Open ในการแสดงผลลัพธ์ที่ได้

ออปเจ็กต์ **Cellset** จะเก็บคอลเล็คชันที่เรียกว่า **Axes** ซึ่งจะใช้อธิบายในแต่ละแกน จะมีออปเจ็กต์ **Axis** 1 ตัวในคอลเล็คชันนี้สำหรับแต่ละไดเมนชันที่ต้องการ แต่ละออปเจ็กต์ **Axis** จะเก็บคอลเล็คชันของ **Positions** ซึ่งเก็บข้อมูลในแต่ละแถว (row), คอลัมน์ (column), หน้า (page) และเซตของผลลัพธ์

```
Dim conn As New ADODB.Connection
Dim cst As New ADOMD.Cellset
Dim axs As ADOMD.Axis
Dim pos As ADOMD.Position
Dim iCol As Integer, cCol As Integer
Dim iRow As Integer, cRow As Integer
Dim nFixedCols As Integer, nFixedRows As Integer
  'Set up the connection to the server.
  conn.ConnectionString = "Datasource=LocalHost; Provider=msolap; Initial Catalog=FoodMart
2000;"
  conn.Open
  Set cst.ActiveConnection = conn ' You must use Set.
  cst.Source = "Select CrossJoin [Product].[Product Family].Members, " & _
    "[Promotion Media].Members)  on rows," & _
    "[Measures].Members on Columns " & _
    "From Sales"
  cst.Open

  'Set up the FlexGrid control.
  MSFlexGrid1.Clear
  nFixedCols = 2
  nFixedRows = 1
  cCol = cst.Axes(0).Positions.Count
  MSFlexGrid1.Cols = cCol + nFixedCols
  cRow = cst.Axes(1).Positions.Count
  MSFlexGrid1.Rows = cRow + nFixedRows
  MSFlexGrid1.FixedCols = nFixedCols
```

```
MSFlexGrid1.FixedRows = nFixedRows
   MSFlexGrid1.MergeCol(0) = True
MSFlexGrid1.MergeCol(1) = True

'Add column headers.
iCol = 2
For Each pos In cst.Axes(0).Positions
   'The caption for each member is used as the header.
   MSFlexGrid1.TextMatrix(0, iCol) = pos.Members(0).Caption
   iCol = iCol + 1
Next

'Add row headers.
iRow = 1
For Each pos In cst.Axes(1).Positions
   'The CrossJoin function in MDX indicates that this axis will have two members per position.
   MSFlexGrid1.TextMatrix(iRow, 0) = pos.Members(0).Caption
   MSFlexGrid1.TextMatrix(iRow, 1) = pos.Members(1).Caption
   iRow = iRow + 1
Next

'Iterate through the cellset array values.
For iCol = 0 To cCol - 1
   For iRow = 0 To cRow - 1
      ' Retrieve each value with the default method of the cst object.
      MSFlexGrid1.TextMatrix(iRow + nFixedRows, iCol + nFixedCols) = cst(iCol, iRow).Value
   Next
Next
```

บทที่ 7
การออกแบบและพัฒนาโปรแกรมประยุกต์ ในลักษณะ Client - Server

โปรแกรมประยุกต์ที่ต้องการพัฒนาขึ้นมาจะมีลักษณะ รันอยู่บนเครื่องฝั่งไคลเอ็นต์โดยมีหน้าที่การทำงานของโปรแกรมคือ สามารถเรียกดูและใช้งานข้อมูลที่อยู่ภายในคิวบ์ซึ่งอยู่ที่เครื่อง Analysis Server ได้

7.1 จุดประสงค์ในการพัฒนาโปรแกรม

1. เพื่อให้ผู้ใช้สามารถเรียกใช้งานระบบฐานข้อมูลคลาต้าแวร์เฮาส์ที่อยู่บนฝั่งซิร์ฟเวอร์จากเครื่องของตนเองซึ่งไม่ได้ลงโปรแกรม Analysis Service ได้
2. เพื่อทดสอบประสิทธิภาพและเวลาตอบสนอง (response time) เมื่อเรียกใช้งานระบบฐานข้อมูลคลาต้าแวร์เฮาส์ผ่านทางเครื่องไคลเอ็นต์

7.2 ทูลส์ที่นำมาใช้ในการพัฒนาโปรแกรม

ในการพัฒนาโปรแกรมประยุกต์นี้ ได้เลือกใช้ Visual Basic เวอร์ชัน 6.0 มาเป็นทูลส์ในการพัฒนา เนื่องจากเป็นทูลส์ที่ใช้งานง่ายและเนื่องจากเป็นผลิตภัณฑ์ของ Microsoft เองด้วยจึงมีการเข้ากันได้ดีกับ Microsoft SQL2000 Server Analysis Service และอีกประการหนึ่งที่เลือกใช้ทูลส์ตัวนี้คือเป็นทูลส์ที่มีความโดดเด่นทางด้านคอมโพเนนต์ด้วย

7.3 การออกแบบโปรแกรมประยุกต์

โดยหลักการของคลาต้าแวร์เฮาส์แล้ว การเรียกใช้งานจะมุ่งเน้นไปที่การเรียกดูข้อมูลเพื่อการวิเคราะห์เพื่อให้ได้มาซึ่งคำตอบที่ต้องการ จะนำไปใช้ตัดสินใจอย่างรวดเร็ว ดังนั้นโปรแกรมที่พัฒนาจึงไม่ได้มุ่งไปที่การอัพเดตข้อมูล และได้ออกแบบมาอย่างเรียบง่ายเพื่อให้ง่ายต่อการทำความเข้าใจและใช้งานได้อย่างสะดวก

โปรแกรมที่ได้ทำการออกแบบขึ้นมาจะมีฟอร์มอยู่ด้วยกัน 2 ฟอร์ม คือ

- ฟอร์มหลักใช้ติดต่อกับ Analysis Server
  ฟอร์มนี้จะประกอบไปด้วย เท็กบ็อก (Text Box) 3 ช่อง, บัททอน (Button) 2 ปุ่ม, และมีเพลน (Plain) สำหรับแสดงผลอีก 2 เพลน โดยเพลนด้านบนจะแสดงผล Dimension ที่มีใน Cube ที่ทำการเลือก ส่วนด้านล่างจะเป็นรายละเอียดของข้อมูล
- ฟอร์มในส่วนของ MDX Query
  ฟอร์มนี้จะประกอบไปด้วยคอมโบบ็อกซ์ (Combo Box) 3 ช่อง,เมนู ,toolbar และมีเพลน (Plain) สำหรับแสดงผลอีก 2 เพลน โดยเพลนด้านบนจะมีไว้ใส่คำสั่ง MDX ที่มีในคิวบ์ที่ผู้ใช้เลือก ส่วนในเพลนด้านล่างจะใช้สำหรับแสดงผลลัพธ์จากการคิวรีของผู้ใช้

7.4 คอมโพเนนต์หลักที่ใช้ในการพัฒนาโปรแกรม

คอมโพเนนต์ที่ใช้พัฒนาโปรแกรมประกอบไปด้วย

Button ปุ่มกด เมื่อมีการกดปุ่มโปรแกรมจะไปทำงานตามขั้นตอนที่กำหนดไว้

Menu มีไว้เลือกการทำงานของส่วนต่างๆๆของโปรแกรม

Toolbar ใช้เป็นปุ่มลัดของเมนูการทำงานต่างๆ

Text Box เป็นช่องใช้สำหรับให้ผู้ใช้กรอกชื่อเซิร์ฟเวอร์ที่ต้องการติดต่อ กับเลือก Provider ของ Server

Combo Box จะเป็นลักษณะของ drop down list ข้อมูลที่เป็นไปได้ทั้งหมดประกอบอยู่ในนี้แล้ว ผู้ใช้เพียงแต่เลือกอันใดอันหนึ่งเท่านั้น

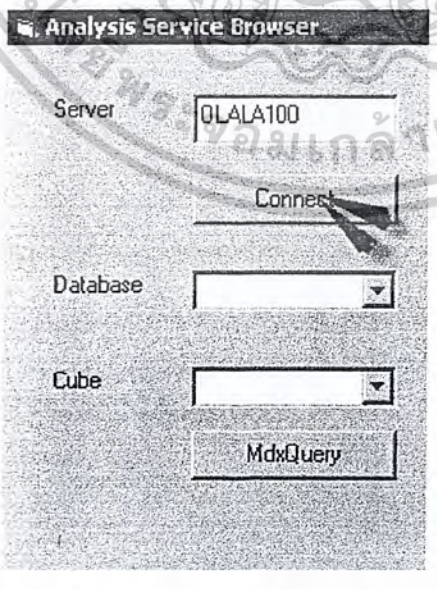
Cube Browser เป็นคอมโพเนนต์ที่ไม่ใช่คอมโพเนนต์พื้นฐานของวิชวลเบสิค โดยเมื่อเราต้องการจะใช้งานเราจะต้องเพิ่มคอมโพเนนต์นี้เข้ามาก่อน ซึ่งคอมโพเนนต์นี้จะมีให้หลังจากเราได้ลงโปรแกรม Microsoft SQL2000 Analysis Service, Cube Browser เป็นคอมโพเนนต์ที่ทำให้เราสามารถพัฒนาโปรแกรมได้อย่างง่าย สะดวกและรวดเร็ว เนื่องจากเราเพียงแค่กำหนดพารามิเตอร์ที่จำเป็นในการเชื่อมต่อกับ Analysis Server แค่นั้นก็พอ หลังจากนั้นตัวคอมโพเนนต์จะไปเชื่อมต่อและดึงข้อมูลจาก Analysis Server มาให้ ซึ่งลักษณะการทำงานของโปรแกรมจะเหมือนกันเราทำการดูข้อมูลในคิวบ์ ในโปรแกรม Analysis manager เลยสามารถ drill up, drill down, slice และเลือกไดเมนชันที่ต้องการให้แสดงผลได้

Query Pane เป็น User Control สร้างขึ้นมาเพื่อไว้ใช้ใส่คำสั่ง MDX

Result Pane เป็น User Control ที่ไว้แสดง ผลลัพธ์ที่ได้จากการใช้คำสั่ง MDX

7.5 การใช้งานโปรแกรมประยุกต์

เริ่มแรกเมื่อรันโปรแกรม สิ่งที่ผู้ใช้ต้องทำคือ กรอกชื่อ Analysis Server ที่ต้องการติดต่อ หลังหลังจากนั่นให้คลิกปุ่มconnect หากข้อมูลที่ใส่ไปนั้นถูกต้อง โปรแกรมก็จะไปเชื่อมต่อกับเซิร์ฟเวอร์เครื่องนั้น



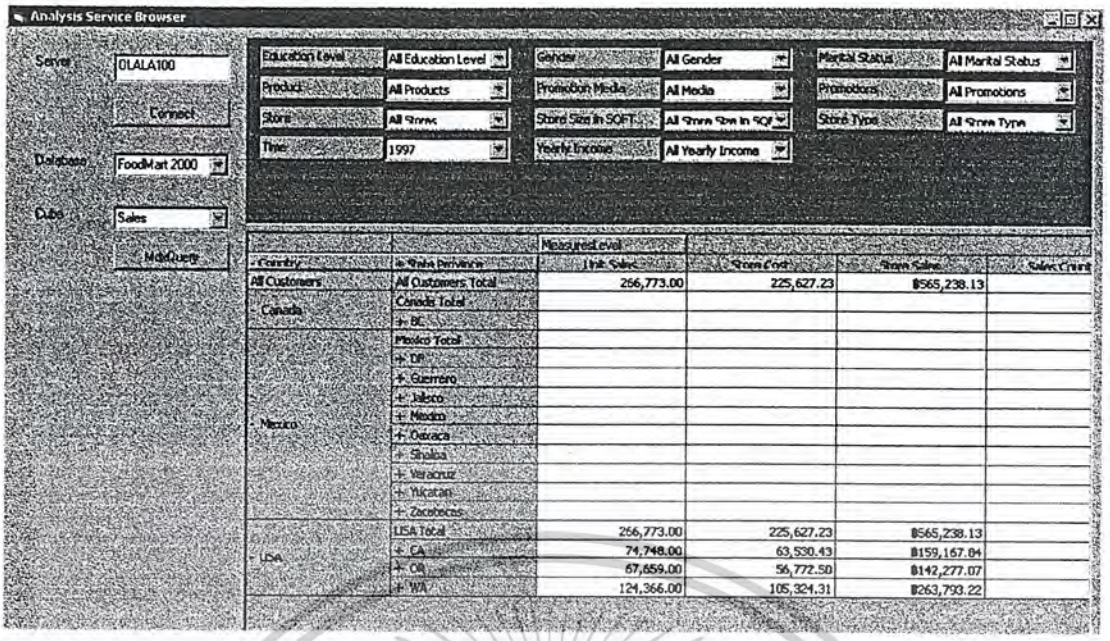
รูปที่ 7-3 แสดงการติดต่อกับ Analysis Server

โดยในช่อง Database name เมื่อคลิกที่ปุ่มลูกศร โปรแกรมจะแสดงรายชื่อฐานข้อมูลทั้งหมดที่มีใน Analysis Server นี้ หลังจากเราเลือกฐานข้อมูลเรียบร้อยแล้ว ในช่อง Cube name ก็จะปรากฏชื่อคิวบ์ทั้งหมดที่มีในฐานข้อมูลนี้ และหลังจากเลือกคิวบ์แล้ว ข้อมูลจากคิวบ์ที่เลือกก็จะปรากฏออกมา ดังรูปที่ 7-5

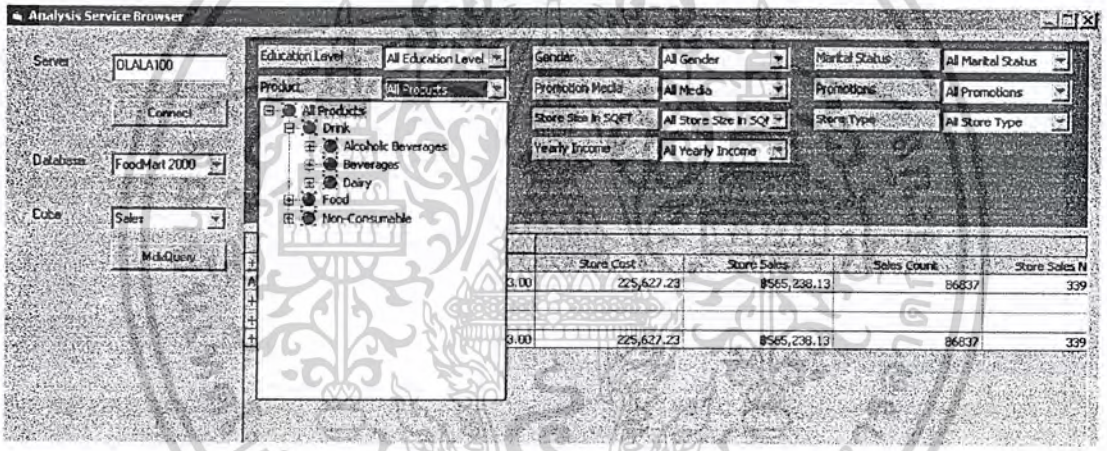
รูปที่ 7-4 แสดงการเลือก Cube

จากนั้นเราก็สามารถใช้งานโปรแกรมนี้ได้เหมือนกับเหมือนกับทำ Browse data ในโปรแกรม Analysis manager ทุกอย่าง เช่น การ drill down จะสังเกตเห็นว่าบางแถวมีเครื่องหมาย + อยู่ข้างหน้านั่นคือ สามารถ drill down ดูข้อมูลในระดับที่ลึกลงไปได้โดยการดับเบิลคลิก ซึ่งจะได้ผลดังรูปที่ 7-5

จากรูปที่ 7-6 จะเห็นว่าเราจะเห็นรายละเอียดในระดับที่ลึกลงไปของแถว Food การ slice ดูบางส่วนของข้อมูลก็สามารถทำได้โดยการคลิกที่ปุ่มลูกศรของไดเมนชันใดๆ ที่อยู่ในไดเมนชันเพลน ซึ่งเมื่อคลิกแล้วโปรแกรมจะแสดงรายละเอียดในเลเวลต่างๆ ของไดเมนชันนั้น



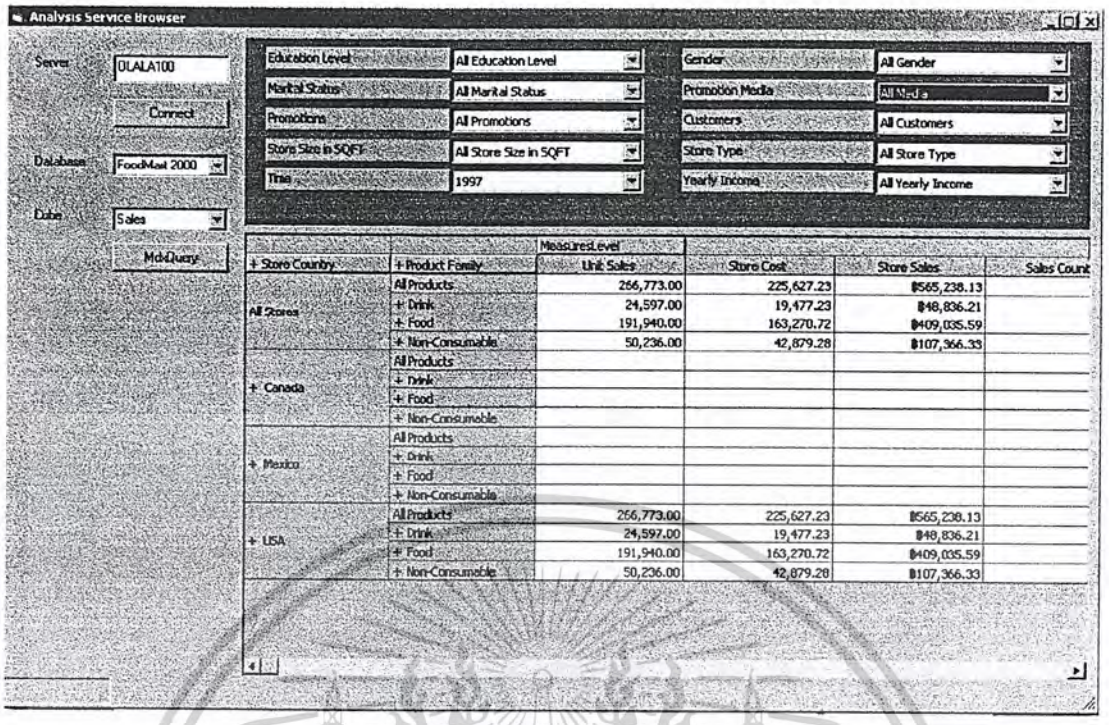
รูปที่ 7-5 แสดงการเลือก drill down



รูปที่ 7-6 แสดงการเลือก slice

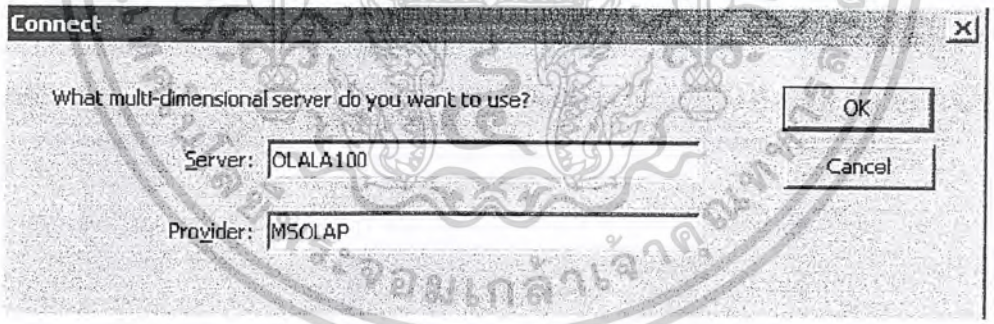
การทำ drill-up ก็จะสามารถทำได้ในลักษณะที่คล้ายกับการทำ drill-down โดยการ drill-up สามารถทำได้โดยการดับเบิลคลิกตรงช่องที่มีเครื่องหมาย – อยู่ข้างหน้า โปรแกรมก็จะรวมช่องแสดงผลย่อยขึ้นมาแสดงรวมในช่องเดียว

การทำ dice สามารถทำได้โดยการคลิกที่ชื่อไดเมนชันในไดเมนชันเพลนแล้วลากมาวางยังตารางแสดงผล ซึ่งจะเป็นการเพิ่มไดเมนชันเข้าไปในตารางหรือเป็นการสลับกับไดเมนชันที่มีอยู่แล้วในตารางก็ขึ้นอยู่กับตำแหน่งที่วางลงไป อีกทั้งยังสามารถสลับการแสดงผลระหว่างแถวกับคอลัมน์ของไดเมนชันในตารางได้



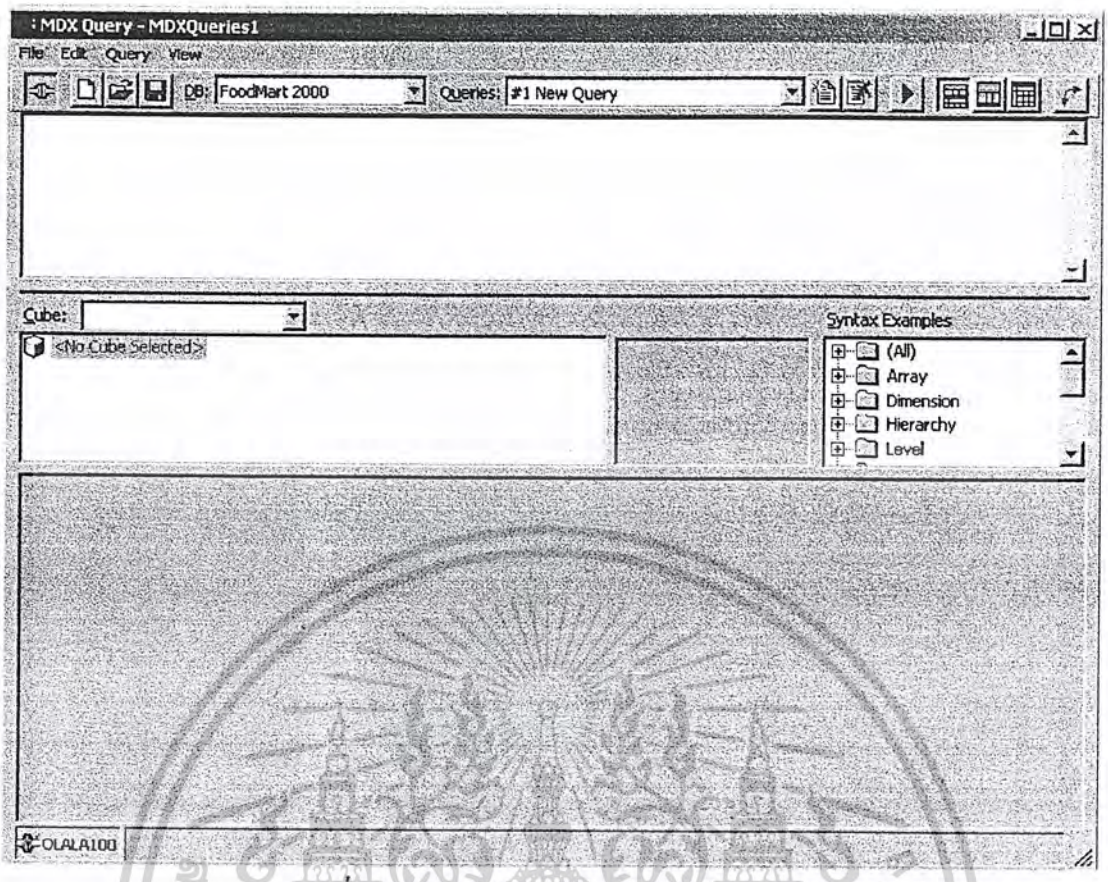
รูปที่ 7-7 แสดงการสลับไดเมนชันและการแสดงผลหลาย ๆ ไดเมนชัน

เมื่อต้องการทำ MDX Query ให้คลิกที่ปุ่ม MDX Query เมื่อคลิกแล้วจาปรากฏฟอร์มขึ้นมาให้เราใส่ชื่อ เซฟเวอร์ขึ้นมาดังรูปที่ 7-8



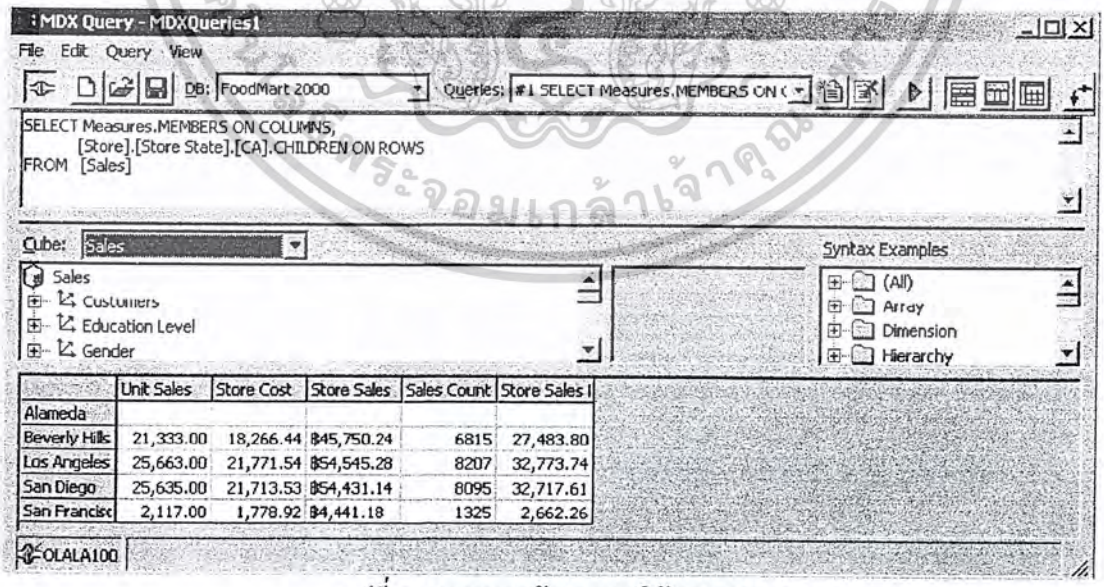
รูปที่ 7-8 แสดงหน้าจออ็อกอินใช้งาน MDX Query

จากนั้นจะปรากฏฟอร์มการใช้งานMDX Query ขึ้นมาโดยจะมีเมนูและทูลบาร์ให้เลือกใช้ตามการใช้งานการทำงานดังรูปที่ 7-9



รูปที่ 7-9 แสดงหน้าจอล็อกอินใช้งาน ใช้งานMDX Query

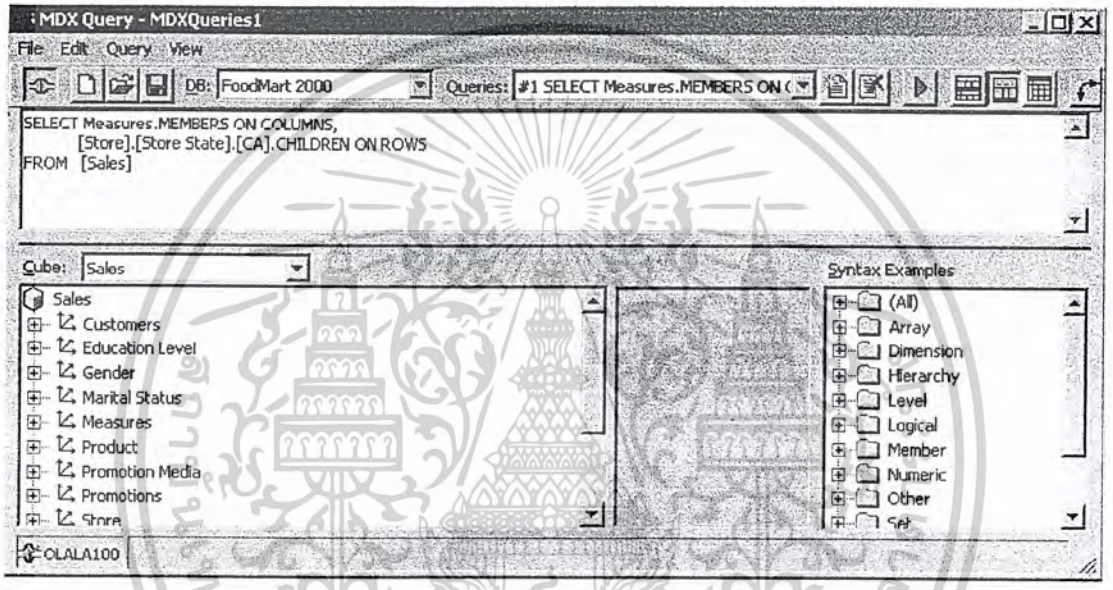
การใช้งานเบื้องต้นของฟอร์มนี้จากรูปที่ 7-9 จะมีPaneให้ใส่คำสั่ง MDX จากนั้น ให้คลิกปุ่ม ▶ Run Query หรือ เมนู Query>Run หรือ F5 ผลลัพธ์ที่ได้จะปรากฏดังรูปที่ 7-10



รูปที่ 7-10 แสดงหน้าทดลอง ใช้งานMDX Query

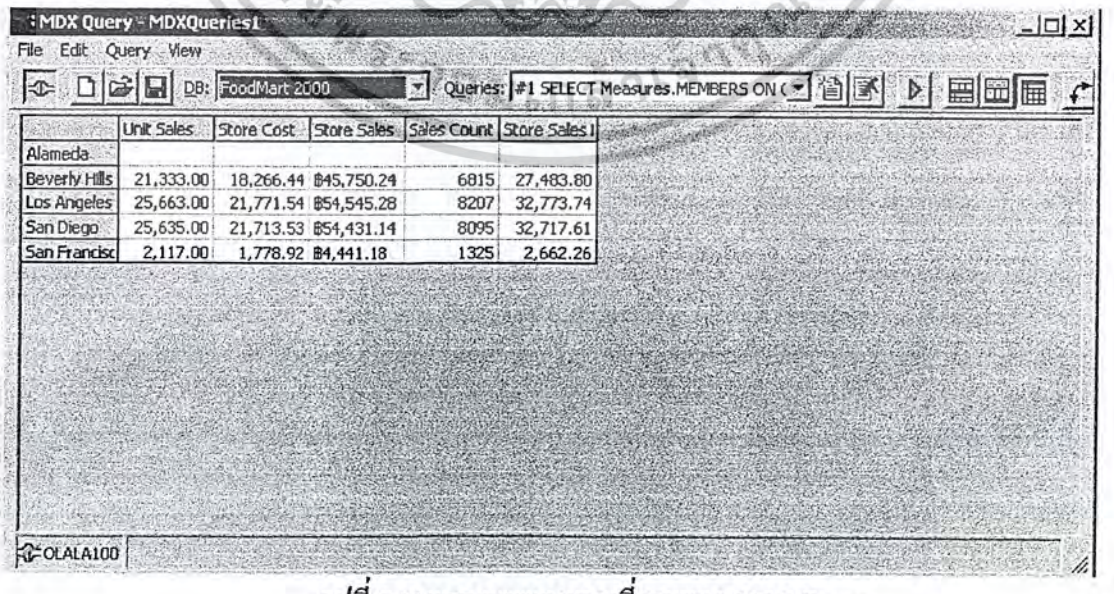
เมื่อเขียนคำสั่งที่ต้องการเรียบร้อยแล้วเราสามารถเซฟคำสั่งMDX ได้โดยใช้ปุ่ม หรืออาจจะใช้เมนู File>save ได้เช่นกัน ถ้าต้องการสร้างไฟล์ใหม่ให้ใช้เมนู File>New หรือปุ่ม

เมื่อต้องการเปิด file ที่เขียนคำสั่ง MDX ไว้ให้ใช้ ปุ่ม หรือเมนู File>Open เมื่อเปิดไฟล์ขึ้นมาแล้วแล้วต้องการเขียนคำสั่งใหม่ให้ใช้ปุ่ม หรือเมนู Query>new และถ้าต้องการลบคำสั่งMDX ให้ใช้ปุ่ม หรือเมนู Query>Delete เมื่อต้องการเปลี่ยน view ของฟอร์มนี้เราสามารถเปลี่ยนได้ 3 แบบคือ แบบ Split , Query และResults ถ้าเราเปลี่ยนเป็นมุม Split จะเป็นรูปแบบปกติ แบบที่สองแบบQuery จะแสดงดังรูปที่ 7-11



รูปที่ 7-11 แสดงมุมมองแบบที่สองของ MDX Query

แต่ถ้าเปลี่ยนเป็น Results จะได้เป็นดังรูปที่ 7-12



รูปที่ 7-12 แสดงมุมมองแบบที่สามของ MDX Query

ในบางครั้งผลลัพธ์ที่แสดงมาสามารถที่จะสลับแกนของผลลัพธ์ได้โดยใช้ปุ่ม หรือเมนู View>Pivot Results ผลลัพธ์ของการกดปุ่มจาแสดงดังรูปที่ 7-13ก กับรูปที่ 7-13ข

	Unit Sales
Daily Paper,	9,513.00
Daily Paper	7,738.00
Product Atta	7,544.00
Daily Paper,	6,891.00
Cash Regist	6,697.00
Sunday Pap	5,945.00
Street Hand	5,753.00
Sunday Pap	4,339.00
Bulk Mail	4,320.00
In-Store Cou	3,798.00
TV	3,607.00
Sunday Pap	2,726.00
Radio	2,454.00

	Daily Paper,	Daily Paper	Product Atta	Daily Paper,
Unit Sales	9,513.00	7,738.00	7,544.00	6,891.00

ก : แสดง view ก่อนใช้ Pivot Results **ข : แสดง view หลังการใช้ Pivot Results**

รูปที่ 7-13 แสดง View ก่อนใช้pivot Results และ แสดงภาพหลังการใช้ Results

และเมื่อต้องการDisconnect กับ Analysis Server ให้ใช้ปุ่ม หรือเมนู File>Disconnect

บทที่ 8
การทดสอบประสิทธิภาพการทำงานของโปรแกรมประยุกต์และการเก็บข้อมูลของ Analysis Service

ประสิทธิภาพในการคิวรีข้อมูลมีความสำคัญอย่างยิ่งในการศึกษาการเก็บข้อมูลแบบคาต้าแวเฮาส์ เนื่องจากต้องออกรายงานเพื่อการวิเคราะห์ให้ผู้บริหารให้รวดเร็ว การศึกษาทั้งความเร็วในการคิวรีของโปรแกรมประยุกต์ เวลาและเนื้อที่ที่ต้องเสียไปในการสร้างวิธีการเก็บข้อมูลแบบต่างๆ จึงมีความสำคัญเป็นอย่างมากเพื่อใช้ในการตัดสินใจของผู้ดูแลระบบว่าควรเลือกใช้วิธีการแบบใดในการเก็บข้อมูลที่ดีที่สุด

8.1 การทดสอบประสิทธิภาพ

การทดสอบการทำงานในด้าน ความสามารถออกรายงานได้มีประสิทธิภาพแค่ไหนในสภาวะแวดล้อมที่ต่างกัน ซึ่งการทดสอบจะแบ่งเป็น 2 ส่วน ในส่วนแรกเป็นการทดสอบการทำงานของ Microsoft SQL Server 2000 Analysis Service ในเรื่องจำนวนที่มากขึ้นของปริมาณข้อมูลเปรียบเทียบกับประสิทธิภาพการทำงาน ในส่วนที่สองจะเป็นการวัดประสิทธิภาพการทำงานของโปรแกรมประยุกต์โดยการเพิ่มปริมาณคำสั่งให้มากขึ้น

8.1.1 ส่วนที่หนึ่ง ทดสอบประสิทธิภาพของ Microsoft SQL Server 2000 Analysis Service

โดยใช้ฮาร์ดแวร์ดังนี้

1. Mainboard – P4C800
2. CPU Processor – Pentium IV 2.4 GHz
3. DDR-RAM – 2 GB
4. HardDisk – Serial ATA 120 GB
5. Network Interface Card (NIC) – 1 Gbit
6. Category 5 100 Mbit

รายละเอียดของ Dimension ใน Bill ทั้งหมด 12 Dimensions

Level	Source Column	Source Table	Member Count
Base Unit	Base_unit.Base_unit	Base_unit	12
Base Unit Engdes	Base_unit.Base_unit_EngDes	Base_unit	12

ตารางที่ 8-1 แสดงองค์ประกอบของ Base Unit Dimensions

Level	Source Column	Source Table	Member Count
Sales Unit Engdes	Sales_unit.Sales _Unit_EngDes	Sales_unit	21
Sales Unit	Sales_unit.Sales_unit	Sales_unit	23

ตารางที่ 8-2 แสดงองค์ประกอบของ Sale Unit Dimensions

Level	Source Column	Source Table	Member Count
Sales Org Des	Sales_org.Sales_Org_Des	Sales_Org	19
Sales Org	Base_unit.Sales_Org	Sales_Org	19

ตารางที่ 8-3 แสดงองค์ประกอบของ Sale Org Dimensions

Level	Source Column	Source Table	Member Count
Sales Man Name	Sales_Man.Sales_Man_Name	Sales_man	81
Sales Man	Sales_Man.Sales_Man	Sales_man	81

ตารางที่ 8-4 แสดงองค์ประกอบของ Sale Man Dimensions

Level	Source Column	Source Table	Member Count
Company Code	Company_Code.Company_Code	Company_code	4

ตารางที่ 8-5 แสดงองค์ประกอบของ Company Dimensions

Level	Source Column	Source Table	Member Count
Customer Group Des	Customer_Group.Customer_Group_Des	Customer_Group	3
Business Group Des	Business_Group.Business_Group_Des	Business_Group	25
Customer	Customer.Customer	Customer	1225

ตารางที่ 8-6 แสดงองค์ประกอบของ Customer Dimensions

Level	Source Column	Source Table	Member Count
Distr Channel Group	Distr_Channel.Distr_Channel_Group	Distr_Channel	3
Distr Channel Des	Distr_Channel.Distr_Channel_Des	Distr_Channel	6
Distr Channel	Distr_Channel.Distr_Channel	Distr_Channel	6

ตารางที่ 8-7 แสดงองค์ประกอบของ Distri Channel Dimensions

Level	Source Column	Source Table	Member Count
End User Name	End_User.End_User_Name	End_User	10
End User	End_User.End_User	End_User	10

ตารางที่ 8-8 แสดงองค์ประกอบของ End User Dimensions

Level	Source Column	Source Table	Member Count
SP Group	SP_Group.Sp_Group	SP_Group	1
Division	Division.Division	Division	23
Material 1	Material.Material_1	Material	51
Material 3	Material.Material_3	Material	266
Material 4	Material.Material_4	Material	731
Material 9	Material.Material_9	Material	9054
Material	Material.Material	Material	83776

ตารางที่ 8-9 แสดงองค์ประกอบของ Material1 Dimensions

Level	Source Column	Source Table	Member Count
Year	DatePart(year,Bill_Fact.Bill_Date)	Bill_Fact	3
Quarter	DatePart(quarter,Bill_Fact.Bill_Date)	Bill_Fact	9
Month	DatePart(month,Bill_Fact.Bill_Date)	Bill_Fact	25

ตารางที่ 8-10 แสดงองค์ประกอบของ Bill Date Dimensions

Level	Source Column	Source Table	Member Count
Billing Type Group	Billing_Type.Billing_Type_Group	Billing_Type	4
Billing Type	Billing_Type.Billing_Type	Billing_Type	334

ตารางที่ 8-11 แสดงองค์ประกอบของ Bill Type Dimension

Level	Source Column	Source Table	Member Count
BOI Des	Storage_Location.BOI_Des	Storage_Location	7
Storage Location Des	Storage_Location.Storage_Location_Des	Storage_Location	60
Storage Location	Storage_Location.Storage_Location	Storage_Location	457

ตารางที่ 8-12 แสดงองค์ประกอบของ Storage Location Dimensions

Measures ที่ใช้ใน Bill คือ

- Bill R Qty Sale
- Bill R Qty Base
- CD Qty Sale
- CD Qty Base
- Amount Net

Response Time ที่ได้จากการ ทำงาน

- **Designing Storage**

Designing Storage							
จำนวน Rows(Rows)	22,009	42,213	67,370	102,484	125,522	145,724	174,598
ช่วงข้อมูล	3 เดือน	6 เดือน	9 เดือน	1 ปี	1 ปี 3 เดือน	1 ปี 6 เดือน	1 ปี 9 เดือน
15% Performance	0.14	0.21	0.29	0.56	1.01	1.29	1.3
25 % Performance	1.25	2.57	4.48	10	13.35	15.21	17.33
35 % Performance	2.28	5.15	9.1	21.38	29.34	39.1	44.26
50 % Performance	4.87	22.06	30.41	42.31	46.1	48.3	
75 % Performance	12.4	44.03	60.4	64.3	119.45	145	127.35

ตารางที่ 8-13 แสดงผลลัพธ์ของ Designing Storage จากการทดสอบ

- จำนวน Aggregations

Aggregation Numbers							
จำนวน Rows(Rows)	22,009	42,213	67,370	102,484	125,522	145,724	174,598
ช่วงข้อมูล	3 เดือน	6 เดือน	9 เดือน	1 ปี	1 ปี 3 เดือน	1 ปี 6 เดือน	1 ปี 9 เดือน
15 % Performance	57	74	89	98	108	120	128
25 % Performance	163	211	264	300	321	330	351
35 % Performance	244	314	385	448	490	511	548
50 % Performance	291	379	457	685	736	762	
75 % Performance	529	661	817	1,188	1,266	1,318	1,453

ตารางที่ 8-14 แสดงผลลัพธ์ของจำนวน Aggregations จากการทดสอบ

- **Storage Aggregations**

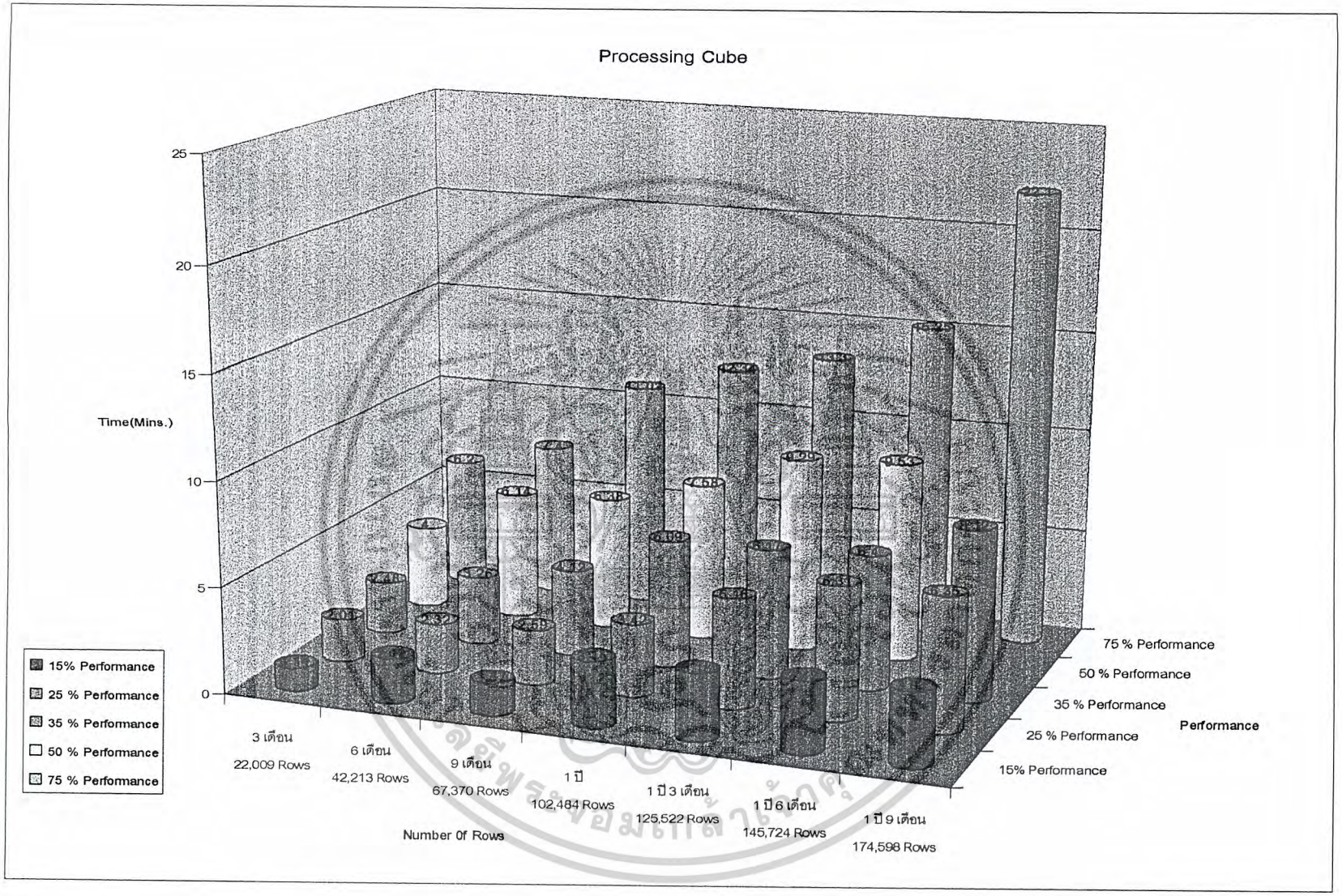
Aggregation Storage							
จำนวน Rows(Rows)	22,009	42,213	67,370	102,484	125,522	145,724	174,598
ช่วงข้อมูล	3 เดือน	6 เดือน	9 เดือน	1 ปี	1 ปี 3 เดือน	1 ปี 6 เดือน	1 ปี 9 เดือน
15 % Performance	0.2	0.3	0.5	0.6	0.7	1	1.1
25 % Performance	2.1	4.4	8	12	14.8	16.1	19.1
35 % Performance	6	12.8	24	38.4	49.8	58.6	72.8
50 % Performance	13.7	29.9	54.1	116.6	149	174.8	
75 % Performance	46.5	108.9	203.5	437.1	560	668.2	862.8

ตารางที่ 8-15 แสดงผลลัพธ์ของจำนวน Aggregation Storage จากการทดสอบ

- **Processing Cube**

Processing Cube							
จำนวน Rows(Rows)	22,009	42,213	67,370	102,484	125,522	145,724	174,598
ช่วงข้อมูล	3 เดือน	6 เดือน	9 เดือน	1 ปี	1 ปี 3 เดือน	1 ปี 6 เดือน	1 ปี 9 เดือน
15% Performance	1.11	2.12	1.57	3.14	3.18	3.42	3.54
25 % Performance	2.01	2.32	2.59	3.42	5.16	6.31	6.35
35 % Performance	2.48	3.26	4.12	6.09	6.17	6.4	8.12
50 % Performance	4	6.14	6.38	7.58	9.29	9.53	
75 % Performance	6.2	7.4	11.02	12.34	13.3	15.25	22.1

ตารางที่ 8-16 แสดงผลลัพธ์ของจำนวน Processing Cube จากการทดสอบ



รูปที่ 8-1 แสดงผลลัพธ์ของจำนวน Processing Cube จากการทดสอบ

8.1.2 ส่วนที่สอง การวัดประสิทธิภาพการทำงานของโปรแกรมประยุกต์

โดยใช้ฮาร์ดแวร์และซอฟต์แวร์ดังนี้

เซิร์ฟเวอร์

- AMD Duron 1.2GHz
- DDR Ram 768MB
- HD 40 GB ATA/100
- Analysis Server , Microsoft SQL Server 2000

ไคลเอ็นต์

- Pentium 4 1.6GHz
- RD Ram 256MB PC800
- HD 40 GB ATA/100
- PivotTable service
- โปรแกรมประยุกต์ที่ใช้ คอมโพเนนท์ ของ Microsoft SQL Server 2000 นำมาใช้กับ Visual Basic 6.0

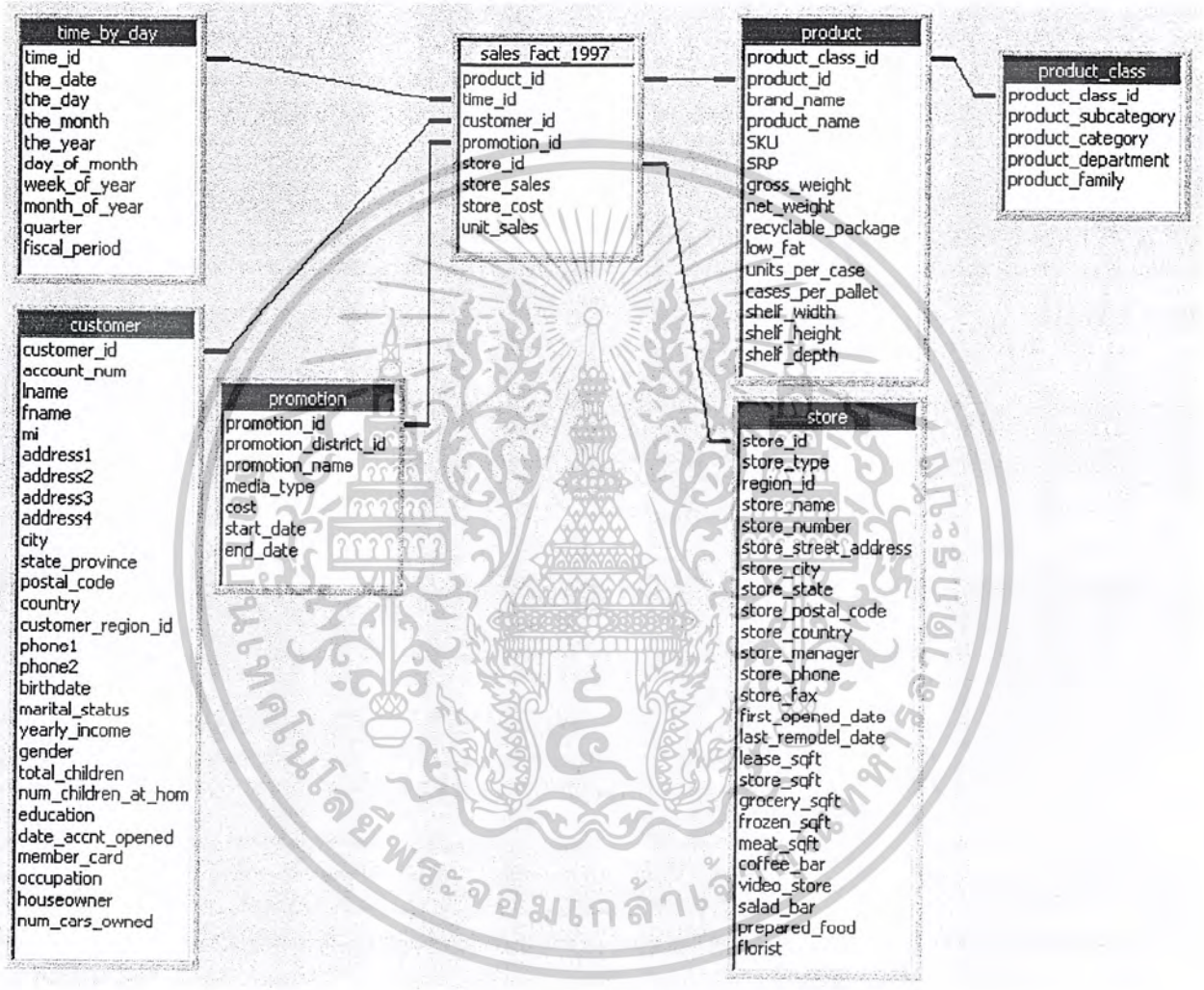
ผลจากการทดลอง

จำนวนคำสั่ง	1	5	10	15	20	25	30	35	50	100	200	500
เวลาที่ใช้ (วินาที)	0.79	2.18	4.53	6.80	9.24	11.66	14.12	16.17	23.23	48.41	1.41.13	5.01.72

ตารางที่ 8-17 แสดงผลลัพธ์ จากการทดสอบการเพิ่มจำนวนคำสั่ง

ภาคผนวก ก
ตัวอย่างคำสั่ง MDX

ก่อนที่จะดูตัวอย่างการใช้งานคำสั่ง MDX จะขออธิบาย Cube ที่จะใช้ในตัวอย่างก่อน Cube ที่ใช้อยู่ภายใต้ Database ที่ชื่อ FoodMart2000 มีชื่อว่า Sales Cube รายละเอียดมีดังนี้



รูปที่ ก-1 แสดง Schema Cube Sales

Dimension name	Level(s)	Description
Store	Store Country Store State Store City Store Name	Geographical hierarchy for stores โดยใน Store Name จะมี Member Properties คือ Store Manager, Store SQFT, Store Type
Time	Year Quarter Month	Time period
Product	Product Family Product Department Product Category Product Subcategory Band Name Product Name	Product ที่อยู่ใน Store
Promotion media	Media Type	Media ที่ใช้เพื่อการ promotion
Promotion	Promotion Name	name of promotion
Customers	Country State City Name	Geographical hierarchy for Customer ของแต่ละ stores ในส่วน NAME มี Member Properties เป็น Marital Status, Education, Yearly Income, Member Card
Gender	Gender	เพศของ Customer : "M" หรือ "F"
Marital Status	Marital Status	สถานะภาพการสมรส : "S" หรือ "M"
Yearly Income	Yearly Income	เงินเดือนของ Customer

ตารางที่ ก-1 แสดง Sales Cube Dimension

Measures	Description
Unit Sales	จำนวนของ Units sale :Sum
Store Cost	Cost ของ goods sale L: Sum
Store Sales	Value of sales transactions. : Sum
Sales Count	Number of sales transactions. :Count
Store Sales Net	Value of sales transactions less cost of goods sold. :Sum

***ตารางที่ ก-2 แสดง* Sales cube Measures**

Calculated Members	Description
Sale Average	[Measures].[Store Sales]-[Measures].[Store Cost]
Profit	[Measures].[Store Sales]/[Measures].[Sales Count]

ตารางที่ ก-3 แสดง Sales cube Calculated Members

คำสั่งใน MDX สามารถแบ่งตามลักษณะการใช้งานเป็นกลุ่ม ๆ ได้ดังนี้

1. Array Functions

1.1 SetToArray

Syntax

SetToArray(«Set»[, «Set»...][, «Numeric Expression»])

2. Dimension, Hierarchy, and Level Functions

2.1 Dimension

Return dimension ซึ่ง member, level หรือ hierarchy นั้นเป็นสมาชิกอยู่

Syntax

Member «Member».**Dimension**

Level «Level».**Dimension**

Hierarchy «Hierarchy».**Dimension**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[Store Country].Dimension} ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
All Stores	266,773.00	225,627.23	฿565,238.13	86837	339,610.90

รูปที่ ก-2 แสดงตัวอย่างการใช้ฟังชันก์ Dimension

2.2 Dimensions

Return dimension ที่ต้องการโดยใช้การกำหนด dimension ด้วยค่า Numeric หรือ String

Syntax

Numeric **Dimensions**(«Numeric Expression»)

String **Dimensions**(«String Expression»)

ตัวอย่าง

```
SELECT {Dimensions(0).Members} ON COLUMNS,
        {Dimensions(1)} ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
All Stores	266,773.00	225,627.23	$565,238.13	86837	339,610.90

รูปที่ ก-3 แสดงตัวอย่างการใช้ฟังชันก์ Dimensions

2.3 Hierarchy

Return hierarchy ที่ member หรือ level นั้นเป็นสมาชิก

Syntax

Member «Member».**Hierarchy**

Level «Level».**Hierarchy**

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
        [Store].[Store Name].Hierarchy.MEMBERS ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
All Stores	266,773.00	225,627.23	$565,238.1	86837	339,610.90
Canada					
BC					
Vancouver					
Store 19					
Victoria					
Store 20					
Mexico					
DF					
Mexico City					
Store 9					
San Andres					
Store 21					
Guerrero					
Acapulco					
Store 1					

รูปที่ ก-4 แสดงตัวอย่างการใช้ฟังชันก์ Hierarchy

2.4 Level

Return level ที่ member นั้นเป็นสมาชิกอยู่

Syntax

«Member».**Level**

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
    [Store].[Store 18].Level.MEMBERS ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales
Store 19					
Store 20					
Store 9					
Store 21					
Store 1					
Store 5					
Store 10					
Store 8					
Store 4					
Store 12					
Store 18					
HQ					
Store 6	21,333.00	18,266.44	฿45,750.24	6815	27,483.80
Store 7	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
Store 24	25,635.00	21,713.53	฿54,431.14	8095	32,717.61
Store 14	2,117.00	1,778.92	฿4,441.18	1325	2,662.26
Store 11	26,079.00	21,948.94	฿55,058.79	8264	33,109.85
Store 13	41,580.00	34,823.56	฿87,218.28	13347	52,394.72
Store 2	2,237.00	1,896.62	฿4,739.23	1380	2,842.61
Store 3	24,576.00	21,121.96	฿52,896.30	7876	31,774.34
Store 15	25,011.00	20,956.80	฿52,644.07	7956	31,687.27
Store 16	23,591.00	19,795.49	฿49,634.46	7397	29,838.97
Store 17	35,257.00	29,959.28	฿74,843.96	11184	44,884.68
Store 22	2,203.00	1,880.34	฿4,705.97	1339	2,825.63
Store 23	11,491.00	9,713.81	฿24,329.23	3652	14,615.42

รูปที่ ก-5 แสดงตัวอย่างการใช้ฟังชันก์ **Level**

2.5 Levels

Return level ที่ต้องการโดยใช้การกำหนด level ด้วยค่า Numeric หรือ String

Syntax

Numeric «Dimension».**Levels**(«Numeric Expression»)

String **Levels**(«String Expression»

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
        {[Store].Levels(4).MEMBERS}  ON ROWS
FROM Sales
```

2. Logical Functions

2.1 Is

Returns TRUE ถ้า Object ทั้งสองเท่ากัน ถ้าไม่เท่า return FALSE

Syntax

«Object 1» **IS** «Object 2»

Alternate Syntax

«Object 1» **IS NULL**

2.2 IsAncestor

Returns TRUE ถ้า Member1 เป็น Ancestor ของ Member2 ถ้าไม่ใช่ return FALSE

Syntax

IsAncestor(«Member1»,«Member2»)

ตัวอย่าง

```
WITH  MEMBER [Measures].[Leaf] AS
'IIF(IsAncestor([Time].[1998],[Time].[1997]),"Yes","No"  )'
SELECT {[Time].[1998]}  ON COLUMNS,
        { [Measures].[Leaf] }ON ROWS
FROM Sales
```

	1998
Leaf	No

รูปที่ ก-6 แสดงตัวอย่างการใช้ฟังชันก์ **IsAncestor**

2.3 IsEmpty

Returns TRUE ถ้าค่าของ Value Expression เป็น Empty ถ้าไม่ใช่ return FALSE

Syntax

IsEmpty(«Value Expression»)

ตัวอย่าง

```
WITH MEMBER [Measures].[Empty] AS 'IIF( IsEmpty([Measures].[Unit Sales]),
        "Yes","No" )'
SELECT {[Store].[Store Name].MEMBERS}  ON COLUMNS,
        { [Measures].[Store Sales],[Measures].[Empty] }ON ROWS
FROM Sales
```

	Store Sales	Empty
Store 19		Yes
Store 20		Yes
Store 9		Yes
Store 21		Yes
Store 1		Yes
Store 5		Yes
Store 10		Yes
Store 8		Yes
Store 4		Yes
Store 12		Yes
Store 18		Yes
HQ		Yes
Store 6	$45,750.24	No
Store 7	$54,545.28	No
Store 24	$54,431.14	No
Store 14	$4,441.18	No
Store 11	$55,058.79	No
Store 13	$87,218.28	No
Store 2	$4,739.23	No
Store 3	$52,896.30	No
Store 15	$52,644.07	No
Store 16	$49,634.46	No
Store 17	$74,843.96	No
Store 22	$4,705.97	No
Store 23	$24,329.23	No

รูปที่ ก-7 แสดงตัวอย่างการใช้ฟังชันก์ **IsEmpty**

2.4 IsGeneration

Returns TRUE ถ้า Member เป็น Generation ของค่าที่กำหนดใน Numeric Expression ถ้าไม่ใช่ return FALSE

Syntax

IsGeneration(«Member»,«Numeric Expression»)

2.5 IsLeaf

Returns TRUE ถ้า Member เป็น Leaf member ถ้าไม่ใช่ return FALSE

Syntax

IsLeaf(«Member»)

ตัวอย่าง

```
WITH  MEMBER [Measures].[Leaf] AS 'IIF (IsLeaf(  [Store].[Canada].[BC].[Vancouver].
            [Store 19]),"Yes","No" )'
SELECT {[Store].[Canada].[BC].[Vancouver].[Store 19]}  ON COLUMNS,
        { [Measures].[Leaf] }ON ROWS
FROM Sales
```

	Store 19
Leaf	Yes

รูปที่ ก-8 แสดงตัวอย่างการใช้ฟังชันก์ **IsLeaf**

2.6 IsSibling

Returns TRUE ถ้า Member เป็น Sibling member ถ้าไม่ใช่ return FALSE

Syntax

IsSibling(«Member1»,«Member2»)

```
WITH   MEMBER [Measures].[Leaf] AS 'IIF (IsSibling([Time].[1998],
Time].[1997]),"Yes","No" )'
SELECT {[Time].[1998]} ON COLUMNS,
        { [Measures].[Leaf] }ON ROWS
FROM Sales
```

	1998
Leaf	Yes

รูปที่ ก-9 แสดงตัวอย่างการใช้ฟังชันก์ IsSibling

3. Member Functions

3.1 Ancestor

Return ancestor ของ member ใน level ที่กำหนดใน «Level» หรือ«Numeric Expression»

Syntax

Level **Ancestor**(«Member», «Level»)

Distance **ncestor**(«Member», «Numeric Expression»)

3.2 ClosingPeriod

Return member ตัวสุดท้ายใน level ที่กำหนด

Syntax

ClosingPeriod([«Level»[, «Member»]])

```
SELECT {Measures.Members} ON COLUMNS,
        { ClosingPeriod([Product].[Product Department],[Product].[Food])}  ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Starchy Foo	5,262.00	4,705.91	฿11,756.07	1691	7,050.16

รูปที่ ก-10 แสดงตัวอย่างการใช้ฟังชันก์ ClosingPeriod

3.3 Cousin

Returns child member ของ Member2 ภายใต้ parent member เดียวกัน

Syntax

Cousin(«Member1», «Member2»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {Cousin([Time].[1998].[Q3],[Time].[1997])} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Q3	65,848.00	55,904.87	฿140,271.8	21453	84,367.02

รูปที่ ก-11 แสดงตัวอย่างการใช้ฟังชันก์ Cousin

3.4 CurrentMember

Returns current member

Syntax

«Dimension».**CurrentMember**

3.5 DataMember

Returns «Member» ถ้า «Member» เป็น leaf member

Syntax

«Member».**DataMember**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        { [Time].[1].DataMember,[Time].[1] } ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
1	21,628.00	18,178.49	฿45,539.69	7034	27,361.20
	21,628.00	18,178.49	฿45,539.69	7034	27,361.20

รูปที่ ก-12 แสดงตัวอย่างการใช้ฟังชันก์ DataMember

3.6 DefaultMember

Returns default member ของ dimension หรือ hierarchy ที่กำหนด

Syntax

Dimension «Dimension».**DefaultMember**

Hierarchy «Hierarchy».**DefaultMember**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        { [Time].DefaultMember } ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
1997	266,773.00	225,627.23	$565,238.13	86837	339,610.90

รูปที่ ก-13 แสดงตัวอย่างการใช้ฟังชันก์ DefaultMember

3.7 FirstChild

Returns member ที่เป็น first child ของ member ที่กำหนด

Syntax

«Member».**FirstChild**

ตัวอย่าง

```
SELECT  {Measures.Members} ON COLUMNS,
   {[1997].FIRSTCHILD} ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Q1	66,291.00	55,752.24	$139,628.35	21588	83,876.11

รูปที่ ก-14 แสดงตัวอย่างการใช้ฟังชันก์ FirstChild

3.8 FirstSibling

Returns member ที่เป็น first sibling ของ member ที่กำหนด

Syntax

«Member».**FirstSibling**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[USA].[OR].FirstSibling} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
CA	74,748.00	63,530.43	$159,167.84	24442	95,637.41

รูปที่ ก-15 แสดงตัวอย่างการใช้ฟังชันก์ FirstSibling

3.9 Item

Returns member จาก tuple ที่กำหนด หรือ returns tuple จาก set

Syntax

Member «Tuple».**Item**(«Index»)

Tuple «Set».**Item**(«String Expression»[, «String Expression»...] | «Index»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        { ([Time].[1997] ,[Store].[USA] ).Item(1)}  ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90

รูปที่ ก-16 แสดงตัวอย่างการใช้ฟังชันก์ Item

3.10 Lag

Return member ตัวที่ต้องการ ที่อยู่ก่อน member ที่กำหนด

Syntax

«Member».**Lag**(«Numeric Expression»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        { [Time].[1997].[Q3].lag(2)} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Q1	66,291.00	55,752.24	฿139,628.35	21588	83,876.11

รูปที่ ก-17 แสดงตัวอย่างการใช้ฟังชันก์ Lag

3.11 LastChild

Returns last child ของ member ที่กำหนด

Syntax

«Member».**LastChild**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[USA].LastChild}  ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91

รูปที่ ก-18 แสดงตัวอย่างการใช้ฟังชันก์ LastChild

3.12 LastSibling

Returns last sibling ของ member ที่กำหนด

Syntax

«Member».**LastSibling**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[USA].[OR].LastSibling} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
WA	124,366.00	105,324.31	$263,793.22	40784	158,468.91

รูปที่ ก-19 แสดงตัวอย่างการใช้ฟังชันก์ LastSibling

3.13 Lead

Return member ตัวใด ๆ ที่อยู่หลัง member ที่กำหนด

Syntax

«Member».**Lead**(«Numeric Expression»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        { [Time].[1997].[Q2].lead(2)} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Q4	72,024.00	61,005.90	$152,671.62	23428	91,665.72

รูปที่ ก-20 แสดงตัวอย่างการใช้ฟังชันก์ Lead

3.14 LinkMember

Returns member จาก member ที่กำหนด ใน hierarchy ที่ต้องการ

Syntax

LinkMember(«Member», «Hierarchy»)

ตัวอย่าง

```
SELECT  [Measures].MEMBERS ON COLUMNS,
        {LinkMember([Time].[Calendar].[1997], [Time].[Fiscal])} ON ROWS
FROM Sales
```

3.15 Members

Returns set ของ members ที่เป็นสมาชิกใน dimension, level หรือ hierarchy ที่ต้องการหรือ returns member ที่กำหนดโดย string expression.

Syntax

Dimension	«Dimension».**Members**
Hierarchy	«Hierarchy».**Members**
Level	«Level».**Members**
String	**Members**(«String Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        [Store].[Store Name].MEMBERS ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Store 19					
Store 20					
Store 9					
Store 21					
Store 1					
Store 5					
Store 10					
Store 8					
Store 1					
Store 12					
Store 18					
HQ					
Store 6	21,333.00	18,266.44	$45,750.24	6815	27,483.80
Store 7	25,663.00	21,771.54	$54,545.28	8207	32,773.74
Store 24	25,635.00	21,713.53	$54,431.14	8095	32,717.61
Store 14	2,117.00	1,778.92	$4,441.18	1325	2,662.26
Store 11	26,079.00	21,948.94	$55,058.79	8264	33,109.85
Store 13	41,580.00	34,623.56	$87,218.28	13347	52,394.72
Store 2	2,237.00	1,896.62	$4,739.23	1380	2,842.61
Store 3	24,576.00	21,121.96	$52,896.30	7876	31,774.34
Store 15	25,011.00	20,956.80	$52,644.07	7956	31,687.27
Store 16	23,591.00	19,795.49	$49,634.46	7397	29,838.97
Store 17	35,257.00	29,959.28	$74,843.96	11184	44,884.68
Store 22	2,203.00	1,880.34	$4,705.97	1339	2,825.63
Store 23	11,491.00	9,713.81	$24,329.23	3652	14,615.42

รูปที่ ก-21 แสดงตัวอย่างการใช้ฟังชันก์ Members

3.16 NextMember

Returns member ตัวถัดไปใน level เดียวกันของ member ที่กำหนด

Syntax

«Member».**NextMember**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[USA].[CA].nextmember} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
OR	67,659.00	56,772.50	$142,277.07	21611	85,504.57

รูปที่ ก-22 แสดงตัวอย่างการใช้ฟังชันก์ NextMember

3.17 OpeningPeriod

Return member ตัวแรกใน level ที่กำหนด

Syntax

OpeningPeriod([«Level»[, «Member»]])

3.18 ParallelPeriod

Returns member จากช่วงเวลาเดียวกันกับช่วงเวลาที่กำหนดของ member ที่ต้องการ

Syntax

ParallelPeriod([«Level»[, «Numeric Expression»[, «Member»]]])

3.19 Parent

Returns parent ของ member

Syntax

«Member».**Parent**

ตัวอย่าง

```
SELECT {[Measures].MEMBERS} ON COLUMNS,
        {[CA].PARENT} ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90

รูปที่ ก-23 แสดงตัวอย่างการใช้ฟังชันก์ **Parent**

3.20 PrevMember

Returns member ตัวก่อนหน้า member ที่กำหนดภายใต้ levelเดียวกัน

Syntax

«Member».**PrevMember**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {[Store].[USA].[OR].PrevMember} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
CA	74,748.00	63,530.43	฿159,167.84	24442	95,637.41

รูปที่ ก-24 แสดงตัวอย่างการใช้ฟังชันก์ PrevMember

3.21 StrToMember

Returns member จาก string expression

Syntax

StrToMember(«String Expression»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        {StrToMember(" [Store].[USA].[OR]")} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57

รูปที่ ก-25 แสดงตัวอย่างการใช้ฟังชันก์ StrToMember

3.22 ValidMeasure

Syntax

ValidMeasure(«Tuple»)

4. Numeric Functions

4.1 Aggregate

Syntax

Aggregate(«Set»[, «Numeric Expression»])

4.2 Avg

Returns ค่า average ของ numeric expression ภายใต้ set ที่ต้องการ

Syntax

Avg(«Set»[, «Numeric Expression»])

ตัวอย่าง

```
WITH MEMBER [Time].[Average Sales] AS 'AVG(DESCENDANTS([Time].[1997],
[Time].[Month]))'
MEMBER [Time].[Average Count] AS 'COUNT(DESCENDANTS([Time].[1997],
[Time].[Month]),EXCLUDEEMPTY)'
SELECT {[Time].[1997], [Time].[Average Sales], [Time].[Average Count]} ON COLUMNS,
[Product].[Brand Name].MEMBERS ON ROWS
FROM [Sales]
WHERE (Measures.[Unit Sales])
```

	1997	Average Sal	Average Co
Good	269.00	22.42	12
Pearl	385.00	32.08	12
Portsmouth	362.00	30.17	12
Top Measure	306.00	25.50	12
Walrus	361.00	30.08	12
Good	945.00	78.75	12
Pearl	1,206.00	100.50	12
Portsmouth	1,015.00	84.58	12
Top Measure	951.00	79.25	12
Walrus	1,038.00	86.50	12
Excellent	738.00	61.50	12
Fabulous	632.00	52.67	12
Skinner	655.00	54.58	12
Token	735.00	61.25	12
Washington	647.00	53.92	12

รูปที่ ก-26 แสดงตัวอย่างการใช้ฟังชันก์ Avg

4.3 CalculationCurrentPass

Syntax

CalculationCurrentPass()

4.4 CalculationPassValue

Syntax

CalculationPassValue(«Numeric Expression», «Pass Value»[, «Access Flag»])

4.5 CoalesceEmpty

Returns the coalesced value.

Syntax

Numeric

CoalesceEmpty(«Numeric Expression»[, «Numeric Expression»]...)

String

CoalesceEmpty(«String Expression»[, «String Expression»]...)

ตัวอย่าง

```
WITH MEMBER Measures.[Profit Growth] AS '(Measures.[Profit]) /
COALESCEEMPTY((Measures.[Profit], [Time].PREVMEMBER), Measures.[Profit])',
 FORMAT_STRING = '#.00%'
SELECT {Measures.[Profit], Measures.[Profit Growth]} ON COLUMNS,
{DESCENDANTS([Time].[1997], [Month])} ON ROWS
FROM [Sales]
```

4.11 LinRegIntercept

Syntax

LinRegIntercept(«Set», «Numeric Expression»[, «Numeric Expression»])

4.12 LinRegPoint

Syntax

LinRegPoint(«Numeric Expression», «Set», «Numeric Expression»[,

«Numeric Expression»])

4.13 LinRegR2

Syntax

LinRegR2(«Set», «Numeric Expression»[, «Numeric Expression»])

4.14 LinRegSlope

Syntax

LinRegSlope(«Set», «Numeric Expression»[, «Numeric Expression»])

4.15 LinRegVariance

Syntax

LinRegVariance(«Set», «Numeric Expression»[, «Numeric Expression»])

4.16 LookupCube

Returns ค่าของ Multidimensional Expressions (MDX) ใน cube ที่ต้องการภายใต้ database เดียวกัน

Syntax

Numeric

LookupCube(«Cube String», «Numeric Expression»)

String

LookupCube(«Cube String», «String Expression»)

ตัวอย่าง

```
WITH MEMBER Measures.[Store Unit Sales] AS
    'LookupCube("Sales","(" + MemberToStr(Store.CurrentMember) + ", Measures.[Unit
Sales])")'
SELECT {Measures.Amount, Measures.[Store Unit Sales]} ON COLUMNS,
        Store.CA.CHILDREN ON ROWS
FROM Budget
```

4.17 Max

Returns the maximum value ของ numeric expression จาก set ที่กำหนด

Syntax

Max(«Set»[, «Numeric Expression»])

ตัวอย่าง

```
WITH MEMBER Measures.[Maximum Sales] AS
'MAX(DESCENDANTS([Time].CURRENTMEMBER, [Time].[Month]),
Measures.[Unit Sales])'
SELECT {[Time].[1997]} ON COLUMNS,
[Product].[Product Category].MEMBERS ON ROWS
FROM [Sales]
WHERE (Measures.[Maximum Sales])
```

4.18 Median

Returns the median value ของ a numeric expression จาก set ที่กำหนด

Syntax

Median(«Set»[, «Numeric Expression»])

ตัวอย่าง

```
WITH MEMBER Measures.[Maximum Sales] AS
'Median(DESCENDANTS([Time].CURRENTMEMBER, [Time].[Month]),
 Measures.[Unit Sales])'
SELECT {[Time].[1997]} ON COLUMNS,
[Product].[Product Category].MEMBERS ON ROWS
FROM [Sales]
WHERE (Measures.[Maximum Sales])
```

4.19 Min

Returns the minimum value ของ numeric expression จาก set ที่กำหนด

Syntax

Min(«Set»[, «Numeric Expression»])

ตัวอย่าง

```
WITH MEMBER Measures.[Maximum Sales] AS
'MIN(DESCENDANTS([Time].CURRENTMEMBER, [Time].[Month]), Measures.[Unit
Sales])'
SELECT {[Time].[1997]} ON COLUMNS,
```

[Product].[Product Category].MEMBERS ON ROWS

FROM [Sales]

WHERE (Measures.[Maximum Sales])

4.20 Ordinal

Syntax

«Level».**Ordinal**

4.21 Predict

Syntax

Predict(«Mining Model Name», «Numeric Expression»)

4.22 Rank

Syntax

Rank(«Tuple», «Set»[, «Calc Expression»])

4.23 RollupChildren

Syntax

RollupChildren(«Member», «String Expression»)

4.24 Stddev

Alias for **Stdev**

4.25 StddevP

Alias for **StdevP**

4.26 Stdev

Syntax

Stdev(«Set»[, «Numeric Expression»])

4.27 StdevP

Syntax

StdevP(«Set»[, «Numeric Expression»])

4.28 StrToValue

Syntax

StrToValue(«String Expression»)

4.29 Sum

Returns ผล sum ของ numeric expression

Syntax

Sum(«Set»[, «Numeric Expression»])

ตัวอย่าง

```
WITH MEMBER Measures.SaleCA_WA AS
        'SUM({[Store].[Store State].[CA], [Store].[Store State].[WA]},Measures.[Unit
        Sales])', FORMAT_STRING = '#.00'
SELECT {DESCENDANTS([Time].[1997], [Month])} ON COLUMNS,
        {[Product].[Product Family].MEMBERS} ON ROWS
FROM [Sales]
WHERE (Measures.SaleCA_WA)
```

4.30 Value

Syntax

«Member».**Value**

4.31 Var

Syntax

Var(«Set»[, «Numeric Expression»])

4.32 Variance

Alias for **Var**

4.33 VarianceP

Alias for **VarP**.

4.44 VarP

Syntax

VarP(«Set»[, «Numeric Expression»])

5. Set Functions

5.1 AddCalculatedMembers

Return members ทั้งหมดที่เป็นสมาชิกของ set ตามที่ระบุใน («Set») โดย return calculated members ด้วย (โดยปกติจะไม่แสดง) ใช้ในกรณีที่อ้างอิงถึง Set

Syntax

AddCalculatedMembers(«Set»)

ตัวอย่าง

```
SELECT ADDCALCULATEDMEMBERS(Measures.MEMBERS) ON COLUMNS,
        {[Store].[Store State].MEMBERS} ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I	Profit	Sales Average
BC							
DF							
Guerrero							
Jalisco							
Veracruz							
Yucatan							
Zacatecas							
CA	74,748.00	63,530.43	$159,167.8	24442	95,637.41	95,637.41	6.51
OR	67,659.00	56,772.50	$142,277.0	21611	85,504.57	85,504.57	6.58
WA	124,366.00	105,324.31	$263,793.2	40784	158,468.91	158,468.91	6.47

รูปที่ ก-27 แสดงตัวอย่างการใช้ฟังชันก์ AddCalculatedMembers

5.2 AllMembers

Return members ทั้งหมดของ Dimension หรือ Level ที่กำหนด โดย return calculated members ด้วย (โดยปกติจะไม่แสดง) ใช้ในกรณีที่อ้างอิงถึง Dimension หรือ Level

Syntax

Dimension «Dimension».**AllMembers**

Level «Level».**AllMembers**

ตัวอย่าง

```
SELECT Measures.AllMEMBERS ON COLUMNS,
  [Store].[Store State].Members ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I	Profit	Sales Average
BC							
DF							
Guerrero							
Jalisco							
Veracruz							
Yucatan							
Zacatecas							
CA	74,748.00	63,530.43	$159,167.8	24442	95,637.41	95,637.41	6.51
OR	67,659.00	56,772.50	$142,277.0	21611	85,504.57	85,504.57	6.58
WA	124,366.00	105,324.31	$263,793.2	40784	158,468.91	158,468.91	6.47

รูปที่ ก-28 แสดงตัวอย่างการใช้ฟังชันก์ AllMembers

5.3 Ancestors

Return ancestor ใน level ที่ต้องการของ member ที่กำหนด

Syntax

Level **Ancestors**(«Member», «Level»)

Distance **Ancestors**(«Member», «Numeric Expression»)

ตัวอย่าง

```
SELECT {[Measures].MEMBERS} ON COLUMNS,
        ANCESTORS ([Store].[USA].[CA].[Los Angeles],[Store State]) ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41

รูปที่ ก-29 แสดงตัวอย่างการใช้ฟังชันก์ Ancestors

5.4 Ascendants

Return ascendants ของ member ที่กำหนด

Syntax

Ascendants(«Member»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        {Ascendants([Los Angeles])} ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Los Angeles	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41
USA	266,773.00	225,627.23	฿565,238.1	86837	339,610.90
All Stores	266,773.00	225,627.23	฿565,238.1	86837	339,610.90

รูปที่ ก-30 แสดงตัวอย่างการใช้ฟังชันก์ **Ascendants**

5.5 Axis

Return set ตามที่กำหนดใน axis

Syntax

Axis(«Numeric Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON AXIS(0),
        {Ascendants([Los Angeles])} ON AXIS(1)
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Los Angeles	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41
USA	266,773.00	225,627.23	฿565,238.1	86837	339,610.90
All Stores	266,773.00	225,627.23	฿565,238.1	86837	339,610.90

รูปที่ ก-31 แสดงตัวอย่างการใช้ฟังชันก์ **Axis**

5.6 BottomCount

Return member ตามจำนวนที่กำหนด โดยทำการเรียง member จากน้อยไปมากก่อน

Syntax

BottomCount(«Set», «Count»[, «Numeric Expression»])

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        BottomCount({[Store].[Store State].MEMBERS}, 8, [Measures].[Unit Sales]) ON
ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Zacatecas					
Yucatan					
Veracruz					
Jalisco					
Guerrero					
DF					
BC					
OR	67,659.00	56,772.50	฿142,277.0	21611	85,504.57

รูปที่ ก-32 แสดงตัวอย่างการใช้ฟังชันก์ BottomCount

5.7 BottomPercent

Return member ตามจำนวนที่กำหนดเป็นเปอร์เซ็นต์ โดยทำการเรียง member จากน้อยไปมากก่อน

Syntax

BottomPercent(«Set», «Percentage», «Numeric Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        BottomPERCENT({[Store].[Store State].MEMBERS}, 10, [Measures].[Unit Sales])
        ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Zacatecas					
Yucatan					
Veracruz					
Jalisco					
Guerrero					
DF					
BC					
OR	67,659.00	56,772.50	฿142,277.0	21611	85,504.57
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41

รูปที่ ก-33 แสดงตัวอย่างการใช้ฟังชันก์ BottomPercent

5.8 BottomSum

Return member ที่มีค่าน้อยกว่าค่าที่กำหนด

Syntax

BottomSum(«Set», «Value», «Numeric Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        BottomSum({[Store].[Store State].MEMBERS}, 100000, Measures.[Unit Sales])
        ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Zacatecas					
Yucatan					
Veracruz					
Jalisco					
Guerrero					
DF					
BC					
OR	67,659.00	56,772.50	$142,277.0	21611	85,504.57
CA	74,748.00	63,530.43	$159,167.8	24442	95,637.41

รูปที่ ก-34 แสดงตัวอย่างการใช้ฟังชันก์ BottomSum

5.9 Children

Return children ของ member ที่กำหนด

Syntax

«Member».**Children**

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
        {[Store].[Store State].[CA].CHILDREN} ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Alameda					
Beverly Hills	21,333.00	18,266.44	$45,750.24	6815	27,483.80
Los Angeles	25,663.00	21,771.54	$54,545.28	8207	32,773.74
San Diego	25,635.00	21,713.53	$54,431.14	8095	32,717.61
San Francisc	2,117.00	1,778.92	$4,441.18	1325	2,662.26

รูปที่ ก-35 แสดงตัวอย่างการใช้ฟังชันก์ Children

5.10 Crossjoin

Return cross product ของ 2 sets

Syntax

Crossjoin(«Set1», «Set2»)

ตัวอย่าง

```
SELECT [Product].[Product Family].MEMBERS ON COLUMNS,
 CROSSJOIN ( [Customers].[Country].MEMBERS ,
[Time].[Year].MEMBERS) } ON ROWS
FROM Sales
WHERE ([Measures].[Unit Sales])
```

		Drink	Food	Non-Consumable
Canada	1997			
	1998			
Mexico	1997			
	1998			
USA	1997	24,597.00	191,940.00	50,236.00
	1998			

รูปที่ ก-36 แสดงตัวอย่างการใช้ฟังชันก์ Crossjoin

5.11 Descendants

Return member ที่เป็น descendants ของ member ที่กำหนด ใน level ที่ต้องการ

Syntax

Level **Descendants**(«Member», [«Level»[, «Desc_flags»]])

Distance **Descendants**(«Member», «Distance»[, «Desc_flags»])

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
    { DESCENDANTS([Store].[Store State].[CA],1)} ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Alameda					
Beverly Hills	21,333.00	18,266.44	$45,750.24	6815	27,483.80
Los Angeles	25,663.00	21,771.54	$54,545.28	8207	32,773.74
San Diego	25,635.00	21,713.53	$54,431.14	8095	32,717.61
San Francisc	2,117.00	1,778.92	$4,441.18	1325	2,662.26

รูปที่ ก-37 แสดงตัวอย่างการใช้ฟังชันก์ Descendants

5.12 Distinct

Return set โดย remove tuple ที่ซ้ำกันออก

Syntax

Distinct(«Set»)

ตัวอย่าง

```
SELECT [Measures].Members ON COLUMNS,
 Distinct({ [Store].[USA].[CA],[Store].[USA].[OR],
[Store].[USA].[WA],[Store].[USA].[CA] }) ON ROWS
```

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41
OR	67,659.00	56,772.50	฿142,277.0	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.2	40784	158,468.91

รูปที่ ก-38 แสดงตัวอย่างการใช้ฟังชันก์ Distinct

5.13 DrilldownLevel

Drilldown member ที่กำหนดลง 1 level

Syntax

DrilldownLevel(?Set?[, ?Level?])

ตัวอย่าง

SELECT [Measures].MEMBERS ON COLUMNS,

DrillDownLevel({[Store].[USA] },[Store Country]) ON ROWS

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.1	86837	339,610.90
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41
OR	67,659.00	56,772.50	฿142,277.0	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.2	40784	158,468.91

รูปที่ ก-39 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownLevel

5.14 DrilldownLevelBottom

Drilldown member ที่กำหนดลง 1 level, ทำการเรียงจากน้อยไปมาก แล้วแสดงเฉพาะจำนวนที่กำหนด

Syntax

DrilldownLevelBottom(«Set», «Count»[, [«Level»][, «Numeric Expression»]])

ตัวอย่าง

SELECT [Measures].Members ON COLUMNS,

DrilldownLevelBottom({ [Store].[USA] },2,,[Unit Sales]) ON ROWS

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.1	86837	339,610.90
WA	124,366.00	105,324.31	฿263,793.2	40784	158,468.91
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41

รูปที่ ก-40 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownLevelBottom

5.15 DrilldownLevelTop

Drilldown member ที่กำหนดลง 1 level, ทำการเรียงจากมากไปน้อย แล้วแสดงเฉพาะจำนวนที่กำหนด

Syntax

DrilldownLevelTop(«Set», «Count»[, [«Level»][, «Numeric Expression»]])

ตัวอย่าง

```
SELECT [Measures].Members ON COLUMNS,
    DrilldownLevelTop({ [Store].[USA] },2,,[Unit Sales]) ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.1:	86837	339,610.90
WA	124,366.00	105,324.31	฿263,793.2:	40784	158,468.91
CA	74,748.00	63,530.43	฿159,167.8•	24442	95,637.41

รูปที่ ก-41 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownLevelTop

5.16 DrilldownMember

Drills down members ใน set ที่กำหนด หรือใช้ drills down set ของ tuples

Syntax

DrilldownMember(«Set1», «Set2»[, RECURSIVE])

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
    DrillDownMember( {[Store].[USA],[Store].[Canada],[Store].[Mexico] },
                     {[Store].[USA],[Store].[Vancouver],[Store].[Mexico] } ) ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90
CA	74,748.00	63,530.43	฿159,167.84	24442	95,637.41
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91
Canada					
Mexico					
DF					
Guerrero					
Jalisco					
Veracruz					
Yucatan					
Zacatecas					

รูปที่ ก-42 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownMember

5.17 DrilldownMemberBottom

Drills down members ใน set ที่กำหนด, ทำการเรียงจากน้อยไปมากแล้วแสดงตามจำนวนที่กำหนด

Syntax

DrilldownMemberBottom(«Set1», «Set2», «Count»[, [«Numeric Expression»]

[, RECURSIVE]])

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
DrilldownMemberBottom( {[Store].[USA],[Store].[Canada],[Store].[Mexico] },
{[Store].[USA],[Store].[Vancouver],[Store].[Mexico] },2,[Measures].[Unit Sales] ) ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57
CA	74,748.00	63,530.43	฿159,167.84	24442	95,637.41
Canada					
Mexico					
Zacatecas					
Yucatan					

รูปที่ ก-43 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownMemberBottom

5.18 DrilldownMemberTop

Drills down members ใน set ที่กำหนด, ทำการเรียงจากมากไปน้อยแล้วแสดงตามจำนวนที่กำหนด

Syntax

DrilldownMemberTop(«Set1», «Set2», «Count»[, [«Numeric Expression»]
[, RECURSIVE]])

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
DrilldownMemberTop( {[Store].[USA],[Store].[Canada],[Store].[Mexico] },
{[Store].[USA],[Store].[Vancouver],[Store].[Mexico] },2,[Measures].[Unit Sales] ) ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Ne
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91
BC					

รูปที่ ก-44 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownMemberTop

5.19 DrillupLevel

Drills up members จาก set ที่กำหนดไปยัง level ที่ต้องการ

Syntax

DrillupLevel(«Set»[, «Level»])

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
```

DrillUpLevel({[Store].[CA],[USA] },[Store Country]) ON ROWS

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
USA	266,773.00	225,627.23	฿565,238.1:	86837	339,610.90

รูปที่ ก-45 แสดงตัวอย่างการใช้ฟังชันก์ DrilldownMember

5.20 DrillupMember

Drills up members จาก set ที่กำหนดไปยัง level ที่กำหนดกับอีก set

Syntax

DrillupMember(«Set1», «Set2»)

ตัวอย่าง

SELECT Measures.MEMBERS ON COLUMNS,

DrillUpMember({WA,[OR],WA,BC},{USA}) ON ROWS

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91
BC					

รูปที่ ก-46 แสดงตัวอย่างการใช้ฟังชันก์ DrillupMember

5.21 Except

หาความแตกต่างของ 2 set

Syntax

Except(«Set1», «Set2»[, ALL])

ตัวอย่าง

SELECT [Measures].MEMBERS ON COLUMNS,

Except({[Canada],[Mexico],[USA]},{[Canada]}) ON ROWS

FROM Sales

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Ne
Mexico					
USA	266,773.00	225,627.23	฿565,238.13	86837	339,610.90

รูปที่ ก-47 แสดงตัวอย่างการใช้ฟังชันก์ Except

5.22 Extract

Returns a set จาก tuples เฉพาะ dimension ที่ต้องการ

Syntax

Extract(«Set», «Dimension»[, «Dimension»...])

ตัวอย่าง

```
SELECT { [Measures].MEMBERS} ON COLUMNS,
        Extract({ ([1997], WA),([1997],[CA])}, Store)  ON ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
WA	124,366.00	105,324.31	$263,793.22	40784	158,468.91
CA	74,748.00	63,530.43	$159,167.84	24442	95,637.41

รูปที่ ก-48 แสดงตัวอย่างการใช้ฟังชันก์ Extract

5.23 Filter

Returns set ที่ได้จากการ filter set ที่กำหนดตาม search condition

Syntax

Filter(«Set», «Search Condition»)

ตัวอย่าง

```
SELECT [Measures].MEMBERS ON COLUMNS,
        FILTER( {[Store].[Store City].MEMBERS},([Measures].[Unit Sales],
                [Time].[1997]) < 10000) ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Vancouver					
Victoria					
Mexico City					
San Andres					
Acapulco					
Guadalajara					
Orizaba					
Merida					
Camacho					
Hidalgo					
Alameda					
San Francisc	2,117.00	1,778.92	$4,441.18	1325	2,662.26
Bellingham	2,237.00	1,896.62	$4,739.23	1380	2,842.61
Walla Walla	2,203.00	1,880.34	$4,705.97	1339	2,825.63

รูปที่ ก-49 แสดงตัวอย่างการใช้ฟังชันก์ **Filter**

5.24 Generate

แสดง set ของ each member

Syntax

Set **Generate**(«Set1», «Set2»[, ALL])

String **Generate**(«Set», «String Expression»[, «Delimiter»])

ตัวอย่าง

```
SELECT GENERATE([Time].[Quarter].MEMBERS,
        {[Time].CurrentMember.FIRSTCHILD}) ON COLUMNS,
```

Non Empty [Store].[Store Name].MEMBERS ON ROWS

FROM [Sales]

	1	4	7	10	1	4	7	10
Store 6	1,587.41	3,041.48	1,670.48	2,870.42				
Store 7	2,501.14	2,341.79	2,517.29	2,676.10				
Store 24	2,843.79	2,582.96	2,612.95	2,075.95				
Store 14	133.98	194.32	197.75	227.19				
Store 11	2,649.01	2,285.92	2,348.61	2,451.48				
Store 13	5,936.51	2,734.00	7,426.10	2,948.55				
Store 2	228.83	236.31	213.52	279.81				
Store 3	2,054.28	2,546.57	2,470.84	2,040.68				
Store 15	2,415.73	2,660.57	2,779.09	2,637.50				
Store 16	1,962.18	2,469.23	2,733.38	2,199.80				
Store 17	3,651.69	3,241.93	3,985.99	3,725.12				
Store 22	172.24	215.06	259.00	190.55				
Store 23	1,224.39	1,216.40	1,026.09	1,117.10				

รูปที่ ก-50 แสดงตัวอย่างการใช้ฟังชันก์ Generate

5.25 Head

Returns members แรกตามจำนวนที่กำหนดของสมาชิกใน set

Syntax

Head(«Set»[, «Numeric Expression»])

ตัวอย่าง

SELECT Measures.MEMBERS ON COLUMNS,

HEAD({[Store].[Store City].MEMBERS}, 13) ON ROWS

FROM [Sales]

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales
Vancouver					
Victoria					
Mexico City					
San Andres					
Acapulco					
Guadalajara					
Orizaba					
Merida					
Camacho					
Hidalgo					
Alameda					
Beverly Hills	21,333.00	18,266.44	$45,750.24	6815	27,483.80
Los Angeles	25,663.00	21,771.54	$54,545.28	8207	32,773.74

รูปที่ ก-51 แสดงตัวอย่างการใช้ฟังชันก์ Head

5.26 Hierarchize

Return members จาก set ตามลำดับใน hierarchy

Syntax

Hierarchize(«Set»[, POST])

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
    Hierarchize( [Time].Members,POST) ON ROWS
FROM  [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
1	21,628.00	18,178.49	฿45,539.69	7034	27,361.20
2	20,957.00	17,599.67	฿44,058.79	6844	26,459.12
3	23,706.00	19,974.08	฿50,029.87	7710	30,055.79
Q1	66,291.00	55,752.24	฿139,628.35	21588	83,876.11
4	20,179.00	17,111.70	฿42,878.25	6590	25,766.55
5	21,081.00	17,782.56	฿44,456.29	6866	26,673.73
6	21,350.00	18,069.97	฿45,331.73	6912	27,261.76
Q2	62,610.00	52,964.22	฿132,666.27	20368	79,702.05
7	23,763.00	20,005.79	฿50,246.88	7752	30,241.09
8	21,697.00	18,435.96	฿46,199.04	7038	27,763.08
9	20,388.00	17,463.12	฿43,825.97	6663	26,362.85
Q3	65,848.00	55,904.87	฿140,271.89	21453	84,367.02
10	19,958.00	16,902.01	฿42,342.27	6479	25,440.26
11	25,270.00	21,357.76	฿53,363.71	8232	32,005.95
12	26,796.00	22,746.13	฿56,965.64	8717	34,219.51
Q4	72,024.00	61,005.90	฿152,671.62	23428	91,665.72
1997	266,773.00	225,627.23	฿565,238.13	86837	339,610.90
1					
2					
3					
Q1					
4					
5					
6					
Q2					
7					
8					
9					
Q3					
10					
11					
12					
Q4					
1998					

รูปที่ ก-52 แสดงตัวอย่างการใช้ฟังชันก์ Hierarchize

5.27 Intersect

Returns intersection ของ 2 sets

Syntax

Intersect(«Set1», «Set2»[, ALL])

ตัวอย่าง

```
Select [Measures].Members  on Columns,
Intersect( {[Time].[1997].[Q1],[Time].[1998].[Q3]},
        {[Time].[1997].[Q1],[Time].[1997].[Q2]} ) on Rows
from Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Q1	66,291.00	55,752.24	$139,628.3	21588	83,876.11

รูปที่ ก-53 แสดงตัวอย่างการใช้ฟังชันก์ Intersect

5.28 LastPeriods

Returns set ของ members ในลำดับก่อนหน้าหรือหลัง members นั้น รวมทั้ง members ที่กำหนดด้วย

Syntax

LastPeriods(«Index»[, «Member»])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
    LastPeriods(3,[Store].[OR])  ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Zacatecas					
CA	74,748.00	63,530.43	$159,167.8	24442	95,637.41
OR	67,659.00	56,772.50	$142,277.0	21611	85,504.57

รูปที่ ก-54 แสดงตัวอย่างการใช้ฟังชันก์ LastPeriods

5.29 Members

Returns set ของ members ที่เป็นสมาชิกใน dimension, level หรือ hierarchy ที่ต้องการหรือ returns member ที่กำหนดโดย string expression

Syntax

Dimension	«Dimension».**Members**
Hierarchy	«Hierarchy».**Members**
Level	«Level».**Members**
String	**Members**(«String Expression»)

5.30 Mtd

Returns set ของ members จาก level Mount ใน Time dimension โดยเริ่มจากค่าแรกจนถึงค่าที่กำหนด

Syntax

Mtd([«Member»])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        Mtd( [8]) ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
8	21,697.00	18,435.96	฿46,199.04	7038	27,763.08

รูปที่ ก-55 แสดงตัวอย่างการใช้ฟังชันก์ Generate

5.31 NameToSet

Returns set ของ single member จาก string expression ที่เป็น member name

Syntax

NameToSet(«Member Name»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        NameToSet("[Store].[USA].[WA].[Bellingham].[Store 2]")   ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
CA	74,748.00	63,530.43	฿159,167.84	24442	95,637.41

รูปที่ ก-56 แสดงตัวอย่างการใช้ฟังชันก์ NameToSet

5.32 NonEmptyCrossjoin

Returns cross product ของ 2 sets หรือมากกว่า โดยไม่รวม empty tuples

Syntax

NonEmptyCrossjoin(«Set1», «Set2»[, «Set3»...][, «Crossjoin Set Count»])

ตัวอย่าง

```
Select {Measures.Members} ON COLUMNS,
        NonEmptyCrossJoin ([Store Type].Children, {[Product].[Product
        Family].MEMBERS}) ON ROWS
FROM Sales
```

		Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Deluxe Supe	Drink	6,827.00	5,368.11	฿13,487.16	2186	8,119.05
	Food	55,358.00	46,812.76	฿117,088.8:	17655	70,276.11
	Non-Consum	14,652.00	12,601.97	฿31,486.21	4690	18,884.24
Gourmet Sup	Drink	1,945.00	1,547.71	฿3,940.54	618	2,392.83
	Food	15,438.00	13,397.99	฿33,424.17	4923	20,026.18
	Non-Consum	3,950.00	3,320.74	฿8,385.53	1274	5,064.79
Mid-Size Gro	Drink	1,159.00	939.29	฿2,348.79	366	1,409.50
	Food	8,192.00	6,922.05	฿17,314.24	2605	10,392.19
	Non-Consum	2,140.00	1,852.47	฿4,666.20	681	2,813.73
Small Grocer	Drink	574.00	456.72	฿1,142.61	360	685.89
	Food	4,764.00	4,065.58	฿10,175.30	2925	6,109.72
	Non-Consum	1,219.00	1,033.57	฿2,568.47	759	1,534.90
Supermarket	Drink	14,092.00	11,165.40	฿27,917.11	4448	16,751.71
	Food	108,188.00	92,072.34	฿231,033.0:	34337	138,960.67
	Non-Consum	28,275.00	24,070.52	฿60,259.92	9010	36,189.40

รูปที่ ก-57 แสดงตัวอย่างการใช้ฟังชันก์ NonEmptyCrossjoin

5.33 Order

เรียงลำดับ members ตามที่กำหนด

Syntax

Order(«Set», {«String Expression» | «Numeric Expression»}

[, ASC | DESC | BASC | BDESC])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
    ORDER ({[Store].[Store City].MEMBERS}, [Measures].[Unit Sales], BASC) ON
    ROWS
FROM Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Yancouver					
Victoria					
Mexico City					
San Andres					
Acapulco					
Guadalajara					
Orizaba					
Merida					
Camacho					
Hidalgo					
Alameda					
San Francisc	2,117.00	1,778.92	$4,441.18	1325	2,662.26
Walla Walla	2,203.00	1,880.34	$4,705.97	1339	2,825.63
Bellingham	2,237.00	1,896.62	$4,739.23	1380	2,842.61
Yakima	11,491.00	9,713.81	$24,329.23	3652	14,615.42
Beverly Hills	21,333.00	18,266.44	$45,750.24	6815	27,483.80
Spokane	23,591.00	19,795.49	$49,634.46	7397	29,838.97
Bremerton	24,576.00	21,121.96	$52,896.30	7876	31,774.34
Seattle	25,011.00	20,956.80	$52,644.07	7956	31,687.27
San Diego	25,635.00	21,713.53	$54,431.14	8095	32,717.61
Los Angeles	25,663.00	21,771.54	$54,545.28	8207	32,773.74
Portland	26,079.00	21,948.94	$55,058.79	8264	33,109.85
Tacoma	35,257.00	29,959.28	$74,843.96	11184	44,884.68
Salem	41,580.00	34,823.56	$87,218.28	13347	52,394.72

รูปที่ ก-58 แสดงตัวอย่างการใช้ฟังชันก์ Order

5.34 PeriodsToDate

Returns set ของ periods members จาก level ที่กำหนด เริ่มจากตัวแรกถึงตัวที่กำหนด

Syntax

PeriodsToDate([«Level»[, «Member»]])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        PeriodsToDate(Quarter,[8])  ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
7	23,763.00	20,005.79	$50,246.88	7752	30,241.09
8	21,697.00	18,435.96	$46,199.04	7038	27,763.08

รูปที่ ก-59 แสดงตัวอย่างการใช้ฟังชันก์ PeriodsToDate

5.35 Qtd

Returns set ของ members จาก level Quarter ใน Time dimension โดยเริ่มจากค่าแรกจนถึงค่าที่กำหนด

Syntax

Qtd([«Member»])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        QTD( [5]) ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
4	20,179.00	17,111.70	฿42,878.25	6590	25,766.55
5	21,081.00	17,782.56	฿44,456.29	6866	26,673.73

รูปที่ ก-60 แสดงตัวอย่างการใช้ฟังชันก์ Qtd

5.36 Siblings

Returns siblings ของ member ที่กำหนดรวมทั้ง member นั้นด้วย

Syntax

«Member».**Siblings**

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
    { [Time].[5].Siblings}   ON ROWS
FROM   [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
4	20,179.00	17,111.70	฿42,878.25	6590	25,766.55
5	21,081.00	17,782.56	฿44,456.29	6866	26,673.73
6	21,350.00	18,069.97	฿45,331.73	6912	27,261.76

รูปที่ ก-61 แสดงตัวอย่างการใช้ฟังชันก์ Siblings

5.37 StripCalculatedMembers

Returns set ที่สร้างโดย remove calculated members ออกจาก set ที่กำหนด

Syntax

StripCalculatedMembers(«Set»)

ตัวอย่าง

```
SELECT StripCalculatedMembers(ADDCALCULATEDMEMBERS(Measures.MEMBERS))
        ON COLUMNS,
        {[Store].[Store State].MEMBERS} ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
BC					
DF					
Guerrero					
Jalisco					
Veracruz					
Yucatan					
Zacatecas					
CA	74,748.00	63,530.43	฿159,167.8	24442	95,637.41
OR	67,659.00	56,772.50	฿142,277.0	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.2	40784	158,468.91

รูปที่ ก-62 แสดงตัวอย่างการใช้ฟังชันก์ StripCalculatedMembers

5.38 StrToSet

สร้าง set จาก string ที่กำหนด

Syntax

StrToSet(«String Expression»)

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
        StrToSet("Time.Members")  ON ROWS
FROM  [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
1997	266,773.00	225,627.23	฿565,238.1	86837	339,610.90
Q1	66,291.00	55,752.24	฿139,628.3	21588	83,876.11
1	21,628.00	18,178.49	฿45,539.69	7034	27,361.20
2	20,957.00	17,599.67	฿44,058.79	6844	26,459.12
3	23,706.00	19,974.08	฿50,029.87	7710	30,055.79
Q2	62,610.00	52,964.22	฿132,666.2	20368	79,702.05
4	20,179.00	17,111.70	฿42,878.25	6590	25,766.55
5	21,081.00	17,782.56	฿44,456.29	6866	26,673.73
6	21,350.00	18,069.97	฿45,331.73	6912	27,261.76
Q3	65,848.00	55,904.87	฿140,271.8	21453	84,367.02
7	23,763.00	20,005.79	฿50,246.88	7752	30,241.09
8	21,697.00	18,435.96	฿46,199.04	7038	27,763.08
9	20,388.00	17,463.12	฿43,825.97	6663	26,362.85
Q4	72,024.00	61,005.90	฿152,671.6	23428	91,665.72
10	19,958.00	16,902.01	฿42,342.27	6479	25,440.26
11	25,270.00	21,357.76	฿53,363.71	8232	32,005.95
12	26,796.00	22,746.13	฿56,965.64	8717	34,219.51

รูปที่ ก-63 แสดงตัวอย่างการใช้ฟังชันก์ StrToSet

5.39 Subset

Returns subset ของ tuples จาก set ที่กำหนด

Syntax

Subset(«Set», «Start»[, «Count»])

ตัวอย่าง

```
SELECT {Measures.Members} ON COLUMNS,
    SubSet([Store].[Store State].MEMBERS,5,10)  ON ROWS
FROM  [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Yucatan					
Zacatecas					
CA	74,748.00	63,530.43	฿159,167.84	24442	95,637.41
OR	67,659.00	56,772.50	฿142,277.07	21611	85,504.57
WA	124,366.00	105,324.31	฿263,793.22	40784	158,468.91

รูปที่ ก-64 แสดงตัวอย่างการใช้ฟังชันก์ Subset

5.40 Tail

Returns subset(ตามจำนวนที่ต้องการ) ของ set ที่กำหนด โดย return จากตัวสุดท้ายก่อน

Syntax

Tail(«Set»[, «Count»])

ตัวอย่าง

```
SELECT  {[Measures].[Unit Sales]} ON COLUMNS,
    Tail([Store].[Store State].MEMBERS,10)  ON ROWS
FROM Sales
```

	Unit Sales
BC	
DF	
Guerrero	
Jalisco	
Veracruz	
Yucatan	
Zacatecas	
CA	74,748.00
OR	67,659.00
WA	124,366.00

รูปที่ ก-65 แสดงตัวอย่างการใช้ฟังชันก์ Tail

5.41 ToggleDrillState

Syntax

ToggleDrillState(«Set1», «Set2»[, RECURSIVE])

5.42 TopCount

เรียงค่าจำนวนที่ต้องการจากมากไปน้อย แล้ว return เฉพาะที่รวมกันแล้วได้มากกว่าจำนวนที่กำหนดเป็น Percent

Syntax

TopCount(«Set», «Count»[, «Numeric Expression»])

ตัวอย่าง

SELECT Measures.MEMBERS ON COLUMNS,

TOPCOUNT({[Store].[Store City].MEMBERS}, 12, Measures.[Sales Count]) ON ROWS

FROM [Sales]

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Salem	41,580.00	34,823.56	฿87,218.28	13347	52,394.72
Tacoma	35,257.00	29,959.28	฿74,843.96	11184	44,884.68
Portland	26,079.00	21,948.94	฿55,058.79	8264	33,109.85
Los Angeles	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
San Diego	25,635.00	21,713.53	฿54,431.14	8095	32,717.61
Seattle	25,011.00	20,956.80	฿52,644.07	7956	31,687.27
Bremerton	24,576.00	21,121.96	฿52,896.30	7876	31,774.34
Spokane	23,591.00	19,795.49	฿49,634.46	7397	29,838.97
Beverly Hills	21,333.00	18,266.44	฿45,750.24	6815	27,483.80
Yakima	11,491.00	9,713.81	฿24,329.23	3652	14,615.42
Bellingham	2,237.00	1,896.62	฿4,739.23	1380	2,842.61
Walla Walla	2,203.00	1,880.34	฿4,705.97	1339	2,825.63

รูปที่ ก-66 แสดงตัวอย่างการใช้ฟังชันก์ TopCount

5.43 TopPercent

เรียงค่าจำนวนที่ต้องการจากมากไปน้อยแล้ว return เฉพาะที่รวมกันแล้วได้มากกว่าจำนวนที่กำหนดเป็น Percent

Syntax

TopPercent(«Set», «Percentage», «Numeric Expression»)

ตัวอย่าง

SELECT Measures.MEMBERS ON COLUMNS,

TOPPERCENT({[Store].[Store City].MEMBERS}, 50, Measures.[Sales Count]) ON ROWS

FROM [Sales]

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales Net
Salem	41,580.00	34,823.56	฿87,218.28	13347	52,394.72
Tacoma	35,257.00	29,959.28	฿74,843.96	11184	44,884.68
Portland	26,079.00	21,948.94	฿55,058.79	8264	33,109.85
Los Angeles	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
San Diego	25,635.00	21,713.53	฿54,431.14	8095	32,717.61

รูปที่ ก-67 แสดงตัวอย่างการใช้ฟังชันก์ TopPercent

5.44 TopSum

เรียงค่าจำนวนที่ต้องการจากมากไปน้อยแล้ว return เฉพาะที่รวมกันแล้วได้มากกว่าจำนวนที่กำหนด

Syntax

TopSum(«Set», «Value», «Numeric Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
Topsum( [Product].[Product Name].Members, 100 , [Measures].[Sales Count]) ON ROWS
FROM [Sales]
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Special Whe	267.00	310.70	฿776.97	89	466.27
Hermanos Bi	257.00	291.49	฿742.73	82	451.24

รูปที่ ก-68 แสดงตัวอย่างการใช้ฟังชันก์ TopSum

5.45 Union

Returns set ที่เกิดจากการ union ของ 2 sets ที่กำหนด

Syntax

Union(«Set1», «Set2»[, ALL])

Alternate Syntax 1

{«Set1», «Set2»}

Alternate Syntax 2

«Set1» + «Set 2»

ตัวอย่าง

```
Select Measures.MEMBERS on Columns,
UNION(TopCount([Store Name].Members, 5, (Profit,[1997].[Q1])),
      TopCount([Store Name].Members,5,(Profit,[1997].[Q2])))
on Rows from Sales
```

	Unit Sales	Store Cost	Store Sales	Sales Count	Store Sales I
Store 13	41,580.00	34,823.56	฿87,218.28	13347	52,394.72
Store 17	35,257.00	29,959.28	฿74,843.96	11184	44,884.68
Store 11	26,079.00	21,948.94	฿55,058.79	8264	33,109.85
Store 7	25,663.00	21,771.54	฿54,545.28	8207	32,773.74
Store 24	25,635.00	21,713.53	฿54,431.14	8095	32,717.61
Store 3	24,576.00	21,121.96	฿52,896.30	7876	31,774.34

รูปที่ ก-69 แสดงตัวอย่างการใช้ฟังชันก์ Union

5.46 VisualTotals

Syntax

VisualTotals(«Set», «Pattern»)

5.47 Wtd

Returns set ของ members จาก level Week ใน Time dimension โดยเริ่มจากค่าแรกจนถึงค่าที่กำหนด

Syntax

Wtd([«Member»])

5.48 Ytd

Returns set ของ members จาก level Year ใน Time dimension โดยเริ่มจากค่าแรกจนถึงค่าที่กำหนด

Syntax

Ytd([«Member»])

6 String Functions

6.1 CalculationPassValue

Syntax

Numeric

CalculationPassValue(«Numeric Expression», «Pass Value»[, «Access Flag»])

String

CalculationPassValue(«String Expression», «Pass Value»[, «Access Flag»])

6.2 CoalesceEmpty

ตรวจสอบค่าที่ต้องการ ถ้าเป็น Empty ให้ return ค่าหนึ่ง ถ้าไม่ใช่ให้ return อีกค่าหนึ่ง

Syntax

Numeric **CoalesceEmpty**(«Numeric Expression»[, «Numeric Expression»]...)

String **CoalesceEmpty**(«String Expression»[, «String Expression»]...)

ตัวอย่าง

```
WITH MEMBER Measures.[Profit Growth] AS
    '(Measures.[Profit]) /
    COALESCEEMPTY((Measures.[Profit], [Time].PREVMEMBER),Measures.[Profit])',
    FORMAT_STRING = '#.00%'
SELECT {Measures.[Profit], Measures.[Profit Growth]} ON COLUMNS,
     {DESCENDANTS([Time].[1997], [Month])} ON ROWS
FROM [Sales]
```

	Profit	Profit Growt
1	27,361.20	100.00%
2	26,459.12	96.70%
3	30,055.79	113.59%
4	25,766.55	85.73%
5	26,673.73	103.52%
6	27,261.76	102.20%
7	30,241.09	110.93%
8	27,763.08	91.81%
9	26,362.85	94.96%
10	25,440.26	96.50%
11	32,005.95	125.81%
12	34,219.51	106.92%

รูปที่ ก-70 แสดงตัวอย่างการใช้ฟังชันก์ CoalesceEmpty

6.3 Generate

Return set ของ member โดยการ joins คำตอบของ sets ด้วยการ union

Syntax

Set **Generate**(«Set1», «Set2»[, ALL])

String **Generate**(«Set», «String Expression»[, «Delimiter»])

6.4 IIf

Returns one ของ 2 numeric หรือ string values ที่ได้จาก logical test

Syntax

Numeric **IIf**(«Logical Expression», «Numeric Expression1», «Numeric Expression2»)

String **IIf**(«Logical Expression», «String Expression1», «String Expression2»)

6.5 LookupCube

Syntax

Numeric **LookupCube**(«Cube String», «Numeric Expression»)

String **LookupCube**(«Cube String», «String Expression»)

6.6 MemberToStr

สร้าง string จาก member ที่กำหนด

Syntax

MemberToStr(«Member»)

6.7 Name

Returns name ของ level, dimension, member, หรือ hierarchy ที่กำหนด

Syntax

Dimension «Dimension».**Name**

Level «Level».**Name**

Member «Member».**Name**

Hierarchy «Hierarchy».**Name**

6.8 Properties

Returns string ที่เป็น property ของ member

Syntax

«Member».**Properties**(«String Expression»)

ตัวอย่าง

```
WITH   MEMBER [Measures].[Store Type] AS
        '[Store].CurrentMember.Properties("Store Type")'
SELECT
  {[Measures].[Unit Sales], [Measures].[Store Type]} ON COLUMNS,
  {[Store].[Store Name].Members} ON ROWS
From Sales
```

	Unit Sales	Store Type
Store 19		Deluxe Supermarket
Store 20		Mid-Size Grocery
Store 9		Mid-Size Grocery
Store 21		Deluxe Supermarket
Store 1		Supermarket
Store 5		Small Grocery
Store 10		Supermarket
Store 8		Deluxe Supermarket
Store 4		Gourmet Supermarket
Store 12		Deluxe Supermarket
Store 18		Mid-Size Grocery
HQ		HeadQuarters
Store 6	21,333.00	Gourmet Supermarket
Store 7	25,663.00	Supermarket
Store 24	25,635.00	Supermarket
Store 14	2,117.00	Small Grocery
Store 11	26,079.00	Supermarket
Store 13	41,580.00	Deluxe Supermarket
Store 2	2,237.00	Small Grocery
Store 3	24,576.00	Supermarket
Store 15	25,011.00	Supermarket
Store 16	23,591.00	Supermarket
Store 17	35,257.00	Deluxe Supermarket
Store 22	2,203.00	Small Grocery
Store 23	11,491.00	Mid-Size Grocery

รูปที่ ก-71 แสดงตัวอย่างการใช้ฟังชันก์ CurrentMember.Properties

6.9 SetToStr

สร้าง string จาก set ที่กำหนด

Syntax

SetToStr(«Set»)

6.10 TupleToStr

Returns string จาก tuple ที่กำหนด

Syntax

TupleToStr(«Tuple»)

6.11 UniqueName

Returns unique name ของ level, dimension, member, or hierarchy ที่กำหนด

Syntax

Dimension «Dimension».**UniqueName**

Level «Level».**UniqueName**

Member «Member».**UniqueName**

Hierarchy «Hierarchy».**UniqueName**

6.12 UserName

7 Tuple Functions

7.1 Current

7.2 StrToTuple

Constructs a tuple from a specified string expression in Multidimensional Expressions (MDX) format.

Syntax

StrToTuple(«String Expression»)

ตัวอย่าง

```
SELECT Measures.MEMBERS ON COLUMNS,
  {{ StrToTuple("( [1997],product)"),StrToTuple("( [1997],product)") }.Item(1)}  ON ROWS
FROM  [Sales]
```

8 Other Functions

8.1 Call

Syntax

Call «UDF Name»

ภาคผนวก ข

คู่มือสำหรับโปรแกรมเมอร์

1. การพัฒนาโปรแกรมประยุกต์ทำงานในลักษณะ Client - Server

โปรแกรมประยุกต์ได้พัฒนาขึ้นมาโดยใช้ภาษา Visual Basic 6.0 โดยสิ่งที่จำเป็นต้องทำก่อนก็คือ ติดตั้ง PivotTable Service ซึ่งสามารถติดตั้งในได้จากแผ่น CD Microsoft SQL Server 2000 และต้องมีการลงทะเบียนไฟล์ msmdcb.ocx บนเครื่องคอมพิวเตอร์เสียก่อน เพราะในโปรแกรมได้เรียกใช้คอมโพเนนท์ในไฟล์นี้ หลังจากนั้นเวลาเขียนโปรแกรมในภาษา Visual Basic ก็ต้องไปเพิ่มคอมโพเนนท์ OLAP Manager Cube Browser เข้ามา โดยคลิกที่ **Project -> Component** แล้วที่แท็ป Controls ให้คลิกเลือก OLAP Manager Cube Browser

1.1 ฟอร์มทั้งหมดที่ใช้ในโปรแกรม

1.1.1 CubeBrowserForm

1.1.2 ConnectForm

Procedure และ Function

- CheckConnectInfo() ตรวจสอบว่ามีการเชื่อมต่ออยู่หรือไม่
- FopenConnection() ทำการเชื่อมต่อ connection ถ้าเชื่อมต่อสำเร็จจะส่งค่ากลับมา
- GetConnection () สร้างการเชื่อมต่อไปยังเซิร์ฟเวอร์และเลือกเลือก cube
- GetConnectionStr() สร้าง connection และ ส่งค่าstring กลับจาก user
- InitForm() กำหนดค่าเริ่มต้นให้กับ form
- SgetComputerName() รับค่า Computer Name
- StrimNullChars() แก้ไข ตัวอักขระหลังจากที่ได้มาเป็น null

1.1.3 MDX QueryFrom

Procedure และ Function

- Connect () ใช้เชื่อมต่อไปยัง Multidimensional data source
- Disconnect() ใช้ยกเลิกการเชื่อมต่อไปยัง Multidimensional data source
- UpdateConnectionStateUI() ใช้ update form
- UpdateDatabaseList() ใช้ update ชื่อของดาต้าเบส
- fconnected () จะทำการ ส่งค่า true กลับ ถ้าสามารถเชื่อมต่อกับ data source ได้
- FileNew() สร้าง file ใหม่สำหรับป้อน Query
- FileOpen() เปิด file ที่มีQuery อยู่ขึ้นมา
- FileSave() บันทึก file
- FileSaveAs() บันทึกเป็น file

- Form_Resize() เมื่อเมื่อใช้ทำการ Resize form โปรแกรมก็ทำการแก้ไขค่าต่างๆของคอมโพแนนต์ ในส่วนของ UI

บรรณานุกรม

หนังสืออ้างอิง

[1] Harry Singh : " *Data Warehousing : Concepts, Technologies, Implementations and Management* ", Prentice-Hall , 1996.

[2] Olap Train and Reed Jacobson : " *Microsoft SQL Server 2000 Analysis Services Step by Step* ", Microsoft Press , 2001.

[3] อำไพ สินลิขิตกุล : " *อินไซท์ SQL Server 7 Step by Step ครอบคลุมเวอร์ชัน 2000* " , โปรวิชั่น , 2544.

เว็บไซต์อ้างอิง

[1] http://www.msdn.microsoft.com