การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

DATA WAREHOUSE

โดย

นายกิตติพงษ์ ประโยชน์บริบูรณ์

นายภัทรชัย วิเศษบรรจง

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

DATA WAREHOUSE

ผู้จัดทำ

1. นายกิตติพงษ์ ประโยชน์บริบูรณ์ รหัสประจำตัว 40010052
2. นายภัทรชัย วิเศษบรรจง รหัสประจำตัว 40010548

อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

การศึกษาและพัฒนาระบบดาต้าแวร์เฮาส์

นายกิตติพงษ์ ประโยชน์บริบูรณ์ 40010052
นายภัทรชัย วิเศษบรรจง 40010548
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและออกแบบระบบฐานข้อมูลดาต้าแวร์เฮาส์ (Data Warehouse) ซึ่งเป็นระบบฐานข้อมูลแบบใหม่ โดยเปรียบเทียบกับระบบฐานข้อมูลทั่วไป (Relational Database) โดยระบบฐานข้อมูลดาต้าแวร์เฮาส์ที่ศึกษาจะใช้หลักการจัดเก็บฐานข้อมูลแบบหลายมิติ (Multidimensional Database) ซึ่งได้ทำการศึกษาจากซอฟท์แวร์ที่มีความสามารถในการทำงานกับระบบฐานข้อมูลหลายมิติ (Multidimensionality) ที่มีชื่อว่า OLAP Manager และทำการพัฒนาโปรแกรมประยุกต์ (Application) โดยใช้โปรแกรมวิชวลเบสิก (Visual Basic) รุ่น 6.0 ทำการติดต่อกับผู้ใช้ เพื่อให้สามารถเรียกใช้ข้อมูลที่เก็บในระบบฐานข้อมูลดาต้าแวร์เฮาส์ ที่สร้างขึ้นบนระบบจัดการฐานข้อมูล SQL Server ได้ โดยโปรแกรมประยุกต์ที่พัฒนาขึ้นจะสามารถนำตารางต่างๆ ที่มีโครงสร้างฐานข้อมูลเป็นแบบหลายมิติ มาสร้างเป็นแบบจำลอง (Model) และทำการวิเคราะห์ได้ตามความต้องการ ทำให้เกิดความยืดหยุ่นในการใช้งานมากยิ่งขึ้น

Data Warehouse

Kittipong Prayotboriboon
Phattarachai Wisetbunchong
Dr.Worrawat Limpoka Advisor

Abstract

This project deals with the studying and designing of Data Warehouse, a new kind of Database, and compares with Relational Database. The studying about the Multidimensional database theory and concept from a software based on multidimensionality, named " OLAP Manager " and the application has been developed with Visual Basic version 6.0 for searching data from the SQL Server database. This application can build models from any tables that a structure is multidimensional database and users can analyze the data from data warehouse with flexibility.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ

อาจารย์ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำและ ช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

อาจารย์ทุกๆท่านที่ประสิทธิ์ประสาทวิชาความรู้ต่างๆ ตลอดระยะเวลาสี่ปีที่ผ่านมา

เพื่อนๆ 4D ทุกคน สำหรับคำแนะนำ คอยให้ความช่วยเหลือทุกครั้งที่เกิดปัญหาและให้กำลังใจเสมอมา

และต้องขอขอบพระคุณบุคคลที่สำคัญที่สุด ที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดาและมารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุกๆด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

สุดท้ายขอมอบคุณความดีจากปริญญานิพนธ์ฉบับนี้ให้แด่ คุณ บิดา มารดาและครูอาจารย์ผู้มีพระคุณทุกท่าน

นายกิตติพงษ์ ประโยชน์บริบูรณ์
นายภัทรชัย วิเศษบรรจง

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญภาพ	VIII
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของการทำโครงงาน	1
1.3 ขอบเขตของโครงงาน	2
1.4 ขั้นตอนการดำเนินงาน	2
บทที่ 2 ทฤษฎีเบื้องต้น	3
2.1 นิยามของดาต้าแวร์เฮาส์	3
2.2 จุดมุ่งหมายของดาต้าแวร์เฮาส์	3
2.3 ระบบฐานข้อมูลแบบทรานแซคชัน (OLTP)	4
2.4 ความแตกต่างของดาต้าแวร์เฮาส์กับระบบฐานข้อมูลประจำวัน	4
2.5 ความแตกต่างของดาต้าแวร์เฮาส์กับ Operational Data Stores	6
2.6 ความแตกต่างของการจัดการฐานข้อมูลแบบมัลติไดเมนชันแนลและรีเลชันแนล	7
2.7 OLAP Services กับดาต้าแวร์เฮาส์ในระบบจัดการฐานข้อมูล SQL Server	9
2.7.1 จุดเด่นของ OLAP Services	10
บทที่ 3 โครงสร้างและการออกแบบของดาต้าแวร์เฮาส์	12
3.1 Dimensional Modeling	12
3.2 การออกแบบข้อมูล (Data Modeling)	13
3.2.1 Data Cubes	13
3.2.2 โอเปอเรชัน (Operations) ที่ใช้ในการวิเคราะห์ข้อมูล	14
3.3 โครงสร้างฐานข้อมูล (Database Schema)	17
3.3.1 Star Schema	17
3.3.2 Snowflake Schema	18
3.4 การรวบรวมข้อมูล (Aggregation)	19
3.5 Cross-tabulations หรือ Crosstabs	19
3.6 ขั้นตอนการออกแบบดาต้าแวร์เฮาส์	21

สารบัญ (ต่อ)

หน้า

บทที่ 4 OLAP Server Architecture 22

4.1 Microsoft OLAP Services 22

4.2 OLAP Server Architecture 26

4.2.1 ซอฟท์แวร์ OLAP Manager 27

4.2.2 คอมโพเนนต์ (Components) ต่างๆ ของซอฟท์แวร์ OLAP Manager 29

4.2.3 ขั้นตอนการใช้งานซอฟท์แวร์ OLAP Manager 31

บทที่ 5 OLAP Client Architecture 47

5.1 OLAP Client Architecture 47

5.2 PivotTable Service 47

5.2.1 ลักษณะเด่นของ PivotTable Service 48

5.2.2 คอมโพเนนต์ที่สำคัญที่ใช้กับ PivotTable Service 48

5.2.3 การติดตั้งและลงทะเบียนคอมโพเนนต์ 49

5.2.4 ลักษณะของการเชื่อมต่อของ PivotTable Service 49

5.2.5 Client Operations ใน PivotTable Service 51

บทที่ 6 การออกแบบ พัฒนาและการใช้งานโปรแกรมประยุกต์ 56

6.1 ทูลส์ที่นำมาใช้ในการพัฒนาโปรแกรมประยุกต์ 56

6.2 การออกแบบฐานข้อมูล (Database Design) 57

6.3 การออกแบบโปรแกรมประยุกต์ 57

6.4 คอมโพเนนต์ใน Visual Basic 6.0 ที่นำมาใช้สร้างโปรแกรมประยุกต์ทางด้านฐานข้อมูล 59

6.5 ขั้นตอนการใช้งานโปรแกรมประยุกต์ 61

บทที่ 7 การทดสอบประสิทธิภาพของโปรแกรมประยุกต์ 75

7.1 การใช้งานอินเด็กซ์ (Index) ในฐานข้อมูลดาต้าแวร์เฮาส์ 75

7.2 การทดลองวัดประสิทธิภาพของการ Query เทียบกับเวลาในการจัดเก็บลักษณะต่างๆ 75

7.2.1 ฐานข้อมูลดาต้าแวร์เฮาส์ที่มีการใช้อินเด็กซ์ 75

7.2.2 ฐานข้อมูลดาต้าแวร์เฮาส์ที่ไม่มีการใช้อินเด็กซ์ 77

7.3 สรุปผลการทดลอง 78

บทที่ 8 บทสรุปและวิจารณ์ 79

8.1 บทสรุปและวิจารณ์ 79

8.2 แนวทางในการพัฒนาเพิ่มเติม 79

ภาคผนวก ก Decision Support Objects (DSO) 80

ภาคผนวก ข Multidimensional Expression (MDX) 86

ภาคผนวก ค Microsoft Hierarchical Flex Grid (MSHFlexGrid) 91
ภาคผนวก ง คู่มือสำหรับโปรแกรมเมอร์ 94
บรรณานุกรม 106

สารบัญตาราง

		หน้า
ตารางที่ 2.1	สรุปความแตกต่างของดาต้าแวร์เฮาส์กับระบบฐานข้อมูลปฏิบัติการ	7
ตารางที่ 3.1	แสดงการ Drill-down และ Drill-up	15
ตารางที่ 3.2	แสดงการตัดข้อมูล (Slicing)	15
ตารางที่ 3.3	แสดงการหมุนข้อมูล (Dicing)	16
ตารางที่ 3.4	แสดงการตัดข้อมูลจากข้อมูลในตารางที่ 3.3	16
ตารางที่ 5.1	แสดง File set ของไฟล์คอมโพเนนต์	49
ตารางที่ 5.2	แสดง File set ที่ใช้กับลักษณะงานในแอพลิเคชันทางฝั่ง Client	49
ตารางที่ 7.1	เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ MOLAP	76
ตารางที่ 7.2	เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ ROLAP	76
ตารางที่ 7.3	เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ HOLAP	76
ตารางที่ 7.4	เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ MOLAP	77
ตารางที่ 7.5	เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ ROLAP	77
ตารางที่ 7.6	เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ HOLAP	77
ตารางที่ ก-1	DSO Objects และ Interfaces	81
ตารางที่ ข-1	แสดงผลลัพธ์ของการ Slice ข้อมูลโดยใช้ MDX Statement	88
ตารางที่ ข-2	แสดงผลลัพธ์ของการใช้ฟังก์ชัน CrossJoin ( )	89
ตารางที่ ค-1	แสดงการตั้งค่าใน AllowUserResizing property	91
ตารางที่ ค-2	แสดงการตั้งค่าใน MergeCells property	93

สารบัญภาพ

หน้า

รูปที่ 2.1 โครงสร้างของสถาปัตยกรรมของ SQL Server data warehousing and OLAP 10

รูปที่ 3.1 แสดงไดเมนชันใน Dimensional Modeling 13

รูปที่ 3.2 แสดง Data Cubes 14

รูปที่ 3.3 แสดงโครงสร้างฐานข้อมูลแบบ Star Schema 17

รูปที่ 3.4 แสดงโครงสร้างฐานข้อมูลที่เป็นแบบสโนว์เฟลก (Snowflake Schema) 18

รูปที่ 3.5 แสดงการรวบรวม (Aggregation) 19

รูปที่ 3.6 แสดงข้อมูลในตารางแบบปกติ 20

รูปที่ 3.7 แสดงข้อมูลในตารางแบบ CrossTabs 20

รูปที่ 4.1 ภาพรวมของ OLAP Services Architecture 22

รูปที่ 4.2 ส่วนประกอบในระดับต่างๆของ OLAP Services 23

รูปที่ 4.3 แสดงการเก็บข้อมูลใน Repository ของดาต้าคิวบ์ 24

รูปที่ 4.4 แสดงโครงสร้างของ Repository 25

รูปที่ 4.5 OLAP Server Architecture 26

รูปที่ 4.6 OLAP Manager 27

รูปที่ 4.7 OLAP Manager Tree pane 28

รูปที่ 4.8 OLAP Manager HTML-based detail pane 29

รูปที่ 4.9 แสดงการติดตั้ง Data Source Name 33

รูปที่ 4.10 แสดงการสร้าง Data Source ขึ้นมาใหม่ 34

รูปที่ 4.11 แสดงการทดสอบการเชื่อมต่อกับ Data Source ที่เลือกมา 35

รูปที่ 4.12 แสดงการเลือกไดเมนชันต่างๆ ที่จะใช้ใน Cube 36

รูปที่ 4.13 แสดงการเลือกชนิดของไดเมนชัน 37

รูปที่ 4.14 แสดงการเชื่อมต่อกันของไดเมนชันที่เลือก 38

รูปที่ 4.15 แสดงหน้าต่างของ Cube Editor Wizard 39

รูปที่ 4.16 แสดงการเลือกชนิดของการจัดเก็บข้อมูลใน Cube 40

รูปที่ 4.17 แสดงการวัดประสิทธิภาพและขนาดของ Cube ที่สร้างขึ้นมา 41

รูปที่ 4.18 แสดงรายละเอียดข้อมูล (Metadata) ใน Cube 42

รูปที่ 4.19 แสดงหน้าต่างของ Cube Browser 43

รูปที่ 4.20 แสดงการเลือกดูข้อมูลในระดับต่างๆ ของ Time Dimension 44

รูปที่ 4.21 แสดงการ Drill-down ข้อมูลใน Cube Browser 45

รูปที่ 4.22 แสดงผลของการประมวลผล Cube ที่ทำการสร้างขึ้นมา 46

สารบัญภาพ (ต่อ)

หน้า

รูปที่ 5.1 โครงสร้างของ PivotTable Service Architecture 47
รูปที่ 5.2 แสดงถึงการเชื่อมต่อระหว่าง PivotTable Service กับ Analysis Services 50
รูปที่ 5.3 แสดงการเชื่อมต่อกับ Local ROLAP Cube 50
รูปที่ 5.4 แสดงการเชื่อมต่อระหว่าง PivotTable Service กับ MOLAP Cube File 51
รูปที่ 5.5 แสดง CubeDef Object Model ที่ใช้ ADO MD 53
รูปที่ 6.1 แสดงการออกแบบฐานข้อมูลแบบสโนว์เฟลก 57
รูปที่ 6.2 ฟอร์มสำหรับติดต่อกับเซิร์ฟเวอร์และดาต้าเบส 58
รูปที่ 6.3 ฟอร์มการทำงานหลักของโปรแกรมประยุกต์ 59
รูปที่ 6.4 แสดงคอมโพเนนต์ต่างๆ ที่ใช้ในการสร้างแอพลิเคชันทางด้านฐานข้อมูล 60
รูปที่ 6.5 แสดงการติดต่อกับเซิร์ฟเวอร์และดาต้าเบส 61
รูปที่ 6.6 แสดงหน้าจอหลักที่ใช้ในการวิเคราะห์และแสดงข้อมูล 61
รูปที่ 6.7 แสดงการ Add ไดเมนชันให้เป็นคอลัมน์ 62
รูปที่ 6.8 แสดงการ Add ไดเมนชันให้เป็นแถว 63
รูปที่ 6.9 แสดงการ Add ตัววัดค่าให้เป็นแถว 63
รูปที่ 6.10 แสดงการตัดข้อมูลที่สนใจ เพื่อนำไปวิเคราะห์ 64
รูปที่ 6.11 แสดงการนำข้อมูลที่ต้องการขึ้นมาวิเคราะห์ 65
รูปที่ 6.12 แสดงการลบไดเมนชันที่ไม่ต้องการวิเคราะห์ 66
รูปที่ 6.13 แสดงข้อมูลที่ลบไดเมนชันที่ไม่ต้องการออกไปแล้ว 67
รูปที่ 6.14 แสดงวิธีการหมุนข้อมูล 68
รูปที่ 6.15 แสดงการหมุนไดเมนชันจาก Column ไปเป็น Row 69
รูปที่ 6.16 แสดงการเลือกระดับในการ Drill Down 70
รูปที่ 6.17 แสดงการ Drill Down ข้อมูล 71
รูปที่ 6.18 แสดงการเลือกดูข้อมูลแบบ Split View 72
รูปที่ 6.19 แสดงการเลือกดูข้อมูลแบบ Result View 72
รูปที่ 6.20 แสดงการเลือกดูข้อมูลแบบ Query View 73
รูปที่ 6.21 แสดงการเคลียร์ข้อมูลและโมเดล 73
รูปที่ 6.22 แสดงการ Disconnect กับเซิร์ฟเวอร์และดาต้าเบส 74
รูปที่ 7.1 แสดงโครงสร้างสำหรับตารางที่มีการสร้างอินเด็กซ์ 75
รูปที่ ก-1 DSO Object Model Hierarchy 80
รูปที่ ก-2 DSO Database Object 82

สารบัญภาพ (ต่อ)

หน้า

รูปที่ ก-3 DSO Cube Object 82

รูปที่ ก-4 แสดง DSO Partition Objects 83

รูปที่ ก-5 แสดง DSO Aggregations 84

รูปที่ ก-6 แสดง DSO Virtual Cubes 84

รูปที่ ข-1 แสดงโครงสร้างของ MDX 86

รูปที่ ข-2 แสดงการ DrillDownMember 89

รูปที่ ข-3 แสดงการ DrillUpMember 89

รูปที่ ข-4 แสดงการ DrillDownLevel 90

รูปที่ ข-5 แสดงการ DrillUpLevel 90

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันนี้มีการใช้ฐานข้อมูลอย่างกว้างขวางในระบบงานทั่วๆไป จึงมีการวิจัยและพัฒนาวิธีการเก็บ ข้อมูลจำนวนมาก รวมถึงการค้นหาและนำข้อมูลที่ต้องการออกมาจากระบบฐานข้อมูลด้วย แต่เนื่องจากฐานข้อมูลประจำวัน (Operational Database) ที่นิยมใช้กันอยู่ในปัจจุบันนี้ มีหลักในการเก็บข้อมูลที่เน้นในเรื่อง การลดความซ้ำซ้อน (Redundancy), รักษาความถูกต้อง (Integrity), ลดการสูญหายของข้อมูล (Information Lost) และลดความผิดพลาดที่เกิดขึ้นจากการแก้ไขข้อมูล (Update Anomalies)

เนื่องจากฐานข้อมูลประจำวันมีลักษณะดังได้กล่าวมาแล้ว จึงมีความสามารถเพียงแค่การเรียกใช้ข้อมูลที่มีอยู่แต่ไม่สามารถจะนำมาช่วยในการสนับสนุนการตัดสินใจได้เพราะเมื่อมีการเรียกใช้ข้อมูล จะต้องเรียกใช้ข้อมูลจากฐานข้อมูลขนาดใหญ่ ซึ่งมีข้อมูลจำนวนมหาศาลและมีการแตกตารางที่นอร์มอลไลซ์แล้ว (Normalized tables) ออกเป็นหลายตาราง จึงไม่รองรับคำถามที่ต้องการจะนำมาใช้ช่วยสนับสนุนการตัดสินใจ (Decision support queries) เพราะ คำถาม (Queries) ทั้งหมดของระบบที่ช่วยสนับสนุนการตัดสินใจ (Decision support system) จะมีการรวมกัน (join) ของตารางต่างๆที่ซับซ้อน และนอกจากนี้การเก็บข้อมูลในระบบฐานข้อมูลประจำวัน ยังไม่มีการเก็บข้อมูลย้อนหลัง (Historical Data) เพื่อใช้ช่วยในการคาดคะเนแนวโน้มที่คาดว่าจะเป็นไปได้ในอนาคต

ดังนั้นจึงทำการศึกษาถึงระบบฐานข้อมูลที่สามารถให้ข้อมูลที่นำมาช่วยสนับสนุนการตัดสินใจได้ เพื่อจะได้สามารถใช้ข้อมูลที่มีอยู่ได้อย่างมีประสิทธิภาพสูงสุด ซึ่งระบบฐานข้อมูลที่ได้ทำการศึกษานั้นก็คือ ระบบฐานข้อมูลดาต้าแวร์เฮาส์ (Data Warehouse) เนื่องจากระบบฐานข้อมูลดาต้าแวร์เฮาส์ถูกคิดขึ้นมาเพื่อช่วยให้ผู้ใช้เรียกใช้ข้อมูลด้วยวิธีที่สร้างสรรค์ เพราะธรรมชาติที่แตกต่างกันของระบบฐานข้อมูลดาต้าแวร์เฮาส์และระบบฐานข้อมูลประจำวัน

โดยระบบฐานข้อมูลดาต้าแวร์เฮาส์ จะแยกกลุ่มข้อมูลสารสนเทศที่ใช้ในการวิเคราะห์ทางธุรกิจออกจากฐานข้อมูลที่ใช้ประจำวัน มาเก็บอยู่ในระบบจัดการฐานข้อมูลรีเลชันแนล (Relational Database Management System) ที่มีประสิทธิภาพสูง และทำให้การเรียกใช้ข้อมูลชุดนี้ทำได้อย่างยืดหยุ่นจากเครื่องมือที่อยู่บนเครื่องเดสก์ทอปทั่วไป โดยลดออฟโหลดดิ้ง (off-loading) เพิ่มกลไกการช่วยตัดสินใจ ปรับปรุงเวลาที่ตอบสนอง (Response time) ให้รวดเร็วขึ้นอย่างมากและผู้บริหารสามารถเรียกข้อมูลรายละเอียดที่จำเป็น ที่ถูกเก็บมาก่อนหน้านี้ (Historical Data) มาใช้ช่วยในการตัดสินใจทางธุรกิจให้เกิดความแม่นยำมากยิ่งขึ้น โดยระบบนี้จะพิจารณาถึงการใช้ประโยชน์จากข้อมูลเป็นหลัก

1.2 วัตถุประสงค์ของการทำโครงงาน

1.2.1 ศึกษาว่าระบบฐานข้อมูลดาต้าแวร์เฮาส์ คืออะไร มีประโยชน์อย่างไรในเชิงธุรกิจ

1.2.2 ศึกษาว่าระบบฐานข้อมูลดาต้าแวร์เฮาส์นี้ มีโครงสร้างข้อมูล (Data Structure) เป็นอย่างไร และมี หลักการในการออกแบบระบบฐานข้อมูลดาต้าแวร์เฮาส์อย่างไร

1.2.3 ศึกษาระบบจัดการฐานข้อมูล SQL Server ว่ามีฟังก์ชันการทำงานเป็นอย่างไร และสามารถสร้างระบบฐานข้อมูลดาต้าแวร์เฮาส์บน SQL Server ได้อย่างไร

1.2.4 ศึกษาการออกแบบและการสร้างระบบฐานข้อมูลดาต้าแวร์เฮาส์ บนระบบจัดการฐานข้อมูล SQL Server ว่าต้องมีการออกแบบส่วนใดบ้าง และจะมีวิธีการใดในการทดสอบว่า ระบบฐานข้อมูลดาต้าแวร์เฮาส์ ที่ทำการออกแบบไว้นั้นสามารถดึงข้อมูลที่ต้องการออกมาได้อย่างถูกต้อง

1.2.5 ศึกษาวิธีการและทดลองทำการเขียนทูลส์ ทำการติดต่อกับผู้ใช้ (User Interface) เพื่อให้ผู้ใช้สามารถเรียกใช้ข้อมูลที่เก็บในระบบฐานข้อมูลดาต้าแวร์เฮาส์ ที่สร้างบนระบบจัดการฐานข้อมูล SQL Server ได้โดยง่าย

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการศึกษาระบบฐานข้อมูลดาต้าแวร์เฮาส์ ที่สามารถให้ข้อมูลที่นำมาช่วยสนับสนุน การตัดสินใจเพื่อจะได้เรียนรู้ถึงประโยชน์ โครงสร้าง การออกแบบและวิธีการใช้งานระบบฐานข้อมูลดาต้าแวร์เฮาส์ เพื่อให้ได้ประสิทธิภาพสูงสุด

หลังจากที่ได้ทำการศึกษาแล้ว จึงทำการออกแบบฐานข้อมูลดาต้าแวร์เฮาส์และสร้างโปรแกรมประยุกต์ทางฝั่ง Client (Client Application) โดยใช้โปรแกรมวิชวลเบสิก รุ่น 6.0 (Visual Basic Version 6.0) เพื่อให้ผู้ใช้สามารถทำการเรียกใช้และติดต่อกับระบบจัดการฐานข้อมูล SQL Server ได้ โดยโปรแกรมประยุกต์ที่พัฒนาขึ้นนี้จะมีความอ่อนตัวเป็นอย่างมาก เนื่องจากทูลส์ที่สร้างขึ้นนี้สามารถใช้ในการขับเคลื่อนตารางใดๆก็ได้ ที่มีโครงสร้างฐานข้อมูลเป็นแบบมัลติไดเมนชันแนล (Multidimensional Database)

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ทำการศึกษาเกี่ยวกับระบบฐานข้อมูลดาต้าแวร์เฮาส์ และประโยชน์ของระบบฐานข้อมูลดาต้าแวร์เฮาส์ในเชิงธุรกิจ โดยการศึกษาจากหนังสืออ้างอิงและทางอินเตอร์เน็ต

1.4.2 ทำการศึกษาโครงสร้างของระบบฐานข้อมูลดาต้าแวร์เฮาส์ และศึกษาหลักการในการออกแบบระบบฐานข้อมูลดาต้าแวร์เฮาส์

1.4.3 ศึกษารายละเอียดและประสิทธิภาพของทูลส์ ที่จะนำมาใช้ในการสร้างโปรแกรมประยุกต์ซึ่งทูลส์ที่นำมาใช้นั้นก็คือโปรแกรมวิชวลเบสิก รุ่น 6.0 เพื่อดูฟังก์ชันการทำงานและแนวคิดในการวิเคราะห์ ข้อมูลของโปรแกรม

1.4.4 ศึกษาระบบจัดการฐานข้อมูล SQL Server เพื่อให้ติดต่อกับโปรแกรมวิชวลเบสิก รุ่น 6.0ได้ และทำการสร้างระบบฐานข้อมูลดาต้าแวร์เฮาส์บนฐานข้อมูล SQL Server ได้

1.4.5 ทำการสร้างโปรแกรมประยุกต์ด้วยโปรแกรมวิชวลเบสิก รุ่น 6.0 เพื่อทำการติดต่อกับฐานข้อมูล SQL Server

1.4.6 ทำการออกแบบและเขียนโปรแกรมเพื่อสร้างฐานข้อมูลแบบมัลติไดเมนชันแนล

1.4.7 ทำการทดสอบประสิทธิภาพโปรแกรมประยุกต์ที่สร้างขึ้นมา ถ้ายังพบข้อผิดพลาด หรือไม่พอใจในประสิทธิภาพในการทำงานของโปรแกรม ก็แก้ไขจนกว่าจะสามารถใช้งานได้เป็นอย่างดี

บทที่ 2
ทฤษฎีเบื้องต้น

2.1 นิยามของดาต้าแวร์เฮาส์

ดาต้าแวร์เฮาส์ คือ การแยกฐานข้อมูลที่จะใช้งานกับระบบช่วยเหลือการตัดสินใจ ออกจากระบบฐานข้อมูลประจำวัน โดยฐานข้อมูลดาต้าแวร์เฮาส์อาจจะนำข้อมูลมาจากฐานข้อมูลประจำวันหรือมีการประมวลผลข้อมูลจากฐานข้อมูลประจำวันให้กลายเป็นข้อมูลสารสนเทศ (Informations) หรือกระทั่งนำข้อมูลอื่นๆ เพิ่มเติมเข้ามาก็ได้

นิยามของดาต้าแวร์เฮาส์ คือ ฐานข้อมูลที่ช่วยในการจัดหา สำหรับกระบวนการในการตัดสินใจ โดยฐานข้อมูลดังกล่าวจะต้องมีคุณสมบัติ ดังต่อไปนี้

2.1.1 Subject Oriented : ข้อมูลจะต้องถูกสร้างขึ้นมาจากหัวข้อ (subject) ธุรกิจที่สนใจ เช่น ถ้าบริษัทประกันภัยต้องการใช้ดาต้าแวร์เฮาส์ ฐานข้อมูลที่ได้จะต้องสร้างขึ้นจากประวัติลูกค้า เบี้ยประกันและการเรียกร้องแทนที่จะแยกตามชนิดของผลิตภัณฑ์ หรือบริการประกันภัย/ประกันชีวิต ข้อมูลที่สร้างขึ้นจะประกอบด้วยหัวข้อที่เก็บเฉพาะข่าวสารที่จำเป็นสำหรับกระบวนการตัดสินใจเท่านั้น

2.1.2 Integrated : ข้อมูลถูกรวบรวมจากแหล่งต่างๆ จากระบบปฏิบัติการ รูปแบบของข้อมูลแพลตฟอร์ม (Plat form) ที่หลากหลาย สร้างขึ้นเป็นฐานข้อมูลที่สอดคล้องเป็นหนึ่งเดียว เช่น ค่าของตัวแปรตัวเดียวกันในแต่ละฐานข้อมูลอาจต่างกัน ฐานข้อมูลหนึ่งอาจใช้ 0 และ 1 อีกฐานข้อมูลหนึ่งอาจใช้ T และ F ดังนั้นฐานข้อมูลที่สร้างขึ้นใหม่จะต้องได้รับการกำหนดค่าตัวแปรให้เหมือนกันเป็นหนึ่งเดียว

2.1.3 Time-variant : ข้อมูลซึ่งใช้ตัดสินใจที่เก็บไว้ จะต้องมีอายุประมาณ 5 ถึง 10 ปี เพื่อใช้เปรียบเทียบหาแนวโน้ม และทำนายผลลัพธ์ในอนาคตได้

2.1.4 Non-volatile : ข้อมูลจะไม่อัพเดท (update) หรือถูกทำให้เปลี่ยนแปลงได้ง่ายๆ ผู้ใช้สามารถใช้ฐานข้อมูลดาต้าแวร์เฮาส์ ได้เพียงแค่โหลด (load) และเข้าใช้ข้อมูล (access) เท่านั้น

2.2 จุดมุ่งหมายของดาต้าแวร์เฮาส์

เป้าหมายของการออกแบบดาต้าแวร์เฮาส์ คือ การแยกกลุ่มข้อมูลสารสนเทศที่ใช้ในการวิเคราะห์ทางธุรกิจออกจากฐานข้อมูลประจำวัน มาเก็บอยู่ใน RDBMS (Relational Database Management System) ประสิทธิภาพสูง และทำให้การเรียกใช้ข้อมูลชุดนี้ทำได้อย่างยืดหยุ่นจากทูลส์ที่อยู่บนเครื่องเดสก์ทอปทั่วไป โดยลดออฟโหลดดิ้ง เพิ่มกลไกการช่วยตัดสินใจ ปรับปรุงเวลาที่ตอบสนองรวดเร็วขึ้นอย่างมาก และผู้บริหารสามารถเรียกข้อมูลรายละเอียดที่จำเป็นที่ถูกเก็บมาก่อนหน้านี้ (historical data) มาใช้ช่วยให้การตัดสินใจทางธุรกิจแม่นยำขึ้น

เป้าหมายในการจัดทำดาต้าแวร์เฮาส์ มีดังต่อไปนี้

2.2.1 ดาต้าแวร์เฮาส์ทำให้สามารถเข้าถึงข้อมูลขององค์กรได้ ผู้จัดการและนักวิเคราะห์ขององค์กรสามารถเชื่อมต่อเข้าไปยังดาต้าแวร์เฮาส์จากเครื่องคอมพิวเตอร์ตนได้ ซึ่งการเชื่อมต่อนี้สามารถทำได้ในทันทีตามความต้องการและมีประสิทธิภาพสูง ทูลส์ที่มีให้กับผู้จัดการและนักวิเคราะห์สามารถใช้งานได้ง่าย

2.2.2 ข้อมูลในดาต้าแวร์แวร์เฮาส์จะต้องถูกต้องตรงกันหมด คำถามเดียวกันต้องได้รับคำตอบที่เหมือนกันเสมอ ไม่ว่าผู้ถามจะเป็นใคร ถามเวลาใดก็ตาม

2.2.3 ข้อมูลในดาต้าแวร์เฮาส์สามารถถูกตัดและหมุนดูได้ทุกแกน หมายถึง ข้อมูลสามารถถูกวิเคราะห์จากหัวข้อในธุรกิจประเภทนั้น โดยแบ่งข้อมูลหรือรวมข้อมูลมาวิเคราะห์ตามความต้องการได้

2.2.4 ร้อยละ 60 ของดาต้าแวร์เฮาส์ คือ ฮาร์ดแวร์ของดาต้าแวร์เฮาส์ส่วนกลาง (central data warehouse), ซอฟท์แวร์ฐานข้อมูลรีเลชันแนล (relational database software) และข้อมูลส่วนที่เหลืออีกร้อยละ 40 นั้นคือ กลุ่มของฟร้อนต์เอนด์ทูลส์ (set of front end tools) ที่ทำหน้าที่ในการป้อนคำถาม วิเคราะห์และแสดงข้อมูลออกมา

2.2.5 ดาต้าแวร์เฮาส์เป็นส่วนที่ผลิตข้อมูลจาก OLTP ข้อมูลไม่เพียงแต่ถูกรวบรวมมาไว้ที่ศูนย์กลางอย่างเดียว แต่จะถูกรวบรวมอย่างระมัดระวังจากแหล่งข้อมูลหลายๆแหล่งภายนอกองค์กรด้วย แล้วมาปรับปรุงให้เหมาะสมกับการใช้งานเท่านั้น ถ้าข้อมูลเชื่อถือไม่ได้หรือไม่สมบูรณ์ก็จะไม่ได้รับอนุญาตให้นำไปใช้ได้

2.2.6 คุณภาพของข้อมูลในดาต้าแวร์เฮาส์เป็นตัวผลักดันให้สามารถทำการรีเอ็นจิเนียริ่งธุรกิจได้

2.3 ระบบฐานข้อมูลแบบทรานแซคชั่น (OnlineTransaction Processing : OLTP)

ตามธรรมชาติของระบบฐานข้อมูลประจำวันจะเป็นระบบทรานแซคชั่น (Transaction System) แต่เนื่องจากทรานแซคชั่นมักจะต้องเรียกใช้ข้อมูลจากฐานข้อมูลขนาดใหญ่ เมื่อต้องการเรียกใช้ข้อมูลเพื่อช่วยในการตัดสินใจ ทำให้มีข้อมูลจำนวนมหาศาล ทรานแซคชั่นจึงมีปัญหาในการเรียกข้อมูลทรานแซคชั่น ดังต่อไปนี้

2.3.1 ข้อมูลทรานแซคชั่นเรียกใช้ได้ยาก

2.3.2 มีการแตกตารางที่นอร์มอลไลซ์แล้ว (Normalized tables) ออกเป็นหลายตาราง

2.3.3 ไม่รองรับคำถามที่สนับสนุนการตัดสินใจ เพราะคำถามทั้งหมดของระบบสนับสนุนการตัดสินใจมีการรวมกันของตารางต่างๆที่ซับซ้อน ซึ่งจะทำให้ประสิทธิภาพของ Transaction operation database ตกลงและทำงานช้าลงไป

2.3.4 มีข้อมูลย้อนหลังน้อย

ดังนั้นดาต้าแวร์เฮาส์จึงถูกคิดขึ้นมา เพื่อช่วยให้ผู้ใช้เรียกใช้ข้อมูลด้วยวิธีที่สร้างสรรค์ เพราะธรรมชาติที่แตกต่างกันของดาต้าแวร์เฮาส์และฐานข้อมูลประจำวัน โดยดาต้าแวร์เฮาส์จะแยกฐานข้อมูลออกมา เก็บข้อมูลสารสนเทศที่สรุปมาเฉพาะหัวข้อที่สนใจ เพื่อวิเคราะห์ใช้ในการบริหารและควบคุมธุรกิจ โดยระบบนี้พิจารณาถึงการใช้ประโยชน์จากข้อมูลเป็นหลัก

2.4 ความแตกต่างของดาต้าแวร์เฮาส์กับระบบฐานข้อมูลประจำวัน

ดาต้าแวร์เฮาส์และระบบฐานข้อมูลประจำวันนั้นมีความแตกต่างกันมากมาย ดังต่อไปนี้

2.4.1 **ความถูกต้อง (Consistency)**

ทั้ง OLTP และดาต้าแวร์เฮาส์ ต่างก็ให้ความสำคัญในเรื่องของความถูกต้องของข้อมูล (data consistency) สำหรับ OLTP ซึ่งมีการทำทรานแซคชั่นจำนวนมากๆนั้น สิ่งที่ต้องการคือ การทำทรานแซคชั่นให้ครบ (ไม่มีการสูญหาย) ดังนั้นจึงมีความจำเป็นที่ผู้ส่งและผู้รับจะต้องรับรู้และตรวจสอบอยู่ตลอดเวลา ว่าขณะนี้มีการทำทรานแซคชั่นเกิดขึ้นหรือไม่

สำหรับดาต้าแวร์เฮาส์จะไม่สนใจการทำทรานแซคชั่นแต่ละครั้ง แต่จะสนใจว่า การโหลดข้อมูลใหม่เข้ามานั้นทำเสร็จแล้วหรือยัง (โหลดข้อมูลเข้ามาทั้งหมดและข้อมูลถูกต้อง)

2.4.2 **ทรานแซคชั่น (Transaction)**

สำหรับระบบ OLTP นั้นในแต่ละวันอาจมีการทำทรานแซคชั่นเป็นหมื่นเป็นแสนครั้ง ซึ่งการทำทรานแซคชั่นแต่ละครั้งจะใช้ข้อมูลเพียงเล็กน้อยเท่านั้น

สำหรับดาต้าแวร์เฮาส์ แต่ละวันจะทำเพียงแค่ 1 ทรานแซคชั่น ซึ่งการทำทรานแซคชั่นนี้อาจต้องใช้ข้อมูลเป็นล้านเรคอร์ด (records) เลยก็ได้ ดังนั้นเราจึงเรียกกระบวนการนี้ว่า Production data load แทน สิ่งที่เราสนใจในกระบวนการนี้มีเพียงแค่ production data load เท่านั้น ถ้าการทำ production data load ถูกทำให้หยุดกลางคัน ระบบจะทำการเอาข้อมูลที่เคยมีมาก่อนที่ production data load จะทำ มาเขียนทับลงทันที

2.4.3 **ผู้ใช้และผู้จัดการ (User and Manager)**

สำหรับ OLTP นั้น ผู้ใช้ คือ ผู้ที่ทำงานอยู่ในองค์กรนั้น (อาจจะเป็นเจ้าหน้าที่ที่ทำหน้าที่รับออร์เดอร์, เจ้าหน้าที่ที่ทำหน้าที่ตรวจสอบและดูแลเงิน, เจ้าหน้าที่ที่ทำหน้าที่คอยบริการลูกค้าใหม่, เจ้าหน้าที่ที่ทำหน้าที่ป้อนข้อมูลเข้าไป) ซึ่งส่วนใหญ่ผู้ใช้เหล่านี้ ในช่วงเวลาหนึ่งจะทำงานกับ 1 แอคเคาท์ (account) เท่านั้น ผู้ใช้ OLTP มักจะทำงานที่ลักษณะเป็นงานซ้ำๆ เดิมๆ รายงานส่วนใหญ่ที่ได้จากการทำงานบนระบบ OLTP นี้มักจะมีลักษณะเป็นลิสต์ (List) ของทั้งตารางเลย

สำหรับดาต้าแวร์เฮาส์ ผู้ใช้คือผู้ที่ทำหน้าที่คอยดูแลการทำงานของพนักงานในองค์กรเท่านั้น (คอยนับจำนวนออร์เดอร์ใหม่ๆ , หาเหตุผลว่าทำไมลูกค้าจึงไม่พอใจ, คอยตรวจดูว่ามีข้อมูลอะไรใหม่ๆ เข้ามาบ้าง, คอยตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูล) ผู้ใช้ของดาต้าแวร์เฮาส์จะไม่ทำงานทีละ 1 แอคเคาท์ แต่จะพิจารณาจากแอคเคาท์ทั้งหมด แล้วหาคำตอบที่ต้องการออกมา (Answer set ขนาดเล็กๆ) และคำถามที่ให้กับดาต้าแวร์เฮาส์ก็สามารถเปลี่ยนแปลงได้เรื่อยๆ (ไม่ใช่คำถามเดิมที่เคยถามก็ได้)

2.4.4 **The Time Dimension**

OLTP จะทำงานอย่างรวดเร็วและทำทรานแซคชั่นอย่างสม่ำเสมอ (การวัดเวลาใช้หน่วยเป็นนาทีและวินาที) สถานะของข้อมูลต่างๆ มีการเปลี่ยนแปลงอยู่ตลอดเวลา และความสัมพันธ์ระหว่างเอนติตี้ (entity) ต่างๆ ก็เปลี่ยนแปลงไปด้วย

ฐานข้อมูลใน OLTP จะขาดการสนับสนุน (การอ้างอิง) จากข้อมูลในอดีต เพราะในดาต้าแวร์เฮาส์มักจะมีคำถามถึงการวิเคราะห์ข้อมูลในอดีต แม้ว่าเราสามารถเก็บข้อมูลอดีตไว้ใน OLTP ได้ แต่ก็เป็นภาระหนักของระบบในการทำให้มองเห็นภาพในอดีต และจะทำให้ยากในการทำทรานแซคชั่น (ข้อมูลมีมากขึ้น ก็ต้องใช้เวลามากขึ้นด้วย)

" Data Warehouse is a time series "

ดาต้าแวร์เฮาส์จะต้องไม่ถูกเปลี่ยนแปลงตลอดวัน ดังนั้น Twinkling ก็จะไม่มีทางเกิดขึ้นและมีการระมัดระวังในการเก็บข้อมูลลงไปในดาต้าแวร์เฮาส์แต่ละครั้งด้วย

2.4.5 **การสร้างข้อมูล (Data Model)**

ความแตกต่างที่สำคัญที่สุดของ OLTP และดาต้าแวร์เฮาส์ คือ การสร้างข้อมูลหรือ Data Model นั่นเอง

OLTP จะใช้ Entity Relation Modeling คือ การกำจัดความซ้ำซ้อนให้หมดไป เพื่อการทำทรานแซคชั่นจะได้สามารถทำได้ง่ายและรวดเร็วยิ่งขึ้น (สามารถทำทรานแซคชั่นได้โดยเข้าถึง ข้อมูลเพียงตัวเดียวเท่านั้น) การทำ

entity relation modeling นั้น จะใช้วิธีแยกข้อมูลออกเป็นตารางเล็กๆ และแต่ละตารางสามารถเชื่อมต่อไปยังตารางอื่นได้

ข้อสังเกตเกี่ยวกับ OLTP ซึ่งจะใช้ entity relation diagram

- ไดอะแกรมนี้ซิมเมทริก (symmetric) หมายถึง ทุกๆ ตารางเหมือนกันหมด ไดอะแกรมนี้ไม่สามารถบอกได้ว่าตารางไหนสำคัญกว่าหรือใหญ่กว่า และไม่สามารถบอกได้ว่า ตารางใดบรรจุตัววัดที่เป็นชนิดตัวเลข (Numerical measurement) ทางธุรกิจ ซึ่งไดอะแกรมรูปแบบนี้ ผู้ใช้จะทำความเข้าใจและจดจำได้ยาก
- ถ้ามี 2 ตารางในไดอะแกรมต้องการที่จะรวมกันก็มีหลายวิธีที่จะทำได้และไม่ว่าจะวิธีไหนก็จะได้คำตอบเดียวกัน ซึ่งเป็นเรื่องยากในการเลือกว่าจะใช้เส้นทาง (path) ใดในการเชื่อม (link) ระหว่างตารางที่ต่างกัน เพราะถึงแม้ท้ายสุดจะได้คำตอบเดียวกัน แต่ในระหว่างการทำ Inner join นั้นจะมีการใช้ data element ไม่เหมือนกัน

สำหรับดาต้าแวร์เฮาส์จะใช้ไดเมนชันแนลโมเดล (Dimensional Model) หรือ Star Join Schema ซึ่งเป็นชื่อที่นักออกแบบฐานข้อมูลใช้กันมานานแล้ว เนื่องจากไดอะแกรมมีรูปร่างคล้ายดาว ซึ่งมีตารางหลัก 1 ตาราง อยู่ตรงกลางและมีตารางเล็กๆ ที่มีความสัมพันธ์กับตารางหลักนั้นอยู่รอบๆ ซึ่งตารางหลักนี้เป็นตารางเดียวที่ใช้ Multiple join เพื่อเชื่อมต่อกับตารางอื่นๆ แต่ตารางอื่นๆ ที่อยู่รอบๆ นั้นจะมีเพียงแค่ Single join เพื่อเชื่อมเข้ากับตารางหลักเท่านั้น โดยตารางหลัก เรียกว่าตารางแฟ็ค (Fact table) ส่วนตารางอื่นๆ จะเรียกว่าตารางไดเมนชัน (Dimension table)

2.4.6 **The standard template query**

ดาต้าแวร์เฮาส์จะใช้ SQL เป็น template มาตรฐานสำหรับการตั้งคำถามทั้งหมดในดาต้าแวร์เฮาส์ ซึ่งเกี่ยวกับตารางแฟ็ค

- Select list : จะเลือกคอลัมน์ (column) ที่ต้องการให้ปรากฏในเซตคำตอบ (Answer set) ของผู้ใช้ไว้ ซึ่งปกติแล้วใน Select list เราจะจัด row header ไว้เป็นไอเท็ม (item) แรกใน Select list
- From clause : จะได้มาจากชื่อของตารางที่จะต้องใช้ในคำถาม
- Where clause : การทำเงื่อนไขของการรวม (join constrain) ซึ่งในคำสั่งนั้นจะเขียนถึงความสัมพันธ์ระหว่างตารางแฟ็ค และตารางไดเมนชัน โดยคีย์หลัก (primary key) ของตารางไดเมนชันจะมีสถานะเป็น foreign key ของตารางแฟ็ค และหลังจากที่ทำการรวม จะพบว่ามีการทำ Application constrain สำหรับไดเมนชัน การสร้าง Application constrain ทำได้โดยพิจารณาแต่ละไดเมนชัน ดูว่ามีค่าอะไรบ้างที่อยู่ในขอบเขตที่ต้องการ
- Group by clause : บอก SQL ให้สรุปข้อมูลตาม row header ซึ่ง row header นี้มีลิสต์อยู่ใน Select list นั่นเอง
- Order by clause : ทำการตัดสินใจว่าจะทำการเรียงลำดับค่าในเซตคำตอบให้ผู้ใช้เห็นอย่างไร

2.5 ความแตกต่างของดาต้าแวร์เฮาส์กับ Operational Data Stores

ดาต้าแวร์เฮาส์นั้นแตกต่างจากระบบฐานข้อมูลปฏิบัติการหลายส่วน ส่วนหนึ่งที่แตกต่างกันก็คือ ข้อมูลที่จัดเก็บ ในระบบปฏิบัติการ (OLTP System) จะเรียกข้อมูลว่า " ข้อมูลเชิงปฏิบัติการ " หรือ " Operational Data " ส่วนในดาต้าแวร์เฮาส์จะเรียกข้อมูลว่า " ข้อมูลที่ช่วยในการตัดสินใจ " หรือ " Decision Support Data " ความแตกต่างของทั้งสองระบบนี้แสดงในตารางที่ 2.1

Operational Data	Decision Support Data
จัดการข้อมูลระหว่างแอพลิเคชันกับแอพลิเคชัน (Application Oriented)	จัดการข้อมูลตามหัวข้อเรื่องที่ต้องการ (Subject Oriented)
เป็นข้อมูลแบบบอกรายละเอียด	เป็นข้อมูลแบบสรุป
ข้อมูลสามารถเปลี่ยนแปลงได้ (Volatile)	ข้อมูลเปลี่ยนแปลงไม่ได้ (Non – volatile)
ข้อมูลมีการเคลื่อนไหวตลอดเวลา	ข้อมูลจะคงที่จนกว่าจะมีการปรับปรุงใหม่
ประสิทธิภาพเป็นสิ่งสำคัญ เพราะต้องรองรับการเข้าใช้งานพร้อมๆ กันของผู้ใช้จำนวนมาก	ประสิทธิภาพยืดหยุ่นกว่า เพราะในเวลาเดียวกันจะมีผู้ใช้ค่อนข้างน้อยเข้าใช้ข้อมูล
ใช้ข้อมูลในปัจจุบัน	ใช้ข้อมูลในช่วงเวลาที่ผ่านมาแล้ว (Historical Data)
แหล่งข้อมูลมาจากภายในองค์กร	แหล่งข้อมูลมาจากทั้งภายในและภายนอกองค์กร
ขนาดข้อมูลประมาณกิกะไบต์	ขนาดข้อมูลอยู่ระหว่างกิกะไบต์ถึงเทราไบต์
การประมวลผลแต่ละครั้ง มักมีข้อมูลไม่มาก เป็นเพียงยูนิตหรือเรคอร์ด	การประมวลผลครั้งหนึ่งๆ จะมีข้อมูลจำนวนมาก เช่น หลายๆ เรคอร์ด (Set of records)
ใช้เวลาประมวลผลน้อย เพียงไม่กี่วินาที	ใช้เวลาประมวลผลไม่แน่นอน ตั้งแต่วินาทีจนถึงนาที

Subject Oriented : ข้อมูลสารสนเทศสำหรับการทำธุรกิจ จะถูกจัดวางตามหัวข้อที่องค์กรนั้นสนใจ เช่น หัวข้อของสินค้าอุปโภคและบริโภคอาจจะเป็นผลิตภัณฑ์, ลูกค้า, สื่อโฆษณาและเวลา ซึ่งข้อมูลจากดาต้าแวร์เฮาส์นั้นจะมาจากแหล่งข้อมูลที่ยืนยันว่าถูกต้องแล้ว

Application Oriented : เช่น แอพลิเคชันด้านการสั่งซื้อสินค้า ฐานข้อมูลก็จะรวมเอาข้อมูลมาใส่ในแบบฟอร์มที่มีอยู่

ข้อมูลในอดีต : เป็นสิ่งที่ระบบฐานข้อมูลประจำวันใช้น้อยมาก เพราะมีจุดประสงค์เพื่อจับความเป็นไปในปัจจุบันมากกว่า แต่ในทางธุรกิจจะใช้ข้อมูลในอดีตนี้ เพื่อหาความเป็นไปและแนวโน้มในอนาคตได้

ข้อมูลคงที่ : เนื่องจากการวิเคราะห์และคำนวณข้อมูลทางธุรกิจ จะทำได้ต่อเมื่อข้อมูลมีปริมาณคงที่เท่านั้น แต่ข้อมูลจากฐานข้อมูลประจำวันมีการเคลื่อนไหวตลอดเวลา (Dynamic) ซึ่งเป็นเหตุผลหนึ่งที่ทำให้ข้อมูลในดาต้าแวร์เฮาส์ไม่มีการอัพเดท (Update) โดยตรง แต่จะใช้ข้อมูลที่ประมวลผลใหม่ ซ้อนทับขึ้นไปเรื่อยๆ

ตารางที่ 2.1 สรุปความแตกต่างของดาต้าแวร์เฮาส์กับระบบฐานข้อมูลปฏิบัติการ

2.6 ความแตกต่างของระบบจัดการฐานข้อมูลแบบมัลติไดเมนชันแนลและรีเลชันแนล

2.6.1 การตอบคำถามของระบบจัดการฐานข้อมูลแบบมัลติไดเมนชันแนล จะสามารถทำได้ง่ายกว่าแบบรีเลชันแนล

2.6.2 ระบบจัดการฐานข้อมูลแบบมัลติไดเมนชันแนล ใช้ดูแนวโน้มของแอททริบิวท์(Attribute) ว่าค่าของแอททริบิวต์นั้นมีทิศทางการเปลี่ยนแปลงในช่วงเวลาที่เราสนใจเป็นอย่างไร เพราะฐานข้อมูลแบบไดเมนชันแนลสามารถสรุปออกมาให้อยู่ในรูปแบบที่เข้าใจได้ง่ายและรวดเร็วกว่า เช่น ในการค้นหาที่มีการแสดงผลข้อมูลเกินกว่า 1 หน้าจอ ผู้ใช้ที่ใช้ฐานข้อมูลแบบรีเลชันแนล จะรู้สึกไม่สะดวกในการดูแนวโน้มของข้อมูลเท่ากับการใช้ฐานข้อมูลแบบมัลติไดเมนชันแนล (เพราะต้องทำการเลื่อนหน้าจอ)

2.6.3 ระบบจัดการฐานข้อมูลแบบรีเลชันแนล จะแสดงข้อมูลระดับเล็กที่สุด ไม่สามารถย่อยได้อีก (Atomic data) และเป็นข้อมูล ณ เวลาใดเวลาหนึ่ง (record at a time) แต่ระบบจัดการฐานข้อมูลแบบมัลติไดเมนชันแนล จะแสดงข้อมูลโดยรวม ซึ่งเน้นผลสรุปของข้อมูลและมีประโยชน์ในการวิเคราะห์ข้อมูลนั้นๆ

2.6.4 ฐานข้อมูลแบบรีเลชันแนล จะทำการนอร์มอลไลซ์ (Normalizeation) ซึ่งการนอร์มอลไลซ์นั้นเป็นขั้นตอนของการออกแบบฐานข้อมูล โดยใช้เทคนิคดีคอมโพซิชัน (Decomposition) ซึ่งจะแตกแอททริบิวต์ของ Unsatisfactory Relation Schema ไปเป็น Relation Schema ที่เล็กกว่า ซึ่งมีคุณสมบัติตามที่ต้องการตาม Normal form ขั้นต่างๆ

กระบวนการของการนอร์มอลไลซ์ซึ่งผ่านการแยกส่วน (Decomposition) จะต้องมีคุณสมบัติดังนี้

- Lossless join หรือ Nonadditive join property เป็นคุณสมบัติที่รับประกันว่าจะไม่เกิดปัญหา Spurious Tuple ขึ้น
- Dependency preservation property เมื่อมีการอัพเดทฐานข้อมูล ระบบควรจะสามารถตรวจสอบได้ว่าการอัพเดทไม่ได้สร้าง illegal relation ขึ้น

Normal Form (NF) ตั้งแต่ขั้นที่ 1 ถึงขั้นที่ 3 และ Boyce – Codd อ้างอิงอยู่บน data dependency แบบ functional dependencies ระหว่างแอททริบิวท์ของรีเลชัน ดังนี้

1NF : A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. หมายความว่า ความสัมพันธ์ R จะอยู่ใน normal form ขั้นที่ 1 ได้ก็ต่อเมื่อแอททริบิวท์ทุกตัวในโดเมนมีค่าเพียงค่าเดียวเท่านั้น

2NF : A relation R is in second normal form (2NF) if and only if it is in 1NF and every nonkey attributes is fully dependent on the primary key. หมายความว่า ความสัมพันธ์ R จะอยู่ใน normal form ขั้นที่ 2 ได้ก็ต่อเมื่อ ความสัมพันธ์ R อยู่ใน normal form ขั้นที่ 1 แล้วและทุกๆ nonkey attributes จะขึ้นอยู่กับคีย์หลัก (ไม่ขึ้นกับสับเซตคีย์หลัก)

3NF : A relation R is in third normal form (3NF) if and only if it is in 2NF and every nonkey attributes is nontransitively dependent on the primary key. หมายความว่า ความสัมพันธ์ R จะอยู่ใน normal form ขั้นที่ 3 ได้ก็ต่อเมื่อ ความสัมพันธ์ R อยู่ใน normal form ขั้นที่ 2 แล้วและทุกๆ nonkey attributes จะไม่ขึ้นอยู่กันเอง

Boyce – Codd Normal Form (BCNF) : A Relation R is in Boyce – Codd Normal Form if and only if every determinant is a candidate key. หมายความว่า ความสัมพันธ์ R จะอยู่ใน BCNF ก็ต่อเมื่อ determinant ทุกๆ ตัวเป็น candidate key

จุดประสงค์ของการทำนอร์มอลไลซ์ คือ เพื่อสร้างแบบจำลองความสัมพันธ์ (Relation Schema) ซึ่งเก็บข้อมูลโดยปราศจากความซ้ำซ้อน (Redundancy) ที่ไม่จำเป็น เพราะฐานข้อมูลที่ดีควรจะมีความซ้ำซ้อนน้อยที่สุด เนื่องจากความซ้ำซ้อนมีผลเสีย คือ สิ้นเปลืองเนื้อที่, สิ้นเปลืองค่าใช้จ่ายในการป้อนข้อมูล (data entry cost) และทำให้เกิดความซับซ้อนในการอัพเดท ซึ่งนำไปสู่ปัญหา update anomalies ดังต่อไปนี้

- Insert Anomalies
  - insert fact ซึ่งขัดแย้งกับ fact เดิม
  - insert fact บาง fact ไม่ได้

- Deletion Anomalies
  - การลบ fact หนึ่งออก อาจทำให้ fact อื่นถูกลบไปด้วย
- Modification Anomalies
  - เนื่องจาก fact ซ้ำซ้อน เมื่อเปลี่ยนแปลงค่าในแอททริบิวท์หนึ่งเราจะต้องอัพเดทหลายที่ (Multiple update) ให้ครบ ซึ่งอาจทำให้ข้อมูลเกิดความขัดแย้ง (Inconsistent) กันได้

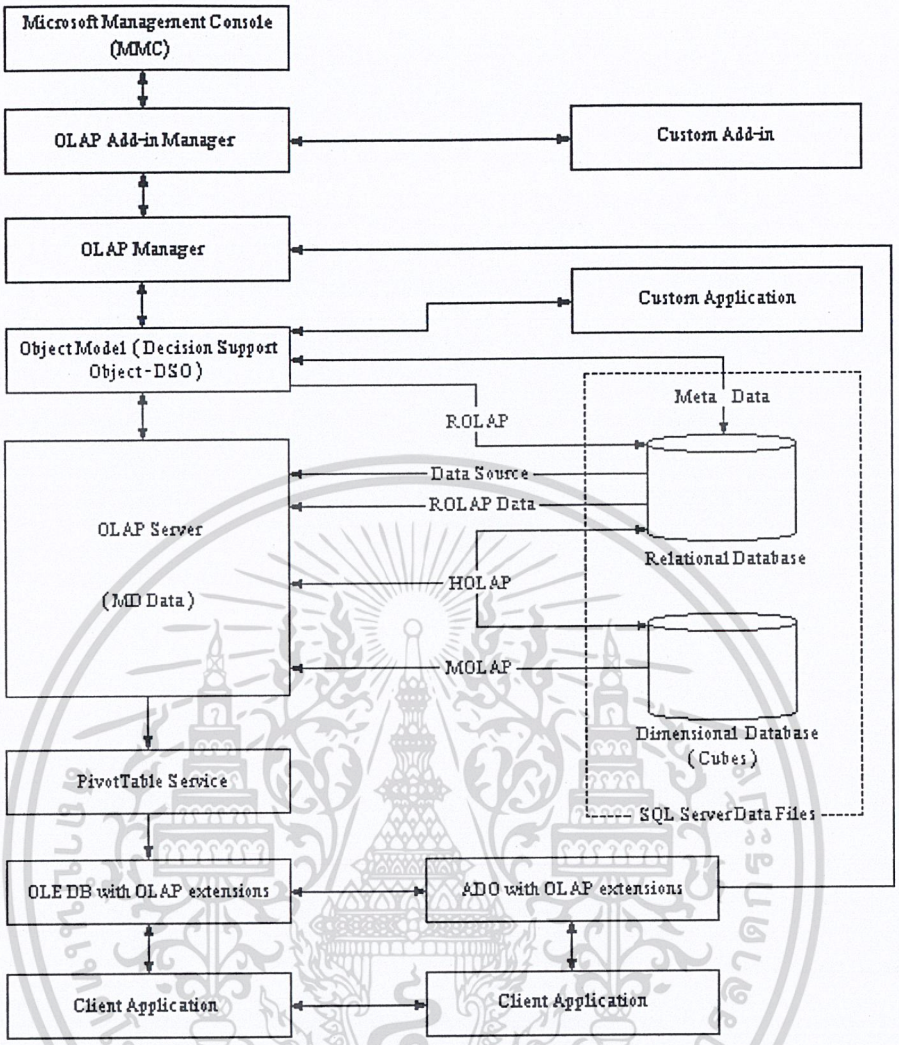
แต่สำหรับดาต้าแวร์เฮาส์นั้น ข้อมูลจะเป็นข้อมูลที่ได้รับการกลั่นกรองมาแล้ว เพื่อใช้ในการวิเคราะห์และตอบคำถามของผู้บริหาร ประเด็นสำคัญจึงไม่ได้อยู่ที่การอัพเดทข้อมูล จึงทำให้ข้อมูลในดาต้าแวร์เฮาส์สามารถมีความซ้ำซ้อนได้ เพราะความซ้ำซ้อนก็มีข้อดีสำหรับการตอบคำถามและการสร้างรายงาน คือ สามารถทำได้เร็ว เนื่องจากไม่ต้องมีการรวมกันของหลายตาราง ดังนั้นในดาต้าแวร์เฮาส์จึงไม่มีความจำเป็นต้องทำการนอร์มอลไลซ์

2.7 OLAP Services กับดาต้าแวร์เฮาส์ในระบบจัดการฐานข้อมูล SQL Server

ในระบบจัดการฐานข้อมูล SQL Server จะมีบริการ (Services) ต่างๆ ให้ใช้งาน บริการอันหนึ่งที่ใช้สำหรับระบบดาต้าแวร์เฮาส์ ก็คือ OLAP Services หรือเรียกสั้นๆ ว่า OLAP ซึ่งย่อมาจาก On-line Analytical Processing มักจะใช้ในการแปลงข้อมูลดิบให้กลายเป็นข้อมูลสารสนเทศ (Information) และความรู้ (Knowledge)

ดาต้าแวร์เฮาส์จะเป็นส่วนที่ใช้เก็บข้อมูลสารสนเทศและความรู้เหล่านั้น ซึ่ง OLAP อาจจะเข้าไปเรียกใช้เพื่อทำการวิเคราะห์ ในการประมวลผล ข้อมูลจะถูกเกลา (Scrubbed) และถูกแปลง (Transformed) เพราะข้อมูลดิบในระบบเชิงปฏิบัติการ (Operational System) อาจจะไม่อยู่ในรูปแบบที่สามารถนำไปสู่การจัดเก็บที่มีประสิทธิภาพในดาต้าแวร์เฮาส์หรือประสิทธิภาพในการวิเคราะห์โดยทูลส์ของ OLAP

โครงสร้างของสถาปัตยกรรมของ SQL Server data warehousing และ OLAP services จะแบ่งการทำงานออกเป็นหลายส่วนด้วยกัน เช่น ส่วนของแหล่งข้อมูล (Data Source), ส่วนของการแปลงและส่งข้อมูล (Data Transformation and Export), ส่วนของการจัดเก็บข้อมูล (Data Storage), ส่วนของการวิเคราะห์ข้อมูล (Data Analysis) และส่วนของการนำเสนอข้อมูล (Data Presentation) ซึ่งแสดงส่วนต่างๆ ทั้งหมดได้ดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างของสถาปัตยกรรมของ SQL Server Data Warehousing and OLAP

2.7.1 จุดเด่นของ OLAP Services

- **ใช้งานง่าย (Ease to use) :** OLAP Services จะใช้งานง่าย เพราะจะมีทูลส์ให้ใช้งานหลายรูปแบบ เช่น วิซาร์ด (Wizard), อิดิเตอร์ (Editor) และ Help คอยช่วยเหลือ โดยเฉพาะในส่วนของ OLAP Manager User Interface จะเป็นทูลส์ที่ดีในการเข้าใช้และเรียกดูข้อมูลแบบ Metadata และ Cube data อีกทั้งยังมีวิซาร์ดในการสร้างและแก้ไข Cubes, ไดเมนชัน (Dimensions) และระดับ (Levels) ด้วย
- **มีความยืดหยุ่น (Flexibility) :** OLAP Server จะมีการเก็บข้อมูลได้หลายรูปแบบ คือ แบบ Multidimensional OLAP (MOLAP), Relational OLAP (ROLAP) และ Hybrid OLAP (HOLAP) และยังสามารถทำ Cube Partitioning ได้อีกด้วย
- **Scalability :** OLAP Client สามารถทำงานได้ทั้งในวินโดวส์ 9x (Windows 9x), วินโดวส์ NT (Windows NT) และวินโดวส์ 2000 (Windows 2000)

- **Integration :** OLAP Services จะถูกรวมไว้กับ Microsoft Management Console (MMC), SQL Server และวินโดวส์ NT ซึ่งจะมีการใช้ระบบรักษาความปลอดภัยร่วมกัน (Shared Security) และใช้ OLE DB data source เป็นตัวทำหน้าที่ติดต่อ (Provider)
- **Widely suuport APIs and functions :** OLAP Server และ PivotTable Services จะรองรับ OLE DB, ActiveX Data Objects (ADO), User-defined functions และ Decision Support Objects (DSO)

บทที่ 3
โครงสร้างและการออกแบบของดาต้าแวร์เฮาส์

3.1 Dimensional Modeling

Dimensional Modeling เป็นเทคนิคที่ใช้ในการมองรูปแบบข้อมูล เหมือนเป็นกลุ่มของตัววัด ซึ่งใช้อธิบายในเชิงธุรกิจทั่วไป และเป็นสิ่งจำเป็นที่จะต้องใช้ในการหาค่ารวมและแสดงข้อมูลเมื่อทำการวิเคราะห์ ข้อมูลที่นำมาใช้จะเป็นข้อมูลชนิดที่เป็นตัวเลข เช่น values, counts, weights, balances ในดาต้าแวร์เฮาส์นั้นมักนิยมใช้ Dimensional Modeling มากกว่าที่จะใช้ ER Modeling เพราะมีความชัดเจนและเข้าใจง่ายกว่า

Dimensional Modeling จะนิยามพื้นฐาน 3 อย่างคือ แฟ็ค (Fact), ไดเมนชัน (Dimension) และตัววัด (Measure)

- แฟ็ค (Fact)

ทำหน้าที่เก็บข้อมูลที่มีความสัมพันธ์กัน ประกอบไปด้วยตัววัดและตัวข้อมูล ซึ่งแสดงถึงข้อมูลทางธุรกิจ, การดำเนินการทางธุรกิจหรือเหตุการณ์ (event) ที่สามารถใช้ในการวิเคราะห์หรือประมวลผลทางธุรกิจ ใน OLAP Services ส่วนของแฟ็ค จะอยู่ในรูปของตารางหลักของฐานข้อมูล OLAP ซึ่งจะเก็บข้อมูลที่เป็นตัวเลข ตารางเหล่านี้จะหมายถึงตารางแฟ็ค (Fact table) และมักจะได้รับมาจากข้อมูลดิบในฐานข้อมูลเชิงปฏิบัติการ

ข้อมูลในตารางแฟ็ค ควรจะมีลักษณะ คือ เป็นตัวเลข (Numeric), มีค่าต่อเนื่อง (Continuously valued) และมาจากทุกๆ ตาราง ไดเมนชัน (Additive)

- **Numeric :** เหตุผลที่ต้องเป็นตัวเลข ก็คือ ในความเป็นจริงคำถาม (query) ทุกคำถามที่ทำกับตารางแฟ็คนั้น ระบบจัดการฐานข้อมูล (DBMS) จะใช้เรคอร์ด (records) เป็นพันๆหมื่นๆ หรือเป็นล้านๆ เรคอร์ด เพื่อสร้างเซตคำตอบ (Answer set) ขึ้นมา จำนวนเรคอร์ดที่มากมายนี้จะถูกบีบอัด (compressed) เป็นจำนวนแถว (row) เพียงไม่กี่แถว ในเซตคำตอบของผู้ใช้นั่นก็คือ การทำคำสั่ง Select list หรือ Built-in function : SUM ของ SQL นั่นเอง วิธีเดียวที่จะบีบอัดเรคอร์ดเหล่านี้ให้กลายเป็นเซตคำตอบได้ ก็คือ การบวกค่าของเรคอร์ดต่างๆ เข้าด้วยกัน ดังนั้นเรคอร์ดที่มีลักษณะเป็นตัวเลขและ additive จะทำให้สร้างเซตคำตอบได้ง่าย

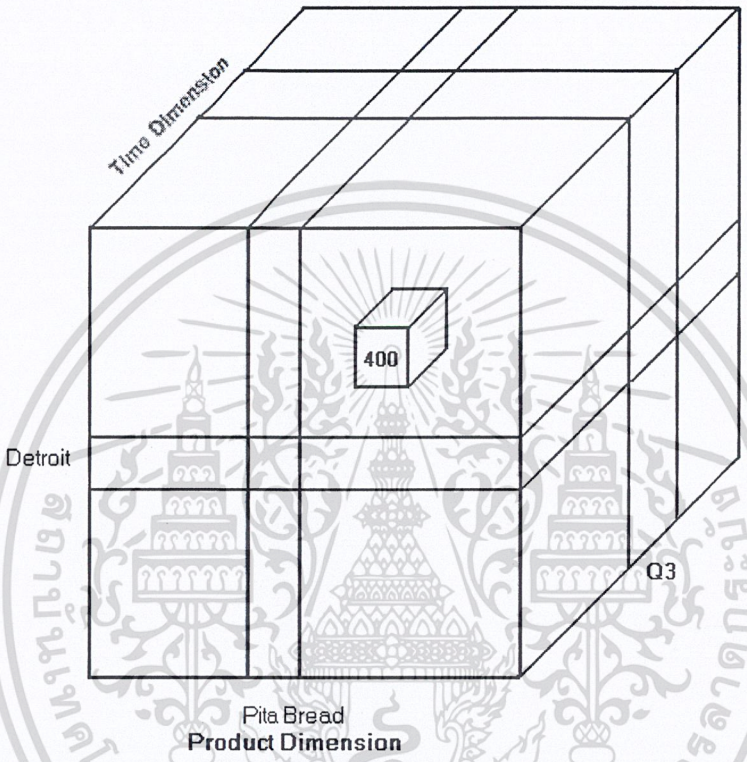
- **Continuously valued :** ค่าในตารางแฟ็ค ควรจะเป็นค่าที่มีความต่อเนื่อง เพื่อที่จะสามารถนำทางให้กับผู้ออกแบบฐานข้อมูล ให้สามารถแยกได้ว่าอะไรคือแฟ็ค อะไรคือแอททริบิวต์ของตารางไดเมนชัน

- **Additive :** ค่าของตัววัดในตารางแฟ็ค จะเป็นค่า additive แต่ก็จะมีค่าแฟ็ค ที่ไม่เป็น additive ด้วย เป็นแฟ็คที่เป็น Semiadditive หรือ Nonadditive ซึ่ง Semiadditive คือ เกรน (grain) ที่ไม่ได้มาจากทุกตารางไดเมนชัน ซึ่งจะสามารถหาผลรวมตามไดเมนชันบางไดเมนชันเท่านั้น ส่วน Nonadditive คือ เกรนที่ไม่สามารถหาผลรวมตามไดเมนชันใดๆ ได้

- ไดเมนชัน (Dimension)

ทำหน้าที่เก็บคำอธิบายของแต่ละไดเมนชันของธุรกิจเอาไว้ ซึ่งคำอธิบายเหล่านี้จะช่วยในการอธิบายถึงสมาชิกในทุกๆ ไดเมนชันและในตารางไดเมนชันจะประกอบด้วยหลายๆ แอททริบิวท์ ซึ่งแอททริบิวท์ที่ดีจะต้องเป็นตัวอักษร และแต่ละแอททริบิวท์ต้องแยกออกจากกัน ซึ่งแอททริบิวท์เหล่านี้ถูกใช้เป็นแหล่งที่มาของข้อบังคับ (Constrains) และ row header ในเซตคำตอบของผู้ใช้

ข้อมูลในไดเมนชันจะแตกต่างจากในตารางแฟ็ค เช่น ยอดขายรวม (total sales) สามารถมองในเทอมของ Store, City หรือ Region ได้ ทั้ง 3 ระดับนี้จะเป็นสมาชิกของไดเมนชันใน Dimensional Model ข้อมูลทุกตัวในตารางแฟ็ค จะมีความสัมพันธ์กับสมาชิกในหลายๆ ไดเมนชัน จะเห็นได้ชัดจากรูปที่ 3.1 มีเซลล์ (Cell) ที่มีค่า 400 ซึ่งเกิดจากความสัมพันธ์ในไดเมนชันที่ประกอบไปด้วย Q3 (3rd quarter), Pita Bread และ Detroit



รูปที่ 3.1 แสดงไดเมนชันใน Dimensional Modeling

- **ตัววัดค่า (Measure)**

เป็นแอททริบิวต์ที่เป็นตัวเลขของแฟ็ค แสดงประสิทธิภาพหรือพฤติกรรมของธุรกิจที่เกี่ยวข้องกับไดเมนชัน ซึ่งสมาชิกที่แท้จริง จะเรียกว่า ตัวแปร (Variables)

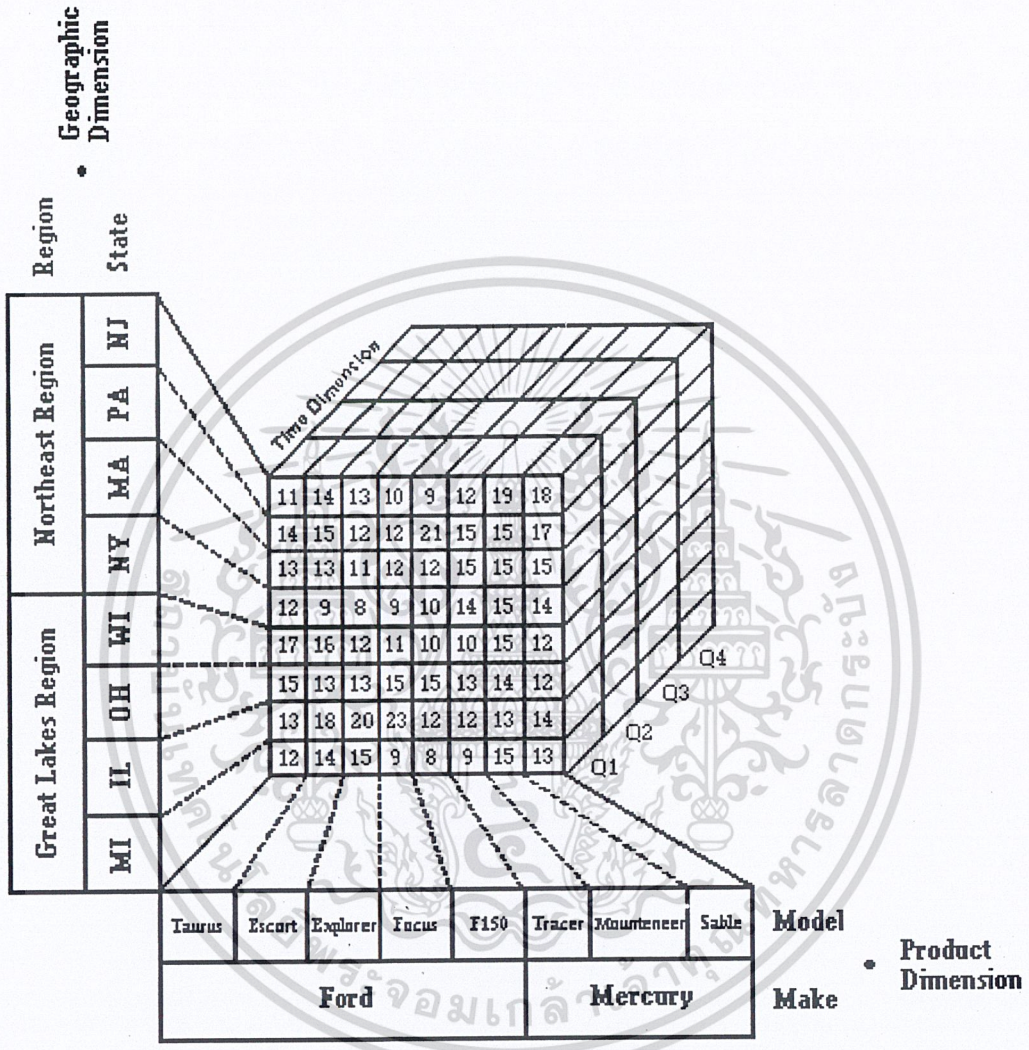
3.2 การออกแบบข้อมูล (Data Modeling)

การออกแบบข้อมูล เป็นขั้นตอนที่สำคัญในการสร้างดาต้าแวร์เฮาส์ เพราะการออกแบบจะทำให้สามารถมองระบบที่กำลังจะสร้างได้ชัดเจนยิ่งขึ้น

3.2.1 Data Cubes

ในดาต้าแวร์เฮาส์นั้น ข้อมูลจะถูกสร้างให้อยู่ในรูปแบบที่ง่ายต่อการมองและวิเคราะห์ โดยจะสร้างเป็นแบบลูกบาศก์ (Cube) โดยส่วนใหญ่แล้วมักจะเป็น 3 มิติหรือมากกว่า 3 มิติ (มากกว่า 3 มิติ จะเรียกว่า Hypercube) ตามรูป

ที่ 3.2 เป็นการวัดปริมาณยอดขาย ซึ่งประกอบไปด้วย 3 ไดเมนชัน คือ Geography, Product และ Time โดยไดเมนชัน Geography และ Product จะมีระดับ (level) ย่อยไปอีก ในไดเมนชัน Geography จะมีระดับ region และ state อีก ส่วนในไดเมนชัน Product จะมีระดับของ make และ model อีก ในส่วนของไดเมนชัน Time ก็จะมีระดับของ quarter ด้วย



รูปที่ 3.2 แสดง Data Cubes

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 3.1

	Quarter	Ford	Mercury
ABC-Autos	Q1 - 1998	120	89
	Q2 - 1998	123	91
	Q3 - 1998	101	75
	Q4 - 1998	99	73
Drill - Up Drill - Down			
	Month	**Ford**	**Mercury**
ABC-Autos	Oct - 1998	40	31
	Nov - 1998	35	29
	Dec - 1998	24	13

ตารางที่ 3.1 แสดงการ Drill-down และ Drill-up

จากตารางที่ 3.1 เป็นข้อมูลแสดงการขายรถยนต์ของตัวแทนจำหน่าย ซึ่งจะแบ่งเป็น 4 quarters ด้วยกัน เมื่อทำการ Drill-down ใน quarter ที่ 4 (Q4-1998) แล้วจะเห็นว่าจะเป็นการแสดงข้อมูลในระดับของเดือนใน Q4 (เดือนตุลาคมถึงเดือนธันวาคม) ส่วนการ Drill-up จะแสดงผลตรงกันข้ามกับการ Drill-down นั่นเอง เช่น ถ้า Drill-up ข้อมูลในระดับเดือน ก็จะเห็นข้อมูลแสดงในระดับของ quarter

- **Slicing และ Dicing**

การตัดข้อมูล (Slicing) เวลาที่ทำการวิเคราะห์ข้อมูล สามารถที่จะวิเคราะห์ข้อมูลได้เพียงบางช่วงหรือเฉพาะส่วนที่สนใจเท่านั้น ส่วนการหมุนข้อมูล (Dicing) ก็สามารถหมุนหรือสับเปลี่ยนตำแหน่งของไดเมนชันได้ ดังตารางที่ 3.2 , 3.3 และ 3.4

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 3.2 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 3.3 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 3.4 แสดงการตัดข้อมูลจากข้อมูลในตารางที่ 3.3

จากตารางข้างต้น ประกอบไปด้วย 3 ไดเมนชัน คือ Geographic location, Time และ Product (รถยนต์) ในตารางที่ 3.2 เป็นการตัดข้อมูลมาวิเคราะห์เฉพาะรถยนต์ยี่ห้อ Ford และ Mercury ใน 4 เมืองทางเขต Midwest และ Northeast ในปี 1998 ในตารางที่ 3.3 แสดงการหมุนข้อมูล โดย Time จะกลายเป็นไดเมนชันหลัก ส่วนในตารางที่ 3.4 เป็นการตัดข้อมูลจากตารางที่ 3.3 ที่เลือกเฉพาะรถยนต์ยี่ห้อ Ford ขึ้นมาวิเคราะห์เท่านั้น

3.3 โครงสร้างฐานข้อมูล (Database Schema)

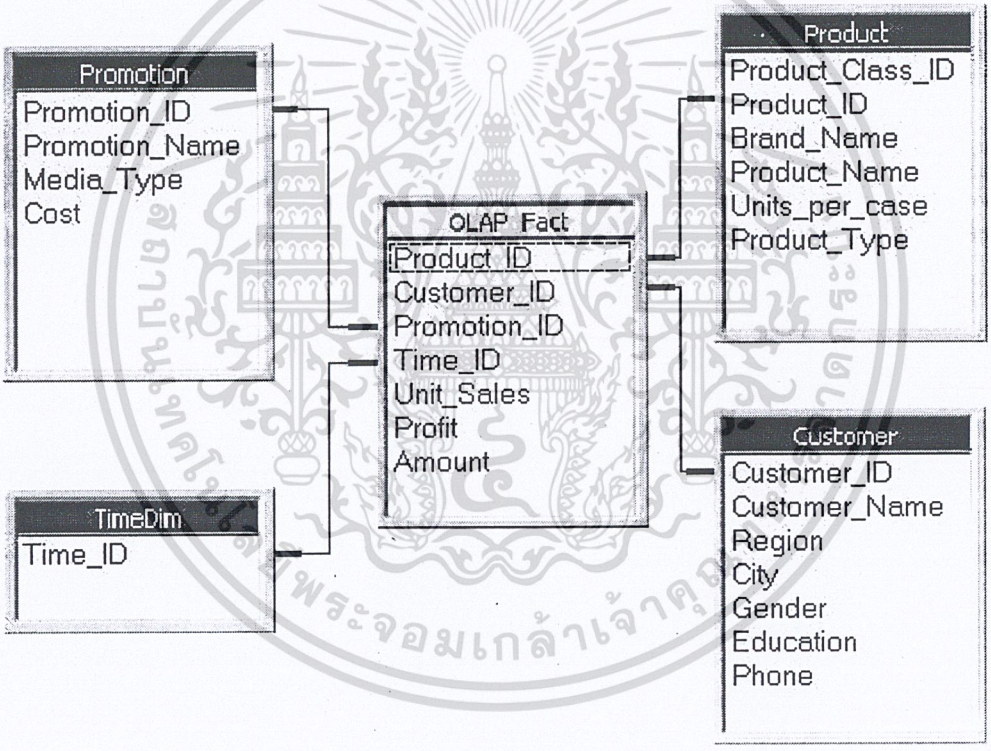
โครงสร้างฐานข้อมูลหลักๆ จะมีอยู่ 2 รูปแบบ คือ Star Schema และ Snowflake Schema

3.3.1 Star Schema

Star Schema เป็นอีกชื่อหนึ่งของ Dimensional Model ซึ่งเป็นชื่อที่ใช้กันมานานแล้ว เนื่องจากไดอะแกรม (diagram) มีรูปร่างคล้ายกับดาว ซึ่งประกอบไปด้วยตารางใหญ่ 1 ตาราง อยู่ตรงกลาง และมีตารางเล็กๆ ที่มีความสัมพันธ์กับตารางหลัก อยู่รอบๆ ตารางหลักนี้จะเป็นตารางเดียวที่ใช้การเชื่อมต่อแบบหลายจุด (Multiple join) เพื่อเชื่อมต่อกับตารางอื่นๆ แต่ตารางอื่นๆ ที่อยู่รอบๆ นั้น จะมีการเชื่อมต่อเพียงแค่จุดเดียว (Single join) เพื่อเชื่อมเข้ากับตารางหลักเท่านั้น

ตารางหลัก จะเรียกว่า ตารางแฟ็ค (Fact Table)

ตารางอื่นๆ จะเรียกว่า ตารางไดเมนชัน (Dimension Table)

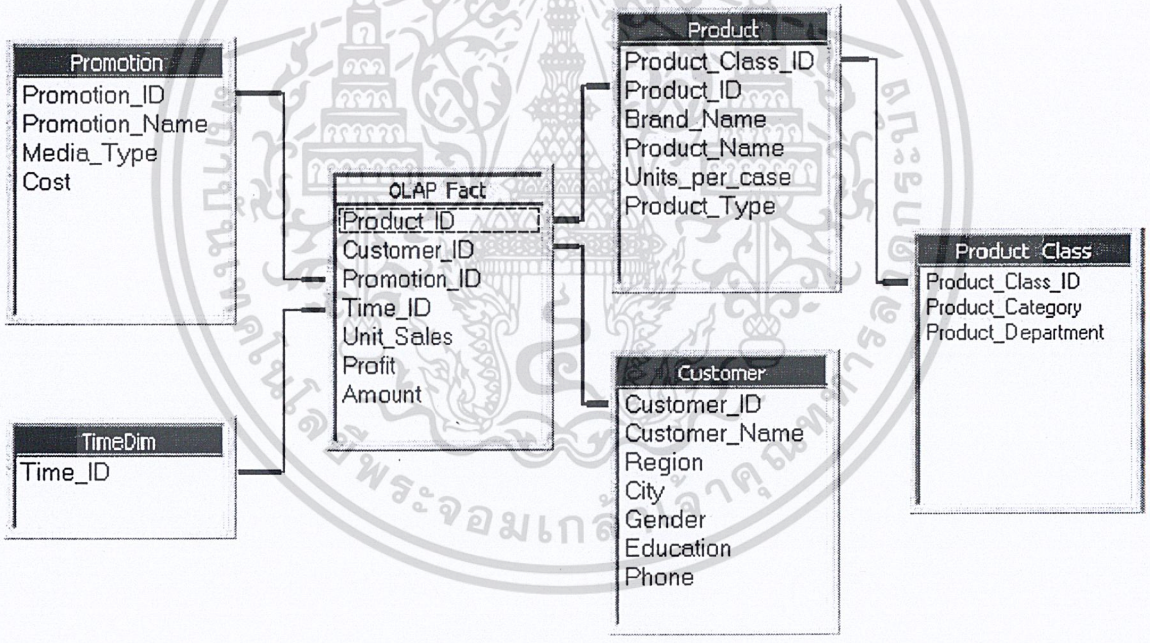


รูปที่ 3.3 แสดงโครงสร้างฐานข้อมูลแบบ Star Schema

3.3.2 Snowflake Schema

ตารางไดเมนชันที่มีข้อมูลแบบเป็นลำดับขั้น (Hierarchy) อาจจะมีการแตกข้อมูลออกมาเป็นตารางย่อยๆ เพื่อให้ผู้ออกแบบสามารถเข้าใจได้ง่ายยิ่งขึ้น ซึ่งเรียกโครงสร้างนี้ว่า สโนว์เฟลก (Snowflake) ความสัมพันธ์ที่เป็นแบบ many – to – one แต่ละอันจะถูกแยกออกเป็นตารางย่อยๆ โดยคีย์หลัก (Primary key) ของตารางย่อย จะต้องมาเป็น Foreign key ของตารางที่ตัวเองไปเกาะอยู่ ดังแสดงในรูปที่ 3.4

การทำโครงสร้างแบบสโนว์เฟลก อาจทำให้ประสิทธิภาพในการขับเคลื่อนลดต่ำลง เนื่องจากต้องมีการรวมกันระหว่างตารางย่อยกับตารางไดเมนชันอีกทีหนึ่ง รวมทั้งต้องสร้าง SQL ที่มากขึ้น แต่อย่างไรก็ตาม สโนว์เฟลกก็เหมาะสำหรับฐานข้อมูลที่มีขนาดใหญ่ เช่น มีจำนวนหลายแสนเรคอร์ด เพราะมีส่วนช่วยลดความซ้ำซ้อนลงได้ รวมทั้งโครงสร้างแบบสโนว์เฟลกมีข้อดี เช่น เมื่อต้องการเรียกดูเฉพาะฟิลด์ (field) ในตารางไดเมนชันที่เป็นสโนว์เฟลก ก็ทำเพียงแค่ดึงข้อมูลจากตารางไดเมนชันนั้นเพียงตารางเดียว และทำการเชื่อมต่อกับตารางแฟ็ค เท่านั้น ไม่ต้องทำการเชื่อมต่อระหว่างตารางไดเมนชันนั้นกับตารางย่อยของตัวมันเองอีก และจะไม่ทำให้เกิดปัญหาการอัพเดท โดยไม่ต้องทำการอัพเดทหลายที่ (Multiple update)



รูปที่ 3.4 แสดงโครงสร้างฐานข้อมูลที่เป็นแบบสโนว์เฟลก (Snowflake Schema)

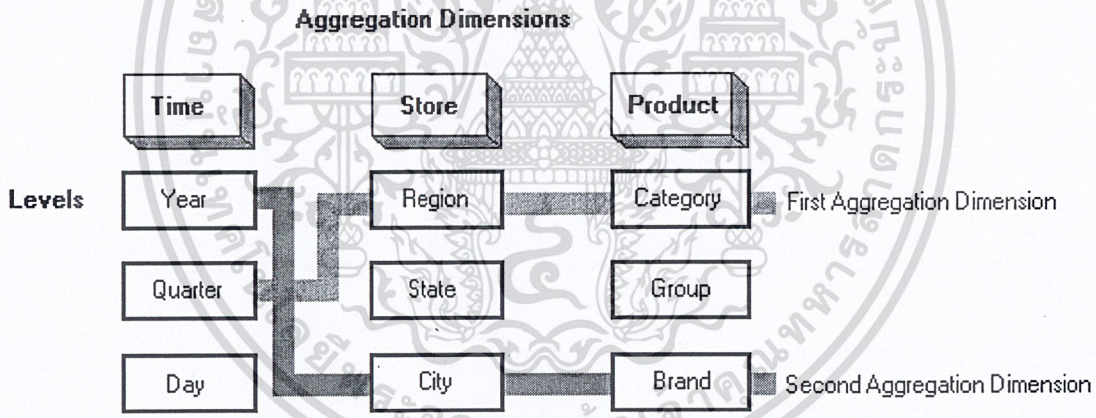
3.4 การรวบรวมข้อมูล (Aggregation)

เนื่องจากข้อมูลพื้นฐานของไดเมนชันแนลดาต้าแวร์เฮาส์ (Dimensional Data Warehouse) จะประกอบด้วยเรคอร์ดจำนวนมาก ถ้านักวิเคราะห์ต้องการทำ query โดยไม่มีการกำหนดข้อบังคับให้กับบางไดเมนชัน เนื่องจากในดาต้าแวร์เฮาส์ จะมีเฉพาะแต่ข้อมูลพื้นฐานเท่านั้น (Based-level data) จึงทำให้เวลา query จะต้องทำการรวม (sum) ข้อมูลภายในเรคอร์ดจำนวนมากมาย ซึ่งถ้าเป็นเช่นนี้จะทำให้การทำ query เกิดความสิ้นเปลืองสูงมาก ดังนั้นจึงได้มีการทำการรวบรวมข้อมูล (Aggregate) เอาไว้ล่วงหน้า เพื่อเร่งให้การทำ query สามารถทำได้เร็วขึ้น มีประสิทธิภาพ (Performance) ที่ดียิ่งขึ้น

การรวบรวม คือ fact table record ที่แสดงถึงข้อมูลสรุป ที่ได้จากตารางแฟ็ค ระดับพื้นฐาน (Based-level fact table)

การรวบรวมมีหลายชนิด และแต่ละชนิดจะมีตารางแฟ็คเป็นของตัวเอง ซึ่งตารางแฟ็คเหล่านี้จะถูกเรียกว่าเป็น Derivative fact table เพราะข้อมูลเหล่านั้นได้รับ (derive) มาจากตารางแฟ็คระดับพื้นฐาน

เราสามารถสร้างการรวบรวมได้มากมายตามที่เราต้องการ ซึ่งในความเป็นจริง เราจะสร้างการรวบรวมไว้เฉพาะส่วนที่เราต้องการเท่านั้น ไม่จำเป็นต้องสร้างการรวบรวมทุกๆ ฟิลด์ในแต่ละไดเมนชันและไม่จำเป็นต้องทำการรวบรวมทุกๆ ไดเมนชันด้วย



รูปที่ 3.5 แสดงการรวบรวม (Aggregation)

3.5 Crosstabulations หรือ Crosstabs

CrossTabs เป็นวิธีในการแสดงกลุ่มของข้อมูล เพื่อให้มีความสัมพันธ์กันและสามารถมองเห็นแนวโน้มได้ง่าย ซึ่งฟิลด์ (Field) ของตารางไดเมนชันจะอยู่ในรูปของ CrossTabs และค่าในแต่ละฟิลด์จะถูกแบ่งกลุ่มและถูกสรุปภายในไดเมนชัน ทำให้สามารถมองได้ง่ายกว่าการแสดงแบบตารางปกติ

	Media_Type	Product_Tpye	Sum of Unit_Sales
▶	Brochure	Drink	7909
	Mail	Drink	8576
	News Paper	Drink	7695
	Product Attachm	Drink	8319
	Radio	Drink	7847
	TV	Drink	8719
	Brochure	Food	68177
	Mail	Food	72836
	News Paper	Food	63504
	Product Attachm	Food	68144
	Radio	Food	69268
	TV	Food	73307

ระเบียน: 1 จาก 36

รูปที่ 3.6 แสดงข้อมูลในตารางแบบปกติ

	Media Type						
Product Typ	All Promotior	Brochure	Mail	News Paper	Product Attachment	Radio	TV
All Product	761759	92861	99710	86753	93760	94206	99992
Drink	66130	7909	8576	7695	8319	7847	8719
Food	557554	68177	72836	63504	68144	69268	73307
Non-Consun	138075	16775	18298	15554	17297	17091	17966

รูปที่ 3.7 แสดงข้อมูลในตารางแบบ CrossTabs

3.6 ขั้นตอนการออกแบบดาต้าแวร์เฮาส์

ในการออกแบบดาต้าแวร์เฮาส์ จะประกอบไปด้วย 4 ขั้นตอน คือ

3.6.1 เลือกกระบวนการทางธุรกิจ (Business process) ที่ต้องการสร้าง

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

3.6.2 เลือกเกรน (Grain) ของกระบวนการทางธุรกิจ

เกรน : เป็นข้อมูลพื้นฐาน ข้อมูลที่เป็นอะตอมมิก (Atomic) ซึ่งถูกแสดงอยู่ในตารางแฟ็ค สำหรับกระบวนการนี้ เกรนที่มีอยู่ทั่วไป เช่น ข้อมูลการทำทรานแซคชันในแต่ละครั้ง (Individual Transaction), ข้อมูลของการทำงานในแต่ละวันหรือสรุปในแต่ละวัน (Individual daily snapshots), ข้อมูลสรุปการทำงานในแต่ละเดือน (Individual monthly snapshots)

3.6.3 เลือกไดเมนชันที่จะถูกนำมาใช้กับแต่ละเรคอร์ดของตารางแฟ็ค

ไดเมนชันที่มีอยู่โดยทั่วไป เช่น Product, Customer, Time ซึ่งแต่ละไดเมนชันจะถูกอธิบายแยกกันในลักษณะของแอททริบิวท์และจะถูกอธิบายเป็นตัวหนังสือ

3.6.4 เลือกแฟ็คที่มีการวัดหรือมีการประมวลผลหรือมีการคำนวณไว้แล้ว (measured fact) ที่จะเก็บอยู่ในแต่ละเรคอร์ดของตารางแฟ็ค

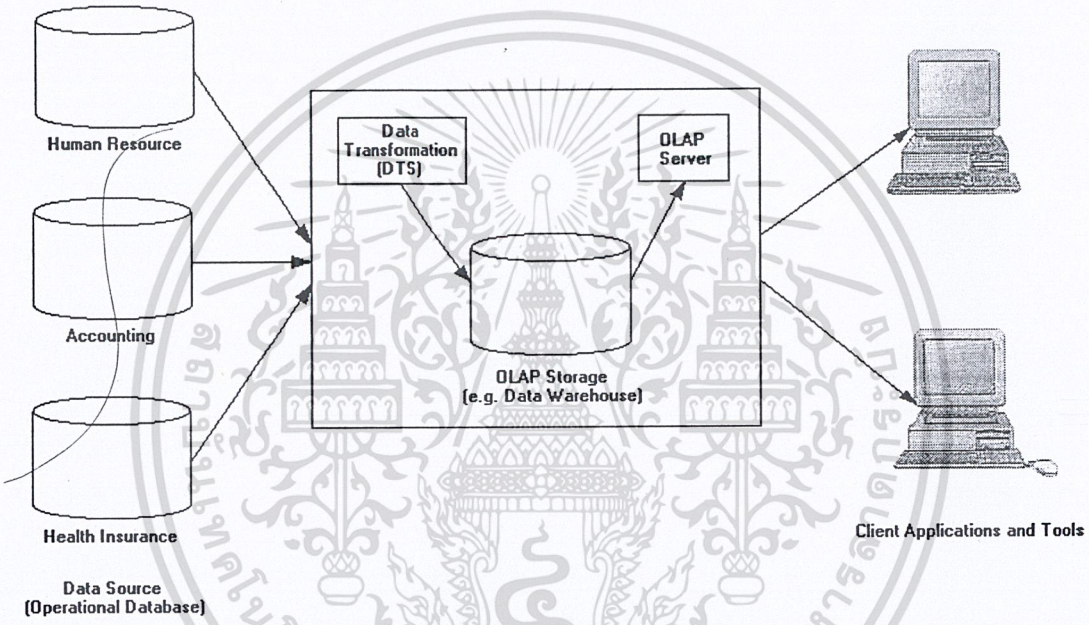
ตัววัดที่มีอยู่โดยทั่วไป เช่น ปริมาณต่างๆ ซึ่งจะมีลักษณะเป็นตัวเลข ได้แก่ ยอดขาย, จำนวนสินค้าที่ขายได้, ค่าใช้จ่ายทั้งหมด

บทที่ 4

OLAP Server Architecture

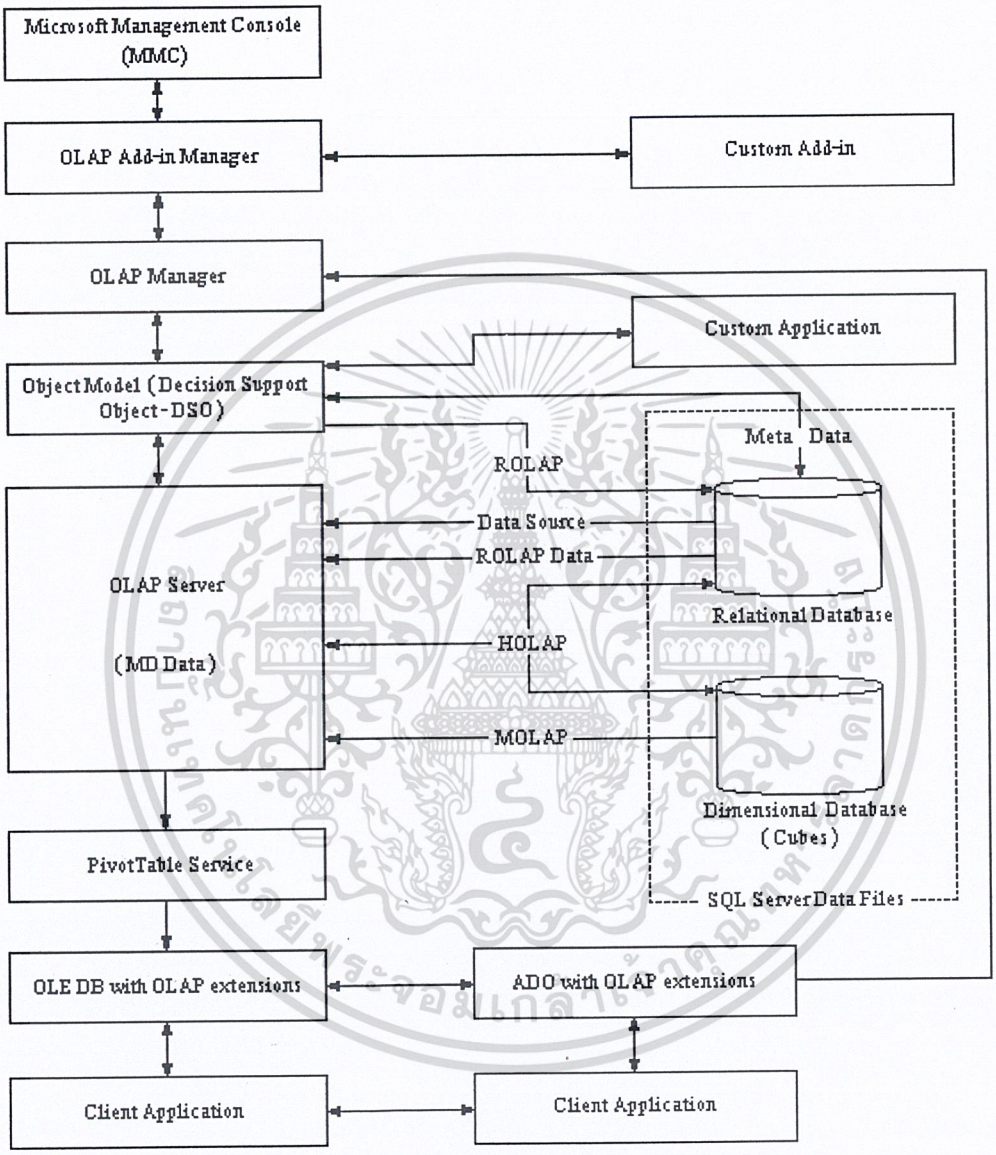
4.1 Microsoft OLAP Services

ส่วนประกอบต่างๆ ของ OLAP Services Architecture แบ่งออกเป็นส่วนหลักๆได้ดังต่อไปนี้ คือ แหล่งข้อมูล (Data Source), การเก็บข้อมูล (Data Storage), OLAP Server และส่วนของ Client ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 ภาพรวมของ OLAP Services Architecture

เมื่อทำการมองลึกเข้าไปในรายละเอียด จะเห็นถึงส่วนประกอบในแต่ละระดับมากขึ้นไปอีก แสดงในรูปที่ 4.2



รูปที่ 4.2 ส่วนประกอบในระดับต่างๆ ของ OLAP Services

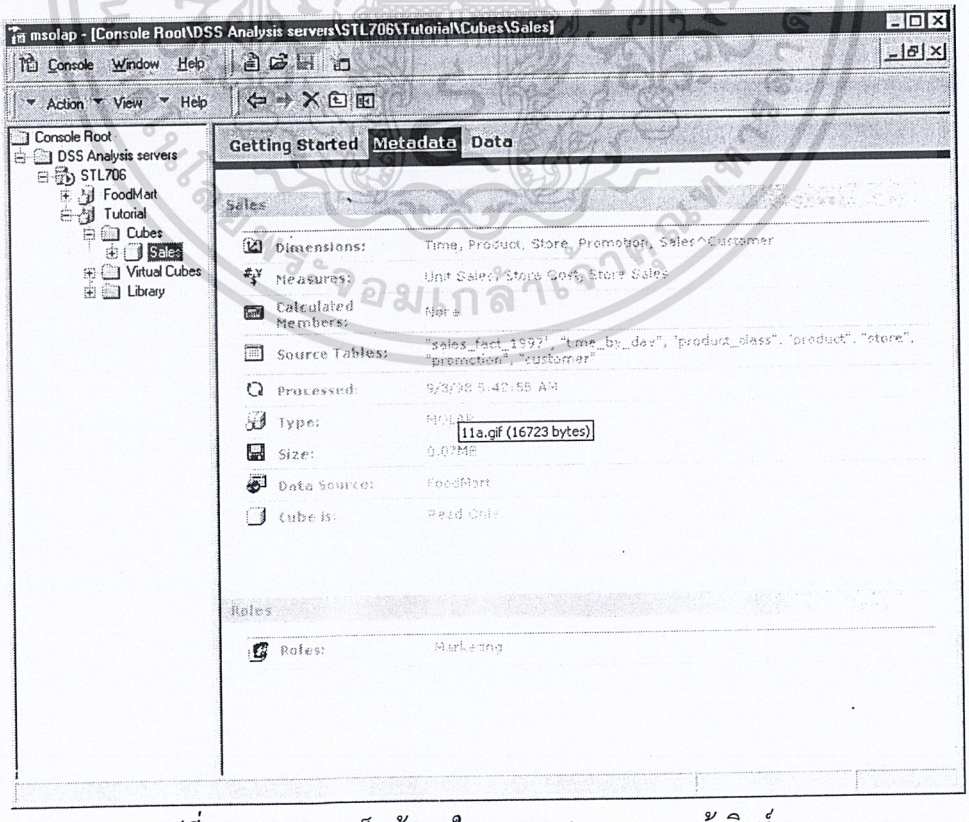
รูปภาพนี้จะแสดง OLAP Server ที่สร้างและจัดการกับมัลติไดเมนชันแนลดาต้าคิวบ์ (Multidimensional data cube) โดย OLAP Server จะใช้ PivotTable Service (PivotTable Service) ในการจัดหา OLAP data คือ เป็นตัวส่งข้อมูลกลับไปยัง Client Application ซึ่ง PivotTable Service จะทำการติดต่อผ่านทาง Multidimensional ActiveX Data Object (ADO MD) และ OLE DB Provider for Microsoft OLAP Services

ข้อมูลของระบบ OLAP มักจะเป็นฐานข้อมูลรีเลชันแนล ซึ่งจะถูกแปลงให้อยู่ในโครงสร้างแบบ Star Schema หรือ Snowflake Schema เพื่อใช้ในระบบดาต้าแวร์เฮาส์ของ OLAP ข้อมูลที่เก็บอยู่ในฐานข้อมูลของดาต้าแวร์เฮาส์สามารถรวบรวมมาจากหลายๆแหล่งข้อมูลได้ เช่น ระบบฐานข้อมูลประจำวัน (OLTP) หรือฐานข้อมูลสะสม ซึ่งแหล่งข้อมูลเหล่านี้ต้องติดต่อผ่านทาง OLE DB หรือ ODBC ได้ , Data Transformation Services (DTS) ของ SQL Server เป็นส่วนที่ทำหน้าที่จัดการดาต้าแวร์เฮาส์ โดยทำการแปลงข้อมูลจากแหล่งข้อมูลอื่นๆ

OLAP Server จะใช้ข้อมูลในดาต้าแวร์เฮาส์ เพื่อทำการสร้างดาต้าคิวบ์ (Data Cube) โดยที่ขอบเขตและรายละเอียดของ Cube จะเรียกว่า Cube Metadata ซึ่งจะเก็บไว้ที่ Repository ได้ 3 รูปแบบ คือ

- เป็นไฟล์ฐานข้อมูลแบบมัลติไดเมนชันแนล (Multidimensional Database Files) จะเรียกการเก็บข้อมูลแบบนี้ว่า Multidimensional OLAP หรือ MOLAP ซึ่งจะเก็บไว้ในฐานข้อมูลแบบมัลติไดเมนชันแนล
- เป็นตารางในฐานข้อมูลแบบรีเลชันแนล จะเรียกการเก็บข้อมูลแบบนี้ว่า Relational OLAP หรือ ROLAP
- เป็นแบบผสมของไฟล์ฐานข้อมูลแบบมัลติไดเมนชันแนลและตารางแบบรีเลชันแนล (Relational Tables) จะเรียกการเก็บข้อมูลแบบนี้ว่า Hybrid OLAP หรือ HOLAP

Repository : เป็นส่วนที่ใช้เก็บและใช้วัตถุ (Objects) ร่วมกันระหว่างซอฟท์แวร์ทูลส์ (Software tools) เช่น Add-ins, Services, System descriptions, Instances of system objects และ Custom reusable components ซึ่งใช้ใน SQL Server ในการเก็บข้อมูลเกี่ยวกับวัตถุที่แตกต่างกัน ซึ่งแสดงโดยใช้เซอร์วิส (Services) และทูลส์ (Tools) ของ SQL Server ดังแสดงได้ในรูปที่ 4.3

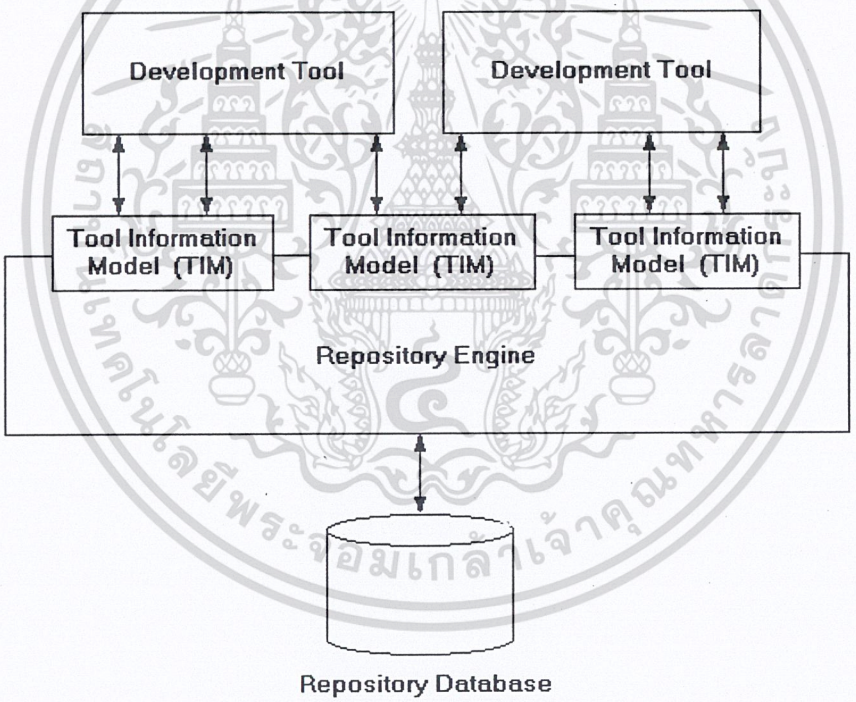


รูปที่ 4.3 แสดงการเก็บข้อมูลใน Repository ของดาต้าคิวบ์

หัวใจหลักของ Repository ก็คือ เอ็นจิน (Engine) ซึ่งเป็นกลุ่มของ COM และ OLE Automation Interface ซึ่งอินเทอร์เฟสเหล่านี้จะเป็นตัวที่เข้าใช้ Repository เพื่อยอมให้มีการอ่านและเขียนข้อมูลเข้าไปได้ ข้อมูลของ Repository จะถูกเก็บเข้าไปในฐานข้อมูลรีเลชันแนล ซึ่งใน SQL Server ก็จะเป็น MSDB

ส่วนประกอบของ Repository มีด้วยกัน 4 ส่วน คือ

- *Repository Database* : เป็นฐานข้อมูลรีเลชันแนล มักจะเป็น SQL Server หรือ Microsoft Access database
- *Repository Engine* : เป็นทูลส์ที่ใช้ COM ซึ่งคอยจัดการข้อมูลใน Repository
- *Tool Information Model (TIM)* : เป็นการรวมตัวกันของโมเดลวัตถุ ซึ่งระบุชนิดของข้อมูลของทูลส์ที่สนใจ
- *Application Development Tools* : เป็นซอฟท์แวร์ทูลส์ที่ใช้สำหรับจัดการในเรื่องการพัฒนาการประมวลผล, การขยายซอฟท์แวร์ (Deploying) และการดูแลซอฟท์แวร์แอพลิเคชัน



รูปที่ 4.4 แสดงโครงสร้างของ Repository

OLAP Manager Graphical User Interface (GUI) ใช้ในการควบคุม OLAP Manager และยังสามารถพัฒนาโปรแกรมประยุกต์ขึ้นใช้เองได้ โดยใช้ Decision Support Objects (DSO) เพื่อควบคุม OLAP Server ซึ่ง DSO Application จะควบคุมการสร้างและการจัดการของดาต้าคิวบ์ โดยเซิร์ฟเวอร์ (Server) และยังสามารถออกแบบให้จัดการกับคิวบ์เมต้าดาต้า (Cube Metadata) ใน Repository ได้ด้วยวิชวลเบสิก ซึ่งเป็นภาษาที่นิยมใช้ในการพัฒนา DSO

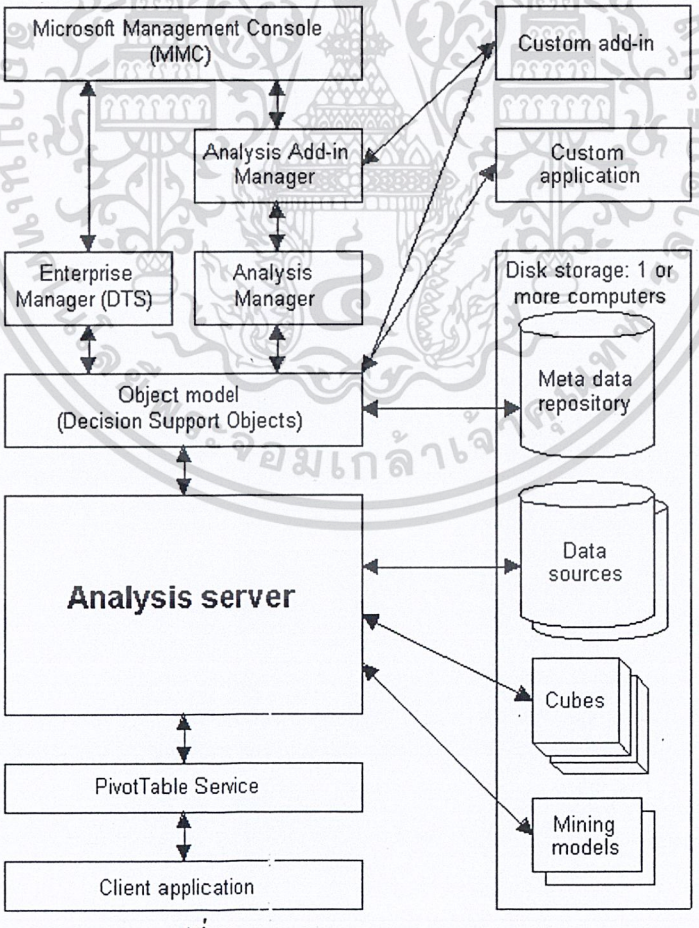
Object Model เพื่อควบคุม OLAP Server โดยการติดต่อกับ OLAP Manager User Interface ดูรายละเอียดของ DSO ได้ในภาคผนวก ก

4.2 OLAP Server Architecture

OLAP Services จะจัดการกับทางเซิร์ฟเวอร์ ในการสร้างและจัดการข้อมูลของ Multidimensional OLAP และจัดการกับข้อมูลทางฝั่ง Client โดยผ่าน PivotTable Service การทำงานฝั่งเซิร์ฟเวอร์จะรวมทั้งการสร้างมัลติไดเมนชันแนลดาต้าคิวบ์ (Multidimensional data cube) จากฐานข้อมูลรีเลชันแนลดาต้าแวร์เฮาส์ (relational data warehouse) และจัดเก็บ Cube ในโครงสร้างฐานข้อมูลแบบมัลติไดเมนชันแนลหรือแบบรีเลชันแนลหรือทั้งสองแบบ ในส่วนของเมต้าดาต้าของ Cube จะเก็บไว้ใน Repository ในฐานข้อมูลแบบรีเลชันแนล

ส่วนที่ติดต่อกับผู้ใช้ จะเป็นหน้าที่ของ OLAP Manager ที่ทำงานภายใต้ Microsoft Management Console (MMC) และสามารถพัฒนาโปรแกรมประยุกต์เพื่อใช้ในการควบคุมเซิร์ฟเวอร์ได้เหมือนกับ OLAP Manager

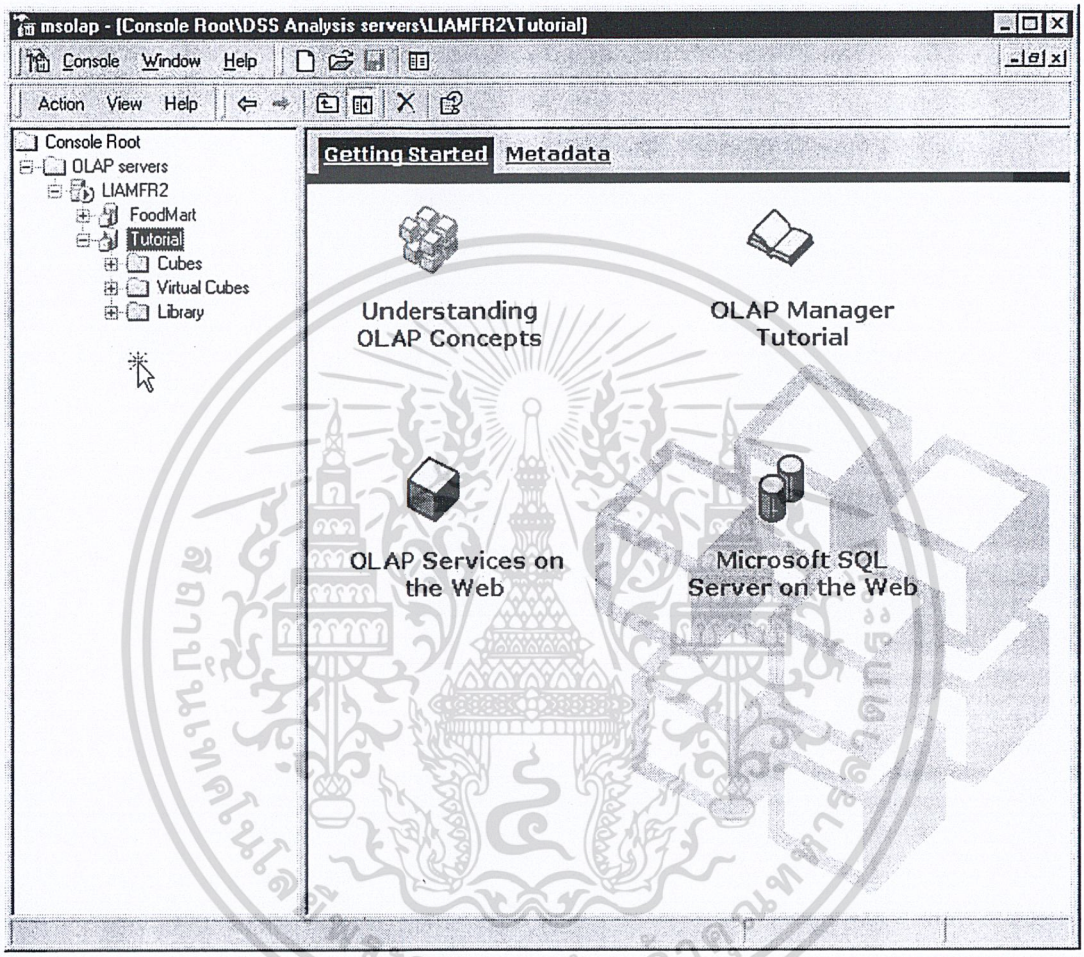
สรุปส่วนประกอบที่สำคัญของ OLAP Server Architecture แสดงได้ดังรูปที่ 4.5



รูปที่ 4.5 OLAP Server Architecture

4.2.1 ซอร์ฟแวร์ OLAP Manager

SQL Server ในส่วนของ OLAP Services จะมี OLAP Manager มาให้ใช้งาน แอพลิเคชันนี้จะเป็นตัวจัดการด้านการติดต่อกับผู้ใช้ (User Interface) สำหรับการเข้าใช้งาน OLAP Server และฐานข้อมูลขนาดใหญ่ๆ ที่มีโครงสร้างแบบมัลติไดเมนชันแนล ดังแสดงในรูปที่ 4.6



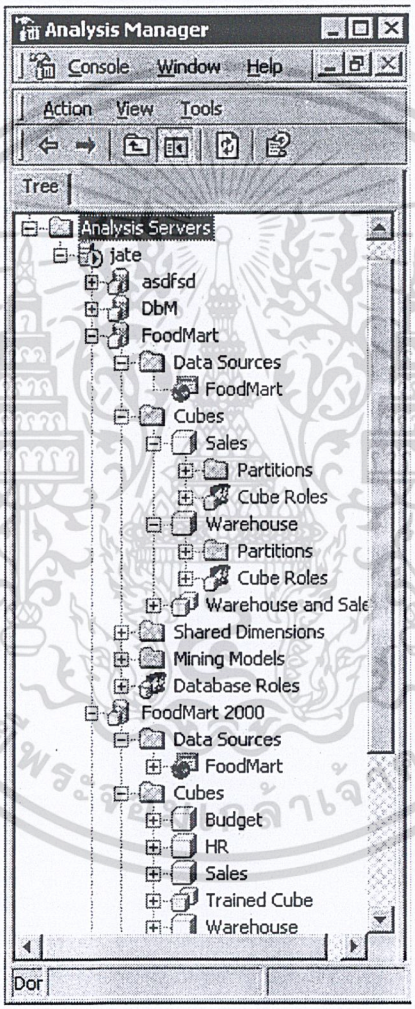
รูปที่ 4.6 OLAP Manager

OLAP Manager จะประกอบไปด้วย Pane 2 Pane หลักๆ คือ Tree pane และ HTML-based detail pane

- **Tree pane :** จะเป็น Pane ที่แสดงการมองวัตถุและฟังก์ชันอย่างเป็นลำดับ ดังรูปที่ 4.7 ซึ่งมีส่วนประกอบต่างๆ ดังต่อไปนี้

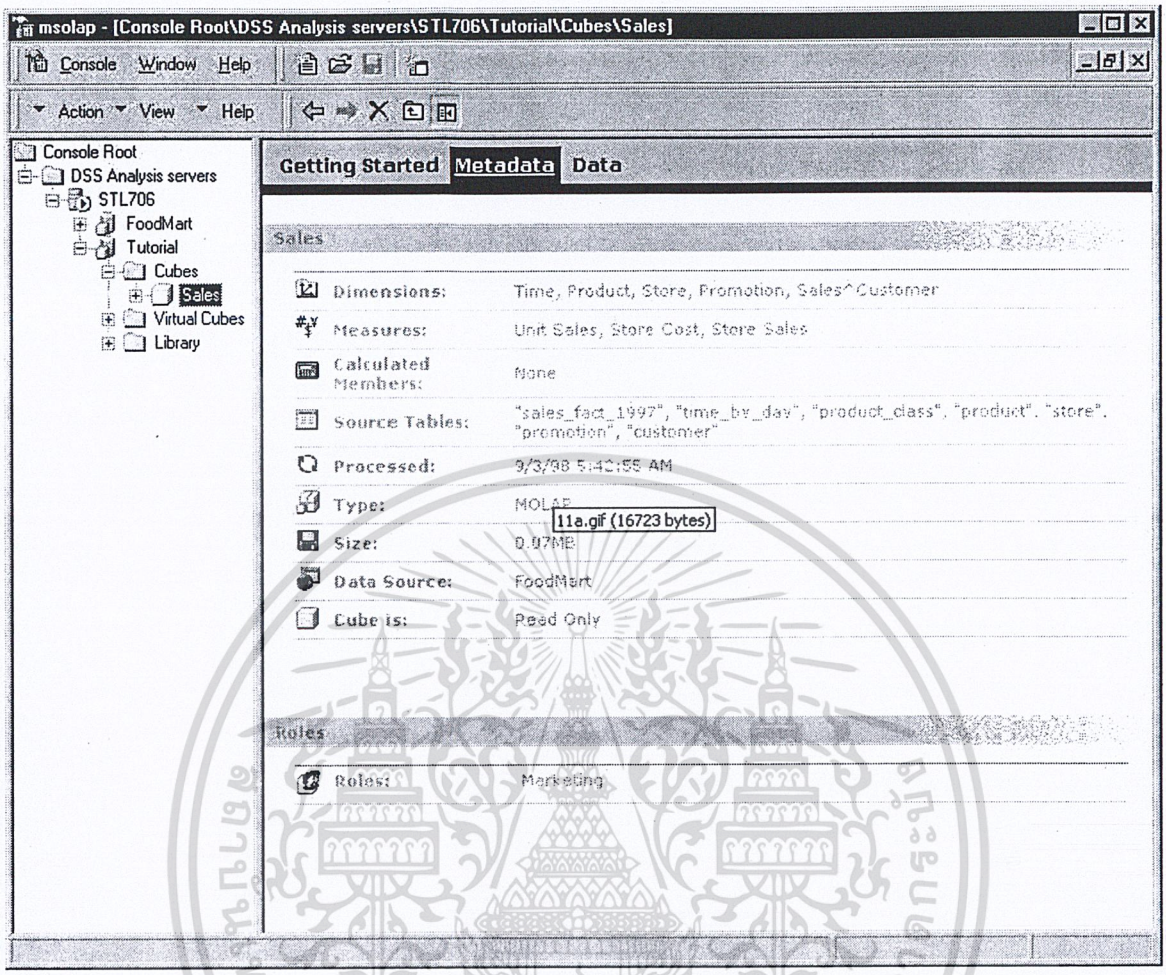
  - Registerd OLAP Servers : เป็นโหนด (Node) แสดงชื่อของ OLAP Server ที่มีใน OLAP Service
  - Database : OLAP Server แต่ละตัวสามารถมีฐานข้อมูลได้หลายอัน เช่น ในรูปที่ 4.7 จะมีฐานข้อมูลอยู่ 2 อัน คือ FoodMart และ Pubs ซึ่งในฐานข้อมูลแต่ละอันก็ยังสามารถมี Cubes, Virtual Cubes หรือ Library หลายอันเช่นกัน
  - Cubes : เป็นระดับ (Level) ที่อยู่ถัดจาก Database เช่น ในรูปที่ 4.7 จะมี Cubes เป็น Sales และ Warehouse

- Virtual Cubes : เป็นการนำเอา Cube มากกว่า 2 Cube ในฐานข้อมูลมารวมกัน เพื่อช่วยในการมองข้อมูล
- Library : เป็นส่วนที่เก็บ Data Source, Virtual dimensions, Shared dimensions และ Roles ในฐานข้อมูล
- Data Source : เป็นส่วนที่มีข้อมูลเกี่ยวกับ OLE DB Provider เช่น connection information, network settings, connection time-out และ access permissions
- Dimensions : เป็นโฟลเดอร์ (Folder) ที่เก็บโหนดของแต่ละไดเมนชันใน Cube
- Measures : แสดงถึงข้อมูลที่สามารถค้นหาได้จากการวิเคราะห์แบบไดเมนชันแนล เช่น Unit sales, Store cost
- Roles : แสดงถึงผู้ใช้แต่ละกลุ่มที่มีสิทธิ์เข้าใช้ Cubes



รูปที่ 4.7 OLAP Manager Tree pane

- **HTML-based detail pane** : จะแสดงให้เห็นการเลือกแบบไดนามิก (Dynamic) คือ จะมีการเปลี่ยนแปลงเกิดขึ้นเวลาทำการเลือกใน Tree view pane ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 OLAP Manager HTML-based detail pane

4.2.2 คอมโพเนนต์ (Components) ต่างๆ ของซอฟท์แวร์ OLAP Manager

4.2.2.1 OLAP Server

เป็นโหนด (Node) ที่แสดงถึงชื่อของเซิร์ฟเวอร์ (Server) ต่างๆ

4.2.2.2 Databases

เป็นโหนดที่ในส่วนถัดจากโหนด Server ลงไป ซึ่ง Database เหล่านี้จะเป็นตัวเก็บโฟลเดอร์ของ Cubes, Virtual Cube และ Library

4.2.2.3 Cubes

เป็นส่วนที่อยู่ภายใต้โหนดของ Databases โดยชื่อของแต่ละ Cube จะถูกแสดงเป็นไอคอน (icon) ซึ่งประกอบไปด้วย

- **Data source** 

โหนดแรกของ Cube ก็คือ Data source จะทำหน้าที่ดูแลในส่วน OLE DB เช่น ข้อมูล, ข้อมูลที่ใช้ติดต่อกับเซิร์ฟเวอร์, การติดตั้งระบบเครือข่าย, เวลาในการติดต่อ และการอนุญาตในการเข้าใช้ข้อมูล

- **Dimensions** 

โฟลเดอร์นี้จะเก็บโหนดของแต่ละ dimension ใน cube ซึ่ง dimension นี้สามารถทำเป็นแบบ private, shared หรือ virtual ก็ได้

Private dimensions จะมีอยู่ในเฉพาะ cube ที่เจาะจงเท่านั้น

Shared dimensions มีใน cube ที่อยู่ใน database และปรากฏในโฟลเดอร์ Shared Dimension ซึ่งอยู่ในโฟลเดอร์ Library ของ database

Virtual dimensions มีใน cube ที่อยู่ใน database และปรากฏในโฟลเดอร์ Virtual Dimension ซึ่งอยู่ในโฟลเดอร์ Library ของ database

- **Levels**
  - Year
  - Quarter
  - Month

แต่ละ dimension ของ cube จะเก็บโหนดของแต่ละ levels ของมันเอง โดย Level ระดับที่ 1 จะแทนด้วยจุดเดียว, Level ระดับที่ 2 จะแทนด้วยจุด 2 จุด, Level ระดับที่ 3 ก็จะแทนด้วยจุด 3 จุด อย่างนี้ไปเรื่อยๆ

- **Measures** 

เป็นโฟลเดอร์ที่เก็บค่าของตัววัดแต่ละตัวใน cube

- **Partitions** 

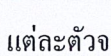
เป็นโฟลเดอร์ที่เก็บโหนดของแต่ละ partition ใน cube

- **Roles** 

โฟลเดอร์ Roles นี้จะเก็บโหนดของกลุ่มผู้ที่ได้รับอนุญาตเข้าใช้ข้อมูล

4.2.2.4 Virtual Cubes 

Database แต่ละตัวจะมีการเก็บโฟลเดอร์ Virtual Cube ซึ่ง Virtual Cube นี้คือการเอา cube 2 cube ขึ้นไปมารวมกันเป็นหนึ่งเดียว ภายในโหนด Virtual Cube นี้เป็นเซตของโฟลเดอร์ที่เก็บข้อมูลทั้งหมด ได้แก่ โฟลเดอร์ Measures, Dimensions และ Roles เช่นเดียวกับใน Cube

4.2.2.5 Library 

Database แต่ละตัวจะมีโฟลเดอร์ Library ซึ่งโฟลเดอร์นี้จะทำหน้าที่เก็บ Data sources, Shared dimensions, Virtual dimensions และ Roles

- **Data sources** 

Data source นั้นเป็นตัวที่ดูแลเกี่ยวกับ OLE DB ไม่ว่าจะเป็นด้านข้อมูล, ข้อมูลที่ใช้ติดต่อกับ Server, การติดตั้งระบบเครือข่าย, เวลาในการติดต่อ และการอนุญาตในการเข้าใช้ข้อมูล Database หนึ่งสามารถที่จะเก็บได้หลายๆ Data source ภายในโฟลเดอร์ Data Sources

- **Shared Dimensions** 

โฟลเดอร์นี้จะเก็บโหนดของแต่ละ Shared dimension ใน database ส่วน dimension ที่เก็บอยู่ในโฟลเดอร์นี้สามารถจะอยู่ใน Cube ใดๆใน database ก็ได้ และโฟลเดอร์นี้ก็สามารถทำเป็น Levels ได้

- **Virtual Dimensions** 

โฟลเดอร์นี้จะเก็บโหนดของแต่ละ Virtual dimension ใน database ส่วน dimension ที่เก็บอยู่ในโฟลเดอร์นี้สามารถจะอยู่ใน Cube ใดๆใน database ก็ได้ และโฟลเดอร์นี้ก็สามารถทำเป็น Levels ได้ แต่จะทำได้เพียงแค่ Level ระดับที่ 1 เท่านั้น

4.2.3 ขั้นตอนการใช้งานซอฟท์แวร์ OLAP Manager

4.2.3.1 การลงทะเบียนเป็นเครื่องเซิร์ฟเวอร์ (Register Server)

เครื่องคอมพิวเตอร์ที่ทำการติดตั้ง SQL Server with OLAP Services ลงไป จะได้รับการลงทะเบียนให้เป็นเสมือนเครื่องเซิร์ฟเวอร์โดยอัตโนมัติ จะปรากฏให้เห็นใน OLAP Manager Tree View ทันที ซึ่งอยู่ในโฟลเดอร์ OLAP Servers

4.2.3.2 การระบุ Databases และ Data Sources

ก่อนที่จะสร้าง Cubes จะต้องทำการระบุ Databases และ Data Sources อย่างน้อยอย่างละ 1 โดยระบุ database หนึ่งตัวเพื่อเก็บกลุ่มของ cubes ที่มีความสัมพันธ์กัน, เก็บ data source และ roles ข้อมูลขนาดใหญ่ (Metadata) ที่หลายๆ Cube ต้องใช้ร่วมกัน ก็ควรจะเก็บไว้ใน Database เดียวกัน

ในแต่ละ Database เราจะต้องทำการระบุ Data Sources ไว้อย่างน้อย 1 data source ซึ่งเป็นตัวจัดการข้อมูลให้กับ Cube สำหรับ OLAP Manager นั้นจะทำการแก้ไขหรือจัดการกับ OLE DB data sources และ ODBC data sources ที่กำหนดอยู่ใน ODBC Data Source Administrator ได้ง่ายเพียงไปเลือก data source ที่มีอยู่ในลิสต์ โดยจะใช้ *Data Link Properties* dialog box

4.2.3.3 การสร้าง Shared Dimensions

Dimensions จะถูกจัดเป็นประเภทตามข้อมูลแบบตัวเลข (Numeric)ใน Cube เช่น ถ้าใช้ตัววัดค่าใน cube เป็นค่า *Cost* และ dimensions คือ Time, Supplier และ Item Description ผู้ที่ใช้ cube ก็สามารถแยกค่า Cost ที่อยู่ใน dimensions เหล่านี้ได้ ซึ่ง Shared Dimension คือ dimension ที่มีอยู่ในหลายๆ cubes ใน database ส่วนใหญ่ก็จะเป็น dimension ธรรมดาทั่วไป เช่น Time dimension

ในการสร้าง Shared Dimension สามารถทำได้ง่ายและรวดเร็ว เพราะจะมี Dimension Wizard ให้ใช้งาน โดยทำตามขั้นตอนใน wizard จนเสร็จก็จะได้ Shared Dimension แล้ว สิ่งที่น่าสนใจใน wizard มีดังนี้

- ***One or multiple dimension tables***

One : ถ้าจะเพิ่ม shared dimension ใน cube และ shared dimension table นั้นทำการเชื่อมกับตาราง Fact ก็จะไม่มีผลต่อประเภทของ cube's schema (star/snowflake) แต่ถ้า shared dimension table ไปเชื่อมกับไดเมนชันอื่นๆ จะทำให้ cube มีโครงสร้างเป็นแบบสโนว์เฟลก

Multiple : ถ้าจะเพิ่ม shared dimension ใน cube จะต้องเป็นแบบสโนว์เฟลกอย่างเดียว

- ***Data source***

ทำการเลือก data source ที่เก็บตารางและคอลัมน์ (columns) ซึ่งใช้ใน shared dimension ชื่อของ data source เป็นตัวบ่งบอกการใช้ database resource และ parameters

- ***Dimension tables***

ทำการเลือกตารางไดเมนชันที่เก็บคอลัมน์ตามต้องการ ซึ่งตารางไดเมนชัน ก็คือ ตารางทั่วไปที่เชื่อมต่อกับตารางแฟ็คใน Cube's Schema

- ***Dimension Levels***

ทำการเลือกระดับ (Levels) ที่ต้องการมีใน Shared dimension โดยที่ระดับ ก็คือ คอลัมน์ซึ่งยกเว้นบาง Time dimension ระดับภายในไดเมนชันจะถูกจัดการอย่างเป็นลำดับชั้น เช่น ใน Location Dimension ระดับอาจจะเป็น Country, State และ City โดยที่ค่าในแต่ละระดับ จะกำหนดว่าเป็น Column และ Row heading ซึ่งผู้ใช้จะเห็นเมื่อทำการ browse cube ใน Tabular Browser

4.2.3.4 การสร้าง Cubes

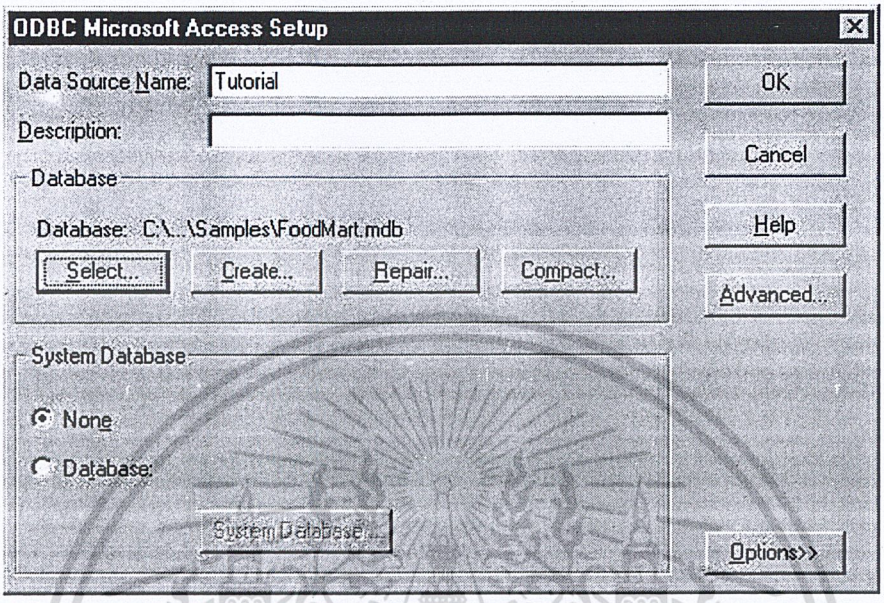
การสร้าง Cube ใน OLAP Manager นั้น จะมี Wizard มาให้ใช้งาน ทำให้สะดวกขึ้นอย่างมาก ขั้นตอนต่างๆใน Wizard มีดังนี้

ทำการติดตั้งการติดต่อกับ Data Source (Setting Up Data Source Connection)

ก่อนจะทำงานกับ OLAP Manager จะต้องทำการเชื่อมต่อ (Connect) กับแหล่งข้อมูลใน ODBC Data Source Administrator

- ติดตั้ง Data Source Name (DNS)

- คลิกเลือก ODBC Data Source จาก Control Panel
- จากแท็บ System DSN คลิก Add
- เลือก Microsoft Access Driver (*.mdb) แล้วใส่ชื่อของ Data Source เป็น Tutorial แล้วเลือก Database



รูปที่ 4.9 แสดงการติดตั้ง Data Source Name

- ภายใน Select Database dialog box เลือกไปที่โฟลเดอร์ C:\Program Files\OLAP Services\Samples และเลือก FoodMart.mdb

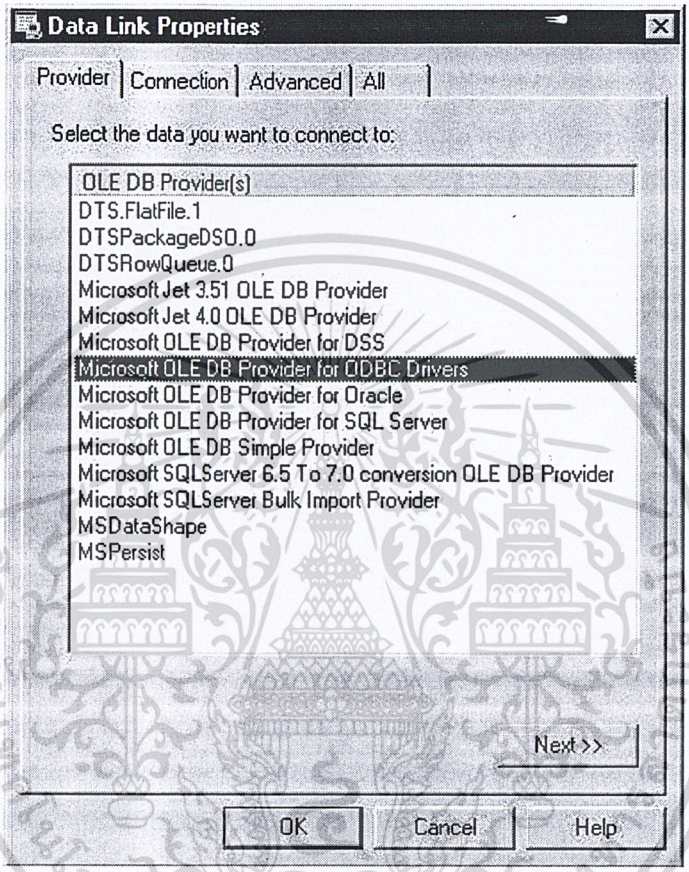
ทำการติดตั้ง Database และ Data Source (Setting Up Database and Data Source)

- ติดตั้งฐานข้อมูล

- ภายใน OLAP Manager Tree View ให้เลือกชื่อเซิร์ฟเวอร์ที่ต้องการติดต่อกับ OLAP Server แล้วสร้างฐานข้อมูลใหม่ที่ชื่อ Tutorial ขึ้นมาซึ่งประกอบด้วย 3 โฟลเดอร์คือ Cubes, Virtual Cubes และ Library ดูได้จากรูปที่ 4.6

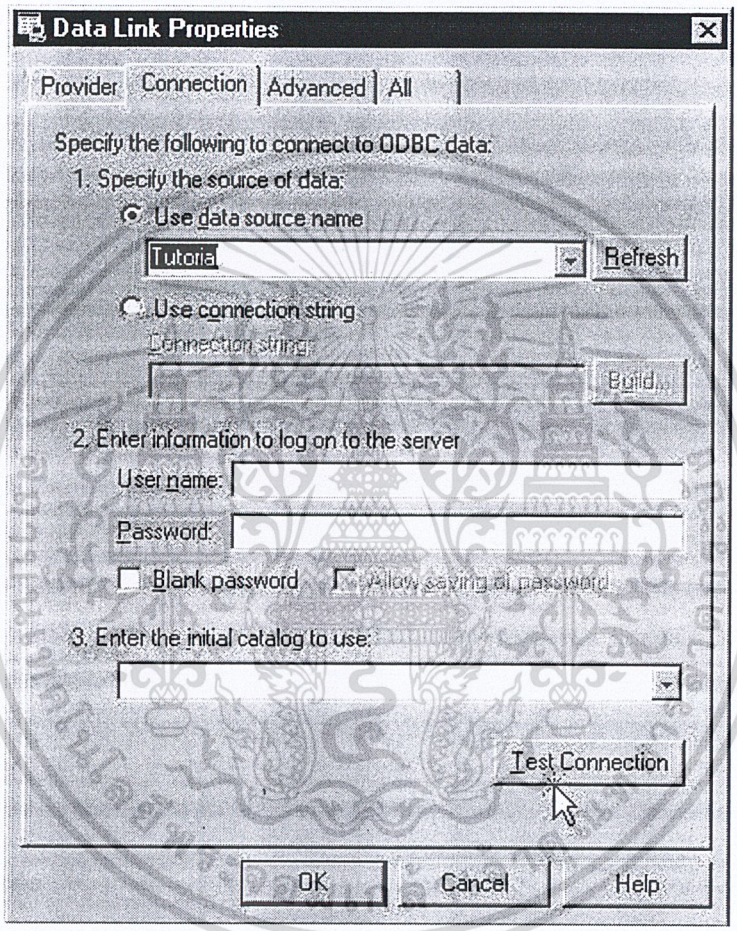
- ติดตั้ง Data Source

- ทำการสร้าง Data Source ใหม่ขึ้นที่โฟลเดอร์ Library ดังรูปที่ 4.10



รูปที่ 4.10 แสดงการสร้าง Data Source ขึ้นมาใหม่

- ใน Data Link Properties ให้เลือก Microsoft OLE DB Provider for ODBC Drivers แล้วเลือกชื่อ Data Source เป็น Tutorial จากนั้นก็ลองติดต่อด้วยการทดสอบการเชื่อมต่อ (Test Connection) ดังรูปที่ 4.11

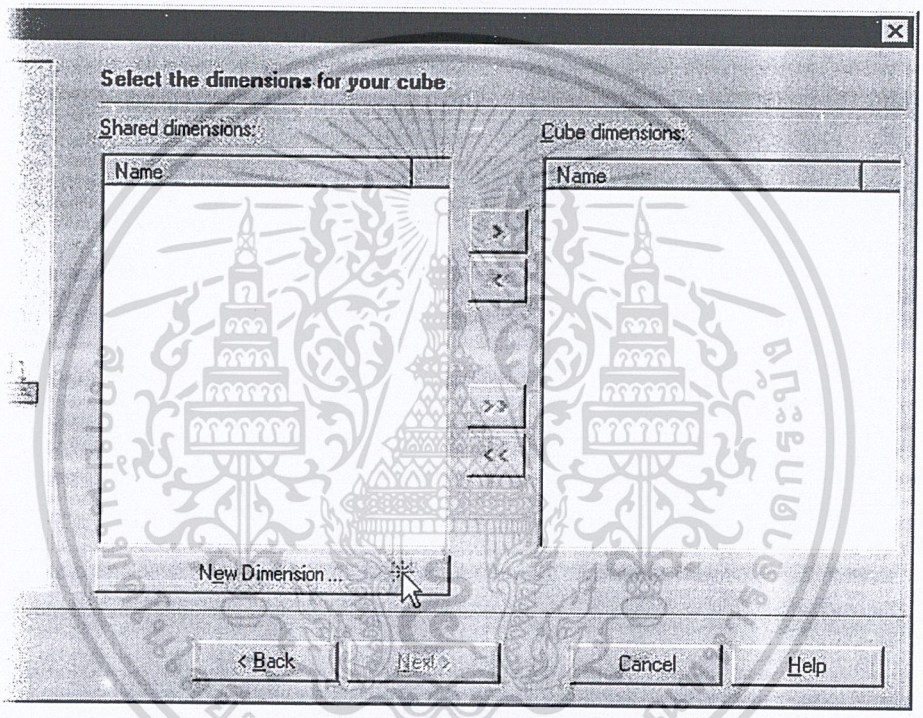


รูปที่ 4.11 แสดงการทดสอบการเชื่อมต่อกับ Data Source ที่เลือกมา

ลงมือสร้าง Cube (Building Cube)

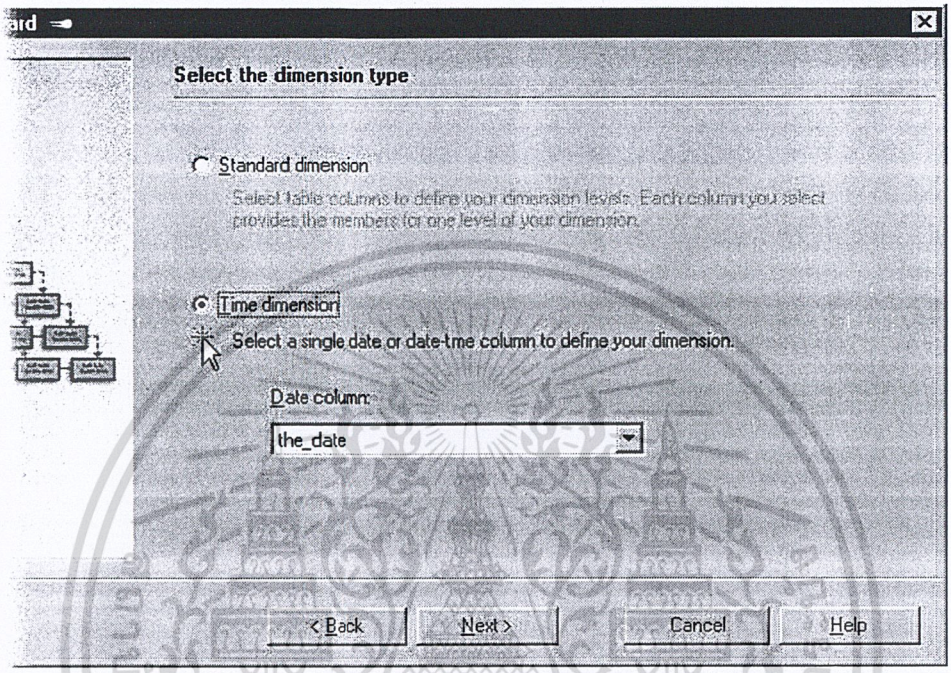
ในขั้นตอนนี้ จะใช้ Cube Wizard เข้ามาช่วยสร้าง Cube ซึ่ง Cube ที่จะสร้างนี้จะให้เป็น Cube ของ Marketing Department เพื่อนำมาใช้ในการวิเคราะห์ค่า Sales

- ทำการเปิด Cube wizard จาก Tutorial database
- ทำการใส่ตัววัดค่าที่จะใช้ใน Cube โดยเลือกตัววัดค่าในขั้นตอนของการเลือกตารางแฟ็ค
- ทำการสร้าง Time dimension ในขั้นตอนที่ให้เลือกไดเมนชัน ให้คลิกที่ New Dimension ดังรูปที่ 4.12



รูปที่ 4.12 แสดงการเลือกไดเมนชันต่างๆ ที่จะใช้ใน Cube

ใน Dimension Wizard เข้าไปที่ New dimension ให้เลือกที่ A Single Dimension table (flat or star schema) และสามารถคลิกที่ time_by_day table ได้ ส่วนขั้นตอนที่ให้เลือกชนิดของไดเมนชันนั้นก็เลือกไปที่ Time dimension ดังรูปที่ 4.13

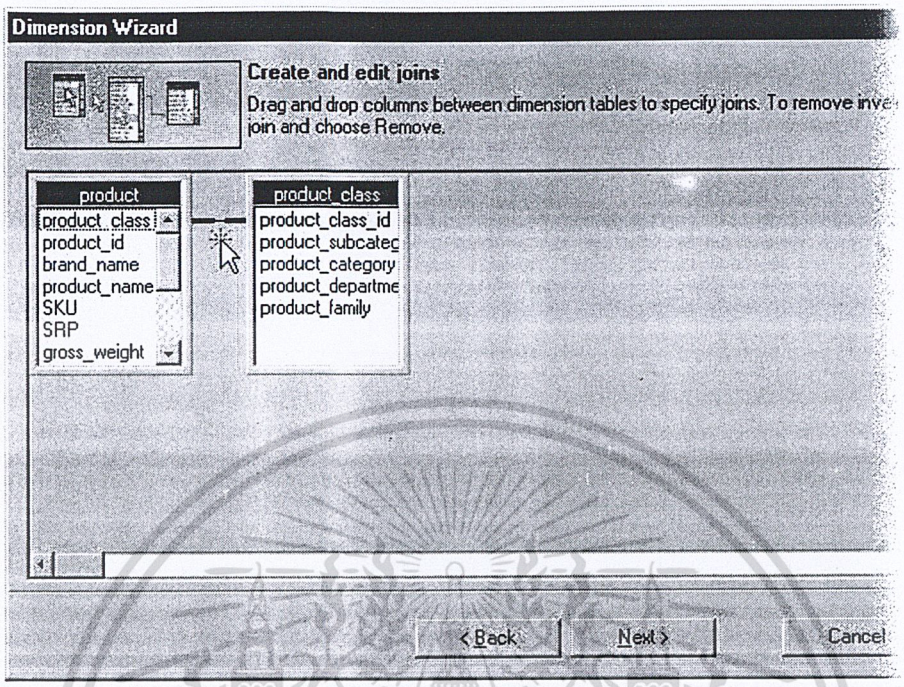


รูปที่ 4.13 แสดงการเลือกชนิดของไดเมนชัน

ในการกำหนด Levels ของไดเมนชันก็สามารถทำได้ โดยเลือกไปที่ Select time levels แล้วก็เลือกได้ตามต้องการว่าจะให้เป็น Year, Quarter หรือ Month

ขั้นตอนสุดท้ายก็คือ การตั้งชื่อของไดเมนชันที่เพิ่งจะสร้างขึ้นมาใหม่ ในที่นี้ให้ชื่อว่า " Time "

- ทำการสร้าง Product dimension โดยเข้าไปที่ New dimension อีกครั้ง แล้วเลือกที่ Multiple dimension tables (snowflake schema) จากนั้นก็เลือกตารางที่จะให้เป็น Product dimension ในที่นี้จะเลือก product และ product_class จากนั้นทั้ง 2 ตารางที่เลือกมานั้นจะแสดงการเชื่อมต่อกันในขั้นตอน Create and edit joins ดังรูปที่ 4.14



รูปที่ 4.14 แสดงการเชื่อมต่อกันของไดเมนชันที่เลือก

จากนั้นก็สามารถกำหนด Levels ให้กับไดเมนชันได้อีก โดยการดับเบิ้ลคลิกที่คอลัมน์ในตารางที่เลือกมา ชื่อของ Levels ก็จะปรากฏใน Dimension levels ตามลำดับการเลือก

ขั้นตอนสุดท้ายให้ตั้งชื่อไดเมนชัน ในที่นี้เป็น " Product " แล้วก็ให้เลือกช่อง Share this dimension with other cubes ออกไป

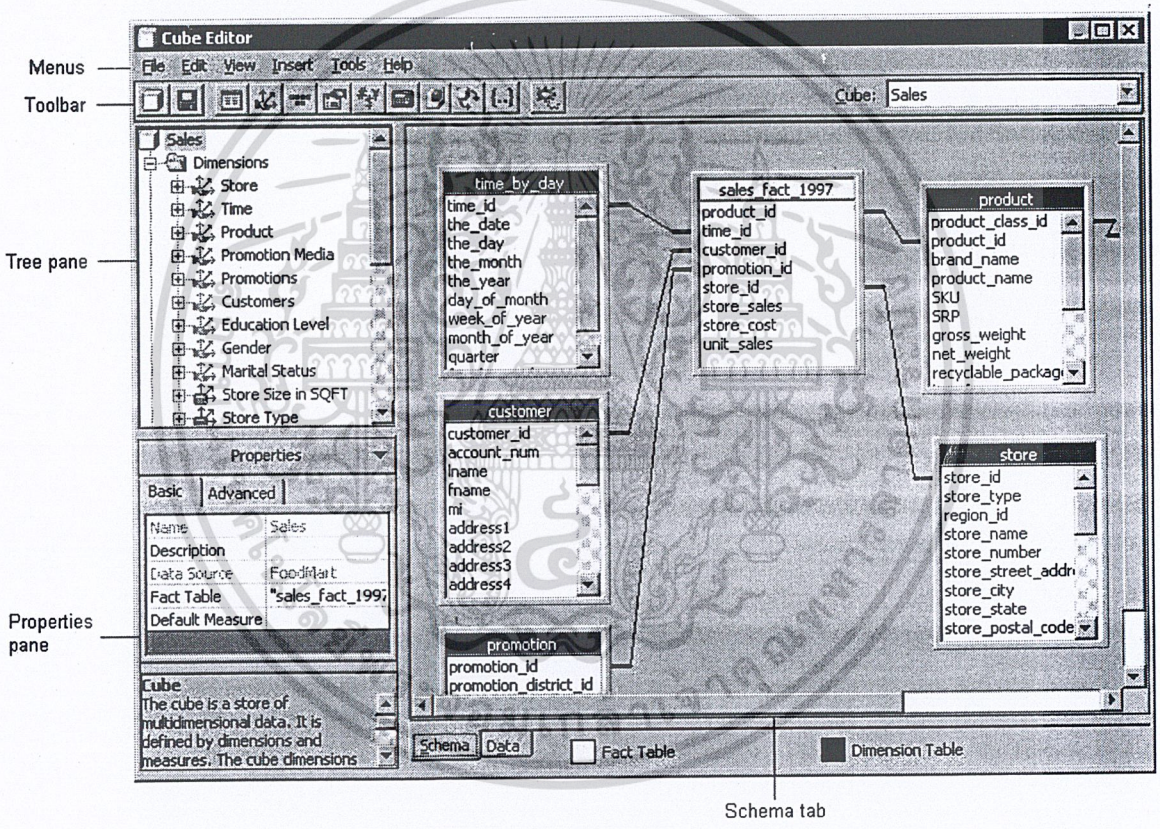
- ทำการสร้าง Store dimension เข้าไปที่ New dimension เลือก A Single Dimension table (flat or star schema) จากนั้นเลือก Store ส่วน Levels ให้เลือกเป็น store_country, store_state, store_city, และ store_name columns ตามลำดับ แล้วตั้งชื่อไดเมนชันนี้ว่า " Store " แล้วก็ให้เลือกช่อง Share this dimension with other cubes ออกไป เช่นกัน

- ทำการสร้าง Promotion dimension เข้าไปที่ New dimension เลือก A Single Dimension table (flat or star schema) จากนั้นเลือก Promotion ส่วน Levels ให้เลือกเป็น Media_type และ promotion_name columns ตามลำดับ แล้วตั้งชื่อไดเมนชันนี้ว่า " Promotion " แล้วก็ให้เลือกช่อง Share this dimension with other cubes ออกไป เช่นกัน

ทั้ง Time, Product, Store และ Promotion Dimension ที่เลือกมาทั้งหมดนั้น จะถูกเก็บอยู่ในลิสต์ของ Cube Dimension

เป็นอันเสร็จสิ้นขั้นตอนของการสร้าง Cube ให้ตั้งชื่อ Cube นี้ว่า " Sales "

- การแก้ไข Cube โดยใช้ Cube editor wizard ในหน้าต่างที่แสดง Schema ของ Cube editor เราจะเห็นตารางแฟ็ค เป็นสีเหลือง ส่วนไดเมนชันที่มาเชื่อมต่อ จะเป็นสีน้ำเงิน และใน Cube editor tree view จะแสดงโครงสร้างของ Cube ในแบบลำดับขั้นต้นไม้ (Hierarchical Tree) ดังรูปที่ 4.15

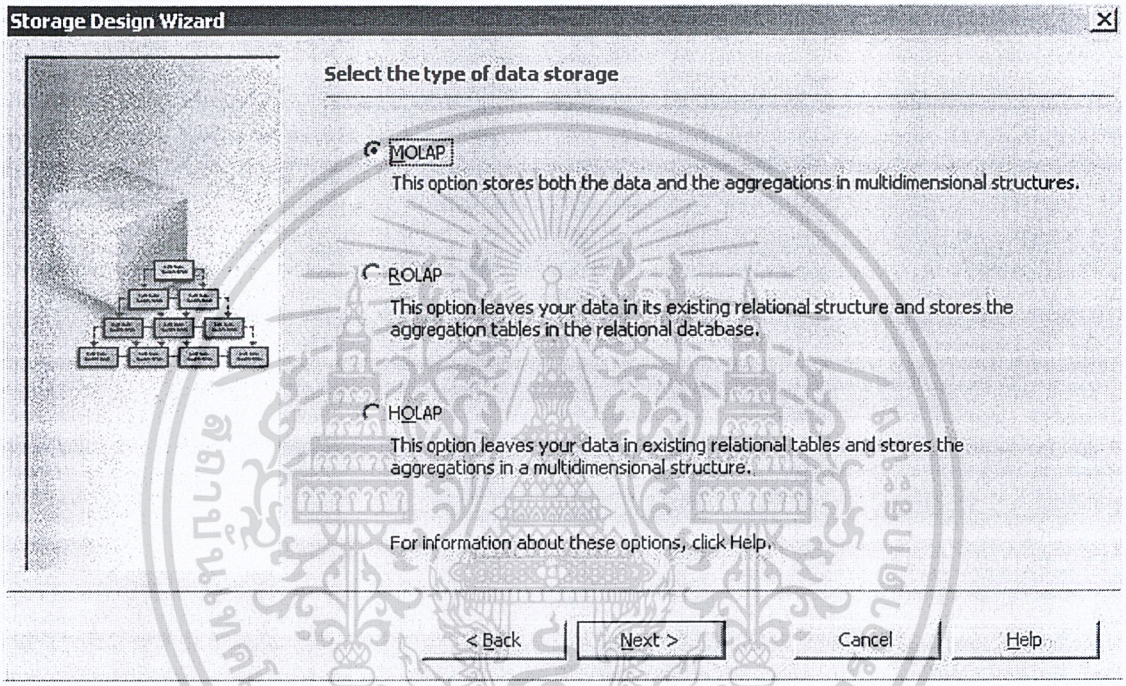


รูปที่ 4.15 แสดงหน้าต่างของ Cube Editor Wizard

ใน Cube editor นี้ สามารถที่จะสร้างไดเมนชันที่ต้องการใช้เพิ่มได้ แต่ไดเมนชันที่สร้างเพิ่มนี้จะเป็น Private dimension คือ ใช้ได้เฉพาะใน Cube ที่กำลังใช้งานอยู่เท่านั้น ไม่สามารถใช้ร่วมกับ Cube อันอื่นได้

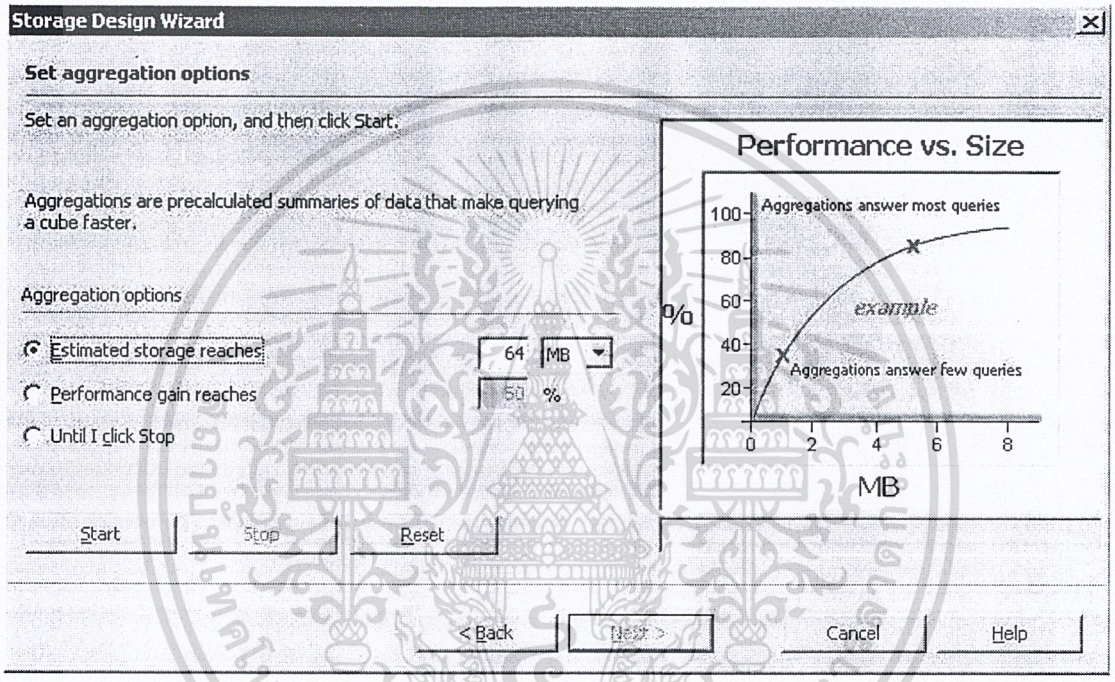
การออกแบบการจัดเก็บข้อมูลและการประมวลผล Cube (Designing Storage and Processing Cube)

OLAP Services นั้น เราสามารถปรับประสิทธิภาพและเวลาที่ใช้ในการตั้งคำถามได้ โดยใช้ Storage Design Wizard ระบุวิธีการจัดเก็บได้ 3 ชนิด คือ MOLAP, ROLAP หรือ HOLAP ดังรูปที่ 4.16



รูปที่ 4.16 แสดงการเลือกชนิดของการจัดเก็บข้อมูลใน Cube

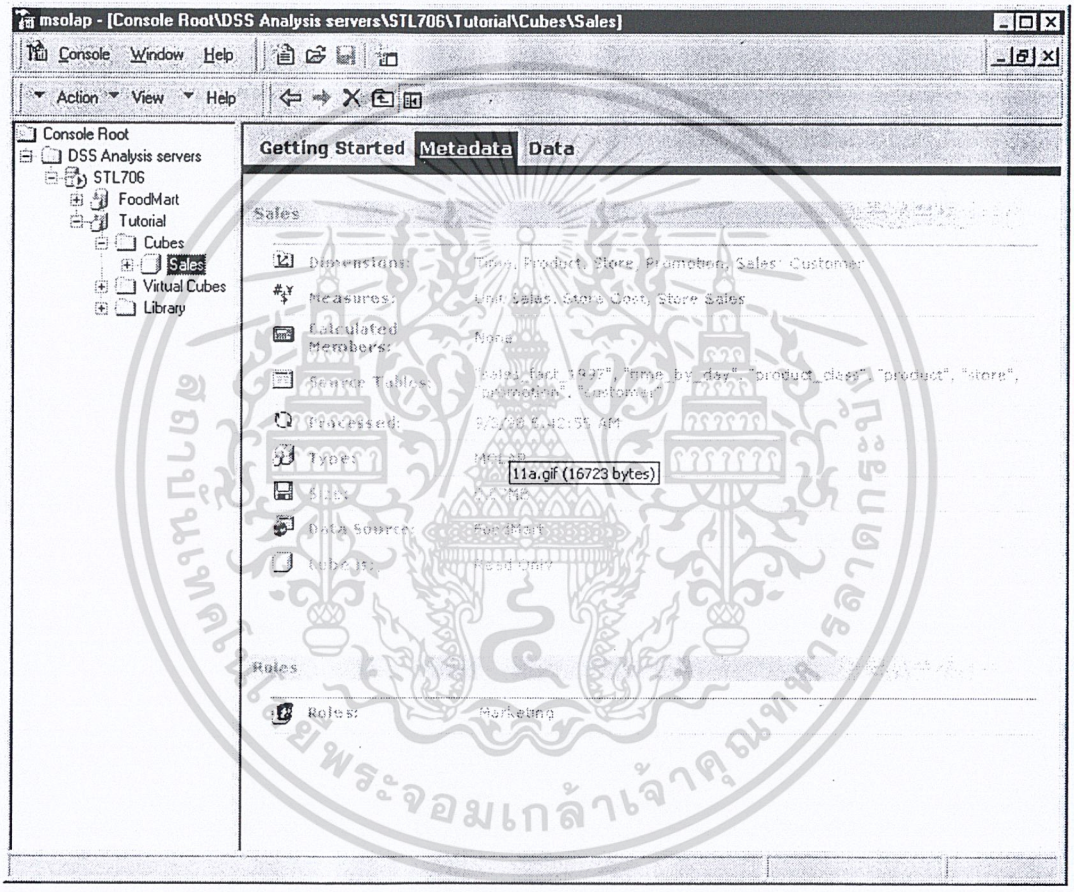
หลังจากที่เลือกชนิดของการจัดเก็บข้อมูลแล้ว สามารถดูกราฟที่แสดงถึงประสิทธิภาพ (Performance) และขนาด (Size) ของ Cube ได้ดังรูปที่ 4.17



รูปที่ 4.17 แสดงการวัดประสิทธิภาพและขนาดของ Cube ที่สร้างขึ้นมา

Viewing Metadata for Cube

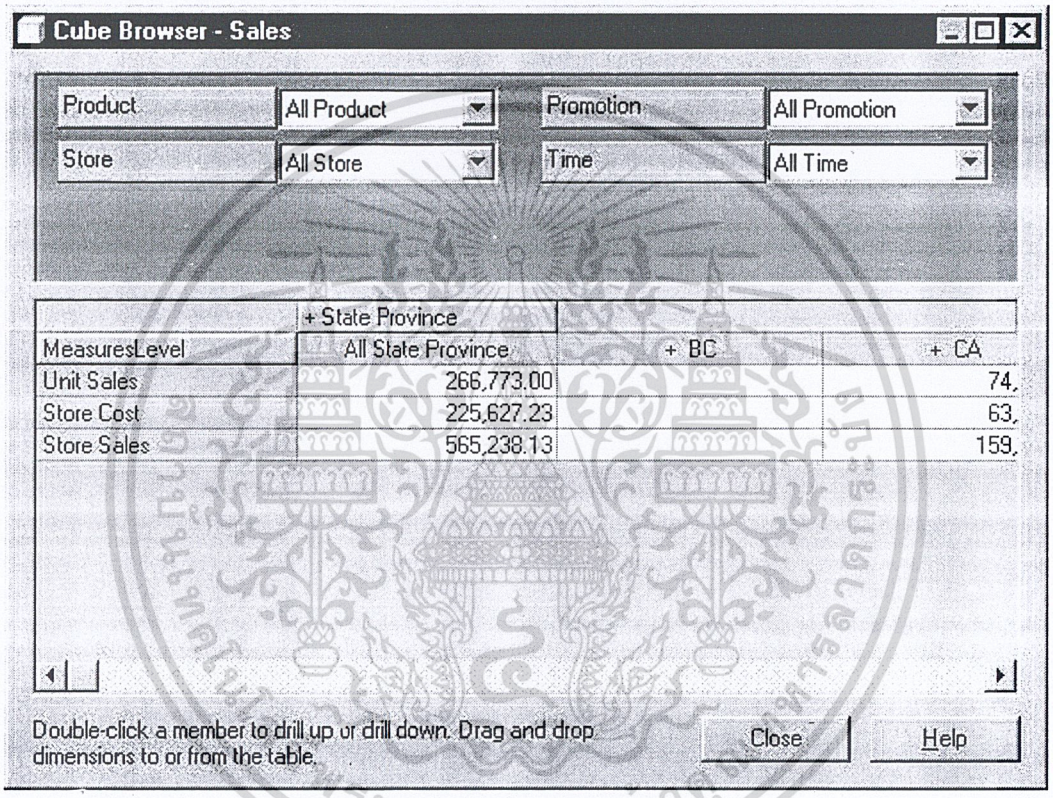
ใน OLAP Services สามารถดูรายละเอียดข้อมูลที่มีอยู่ใน Cube หรือ Metadata ได้แก่ Dimensions, Measures, Calculated Members, Source Tables, Processed, Type, Size และ Data Source ดังแสดงในรูปที่ 4.18



รูปที่ 4.18 แสดงรายละเอียดข้อมูล (Metadata) ใน Cube

Browsing Cube's Data

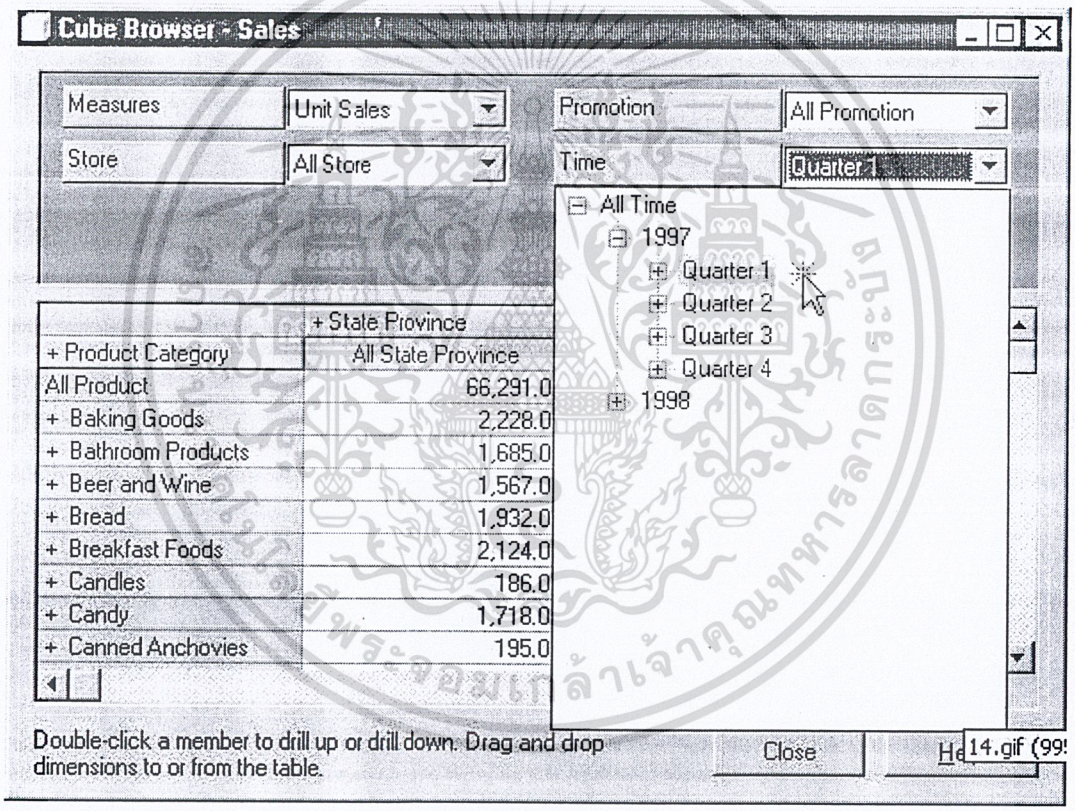
สามารถใช้ Cube Browser เพื่อดูข้อมูลในลักษณะต่างๆ กันได้ สามารถใส่จำนวนไดเมนชันที่จะดูข้อมูลและสามารถที่จะ Drill down ในรายละเอียดที่มากขึ้นได้ด้วย หน้าต่าง Cube Browser แสดงได้ดังรูปที่ 4.19



รูปที่ 4.19 แสดงหน้าต่างของ Cube Browser

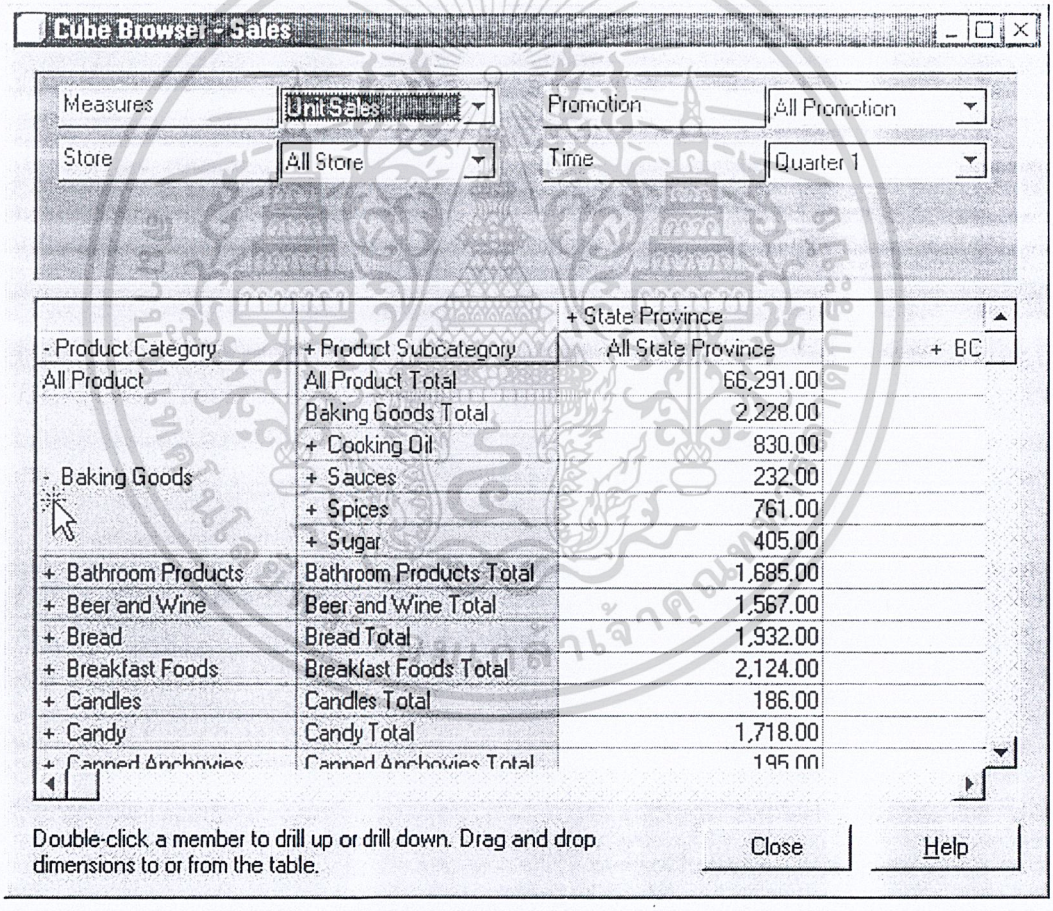
และยังสามารถเพิ่มไดเมนชันที่ต้องการดู เข้าไปแทนที่ไดเมนชันที่แสดงอยู่ใน Grid ได้ เพียงแค่ลากไดเมนชันจากส่วนของ Palette ด้านบน ลงมาวางแทนไดเมนชันใน Grid ส่วนไดเมนชันที่ถูกแทนที่ก็จะถูกนำขึ้นไปเก็บไว้ที่ Palette แทน

ในส่วนของ Time Dimension เราสามารถเลือกดูลักษณะของเวลาได้ทั้งแบบ Year, Quarter หรือ Month ดังแสดงในรูปที่ 4.20



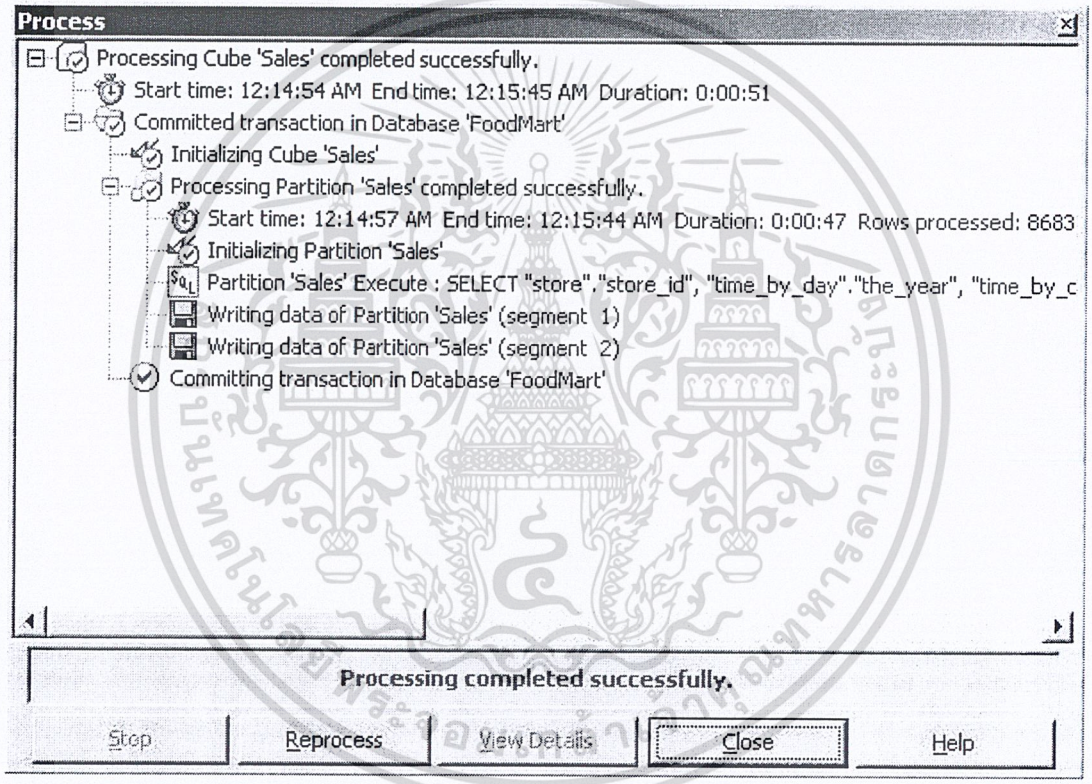
รูปที่ 4.20 แสดงการเลือกดูข้อมูลในระดับต่างๆ ของ Time Dimension

ส่วนการทำ Drill down ข้อมูล ก็สามารถทำได้ง่ายเช่นกัน เพียงแค่ดับเบิ้ลคลิกที่ Cell ใน Grid ได้ดังรูปที่ 4.21



รูปที่ 4.21 แสดงการ Drill-down ข้อมูลใน Cube Browser

ในส่วนสุดท้ายที่ Wizard มีมาให้ ก็คือ การประมวลผล Cube (Process Cube) ขั้นตอนนี้จะมีการแจ้งเวลาที่ใช้ในการประมวลผลทั้งหมดให้ทราบ เมื่อประมวลผลเสร็จแล้ว จะได้ผลดังรูปที่ 4.22



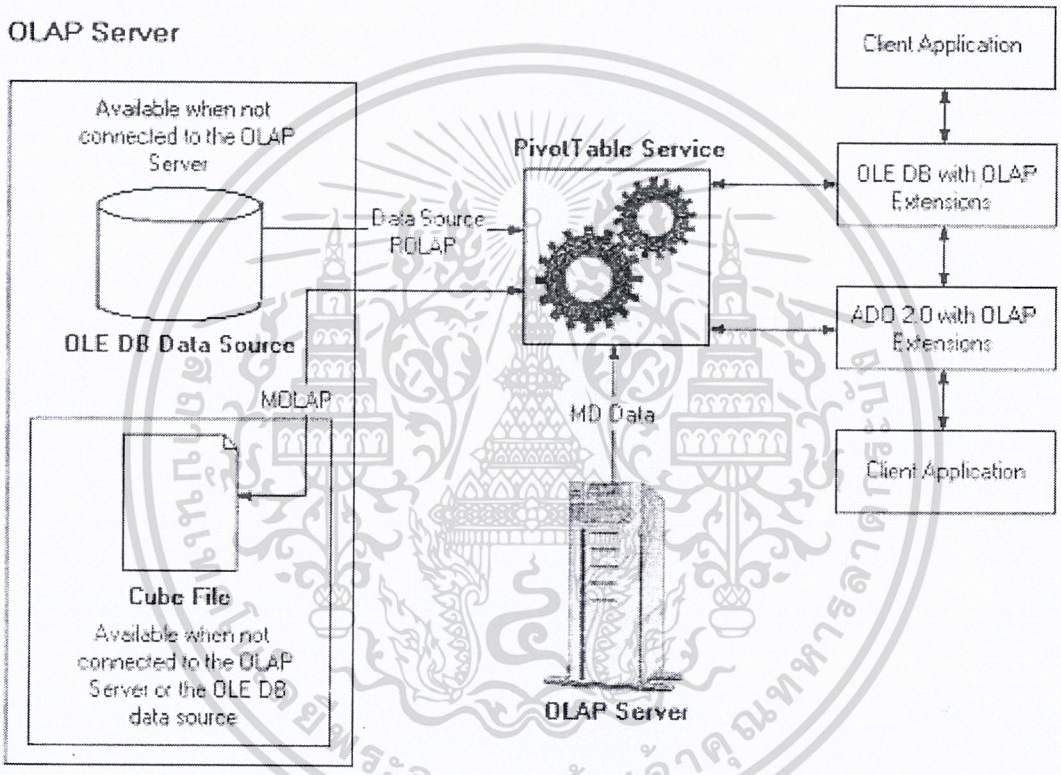
รูปที่ 4.22 แสดงผลของการประมวลผล Cube ที่ทำการสร้างขึ้นมา

บทที่ 5

OLAP Client Architecture

5.1 OLAP Client Architecture

หัวใจของสถาปัตยกรรมด้าน Client คือ PivotTable Service จะเป็นตัวติดต่อที่อยู่บนพื้นฐานของ Component Object Model (COM) เพื่อให้ Client Application เข้าใช้ข้อมูลที่อยู่ใน OLAP Server แอพลิเคชันเหล่านี้จะติดต่อกับ PivotTable Service ได้โดยใช้ OLE DB 2.0 หรือใช้ Microsoft ActiveX Data Objects (ADO 2.x) ดังรูปที่ 5.1



รูปที่ 5.1 โครงสร้างของ PivotTable Service Architecture

5.2 PivotTable Service

PivotTable Service เป็นส่วนที่ทำหน้าที่ในการติดต่อสื่อสารระหว่าง Analysis Server กับอินเทอร์เฟสของแอพลิเคชันทางฝั่ง Client โดยแอพลิเคชันทางฝั่ง Client จะเชื่อมต่อกับ PivotTable Service โดยใช้ OLE DB สำหรับโปรแกรมที่ใช้ภาษา C++ หรือใช้ Microsoft ActiveX Data Objects (ADO) ซึ่งมีออบเจ็กต์โมเดล (object model) แบบ Component Object Model (COM) สำหรับโปรแกรมที่ใช้ภาษาวิชวลเบสิก

PivotTable Service สามารถสร้างไฟล์ local cube ที่เก็บข้อมูลจาก Cube บนเครื่องเซิร์ฟเวอร์หรือข้อมูลจาก OLE DB relational databases , Local cubes สามารถเก็บข้อมูลที่เป็น multidimensional cube file บน Client

และยังสามารถใช้ในการวิเคราะห์แบบ Off-line ได้ นั่นคือ ถ้า Local cubes มีการเก็บข้อมูลแบบ Multidimensional OLAP (MOLAP) ก็จะสามารถตั้งคำถาม (Query) ได้ โดยไม่ต้องทำการเชื่อมต่อไปยัง Analysis Server

PivotTable Service ถูกออกแบบมาสำหรับการวิเคราะห์ข้อมูลทั้งแบบ On-line และ Off-line เมื่อมีการร้องขอมาจาก Client ทาง OLAP Server ก็จะทำการประมวลผลการร้องขอเหล่านั้นและสร้างเซตข้อมูล (data set) จาก Pre-stored Aggregate ถ้าเป็นไปได้ หรือจากการประมวลผลการรวบรวม (Aggregation) ใหม่ เมื่อทำเสร็จ OLAP Server ก็จะส่งเซตข้อมูลกลับไปให้ยัง Client ที่ร้องขอมา

PivotTable Service จะยอมให้ Client Application เข้าใช้ OLE DB data source เมื่อยังไม่มีการติดต่อกับ OLAP Server , Data Source เหล่านี้มักจะสำรองฐานข้อมูลเชิงปฏิบัติการ (Operational database) หรือดาต้ามาร์ท (Data Mart) บางส่วนที่อยู่ในฝั่ง Client ซึ่ง PivotTable Service จะยอมให้มีการสร้างและเข้าถึง Local cube บน Client

5.2.1 ลักษณะเด่นของ PivotTable Service

- PivotTable Service มีความสามารถในการปรับปรุงประสิทธิภาพของ Client Application ในการเข้าถึงข้อมูลของ OLAP
- PivotTable Service จะถูกรวมเข้าไปใน Microsoft Excel 2000 ซึ่งจะทำให้ Excel กลายเป็นทูลส์ที่มีประสิทธิภาพในการนำเสนอข้อมูล OLAP
- PivotTable Service สามารถสร้าง Cube ของฝั่ง Client ได้ ซึ่งยอมให้นำไปวิเคราะห์บนเครื่องคอมพิวเตอร์เครื่องใดก็ได้ ในขณะที่ไม่ได้เชื่อมต่อกับระบบเครือข่ายและ Data Source หลัก , Cube ของฝั่ง Client มักจะเก็บไว้ในไฟล์ Cube บนเครื่อง Client ไฟล์เหล่านี้สามารถเปิดและอ่านได้โดย OLAP Manager
- เนื่องจาก PivotTable Service ถูกติดตั้งบนเครื่อง Client เช่น เมื่อทำการติดตั้ง Microsoft Excel ผู้พัฒนาโปรแกรมก็สามารถสร้างแอพลิเคชัน โดยการเขียนโปรแกรมด้วยภาษาใดๆ ก็ตามที่ถนัด เพื่อให้เข้าใช้ข้อมูล OLAP หรือ Relational data source โดยใช้เทคโนโลยีของ OLE DB
- PivotTable Service ทำงานกับ OLAP Services เหมือนกับเป็นส่วนหนึ่งของสถาปัตยกรรมของเซอร์วิสเหล่านี้ ดังนั้น PivotTable Service จึงยอมให้หลายๆ Client สามารถเข้าใช้ Cube อันเดียวกันได้ โดยแบ่งการประมวลผลตามผู้ใช้
- PivotTable Service จะสนับสนุน Multidimensional Expressions (MDX) ซึ่งเป็นภาษาที่ใช้ในการตั้งคำถาม (Query) และอนุญาตให้ทำการเข้าใช้ดาต้าคิวบ์, ไดเมนชันและส่วนอื่นๆ ได้โดยใช้ SQL Statement
- PivotTable Service อนุญาตให้ทำการดาวน์โหลดข้อมูลจาก Data Source และเก็บข้อมูลในโครงสร้างแบบมัลติไดเมนชันแนล บนโลคอลคอมพิวเตอร์ (Local Computer) เพื่อใช้สำหรับการวิเคราะห์แบบ Off-line

5.2.2 คอมโพเนนต์ที่สำคัญที่ใช้กับ PivotTable Service

ในแอพลิเคชันทางฝั่ง Client ที่ใช้ PivotTable Service จะต้องมีไฟล์ที่เป็น dynamic-link libraries (DLL) ซึ่งจะใช้ไฟล์ตัวใดขึ้นอยู่กับลักษณะงานที่นำไปใช้

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 5.1 แสดง File set ของไฟล์คอมโพเนนต์

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 5.2 แสดง File set ที่ใช้กับลักษณะงานในแอพลิเคชันทางฝั่ง Client

5.2.3 การติดตั้งและลงทะเบียนคอมโพเนนต์

ก่อนจะทำการติดตั้ง PivotTable Service จะต้องติดตั้งไฟล์ Microsoft Data Access Components (MDAC) เสียก่อน แล้วจึงทำการติดตั้งและลงทะเบียนคอมโพเนนต์ที่ใช้ให้ถูก location ดังต่อไปนี้

สำหรับคอมโพเนนต์ที่เป็น Resource-link libraries (.rll) ได้แก่ Msolap80.rll, Olapuir.rll และ Msdmine.rll จะต้องติดตั้งไปที่ C:\Program Files\Common Files\System\OLE DB\Resources\1033

สำหรับคอมโพเนนต์ที่เป็น Dynamic-link libraries (.dll) ได้แก่ Msolap80.dll, Msolui80.dll, Msmdgd80.dll, Msmdcb80.dll, Msmdun80.dll, Msdmine.dll และ Msdmeng.dll จะต้องลงทะเบียนโดยใช้ Regsvr32.exe

5.2.4 ลักษณะของการเชื่อมต่อของ PivotTable Service

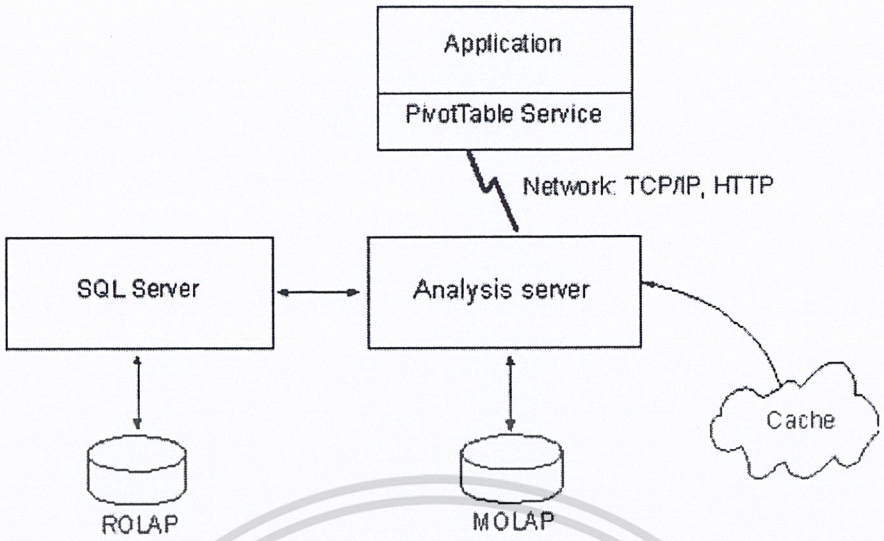
PivotTable Service สามารถเชื่อมต่อได้ 3 รูปแบบ คือ

- เชื่อมต่อกับ SQL Server 2000 Analysis Services
- เชื่อมต่อกับ OLE DB Provider
- เชื่อมต่อกับ Local cube

5.2.4.1 เชื่อมต่อกับ SQL Server 2000 Analysis Services

ทำให้รู้วิธีการเชื่อมต่อและคุณสมบัติที่พร้อมใช้งานบนแอพลิเคชันทางฝั่ง Client เช่น ถ้าจะเชื่อมต่อผ่านทางอินเตอร์เนต ก็ให้ใช้ HTTP และควรพิจารณาถึงคุณสมบัติของ User ID และ Password

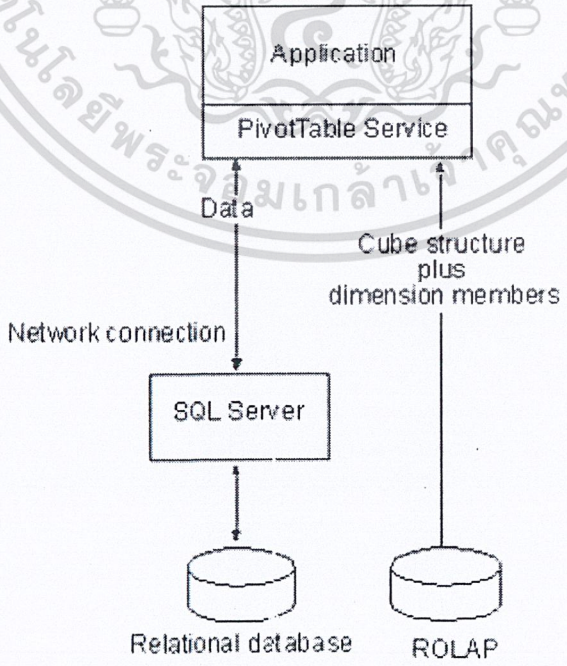
เมื่อใช้ PivotTable Service แสดงข้อมูลจาก Analysis Services, PivotTable Service จะติดต่อกับ Remote Analysis Server ผ่านระบบเครือข่ายที่มีโพรโตคอล (protocol) เป็น TCP/IP หรือ HTTP ดังรูปที่ 5.2



รูปที่ 5.2 แสดงถึงการเชื่อมต่อระหว่าง PivotTable Service กับ Analysis Services

5.2.4.2 เชื่อมต่อกับ OLE DB Provider

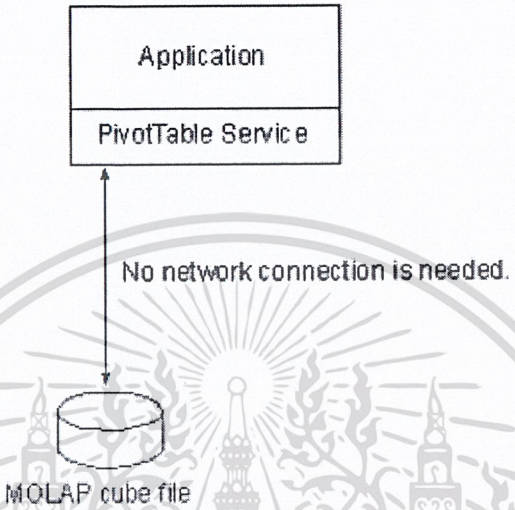
เมื่อแสดงข้อมูลจาก Relational OLAP (ROLAP) Local Cube : เมื่อทำการเชื่อมต่อกับ ROLAP Cube File จะต้องเชื่อมต่อกับ relational data provider ในการแสดงข้อมูลจะต้องติดต่อไปยัง tabular data provider กระบวนการนี้จะถูกส่งไปยังแอพลิเคชันทางฝั่ง Client



รูปที่ 5.3 แสดงการเชื่อมต่อกับ Local ROLAP Cube

5.2.4.3 เชื่อมต่อกับ Local Cube File

ในการแสดงข้อมูลจาก MOLAP Cube, PivotTable Service จะเชื่อมต่อกับ local cube file ในแบบเดียวกับที่เชื่อมต่อกับ data source อื่นๆ โดย PivotTable Service จะประมวลผลจากคำถาม (query) แล้วส่งค่าที่ได้คืนกลับไปยังแอพลิเคชัน ซึ่งแอพลิเคชันทางฝั่ง Client สามารถเข้าถึงไดเมนชัน, เลเวล, พร็อพเพอร์ตี้ (property) และ particular cube ได้โดยที่ไม่ต้องทำการเชื่อมต่อกับ Remote Server



รูปที่ 5.4 แสดงการเชื่อมต่อระหว่าง PivotTable Service กับ MOLAP Cube File

5.2.5 Client Operations ใน PivotTable Service

- การเชื่อมต่อไปยัง Data Source
- การแสดงโครงสร้างของข้อมูล (Retrieving Schema Information)
- การแสดงข้อมูล (Retrieving Data)

- **การเชื่อมต่อไปยัง Data Source**

ขั้นแรกของการใช้งาน PivotTable Service คือ การเชื่อมต่อกับ Data Source โดยใช้ *Connection object* หรือ *ActiveConnection property* ของ *Catalog object* พารามิเตอร์ (parameter) สำหรับการเชื่อมต่อนี้สามารถใช้ *Connection String* กำหนดได้

ADO Connection Object

เมธอด (Method) *Open* ของ Connection Object ได้รวมเอาพารามิเตอร์สำหรับใช้ในการเชื่อมต่อเข้าไปใน ConnectionString Property เมื่อเมธอดนี้ถูก executed ก็จะทำการเชื่อมต่อกับ Data Source

Syntax ของเมธอด Open : *connect.Open ConnectionString, UserId, Password, OpenOptions*

การเชื่อมต่อไปยัง Analysis Service

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

- **การแสดงโครงสร้างของข้อมูล (Retrieving Schema Information)**

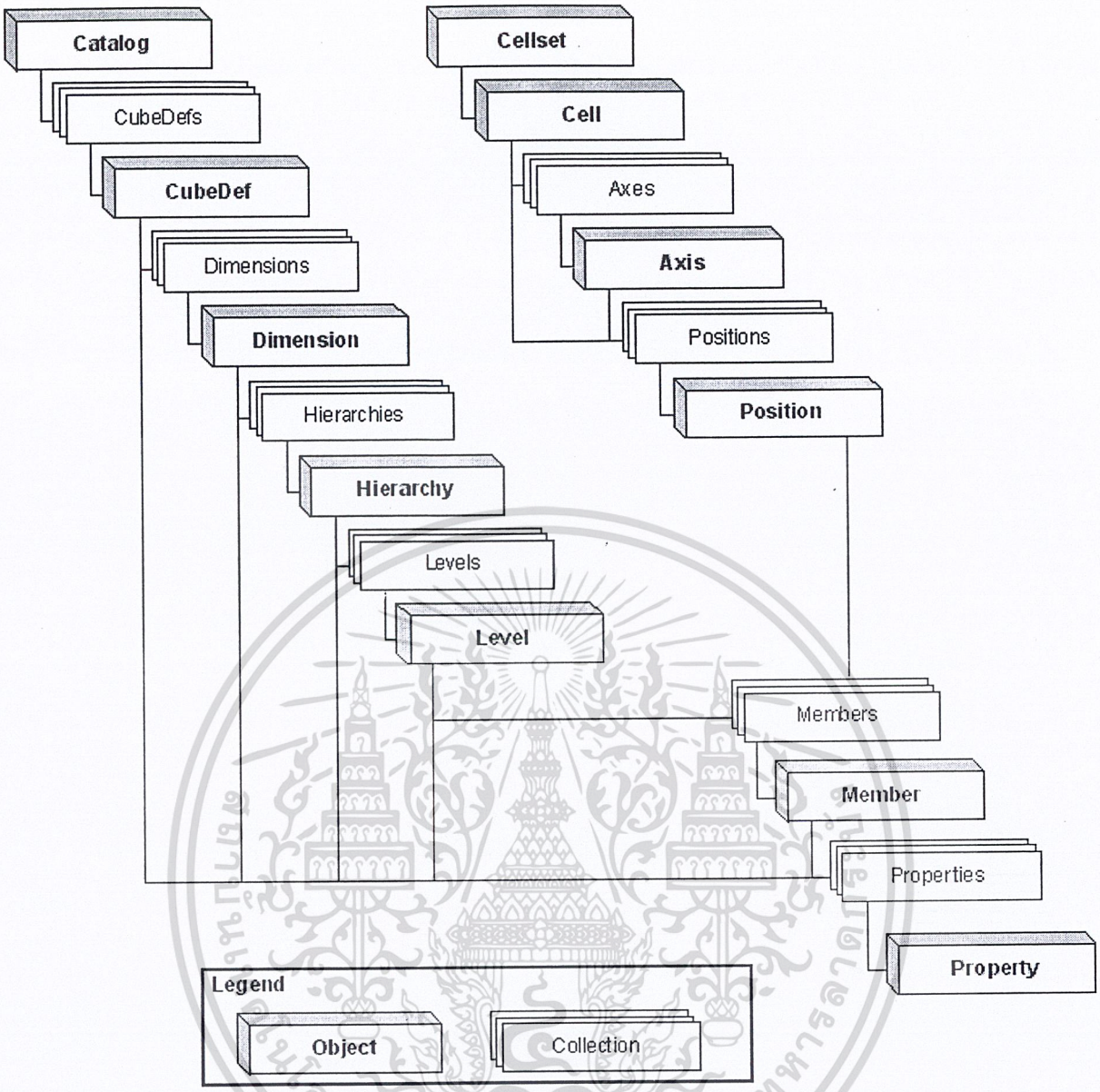
สามารถใช้ Microsoft ActiveX Data Objects (Multidimensional) (ADO MD), ADO หรือ OLE DB ในการแสดง schema rowsets โดยใช้ PivotTable Service

ในการแสดงโครงสร้างของข้อมูลของ Cube จะใช้ *CubeDef object* ใน ADO MD หรือใช้เมธอด OpenSchema ใน ADO CubeDef object จะมีโครงสร้าง cube เป็นแบบ hierarchy จะมีบางข้อมูลเกี่ยวกับ cube ที่ไม่ได้เก็บอยู่ใน CubeDef object เช่น การกำหนด actions และ cell formulas ซึ่งจะต้องใช้เมธอด OpenSchema ในการแสดงข้อมูลเหล่านี้

ในการแสดง schema rowsets จะใช้ ADO หรือ OLE DB สำหรับใน ADO จะใช้เมธอด OpenSchema ของ Connection object ส่วนใน OLE DB จะใช้อินเทอร์เฟส IDBSchemaRowset COM

การใช้งาน CubeDef Object

CubeDef Object จะเก็บไดเมนชันของ Local cube โดยใช้ *Dimensions collection* ของตัวมันเอง ซึ่งมีรูปแบบเป็น *Hierarchies collection* ดังรูปที่ 5.5



รูปที่ 5.5 แสดง CubeDef Object Model ที่ใช้ ADO MD

- **การแสดงข้อมูล (Retrieving Data)**

จะมีเมธอดสำหรับใช้แสดงข้อมูลอยู่ 2 เมธอด คือ *Cellset object* และ *Recordset objects* สามารถใช้ ADO MD ในการแสดง Multidimensional Expression (MDX) query result จาก local cube โดยใช้ Cellset Object ได้ ส่วนการแสดง tabular result set จะใช้ ADO Command และ Recordset object

ตัวอย่างการใช้ Cellset object

จะใช้ *Connection object* ในการเชื่อมต่อกับ Analysis Server , *Source property* ของ Cellset object จะอยู่ในรูปแบบของ MDX query, *ActiveConnection property* ของ Cellset object จะเหมือนกับ ActiveConnection property ของ Connection object และมีการใช้เมธอด *Open* ในการแสดง actual result

Cellset object จะเก็บ collection ที่เรียกว่า Axes ซึ่งจะใช้อธิบายในแต่ละแกน จะมี 1 Axis object ใน collection นี้สำหรับแต่ละไดเมนชันที่ต้องการ แต่ละ Axis object จะเก็บ Positions collection ซึ่งเก็บข้อมูลในแต่ละแถว (row), คอลัมน์ (column), หน้า (page) และ result set

ตัวอย่าง source code

```
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 6
การออกแบบ พัฒนาและการใช้งานโปรแกรมประยุกต์

6.1 ทูลส์ที่นำมาใช้ในการพัฒนาโปรแกรมประยุกต์

ในโปรแกรมประยุกต์ที่สร้างขึ้นมา ได้เลือกใช้วิชวลเบสิก (Visual Basic) รุ่น 6.0 เป็นทูลส์ในการพัฒนาโปรแกรม เนื่องจากทูลส์ตัวนี้มีความเหมาะสมที่จะใช้ในการพัฒนาโปรแกรมที่เกี่ยวข้องกับงานด้านฐานข้อมูล (Database Application) เช่น ในเรื่องของการเข้าถึงข้อมูล วิชวลเบสิก รุ่น 6.0 นี้จะถูกเพิ่มความสามารถในการติดต่อกับฐานข้อมูลมากขึ้น โดยสามารถติดต่อเชื่อมโยงฐานข้อมูลที่เป็น Microsoft SQL Server หรือแม้กระทั่ง Oracle ซึ่งทำให้วิชวลเบสิกมีความโดดเด่นในด้านนี้ขึ้นมาด้วยคอมโพเนนต์ (Component) หรือ Object Model ใหม่ที่มีมาให้ ได้แก่

6.1.1 ADO (ActiveX Data Object)

การเข้าถึงข้อมูลที่เป็น OLE DB ระดับแอพลิเคชัน โดยใช้ ADO จะทำได้ง่ายและสะดวก ไม่ว่าจะเป็นฐานข้อมูลแบบรีเลชันแนล (relational) หรือนอน – รีเลชันแนล (non – relational) รวมไปถึงการเชื่อมโยงกับแหล่งข้อมูลแบบ ODBC ที่ใช้กันอยู่ในปัจจุบันด้วย หรือพูดอย่างง่ายๆ ก็คือ การเข้าถึงข้อมูลใดๆ ในระดับองค์กรสามารถทำได้ด้วยเทคโนโลยีการเข้าถึงข้อมูลแบบ ADO ส่วนคุณสมบัติของ ADO คือ ใช้งานง่าย, สะดวกในการติดตั้ง, ใช้ทรัพยากรเครือข่ายน้อย และมีขั้นตอนการติดต่อระหว่างแอพลิเคชันกับฐานข้อมูลน้อย จึงเป็นการลดความยุ่งยากด้วย หรือสรุปอีกอย่างหนึ่งได้ว่า เป็นเทคโนโลยีที่ใช้ทรัพยากรของระบบน้อย แต่มีประสิทธิภาพสูงในการเข้าถึงข้อมูล

6.1.2 ADO Data Control

คอนโทรลข้อมูล ADO ทำให้การเชื่อมโยงระหว่างคอนโทรลที่เป็น Data bound กับแหล่งข้อมูลเป็นไปได้อย่างรวดเร็ว เราสามารถควบคุมคุณสมบัติของคอนโทรลเหล่านี้ผ่านทางคุณสมบัติ Datasource และสามารถเชื่อมโยงกับทุกแหล่งข้อมูลที่สนับสนุนมาตรฐาน OLE DB ทั้งยังสามารถสร้างแหล่งข้อมูลได้เองด้วยการสร้าง Class Module นอกจากนี้การใช้ ADO ในการเชื่อมโยงข้อมูล ยังช่วยลดการเขียนคำสั่งลงไปมากทีเดียว

6.1.3 การสนับสนุน OLE DB

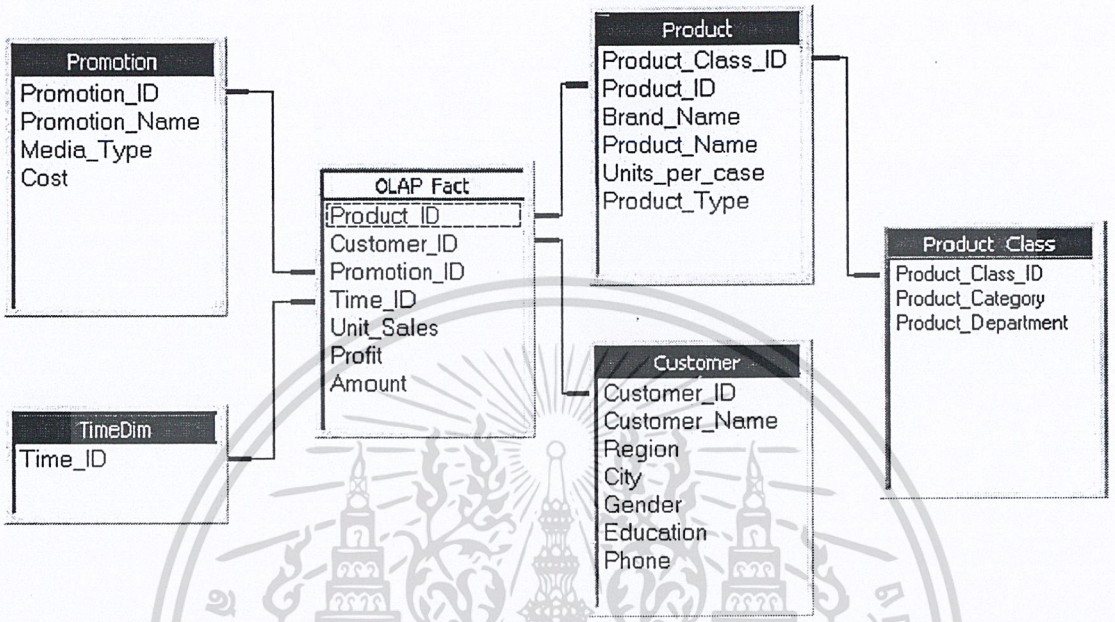
OLE DB เป็นอินเทอร์เฟส (Interface) ระดับล่างที่สามารถเข้าถึงข้อมูลได้หลากหลายรูปแบบ และไม่จำเป็นต้องเขียนคำสั่งแยกตามชนิดของข้อมูล แต่ OLE DB จะทำการแปลงข้อมูลให้อยู่ในรูปแบบมาตรฐานของตัวเอง แล้วให้แอพลิเคชันต่างๆ นำไปใช้งานได้เลยดังนั้นจึงทำให้สามารถติดต่อกับเวิร์กชีต (worksheet) ของ Excel หรือไฟล์ข้อความแบบ Text ได้

6.1.4 Hierarchical FlexGrid Control

คอนโทรล Hierarchical FlexGrid (MSHFlexGrid) มีความคล้ายคลึงกับคอนโทรล MS Data Bound Grid (DataGrid) ในวิชวลเบสิก รุ่น 5.0 มาก เพียงแต่ไม่อนุญาติให้ผู้ใช้แก้ไขหรือเปลี่ยนแปลงข้อมูลที่แสดงอยู่ โดยผู้ใช้มีสิทธิ์เข้าถึงและดูข้อมูลได้เท่านั้น การใช้ MSHFlexGrid มีข้อดี คือ ทำให้มั่นใจได้ว่า เมื่อแสดงข้อมูลให้ผู้ใช้ดูแล้ว ข้อมูลจะไม่ถูกเปลี่ยนแปลงแก้ไขด้วยวิธีใดๆ นั่นคือ ข้อมูลยังคงเหมือนต้นฉบับตลอดเวลา และที่แตกต่างกันอีกประการคือ FlexGrid สามารถกำหนดรูปแบบการแสดงผลให้เหมาะสมกับหน้าจอโปรแกรมของเราได้ตามที่ต้องการ ซึ่งจะทำให้การแสดงผลข้อมูลถูกต้องและสวยงาม

6.2 การออกแบบฐานข้อมูล (Database Design)

ฐานข้อมูลที่ได้ทำการออกแบบ จะเป็นฐานข้อมูลแบบมัลติไดเมนชันแนล (Multidimensional Database) ซึ่งหากตารางไดเมนชันมีข้อมูลที่มีโครงสร้างเป็นลำดับชั้นภายในแล้ว จะต้องทำให้เป็นโครงสร้างแบบสโนว์เฟลก (Snowflake) ดังรูปที่ 6.1



รูปที่ 6.1 แสดงการออกแบบฐานข้อมูลแบบสโนว์เฟลก

ข้อดีของการออกแบบฐานข้อมูลให้มีโครงสร้างเป็นแบบสโนว์เฟลกนี้ ก็เพื่อรองรับการใช้งานกับฐานข้อมูลขนาดใหญ่ที่มีจำนวนฐานข้อมูลหลายแสนเรคอร์ด ทำให้ง่ายต่อการเข้าใจ รวมทั้งลดความซ้ำซ้อน ซึ่งจะทำให้ลดลดปัญหาเรื่องการอัพเดทฐานข้อมูลลงได้ และอาจเพิ่มความเร็วในการ Select ข้อมูลขึ้นมา

6.3 การออกแบบโปรแกรมประยุกต์

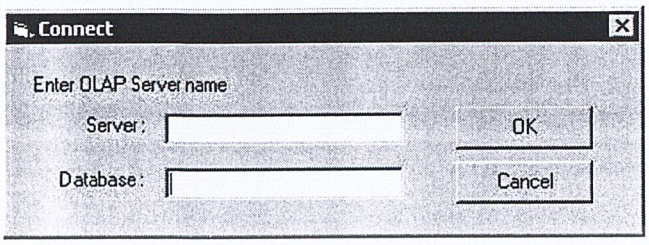
เนื่องจากหลักการของดาต้าแวร์เฮาส์เป็นหลักการที่มุ่งเน้นในเรื่องการตั้งคำถาม (Query) ฐานข้อมูลที่มีพื้นฐานของดาต้าแวร์เฮาส์ (Data Warehouse based) เพื่อให้ได้มาซึ่งคำตอบที่ต้องการจะนำไปใช้ตัดสินใจอย่างรวดเร็ว ดังนั้นโปรแกรมประยุกต์ที่เกี่ยวกับดาต้าแวร์เฮาส์นี้ จะไม่มีการอัพเดทฐานข้อมูลเหมือนในโปรแกรมประยุกต์สำหรับฐานข้อมูลทั่วไป การทำงานของโปรแกรมประยุกต์ที่ได้ออกแบบขึ้นมา จะเป็นการสร้าง MDX Statement ขึ้นมาตามคำสั่งที่ผู้ใช้ต้องการใช้ในการตั้งคำถามแต่ละครั้ง

รูปแบบของคำสั่ง MDX จะประกอบด้วย Select list, From clause, Where clause ซึ่ง Where clause นั้นจะมี 2 ส่วนคือส่วนที่ใช้เชื่อมต่อระหว่างตารางแฟ็คกับตารางไดเมนชัน (Join constrain) และส่วนที่ใช้เลือกเฉพาะแถวที่ต้องการ (Application constrain) หรืออาจเรียกว่า Search condition ก็ได้

ในโปรแกรมที่ได้ทำการออกแบบขึ้นมา จะมีฟอร์ม (Form) ที่ใช้รับคำสั่งจากผู้ใช้อยู่ด้วยกัน 2 ฟอร์ม ซึ่งจะทำการรับคำสั่งตามที่ผู้ใช้ต้องการ โดยแบ่งเป็นส่วนๆ ตามลักษณะการใช้งาน ดังนี้

- **ฟอร์มที่ 1 : ฟอร์มที่ใช้ในการติดต่อกับเซิร์ฟเวอร์และดาต้าเบส**

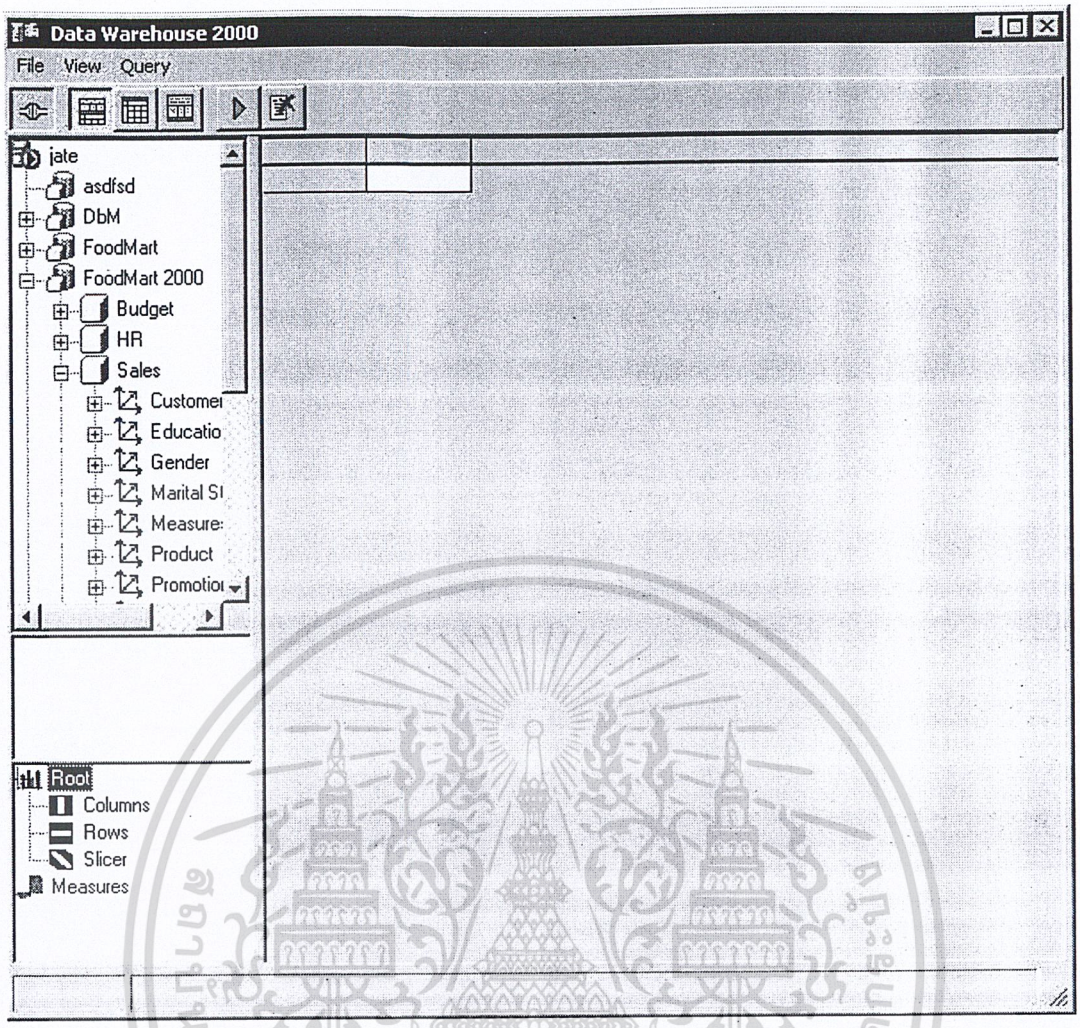
- เป็นฟอร์มที่ใช้ในการติดต่อกับเซิร์ฟเวอร์และดาต้าเบสของ OLAP โดยจะต้องทำการระบุพารามิเตอร์ (parameter) ที่เป็นชื่อเซิร์ฟเวอร์และชื่อดาต้าเบสใดดาต้าเบสหนึ่งของ OLAP ดังรูปที่ 6.2



รูปที่ 6.2 ฟอร์มสำหรับติดต่อกับเซิร์ฟเวอร์และดาต้าเบส

- **ฟอร์มที่ 2 : ฟอร์มการทำงานหลัก (Main Form)**

- เป็นฟอร์มที่ใช้ในการวิเคราะห์และแสดงข้อมูลตามที่ผู้ใช้เลือก โดยจะแบ่งออกเป็น 4 ส่วนที่สำคัญ คือ ส่วนที่แสดงข้อมูลในรูปแผนภาพต้นไม้ (Tree pane), ส่วนที่ใช้วิเคราะห์ข้อมูลในรูปของตาราง (MSHFlexGrid), ส่วนที่ใช้แสดง Multidimensional Expression (MDX) หรือ SQL Script สำหรับ Multidimensional Database และส่วนที่ควบคุมการใช้งานโปรแกรมประยุกต์ ดังรูปที่ 6.3



รูปที่ 6.3 ฟอร์มการทำงานหลักของโปรแกรมประยุกต์

6.4 คอมโพเนนต์ใน Visual Basic 6.0 ที่นำมาใช้ในการสร้างโปรแกรมประยุกต์ทางด้านฐานข้อมูล

วิชวลเบสิก 6.0 จะมีคอมโพเนนต์ต่างๆ ที่สามารถนำมาใช้ในการพัฒนาโปรแกรมประยุกต์ทางด้านฐานข้อมูล คอมโพเนนต์ที่สำคัญมีดังต่อไปนี้

6.4.1 ActiveX Data Objects Multidimensional (ADO MD)

ADO แสดงถึงโมเดลวัตถุ (Object Model) ซึ่งทำหน้าที่ติดต่อกับ OLE DB provider, วัตถุ, เมธอดและพร็อพเพอร์ตี้ ส่วน ADO MD เป็นส่วนที่เพิ่มขึ้นจาก ADO ตรงที่สามารถให้แอพลิเคชันเข้าถึงวัตถุที่ถูกกำหนดโดย OLE DB ใน OLAP เช่น PivotTable Service

ใน ADO MD จะมีวัตถุหลายอย่าง เช่น วัตถุ CubeDef ซึ่งเป็นวัตถุเมต้าดาต้าที่อธิบายถึง Cube และเป็นตัวกำหนดโครงสร้างแบบมัลติไดเมนชันแนลที่ประกอบไปด้วยลำดับชั้น (Hierarchy), ระดับ (levels) และสมาชิก (members) นอกจากนี้ยังมีวัตถุ CellSet ทำหน้าที่ในการแสดงข้อมูลที่ถูกส่งกลับมาจากคำถาม และจะเก็บวัตถุ Cell และวัตถุ Axis

6.4.2 Microsoft Hierarchical Flex Grid (MSHFlexGrid)

MSHFlexGrid เป็นคอมโพเนนต์ที่ใช้แสดงและจัดการกับ tabular data โดยจะมีความยืดหยุ่นทางการเรียง (Sort), การผสาน (Merge) และรูปแบบตารางที่ใช้เก็บสตริงและรูปภาพ ซึ่งข้อมูลที่ใช้กับ MSHFlexGrid จะเป็นข้อมูลชนิดที่อ่านเพียงอย่างเดียว (Read-only data) รายละเอียดเพิ่มเติม ดูได้ในภาคผนวก ค

6.4.3 Tree pane

เป็นคอมโพเนนต์ที่ใช้ในการแสดงรายละเอียดข้อมูลที่อยู่ในฐานข้อมูล ไม่ว่าจะเป็น ฐานข้อมูล (Database), Cube, ไดเมนชัน (Dimension)

6.4.4 Toolbar

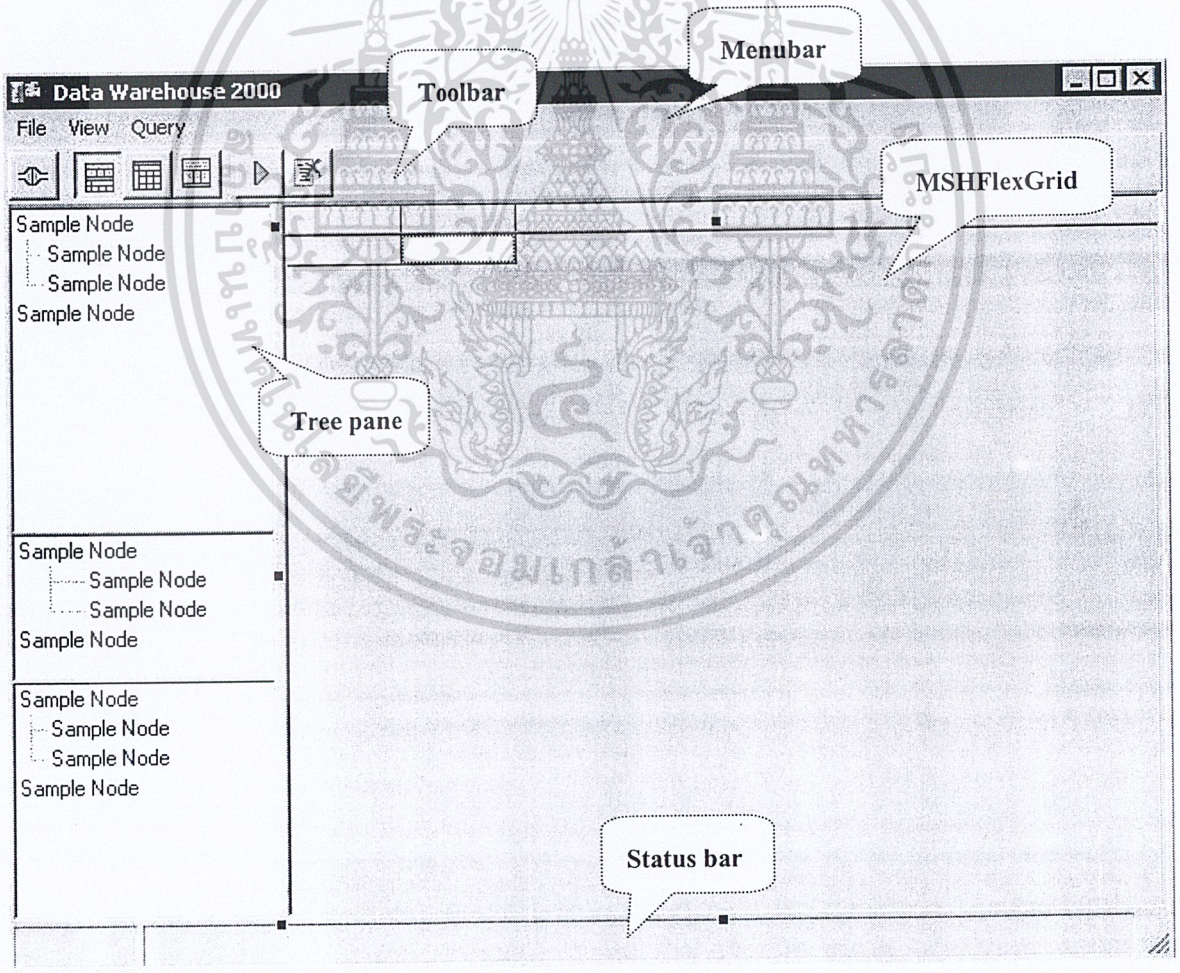
เป็นคอมโพเนนต์ที่ใช้แสดงโอเปอเรชันต่างๆ

6.4.5 Statusbar

เป็นคอมโพเนนต์ที่ใช้แสดงข้อมูลทั่วๆไป เกี่ยวกับกิจกรรมในวินโดวส์

6.4.6 Menubar

เป็นคอมโพเนนต์ที่ใช้แสดงโอเปอเรชันต่างๆ ที่ใช้ในแอพลิเคชัน



รูปที่ 6.4 แสดงคอมโพเนนต์ต่างๆ ที่ใช้ในการสร้างแอพลิเคชันทางด้านฐานข้อมูล

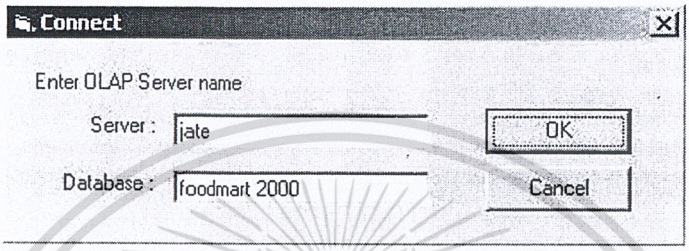
6.5 ขั้นตอนการใช้งานโปรแกรมประยุกต์

6.5.1 ความต้องการของระบบ : เครื่องที่จะติดตั้งโปรแกรมประยุกต์ควรจะมี

- Microsoft Data Aaccess Component (MDAC)
- PivotTable Service (PTS)

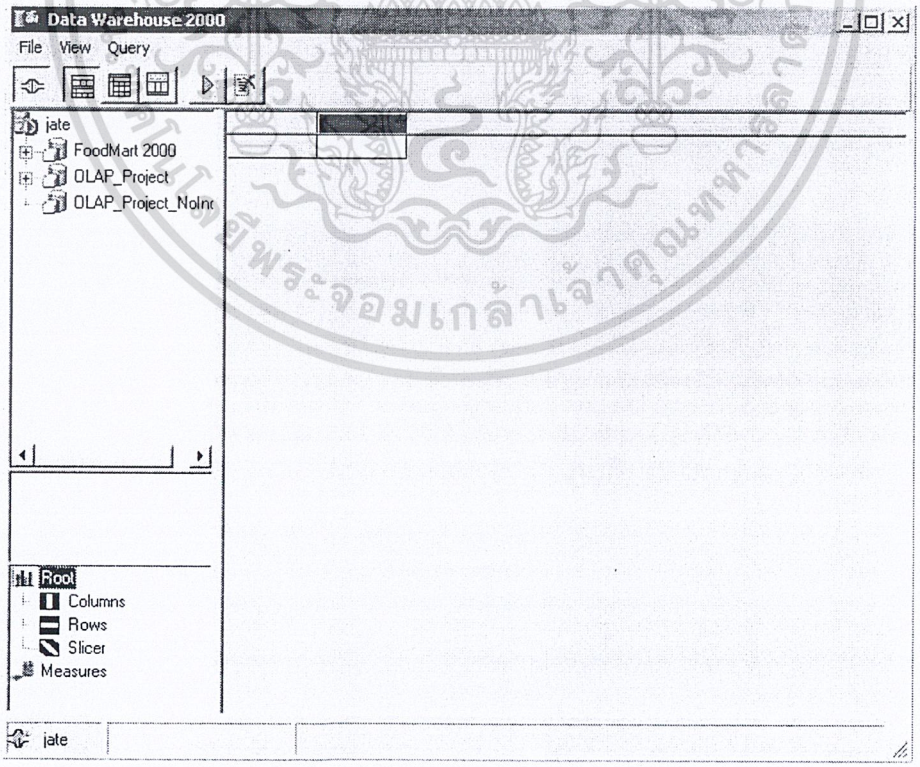
6.5.2 การใช้งานโปรแกรม

- **การติดต่อกับเซิร์ฟเวอร์และดาต้าเบสของ OLAP :** จะต้องทำการระบุพารามิเตอร์ที่เป็นชื่อเซิร์ฟเวอร์ และชื่อดาต้าเบสใดดาต้าเบสหนึ่งของ OLAP แล้วเลือกปุ่ม OK ดังรูปที่ 6.5



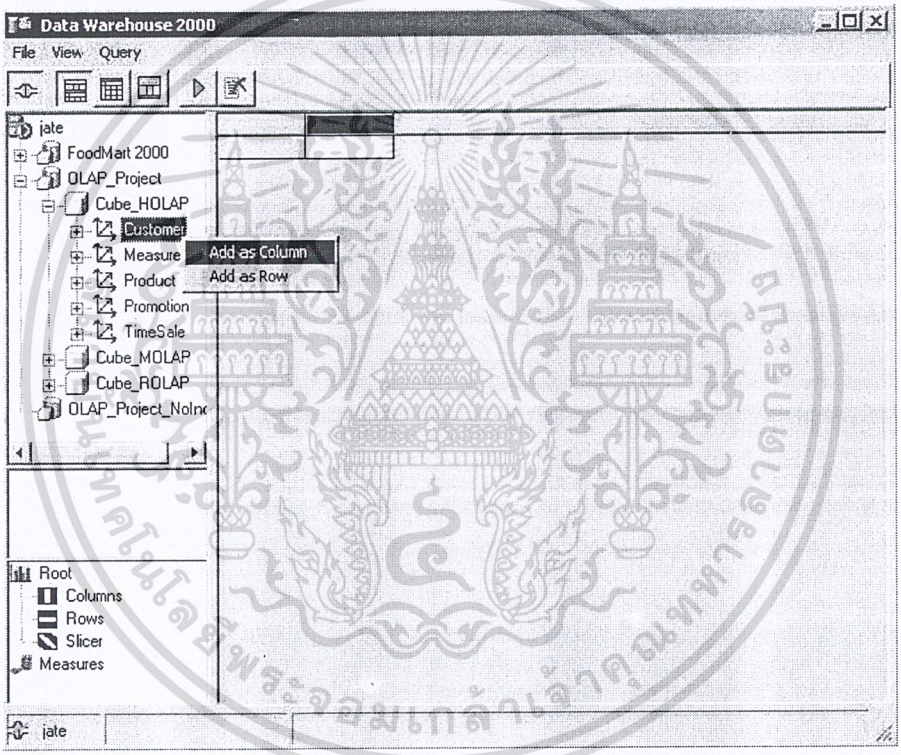
รูปที่ 6.5 แสดงการติดต่อกับเซิร์ฟเวอร์และดาต้าเบส

หลังทำการติดต่อกับเซิร์ฟเวอร์และดาต้าเบสได้แล้ว จะได้หน้าจอการทำงานดังรูปที่ 6.6

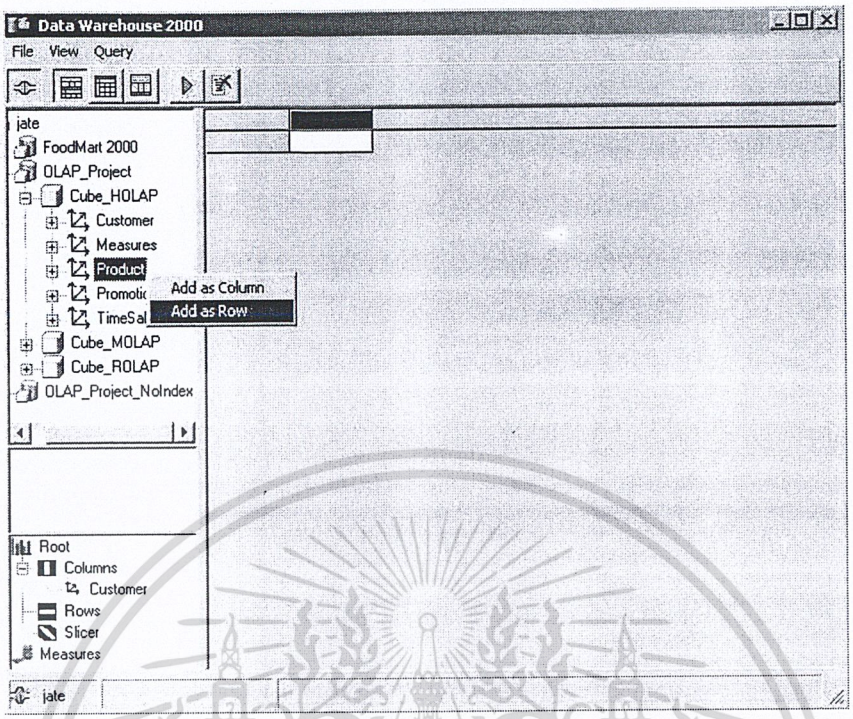


รูปที่ 6.6 แสดงหน้าจอหลักที่ใช้ในการวิเคราะห์และแสดงข้อมูล

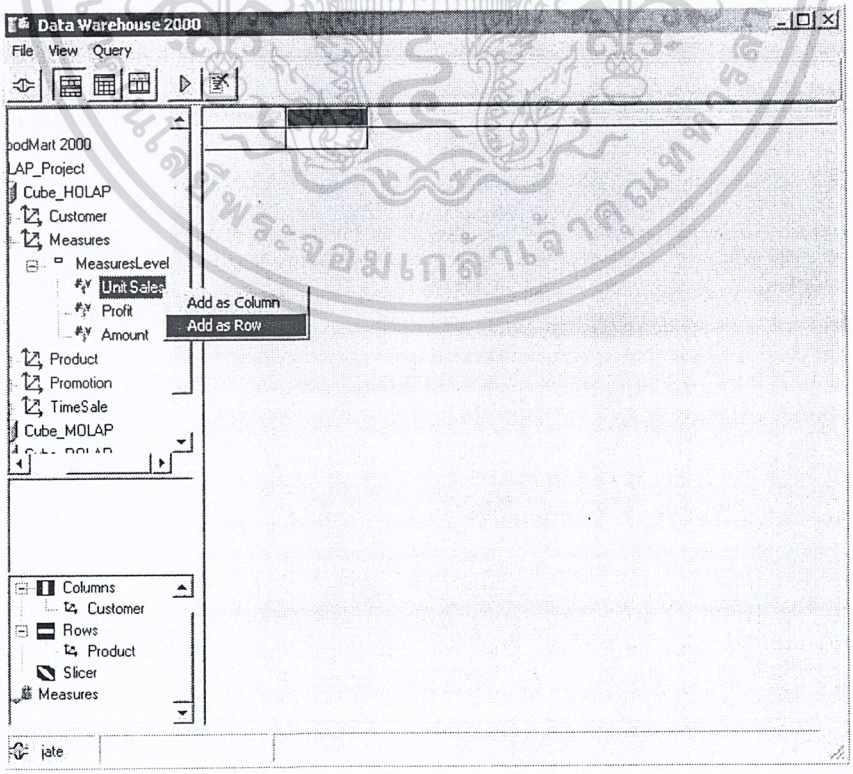
- **การเลือกไดเมนชันและตัววัดค่า :** เป็นขั้นตอนในการวิเคราะห์ข้อมูล ซึ่งสามารถเลือกไดเมนชันต่างๆ ที่เก็บอยู่ใน Cube ขึ้นมาวิเคราะห์ในแกนคอลัมน์และแถวได้ตามต้องการ โดยทำการคลิกขวาที่ชื่อไดเมนชัน จะสามารถเลือกได้ว่าจะให้นำไปวิเคราะห์ในแกนใด ได้แก่ Add as Column หรือ Add as Row เช่น เลือกไดเมนชัน Customer เป็น Column ดังรูปที่ 6.7 และเลือกไดเมนชัน Product เป็น Row ดังรูปที่ 6.8 อีกทั้งยังสามารถเลือกตัววัดค่าที่ต้องการจะนำมาใช้ในการวิเคราะห์ได้เช่นเดียวกันกับการเลือกไดเมนชัน แต่การเลือกตัววัดค่านั้น จะต้องเลือกสมาชิกที่อยู่ในระดับ Member ดังรูปที่ 6.9



รูปที่ 6.7 แสดงการ Add ไดเมนชันให้เป็นคอลัมน์

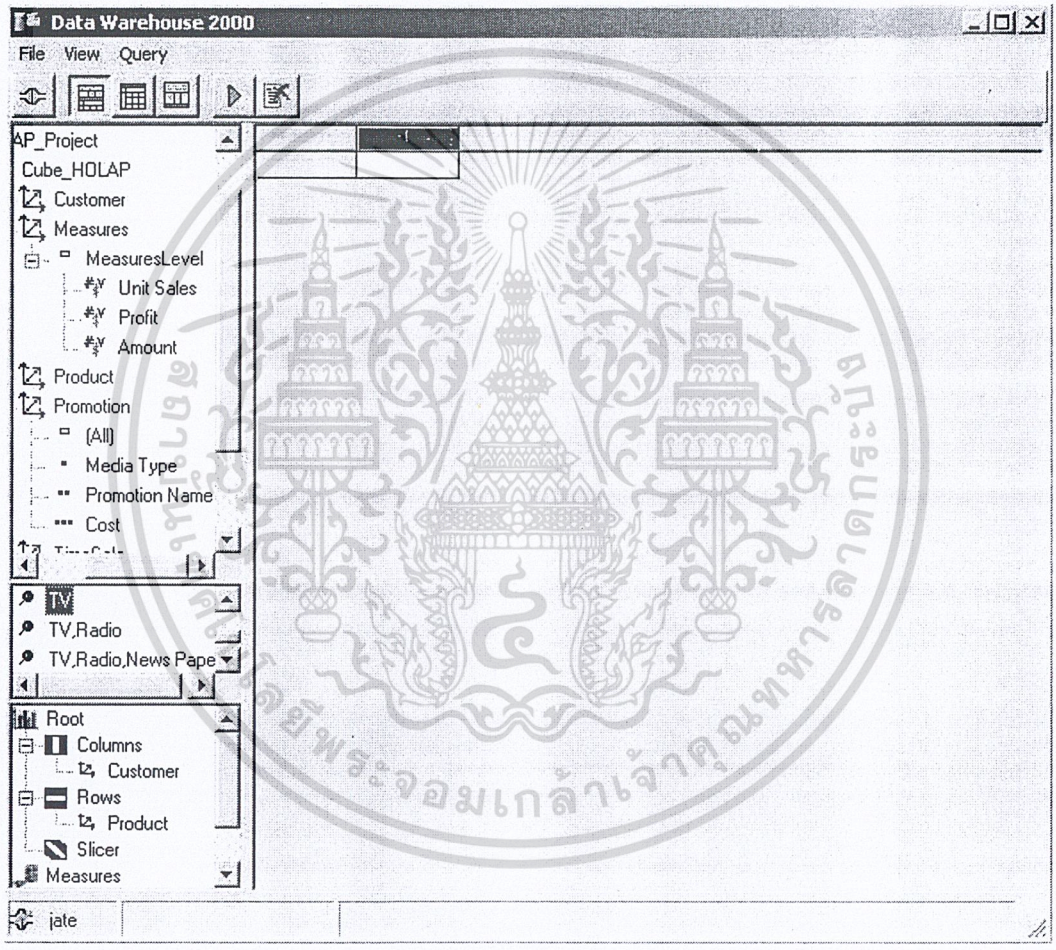


รูปที่ 6.8 แสดงการ Add ไดเมนชันให้เป็นแถว



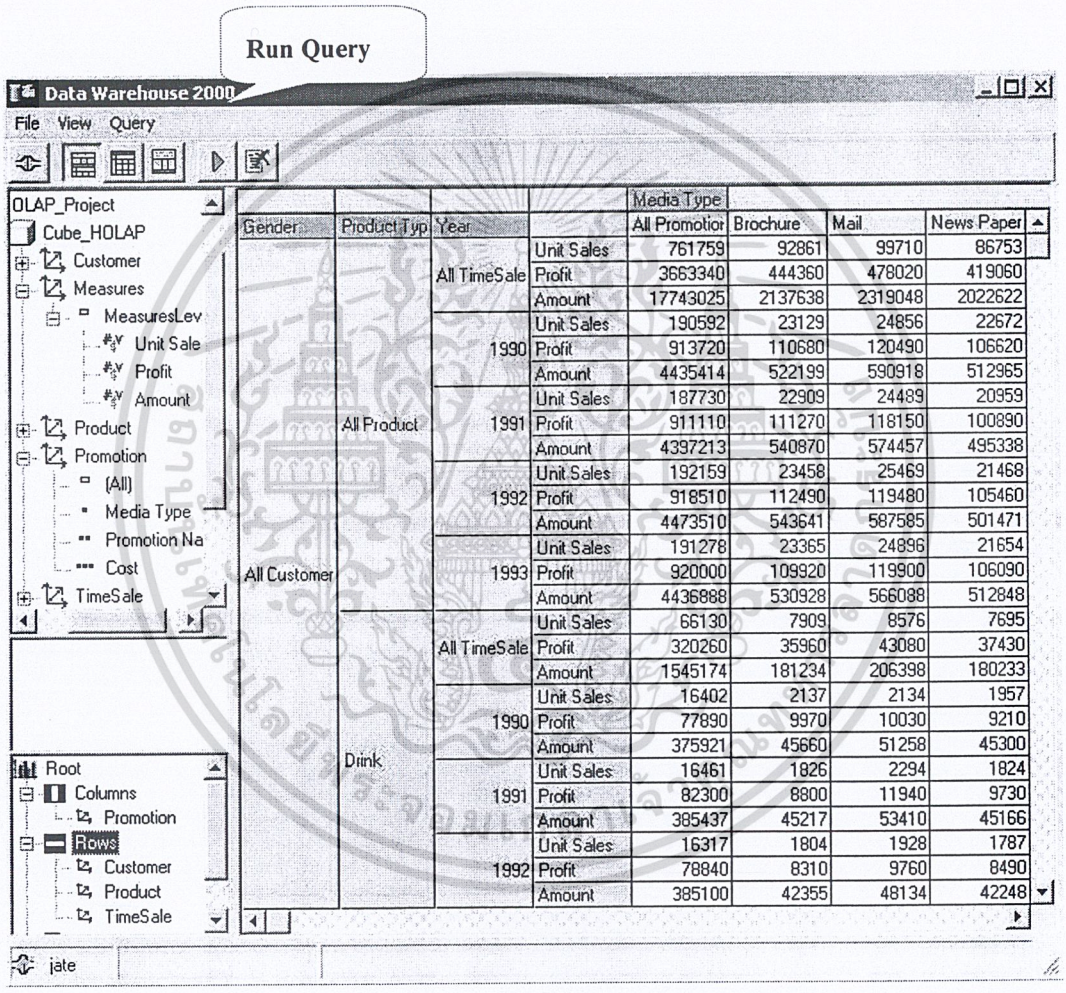
รูปที่ 6.9 แสดงการ Add ตัววัดค่าให้เป็นแถว

- **การตัดข้อมูล (Slicing Data) :** สามารถทำการตัดข้อมูลเฉพาะส่วนที่สนใจ เพื่อนำมาวิเคราะห์ได้ โดยการเลือกสมาชิกในระดับ (Level) ของไดเมนชันใดๆ ให้ดับเบิ้ลคลิกที่สมาชิกของไดเมนชันที่ต้องการที่อยู่ใน Tree pane ตรงกลาง สมาชิกตัวนั้นก็จะถูก Add เข้าไป แล้วจะแสดงใน Tree pane ด้านล่างซ้ายของหน้าจอ



รูปที่ 6.10 แสดงการตัดข้อมูลที่สนใจ เพื่อนำไปวิเคราะห์

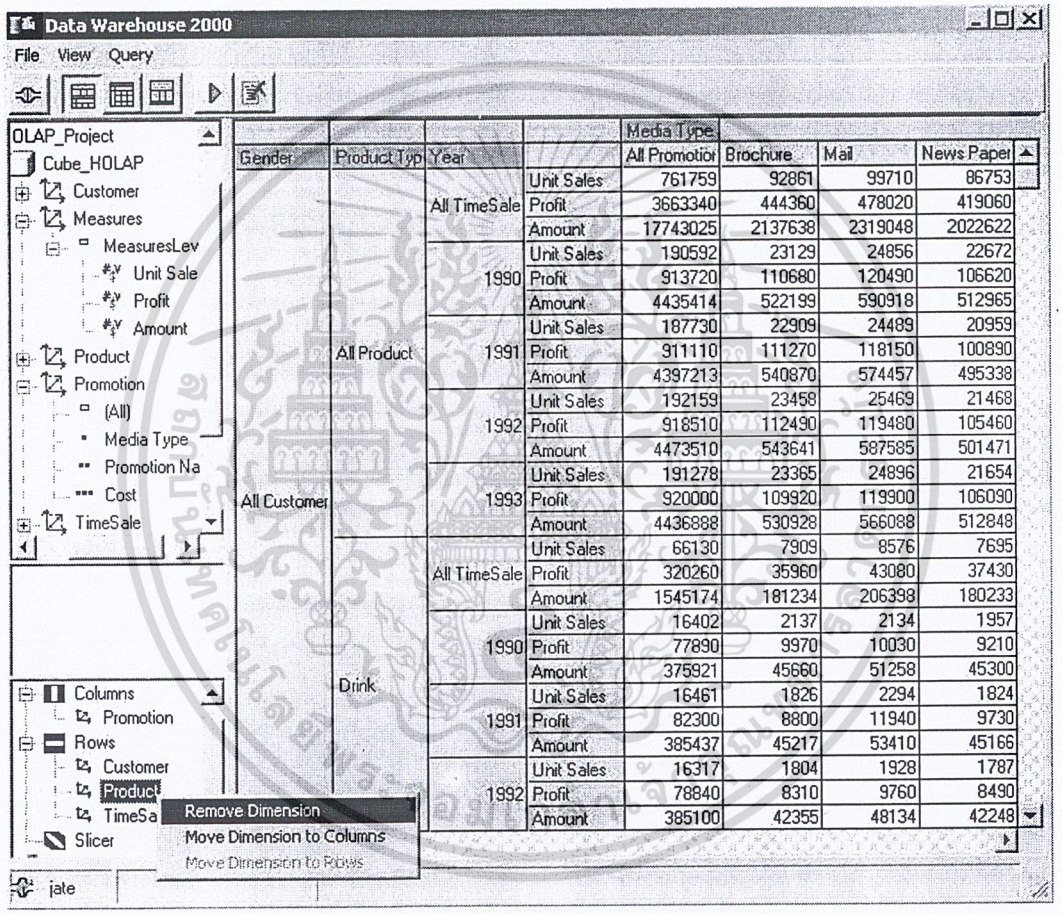
- **การรันโปรแกรม :** หลังจากทำการเลือกไดเมนชันที่จะใช้ในการวิเคราะห์เสร็จหมดแล้ว ก็จะทำการรันโปรแกรม เพื่อแสดงการวิเคราะห์ข้อมูล โดยคลิกที่ปุ่ม Run Query ที่อยู่บน Toolbar หรือเลือกคำสั่ง Run Query จาก Menubar ก็ได้เช่นเดียวกัน เมื่อทำการรันโปรแกรมแล้ว ข้อมูลที่ถูกเลือก เพื่อนำมาวิเคราะห์ ก็จะแสดงขึ้นมาบน Grid (pane ด้านขวามือ) ดังรูปที่ 6.11



Gender	Product Typ	Year		Media Type: All Promotion	Brochure	Mail	News Paper
All Customer	All Product	All TimeSale	Unit Sales	761759	92861	99710	86753
			Profit	3663340	444360	478020	419060
			Amount	17743025	2137638	2319048	2022622
		1990	Unit Sales	190592	23129	24856	22672
			Profit	913720	110680	120490	106620
			Amount	4435414	522199	590918	512965
		1991	Unit Sales	187730	22909	24489	20959
			Profit	911110	111270	118150	100890
			Amount	4397213	540870	574457	495338
		1992	Unit Sales	192159	23458	25469	21468
			Profit	918510	112490	119480	105460
			Amount	4473510	543641	587585	501471
		1993	Unit Sales	191278	23365	24896	21654
			Profit	920000	109920	119900	106090
			Amount	4436888	530928	566088	512848
	Drink	All TimeSale	Unit Sales	66130	7909	8576	7695
			Profit	320260	35960	43080	37430
			Amount	1545174	181234	206398	180233
		1990	Unit Sales	16402	2137	2134	1957
			Profit	77890	9970	10030	9210
			Amount	375921	45660	51258	45300
		1991	Unit Sales	16461	1826	2294	1824
			Profit	82300	8800	11940	9730
			Amount	385437	45217	53410	45166
		1992	Unit Sales	16317	1804	1928	1787
			Profit	78840	8310	9760	8490
			Amount	385100	42355	48134	42248

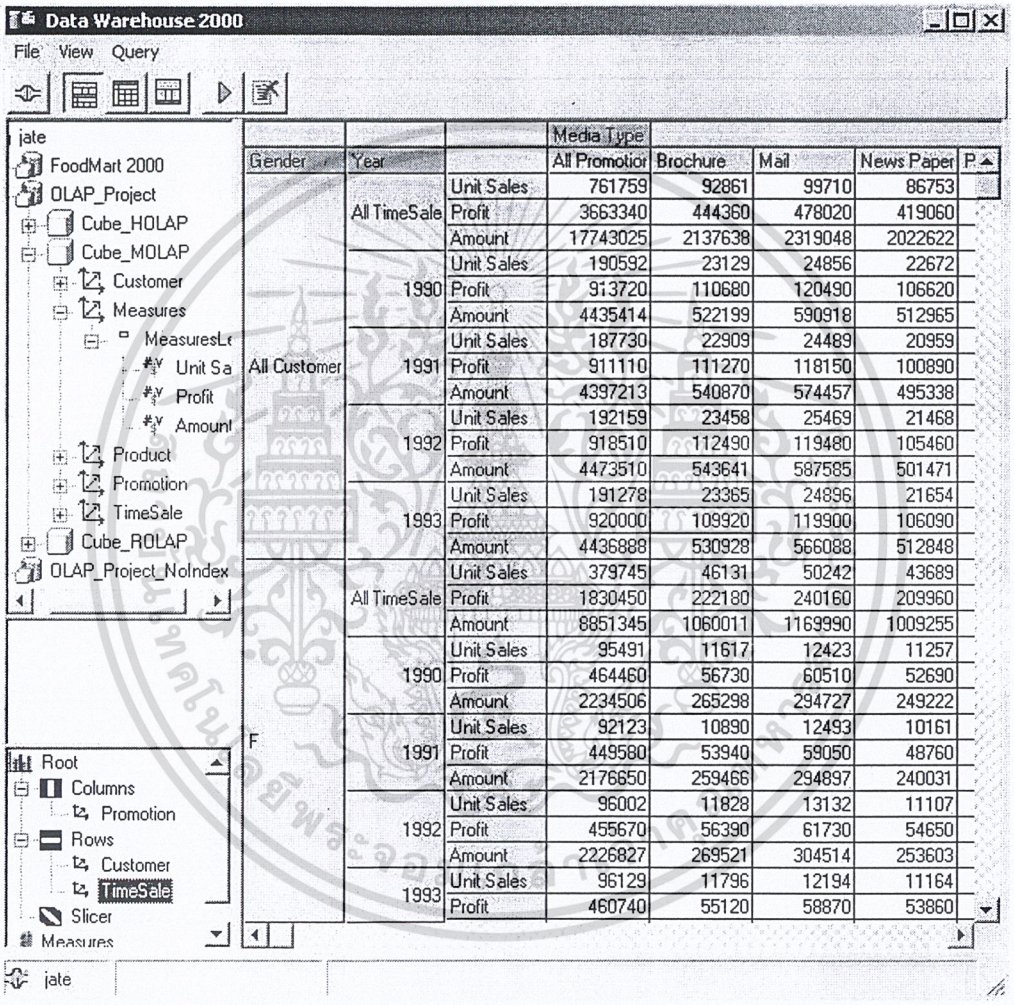
รูปที่ 6.11 แสดงการนำข้อมูลที่ต้องการขึ้นมาวิเคราะห์

- **การลบไดเมนชัน :** เมื่อทำการเลือก Add ไดเมนชันไปแล้ว ก็สามารถเลือกทิ้ง (Remove) ออกไปได้เช่นกัน โดยทำการคลิกขวาที่ไดเมนชันที่ต้องการจะเลือกทิ้ง แล้วเลือกคำสั่ง Remove Dimension ดังรูปที่ 6.12



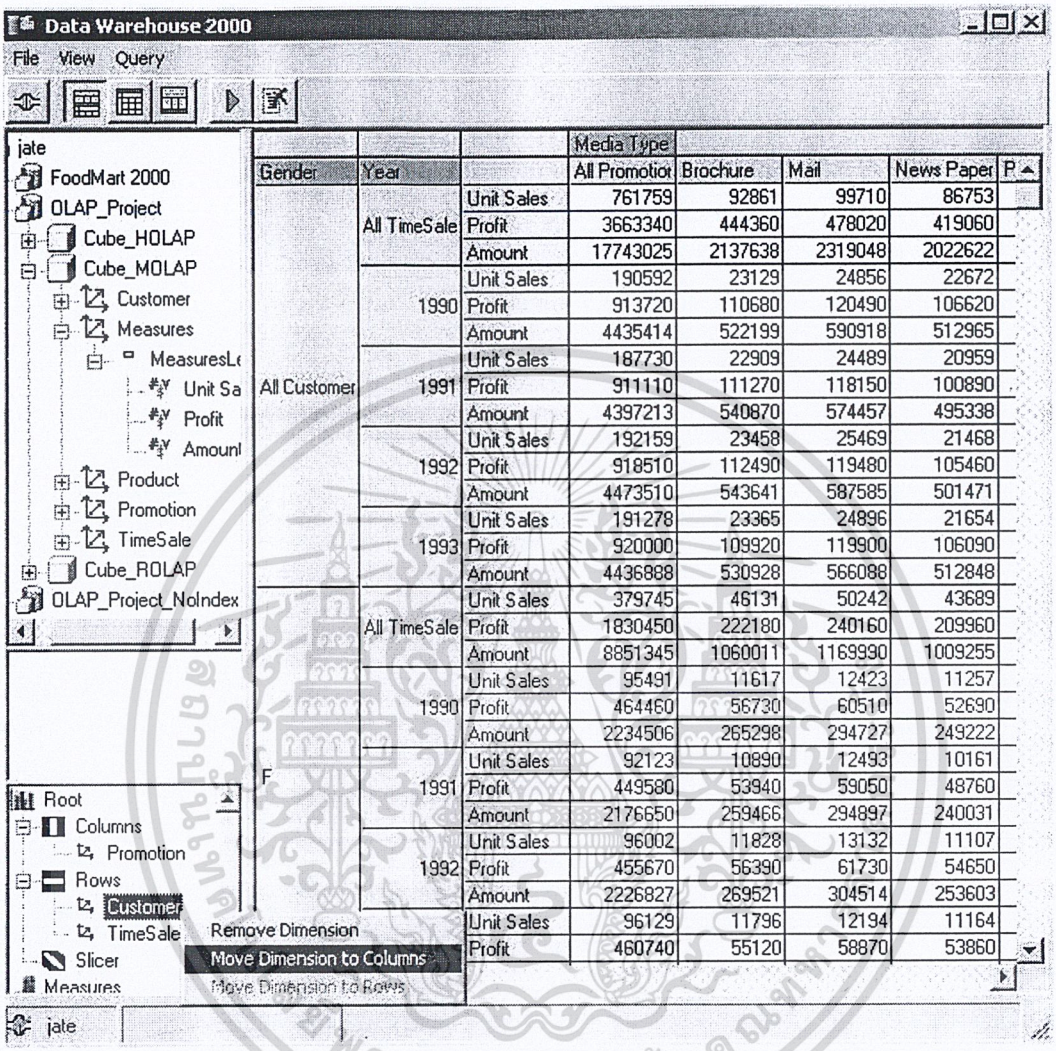
รูปที่ 6.12 แสดงการลบไดเมนชันที่ไม่ต้องการวิเคราะห์

เมื่อทำการลบไดเมนชันแล้ว จะต้องคลิกปุ่ม Run Query อีกครั้ง เพราะจะต้องทำการส่ง MDX Statement อันใหม่ไปแทน และข้อมูลอันใหม่ก็จะแสดงขึ้นมาอีกครั้ง ดังรูปที่ 6.13



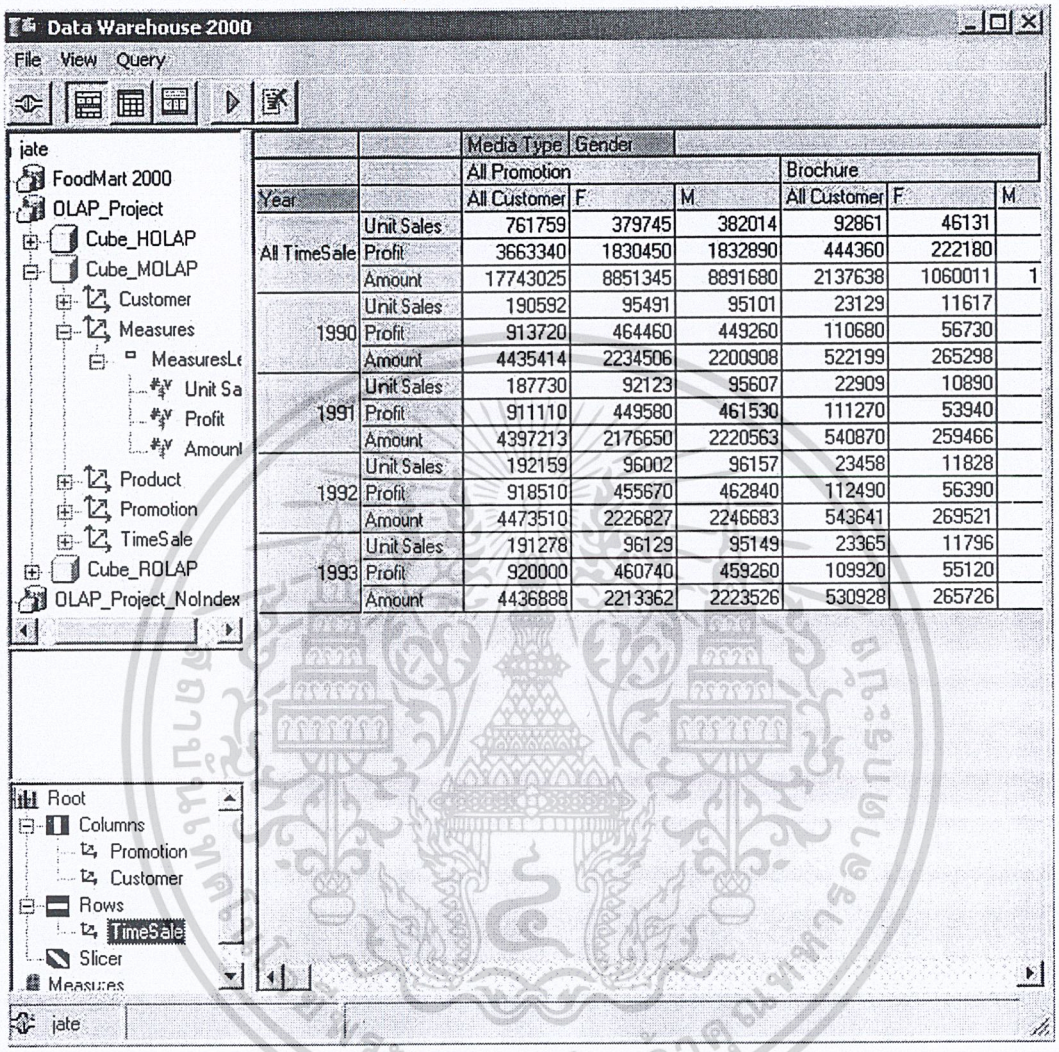
รูปที่ 6.13 แสดงข้อมูลที่ลบไดเมนชันที่ไม่ต้องการออกไปแล้ว

- **การหมุนข้อมูล (Dicing Data) :** สามารถทำการย้ายข้อมูลที่เลือกมาวิเคราะห์ จากแกนคอลัมน์ไปเป็นแกนแถว หรือจากแกนแถวไปเป็นแกนคอลัมน์ได้ โดยทำการคลิกขวาที่ไดเมนชันที่ต้องการหมุน แล้วเลือกคำสั่ง Move Dimension to Columns หรือ Move Dimension to Rows ก็ได้ ดังรูปที่ 6.14



รูปที่ 6.14 แสดงวิธีการหมุนข้อมูล

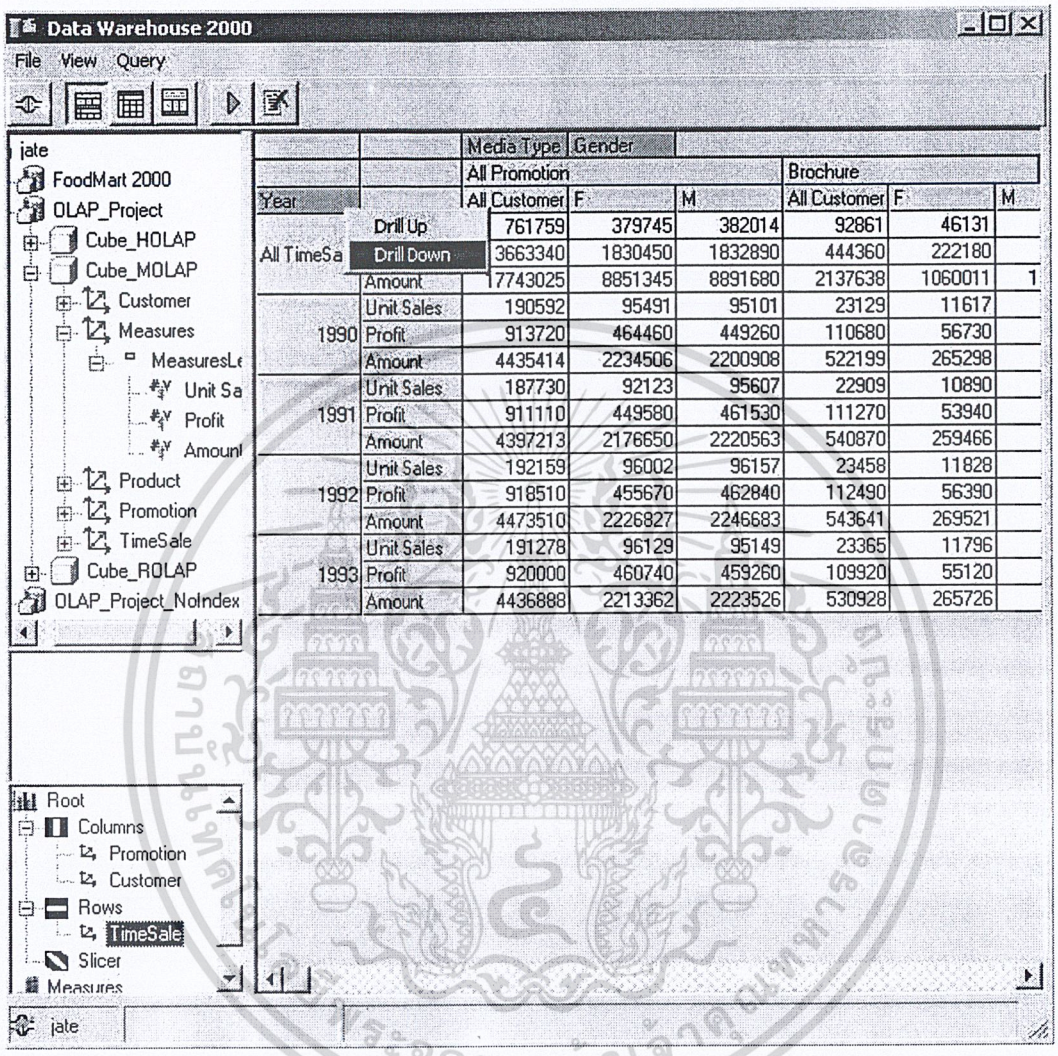
ตามรูปที่ 6.14 จะเป็นการหมุนไดเมนชัน Customer จากแกนแถวไปเป็นแกนคอลัมน์ โดยใช้คำสั่ง Move Dimension to Columns จากนั้นจะต้องคลิกปุ่ม Run Query อีกครั้ง จึงจะเห็นการเปลี่ยนแปลง ดังรูปที่ 6.15



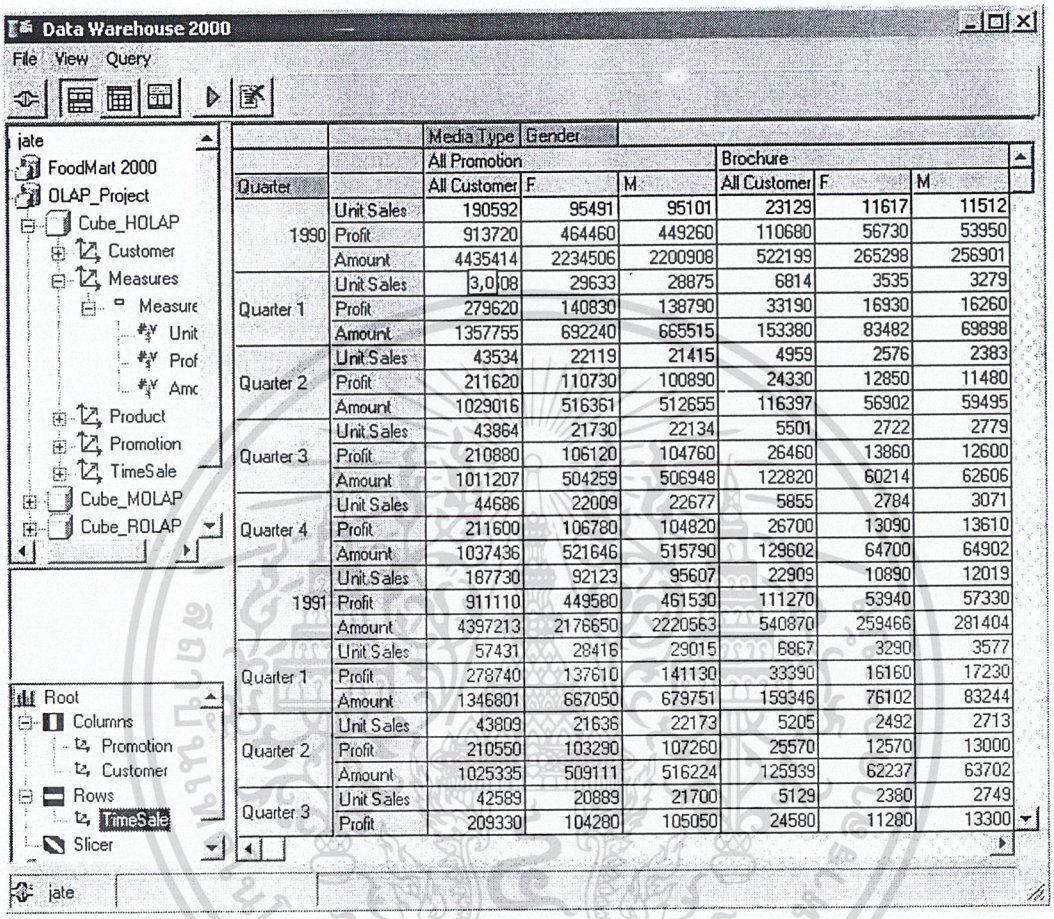
		Media Type	Gender				
		All Promotion			Brochure		
Year		All Customer	F	M	All Customer	F	M
All TimeSale	Unit Sales	761759	379745	382014	92861	46131	
	Profit	3663340	1830450	1832890	444360	222180	
	Amount	17743025	8851345	8891680	2137638	1060011	1
1990	Unit Sales	190592	95491	95101	23129	11617	
	Profit	913720	464460	449260	110680	56730	
	Amount	4435414	2234506	2200908	522199	265298	
1991	Unit Sales	187730	92123	95607	22909	10890	
	Profit	911110	449580	461530	111270	53940	
	Amount	4397213	2176650	2220563	540870	259466	
1992	Unit Sales	192159	96002	96157	23458	11828	
	Profit	918510	455670	462840	112490	56390	
	Amount	4473510	2226827	2246683	543641	269521	
1993	Unit Sales	191278	96129	95149	23365	11796	
	Profit	920000	460740	459260	109920	55120	
	Amount	4436888	2213362	2223526	530928	265726	

รูปที่ 6.15 แสดงการหมุนไดเมนชันจาก Column ไปเป็น Row

- **การ Drill Up และ Drill Down :** สามารถทำการ Drill Up และ Drill Down ได้ทุกไดเมนชันที่มีจำนวนระดับ (Level) มากกว่า 1 ระดับ โดยการคลิกขวาที่เซลล์ที่เป็นชื่อของระดับในไดเมนชัน (จะเป็นเซลล์ที่มีสีฟ้า) แล้วทำการเลือกคำสั่ง Drill Up หรือ Drill Down ได้ตามต้องการ เช่น ถ้าทำการ Drill Down ระดับ " Year " ของไดเมนชัน TimeSale ดังรูปที่ 6.16 ก็จะได้ผลลัพธ์ดังรูปที่ 6.17 ซึ่งแสดงข้อมูลในระดับ " Quarter " นั่นเอง



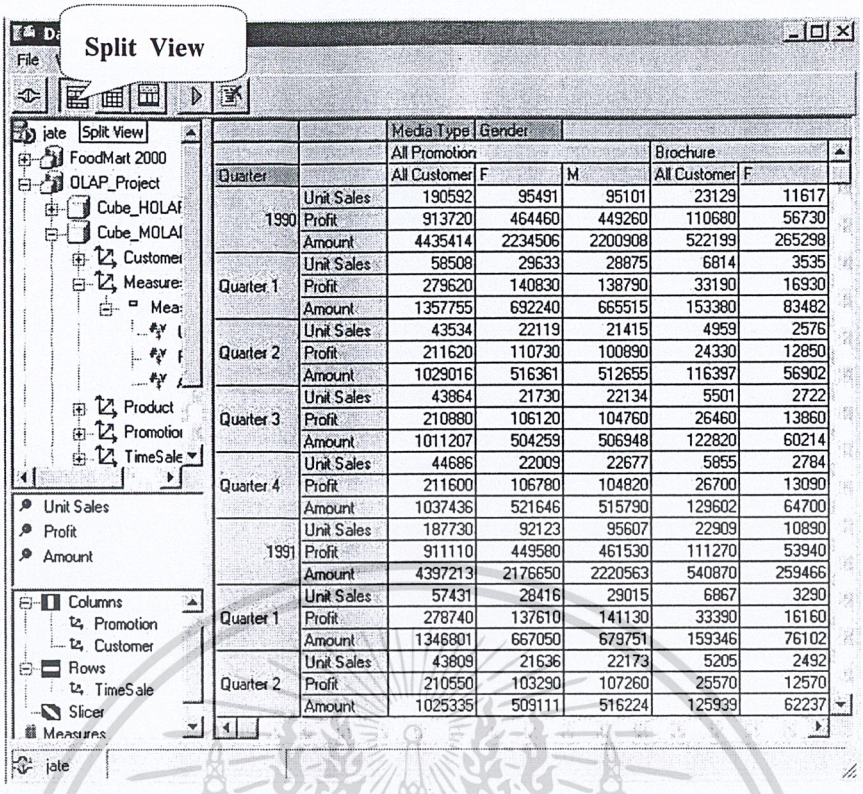
รูปที่ 6.16 แสดงการเลือกระดับในการ Drill Down



		Media Type	Gender				
		All Promotion			Brochure		
Quarter		All Customer	F	M	All Customer	F	M
1990	Unit Sales	190592	95491	95101	23129	11617	11512
	Profit	913720	464460	449260	110680	56730	53950
	Amount	4435414	2234506	2200908	522199	265298	256901
Quarter 1	Unit Sales	3,008	29633	28875	6814	3535	3279
	Profit	279620	140830	138790	33190	16930	16260
	Amount	1357755	692240	665515	153380	83482	69898
Quarter 2	Unit Sales	43534	22119	21415	4959	2576	2383
	Profit	211620	110730	100890	24330	12850	11480
	Amount	1029016	516361	512655	116397	56902	59495
Quarter 3	Unit Sales	43864	21730	22134	5501	2722	2779
	Profit	210880	106120	104760	26460	13860	12600
	Amount	1011207	504259	506948	122820	60214	62606
Quarter 4	Unit Sales	44686	22009	22677	5855	2784	3071
	Profit	211600	106780	104820	26700	13090	13610
	Amount	1037436	521646	515790	129602	64700	64902
1991	Unit Sales	187730	92123	95607	22909	10890	12019
	Profit	911110	449580	461530	111270	53940	57330
	Amount	4397213	2176650	2220563	540870	259466	281404
Quarter 1	Unit Sales	57431	28416	29015	6867	3290	3577
	Profit	278740	137610	141130	33390	16160	17230
	Amount	1346801	667050	679751	159346	76102	83244
Quarter 2	Unit Sales	43809	21636	22173	5205	2492	2713
	Profit	210550	103290	107260	25570	12570	13000
	Amount	1025335	509111	516224	125939	62237	63702
Quarter 3	Unit Sales	42589	20889	21700	5129	2380	2749
	Profit	209330	104280	105050	24580	11280	13300

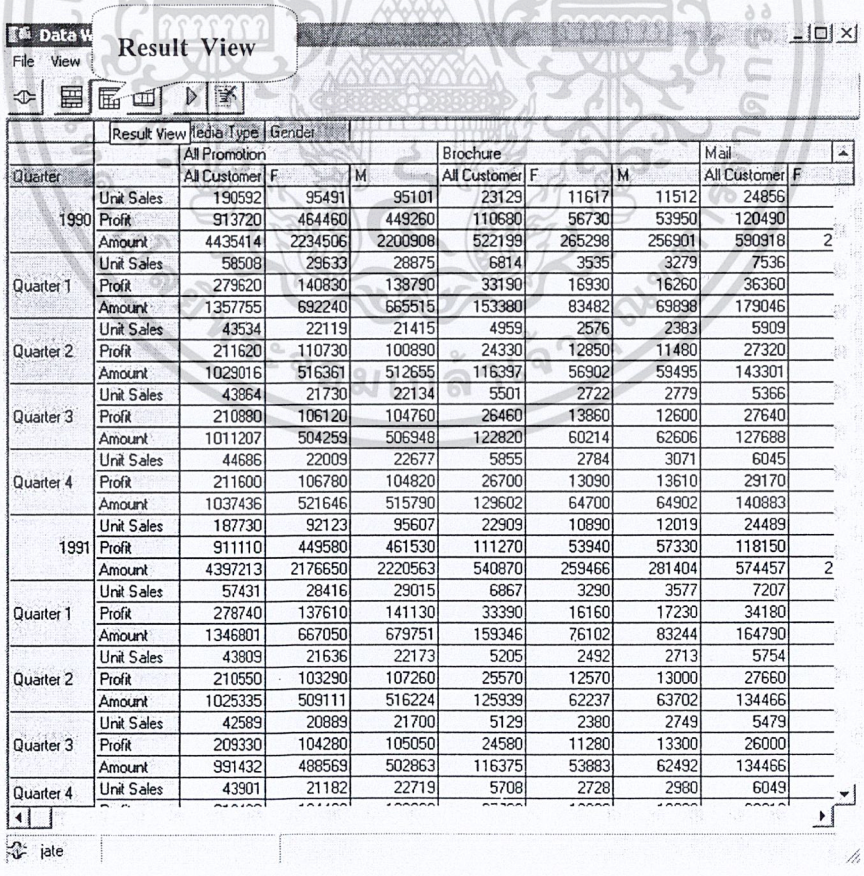
รูปที่ 6.17 แสดงการ Drill Down ข้อมูล

- **รูปแบบเพิ่มเติมในการแสดงข้อมูลของโปรแกรมประยุกต์**
  - **Split View :** เป็นปุ่มที่เลือกแสดงข้อมูลในส่วนที่เป็น Tree pane และ Grid ดังรูปที่ 6.18
  - **Result View :** เป็นปุ่มที่เลือกแสดงข้อมูลเฉพาะส่วนที่เป็น Grid เท่านั้น ดังรูปที่ 6.19
  - **Query View :** เป็นปุ่มที่เลือกแสดงข้อมูลที่เพิ่มการแสดง MDX Statement ขึ้นมา ดังรูปที่ 6.20
  - **Clear Query :** เป็นปุ่มที่ใช้ในการเคลียร์ข้อมูลและโมเดลที่ได้ทำการเลือกมาวิเคราะห์ทั้งหมด ดังรูปที่ 6.21
  - **Disconnect :** เป็นปุ่มที่ใช้สำหรับการยกเลิกการติดต่อกับเซิร์ฟเวอร์และดาต้าเบส ดังรูปที่ 6.22



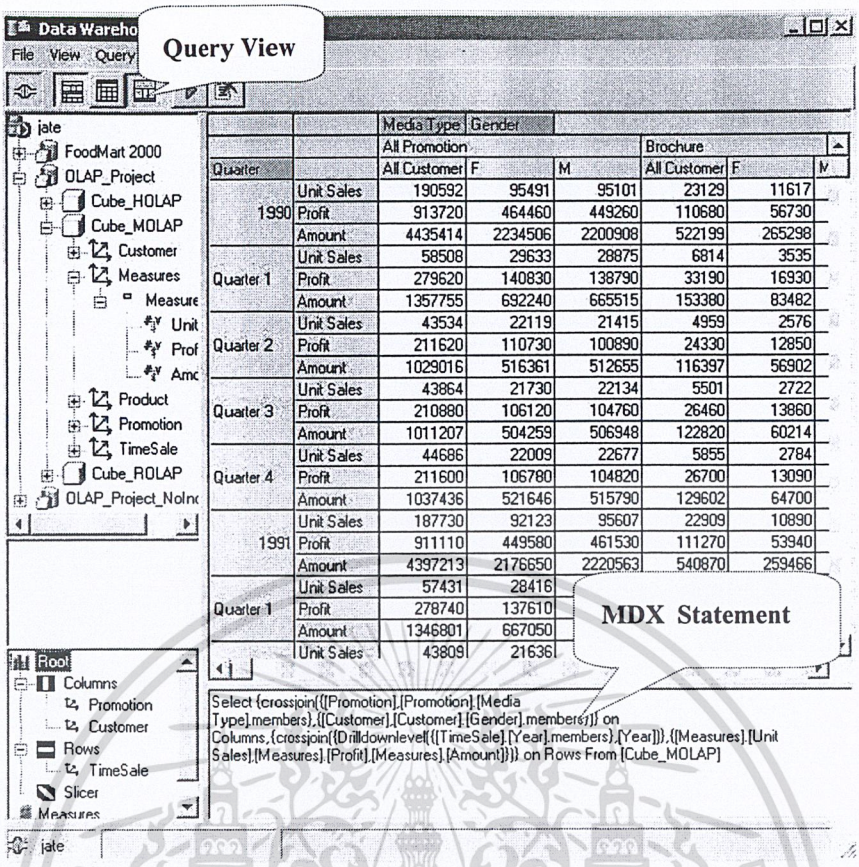
		Media Type	Gender			
		All Promotion			Brochure	
Quarter		All Customer	F	M	All Customer	F
1990	Unit Sales	190592	95491	95101	23129	11617
	Profit	913720	464460	449260	110680	56730
	Amount	4435414	2234506	2200908	522199	265298
Quarter 1	Unit Sales	58508	29633	28875	6814	3535
	Profit	279620	140830	138790	33190	16930
	Amount	1357755	692240	665515	153380	83482
Quarter 2	Unit Sales	43534	22119	21415	4959	2576
	Profit	211620	110730	100890	24330	12850
	Amount	1029016	516361	512655	116397	56902
Quarter 3	Unit Sales	43864	21730	22134	5501	2722
	Profit	210880	106120	104760	26460	13860
	Amount	1011207	504259	506948	122820	60214
Quarter 4	Unit Sales	44686	22009	22677	5855	2784
	Profit	211600	106780	104820	26700	13090
	Amount	1037436	521646	515790	129602	64700
1991	Unit Sales	187730	92123	95607	22909	10890
	Profit	911110	449580	461530	111270	53940
	Amount	4397213	2176650	2220563	540870	259466
Quarter 1	Unit Sales	57431	28416	29015	6867	3290
	Profit	278740	137610	141130	33390	16160
	Amount	1346801	667050	679751	159346	76102
Quarter 2	Unit Sales	43809	21636	22173	5205	2492
	Profit	210550	103290	107260	25570	12570
	Amount	1025335	509111	516224	125939	62237

รูปที่ 6.18 แสดงการเลือกดูข้อมูลแบบ Split View

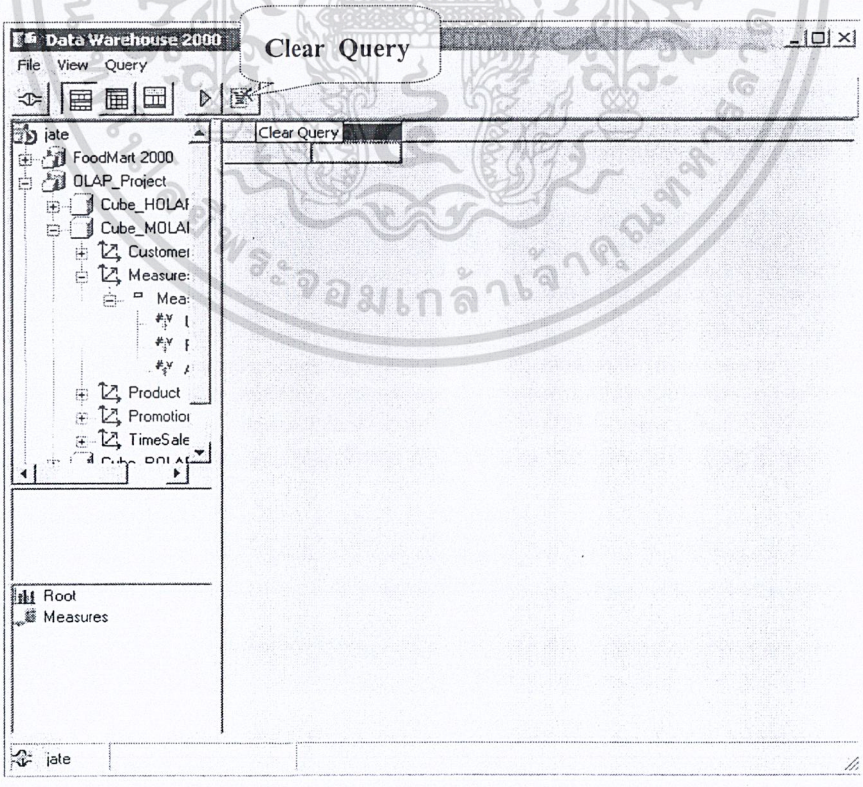


		All Promotion			Brochure			Mail	
Quarter		All Customer	F	M	All Customer	F	M	All Customer	F
1990	Unit Sales	190592	95491	95101	23129	11617	11512	24856	
	Profit	913720	464460	449260	110680	56730	53950	120490	
	Amount	4435414	2234506	2200908	522199	265298	256901	590918	2
Quarter 1	Unit Sales	58508	29633	28875	6814	3535	3279	7536	
	Profit	279620	140830	138790	33190	16930	16260	36360	
	Amount	1357755	692240	665515	153380	83482	69898	179046	
Quarter 2	Unit Sales	43534	22119	21415	4959	2576	2383	5909	
	Profit	211620	110730	100890	24330	12850	11480	27320	
	Amount	1029016	516361	512655	116397	56902	59495	143301	
Quarter 3	Unit Sales	43864	21730	22134	5501	2722	2779	5366	
	Profit	210880	106120	104760	26460	13860	12600	27640	
	Amount	1011207	504259	506948	122820	60214	62606	127688	
Quarter 4	Unit Sales	44686	22009	22677	5855	2784	3071	6045	
	Profit	211600	106780	104820	26700	13090	13610	29170	
	Amount	1037436	521646	515790	129602	64700	64902	140883	
1991	Unit Sales	187730	92123	95607	22909	10890	12019	24489	
	Profit	911110	449580	461530	111270	53940	57330	118150	
	Amount	4397213	2176650	2220563	540870	259466	281404	574457	2
Quarter 1	Unit Sales	57431	28416	29015	6867	3290	3577	7207	
	Profit	278740	137610	141130	33390	16160	17230	34180	
	Amount	1346801	667050	679751	159346	76102	83244	164790	
Quarter 2	Unit Sales	43809	21636	22173	5205	2492	2713	5754	
	Profit	210550	103290	107260	25570	12570	13000	27660	
	Amount	1025335	509111	516224	125939	62237	63702	134466	
Quarter 3	Unit Sales	42589	20889	21700	5129	2380	2749	5479	
	Profit	209330	104280	105050	24580	11280	13300	26000	
	Amount	991432	488569	502863	116375	53883	62492	134466	
Quarter 4	Unit Sales	43901	21182	22719	5708	2728	2980	6049	

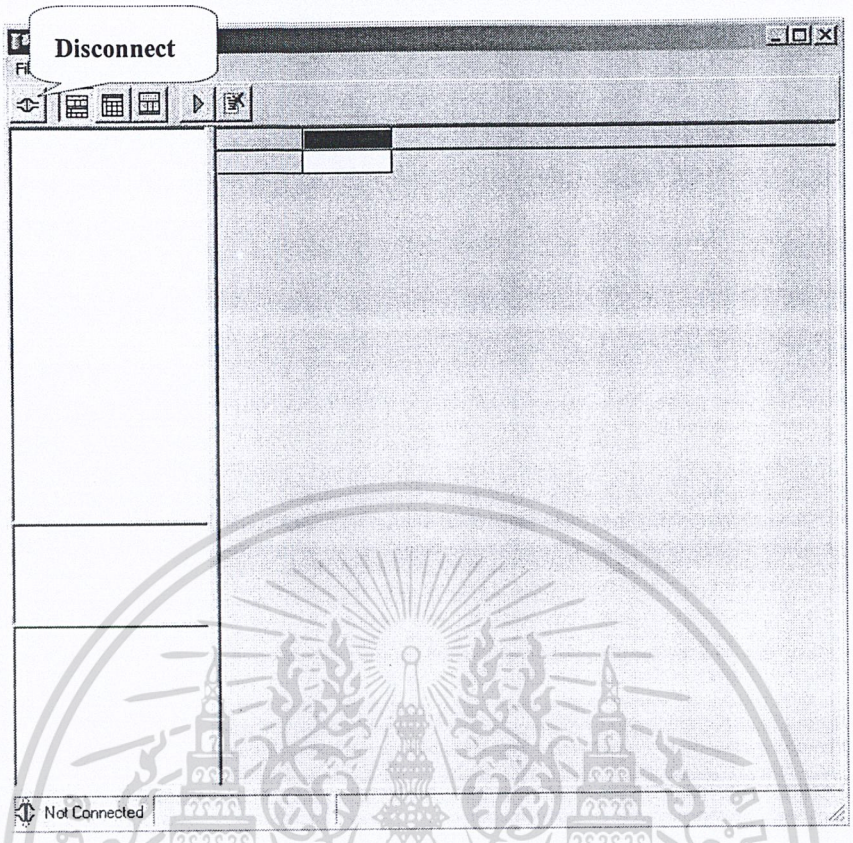
รูปที่ 6.19 แสดงการเลือกดูข้อมูลแบบ Result View



รูปที่ 6.20 แสดงการเลือกดูข้อมูลแบบ Query View



รูปที่ 6.21 แสดงการเคลียร์ข้อมูลและโมเดล

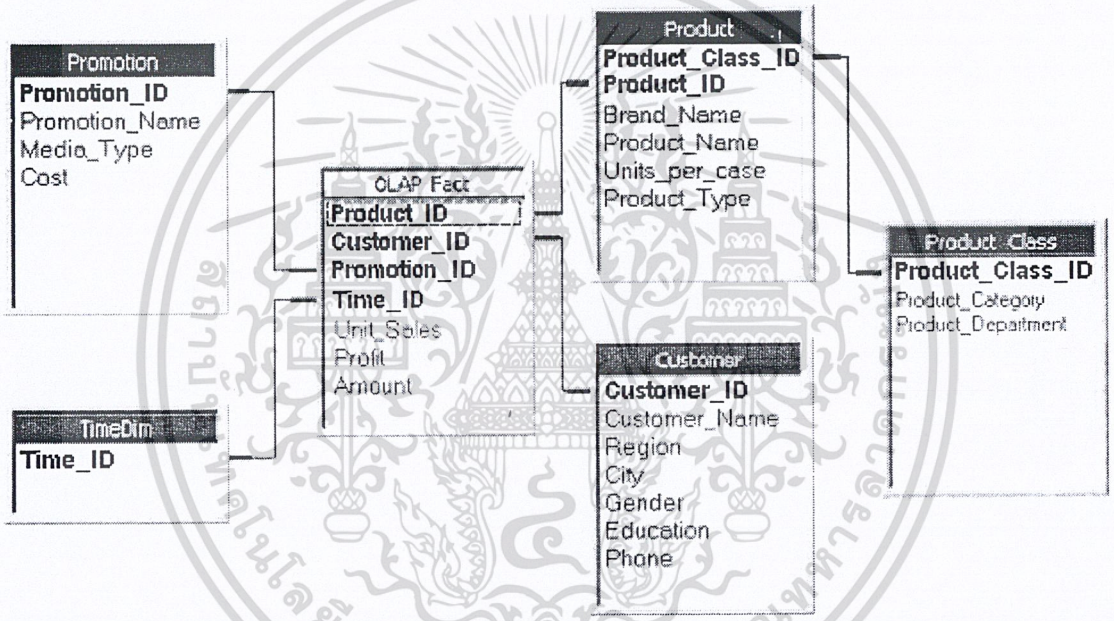


รูปที่ 6.22 แสดงการ Disconnect กับเซิร์ฟเวอร์และดาต้าเบส

บทที่ 7
การทดสอบประสิทธิภาพของโปรแกรมประยุกต์

7.1 การใช้งานอินเด็กซ์ (Index) ในฐานข้อมูลดาต้าแวร์เฮาส์

วิธีการสร้างอินเด็กซ์บนคอลัมน์นี้จะเหมาะสมกับตารางที่มีเรคอร์ดจำนวนมากๆ ถ้าสร้างอินเด็กซ์บนตารางที่มีข้อมูลน้อยๆ จะทำให้เกิด Overhead แทนที่จะทำให้ใช้เวลาน้อยลง จะกลับทำให้ใช้เวลาในการ Query เพิ่มมากขึ้น และจะต้องพิจารณาด้วยว่า ควรจะสร้างอินเด็กซ์บนคอลัมน์ไหน เพื่อให้การทำงานเกิดประสิทธิภาพสูงสุด สำหรับฐานข้อมูลที่ทำการสร้างขึ้นจะทำการสร้างอินเด็กซ์บน Primary Key ของทุกๆ ตารางดังรูปที่ 7.1 ซึ่งตัวหนาจะเป็นคอลัมน์ที่มีการสร้างอินเด็กซ์



รูปที่ 7.1 แสดงโครงสร้างสำหรับตารางที่มีการสร้างอินเด็กซ์

7.2 การทดลองวัดประสิทธิภาพของการ Query เทียบกับเวลาในการจัดเก็บลักษณะต่างๆ

7.2.1 ฐานข้อมูลดาต้าแวร์เฮาส์ที่มีการใช้อินเด็กซ์

- **การจัดเก็บแบบ Multidimensional OLAP (MOLAP)**

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:11
4,200	12	50,400	0:20	0:53

37,860	12	454,320	2:20	5:50
211,620	12	2,539,440	12:13	35:11

ตารางที่ 7.1 เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ MOLAP

- **การจัดเก็บแบบ Relational OLAP (ROLAP)**

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:13
4,200	12	50,400	0:20	0:55
37,860	12	454,320	2:20	6:00
211,620	12	2,539,440	15:17	44:26

ตารางที่ 7.2 เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ ROLAP

- **การจัดเก็บแบบ Hybrid OLAP (HOLAP)**

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:11
4,200	12	50,400	0:20	0:55
37,860	12	454,320	2:20	5:55
211,620	12	2,539,440	14:03	41:44

ตารางที่ 7.3 เปรียบเทียบเวลาของการ Query ข้อมูลที่ใช้อินเด็กซ์และเก็บแบบ HOLAP

7.2.2 ฐานข้อมูลดาต้าแวร์เฮาส์ที่ไม่มีการใช้อินเด็กซ์

- การจัดเก็บแบบ Multidimensional OLAP (MOLAP)

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:11
4,200	12	50,400	0:20	0:53
37,860	12	454,320	2:23	6:02
211,620	12	2,539,440	12:52	35:46

ตารางที่ 7.4 เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ MOLAP

- การจัดเก็บแบบ Relational OLAP (ROLAP)

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:13
4,200	12	50,400	0:20	0:55
37,860	12	454,320	2:36	6:18
211,620	12	2,539,440	16:10	46:55

ตารางที่ 7.5 เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ ROLAP

- การจัดเก็บแบบ Hybrid OLAP (HOLAP)

จำนวนแถว (Rows)	จำนวนคอลัมน์ (Columns)	จำนวนเซลล์ (Cells)	เวลา (นาที)	
			Server	Client
180	12	2,160	0:01	0:03
720	12	8,640	0:03	0:11
4,200	12	50,400	0:20	0:55
37,860	12	454,320	2:36	6:10
211,620	12	2,539,440	15:28	42:15

ตารางที่ 7.6 เปรียบเทียบเวลาของการ Query ข้อมูลที่ไม่มีการใช้อินเด็กซ์และเก็บแบบ HOLAP

คุณสมบัติของเครื่องที่ใช้ในการทดสอบ

เครื่องเซิร์ฟเวอร์ : Pentium III 667 MHz
RAM 128 MB

เครื่องไคลเอนท์ : Pentium II 266 MHz
RAM 64 MB

7.3 สรุปผลการทดลอง

7.3.1 สำหรับฐานข้อมูลที่มีการใช้อินเด็กซ์ จะใช้เวลาน้อยกว่าฐานข้อมูลที่ไม่มีการใช้อินเด็กซ์ โดยจะเห็นความแตกต่างของเวลาอย่างชัดเจน เมื่อเซลล์เซต (Cell Sets) มีจำนวนมากๆ (หลายแสนเซลล์) ดังเช่นตัวอย่างจากการทดลอง เมื่อข้อมูลมีจำนวนเซลล์ประมาณ 50,400 เซลล์ จะใช้เวลาในการ Query ข้อมูลจากฐานข้อมูลที่มีการใช้อินเด็กซ์และไม่ใช้อินเด็กซ์ พอๆกัน ประมาณ 20 วินาที ทางฝั่งเซิร์ฟเวอร์ และ 55 วินาที ทางฝั่งไคล์เอนท์

7.3.2 จากการทดลอง ทั้งฐานข้อมูลที่มีการใช้อินเด็กซ์และไม่มีการใช้อินเด็กซ์ จะพบว่า การจัดเก็บข้อมูลแบบ MOLAP จะใช้เวลาในการ Query ข้อมูลน้อยที่สุด รองลงมา คือ HOLAP และ ROLAP ตามลำดับ

บทที่ 8
บทสรุปและวิจารณ์

8.1 บทสรุปและวิจารณ์

ในการใช้งานระบบฐานข้อมูลดาต้าแวร์เฮาส์จริง จะต้องทำการวิเคราะห์กับจำนวนข้อมูลที่มีขนาดใหญ่มากๆ ซึ่งเวลาที่ใช้ในการตอบสนองต่อคำถาม (Response Time) จะต้องอยู่ในช่วงที่ยอมรับได้ (ไม่ช้าจนเกินไป) แต่ในการพัฒนาโครงงานนี้ ได้พบข้อจำกัดอยู่หลายประการ ดังต่อไปนี้

1. ในการใช้งานโปรแกรมประยุกต์ทางฝั่งไคลเอนท์ จำเป็นที่จะต้องมีการติดตั้งคอมโพเนนต์เพิ่มเติมอีกหลายคอมโพเนนต์ เช่น MDAC, PivotTable Service และ MSHFlexGrid จึงทำให้มีความยากลำบากในการติดตั้งโปรแกรมประยุกต์

2. เครื่องเซิร์ฟเวอร์ที่ทำหน้าที่ให้บริการ OLAP Services จะต้องเป็นเครื่องที่มีประสิทธิภาพสูง เพื่อรองรับการเข้าใช้งานจากเครื่องไคลเอนท์หลายๆ เครื่อง และมีผลต่อเวลาในการ Process Cubes โดยเฉพาะอย่างยิ่งในการ Process Cubes แบบ ROLAP

3. ในรายละเอียดเกี่ยวกับ Multidimensional Expression (MDX) ยังทำการศึกษาไม่ละเอียดเพียงพอต่อการพัฒนาทางด้านประสิทธิภาพของโปรแกรมประยุกต์

8.2 แนวทางการพัฒนาเพิ่มเติม

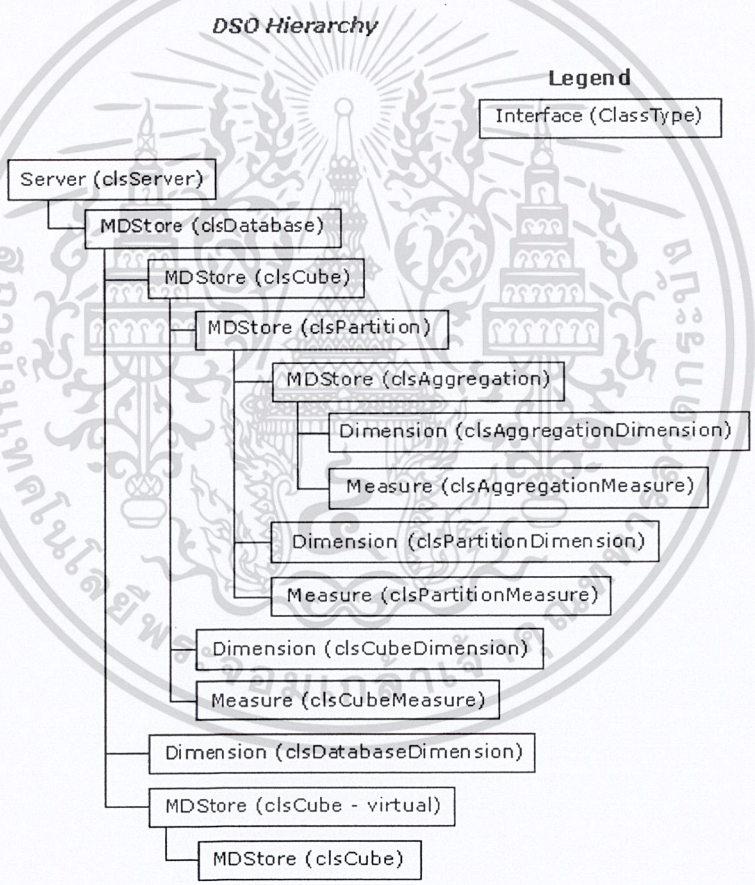
1. ควรทำการศึกษาความสามารถและรายละเอียดของ MDX เพื่อเพิ่มประสิทธิภาพในการตอบคำถาม
2. ควรทำการศึกษาการใช้งานทูลส์ของ OLAP ให้มากขึ้น โดยเฉพาะการจัดการกับทรัพยากรของระบบ
3. ควรทำการศึกษาโครงสร้างและการทำงานของ PivotTable Service เพื่อใช้ในการพัฒนาโปรแกรมประยุกต์ให้มีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก

Decision Support Objects (DSO)

เมื่อ OLAP Service รวมกับโมเดลวัตถุ จะเรียกใหม่ว่า Decision Support Objects (DSO) เป็นส่วนที่ทำหน้าที่ในการติดต่อระหว่าง OLAP Manager และ OLAP Server, DSO Application อาจจะควบคุมการสร้างและการจัดการ Data Cube โดยเซิร์ฟเวอร์อาจจะถูกออกแบบมาเพื่อจัดการกับ Metadata ของ cube ใน Repository , OLAP Manager นั้นเป็นแอพลิเคชันที่อยู่บนพื้นฐานของ DSO Object Model ที่ใช้ในการควบคุม OLAP Server และ Cube metadata

โมเดลของ DSO Object จะแสดงวัตถุของ OLAP Service ในแบบลำดับขั้น โดยเริ่มจากส่วนบนสุด ซึ่งเป็นระดับ Server แล้วจึงตามมาด้วยระดับของ Database, Cubes, Partitions และ Aggregations ดังรูปที่ ก-1



รูปที่ ก-1 DSO Object Model Hierarchy

จากรูปที่ ก-1 สังเกตได้ว่าแต่ละวัตถุจะเก็บกลุ่มของวัตถุอื่นๆ อีก วัตถุ Server จะเก็บกลุ่มของวัตถุที่เป็น OLAP database วัตถุ Database แต่ละตัวก็จะเก็บ Cubes, Shared dimensions และ Virtual cubes ซึ่งแต่ละวัตถุก็จะมีการเก็บวัตถุอื่นๆ อีกต่อไปเป็นลำดับๆ

DSO Objects และ Interfaces

DSO Objects สามารถแบ่งได้ดังนี้

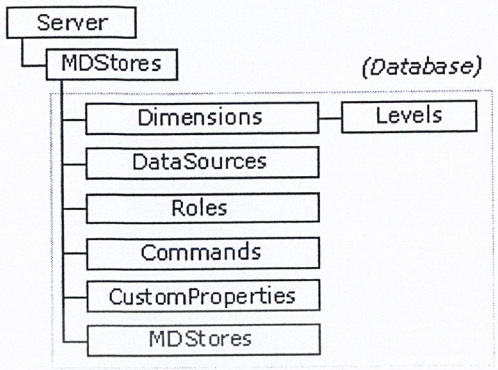
- วัตถุที่สามารถเข้าใช้และจัดการข้อมูลได้โดยตรง วัตถุเหล่านี้จะเก็บเมธอด (method) และพร็อพเพอร์ตี้ (properties) เป็นของตัวเอง ได้แก่ *clsCubeAnalyzer* , *clsPartitionAnalyzer* , *clsDataSource* , *clsServer* และ *clsMemberProperties*
- วัตถุที่ใช้ใน DSO Interfaces จะเป็นวัตถุที่ไม่สามารถควบคุมได้โดยตรง ขึ้นอยู่กับอินเทอร์เฟส (Interface) ที่ใช้ ดังตารางที่ ก-1

Interface	Implemented by
Command	ClsDatabaseCommand clsCubeCommand
Dimension	ClsDatabaseDimension clsCubeDimension clsPartitionDimension clsAggregationDimension
Level	ClsDatabaseLevel clsCubeLevel clsPartitionLevel clsAggregationLevel
MDStore	ClsDatabase clsCube clsPartition clsAggregation
Measure	ClsCubeMeasure clsPartitionMeasure clsAggregationMeasure
Role	ClsDatabaseRole clsCubeRole

ตารางที่ ก-1 DSO Objects และ Interfaces

ตามคำอธิบายข้างต้น จะสามารถอธิบายเป็นลำดับขั้นของคลาสต่างๆได้ดังนี้

1. DSO Database Object



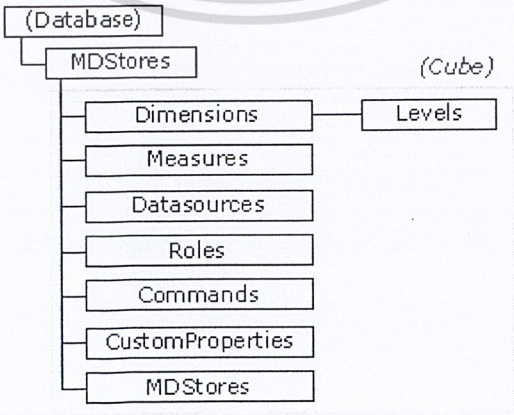
รูปที่ ก-2 DSO Database Object

DSO Database Object จะประกอบด้วยหลายๆ collections คือ

- Dimensions : จะรวมทั้ง Shared และ Private Dimensions
- Levels : เป็น collection ของ Database Dimension
- DataSources : จะใช้ในทุก Object ภายใน Database
- Roles : เป็น collection ของ Role Object เพื่อควบคุมการเข้าใช้งาน Database และ Cube
- Commands : เป็น collection ที่เก็บไว้เพื่อใช้งานในอนาคต
- CustomProperties : อนุญาตให้ผู้ใช้ทำการเพิ่ม Object นี้ตามที่ต้องการได้
- MDStore : เป็น collection ของ Cube และ Virtual Cube

Database จะถูกเก็บอยู่ใน MDStore collection ภายใต้ Server Object (clsServer)

2. DSO Cube Object

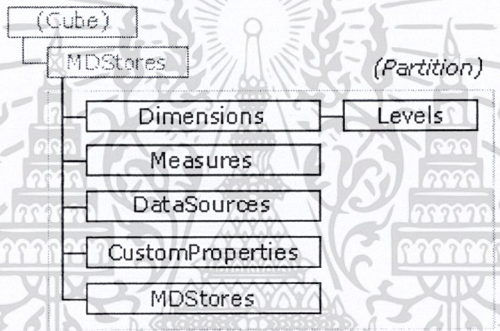


รูปที่ ก-3 DSO Cube Object

DSO Cube Object จะเก็บ collection และ object ต่างๆ ดังนี้

- Dimensions : จะรวม Shared Dimensions ที่ถูกกำหนดไว้ใน Database Level และรวมกับ Dimension ใน cube และ Private Dimension
- Measures : เป็น collection ของ Cube Measures
- DataSources : เป็น collection ที่เก็บ Data Sources ที่ถูกใช้ใน Cubes หรือใน Partitions
- Roles : จะเก็บ Role Object ที่มีอยู่ใน Role collection ในระดับ Database และถูกส่งไปยัง Cube
- Commands : จะเก็บ MDX command และ statements ที่ใช้ในการระบุข้อมูลที่ถูกใช้ใน Cube
- CustomProperties
- MDStore : เป็นตัวเก็บ Partition ใน Cube

3. DSO Partition Objects

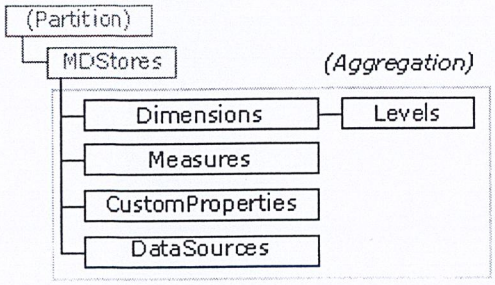


รูปที่ ก-4 แสดง DSO Partition Objects

ทุกๆ Cube จะต้องมีอย่างน้อย 1 Partition และถ้าใช้ OLAP Service version Enterprise จะทำให้ Cube มีได้หลาย Partitions แต่ละ Partition จะประกอบด้วย Objects และ collections ต่างๆ ดังนี้

- Dimensions : Cube dimension จะถูกเก็บอยู่ใน Partition
- Measures : จะเป็น Measure ของ Cube ที่อยู่ใน Partition
- DataSources : เป็น Data Source ของ Partition ถ้ามีเพียง Partition เดียว จะหมายถึง Data Source ของ Cube
- CustomProperties
- MDStore : เป็น collection ที่เก็บ Aggregation ที่สัมพันธ์กับ Partition นั้น

4. DSO Aggregations

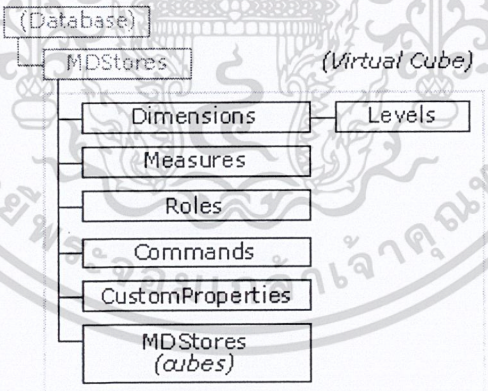


รูปที่ ก-5 แสดง DSO Aggregations

Aggregation จะประกอบด้วย collection ของ

- Dimensions : จะเก็บ Dimension ที่ถูกใช้ โดยการ Aggregate
- Measures : จะเก็บ Measure ที่ใช้ในการคำนวณ Aggregation
- DataSources : เก็บตำแหน่งของข้อมูลที่ถูกใช้ในการ Aggregate
- CustomProperties

5. DSO Virtual Cubes



รูปที่ ก-6 แสดง DSO Virtual Cubes

Virtual Cubes จะถูกใช้เหมือนกับในกรณีของ View ใน Relational Database โดย Virtual Cubes จะทำการรวมหลายๆ Cubes เข้าด้วยกัน และอนุญาตให้ทำการกำหนด Roles และ Permissions เพื่อกำหนดผู้ใช้ในการเข้าถึง Virtual Cubes

DSO Virtual Cubes จะประกอบด้วย collections และ Objects ดังนี้

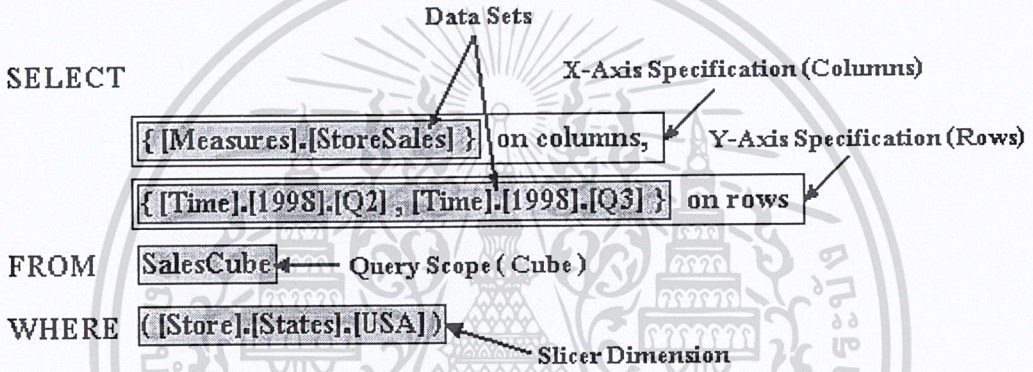
- Dimensions : จะเก็บไดเมนชันต่างๆที่ถูกใช้ใน Virtual Cubes ซึ่งไดเมนชันนั้นต้องมีอยู่จริงใน Cube
- Measures : จะเก็บตัววัดต่างๆ ที่ถูกใช้ใน Virtual Cubes ที่มีอยู่จริงใน Cubes

- Roles : จะเก็บ Roles ที่อนุญาตให้เข้าใช้ Virtual Cubes ซึ่งถูกกำหนดบน Database Level
- Commands : จะเก็บ MDX Expression และ Statements ที่ใช้ในการระบุข้อมูลที่ถูกใช้ใน Cube
- CustomProperties
- MDStore : จะเก็บ Cubes ต่างๆ

ภาคผนวก ข

Multidimensional Expressions (MDX)

Structure Query Language (SQL) เป็นภาษาที่ใช้ในการตั้งคำถามสำหรับฐานข้อมูลรีเลชันแนล มักจะส่งเซตคำตอบแบบ 2 มิติ แต่ถ้าต้องการเซตคำตอบแบบหลายมิติ (Multidimensional) ก็ควรจะใช้ MDX (Multidimensional Expressions) แทน เพราะ MDX ไม่ได้คืนคำตอบมาเป็น spreadsheet แต่จะคืนกลับมาในรูปของ Cube เลย สรุปได้ว่า MDX คือ ภาษาที่กำหนดโดย OLE DB มีไว้สำหรับให้ OLAP นำไปใช้งานกับ Multidimensional data stores ซึ่ง MDX จะประกอบไปด้วย statements และ clauses ที่คล้ายคลึงกับ SQL ที่คล้ายคลึงกับ SQL แต่จะมีความแตกต่างกันตรง Syntax



รูปที่ ข-1 แสดงโครงสร้างของ MDX

จากรูปที่ ข –1 แสดงถึงไวยากรณ์ของการตั้งคำถาม จะได้ว่า

- MDX จะใช้เครื่องหมายวงเล็บปีกกา “{ }” เพื่อแสดงถึงเซตของสมาชิกของไดเมนชันหรือกลุ่มของไดเมนชัน
- แกนทั้งหลายจะถูกกำหนดในการตั้งคำถาม โดยการระบุไดเมนชันลงไป 3 แกนแรกของ MDX จะเรียกว่า Rows, Columns และ Pages ถ้าหากมีมากกว่า 1 ไดเมนชันบน 1 แกน แกนนั้นก็จะประกอบไปด้วย Tuples (ส่วนประกอบ) ซึ่งจะรวมเอาสมาชิกของแต่ละไดเมนชันที่ระบุไว้เข้าด้วยกัน
- ลำดับของแกนจะเริ่มจาก Columns ตามด้วย Rows และสุดท้าย คือ Pages
- ใน From clause จะเป็นชื่อของ Cube ที่ใช้ในการตั้งคำถาม จะมีเพียง 1 Cube เท่านั้นต่อการตั้งคำถาม 1 ครั้ง
- ใน Where clause จะเป็นไดเมนชันที่ถูกตัดข้อมูล โดยจะระบุสมาชิก 1 ตัว ที่อยู่ในไดเมนชันนั้นในคำถาม

Tuple

Tuple เป็นส่วนประกอบของสมาชิกจากไดเมนชันต่างๆ เช่น (Michigan, Q2) เป็น tuple ที่ประกอบไปด้วยสมาชิกจาก 2 ไดเมนชัน คือ Geography และ Time , (Michigan, Q2, food items) เป็น tuple ที่ประกอบไปด้วยสมาชิกจาก 3 ไดเมนชัน คือ Geography, Time และ Product, Tuple นั้นสามารถมีสมาชิกเพียงตัวเดียวก็

ได้ จะเรียกว่า " Degenerated tuple " เช่น Michigan เป็น Degenerated tuple ที่แสดงถึงระดับของ state ที่เป็นสมาชิกของไดเมนชัน Geography และ Tuple นั้นเป็นหน่วยพื้นฐานของแกน (Axis)

แกน (Axis)

แกน เป็นส่วนประกอบของสมาชิกจากไดเมนชัน 1 หรือมากกว่า 1 ไดเมนชัน แกนนั้นใช้สำหรับบอกที่อยู่หรือใช้เป็นตัวแสดงค่าใน Cube เช่น ในการตอบคำถามจาก Cube Sales โดยส่ง State และ Quarter ของ Product family อาจกล่าวได้ว่า จะมีแกนมากสุด 3 แกนที่สามารถเป็นผลลัพธ์ (Result) ของการตั้งคำถาม ซึ่งแต่ละแกนจะแสดง 1 ใน 3 ไดเมนชัน (Geography, Time และ Product) ถ้าเอาบางไดเมนชันออกไป จำนวนแกนก็จะลดลงด้วย

เซลล์เซต (Cellset)

Cellset เป็นผลลัพธ์ของการตั้งคำถาม โดยใช้ MDX บนข้อมูลแบบมัลติไดเมนชันแนล ซึ่ง Cellset จะประกอบไปด้วยส่วนประกอบของแกน ปกติมักจะมีเพียง 2 หรือ 3 แกน

เซลล์ (Cell)

เซลล์เป็นวัตถุที่กำหนดอยู่บนตำแหน่งของแกน แต่ละเซลล์จะมีข้อมูลหลายส่วนประกอบกัน ได้แก่ ตัวข้อมูล, รูปแบบสตริง และค่าแสดงลำดับของเซลล์ (แต่ละเซลล์จะมีค่าลำดับเพียงค่าเดียวใน Cellset) และเซลล์แรกจะมีค่าลำดับเป็น 0

Slicer

ไดเมนชันที่เป็น Slicer จะปรากฏในส่วนของ Where clause ของ MDX statement เช่น จะดูข้อมูลการขายผลิตภัณฑ์หลายประเภทในประเทศ USA ของทุก Quarter ในปี 1998 สามารถสร้าง MDX statement ดังนี้

```
SELECT
    { [Product] . [Product Family] . Members } on columns ,
    { [Time] . [1998] . [Quarters] . Members } on rows ,
    { [Measures] . [Store Sales] } on pages
FROM  SalesCube
WHERE  ( [Store] . [All Stores] . [USA] )
```

ผลลัพธ์จากการส่งค่ากลับมา จะแสดงในให้เห็นในตารางที่ ข – 1 ซึ่งจะทำให้เห็นรายละเอียดที่ชัดเจนขึ้น

Total Sales in the USA			
Product Families			
1998 Quarters	**Food**	**Drinks**	**Cloths**
Q1/1998	$12,525,452	$9,235,765	$11,455,874
Q2/1998	$14,142,343	$12,525,452	$12,525,452
Q3/1998	$15,123,042	$12,525,452	$12,525,452
Q4/1998	$11,002,002	$12,525,452	$12,525,452

ตารางที่ ข – 1 แสดงผลลัพธ์ของการ Slice ข้อมูลโดยใช้ MDX Statement

ฟังก์ชันที่นำมาใช้งาน

CrossJoin

เป็นฟังก์ชันที่ใช้ในการรวมสมาชิกจากเซต 2 เซต เข้าไว้ด้วยกันบนแกนๆ เดียว เช่น

```
SELECT
    CROSSJOIN (
        { [Customers] . [All Customers] . [USA] . [ABC Inc.] ,
            [All Customers] . [Canada] . [XYZ Ltd.] } ,
        { [Product] . [Product Family] . Members } )  on columns ,
        { [Time] . [Year] . [1998] . Children }  on rows ,
        { [Measures] . [Store Sales] }  on pages
FROM   SalesCubes
WHERE  ( [Store] . [All Stores] . [USA] )
```

จาก MDX Statement ข้างต้น จะได้ผลลัพธ์ดังตารางที่ ข – 2

Total Store Sales in the USA						
1998 Quarters	**ABC Inc.**			**XYZ Ltd.**		
	Product Families			**Product Families**		
	Food	**Drinks**	**Cloths**	**Food**	**Drinks**	**Cloths**
Q1/1998	$12,525,452	$12,525,452	$9,235,765	$4,655,874	4,235,765$	$5,455,874

Q2/1998	$14,142,343	$14,142,343	$12,525,452	$5,525,452	$5,525,452	$6,525,452
Q3/1998	$15,123,042	$15,123,042	$12,525,452	$5,525,452	$8,525,852	$7,525,452
Q4/1998	$11,002,002	$11,002,002	$12,525,452	$6,525,452	$7,553,452	$8,525,452

ตารางที่ ข – 2 แสดงผลลัพธ์ของการใช้ฟังก์ชัน CrossJoin( )

Drill – Up และ Drill – Down

การ Drill – Up และ Drill – Down สามารถแบ่งออกได้เป็น 2 ประเภทด้วยกัน คือ

1. Drill by Member มีอยู่ 2 อย่าง ได้แก่

1.1 DrillDownMember ( ) : เป็นการดูรายละเอียดข้อมูลในระดับที่ลึกลงไป 1 ระดับ เฉพาะสมาชิกที่ระบุชื่อไว้เท่านั้น

Syntax : DrillDownMember (Set1 , Set2 [, Recursive])

ตัวอย่าง : DrillDownMember ({ Ford , Honda } , { Ford })

ผลลัพธ์ : { Ford , Taurus , Explorer , Honda }

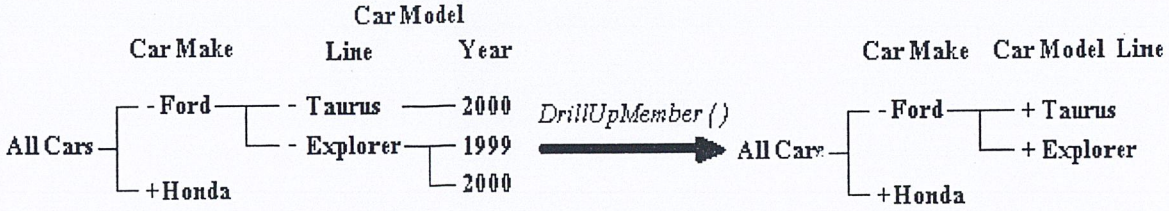
รูปที่ ข – 2 แสดงการ DrillDownMember

1.2 DrillUpMember ( ) : เป็นการดูรายละเอียดข้อมูลที่สูงขึ้น 1 ระดับ

Syntax : DrillUpMember (Set1 , Set2)

ตัวอย่าง : DrillUpMember ({ Ford, Taurus, [Taurus – 2000], Explorer, [Explorer – 1999], [Explorer – 2000], Honda } , { Taurus , Explorer })

ผลลัพธ์ : { Ford , Honda , Taurus , Explorer }



รูปที่ ข – 3 แสดงการ DrillUpMember

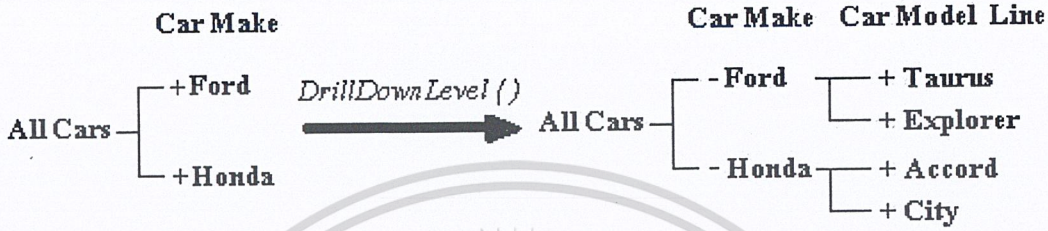
2. Drill by Level มีอยู่ 2 อย่าง เช่นเดียวกันกับ Drill by Member ได้แก่

2.1 DrillDownLevel ( ) : เป็นการดูรายละเอียดข้อมูลในระดับที่ลึกลงไปตามชื่อของระดับที่ระบุไว้

Syntax : DrillDownLevel (Set [, Level])

ตัวอย่าง : DrillDownLevel ({ Ford, Honda } , [Car Make])

ผลลัพธ์ : { Ford, Taurus, Explorer, Honda, Accord, City }



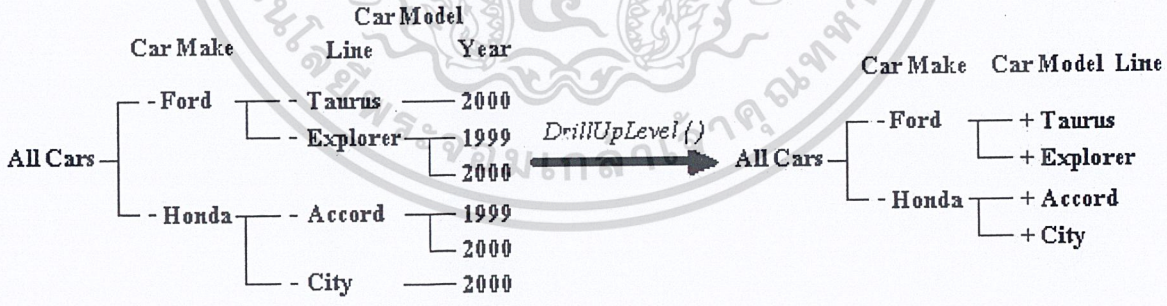
รูปที่ ข – 4 แสดงการ DrillDownLevel

2.2 DrillUpLevel ( ) : เป็นการดูรายละเอียดข้อมูลในระดับที่สูงขึ้น ตามชื่อของระดับที่ระบุไว้

Syntax : DrillUpLevel (Set [, Level])

ตัวอย่าง : DrillUpLevel ({ Ford, Taurus, [Taurus - 2000], Explorer, [Explorer - 1999], [Explorer – 2000], Honda, Accord, [Accord – 1999], [Accord – 2000], City, [City – 2000] } , [Car Model])

ผลลัพธ์ : { Ford , Honda , Taurus , Explorer , Accord , City }



รูปที่ ข – 5 แสดงการ DrillUpLevel

ภาคผนวก ค

Microsoft Hierarchical Flex Grid (MSHFlexGrid)

MSHFlexGrid เป็นคอมโพเนนต์ที่ใช้แสดงและจัดการกับ tabular data โดยจะมีความยืดหยุ่นทางการเรียง (Sort), การผสาน (Merge) และรูปแบบตารางที่ใช้เก็บสตริงและรูปภาพ ซึ่งข้อมูลที่ใช้กับ MSHFlexGrid จะเป็นข้อมูลชนิดที่อ่านเพียงอย่างเดียว (Read-only data)

เราสามารถใส่ข้อความ, รูปภาพหรือทั้งข้อความและรูปภาพ เข้าไปในเซลล์ใดก็ตามของ MSHFlexGrid คุณสมบัติ Row และ Col จะเป็นตัวระบุเซลล์ปัจจุบันใน MSHFlexGrid โดยสามารถระบุเซลล์ปัจจุบันด้วยการเขียนโค้ด หรือผู้ใช้สามารถเปลี่ยนแปลงในขณะ run time ก็ได้ ส่วนคุณสมบัติทางข้อความ (Text) ก็เกี่ยวข้องกับเซลล์ปัจจุบันด้วย ถ้าข้อความในเซลล์มีความยาวมากเกินไปที่จะแสดงภายในเซลล์ และยังมีคุณสมบัติอื่นๆ ที่ใช้ในการสร้างแอพลิเคชัน ได้แก่คุณสมบัติต่างๆ ดังต่อไปนี้

1. AllowUserResizing Property

เป็นคุณสมบัติที่ใช้ในการปรับขนาดของ Row และ Column ของ MSHFlexGrid โดยการใช้เมาส์

Syntax : *object.* **AllowUserResizing** [=*value*]

ค่าคงที่	Value	คำอธิบาย
FlexResizeNone	0	ผู้ใช้ไม่สามารถปรับเปลี่ยนขนาดโดยใช้เมาส์ได้
FlexResizeColumns	1	ผู้ใช้สามารถปรับเปลี่ยนขนาดของคอลัมน์โดยใช้เมาส์ได้
FlexResizeRows	2	ผู้ใช้สามารถปรับเปลี่ยนขนาดของแถวโดยใช้เมาส์ได้
FlexResizeBoth	3	ผู้ใช้สามารถปรับเปลี่ยนขนาดของแถวและคอลัมน์โดยใช้เมาส์ได้

ตารางที่ ค-1 แสดงการตั้งค่าใน AllowUserResizing property

2. Col, Row Properties

เป็นคุณสมบัติที่ใช้บอกตำแหน่ง (Coordinate) ของแถวและคอลัมน์ของเซลล์ที่กำลังใช้งานใน MSHFlexGrid

Syntax : *object*.**Col** [=*number*]

object.**Row** [=*number*]

3. Cols, Rows Properties

- **Cols** - ใช้กำหนดจำนวนของคอลัมน์ใน MSHFlexGrid
- **Rows** - ใช้กำหนดจำนวนของแถวใน MSHFlexGrid

สามารถใช้คุณสมบัติเหล่านี้ในการขยายและเชื่อมต่อ MSHFlexGrid ได้แบบไดนามิกในขณะที่ run time จำนวนแถวและคอลัมน์ที่น้อยที่สุด คือ 0 ,และมีได้มากที่สุดตามขีดจำกัดของหน่วยความจำในเครื่องคอมพิวเตอร์

ค่า (value) ของ Cols และ Rows จะต้องมีค่ามากกว่าค่า FixedCols และ FixedRows อยู่อย่างน้อย 1

Syntax : *object*.**Cols** [= *value*]

object.**Rows** (*number*) [= *value*]

4. FixedCols, FixedRows Properties

- **FixedCols** - ใช้กำหนดจำนวนคอลัมน์ที่ตายตัว (Fixed Columns) ทั้งหมดใน MSHFlexGrid
- **FixedRows** - ใช้กำหนดจำนวนแถวที่ตายตัว (Fixed Rows) ทั้งหมดใน MSHFlexGrid

โดยปกติแล้ว MSHFlexGrid จะมีคอลัมน์และแถวที่ตายตัวอยู่อย่างละ 1

Syntax : *object*.**FixedCols** [= *value*]

object.**FixedRows** [= *value*]

5. CellBackColor Property

เป็นคุณสมบัติที่ใช้กำหนดสีพื้นหลังของเซลล์แต่ละเซลล์

Syntax : *object*.**CellBackColor** [=*color*]

6. Height, Width Properties

เป็นคุณสมบัติที่ใช้กำหนดไดเมนชันของวัตถุหรือความกว้างของคอลัมน์ของ DataGrid control

Syntax : *object*.**Height** [= *number*]

object.**Width** [= *number*]

7. MergeCells Property

เป็นคุณสมบัติในการผสานเซลล์ที่มีรายละเอียดข้อมูลเหมือนกัน (มีหลายแถวหรือหลายคอลัมน์) ให้เป็นเซลล์อันเดียวกัน

Syntax : *object*.**MergeCells** [=*value*]

ค่าคงที่	Value	คำอธิบาย
FlexMergeNever	0	เซลล์ที่มีรายละเอียดข้อมูลแบบไม่เป็นกลุ่ม
FlexMergeFree	1	เซลล์ที่มีการผสานกันเสมอ
FlexMergeRestrictRows	2	เฉพาะเซลล์สุดท้ายของแถวเท่านั้น (ไปทางซ้ายมือ) ที่ทำการผสาน

FlexMergeRestrictColumns	3	เฉพาะเซลล์สุดท้ายของคอลัมน์เท่านั้น(ขึ้นด้านบน) ที่ทำการผสาน
FlexMergeRestrictBoth	4	เฉพาะเซลล์สุดท้ายของแถวหรือคอลัมน์เท่านั้นที่ทำการผสาน

ตารางที่ ค-2 แสดงการตั้งค่าใน MergeCells property

8. MergeCol, MergeRow Properties

เป็นคุณสมบัติที่กำหนดค่าที่ตรวจสอบว่าแถวและคอลัมน์ใดที่สามารถทำการผสานกันได้ ถ้าจะเรียกใช้คุณสมบัติ MergeCells จะต้องกำหนดค่าเหล่านี้ให้เป็น True ก่อน

Syntax : *object*.**MergeCol**(*number*) [=*Boolean*]

object.**MergeRow**(*number*) [=*Boolean*]

9. MouseCol, MouseRow Properties

เป็นคุณสมบัติที่ใช้ตรวจสอบตำแหน่ง (แถวและคอลัมน์) ของเมาส์

Syntax : *object*.**MouseCol** [=*value*]

object.**MouseRow** [=*value*]

10. TextMatrix Property

เป็นคุณสมบัติที่กำหนดข้อความของเซลล์

Syntax : *object*.**TextMatrix**(*rowindex, colindex*) [=*string*]

11. Left, Top Properties

- **Left** - ใช้กำหนดระยะห่างระหว่างขอบซ้ายสุดของวัตถุและขอบซ้ายของ container ของมัน
- **Top** - ใช้กำหนดระยะห่างระหว่างขอบบนสุดของวัตถุและขอบบนของ container ของมัน

Syntax : *object*.**Left** [= *value*]

object.**Top** [= *value*]

ภาคผนวก ง
คู่มือสำหรับโปรแกรมเมอร์

ฟอร์มทั้งหมดที่แบบสำหรับการใช้งาน

Form1 : มีชื่อว่า DialogConnect เป็นฟอร์มแรกที่จะแสดงเมื่อทำการรันโปรแกรม เพื่อใช้ในการทำการติดต่อกับ OLAP Server

Form2 : มีชื่อว่า FormMainClient เป็นฟอร์มหลักในการทำงานเพื่อใช้ในการวิเคราะห์ข้อมูล

รายละเอียดโปรแกรมย่อย

Form1 : DialogConnect

ตัวแปรต่างๆ

1. OKButtonSt As String ใช้ในการเก็บค่าสถานะในการกดปุ่ม OK, Cancel

Procedure และ function ต่างๆ

1. Private Sub CancelButton_Click( )
   - เมื่อทำการคลิก ที่ปุ่ม cancel ฟอร์ม Dialog จะถูกปิดลง
   - เข้าสู่ฟอร์ม Main ของโปรแกรมโดยที่ไม่มีการติดต่อกับเซิร์ฟเวอร์
2. Private Sub OKButton_Click( )
   - เมื่อทำการคลิก ที่ปุ่ม OK ฟอร์ม Dialog จะถูกปิดลง
   - ทำการติดต่อกับ OLAP Server
   - เข้าสู่ฟอร์ม Main ของโปรแกรม
3. Private Sub Form_Load( )
   - เตรียมสถานะปุ่มต่างๆ ให้กับฟอร์มหลัก
4. Private Sub txtInitialDB_KeyPress(KeyAscii As Integer)
   - ใช้ตรวจสอบ Enter Key เมื่อมีการกด enter โปรแกรมจะเก็บชื่อ Database ไว้
5. Private Sub txtServer_KeyPress(KeyAscii As Integer)
   - ใช้ตรวจสอบ Enter Key เมื่อมีการกด enter โปรแกรมจะเก็บชื่อเซิร์ฟเวอร์ไว้

Form2 : FormMainClient

ตัวแปรต่างๆ

1.	m_ctActiveCatalog As Catalog	ใช้เก็บชื่อดาต้าเบสที่ได้ทำการติดต่ออยู่
2.	m_cn As New ADODB.Connection	สร้างอ็อบเจ็ค Connection ขึ้นเพื่อใช้ในการติดต่อกับ OLAP Server
3.	cat As New ADOMD.Catalog	สร้างอ็อบเจ็ค Catalog ขึ้นเพื่อใช้ในการเก็บชื่อ Database
4.	cdf As ADOMD.CubeDef	ใช้ในการตรวจสอบและเก็บโครงสร้างของ Cube
5.	dm As ADOMD.Dimension	ใช้ในการตรวจสอบและเก็บโครงสร้างของ Dimension
6.	hr As ADOMD.Hierarchy	ใช้ในการตรวจสอบโครงสร้างแบบลำดับขั้น
7.	lv As ADOMD.Level	ใช้ในการตรวจสอบและเก็บโครงสร้างของ Level
8.	mb As ADOMD.Member	ใช้ในการตรวจสอบและเก็บโครงสร้างของ Member
9.	sCatalogCur As String	เป็นตัวแปรชั่วคราวที่ใช้เก็บชื่อของ Database
10.	sConnectStr As String	ใช้เก็บสตริงที่ใช้ในการติดต่อกับ OLAP Server
11.	WindowSt As String	ใช้เก็บสถานะของ windows ว่าอยู่ในสถานะใด
12.	countMeasures As Integer	ใช้เก็บจำนวนของตัววัดที่มีใน Cube
13.	cbNameTemp As String	เป็นตัวแปรชั่วคราวที่ใช้เก็บชื่อของ Cube
14.	cubeName As String	ใช้เก็บชื่อของ Cube
15.	membercol As String	ใช้เก็บชื่อของสมาชิกในแกนตั้ง เพื่อสร้าง MDX
16.	memberrow As String	ใช้เก็บชื่อของสมาชิกในแกนนอน เพื่อสร้าง MDX
17.	memberwhere As String	ใช้เก็บชื่อของสมาชิกในแกน slicer เพื่อสร้าง MDX
18.	strMeasure As String	ใช้สร้าง MDX ในส่วนของตัววัด
19.	memberRowMeasure As String	ใช้สร้าง MDX ในส่วนของตัววัดที่อยู่ในแกนนอน
20.	allowMeasure As String	ใช้เพื่อระบุว่าตัววัดอยู่ในแกนตั้งหรือแกนนอน
21.	nFixedCols As Integer	ใช้เพื่อกำหนดว่าจำนวนแกนตั้งมีทั้งหมดเท่าใด
22.	nFixedRows As Integer	ใช้เพื่อกำหนดว่าจำนวนแกนนอนมีทั้งหมดเท่าใด
23.	iCol As Long	ใช้ระบุตำแหน่งของคอลัมน์ที่จะทำการใส่ค่า caption ของเซลล์ในแกนตั้ง
24.	cCol As Long	ใช้ระบุตำแหน่งของคอลัมน์ที่จะทำการใส่ค่า value ของเซลล์
25.	iRow As Long	ใช้ระบุตำแหน่งของ row ที่จะทำการใส่ค่า caption ของเซลล์ในแกนนอน
26.	cRow As Long	ใช้ระบุตำแหน่งของ row ที่จะทำการใส่ค่า value ของเซลล์
27.	conn As New ADODB.Connection	สร้างอ็อบเจ็ค Connection ขึ้นเพื่อใช้ในการติดต่อกับ OLAP Server

28. cst As New ADOMD.Cellset	สร้างอ็อบเจ็คเพื่อใช้ในการอ้างอิงถึงแต่ละเซลล์
29. axs As ADOMD.Axis	ใช้ในการระบุถึงแกนตั้งและแกนนอน
30. pos As ADOMD.Position	ใช้ในการระบุตำแหน่งของไดเมนชันในแต่ละแกน
31. colPosDrill As Integer	ใช้เพื่อระบุตำแหน่งของคอลัมน์ที่จะทำการ drilldown หรือ drillup
32. temDMdrill As String	เป็นตัวแปรชั่วคราวเพื่อใช้เก็บชื่อไดเมนชันที่จะทำการ drilldown หรือ drillup

Procedure และ Function ต่างๆ

1. Private Sub Command1_Click ( )
   - ทำการตรวจสอบตัววัดว่ามีอะไรบ้าง และสร้าง MDX statement ในส่วนของตัววัดขึ้นมา
   - ทำการสร้าง MDX statement ทั้งหมดซึ่งประกอบขึ้นด้วยส่วนของแกน row , column , slicer
   - นำ MDX statement ที่สมบูรณ์แล้วนั้นมาเก็บไว้ที่ TxtQuery
2. Private Sub Command2_Click ( )
   - ทำการตรวจสอบว่ามีการเลือกไดเมนชันมาแสดงเป็นแกน row และ แกน column หรือไม่
   - ในกรณีที่มีทั้งแกน row และ แกน column ก็จะทำการลบโครงสร้างของ grid และทำการเรียก Procedure FillGrid เพื่อทำการส่ง MDX statement ไปยัง OLAP Server และเรียก Procedure FillButton เพื่อทำการใส่ชื่อของ level ลงใน grid
   - ในกรณีที่ไม่มีแกนใดแกนหนึ่ง ก็จะขึ้นกรอบข้อความเตือนว่า "Please select Rows and Columns Axis"
3. Private Sub Command3_Click ( )
   - ทำการลบ MDX Statement ที่เก็บอยู่ที่ TxtQuery
   - ทำการลบโครงสร้างทั้งหมดที่โปรแกรมได้ทำการจดจำไว้ทั้งหมด เช่น โครงสร้างของ tree
4. Private Sub CubeDefTree_BeforeLabelEdit (Cancel As Integer)
   - ทำการตั้งค่าไว้ว่าไม่อนุญาต ให้ทำการแก้ไขชื่อโหนดของ CubeDefTree
5. Private Sub CubeDefTree_Click ( )
   - เป็น Procedure ที่ทำงานร่วมกับ CubeDefTree_BeforeLabelEdit (Cancel As Integer)
6. Private Sub CubeDefTree_DblClick ( )
   - เป็น Procedure ที่ทำการติดต่อกับ OLAP Server อีกครั้งหนึ่งหลังจากการติดต่อครั้งแรกแล้ว โดยที่เมื่อมีการดับเบิ้ลคลิกบน level โปรแกรมจะทำการแสดงสมาชิกของ level นั้น
7. Private Sub CubeDefTree_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
   - ทำการตรวจสอบการคลิกเมาส์ ถ้าเป็นปุ่มขวา ให้ทำการแสดง popup menu ขึ้นมา
   - ในกรณีที่คลิกขวานั้นเป็นอยู่บนไดเมนชัน และ ไม่ใช่ไดเมนชันที่เป็นตัววัดค่า ก็จะปรากฏ popup menu mnuRowCol ขึ้นมา

- ในกรณีที่คลิกขวานั้นเป็นอยู่บนตัววัด ก็จะปรากฏ popup menu mnuRowColMB ขึ้นมา

8. Private Sub Form_Load ( )
   - จะทำการแสดงฟอร์มขึ้นมาพร้อมกับฟอร์ม DialogConnect
   - ทำการตั้งค่าเริ่มต้นให้กับปุ่มต่างๆบน ToolBar และสถานะต่างๆ
9. Private Sub Form_Resize ( )
   - จะทำการกำหนดค่าตำแหน่งและขนาดของอ็อบเจ็คต่างๆ บนฟอร์มหลัก โดยจะมีการตรวจสอบสถานะต่างๆ เช่น ขณะนั้นเป็น Split View หรือ Result View และ Windows อยู่ในสถานะ Maximize หรือ Normal
10. Private Sub mnuAsCol_Click ( )
    - ทำการตรวจสอบว่าไดเมนชันที่เลือกเพื่อนำมาแสดงในแกนตั้ง นั้นมาจาก cube เดียวกันกับที่ทำงานอยู่หรือไม่
    - ในกรณีที่มาจาก cube เดียวกัน ก็จะทำการตรวจสอบต่อไปว่าไดเมนชันนี้เคยถูกเลือกหรือยัง
    - ในกรณีที่ไดเมนชันนั้นยังไม่เคยถูกเลือก ก็จะทำการสร้างส่วนของ MDX statement ของไดเมนชันนั้นเก็บไว้ที่ ListBox ต่างๆ
    - ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้
11. Private Sub mnuAsColMB_Click ( )
    - ตรวจสอบว่าตัววัดที่เลือกสามารถแสดงในแกนตั้ง และอยู่ใน cube เดียวกันหรือไม่
    - ในกรณีที่สามารถอยู่ในแกนตั้งได้และถูกเลือกจาก cube เดียวกัน ก็จะทำการเพิ่มชื่อตัววัดนั้นลงใน ListBox เพื่อเก็บค่าแต่ละตัววัดไว้
    - ทำการเพิ่มโหนดของ PivotTree ภายใต้ root ที่ชื่อ Measure
    - ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้
12. Private Sub mnuAsRow_Click ( )
    - ทำการตรวจสอบว่าไดเมนชันที่เลือกเพื่อนำมาแสดงในแกนนอน นั้นมาจาก cube เดียวกันกับที่ทำงานอยู่หรือไม่
    - ในกรณีที่มาจาก cube เดียวกัน ก็จะทำการตรวจสอบต่อไปว่าไดเมนชันนี้เคยถูกเลือกหรือยัง
    - ในกรณีที่ไดเมนชันนั้นยังไม่เคยถูกเลือก ก็จะทำการสร้างส่วนของ MDX statement ของไดเมนชันนั้นเก็บไว้ที่ ListBox ต่างๆ
    - ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้
13. Private Sub mnuAsRowMB_Click ( )
    - ตรวจสอบว่าตัววัดที่เลือกสมารถแสดงในแกนนอน และอยู่ใน cube เดียวกันหรือไม่
    - ในกรณีที่สามารถอยู่ในแกนนอนได้และถูกเลือกจาก cube เดียวกัน ก็จะทำการเพิ่มชื่อตัววัดนั้นลงใน ListBox เพื่อเก็บค่าแต่ละตัววัดไว้
    - ทำการเพิ่มโหนดของ PivotTree ภายใต้ root ที่ชื่อ Measure

- ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้

14. Private Sub mnuClearQuery_Click ( )
    - ทำการเรียก Command3_Click เพื่อทำการลบ MDX statement
15. Private Sub mnuColDrillDown_Click ( )
    - ถ้าเซลล์ที่ทำการคลิกขวาเป็นสีฟ้า จะทำการเรียก Procedure DrillDownCol
16. Private Sub mnuColDrillUp_Click ( )
    - ถ้าเซลล์ที่ทำการคลิกขวาเป็นสีฟ้า จะทำการเรียก Procedure DrillUpCol
17. Private Sub mnuConnect_Click ( )
    - จะปรากฏฟอร์ม DialogConnect และตั้งค่าสถานะปุ่ม connect ให้ถูกกดลง
18. Private Sub mnuD2C_Click ( )
    - ทำการย้ายไดเมนชันจากแกนนอนไปยังแกนตั้ง โดยทำการเพิ่มโหนดใน PivotTree ภายใต้ root ชื่อ columns
    - ทำการเพิ่มชื่อไดเมนชันนั้นไปเก็บไว้ใน ListBox ต่างๆ ที่ทำการเก็บชื่อของไดเมนชันที่จะนำมาแสดงในแกนตั้ง
    - ทำการลบโหนดใน PivotTree ที่ได้ทำการเลือกที่จะย้าย และลบชื่อไดเมนชันนั้นออกจาก ListBox ที่เก็บชื่อของไดเมนชันที่จะนำมาแสดงในแกนนอน
    - ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้
19. Private Sub mnuD2R_Click ( )
    - ทำการย้ายไดเมนชันจากแกนตั้งไปยังแกนนอน โดยทำการเพิ่มโหนดใน PivotTree ภายใต้ root ชื่อ rows
    - ทำการเพิ่มชื่อไดเมนชันนั้นไปเก็บไว้ใน ListBox ต่างๆ ที่ทำการเก็บชื่อของไดเมนชันที่จะนำมาแสดงในแกนนอน
    - ทำการลบโหนดใน PivotTree ที่ได้ทำการเลือกที่จะย้าย และลบชื่อไดเมนชันนั้นออกจาก ListBox ที่เก็บชื่อของไดเมนชันที่จะนำมาแสดงในแกนตั้ง
    - ทำการเรียก Command1_Click เพื่อนำไปสร้างเป็น MDX statement เก็บไว้
20. Private Sub mnuDisconnect_Click ( )
    - ทำการตั้งค่าสถานะปุ่ม connect ให้มีลักษณะไม่ถูกกด และตั้งค่าของโปรแกรมให้อยู่ในสถานะ disconnect
    - เรียก Procedure DisConnect เพื่อทำการยกเลิกการติดต่อจาก OLAP Server
21. Private Sub mnuQuery_Click ( )
    - ทำการตรวจสอบปุ่ม query บน ToolBar ว่าอยู่ในสถานะถูกกดหรือไม่
    - ในกรณีที่ถูกกด จะทำการตั้งค่า windowSt1 ให้เป็น "querystate" และทำการเรียก Procedure Form_Resize

- ในกรณีที่ไม่ถูกกด จะทำการตั้งค่า windowSt1 ให้เป็น " notquerystate " และทำการเรียก Procedure Form_Resize

22. Private Sub mnuRemove_Click ( )
    - ทำการลบ dimension ที่ถูกเลือกออกจากแกนตั้ง , แกนนอน และ แกน slicer โดยที่จะทำการลบโหนดออกจาก PivotTree ภายใต้ชื่อ root ที่ได้เลือกไว้ และทำการลบ item ออกจาก ListBox ต่างๆด้วย
23. Private Sub mnuResult_Click ( )
    - ทำการตรวจสอบเครื่องหมายหน้า menu result และทำการนำเครื่องหมายหน้า menu split ออก และทำการตั้งค่าสถานะปุ่ม result บน ToolBar ให้มีลักษณะถูกกดลง
    - ทำการตั้งค่า windowSt ให้เป็น " resultstate " และทำการเรียก Procedure Form_Resize
24. Private Sub mnuRowDrillDown_Click ( )
    - ถ้าเซลล์ที่ทำการคลิกขวาเป็นสีฟ้า จะทำการเรียก Procedure DrillDownRow
25. Private Sub mnuRowDrillUp_Click ( )
    - ถ้าเซลล์ที่ทำการคลิกขวาเป็นสีฟ้า จะทำการเรียก Procedure DrillUpRow
26. Private Sub mnuRunQuery_Click ( )
    - ทำการเรียก Procedure Command2_Click เพื่อทำการส่ง query ไปยัง OLAP Server
27. Private Sub mnuSplit_Click ( )
    - ทำการตรวจสอบเครื่องหมายหน้า menu split และทำการนำเครื่องหมายหน้า menu result ออก และทำการตั้งค่าสถานะปุ่ม split บน ToolBar ให้มีลักษณะถูกกดลง
    - ทำการตั้งค่า windowSt ให้เป็น " splitstate " และทำการเรียก Procedure Form_Resize
28. Private Sub MSFlexGrid_Click ( )
    - ทำการแสดงตำแหน่งของเซลล์ ว่ามีพิกัด row และ column ที่เท่าไร
29. Private Sub MSFlexGrid_DblClick ( )
    - เก็บค่าตำแหน่งของ row และ column ที่ทำการดับเบิ้ลคลิก
    - ในกรณีที่เซลล์นั้นมีสีฟ้า ให้นำค่าของเซลล์นั้นเก็บไว้ที่ Text4 และทำการเรียก Procedure DrillDownRow
30. Private Sub MSFlexGrid_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
    - ทำการตรวจสอบการคลิกเมาส์ ถ้าเป็นปุ่มขวา ให้ทำการเก็บค่า row และ column ไว้
    - ในกรณีที่เซลล์นั้นมีสีฟ้า ให้นำค่าของเซลล์นั้นเก็บไว้ที่ Text4 และทำการเรียก PopupMenu mnuGridDrillRow
31. Private Sub MSHFlexGrid1_DblClick ( )
    - เก็บค่าตำแหน่งของ row และ column ที่ทำการดับเบิ้ลคลิก

- ในกรณีที่เซลล์นั้นมีสีฟ้า ให้นำค่าของเซลล์นั้นเก็บไว้ที่ Text4 และทำการเรียก Procedure DrillDownCol

1. Private Sub MSHFlexGrid1_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
   - ทำการตรวจสอบการคลิกเมาส์ ถ้าเป็นปุ่มขวา ให้ทำการเก็บค่า row และ column ไว้
   - ในกรณีที่เซลล์นั้นมีสีฟ้า ให้นำค่าของเซลล์นั้นเก็บไว้ที่ Text4 และทำการเรียก PopupMenu mnuGridDrillCol
2. Private Sub munExit_Click ( )
   - ทำการเรียก Procedure DisConnect
   - และทำการปิดฟอร์ม DialogConnect และ FormMainClient เพื่อคืนทรัพยากรให้กับระบบ
3. Private Sub PivotTree_MouseUp (Button As Integer, Shift As Integer, x As Single, y As Single)
   - ทำการตรวจสอบการคลิกเมาส์ ว่าเป็นปุ่มขวาหรือไม่
   - ในกรณีที่คลิกขวาที่โหนดที่อยู่ภายใต้รูปที่แสดงถึงแกนตั้งให้ทำการ disable " mnuD2C " และ enable " mnuD2R " จากนั้นจะปรากฏ PopupMenu " mnuPivot " ขึ้นมา
   - ในกรณีที่คลิกขวาที่โหนดที่อยู่ภายใต้รูปที่แสดงถึงแกนนอนให้ทำการ disable " mnuD2R " และ enable " mnuD2C " จากนั้นจะปรากฏ PopupMenu " mnuPivot " ขึ้นมา
   - ในกรณีที่คลิกขวาที่โหนดที่อยู่ภายใต้รูปที่แสดงถึงแกน slicer ให้ทำการ disable " mnuD2C " และ " mnuD2R " จากนั้นจะปรากฏ PopupMenu " mnuPivot " ขึ้นมา
4. Private Sub Toolbar1_ButtonClick (ByVal Button As MSComctlLib.Button)
   - ทำการตรวจสอบว่าปุ่มที่กดบน ToolBar นั้นมี key คืออะไร
   - ในกรณีที่ key คือ " disconnect " ก็จะทำการตั้งค่าสถานะปุ่มให้ถูกกดลง และทำการลบโหนดทั้งหมดของ CubeDefTree เพื่อเตรียมพร้อมในการติดต่อกับ OLAP Server ต่อไป ทำการแสดงฟอร์ม DialogConnect เพื่อทำการรับค่าต่างๆ ที่ใช้ในการติดต่อกับเซิร์ฟเวอร์ จากนั้นทำการเรียก Procedure GetConnect
   - ในกรณีที่ key คือ " connect " ก็จะทำการตั้งค่าสถานะปุ่มให้ถูกยกขึ้น จากนั้นทำการเรียก Procedure DisConnect
   - ในกรณีที่ key คือ " Resultview " ก็จะทำการตั้งค่าสถานะของ WindowSt ให้เป็น " resultstate " และทำการตรวจสอบเครื่องหมายถูกหน้า result พร้อมกับนำเครื่องหมายถูกหน้า split บนไฟล์เมนูออก จากนั้นทำการเรียก Procedure Form_Resize
   - ในกรณีที่ key คือ " Splitview " ก็จะทำการตั้งค่าสถานะของ WindowSt ให้เป็น " splitstate " และทำการตรวจสอบเครื่องหมายถูกหน้า " split " พร้อมกับนำเครื่องหมายถูกหน้า result บนไฟล์เมนูออก จากนั้นทำการเรียก Procedure Form_Resize

- ในกรณีที่ key คือ " Queryview " ก็จะทำการตั้งค่าสถานะของ WindowSt ให้เป็น "Querystate" และทำการตรวจสอบเครื่องหมายถูกหน้า "query "จากนั้นทำการเรียก Procedure Form_Resize
- ในกรณีที่ key คือ " notqueryview " ก็จะทำการตั้งค่าสถานะของ WindowSt ให้เป็น "notquerystate " และทำการนำเครื่องหมายถูกหน้า "query " ออกจากนั้นทำการเรียก Procedure Form_Resize
- ในกรณีที่ key คือ " RunQuery " จะทำการเรียก Procedure Command2_Click
- ในกรณีที่ key คือ " ClearQuery " จะทำการเรียก Procedure Command3_Click

5. Public Sub GetConnect ( )
   - เป็น Procedure หลักในการติดต่อกับ OLAP Server โดยจะทำการสร้าง Connection String ขึ้นมา และทำการส่ง Connection String โดยเมธอด open ของอ็อบเจ็ค connection
   - ทำการเรียก Procedure UpdateDatabaseTree และทำการสร้าง root ของ PivotTree ทั้งหมด
6. Public Sub UpdateServerTree ( )
   - ทำการสร้าง root ชื่อเซิร์ฟเวอร์ ให้กับ CubeDeftree โดยชื่อของเซิร์ฟเวอร์นั้นได้มาจากชื่อที่ผู้ใช้ใส่ในฟอร์ม DialogConnect
7. Private Sub UpdateDatabaseTree (cnActive As ADODB.Connection)
   - ทำการลบโหนดทั้งหมดของ CubeDefTree และเรียก Procedure UpdateServerTree
   - ทำการตรวจสอบว่าสามารถติดต่อเซิร์ฟเวอร์ได้หรือไม่ โดยอาศัยเมธอด state ของอ็อบเจ็ค connection
   - ในกรณีที่ติดต่อเซิร์ฟเวอร์ได้ ต่อไปจะทำการแสดงชื่อ database โดยใช้เมธอด OpenSchema และ Fields และทำการเรียก Procedure UpdateCubeTree
8. Public Sub UpdateCubeTree ( )
   - ทำการสร้าง Connection String ขึ้นมา และทำการส่ง Connection String โดยเมธอด open ของอ็อบเจ็ค connection
   - ในกรณีที่ติดต่อเซิร์ฟเวอร์ได้ ต่อไปจะทำการแสดงชื่อของ cube โดยใช้เมธอด OpenSchema และ CubeDefs และทำการเรียก Procedure UpdateDimension
9. Public Sub DisConnect ( )
   - ทำการตั้งค่า ActiveConnection ให้เป็น Nothing
   - ทำการลบโหนดทั้งหมดของ CebeDefTree , PivotTree และ TreeMem ออกเพื่อเตรียมสำหรับการติดต่อเซิร์ฟเวอร์ ครั้งต่อไป
10. Public Sub StatusBarSt (stat As String)
    - ทำการตรวจสอบค่าตัวแปรที่ถูกส่งเข้ามา
    - ในกรณีที่ตัวแปรมีค่าเป็น " connst " ก็จะทำการโหลดรูป "connst.bmp " เพื่อแสดงถึงการติดต่อกับเซิร์ฟเวอร์และทำการแสดงชื่อของเซิร์ฟเวอร์บน StatusBar

- ในกรณีที่ตัวแปรมีค่าเป็น " disconnst " ก็จะทำการโหลดรูป " disconst.bmp " เพื่อแสดงถึงการยกเลิกการติดต่อกับเซิร์ฟเวอร์และทำการแสดงคำว่า " Not Connected " บน StatusBar

11. Public Sub UpdateDimension ( )
    - ทำการตรวจสอบว่าใน OLAP Server นั้นมี cube อยู่อย่างน้อย 1 cube
    - ในกรณีที่ที่ cube อยู่ โปรแกรมจะทำการแสดงไดเมนชันทั้งหมดของแต่ละ cube โดยใช้เมธอด Dimensions.Item (index).Name และทำการเรียก Procedure UpdateLevel ต่อไป
12. Public Sub UpdateLevel ( )
    - โปรแกรมจะทำการแสดง level ทั้งหมดของแต่ละไดเมนชัน โดยใช้เมธอด hr.Levels.Item (index).Name
    - ทำการตรวจสอบว่าไดเมนชันนั้นมีชื่อว่า " Measures" หรือไม่ ถ้าใช่จะทำการเรียก Procedure UpdateMember
13. Public Sub UpdateMember ( )
    - โปรแกรมจะทำการแสดงสมาชิกทั้งหมดของแต่ละ level ออกมา โดยใช้เมธอด mb.Name ซึ่ง mb ก็คือ อ็อบเจ็ค member นั่นเอง
14. Public Function checkcube ( ) As Boolean
    - ทำการตรวจสอบว่าไดเมนชันที่เลือกนั้นมาจาก cube เดียวกันหรือไม่
    - ในกรณีที่ " ใช่ " ฟังก์ชันจะส่งค่าคืนกลับเป็น true
    - ในกรณีที่ " ไม่ใช่ " ฟังก์ชันจะส่งค่าคืนกลับเป็น false พร้อมกับขึ้นกรอบข้อความ " Select dimension from same cube "
15. Public Function CheckList (listItem As String) As Boolean
    - ทำการตรวจสอบว่าไดเมนชันที่เลือกนั้นเคยถูกเลือกแล้วหรือยัง
    - ในกรณีที่ " เคยถูกเลือก " ฟังก์ชันจะส่งค่าคืนกลับเป็น false พร้อมกับขึ้นกรอบข้อความ "Dimension exist "
    - ในกรณีที่ " ไม่เคยถูกเลือก " ฟังก์ชันจะส่งค่าคืนกลับเป็น true
16. Public Sub FillGrid ( )
    - ทำหน้าที่แสดง Cellset ลงใน MSHFlexGrid ทั้งหมด โดยแบ่งการทำงานออกเป็น 4 ส่วนหลัก
    - ส่วนที่1 ทำการส่ง MDX statement ไปยัง OLAP server และทำการตั้งค่าของ component ต่างๆ ให้ถูกต้อง และเหมาะสมกับข้อมูลที่จะแสดง
    - ส่วนที่2 ทำการแสดงค่า caption บนแกนตั้ง
    - ส่วนที่3 ทำการแสดงค่า caption บนแกนนอน
    - ส่วนที่4 ทำการแสดงค่า value บนพิกัด row และ column

17. Public Sub GenColumn ( )
    - ทำหน้าที่สร้างส่วนของ MDX statement ในส่วนของแกนตั้ง แล้วเก็บไว้ที่ตัวแปร membercol
18. Public Sub GenRow ( )
    - ทำหน้าที่สร้างส่วนของ MDX statement ในส่วนของแกนตั้ง แล้วเก็บไว้ที่ตัวแปร memberrow
19. Public Sub GenWhere ( )
    - ทำหน้าที่สร้างส่วนของ MDX statement ในส่วนของแกน slicer แล้วเก็บไว้ที่ตัวแปร memberwhere และนำตัวแปรนั้นมาเก็บไว้ที่ ListMem ซึ่งเป็น ListBox ที่ใช้เก็บ MDX ในส่วนของ where
20. Public Sub FillButton ( )
    - ทำหน้าที่นำเซลล์บางเซลล์ของ MSHFlexGrid ทั้งสองมาใช้เป็นส่วนที่แสดงชื่อของ level ที่แสดงในขณะนั้น โดยจะทำการใส่สีเซลล์นั้นให้เป็นสีฟ้าเพื่อให้แตกต่างจากเซลล์อื่น
21. Public Function DrillDownRow (levelBefore As String)
    - ทำการตรวจสอบว่า level ถัดไปของไดเมนชันที่เลือกในแกนนอนนั้นมีชื่อว่าอะไร แล้วนำมาเก็บไว้ใน TextBox แล้วทำการเรียก Procedure CreateGridDrillRow
    - ในกรณีที่ไม่มี level ถัดไป ก็จะขึ้นกรอบข้อความว่า " Last Level "
22. Public Function DrillDownCol (levelBefore As String)
    - ทำการตรวจสอบว่า level ถัดไปของไดเมนชันที่เลือกในแกนตั้งนั้นมีชื่อว่าอะไร แล้วนำมาเก็บไว้ใน TextBox แล้วทำการเรียก Procedure CreateGridDrillCol
    - ในกรณีที่ไม่มี level ถัดไป ก็จะขึ้นกรอบข้อความว่า " Last Level "
23. Public Sub CreateListDrillRow ( )
    - ทำการสร้าง MDX statement ขึ้นมาเมื่อมีการ drilldown ในแกนนอน เพื่อแทน MDX statement เดิม แล้วนำมาเก็บไว้ที่ TxtQuery ซึ่งเป็น TextBox อันหนึ่ง
    - ทำการเรียก Procedure Command1_Click เพื่อสร้าง MDX statement ที่สมบูรณ์พร้อมที่จะส่งไปยัง OLAP Server
    - ทำการเรียก Procedure FillGrid เพื่อทำการส่ง MDX statement ไปยัง OLAP Server และเรียก Procedure FillButton เพื่อทำการใส่ชื่อของ level ลงใน grid
24. Public Sub CreateListDrillCol ( )
    - ทำการสร้าง MDX statement ขึ้นมาเมื่อมีการ drilldown ในแกนตั้ง เพื่อแทน MDX statement เดิม แล้วนำมาเก็บไว้ที่ TxtQuery ซึ่งเป็น TextBox อันหนึ่ง
    - ทำการเรียก Procedure Command1_Click เพื่อสร้าง MDX statement ที่สมบูรณ์พร้อมที่จะส่งไปยัง OLAP Server
    - ทำการเรียก Procedure FillGrid เพื่อทำการส่ง MDX statement ไปยัง OLAP Server และเรียก Procedure FillButton เพื่อทำการใส่ชื่อของ level ลงใน grid

25. Public Function DrillUpRow (levelBefore As String)
    - ทำการตรวจสอบว่า level ก่อนหน้าของ dimension ที่เลือกในนอนนั้นมีชื่อว่าอะไร แล้วนำมาเก็บไว้ใน TextBox แล้วทำการเรียก Procedure CreateGridDrillUpRow
    - ในกรณีที่ไม่มี level ก่อนหน้า ก็จะขึ้นกรอบข้อความว่า " First Level "
26. Public Function DrillUpCol (levelBefore As String)
    - ทำการตรวจสอบว่า level ก่อนหน้าของไดเมนชันที่เลือกในตั้ง นั้นมีชื่อว่าอะไร แล้วนำมาเก็บไว้ใน TextBox แล้วทำการเรียก Procedure CreateGridDrillUpCol
    - ในกรณีที่ไม่มี level ก่อนหน้า ก็จะขึ้นกรอบข้อความว่า " First Level "
27. Public Sub CreateGridDrillUpCol ( )
    - ทำการสร้าง MDX statement ขึ้นมาเมื่อมีการ drillup ในแกนตั้ง เพื่อแทน MDX statement เดิม แล้วนำมาเก็บไว้ที่ TxtQuery ซึ่งเป็น TextBox อันหนึ่ง
    - ทำการเรียก Procedure Command1_Click เพื่อสร้าง MDX statement ที่สมบูรณ์พร้อมที่จะส่งไปยัง OLAP Server
    - ทำการเรียก Procedure FillGrid เพื่อทำการส่ง MDX statement ไปยัง OLAP Server และเรียก Procedure FillButton เพื่อทำการใส่ชื่อของ level ลงใน grid
28. Public Sub CreateGridDrillUpRow ( )
    - ทำการสร้าง MDX statement ขึ้นมาเมื่อมีการ drillup ในแกนนอน เพื่อแทน MDX statement เดิม แล้วนำมาเก็บไว้ที่ TxtQuery ซึ่งเป็น TextBox อันหนึ่ง
    - ทำการเรียก Procedure Command1_Click เพื่อสร้าง MDX statement ที่สมบูรณ์พร้อมที่จะส่งไปยัง OLAP Server
    - ทำการเรียก Procedure FillGrid เพื่อทำการส่ง MDX statement ไปยัง OLAP Server และเรียก Procedure FillButton เพื่อทำการใส่ชื่อของ level ลงใน grid
29. Private Sub TreeMem_DblClick ( )
    - ทำการตรวจสอบว่า member ที่เลือก โดยการดับเบิ้ลคลิกที่ TreeMem นั้นมาจากไดเมนชันใด และเคยถูกเลือกหรือไม่ ในกรณีที่เคยถูกเลือก ก็จะปรากฏกรอบข้อความ " Duplicate with Row Axis " หรือ " Duplicate with Column Axis "
    - ทำการตรวจสอบว่า member ที่เลือกจากไดเมนชันนั้นมาจาก cube เดียวกันหรือไม่ ในกรณีที่มาจาก cube เดียวกัน ก็จะทำการเรียก Procedure GenWhere และทำการเพิ่ม node ของ PivotTree ภายใต้ root ชื่อ " Slicer " ด้วย
    - ทำการเรียก Procedure Command1_Click เพื่อสร้าง MDX statement ในส่วนของ Slicer ขึ้นมา
30. Public Sub CheckMeasure ( )
    - คอยทำการตรวจสอบตัววัดที่ได้ทำการเลือกเข้ามา และสร้างเป็น MDX statement ในส่วนของ MDX ขึ้นมา

- ทำการตรวจสอบว่าตัววัดทั้งหมดที่เลือกเข้ามานั้น ถูกนำมาแสดงเป็นแกนนอนหรือแกนตั้ง เพื่อที่จะได้เก็บตำแหน่งได้ถูกต้อง

บรรณานุกรม

หนังสืออ้างอิง

[1] Sakhr Youness : " Data Warehousing with SQL Server 7.0 and OLAP Services ", Wrox Press, 2000.

[2] William C. Amo : " Microsoft SQL Server OLAP Developer's Guide ", M&T Books, 2000.

[3] สมพร จิวรสกุล : " คู่มือการติดตั้งและใช้งาน SQL Server 7.0 ฉบับสมบูรณ์ ", Info Press, 2000

เว็บไซต์อ้างอิง

[1] http://www.wrox.com

[2] http://www.idgbooks.com

[3] http://www.olapatwork.com

[4] http://www.microsoft.com