การพัฒนาระบบฐานข้อมูลเชิงวัตถุ

DEVELOPMENT OF OBJECT - ORIENTED DATABASE

โดย

นาย ไพศาล ขวัญเมือง

นาย ภราดร สมัครพันธ์

อาจารย์ที่ปรึกษา

ผศ.ดร. เอื้อน ปิ่นเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบฐานข้อมูลเชิงวัตถุ

ผู้จัดทำ

1. นาย ไพศาล ขวัญเมือง รหัสประจำตัว 39013246
2. นาย ภราดร สมัครพันธ์ รหัสประจำตัว 39013247

อาจารย์ที่ปรึกษา

(ผศ.ดร. เอื้อน ปิ่นเงิน)

การพัฒนาระบบฐานข้อมูลเชิงวัตถุ

นาย ไพศาล ขวัญเมือง
นาย ภราดร สมัครพันธ์

อาจารย์ที่ปรึกษา
ผศ.ดร.เอื้อน ปิ่นเงิน
ปีการศึกษา 2541

บทคัดย่อ

ในยุคปัจจุบัน การแข่งขันกันในด้านธุรกิจมีความรุนแรงมาก ดังนั้นการดำเนินงานขององค์กรต่างๆ จึงจำเป็นต้องมีข้อมูลที่ถูกต้องทันสมัยและรวดเร็ว โดยอาศัยเทคโนโลยีทางด้านระบบฐานข้อมูลและทางด้านระบบเครือข่ายคอมพิวเตอร์ที่มีการพัฒนาไปมาก ทั้งในแง่ของประสิทธิภาพการทำงานที่รวดเร็วขึ้น และในแง่ของความสามารถที่มีมากขึ้นด้วย ในปัจจุบันได้มีการมุ่งเน้นประสิทธิภาพของงานในด้านของมัลติมิเดียในการนำเสนอข้อมูลข่าวสารในเชิงธุรกิจ ทำให้มีความได้เปรียบทางด้านการค้า เช่น การโฆษณาขายสินค้าในระบบเครือข่ายอินเตอร์เน็ทโดยลูกค้าสามารถดูภาพลักษณะของสินค้าได้ เป็นต้น

อย่างไรก็ตาม ระบบของฐานข้อมูลในปัจจุบันก็ได้มีการพัฒนาเพื่อรองรับการพัฒนาของธุรกิจเช่นกัน วิทยานิพนธ์ฉบับนี้จึงเป็นอีกแนวทางหนึ่งที่แสดงให้เห็นถึงการสร้างระบบฐานข้อมูลแบบเชิงวัตถุโดยอาศัยทฤษฎีของโปรแกรมเชิงวัตถุเข้าช่วย และยังเปรียบเทียบให้เห็นถึงความแตกต่างของระบบฐานข้อมูลเชิงสัมพันธ์กับระบบฐานข้อมูลเชิงวัตถุอีกด้วย

DEVELOPMENT OF OBJECT–ORIENTED DATABASE

Mr. Paisarn Kwanmuang

Mr. Paradon Samakpun

Thesis Advisor

Asst.Prof.Dr.Ouen Pin - ngern

1998

ABSTRACT

Now a day, business competition is very violent and critical. Therefore, a number of organizations do need to have up – date , valid and rapidly data which provides by means of database technology and modern computer network system which is rapidly developed in both efficiency of rapid operation and much more capacity. Business information and data that can be presented efficiently by means of multimedia is now emphasized. This means make more advantage in trade, for example, products sale advertising in Internet, which its consumers can see pictures of various kinds of products they to sell.

However, database system is now developed for business development as well. This thesis is an alternative means show object database system by theory of object programming. Furthermore, this means can also compare and show different of relational database system and object - oriented database system.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและความร่วมมือ จากท่านผู้มีพระคุณทั้งหลายต่อไปนี้

- ขอขอบพระคุณ ท่านอาจารย์ ดร.เอื้อน ปิ่นเงิน ที่ได้ให้ความรู้และคำชี้แนะในการค้นคว้าและให้คำปรึกษาในด้านต่าง ๆ เป็นอย่างดี
- ขอขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้การศึกษาและเลี้ยงดูให้เป็นคนดีของสังคม
- ขอขอบพระคุณ คุณอัญชุลี เวกชาลิกานน ที่ช่วยในการพิมพ์งาน
- ขอขอบคุณ นาย ไพศาล ขวัญเมือง และ นาย ภราดร สมัครพันธ์ ที่ช่วยกันศึกษาค้นคว้าและจัดทำวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงด้วยดี
- สุดท้ายขอขอบคุณเพื่อน ๆ ที่ช่วยบอกแหล่งในการค้นหาข้อมูลและแลกเปลี่ยนความรู้กันมา ณ. ที่นี้ด้วย

สารบัญ

สารบัญรูปภาพ

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของงานวิจัย

อนาคตของการพัฒนาฐานข้อมูลจะต้องใช้เทคโนโลยีฐานข้อมูลที่สามารถรองรับข้อมูลประเภท MultiMedia Information ได้ด้วยข้อจำกัดของ Technology ที่มีอยู่ทำให้ต้องมีรูปแบบของการจัดการข้อมูลระบบใหม่จากผลงานวิจัยในช่วง 5-10 ปีที่ผ่านมาเป็นที่ประจักษ์ว่า Technology ฐานข้อมูลเชิงวัตถุเป็นทางออกที่เหมาะสมและจะยังคงอยู่อีกนาน การทำโครงงานเรื่องนี้จึงเน้นการทดสอบและพัฒนา Application โดยใช้ Object-Oriented Technology กับระบบฐานข้อมูล Object-Oriented Database และทำการเปรียบเทียบกับระบบฐานข้อมูลแบบ Relational Database

ระบบฐานข้อมูลแบบ Object –Oriented Database System เกิดจากแนวความคิดที่ใช้โปรแกรมแบบออบเจ็กต์(Object-OrientedProgram) หรือ OOP มาสร้างเป็นฐานข้อมูล เช่นC++,ADA,SMALLTALK เป็นต้น ระบบนี้เป็นระบบที่ให้ความสนใจกับออบเจ็กต์ (Object) ซึ่งเป็นข้อมูลที่ถูกสร้างขึ้นโดยมีคลาส (Class)เป็นตัวกำหนดรายละเอียดของข้อมูล และขั้นตอนการทำงานต่าง ๆ ของข้อมูลไว้ด้วยกัน ระบบจัดการฐานข้อมูลที่ใช้แนวความคิดของเชิงวัตถุที่ใช้อยู่ในปัจจุบันเช่นGEMSTONE/OPAL,VBASE,ORION หรือ Jasmine ที่นำมาเป็นทูลในการทำเป็นต้น ซึ่งระบบนี้จะเป็นระบบที่แพร่หลายเป็นอย่างมากในอนาคต

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษารูปแบบของ Object-Oriented Programmingและศึกษาหลักการของแบบจำลองเชิงวัตถุ(Object – Oriented Model)
2. ศึกษา Object- Oriented Database Management System (OODBMS)ซึ่งมีคุณสมบัติเป็น Object –Oriented Model ชื่อ JASMINE ว่ามีคุณสมบัติตามทฤษฎีและสามารถนำมาพัฒนาใช้กับฐานข้อมูลที่สลับซับซ้อนได้อย่างไร
3. ออกแบบโครงสร้าง Application ที่จะใช้กับระบบฐานข้อมูล (Billing System)
4. เขียน Application ใช้กับฐานข้อมูลแบบ RDBMS เปรียบเทียบกับ OODBMS
5. ศึกษาเพื่อเปรียบเทียบความแตกต่างของการจัดการข้อมูลระหว่าง RDBMS และ OODBMS

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้สร้างอยู่บนระบบฐานข้อมูลเชิงวัตถุที่อนุมานขึ้นมาโดยจะมุ่งเน้นการศึกษาการประยุกต์ใช้งานตามทฤษฎีเชิงวัตถุที่ระบบฐานข้อมูลเชิงวัตถุโดยได้นำมาใช้เพื่อเปรียบเทียบความแตกต่างกับระบบฐานข้อมูลเชิงสัมพันธ์ในด้านของการสร้างโปรแกรมประยุกต์ใช้งานกับระบบฐานข้อมูลทั้งสองซึ่งจะมุ่งเน้นที่ระบบฐานข้อมูลเชิงวัตถุจริงในเชิงการค้าว่ามีการนำทฤษฎีมาประยุกต์ใช้จริงเช่นไร

1.4 วิธีการดำเนินงาน

สำหรับงานวิจัยในโครงงานนี้จะเริ่มต้นด้วยการศึกษาถึงทฤษฎีพื้นฐานต่างๆของทฤษฎีเชิงวัตถุและการโปรแกรมเชิงวัตถุเพื่อนำมาประยุกต์ใช้กับระบบฐานข้อมูลเชิงวัตถุ โดยมีเนื้อหากล่าวตามบทที่ 2 โดยมีเนื้อหาสำคัญในการศึกษาคือ ระบบฐานข้อมูลทั้งในเชิงสัมพันธ์และเชิงวัตถุ โดยอ้างตามเนื้อหาของบทที่ 3 และ บทที่ 4 จะกล่าวถึง การประยุกต์ทฤษฎีเชิงวัตถุกับระบบฐานข้อมูลเชิงวัตถุ ซึ่งจะสัมพันธ์กับบทที่ 5 คือการสร้างโปรแกรมประยุกต์กับระบบฐานข้อมูลเชิงวัตถุ

บทที่ 6 จะเป็นการทดสอบและสรุปผลการทำงาน ส่วนเนื้อหาในบทสุดท้ายเป็นแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม

บทที่ 2
แนวความคิดเชิงวัตถุ

2.1 คุณสมบัติของแนวความคิดแบบเชิงวัตถุ

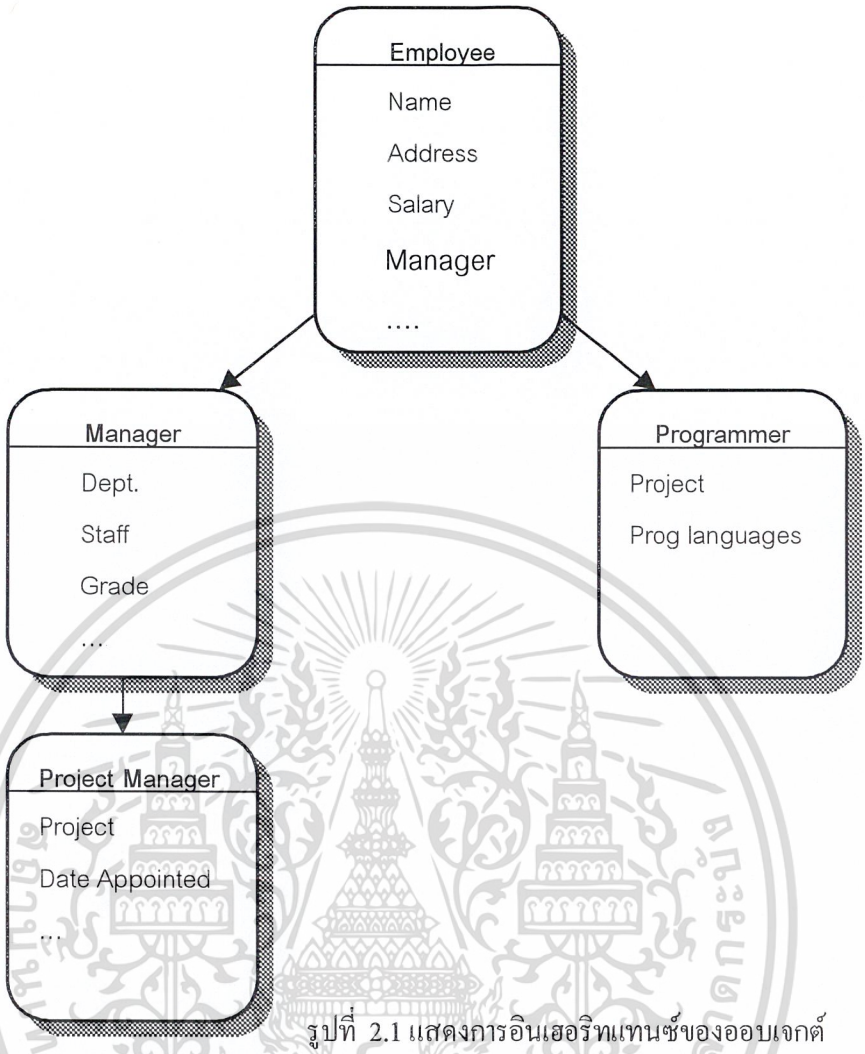
แนวความคิดแบบ Object-Oriented เป็นแนวคิดที่มีคุณสมบัติแตกต่างจากแนวความคิดเดิมโดยเฉพาะแนวคิดในการพัฒนาโปรแกรมแบบเดิมที่เป็นโปรแกรมโครงสร้าง(Structure Programming)ซึ่งส่วนที่เป็นโปรแกรมจะแยกออกจากกัน(Separates data and program)ในขณะที่แนวความคิดของ Object-Oriented จะมีออบเจ็กต์เป็นที่รวมของข้อมูลและวิธีการ(Object Combine Data and Methods)โดยมีคลาสเป็นตัวกำหนดคุณสมบัติของออบเจ็กต์ ซึ่งเป็นเครื่องมือในการพัฒนาต้นแบบ(Prototyping Development) รวมถึงการนำออบเจ็กต์กลับมาใช้ใหม่ได้อีก คุณสมบัติของแนวความคิดแบบ Object-Oriented พอสรุปได้ดังนี้

1. การกำหนดรายละเอียดขั้นตอนการประมวลผลว่าควรมีขั้นตอนอะไรบ้าง เช่น มีการกำหนดข้อมูลที่ใช้ว่ามีการปรับปรุงหรือเปลี่ยนแปลงข้อมูลอย่างไร มีการคำนวนหรือฟังก์ชั่นที่เกี่ยวข้องอะไร การแสดงผลของข้อมูลจะแสดงอย่างไร ขั้นตอนการทำงานที่เชื่อมต่อกับผู้ใช้การแสดงผลออกมานี้ เรียกว่า Data Abstraction เพื่อใช้ในการกำหนดคำนิยามของคลาสได้ ลักษณะการกำหนดรายละเอียดเหล่านี้จะเป็นลักษณะจากล่างขึ้นบน (Bottom-up Design)
2. Object จะถูกกำหนดว่ามีสมาชิกหรือรายละเอียดของข้อมูล(Attributes) และฟังก์ชั่นหรือขั้นตอนที่เกี่ยวข้องอะไรบ้าง(Methods) โดยจะกำหนดคำนิยามในคลาส
3. การนิยามคลาสต่าง ๆ สามารถนำมาใช้ใหม่ได้ถ้าต้องการซึ่งเป็นวิธีการสืบทอดคลาส (Class Inheriance)จากโปรแกรมเดิมมายังโปรแกรมใหม่ เพื่อทำการปรับปรุงเพิ่มเติมแทนที่จะต้องเสียเวลาเขียนทั้งโปรแกรม
4. โครงสร้างของโปรแกรม Object-Oriented จะขึ้นอยู่กับข่าวสาร(Message)ที่ส่งไปยังออบเจ็กต์เพื่อให้ทำงานตามขั้นตอนที่ระบุไว้

2.2 คุณสมบัติที่ใช้แสดงให้เห็นถึงการสื่อสารข้อมูลของออบเจ็กต์

- **Object** ออบเจกต์เปรียบเหมือนรายละเอียดที่ใช้ในโปรแกรมทั่วไปที่มีการระบุถึงประเภทข้อมูลที่กำหนดให้ตัวแปรรวมถึงค่าของตัวแปรนั้น ๆ ด้วย หรืออีกนัยหนึ่งออบเจ็กต์คือ ตัวแปรของคลาส โดยมีรายละเอียดของข้อมูล (Attributes) และขั้นตอนการทำงานหรือฟังก์ชั่นไว้ในออบเจ็กต์นั้น(Methods)

- **Inheritance** เป็นการสืบทอดคุณสมบัติคลาสที่มีอยู่ในปัจจุบัน(Base class) มาเป็นคลาสใหม่ โดยที่คลาสใหม่ที่สืบทอดจะรับคุณสมบัติของรายละเอียดและวิธีการต่าง ๆจากคลาสเก่า การสืบทอดจะช่วยให้การพัฒนาโปรแกรมใหม่ทำได้เร็วยิ่งขึ้น การสืบทอดสามารถจะกระทำได้ดังรูปที่ 2.1



รูปที่ 2.1 แสดงการอินเฮอริทแทนซ์ของออบเจกต์

- **Encapsulation** เป็นคุณลักษณะของรายละเอียดของข้อมูล และวิธีการของออบเจ็กต์ที่ระบุให้ทราบถึงการที่จะได้มาซึ่งผลลัพธ์ โดยในระบบจะมีตัวประสานของออบเจ็กต์เป็นตัวจัดการให้ว่าต้องทำอะไรบ้าง

- **Association** เป็นการเชื่อมกันทางด้าน Physical หรือ Conceptual ของแต่ละออบเจ็กต์เป็นกลุ่ม การจัดกลุ่มของออบเจ็กต์ที่มีความสัมพันธ์กันมากกว่าสองออบเจ็กต์ขึ้นไปก็จะเป็น กรุ๊ป(Group)ที่มีโครงสร้างและความสัมพันธ์ร่วมกัน ในแต่ละกรุ๊ปสามารถเชื่อมกับกรุ๊ปอื่น ๆ ที่มีคุณสมบัติในลักษณะเดียวกันได้

- **Message passing** ออบเจ็กต์ต่าง ๆ ในแต่ลเคลาสจะถูกสั่งให้ทำงานเมื่อมีการส่งข่าวจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่งโดยตัวประสานหรือตัวเชื่อมข่าวสาร (Message connection) จะแสดงเส้นทางการติดต่อสื่อสารระหว่างคลาส

- **Polymorphism** เป็นลักษณะเด่นของแนวคิดแบบออบเจ็กต์โอเรียนเท็ด คือ ความสามารถของฟังก์ชั่นหรือโอเปอเรเตอร์ต่าง ๆ กันใช้ชื่อเดียวกันได้โดยโปรแกรม Object-Oriented จะแยกความแตกต่างของการใช้ชื่อที่เหมือนกันได้ด้วยจำนวนและชนิดของพารามิเตอร์

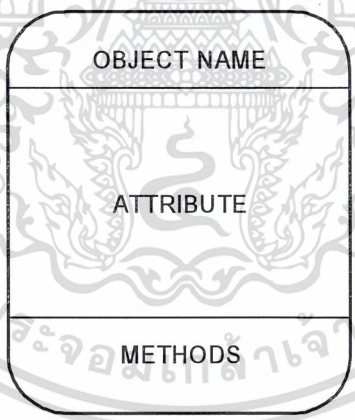
เรียกว่า 'One interface, multiple methods' เพื่อให้ใช้งานได้ง่าย โพลิมอร์ฟิซึม แบ่งออกได้เป็น 2 ลักษณะ คือ

1. ช่วงการคอมไพล์โปรแกรม (Compile time) การตรวจสอบชนิดออบเจ็กต์ของฟังก์ชันที่ถูกเรียกมาทำงาน รวมทั้งเตรียมข้อมูลบางอย่างที่จำเป็นจะทำขณะคอมไพล์โปรแกรมการตัดสินใจใช้ฟังก์ใดขึ้นอยู่กับคุณสมบัติของพารามิเตอร์ของฟังก์ชันนั้น
2. ช่วงการรันโปรแกรม (Run time) การตรวจสอบนั้นจะทำขณะที่ โปรแกรมกำลังทำงานจะทำการตัดสินใจเลือกฟังก์ชันใดฟังก์ชันหนึ่งที่ มีชื่อซ้ำกันในแต่ละคลาสมาใช้งาน

- **Object Identity** การระบุคุณลักษณะเฉพาะของแต่ละออบเจ็กต์จะเป็นอิสระไม่ข้องเกี่ยวกัน โดยจุดประสงค์ในการระบุนั้นสามารถอ้างอิงถึงออบเจ็กต์อื่นได้ เมื่อมีการรวบรวมหลาย Object Identity แล้วก็หาความสัมพันธ์และสามารถจัดทำขึ้นเป็นโปรแกรมได้

2.3 การสร้างโมเดลของฐานข้อมูลแบบ Object

การสร้างนี้มีวัตถุประสงค์เพื่อช่วยให้การออกแบบฐานข้อมูลในระดับแนวคิดเพื่อนำไปใช้ในการสร้างฐานข้อมูลดังรูปที่ 2.2 โดยมีหลักการคล้ายกับการสร้างโมเดล E-R ในการสร้างโมเดลแบบนี้พอสรุปขั้นตอนได้ดังนี้



รูปที่ 2.2 แสดงโครงสร้างของออบเจ็กต์

1. กำหนดว่ามีออบเจ็กต์อะไร(Object Name) และออบเจ็กต์นั้น ๆ มีรายละเอียดอะไรบ้าง (Attribute)รวมถึงวิธีการ(Methods)
2. กำหนดความสัมพันธ์ระหว่างออบเจ็กต์ว่ามีความสัมพันธ์อะไรบ้าง
3. การเชื่อมโยงการส่งข่าวสารว่ามีอะไรบ้าง

บทที่ 3

ทฤษฎีระบบฐานข้อมูล

3.1 การจำแนกระบบฐานข้อมูลตามคุณลักษณะ

รูปแบบของระบบฐานข้อมูลแบ่งเป็น 4 ชนิดคือ

1. Relational DataBase

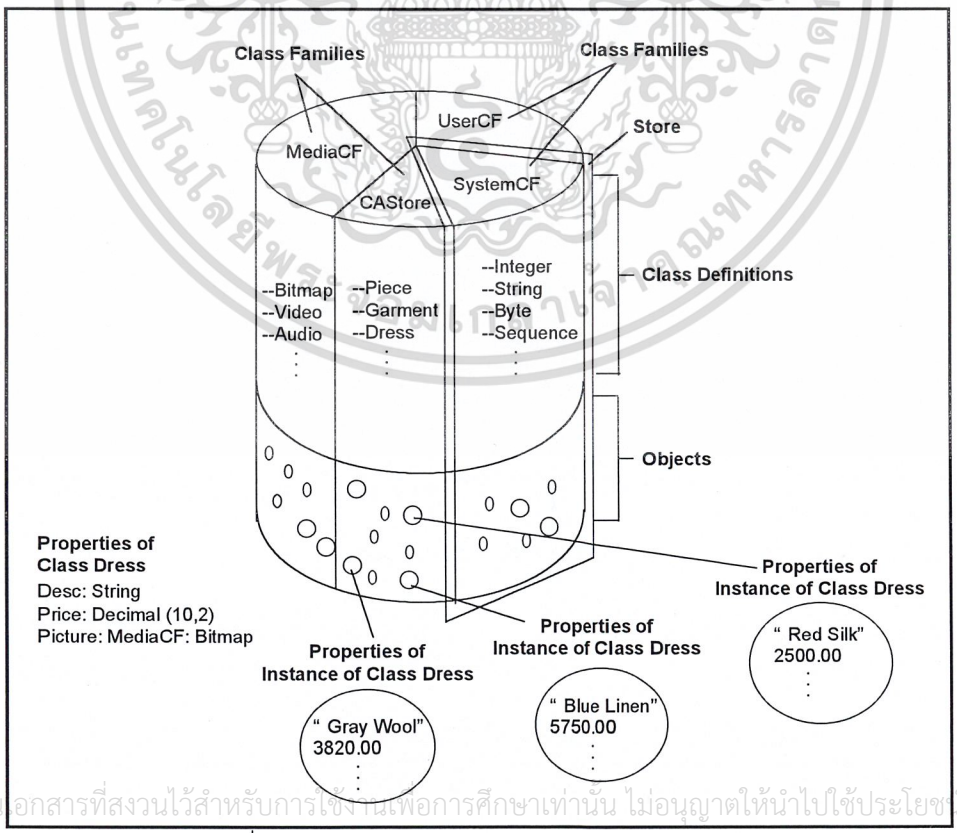
   เป็นระบบฐานข้อมูลที่ออกแบบเพื่อทำการเก็บข้อมูลที่มีรูปแบบไม่ซับซ้อน แต่ว่ามีการเชื่อมโยงความสัมพันธ์เป็น element data ซึ่งมักจะใช้ในระบบธุรกิจที่มีลักษณะข้อมูลเป็น *text* หรือ *หมายเลข*

2. Object-relational (Hybrid) Database

   เป็นการเพิ่มคุณลักษณะของ Object ลงไปใน relational technology ตัวอย่างของระบบฐานข้อมูลจำพวกนี้คือ Infomix Universal Server หรือ Oracle 8 Universal Data Server แต่ว่าฐานข้อมูลเหล่านี้จะไม่ Support คุณลักษณะพื้นฐานของ Object โดยตรงเช่น *การ Inheritance* แต่จะเป็นการโยนภาระนี้ให้กับ application code แทน

3. Extended-relational(ER) systems

   ฐานข้อมูลชนิดนี้เป็นแบบที่ใช้การเพิ่ม *"user-defined data types (UDT)"* ให้กับโครงสร้างแบบ relation เท่านั้นเองตัวอย่างฐานข้อมูลจำพวกนี้คือ DB2 ของ IBM



รูปที่ 3.1 แสดงโมเดลการเก็บข้อมูลของฐานข้อมูลเชิงวัตถุ

4. Object-Oriented-DataBase

มีลักษณะการเก็บข้อมูลและการเรียกใช้ข้อมูลที่มีรูปแบบแตกต่างจากระบบข้างต้นคือจะเป็นการมอง data ที่ store ในรูปของ *Object* โดยจะทำการ Classified โดย *class type* และจัดกลุ่มในรูปของ *class family hierarchy* ดังรูปด้านบน

จากรูปที่ 3.1 แสดงการเก็บข้อมูลของฐานข้อมูลเชิงวัตถุ ที่สามารถอธิบายในแต่ละส่วนได้ดังนี้

- **Class families** คือ collection ของ class ที่มีลักษณะเดียวกันซึ่ง classes ใน class family จะอยู่ในรูปของ tree หรือ set ของ tree

- **Classes** จะหมายถึง collection ของ object ที่มี feature พื้นฐานเดียวกัน(common-feature) ตัวอย่างเช่น class ของ dress จะสามารถประกอบด้วย dress objects บวก common feature (Description,ราคา,รูปของชุด)

- **Objects/Instances** คือ ส่วนของ dataใน class เช่น แต่ละ Dress ใน Dress class คือ Object(หรือ instance)โดยแต่ละ object จะประกอบด้วย information(Properties) เช่น ราคา รายละเอียด รูปภาพ เป็นต้น

- **Method** เป็น functionsที่จะยอมให้เราสามารถ

  - calculate derived data(เช่น ราคาและภาษี เป็นต้น)

  - Maintain redundant data

  - Access Information ที่อยู่นอกระบบ database

  - ซ่อนรายละเอียดของ Information ภายใน-

- **Subclass** คือ การ inherit properties และ methods ของ class (superclass)เช่น Class Manager นั้น inherits properties และ method ของ Class Employee นั่นก็หมายความว่า Manager จะมี properties และ method ที่มีรูปแบบเดียวกับ Employee (เช่น Name,DOB,Phone#)และเราสามารถเพิ่ม properties(หรือ method) ใหม่ๆให้กับ class Manager ก็ได้

- ถ้า class นั้นๆมี Superclass มากกว่าหนึ่ง superclass เราจะเรียกว่า multiple inheritance

- เมื่อเราพิจารณาที่ class families ที่ CAStore จะพบว่าประกอบด้วย class definition เช่น piece(ผ้าเป็นชิ้น) , garment(เสื้อผ้า) , Dress(กระโปรง) เห็นได้ว่าจะประกอบด้วย Object พื้นฐานเดียวกันคือเป็นผ้า และนอกจากนี้ properties ของ dress คือ desc(String),Price(Decimal[10,2]),Picture (MediaCF:Bitmap) ถูกแบ่งข้อมูลเป็นแต่ละส่วนชัดเจน

3.2 ระบบจัดการฐานข้อมูลเชิงวัตถุที่แท้จริง (Pure OODBMS)

ถ้าเราได้ตัวจัดการข้อมูลมาสักตัวหนึ่งที่เป็น ODBMS เราอาจสงสัยว่า เทคโนโลยี ODBMS ต่างจาก DBMS อย่างไร ที่ผ่านมาเราจะเห็นได้ว่าได้มีการโต้แย้งหรือถกเถียงกันถึงองค์ประกอบของ ODBMS ผู้ขายบางคนถึงกับเคลม(Claim)ว่าเป็นผลิตภัณฑ์ที่ไม่ได้มาตราฐาน และเมื่อต่อมา เทคโนโลยีทางด้าน Object – Oriented ได้มีเข้ามามีบทบาทมากขึ้น จึงเหตุผลที่ทำให้เกิดความต้องการในการใช้เทคโนโลยี

ODBMS มากขึ้นตามลำดับ ดังนั้นเมื่อพยายามกำหนดนิยามของการเปรียบเทียบของทั้งสองเทคโนโลยี แล้วคิดว่า ODBMS น่าจะมีประสิทธิภาพที่ดีกว่า

จากการนิยามในเบื้องต้นสามารถแบ่งตัวจัดการฐานข้อมูล(Database Management system) ออกเป็น 2 ส่วน ส่วนแรก คือตัวจัดการฐานข้อมูลทั่วไป (DBMS) ส่วนที่สองคือส่วนที่ถูกพัฒนามาจากส่วนแรกที่เรียกว่า ODBMS ซึ่งทั้งสองส่วนจะมีคุณลักษณะที่สามารถแยกออกได้เป็นดังนี้

DBMS จะมีคุณสมบัติการทำงาน

Persistence DBMS จะสามารถเก็บข้อมูลให้อยู่ตลอดเวลา และสามารถเข้าถึงข้อมูลได้ รวมทั้งยังสนับสนุนการสร้าง การกู้ข้อมูล การแก้ไขของข้อมูล โดยเฉพาะอย่างยิ่งถ้าข้อมูลนั้นเก็บอยู่บน Hard disk

Concurrency DBMS จะบริหารการเข้าถึงข้อมูลของ Multiple Clients โดยเฉพาะการเข้าถึงข้อมูลที่ผ่านทางระบบเครือข่าย(network system) จะสามารถทำการ Locking Policy เพื่อควบคุมการเข้าถึง (Access)ได้

Transaction DBMS จะสนับสนุนงานที่เกิดขึ้นในสภาวะปกติและจะเก็บข้อมูลเหล่านั้นไว้ ในหนึ่งทรานแซคชัน(Transaction)อาจจะมีข้อมูลหรือการเปลี่ยนแปลงของข้อมูลเกิดขึ้นได้หรืออาจไม่มีเลยก็ได้ ทั้งนี้ขึ้นอยู่กับว่าทรานแซคชันล่าสุด(Least Transaction)นั้นมีการกระทำอย่างไร เช่น Start,Commit,Abort เป็นต้น

Query DBMS จะสนับสนุนการรายงานผลของข้อมูลเป็นในลักษณะของ Queyry โดยผู้ใช้จะสามารถทำคิวรี่(Query)ผลลัพธ์ตามที่ต้องการ โดยใช้ค่าของโอเปอเรเตอร์ต่าง ๆ เช่น OR หรือ AND เป็นต้น นอกจากนี้ยังสนับสนุนคิวรี่ที่มีหลาย ๆ ชั้นได้อีกด้วย

Recovery DBMS จะสนับสนุนการกู้ข้อมูลที่เกิดจากการที่ระบบใช้การไม่ได้ซึ่งส่วนนี้เป็นความสามารถของ DBMS นอกจากนี้ยังรวมไปถึงความสามารถในการทำ Backup File เพื่อให้สามารถรีสตอร์(Restore) ข้อมูลที่ Backup ไว้นำกลับมาใช้ได้หลังจากระบบไม่สามารถใช้การได้

อย่างไรก็ตาม ยังมีอีกหลายคุณสมบัติที่มีอยู่ใน DBMS เช่น Database distribution,Data replication,and fault tolerance เป็นต้น

ODBMS จะมีคุณสมบัติที่ครอบคุมการทำงานของ DBMS ทั้งหมดดังนี้

Object – Oriented Programming Language(OOPL) Interface ODBMS จะสนับสนุนการสร้างหรือการกระทำการใด ๆ กับออบเจกต์ได้โดยตรงภายใต้ OOPL

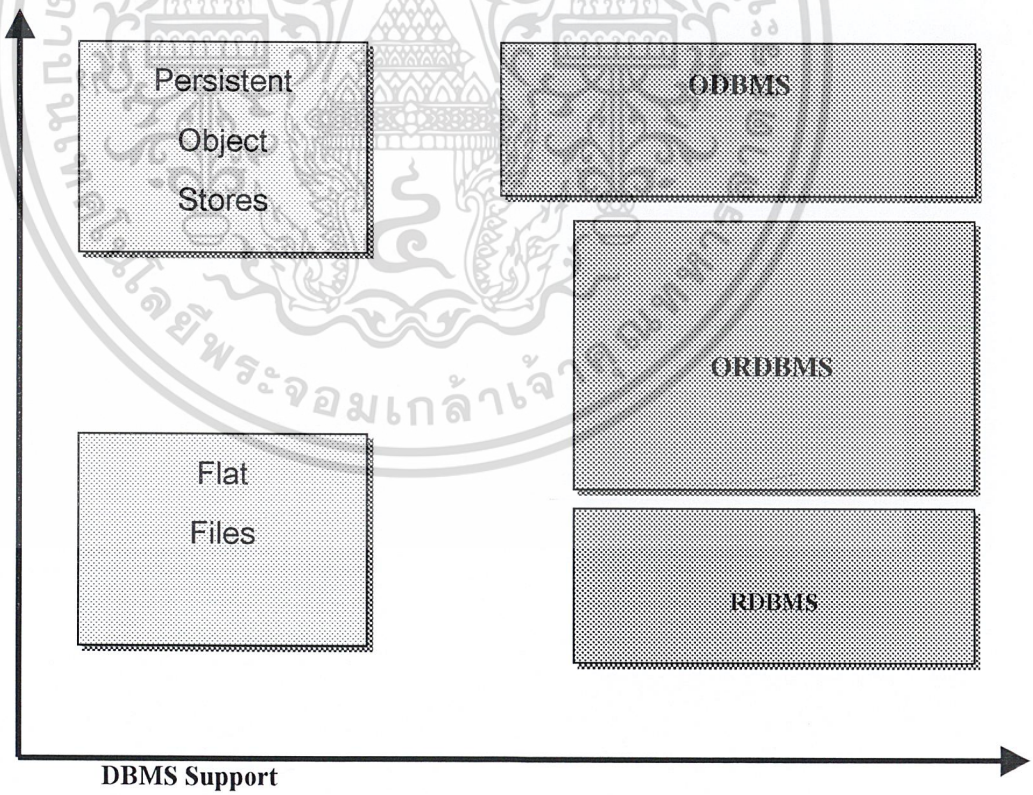
User – Defined type ODBMS จะสนับสนุนการนิยามและการกำหนดชนิดของข้อมูลได้หลากหลาย รวมทั้ง Compound Type และ Variable –Length Type ด้วย โดย ODBMS จะทำการเก็บค่าของข้อมูลที่ผู้ใช้กำหนดไว้เป็นหนึ่งส่วนของข้อมูล ต่อจากนั้นก็จะแปลงข้อมูลเหล่านั้นไปเป็นฐานข้อมูล

Object Identity ODBMS จะสนับสนุนไอเดนทิตี้(Identity) ของ Object Instance โดยไม่สนใจว่าค่าที่เป็นลักษณะเฉพาะ(Attribute)ของแต่ละ Object จะเป็นเช่นไร และแต่ละ Object ต้องมีค่าของ Object Identity เพื่อไว้ใช้อ้างอิงไปยัง Object ตัวอื่นในแอพพลิเคชัน นอกจากนี้ Ojbject Identity ยังต้องระบุไว้ให้ชัดเจนในแอพพลิเคชันด้วย

Network of Object ODBMS จะสนับการเก็บและการจัดหา Object บน Network ซึ่งออบเจกต์เหล่านั้นจะต้องมีการระบุ Object Identity ไว้อย่างชัดเจน ภาษาที่ใช้กับ Object บน Network จะต้องเป็นภาษาที่สนับสนุนการเข้าถึงค่าเฉพาะ (Object Attribute)ของ Object และจะสนับสนุนการจัดการในด้านการติดต่อของฐานข้อมูลในลักษณะของ One –to-One , One-to-many , Many-to-Many โดยใช้ Object Identifier(OID) อีกด้วย

Locality of reference optimization ODBMS จะสนับสนุนการทำให้ ระบบทำงานได้ดีที่สุด โดยนำเอาข้อดีของการจัดโครงสร้างทางด้านเครือข่าย(Network)ของ Object มาใช้งานรวมทั้งการทำ Optimization บน Cluster ของ hard disk ด้วย ความสามารถในการล๊อค(Lock)กลุ่มที่เกี่ยวข้องกันของออบเจกต์เข้าด้วยกัน และการจัดการกับ Client cache บนพื้นฐานของความสัมพันธ์กันของออบเจกต์

จะสังเกตุได้ว่า ODBMS จะมีคุณสมบัติที่นอกเหนือจาก DBMS ทั่วไป คือ จะเป็นการนำเอางานในส่วนต่าง ๆ มารวมไว้ด้วยกันเพื่อให้ง่ายต่อการใช้งาน เช่น สนับสนุนการติดต่อโดยตรงจาก OOPL และการระบุความต้องการที่ชัดเจน การมีจุดอ้างอิงของ Object ที่แน่นอน เป็นต้น การที่จะกล่าวว่า คุณสมบัติของตัวจัดการข้อมูลแบบใดจึงจะเหมาะสมกับงานที่สร้างขึ้นมา จะต้องดูว่างานนั้นต้องการคุณสมบัติของตัวจัดการข้อมูลเป็นแบบใด ดังเช่นรูปที่ 3.2 เป็นเปรียบเทียบระหว่าง DBMS(RDBMS) ,Object-Relational DBMS (ORDBMS)และ Object-Oriented DBMS (OODBMS) ที่กระทำกับฐานข้อมูลที่เป็น Object-Oriented



รูปที่ 3.2 แสดงการเปรียบเทียบของ DBMS ที่กระทำกับข้อมูลแบบ Object – Oriented

การพิจารณา ODBMS กับข้อมูลที่ซับซ้อน

Complex Datatye ข้อมูลอย่างง่าย ๆ (Simple datatype)โดยทั่วไปจะประกอบไปด้วย Interger, Floating Point และ String ส่วนข้อมูลที่ซับซ้อนจะประกอบด้วยการนำเอา Simple datatype มารวมเป็นโครงสร้างตามลำดับของ String และ Array ที่มีอยู่ในแอพพลิเคชันโปรแกรม การรวมกันของสิ่งเหล่านี้จะต้องการฟังก์ชันการทำงานที่นอกเหนือจากที่มีอยู่ใน Storage Engine จากจุดนี้เองที่ทำให้มีการนำเอา ODBMS มาช่วยให้เกิดการรวมกันของข้อมูลโดยผ่านทาง OOPL เพื่อให้ง่ายต่อการสร้างหรือการรวมกันของข้อมูล เมื่อทำการสร้างและเก็บไว้เป็นฐานข้อมูลโดยผ่านทาง ODBMS ที่เป็นตัวจัดการก็จะได้ฐานข้อมูลขนาดหนึ่งบล๊อก(Block) ที่ไม่สามารถแตกย่อยหรือกระจายให้เป็นส่วนของข้อมูลที่เล็กลงได้อีก

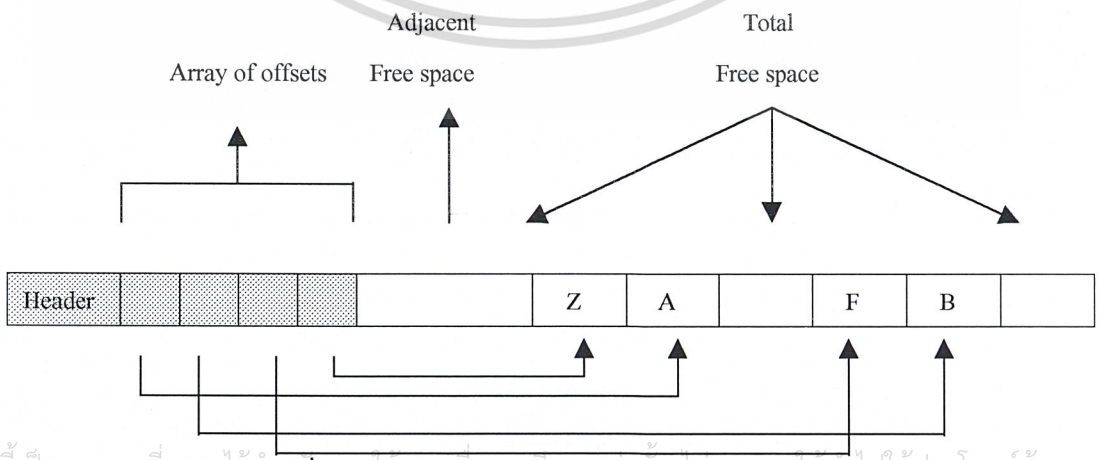
Complex Object Network การจัดการทางด้านเครือข่าย(Network)ของออบเจกต์ที่เชื่อมต่อไปยังส่วนต่าง ๆ จะมี Object Identifier (OID)เป็นตัวกำหนดทางด้าน Mechnism เพื่อให้ง่ายต่อการกำหนดในเครือข่าย (Network) ใน ODBMS โปรแกรมเมอร์จะ Declare การติดต่อกัน(Association)ของออบเจกต์ได้โดยตรง ซึ่งการติดต่อกันใน Storage Engine นั้นมีหลายชนิด โปรแกรมเมอร์ไม่จำเป็นต้องสร้างการติดต่อกัน(Associate)ของออบเจกต์ให้รองรับการติดต่อแบบ Many-to-Many เหมือน RDBMS และสิ่งที่สำคัญอีกอย่างก็คือว่า ODBMS จะมีการจัดการในการเข้าถึงข้อมูลของออบเจกต์ให้ประสิทธิภาพสูงสุดโดยผ่านทางระบบเครือข่าย(Network)

จะเห็นได้ว่า ODBMS นั้นจะส่งเสริมการใช้งานที่ง่าย ๆ และมีประสิทธิภาพสูงเมื่อนำมาใช้กับงานที่มีชนิดของข้อมูลซับซ้อนหรือออบเจกต์ที่ซับซ้อนในระบบเครือข่าย(Network)หรือสนับสนุนงานในทั้งสองกรณี ดังเช่น แอปพลิเคชันชันโปรแกรมในโรงงานอุตสาหะกรรมขนาดใหญ่ ในระบบโทรคมนาคม ระบบการเงิน ระบบการผลิต ระบบขนส่ง และระบบสาธารณะสุข เป็นต้น

การเก็บข้อมูลและอินเด็กในการอ้างอิงของ DBMS (Storage Management and Indexing Techniques)

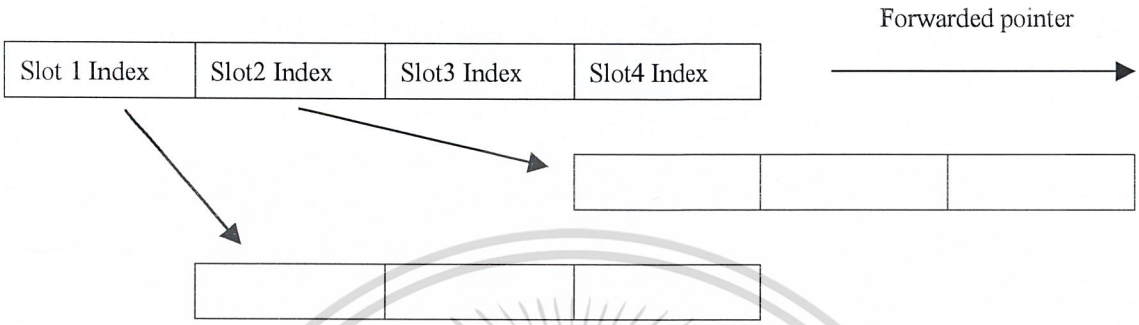
Storage Techniques สำหรับ RDBMS

Disk ถูกแบ่งออกเป็นเซทของ พาทิชั่น (partitions)ซึ่งแต่ละพาทิชั่นจะประกอบด้วยตัวเลขของ เพจ (pages) หรือ บล็อก (blocks) ส่วนสำคัญของ Disk คือเรียกว่า Header ดังรูปที่ 3.3 ซึ่งจะประกอบด้วยหมายเลขของ พาทิชั่น , address และ ขนาดของแต่ละพาทิชั่นรวมถึง log สำหรับการรีโคเวอรี่ในกรณีที่ระบบเกิดเสียหายและ segment จะบอกถึง table ที่อยู่ในแต่ละเพจแอดเดรสใด



รูปที่ 3.3 แสดงความสัมพันธ์ของการเก็บแบบเก่า (page storage)

จากรูปที่ 3.3 แสดงถึงความสัมพันธ์ของเพจซึ่งแต่ละเพจจะประกอบด้วย header สำหรับเก็บข้อมูลของเพจและอาเรย์ที่จะเป็นตัวเก็บออฟเซ็ทในระบบของ RDBMS เรคคอร์ดจะเป็นส่วนจัดการ ทัพเปิล(tuple) ของ relation ซึ่งจะเก็บติดกันในดิสถ้า attribute ของrelation มีขนาดที่จำกัดข้อมูลก็สามารถเก็บอยู่ติดกันได้ไม่ต้องใช้ช่องว่างเผื่อไว้ทำให้เราสามารถเก็บไว้ในรูปแบบของไฟล์เดียวได้แต่ถ้า attribute มีขนาดที่สามารถเปลี่ยนแปลงได้เรามักใช้การใส่ค่า ID ให้กับแต่ละ record เพื่อใช้ในการอ้างอิง



เทคนิคของการ Identifiles record มีหลักสำคัญคือ

1. ในการ access disk จะสามารถทำได้อย่างรวดเร็วเท่าๆกับการอ้าง address แต่ว่าสามารถจะเปลี่ยนขนาดความยาวของ record ได้และสามารถเก็บ,เพิ่ม,ลบหรือย้ายข้อมูลไปยัง page ใดๆ ได้ง่าย
2. สามารถทำงานได้เร็วกว่าการใช้ Logical ID ในการอ้างอิงข้อมูลซึ่งต้องการการอ้างอิง address แบบอื่นๆช่วยในการเข้าถึงข้อมูลเช่น Hash table

Storage Techniques สำหรับ Objects

ระบบของ Object-Oriented database model นั้นมีความซับซ้อนกว่า RDBMS ดังนั้นระบบการบริหารหน่วยความจำใน OODBMS จะต้องมีประสิทธิภาพเพื่อรองรับ

1. Object ที่มี complex attributes และ atomic
2. Object ที่มีค่าเป็นแบบ multi- value ได้
3. Object ที่มี attribute แตกต่างกัน
4. Object ที่มีลักษณะ “long field” attribute เช่น ข้อมูลจำพวก ภาพเคลื่อนไหว เสียง เป็นต้น

ระบบการ storage ของ OODBMS นั้นในปัจจุบันแบ่งออกเป็น 2 model

1. Direct model

Object ถูกเก็บตามรูปแบบที่ได้กำหนดไว้ใน conceptual schema และ Object ของ class เดียวกันจะเก็บในไฟล์เดียวกันโดยแต่ละ file record คือ Object instance ของ class ซึ่งการเก็บในแบบนี้ค่อนข้างมีประสิทธิภาพในการทำงาน เช่นในการ join ข้อมูลและการนำข้อมูลมาใช้ แต่ก็มีข้อเสียคือการ access สู่ set ของ attribute ของ Object ยุ่งยากโดยเฉพาะเมื่อมีการเก็บข้อมูลแบบหลายมิติมากๆ

2. normalized model

ในแบบ normalized model Object จะถูกแยกออกเป็น atomic component ซึ่งจะถูกแยกเก็บในหลายๆไฟล์ซึ่งความสัมพันธ์ของแต่ละ Object จะถูกเชื่อมโยงโดย OIDs.

เมื่อ attribute มีการปรับเปลี่ยนข้อมูลได้ลักษณะนี้จะเหมาะกับ model แบบ normalized โดยการเก็บ attribute แบบแยก Object เก็บในพื้นที่หลายๆส่วนโดยสร้างความสัมพันธ์ภายในโดย OID

เมื่อ attribute ใหม่ได้ถูกสร้างขึ้นเช่นถ้า schema ถูกแก้ไขแต่ว่า space ได้ถูกจำกัดสำหรับ attribute ของ Object ทำให้เราไม่สามารถสร้าง attribute ขึ้นมาได้ยกเว้นเราได้มีการสร้างพื้นที่เกินสำหรับส่วนนี้ไว้แล้วหรือเมื่อค่าส่วนใหญ่ใน attribute มีค่าเป็น null ค่าส่วนที่เราได้กันไว้ก็จะสูญเปล่า

ดังนั้นเราจึงได้มีการสร้าง property list เพื่อแก้ปัญหาข้างต้นซึ่งจะประกอบด้วย 3 ส่วนคือ <identifier , size , value > สำหรับแต่ละ attribute ของแต่ละ Object โดยจะไม่นำไปปะปนกับ OIDs (Size = จำนวน Byte ที่เก็บ , Value = คือค่าที่จะให้มีการเปลี่ยนแปลงได้ของ attribute

Property list นั้นมีความยืดหยุ่นในการใช้งานเราอาจนำมาใช้แก้ปัญหากรณีที่ค่าข้อมูลส่วนใหญ่เป็น null ได้โดยไม่จำเป็นต้องให้ช่องว่างเกิดขึ้นเผื่อไว้ให้แต่ property list ก็มีส่วนด้อยคือการส่งค่าหรือการเก็บข้อมูลต้องเสียเวลาในการทำงานในส่วนของ property list เพิ่มขึ้นมาและนอกจากนี้ รูปแบบขอลการส่งข้อมูลของ property list ต้องเข้ากันได้กับโปรแกรมที่จะสร้างขึ้นมาด้วย

การเก็บข้อมูลแบบ Inheritance เก็บข้อมูลตามกลุ่มของ class ที่มีความสัมพันธ์กันทั้งในแบบ single Inheritance และ Multiple Inheritance โดยจะนำหลักการของ property list มาใช้เช่นกัน

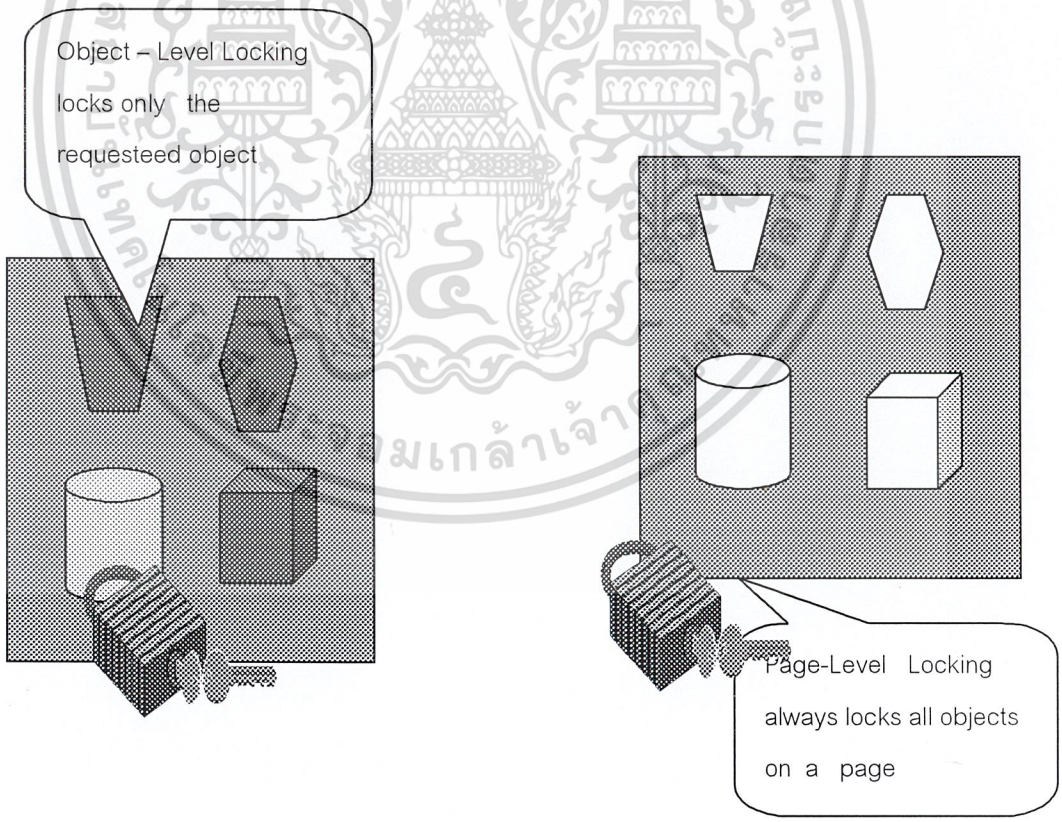
3.3 Concurrency and Lock Granularity

กุญแจสำคัญที่เป็นตัวกำหนดเปอฟอร์แมนซ์(Performance)ของระบบ ก็คือ ระดับของการทำ Concurrent access กับฐานข้อมูล และเป้าหมายที่สำคัญของฐานข้อมูลโดยทั่วไปก็คือ การใช้ Concurrent สูงสุดในขณะที่การใช้การช่วงชิงของทรัพยากรของระบบในการเข้าถึงข้อมูลต่ำสุด เป็นการเริ่มต้นที่ดีที่ในเบื้องต้นการทำการ Locking เพื่อที่จะเข้าถึงข้อมูลนั้นมีอยู่ใน RDBMS ซึ่งเป็นลักษณะของการล็อกระดับล่าง (Low-Level Locking)โดยใช้ทรัพยากรของระบบน้อยมาก ผู้นำทางด้านฐานข้อมูลที่เป็น Relational Database เช่น DB2 Oracle ก็จะมีการ Locking ในระดับล่างเช่นกัน ส่วนการทำ Locking ในงานที่เป็นออบเจกต์นั้นจะต้องมีคุณลักษณะที่สำคัญมากกว่า จากการทดสอบของ Industrial Benchmark พบว่าการทำระบบแบบRow-locking จะเร็วกว่าการทำระบบแบบ Page Locking จากจุดนี้เองจึงได้เกิดความต้องการทดสอบโดยเปรียบเทียบผลลัพธ์จากการทดสอบที่กระทำโดยใช้ Oracle และผลลัพธ์ที่กระทำโดยการทดสอบที่เหมือนกันโดยใช้ Sybase ทำให้ได้ว่าใน Relational Database การทำการ Locking ถ้ามีการจัดระบบเป็นแบบ Page-Locking System จะทำงานได้เร็วที่สุด

นอกจากนี้เมื่อพิจารณาการทำ Object-Level Locking ก็คือการทำระบบแบบ Page-Level Locking ซึ่งโดยส่วนใหญ่แล้ว ODBMS จะทำการล๊อกในระดับ Page – Level การทำระบบแบบ Page-Level Locking นั้นสามารถที่จะช่วยให้การบริการหรือช่วยสนับสนุนการแก้ไขปัญหาของ Concurrency

ในการใช้งานแบบ มัลติยูสเซอร์(Multi-User) ได้ จากการเกี่ยวข้องกันในระดับของการทำ Locking จะให้ผลลัพธ์ออกมาในรูปของ " False Waits " เมื่อข้อมูลนั้นถูกการเข้าถึง(Access) ของผู้ใช้สองคนในเวลาเดียวกัน จะเป็นเหตุให้ผู้ใช้คนหนึ่งจะต้องคอยเพราะว่าส่วนของข้อมูลที่จะใช้นั้นจะอยู่บนเพจ(Page)ที่ถูกผู้ใช้คนอื่นใช้งานอยู่ การเกิด " Fasle Waits " ไม่ได้เกิดเฉพาะมาจากการเกิดของ Lock Waits แต่อาจจะมาจากการเกิดของ Deadlocks ซึ่งทรานส์แซคชัน(Transaction)ที่เกิดขึ้นเหล่านั้นต้องการเพียงการเกิด Rollback เพื่อให้สามารถกลับมาทำงานใหม่ได้

มีการกล่าวถึงระดับของ Lock Granularity ของ ODBMS ที่มีขายในท้องตลาด ซึ่งระดับของการ Lock ที่ว่านี้จะอยู่ในระดับของการล๊อกในเทอมของ " Contrainer Level Lock" ซึ่งก็คือการทำการ Lock ที่หยาบมากกว่าการล๊อกของ Page –Level Lock ใน Contriner – Base Model ออบเจกต์ทั้งหมดที่อยู่ในฐานข้อมูลนั้นสามารถที่จะประกอบเข้าด้วยกันเป็น Contrainer ได้ ซึ่งสามารถมีได้หลายคอนเทรนเนอร์ แต่คอนเทรนเนอร์ทั้งหมดจะต้องมีจำนวนที่แน่นอน ถ้า Page-Level Locking นั้นถูกทำการ Lock จะเป็นผลทำให้คอนเทรนเนอร์นั้นถูกล๊อกไปด้วย ซึ่งในแต่ละคอนเทรนเนอร์จะสามารถรองรับออบเจกต์ที่ทีขนาดใหญ่ได้ ดังนั้นจะเห็นได้ว่าโมเดลของการทำการ Locking แบบนี้จะให้การบริการหรือการแก้ไขปัญหาของ Concurrecy ระบบงานแบบ Multi- User ได้โดยการใช้เทคนิคของ Page-Locking Model



รูปที่ 3.4 แสดง Object และ Page-Level Locking

จากรูปที่ 3.4 จะเห็นได้ว่า การทำ Object-Locking นั้นจะให้ประสิทธิภาพของ Concurrency สูงสุดและจะใช้ทรัพยากรของระบบต่ำสุด และตรงกันข้ามการทำ Locking แบบPage Locking นั้นจะทำ

การล๊อกทั้งเพจ อย่างน้อยก็เป็นการป้องกันการเข้าถึง(Acess)ของข้อมูลซึ่งการล๊อคแบบนี้อาจเป็นสาเหตุที่ทำให้เกิด Deadlock หรือทำให้ประสิทธิภาพโดยรวมของระบบต่ำลงได้

3.4 การไอเดนติฟายออบเจ็กต์ (Object Identifilers ,OID)

OID เป็นสิ่งที่ใช้ระบุหรืออ้างอิงออปเจ็กต์เพื่อบอกถึงความสัมพันธ์ของออบเจ็กต์มักอยู่ในรูปค่าของตัวเลขและตัวอักษรที่ไม่ซ้ำกันเพื่อจะตั้งค่าให้กับออบเจ็กต์ซึ่งหลักการสร้าง ODI ให้กับออปเจ็กต์มีผลอย่างมากกับประสิทธิภาพของ OODBMS

ODI สามารถนำเสนอได้ในหลาย ๆ รูปแบบซึ่งสามารถเป็นได้ทั้งทาง ฟิสิคอลและลอจิคอล หลักการที่จะนำมาสร้างเป็น OID แบ่งเป็น 4 หลักพื้นฐานคือ

1. ใช้ฟิสิคอลแอดเดรส

เป็นการให้ค่า OID โดยใช้เป็นฟิสิคอลแอดเดรสของออบเจ็กต์ซึ่งมักใช้กับภาษาที่ใช้ในการโปรแกรมมีข้อได้เปรียบที่มีประสิทธิภาพสูงมากในด้านความเร็วแต่ว่าหากมีการเคลื่อนย้ายทางฟิสิคอลงการเปลี่ยนแปลงค่าของแอดเดรสหรือมีการลบค่าใดๆจะทำให้การแก้ไขทำได้ไม่สะดวกและทำให้การทำงานผิดพลาดได้

2. แอดเดรสแบบโครงสร้าง (Structure address)

ค่าของ ODI จะถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนแรกจะเป็นส่วนที่เก็บหมายเลขเซกเม้นท์และหมายเลขเพจซึ่งจะมีประโยชน์ในการดึงข้อมูลจากดิสก์ส่วนที่สองจะประกอบด้วยหมายเลขของลอจิคอลสลอท (Logical slot number)เพื่อใช้สำหรับอ้างอิงของออบเจ็กต์ในเพจซึ่งวิธีนี้สามารถเคลื่อนย้ายข้อมูลได้สะดวกกว่า

3. เซอโรเกต (Surrogate)

วิธีนี้ค่า OID จะถูกสร้างขึ้นด้วยอัลกอริทึมที่จะการันตีได้ว่า จะไม่มีการซ้ำกันของ OID เช่นใช้ วัน+เวลา หรือใช้การเพิ่มค่าหมายเลขขึ้นเรื่อยๆค่าเซอโรเกต OID จะถูกแปลงกลับเป็นแอดเดรสทางฟิสิคอลและจะมีอินเด็กที่ใช้ในการค้นหา

4. เซอโรเกตแบบอื่นๆ (Typed surrogates)

วิธีการเซอโรเกตจะสร้างให้ OID มีการเก็บค่า 2 ส่วนคือ ชนิดของ ID และออปเจ็กต์ไอเดนติฟายจากนั้นจะมีตัวนับเพิ่มค่าให้กับแต่ละออบเจ็กต์ไปเรื่อย ๆ ซึ่งวิธีนี้นิยมใช้มากทั้งในภาษาที่มีลักษณะเป็นออบเจ็กต์วิชวลทูลและฐานข้อมูลเชิงวัตถุรุ่นใหม่ๆ

3.5 การสวิสซิ่งในการเก็บข้อมูลของฐานข้อมูลเชิงวัตถุ (Swizzling)

จากที่กล่าวมาว่า OID เป็นสิ่งที่ใช้อ้างอิงออบเจ็กต์ในการใช้งาน OID มักต้องมีการแปลงแอดเดรสเป็นเมมโมรี่แอดเดรสหรือเมื่อออบเจ็กต์ถูกดึงมาจากดิสก์และส่งไปยังส่วนเมมโมรี่กลางสวิสซิ่งเป็นเราเรียกการแปลงข้อมูลลักษณะนี้ว่าสวิสซิ่งซึ่งจะเป็นการเพิ่มประสิทธิภาพการทำงานของระบบประโยชน์ของการสวิสซิ่งคือเพิ่มความเร็วโดยใช้การเนวิเกทระหว่างออบเจ็กเป็นการดึงข้อมูลและเพราะการใช้แอดเดรสทางเมมโมรี่ส่วนกลางนั้นมีความเร็วกว่าการค้นหาผ่าน OID อีกทั้งถ้าเราประยุกต์ใช้ทฤษฎีนี้โดยใช้เวอทวลเมมโมรี่ก็จะเป็นการขยายขีดความสามารถเพราะออบเจกต์สามารถเก็บลงในเวอทวลเมมโมรี่ได้

3.6 การแมพออบเจกต์จากดิสสู่เมมโมรี่

DBMS ทั่วไปจะต้องมีส่วนเก็บข้อมูลกลางเรียกว่า บัฟเฟอร์เมเนเจอร์ ซึ่งจะทำการจัดการระบบบัฟเฟอร์ของฐานข้อมูลอีกทั้งยังทำหน้าที่พยายามดึงข้อมูลที่คิดว่าน่าจะใช้ถัดไปเข้ามาสู่บัฟเฟอร์เพื่อลดเวลาที่สูญเสียไปในการรอคอยจากหน่วยเก็บข้อมูลสำรองทรานเซคชั่นมักต้องการให้บัฟเฟอร์ทำการโหลดข้อมูลไว้ก่อนที่ทรานเซคชั่นจะเกิดขึ้นเมื่อทรานเซคชั่นจบลงจึงจะค่อยเก็บข้อมูลลงฐานข้อมูลบัฟเฟอร์เมเนเจอมักใช้ระบบการบริหารแบบ LRU (least recently used) อัลกอริทึมโดยจะแทนที่ข้อมูลที่ไม่มีการใช้นานที่สุด

3.7 ออบเจกต์บัฟเฟอร์ริ่ง (Object buffering)

ออบเจกต์บัฟเฟอร์ริ่งเป็นส่วนย่อยของแต่ละเพจบัฟเฟอร์เมื่อหลังจากข้อมูลนำขึ้นมาจากดิสก์แล้วเพจบัฟเฟอร์ก็จะนำข้อมูลส่งให้กับออบเจกต์บัฟเฟอร์เพื่อแยกข้อมูลออกและใช้ประโยชน์พื้นที่ของแต่ละเพจให้เต็มที่หากว่าข้อมูลมีขนาดใหญ่การเก็บข้อมูลจะกระทำแบบแสปนนิ่งทรี (tree spanning)ในหลายๆเพจซึ่งทำให้การเปลี่ยนแปลงแก้ไขข้อมูลสามารถทำได้โดยไม่จำเป็นต้องโหลดออบเจกต์ทั้งตัว

การบริหารเมมโมรี่ของระบบฐานข้อมูลเชิงวัตถุ

ในระบบฐานข้อมูลเชิงวัตถุนั้นลักษณะของข้อมูลมีไม่แน่นอน ซึ่งมักทำให้เกิดกาเบจคอลเลคชั่น(garbage-collection)ขึ้นเรามีวิธีพื้นฐาน 3 แบบในการบริหารเมมโมรี่คือ

1. ให้ช่วงเวลาหนึ่งๆ มีการลบข้อมูล

เป็นเทคนิคที่ง่ายแต่ไม่ค่อยมีประสิทธิภาพโดยเราจะทำการนับเวลาเมื่อถึงช่วงเวลาหนึ่งหากว่ามีออบเจ็กต์ใดไม่ได้ใช้ก็จะนำออกจากเมมโมรี่เป็นการประยุกต์ใช้อัลกอริทึม LRU

2. ให้โปรแกรมเมอร์เป็นผู้จัดการเอง

วิธีนี้เป็นงานหนักของโปรแกรมเมอร์ที่จะต้องทำการสร้างส่วนของการตรวจสอบเมมโมรี่เอง

3. กาเบจคอลเลคชั่นอัตโนมัติ

วิธีนี้ค่อนข้างสะดวกแต่ไม่ยืดหยุ่นเพราะบางทีเรายังอยากให้มีการเก็บข้อมูลนั้นอยู่แต่ว่ายังไม่ใช้วิธีนี้จะพบได้ในภาษา

3.8 คลัสเตอร์ริ่ง(Clustering)

คลัสเตอร์ริ่งเป็นการกล่าวถึงขั้นตอนการจัดกลุ่มของออบเจ็กต์ที่เกี่ยวข้องกันเข้าไว้ด้วยกันโดยอยู่ในระดับของคอนเซ็ปช่วลในหน่วยความจำหลักหรือในหน่วยความจำสำรองหรือเรียกว่าคอนเซ็ปช่วล คลัสเตอร์ริ่ง หน่วยในการเก็บข้อมูลของคลัสเตอร์เราเรียกว่า เซกเม้น(Segment) ซึ่งเซกเม้นจะเป็นส่วนที่ใช้ในการส่งข้อมูลจากหน่วยความจำสำรองไปยังออบเจ็กบัฟเฟอร์ริ่งเราอาจแบ่งการเก็บข้อมูลในคลัสเตอร์ได้เป็น 4 รูปแบบคือ

1. เก็บ 1 ออบเจ็กตีต่อหนึ่งคลัสเตอร์

มักใช้กับในกรณีที่ออบเจ็กต์มีขนาดใหญ่มากๆวิธีนี้ไม่ค่อยดีนักในการส่งข้อมูล

2. เก็บออบเจ็กต์ร่วมกับออบเจ็กต์ย่อยของมันเข้าด้วยกัน
   ใช้เก็บในกรณีที่ต้องใช้งานร่วมกัน
3. เก็บออบเจ็กต์ที่มีคุณสมบัติเหมือนกันเข้าด้วยกัน
   เช่นเก็บข้อมูลที่ค่าเป็น color = "red " เหมือนกันหมด

การจัดการเกี่ยวกับคลัสเตอร์

1. การเพิ่มค่าออบเจ็กต์เข้าไปในคลัสเตอร์
2. การรวมออบเจ็กต์ (Object migration)
3. การเปลี่ยนแปลงคลัสเตอร์ (Reclustering)

เวอร์ชั่น

ในบางครั้งเมื่อเราทำการแก้ไขข้อมูลเดิมข้อมูลที่ถูกแก้ก็เปลี่ยนไปดังนั้นหากเรายังต้องใช้ข้อมูลนั้นอยู่เช่นในกรณีที่ concurrency control เราจำเป็นต้องมีการเก็บข้อมูลเก่าไว้หลายเวอร์ชั่นความสามารถพื้นฐานของระบบที่จะสามารถทำเวอร์ชั่นคือ

1. เวอร์ชั่นต้องทำกับออบเจกต์ไม่ใช่คลาส
2. ทุกออบเจกต์สามารถมีได้หลายเวอร์ชั่น
4. ต้องสามารถใช้ข้อมูลได้ทุกเวอร์ชั่นแต่สามารถแก้ได้เฉพาะเวอร์ชั่นล่าสุดส่วนเวอร์ชั่นเก่าให้อ่านได้อย่างเดียว

ข้อมูลโดยทั่วไปของ ODBMS และ RDBMS

ODBMS ได้รับความสนใจอย่างมากตั้งแต่ปี 1980 ในตอนนั้นออบเจกต์นับเป็นคลื่นลูกใหม่ของวงการคอมพิวเตอร์แต่ปัญหาของ ODBMS ยุคแรกๆที่พบกันในตอนนั้นคือยังไม่เป็นระบบฐานข้อมูลที่สมบูรณ์ขาดความสามารถในการแบ็กอัพแอนด์รีโคเวอรี่ , คาต้าโมเดลก็ยังสับสนอยู่ , ภาษาก็ไม่มีมาตรฐาน การสืบค้นก็มีปัญหาในปัจจุบัน ODBMS ได้พัฒนาไปมากจนมีความเสถียรของระบบใกล้เคียงกับ RDBMS

RDBMS เหมาะสำหรับเก็บข้อมูลที่ไม่ซับซ้อนเช่น ชื่อ, สกุล, ที่อยู่ หรือตัวเลขต่างๆข้อดีคือได้รับการพัฒนามานานจนค่อนข้างเสถียรมากแล้วและยังทำงานได้เร็วคุณสมบัติเหล่านี้ทำให้ RDBMS เหมาะกับงานด้านธุรกิจและ Application ทางด้านการเงินอย่างไรก็ตามโมเดล RDBMS เป็นแบบ 2 มิติคือเป็นแถวกับคอลัมน์เท่านั้นในขณะที่โมเดลแบบออบเจ็กเก็บโมเดลที่ซับซ้อนตามความเป็นจริงได้ไม่ต้องไปพยายามบีบข้อมูลให้เหลือเพียงแค่ 2 มิติอย่าง RDBMS ทำให้ ODBMS เหมาะกับงานทางด้านมัลติมีเดียและอินเทอร์เน็ต ในด้าน Query RDBMS เหมาะกับลักษณะของ Query ที่ซับซ้อนแต่ข้อมูลที่นำมาใช้ไม่ซับซ้อนซึ่งตรงข้ามกับ ODBMS ที่เน้นในจุดของการเก็บข้อมูลที่ซับซ้อนมากๆแต่ Query ไม่ซับซ้อน

ผลิตภัณฑ์ประเภทออบเจกต์รีเลชันนอลเก็บข้อมูลทั้ง Relational และ Object ไว้ในระบบเดียวกันสิ่งที่แตกต่างจาก ODBMS คือการจัดการกับ Object ไม่ได้เกิดขึ้นโดยตรงที่ส่วนแก่นของระบบจัดการฐานข้อมูลแต่มีเชลล์มาจัดการกับออบเจ็กต์แทนแล้วค่อยส่งต่อให้ตัวจัดการฐานข้อมูลเก็บไว้ในรูปของ Relational ต่อไป

ส่วนฐานข้อมูลแบบเอ็กซ์เท็นด์รีเลชั่นนอลเช่น Infomix (Data blades) เราสามารถกำหนดชนิดของข้อมูลเป็นออบเจ็กต์ในฐานข้อมูลแบบเอ็กซ์เท็นด์รีเลชันนอลค่อนข้างง่ายซึ่งเหมาะกับการเก็บข้อมูลจำพวกลายนิ้วมือหรือรูปถ่ายแต่ไม่เหมาะกับการเก็บภาพเคลื่อนไหว

ODBMS มีลักษณะเป็นแบบแอคทีฟเนื่องจากในตัว Object มีคุณสมบัติเรื่อง class และ method อยู่ด้วยเมื่อมีการเรียกใช้ Object โดยส่ง Massage ไป Object จะทำงานตาม method ที่ผูกติดกับตัวเองโดยผู้ใช้ไม่จำเป็นต้องเขียนโปรแกรมส่วนนั้นอีก

บทที่ 4
การประยุกต์ใช้ทฤษฎีกับระบบฐานข้อมูลเชิงวัตถุ

4.1 การประยุกต์ใช้ ทฤษฎีของ Object – Oriented ใน OODBMS (Jasmine)

Object กับ Jasmine

ตามหลักการตำราต่างประเทศมักมีการสร้างนิยามของ Object ต่างกันออกไปแต่มักมีแนวทางเดียวกันคือ Object เป็นสิ่งทุกๆสิ่งที่เรารู้จักและมีชื่อปรากฏให้เรียกได้เช่น ในระบบธุรกิจก็จะมี Object employee, manager etc. ในด้าน Web page Object ก็คือ text pointer ไปสู่ Web page อื่นและ Multimedia data types ต่างๆ

Object มีองค์ประกอบพื้นฐาน 2 ส่วนหลักคือ

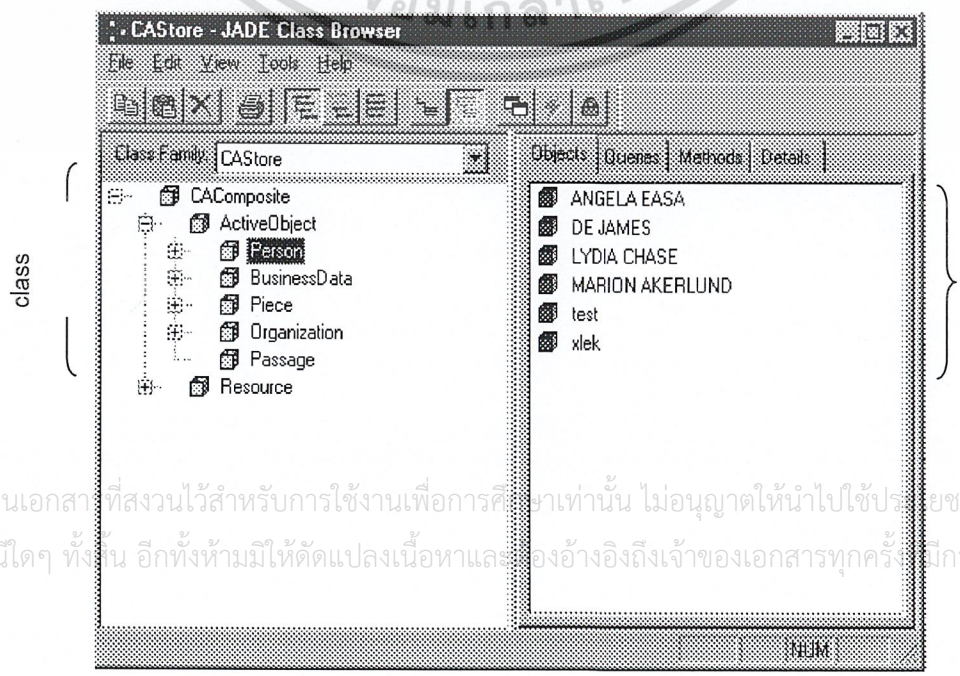
1. Properties – เป็นสิ่งที่บอกถึงค่าหรือข้อมูลบน Object นั้นๆ เช่น employee has a name , salary เป็นต้น
2. Methods – จะบอกถึงคุณลักษณะหรือ behavior ของแต่ละ Object เช่น employee can take a new project or change job.

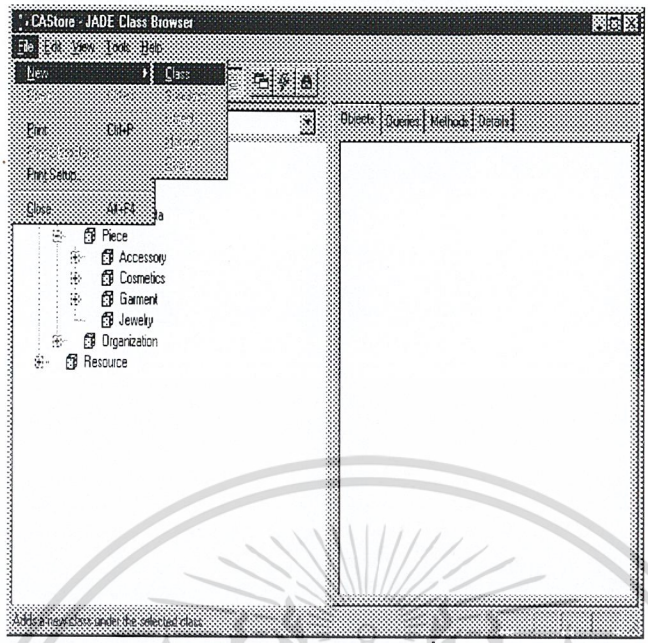
4.2 Class กับ Jasmine

คือ Abstract definition ของสิ่งที่เราจะระบุ class มีประโยชน์ในการแบ่งกลุ่มเช่นเมื่อเราใช้ Object – Oriented Technology เราต้องการระบุ Object ใดๆที่เราต้องการนำเสนอในระบบของเรา class จะเป็นตัวแบ่ง object เหล่านั้นออกเป็นหมวดหมู่ อาจกล่าวได้ว่า class จะ specifies Properties และ Method ของ Object ภายในตัว class นั้นๆเมื่อมีการสร้าง Object ใหม่นั่นคือเรากำลังสร้าง “ Instance” ของ class

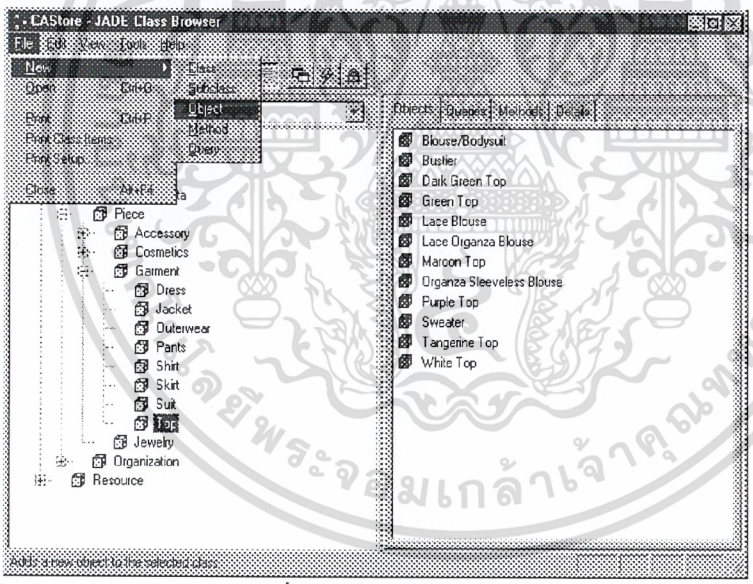
ในระบบของ Jasmine เราสามารถระบุ class และ Object ได้ง่ายโดยใช้ Jasmine Class Browser ซึ่งจะแสดงดังรูปที่ 4.1

จากรูป Class Person ประกอบด้วย Object บุคลที่เป็นสมาชิกของร้านค้า CA Store การสร้าง Class ใหม่ทำได้โดย





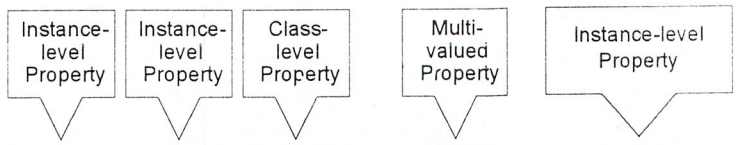
จากนั้นเราสามารถกำหนด Property ของ Object ซึ่งจะถูกสร้างเป็นส่วนหนึ่งของ Class's properties นั้นจะเรียกว่า Metadata ของ class



รูปที่ 4.1 แสดงการสร้าง Object ใหม่

Properties ใน Jasmine แบ่งเป็น 5 ส่วนคือ

1. Instance – Level Properties
2. Class – Level Properties
3. Multi – Values Properties
4. Attribute Properties
5. Relationship Properties



Name	Extension	Payment Schedule	Languages Spoken	Manager
John Brown	222	Semi-monthly	English, French	Frank Nelson
Fred Smith	201		English, Spanish	Mary Johnson
Jane Jones	208		English, French, Italian	Bob Williams

ดังรูปที่ 4.2 การแสดงค่าภายในออบเจกต์

Instance – Level Properties

จะเป็น Object เฉพาะซึ่งไม่ซ้ำกันจากรูปที่ 4.2 คือ Name และ Extension

Class – Level Properties

จะบอกถึง Class ที่มีอยู่ของ Object เช่น การ share ค่าเดียวกันของ Instance ภาย ใน Class ซึ่งไม่ได้หมายความว่าจะเป็นการเก็บค่าซ้ำกันแต่เป็นการเรียกโดย Method ซึ่งมีประโยชน์ในการ Maintain และ Update เพราะมีเพียงค่าเดียวใน Class

Multi – Values Properties

เป็นความสามารถในการใส่ค่ามากกว่า 1 ค่าได้

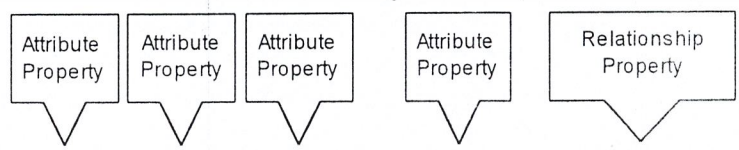
Attribute and Relationship

Attribute คือ property ที่มีค่าเป็น constance เช่น number, string, date ดังรูปที่ 4.3

Relationship คือ property ที่มีค่าเป็น Object ตัวอื่นๆ

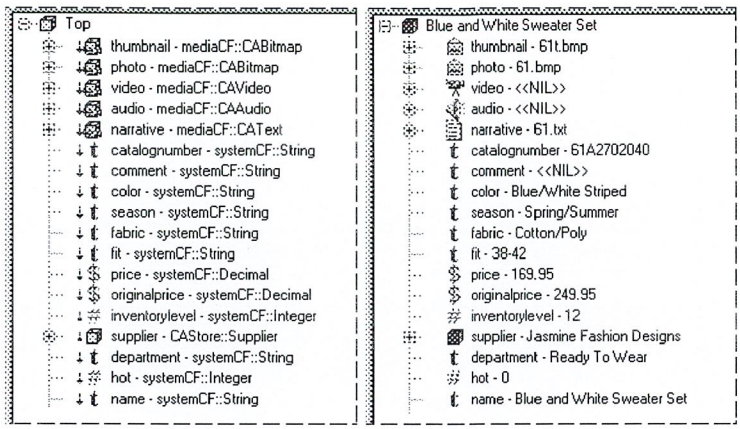
Many to Many Relationship

คล้ายใน properties relation เป็นสิ่งที่สามารถดึงความสามารถคล้ายของ relational database มาใช้ได้ในด้าน Many to Many เช่น employee หลายๆคน ถูกใช้ให้ทำงานหลายๆงานใน Jasmine เราสามารถใช้คุณสมบัติของ OODBMS คือใช้ relationship properties โดยใช้ relationship properties โดยใช้ properties multi – value ทั้ง 2 ด้าน ดังรูปที่ 4.4



Name	Extension	Payment Schedule	Languages Spoken	Manager
John Brown	222	Semi-monthly	English, French	Frank Nelson
Fred Smith	201		English, Spanish	Mary Johnson
Jane Jones	208		English, French, Italian	Bob Williams

รูปที่ 4.3 แสดงตัวอย่าง Property ใน Jasmine

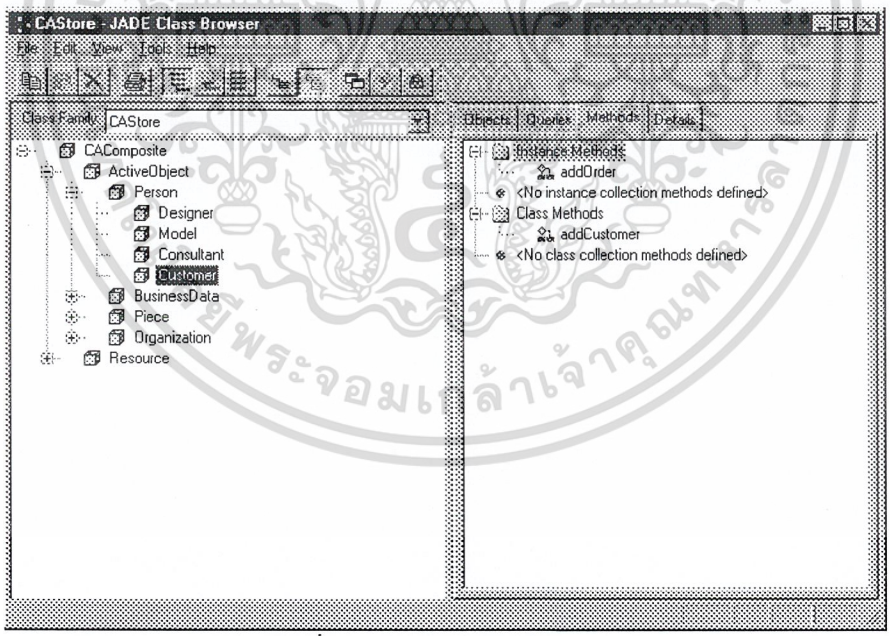


รูปที่ 4.4 ตัวอย่าง class property และ Object property ใน Jasmine

4.3 Method ใน Jasmine

แบ่งเป็น 3 แบบดังรูปที่ 4.5

1. Instance – Level Methods
2. Class – Level Methods
3. Collection Level Methods



รูปที่ 4.5 แสดง Method แบบ ต่างๆ

Instance – Level Methods

จะใส่เข้าไปใน Object เช่น ใน Method ที่จะบอกถึง ปีที่แต่ละ employee ทำงานให้องค์กร

Class – Level Methods

ใส่เข้าไปใน Class ทั้งหมดเช่น เราสร้าง Method ที่ทำการสร้าง Object ใหม่โดยใช้การเก็บใน external file

Collection – Level Methods

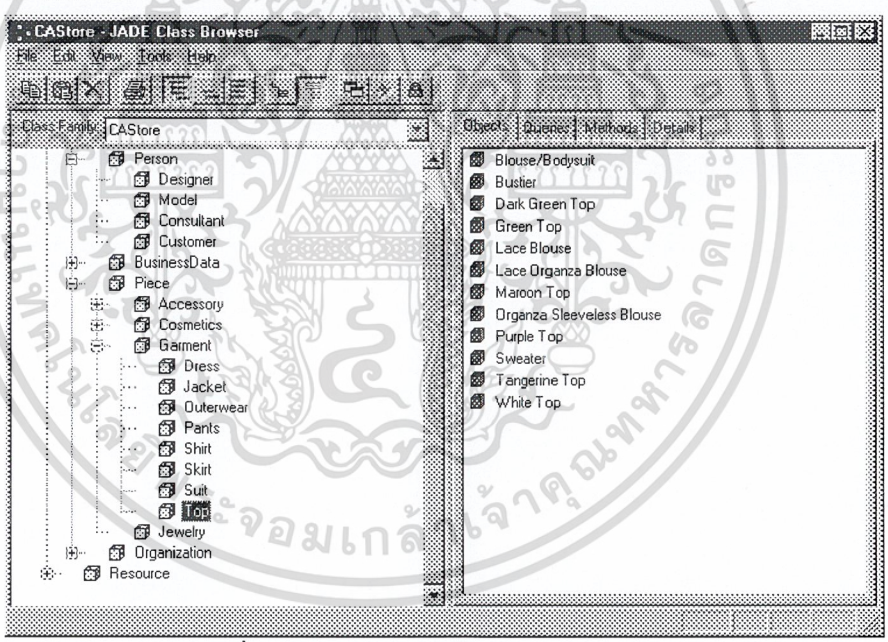
เก็บเข้าในกลุ่มที่ตั้งขึ้น

จากตัวอย่างประกอบด้วย Instance method คือ addOrder ซึ่งจะทำการเพิ่ม Order ใน User เพียงคนเดียวและอีก Method คือ Class Method ชื่อ addCustomer ซึ่งจะสร้าง Instance ใหม่ของ class นั่นคือ addCustomer จะเป็น Class- Level Method ของ Customer class

4.4 Inheritance ใน Jasmine

เป็นการที่อนุญาติให้สร้าง Class definition (หรือ Sub Class) ซึ่งประกอบด้วย propertie และ method ของ parent (หรือ Super Class) โดยสามารถเพิ่ม Property หรือ Methods ใหม่ๆหรือทำการเปลี่ยนแปลงแก้ไขก็ได้

ตัวอย่าง ใน Jasmine

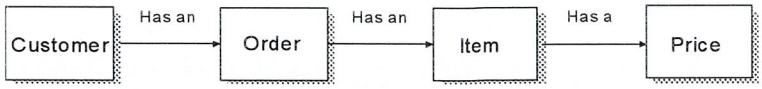


รูปที่ 4.6 แสดงการ Inheritance ใน Jasmine

ในรูปที่4.6 คือ Class Dress , Jacket Top etc. ซึ่งจะประกอบด้วย Sub Class ดังรูปด้านขวามือในการแบ่งการ Inheritance ง่ายๆทำได้โดยใช้ " is-a" rule เช่น TOP is a Garment นั่นคือ TOP is a Subclass of Garment และ Inherit properties ทั้งหมดของ Garment การ Multiple Inheritance คือ การ Inherite properties และ Method จาก Super Class มากกว่า 1

4.5 Agregation กับ Jasmine

เป็นการบอกถึงความสัมพันธ์ของ Object ในแต่ละ Object จะประกอบด้วย references สู่ Object ตั้งแต่ 1 Object โดยเราจะใช้คำแทน Aggregation ว่า "has-a" relationship ดังรูป



Concept ของ aggregation หมายถึง การที่เราสามารถซ่อนความาสัมพันธ์ระหว่าง Objec และเลือกดูเฉพาะ Object ที่เกี่ยวข้องกับที่ต้องการใช้

เช่นเราสามารถสร้าง Method คำนวณ การสั่งซื้อสินค้าของ Customer เราต้องการเพียง Custommer Object และ Price Object เราไม่จำเป็นต้องทราบเกี่ยวกับความสัมพันธ์อื่นๆของทั้ง 2 อีก

4.6 เมจเสจในระบบฐานข้อมูลเชิงวัตถุและการประยุกต์

ในระบบฐานข้อมูลเชิงวัตถุก็ต้องมีการติดต่อซึ่งกันและกันในระหว่างออบเจกต์ด้วยกัน ซึ่งการติดต่อนี้โดยพื้นฐานทั่วไปก็ใช้การติดต่อโดยเมจเสจซึ่งในแต่ละออบเจกต์จะทำการสร้างต้นแบบของเมจเสจไว้เพื่อตรวจสอบเมื่อมีการส่งเมจเสจมาให้ว่าจะให้เกิดการกระทำ(Action)ใดเกิดขึ้นซึ่งการสร้างต้นแบบของเมจเสจนี้จะประกาศไว้ที่เมธอดภายในออบเจกต์ตั้งนั้นในเมธอดของออเจกต์นั้นมีส่วนประกอบพื้นฐาน 2 ส่วนคือ เมจเสจแพทเทอน (message pattern) และตัวบอดี้ (Body of method) ถ้าเรามองในส่วนที่ลึกลงไปในเมจเสจก็จะประกอบด้วย

1. OID หรือ ส่วนที่จะแสดงว่าออบเจกต์ใดเป็นผู้ส่งเมจเสจ
2. ส่วนระบุ (selector) จะบอกถึงว่าเมธอดใดเป็นผู้ร้องขอ
3. นิพจน์ที่จะกระทำด้วย

ตัวอย่างเช่นในเมจเสจ 2+8 เราจะบอกได้ว่า 2 คือ ออบเจกต์ที่เป็นผู้รับ (receiving Object) + คือเมธอดที่จะทำการเอ็กซิคิ้วและ 8 คือนิพจน์ร่วม

เมจเสจในทางทฤษฎีแบ่งได้เป็น 3 แบบ คือ

1. Unary message

   เป็นชนิดของเมจเสจที่ง่ายที่สุดประกอบด้วยออปเจ็กต์ที่เป็นผู้รับและส่วนระบุเพียงตัวเดียวเช่น 7 neg ค่าที่จะรีเทอนคือ -7

2. Binary message

   ประกอบด้วยออบเจ็กต์ผู้รับ,ส่วนระบุและนิพจน์ร่วมเพียงตัวเดียว เช่น myObject = youObject ค่าที่รีเทอนจะมี 2 แบบคือ จริงและเท็จ โดยจะเป็นจริงถ้าค่าทั้งสองเท่ากันหรือมี ODI ที่เท่ากันและจะเป็นเท็จถ้าค่าทั้งสองไม่เท่ากัน

3. Keyed message

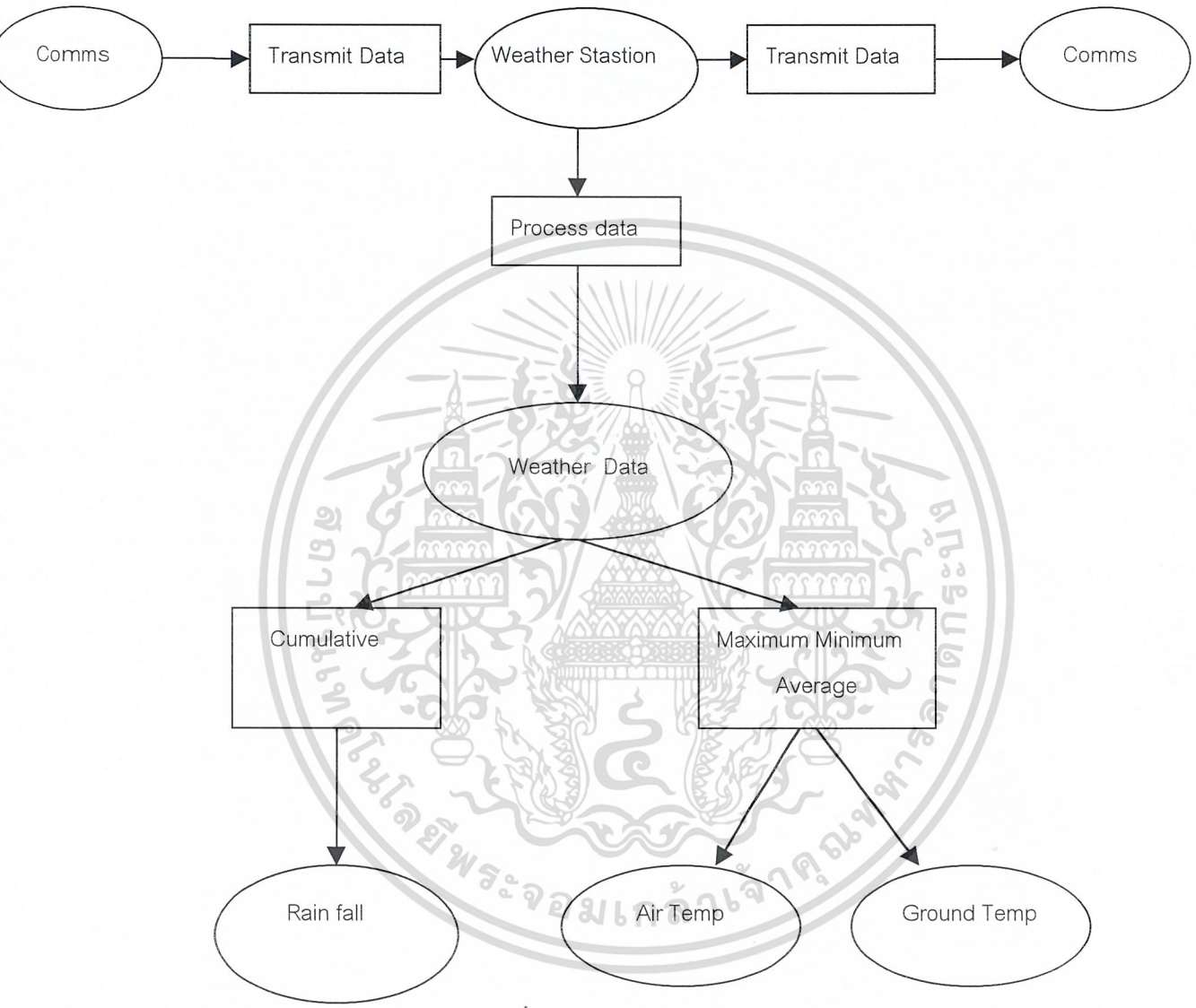
   ประกอบด้วย 1 ออบเจ็กต์ผู้รับและมีคู่ของนิพจน์ที่กระทำร่วม 1 ตัวขึ้นไปโดยมักใช้การแบ่งการกระทำโดยเครื่องหมาย " : " เช่น

   arrayOfstring at : (2+1) put : 'Black'

   ซึ่งก็หมายถึงให้ใส่ค่า 'Black' ที่ตำแหน่งข้อมูลที่ 3 เป็นต้น

4.7 ลักษณะโมเดลในการส่งเมจเสจ

ในการส่งเมจเสจระหว่างออบเจกต์เราสามารถนำเสนอในรูปของโมเดลได้เพื่อความสะดวกในการเข้าใจ การส่งข้อมูลภายในของแต่ละออบเจ็กต์ว่ามีการส่งเมจเสจเพื่อขอบริการได้อย่างไรโดยในแต่ละออบเจกก็จะมีการสร้างต้นแบบของเมจเสจให้ตรงกับเมจเสจที่จะได้รับ



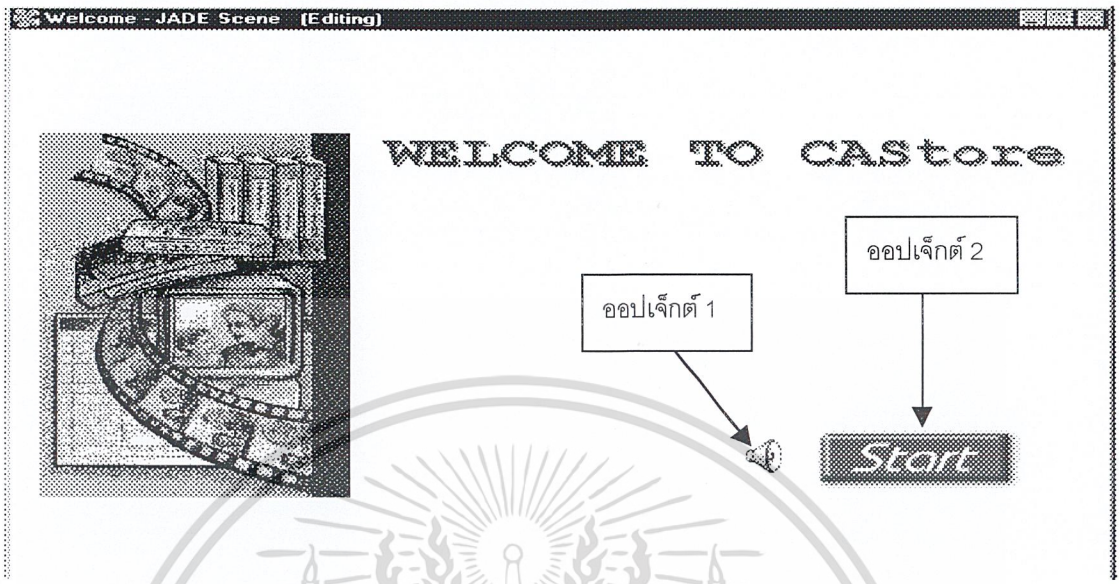
รูปที่ 4.7 แสดงโมเดลในการส่งเมจเสจ

จากรูปที่ 4.7 แสดงการส่งเมจเสจ (ในกรอบสี่เหลี่ยม) ไปยังออบเจกต์ที่ต้องการติดต่อ (วงรี)เพื่อร้องขอบริการจากออบเจกต์เป้าหมายให้ส่งข้อมูลมาให้จากตัวอย่างเป็นระบบอุตุนิยมวิทยามีการวัดค่าระดับน้ำฝนและการวัดค่าอุณหภูมิมาเก็บไว้จากนั้นหากมีออบเจกต์อื่นขอข้อมูลหรือบริการซึ่งตรงกับที่ออบเจกต์เป้าหมายมีก็จะทำการส่งข้อมูลที่ต้องการกลับไปให้

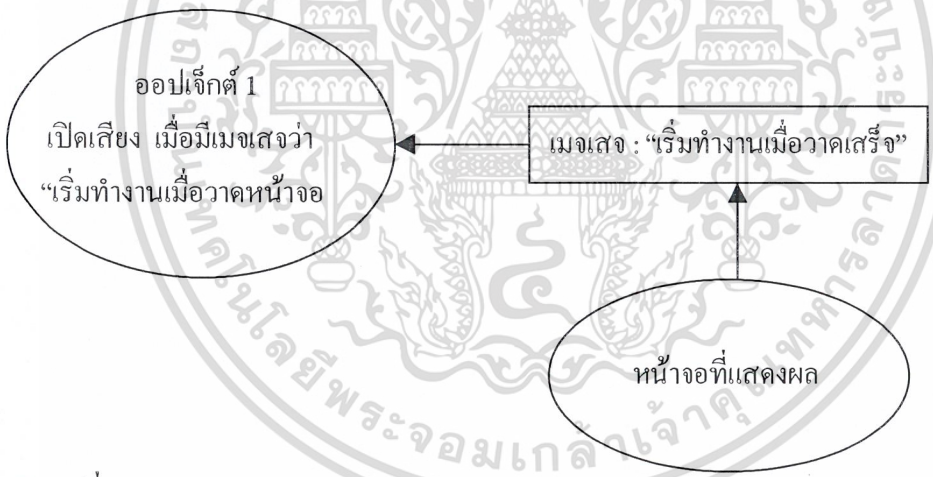
4.8 Jasmine กับ ประยุกต์วิธีการส่งเมจเสจ

ในจัสมินการส่งเมจเสจและการสร้างต้นแบบเมจเสจทำได้ค่อนข้างง่ายเนื่องจากทูลที่มีมากับระบบจัสมินเป็นลักษณะกราฟฟิกทูลการจัดการเกี่ยวกับเมจเสจจึงดูเข้าใจง่าย

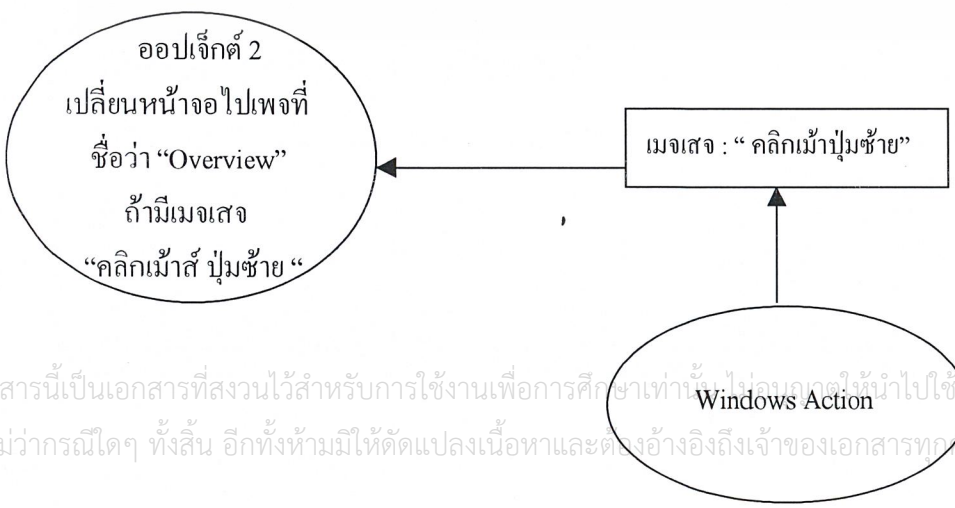
จากรูปจะยกการทำงานของออบเจกต์ทั้งสองมาอธิบายเพื่อให้ง่ายแก่การเข้าใจรูปโมเดลด้านล่างจะแสดงรูปแบบการส่งเมจเสจซึ่งกันและกันในหน้าจอนี้



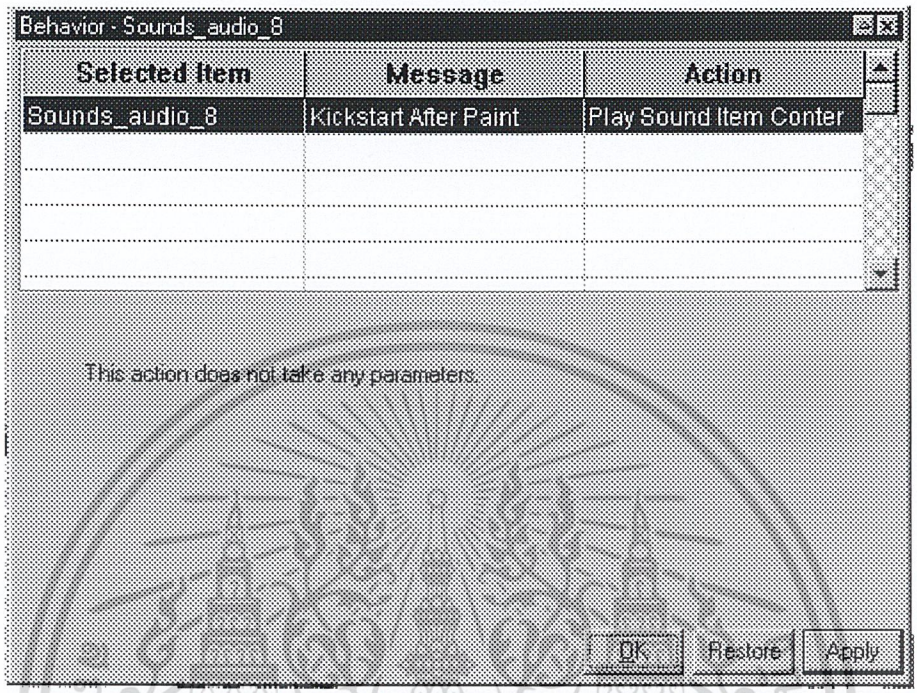
ออปเจ็กต์ 1 (sound_audio8)



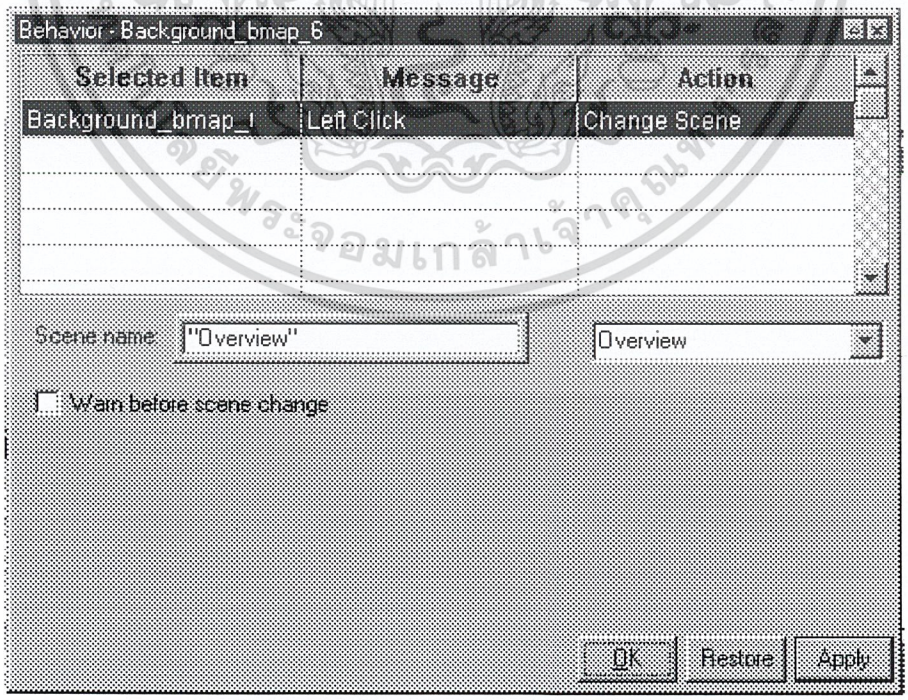
ออปเจ็กต์ที่ 2



จากรูปของออปเจ็กต์บีฮาฟวิเออซึ่งจะแสดงการทำงานที่ออปเจ็กต์นั้นถูกกำหนดไว้ได้ โดยเราสามารถแสดงให้เห็นในรูปของการประกาศเมจเสจ ใน Jasmine ได้ดังรูปที่ 4.8



รูปที่ 4.8 แสดงการประการค่าให้กับออปเจ็กต์ที่ 1

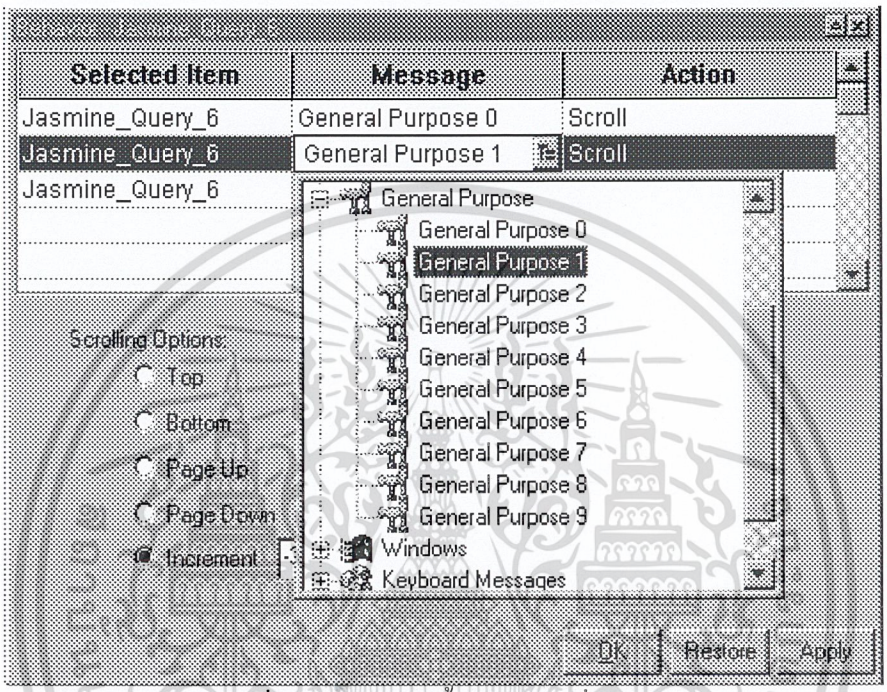


รูปที่ 4.9 แสดงการประกาศค่าให้กับออปเจ็กต์ที่ 2

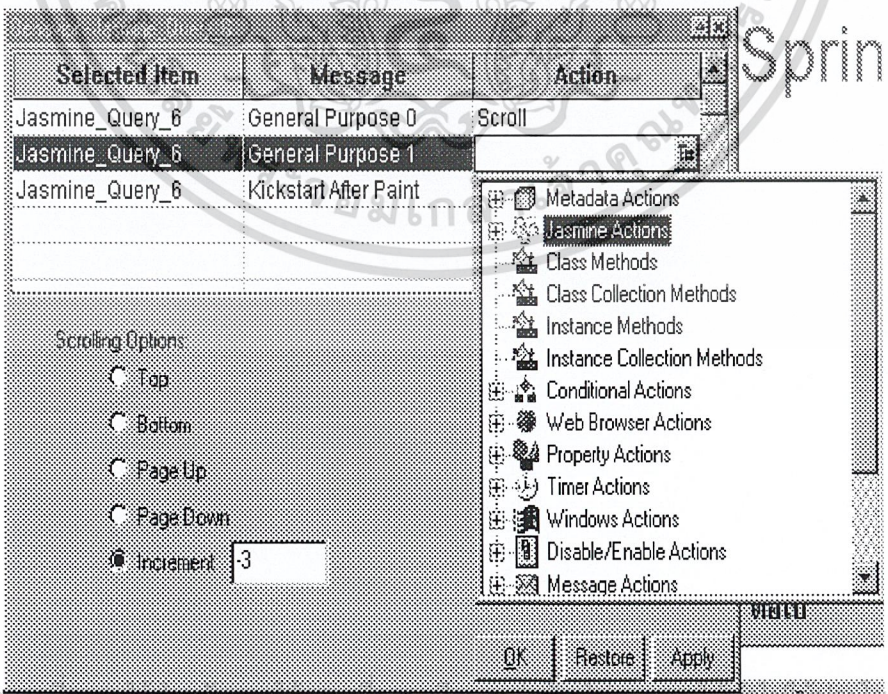
จากรูปที่ 4.9 เราจะเห็นถึงการส่งเมจเสจระหว่างออบเจกต์เมื่อมีเหตการณ์ใดๆเกิดขึ้นจะมีผลต่อการกระทำของออบเจกต์นั้นด้วยหากว่ามีการกำหนดบีฮาฟวิเอออะไรไว้

4.9 การกำหนดการกระทำตามเมจเสจที่กำหนดเอง

ใน Jasmine ได้ให้ความสามารถในการที่เราสามารถกำหนดเมจเสจที่ออปเจ็กต์หนึ่งๆส่งไปยังออปเจ็กต์อื่นๆได้ด้วยตัวผู้ใช้งานเองเรียกเมจเสจลักษณะนี้ว่า เมจเสจเอนกประสงค์ ดังรูปที่ 4.10



รูปที่ 4.10 แสดงการตั้งค่าเมจเสจเพื่อใช้งานเอง



รูปที่ 4.11 แสดงให้เห็นถึงการตั้งค่าการกระทำต่างๆซึ่งมีการกระทำให้เลือกใช้หลายชนิด

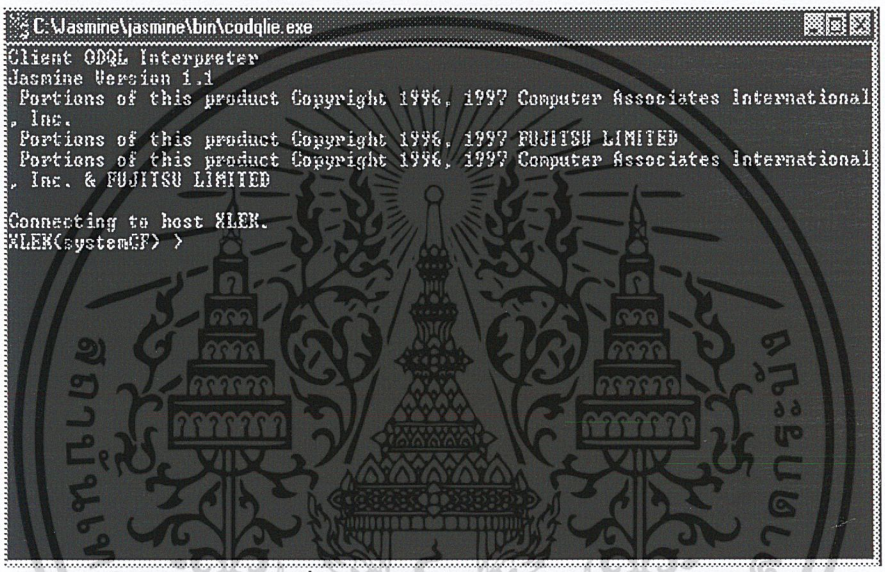
4.10 ภาษา OQL กับการประยุกต์ใช้ใน Jasmine เป็นภาษา ODQL

ภาษา ODQL เป็นภาษาที่ใช้เรียกค้นข้อมูลภายในของ Jasmine ซึ่งเป็นภาษาที่ยืดหยุ่น สามารถทำการเรียกค้นข้อมูลตามเงื่อนไขที่เราระบุได้ในเบื้องต้นพื้นฐานของ ODQL มีลักษณะการเขียนและค่ารีเทอนเป็นข้อความอักษรซึ่งในจัสมินแบ่งการทำงานออกเป็น 2 ส่วนคือ

1. การเอ็กซิคิ้วภาษา ODQL (ODQL statement)
2. การอินเตอพรีดภาษา ODQL (ODQL interpreter)

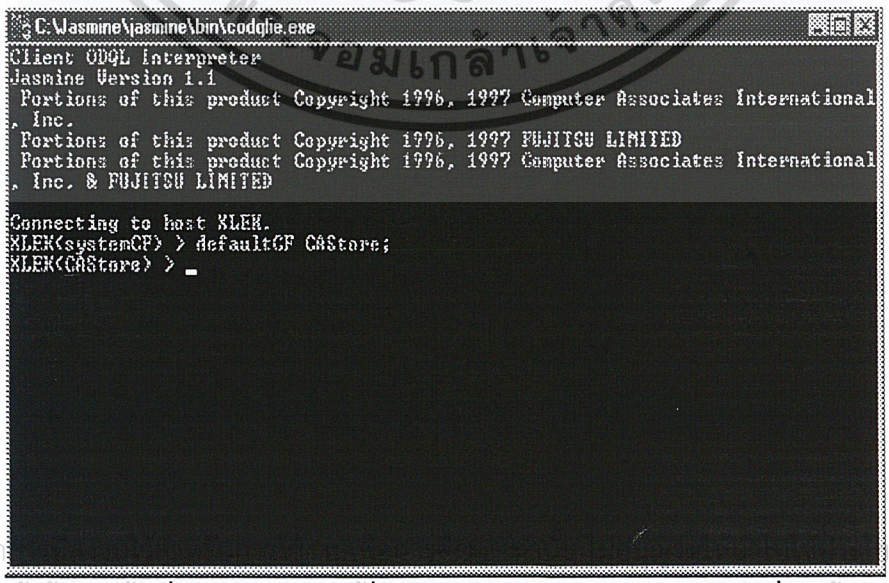
การทำงานของ ODQL ใน Jasmine ดังรูปที่ 4.12 – 4.15

1. เรียกใช้ ODQL อินเตอพรีด (codqlie)



รูปที่ 4.12 แสดงหน้าจอ ODQL interprete

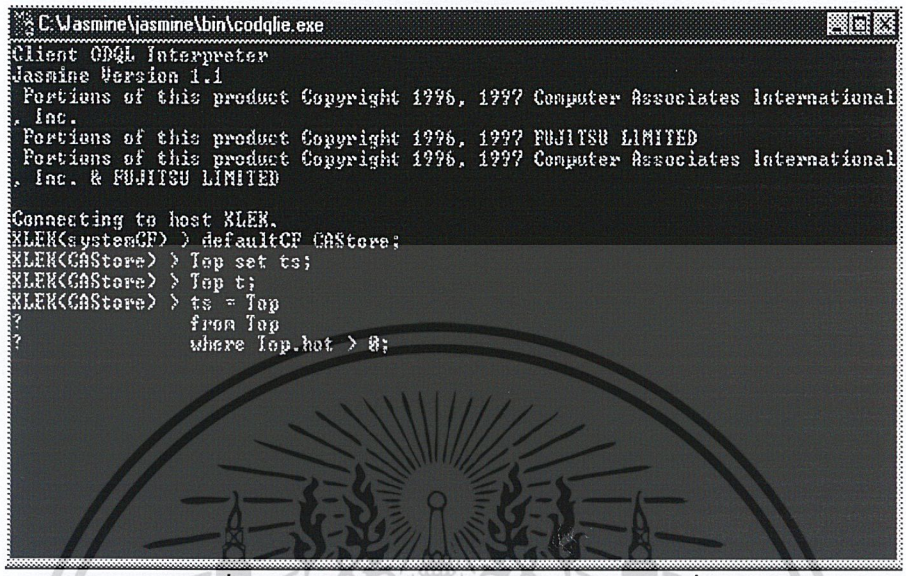
2. เลือกค่าฐานข้อมูลที่จะใช้งาน



รูปที่ 4.13 แสดงการเปลี่ยนฐานข้อมูลดีฟอลให้เป็นฐานข้อมูลที่เราต้องการ

3. ตั้งค่าตัวแปรและสร้างคิวรี่

จากรูปเป็นคิวรี่ที่จะเลือกสินค้าที่มียอดขายได้แล้วออกมาแสดง



รูปที่ 4.14 แสดงการประกาศตัวแปรและสร้างคิวรี่อย่างง่าย

4. เริ่มทำงานทรานเซคชั่นและตรวจคิวรี่

5. ตรวจผลลัพธ์

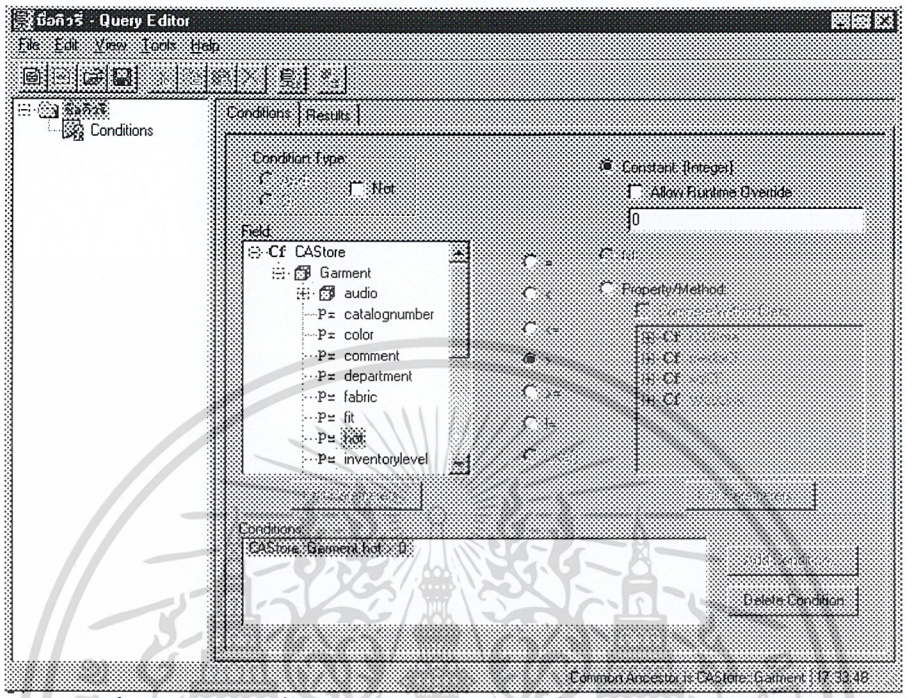
เราจะตรวจผลลัพธ์โดยใส่คำสั่ง scan(ts,t) {t.print(); };

รูปที่ 4.15 ตัวอย่างผลลัพธ์ที่ได้

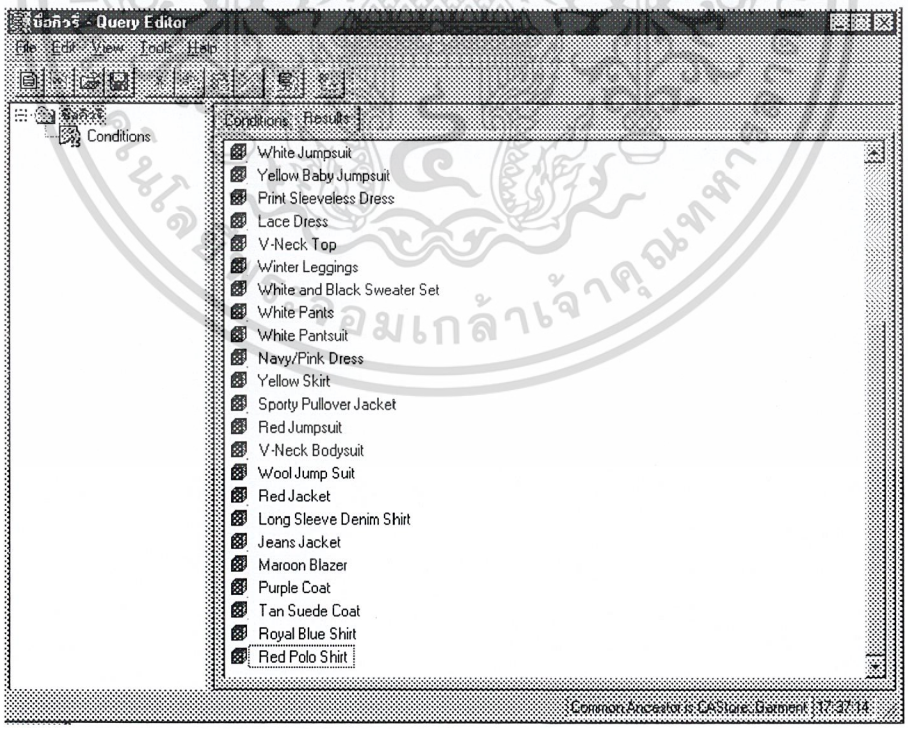
นอกจากนี้ ODQL ยังมีคำสั่งอื่นให้ใช้อีกมากเช่นการนับทำได้โดย

```
Interger cnt;
cnt = ts.count();
cnt.print();
```

จากที่กล่าวมาเป็นพื้นฐานเพื่อให้เข้าใจการทำงานของทูลแบบกราฟฟิกที่ Jasmine มีมาให้ซึ่งสามารถทำคิวรี่ได้ง่ายกว่าโดยเป็นลักษณะการใส่ออปเจ็กต์ของคิวรี่นี้ลงไปในหน้าจอที่เราต้องการสร้าง ดังรูปที่ 4.16

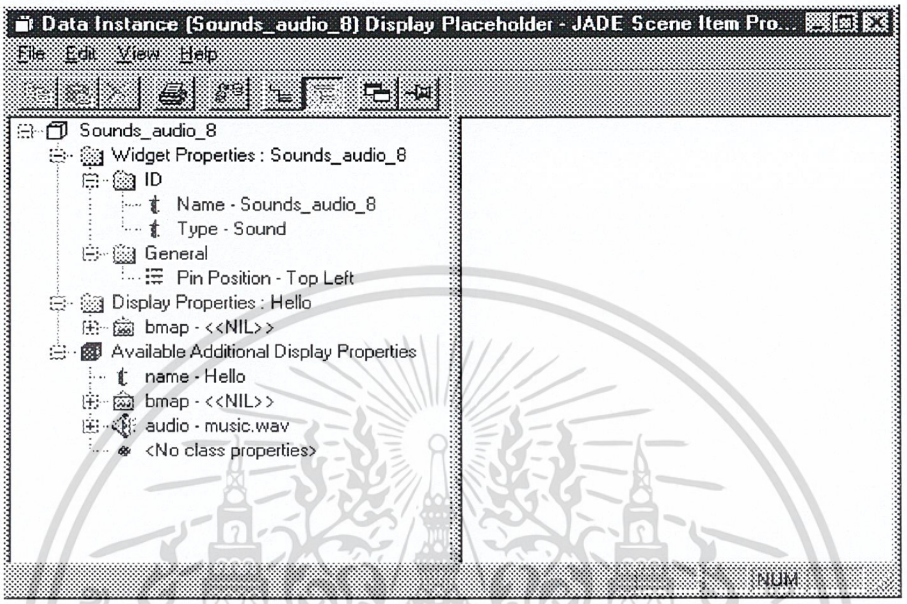


รูปที่ 4.16 แสดงคิวรี่จากตัวอย่างในโหมดเท็กซ์ซึ่งจะให้ผลลัพธ์เหมือนกันคือ

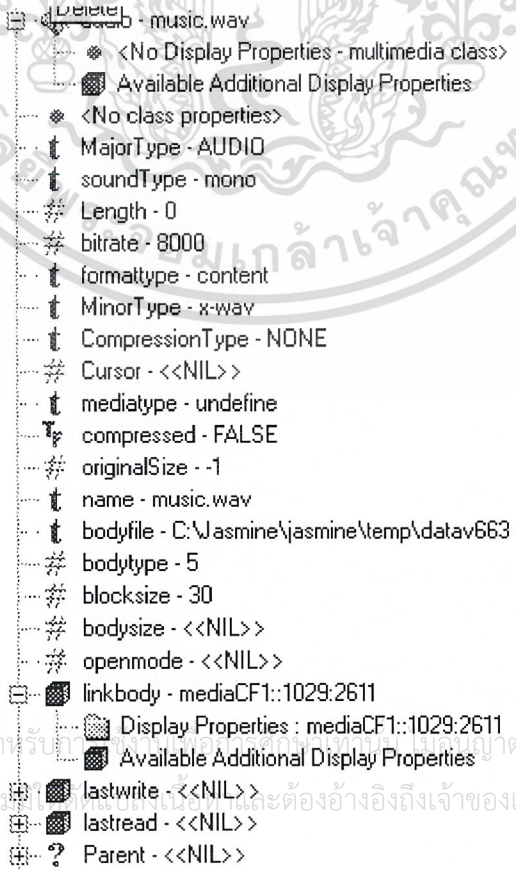


4.11 Jasmine กับการเก็บข้อมูล

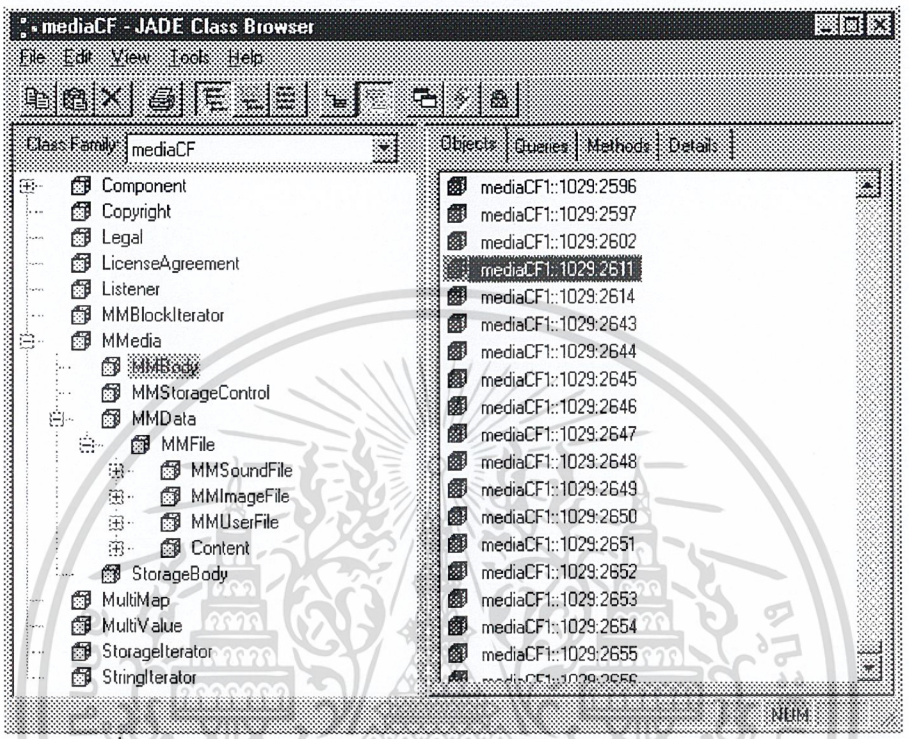
จากทฤษฎีในการเก็บข้อมูลของระบบฐานข้อมูลเชิงวัตถุที่มี 2 แบบพื้นฐาน(ตามบทที่ 3)นั้นตัว Jasmine ได้ใช้การเก็บข้อมูลแบบ normalized model ซึ่งมีลักษณะการแยกเก็บข้อมูลออกเป็นส่วนๆโดยแต่ละส่วนจะถูกเชื่อมโยงโดย OID ดังรูปที่ 4.17



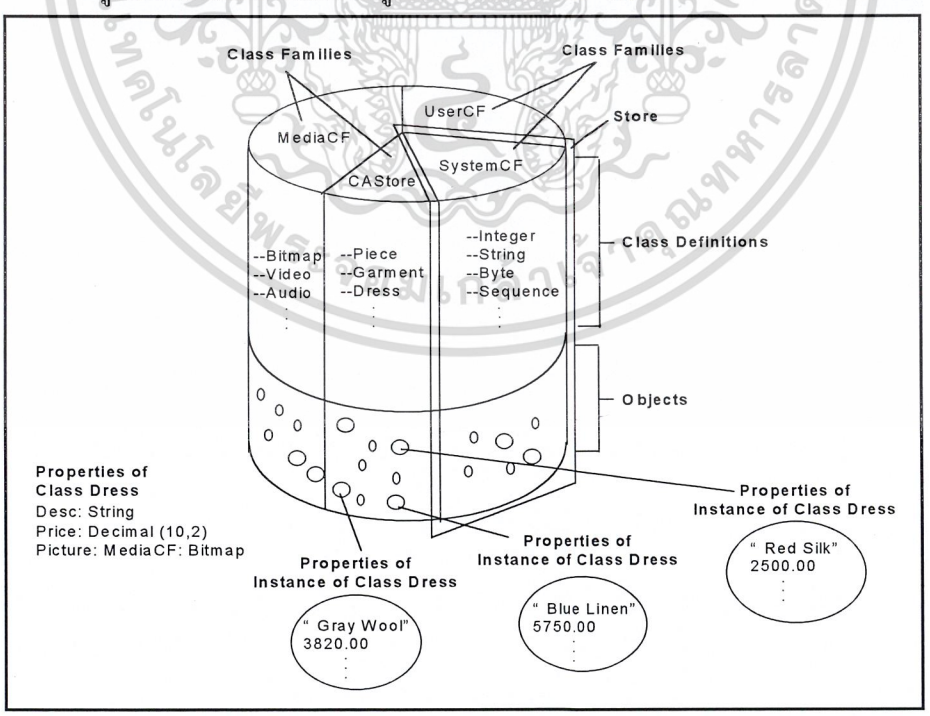
รูปที่ 4.17 แสดงคุณสมบัติของออปเจ็กตัวหนึ่งในฐานข้อมูลของ Jasmine มีลักษณะเป็นไฟล์เสียง



จากรูปที่ 4.17 จะพบว่าใน 1 ออปเจ็กต์จะมีคุณสมบัติมากมายแต่เมื่อเรามองลึกลงถึงเฉพาะตัวไฟล์ข้อมูลเสียงนี้จะพบว่ามี ID ในการอ้างบนหน้าจอคือ Sounds_audio_8 และมีการเชื่อมโยงในการเก็บข้อมูลในส่วนฐานข้อมูล “media” และมี ID อ้างอิงดังรูปที่4.18 ที่ mediaCF1::1029:2611 เมื่อเราเปิดที่ฐานข้อมูล mediaCFนี้จะพบข้อมูลนี้ถูกเก็บในฐานข้อมูลแบบmedia และต้องการดูภาพรวมก็แสดงได้ดังรูปที่ 4.19



รูปที่4.18 แสดงการเก็บข้อมูลของ Jasmine เป็น OID



รูปที่ 4.19 แสดงลักษณะการเก็บข้อมูลเป็นแบบโมเดล

บทที่ 5
การสร้างโปรแกรมประยุกต์กับระบบฐานข้อมูลเชิงวัตถุ

5.1 ขั้นตอนพื้นฐานในการสร้างโปรแกรมประยุกต์บน Jasmine

การสร้างแอพพิเคชันโดยใช้ความสามารถของโปรแกรมที่เป็นยูทิลิตี้ของจัสมิน(Jasmie engine) สามารถกระทำได้ตามขั้นตอนพื้นฐานของการพัฒนาดังต่อไปนี้

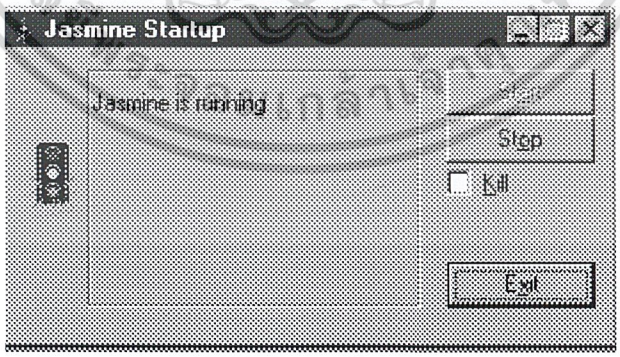
5.1.1 Start Jasmine

เมื่อติดตั้งจัสมินบนเซิร์ฟเวอร์(Server)ที่จะใช้เก็บระบบฐานข้อมูลหรือเครื่องคอมพิวเตอร์ที่เป็นลักษณะ Stand alone แล้วก็จะต้องทำการสตาร์ทเซอร์วิส(Service Start)เพื่อให้สามารถใช้ตัวจัดการระบบฐานข้อมูลได้(ODBMS) ขั้นตอนการเข้าสู่ ODBMS กระทำได้ดังต่อไปนี้

5.1.1.1 ให้คลิ๊กสตาร์ทบนทูลบาร์และรอการเอ็กซ์ซีคิว(Execute)โปรแกรมจัสมินดังรูปที่ 5.1

รูปที่ 5.1 แสดงการรอเพื่อเตรียม running ของจัสมิน

5.1.1.2 หลังจากนั้นชั่วขณะจัสมินจะเริ่มรันโปรแกรมจนเข้าสู่การใช้งานดังรูปที่ 5.2



รูปที่ 5.2 แสดงการ running ของจัสมิน

5.1.1.3 คลิ๊ก Exit เพื่อปิดไดอะล็อค(Dialog)

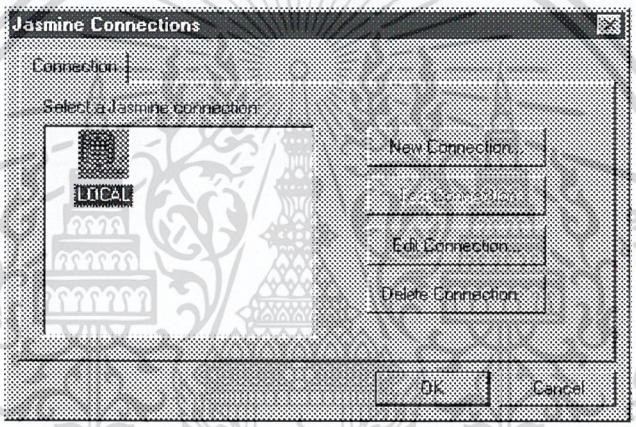
5.1.2 การติดต่อกับฐานข้อมูล

ถ้าติดตั้ง Jasmine ลงบนเครื่องคอมพิวเตอร์เครื่องเดียว(Stand Alone) การติดต่อแบบโลคอล(Local)จะถูกสร้างขึ้นมาเองอัตโนมัติและสามารถใช้ติดต่อกับฐานข้อมูลได้เลย โดยไม่สามารถจะทำการแก้ไขการติดต่อแบบโลคอลนี้ได้และไม่สามารถใช้ชื่อ โลคอล ในการติดต่อกับฐานข้อมูลอื่นได้อีก แต่ถ้าเป็นการติดตั้งบน Server ที่เป็นระบบเครือข่ายแบบ Client – Server จะสามารถกระทำการติดต่อได้หลายทางซึ่งกระทำได้ดังต่อไปนี้

5.1.2.1 การติดต่อผ่านทาง Jasmine Application Development Engine(JADE)

เป็นการติดต่อระหว่าง Application ที่ถูกสร้างขึ้นจากการใช้ยูทิลิตี้ของจัสมินเองกับระบบฐานข้อมูลที่เป็นแบบออบเจกต์ของจัสมิน สามารถกระทำได้ดังนี้

1. คลิ๊กสตาร์ทบนทูลบาร์ จากนั้นเลือก Program,Jasmine และ Jasmine Studio จะปรากฏหน้าต่างของ Jasmine Application Managerและ Jasmine Connection



รูปที่ 5.3 แสดงการเลือกการเชื่อมต่อกับฐานข้อมูล

Dialog ให้ทำการเลือกonnection Dialog เพื่อทำการติดต่อกับระบบฐานข้อมูลดังรูปที่ 5.3 ถ้าติดตั้งบน Server ของระบบเครือข่ายก็จะมีไอคอนต่าง ๆ ที่แสดงขึ้นมาเพื่อให้สามารถเลือกการเชื่อมต่อได้

2. เมื่อเลือกติดต่อฐานข้อมูลได้ตามต้องการแล้วให้คลิ๊ก OK

5.1.2.2 การติดต่อฐานข้อมูลผ่านทาง ODQL Interpreter

การติดต่อฐานข้อมูลแบบนี้จะต้องทำการป้อนคำสั่งของ ODQL เพื่อทำการเอ็กซีคิว(Execute)ผ่านทาง Local ของ Workstation หรือ Remote Server ซึ่งสามารถกระทำได้

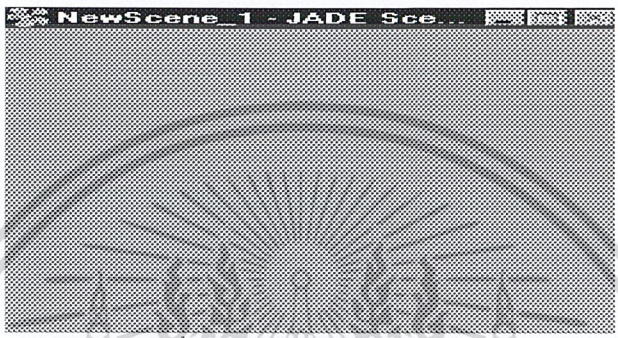
5.1.2.3 การติดต่อฐานข้อมูลผ่านทาง C API

ถ้าใช้โปรแกรมทูล(Tools)จากภายนอกมาสร้างโปรแกรมประยุกต์(Application)ก็จะสามารถติดต่อกับระบบฐานข้อมูลโดยใช้ฟังก์ชันเฉพาะผ่านทาง ODQLคือ odbExecODQL function

5.1.3 การสร้างโปรแกรมแอพพิเคชันใน Application Manager

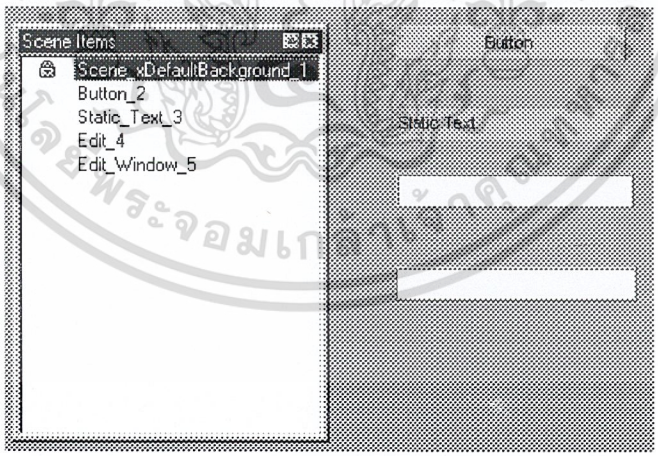
การสร้างแอพพิเคชันนั้นจะต้องเริ่มต้นด้วยการสร้างซิลซ์(Scene)ซึ่งจะทำหน้าที่เปรียบเหมือนกับเพจหรือเป็น GUI กับผู้ใช้ โดยผู้ใช้สามารถจะทำการ drag หัวข้อ ของ Class ,object ,local widget และ Queries มาไว้บนซิลซ์ของผู้ใช้ได้ดังรูปที่ 5.4 และมีขั้นตอนการทำดังนี้

5.1.3.1 จากทูลบาร์ของ Jasmine Application Manager ให้ทำการคลิ๊กไฟล์(File) เลือก New Scene จาก นั้นจะได้ Scene ว่าง ๆ



รูปที่ 5.4 แสดงการสร้าง Scene

5.1.3.2 จาก Application Manager ให้ทำการคลิ๊ก Database Administrator เพื่อเข้าสู่ Class Browser และเลือก Widget ใน Classes Family เพื่อต้องการสร้างออบเจกต์ที่เป็นปุ่มข้อความหรือเป็นออบเจกต์วินโดว์ตามความต้องการ จาก Class Browser Windows เลือกปุ่มใน Local Widget แล้ว Drag ไปวางไว้ใน Scene ที่ว่างตามความต้องการและสามารถแสดง Scene Item เพื่อดูว่ามี

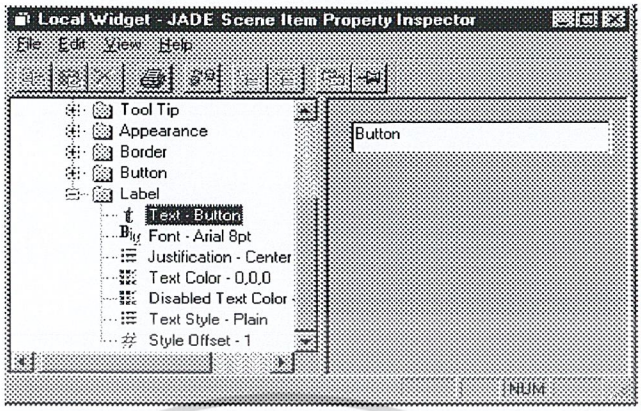


รูปที่ 5.5 แสดงการสร้างออบเจกต์บน Scene

ออบเจกต์อะไรบ้างบน Scene ที่สร้างขึ้นดังรูปที่ 5.5 และสามารถเลือกกำหนดค่า Properties ต่อไป

5.1.4 การกำหนดค่า Properties ของออบเจกต์

เมื่อสร้างออบเจกต์ได้ตามความต้องการแล้วสามารถทำการแก้ไข Properties ของแต่ละออบเจกต์โดยสามารถกระทำได้ดังนี้ เช่น ต้องการใส่ชื่อตัวอักษรบนออบเจกต์ที่เป็นปุ่มตัวอักษรก็

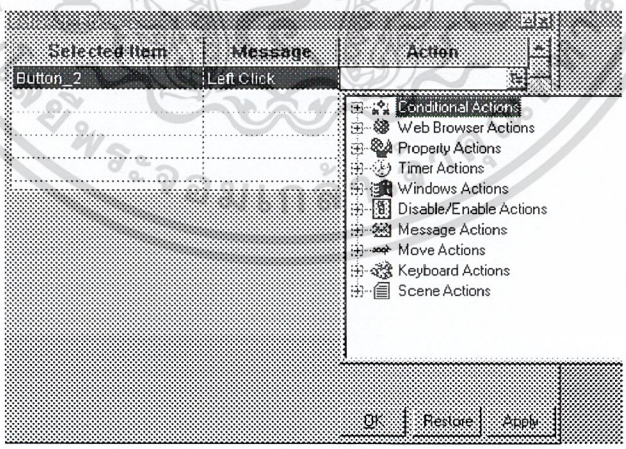


รูปที่ 5.6 แสดง การแก้ไข properies ของออบเจกต์

สามารถกระทำได้โดยใช้เม้าท์(Mouse)คลิ๊กที่ Label Properties และ Text Attribute จากนั้นก็สามารถแก้ไขตัวอักษรบนปุ่มออบเจกต์ได้ดังรูปที่ 5.6 เมื่อแก้ไขตามความต้องการแล้วให้ทำการ Save

5.1.5 การกำหนดค่า Behavior ของออบเจกต์

เมื่อทำการแก้ไข properies ของแต่ละออบเจกต์ตามความต้องการแล้วจะต้องทำการให้ค่า behavior ของแต่ละออบเจกต์ด้วยเช่นกัน โดยการกำหนดค่าต่าง ๆ ใน behavior จะต้องมีความหมายที่สัมพันธ์กันเพื่อใช้ในการส่งค่า Message ของออบเจกต์ เช่น ถ้าต้องการให้ออบเจกต์หนึ่งรับค่าของ message เข้ามาและกระทำการเปลี่ยน Scene จากที่อยู่ไปเป็น Scene อื่นสามารถกระทำได้ดังรูปที่ 5.7



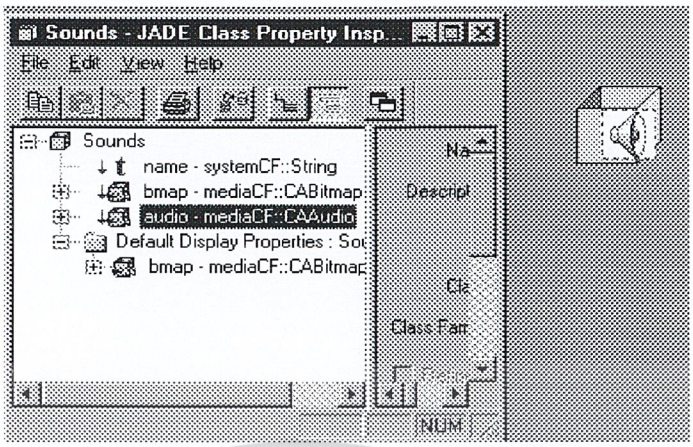
รูปที่ 5.7 แสดงการใส่ค่าของ Behavior แต่ละออบเจกต์

5.1.6 การสร้างมัลติมิเดียให้ออบเจกต์

ในกรณีที่ต้องการสร้างให้ออบเจกต์สามารถกระการรับค่า Message จากออบเจกต์อื่นมาและกระทำการเปิดไฟล์เพื่อเล่นไฟล์ของภาพและเสียงสามารถกระทำได้ดังต่อไปนี้

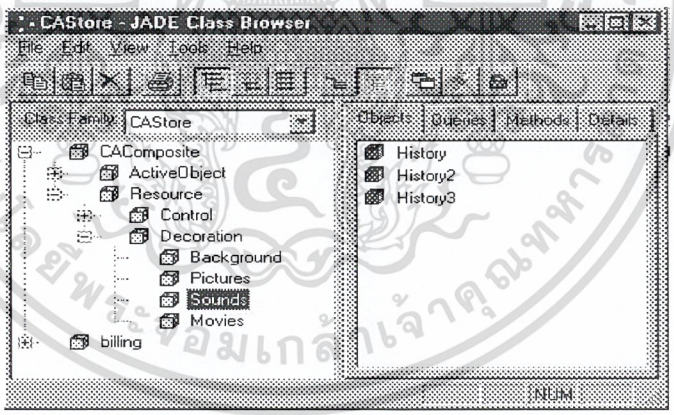
5.1.6.1 ให้ทำการคลิ๊กที่ Jasmine Class Browser แล้วเลือก CAStore,

CAComposite,Resource Object ,Decoration Subobject และ เลือก Sound ดังรูปที่ 5.8



รูปที่ 5.8 แสดงการสร้างออบเจกต์ให้สามารถแสดงผลแบบมัลติมิเดีย

5.1.6.2 จากนั้นให้ Drag subobject sound ดังรูปที่ 5.9 ไปวางไว้ใน Scene ที่สร้างขึ้นจะปรากฏเป็นรูป ลำโพง จากนั้นให้ค่า Behavior ของออบเจกต์ของลำโพงโดยเมื่อออบเจกต์ใด ๆ ที่กำหนดส่งค่าของ Message มาให้ออบเจกต์ของ Sound ซึ่งออบเจกต์ Sound ถูกกำหนดให้มี Method คือการเล่นไฟล์ภาพหรือเสียง ได้ตามต้องการ



รูปที่ 5.9 แสดงการสร้าง Audio Object

5.1.6.3 ถ้าเป็นออบเจกต์ของภาพก็มีขั้นตอนการทำคล้ายกัน แต่ใช้ Subobject ของ Movies

5.1.6.4 ใส่ค่าของ Behavior ภายใน ออบเจกต์ที่ต้องการถ้าเป็นออบเจกต์ของเสียงจะต้องให้กำหนด Play Sound Start ใน Message และกำหนด Sound Action ใน Method ส่วนของออบเจกต์จะกำหนด Play Vedio Start ใน Message และกำหนด Play vedio item content โดยมีสามารถเลือกได้ว่าจะต้องการให้แสดงภาพแบบเต็มจอหรือกำหนดให้มีภาพตรงตามตำแหน่งที่กำหนดก็ได้

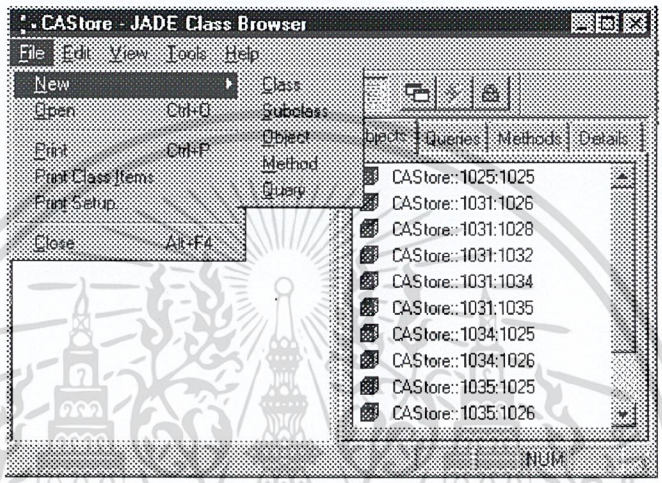
5.1.7. การสร้าง Class ใน Class Browser

เป็นการสร้าง Class ของออบเจกต์ขึ้นมาใหม่ดังรูปที่ 5.10 โดยใช้ยูทิลิตี้(Utility)ของจัสมินซึ่งมีขั้นตอนดังนี้

5.1.7.1 เข้าสู่ Class Browser เลือก CAStore

5.1.7.2 เลือกการสร้างว่าต้องการให้สร้างในส่วนใด เช่น ใน Sub Class ใด ๆ

5.1.7.3 เลือก File , new และ Object เมื่อสร้างก็สามารถแก้ไขหรือ Properties ได้และยังมาสารถดูภาพออบเจกต์ที่ถูกสร้าขึ้นมาได้อีกด้วย



รูปที่ 5.10 แสดงการสร้างออบเจกต์

5.1.8 การทดสอบ Scene ใน Run Mode

ในสภาวะปกติ Application Manager จะอยู่ใน Edit Mode หลังจากทำการสร้าง Scene และจัดวางออบเจกต์ต่าง ๆ ของแอพพิเคชันตามต้องการแล้วรวมทั้งการกำหนดค่าของ Properties,Message และ Method ด้วย เมื่อต้องการทดสอบ Scene ก็จะสามารถกระทำได้ดังนี้

5.1.8.1 เลือกเปิด Scene และเข้าสู่ Application Manager

5.1.8.2 เลือก Tools

5.1.8.3 เลือก Run

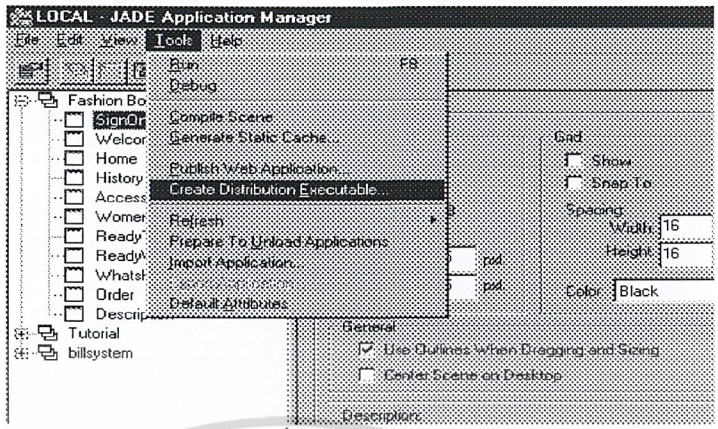
5.1.8 การคอมไพล์แอพพิเคชันโปรแกรม

เนื่องจากการเขียนแอพพิเคชันโปรแกรมโดยอาศัยยูทิลิตี้ของ Jasmine นั้นกระทำได้ง่ายสะดวก และรวดเร็ว ถ้าเป็นโปรแกรมใหญ่จะมี Scene จำนวนมากอยู่ในแอพพิเคชันโปรแกรมเมื่อต้องการคอมไพล์เป็นไฟล์เอ็กซีคิวโดยอาศัยยูทิลิตี้ของ Jusmine ดังรูปที่ 5.11 จะสามารถกระทำได้ดังนี้

5.1.9.1 เลือกเปิด Scene และเข้าสู่ Application Manager

5.1.9.2 เลือก Tools

5.1.9.3 เลือก Cerate Distribution Exceutable



รูปที่ 5.11 แสดงการสร้าง Execute Program

5.2 การสร้างโปรแกรม CAStore

จากการศึกษาของผู้จัดทำเกี่ยวกับการทำงานของ ODBMS(Jasmine) จึงได้เขียนโปรแกรมขึ้นมาทดสอบการทำงานของระบบจัดการฐานข้อมูล ซึ่งระบบที่เขียนขึ้นมานี้จะเป็นระบบการเก็บข้อมูลของลูกค้าจากการขายสินค้าประเภทเสื้อผ้า รองเท้า กระเป๋า โดยผู้ใช้โปรแกรมสามารถตรวจสอบลักษณะของเสื้อผ้ารวมทั้งราคา การสั่งซื้อของลูกค้า โดยโปรแกรมนั้นจะมุ่งเน้นให้เห็นถึงความสามารถของ ODBMS ที่มีประสิทธิภาพในงานที่มีลักษณะเป็นมัลติมิเดีย เพื่อให้ผู้ใช้สามารถดูภาพของสินค้า รวมทั้งราคาของสินค้าได้ตามความต้องการ นอกจากนี้ยังสามารถสร้างความเพลิดเพลินจากมัลติมิเดียในขณะเลือกราคาของสินค้าได้ การเขียนโปรแกรมในการเข้าถึงข้อมูลในระบบการจัดฐานข้อมูลแบบออบเจกต์(Object – Oriented Databae)นี้สามารถกระทำได้ 2 วิธีด้วยกันคือ แบบแรกเป็นการใช้ยูทิลิตี้ของจัสมินเองโดยสามารถกระทำการเขียน User Interface และคอมไพล์เป็นเอ็กซีคิว(Execute)โปรแกรมได้ แบบที่สองเป็นการใช้โปรแกรม Tools จากภายนอก เช่น Visual Basic, Visual C เป็นต้น ในการทำโครงงานนี้ได้เลือกการเขียนโปรแกรมทดสอบแบบที่ใช้ยูทิลิตี้ของจัสมิน ทั้งนี้เพื่อให้ง่ายต่อความเข้าใจ โดยจะมีโครงสร้างการทำงานดังนี้

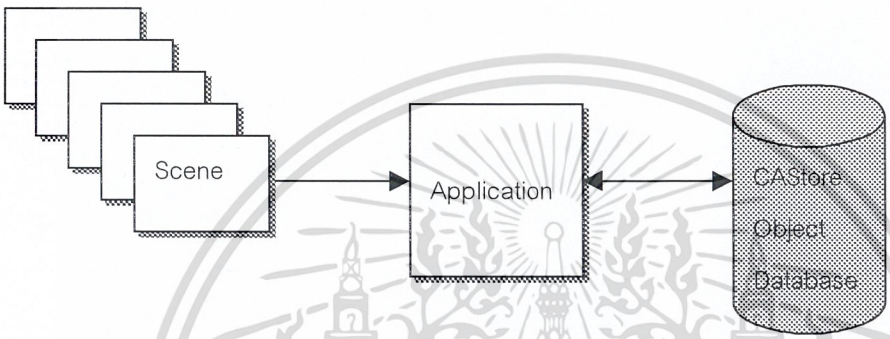
การทำโครงงานนี้จะสามารถแบ่งออกได้เป็น 2 ส่วนคือ

1.ส่วนของการออกแบบข้อมูล

2.ส่วนของการสร้างโปรแกรมประยุกต์(Application)

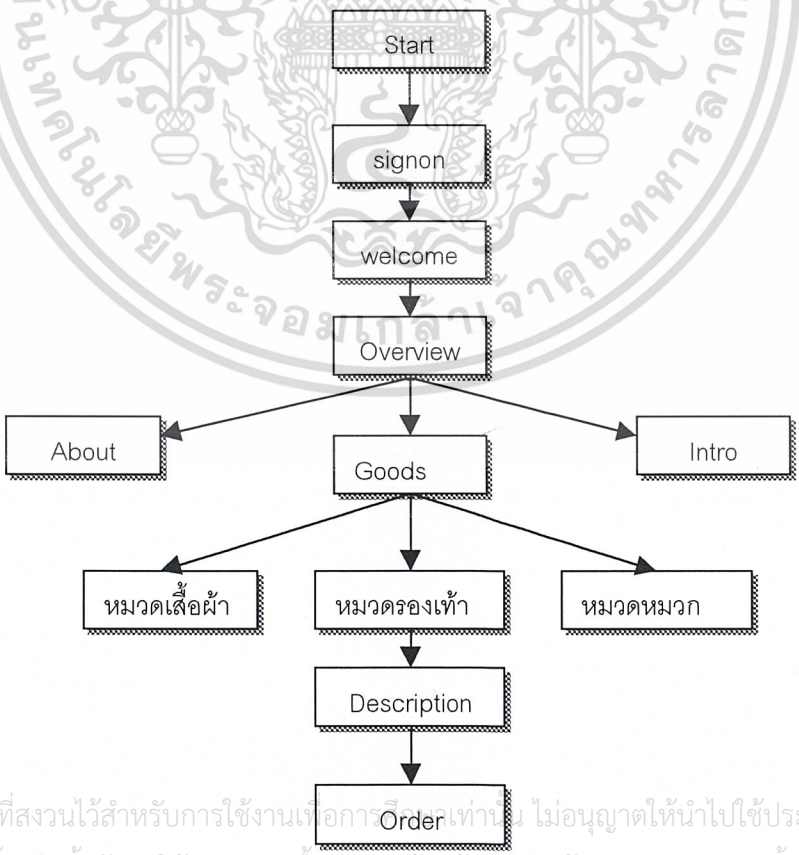
การออกแบบข้อมูลเบื้องต้น

โครงงานนี้เป็นการสร้างระบบฐานข้อมูลแบบ Object – Oriented ที่เกี่ยวข้องกับการสั่งซื้อสินค้าของลูกค้าในร้านค้าที่มีขนาดใหญ่โดยลูกค้าสามารถให้บริการตัวเองเกี่ยวกับการเลือกสินค้าและราคา รวมทั้งยังสามารถสั่งซื้อสินค้าไว้ล่วงหน้าได้ การออกแบบนั้นเป็นเพียงการทดสอบเพื่อศึกษาในเชิงการวิเคราะห์ถึงความสามารถของ ODBMS ในด้านการเก็บข้อมูลแบบออบเจกต์และการทำงานที่สนับสนุนงานในลักษณะมัลติมิเดีย ซึ่งการออกแบบในเบื้องต้นแสดงได้ดังรูปที่ 5.12

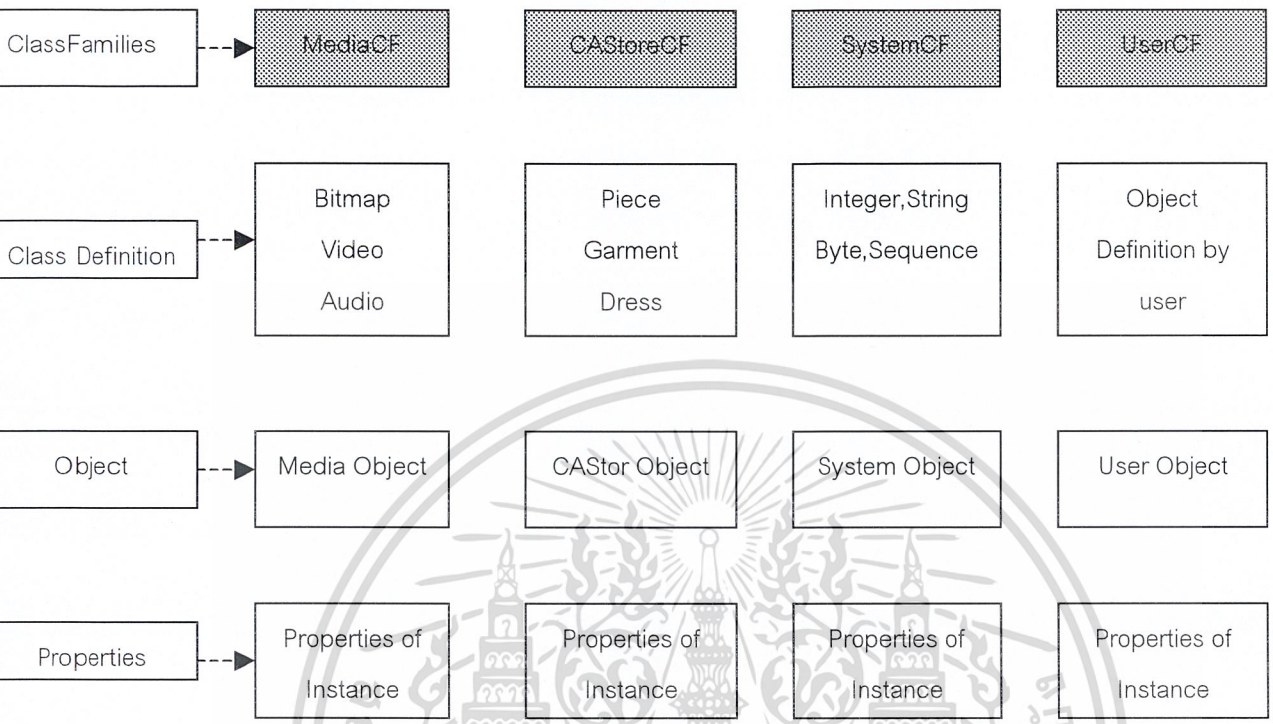


รูปที่ 5.12 แสดงการลักษณะทั่วไปของโปรแกรม CAStore

จากการใช้ยูทิลิตี้ของจัสมินจะสร้าง Scene ออกตามโครงสร้างต่อไปนี้



ภายในระบบฐานข้อมูลแบบออบเจกต์ดาต้าเบสของโครงงานนี้ประกอบประกอบด้วย 4 ส่วนใหญ่ๆ ซึ่งแสดงได้ดังรูปที่ 5.13 ดังต่อไปนี้



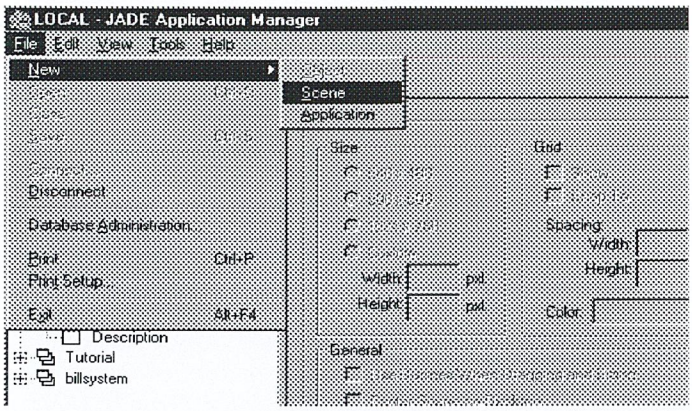
รูปที่ 5.13 แสดงการเก็บข้อมูลของระบบฐานข้อมูลในโปรแกรม CAStore

การสร้างโปรแกรมประยุกต์(Application Program)

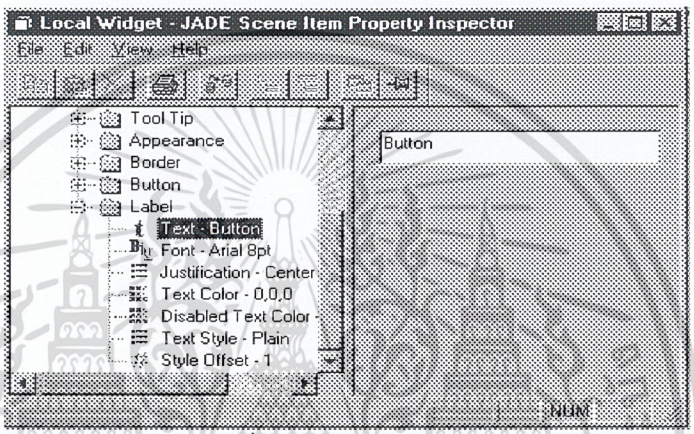
การออกแบบ Scene

การออกแบบ Scene ของทุก ๆ ซิลซ์จะมีลักษณะคล้าย ๆ กันเพียงแต่ว่า แต่ละซิลซ์จะมีออบเจกต์มากน้อยที่แตกต่างกันตามลักษณะของการใช้งาน ซึ่งมีขั้นตอนการออกแบบมีดังต่อไปนี้

1. จาก Jasmine Application Manager เลือกไฟล์และเลือก New Scene ดังรูปที่ 5.14
2. ทำการคลิ๊ก Database Administrator icon จาก Application Manager เพื่อเข้าสู่ Class Browser และเลือก Widget จาก Class Families เพื่อสร้างปุ่มข้อความหรือข้อความหรือการทำ Edit Windows ต่าง ๆ โดยการใช้เม้าท์คลิ๊กที่ไอคอนภายใต้ Local Widget และ Drag ไปวางไว้ตามที่ต้องการ ซึ่งสามารถกระทำได้หลายออบเจกต์ใน Scene เดียวกันก็ได้
3. ทำการแก้ไข้ค่า Properties ของแต่ละออบเจกต์หรือแม้กระทั่งการแก้ Properties ของแบคกราวด์ของ Scene ก็ได้ ดังรูปที่ 5.15



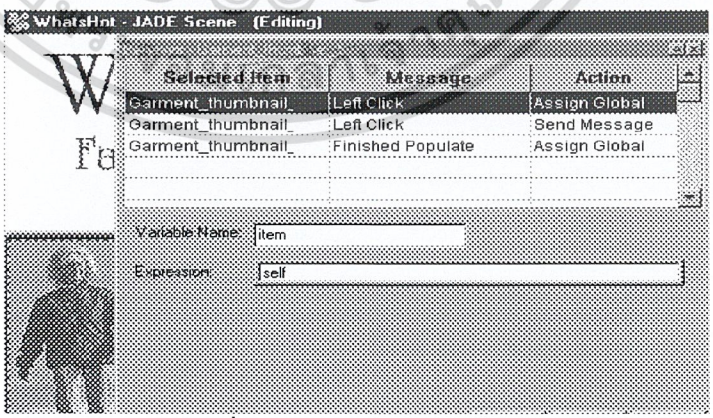
รูปที่ 5.14 แสดงการสร้าง Scene



รูปที่ 5.15 แสดงการแก้ไข Properties

การออกแบบ Behavior

เป็นการกำหนดค่าการกระทำต่าง ๆ ของออบเจกต์รวมทั้งเงื่อนไขที่ออบเจกต์นั้นกระทำเมื่อได้รับเมสเสจจากออบเจกต์อื่น นอกจากนี้ยังสามารถกำหนดค่าการ Assign Global Variable ให้กับออบเจกต์อื่นที่สามารถนำเอาค่านี้ไปใช้งานได้ ดังรูปที่ 5.16



รูปที่ 5.16 แสดงการ Assign Global Variable

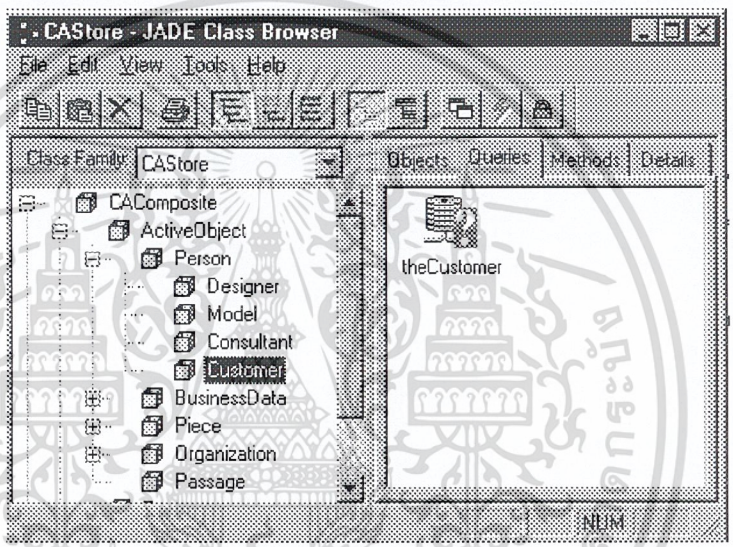
การออกแบบ Query

เป็นการกำหนดเงื่อนไขต่าง ๆ ตามความต้องการของการแสดงผลลัพธ์เมื่อคิวรีนั้นถูกเอ็กซีคิว โดยการกำหนดค่า Behavior ของแต่ละออบเจกต์เมื่อต้องการให้แสดงผลลัพธ์หลังจากออบเจกต์นั้นได้รับเมสเสจจากออบเจกต์อื่นให้ทำการแสดงผล ขั้นตอนการทำคิวรีมีดังต่อไปนี้

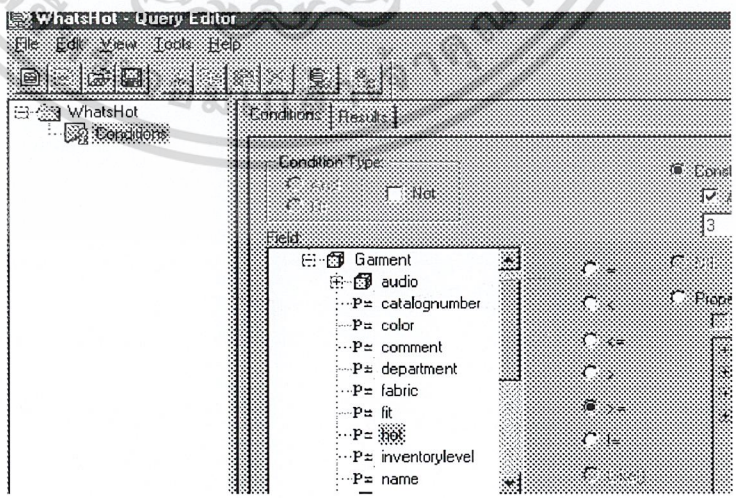
1.ให้ทำการเข้าสู่ Class Browser จาก Application Manager และเลือกหาออบเจกต์ที่ต้องการทำคิวรี เมื่อเลือกได้แล้ว

2. คลิกเม้าท์ที่แท็ปคิวรีทางขวามือ ดังรูปที่ 5.17

3.ให้คลิ๊กไฟล์ใน Application Manager แล้วเลือก New และ Queries จากนั้นโปรแกรมจะเข้าสู่ Queries Editor ให้ทำการเปลี่ยนชื่อได้ตามความต้องการ และต้องกำหนดเงื่อนไขของคิวรีในออบเจกต์นั้นว่าต้องการเงื่อนไขอย่างไร ดังรูปที่ 5.18



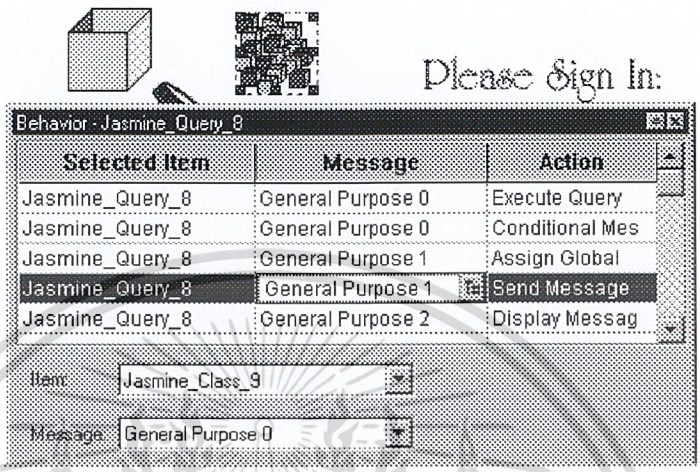
รูปที่ 5.17 แสดงการสร้าง Queries



รูปที่ 5.18 การกำหนดเงื่อนไขภายใน Quires

4.จากนั้นให้คลิ๊กที่คิวรีไอคอนภายในแทปคิวรีและทำการ Drag ไปวางไว้ใน Scene ที่ต้องการแสดงผลลัพธ์จากคิวรีจะได้เป็นรูป Query Anchor บน Scene ดังรูปที่ 5.19

5.ทำการกำหนดค่า Behavior ให้กับออบเจกต์ที่ต้องการเอ็กซ์ซีคิวคิวรีเมื่อได้รับเมสเสจจากออบเจกต์อื่น



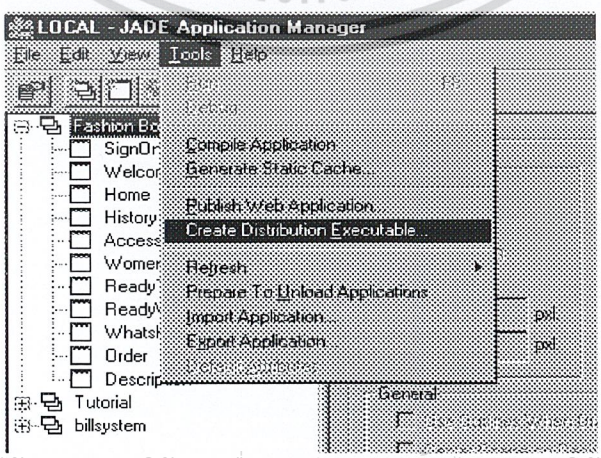
รูปที่ 5.19 แสดงการกำหนดค่า Behavior ที่กระต่อคิวรี

การสร้างโปรแกรมเอ็กซ์ซีคิว

เมื่อออกแบบโปรแกรมโดยใช้ยูทิลิตี้ของจัสมินและกำหนดค่าต่าง ๆ ของแต่ละ scene แล้วให้ทำการทดสอบการทำงานของแต่ละ Scene ก่อนโดยการ Run จาก Application Manager เพื่อให้แน่ใจว่าในแต่ละ Scene นั้นสามารถกระทำตามเงื่อนไขหรือตาม Behavior ที่กำหนดแล้วหรือไม่ เมื่อทำการตรวจสอบจนครบแล้วให้ทำการคอมไพล์ Application ที่สร้างไว้ใน Application Manager ผลที่ได้จากการคอมไพล์จะเป็นไฟล์ที่สามารถเอ็กซ์ซีคิวได้ การคอมไพล์จะมีขั้นตอนดังนี้

1.ทำการเปิดแอพพลิเคชันที่สร้างไว้และเลือก Tools จาก Application Manager

2.เลือก Create distribution Exceutable ดังรูปที่ 5.20



รูปที่ 5.20 แสดงการสร้างโปรแกรมเอ็กซ์ซีคิว

บทที่ 6
บทวิจารณ์และสรุป

6.1 สรุปผลการทำงานโครงงาน

ในการทำโครงงาน สามารถเรียนรู้ได้ตามวัตถุประสงค์ที่ได้กำหนดคือ

1. ศึกษาหลักการของแบบจำลองเชิงวัตถุ (Object – Oriented Model) ว่ามีการทำงานเช่นไรและมีคุณลักษณะอย่างไร
2. ศึกษาการทำงานของ RDBMS(INTERBASE) เปรียบเทียบกับ OODBMS(JASMINE)
3. ทำการสร้างแอพพลิเคชันการออกบิลสำหรับร้านค้าแบบง่าย ๆ เพื่อใช้ทดสอบคุณสมบัติ Object – Oriented Model ของ Jasmine
4. ทำให้เข้าใจการเก็บข้อมูลด้วยเทคโนโลยีของ OODBMS
5. ได้ศึกษาถึงข้อจำกัดบางประการของ OODBMS(Jasmine)

6.2 ปัญหาที่เกิดขึ้น

ปัญหาที่เกิดในการทำโครงงานนี้คือ

1. ปัญหาในการทำ Report ของ RDBMS (INTERBASE)
   ซึ่งเป็นปัญหาของการทำ Report โดยให้สามารถรายงานผลลัพธ์ที่ได้ออกมาเป็นสองฝั่งคือ ต้องการให้ฝั่งซ้าย เป็นรายการส่งสินค้า และฝั่งขวาเป็นรายการคืนสินค้า จากการทดสอบพบว่าเกิดจากการใช้ Tools ในการเขียนโปรแกรมที่ไม่ค่อยสนับสนุนการทำ Report มากนัก จึงได้พยายามแก้ปัญหาและใช้เทคนิคดังต่อไปนี้

   1.1 การแปลงค่า String -> Float,Interget

   ```
   atof (Edit1->Text.c_str());
   se = atoi(COUNTSE->Text.c_str());
   ```

   1.2 การตั้งค่า Parameter ให้กับ Query

   ค่าใน Query = select * from table where x = _parameter

   เราจะส่งค่าได้โดย

   ```
   INXBILL_Q->Params->Items[0]->AsString = DAY1->Text;
   ```

   1.3 การอ่านค่าจาก Query ลงไปสู่ Text File

   ```
   //============for send Bill save to file "Send.db"
   TTable *myOutPut = new TTable(this);
   TBatchMove *myBatch = new TBatchMove(this);
   SAV_SE_Q->Close();
   SAV_SE_Q->Params->Items[0]->AsString  = IDS->Text;
   SAV_SE_Q->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
   SAV_SE_Q->Params->Items[2]->AsString = YEARS->Text;
   SAV_SE_Q->ExecSQL();
   ```

```
SAV_SE_Q->Open();
myOutPut->TableName = "Send.db";
myBatch->Source = SAV_SE_Q;
myBatch->Destination = myOutPut;
myBatch->Mode = batCopy;
myBatch->Execute();
SAV_SE_Q->Close();
delete myOutPut;
delete myBatch;
```

1.4 การสั่งงาน Query แบบฝังใน Code

```
G_DATAQuery->Close();
G_DATAQuery->SQL->Clear();
G_DATAQuery->SQL->Add("SELECT * FROM GOODS_INFO ");
G_DATAQuery->ExecSQL();
G_DATAQuery->Open();
DataSource4->DataSet = G_DATAQuery;
DBGrid3->DataSource = DataSource4;
```

1.5 การสั่งงานแบบลบทั้ง Table

```
DEL_ALL->Close();
DEL_ALL->ExecSQL();
DEL_ALL->Close();
DEL_ENTRY->Close();
DEL_ENTRY->ExecSQL();
DEL_ENTRY->Close();
DEL_ALL->SQL->Clear();
DEL_ALL->SQL->Add("DELETE FROM BILLING");
DEL_ALL->ExecSQL();
DEL_ALL->Close();
```

1.6 อัลกอริทึมการสร้าง Report แบบ 2 ฝั่ง

1.6.1 ใช้คิวรี่เลือกบิลส่งที่ต้องการใส่ Grid

1.6.2 อ่านค่าทีละแถวมาใส่ Table

1.6.3 Save table ลงไฟล์ Send.db

1.6.4 ใช้คิวรี่เลือกบิลรับที่ต้องการใส่ Grid

1.6.5 อ่านค่าทีละแถวมาใส่ Table

1.6.6 Save table ลงไฟล์ Back.db

1.6.7 เปิดทั้งสองไฟล์ใส่เทเบิลเมื่อต้องการสร้างรายงาน

1.6.8 ทำการ INSERT โดยผ่านคิวรี่ใส่ค่าลงใน Table ใหม่ที่สร้างในฐานข้อมูล

1.6.9 ใช้เทเบิลที่สร้างใหม่นี้ในการคำนวณค่าส่วนลดและผลยอดรวม

1.6.10ใช้ในการสร้างผังสถิติเป็นกราฟเพื่อช่วยผู้บริหารในการตัดสินใจ

2. ปัญหาของ Jasmine เนื่องจากเป็น OODBMS ที่เป็นเพียงชุดทดลองทำให้ไม่สามารถสร้างกลุ่ม Classses ของ Class ตามที่ต้องการได้ แต่ผู้จัดทำโครงงานนี้ได้พยายามออกแบบส่วนของฐานข้อมูลใหม่เพื่อให้สามารถกระทำตามจุดประสงค์ที่ได้วางไว้ โดยการออกแบบจะเป็นส่วนของออบเจกต์ที่จะนำมาใช้ในการสั่งซื้อสินค้าของลูกค้า ซึ่งลูกค้าสามารถตรวจสอบลักษณะของสินค้า รูปแบบ สีของสินค้าที่ต้องการ และสามารถสั่งซื้อได้ จากการศึกษารายละเอียดของ Jasmine พบว่ายังรายละเอียดไม่เพียงพอ โดยเฉพาะรายละเอียดเกี่ยวกับการทำ Queries การแสดงผลแบบ Report การตั้งค่า Behavior ของออบเจกต์ต่าง ๆ การเชื่อมต่อในลักษณะของระบบฐานข้อมูลที่เป็นเครือข่าย(Network)

6.3 แนวทางการพัฒนาเพิ่มเติม

จากการโครงงานนี้ ยังเป็นเพียงการทดสอบทาง Object – Oriented Database อย่างง่าย ๆ ภายใต้ OODBMS(Jusmine) เท่านั้น ยังไม่ได้นำไปประยุกต์กับงานที่สลับซับซ้อนมาก ๆ เช่น งานทางด้านมัลติมิเดียที่ซับซ้อนมาก ๆ การเก็บลายนิ้วมือ งานออกแบบทางด้านอุตสาหกรรม เป็นต้น ดังนั้นถ้ามีการพัฒนาต่ออาจเพิ่มความสามารถให้ใช้กับงานที่มีชนิดของข้อมูลที่ซับซ้อนมากได้ เพื่อประยุกต์ใช้งานจริงต่อไป โดยควรศึกษาเพิ่มเติมเกี่ยวกับ OODBMS และ Tools ที่จะนำมาใช้กับงานในลักษณะของระบบฐานข้อมูลแบบเชิงวัตถุ ซึ่งจากการศึกษาถึงความสามารถของ Jusmine พบว่าสามารถกระการติดต่อกับ Tools อื่น ๆ ได้ นอกจากภาษาซี เช่น Virtual Basic 6.0 เป็นต้น ซึ่งสิ่งที่ควรศึกษาเพิ่มเติมมีดังต่อไปนี้

1. ศึกษาการทำ Queries ของ Jusmine ในกรณีที่มีความซับซ้อนมาก ๆ
2. ศึกษาการจัดทำ Report และ รูปแบบต่าง ๆ ของ Report
3. ศึกษาการตั้งค่าใน Place Holder และ Anchor ต่าง ๆ ของ Jusmine
4. ศึกษาการติดตั้งและการพัฒนาระบบฐานข้อมูลแบบเชิงวัตถุ บนระบบเครือข่าย(Network)
5. ศึกษารายละเอียดในการกำหนดค่าของการส่งเมสเสจ(Message)ของออบเจกต์
6. การพัฒนาโปรแกรมที่ใช้กับระบบข้อมูลที่ซับซ้อนมาก ๆ เช่น การนำเอาเทคโนโลยีระบบฐานข้อมูลแบบเชิงวัตถุ ไปใช้ในการตรวจสอบธนบัตร เป็นต้น
7. ศึกษาการทำ OQL จาก Tools ต่าง ๆ ที่ต้องการจะติดต่อกับฐานข้อมูลที่ถูกจัดการด้วย Jusmine

6.4 แนวทางการนำไปประยุกต์ใช้งาน

จากการได้ศึกษาระบบจัดการฐานข้อมูล(Jusmine) พบว่าสามารถนำระบบข้อมูลที่ถูกจัดการด้วยตัวจัดการนี้มาใช้กับระบบอินเตอร์เน็ตได้ โดย Jusmine จะมีออปชั่นพิเศษที่ให้ผู้ใช้สามารถเชื่อมต่อกับระบบฐานข้อมูลในอินเตอร์เน็ต และจากโครงงานนี้เป็นการสร้างระบบฐานข้อมูลที่ใช้ในการให้ลูกค้าสามารถสั่งซื้อสินค้าได้ล่วงหน้าโดยลูกค้าสามารถเลือกรูปแบบ ขนาด สี และราคา ของสินค้าได้เองตามความต้องการ ดังนั้นการนำไปประยุกต์ใช้งานก็อาจจะใช้ในร้านค้าขนาดใหญ่หรือห้างสรรพสินค้าที่มีสินค้าจำนวนมาก โดยการทำเว็ปไซต์ (Web Site)ของร้านค้าหรือห้างสรรพสินค้านั้น ๆ และติดต่อกับระบบฐานข้อมูลที่เป็นสินค้า เพื่อให้ลูกค้าสั่งซื้อได้ตามความต้องการ นอกจากนี้ยังสร้างความสะดวกในการตรวจสอบรายการสินค้ารวมทั้งรายการทางด้านบัญชีอีกด้วย

บรรณานุกรม

[1] Gio Wiederhold (1998) : " Database and Programming Design" ,Miller Freeman Inc,1998.

[2] John G Hughes (1991) : " Object – Oriented Database", Prentice Hall International Group, 1991.

[3] Bancihon, F. (1988) : " Object – Oriented Database System", Proceeding of the ACM SIGACT-SIGMOD-SIGART Conference on the principle of database system, Austin Texas, May 1988.

[4] CA group (1997) : " Introduction and Tutorial of OODBMS (Jusmine) ", Computer Associates Inc, 1997.

[5] Agrawal, R. and N. Gehani (1989) : " ODE (Object Database and Environment) ", The language and data model . Proc. ACM SIGMOD,Protland,OR, June 1989.

[6] Alashqur, A., S. So, and H. Lam (1989) : " OQL (Object Query Language)", for mainipulating object – oriented database, Amterdam, pp 433-442, 1989.

[7] Atwood, T. ODMG – 93(1993) : " The Object – DBMS standard " , object magazine , 3(3) pp 37 – 44 , September 1993.

ภาคผนวก

Source Code of Billing Program

```
//---------------------------------------------------------------------------
#include <vcl\vcl.h>
#include <math.h>
#pragma hdrstop

#include "Main.h"
#include "POP_INSERT.h"
#include "POP_DELETE.h"
#include "POP_EDIT_EMP.h"
#include "REP_EMP_ALL.h"
#include "REP_EMP_PRIVATE.h"
#include "POP_GOODS_IN.h"
#include "POP_GOODS_DELETE.h"
#include "POP_GOODS_EDIT.h"
#include "REP_GOODS_ALL.h"
#include "POP_BILL_INSERT.h"
#include "POP_BILL_DELETE.h"
#include "POP_BILL_EDIT.h"
#include "REP_BILL0.h"
#include "POP_BB_INSERT.h"
#include "POP_BB_DELETE.h"
#include "POP_BB_EDIT.h"
#include "PrgAbout.h"
#include "POP_DEL_ALL.h"
#include "warning.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
        : TForm(Owner)
```

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
        Application->OnHint = ShowHint;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ShowHint(TObject *Sender)
{
        StatusLine->SimpleText = Application->Hint;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditUndo(TObject *Sender)
{
        //---- Add code to perform Edit Undo ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditCut(TObject *Sender)
{
        //---- Add code to perform Edit Cut ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditCopy(TObject *Sender)
{
        //--- Add code to perform Edit Copy ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::EditPaste(TObject *Sender)
{
        //---- Add code to perform Edit Paste ----
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpContents(TObject *Sender)
```

```
{
        Application->HelpCommand(HELP_CONTENTS, 0);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpSearch(TObject *Sender)
{
        Application->HelpCommand(HELP_PARTIALKEY, Longint(""));
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpHowToUse(TObject *Sender)
{
        Application->HelpCommand(HELP_HELPONHELP, 0);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpAbout(TObject *Sender)
{
        //---- show program's About Box ----
   About -> Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TMainForm::ENDPROGRAMClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TMainForm::DISPLAY_BILL_SENDClick(TObject *Sender)
{
```

```
    BILL_SEQuery->SQL->Clear();
    BILL_SEQuery->SQL->Add("SELECT *");
    BILL_SEQuery->SQL->Add("FROM BILLING Billing");
    BILL_SEQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND YEAR =
  :YEAR_ AND BACK = 'ส่ง'");
    BILL_SEQuery->Params->Items[0]->AsString  = ID_SEND->Text;
    BILL_SEQuery->Params->Items[1]->AsInteger  = MONTH_SEND->ItemIndex+1;
    BILL_SEQuery->Params->Items[2]->AsString  = YEAR_SEND->Text;
    BILL_SEQuery->ExecSQL();
    BILL_SEQuery->Active = true;
  }
  //---------------------------------------------------------------------------


  void __fastcall TMainForm::BitBtn4Click(TObject *Sender)
  {
     EMPQuery->Close();
     EMPQuery->SQL->Clear();
     EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
     EMPQuery->SQL->Add("WHERE ID = :ID_");
     EMPQuery->Params->Items[0]->AsString = ID_EMP->Text;
     EMPQuery->ExecSQL();
     EMPQuery->Open();
     DataSource3->DataSet = EMPQuery;
     DBGrid2->DataSource = DataSource3;
  }
  //---------------------------------------------------------------------------


  void __fastcall TMainForm::BitBtn5Click(TObject *Sender)
  {
     EMPQuery->Close();
     EMPQuery->SQL->Clear();
     EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
     EMPQuery->SQL->Add("WHERE FIRSTNAME = :FIRSTNAME_");
```

```
    EMPQuery->Params->Items[0]->AsString = NAME_EMP->Text;
    EMPQuery->ExecSQL();
    EMPQuery->Open();
    DataSource3->DataSet = EMPQuery;
    DBGrid2->DataSource = DataSource3;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn6Click(TObject *Sender)
{
    EMPQuery->Close();
    EMPQuery->SQL->Clear();
    EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
    EMPQuery->SQL->Add("WHERE LASTNAME = :LASTTNAME_");
    EMPQuery->Params->Items[0]->AsString = LASTNAME_EMP->Text;
    EMPQuery->ExecSQL();
    EMPQuery->Open();
    DataSource3->DataSet = EMPQuery;
    DBGrid2->DataSource = DataSource3;
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::DISPLAY_EMP_ALLClick(TObject *Sender)
{
    EMPQuery->Close();
    EMPQuery->SQL->Clear();
    EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
    EMPQuery->ExecSQL();
    EMPQuery->Open();
    DataSource3->DataSet = EMPQuery;
    DBGrid2->DataSource = DataSource3;
}
```

```
//---------------------------------------------------------------------------


void __fastcall TMainForm::INSERT_EMPClick(TObject *Sender)
{
   INSERT_FORM -> Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DELETE_EMPClick(TObject *Sender)
{
   DELETE_FORM -> Show();
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::EDIT_EMPClick(TObject *Sender)
{
 EDIT_FORM->Show();
 EDIT_FORM->EDIT_EMP_ID->Text       = EDIT_FORM->TEMP_EDIT_ID_EMP->Text;
 EDIT_FORM->EDIT_EMP_FIRSTNAME->Text = EDIT_FORM-
>TEMP_EDIT_FIRSTNAME_EMP->Text;
 EDIT_FORM->EDIT_EMP_LASTNAME->Text  = EDIT_FORM-
>TEMP_EDIT_LASTNAME_EMP->Text;
 EDIT_FORM->EDIT_EMP_TELEPHONE->Text = EDIT_FORM-
>TEMP_EDIT_TELEPHONE_EMP->Text;
 EDIT_FORM->EDIT_EMP_INFO->Text     = EDIT_FORM->TEMP_EDIT_INFO_EMP->Text;
}
//---------------------------------------------------------------------------



void __fastcall TMainForm::PRINT_ALL_EMPClick(TObject *Sender)
{
```

```
    REPORT_ALL->QuickRep1->Preview();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::PRINT_PRI_EMPClick(TObject *Sender)
{
    PRIVATE_EMP->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn8Click(TObject *Sender)
{
    GQuery->Close();
    GQuery->Params->Items[0]->AsString = GOODS_SEARCH->Text;
    GQuery->ExecSQL();
    GQuery->Open();
    DataSource4->DataSet = GQuery;
    DBGrid3->DataSource = DataSource4;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::ALL_GOODSClick(TObject *Sender)
{
   G_DATAQuery->Close();
   G_DATAQuery->SQL->Clear();
   G_DATAQuery->SQL->Add("SELECT * FROM GOODS_INFO ");
   G_DATAQuery->ExecSQL();
   G_DATAQuery->Open();
   DataSource4->DataSet = G_DATAQuery;
   DBGrid3->DataSource = DataSource4;
}
//---------------------------------------------------------------------------
```



```
void __fastcall TMainForm::INSERT_GOOD_DATAClick(TObject *Sender)
```

```
{
    INSERT_GOODS->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DELETE_GOODS_DATAClick(TObject *Sender)
{
     DELETE_GOODS_FORM->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::EDIT_GOODS_DATAClick(TObject *Sender)
{
 EDIT_GOODS_FORM->Show();
 EDIT_GOODS_FORM->EDIT_GOODS_GOODS->Text = EDIT_GOODS_FORM->
TEMP_EDIT_GOODS_GOODS->Text;
 EDIT_GOODS_FORM->EDIT_GOODS_INFO->Text  = EDIT_GOODS_FORM->
TEMP_EDIT_GOODS_INFO->Text;
 EDIT_GOODS_FORM->EDIT_GOODS_PRICE->Text = EDIT_GOODS_FORM->
TEMP_EDIT_GOODS_PRICE->Text;
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::PRINT_GOODSClick(TObject *Sender)
{
      REPORT_GOODS->QuickRep1->Preview();
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::BitBtn9Click(TObject *Sender)
{
    NAME->Visible   = true;
```

```
    ADDRESS->Visible = true;
    EDIT_PUB_OK->Visible     = true;
    EDIT_PUB_CANCEL->Visible = true;
    EDIT_PUB_NAME->Visible   = true;
    EDIT_ADDRESS->Visible    = true;
    EDIT_PUB_NAME->Text  = NAME_CONFIG->Text;
    EDIT_ADDRESS->Text   = ADDRESS_CONFIG->Text;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::EDIT_PUB_OKClick(TObject *Sender)
{
  PUBQuery->Close();
  // INSERT VALUE
  PUBQuery->Params->Items[0]->AsString = EDIT_ADDRESS->Text;
  PUBQuery->Params->Items[1]->AsString = EDIT_PUB_NAME->Text;
  PUBQuery->Params->Items[2]->AsString = NAME_CONFIG->Text;
  PUBQuery->ExecSQL();
  PUBQuery->Close();
  NAME->Visible    = false;
  ADDRESS->Visible = false;
  EDIT_PUB_OK->Visible     = false;
  EDIT_PUB_CANCEL->Visible = false;
  EDIT_PUB_NAME->Visible   = false;
  EDIT_ADDRESS->Visible    = false;
  CONFIGURE_TABLE->Active  = false;
  CONFIGURE_TABLE->Active  = true;

}
//---------------------------------------------------------------------------

void __fastcall TMainForm::EDIT_PUB_CANCELClick(TObject *Sender)
{
  NAME->Visible    = false;
```

```
  ADDRESS->Visible = false;
  EDIT_PUB_OK->Visible    = false;
  EDIT_PUB_CANCEL->Visible = false;
  EDIT_PUB_NAME->Visible  = false;
  EDIT_ADDRESS->Visible   = false;
  CONFIGURE_TABLE->Active  = false;
  CONFIGURE_TABLE->Active  = true;

}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn1Click(TObject *Sender)
{
    IN_BILL->ID_TO_IN->Text = ID_SEND->Text;
    IN_BILL->Show();
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::BitBtn2Click(TObject *Sender)
{
    BILL_DELETE->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn3Click(TObject *Sender)
{

  EDIT_BILL->DAY_ED->Text   =  EDIT_BILL->DAY_b->Text;
  EDIT_BILL->YEAR_ED->Text  =  EDIT_BILL->YEAR_b->Text;
  EDIT_BILL->GOODS_ED->Text =  EDIT_BILL->GOODS_b->Text;
  EDIT_BILL->TOTAL_ED->Text =  EDIT_BILL->TOTAL_b->Text;
  EDIT_BILL->Show();
```

```
            // ComboBox1->Items->Strings[ComboBox1->ItemIndex];

  }
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn7Click(TObject *Sender)
{
    REP_BILL->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::DISPLAY_BILL_BACKClick(TObject *Sender)
{
 BILL_BAQuery->SQL->Clear();
 BILL_BAQuery->SQL->Add("SELECT *");
 BILL_BAQuery->SQL->Add("FROM BILLING Billing");
 BILL_BAQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND YEAR =
:YEAR_ AND BACK = 'คืน'");
 BILL_BAQuery->Params->Items[0]->AsString  = ID_BACK->Text;
 BILL_BAQuery->Params->Items[1]->AsInteger  = MONTH_BACK->ItemIndex+1;
 BILL_BAQuery->Params->Items[2]->AsString  = YEAR_BACK->Text;
 BILL_BAQuery->ExecSQL();
 BILL_BAQuery->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn10Click(TObject *Sender)
{
    BACK_BILL->ID_TO_IN->Text = ID_BACK->Text;
    BACK_BILL -> Show();
}
//---------------------------------------------------------------------------
```



```
void __fastcall TMainForm::BitBtn11Click(TObject *Sender)
```

```
{
    BB_DELETE -> Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn12Click(TObject *Sender)
{
   EDIT_BABILL->DAY_ED->Text   =  EDIT_BABILL->DAY_b->Text;
   EDIT_BABILL->YEAR_ED->Text  =  EDIT_BABILL->YEAR_b->Text;
   EDIT_BABILL->GOODS_ED->Text =  EDIT_BABILL->GOODS_b->Text;
   EDIT_BABILL->TOTAL_ED->Text =  EDIT_BABILL->TOTAL_b->Text;
   EDIT_BABILL -> Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Bill1Click(TObject *Sender)
{
    PageControl1->ActivePage = TabSheet1;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Bill4Click(TObject *Sender)
{
  PageControl1->ActivePage = TabSheet2;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N4Click(TObject *Sender)
{
   PageControl1->ActivePage = TabSheet3;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N5Click(TObject *Sender)
```

```
{
    PageControl1->ActivePage = TabSheet4;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N6Click(TObject *Sender)
{
    PageControl1->ActivePage = TabSheet5;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N31Click(TObject *Sender)
{
    IN_BILL->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N35Click(TObject *Sender)
{
    REP_BILL->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N8Click(TObject *Sender)
{
     BACK_BILL -> Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N2Click(TObject *Sender)
{
    PageControl1->ActivePage = TabSheet3;
    EMPQuery->Close();
    EMPQuery->SQL->Clear();
```

```
    EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
    EMPQuery->ExecSQL();
    EMPQuery->Open();
    DataSource3->DataSet = EMPQuery;
    DBGrid2->DataSource = DataSource3;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N13Click(TObject *Sender)
{
    INSERT_FORM -> Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N16Click(TObject *Sender)
{
    PRIVATE_EMP->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N17Click(TObject *Sender)
{
    REPORT_ALL->QuickRep1->Preview();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N9Click(TObject *Sender)
{
    PageControl1->ActivePage = TabSheet4;
    G_DATAQuery->Close();
    G_DATAQuery->SQL->Clear();
    G_DATAQuery->SQL->Add("SELECT * FROM GOODS_INFO ");
    G_DATAQuery->ExecSQL();
    G_DATAQuery->Open();
```

```
    DataSource4->DataSet = G_DATAQuery;
    DBGrid3->DataSource = DataSource4;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N28Click(TObject *Sender)
{
    REPORT_GOODS->QuickRep1->Preview();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N19Click(TObject *Sender)
{
    INSERT_GOODS->Show();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N7Click(TObject *Sender)
{
    PageControl1->ActivePage = TabSheet5;
    NAME->Visible    = true;
    ADDRESS->Visible = true;
    EDIT_PUB_OK->Visible     = true;
    EDIT_PUB_CANCEL->Visible = true;
    EDIT_PUB_NAME->Visible   = true;
    EDIT_ADDRESS->Visible    = true;
    EDIT_PUB_NAME->Text  = NAME_CONFIG->Text;
    EDIT_ADDRESS->Text   = ADDRESS_CONFIG->Text;
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::Bill3Click(TObject *Sender)
{
    POP_DEL_CFG->Show();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::N27Click(TObject *Sender)
{
  warn -> Show();
}
//---------------------------------------------------------------------------


void __fastcall TMainForm::BitBtn13Click(TObject *Sender)
{
  PRIVATE_SUM->Close();
  PRIVATE_SUM->Params->Items[0]->AsString  = ID_SUM->Text;
  PRIVATE_SUM->ExecSQL();
  PRIVATE_SUM->Open();
  //----------------------------PINE CHART FOR SELECT ID
  PRI_GOODS->Close();
  PRI_GOODS->Params->Items[0]->AsString  = ID_SUM->Text;
  PRI_GOODS->ExecSQL();
  PRI_GOODS->Open();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn15Click(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn14Click(TObject *Sender)
{
    SUM_ALL_GBACK->Close();
    SUM_ALL_GOODS->Close();
```

```
    SUM_ALL_GOODS->ExecSQL();
    SUM_ALL_GOODS->Open();
    SUM_ALL_GBACK->ExecSQL();
    SUM_ALL_GBACK->Open();
}
//---------------------------------------------------------------------------

void __fastcall TMainForm::BitBtn16Click(TObject *Sender)
{ float tik,tikb,tnik,tnikb,tsum,tdik,tdnik;;
 // tik = temp sum IK send , tikb = temp sum IK back
 // tnik  = temp sum Non IK , tnikb = temp sum non IK back
 // tdik = temp dec % of IK , tdnik = temp dec % of non IK
   DELETE_SUM->Close();
   DELETE_SUM->ExecSQL();
   ID_LIST->First();
   while (!ID_LIST->Eof)
    {
     tdik  = atof(DIK->Text.c_str());
     tdik  = (100 - tdik)/100;
     tdnik  = atof(DNIK->Text.c_str());
     tdnik  = (100 - tdnik)/100;
     LIST_NIK->Close();
     LIST_NIK->Params->Items[0]->AsString  = S_ID->Text;
     LIST_NIK->Params->Items[1]->AsInteger  = SCOM_MON->ItemIndex+1;
     LIST_NIK->Params->Items[2]->AsString  = SCOM_YEAR->Text;
     LIST_NIK->ExecSQL();
     LIST_NIK->Open();
     tnik = atof(SUMNIKS->Text.c_str());
  //-----------------------------------------
     LIST_NIKB->Close();
     LIST_NIKB->Params->Items[0]->AsString  = S_ID->Text;
     LIST_NIKB->Params->Items[1]->AsInteger  = SCOM_MON->ItemIndex+1;
     LIST_NIKB->Params->Items[2]->AsString  = SCOM_YEAR->Text;
     LIST_NIKB->ExecSQL();
```

```
 LIST_NIKB->Open();
 tnikb = atof(SUMNIKB->Text.c_str());
//----------------------------------------
 LIST_IK->Close();
 LIST_IK->Params->Items[0]->AsString  = S_ID->Text;
 LIST_IK->Params->Items[1]->AsInteger = SCOM_MON->ItemIndex+1;
 LIST_IK->Params->Items[2]->AsString  = SCOM_YEAR->Text;
 LIST_IK->ExecSQL();
 LIST_IK->Open();
 tik = atof(SUMIKS->Text.c_str());
//----------------------------------------
 LIST_IKB->Close();
 LIST_IKB->Params->Items[0]->AsString  = S_ID->Text;
 LIST_IKB->Params->Items[1]->AsInteger = SCOM_MON->ItemIndex+1;
 LIST_IKB->Params->Items[2]->AsString  = SCOM_YEAR->Text;
 LIST_IKB->ExecSQL();
 LIST_IKB->Open();
 tikb = atof(SUMIKB->Text.c_str());
//----------------------------------------
 tnik = tnik - tnikb;
 tnik = tnik * tdnik;
 tik  = tik - tikb;
 tik  = tik * tdik;
 tsum = tik+tnik;
 SUMMER->Text = FloatToStrF(tsum,ffNumber,15,2);
//--------------------INSERT VALUE TO TABLE SUMEMP
 IN_SUM->Close();
 IN_SUM->Params->Items[0]->AsString  = S_ID->Text;
 IN_SUM->Params->Items[1]->AsString  = S_FIRST->Text;
 IN_SUM->Params->Items[2]->AsFloat   = tsum;
 IN_SUM->Params->Items[3]->AsInteger = SCOM_MON->ItemIndex+1;
 IN_SUM->Params->Items[4]->AsString  = SCOM_YEAR->Text;
 IN_SUM->ExecSQL();
 IN_SUM->Close();
```

```
        ID_LIST->MoveBy(1);
      }
      SHOW_ALL->Close();
      SHOW_ALL->ExecSQL();
      SHOW_ALL->Open();
  }
  //---------------------------------------------------------------------------

  void __fastcall TMainForm::BitBtn17Click(TObject *Sender)
  {
       SUM_ALL_BK->Close();
       SUM_ALL_EMP->Close();
       SUM_ALL_BK->ExecSQL();
       SUM_ALL_BK->Open();
       SUM_ALL_EMP->ExecSQL();
       SUM_ALL_EMP->Open();
  }
  //---------------------------------------------------------------------------




  //---------------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_BB_INSERT.h"
  #include "Main.h"
  //---------------------------------------------------------------------------
  #pragma package(smart_init)
  #pragma resource "*.dfm"
  TBACK_BILL *BACK_BILL;
  //---------------------------------------------------------------------------
  __fastcall TBACK_BILL::TBACK_BILL(TComponent* Owner)
```

```
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TBACK_BILL::BitBtn1Click(TObject *Sender)
{
  BBILLQuery->Close();
  // INSERT VALUE
  BBILLQuery->Params->Items[0]->AsInteger =  atoi(ENTRY_CONFIG->Text.c_str());
  UP_ENTRYQuery->Close();
  UP_ENTRYQuery->Params->Items[0]->AsInteger = (1+atoi(ENTRY_CONFIG->Text.c_str()));
  UP_ENTRYQuery->Params->Items[1]->AsInteger = atoi(ENTRY_CONFIG->Text.c_str());
  UP_ENTRYQuery->ExecSQL();
  UP_ENTRYQuery->Close();
  MainForm->CONFIGURE_TABLE->Active = false;
  MainForm->CONFIGURE_TABLE->Active = true;
  BBILLQuery->Params->Items[1]->AsString  = ID_TO_IN->Text;
  BBILLQuery->Params->Items[2]->AsString  = GOODSBILL->Text;
  BBILLQuery->Params->Items[3]->AsFloat   = atof(TOTALBILL->Text.c_str());
  BBILLQuery->Params->Items[4]->AsString  = BACK_B->Text;
  BBILLQuery->Params->Items[5]->AsString  = DAYBILL->Text;
  BBILLQuery->Params->Items[6]->AsInteger = MONTHBILL->ItemIndex+1;
  BBILLQuery->Params->Items[7]->AsString  = YEARBILL->Text;
  BBILLQuery->ExecSQL();
  BBILLQuery->Close();
  BACK_BILL->Close();
// Update Screen for refresh data new entry*/
  MainForm->BILL_BAQuery->SQL->Clear();
  MainForm->BILL_BAQuery->SQL->Add("SELECT *");
  MainForm->BILL_BAQuery->SQL->Add("FROM BILLING Billing");
  MainForm->BILL_BAQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND
YEAR = :YEAR_ AND BACK = 'คืน'");
  MainForm->BILL_BAQuery->Params->Items[0]->AsString  = ID_TO_IN->Text;
  MainForm->BILL_BAQuery->Params->Items[1]->AsInteger = MONTHBILL->ItemIndex+1;
```

```
    MainForm->BILL_BAQuery->Params->Items[2]->AsString  = YEARBILL->Text;
    MainForm->BILL_BAQuery->ExecSQL();
    MainForm->BILL_BAQuery->Active = true;
  }
  //---------------------------------------------------------------------------

  void __fastcall TBACK_BILL::BitBtn2Click(TObject *Sender)
  {
     BACK_BILL->Close();
  }
  //---------------------------------------------------------------------------




  //---------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_BB_EDIT.h"
  #include "Main.h"
  //---------------------------------------------------------------------
  #pragma package(smart_init)
  #pragma resource "*.dfm"
  TEDIT_BABILL *EDIT_BABILL;
  //---------------------------------------------------------------------
  __fastcall TEDIT_BABILL::TEDIT_BABILL(TComponent* Owner)
    : TForm(Owner)
  {
    DAY_ED->Text   =  DAY_b->Text;
    YEAR_ED->Text  =  YEAR_b->Text;
    GOODS_ED->Text =  GOODS_b->Text;
    TOTAL_ED->Text =  TOTAL_b->Text;
```

```
}
//---------------------------------------------------------------------------
void __fastcall TEDIT_BABILL::BitBtn1Click(TObject *Sender)
{
  ED_BBQuery->Close();
  // INSERT VALUE
  ED_BBQuery->Params->Items[0]->AsString  = GOODS_ED->Text;
  ED_BBQuery->Params->Items[1]->AsString  = TOTAL_ED->Text;
  ED_BBQuery->Params->Items[2]->AsString  = DAY_ED->Text;
  ED_BBQuery->Params->Items[3]->AsString  = YEAR_ED->Text;
  ED_BBQuery->Params->Items[4]->AsInteger = atoi(DATA_ENTRY->Text.c_str());
  ED_BBQuery->ExecSQL();
  ED_BBQuery->Close();
  //Update Screen
  EDIT_BABILL->Close();
  MainForm->BILL_BAQuery->SQL->Clear();
  MainForm->BILL_BAQuery->SQL->Add("SELECT *");
  MainForm->BILL_BAQuery->SQL->Add("FROM BILLING Billing");
  MainForm->BILL_BAQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND
YEAR = :YEAR_ AND BACK = 'คืน'");
  MainForm->BILL_BAQuery->Params->Items[0]->AsString  = MainForm->ID_BACK->Text;
  MainForm->BILL_BAQuery->Params->Items[1]->AsInteger  = MainForm->MONTH_BACK->
ItemIndex+1;
  MainForm->BILL_BAQuery->Params->Items[2]->AsString  = MainForm->YEAR_BACK->Text;
  MainForm->BILL_BAQuery->ExecSQL();
  MainForm->BILL_BAQuery->Active = true;
}
//---------------------------------------------------------------------------
void __fastcall TEDIT_BABILL::BitBtn2Click(TObject *Sender)
{
  EDIT_BABILL->Close();
}
//---------------------------------------------------
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "POP_GOODS_EDIT.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TEDIT_GOODS_FORM *EDIT_GOODS_FORM;
//---------------------------------------------------------------------------
__fastcall TEDIT_GOODS_FORM::TEDIT_GOODS_FORM(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TEDIT_GOODS_FORM::BitBtn2Click(TObject *Sender)
{
      EDIT_GOODS_FORM -> Close();
}
//---------------------------------------------------------------------------
void __fastcall TEDIT_GOODS_FORM::BitBtn1Click(TObject *Sender)
{
  EDIT_GQuery->Close();
  // INSERT VALUE
  EDIT_GQuery->Params->Items[0]->AsString = EDIT_GOODS_GOODS->Text;
  EDIT_GQuery->Params->Items[1]->AsString = EDIT_GOODS_INFO->Text;
  EDIT_GQuery->Params->Items[2]->AsFloat  = atof(EDIT_GOODS_PRICE->Text.c_str());
  EDIT_GQuery->Params->Items[3]->AsString = TEMP_EDIT_GOODS_GOODS->Text;
  EDIT_GQuery->ExecSQL();
  EDIT_GQuery->Close();
  EDIT_GOODS_FORM->Close();
  // Update Screen for refresh data new entry
```

```
    MainForm->G_DATAQuery->Close();
    MainForm->G_DATAQuery->SQL->Clear();
    MainForm->G_DATAQuery->SQL->Add("SELECT * FROM GOODS_INFO ");
    MainForm->G_DATAQuery->ExecSQL();
    MainForm->G_DATAQuery->Open();
    MainForm->DataSource4->DataSet = MainForm->G_DATAQuery;
    MainForm->DBGrid3->DataSource = MainForm->DataSource4;
  }
  //---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "POP_BILL_DELETE.h"
#include "Main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TBILL_DELETE *BILL_DELETE;
//---------------------------------------------------------------------
__fastcall TBILL_DELETE::TBILL_DELETE(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TBILL_DELETE::BitBtn1Click(TObject *Sender)
{
  DELETE_BILLQuery->Close();
  // INSERT VALUE
```

```
    DELETE_BILLQuery->Params->Items[0]->AsInteger = atoi(DELETE_ENTRY_BILL->Text.c_str
  ());
    DELETE_BILLQuery->ExecSQL();
    DELETE_BILLQuery->Close();
    BILL_DELETE->Close();
  // Update Screen for refresh data new entry*/
    MainForm->BILL_SEQuery->SQL->Clear();
    MainForm->BILL_SEQuery->SQL->Add("SELECT *");
    MainForm->BILL_SEQuery->SQL->Add("FROM BILLING Billing");
    MainForm->BILL_SEQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND
  YEAR = :YEAR_ AND BACK = 'ส่ง'");
    MainForm->BILL_SEQuery->Params->Items[0]->AsString  = MainForm->ID_SEND->Text;
    MainForm->BILL_SEQuery->Params->Items[1]->AsInteger = MainForm->MONTH_SEND->
  ItemIndex+1;
    MainForm->BILL_SEQuery->Params->Items[2]->AsString  = MainForm->YEAR_SEND->Text;
    MainForm->BILL_SEQuery->ExecSQL();
    MainForm->BILL_SEQuery->Active = true;
  }
  //---------------------------------------------------------------------------
  void __fastcall TBILL_DELETE::BitBtn2Click(TObject *Sender)
  {
    BILL_DELETE->Close();
  }
  //---------------------------------------------------------------------------
```

```
  //---------------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_BILL_EDIT.h"
  #include "Main.h"
```

```
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TEDIT_BILL *EDIT_BILL;
//---------------------------------------------------------------------------
__fastcall TEDIT_BILL::TEDIT_BILL(TComponent* Owner)
  : TForm(Owner)
{
  DAY_ED->Text   =  DAY_b->Text;
  YEAR_ED->Text  =  YEAR_b->Text;
  GOODS_ED->Text =  GOODS_b->Text;
  TOTAL_ED->Text =  TOTAL_b->Text;

}
//---------------------------------------------------------------------------
void __fastcall TEDIT_BILL::BitBtn1Click(TObject *Sender)
{
  ED_BILLQuery->Close();
  // INSERT VALUE
  ED_BILLQuery->Params->Items[0]->AsString  = GOODS_ED->Text;
  ED_BILLQuery->Params->Items[1]->AsString  = TOTAL_ED->Text;
  ED_BILLQuery->Params->Items[2]->AsString  = DAY_ED->Text;
  ED_BILLQuery->Params->Items[3]->AsString  = YEAR_ED->Text;
  ED_BILLQuery->Params->Items[4]->AsInteger = atoi(DATA_ENTRY->Text.c_str());
  ED_BILLQuery->ExecSQL();
  ED_BILLQuery->Close();
  //Update Screen
  EDIT_BILL->Close();
  MainForm->BILL_SEQuery->SQL->Clear();
  MainForm->BILL_SEQuery->SQL->Add("SELECT *");
  MainForm->BILL_SEQuery->SQL->Add("FROM BILLING Billing");
  MainForm->BILL_SEQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND
YEAR = :YEAR_ AND BACK = 'ส่ง'");
  MainForm->BILL_SEQuery->Params->Items[0]->AsString  = MainForm->ID_SEND->Text;
```

```
    MainForm->BILL_SEQuery->Params->Items[1]->AsInteger  = MainForm->MONTH_SEND->
  ItemIndex+1;
    MainForm->BILL_SEQuery->Params->Items[2]->AsString  = MainForm->YEAR_SEND->Text;
    MainForm->BILL_SEQuery->ExecSQL();
    MainForm->BILL_SEQuery->Active = true;
  }
  //---------------------------------------------------------------------------
  void __fastcall TEDIT_BILL::BitBtn2Click(TObject *Sender)
  {
    EDIT_BILL->Close();
  }
  //---------------------------------------------------------------------------



  //---------------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_BILL_INSERT.h"
  #include "Main.h"
  //---------------------------------------------------------------------------
  #pragma package(smart_init)
  #pragma resource "*.dfm"
  TIN_BILL *IN_BILL;
  //---------------------------------------------------------------------------
  __fastcall TIN_BILL::TIN_BILL(TComponent* Owner)
    : TForm(Owner)
  {
  }
  //---------------------------------------------------------------------------
  void __fastcall TIN_BILL::BitBtn1Click(TObject *Sender)
  {
    IN_BILLQuery->Close();
    // INSERT VALUE
```

```
    IN_BILLQuery->Params->Items[0]->AsInteger =  atoi(ENTRY_CONFIG->Text.c_str());
    UP_ENTRYQuery->Close();
    UP_ENTRYQuery->Params->Items[0]->AsInteger = (1+atoi(ENTRY_CONFIG->Text.c_str()));
    UP_ENTRYQuery->Params->Items[1]->AsInteger = atoi(ENTRY_CONFIG->Text.c_str());
    UP_ENTRYQuery->ExecSQL();
    UP_ENTRYQuery->Close();
    MainForm->CONFIGURE_TABLE->Active = false;
    MainForm->CONFIGURE_TABLE->Active = true;
    IN_BILLQuery->Params->Items[1]->AsString  = ID_TO_IN->Text;
    IN_BILLQuery->Params->Items[2]->AsString  = GOODSBILL->Text;
    IN_BILLQuery->Params->Items[3]->AsFloat   = atof(TOTALBILL->Text.c_str());
    IN_BILLQuery->Params->Items[4]->AsString  = BACK_B->Text;
    IN_BILLQuery->Params->Items[5]->AsString  = DAYBILL->Text;
    IN_BILLQuery->Params->Items[6]->AsInteger = MONTHBILL->ItemIndex+1;
    IN_BILLQuery->Params->Items[7]->AsString  = YEARBILL->Text;
    IN_BILLQuery->ExecSQL();
    IN_BILLQuery->Close();
    IN_BILL->Close();
  // Update Screen for refresh data new entry*/
    MainForm->BILL_SEQuery->SQL->Clear();
    MainForm->BILL_SEQuery->SQL->Add("SELECT *");
    MainForm->BILL_SEQuery->SQL->Add("FROM BILLING Billing");
    MainForm->BILL_SEQuery->SQL->Add("WHERE ID = :ID_ AND MONTH = :MONTH_ AND
 YEAR = :YEAR_ AND BACK = 'ส่ง' ");
    MainForm->BILL_SEQuery->Params->Items[0]->AsString  = MainForm->ID_SEND->Text;
    MainForm->BILL_SEQuery->Params->Items[1]->AsInteger  = MainForm->MONTH_SEND->
 ItemIndex+1;
    MainForm->BILL_SEQuery->Params->Items[2]->AsString  = MainForm->YEAR_SEND->Text;
    MainForm->BILL_SEQuery->ExecSQL();
    MainForm->BILL_SEQuery->Active = true;
}
//---------------------------------------------------------------------------
```



```
void __fastcall TIN_BILL::BitBtn2Click(TObject *Sender)
```

```
  {
    IN_BILL->Close();
  }
  //---------------------------------------------------------------------------
```

```
  //---------------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_DEL_ALL.h"
  //---------------------------------------------------------------------------
  #pragma package(smart_init)
  #pragma resource "*.dfm"
  TPOP_DEL_CFG *POP_DEL_CFG;
  //---------------------------------------------------------------------------
  __fastcall TPOP_DEL_CFG::TPOP_DEL_CFG(TComponent* Owner)
    : TForm(Owner)
  {
  }
  //---------------------------------------------------------------------------

  void __fastcall TPOP_DEL_CFG::POP_DELClick(TObject *Sender)
  {
      DEL_ALL->Close();
      DEL_ALL->ExecSQL();
      DEL_ALL->Close();
      DEL_ENTRY->Close();
      DEL_ENTRY->ExecSQL();
      DEL_ENTRY->Close();
      DEL_ALL->SQL->Clear();
      DEL_ALL->SQL->Add("DELETE FROM BILLING");
```

```
        DEL_ALL->ExecSQL();
        DEL_ALL->Close();
        POP_DEL_CFG->Close();

}
//---------------------------------------------------------------------

void __fastcall TPOP_DEL_CFG::BitBtn2Click(TObject *Sender)
{
    POP_DEL_CFG->Close();
}
//---------------------------------------------------------------------


//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "POP_DELETE.h"
#include "Main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TDELETE_FORM *DELETE_FORM;
//---------------------------------------------------------------------
__fastcall TDELETE_FORM::TDELETE_FORM(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TDELETE_FORM::BitBtn1Click(TObject *Sender)
{
  DELETEQuery_EMP->Close();
  // INSERT VALUE
```

```
    DELETEQuery_EMP->Params->Items[0]->AsString = DELETE_EMP_ID->Text;
    DELETEQuery_EMP->ExecSQL();
    DELETEQuery_EMP->Close();
    DELETE_FORM->Close();
   // Update Screen for refresh data new entry
    MainForm->EMPQuery->Close();
    MainForm->EMPQuery->SQL->Clear();
    MainForm->EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
    MainForm->EMPQuery->ExecSQL();
    MainForm->EMPQuery->Open();
  }
  //---------------------------------------------------------------------------
  void __fastcall TDELETE_FORM::BitBtn2Click(TObject *Sender)
  {
     DELETE_FORM->Close();
  }
  //---------------------------------------------------------------------------



  //---------------------------------------------------------------------------
  #include <vcl.h>
  #pragma hdrstop

  #include "POP_EDIT_EMP.h"
  #include "Main.h"
  //---------------------------------------------------------------------------
  #pragma package(smart_init)
  #pragma resource "*.dfm"
  TEDIT_FORM *EDIT_FORM;
  //---------------------------------------------------------------------------
  __fastcall TEDIT_FORM::TEDIT_FORM(TComponent* Owner)
    : TForm(Owner)
  {
  }
```

```
//---------------------------------------------------------------------------
void __fastcall TEDIT_FORM::BitBtn1Click(TObject *Sender)
{
  EDITQuery_EMP->Close();
  // INSERT VALUE
  EDITQuery_EMP->Params->Items[0]->AsString = EDIT_EMP_ID->Text;
  EDITQuery_EMP->Params->Items[1]->AsString = EDIT_EMP_FIRSTNAME->Text;
  EDITQuery_EMP->Params->Items[2]->AsString = EDIT_EMP_LASTNAME->Text;
  EDITQuery_EMP->Params->Items[3]->AsString = EDIT_EMP_TELEPHONE->Text;
  EDITQuery_EMP->Params->Items[4]->AsString = EDIT_EMP_INFO->Text;
  EDITQuery_EMP->Params->Items[5]->AsString = TEMP_EDIT_ID_EMP->Text;
  EDITQuery_EMP->ExecSQL();
  EDITQuery_EMP->Close();
  EDIT_FORM->Close();
 // Update Screen for refresh data new entry
  MainForm->EMPQuery->Close();
  MainForm->EMPQuery->SQL->Clear();
  MainForm->EMPQuery->SQL->Add("SELECT * FROM EMPLOYEE ");
  MainForm->EMPQuery->ExecSQL();
  MainForm->EMPQuery->Open();
}
//---------------------------------------------------------------------------
void __fastcall TEDIT_FORM::BitBtn2Click(TObject *Sender)
{
   EDIT_FORM->Close();
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("Prototype_BILL.res");
```

```
USEFORM("Main.cpp", MainForm);
USEFORM("POP_INSERT.cpp", INSERT_FORM);
USEFORM("POP_DELETE.cpp", DELETE_FORM);
USEFORM("POP_EDIT_EMP.cpp", EDIT_FORM);
USEFORM("REP_EMP_ALL.cpp", REPORT_ALL);
USEFORM("REP_EMP_PRIVATE.cpp", PRIVATE_EMP);
USEFORM("POP_GOODS_IN.cpp", INSERT_GOODS);
USEFORM("POP_GOODS_DELETE.cpp", DELETE_GOODS_FORM);
USEFORM("POP_GOODS_EDIT.cpp", EDIT_GOODS_FORM);
USEFORM("REP_GOODS_ALL.cpp", REPORT_GOODS);
USEFORM("POP_BILL_INSERT.cpp", IN_BILL);
USEFORM("POP_BILL_DELETE.cpp", BILL_DELETE);
USEFORM("POP_BILL_EDIT.cpp", EDIT_BILL);
USEFORM("REP_BILL0.cpp", REP_BILL);
USEFORM("REP_BILL_00.cpp", REP_PRE_BILL);
USEFORM("POP_BB_INSERT.cpp", BACK_BILL);
USEFORM("POP_BB_DELETE.cpp", BB_DELETE);
USEFORM("POP_BB_EDIT.cpp", EDIT_BABILL);
USEFORM("E:\Picture\Etc\PrgAbout.cpp", About);
USEFORM("POP_DEL_ALL.cpp", POP_DEL_CFG);
USEFORM("warning.cpp", warn);
USEFORM("inx.cpp", GEN_REP);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
     Application->Initialize();
     Application->CreateForm(__classid(TMainForm), &MainForm);
     Application->CreateForm(__classid(TINSERT_FORM), &INSERT_FORM);
     Application->CreateForm(__classid(TDELETE_FORM), &DELETE_FORM);
     Application->CreateForm(__classid(TEDIT_FORM), &EDIT_FORM);
     Application->CreateForm(__classid(TREPORT_ALL), &REPORT_ALL);
     Application->CreateForm(__classid(TPRIVATE_EMP), &PRIVATE_EMP);
```

```
        Application->CreateForm(__classid(TINSERT_GOODS), &INSERT_GOODS);
        Application->CreateForm(__classid(TDELETE_GOODS_FORM), &DELETE_GOODS_FORM);
        Application->CreateForm(__classid(TEDIT_GOODS_FORM), &EDIT_GOODS_FORM);
        Application->CreateForm(__classid(TREPORT_GOODS), &REPORT_GOODS);
        Application->CreateForm(__classid(TIN_BILL), &IN_BILL);
        Application->CreateForm(__classid(TBILL_DELETE), &BILL_DELETE);
        Application->CreateForm(__classid(TEDIT_BILL), &EDIT_BILL);
        Application->CreateForm(__classid(TREP_BILL), &REP_BILL);
        Application->CreateForm(__classid(TREP_PRE_BILL), &REP_PRE_BILL);
        Application->CreateForm(__classid(TBACK_BILL), &BACK_BILL);
        Application->CreateForm(__classid(TBB_DELETE), &BB_DELETE);
        Application->CreateForm(__classid(TEDIT_BABILL), &EDIT_BABILL);
        Application->CreateForm(__classid(TAbout), &About);
        Application->CreateForm(__classid(TPOP_DEL_CFG), &POP_DEL_CFG);
        Application->CreateForm(__classid(Twarn), &warn);
        Application->CreateForm(__classid(TGEN_REP), &GEN_REP);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "REP_BILL0.h"
```

```
#include "Main.h"
#include "REP_BILL_00.h"
#include "inx.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TREP_BILL *REP_BILL;
//---------------------------------------------------------------------------
__fastcall TREP_BILL::TREP_BILL(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TREP_BILL::BitBtn1Click(TObject *Sender)
{
    float temp,temp2,temp3,dec,temp4,temp5;
  // temp1 = dec IK ,temp2 = sum all temp3 = dec no IK
  // temp4 = sum back IK ,temp5 = sum back non IK
  // dec = IK dec,No IK dec
    REP_PRE_BILL->NAME0->Caption   = NAMES->Text;
    REP_PRE_BILL->ADDRESS0->Caption =  ADDRESSS->Text;
    REP_PRE_BILL->MONTH0->Caption   =  MONTHS->Items->Strings[MONTHS->ItemIndex];
    REP_PRE_BILL->YEAR0->Caption   =  YEARS->Text;
    REP_PRE_BILL->ID0->Caption    =  IDS->Text;
    FIND_NAMEQuery->Close();
    FIND_NAMEQuery->Params->Items[0]->AsString = IDS->Text;
    FIND_NAMEQuery->ExecSQL();
    FIND_NAMEQuery->Open();
    REP_PRE_BILL->FIRSTNAME0->Caption  =  FIRSTNAMES->Text;
    REP_PRE_BILL->LASTNAME0->Caption  =  LASTNAMES->Text;
    REP_BILL->Close();

   // Count back bill IK AND sum it
    SUM_BK_IK->Close();
```

```
SUM_BK_IK->Params->Items[0]->AsString  = IDS->Text;
SUM_BK_IK->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
SUM_BK_IK->Params->Items[2]->AsString  = YEARS->Text;
SUM_BK_IK->ExecSQL();
SUM_BK_IK->Open();
temp4 = atof(SUMBK_IK->Text.c_str());

// Count send bill  IK AND sum it
SUM_IK->Close();
SUM_IK->Params->Items[0]->AsString  = IDS->Text;
SUM_IK->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
SUM_IK->Params->Items[2]->AsString  = YEARS->Text;
SUM_IK->ExecSQL();
SUM_IK->Open();
temp = atof(SUM_SE_IK->Text.c_str());
REP_PRE_BILL->DE->Caption = DEC_PS->Text;
dec  = atof(DEC_PS->Text.c_str());
dec  = dec / 100;
temp = temp - temp4;     // sum ik - dec %
temp = temp * dec;
//------------------- FloatToStrF(temp,ffFixed,15,2);
REP_PRE_BILL->SUM_SE_IK->Caption = FloatToStrF(temp,ffNumber,15,2);
SUM_IK->Close();

// Count back bill and NO IK AND sum it
SUM_BK_NIK->Close();
SUM_BK_NIK->Params->Items[0]->AsString  = IDS->Text;
SUM_BK_NIK->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
SUM_BK_NIK->Params->Items[2]->AsString  = YEARS->Text;
SUM_BK_NIK->ExecSQL();
SUM_BK_NIK->Open();
temp5 = atof(SUMBK_NIK->Text.c_str());
```



```
// Count send bill and NO IK AND sum it
```

```
SUM_NIK->Close();
SUM_NIK->Params->Items[0]->AsString  = IDS->Text;
SUM_NIK->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
SUM_NIK->Params->Items[2]->AsString  = YEARS->Text;
SUM_NIK->ExecSQL();
SUM_NIK->Open();
REP_PRE_BILL->GS->Caption = DEC_GS->Text;
temp3 = atof(SUM_NOIK->Text.c_str());
dec  = atof(DEC_GS->Text.c_str());
dec  = dec / 100;
temp3 = temp3 - temp5;       //sum of non ik - dec %
temp3 = temp3 * dec;         //send to report for no IK dec
REP_PRE_BILL->SUM_DEC_GS->Caption = FloatToStrF(temp3,ffNumber,15,2);
SUM_IK->Close();

// Count send bill
Count_S->Close();
Count_S->Params->Items[0]->AsString  = IDS->Text;
Count_S->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
Count_S->Params->Items[2]->AsString  = YEARS->Text;
Count_S->ExecSQL();
Count_S->Open();
REP_PRE_BILL->COUNT_SEND_TEMP->Caption = COUNT_SE->Text;
REP_PRE_BILL->TOTAL_SE->Caption = SUM_SE->Text;

// Count back Bill ------------------------------
SUM_BACK->Close();
SUM_BACK->Params->Items[0]->AsString  = IDS->Text;
SUM_BACK->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
SUM_BACK->Params->Items[2]->AsString  = YEARS->Text;
SUM_BACK->ExecSQL();
SUM_BACK->Open();
REP_PRE_BILL->COUNT_REC->Caption = SUM_BA->Text;
// ---------Calculate Summary of Report
```

```
    SUM_CAL->Close();
    SUM_CAL->Close();
    SUM_CAL->Params->Items[0]->AsString  = IDS->Text;
    SUM_CAL->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
    SUM_CAL->Params->Items[2]->AsString  = YEARS->Text;
    SUM_CAL->ExecSQL();
    SUM_CAL->Open();
    temp2 = atof(SUM_SE2->Text.c_str());
    temp  = temp2 - temp - temp3 - temp5 - temp4;
    REP_PRE_BILL->SUM_DEC->Caption = FloatToStrF(temp,ffNumber,15,2);
//---------------------------------------------------------------
//          SECTION DETAIL IN REPORT
//---------------------------------------------------------------
   //============for send Bill save to file "Send.db"
    TTable *myOutPut = new TTable(this);
    TBatchMove *myBatch = new TBatchMove(this);
    SAV_SE_Q->Close();
    SAV_SE_Q->Params->Items[0]->AsString  = IDS->Text;
    SAV_SE_Q->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
    SAV_SE_Q->Params->Items[2]->AsString  = YEARS->Text;
    SAV_SE_Q->ExecSQL();
    SAV_SE_Q->Open();
    myOutPut->TableName = "Send.db";
    myBatch->Source = SAV_SE_Q;
    myBatch->Destination = myOutPut;
    myBatch->Mode = batCopy;
    myBatch->Execute();
    SAV_SE_Q->Close();
    delete myOutPut;
    delete myBatch;
   //========For back Bill save to file "Back.db"
    TTable *myOutPut2 = new TTable(this);
    TBatchMove *myBatch2 = new TBatchMove(this);
    SAV_BK_Q->Close();
```

```
    SAV_BK_Q->Params->Items[0]->AsString  = IDS->Text;
    SAV_BK_Q->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
    SAV_BK_Q->Params->Items[2]->AsString  = YEARS->Text;
    SAV_BK_Q->ExecSQL();
    SAV_BK_Q->Open();
    myOutPut2->TableName = "Back.db";
    myBatch2->Source = SAV_BK_Q;
    myBatch2->Destination = myOutPut2;
    myBatch2->Mode = batCopy;
    myBatch2->Execute();
    SAV_BK_Q->Close();
    delete myOutPut2;
    delete myBatch2;
   // Detail in report
   /*  BILLQuery->Close();
    BILLQuery->Params->Items[0]->AsString  = IDS->Text;
    BILLQuery->Params->Items[1]->AsInteger = MONTHS->ItemIndex+1;
    BILLQuery->Params->Items[2]->AsString  = YEARS->Text;
    BILLQuery->ExecSQL();
    BILLQuery->Open();*/
    Table1->Active = false;
    REP_BILL->Close();
    GEN_REP->Show();
    /*REP_PRE_BILL->BILLREPORT->Preview();*/

}
//---------------------------------------------------------------------------
void __fastcall TREP_BILL::BitBtn2Click(TObject *Sender)
{
    REP_BILL->Close();
}
//---------------------------------------------------------------------------
```