# การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์

# 3-TIER CLIENT/SERVER DEVELOPMENT


นางสาว กิ่งดาว วิวรรธน์สมบัติ
นางสาว สุญาดา ลิบประภากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

# การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์
# 3-TIER CLIENT/SERVER DEVELOPMENT


โดย

นางสาว กิ่งดาว วิวรรธน์สมบัติ

นางสาว สุญาดา ลิบประภากร


อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนา Client/Server แบบ 3 เทียร์

ผู้จัดทำ

1. นางสาว กิ่งดาว วิวรรธน์สมบัติ รหัสประจำตัว 39014031
2. นางสาว สุญาดา ลิบประภากร รหัสประจำตัว 39014583

อาจารย์ที่ปรึกษา

( ดร. วรวัฒน์ ลิ้มโภคา )

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถสำเร็จลุล่วงไปได้หากขาด ความช่วยเหลือและความร่วมมือจาก

คุณ มารดา บิดา อันเป็นที่เคารพรักยิ่งของลูก คอยอบรมสั่งสอน ตักเตือน ตั้งแต่เล็กจนโต

อาจารย์ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา ผู้คอยแนะนำ ให้คำปรึกษา และให้ความรู้ในทุกๆ ด้าน ตลอดหนึ่งปีเต็มที่ผ่านมา

อาจารย์ทุก ๆ ท่านที่ประสิทธิประสาทวิชาความรู้ต่าง ๆ ตลอดสี่ปีที่ผ่านมา

คุณ สมัย มูลโคตร บริษัท Textile Prestige Public Company Limited ผู้คอยให้คำแนะนำ และช่วยเหลือเมื่อเกิดปัญหา

ห้องวิจัยและพัฒนาระบบฐานข้อมูล ( ห้องดาต้าเบส ) สำหรับสถานที่ทำงานและอุปกรณ์การทำงานทางคอมพิวเตอร์ทุกชิ้น

เพื่อน ๆ 4D และ 3P ทุกคน สำหรับคำแนะนำ ความร่วมมือ คอยให้ความช่วยเหลือทุกครั้งที่เกิดปัญหา และคอยให้กำลังใจเสมอมา

สุดท้ายขอมอบคุณความดีจากปริญญานิพนธ์ ฉบับนี้ให้แด่ คุณ มารดา บิดา และครูอาจารย์ผู้มีพระคุณทุกท่าน

กิ่งดาว วิวรรธน์สมบัติ
สุญาดา ลิบประภากร

# เรื่อง การพัฒนา Client/Server แบบ 3 เทียร์

กิ่งดาว วิวรรธน์สมบัติ 39014031
สุญาดา ลิบประภากร 39014583
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

## บทคัดย่อ

ปัจจุบันแนวคิดการพัฒนาแอพพลิเคชั่นเชิงวัตถุได้รับความนิยมจากผู้พัฒนาซอฟต์แวร์ในยุคใหม่กันอย่างแพร่หลาย เนื่องจากมีความสะดวกและง่ายต่อการพัฒนา การแก้ไข และการบำรุงรักษา การพัฒนาแอพพลิเคชั่นผ่านระบบเครือข่ายคอมพิวเตอร์โดยใช้สถาปัตยกรรมแบบ 3 เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ ปัจจุบันได้รับความนิยมเป็นอย่างมาก เนื่องจากมีข้อดีกว่าสถาปัตยกรรม 2 เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ ทั้งในด้านของระบบเครือข่ายที่มีประสิทธิภาพมากขึ้นและสามารถลดค่าใช้จ่ายในการบำรุงรักษาระบบได้อีกด้วย

ปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอแนวทางวิเคราะห์ ออกแบบและพัฒนาระบบงานโดยใช้หลักการเชิงวัตถุ โดยใช้หลักการเป็นไปตามแนวคิดของยูเอ็มแอล ( UML: Unified Modeling Langauge) ผ่านสถาปัตยกรรม 3 เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ ต่อเชื่อมกับระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle 8.0.5 มีการพัฒนาโปรแกรมโดยใช้บอร์แลนด์เดลไฟ 4 โดยมีการนำมาประยุกต์ใช้งานจริงกับระบบทรัพยากรบุคคล (Human Resource System)

# 3-Tier Client/Server Development

Kingdaw Wiwutsombut 39014031
Suyada Libprapakorn 39014583
Dr. Worawat Limpoka Advisor
Year 1999

**Abstract**

The object oriented concept becomes popular in modern application development because of its useful characteristic such as the extensibility and reusability.

Client/server application developers have developed their application using 3-tier client/server architecture instead of 2-tier client/server architecture because of the several advantages of architecture.

This thesis is concerned with an object oriented application development. Analysis, design using Unified Modeling Language (UML). Develop object oriented application using 3-tier client/server architecture connecting with " Oracle 8.0.5 " Object Relational Database Management System( ORDBMS) . Developing tool is Delphi 4 client/server suit. We have implemented object oriented application for the Human Resource System.

# สารบัญ

# สารบัญรูปภาพ

# สารบัญตาราง

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญและที่มา

การเขียนโปรแกรมเชิงวัตถุ (OOP: Object Oriented Programming) ในปัจจุบันเป็นที่นิยมกันอย่างแพร่หลายเนื่องจากวิธีการเขียนโปรแกรมมีแนวความคิดในการมองภาพวัตถุ (Object) เป็นไปตามโลกของความเป็นจริง การจัดโครงสร้างของโปรแกรมมีความเป็นระเบียบมากยิ่งขึ้นเพื่ออำนวยต่อการพัฒนาโปรแกรมในรุ่นต่อ ๆ ไป การแก้ไขปรับปรุงสามารถทำได้ง่าย และไม่มีผลกระทบต่อโครงสร้างส่วนใหญ่ของโปรแกรม

หลักการทำงานพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ จะมีการกำหนดสิ่งต่าง ๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจ็กต์ ซึ่งในแต่ละออบเจ็กต์จะประกอบด้วยแอตทริบิวต์ (Attribute) และเม็ทธอด (Method) รวมกันอยู่ภายใน ซึ่งจะถูกซ่อนรายละเอียดไว้ (Encapsulation) เพื่อเป็นการปกป้องข้อมูลไม่ให้ข้อมูลไปปะปนกับออบเจ็กต์อื่น จะมีการติดต่อสื่อสารกันโดยการเรียกใช้เม็ทธอด โดยจะมีการส่งเมสเสจ (Message) เพื่อเป็นการติดต่อสื่อสารกันระหว่างออบเจ็กต์

จุดมุ่งหมายหลักของการพัฒนาระบบเชิงวัตถุ ได้แก่

1. ทำให้การพัฒนาระบบใช้เวลาสั้นลง ต้นทุนต่ำ โดยความสามารถในการนำคลาสกลับมาใช้งานใหม่อีกครั้ง
2. ต้นทุนและค่าใช้จ่ายในการบำรุงรักษาและแก้ไขระบบต่ำลง เพราะสามารถหาสิ่งที่ต้องการเปลี่ยนแปลงในระบบได้ง่าย และการเปลี่ยนแปลงจะไม่มีผลต่อส่วนภายนอกคลาส

ปัจจุบันได้มีการพัฒนาภาษาใหม่ขึ้นมาเพื่อรองรับหลักการของการเขียนโปรแกรมเชิงวัตถุ เช่น C++, Delphi 4, JAVA

ในส่วนของฐานข้อมูลเนื่องจากฐานข้อมูลเชิงสัมพันธ์(Relational Database System : RDB) มีขีดความสามารถจำกัดอาทิ เช่น

- ฐานข้อมูลเชิงสัมพันธ์ไม่รองรับข้อมูลที่มีความซับซ้อน
- การพัฒนาระบบโดยฐานข้อมูลเชิงสัมพันธ์ มีความยุ่งยากในการแก้ไขเปลี่ยนแปลงข้อมูลภายหลัง

ดังนั้นจึงมีการนำหลักการเชิงวัตถุมาประยุกต์เข้ากับระบบฐานข้อมูลเชิงสัมพันธ์ เพื่อทำให้ฐานข้อมูลเชิงสัมพันธ์มีความสามารถทางออบเจ็กต์ได้ เรียกว่า ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database System : ORDBMS) โดยระบบฐานข้อมูลเชิงวัตถุสัมพันธ์จะจัดเก็บข้อมูลในรูปแบบของตาราง และจัดการกับฐานข้อมูลเชิงสัมพันธ์ แต่มีคุณสมบัติทางออบเจ็กต์ เช่น การซ่อนรายละเอียด (Encapsulation) การถ่ายทอดจากบรรพบุรุษ ( Inheritance) ในส่วนของการติดต่อกับผู้ใช้งานจะนำหลักการของออปเจ็กต์โอเรียนเต็ดมาใช้

โครงงานนี้นำเสนอการพัฒนาซอฟต์แวร์ระบบทรัพยากรบุคคล (Human Resource System) ผ่านสถาปัตยกรรม 3-เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ โดยใช้วิธีการเชิงวัตถุ (Object-Oriented)

**1.2 วัตถุประสงค์ของโครงงาน**

1. วิเคราะห์และศึกษาค้นคว้าหลักการทำงานของสถาปัตยกรรม 3-เทียร์
2. วิเคราะห์และศึกษาทฤษฎีและหลักการพัฒนาโปรแกรมเชิงวัตถุ
3. ศึกษาและใช้งานโมเดลที่ใช้ในการวิเคราะห์และออกแบบระบบตามแนวคิดของยูเอ็มแอล (UML:Unified Modeling Language)
4. วิเคราะห์และออกแบบระบบทรัพยากรบุคคล (Human Resource System) โดยใช้วิธีการเชิงวัตถุ
5. ศึกษาการสร้างและออกแบบระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (ORDBMS)
6. ศึกษาและทดลองใช้งานระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle 8.0.5
7. ศึกษาและพัฒนาระบบโดยใช้เดลไฟ 4 ( Delphi 4 Client/Server Suit)

**1.3 ขอบเขตของโครงงาน**

โครงงานนี้จะเป็นการวิเคราะห์และออกแบบเชิงวัตถุ (Object Oriented Analysis and Design) โดยจะพัฒนาผ่านสถาปัตยกรรม 3-เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ ซึ่งแบ่งออกเป็น 3 ส่วนคือ

- **Client application:** ไคลเอนต์เป็นลักษณะของ Thin Client ซึ่งจะใช้ระบบปฏิบัติการ Window 98
- **Application server:** เซิร์ฟเวอร์จะดูแลการรันแอพลิเคชั่นที่รันบนเครื่องและทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่างไคลเอนต์และดาต้าเบสเซิร์ฟเวอร์ โดยจะใช้ระบบปฏิบัติการ Windows NT 4.0
- **Remote data server:** เป็นดาต้าเบสเซิร์ฟเวอร์ ที่จะจัดการกับฐานข้อมูลเพียงอย่างเดียว ซึ่งจะรันบนระบบปฏิบัติการ Windows NT 4.0

ระบบฐานข้อมูลโครงงานนี้จะใช้ระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ (ORDBMS) โดยเลือกใช้ระบบจัดการฐานข้อมูล Oracle 8.0.5 ในการพัฒนาโปรแกรมจะพัฒนาโดยใช้ Delphi 4 เป็นเครื่องมือในการพัฒนา ซึ่งระบบที่พัฒนาขึ้นนี้คือ ระบบทรัพยากรบุคคล (Human Resource System)

**1.4 ขั้นตอนการดำเนินงาน**

เริ่มด้วยการศึกษาถึงทฤษฎีและหลักการเชิงวัตถุ ทั้งในส่วนของการวิเคราะห์, ออกแบบ และการโปรแกรมเชิงวัตถุ จากนั้นทำการศึกษาโมเดลในการวิเคราห์และออกแบบเชิงวัตถุ ซึ่งโมเดลที่เลือกใช้คือ UML ศึกษาหลักการและการทำงานของสถาปัตยกรรม 3-เทียร์ ไคลเอนต์/เซิร์ฟเวอร์ การพัฒนาจะใช้ Delphi 4 ซึ่งเป็นเครื่องมือที่สนับสนุนการทำงานของ ระบบไคลเอนต์/เซิร์ฟเวอร์โดยพื้นฐานของ MIDAS (Multi-tier Distributed Application Service Suit) สุดท้ายทำการศึกษาระบบฐานข้อมูลเชิงวัตถุสัมพันธ์

(ORDBMS) โดยเลือกระบบจัดการฐานข้อมูลของ Oracle 8.0.5 โดยหลักการที่ศึกษาทั้งหมดจะนำมาประยุกต์ใช้งาน เพื่อทำการพัฒนาระบบทรัพยากรบุคคล (Human Resource System)

# บทที่ 2
# ทฤษฎีและหลักการการพัฒนาโปรแกรมเชิงวัตถุ

## 2.1 การวิเคราะห์และออกแบบโปรแกรมเชิงวัตถุโดยใช้ UML

UML ใช้ในการสร้าง System Model ขึ้นมาทำให้สามารถวิเคราะห์และออกแบบระบบได้อย่างถูกต้องโดย UML จะประกอบด้วยส่วนต่างๆ ดังนี้

1. **Views :** จะทำการแสดงด้านต่างๆ ของระบบออกมาในรูปแบบของโมเดล (Model) โดยวิว (View) จะไม่ใช่กราฟ (Graph) แต่เป็นแอ็บสแตร็กชั่น (Abstraction) ที่ประกอบด้วยไดอะแกรม (Diagram) ต่างๆ
2. **Diagrams :** คือกราฟที่อธิบายถึงข้อมูลและรายละเอียดของวิว
3. **Model Elements :** เป็นแนวคิดที่ถูกใช้ในไดอะแกรมซึ่งจะแสดงออกมาในรูปแบบตามหลักการของออบเจ็กต์โอเรียนเต็ด เช่น คลาส, ออบเจ็กต์, เมสเสจ และความสัมพันธ์ซึ่งจะมี Association, Dependency และ Generalization
4. **General mechanisms :** เราสามารถเพิ่มคำอธิบาย (Comments),ข้อมูล (Information) หรือความหมาย (Semantics) เข้าไปในส่วนประกอบของโมเดลได้

ซึ่งเราจะสนใจในส่วนของไดอะแกรมที่ใช้ในการอธิบายระบบ

### 2.1.1 การวิเคราะห์เชิงวัตถุ ( Object-Oriented Analysis )

#### 2.1.1.1 UML Diagram

**1. Static Modeling**

**Use-Case Diagram**

Use Case นั้นจะช่วยในการอธิบายภาพรวมของระบบไม่ว่าจะเป็นระบบที่สร้างขึ้นใหม่หรือระบบที่มีอยู่แล้ว ซึ่งใช้ในการทำความเข้าใจร่วมกันระหว่างผู้พัฒนาระบบกับลูกค้า (End-users) ว่าระบบที่ลูกค้าต้องการนั้นจะมีฟังก์ชั่นการทำงานเป็นอย่างไร

Use Case Diagram นั้นจะประกอบด้วย

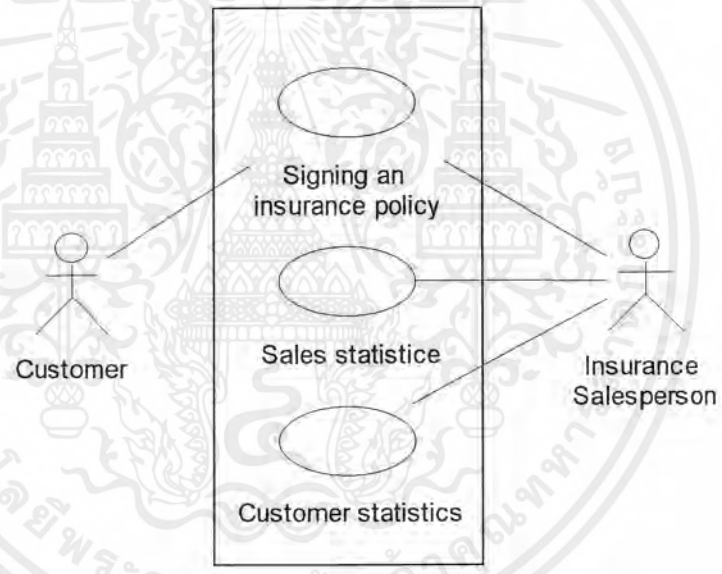
- **ระบบ** (System) : ระบบที่พิจารณาซึ่งไม่จำเป็นต้องเป็นระบบซอฟท์แวร์ (Software System) อาจเป็น ระบบธุรกิจ หรือเครื่องจักรก็ได้
- **แอ็คเตอร์** (Actor) : เป็นบุคคลหรือเป็นอย่างอื่นก็ได้ที่มีการกระทำกับระบบ ซึ่งจะถูกระบุเป็นคลาสตามบทบาทหน้าที่ของแอ็คเตอร์ ไม่ใช่เป็นอินสแตนซ์ (Instance)
- **Use case** : เป็นฟังก์ชั่นการทำงานที่เกิดขึ้นโดยแอ็คเตอร์
- **ความสัมพันธ์** :

  1. ความสัมพันธ์ ระหว่างแอ็คเตอร์ : มีเพียง Generalization

2. ความสัมพันธ์ ระหว่างแอ็คเตอร์กับ Use Cases : มีเพียง Association ซึ่งไม่มีทิศทาง สามารถทำการติดต่อสื่อสารได้สองทิศทาง
3. ความสัมพันธ์ ระหว่าง Use Cases : จะมี ความสัมพันธ์แบบขยาย (Extends Relationship) และความสัมพันธ์แบบนำมาใช้ (Uses Relationship)

**การหา Use Cases**

- ฟังก์ชั่นที่แอ็คเตอร์ต้องการจากระบบ
- แอ็คเตอร์ต้องการให้เกิดอีเวนต์บางอย่างในระบบหรือไม่ถ้ามีแสดงว่าต้องมีฟังก์ชั่นมารองรับอีเวนต์ตัวนั้น
- อินพุท/เอาต์พุตที่สำคัญของระบบ
- ปัญหาหลักที่เกิดขึ้นกับระบบปัจจุบัน




*รูปที่ 2-1 แสดงตัวอย่าง Use-Case Diagram โดยแสดงถึงแอ็คเตอร์, Use cases และ Relationships*

**คลาสไดอะแกรม (Class Diagram)**

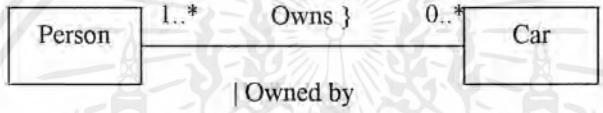
คลาสไดอะแกรมจะทำการแสดง Static structure ของคลาสในระบบ ซึ่งแต่ละคลาสเชื่อมต่อกันด้วยความสัมพันธ์ประเภทต่างๆ นอกจากนี้คลาสไดอะแกรมยังแสดงถึงพฤติกรรม (Behavior) ด้วยซึ่งยึดตามหลักของออบเจ็กต์โอเรียนเต็ด ทำให้เราสามารถเปลี่ยนคลาสต่างๆ ในคลาสไดอะแกรมไปเป็นการโปรแกรมเชิงวัตถุได้เลย

หลักในการหาคลาส

- สิ่งที่มีการเกี่ยวข้องกับข้อมูล เช่น Store, Transform, Analyze, Handle น่าจะเป็นคลาส
- สิ่งแวดล้อมภายนอกระบบ (External System)
- Patterns, Class libraries, Components ถ้ามีอยู่ในระบบแล้วน่าจะเป็นคลาส
- อุปกรณ์ที่ระบบควบคุมอยู่
- บทบาทหน้าที่ของแอ็คเตอร์ เช่น ผู้ใช้, ผู้ดูแลระบบ, ลูกค้า

ความสัมพันธ์ระหว่างคลาส

- **Association** เป็นความสัมพันธ์แบบสองทิศทางโดยจะแสดงเหมือนเป็นตัวเชื่อมระหว่างสอง ออบเจ็กต์ในคลาสนั้นโดย Association นั้นสามารถมีชื่อได้ 2 ชื่อตามแต่ละทิศทางและสามารถมีเครื่องหมายแสดงแสดงจำนวนของความสัมพันธ์นี้ไว้ด้วย

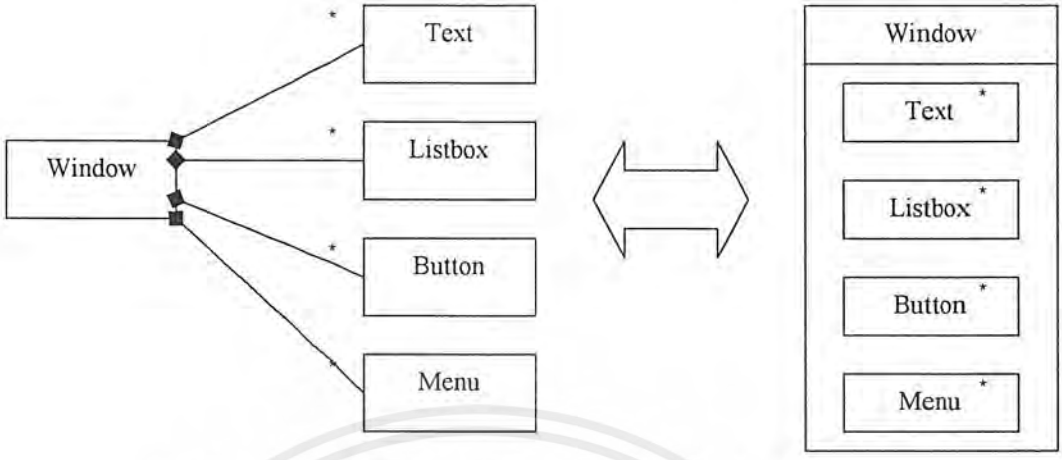



*รูปที่ 2-2 แสดงความสัมพันธ์แบบ Association โดยมีบุคคลตั้งแต่ 1 คนขึ้นไป สามารถมีรถได้ตั้งแต่ 0 คันขึ้นไป*

- **Aggregation** จะเป็นความสัมพันธ์แบบเป็นส่วนหนึ่งของคลาสใหญ่ ซึ่งมีคีย์เวิร์ด (Keyword) เช่น Consists of, Contains, Is part of

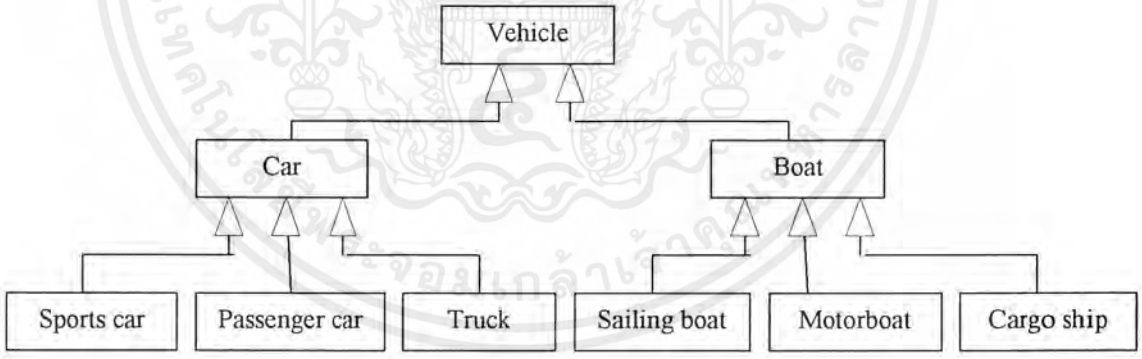

*รูปที่ 2-3 แสดงความสัมพันธ์แบบ Aggregation*

- **Composition** จะเป็นความสัมพันธ์ในลักษณะเป็นส่วนประกอบที่อยู่ในคลาสใหญ่



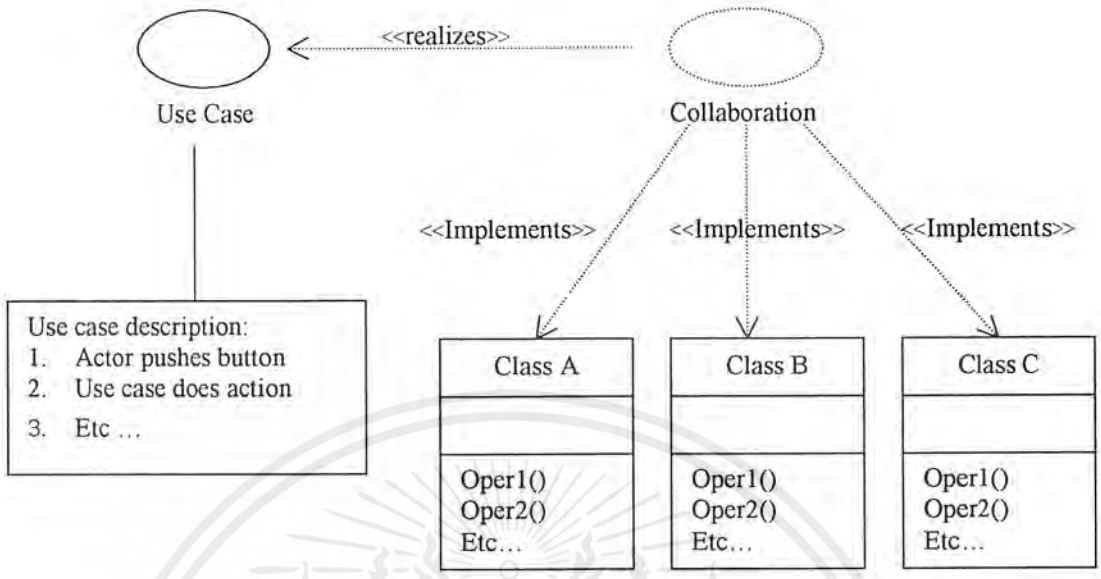

*รูปที่ 2-4 แสดงความสัมพันธ์แบบ Composition ซึ่งแสดงได้ 2 รูปแบบ*

- **Generalization** เป็นความสัมพันธ์ในลักษณะของเจเนอรัลคลาส (General Class) หรือเรียกว่าคลาสแม่ (Super Class) และ Specific Class หรือคลาสลูก(Sub Class) โดยคลาสลูกจะได้รับการถ่ายทอดแอตทริบิวต์, โอเปอร์เรชั่น และ แอสโซสิเอชั่น (Associations) จากคลาสแม่ และ คลาสแม่จะทำการเพิ่มแอตทริบิวต์, โอเปอร์เรชั่น และ แอสโซสิเอชั่น ขึ้นมาเองด้วย




*รูปที่ 2-5 แสดงความสัมพันธ์แบบ Generalization*

ความสัมพันธ์ระหว่าง Use Case กับคลาสไดอะแกรม




*รูปที่ 2-6 แสดงความสัมพันธ์ระหว่าง Use case กับ Class ซึ่ง 1 Use case อาจมาจาก Operation ของ Class ต่างๆ มารวมกัน*

**2. Dynamic Modeling**

เป็นโมเดลที่แสดงถึงพฤติกรรมของระบบ (Behavior of a System) และยกตัวอย่างถึงการที่ออบเจ็กต์ติดต่อถึงกัน (Interact) ในช่วงเวลาต่างๆในขณะที่ทำการรันโปรแกรม(Execute) ของระบบ

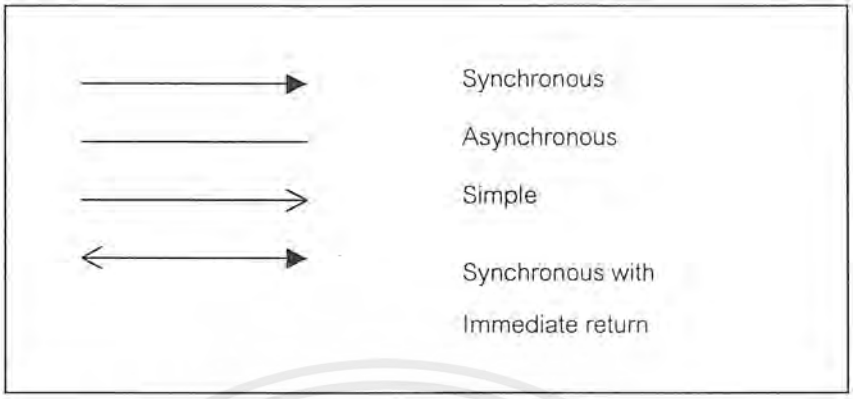
Dynamic Diagram แบ่งออกเป็น 4 แบบ

- State diagrams
- Sequence diagrams
- Collaboration diagrams
- Activity diagrams

ในขั้นตอนการวิเคราะห์ความต้องการของระบบไม่จำเป็นต้องสร้างทั้ง 4 ไดอะแกรม สร้างเอกสารเฉพาะที่มีความสัมพันธ์ก็ได้

**Interaction between Object (Message)**

ในการโปรแกรมเชิงวัตถุ การติดต่อระหว่างออบเจ็กต์ จะใช้การส่งเมสเสจจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่ง




*รูปที่ 2-7 แสดงการติดต่อระหว่าง Object*

จากรูปแสดงรูปแบบของเมสเสจ

- **Simple** เป็นแบบที่ไม่บอกรายละเอียดของการติดต่อ
- **Synchronous** เป็นแบบที่ต้องรอการตอบสนองก่อนที่จะทำโอเปอร์เรชั่นต่อไปได้
- **Asynchronous** เป็นแบบที่ไม่ต้องรอผลตอบสนอง ปกติแล้วใช้ในระบบแบบ real-time ซึ่งแต่ละ ออบเจ็กต์สามารถทำโอเปอร์เรชั่นพร้อมกันได้

**State Diagram**

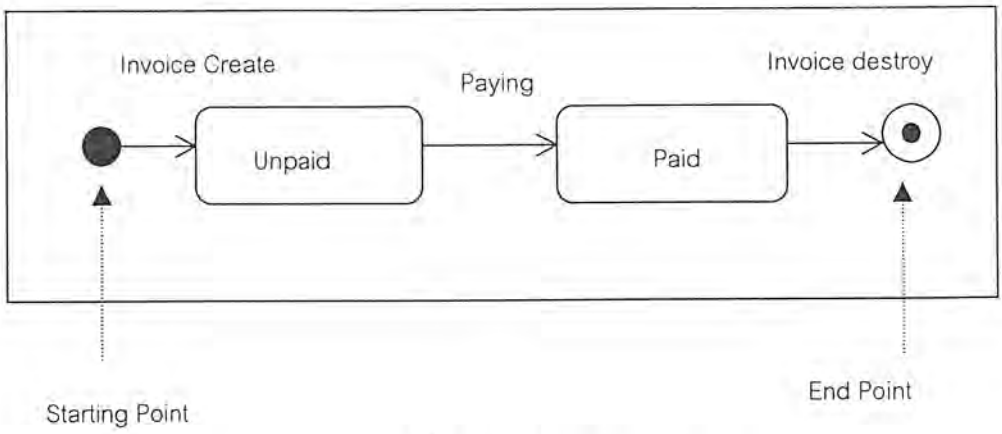
เป็นไดอะแกรมที่แสดงวงจร (cycle) ของออบเจ็กต์, ระบบย่อย (subsystem), และระบบแสดงพฤติกรรมของสถานะนั้น โดยที่จะบอกว่าออบเจ็กต์หนึ่งๆ นั้นมีสถานะใดได้บ้างและเหตุการณ์ (Event) มีผลกระทบอย่างไรต่อสถานะของออบเจ็กต์บ้าง ซึ่งจะขึ้นอยู่กับสถานะก่อนหน้า (previous state) ของออบเจ็กต์ด้วย

สถานะ (State) คือ ผลของการกระทำก่อนหน้า (Previous Activity)

**State transition** คือ การเปลี่ยนสถานะ ซึ่งจะถูกแสดงโดยเส้นตรงลากเชื่อมโยงระหว่างสถานะหนึ่งกับอีกสถานะหนึ่ง ซึ่งอาจจะมีการกำหนดชื่อซึ่งเป็นสิ่งที่บอกถึงสาเหตุของเหตุการณหรือไม่กำหนดก็ได้

**Event** คือ บางสิ่งบางอย่างที่เกิดขึ้นและอาจจะก่อให้เกิดการกระทำบางอย่าง

ในรูปแสดงตัวอย่างของ State Diagram ที่มีผลต่อ event




*รูปที่ 2-8 แสดง State Diagram*

**State** อาจแบ่งออกได้เป็น 3 ส่วน คือ

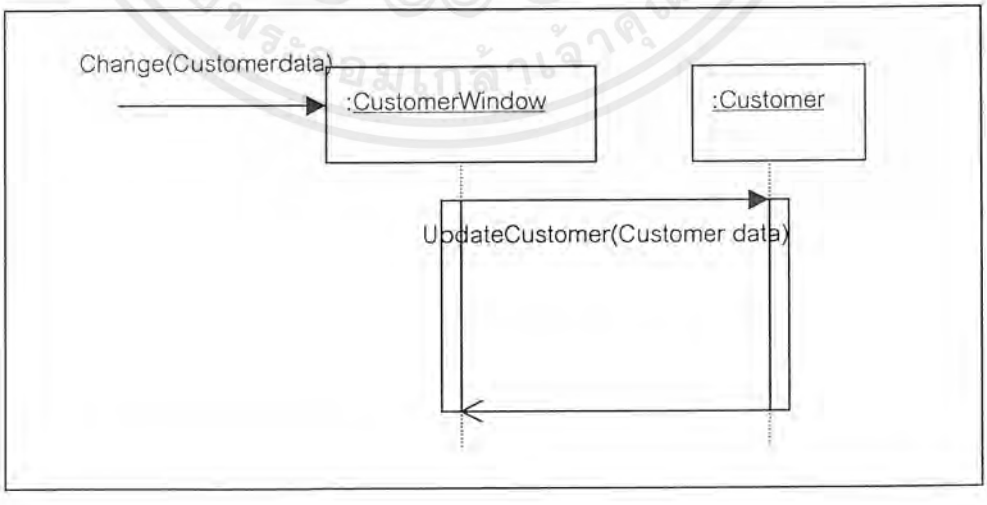
1. ชื่อของ state
2. State variable
3. Activity

**Sequence Diagram**

เป็นไดอะแกรมที่แสดงว่า ออบเจ็กต์ติดต่อซึ่งกันและกันได้อย่างไร โดยจะเน้นไปที่ "ลำดับขั้นของข้อความ" (Message Sequence) หรือเป็นการเน้นที่เวลา (Time) ของการส่งเมสเสจของฝ่ายส่งและฝ่ายรับ ซึ่งหมายถึง ข้อความถูกส่งและรับระหว่าง ออบเจ็กต์ หลายๆ ออบเจ็กต์ ได้อย่างไร

ในไดอะแกรมนี้จะประกอบด้วย 2 แกน คือ

- Vertical Axis (แกนแนวตั้ง) แสดงถึง Time
- Horizontal Axis (แกนแนวนอน) แสดงถึง กลุ่มของ ออบเจ็กต์



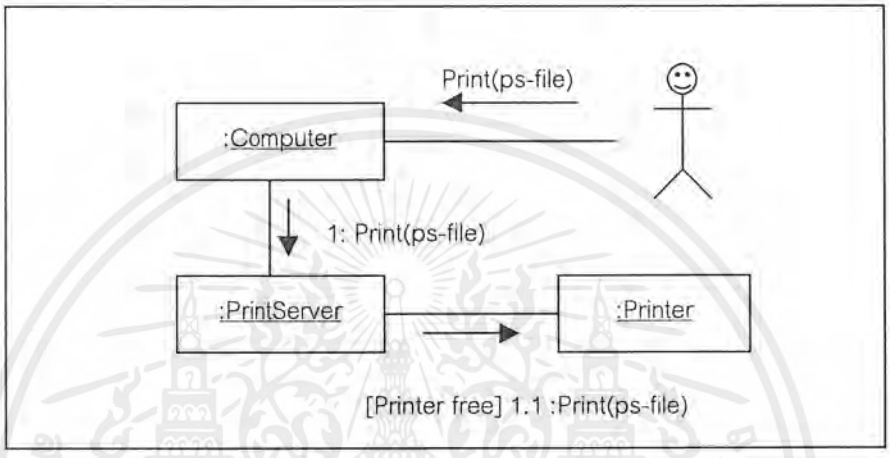

*รูปที่ 2-9 แสดงตัวอย่าง Sequence Diagram*

**Collaboration Diagram**

ไดอะแกรมนี้จะเน้นการติดต่อ (Interaction) และการเชื่อมโยง (Link) ระหว่าง ออบเจ็กต์ ที่มีความสัมพันธ์กัน

ในการวาดไดอะแกรมจะวาดคล้ายๆกับ การวาดคลาสแต่จะแตกต่างตรงที่ชื่อจะต้องขีดเส้นใต้ เราอาจจะเขียนข้อความลงบนเส้นที่แสดงการติดต่อระหว่าง ออบเจ็กต์ แล้วจะต้องมีการกำหนดตัวเลขเพื่อบอกลำดับด้วย

Collaboration diagram จะเริ่มต้นด้วยการเรียก operation call ดังตัวอย่าง




*รูปที่ 2-10 แสดง Collaboration Diagram*

## 2.1.2 การออกแบบเชิงวัตถุ

ในส่วนของการออกแบบระบบนั้น จะกำหนดถึงรายละเอียดตามโมเดลที่ได้จากการวิเคราะห์ การออกแบบจะแบ่งออกเป็น 3 ส่วน คือ

1. **Architectural Design :** การออกแบบในระดับสถาปัตยกรรม
2. **Mechanistic Design :** การออกแบบในระดับโครงสร้างการทำงาน
3. **Detailed Design :** การออกแบบในระดับของการทำงานโดยละเอียด

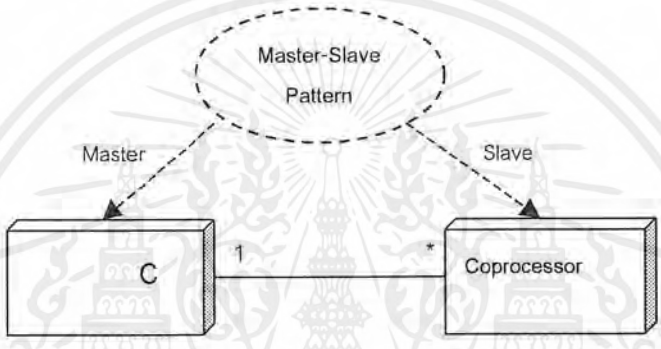
### 2.1.2.1 Architectural Design

Architectural Design เป็นการออกแบบขั้นสูง ที่จะทำการกำหนดแพ็คเกจ (Packages หรือ Subsystems) ที่ประกอบด้วย ความสัมพันธ์และการติดต่อสื่อสารระหว่างแพ็คเกจที่สำคัญในระบบ โดยที่สถาปัตยกรรม (Architecture) ที่ได้จะต้องง่าย และทำความเข้าใจได้ง่าย ในการออกแบบในส่วนนี้ จะเกี่ยวข้องกับทั้งซอฟท์แวร์และ Physical Architecture ซึ่งทั้งสองส่วนนี้จะต้องสามารถทำงานร่วมกันได้อย่างดี

**Architectural Patterns**

เนื่องจากการออกแบบในระดับสถาปัตยกรรมนั้น ส่วนใหญ่จะมีรูปแบบที่ตายตัว ดังนั้นจึงมีการใช้แพ็ทเทิร์นซึ่งจะเรียกว่าไดอะแกรมแพ็ทเทิร์น (Design Pattern) ที่มีอยู่แล้วนำกลับมาใช้ใหม่ ซึ่งทำให้การออกแบบระบบทำได้สะดวกและรวดเร็วขึ้น โดย Design Pattern จะแสดงโครงสร้างในการติดต่อกันของ ออบเจ็กต์ แล้วเรียบเรียงโครงสร้างที่มีขนาดใหญ่ให้ได้เป็นระบบที่ต้องการ

เครื่องหมายสำหรับ Design Pattern จะใช้รูปวงรีที่เป็นเส้นประ แทนแพ็ทเทิร์นซึ่งจะใช้เส้นประแทนการติดต่อกับเอ็นติตี้ (Entity) อื่นๆ ตามหน้าที่ของเอ็นติตี้นั้นต่อระบบ โดยใช้โหนด หรือ แพ็คเกจแทน ออบเจ็กต์




*รูปที่ 2-11 แสดงตัวอย่างของการเขียนดีไซน์แพทเทิร์น*

Pattern ที่ใช้ในการออกแบบ แบ่งออกเป็น 2 ประเภท คือ Control Pattern และ Safety and Reliability Pattern ดังข้อมูลในตาราง

| ประเภท | ชื่อแพทเทิร์น | จุดประสงค์ |
|---|---|---|
| Control Pattern | Master-Slave Pattern | เพื่อจัดการควบคุมงานบนหลายๆ โพรเซสเซอร์ |
| | Microkernel | ลดความซับซ้อนของส่วนประกอบในระบบให้เป็นกลุ่มของส่วนประกอบ ซึ่งแต่ละกลุ่มก็จะรับผิดชอบหน้าที่ในระดับที่ต่างกัน ซึ่งช่วยให้สามารถนำไปใช้กับระบบใหม่ได้ง่าย |
| | Proxy | ใช้จัดการกับการกระจายของข้อมูลจากเซิร์ฟเวอร์กลางไปยังไคลเอนท์หลายๆ เครื่องที่อยู่ในระยะไกล |
| | Broker | ใช้จัดการกับการกระจายของข้อมูลจากเซิร์ฟเวอร์กลางไปยังไคลเอนท์หลายๆ เครื่องที่อยู่ในระยะไกล ซึ่งเซิร์ฟเวอร์จะยังไม่รู้ตำแหน่งของไคลเอนท์ในระหว่างการพัฒนาระบบ |
| Safety and Reliability | Homogeneous Redundancy | เพิ่มความเชื่อถือได้และความปลอดภัยเมื่อเกิดความผิดพลาดขึ้นที่จุดเดียว |
| | Diverse Redundancy | เพิ่มความเชื่อถือได้และความปลอดภัยเมื่อเกิดความผิดพลาดขึ้นที่ใดเพียงจุดเดียว และการผิดพลาดเนื่องจากซอฟท์แวร์ |
| | Monitor-Actuator | ใช้เพิ่มความปลอดภัยให้กับระบบโดยใช้ช่องสัญญาณ monitor และ actuator |
| | Watchdog | เป็นวิธีการตรวจจับความผิดพลาดที่อาจจะเกิดขึ้นได้ระหว่างการทำงาน |
| | Safety Executive | จัดการกระบวนการความปลอดภัยของระบบเมื่อมีความผิดพลาดเกิดขึ้น |

***ตารางที่ 2-1 แสดงข้อมูลของดีไซน์แพทเทิร์นของการออกแบบระดับสถาปัตยกรรม***

**2.1.2.2 Mechanistic Design Patterns**

Mechanistic design เป็นการจัดการกับคลาสและการร่วมมือของออบเจ็กต์ต่าง ๆ ให้บรรลุตามเป้าหมาย Mechanistic Design คือ สถาปัตยกรรม พื้นฐานที่ครอบคลุมถึงการค้นหาและการใช้แพ็ทเทิร์นของ object collaboration

การ Design Patterns ประกอบด้วย

- ปัญหาพื้นฐานรวมทั้งขอบเขตของปัญหา
- การทำให้เป็น Solution
- ผลลัพธ์ของ Pattern

การออกแบบโครงสร้างการทำงานของระบบก็จะใช้ดีไซน์แพทเทิร์นเช่นเดียวกัน ซึ่งแบ่งแพทเทิร์น ออกเป็น 3 ประเภท คือ Simple pattern, Reuse pattern และ State behavior pattern ซึ่งข้อมูลของแพทเทิร์นต่างๆ จะแสดงในตาราง

| ประเภท | ชื่อแพทเทิร์น | จุดประสงค์ |
|---|---|---|
| Simple Patterns | Observer | อนุญาตให้ multiple clients สามารถ share ประสิทธิภาพจาก server ได้และสามารถ update ได้โดยอัตโนมัติ |
| | Model View Controller | สามารถแยก user input , data maintenance , manipulation และการแสดงผล |
| | Transaction | ควบคุมการสื่อสารระหว่าง object ด้วยค่าของ level ที่เปลี่ยนแปลง |
| | Smart Pointer | หลีกเลี่ยงปัญหาด้วย dumb pointers |
| Reuse | Container | เป็น data structuring concepts ที่มาจาก application domain เป็นการนำ class กลับมาใช้งานใหม่อย่างมีประสิทธิภาพ |
| | Interface | support multiple implementations of given type และ support multiple types ด้วยโครงสร้างแบบภายใน |
| | Policy | จัดเตรียมเพื่อให้ง่ายต่อการเปลี่ยนแปลงอัลกอลิทึมและ Procedure |
| | Rendezvous | จัดเตรียม flexible mechanism สำหรับ light-weight intertask communication |
| State Behavior | State | จัดเตรียม optimal state machine implementation เมื่อมีการเปลี่ยนแปลงของ state |
| | State Table | จัดเตรียมเพื่อให้มีประสิทธิภาพในการ maintain และ การ execute state machines ที่ขนาดใหญ่และมีความซับซ้อน |

***ตารางที่ 2-2 แสดงข้อมูลของดีไซน์แพทเทิร์นของการออกแบบระดับโครงสร้าง***

## 2.2 การเขียนโปรแกรมเชิงวัตถุ ( Object Oriented Programming : OOP)

การเขียนโปรแกรมเชิงวัตถุเป็นเทคนิคใหม่ของการเขียนโปรแกรมที่เราสามารถโมเดล ออบเจ็กต์บนโลกแห่งความจริงได้ โครงสร้างพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ ได้แก่ คลาสของออบเจ็กต์และออบเจ็กต์ที่เป็นสมาชิกของคลาสดังกล่าว

การเขียนโปรแกรมเชิงวัตถุ คือวิธีการเขียนโปรแกรมที่มีแนวความคิดในการมองภาพวัตถุ (Object) ในโลกแห่งความเป็นจริง เป็นการจัดโครงสร้างของโปรแกรมให้เป็นระเบียบมากยิ่งขึ้น เพื่อเอื้ออำนวยต่อการพัฒนาโปรแกรมในรุ่นต่อ ๆ ไป การแก้ไขปรับปรุง สามารถทำได้ง่าย และไม่มีผลกระทบต่อโครงสร้างส่วนใหญ่ของโปรแกรม

หลักการทำงานพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ จะมีการกำหนดสิ่งต่าง ๆ ที่อยู่ ภายในขอบเขตของระบบงานเป็นออบเจ็กต์ ซึ่งในแต่ละ ออบเจ็กต์ จะประกอบด้วยแอตทริบิวต์ (Attribute) และเมทธอด (Method) รวมกันอยู่ภายใน ซึ่งจะถูกซ่อนรายละเอียดไว้ (Encapsulation) เพื่อเป็นการปกป้องข้อมูล ไม่ให้ข้อมูลไปปะปนกับ ออบเจ็กต์อื่น ในการติดต่อกันระหว่างออบเจ็กต์นั้น จะมีการติดต่อกันโดยการเรียกใช้เมทธอด โดยจะมีการส่งเมสเสจเพื่อเป็นการติดต่อสื่อสารกันระหว่างออบเจ็กต์

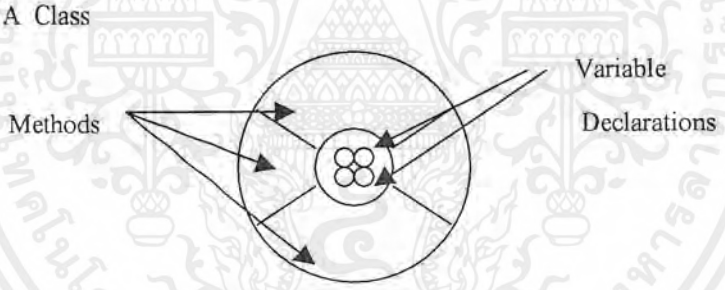
จุดมุ่งหมายหลักของการพัฒนาระบบเชิงวัตถุ ได้แก่

1. ทำให้การพัฒนาระบบใช้เวลาสั้นลง ต้นทุนต่ำ โดยความสามารถในการนำคลาสกลับมาใช้งานใหม่อีกครั้ง
2. ต้นทุนและค่าใช้จ่ายในการบำรุงรักษาและแก้ไขระบบต่ำลง เพราะเราสามารถหาสิ่งที่ต้องการเปลี่ยนแปลงในระบบได้ง่าย และการเปลี่ยนแปลงจะไม่มีผลต่อส่วนภายนอกคลาส

### 2.2.1 คำศัพท์ที่เกี่ยวข้องกับการพัฒนาโปรแกรมเชิงวัตถุ

#### 2.2.1.1 คลาส ( class )

คลาส คือ กลุ่มของออบเจ็กต์ที่แบ่งตามลักษณะเฉพาะ และการใช้งาน หรือออบเจ็กต์ที่มีคุณสมบัติพื้นฐานเหมือนกัน ข้อมูลเชิงออบเจ็กต์ที่เก็บอยู่ในคลาส เรียกว่า อินสแตนท์ (Instance) ของคลาส โดยที่อินสแตนท์จะเป็นส่วนขยายของคลาสและเก็บอยู่ในส่วนฐานข้อมูล ดังนั้นข้อมูลต่าง ๆ ที่เป็นตัวกำหนดคลาสจะถูกเข้าถึงผ่านทางออบเจ็กต์ของอินสแตนท์และเมทธอดของคลาส โดยคลาสจะห่อหุ้มคุณลักษณะทั้งหมดของออบเจ็กต์ต่าง ๆ และกำหนดชนิดของข้อมูลที่บรรจุอยู่ในออบเจ็กต์พร้อมกับกำหนดเมทธอดต่าง ๆสำหรับการเข้าถึงข้อมูลไว้ด้วย โดยปกติแล้วคลาสจะถูกออกแบบให้มีความสัมพันธ์กับคลาสอื่น และคลาสยังสามารถสืบทอดคุณสมบัติไปยังลูกหลานได้อีกด้วย




*รูปที่ 2-12 แสดงลักษณะของคลาส*

#### 2.2.1.2 ออบเจ็กต์ (Object)

ออบเจ็กต์เป็นโมดูลของข้อมูล (Data) และโค้ดที่สามารถถูกรวมเข้ากับออบเจ็กต์อื่น ๆ เพื่อรวมกันสร้างซอฟต์แวร์ขนาดใหญ่ขึ้น ภายในออบเจ็กต์จะประกอบด้วย 2 ส่วนได้แก่ แอตทริบิวต์ ซึ่งเป็นส่วนของคุณสมบัติต่าง ๆ ของออบเจ็กต์นั้น และส่วนโพรซีเยอร์หรือฟังก์ชั่น ที่เป็นส่วนของการกระทำ ซึ่งเราเรียกว่า เมทธอด

**2.2.1.3 แอตทริบิวต์ (Attribute)**

แอตทริบิวต์ คือ ค่าของข้อมูลที่เป็นส่วนบอกคุณสมบัติของออบเจ็กต์ ที่อยู่ในคลาส เช่น ออบเจ็กต์รถยนต์มีแอตทริบิวต์ สี, ยี่ห้อ ,แรงม้า เป็นต้น

**2.2.1.4 เมทธอด (Method)**

เมทธอด คือ การที่ออบเจ็กต์ สามารถทำสิ่งต่าง ๆ ให้ตัวมันเอง หรือมีสิ่งต่าง ๆ ที่ทำให้มันเช่น ออบเจ็กต์ รถยนต์ อาจมีเมทธอดได้แก่ สตาร์ทเครื่อง,เร่งความเร็ว,เปลี่ยนเกียร์ เป็นต้น

ตัวอย่าง การเขียนโปรแกรมเชิงวัตถุของภาษาเดลไฟ 4 (Delphi4) แสดงในส่วนของออบเจ็กต์หรือคลาส เป็นดาต้าไทป์ (data type) ที่ ซ่อนข้อมูลและเมทธอดเอาไว้

ใน Delphi เรากำหนดคลาสดังนี้

```
type
 TDate = class
  Month, Day, Year: Integer;
  procedure SetValue(m,d,y: Integer);
  function LeapYear: Boolean;
end;
```

และเขียนโค้ดในส่วนของเมธอดของคลาส ดังนี้

```
Procedure TDate.SetValue(m,d,y: Integer);
begin
 Month := m;
 Day := d;
 Year := y;
end;
```

เราสาสามารถกำหนดออบเจคต์ให้แก่คลาสได้ดังนี้

```
var
 ADay: TDay;
```

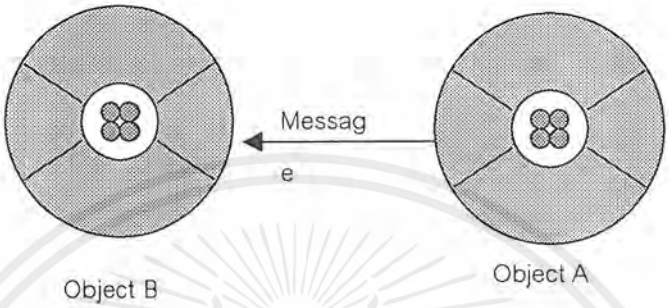
โดยที่ ADay จะเป็น Object ของ class TDay

ในการเรียกใช้ Procedure หรือ function จะทำได้โดยใช้ชื่อของออปเจคต์นำหน้าแล้วตามด้วยชื่อ procedure หรือ function นั้น ดังนี้

```
Aday.SetValue(9,9,1999);
```

**2.2.1.5 เมสเซจ (Message)**

เมื่อเราต้องการอ้างถึงออบเจ็กต์ในขณะนั้นเพื่อเรียกใช้เมทธอดหรือข้อมูลของออบเจ็กต์นั้น เราสามารถกระทำการติดต่อสื่อสารระหว่างออบเจ็กต์ ได้โดยการส่งเมสเสจ ตัวอย่างเช่น ออบเจ็กต์ A ต้องการติดต่อกับออบเจ็กต์ B ออบเจ็กต์ A ก็จะทำการส่งเมสเสจไปยังออบเจ็กต์ B ดังรูป




*รูปที่ 2-13 แสดงการส่งเมสเซจระหว่างออบเจ็กต์*

**2.2.1.6 ตัวแปร Self**

ใช้เมื่อเราต้องการอ้างถึงออบเจ็กต์ปัจจุบันในคลาสนั้น ซึ่งมีประโยชน์อย่างมากเมื่อเราต้องการอ้างถึงออบเจ็กต์ในขณะนั้นเพื่อเรียกใช้เมทธอดหรือข้อมูลของออบเจ็กต์นั้น ตัวแปร Self เป็นฟิลด์ที่มีอยู่ในทุกคลาส ซึ่งจะถูกซ่อนเอาไว้ ซึ่งจะเป็นพอยเตอร์ที่ชี้ไปยังหน่วยความจำที่เก็บอินสแตนซ์ของคลาสนั้น ซึ่งจะใช้ Self เมื่อต้องการสร้างคอมโพเนนท์ในขณะที่รันโปรแกรมอยู่ (Creating Component Dynamically) เช่นเราต้องการสร้างคอมโพเนนท์ Button เมื่อมีการคลิกที่ Button ที่มีอยู่แล้ว ดังโค้ด

```
procedure Tform1.Button1Click(Sender: Tobject);
var
 button : TButton;
begin
 Button := TButton.Create(self);
 Button.Parent := self;
 Button.Caption := 'New Button';
 Button.Left := 30;
 Button.Top := 30;
end;
```

**2.2.1.7 คอนสตรักเตอร์ (Constructor)**

คอนสตรักเตอร์เป็นเมทธอดชนิดพิเศษ ซึ่งจะถูกเรียกใช้ทุกครั้งที่มีการสร้างอินสแตนซ์ของคลาสขึ้นมาโดยเมทธอดนี้จะทำการจองทรัพยากรสำหรับอินสแตนซ์ และกำหนดค่าเริ่มต้น ซึ่งมีตัวอย่างการใช้งานดังนี้

```
Constructor TDog.CreateDog;
Begin
        Name := 'MyDog1';
        Family :='Family1';
{ ตรงนี้อาจจะมีการ จองทรัพยากรจากระบบปฏิบัติการ ไปใช้งานบ้าง }
end;0
```

**2.2.1.8 ดีสตรักเตอร์ (Destructor)**

ดีสตรักเตอร์เป็นเมทธอดที่ทำหน้าที่ตรงกันข้ามกับคอนสตรักเตอร์ คือการคืนทรัพยากรที่อินสแตนซ์ของคลาสใช้ไปเมื่อต้องการเลิกใช้งานอินสแตนซ์ของคลาสนั้น ๆ ตัวอย่างการใช้งานมีดังนี้

```
Destructor TDog.Destroy;
    Begin
    { ตรงนี้จะมีการคืนทรัพยากรในส่วนที่ได้จองเอาไว้ให้กับระบบปฏิบัติการ }
    End;
    เรียกใช้ constructor ได้ดังนี้
    MyAnimal := TAnimal.create;
    และ เรียก destructor ได้โดย
    MyAnimal .free;
```

**2.2.1.9 Overloading**

เมื่อเราต้องการให้คลาสที่สร้างขึ้นสามารถใช้เมทธอดที่มีชื่อซ้ำกันได้ โดยใช้พารามิเตอร์ของเมทธอดเป็นตัวแยกความแตกต่างเราสามารถกระทำได้ดังนี้

```
function AddNum(Num1,Num2: interger): integer; overload;
function AddNum(Num1,Num2:real): real; overload;
```

เมื่อเราใช้งานเมธอดนี้ ถ้าค่าที่กำหนดให้ Num เป็น Integer ก็จะไปเรียกใช้ function แรก แต่ถ้าเป็น real ก็จะไปเรียกใช้ที่ function ที่สอง

**2.2.2 คุณสมบัติที่สำคัญของการพัฒนาโปรแกรมเชิงวัตถุ**

คุณสมบัติที่สำคัญของการพัฒนาโปรแกรม 3 ประการ ได้แก่

1. คุณสมบัติเอนแคปซูเลชัน ( Encapsulation )
2. คุณสมบัติอินเฮอริแตนซ์ ( Inheritance )
3. คุณสมบัติโพลีเมอร์ฟิซึม ( Polymorphism )

**2.2.2.1 คุณสมบัติเอนแคปซูเลชั่น**

คุณสมบัติเอนแคปซูเลชั่น ได้แก่ โครงสร้างของโครงสร้างของโปรแกรมต้องจบในตัวเอง, ทำงานเป็นอิสระ และทำงานได้ด้วยตัวเอง แต่ในขณะเดียวกันก็ต้องสามารถแยกส่วนออกมาและทำงานได้เป็นอิสระเช่นกัน มีการรวมแอตทริบิวต์และเมทธอดเพื่อให้เป็นข้อมูลเชิงออบเจ็กต์ ซึ่งจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างแอตทริบิวต์ภายในและเมทธอดของออบเจ็กต์ด้วย โดยออบเจ็กต์หนึ่งจะมีคุณสมบัติเหมือนสิ่งของอย่างหนึ่งคือ มีทั้งส่วนประกอบและการทำงาน ซึ่งสามารถใช้งานได้ทันที

ข้อดีของการเอนแคปซูเลชัน ได้แก่เมื่อส่วนใดส่วนหนึ่งของระบบถูกเปลี่ยนแปลง จะมีการพิจารณาแก้ไขข้อมูลเพียงส่วนเดียว โดยไม่มีผลกระทบกับส่วนอื่น ๆ ในระบบ

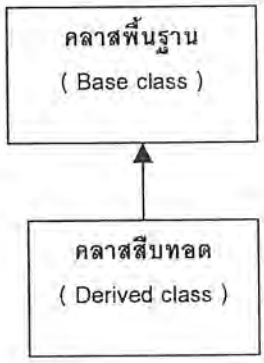
วิธีการในการป้องกันนั้นการเข้าถึงข้อมูลภายในออบเจ็กต์จากฟังก์ชันภายนอก (Data Hiding) จะอยู่ที่ชนิดของการประกาศข้อมูลภายในคลาส ซึ่งมีอยู่ 3 ชนิด ได้แก่

1. **Private** การประกาศข้อมูลแบบนี้จะมีผลทำให้ไม่มีคำสั่งใดที่จะอ้างถึงข้อมูลภายในคลาสนั้นได้เลย นอกจากเมทธอดภายในคลาสนั้นเอง ซึ่งตามปกติแล้วจะมีอย่างน้อยหนึ่งเมทธอด ที่ไม่ได้ประกาศให้เป็นข้อมูลชนิดนี้ เพื่อที่จะได้ใช้ในการเข้าถึงข้อมูลที่ประกาศอยู่ภายในส่วน private ของคลาสได้
2. **Public** การประกาศข้อมูลแบบนี้จะมีผลทำให้สามารถอ้างอิงข้อมูลภายในคลาสได้โดยอิสระ
3. **Protected** การประกาศข้อมูลแบบนี้จะมีผลให้มีเฉพาะเมทธอดในคลาสสืบทอดเท่านั้นที่ จะอ้างอิงข้อมูลที่ประกาศภายในส่วนนี้ได้

**2.2.2.2 คุณสมบัติอินเฮอริแตนซ์**

คุณสมบัติอินเฮอริแตนซ์ได้แก่การที่คลาสเก่าที่มีอยู่แล้วเป็นต้นแบบให้กับคลาสใหม่ที่จะถูกสร้างขึ้น เกิดการถ่ายทอดคุณสมบัติจากคลาสพ่อ (Base Class) ไปสู่คลาสลูก (Derived Class) นอกจากคลาสลูกสืบทอดคุณสมบัติทุกอย่างมาจากคลาสพ่อแล้ว คลาสลูกยังสามารถที่จะเพิ่มคุณสมบัติให้กับตัวเองได้อีกด้วย เช่น การเพิ่มแอตทริบิวต์หรือเมทธอดให้กับตัวเอง คลาสลูกสามารถที่จะใช้ชื่อของเมทธอดเหมือนกับชื่อเมทธอดของคลาสพ่อได้ ทั้งนี้เพื่อเพิ่มความสามารถให้แก่เมทธอดเดิมโดยใช้หลักการของเวอร์ชวล / โอเวอร์ราย (Virtual / Override) การสืบทอดแบบนี้นับว่าเป็นคุณสมบัติเด่นของการเขียนโปรแกรมแบบโอโอพี เพราะทำให้เราสามารถใช้โค้ดเดิมได้โดยไม่ต้องมีการเขียนโปรแกรมที่เคยเขียนไว้แล้วขึ้นมาใหม่อีก เพราะคอมไพเลอร์ (Compiler) จะค้นหาโค้ดเองอีทำการคอมไพล์โปรแกรม ผู้ใช้

สามารถกำหนดต้นแบบของคลาส (Abstract Classes) เพื่อให้ผู้ใช้คนอื่น ๆ นำไปพัฒนาต่อโดยการเพิ่มรายละเอียดเฉพาะของตัวเองเข้าไป




*รูปที่ 2-14 แสดงการสืบทอดจากคลาสพื้นฐานที่หนึ่งคลาส*

- คลาสพื้นฐาน ( Base class ) บางครั้งเรียกคลาสพ่อ
- คลาสสืบทอด ( Derived class ) บางครั้งเรียกคลาสลูก

เราสามารถเขียนเป็นซอสโค้ดได้ดังนี้คือ

```
type
 TForm1 = class(TForm);
End;
```

นั่นคือคลาส TForm1 ถ่ายทอดมาจากคลาส TForm ซึ่งจะมี properties และ methods เหมือนกับคลาสแม่ (Ancestor class) และสามารถเพิ่มพรอพเพอร์ตี้และ เมทธอดใหม่ให้แก่คลาสใหม่นี้ได้

**Override**

การโอเวอร์รายเมทธอด คลาสสืบทอด ( คลาสลูก ) จะโอเวอร์รายเมทธอดจากคลาสพ่อมาเพื่อเพิ่มเติมเมทธอดให้แตกต่างจากคลาสพ่อ ทำได้โดยการเติมคีย์เวิร์ด " Override" หลังการประกาศเมทธอดนั้น เช่น

```
procedure Flash; override;
```

**Virtual Method**

การประกาศเวอร์ชวลเมทธอด ทำเมื่อเราต้องการสร้างรูปแบบการเข้าใช้งานโค้ดที่แตกต่างกัน เมื่อเรียกใช้เมื่อเรียกใช้เมทธอดชื่อเดียวกัน โดยใช้หลักการของ " Late Binding " เมื่อเราประกาศเวอร์ชวลเมทธอดให้กับเมทธอดในคลาสพ่อแล้ว การประกาศเมทธอดของคลาสลูกที่มีชื่อเดียวกับเมทธอดของคลาสพ่อ สามารถทำได้โดยใช้การโอเวอร์รายเมทธอด ซึ่งทำโดยใช้คีย์เวิร์ด Overide ต่อท้ายการประกาศเมทธอดนั้น หรือไม่ใช้การโอเวอร์รายก็ได้

**เออร์ลีไบน์ดิง (Early Binding)**

ในขณะที่คอมไพเลอร์เรียกใช้โพรซีเยอร์หรือฟังก์ชัน คอมไพเลอร์จะรู้ตำแหน่งที่แน่นอนของโค้ดแล้ว เนื่องจากโค้ดได้ผ่านการคอมไพล์และมีการจองพื้นที่ไว้สำหรับโพรซีเยอร์หรือฟังก์ชันแล้ว เมื่อมีการเรียกใช้โพรซีเยอร์หรือฟังก์ชันนั้นแล้วข้ามไปทำงานต่อที่แอดเดรสดังกล่าวเลย การทำงานรูปแบบนี้เรียกว่า "Compile time Binding" หรือ "Early Binding"

**เลตไบน์ดิง (Late Binding)**

การทำงานแบบ Early Binding นั้นไม่สามารถใช้งานได้กับการทำโพลีมอร์ฟิซึม ดังนั้นจึงมีวิธีการอื่น เรียกว่า Late Binding หรือ Runtime Binding ผลของเลตไบน์ดิงก์ทำให้โปรแกรมไม่รู้ล่วงหน้าว่าจะเรียกเมทธอดนั้นได้จากที่ไหน จะรู้ในตอนรันโปรแกรมเท่านั้นว่าจะเรียกใช้เมทธอดใดให้เหมาะสม แสดงว่าการทำงานแบบเลตไบน์ดิงก์จะไม่ผูกติดกับเมทธอดใด ๆ ไว้ในลักษณะแน่นอนตายตัวตั้งแต่ตอนคอมไพล์แต่จะมีการเรียกเมทธอดต่าง ๆ เปลี่ยนแปลงไปตามความเหมาะสมในตอนรันโปรแกรม

### 2.2.2.3 คุณสมบัติโพลีมอร์ฟิซึม

คุณสมบัติโพลีมอร์ฟิซึม ได้แก่ ความสามารถที่จะใช้ได้ทั่วไปคุณสมบัติที่เมื่อออบเจ็กต์ต่าง ๆ ได้รับคำสั่งเดียวกันจากโปรแกรมแล้วแต่ละออบเจ็กต์จะทำงานตามแบบของตัวเอง ซึ่งทำให้ได้ผลลัพธ์แตกต่างกัน ซึ่งเป็นคลาสที่ได้รับการถ่ายทอดคุณสมบัติจากคลาสเดียวกัน

การทำโพลีมอร์ฟิซึมนั้นเป็นคุณส มบัติการเรียกใช้เมทธอดที่ใช้ชื่อเดียวกันโดยมีการเลือกใช้การทำงานของเมทธอด ขึ้นกับตำแหน่งของออบเจ็กต์ที่มีการเรียกใช้เมทธอดนั้น คลาสหลักจะต้องประกาศเมทธอดที่จะใช้เป็นเวอร์ชวลเมทธอด การใช้เมทธอดเดียวกันจะทำงานต่างกันตามชนิดของคลาส

คุณสมบัติโพลีมอร์ฟิซึมแบ่งได้เป็น 3 ประเภท ได้แก่

1. **Inherited Polymorphism** เกิดขึ้นเมื่อออบเจ็กต์ของคลาสที่ต่างกัน เมทธอดที่มีชื่อเดียวกันเนื่องจากคลาสเหล่านั้นสืบทอดมาจากคลาสเดียวกัน เช่น รถยนต์ และรถจักรยานยนต์สืบทอดมาจากคลาสพาหนะเหมือนกัน เพราะฉะนั้น ทั้งรถยนต์และรถจักรยานยนต์จึงวิ่งได้ , เบรกได้เป็นต้น
2. **Independent Polymorphism** เกิดขึ้นเมื่อออบเจ็กต์ของคลาสที่ต่างกัน ใช้เมทธอดที่มีชื่อเดียวกัน แต่วิธีการนั้นมีลักษณะการทำงานต่างกัน เช่นรถยนต์วิ่งได้โดยการกดคันเร่งแต่รถจักรยานยนต์วิ่งได้โดยการบิด
3. **Coincidential Polymorphism** เกิดขึ้นเมื่อออบเจ็กต์ของคลาสที่ต่างกัน ใช้เมทธอดที่มีชื่อเดียวกัน แต่วิธีการนั้นไม่มีความสัมพันธ์กันเลย

ตัวอย่างโปรแกรมมีดังนี้

```
type
        TAnimal = class
Public
        Function Voice: String; virtual;
        TDog = class
```

```
Public
    Function Voice: String; override;
```

ถ้าเราต้องการเรียกใช้ method นี้ โดยใช้ชื่อออบเจคต์เดียวกัน สามารถกำหนดอินสแตนซ์ได้ ตามตัวอย่าง

```
var
    MyAnimal: TAnimal;
    MyDog: TDog;
```

โดยที่ MyAnimal สามารถเป็นได้ทั้งคลาส TAnimal และคลาส TDog โดยกำหนดในโปรแกรมได้ดังนี้

```
MyAnimal := TAnimal.create;
MyDog := TDog.create;
```

จะได้ MyAnimal เป็นอินสแตนซ์ของคลาส Tanimal

```
MyAnimal := MyDog;
```

จะได้ MyAnimal เป็นอินสแตนซ์ของคลาส Tdog
ในการเรียกใช้ method ก็จะเรียกใช้ด้วยคำสั่ง

```
MyAnimal.Voice;
```

ซึ่งถ้า MyAnimal เป็นอินสแตนซ์ของคลาส Tanimal ก็จะเรียก function Tanimal.Voice; และถ้าเป็นอินสแตนซ์ของ Tdog ก็จะเรียก function Tdog.Voice;

การทำ override ทำได้สองแบบ คือ เป็นเมธอดใหม่ที่ไม่เหมือนของเดิมเลย หรือมีโค้ดของเมธอดเดิมอยู่ด้วย ซึ่งเราจะใช้คำสั่ง *inherited* เพื่ออ้างไปยังโค้ดของเมธอดเดิม นอกจากนี้การกำหนดเมธอดให้เป็น virtual และ override นั้น จะเป็นการทำ late binding ซึ่งจะทำการจองพื้นที่ในหน่วยความจำในตอนที่กำลังรันโปรแกรมอยู่

### 2.2.3 การเปรียบเทียบข้อแตกต่างของ Object Oriented Programming กับ Procedural Structure Programming

ข้อแตกต่างที่สำคัญของการเขียนโปรแกรมเชิงวัตถุกับการเขียนโปรแกรมแบบโครงสร้างคือ

#### 2.2.3.1 วิธีการจัดเก็บข้อมูล ( Method to accomplishing actions with data )

**การโปรแกรมเชิงวัตถุ** : เมื่อออบเจ็กต์หนึ่งต้องการทำงานกับเมทธอดของออบเจ็กต์อื่น การทำงานกับเมทธอดของออบเจ็กต์นั้นจะส่งเมสเสจไปที่ออบเจ็กต์ที่ต้องการ เมื่อออบเจ็กต์นั้นได้รับเมสเสจก็จะตอบสนองต่อเมสเสจนั้นตามเมธธอดที่เหมาะสมกับเมสเสจที่ได้รับ

**Procedural:** ก่อนที่โพรซีเยอร์จะทำการจัดการกับข้อมูลนั้น จะมีการส่งค่าพารามิเตอร์ไปที่โพรซีเยอร์นั้นก่อน โดยจะต้องกำหนดโพรซีเยอร์ที่ต้องการใช้งานเลย ซึ่งตัวโปรแกรมจะไม่สามารถเลือกการทำงานได้เอง

**2.2.3.2 ความสามารถในการเขียนอธิบายโครงสร้างชีวิตจริง (Abstraction)**

**การโปรแกรมเชิงวัตถุ :** โครงสร้างของข้อมูล ( Data Abstraction ) จะถูกแทนโดยคลาสของออบเจ็กต์ ส่วนโครงสร้างการทำงาน ( Function Abstraction ) จะถูกแทนโดยเมสเสจ

**Procedural :** โครงสร้างของข้อมูลจะถูกแทนโดยชนิดของข้อมูล ( Data Type ) ส่วนโครงสร้างการทำงานจะถูกแทนโดยการทำงานของโพรซีเยอรัล

**2.2.3.3 โครงสร้างของโปรแกรมต้องจบในตัวเอง (Encapsulation)**

**การโปรแกรมเชิงวัตถุ** : มีการรวมข้อมูลเข้ากับโพรซีเยอร์และฟังก์ชั่นเพื่อให้เป็นข้อมูลชนิดออบเจ็กต์ ซึ่งจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างข้อมูลภายในและเมทธอดของออบเจ็กต์ด้วย โดยออบเจ็กต์หนึ่งจะมีคุณสมบัติเหมือนสิ่งของอย่างหนึ่งคือ มีทั้งส่วนประกอบและการทำงานซึ่งสามารถใช้งานได้ทันที

**Procedural** : การเอนแคปซูเลชั่นจะอยู่ในรูปของไลบารีซอฟท์แวร์คอมโพเนนท์ ( Library Software Component ) ซึ่งแต่ละคอมโพเนนท์จะมีมากกว่าหนึ่งโครงสร้างของข้อมูลอยู่ด้วยกัน

**2.2.3.4 การสืบทอดความสัมพันธ์และความหลากหลาย (Inheritance and Polymorphism)**

**การโปรแกรมเชิงวัตถุ :** สามารถรองรับการสืบทอดและความหลากหลายได้

**Procedural :** ไม่สามารถรองรับการสืบทอดและความหลากหลายได้

**2.2.3.5 การอนุญาตให้ผู้ใช้กำหนดโครงสร้างของโปรแกรมใหม่ได้ (Extensibility and Relative status of protocol)**

**การโปรแกรมเชิงวัตถุ :** อนุญาตให้ผู้ใช้สามารถกำหนดโครงสร้างขึ้นมาใหม่ได้ โดยที่โครงสร้างที่สร้างขึ้นมาใหม่จะมีสถานะเทียบเท่ากับที่ระบบเป็นผู้กำหนด

**Procedural :** ถึงแม้ว่าจะยอมให้ผู้ใช้กำหนดโครงสร้างขึ้นมาใหม่ได้ แต่โครงสร้างที่สร้างขึ้นมาใหม่จะมีความสำคัญและประสิทธิภาพน้อยกว่าโครงสร้างที่สร้างจากระบบ

**2.2.3.6 ลำดับชั้นของคลาส (Class hierarchy)**

**การโปรแกรมเชิงวัตถุ :** สามารถรองรับลำดับชั้นของคลาสได้ ดังนั้นจึงมีประโยชน์ในกรณีที่ต้องการเพิ่มความสามารถให้กับลำดับชั้นของคลาสหรือเพิ่มคลาสย่อย

**Procedural :** แต่ละคอมโพเนนท์ของแต่ละซอฟท์แวร์จะมีสถานะเท่ากันหมด จึงทำให้เกิดความซ้ำซ้อนได้

**2.2.3.7 การส่งผ่านพารามิเตอร์ (Passing Parameter)**

**การโปรแกรมเชิงวัตถุ :** การส่งผ่านพารามิเตอร์เข้า- ออก จะเป็นการส่งผ่านจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่ง

**Procedural :** การส่งผ่านพารามิเตอร์จะส่งผ่านชนิดของข้อมูลเท่านั้น

**2.2.4 ข้อดีของการเขียนโปรแกรมในเชิงวัตถุ**

**Security:** การโปรแกรมเชิงวัตถุ มีการทำอินฟอร์เมชั่นไฮดิงค์ (Information hiding) เนื่องจากการโปรแกรมเชิงวัตถุ จะเก็บข้อมูลไว้ในออบเจ็กต์ซึ่งจะถูกจัดการโดยเมทธอดที่เหมาะสมเท่านั้น ดังนั้นจึงทำให้ข้อมูลมีความปลอดภัยมากกว่าแบบ procedural

**Reusability :** การโปรแกรมเชิงวัตถุ สามารถนำคอมโพเนนท์ที่สร้างไว้แล้วกลับมาใช้งานได้อีก เช่น อนุญาตให้โปรแกรมที่ถูกสร้างขึ้นมาใหม่สามารถเรียกใช้เมทธอดจากโปรแกรมเดิมได้

**Portability :** เนื่องจากการโปรแกรมเชิงวัตถุ สามารถรองรับคุณสมบัติความหลากหลาย ทำให้ไม่มีปัญหาในการตั้งชื่อเมทธอด ซึ่งสามารถตั้งชื่อซ้ำกันได้เพราะออบเจ็กต์สามารถเลือกตอบสนองเมสเสจโดยใช้เมทธอดที่เหมาะสม

**Reliability and Integrity :** เนื่องจากการโปรแกรมเชิงวัตถุ มีคุณสมบัติในการเอ็นแคปซูเลชั่นจึงทำให้เกิดความผิดพลาดได้ยากและความน่าเชื่อถือสูง

**Flexibility :** เนื่องจากการโปรแกรมเชิงวัตถุ มีคุณสมบัติในการสืบทอดความสัมพันธ์จึงทำให้มีความยืดหยุ่นมาก คือ สามารถเพิ่มความสามารถให้แต่ละคลาสย่อยได้ โดยอาจเพิ่มที่คลาสต้นตระกูลเท่านั้น ทุก ๆ คลาสย่อยก็จะถูกเพิ่มความสามารถนั้น ๆ ไปด้วย

**Understandability :** การโปรแกรมเชิงวัตถุ มีลักษณะของโปรแกรมตรงกับความเข้าใจของผู้ใช้ ซึ่งทำให้ผู้ใช้สามารถเข้าใจการทำงานได้โดยง่าย

**Maintainability :** การโปรแกรมเชิงวัตถุ ง่ายต่อการเพิ่มเติมเมื่อมีความต้องการเพิ่มเติมเมื่อมีความต้องการต่าง ๆ เพิ่มขึ้น ง่ายต่อการแก้ไขและบำรุงรักษาโปรแกรม

**Easy to Development :** ในการพัฒนาการโปรแกรมเชิงวัตถุ ไม่จำเป็นต้องสนใจโค้ดของโปรแกรมว่าเป็นอย่างไร แค่ทราบว่าแค่ส่งเมสเสจนั้นไปในออบเจ็กต์แล้วได้เอาท์พุทออกมาอย่างไรจึงทำให้สามารถพัฒนาโปรแกรมแบบขนานได้

**Visibility :** การโปรแกรมเชิงวัตถุ สามารถตรวจสอบหาข้อผิดพลาดได้ง่าย

**2.2.5 ข้อเสียของการเขียนโปรแกรมในเชิงวัตถุ**

- เนื่องจากการโปรแกรมเชิงวัตถุ เป็นเทคนิคแบบใหม่ ดังนั้นทูลส์ (Tools) ต่าง ๆ จึงยังไม่ค่อยพัฒนามากนัก โดยเฉพาะทูลส์ที่เป็นการเขียนโปรแกรมในเชิงวัตถุอย่างแท้จริง ( Pure Object Oriented Programming )
- ไม่มีสัญลักษณ์ที่เป็นมาตรฐานที่ใช้ในการวิเคราะห์และออกแบบ
- การพัฒนาทางด้านโปรแกรมภาษายังพัฒนาไปไม่มาก
- จากคุณสมบัติการสืบทอดอาจจะให้เกิดปัญหาในเรื่องความสัมพันธ์ระหว่างคลาสได้
- จากคุณสมบัติความหลากหลายอาจทำให้การตรวจสอบโปรแกรมมีความยุ่งยากซับซ้อนขึ้น

# บทที่ 3
# ทฤษฎีและหลักการสถาปัตยกรรม 3 เทียร์

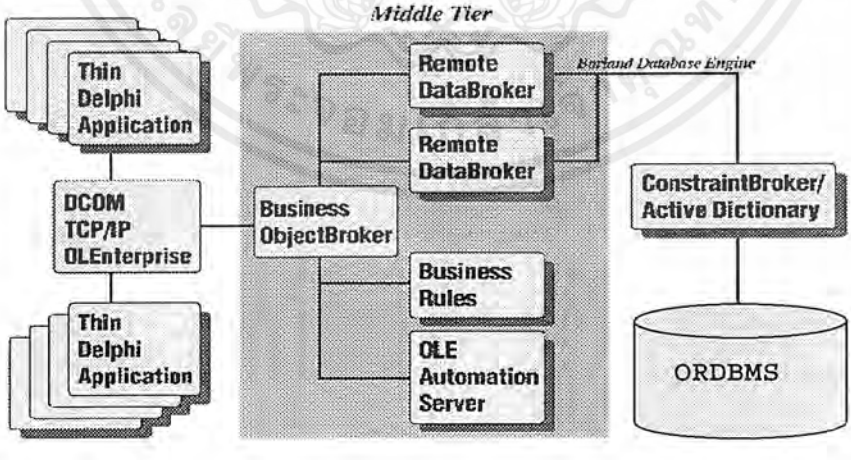
การพัฒนาแอปพลิเคชั่นที่เกี่ยวกับฐานข้อมูลในอดีตนั้น มักจะทำเป็นสถาปัตยกรรมแบบทู-เทียร์ ซึ่งจะประกอบด้วย ไคลเอนต์กับดาต้าเบสเซิร์ฟเวอร์เท่านั้น แต่การใช้สถาปัตยกรรมแบบนี้ดาต้าเบสเซิร์ฟเวอร์จะถูกเข้าถึงจากไคลเอนต์ได้โดยตรง ซึ่งจะเป็นปัญหาในการดูแล และมีผลต่อประสิทธิภาพของระบบด้วย ต่อมาจึงได้ใช้สถาปัตยกรรมแบบมัลติเทียร์

## 3.1 สถาปัตยกรรมของ 3-เทียร์

Multi-tiered Client/Server Application จะแบ่งแอปพลิเคชั่นออกเป็นส่วนย่อยทางลอจิก และให้รันบนคนละเครื่องกัน โดยจะใช้ข้อมูลร่วมกันและติดต่อกันโดยผ่าน LAN หรือ Internet ซึ่งมีข้อดีหลายอย่าง เช่น การเก็บ business rules เอาไว้ที่ส่วนกลาง การใช้ thin client เป็นต้น

รูปแบบของ Multi-tiered application ที่ง่ายที่สุดคือ Three-tiered Model ซึ่งประกอบด้วย

- **ไคลเอนต์แอพพลิเคชั่น** (Client application) : ไคลเอนท์เป็นลักษณะของ thin client
- **แอพพลิเคชั่นเซิร์ฟเวอร์** (Application server) : เซิร์ฟเวอร์จะดูแลการรันแอปพลิเคชั่นที่รันบนเครื่อง และทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่างไคลเอนท์และดาต้าเบสเซิร์ฟเวอร์
- **รีโมทดาต้าเบสเซิร์ฟเวอร์** (Remote database server) : เป็นดาต้าเบสเซิร์ฟเวอร์ ที่จะจัดการกับฐานข้อมูลเพียงอย่างเดียว




*รูปที่ 3-1 แสดงสถาปัตยกรรมแบบ 3-tiered*

3.2 สภาวะแวดล้อมของระบบ

3.2.1 ไคลเอนต์

- เครื่องคอมพิวเตอร์ รันระบบปฏิบัติการ Window 98

3.2.1 แอพอลิเคชั่นเซิร์ฟเวอร์

- เครื่องคอมพิวเตอร์ รันระบบปฏิบัติการ WindowsNT Server 4.0

3.2.2 ดาต้าเบสเซิร์ฟเวอร์

- เครื่องคอมพิวเตอร์ รันระบบปฏิบัติการ WindowsNT Server 4.0

3.3 ข้อเปรียบเทียบระหว่างสถาปัตยกรรมแบบ 3 - เทียร์ และ 2 - เทียร์

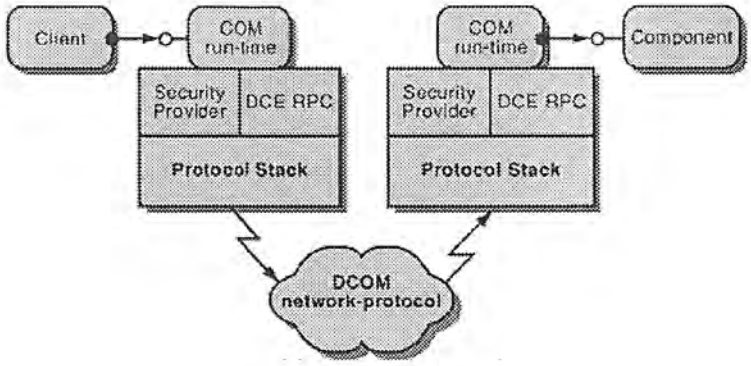
เนื่องจาก multi-tier model จะแบ่งแอพพลิเคชั่นออกเป็นส่วนย่อยทางลอจิก ไคลเอนต์แอพพลิเคชั่นจะทำเพียงการแสดงข้อมูลและการติดต่อกับผู้ใช้ ซึ่งไม่จำเป็นต้องรู้เกี่ยวกับการเก็บหรือดูแลข้อมูล แอพพลิเคชั่น จะรวมการทำงานและจัดการกับการร้องขอหรือการเปลี่ยนแปลงข้อมูลจากไคลเอนท์หลายๆ ตัว โดยจะควบคุมรายละเอียดในการจัดการกับดาต้าเซ็ต (Dataset) และการติดต่อกับรีโมทดาต้าเบสเซิร์ฟเวอร์

ข้อได้เปรียบของ Multi-tiered Model คือ

- รวมการประมวลผลเอาไว้ที่มิดเดิลเทียร์หรือแอพพลิเคชั่นเซิร์ฟเวอร์ โดยไคลเอนท์ทุกตัวสามารถจะเข้าถึงมิดเดิลเทียร์ตัวเดียวกัน ซึ่งจะหลีกเลี่ยงการทำ business rules เอาไว้ในไคลเอนท์แต่ละตัว
- ไคลเอนท์แอพพลิเคชั่นที่เขียนขึ้นจะมีขนาดเล็ก และใช้เป็นฟุตปรินท์(footprint) เพื่อติดต่อไปยังมิดเดิลเทียร์ และง่ายใจการนำมาใช้ โดยไม่ต้องกังวลกับการติดตั้ง การกำหนดค่าและการดูแลซอฟท์แวร์ที่ใช้ในการเชื่อมต่อกับฐานข้อมูล
- กระจายแอพพลิเคชั่นไปยังไคลเอนท์แต่ละตัว ทำให้การทำงานของเซิร์ฟเวอร์มีประสิทธิภาพมากยิ่งขึ้น
- เนื่องจากการทำงานต่างจะต้องผ่านเซิร์ฟเวอร์ ดังนั้นจึงสามารถควบคุมการทำงานและ ทำให้มีความปลอดภัยของข้อมูลด้วย

3.4 Distributed Common Object Model (DCOM)

DCOM เป็นโปรโตคอลที่ยอมให้ซอฟท์แวร์คอมโพเนนท์ทำการเชื่อมต่อกันโดยตรงในเครื่อข่าย ซึ่งมีความน่าเชื่อถือ, ความปลอดภัย และมีประสิทธิภาพ ซึ่งบริษัทไมโครซอฟท์ได้พัฒนาขึ้นมาใช้กับ WindowsNT 4.0 เพื่อนำมาใช้ในการสร้าง Multi-tiered Application โดย DCOM ขยายมาจาก COM ที่มีอยู่ใน Windows 95 อยู่แล้ว โดยที่ COM จะทำให้แอพพลิเคชั่นที่รันอยู่บนเครื่องเดียวกัน สามารถติดต่อกันได้ แต่ DCOM จะทำให้แอพพลิเคชั่นที่รันบนคนละเครื่องสามารถติดต่อกันได้ ภายในเครือข่าย




*รูปที่ 3-2 แสดง DCOM Architecture*

# บทที่ 4
# การสร้าง 3-เทียร์ ไคลเอนท์/เซิร์ฟเวอร์แอพพลิเคชั่น โดยใช้เดลไฟ 4

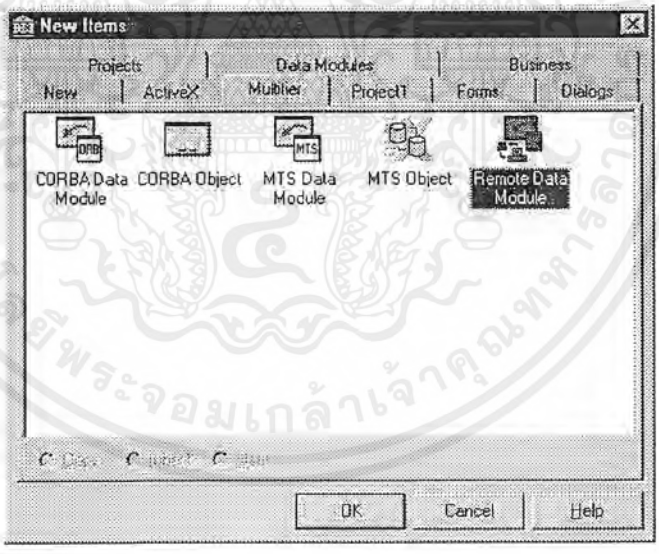
**การสร้าง 3-เทียร์แอพพลิเคชั่นมีขั้นตอนดังนี้**

1. สร้างแอพพลิเคชั่นเซิร์ฟเวอร์
2. รีจิสเตอร์และติดตั้งแอพพลิเคชั่นเซิร์ฟเวอร์
3. สร้างไคลเอนต์แอพพลิเคชั่น

## 4.1 การสร้างแอพพลิเคชั่นเซิร์ฟเวอร์

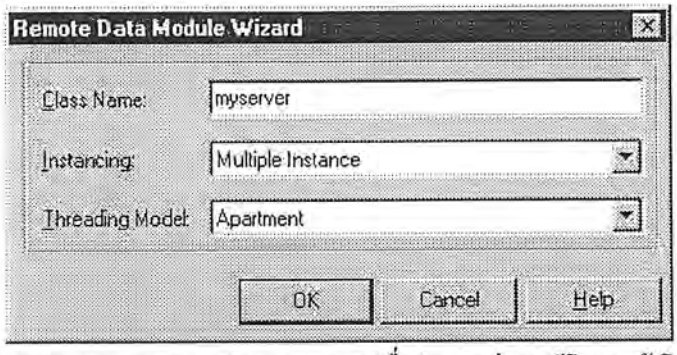
ในการสร้างแอพพลิเคชั่นเซิร์ฟเวอร์ ให้เปิดโปรเจ็กต์ใหม่ขึ้น บันทึกและทำตามขั้นตอนดังต่อไปนี้

1. เพิ่มรีโมทดาต้าโมดูลในโปรเจ็กต์ จากเมนเมนู เลือก File | New เลือกมัลติเทียร์เพจ (Multitier Page) แล้วเลือก Remote Data Module จากมัลติเทียร์เพจ ใน New Item Dialog



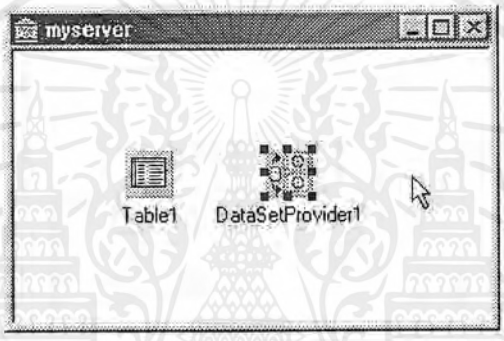

*รูปที่ 4-1 แสดง Multitier Page ใน New Items Dialog เพื่อสร้างรีโมทดาต้าโมดูล*

โดย Remote Data Module จะทำหน้าที่เป็น COM Automation Server เพื่อจัดการการติดต่อของไคลเอนต์เพื่อใช้โพรไวเดอร์ (provider) ที่มีอยู่ในรีโมทดาต้าโมดูลนี้




*รูปที่ 4-2 แสดง Remote Data Module Wizard เพื่อกำหนดค่าของรีโมทดาต้าโมดูล*

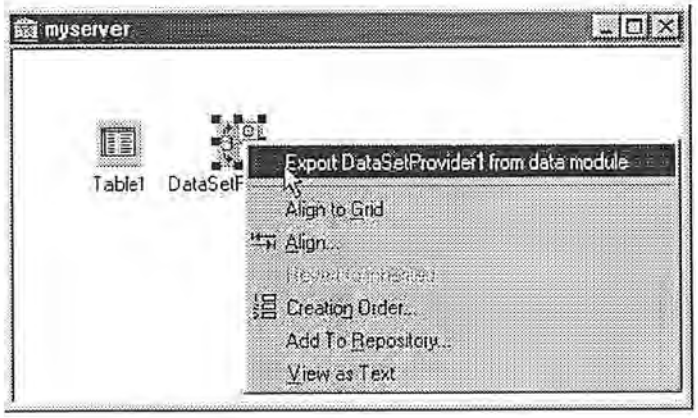
2. วางคอมโพเน็นต์ที่ต้องการเช่น table, query ลงในดาต้าโมดูลและกำหนดคอมโพเน็นต์เหล่านั้นให้เข้าถึงดาต้าเบสเซิร์ฟเวอร์




*รูปที่ 4-3 แสดงรีโมทดาต้าโมดูลในแอพพลิเคชั่นเซิร์ฟเวอร์*

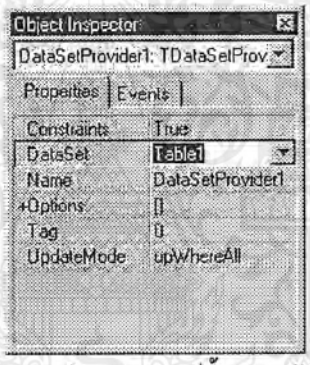
กำหนดพรอพเพอร์ตี้ DatabaseName เป็นชื่อ Alias ที่ตั้งไว้ใน BDE Administrator ที่กำหนดการไดรเวอร์ไปยังดาต้าเบส และกำหนดพรอพเพอร์ตี้ TableName เป็นชื่อตารางในดาต้าเบส

3. วางโพรไวเดอร์คอมโพเน็นต์หรือดาต้าเซ็ตของBDE สำหรับแต่ละดาต้าเซตเพื่อจัดการข้อมูลระหว่างเซิร์ฟเวอร์และไคลเอนต์ โดยจะต้องทำการเอกซ์พอร์ตโพรไวเดอร์(Export Provider) แต่ละตัวในรีโมทดาต้าโมดูล โดยคลิกขวาที่โพรไวเดอร์คอมโพเน็นต์และเลือก Export From Remote Data Module




*รูปที่ 4-4 แสดงการ Export คาต้าโพรไวเดอร์ออกจากรีโมทดาต้าโมดูล*

4. กำหนดคาต้าเซตของโพรไวเดอร์คอมโนเนนท์ โดยเลือกชื่อคอมโพเน็นต์ที่เป็นคาต้าเซตที่อยู่ในรีโมทดาต้าโมดูล เพื่อเข้าถึงข้อมูลของคาต้าเซตนั้นๆ เช่น คอมโพเน็นต์เทเบิล




*รูปที่ 4-5 แสดงการกำหนดค่าพรอพเพอร์ตี้ของดาต้าโพรไวเดอร์*

5. เขียนโค้ดเพื่ออิมพลีเมนท์ (Implement) แอพพลิเคชั่นเซิร์ฟเวอร์ในส่วนของอีเวนต์, การใช้ข้อมูลร่วมกัน, การกำหนดความปลอดภัยและ Business Rule ต่างๆ
6. บันทึก, คอมไพล์ แล้วรันโปรแกรมเพื่อรีจิสเตอร์หรือติดตั้งแอพพลิเคชั่นเซิร์ฟเวอร์

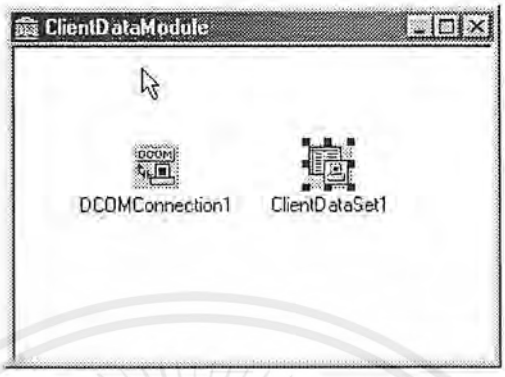
**4.2 การสร้างไคลเอนท์แอพพลิเคชั่น**

การสร้างไคลเอนท์แอพพลิเคชั่นใน 3-เทียร์แอพพลิเคชั่นจะต่างจากไคลเอนท์ใน 2-เทียร์ดังนี้

- คอมโนเนนท์ที่ใช้ในสร้างการเชื่อมต่อไปยังแอพพลิเคชั่นเซิร์ฟเวอร์
- ใช้คอมโพเน็นต์ TClientDataset เพื่อติดต่อกับคาต้าโพรไวเดอร์ในแอพพลิเคชั่นเซิร์ฟเวอร์ ซึ่งจะเชื่อมต่อผ่านไปยังเทเบิลคอมโพเน็นต์หรือคิวรี่คอมโพเน็นต์

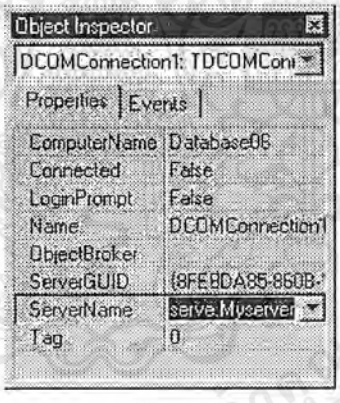
ขั้นตอนการสร้างไคลเอนท์แอพพลิเคชั่น เมื่อเปิดโปรเจกต์ใหม่ขึ้นแล้ว ทำตามขั้นตอนต่อไปนี้

1. เพิ่มดาต้าโมดูลลงในโปรเจกต์ หรือใช้ฟอร์มก็ได้
2. วางคอมโพเน็นต์ที่ใช้ในการเชื่อมต่อ คือ TDCOMConnection ลงในดาต้าโมดูลหรือในฟอร์ม



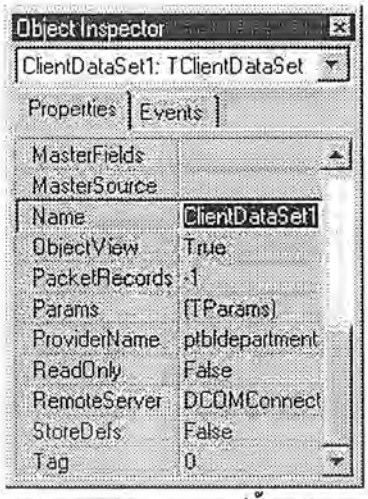

*รูปที่ 4-6 แสดงดาต้าโมดูลที่วางคอมโพเน็นต์การเชื่อมต่อและไคลเอนท์ดาต้าเซ็ต*

3. กำหนดพรอพเพอร์ตี้ของ TDCOMConnection เพื่อเรียกไปยังแอพพลิเคชั่นเซิร์ฟเวอร์




*รูปที่ 4-7 แสดงการกำหนดพรอพเพอร์ตี้ของ DCOMConnection*

4. วางคอมโพเน็นต์ TClientDataset ลงในดาต้าโมดูล และกำหนดค่า RemoteServer เป็นชื่อคอมโพเน็นต์ TDCOMConnection
5. กำหนดพรอพเพอร์ตี้ ProviderName สำหรับคอมโพเน็นต์ TClientDataset แต่ละตัว ซึ่งจะเป็นดาต้าโพรไวเดอร์ที่ทำการเอกซ์พอร์ตจากรีโมทดาต้าโมดูลแล้ว




*รูปที่ 4-8 แสดงการกำหนดค่าพรอพเพอร์ตี้ของไคลเอนท์ดาต้าเซ็ต*

6. เขียนโปรแกรมเพื่อจัดการกับฐานข้อมูลได้เช่น อัพเดทเรคอร์ด โดยใช้คอมโพเน็นต์ที่อยู่ในเพจ Data Control ของคอมโพเน็นต์พาเลตต์ เช่น TDBGrid , TDBEdit โดยกำหนด Datasource เป็นชื่อดาต้าซอร์สที่มีดาต้าเซ็ตเป็นคอมโพเน็นต์ไคลเอนท์ดาต้าเซ็ต

**4.3 การอัพเดทเรคอร์ดในตารางลงฐานข้อมูล**

เมื่อไคลเอนท์แอพพลิเคชั่นถูกเชื่อมต่อกับแอพพลิเคชั่นเซิร์ฟเวอร์แล้วนั้น ไคลเอนท์แอพพลิเคชั่นจะทำงานกับข้อมูลที่ถูกส่งมาจากแอพพลิเคชั่นเซิร์ฟเวอร์แล้วเก็บเอาก๊อปปี้เอาไว้ ผู้ใช้สามารถมองเห็นและแก้ไขข้อมูลเหล่านี้ได้โดยคอมโพเน็นต์ Data Control ต่างๆ ที่อยู่ในเพจ DataControl ของคอมโพเน็นต์พาเลตต์ เช่น TDBEdit การเปลี่ยนแปลงข้อมูลจะถูกเก็บไว้ชั่วคราวใน Change Log ของไคลเอนท์ และไคลเอนท์สามารถจะอัพเดทข้อมูลเหล่านี้ไปยังดาต้าเบส โดยใช้คำสั่ง ApplyUpdates ของคอมโพเน็นต์ TClientDataset

**4.4 การรับค่าพารามิเตอร์มาจากแอพพลิเคชั่นเซิร์ฟเวอร์**

ในการรับส่งค่าพารามิเตอร์ของคอมโพเน็นต์ TQuery หรือ TStoredProc ที่อยู่ในรีโมทดาต้าโมดูลนั้น ทำได้โดยกำหนดพรอพเพอร์ตี้ Params ของคอมโพเน็นต์ TClientDataset โดยกำหนดชื่อพารามิเตอร์ให้เหมือนกับพารามิเตอร์ของดาต้าเซ็ตในรีโมทดาต้าโมดูล

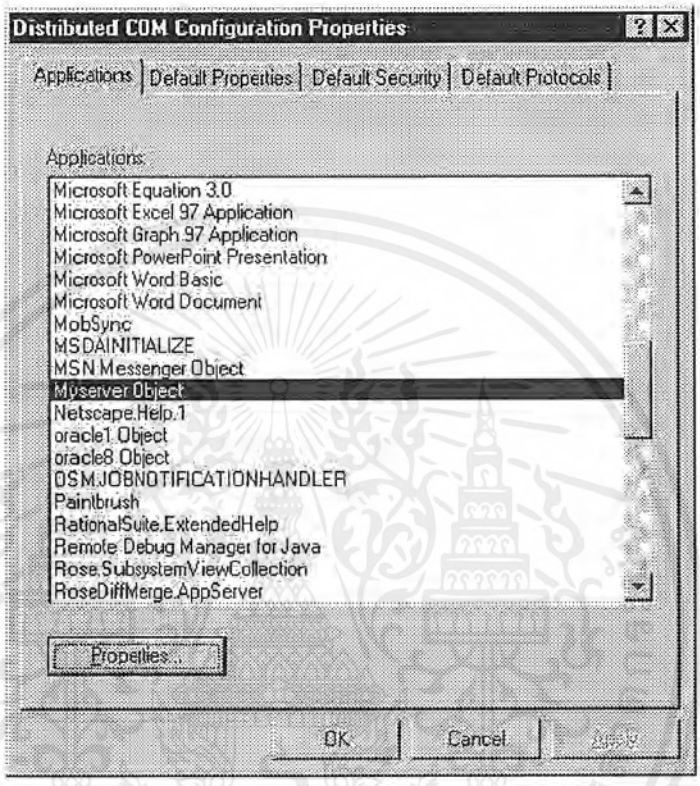
โดยเมื่อต้องการส่งค่าพารามิเตอร์กลับไปยังแอพพลิเคชั่นเซิร์ฟเวอร์ ทำได้โดยเขียนโค้ดดังต่อไปนี้

```
ClientDataset1.Close;
ClientDataset1.Param[0].AsInteger := 1;        // กำหนดค่าให้กับพารามิเตอร์
ClientDataset1.Open;
ClientDataset1.SendParams;                     // ส่งพารามิเตอร์ไปให้แก่แอพพลิเคชั่นเซิร์ฟเวอร์
```

**4.5 การกำหนดค่าคอนฟิกูเรชั่น DCOM บนเครื่องแอพพลิเคชั่นเซิร์ฟเวอร์**

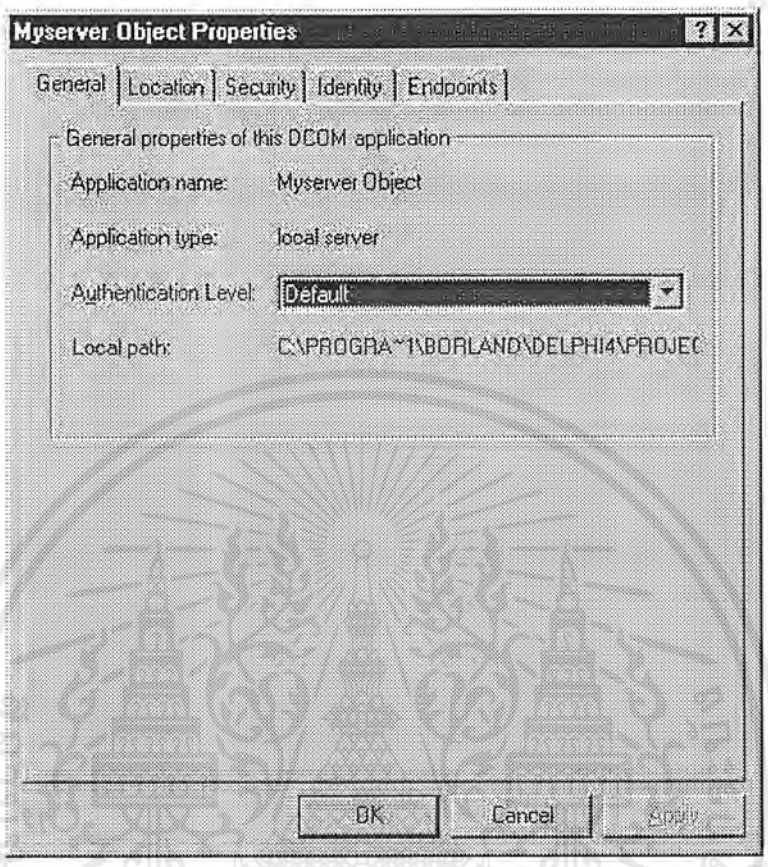
เราจำเป็นต้องมีการตั้งค่าคอนฟิกูเรชั่นให้กับแอพพลิเคชั่นที่เราได้สร้างขึ้นมา และรีจิสเตอร์ไว้บนเครื่อง เพื่อกำหนดสิทธิ์การเข้าใช้งานแอพพลิเคชั่นนั้นๆ โดยมีขั้นตอนดังนี้

1. เรียกเมนูบาร์ สั่งรัน dcomcnfg
2. เลือกชื่อแอพพลิเคชั่นที่เราใช้เป็นเซิร์ฟเวอร์ แล้วคลิกปุ่ม Properties



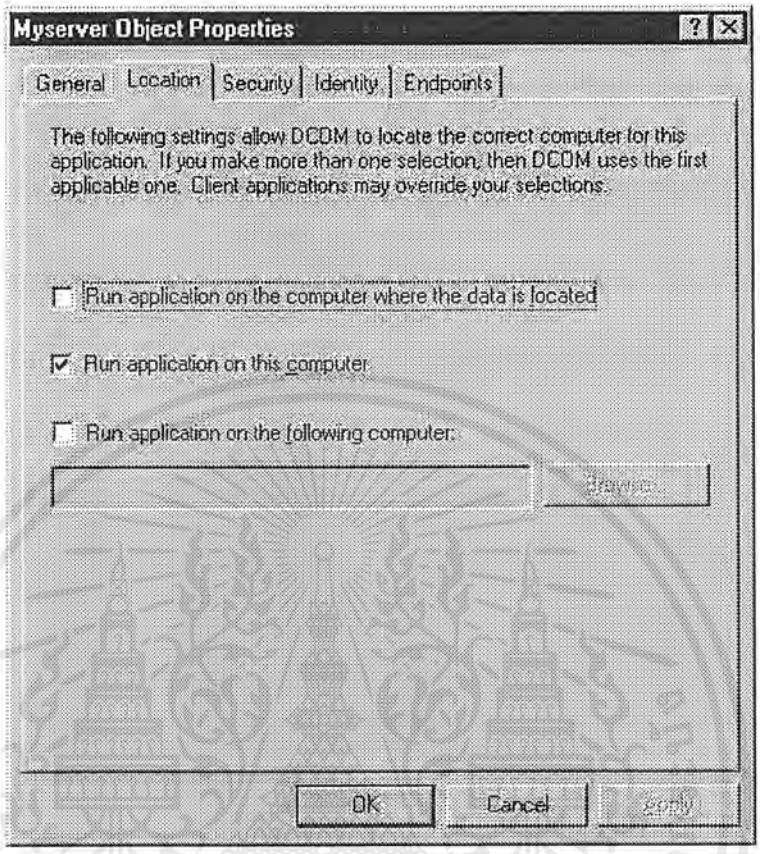

*รูปที่ 4-9 แสดงวินโดว์ของ DCOMCnfg เพื่อกำหนดพรอพเพอร์ตี้ให้กับแอพพลิเคชั่น*

3. ในพรอพเพอร์ตี้วินโดว์จะประกอบด้วย 4 แถบ ในแถบ General เป็นการแสดงรายละเอียดของแอพพลิเคชั่นที่เลือก



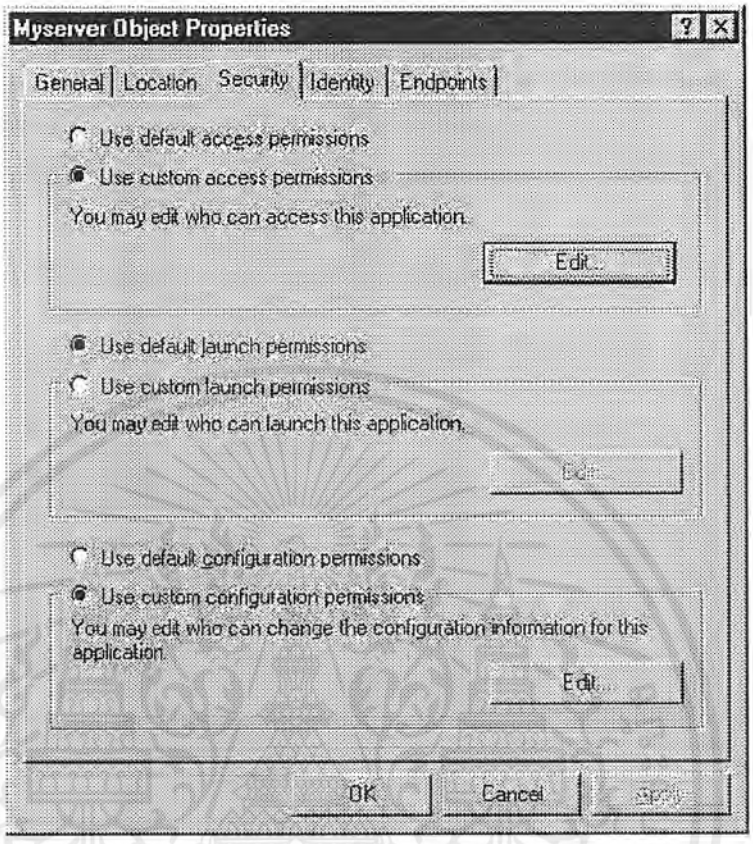

*รูปที่ 4-10 แสดงการรายละเอียดของแอพพลิเคชั่น*

4. แถบ Location เป็นการเลือกเครื่องคอมพิวเตอร์ที่จะทำการรันแอพพลิเคชั่นนี้ ซึ่งเป็นหน้าที่ของเครื่องที่ทำหน้าที่เป็นแอพพลิเคชั่นเซิร์ฟเวอร์



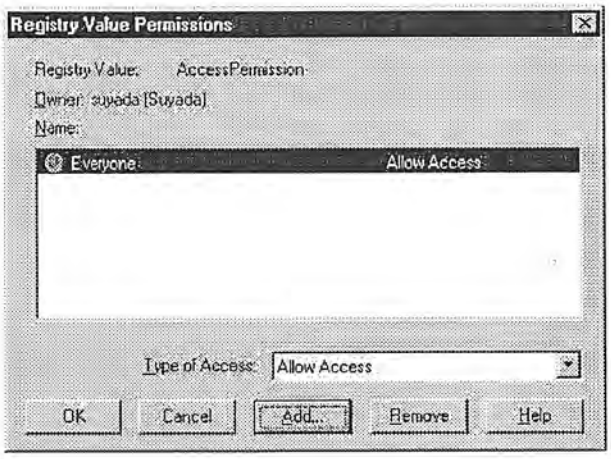

*รูปที่ 4-11 แสดงการกำหนดเครื่องที่ใช้รันแอพพลิเคชั่น*

5. แถบ Security เป็นส่วนการเลือกเพอร์มิสชั่นต่างๆ ว่าจะได้กำหนดแบบดีฟอลต์ของเครื่อง หรือจะเข้าไปกำหนดเองสำหรับแอพพลิเคชั่นนี้โดยเฉพาะก็ได้



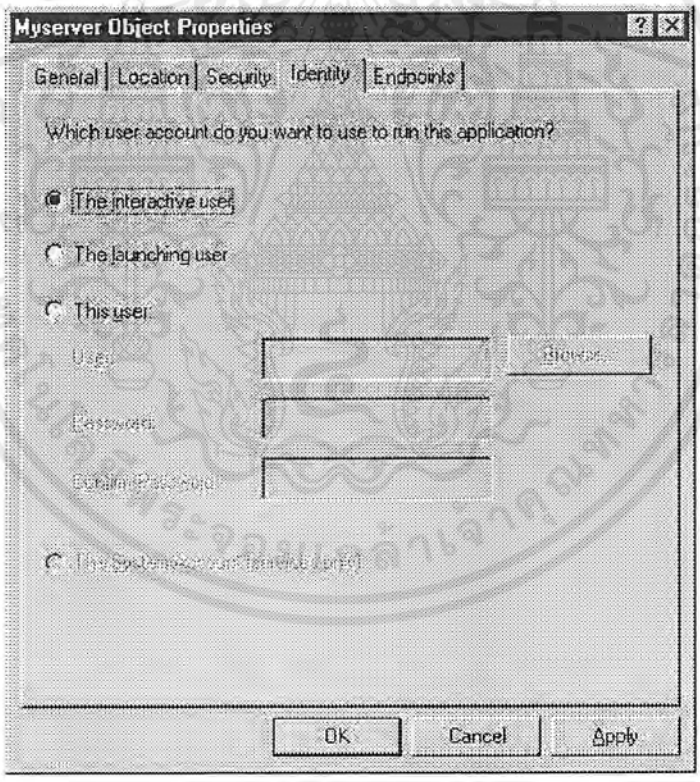

*รูปที่ 4-12 แสดงการกำหนดเพอร์มิสชั่นการเข้าใช้งานแอพพลิเคชั่น*

ถ้าเราเลือกการกำหนดค่าสิทธิการใช้งาน (Permission) เป็นแบบ Custom จะมีหน้าจอให้เพิ่มชื่อผู้ใช้ที่เราจะให้สิทธิในการเข้าใช้แอพพลิเคชั่นนี้ โดยผู้ที่ได้รับสิทธิการใช้งานนั้นต้องเป็นผู้ที่มี User Account อยู่บนเครื่องเซิร์ฟเวอร์ด้วย




*รูปที่ 4-13 แสดงการกำหนดสิทธิการเข้าใช้เครื่องของผู้ใช้*

6. แถบ Identity เป็นการกำหนดชนิดของผู้ใช้ ที่สามารถเข้ามาใช้งาน ควรกำหนดเป็น The interactive user เพื่อรองรับการใช้งานทุกสภาวะ




*รูปที่ 4-14 แสดงการกำหนดชนิดของผู้ที่เข้าใช้แอพพลิเคชั่นนี้*

# บทที่ 5
# ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์

## 5.1 ฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database : ORDB)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ คือ ฐานข้อมูลชนิดที่นำแนวความคิดแบบออบเจ็กต์โอเรียนเต็ด ( Object-Oriented Database) มาผสมผสานกับระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เพื่อทำให้ระบบฐานข้อมูลเชิงสัมพันธ์มีความสามารถทางด้านออปเจ็กต์ได้ โดยระบบฐานข้อมูลเชิงวัตถุสัมพันธ์จะมองข้อมูลและทำการจัดเก็บข้อมูลเป็นแบบตารางโดยมีการจัดการฐานข้อมูลแบบเชิงสัมพันธ์ แต่การติดต่อกับผู้ใช้งานได้นำระบบของออบเจ็กต์โอเรียนเต็ดมาใช้งาน ซึ่งระบบออปเจ็กต์โอเรียลเต็ดจะมีการสนับสนุนการพัฒนาและมีการดูแลระบบฐานข้อมูลขนาดใหญ่ ๆ ได้ดีกว่า

ฐานข้อมูลเชิงวัตถุสัมพันธ์เสมือนการเพิ่มชั้นของออปเจกต์โอเรียลเต็ดครอบบนชั้นของระบบฐานข้อมูลสัมพันธ์ที่มีอยู่

## 5.2 คุณสมบัติของฐานข้อมูลเชิงวัตถุสัมพันธ์

1. มีความสามารถเดิมของฐานข้อมูลเชิงสัมพันธ์
2. สนับสนุนการสร้างออบเจ็กต์ที่มีความซับซ้อน
3. มีความยืดหยุ่นสูงยอมให้สร้างชนิดข้อมูล ฟังก์ชั่นและตัวดำเนินการ(Operator)ใหม่ได้
4. สามารถสร้างแอททริบิวเป็นออบเจ็กต์หรือเป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีกได้

## 5.3 การเปรียบเทียบระหว่างฐานข้อมูลเชิงวัตถุสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

### 5.3.1 ชนิดข้อมูล

ในระบบฐานข้อมูลเชิงสัมพันธ์นั้นแอททริบิวต์ของรีเลชั่นไม่สามารถแยกออกเป็นส่วนๆได้อีก คือต้องเป็นแอททริบิวต์ที่ไม่สามารถแบ่งแยกย่อยได้อีกและมีชนิดของข้อมูลเป็นชนิดที่ค่อนข้างง่าย ได้แก่

- Integer
- Floating-point number
- Character string, fixed or variable length
- Date and time, time interval
- Numeric and decimal

การดำเนินการ ต่างๆบนรีเลชั่นจำกัดอยู่ที่การเรียกดูข้อมูลและการแก้ไข และมีความสามารถจำกัดในการจัดการกับข้อมูลที่เป็นไบนารี่ (Binary) เช่น BLOB(Binary Large Object) จากขีดความสามารถที่จำกัดดังกล่าวส่งผลให้ต้องมีการพัฒนาโมเดลอื่นที่สามารถจัดการข้อมูลที่ซับซ้อนได้

ฐานข้อมูลเชิงวัตถุสัมพันธ์อนุญาตให้มีชนิดของข้อมูลเป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีกหรือเป็นออปเจกต์ได้ดังนั้นชนิดของข้อมูลในฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถเป็นข้อมูลที่มีความซับซ้อนกว่าในรีเลชั่นเช่นข้อมูลที่เป็นอิมเมจ (Image) และข้อมูลชนิดที่ใช้ในงานด้านมัลติมีเดียเป็นต้น ชนิดของข้อมูลที่เพิ่มเข้ามาในฐานข้อมูลเชิงสัมพันธ์ เช่น

- Row Type
- คอลเลคชั่นไทป์ (Collection Type)
- แอบสแทรคดาต้าไทป์ (Abstract Data Type)

**5.3.2 การควบคุมความถูกต้องของข้อมูล (Data Integrity)**

ความสามารถในการทำการควบคุมความถูกต้อง (Integrity Constraint) ของฐานข้อมูลเชิงสัมพันธ์ยึดตามมาตรฐานของ SQL92 ในขณะที่ฐานข้อมูลเชิงวัตถุสัมพันธ์มีความสามารถตามที่กำหนดไว้ในมาตรฐาน SQL3 ดังนี้

- การกำหนดคีย์หลัก (Primary Key)
- Referential Integrity Constraint การกำหนด Foreign Key
- Attribute Constraint การกำหนด Not Null Constraint,Attribute-Based Check,Domain Constraint
- Global Constraint การกำหนด Tuple-Based Check และ Assertion
- Trigger เป็น แอคทีพคอมโพเนนท์ (Active Component) ที่จะเข้ามาทำแอคชั่น (Action) เกิดเหตุการณ์ (Event) ตามที่กำหนดไว้

**5.3.3 ภาษาที่ใช้**

ฐานข้อมูลเชิงสัมพันธ์ใช้มาตรฐาน SQL92 ในการจัดการกับฐานข้อมูล ในขณะที่ฐานข้อมูลเชิงวัตถุสัมพันธ์ใช้ SQL3 ซึ่งเป็น SQL มาตรฐานใหม่ซึ่งเพิ่มความสามารถของแนวคิดเชิงวัตถุเข้าไป ในการสร้าง ADT และเพิ่มความสามารถของการจัดการออบเจ็กต์เข้าไปเช่น การปกป้องข้อมูล, การถ่ายทอดคุณสมบัติ และ โพลิมอร์ฟิซึ่มรวมถึงการเพิ่มการควบคุมความถูกต้องได้แก่ แอสเสิร์ทชั่น (Assertion) และทริกเกอร์ (Trigger)

ชนิดของ Statement ที่เพิ่มเติมใน SQL3

- New Statement เป็นสเตทเมนท์ที่ใช้ในการสร้างADT
- Destroy Statement เป็นสเตทเมนท์ที่ใช้ในการ Destroy ADT
- Assignment Statement เป็นสเตทเมนท์ที่นำ Result SQL ของ Value ไปเก็บไว้ใน Local Variable
- Call Statement เป็นสเตทเมนท์ที่ใช้เรียก SQL Procedure
- Return Statement เป็นสเตทเมนท์ที่ใช้ในการ Return ค่าจาก SQL Function

**5.4 ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle 8**

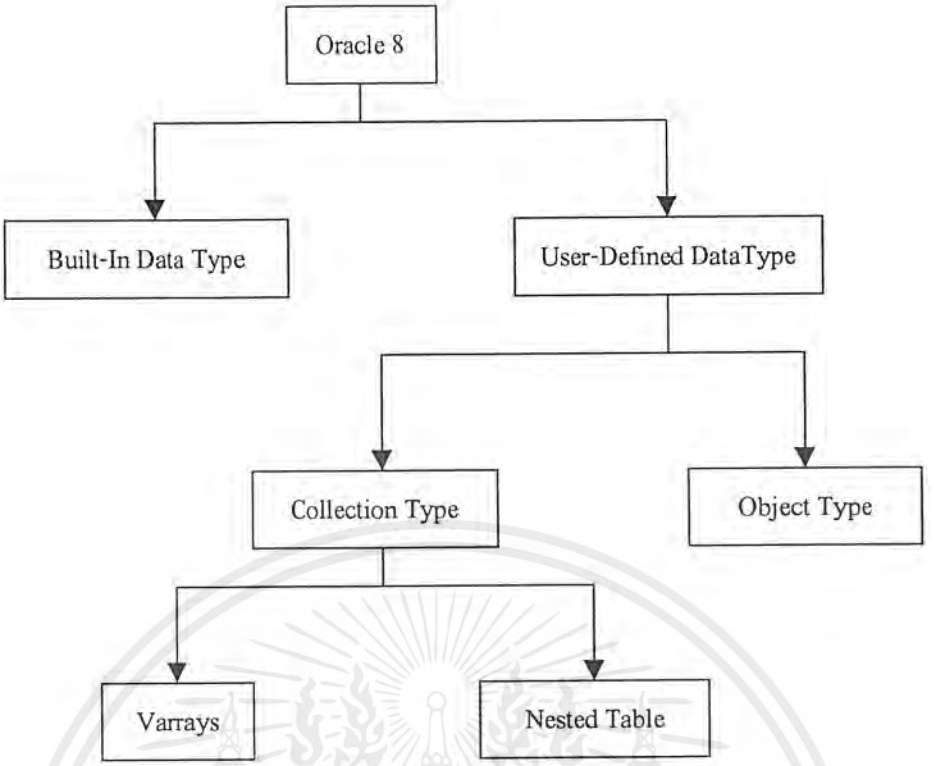
Oracle 8 เป็นระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ ( Object Relational Database Management System : ORDBMS ) ซึ่งเป็นระบบฐานข้อมูลที่สามารถรองรับข้อมูลที่มีความซับซ้อนมาก ๆ เช่น ข้อมูลมัลติมีเดีย ( Multimedia ) , รูปภาพ เสียง เป็นต้น สามารถจัดการกับข้อมูลเหล่านี้ได้อย่างมีประสิทธิภาพ โดย Oracle 8 ได้จัดเตรียมชนิดข้อมูลชนิดใหม่ขึ้นมาเพื่อรองรับกับข้อมูลที่มีความซับซ้อนมาก ๆ เหล่านี้ โดยเฉพาะยอมให้มีข้อมูลชนิดที่ผู้ใช้กำหนดขึ้นมาเอง ( User-Defined Datatypes ) ข้อมูลชนิดนี้จะมีคุณสมบัติทางออบเจ็กต์โอเรียนเต็ด เช่นเดียวกับออบเจ็กต์ในโปรแกรมภาษาที่สามารถเขียนโปรแกรมเชิงวัตถุ ( Object Oriented Programming ) ต่าง ๆ แต่คุณสมบัติบางอย่างอาจไม่เทียบเท่าโปรแกรมภาษาเหล่านั้น

**5.4.1 คุณสมบัติทาง Object Oriented ของ Object type ใน Oracle 8**

- การปกป้องข้อมูล (Encapsulation) Object Type ใน Oracle 8 จะแบ่งการ ปกป้องข้อมูล ออกเป็น 2 ส่วนคือในส่วนของ Specification จะมีการปกป้องแบบ Public และในส่วนของ Body จะมีการปกป้องเป็นแบบ private
- โพลิมอร์ฟิซึม (Polymorphism) Object Type ใน Oracle 8 สามารถมีเมทธอดที่ชื่อซ้ำกันแต่มีการทำงานที่แตกต่างกันได้

**5.5 ชนิดของข้อมูลในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle8 (Oracle8 Datatype)**

ในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle เวอร์ชัน 8.0.5 มีการจัดเก็บชนิดข้อมูลแยกออกเป็นชนิดต่าง ๆ ดังรูปที่ 5-1




*รูปที่ 5-1 แสดงโครงสร้างของ Type ใน Oracle 8*

**5.5.1 Built-In DataType**

Built-In DataType เป็นชนิดของข้อมูลที่มีอยู่ใน Oracle ได้แก่

**5.5.1.1 ข้อมูลชนิด Character**

**5.5.1.1.1 ข้อมูลชนิด CHAR**

ชนิดของข้อมูลประเภท CHAR เป็นชนิดของข้อมูลตัวอักษรที่มีขนาดของความยาวคงที่ โดยที่เมื่อมีการสร้างตารางที่มีคอลัมน์เป็นชนิดข้อมูลประเภท CHAR จะต้องมีการกำหนดความยาวข้อมูลเป็นไบต์ระหว่าง 1 และ 2000 ไบต์ ( โดยจะมีค่าเริ่มต้นเป็น 1 )

**5.5.1.1.2 ข้อมูลชนิด VARCHAR2**

ชนิดของข้อมูลประเภท VARCHAR2 เป็นชนิดของข้อมูลตัวอักษรที่มีขนาดของความยาวเปลี่ยนแปลงได้โดยที่เมื่อมีการสร้างตารางที่มีคอลัมน์เป็นชนิดของข้อมูลประเภท VARCHAR2 จะต้องมีการกำหนดความยาวของข้อมูลที่มากที่สุดเป็นไบต์ โดยสามารถมีค่าได้ระหว่าง 1 และ 4000 ดังนั้นข้อมูลที่เก็บไว้จะมีขนาดเท่าใดก็ได้ที่อยู่ระหว่าง 1 และ ความยาวสูงสุด (แต่จะมีการ return ค่า error ออกมาถ้าหากข้อมูลมีขนาดใหญ่กว่าความยาวสูงสุด )

**5.5.1.1.3 ข้อมูลชนิด VARCHAR**

ชนิดของข้อมูลประเภท VARCHAR เป็นชนิดของข้อมูลที่มีลักษณะเหมือนกับชนิดของข้อมูลประเภท VARCHAR2 แต่อย่างไรก็ตามเวอร์ชั่นของออราเคิลในอนาคต ชนิดของข้อมูล VARCHAR จะมีการเก็บขนาดของข้อมูลที่สามารถเปลี่ยนแปลงได้

**5.5.1.1.4 ข้อมูลชนิด NCHAR และ NVARCHAR2**

ชนิดของข้อมูลประเภท NCHAR และ NVARCHAR2 เก็บข้อมูลประเภท NLS โดยที่ชนิดของข้อมูลประเภท NCHAR เก็บข้อมูลที่มีขนาดคงที่ที่สอดคล้องกับข้อมูลประเภทความยาวคงที่ ( fixed-length national character set ) หรือความยาวเปลี่ยนแปลงได้ ( variable-length national character set ) สำหรับชนิดของข้อมูลประเภท NVARCHAR2 จะเก็บข้อมูลที่มีความยาวเปลี่ยนแปลงได้ ดังนั้นเมื่อมีการสร้างตารางที่มีคอลัมน์ของข้อมูลประเภท NCHAR หรือ NVARCHAR2 จะต้องมีการกำหนดขนาดที่มากที่สุดได้ทั้งหน่วยตัวอักษร ( สำหรับข้อมูลประเภทความยาวคงที่ fixed-length national character set ) หรือหน่วยไบต์ ( สำหรับข้อมูลประเภทความยาวเปลี่ยนแปลงได้ variable-length national character set )

**5.5.1.2 ข้อมูลชนิด LONG**

ชนิดของข้อมูลประเภท LONG จะใช้ในการเก็บข้อมูลที่มีขนาดมากถึง 2 gigabytes

**5.5.1.3 ข้อมูลชนิด NUMBER**

ชนิดของข้อมูลประเภทตัวเลข จะใช้ในการเก็บตัวเลขจำนวนเต็ม ( Fixed numbers) และตัวเลขที่มีจุดทศนิยม (floating-point numbers) เก็บขนาดได้ถึง 21 ไบต์

**5.5.1.4 ข้อมูลชนิด DATE**

ชนิดของข้อมูลประเภทวัน จะเก็บข้อมูลที่เป็นวันและเวลา โดยจะมีการเก็บปี , เดือน , วัน , ชั่วโมง , นาที และวินาที มีขนาดคงที่ 7 ไบต์

**5.5.1.5. ข้อมูลชนิด LOB**

**5.5.1.5.1 ข้อมูลชนิด BLOB**

ชนิดของข้อมูลประเภท BLOB จะเก็บข้อมูลประเภทไบนารี โดยสามารถเก็บข้อมูลไบนารีได้ถึง 4 gigabytes

**5.5.1.5.2 ข้อมูลชนิด CLOB และ NCLOB**

ชนิดของข้อมูล CLOB และ NCLOB จะเก็บข้อมูลประเภท ตัวอักษร ชนิดของข้อมูลประเภท CLOB จะเก็บข้อมูลประเภทที่เป็น Single-byte สำหรับชนิดของข้อมูลประเภท NCLOB จะใช้เก็บข้อมูลประเภท fixed-length หรือ NCHAR โดยชนิดของข้อมูลทั้งสองนี้สามารถเก็บข้อมูล 4 gigabytes

**5.5.1.5.3 ข้อมูลชนิด BFILE**

ชนิดของข้อมูลประเภท BFILE จะเก็บข้อมูลประเภทไบนารีที่เป็นแฟ้มข้อมูลนอกฐานข้อมูล โดยในคอลัมน์ BFILE จะเก็บค่า file locator ที่ชี้ตำแหน่งของแฟ้มข้อมูล โดยชนิดของข้อมูลประเภทนี้สามารถเก็บข้อมูลได้ถึง 4 gigabytes (ชนิดของข้อมูลประเภทนี้จะเป็นอ่านได้อย่างเดียว read-only ไม่สามารถเปลี่ยนแปลงแก้ไขข้อมูลได้ )

**5.5.1.6 ข้อมูลชนิด RAW และ LONG RAW**

ชนิดของข้อมูล RAW จะใช้ในการเก็บข้อมูลไบนารีประเภทเปลี่ยนแปลงขนาดได้ โดยที่ขนาดที่มากที่สุดจะต้องมีการกำหนดไว้ โดยสามารถมีขนาดได้สูงสุดถึง 2000 ไบต์ สำหรับชนิดของข้อมูลประเภท LONG RAW จะใช้เก็บข้อมูลไบนารีประเภทเปลี่ยนแปลงขนาดได้ โดยสามารถมีขนาดของข้อมูลสูงสุดถึง 2^31-1 ไบต์หรือ 2 gigabytes

**5.5.1.7 ข้อมูลชนิด ROWID**

ชนิดของข้อมูลประเภท ROWID จะใช้เก็บข้อมูลประเภทข้อมูลไบนารีที่บอกถึงตำแหน่งของแถว โดยที่มีขนาด 10 ไบต์สำหรับประเภท extended ROWID หรือ 6 ไบต์สำหรับประเภท restricted ROWID

**5.5.1.8 ข้อมูลชนิด MLSLABEL**

ชนิดของข้อมูลประเภท MLSLABEL เป็นชนิดของข้อมูลที่มีอยู่ใน Trusted Oracle โดยจะเก็บค่า Label ที่ถูกสร้างโดย multilevel secure operating Systems ซึ่ง Trusted Oracle จะใช้ Label นี้ในการควบคุมการเข้าถึงฐานข้อมูล

**5.5.2 User-Defined Datatype**

User-Defined Datatype เป็นชนิดข้อมูลที่ผู้ใช้กำหนดขึ้นมาเองซึ่งจะแบ่งออกเป็น 2 ชนิด คือ Object Type และ Collection Type

**5.5.2.1 Object Type**

Object Type ใน Oracle8 เป็นชนิดของข้อมูลที่ผู้ใช้กำหนดขึ้นมาใช้เอง (user-defined) ซึ่งสามารถ encapsulates โครงสร้างของข้อมูลประกอบด้วย function และ procedures ซึ่งเป็นตัวจัดการข้อมูล Object Type ประกอบด้วย 3 ส่วน คือ

- Name เป็นชื่อที่ใช้ในการอ้างถึง Object Type ซึ่งจะไม่ซ้ำกันภายใน schema เดียวกัน
- Attribute เป็นส่วนประกอบของ Object Type Attribute มีชนิดเป็น Built-In datatype หรือชนิดของข้อมูลที่ผู้ใช้กำหนดขึ้นมาเอง (User-define type)
- Methods เป็น function หรือ procedure ซึ่งเขียนโดยใช้ภาษา PL/SQL และเก็บไว้ในฐานข้อมูล หรืออาจเขียนโดยใช้ภาษา C และจะถูกเก็บไว้ข้างนอกฐานข้อมูล

**5.5.2.2 Collection type**

Collection type มีลักษณะที่เป็นหน่วยของข้อมูล (data unit) ที่มีจำนวนสมาชิกไม่จำกัดและสมาชิกทั้งหมดมีชนิดข้อมูลเหมือนกัน Collection type ได้แก่ array type และ table type ซึ่งหน่วยของข้อมูลจะเรียกว่า varrays และ nestable table

Collection type จะมี constructor methods ชื่อของ constructor methods ก็คือชื่อของ collection types นั้น ๆ

**5.5.2.2.1 Varrays**

มีลักษณะเป็นเซ็ตของข้อมูลแบบมีลำดับ(Array)โดยสมาชิกทุกตัวจะเป็นข้อมูลชนิดเดียวกัน สมาชิกแต่ละตัวจะมีตัวชี้ (Index) ซึ่งเป็นหมายเลขที่สอดคล้องกับตำแหน่งของสมาชิกใน array จำนวนของสมาชิกใน array จำนวนของสมาชิกใน array ก็คือขนาดของ array Oracle ยอมให้ขนาดของ array เปลี่ยนแปลงได้ ซึ่งนี่ก็คือสาเหตุที่ถูกเรียกว่า Varrays เราจะต้องระบุขนาดที่ใหญ่ที่สุดเมื่อเรามีการกำหนดข้อมูลชนิด array

**5.5.2.2.2 Nested table**

Nested Table เป็นเซ็ตของข้อมูลแบบที่ไม่เรียงลำดับ ซึ่งสมาชิกทุก ๆ ตัว จะเป็นข้อมูลชนิดเดียวกันและ Nested Table จะมีเพียง คอลัมน์ (column) เดียวและ คอลัมน์ จะเป็นข้อมูลชนิด built-in datatype หรือ เป็น Object type เราสามารถมองเป็นเหมือนกับเป็นตาราง ที่มีหลาย ๆ คอลัมน์ ได้ซึ่ง คอลัมน์ ก็คือ Attribute ของ Object type ตัวอย่างเช่น purchase

การสร้างข้อมูลชนิดตาราง จะไม่มีการจองเนื้อที่ แต่จะเป็นการกำหนดชนิดของข้อมูลซึ่งเราสามารถนำไปใช้เป็น

- ชนิดข้อมูลของคอลัมน์ใน Relational table
- Attribute ของ Object type
- เป็นตัวแปร PL/SQL, parameter หรือ ชนิดของข้อมูลที่ส่งค่ากลับมาจาก function

## 5.6 Oracle 8 กับฐานข้อมูลเชิงวัตถุสัมพันธ์

### 5.6.1 คำศัพท์ที่ควรรู้

**Object Type** เป็นชนิดของข้อมูลที่มีความซับซ้อนและมีความใกล้เคียงกับโลกของความเป็นจริงมากขึ้นซึ่งจะประกอบด้วยข้อมูล (Data) และการกระทำ (Operation) ผู้พัฒนาสามารถพัฒนาได้โดยไม่จำเป็นต้องทราบรายละเอียดภายในของข้อมูลชนิดนี้ และยังสามารถนำข้อมูลชนิดออบเจ็กต์นี้กลับมาใช้ใหม่ได้อีกด้วย

**Object** คือกลุ่มของข้อมูลที่มีความสัมพันธ์กันซึ่งโดยปกติในฐานข้อมูลใน Oracle มักจะแทนด้วยสิ่งที่อยู่ในโลกของความจริงเช่น บุคคล,สถานที่ เป็นต้น หรือมีความหมายอีกนัยหนึ่งคือเป็นอินสแตนซ์ของคลาสหรือออบเจ็กต์ไทป์

**Attribute** คือส่วนของข้อมูลที่อยู่ในออบเจ็กต์ ตัวอย่างเช่น ID, description เป็นต้น ( ใน ระบบ Relational ออบเจ็กต์ คือ แถวในตาราง)

**Method** คือส่วนติดต่อกับข้อมูลซึ่งจะทำการห่อหุ้ม (Encapsulation) ข้อมูลไว้ทำให้ผู้ใช้ไม่จำเป็นต้องรู้รายละเอียดภายใน เพียงแต่เรียกใช้เมทธอดตามที่ต้องการเท่านั้น ( ในระบบ Relational เมทธอด คือ ชุดของ Procedure, Function ที่มีไว้เพื่อติดต่อทำงานกับข้อมูลในตาราง)

**Subclass** คือ คลาสเฉพาะของออบเจ็กต์ไทป์ที่ทำการสืบทอด (Inherit) แอตทริบิวต์และเมทธอดจากคลาสพ่อ และสามารถเพิ่มเติมคุณลักษณะเฉพาะที่ต้องการได้เช่นกัน (Oracle8 ไม่สนับสนุนการสืบทอดทั้งแอตทริบิวต์และเมทธอด)

**Class** คือ Attributes, Methods

**Instance** หรือ Object ของ Object Type คือ ข้อมูล

**Object Type** สามารถใช้งานได้ในลักษณะต่อไปนี้

- สนับสนุนชนิดของข้อมูลที่ผู้ใช้กำหนดขึ้นเอง ที่สนับสนุนการสร้าง ฐานข้อมูลแบบความสัมพันธ์
- สร้าง Nested Table ใน ฐานข้อมูลแบบสัมพันธ์ได้
- ใช้สร้าง Object Tables ที่สนับสนุนการออกแบบฐานข้อมูลแบบวัตถุมากกว่าแบบสัมพันธ์

ข้อได้เปรียบของการใช้ Object Type คือ การสร้าง Nested Table Type ซึ่งจะดีกว่าการ Join Table โดยสามารถ Query ข้อมูลออกมาได้เลยในคราวเดียว

### 5.6.2 การสร้างฐานข้อมูลเชิงวัตถุสัมพันธ์โดยใช้ SQL

#### 5.6.2.1 การกำหนดไทป์

1. VARRAY Type

การใช้ดาต้าไทป์ชนิด VARRAY นั้น จะต้องกำหนดขึ้นเป็นไทป์ โดยใช้สเตทเมนท์ดังต่อไปนี้

```
CREATE TYPE phone_list_t AS VARRAY(10) OF VARCHAR2(20) ;
```

การใช้ VARRAY ต่างจากการใช้ Nested Table คือ Nested Table สามารถทำการคิวรี่ข้อมูลที่เก็บเอาไว้ได้ แต่ VARRAY จะใช้ในกรณีที่ต้องการข้อมูลที่เก็บเอาไว้ทั้งหมด ไม่ต้องมีการคิวรี่

**2. Object Type**

เราสามารถสร้างออบเจ็กต์ไทป์โดยใช้คำสั่งต่อไปนี้

```
CREATE TYPE address_t AS OBJECT
( street VARCHAR2(200), city VARCHAR2(200), state CHAR(2), zip VARCHAR2(20) ) ;
```

และเราสามารถใส่ข้อมูลที่เป็นไทป์อื่นลงในออบเจ็กต์ไทป์ และกำหนดเมธอดให้กับออบเจ็กต์ไทป์ได้ ดังนี้

```
CREATE TYPE customer_info_t AS OBJECT (
custno NUMBER,
custname VARCHAR2(200),
address address_t,
phone_list phone_list_t,
ORDER MEMBER FUNCTION
cust_order(x IN customer_info_t) RETURN INTEGER ) ;
```

**5.6.2.2 การสร้างออบเจ็กต์เทเบิล**

ออบเจ็กต์เทเบิลจะเป็นเทเบิลที่จับกับออบเจ็กต์ไทป์ที่ได้สร้างขึ้น ซึ่งเป็นการประกาศการใช้ข้อมูลแบบออบเจ็กต์ เช่นออบเจ็กต์เทเบิลที่จะใช้จับกับออบเจ็กต์ไทป์ customer_info_t สามารถสร้างได้ดังนี้

```
CREATE TABLE customer_tab OF customer_info_t
(custno PRIMARY KEY);
```

ซึ่งเทเบิลนี้จะมีคอลัมภ์เป็น attribute ของออบเจ็กต์ไทป์ customer_info_t

**5.6.2.3 การสร้าง Nested Table**

Nested Table เป็นการเพิ่มตารางหนึ่งเข้าไปในอีกตารางหนึ่ง โดยใช้ออบเจ็กต์ไทป์ ตัวอย่างการสร้าง Nested Table มีดังต่อไปนี้

สร้างออบเจ็กต์ไทป์ของตารางที่เป็นตารางลูก

```
CREATE TYPE address_type AS OBJECT
```

( street VARCHAR2(200), city VARCHAR2(200), state CHAR(2), zip VARCHAR2(20) ) ;

สร้างเทเบิลไทป์ (Table Type) ของออบเจ็กต์ไทป์ address_type

```
CREATE TYPE address_t AS TABLE OF address_type;
```

หลังจากนั้นสร้างออบเจ็กต์ไทป์ของตารางแม่ ดังนี้

```
CREATE customer_info AS OBJECT
(custno    NUMBER,
custname   VARCHAR2(200),
address    address_t,
phone_list phone_list_t);
```

และสร้างออบเจ็กต์เทเบิล customer_info_t ดังนี้

```
CREATE TABLE customer_info_t Of customer_info
(custno PRIMARY KEY)
NESTED TABLE address STORE AS cust_address;
```

**5.6.2.4 การใช้งาน Nested Table**

การ Insert ข้อมูลลงในตาราง ใช้คำสั่งต่อไปนี้

```
INSERT INTO customer_info_t VALUES
(1,'John Nance',
address_t ( '2 Acocet Drive', 'Redwood Shores', 'CA','95054' ),
phone_list_t( '4155551212' ) );
```

และการ Insert ข้อมูลลงใน Nested Table โดยใช้คำสั่ง THE ดังนี้

```
INSERT INTO THE( SELECT address FROM customer_info_t WHERE custno=3)
VALUES(`55 Madison Ave', 'Madison', 'WI', '53715' );
```

นอกจากนี้การ Update และ Select ข้อมูลของ Nested Table ก็ต้องใช้คำสั่ง THE เช่นกัน ดังนี้

```
SELECT state FROM THE (SELECT address FROM customer_info_t WHERE custno=1);
```

Nested Table จะช่วยลดความซับซ้อนในการ JOIN ตาราง ซึ่งสามารถเรียกข้อมูลจาก Nested Table ได้เลย

# บทที่ 6
# แนวความคิดและการออกแบบแอพพลิเคชั่น สำหรับระบบทรัพยากรบุคคล

## 6.1 Problem Statement

ระบบทรัพยากรบุคคลขององค์กร เป็นระบบที่จะใช้จัดเก็บฐานข้อมูลที่เกี่ยวกับพนักงานขององค์กร โดยระบบสามารถทำงานได้ดังนี้คือ

- ผู้ใช้สามารถเรียกดูข้อมูลต่าง ๆ จากฐานข้อมูลได้ เช่น ข้อมูลประวัติพนักงาน ข้อมูลของพนักงานที่เกี่ยวข้องกับองค์กร ข้อมูลในการทำงานต่าง ๆ เป็นต้น
- ผู้ใช้ระบบ มีดังนี้คือ

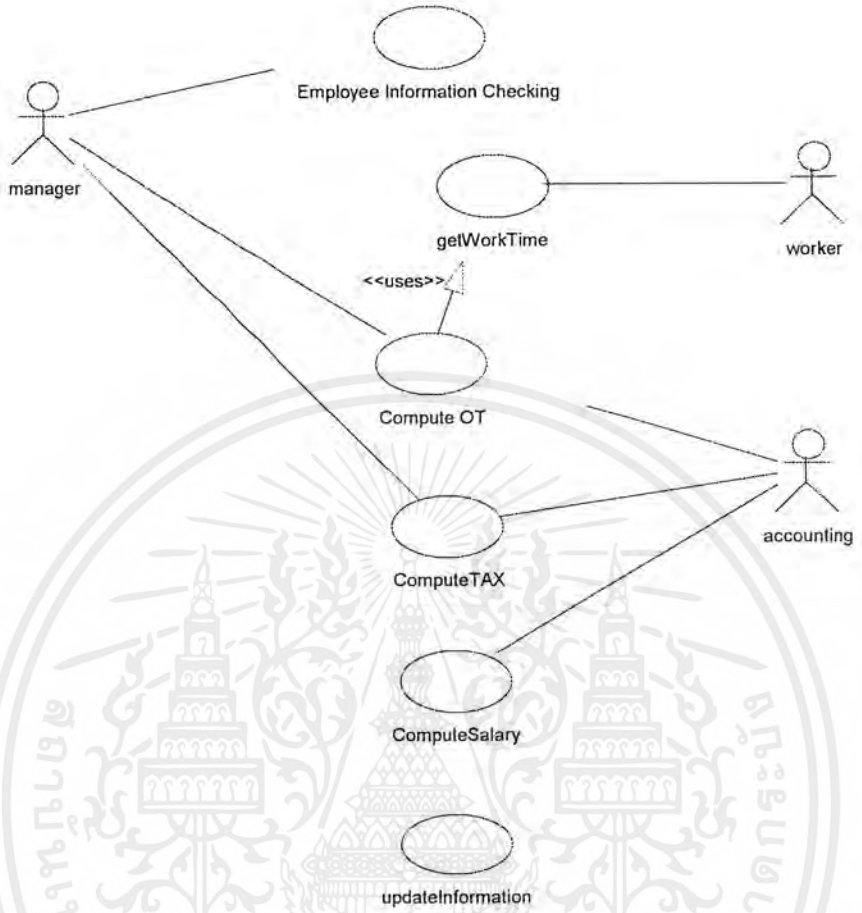
  **Manager :** สามารถเรียกดูข้อมูลต่าง ๆ ของบริษัทได้รวมทั้งข้อมูลที่เป็นความลับ

  **พนักงานบัญชี :** สามารถเรียกดูข้อมูลในส่วนที่เกี่ยวข้องกับการเงินของบริษัท

  **DBA:** เป็นผู้ดูแลฐานข้อมูล รวมทั้งเป็นผู้ทำการ Insert, Delete, Update ข้อมูลภายในฐานข้อมูล

  ผู้ใช้ระบบจะติดต่อกับระบบโดยการใช้งานผ่านเครื่อง Client
- ระบบสามารถคำนวณเงินเดือนของพนักงาน การทำ OT และการคิดภาษีเงินได้ส่วนบุคคลของพนักงานทุกคน

## 6.2 Use Case Diagram

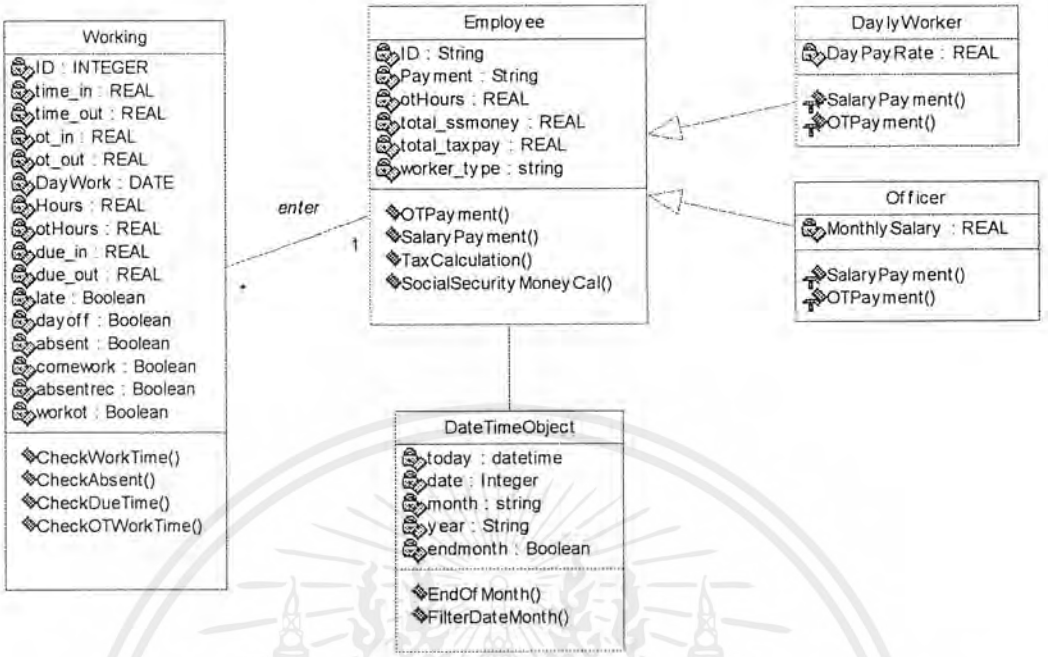



*รูปที่ 6-1 Use Case Diagram ระบบทรัพยากรบุคคล*

Use Case Diagram จะแสดงฟังก์ชั่นของระบบ ซึ่งระบบทรัพยากรบุคคลประกอบด้วยฟังก์ชั่นดังต่อไปนี้ คือ

- ตรวจสอบข้อมูลและประวัติของพนักงาน
- รับเวลาการเข้าออกงานของพนักงาน
- คำนวณเงินค่าทำงานนอกเวลา
- คำนวณเงินภาษี
- คำนวณเงินเดือน
- แก้ไขข้อมูลพนักงานของบริษัท

## 6.3 คลาสไดอะแกรม




*รูปที่ 6-2 คลาสไดอะแกรมของระบบทรัพยากรบุคคล*

คลาสไดอะแกรมของระบบทรัพยากรบุคคล จะประกอบด้วยคลาสดังต่อไปนี้

**คลาส Employee**

มีไว้สำหรับจัดการกับข้อมูลของพนักงาน โดยมีเมทธอดดังต่อไปนี้

- OTPayment() คำนวณเงินค่าทำงานนอกเวลาให้แก่พนักงาน ทุกสิ้นเดือน
- SalaryPayment() คำนวณเงินเดือนให้แก่พนักงานทุกสิ้นเดือน
- TaxCalculation() คำนวณภาษี ณ ที่จ่ายให้พนักงานทุกสิ้นเดือน
- SocialSecurityMoneyCal() คำนวณเงินประกันสังคมให้พนักงานทุกสิ้นเดือน

**คลาส Officer**

เป็นคลาสที่ถ่ายทอดมาจากคลาส Employee สำหรับพนักงานประจำ ซึ่งคิดเงินให้เป็นเงินเดือน โดยจะมีแอตทริบิวต์ MonthlySalary เพิ่มขึ้นมา โดยมีการใช้เมธอด SalaryPayment() เพื่อใช้คิดเงินเดือน ซึ่งเป็นการใช้คุณสมบัติโพลิมอร์ฟิซึมของออบเจ็กต์

**คลาส DaylyWorker**

เป็นคลาสที่ถ่ายทอดมาจากคลาส Employee สำหรับพนักงานที่ทำงานเป็นกะ โดยมีแอตทริบิวต์ DayPayRate และใช้เมธอด SalaryPayment() สำหรับคิดเงินเดือน

**คลาส Working**

คลาสนี้มีไว้สำหรับเก็บเวลาการเข้าออกการทำงานของพนักงาน เพื่อตรวจสอบการทำงาน ซึ่งมีเมธอดดังต่อไปนี้

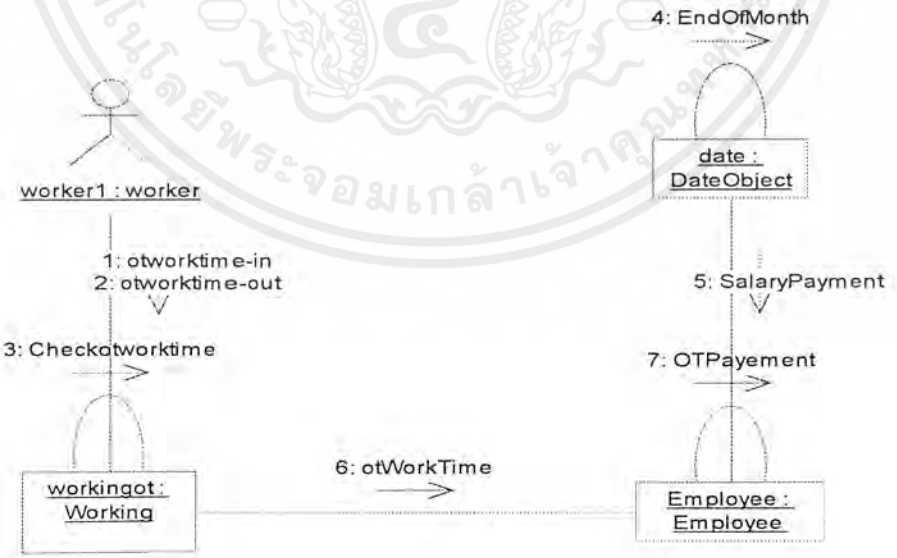
- CheckWorkTime() ตรวจสอบว่าเวลาการทำงานครบตามจำนวนชั่วโมง สำหรับพนักงานที่ทำงานเป็นกะ
- CheckAbsent() ตรวจสอบว่าพนักงานมาทำงานหรือไม่ รวมทั้งตรวจสอบว่ามาทำงานสายหรือไม่ หรือมีการลา โดยตรวจสอบจากตารางที่เก็บข้อมูลการขาด, ลาและมาสาย และถ้ามาสายหรือขาดก็จะอัพเดทลงในตารางขาดหรือมาสาย
- CheckDueTime() ตรวจอสอบเวลาการเข้าออกจากงาน สำหรับพนักงานแต่ละชนิด เช่นถ้าเป็นพนักงานที่ทำงานกะกลางคืน จะเข้างานเวลา 17.00 น. และออกจากงานเวลา 0.00 น.
- CheckOTWorkTime() ตรวจสอบว่าในวันนี้พนักงานทำงานนอกเวลาหรือไม่ และทำเป็นจำนวนกี่ชั่วโมง เพื่อเก็บข้อมูลลงในตารางที่เก็บเวลาการทำงานนอกเวลา

**คลาส DateTimeObject**

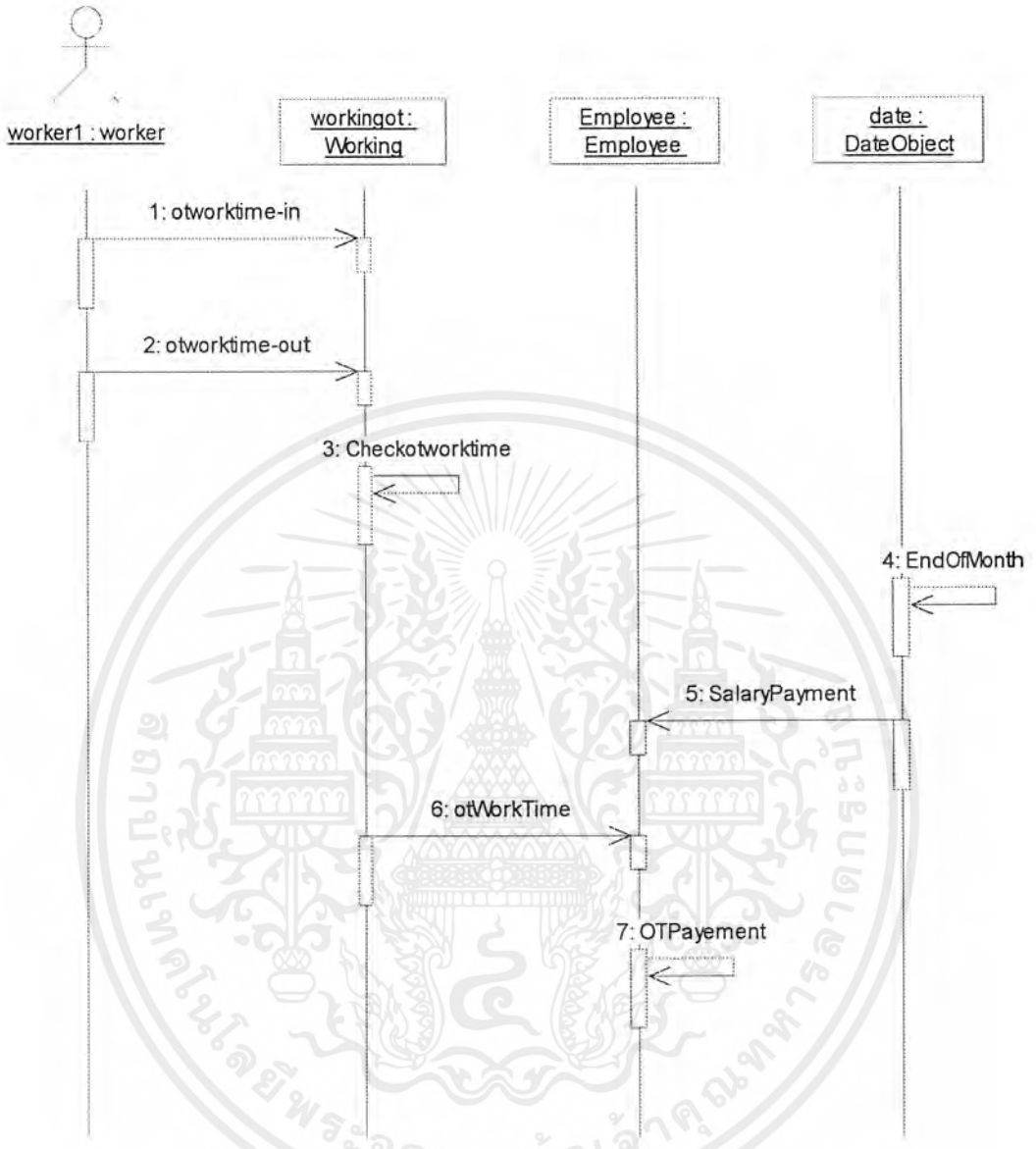
คลาสนี้มีไว้สำหรับตรวจสอบเวลา เพื่อให้ระบบทำงานได้โดยอัตโนมัติ โดยมีเมธอดดังต่อไปนี้

- EndOfMonth() ตรวจสอบว่าเป็นวันสิ้นเดือนหรือไม่ เพื่อให้ระบบจัดการคำนวณเงินเดือนให้แก่พนักงาน โดยส่งเมสเสจให้แก่ออบเจ็กต์ Employee
- FilterDateonth() ตรวจสอบว่าเป็นวัน, เดือน, ปีอะไร

## 6.4 Scenario ของระบบ




*รูปที่ 6-3 แสดง Collaboration Diagram ของการรับเวลาการทำงานนอกเวลาและคำนวณเงินเดือนตอนสิ้นเดือน*




*รูปที่ 6-4 Sequence Diagram ของการรับเวลาการทำงานนอกเวลาและคำนวณเงินเดือนตอนสิ้นเดือน*

**6.5 การออกแบบฐานข้อมูลสำหรับระบบทรัพยากรบุคคล**

ในการออกตารางแบบ Object-Relational นั้น เราจะกำหนดออบเจ็กต์ขึ้นมา โดยถ้าเทียบกับการออกแบบตารางโดยใช้ Entity-Relationship Model นั้น ออบเจ็กต์ก็จะเป็นเอ็นติตี้ (Entity) หลัก ที่มีความสัมพันธ์กับเอ็นติตี้อื่นแบบ n:1 แล้วนำมารวมกันไว้ในออบเจ็กต์เดียวกัน

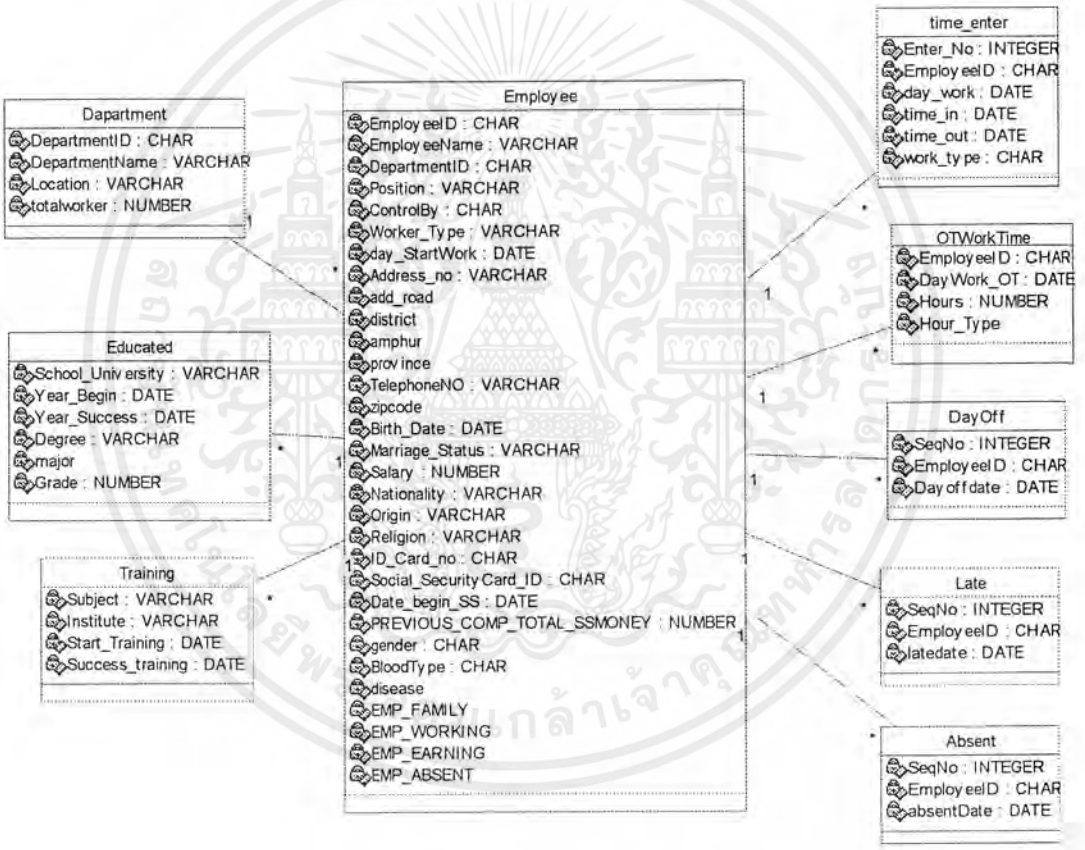
เอ็นติตี้ที่นำมารวมกันไว้ในตารางเดียวกันนี้ จะต้องเป็นเอ็นติตี้ที่จะแสดงคุณสมบัติของออบเจ็กต์ได้ ในการออกแบบตารางของระบบทรัพยากรบุคคลนั้น จะมีออบเจ็กต์หลักคือ พนักงาน ซึ่งจะเก็บข้อมูลต่างๆ ของพนักงาน(Employee) เช่น รหัสพนักงาน, ชื่อสกุล, ที่อยู่ เป็นต้น ซึ่งเอ็นติตี้ที่จะแสดงคุณสมบัติของพนักงานได้ก็คือครอบครัวของพนักงาน, การศึกษา, การฝึกอบรม, ประวัติการทำงาน, ความตั้งใจในการทำงาน(ดูจากการขาด,ลา,มาสาย)


*รูปที่ 6-5 แสดงออบเจกต์ Employee*

ดังนั้นในส่วนข้อมูลของพนักงานนั้น จึงกำหนดให้เป็นข้อมูลแบบออบเจ็กต์ โดยประกอบด้วยออบเจ็กต์ของข้อมูลที่แสดงคุณสมบัติตามที่ได้กล่าวมาข้างต้น ซึ่งจะรวมมาอยู่ในตารางเดียวกัน เพื่อรวมเป็นออบเจ็กต์เดียว โดยใช้ nested table เก็บข้อมูลเหล่านี้ ซึ่งจะประกอบด้วย ตาราง nested table ดังต่อไปนี้

- **MonthlyAbsent** เก็บข้อมูลการขาด,ลา,มาสาย ในแต่ละเดือน
- **MonthlyEarning** เก็บข้อมูลการได้รับเงินเดือนของแต่ละเดือน
- **Family** เก็บข้อมูลบุคคลในครอบครัว ซึ่งประกอบด้วยพ่อ,แม่,สามี และบุตร เป็นต้น
- **Working** จะเก็บข้อมูลประวัติการทำงานของพนักงานเอาไว้

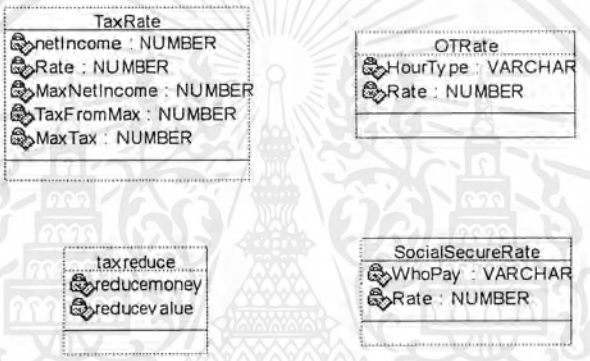
เอ็นติตี้อื่นๆ ที่มีความสัมพันธ์กับออบเจกต์ Employee มีดังต่อไปนี้




*รูปที่ 6-6 แสดงเอ็นติตี้ที่มีความสัมพันธ์กับออบเจ็กต์ Employee*

- ตาราง **Department** : จะเก็บข้อมูลของแผนกงานที่มีในบริษัท
- ตาราง **Educated** : เก็บข้อมูลประวัติการศึกษาของพนักงาน
- ตาราง **Training** : เก็บข้อมูลประวัติการฝึกอบรมของพนักงาน
- ตาราง **Time_Enter** : เก็บข้อมูลเวลาการเข้าออกจากงานในแต่ละวันของพนักงาน
- ตาราง **OTWorkTime** : เก็บข้อมูลการทำงานนอกเวลาของพนักงาน
- ตาราง **Dayoff, Absent และ Late** : เป็นตารางเก็บข้อมูลการขาด, ลา, มาสาย ตามลำดับ ของพนักงานในแต่ละวัน

นอกจากข้อมูลเหล่านี้ ฐานข้อมูลของระบบทรัพยากรบุคคล มีการเก็บค่าตัวแปรต่างๆ ที่เป็นค่าคงที่ ที่จะนำมาใช้ในการทำงานของระบบ เช่นอัตราภาษี ,อัตราลดหย่อนภาษี ซึ่งมีดังนี้




*รูปที่ 6-7 แสดงตารางที่ใช้เก็บค่าคงที่ต่างๆ*

- ตาราง **TaxRate** : เก็บข้อมูลอัตราภาษี
- ตาราง **OTRate** : เก็บข้อมูลอัตราการจ่ายเงินค่าทำงานนอกเวลา
- ตาราง **TaxReduce** : เก็บข้อมูลของเงินค่าลดหย่อนต่าง ๆ ในการคำนวณภาษี
- ตาราง **SocialSecureRate** : เก็บข้อมูลอัตราการเก็บเงินประกันสังคมของพนักงาน

# บทที่ 7
# ผลการดำเนินงาน

การพัฒนาระบบทรัพยากรบุคคล โดยพัฒนาบนสถาปัตยกรรมแบบ 3-เทียร์ไคลเอนต์/เซิร์ฟเวอร์ แอพพลิเคชั่นนั้น จะต้องพัฒนาด้วยกันสามส่วน คือ

1. ฐานข้อมูล เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์
2. แอพพลิเคชั่นเซิร์ฟเวอร์
3. ไคลเอนต์แอพพลิเคชั่น

## 7.1 ฐานข้อมูลของระบบทรัพยากรบุคคล

ฐานข้อมูลของระบบทรัพยากรบุคคล จะเป็นฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยนำข้อมูลชนิดเนสเต็ดเทเบิลและ ออบเจ็กต์ไทป์มาใช้ โดยมีตาราง Employeeinfo เป็นตารางใช้ข้อมูลทั้งสองชนิดนี้

ในการเพิ่มข้อมูลลงในตาราง Employeeinfo นั้น สามารถใช้คำสั่งได้ดังนี้

```
insert into
employeeinfo(employeeid,name,surname,emp_family,emp_working,emp_absent,emp_earning)
values(1,'เทียนชัย','สิริสัมพันธ์',
family_info(family(null,null,null,null,null,null)),
working_info(working(null,null,null,null,null,null,null,null)),
monthly_absent(absent_info(null,null,null,null)),
monthly_earning(earning(null,null,null,null,null,null,null,null)));
```

ซึ่งการเพิ่มข้อมูลลงตารางที่มีเนสเต็ดเทเบิลนั้น จะต้องทำการเพิ่มเร็คคอร์ดลงในเนสเต็ดเทเบิลด้วย เพื่อให้มีตัวชี้ไปยังเนสเต็ดเทเบิล ถ้าไม่ทำการเพิ่มเนสเต็ดเทเบิลลงไป เมื่อเราต้องการเพิ่มเร็คคอร์ดลงในเนสเต็ดเทเบิล จะไม่สามารถใช้งานเนสเต็ดเทเบิลของเร็คคอร์ดนั้นได้เลย เช่น

```
insert into employeeinfo(employeeid,name,surname)
values(22,'สายสุดา','ธนาธรรมนันท์');
```

เมื่อทำการเพิ่มเร็คคอร์ดลงในเนสเต็ดเทเบิล Emp_family ของเร็คคอร์ดนี้ โดยใช้คำสั่ง

```
insert into the(select emp_family from employeeinfo where employeeid=22)
values('กมล ธนาธรรมนันท์','16 ก.พ. 2500','ชาย','ค้าขาย','','บิดา');
```

จะมีเออร์เรอร์เมสเสจ ดังนี้

ERROR at line 1:

ORA-22908: reference to NULL table value

แต่ถ้าเราได้ทำการเพิ่มเร็คคอร์ดของเนสเต็ดเทเบิลเอาไว้ตั้งแต่ตอนเพิ่มเร็คคอร์ดในตาราง Employeeinfo แล้ว เมื่อทำการเพิ่มเร็คคอร์ดของ emp_family ก็จะสามารถเพิ่มเร็คคอร์ดได้

ส่วนตารางอื่น จะจัดการเหมือนกันตารางของฐานข้อมูลเชิงสัมพันธ์

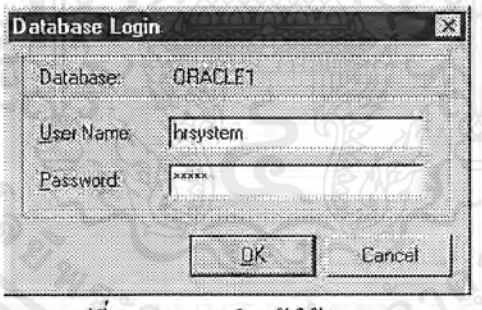
**7.2 แอพพลิเคชั่นเซิร์ฟเวอร์**

แอพพลิเคชั่นเซิร์ฟเวอร์จะประกอบด้วยข้อบังคับทางธุรกิจ (Business Rules) และส่วนที่รอรับการขอเชื่อมต่อจากไคลเอนต์แอพพลิเคชั่น

ข้อบังคับทางธุรกิจของระบบทรัพยากรบุคคล จะประกอบด้วยการคำนวณเงินเดือนพนักงาน , คำนวณเงินค่าทำงานนอกเวลา, การคำนวณเงินภาษีรายได้ส่วนบุคคล, การคำนวณเงินประกันสังคม ,การตรวจสอบเวลาเข้าออกจากงาน ,ตรวจสอบเวลาที่ทำงานนอกเวลา

และส่วนที่จะรอรับการเชื่อมต่อจากไคลเอนต์ จะเป็นรีโมทดาต้าโมดูล โดยจะประกอบด้วยโพรไวด์เดอร์คอมโพเนนท์ ที่จะคอยจัดการข้อมูลที่ไคลเอนต์ต้องการจากฐานข้อมูลให้

โดยเมื่อทำการรันไคลเอนต์แอพพลิเคชั่น เครื่องที่เป็นเซิร์ฟเวอร์จะปรากฏไดอะล็อกบล็อกเพื่อทำการลอกอินเข้าสู่ระบบจัดการฐานข้อมูล ตามรูปที่ 7-1



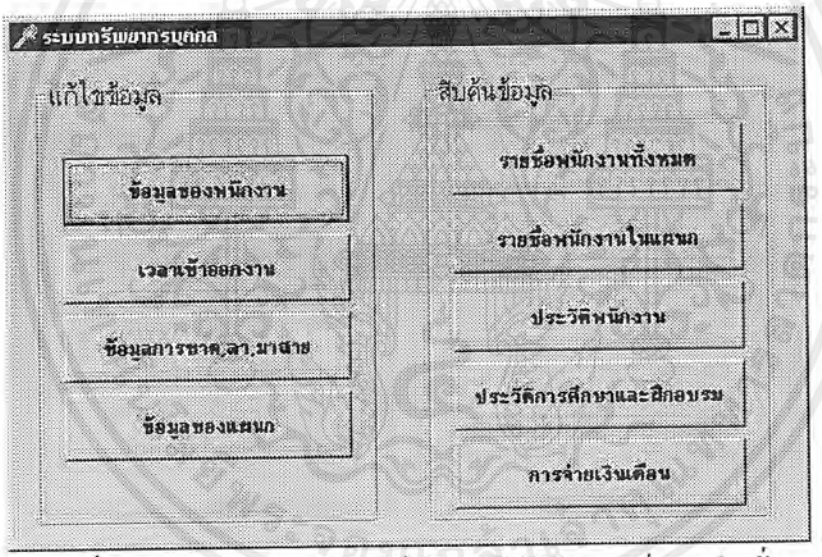

*รูปที่ 7-1 การลอกอินเข้าใช้งานระบบ*

เมื่อทำการใส่พาสเวิร์ดได้ถูกต้อง ก็จะสามารถใช้งานเซิร์ฟเวอร์ของระบบได้ ดังรูปที่ 7-2 โดยเมื่อคลิกปุ่มทำรายงานเงินเดือน ระบบจะทำการจัดการทำรายงานเงินเดือนให้แก่พนักงานทุกคน ซึ่งจะใส่ข้อมูลที่ได้ทั้งหมดลงในตารางในฐานข้อมูล และสามารถเรียกดูรายงานได้จากไคลเอนต์


*รูปที่ 7-2 แสดงหน้าจอของแอพพลิเคชั่นเซิร์ฟเวอร์*

**7.3 ไคลเอนต์แอพพลิเคชั่น**

ไคลเอนท์แอพพลิเคชั่น เป็นอินเตอร์เฟซของระบบสำหรับผู้ใช้ โดยจะประกอบด้วย 2 ส่วนหลัก คือ การแก้ไขข้อมูล และการสืบค้นข้อมูล

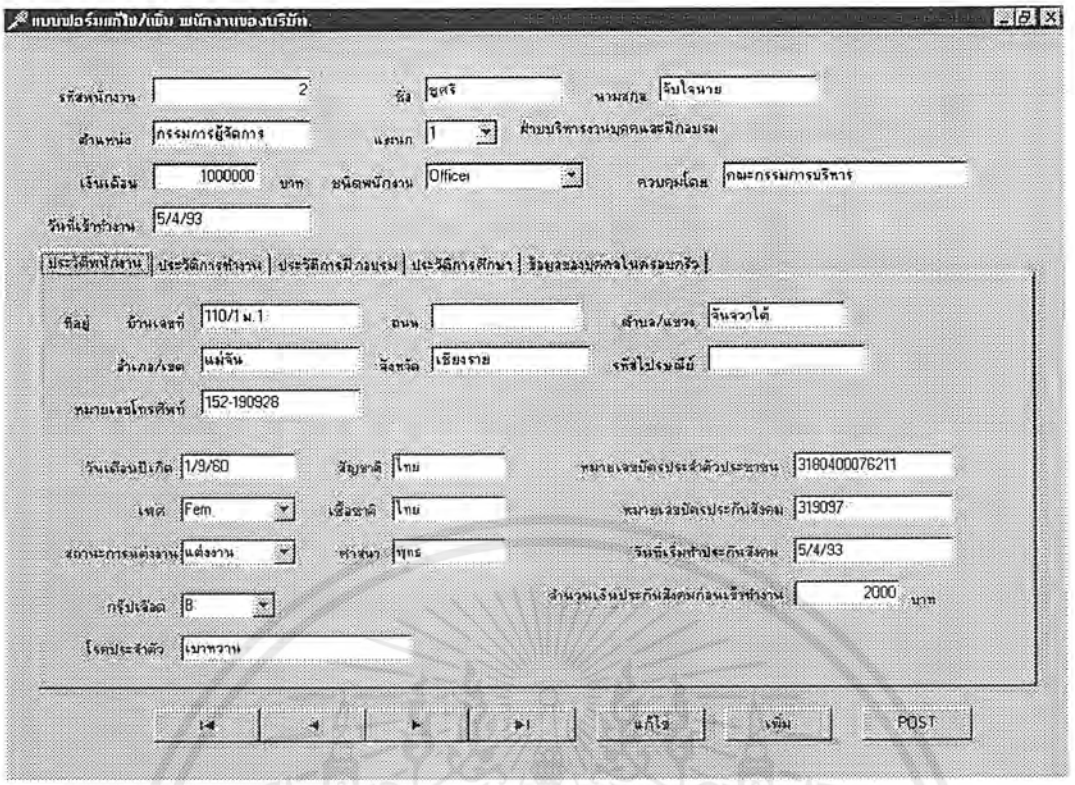



*รูปที่ 7-3 แสดงเมนูหลักของอินเตอร์เฟซของผู้ใช้ในไคลเอนท์แอพพลิเคชั่น*

**7.3.1 การใช้งานในส่วนของการแก้ไขข้อมูล**

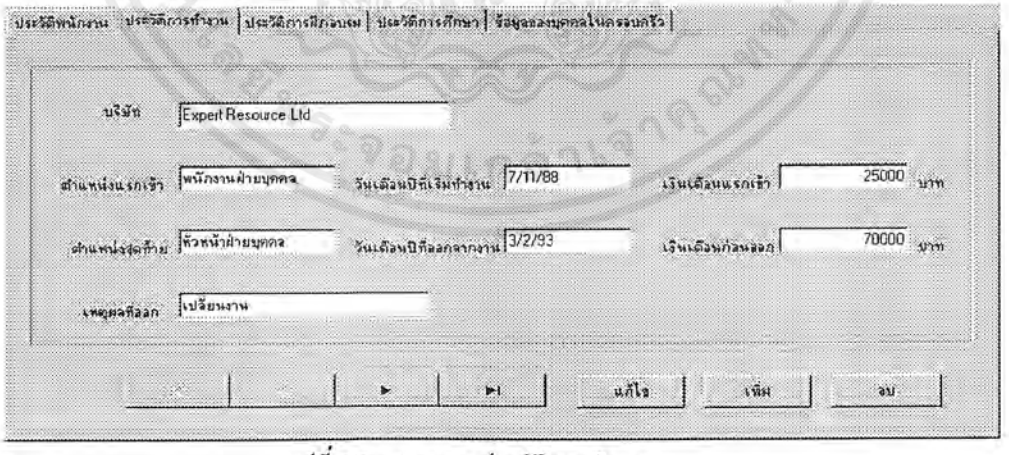
**7.3.1.1 การแก้ไขข้อมูลของพนักงาน**

แบบฟอร์มการแก้ไขข้อมูลของพนักงาน จะใช้สำหรับแก้ไขข้อมูลของพนักงาน และเพิ่มข้อมูลของพนักงานใหม่ลงไป ซึ่งจะประกอบด้วยเพจ ประวัติพนักงาน, ประวัติการทำงาน, ประวัติการฝึกอบรม, ประวัติการศึกษา และข้อมูลของคนในครอบครัว

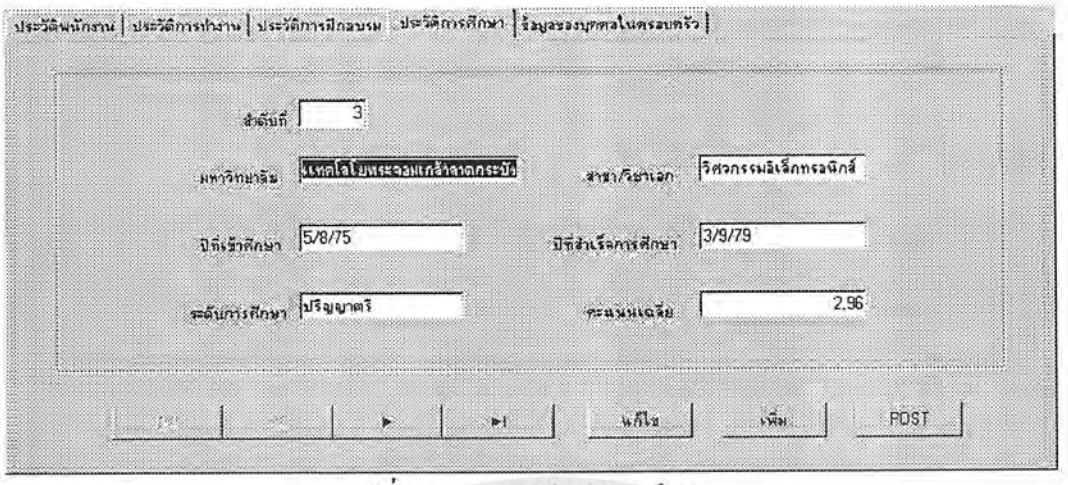



*รูปที่ 7-4 แสดงแบบฟอร์มในการแก้ไขเพิ่มเติมข้อมูลของพนักงาน*

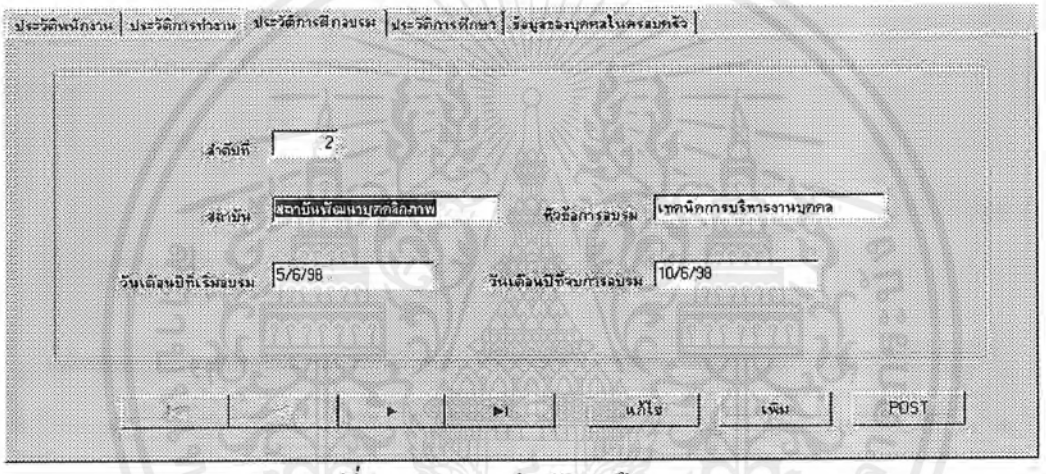
ในเพจประวัติการศึกษา, ประวัติการฝึกอบรม, ประวัติการศึกษา และข้อมูลบุคคลในครอบครัว จะเป็นการเข้าถึงข้อมูลที่อยู่ใน nested table ของตาราง Employeeinfo ซึ่งจะต้องสามารถ Insert ,Delete และ Update ตาราง nested table เหล่านี้ได้



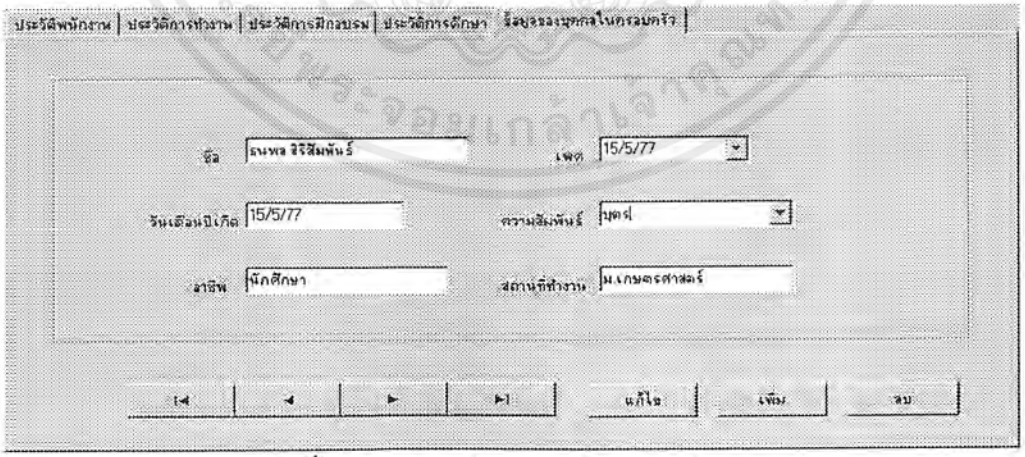

*รูปที่ 7-5 แสดงเพจประวัติการทำงาน*




*รูปที่ 7-6 แสดงเพจประวัติการศึกษา*




*รูปที่ 7-7 แสดงเพจประวัติการฝึกอบรม*

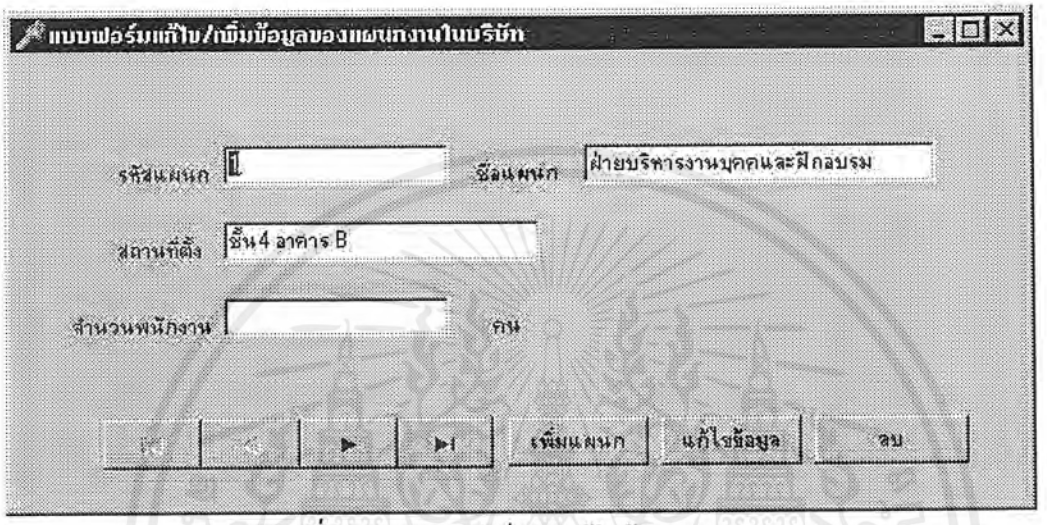



*รูปที่ 7-8 แสดงเพจข้อมูลบุคคลในครอบครัว*

ทุกเพจข้างต้นนี้ จะใช้ปุ่มข้างล่างเพื่อสั่งทำการเซ็ตสถานะของคาต้าเซ็ต เพื่อให้ทำการเพิ่ม, แก้ไข หรือลบได้

**7.3.1.2 การแก้ไขข้อมูลของแผนก**

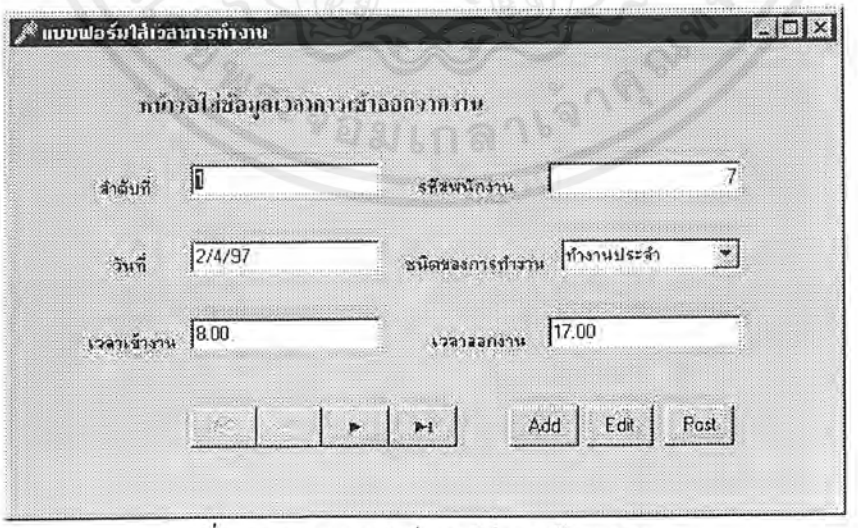
ข้อมูลของแผนกนั้น ปกติแล้วไม่ค่อยจะมีการเพิ่มเท่าไรนัก แต่อาจจะมีการเปลี่ยนแปลงข้อมูลจำนวนพนักงานในแผนก หรือสถานที่ตั้ง




*รูปที่ 7-9 แสดงแบบฟอร์มการแก้ไขข้อมูลของแผนก*

**7.3.1.3 แบบฟอร์มการกรอกเวลาการทำงาน**

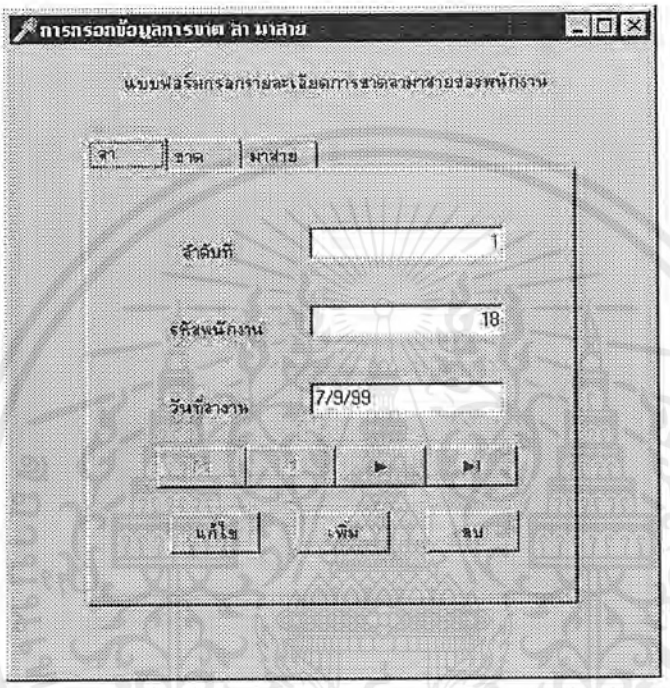
ในการเก็บเวลาเข้าออกจากงานนั้น ในระบบจริงอาจจะมีการใช้วิธีการตอกบัตร ซึ่งก็จะเก็บเวลาเข้าและออกจากงานเอาไว้ แบบฟอร์มการใส่เวลาเข้าออกงานเป็นดังนี้




*รูปที่ 7-10 แสดงแบบฟอร์มการใส่เวลาเข้าออกงาน*

**7.3.1.4 แบบฟอร์มกรอกข้อมูลการลา แก้ไขข้อมูลการขาดหรือมาสาย**

ข้อมูลการขาดและมาสายนั้น ระบบสามารถตรวจสอบได้จากตารางเวลาการเข้าออกจากงาน แต่ข้อมูลการลานั้น จะต้องเป็นการกรอกลงไปในแบบฟอร์มนี้ แต่ทั้งนี้ อาจจะมีความผิดพลาดได้ ในกรณีที่ตรวจสอบผิดพลาด เพราะยังไม่ได้กรอกข้อมูลการลาลงไป จึงต้องมีแบบฟอร์มการขาด และมาสาย สำหรับแก้ไขได้

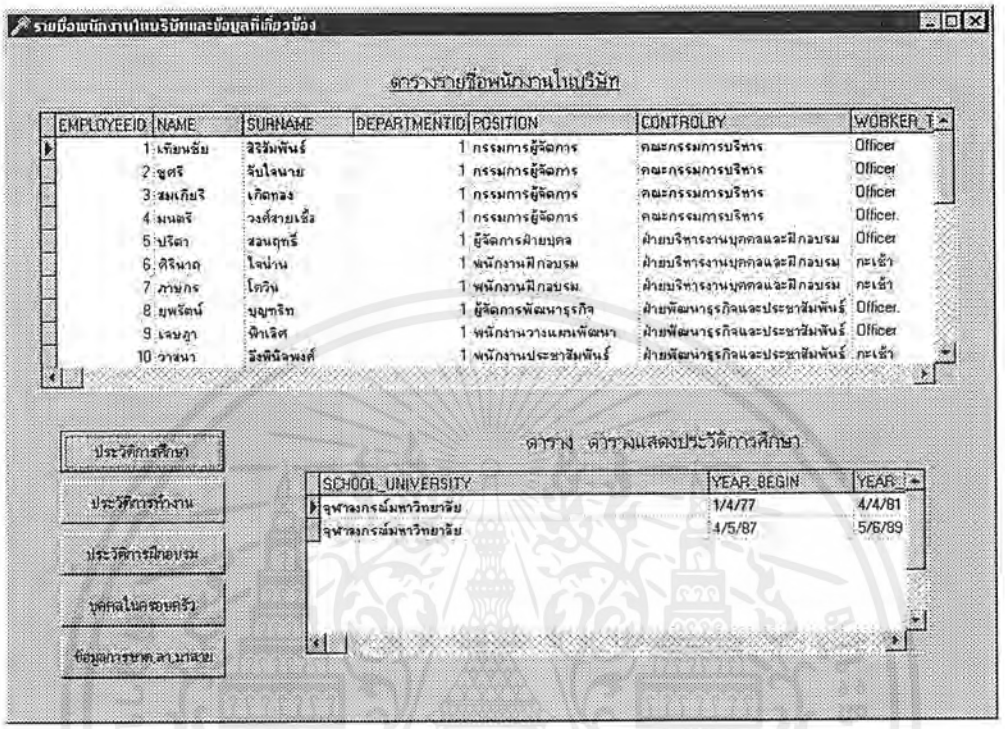



*รูปที่ 7-11 แสดงแบบฟอร์มแก้ไขเพิ่มเติมข้อมูลขาด,ลา,มาสาย*

### 7.3.2 การใช้งานในส่วนของการสืบค้น

#### 7.3.2.1 รายชื่อพนักงานในบริษัท

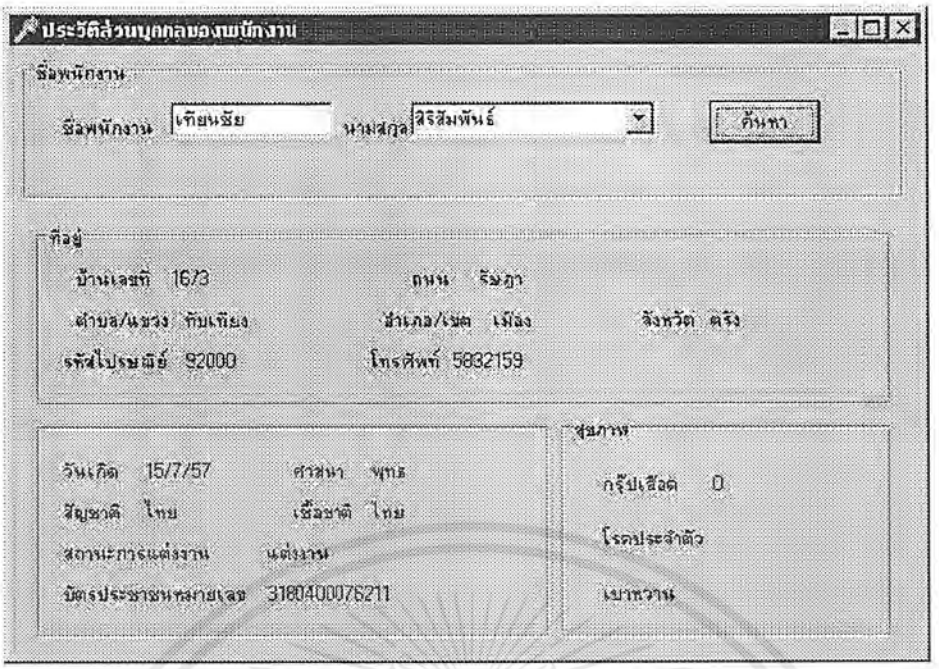
หน้าจอนี้จะแสดงตารางรายชื่อพนักงาน และข้อมูลที่เกี่ยวข้อง นอกจากนี้ ถ้าต้องการรู้ข้อมูลอื่นๆ ก็สามารถกดปุ่มเลือกให้แสดงที่ตารางข้างล่างได้




*รูปที่ 7-12 แสดงรายชื่อพนักงานทุกคนในบริษัท*

#### 7.3.2.2 การสืบค้นประวัติของพนักงานแต่ละคน

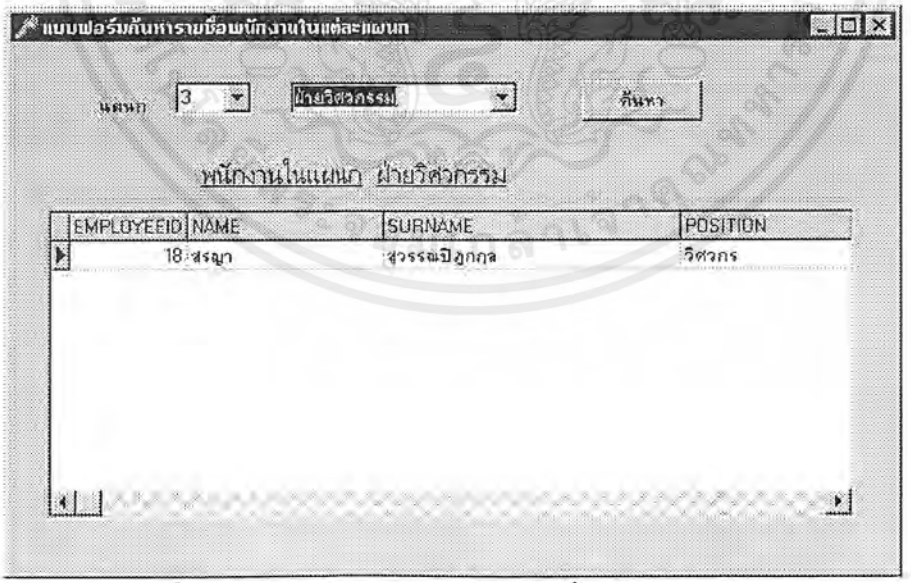
เมื่อต้องการรู้ประวัติ หรือที่อยู่ ของพนักงาน โดยพิมพ์ชื่อพนักงานลงในกล่องข้อความ แล้วไปกดที่คอมโพเนนท์คอมโบบอกซ์นามสกุล ซึ่งจะปรากฏนามสกุลของพนักงานให้เลือก แล้วกดปุ่มค้นหา จะได้ประวัติของพนักงานคนนั้นขึ้นมา




*รูปที่ 7-13 แสดงหน้าจอการสืบค้นประวัติพนักงาน*

### 7.3.2.3 การสืบค้นรายชื่อพนักงานในแต่ละแผนก

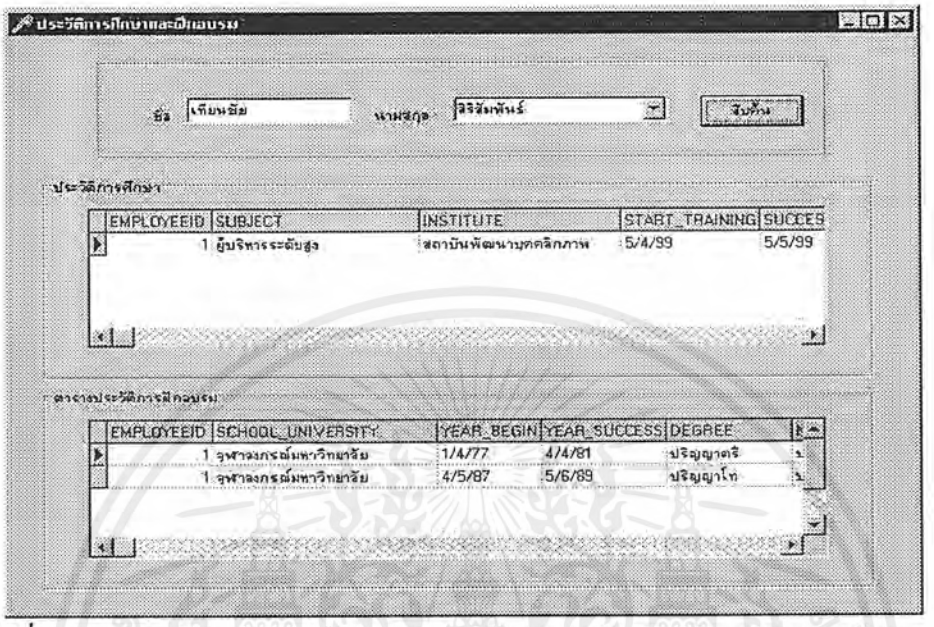
หน้าจอนี้จะใช้สำหรับสืบค้นรายชื่อพนักงานในแผนกที่ต้องการ โดยเลือกรหัสแผนกจากคอมโพเนนท์คอมโบบอกซ์ แล้วคอมโพเนนท์คอมโบบอกซ์อีกตัวจะแสดงชื่อแผนกขึ้นมา แล้วกดปุ่มค้นหา เพื่อให้แสดงรายชื่อพนักงานทุกคนของแผนก




*รูปที่ 7-14 แสดงหน้าจอแสดงการค้นหารายชื่อพนักงานในแผนก*

### 7.3.2.4 การสืบค้นประวัติการศึกษาและการฝึกอบรมของพนักงาน

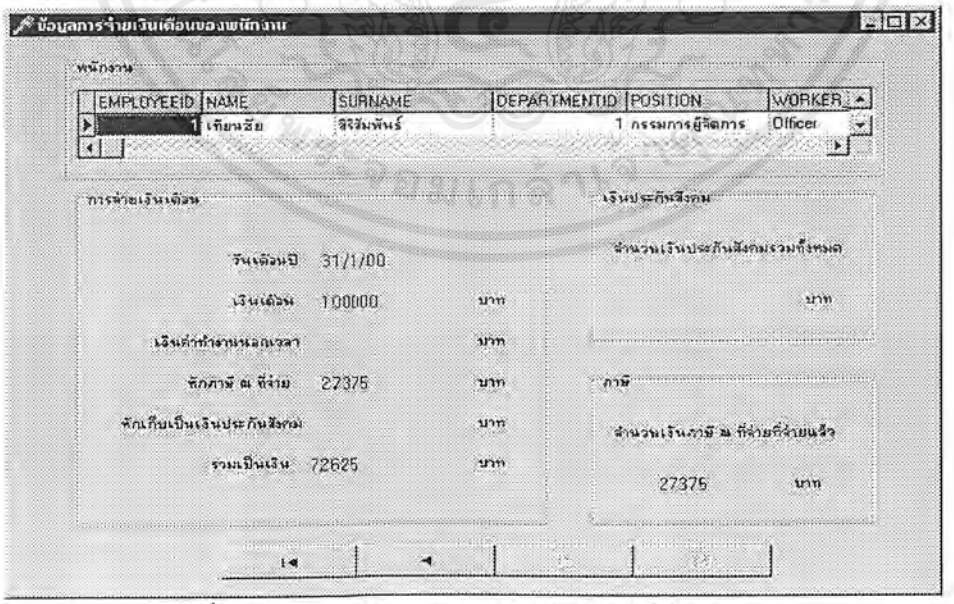
หน้าจอนี้จะใช้เมื่อต้องการค้นหาประวัติการศึกษาและประวัติการฝึกอบรม โดยใส่ชื่อพนักงานลงในกล่องข้อความ แล้วคลิกเลือกนามสกุลที่คอมโพเนนท์คอมโบบอกซ์ แล้วกดปุ่มค้นหา จะได้ประวัติการศึกษาและประวัติการฝึกอบรม เพื่อใช้สำหรับหาพนักงานที่เหมาะสมสำหรับงานแต่ละอย่าง




*รูปที่ 7-15 แสดงหน้าจอแสดงการสืบค้นประวัติการศึกษาและประวัติการฝึกอบรมของพนักงาน*

### 7.3.2.5 ข้อมูลการจ่ายเงินเดือนของพนักงาน

หน้าจอนี้จะใช้แสดงข้อมูลการจ่ายเงินเดือนสำหรับพนักงานทุกๆ เดือน และอาจจะต้องพิมพ์ออกมาเป็นเอกสารเพื่อใช้เป็นหลักฐานด้วย




*รูปที่ 7-16 แสดงหน้าจอแสดงการจ่ายเงินเดือนให้แก่พนักงาน*

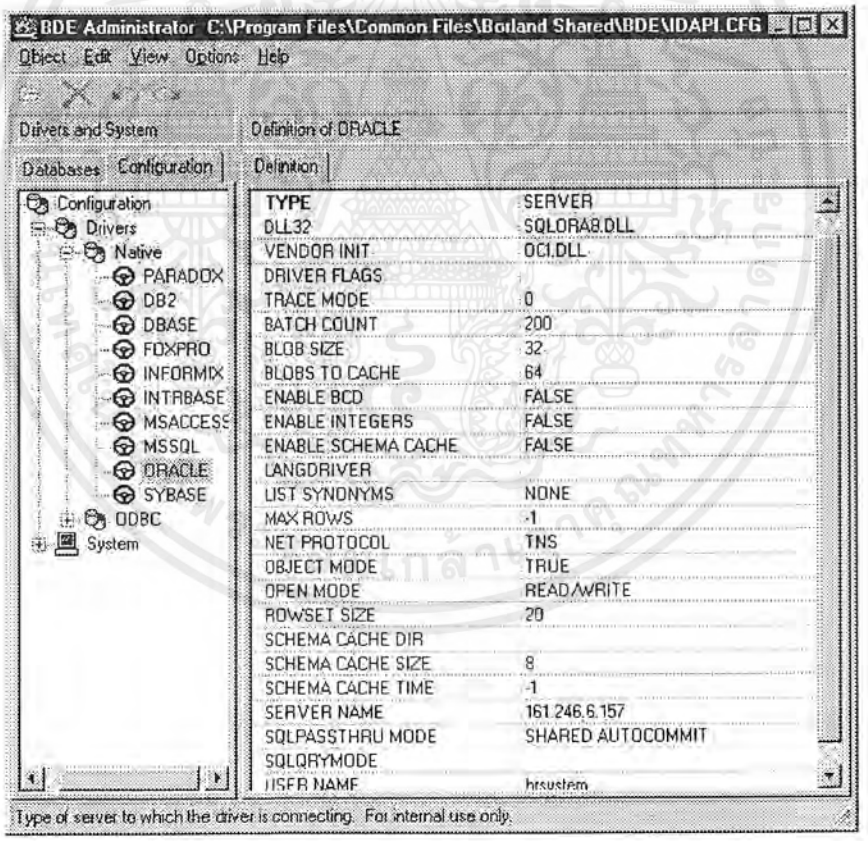
# ภาคผนวก ก
# การตั้งค่า BDE เพื่อทำการเชื่อมต่อแอพพลิเคชั่นของ Delphi เข้ากับฐานข้อมูล Oracle8

**การกำหนดค่าใน Borland Database Engine**

คอมโพเนนท์ของ Delphi ที่ใช้ในการดึงข้อมูลมาใช้นั้น จะใช้ Borland Database Engine (BDE) ในการเข้าถึงฐานข้อมูลที่กำหนดเอาไว้ BDE จะประกอบด้วยไดรฟ์เวอร์ของฐานข้อมูลประเภทต่างๆ โดย Delphi จะรู้จักฐานข้อมูลนั้นๆ ได้จากการกำหนด Alias ซึ่งเป็นชื่อฐานข้อมูลที่กำหนดไว้ โดยเรียกไปยังฐานข้อมูล

การกำหนดค่าใน BDE มีขั้นตอนดังต่อไปนี้

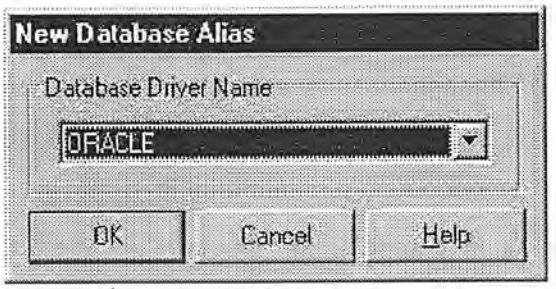
กำหนด Configuration ของฐานข้อมูล โดยกำหนดไดรฟ์เวอร์ของ ORACLE ดังรูป




*รูปที่ ก-1 แสดงการกำหนดค่าให้กับไดรเวอร์ของฐานข้อมูล ORACLE 8*

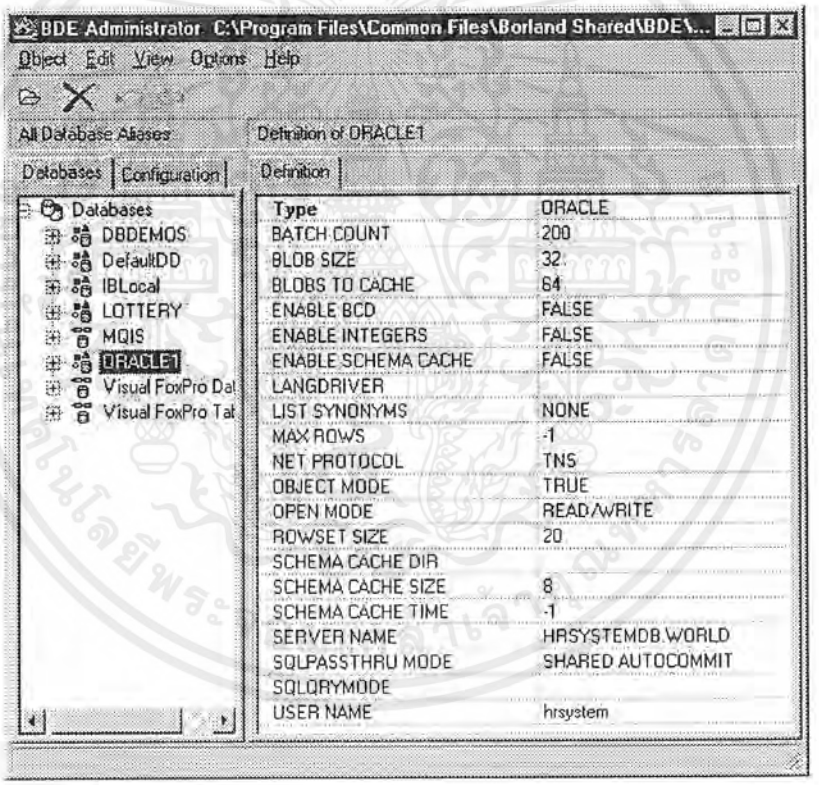
เมื่อกำหนดไดรฟ์เวอร์ของ ORACLE เสร็จแล้วให้ Apply โดยเลือก Object | Apply

หลังจากนั้นกำหนด Alias ในเพจ Database โดยเลือก Object | New เพื่อสร้าง Alias ใหม่ แล้วกำหนดฐานข้อมูลเป็น ORACLE



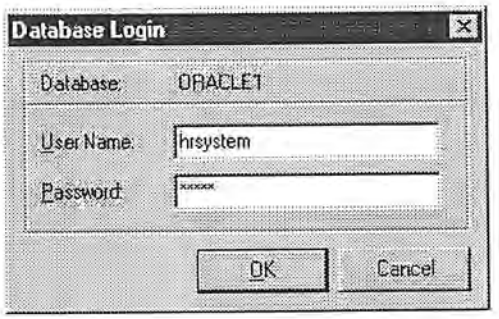

*รูปที่ ก-2 แสดงการเลือกชนิดของฐานข้อมูล*

กำหนดค่าใน Alias ดังรูป



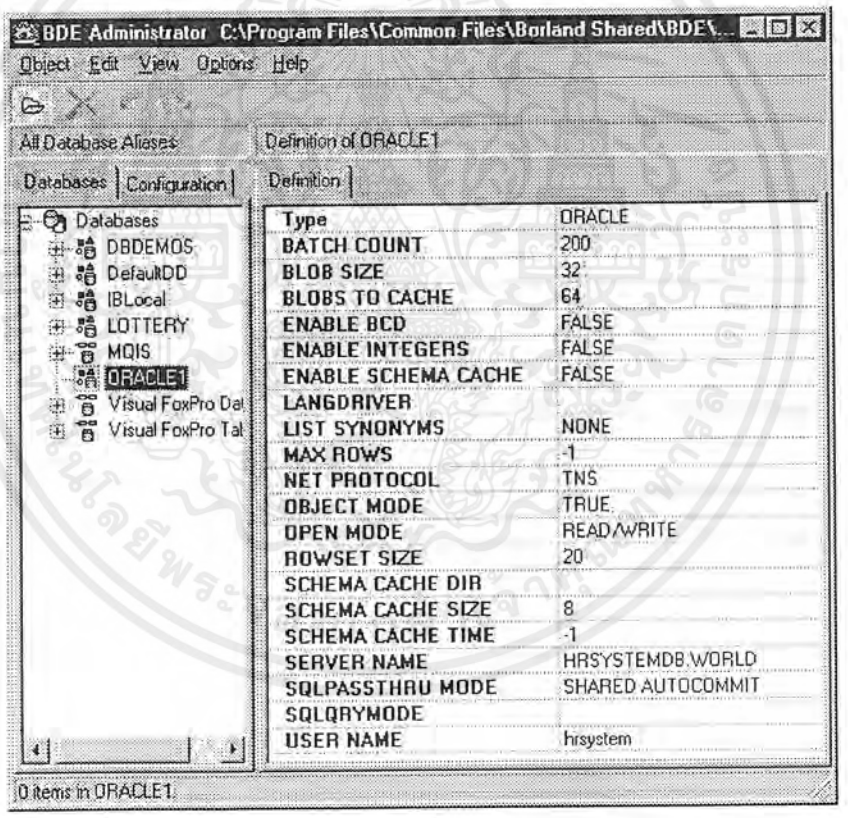

*รูปที่ ก-3 แสดงการกำหนดค่าของ Alias*

แล้ว Apply ทดลองว่าสามารถเชื่อมต่อกับฐานข้อมูลได้จริงหรือไม่ โดยคับเบิลคลิกที่ชื่อ Alias จะขึ้นวินโดว์ให้ login ดังรูป




*รูปที่ ก-4 แสดงการวินโดว์ที่ login ฐานข้อมูล*

เมื่อ BDE เชื่อมต่อกับฐานข้อมูลแล้วจะได้ดังรูป




*รูปที่ ก-5 แสดง BDE เมื่อสามารถลอกอินเข้าสู่ฐานข้อมูลได้*

เมื่อทำการเขียนแอพพลิเคชั่นที่เชื่อมต่อกับฐานข้อมูล ORACLE ก็จะใช้ชื่อฐานข้อมูลตาม Alias ที่ได้ตั้งไว้

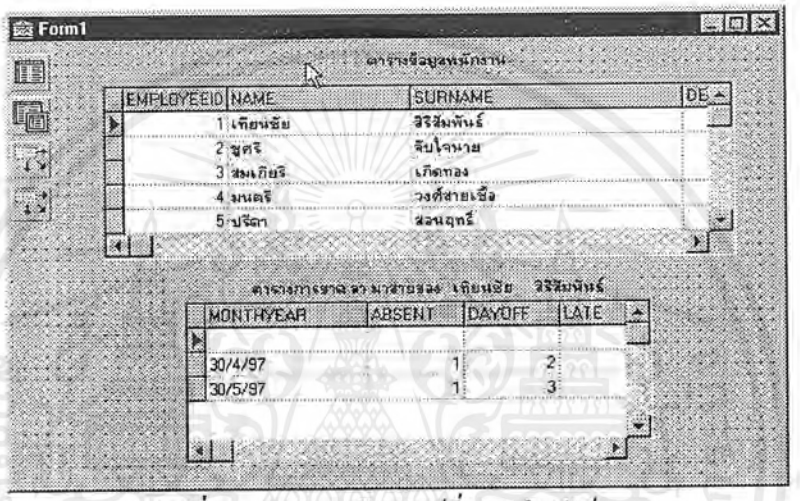
# ภาคผนวก ข

# การใช้งาน Nested Table ใน Delphi

## 1. การพัฒนาบนสถาปัตยกรรมแบบ 2-เทียร์

### 1.1 การสร้างแอพพลิเคชั่นเพื่อเข้าถึงตารางแบบ Nested Table

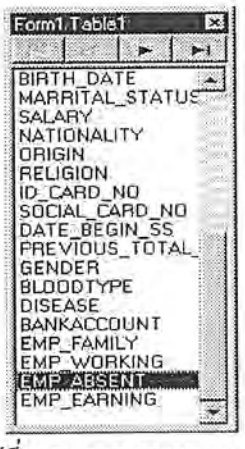
วางคอมโพเนนท์ TTable , TNestedTable , TDataSource และ TDBGrid ลงในฟอร์ม และกำหนดพรอพเพอร์ตี้ดังต่อไปนี้




*รูปที่ ข-1 แสดงคอมโพเนนท์ที่วางลงในฟอร์ม*

**พรอพเพอร์ตี้ของ Table1 : TTable**

- DatabaseName : ชื่อฐานข้อมูล
- TableName : ชื่อตารางที่เป็น nested table
- กำหนด Field ให้กับคอมโพเนนท์เทเบิล โดยคลิกขวาที่คอมโพเนนท์เทเบิล แล้วเลือก Fields Editor จะขึ้นวินโดว์ field editor ของคอมโพเนนท์เทเบิลนั้น แล้วคลิกขวาในวินโดว์ เลือก Add Fields เลือกทุกฟิลด์




*รูปที่ ข-2 แสดง Fields Editor*

**พรอพเพอร์ตี้ของ NestedTable1 : TNestedTable**

- DatasetField : เป็นฟิลด์ที่เป็น nested table ของคอมโพเนนท์เทเบิล Table1

พรอพเพอร์ตี้ของคอมโพเนนท์ Datasource กำหนดพรอพเพอร์ตี้ Datasource เป็นชื่อของคอมโพเนนท์เทเบิล และอีกหนึ่งตัวให้มี Dataset เป็นชื่อของคอมโพเนนท์ nested table วางคอมโพเนนท์ Data Control เช่น TDBGrid แล้วกำหนด Datasource เป็นคอมโพเนนท์ Datasource ที่มีอยู่


*รูปที่ ข-3 แสดงการกำหนดพรอพเพอร์ตี้ของคอมโพเนนท์ Nested Table*

เมื่อต้องการให้ข้อมูลปรากฏบนคอมโพเนนท์คาต้าคอนโทรล ให้กำหนดพรอพเพอร์ตี้ Active ของคอมโพเนนท์เทเบิลและ nested table เป็น True

**1.2 การใช้งาน**

1. การ Insert ข้อมูลลงใน nested table

```
NestedTable1.Last;
NestedTable1.Insert;
NestedTable1.FieldByName('street').AsString := Edit1.Text;
```

```
NestedTable1.FieldByName('city').AsString := Edit2.Text;
NestedTable1.Post;
```

2. การ Edit ข้อมูลใน nested table

```
NestedTable1.Edit;
NestedTable1.FieldByName('street').AsString := Edit1.Text;
NestedTable1.FieldByName('city').AsString := Edit2.Text;
NestedTable1.UpdateRecord;
```

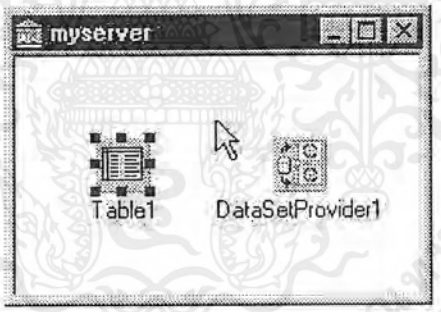
3. การ Delete ข้อมูลใน nested table

```
NestedTable1.Edit;
NestedTable1.Delete;
```

**2. การพัฒนาบนสถาปัตยกรรมแบบ 3-เทียร์**

**2.1 ในส่วนของแอพพลิเคชั่นเซิร์ฟเวอร์**

ในรีโมทดาต้าโมดูล วางคอมโพเนนท์เทเบิลที่เข้าถึงตารางที่เป็น nested table และวางคอมโพเนนท์ DataSetProvider โดยกำหนดพรอพเพอร์ตี้ของคอมโพเนนท์เทเบิลเหมือนกับการใช้บนสถาปัตยกรรมแบบ 2-เทียร์ แล้วเพิ่มฟิลด์ทุกฟิลด์ลงใน Fields Editor




*รูปที่ ข-4 คอมโพเนนท์ที่วางในรีโมทเซิร์ฟเวอร์*

พรอพเพอร์ตี้ของ DatasetProvider ให้มี Dataset เป็นชื่อของคอมโพเนนท์เทเบิล คลิกขวาแล้วเลือก Export DatasetProvider from Remote Data Module แล้วรันโปรแกรม เพื่อรีจิสเตอร์เซิร์ฟเวอร์ลงบนเครื่อง

**2.2 ในส่วนของไคลเอนท์แอพพลิเคชั่น**

วางคอมโพเนนท์ DCOMConnection แล้วกำหนดพรอพเพอร์ตี้ดังต่อไปนี้

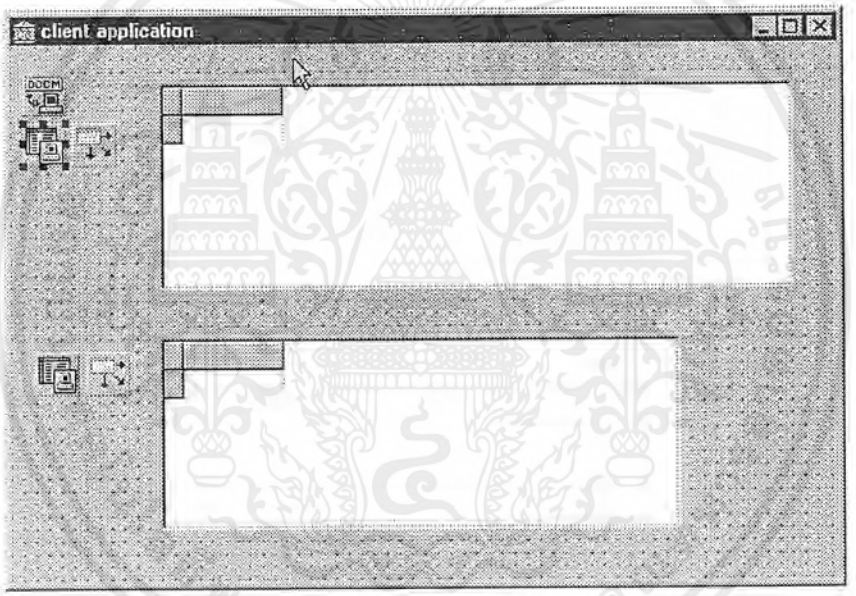
- ServerName เป็นชื่อแอพพลิเคชั่นเซิร์ฟเวอร์
- ComputerName เป็นชื่อคอมพิวเตอร์ที่รันแอพพลิคเชั่นเซิร์ฟเวอร์

วางคอมโพเนนท์ ClientDataset แล้วกำหนดพรอพเพอร์ตี้ดังต่อไปนี้

- RemoteServer เป็นชื่อคอมโพเนนท์ DCOMConnection
- ProviderName เป็นชื่อคอมโพเนนท์คาต้าโพรไวเดอร์ที่อยู่ในรีโมทคาต้าโมดูลของแอพพลิเคชั่นเซิร์ฟเวอร์
- กำหนดฟิลด์ให้กับคอมโพเนนท์ไคลเอนท์คาต้าเซต โดยใช้ Fields Editor เลือกทุกฟิลด์

วางคอมโพเนนท์ ClientDataset อีกหนึ่งตัว เพื่ออ้างถึง nested table โดยกำหนดพรอพเพอร์ตี้ DatasetField เป็นฟิลด์ที่เป็น nested table ของคอมโพเนนท์ไคลเอนท์คาต้าเซตที่เป็นตารางใหญ่

กำหนดพรอพเพอร์ตี้ Dataset ของคอมโพเนนท์คาต้าซอร์ส เป็นคอมโพเนนท์ไคลเอนท์คาต้าเซต กำหนดพรอพเพอร์ตี้ Datasource ของคอมโพเนนท์ DBGrid เป็น Datasource




*รูปที่ ข-5 แสดงคอมโพเนนท์ที่วางบนฟอร์มของไคลเอนท์แอพพลิเคชั่น*

**2.3 การใช้งาน**

1. การ Insert

```
Nested_ClientDataset.Last;
Nested_ClientDataset.Insert;
Nested_ClientDataset.FieldByName('street').AsString := Edit1.Text;
Nested_ClientDataset.FieldByName('city').AsString := Edit1.Text;
Nested_ClientDataset.Post;
ClientDataset1.ApplyUpdates(100);
```

2. การ Edit

```
Nested_ClientDataset.Edit;
Nested_ClientDataset.FieldByName('street').AsString := Edit1.Text;
Nested_ClientDataset.FieldByName('city').AsString := Edit1.Text;
Nested_ClientDataset.UpdateRecord;
ClientDataset1.ApplyUpdates(100);
```

3. การ Delete

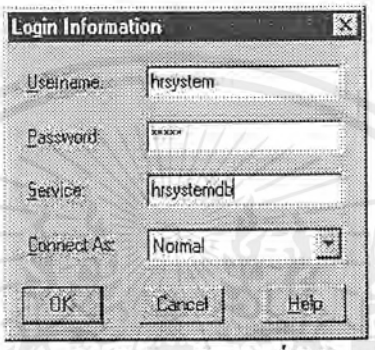
```
Nested_ClientDataset.Edit;
Nested_ClientDataset.Delete;
ClientDataset1.ApplyUpdates(100);
```

# ภาคผนวก ค

# การสร้าง Nested Table โดยใช้ Oracle Schema Manager

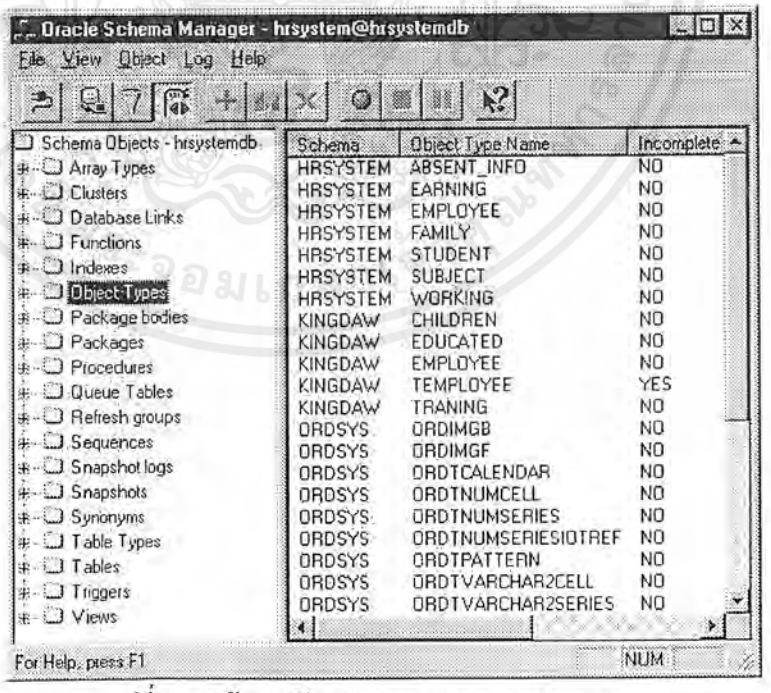
การใช้ Oracle Schema Manager นั้น จะช่วยให้สามารถสร้างตารางได้เร็วขึ้น และยังสามารถสร้างคำสั่ง SQL ออกมาให้โดยอัตโนมัติ

การใช้งานนั้น เมื่อเราเรียก Oracle Schema Manager โดยจะขึ้นหน้าจอลอกอินเข้าสู่ระบบจัดการฐานข้อมูล ดังรูปที่ ค-1




*รูปที่ ค-1 หน้าจอการลอกอินเข้าสู่ฐานข้อมูล เพื่อใช้ Oracle Schema Manager*

เมื่อใส่ทำการลอกอินเข้าสู่ระบบได้แล้ว จะได้หน้าจอดังรูปที่ ค-2




*รูปที่ ค-2 หน้าจอหลักของ Oracle Schema Manager*

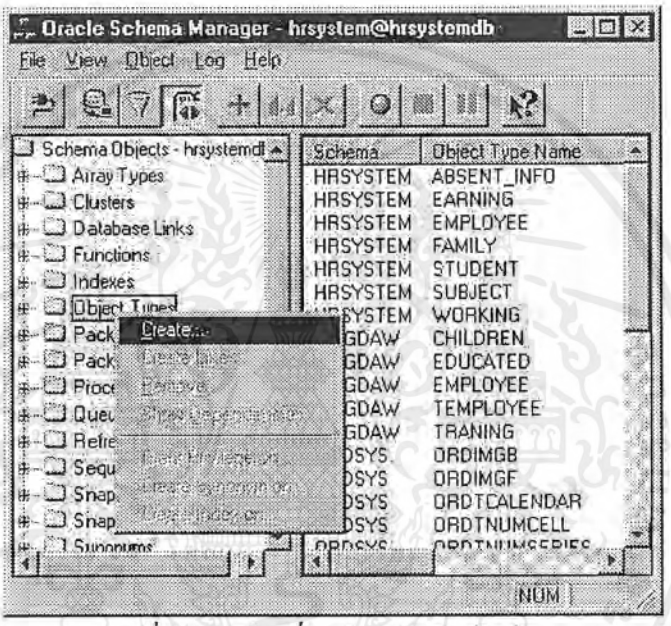
**การสร้างตารางเนสเต็ดเทเบิลโดยใช้ Oracle Schema Manager**

มี 4 ขั้นตอนดังนี้

1. สร้างออบเจ็กต์ไทป์สำหรับตารางที่เป็นเนสเต็ดเทเบิล (หรือตารางย่อย)
2. สร้างเทเบิลไทป์ของออบเจ็กต์ไทป์ของเนสเต็ดเทเบิลที่สร้างขึ้นจากข้อ 1
3. สร้างออบเจ็กต์ไทป์ของตารางที่มีเนสเต็ดเทเบิล หรือตารางใหญ่
4. สร้างตารางของออบเจ็กต์ไทป์ที่สร้างเป็นตารางใหญ่

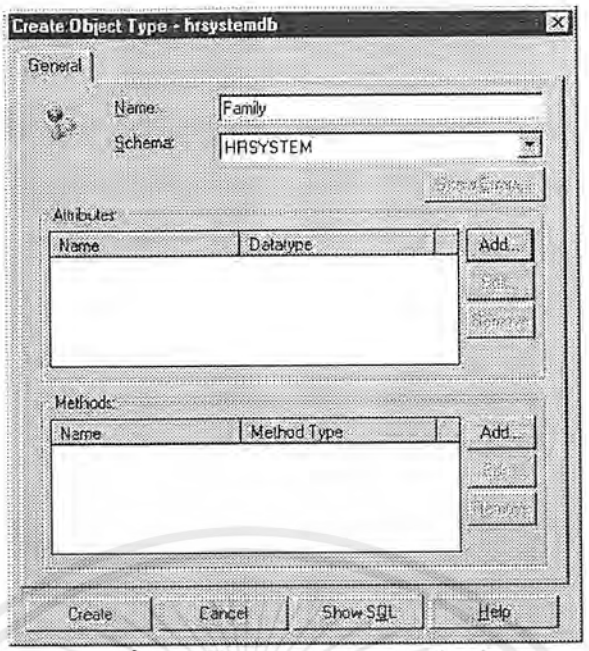
**1. การสร้างออบเจ็กต์ไทป์**

1.1 คลิกขวาที่ Object Types แล้วเลือก Create ดังรูปที่ ค-3



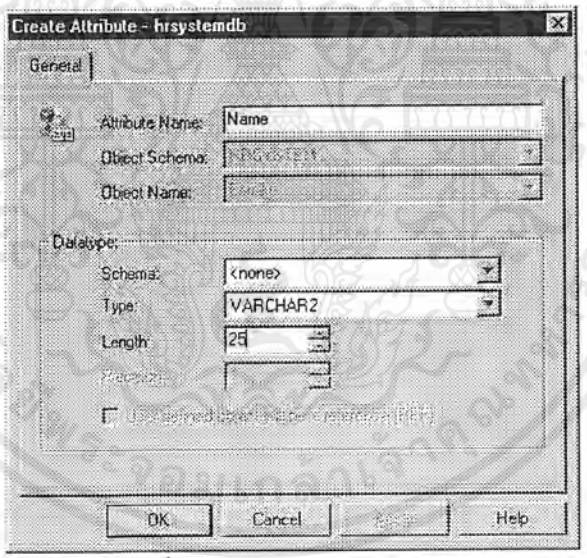

*รูปที่ ค-3 การลือกที่จะสร้างออบเจ็กต์ไทป์*

1.2 กำหนดชื่อออบเจ็กต์ไทป์ ชื่อ Schema และกำหนดแอตทริบิวต์ของออบเจ็กต์ไทป์ ได้โดยคลิกปุ่ม Add



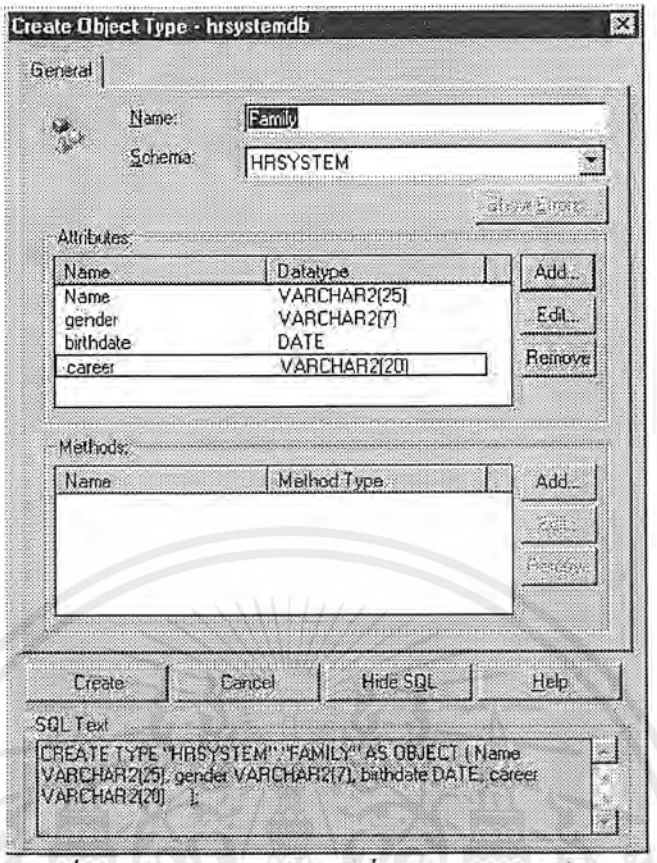

*รูปที่ ค-4 หน้าจอการสร้างออบเจ็กต์ไทป์*

1.3 กำหนดชื่อ ชนิดและขนาดของข้อมูลให้แก่แอตทริบิวต์แต่ละตัว แล้วคลิกปุ่ม OK เมื่อถูกต้อง




*รูปที่ ค-5 การกำหนดชื่อ ชนิดและขนาดของข้อมูลให้แก่แอตทริบิวต์*

1.4 เมื่อกำหนดแอตทริบิวต์ให้แก่ออบเจ็กต์ครบแล้ว สามารถดูคำสั่ง SQL ที่สร้างขึ้นโดยอัตโนมัติได้ โดยคลิกปุ่ม Show SQL จะได้ดังรูปที่ ค-6 แล้วคลิกปุ่ม Create เพื่อสร้างออบเจ็กต์ไทป์

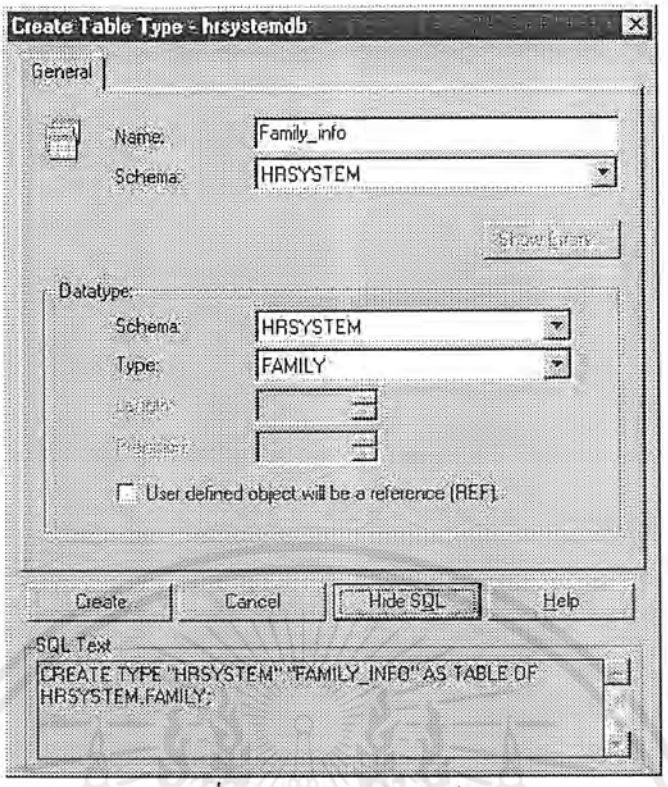



*รูปที่ ค-6 แสดงการออบเจ็กต์ไทป์ที่กำหนดแอตทริบิวต์แล้ว*

**2. การสร้างเทเบิลไทป์**

2.1 เมื่อสร้างออบเจ็กต์ไทป์แล้ว สร้างเทเบิลไทป์โดยคลิกขวาที่ Table Type แล้วเลือก create

2.2 กำหนดชื่อและ Schema ของเทเบิลไทป์ แล้วกำหนดชนิดของข้อมูล โดยเลือก Schema ที่เก็บออบเจ็กต์ไทป์นั้นเอาไว้ และเลือกชนิด (Type) เป็นออบเจ็กต์ที่ต้องการสร้างเทเบิลไทป์ ดังรูปที่ ค-7 แล้วคลิกปุ่ม Create เพื่อสร้างเทเบิลไทป์นี้

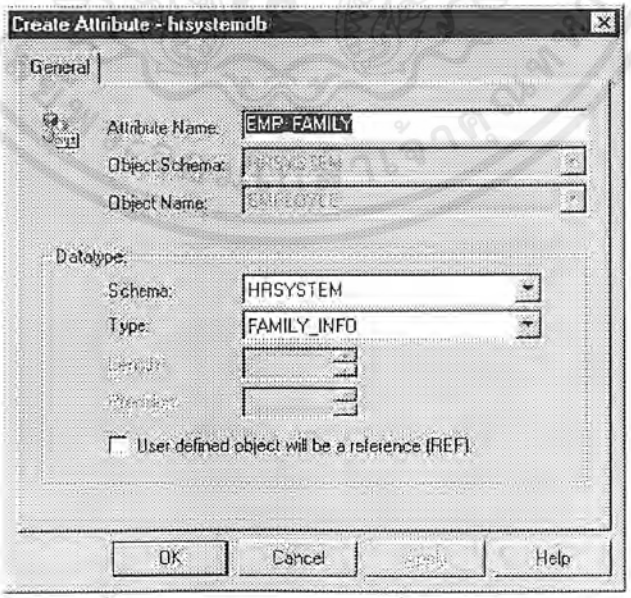



*รูปที่ ค-7 การสร้างเทเบิลไทป์*

**3. การสร้างออบเจ็กต์ไทป์ที่ประกอบด้วยออบเจ็กต์ไทป์อื่น**

3.1 คลิกขวาที่ Object Types แล้วเลือก Create เหมือนข้อ 1.1

3.2 กำหนดแอตทริบิวต์ให้แก่ออบเจ็กต์ไทป์ โดยคลิกปุ่ม Add แอตทริบิวต์อื่นๆ กำหนดเหมือนกับการกำหนดแอตทริบิวต์ให้กับออบเจ็กต์ไทป์ในข้อ 1 ส่วนแอตทริบิวต์ที่ต้องการให้เป็นเนสเต็ดเทเบิลนั้น ให้กำหนด Schema และชนิดข้อมูล ดังรูปที่ ค-8

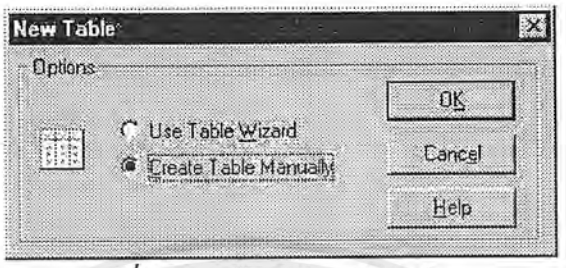



*รูปที่ ค-8 การกำหนดแอตทริบิวต์ที่เป็นออบเจ็กต์ไทป์*

3.3 เมื่อกำหนดแอตทริบิวต์ครบแล้วคลิกปุ่ม Create เพื่อสร้างออบเจ็กต์

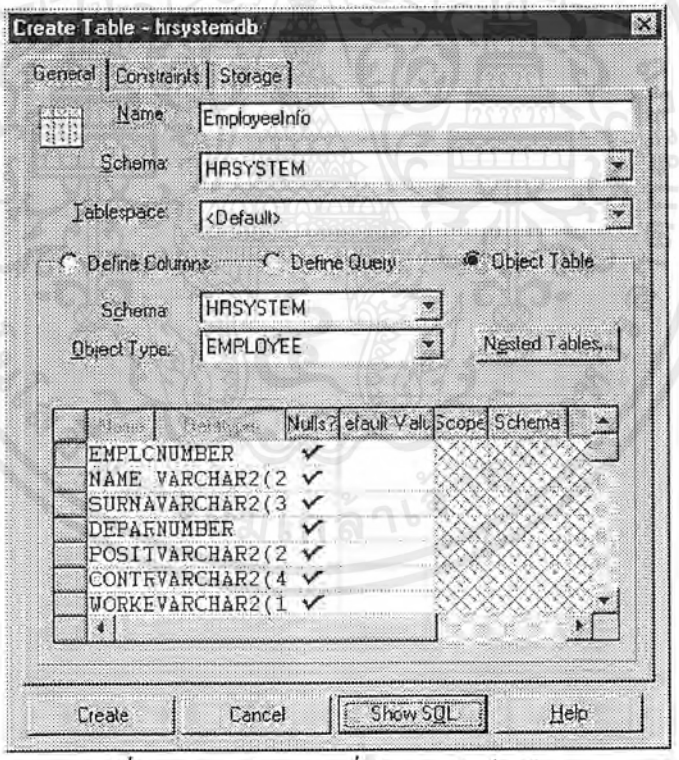
**4. การสร้างตารางสำหรับออบเจ็กต์ไทป์**

4.1 คลิกขวาที่ Tables แล้วเลือก create จะปรากฏไดอะล็อกบล็อกขึ้นมาให้เลือกว่าจะทำการสร้างเทเบิลโดยใช้วิธีใด เลือก Create table manually แล้วคลิกปุ่ม OK



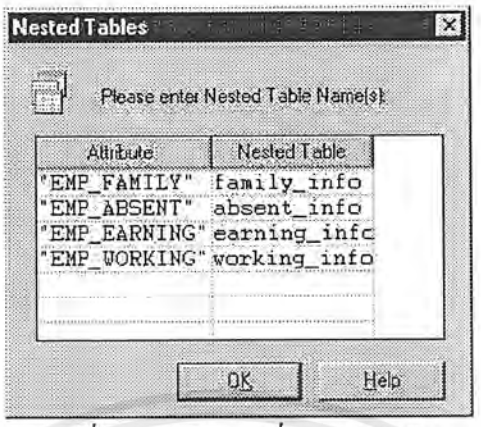

*รูปที่ ค-9 การเลือกสร้างเทเบิลใหม่*

4.2 กำหนดชื่อ , Schema แล้วเลือกตารางแบบ Object Table แล้วกำหนด Schema และชื่อออบเจ็กต์ไทป์ ดังรูปที่ ค-10




*รูปที่ ค-10 การสร้างตารางที่เป็นออบเจ็กต์เทเบิล*

4.3 เมื่อกำหนดชื่อออบเจ็กต์ไทป์ให้แก่ตารางแล้ว จะปรากฏไดอะล็อกบล็อก Nested Tables เพื่อกำนดชื่อให้แก่เนสเต็ดเทเบิล ดังรูปที่ ค-11 แล้วคลิกปุ่ม OK




*รูปที่ ค-11 การกำหนดชื่อเนสเต็ดเทเบิล*

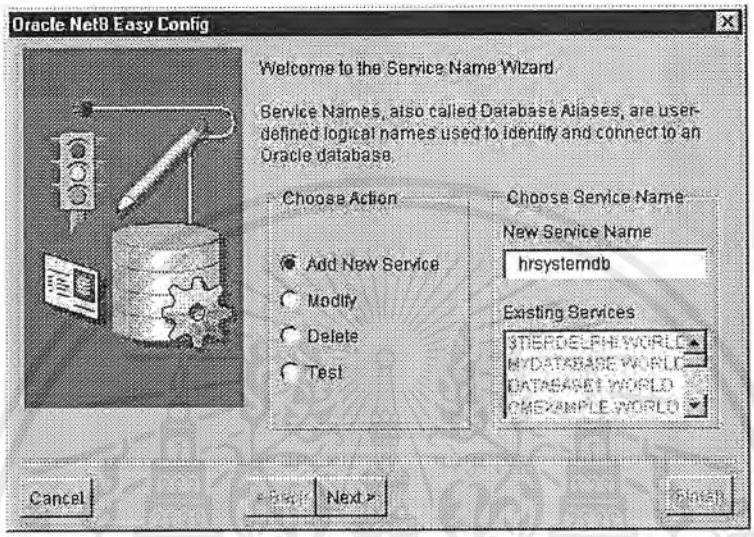
4.4 เมื่อเสร็จแล้วคลิกปุ่ม Create เพื่อสร้างตาราง จะได้ตารางที่มีเนสเต็ดเทเบิล

# ภาคผนวก ง

# การสร้าง Service สำหรับ Oracle Client

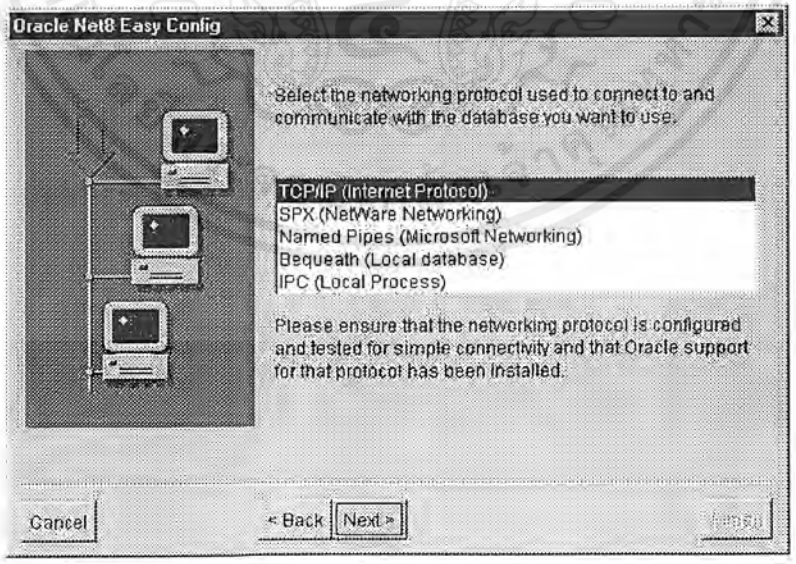
ในการที่จะเชื่อต่อออราเคิลไคลเอนท์เข้ากับออราเคิลเซิร์ฟเวอร์ จะต้องกำหนดเซอร์วิสสำหรับไคลเอนท์ขึ้นมาก่อน โดยจะใช้ Net8 Easy Config ของ Oracle for Windows NT ซึ่งมีขั้นตอนดังต่อไปนี้

1. กำหนดชื่อเซอร์วิส โดยเลือก Add New Service และพิมพ์ชื่อเซอร์วิสในกล่อง New Service Name



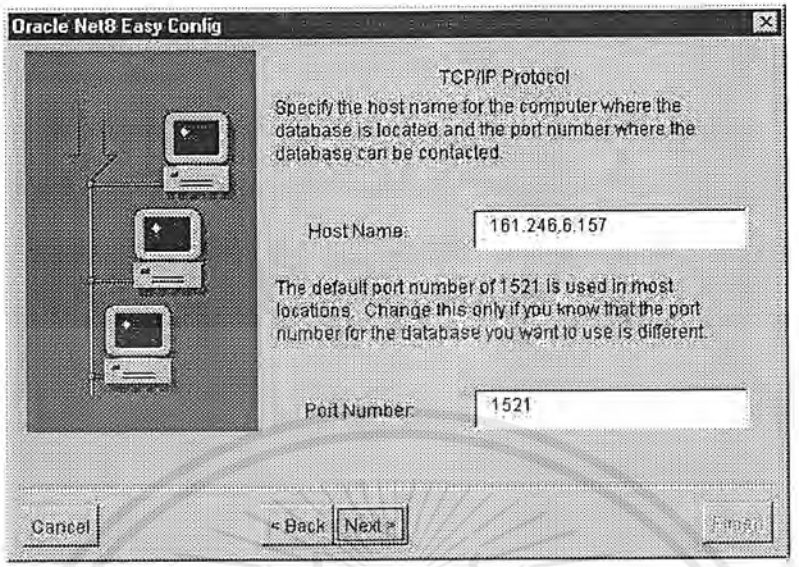

*รูปที่ ง-1 แสดงการกำหนดชื่อเซอร์วิส*

2. กำหนดโปรโตคอลที่ใช้เชื่อมต่อกับดาต้าเบสเซิร์ฟเวอร์ โดยกำหนดเป็น TCP/IP



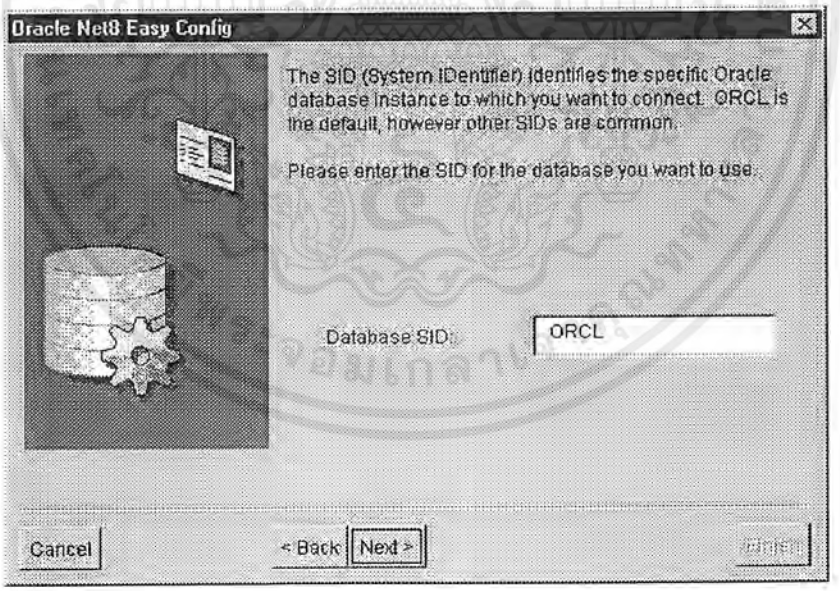

*รูปที่ ง-2 แสดงการกำหนดโปรโตคอลที่ใช้เชื่อมต่อกับดาต้าเบสเซิร์ฟเวอร์*

3. กำหนด IP Address ของดาต้าเบสเซิร์ฟเวอร์ โดยใส่ IP Address ไว้ที่ Host Name และหมายเลขพอร์ตที่ใช้ คือ 1521 ซึ่งถ้าใช้พอร์ตอื่นง่ายกว่า ก็สามารถเปลี่ยนหมายเลขพอร์ตได้



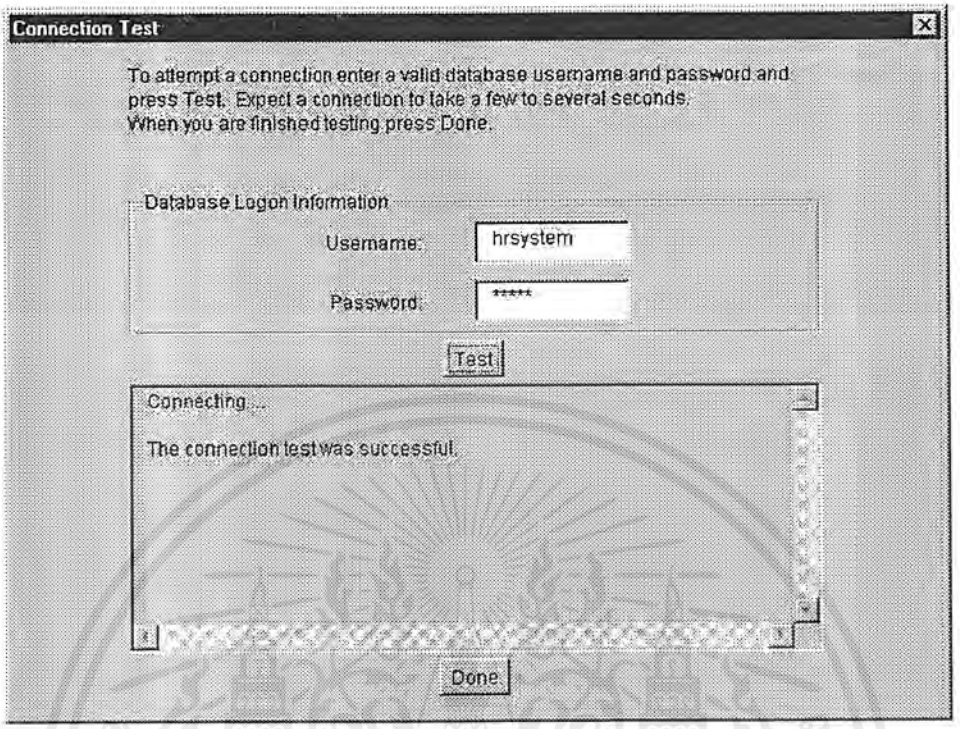

*รูปที่ ง-3 แสดงการกำหนดแอดเดรสของดาต้าเบสเซิร์ฟเวอร์*

4. กำหนด System Identifier เป็น ORCL ซึ่งเป็นดีฟอลต์ที่ระบบได้สร้างไว้ให้แล้ว




*รูปที่ ง-4 แสดงการกำหนด System Identifier*

5. เมื่อกำหนดค่าต่างๆ แล้ว ให้ทดสอบระบบ โดยกดปุ่ม Test Service แล้วทดสอบโดยใส่ชื่อ user name และรหัสผ่านลงไป




*รูปที่ ง-5 แสดงการทดสอบเซอร์วิสที่ตั้งขึ้น*

เมื่อสามารถเชื่อมต่อกับดาต้าเบสเซิร์ฟเวอร์ได้แล้ว ให้กดปุ่ม Done และกดปุ่ม Finish เพื่อเซฟเซอร์วิสที่ตั้งขึ้น หลังจากนี้ เมื่อมีการเรียกใช้ดาต้าเบสเซิร์ฟเวอร์ผ่านโปรแกรมต่างๆ ก็ต้องกำหนดเซอร์วิสลงไปด้วย เพื่อเรียกไปยังดาต้าเบสเซิร์ฟเวอร์ที่ต้องการ

# บรรณานุกรม

**หนังสืออ้างอิง**


[1] Bruce Powel Douglass, " Real-Time UML:Developing Efficient Object for Embeded Systems",Addison-Westley,1998

[2] Hans-Erik Eriksson, Magnus Penker, "UML Toolkit",Wiley Computer Publishing 1998

[3] Marco Cantu,"Mastering Delphi 4",Sybex,1997

[4] C.J. Date," An Introduction to Database System 6 th Edition, Addison Wesley Publishing company

[5] CRAIG LARMAN, "APPLYING UML AND PATTERNS : An introduction to object-oriented analysis and design" ,Prentice Hall PTR,1998

[6] รองศาสตราจารย์ สมคิด บางโม," องค์การและการจัดการ " ,บริษัทวิทยพัฒน์ จำกัด

[7] กนก กุสุมาลย์นุกูล,ไกรวุฒิ มั่นเสถียรสิน," คู่มือการเขียนโปรแกรม Delphi 4 ", สำนักพิมพ์ซัคเซสมีเดีย

[8] ธวัชชัย สุทธิทศธรรม," ระบบอัตโนมัติช่วยในการออกแบบฐานข้อมูลแบบรีเลชั่นแนล" ,วิทยานิพนธ์ บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[9] จันทรกานต์ จุหอม, ทิพย์เนตร แก้วปัตตะ, " การพัฒนาโปรแกรมประยุกต์ในเชิงวัตถุ ", ปริญญานิพนธ์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[10] ชญานนท์ โตวิกกัย,ชาญชัย จอประเสริฐกุล,"การพัฒนาแอพพลิเคชั่นเชิงวัตถุโดยใช้สถาปัตยกรรม 3-เทียร์" ปริญญานิพนธ์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง


**เวปไซต์อ้างอิง**


[11] http://www.borland.com/techpubs/delphi

[12] http://www.borland.com/midas/papers

[13] http://www.microsoft.com/com/tech/dcom.asp

[14] http://www.oracle.com