ฐานข้อมูลเชิงวัตถุสัมพันธ์ ประยุกต์ใช้กับงานแคด/แคม

OBJECT-RELATIONAL DATABASE APPLICATION IN CAD/CAM

โดย

นางสาวทอรุ้ง กลิ่นศรีสุข 40013251

นายปรีชา ชิณราด 40013257

อาจารย์ที่ปรึกษา

รศ.ดร. ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ประยุกต์ใช้กับงานแคด/แคม

OBJECT-RELATIONAL DATABASE APPLICATION IN CAD/CAM

ผู้จัดทำ

1. นางสาวทอรุ้ง กลิ่นศรีสุข รหัสประจำตัว 40013251
2. นายปรีชา ชิณราด รหัสประจำตัว 40013257

อาจารย์ที่ปรึกษา

(รศ.ดร. ศุภมิตร จิตตะยโศธร)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ประยุกต์ใช้กับงานแคด/แคม

นางสาวทอรุ้ง กลิ่นศรีสุข 40013251
นายปรีชา ชิณราด 40013257
รศ.ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

เนื่องจากข้อมูลในปัจจุบันมีความหลากหลายและซับซ้อนมากขึ้นไม่ว่าจะเป็นข้อมูลทางด้านมัลติมีเดีย ข้อมูลทางด้านธุรกิจ และเอกสารที่ซับซ้อนอื่น ๆ จำเป็นอย่างยิ่งที่จะต้องมีระบบจัดการฐานข้อมูลที่มีรองรับข้อมูลเหล่านี้ มีประสิทธิภาพสูง และไว้วางใจได้ นอกจากนี้ยังจำเป็นต้องมีแนวความคิดใหม่ ๆ ที่จะมาสนับสนุนการจัดการข้อมูลเหล่านี้

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database) เป็นแนวความคิดที่จะผสมผสานความสามารถ ความเร็ว และความเชื่อถือได้ของระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ที่ใช้งานมายาวนาน เข้ากับฐานข้อมูลเชิงวัตถุ (Object-Oriented) ที่สามารถจัดเก็บและบริหารข้อมูลที่ซับซ้อนได้ ซึ่งจะนำอินฟอร์มิกส์ยูนิเวอร์แซลเซิร์ฟเวอร์ ซึ่งสนับสนุนแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์มาใช้ในการจัดสร้างและพัฒนา

โครงงานนี้ได้นำเสนอระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ด้วยการนำไปใช้ในการพัฒนางานด้านแคด/แคม (Computer Aided Design / Computer-Aided Manufacturing หรือ CAD/CAM) เพื่อจัดเก็บและบริหารวงจรดิจิตอล (Digital circuit) โดยการจัดเก็บฐานข้อมูลขององค์ประกอบต่าง ๆ ของวงจรดิจิตอลเป็นออปเจกต์ต่าง ๆ ที่มีความสัมพันธ์กัน และมีการทำงานร่วมกัน ผู้ใช้สามารถใช้ระบบได้เต็มรูปแบบ คือ การสร้าง เปิด จัดเก็บ แก้ไข และจำลองการทำงานของวงจร

Object-Relational Database Application in CAD/CAM

Toroong Klinsrisuk 40013251

Preecha Chinarad 40013257

Assoc. Prof. Dr. Suphamit Chittayasothorn Advisor

2000

ABSTRACT

Now a day, Data was more complex such as Multimedia data, Business data and other complex documents. That need Database Management System can supports these data which has high efficiency and reliability. Not only that , new database model and concept are necessary to support these data.

Object-Relation database is new database model that combines the speed and reliability of relational database and new features of object-oriented database together. Object-Relational can store and manage complex data. Object-Relational database are combination of Relational database and Object Oriented database. We were used Informix Universal Server to develop them because it supports some property of Object-Relational database concept.

The thesis is represents Object-Relational database to develops CAD/CAM (Computer Aided Design / Computer Aided Manufacturing) application for store and manage digital circuit. Main objective is storing several components and their relation of digital circuit in object form and working together. User can create , open , save, modify and simulate digital circuit.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์ด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษา รศ.ดร.ศุภมิตร จิตตะยโศธร ซึ่งคอยให้คำปรึกษาที่ดี ทั้งในทางทฤษฎีและทางปฏิบัติ ท่านคอยแนะทางที่ดีเสมอ นอกจากความรู้ที่ท่านมีมากอยู่แล้ว ท่านยังมีความรับผิดชอบและเป็นอาจารย์ที่ปรึกษาที่ดีตลอดเวลา

นอกจากนี้ต้องขอขอบคุณพี่บัณฑิตที่คอยช่วยเหลือในเรื่องอุปกรณ์ในการจัดทำโปรเจ็ก และคอยช่วยเหลือในเรื่องคำปรึกษาเกี่ยวกับระบบฐานข้อมูลที่เป็นประโยชน์ในการจัดทำโปรเจ็กให้สำเร็จลุล่วงด้วยดีเสมอมา และสุดท้ายขอขอบคุณพี่พิทักษ์ที่ให้ความอนุเคราะห์เรื่องหนังสือคู่มือ ซึ่งเป็นประโยชน์มาก

ทอรุ้ง กลิ่นศรีสุข

ปรีชา ชิณณาด

สารบัญ

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ข้อจำกัดของฐานข้อมูลที่เป็นแบบโครงสร้างข้อมูลเชิงสัมพันธ์ (Relational Data Model) ทำให้ไม่เหมาะสมกับบางแอพพลิเคชั่นที่มีความซับซ้อน และไม่สามารถกำหนดชนิดข้อมูลบางประเภทเพื่อเก็บข้อมูลได้อย่างมีประสิทธิภาพได้แต่ในปัจจุบันฐานข้อมูลมีบทบาทในการใช้งานแต่ละด้านมากขึ้น ซึ่งเห็นได้ชัดในการนำไปใช้กับองค์กรธุรกิจและภาครัฐต่างๆ เป็นต้น จากการนำไปงานในด้านต่างๆ นี้เองทำให้พบปัญหาในการจัดเก็บข้อมูลมากขึ้น จนมีการหาแนวทางอื่นๆ ไม่ว่าจะเป็นโมเดลที่ใช้ในการออกแบบหรือโครงสร้างข้อมูลที่สามารถรองรับความซับซ้อนได้

โครงสร้างข้อมูลเชิงวัตถุ (Object-Oriented Data Model) เป็นแนวทางหนึ่งที่น่าสนใจ การออกแบบทำได้ใกล้เคียงกับโลกแห่งความเป็นจริง และหากมีการออกแบบที่ดีจะทำให้มีความยืดหยุ่นสูง แต่ระบบจัดการฐานข้อมูล (Database Management System) ในปัจจุบันยังไม่สามารถที่สนับสนุนแนวความคิดของโครงสร้างข้อมูลเชิงวัตถุได้อย่างเต็มที่เนื่องจากในตลาดมีการใช้ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) อยู่มาก จึงต้องทำให้สามารถรองรับแบบเก่าและทำการเพิ่มการทำงานใหม่ๆ เข้าไปให้มีคุณสมบัติของโครงสร้างข้อมูลเชิงวัตถุได้มากขึ้นจึงกลายเป็นโครงสร้างข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Data Model) ซึ่งทำให้แก้ปัญหาของโครงสร้างข้อมูลเชิงสัมพันธ์ได้บางประการ

1.2 วัตถุประสงค์งานวิจัย

1. ศึกษาหลักการของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์
2. ศึกษาและใช้งานอินฟอร์มิกซ์ ยูนิเวอร์แซลเซิร์ฟเวอร์ (Informix Universal Server)
3. ศึกษาและใช้งานดาต้าเบลดโมดูล (DataBlade Module) ในการเพิ่มความสามารถให้กับดาต้าเบสเซิร์ฟเวอร์
4. ศึกษาและประยุกต์ใช้งานชนิดข้อมูลแบบโอเปค (Opaque Data type)
5. ออกแบบและจัดทำฐานข้อมูลวงจรดิจิตอล โดยใช้แนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์
6. จำลองการทำงานของวงจรดิจิตอลที่ถูกจัดเก็บไว้ในฐานข้อมูล
7. สร้างแอปพลิเคชั่นเพื่อใช้งานฐานข้อมูลวงจรดิจิตอล

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้เป็นการออกแบบระบบฐานข้อมูลเพื่อทำการเก็บวงจรดิจิตอล ซึ่งอุปกรณ์พื้นฐาน เช่น แอนด์เกท(And Gate), ออร์เกท (Or Gate) ถูกจัดเก็บไว้บนฐานข้อมูลเรียกว่าชนิดของอุปกรณ์ ผู้ใช้สามารถสร้างวงจรดิจิตอลโดยการนำอุปกรณ์วาง และมีการเชื่อมต่อกันระหว่างขาของอุปกรณ์ และผู้ใช้

งานสามารถป้อนค่าลอจิกและทำการจำลองการทำงาน และสังเกตค่าทางลอจิก ณ จุดต่าง ๆ ของวงจรได้จากจอภาพ

สำหรับในขั้นตอนการออกแบบได้ออกแบบตามแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยการใช้ อีอาร์ไดอะแกรมในการนำเสนอ สำหรับเครื่องมือที่นำมาใช้ในการสร้างฐานข้อมูลคืออินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์และใช้ดาต้าเบลดโมดูลเดเวลลอปเปอร์คิท (DataBlade Module Developer Kit) ในการเพิ่ม ดาต้าเบลด(DataBlade) เข้าไปในฐานข้อมูล ซึ่งภายในดาต้าเบลดจะประกอบด้วยชนิดข้อมูลแบบโอเปคและรูทีน (Routine) เป็นสำคัญ บนฐานข้อมูลจะประกอบด้วยตารางต่าง ๆ และมีการควบคุมความถูกต้องของตารางโดยการสร้างทริกเกอร์ (Trigger) บนระหว่างตารางมีการสร้างโปรซีเดอร์ (Procedure) ที่เขียนด้วยภาษา SPL (Stored Procedure Language) ฝังไว้บนฐานข้อมูลเพื่อให้แอปพลิเคชั่นสามารถเรียกใช้ได้ง่าย แต่บางรูทีนที่กระทำกับข้อมูลชนิดโอเปคจะถูกเขียนด้วยภาษาซี ซึ่งจะถูกสร้างเป็นไลบรารีไฟล์ และถูกลงทะเบียน (Register) ลงบนฐานข้อมูลโดยใช้โปรแกรมเบลดเมเนเจอร์ (Blade Manager)

ด้านแอปพลิเคชั่นจะถูกเขียนด้วยวิชวลเบสิก (Visual Basic) และทำการเชื่อมต่อกับอินฟอร์มิกซ์เซิร์ฟเวอร์ด้วยดาต้าไดเร็คเตอร์ (Data Director) ซึ่งแอปพลิเคชั่นจะสามารถสร้างวงจรดิจิตอล บันทึกลงบนฐานข้อมูล ดึงวงจรบนฐานข้อมูลแสดงบนจอภาพ จำลองการทำงานของวงจร และแก้ไขวงจรดิจิตอลได้

1.4 ขั้นตอนการดำเนินงาน

ขั้นตอนในการจัดทำงานวิจัยประกอบด้วยส่วนต่างๆ ที่ทำงานแยกจากกันโดยเป็นส่วนที่ต้องทำบนเซิร์ฟเวอร์ และส่วนที่ต้องทำบนไคลเอนท์ (Client) แบ่งออกเป็น 5 ส่วนดังต่อไปนี้

ส่วนที่ 1 ทำการออกแบบฐานข้อมูลวงจรดิจิตอล โดยเริ่มจากทำการค้นคว้าจากงานวิจัยเก่า ที่มีผู้ทำมาแล้วและทดลองใช้ระบบฐานข้อมูลแบบสัมพันธ์ในการออกแบบ จนได้เห็นความยากลำบากในการจัดทำและเหตุผลว่าทำไมจึงจำเป็นต้องใช้ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์มาใช้ในการออกแบบ หลังจากนั้นก็เริ่มทำการออกแบบระบบฐานข้อมูลแบบวัตถุสัมพันธ์โดยนำเสนอโดยใช้ E-R ไดอะแกรม หลังจากนั้นทำการยุบกลุ่มของแอตทริบิวต์ใส่เป็นชนิดข้อมูลแบบโอเปค จะได้ชนิดข้อมูลโอเปคซึ่งประกอบด้วย Component, Pin และ Wire หลังจากนั้นทำการสร้างรูทีนเพื่อสนับสนุนชนิดข้อมูลแบบโอเปค

ส่วนที่ 2 ทำการสร้างดาต้าเบลดโมดูล โดยใช้เครื่องมือของอินฟอร์มิกซ์ที่มีให้ ภายในประกอบด้วยชนิดข้อมูลแบบโอเปคและรูทีนที่จำเป็น หลังจากนั้นทำการสร้างโค้ด (Code) ให้อัตโนมัติ และทำการแก้ไขโค้ดให้ตรงตามต้องการ และคอมไพล์เป็นไลบรารีไฟล์ และทำการลงทะเบียนลงในฐานข้อมูล

ส่วนที่ 3 จัดสร้างส่วนต่าง ๆ บนฐานข้อมูล ได้แก่ สร้างตาราง ความสัมพันธ์ระหว่างตาราง ทริกเกอร์ เขียนโปรซีเดอร์ที่จำเป็น และทำการป้อนข้อมูลลงบนฐานข้อมูล ซึ่งเป็นข้อมูลชนิดอุปกรณ์ ซึ่งผู้ใช้ไม่สามารถสร้างเองได้

ส่วนที่ 4 ทำการเขียนแอปพลิเคชั่นโดยใช้วิชวลเบสิก เวอร์ชัน 6 ซึ่งผู้ใช้สามารถสร้างวงจรดิจิตอลเอง บันทึกลงบนฐานข้อมูล และนำวงจรดิจิตอลที่จัดเก็บอยู่บนฐานข้อมูลมาแสดงบนหน้าจอได้

ส่วนที่ 5 ทำการสร้างโปรซีเดอร์ในการจำลองการทำงานของวงจรดิจิตอล ซึ่งเขียนด้วยวิชวลเบสิกเช่นเดียวกัน โดยสร้างเป็นโปรซีเดอร์หนึ่งในแอปพลิเคชั่น

ซึ่งในการติดต่อกันระหว่างไคลเอนท์กับเซิร์ฟเวอร์จะติดต่อกันโดยใช้คำสั่ง SQL เป็นสำคัญ โดยโปรซีเดอร์ต่าง ๆ ถูกเก็บไว้บนเซิร์ฟเวอร์ ยกเว้นโปรซีเดอร์ในการจำลองการทำงานซึ่งใช้หน่วยความจำค่อนข้างมากจะถูกนำมาทำที่ฝั่งไคลเอนท์

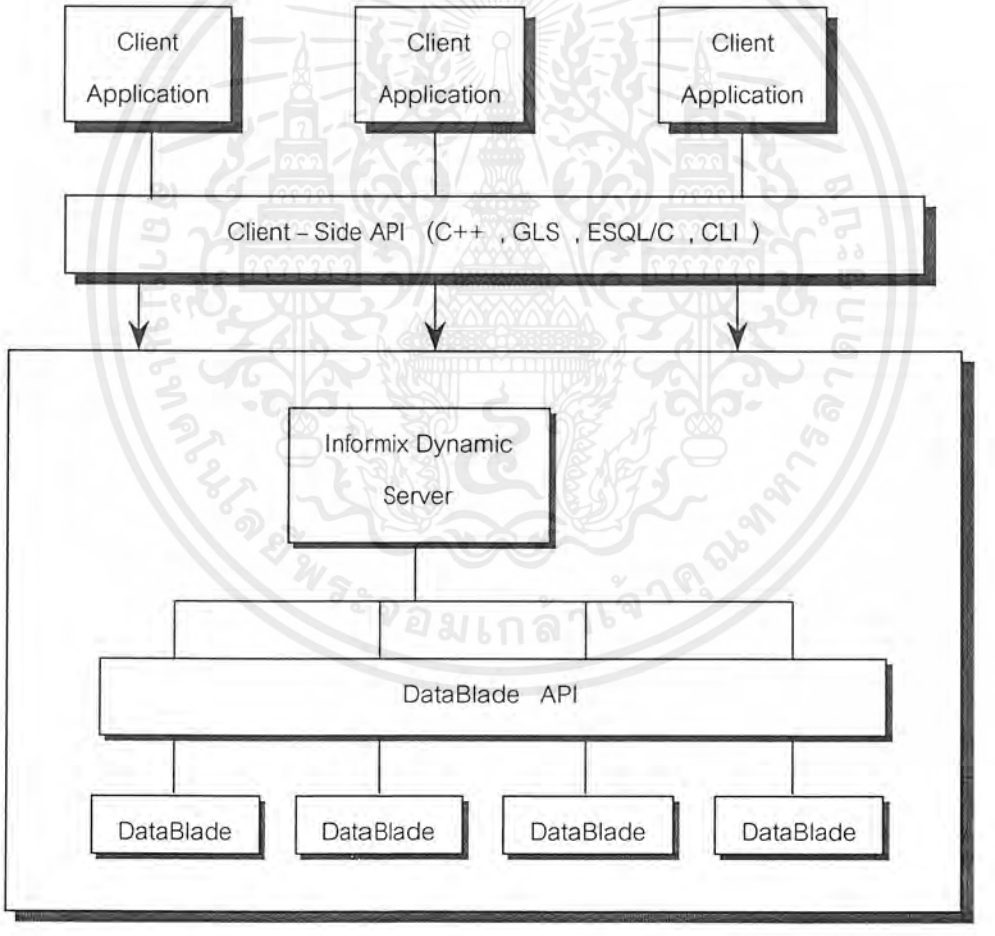
บทที่ 2
อินฟอร์มิกซ์ ยูนิเวอร์แซล เซิร์ฟเวอร์

2.1 สถาปัตยกรรมของอินฟอร์มิกซ์ ยูนิเวอร์แซล เซิร์ฟเวอร์

องค์ประกอบที่สำคัญของอินฟอร์มิกซ์ ยูนิเวอร์แซล เซิร์ฟเวอร์มีดังนี้

- ดาต้าเบลดโมดูล (DataBlade Module)
- ดาต้าเบลดเอพีไอ (DataBlade API)
- ไคลเอนท์ไซด์เอพีไอ (Client-Side API)
- ไคลเอนท์ไซด์แอปพลิเคชั่น (Client-Side Application)

การติดต่อระหว่างอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์กับไคลเอนท์ แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 สถาปัตยกรรมของอินฟอมิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์

2.1.1 ดาต้าเบลดโมดูล

ดาต้าเบลดโมดูลเป็นเครื่องมือที่จะทำให้นักพัฒนา สามารถเพิ่มความสามารถให้กับอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ในการจัดเตรียมชนิดข้อมูลและรูทีนใหม่ ๆ เพื่อรองรับกับความต้องการของแต่ละแอปพลิเคชั่น ซึ่งดาต้าเบลดจะเป็นแพ็กเกจ (Package) ที่รวบรวมส่วนประกอบต่าง ๆ เข้าไว้ด้วย เมื่อทำการเพิ่มดาต้าเบลดเข้าไปในดาต้าเบสเซิร์ฟเวอร์ ส่วนประกอบต่าง ๆ เหล่านี้จะรวมเข้าไปอยู่ในระดับเดียวกันกับส่วนประกอบที่มีอยู่เดิมของดาต้าเบสเซิร์ฟเวอร์

2.1.2 ดาต้าเบลดเอพีไอ

จะเป็นองค์ประกอบที่ทำให้นักพัฒนา สามารถสร้างแอพพลิเคชั่นบนฝั่งไคลเอนท์ และดาต้าเบสเซิร์ฟเวอร์ เพื่อทำการเข้าถึงข้อมูลที่เก็บในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์จะจัดเตรียมความสามารถต่าง ๆ ดังต่อไปนี้

- จัดการกับการเชื่อมต่อเข้ากับฐานข้อมูล
- ส่งคำสั่ง SQL ไปยังดาต้าเบสเซิร์ฟเวอร์
- ประมวลผลผลลัพธ์ของการคิวรี (Query)
- จัดการกับเซิร์ฟเวอร์ อีเวนต์ (Server Events) และข้อผิดพลาดต่าง ๆ
- สร้างฟังก์ชันของดาต้าเบสเซิร์ฟเวอร์
- จัดการกับหน่วยความจำของดาต้าเบสเซิร์ฟเวอร์

2.1.3 ไคลเอนท์ไซด์เอพีไอ

อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์จะมีชุดของไคลเอนท์ไซด์เอพีไอที่อนุญาตให้ผู้เขียนโปรแกรมฝั่งไคลเอนต์สามารถส่ง SQL สเตจเมนต์ เข้าไปในโปรแกรมได้โดยตรง ซึ่งไคลเอนท์ไซด์เอพีไอที่มีให้ใช้คือ

- C++ Interface
- Java API
- GLS API
- Informix - ESQL/C
- Informix – CLI

2.1.4 ไคลเอนท์ไซด์แอปพลิเคชั่น

แอพพลิเคชั่นฝั่งไคลเอนท์อนุญาตให้ผู้ใช้สามารถเข้าถึงดาต้าเบลดโมดูลฟังก์ชัน (DataBlade Module Function) และที่เก็บไว้ในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ได้

2.2 ชนิดข้อมูลในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์

อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์จะสนับสนุนชนิดข้อมูลดังต่อไปนี้

2.2.1 ชนิดข้อมูลแบบบิวท์อิน (Built-in data types) เช่น alphanumeric, integer, date, time, char, string ฯลฯ

2.2.2 ชนิดข้อมูลแบบผู้ใช้กำหนดเอง (User – defined data type : UDT)

2.2.3 ชนิดข้อมูลแบบซับซ้อน (Complex data type)

2.2.1 ชนิดข้อมูลแบบบิวท์อิน

จะเป็นชนิดข้อมูลที่สนับสนุนในรีเลชันนอลดาต้าเบสเซิร์ฟเวอร์ (Relational Database Server) ที่รวมอยู่ในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ ตัวอย่างของ Built-in data types คือ

- *Character data types* เช่น CHAR, VARCHAR และ LVARCHAR
- *Numeric data type* เช่น Decimal, Money, Smallint, integer และ Float
- *Large – Object data type* เช่น TEXT และ BYTE ซึ่งเป็น simple – larger – object data types และ CLOB, BLOB ซึ่งเป็น Smart – Large – Object Type
- *Time data type* เช่น Date, Datetime และ Interval
- *Miscellaneous data type* เช่น Boolean

2.2.2 ชนิดข้อมูลแบบผู้ใช้กำหนดเอง

จะอนุญาตให้ผู้ใช้สามารถนิยามโครงสร้างข้อมูลขึ้นมาเองได้ ซึ่งจะสามารถกระทำกับ ชนิดข้อมูลมาตรฐาน เช่น สามารถอินเด็กซ์ (Index) และปรับปรุงให้มีการทำงานที่ดีขึ้น (Optimized) ได้ สามารถคำนวณระหว่างแต่ละตัวได้ซึ่ง UDT สามารถแบ่งได้เป็น 2 ชนิด ได้แก่โอเปค และดิสทิงค์ (Distinct)

- *ชนิดข้อมูลแบบโอเปค* จะอนุญาตให้ข้อมูลต่างๆ สามารถถูกนำเสนอในรูปแบบของภาษาซี ในการที่จะถูกประมวลผลและจัดเก็บในเซิร์ฟเวอร์ โดยทำการซ่อนแอตทริบิวต์ (Encapsulate Attribute) ต่างๆ ไว้ภายในตัวมัน ในระดับไพรเวท (Private) ซึ่งชนิดข้อมูลแบบโอเปคจะจัดเตรียมการควบคุมที่สมบูรณ์สำหรับนักพัฒนา บนข้อมูลที่ถูกจัดเก็บและถูกประมวลผล และให้ประสิทธิภาพสูงสุดในการเข้าถึงในระดับต่างๆ และยังคงสามารถจัดการ ได้เช่นเดียวกันกับชนิดข้อมูลแบบบิวท์อิน
- *ดิสทิงค์* จัดเตรียมวิธีที่รวดเร็วในการสร้างชนิดข้อมูลใหม่ ให้มีโครงสร้างเดียวกับชนิดข้อมูลที่มีอยู่แล้ว ซึ่งอาจจะเป็นชนิดข้อมูลแบบผู้ใช้กำหนดเอง หรือข้อมูลชนิดอื่นๆ เช่น ผลกำไรสามารถเป็นชนิดข้อมูลแบบดิสทิงค์ซึ่งถูกสร้างขึ้นจากการลบรายจ่ายออกจากรายได้ เป็นต้น

2.2.3 ชนิดข้อมูลแบบซับซ้อน

เป็นข้อมูลที่ถูกสร้างขึ้นจากชนิดข้อมูลที่มีอยู่ ตัวอย่างเช่น อาจจะประกอบขึ้นจากชนิดข้อมูลแบบบิวท์อิน ชนิดข้อมูลแบบโอเปค ชนิดข้อมูลแบบดิสทิงค์ หรือชนิดข้อมูลแบบซับซ้อนอื่นๆ ซึ่งข้อมูลแบบซับซ้อนนี้ จะสามารถเข้าถึงโดย SQL สเตจเมนต์ สำหรับชนิดข้อมูลแบบซับซ้อนจะประกอบด้วย 2 ชนิดคือ

- *ชนิดของข้อมูลแบบแถว (Row data type)* ซึ่งเป็นแถว (Row) ของคอลัมน์ (Column) โดยอาจแบ่งเป็นหลายๆ คอลัมน์เรียกว่าฟิลด์ (Field) ซึ่งแต่ละฟิลด์อาจมีหลายๆชนิดข้อมูล ซึ่งสามารถทำการเข้าถึงแต่ละฟิลด์ได้โดยตรง ตัวอย่างเช่น ข้อมูลเกี่ยวกับที่อยู่ของลูกค้า อาจแบ่งเป็นหลายๆ คอลัมน์ เช่น ถนน, เมือง, รัฐ และ รหัสไปรษณีย์ เป็นต้น ตัวอย่างของข้อมูลชนิดนี้แสดงได้ดังรูปที่ 2.2

Employee Table

NAME	ADDRESS	Contact Info
David J Moove	Address_t	SET(LVARCHAR)

Row Type address_t

Street	City	State	Zip_code
252 Grand Ave.	Oakland	CA	94610

รูปที่ 2.2 ตัวอย่างของการสร้างชนิดข้อมูลแบบแถว

- *ชนิดข้อมูลแบบคอลเลคชัน (Collection type)* คือ ชนิดข้อมูลที่มีลักษณะคล้ายกับเป็น เนสท์เทดเทเบิล (nested table) ระหว่างคอลัมน์จะเป็นเซตของฟิลด์ที่มีชนิดข้อมูลที่เหมือนกัน จะแก้ปัญหาของฐานข้อมูลเชิงสัมพันธ์ที่กำหนดข้อมูลแบบหลายค่า (multiple value) เป็น 1 คอลัมน์เป็นเรคคอร์ด (Record) เดียว จะทำให้มีประสิทธิภาพสูงในการออกแบบฐานข้อมูล และทำให้การเขียนโปรแกรมทำได้ง่ายขึ้น ชนิดข้อมูลแบบคอลเลคชันอาจแบ่งเป็น เซต (SET), ลิสต์ (LIST) หรือ มัลติเซต (MULTISET) ซึ่งสามารถเลือกใช้งานได้ตามลักษณะการใช้งาน ซึ่งเซตจะเป็นชุดของข้อมูลประเภทเดียวกันที่ไม่มีลำดับ และสมาชิกภายในไม่ซ้ำ ส่วน ลิสต์จะเป็นชนิดข้อมูลที่เรียงลำดับ และสมาชิกภายในซ้ำได้ ส่วนมัลติเซตคือชุดของข้อมูลที่ไม่เรียงลำดับ แต่สมาชิกภายในสามารถซ้ำได้ ตัวอย่างของชนิดข้อมูลแบบคอลเลคชันแสดงได้ดังรูปที่ 2.3

Employee table

Name	Street	City	Zip_code	Contact_Info
David J. Moore	252 Grand Ave	Oakland	94610	SET(LVARCHAR)

Home : 510-555-1234
Work : 515-555-0000
Fax : 510-555-9999
e-mail : abc@colltype.com

รูปที่ 2.3 ตัวอย่างของชนิดข้อมูลแบบคอลเลคชัน

บทที่ 3
ดาต้าเบลดโมดูล

ดาต้าเบลดโมดูลคือ ชุดของซอฟต์แวร์ซึ่งเป็นดาต้าเบสออบเจ็ค (Database Object) หรือโค้ดต่างๆ ที่สามารถเพิ่มเติมความสามารถให้กับอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ไม่ว่าจะเป็นชนิดของข้อมูลใหม่ๆ หรือรูทีนที่เซิร์ฟเวอร์ไม่มีรองรับ เมื่อทำการเพิ่มเติมสิ่งเหล่านี้เข้าไปเซิร์ฟเวอร์แล้ว ดาต้าเบสเซิร์ฟเวอร์จะมองเห็นในระดับเดียวกันกับชนิดของข้อมูลแบบบิวท์อิน หรือ บิวท์อินฟังก์ชัน (Built-in Function)

สิ่งที่ควรเพิ่มเข้าไปในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ผ่านทางดาต้าเบลดโมดูลคือ

- ชนิดข้อมูลที่มีความซับซ้อนสูง ซึ่งโอเปอเรชัน (Operation) พื้นฐานไม่สามารถจัดการได้
- รูทีนแบบกำหนดเอง (User-Defined Routine) ที่มีลักษณะเฉพาะ และมีการทำงานที่พิเศษ
- กฎ (Rules) ที่เรากำหนดขึ้นเอง เพื่อความคุมการเปลี่ยนแปลงของของฐานข้อมูลที่สร้างขึ้น

3.1 เหตุผลที่จำเป็นต้องมีดาต้าเบลดโมดูล

- ชนิดข้อมูลในปัจจุบันที่มีความซับซ้อนในการใช้งานหรือการจัดเก็บมากยิ่งขึ้น ยากที่จะใช้ชนิดข้อมูลพื้นฐานในการจัดการ
- เพิ่มความสามารถให้กับดาต้าเบสเซิร์ฟเวอร์ที่สามารถเรียกใช้รูทีนที่มีความสามารถมากขึ้นและทำงานเฉพาะอย่างมากขึ้น
- ลดความหนาแน่นของข้อมูลในสายส่ง (Traffic) ของเครือข่าย โดยสามารถลดปริมาณข้อมูลที่ส่งระหว่างเซิร์ฟเวอร์กับไคลเอนท์ได้ โดยการรวมชนิดข้อมูล รูทีน และออบเจกต์อื่นๆ ให้ประมวลผลบนเซิร์ฟเวอร์ได้

3.2 องค์ประกอบของดาต้าเบลดโมดูล

- ข้อมูลชนิดใหม่ที่จะเพิ่มเข้าไปในเซิร์ฟเวอร์ เช่น ชนิดข้อมูลแบบโอเปค
- รูทีนอาจสร้างขึ้นโดยภาษาซีหรือ SPL ที่สามารถถูกเรียกใช้โดยดาต้าเบสเซิร์ฟเวอร์
- คาส์ท(Cast) เป็นรูทีนที่จะทำการแปลงชนิดข้อมูล
- อินเทอร์เฟส (Interface) การเรียกใช้ดาต้าเบลดโมดูลอื่น ๆ
- ชนิดข้อมูลแบบควอลิไฟด์ (Qualified Data type) เป็นการกำหนดชนิดข้อมูลแบบบิวท์อินให้เฉพาะมากยิ่งขึ้น โดยมีการจัดเตรียมข้อมูลเกี่ยวกับขนาดของพื้นที่ในการเก็บข้อมูล (Storage size) และขอบเขตของค่าข้อมูล ซึ่งบิวท์อินฟังก์ชัน แต่ละชนิดจะมีความสามารถในการกำหนดชนิดข้อมูลแบบควอลิไฟด์ที่แตกต่างกัน เช่น ชนิดข้อมูลแบบ CHAR ก็สามารถกำหนดขนาดได้เช่น CHAR(16) หรือชนิดข้อมูลแบบ DECIMAL ก็สามารถกำหนดเป็น DECIMAL(precision,scale) เป็นต้น

- ตาราง (Table) และอินเด็กซ์ที่จะสนับสนุนการทำงานร่วมกับชนิดข้อมูลแบบใหม่ๆ
- รหัสชนิดความผิดพลาด (Error Code) เป็นการกำหนดข้อความอธิบายความผิดพลาด (Error Message) ให้ตรงกับความต้องการของผู้ใช้
- ไคลเอนท์โค้ด (Client Code) เป็นโค้ดที่จะทำการคัดลอกไปยังไคลเอนท์ขณะทำการเพิ่ม ดาต้าเบลดโมดูลเข้าไปในดาต้าเบสเซิร์ฟเวอร์

3.3 ขั้นตอนการจัดทำดาต้าเบลดโมดูล

ขั้นตอนในการสร้างดาต้าเบลดโมดูลมีดังนี้

1. ออกแบบดาต้าเบลดโมดูล

ทำการออกแบบว่าจะรวมส่วนประกอบอะไรบ้างเข้าไปในดาต้าเบลดโมดูล ซึ่งอาจจะเป็นชนิดข้อมูลแบบผู้ใช้กำหนดเอง รูทีนแบบกำหนดเอง หรือองค์ประกอบอื่น ๆ ที่ต้องการ

2. ทำการจัดสร้างดาต้าเบลดโปรเจ็ค (DataBlade Project)

โดยใช้โปรแกรมเบลดสมิธ (Blade Smith) ซึ่งโปรแกรมเบลดสมิธจะสามารถเพิ่มส่วนต่าง ๆ เข้าไปได้ ซึ่งองค์ประกอบเหล่านั้นประกอบด้วย

- ออบเจกต์ที่สามารถกำหนดเองได้ (User – Defined Object) เป็นส่วนที่ผู้ใช้สามารถสร้างขึ้นเองได้ ประกอบด้วย
  - *แอกกริเกตส์ (Aggregates)*
  - *คาส์ท (Casts)*
  - *ความผิดพลาด (Errors)*
  - *อินเทอร์เฟส (Interface)*
  - *รูทีน (Routines)*
  - *ชนิดของข้อมูล (Types)*
    - ชนิดข้อมูลแบบคอลเลคชัน
    - ชนิดข้อมูลแบบดิสทิงค์
    - ชนิดข้อมูลแบบโอเปค
- อิมพอร์ตออบเจ็ค (Import Objects) เป็นชนิดข้อมูลแบบบิวท์อิน และอินเทอร์เฟสจากโมดูลอื่น ๆ
- ไฟล์ (File) ประกอบด้วย
  - *SQL files* เป็นการกำหนดรูปแบบคำสั่ง SQL ขึ้นมาเอง เพื่อเพิ่มเข้าไปในสคริปต์ (Script) ที่ใช้ในการอธิบายถึงดาต้าเบลดโมดูล และออบเจ็คของมัน
  - *Client files* เป็นไฟล์ที่จะถูกดาวน์โหลด (Download) ไปยังเครื่องเวิร์คสเตชัน (Work Station) เมื่อดาต้าเบลดโมดูลถูกติดตั้ง (Install) เข้าไปในอินฟอร์มิกซ์เซิร์ฟเวอร์

เมื่อสร้างดาต้าเบลดโปรเจ็คเสร็จแล้ว จะทำการสร้างไฟล์โดยอัตโนมัติซึ่งไฟล์ที่ถูกสร้างขึ้นแบ่งออกเป็นส่วนต่าง ๆ ดังนี้

- *SQL Scripts* จะถูกเอ็กซิคิวต์ (Execute) ในระหว่างที่เบลดเมเนเจอร์ทำการรีจิสเตอร์ดาต้าเบลดโมดูลเข้าไปในฐานข้อมูล
- *Source files* เป็นโค้ดพื้นฐาน ของรูทีนที่นิยามไว้ในโปรเจ็ค
- *Fundamental test files*
- *Setup files* จะใช้ในการสร้างแพ็กเกจที่ใช้ในการติดตั้ง (Install Package) โดยโปรแกรมเบลดแพ็ก (Blade Pack)

3. *โปรแกรมเบลดแพ็ก*

เปิดโปรเจ็คที่สร้างโดยเบลดสมิธซึ่งมีนามสกุล (.PRD) และทำการสร้างแพ็กเกจที่ใช้ในการติดตั้ง ซึ่งจะเป็นดาต้าเบลดโมดูลที่จะเพิ่มเข้าไปในดาต้าเบสเซิร์ฟเวอร์

4. *โปรแกรมเบลดเมเนเจอร์*

จะเป็นโปรแกรมที่จะนำดาต้าเบลดโมดูลที่มีอยู่ลงทะเบียนเข้าไปในฐานข้อมูลที่ต้องการ โดยจะต้องเลือกดาต้าเบสเซิร์ฟเวอร์ และดาต้าเบสเนม (Database name) ที่ต้องการก่อน

5. *โปรแกรม Schema Knowledge*

จะมีหน้าที่ในการตรวจสอบว่าดาต้าเบลดโมดูลที่เรารีจิสเตอร์เข้าไปอยู่ในดาต้าเบสเนมที่เราต้องการหรือไม่ นอกจากนั้นยังดูข้อมูลรายละเอียด (Information) อื่น ๆ ของฐานข้อมูลได้

3.4 สรุปการวิเคราะห์คุณสมบัติทางออบเจกต์ของดาต้าเบลดโมดูล

ดาต้าเบลดโมดูลไม่ถือว่าเป็นออบเจกต์ด้วยเหตุผลดังนี้

- ดาต้าเบลดโมดูลเปรียบได้เสมือนกับคลาส (Class) แต่ไม่สามารถสร้างออบเจ็คในคลาสนั้นได้ ภายในดาต้าเบลดโมดูลเป็นเพียงการกำหนดชนิดของข้อมูล และรูทีนใหม่ๆ เพิ่มเข้าไปในดาต้าเบสเซิร์ฟเวอร์เท่านั้น
- เราไม่สามารถเพิ่มหรือลบดาต้าเบลดโมดูลได้การเขียนคำสั่งในโค้ด การเพิ่มดาต้าเบลดโมดูล 1 โมดูล จะต้องทำการรีจิสเตอร์หรือทำการติดตั้งเข้าไปในเซิร์ฟเวอร์ผ่านทางโปรแกรม
- ดาต้าเบลดโมดูลไม่ได้ซ่อนข้อมูล (Data) และรูทีนไว้ภายใน เมื่อเรารีจิสเตอร์ดาต้าเบลดโมดูลเข้าไปในเซิร์ฟเวอร์แล้ว รูทีนและชนิดของข้อมูลใหม่เหล่านั้นจะถูกรวม และถูกใช้ในระดับเดียวกันกับบิวท์อินฟังก์ชัน (Built-in function) หรือชนิดข้อมูลแบบบิว์อินซึ่งไม่สนับสนุนหลักการของออบเจกต์
- ถ้าออบเจ็ค หมายถึง สิ่งของ (things) สิ่งหนึ่ง ซึ่งประกอบด้วยข้อมูลและพฤติกรรมของมัน ดาต้าเบลดโมดูลไม่ได้สนับสนุนคำพูดนี้ เนื่องจากภายในดาต้าเบลดโมดูล นอกจากจะมีรูทีน

และชนิดของข้อมูล แล้วยังมีส่วนอื่นๆอีก เช่น SQL code ฟังก์ชัน ข้อมูลทดสอบ (test) คู่มือ (help) และ เอกสาร (Document)

- ดาต้าเบลดโมดูลไม่สนับสนุนการสืบทอดคุณสมบัติ (Inheritance) เนื่องจากดาต้าเบลดโมดูลอยู่ในระดับเดียวกันในเซิร์ฟเวอร์ไม่มีคุณสมบัติ "type of" หรือ คลาส หรือ คลาสย่อย (subclass)

บทที่ 4
ข้อมูลชนิดโอเปค

ชนิดของข้อมูลแบบโอเปค คือโครงสร้างข้อมูลพื้นฐาน ที่โครงสร้างภายในไม่สามารถเข้าถึงได้โดยดาต้าเบสเซิร์ฟเวอร์ ซึ่งจะไม่เหมือนโครงสร้างภายในของชนิดของข้อมูลบิวท์อิน เมื่อมีการสร้าง ชนิดของข้อมูลแบบโอเปคจะต้องมีการจัดเตรียมสิ่งต่าง ๆ ดังต่อไปนี้

4.1 การสร้างชนิดของข้อมูลแบบโอเปค

ชนิดของข้อมูลแบบโอเปคถือว่าเป็นชนิดของข้อมูลที่ไม่สามารถแยกย่อยได้อีกแล้ว (Atomic Data type) ที่ถูกกำหนดขึ้นมาเพื่อใช้ในฐานข้อมูล ซึ่งดาต้าเบสเซิร์ฟเวอร์จะไม่รู้จักโครงสร้างภายในของมัน นั่นคือชนิดของข้อมูลแบบโอเปคจะซ่อนบัง (Encapsulated) ข้อมูลไว้ภายใน ทำให้อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์จะไม่ทราบถึงข้อเท็จจริงเกี่ยวกับรูปแบบการจัดเก็บข้อมูลภายในของมัน

เมื่อมีการนิยามชนิดของข้อมูลแบบโอเปคจะเป็นการขยายหรือเพิ่มเติมชนิดข้อมูลที่ดาต้าเบสเซิร์ฟเวอร์ไม่มี เมื่อทำการติดตั้งชนิดของข้อมูลแบบโอเปคเข้าไปในดาต้าเบสเซิร์ฟเวอร์แล้ว การใช้งานจะอยู่ในระดับเดียวกันกับชนิดของข้อมูลแบบบิวท์อิน

สำหรับชนิดของข้อมูลแบบโอเปคนั้นจะต้องเขียนโครงสร้างภายในด้วยภาษาซี ซึ่งก่อนที่จะทำการจัดสร้าง จะต้องมีการเตรียมข้อมูลต่าง ๆ ดังต่อไปนี้

- โครงสร้างข้อมูล (Data Structure) ที่จะระบุถึงการจัดเก็บข้อมูลภายใน (Internal Storage) ของชนิดของข้อมูลแบบโอเปค
- ฟังก์ชันสนับสนุนการทำงาน (Support Function) ที่อนุญาตให้ดาต้าเบสเซิร์ฟเวอร์ติดต่อกับชนิดของข้อมูลแบบโอเปคได้
- รูทีนอื่น ๆ ที่จำเป็นของฟังก์ชันสนับสนุนการทำงาน ที่เป็นการเรียกใช้ของเอนท์ยูสเซอร์ (End-users)

4.2 การจัดเก็บข้อมูลภายใน (Internal Storage)

ในการสร้างชนิดของข้อมูลแบบโอเปค อย่างแรกที่ต้องทำคือ การจัดเตรียมโครงสร้างข้อมูลที่จะเก็บข้อมูลลงในฐานข้อมูลโดยจะเป็นอินเทอร์นอลเรพพลิเซนเทชัน (Internal Representation) ซึ่งเรียกโครงสร้างข้อมูลนี้ว่าอินเทอร์นอลสตอเรจของชนิดของข้อมูลแบบโอเปคจะเป็นตัวบอกว่าข้อมูลจะถูกเก็บลงบนอุปกรณ์จัดเก็บอย่างไร ฟังก์ชันสนับสนุนการทำงานที่ถูกเขียนขึ้นมาจะทำงานบนอินเทอร์นอลสตรัคเจอร์ (Internal Structure) ซึ่งดาต้าเบสเซิร์ฟเวอร์ จะไม่สามารถเห็นมัน

เราสามารถเขียนโครงสร้างข้อมูลภายในได้โดยใช้ภาษาซี สำหรับโครงสร้างข้อมูลภายในซึ่งจะสนับสนุนชนิดของข้อมูลแบบโอเปค 2 ชนิด คือ

- *ชนิดของข้อมูลแบบโอเปคที่มีขนาดคงที่ (Fixed – Length Opaque type)*

มีโครงสร้างข้อมูลภายในซึ่งมีขนาดของชนิดของโอเปค (Opaque type) ทุก ๆ ตัวเท่า ๆ กัน หมด ซึ่งจะมีประโยชน์สำหรับข้อมูลที่จะสนับสนุนที่มีความยาวของแต่ละฟิลด์คงที่ เช่น ข้อมูลชนิดตัวเลข (Numeric) เป็นต้น

- *ชนิดของข้อมูลแบบโอเปคที่มีขนาดเปลี่ยนแปลงได้ (Vary – Length Opaque type)*

ค่าข้อมูลในชนิดของโอเปคอาจมีขนาดไม่คงที่ เมื่อเป็นข้อมูลคนละตัวจะมีประโยชน์ในการเก็บมัลติเรพพลิเซนเทชันดาต้า (Multi-representation data) เช่น รูปภาพ ซึ่งรูปภาพแต่ละรูป สามารถที่จะมีขนาดที่แตกต่างกันได้ แต่จะต้องมีการเก็บข้อมูลไม่เกินขนาดที่กำหนดไว้ และจะใช้สมาร์ทลาก์จออบเจกต์ (Smart-Large Object) เมื่อข้อมูลชนิดของโอเปคมีขนาดเกินกว่าที่กำหนด

4.3 ฟังก์ชันสนับสนุนการทำงาน (Support Function)

เป็นตัวที่จะบอกดาต้าเบสเซิร์ฟเวอร์ว่าจะตอบโต้กับโครงสร้างภายในของโอเปคได้อย่างไร เนื่องจากว่าชนิดของโอเปคนั้นถูกซ่อนบังเอาไว้ภายใน ดาต้าเบสเซิร์ฟเวอร์ซึ่งไม่รู้ถึงข้อมูลรายละเอียดภายในโอเปคได้ ดังนั้นการติดต่อกับโอเปคจึงต้องใช้ฟังก์ชันสนับสนุนการทำงาน

ตารางต่อไปนี้แสดงฟังก์ชันสนับสนุนการทำงาน ที่สำคัญ ๆ

ฟังก์ชันสนับสนุนการทำงาน	คำอธิบาย
อินพุต (Input)	- ทำการแปลงชนิดของข้อมูลแบบโอเปคจากเอ็กซ์เทอร์นอลเรพพลิเซนเทชัน (External representation) ไปเป็นอินเทอร์นอลเรพพลิเซนเทชัน (Internal representation)
เอาต์พุต (Output)	- ทำการแปลงชนิดของข้อมูลแบบโอเปคจากอินเทอร์นอลเรพพลิเซนเทชันไปเป็นเอ็กซ์เทอร์นอลเรพพลิเซนเทชัน
รีซีฟ (Receive)	- ทำการแปลงชนิดของข้อมูลแบบโอเปค จากอินเทอร์นอลเรพพลิเซนเทชันบนเครื่องไคลเอนท์เป็น เอ็กซ์เทอร์นอลเรพพลิเซนเทชันของเครื่องเซิร์ฟเวอร์
เซนด์ (Send)	- ทำการแปลงอินเทอร์นอลเรพพลิเซนเทชันของเครื่องเซิร์ฟเวอร์ไปเป็นอินเทอร์นอลเรพพลิเซนเทชันของเครื่องไคลเอนท์
อิมพอร์ต (Import)	- กระทำงานต่าง ๆ ที่ต้องใช้ในการประมวลผลโอเปค เมื่อมีการนำเข้า ชนิดของโอเปคขนาดใหญ่ในเอ็กซ์เทอร์นอลเรพพลิเซนเทชัน
เอ็กซ์พอร์ต (Export)	- กระทำงานต่าง ๆ ที่ต้องการในการประมวลผลชนิดของโอเปคเมื่อมีการนำออกชนิดของโอเปคขนาดใหญ่ ในเอ็กซ์เทอร์นอลเรพพลิเซนเทชัน

4.4 SQL อินโวครูทีน (SQL invoke routine)

ฟังก์ชันสนับสนุนการทำงานได้จัดเตรียมฟังก์ชันพื้นฐานที่จำเป็นมาให้เป็นการที่ดาต้าเบสเซิร์ฟเวอร์จะตอบโต้กับชนิดของข้อมูลแบบโอเปคอย่างไรก็ตามรูทีนที่กำหนดขึ้นเองเพิ่มขึ้นมา ซึ่งชนิดของรูทีนที่สามารถกำหนดเองได้แบ่งตามหน้าที่การทำงานได้ดังนี้

1. บิวท์อินฟังก์ชัน (Built-in Function)

เป็นฟังก์ชันที่กำหนดไว้ก่อนแล้ว โดยดาต้าเบสเซิร์ฟเวอร์เพื่อใช้ในคำสั่ง SQL โดยบิวท์อินฟังก์ชันสามารถทำงานได้เลยบนชนิดของข้อมูลแบบบิวท์อิน ในการใช้งานบิวท์อินฟังก์ชันเหล่านั้นชนิดของข้อมูลแบบโอเปคจะต้องเขียนเวอร์ชัน (Version) ใหม่ที่ยอมรับชนิดของโอเปคนั้นบนพารามิเตอร์ลิสต์ (Parameter list)

2. โอเปอเรชันฟังก์ชัน (Operator function)

คือฟังก์ชันที่กำหนดขึ้นเอง (User-defined Function) ที่ตรงกับสัญลักษณ์ (Symbol) ของโอเปอเรเตอร์ (Operator) ซึ่งการใช้โอเปอเรชันฟังก์ชันจะต้องเขียนเวอร์ชันที่ยอมรับ ชนิดของโอเปคนั้นบนพารามิเตอร์ลิสต์

3. แอกกริเกชันฟังก์ชัน (Aggregation function)

จะคืนค่า 1 ค่า ในคำสั่งคิวรี สำหรับอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ของอินฟอร์มิกซ์จะยอมรับแอกกริเกชันฟังก์ชันเหล่านี้เท่านั้น สำหรับชนิดของข้อมูลแบบโอเปค

- COUNT
- MIN
- MAX

4. เอนท์ยูสเซอร์รูทีน (End-user routines)

ยูนิเวอร์แซลเซิร์ฟเวอร์อนุญาติให้เราสามารถที่จะนิยาม SQL อินโวคฟังก์ชัน หรือโปรซีเดอร์ที่เอนท์ยูสเซอร์สามารถที่จะใช้ได้ผ่าน SQL Statement ซึ่งเอนท์ยูสเซอร์รูทีนจะจัดเตรียมหน้าที่ต่าง ๆ ที่เอนท์ยูสเซอร์สามารถทำงานได้ บนชนิดของข้อมูลแบบโอเปค ตัวอย่างของเอนท์ยูสเซอร์รูทีนคือ

- ฟังก์ชันที่จะส่งกลับค่า ๆ หนึ่งในโอเปคเนื่องจากโอเปคได้ซ่อนข้อมูลไว้ภายใน ดังนั้นการสร้างเอนท์ยูสเซอร์รูทีนจึงเป็นวิธีเดียวที่จะเข้าถึงข้อมูลภายในโอเปค
- คาส์ททิงฟังก์ชัน (Casting function) สามารถเขียนคาส์ททิงฟังก์ชัน ระหว่างชนิดของโอเปคกับชนิดของข้อมูลอื่นได้เช่นกัน
- ฟังก์ชันหรือโปรซีเดอร์ที่จะกระทำโอเปอเรชันร่วมกันบน ชนิดของข้อมูลแบบโอเปค

4.5 ขั้นตอนการจัดสร้าง ชนิดของข้อมูลแบบโอเปค

การสร้างชนิดของข้อมูลแบบโอเปคมีขั้นตอนดังนี้

1. สร้างโครงสร้างภายใน ด้วยภาษาซี สำหรับชนิดของข้อมูลแบบโอเปคที่ต้องการ
2. เขียนฟังก์ชันสนับสนุนการทำงาน ด้วยภาษาซี
5. ริจิสเตอร์ชนิดของข้อมูลแบบโอเปคในฐานข้อมูลด้วยคำสั่ง CREAT OPAQUE TYPE

6. รีจิสเตอร์ฟังก์ชันสนับสนุนการทำงาน ในฐานข้อมูลด้วยคำสั่ง CREAT FUNCTION
7. จัดเตรียมการเข้าถึงชนิดของโอเปค และฟังก์ชันสนับสนุนการทำงาน โดยใช้คำสั่ง GRANT
8. เขียน SQL อินโวคฟังก์ชัน เพื่อสนับสนุน ชนิดของข้อมูลแบบโอเปค
9. จัดเตรียมการกำหนดวิธีการเข้าถึงที่กำหนดเอง (Customized Secondary Access Methods) ที่ชนิดของโอเปคต้องการ

4.6 การออกแบบคลาสด้วยชนิดของข้อมูลแบบโอเปค

1. เราสามารถสร้างคลาสของออบเจกต์ได้โดยการสร้างชนิดของข้อมูลแบบโอเปค
2. ภาษาที่ใช้เขียนโอเปคขึ้นมาคือภาษาซีเท่านั้น
3. เราสามารถสร้างออบเจ็คที่อยู่ในคลาสได้โดยการประกาศในดาต้าเบลดโมดูลหรือที่อื่นๆ เช่นเดียวกับการประกาศตัวแปรของชนิดของข้อมูลแบบบิวท์อิน
4. เราสามารถประกาศ และใช้ออบเจกต์ได้ในส่วนของโค้ด
5. ชนิดของข้อมูลแบบโอเปคมีคุณสมบัติบางอย่างที่เป็นออบเจ็คเท่านั้นคือ
   - การซ่อนบังข้อมูล (Data Encapsulation) สามารถรวมชุดของข้อมูลไว้ในโครงสร้างข้อมูล การจัดการเก็บข้อมูลภายในสามารถทำได้โดยผ่านฟังก์ชันสนับสนุนการทำงาน แต่ชนิดของข้อมูลแบบโอเปคไม่ได้รวมส่วนที่เป็นรูทีนไว้ภายใน จึงไม่มีระดับการเข้าถึง เช่น ไพรเวท (private), พลับบลิค(public) หรือหลักการของเฟรนด์คลาส (friend class)
   - การสืบทอดคุณสมบัติ ไม่มีหลักการนี้เลยเนื่องจากชนิดของข้อมูลแบบโอเปคที่สร้างขึ้นเป็นเพียงการกำหนดโครงสร้างข้อมูลเท่านั้น ไม่ได้สร้างเป็นคลาสจึงไม่สามารถสร้างคลาสย่อยขึ้นมาได้
   - โพลิมอร์ฟิซึม (Polymorpism) อยู่ในรูปของรูทีนโอเวอร์โหลดดิง (routine-overloading) เราสามารถสร้างรูทีนต่างๆ เพื่อจัดการเก็บข้อมูลในโอเปคได้โดยอิสระ เพียงแต่รูทีนต่างๆไม่ได้ถูกรวมเข้าไว้ในชนิดของข้อมูลแบบโอเปคเท่านั้น รูทีนและโอเปอเรเตอร์ต่างๆที่สร้างขึ้นในเบลดสมิธเมื่อทำการสร้างโค้ดเป็นภาษาซีแล้วจะถูกสร้างเป็นรูทีนในระดับเดียวกันหมดและถูกเก็บไว้ในไฟล์
6. โครงสร้างข้อมูลของโอเปคหรือชนิดของข้อมูลอื่นที่สร้างขึ้นโดยเบลดสมิธจะถูกสร้างเก็บไว้ในเฮดเดอร์ไฟล์ (Header file)

บทที่ 5

ภาษา SPL

ภาษา SPL (Stored Procedure Language) คือรูทีนของผู้ใช้ที่สามารถเขียนขึ้นเอง และถูกจัดเก็บไว้ในดาต้าเบสเซิร์ฟเวอร์ซึ่งเป็นการขยายความสามารถของการเขียนภาษา SQL ซึ่ง SPL จะเป็นการนำภาษา SQL เข้ามาเขียนเป็นชุดคำสั่ง มีเงื่อนไข และมีลูปการทำงานที่สามารถกำหนดได้โดยผู้ใช้ เพื่อความคุมฐานข้อมูล, แก้ไข และจัดการข้อมูลในฐานข้อมูลได้อย่างมีระบบ ผู้ใช้สามารถใช้ภาษา SPL ในการทำงานใด ๆ ก็ได้เช่นเดียวกันกับการใช้ภาษา SQL ซึ่งการเก็บ SPL procedure ไว้ในฝั่งของเซิร์ฟเวอร์จะเป็นการช่วยลดความหนาแน่นของข้อมูลระหว่างแอพพลิเคชันฝั่งไคลเอนท์กับดาต้าเบสเซิร์ฟเวอร์ได้

ในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์นั้น SPL รูทีน จะแบ่งเป็น 2 ชนิด คือ SPL โปรซีเดอร์ คือรูทีนที่มีการทำงานแต่ไม่มีการส่งค่ากลับมายังผู้เรียกใช้ และ SPL ฟังก์ชัน จะเป็นรูทีนที่มีการส่งค่ากลับสู่ผู้เรียกใช้ ซึ่งสามารถที่จะส่งค่ากลับเพียงค่าเดียวหรือส่งกลับเป็นหลายค่า

5.1 การเขียน SPL รูทีน

SPL Routine จะประกอบด้วย ส่วนเริ่มต้นบล็อคของคำสั่ง SPL และส่วนปิดท้าย ซึ่งมีรูปแบบ 2 รูปแบบ คือรูปแบบของการสร้าง SPL โปรซีเดอร์ เป็นดังนี้

```
CREATE PROCEDURE new_price (per_cent REAL)
    ...
    ...
    ...
END PROCEDURE;
```

รูปแบบการสร้าง SPL ฟังก์ชัน เป็นดังนี้

```
CREATE FUNCTION discount_price (per_cent REAL)
    RETURNING MONEY;
    ...
    ...
    ...
END FUNCTION;
```

ซึ่ง SPL ฟังก์ชัน สามารถส่งกลับค่าเดียวหรือหลายค่าได้ โดยกำหนดในส่วนของ RETURNING ซึ่งอาจจะเป็นชนิดที่มีอยู่แล้วหรืออาจจะเป็นชนิดข้อมูลที่สร้างขึ้นมาเองก็ได้ เช่น ชนิดข้องข้อมูลแบบโอเปค เป็นต้น

5.1.1 การกำหนดชื่อรูทีน

การกำหนดชื่อของรูทีน สามารถทำได้โดยตรงโดยการใช้คำสั่ง CREATE PROCEDURE หรือ CREATE FUNCTION ซึ่งอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์อนุญาตให้ผู้ใช้สามารถสร้าง SPL รูทีนมากกว่า 1 รูทีนที่มีชื่อเดียวกัน แต่มีพารามิเตอร์ที่แตกต่างกันได้ ซึ่งเรียกว่ารูทีนโอเวอร์โหลดดิง (Routine Overloading) ดังตัวอย่างต่อไปนี้

CREATE PROCEDURE multiply (a INT,b BASETYPE1) ...
CREATE PROCEDURE multiply (a INT,b BASETYPE2) ...
CREATE PROCEDURE multiply (a INT,b BASETYPE3) ...

เมื่อผู้ใช้เรียกใช้รูทีน Multiply ดาต้าเบสเซิร์ฟเวอร์จะทำการประเมินชื่อของรูทีน และอาร์กิวเมนท์ (Agrument) เพื่อกำหนดรูทีนที่จะทำการเอ็กซิคิวท์ รูทีนรีโซลูชัน (Execute Routine Resolution) คือกระบวนการที่ดาต้าเบสเซิร์ฟเวอร์ค้นหาสำหรับรูทีนซิกเนเจอร์ (Routine Signature) ที่จะใช้ แต่ละรูทีนจะมีซิกเนเจอร์ที่แตกต่างกันที่จะระบุถึงแต่ละรูทีน ซึ่งรูทีนซิกเนอเจอร์จะประกอบด้วยสิ่งต่าง ๆ ดังต่อไปนี้

- ชนิดของรูทีน (เป็นฟังก์ชัน หรือโปรซีเดอร์)
- ชื่อของรูทีน
- จำนวนของพารามิเตอร์
- ชนิดข้อมูลของพารามิเตอร์
- การจัดลำดับของพารามิเตอร์

5.1.2 การเพิ่มชื่อเฉพาะ

SPL รูทีน ที่มีการโอเวอร์โหลดดิง สามารถกำหนดชื่อที่ไม่ซ้ำกันได้เพื่อการป้องกันการสับสนของผู้เขียนโปรแกรม ซึ่งรูทีนสามารถมีชื่อเฉพาะ ซึ่งจะไม่ซ้ำกันในฐานข้อมูลเดียวกัน เว้นแต่ว่าจะมีเจ้าของ (Owner) แตกต่างกัน ซึ่งสามารถสร้างชื่อเฉพาะได้ในประโยคคำสั่ง CREATE PROCEDURE หรือ CREATE FUNCTION ดังตัวอย่างต่อไปนี้

```
CREATE FUNCTION calculate (a INT , b INT , c INT)
        RETURNING INT;
        SPECIFIC calc1;
        ...
        ...
END FUNCTION;
```

5.1.3 การเพิ่มพารามิเตอร์ลิสต์

พารามิเตอร์ของ SPL รูทีน จะมีหรือไม่มีก็ได้ ถ้ามีก็สามารถมีได้มากกว่า 1 ตัว ชนิดของข้อมูลพารามิเตอร์สามารถเป็นชนิดข้อมูลดังต่อไปนี้

- ชนิดของข้อมูลแบบบิวท์อิน
- ชนิดของข้อมูลแบบโอเปค
- ชนิดของข้อมูลแบบดิสทิงค์
- ชนิดของข้อมูลแบบแถว
- ชนิดของข้อมูลแบบคอลเลคชัน

ชนิดข้อมูลที่ไม่สามารถใช้เป็นพารามิเตอร์ได้ มีดังนี้

- SERIAL
- SERIAL8
- TEXT
- BYTE
- CLOB
- BLOB

แม้ว่าเราจะไม่สามารถใช้ TEXT และ BYTE เป็นพารามิเตอร์ได้ แต่เราก็สามารถใช้การอ้างอิง (Reference) ช่วยได้ ซึ่งเป็นพอยน์เตอร์ (Pointer) ที่ชี้ไปยัง TEXT และ BYTE แทน ดังนี้

```
CREATE PROCEDURE pro1(lo_text,REFERENCES TEXT)
```

5.1.4 การเพิ่มประโยคคำสั่ง RETURN

เมื่อสร้างฟังก์ชันจะต้องมีการกำหนดค่าที่จะส่งกลับ อาจจะเป็นค่าเพียงค่าเดียวหรือว่าหลายค่าโดยใช้ RETURNING ตามด้วยชนิดของข้อมูลที่จะส่งกลับซึ่งยกเว้น SERIAL SERIAL8 TEXT BYTE CLOB หรือ BLOB ตัวอย่างที่จะแสดง จะเป็นรูทีน ที่จะส่งกลับค่าข้อมูลที่ INTEGER และ REAL

```
CREATE FUNCTION find_group(id INT)
        RETURNING INT,REAL;
        ...
        ...
END FUNCTION;
```

เมื่อสร้างฟังก์ชันที่มีการส่งกลับ จะต้องมีการกำหนดคำสั่ง RETURN ภายในฟังก์ชันด้วย เมื่อฟังก์ชันมีการกระทำกับข้อมูลชนิด TEXT หรือ BYTE ก็จะต้องมีการใช้ประโยค REFERENCES เป็นพอยน์เตอร์ชี้ไปยังข้อมูลแบบ TEXT หรือ BYTE ดังนี้

```
CREATE FUNCTION find_obj (id INT)
        RETURNING REFERENCES BYTE;
        ...
        ...
END FUNCTION;
```

5.1.5 การดร็อป SPL รูทีน

เมื่อมีการสร้าง SPL รูทีน จะไม่สามารถแก้ไขภายในรูทีนได้ จะต้องมีการดรอป (Drop) รูทีนนั้นทิ้ง และสร้างขึ้นใหม่ เพราะฉะนั้นหลังจากมีการคัดลอกคำสั่งในการสร้างรูทีนไว้นอกฐานข้อมูลก่อนที่จะมีการดรอปรูทีน การดรอปจะต้องให้ตรงกับชนิดของรูทีนด้วย ได้แก่

```
DROP PROCEDURE raise_prices;
DROP FUNCTION  calculate;
```

ถ้ารูทีนมีการใช้รูทีนโอเวอร์โหลดดิง หรือการใช้ชื่อเฉพาะ การดรอปจะต้องมีการกำหนดซิกเนอเจอร์ หรือชื่อเฉพาะด้วย ดังนี้

```
DROP ROUTINE calculate;
DROP SPECIFIC ROUTINE calc1;
```

นอกจากนี้สามารถจะดรอปรูทีน โดยอ้างถึงชื่อแบบเดิมได้ โดยมีรูปแบบ

database@dbservername:owner.routinename;

5.2 การนิยามและการใช้ตัวแปร

ตัวแปรใน SPL รูทีน สามารถกำหนดได้ในพารามิเตอร์ลิสต์ และภายในตัวของรูทีน ซึ่งตัวแปรจะถูกในเก็บไว้ในหน่วยความจำ (memory) ไม่ใช่ในฐานข้อมูล ดังนั้น การโรลแบ็ก (Rollback) ของทรานแซกชัน (Transaction) จะไม่มีการคืนค่าของตัวแปร

การนิยามตัวแปร ใน SPL รูทีน สามารถใช้ประโยคคำสั่ง DEFINE ซึ่งปรากฏหลังจากประโยคคำสั่ง CREATE PROCEDURE และก่อนที่จะเขียนประโยคคำสั่งอื่น ๆ ดังตัวอย่างต่อไปนี้

```
DEFINE a INT;
DEFINE colors COLLECTION;
DEFINE GLOBAL gl_out INT DEFAULT 13;
```

5.2.1 การประกาศตัวแปรโลคอล

ตัวแปรโลคอล (Local variable) จะมีลักษณะดังต่อไปนี้

- จะมองเห็นและใช้งานได้เฉพาะภายใน SPL รูทีนเท่านั้น
- จะมีการรีเซตค่าของตัวแปรทุกครั้งที่มีการเอ็กซีคิวต์รูทีน
- ไม่สามารถมีค่าเริ่มต้น (Default value) ของตัวแปรได้

การกำหนดค่าให้กับตัวแปร สามารถใช้ประโยคคำสั่ง LET เช่น

5.2.1.1 การนิยามตัวแปรชนิดบิวต์อิน

ตัวแปรชนิดบิวต์อินจะได้จากการรับค่าจากชนิดข้อมูลแบบบิวต์อิน เราสามารถประกาศตัวแปรแบบบิวต์อินได้ ยกเว้นตัวแปรชนิด SERIAL SERIAL CLOB และ BLOB การประกาศแสดงได้ดังนี้

```
DEFINE x INT;
DEFINE y INT8;
DEFINE name CHAR(15);
DEFINE today DATETIME YEAR TO DAY;
```

5.2.1.2 การประกาศตัวแปรแบบ Simple Large Objects

ตัวแปรแบบ Simple Large Objects ได้แก่ TEXT หรือ BYTE ไม่สามารถบรรจุตัวออบเจกต์ โดยตรง แต่จะสามารถอ้างถึงโดยการใช้พอยน์เตอร์ชี้ไปยังออบเจ็คนั้น ดังต่อไปนี้

```
DEFINE t REFERENCES TEXT;
DEFINE b REFERENCES BYTE;
```

5.2.1.3 การประกาศตัวแปรแบบคอลเลคชัน

ในการจัดเก็บตัวแปรแบบคอลเลคชันจากฐานข้อมูล ตัวแปรจะต้องเป็นชนิด COLLECTION ,LIST SET หรือ MULTISET ตัวแปรชนิดคอลเลคชันถูกเรียกว่าตัวแปร untyped collection variable เนื่องจากสามารถจัดเก็บข้อมูลชนิดคอลเลคชันแบบใดก็ได้ ซึ่งสามารถนิยามได้ดังนี้

DRFINE a COLLECTION;

ซึ่งตัวแปร a สามารถจัดเก็บได้ทั้งข้อมูลที่เป็น LIST SET หรือ MULTISET สำหรับชนิดข้อมูลแบบ SET MULTISET และ LIST นั้นถูกเรียกว่า type collection variable ซึ่งจะจัดเก็บได้เฉพาะชนิดของข้อมูลแบบคอลเลคชัน ชนิดที่ระบุเท่านั้น ดังตัวอย่างต่อไปนี้

DEFINE a SET (INT NOT NULL);
DEFINE b MULTISET (ROW(b1 INT ,b2 CHAR(50)) NOT NULL);
DEFINE c LIST (SET (DECIMAL NOT NULL) NOT NULL);

5.2.1.4 การประกาศตัวแปรชนิดแถว

การกำหนดข้อมูลชนิดแถวใน SPL routine จะมีอยู่ 3 ลักษณะ คือ generic row variable , name row variable และ unnamed row variable สำหรับ generic row variable สามารถจัดเก็บค่าข้อมูล row type ชนิดต่าง ๆ ได้ กำหนดได้ดังต่อไปนี้

DEFINE d ROW;

สำหรับ name row variable คือมี Row type ที่ถูกสร้างไว้แล้วโดยใช้ประโยคคำสั่ง CREATE ROW TYPE และประกาศใน SPL routine ดังตัวอย่างต่อไปนี้

DEFINE person employee_t;

ตัวแปรที่ใช้จัดเก็บ unname row type จะใช้ประโยค ROW ตามด้วยรายละเอียดอื่น ๆ ดังตัวอย่าง

```
DEFINE manager ROW( Name VARCHAR(30),
                    Department VARCHAR(30),
                    Salary INTEGER);
```

5.2.1.5 การประกาศตัวแปรแบบโอเปคและ Distinct Type

ตัวแปรชนิดโอเปคจะเก็บค่าข้อมูลจากชนิดข้อมูลแบบโอเปค ส่วน Distinct – type variable จะเก็บข้อมูลแบบ Distinct ตัวอย่างเช่น มีการนิยามชนิดของข้อมูลแบบโอเปคชื่อ point และ Distinct – type ชื่อ centerpoint สามารถประกาศตัวแปรดังต่อไปนี้

```
DEFINE a point;
DEFINE b centerpoint;
```

5.2.2 การประกาศตัวแปรแบบโกลบอล

ตัวแปรแบบโกลบอล (Global) คือค่าที่เก็บไว้ใน Memory และถูกมองเห็นโดย SPL รูทีนอื่น ๆ ที่ถูกใช้โดยยูสเซอร์เซสชั่น (User Session) เดียวกัน บนฐานข้อมูลเดียวกัน ซึ่งตัวแปรแบบโกลบอล มีคุณสมบัติดังต่อไปนี้

- ต้องการค่าเริ่มต้น (Default Value)
- จะสามารถถูกเรียกใช้โดย SPL รูทีนใด ๆ แม้ว่ามันจะถูกประกาศในรูทีนหนึ่ง ๆ ก็ตาม
- ค่าของมันจะคงอยู่และมีผลต่อรูทีนอื่น ๆ จนกว่าจะจบเซสชั่น

สังเกตฟังก์ชันต่อไปนี้

```
CREATE FUNCTION func1 ()
        RETURNING INT;
        DEFINE GLOBAL gvar INT DEFAULT 2;
        LET gvar=gvar+1;
        RETURN gvar;
END FUNCTION;

CREATE FUNCTION function ()
        RETURNING INT;
        DEFINE GLOBAL gvar INT DEFAULT 5;
        LET gvar=gvar+1;
        RETURN gvar;
END FUNCTION;
```

แม้ว่าจะมีการกำหนดค่า DEFAULT มากกว่า 1 ครั้ง แต่จะมีผลเมื่อฟังก์ชันถูกเรียกใช้ครั้งแรกเท่านั้น ถ้ามีการเอกซีคิวต์ func1 ตามด้วย func2 จะทำให้ได้ผลลัพธ์สุดท้าย gvar = 4 แต่ถ้ามีการเอกซีคิวต์ func2 ก่อน func1 ทำให้ผลลัพธ์สุดท้าย gvar = 7

5.2.3 การกำหนดค่าให้กับตัวแปร

การกำหนดค่าให้กับตัวแปรจะต้องใช้ประโยคคำสั่ง LET ในการกำหนดค่าให้กับตัวแปรที่มีการนิยามไว้แล้ว ถ้าไม่มีการกำหนดค่าให้กับตัวแปร ซึ่งอาจจะผ่านทางอาร์กิวเมนท์ หรือผ่านทางคำสั่ง LET ตัวแปรจะมีค่าที่ไม่ได้กำหนด (undefined value) ซึ่งจะแตกต่างกับค่า NULL ซึ่งถ้าพยายามใช้ตัวแปรที่ไม่ได้กำหนดจะได้รับค่า Error

ผู้ใช้สามารถกำหนดค่าให้กับตัวแปรหลายทาง ดังนี้

- ใช้ประโยคคำสั่ง LET
- ใช้ประโยคคำสั่ง SELECT ... INTO
- ใช้ประโยคคำสั่ง CALL ไปยังโปรซีเดอร์ที่มี RETURNNING clause
- ใช้ประโยคคำสั่ง EXECUTE FUNCTION ... INTO

5.2.3.1 LET statement

สามารถใช้ตัวแปร 1 ตัว หรือมากกว่า เพื่อกำหนดค่าให้กับมัน ดังตัวอย่างต่อไปนี้

```
LET a=5;
LET a,b=10,c+d;
LET a,b =(SELECT cola,colb FROM tab1 WHERE cola=10);
LET d =func1(x,y);
```

ถ้ามีการกำหนดค่าข้อมูลโอเปคให้กับตัวแปร ผู้ใช้สามารถส่งกลับค่าของเอกเทอร์นอลฟังก์ชัน (External function) หรือ SPL ฟังก์ชันอื่นๆ ให้กับตัวแปรได้ นอกจากนี้ยังสามารถกำหนดค่าของชนิดข้อมูลแบบแถว ฟิลด์ของข้อมูลแบบแถวได้เช่นกัน

5.2.3.2 วิธีอื่น ๆ ในการกำหนดค่าให้กับตัวแปร

ผู้ใช้สามารถกำหนดได้โดยการใช้ประโยคคำสั่ง SELECT โดยการเฟตช์ค่าจากฐานข้อมูล และกำหนดค่าให้กับตัวแปรโดยตรง ดังต่อไปนี้

```
SELECT fname,lname INTO a,b FROM customer
    WHERE customer customer_num=101
```

สามารถใช้คำสั่ง CALL และ EXECUTE PROCEDURE และนำค่าที่ส่งกลับมาส่งไปเก็บไว้ในตัวแปร ซึ่งอาจจะส่งกลับ ค่าเดียวหรือมากกว่า 1 ค่าก็ได้ ดังนี้

```
EXECUTE FUNCTION read_address ('Smith')
        INTO p_fname ,p_lname,p_addr,p_city,p_state,p_zip
CALL read_address ('Smith')
        INTO p_fname ,p_lname,p_addr,p_city,p_state,p_zip
```

5.3 การใช้ เคอร์เซอร์

ฟอร์อีชลูป (foreach loop) จะนิยามเคอร์เซอร์ (cursors) ซึ่งเป็นตัวชี้ไปยังไอเท็ม (item) หนึ่ง ๆ ในกลุ่มของแถวหรืออีลีเมนต์ในตัวแปรแบบคอลเลคชัน

ฟอร์อีชลูป จะประกาศและเปิดเคอร์เซอร์ จะทำการเฟตช์แถวหรือชุดของแถวจากฐานข้อมูล ทำงานบนแต่ละไอเท็มในกลุ่ม หลังจากนั้นจะทำการปิดเคอร์เซอร์ ซึ่งผู้ใช้สามารถประกาศเคอร์เซอร์ ถ้าประโยคคำสั่ง SELECT , EXECUTE PROCEDURE หรือ EXECUTE FUNCTION จะมีการส่งกลับค่ามากกว่า 1 แถว ซึ่งรูทีนที่ส่งกลับมาจะเป็นกลุ่มของแถว ที่ถูกเรียกว่าเคอร์เซอร์รูทีน (Cursor Routine)

5.3.1 ฟอร์อีชลูป

จะขึ้นต้นด้วยคีเวิร์ด FOREACH และปิดท้ายด้วย END FOREACH ซึ่งในระหว่าง FOREACH และ END FOREACH จะสามารถประกาศเคอร์เซอร์ หรือใช้ EXECUTE FUNCTION หรือ EXECUTE PROCEDURE ดังตัวอย่างต่อไปนี้

```
FOREACH cursor_name FOR
        SELECT column FROM table INTO variable_name
        ...
        ...
END FOREACH
FOREACH
        EXECUTE FUNCTION func_name() INTO variable_name
END FOREACH
```

ตัวอย่างต่อไปนี้จะเป็นการ SELECT เงินเดือนของพนักงานที่มากกว่า 35,000 บาท แล้วทำการเพิ่มเงินเดือนตามเปอร์เซนต์ที่กำหนด

```
CREATE PROCEDURE increase_by_pct (pct INTEGER)
    DEFINE s INTEGER;
    FOREACH sal_cursor FOR
        SELECT salary INTO s FROM employee
            WHERE salary>35000
        LET s=s+s*(pct/100);
        UPDATE employee SET salary=s
            WHERE CURRENT OF sal_cursor;
    END FOREACH
END PROCEDURE
```

ในฟอร์อีชลูปในตัวอย่างจะมีลำดับการทำงานที่สำคัญดังต่อไปนี้

- ประกาศเคอร์เซอร์
- SELECT salary 1 ค่าในเวลานั้น ๆ จากตาราง Employee
- เพิ่มค่า Salary ตามเปอร์เซนต์ pct
- UPDATE employee ด้วยค่า Salary ใหม่
- เฟตช์ค่า Salary ต่อไปที่ตรงตามเงื่อนไข

ซึ่งประโยค WHERE CURRENT OF ในประโยคคำสั่ง UPDATE จะเป็นการ UPDATE เฉพาะแถว ที่ตำแหน่งปัจจุบันของเคอร์เซอร์เท่านั้น ซึ่งการ UPDATE จะทำการล็อค (Lock) แถวนั้น ไม่ให้ผู้ใช้คนอื่น ๆ สามารถ UPDATE ได้ จนกระทั่งจะทำการ UPDATE เสร็จ

5.4 การใช้งาน IF-THEN-ELSE

การใช้งาน IF จะเหมือนกับภาษาโครงสร้างทั่วไป โดยมีโครงสร้างดังต่อไปนี้

```
IF  เงื่อนไข –1  THEN
    ...
ELIF  เงื่อนไข – 2 THEN
    ...
ELSE
    ...
END IF
```

5.5 การเพิ่ม WHILE และ FOR ลูป

ทั้ง WHILE และ FOR ลูป จะเป็นประโยคคำสั่งที่ทำให้เกิดลูปการทำงานใน SPL รูทีนสำหรับ WHILE LOOP จะเริ่มต้นด้วย WHILE เงื่อนไข , และสิ้นสุดด้วย END WHILE ซึ่งจะทำ SPL สเตจเมนต์เมื่อเงื่อนไขเป็นจริง ดังต่อไปนี้

```
WHILE เงื่อนไข
        สเตจเมนต์
END WHILE
```

สำหรับ FOR ลูปจะเริ่มต้นด้วยคีเวร์ด FOR ตามด้วยชุดคำสั่ง SPL และปิดท้ายด้วย END FOR ตัวอย่างต่อไปนี้จะเป็นกรณีต่าง ๆ ของการใช้ FOR LOOP

```
FOR I=1 TO 10
        ...
END FOR;
FOR i=1 TO 10 STEP 2
        ...
END FOR;
FOR i IN (2,4,8,14,22,32)
        ...
END FOR;
```

5.5.1 การออกจาก LOOP

ในประโยคคำสั่ง FOR FOREACH หรือ WHILE ลูป สามารถใช้คำสั่ง CONTINUE หรือ EXIT ในการควบคุมการทำงานในลูปได้ คำสั่ง CONTINUE จะข้ามการทำงานในลูปนั้นและไปทำรอบต่อไป คำสั่ง EXIT จะเป็นการออกจากลูปและทำงานในคำสั่งแรกที่ตามหลัง END LOOP ซึ่ง EXIT จะต้องตามหลังด้วยประเภทของลูปนั้น เช่น EXIT FOR หรือ EXIT FOREACH ตัวอย่างต่อไปนี้จะเป็นการใช้ CONTINUE และ EXIT ใน FOR ลูป

```
FOR i=1 TO 10
```

```
        IF i=5 THEN
                CONTINUE FOR;
        ...
        ...
        ELIF i=8 THEN
                EXIT FOR
        END IF;
    END FOR;
```

5.6 การส่งกลับจาก SPL ฟังก์ชัน

เมื่อ SPL ฟังก์ชันจะมีการส่งกลับค่าข้อมูล ซึ่งอาจส่งกลับหลายค่าหรือค่าเดียว จำเป็นต้องประกอบด้วย 2 ส่วน คือ

- เขียน RETURNING clause ในประโยคคำสั่ง CREATE PROCEDURE หรือ CREATE FUNCTION ซึ่งจะระบุว่าจะมีการส่งกลับกี่ค่าและมีชนิดข้อมูลอะไร
- ในตัวของรูทีนมีการเขียนประโยคคำสั่ง RETURN เพื่อส่งกลับค่าที่กำหนด

5.6.1 การส่งกลับค่าเดียว

การส่งกลับค่าเดียว แสดงได้ดังตัวอย่างต่อไปนี้

```
    CREATE FUNCTION increase_by_pct (amt DECIMAL , pct DECIMAL)
        RETURNING DECIMAL;
        DEFINE result DECIMAL;
        LET result=amt+amt * (pct/100)
        RETURN result;
    END FUNCTION;
```

5.6.2 การส่งกลับหลายค่า

การส่งกลับหลายค่าแสดงได้ดังตัวอย่างต่อไปนี้

```
    CREATE FUNCTION b_date (num INTEGER)
        RETURNING VARCHAR(30) , DATE;
        DEFINE n CARCHAE(30);
        DEFINE b DATE;
        SELECT name,bdate INTO n,b FROM person
```

```
        WHERE emp_no =num;
    RETURN n,b;
END FUNCTION;
```

5.7 การจัดการคอลเลคชัน

คอลเลคชัน คือกลุ่มของข้อมูลชนิดเดียวกัน เช่น SET MULTISET หรือ LIST ซึ่งข้อมูลแบบคอลเลคชัน สามารถเป็นได้ทั้งคอลัมน์หนึ่งในตาราง และสามารถเป็นชนิดข้อมูลแถวซึ่ง คอลเลคชัน มีอยู่ 2 ประเภท คือ

- ซิมเปิลคอลเลคชัน (Simple collection) คือ SET MULTISET หรือ LIST ที่ใช้เก็บ Built-in Opaque หรือ Distince Type
- เนสเตดคอลเลคชั่น (Nested collection) คือ คอลเลคชันที่เก็บข้อมูลแบบคอลเลคชันชนิดอื่นๆ

ตัวอย่างของชนิดของคอลเลคชันซึ่งอยู่ในตาราง numbers แสดงได้ดังต่อไปนี้

```
CREATE TABLE numbers
(       id INTEGER PRIMARY KEY;
        primes SET (INTEGER NOT NULL),
        evens LIST (INTEGER NOT NULL),
        twin_primes LIST(SET (INTEGER NOT NULL) NOT NULL)
);
```

primes และ evens จะจัดเก็บข้อมูลแบบซิมเปิลคอลเลคชัน และ twin_primes จะจัดเก็บข้อมูลแบบเนสเตดคอลเลคชัน นอกจากนี้คอลเลคชันยังบรรจุชนิดแบบโอเปค เช่นตาราง polycons ซึ่งจะจัดเก็บเซตของโคออดิเนตของรูปหลายเหลี่ยม จัดเก็บชนิดข้อมูลแบบ point ซึ่งเป็นชนิดของโอเปค ซึ่งภายในประกอบด้วยค่า x และ y เช่น '3.0 1.0' เป็นต้น แสดงการสร้างตาราง polygons ดังนี้

```
CREATE OPAQUE TYPE point (INTERNALLENGTH=8);
CREATE TABLE polycons
(       id INTEGER PRIMARY KEY,
        definition SET (point NOT NULL)
);
```

5.7.1 การประกาศตัวแปรแบบคอลเลคชัน

ก่อนที่จะทำการดึงข้อมูลคอลเลคชันจากฐานข้อมูลเข้าไปใน SPL รูทีนจะต้องมีการประกาศตัวแปรแบบคอลเลคชันก่อน ซึ่งสามารถประกาศได้ทั้งแบบ typed หรือ untyped เช่น ถ้าต้องการดึงข้อมูล primes จากตาราง numbers สามารถประกาศตัวแปรได้ 2 ลักษณะ คือ

DEFINE p_coll COLLECTION

DEFINE p_coll SET (INTEGER NOT NULL);

5.7.2 การประกาศอีลีเมนต์ของตัวแปรคอลเลคชัน

หลังจากประกาศชนิดข้อมูลของตัวแปรแล้ว จะมีการประกาศตัวแปรที่ใช้เก็บแต่ละอีลีเมนต์ในคอลเลคชันซึ่งจะต้องประกาศให้ตรงกัน เช่น primes ในตาราง numbers มีอีลีเมนต์เป็น INTEGER เมื่อมีการประกาศตัวแปร p เมื่อเก็บอีลีเมนต์ใน primes จะต้องมีการกำหนดดังนี้

DEFINE p INTEGER

ส่วนในกรณีของ twin_primes ซึ่งเก็บเนสเตดคอลเลคชันควรมีการประกาศตัวแปรดังนี้

DEFINE s SET (INTEGER NOT NULL);

5.7.3 การ SELECT คอลเลคชันใส่ในตัวแปรคอลเลคชัน

จะใช้ประโยค SELECT...INTO เพื่อดึงข้อมูลจากคอลัมน์ที่เป็นคอลเลคชันจากตารางใส่ลงในตัวแปรชนิดคอลเลคชันที่ได้ประกาศเอาไว้ เช่น การดึงข้อมูลจากคอลัมน์ primes เก็บในตัวแปรชื่อ p ทำได้ดังต่อไปนี้

SELECT primes INTO p_coll FROM numbers
WHERE id=220;

5.7.4 การแทรกอีลีเมนต์ในตัวแปรคอลเลคชัน

5.7.4.1 การแทรกใน set และ multiset

การแทรกข้อมูลเข้าไปในตัวแปรแบบ SET และ MULTISET จะใช้ประโยคคำสั่ง INSERT และใช้คำเฉพาะ TABLE ช่วย เช่น

INSERT INTO TABLE (p_coll) VALUES(3);

คำว่า TABLE จะสร้างตัวแปรคอลเลคชันที่เรียกว่า Collection-derived table ซึ่งจะคิดว่า คอลเลคชันเป็นตารางที่มีคอลัมน์เดียว แต่ละอีลีเมนต์ในตัวแปรคอลเลคชันเป็นแต่ละแถวในตาราง ในตัวอย่างที่ผ่านมาจะมอง p_coll เป็นเทเบิล ซึ่งสมมุติว่า p_coll=SET{5,7,31,19,13} จะมองเป็นตาราง ดังนี้

5

7

31

19

13

หลังจาก INSERT ค่า 3 จะได้เป็นดังนี้

5

7

31

19

13

3

เนื่องจาก คอลเลคชันเป็น SET ค่าใหม่ที่แทรกเข้าไป จะไม่มีการระบุตำแหน่งของมัน และหลักการนี้จะใช้ได้ตัวแปรที่เป็น MULTISET ด้วยเช่นกัน

5.7.4.2 การ INSERT ใน LIST

การแทรกข้อมูลเข้าไปในตัวแปรแบบ LIST จะต้องมีการระบุตำแหน่งด้วย เช่น e_coll มีค่าเป็น LIST (2,4,6,8,10) จะมีการแทรกค่า 12 ไปในตำแหน่งที่ 3 ทำได้ดังนี้

INSERT AT 3 INTO TABLE(e_coll) VALUES (12);

จะทำให้ LIST e_coll มีค่า {2,4,12,6,8,10} ตามลำดับ ซึ่งตำแหน่งที่ตามหลัง AT จะเป็นชนิด INTEGER หรือ SMALLINT เท่านั้น ไม่สามารถเป็นตัวอักษร ทศนิยม หรือเป็น Expression ได้

เราสามารถหาจำนวนอีลีเมนต์ในคอลเลคชันแบบ LIST ได้ โดยใช้คำสั่ง cardinality เพื่อประโยชน์ในการแทรกข้อมูลในตำแหน่งสุดท้าย เช่น

SELECT cardinality (evens) FROM numbers INTO n WHERE id=100

5.7.5 การ SELECT แต่ละ Element จาก Collection

ใน SPL รูทีนที่ต้องการเลือกข้อมูลจากตัวแปรคอลเลคชันออกมา จะมีการใช้เคอร์เซอร์เข้ามาช่วยซึ่งจะใช้ประโยคคำสั่ง FOREACH ซึ่งมีรูปแบบดังนี้

FOREACH cursor1 FOR

...

END FOREACH;

5.7.5.1 คอลเลคชันคิวรี

ระหว่าง FOREACH และ END FOREACH จะมีการใส่คำสั่ง SELECT ซึ่งมีลักษณะพิเศษ และมีการใช้งานเฉพาะ เรียกว่า คอลเลคชันคิวรี (Collection Query) ซึ่งจะมีคำเฉพาะ FROM TABLE เข้าช่วย ดังนี้

FOREACH เคอร์เซอร์ FOR

*SELECT * FROM pnt FROM TABLE (vertexs)*

...

END FOREACH;

ซึ่ง vertexes เป็นตัวแปรคอลเลคชันที่เก็บข้อมูลจุดโคออร์ดิเนต ตัวอย่างเช่น vertexes มี แต่ละอีลีเมนต์เป็นดังนี้

'(3.0 1.0)'

'(8.0 1.0)'

'(3.0 4.0)'

'(8.0 4.0)'

ซึ่งในรอบแรกของ FOREACH ในตัวอย่างที่ผ่านมาคอลเลคชันคิวรี จะเลือกอีลีเมนต์แรก ของ vertex เข้าไปเก็บไว้ใน pnt ซึ่งมีค่าเท่ากับ '(3.0 1.0)'

5.7.6 การลบแต่ละ Element ในคอลเลคชัน

การลบอีลีเมนต์ที่ต้องการ ออกจากคอลเลคชัน สามารถทำได้โดยใช้คอลเลคชันคิวรี มีขั้นตอนดังต่อไปนี้

1) ประกาศตัวแปรของคอลเลคชัน และตัวแปรของอีลีเมนต์ในคอลเลคชัน
2) SELECT คอลเลคชันจากฐานข้อมูลเก็บในตัวแปรคอลเลคชัน
3) ประกาศเคอร์เซอร์ที่จะ SELECT แต่ละอีลีเมนต์ในคอลเลคชันในเวลาหนึ่ง ๆ เก็บในตัวแปรอีลีเมนต์ของคอลเลคชัน
4) เขียนลูป หรือกระโดด (Branch) ที่จะเป็นการระบุตำแหน่งของอีลีเมนต์ที่จะลบ

5) ลบอีลีเมนต์ออกจากคอลเลคชัน โดยใช้คำสั่ง DELETE … WHERE CURRENT OF;

ตัวอย่างต่อไปนี้เป็นการลบอีลีเมนต์ในคอลเลคชัน vertexes ที่มีค่า '(3 4)' ซึ่งอยู่ในคอลัมน์ definition บนตาราง Polygons บนฐานข้อมูล ที่มี ID = 207

```
CREATE PROCEDURE shape()
        DEFINE vertexes SET (point NOT NULL);
        DEFINE pnt point;
        SELECT definition INTO vertexes FROM polygons     WHERE id=207;
        FOREACH cursor1 FOR   SELECT * INTO pnt FROM TABLE(vertexes)
            IF pnt = '(3,4)' THEN
                DELETE FROM TABLE(vertexes) WHERE CURRENT OF cursor1;
                EXIT FOREACH;
            ELSE
                CONTINUE FOREACH;
            END IF;
        END FOREACH;
        END PROCEDURE
        ;
```

5.7.6.1 การอัพเดตคอลเลคชันในฐานข้อมูล

การอัพเดตข้อมูลที่เป็นคอลเลคชันจะต้องอัพเดตข้อมูลนั้นด้วยฐานข้อมูลใหม่ โดยจะต้องดึงข้อมูลออกมาเก็บไว้ในตัวแปรก่อน จากนั้นทำการเปลี่ยนแปลงค่าในตัวแปร และทำการอัพเดตในฐานข้อมูลนั้นด้วย ตัวแปรแบบคอลเลคชันใหม่ โดยใช้ประโยคคำสั่ง UPDATE เช่นเดิม

5.7.6.2 การลบอีลีเมนต์ทั้งหมดออกจากคอลเลคชัน

สามารถใช้เพียง SQL สเตจเมนต์เดียว โดยไม่ต้องใช้เคอร์เซอร์ซึ่งการที่จะลบอีลีเมนต์ทั้งหมดออกจากคอลเลคชันจะต้องกระทำงานดังต่อไปนี้

- นิยามตัวแปรคอลเลคชัน
- SELECT ข้อมูลจากฐานข้อมูลเก็บไว้ในตัวแปรคอลเลคชัน
- ใช้ DELETE สเตจเมนต์
- ทำการอัพเดทคอลเลคชันในฐานข้อมูล

ตัวอย่างต่อไปนี้ เป็นการอ่าน คอลัมน์ Definion จากตาราง Polygons ในฐานข้อมูล ที่มี ID = 207 จากนั้นทำการลบทุก ๆ อีลีเมนต์ ในคอลเลคชัน และทำการอัพเดตลงบนฐานข้อมูล ที่ข้อมูลเดิม

```
DEFINE  vertexes SET (INTEGER NOT NULL);
SELECT definition INTO vertexes FROM polygons
        WHERE id=207;
DELETE FROM TABLE(vertexes);
UPDATE polygons SET definition = vertexes
        WHERE id=207;
```

5.7.7 การอัพเดตแต่ละอีลีเมนต์ของคอลเลคชัน

การอัพเดตคอลเลคชันสามารถทำได้โดยอยู่ในคอลเลคชันคิวรี และใช้ประโยคคำสั่ง UPDATE ดังตัวอย่างต่อไปนี้

```
FOREACH cursor1 FOR   SELECT * INTO n FROM TABLE(s)
        IF (n==500) THEN
                UPDATE TABLE(s)(x) SET x=400  WHERE CURRENT OF cursor1;
                EXIT FOREACH;
        END IF;
END FOREACH;
```

5.7.8 การอัพเดตอีลีเมนต์ทั้งหมดในคอลเลคชัน

ถ้าต้องการอัพเดตให้ทุก ๆ อีลีเมนต์ในคอลเลคชันเป็นค่าเดียวกันหมดทุก ๆ อีลีเมนต์จะใช้เพียงคำสั่งเดียว โดยไม่ต้องใช้เคอร์เซอร์ กระทำได้โดยการดึงข้อมูลคอลเลคชันจากฐานข้อมูล เก็บไว้ในตัวแปรคอลเลคชัน และทำการอัพเดตข้อมูลในตัวแปรคอลเลคชัน ดังตัวอย่างต่อไปนี้

```
UPDATE TABLE(s)(x) SET x=0
```

จากตัวอย่างจะทำการอัพเดตให้ทุก ๆ อีลีเมนต์ใน s มีค่าเป็น 0

บทที่ 6
การใช้งานทริกเกอร์

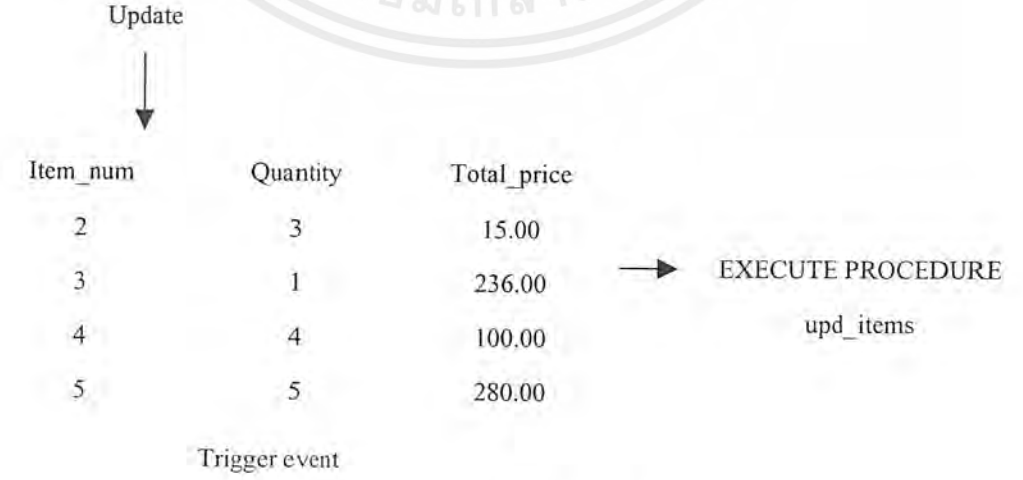
ทริกเกอร์เป็นกลไกที่ฝังอยู่ในฐานข้อมูล ซึ่งผู้ที่มีสิทธิ์ใช้งาน (Permission) จะสามารถใช้งานได้ โดยทริกเกอร์จะตรวจสอบเหตุการณ์ (Event) ที่เกิดขึ้น คือการ INSERT DELETE หรือ UPDATE บนตารางใดๆ ทริกเกอร์จะทำให้เกิดแอคชัน (Action) ใดๆ ขึ้นโดยอัตโนมัติ ซึ่งแอคชันอาจจะเป็นการ INSERT, DELETE , UPDATE , EXECUTE PROCEDURE หรือ EXECUTE FUNCTION ก็ได้

ผู้ใช้สามารถใช้ทริกเกอร์เพื่อวัตถุประสงค์ดังต่อไปนี้

1. สร้างออดิตเทรล (Audit Trail) ของกิจกรรมต่าง ๆ บนฐานข้อมูล เช่น เราสามารถติดตามการอัพเดตบนตารางสั่งของ (orders) และคอยอัพเดตข้อมูลต่าง ๆ ที่มีความสัมพันธ์กัน
2. สร้างกฎของธุรกิจ (Business Rule) เช่น เมื่อจำนวนที่สั่งเกินกว่าเครดิตของลูกค้า จะแสดงข้อความแจ้งให้ทราบโดยอัตโนมัติ
3. หาข้อมูลที่ไม่มีในตารางหรือในฐานข้อมูล เช่น เมื่อมีการอัพเดตจำนวนสินค้า จะทำการคำนวณหาค่าบนคอลัมน์ Total_price อัตโนมัติ
4. บังคับให้เป็นไปตาม referential Integrity เช่น เมื่อทำการลบลูกค้ารายหนึ่ง ๆ สามารถใช้ทริกเกอร์ในการลบแถวที่เกี่ยวข้องกับลูกค้ารายนั้นออกไป

6.1 การสร้างทริกเกอร์

เราสามารถใช้ประโยคคำสั่ง CREATE TRIGGER ในการสร้างทริกเกอร์ซึ่งจะเป็นดาต้าเดฟินิชันสเตจเมนต์ (Data Definition Statement) ซึ่งเมื่อเกิดเหตุการณ์ต่าง ๆ บนตาราง ที่ถูกบังคับโดยทริกเกอร์จะทำให้ฐานข้อมูลทำแอคชันที่กำหนด ซึ่งรูปที่ 6.1 แสดงความสัมพันธ์ทริกเกอร์อีเวนต์ (Trigger event) และ ทริกเกอร์แอคชัน (Triggered Action)



รูปที่ 6.1 กระบวนการของทริกเกอร์

ซึ่งประโยคคำสั่ง CREATE TRIGGER จะประกอบด้วยประโยคคำสั่งที่มีหน้าที่ต่าง ๆ ดังต่อไปนี้

- การกำหนดชื่อของทริกเกอร์
- การระบุทริกเกอร์อีเวนต์
- การนิยาม SQL แอคชันที่จะถูกทริกเกอร์

6.1.1 การกำหนดชื่อของทริกเกอร์

ชื่อของทริกเกอร์จะเป็นตัวกำหนดทริกเกอร์ ซึ่งชื่อทริกเกอร์จะยาวที่สุด 18 ตัวอักษร และประกอบด้วยตัวอักษร , ตัวเลข 0 ถึง 9 และขีดล่าง ซึ่งตัวอย่างต่อไปนี้แสดงการสร้างทริกเกอร์ชื่อ ugqty

CREATE TRIGGER upqty

6.1.2 การระบุทริกเกอร์อีเวนต์

ทริกเกอร์อีเวนต์คือชนิดของสเตจเมนต์ที่จะทำให้ทริกเกอร์ทำงาน เมื่อสเตจเมนต์นี้ถูกกระทำบนตาราง ดาต้าเบสเซิร์ฟเวอร์จะทำการเอกซีคิวต์ SQL สเตจเมนต์ที่จะก่อให้เกิดทริกเกอร์แอคชัน ซึ่งทริกเกอร์อีเวนต์สามารถเป็นได้ทั้ง INSERT , DELETE หรือ UPDATE

ตัวอย่างต่อไปนี้จะเป็นการสร้างทริกเกอร์ซึ่งเป็นการอัพเดตคอลัมน์ Quantity บนตาราง Items

CREATE TRIGGER upqty
UPDATE OF quantity ON items

ตัวอย่างต่อไปนี้ จะเป็นการสร้างทริกเกอร์ที่ตรวจสอบการแทรกข้อมูลบนตาราง Items

CREATE TRIGGER ins_qty
INSERT ON items

6.1.3 การนิยามทริกเกอร์แอคชัน

ทริกเกอร์แอคชันคือ SQL สเตจเมนต์ที่จะถูกกระทำเมื่อเกิดทริกเกอร์อีเวนต์ขึ้น ซึ่งประกอบด้วย INSERT , DELETE , UPDATE , EXECUTE PROCEDURE หรือ EXECUTE FUNCTION สเตจเมนต์จะมีทางเลือกในการกระทำดังต่อไปนี้

- ก่อนที่ทริกเกอร์อีเวนต์จะถูกเอ็กซีคิวต์
- หลังที่ทริกเกอร์อีเวนต์จะถูกเอ็กซีคิวต์
- แต่ละแถวที่มีผลกระทบต่อทริกเกอริงสเตจเมนต์ (Triggering Statement)

ซึ่งในแต่ละกรณี จะมีการใช้คำเฉพาะ BEFORE , AFTER และ FOR EACH ROW ตามลำดับ ตัวอย่างต่อไปนี้ จะเป็นทริกเกอร์แอคชันที่จะไปเอ็กซีคิวต์ SPL รูทีนชื่อ upd_item_p1 ซึ่งเป็นโปรซีเดอร์ จะถูกเอ็กซีคิวต์ก่อนที่ทริกเกอร์อีเวนต์จะถูกกระทำ

BEFORE (EXECUTE PROCEDUTE upd_items_p1)

6.1.4 สเตจเมนต์ CREATE TRIGGER แบบสมบูรณ์

ประโยคคำสั่ง CREATE แบบสมบูรณ์ จะเป็นการรวมชื่อทริกเกอร์ (Trigger_name), ทริกเกอร์อีเวนต์ และทริกเกอร์แอคชัน เข้าด้วยกัน ดังตัวอย่างต่อไปนี้

CREATE TRIGGER updqty
UPDATE OF quantity ON items
BEFORE (EXECUTE PROCEDURE upd_items_p1)

ซึ่งหมายความว่า เมื่อมีการอัพเดตคอลัมน์ Quantity บนตาราง Items ก่อนที่จะถูกอัพเดตจะมีการเอ็กซิคิวฟังก์ชัน upd_items_p1 ซึ่งถ้าโปรซีเดอร์ upd_item_p1 ไม่มีในดาต้าเบสเซิร์ฟเวอร์ขบวนการสร้างทริกเกอร์จะส่งความผิดพลาดกลับมา

6.2 การใช้ทริกเกอร์แอคชัน

6.2.1 การใช้ BEFORE และ AFTER ทริกเกอร์แอคชัน

BEFORE ทริกเกอร์แอคชัน จะเอ็กซีคิวต์ก่อน Triggering สเตจเมนต์ นั่นคือ ก่อนที่จะเกิดทริกเกอร์อีเวนต์ขึ้น ส่วน AFTER ทริกเกอร์แอคชันจะถูกเอ็กซีคิวต์หลังจากที่แอคชันของทริกเกอริ่งสเตจเมนต์เสร็จเรียบร้อยแล้ว

ตัวอย่างเช่น ก่อนที่จะมีการอัพเดตคอลัมน์ Quality ในตาราง Items สามารถเรียก (Call) ไปยัง SPL รูทีนชื่อ upd_items_p1 ซึ่งจะทำหน้าที่ในการคำนวณหาจำนวนของ Order สำหรับทุก ๆ items ในตารางรูทีนจะจัดเก็บจำนวนรวมไว้เป็นตัวแปรแบบโกลบอลชื่อ old_gty ดังต่อไปนี้

```
CREATE PROCEDURE upd_items_p1()
        DEFINE GLOBAL old_gty INT DEFAULT 0;
        LET old_gty=(SELECT SUM(quantity) FROM items);
END PROCEDURE;
```

หลังจากมีการอัพเดต quantity เรียบร้อยแล้ว เราสามารถคำนวณจำนวนรวมอีกครั้งหนึ่ง เพื่อหาว่ามีการเปลี่ยนแปลงไปเท่าใด SPL ROUTINE ต่อไปนี้ ชื่อ upd_items_p2 จะคำนวณ quantity รวม อีก

ครั้งหนึ่งและมีการเก็บผลลัพธ์ไว้ในตัวแปรแบบโลคอลชื่อ new_qty หลังจากนั้นก็จะทำการเปรียบเทียบ new_qty กับ old_qty และดูว่า ถ้า quantity รวมสำหรับทุก ๆ order เพิ่มมากกว่า 50 เปอร์เซนต์หรือไม่ ถ้าเป็นเช่นนั้น จะมีการแสดงความผิดพลาดออกมา (RAISE EXCEPTION)

```
CREATE PROCEDURE upd_items_p2()
        DEFINE GLOBAL old_qty INT DEFAULT 0;
        DEFINE new_qty INT;
        LET new_qty=(SELECT SUM(quantity) FROM ITEMS);
        IF new_qty > old_qty*1.50 THEN
                RIASE EXCEPTION , -746 , 0 , 'Not Allow –rule violation';
        END IF;
END PROCEDURE;
```

ซึ่งการนำแอคชันรวมกันจะได้ทริกเกอร์ up_items ซึ่งจะป้องกันการอัพเดตค่าที่ไม่ถูกต้องลงไปบนคอลัมน์ quantity บนตาราง items

```
CREATE TRIGGER up_items
UPDATE OF quantity ON items
BEFORE (EXECUTE PROCEDURE upd_items_p1())
AFTER (EXECUTE PROCEDURE upd_items_p2());
```

เมื่อทริกเกอร์ได้รับความผิดพลาดกลับมาอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์และฐานข้อมูลจะมีล็อกกิ้ง (Logging) และดาต้าเบสเซิร์ฟเวอร์จะโรลแบ็กทั้งแอคชันของทริกเกอร์อีเวนต์ และทริกเกอร์แอคชัน

6.2.2 การใช้ FOREACH EACH ROW ทริกเกอร์แอคชัน

FOREACH ROW ทริกเกอร์แอคชันจะทำการเอ็กซีคิวต์แต่ละแถวที่มีผลจากทริกเกอริ่งสเตจเมนต์ เช่น FOREACH ROW ทริกเกอร์แอคชันที่เอ็กซีคิวต์แต่ละแถวบนตาราง Items ที่มี manu_code มีค่าเป็น 'KAR'

```
UPDATE items SET quantity = quantity * 2 WHERE manu_code='KAR'
```

ถ้าทริกเกอริงสเตจเมนต์ไม่มีการเอ็กซีคิวต์แถวใด ๆ FOREACH ROW ทริกเกอร์แอคชันจะไม่มีการเอ็กซีคิวต์

6.2.2.1 การใช้ประโยคคำสั่ง REFERENCING

เมื่อมีการสร้าง FOREACH ROW ทริกเกอร์แอคชัน เราสามารถมีสเตจเมนต์ที่อ้างถึงค่าของคอลัมน์ก่อนและหลังการเกิดทริกเกอริ่งสเตจเมนต์โดยใช้ประโยคคำสั่ง REFERENCING ซึ่งจะมีการสร้างพรีฟิก (Prefix) 2 ตัว ในการอ้างถึงชื่อของคอลัมน์ คือ ตัวที่อ้างถึงค่าเดิม และค่าใหม่ ซึ่งพรีฟิกนี้ถูกเรียกว่าโครีเลชัน (Correlation) ซึ่งสามารถสร้าง แต่ละพรีฟิกหรือทั้งสองตัวก็ได้ โดยการใช้คำเฉพาะ OLD และ NEW เช่น pre_upd อ้างถึงค่าเดิมและ post_upd อ้างถึงค่าใหม่ สามารถเขียนได้ดังนี้

REFERENCING OLD AS pre_upd NEW AS post_upd

ทริกเกอร์แอคชันต่อไปนี้ จะสร้างแถวในล็อคเรคอร์ด (Log record) เมื่อ guantity ถูกอัพเดตใส่ตาราง items INSERT สเตจเมนต์จะอ้างถึงค่าเดิมของคอลัมน์ item_num และ order_num และค่าเดิมและค่าใหม่ของ quantity

```
FOREACH ROW (    INSERT log_record
                 VALUES (pre_upd.item_num,pre_upd.order_num,USER,
                         CURRENT,pre_upd.quantity,
                         post_upd.quantity));
```

ซึ่งรายละเอียดของตาราง log_record คือ

```
CREATE TABLE log_record
    (    item_num      SMALLINT,
         ord_num       INTEGER,
         username      CHARACTER(8),
         update_time   DATETIME YEAR TO MONTH,
         old_qty       SMALLINT,
         new_qty       SMALLINT);
```

6.2.2.2 การใช้เงื่อนไข WHEN

เราสามารถกำหนดการทำทริกเกอร์แอคชันโดยใช้ประโยคคำสั่ง When ถ้าเงื่อนไขเป็นจริง ทริกเกอร์แอคชันจะถูกเอ็กซีคิวต์ ถ้าเป็นเท็จหรือเป็นทริกเกอร์แอคชันที่ไม่รู้จัก (unknown Trigger Action) จะไม่ถูกเอ็กซีคิวต์ ถ้ามีการใช้ประโยคคำสั่ง FOREACH ROW เงื่อนไขนี้จะมีผลกับทุก ๆ แถว

ตัวอย่างต่อไปนี้ ทริกเกอร์แอคชันจะถูกเอ็กซิคิวก็ต่อเมื่อ ราคาต่อหน่วยค่าเดิมมีค่ามากกว่า 2 เท่าของราคาต่อหน่วยค่าใหม่

```
CREATE TRIGGER up_price
UPDATE OF unit_price ON stack
REFERENCING OLD AS pre NEW AS post
FOREACH ROW WHEN (post.unit_price > pre.unit_price*2)
        (INSERT INTO warn_tab VALUES (pre.stock_num,pre.order_num,
                pre.unit_price,post.unit_price,CURRENT));
```

6.2.3 การใช้ SPL รูทีนเป็นทริกเกอร์แอคชัน

เมื่อมีการใช้ SPL รูทีน เป็นทริกเกอร์แอคชัน เราสามารถส่งผ่านข้อมูลจากตารางทริกเกอริ่ง (Triggering Table) ไปให้กับ SPL รูทีนได้

การส่งผ่านค่าข้อมูลไปยัง SPL รูทีนสามารถทำได้โดยการการใส่ค่าข้อมูลเข้าไปในลิสต์ของอาร์กิวเมนต์ (Argument List) ของ EXECUTE PROCEDURE หรือ EXECUTE FUNCTION สเตจเมนต์ ตัวอย่างต่อไปนี้ จะเป็นการใช้ฟังก์ชันเป็นทริกเกอร์แอคชันโดยการส่งผ่านข้อมูล Quantity และ Total_price ของตาราง items ไปให้กับ SPL รูทีนชื่อ calc_totpr

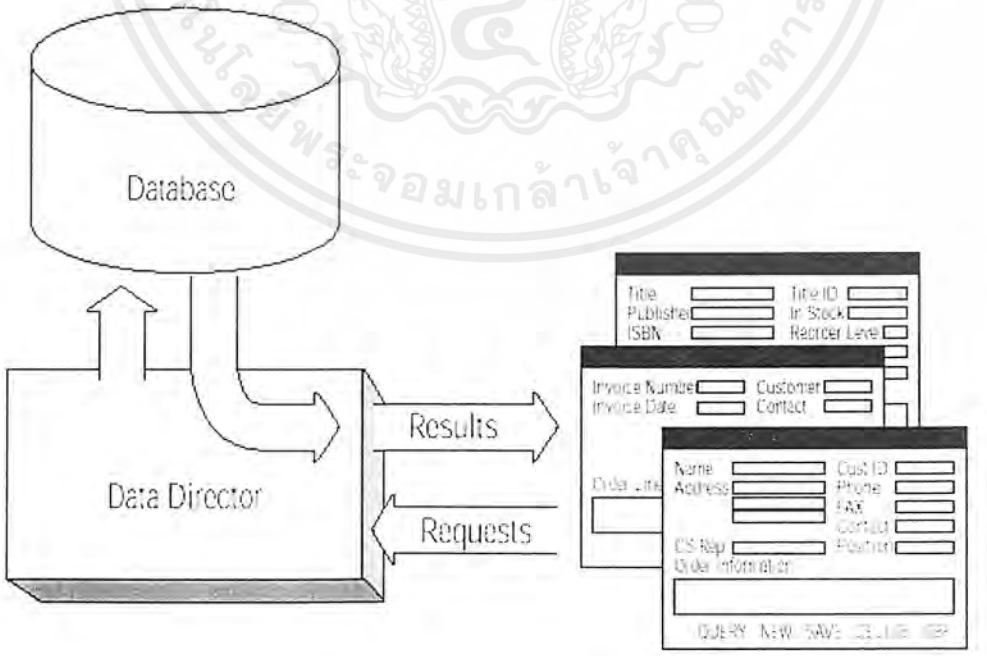
```
CREATE TRIGGER upd_totpr
UPDATE OF quantity ON items
REFERENCING OLD AS pre_upd  NEW AS post_upd
FOR EACH ROW (EXECUTE FUNCTION calc_totpr(pre_upd.quantity,
                Post_upd.quantity,pre_upd.total_price) INTO total_price)
```

บทที่ 7
ดาต้าไดเรกเตอร์

ดาต้าไดเรกเตอร์ (Data Director) เป็นเครื่องมือที่เพิ่มหน้าที่ทางฐานข้อมูลเข้าไปในส่วนที่ใช้พัฒนาโปรแกรม ทำให้ผู้พัฒนาโปรแกรมแอพพลิเคชันเขียนโปรแกรมน้อยลง ซึ่งคุณสมบัติของดาต้าไดเรกเตอร์มีดังนี้

1. ง่ายต่อการเชื่อมต่อระหว่างโครงสร้างข้อมูลในฐานข้อมูลกับส่วนพัฒนาแอพพลิเคชัน
2. ทำหน้าที่เกี่ยวกับการจัดการข้อมูล (Data-Management) ในการทำ Visual Access โดยอัตโนมัติ เช่น master-detail coordination ทำให้ประหยัดเวลาในการเขียนโปรแกรมและมีประสิทธิภาพ
3. สามารถใช้งานกับอินฟอร์มิกส์ยูนิเวอร์แซลเซิร์ฟเวอร์ที่มียูนิเวอร์แซลดาต้าออปชันได้

การเข้าถึงข้อมูลจากการคิวรี , การจัดการการเชื่อมต่อกับฐานข้อมูลและเคอร์เซอร์ต่างๆ เตรียมการจัดการล็อกอิน (log in) และความปลอดภัยของการล็อคออน (logon security) และการจัดเรียงข้อมูล ดาต้าไดเรกเตอร์เป็นตัวจัดการให้โดยอัตโนมัติ ดาต้าไดเรกเตอร์ยังช่วยในการจัดการข้อมูลได้ง่ายขึ้นจากการทำงานที่เป็นพื้นฐาน เช่น บันทึก , อัพเดต และเนวิเกชั่น (Navigation) นอกจากนั้นยังเตรียมการทำงานเกี่ยวกับมาสเตอร์ดีเทล (master-detail) และแวลูซิงโคไนเซชั่น (value synchronization) ระหว่างหลายๆ แอพพลิเคชันบนวินโดวส์ การจัดการเกี่ยวกับคอนเคอเรนซี (concurrency) และทรานแซคชั่น (transaction) , ดาต้าแคชชิ่ง (data caching) และใช้งานบุคมาร์ค (Bookmarks) ได้



รูปที่ 7.1 แสดงดาต้าไดเรกเตอร์ที่ทำงาน Data access และ Data management

การจัดการข้อมูลของดาต้าไดเรกเตอร์ไม่ได้เพิ่มภาระการทำงาน (Overhead) หรือลดประสิทธิภาพและความเร็วของแอพพลิเคชันเลย ดังรูปที่ 7.1 เป็นรูปแสดงการทำงานของดาต้าไดเรกเตอร์

นอกจากนั้นดาต้าไดเรกเตอร์ยังเก็บโครงสร้างข้อมูลภายในฐานข้อมูลด้วย ดังนั้นขณะเราเขียนแอพพลิเคชัน เราสามารถอ้างอิงข้อมูลจากดาต้าไดเรกเตอร์ได้ทันที โดยดาต้าไดเรกเตอร์ไม่ได้ทำการเปลี่ยนแปลงโครงสร้างข้อมูลในฐานข้อมูลเลย ที่จะกล่าวต่อไปเป็นการอธิบายว่าดาต้าไดเรกเตอร์จะทำงานอย่างไร เมื่อผู้ใช้เรียกใช้งานแอพพลิเคชัน

- ลำดับการทำงาน

เมื่อเริ่มต้นทำงานโปรแกรมดาต้าไดเรกเตอร์อ่านไฟล์ที่ต้องใช้งานโปรเจ็คก่อน หลังจากนั้นจึงทำการเชื่อมต่อกับฐานข้อมูลแล้วมันจะสร้างคำสั่ง SQL ที่เหมาะสมส่งไปเรียกข้อมูลจากฐานข้อมูลจากนั้นมันจะสร้าง SQL เพื่อส่งไปยังฐานข้อมูลอีกครั้งก็ต่อเมื่อต้องการตอบสนองการใช้งานของผู้ใช้ เช่น การเปลี่ยนแปลงและการเพิ่มข้อมูล

- ดาต้าแคชชิ่ง

ดาต้าไดเรกเตอร์จะเรียกข้อมูลที่ต้องการแสดงผลตามหลักการ โดยผู้ใช้ต้องการข้อมูลมันจะเอามาจากบัฟเฟอร์ (Buffer) ภายในของดาต้าไดเรกเตอร์บนฝั่งไคลเอนท์จนกว่าผู้ใช้ต้องการดูและเปลี่ยนแปลงข้อมูลดาต้าไดเรกเตอร์สามารถทำให้แสดงผลข้อมูลตัวเดียวกันบนการควบคุมจากหลายๆทางแต่คิวรีบนข้อมูลตัวเดียวกัน และยังสามารถแสดงผลข้อมูลตัวเดียวกันบนแอพพลิเคชันวินโดว์สหลายๆประเภทได้

- ดาต้าซิงโครไนเซชัน

เมื่อมีการเปลี่ยนแปลงข้อมูลที่แสดงอยู่ในหลายๆ คอนโทรล (Control) นั้นดาต้าไดเรกเตอร์จะเป็นตัวทำการซิงโครไนซ์ข้อมูลบนทุกๆ คอนโทรลทุกแอพพลิเคชัน เช่น ถ้าเราเปลี่ยนชื่อลูกค้า มันจะเปลี่ยนชื่อลูกค้าคนนั้นในทุกๆคอนโทรลที่ใช้แสดงผล

- ดาต้าลุคอัพ

ดาต้าลุคอัพ คือ ข้อมูลที่แสดงผลบนคอมโบบ๊อกซ์ (Combo Box) เช่นรายการของชื่อ ข้อมูลที่จะใช้เป็นลุคอัพ เรียกว่ากลุ่มอ้างอิง (*reference group)*

ตัวอย่างการแสดงดาต้าลุคอัพในแอพพลิเคชันบริการลูกค้า (customer service application) มีตาราง customer เป็น foreign key ของตาราง reps ซึ่งเก็บรายชื่อผู้แทนขายแต่ละคนไว้โดยความสัมพันธ์ระหว่าง 2 ตารางนี้คือ m:n (1 ผู้แทนขายสามารถบริการลูกค้าได้หลายคน) ในแอพพลิเคชันจะแสดงชื่อของลูกค้าและที่อยู่ในเท็กซ์บ๊อกซ์ (Text Box) และชื่อของผู้แทนขายในคอมโบบ๊อกซ์ และให้ผู้ใช้แสดงผู้แทนขายเมื่อต้องการเพิ่มลูกค้า

ในขณะออกแบบเราใช้ดาต้าไดเรกเตอร์ ส่วนกราฟิกอินเทอร์เฟส (graphic interface) ในการเชื่อมระหว่างตาราง reps และคอมโบบ๊อกซ์ ในขณะทำงานแอพพลิเคชัน เมื่อใดที่เรคคอร์ดของลูกค้าถูกเลือก ดาต้าไดเรกเตอร์จะทำการเรียก (Look Up) ชื่อของผู้แทนขายที่สัมพันธ์กันในตาราง reps โดยอัตโนมัติ และแสดงผลในคอมโบบ๊อกซ์

7.1 แนวความคิดของดาต้าไดเรกเตอร์

7.1.1 ดาต้าโมเดลลิ่ง

โมเดล (Model) คือ มุมมองหรือภาพของโครงสร้างระดับฟิสิคอล (Physical) ในฐานข้อมูลซึ่งเป็นตาราง , คอลัมน์ , และความเกี่ยวดอง(Relationship) ตัวโมเดลนี้จะบอกดาต้าไดเรกเตอร์เกี่ยวกับโครงสร้างข้อมูลที่จะใช้ในแอพพลิเคชันว่าจะเข้าใช้งานอย่างไรและง่ายต่อการใช้ออกแบบและทำงาน เราสร้างโมเดลได้โดยการเรียกใช้อิมพอร์ตโมเดล (Import model) ของดาต้าไดเรกเตอร์

โมเดลนี้จะเก็บข้อมูลเกี่ยวกับส่วนประกอบในฐานข้อมูลดังนี้

- ชื่อตาราง
- ชื่อคอลัมน์
- ความเกี่ยวดอง
- ผู้ที่สร้างตารางนั้นๆ
- ขนาดของคอลัมน์
- ชนิดของข้อมูลของคอลัมน์นั้นๆ
- แอตทริบิวต์ที่เป็นคีย์หลัก (Primary Key)
- คีย์ภายนอก (Foreign keys)

ดาต้าไดเรกเตอร์จะใช้ข้อมูลเหล่านี้ในการออกแบบและทำงานโปรแกรม ในระหว่างออกแบบดาต้าไดเรกเตอร์จะใช้โมเดลไฟล์ (Model File) เพื่ออ่านข้อมูลเกี่ยวกับตารางและคอลัมน์ต่างๆ ในระหว่างทำงานโปรแกรมจะใช้โมเดลไฟล์เพื่อตัดสินใจในการสร้าง SQL สเตจเมนต์ที่เหมาะสมที่เหมาะสม

7.1.2 การบริหารข้อมูลในโมเดล

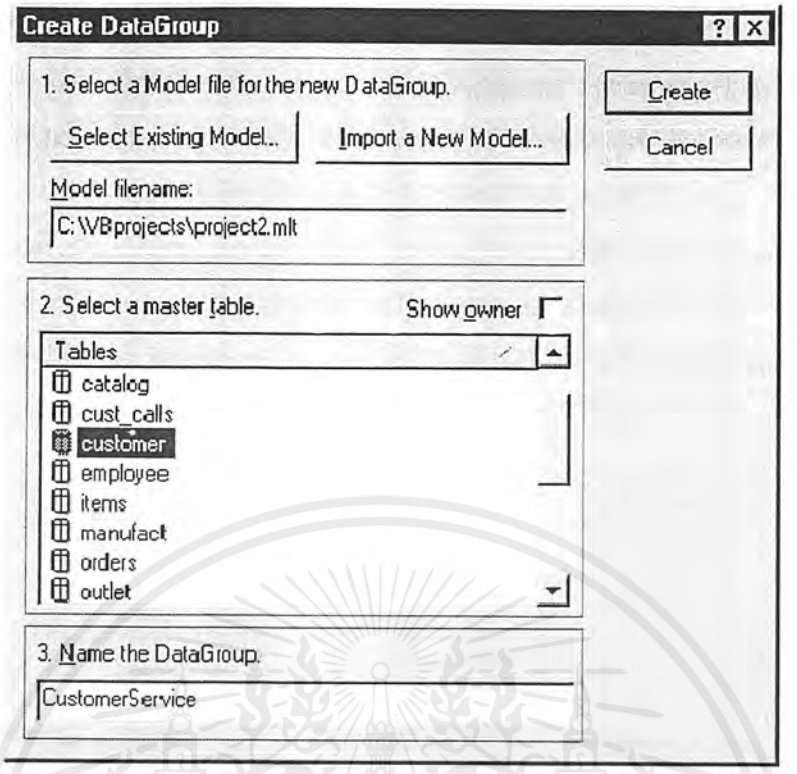
ก่อนอื่นเราต้องกล่าวถึงดาต้ากรุ๊ป (DataGroup) ก่อน แต่ละแอพพลิเคชันใช้ข้อมูลไม่เหมือนกันทั้งหมด บางแอพพลิเคชันจะมีหน้าที่ต่างกันโดยสิ้นเชิง ดังนั้นดาต้าไดเรกเตอร์มีการสร้างดาต้ากรุ๊ปเพื่อเลือกเอาตารางและคอลัมน์ที่ต้องการใช้ในแอพพลิเคชันต่างๆ

การสร้างดาต้ากรุ๊ปเราสร้างโดยใช้ดาต้าลิงค์เมเนเจอร์ (DataLink Manager) ของดาต้าไดเรกเตอร์ ซึ่งระหว่างการสร้างจะต้องมีการกำหนดมาสเตอร์เทเบิล (Master table) ซึ่งเลือกให้เป็นตารางหลักที่ใช้ในการเริ่มต้นหาทางไปยังตารางอื่นๆที่ใช้ในแอพพลิเคชัน

ดาต้ากรุ๊ปจะเก็บข้อมูลอื่นๆของแต่ละแอพพลิเคชันด้วย เช่น การเรียงข้อมูลในตารางและเงื่อนไขในคิวรี , คอนเคเรนซีคอนโทรลที่กำหนด , การจัดการกับข้อผิดพลาด (Error-handling) และอื่นๆอีกมาก

7.1.3 การเชื่อมต่อข้อมูลที่ต้องการควบคุม

ดาต้าลิงค์ คือตัวเชื่อมต่อข้อมูลของฐานข้อมูลคอลัมน์กับฟอร์มของแต่ละแอพพลิเคชัน และเป็นตัวบอกดาต้าไดเรกเตอร์ว่าควรแสดงผลอย่างไร ในดาต้าลิงค์จะเก็บข้อมูลให้ดาต้าไดเรกเตอร์ใช้เมื่อต้องการ คิวรีข้อมูล เช่น ค่าเริ่มต้น และ คุณสมบัติพิเศษต่างๆที่ใช้ในการเชื่อมต่อคอนโทรลอื่นๆ



รูปที่ 7.2 แสดงไดอะล็อกบ็อกซ์ที่ใช้สร้างดาต้ากรุ๊ป

7.1.4 ความสัมพันธ์ระหว่าง ดาต้าลิงค์ , ดาต้ากรุ๊ป และโมเดล

ดาต้าลิงค์จะเป็นตัวบริหารดาต้ากรุ๊ปและตัดสินใจว่าตารางไหนที่ดาต้าไดเรกเตอร์สมควรคิวรีในขณะรันไทม์ (Runtime) เมื่อทำการคิวรี ตัวดาต้ากรุ๊ปจะบอกดาต้าไดเรกเตอร์ว่า คิวรี ควรเริ่มต้นจากมาสเตอร์เทเบิลไหนและดาต้าลิงค์ในดาต้ากรุ๊ปจะบอกดาต้าไดเรกเตอร์ว่าข้อมูลอะไรที่จะคิวรีและแสดงผลอะไร

ดาต้ากรุ๊ปจะอ้างอิงถึงโมเดลอีกทีทำให้ดาต้าไดเรกเตอร์รู้ถึงความเกี่ยวดองได้อัตโนมัติ เช่น one-to-many ตัวโมเดลจะบอกดาต้าไดเรกเตอร์อัตโนมัติว่าให้คิวรีข้อมูลรายละเอียดของมาสเตอร์เรคอร์ด ถ้าดาต้าลิงค์บอกว่าเป็นดีเทลเรคอร์ด (Detail record)

7.1.5 ดาต้าแอคเซส

เป็นคุณสมบัติที่แข่งขันกันในการออกแบบและพัฒนาไคลเอนท์/เซิร์ฟเวอร์แอพพลิเคชัน (client/server application) ทางฐานข้อมูล ซึ่งฐานข้อมูลที่เป็นฐานข้อมูลท้องถิ่น (Local Database) หรือ Coorperate-wide SQL Database (เป็นฐานข้อมูลที่มี Front-end Client) ดังนั้นดาต้าแอคเซส (Data Access) จึงเป็นตัวแลกเปลี่ยนข้อมูลระหว่างฟรอนต์-เอนด์ไคลเอนต์ (Front-end Client) และแบคเอนด์เซิร์ฟเวอร์ (Back-end Server) อัตโนมัติด้วยดาต้าไดเรกเตอร์

7.1.6 การตรวจสอบข้อมูล (Data Validation)

เมื่อเราเรียกใช้แอพพลิเคชันแล้ว ดาต้าไดเรกเตอร์จะเป็นตัวตรวจสอบว่าค่าของข้อมูลที่เพิ่มเข้าไปในคอลัมน์ที่เราต้องการและคีย์หลักเป็นตัวแสดงในการเปลี่ยนแปลงหรือลบเรคคอร์ด

7.2 ดาต้าไดเรกเตอร์โปรแกรมเมติกคอลลี (Data Director Programmatically)

ดาต้าไดเรกเตอร์มีรูปแบบการใช้งานอยู่ 2 ส่วน คือส่วน GUI (Graphic User Interface) และโปรแกรมเมติกคอลลี (Programmatically : DDO) ซึ่งการนำไปใช้ขึ้นอยู่กับความเหมาะสมของแอพพลิเคชั่นโดย GUI จะเหมาะกับแอพพลิเคชันที่ไม่ซับซ้อนและมีรูปแบบตายตัว การเลือกใช้รูปแบบ โปรแกรมเมติกคอลลี มีเหตุผลหลายประการดังนี้

- เพิ่มการทำงานที่ซับซ้อนมากขึ้น ปกติ GUI จะใช้ในการพัฒนาต้นแบบขึ้นมาแล้วจะใช้ส่วนโปรแกรมเมติกคอลลีเพิ่มหน้าที่การทำงานใหม่ๆ ขึ้นมา
- ประสิทธิภาพ การออกแบบโดยการแดร็ก-ดร็อป (Drag-Drop) ในส่วน GUI ลงบนฟอร์มนั้น เมื่อผู้ใช้เรียกใช้งานโปรแกรมมันจะถูกดึงข้อมูลมาโดยอัตโนมัติซึ่งในขณะนั้นอาจจะยังไม่จำเป็นที่จะต้องดึงข้อมูลมา ถ้าใช้โปรแกรมเมติกคอลลี เราสามารถควบคุมการดึงข้อมูลในเวลาที่เหมาะสมได้
- เพื่อใช้งานร่วมกับ C++ หรือ J++
- ความถนัดของผู้พัฒนาซึ่งถนัดเขียนโปรแกรมโดยตรงมากกว่า
- ใช้งานร่วมกับชนิดของข้อมูลแบบใหม่ๆ ที่ไม่ต้องการเพียงแค่ข้อมูลตัวอักษร เช่น ชนิดของโอเปคที่เก็บพวกเสียง , วิดีโอ , และสีต่างๆ ซึ่งเราต้องการความหลากหลายเหล่านี้ในแอพพลิเคชั่น
- ความยืดหยุ่นในขณะโปรแกรมทำงาน เช่นการเปลี่ยนแปลงโมเดลต่างๆ (เช่นการเพิ่มตารางใหม่ หรือ virtual table)

7.2.1 ลำดับขั้นของออบเจ็กต์

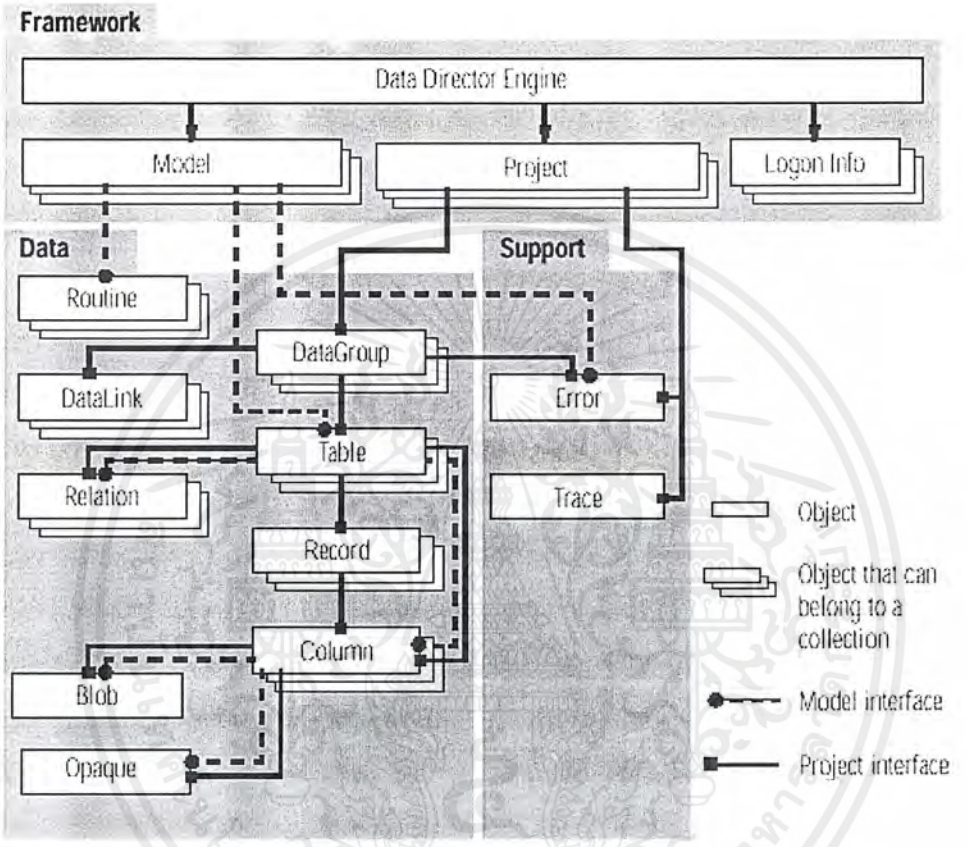
ก่อนที่เราจะสามารถเขียนโปรแกรมด้วยดีดีโอ (DDO) ได้ เราต้องเข้าใช้ลำดับการเข้าถึงแต่ละส่วนของออบเจ็กต์ในระดับความคิดของดีดีโอก่อน ดังรูปที่ 7.3 แสดงลำดับชั้นลอจิคอลของ ดีดีโอ (DDO Logical hierarchy)

7.2.2 ออบเจ็กต์และคอลเลคชัน

คอลเลคชันประกอบด้วยสมาชิกของออบเจ็กต์หลายๆชนิดเช่น ตาราง , โปรเจ็ค และโมเดล เช่น ดาต้ากรุ๊ปออบเจ็กต์เป็นสมาชิกของดาต้ากรุ๊ปคอลเลคชัน ซึ่งภายในอาจมีหลายดาต้ากรุ๊ปก็ได้ (ปกติคอลเลคชันจะถูกแทนในรูปของพหูพจน์)

เราสามารถกระทำกับคอลเลคชัน ได้ดังนี้

- ดึงออบเจ็กต์ที่ต้องการในคอลเลคชัน
- ลบออบเจ็กต์ที่ต้องการ ในคอลเลคชัน
- นับจำนวนออบเจ็กต์ในคอลเลคชัน



รูปที่ 7.3DDO logical hierarchy

7.2.3 ชนิดของออบเจ็กต์

เราแบ่งออบเจ็กต์เป็น 3 ประเภทคือ เฟรมเวิร์คออบเจ็กต์ (Framework Objects) , ดาต้าออบเจ็กต์ (Data Objects) , ซัพพอร์ตออบเจ็กต์ (Support Objects)

7.2.3.1 เฟรมเวิร์คออบเจ็กต์

ใช้แสดงโครงสร้างที่อยู่ในฐานข้อมูลและเตรียมการเข้าถึงระหว่างแต่ละออบเจ็กต์ที่อยู่ในแอพพลิเคชันและข้อมูลที่แอพพลิเคชันต้องการเข้าถึง แต่ละออบเจ็กต์อยู่ในประเภทนี้มีดังนี้

- เอ็นจิน (Engine)

เป็นออบเจ็กต์ที่อยู่ที่รูท (Root) เราต้องสร้างออบเจ็กต์นี้เป็นออบเจ็กต์แรกก่อนอื่น ซึ่งเราจะใช้ออบเจ็กต์นี้ในการควบคุมโปรเจ็คและโมเดลออบเจ็กต์ ซึ่งหนึ่งเอ็นจินออบเจ็กต์ สามารถมีหลายโปรเจ็ค

และหลายโมเดลได้ ภายในโปรแกรมเราต้องการแต่เพียงหนึ่งเอ็นจินออบเจ็กต์เท่านั้น เราสามารถกระทำกับเอ็นจินได้ดังนี้

- สร้างโปรเจ็คออบเจ็กต์
- สร้างโมเดลออบเจ็กต์
- สร้างสร้างล็อกออนอินโฟออบเจ็กต์

ออบเจ็กต์อื่นไม่สามารถเป็นเจ้าของเอ็นจินออบเจ็กต์ได้

เอ็นจินออบเจ็กต์เป็นเจ้าของโปรเจ็คออบเจ็กต์ , โมเดลออบเจ็กต์ และ ล็อกออนอินโฟออบเจ็กต์

- ล็อกออนอินโฟ

เป็นออบเจ็กต์ที่ใช้กำหนดชนิดของฐานข้อมูล (database type) , ชื่อผู้ใช้ (user name) , รหัสผ่าน (password) , หมายเลขเซิร์ฟเวอร์ (server ID) และชื่อฐานข้อมูล (database name) ที่ต้องใช้ในการเข้าสู่ระบบฐานข้อมูล เราสามารถกระทำกับล็อกออนอินโฟออบเจ็กต์ได้ดังนี้

- ล็อกออนสู่ฐานข้อมูล
- นำเข้าโครงสร้างข้อมูลในฐานข้อมูลเป็นโมเดล

ล็อกออนอินโฟออบเจ็กต์ มี เอ็นจินออบเจ็กต์เป็นเจ้าของ

ล็อกออนอินโฟออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ใดๆ ได้

- โปรเจ็กต์

เป็นออบเจ็กต์ที่ใช้ในการจัดการดาต้ากรุ๊ป , ตาราง , คอลัมน์ และรีเลชั่นออบเจ็กต์ (Relation object) ทุกออบเจ็กต์ที่ใช้บนแอพพลิเคชัน ซึ่งต้องสัมพันธ์กับไฟล์ดาต้าไดเรกเตอร์โปรเจ็กต์ (.ddx) เราสามารถกระทำกับโปรเจ็กต์ออบเจ็กต์ได้ดังนี้

- สร้างดาต้ากรุ๊ปออบเจ็กต์
- อีนาเบิล (Enable) และ ดิสเอเบิล (Disable) การควบคุมความผิดพลาด ขณะโปรแกรมทำงาน

โปรเจ็กต์ออบเจ็กต์ มี เอ็นจินออบเจ็กต์เป็นเจ้าของ

โปรเจ็กต์ออบเจ็กต์ เป็นเจ้าของดาต้ากรุ๊ปออบเจ็กต์ , เออเรอร์ออบเจ็กต์ (Error Object) และ เทรซออบเจ็กต์ (Trace object)

- โมเดล

เป็นออบเจ็กต์ที่ใช้ในการเข้าถึงและแก้ไขข้อมูลดาต้าไดเรกเตอร์โมเดล ซึ่งสัมพันธ์กับโมเดลไฟล์ (.mlt หรือ .mlx) เราสามารถกระทำกับโมเดลออบเจ็กต์ได้ดังนี้

- อิมพอร์ตโครงสร้างที่อยู่ในฐานข้อมูลเป็นโมเดล
- คัดลอกและบันทึกโมเดล
- สร้างเทเบิลออบเจ็กต์
- สร้างล็อกออนอินโฟออบเจ็กต์

โมเดลออบเจ็กต์ มี เอ็นจินออบเจ็กต์เป็นเจ้าของ

โมเดลออบเจ็กต์ เป็นเจ้าของ เทเบิลออบเจ็กต์ , รูทีนออบเจ็กต์ และ เออเรอร์ออบเจ็กต์

7.2.3.2 ดาต้าออบเจ็กต์

แต่ละฐานข้อมูลประกอบไปด้วยส่วนต่างๆ ของตาราง , คอลัมน์ และแถว ซึ่งดาต้าออบเจ็กต์นี้แสดงและทำให้เราสามารถทำงานในแต่ละส่วนของฐานข้อมูลได้

- ดาต้ากรุ๊ป

ดาต้ากรุ๊ป เป็นกลุ่มของความสัมพันธ์ของตารางในฐานข้อมูลกับดาต้าลิงค์ เราสามารถกระทำกับดาต้ากรุ๊ปออบเจ็กต์ได้ดังนี้

- สร้าง ล็อกออนอินโฟออบเจ็กต์
- สร้าง ดาต้าลิงค์ออบเจ็กต์
- สร้าง รูทีนออบเจ็กต์
- ล็อกออน และ ล็อกออฟ ฐานข้อมูล
- ตั้งค่าของการแสดงเออเรอร์รีพอร์ต (Error Report)
- ทำงาน SQL สเตจเมนต์

ดาต้ากรุ๊ปออบเจ็กต์ มี โปรเจ็กต์ เป็นเจ้าของ

ดาต้ากรุ๊ปออบเจ็กต์ เป็นเจ้าของ ดาต้าลิงค์ออบเจ็กต์ , เทเบิลออบเจ็กต์ และ เออเรอร์ออบเจ็กต์

- ดาต้าลิงค์

ดาต้าลิงค์คือตัวเชื่อมความสัมพันธ์ระหว่างคอลัมน์ในฐานข้อมูลและวิชวลเบสิกคอนโทรล (Visual Basic Control) เราสามารถกระทำกับ ดาต้าลิงค์ออบเจ็กต์ ได้ดังนี้

- คัดลอกดา ต้าลิงค์
- ติดตั้งค่าคุณสมบัติ ดาต้าลิงค์
- แสดงว่า ดาต้าลิงค์ นั้นเป็น ดาต้าลิงค์ จริง หรือ เวอร์ชวลดาต้าลิงค์ (virtual DataLink)

ดาต้าลิงค์ออบเจ็กต์ มี ดาต้ากรุ๊ปออบเจ็กต์ เป็นเจ้าของ

ดาต้าลิงค์ออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ใดๆได้

- เทเบิล

เพื่อสามารถทำงานกับตารางที่อยู่ในฐานข้อมูลได้ เราสามารถกระทำกับเทเบิลออบเจ็กต์ได้ดังนี้

- สร้าง รีเลชั่นออบเจ็กต์
- สร้าง คอลัมน์ออบเจ็กต์
- สร้าง เรคอร์ดออบเจ็กต์
- ค้นหา เรคอร์ดในรีซัลต์เซต (Result set)
- ค้นหา เรคอร์ด ที่ทำการ บุคมาร์ค ไว้
- กำหนด และสั่งใช้งาน สโตร์โปรซีเดอร์ (stroed procedures) , ฟังก์ชัน (functions) และ รูทีนผู้ใช้กำหนดเอง (user-defined routines)

เทเบิลออบเจ็กต์ มี ดาต้ากรุ๊ป และ โมเดลออบเจ็กต์ เป็นเจ้าของ

เทเบิลออบเจ็กต์ เป็นเจ้าของ คอลัมน์คอลเลคชัน (Column collections) และ รีเลชั่น (Relation collections) ถ้า เทเบิลออบเจ็กต์ เป็นส่วนหนึ่งของดาต้ากรุ๊ป มันสามารถเป็นเจ้าของเรคอร์ดคอลเลคชัน (Records Collection)ด้วย

- คอลัมน์

เพื่อสามารถทำงานกับคอลัมน์ที่อยู่ในฐานข้อมูลได้ เราสามารถกระทำกับคอลัมน์ออบเจ็กต์ได้ดังต่อไปนี้

- ใช้ในการเข้าถึงไบนารีลาร์กออบเจ็กต์ (blob values) ในฐานข้อมูล
- ใช้ในการเข้าถึงชนิดข้อมูลผู้ใช้กำหนดเองในฐานข้อมูล
- ติดตั้งค่าออปชันในการดึงข้อมูลและแคชชิ่ง blob values
- ติดตั้งค่าชนิดข้อมูลแบบคอลัมน์ (column data types,) ขนาด (size) และวิธีเรียงลำดับ (sorting)
- กำหนดคอลัมน์ของคีย์หลัก (Primary key columns)

คอลัมน์ออบเจ็กต์ มี เทเบิลและ เรคอร์ดออบเจ็กต์ เป็นเจ้าของ

คอลัมน์ออบเจ็กต์ เป็นเจ้าของ Blob ออบเจ็กต์ , และ โอเปคออบเจ็กต์ (Opaque Object)

- เรคอร์ด

เพื่อสามารถทำงานกับเรคคอร์ดที่อยู่ในฐานข้อมูลได้ เราสามารถกระทำกับเรคอร์ดออบเจ็กต์ ได้ดังนี้

- คัดลอกเรคอร์ด
- เคลียร์ค่าแฟล็กที่เปลี่ยนจากเดิม
- รีเซตข้อมูลเรคอร์ดเป็นค่าเริ่มต้นใหม่
- ล็อคเรคอร์ดที่อยู่ในระหว่างมีการเปลี่ยนแปลง

เรคอร์ดออบเจ็กต์ มี เทเบิลออบเจ็กต์ เป็นเจ้าของ

เรคอร์ด เป็นเจ้าของ คอลัมน์ออบเจ็กต์

- รีเลชั่น

เป็นออบเจ็กต์ที่กำหนดความสัมพันธ์ระหว่าง 2 ตาราง เราสามารถกระทำกับรีเลชันออบเจ็กต์ ได้ดังนี้

- สร้างจุดเริ่มต้นและปลายทางของตารางในความสัมพันธ์
- สร้างจุดเริ่มต้นและปลายทางของคอลัมน์ในความสัมพันธ์
- เพิ่มจุดเริ่มต้นและปลายทางของคอลัมน์ในความสัมพันธ์
- แสดงชนิดของความสัมพันธ์

รีเลชันออบเจ็กต์ มี เทเบิลออบเจ็กต์ เป็นเจ้าของ

รีเลชันออบเจ็กต์ไม่สามารถเป็นเจ้าของออบเจ็กต์ใดๆ ได้

- Blob

Blob value เก็บข้อมูลที่มีขนาดใหญ่ เช่น ไฟล์บิตแมป (Bitmap file) เราสามารถกระทำกับ Blob ออบเจ็กต์ ได้ดังนี้

- ดึงข้อมูล Blob
- ติดตั้งค่าลง Blob
- บันทึกและโหลด Blob ออบเจ็กต์และแปลงเป็นไฟล์
- ปลดปล่อยแคช (cached) ของ Blob ดาต้า

Blob ออบเจ็กต์ มี คอลัมน์ออบเจ็กต์ เป็นเจ้าของ

Blob ออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ใดๆ ได้

- โอเปค

เราสามารถกระทำกับโอเปคออบเจ็กต์ได้ดังนี้

- ดึงชนิดข้อมูลที่ผู้ใช้กำหนดเอง
- บันทึกและโหลดโอเปคออบเจ็กต์ไปยังและจากแอพพลิเคชันบัฟเฟอร์
- บันทึกและโหลดโอเปคออบเจ็กต์ไปยังและจากไฟล์
- ทดสอบโอเปคออบเจ็กต์ ที่เป็นแบบ ความยาวคงที่ (fixed length)
- ทดสอบ โอเปคออบเจ็กต์ ที่เป็น null
- เปรียบเทียบระหว่าง 2 ออบเจ็กต์ ถ้า ออบเจ็กต์ ทั้งสองมีชนิดข้อมูลโอเปคเดียวกัน

โอเปคออบเจ็กต์ มี คอลัมน์ออบเจ็กต์ เป็นเจ้าของ

โอเปคออบเจ็กต์ ไม่สามารถเป็นเจ้าของ ออบเจ็กต์ใดๆได้

- รูทีน

ออบเจ็กต์นี้แสดง user-defined function หรือ system-defined function หรือ สโตร์โปรซีเดอร์ ในฐานข้อมูล การใช้รูทีนออบเจ็กต์ เราสามารถเพิ่มสโตร์โปรซีเดอร์ และ รูทีนผู้ใช้กำหนดเอง จากฐานข้อมูลเป็นโมเดลออบเจ็กต์ ของเราเอง

รูทีนออบเจ็กต์ มี โมเดลออบเจ็กต์ เป็นเจ้าของ

รูทีนออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ใดๆ ได้

7.2.3.2 Support Objects

แต่ละโปรเจ็คที่ทำงานโดยใช้ดาต้าไดเรกเตอร์ ต้องการรูปแบบ "housekeeping" ฟังก์ชัน เช่น การเก็บสิ่งที่เกิดความผิดพลาดและที่เก็บไฟล์ ต่างๆ

- เออเรอร์

แสดงชนิดของเออเรอร์ , เวลา และคำอธิบายเออเรอร์นั้น

เออเรอร์ออบเจ็กต์ มี โปรเจ็กต์ , โมเดล และ ดาต้ากรุ๊ปออบเจ็กต์ เป็นเจ้าของ

เออเรอร์ออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ ใดๆได้

- เทรซ

ใช้ร่วมกับดาต้าไดเรกเตอร์เทรซไฟล์ (Data Director Trace file) ซึ่งเทรซไฟล์ คือข้อมูลที่แสดงว่าคำสั่งใดของ ดาต้าไดเรกเตอร์ ถูกเรียกใช้งาน เราสามารถกระทำกับเทรซได้ดังนี้

- อีนาเบิล (Enable) และ ดิสอะเบิลเทรซซิง (Disable tracing)
- เพิ่มคอลัมน์ที่ต้องการจะแสดงในเทรซไฟล์

เทรซออบเจ็กต์ มี โปรเจ็คออบเจ็กต์ เป็นเจ้าของ

เทรซออบเจ็กต์ ไม่สามารถเป็นเจ้าของออบเจ็กต์ ใดๆได้

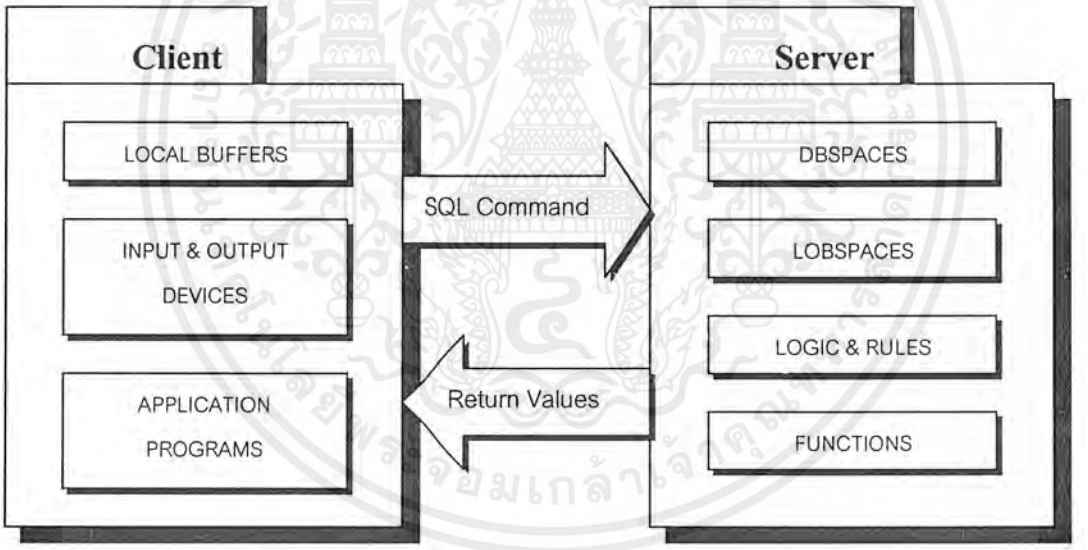
การเขียนโปรแกรมดีดีโอในแต่ละส่วน เราสามารถดูได้จากคู่มือของดาต้าไดเรกเตอร์

บทที่ 8
การออกแบบฐานข้อมูลแคด/แคม

8.1 สถาปัตยกรรมของฐานข้อมูลแคด/แคม

สถาปัตยกรรมของฐานข้อมูลแคด/แคม (CAD/CAM Database) จะเป็นแบบทูเทียร์ทิกค์ไคลเอนต์ (Two-Tier Thick Client Database Model) ซึ่งประกอบด้วยไคลเอนท์และเซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์มีหน้าที่เก็บฐานข้อมูลทั้งหมด รวมทั้งการตัดสินใจ (Logic) ต่าง ๆ , กฎเกณฑ์ และฟังก์ชันการทำงานอื่น ๆ ไคลเอนท์มีหน้าที่ในการนำข้อมูลในฐานข้อมูลมาแสดงและเรียกใช้บริการต่างๆ ที่อยู่บนเซิร์ฟเวอร์และใส่ข้อมูลเข้าไปในเซิร์ฟเวอร์

ไคลเอนท์สามารถติดต่อกับเซิร์ฟเวอร์ได้โดยตรงโดยใช้คำสั่ง SQL ภายในเซิร์ฟเวอร์จะประกอบด้วยกลไกทางทางเชิงวัตถุสัมพันธ์ (Object-Relational) ที่ควบคุมการเข้ามาใช้ข้อมูลต่างๆ จากผู้ใช้ สถาปัตยกรรมของฐานข้อมูลแคด/แคมแสดงได้ดังรูปที่ 8.1



รูปที่ 8.1 สถาปัตยกรรมของฐานข้อมูลแคด/แคม

การสั่งงานทั้งหมดของผู้ใช้จะถูกกระทำผ่านทางไคลเอนท์ ซึ่งจะมีโปรแกรมแอพพลิเคชั่นที่ถูกเขียนขึ้นเพื่อใช้งานเฉพาะ ผู้ใช้สามารถที่จะสร้างโปรเจ็คใหม่ หรือเปิดโปรเจ็คที่มีอยู่แล้วขึ้นมาได้ นอกจากนี้ผู้ใช้ยังสามารถเปลี่ยนแปลงแก้ไขวงจรในโปรเจ็ค และสามารถบันทึกข้อมูลต่าง ๆ เข้าไปในฐานข้อมูล การส่งข้อมูลจากไคลเอนท์ไปยังเซิร์ฟเวอร์เป็นหน้าที่ของโปรแกรมแอพพลิเคชัน ซึ่งผู้ใช้ไม่สามารถมองเห็นคำสั่ง SQL ที่ส่งไปยังเซิร์ฟเวอร์สิ่งที่ผู้ใช้สามารถเห็นได้มีเพียงรูปภาพของวงจรที่มีอยู่ในฐานข้อมูล

ความสามารถของผู้ใช้ในการใช้งานฐานข้อมูลแคด/แคม คือ

1. ผู้ใช้สามารถดึงข้อมูลวงจรดิจิตอลบนเซิร์ฟเวอร์มาแสดงผลบนจอภาพได้
2. ผู้ใช้สามารถสร้างวงจรดิจิตอลใหม่
3. ผู้ใช้สามารถเปลี่ยนแปลงแก้ไขวงจรดิจิตอลที่ดึงขึ้นมาแสดงผลได้โดยอิสระ
4. ผู้ใช้สามารถบันทึกวงจรดิจิตอลลงไปบนฐานข้อมูล ไม่ว่าจะอยู่ในรูปของการบันทึกทับวงจรเดิม (Save) และบันทึกในชื่อใหม่ (Save as)
5. จำลองการทำงานของวงจร (Simulation) แสดงผลลัพธ์ในรูปแบบตัวเลขลอจิก ณ ขาต่าง ๆ บนอุปกรณ์
6. ผู้ใช้สามารถวัดสัญญาณลอจิก ณ จุดต่าง ๆ บนวงจรดิจิตอล

สิ่งที่ผู้ใช้ไม่สามารถทำได้ เมื่อใช้งานแอพพลิเคชัน คือ

1. ทำการเปลี่ยนแปลงแก้ไขรายการอุปกรณ์ต้นแบบ (Component type) ที่เก็บไว้บนฐานข้อมูล
2. ทำการเพิ่มอุปกรณ์ใหม่ ๆ เข้าไปในระบบฐานข้อมูล
3. ตรวจดูคำสั่ง SQL ที่โปรแกรมแอพพลิเคชันส่งไปยังเซิร์ฟเวอร์รวมทั้งผลลัพธ์ที่เซิร์ฟเวอร์ส่งมาให้กับโปรแกรมแอพพลิเคชัน

8.2 การออกแบบโครงสร้างของฐานข้อมูลแคด/แคม

ในการจัดสร้างฐานข้อมูลที่ใช้ในการเก็บวงจรดิจิตอล จะต้องมีการออกแบบองค์ประกอบต่าง ๆ ของวงจร ซึ่งวงจรดิจิตอลนั้นสามารถจัดสร้างได้ง่ายกว่าวงจรอีเล็กทรอนิกส์โดยทั่วไป เพราะไม่ต้องมีการคำนึงถึงระดับสัญญาณอนาล็อก และการทำงานของอุปกรณ์ที่มีความซับซ้อนอื่นๆ เช่น ทรานซิสเตอร์ (Transister), รีซีสเตอร์ (Resister), คาปาซิเตอร์ (Capacitor) เป็นต้น

ในการออกแบบวงจรดิจิตอลจะมีการจัดสร้างอุปกรณ์ในระดับเกท และอุปกรณ์ในแต่ละชิ้นที่มีการทำงานเฉพาะ มีระดับสัญญาณในระดับลอจิกคือ 0 หรือ 1 เท่านั้น จึงมีความง่ายในการคำนวณ โดยการจัดเก็บวงจรดิจิตอลในฐานข้อมูลจะมีการจัดเก็บอุปกรณ์ต่าง ๆ และการเชื่อมต่อระหว่างอุปกรณ์ต่าง ๆ พร้อมทั้งฟังก์ชันการทำงานที่สามารถจำลองการทำงานของวงจรนั้น ๆ ซึ่งจะอยู่ในรูปของฟังก์ชันที่เก็บอยู่ในดาต้าเบสเซิร์ฟเวอร์ที่สามารถกระทำโอเปอเรชันกับวงจรดิจิตอลได้ทันที

ส่วนประกอบต่าง ๆ ของวงจรดิจิตอลที่จะมีการจัดเก็บในฐานข้อมูล ประกอบด้วยองค์ประกอบที่สำคัญ 3 ส่วนด้วยกัน คือ

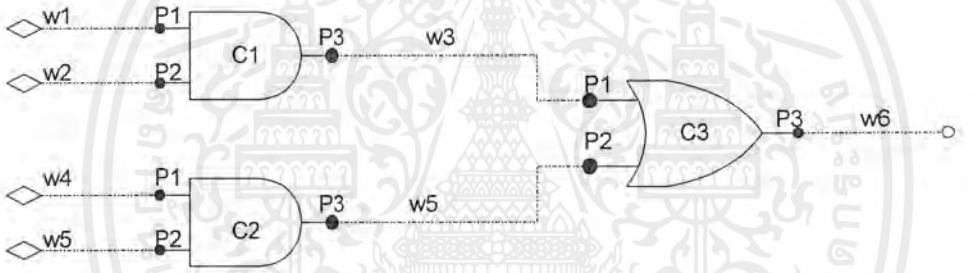
- *คอมโพเนนท์ (Component)* คือส่วนของตัวอุปกรณ์แต่ละตัว เช่น OR เกท , AND เกท , JK ฟลิป-ฟลอป ซึ่งจะมีการทำงานเฉพาะภายใน ซึ่งอุปกรณ์แต่ละตัวจะถูกจัดเก็บเอาไว้ให้ผู้ใช้สามารถดึงอุปกรณ์เหล่านี้ไปใช้ได้ในการจัดสร้างวงจรดิจิตอล ในส่วนของแอพพลิเคชันโปรแกรม

- *พิน (Pin)* คือขาแต่ละขาของอุปกรณ์แต่ละตัว ซึ่ง Pin จะต้องเกาะติดเข้ากับ Component ซึ่งแต่ละ Component จะมีขาของตัวเองเป็นจำนวนที่คงที่ การลบ Component ใด ๆ ออกจากฐานข้อมูลจะทำให้ขาของอุปกรณ์ตัวนั้นถูกลบออกไปด้วยเช่นกัน การเชื่อมต่อกันระหว่าง Component จะเชื่อมต่อกันผ่าน Pin เท่านั้น

- *ไวร์ (Wire)* คือสายสัญญาณที่เชื่อมต่ออุปกรณ์แต่ละตัวเข้าด้วยกัน จะเป็นการเชื่อมต่อระหว่างอุปกรณ์เพียง 2 อุปกรณ์ คืออุปกรณ์ต้นทาง และอุปกรณ์ปลายทางเท่านั้น ซึ่งจะเป็นการเชื่อมต่อผ่าน Pin ที่เกาะติดกับอุปกรณ์เหล่านั้น

ในฐานข้อมูลหนึ่งๆ ที่เก็บวงจรดิจิตอลเอาไว้ อาจมีการเก็บวงจรดิจิตอลไว้รวมกันหลาย ๆ วงจร จะมีการนิยามคำว่าโปรเจ็ค ซึ่งในโปรเจ็คหนึ่ง ๆ คือวงจรดิจิตอลจำนวน 1 วงจรเท่านั้น ซึ่งในวงจรนั้นสามารถทำงานด้วยตัวเอง มีชุดของอินพุตและชุดของเอาท์พุตของตัวมันเอง

วงจรในรูปแสดงตัวอย่างขององค์ประกอบพื้นฐานของวงจรดิจิตอล

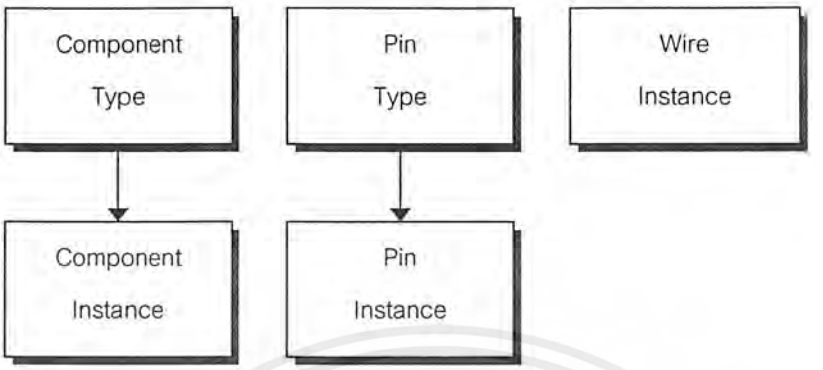


รูปที่ 8.2 วงจรตัวอย่างวงจรดิจิตอล

จากรูปที่ 8.2 ประกอบด้วยคอมโพเนนท์ 3 คอมโพเนนท์ คือ C1 , C2 และ C3 เป็นคอมโพเนนท์ชนิด AND เกท 2 คอมโพเนนท์ ได้แก่ C1 และ C2 และ คอมโพเนนท์ OR เกท 1 คอมโพเนนท์ ได้แก่ C3 ทั้ง 3 คอมโพเนนท์มีพินเกาะติดอยู่ 3 พินเท่ากันได้แก่ P1 , P3 และ P3 และแต่ละคอมโพเนนท์ เชื่อมต่อกันด้วยไวร์ ได้แก่ W1 ถึง W7 แต่ละไวร์จะเชื่อมต่อระหว่างพิน 2 พิน ซึ่งพินอาจจะอยู่ในคอมโพเนนท์เดียวกัน หรือว่าอยู่คนละคอมโพเนนท์ก็ได้เช่นกัน เช่น จากรูป W13 เชื่อมต่อระหว่าง P3 ของ คอมโพเนนท์ C1 กับ P1 ของคอมโพเนนท์ C3 นอกจากนี้ยังมี Pin ยังมี 2 ประเภท ได้แก่พินที่เป็นชุดเชื่อมต่อหรือเป็นพินที่ไม่ได้ลอยอยู่ และอีกประเภทหนึ่งคือเทอร์มินอล (Terminal) คือพินที่ต้องการให้เป็นอินพุตหรือ เอาท์พุตของวงจร

ออบเจ็กต์ทั้ง 3 ออบเจ็กต์ได้ถูกสร้างเป็นคลังเก็บข้อมูลต้นแบบเอาไว้ ซึ่งถือว่าเป็นชนิดของออบเจ็กต์ของทุก ๆ ออบเจ็กต์ โดยที่คอมโพเนนท์และพินได้ถูกสร้างเป็นชนิดของออบเจ็กต์เอาไว้ แต่ไวร์ไม่ได้ถูกสร้างเอาไว้ เนื่องจากไวร์ไม่ได้ถูกแบ่งแยกออกเป็นหลายชนิด จึงสร้างเฉพาะอินสแตนซ์ (Instance) ของไวร์เท่านั้น โดยที่อินสแตนซ์ของคอมโพเนนท์จะอยู่ในชนิดข้อมูลของคอมโพเนนท์ไทพ์

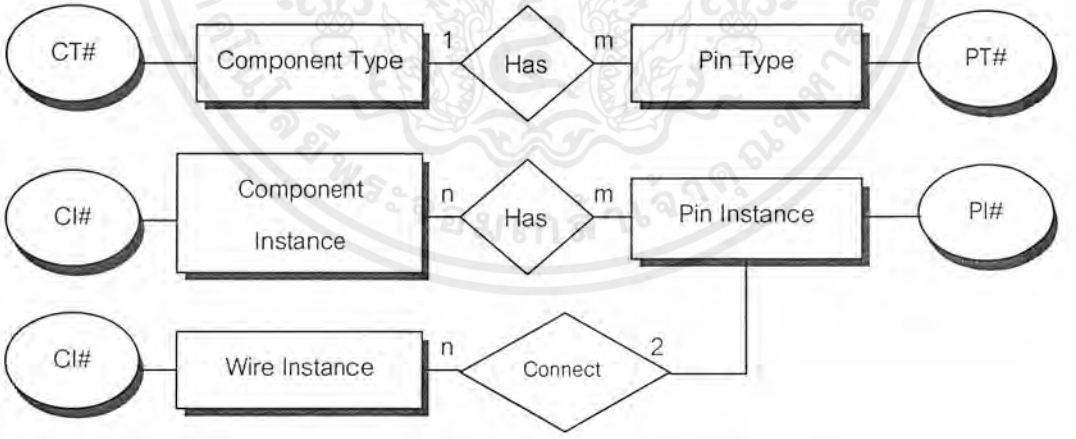
(Component type) และอินสแตนซ์ของพินเป็นชนิดของข้อมูลของพินไทพ์ (Pin Type) ส่วนไวร์มีเฉพาะอินสแตนซ์ของไวร์เท่านั้น แสดงการสร้างอินสแตนซ์ต่าง ๆ ได้ดังรูปที่ 8.3



รูปที่ 8.3 ออบเจ็กต์ต่าง ๆ ของวงจรดิจิตอล

สำหรับความสัมพันธ์ของแต่ละออบเจกต์คือ คอมโพเนนท์ไทพ์ จะมีความสัมพันธ์กับพินไทพ์เช่นเดียวกันกับความสัมพันธ์ระหว่างอินสแตนซ์ของคอมโพเนนท์กับอินสแตนซ์ของพินในหนึ่ง คอมโพเนนท์สามารถมีพินได้หลายพิน เช่น เกตเบอร์ 74LS245 มีพิน 1 ถึง พิน 20 เป็นต้น

การออกแบบสามารถสร้างความสัมพันธ์ระหว่างแต่ออบเจ็กต์ในรูปแบบของอีอาร์ไดอะแกรม (ER Diagram) แสดงได้ดังรูปที่ 8.4



รูปที่ 8.4 ความสัมพันธ์ระหว่างออบเจกต์ต่าง ๆ ใน วงจรดิจิตอล

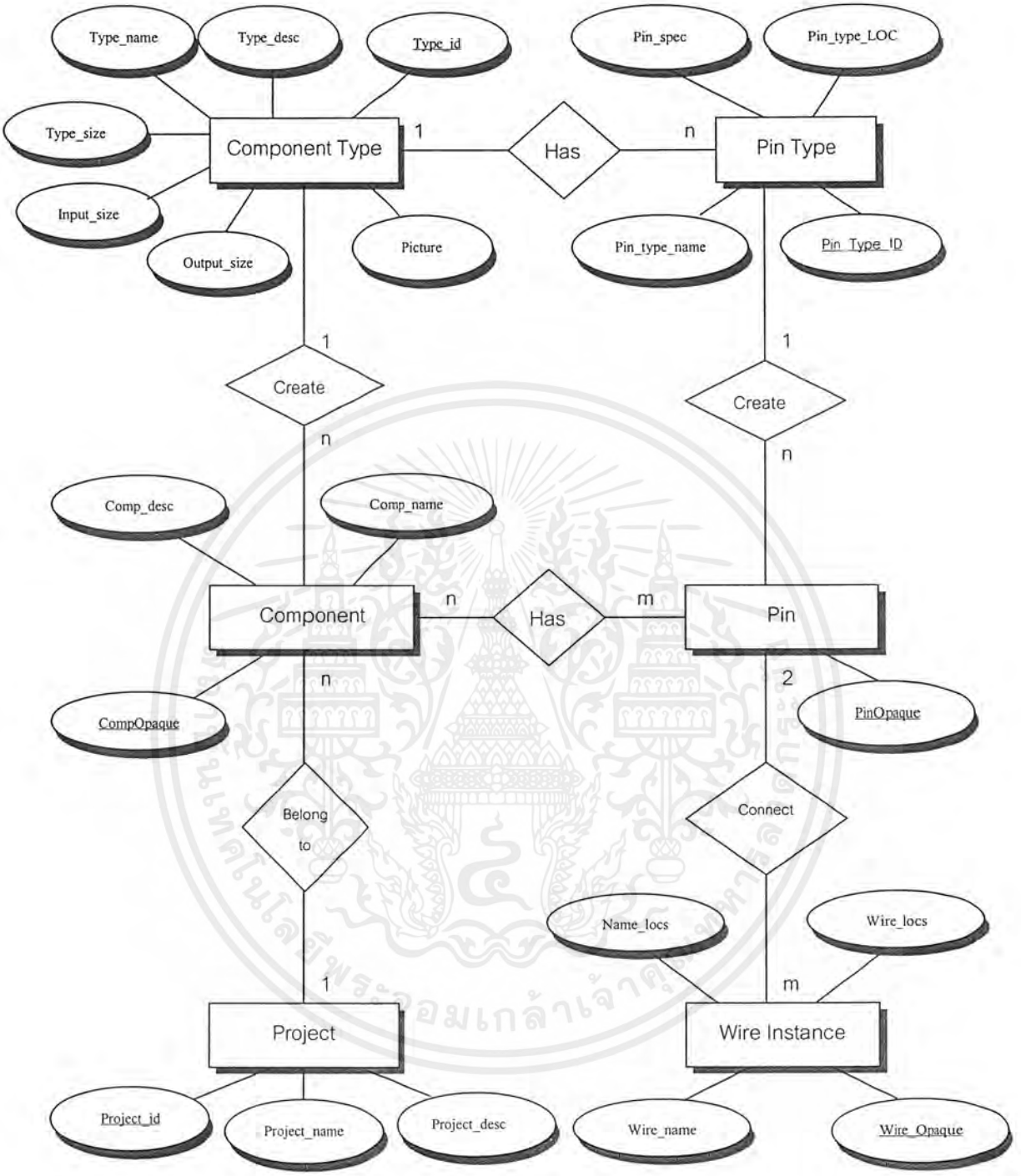
จากรูปที่ 8.4 คอมโพเนนท์ไทพ์ มีความสัมพันธ์กับ พินไทพ์ แบบ One to many และอินสแตนซ์ของคอมโพเนนท์มีความสัมพันธ์ กับอินสแตนซ์ของพินแบบ one to many เช่นกัน ส่วนอินสแตนซ์ของไวร์จะมีความสัมพันธ์กับอินสแตนซ์ของพินแบบ 2-to-many ซึ่งไวร์จะเชื่อมต่อระหว่างพิน 2 พิน แต่พิน 1 พินสามารถมีไวร์มาเชื่อมต่อได้มากกว่า 1 ไวร์ได้ เมื่อได้สร้างความสัมพันธ์ระหว่างออบเจ็กต์ได้แล้ว

สามารถสร้างโมเดลที่จะสร้างฐานข้อมูลที่ใช้เก็บวงจรดิจิตอล ที่แสดงความสัมพันธ์ระหว่างองค์ประกอบหลัก ๆ ได้ดังรูปที่ 8.4

จุดประสงค์ในการสร้างคอมโพเนนท์ไทพ์ และ พินไทพ์ ก็เพื่อสร้างไว้สำหรับการอ้างอิงในการสร้างอินสแตนซ์ออกมาใช้ในโปรเจ็ค สิ่งที่ผู้ใช้สามารถทำได้คือการสร้างคอมโพเนนท์ของอุปกรณ์ที่ต้องการซึ่งฐานข้อมูลมีให้ใช้ เมื่อทำการสร้างคอมโพเนนท์ใด ๆ พินที่เกาะติดอยู่กับคอมโพเนนท์ ก็จะถูกสร้างขึ้นมาด้วยเช่นเดียวกัน นอกจากนั้นผู้ใช้สามารถที่จะสร้างสายสัญญาณที่ใช้ในการเชื่อมต่อระหว่างอุปกรณ์แต่ละตัวได้เช่นกัน

ในโครงงานนี้ผู้ใช้ยังไม่สามารถที่จะทำการอัพเดตค่าข้อมูลต่าง ๆ ในคอมโพเนนท์ไทพ์และพินไทพ์ได้ การเพิ่มหรือการทำการอัพเดตค่าต่าง ๆ เป็นหน้าที่ของดาต้าเบสแอดมินนิสเตรเตอร์ (Database Administrator) เพื่อป้องกันข้อมูลในฐานข้อมูลให้มีความปลอดภัยมากยิ่งขึ้น

รูปที่ 8.5 แสดงโครงสร้าง และความสัมพันธ์ในรูปแบบของอีอาร์ไดอะแกรมของวงจรดิจิตอลในฐานข้อมูลแคด/แคมโดยสมบูรณ์



รูปที่ 8.5 อีอาร์ไดอะแกรมของฐานข้อมูลแคด/แคม

รายละเอียดของแต่ละองค์ประกอบมีดังต่อไปนี้

- *Project*

โดยการจัดเก็บวงจรดิจิตอลจำนวน 1 วงจร ซึ่งในหนึ่งโปรเจ็ค จะมีอินพุตและเอาท์พุตของวงจรจำนวนหนึ่ง ซึ่งจะมีตารางในการจัดเก็บโปรเจ็คจำนวน 1 ตาราง โดยแอตทริบิวต์ต่าง ๆ ของ Project มีดังนี้

- *Project_ID* เป็นคีย์หลักที่ใช้ในการระบุถึงโปรเจ็คแต่ละโปรเจ็ค จะมีการสร้างโดยอัตโนมัติ จะเป็นชนิดข้อมูลแบบ Serial โดยหมายเลขของโปรเจ็คแอพพลิเคชั่นโปรแกรมจะเป็นตัวหาหมายเลขที่เหมาะสมให้
- *Project_name (Candidate key)* คือชื่อของโปรเจ็คที่ผู้สร้างสามารถตั้งชื่อเอาไว้ โดยจะต้องมีชื่อไม่ซ้ำกับชื่อของโปรเจ็คที่มีอยู่แล้ว ผู้ใช้สามารถใช้ชื่อในการสืบค้นหาโปรเจ็คที่ต้องการได้ ในการใช้งานจริง
- *Project Description* เป็นคำอธิบายอื่นๆ ที่ผู้ใช้ที่จัดสร้างวงจรต้องการอธิบาย เพื่อความสะดวกในการค้นหาวงจรที่ผู้ใช้ต้องการ แต่ไม่สามารถจำชื่อโปรเจ็คได้

❑ *Component_Type*

จะเป็นองค์ประกอบที่ใช้เก็บรายการหรือชนิดของอุปกรณ์ที่ผู้ใช้สามารถดึงมาใช้งานได้ มีรายการของแอตทริบิวต์ต่าง ๆ ดังนี้

- *Type_ID* เป็นคีย์หลักจะเป็นหมายเลขที่ใช้ระบุถึงรายการอุปกรณ์แต่ละรายการ
- *Type_name* คือชื่อของอุปกรณ์แต่ละรายการ บ่งบอกได้ว่าเป็นอุปกรณ์ประเภทใด เช่น AND2 แสดงว่าเป็น AND เกต 2 Input หรือ JKFF แสดงว่าเป็น J-K ฟลิป-ฟลอป เป็นต้น ซึ่ง Type_name จะเป็นชนิดข้อมูลแบบสตริง ซึ่ง Type_name สามารถซ้ำกันได้ในกรณีที่อุปกรณ์หนึ่ง ๆ มีหลาย ๆ ตระกูล แต่ทำงานหน้าที่เดียวกัน
- *Type_desc* คือคำอธิบายเพิ่มเติมของรายการของอุปกรณ์นั้นๆ ซึ่งอาจจะเป็นคำอธิบายที่เกี่ยวกับหน้าที่การทำงาน การประยุกต์ใช้ หรือรายละเอียดอื่น ๆ ที่ผู้ใช้สามารถอ่านและทำความเข้าใจและสามารถนำอุปกรณ์เหล่านั้นไปทำงานได้อย่างถูกต้อง
- *Type_delay_time* คือค่าของช่วงเวลาหลังจากที่มีอินพุตเข้ามาที่อุปกรณ์และสามารถให้ผลลัพธ์ที่เอาท์พุตออกมาได้ ซึ่งการทำงานขั้นสูงอาจจะมีผลต่อการทำงานของวงจรโดยรวม ซึ่งในโปรเจ็คที่ได้ทำนี้ยังไม่ได้มีการจัดทำอุปกรณ์ที่มีคุณสมบัติของการหน่วงเวลา
- *Picture* คือรูปภาพของอุปกรณ์ ซึ่งจะใช้แสดงในแอพพลิเคชั่นโปรแกรมรูปภาพต่าง ๆ จะถูกเก็บไว้ในเซิร์ฟเวอร์อยู่ใน ในรูปแบบของ Binary Large Object (BLOB) แอพพลิเคชั่นสามารถดึงรูปภาพของอุปกรณ์นั้นมาแสดงบนหน้าจอในตำแหน่งที่ต้องการได้
- *Type_Size* เป็นขนาดของรูปภาพ ซึ่งกำหนดว่าแต่ละรูปของชนิดอุปกรณ์จะมีขนาดกว้างคูณยาวที่เท่ากัน เพื่อสะดวกในการวาด ในโครงงานนี้รูปภาพของอุปกรณ์จะมีขนาดกว้างและยาวเท่ากัน เพื่อสะดวกในการเขียนแอพพลิเคชัน
- *Type_input_size* เป็นขนาดของขาอินพุตของชนิดอุปกรณ์นั้น ๆ
- *Type_output_size* เป็นขนาดของขาเอาท์พุตของชนิดอุปกรณ์นั้น ๆ

❑ *Pin_type*

คือองค์ประกอบที่จัดการเก็บรายละเอียดของขาที่ใช้งานต่าง ๆ ของแต่ละรายการอุปกรณ์ โดยแสดงว่าอุปกรณ์รายการต่าง ๆ ในคอมโพเนนท์ไทพ์มีขาอะไรบ้าง แต่ละขามีหน้าที่อย่างไร มีคุณสมบัติอะไรบ้าง ตาราง Pin_type มีรายการของแอตทริบิวต์ต่าง ๆ ดังต่อไปนี้

- *Type_ID* คือค่าที่ใช้ในการอ้างถึงคอมโพเนนท์ไทพ์ว่าขาหมายเลขดังกล่าวเป็นของรายการอุปกรณ์ใดบนตาราง Componen_type
- *Pin_type_ID* คือหมายเลขของขาแต่ละตำแหน่งบนตัวอุปกรณ์ใด ๆ ซึ่งระบุใน Type_ID ซึ่งจะไม่สามารถซ้ำกันได้บนตัวอุปกรณ์เดียวกัน
- *Pin_type_name* คือชื่อของขาหมายเลขนั้นบนตัวอุปกรณ์ ซึ่งเป็นชนิดข้อมูลแบบสตริง ซึ่งจะไม่ซ้ำกันบนตัวอุปกรณ์เดียวกัน ชื่อนี้จะสื่อความหมายในทางเทคนิค เช่น J-K ฟลิป-ฟลอป มีรายการของชื่อขาแต่ละขา ได้แก่ 'J', 'K', 'Q', 'NotQ' เป็นต้น
- *Pin_spec* คือ ชนิดของขาที่บ่งบอกว่าเป็นอินพุต หรือว่าเอาท์พุต ซึ่ง 0 หมายถึงอินพุต และ 1 เป็นขาเอาท์พุต
- *Pin_type_LOC* คือ ตำแหน่งจุดพิกัดของขาบนตัวอุปกรณ์ ซึ่งจะมีประโยชน์ในการเชื่อมต่อกับสัญญาณที่มาเชื่อมต่อ เพื่อแสดงการเชื่อมต่อขานั้นบนแอพลิเคชันโปรแกรม

❑ *Component*

คือ อุปกรณ์ที่ผู้ใช้จัดสร้างขึ้น เพื่อประกอบเข้าเป็นวงจรในโปรเจ็คหนึ่ง ๆ ซึ่งถือว่าเป็นอินสแตนซ์ของข้อมูลในตาราง Component_type จะเป็นส่วนที่ผู้ใช้จัดสร้างขึ้น ซึ่งในโปรเจ็คหนึ่ง ๆ จะประกอบด้วยอุปกรณ์หลายอุปกรณ์ประกอบเข้าด้วยกัน แอตทริบิวต์ต่างๆ ที่สำคัญของตาราง Component มีดังนี้

- *Component_ID* คือ หมายเลขระบุถึงอุปกรณ์แต่ละตัว ซึ่งจะไม่ซ้ำกันในโปรเจ็คเดียวกัน
- *Type_ID* คือ ตัวเลขระบุถึงชนิดของอุปกรณ์นั้นๆ ซึ่งจะต้องมีอยู่ในตาราง Component_type
- *Component_name* คือ ชื่อที่ตั้งให้กับอุปกรณ์แต่ละตัว ซึ่งเป็นหน้าที่ของผู้สร้างวงจร ที่จะต้องการใส่ชื่อให้อุปกรณ์หรือไม่ ชื่ออุปกรณ์อาจสื่อความหมายในหลายๆ ด้าน แล้วแต่ผู้สร้างวงจรต้องการ เช่น 'U1', 'FF1' เป็นต้น
- *Component_desc* คือ คำอธิบายเพิ่มเติมของอุปกรณ์ ซึ่งเป็นรายละเอียดที่ไม่สามารถแสดงได้บนอุปกรณ์ได้ หรืออาจจะเป็นหมายเหตุ ที่อธิบายหน้าที่ของอุปกรณ์ เมื่อผู้ที่เปิดดูวงจรไม่ใช่ผู้สร้างวงจร สามารถทำความเข้าใจวงจรได้ง่ายขึ้น
- *Component_LOC* คือ ตำแหน่งจุดพิกัดของอุปกรณ์ที่จะต้องแสดงบนแอปพลิเคชัน ซึ่งเป็นตำแหน่งซ้ายบนของอุปกรณ์ที่จะจัดวางบนจอภาพ ซึ่งตำแหน่งนี้เป็นค่าพิกัด X และ Y

❑ *Pin*

คือตารางที่เก็บขาต่าง ๆ ของอุปกรณ์ที่ถูกสร้างขึ้น เมื่อผู้ใช้สร้างอุปกรณ์ใหม่ ซึ่งพินจะผูกติดกับคอมโพเนนท์กลายเป็นอุปกรณ์ที่ประกอบด้วยชิ้นอุปกรณ์และชุดของขา เมื่อผู้ใช้สร้างคอมโพเนนท์ใหม่ พินที่ต้องมีบนอุปกรณ์ชิ้นนั้นจะถูกสร้างขึ้นด้วย และเช่นเดียวกัน ถ้าผู้ใช้ทำการลบอุปกรณ์นั้นทิ้ง พินที่เกี่ยวข้องจะถูกลบออกด้วย ตาราง Pin ประกอบด้วยแอตทริบิวต์ต่าง ๆ ดังต่อไปนี้

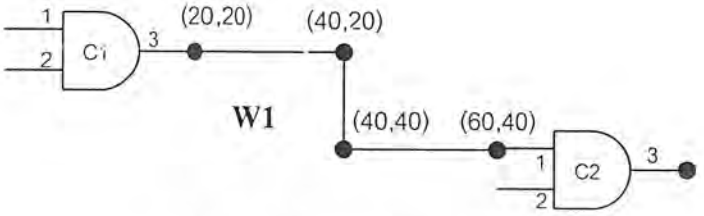
- *Pin_ID* คือหมายเลขของขาบนตัวอุปกรณ์ มีความหมายเดียวกับ Pin_type แต่ Pin_ID จะใช้ในการสร้างอินสแตนซ์ Instance สำหรับสร้างวงจร แต่ Pin_type เป็นเพียงชนิดของพินที่กำหนดไว้เพื่อประโยชน์ในการอ้างอิง Pin_ID จะไม่ซ้ำกันบนคอมโพเนนท์เดียวกัน
- *Component_ID* คือ หมายเลขของอุปกรณ์ที่ขานั้น ๆ เกาะติดอยู่
- *Pin_value* คือ ค่าทางลอจิกของขา ซึ่งจะเป็นสัญญาณทางไฟฟ้า และเป็นระดับสัญญาณแบบดิจิตอล ซึ่งอาจ '0' ,'1'
- *Pin_io* คือ ชนิดของขา จะแสดงว่าขานั้นเป็นอินพุต หรือเอาท์พุต ซึ่ง 1 เป็นเอาท์พุต และ 0 เป็นอินพุต ซึ่งค่าจะสัมพันธ์กับ Pin_spec ในตาราง Pin_type แต่เพื่อความสะดวกและรวดเร็วในการขั้นตอนการ Query จึงได้นำมารวมไว้ใน Opaque type Pin
- *Is_terminal* คือตัวบ่งบอกว่าขาอุปกรณ์นั้นเป็นจุดเริ่มต้นหรือจุดสิ้นสุดของวงจรหรือไม่ ซึ่งอาจจะเป็นจุดที่เป็นอินพุตของวงจร หรืออาจจะเป็นจุดที่เป็นเอาท์พุตของวงจรก็ได้ ซึ่งสามารถทำการตรวจสอบได้ว่าเป็นอินพุต และเอาท์พุตโดยการตรวจสอบคุณสมบัติขานั้น ๆ ใน Pin_type ค่าของ Is_terminal อาจจะเป็น 0 คือเป็นไม่เป็นขาเทอร์มินอล และ 1 เป็นขาเทอร์มินอล

❑ *Wire*

คือ สายสัญญาณที่เชื่อมต่อระหว่างขาใด ๆ จำนวน 2 ขา ซึ่งในวงจรดิจิตอลจริง ๆ ไวร์ คือสายไฟหรือ ลายวงจร ซึ่งเป็นทางเดินของกระแสไฟฟ้าในวงจร มีแอตทริบิวต์ดังต่อไปนี้

- *Wire_ID* คือหมายเลขของสายสัญญาณ ซึ่งจะควบคุมไม่ให้ซ้ำกันในวงจรหนึ่ง ๆ
- Wire_name คือชื่อของสายสัญญาณ ซึ่งผู้ใช้เป็นผู้กำหนด เพื่ออธิบายถึงสายสัญญาณหนึ่ง ๆ ซึ่งอาจจะมีบางสายสัญญาณที่อาจไม่มีความจำเป็นต้องมีการใส่ชื่อของสายสัญญาณก็ได้
- *First_comp_ID* คือหมายเลขที่ระบุถึงอุปกรณ์ต้นทางของการเชื่อมต่อ
- *First_pin_ID* คือหมายเลขของขาของอุปกรณ์ต้นทางของการเชื่อมต่อ
- *Second_comp_ID* คือหมายเลขที่ระบุถึงอุปกรณ์ปลายทางของการเชื่อมต่อ
- *Second_pin_ID* คือหมายเลขของขาของอุปกรณ์ปลายทางของการเชื่อมต่อ
- *Wire_value* คือค่าสัญญาณทางไฟฟ้าบนสายสัญญาณ จะมีการตรวจสอบสถานะทางไฟฟ้าของขาของอุปกรณ์ที่ทำการเชื่อมต่อ และมีการปรับค่าที่เหมาะสมให้กับสายสัญญาณ
- *Name_LOC* คือจุดพิกัดที่แสดงตำแหน่งของชื่อของสายสัญญาณที่จะใช้แสดงบนจอภาพ ซึ่งจะมีประโยชน์ในการเปิดวงจรขึ้นมาในภายหลัง จะให้การแสดงผลที่เหมือนเดิม

- *Wire_LOCS* คือ ชุดของพิกัดตำแหน่งบนจอภาพของสายสัญญาณที่เชื่อมต่อระหว่างอุปกรณ์ 2 อุปกรณ์ ซึ่งจะมีประโยชน์ในการเปิดวงจรดูในภายหลัง จะได้การแสดงผลต่าง ๆ ที่เหมือนเดิมทุกประการ ดังรูปที่ 8.6



รูปที่ 8.6 ตัวอย่างแอตทริบิวต์ Wire_LOCS

จากรูป W12 เป็นสายสัญญาณที่เชื่อมต่อระหว่าง ขา 3 ของคอมโพเนนท์ C1 และ ขา 1 ของคอมโพเนนท์ C2 เข้าด้วยกัน โดยมีการหักมุมต่าง ๆ ของสายสัญญาณ ทำให้มีการเก็บจุดต่าง ๆ ของสายสัญญาณเอาไว้ใน Wire_LOCS ซึ่งประกอบด้วยจุดพิกัด 4 จุด ได้แก่ (20,20) , (40,20) , (40,40) , (60,40) ซึ่งจุดพิกัดทั้งหมด จะถูกเก็บไว้ในลิสต์ ซึ่งเป็นชนิดข้อมูลหนึ่งของอินฟอร์มิกซ์ ซึ่งลำดับของแต่ละจุดพิกัดมีความสำคัญ

8.3 ชนิดของข้อมูลแบบโอเปค

ในการจัดสร้างระบบฐานข้อมูลเชิงสัมพันธ์ โดยการใช้อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์จะสามารถสร้างชนิดข้อมูลใหม่ขึ้นมาได้ โดยใช้ชนิดของข้อมูลแบบโอเปค โดยประกอบด้วยแอตทริบิวต์ต่าง ๆ อัดแน่นอยู่ภายใน มีการปกป้องข้อมูล (Encapsulated) ระดับไพรเวท และสามารถสร้างฟังก์ชันต่างๆ ที่กระทำกับชนิดข้อมูลใหม่นี้ได้

ดังนั้น การจัดสร้างอินสแตนซ์ทั้ง 3 อย่างได้แก่ Component , Pin และ Wire สามารถแสดงแอตทริบิวต์ บางแอตทริบิวต์ที่สื่อความหมายทางออบเจ็กต์ในโลกแห่งความเป็นจริง เข้ารวมไว้ด้วยกันภายในชนิดของข้อมูลแบบโอเปค ซึ่งชนิดของข้อมูลแบบโอเปคจะมีโครงสร้างเหมือนกับโครงสร้างในภาษาซี ซึ่งรายละเอียดของการกำหนดชนิดของข้อมูลแบบโอเปคมีดังนี้

- ชนิดของข้อมูลแบบโอเปคของ LOC ซึ่งเก็บจุดพิกัดต่าง ๆ

```
struct   LOC
{
        int xpos;
        int ypos;
}
```

LOC มีฟังก์ชันสนับสนุนดังต่อไปนี้

- *Equal(Loc1,Loc2)* จะเปรียบเทียบระหว่างจุดพิกัด 2 จุดว่าเท่ากันหรือไม่ ซึ่งสามารถเปรียบเทียบโดยการใช้โอเปอเรเตอร์ได้เช่นกัน เช่น A=B เป็นต้น ซึ่งถ้า LOC 2 ตัวเท่ากัน จะมีค่า xpos และ ypos เท่ากัน
- *Int Getxpos(Loc)* ทำการอ่านค่า xpos
- *Int Getypos(Loc)* ทำการอ่านค่า ypos
- *Setxpos (Loc,xpos)* ทำการเซตค่า xpos
- *Setypos (Loc,ypos)* ทำการเซตค่า ypos

❑ ชนิดของข้อมูลแบบโอเปคของ Component

```
struct    component
{
          int component_id;
          int component_type;
          loc component_loc;
}
```

Component มีฟังก์ชันสนับสนุนดังต่อไปนี้

- *Equal(Component1,Component2)* เปรียบเทียบระหว่าง 2 Component ว่าเท่ากันหรือไม่ ซึ่ง 2 Component เท่ากันก็ต่อเมื่อมี Component_ID และ Component_type เท่ากัน ซึ่ง Equal จะมีประโยชน์ในการที่จะใช้โอเปคเป็นคีย์ในตาราง
- *Int GetID(Component)* ทำการอ่านค่าหมายเลขของอุปกรณ์ (Component_ID)
- *Int GetType(Component)* ทำการอ่านค่าชนิดของอุปกรณ์ (Component_type)
- *Int GetX(Component)* ทำการอ่านค่าจุดพิกัด X ของอุปกรณ์
- *Int GetY(Component)* ทำการอ่านค่าจุดพิกัด Y ของอุปกรณ์
- *Setall(Component,Comp_id,Comp_type,Comp_xpos,Comp_ypos)* ทำการเซตค่าภายใน Component ทุกค่าให้เป็นไปตามค่าที่กำหนด
- *Setid(Component,Comp_id)* ทำการเซตค่า component_id ให้มีค่าเท่ากับ Comp_id
- *Settype(Component,Comp_type)* ทำการเซตค่า component_type ให้มีค่าเท่ากับ Comp_type
- *Setxpos(Component,Comp_xpos)* ทำการเซตค่า component_loc.xpos ให้มีค่าเท่ากับ Comp_xpos

- *Setypos(Component,Comp_ypos)* ทำการเซตค่า component_loc.ypos ให้มีค่าเท่ากับ Comp_ypos

❑ ชนิดของข้อมูลแบบโอเปคของ Pin

```
struct  pin
{
        int  pin_id;
        int  component_id;
        int  pin_value;
        int Pin_io;
        int is_terminal;
}
```

Pin มีฟังก์ชันสนับสนุนดังต่อไปนี้

- *Equal(pin1,pin2)* เปรียบเทียบระหว่าง Pin 2 Pin ซึ่ง Pin 2 Pin จะเท่ากันเมื่อ หมายเลขของขาและอุปกรณ์ที่ขานั้นเกาะติดอยู่เท่ากัน (Pin_id และ Component_id เท่ากัน)
- *Int GetID(pin)* ทำการอ่านค่าหมายเลขขา (Pin_id)
- *Int GetComp(pin)* ทำการอ่านค่าหมายเลขของอุปกรณ์ที่ขานั้นเกาะติดอยู่ (Component_id)
- *Int GetValue(pin)* ทำการอ่านค่าทางลอจิกของขานั้น ๆ (pin_value)
- *Int Getio(pin)* ทำการอ่านค่าความเป็นอินพุต หรือเอาท์พุตของขา (Pin_io)
- *CheckTerm(pin)* ทำการตรวจสอบว่าขาดังกล่าวเป็นอินพุตหรือเอาท์พุตของวงจรหรือไม่ (is_teminal)
- *Setall(pin,pin_id,comp_id,pin_value,pin_io,pin_term)* จะทำการเซตค่าภายใน pin ให้เป็นไปตามค่าที่กำหนด โดย pin_io ค่า 0 คืออินพุต ค่า 1 เป็นเอาท์พุต และค่า Pin_term ค่า 0 หรือ ไม่เป็นเทอร์มินอล ค่า 1 ค่าขานั้นมีสถานะเป็นเทอร์มินอล
- *Setid(pin,pin_id)* ทำการเซตค่า pin_id
- *Setcomp(pin,comp_id)* ทำการเซตค่า component_id
- *Setvalue(pin,pin_value)* ทำการเซตค่า pin_value
- *Setio(pin,pin_io)* ทำการเซตค่า pin_io
- *Setterm(pin,pin_term)* ทำการเซตค่า is_terminal

- ชนิดของข้อมูลแบบโอเปคของ Wire

```
struct   pin
{
        int wire_id;
        int first_comp_id;
        int first_pin_id;
        int second_comp_id;
        int second_pin_id;
        int wire_value;
}
```

Wire มีฟังก์ชันสนับสนุนดังต่อไปนี้

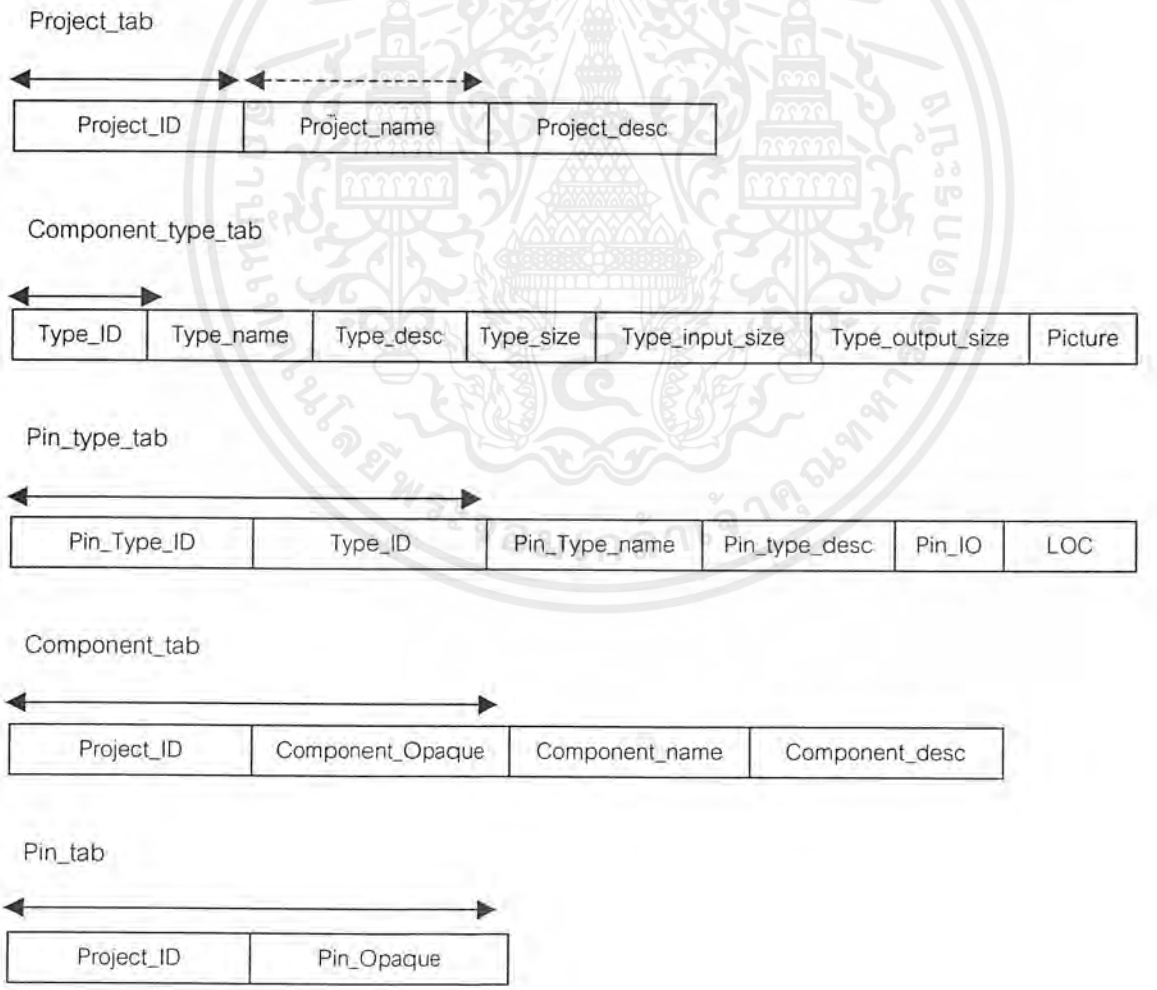
- *Equal(wire1,wire2)* เปรียบเทียบสายสัญญาณว่าเท่ากันหรือไม่ ซึ่ง Wire จะเท่ากันเมื่อหมายเลขของสายสัญญาณ (Wire_ID) มีค่าเท่ากัน
- *Int GetID()* ทำการอ่านค่าหมายเลขของสายสัญญาณ (Wire_ID)
- *Int GetFirstComp()* ทำการอ่านค่าของคอมโพเนนท์ต้นทาง (first_comp_id)
- *Int GetFirstPin()* ทำการอ่านค่าของพินต้นทาง (first_pin_id)
- *Int GetSecComp()* ทำการอ่านค่าของคอมโพเนนท์ปลายทาง (sec_comp_id)
- *Int GetSecPin()* ทำการอ่านค่าของพินปลายทาง (sec_pin_id)
- *Int GetValue()* ทำการอ่านค่าทางลอจิกของสายสัญญาณ (wire_value)
- *Setall (wire,wire_id,wire_firstcomp,wire_firstpin,wire_seccomp,wire_secpin,wire_value)* ทำการเซตค่าทั้งหมดใน Wire ให้เป็นไปตามค่าที่กำหนด
- *Setid(wire,wire_id)* ทำการเซตค่า wire_id
- *Setfirstcomp(wire,wire_firstcomp)* ทำการเซตค่า first_comp_id
- *Setfirstpin(wire,wire_firsttpin)* ทำการเซตค่า first_pin_id
- *Setseccomp(wire,wire_seccomp)* ทำการเซตค่า second_comp_id
- *Setsecpin(wire,wire_secpin)* ทำการเซตค่า second_pin_id
- *Setvalue(wire,wire_value)* ทำการเซตค่า wire_value

8.4 ตารางความจริง

ตารางความจริง (Truth Table) จะเก็บอยู่ในตาราง Truth_tab ซึ่งจะแจกแจงความเป็นไปได้ทุกกรณีของสัญญาณอินพุต และแสดงสัญญาณเอาท์พุตของอุปกรณ์ชนิดนั้น ๆ ซึ่งในโครงงานนี้อนุญาตให้สัญญาณเอาท์พุตมีได้เพียงสัญญาณเดียว เพื่อสะดวกในการจำลองการทำงาน ในตาราง Truth_tab จะมีแอตทริบิวต์ต่าง ๆ ดังต่อไปนี้

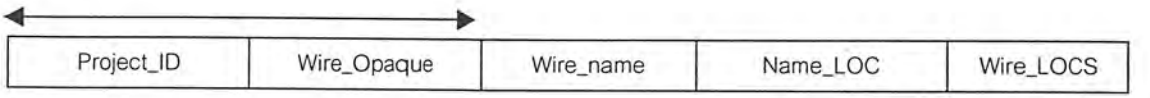
- *Type_id* อ้างถึงหมายเลขของอุปกรณ์บนตาราง Component_type_tab
- *Case* คือตัวเลขแสดงแสดงถึงกรณีต่าง ๆ เพื่อรวมเข้ากับ Type_id ให้เป็นคีย์หลัก
- *Input_list* เป็นชุดของสัญญาณอินพุต ซึ่งจัดเก็บเป็นตัวแปรคอลเลคชัน ภายในมีสัญญาณเป็น 0 หรือ 1
- *Outputvalue* เป็นค่าของเอาท์พุตที่ได้

เมื่อทำการสร้างชนิดของข้อมูลแบบโอเปค และฟังก์ชันสนับสนุนเสร็จแล้ว สามารถสร้างตารางเพื่อการเก็บข้อมูลจริง ๆ ในฐานข้อมูล โดยมีโครงสร้างของตารางที่จะเก็บในฐานข้อมูลดังรูปที่ 8.7

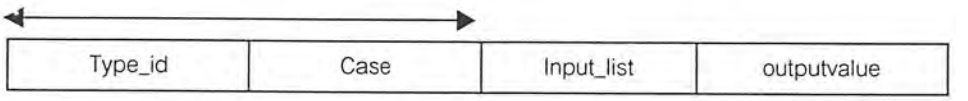


รูปที่ 8.7 โครงสร้างตารางบนฐานข้อมูล

Wire_Tab



Truth_tab



รูปที่ 8.7 โครงสร้างตารางบนฐานข้อมูล (ต่อ)

บทที่ 9
รูทีนใช้งานและการจำลองการทำงาน

การจำลองการทำงาน (Simulation) เป็นการจำลองการไหลของกระแสไฟฟ้า เกิดสัญญาลอจิก ณ จุดต่าง ๆ บนวงจร ซึ่งวงจร 1 วงจร จะถูกประกอบขึ้นจาก คอมโพเนนท์ , พิน และ ไวร์ การจำลองการทำงานจะเป็นการอัพเดตค่าลอจิก ซึ่งเป็นคุณสมบัติหนึ่งของ ไวร์ และ พิน ซึ่งบรรจุไว้ในชนิดของโอเปค การอ่านค่าลอจิก และ การเปลี่ยนแปลงค่าลอจิกของแต่ละ ไวร์ หรือ พิน สามารถทำได้โดยการเรียกใช้ฟังก์ชันสนับสนุนการทำงานซึ่งสนับสนุนชนิดของโอเปคทั้งสองชนิด

สำหรับฟังก์ชันในการจำลองการทำงาน จะถูกเขียนขึ้นด้วยภาษาวิชวลเบสิกที่ฝั่งไคลเอนท์เพื่อลดการทำงานของเซิร์ฟเวอร์ และด้วยเหตุผลของการจัดการหน่วยความจำ ซึ่งวิชวลเบสิกทำได้ดี และทำความเข้าใจได้ง่าย การเรียกใช้ฟังก์ชัน สามารถทำได้โดยการเรียกใช้ฟังก์ชันชื่อ Simall(Proj_id) ซึ่ง Proj_id จะเป็นหมายเลขของโปรเจ็คที่ต้องการจำลองการทำงาน และในฟังก์ชัน Simall จะมีการเรียกใช้ฟังก์ชันอีกฟังก์ชันหนึ่งคือ Sim1comp (Proj_id,Compid) ซึ่งจะเป็นฟังก์ชันในการจำลองการทำงานสำหรับคอมโพเนนท์หมายเลข Compid ในโปรเจ็กหมายเลข Proj_id เมื่อมีค่าอินพุตใด ๆ

ในวงจรดิจิตอลหนึ่งๆ แต่ละคอมโพเนนท์ จะมีพินที่สัมพันธ์กันเกาะติดอยู่ ซึ่งพินจะมี 2 ประเภท คือที่เป็นเทอร์มินอล และที่ไม่เป็นเทอร์มินอล พินที่เป็นเทอร์มินอลคือพินที่จะไม่ถูกต่อกับสายสัญญาณหรืออุปกรณ์ใด ๆ ผู้ใช้สามารถป้อนค่าทางลอจิกได้ถ้าเป็นพินอินพุต แต่สำหรับค่าเอาท์พุตก็สามารถที่จะดูค่าทางลอจิกได้ เมื่อพินถูกต่อเข้าด้วยกันด้วยสายสัญญาณหนึ่ง ๆ ขาจะถูกพิจารณาว่า ควรจะถูกเปลี่ยนแปลงไม่ให้เป็นเทอร์มินอลหรือไม่ แล้วแต่ว่าจะถูกต่อกันในลักษณะใด ซึ่งจะเป็นหน้าที่ของแอพพลเคชันฝั่งไคลเอนท์ที่จะต้องทำการตรวจสอบทุกครั้งที่มีการลากสายสัญญาณ

โปรแกรมภาษาวิชวลเบสิกสามารถที่จะทำการเรียกใช้ SQL สเตจเมนต์ เพื่อทำการอ่านค่าทางลอจิกของขาและสายสัญญาณใด ๆ และยังสามารถเรียกใช้ฟังก์ชันที่เก็บอยู่ในเซิร์ฟเวอร์ ซึ่งฟังก์ชันอาจจะถูกเขียนขึ้นด้วยภาษาซี ซึ่งถูกเพิ่มเข้ามาผ่านดาต้าเบลดโมดูลและฟังก์ชันที่ถูกเขียนขึ้นด้วยภาษา SPL ที่ถูกเพิ่มเข้าสู่ดาต้าเบสเซิร์ฟเวอร์ โดยผ่านดาต้าเบลดโมดูล หรือผ่านทางการเขียนลงไปโดยตรงโดยใช้คำสั่ง CREATE PROCEDURE หรือ CREATE FUNCTION

9.1 รูทีนใช้งาน

สำหรับในโปรเจ็คนี้ ฟังก์ชันที่ถูกเขียนขึ้นด้วยภาษาซี จะมีเฉพาะฟังก์ชันสนับสนุนการทำงานที่สนับสนุนชนิดของข้อมูลแบบโอเปคเท่านั้น ซึ่งอธิบายไปแล้วในหัวข้อของการสร้างชนิดของข้อมูลแบบโอเปค แต่ถ้าเป็นฟังก์ชันที่ใช้งานอย่างอื่น ก็จะถูกเขียนด้วยภาษา SPL จะเขียนได้ง่ายกว่า และสามารถเรียกใช้คำสั่ง SQL ได้โดยตรง ซึ่งโปรซีเดอร์ที่เขียนขึ้นด้วยภาษา SPL มีดังต่อไปนี้

9.1.1 InsNewPin(Project_id,Component)

เป็นโปรซีเดอร์ที่ใช้เป็นทริกเกอร์แอคชัน ในกรณีที่มีการแทรกคอมโพเนนท์ใหม่เข้าไปตาราง Comp_tab ซึ่งจะมีหน้าที่ในการ Insert พินทุก ๆ พิน ของอุปกรณ์ชนิดนั้น ลงไปในตาราง Pin_tab ซึ่งข้อมูลของพินที่จะแทรกจะได้มาจากตาราง Pin_type_tab ซึ่งจะแจกแจงว่าอุปกรณ์ชนิดนั้นมีพินกี่พิน แต่ละพินมีหมายเลขอะไร เป็นพินอินพุต หรือเอาท์พุต โดยโปรซีเดอร์นี้จะให้ค่าเริ่มต้นค่าลอจิก และค่าเทอร์มินอลของพิน โดยจะให้มีค่าลอจิกเป็น 0 และให้เป็นพินเทอร์มินอล

9.1.2 DelRelatePin(Project_id,Component)

ใช้เป็นทริกเกอร์แอคชัน กรณีที่มีการลบคอมโพเนนท์หนึ่ง ๆ ออกจากตาราง Comp_tab จะมีการลบพินทุกพินที่เป็นของคอมโพเนนท์นั้น ออกจากตาราง Pin_tab โดยอัตโนมัติ

9.1.3 PinUpdateValue (Project_id,Component_id,Pin_id,Value)

เป็นโปรซีเดอร์ที่จะอัพเดตค่าลอจิกของพินหมายเลข Pin_id ที่อยู่บนคอมโพเนนท์ที่มีหมายเลขขาเท่ากับ Component_id บนโปรเจ็คหมายเลข Project_id ให้มีค่าทางลอจิกเท่ากับ Value ซึ่ง โปรซีเดอร์นี้จะมีการเรียกใช้ฟังก์ชันสนับสนุนการทำงาน setvalue(Comp_opaque,value) ของโอเปค Component ซึ่งค่าที่เซตจะมีเพียง 2 ค่าเท่านั้นคือ 0 หรือ 1

9.1.4 PinUpdateTerm (Project_id,Component_id,Pin_id,Term)

ทำหน้าที่อัพเดตสถานะความเป็นเทอร์มินอลของขาหมายเลข Pin_id ที่อยู่บนคอมโพเนนท์ ที่มีหมายเลขเท่ากับ Component_id บนโปรเจ็กหมายเลข Project_id ให้มีค่าเท่ากับ Term ซึ่งค่าเทอร์มินอล 0 หมายถึง ขานั้นไม่มีความเป็นเทอร์มินอล และถ้าเป็น 1 แสดงว่าขานั้นเป็นเทอร์มินอล

9.1.5 WireUpdateValue (Project_id,Wire_id,Value)

จะทำหน้าที่อัพเดตค่าลอจิกของสายสัญญาณหมายเลข Wire_id ของโปรเจ็คหมายเลข Project_id ให้มีค่าเท่ากับ Value ซึ่งการเชื่อมต่อสายสัญญาณเข้ากับขาหนึ่ง ๆ จะมีการอัพเดตให้ลอจิกของสายเท่ากับลอจิกของสายสัญญาณขณะที่มีการจำลองการทำงาน ซึ่งในขณะที่มีการวาดวงจร ค่าลอจิกจะยังไม่ได้มีการอัพเดตให้ถูกต้อง

9.1.6 PinGetValue (Project_id,Component_id,Pin_id)

มีหน้าที่อ่านค่าของลอจิกของขาหมายเลข Pin_id ที่เกาะติดกับคอมโพเนนท์หมายเลข Component_id ของโปรเจ็คหมายเลข Project_id ส่งกลับสู่ผู้เรียก ซึ่งค่าที่ส่งกลับจะมีค่าเป็น 0 หรือ 1 ซึ่งภายใน การทำงานจะมีการเรียกใช้ฟังก์ชันสนับสนุนการทำงาน getvalue ของชนิดข้อมูลโอเปค Pin

9.1.7 PinGetTerm (Project_id,Component_id,Pin_id)

มีหน้าที่อ่านค่าสถานะความเป็นเทอร์มินอลของขานั้น ๆ ภายในมีการเรียกใช้ฟังก์ชันสนับสนุนการทำงาน checkterm ของโอเปค Pin ซึ่งค่าที่ส่งกลับจากฟังก์ชันนี้เป็น Integer มีค่า 0 หรือ 1

9.1.8 WireGetValue (Project_id,Wire_id)

จะส่งกลับค่าลอจิกของสายสัญญาณหมายเลข Wire_id ของโปรเจ็คหมายเลข Project_id ค่าที่ส่งกลับเป็น 0 หรือ 1 เท่านั้น ซึ่งจะมีการเรียกใช้ฟังก์ชันสนับสนุน getvalue ของชนิดข้อมูลโอเปค Wire

9.1.9 InsertNewComp

(Project_id,Comp_id,Comp_type,Xpos,Ypos,Comp_name,Comp_desc)

เป็นฟังก์ชันที่จะทำการเพิ่มคอมโพเนนท์ใหม่เข้าไปในตาราง Comp_tab อยู่ในโปรเจ็คที่มีหมายเลขเท่ากับ Project_id มีหมายเลขคอมโพเนนท์เท่ากับ Comp_id มีชนิดของอุปกรณ์เท่ากับ Comp_type ตำแหน่งของอุปกรณ์บนหน้าจอเท่ากับ (Xpos,Ypos) มีชื่อของอุปกรณ์เท่ากับ Comp_name มี Description เท่ากับ Comp_desc ซึ่งฟังก์ชันนี้จะทำให้สะดวกในการเขียนโปรแกรมในฝั่งไคลเอนท์

สำหรับค่าที่ส่งกลับ ถ้าสามารถแทรกอุปกรณ์ใหม่ได้สำหรับจะมีการส่งค่า 0 กลับมา แต่ถ้า โปรเจ็คหมายเลขกำหนดในอาร์กิวเมนท์ไม่มีอยู่ในฐานข้อมูล จะส่งกลับค่า -998 กลับมา แต่ถ้าคอมโพเนนท์หมายเลขนั้นมีอยู่แล้วในตาราง Comp_tab จะส่งกลับค่า –999 กลับมา ซึ่งถ้าเราไม่สามารถแทรกอุปกรณ์ได้สำเร็จ จะไม่มีผลต่อฐานข้อมูลที่มีอยู่เดิม

เมื่อทำฟังก์ชันนี้เสร็จ ขาที่สัมพันธ์กันจะถูกแทรกเข้าไปในตาราง Pin_tab โดยอัตโนมัติ เนื่องจากมีการสร้างทริกเกอร์ที่จะคอยตรวจสอบการแทรกเข้าไปในตาราง Comp_tab ไว้แล้ว

9.1.10 InsertNewWire

(Project_id,Wire_id,First_comp,First_pin,Second_comp,Second_pin)

จะทำการแทรกสายสัญญาณที่มีหมายเลขเท่ากับ Wire_id เข้าไปในตาราง Wire_tab ของโปรเจ็ค Project_id โดยสายสัญญาณหนึ่ง ๆ จะมีขาต้นทางและปลายทาง โดยขาสัญญาณต้นทางจะกำหนดไว้ในตัวแปร First_comp และ First_pin และขาสัญญาณต้นทางจะถูกกำหนดไว้ในตัวแปร Second_comp และ Second_pin ซึ่งกำหนดให้ค่าของสายสัญญาณเริ่มต้นเมื่อทำการแทรกเข้าไปมีค่าเท่ากับ 0

สำหรับค่าข้อมูลที่จะส่งกลับ จะเป็นชนิดจำนวนเต็ม จะส่งกลับค่า 0 เมื่อการแทรกทำได้สำเร็จ จะส่งกลับค่า –998 เมื่อโปรเจ็คนั้นไม่มีในฐานข้อมูล และจะส่งกลับค่า –999 เมื่อสายสัญญาณมีอยู่แล้วในตาราง Wire_tab

9.1.11 InsertWireLoc (Project_id,Wire_id,Xpos,Ypos)

เนื่องจากสายสัญญาณ 1 สายที่เชื่อมต่อระหว่างขาสัญญาณ 2 ขา ไม่ได้เป็นเส้นตรงเส้นเดียวเสมอไป สายสัญญาณสามารถที่จะหักเหเป็นมุมฉาก เพื่อให้วงจรเกิดความสวยงาม และใกล้เคียงกับวงจรดิจิตอลจริง ๆ ดังนั้น สายสัญญาณหนึ่ง ๆ จึงมีการเก็บโคออร์ดิเนตของสายสัญญาณ ซึ่งจะมีอย่างน้อย 2 จุดเป็นต้นไป สำหรับโคออร์ดิเนตจะมีชนิดเป็น Loc ซึ่งเป็นชนิดของข้อมูลแบบโอเปคภายในมีการเก็บค่า X และ ค่า Y การอ่านค่าแต่ละค่าทำได้ผ่านทางฟังก์ชันสนับสนุนการทำงาน getxpos() และ getypos() การเซตค่าแต่ละค่าทำได้โดยผ่านฟังก์ชันสนับสนุนการทำงาน setxpos() และ setypos()

สำหรับชุดของจุดโคออร์ดิเนตจะเก็บอยู่ในตัวแปรชนิดคอลเลคชันแบบลิสต์ (LIST) ซึ่งจะไม่สามารถเป็น NULL ได้ การเพิ่มจุดโคออร์ดินเนตค่าใหม่ สามารถทำได้โดยการแทรกค่าลงในตัวแปรคอลเลคชันได้โดยตรง

สำหรับค่าที่ส่งกลับ ค่า 0 แสดงว่าเพิ่มโคออร์ดิเนตสำเร็จ ค่า –998 แสดงว่าไม่มีโปรเจ็คนี้อยู่ในฐานข้อมูล และค่า –997 แสดงว่า ไม่มีสายสัญญาณหมายเลข Wire_id อยู่ในฐานข้อมูล จะไม่สามารถแทรกโคออร์ดิเนตได้สำเร็จ

9.1.12 ClearWireLoc (Project_id,Wire_id)

เป็นฟังก์ชันที่จะทำการลบทุก ๆ จุดโคออร์ดิเนตออกจากสายสัญญาณ เพื่อทำการเพิ่มค่าใหม่เข้าไป ซึ่งทำสำเร็จจะส่งกลับค่า 0 ถ้าไม่มีโปรเจ็คนี้จะทำการส่งกลับค่า –998 แต่ถ้าไม่มีไวร์หมายเลขนี้อยู่ในตาราง Wire_tab จะส่งค่า –997 กลับมา

9.1.13 GetWireLoc(Project_id,Wire_id)

ฟังก์ชันนี้มีประโยชน์มากสำหรับการเขียนโปรแกรมแอพพลิเคชัน เนื่องจากจุดโคออร์ดิเนตของสายสัญญาณ ถูกจัดเก็บไว้ในตัวแปรแบบคอลเลคชัน ซึ่งวิชวลเบสิกจะไม่เข้าใจตัวแปรชนิดนี้ ฟังก์ชันนี้จะส่งกลับข้อมูลที่เป็น VARCHAR(255) ซึ่งวิชวลจะเข้าใจในรูปแบบของสตริง

ค่าที่ส่งกลับมาจะอยู่ในรูปแบบของสตริงที่สื่อความหมาย ดังนี้

" 50&60 70&150 150&160 "

ซึ่งสื่อความหมายว่า จุดโคออร์เนตของสายสัญญาณมี 3 จุด ซึ่งมีโคออร์ดิเนต (50,60) (70,150) และ (150,160) ตามลำดับ ซึ่งโปรแกรมแอพพลิเคชันจะทำการลากเส้นตามลำดับแต่ละจุด

9.1.14 tosequence(Listvar)

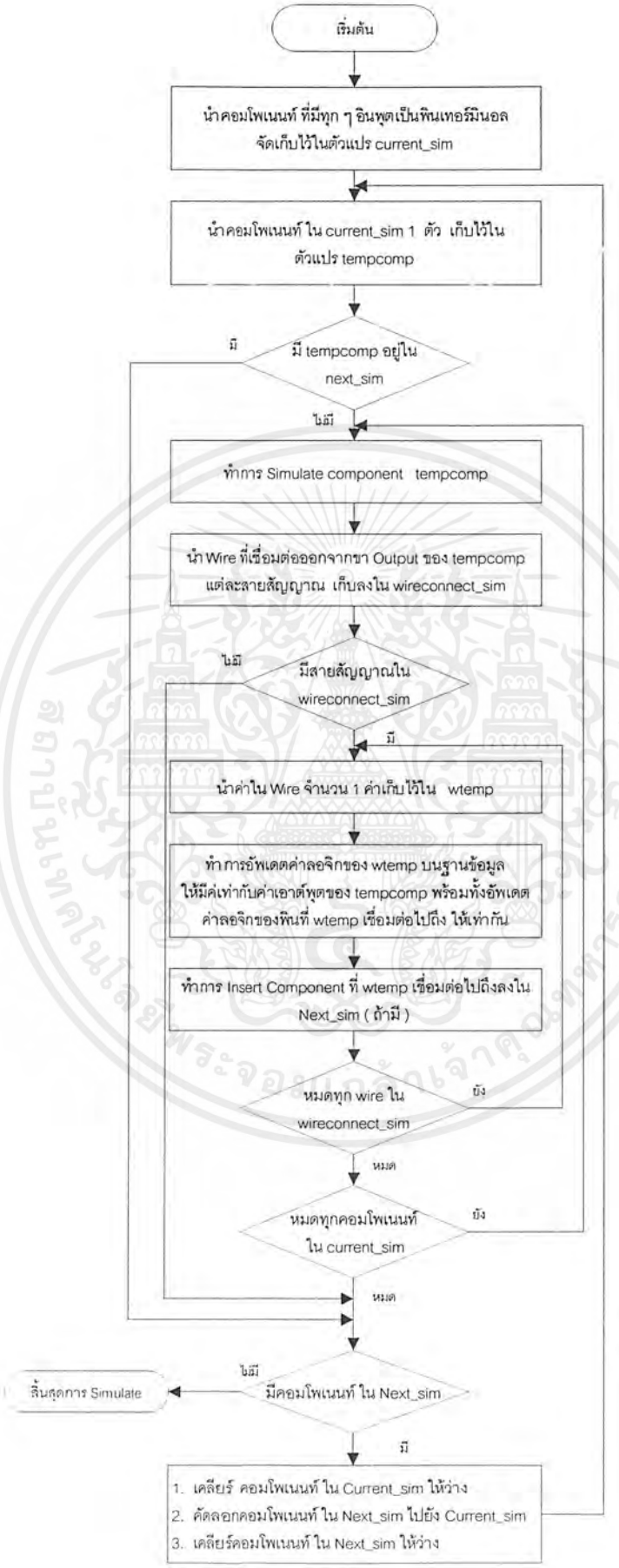
เป็นฟังก์ชันที่ที่จะทำการแปลงตัวแปรที่เป็ลิสต์ให้อยู่ในรูปของสตริง เช่นข้อมูลในลิสต์เป็น {10,20,30,40} จะถูกแปลงเป็น "10 20 30 40" ซึ่งจะมีประโยชน์ในการเรียกใช้ข้อมูลในลิสต์จากโปรแกรมแอพพลิเคชัน

9.2 ฟังก์ชันจำลองการทำงานของวงจรดิจิตอล

การจำลองการทำงานจะถูกทำที่แอพพลิเคชันฝั่งไคลเอนท์ ถูกเขียนโดยวิชวลเบสิก และจะมีการอ่านข้อมูลจากฐานข้อมูล และมีการอัพเดตข้อมูลในฐานข้อมูลโดยตรง โดยอาศัยดาต้าไดเรกเตอร์ช่วยในการเขียนโปรแกรม สำหรับตัวแปรแบบลิสต์บนฐานข้อมูล ในวิชวลเบสิกจะใช้ตัวแปรแบบคอลเลคชันที่วิชวลเบสิกมีให้ ซึ่งสามารถจัดเก็บออบเจ็คต่าง ๆ ไว้ภายใน สามารถเพิ่มและลบได้โดยสะดวก ซึ่งคอลเลคชันนี้ไม่ใช่ข้อมูลชนิดเดียวกับคอลเลคชัน ในอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์แต่มีหลักการทำงานที่คล้ายกันมาก

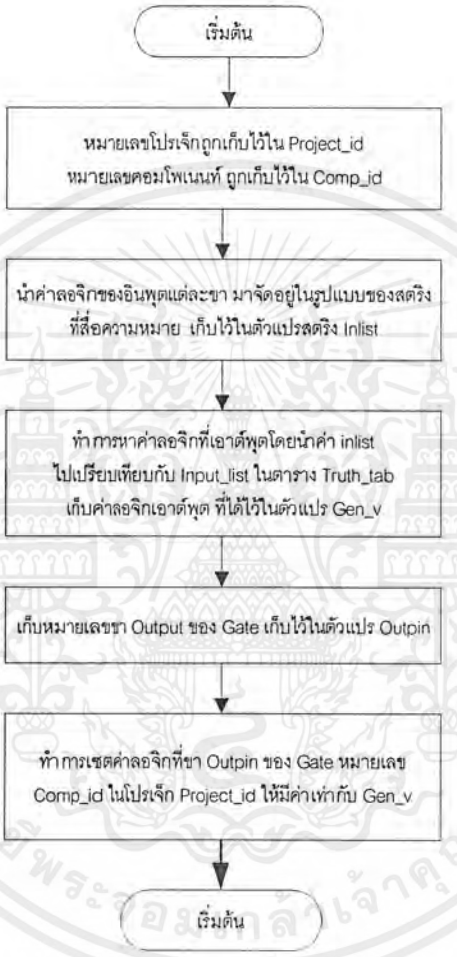
การเรียกใช้ฟังก์ชันเพื่อทำการจำลองการทำงานทำได้โดยเรียกฟังก์ชัน Simall (project_id) ซึ่งในขั้นตอนการจำลองการทำงานจะมีตัวแปร Current_sim, Next_sim และ Wireconnect_sim เป็นตัวแปรแบบคอลเลคชัน โดยเริ่มต้นตัวแปรเหล่านี้จะไม่มีข้อมูลอยู่ภายใน

สำหรับขั้นตอนการทำงานของฟังก์ชัน Simall แสดงได้ดังรูปที่ 9.1



รูปที่ 9.1 ขั้นตอนการจำลองการทำงาน

สำหรับการจำลองการทำงาน 1 คอมโพเนนท์จะมีหน้าที่ในการประมวลผลสัญญาณอินพุตทุกขา และทำการเปรียบเทียบว่าตรงตามเงื่อนไขใด ในตาราง Truth_tab และจะทำการเซตค่าเอาท์พุตให้ตรงกับค่าในตารางความจริง สำหรับค่าที่ส่งกลับจะมีการส่งค่า 0 กลับมา เมื่อทำการจำลองการทำงานได้สำเร็จ แต่ถ้าเกิดข้อผิดพลาดขึ้น ก็จะส่งกลับค่า -999 กลับมา รายละเอียดของการจำลองการทำงาน 1 คอมโพเนนท์ แสดงได้ดังรูปที่ 9.2



รูปที่ 9.2 การจำลองการทำงาน 1 คอมโพเนนท์

บทที่ 10
การออกแบบและใช้งานแอพพลิเคชั่นฝั่งไคลเอนท์

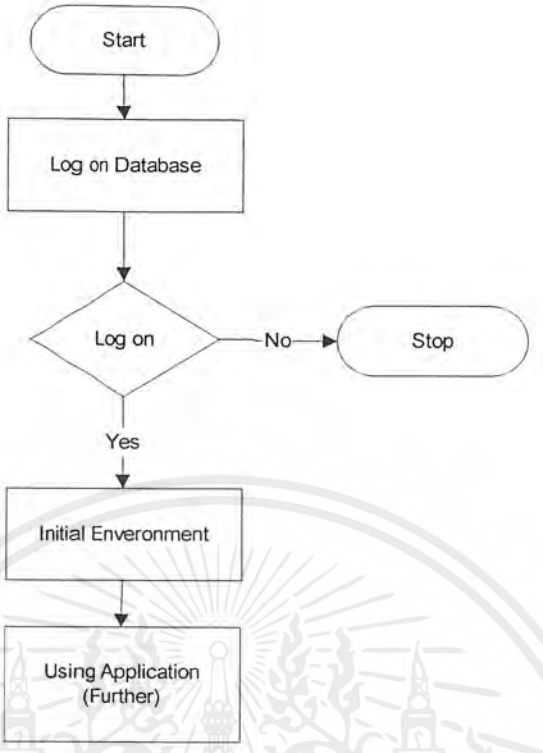
10.1 การออกแบบแอพพลิเคชั่น

ในการออกแบบฐานข้อมูลส่วนเซิร์ฟเวอร์ เราได้มีการใช้ชนิดของข้อมูลแบบใหม่ เช่น ชนิดของข้อมูลแบบโอเปค, Blob เป็นต้น ซึ่งการเข้าถึงข้อมูลเหล่านี้จากไคลเอนท์จำเป็นต้องใช้ดาต้าไดเรกเตอร์ในการเข้าถึงข้อมูล และดาต้าไดเรกเตอร์สนับสนุนการทำงานบนวิชวลเบสิก จึงเลือกใช้วิชวลเบสิกเวอร์ชัน 6 มาใช้ในการสร้างแอพพลิเคชั่น

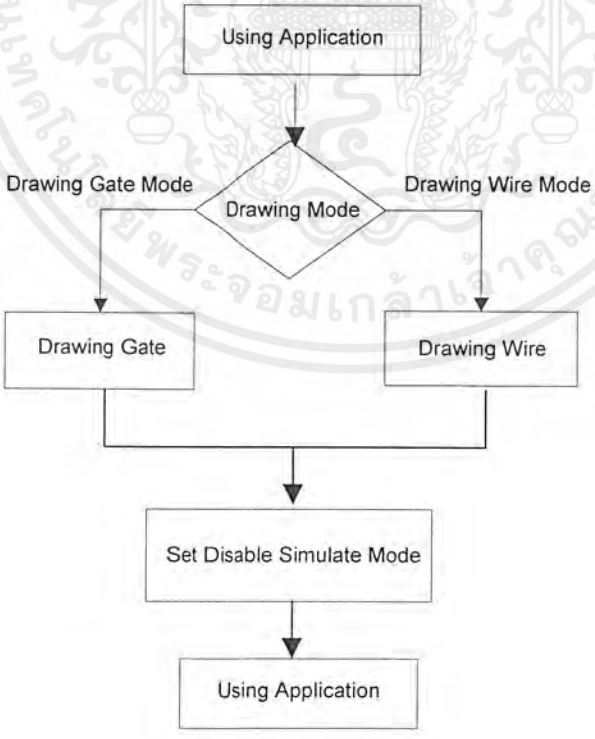
หลังจากศึกษาดาต้าไดเรกเตอร์และวิชวลเบสิกจนสามารถสร้างแอพพลิเคชั่นได้แล้วจึงลำดับการสร้างแอพพลิเคชั่นดังนี้

1. ออกแบบการทำงานซึ่งแบ่งออกเป็นโหมดต่าง ๆ ดังนี้
   1.1 โหมดการวาด (Drawing Mode) แบ่งโหมดย่อยได้เป็น
      - โหมดการวาดเกต (Drawing Gate Mode)
      - โหมดการวาดสายสัญญาณ (Drawing Wire Mode)
   1.2 โหมดจำลองการทำงาน (Simulate Mode)
2. ออกแบบรูปแบบและสร้างแอพพลิเคชั่น
3. ส่วนของการเขียนโปรแกรมแบ่งออกเป็น 4 ส่วนหลักๆ คือ
   3.1 ส่วนเตรียมสภาวะแวดล้อมต่างๆ ของ แอพพลิเคชั่น เช่น การโหลดชื่อของอุปกรณ์
   3.2 ส่วนวาดวงจร เช่น การลากสายสัญญาณหรือการสร้างและลบเกต เป็นต้น
   3.3 ส่วนการเปลี่ยนแปลงสถานะการทำงานและโหมดต่างๆของ แอพพลิเคชั่น เช่น เมื่อหลังจากบันทึกวงจรที่สร้างลงฐานข้อมูลจะสามารถเปลี่ยนไปยังโหมดจำลองการทำงานได้
   3.4 ส่วนที่แอพพลิเคชั่นต้องการติดต่อกับฐานข้อมูล เช่น การบักทึกวงจร , การเปิดวงจรขึ้นมาจากฐานข้อมูล เป็นต้น
4. ทดลองใช้งานและปรับปรุง แอพพลิเคชั่น

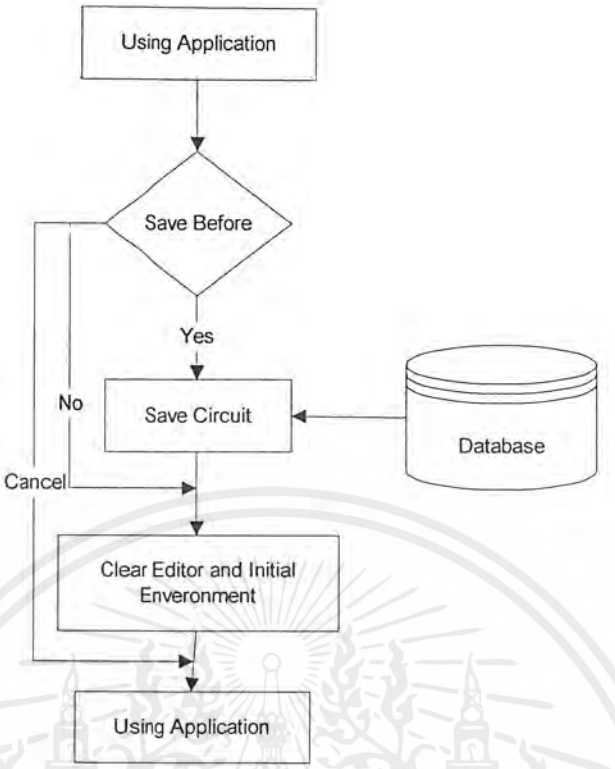
การทำงานใน แอพพลิเคชั่น สามารถแสดงได้ดังโฟวชาร์ต (Flow Chart) ซึ่งแบ่งตามความสามารถที่ แอพพลิเคชั่น มี ดังรูปที่ 10.1



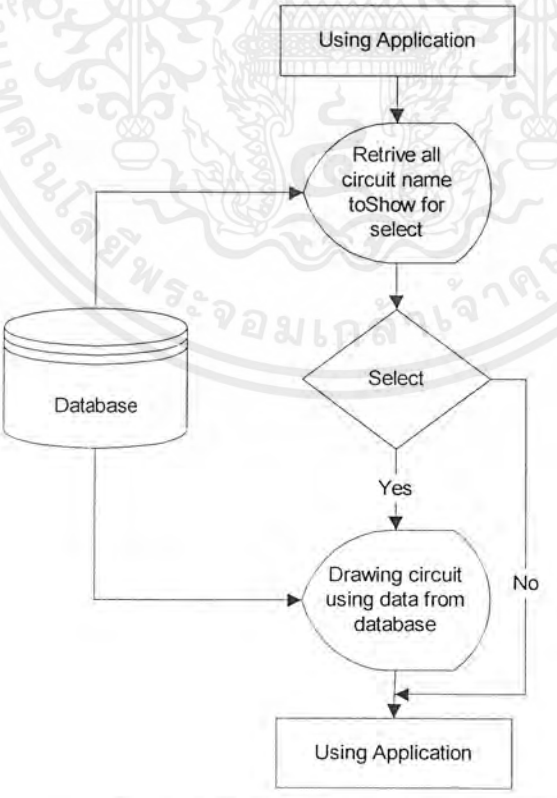
รูปที่ 10.1 โฟวชาร์ตแสดงการเริ่มต้นใช้งานแอพพลิเคชั่น



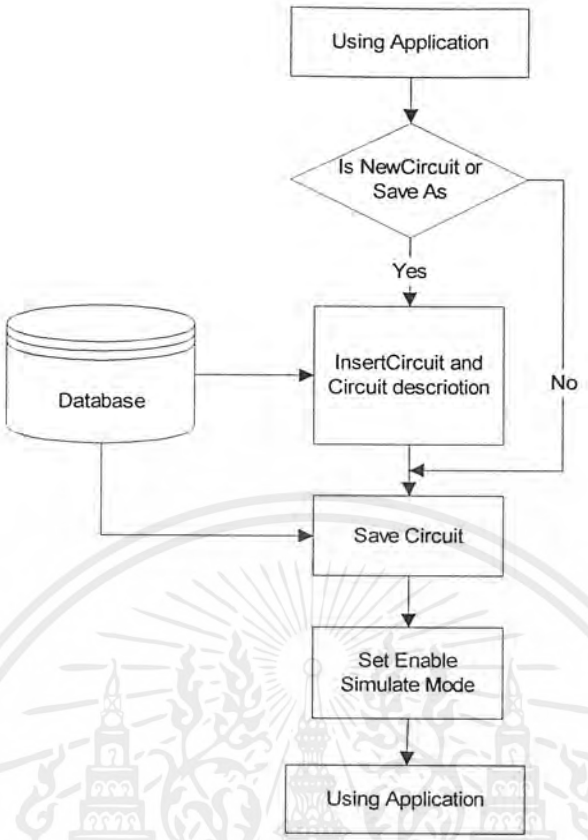
รูปที่ 10.2 โฟวชาร์ตแสดงการสร้างและแก้ไขวงจร



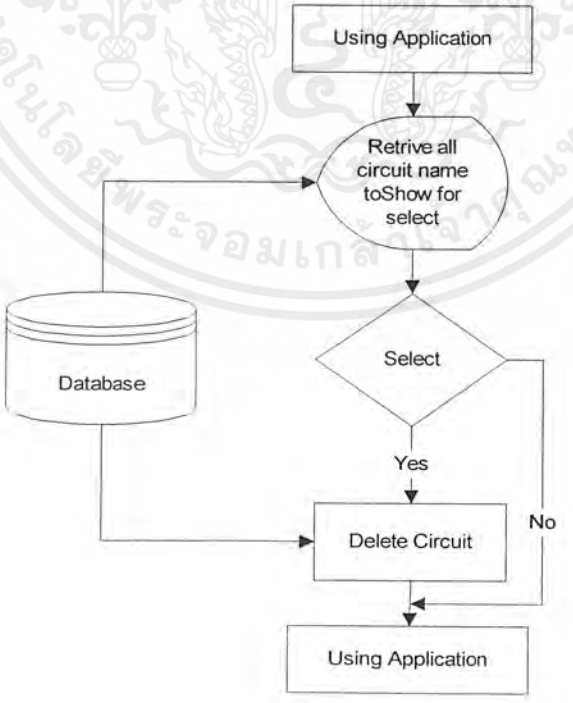
รูปที่ 10.3 โฟวชาร์ตแสดงการสร้างวงจรใหม่



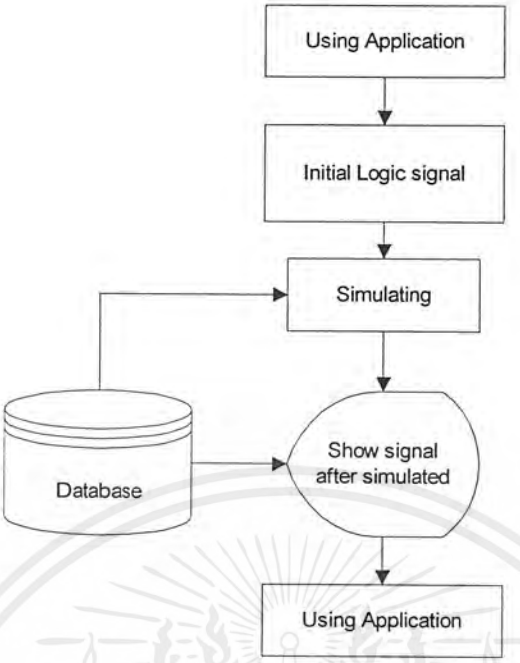
รูปที่ 10.4 โฟวชาร์ตแสดงการเปิดวงจรจากฐานข้อมูล



รูปที่ 10.5 โฟวชาร์ตแสดงการบันทึกวงจรลงฐานข้อมูล



รูปที่ 10.6 โฟวชาร์ตแสดงการลบวงจรจากฐานข้อมูล



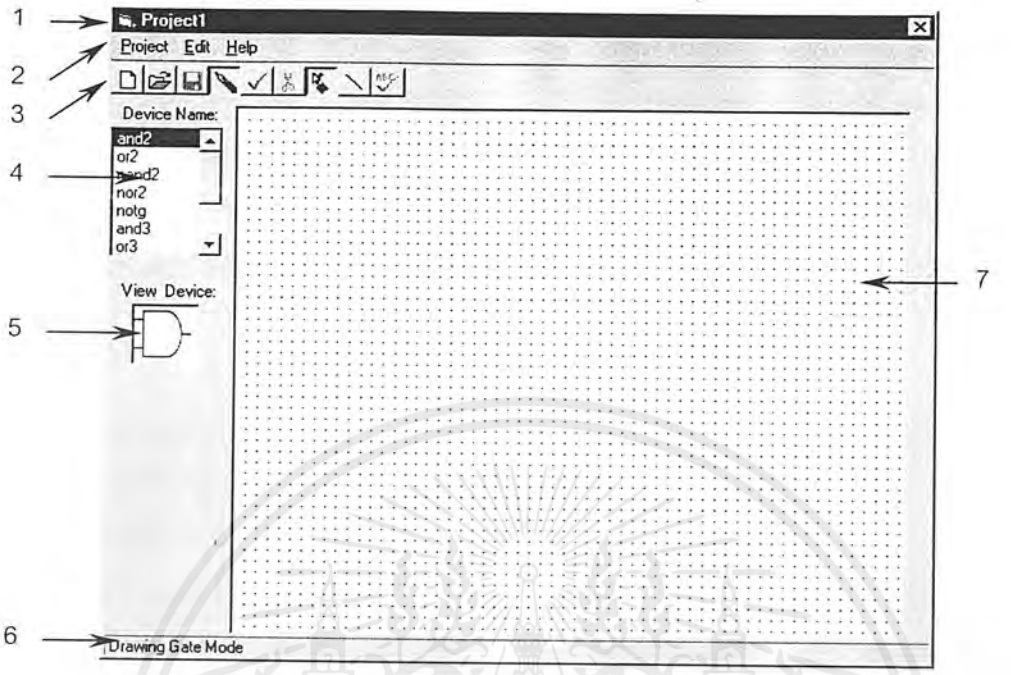
รูปที่ 10.7 โฟวชาร์ตแสดงการจำลองการทำงานของวงจร

10.2 การใช้งานแอพพลิเคชั่น

เมื่อมีการเรียกใช้งานโปรแกรม จะขึ้นหน้าต่างเพื่อเข้าระบบฐานข้อมูล ดังรูปที่ 10.8 หน้าต่างนี้เราจะต้องเข้าสู่ฐานข้อมูลที่เก็บฐานข้อมูลของลอจิกซิมมูเลเตอร์ ถ้าเราเข้าระบบผิดโปรแกรมจะทำการจบการทำงานทันที

รูปที่ 10.8 Log on Database

เมื่อเราเข้าสู่ระบบฐานข้อมูลได้ถูกต้องแล้ว เราจะเข้าสู่หน้าต่างที่จะใช้สร้าง แก้ไข และเก็บข้อมูลวงจรลอจิก พร้อมทั้งจำลองการทำงานได้ด้วย รูปที่ 10.9 แสดงส่วนต่างๆของแอพพลิเคชัน



รูปที่ 10.9 แอพพลิเคชั่น Logic Simulator

แต่ละส่วนในแอพพลิเคชั่น

ส่วนที่ 1 คือ ที่แสดงชื่อของวงจรที่เปิดอยู่ขณะนั้น ปกติเมื่อเปิดโปรแกรมขึ้นมาให้ชื่อ Project1

ส่วนที่ 2 คือ ส่วนของเมนูโปรแกรมซึ่งมีอยู่ 3 ส่วน ประกอบด้วย Project , Edit และ Help

ส่วนที่ 3 คือ ทูลบาร์

ส่วนที่ 4 คือ รายชื่อของอุปกรณ์ที่มีอยู่ในฐานข้อมูล

ส่วนที่ 5 คือ ที่แสดงรูปของอุปกรณ์ที่ถูกเลือกจากส่วนที่ 4

ส่วนที่ 6 คือ แถบสถานะแสดงถึงโหมดที่แอพพลิเคชั่นกำลังทำงาน

ส่วนที่ 7 คือ ส่วนอิดิเตอร์ หรือส่วนที่ใช้ในการวาดวงจรนั่นเอง

10.2.1 การสร้างวงจรและการแก้ไขวงจร (Modify Project)

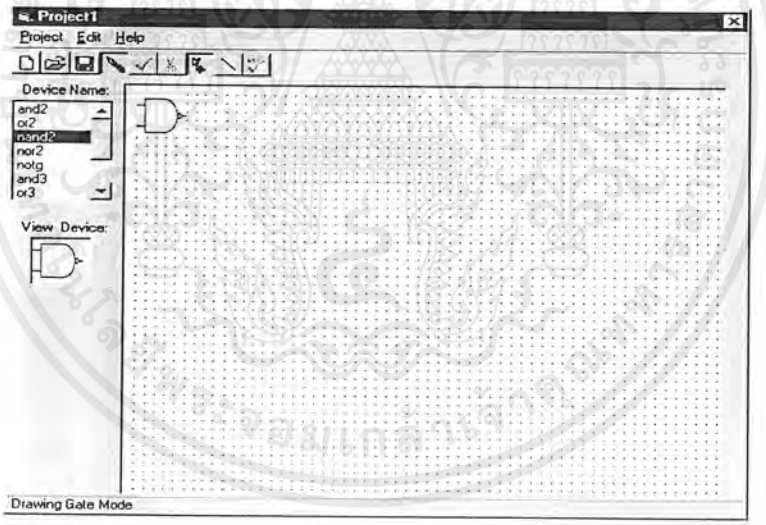
ปกติแล้วอิดิเตอร์ (Editor) จะมีอยู่ 2 โหมด คือ โหมดการวาดวงจร (Drawing Mode) และ โหมดการจำลองการทำงานของวงจร (Simulate Mode) การสร้างวงจรจะอยู่ในโหมดการสร้างแก้ไขวงจร เราสามารถทราบว่าเราอยู่ในโหมดใดสังเกตได้จากสถานะ (Status bar) ที่อยู่ส่วนล่างสุดของหน้าต่างโปรแกรมนั่นเอง

เราสามารถเปลี่ยนโหมดของอิดิเตอร์มายังโหมดการสร้างแก้ไขวงจรนี้ได้โดยคลิ้กที่ทูลบาร์ หรือไปที่ Menu -> Edit -> Drawing Mode หรือกดปุ่มคอนโทรลพร้อมกับปุ่มอักษรดี (Ctrl + D)
ในโหมดการสร้างแก้ไขวงจรมี 2 โหมดย่อยคือ

10.2.1.1 โหมดการวาดเกต

เป็นโหมดเกี่ยวกับการสร้าง ลบ เคลื่อนย้ายเกต เราสามารถเปลี่ยนโหมดให้มาอยู่โหมดนี้ได้ เมื่อคลิ้กที่ทูลบาร์ การสร้าง ลบ และเคลื่อนย้ายไม่สามารถทำในโหมดอื่นได้

- การสร้างเกต เราทำได้โดยเลือกชื่ออุปกรณ์ที่ต้องการจะสร้างจากรายการชื่ออุปกรณ์ (Device Name Listbox) แล้วดับเบิลคลิ้กที่ชื่อนั้นหรือดับเบิลคลิ้กที่ช่องแสดงรูปเกต (View Device) แล้วเกตตัวนั้นจะไปปรากฏที่มุมซ้ายบนของอิดิเตอร์ ดังรูปที่ 10.10



รูปที่ 10.10 การทำงานหลังจากเลือกอุปกรณ์แล้ว

- การลบเกตสามารถทำได้โดยคลิ้กเกตที่เราต้องการลบ แล้วคลิ้กที่ทูลบาร์ ถ้าเกตที่ลบมีสายไวร์ (Wire) ต่ออยู่โปรแกรมจะทำการลบไวร์เส้นนั้นอัตโนมัติด้วย
- การเคลื่อนย้ายเกต สามารถเคลื่อนย้ายเกตที่ไม่ได้ต่อสายไวร์เลยเท่านั้น ทำได้โดยคลิ้กเมาส์ค้างที่เกตที่ต้องการเคลื่อนย้าย แล้วลากเมาส์เมื่ออยู่ในตำแหน่งที่ต้องการจึงปล่อยปุ่มเมาส์ เกตจะเคลื่อนย้ายมายังตำแหน่งที่เราปล่อยนั่นเอง

10.2.1.2 โหมดการวาดสายสัญญาณ

เป็นโหมดสำหรับลากสายไวร์ เราสามารถเปลี่ยนโหมดให้มาอยู่โหมดนี้ได้เมื่อคลิ้กที่ทูลบาร์

- การลากสายไวร์ การลากทำได้โดยคลิ้กที่ขาของเกตเมื่อลากเมาส์จะเห็นว่ามีสายไวร์เกิดขึ้น เมื่อต้องการลากสายต่อไปในแนวอื่นให้คลิ้กลงบนอิดิเตอร์จะพบว่าเมื่อลากเมาส์ต่อไปจะมีสายไวร์เกิดขึ้นตามเมาส์โดยมีจุดเริ่มต้นใหม่ที่ตำแหน่งที่คลิ้กเอาไว้ การลากไวร์จะสิ้นสุดเมื่อคลิ้กที่ขาของเกตที่เป็นอินพุต

  ลักษณะของการลากไวร์ที่สามารถเชื่อมต่อได้มีดังนี้

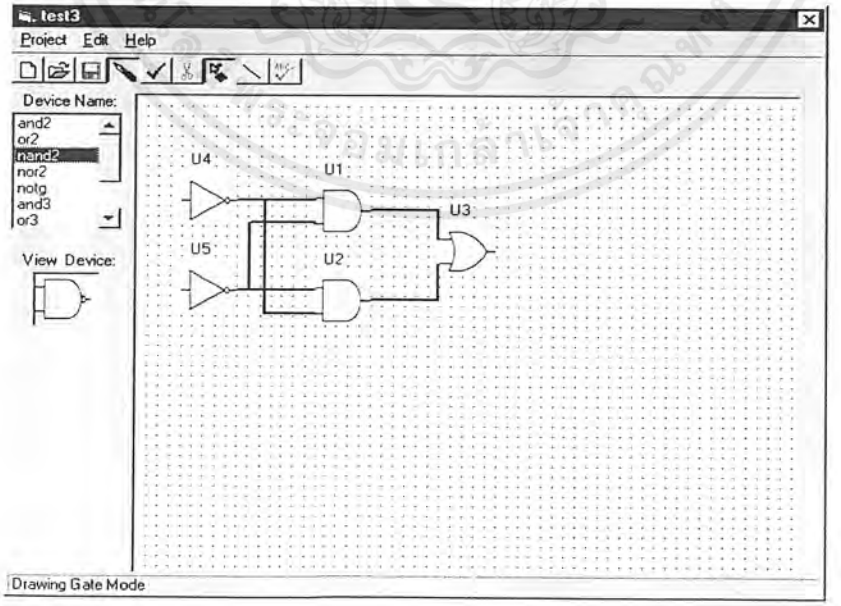
  - อินพุตกับอินพุต
  - เอาต์พุตกับอินพุต

- การลบสายไวร์ สามารถทำได้โดยคลิ้กที่สายที่ต้องการลบแล้วคลิ้กที่ทูลบาร์ แต่ถ้ามีสายไวร์สองเส้นทับกันอยู่ ณ ตำแหน่งนั้นจะไม่สามารถลบได้

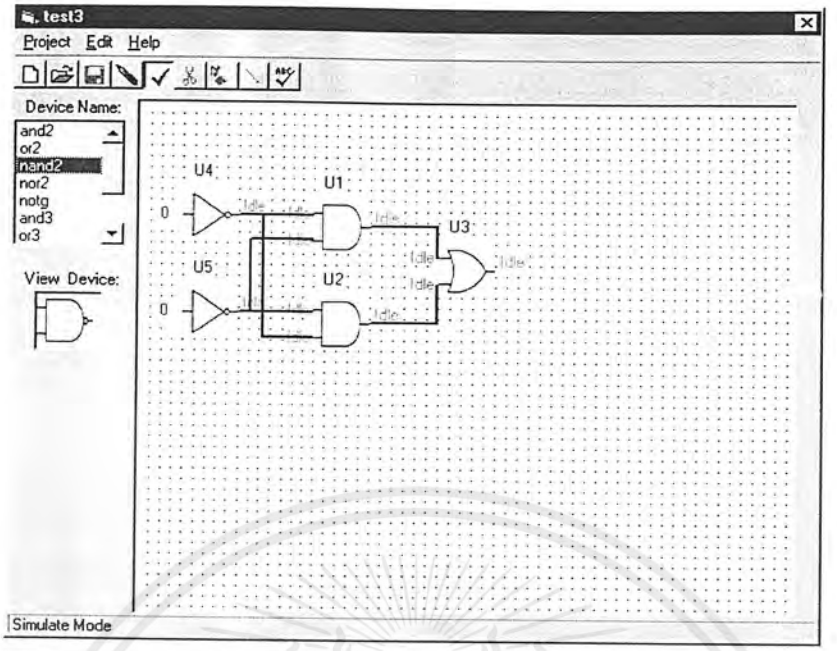
  ระหว่างการลากสายเราสามารถยกเลิกการลากสาได้โดยคลิ้กขวาที่ปุ่มเมาส์

10.2.2 การจำลองการทำงานของวงจร (Simulate Project)

การจำลองการทำงานวงจรที่ต้องการนั้น จะสามารถทำได้ก็ต่อเมื่อวงจรนั้นถูกบันทึกลงในฐานข้อมูลก่อน หรือเมื่อเปิดวงจรขึ้นมาแล้วยังไม่มีการแก้ไข เมื่อวงจรอยู่ในสถานะดังกล่าวโปรแกรมจะยอมให้มีการเปลี่ยนการทำงานไปสู่ โหมดการจำลองการทำงานวงจร ซึ่งเราสามารถเปลี่ยนโหมดได้ โดยคลิ้กที่ทูลบาร์ หรือไปที่ Menu -> Edit -> Simulate Mode หรือกดปุ่มคอนโทรลพร้อมกับปุ่มอักษรแซด (Ctrl + Z) เมื่อเปลี่ยนการทำงานสู่โหมดจำลองการทำงานแล้ว ที่ขาของแต่ละเกตจะมีสัญญานสถานะของลอจิกอยู่ที่แต่ละขา ดังตัวอย่างรูปที่ 10.11

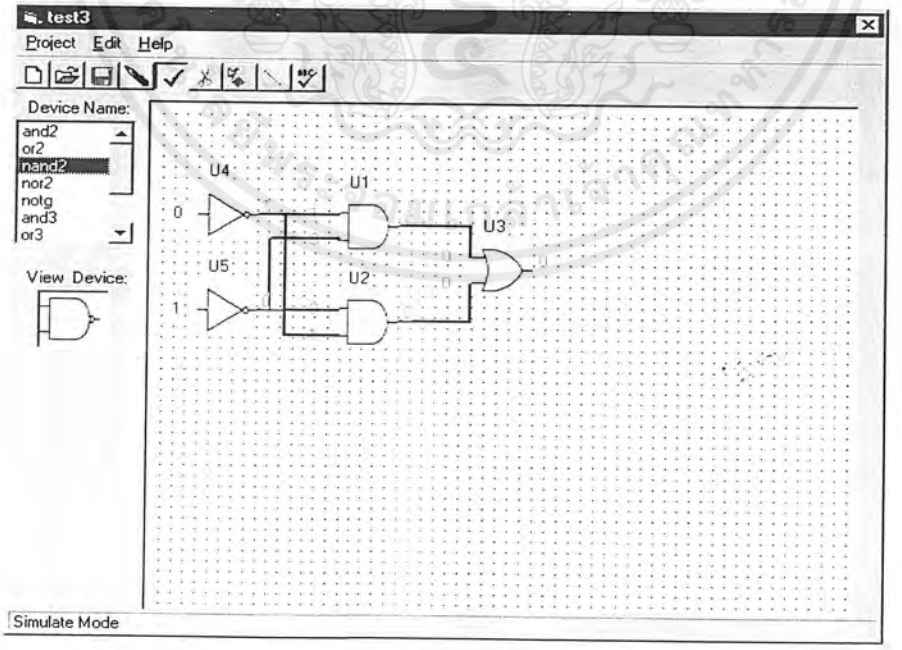


รูปที่ 10.11 การทำงานก่อนเข้าสู่โหมดจำลองการทำงาน



รูปที่ 10.12 การทำงานหลังจากเข้าสู่โหมดจำลองการทำงาน

ตอนนี้เราสามารถเปลี่ยนสถานะของลอจิกได้โดยคลิ้กที่สัญญาณที่ต้องการเปลี่ยนมันจะสลับสถานะลอจิก 0 กับ 1 ผู้ใช้สามารถสลับสัญญาณลอจิกที่แสดงด้วยสีแดง เมื่อเราตั้งค่าสถานะของลอจิกได้ตามต้องการแล้วให้คลิ้กที่ เพื่อให้โปรแกรมทำการจำลองการทำงาน เมื่อจำลองการทำงานเสร็จแล้วสัญญาณทางด้านเอาต์พุตจะเปลี่ยนสถานะลอจิกตามการทำงานของวงจรดังตัวอย่างรูปที่ 10.13



รูปที่ 10.13 การทำงานหลังจากจำลองการทำงานเสร็จแล้ว

10.2.3 การสร้างวงจรใหม่ (New Project)

ทำได้โดยไปที่ Menu -> File -> New Project หรือคลิ้กที่ทูลบาร์ โปรแกรมจะการติดตั้งค่าเริ่มต้นใหม่ทั้งหมด ถ้าวงจรที่เปิดอยู่ถูกแก้ไขแต่ยังไม่ได้มีการบันทึกโปรแกรมจถถามก่อนว่าต้องการบันทึกก่อนหรือไม่

10.2.4 การบันทึกวงจรลงฐานข้อมูล (Save Project)

เมื่อวงจรถูกสร้าง หรือมีการแก้ไขจึงจะสามารถบันทึกได้ เราสามารถบันทึกได้โดยไปที่ Menu-> File -> Save Project หรือคลิ้กที่ทูลบาร์

ถ้าวงจรที่เปิดอยู่นั่นเป็นวงจรที่มีอยู่ในฐานข้อมูลแล้ว โปรแกรมจะทำการบันทึกลงชื่อเดิม แต่ถ้าเป็นวงจรที่สร้างขึ้นใหม่และยังไม่เคยบันทึกลงฐานข้อมูลเลย โปรแกรมจะแสดงหน้าต่างดังรูปที่ 10.14 เพื่อให้ใส่ชื่อของวงจร โดยชื่อนั้นต้องไม่ซ้ำกับชื่อที่มีอยู่ในฐานข้อมูลถ้าผู้ใช้ใส่ชื่อซ้ำโปรแกรมจะทำการฟ้องและให้ใส่ชื่อวงจรใหม่

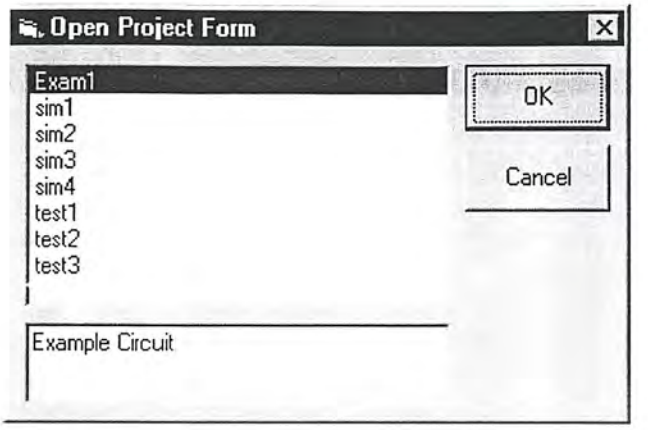
รูปที่ 10.14 ไดอะล็อกใส่ชื่อวงจรและ Description

10.2.5 การบันทึกวงจรลงฐานข้อมูลโดยเปลี่ยนชื่อวงจรใหม่ (Save as Project)

ทำได้โดย Menu -> File -> Save as Project จะปรากฏหน้าต่างลักษณะเดียวกันกับรูปที่ 10.14 เพื่อให้ใส่ชื่อของวงจรใหม่โดยชื่อที่ใส่ต้องไม่ซ้ำกับชื่อที่มีอยู่ในฐานข้อมูลถ้าผู้ใช้ใส่ชื่อซ้ำโปรแกรมจะทำการฟ้องและให้ใส่ชื่อวงจรใหม่

10.2.6 การเปิดวงจรจากฐานข้อมูล (Open Project)

ทำได้โดยไปที่ Menu -> File -> Open Project หรือคลิ้กที่ทูลบาร์ จะปรากฏหน้าต่างแสดงรายชื่อวงจรให้เลือกวงจรที่ต้องการเปิดขึ้นมาจากรายชื่อนั้น ดังรูปที่ 10.15 ซึ่งถ้ามีวงจรเปิดอยู่ก่อนหน้านี้โปรแกรมจะถามก่อนว่าต้องการบันทึกหรือไม่



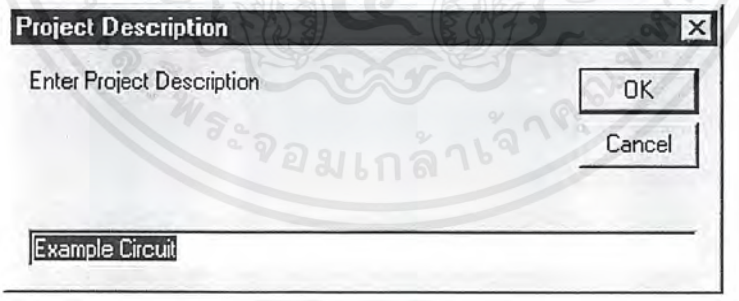
รูปที่ 10.15 ไดอะล็อกรายชื่อของวงจร

10.2.7 การลบวงจรจากฐานข้อมูล (Remove Project)

ทำได้โดยไปที่ Menu -> File -> Remove Project จะปรากฏหน้าต่างแสดงรายชื่อวงจรให้เลือกวงจรที่ต้องการลบจากรายชื่อนั้น ลักษณะเดียวกับรูปที่ 10.15

10.2.8 การแก้ไขส่วนรายละเอียดของวงจร (Project Description)

สามารถแก้ไขได้โดยไปที่ Menu -> File -> Project Description หลังจากนั้นจะปรากฏหน้าต่างแสดงให้ใส่ส่วนรายละเอียดวงจรใหม่ ดังรูปที่ 10.16 เมื่อแก้ไขแล้วจะยังไม่ถูกแก้ไขที่ฐานข้อมูลจนกว่าจะบันทึกวงจรลงฐานข้อมูล



รูปที่ 10.16 ไดอะล็อก Project Descriptiom

บทที่ 11
สรุปผลและวิจารณ์

11.1 สรุปผลโครงงาน

จากการที่ได้ศึกษาแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์ สามารถจัดทำระบบฐานข้อมูลที่มีคุณสมบัติทางทฤษฎีบางคุณสมบัติเท่านั้น เนื่องจากข้อจำกัดเรื่องเทคโนโลยีของระบบจัดการฐานข้อมูล ข้อจำกัดเรื่องเวลา และข้อจำกัดเรื่องความรู้ความเข้าใจและทักษะในการใช้งานระบบจัดการฐานข้อมูล ซึ่งก็คืออินฟอร์มิกส์ยูนิเวอร์แซลเซิร์ฟเวอร์

การศึกษาแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์ จะศึกษาจากวิทยานิพนธ์และบทความต่าง ๆ จากต่างประเทศ ควบคู่ไปกับการศึกษาความสามารถของระบบจัดการฐานข้อมูลอินฟอร์มิกส์ยูนิเวอร์แซลเซิร์ฟเวอร์ว่ามีความสามารถรองรับแนวความคิดที่ได้ศึกษาหรือไม่ และเริ่มต้นออกแบบฐานข้อมูลวงจรดิจิตอล ซึ่งในขั้นตอนการออกแบบ เกิดปัญหาคือ ไม่มีเครื่องมือหรือขั้นตอนที่เป็นมาตรฐานสำหรับช่วยในการออกแบบฐานข้อมูลเชิงวัตถุสัมพันธ์ จำเป็นต้องทำการออกแบบโดยใช้ อีอาร์ไดอะแกรมและทำการจัดสร้างออบเจ็กต์ต่าง ๆ เป็นข้อมูลชนิดโอเปคอีกครั้ง

ในการจัดสร้างระบบจัดการฐานข้อมูล จะทำการสร้างตารางที่ประกอบด้วยแอตทริบิวต์ที่เป็นทั้งข้อมูลบิวต์-อิน ข้อมูลชนิดรูปภาพ และข้อมูลชนิดโอเปค ซึ่งข้อมูลชนิดรูปภาพจะถูกสร้างด้วยการใช้ชนิดข้อมูลแบบ BLOB ส่วนชนิดข้อมูลแบบโอเปคจะใช้สร้างออบเจ็กต์ ซึ่งจะถูกสร้างขึ้นในดาต้าเบลดโมดูลที่เพิ่มเข้าไปในฐานข้อมูลก่อนที่จะถูกนำไปใช้ ซึ่งดาต้าเบลดโมดูลสามารถใช้งานได้ง่ายและมีประสิทธิภาพ สามารถเพิ่ม ลบ และอัพเกรดได้เป็นหลาย ๆ เวอร์ชัน แต่ในการสร้างออบเจ็กต์ด้วยการใช้ดาต้าเบลดโมดูลจะมีข้อจำกัดคือ เราไม่สามารถสร้างออบเจ็กต์ที่คุณสมบัติที่ครบถ้วนได้ ดาต้าเบลดโมดูลสามารถสร้างออบเจ็กต์ที่คุณสมบัติของการปกป้องข้อมูลในระดับไพรเวทซึ่งเป็นคุณสมบัติที่สำคัญที่ออบเจ็กต์ต้องมี การสื่อสารกับออบเจ็กต์จะผ่านฟังก์ชันสนับสนุนแต่ฟังก์ชันสนับสนุนไม่ได้ถูกฝังไว้ในออบเจ็กต์ แต่เป็นเพียงรูทีนหนึ่งในดาต้าเบสเซิร์ฟเวอร์เท่านั้น ซึ่งไม่ใช่คุณสมบัติของออบเจ็กต์ แต่ก็ทดแทนด้วยการมีคุณสมบัติของรูทีนโอเวอร์โหลดดิ้งซึ่งสามารถทำได้อย่างสมบูรณ์กลับรูทีนที่สร้างขึ้น คุณสมบัติที่สำคัญอีกอย่างหนึ่งที่ดาต้าเบลดไม่สามารถทำได้คือการถ่ายทอดคุณสมบัติของออบเจ็กต์ไทพ์ ซึ่งอินฟอร์มิกส์อนุญาติให้ทำการถ่ายทอดคุณสมบัติของข้อมูลชนิดแถวและข้อมูลชนิดเทเบิลเท่านั้น

บนฐานข้อมูลประกอบด้วยตารางต่าง ๆ รูทีน และทริกเกอร์ที่คอยควบคุมความถูกต้องของฐานข้อมูล ซึ่งรูทีนที่ใช้งานทั่วไปบนฐานข้อมูลจะถูกเขียนขึ้นด้วยภาษา SPL ซึ่งเป็นชุดของคำสั่ง SQL ที่ทำงานอย่างเป็นลำดับ มีการตัดสินใจ และลูปการทำงาน ได้มีการพยายามสร้างรูทีนในการจำลองการทำงานด้วยภาษา SPL ฝังไว้บนดาต้าเบสเซิร์ฟเวอร์ แต่เกิดปัญหาในเรื่องของการจัดหน่วยความจำบนดาต้าเบสเซิร์ฟเวอร์ เนื่องจากการจำลองการทำงานมีการใช้เคอร์เซอร์ซึ่งมีการจองหน่วยความจำเป็นจำนวนมากและมีข้อจำกัดเรื่องเวลา ดังนั้นจึงได้มีการเขียนโปรแกรมส่วนจำลองการทำงานที่ฝั่งไคลเอนต์ด้วย

ภาษาวิชวลเบสิกโดยใช้ดาต้าไดเรกเตอร์ช่วยในการเขียนโปรแกรมติดต่อกับฐานข้อมูล ซึ่งมีความคุ้ยเคยเข้าใจง่าย และทำได้สะดวกกว่า

โดยรวมแล้วโครงงานนี้ยังขาดคุณสมบัติของฐานข้อมูลเชิงวัตถุสัมพันธ์บางประการและยังไม่ใช่ระบบฐานข้อมูลที่สมบูรณ์ เนื่องด้วยข้อจำกัดของระบบจัดการฐานข้อมูล ซอฟต์แวร์ที่ใช้ในการจัดสร้างเครื่องมือที่ใช้ในการออกแบบ และตัวอย่างระบบที่จัดหาได้ยาก

11.2 แนวทางในการพัฒนาและแนวทางในการประยุกต์ใช้งาน

สำหรับแนวทางที่สามารถทำการพัฒนาต่อคือการพยามยามสร้างออบเจ็กต์ที่มีคุณสมบัติของฐานข้อมูลเชิงวัตถุสัมพันธ์อย่างครบถ้วย นั่นคือสามารถรวมทั้งข้อมูลและพฤติกรรมหรือรูทีนต่าง ๆ เข้าไว้ในออบเจ็กต์ มีการป้องกันข้อมูลในระดับต่าง ๆ มีการถ่ายทอดคุณสมบัติ และสามารถทำโอเวอร์โหลดดิ้งของรูทีนได้ ซึ่งอาจจะด้วยการเขียนไลบรารีขึ้นใช้เอง หรือการจัดหาระบบจัดการฐานข้อมูลที่สนับสนุนการสร้างข้อมูลชนิดออบเจ็กต์ที่มากขึ้น

ในส่วนของฐานข้อมูลวงจรดิจิตอลสามารถที่พัฒนาต่อในส่วนของการสร้างชนิดอุปกรณ์ใหม่จากชนิดอุปกรณ์ที่มีอยู่แล้ว คุณสมบัติของการนำมาใช้ใหม่ การพัฒนาคุณสมบัติการหน่วงเวลาของอุปกรณ์ในวงจรเพื่อให้ใกล้เคียงกับความเป็นจริงมากยิ่งขึ้น นอกจากนั้นยังพัฒนาในส่วนของแอปพลิเคชันโปรแกรมที่อนุญาติให้ผู้ใช้สร้างอุปกรณ์ชนิดใหม่และเพิ่มเข้าไปในฐานข้อมูล

นอกจากนั้นยังสามารถที่ศึกษาและพัฒนาเรื่องคุณสมบัติของระบบจัดการฐานข้อมูลแบบวัตถุสัมพันธ์เพื่อจัดการกับออบเจ็กต์ที่สร้างขึ้นเอง เช่น ความปลอดภัย การกู้ข้อมูลคืน การทำทรานแซคชัน การปรับปรุงประสิทธิภาพ เป็นต้น

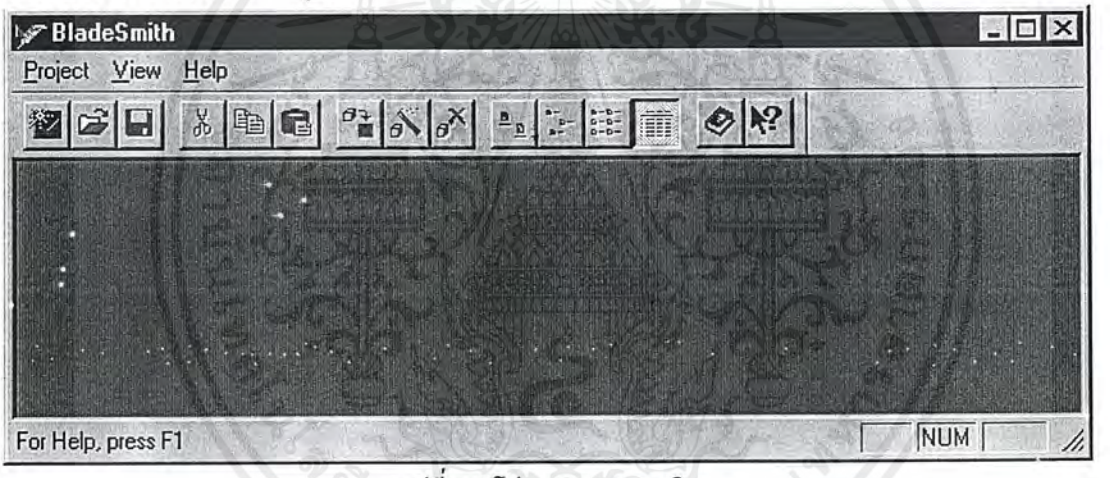
ภาคผนวก

ภาคผนวก ก.
ขั้นตอนการสร้างและใช้งานดาต้าเบลดโมดูล

ก-1 การสร้างดาต้าเบลดโมดูลด้วยโปรแกรมเบลดสมิธ

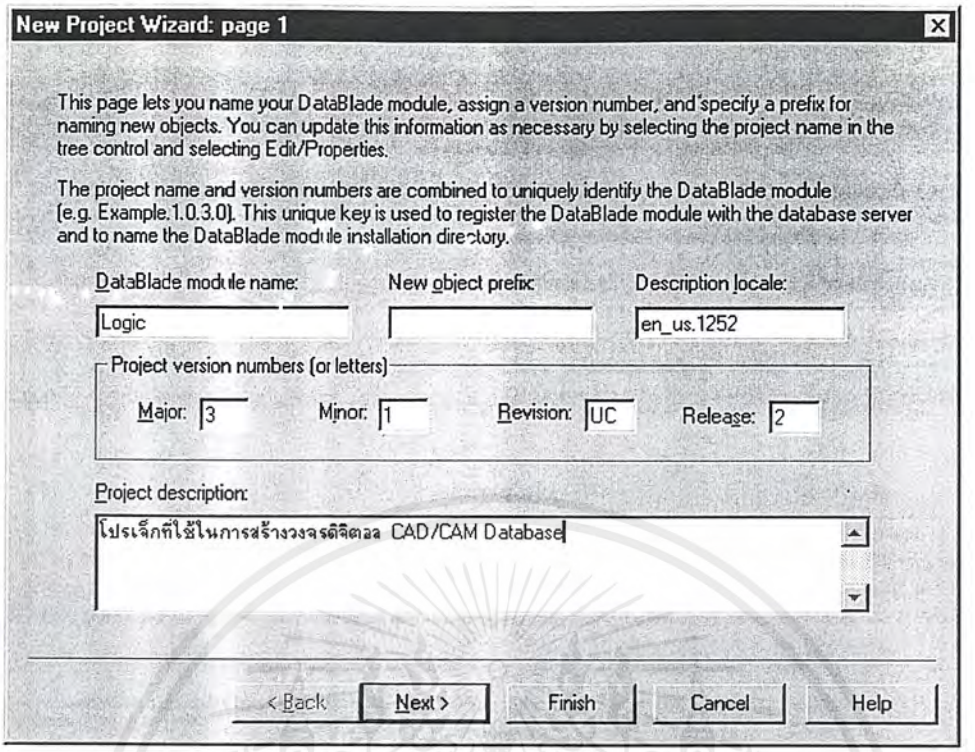
การจัดสร้างดาต้าเบลดโมดูลมีวัตถุประสงค์ที่สำคัญคือ การเพิ่มองค์ประกอบใหม่ ๆ เข้าไปในดาต้าเบสเซิร์ฟเวอร์ซึ่งประกอบเข้าด้วยกันเป็นชุดและใส่เข้าไปครั้งเดียว ทำให้สะดวกในการเพิ่มหรือลบองค์ประกอบต่าง ๆ ซึ่งประกอบด้วยรูทีนและชนิดข้อมูลที่ผู้ใช้ต้องการสร้าง สำหรับขั้นตอนในการสร้างดาต้าเบลดโมดูลมีดังต่อไปนี้

1. เปิดโปรแกรมเบลดสมิธ ซึ่งเป็นโปรแกรมหนึ่งของชุดโปรแกรมดาต้าเบลดดิเวลลอปเปอร์คิท (DataBlade Module Developer Kid) ซึ่งผู้ใช้สามารถสร้างองค์ประกอบที่ต้องการเพิ่มเข้าไปในดาต้าเบสเซิร์ฟเวอร์ได้ ซึ่งมีหน้าจอดังรูปที่ ก-1



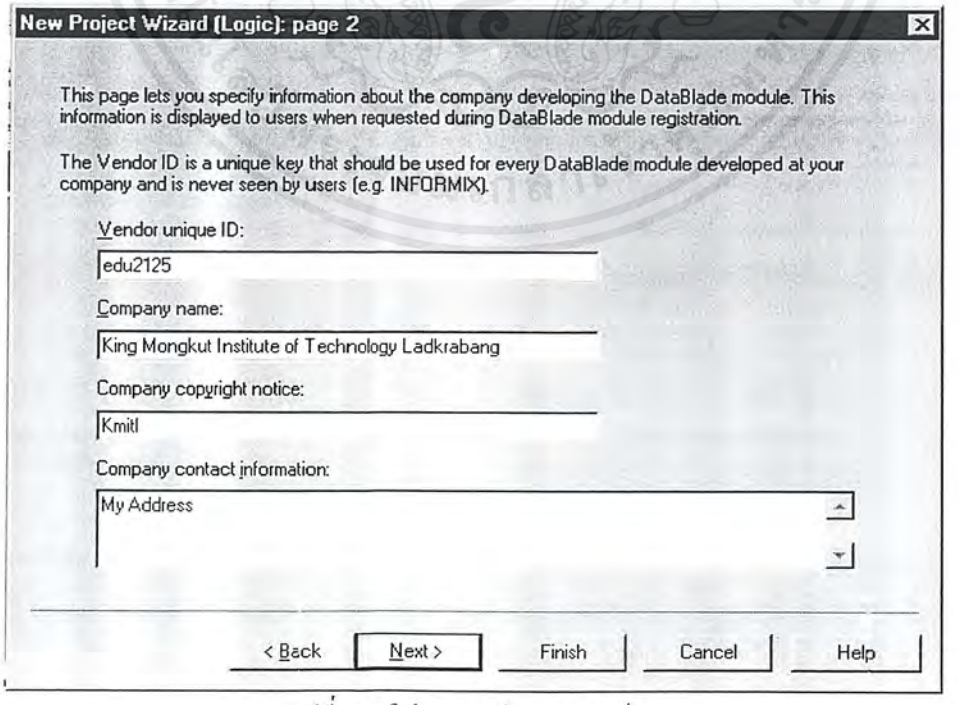
รูปที่ ก-1 โปรแกรมเบลดสมิธ

2. เลือกเมนู Project / New เพื่อสร้างโปรเจ็คใหม่ ซึ่งสิ่งที่สำคัญที่ควรสนใจคือ DataBlade Module Name คือชื่อของดาต้าเบลดโมดูล ซึ่งจะมีชื่อไม่ซ้ำกับดาต้าเบลดโมดูลที่มีอยู่แล้วในระบบฐานข้อมูล และเลขเวอร์ชันของดาต้าเบลดโมดูล ซึ่งประกอบขึ้นจาก Major , Minor , Revision และ Release ซึ่งตัวเลขที่ใส่ทั้งหมดจะถูกนำมาต่อกัน เช่น 3.20.UC1 เป็นต้น ซึ่งส่วนที่สำคัญที่สุดคือ Major ซึ่งเสมือนเป็นเวอร์ชันที่แท้จริงของดาต้าเบลดโมดูล การเพิ่มดาต้าเบลดโมดูลชื่อเดียวกันเข้าไปในฐานข้อมูล จะมีการตรวจสอบเวอร์ชัน ถ้าเวอร์ชันใหม่กว่า (ตัวเลขมากกว่า) เวอร์ชันใหม่จะถูกเพิ่มแทนที่เวอร์ชันเก่า ถ้าเวอร์ชันเก่ากว่าจะไม่สามารถเพิ่มเข้าได้ รายละเอียดแสดงได้ดังรูปที่ ก-2 หลังจากนั้นทำการกดปุ่ม Next



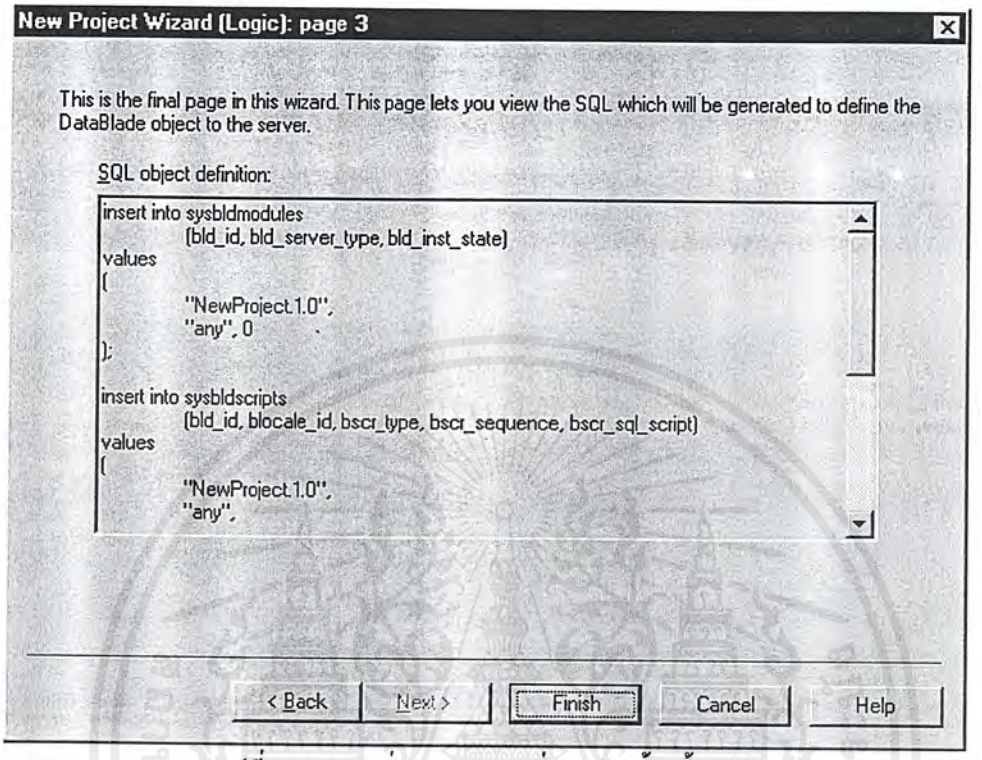
รูป ก-2 การใส่รายละเอียดของดาต้าเบลดโมดูล

2. ใส่ข้อมูลเกี่ยวกับบริษัทหรือองค์กร ได้แก่ ชุดของตัวอักษรหรือตัวเลขที่แสดงความเป็นเจ้าของ ดาต้าเบลดโมดูลนั้น (Vendor Unique ID) , ชื่อของบริษัท (Company Name) , หมายเลขมาตรฐานแสดง Copyright (Company copyright notice) และสถานที่ติดต่อของบริษัท (Company contact information) หน้าต่างนี้แสดงได้ดังรูปที่ ก-3 หลังจากนั้นทำการกดปุ่ม Next



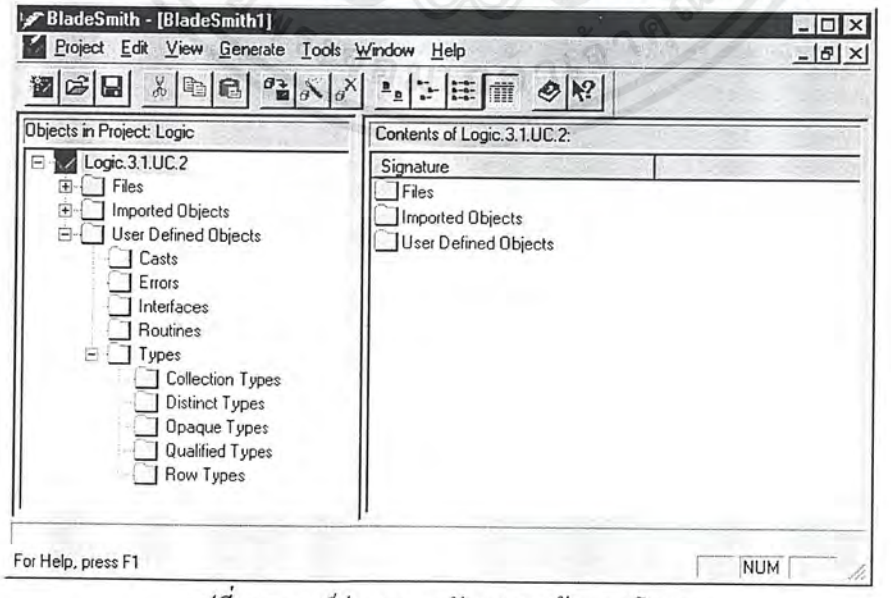
รูปที่ ก-3 ใส่รายละเอียดขององค์กร

4. เบลดสมิธจะทำการสร้างชุดคำสั่ง SQL ที่จะทำการเอกซีคิวต์ขณะที่ทำการเพิ่มดาต้าเบลดโมดูลเข้าไปในดาต้าเบสเซิร์ฟเวอร์ แสดงได้ดังรูปที่ ก-4 หลังจากนั้นทำการกดปุ่ม Finish เพื่อเริ่มทำการสร้างองค์ประกอบต่าง ๆ ของดาต้าเบลดโมดูล



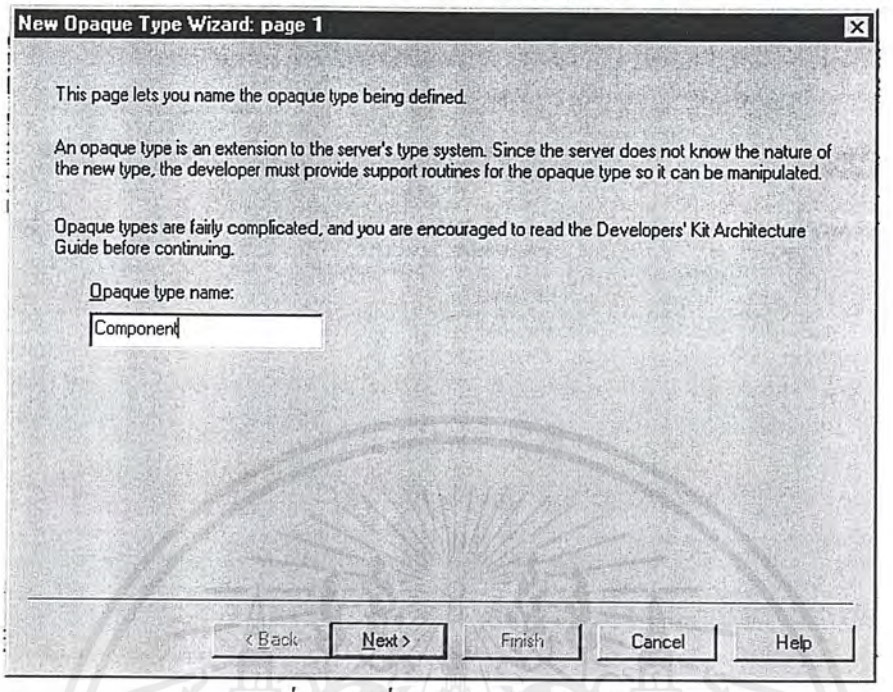
รูปที่ ก-4 ชุดคำสั่งภาษา SQL ที่ถูกสร้างขึ้นเบื้องต้น

5. เริ่มทำการสร้างองค์ประกอบต่าง ๆ โดยในหน้าต่างหลักนี้แสดงได้ดังรูปที่ ก-5 ซึ่งประกอบด้วยองค์ประกอบหลัก 3 ส่วนด้วยกันคือ Files, Imported Objects และ User Defined Objects ซึ่งส่วนที่จำเป็นต้องใช้ในโครงงานนี้คือ User Defined Objects ซึ่งมีหน้าที่สร้างรูทีนและไทพ์ขึ้นมาใหม่



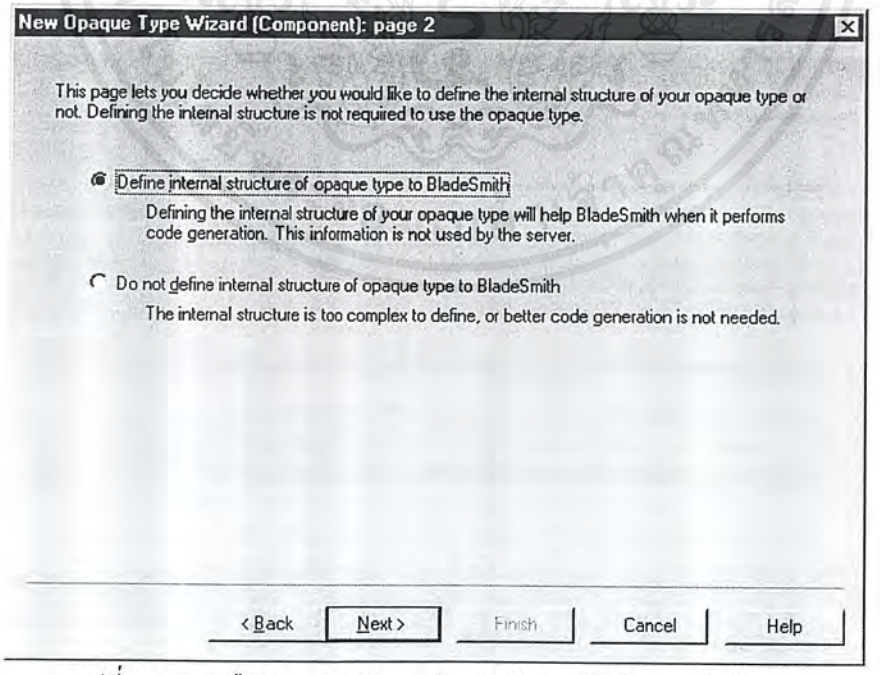
รูปที่ ก-5 องค์ประกอบหลักของดาต้าเบลดโมดูล

6. ทำการสร้างชนิดข้อมูลโอเปคโดยเลือกเมนู Edit / Insert / Opaque Type จะมีหน้าต่างขึ้นมาให้ใส่ชื่อของโอเปคซึ่งเป็นชื่อที่เป็นไปตามกฎการตั้งชื่อทั่วไปแสดงได้ดังรูปที่ ก-5 หลังจากนั้นทำการกด Next



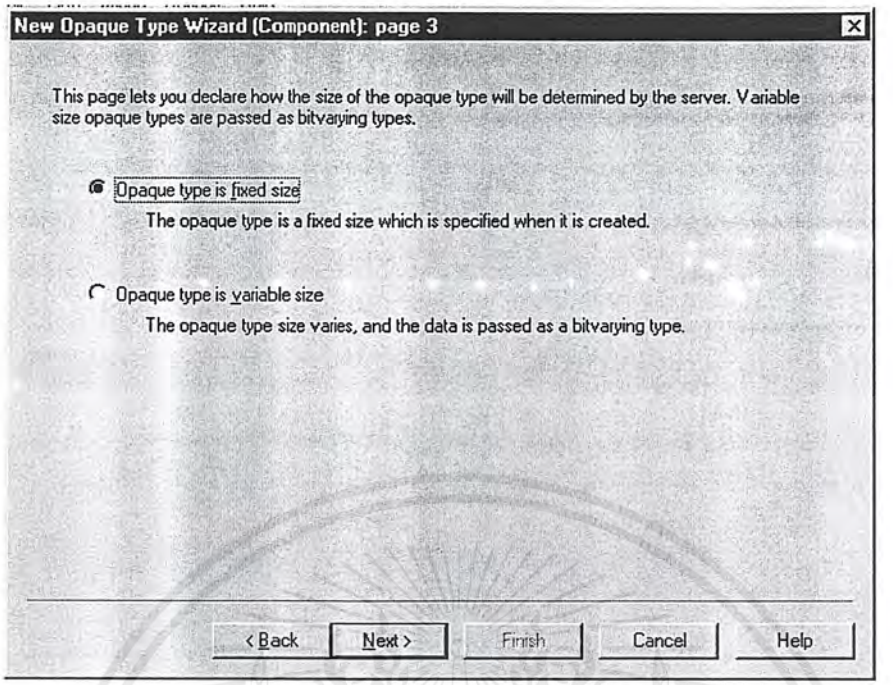
รูปที่ ก-6 ใส่ชื่อของชนิดข้อมูลโอเปค

7. เมื่อกด Next จะมีหน้าต่างดังรูป โปรแกรมจะให้ทำการเลือกว่าจะกำหนดโครงสร้างภายในโอเปคหรือไม่ ถ้าต้องการกำหนดโครงสร้างภายในในกรณีที่มีโครงสร้างไม่ซับซ้อนมากนัก เลือกทางเลือกแรก แต่ถ้าโครงสร้างภายในมีความซับซ้อนเกินไปก็ให้เลือกอีกทางเลือกหนึ่ง



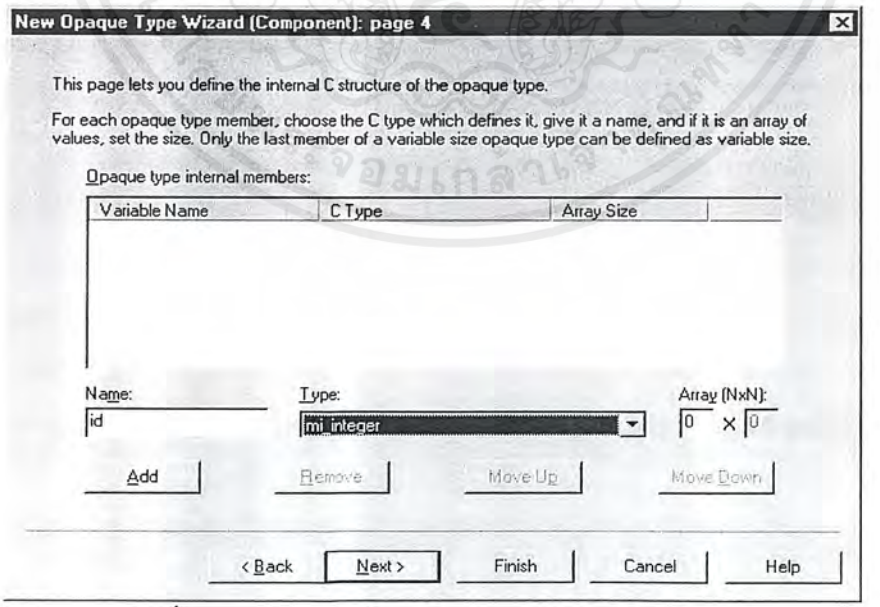
รูปที่ ก-7 ทางเลือกของการกำหนดโครงสร้างภายในข้อมูลชนิดโอเปค

8. เมื่อทำการกด Next โปรแกรมจะให้เลือกชนิดของโอเปคว่ามีขนาดคงที่หรืออาจจะมีขนาดที่เปลี่ยนแปลงได้ แสดงได้ดังรูป หลังจากนั้น ทำการกด Next



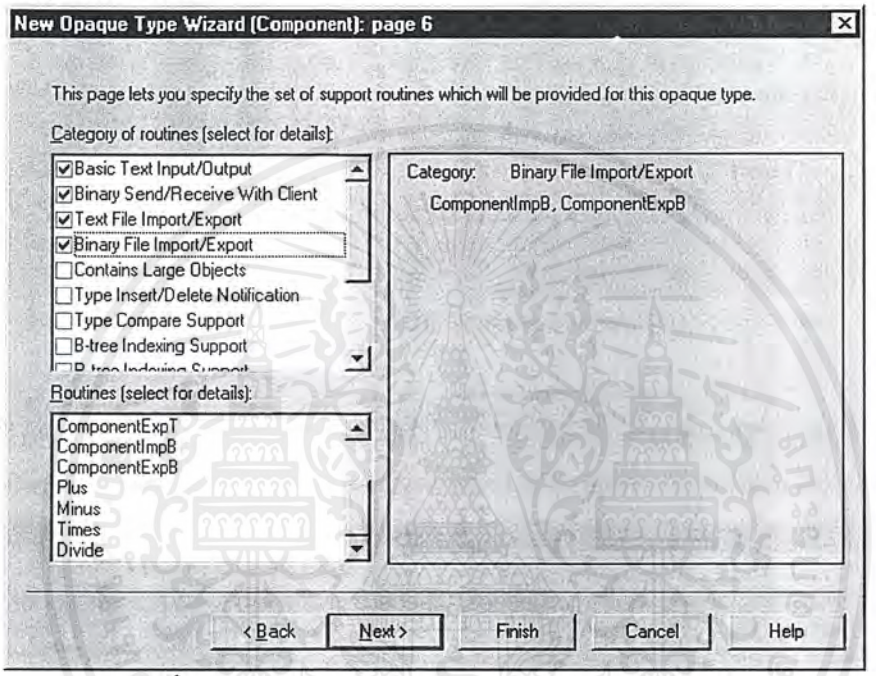
รูป ก-8 การกำหนดขนาดของข้อมูลชนิดโอเปค

9. ที่หน้านี้จะเป็นหน้าที่จะใช้ในการเพิ่มและลบ โครงสร้างภายในของโอเปค แสดงได้ดังรูปที่ ก-9 ซึ่งโครงสร้างภายในแต่ละตัวจะประกอบด้วยชื่อ ชนิดข้อมูลในภาษาซี และขนาดของอาร์เรย์ เมื่อทำการกำหนดแล้วก็ทำการกดปุ่ม Add เมื่อต้องการลบออกก็กดปุ่ม Remove เมื่อจัดโครงสร้างภายในเสร็จแล้วก็กดปุ่ม Next 2 ครั้ง ถ้าต้องการกำหนดรายละเอียดต่อไป ถ้าไม่ต้องการกำหนดรายละเอียดก็สามารถจบตรงนี้ได้เลย โดยการกดปุ่ม Finish



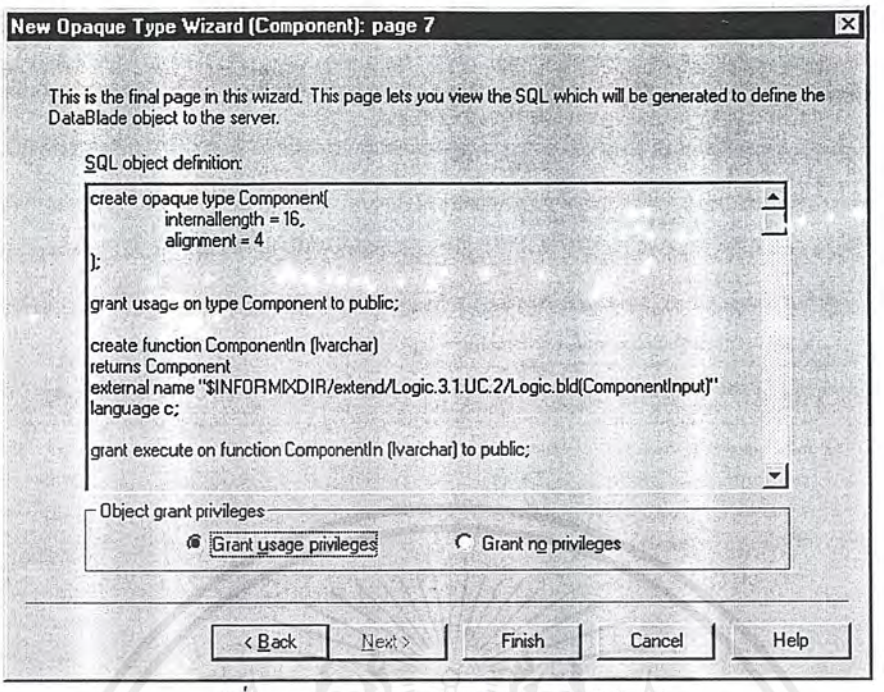
รูปที่ ก-9 การกำหนดข้อมูลต่าง ๆ ภายในชนิดข้อมูลโอเปค

10. กำหนดรูทีนสนับสนุนการทำงาน (Support Routine) พื้นฐานที่เบลดสมิธมีให้ ซึ่งจะแบ่งเป็นประเภทต่าง ๆ ซึ่งชื่อของฟังก์ชันสนับสนุนการทำงานจะขึ้นต้นด้วยชื่อของโอเปค เช่น ฟังก์ชันสนับสนุนการทำงานที่เกี่ยวกับฟังก์ชัน Input/Output ของโอเปคที่ชื่อ Component จะมี ComponentIn และ ComponentOut บางฟังก์ชันไม่จำเป็นต้องมีชื่อโอเปคนำหน้าเช่น ฟังก์ชันที่เกี่ยวกับการคำนวณทางคณิตศาสตร์ จะมี Plus,Minus,Times และ Devide ซึ่งมีการทำงานที่แตกต่างออกไปในรายละเอียด เมื่อกำหนดฟังก์ชันสนับสนุนการทำงานที่จำเป็นแล้วก็ทำการกด Next เพื่อกำหนดรายละเอียดอื่นหรือกด Finish เพื่อสิ้นสุดการกำหนด



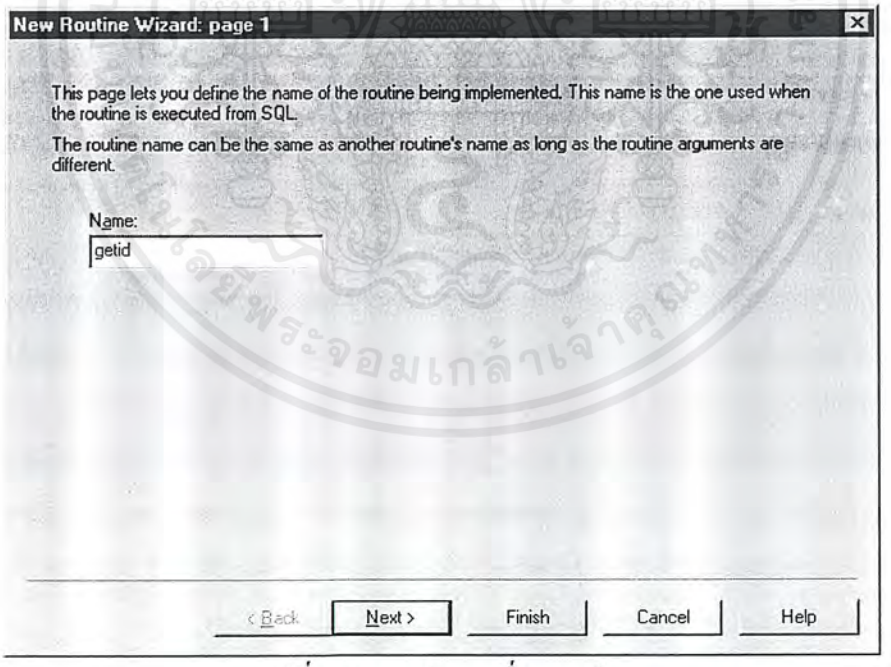
รูปที่ ก-10 การกำหนดรูทีนสนับสนุนการทำงานของโอเปค

11. เมื่อกด Next จะมีการแสดงชุดคำสั่ง SQL ที่จะทำการรันเมื่อทำการเพิ่มเข้าไปในเซิร์ฟเวอร์เพื่อสร้างโอเปคที่สร้างขึ้น สำหรับ Objects grant Privileges จะเป็นการกำหนดสิทธิในการเข้าถึงโอเปค ซึ่งถ้าเป็น Grant Usage Privileges จะกำหนดให้ผู้ใช้ทุกคนสามารถเข้าถึงโอเปคได้ แต่มีเฉพาะ Owner เท่านั้นที่สามารถลบได้ ซึ่ง Owner คือผู้ที่ทำการ Login เข้าใช้เวลานั้น สำหรับ Grant none Privileges จะอนุญาตให้เฉพาะ Owner เท่านั้นที่สามารถทำการเข้าถึงได้และลบมันได้ เมื่อเสร็จขั้นตอนนี้ทำการกดปุ่ม finish เป็นการสิ้นสุดการสร้าง Opaque Type แสดงได้ดังรูปที่ ก-11



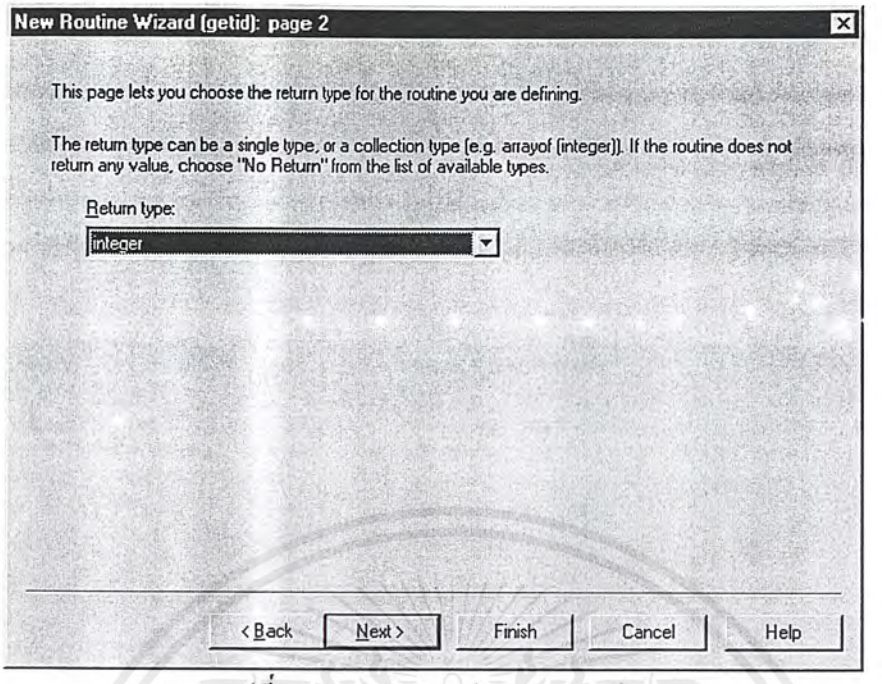
รูปที่ ก-11 การกำหนดสิทธิในการเข้าถึงโอเปค

12. ทำการสร้างรูทีนเพื่อทำการอ่านหรือเขียนข้อมูลลงในโอเปคที่สร้างขึ้น โดยเลือกเมนู Edit / Insert / Routine จะมีหน้าต่างที่จะให้ใส่ชื่อรูทีน แสดงได้ดังรูปที่ ก-12



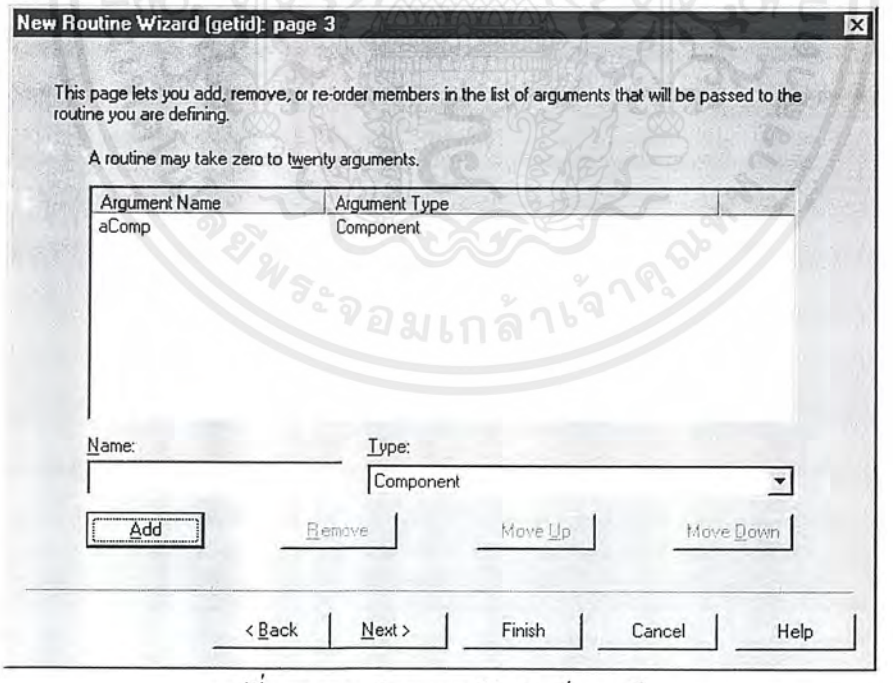
รูปที่ ก-12 การกำหนดชื่อของรูทีน

13. เมื่อกด Next โปรแกรมจะให้เลือกชนิดข้อมูล ที่ต้องการส่งกลับ ถ้าระบุชนิดข้อมูลที่จะส่งกลับ โปรแกรมจะถือว่ารูทีนนั้นเป็นฟังก์ชัน แต่ถ้ากำหนดให้เป็น No Return Type จะถือว่ารูทีนนั้นเป็นโปรซีเดอร์ ดังรูปที่ ก-13 เมื่อเลือกแล้วทำการกด Next



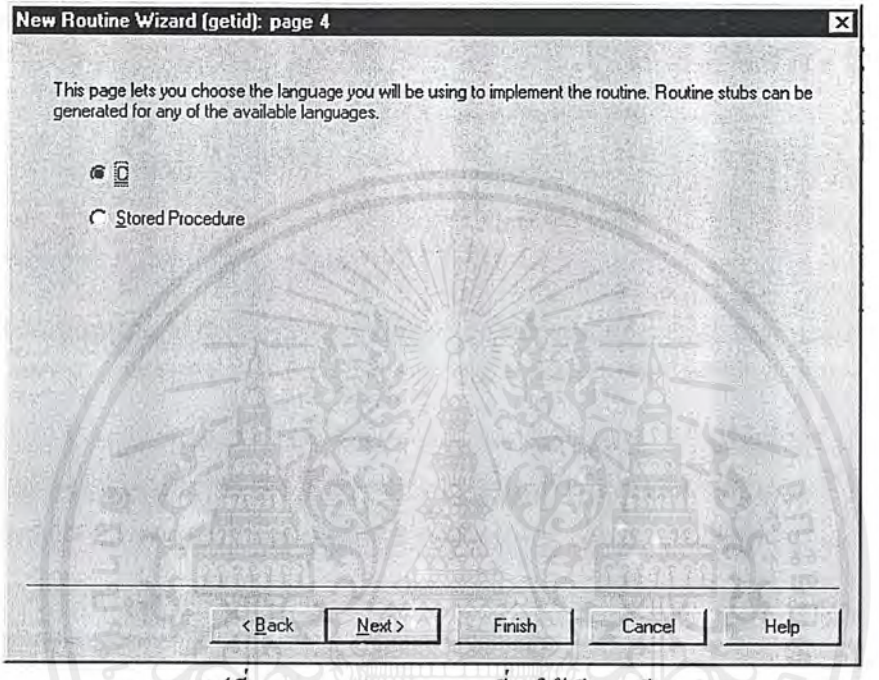
รูปที่ ก-13 การกำหนดประเภทของรูทีน

14. หลังจากกด Next จะให้กำหนดอากิวเมนท์ โดยจะต้องกำหนด ชื่ออากิวเมนท์ และ ชนิดของอากิวเมนท์สามารถเพิ่มและลบโดยการกดปุ่ม Add และ Remove ดังรูปที่ ก-14 เมื่อเสร็จแล้วกด Next เพื่อกำหนดรายละเอียดอื่น ๆ แต่ถ้าไม่ต้องการก็ทำการกด Finish



รูปที่ ก-14 การกำหนดอากิวเมนต์ของรูทีน

15. เมื่อทำการกดปุ่ม Next จะให้เลือกภาษาที่จะใช้เขียนรูทีน ซึ่งจะเป็นภาษาซีหรือภาษา SPL ซึ่งถ้าเป็นการเขียนรูทีนเพื่อเข้าถึงโครงสร้างภายในของโอเปคจะต้องเขียนด้วยภาษาซี แต่ถ้าเป็นรูทีนที่ใช้งานทั่วไปฝังไว้ในดาต้าเบสเซิร์ฟเวอร์ ก็สามารถเขียนด้วยภาษา SPL ซึ่งถ้าเขียนด้วยภาษาซี จะไม่มีการให้เขียนโค้ดลงไป จะต้องไปแก้ไขโค้ดที่สร้างออกมาในภายหลัง แต่ถ้าเป็นภาษา SPL เบลดสมิธจะมีอิดิเตอร์ (Editor) ให้เขียนลงไปได้ทันที ดังรูปที่ ก-15

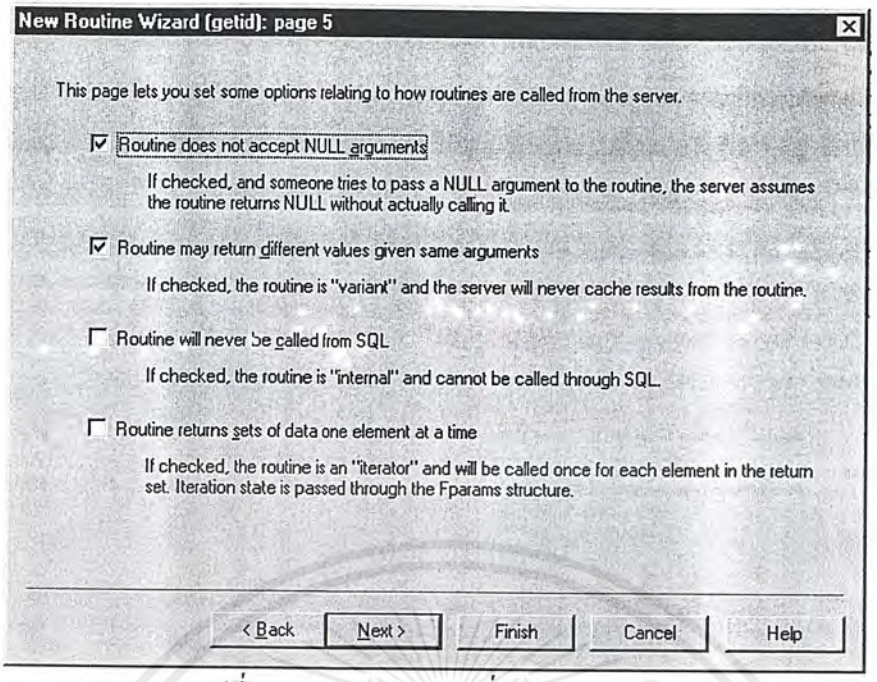


รูปที่ ก-15 การกำหนดภาษาที่จะใช้เขียนรูทีน

16. เมื่อทำการกด Next โปรแกรมจะให้เลือก Option ต่าง ๆ ได้แก่

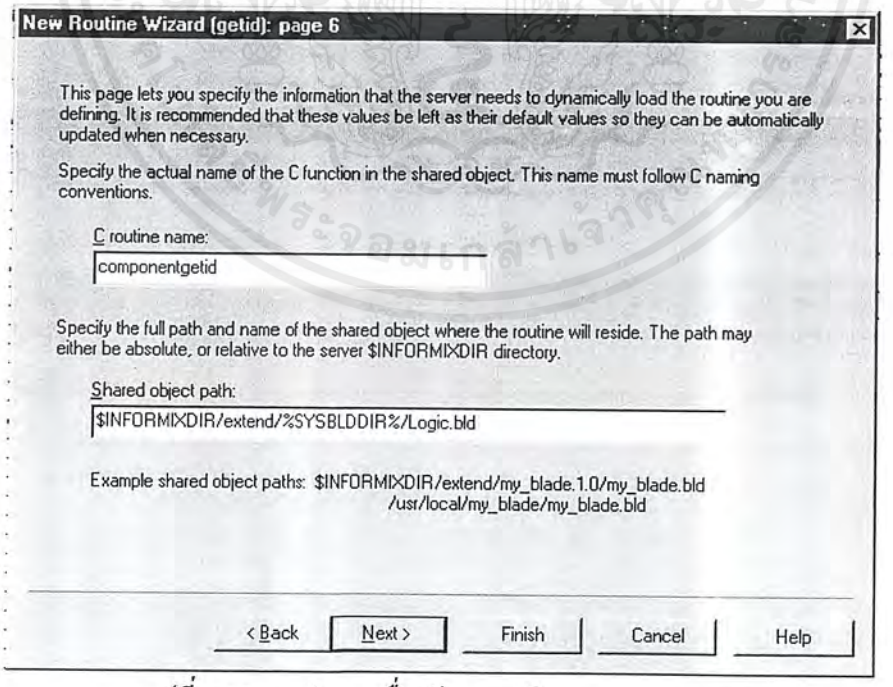
- รูทีนสามารถมีอาก้วเมนท์มีค่า NULL ได้หรือไม่
- รูทีนสามารถส่งค่ากลับมีค่าแตกต่างกันในขณะที่ อาก้วเมนท์เหมือนกันได้หรือไม่
- รูทีนเป็นรูทันภายใน (Internal routine) ซึ่งจะไม่สามารถถูกเรียกจากคำสั่ง SQL หรือไม่
- รูทีนสามารถส่งค่ากลับเป็นเซตได้หรือไม่

เมื่อกำหนดค่าแล้วกดปุ่ม Next เมื่อต้องการกำหนดรายละเอียดอื่น ๆ ของรูทีน หรือกดปุ่ม Finish เมื่อต้องการสิ้นสุดการกำหนด ดังรูปที่ ก-16



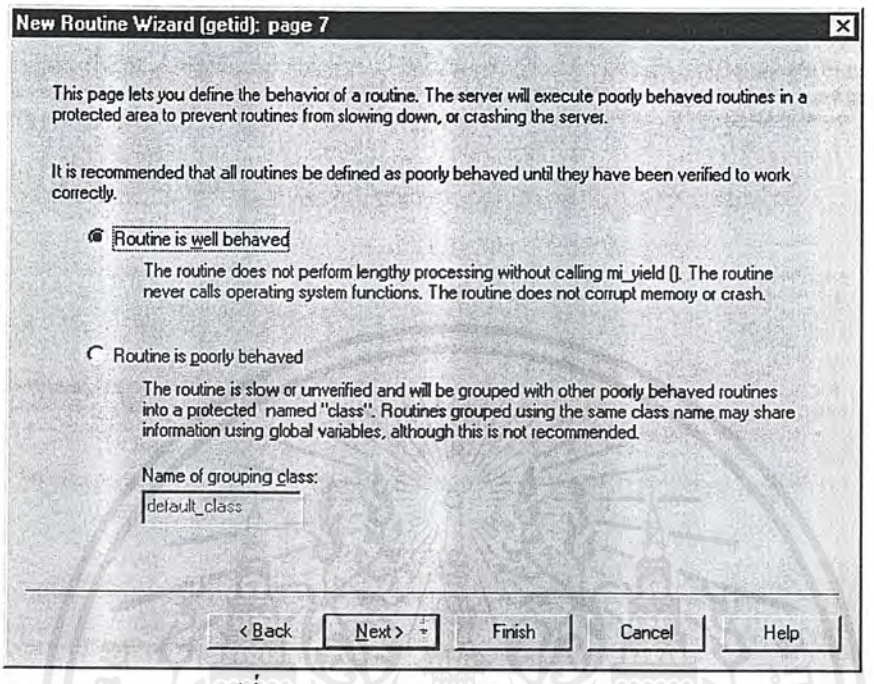
รูปที่ ก-16 การกำหนดออปชั่นต่าง ๆ ของรูทีน

17. เมื่อกดปุ่ม Next โปรแกรมให้ให้ใส่ชื่อรูทีนที่จะถูกสร้างเป็นภาษาซี ซึ่งมีประโยชน์สำหรับการทำโอเวอร์โหลดดิงรูทีน คือ รูทีนสามารถมีชื่อเดียวกันได้ แต่มีชื่อรูทีนภาษาซีที่แตกต่างกัน ซึ่งค่า Default จะมีค่าเป็นชื่อรูทีนที่ตั้งไว้ตอนแรก ผู้ใช้สามารถเปลี่ยนแปลงได้เมื่อต้องการให้ชื่อที่แตกต่างกัน รวมทั้งยังสามารถกำหนดไดเรกทอรีที่จะทำการเก็บ Shared Object ซึ่งก็คือไฟล์นามสกุล .BLD ที่จะใช้ในการเก็บรูทีนนี้ไว้ภายใน ดังรูปที่ ก-17



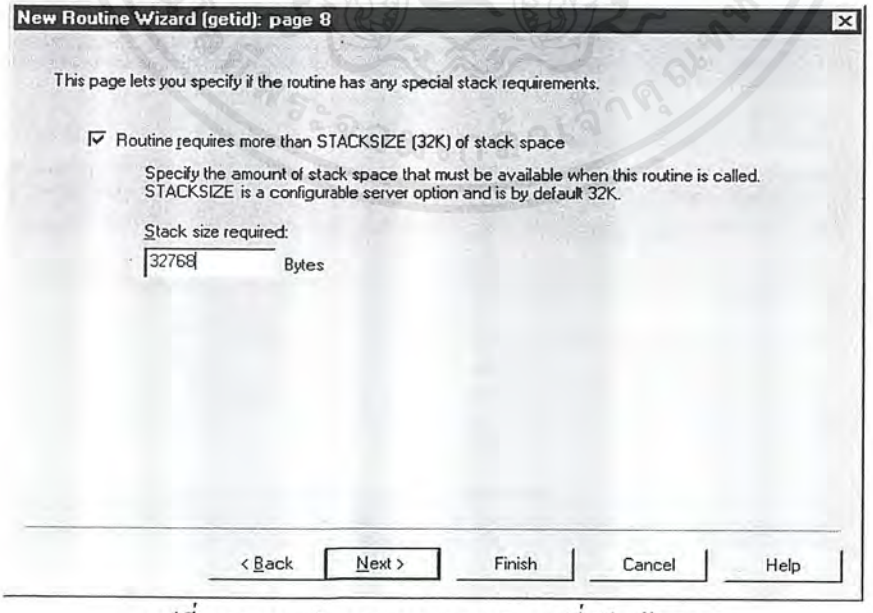
รูปที่ ก-17 การกำหนดชื่อรูทีนภาษาซีและ shared object

18. เมื่อกดปุ่ม Next โปรแกรมจะให้กำหนดพฤติกรรมของรูทีนในขณะที่รูทีนถูกรันว่าจะให้เป็น Well Behaved หรือ Poorly Behaved ถ้าเป็น Poorly Behaved ดาต้าเบสเซิร์ฟเวอร์จะรันรูทีนที่พื้นที่ที่ถูกป้องกันไว้เพื่อป้องกันระบบหรือหน่วยความจำ Crash ดังรูปที่ ก-18 หลังจากนั้นกด Next เพื่อกำหนดรายละเอียดเพิ่มเติม ถ้าไม่ต้องการกดปุ่ม Finish



รูปที่ ก-18 การกำหนดพฤติกรรมของรูทีน

19. เมื่อทำการกด Next จะเป็นการกำหนดว่า เมื่อมีการเรียกใช้รูทีนจะให้มีการสร้าง Stack ไว้เท่าใด ซึ่งค่า Default คือ 32 K แต่ถ้าต้องการมากกว่านี้ก็สามารถกำหนดได้ ดังรูปที่ ก-19 หลังจากนั้น กดปุ่ม Next ถ้าต้องการกำหนดรายละเอียดอื่น ๆ

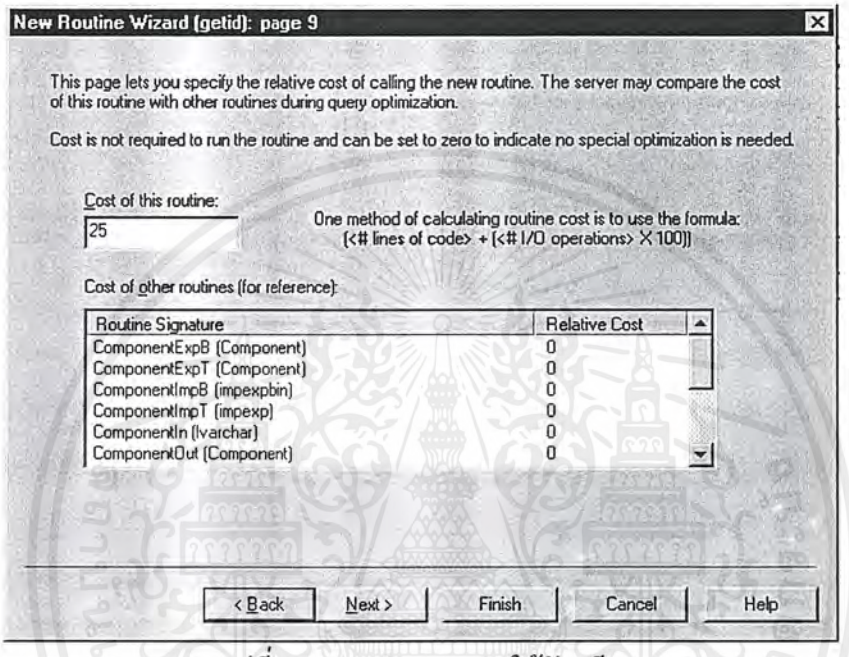


รูปที่ ก-19 การกำหนดขนาดของ Stack ที่รูทีนต้องการ

20. เมื่อกด Next จะเป็นการกำหนด Relative Cost ให้กับรูทีน ซึ่ง Cost นี้จะใช้เปรียบเทียบกับรูทีนอื่น ๆ ในขั้นตอน Query Optimization ซึ่งสูตรในการคำนวณ Cost อย่างง่าย ๆ คือ

Cost = (จำนวนบรรทัดของ Source Code) X ((จำนวนการทำคำสั่งเกี่ยวกับ I/O) X 100)

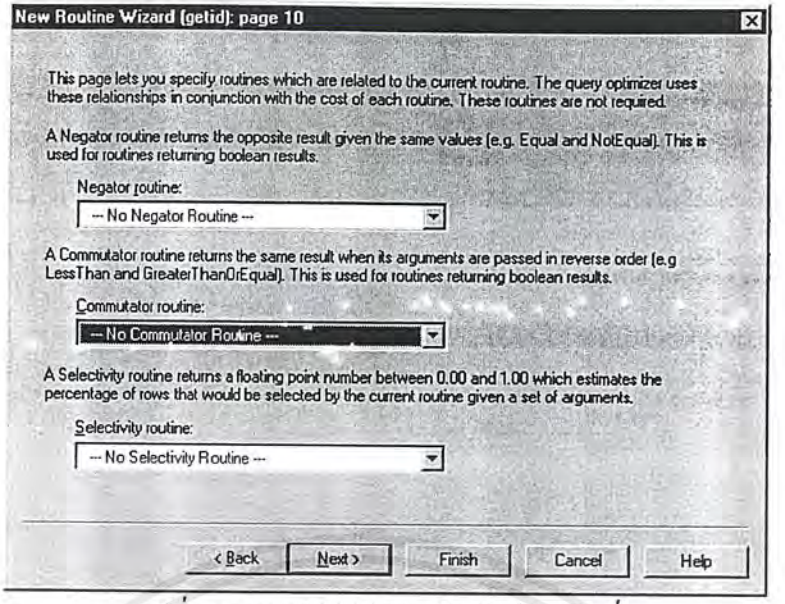
ซึ่งการใส่ Cost เท่ากับ 0 แสดงว่าเราไม่ต้องการนำเอารูทีนไปช่วยในขั้นตอนการ Optimization การกำหนดแสดงได้ดังรูปที่ ก-20



รูปที่ ก-20 การกำหนด cost ให้กับรูทีน

21. เมื่อกด Next จะเป็นการกำหนดความสัมพันธ์กับรูทีนอื่น ๆ ซึ่งความสัมพันธ์เป็นดังนี้
    - Negator Routine จะส่งกลับค่าตรงข้ามกันเมื่อให้ค่าที่เท่ากัน เช่น Equal หรือ NotEqual สำหรับรูทีนที่ส่งกลับเป็นค่า Boolean เท่านั้น
    - Commutator Routine จะส่งกลับค่าเดียวกันถ้าเรียงลำดับอากิวเมนท์ แบบ Reverse Order เช่น GreaterThan หรือ LessThan สำหรับรูทีนที่ส่งกลับค่าที่เป็นรูทีนเท่านั้น
    - Selectivity Routine จะส่งกลับค่า Floating – Point ระหว่าง 0.00 ถึง 1.00 เพื่อแสดงเปอร์เซนต์ของ Row ที่ถูกเลือก สำหรับรูทีนที่ส่งกลับค่าเป็นเซตของ Row

    การกำหนดแสดงได้ดังรูปที่ ก-21



รูปที่ ก-21 การกำหนดความสัมพันธ์กับรูทีนอื่น

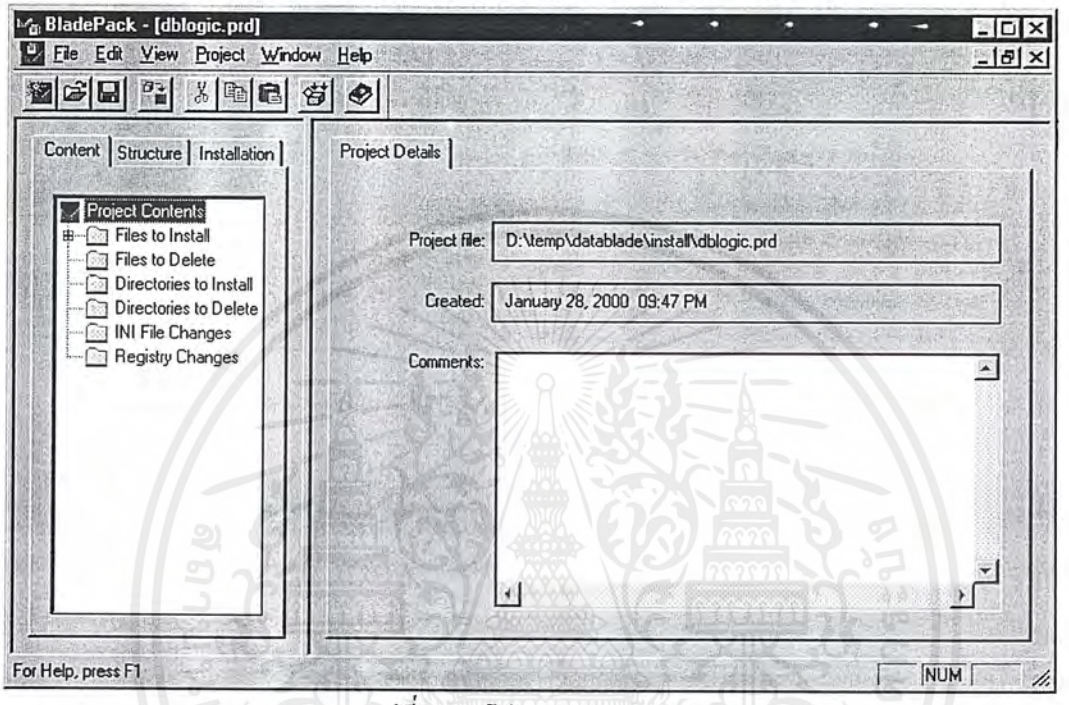
22. เมื่อกดปุ่ม Next จะเป็นการแสดง SQL Object Definition ซึ่งเป็นชุดคำสั่ง SQL ที่จะถูกรันตอนที่จะเพิ่มเข้าไปในดาต้าเบสเซิร์ฟเวอร์ รวมทั้งให้เลือกสิทธิการเข้าถึงรูทีน ซึ่งมี 2 อย่างคือ
    - Grant execute privileges คือ ผู้ใช้ทุกคนสามารถรันรูทีนนี้ได้ แต่มีเฉพาะ Owner ของรูทีนเท่านั้นที่จะสามารถลบรูทีนได้
    - Grant no privileges คือมีเฉพาะ Owner เท่านั้นที่จะสามารถรันและลบรูทีนนี้ได้

    เมื่อกำหนดเรียบร้อย ก็กดปุ่ม Finish เพื่อจบขั้นตอนการสร้างรูทีน
23. ทำการสร้าง SQL Script โดยใช้เมนู Generate / SQL Scripts... ซึ่งจะเป็นชุดของคำสั่ง SQL ที่จะต้องถูกรันขณะที่เพิ่มดาต้าเบลดโมดูลเข้าไปในฐานข้อมูล
24. ทำการสร้างโค้ดโดยใช้เมนู Generate / Source Code (needed) ... ซึ่งจะสร้างเป็นภาษาซี ในส่วนของการสร้างโอเปคและรูทีนที่สร้างขึ้น
25. ทำการสร้างแพ็กเกจในการติดตั้ง โดยใช้เมนู Generate / Installtaion Package (needed) ... ซึ่งจะทำการสร้างชุดที่ใช้ติดตั้งเข้าไปดาต้าเบสเซิร์ฟเวอร์ซึ่งจะถูกใช้โดยโปรแกรมเบลดแพ็กต่อไป

ก-2 การทำแพ็กเกจในการติดตั้งดาต้าเบลดโมดูลด้วยเบลดแพ็ก

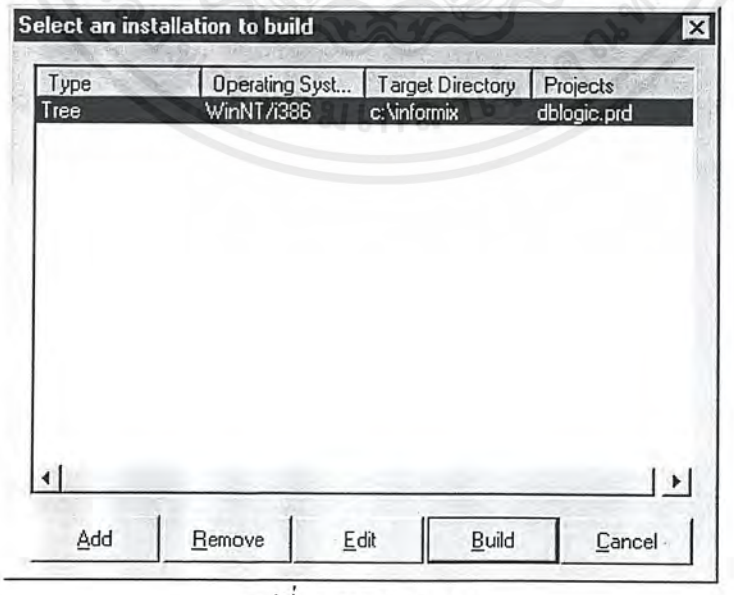
โปรแกรมเบลดแพ็กเป็นโปรแกรมที่นำดาต้าเบลดโมดูลที่ได้จากโปรแกรมเบลดสมิธมาทำการติดตั้งลงในไดเรกทอรีที่พร้อมที่จะถูกรีจิสเตอร์เข้าไปในระบบฐานข้อมูล ซึ่งไฟล์นามสกุล .PRD จะถูกสร้างจากเบลดสมิธ ในเมนู Generate Installation Package มีขั้นตอนการติดตั้งดังนี้

1. เปิดโปรแกรมเบลดแพ็กและทำการเปิดไฟล์นามสกุล .PRD แสดงได้ดังรูปที่ ก-22



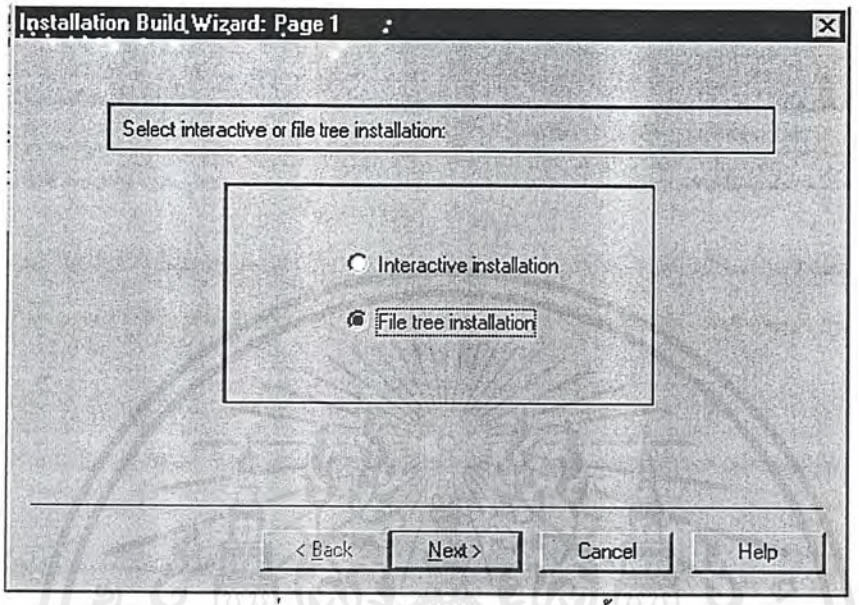
รูปที่ ก-22 โปรแกรมเบลดแพค

2. เลือกเมนู Project / Build Installtion ... จะเป็นหน้าจอการ Build แสดงได้ดังรูปที่ ก-23



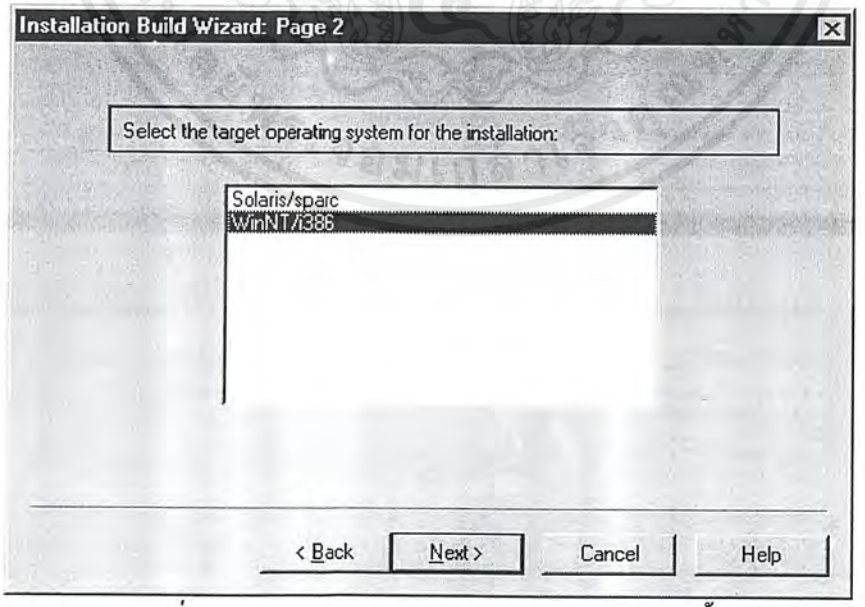
รูปที่ ก-23 การ Build

3. กดปุ่ม Edit เพื่อกำหนดรายละเอียดต่าง ๆ ของการติดตั้ง จะได้ดังรูปที่ ก-24 เป็นประเภทของการติดตั้ง ซึ่งจะแบ่งเป็น 2 ประเภทคือ Interactive Installation จะเป็นการติดตั้งแบบ Graphic ซึ่งต้องอาศัยโปรแกรม Install Shield และแบบที่ 2 จะเป็นแบบ File Tree Installation ซึ่งไม่ต้องใช้โปรแกรมอื่น ๆ จะเป็นการคัดลอกไฟล์ต่าง ๆ ไปยังไดเรกทอรีที่กำหนดโดยตรง ไม่มีการ Interactive หลังจากนั้นให้กดปุ่ม Next



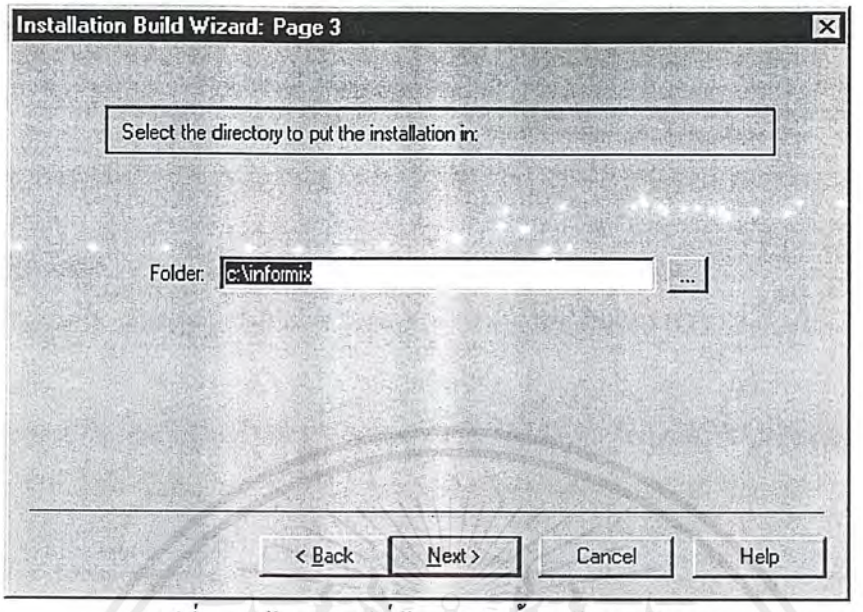
รูปที่ ก-24 การเลือกรูปแบบการติดตั้ง

4. เลือกระบบปฏิบัติการที่จะต้องทำการติดตั้งลงไป มีให้เลือก 2 ระบบ คือ Soralis / Sparc และ WiNT/i386 ให้เลือก WinNT ถ้าใช้ระบบปฏิบัติการ Windows NT อยู่ แล้วทำการกดปุ่ม Next การกำหนดแสดงได้ดังรูปที่ ก-25



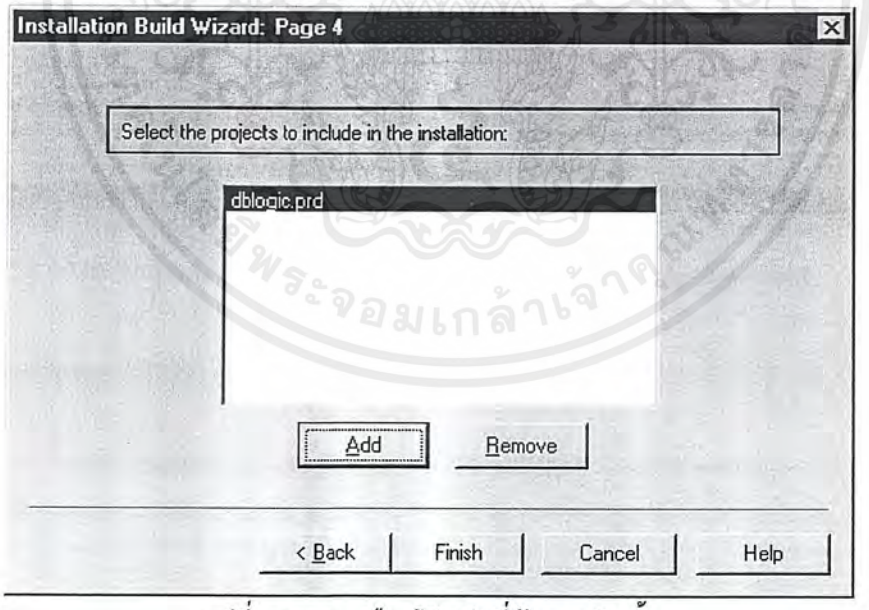
รูปที่ ก-25 การกำหนดระบบปฏิบัติการสำหรับการติดตั้ง

5. เลือกไดเรกทอรีที่จะทำการติดตั้งไฟล์ลงไป ซึ่งโดยทั่วไปจะเป็นไดเรกทอรีของอินฟอร์มิกซ์เซิร์ฟเวอร์ ดังรูปที่ ก-26 หลังจากนั้นให้กดปุ่ม Next



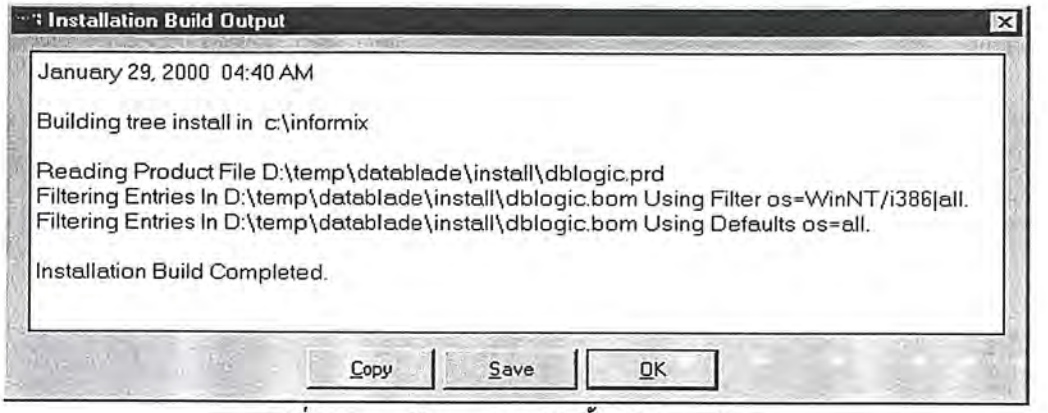
รูปที่ ก-26 ไดเรกทอรีที่ต้องการติดตั้งดาต้าเบลดโมดูล

6. หากต้องการเพิ่มโปรเจ็คอื่น ๆ ที่จะติดตั้งเข้าไปพร้อมๆ กัน สามารถเพิ่มได้โดยการกดปุ่ม Add หรือไม่ต้องการจะเอาออกก็ใช้ปุ่ม Remove ซึ่งโปรเจ็คที่จะเพิ่มเข้ามาจะต้องมีนามสกุล .PRD หลังจากที่เพิ่มเรียบร้อยแล้วให้กดปุ่ม finish เพื่อสิ้นสุดการ Edit แสดงได้ดังรูปที่ ก-27



รูปที่ ก-27 การเลือกโปรเจ็คที่ต้องการติดตั้ง

7. ทำการกดปุ่ม Build เพื่อทำการ Copy ไฟล์ที่จำเป็นต่าง ๆ เข้าไปเก็บไว้ในไดเรกทอรีที่กำหนด ซึ่งจะแสดงผลดังนี้ หลังจากนั้นกดปุ่ม OK เป็นการเสร็จสิ้นการติดตั้ง



รูปที่ ก-28 เอาต์พุตของการติดตั้งดาต้าเบลดโมดูล

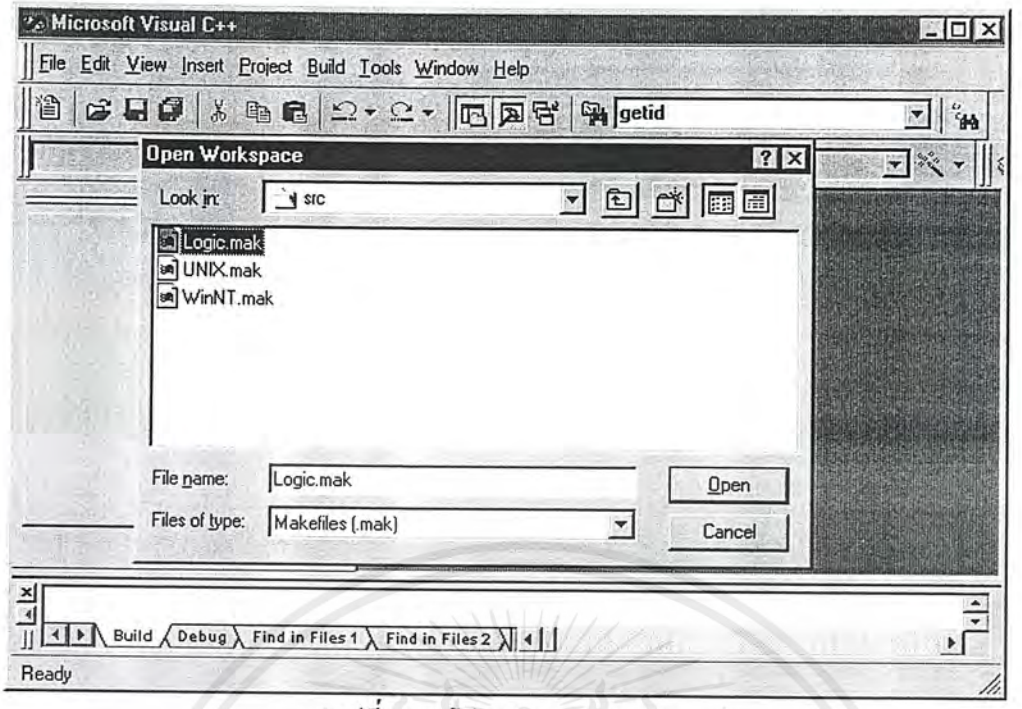
หลังจากนั้นทำการออกจากเบลดแพ็ก โปรแกรมจะถามว่าต้องการจะบันทึกการเปลี่ยนแปลงหรือไม่ ให้เลือกบันทึกเพื่อเก็บค่ารายละเอียดที่เราได้กำหนดไว้

ก-3 การสร้าง Shared Object

เมื่อทำการติดตั้งดาต้าเบลดโมดูลไฟล์ต่าง ๆ จะถูกคัดลอกเข้าไปเก็บไว้ไว้ในไดเรกทอรีที่พร้อมที่จะทำการรีจิสเตอร์เข้าไปเซิร์ฟเวอร์ โดยใช้เบลดเมเนเจอร์ แต่ยังมีไฟล์ที่เป็นไลบรารีเป็นไฟล์นามสกุล .BLD ซึ่งยังไม่ถูกคัดลอกไปด้วย เนื่องจากยังไม่มีการสร้าง ไฟล์นี้เรียกว่า Shared Object ซึ่งถ้าไม่มีไฟล์นี้จะไม่สามารถเรียกรูทีนต่าง ๆ ที่สร้างขึ้นได้ และที่สำคัญ เรายังไม่มีการเขียนโค้ดภายในของรูทีนที่เราสร้างขึ้นให้ตรงความต้องการเลย

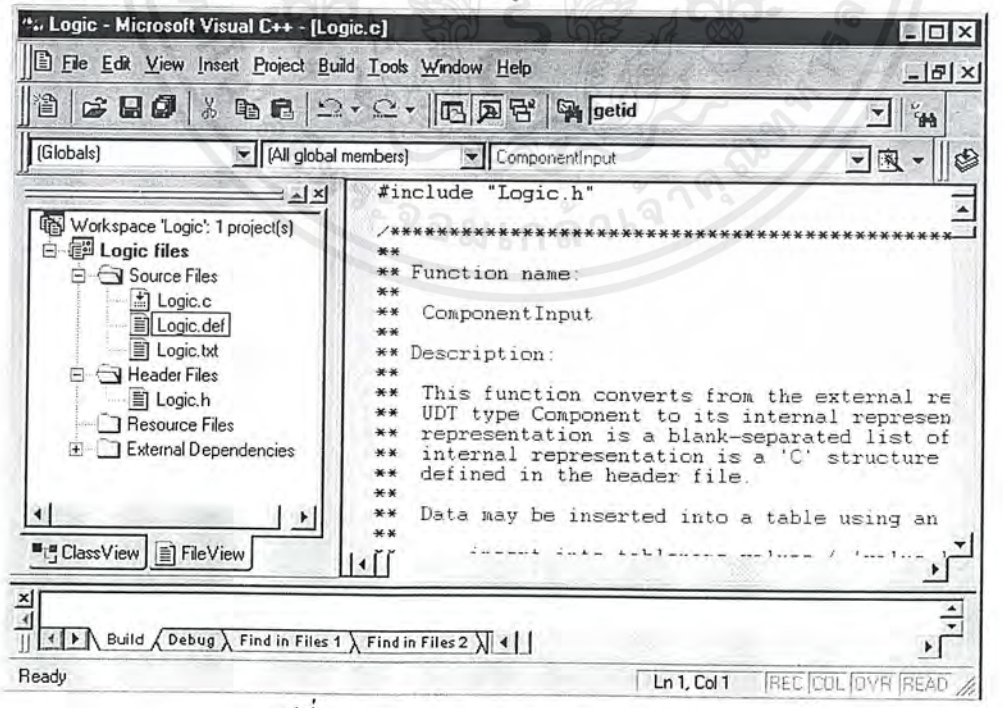
ไฟล์ไลบรารีจะสามารถสร้างจากโค้ดที่ถูกสร้างโดยเบลดสมิธ ซึ่งจะเป็นไฟล์นามสกุล .C อยู่ในไดเรกทอรี /src ของไดเรกทอรีที่สร้างลงไป การจัดสร้างไฟล์ shared object มีขั้นตอนดังต่อไปนี้

1. เปิดโปรแกรม Visual C++ (เวอร์ชัน 4 ขึ้นไป) แล้วเลือกเมนู File / Open Workspace จะมีหน้าต่าง Open Workspace ขึ้นมาแล้วหาไฟล์ที่มีนามสกุล *.MAK ในไดเรกตอรี /src ที่ถูกสร้างโดยโปรแกรมเบลดสมิธ ดังรูปที่ ก-29



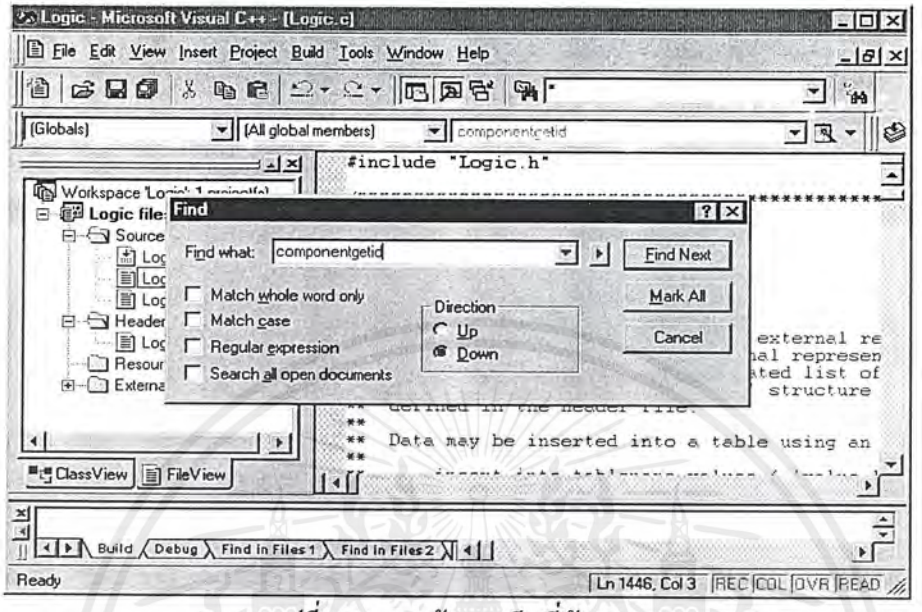
รูปที่ ก-29 โปรแกรม Visual C++

2. เมื่อเปิดไฟล์นามสกุล .MAK ขึ้นมา จะสามารถมอง Workspace ได้ 2 แบบคือ Class View และ File View เนื่องจากโค้ดที่ถูกสร้างขึ้นมาเป็นภาษาซี ไม่มีคุณสมบัติของคลาสให้เลือก File View ซึ่งไฟล์จะแบ่งออกเป็น 2 ส่วนที่มีคือ Source Files และ Head Files ในส่วนของ Source Files จะประกอบด้วยไฟล์นามสกุล .def , .c และ .txt ส่วน Header File จะเป็นไฟล์นามสกุล .h ซึ่งไฟล์ที่จะเข้าไปแก้ไขและเพิ่มเติมคือไฟล์นามสกุล .c ซึ่งเก็บรูทีนต่าง ๆ แสดงได้ดังรูปที่ ก-30



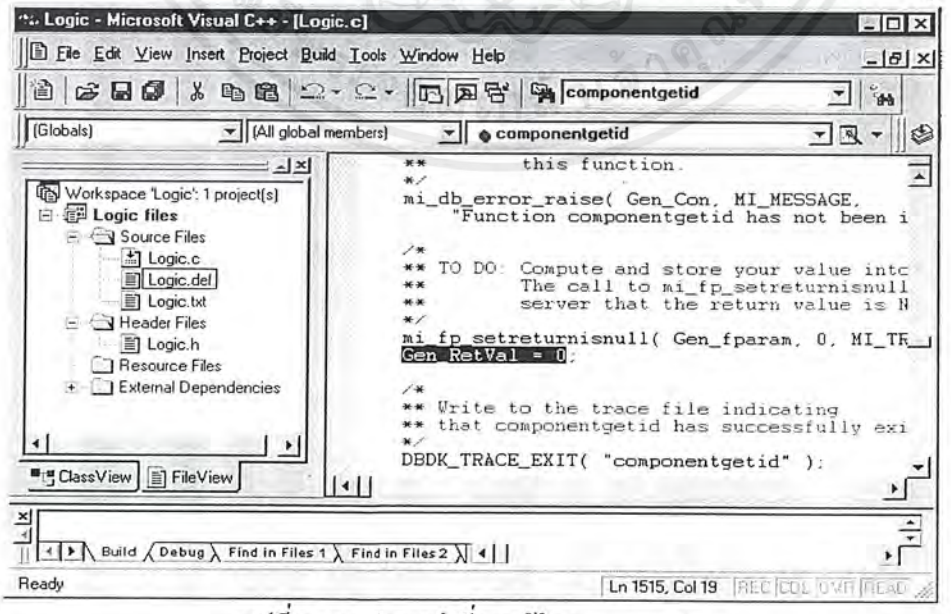
รูปที่ ก-30 Source code ของ shared object

3. ทำการแก้ไขการทำงานภายในของรูทีนที่สร้างขึ้นในเบลดสมิธ โดยการค้นหารูทีนใน Source Code เพื่อความรวดเร็วสามารถใช้ปุ่ม Ctrl + F เพื่อค้นหารูทีนที่ต้องการ เช่น ต้องการค้นหารูทีน getid เราก็ควรจำได้ว่าเรากำหนดชื่อรูทีนภาษาซีไว้อย่างไร เช่น getid กำหนดชื่อรูทีนภาษาซีเป็น componentgetid ก็ทำการค้นหาได้ดังรูปที่ ก-31



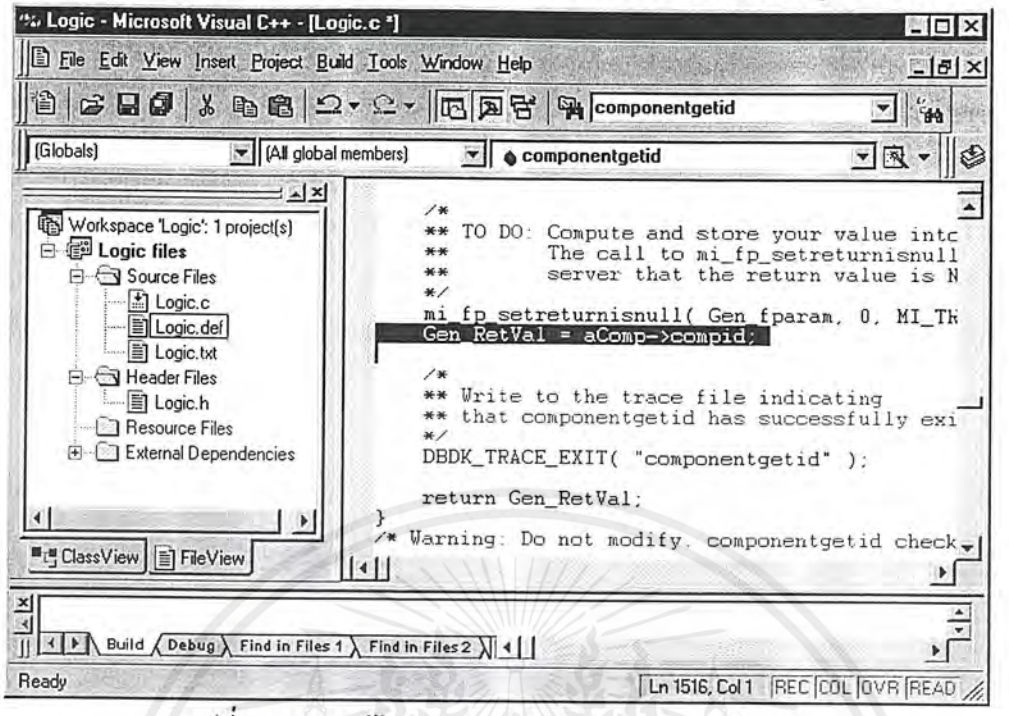
รูปที่ ก-31 การค้นหารูทีนที่ต้องการ

4. เมื่อหารูทีนที่ต้องการพบแล้ว ให้หาบรรทัดที่เขียนว่า ***TO DO:*** ซึ่งจะเป็นบรรทัดที่สามารถเขียนโค้ดภาษาซี เพื่อแสดงการทำงานที่เราต้องการลงไป ข้อสังเกตอย่างหนึ่งคือ ถ้ารูทีนนั้นเป็นฟังก์ชัน จะมีบรรทัด Gen_RetVal = 0 ซึ่งตัวแปร Gen_RetVal คือตัวแปรที่จะถูกส่งกลับไปยังผู้เรียกฟังก์ชัน เราสามารถเขียนการทำงานอื่นๆ ลงไปก่อนคำสั่งนี้ แล้วให้ค่า Gen_RetVal เท่ากับค่าที่เราต้องการให้มีการส่งกลับ แต่ถ้าเป็นโปรซีเดอร์จะไม่มีตัวแปรตัวนี้ ผู้ใช้เพียงแต่ใส่โค้ดที่ต้องการลงไปเท่านั้น ดังรูป ก-32



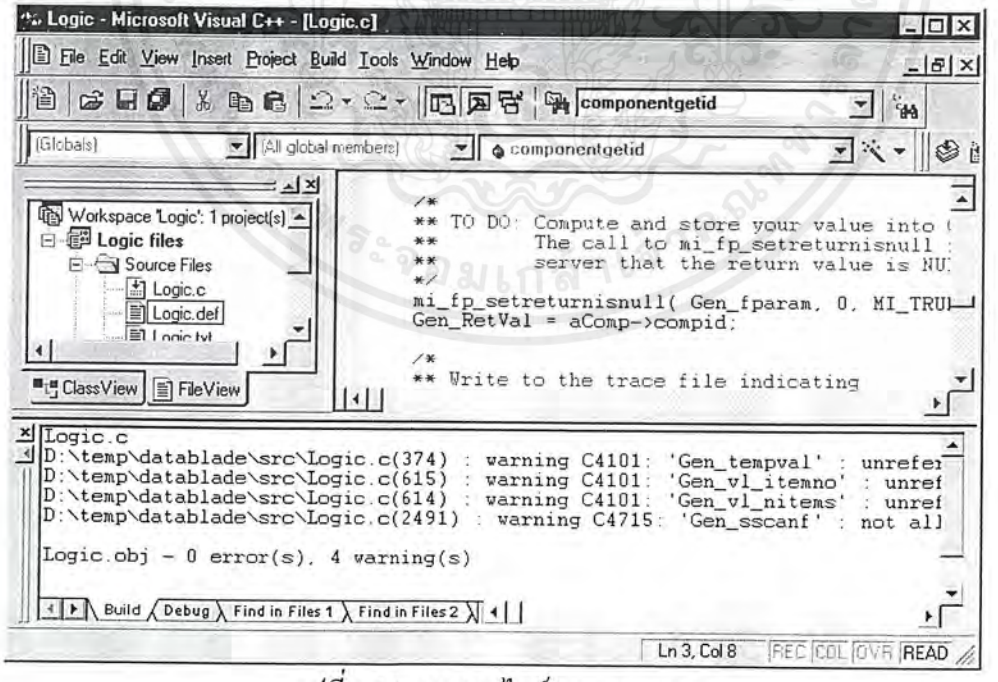
รูปที่ ก-32 ตำแหน่งที่จะแก้ไข Source code

5. สมมติว่าฟังก์ชัน getid เพื่อส่งค่าฟิลด์ compid กลับ สามารถเพิ่มค่าเข้าไปได้ดังรูปที่ ก-33



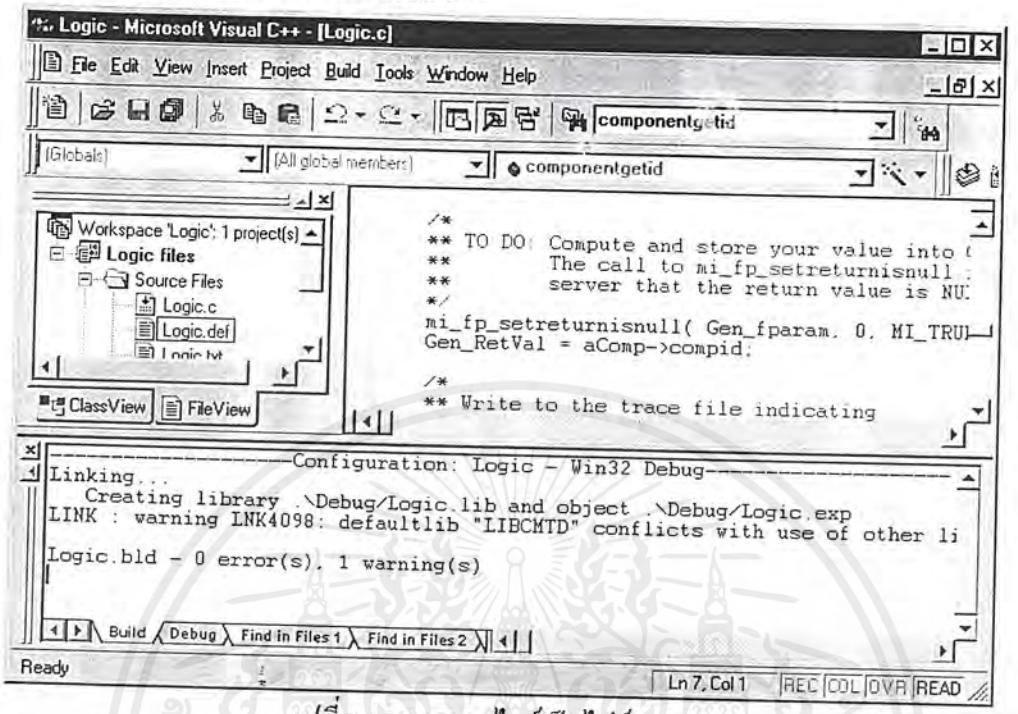
รูปที่ ก-33 การแก้ไข Source code ให้ตรงตามความต้องการ

6. เมื่อทำการแก้ไขโค้ดเรียบร้อยแล้ว ทำการ Compile Source Code โดยใช้เมนู Build / Compile filename.c หรือใช้ Ctrl+F7 เมื่อ Compile เรียบร้อย ให้ทำการตรวจสอบใน docked windows ข้างล่าง ว่ามี Error บรรทัดที่เท่าใด ถ้ามีให้ทำการแก้ไข ดังรูปที่ ก-34



รูปที่ ก-34 การคอมไพล์ Source code

7. ทำการสร้างไฟล์นามสกุล .BLD เพื่อสร้างไลบรารีไฟล์ที่จะทำการคัดลอกไปไว้ในไดเรกทอรีของอินฟอร์มิกซ์ที่ติดตั้งไว้โดยโปรแกรมเบลดแพ็ก โดยใช้เมนู Build / Build filename.bld F7 ดังรูปที่ ก-35 หลังจากนั้นทำการปิดโปรแกรม Visual C++



*รูปที่ ก-35 การคอมไพล์เป็นไฟล์ *.BLD*

ก-4 การคัดลอก Shared object เข้าไปเก็บในไดเรกตอรีการติดตั้ง

โดยทำการคัดลอกไฟล์นามสกุล .bld ไปเก็บไว้ในไดเรกทอรีที่ทำการติดตั้งโดยเบลดแพ็ก ซึ่งจะอยู่ใน Informix server dir \ extend \ datablade Module name . major . minor . revision. release

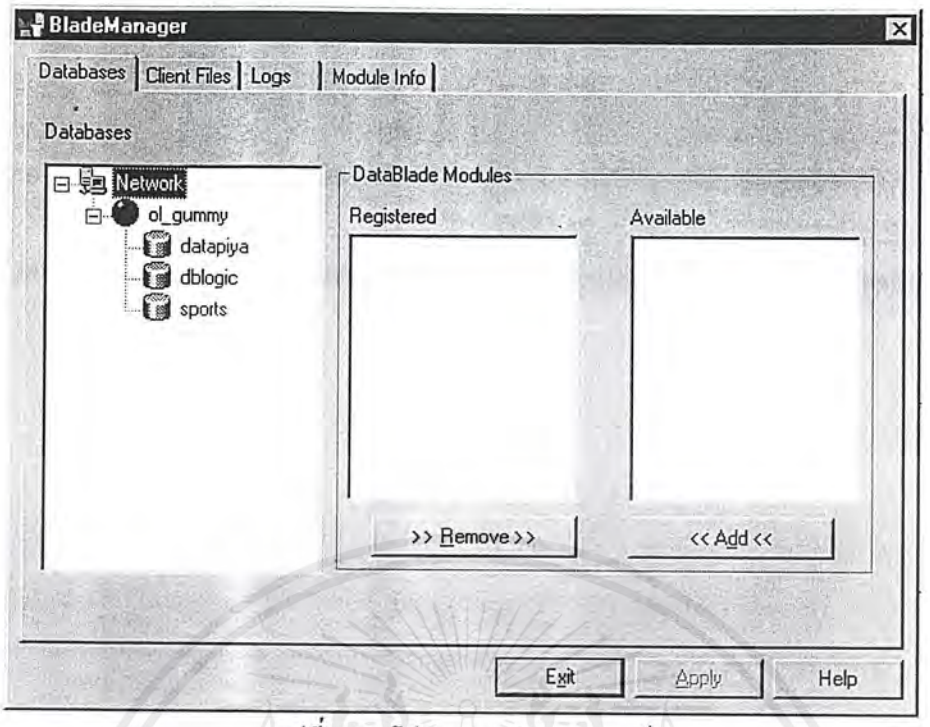
เช่น

c:\informix\extend\Logic.1.0.UC.0 \

หลังจากจากนั้นทำการเปลี่ยนชนิดของไฟล์ให้เป็น Read-only ซึ่งถ้าไม่ทำการกำหนดจะไม่สามารถทำการเรียกใช้รูทีนที่สร้างขึ้นได้เลย

ก-5 การติดตั้งดาต้าเบลดโมดูลเข้าไปในฐานข้อมูล

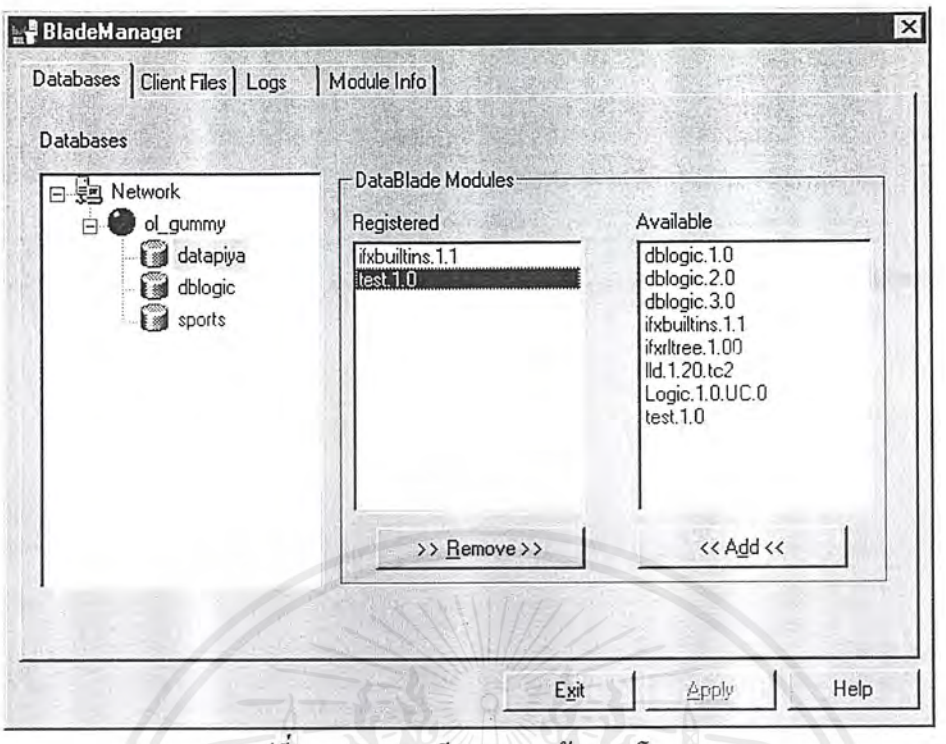
1. ทำการรันโปรแกรมเบลดเมเนเจอร์ ซึ่งเป็นโปรแกรมที่จะทำการรีจิสเตอร์ดาต้าเบลดโมดูลที่มีอยู่เข้าไปใส่ดาต้าเบสเซิร์ฟเวอร์ ซึ่งดาต้าเบลดโมดูลที่สามารถรีจิสเตอร์เข้าไปได้ จะต้องอยู่ในไดเรกทอรี Informix server dir \ extend \ เมื่อรันโปรแกรมเบลดเมเนเจอร์ จะได้หน้าต่างดังรูปที่ ก-36



รูปที่ ก-36 โปรแกรมเบลดเมเนเจอร์

2. ทำการเลือกฐานข้อมูลที่เราต้องการจะเพิ่มดาต้าเบลดโมดูลเข้าไป จะได้หน้าจอดังรูปที่ ก-37 คือ โปรแกรมจะแสดงดาต้าเบลดโมดูลที่มีอยู่แล้วในฐานข้อมูลและดาต้าเบลดโมดูลที่พร้อมที่จะทำการรีจิสเตอร์ ให้เลือกดาต้าเบลดโมดูลที่ต้องการทางขวามือ ในช่อง Available แล้วกดปุ่ม Add ถ้าต้องการนำดาต้าเบลดโมดูลที่ต้องการออกจาก Database ก็สามารถทำได้โดยการเลือกแล้วทำการกดปุ่ม Remove ทางซ้ายมือ ถ้าฐานข้อมูลที่ถูกสร้างขึ้นใหม่ ยังไม่มีดาต้าเบลดโมดูลใด ๆ อยู่เลย เราจะต้องทำการเพิ่ม ดาต้าเบลดโมดูลชื่อ ifxbuiltins.1.1 เข้าไปในฐานข้อมูลก่อน แล้วจึงเพิ่มดาต้าเบลดโมดูลที่ต้องการเข้าไปทีหลัง ข้อสังเกตสำหรับการจัดการดาต้าเบลดโมดูลคือเราไม่สามารถเพิ่มดาต้าเบลดโมดูลชื่อเดียวกันแต่ว่ามีเวอร์ชันที่ต่ำกว่าเข้าไปได้ เราจะต้อง Remove เวอร์ชันนั้นออกมาก่อน หรือปรับให้มีเวอร์ชันที่ใหม่กว่า และ เราไม่สามารถ Remove ดาต้าเบลดโมดูลที่ยังมี องค์ประกอบในฐานข้อมูลยังคงใช้ออบเจกต์ภายในดาต้าเบลดโมดูลนั้นอยู่ เช่น ยังมีตารางที่คอลัมน์หนึ่งเป็นชนิดข้อมูลแบบโอเปคที่อยู่ดาต้าเบลดโมดูลที่จะ Remove เป็นต้น

เมื่อ Add หรือ Remove ดาต้าเบลดโมดูลเสร็จแล้ว ก็ทำการกดปุ่ม Apply เพื่อทำการ Register หรือ Unregister ดาต้าเบลดโมดูลที่ต้องการ แสดงได้ดังรูปที่ ก-37



รูปที่ ก-37 รายละเอียดของดาต้าเบลดโมดูล

ข้อสำคัญอย่างหนึ่งคือ ขณะที่ใช้งานเบลดเมเนเจอร์ จะต้อง start service ของอินฟอร์มิกซ์ไว้เสมอ และในการสร้างและจัดการกับฐานข้อมูล และมีการใช้ชนิดข้อมูลหรือเรียกใช้รูทีนที่อยู่ในดาต้าเบลดโมดูลควรเขียนเป็นสคริปต์ไว้หรือโน๊ตไว้ ถ้าหากเราลืมว่าได้ตารางชื่ออะไรไว้บ้าง อาจจะทำให้ยุ่งยากในการที่จะ unregister ดาต้าเบลดโมดูลออกจากฐานข้อมูล

ภาคผนวก ข.
ตัวอย่างการจัดเก็บข้อมูล

ข-1 ตัวอย่างการจัดเก็บชนิดของอุปกรณ์

การจัดเก็บชนิดของอุปกรณ์จะจัดเก็บในตาราง component_type_tab , Pin_type_tab และตาราง Truth_tab ถ้ามีรายการอุปกรณ์ดังรูปต่อไปนี้ อยู่ในระบบ

	And gate 2 input 1 output
	Or gate 2 input 1 output
	Not gate

จะมีการจัดเก็บตารางชนิดอุปกรณ์ดังต่อไปนี้

1. ตาราง Component_type_tab

type_id	type_name	type_desc	type_input_size	type_output_size	type_size	type_delay_	picture
1	and2	and gate 2 input	2	1	40	0	<blob; length = 5887>
2	or2	or gate 2 input	2	1	40	0	<blob; length = 5932>
5	notg	not gate	1	1	40	0	<blob; length = 5831>

2. ตาราง pin_type_tab

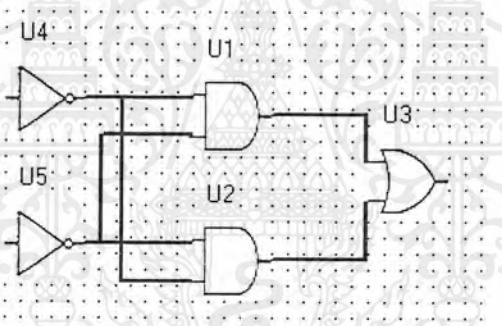
pin_type_id	type_id	pin_type_name	pin_type_desc	pin_spec	pin_type_loc
1	1	in1	input 1	0	(0, 10)
2	1	in2	input 2	0	(0, 30)
3	1	out1	output 1	1	(40, 19)
1	2	in1	input 1	0	(0, 10)
2	2	in2	input 2	0	(0, 30)
3	2	out1	output 1	1	(40, 19)
1	5	in1	input 1	0	(0, 19)
2	5	out1	output 1	1	(40, 19)

3. ตาราง Truth_tab

type_id	case	input_list	outputval
1	1	{0,0}	0
1	2	{0,1}	0
1	3	{1,0}	0
1	4	{1,1}	1
2	1	{0,0}	0
2	2	{0,1}	1
2	3	{1,0}	1
2	4	{1,1}	1
5	1	{0}	1
5	2	{1}	0

ข-2 ตัวอย่างการจัดเก็บอุปกรณ์ประกอบเป็นวงจร

เมื่อผู้ใช้ทำการสร้างวงจรที่มีการเชื่อมต่อกัน มีลักษณะดังรูป



ถ้าโปรเจ็คนี้มีชื่อโปรเจ็คคือ test3 และให้ project description คือ test project จะได้ตารางในฐานข้อมูลดังต่อไปนี้

4. ตาราง project_tab

project_id	project_name	project_desc
6	test3	test project

5. ตาราง comp_tab

project_id	comp_opaque	comp_name	comp_desc
6	(1, 1, (1920, 1080))	U1	
6	(2, 1, (1920, 2160))	U2	
6	(3, 2, (3240, 1560))	U3	
6	(4, 5, (480, 960))	U4	
6	(5, 5, (480, 2040))	U5	

6. ตาราง pin_tab

project_id	pin_opaque
6	(1, 1, 0, 0, 0)
6	(2, 1, 0, 0, 0)
6	(3, 1, 0, 1, 0)
6	(1, 2, 0, 0, 0)
6	(2, 2, 0, 0, 0)
6	(3, 2, 0, 1, 0)
6	(1, 3, 0, 0, 0)
6	(2, 3, 0, 0, 0)
6	(3, 3, 0, 1, 1)
6	(1, 4, 0, 0, 1)
6	(2, 4, 0, 1, 0)
6	(1, 5, 0, 0, 1)
6	(2, 5, 0, 1, 0)

7. ตาราง wire_tab

project_id	wire_opaque
6	(1, (1, 3), (3, 1), 0)
6	(2, (2, 3), (3, 2), 0)
6	(3, (4, 2), (1, 1), 0)
6	(4, (5, 2), (2, 1), 0)
6	(5, (4, 2), (2, 2), 0)
6	(6, (5, 2), (1, 2), 0)

ภาคผนวก ค.
ซอร์สโค้ด

ส่วนของซอร์สโค้ดจะประกอบด้วย 2 ส่วนที่สำคัญคือ ชุดคำสั่งภาษา SQL ที่อยู่ฝั่งเซิร์ฟเวอร์ และซอร์สโค้ดภาษาวิชวลเบสิกที่อยู่ฝั่งไคลเอนต์ ซึ่งทั้งสองส่วนมีความสำคัญกับระบบฐานข้อมูลแคด/แคม มีรายละเอียดดังต่อไปนี้

ค-1 ชุดคำสั่งภาษา SQL

ชุดคำสั่งภาษา SQL จะประกอบด้วยคำสั่งที่ใช้ในการสร้างตารางและทริกเกอร์ใช้งานต่าง ๆ ชุดคำสั่งที่เป็นภาษา SPL ที่ใช้ในการเขียนรูทีน ซึ่งอาจจะเป็นรูทีนที่ใช้งานทั่วไป และรูทีนที่เป็นทริกเกอร์แอคชัน รายละเอียดมีดังนี้

```
********** ชุดคำสั่งในการสร้างตาราง **********
CREATE TABLE project_tab
(       project_id SERIAL NOT NULL,
        project_name VARCHAR(20) NOT NULL,
        project_desc VARCHAR(255),
        PRIMARY KEY (project_id),
        UNIQUE (project_name)
);

CREATE TABLE component_type_tab
(       type_id  INTEGER NOT NULL ,
        type_name  VARCHAR(20) NOT NULL,
        type_desc  VARCHAR(255) ,
        type_input_size INTEGER NOT NULL,
        type_output_size INTEGER NOT NULL,
        type_size  INTEGER NOT NULL ,
        type_delay_time INTEGER NOT NULL,
        picture BLOB,
        PRIMARY KEY (type_id),
        CHECK (type_id>0 AND type_size>0)
);
```

```
CREATE  TABLE pin_type_tab
(       pin_type_id  INTEGER NOT NULL,
        type_id      INTEGER NOT NULL,
        pin_type_name  VARCHAR(20) NOT NULL,
        pin_type_desc   VARCHAR(255),
        pin_spec    INTEGER,
        pin_type_loc   LOC,
        PRIMARY KEY (pin_type_id,type_id),
        FOREIGN KEY (type_id) REFERENCES component_type_tab(type_id)
        ON DELETE CASCADE,
        CHECK (pin_type_id>0 AND pin_spec IN (0,1))
);

CREATE TABLE comp_tab
(       project_id INTEGER NOT NULL,
        comp_opaque COMPONENT NOT NULL,
        comp_name   VARCHAR(20) ,
        comp_desc   VARCHAR(255),
        PRIMARY KEY(project_id,comp_opaque)
);

CREATE TABLE pin_tab
(       project_id INTEGER NOT NULL,
        pin_opaque PIN NOT NULL,
        PRIMARY KEY(project_id,pin_opaque),
        FOREIGN KEY (project_id) REFERENCES project_tab(project_id) ON DELETE CASCADE
);

CREATE TABLE wire_tab
(       project_id  INTEGER NOT NULL,
        wire_opaque  WIRE NOT NULL,
        wire_name   VARCHAR(20),
        name_loc            LOC,
```

```
        wire_locs     LIST(LOC NOT NULL),
        PRIMARY KEY (project_id,wire_opaque),
        FOREIGN KEY (project_id) REFERENCES project_tab(project_id) ON DELETE CASCADE
);

CREATE TABLE truth_tab
(
        type_id   INTEGER NOT NULL,
        case      INTEGER NOT NULL,
        input_list LIST (INTEGER NOT NULL) NOT NULL,
        outputval INTEGER,
        primary key (type_id,case)
);
```

********** ชุดคำสั่งในการสร้างทริกเกอร์และทริกเกอร์แอคชันรูทีน **********

```
CREATE PROCEDURE InsNewPin(Proj_ID INTEGER,Comp Component)
        DEFINE  Comp_id,Comp_type ,Pin_id,Pspec  INTEGER;
        DEFINE  tempPin Pin;
        LET Comp_id=getid(Comp);
        LET Comp_type=gettype(Comp);
        LET tempPin='0 0 0 0 0';
        FOREACH  InsPinCursor FOR
                SELECT pin_type_id,pin_spec  INTO Pin_id,Pspec  FROM pin_type_tab
                        WHERE type_id=Comp_type
                EXECUTE PROCEDURE setall(tempPin,Pin_id,Comp_id,0,Pspec,1);
                INSERT  INTO Pin_tab VALUES (Proj_ID,tempPin) ;
        END FOREACH
END PROCEDURE;

CREATE TRIGGER InsComp
        INSERT ON Comp_tab
        REFERENCING NEW as new
        FOR EACH ROW
          (
                EXECUTE PROCEDURE InsNewPin(new.project_id,new.comp_opaque)
          );

CREATE PROCEDURE DelRelatePin(Proj_ID INTEGER,Comp Component)
        DEFINE  Comp_id  INTEGER;
        LET Comp_id=getid(Comp);
        DELETE FROM pin_tab
                WHERE project_id=Proj_ID and getcomp(Pin_opaque)=Comp_id;
END PROCEDURE;
```

```
CREATE TRIGGER DelComp
        DELETE ON Comp_tab
        REFERENCING OLD as old
        FOR EACH ROW
          (
                EXECUTE PROCEDURE DelRelatePin(old.Project_id,old.comp_opaque)
          );

CREATE TRIGGER Delproject
        DELETE ON project_tab
        REFERENCING OLD as old
        FOR EACH ROW
        (
                DELETE FROM comp_tab  WHERE old.project_id=project_id
        );

********** ชุดคำสั่งในการสร้างรูทีนใช้งานทั่วไป **********
CREATE PROCEDURE PinUpdateValue(Proj_id INTEGER,Cid INTEGER,Pid INTEGER,
                                Val  INTEGER)
        DEFINE Ptemp Pin;
        SELECT pin_opaque INTO Ptemp  FROM pin_tab
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
        EXECUTE PROCEDURE setvalue(Ptemp,Val);
        UPDATE  pin_tab
                SET pin_opaque=Ptemp
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
END PROCEDURE;
```

```
CREATE PROCEDURE PinUpdateTerm(Proj_id INTEGER,Cid INTEGER,Pid INTEGER,
                                        Term INTEGER)
        DEFINE Ptemp Pin;
        SELECT pin_opaque INTO Ptemp  FROM pin_tab
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
        EXECUTE PROCEDURE setterm(Ptemp,Term);
        UPDATE  pin_tab
                SET pin_opaque=Ptemp
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
END PROCEDURE;

CREATE PROCEDURE WireUpdateValue(Proj_id INTEGER,Wid INTEGER,Val INTEGER)
        DEFINE Wtemp Wire;
        SELECT wire_opaque INTO Wtemp  FROM wire_tab
                WHERE project_id=Proj_id and getid(wire_opaque)=Wid;
        EXECUTE PROCEDURE setvalue(Wtemp,Val);
        UPDATE  wire_tab
                SET wire_opaque=Wtemp
                WHERE project_id=Proj_id and getid(wire_opaque)=Wid;
END PROCEDURE;

CREATE FUNCTION  PinGetValue(Proj_id INTEGER , Cid INTEGER , Pid INTEGER)
        RETURNING INTEGER;
        DEFINE Ptemp Pin;
        SELECT pin_opaque INTO Ptemp  FROM pin_tab
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
        RETURN getvalue(Ptemp);
END FUNCTION;
```

```
CREATE FUNCTION PinGetTerm(Proj_id INTEGER,Cid INTEGER,Pid INTEGER)
        RETURNING INTEGER;
        DEFINE Ptemp Pin;
        SELECT pin_opaque INTO Ptemp  FROM pin_tab
                WHERE project_id=Proj_id and getcomp(pin_opaque)=Cid
                        and  getid(pin_opaque)=Pid;
        RETURN getterm(Ptemp);
END FUNCTION;

CREATE FUNCTION WireGetValue(Proj_id INTEGER,Wid  INTEGER) RETURNING INTEGER;
        DEFINE Wtemp Wire;
        SELECT wire_opaque INTO Wtemp  FROM wire_tab
                WHERE project_id=Proj_id and getid(wire_opaque)=Wid;
        RETURN getvalue(Wtemp);
END FUNCTION;

CREATE FUNCTION  InsertNewComp(Proj_id INTEGER,Comp_id INTEGER,
                        Comp_type INTEGER,Xpos INTEGER,Ypos INTEGER,
                        Comp_name CHAR(20),Comp_desc CHAR(255))
        RETURNING INTEGER
        DEFINE tcomp component;
        DEFINE tnum INTEGER;
        SELECT count(*) INTO tnum FROM project_tab WHERE project_id=Proj_id;
        IF (tnum=0) THEN
                return -998;
        END IF;
        SELECT count(*)  INTO tnum FROM comp_tab WHERE project_id=Proj_id AND
                getid(comp_opaque)=Comp_id;
        IF (tnum>0) THEN
                return -999;
        END IF;
        LET tcomp='0 0 0 0';
        EXECUTE PROCEDURE setid(tcomp,Comp_id);
        EXECUTE PROCEDURE settype(tcomp,Comp_type);
```

```
        EXECUTE PROCEDURE setxpos(tcomp,Xpos);
        EXECUTE PROCEDURE setypos(tcomp,Ypos);
        insert into comp_tab values(Proj_id,tcomp,Comp_name,Comp_desc);
        RETURN 0;
END FUNCTION;

CREATE FUNCTION  InsertNewWire(Proj_id INTEGER,Wire_id INTEGER,C1 INTEGER,
                P1 INTEGER,C2 INTEGER,P2 INTEGER)
        RETURNING INTEGER
        DEFINE twire wire;
        DEFINE tnum INTEGER;
        SELECT count(*) INTO tnum FROM project_tab WHERE project_id=Proj_id;
        IF (tnum=0) THEN
                return -998;
        END IF;
        SELECT count(*)  INTO tnum FROM wire_tab WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wire_id;
        IF (tnum>0) THEN
                return -999;
        END IF;
        LET twire='0 0 0 0 0 0';
        EXECUTE PROCEDURE setid(twire,Wire_id);
        EXECUTE PROCEDURE setfirstcomp(twire,C1);
        EXECUTE PROCEDURE setfirstpin(twire,P1);
        EXECUTE PROCEDURE setseccomp(twire,C2);
        EXECUTE PROCEDURE setsecpin(twire,P2);
        INSERT INTO wire_tab values (Proj_id,twire,' ','0 0',"LIST{}");
        RETURN 0;
END FUNCTION;

CREATE FUNCTION  InsertWireLoc(Proj_id INTEGER,Wid INTEGER, Xpos INTEGER,
                Ypos INTEGER) RETURNING INTEGER
        DEFINE tnum INTEGER;
        DEFINE wloc loc;
```

```
        DEFINE templocs LIST(loc NOT NULL);
        SELECT count(*) INTO tnum FROM project_tab WHERE project_id=Proj_id;
        IF (tnum=0) THEN
                return -998;
        END IF;
SELECT count(*)  INTO tnum FROM wire_tab WHERE project_id=Proj_id AND
         Getid (wire_opaque)=Wid;
        IF (tnum=0) THEN
                return -997;
        END IF;
        LET wloc ='0 0 ';
        EXECUTE PROCEDURE setxpos(wloc,Xpos);
        EXECUTE PROCEDURE setypos(wloc,Ypos);
        SELECT wire_locs INTO templocs  FROM wire_tab WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wid;
        INSERT  INTO TABLE(templocs) VALUES(wloc);
        UPDATE wire_tab SET wire_locs=templocs WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wid;
        RETURN 0;
END FUNCTION;

CREATE FUNCTION  ClearWireLoc(Proj_id INTEGER,Wid INTEGER)
        DEFINE tnum INTEGER;
        DEFINE templocs LIST(loc NOT NULL);
        SELECT count(*) INTO tnum FROM project_tab WHERE project_id=Proj_id;
        IF (tnum=0) THEN
                return -998;
        END IF;
        SELECT count(*)  INTO tnum FROM wire_tab WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wid;
        IF (tnum=0) THEN
                return -997;
        END IF;
        LET templocs="LIST{}";
```

```
        UPDATE wire_tab SET wire_locs=templocs WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wid;
        RETURN 0;
END FUNCTION;

CREATE FUNCTION GetWireLoc(Proj_id INTEGER,Wid INTEGER)
        RETURNING VARCHAR(255)
        DEFINE tnum,x,y INTEGER;
        DEFINE stemp VARCHAR(255);
        DEFINE wloc loc;
        DEFINE templocs LIST(loc NOT NULL);
        LET stemp="";
        SELECT wire_locs INTO templocs FROM wire_tab WHERE project_id=Proj_id
                AND getid(wire_opaque)=Wid;
        FOREACH selectloc FOR SELECT * INTO wloc FROM TABLE(templocs)
                LET stemp=stemp || " ";
                LET x=Getxpos(wloc);
                LET y=Getypos(wloc);
                LET stemp=stemp || x || "&" || y || " ";
        END FOREACH;
        return stemp;
END FUNCTION;

CREATE FUNCTION tosequence(alist LIST(INTEGER NOT NULL) )
        RETURNING VARCHAR(255)
        DEFINE stemp VARCHAR(255);
        DEFINE ltemp INTEGER;
        LET stemp="";
        FOREACH selectloc FOR SELECT * INTO ltemp FROM TABLE(alist)
                LET stemp=stemp || ltemp || " ";
        END FOREACH;
        return stemp;
END FUNCTION;
```

```
******************** คำสั่งในการดร็อปสิ่งต่าง ๆ ในฐานข้อมูล ********************
DROP PROCEDURE InsNewPin(Integer,component);
DROP PROCEDURE DelRelatePin(Integer,component);
DROP TRIGGER insComp;
DROP TRIGGER DelComp;
DROP TRIGGER Delproject;
DROP TABLE project_tab;
DROP TABLE component_type_tab;
DROP TABLE pin_type_tab;
DROP TABLE comp_tab;
DROP TABLE pin_tab;
DROP TABLE wire_tab;
DROP TABLE truth_tab;
DROP PROCEDURE PinUpdateValue(integer,integer,integer,integer);
DROP PROCEDURE PinUpdateTerm(integer,integer,integer,integer);
DROP PROCEDURE WireUpdateValue(integer,integer,integer);
DROP FUNCTION PinGetValue(integer,integer,integer);
DROP FUNCTION PinGetTerm(integer,integer,integer);
DROP FUNCTION WireGetValue(integer,integer);
DROP FUNCTION InsertNewComp(INTEGER,INTEGER,INTEGER, INTEGER, INTEGER,
                            CHAR(20), CHAR(255)) ;
DROP FUNCTION InsertNewWire (INTEGER,INTEGER,INTEGER,INTEGER,
                            INTEGER,INTEGER);
DROP FUNCTION InsertWireLoc(INTEGER,INTEGER,  INTEGER,  INTEGER);
DROP FUNCTION  ClearWireLoc(INTEGER,INTEGER);
DROP FUNCTION  GetWireLoc( INTEGER, INTEGER) ;
DROP FUNCTION  tosequence(alist  LIST(INTEGER NOT NULL) ) ;
```

```
********** คำสั่งในการใส่ค่าเข้าไปในตารางที่เป็นไทพ์ **********
insert into component_type_tab
values(1,'and2','and gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\and2.jpg','Server'));
insert into component_type_tab
values(2,'or2','or gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\or2.jpg','Server'));
insert into component_type_tab
values(3,'nand2','nand gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\nand2.jpg','Server'));
insert into component_type_tab
values(4,'nor2','nor gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\nor2.jpg','Server'));
insert into component_type_tab
values(5,'notg','not gate',1,1,40,0,FILETOBLOB('c:\informix\dblogic\notg.jpg','Server'));
insert into component_type_tab
values(6,'and3','and gate 3 input',3,1,40,0,FILETOBLOB('c:\informix\dblogic\and3.jpg','Server'));
insert into component_type_tab
values(7,'or3','or gate 3 input',3,1,40,0,FILETOBLOB('c:\informix\dblogic\or3.jpg','Server'));
insert into component_type_tab
values(8,'nand3','nand gate 3 input',3,1,40,0,FILETOBLOB('c:\informix\dblogic\nand3.jpg','Server'));
insert into component_type_tab
values(9,'nor3','nor gate 3 input',3,1,40,0,FILETOBLOB('c:\informix\dblogic\nor3.jpg','Server'));
insert into component_type_tab
values(10,'xor2','xor gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\xor2.jpg','Server'));
insert into component_type_tab
values(11,'xnor2','xnor gate 2 input',2,1,40,0,FILETOBLOB('c:\informix\dblogic\xnor2.jpg','Server'));

insert into pin_type_tab  values(1,1,'in1','input 1',      0,'0 10');
insert into pin_type_tab  values(2,1,'in2','input 2',      0,'0 30');
insert into pin_type_tab  values(3,1,'out1','output 1',    1,'40 19');
insert into pin_type_tab  values(1,2,'in1','input 1',      0,'0 10');
insert into pin_type_tab  values(2,2,'in2','input 2',      0,'0 30');
insert into pin_type_tab  values(3,2,'out1','output 1',    1,'40 19');
insert into pin_type_tab  values(1,3,'in1','input 1',      0,'0 10');
insert into pin_type_tab  values(2,3,'in2','input 2',      0,'0 30');
insert into pin_type_tab  values(3,3,'out1','output 1',    1,'40 19');
insert into pin_type_tab  values(1,4,'in1','input 1',      0,'0 10');
insert into pin_type_tab  values(2,4,'in2','input 2',      0,'0 30');
insert into pin_type_tab  values(3,4,'out1','output 1',    1,'40 19');
```

```
insert into pin_type_tab  values(1,5,'in1','input 1',        0,'0 19');
insert into pin_type_tab  values(2,5,'out1','output 1',      1,'40 19');
insert into pin_type_tab  values(1,6,'in1','input 1',        0,'0 10');
insert into pin_type_tab  values(2,6,'in2','input 2',        0,'0 19');
insert into pin_type_tab  values(3,6,'in3','input3',         0,'0 29');
insert into pin_type_tab  values(4,6,'out1','output 1',      1,'40 19');
insert into pin_type_tab  values(1,7,'in1','input 1',        0,'0 10');
insert into pin_type_tab  values(2,7,'in2','input 2',        0,'0 19');
insert into pin_type_tab  values(3,7,'in3','input3',         0,'0 29');
insert into pin_type_tab  values(4,7,'out1','output 1',      1,'40 19');
insert into pin_type_tab  values(1,8,'in1','input 1',        0,'0 10');
insert into pin_type_tab  values(2,8,'in2','input 2',        0,'0 19');
insert into pin_type_tab  values(3,8,'in3','input3',         0,'0 29');
insert into pin_type_tab  values(4,8,'out1','output 1',      1,'40 19');
insert into pin_type_tab  values(1,9,'in1','input 1',        0,'0 10');
insert into pin_type_tab  values(2,9,'in2','input 2',        0,'0 19');
insert into pin_type_tab  values(3,9,'in3','input3',         0,'0 29');
insert into pin_type_tab  values(4,9,'out1','output 1',      1,'40 19');
insert into pin_type_tab  values(1,10,'in1','input 1',       0,'0 10');
insert into pin_type_tab  values(2,10,'in2','input 2',       0,'0 30');
insert into pin_type_tab  values(3,10,'out1','output 1',     1,'40 19');
insert into pin_type_tab  values(1,11,'in1','input 1',       0,'0 10');
insert into pin_type_tab  values(2,11,'in2','input 2',       0,'0 30');
insert into pin_type_tab  values(3,11,'out1','output 1',     1,'40 19');

insert into truth_tab values (1,1,"LIST{0,0}",0);
insert into truth_tab values (1,2,"LIST{0,1}",0);
insert into truth_tab values (1,3,"LIST{1,0}",0);
insert into truth_tab values (1,4,"LIST{1,1}",1);
insert into truth_tab values (2,1,"LIST{0,0}",0);
insert into truth_tab values (2,2,"LIST{0,1}",1);
insert into truth_tab values (2,3,"LIST{1,0}",1);
insert into truth_tab values (2,4,"LIST{1,1}",1);
insert into truth_tab values (3,1,"LIST{0,0}",1);
insert into truth_tab values (3,2,"LIST{0,1}",1);
insert into truth_tab values (3,3,"LIST{1,0}",1);
```

```
insert into truth_tab values (3,4,"LIST{1,1}",0);
insert into truth_tab values (4,1,"LIST{0,0}",1);
insert into truth_tab values (4,2,"LIST{0,1}",0);
insert into truth_tab values (4,3,"LIST{1,0}",0);
insert into truth_tab values (4,4,"LIST{1,1}",0);
insert into truth_tab values (5,1,"LIST{0}",1);
insert into truth_tab values (5,2,"LIST{1}",0);
insert into truth_tab values (6,1,"LIST{0,0,0}",0);
insert into truth_tab values (6,2,"LIST{0,0,1}",0);
insert into truth_tab values (6,3,"LIST{0,1,0}",0);
insert into truth_tab values (6,4,"LIST{0,1,1}",0);
insert into truth_tab values (6,5,"LIST{1,0,0}",0);
insert into truth_tab values (6,6,"LIST{1,0,1}",0);
insert into truth_tab values (6,7,"LIST{1,1,0}",0);
insert into truth_tab values (6,8,"LIST{1,1,1}",1);
insert into truth_tab values (7,1,"LIST{0,0,0}",0);
insert into truth_tab values (7,2,"LIST{0,0,1}",1);
insert into truth_tab values (7,3,"LIST{0,1,0}",1);
insert into truth_tab values (7,4,"LIST{0,1,1}",1);
insert into truth_tab values (7,5,"LIST{1,0,0}",1);
insert into truth_tab values (7,6,"LIST{1,0,1}",1);
insert into truth_tab values (7,7,"LIST{1,1,0}",1);
insert into truth_tab values (7,8,"LIST{1,1,1}",1);
insert into truth_tab values (8,1,"LIST{0,0,0}",1);
insert into truth_tab values (8,2,"LIST{0,0,1}",1);
insert into truth_tab values (8,3,"LIST{0,1,0}",1);
insert into truth_tab values (8,4,"LIST{0,1,1}",1);
insert into truth_tab values (8,5,"LIST{1,0,0}",1);
insert into truth_tab values (8,6,"LIST{1,0,1}",1);
insert into truth_tab values (8,7,"LIST{1,1,0}",1);
insert into truth_tab values (8,8,"LIST{1,1,1}",0);
insert into truth_tab values (9,1,"LIST{0,0,0}",1);
insert into truth_tab values (9,2,"LIST{0,0,1}",0);
insert into truth_tab values (9,3,"LIST{0,1,0}",0);
insert into truth_tab values (9,4,"LIST{0,1,1}",0);
insert into truth_tab values (9,5,"LIST{1,0,0}",0);
```

```
insert into truth_tab values (9,6,"LIST{1,0,1}",0);
insert into truth_tab values (9,7,"LIST{1,1,0}",0);
insert into truth_tab values (9,8,"LIST{1,1,1}",0);
insert into truth_tab values (10,1,"LIST{0,0}",0);
insert into truth_tab values (10,2,"LIST{0,1}",1);
insert into truth_tab values (10,3,"LIST{1,0}",1);
insert into truth_tab values (10,4,"LIST{1,1}",0);
insert into truth_tab values (11,1,"LIST{0,0}",1);
insert into truth_tab values (11,2,"LIST{0,1}",0);
insert into truth_tab values (11,3,"LIST{1,0}",0);
insert into truth_tab values (11,4,"LIST{1,1}",1);
```

ค-2 ซอร์สโค้ดของแอปพลิเคชันโปรแกรม

แอปพลิเคชันจะถูกเขียนขึ้นด้วยภาษาไมโครซอฟท์วิชวลเบสิกเวอร์ชัน 6.0 ซึ่งมีรายละเอียดมาก แบ่งออกเป็นหลาย ๆ ฟอร์ม และหลาย ๆ รูทีน มีจำนวนบรรทัดมาก จึงไม่สามารถนำมาใส่ในภาคผนวกได้

หนังสืออ้างอิง

[1] Michael Stonebraker , Paul Brown, "*Object-Relational DBMS Tracking the next great wave*" , Morgan Kaufmann Publishers, 1998.

[2] DS Batory,Won Kim , *"Modeling Concept for VLSI Object"* , Preceedings of ACM Transactions on Database Systems , Vol.10 , No.3 , September 1985 , Pages 322-346.

[3] Anoop Singhal , Nish Parikh , Dave Dutt , and Chi Yuan Lo , *"A Data Model and Architecture for VLSI/CAD"* , AT&T Bell Laboratory

[4] Richard S.Sandige , *"Modern digital circuit"* . McGraw-Hill Publishing Company , 1990

[5] Informix software inc. , *"Informix Universal Server : Informix Guide to SQL Tutorial"* , Informix press,1997.

[6] Informix software inc , *"Informix Universal Server : Informix Guild to SQL Syntax"* , Informix press , 1997.

[7] Informix software inc , *"DataBlade Developers Kit : User's Guide"* ,Informix press,1997.

[8] Informix software inc , *"Extending Informix Universal Server : User-Defined Routines"* , Informix press,1997.

[9] Informix software inc , *"Informix Data Director for Visual Basic"* , Informix press,1997.

[10] Edward Louis Brann , *"Digital Computer Design : Logic , Circuitry and Synthesis"* , NewYork Academic Press , 1963.

[11] C.J.Date C.J. (1986) , "*An introduction to Database System* " , Addison- Wesley press , 1986.