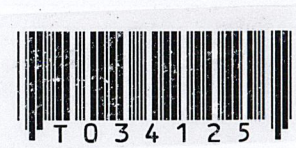
การจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์

STORING DIGITAL CIRCUIT BY OBJECT-RELATIONAL DATABASE



โดย

นางสาว พิมพ์ผกา อังสวานนท์

นาย ภุชงค์ ตั้งเจตน์

อาจารย์ที่ปรึกษา

รศ.ดร. ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์

STORING DIGITAL CIRCUIT BY OBJECT-RELATIONAL DATABASE

ผู้จัดทำ

1.นางสาว พิมพ์ผกา	อังสวานนท์	รหัสประจำตัว	38014347
2.นาย ภุชงค์	ตั้งเจตน์	รหัสประจำตัว	38014378

อาจารย์ที่ปรึกษา

(รศ.ดร. ศุภมิตร จิตตะยโศธร)

การจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์

นางสาว พิมพ์ผกา อังสวานนท์ 38014347
นาย ภุชงค์ ตั้งเจตน์ 38014378
รศ.ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

โมเดลของระบบฐานข้อมูลที่มีใช้กันอยู่ในปัจจุบันเป็นฐานข้อมูลเชิงสัมพันธ์ (Relational Database) แต่ฐานข้อมูลเชิงสัมพันธ์นี้มีข้อจำกัดในหลายๆด้าน เช่น การใช้งานกับชนิดข้อมูลจะมีเพียงแต่ชนิดข้อมูลที่ง่ายๆเช่น Integer , Character และ String เป็นต้นซึ่งไม่สามารถรองรับกับการใช้งานฐานข้อมูลในปัจจุบันที่ได้มีการใช้งานข้อมูลชนิดที่มีความซับซ้อนมากกว่า เช่น ข้อมูลทางมัลติมีเดีย ข้อมูล Geographic Information System (GIS) และไม่สามารถจัดการกับข้อมูลที่เป็นออบเจกต์ได้เพราะระบบฐานข้อมูลเชิงสัมพันธ์สามารถแสดงข้อมูลได้แต่เพียงตาราง ด้วยเหตุนี้จึงมีผู้ที่พยายามพัฒนาโมเดลอื่นๆเพื่อ ที่จะมาทดแทนข้อเสียของโมเดลนี้

วิทยานิพนธ์ฉบับนี้เป็นการนำระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational) มาเป็นโมเดลข้อมูล ซึ่งโมเดลนี้เป็นโมเดลที่ได้แนวคิดมาจากโมเดลฐานข้อมูลเชิงสัมพันธ์และแนวคิดเชิงวัตถุ (Object-Oriented Concept) ซึ่งทำให้โมเดลนี้มีความสามารถทั้งในทางฐานข้อมูลเชิงสัมพันธ์ ได้แก่ การจัดการข้อมูลที่มีประสิทธิภาพ ความสามารถในการเข้าถึงข้อมูลของภาษา SQL และความสามารถในทางแนวคิดเชิงวัตถุ ได้แก่ ความสามารถในการปกป้องข้อมูล (Encapsulation) การถ่ายทอดคุณลักษณะ (Inheritance) และโพลิมอร์ฟิซึ่ม

โดยฐานข้อมูลเชิงวัตถุสัมพันธ์นี้จะยังคงมองเห็นข้อมูลเป็นตารางในแบบเดียวกับฐานข้อมูลเชิงสัมพันธ์ โดยสามารถมีแอททริบิว (Attribute) เป็นแอททริบิวที่สามารถแบ่งย่อยได้อีก (Non-atomic Attribute) ในคอลัมน์สามารถเป็นได้ทั้งออบเจกต์หรือเป็นค่าหลายๆค่า (Multiple Value) เช่น ลิสท์ (List) หรือ เซต (Set) โดยทั่วไปโมเดลที่เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถใช้งานเสมือนเป็นฐานข้อมูลเชิงสัมพันธ์ได้

ในวิทยานิพนธ์ฉบับนี้ได้ทดลองสร้างโปรแกรมประยุกต์ (Application) ที่นำโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์มาใช้ในการจัดการกับข้อมูลที่เป็นออบเจกต์ซึ่งได้แก่วงจรเกททางดิจิตอลเพื่อเป็นกรณีศึกษาสำหรับการพัฒนาโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์

STORING DIGITAL CIRCUIT BY OBJECT-RELATIONAL DATABASE

Pimphaka Ungsvanonda 38014347

Puchong Tangjade 38014378

Assoc. Prof. Dr. Suphamit chittayasothonAdvisor

ABSTRACT

Relational Database is the database model that is widely used now but it has some limitations such as data type,which provides only simple type,integer,character,and string.In case that,it can not support some kind of application using complex data type,multimedia data or geographic information.Because of representing in table form,it is not appropriate for storing object or non-atomic data.However,some researchers try to develop other model which will instead of some disadvantage of this model.

This thesis is concerned with using object-relational database,which is combined with concept of object-oriented database and relational database,be a data model.According to having feature of both relational and object-oriented concept,this model manage data efficiently and has a capability for accessing data by SQL command.Not only having feature of SQL language it also has an advantage of object-oriented concept such as encapsulation,inheritance and polymorphism .

By the way,object-relational database still manage data in form of a relation,which like relational database,and its column can have non-atomic attribute,object or multiple value such as set and list.It can normally manipulate data as if it is a relational database.

In thesis,it develops application that is used object-relational database model for managing object data type,digital circuit is our case study in term of applying data model with real world appliaction.We name this project " STORING DIGITAL CIRCUIT BY OBJECT-RELATIONAL DATABASE"

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้อาจไม่เสร็จสมบูรณ์ด้วยดีหากไม่ได้รับความช่วยเหลือ ความร่วมมือและคำแนะนำจากบุคคลหลายๆท่านด้วยกัน บุคคลแรกเป็นคนที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ อาจารย์ ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาวิทยานิพนธ์ อาจารย์เอาใจใส่ดูแลและช่วยเหลืออย่างดีตลอดระยะเวลาหนึ่งปีขอขอบพระคุณอาจารย์เป็นอย่างมาก

อีกท่านที่ต้องขอบคุณก็คือพี่บัณฑิตสำหรับความช่วยเหลือต่างๆทั้งหาหนังสือมาให้ทั้งคิดต่อประสานงานกับอาจารย์ และสุดท้ายขอขอบคุณพี่พิทักษ์ที่ช่วยอำนวยความสะดวกในการใช้งานระบบฐานข้อมูลบนแคแสดและหนังสือคู่มือของระบบที่ให้มา

พิมพ์ผกา อังสวานนท์

ภุชงค์ ตั้งเจตน์

สารบัญ

หน้าที่

สารบัญรูปภาพ

หน้าที่

สารบัญตาราง

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในการใช้งานระบบฐานข้อมูลในปัจจุบันโมเดลข้อมูล (Data Model) ที่นิยมใช้งานกันแพร่หลายที่สุดได้แก่ ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) แต่ฐานข้อมูลเชิงสัมพันธ์ยังมีข้อจำกัดหลายๆด้าน เช่น ชนิดของข้อมูลที่ใช้ยังคงเป็นชนิดข้อมูลแบบง่ายๆ ไม่สามารถรองรับการใช้งานการเข้าถึงและการจัดเก็บข้อมูลที่มีความซับซ้อนได้ เช่น ข้อมูลทางมัลติมีเดีย , งานทางด้าน GIS (Geographic Information System) และ การจัดเก็บวงจรไฟฟ้า เป็นต้น

จากการที่ระบบฐานข้อมูลได้เข้ามามีบทบาทในวงการต่างๆเช่นวงการธุรกิจมากขึ้น ทำให้ต้องพบกับปัญหาการใช้งานและการจัดเก็บข้อมูลชนิดต่างๆที่มีความซับซ้อนเกินกว่าชนิดข้อมูลเดิมที่มีอยู่จึงทำให้ต้องมีการศึกษาโมเดลข้อมูลแบบใหม่ๆเพื่อสนองความต้องการในการจัดเก็บข้อมูลดังกล่าวซึ่งในโครงงานนี้มุ่งเน้นไปที่การศึกษาและใช้งานโมเดลข้อมูลที่เป็นโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database) ซึ่งสามารถแก้ปัญหาบางประการของชนิดข้อมูลในระบบฐานข้อมูลเชิงสัมพันธ์ได้

ฐานข้อมูลเชิงวัตถุสัมพันธ์ คือฐานข้อมูลที่นำแนวคิดเชิงวัตถุ (Object-Oriented) มารวมเข้ากับฐานข้อมูลเชิงสัมพันธ์เพื่อทำให้ฐานข้อมูลเชิงสัมพันธ์มีความสามารถทางด้านแนวคิดเชิงวัตถุเช่นการปกป้องข้อมูล (Encapsulation) การสืบทอดคุณสมบัติ (Inheritance) และโพลิมอร์ฟิซึ่มเป็นต้น

ฐานข้อมูลเชิงวัตถุสัมพันธ์เป็นฐานข้อมูลเชิงสัมพันธ์ที่สามารถมีแอททริบิว (Attribute) เป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีก (Non-atomic Attribute) โดยภาพรวมแล้วเรายังมองเห็นฐานข้อมูลเป็นฐานข้อมูลเชิงสัมพันธ์โดยในคอลัมน์สามารถเป็นได้ทั้งออบเจกต์หรือเป็นแอททริบิวที่มีหลายค่า (Multiple Value) เช่น ลิสท์ (List) หรือ เซต (Set) โดยทั่วไปโมเดลที่เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถใช้งานเสมือนเป็นฐานข้อมูลเชิงสัมพันธ์ได้

จากความสามารถของฐานข้อมูลเชิงวัตถุสัมพันธ์ตามที่ได้กล่าวมาจะทำให้เราสามารถพัฒนาฐานข้อมูลที่มีความสามารถในการจัดการกับข้อมูลที่มีความซับซ้อนได้อย่างมีประสิทธิภาพซึ่งจะรองรับการใช้งานในปัจจุบันได้อย่างเต็มที่

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาภาษาฐานข้อมูล SQL3 ว่ามีความสามารถอะไรเพิ่มขึ้นมาในการจัดการกับออบเจกต์บ้าง ความสามารถของภาษาในด้านการใช้งานกับโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์

1.2.2 ศึกษาหลักการของโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ เพื่อนำมาเป็นแบบจำลองข้อมูลระดับแนวคิดและศึกษาลักษณะการแทนข้อมูลลงในฐานข้อมูล

1.2.3 ศึกษาการออกแบบและการใช้งานฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อประยุกต์ใช้ในงานต่างๆ ได้

1.2.4 ทดลองการใช้งานระบบจัดการฐานข้อมูล (Database Management System : DBMS) ที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งในโครงงานนี้ได้ใช้อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ (Informix Universal Server) ซึ่งเป็นระบบจัดการฐานข้อมูลที่สามารถจัดเก็บข้อมูลได้ทั้งที่เป็นโมเดลฐานข้อมูลเชิงสัมพันธ์และโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์

1.2.5 ศึกษาและทำการทดลองใช้งานชนิดข้อมูลต่างๆที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ที่ระบบจัดการฐานข้อมูลมีให้และศึกษาการใช้งานของระบบจัดการฐานข้อมูลเทียบกับมาตรฐาน SQL3

1.2.6 ทดลองสร้างโปรแกรมประยุกต์ (Application) ที่ทำการเก็บข้อมูลที่มีความซับซ้อนซึ่งได้ทำการเลือกการจัดเก็บวงจรดิจิตอลซึ่งประกอบไปด้วยเกทชนิดต่างๆโดยใช้คุณลักษณะของฐานข้อมูลเชิงวัตถุสัมพันธ์ในการจัดเก็บข้อมูล

1.2.7 เปรียบเทียบการจัดเก็บวงจรดิจิตอลในการใช้ฐานข้อมูลเชิงสัมพันธ์กับฐานข้อมูลเชิงวัตถุสัมพันธ์

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะทำการสร้างฐานข้อมูลที่ใช้โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์โดยจะทำการพัฒนาบนระบบจัดการฐานข้อมูลของอินฟอร์มิกซ์ที่ชื่อยูนิเวอร์แซลเซิร์ฟเวอร์ทำการศึกษาความสามารถของระบบฐานข้อมูลที่เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์เทียบกับมาตรฐานภาษาฐานข้อมูล SQL3 ซึ่งเป็นภาษาฐานข้อมูลมาตรฐานใหม่จาก SQL92 โดยภาษา SQL3 นี้ได้เพิ่มความสามารถทางออบเจกต์เข้าไปเพื่อรองรับกับการใช้งานในทางฐานข้อมูลเชิงวัตถุสัมพันธ์ ในงานวิจัยจะเน้นความสามารถของภาษา SQL ที่เพิ่มขึ้นมาในการจัดการกับข้อมูลในแบบออบเจกต์แล้วทำการศึกษาระบบจัดการฐานข้อมูลที่เลือกมาเป็นกรณีศึกษาว่ามีการใช้งานโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์อย่างไร มีความสามารถใดเพิ่มขึ้นมาบ้าง ศึกษาความสามารถทางแนวคิดเชิงวัตถุที่เพิ่มเข้ามา หลังจากการศึกษาระบบจัดการฐานข้อมูลแล้วจะทำการพัฒนาโปรแกรมประยุกต์ที่ทำการติดต่อกับฐานข้อมูลที่เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งในงานวิจัยนี้เลือกการจัดเก็บวงจรทางดิจิตอลเป็นกรณีศึกษา โดยจะทำการพัฒนาโดยใช้วิชวลเบสิก (Visual Basic) ทำการติดต่อกับฐานข้อมูลผ่านดาต้าไดเร็คเตอร์ (Data Director) ซึ่งเป็นตัวจัดการการติดต่อกับฐานข้อมูลของระบบจัดการฐานข้อมูลอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์โดยความสามารถของโปรแกรมประยุกต์จะเน้นการพัฒนาการใช้งานชนิดข้อมูลที่มีความซับซ้อน ความสามารถในการจัดเก็บมากกว่าพัฒนาสร้างโปรแกรมประยุกต์ที่สามารถสร้างวงจรที่มีความซับซ้อน โปรแกรมประยุกต์ที่สร้างขึ้นจะสามารถสร้างวงจรทางดิจิตอลอย่างง่ายซึ่งประกอบไปด้วยเกทชนิดต่างๆ สร้างคอมโพเนนท์ใหม่จากเกทพื้นฐานที่มีได้ นำคอมโพเนนท์ใหม่ที่สร้างขึ้นไปใช้งานในวงจรเหมือนกับเกทพื้นฐานอื่นๆได้และสามารถทำการจำลองการทำงานของวงจรได้

1.4 วิธีการดำเนินงาน

งานวิจัยแบ่งออกเป็น 3 ส่วนด้วยกันคือ ส่วนที่ 1 เป็นการศึกษาทฤษฎีพื้นฐานต่างๆที่จำเป็นได้แก่ แนวคิดเชิงวัตถุ โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ และ มาตรฐาน SQL3 ซึ่งจะมีรายละเอียดอยู่ในบทที่ 2 และ 3

ส่วนที่ 2 จะเป็นการศึกษาระบบจัดการฐานข้อมูลที่ใช้ การใช้งานระบบ ความสามารถของระบบในการรองรับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ชนิดข้อมูลการจัดการกับข้อมูลซึ่งจะมีรายละเอียดอยู่ในบทที่ 4 รวมถึงมีวิธีการจัดติดตั้งระบบและตัวอย่างการใช้งานระบบจัดการฐานข้อมูลอยู่ในภาคผนวก ก. และ ในบทที่ 5 จะเป็นตัวอย่างการสร้างการใช้งานชนิดข้อมูล 2 ชนิดได้แก่ Large Object และ Opaque Type โดยจะมีตัวอย่างการสร้างโอเปคโดยใช้ดาต้าเบลดโมดูลเดเวลลอปเปอร์ (DataBlade Module Developer) อยู่ในภาคผนวก ข. ในบทที่ 6 จะกล่าวถึงภาษา Informix Stored Procedure Language ที่ถูกเขียนขึ้นมาเพื่อใช้งานบนดาต้าเบสเซิร์ฟเวอร์

ส่วนที่ 3 เป็นส่วนการทดลองสร้างโปรแกรมประยุกต์เพื่อใช้งานกับฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งจะมีรายละเอียดอยู่ในบทที่ 7 ในบทที่ 8 จะเป็นตัวโปรแกรมประยุกต์ที่ได้ทำการพัฒนาขึ้นมาและการเปรียบเทียบการจัดเก็บวงจรดิจิตอลในการใช้ฐานข้อมูลเชิงสัมพันธ์กับฐานข้อมูลเชิงวัตถุสัมพันธ์ และในบทที่ 9 ซึ่งเป็นบทสุดท้ายก็จะเป็นการสรุปการทำงาน ผลที่ได้รับจากงานวิจัยชิ้นนี้ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
แนวคิดเชิงวัตถุและฐานข้อมูลเชิงวัตถุสัมพันธ์

ในบทนี้จะกล่าวถึงแนวคิดเชิงวัตถุ (Object-Oriented),ฐานข้อมูลเชิงวัตถุสัมพันธ์และการเปรียบเทียบระหว่างฐานข้อมูลเชิงวัตถุสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์

2.1 แนวคิดเชิงวัตถุ (Object-Oriented)

แนวคิดเชิงวัตถุเป็นการรวมโค้ด (Code) และตัวแปรข้อมูล (Data) ไว้ด้วยกันหรือเป็นการปกป้องข้อมูลไว้ภายใต้โค้ด (Encapsulate) ซึ่งเรียกว่าออบเจกต์ (Object) แต่ละออบเจกต์จะเชื่อมต่อกันผ่านทางแมสเสจ (Message) ซึ่งแมสเสจคือการส่งการร้องขอ (Request) ระหว่างออบเจกต์โดยไม่คำนึงถึงรายละเอียดในการอิมพลีเมนต์ (Implementation) หรือโครงสร้างภายใน (Internal Structure) ของออบเจกต์นั้น ออบเจกต์จะตอบสนองการติดต่อจากภายนอกที่เป็นแมสเสจเท่านั้น ทำให้การแก้ไขหรือปรับปรุงออบเจกต์หนึ่งไม่มีผลต่อออบเจกต์อื่นๆ อีกทั้งเรายังสามารถปกป้องข้อมูลภายในออบเจกต์จากการใช้ที่ผิดๆอีกด้วย ซึ่งเป็นข้อดีที่สำคัญอย่างหนึ่งของแนวคิดเชิงวัตถุ

ออบเจกต์สามารถถ่ายทอดคุณสมบัติ (Property)และการดำเนินการ (Operation)ของ ออบเจกต์ได้โดยการทำการถ่ายทอดคุณสมบัติ ซึ่งจะทำให้มีการใช้คุณสมบัติและฟังก์ชั่นร่วมกันจัดความสัมพันธ์ให้อยู่ในรูป ซูเปอร์คลาส (Superclass) และ สับคลาส (Subclass) โดยสับคลาสสามารถรับการถ่ายทอดคุณสมบัติจากซูเปอร์คลาสๆเดียวหรือหลายคลาสก็ได้

อีกทั้งออบเจกต์ยังมีความสามารถในการทำโพลิมอร์ฟิซึ่ม คือความสามารถในการอ้างถึงอินสแตนท์หลายชนิดหรือคลาส ณ ขณะกำลังทำงาน โดยสามารถสร้างรูทีน (Routine) ที่มีชื่อเดียวกันแต่ต่างอาร์กิวเมนท์ (Argument) เพื่อให้ง่ายต่อการใช้งาน (One Interface Multiple Method) เรียกว่ารูทีนโอเวอร์โหลดดิง (Routine Overloadind) โดยดาต้าเบสเซิร์ฟเวอร์ (Database Server) จะทำการตัดสินใจว่าจะเรียกใช้รูทีนใดดูจากรูทีน ซิกเนเจอร์ (Routine Signature)

สรุปคุณสมบัติที่ออบเจกต์พึงมี

- เป็นการมองภาพของสิ่งที่เราสนใจ มีลักษณะเฉพาะของแต่ละออบเจกต์ซึ่งเรารู้ว่าออบเจกต์นี้ทำอะไรและจะต้องเรียกผ่านแมสเสจใด
- มีขอบเขตที่ชัดเจน กำหนดการปกป้องข้อมูลว่าใครจะใช้ได้ใครใช้ไม่ได้
- มีวิธีการติดต่อระหว่างกันชัดเจน
- ปกป้องข้อมูลที่มีอยู่ภายในตามประเภทของข้อมูลที่มีอยู่สามแบบคือ Public , Private และ Protected
- สามารถถ่ายทอดคุณสมบัติของออบเจกต์หนึ่งไปยังอีกออบเจกต์ได้

- มีความสามารถต่างๆ เช่น มีความสามารถในการทำ Overloading , Late Binding เป็นต้น

- Abstract Data Type (ADT)

แนวความคิดพื้นฐานของแอบสแทรคดาต้าไทป์ คือการแยกภาพการมองของผู้ใช้ออกจากรายละเอียดในการสร้าง ADT (Implement ADT) มีบทบาทต่อโปรแกรมประยุกต์ทางฐานข้อมูลโดยใช้ขยายความสามารถจัดการกับชนิดข้อมูลให้มีความซับซ้อนมากยิ่งขึ้น

2.2 Object-Relational Database

เนื่องจากการจัดเก็บข้อมูลในปัจจุบันโดยทั่วไปมักใช้ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ แต่เมื่อมีปัญหากับการใช้งานชนิดข้อมูลที่มีแต่ข้อมูลชนิดพื้นฐานไม่สามารถรองรับการใช้งานในปัจจุบัน ซึ่งนอกจากฐานข้อมูลเชิงสัมพันธ์แล้วยังมีระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) เป็นระบบฐานข้อมูลที่มีลักษณะเป็นแบบแนวคิดเชิงวัตถุโดยใช้ออบเจกต์เป็นพื้นฐานในการสนับสนุนการใช้งานข้อมูลที่มีความซับซ้อนแต่ก็ยังมีข้อเสียบางประการอีกทั้งยังไม่สามารถสนับสนุนงานในลักษณะของฐานข้อมูลได้เต็มที่ ภาษาที่ใช้ในการจัดการฐานข้อมูลยังไม่เป็นมาตรฐานจึงเกิดแนวคิดในการนำข้อดีของฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูลเชิงวัตถุมารวมกัน

ฐานข้อมูลเชิงวัตถุสัมพันธ์ คือ ฐานข้อมูลที่นำแนวคิดเชิงวัตถุมารวมเข้ากับฐานข้อมูลเชิงสัมพันธ์เพื่อทำให้ฐานข้อมูลเชิงสัมพันธ์มีความสามารถทางด้านแนวคิดเชิงวัตถุ เช่นการปกป้องข้อมูล การสืบทอดคุณสมบัติ และ โพลิมอร์ฟิซึ่มเป็นต้น

ฐานข้อมูลเชิงวัตถุสัมพันธ์เป็นฐานข้อมูลเชิงสัมพันธ์ที่สามารถมีแอททริบิวเป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีก โดยภาพรวมแล้วเรายังมองเห็นฐานข้อมูลเป็นฐานข้อมูลเชิงสัมพันธ์ ในคอลัมน์สามารถเป็นได้ทั้งออบเจกต์หรือเป็นแอททริบิวที่มีหลายค่าเช่นลิสท์หรือเซต โดยทั่วไปโมเดลที่เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถใช้งานเสมือนเป็นฐานข้อมูลเชิงสัมพันธ์ ได้

คุณสมบัติของฐานข้อมูลเชิงวัตถุสัมพันธ์

1. มีความสามารถเดิมของฐานข้อมูลเชิงสัมพันธ์
2. สนับสนุนการสร้างออบเจกต์ที่มีความซับซ้อน
3. มีความยืดหยุ่นสูงยอมให้สร้างชนิดข้อมูล ฟังก์ชั่นและตัวดำเนินการ(Operator)ใหม่ได้
4. สามารถสร้างแอททริบิวเป็นออบเจกต์หรือเป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีกได้
5. มีคุณสมบัติแนวคิดเชิงวัตถุ เช่นการปกป้องข้อมูล การสืบทอดคุณสมบัติ และ โพลิมอร์ฟิซึ่มเป็นต้น

2.3 การเปรียบเทียบระหว่างฐานข้อมูลเชิงวัตถุสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์

2.3.1 ชนิดข้อมูล

ในระบบฐานข้อมูลเชิงสัมพันธ์นั้นแอททริบิวของรีเลชั่นไม่สามารถแยกออกเป็นส่วนๆได้อีกคือต้องเป็นแอททริบิวที่ไม่สามารถแบ่งแยกย่อยได้อีกและมีชนิดของข้อมูลเป็นชนิดที่ค่อนข้างง่าย ได้แก่

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

การดำเนินการต่างๆบนรีเลชั่นจำกัดอยู่ที่การเรียกดูข้อมูลและการแก้ไข และมีความสามารถจำกัดในการจัดการกับข้อมูลที่เป็นไบนารี่ (Binary) เช่น BLOB(Binary Large Object) จากขีดความสามารถที่จำกัดดังกล่าวส่งผลให้ต้องมีการพัฒนาโมเดลอื่นที่สามารถจัดการข้อมูลที่ซับซ้อนได้

ฐานข้อมูลเชิงวัตถุสัมพันธ์อนุญาตให้มีชนิดของข้อมูลเป็นแอททริบิวที่สามารถแบ่งแยกย่อยได้อีกหรือเป็นออบเจกต์ได้ดังนั้นชนิดของข้อมูลในฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถเป็นข้อมูลที่มีความซับซ้อนกว่าในรีเลชั่นเช่นข้อมูลที่เป็นอิมเมจ (Image) และข้อมูลชนิดที่ใช้ในงานด้านมัลติมีเดียเป็นต้น ชนิดของข้อมูลที่เพิ่มเข้ามาในฐานข้อมูลเชิงสัมพันธ์ เช่น

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

2.3.2 การควบคุมความถูกต้องของข้อมูล (Data Integrity)

ความสามารถในการทำการควบคุมความถูกต้อง (Integrity Constraint) ของฐานข้อมูลเชิงสัมพันธ์ยึดตามมาตรฐานของ SQL92 ในขณะที่ฐานข้อมูลเชิงวัตถุสัมพันธ์มีความสามารถตามที่กำหนดไว้ในมาตรฐาน SQL3 ดังนี้

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

2.3.3 ภาษาที่ใช้

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

ชนิดของ Statement ที่เพิ่มเติมใน SQL3

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

บทที่ 3
ภาษาเรียกค้น SQL3

ในบทที่จะกล่าวถึงภาษาที่ใช้งานกับฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งคือภาษา SQL3 ในปัจจุบันยังไม่เป็นมาตรฐานที่สมบูรณ์ในที่นี้จะอ้างถึงเฉพาะที่เป็นดราฟท์ (Draft) มาตรฐานเท่านั้น ซึ่ง SQL3 ได้เพิ่มความสามารถต่างๆจากมาตรฐานเดิมหรือ SQL92 ดังมีรายละเอียดดังนี้

3.1 ชนิดข้อมูล

3.1.1 Predefined Data Type หรือ Built-in Data Type

ประกอบไปด้วย Character , Character Varying , Charater Large Object , Binary Large Object , Bit , Bit Varying , Numeric , Decimal , Integer , Smallint , Enumerated , Float , Real , Double , Precision , Boolean , Date , Time , Timestamp และ Interval Boolean ใน SQL3 ได้เพิ่มชนิดข้อมูลอีกสองชนิดเป็นออบเจกต์คาต้าไทป์ คือ Row Type และ Abstract Data Type

3.1.2 Row Type

สามารถกำหนดชนิดของทัปเปิล (Tuple) ซึ่งจะประกอบไปด้วยข้อมูลชนิดอื่นๆหรือ Row Type อื่น การใช้งาน Row Type สามารถใช้เหมือนเป็นหน่วยเดียวหรือแตกเป็นหน่วยย่อยก็ได้ โดยการประกาศ Row Type มีรูปแบบดังนี้

```
CREATE ROW TYPE <Name_of_RowType>
(<Component Declaration>));
```

สามารถนำ Row Type ไปอ้างเหมือนเป็นข้อมูลชนิดอื่นๆได้ทั้งใน Row Type อื่นและในเทเบิล ดังตัวอย่างการประกาศ Row Type

```
CREATE ROW TYPE Name_of_RowType
(<Component Declaration>,
ColumnName      RowTypeName));
```

```
CREATE TABLE TableName
(<Component      Declaration>,
```

```
ColumnName        RowTypeName));
```

ตัวอย่างการสร้างเทเบิล โดยใช้ Row Type

```
CREATE TABLE <TableName>  OF TYPE <RowTypeName>
```

ตัวอย่างการใช้งาน

```
CREATE ROW TYPE ADDRESSType
(STREET          CHAR(50)
 CITY            CHAR(20));

CREATE ROW TYPE StarType
(NAME            CHAR(30)
 ADDRESS         ADDRESSType);

CREATE TABLE  MovieStar OF TYPE StarType;
```

จากตัวอย่างการสร้าง Row Type ADDRESSType เป็น Row Type ที่ประกอบไปด้วย 2 แอททริบิวคือ Street และ City สำหรับ Row Type StarType เป็น Row Type ที่ประกอบไปด้วย 2 แอททริบิวคือ Name และ Row Type ADDRESSType จากการสร้างเทเบิล MovieStar จะได้เทเบิลที่ประกอบไปด้วย 2 คอลัมน์มีลักษณะแอททริบิว ดังนี้

Name	Address
Sharon Stone	Maple St.,Beverly Hills
Mel Gibson	Oak St.,Brentwood

ตารางที่ 3.1 ตัวอย่างการจัดเก็บข้อมูลที่เป็น Row Type

สำหรับการเข้าถึงข้อมูลที่เป็น Row Type สามารถใช้งานเหมือนเป็นหน่วยเดียวหรือเข้าถึงส่วนย่อยๆ โดยใช้ Dot Notation 2 ครั้ง ".." ดังนี้

```
SELECT  MovieStar.NAME,MovieStar.ADDRESS..STREET
```

```
FROM      MovieStar
WHERE   MovieStar.ADDRESS..CITY = 'Beverly Hills';
```

3.1.3 Reference

SQL3 มีความสามารถในการอ้างถึงออบเจกต์อื่นๆ(Reference) โดยมีรูปแบบการอ้างถึงดังนี้

```
CREATE ROW TYPE MovieType
(TITLE          CHAR(30)
 YEAR           INTEGER);

CREATE ROW TYPE StarType
(NAME           CHAR(30)
 ADDRESS        ADDRESSType
 BESTMOVIES     REF(MovieType));
```

โดยในคอลัมน์ที่เรากำหนดให้อ้างถึงนี้จะไม่ได้ทำการเก็บข้อมูลจริงๆแต่จะทำการเก็บ ObjectID (OID) ซึ่งใช้อ้างถึงไปยังออบเจกต์อื่น

3.1.4 Object Identifier (OID)

เป็นตัวระบุความแตกต่าง (Identifier) ของออบเจกต์แต่ละตัวจะไม่มีการเปลี่ยนแปลงค่า โดยเราสามารถทำการสร้างโอไอดี (OID) ขึ้นมาเป็นคอลัมน์หนึ่งซึ่งสามารถเรียกดูผ่าน SQL Command ได้เหมือนกับแอททริบิวชนิดอื่นๆ โดยมีรูปแบบการกำหนดดังนี้

```
VALUES FOR <ATTRIBUTE> ARE SYSTEM GENERATED
```

ตัวอย่างการใช้งาน

```
CREATE TABLE MOVIE OF TYPE MOVIETYPE
VALUES FOR MOVIE_ID ARE SYSTEM GENERATE
```

3.1.5 Abstract Data Type (ADT)

จากความสามารถในการสร้าง Row Type และการอ้างถึง (Reference) ไปยัง Row Type อื่นซึ่งเป็นความสามารถของแนวคิดเชิงวัตถุแต่ Row Type ไม่สามารถปกป้องข้อมูลได้อย่างสมบูรณ์ SQL3 จึงเพิ่มข้อมูลชนิดใหม่ที่สามารถทำการปกป้องข้อมูลได้อย่างสมบูรณ์คือแอบสแทรคดาต้าไทป์ (Abstract Data Type) หรือ ADT ซึ่งออบเจกต์ของ ADT จะใช้เป็นส่วนหนึ่งของทัปเปิล (Tuple)เหมือนกับชนิดข้อมูลพื้นฐานทั่วไป การประกาศ ADT ทำได้ดังนี้

```
CREATE TYPE <Type NAME>
( <List of Attributes and their Types>,
 Optional declaration of = and < Functions for the Type ,
 Declaration of Functions (Method) for the Type);
```

ตัวอย่าง

```
1)   CREATE TYPE ADDRESS_ADT
2)   (STREET      CHAR(50),
3)   CITY         CHAR(20),
4)   EQUALS       ADDREQ,
5)   LESS THAN    ADDRLT
6)   other functions should be declare here
```

โดยที่บรรทัดที่ 1 เป็นการประกาศ ADT และกำหนดชื่อของ ADT บรรทัดที่ 2 และ 3 เป็นการกำหนดแอททริบิวของ ADT และบรรทัดที่ 4-5 เป็นออพชั่นนอล (Optional) ในการกำหนดโอเปอร์เรเตอร์ที่ใช้ในการเปรียบเทียบ (Comparison Operator) เท่ากับและน้อยกว่าสำหรับการเปรียบเทียบใน Where Clause บรรทัดที่ 6 เป็นการประกาศฟังก์ชั่นหรือเมธทอด (Method) ซึ่ง SQL3 มี Built-In Function ให้กับแต่ละ ADT ดังนี้

1.*Constructor Function* เป็นการสร้างอินสแตนท์ของ ADT จะทำการส่งค่า (Return) ออบเจกต์ใหม่ของ ADT ทุกๆแอททริบิวของออบเจกต์จะกำหนดค่าตั้งต้นเป็น NULL โดยชื่อของคอนสตรัคเตอร์ฟังก์ชั่นจะมีชื่อเดียวกับ ADT ที่เราสร้างขึ้น

2.*Observer Function* เป็นฟังก์ชั่นที่ทำการส่งคืนค่าของทุกๆแอททริบิวโดยที่ออบเซิร์ฟเวอร์ฟังก์ชั่นจะถูกกำหนดให้มีชื่อเดียวกับแอททริบิวและมีพารามิเตอร์ 1 ตัวเป็นชนิดเดียวกับ ADT ที่กำหนด

3.*Mutator Function* เป็นการกำหนดค่าของแอททริบิวนั้นให้เป็นค่ามิวเตเตอร์ฟังก์ชั่นจะถูกกำหนดให้มีชื่อเดียวกับแอททริบิวและมีพารามิเตอร์ 2 ตัว ตัวแรกเป็นชนิดเดียวกับ ADT ที่กำหนด ตัวที่สองเป็นชนิดของข้อมูลของแอททริบิวนั้น

การปกป้องข้อมูลทำได้ถึง 3 ระดับ คือ Public , Private และ Protect ใหม่ การปกป้องข้อมูลให้สมบูรณ์ต้องทำการป้องกันการเรียกใช้ฟังก์ชั่นแบบ Public

3.1.5.1 Function Declaration

สำหรับฟังก์ชั่นอื่นๆสามารถกำหนดได้ทั้งในและนอกคำสั่ง CREATE TYPE แต่ถ้าอยู่ภายนอกจะสามารถใช้ได้แต่ฟังก์ชั่นที่กำหนดไว้ภายในและบิวท์อิน (Built-In) ที่มีให้เท่านั้นจะไม่สามารถเรียกใช้ฟังก์ชั่นที่อยู่ภายนอกอื่นๆได้ หลังจากส่วนของการประกาศแอททริบิวใน ADT แล้วจะเป็นส่วนของการประกาศฟังก์ชั่นซึ่งมีรูปแบบ ดังนี้

```
FUNCTION <NAME> (<Argumenr>) RETURNS <Type> ;
```

แต่ละอาร์กิวเมนท์ประกอบด้วยชื่อของตัวแปรและชนิดของตัวแปรแต่ละอาร์กิวเมนท์จะแยกจากกันด้วยการใช้ " , " คั่น การเขียนฟังก์ชั่นมี 2 แบบคือแบบภายใน (Internal) และแบบภายนอก (External) โดยถ้าเป็นฟังก์ชั่นภายนอก ADT จะมีเมธทอดที่เขียนด้วย Host Language โดยจะต้องมีการประกาศซิกเนเจอร์ (Signature) ไว้ใน ADT โดยมีรูปแบบการประกาศฟังก์ชั่นภายนอกดังนี้

```
DECLARE EXTERNAL <Function NAME> <signature>
LANGUAGE      <language NAME>
```

ตัวอย่าง

```
DECLARE EXTERNAL findzip CHAR(50) CHAR(20)
RETURNS CHAR(10)
LANGUAGE C;
```

ส่วนฟังก์ชั่นภายในมีรูปแบบการประกาศ ดังนี้

```
FUNCTION AddressADT (:s char(50),:c char(20)
RETURNS    AddressADT;
```

```
:a   AddressADT;
BEGIN
    :a  :=  AddressADT( );
    :a.street := :s;
    :a.city := a;
    RETURN :A;
END;
```

จากตัวอย่างเป็นการสร้างคอนสตรัคเตอร์ฟังก์ชั่นให้กับเอดีที

3.2 Integrity Constraint

ความสามารถในการทำควบคุมรักษาความถูกต้อง (Integrity Constraint) ของ SQL3 มีดังต่อไปนี้

3.2.1 Key Constraint

เป็นการประกาศคีย์หลักจะประกาศในขั้นตอนการ CREATE TABLE ทำได้สองแบบคือการใช้คีย์เวิร์ด (Keyword) PRIMARY KEY และ UNIQUE

<ColumnName> <DataType> <PRIMARY KEY|UNIQUE> หรือ <PRIMARY KEY|UNIQUE> (<ColumnName>)

ตัวอย่างทั้ง 2 วิธีในการประกาศคีย์หลัก

```
CREATE TABLE MovieStar
(NAME          CHAR(30) PRIMARY KEY,
 ADDRESS       VARCHAR(255),
 GENDER        CHAR(1));

CREATE TABLE MovieStar
(NAME          CHAR(30),
 ADDRESS       VARCHAR(255),
 GENDER        CHAR(1),
```

```
PRIMARY KEY        (NAME));
```

สำหรับการประกาศคีย์หลักที่มีมากกว่า 1 แอททริบิวทำได้ดังนี้

PRIMARY KEY (<ColumnName>,<ColumnName>)

ส่วนการประกาศโดยใช้คีย์เวิร์ด UNIQUE เป็นการกำหนดให้คอลัมน์ดังกล่าวต้องไม่มีค่าซ้ำ รูปแบบการประกาศเหมือนกับการประกาศคีย์หลักทุกประการ

3.2.2 Referential Integrity Constraint

เป็นการกำหนดความสัมพันธ์ระหว่างคีย์หลักและฟอร์เรียนคีย์ (ForeignKey) การประกาศ ทำได้ 2 แบบดังนี้

REFERENCE <Table>(<Attribute>) หรือ FOREIGN KEY <Attributes> REFERENCES <Table>(<Attributes>)

ตัวอย่างการประกาศ Foreign Key

```
CREATE TABLE Studio
(       NAME    CHAR(30)    PRIMARY KEY,
        ADDRESS  VARCHAR(255),
        PRESC#    INT  REFERENCES MovieExec(CERT#)

CREATE TABLE Studio
(       NAME     CHAR(30)    PRIMARY KEY,
        ADDRESS   VARCHAR(255),
        PRESC#     INT
        FOREIGN KEY PRESC# REFERENCES MovieExec(CERT#))
```

3.2.3 Attribute Constraint

การกำหนดการบังคับความถูกต้อง (Constraint) จะกำหนดได้ 2 แบบ คือ

1.ในการกำหนดรีเลชั่น สกีมา (Definition ของ Relation Schema)

2.ในโดเมน (Domain) ซึ่งจะประกาศ เป็น Domain Constraint

โดยมีรายละเอียดของแต่ละประเภทดังนี้

3.2.3.1 NOT NULL Constraint

เป็นการกำหนดไม่ให้แอททริบิวนั้นมีค่า NULL โดยใช้คีย์เวิร์ด "NOT NULL" ในตอนที่ประกาศแอททริบิวดังนี้

```
PRESC#    INT    REFERENCES    MovieExec(CERT#)   NOT NULL
```

3.2.3.2 Attribute-based CHECK Constraint

จะทำการเช็คความถูกต้องของแอททริบิวทุกครั้งที่มีการแก้ไขข้อมูล เช่น การอินเสิร์ท (Insert) หรืออัพเดท (Update) จะต้องมีการเช็คความถูกต้องของแอททริบิวที่กำหนดไว้ โดยมีรูปแบบ ดังนี้

```
PRESC#   INT     REFERENCES    MovieExec(CERT#)  CHECK  (PRESC# >= 100000)
```

หรือ

```
GENDER      CHAR(1)     CHECK   (GENDER  IN   ('F','M')),
```

3.2.3. Domain Constraint

ทำการสร้างโดเมนใหม่ขึ้นมา แล้วทำการกำหนดค่าในโดเมน ดังนี้

```
CREATE DOMAIN GenderDomain  CHAR(1)
CHECK  (VALUE  IN  ('F','M'));
```

เมื่อมีการสร้างรีเลชั่นสามารถนำโดเมนที่สร้างขึ้นไปอ้างถึงเหมือนกับเป็นข้อมูลชนิดอื่นๆได้ การสร้างโดเมนคอนสเตรนท์ดีกว่าสองแบบแรกเพราะสามารถใช้งานได้ในทุกๆรีเลชั่น

3.2.4 Global Constraint

เป็นการประกาศคอนสเทรนท์ที่มีความซับซ้อนมากขึ้นมีการประกาศ ได้สองลักษณะ

3.2.4.1. Tupled-Based CHECK Constraint

จะทำการตรวจสอบทุกครั้งที่มีการอินเสิร์ทหรืออัพเดททัปเปิลโดยจะควบคุมความถูกต้องแค่บนรีเลชั่นที่สร้างขึ้นมา ดังตัวอย่างการประกาศ Tupled-Based CHECK บนคอลัมน์ GENDER

```
CREATE TABLE MovieStar
(       NAME        CHAR(30)        PRIMARY KEY,
        ADDRESS     VARCHAR(255),
        GENDER      CHAR(1),
        CHECK       (GENDER ='F'  OR  NAME  NOT LIKE  'Ms.%')
```

3.2.4.2. Assertion

เป็นการกำหนดเงื่อนไขความถูกต้อง โดยเงื่อนไข (Condition) ในแอสเสิร์ทชั่น (Assertion) จะต้องเป็นจริงเสมอถ้าการแก้ไข (Modified) ใดทำให้เงื่อนไขเป็นเท็จจะถูกปฏิเสธทันที (Reject) ทันที แอสเสิร์ทชั่นแต่เดิมในมาตรฐาน SQL92 จะควบคุมความถูกต้องทุกๆรีเลชั่นแต่ใน SQL3 เหลือควบคุมแค่บนรีเลชั่นเดียว

```
CREATE ASSERTION  RichPRES  CHECK
(NOT EXISTS
        (SELECT   *
        FROM      Studio,MovieExec
        WHERE     PRESC# = CERT# AND NETWORTH < 1000000));
```

3.2.5 Triggers

เป็น แอคทีฟคอมโพเนนท์ที่จะเข้ามาทำแอคชั่น (Action) เมื่อเกิดเหตุการณ์ (Event) ตามที่กำหนดไว้ โดย เหตุการณ์ที่เกิดได้ทั้ง Before , After , Instead of โดยถ้าเป็น Before จะทำการเช็คเหตุการณ์ก่อนที่จะทำแอคชั่น ถ้า After จะทำแอคชั่นหลังจากเกิดเหตุการณ์ แต่ถ้าเป็น Instead of ถ้าเกิดเหตุการณ์จะไม่เกิดแอคชั่น

ตัวอย่างการสร้างทริกเกอร์

```
CREATE TRIGGER NETWORTHTrigger
AFTER UPDATE OF NETWORTH ON MovieExec
REFERENCING
        OLD  AS  OldTuple,
        NEW AS  NewTuple
WHEN  (OldTuple.NETWORTH > NewTuple.NETWORTH)
        UPDATE  MovieExec
        SET  NETWORTH = OldTuple.NETWORTH
        WHERE  CERT# = NewTuple.CERT#
FOR EACH ROW
```

3.3 Inheritance

เป็นแอบสแทรคแมคคานิซึ่ม (Abstract Mechanism) ที่อนุญาตให้ชนิดข้อมูลมีความสัมพันธ์กันแบบลำดับชั้น (Hierarchy) สามารถกำหนดชนิดข้อมูลเป็นสับไทป์ (SubType) และสามารถถ่ายทอดคุณสมบัติ พฤติกรรม และคอนสเตรนท์ของซูเปอร์ไทป์ (SuperType) ซึ่งในการถ่ายทอดคุณสมบัติสามารถเป็นได้ทั้งการถ่ายทอดคุณสมบัติจากไทป์เดียว (Single Inheritance) และถ่ายทอดคุณสมบัติจากหลายไทป์ (Multiple Inheritance) โดยใน SQL3 สามารถถ่ายทอดคุณสมบัติได้ทั้งในระดับแอททริบิว, Row Type , ADT , เทเบิล , คอนสเตรนท์ และ เมธทอด ตัวอย่าง การถ่ายทอดคุณสมบัติ ระดับ Row Type

```
CREATE ROW TYPE PERSON
(   NAME                Char(30),
    SOCIAL-SECURITY     Integer)

CREATE ROW TYPE STUDENT
(   DEGREE              Char(10),
    DEPARTMENT          Char(15))
UNDER     PERSON
```

```
CREATE ROW TYPE TEACHER
(   SALARY              Integer,
    DEPARTMENT          Char(15))
UNDER     PERSON

CREATE ROW TYPE TEACHER_ASSISTANT
UNDER STUDENT,TEACHER
```

จากตัวอย่าง Row Type STUDENT และ TEACHER ได้รับการถ่ายทอดคุณสมบัติจาก Row Type PERSON จะเรียก Row Type STUDENT และ TEACHER ว่าเป็นสับไทป์ของ Row Type PERSON และเรียก Row Type PERSON ว่าเป็นซูเปอร์ไทป์ของ Row Type STUDENT และ TEACHER

การถ่ายทอดคุณสมบัติของ Row Type STUDENT และ TEACHER เป็นการถ่ายทอดคุณสมบัติแบบถ่ายทอดคุณสมบัติจากไทป์เดียวซึ่งถ้าเราต้องการ Row Type ของผู้ช่วยครูซึ่งมีฐานะเป็นทั้งครูและนักเรียนในเวลาเดียวกันสามารถทำการถ่ายทอดคุณสมบัติจากไทป์หลายๆไทป์ได้ ดังตัวอย่างการถ่ายทอดคุณสมบัติของ TEACHER_ASSISTANT

การถ่ายทอดคุณสมบัติแบบถ่ายทอดจากไทป์หลายๆไทป์อาจเกิดกรณีการที่มีแอททริบิวซ้ำกันจากตัวอย่างแอททริบิวที่ซ้ำกันได้แก่ NAME , SOCIAL-SECURITY และ DEPARTMENT โดย NAME และ SOCIAL-SECURITY ไม่มีปัญหาเพราะถ่ายทอดคุณสมบัติมาจาก PERSON เหมือนกันแต่ DEPARTMENT จะขัดแย้งเพราะสืบทอดมาจากคนละที่กันเป็นข้อมูลคนละตัวกัน ซึ่งมีทางแก้คือ เปลี่ยนชื่อแอททริบิวนั้นดังตัวอย่าง

```
CREATE ROW TYPE TEACHER_ASISTANT
UNDER   STUDENT    WITH   (DEPARTMANT AS STUDENT-DEPT
UNDER   TEACHER    WITH   (DEPARTMANT AS TEACHER-DEPT
```

ตัวอย่างการถ่ายทอดคุณสมบัติระดับเทเบิล เป็นการสร้างเทเบิล EMPLOYEE จากการถ่ายทอดคุณสมบัติจากเทเบิล PERSON

```
CREATE TABLE PERSON
(   NAME                Char(30),
    SOCIAL-SECURITY     Integer)
```

```
CREATE TABLE EMPLOYEE
(   SALARY              Integer,
    DEPARTMENT          Char(15))
UNDER    PERSON
```

บทที่ 4
อินฟอร์มิกซ์ ยูนิเวอร์แซล เซิร์ฟเวอร์
Informix Universal Server

4.1 บทนำ

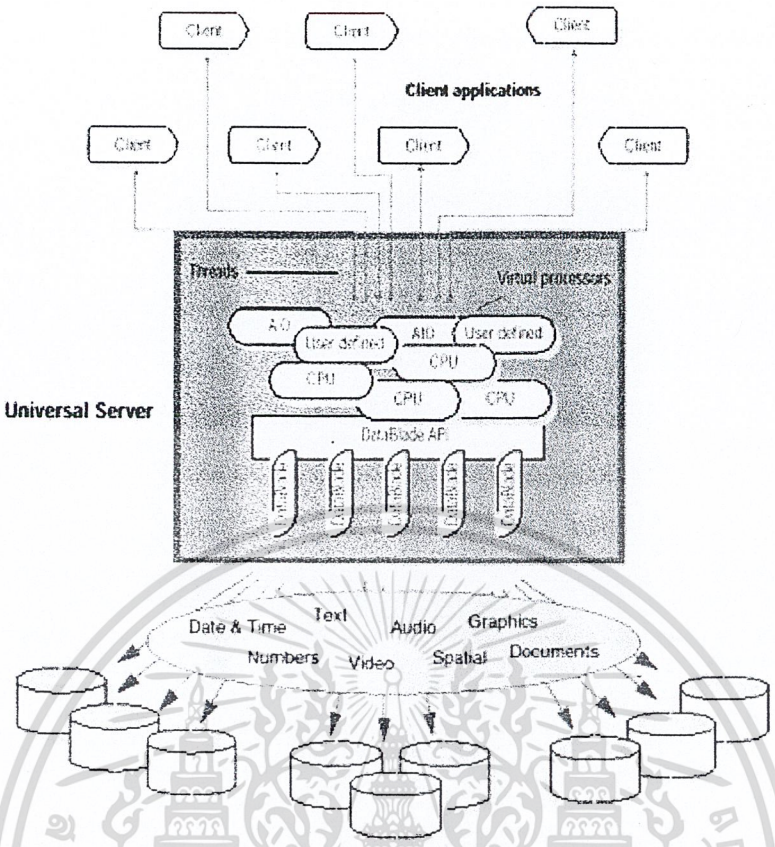
อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์เป็นดาต้าเบสเซิร์ฟเวอร์ (Database Server) ที่สามารถจัดการกับข้อมูลที่มีความซับซ้อนเช่น ภาพ2มิติ3มิติ (2D/3D Images), ข้อมูลเสียง (Sound), ข้อมูลวิดีโอ (Video),เอกสารทางอิเลคโทรนิคส์ (Electronic Documents),เอกสารเอชทีเอ็มแอล (HTML Pages)เป็นต้น โดยเป็นระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database Management System) โดยมีลักษณะพิเศษต่างๆดังนี้

- Data Types Supported
  - Built-In
  - User-Defined
  - Complex
- User-Defined Routines
- DataBlade Modules
- Large-Object Support
- Row Type,Typed Table Inheritance

4.2 สถาปัตยกรรมของอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์

สนับสนุนการทำงานแบบไคลเอนท์/เซิร์ฟเวอร์ (Client/Server) การติดต่อระหว่างโปรแกรมประยุกต์ไคลเอนท์กับดาต้าเบสเซิร์ฟเวอร์ โดยฝั่งเซิร์ฟเวอร์จะประกอบไปด้วย

- **Parser** ทำหน้าที่อ่านและตรวจสอบไวยากรณ์ โดยฟังก์ชั่นที่ผู้ใช้สร้างขึ้นมา (User-defined Function) และชนิดข้อมูลจะถูกบันทึกลงในแค้ตตาลอกเทเบิล (Catalog Table) และจะถูกตรวจสอบโดยพาร์เซอร์
- **Optimizer** ทำหน้าที่เลือกทางที่ดีที่สุดเพื่อการเข้าถึง (Access) ข้อมูลที่ถูกคิวรี (Query)
- **Function Manager** ทำหน้าที่หาและจัดการ User-defined Function เพราะว่าฟังก์ชั่นที่ผู้ใช้สร้างขึ้นมาจะถูกเก็บไว้ในแค้ตตาลอกเทเบิล
- **Access Method** ที่เหมาะสมจะถูกเลือกมาสำหรับข้อมูลที่กำลังถูกแก้ไข
- **Data Manager** ทำหน้าที่เคลื่อนย้ายข้อมูลเข้าออกจากดิสก์



รูปที่ 4.1 สถาปัตยกรรมอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์

ที่ฝั่งไคลเอนท์จะสร้างลอจิกคอลคอนเนคชั่น (Logical Connection) กับเซิร์ฟเวอร์โดยมีชนิดของการเชื่อมต่อ ดังนี้

- **Local Connection** ได้แก่ Shared Memory และ Stream Pipe
- **Network Connection** ได้แก่ Computer-to-Computer และ Local Loopback

Interface	Network Protocol
Socket	TCP/IP
TLI	TCP/IP
TLI	SPX/IPX

ตารางที่ 4.1 ชนิดของอินเตอร์เฟซกับโปรโตคอลที่ยูนิเวอร์แซลเซิร์ฟเวอร์สนับสนุน

มีส่วนประกอบอื่นๆที่จำเป็นต่อการใช้งานต่างๆดังนี้

4.2.1 Connectivity File : SQLhosts เป็นไฟล์ที่เก็บข้อมูลที่ไคลเอนท์ใช้ในการเชื่อมต่อกับเซิร์ฟเวอร์ ชื่อของคาต้าเบสเซิร์ฟเวอร์,ชนิดของอินเตอร์เฟซและโปรโตคอล , ชื่อของโฮสต์ , Servicename Field ,Options Field

4.2.2 ONCONFIG Parameters สำหรับการเชื่อมต่อ เป็นคอนฟิกูเรชั่นไฟล์ (Configuration File) อยู่ใน $INFORMIXDIR/etc Directory ระหว่างการติดตั้งเก็บข้อมูลที่ใช้ในการกำหนดค่าเริ่มต้น (Initial Setting) และพารามิเตอร์ในการคอนฟิก (Configuration Parameter)

4.2.3 Utility Enhancement ตัวอย่างของยูทิลิตี้ที่ใช้งาน

- Oninit — Initialize Disk Space ต้องล็อกอิน (Log In) เป็นรูท (Root)หรือยูเซอร์อินฟอร์มิกซ์ (User Informix) เพื่อที่จะเรียกใช้ oninit
- Onmode — ต้องล็อกอินเป็นรูท หรือ ยูเซอร์อินฟอร์มิกซ์ onmode จะทำการเปลี่ยนโหมดในการทำงานของยูนิเวอร์แวลเซิร์ฟเวอร์ (Universal Server Operating Mode)
- Onspace — ต้องล็อกอินเป็นรูท หรือ ยูเซอร์อินฟอร์มิกซ์ โดยจะต้องทำการกำหนด Create/Drop Sbspace หรือ Dbspace , Add/Drop Chunk

onmode กับ onspace ต้องกำหนดก่อนที่จะทำยูทิลิตี้อื่นๆ

4.2.4 SetNet32 ด้วย SetNet32 Utility สามารถกำหนดหรือแก้ไข Environment Variable และ Network Parameter ขณะรันไทม์ (Runtime) Setnet32 เป็นยูทิลิตี้ที่ติดตั้งบนไคลเอนท์พีซี, Setnet32 Utility มี 4 ส่วนที่สำคัญ คือ

- Environment Enable ใช้กำหนด Environment Variables
- Server Information ใช้กำหนดข้อมูลของคาต้าเบสเซิร์ฟเวอร์
- Host Information ใช้กำหนดคอนฟิกของโฮสต์แมชชีน (Host Machine)
- About Setnet32 ใช้แสดงข้อมูล Setnet32 Utility

4.2.5 Schema Knowledge เป็นทูลที่แสดงข้อมูลต่างๆในรูปแบบของกราฟิก (Graphical Knowledge Tool) ใช้แสดงเมต้าดาต้า (Meta Data) ของฐานข้อมูล ไม่สามารถใช้แก้ไขข้อมูลใดๆบนฐานข้อมูลได้ เราสามารถใช้ Schema Knowledge ดูเมต้าดาต้าต่อไปนี้ได้

- Database
- Routine
- Table และ View

- Row Type
- Inheritance View

ก่อนจะเข้ามาใช้ต้องมีการล็อกอิน ก่อนโดยจะต้องทำการใส่ ชื่อเซิร์ฟเวอร์,ชื่อฐานข้อมูล, ชื่อผู้ใช้และรหัสผ่าน

4.2.6 SQLEditor เป็นทูลที่แสดงผลเป็นกราฟิก (Graphical Tool) ซึ่งใช้ทำการติดต่อกับยูนิเวอร์แซลเซิร์ฟเวอร์ และทำการคิวรีข้อมูลจากฐานข้อมูล (Query Database Information) และดูผลของการคิวรีของ SQL Statement ต้องมีการกำหนดค่าต่างๆใน SetNet32

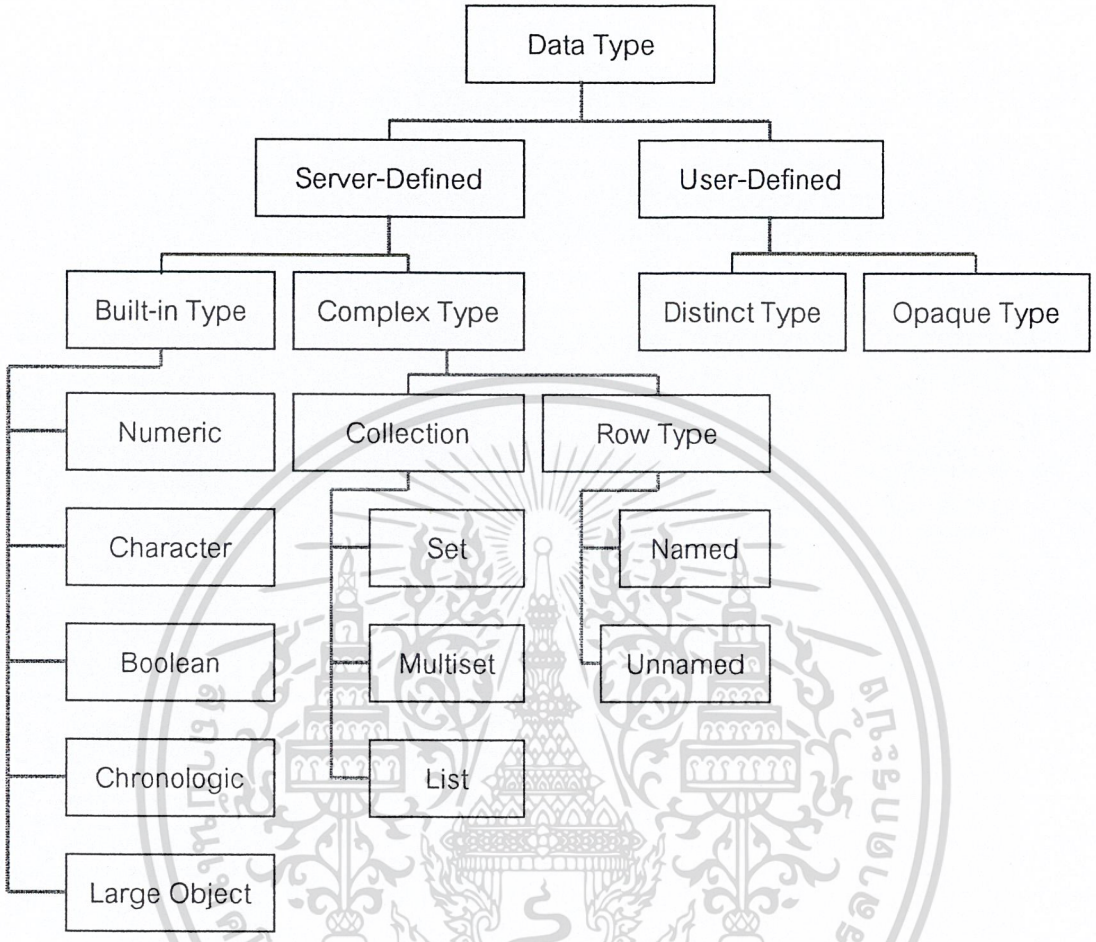
4.2.7 DBDK : DataBlade Developer Kit เป็นทูลที่ใช้พัฒนาดาต้าเบลดโมดูล(Datablade Module) มีรายละเอียดอยู่ในส่วนถัดไป

4.2.8 Iconnect เป็น ODBC ของยูนิเวอร์แซลเซิร์ฟเวอร์

4.3 คุณสมบัติของ Universal เซิร์ฟเวอร์

- มีความสามารถสูง (High Performance) สนับสนุนการทำงานที่เป็น Raw Disk Management , Memory Management (Dynamic Sharing Memory, Buffering Transaction) , Dynamic Thread Allocation และ Parallelization
- มีความสามารถในการจัดการ Logging-Recovery เพื่อทำการควบคุมความถูกต้องของฐานข้อมูล (Integrity Consistency) เมื่อมีเดียชำรุด (Media Failed)
- สนับสนุนการทำ Fast Recovery,Mirroring,Data Replication
- Distributed Database สนับสนุนสถาปัตยกรรมในแบบกระจาย (Distributed)
- ยูนิเวอร์แวลเซิร์ฟเวอร์สนับสนุนการใช้งานหลายภาษา (Global Language) โดยมี Global Language Support Function (GLS) สนับสนุนการทำงานหลายภาษา

4.4 ลักษณะเด่นต่างๆของอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์



รูปที่ 4.2 ชนิดข้อมูลที่อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์สนับสนุน

4.4.1 ชนิดข้อมูล

ก่อนจะกล่าวถึงข้อมูลชนิดต่างๆขอกล่าวถึงโอเปอร์เรชั่น (Operation) , โอเปอร์เรเตอร์ คลาส (Operator Class) และ แคสติงฟังก์ชั่น (Casting Function) ก่อน ดังนี้

- *Operation* แบ่งเป็น 4 ชนิด คือ

1. Operator Function ใช้ในการสร้างโอเปอร์เรเตอร์ ซิมโบล (Operator Symbol) เช่น " + " โอเปอร์เรเตอร์ฟังก์ชั่นที่มีให้กับชนิดข้อมูลที่เป็นบิวท์อิน ดังนี้

Arithmetic Operator

+(binary)	plus()
-(binary)	minus()
*	times()
/	divide()
+(unary)	positive()
-(unary)	negative()

Text Operator

LIKE	like()
MATCHES	matches()
||	concat()

Relational Operator

=	equal()
<> and !=	notequal()
>	greaterthan()
<	lessthan()
>=	greaterthanorrqual()
<=	lessthanorequal()

2. Built-in Function เป็นฟังก์ชั่นที่กำหนดให้ใช้ใน SQL Statement เป็นโอเปอร์เรชั่นทางคณิตศาสตร์ (Mathematical Operation) เช่น abs() , mod() , pow() , root() , round() , sqrt() , trunc() , exp() , log() , logn() , sin() , cos() , tan() , asin() , acos() , atsn() , atan2() , hex() , length() , char_length() , character_length() , octet_length() , user() , current() ,dbservername() , sitename() , today() , extend()
3. Aggregate Function เป็นฟังก์ชั่นที่มีการส่งค่ากลับเป็นค่าเดียว (Single Value) หรือหลายค่า (Set of Value) ใช้ใน SQL Statement ได้แก่ COUNT , AVG , MIN , MAX และ SUM

4. End-user Routines เป็นรูทีนที่ผู้ใช้สร้างขึ้น (User-defined Routine) มีวิธีการกำหนด ได้ 2 แบบ คือ เขียนโดยใช้ภาษา SPL (Stored Procedure Language) และ ใช้ External Language (เช่น ภาษาซี) มี 2 ชนิด คือ ฟังก์ชั่น (Function) ซึ่งจะต้องมีการส่งค่ากลับ และ โพรซีเดอร์ (Procedure)

- *Operator Class* คือกลุ่มของฟังก์ชันที่ทำหน้าที่ในการเก็บและค้นหาค่าต่างๆของเฉพาะของชนิดข้อมูล (Particular Data Type) ที่ถูกกำหนดไว้ให้เป็นดัชนี (Index) แบ่งออกเป็น 2 ชนิดคือ
  1. Strategy Function ทำหน้าที่ช่วยคิวรีออพติไมเซอร์ (Query Optimizer) ตัดสินใจว่าดัชนีจะถูกนำมาใช้แล้วเกิดประโยชน์กับการจัดการข้อมูลหรือไม่
  2. Support Function ใช้ในการสร้างและเข้าถึงดัชนี

- *Casting Function* ถือเป็นโอเปอร์เรเตอร์ฟังก์ชั่นอีกประเภทหนึ่ง ใช้สำหรับแปลงชนิดข้อมูลจากชนิดหนึ่งเป็นอีกชนิด

สำหรับชนิดข้อมูลที่อินฟอร์มิกซ์สนับสนุนประกอบไปด้วยข้อมูลชนิดบิวท์อิน (Built-In Type), ข้อมูลชนิดคอมเพล็กซ์ (Complex Type) และ ข้อมูลชนิดที่อนุญาตให้ผู้ใช้สร้างขึ้น (User-Defined Type) ดังนี้

4.4.1.1 Built-in Data Type

เป็นข้อมูลชนิดพื้นฐานไม่สามารถแบ่งย่อยได้อีก (Atomic Value) ประกอบไปด้วย

- *Character* : CHAR, CHARACTER VARYING (or VARCHAR),LVARCHAR
- *Numeric*
  - Exact Numeric Type : DECIMAL, MONEY, SMALLINT, INTEGER,INT8, SERIAL, และ SERIAL8
  - Approximate Numeric Type : SMALLFLOAT และ FLOAT
- *Miscellaneous* : BOOLEAN
- *Time* : DATE, DATETIME, และ INTERVAL
- *Large-object*
  - Simple-Large-Object Type : TEXT และ BYTE
  - Smart-Large-Object Type : CLOB และ BLOB

4.4.1.2 Complex Type

ชนิดข้อมูลที่ประกอบไปด้วยข้อมูลโอเปอร์เรเตอร์ ชนิดอื่นๆไม่ว่าจะเป็นบิวท์อิน, โอเปค (Opaque) ,ดิสทิลท์ (Distinct) หรือ ข้อมูลชนิดคอมเพล็กซ์ (Complex Type) อื่นๆ Complex Type แบ่งเป็นสองชนิดคือคอลเลคชั่น (Collection Type) และ Row Type

- **Collection Type**

กลุ่มของสมาชิก (Element) ที่มีชนิดข้อมูลชนิดเดียวกัน สามารถจัดการได้เหมือนเป็น Row เดียว (Single Row) ประกอบไปด้วย 2 คอมโพเนนท์ (Component) คือ

- Type Structor ได้แก่ เซต (Set), มัลติเซต (Multiset) , ลิสท์ (List)
  - *Set* สมาชิกไม่ซ้ำและสมาชิกแต่ละตัวไม่มีลำดับ
  - *Multiset* สมาชิกซ้ำกันได้ (มีมากกว่า 1 ตัวได้) และแต่ละตัวไม่มีลำดับสมาชิก
  - *List* สมาชิกซ้ำกันได้และสมาชิกมีลำดับ
- Element Type หรือ Data Type เป็นข้อมูลชนิดใดก็ได้ยกเว้น SERIAL และ SERIAL8

ตัวอย่างการ กำหนดข้อมูลชนิดคอลเลคชั่น โดยแต่ละชนิดมีรูปแบบเดียวกันต่างกันแค่คีย์เวิร์ดที่ใช้ในการกำหนดชนิด

```
SET (<component list>)
MULTISET (<component list>)
LIST (<component list>)
```

ตัวอย่าง

```
CREATE TABLE Employee
(name CHAR(30),
address CHAR (40),
salary INTEGER,
dependents SET(VARCHAR(30) NOT NULL));

CREATE TABLE Person
(name CHAR(30),
address CHAR (40),
salary INTEGER,
```

bonus **MULTISET**(MONEY NOT NULL));

CREATE TABLE Sales_person

(name CHAR(30),

month_sales **LIST**(MONEY NOT NULL));

ตัวอย่างการใช้งาน Collection Type

INSERT INTO Employee VALUES ('Puchong','Bangkok',100000,"**set**{'a','b','c'}")

INSERT INTO Person VALUES ('Puchong','Bangkok',100000,"**multiset**{1000,2000,3000,1000}")

INSERT INTO Sales_person VALUES ('Pimphaka',"**list**{20000,10000,10000,30000}")

- **Row Type**

คือฟิลด์หลายๆฟิลด์ที่นำมารวมกัน แต่ละฟิลด์เทียบได้กับคอลัมน์ในเทเบิล, Row Type ไม่สามารถทำคอนสเตรนท์บนแต่ละฟิลด์ได้ สามารถนำไปใช้ CREATE TABLE และสามารถถ่ายทอดคุณสมบัติได้เหมือนกับมาตรฐานของ SQL3 ในแต่ละฟิลด์ประกอบไปด้วยชื่อของตัวแปรและชนิดของข้อมูล Row Type มีสองชนิดคือ

- *Named Row Type* สร้างโดยใช้คำสั่ง CREATE ROW TYPE ซึ่งจะกำหนดชื่อ และ สามารถถ่ายทอดคุณสมบัติได้ เราสามารถนำไปใช้ กำหนด Typed Table ได้และ Row Type สามารถใช้อ้างถึงได้เหมือนกับข้อมูลชนิดอื่นๆ
- *Unnamed Row Type* สร้างโดยใช้คำสั่ง ROW ไม่ต้องกำหนดชื่อ สามารถใช้งานได้เพียงในเทเบิลที่สร้างขึ้นเท่านั้น ไม่สามารถถ่ายทอดคุณสมบัติได้ และไม่สามารถกำหนดเมธอดฟังก์ชั่นได้

การสร้าง Row Type จะต้องใช้คีย์เวิร์ด CREATE ROW TYPE ดังตัวอย่างการ Create Named Row Type ดังนี้

CREATE ROW TYPE <Row Type NAME> (<Field Definition>) UNDER <Super Type NAME>

ตัวอย่างการสร้าง ROW TYPE ADDRESS_t และ CREATE ROW TYPE Employee_t ซึ่งทำการถ่ายทอดคุณสมบัติจาก ROW TYPE PERSON_t

CREATE ROW TYPE ADDRESS_t

```
(STREET         VARCHAR(20),
 CITY           VARCHAR(20),
 STATE          VARCHAR(20));

CREATE ROW TYPE Employee_t
 (SALARY        NUMERIC(10.2),
  BONUS         NUMERIC(10.2))
UNDER  Person_t;
```

สำหรับการกำหนด Named Row Type นั้นไม่สามารถ กำหนดฟิลด์เป็น SERIAL หรือ SERAIL8 ได้ และ Named Row Type สามารถนำไปใช้สร้างคอลัมน์ ,Row Type อื่น และเทเบิลได้ ดังนี้

```
CREATE TABLE Emp_Add
 (NAME          CHAR(20),
  ADDRESS       ADDRESS_t,);

CREATE TABLE Employee OF TYPE Employee_t
```

การอ้างถึง Named Row Type ทำได้โดยการใช้คีย์เวิร์ด "ROW" ตามด้วยวงเล็บซึ่งภายในจะประกอบไปด้วยฟิลด์ย่อยของ Row Type ดังนี้

```
INSERT INTO Emp_Add
VALUES ('John Bryant',ROW('10 Bay Street','Madera','Californai')::ADDRESS_t);
```

การสร้าง Unnamed Row Type จะแตกต่างจากการสร้าง Name Row Type โดยการ Create Unnamed Row Type ทำได้โดยการใช้คีย์เวิร์ด "ROW" ดังนี้

```
ROW(<component list>)
```

```
CREATE TABLE Student
```

```
(S_NAME ROW(F_NAME VARCHAR(20),M_init CHAR(1),L_NAME VARCHAR(20) NOT NULL),
 S_ADDRESS  ROW(STREET VARCHAR(20),CITY VARCHAR(20),STATE VARCHAR(20)));
```

สำหรับการกำหนด Unnamed Row Type นั้นไม่สามารถ กำหนดฟิลด์เป็น SERIAL, SERAIL8,BYTE,TEXT ได้ ไม่สามารถนำไปอ้างถึงเหมือนกับ Named Row Type ได้ การอ้างถึง Unnamed Row Type ทำได้ดังนี้

```
INSERT INTO Student
VALUES (ROW('Jim','K','Johnson'),ROW('10 Grove Street','Eldorado','California'));
```

4.4.1.3 User-Defined Data Type

แบ่งออกเป็น 2 ชนิดคือ Opaque Type และ Distinct Type

- **Opaque Type**

เป็นชนิดข้อมูลที่ไม่สามารถแบ่งแยกย่อยได้อีก (Single Value หรือ Atomic) โอเปคเป็นชนิดข้อมูลที่ทำการปกป้องข้อมูลได้สมบูรณ์แบบ (Encapsulation) โดยซ่อนโครงสร้างภายใน (Internal Structure)ไว้ ดังนั้นคาต้าเบสเซิร์ฟเวอร์ จะมองไม่เห็นโครงสร้างภายในของโอเปคแต่จะสามารถเข้าถึง (Access) ข้อมูลได้โดยผ่านทางซัพพอร์ทฟังก์ชั่น (Support Function) โดยเราจะต้องกำหนดข้อมูลต่อไปนี้เองโดยเราจะสร้าง ด้วยภาษาซี

- *Data Structure* Internal Structure หรือ โครงสร้างภายในโอเปค ว่าจะประกอบไปด้วยข้อมูลชนิดใดบ้าง
- *Support Function* เป็นฟังก์ชั่นที่ใช้จัดการกับโอเปค เป็นตัวเชื่อมระหว่างคาต้าเบสเซิร์ฟเวอร์กับโอเปค
- *Operation* , *Operator Class*
- *Secondary Access Method* กำหนดการใช้งานดัชนี
- *User Function* เป็นฟังก์ชั่นที่ผู้ใช้สร้างขึ้นเพื่อใช้งานโอเปค
- *Casting* เป็นการแปลงค่าระหว่างข้อมูลที่ต่างชนิดกัน

โดยตัวอย่างการสร้างการใช้งานโอเปคจะอยู่ในบทที่ 5 และ ภาคผนวก ข.

- **Distinct Type**

อนุญาตให้มีการแบ่งแยกความแตกต่างระหว่างข้อมูลชนิดเดียวกันเป็นข้อมูลที่มีโครงสร้างภายในเหมือนชนิดข้อมูลเดิมที่มีอยู่แล้ว ข้อแตกต่างคือเราจะต้อง กำหนดซอร์สดาต้าไทป์ (Source Data Type) ซึ่งได้แก่ โครงสร้างข้อมูลภายใน (Internal Structure) , โอเปอร์เรชั่น (Operation) , โอเปอร์เรเตอร์ คลาส (Operator Class), และ

แคสติง (Casting) โดยปกติระบบจะสร้างแคสติงของดิสทิคท์ไทป์กับซอร์สไทป์ให้โดยอัตโนมัติและดิสทิคท์จะถ่ายทอดคุณสมบัติในการทำแคสติงของซอร์สไทป์กับข้อมูลชนิดอื่นๆมาด้วย การประกาศดิสทิคท์ไทป์ทำได้ดังนี้

CREATE DISTINCT TYPE <Distinct Type NAME> AS <Source Type NAME>

```
CREATE DISTINCT TYPE Fahrenheit  AS Integer;
CREATE DISTINCT TYPE Kelvin  AS Integer;
```

การสร้างเทเบิลโดยใช้ดิสทิคท์ไทป์

```
CREATE DISTINCT TYPE  Dist_typ AS Numeric;
CREATE TABLE  TI(Col1 Dist_typ , Col2 Numeric);
```

ข้อแตกต่างระหว่างข้อมูลชนิดคอมเพล็กซ์ (Complex Type) และ ชนิดข้อมูลที่อนุญาตให้ผู้ใช้ทำการสร้างขึ้น (User-Defined Type) คือ ข้อมูลชนิดคอมเพล็กซ์เราสามารถเข้าถึงแต่ละคอมโพเนนท์ย่อยๆได้โดยใช้ภาษา SQL

4.4.2 Large Object

เป็นชนิดข้อมูลที่ทำการจัดเก็บข้อมูลที่มีขนาดใหญ่แบ่งออกเป็น 2 ชนิดคือ Simple Large Object ซึ่งได้แก่ Text และ Byte กับ Smart Large Object ซึ่งได้แก่ Binary Large Object และ Character Large Object

- **Smart Large Object**

เป็นชนิดข้อมูลที่ลอจิกคอลมองดูเหมือนเก็บอยู่ในเทเบิลแต่ทางกายภาพจริงๆเก็บไว้นอกเทเบิลเพราะข้อมูลมีขนาดใหญ่ ขนาดข้อมูลสูงสุดได้ถึง 4 เทราไบท์ แบ่งออกเป็นสองชนิด Binary Large Object : BLOB และ Character Large Object : CLOB

- Character Large Object : CLOB เก็บข้อมูลที่เป็นเท็กซ์ไฟล์ เช่น PostScript หรือ HTML File
- Binary Large Object : BLOB เก็บข้อมูลที่เป็นไบนารี่ไฟล์ เช่น ภาพหรือเสียง

- **Simple Large Object**

ได้แก่ เท็กซ์ (TEXT) และไบท์ (BYTE) เป็นการเก็บข้อมูลที่เป็นเท็กซ์และไบนารี่เช่นเดียวกับ Smart Large Object เพียงแต่ขนาดจะถูกจำกัดอยู่ที่ 2^{31}

4.4.3 Integrity Constraint

ความสามารถในการควบคุมรักษาความถูกต้องฐานข้อมูล (Integrity Constraint) ของอินฟอร์มิกซ์สนับสนุนแค่การทำ Entity Integrity , Semantic Integrity และ Referential Integrity Constraint

- Entity Integrity ทุกๆ Row จะต้องมีตัวระบุความแตกต่าง (Unique Identifier)
- Semantic Integrity ทุกๆคอลัมน์จะต้องมีค่าถูกต้องตามที่กำหนดไว้
- Referential Integrity Constraint ความสัมพันธ์ระหว่างเทเบิลในเรื่องคีย์หลัก (Primary Key) Foreign Key ต้องถูกต้อง

- **Trigger** ข้อดีของทริกเกอร์

- ลดโค้ดที่ไม่จำเป็นลง (Redundant Code)
- ช่วยในการสร้างกฎต่างๆทางธุรกิจ (Implement Business Rule) เมื่อเราสร้างโปรแกรมประยุกต์ขึ้นใช้งาน
- ทำการได้มาซึ่งข้อมูล (Derive Data)
- ผู้ดูแลระบบฐานข้อมูล (DBA) สามารถใช้ควบคุมการแก้ไขข้อมูลได้โดยแทนที่จะ Grant สิทธิซึ่งควบคุมยากก็ให้ทำเรียกใช้รูทีนแทน

ตัวอย่างการ Create Trigger

```
CREATE TRIGGER ins_tr INSERT ON newtab
REFERENCING        new AS post_ins
FOR EACH ROW (EXECUTE PROCEDURE nt_pct (post_ins.mc));
```

4.4.4 Inheritance

สนับสนุนการถ่ายทอดคุณสมบัติ (Inheritance) ได้แค่ Named Row Type และ Typed Table ในลักษณะการจัดลำดับชั้น (Hierarchy) และให้มีแค่การทำการถ่ายทอดคุณสมบัติจากไทป์เดียว (Single Inheritance) เท่านั้น โดย Type หรือเทเบิลจะสามารถถ่ายทอดคุณสมบัติ (Inheritance Property) และสามารถเพิ่มคุณสมบัติของตัวเองได้อีกด้วย ความสามารถในการถ่ายทอดคุณสมบัติแบ่งได้สองชนิดคือ

- *Type Inheritance* เป็นความสามารถในการถ่ายทอดคุณสมบัติของ Named Row Type เท่านั้น โดยสามารถนำ Named Row Type มาจัดให้สัมพันธ์กันแบบ Type Hierarchy ตัวอย่างการทำ Type Hierarchy โดยใช้คีย์เวิร์ด "UNDER" ดังตัวอย่าง

UNDER <Supertype name>

```
CREATE ROW TYPE Person_t
(NAME          VARCHAR(30) NOT NULL,
 ADDRESS       VARCHAR(40));

CREATE ROW TYPE Employee_t
(SALARY        NUMERIC(10.2),
 BONUS         NUMERIC(10.2))
UNDER  Person_t;
```

จากตัวอย่างเป็นการสร้าง ROW TYPE Person_t และ ROW TYPE Employee_t ซึ่งทำการถ่ายทอดคุณสมบัติมาจาก ROW TYPE Person_t

- *Table Inheritance* เทเบิลที่กำหนด จาก Named Row Type จะสามารถทำการถ่ายทอดคุณสมบัติในระดับเทเบิลได้ (Table Inheritance) ได้โดยจะสามารถถ่ายทอดคุณสมบัติคุณลักษณะต่างๆของเทเบิล ดังนี้
  - คอลัมน์ทั้งหมดของซูเปอร์เทเบิล
  - Referential Integrity
  - คอนสเตรนท์ต่างๆ
  - ดัชนี
  - รายละเอียดในการจัดเก็บ (Storage Option)
  - ทริกเกอร์
  - Access Method

เราสามารถทำความสัมพันธ์เป็นลำดับชั้นของเทเบิล (Table Hierarchy) ได้ระหว่างแต่ละเทเบิลโดยเทเบิลที่จะถ่ายทอดคุณสมบัติได้จะต้องเป็น Row Type Table เท่านั้น ตัวอย่างการทำลำดับชั้นของเทเบิล

```
CREATE TABLE Person OF TYPE Person_t;
```

```
CREATE TABLE Employee OF TYPE Employee_t
UNDER Person;
```

จากตัวอย่างถ้าเราต้องการถ่ายทอดคุณสมบัติของเทเบิล Employee จากเทเบิล Person เราจะต้องทำขั้นตอนต่างๆ ดังนี้

1. สร้าง Row Type Person_t
2. สร้าง Typed Table จาก Row Type Person_t
3. สร้าง Row Type Employee_t Inherit มาจาก Row Type Person_t
4. สร้าง Table Employee ซึ่ง Inherit มาจาก Row Type Employee_t

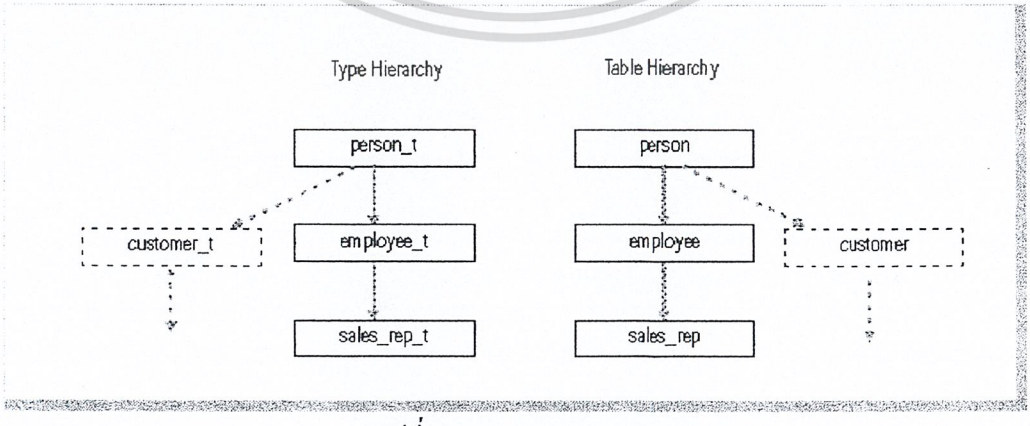
การจัดการกับซับไทป์ เช่นการอินเสิร์ทจะต้องรวมฟิลด์ข้อมูลของซูเปอร์ไทป์เข้าไปด้วยดังตัวอย่าง

```
INSERT INTO EMPLOYEE VALUES ("Puchong Tangjade",
                             "57/10 Soi.Narkbumrong Pomprab Bangkok Thailand",
                             "12,000",
                             "2,000")
```

การ Select ข้อมูลจากซูเปอร์ไทป์จะทำการนำข้อมูลจากสับไทป์มาด้วยหากเราต้องการพิจารณาแค่ข้อมูลของซูเปอร์ไทป์เท่านั้นสามารถทำได้โดยใช้คีย์เวิร์ด " ONLY " ดังตัวอย่าง

```
SELECT * FROM PERSON ONLY (PERSON)
```

เช่นเดียวกับการเรียกใช้รูทีนที่ผู้ใช้สร้างขึ้น (User-defined Routine) เวลาซูเปอร์ไทป์เรียกใช้รูทีนจะทำการเรียกฟิลด์ของซูเปอร์ไทป์ทั้งหมดและเรียกฟิลด์ที่ทำการถ่ายทอดคุณสมบัติของซูเปอร์ไทป์ที่สับไทป์มาด้วย



รูปที่ 4.3 Type และ Table Hierarchy

4.4.5 Polymorphism

เป็นการสร้างรูทีนที่ทำงานแตกต่างกัน หรือ มีอาร์กิวเมนท์ต่างกัน แต่มีชื่อเดียวกัน ซึ่งมีประโยชน์เมื่อ

- สร้างรูทีนที่มีชื่อเหมือนกับรูทีนของบิวท์อินฟังก์ชั่นเพื่อใช้งานกับชนิดข้อมูลที่ผู้ใช้สร้างขึ้นใหม่
- สร้าง Type Hierarchy โดยที่สับไทป์มีการถ่ายทอดคุณสมบัติต่างๆจากซูเปอร์ไทป์รวมถึงรูทีนด้วย
- สร้าง Distinct Type ที่มีโครงสร้างภายในเหมือนกับชนิดข้อมูลเดิมที่มีอยู่แล้ว เพียงแต่มีชื่อต่างกัน ซึ่งจะมีการถ่ายทอดรูทีนที่ใช้ของซอร์สไทป์มาด้วยซึ่งเราสามารถทำโอเวอร์โหลด (Overload) กับรูทีนเหล่านี้

ในการทำรูทีนโอเวอร์โหลดดิง (Routine Overloading) จะใช้ซิกเนเจอร์ (Signature) ของรูทีนในการเลือกหรือกำหนดการใช้งานรูทีนที่ถูกต้องโดยพิจารณาจากข้อมูลเหล่านี้

- ชนิดของรูทีน
- ชื่อของรูทีน
- จำนวนพารามิเตอร์
- ชนิดของพารามิเตอร์ของรูทีน
- ลำดับของพารามิเตอร์

4.4.6 User-Defined Routines

เป็นรูทีนที่สร้างขึ้นมา สามารถทำได้ทั้งแบบเป็น Stored Procedure Language (SPL) ซึ่งเป็น SQL Statement และ External Statement ที่เป็นภาษาอื่นโดยจะสามารถเรียกใช้ SPL Routine ได้การประกาศทำได้ทั้งฟังก์ชั่นและโพรซีเยอร์โดยฟังก์ชั่นสามารถใช้ได้ใน SQL Statement แต่โพรซีเดอร์ไม่ได้เพราะจะไม่มีค่าที่ส่งคืนกลับมา ข้อดีของ UDR คือ

- สามารถรวบรวมคำสั่งภาษา SQL หลายๆคำสั่งไว้ด้วยการเรียกใช้รูทีนเพียงครั้งเดียว (Encapasulate Multiple SQL Statement)
- ขยายขีดความสามารถของบิวท์อินฟังก์ชั่นที่มีอยู่เดิม (Extend Function Built-In Data Type)
- กำหนดว่าใครมีสิทธิอ่านเขียนเปลี่ยนแปลงข้อมูลและสิทธิในการสร้างออบเจกต์
- ใช้จัดการกับ Large Object
- ติดต่อจัดการกับข้อมูลที่เป็นมัลติมีเดีย

การกำหนดรูทีนสามารถกำหนดชื่อซ้ำได้เพราะมีความสามารถทำรูทีนโอเวอร์โหลดดิง (Routine Overloading) ซึ่งอินฟอร์มิกซ์มี Routine Resolution ที่จะช่วยจัดการให้ สำหรับรายละเอียดการสร้างและใช้งาน SPL รูทีนจะกล่าวถึงในบทที่ 6

ตัวอย่างการ กำหนด SPL Routine

```
CREATE FUNCTION Update_by_pct ( pct INTEGER, pit CHAR(10))
        RETURNING    INT;
        DEFINE N  INT;

        UPDATE          inventory SET  price = price + price*( pct/100 )
        WHEREpart_id = pid;

        LET     N = price;
        RETURN    price;
END FUNCTION

CREATE PROCEDURE Raise_price (per_cent INT)
        UPDATE stock SET unit_price = unit_price + unit_price*(per_cent/100);
END PROCEDURE;
```

สำหรับรูทีนภายนอก (External Routine) มีขั้นตอนในการสร้างดังนี้

1. เขียน โค้ดด้วยภาษาที่เป็น External Language เช่น C
2. Compile Source Code แล้วทำการ Load เข้ามาเก็บไว้ใน Shared Library
3. ทำการ ประกาศ Routine ไว้ใน DB เซิร์ฟเวอร์ ซึ่งมีตัวอย่างดังนี้

```
CREATE FUNCTION Equal (argu1 OpaqueTyp1,argu2 OpaqueTyp2)
RETURNING    boolean;
EXTERNAL     NAME  */usr/lib/opaquetyp1/lib/libbtyp1.so(OpaqueTyp1_Equal)*
LANGUAGE     C
END FUNCTION;

CREATE PROCEDURE Check_owner ( owner lVARCHAR)
EXTERNAL     NAME  */usr/lib/ext_lib/genlib.so(unix_owner)*
LANGUAGE     C
```

```
END PROCEDURE;
```

การเรียกใช้รูทีนใช้คำสั่ง EXECUTE , CALL สามารถเรียกใช้ได้จาก SPL , DBACCESS และ ESQL/C สำหรับฟังก์ชั่นจะเรียกใช้ได้จาก SQL Statement ด้วย ตัวอย่างการเรียกใช้ฟังก์ชั่นและตัวอย่างการเรียกใช้โพรซีเดอร์

```
EXECUTE FUNCTION Equal(arg1_udtype1,arg2_udtype1) INTO Result
        CALL Equal(arg1_udtype1,arg2_udtype1)
        RETURNING BOOLEAN...

        EXECUTE PROCEDURE log_compare (arg1, arg2)
        CALL log_compare (arg1, arg2)
```

4.4.7 DataBlade Module

ดาต้าเบลดโมดูล (Datablade Module) เป็นกลุ่มที่รวมออบเจกต์และโค้ดไว้ด้วยกันและสามารถติดตั้งเข้าไปในดาต้าเบสเซิร์ฟเวอร์ได้โดยดาต้าเบสเซิร์ฟเวอร์ จะสนับสนุนการทำงานได้เหมือนกับชนิดข้อมูลที่เป็นบิวท์อิน ดาต้าเบลดโมดูลเป็นโมดูลซอฟท์แวร์มาตรฐานที่สามารถพลักอิน (Plug-In) เข้าไปในฐานข้อมูลเพื่อเพิ่มความสามารถให้กับฐานข้อมูลได้ ดาต้าเบลดโมดูลจะประกอบไปด้วยคอมโพเนนท์ต่างๆดังนี้

- *Data Type* ดาต้าเบลดโมดูลสามารถกำหนดชนิดของข้อมูลหรือกลุ่มของข้อมูลที่ผู้ใช้ทำการสร้างขึ้นมาได้ดังนี้
  1. Built-In เป็นชนิดข้อมูลที่ดาต้าเบสเซิร์ฟเวอร์มีให้
  2. Qualified Built-In เป็นบิวท์อินที่ต้องกำหนดขนาด
  3. Import เป็นชนิดข้อมูลที่นำเข้ามาจากดาต้าเบลดโมดูลอื่นอื่น
  4. Opaque เป็นชนิดข้อมูลที่ปกป้องข้อมูลภายในไว้ (Encapsulation) โดยสร้างจากภาษาซี เราต้องกำหนดส่วนของโค้ดเองด้วย
  5. Distinct เป็นชนิดข้อมูลที่มีพื้นฐานมาจากชนิดข้อมูลอื่นแต่จะมีชื่อเป็นของตัวเอง
  6. Collection เป็นชนิดข้อมูลที่ประกอบด้วยกลุ่มของสมาชิก (Element) ที่มีชนิดข้อมูลเหมือนกัน
  7. Row Type เป็นชนิดข้อมูลที่เป็นตัวบรรจุกลุ่มชนิดข้อมูลต่างๆ
- *Routine Datablade Module* สามารถกำหนดรูทีนเพื่อใช้จัดการกับข้อมูลและส่งคืนค่าข้อมูลตามที่เรากำหนดในรูทีนได้

1. Built-In เป็นฟังก์ชั่นพื้นฐานที่ใช้ในการคำนวณที่สามารถรวมไว้ใน SQL Statement ได้
2. End-User Routine เป็นฟังก์ชั่นที่เรากำหนดขึ้นมาใช้เองและสามารถเรียกใช้ใน SQL Statement
3. Support Function เป็นฟังก์ชั่นที่เซิร์ฟเวอร์จะใช้จัดการกับชนิดข้อมูลชนิดต่างๆ เราต้องสร้างซัพพอร์ทฟังก์ชั่นสำหรับโอเปค
4. Operator Class Support Function เป็นฟังก์ชั่นที่กำหนดถึง Secondary Access Method ที่สร้างดัชนีบนข้อมูลเพื่อเพิ่มประสิทธิภาพในการทำคิวรี

- *Cast* เป็นฟังก์ชั่นพิเศษที่ใช้เพื่อแปลงชนิดข้อมูลหนึ่งไปเป็นชนิดข้อมูลชนิดอื่น
- *Interface* เป็นการกำหนดให้ส่วนหนึ่งหรือทั้งหมดของคาต้าเบลดโมดูลหนึ่งสามารถถูกนำมาใช้ในอีกคาต้าเบลดโมดูลหนึ่งได้
- *Error* เป็นส่วนที่เราสามารถใช้กำหนดการแจ้งข่าวสารข้อผิดพลาดที่เกิดขึ้น (Error Message)
- *Test* เป็นส่วนที่ใช้ทดสอบความผิดพลาดของโปรแกรม โดยจะใส่ข้อมูลเข้าไปและคาดหวังถึงผลลัพธ์ และแสดงโค้ดที่ผิดพลาด (Error Code) ถ้าอินพุทที่ใส่เข้าไปมีข้อผิดพลาด
- *Import File* เป็นส่วนที่รวมไฟล์ที่นำเข้ามาในคาต้าเบลดโมดูล ประกอบด้วย 2 ชนิดคือ
  1. SQL ไฟล์ สามารถรวม SQL Command เมื่อสร้างหรือทำลาย Operator Class , Support Table และ SPL Procedure
  2. ไคลเอนท์ไฟล์ สามารถรวมโปรแกรมประยุกต์,เอกสารประกอบการใช้งานอินเตอร์เฟซ (Graphical User Interface Document) , เฮลป์ไฟล์ (Helpfile) และ โปรแกรมที่จะใช้ติดตั้ง (Installation Program)

- ข้อดีของคาต้าเบลดโมดูล

- Automation Developer Kit จะช่วยสร้าง SQL Script , Shared Object File และ Installation File ให้
- Convenience ง่ายต่อการติดตั้งการอัพเกรดและการยกเลิกใช้
- Modularity ง่ายต่อการออกแบบและสร้างฐานข้อมูลเนื่องจากสามารถแยกพัฒนาแต่ละส่วนออกจากกันได้และสามารถนำมาประกอบกันได้ง่าย

คาต้าเบลดโมดูลเดเวลลอปเปอร์คิทมียูเซอร์อินเตอร์เฟซเป็นกราฟิกสำหรับการสร้างและใช้งานคาต้าเบลดโมดูล ได้แก่

- **BladeSmith** เป็นทูลที่ใช้สร้างโปรเจคและ กำหนดออบเจกต์ต่างๆ เช่นชนิดข้อมูลและรูทีนเป็นต้นเบลดสมิธจะทำการสร้างซอร์สไฟล์, เฮดเดอร์ไฟล์ (Header File) , เมคไฟล์ (Make File) , ฟังก์ชั่นนอลเทสท์ไฟล์ (Functional Test File) , SQL สคริปปี้ (Script) , แมสเสจ (Message) และ แพคเกจจิงไฟล์ (Packaging File)
- **BladePack** เป็นทูลที่จะนำแพคเกจจิงไฟล์ที่เบลดสมิธสร้างซึ่งเป็นคาต้าเบลดโมดูลที่พร้อมจะนำไปติดตั้ง (Installable DataBlade Module) ไปเก็บไว้ในยูนิเวอร์แซลเซิร์ฟเวอร์ เพื่อรอการรีจิสเตอร์ (Registry)

- **BladeManager** เป็นยูทิลิตี้ที่ทำการลงทะเบียนและยกเลิกการลงทะเบียนคาต้าเบลดโมดูลในฐานข้อมูล

บทที่ 5

Large Object และ Opaque Type

ในบทนี้จะกล่าวถึง Large Object และ Opaque Type ในด้านการสร้าง รูปแบบและวิธีการใช้งานอีกทั้งข้อแตกต่างระหว่างการใช้งาน Large Object และ Opaque Type

5.1 Large Object

เป็นชนิดข้อมูลที่ทำการจัดเก็บข้อมูลที่มีขนาดใหญ่แบ่งออกเป็น 2 ชนิดคือ Simple Large Object ซึ่งได้แก่ Text และ Byte กับ Smart Large Object ซึ่งได้แก่ Binary Large Object และ Character Large Object

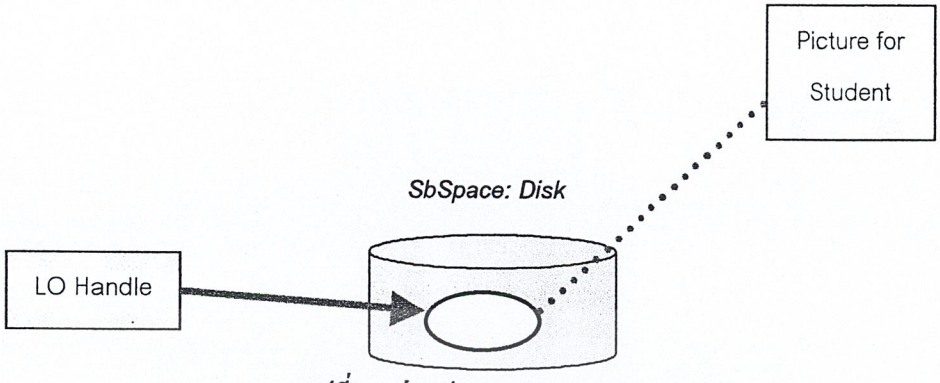
5.1.1 Smart Large Object ได้แก่ Binary Large Object และ Character Large Object คุณสมบัติของ Smart Large Object

- สามารถทำการจัดเก็บและดึงข้อมูลจากฐานข้อมูล (Store/Retrive) และทำความสามารถอื่นๆเช่น Transaction Roll Back หรือ Recovery ได้
- ขนาดของ Smart Large Object ไม่จำกัด มีขนาดใหญ่สุดได้ถึง 4 เทราไบท์ (TeraByte)
- สามารถเข้าถึงข้อมูลโดยใช้ SQL Statement
- การเก็บข้อมูลในคอลัมน์ที่เรากำหนดชนิดให้เป็น Smart Large Object จะทำการเก็บเพียงพอยน์เตอร์ (LO Handle) เท่านั้น
- คาต้าเบสเซิร์ฟเวอร์จะไม่ส่งข้อมูลให้กับโปรแกรมประยุกต์ที่ไคลเอนท์แต่จะส่งแค่พอยน์เตอร์ไปให้โปรแกรมประยุกต์ที่ไคลเอนท์และโปรแกรมประยุกต์ที่ไคลเอนท์จะใช้พอยน์เตอร์นี้ติดต่อกับคาต้าเบสเซิร์ฟเวอร์ในการเขียนอ่านข้อมูล บน Smart Large Object
- BLOB และ CLOB ไม่มีความสามารถในการทำแคสติง ไม่สามารถเปลี่ยนเป็นข้อมูลชนิดอื่นๆได้

Smart Large Object แบ่งออกเป็น 2 ชนิดคือ

- **Character Large Object :** CLOB เก็บข้อมูลที่เป็นเท็กซ์ไฟล์ขนาดใหญ่ (Block of Text) โดยปกติใช้เก็บข้อมูลที่เป็นแอสกี (ASCII Text Data) รวมไปถึงข้อมูลที่เป็นโพสท์สคริปป์ (PostScript) หรือเอชทีเอ็มแอลไฟล์ (HTML File) แต่อินฟอร์มิกซ์อนุญาตให้เก็บเพียงแค่แอสกีที่สามารถพริ้นท์ออกมาได้ (ASCII Printable Text Data) เท่านั้น ซึ่งการใช้ CLOB มีข้อดีคือ สามารถอ่านหรือเขียนส่วนใดของไฟล์ก็ได้ และ สามารถใช้โอเปอร์เรเตอร์ในการเปรียบเทียบเท่ากับ (Equal Opertor) " = " ในการเปรียบเทียบ CLOB สองตัวได้
- **Binary Large Object :** BLOB เก็บข้อมูลที่เป็นไบนารี่ไฟล์ เช่น ภาพหรือเสียงได้มีข้อดีคือ สามารถอ่านหรือเขียนส่วนใดของไฟล์ก็ได้

5.1.2 ส่วนประกอบของ Smart Large Object ประกอบไปด้วย 2 ส่วนคือ



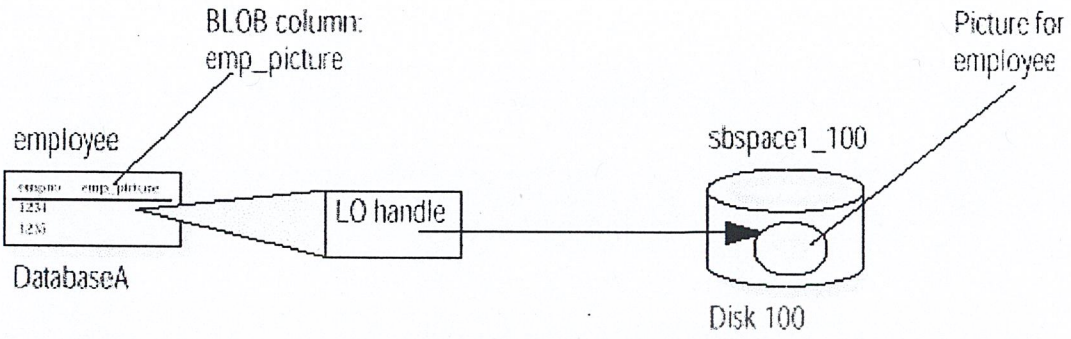
รูปที่ 5.1 ส่วนประกอบของ Smart Large Object

1. ***Sbspace*** เป็นพื้นที่ที่ใช้เก็บ Smart Large Object โดยเฉพาะ จะประกอบไปด้วยส่วนต่างๆ ดังนี้

- *Metadata Area* ใช้เก็บข้อมูลต่างๆของ Smart Large Object ซึ่งรวมถึง
  - Internal Information ซึ่งจะช่วยให้การจัดการกับข้อมูลทำได้อย่างมีประสิทธิภาพ
  - Stroage Characteristic ของ Smart Large Object ประกอบไปด้วย
    - Disk Storage Information หรือข้อมูลทั้งหมดเกี่ยวกับการจัดเก็บบนดิสก์ ได้แก่ Allocation Extend Information , Sizing Information , Location Information
    - Attribute Information เป็นข้อมูลเกี่ยวกับการกำหนดออพชั่น (Option) ต่างๆเกี่ยวกับการเข้าถึงข้อมูล ประกอบด้วย Logging Indicator , Last Access Time เป็นต้น

    ซึ่งการระบุ Storage Characteristic ทำได้หลายวิธี ดังนี้
    - สร้าง Sbspace ใช้ onspace Utility ซึ่งสามารถกำหนดพารามิเตอร์ต่างๆได้
    - สร้างเทเบิลโดยใช้คีย์เวิร์ด PUT CREATE TABLE Statement
    - สร้าง Smart Large Object โดยใช้ DataBlade API
  - Status Information ของ Smart Large Object เก็บสถานะของข้อมูล
- *User Data Area* ใช้เก็บตัวข้อมูล Smart Large Object

2. LO handle เป็นพอยน์เตอร์ที่ชี้ตำแหน่งของ Smart Large Object ที่เก็บอยู่บน Sbspace



รูปที่ 5.2 Smart Large Object :BLOB

5.1.3 การใช้งาน Smart Large Object

การใช้งาน BLOB,CLOB เราสามารถอ้างถึงได้เหมือนเป็นข้อมูลชนิดบิวท์อินชนิดอื่นๆ ดังตัวอย่าง

```
CREATE TABLE Student
(ID          integer,
IMAGE        BLOB,
TEXT         CLOB);

CREATE TABLE Probation_list
(ID          integer,
PICT         BLOB) PUT PICT IN (sbspace1);
```

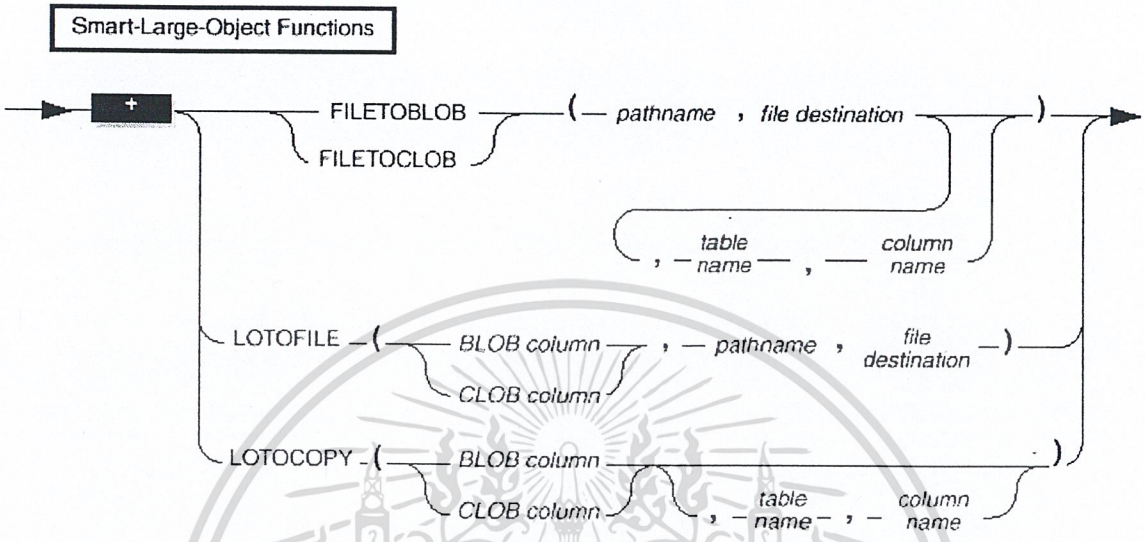
5.1.4 การเข้าถึง Smart Large Object สามารถทำได้ 2 วิธีคือ

1. ใช้บิวท์อินฟังก์ชั่นที่มีให้
2. เรียกใช้ผ่านคาต้าเบลดเอพีไอ (Datablade API)

5.1.4.1 SQL Built-in Function ที่ใช้งานกับ BLOB,CLOB

- **FILETOBLOB** โอนโอเอสไฟล์มาใส่ใน BLOB คอลัมน์หรือ
- **FILETOCLOB** โอนโอเอสไฟล์มาใส่ใน CLOB คอลัมน์

- **LOTOFILE** โอน BLOB หรือ CLOB มาเก็บในโอเอสไฟล์ (Operating-System File)
- **LOCOPY** โอน Smart Large Object อันหนึ่งมาเก็บใน Smart Large Object ใหม่



รูปที่ 5.3 Built-in Function ของ Smart Large Object

การใช้งานโดยผ่านบิวท์อินฟังก์ชั่นจะเป็นการเรียกผ่าน SQL สเตตเมนท์ดังตัวอย่าง

```
INSERT INTO Student VALUES(1,FILETOBLOB('C:\Sand.bmp','Server'),FILETOCLOB
('C:\Sand.txt','Server'))

SELECT LOTOFILE(IMAGE,'C:\Image.bmp','Server'),LOTOFILE(TEXT,'C:\Text.txt','Server')
FROM   Student
WHERE  ID = 1

        UPDATE Student (IMAGE)
            SET IMAGE = LOCOPY(PICT,'Probation_list','PICT')
        WHERE  STUDENT.ID = 378 AND Probation_list = 912
```

การอินเสิร์ท BLOB และ CLOB จะใช้บิวท์อินฟังก์ชั่น FILETOBLOB หรือ FILETOCLOB ในคำสั่ง INSERT INTO ส่วนการ Select ข้อมูลที่เป็น BLOB และ CLOB จะใช้บิวท์อินฟังก์ชั่น LOTOFILE ในคำสั่ง SELECT การอัพเดทข้อมูลใช้บิวท์อินฟังก์ชั่น LOCOPY ในคำสั่ง UPDATE

5.1.4.2 การติดต่อผ่าน Datablade API

นอกจากการติดต่อผ่าน Built-in Function แล้วยังสามารถติดต่อผ่านดาต้าเบลดเอพีไอ ซึ่งเป็นวิธีการติดต่อที่เราใช้ในการติดต่อโดยตรงผ่านการโปรแกรมมิ่ง ดังต่อไปนี้

ตารางที่ 5.1 เป็น API พื้นฐานที่ใช้ในการจัดการ Smart Large Object ได้แก่ API ที่ใช้ในการสร้างแบบต่างๆ

Smart-Large-Object Function	Smart-Large-Object Operation
mi_lo_create()	สร้าง smart large object ใหม่และทำเป็น Empty ไว้
mi_lo_copy()	สร้าง smart large object ใหม่และทำการก๊อปปี้ค่าจาก smart large object อื่นที่มีอยู่แล้ว
mi_lo_expand()	สร้าง smart large object ใหม่จากข้อมูลที่เป็น multirepresentational data
mi_lo_from_file()	สร้าง smart large object ใหม่โดยนำข้อมูลมาจาก operating-system file

ตารางที่ 5.1 สร้าง Smart Large Object ใหม่

ตารางที่ 5.2 เป็น API ที่ใช้ในเข้าถึงและแก้ไข Smart Large Object การเปิดการอ่านการเขียนเป็นต้น

Smart-Large-Object Operation	Smart-Large-Object Function	Operating-System System Call
เปิดการติดต่อกับ smart large object จะได้ LO handle เป็นตัว identifier ในการทำการ open จะทำการ generates LO file descriptor จาก smart large object.	Mi_lo_open()	Open()
ค้นหาตำแหน่งที่ต้องการตาม LO seek position เพื่อทำการเริ่มต้นการเขียนอ่าน	Mi_lo_seek()	Seek()
เรียกดูค่าตำแหน่งปัจจุบันของ LO seek	Mi_lo_tell()	tell()
	Mi_lo_read(),	read(),

ทำการอ่านเขียนข้อมูลเป็นจำนวนไบท์ตามที่กำหนด	mi_lo_readwithseek(), mi_lo_write(), mi_lo_writewithseek()	write()
เรียกดูค่าสถานะของข้อมูลเฉพาะของ smart large object.	Mi_lo_stat()	stat()
ทำการ Truncate smart-large-object ณ ตำแหน่งที่กำหนด	Mi_lo_truncate()	truncate()
ปิดการติดต่อกับ smart large object และทำการคืนค่า LO file descriptor.	Mi_lo_close()	close()

ตารางที่ 5.2 การเข้าถึงและการแก้ไขข้อมูลใน Smart Large Object

ตารางที่ 5.3 เป็น API ที่ใช้ในการจัดการกับ Lo Handle

Smart-Large-Object Function	Smart-Large-Object Operation
mi_get_lo_handle()	Obtains an LO handle from a user-defined buffer
mi_lo_alter()	Alters the storage characteristics of the smart large object that the LO handle identifies
mi_lo_copy()	Copies the contents of a smart large object (that an LO handle identifies) into a new smart large object and initializes the LO handle of the new smart large object
mi_lo_create()	Creates a new smart large object and initializes its LO handle
mi_lo_decrefcount()	Decrements the reference count of the smart large object that the LO handle identifies
mi_lo_expand()	Copies multirepresentational data into a new smart large object and initializes the LO handle
mi_lo_filename()	Returns the name of the file where the mi_lo_to_file() function would store the smart large object that the LO handle identifies
mi_lo_from_buffer()	Copies a specified number of bytes from a user-defined buffer into a smart large object that the LO handle identifies
mi_lo_from_file()	Copies the contents of an operating-system file to a smart large object that the LO handle identifies

mi_lo_from_string()	Converts an LO handle from its text representation to its binary representation
mi_lo_increfcount()	Increments the reference count of the smart large object that the LO handle identifies
mi_lo_invalidate()	Marks an LO handle as invalid
mi_lo_lolist_create()	Converts an array of LO handles into an MI_LO_LIST structure
mi_lo_open()	Opens the smart large object that the LO handle identifies
mi_lo_ptr_cmp()	Compares two LO handles to see if they identify the same smart large object
mi_lo_release()	Releases resources held by a temporary smart large object, including its LO handle
mi_lo_to_buffer()	Copies a specified number of bytes from a smart large object that the LO handle identifies into a user-defined buffer
mi_lo_to_file()	Copies the smart large object that the LO handle identifies to an operating-system file
mi_lo_to_string()	Converts an LO handle from its binary representation to its text representation
mi_lo_validate()	Checks whether an LO handle is valid
mi_put_lo_handle()	Puts an LO handle into a user-defined buffer

ตารางที่ 5.3 การจัดการกับ LO handles

ตารางที่ 5.4 เป็นตารางที่ใช้ในการจัดการกับสถานะของข้อมูล เช่นใช้ในการแรียกดูข้อมูลของสถานะเป็นต้น

Smart-Large-Object Function	Smart-Large-Object Operation
mi_lo_stat()	Allocates and initializes an LO-status structure with status information of an open smart large object
mi_lo_stat_atime()	Accessor function to get the last-access time
mi_lo_stat_cspec()	Accessor function to get the storage characteristics
mi_lo_stat_ctime()	Accessor function to get the last-change time
mi_lo_stat_free()	Frees the resources of the LO-status structure
mi_lo_stat_mtime()	Accessor function to get the last-modification time
mi_lo_stat_refcnt()	Accessor function to get the reference count

mi_lo_stat_size()	Accessor function to get the size of smart large object

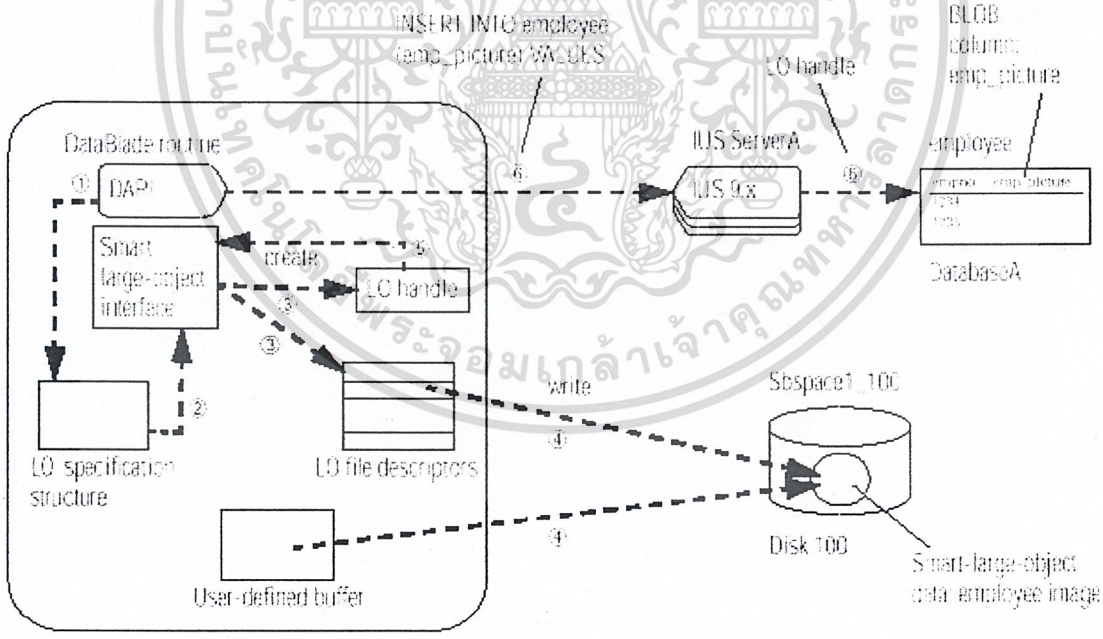
ตารางที่ 5.4 ข้อมูลสถานะของ Smart Large Object

ตารางที่ 5.5 เป็นตารางที่แสดง API ที่ใช้ในการจัดการ Smart Large Object กับ OS File

Smart-Large-Object Function	Smart-Large-Object Operation
mi_file_to_file()	คัดลอกค่าของ Smart Large Object จาก operating-system file หนึ่งไปยังอีกไฟล์
mi_lo_from_file()	คัดลอกค่าของ Smart Large Object จาก operating-system file ไปยังคอลัมน์ที่เป็น smart large object โดยใช้พารามิเตอร์เป็นชื่อไฟล์
mi_lo_from_file_by_lofd()	คัดลอกค่าของ Smart Large Object จาก operating-system ไปยัง smart large object โดยใช้พารามิเตอร์เป็น LO File Descritor
mi_lo_to_file()	คัดลอกค่าของคอลัมน์ Smart Large Object ไปยัง operating-system file ใหม่

ตารางที่ 5.5 Smart Large Objects to and from Operating-System Files

- การใช้งาน BLOB และ CLOB ผ่าน Datablade API มีขั้นตอนตามรูปที่ 5.4 และตารางที่ 5.6 ดังนี้



รูปที่ 5.4 Step Task Smart-Large-Object Function For More Information

TASK	Smart Large Object API
1. Obtain an LO-specification structure to hold the storage characteristics for the new smart large object. 2. Ensure that the LO-specification structure contains the desired storage characteristics for the new smart large object.	mi_lo_spec_init(), mi_lo_stat_cspec() System-specified storage characteristics : mi_lo_spec_init() Column-level storage characteristics: mi_lo_colinfo_by_name(), mi_lo_colinfo_by_ids()
3. Create an LO handle for the new smart large object and open the smart large object.	mi_lo_create(), mi_lo_expand(), mi_lo_copy(), mi_lo_from_file()
4. Write a specified number of bytes from a user-defined buffer to the open smart large object.	mi_lo_write(), mi_lo_writewithseek()
5. Pass the LO handle as the column value for an INSERT or UPDATE statement.	C Casting
6. Execute an INSERT or UPDATE statement to save the LO handle of the smart large object in a database column.	mi_exec(), mi_exec_prepared_statement(), mi_value()
7. Close the smart large object.	mi_lo_close()
8. Deallocate resources.	mi_lo_spec_free(), mi_lo_release()

ตารางที่ 5.6 ขั้นตอนการใช้ Datablade API

5.1.5 Simple Large Object ได้แก่ เท็กซ์ *(TEXT)* และไบท์ *(BYTE)* เป็นการเก็บข้อมูลที่เป็นเท็กซ์และ ไบนารี่เช่นเดียวกับ Smart Large Object เพียงแต่ขนาดจะถูกจำกัดอยู่ที่ 2^{31} ไบท์ สามารถจัดเก็บ,เรียกดูข้อมูล , อัพเดท และ ลบเท็กซ์หรือไบท์คอลัมน์ สามารถใช้กับบูลีนในกรณีที่เช็คว่าเป็น NULL หรือไม่และไม่สามารถใช้กับ

- Arithmetic String Operation
- Aggreate Function
- IN Clause
- MATCHES หรือ LIKE Clause
- GROUP BY Clause

- ORDER BY Clause

สำหรับ Simple Large Object ต่างจาก Smart Large Object คือ คาต้าเบสเซิร์ฟเวอร์จะทำการส่งข้อมูลทั้งหมดไปให้กับโปรแกรมประยุกต์ที่ฝั่งไคลเอนท์ ในขณะที่ Smart Large Object จะทำการส่งเพียงพอยน์เตอร์ไปให้กับโปรแกรมประยุกต์ที่ฝั่งไคลเอนท์

5.2 Opaque Type

เป็นชนิดข้อมูลที่ไม่สามารถแบ่งแยกย่อยได้อีก (Single Value หรือ Atomic) โอเปคเป็นชนิดข้อมูลที่ทำการปกป้องข้อมูลได้สมบูรณ์แบบ (Encapsulation) โดยซ่อนโครงสร้างภายใน (Internal Structure)ไว้ ดังนั้นคาต้าเบสเซิร์ฟเวอร์จะมองไม่เห็นโครงสร้างภายในของโอเปคแต่จะสามารถเข้าถึง (Access) ข้อมูลได้โดยผ่านทางซัพพอร์ทฟังก์ชั่น (Support Function) โดยเราจะต้องกำหนดข้อมูลต่อไปนี้เองโดยเราจะสร้าง ด้วยภาษาซี

- *Data Structure* Internal Structure หรือ โครงสร้างภายในโอเปค ว่าจะประกอบไปด้วยข้อมูลชนิดใดบ้าง
- *Support Function* เป็นฟังก์ชั่นที่ใช้จัดการกับโอเปค เป็นตัวเชื่อมระหว่างคาต้าเบสเซิร์ฟเวอร์กับโอเปค
- *Operation , Operator Class*
- *Secondary Access Method* กำหนดการใช้งานดัชนี
- *User Function* เป็นฟังก์ชั่นที่ผู้ใช้สร้างขึ้นเพื่อใช้งานโอเปค
- *Casting* เป็นการแปลงค่าระหว่างข้อมูลที่ต่างชนิดกัน

5.2.1 ขั้นตอนการ Create Opaque Type

การสร้างโอเปคทำได้ 2 วิธี วิธีแรกโดยการใช้ภาษาซีอีกวิธีคือการสร้างโดยใช้คาต้าเบลดโมดูลเดเวลลอปเปอร์คิท (DataBlade Module Developer Kit)

5.2.1.1 ขั้นตอนการสร้าง Opaque Type โดยภาษาซี

1. สร้างโครงสร้างภายใน เช่นใช้ภาษาซี
2. เขียนซัพพอร์ทฟังก์ชั่น เช่นการเขียนฟังก์ชั่นในภาษาซี เป็นตัวบอกคาต้าเบสเซิร์ฟเวอร์ว่าจะมีการติดต่อกับโอเปคอย่างไรเพราะคาต้าเบสเซิร์ฟเวอร์จะไม่รู้โครงสร้างภายในเพราะถูกปกป้องไว้ภายในโอเปค ตัวอย่างซัพพอร์ทฟังก์ชั่น
    - **input** เปลี่ยนโอเปคจาก External Representation เป็น Internal Representation
    - **output** เปลี่ยนโอเปคจาก Internal Representation เป็น External Representation
    - **receive** เปลี่ยนโอเปคจาก Internal Representation บนไคลเอนท์คอมพิวเตอร์เป็น Internal Representation บนเซิร์ฟเวอร์คอมพิวเตอร์

- **send** เปลี่ยนโอเปคจาก Internal Representation บนเซิร์ฟเวอร์คอมพิวเตอร์เป็น Internal Representation บนไคลเอนท์คอมพิวเตอร์
- **import** เป็นการเปลี่ยนเท็กซ์ไฟล์ (Text File) ไปเป็น Internal Representation
- **export** เป็นการเปลี่ยน Internal Representation ไปเป็นเท็กซ์ไฟล์
- **importbinary** เป็นการเปลี่ยนไบนารี่ไฟล์ (Binary File) ไปเป็น Internal Representation
- **exportbinary** เป็นการเปลี่ยน Internal Representation ไปเป็นไบนารี่ไฟล์
- **assign()** กระบวนการที่ต้องทำบนโอเปคก่อนเก็บลงบนดิสก์
- **destroy()** กระบวนการที่ต้องทำบนโอเปคที่ดาต้าเบสเซิร์ฟเวอร์จะ Remove Row
- **lohandles()** Return List of Smart Large Object ที่ซ่อน (embedded) ไว้ในโอเปค
- **compare()** เปรียบเทียบค่าระหว่าง Opaque Type

3. ทำการรีจิสเตอร์โอเปคในฐานข้อมูลโดยใช้คำสั่ง CREATE OPAQUE TYPE การประกาศโอเปคทำได้ 2 แบบ คือ ดังนี้

```
CREATE OPAQUE TYPE <type NAME> (INTERNALLENGTH=<length>,<Opaque-Type Modifier>)
```

- ❑ *แบบ Fiexed-Length* ขนาดของโครงสร้างข้อมูลภายในถูกกำหนดไว้ตายตัว

```
CREATE OPAQUE TYPE Fixlen_typ (INTERNALLENGTH = 8 , CANNOTHASH)
```

- ❑ *แบบ Vary-Length* ขนาดของโครงสร้างข้อมูลเปลี่ยนแปลงไปตามค่าของโอเปค

```
CREATE OPAQUE TYPE Varylen_typ (INTERNALLENGTH = VARIABLE , MAXLEN=1024)
```

4. ทำการรีจิสเตอร์ซัพพอร์ทฟังก์ชั่นในฐานข้อมูล โดยใช้คำสั่ง CREATE FUNCTION โดยซัพพอร์ทฟังก์ชั่นจะเขียนโดยภาษาซี ทำการคอมไพล์ (Compile) แล้วโอนมาเก็บไว้ในแชร์ไลบรารี่ (Shared Library) บนเซิร์ฟเวอร์ และทำการลงทะเบียน ซึ่งมีรูปแบบดังนี้

```
CREATE FUNCTION <FunctionName> (<ParameterList>)
RETURN          <Ret_type>
EXTERNAL NAME   '<PathName>'
```

LANGUAGE C NOT VARIANT

5. กำหนดสิทธิในการเข้าถึงโอเปค (Access Right Opaque Type) และซัพพอร์ทฟังก์ชั่นโดยใช้คำสั่ง GRANT
6. เขียน SQL-Invoke Function ที่สนับสนุนการใช้งานโอเปคได้แก่
   - Built-In Function
   - Aggreatation Function
   - Arithmetic และ Text Operator Function
   - Comparing Data ได้แก่ Relational Operator และ Condition
7. กำหนด Secondary Access Method

■ **ตัวอย่างการสร้าง Opaque Type**

1. กำหนดโครงสร้างภายในด้วยภาษาซี ถ้าเป็นยูนิกซ์จะคอมไพล์เป็น circle.so แต่ถ้าเป็นวินโดว์เอ็นทีจะเป็น circle.bld

```
typedef struct
{double x;
double y;} point_t;
typedef struct
{point_t center;
double radius;} circle_t;
```

2. เขียนซัพพอร์ทฟังก์ชั่น

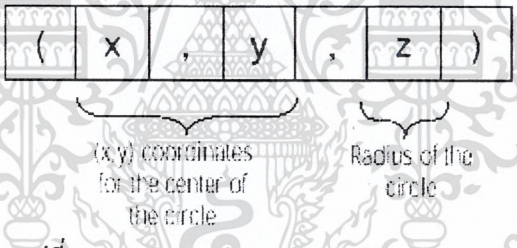
Support Function	Function Signature
input	circle_t * circle_input(extrnl_format)
	mi_lvarchar *extrnl_format;
output	mi_lvarchar * circle_output(intrnl_format)
	circle_t *intrnl_format;
receive	circle_t * circle_receive(client_intrnl_format)
	mi_sendrecv *client_intrnl_format;

```
send                    mi_sendrecv * circle_send(srvr_intrnl_format)
                        circle_t *srvr_intrnl_format;
```

แต่ละฟังก์ชั่นทำหน้าที่ดังนี้

- Circle_input จะทำการแปลง mi_lvarchar ที่เป็น External Representation ให้เป็น Internal Representation ของ Circle_t
- Circle_output จะทำการแปลง Internal Representation ของ Circle_t ให้เป็น mi_lvarchar ที่เป็น External Representation
- Circle_receive จะทำการแปลง Internal Representation ของ ไคลเอนท์ Representation ให้เป็น Internal Representation ของเซิร์ฟเวอร์
- Circle_send จะทำการแปลง Internal Representation ของเซิร์ฟเวอร์ให้เป็น Internal Representation ของไคลเอนท์

เมื่อเราทำการกำหนดอินพุทและเอาท์พุทของระบบหมดแล้ว External Format ของ Circle_จะเป็นดังนี้



รูปที่ 5.5 External Format Opaque Type Circle

3.Register Opaque Type

```
CREATE OPAQUE TYPE circle (INTERNALLENGTH = 24);

CREATE FUNCTION circle_in(txt LVARCHAR) RETURNS circle
EXTERNAL NAME '/usr/lib/circle.so(circle_input)'
LANGUAGE C NOT VARIANT;

CREATE IMPLICIT CAST (LVARCHAR AS circle WITH circle_in);
```

```
CREATE FUNCTION circle_out(cir circle) RETURNS LVARCHAR
EXTERNAL NAME '/usr/lib/circle.so(circle_output)'
LANGUAGE C NOT VARIANT;

CREATE EXPLICIT CAST (circle AS LVARCHAR WITH circle_out);

CREATE FUNCTION circle_rcv(cl_cir SENDRECV) RETURNS circle
EXTERNAL NAME '/usr/lib/circle.so(circle_receive)'
LANGUAGE C NOT VARIANT;

CREATE IMPLICIT CAST (SENDRECV AS circle WITH circle_rcv);

CREATE FUNCTION circle_snd(srv_cir circle) RETURNS SENDRECV
EXTERNAL NAME '/usr/lib/circle.so(circle_send)'
LANGUAGE C NOT VARIANT;

CREATE EXPLICIT CAST (circle AS SENDRECV WITH circle_snd);
```

5.2.1.2 สร้างโอเปคโดยใช้ DataBlade Module Developer

ดาต้าเบลดโมดูลเดเวลลอปเปอร์ (DataBlade Module Developer) เป็นทูลที่ช่วยสร้างโค้ดต่างๆที่จำเป็นต้องสร้างขึ้นสำหรับโอเปคให้ ทำให้เราสามารถสร้างโอเปคขึ้นมาใช้งานได้โดยง่าย ซึ่งสามารถทำได้ดังนี้

ใช้เบลดสมิธ (BladeSmith) ช่วยทำการสร้าง SQL สคริปป์ , ซอร์สโค้ด , Installation Package และ Functional Test ให้และเราสามารถทำการเพิ่มเติมแคสติง , การป้องกันข้อผิดพลาด (Error) , อินเตอร์เฟซ , รูทีน , ข้อมูลชนิดคอลเลคชั่น , ข้อมูลชนิดดิสทิคท์ (Distinct Type) , ข้อมูลชนิดโอเปค (Opaque Type) , Qualified Type และ Row Type หลังจากใช้เบลดสมิธสร้างดาต้าเบลดโมดูลขึ้นมาแล้ว ถ้าเราต้องการแก้ไขเพิ่มเติมสามารถแก้ไขซอร์สโค้ดภาษาซีที่เบลดสมิธสร้างขึ้นมาให้ได้ มีตัวอย่างการสร้างดาต้าเบลดโมดูลอยู่ในภาคผนวก ข.

5.3 เปรียบเทียบการใช้งาน Smart Large Object กับ Opaque Type

ข้อมูลชนิดโอเปคเป็นชนิดข้อมูลที่ปกป้องข้อมูลภายใน (Encapsulation) ไว้ เราจะไม่สามารถเข้าไปแก้ไขข้อมูลที่อยู่ภายในโอเปคได้โดยตรง จะสามารถเข้าไปจัดการกับข้อมูลได้ผ่านทางฟังก์ชั่นที่เรากำหนดขึ้นมาเท่านั้น เพราะโอเปค มีลักษณะของการปกป้องข้อมูลตามแนวคิดเชิงวัตถุ

ข้อเสียของโอเปคคือเราจะต้องทำการเขียนซอร์สโค้ดทั้งหมดขึ้นมาเอง ถึงแม้จะมีทูลอย่างดาต้าเบลดโมดูลเดเวลลอปเปอร์ (DataBlade Module Developer) ขึ้นมาช่วยสร้างโค้ดให้แต่การจะใช้งานโอเปคให้มีประสิทธิภาพนั้นจำเป็นจะต้องมีความรู้ในเรื่องการเขียนโปรแกรมพอสมควร

สำหรับ BLOB CLOB และ โอเปคที่มี Smart Large Object เป็นข้อมูลภายในโอเปคนั้นการใช้งานต้องสร้างฟังก์ชั่นขึ้นมาเหมือนกันโอเปค จะต่างกับชนิดข้อมูลทั่วๆไป ถ้าภายในโอเปคนั้นประกอบไปด้วยชนิดข้อมูลหลายๆชนิดรวมกัน

บทที่ 6

Stored Procedure Language

ในบทนี้จะกล่าวถึง Stored Procedure Language ว่าคืออะไร มีประโยชน์อย่างไร การสร้างและการเรียกใช้งาน

6.1 บทนำ

Stored Procedure Language หรือ SPL เป็นภาษาที่มีความสามารถของภาษา SQL และได้เพิ่มความสามารถในส่วนการควบคุมการทำงาน เช่น การทำลูป การทำทางเลือก เข้าไป โดย SPL เป็นรูทีนที่ฝังอยู่กับระบบจัดการฐานข้อมูล (Database Management System : DBMS) ข้อดีของการมีรูทีนคือสามารถรวบรวมคำสั่งภาษา SQL หลายๆคำสั่งไว้ด้วยการเรียกใช้รูทีนเพียงครั้งเดียว (Encapasulate Multiple SQL Statement) และ ขยายขีดความสามารถของบิวท์อินฟังก์ชั่นที่มีอยู่เดิม (Extend Function Built-In Data Type)

คุณสมบัติที่สำคัญอีกประการของ SPL คือความสามารถในการเข้าไปจัดการแต่ละสมาชิกของคอลเลคชั่นไทป์

6.2 การสร้าง SPL รูทีน

สามารถสร้าง SPL รูทีนได้ 2 ประเภท คือ โพรซีเดอร์ซึ่งไม่มีการส่งคืนค่าข้อมูลและฟังก์ชั่นซึ่งจะต้องมีการกำหนดการส่งคืนค่าข้อมูล การสร้าง SPL รูทีนนั้นสามารถสร้างให้มีชื่อซ้ำกันได้เนื่องจากอินฟอร์มิกซ์สนับสนุนความสามารถในการทำรูทีนโอเวอร์โหลดดิง (Routine Overloading) แต่ๆละรูทีนต้องมีซิกเนเจอร์ต่างกัน โดยเราสามารถกำหนดความแตกต่างของรูทีนได้หลายวิธีดังนี้

- ชนิดของรูทีนต่างกัน
- ชื่อของรูทีนต่างกัน
- จำนวนพารามิเตอร์
- ชนิดของพารามิเตอร์ของรูทีน
- ลำดับของพารามิเตอร์

6.3 กำหนดชื่อ พารามิเตอร์ และชนิดข้อมูลที่จะทำการส่งค่าคืน

เป็นการกำหนดชนิดของรูทีนที่จะทำการสร้างโดยใช้คีย์เวิร์ดเป็นตัวระบุชนิด ถ้ามีการส่งพารามิเตอร์จะต้องทำการกำหนดชื่อและชนิดของพารามิเตอร์ และในกรณีของฟังก์ชั่นจะต้องมีการกำหนดชนิดของข้อมูลที่จะส่งคืนด้วย ดังตัวอย่าง

```
CREATE PROCEDURE New_price(percent REAL)
        .
        .
END PROCEDURE
```

หรือ

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

สำหรับการกำหนดพารามิเตอร์เราไม่สามารถกำหนดพารามิเตอร์เป็น Text หรือ Byte โดยตรงได้ ต้องใช้คีย์เวิร์ด REFERENCES ในการอ้างถึงพารามิเตอร์ที่เป็นข้อมูลชนิด Text หรือ Byte ดังกล่าว ตามตัวอย่าง

```
CREATE PROCEDURE Proc1(lo_text REFERENCES TEXT)
```

ใน SPL จะมีการสร้างบล็อค(Block) การทำงานภายในโดยใช้คีย์เวิร์ด BEGIN และ END ส่วนการกำหนดคอมเมนท์ (Comment) ทำได้โดยการใช้ { } หรือ -- ดังตัวอย่าง

```
CREATE FUNCTION block_demo()            -- SPL Function
RETURNING INT;
DEFINE distance INT;
LET distance = 37;
{ Show how to create implicit statement block}
BEGIN                                   -- Begin implicitstatement block
DEFINE distance INT;
LET distance = 2;
END                                     -- End implicitstatement block
RETURN distance;
```

```
END FUNCTION;
```

6.4 การประกาศค่าตัวแปรและการกำหนดค่าให้ SPL

การประกาศค่าตัวแปรทำได้โดยใช้คีย์เวิร์ด DEFINE ใน SPL รูทีนนั้น สามารถกำหนดตัวแปรขึ้นมาใช้งานได้ 2 ลักษณะ คือ ตัวแปรโกลบอล (Global Variable) และตัวแปรโลคอล (Local Variable) โดยตัวแปรโกลบอลจะสามารถใช้งานร่วมกันได้ในหลายรูทีนเพราะจะทำการจองเนื้อที่บนหน่วยความจำไว้และจะต้องทำการกำหนดค่าดีฟอลท์(Default)ไว้ แต่ตัวแปรโลคอลจะสามารถใช้งานได้เพียงบน SPL นั้นๆ จะถูกเคลียร์ค่าทุกครั้งที่การใช้งานรูทีนนั้นสิ้นสุดลง และไม่สามารถกำหนดค่าของข้อมูลเป็นดีฟอลท์ (Default) ได้ ตัวอย่างการประกาศตัวแปรโลคอลชนิดต่างๆ

```
DEFINE today DATETIME YEAR TO DAY;      -- Built in type
DEFINE b REFERENCES BYTE;               -- Simple large object
DEFINE a COLLECTION;                    -- Collection type
DEFINE c SET                            -- Collection Type
CREATE ROW TYPE zip_t                   -- Named row type
(z_code CHAR(5),                        -- Unnamed row type
z_suffix CHAR(4));
DEFINE manager ROW ( nameVARCHAR(30),
                     department VARCHAR(30),
                     salaryINTEGER
```

การประกาศตัวแปรโกลบอลจะต้องมีคีย์เวิร์ด GLOBAL และการกำหนดค่า DEFAULT เพิ่มเข้ามาดังตัวอย่าง

```
DEFINE GLOBAL gvar INT DEFAULT 2;
```

การกำหนดค่าให้กับตัวแปร 4 ลักษณะดังนี้

1. LET เป็นการกำหนดตัวแปรให้กับค่าใดๆซึ่งจะอยู่ในรูปของ Expression

```
LET a = 5;
LET b = 6; LET c = 10;
LET a,b = 10,c+d;
LET a,b = (SELECT cola,colb FROM tab1 WHERE cola=10);
LET d = func1(x,y);
LET a = ROW ('A Street', 'Nowhere', 'AA',
ROW(NULL, NULL))::address_t
```

2. SELECT...INTO.. เป็นการกำหนดค่าตัวแปรโดยทำการ Select มาจากฐานข้อมูล แล้วทำการเก็บค่าลงไปในตัวแปร
3. EXECUTE FUNCTION...INTO... เป็นการกำหนดตัวแปรโดยใช้ค่าที่ได้จากฟังก์ชั่น เมื่อทำการเอ็กซีคิวฟังก์ชั่นค่าที่รีเทิร์นกลับมาจะเก็บไว้ในตัวแปร
4. CALL...RETURNING... เป็นการกำหนดตัวแปรโดยการคืนค่าจากโพรซีเดอร์

ตัวอย่างการใช้การกำหนดค่าวิธีอื่นๆ

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

EXECUTE FUNCTION read_address('Smith')
INTO p_fname, p_lname, p_add, p_city, p_state, p_zip;

CALL read_address('Smith')
RETURNING p_fname, p_lname, p_add, p_city, p_state, p_zip;
```

6.5 Foreach และ Cursor

ใน SPL มีการใช้เคอร์เซอร์ (Cursor) เนื่องมาจากการกำหนดค่าในตัวแปรตัวหนึ่ง เมื่อมีการคืนค่ากลับขึ้นมาอาจมีค่ามากกว่าหนึ่งค่าถูกส่งมา จึงมีคำสั่ง Foreach มาจัดการกับแต่ละ Row ที่ถูกส่งคืนมา เช่น ในคำสั่ง Select,Execute และ Call ซึ่งในการใช้งาน Foreach และเคอร์เซอร์นี้ สามารถย่อยใช้ได้อีกภายใน Foreach นั้น SPL รูทีนซึ่งรีเทิร์นกลุ่มของ Row จะเรียกว่า เคอร์เซอร์รูทีน (Cursor Routine)

ตัวอย่างการใช้งาน Foreach

```
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;
```

การใช้ Foreach ในการใช้งานคอลเลคชั่นไทป์ทำได้โดย Select เอาคอลัมน์ที่เป็นคอลเลคชั่นไทป์ขึ้นมาเก็บในตัวแปรซึ่งตัวแปรนั้นจะเปรียบเสมือนเทเบิลที่เป็นสมาชิกของคอลเลคชั่นไทป์แล้วใช้ Foreach เข้าไปจัดการกับแต่ละสมาชิกได้

6.6 Flow Control

Flow Control ใน SPL มี 2 ชนิดคือ

- IF-ELIF-ELSE ใช้ในการตรวจเงื่อนไขแล้วแยกทำแต่ละส่วนตามเงื่อนไข ตัวอย่างการใช้ IF-ELIF-ELSE

```
CREATE FUNCTION str_compare( str1 CHAR(20), str2 CHAR(20))
    RETURNING INTEGER;
    DEFINE result INTEGER;
    IF str1 > str2 THEN
        Result = 1;
    ELIF str2 > str1 THEN
        result = -1;
    ELSE
        result = 0;
```

```
    END IF
    RETURN result;
END FUNCTION;
```

- WHILE และ FOR ใช้ในการทำลูป ทำบล็อค (Block) ของสเตตเมนท์โดยจะมีเงื่อนไขในการตรวจสอบเพื่อออกจากลูป ตัวอย่างการใช้งานลูป WHILE และ FOR

```
CREATE PROCEDURE test_rows( num INT )
    DEFINE i INTEGER;
    LET i = 1;
    WHILE i < num
        INSERT INTO table1 (numbers) VALUES (i);
        LET i = i + 1;
    END WHILE;
END PROCEDURE;

FOR i = 1 TO 10
    IF i = 5 THEN
    CONTINUE FOR;
    .
    .
    ELIF i = 8 THEN
    EXIT FOR;
    END IF;
END FOR;
```

การรีเทิร์นค่าของ SPL มีได้ทั้งคืนมาค่าเดียวและคืนมาหลายค่าดังตัวอย่าง

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

CREATE FUNCTION b_date_2( num INTEGER )
RETURNING VARCHAR(30), DATE;
        DEFINE n VARCHAR(30);
        DEFINE b DATE;
        FOREACH cursor1 FOR
            SELECT name, bdate INTO n, b FROM person
                WHERE emp_no > num;
            RETURN n, b WITH RESUME;
        END FOREACH
END FUNCTION;
```

ในกรณีที่การคืนค่ามีมากกว่าหนึ่งเช่นดังตัวอย่างต้องทำการประกาศตัวแปรไว้ และถ้าไม่มี Row Return ค่าของตัวแปรที่ถูกส่งคืนกลับมาจะเป็น Null

6.7 การจัดการกับ Collection Type

การจัดการกับข้อมูลที่เป็นคอลเลคชั่นมีขั้นตอนดังนี้ ประกาศตัวแปรคอลเลคชั่นมประกาศตัวแปรที่จะทำการเก็บสมาชิกแต่ละตัวในคอลเลคชั่น แล้วทำการ Select คอลเลคชั่นจากฐานข้อมูล ดังตัวอย่าง

```
        DEFINE P_coll SET( INTEGER NOT NULL );
        DEFINE P INTEGER;
        SELECT primes INTO P_coll FROM numbers
```

```
WHERE id = 220;
```

เมื่อเราทำการสร้างตัวแปรที่เป็นตัวเก็บค่าที่รีเทิร์นจากคอลเลคชั่น ตัวแปรนั้นจะเสมือนกับเป็นเทเบิลหนึ่ง การจะ Insert ค่าให้กับตัวแปรคอลเลคชั่นนั้นทำได้ดังนี้

```
INSERT INTO TABLE (<Collection Variable>) VALUES (<Value>)
```

```
INSERT INTO TABLE (P_coll)  VALUES (3) ;
```

การ Insert ข้อมูลที่เป็นคอลเลคชั่นนั้นทำได้ 2 วิธีคือใช้เคอร์เซอร์และไม่ใช้เคอร์เซอร์ การ Insert แบบไม่ใช่เคอร์เซอร์จะทำได้เมื่อคอลเลคชั่นที่เราใช้ต้องเป็นเซตหรือมัลติเซตเท่านั้นเพราะลำดับของสมาชิกไม่มีความสำคัญ ดังตัวอย่าง

```
CREATE PROCEDURE new_emp( emp VARCHAR(30), mgr VARCHAR(30) )
        DEFINE r SET(VARCHAR(30) NOT NULL);
        SELECT direct_reports INTO r FROM manager
                WHERE mgr_name = mgr;
        INSERT INTO TABLE (r) VALUES(emp);
        UPDATE manager SET direct_reports = r
                WHERE mgr_name = mgr;
END PROCEDURE;
```

เมื่อเราทำการแก้ไขข้อมูลในตัวแปรคอลเลคชั่นต้องมีการแก้ไขค่าดังกล่าวลงไปบนฐานข้อมูล ดังตัวอย่าง

```
CREATE PROCEDURE shapes()
        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
        UPDATE polygons SET definition = vertexes
            WHERE id = 207;
END PROCEDURE;
```

การจะเข้าไป Select ข้อมูลที่เป็นคอลเลคชั่นต้องเริ่มด้วยการเคอร์เซอร์โดยใช้คีย์เวิร์ด FOREACH

```
FOREACH cursor1 FOR
    SELECT * INTO pnt FROM TABLE(vertexes)
    .
    .
END FOREACH
```

การอัพเดทสมาชิกในคอลเลคชั่นทำได้โดยการดึงข้อมูลมาเก็บในตัวแปรคอลเลคชั่น แล้วทำการประกาศเคอร์เซอร์ขึ้นมา ดังตัวอย่าง

```
DEFINE s SET(INTEGER NOT NULL);
DEFINE n INTEGER;
```

```
SELECT numbers INTO s FROM orders
        WHERE order_num = 10;
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;
        ELSE
                CONTINUE FOREACH;
        END IF;
END FOREACH
```

เมื่อเราต้องการจะอัพเดทข้อมูลคอลเลคชั่นในฐานข้อมูลโดยใช้ตัวแปรคอลเลคชั่นสามารถทำได้ดังตัวอย่าง

```
CREATE PROCEDURE new_report(mgr VARCHAR(30),
        old VARCHAR(30), new VARCHAR(30) )
        DEFINE s SET (VARCHAR(30) NOT NULL);
        DEFINE n VARCHAR(30);
        SELECT direct_reports INTO s FROM manager
                WHERE mgr_name = mgr;
        FOREACH cursor1 FOR
                SELECT * INTO n FROM TABLE(s)
                IF ( n == old ) THEN
                        UPDATE TABLE(s)(x)
                                SET x = new WHERE CURRENT OF cursor1;
                        EXIT FOREACH;
                ELSE
                        CONTINUE FOREACH;
```

```
            END IF;
        END FOREACH
        UPDATE manager SET mgr_name = s
            WHERE mgr_name = mgr;
    END PROCEDURE;
```

6.8 การเรียกใช้รูทีน

สามารถเรียกใช้ได้หลายวิธีดังนี้

- คำสั่ง EXECUTE PROCEDURE หรือ EXECUTE FUNCTION ใน DB-Access
- เรียกจาก SPL รูทีนอื่นหรือจากรูทีนภายนอก ด้วยคำสั่ง CALL
- เรียกผ่าน SQL สเตตเมนท์

ตัวอย่างการเรียกใช้รูทีนแบบต่างๆ

```
EXECUTE PROCEDURE update_orders();
EXECUTE FUNCTION scale_rectangles( 1 07, 1.9 )
        INTO new;
CALL area(rectv.length, rectv.width) RETURNING a;
SELECT increase_by_pct(price, 20) INTO p
        FROM inventory WHERE prod_num = num;
```

บทที่ 7
การออกแบบ Application

ในบทนี้จะกล่าวถึงการนำสิ่งต่างๆที่ได้ศึกษามาทำการสร้างโปรแกรมประยุกต์ซึ่งได้เลือกการเก็บวงจรดิจิตอล(เกท)มาเป็นกรณีศึกษา

7.1 Requirement และ Specification ของ Appliction

เป็นโปรแกรมประยุกต์ที่ทำงานในลักษณะคล้ายกับลอจิกเทรนเนอร์ (Logic Trainer) สนับสนุนการทำงานแบบไคลเอนท์/เซิร์ฟเวอร์ โดยฝั่งไคลเอนท์จะพัฒนาโดยใช้วิชวลเบสิก (Visual Basic) และทางฝั่งดาต้าเบสเซิร์ฟเวอร์ ใช้อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ซึ่งการจัดการกับฐานข้อมูลจะพัฒนาโดยใช้โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ แบ่งการทำงานออกเป็น 3 ส่วน

7.1.1 เซิร์ฟเวอร์

โปรแกรมประยุกต์นี้จะทำการอ่านข้อมูลเพื่อจัดการทุกอย่างจากฝั่งเซิร์ฟเวอร์ โดยเราจะทำการสร้าง SPL Routine ซึ่งเขียนโดยใช้ Informix Stored Procedure Language ซึ่งเป็นภาษาที่มีความสามารถของ SQL และเพิ่มความสามารถในการทำการคอนโทรลการทำงานลงไป (Flow Control) เช่น การทำลูป (Looping) หรือ การทำทางเลือก (Branching) เป็นตัวการในการตรวจสอบและจัดการกับข้อมูลให้กับไคลเอนท์

7.1.2 Data Director

การติดต่อระหว่างโปรแกรมประยุกต์กับฐานข้อมูลจะติดต่อผ่านดาต้าไดเรคเตอร์ (Data Director) ซึ่งจะเป็นตัวจัดการการติดต่อให้ ดาต้าไดเรคเตอร์เป็นทูลที่ทางอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์มีมาให้สำหรับการติดต่อระหว่างฐานข้อมูลกับวิชวลเบสิกโดยเฉพาะดาต้าไดเรคเตอร์จะช่วยให้การติดต่อกับฐานข้อมูลทำได้โดยง่าย ลดการเขียนโค้ดลงไปได้มาก อีกทั้งยังใช้งานง่ายเพียงลากแล้วปล่อยเม้าส์ลงบนคอนโทรลของวิชวลเบสิก (Drag-Drop) ก็สามารถนำข้อมูลจากฐานข้อมูลขึ้นมาแสดงผลได้

7.1.3 ไคลเอนท์

การทำงานของโปรแกรมประยุกต์จะสามารถสร้างวงจรเกทอย่างง่าย เมื่อเริ่มใช้งานจะต้องเลือกการสร้างโปรเจคใหม่หรือเปิดโปรเจคเดิมโดยการใส่ชื่อของโปรเจค นำชื่อของโปรเจคทั้งหมดที่มีขึ้นมาแสดงให้ผู้ใช้เลือกแล้วจะทำการนำข้อมูลจากฐานข้อมูลขึ้นมาแสดงผลตามที่กำหนดไว้ในฟอร์มต่างๆ

สำหรับการสร้างโปรเจคใหม่ เมื่อเราทำการสร้างวงจรเสร็จแล้วจะต้องทำการกำหนดด้วยว่าอะไรคืออินพุทและเอาท์พุทของระบบ ทำการบันทึกรูปวงจรที่ได้สร้างลงบนฐานข้อมูล แล้วจึงทำการเช็คว่าวงจรที่สร้างถูก

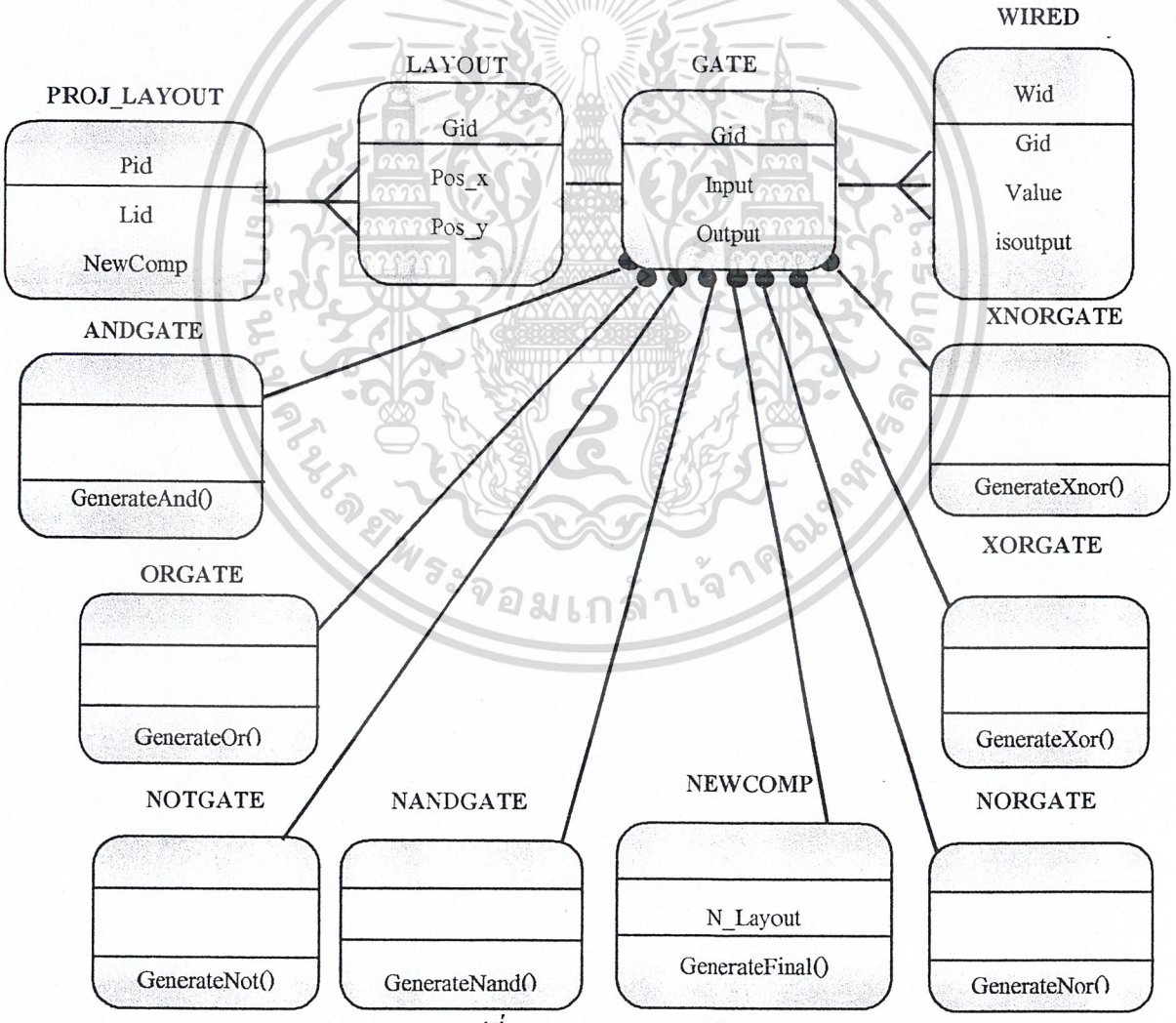
ต้องสามารถทำการจำลองการทำงาน (Simulation) ได้หรือไม่ ถ้าถูกต้องก็จะสามารถทดลองใส่อินพุทเพื่อดูการแสดงเอาท์พุท นอกจากนี้ยังสามารถสร้างคอมโพเนนท์ใหม่จากเกทพื้นฐานที่มีอยู่ได้โดยสามารถทำวงจรที่สร้างขึ้นให้เป็นคอมโพเนนท์ใหม่แล้วสามารถนำไปใช้งานในวงจรอื่นๆได้เหมือนกับคอมโพเนนท์พื้นฐานตัวอื่น

สรุปความสามารถของโปรแกรมประยุกต์

1. สร้างวงจรดิจิตอลอย่างง่ายซึ่งประกอบไปด้วยเกทชนิดต่างๆได้
2. สามารถสร้างคอมโพเนนท์ใหม่จากเกทพื้นฐานที่มีได้
3. สามารถนำคอมโพเนนท์ใหม่ไปใช้งานในวงจรเหมือนกับเกทพื้นฐานอื่นๆได้
4. สามารถทำการ Simulate การทำงานของวงจรได้

7.2 Analysis และ Design

7.2.1 การออกแบบฐานข้อมูล



รูปที่ 7.1 การออกแบบฐานข้อมูล

การจัดเก็บวงจรดิจิตอลที่ประกอบไปด้วยเกทชนิดต่างๆ เป็นตาม อีอาร์ไดอะแกรม (Entity-Relationship : ER Diagram) แบบเคสทูล (Case Tool) ดังภาพ

ซึ่งประกอบไปด้วยออบเจกต์เอนติตี้ (Object Entity) ต่างๆดังนี้

- GATE เป็นออบเจกต์ที่ประกอบไปด้วยสมาชิก ดังนี้
  - GID เป็น Integer ใช้แสดงหมายเลขของเกท
  - INPUT เป็นลิสท์ใช้แสดงลายเส้นวงจรที่ต่อเข้ามาที่อินพุทของเกท
  - OUTPUT เป็นลิสท์ที่ใช้แสดงลายเส้นวงจรที่ต่อออกจากเอาท์พุทของเกท

จากแนวคิดในการออกแบบเชิงวัตถุ เกทแต่ละตัวจะมีคุณสมบัติที่เหมือนกันจะแตกต่างกันที่วิธีในการหาค่าของเอาท์พุทซึ่งเกทแต่ละตัวจะมีวิธีเฉพาะของตัวเอง จากเหตุผลดังกล่าวจึงทำการออกแบบให้เกทแต่ละชนิดทำการรับการถ่ายทอดคุณสมบัติจากคลาสแม่หรือซูเปอร์คลาสในการถ่ายทอดฟิลด์ข้อมูลพื้นฐานซึ่งเหมือนกันมาให้แล้วให้เกทแต่ละชนิดทำการเขียนรูทีน หรือ เมธทอดที่ใช้ในการหาค่าเอาท์พุทเฉพาะแต่ละชนิดของเกทไทป์ ซึ่งการถ่ายทอดคุณสมบัตินี้อินฟอร์มิกซ์สนับสนุนถ่ายทอดคุณสมบัติในกรณีของ Row Type และ Typed Table ทำให้เราต้องทำการสร้าง Row Type GATE ขึ้นมา แล้วทำการสร้าง Table GATE.

- เกทชนิดต่างๆ AND , OR , NOT , NAND , NOR , XOR และ XNOR ต้องเพิ่มส่วนของเมธทอดที่ทำการหาค่าเอาท์พุทเพิ่มขึ้นมา ซึ่งเกทแต่ละชนิดจะต้องทำการถ่ายทอดคุณสมบัติมาจาก Row Type GATE ซึ่งเราต้องทำการสร้าง Row Type ของเกทแต่ละชนิดแล้วทำการสร้างเทเบิลของเกท
- NEWCOMP เป็นออบเจกต์ที่เก็บคอมโพเนนท์ที่สร้างขึ้นมาใหม่ โดยจะมีแอททริบิวเพิ่มจากเกทคือ
  - N_LAYOUT เก็บว่าคอมโพเนนท์นี้โครงสร้างอยู่ที่เลย์เอาท์หมายเลขใด
- WIRED เป็นออบเจกต์ที่ทำการเก็บลายของวงจรที่ทำการเชื่อมต่อแต่ละคอมโพเนนท์เข้าด้วยกัน โดยเราจะมองว่าเส้นลายวงจรหนึ่งเส้นที่ออกจากเอาท์พุทของคอมโพเนนท์หนึ่งไม่ว่าจะเชื่อมต่อไปอีกกี่คอมโพเนนท์ก็คือ 1 ลายวงจรหรือเป็น 1 ออบเจกต์ โดย WIRED ประกอบไปด้วย แอททริบิวต่างๆดังนี้
  - WID เป็น Integer ใช้แสดงหมายเลขของลายวงจร
  - GID เป็น List ของ Integer ใช้แสดงหมายเลขของเกทซึ่งต่ออยู่กับลายวงจรเส้นดังกล่าว โดยลายวงจรเส้นหนึ่งอาจมีลายเกทต่ออยู่ร่วมกัน
  - VALUE เป็น Character เก็บค่าสถานะของลายวงจรเส้นนั้นว่าเป็น Low หรือ High
  - ISOUTPUT เป็นบูลีนที่ทำการเก็บค่าว่าลายวงจรเส้นดังกล่าวเป็นเอาท์พุทของระบบหรือไม่
- LAYOUT เป็นออบเจกต์ที่ทำการเก็บว่าเกทแต่ละตัวที่สร้างขึ้นอยู่ในลายวงจรใด มีตำแหน่งที่อยู่ที่ใดบ้าง ประกอบด้วยแอททริบิวต่างๆดังนี้
  - GID เป็น Integer แสดงหมายเลขของเกท
  - POS_X เป็น Integer แสดงตำแหน่งของเกทตามแนวแกน X
  - POS_Y เป็น Integer แสดงตำแหน่งของเกทตามแนวแกน Y

- LID เป็น Integer แสดงหมายเลขของลายวงจรแต่ละอัน

- PROJ_LAYOUT เป็บออบเจกต์ที่ทำการเก็บว่าแต่ละโปรเจคที่สร้างขึ้น ให้โปรแกรมประยุกต์มาดูว่าโปรเจคหมายเลขดังกล่าวเป็นเลย์เอาท์หมายเลขที่เท่าไหร่ ประกอบไปด้วยแอททริบิวต่างๆดังนี้
  - PID เป็น Character ขนาด 25 ตัวอักษรแสดงหมายเลขของโปรเจค
  - LID เป็น Integer แสดงหมายเลขของเลย์เอาท์

```
CREATE ROW TYPE WIRED
(   Wid int,
    Gid list(int not null),
    Value char(1),
    IsOutput char(1));

CREATE ROW TYPE GATE
(   Gid int,
    Input list(int not null),
    Output list(int not null));

CREATE ROW TYPE AND UNDER GATE;
CREATE ROW TYPE OR UNDER GATE;
CREATE ROW TYPE NOT UNDER GATE;
CREATE ROW TYPE XOR UNDER GATE;
CREATE ROW TYPE XNOR UNDER GATE;
CREATE ROW TYPE NAND UNDER GATE;
CREATE ROW TYPE NOR UNDER GATE;
CREATE ROW TYPE NEW (N_Layout int) UNDER GATE;

CREATE TABLE ANDGATE OF TYPE AND (Primary Key (Gid) );
CREATE TABLE ORGATE OF TYPE OR (Primary Key (Gid) );
CREATE TABLE NOTGATE OF TYPE NOT (Primary Key (Gid) );
CREATE TABLE XORGATE OF TYPE XOR (Primary Key (Gid) );
```

```
CREATE TABLE XNORGATE OF TYPE XOR (Primary Key (Gid) );
CREATE TABLE NANDGATE OF TYPE NAND (Primary Key (Gid) );
CREATE TABLE NORGATE OF TYPE NOR (Primary Key (Gid) );
CREATE TABLE NEWGATE OF TYPE NEW (Primary Key (Gid) );
CREATE TABLE WIRED OF TYPE WIRED (Primary Key (Wid) );

CREATE TABLE PROJ_LAYOUT
(   Pid char(25),
    Lid int,
    NewComp char,
    Primary Key (Pid));

CREATE TABLE LAYOUT
(   Gid int,
    Pos_x int,
    Pos_y int,
    Lid int,
    Primary Key (Gid) );
```

AND

Gid	Input	Output

OR

Gid	Input	Output

NOT

Gid	Input	Output

NAND

Gid	Input	Output

NOR

Gid	Input	Output

XOR

Gid	Input	Output

XNOR

Gid	Input	Output

NEWCOMP

Gid	Input	Output	N_Layout

WIRED

Wid	Gid	Value	Isoutput

LAYOUT

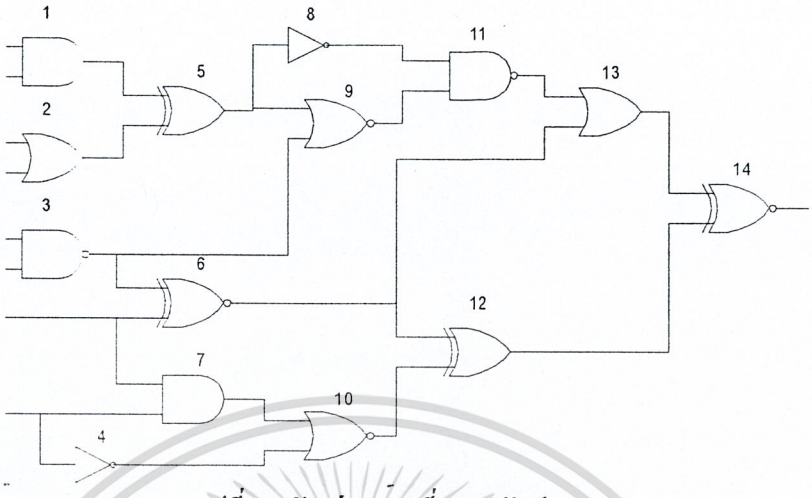
Gid	Pos_x	Pos_y	Lid

PROJ_LAYOUT

Pid	Lid

รูปที่ 7.2 Database Schema

- **ตัวอย่างการจัดเก็บวงจรดิจิตอลลงบนฐานข้อมูล**



รูปที่ 7.3 ตัวอย่างวงจรที่ทำการจัดเก็บ

AND

Gid	Input	Output
1	{1001,1002)	{1009}
7	{1007,1008}	{1014}

ตารางที่ 7.1 การจัดเก็บ AND Gate

OR

Gid	Input	Output
2	{1003,1004}	{1010}
13	{1019,1013}	{1020}

ตารางที่ 7.2 การจัดเก็บ OR Gate

NOT

Gid	Input	Output
4	{1008}	{1015}
8	{1012}	{1016}

ตารางที่ 7.3 การจัดเก็บ NOT Gate

NAND

Gid	Input	Output
3	{1005,1006}	{1011}

11	{1016,1017}	{1019}

ตารางที่ 7.4 การจัดเก็บ NAND Gate

NOR

Gid	Input	Output
9	{1012,1011}	{1017}
10	{1014,1015}	{1018}

ตารางที่ 7.5 การจัดเก็บ NOR Gate

XOR

Gid	Input	Output
5	{1009,1010}	{1012}
12	{1013,1018}	{1021}

ตารางที่ 7.6 การจัดเก็บ XOR Gate

XNOR

Gid	Input	Output
6	{1011,1007}	{1013}
14	{1020,1021}	{1022}

ตารางที่ 7.7 การจัดเก็บ XNOR Gate

WIRED

Wid	Gid	Value	Isoutput
1001	{1}	H	I
1002	{1}	H	I
1003	{2}	L	I
1004	{2}	L	I
1005	{3}	H	I
1006	{3}	L	I
1007	{6,7}	L	I
1008	{4,7}	L	I
1009	{1,5}	H	N
1010	{2,5}	L	N
1011	{3,6,9}	H	N
1012	{5,8,9}	L	N

1013	{6,12,13}	H	N
1014	{7,10}	L	N
1015	{4,10}	H	N
1016	{8,11}	H	N
1017	{9,11}	L	N
1018	{10,12}	L	N
1019	{11,13}	H	N
1020	{13,14}	H	N
1021	{12,14}	L	N
1022	{14}	H	O

ตารางที่ 7.8 การจัดเก็บ WIRED

LAYOUT

Gid	Pos_x	Pos_y	Lid
1	*	*	1
2	*	*	1
3	*	*	1
4	*	*	1
5	*	*	1
6	*	*	1
7	*	*	1
8	*	*	1
9	*	*	1
10	*	*	1
11	*	*	1
12	*	*	1
13	*	*	1
14	*	*	1

ตารางที่ 7.9 การจัดเก็บ LAYOUT

PROJ_LAYOUT

Pid	Lid
Test1	1

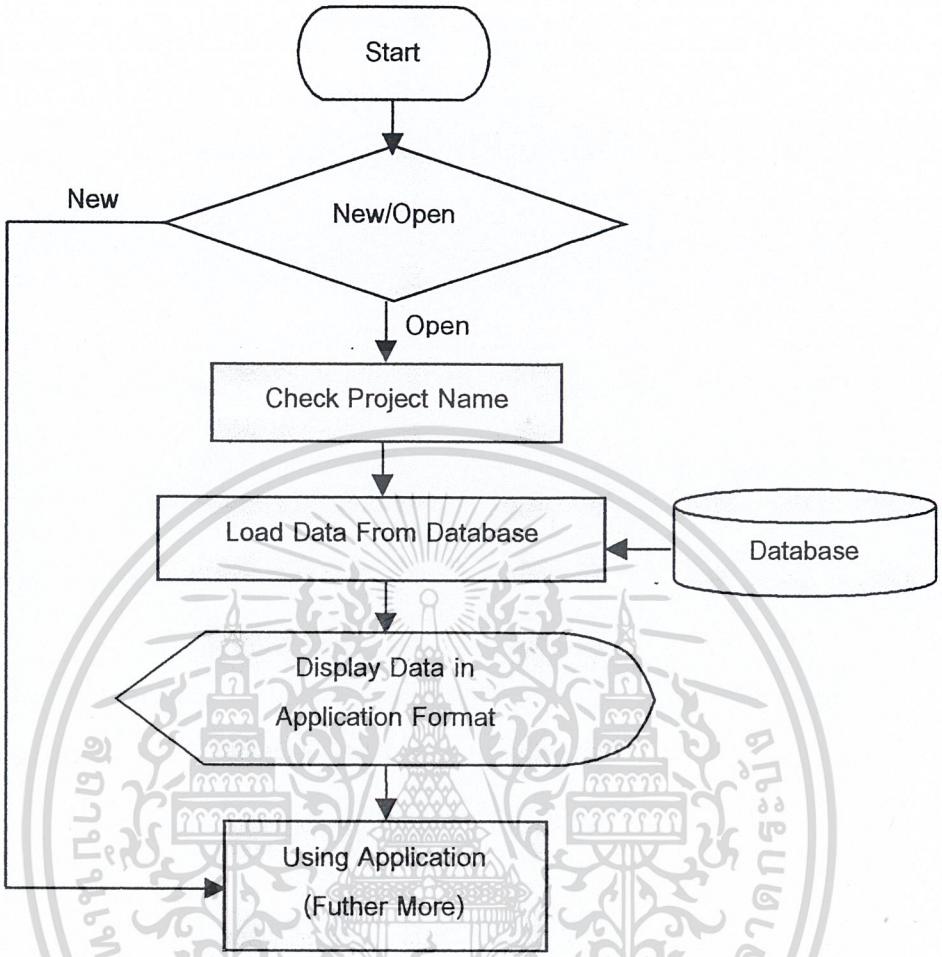
ตารางที่ 7.10 การจัดเก็บ PROJ_LAYOUT

7.2.2 การออกแบบโปรแกรมประยุกต์

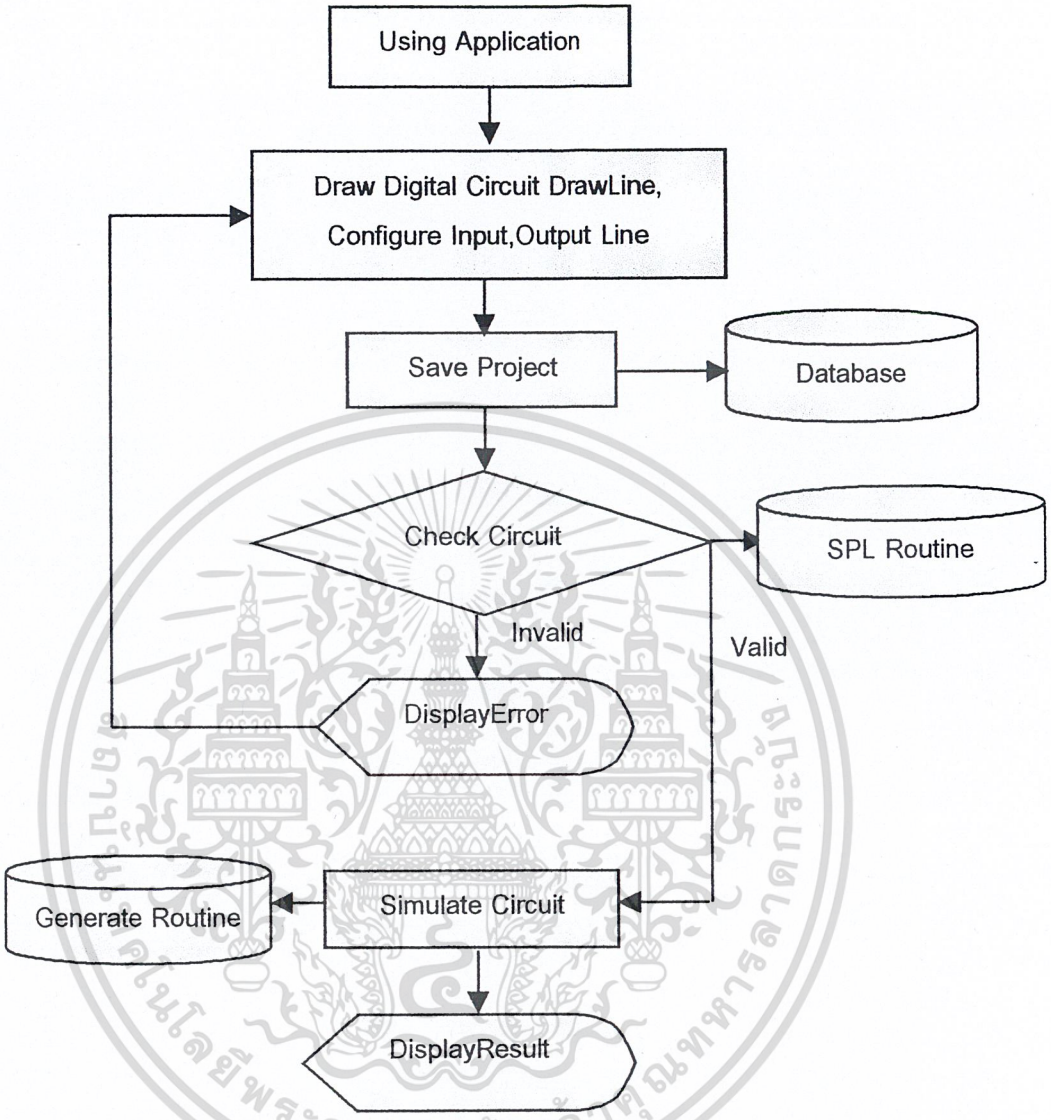
การทำงานของโปรแกรมประยุกต์อธิบายด้วยขั้นตอนการทำงาน ดังรูปที่ 7.4 และ 7.5 โดยการทำงานที่ฝั่งไคลเอนท์จะเน้นที่การสร้างวงจรดิจิตอลแสดงผลข้อมูล มีรูทีนที่ทำการจัดเก็บข้อมูลลงฐานข้อมูล เมื่อต้องการทำการจำลองการทำงานจะทำการเรียกรูทีนที่ฝั่งเซิร์ฟเวอร์โดยส่งคำสั่งไป โดยกำหนดให้การจัดการทั้งหมดอยู่ที่ฝั่งเซิร์ฟเวอร์

เมื่อเริ่มเข้าสู่การทำงานของโปรแกรมประยุกต์จะให้เลือกว่าจะทำการ New หรือ Open โปรเจค โดยถ้าเป็นการเปิดโปรเจคเก่าโปรแกรมประยุกต์จะทำการเรียก SPL รูทีนต่างๆที่มีอยู่บนฝั่งเซิร์ฟเวอร์ไปดึงค่าขึ้นมาแสดงผล ส่วนถ้าเป็นการสร้างโปรเจคใหม่จะต้องทำการวาดวงจรที่ต้องการ โดยเริ่มจากการวาดเกทแล้วลากลายวงจรเชื่อมต่อแต่ละเกท โดยในการลากลายวงจรจะต้องมีการกำหนดเอาท์พุทระบบด้วย นอกจากการใช้งานเกทพื้นฐานที่มีในโปรแกรมแล้วยังสามารถสร้างคอมโพเนนท์ใหม่ขึ้นมาใช่งานและนำไปอ้างถึงในเลย์เอาท์อื่นๆได้อีกด้วย

หลังจากวาดลายวงจรเสร็จจะต้องทำการบันทึกวงจรลงบนฐานข้อมูล โดยจะมีโมดูลของโปรแกรมประยุกต์ทำหน้าที่ในการนำข้อมูลลงไปเก็บบนฐานข้อมูล ต่อจากนั้นเป็นการเช็ควงจรว่าวงจรที่สร้างขึ้นสามารถใช้โปรแกรมประยุกต์คำนวณหาค่าได้หรือไม่ ถ้าได้จะขึ้นแบบฟอร์มให้ใส่อินพุทระบบ เมื่อใส่อินพุทระบบครบก็จะทำการคำนวณหาค่าเอาท์พุทซึ่งเป็นหน้าที่ฝั่งเซิร์ฟเวอร์เมื่อคำนวณได้ก็จะส่งค่าเอาท์พุทคืนมาให้โปรแกรมประยุกต์นำไปแสดงผลต่อไป



รูปที่ 7.4 Flow การทำงานของโปรแกรมประยุกต์(1)



รูปที่ 7.5 Flow การทำงานของโปรแกรมประยุกต์(2)

7.2.3 การออกแบบการจัดการดาต้าเบสเซิร์ฟเวอร์

การทำงานที่ฝั่งดาต้าเบสเซิร์ฟเวอร์นอกจากการจัดเก็บข้อมูลแล้วที่ฝั่งดาต้าเบสเซิร์ฟเวอร์จะมีการฝังรูทีนให้ฝั่งไคลเอนท์มาเรียกใช้ซึ่งรูทีนดังกล่าวเขียนโดยใช้ภาษา Stored Procedure ของอินฟอร์มิกซ์ซึ่งได้กล่าวถึงไปในบทที่ 6 โดยที่ฝั่งเซิร์ฟเวอร์จะประกอบไปด้วยรูทีนต่างๆดังนี้

ชื่อรูทีน	checkInput
หน้าที่	ตรวจสอบว่าวงจรของเรามีอินพุทใดต่อกับเอาท์พุทหรืออินพุทระบบมากกว่า 1 ตัวหรือไม่
ถูกเรียกใช้เมื่อ	จะทำการตรวจสอบว่าวงจรที่สร้างขึ้นถูกต้องตามมาตรฐานของโปรแกรมประยุกต์หรือไม่ โดยฝั่งไคลเอนท์จะทำการเรียกเมื่อทำการเช็ควงจรก่อนที่จะทำการจำลองการทำงาน
รายละเอียด	เริ่มตรวจจากแต่ละขาอินพุทของแต่ละเกทในเลย์เอาท์ ว่ามีเชื่อมต่อกับอินพุทระบบหรือเชื่อมต่อกับเอาท์พุทมากกว่า 1 การเชื่อมต่อหรือไม่ ถ้ามากกว่า 1 จะคืนค่า "N" ถ้ามีแค่ 1 จะคืนค่า "Y"

ชื่อรูทีน	checkloop
หน้าที่	ตรวจสอบว่าวงจรมีการวนลูปหรือไม่
ถูกเรียกใช้เมื่อ	จะทำการตรวจสอบว่าวงจรที่สร้างขึ้นถูกต้องตามมาตรฐานของโปรแกรมประยุกต์หรือไม่ ไคลเอนท์จะทำการเรียกเมื่อทำการเช็ควงจรก่อนที่จะทำการจำลองการทำงาน
รายละเอียด	เริ่มต้นด้วยการใส่เกทเริ่มต้นลงไปในเซตแล้วไล่ใส่เกทที่เชื่อมต่อกับเอาท์พุทลงไปเรื่อยๆจนครบแล้วทำการตรวจสอบว่ามีการวนลูปกลับมาที่เกทตัวเดิมหรอไม่ ถ้าไม่มีการวนลูปจะคืนค่า "Y" แต่ถ้ามีการวนลูปจะคืนค่า "N"

ชื่อรูทีน	clearlayout
หน้าที่	ลบข้อมูลทั้งหมดในเลย์เอาท์
ถูกเรียกใช้เมื่อ	ทำการเคลียร์ข้อมูลในฐานข้อมูลของเลย์เอาท์นั้นๆ ก่อนจะทำการจัดเก็บข้อมูลใหม่ลงไป ไคลเอนท์จะทำการเรียกเมื่อจะทำการ Save
รายละเอียด	ทำการลบข้อมูลในเทเบิล WIRED ตามด้วยเทเบิล GATE และเทเบิล LAYOUT โดยจะลบเฉพาะข้อมูลของเลย์เอาท์นั้น

ชื่อรูทีน	generate<and|or|not|nand|nor|xor|xnor>
หน้าที่	ทำการสร้างค่าเอาท์พุทจากอินพุทที่เป็นพารามิเตอร์ที่ส่งเข้ามา
ถูกเรียกใช้เมื่อ	ถูกเรียกใช้จาก SPL รูทีนอื่นเพื่อทำการหาค่าเอาท์พุทของแต่ละเกท
รายละเอียด	ทำการตรวจสอบค่าอินพุทและคำนวณตามหลักของเกทแต่ละชนิดแล้วคืนค่า "H" หรือ "L"

ชื่อรูทีน	genaratefinal
หน้าที่	ทำการคำนวณค่าของทั้งเลย์เอาท์
ถูกเรียกใช้เมื่อ	ฝั่งไคลเอนท์จะทำการจำลองการทำงานของวงจรเพื่อดูค่าเอาท์พุท

รายละเอียด	วนลูปทำการเรียก SPL getgen<and|or|not|nand|nor|xor|xnor> ของแต่ละเกทและจะหยุดลูปเมื่อทำการหาเอาท์พุทครบทุกเกท

ชื่อรูทีน	getelement
หน้าที่	ทำการดึงข้อมูลแต่ละสมาชิกในคอลเลคชั่นไทป์ออกมา
ถูกเรียกใช้เมื่อ	เมื่อ SPL รูทีนอื่นต้องการสมาชิกลำดับที่ใดๆในคอลเลคชั่นไทป์ไปใช้งาน
รายละเอียด	ทำการวนลูปจนถึงลูปลำดับที่ต้องการ ก็จะคืนค่าสมาชิกนั้นไปใช้

ชื่อรูทีน	getgen<and|or|not|nand|nor|xor|xnor>
หน้าที่	ทำการดึงข้อมูลจากเกทที่มาคำนวณหาค่าเอาท์พุท
ถูกเรียกใช้เมื่อ	จะถูก SPL รูทีน generatefinal เรียกใช้
รายละเอียด	ทำการเรียก SPL รูทีน getgid<and|or|not|nand|nor|xor|xnor> เพื่อดูว่ามีเกทใดพร้อมที่จะทำการคำนวณหาค่าเอาท์พุทแล้วบ้าง โดยจะดูว่ามีอินพุทครบทั้งหมดหรือยัง ถ้าครบจะทำการดึงค่าอินพุทดังกล่าวมาให้กับ SPL รูทีน generate<and|or|not|nand|nor|xor|xnor> เพื่อทำการสร้างเอาท์พุทโดยเอาท์พุทที่ได้จะนำไปใส่กับเอาท์พุทของเกทเดิมแล้วทำการคืนค่าจำนวนเกทที่คำนวณหาเอาท์พุทเสร็จแล้ว

ชื่อรูทีน	getgid<and|or|not|nand|nor|xor|xnor>
หน้าที่	ตรวจสอบว่าเกทใดสามารถหาค่าเอาท์พุทได้บ้าง
ถูกเรียกใช้เมื่อ	ถูกรูทีน getgen<and|or|not|nand|nor|xor|xnor> เรียกเพื่อให้ทราบว่ามีเกทใดพร้อมหาเอาท์พุทบ้าง
รายละเอียด	ทำการตรวจสอบว่าเกทใดมีอินพุทครบและยังไม่มีเอาท์พุท

ชื่อรูทีน	getgidorder
หน้าที่	หาค่า Gid จากลำดับของเกท
ถูกเรียกใช้เมื่อ	ถูกเรียกใช้จากรูทีนอื่นเมื่อต้องการรู้ Gid ของเกทจากลำดับของเกท
รายละเอียด	ทำการเรียงลำดับของเกทที่อยู่ในเลย์เอาท์แล้วทำการวนลูปถึงลำดับของเกท คืนค่า Gid กลับไป

ชื่อรูทีน	getnewwid
หน้าที่	หาค่า Wid ที่ถูกกำหนดค่าเริ่มต้นขึ้นมาใหม่
ถูกเรียกใช้เมื่อ	ฝั่งไคลเอนท์จะเรียกใช้ขณะที่จะทำการบันทึกวงจร

รายละเอียด	จะดึงข้อมูลเฉพาะ Wid ที่เพิ่งจะกำหนดค่าเริ่มต้นขึ้นมาใหม่ ทำการเรียงลำดับและคืนค่า Wid ที่มีลำดับตามที่ต้องการกลับไป

ชื่อรูทีน	getnumbergate
หน้าที่	คำนวณหาค่าจำนวนเกทที่อยู่ในเลย์เอาท์
ถูกเรียกใช้เมื่อ	ไคลเอนท์จะเรียกใช้เพื่อนำค่าไปกำหนดค่าเริ่มต้นใหม่ให้กับตัวแปรในโปรแกรมประยุกต์
รายละเอียด	ทำการนับจำนวน Gid ที่อยู่ในเลย์เอาท์นั้นๆ

ชื่อรูทีน	getnumberwired
หน้าที่	ทำการหาค่าจำนวนลายการเชื่อมต่อที่อยู่ในเลย์เอาท์นั้น
ถูกเรียกใช้เมื่อ	ไคลเอนท์จะเรียกใช้เพื่อนำค่าไปกำหนดค่าเริ่มต้นใหม่ให้กับตัวแปรในโปรแกรมประยุกต์
รายละเอียด	ทำการเรียก SPL getwid แล้วนำมานับว่ามี Wid กี่ตัวแล้วคืนค่า

ชื่อรูทีน	getordergid
หน้าที่	ทำการหาลำดับของ Gid ในเลย์เอาท์
ถูกเรียกใช้เมื่อ	SPL รูทีนอื่นจะเรียกใช้เมื่อต้องการรู้ลำดับของ Gid
รายละเอียด	ทำการวนลูปตรวจสอบว่า Gid ในลำดับที่จัดเรียงตรงกับ Gid ที่เป็นพารามิเตอร์หรือไม่ ถ้าเท่าก็จะทำการคืนค่าลำดับออกไป

ชื่อรูทีน	getorderwid
หน้าที่	ทำการหาลำดับของ Wid ในเลย์เอาท์
ถูกเรียกใช้เมื่อ	SPL รูทีนอื่นจะเรียกใช้เมื่อต้องการรู้ลำดับของ Wid
รายละเอียด	ทำการวนลูปตรวจสอบว่า Wid ในลำดับที่จัดเรียงตรงกับ Wid ที่เป็นพารามิเตอร์หรือไม่ ถ้าเท่าก็จะทำการคืนค่าลำดับออกไป

ชื่อรูทีน	getwid
หน้าที่	ทำการหา Wid ทั้งหมดของเลย์เอาท์
ถูกเรียกใช้เมื่อ	SPL รูทีนอื่นจะเรียกใช้เมื่อต้องการรู้ว่ามี Wid ใดบ้างอยู่ในเลย์เอาท์
รายละเอียด	ทำการหา Gid ที่อยู่ในเลย์เอาท์นั้นๆก่อนแล้วจึงนำมาหาอินพุทและเอาท์พุทในแต่ละเกทแล้วคืนค่า Wid ที่อยู่ในอินพุทและเอาท์พุทออกไป

ชื่อรูทีน	getwidinput
หน้าที่	หา Wid ที่เป็นอินพุทของระบบ
ถูกเรียกใช้เมื่อ	ไคลเอนท์จะทำการเรียกเพื่อให้ทราบว่า wired ใดเป็นอินพุทของระบบ
รายละเอียด	ทำการหา Wid ทั้งหมดของระบบแล้วตรวจสอบว่า isoutput เป็น "I" หรือไม่ ถ้าเป็นก็จะคืนค่า wired นั้นออกไป

ชื่อรูทีน	getwidoutput
หน้าที่	หา Wid ที่เป็นเอาท์พุทของระบบ
ถูกเรียกใช้เมื่อ	ไคลเอนท์จะทำการเรียกเพื่อให้ทราบว่า wired ใดเป็นเอาท์พุทของระบบ
รายละเอียด	ทำการหา Wid ทั้งหมดของระบบแล้วตรวจสอบว่า isoutput เป็น "O" หรือไม่ ถ้าเป็นก็จะคืนค่า wired นั้นออกไป

ชื่อรูทีน	getwidorder
หน้าที่	หา Wid ตามลำดับของเลย์เอาท์
ถูกเรียกใช้เมื่อ	SPL รูทีนอื่นจะทำการเรียกเมื่อต้องการทราบลำดับ Wid จากลำดับ Wid ในเลย์เอาท์
รายละเอียด	ทำการเรียก getwid หา Wid ทั้งหมด แล้ววนลูปหาถึงลูปลำดับที่เท่ากับลำดับของ Wid แล้วคืนค่า Wid ออกไป

ชื่อรูทีน	getwidvalue
หน้าที่	ทำการหาค่า Value จากลำดับของ Wid
ถูกเรียกใช้เมื่อ	โปรแกรมประยุกต์ต้องการแสดง Value ในเลย์เอาท์
รายละเอียด	หาค่า Wid จริงจาก getwidorder ก่อนแล้วจึง Select ค่า Value ออกมา

ชื่อรูทีน	initlayout
หน้าที่	ทำการ Insert Gid เข้าไปในฐานข้อมูล
ถูกเรียกใช้เมื่อ	ไคลเอนท์ทำการบันทึกข้อมูล
รายละเอียด	ทำการหา Gid ที่มากที่สุดแล้วนำค่าดังกล่าวบวก 1 แล้วทำการวนลูป Insert เพิ่มเข้าไปจนครบจำนวนที่เราต้องการจะ Init แต่ถ้าไม่มี Gid ในเลย์เอาท์เลยก็จะทำการเริ่มสร้างตั้งแต่ 1

ชื่อรูทีน	initvalue
หน้าที่	ทำการเคลียร์ Value ใน Wired ทุกเส้นที่อยู่ในเลย์เอาท์

ถูกเรียกใช้เมื่อ ก่อนที่ไคลเอนท์จะเริ่มต้นทำการจำลองการทำงานของวงจร

รายละเอียด ทำการหาค่า Wid ของทั้งเลย์เอาท์นั้นแล้วทำการอัพเดทให้ Value เป็น null

ชื่อรูทีน initwired

หน้าที่ ทำการ Insert Wired

ถูกเรียกใช้เมื่อ ไคลเอนท์ทำการบันทึกข้อมูล

รายละเอียด ทำการหาค่า Wid ที่มากที่สุดแล้วจึงนำค่าดังกล่าวบวก 1 แล้วทำการวนลูป Insert เพิ่มเข้าไปจนครบจำนวนที่เราต้องการจะ Init แต่ถ้าไม่มี Wid ในเลย์เอาท์เลยก็จะทำการเริ่มสร้างตั้งแต่ 1

ชื่อรูทีน loadgate

หน้าที่ ดึงค่าของจำนวนอินพุท เอาท์พุทและชนิดของเกทขึ้นมา

ถูกเรียกใช้เมื่อ เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่

รายละเอียด ทำการ Select ข้อมูลจากเทเบิล Gate ต่างๆอยู่ที่เทเบิลใดก็จะดึงค่าชนิดของเทเบิลนั้นมาด้วย พร้อมกับจำนวนอินพุทและจำนวนเอาท์พุท

ชื่อรูทีน loadgatelist

หน้าที่ ดึงข้อมูลของเกทที่เชื่อมต่อที่อยู่ใน Wired

ถูกเรียกใช้เมื่อ เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่

รายละเอียด ทำการดึงข้อมูลของ Wid ที่ต้องการโดยใช้รูทีน getwidorder ก่อนแล้วจึงดึงข้อมูล Gid ซึ่งอยู่ในคอลเลคชั่นไทป์ออกมาแล้ววนลูปจะทำการลืนค่าสมาชิกแต่ละตัวในคอลเลคชั่นไทป์กลับไป

ชื่อรูทีน loadinput

หน้าที่ ทำการดึงข้อมูล Wired ที่ต่ออยู่กับ Input ต่างๆของเกท

ถูกเรียกใช้เมื่อ เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่

รายละเอียด เรียกรูทีน getgidorder เพื่อให้ได้เกทที่ต้องการแล้วเข้าไปดึงข้อมูล Input ออกมาจากเกทนั้นแล้วทำการวนลูปเพื่อคืนค่าสมาชิกในคอลเลคชั่นไทป์นั้นออกไป

ชื่อรูทีน loadoutput

หน้าที่ ทำการดึงข้อมูล Wired ที่ต่ออยู่กับ Output ต่างๆของเกท

ถูกเรียกใช้เมื่อ เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่

รายละเอียด	เรียกรูทีน getgidorder เพื่อให้ได้เกทที่ต้องการแล้วเข้าไปดึงข้อมูล output ออกมาจากเกทนั้นแล้วทำการวนลูปเพื่อคืนค่าสมาชิกในคอลเลคชั่นไทป์นั้นออกไป

ชื่อรูทีน	loadpos
หน้าที่	ดึงตำแหน่งของเกทที่แสดงผลบนหน้าจอของโปรแกรมประยุกต์จากฐานข้อมูล
ถูกเรียกใช้เมื่อ	เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่
รายละเอียด	ทำการ Select ตำแหน่งหน้าจอทั้งหมดของทุกเกทที่เกี่ยวข้องกับเลย์เอาท์นั้น

ชื่อรูทีน	loadwired
หน้าที่	ทำการคืนค่าชนิดของ Wired และจำนวนของการเชื่อมต่อกับเกทต่างๆ
ถูกเรียกใช้เมื่อ	เมื่อไคลเอนท์ต้องการเปิดโปรเจคเดิมที่มีอยู่
รายละเอียด	ทำการเรียก getwid เพื่อให้ได้ Wid ออกมาทั้งหมด แล้วทำการSelect เอาชนิดของ Wired และจำนวนของเกทที่อยู่ใน Gid ออกมา

ชื่อรูทีน	insertvalue
หน้าที่	ใส่ค่า Value ให้กับอินพุทระบบ
ถูกเรียกใช้เมื่อ	โปรแกรมประยุกต์ต้องการ ใส่ค่า Value ให้กับอินพุทระบบก่อนที่จะทำการ Simulate
รายละเอียด	หาค่า Wid จริงจาก getwidorder แล้วจึง Select ค่า Value ออกมา

7.3 Stored Procedure Language

Informix Stored Procedure Language เป็นภาษาที่มีความสามารถของ SQL (SQL-Extension) และเพิ่มความสามารถในการทำการคอนโทรลการทำงานลงไป (Flow Control) เช่น การทำลูป (Looping) หรือ การทำทางเลือก (Branching) โดยการใช้งานคาต้าเบสเซิร์ฟเวอร์เราจะทำการสร้าง SPL รูทีนต่างๆฝังไว้ที่เซิร์ฟเวอร์ รายละเอียดของ SPL ได้กล่าวแล้วไปในบทที่ 6

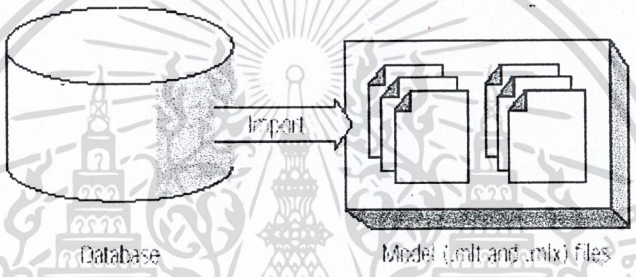
7.4 Data Director

การติดต่อระหว่างโปรแกรมประยุกต์กับฐานข้อมูลจะติดต่อผ่านคาต้าไดเรคเตอร์ (Data Director) คาต้าไดเรคเตอร์เป็นทูลที่ทางอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์มีมาให้สำหรับการติดต่อระหว่างฐานข้อมูลกับวิชวลเบสิกโดยเฉพาะ การใช้งานคาต้าไดเร็คเตอร์จะมี 2 วิธี คือ การใช้งานแบบกราฟิกซึ่งเมื่อทำการติดตั้งคาต้าไดเร็คเตอร์ตัวคาต้าไดเร็คเตอร์ คอมโพเนนท์ต่างๆจะถูกนำเข้าไปไว้ในวิชวลเบสิกทุกครั้งที่เรียกใช้จะปรากฏคาต้าไดเร็คเตอร์

ขึ้นมาเสมอซึ่งจะช่วยให้เราสารถนำข้อมูลขึ้นมาแสดงบนฟอร์มได้ง่ายโดยไม่ต้องมีการเขียนโค้ดใดๆเลย อีกวิธีเป็นการเรียกใช้ API ของคาต้าไดเร็คเตอร์ในการติดต่อกับฐานข้อมูล วิธีนี้จะต้องใช้กับการโปรแกรมมิ่งเท่านั้น

การทำงานของคาต้าไดเร็คเตอร์จะทำการอิมพอร์ตโมเดลของฐานข้อมูลเข้ามาซึ่งจะทำให้โปรแกรมประยุกต์รู้โครงสร้างภายในฐานข้อมูล โดยในการสร้างโปรแกรมประยุกต์เพื่อทำการติดต่อฐานข้อมูลหนึ่งจะต้องใช้งานคาต้าไดเร็คเตอร์ออบเจกต์ (Data Director Object) หรือ ดีดีโอ (DDO) ซึ่งเราจะต้องทำการสร้างออบเจกต์ oEngine ให้กับทุกๆโปรแกรมประยุกต์ oEngine จะทำหน้าที่เป็นรูท (Root) ให้กับทุกๆดีดีโอ แล้วทำการสร้างคาต้ากรุ๊ป (Datagroup) ซึ่งเป็นตัวที่จัดการรวมข้อมูลไว้ด้วยกัน โดยทำการสร้างออบเจกต์ oDatagroup ขึ้นมา

การใช้งานข้อมูลในฐานข้อมูลหากเราต้องการทำการคิวรีข้อมูลต้องทำการสร้างเวอร์ชวลเทเบิล (Virtual Table) เพื่อทำการเก็บผลที่ได้จากการคิวรีเราสามารถใส่คำสั่ง SQL ได้โดยตรง โดยการสร้างเวอร์ชวลเทเบิลจะต้องทำการสร้างออบเจกต์ oTable



รูปที่ 7.6 การอิมพอร์ตโมเดล

บทที่ 8
โปรแกรมประยุกต์ลอจิกเทรนเนอร์

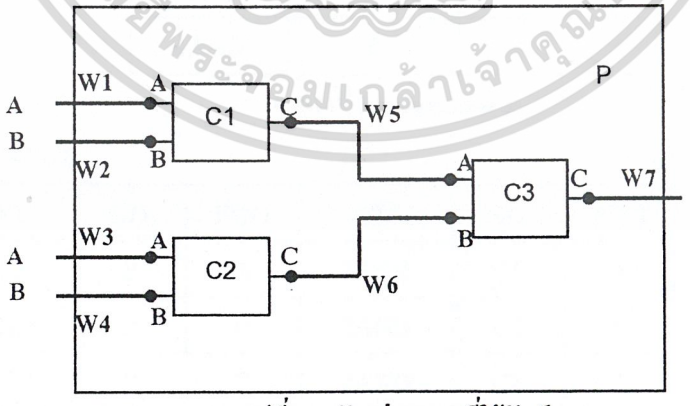
ในบทนี้จะกล่าวถึงการจัดเก็บวงจรดิจิตอลโดยใช้โมเดลฐานข้อมูลเชิงสัมพันธ์เปรียบเทียบกับการใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์ รายละเอียดของตัวโปรแกรมประยุกต์ที่เราได้ทำการสร้าง และวิธีการใช้งานโปรแกรมประยุกต์

8.1 การพัฒนาการจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงสัมพันธ์

จากการที่ฐานข้อมูลเชิงสัมพันธ์ไม่สนับสนุนการจัดการกับคอมเพล็กซ์ออบเจกต์ ดังนั้นการจัดเก็บออบเจกต์หนึ่งจะต้องแตกออกเป็นหลายๆ Row กระจายอยู่ในหลายรีเลชั่น การจัดเก็บในวิธีดังกล่าวจะมีข้อเสีย คือ

1. ยากต่อการทำความเข้าใจของผู้ใช้ จากการที่มีการกระจายไปในหลายรีเลชั่นทำให้ข้อมูลอยู่ในรูปที่ทำความเข้าใจได้ยาก ข้อมูลที่แสดงอาจไม่สื่อความหมายให้เข้าใจได้หากดูจากรีเลชั่นเดียว
2. การทำโอเปอร์เรชั่นใดๆกับคอมเพล็กซ์ออบเจกต์ไม่สามารถทำได้ในครั้งเดียวต้องมีการใช้หลายคำสั่ง เพราะได้มีการกระจายไปหลายรีเลชั่นทำให้การนำข้อมูลมารวมกันเพื่อการทำโอเปอร์เรชั่นยุ่งยาก
3. มีความซ้ำซ้อนในการเก็บข้อมูล เนื่องจากการแตกเป็นหลายส่วนเมื่อมีการจะรวมข้อมูลต้องมีการ Join ระหว่างเทเบิลซึ่งในการจะลดการ Join เทเบิลก็ต้องทำการเพิ่มข้อมูลที่ซ้ำซ้อนบางส่วนลงในเทเบิล เมื่อมีข้อมูลที่ซ้ำซ้อนกันความยุ่งยากในการควบคุมความถูกต้องของฐานข้อมูลจะเพิ่มขึ้น
4. ไม่สนับสนุนการ Reuse คอมโพเนนท์

8.1.1 ตัวอย่างการจัดเก็บโดยใช้ Relational



รูปที่ 8.1 ตัวอย่างวงจรที่ใช้จัดเก็บ

W7	2AND	C3	C	4AND	P	E	4AND	L

ตารางที่ 8.4 WiredInstance

จากตัวอย่างการจัดเก็บวงจรในแบบฐานข้อมูลเชิงสัมพันธ์จะทำการแบ่งจัดเก็บออบเจกต์ออกเป็น 4 รีเลชั่นได้แก่

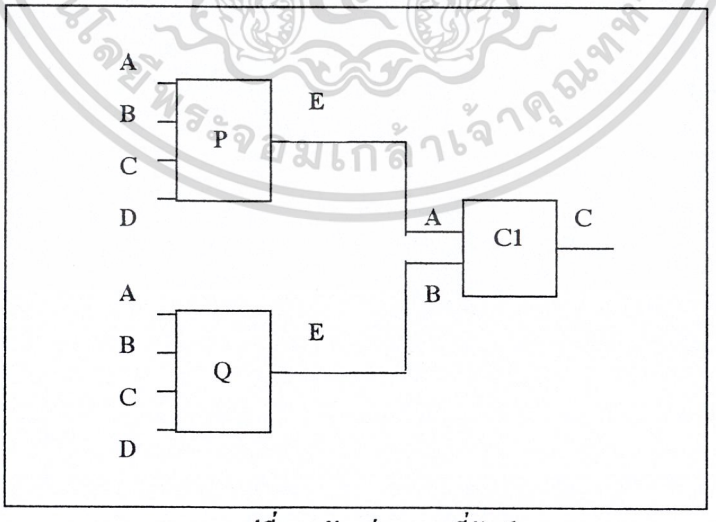
1. GATE_TYPE เป็นรีเลชั่นที่ทำการเก็บชื่อของออบเจกต์และรายละเอียดของออบเจกต์นั้นซึ่งในที่นี้เก็บวิธีการหาค่าเอาท์พุทของวงจร ซึ่งการเก็บเป็นรีเลชั่นจะไม่สามารถทำโอเปอร์เรชั่นกับออบเจกต์ได้โดยตรง
2. PIN_TYPE เป็นการเก็บชนิดของขา ออบเจกต์หนึ่งออบเจกต์จะประกอบไปด้วยหลายขาซึ่งเราต้องทำการบอกไว้ว่าขาดังกล่าวเป็นอินพุทหรือเอาท์พุท
3. GATE_INSTANCE เป็นการเก็บข้อมูลของตัวออบเจกต์ที่มีอยู่จริงในวงจร โดยจะทำการเก็บด้วยว่าเป็นชนิดไหน ชื่ออะไร และอยู่ภายในออบเจกต์อะไร
4. WIRED_TYPE เป็นการเก็บการเชื่อมต่อระหว่างออบเจกต์ 2 ออบเจกต์โดยจะเก็บด้วยว่าเชื่อมกับขาไหน ออบเจกต์ที่ต่อกันเป็นชนิดอะไรและการเชื่อมต่อนั้นอยู่ในออบเจกต์อะไร

8.1.2 ปัญหาในการจัดเก็บของฐานข้อมูลเชิงสัมพันธ์

จากลักษณะการจัดเก็บข้อมูลดังกล่าวจะมีปัญหาเกิดขึ้นหลายประการซึ่งปัญหาหลักๆได้แก่

- ปัญหาการรียูส (Reuse) คอมโพเนนท์

จากการจัดเก็บข้อมูลตามตัวอย่างหากเราต้องการทำการออกแบบวงจรที่มีลักษณะดังรูป



รูปที่ 8.2 ตัวอย่างวงจรที่จัดเก็บ

หากเราต้องการสร้างวงจรที่ประกอบไปด้วย 4AND 2 ตัวในวงจรคือ P และ Q เราจะต้องทำการสร้าง Instance ของทั้ง GATE_INSTANCE และ WIRED_INSTANCE ทั้งหมดของทั้ง P และ Q ซึ่งในความเป็นจริงไม่น่าที่จะต้องทำการสร้างเพราะทั้งสองคอมโพเนนท์มีลักษณะเหมือนกันทุกประการ

- ปัญหาความซับซ้อนของข้อมูล

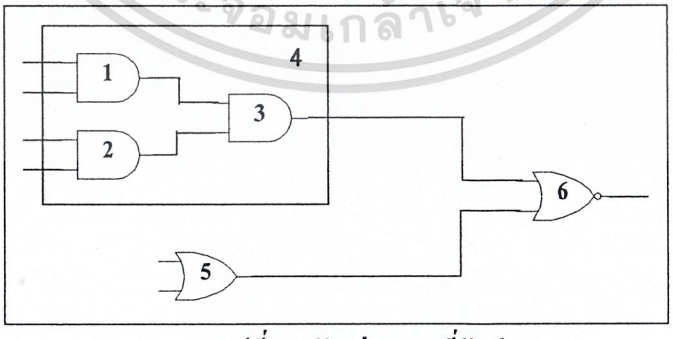
ความซับซ้อนของข้อมูลเกิดในหลายรีเลชั่น เช่น ใน WIRED_INSTANCE จากการที่เก็บความสัมพันธ์ของการเชื่อมต่อระหว่างเกทแค่ 2 เกทในหนึ่งทัปเปิล ทำให้หากมีเอาท์พุทของเกทใดต่อกับอินพุทของเกทหลายๆเกทเราต้องเก็บ WIRED_INSTANCE เป็นจำนวนเท่ากับการเชื่อมต่อทั้งหมด ดังนั้นเมื่อมีการอัพเดทข้อมูลแอททริบิว "VALUE" จะต้องทำมัลติเพิลอัพเดท ทำให้ต้องมีการทำการควบคุมความถูกต้องของฐานข้อมูลเพิ่มขึ้น

8.2 การจัดเก็บวงจรดิจิตอลแบบฐานข้อมูลเชิงวัตถุสัมพันธ์

จากปัญหาต่างๆ ในการจัดเก็บข้อมูลโดยใช้โมเดลฐานข้อมูลเชิงสัมพันธ์แสดงให้เห็นว่าการจัดเก็บวงจรดิจิตอลไม่เหมาะสมกับการใช้ฐานข้อมูลเชิงสัมพันธ์ในโครงงานได้นำเอาโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์มาแก้ปัญหาของฐานข้อมูลเชิงสัมพันธ์ในการใช้งานกับคอมเพล็กซ์ออบเจกต์ซึ่งทำให้

1. มองข้อมูลที่จัดเก็บเป็น Object ไม่ต้องแตกออกเป็นหลายๆ Row อีก ง่ายต่อการทำความเข้าใจของผู้ใช้
2. การทำ Operation ใดๆกับ Complex Object ทำได้ในคำสั่งเดียวโดยการสร้าง Method ให้กับ Object นั้น
3. สนับสนุนการ Reuse คอมโพเนนท์ตามความสามารถของแนวคิดเชิงวัตถุที่เพิ่มเข้ามา ซึ่งทำให้เราสามารถนำคอมโพเนนทไปใช้งานใหม่ได้ง่ายไม่ต้องทำอะไรเพิ่มเติมเหมือนในการใช้งานฐานข้อมูลเชิงวัตถุ
4. ลดความซ้ำซ้อนในการเก็บข้อมูลทำให้การดูแลความถูกต้องฐานข้อมูลทำได้ง่ายกว่า

8.2.1 ตัวอย่างการจัดเก็บโดยใช้ Object-Relational



รูปที่ 8.3 ตัวอย่างวงจรที่จัดเก็บ

PROJ_LAYOUT

PID	LID
4AND	1
CIRCUIT	2

ตารางที่ 8.5 PROJ_LAYOUT

LAYOUT

GID	POS_X	POS_Y	LID
1	*	*	1
2	*	*	1
3	*	*	1
4	*	*	2
5	*	*	2
6	*	*	2

ตารางที่ 8.6 LAYOUT

ANDGATE

GID	INPUT	OUTPUT
1	{1001,1002}	{1005}
2	{1003,1004}	{1006}
3	{1005,1006}	{1007}

ตารางที่ 8.7 ANDGATE

NEWCOMP

GID	INPUT	OUTPUT	N_LAYOUT
4	{1008,1009,1010,1011}	{1014}	1

ตารางที่ 8.8 NEWCOMP

ORGATE

GID	INPUT	OUTPUT
5	{1012,1013}	{1015}

ตารางที่ 8.9 ORGATE

NORGATE

GID	INPUT	OUTPUT
6	{1014,1015}	{1016}

ตารางที่ &10 NORGATE

WIRED

WID	GID	VALUE	ISOUTPUT
1001	{1}	H	I
1002	{1}	H	I
1003	{2}	H	I
1004	{2}	L	I
1005	{1}	H	N
1006	{2}	L	N
1007	{3}	L	O
1008	{4}	H	I
1009	{4}	H	I
1010	{4}	H	I
1011	{4}	L	I
1012	{5}	H	I
1013	{5}	L	I
1014	{3,6}	L	N
1015	{5,6}	H	N
1016	{6}	L	O

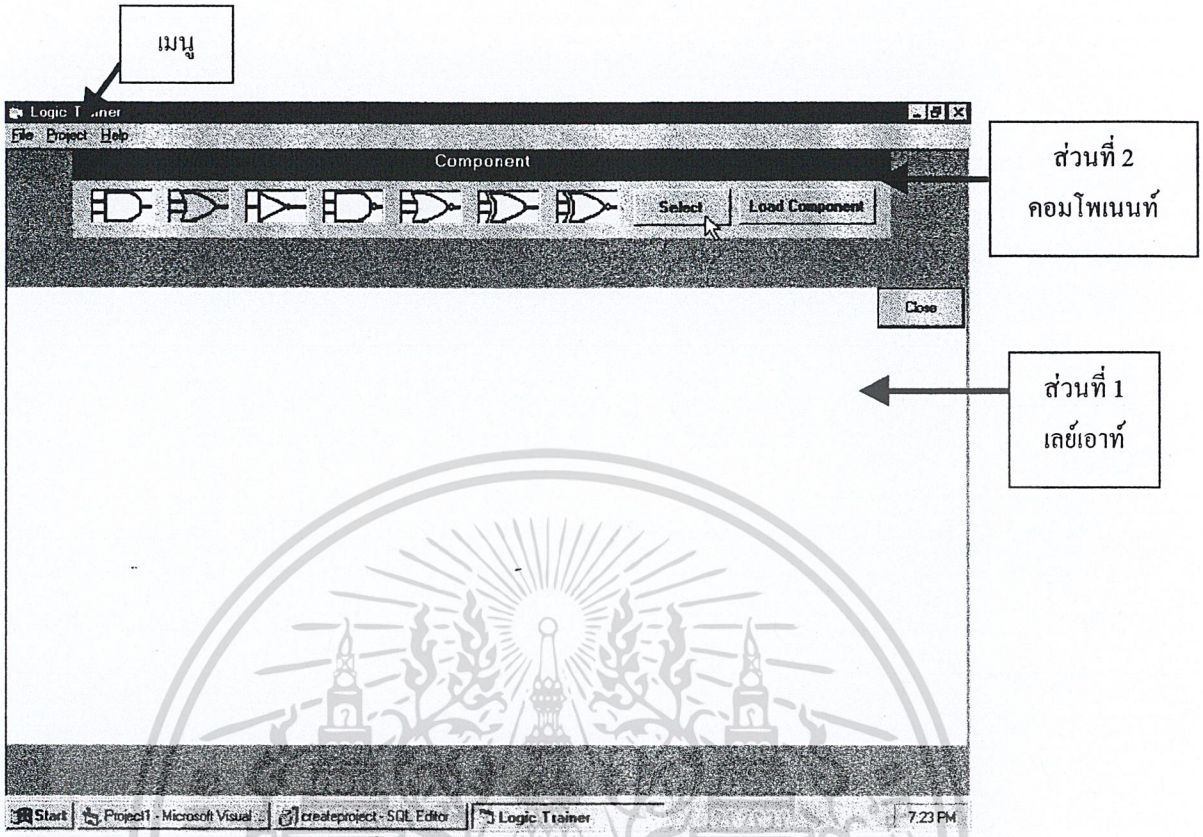
ตารางที่ &11 WIRED

8.3 การใช้งานโปรแกรมประยุกต์

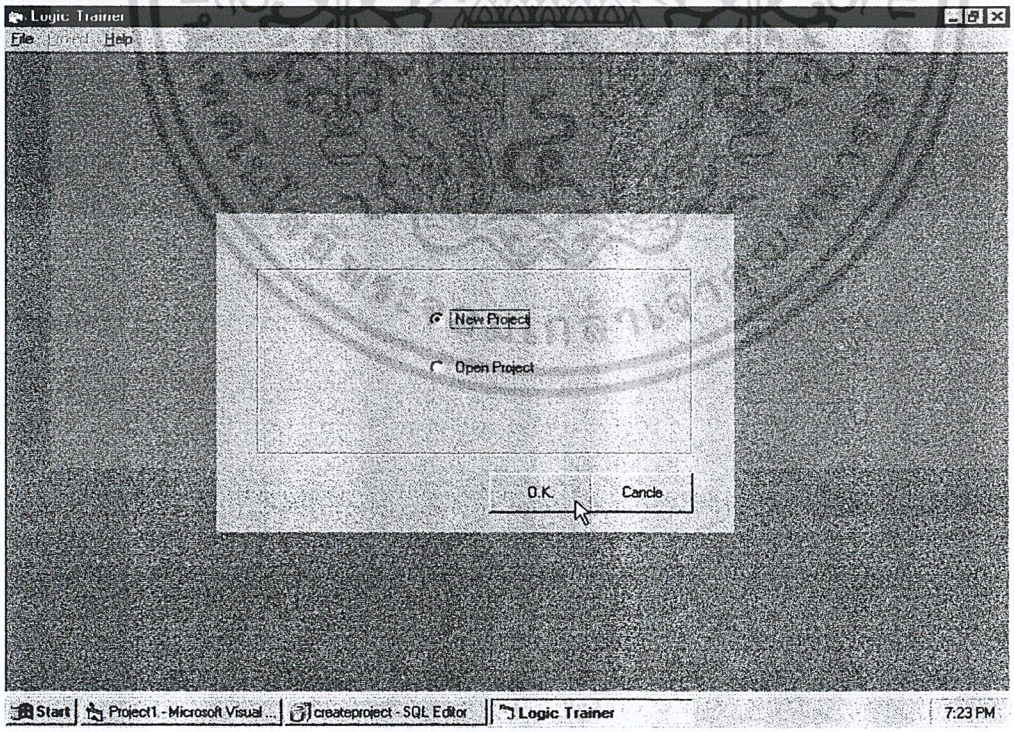
โปรแกรมลอจิกเทรนเนอร์ที่สร้างขึ้นจะมีส่วนประกอบหลักอยู่สองส่วนได้แก่

1. ส่วนเลย์เอาท์ เป็นส่วนที่ใช้ในการวาดภาพ
2. ส่วนคอมโพเนนท์ เป็นส่วนที่แสดงคอมโพเนนท์ต่างๆที่สามารถนำมาใช้สร้างวงจรได้

ในการจะใช้งานเมื่อเราทำการเปิดโปรแกรมขึ้นมาจะสามารถเลือกได้ว่าจะสร้างโปรเจคใหม่หรือเปิดโปรเจคเดิมที่มีอยู่ดังรูปที่ 8.5

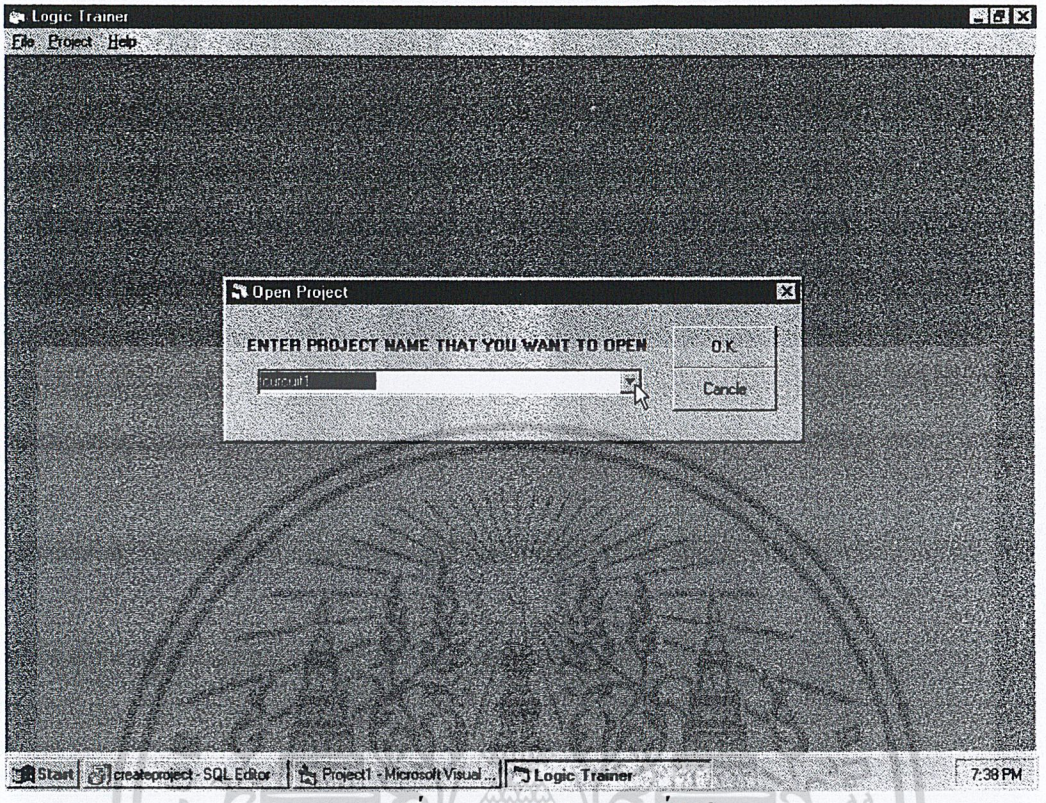


รูปที่ 8.4 ส่วนประกอบของโปรแกรมประยุกต์

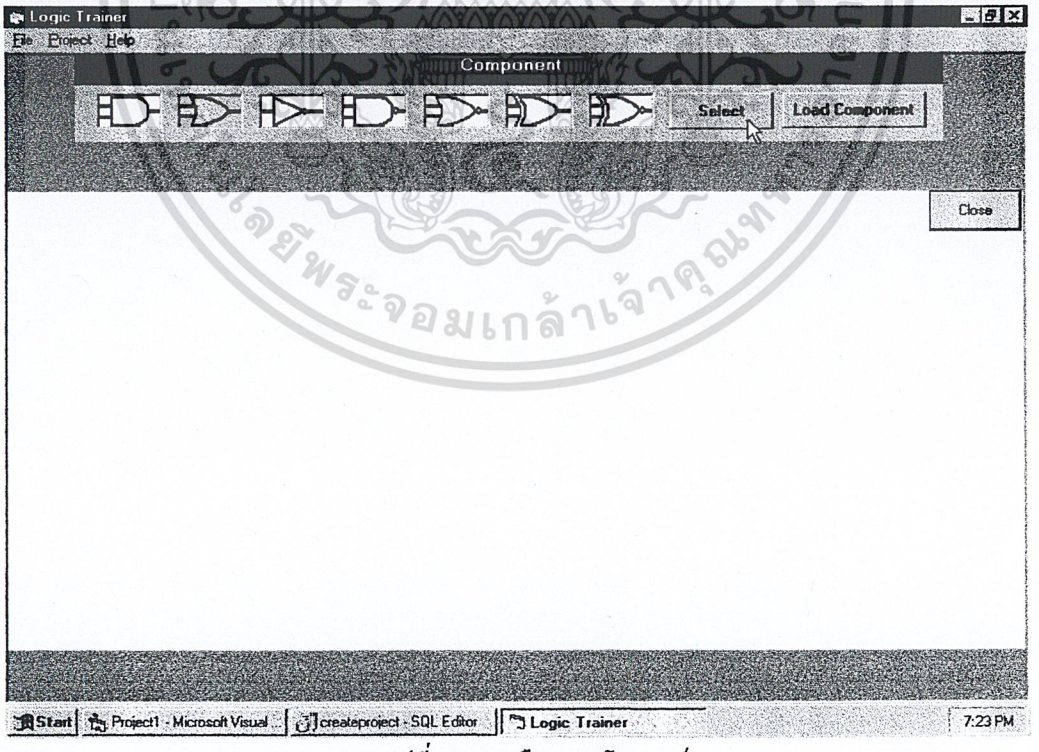


รูปที่ 8.5 การเลือกโหมดการทำงาน

ถ้าเลือกจะทำการเปิดโปรเจคที่มีอยู่จะมีรายชื่อโปรเจคที่มีทั้งหมดให้เลือก ดังรูปที่ 8.6



รูปที่ 8.6 การเปิดโปรเจคเดิมที่มีอยู่



รูปที่ 8.7 การเลือกคอมโพแนนท์

ในการเลือกคอมโพเนนท์ที่ต้องการทำได้โดยการคลิกบนคอมโพเนนท์ที่ต้องการแล้วคลิกที่ปุ่มSELECT คอมโพเนนท์ที่เลือกจะปรากฏบนเลย์เอาท์ให้สามารถเลือกไปวางบนตำแหน่งต่างๆได้ ดังรูปที่ 8.7

การเชื่อมต่อระหว่างแต่ละคอมโพเนนท์ทำได้โดยการคลิกเม้าส์ที่ปุ่มขวาบนคอมโพเนนท์ที่ต้องการจะเชื่อมต่อ จะขึ้นฟอร์มซึ่งเราสามารถเลือกได้ว่าต้องการเชื่อมต่อกับขาใดของตัวคอมโพเนนท์ดังกล่าว ดังรูปที่ 8.8 ถ้าเรากำหนดอินพุทสองขาเชื่อมต่อกัน โปรแกรมจะกำหนดให้อินพุทดังกล่าวเป็นอินพุทของระบบ

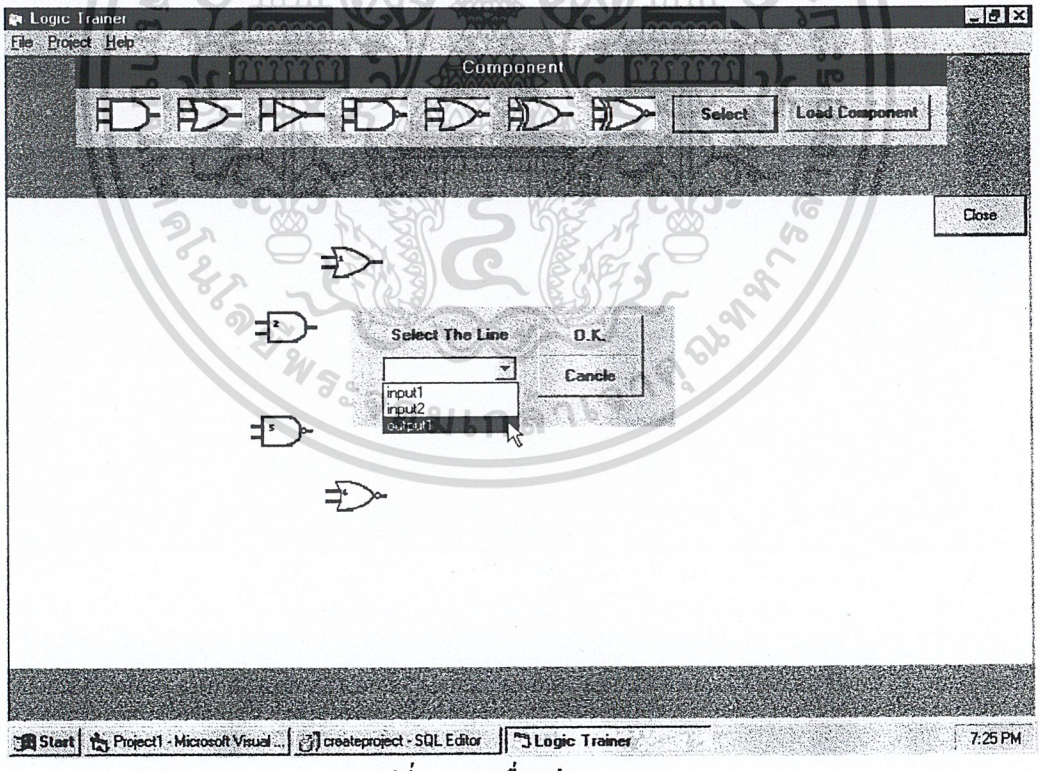
นอกจากนี้ยังสามารถกำหนดเอาท์พุทของระบบโดยจะมีเมนู Project-System Output ให้สามารถกำหนดเอาท์พุทของระบบเพิ่มเติม โดยมีขั้นตอนการกำหนดเอาท์พุทระบบ ดังรูปที่ 8.9 - 8.13

รูปที่ 8.9 และ 8.10 เป็นการเลือกการกำหนดเอาท์พุทระบบ

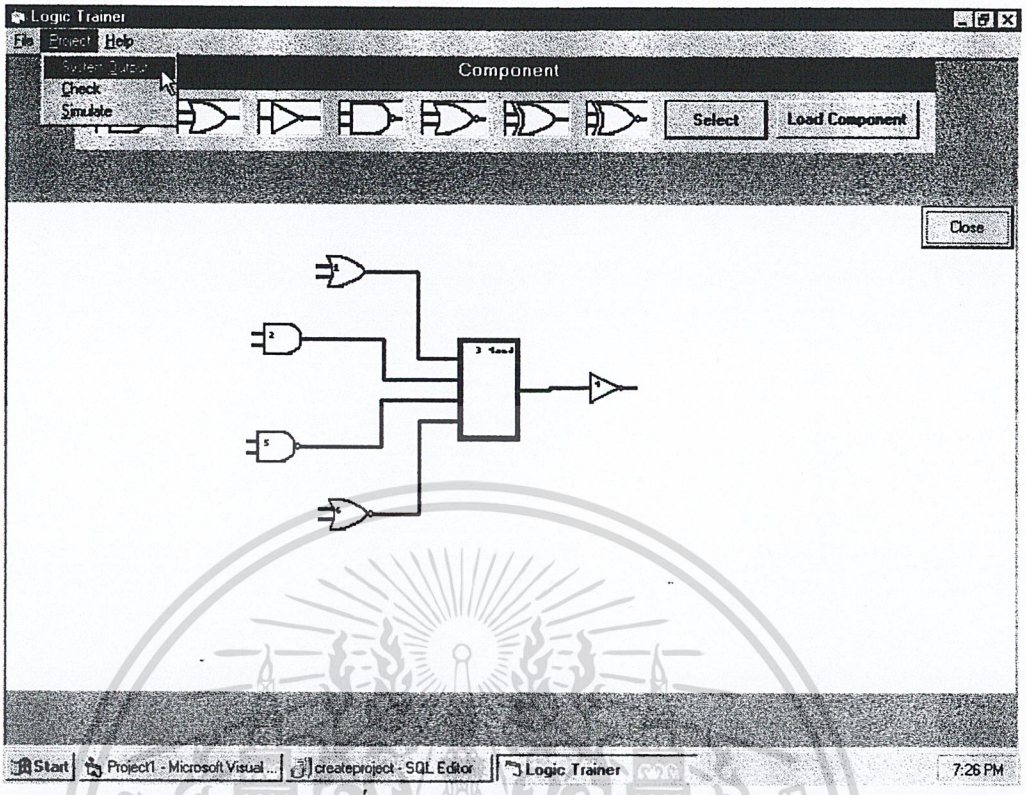
รูปที่ 8.11 เป็นฟอร์มที่ทำการกำหนดเอาท์พุทระบบประกอบไปด้วยคอมโบที่เก็บลิสท์ของขาต่างๆที่สามารถกำหนดเป็นเอาท์พุทระบบได้,ลิสท์ซึ่งแสดงเอาท์พุทระบบทั้งหมด และ ปุ่มที่ใช้ในการเพิ่มหรือลบเอาท์พุทระบบ

รูปที่ 8.12 เป็นการเพิ่มเอาท์พุทระบบทำได้โดยการเลือกขาที่ต้องการคลิกที่ปุ่ม Add ขาที่เลือกจะไปปรากฏในลิสท์

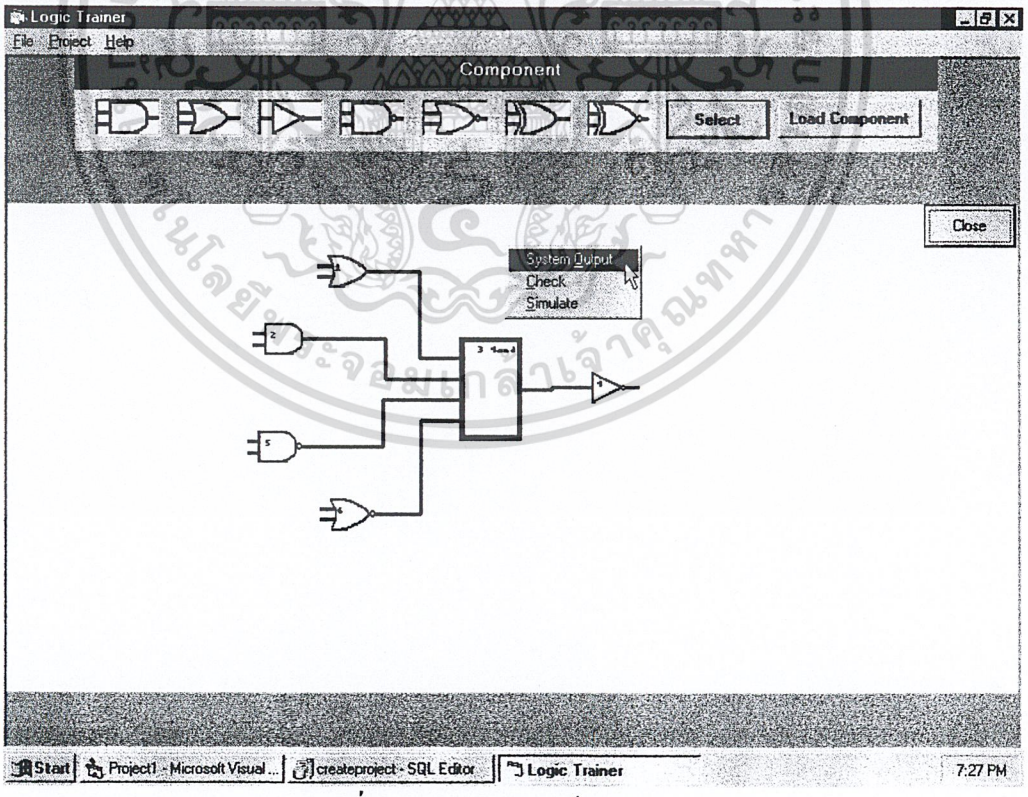
รูปที่ 8.13 เป็นการนำเอาท์พุทระบบที่ไม่ต้องการออกจากลิสท์ โดยเลือกขาที่ต้องการในลิสท์แล้วคลิกปุ่ม Remove



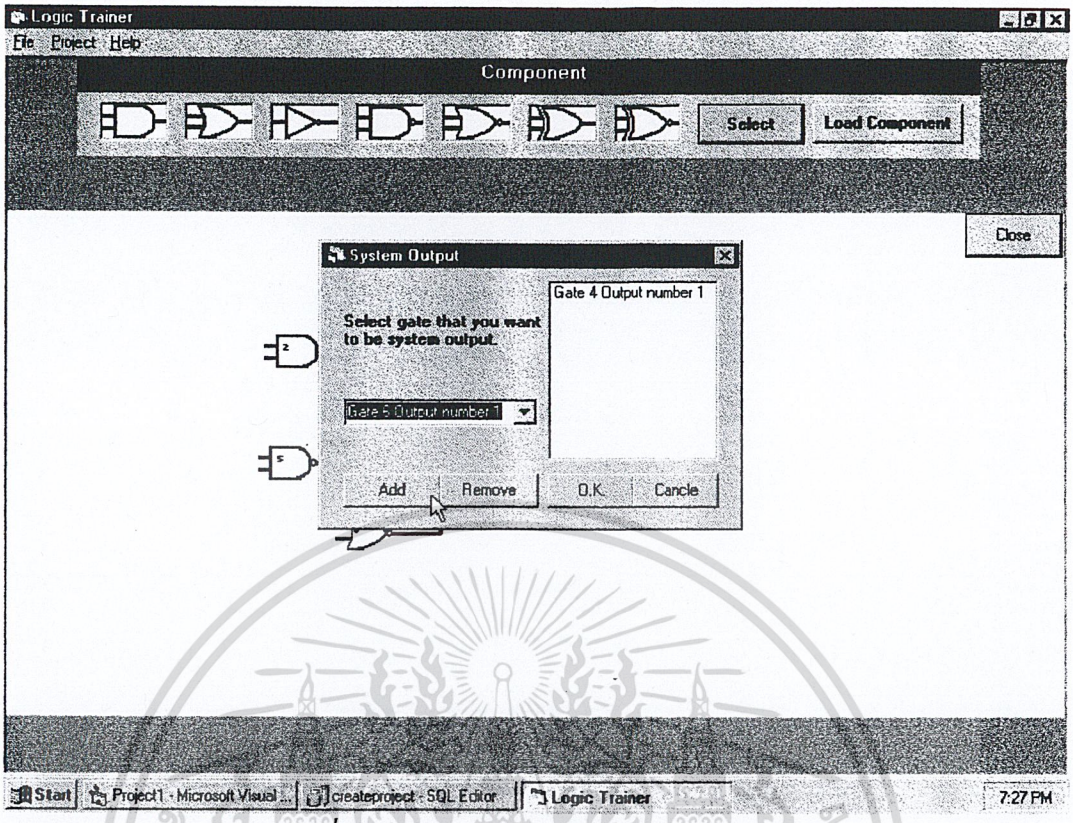
รูปที่ 8.8 การเชื่อมต่อลายวงจร



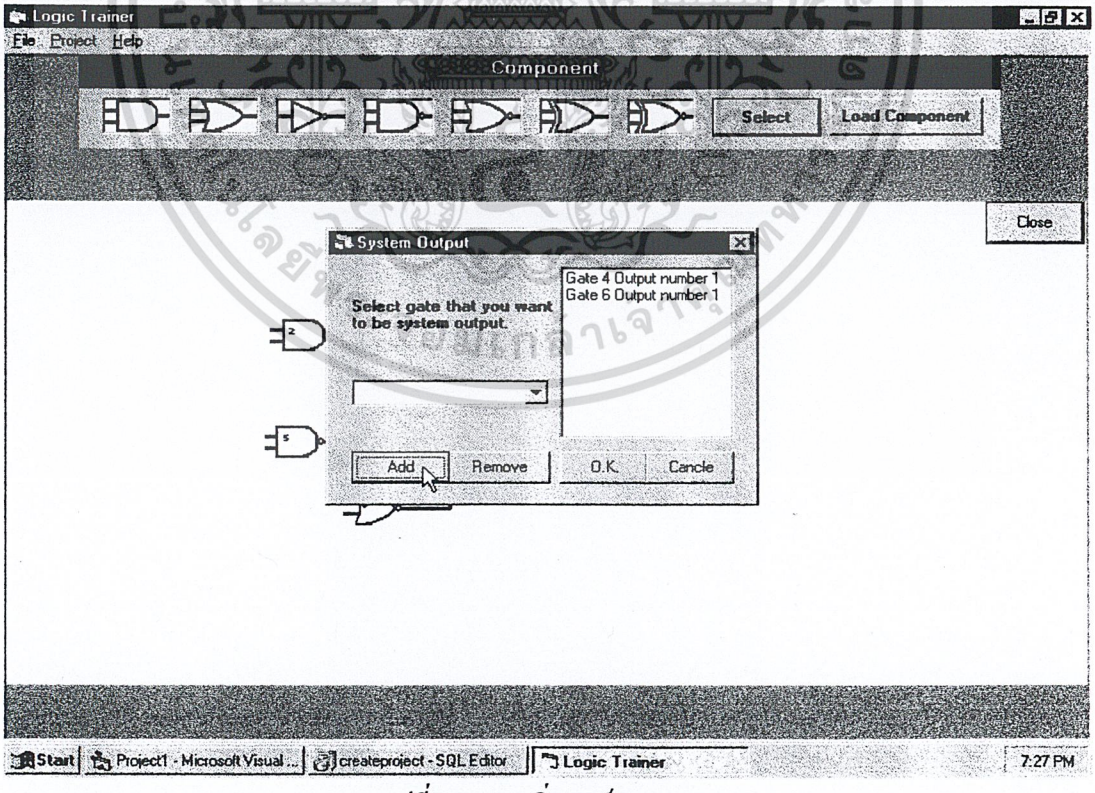
รูปที่ 8.9 การกำหนดเอาท์พุทระบบ



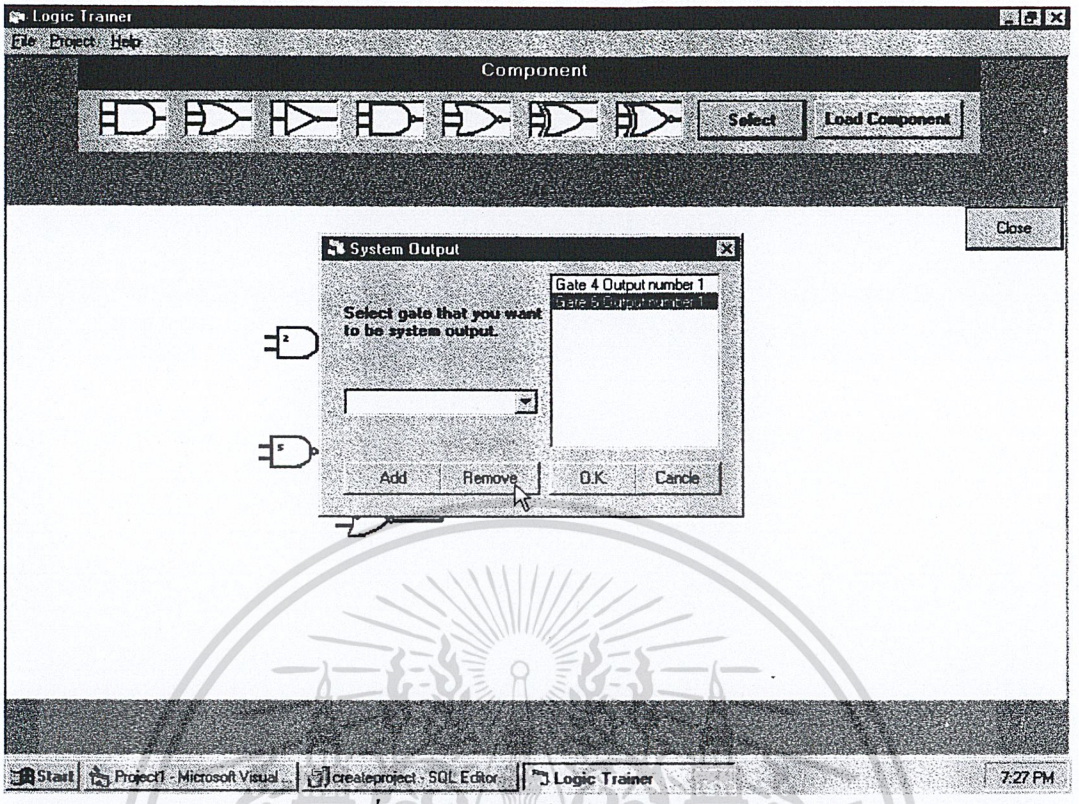
รูปที่ 8.10 การกำหนดเอาท์พุทระบบ



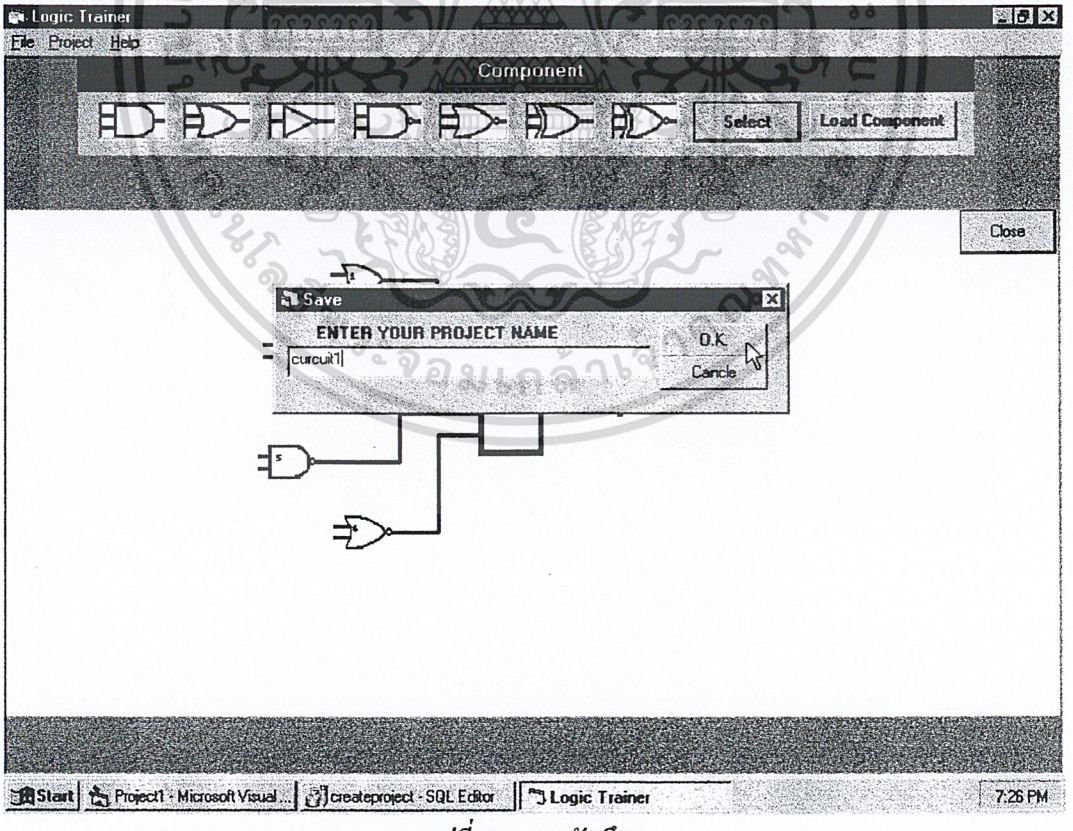
รูปที่ 8.11 ส่วนประกอบแบบฟอร์มเอาท์พุทระบบ



รูปที่ 8.12 การเพิ่มเอาท์พุทระบบ



รูปที่ 8.13 การลบเอาท์พุทระบบออกจากลิสท์



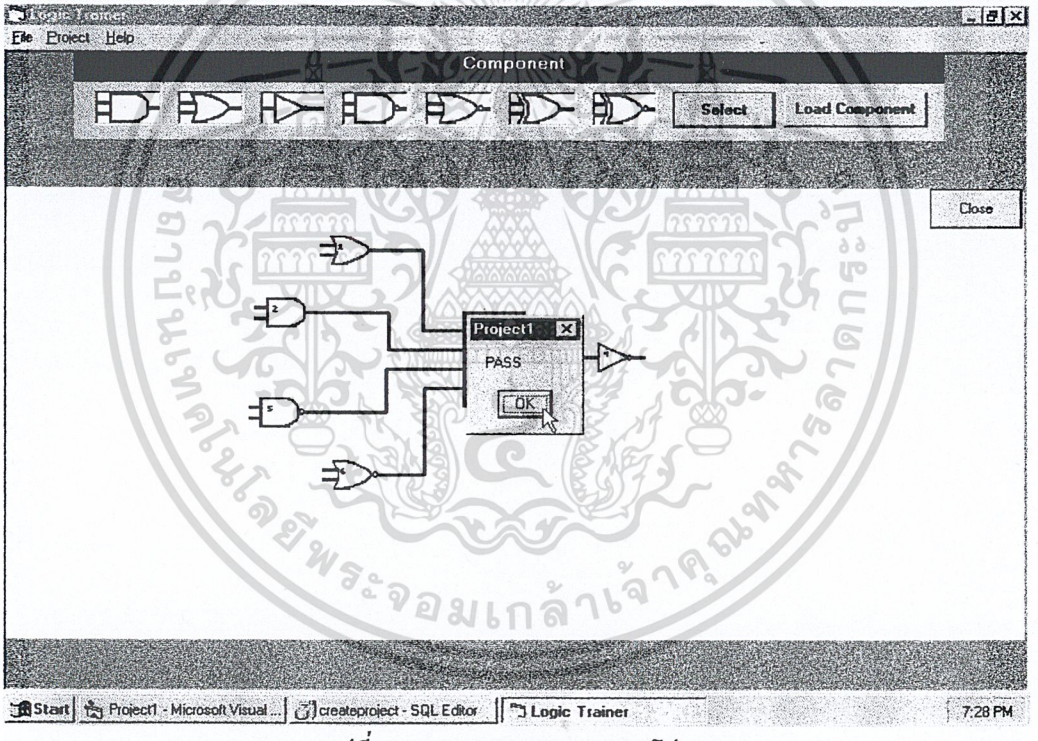
รูปที่ 8.14 การบันทึกวงจร

หลังจาการสร้างวงจรเสร็จสิ้นลงจะต้องทำการบันทึกวงจรลงบนฐานข้อมูลโดยเลือกเมนู File-Save แล้วทำการใส่ชื่อโปรเจคดังรูปที่ 8.14

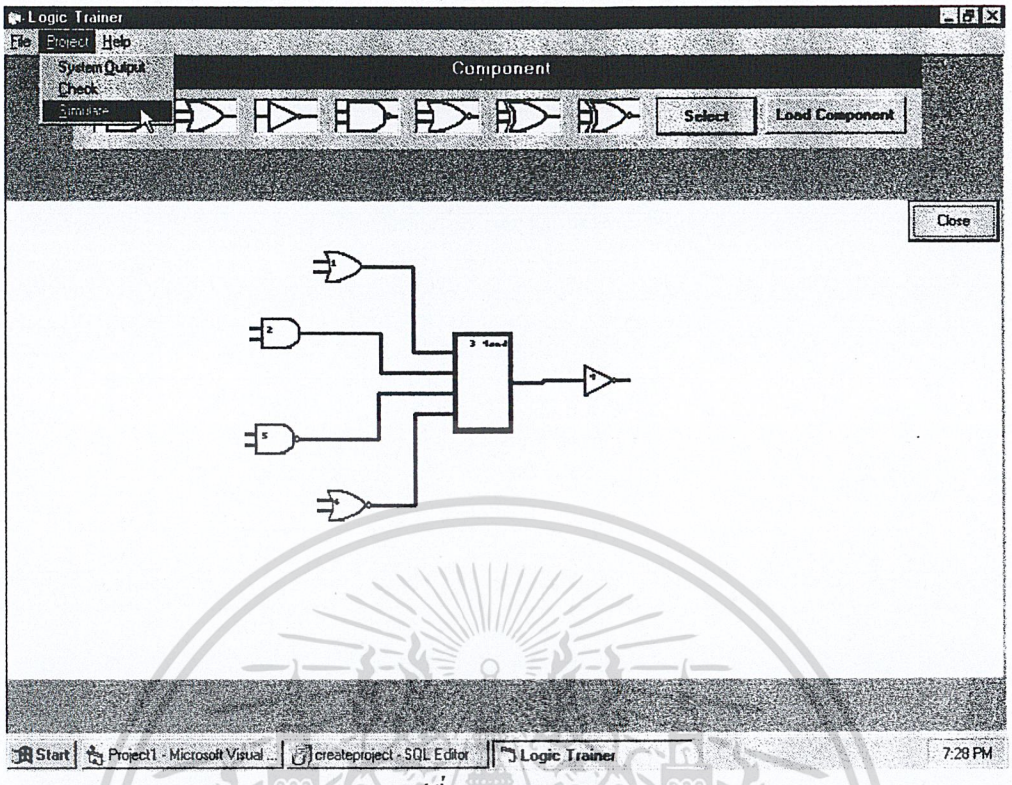
เมื่อจะทำการ Simulate การทำงานจะต้องทำการเช็คความถูกต้องของวงจรที่สร้างขึ้นก่อน ถ้าถูกต้องตามหลักก็จะสามารถทำการจำลองการทำงานได้ แต่ถ้าไม่ถูกต้องก็จะต้องทำการสร้างวงจรใหม่ รูปที่ 8.15 เป็นรูปที่แสดงผลการตรวจสอบวงจร

ในการจำลองการทำงานของโปรแกรมทำได้โดยการเลือกเมนู Project-Simualte ดังรูปที่ 8.16 ซึ่งจะแสดงแบบฟอร์มที่จะให้สามารถใส่ค่าของอินพุทระบบขาต่างๆ โดยเราสามารถเลือกใส่ค่า Low-High ได้แล้วทำการเลือกปุ่ม Set เป็นการกำหนดค่าให้ขานั้นและเมื่อเรียบร้อยก็เลือกที่ปุ่ม O.K. ดังรูปที่ 8.17 หลังจากทำการกำหนดค่าอินพุทระบบเสร็จสามารถเรียกดูได้จากเมนู Project-Input Ouput

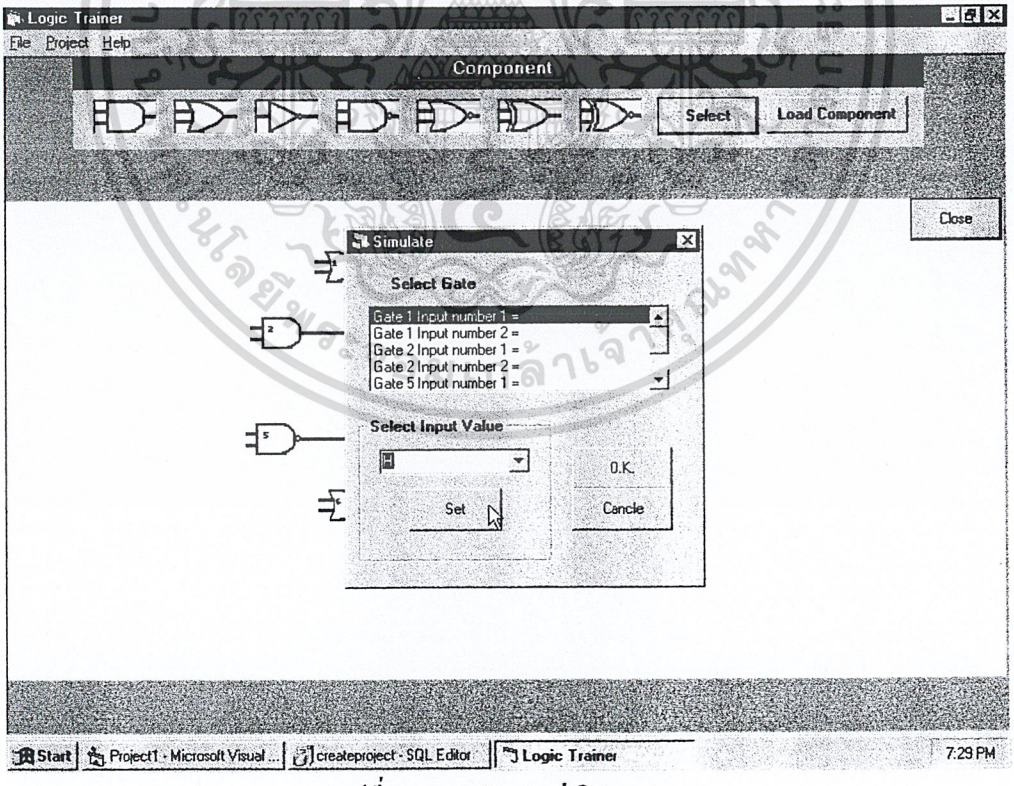
ในการดูค่าเอาท์พุทจะประกอบไปด้วยคอมโบที่แสดงคอมโพเนนท์ต่างๆ ในวงจรที่มีให้เลือกดู และมีลิสท์อีก 2 ลิสท์ของอินพุทและเอาท์พุท ดังรูปที่ 8.18



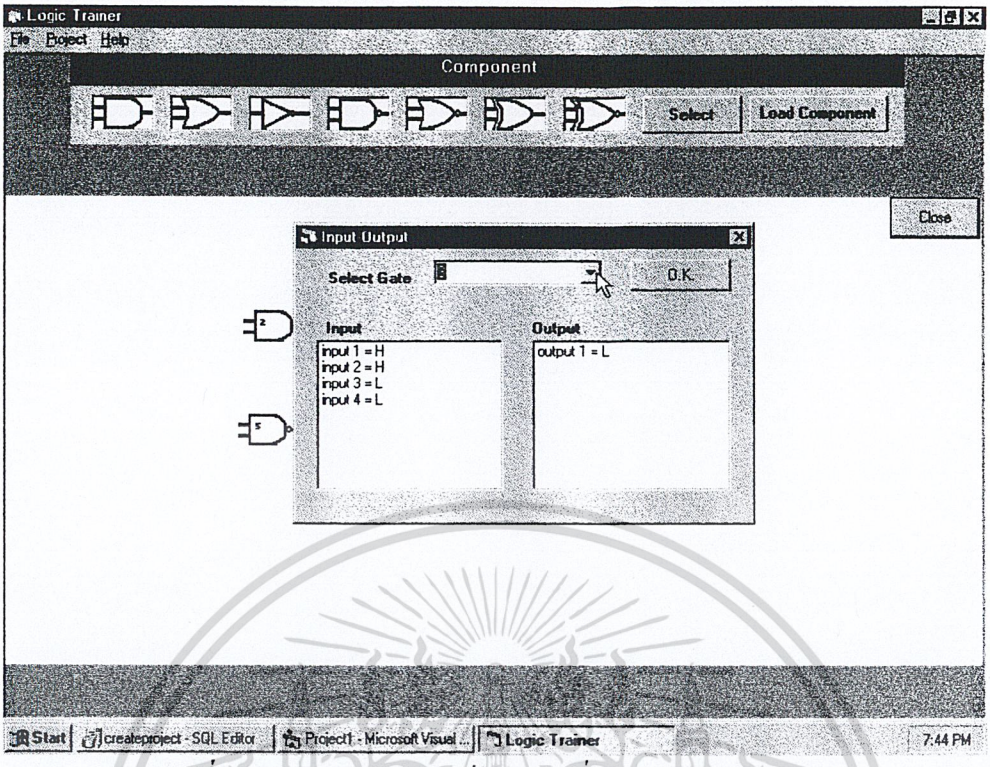
รูปที่ 8.15 แสดงผลการตรวจสอบโปรแกรม



รูปที่ 8.16 เมนู Simulate



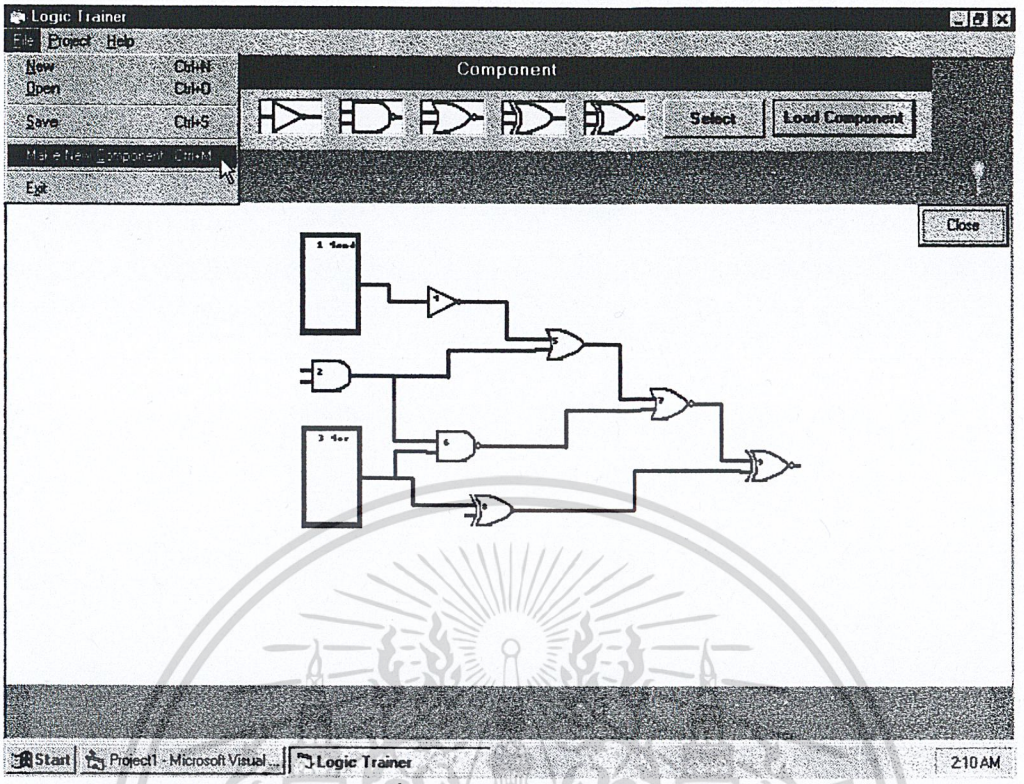
รูปที่ 8.17 การกำหนดค่าอินพุทระบบ



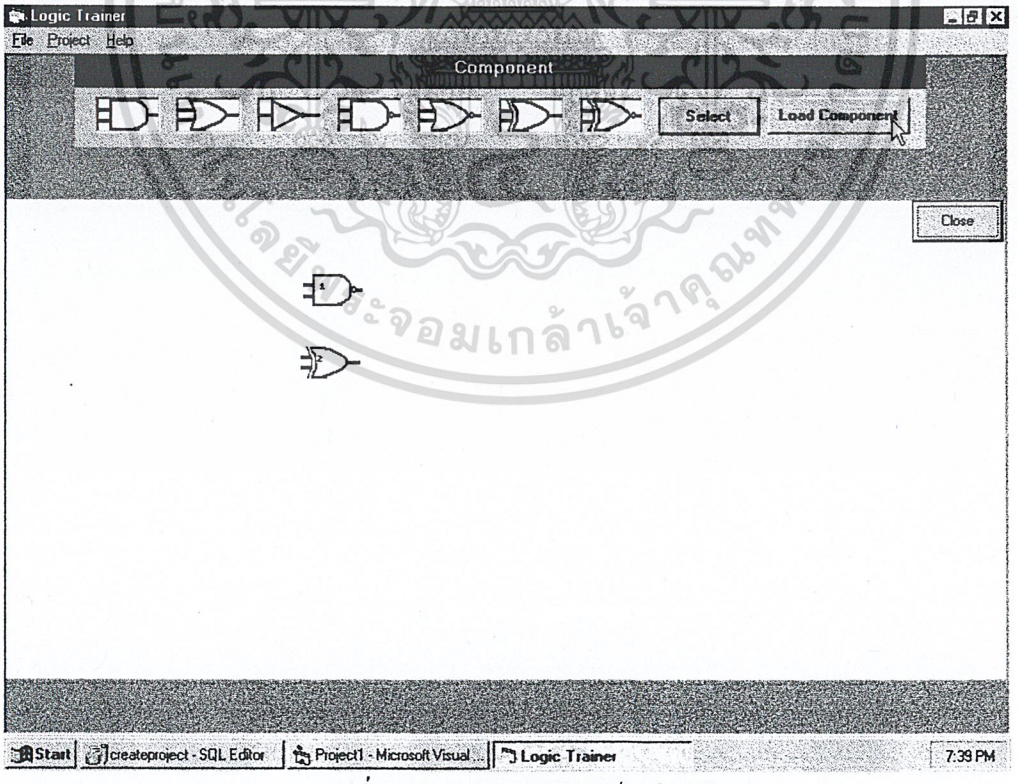
รูปที่ 8.18 การแสดงอินพุทเอาท์พุทระบบที่ทำการจำลองการทำงานแล้ว

นอกจากการใช้งานกับคอมโพเนนท์พื้นฐานแล้วยังมีความสามารถในการสร้างคอมโพเนนท์ใหม่จากคอมโพเนนท์ที่มีอยู่ไว้ใช้งานได้ โดยจะมีเมนู File-Make New Component ให้สามารถสร้างคอมโพเนนท์จากวงจรที่วาดขึ้น ดังรูปที่ 8.19

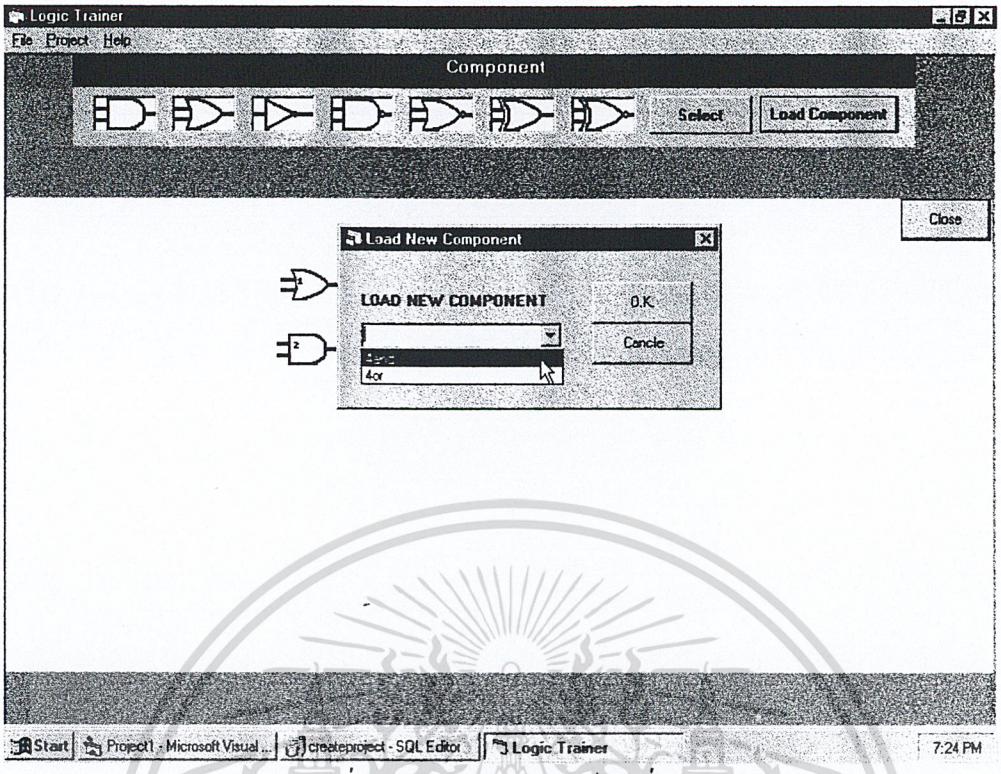
เมื่อเราทำการสร้างคอมโพเนนท์ใหม่ขึ้นมาแล้ว คอมโพเนนท์ดังกล่าวจะปรากฏในคอมโพเนนท์ลิสท์ให้เลือกมาใช้ได้ การจะนำคอมโพเนนท์ใหม่มาใช้ในวงจรทำได้โดยการคลิกที่ปุ่ม Load Component ดังรูปที่ 8.20 แล้วจะขึ้นฟอร์มให้เลือกคอมโพเนนท์ที่ต้องการดังรูปที่ 8.21



รูปที่ 8.19 การสร้างคอมโพเนนท์ใหม่



รูปที่ 8.20 การนำคอมโพเนนท์ใหม่มาใช้งาน



รูปที่ 8.21 แสดงคอมโพเนนท์ใหม่ที่มีให้ใช้งาน

บทที่ 9
บทสรุปและวิจารณ์

ในบทนี้จะกล่าวถึงการทำงานทั้งหมดในภาพรวมผลที่ได้รับจากงานวิจัยชิ้นนี้ แนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

9.1 สรุปผลโครงงาน

ในโครงงานได้ทำการศึกษาหลักการของโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ศึกษาภาษาฐานข้อมูล SQL3 ว่ามีความสามารถอะไรเพิ่มขึ้นมาในการใช้งานร่วมกับโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ แล้วศึกษาการออกแบบและการใช้งานฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อประยุกต์ใช้ในงานต่างๆ

หลังจากมีความรู้และความเข้าใจโมเดลฐานข้อมูลแล้วได้ทำการศึกษาระบบจัดการฐานข้อมูล (Database Management System : DBMS) ที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ในโครงงานใช้อินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ (Informix Universal Server) เป็นระบบจัดการฐานข้อมูลที่สามารถจัดเก็บข้อมูลได้ทั้งที่เป็นโมเดลฐานข้อมูลเชิงสัมพันธ์และโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ศึกษาและทดลองใช้งานชนิดข้อมูลต่างๆที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ที่ระบบจัดการฐานข้อมูลมีให้และศึกษาการใช้งานของระบบจัดการฐานข้อมูลเทียบกับมาตรฐาน SQL3

หลังจากมีความคุ้นเคยกับการใช้งานระบบจัดการฐานข้อมูลก็ได้ทำการทดลองสร้างโปรแกรมประยุกต์ (Application) ที่ทำการเก็บข้อมูลที่มีความซับซ้อนซึ่งได้ทำการเลือกการจัดเก็บวงจรดิจิตอลซึ่งประกอบไปด้วยเกทชนิดต่างๆ โดยใช้คุณลักษณะของฐานข้อมูลเชิงวัตถุสัมพันธ์ในการจัดเก็บข้อมูล โดยในการพัฒนาโปรแกรมประยุกต์เริ่มจากการออกแบบฐานข้อมูลโดยยึดแนวคิดการออกแบบตามคุณลักษณะของฐานข้อมูลเชิงวัตถุสัมพันธ์แล้วทำการสร้างรูทีนที่เขียนด้วย Stored Procedure Language หรือ SPL ซึ่งเป็นภาษาที่มีความสามารถของภาษา SQL แล้วเพิ่มส่วนในการควบคุมการทำงานเช่นการวนลูปและการตัดสินใจเพิ่มเข้าไป โดยรูทีนดังกล่าวจะมีไว้ให้ไคลเอนท์เรียกเพื่อจัดการกับฐานข้อมูล

การติดต่อระหว่างไคลเอนท์กับดาต้าเบสเซิร์ฟเวอร์ไม่ได้ติดต่อกันโดยตรงแต่ทำการติดต่อกันผ่านทางดาต้าไดเร็คเตอร์ซึ่งจะเป็นตัวที่ทำหน้าที่อิมพอร์ตโมเดลในฐานข้อมูล ให้กับโปรแกรมประยุกต์เพื่อทำให้การติดต่อกับฐานข้อมูลเป็นไปได้โดยง่าย

โปรแกรมประยุกต์ซึ่งพัฒนาโดยใช้วิชวลเบสิกมีความสามารถในการทำงานในการใช้สร้างวงจรดิจิตอลพื้นฐานซึ่งประกอบไปด้วยเกทต่างๆจำนวนเท่าใดก็ได้ แต่มีข้อจำกัดในการที่ไม่สามารถนำเอาท์พุทกลับมาเป็นอินพุทหรือเป็นฟีดแบคได้เพราะไม่มีการพิจารณาแกนเวลา อีกทั้งไม่สามารถมีอินพุทซึ่งมาจากเอาท์พุทสองเส้นได้

จากตัวอย่างการจัดเก็บโดยใช้โมเดลฐานข้อมูลเชิงสัมพันธ์จะเห็นว่าการจัดเก็บรูปวงจรเมื่อเทียบกับการใช้โมเดลเชิงวัตถุสัมพันธ์นั้น จะมีความซ้ำซ้อนมากกว่า อาทิเช่น หากมีเส้นลายวงจรใดที่เชื่อมต่อมากกว่า 2 เกทเช่น เอาท์พุทของ C3 เชื่อมต่อกับอีก 2 เกทถ้าเรามีการใส่ Value หรือมีการอัพเดทใดๆต้องมีการทำการอัพเดทหลายที่ (Multiple Update) อีกทั้งยังไม่สามารถใช้ SQL เพียงคำสั่งเดียวในการเรียกดูค่าต่างๆได้ถ้าต้องการดูก็ต้องมีการ Join Table เช่น ถ้าต้องการดูว่าเอาท์พุท C1 ต่อกับเกทชนิดใดที่ขาไหนต้องใช้ถึง 2 เทเบิลในการตอบคำถามดังกล่าวนอกจากความยุ่งยากในการคิวรีแล้วในด้านการใช้งานออบเจกต์เช่นการหาค่าเอาท์พุทยังทำได้ยากกว่าในแบบโมเดลเชิงวัตถุสัมพันธ์อีกด้วยเพราะโมเดลเชิงวัตถุสัมพันธ์เราสามารถเขียนเมธทอดของออบเจกต์นั้นเพื่อใช้ในการคำนวณค่าได้โดยตรง

9.2 แนวทางในการพัฒนาและแนวทางในการประยุกต์ใช้

จากการที่โปรแกรมประยุกต์ที่ได้ทำการพัฒนาใช้โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งสนับสนุนการเก็บข้อมูลที่มีความซับซ้อน เช่นออบเจกต์ อีกทั้งในการออกแบบโปรแกรมประยุกต์และออกแบบฐานข้อมูลได้พยายามออกแบบโดยเน้นแนวคิดเชิงวัตถุและรีเลชั่นไว้ การพัฒนาต่อจึงอาจทำได้โดยการพัฒนาสร้างคอมโพเนนท์ใหม่ซึ่งประกอบไปด้วยเกทพื้นฐานตามโปรแกรมประยุกต์เดิม จากแทนที่จะสร้างโปรเจคหนึ่งโปรเจคเราน่าจะสามารถสร้างคอมโพเนนท์ใหม่ให้กับระบบไว้ใช้งานในโปรเจคอื่นๆได้ หรืออาจทำการพัฒนาเพิ่มในส่วนของฟีดแบกซึ่งจะต้องพิจารณาแกนเวลาเข้ามาในการคำนวณหาเอาท์พุทของวงจร

วัตถุประสงค์การใช้งานโปรแกรมประยุกต์นี้คือการพัฒนาเสมือนเป็นลอจิกเทรนเนอร์ ให้ผู้ที่ทำการออกแบบวงจรดิจิตอลสามารถนำมาสร้างและทดลองวงจรที่ต้องการจะสร้าง ดูผลของเอาท์พุทในกรณีต่างๆซึ่งเมื่อโปรแกรมประยุกต์นี้เสร็จสามารถทำหน้าที่ดังกล่าวได้ และหากมีการพัฒนาต่อก็จะสามารถเป็นโปรแกรมที่ช่วยทำการจำลองการทำงานของวงจรดิจิตอลได้สมบูรณ์

ภาคผนวก

ภาคผนวก ก. การติดตั้ง และ การทดลองใช้งาน Informix Universal Server

1. การติดตั้ง

ยูนิเวอร์แซลเซิร์ฟเวอร์ (Universal Server) เป็นผลิตภัณฑ์แบบไคลเอนท์/เซิร์ฟเวอร์ ดั้งนั้นยูนิเวอร์แซลเซิร์ฟเวอร์จึงแบ่งเป็น2ส่วน คือ ส่วนของเซิร์ฟเวอร์ และส่วนของไคลเอนท์

- ส่วนของเซิร์ฟเวอร์จะประกอบไปด้วยยูนิเวอร์แซลเซิร์ฟเวอร์ (Informix Universal Server) ที่สนับสนุนความสามารถฐานข้อมูลเชิงวัตถุสัมพันธ์และยูทิลิตี้ (Utilities) ต่างๆ เช่น Dbaccess,Onmode,Oninit และ Onspaces เป็นต้น
- ส่วนของไคลเอนท์มีทูล (Tool) เพื่อใช้งานต่างๆดังนี้เช่น
  - Schema Knowledge ทำการแสดงข้อมูลทั้งหมดของยูนิเวอร์แซลเซิร์ฟเวอร์ เช่น รายละเอียดของ Type Table , Routines , Operator Cast , Aggregate และ Access Method
  - SQLeditor เป็น ISQL ของยูนิเวอร์แซลเซิร์ฟเวอร์ที่อยู่ฝั่งไคลเอนท์
  - DBDK DataBlade Developer Kit เป็นทูลที่ใช้พัฒนาคาต้าเบสโมดูล(Datablade Module)
  - Iconnect เป็น ODBC ของยูนิเวอร์แซลเซิร์ฟเวอร์

Connectivity File : SQLhosts เป็นไฟล์ที่เก็บข้อมูลที่ ไคลเอนท์ ใช้ในการเชื่อมต่อกับเซิร์ฟเวอร์ sqlhosts file จะประกอบด้วย 5 ส่วนได้แก่

- Dbservername Field ชื่อของคาต้าเบสเซิร์ฟเวอร์
- Nettype Field ชนิดของอินเตอร์เฟซและโปรโตคอล ประกอบด้วย 3 ส่วนย่อย คือ

d : Database Server Product ได้แก่

on	Universal Server หรือ Online Dynamic Server
ol	Universal Server หรือ Online Dynamic Server
se	INFORMIX-SE
dr	INFORMIX-Enterprise Gateway with DRDA

i : Interface Type ได้แก่

ipc	IPC (Interprocess Communications)
soc	Sockets
tli	TLI (Transport Layer Interface)

p : Network Protocol or IPC Mechanism

shm Shared-Memory Communication

str Stream-Pipe Communication

tcp TCP/IP Network Protocol

spx IPX/SPX Network Protocol

- Hostname Field ชื่อของโฮสต์
- Servicename Field
- Options Field จะประกอบไปด้วยส่วนที่ใช้กำหนดค่าต่างๆ เช่น

Keep-Alive Option : k

Security Option : s

Buffer-Size Option : b

Communications Support Module Option : csm

ONCONFIG Parameters สำหรับการเชื่อมต่อ เป็นคอนฟิกูเรชั่นไฟล์ (Configuration File) อยู่ใน $INFORMIXDIR/etc Directory ระหว่างการติดตั้งเก็บข้อมูลที่ใช้ในการกำหนดค่าเริ่มต้น (Initial Setting) และพารามิเตอร์ในการคอนฟิก (Configuration Parameter)

- RootPath
- RootName
- ServerNum
- DBServerName
- DBServerAliases
- NetType

Utility Enhancement ตัวอย่างของยูทิลิตี้ที่ใช้งาน

- Oninit Initialize Disk Space ต้องล็อกอิน (Log In) เป็นรูท (Root)หรือยูเซอร์อินฟอร์มิกซ์ (User Informix) เพื่อที่จะเรียกใช้ oninit
  - -i Inintial Disk Space Shared Memory และ จะทำการเคลียร์ข้อมูลต่างๆที่อยู่ภายในฐานข้อมูล
- Onmode ต้องล็อกอินเป็นรูทหรือยูเซอร์อินฟอร์มิกซ์ onmode จะทำการเปลี่ยนโหมดในการทำงานของยูนิเวอร์แวลเซิร์ฟเวอร์ (Universal Server Operating Mode)
  - -k เปลี่ยนเป็น Off-Line Mode
  - -m เปลี่ยนจาก Quiescent เป็น On-Line Mode
  - -s Shut Down Mode

- u Shut Down แบบ Immediate

- ❑ Onspace ต้องล็อกอินเป็นรูทหรือยูเซอร์อินฟอร์มิกซ์โดยจะทำการกำหนด Create/Drop Sbspace หรือ Dbspace , Add/Drop Chunk

- c Create
- d Drop
- a Add Chunk
- d Drop Chunk

ส่วนการ Create Dbspace

- d Dbspace
- o Offset
- p Pathname
- s Size

ส่วนการ Create Sbspace

- S Sbspace
- o Offset
- p Pathname
- s Size

onmode กับ onspace ต้องกำหนดก่อนที่จะทำยูทิลิตี้อื่นๆ

SetNet32 ด้วย SetNet32 Utility สามารถกำหนดหรือแก้ไข Environment Variable และ Network Parameter ขณะรันไทม์ (Runtime) Setnet32 เป็นยูทิลิตี้ที่ติดตั้งบนไคลเอนท์พีซี, Setnet32 Utility มี 4 ส่วนที่สำคัญคือ

- ❑ Environment Enable ใช้กำหนด Environment Variables
- ❑ Server Information ใช้กำหนดข้อมูลของคาต้าเบสเซิร์ฟเวอร์
- ❑ Host Information ใช้กำหนดคลอนฟิกของโฮสต์แมชชีน (Host Machine)
- ❑ About Setnet32 ใช้แสดงข้อมูล Setnet32 Utility

Schema Knowledge เป็นทูลที่แสดงข้อมูลต่างๆในรูปแบบของกราฟิก (Graphical Knowledge Tool) ใช้แสดงเมต้าดาต้า (Meta Data) ของฐานข้อมูล ไม่สามารถใช้แก้ไขข้อมูลใดๆบนฐานข้อมูลได้ เราสามารถใช้ Schema Knowledge ดูเมต้าดาต้าต่อไปนี้ได้

- ❑ Database
- ❑ Routine
- ❑ Table และ View
- ❑ Row Type

- Inheritance View

ก่อนจะเข้ามาใช้ต้องมีการล็อกอิน ก่อนโดยจะต้องทำการใส่ ชื่อเซิร์ฟเวอร์,ชื่อฐานข้อมูล, ชื่อผู้ใช้และรหัสผ่าน

SQLEditor เป็นทูลที่แสดงผลเป็นกราฟิก (Graphical Tool) ซึ่งใช้ทำการติดต่อกับยูนิเวอร์แซลเซิร์ฟเวอร์ และทำการคิวรีข้อมูลจากฐานข้อมูล (Query Database Information) และดูผลของการคิวรีของ SQL Statement ต้องมีการกำหนดค่าต่างๆใน SetNet32

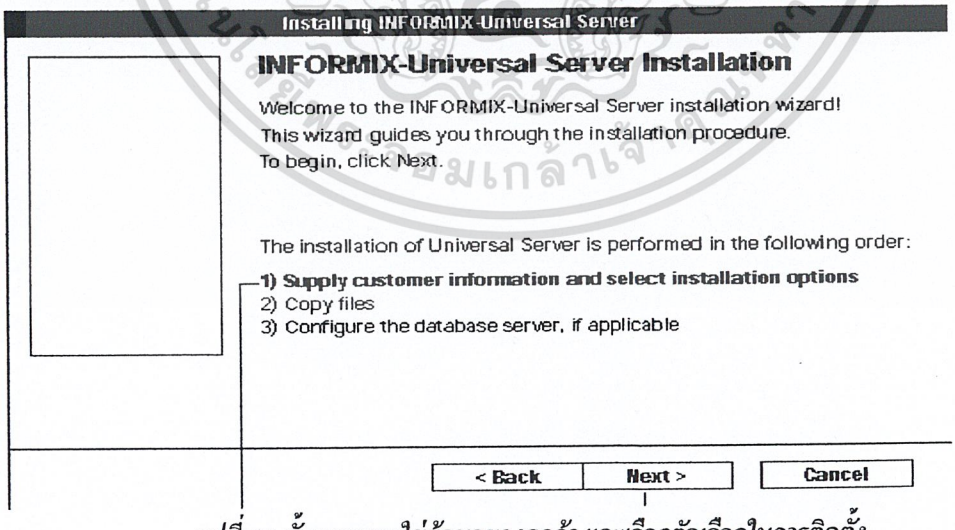
2. การเตรียมการติดตั้ง

การติดตั้งเซิร์ฟเวอร์ของอินฟอร์มิกซ์ยูนิเวอร์แซลเซิร์ฟเวอร์ มีอยู่ 2 แพลทฟอร์ม (Platform) คือบนยูนิกซ์ (UNIX) และวินโดว์เอ็นที (Window NT) ได้ทำการทดลองติดตั้งบนพีซีซึ่งเลือกใช้เวอร์ชั่น (Version) บนวินโดว์เอ็นทีซึ่งมีความต้องการของระบบ ดังนี้

- วินโดว์เอ็นที เวอร์ชั่น 4.0
- โปรโตคอล TCP/IP
- หน่วยความจำแรม (RAM) 16 เมกกะไบท์
- เนื้อที่บนหน่วยความจำ (Disk Space) 40 เมกกะไบท์

3.ขั้นตอนการติดตั้ง

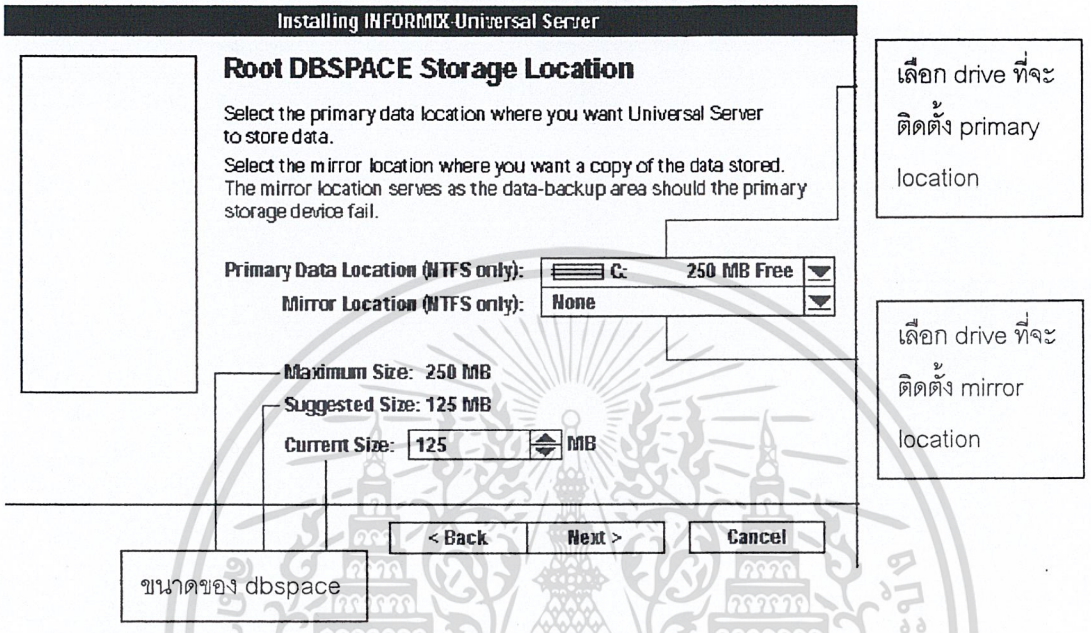
1. ผู้ติดตั้งต้องอยู่ในกลุ่มของ Administration Group
2. Run Setup.exe เพื่อทำการติดตั้ง
3. ใส่ Serial Number และ Cd Key
4. ใส่ Product Registration เช่นชื่อ ตำแหน่ง บริษัท ที่อยู่ และข้อมูลในการติดต่อ



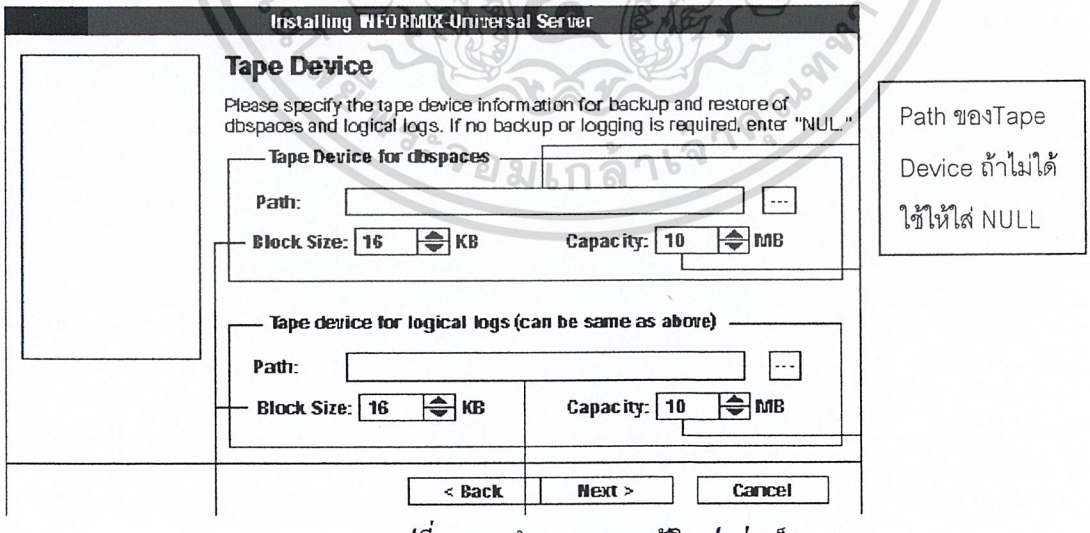
รูปที่ 1ก ขั้นตอนการใส่ข้อมูลของลูกค้า และเลือกตัวเลือกในการติดตั้ง

5. กำหนด Directory ที่จะทำการติดตั้ง
6. เลือกการกำหนด บทบาท ของ DB-Admin ว่าจะแยกหน้าที่กันในแต่ละงาน หรือ รวมกันใช้เพียง Account เดียว
7. กำหนดชื่อของ Database Server และการติดตั้งจะเริ่มถ่ายโอนไฟล์ลงสู่ระบบ

รูปที่ 2ก การกำหนดขนาดฐานข้อมูล



8. กำหนด Root Dbspace
9. กำหนด Sbspace



รูปที่ 3ก การกำหนดคุณสมบัติเทปแม่เหล็ก

10. กำหนด Tape Device

11. การติดตั้ง จะให้กำหนดรหัสผ่าน (Password) สำหรับ Account Informix เนื่องจาก Informix จะสร้างAccount Informix มีหน้าที่ดูแลยูนิเวอร์แซลเซิร์ฟเวอร์ (DBadmin)
12. กำหนดที่เก็บ Sqlhost File
13. เมื่อทุกอย่างเสร็จสิ้น ทำการกหนดค่าเริ่มต้นให้กับระบบ (Initialize)

4. เริ่มการใช้งาน

เมื่อทำการติดตั้งสิ้นสุดระบบจะมียูนิเวอร์แซลเซิร์ฟเวอร์และยูทิลลิตี้ต่างๆ โดยที่ยูนิเวอร์แซลเซิร์ฟเวอร์จะทำหน้าที่เสมือนเป็นบริการ (Service) ของวินโดว์เอ็นที สามารถเริ่มและยุติ (Start/Stop) การให้บริการได้ใน Control Panel - Service

DBaccess เป็นยูทิลิตี้ที่ทำการจัดการกับข้อมูล (Manipulate Data)

- Query-Language เป็นเมนูที่ทำการเขียน SQL และ Run-SQL
- Connection เป็นเมนูสำหรับเปิดการติดต่อกับฐานข้อมูล
- Database เป็นเมนูที่เลือก และดูข้อมูลของฐานข้อมูล
- Table เป็นเมนูที่ทำการ สร้าง เปลี่ยนแปลง ทำลาย และดูข้อมูลในเทเบิล
- Session เป็นเมนูที่ดึงข้อมูล เกี่ยวกับ การติดต่อของ DBaccess ปัจจุบัน

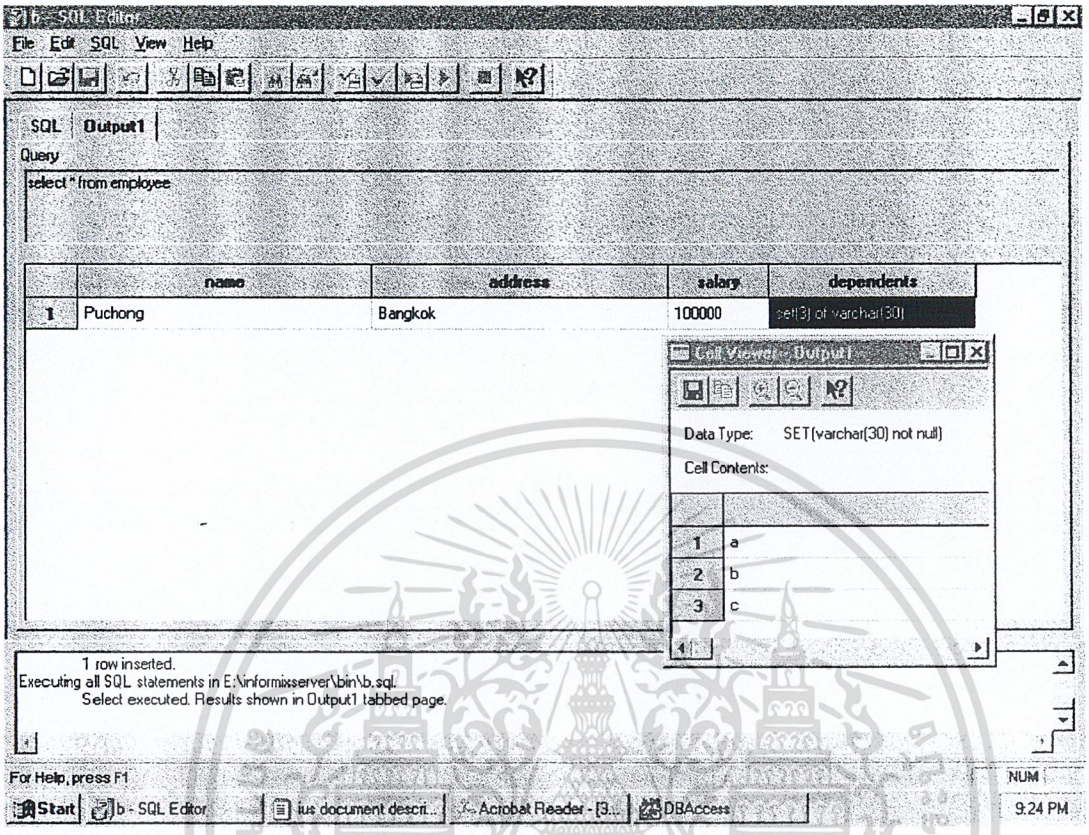
เมื่อเริ่มต้นจะต้องสร้างฐานข้อมูลขึ้นมาก่อนเป็นอันดับแรก โดยใช้ เมนู Database - Create แล้วจึงใส่ชื่อของฐานข้อมูล กำหนด Dbspace ที่จะติดตั้ง และการใช้ Log

5. ตัวอย่างการใช้งาน

Collection type

```
CREATE TABLE Employee
(Name          CHAR(30),
Address        CHAR (40),
Salary         INTEGER,
Dependents     SET(VARCHAR(30) NOT NULL));
```

INSERT INTO Employee VALUES ('Puchong','Bangkok',100000,"set{'a','b','c'}")



รูปที่ 4ก การใช้งาน Collection Type (Set)

CREATE TABLE Employee

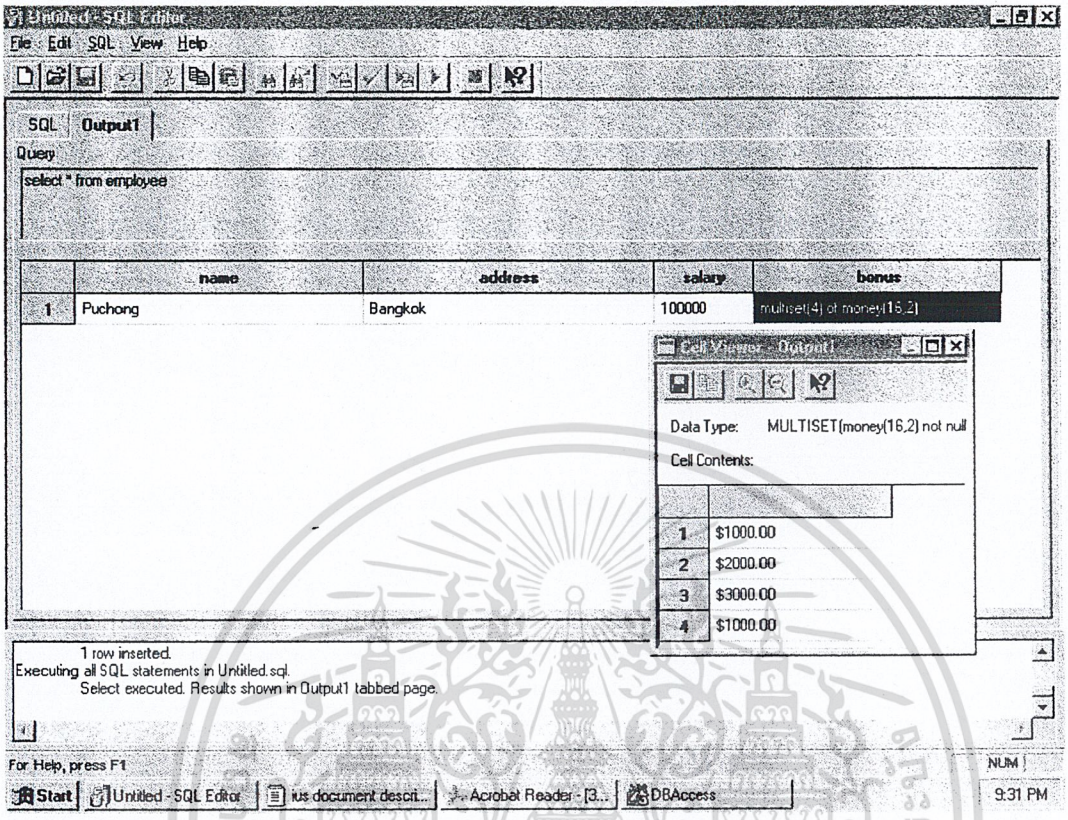
(Name CHAR(30),

Address CHAR (40),

Salary INTEGER,

Bonus MULTISET(MONEY NOT NULL));

INSERT INTO Employee VALUES ('Puchong','Bangkok',100000,"multiset{1000,2000,3000,1000}")



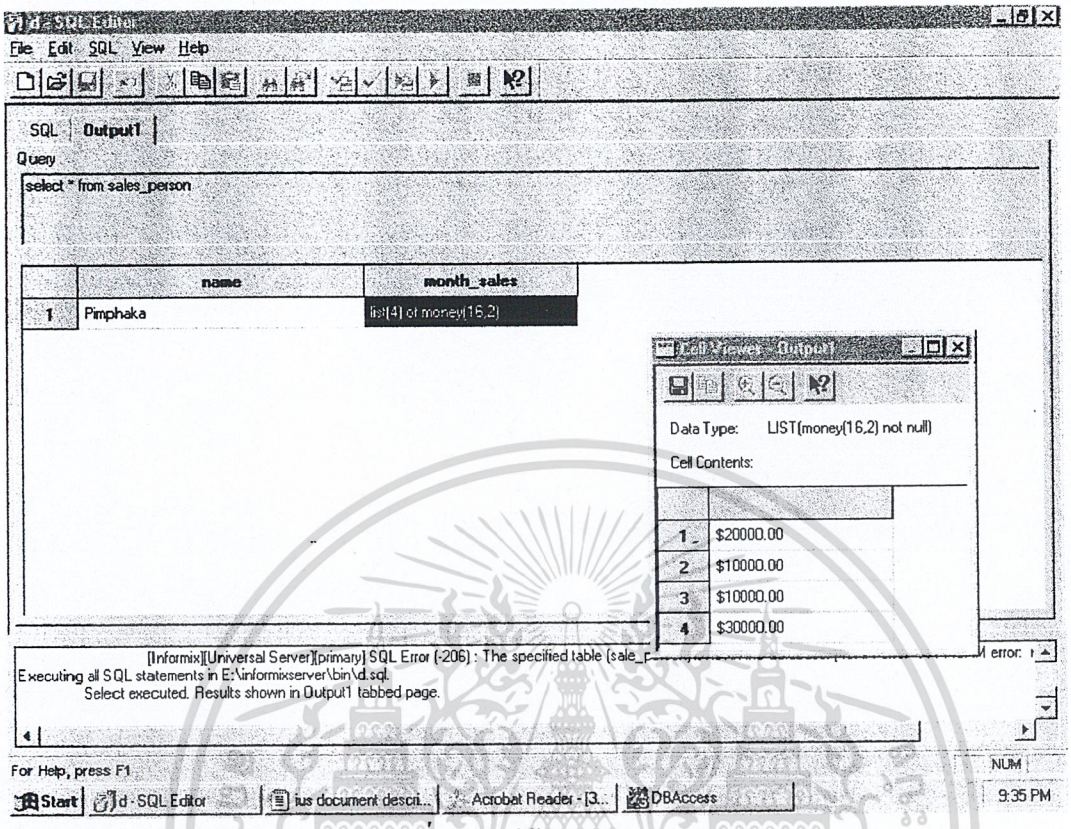
รูปที่ 5ก การใช้งาน Collection Type (Multiset)

CREATE TABLE Sales_person

(Name CHAR(30),

Month_sales LIST(MONEY NOT NULL));

INSERT INTO Sales_person VALUES ('Pimphaka',"list{20000,10000,10000,30000}")



รูปที่ 6ก การใช้งาน Collection Type (List)

Inheritance

CREATE ROW TYPE Person_t

(Name VARCHAR(30) NOT NULL,

Address VARCHAR(20),

City VARCHAR(20),

State CHAR(2),

Zip INTEGER,

Bdate DATE);

CREATE ROW TYPE Employee_t

(Salary INTEGER,

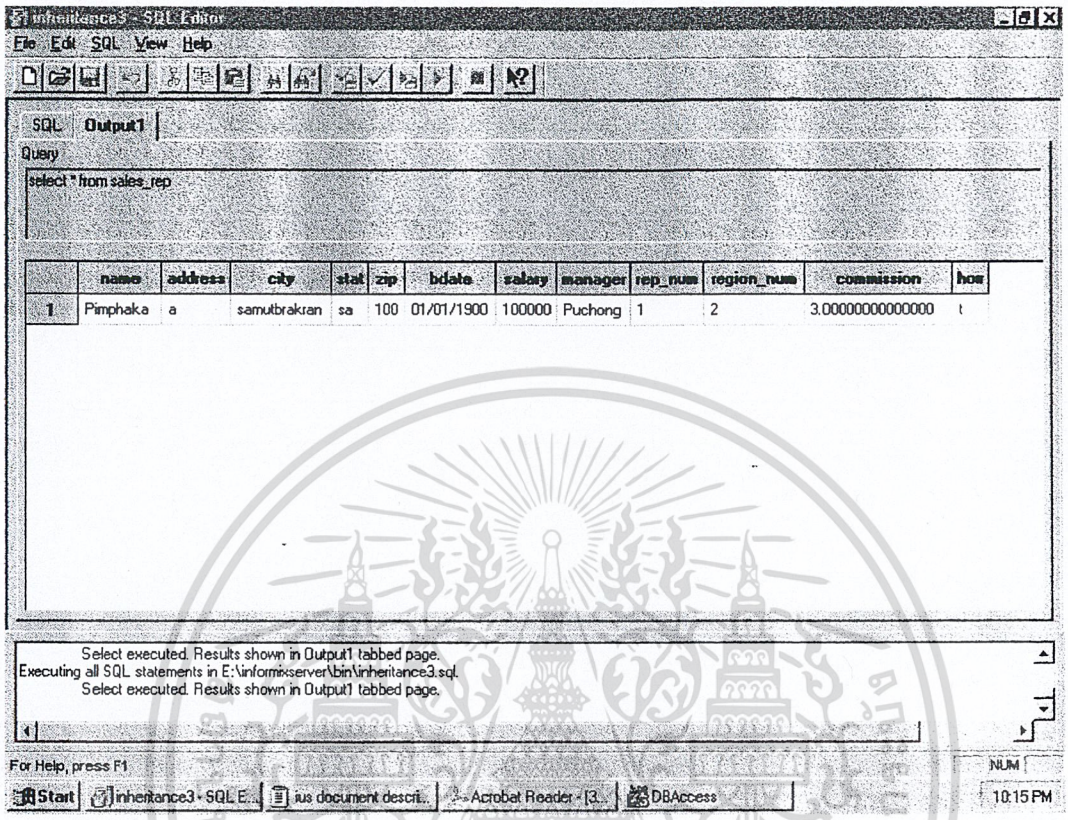
Manager VARCHAR(30))

UNDER Person_t;

```
CREATE ROW TYPE Sales_rep_t
(Rep_num        INT8,
Region_num      INTEGER,
Commission      DECIMAL,
Home_office     BOOLEAN)
UNDER Employee_t;

CREATE ROW TYPE Customer_t
(Cust_num       INTEGER)
UNDER Person_t;

CREATE TABLE Prson OF TYPE Prson_t;

CREATE TABLE Eployee OF TYPE Eployee_t
UNDER Prson;

CREATE TABLE Sales_rep OF TYPE Sales_rep_t
UNDER Employee;

CREATE TABLE Customer OF TYPE Customer_t
UNDER Person;
```

INSERT INTO Sales_rep

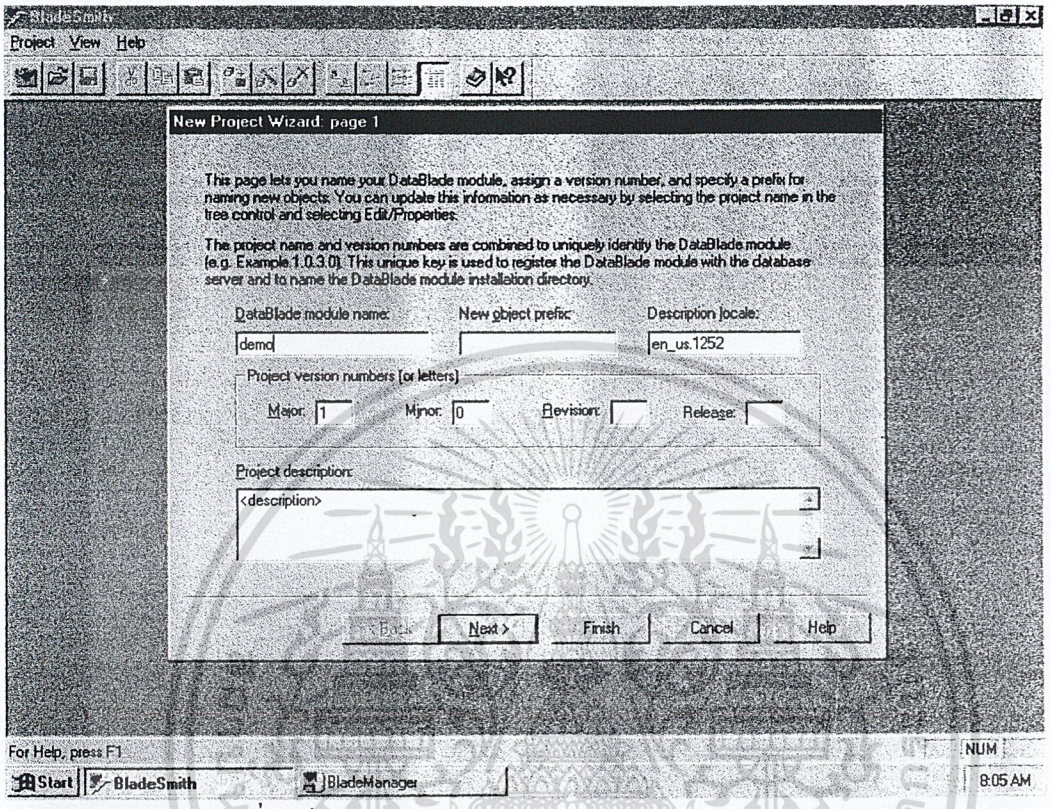
VALUES ('Pimphaka','a','samutprakran','samutphakran',100,1,100000,'Puchong',1,2,3,'t');



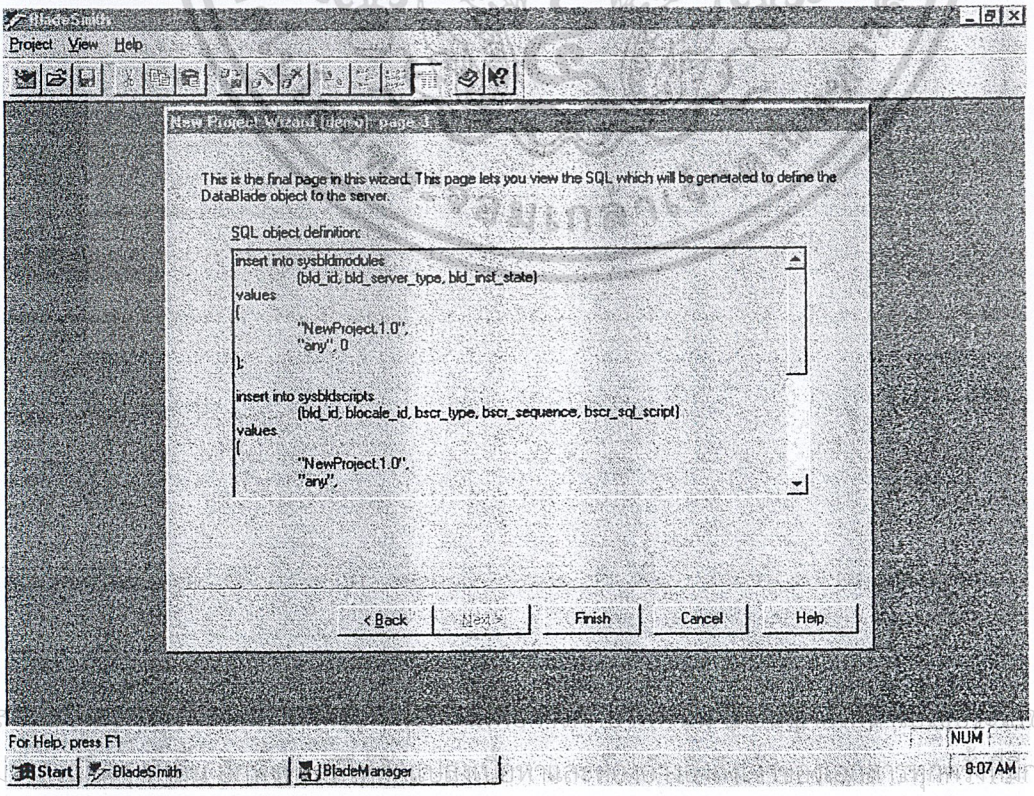
รูป 7ก การ Inheritance

ภาคผนวก ข. ขั้นตอนการสร้าง Opaque Type โดยใช้ DataBlade Developer Kit

1.เริ่มต้นด้วยการใช้ BladeSmith สร้าง Source File โดย เลือกที่เมนู Project-New ใส่ชื่อDataBlade Module และ Description ถ้ามี คลิก Next

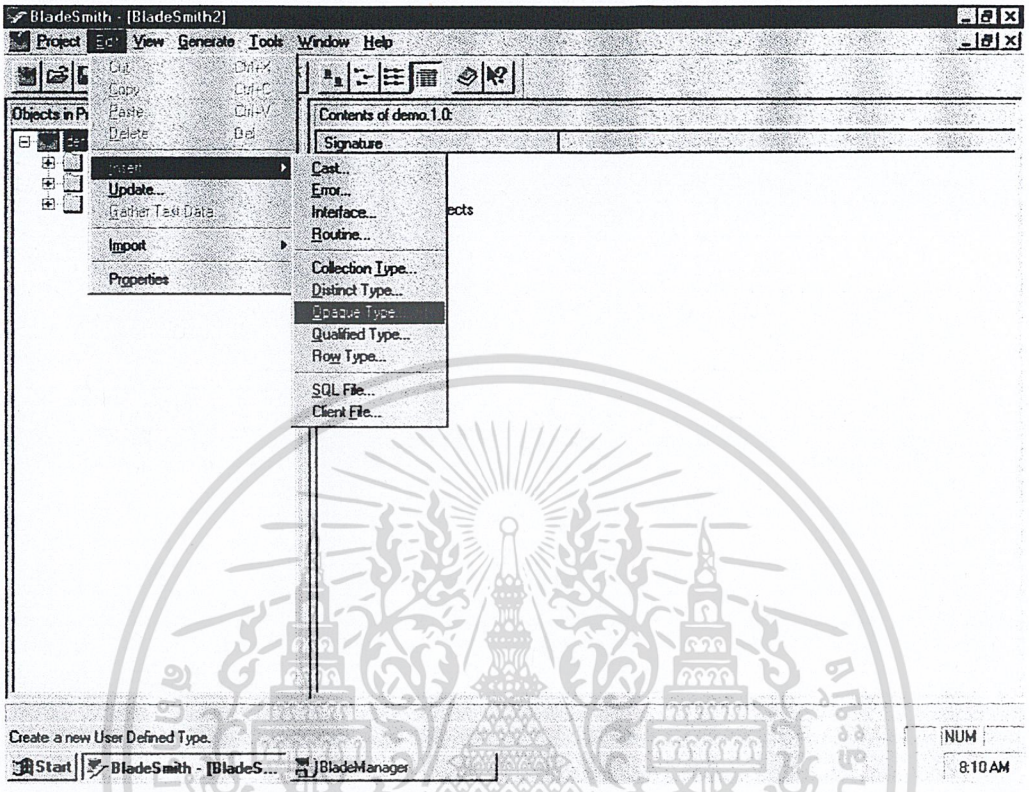


2.ใส่รายละเอียดตามที่มีให้กรอก คลิก Next จน Finish BladeSmith จะทำการ Generate Source Code ที่จำ

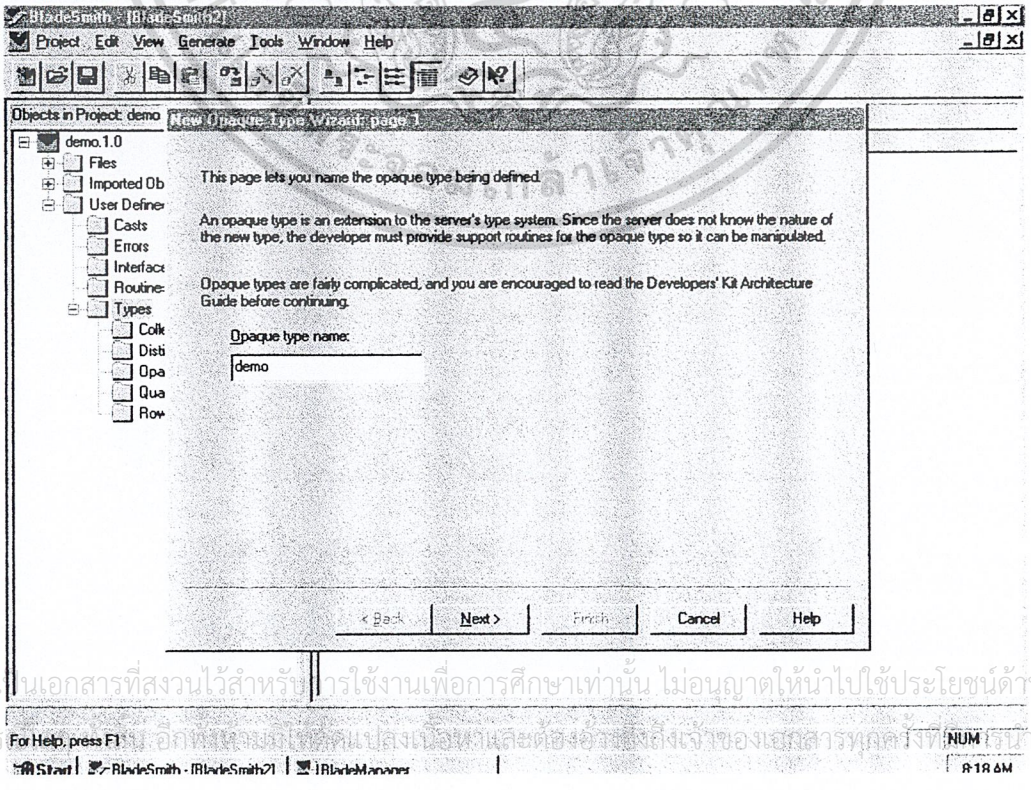


เป็นสำหรับการสร้าง DataBlade Module ให้

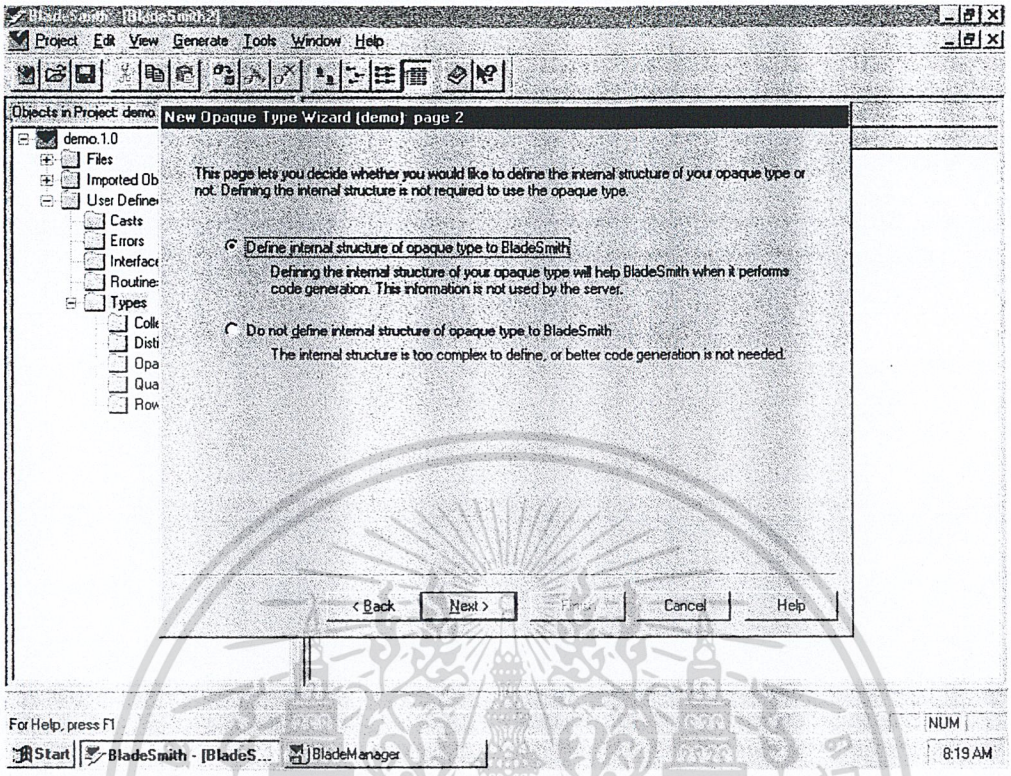
3.เลือกเมนู Edit-Insert เลือกรายการที่ต้องการจะสร้างในที่นี้เลือก Opaque Type



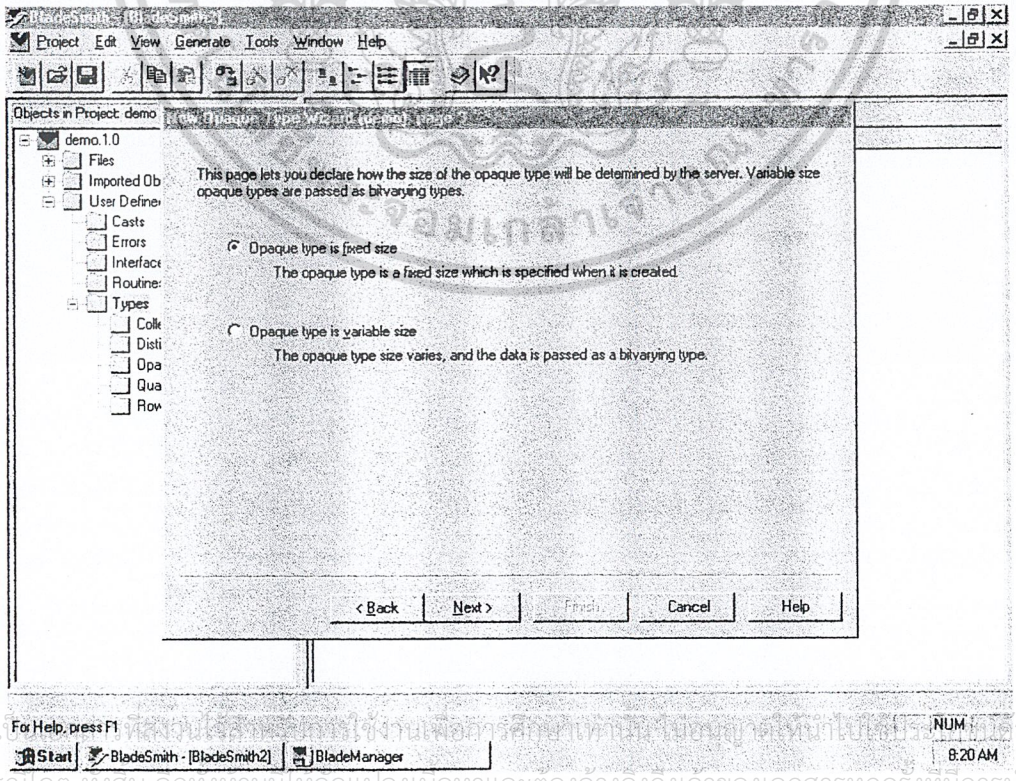
4.กำหนดชื่อ Opaque Type,คลิก Next



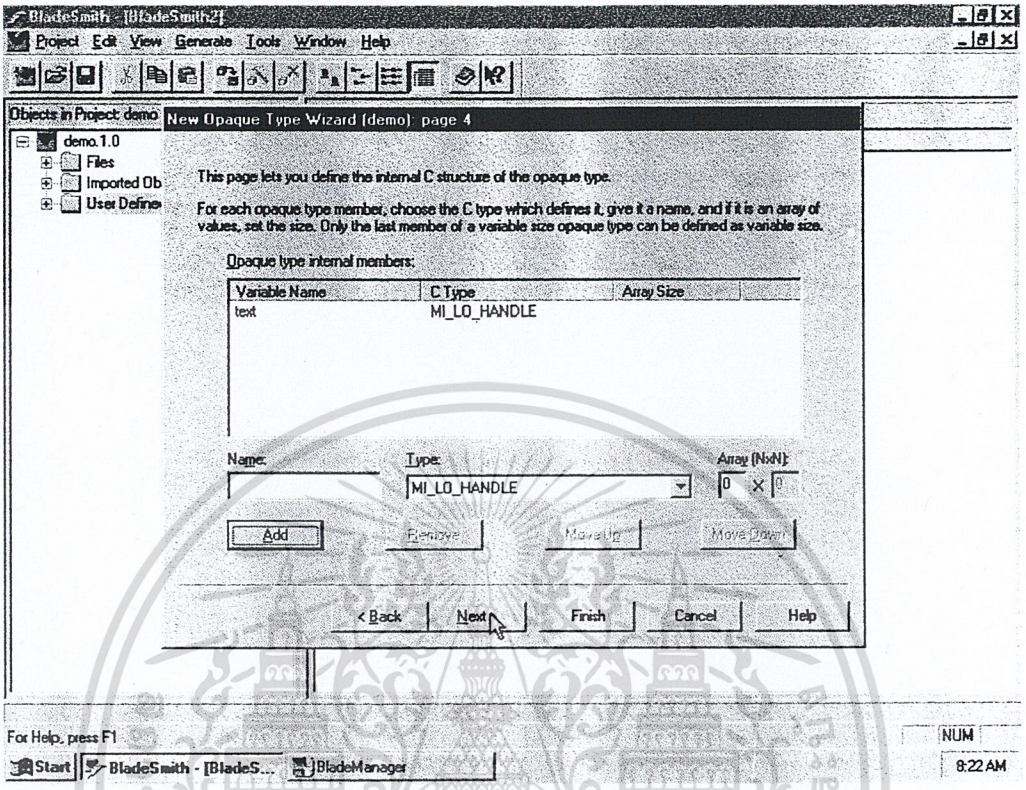
5.เลือกว่าจะให้ BladeSmith สร้าง Internal Structure ให้หรือไม่ ,คลิก Next



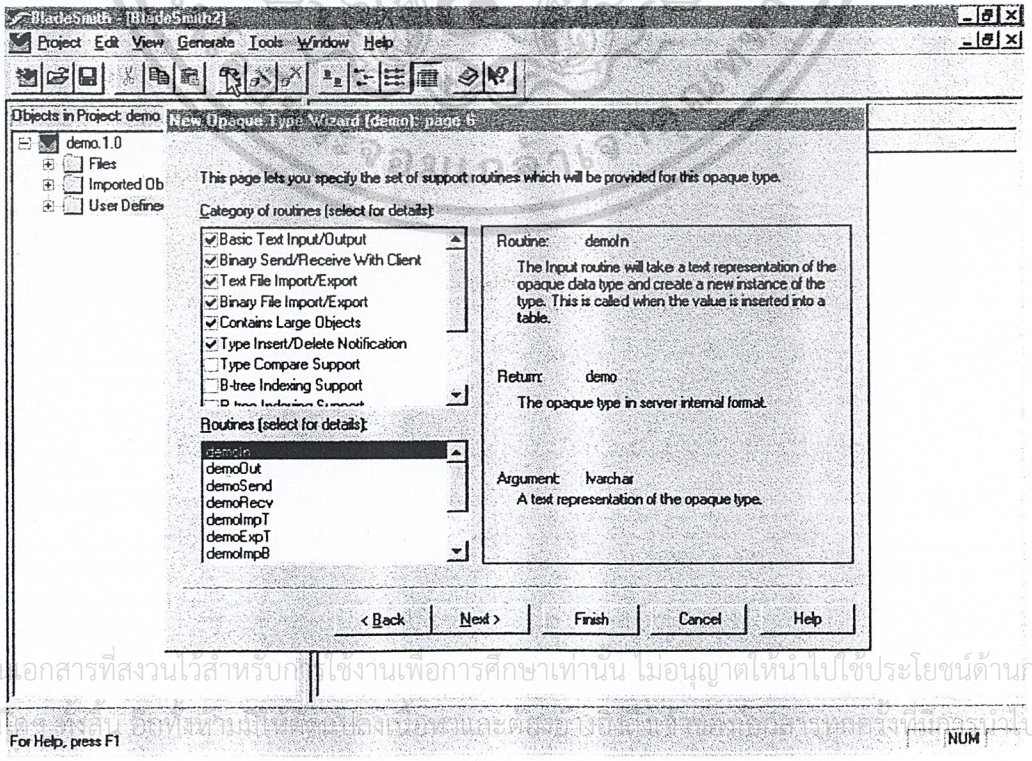
6.เลือกว่าต้องการสร้าง Opaque Type แบบใด Fixed Length หรือ Variable Length,คลิก Next



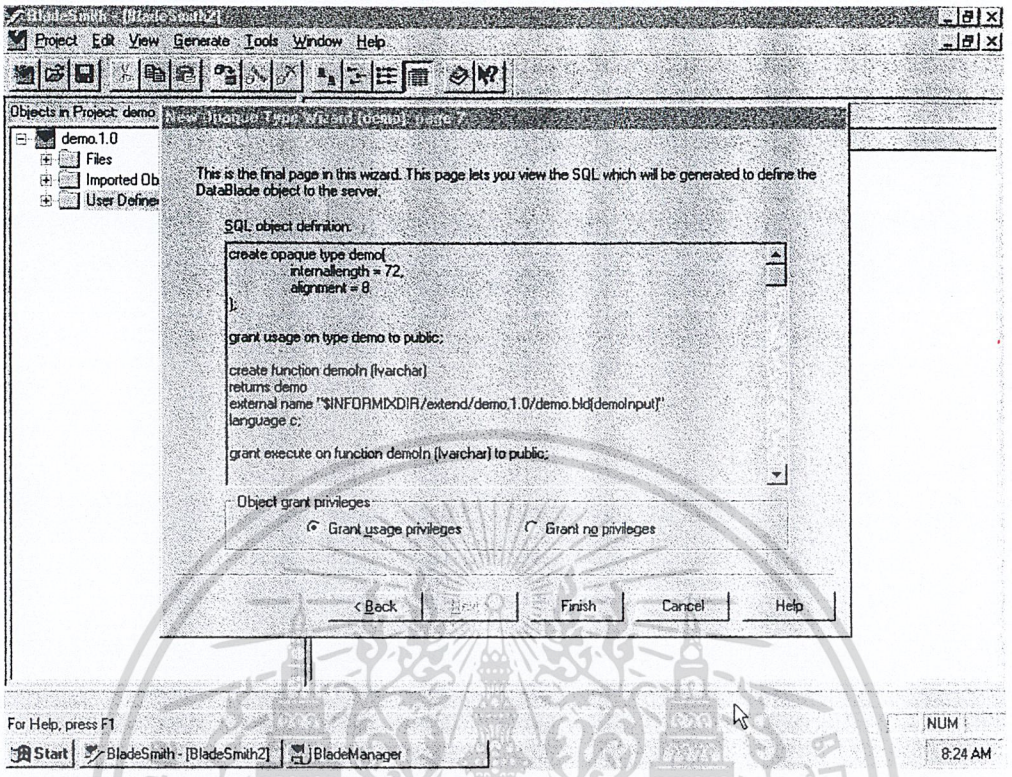
7.ทำการกำหนดว่าใน Opaque Type จะมี Data Field อะไรบ้างโดยใส่ชื่อ ชนิดของข้อมูล แล้วคลิก Add จนได้ Data Field ครบตามที่ต้องการ,คลิก Next



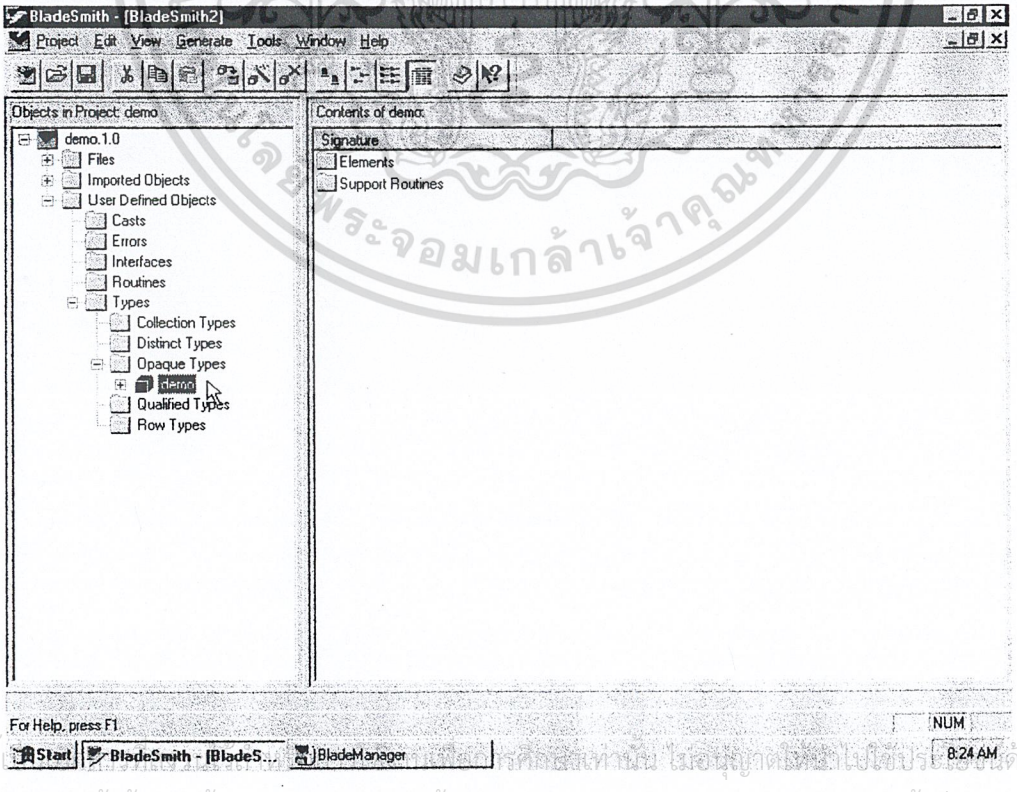
8.เลือกว่าต้องการให้ Opaque Type มี Support Routine อะไรบ้างเมื่อครบตามที่ต้องการ คลิก Next



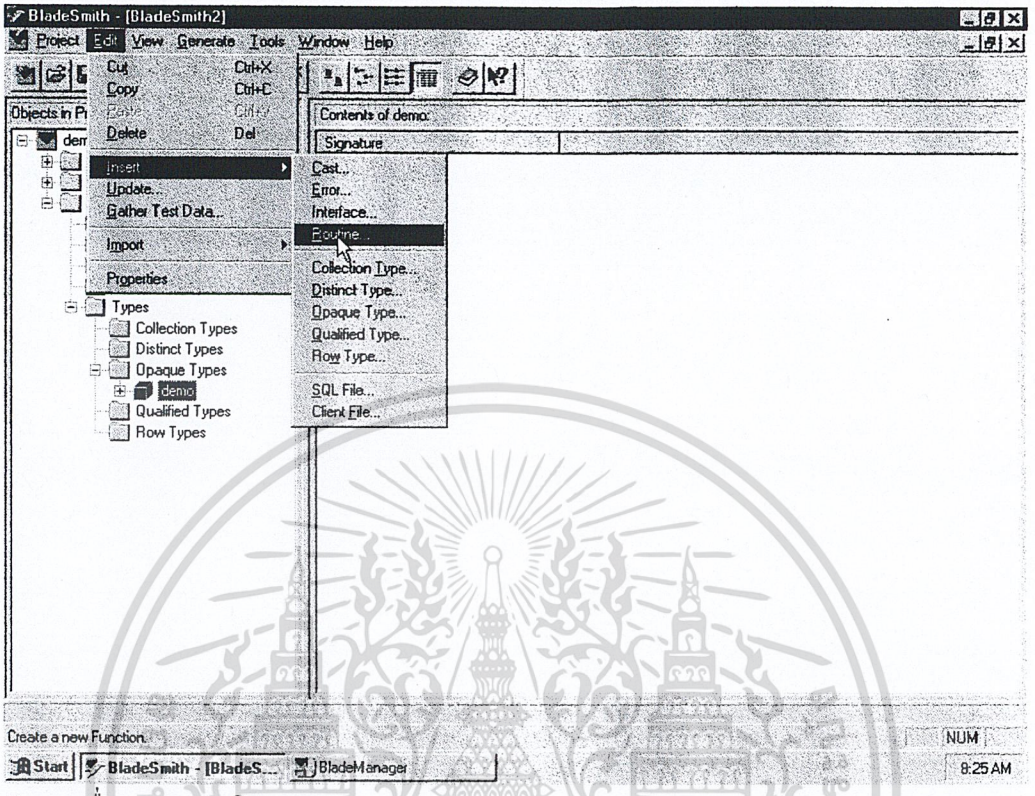
9.เลือกการกำหนดสิทธิในการ Grant,คลิก Finish



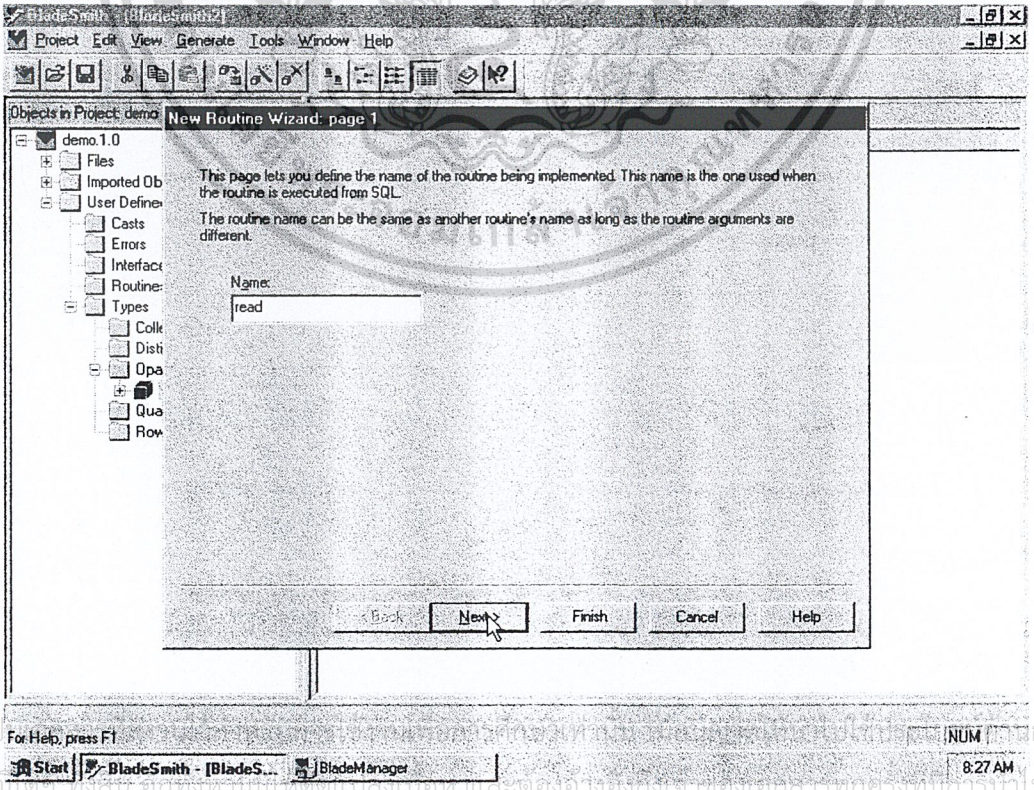
10.Opqaue Type จะมาเก็บที่ Create ไว้ใน Directory ตาม Default ที่กำหนดไว้



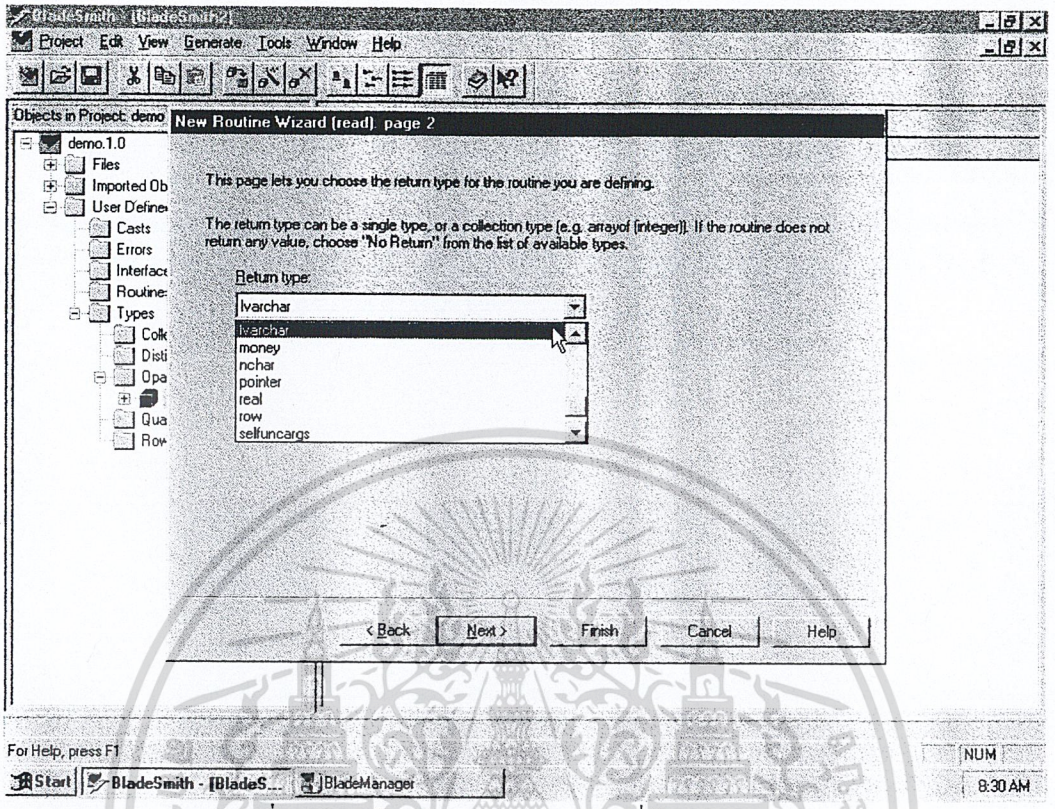
11.การสร้าง Routine เพื่อใช้งานกับ DataBlade Module ที่สร้างขึ้นซึ่งในที่นี้คือ Opaque Type ทำได้โดย เลือกเมนู Edit-Insert-Routine



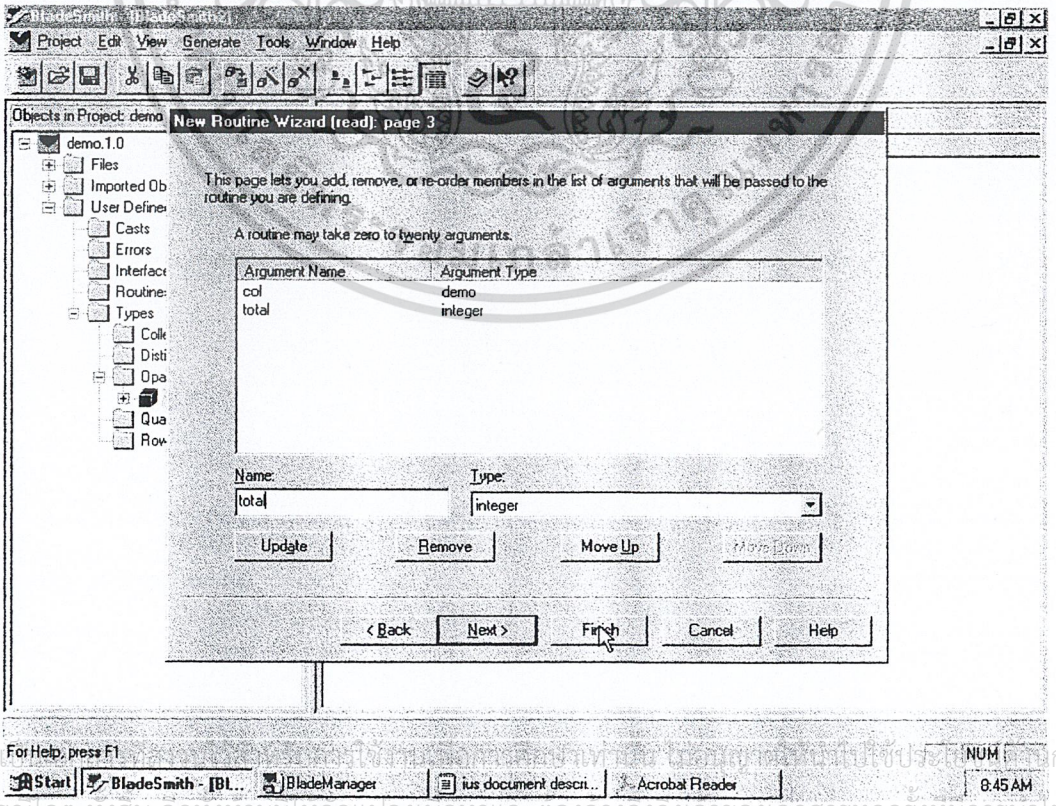
12.กำหนดชื่อ Routine,คลิก Next



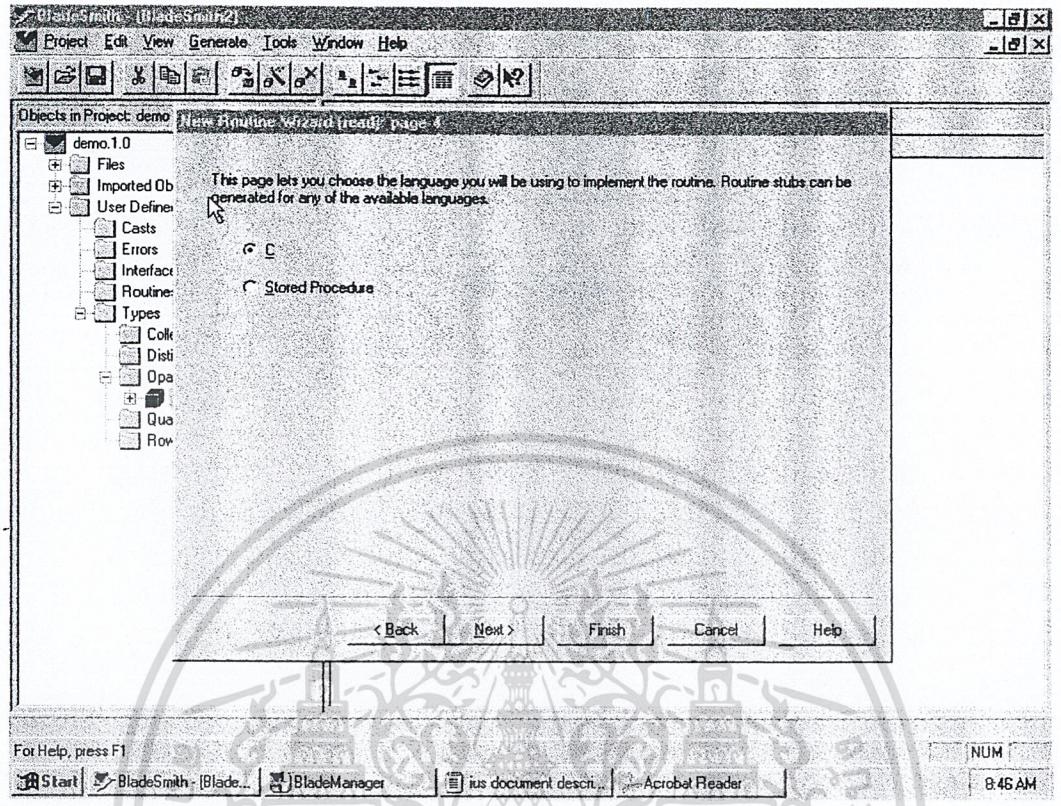
13.เลือก Return Data Type,คลิก Next



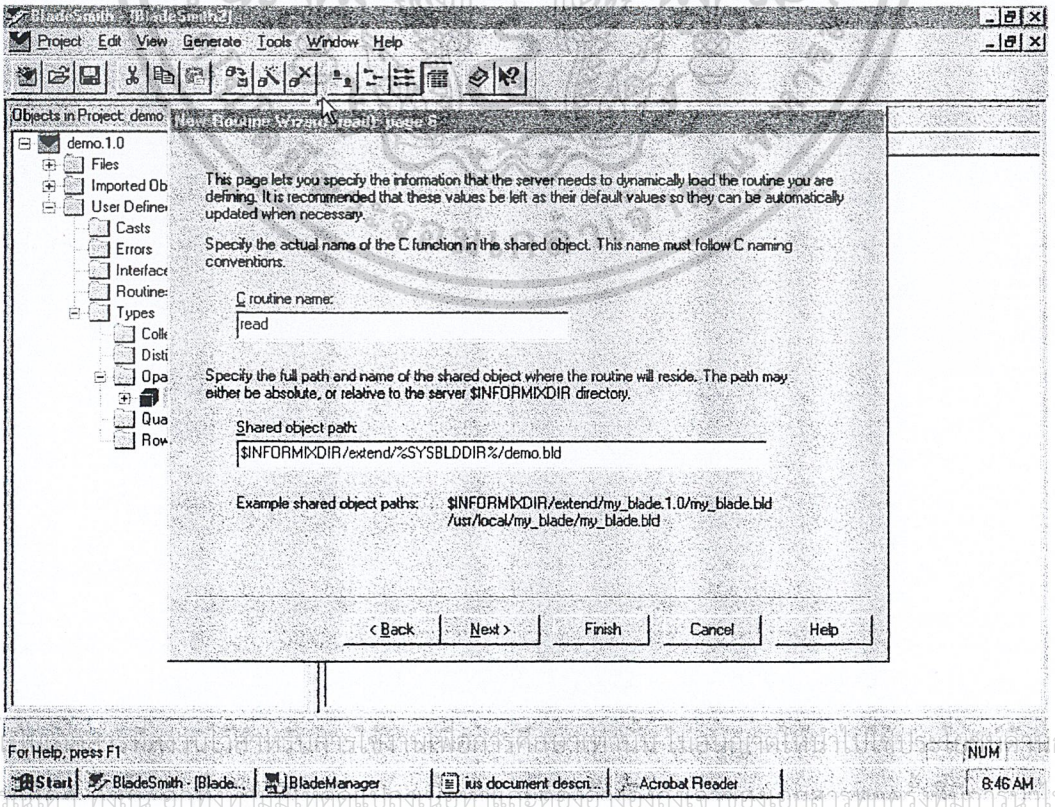
14. กำหนด Parameter ใส่ชื่อ ชนิด แล้วคลิก Add จนครบตามที่ต้องการ,คลิก Next



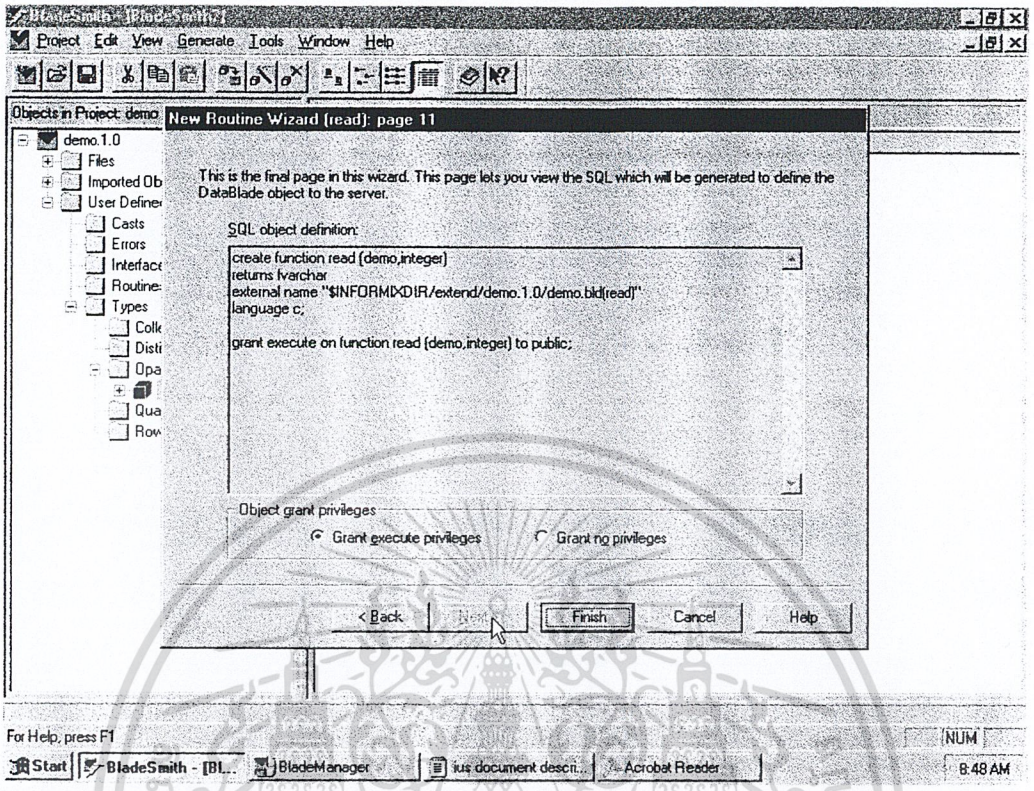
15.เลือกว่าจะใช้ C Language หรือใช้ SQL,คลิก Next



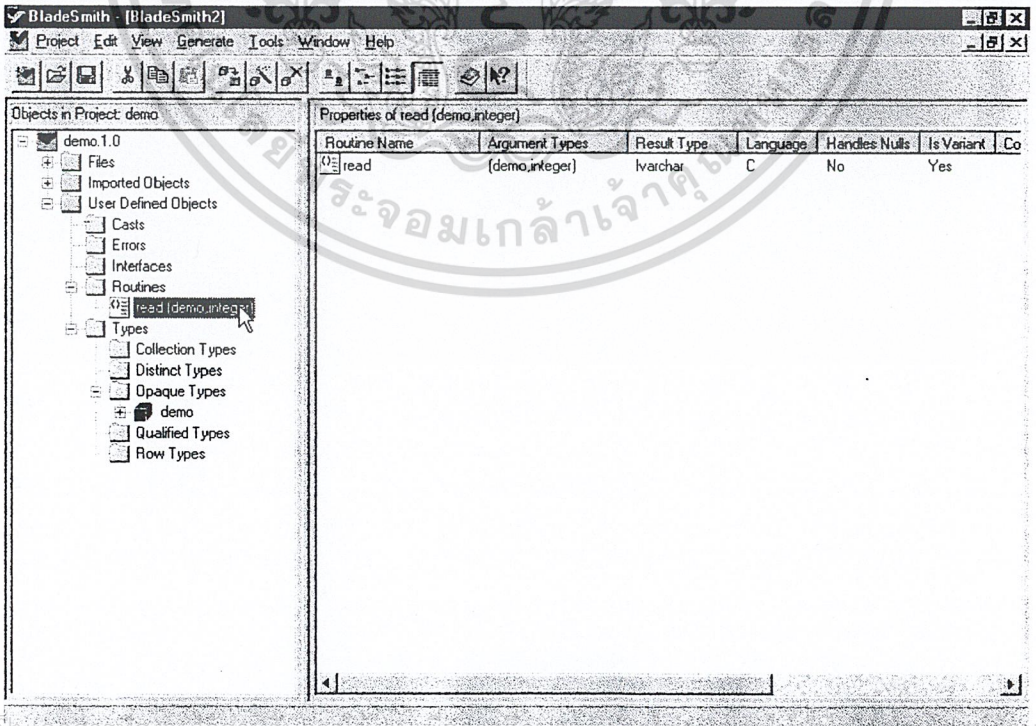
16.ใส่ชื่อ Routine และ Shared Object Directory ที่เก็บ DataBlade Module,คลิก Next



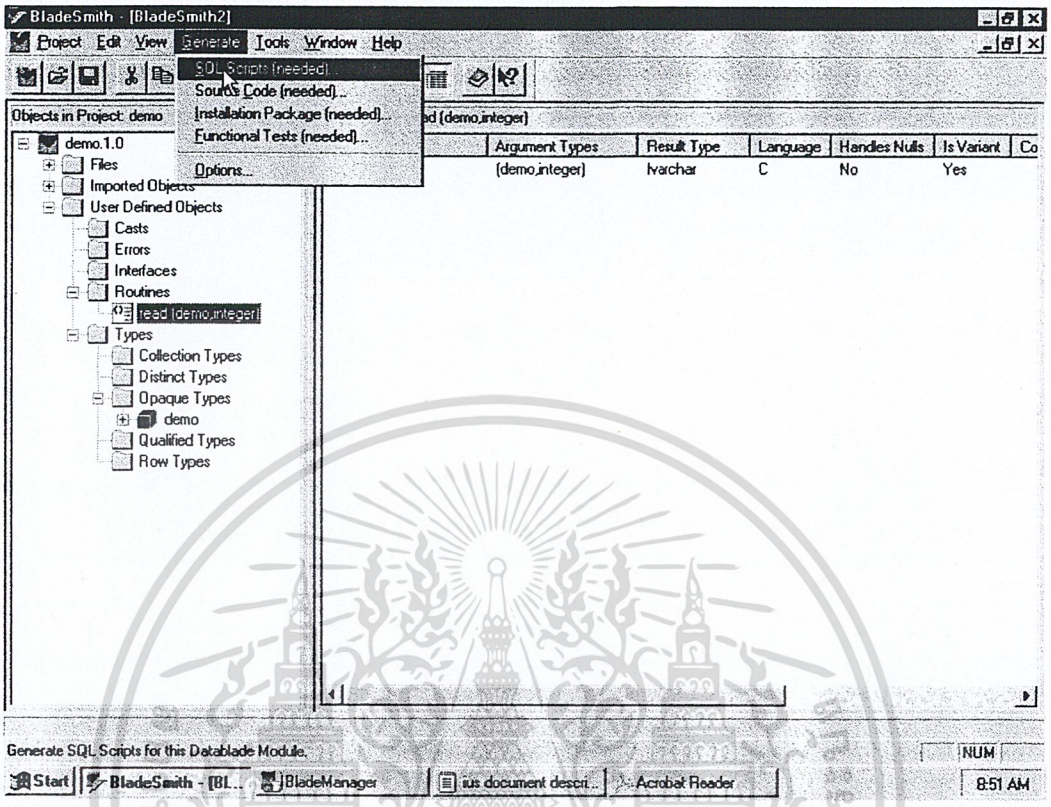
17.เลือกการกำหนดสิทธิในการ Grant,คลิก Finish



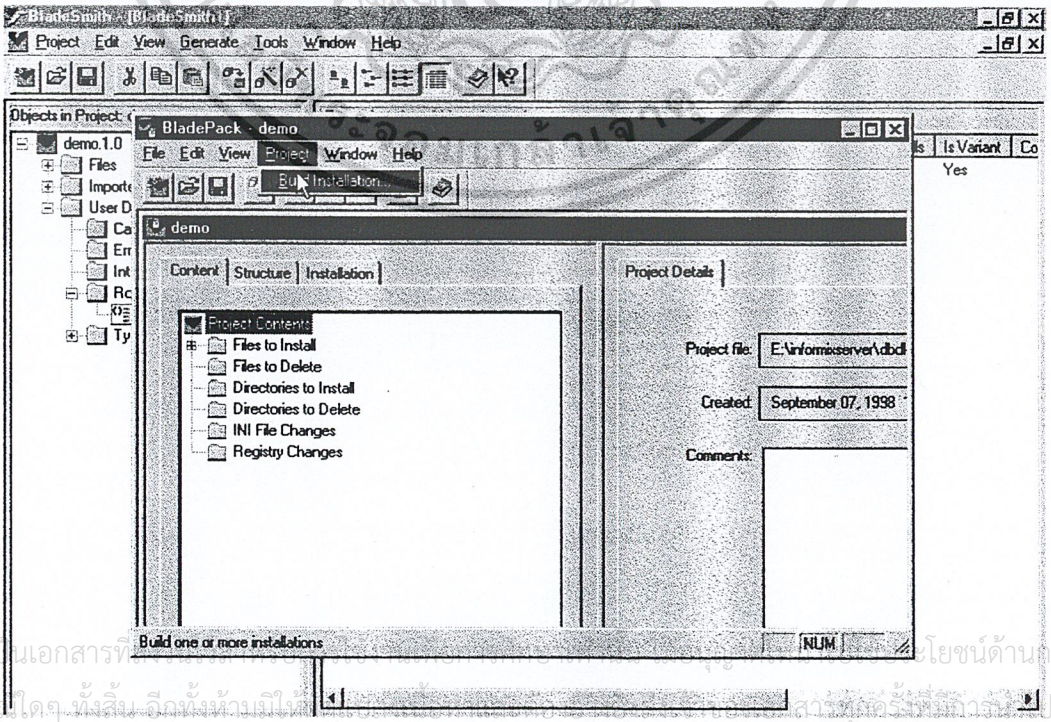
18.Routine ที่สร้างจะถูกเก็บไว้ใน Directory ตาม Default ที่กำหนดไว้



19.ต่อมาเป็นการเริ่ม Generate Source File เริ่มจาก SQL Script ,Source Code,Installation Package และ Functional Test

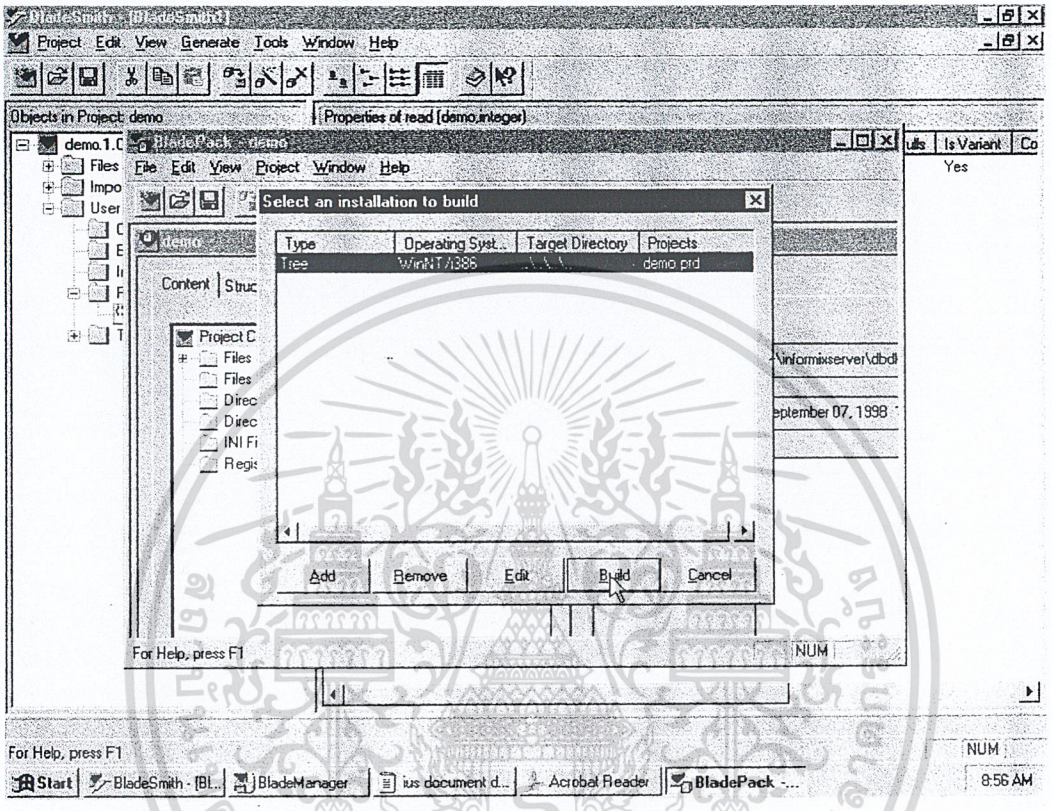


20.ใช้ BladePack เตรียมการ Installation DataBlade Module โดยเรียก BladePack ผ่าน BladeSmith ที่เมนู Tool-BladePack เราสามารถเรียก BladePack โดยไม่ต้องผ่าน BladeSmith ได้เช่นกัน แต่ต้องมีการ Open การ



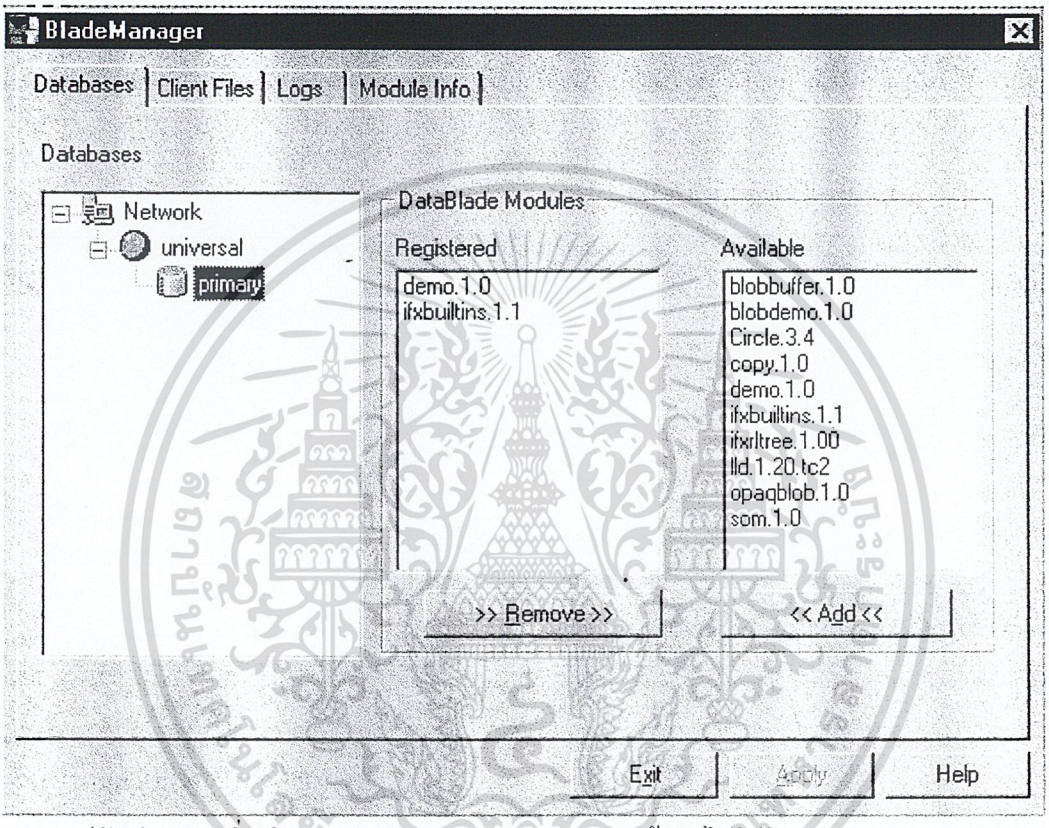
เรียกผ่าน BladeSmith จะทำได้ง่ายและรวดเร็วกว่า เมื่อเข้ามาใน BladePack แล้วเลือก Project-Build Installation

21.คลิก Add เลือกชนิดของการ Install คลิก Build เสร็จการเตรียม Installable DataBlade Module



22.ใช้ BladeManager ทำการ Install DataBlade Module เรียก BladeManager ผ่าน BladeSmith ที่เมนู Tool BladeManager เราสามารถเรียก BladeManager โดยไม่ต้องผ่าน BladeSmith ได้เช่นกัน เมื่อเข้ามาใน BladeManager แล้วเลือก Database Server ที่ต้องการจะเก็บ DataBlade Module เลือก DataBlade Module คลิก Add DataBlade Module ที่สร้าง คลิก Apply เสร็จการ Installation DataBlade Module

23.หากต้องการแก้ไข Source Code ที่ทำการ Generate ขึ้นมาให้ใช้ Visual c++ เปิด Project File .MAK แล้ว



ทำการปรับปรุงและเพิ่มเติม Source Code ทำการ Complie แล้วนำไปใส่ใน Share Object Directory

ภาคผนวก ค. SOURCE CODE

Stored Procedure Routine

```
create function checkinput (l int)
        returning char;
        define inp collection;
        define filter set(int not null);
        define ret int;
        define n int;
        define count int;

        foreach
                select input into inp from andgate where gid in (select gid from layout where lid = l)
                foreach
                        select  * into n from table(inp)
                        let filter = "set{}";
                        foreach
                                select gid into ret from andgate where n in output
                                insert into table(filter) values(ret);
                        end foreach;
                        foreach
                                select gid into ret from orgate where n in output
                                insert into table(filter) values(ret);
                        end foreach;
                        foreach
                                select gid into ret from notgate where n in output
                                insert into table(filter) values(ret);
                        end foreach;
                        foreach
                                select gid into ret from nandgate where n in output
                                insert into table(filter) values(ret);
```

```
                end foreach;
                foreach
                        select gid into ret from norgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xnorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                if (select isoutput from wired where wid = n) = 'I' then let count = 1;
                else let count = 0;
                end if;
                foreach
                        select * into ret from table(filter)
                        let count = count + 1;
                        if count > 1 then return 'N';
                        end if;
                end foreach;
        end foreach;
end foreach;
foreach
        select input into inp from orgate where gid in (select gid from layout where lid = l)
        foreach
                select  * into n from table(inp)
                let filter = "set{}";
                foreach
                        select gid into ret from andgate where n in output
```

```
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from orgate where n in output
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from notgate where n in output
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from nandgate where n in output
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from norgate where n in output
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from xorgate where n in output
            insert into table(filter) values(ret);
end foreach;
foreach
            select gid into ret from xnorgate where n in output
            insert into table(filter) values(ret);
end foreach;
if (select isoutput from wired where wid = n) = 'I' then let count = 1;
else let count = 0;
end if;
foreach
            select * into ret from table(filter)
```

```
                    let count = count + 1;
                    if count > 1 then return 'N';
                    end if;
                end foreach;
            end foreach;
        end foreach;
        foreach
            select input into inp from notgate where gid in (select gid from layout where lid = 1)
            foreach
                select  * into n from table(inp)
                let filter = "set{}";
                foreach
                    select gid into ret from andgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from orgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from notgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from nandgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from norgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
```

```
                foreach
                        select gid into ret from xorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xnorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                if (select isoutput from wired where wid = n) = 'I' then let count = 1;
                else let count = 0;
                end if;
                foreach
                        select * into ret from table(filter)
                        let count = count + 1;
                        if count > 1 then return 'N';
                        end if;
                end foreach;
        end foreach;
end foreach;
foreach
        select input into inp from nandgate where gid in (select gid from layout where lid = l)
        foreach
                select  * into n from table(inp)
                let filter = "set{}";
                foreach
                        select gid into ret from andgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from orgate where n in output
                        insert into table(filter) values(ret);
```

```
        end foreach;
        foreach
                select gid into ret from notgate where n in output
                insert into table(filter) values(ret);
        end foreach;
        foreach
                select gid into ret from nandgate where n in output
                insert into table(filter) values(ret);
        end foreach;
        foreach
                select gid into ret from norgate where n in output
                insert into table(filter) values(ret);
        end foreach;
        foreach
                select gid into ret from xorgate where n in output
                insert into table(filter) values(ret);
        end foreach;
        foreach
                select gid into ret from xnorgate where n in output
                insert into table(filter) values(ret);
        end foreach;
        if (select isoutput from wired where wid = n) = 'I' then let count = 1;
        else let count = 0;
        end if;
        foreach
                select * into ret from table(filter)
                let count = count + 1;
                if count > 1 then return 'N';
                end if;
        end foreach;
end foreach;
```

```
end foreach;
foreach
        select input into inp from norgate where gid in (select gid from layout where lid = 1)
        foreach
                select  * into n from table(inp)
                let filter = "set{}";
                foreach
                        select gid into ret from andgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from orgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from notgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from nandgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from norgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
```

```
                    select gid into ret from xnorgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                if (select isoutput from wired where wid = n) = 'I' then let count = 1;
                else let count = 0;
                end if;
                foreach
                    select * into ret from table(filter)
                    let count = count + 1;
                    if count > 1 then return 'N';
                    end if;
                end foreach;
            end foreach;
        end foreach;
        foreach
            select input into inp from xorgate where gid in (select gid from layout where lid = l)
            foreach
                select  * into n from table(inp)
                let filter = "set{}";
                foreach
                    select gid into ret from andgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from orgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
                foreach
                    select gid into ret from notgate where n in output
                    insert into table(filter) values(ret);
                end foreach;
```

```
                foreach
                        select gid into ret from nandgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from norgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                foreach
                        select gid into ret from xnorgate where n in output
                        insert into table(filter) values(ret);
                end foreach;
                if (select isoutput from wired where wid = n) = 'I' then let count = 1;
                else let count = 0;
                end if;
                foreach
                        select * into ret from table(filter)
                        let count = count + 1;
                        if count > 1 then return 'N';
                        end if;
                end foreach;
        end foreach;
end foreach;
foreach
        select input into inp from xnorgate where gid in (select gid from layout where lid = l)
        foreach
                select  * into n from table(inp)
```

```
let filter = "set{}";
foreach
        select gid into ret from andgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from orgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from notgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from nandgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from norgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from xorgate where n in output
        insert into table(filter) values(ret);
end foreach;
foreach
        select gid into ret from xnorgate where n in output
        insert into table(filter) values(ret);
end foreach;
if (select isoutput from wired where wid = n) = 'I' then let count = 1;
else let count = 0;
```

```
                    end if;
                    foreach
                        select * into ret from table(filter)
                        let count = count + 1;
                        if count > 1 then return 'N';
                        end if;
                    end foreach;
                end foreach;
            end foreach;
            return 'Y';
end function;
```

```
create function checkloop (l int)
            returning char;
            define ret int;
            define ret1 int;
            define ret2 int;
            define ret3 int;
            define work1 set(int not null);
            define path set(int not null);
            define iswork char;
            define temp collection;
            define temp1 set(int not null);
            define count int;
            define tempwork int;

            foreach
                select gid into ret from layout where lid = l
                let path = "set{}";
                let work1 =  "set{}";
                insert into table(work1) values(ret);
```

```
let iswork = 'Y';
while(iswork = 'Y')
        let iswork = 'N';
        let count = 0;
        let temp1 = "set{}";
        foreach
                select * into tempwork from table(work1)
                let count = count +1;
                let iswork = 'Y';
                if count > 1 then
                        insert into table(temp1) values(tempwork);
                elif count = 1 then
                        let ret1 = tempwork;
                end if;
        end foreach;
        if iswork = 'N' then continue while;
        end if;
        delete from table(work1);
        foreach
                select * into tempwork from table(temp1)
                insert into table(work1) values(tempwork);
        end foreach;
        select output into temp from andgate where gid = ret1;
        if temp is null then
                select output into temp from orgate where gid = ret1;
                if temp is null then
                        select output into temp from notgate where gid = ret1;
                        if temp is null then
                                select output into temp from nandgate where gid = ret1;
                                if temp is null then
```

```
                                        select output into temp from norgate where gid =
ret1;
                                        if temp is null then
                                            select output into temp from xorgate
where gid = ret1;
                                            if temp is null then
                                                select output into temp from
xnorgate where gid = ret1;
                                            end if;
                                        end if;
                                    end if;
                                end if;
                            end if;
                        end if;
                    end if;
                foreach
                    select * into ret2 from table(temp)
                    foreach
                        select gid into ret3 from andgate where ret2 in input
                        if ret3 is not null and ret3 not in (select * from table(path)) then
                            insert into table(path) values(ret3);
                            insert into table(work1) values(ret3);
                        end if;
                    end foreach;
                    foreach
                        select gid into ret3 from orgate where ret2 in input
                        if ret3 is not null and ret3 not in (select * from table(path)) then
                            insert into table(path) values(ret3);
                            insert into table(work1) values(ret3);
                        end if;
                    end foreach;
                    foreach
```

```
            select gid into ret3 from notgate where ret2 in input
            if ret3 is not null and ret3 not in (select * from table(path)) then
                  insert into table(path) values(ret3);
                  insert into table(work1) values(ret3);
            end if;
      end foreach;
      foreach
            select gid into ret3 from nandgate where ret2 in input
            if ret3 is not null and ret3 not in (select * from table(path)) then
                  insert into table(path) values(ret3);
                  insert into table(work1) values(ret3);
            end if;
      end foreach;
      foreach
            select gid into ret3 from norgate where ret2 in input
            if ret3 is not null and ret3 not in (select * from table(path)) then
                  insert into table(path) values(ret3);
                  insert into table(work1) values(ret3);
            end if;
      end foreach;
      foreach
            select gid into ret3 from xorgate where ret2 in input
            if ret3 is not null and ret3 not in (select * from table(path)) then
                  insert into table(path) values(ret3);
                  insert into table(work1) values(ret3);
            end if;
      end foreach;
      foreach
            select gid into ret3 from xnorgate where ret2 in input
            if ret3 is not null and ret3 not in (select * from table(path)) then
                  insert into table(path) values(ret3);
```

```
                                        insert into table(work1) values(ret3);
                                    end if;
                                end foreach;
                            end foreach;
                            if ret in (select * from table(path)) then return 'N';
                            end if;
                        end while;
                end foreach;
                return 'Y';
end function;
```

```
create function clearlayout (l int)
        returning int;
        define ret int;

        foreach
                execute function getwid(l) into ret
                delete from wired where wid = ret;
        end foreach;
        delete from andgate where gid in (select gid from layout where lid = l);
        delete from orgate where gid in (select gid from layout where lid = l);
        delete from notgate where gid in (select gid from layout where lid = l);
        delete from nandgate where gid in (select gid from layout where lid = l);
        delete from norgate where gid in (select gid from layout where lid = l);
        delete from xorgate where gid in (select gid from layout where lid = l);
        delete from xnorgate where gid in (select gid from layout where lid = l);
        delete from layout where lid = l;
end function;
```

```
create function genarateand (input1 char,input2 char)
        returning char;
```

```
        if input1 = 'H' and input2 = 'H' then return 'H';
        else return 'L';
        end if
end function;

create function genarateor (input1 char,input2 char)
        returning char;

        if input1 = 'L' and input2 = 'L' then return 'L';
        else return 'H';
        end if
end function;

create function genaratenot (input1 char)
        returning char;

        if input1 = 'H' then return 'L';
        else return 'H';
        end if
end function;

create function genaratenand (input1 char,input2 char)
        returning char;

        if input1 = 'H' and input2 = 'H' then return 'L';
        else return 'H';
        end if
end function;

create function genaratenor (input1 char,input2 char)
```

```
        returning char;

        if input1 = 'L' and input2 = 'L' then return 'H';
        else return 'L';
        end if
end function;

create function genaratexor (input1 char,input2 char)
        returning char;

        if input1 = input2 then return 'H';
        else return 'L';
        end if
end function;

create function genaratexnor (input1 char,input2 char)
        returning char;

        if input1 = input2 then return 'L';
        else return 'H';
        end if
end function;
```

```
create function genaratefinal (l int)
        returning int;
        define n int;
        define count int;
        define ret int;

        let count = 1;
        let ret = 0;
```

```
        while count > 0
                let count = 0;
                execute function getgenand(l) into n;
                let count = count + n;
                execute function getgenor(l) into n;
                let count = count + n;
                execute function getgennot(l) into n;
                let count = count + n;
                execute function getgennand(l) into n;
                let count = count + n;
                execute function getgennor(l) into n;
                let count = count + n;
                execute function getgenxor(l) into n;
                let count = count + n;
                execute function getgenxnor(l) into n;
                let count = count + n;
                let ret = ret + count;
        end while;
        return ret;
end function;
```

```
create function getelement (column list(int not null),number int)
        returning int;
        define n int;
        define count int;

        let count = 0;

        foreach cursor1 for
                select  * into n from table(column)
                let count = count + 1;
```

```
            if count = number then return n;
            end if
        end foreach
end function;
```

```
create function getgenand (l int)
        returning int;
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;

        foreach
            execute function getgidand(l) into n
            select value into input1 from wired where wid = (select getelement(input,1) from andgate
where gid = n);
            select value into input2 from wired where wid = (select getelement(input,2) from andgate
where gid = n);
            execute function genarateand(input1,input2) into output1;
            update wired set value = output1 where wid = (select getelement(output,1) from andgate where
gid = n);
            let have = have + 1;
        end foreach
        return have;
end function;

create function getgenor (l int)
        returning int;
```

```
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;

        foreach
                execute function getgidor(1) into n
                select value into input1 from wired where wid = (select getelement(input,1) from orgate where
gid = n);
                select value into input2 from wired where wid = (select getelement(input,2) from orgate where
gid = n);
                execute function genarateor(input1,input2) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from orgate where
gid = n);
                let have = have + 1;
        end foreach
        return have;
end function;

create function getgennot (1 int)
        returning int;
        define n int;
        define input1 char;
        define output1 char;
        define have int;

        let have = 0;
```

```
        foreach
                execute function getgidnot(l) into n
                select value into input1 from wired where wid = (select getelement(input,1) from notgate
where gid = n);
                execute function genaratenot(input1) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from notgate where
gid = n);
                let have = have + 1;
        end foreach
        return have;
end function;

create function getgennand (l int)
        returning int;
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;

        foreach
                execute function getgidnand(l) into n
                select value into input1 from wired where wid = (select getelement(input,1) from nandgate
where gid = n);
                select value into input2 from wired where wid = (select getelement(input,2) from nandgate
where gid = n);
                execute function genaratenand(input1,input2) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from nandgate
where gid = n);
```

```
            let have = have + 1;
        end foreach
        return have;
end function;

create function getgennor (l int)
        returning int;
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;

        foreach
                execute function getgidnor(l) into n
                select value into input1 from wired where wid = (select getelement(input,1) from norgate
where gid = n);
                select value into input2 from wired where wid = (select getelement(input,2) from norgate
where gid = n);
                execute function genaratenor(input1,input2) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from norgate where
gid = n);
                let have = have + 1;
        end foreach
        return have;
end function;

create function getgenxor (l int)
        returning int;
```

```
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;

        foreach
                execute function getgidxor(1) into n
                select value into input1 from wired where wid = (select getelement(input,1) from xorgate
where gid = n);
                select value into input2 from wired where wid = (select getelement(input,2) from xorgate
where gid = n);
                execute function genaratexor(input1,input2) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from xorgate where
gid = n);
                let have = have + 1;
        end foreach
        return have;
end function;

create function getgenxnor (1 int)
        returning int;
        define n int;
        define input1 char;
        define input2 char;
        define output1 char;
        define have int;

        let have = 0;
```

```
        foreach
                execute function getgidxnor(l) into n
                select value into input1 from wired where wid = (select getelement(input,1) from xnorgate
where gid = n);
                select value into input2 from wired where wid = (select getelement(input,2) from xnorgate
where gid = n);
                execute function genaratexnor(input1,input2) into output1;
                update wired set value = output1 where wid = (select getelement(output,1) from xnorgate
where gid = n);
                let have = have + 1;
        end foreach
        return have;
end function;
```

```
create function getgidand (l int)
        returning int;

        define ret int;

        foreach cursor for
                select gid into ret
                from andgate t1
                where not exists  (select wid  from andgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                        and  not exists  (select wid  from andgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                        and gid in (select gid from layout where lid = l)
                return ret with resume;
        end foreach;
end function;
```

```
create function getgidor (l int)
        returning int;

        define ret int;

        foreach cursor for
                select gid into ret
                from orgate t1
                where not exists (select wid from orgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                        and not exists (select wid from orgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                        and gid in (select gid from layout where lid = 1)
                return ret with resume;
        end foreach;
end function;

create function getgidnot (l int)
        returning int;

        define ret int;

        foreach cursor for
                select gid into ret
                from notgate t1
                where not exists (select wid from notgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                        and not exists (select wid from notgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                        and gid in (select gid from layout where lid = 1)
```

```
            return ret with resume;
        end foreach;
end function;

create function getgidnand (l int)
        returning int;

        define ret int;

        foreach cursor for
                select gid into ret
                from nandgate t1
                where not exists  (select wid  from nandgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                        and  not exists  (select wid  from nandgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                        and gid in (select gid from layout where lid = l)
                return ret with resume;
        end foreach;
end function;

create function getgidnor (l int)
        returning int;

        define ret int;

        foreach cursor for
                select gid into ret
                from norgate t1
                where not exists  (select wid  from norgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
```

```
                and  not exists  (select wid  from norgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                and gid in (select gid from layout where lid = l)
            return ret with resume;
        end foreach;
end function;

create function getgidxor (l int)
        returning int;

        define ret int;

        foreach cursor for
            select gid into ret
            from xorgate t1
            where not exists  (select wid  from xorgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                and  not exists  (select wid  from xorgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                and gid in (select gid from layout where lid = l)
            return ret with resume;
        end foreach;
end function;

create function getgidxnor (l int)
        returning int;

        define ret int;

        foreach cursor for
            select gid into ret
```

```
            from xnorgate t1
            where not exists (select wid from xnorgate t2, wired where value is null and wid in input and
t2.gid = t1.gid)
                    and not exists (select wid from xnorgate t3, wired where value is not null and wid in
output and t3.gid = t1.gid)
                    and gid in (select gid from layout where lid = l)
            return ret with resume;
        end foreach;
end function;
```

```
create function getgidorder (number int,l int)
        returning int;
        define ret int;
        define i int;

        let i = 0;
        foreach
            select gid into ret from layout where lid = l order by 1
            let i = i+1;
            if i = number then return ret;
            end if;
        end foreach;
end function;
```

```
create function getnewwid (number int)
        returning int;
        define ret int;
        define count int;

        let count = 0;
        foreach
```

```
            select wid into ret from wired where isoutput is null order by 1
            let count = count +1;
            if count = number then return ret;
            end if;
        end foreach;
end function;
```

```
create function getnumbergate (l int)
        returning int;
        define ret int;

        select count(*) into ret from layout where lid = l;
        return ret;
end function;

create function getnumberwired (l int)
        returning int;
        define temp int;
        define ret int;

        let ret = 0;
        foreach
            execute function getwid(l) into temp
            let ret = ret + 1;
        end foreach;
        return ret;
end function;
```

```
create function getordergid (number int,l int)
        returning int;
```

```
        define ret int;
        define i int;

        let i = 0;
        foreach
                select gid into ret from layout where lid = l order by 1
                let i = i+1;
                if ret = number then return i;
                end if;
        end foreach;
end function;
```

```
create function getorderwid (number int,l int)
        returning int;
        define ret int;
        define i int;

        let i = 0;
        foreach
                execute function getwid(l) into ret
                let i = i+1;
                if ret = number then return i;
                end if;
        end foreach;
end function;
```

```
create function getwid (l int)
        returning int;
        define tempc collection;
        define filter set(int not null);
        define temp int;
```

```
define ret int;

let filter = "set{}";
foreach
        select input into tempc from andgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from andgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select input into tempc from orgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from orgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
```

```
foreach
        select input into tempc from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select input into tempc from nandgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from nandgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select input into tempc from norgate where gid in (select gid from layout where lid = 1)
        foreach
```

```
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from norgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select input into tempc from xorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select output into tempc from xorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
end foreach;
foreach
        select input into tempc from xnorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into temp from table(tempc)
                insert into table(filter) values(temp);
        end foreach;
```

```
        end foreach;
        foreach
                select output into tempc from xnorgate where gid in (select gid from layout where lid = l)
                foreach
                        select * into temp from table(tempc)
                        insert into table(filter) values(temp);
                end foreach;
        end foreach;
        foreach
                select * into ret from table(filter) order by 1
                return ret with resume;
        end foreach;
end function;
```

```
create function getwidoutput (l int)
        returning int;
        define out collection;
        define filter set(int not null);
        define ret int;
        define is char;

        foreach
                select output into out from andgate where gid in (select gid from layout where lid = l)
                foreach
                        select * into ret from table(out)
                        select isoutput into is from wired where wid = ret;
                        if is = 'O' then insert into table(filter) values(ret);
                        end if;
                end foreach;
        end foreach;
        foreach
```

```
        select output into out from orgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(out)
                select isoutput into is from wired where wid = ret;
                if is = 'O' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select output into out from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(out)
                select isoutput into is from wired where wid = ret;
                if is = 'O' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select output into out from nandgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(out)
                select isoutput into is from wired where wid = ret;
                if is = 'O' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select output into out from norgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(out)
                select isoutput into is from wired where wid = ret;
```

```
                    if is = 'O' then insert into table(filter) values(ret);
                    end if;
                end foreach;
            end foreach;
            foreach
                select output into out from xorgate where gid in (select gid from layout where lid = l)
                foreach
                    select * into ret from table(out)
                    select isoutput into is from wired where wid = ret;
                    if is = 'O' then insert into table(filter) values(ret);
                    end if;
                end foreach;
            end foreach;
            foreach
                select output into out from xnorgate where gid in (select gid from layout where lid = l)
                foreach
                    select * into ret from table(out)
                    select isoutput into is from wired where wid = ret;
                    if is = 'O' then insert into table(filter) values(ret);
                    end if;
                end foreach;
            end foreach;
            foreach
                select * into ret from table(filter)
                return ret with resume;
            end foreach;
    end function;

    create function getwidinput (l int)
            returning int;
            define inp collection;
```

```
define filter set(int not null);
define ret int;
define is char;

foreach
        select input into inp from andgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select input into inp from orgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select input into inp from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
```

```
foreach
        select input into inp from nandgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select input into inp from norgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select input into inp from xorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                select isoutput into is from wired where wid = ret;
                if is = 'I' then insert into table(filter) values(ret);
                end if;
        end foreach;
end foreach;
foreach
        select input into inp from xnorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
```

```
                    select isoutput into is from wired where wid = ret;
                    if is = 'T' then insert into table(filter) values(ret);
                    end if;
                end foreach;
        end foreach;
        foreach
                select * into ret from table(filter)
                return ret with resume;
        end foreach;
end function;
```

```
create function getwidorder (number int,l int)
        returning int;
        define ret int;
        define i int;

        let i = 0;
        foreach
                execute function getwid(l) into ret
                let i = i+1;
                if i = number then return ret;
                end if;
        end foreach;
end function;
```

```
create function getwidvalue (number int,l int)
        returning char;
        define ret int;
        define ret1 char;

        execute function getwidorder(number,l) into ret;
```

```
        select value into ret1 from wired where wid = ret;
        return ret1;
end function;
```

```
create function initlayout (number int,l int)
        returning int;
        define ret int;
        define i int;

        select max(gid) into ret from layout;
        if ret is null then let ret = 0;
        end if;
        for i = 1 to number
                insert into layout values(ret+i,null,null,l);
        end for;
end function;
```

```
create function initvalue (l int)
        returning int;
        define inp list(int not null);
        define ret int;

        foreach
                select input into inp from andgate where gid in (select gid from layout where lid = l)
                foreach
                        select * into ret from table(inp)
                        update wired set value = null where wid = ret;
                end foreach;
        end foreach;
        foreach
                select input into inp from orgate where gid in (select gid from layout where lid = l)
```

```
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select input into inp from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select input into inp from nandgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select input into inp from norgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select input into inp from xorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
```

```
        end foreach;
end foreach;
foreach
        select input into inp from xnorgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select output into inp from andgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select output into inp from orgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
        select output into inp from notgate where gid in (select gid from layout where lid = 1)
        foreach
                select * into ret from table(inp)
                update wired set value = null where wid = ret;
        end foreach;
end foreach;
foreach
```

```
            select output into inp from nandgate where gid in (select gid from layout where lid = l)
            foreach
                    select * into ret from table(inp)
                    update wired set value = null where wid = ret;
            end foreach;
    end foreach;
    foreach
            select output into inp from norgate where gid in (select gid from layout where lid = l)
            foreach
                    select * into ret from table(inp)
                    update wired set value = null where wid = ret;
            end foreach;
    end foreach;
    foreach
            select output into inp from xorgate where gid in (select gid from layout where lid = l)
            foreach
                    select * into ret from table(inp)
                    update wired set value = null where wid = ret;
            end foreach;
    end foreach;
    foreach
            select output into inp from xnorgate where gid in (select gid from layout where lid = l)
            foreach
                    select * into ret from table(inp)
                    update wired set value = null where wid = ret;
            end foreach;
    end foreach;
end function;
```

```
create function initwired (number int)
        returning int;
```

```
        define ret int;
        define i int;

        select max(wid) into ret from wired;
        if ret is null then let ret = 1000;
        end if;
        for i = 1 to number
                insert into wired values(ret+i,null,null,null);
        end for;
end function;
```

```
create function insertgate (number int,posx int,posy int,type char(10),listinput char(100),listoutput char(100),l
int)
        returning int;
        define ret int;

        execute function getgidorder(number,l) into ret;
        update layout set (pos_x,pos_y) = (posx,posy) where gid = ret;
        if type = 'and' then insert into andgate values(ret,'"+listinput+"','"+listoutput+"');
        elif type = 'or' then insert into orgate values(ret,'"+listinput+"','"+listoutput+"');
        elif type = 'not' then insert into notgate values(ret,listinput,listoutput);
        elif type = 'nand' then insert into nandgate values(ret,listinput,listoutput);
        elif type = 'nor' then insert into norgate values(ret,"+listinput+",listoutput);
        elif type = 'xor' then insert into xorgate values(ret,listinput,listoutput);
        elif type = 'xnor' then insert into xnorgate values(ret,listinput,listoutput);
        end if;
end function;
```

```
create function insertvalue (number int,val char,l int)
        returning int;
        define ret int;
```

```
        execute function getwidorder(number,l)into ret;
        update wired set value = val where wid = ret;
end function;
```

```
create function loadgate(l int)
        returning char(4),int,int;
        define ret int;
        define ret1 int;
        define ret2 int;

        foreach
                select gid into ret from layout where lid = l order by gid
                select gid into ret1 from andgate where gid = ret;
                if ret1 is not null then
                        select cardinality(input),cardinality(output) into ret1,ret2 from andgate where gid =
ret;
                        return 'and',ret1,ret2 with resume;
                else
                        select gid into ret1 from orgate where gid = ret;
                        if ret1 is not null then
                                select cardinality(input),cardinality(output) into ret1,ret2 from orgate where
gid = ret;
                                return 'or',ret1,ret2 with resume;
                        else
                                select gid into ret1 from notgate where gid = ret;
                                if ret1 is not null then
                                        select cardinality(input),cardinality(output) into ret1,ret2 from
notgate where gid = ret;
                                        return 'not',ret1,ret2 with resume;
                                else
```

```
                                        select gid into ret1 from nandgate where gid = ret;
                                        if ret1 is not null then
                                                select cardinality(input),cardinality(output) into ret1,ret2
from nandgate where gid = ret;
                                                return 'nand',ret1,ret2 with resume;
                                        else
                                                select gid into ret1 from norgate where gid = ret;
                                                if ret1 is not null then
                                                        select cardinality(input),cardinality(output) into
ret1,ret2 from norgate where gid = ret;
                                                        return 'nor',ret1,ret2 with resume;
                                                else
                                                        select gid into ret1 from xorgate where gid = ret;
                                                        if ret1 is not null then
                                                        select cardinality(input),cardinality(output) into
ret1,ret2 from xorgate where gid = ret;
                                                        return 'xor',ret1,ret2 with resume;
                                                        else
                                                        select cardinality(input),cardinality(output) into
ret1,ret2 from xnorgate where gid = ret;
                                                        return 'xnor',ret1,ret2 with resume;
                                                        end if;
                                                end if;
                                        end if;
                                end if;
                        end if;
                end if;
        end foreach;
end function;
```

```
create function loadgatelist (number int,l int)
        returning int;
        define ret int;
        define ret1 collection;
        define ret2 int;

        execute function getwidorder(number,l) into ret;
        select gid into ret1 from wired where wid = ret;
        foreach
                select * into ret2 from table(ret1)
                execute function getordergid(ret2,l) into ret2;
                return ret2 with resume;
        end foreach;
end function;
```

```
create function loadinput(number int,l int)
        returning int;
        define ret int;
        define ret1 int;
        define ret2 collection;

        execute function getgidorder(number,l) into ret;
        select gid into ret1 from andgate where gid = ret;
        if ret1 is not null then
                select input into ret2 from andgate where gid = ret;
                foreach
                        select * into ret1 from table(ret2)
                        execute function getorderwid(ret1,l) into ret1;
                        return ret1 with resume;
                end foreach;
        else
```

```
select gid into ret1 from orgate where gid = ret;
if ret1 is not null then
        select input into ret2 from orgate where gid = ret;
        foreach
                select * into ret1 from table(ret2)
                execute function getorderwid(ret1,1) into ret1;
                return ret1 with resume;
        end foreach;
else
        select gid into ret1 from notgate where gid = ret;
        if ret1 is not null then
                select input into ret2 from notgate where gid = ret;
                foreach
                        select * into ret1 from table(ret2)
                        execute function getorderwid(ret1,1) into ret1;
                        return ret1 with resume;
                end foreach;
        else
                select gid into ret1 from nandgate where gid = ret;
                if ret1 is not null then
                        select input into ret2 from nandgate where gid = ret;
                        foreach
                                select * into ret1 from table(ret2)
                                execute function getorderwid(ret1,1) into ret1;
                                return ret1 with resume;
                        end foreach;
                else
                        select gid into ret1 from norgate where gid = ret;
                        if ret1 is not null then
                                select input into ret2 from norgate where gid = ret;
                                foreach
```

```
                                        select * into ret1 from table(ret2)
                                        execute function getorderwid(ret1,l) into ret1;
                                        return ret1 with resume;
                                    end foreach;
                                else
                                    select gid into ret1 from xorgate where gid = ret;
                                    if ret1 is not null then
                                        select input into ret2 from xorgate where gid =
ret;
                                        foreach
                                            select * into ret1 from table(ret2)
                                            execute function getorderwid(ret1,l) into
ret1;
                                            return ret1 with resume;
                                        end foreach;
                                    else
                                        select input into ret2 from xnorgate where gid =
ret;
                                        foreach
                                            select * into ret1 from table(ret2)
                                            execute function getorderwid(ret1,l) into
ret1;
                                            return ret1 with resume;
                                        end foreach;
                                    end if;
                                end if;
                            end if;
                        end if;
                    end if;
                end if;
            end if;
        end function;
```

```
create function loadoutput(number int,l int)
        returning int;
        define ret int;
        define ret1 int;
        define ret2 collection;

        execute function getgidorder(number,l) into ret;
        select gid into ret1 from andgate where gid = ret;
        if ret1 is not null then
                select output into ret2 from andgate where gid = ret;
                foreach
                        select * into ret1 from table(ret2)
                        execute function getorderwid(ret1,l) into ret1;
                        return ret1 with resume;
                end foreach;
        else
                select gid into ret1 from orgate where gid = ret;
                if ret1 is not null then
                        select output into ret2 from orgate where gid = ret;
                        foreach
                                select * into ret1 from table(ret2)
                                execute function getorderwid(ret1,l) into ret1;
                                return ret1 with resume;
                        end foreach;
                else
                        select gid into ret1 from notgate where gid = ret;
                        if ret1 is not null then
                                select output into ret2 from notgate where gid = ret;
                                foreach
                                        select * into ret1 from table(ret2)
```

```
                execute function getorderwid(ret1,1) into ret1;
                return ret1 with resume;
        end foreach;
else
        select gid into ret1 from nandgate where gid = ret;
        if ret1 is not null then
                select output into ret2 from nandgate where gid = ret;
                foreach
                        select * into ret1 from table(ret2)
                        execute function getorderwid(ret1,1) into ret1;
                        return ret1 with resume;
                end foreach;
        else
                select gid into ret1 from norgate where gid = ret;
                if ret1 is not null then
                        select output into ret2 from norgate where gid = ret;
                        foreach
                                select * into ret1 from table(ret2)
                                execute function getorderwid(ret1,1) into ret1;
                                return ret1 with resume;
                        end foreach;
                else
                        select gid into ret1 from xorgate where gid = ret;
                        if ret1 is not null then
                                select output into ret2 from xorgate where gid =
ret;
                                foreach
                                        select * into ret1 from table(ret2)
                                        execute function getorderwid(ret1,1) into
ret1;
                                        return ret1 with resume;
```

```
                                        end foreach;
                                    else
                                        select output into ret2 from xnorgate where gid =
ret;
                                        foreach
                                            select * into ret1 from table(ret2)
                                            execute function getorderwid(ret1,l) into
ret1;
                                            return ret1 with resume;
                                        end foreach;
                                    end if;
                                end if;
                            end if;
                        end if;
                    end if;
                end if;
end function;
```

```
create function loadpos(l int)
        returning int,int;
        define ret int;
        define ret1 int;
        define ret2 int;

        foreach
            select gid,pos_x,pos_y into ret,ret1,ret2 from layout where lid = l order by gid
            return ret1,ret2 with resume;
        end foreach;
end function;
```

```
create function loadwired (l int)
```

```
        returning char,int;
        define ret int;
        define ret1 char;
        define ret2 int;

        foreach
                execute function getwid(l) into ret
                select isoutput,cardinality(gid) into ret1,ret2 from wired where wid = ret;
                return ret1,ret2 with resume;
        end foreach;
end function;
```

Visual Basic Module

```
Attribute VB_Name = "Module1"
Option Base 1

Public Type gate
   gateType As String
   posx As Integer
   posy As Integer
   inputlist(1 To 10) As Integer
   nInput As Integer
   outputlist(1 To 10) As Integer
   nOutput As Integer
   linkLayout As Integer
End Type

Public Type Wired
   gatelist(1 To 10) As Integer
```

```
    ngate As Integer
    wiredtype As String
End Type

Public Type group
    name As Integer
    number As Integer
End Type

Public gates() As gate
Public wireds() As Wired
Public oEngine As ddoEngine
Public oDatagroup As ddoDataGroup
Public gateCount As Integer
Public wiredCount As Integer
Public lineCount As Integer
Public pid As String
Public lid As Integer

Public Sub initConnect()

    Set oEngine = CreateObject("DataDirector.Engine")
    Set oDatagroup = oEngine.CurrentProject.CreateDataGroup("proj_layout", "gategroup",
"E:\informix\sql\gatemodel.mlt")
    oDatagroup.AutoCommit = True
    oDatagroup.Logon "informix", "informix"

End Sub

Public Sub disconnect()
    oDatagroup.Logoff
```

```
    Set oDatagroup = Nothing
    Set oEngine = Nothing
End Sub

Public Sub saveModel()
    Dim oInitlayout As ddoTable
    Dim oGetwid As ddoTable
    Dim oGetlid As ddoTable
    Dim oInsertlid As ddoTable
    Dim i As Integer
    Dim j As Integer
    Dim inlist As String
    Dim outlist As String
    Dim temp As Integer

    saveInOutSystem
    For i = 1 To wiredCount
      If wireds(i).wiredtype = "" Then
        wireds(i).wiredtype = "N"
      End If
    Next i
    Set oGetlid = oDatagroup.CreateVirtualTable
    oGetlid.QueryCommand = "select lid,newcomp from proj_layout where pid = '" + pid + "'"
    oGetlid.ExecuteCommand QUERY_CMD
    If oGetlid.TotalRecordCount = 0 Then
      oGetlid.QueryCommand = "select max(lid) from proj_layout"
      oGetlid.ExecuteCommand QUERY_CMD
      If oGetlid.Columns(1) = "" Then
        lid = 1
      Else
        lid = oGetlid.Columns(1).Value + 1
```

```
    End If
    Set oInsertlid = oDatagroup.CreateVirtualTable
    oInsertlid.InsertCommand = "insert into proj_layout values('" + pid + "',?,'N')"
    oInsertlid.SetParam INSERT_CMD, 1, lid
    oInsertlid.ExecuteCommand INSERT_CMD
    oDatagroup.DeleteVirtualTable (oInsertlid.name)
  Else
    lid = oGetlid.Columns(1).Value
    If oGetlid.Columns(2).Value = "Y" Then
      MsgBox "can't save as exist new component"
      oDatagroup.DeleteVirtualTable (oGetlid.name)
      Set oGetlid = Nothing
      Exit Sub
    End If
  End If
  oDatagroup.DeleteVirtualTable (oGetlid.name)
  Set oGetlid = Nothing
  Set oGetlid = oDatagroup.CreateVirtualTable
  oGetlid.QueryCommand = "{call clearlayout(?)}"
  oGetlid.SetParam QUERY_CMD, 1, lid
  oGetlid.ExecuteCommand QUERY_CMD
  oDatagroup.DeleteVirtualTable (oGetlid.name)
  Set oGetlid = Nothing
  Set oInsertlid = Nothing
  Set oInitlayout = oDatagroup.CreateVirtualTable
  oInitlayout.QueryCommand = "{call initlayout(?,?)}"
  oInitlayout.SetParam QUERY_CMD, 1, gateCount
  oInitlayout.SetParam QUERY_CMD, 2, lid
  oInitlayout.ExecuteCommand QUERY_CMD
  oInitlayout.QueryCommand = "{call initwired(?)}"
  oInitlayout.SetParam QUERY_CMD, 1, wiredCount
```

```
oInitlayout.ExecuteCommand QUERY_CMD
oDatagroup.DeleteVirtualTable (oInitlayout.name)
Set oInitlayout = Nothing
oDatagroup.Logoff
oDatagroup.Logon
Set oGetwid = oDatagroup.CreateVirtualTable
oGetwid.QueryCommand = "{call getnewwid(?)}"
Set oInitlayout = oDatagroup.CreateVirtualTable
oInitlayout.QueryCommand = "{call getgidorder(?,?)}"
oInitlayout.SetParam QUERY_CMD, 2, lid
oInitlayout.UpdateCommand = "update layout set(pos_x,pos_y) = (?,?) where gid = ?"
For i = 1 To gateCount
   oInitlayout.SetParam QUERY_CMD, 1, i
   oInitlayout.SetParam UPDATE_CMD, 1, gates(i).posx
   oInitlayout.SetParam UPDATE_CMD, 2, gates(i).posy
   oInitlayout.ExecuteCommand QUERY_CMD
   oInitlayout.SetParam UPDATE_CMD, 3, oInitlayout.Columns(1).Value
   oInitlayout.ExecuteCommand UPDATE_CMD
   If gates(i).nInput = 0 Then
      inlist = "null"
   Else
      inlist = "'list{"
      For j = 1 To gates(i).nInput
         oGetwid.SetParam QUERY_CMD, 1, gates(i).inputlist(j)
         oGetwid.ExecuteCommand QUERY_CMD
         inlist = inlist + CStr(oGetwid.Columns(1))
         If j < gates(i).nInput Then
            inlist = inlist + ","
         End If
      Next j
      inlist = inlist + "}'"
```

```
    End If
    If gates(i).nOutput = 0 Then
      outlist = "null"
    Else
      outlist = "list{"
      For j = 1 To gates(i).nOutput
        oGetwid.SetParam QUERY_CMD, 1, gates(i).outputlist(j)
        oGetwid.ExecuteCommand QUERY_CMD
        outlist = outlist + CStr(oGetwid.Columns(1))
        If j < gates(i).nOutput Then
          outlist = outlist + ","
        End If
      Next j
      outlist = outlist + "}""
    End If
    If gates(i).gateType <> "new" Then
      oInitlayout.InsertCommand = "insert into " + gates(i).gateType + "gate values(?," + inlist + "," + outlist +
")"
      oInitlayout.SetParam INSERT_CMD, 1, oInitlayout.Columns(1).Value
      oInitlayout.ExecuteCommand INSERT_CMD
    Else
      oInitlayout.InsertCommand = "insert into " + gates(i).gateType + "gate values(?," + inlist + "," + outlist +
",?)"
      oInitlayout.SetParam INSERT_CMD, 1, oInitlayout.Columns(1).Value
      oInitlayout.SetParam INSERT_CMD, 2, gates(i).linkLayout
      oInitlayout.ExecuteCommand INSERT_CMD
    End If
  Next i
  oDatagroup.DeleteVirtualTable (oGetwid.name)
  Set oGetwid = Nothing
  Set oGetwid = oDatagroup.CreateVirtualTable
```

```
    oGetwid.QueryCommand = "{call getwidorder(?,?)}"
    oGetwid.SetParam QUERY_CMD, 2, lid
    For i = 1 To wiredCount
        If wireds(i).ngate = 0 Then
          inlist = "null"
        Else
          inlist = "list{"
          For j = 1 To wireds(i).ngate
            oInitlayout.SetParam QUERY_CMD, 1, wireds(i).gatelist(j)
            oInitlayout.ExecuteCommand QUERY_CMD
            inlist = inlist + CStr(oInitlayout.Columns(1))
            If j < wireds(i).ngate Then
              inlist = inlist + ","
            End If
          Next j
          inlist = inlist + "}"
        End If
        oInitlayout.UpdateCommand = "update wired set (gid,isoutput)= (" + inlist + ",?) where wid = ?"
        oGetwid.SetParam QUERY_CMD, 1, i
        oGetwid.ExecuteCommand QUERY_CMD
        oInitlayout.SetParam UPDATE_CMD, 1, wireds(i).wiredtype
        temp = oGetwid.Columns(1)
        oInitlayout.SetParam UPDATE_CMD, 2, temp
        oInitlayout.ExecuteCommand UPDATE_CMD
    Next i
End Sub

Public Sub loadModel()
    Dim oLoadLayout As ddoTable
    Dim oLoadGate As ddoTable
    Dim oLoadinput As ddoTable
```

```
Dim oLoadoutput As ddoTable
Dim oLoadLink As ddoTable
Dim i As Integer
Dim j As Integer

Set oLoadLayout = oDatagroup.CreateVirtualTable
oLoadLayout.QueryCommand = "{call getnumbergate(?)}"
oLoadLayout.SetParam QUERY_CMD, 1, lid
oLoadLayout.ExecuteCommand QUERY_CMD
gateCount = oLoadLayout.Columns(1).Value
oDatagroup.DeleteVirtualTable (oLoadLayout.name)
Set oLoadLayout = Nothing
Set oLoadLayout = oDatagroup.CreateVirtualTable
oLoadLayout.QueryCommand = "{call getnumberwired(?)}"
oLoadLayout.SetParam QUERY_CMD, 1, lid
oLoadLayout.ExecuteCommand QUERY_CMD
wiredCount = oLoadLayout.Columns(1).Value
oDatagroup.DeleteVirtualTable (oLoadLayout.name)
Set oLoadLayout = Nothing
ReDim gates(gateCount + 10)
ReDim wireds(wiredCount + 10)
Set oLoadLayout = oDatagroup.CreateVirtualTable
oLoadLayout.QueryCommand = "{call loadpos(?)}"
oLoadLayout.SetParam QUERY_CMD, 1, lid
oLoadLayout.ExecuteCommand QUERY_CMD
oLoadLayout.FirstRecord
Set oLoadGate = oDatagroup.CreateVirtualTable
oLoadGate.QueryCommand = "{call loadgate(?)}"
oLoadGate.SetParam QUERY_CMD, 1, lid
oLoadGate.ExecuteCommand QUERY_CMD
oLoadGate.FirstRecord
```

```
Set oLoadinput = oDatagroup.CreateVirtualTable
oLoadinput.QueryCommand = "{call loadinput(?,?)}"
oLoadinput.SetParam QUERY_CMD, 2, lid
Set oLoadoutput = oDatagroup.CreateVirtualTable
oLoadoutput.QueryCommand = "{call loadoutput(?,?)}"
oLoadoutput.SetParam QUERY_CMD, 2, lid
For i = 1 To gateCount
   gates(i).posx = oLoadLayout.Columns(1).Value
   gates(i).posy = oLoadLayout.Columns(2).Value
   gates(i).gateType = Trim(oLoadGate.Columns(1).Value)
   gates(i).nInput = oLoadGate.Columns(2).Value
   gates(i).nOutput = oLoadGate.Columns(3).Value
   If gates(i).gateType = "new" Then
      Set oLoadLink = oDatagroup.CreateVirtualTable
      oLoadLink.QueryCommand = "{call loadlink(?,?)}"
      oLoadLink.SetParam QUERY_CMD, 1, i
      oLoadLink.SetParam QUERY_CMD, 2, lid
      oLoadLink.ExecuteCommand QUERY_CMD
      gates(i).linkLayout = oLoadLink.Columns(1).Value
      oDatagroup.DeleteVirtualTable (oLoadLink.name)
      Set oLoadLink = Nothing
   End If
   oLoadinput.SetParam QUERY_CMD, 1, i
   oLoadinput.ExecuteCommand QUERY_CMD
   oLoadinput.FirstRecord
   For j = 1 To gates(i).nInput
      gates(i).inputlist(j) = oLoadinput.Columns(1).Value
      oLoadinput.NextRecord
   Next j
   oLoadoutput.SetParam QUERY_CMD, 1, i
   oLoadoutput.ExecuteCommand QUERY_CMD
```

```
    oLoadoutput.FirstRecord
    For j = 1 To gates(i).nOutput
      gates(i).outputlist(j) = oLoadoutput.Columns(1).Value
      oLoadoutput.NextRecord
    Next j
    oLoadLayout.NextRecord
    oLoadGate.NextRecord
  Next i
  oDatagroup.DeleteVirtualTable (oLoadLayout.name)
  Set oLoadLayout = Nothing
  oDatagroup.DeleteVirtualTable (oLoadGate.name)
  Set oLoadGate = Nothing
  oDatagroup.DeleteVirtualTable (oLoadinput.name)
  Set oLoadinput = Nothing
  oDatagroup.DeleteVirtualTable (oLoadoutput.name)
  Set oLoadoutput = Nothing
  Set oLoadGate = oDatagroup.CreateVirtualTable
  oLoadGate.QueryCommand = "{call loadwired(?)}"
  oLoadGate.SetParam QUERY_CMD, 1, lid
  oLoadGate.ExecuteCommand QUERY_CMD
  oLoadGate.FirstRecord
  Set oLoadinput = oDatagroup.CreateVirtualTable
  oLoadinput.QueryCommand = "{call loadgatelist(?,?)}"
  oLoadinput.SetParam QUERY_CMD, 2, lid
  For i = 1 To wiredCount
    wireds(i).wiredtype = oLoadGate.Columns(1).Value
    wireds(i).ngate = oLoadGate.Columns(2).Value
    oLoadinput.SetParam QUERY_CMD, 1, i
    oLoadinput.ExecuteCommand QUERY_CMD
    oLoadinput.FirstRecord
    For j = 1 To wireds(i).ngate
```

```
        wireds(i).gatelist(j) = oLoadinput.Columns(1).Value
        oLoadinput.NextRecord
      Next j
      oLoadGate.NextRecord
    Next i
    oDatagroup.DeleteVirtualTable (oLoadGate.name)
    Set oLoadGate = Nothing
    oDatagroup.DeleteVirtualTable (oLoadinput.name)
    Set oLoadinput = Nothing
End Sub

Public Sub saveInOutSystem()
    Dim i, j As Integer

    For i = 1 To gateCount
      For j = 1 To gates(i).nInput
        If gates(i).inputlist(j) = 0 Then
          wiredCount = wiredCount + 1
          If UBound(wireds) < wiredCount Then
            ReDim Preserve wireds(wiredCount + 10)
          End If
          gates(i).inputlist(j) = wiredCount
          wireds(wiredCount).gatelist(1) = i
          wireds(wiredCount).ngate = 1
          wireds(wiredCount).wiredtype = "I"
        End If
      Next j
    Next i
    For i = 1 To gateCount
      For j = 1 To gates(i).nOutput
        If gates(i).outputlist(j) = 0 Then
```

```
            wiredCount = wiredCount + 1
            If UBound(wireds) < wiredCount Then
               ReDim Preserve wireds(wiredCount + 10)
            End If
            gates(i).outputlist(j) = wiredCount
            wireds(wiredCount).gatelist(1) = i
            wireds(wiredCount).ngate = 1
            wireds(wiredCount).wiredtype = "O"
         End If
      Next j
   Next i
End Sub

Public Sub checkModel()
   Dim oCheckInput As ddoTable
   Dim oCheckLoop As ddoTable

   Set oCheckInput = oDatagroup.CreateVirtualTable
   oCheckInput.QueryCommand = "{call checkinput(" + CStr(lid) + ")}"
   oCheckInput.ExecuteCommand QUERY_CMD
   If CStr(oCheckInput.Columns(1).Value) = "Y" Then
      Set oCheckLoop = oDatagroup.CreateVirtualTable
      oCheckLoop.QueryCommand = "{call checkloop(" + CStr(lid) + ")}"
      oCheckLoop.ExecuteCommand QUERY_CMD
      If CStr(oCheckLoop.Columns(1).Value) = "Y" Then
         MDI.resultCheck = 0
      Else
         MDI.resultCheck = 1
      End If
      oDatagroup.DeleteVirtualTable (oCheckLoop.name)
      Set oCheckLoop = Nothing
```

```
    Else
      MDI.resultCheck = 2
    End If
    oDatagroup.DeleteVirtualTable (oCheckInput.name)
    Set oCheckInput = Nothing
End Sub
```

บรรณานุกรม

[1] Date C.J. (1986) : "*An introduction to Database System Vol. 1, 4 Edition.* ", Addison-Wesley, Massachusatts, 1986.

[2] Abrah Silberschatz,Henry F. Korth and S.Sudarshan (1996): "*Database System Concept,3 Edition*",McGraw-Hill International Editions

[3] Jeffrey D. Ullman and Jennifer Widom (1997) : "*A First Course in Database Systems*", Prentice-Hall International,Inc.

[4] Jim Melton (1995) : "*ISO/IEC JTC1.21.3.3 : Accomodating SQL3 and ODMG*"

[5] International Organization for Standardization (1992): "*ANSI X3.135-1992 : American National Standard for Information Systems-SQL*" , American National Standard Institue

[6] Jim Melton (1995) : "ISO-ANSI Working Draft) X3H2-95-084/DBL:YOW-004 : Database Language (SQL3)" , International Organization for Standardization and American National Standard Institue

[7] Frank Manola (1994) : TR-0263-08-94-165 "*An Evaluation of Object-Oriented DBMS Development 1994 Edition*" , GTE Laboratories Incorperated

[8] John Harrington,Mark Spenik,Heidi Brumbaugh and Cliff Diamond (1997) :"*Visual Basic 5 Interactive Course*" , Waite Group Press