ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อการรู้จำอักษรภาษาไทย

AN OBJECT RELATIONAL DATABASE SYSTEM FOR THAI CHARACTER RECOGNITION APPLICATION

นาย ชวลิต ศิวบวรวัฒนา

นาย สุรชัย เลิศบุญช่วยกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อการรู้จำอักษรภาษาไทย

AN OBJECT RELATIONAL DATABASE SYSTEM FOR THAI CHARACTER RECOGNITION APPLICATION

โดย

นาย ชวลิต ศิวบวรวัฒนา

นาย สุรชัย เลิศบุญช่วยกุล

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อการรู้จำอักษรภาษาไทย

AN OBJECT RELATIONAL DATABASE SYSTEM FOR THAI CHARACTER RECOGNITION APPLICATION

ผู้จัดทำ

1. นาย ชวลิต ศิวบวรวัฒนา 39013233
2. นาย สุรชัย เลิศบุญช่วยกุล 39013257

อาจารย์ที่ปรึกษา

(รศ.ดร.ศุภมิตร จิตตะยโศธร)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อการรู้จำอักษรภาษาไทย

นาย ชวลิต ศิวบวรวัฒนา 39013233
นาย สุรชัย เลิศบุญช่วยกุล 39013257

รศ.ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในปัจจุบันในโปรแกรมต่างๆ มีการใช้ข้อมูลที่มีลักษณะพิเศษ เช่น ข้อมูลทางมัลติมีเดีย เป็นต้น ซึ่งข้อมูลเหล่านี้มีลักษณะที่ซับซ้อนอยู่ในตัวและระบบฐานข้อมูลแบบสัมพันธ์ที่ใช้กันในปัจจุบันได้ถูกออกแบบมาเพื่อสนับสนุนเฉพาะข้อมูลง่ายๆเท่านั้น

ระบบจัดการฐานข้อมูลแบบเชิงวัตถุสัมพันธ์เป็นเทคโนโลยีใหม่ที่ได้รวมเอาข้อดีของหลักการเชิงวัตถุและระบบจัดการฐานข้อมูลแบบสัมพันธ์เข้าด้วยกัน ระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์นี้ได้ขยายความสามารถของระบบจัดการฐานข้อมูลแบบสัมพันธ์ในเรื่องการสนับสนุนชนิดข้อมูลที่มีความซับซ้อน โดยในคอลัมน์สามารถที่จะมีชนิดข้อมูลที่ไม่เฉพาะชนิดข้อมูลง่ายๆ เท่านั้นแต่ยังสนับสนุนข้อมูลที่มีลักษณะเป็นคอเลคชัน ออบเจ็กต์ และข้อมูลที่มีโครงสร้างซับซ้อน นอกจากนี้เรายังสามารถที่จะเขียนฟังก์ชันและเมธทอดต่างๆ เพื่อใช้ในการจัดการกับข้อมูลที่ซับซ้อนนี้ได้

โดยในปริญญานิพนธ์นี้เป็นส่วนหนึ่งของโครงงานที่ได้ทำการศึกษาถึงความสามารถของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์โดยสร้างโปรแกรมที่ธียนรู้และจดจำอักษรภาษาไทยเพื่อทำการวิเคราะห์ถึงของดีของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์และทำการเปรียบเทียบระหว่างระบบฐานข้อมูลแบบสัมพันธ์กับระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์

An Object Relational Database System for Thai character recognition application

Mr.Chawarit Siwaborwornwattana

Mr.Surachai Lerdboonchuaykul

Assoc. Prof.Dr.Suphamit chittayasothon Advisor

ABSTRACT

Recently, In many applications use special data type such as multimedia data. These datatypes in also very complex in nature and a relational database system (RDBMS) was designed to support simple data type

An object-relational database management system (ORDBMS) is the next evolutionary step in the database technology with combine the best feature of object oriennted concept and relational databases. In an ORDBMS extends the relational to support complex data a column can hold not just the simple datatype, but also collection data type,object data type and complex structured data.Develpers can write functions and methods to manipulate the complex data

This thesis deals with study of new feature of ORDBMS by create Thai character recognition application to analysis advantage of ORDBMS and making the comparison between ORDBMS and RDBMS

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพประกอบ

สารบัญภาพประกอบ(ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในปัจจุบันการใช้งานฐานข้อมูลมีความต้องการชนิดข้อมูลและวิธีการเข้าถึงข้อมูลที่มีความซับซ้อนและต้องการความสามารถสูงขึ้น เพื่อที่จะนำไปใช้งานกับโปรแกรมงานต่างๆ (Application) เช่น โปรแกรมที่มีการใช้ข้อมูลทางมัลติมีเดีย (Multimedia) โปรแกรมเกี่ยวกับข้อมูลภูมิศาสตร์ (Geographical Information Systems (GIS)) และโปรแกรมทางด้าน CAD เป็นต้น ซึ่งโดยมากแล้วข้อมูลที่โปรแกรมเหล่านี้ ใช้มักจะมีขนาดใหญ่หรือไม่ก็มีขนาดไม่คงทีและมักจะต้องการชนิดข้อมูลพิเศษที่ใช้ในการจัดเก็บข้อมูลประเภทนี้ ซึ่งระบบฐานข้อมูลแบบสัมพันธ์ (Relational Database) ที่มีอยู่ในปัจจุบันไม่เหมาะสมที่จะนำมาใช้งาน เนื่องจากระบบฐานข้อมูลแบบสัมพันธ์ไม่สนับสนุนในเรื่องของ

1. โครงสร้างข้อมูล : เนื่องจากชนิดของข้อมูลที่มีอยู่จะเป็นชนิดข้อมูลแบบธรรมดาเช่น integer, floats, character string, date-time ฯลฯ
2. วิธีการเข้าถึงข้อมูล :มีการใช้อินเด็กซ์แบบ B-tree ซึ่งจะมีข้อจำกัดบางประการในการใช้งานกับข้อมูลที่มีลักษณะหลายมิติ (multi-dimensional space) และกับชนิดข้อมูลที่ผู้ใช้สร้างขึ้นเอง(user-defined data types)
3. การเขียนโค้ด(code) : เนื่องจากระบบฐานข้อมูลแบบสัมพันธ์สนับสนุนเฉพาะ SQL-92 ซึ่งไม่สนับสนุนการทำคิวรี่ กับข้อมูลที่มีความซับซ้อนหรือชนิดข้อมูลที่เป็นแบบออบเจ็กต์ที่มีฟังก์ชันสนับสนุนหรือเมธทอดอยู่ได้

ซึ่งนอกจากระบบฐานข้อมูลแบบสัมพันธ์แล้วยังมีระบบฐานข้อมูลแบบเชิงวัตถุ (Object Oriented Database) ซึ่งเป็นระบบฐานข้อมูลที่สนับสนุนหลักการของออบเจ็กต์ (Object-Oriented)โดยใช้ออบเจ็กต์เป็นพื้นฐานที่มีความสามารถในการสนับสนุนชนิดข้อมูลที่มีความซับซ้อนมาก แต่ก็ยังมีข้อเสียบางประการและไม่สนับสนุนกับงานข้างต้นเต็มที่เช่น ในเรื่องของการทำ Concurrency อันเนื่องจากการทำ Object-Based Concurrency control ของระบบฐานข้อมูลแบบเชิงวัตถุยังเป็นหัวข้อในการวิจัยอยู่ และในเรื่องของภาษาที่ใช้กับระบบฐานข้อมูลนี้ยังไม่มีมาตรฐานรองรับที่แน่นอนและอีกทั้งยังเป็นการยากที่จะเปลี่ยนระบบฐานข้อมูลแบบเดิม (RDBMS) ที่ใช้ในปัจจุบันมาเป็นระบบฐานข้อมูลแบบเชิงวัตถุเลยที่เดียว จึงเกิดความคิดที่จะพัฒนาระบบฐานข้อมูลขึ้นมาใหม่โดยพยายามที่จะนำข้อดีของทั้งสองระบบมาร่วมกันเป็นระบบฐานข้อมูลแบบใหม่คือระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) ขึ้น

ซึ่งในความเป็นจริงนั้น การที่นำเอาระบบฐานข้อมูลเชิงวัตถุมาใช้ร่วมกับระบบฐานข้อมูลแบบสัมพันธ์ย่อมจะทำคุณสมบัติในทางเชิงวัตถุลดน้อยลงกว่าระบบฐานข้อมูลแบบเชิงวัตถุอย่างเดียว และเนื่องจากการเกิดของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ ก็จำเป็นที่จะต้องมีภาษาที่มาสนับสนุนคุณ

สมบัติต่างๆ ที่เพิ่มขึ้นมา ดังนั้นทาง ISO และ ANSI จึงได้มีการกำหนดมาตรฐานของภาษา SQL ออกมาเป็น SQL3 ที่มีคุณสมบัติต่างๆ ที่สนับสนุนในเรื่องของออบเจ็กต์ คล้ายกับภาษา OQL (Object Query Language) ที่ใช้สำหรับระบบฐานข้อมูลที่เป็นแบบเชิงวัตถุซึ่งเป็นภาษาที่ทาง ODMG ได้กำหนดมาตรฐานออกมาเพื่อสนับสนุนการทำออบเจ็กต์คิวรี่ ทั้งภาษา SQL และ OQL ก็ยึดหลักความต้องการให้ภาษามีลักษณะที่ง่ายต้องการใช้ ("ease-of-use") ในการทำคิวรี่กับระบบฐานข้อมูลเชิงวัตถุ และระบบฐานข้อมูลแบบสัมพันธ์

โดยในโครงงานนี้จะเป็นการศึกษาถึงความสามารถต่างๆ ของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ในเรื่องของการสนับสนุนหลักการของออบเจ็กต์และข้อมูลที่มีความซับซ้อน ประสิทธิภาพการทำงานกับงานที่ใช้ข้อมูลทางมัลติมีเดีย โดยจะนำระบบฐานข้อมูลของอินฟอร์มิกซ์ (Informix) และภาษา SQL3 (ในปัจจุบันมาตรฐานของ SQL3 ยังไม่เป็นมาตรฐานที่สมบูรณ์ซึ่งในที่นี้จะอ้างถึงเฉพาะ SQL3 ที่เป็นมาตรฐานที่ร่างไว้ (draft standard) เท่านั้น) มาเป็นกรณีศึกษาโดยจะเริ่มจากการศึกษาคุณสมบัติต่างๆ ของระบบเปรียบเทียบกับมาตราฐานที่ได้กำหนดไว้ใน SQL3 และจากนั้นจะทดลองทำโปรแกรมเพื่อที่จะทำการทดสอบโดยจะลองวิ่งโปรแกรมนี้บนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์นี้

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาทฤษฎีหลักการและมาตรฐานของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ในด้านต่างๆ

1.2.2 ศึกษาความความสามารถของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ที่ได้ขยายขีดจำกัดที่มีอยู่ในระบบฐานข้อมูลแบบสัมพันธ์เดิม

1.2.3 วิเคราะห์และเปรียบเทียบข้อดีและข้อเสียของการใช้ระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์กับระบบฐานข้อมูลแบบเชิงสัมพันธ์

1.2.4 วิเคราะห์ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ของอินฟอร์มิกซ์ว่ามีคุณสมบัติต่างๆ ตามมาตรฐาน SQL3 มากน้อยอย่างไร

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้ได้ทำการศึกษาถึงความสามารถต่างๆ ของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ในการที่จะนำมาใช้ประโยชน์กับงานที่มีการใช้หรือทำงานกับข้อมูลแบบมัลติมิเดียหรือกับข้อมูลที่มีความซับซ้อนสูง โดยงานวิจัยนี้จะทำการทดลองสร้างโปรแกรมแอพพริเคชั่นที่ใช้ข้อมูลที่มีลักษณะดังกล่าวมาเพื่อทำการทดสอบความสามารถและคุณสมบัติต่างๆ ที่มีในระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ ซึ่งตัวแอพพริเคชั่นนี้จะไม่ได้มุ่งเน้นในเรื่องของการทำงานของโปรแกรมในลักษณะแนวคิดการแก้ปัญหาที่ไม่เกี่ยวข้องกับการจัดการข้อมูล แต่จะมุ่งเน้นในเรื่องของการได้มีการนำเอาความสามารถต่างๆ ที่ไม่มีในระบบฐานข้อมูลแบบสัมพันธ์มาก่อนหรือมีแต่ทำได้ยากมาทดสอบ และนอกจากนี้ยังมุ่งเน้นในเรื่องของการใช้หลักการและการมองงานเป็นแบบเชิงวัตถุมาคิด

1.4 วิธีการดำเนินการ

งานวิจัยนี้จะเริ่มจากการศึกษาถึงหลักการและแนวคิดของทฤษฎีพื้นฐานต่างๆ ที่เกี่ยงข้องกับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ อันจะประกอบไปด้วย หลักการและแนวคิดของเชิงวัตถุ มาตรฐานของภาษาที่ใช้กับระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ซึ่งก็คือภาษา SQL3 ซึ่งเป็นภาษาที่มีความสามารถที่สนับสนุนความสามารถที่เพิ่มเติมมากับระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ จากนั้นก็จะทำการศึกษาระบบฐานข้อมูลที่เป็นผลิตภัณฑ์ของอินฟอร์มิกซ์ ที่เป็นระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์เทียบกับมาตราฐานที่ได้มีการกำหนดไว้สำหรับระบบฐานข้อมูลแบบนี้

และสุดท้ายของงานวิจัยนี้จะทำการสร้างโปรแกรมที่จะนำเอาหลักการของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์มาใช้ ซึ่งสำหรับโปรแกรมที่จะนำมาทำการทดสอบระบบฐานข้อมูลนี้จะแบ่งออกเป็น 2 โปรแกรม โดยโปรแกรมหนึ่งสำหรับทำการทดสอบความสามารถทางออบเจ็กต์ของระบบฐานข้อมูลว่ามีมากน้อยแค่ไหน และอีกโปรแกรมสำหรับทดสอบการสนับสนุนชนิดข้อมูลที่มีความซับซ้อนซึ่งก็คือโปรแกรมที่ใช้สำหรับให้เครื่องคอมพิวเตอร์สามารถที่จะเรียนรู้จดจำอักษรลายมือภาษาไทยได้นั่นเอง โดยตัวโปรแกรมนี้จะมีการสร้างชนิดข้อมูลมาเพื่อทำการเก็บข้อมูลที่มีลักษณะเป็นรหัสที่ผ่านอัลกอริทึม (Algorithm) ในลักษณะของข้อมูลที่เป็นออบเจ็กต์และจัดให้มีเมธทอดต่างๆ ที่ทำงานกับออบเจ็กต์นี้ ในการเรียนรู้จดจำอักษรภาษาไทยซึ่งสำหรับในการทดลองนี้ได้เลือกวิธีการที่เรียกว่ารหัสพี.คิว (P.Q Code) เพื่อช่วยในการจดจำอักษรภาษาไทยโดยจะมีการเก็บข้อมูลที่เป็นรหัสของอักษรและแบบของอักษรต่างๆ เอาไว้ในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์นี้

บทที่ 2
ทฤษฎีและความรู้พื้นฐานเกี่ยวกับออบเจ็กต์

โดยในบทนี้จะกล่าวในเรื่องของทฤษฎีและความรู้พื้นฐานของออบเจ็กต์ที่จำเป็น เพื่อที่จะนำไปใช้ในการศึกษาและทำความเข้าใจระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์

2.1 หลักการของออบเจ็กต์ (Object – Oriented Concept)

หลักการของออบเจ็กต์เป็นแนวทางและเทคนิคหนึ่งในปัจจุบันที่เลือกใช้ในการวิเคราะห์ ออกแบบและเขียนโปรแกรมของระบบที่มีขนาดใหญ่และซับซ้อน เพื่อให้สามารถพัฒนาและแก้ไขโปรแกรมต่อไปได้ง่ายขึ้นและมีความเป็นอิสระในการปรับปรุงแก้ไข ซึ่งหลักการดังกล่าวมีแนวความคิดอย่างไร มีลักษณะและรูปแบบการออกแบบเป็นอย่างไร มีคุณสมบัติอะไรบ้าง ซึ่งจะได้ทราบถึงรายละเอียดต่างๆ ในหัวข้อต่อๆ ไป

2.1.1 ออบเจ็กต์ (Object)

ออบเจ็กต์ หรือ วัตถุ คือสิ่งที่เรากำหนดขึ้นมาจากนามธรรมขึ้นมาเป็นรูปธรรมที่มีอยู่จริง โดยสิ่งที่มีอยู่จริงไม่จำเป็นต้องจับต้องได้เท่านั้นแต่จะรวมไปถึงอะไรก็ตามที่เราสามารถแสดงลักษณะใดๆ ก็ได้ ซึ่งคุณสมบัติเหล่านี้จะเป็นคุณสมบัติเฉพาะของแต่ละวัตถุซึ่งสามารถเปลี่ยนแปลงไปมาได้ เช่น เรากำหนดว่าสิ่งที่ขับเคลื่อนด้วยเครื่องยนต์และมีสี่ล้อคือรถ นี่คือเป็นการนิยามหรือนามธรรม(Abstraction)ของรถที่เรากำหนดขึ้น ดังนั้นรถกระบะ รถตู้ รถยนต์ รถแวน หรืออื่นๆ ก็คือสิ่งที่เราเรียกว่ารถ เพราะสิ่งที่เรากล่าวมานั้นขับเคลื่อนด้วยเครื่องยนต์ และมีสี่ล้อเหมือนกัน หากแต่เพียงว่ารถกระบะ รถตู้ รถยนต์ รถแวน หรืออื่นๆ ไม่ใช่นามธรรมแล้ว แต่เป็นรูปธรรมที่มีอยู่จริง ดังนั้นรถเหล่านั้นก็คือออบเจ็กต์นั่นเอง โดยส่วนประกอบภายในของออบเจ็กต์นั้นจะมีการกำหนดสถานะ (State) และพฤติกรรม(Behavior) หรือเมธทอด (Methods) เอาไว้ด้วย

สถานะของออบเจ็กต์คือสภาวะการทำงานที่ภายในออบเจ็กต์นั้นมีอยู่ ซึ่งออบเจ็กต์อื่นจะไม่สามารถเห็นสภาวะการทำงานนี้ได้ และพฤติกรรมหรือเมธทอดคือการทำงานของออบเจ็กต์นั้นว่ามีการทำงานอะไรบ้างเป็นส่วนที่ออบเจ็กต์อื่นสามารถทราบได้ว่าออบเจ็กต์นั้นสามารถทำงานอะไรได้ จะเห็นได้ชัดในหัวข้อของการนิยามหรือการนามธรรมต่อไป

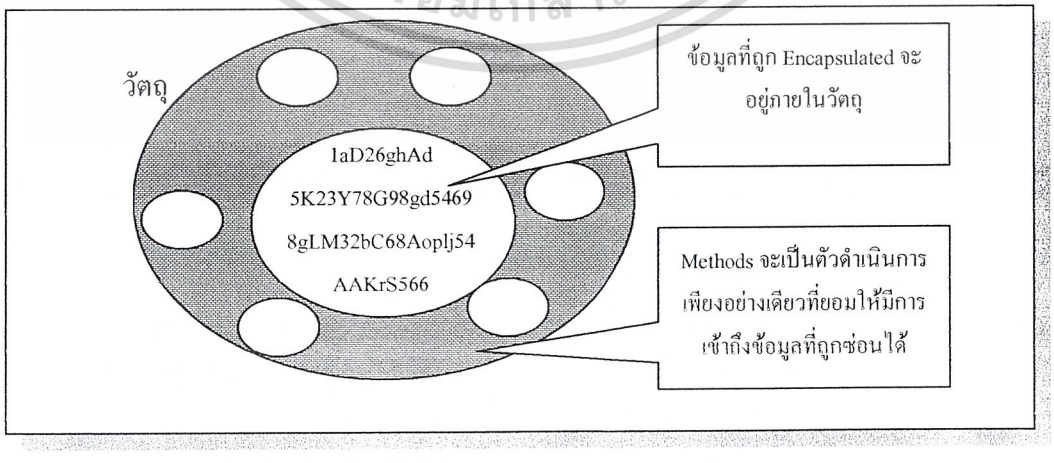
ลักษณะของออบเจ็กต์อีกอย่างที่เราให้ความสำคัญเป็นอย่างมากก็คือการสร้างส่วนที่ใช้ติดต่อหรือเชื่อมโยง (Interface) กันระหว่างออบเจ็กต์โดยที่แต่ละออบเจ็กต์จะไม่สามารถเข้าไปเกี่ยวข้องกับการทำงานของออบเจ็กต์อื่นได้ และไม่จำเป็นที่จะต้องรู้ด้วยว่าภายในออบเจ็กต์นั้นมีการทำงานเป็นอย่างไร แต่เราจะให้ความสนใจที่การเชื่อมโยงหรือติดต่อกับออบเจ็กต์นั้นเท่านั้น ซึ่งเราต้องรู้ว่าต้องป้อนอินพุท (input) อะไรเพื่อให้ได้เอาท์พุท (output) ตามที่ต้องการเท่านั้น ข้อมูลที่ใช้ในการส่งผ่านการเชื่อมโยงหรือติดต่อไปยังแต่ละออบเจ็กต์เราจะใช้คำว่า เมซเซจ (message) การที่มีการเชื่อมโยงแบบนี้ทำให้เราสามารถ

ซ่อนการทำงาน (Information hiding) และข้อมูลต่างๆ ที่ใช้ในการทำงานได้จะยกตัวอย่างการเชื่อมโยง เช่นเรามีเครื่องเล่นซีดีเราสามารถสั่งให้เล่นเพลงให้เราฟังได้ แต่เราไม่จำเป็นต้องรู้เลยว่าการทำงานของเครื่องอ่านแผ่นซีดีนั้นมีการทำงานอย่างไร มีเสียงออกมาได้อย่างไร เราจะสนใจแค่เพียงเราสั่งเล่นเพลงแล้วเครื่องเล่นซีดีจะสามารถเล่นเพลงออกมาให้เราได้ยินได้ก็เพียงพอแล้ว

2.1.2 การนามธรรม (Abstraction)

การนามธรรมคือการกำหนดขอบเขตหรือนิยามขึ้นมาให้เห็นเด่นชัดว่าสิ่งที่เรากำหนดนั้นคืออะไร แตกต่างกับสิ่งอื่นๆ อย่างไรบ้าง โดยข้อกำหนดต่างๆ ที่กำหนดขึ้นจะพิจารณาจากความเป็นจริงของสิ่งหลายๆสิ่งที่จัดอยู่ในประเภทเดียวกัน ว่ามีความเหมือนกันที่ใด สิ่งนั้นมีการทำงานเหมือนกันหรือเปล่า สิ่งนั้นใช้ในสถานการณ์เดียวกันหรือไม่ และมองข้ามสิ่งที่หลายๆ สิ่งมีเหมือนกัน แต่อาจแตกต่างในรายละเอียดเล็กๆ น้อยๆ ที่ไม่มีผลกระทบให้เกิดความแตกต่างว่าไม่ใช่สิ่งเดียวกัน ซึ่งการกำหนดขอบเขตดังกล่าวจะมีเกี่ยวข้องกับมุมมองของผู้ที่พิจารณาด้วยว่าต้องการแสดงให้เห็นถึงสิ่งนั้นในด้านใด เช่น เราพิจารณารถว่ารถแต่ละแบบที่เราใช้ๆ กันอยู่ทุกวันนี้มีลักษณะเหมือนกันที่ใดบ้าง กล่าวคือมีการขับเคลื่อนด้วยเครื่องยนต์เหมือนกัน มีสี่ล้อเหมือนกัน ใช้น้ำมันเชื้อเพลิงเป็นพลังงานในการขับเคลื่อนเครื่องยนต์ มีพวงมาลัย หรืออื่นๆ ที่เหมือนกัน แต่ถ้าเราเลือกพิจารณารถแวนกับรถตู้ ทั้งสองอย่างขับเคลื่อนด้วยเครื่องยนต์เหมือนกัน มีสี่ล้อเหมือนกัน ใช้น้ำมันเชื้อเพลิงเป็นพลังงานในการขับเคลื่อน และมีพวงมาลัยเหมือนกัน แต่มีลักษณะของตัวถังรถที่ต่างกัน มีเครื่องยนต์คนละรุ่นกัน วิ่งได้เร็วไม่เท่ากัน สิ่งที่ไม่เหมือนกันเหล่านั้นเราจะไม่นำมาพิจารณา ถึงแม้ว่าจะมีความต่างกันแต่ก็ยังคงเรียกว่ารถเหมือนกันอยู่ดี

เราสามารถที่จะกำหนดรายละเอียดเจาะจงลงไปได้ว่า ออบเจ็กต์ แต่ละตัวมีจุดเด่นอย่างไร จะเห็นได้ว่าการทำนามธรรมสิ่งต่างๆที่ดีได้นั้นจะช่วยให้เราไม่ต้องมีการทำงานที่ซ้ำซ้อน ไม่ต้องมีการสร้างใหม่หลายครั้ง และยุบเอาหลายๆ สิ่งมาเป็นสิ่งเดียวกัน แต่จะยากในเรื่องของการมองสิ่งต่างๆ เพื่อที่จะนำมากำหนดเป็นการนามธรรม การทำนามธรรมจะเป็นส่วนที่สำคัญมากในเรื่องของการออกแบบโปรแกรมแบบ ออบเจ็กต์แบ่งเป็น 2 ชนิดคือ



รูปที่ 2-1 แสดงรูปวัตถุกับการ Encapsulation

การนามธรรมสถานะ (Abstraction of State)

เป็นการทำการนามธรรม สิ่งที่เกี่ยวกับการทำงานของออบเจ็กต์ว่าแต่ละออบเจ็กต์มีการทำงานอะไรบ้าง ใช้ข้อมูลอะไรบ้างในการทำงาน (แต่ไม่กล่าวถึงว่ามีการทำงานอย่างไร) เมื่อออบเจ็กต์ได้รับเมซเซจเข้ามาทางส่วนของการเชื่อมโยงแล้วออบเจ็กต์ นั้นจะเอาข้อมูลที่ต้องการมาไว้ที่ภายในออบเจ็กต์ เพื่อให้ออบเจ็กต์อื่นไม่สามารถที่จะทราบได้ว่าออบเจ็กต์นั้นเอาข้อมูลไปทำอะไร และมีการทำงานกับข้อมูลนั้นอย่างไรบ้าง โดยเราเรียกการซ่อนรายละเอียดของการทำงานเช่นนี้ว่าการ Encapsulation ซึ่งในส่วนนี้เสมือนเป็นการเตรียมข้อมูลในกับการทำงานภายในออบเจ็กต์

การกำหนดการเข้าถึงข้อมูลภายในออบเจ็กต์จะทำการแยกส่วนของการติดต่อกับข้อมูลภายนอกและวิธีการทำงานออกจากกัน พร้อมทั้งทำการซ่อนการทำงานและรายละเอียดของข้อมูลไว้ภายใน Encapsulation ก้าวมาถึงจุดนี้เพื่อไม่ให้มีการเข้าถึงข้อมูลได้โดยตรงอีกต่อไป นั่นก็คือเป็นการซ่อนข้อมูล (Information hiding) นั้นเอง ซึ่งการเข้าถึงข้อมูลนั้นก็ต้องเรียกผ่านฟังก์ชันที่เรียกว่าเมธทอดเท่านั้น โดยการซ่อนเร้นข้อมูลนี้จะมองว่าวัตถุ(ออบเจ็กต์) แต่ละตัวเกิดจากการนำเอาข้อมูลและฟังก์ชันที่เกี่ยวข้องเข้ามารวมกันโดยมีข้อมูลเป็นแกนกลางซึ่งถูกห่อหุ้มอยู่ โดยเมธทอดจะเป็นตัวดำเนินการเพียงอย่างเดียว ที่ยอมให้มีการเข้าถึงข้อมูลที่ถูกซ่อนอยู่ได้ ดังนั้นจะเห็นว่าเราสามารถที่จะเปลี่ยนแปลงรายละเอียดของโปรแกรมได้โดยไม่ต้องเปลี่ยนแปลงส่วนเชื่อมต่อทำให้โปรแกรมใดก็ตามที่เรียกใช้ยังคงทำงานได้เหมือนเดิมซึ่งทำให้การบำรุงรักษาโปรแกรมเป็นไปได้โดยง่ายกว่าหลักการโปรแกรมแบบอื่นดูรูปที่ 2.1 ประกอบความเข้าใจของการซ่อนเร้นข้อมูลของออบเจ็กต์

ข้อดีอีกประการหนึ่งของการทำซ่อนเร้นข้อมูลก็คือทำให้เราสามารถทำการแบ่งระดับการเข้าถึงข้อมูลได้โดยตนเอง โดยจะใช้เมธทอดของวัตถุนั้นเป็นตัวแบ่งระดับการเข้าถึงข้อมูล

การนามธรรมพฤติกรรม (Abstraction of Behavior)

เป็นการนามธรรมของส่วนที่จะใช้เป็นการเชื่อมโยงกับออบเจ็กต์อื่น เพื่อให้ออบเจ็กต์อื่นทราบว่าถ้าต้องการให้ออบเจ็กต์นี้ทำงานหรือติดต่อกับออบเจ็กต์นี้จะต้องติดต่อผ่านทางส่วนของการเชื่อมโยง (Interface) ที่เรากำหนดไว้เท่านั้นจึงจะสามารถทำงานตามหน้าที่ของออบเจ็กต์นี้ได้

ภายในการทำงานของพฤติกรรม (Behavior) ของแต่ละออบเจ็กต์ที่สามารถเรียกใช้งานออบเจ็กต์อื่นเพื่อจะเอาผลลัพธ์ที่ได้จากออบเจ็กต์อื่นๆนั้นมาใช้งานต่อไป หรือยอมให้มีการใช้งานร่วมกันระหว่างออบเจ็กต์ (Collaboration Among object) ได้นั่นเอง

2.1.3 โพลิมอร์ฟิซึม (Polymorphism)

คำว่าโพลิมอร์ฟิซึมแปลว่ารูปแบบหลายๆ รูปแบบ ในทางการออกแบบโปรแกรมแบบออบเจ็กต์จะให้ความหมายของคำๆ นี้คือแต่ละออบเจ็กต์อาจที่จะใช้ชื่อในการเชื่อมโยงหรือติดต่อเหมือนกันได้ แต่การทำงานภายใน (behavior) ของชื่อที่ใช้ในการเชื่อมต่อของแต่ละออบเจ็กต์นั้นไม่เหมือนกันก็ได้ ซึ่งในการทำงานของโปรแกรมจะเป็นตัวเลือกหรือจัดการเองว่าจะเอาออบเจ็กต์ใดที่ชื่อเหมือนกันในขณะนั้นมาทำงาน ยกตัวอย่างของโพลิมอร์ฟิซึมเช่น เปลี่ยนเกียร์ คือชื่อของการทำงานที่เรากำหนดไว้ในส่วนของการ

ทำการเชื่อมโยงหรือติดต่อแต่มี ออบเจ็กต์ 2 ตัวที่ใช้ชื่อนี้ในส่วนของการติดต่อคือ อัตโนมัติ (Auto) และแบบมือ (Manual) ซึ่งการเปลี่ยนเกียร์ของเกียร์แบบอัตโนมัติจะมีการทำงานโดยใช้เครื่องยนต์ควบคุม แต่การเปลี่ยนเกียร์แบบมือจะทำงานโดยใช้คนควบคุม เป็นต้น

2.1.4 คลาส (Classes)

คลาสคือสิ่งที่เราจะต้องกำหนดขึ้นตามที่ได้นามธรรมที่เราวางไว้เพื่อให้แสดงถึงออบเจ็กต์ต่างๆ ต่อไปภายใน คลาสจะต้องมีการกำหนดการการติดต่อหรือการเชื่อมโยง มีการกำหนดโครงสร้างของข้อมูลที่ใช้ในออบเจ็กต์และรายละเอียดที่เกี่ยวกับการทำงานของเมธทอดโดยเราสามารถที่จะสร้างออบเจ็กต์หลายๆ ออบเจ็กต์ จากคลาสเพียงคลาสเดียวได้ โดยแต่ละออบเจ็กต์ที่เกิดจากคลาส เราจะเรียกว่าอินสแตนซ์(Instance) และในส่วนของการทำงานภายในของออบเจ็กต์ที่เกิดจากคลาสเราจะเรียกว่า Instantiation

2.1.5 การซ่อนเร้นข้อมูล (Encapsulation)

จากที่ได้อธิบายมาแล้วบ้างว่าออบเจ็กต์มีความสามารถที่จะซ่อนเร้นข้อมูลต่างๆ ภายในออบเจ็กต์จากออบเจ็กต์อื่นๆ ได้ซึ่งจากคุณสมบัตินี้เองมีประโยชน์เมื่อเราทำการเปลี่ยนแปลงสิ่งต่างๆ ภายในออบเจ็กต์แล้วจะไม่มีผลกระทบต่อออบเจ็กต์สืบเนื่องมาจากออบกเจ็กต์อื่นไม่สามารถที่จะอ้างถึงหรือเข้าถึงข้อมูลภายในออบเจ็กต์นั้นตรงๆ นั้นเอง สำหรับการกำหนดลักษณะการเข้าถึงนั้นมีอยู่ด้วยกัน 3 แบบ คือ Public Private และ Protect โดยแต่ละแบบมีคุณสมบัติต่างๆ ดังนี้

1. Public เป็นการกำหนดให้ออบเจ็กต์อื่นๆ สามารถที่จะเข้าถึงข้อมูลที่มีการประกาศแบบนี้
2. Private เป็นการกำหนดให้ออบเจ็กต์อื่นๆ ไม่สามารถที่จะเข้าถึงข้อมูลนี้ได้แต่จะต้องอาศัยการเข้าถึงโดยผ่านฟังก์ชัน
3. Protected เป็นแบบที่ออบเจ็กต์อื่นไม่สามารถที่จะอ้างข้อมูลที่ประกาศแบบนี้ได้แต่ออบเจ็กต์ที่ได้รับการสืบทอดมาหรือเป็นออบเจ็กต์ลูกสามารถที่จะอ้างข้อมูลแบบนี้ได้

2.1.6 ความสัมพันธ์ระหว่างคลาส (Class Relationships)

ความสัมพันธ์ระหว่างคลาสแต่ละคลาสสามารถจะแยกตามลักษณะความสัมพันธ์ระหว่างคลาสว่ามีการทำงานร่วมกันอย่างไร ซึ่งเราจะนำแต่ละคลาสมาทำงานร่วมกันตามความสัมพันธ์ที่เหมาะสมที่สุด

- คลาสหนึ่งมีอีกคลาสหนึ่ง(บรรจุ)ภายใน (The “Contain” Relationship)

เป็นลักษณะความสัมพันธ์ที่คลาสหนึ่งจะบรรจุคลาสอื่นลงไปในคลาสซึ่งคลาสที่บรรจุลงไปจะถูกซ่อนเอาไว้ไม่ให้ออบเจ็กต์หรือคลาสอื่นเห็นว่าออบเจ็กต์นี้มีคลาสอื่นอยู่ด้วย ดังนั้นมันจะไม่อยู่ในส่วนของการ Interface ซึ่งการเรียกใช้งานคลาสที่บรรจุลงไปภายในนั้นจะถูกเรียกใช้ หรือ Interface ได้เฉพาะการทำงานภายในออบเจ็กต์หรือคลาสเองเท่านั้น

- คลาสหนึ่งใช้งานอีกคลาสหนึ่ง (The "Uses" Relationship)

เป็นลักษณะความสัมพันธ์ที่คลาสหนึ่งต้องการทำงานร่วมกันคลาสอื่น โดยการส่ง Message ไปให้คลาสที่ต้องการเพื่อให้ได้ผลลัพธ์ออกมาใช้งานต่อไป ความสัมพันธ์แบบนี้เป็นการส่งผ่านข้อมูลกันระหว่างคลาสซึ่งส่วนนี้เราจะกำหนดอยู่ในส่วนของการติดต่อแล้วออบเจ็กต์อื่นสามารถมองเห็นว่าคลาสนั้นมีการส่งผ่านข้อมูลไปยังคลาสอื่นหากต้องการติดต่อหรือส่งเมสเสจ (message) ไปให้คลาสที่มีการส่งผ่านไปให้คลาสอื่นก็จะมีการทำงานเสมือนว่าส่งเมสเสจไปยังคลาสที่ถูกเรียกไปพร้อมๆ กัน

ความสัมพันธ์แบบบรรจุหรือมีอีกออบเจ็กต์อื่นภายในกับแบบเรียกใช้แตกต่างกันที่ แบบบรรจุถูกบรรจุไว้ภายในออบเจ็กต์ เพราะเราทราบแน่นอนว่าออบเจ็กต์นั้นต้องใช้ในคลาสที่ประกาศไว้ทุกครั้ง แต่ที่ใช้แบบเรียกใช้เพราะเราไม่ทราบแน่นอนว่าออบเจ็กต์ที่เรียกใช้ทุกครั้งแน่นอนหรือไม่ บางครั้งอาจจะไม่ต้องใช้งานก็ได้จึงมาใช้แบบเรียกใช้นี้

- คลาสหนึ่งสืบทอดคุณสมบัติจากอีกคลาสหนึ่ง (The "Inheritance" Relationship)

เป็นลักษณะความสัมพันธ์ที่คลาสสามารถสืบทอด Instance Variable, Interface และ Instance Methods จากคลาสอื่นๆ ได้ หมายความว่าคลาสที่ได้รับการสืบทอดหรือ Inherit มาจากคลาสอื่น ไม่ต้องมีการกำหนด Instance Variable, Interface และ Instance Methods อีกโดยสามารถที่จะเรียกใช้งานตัวแปรและการทำงานเหล่านั้นได้ทันทีคลาสที่สืบทอดมาจากคลาสอื่นเรียกว่า Derive Class หรือ Subclass ส่วนคลาสแม่หรือเรียกว่า Superclass ที่ใช้เป็นต้นแบบให้คลาสอื่นๆ สืบทอดไปนั้นไม่จำเป็นต้องมีคุณสมบัติเหมือนกับคลาสของลูกๆ แต่คลาสลูกจะต้องมีคุณสมบัติเหมือนคลาสแม่ทั้งหมด ทั้งยังสามารถเปลี่ยนแปลง และเพิ่มเติมการทำงานบางเมธทอดจากการทำงานเดิมของคลาสแม่ให้เหมาะสมกับการทำงานในส่วนอื่นๆของคลาสลูกได้ โดยลักษณะในการถ่ายทอดคุณสมบัติจะมีอยู่ 2 ลักษณะคือ *Single Inheritance* คือการที่ Subclass มี Superclass เดียวและ *Multiple Inheritance* คือการที่ Subclass ทำการสืบทอดมาจาก Superclass มากกว่าหนึ่งคลาสนั่นเอง

2.1.7 คลาสนามธรรม (Abstract Classes)

คลาสนามธรรมคือคลาสที่ใช้อธิบายถึงออบเจ็กต์บางส่วนเท่านั้น เช่น เราทำการออกแบบการส่วนที่ใช้ในการติดต่อของคลาสเอาไว้ แต่ไม่ได้นำไปใช้งาน ซึ่งส่วนที่ทำการกำหนดการติดต่อไว้จะระบุการทำงานเอาไว้ที่ออบเจ็กต์ที่เกิดจากคลาสนั้น

Program 2-1

```
Class Shape
{
    public:
        Shape(const Box& theBoundingBox)
        : itsBoundingBox(theBoundingBox) {}

        const Box& GetBoundingBox() const
        {return itsBoundingBox }

        virtual double Area() = 0;

    private:
        Box itsBoundingBox;

};
```

รูปที่ 2-2 แสดงโปรแกรม Class Shape

Program 2-2

```
Class Circle : public Shape
{
    public:
        Circle(const Point& theCenter, double theRadius)
        : itsCenter(theCenter)
        , itsRadius(theRadius)
        , Shape(
                Box(
                    Point(theCenter.GetX() - theRadius,
                          theCenter.GetY() + theRadius),
                    Point(theCenter.GetX() + theRadius,
                          theCenter.GetY() - theRadius),
        {}

        virtual double Area()  const {return pi * itsRadius * itsRadius;}

    private:
        Point    itsCenter;
        Double   itsRadius;
};
```

รูปที่ 2-3 แสดงโปรแกรม Class Circle

จากโปรแกรมภาษาซี 2-1 และ 2-2 ที่อยู่ในรูปที่ 2-2 และ 2-3 จะเห็นว่าคลาส Shape กำหนดขึ้นมาเป็นคลาสพื้นฐาน (Base Class) ให้กับคลาส Circle ซึ่งคลาส Circle จะสืบทอดคลาส Shape แบบ Public จะเห็นว่าคลาส Shape จะไม่มีอินสแตนซ์ของตัวมัน แต่จะเป็นคลาสพื้นฐานให้แก่คลาสอื่นๆ ดังนั้นคลาส Shape จะมีคุณสมบัติเป็นคลาสนามธรรม

บทที่ 3
ทฤษฎีและความรู้พื้นฐานเกี่ยวกับภาษา SQL3

SQL3 เป็นมาตรฐานของภาษา SQL ที่ออกมาเพื่อให้มีความสามารถต่างๆ เพิ่มขึ้น โดยเฉพาะความสามารถที่เกี่ยวกับออบเจ็กต์ เช่นมีการสนับสนุนการทำออบเจ็กต์คิวรี่ (Object queries) และการเชื่อมโยงติดต่อกับระหว่างออบเจ็กต์ภายใน (Object traversal) ได้โดยไม่ต้องมีการเชื่อมโยง (join) นอกจากนี้ SQL3 ได้มีการกำหนดชนิดของข้อมูลต่างๆ เพื่อตอบรับกับความต้องการชนิดของข้อมูลที่มีความซับซ้อนมากขึ้นในปัจจุบันและเพื่อนำมาใช้กับระบบฐานข้อมูลที่เป็นแบบเชิงวัตถุสัมพันธ์ด้วย

3.1 ชนิดของข้อมูลใน SQL3

3.1.1 ชนิดข้อมูลพื้นฐาน (Predefined data type) ประกอบด้วย CHARACTER,CHARACTER VARYING, CHARACTER LARGE OBJECT, BINARY LARGE OBJECT, BIT, BITVARYING, NUMERIC, DECIMAL, INTEGER, SMALLINT, ENUMERATED, FLOAT, REAL, DOUBLE PRECISION, BOOLEAN, DATE, TIME, TIMESTAMP และINTERVAL, BOOLEAN

3.1.2 Row type ในภาษา SQL3 นั้นเราสามารถที่จะกำหนดชนิดของ tuple ได้ซึ่งเรียกว่า row type โดยที่ row type ที่ถูกกำหนดขึ้นมานี้ก็คล้ายกับคลาสในทฤษฎีของออบเจ็กต์นั่นเอง เมื่อเราจะทำการกำหนดชนิดของ row type ก็จะมีรูปแบบการประกาศดังนี้

```
CREATE ROW TYPE <name>
(<component declarations>);
```

โดยที่ <name> เป็นชื่อที่กำหนดให้กับ row type ที่ต้องการสร้างและ <component declarations> เป็นการกำหนดแต่ละ คอมโพเนนท์ ภายใน row type ตัวอย่างเช่นสมมุติเราต้องการจะสร้าง row type ที่ใช้สำหรับเก็บรายละเอียดของดาราโดยกำหนดให้มีข้อมูลต่างๆ ดังนี้ ชื่อ,ถนน และ เมือง เราจะต้องทำการประกาศการสร้าง row type ดังนี้

```
CREATE ROW TYPE StarType
(
        Name    CHAR(30),
        Street  CHAR(50),
        City    CHAR(20)
);
```

ซึ่งเมื่อเราทำการสร้าง row type นี้ขึ้นมาแล้วเราก็สามารถที่จะใช้ชนิดข้อมูลนี้ได้เหมือนกับชนิดข้อมูลภายในที่มากับระบบฐานข้อมูลนอกจากนี้เราสามารถที่ประกาศให้ภายใน row type ประกอบด้วย row type อื่นอีกได้เช่นเราสามารถประกาศข้อมูลของดาราใหม่ให้เป็นดังนี้

```
CREATE ROW TYPE AddressType
(
        street    CHAR(50),
        city      CHAR(20)
);
CREATE ROW TYPE StarType
(
        name      CHAR(30),
        address   AddressType
);
```

หรืออาจจะประกาศแบบนี้

```
CREATE ROW TYPE StarType
(
        name      CHAR(30),
        address   ROW (street  CHAR(50),city  CHAR(20))
);
```

จากตัวอย่างในการสร้าง row type แบบที่สองจะเห็นส่วนที่เป็น ROW((street CHAR(50),city CHAR(20)) ก็เป็น row type ชนิดหนึ่งแต่สำหรับในกรณีนี้จะเห็นว่าไม่มีชื่อให้อ้างถึงดังนั้นเราจึงไม่สามารถที่จะนำ row type นี้ไปประกาศใช้ต่อได้

เมื่อได้ row type แล้วเราก็สามารถที่นำไปสร้างเป็นตาราง (relation) ที่มีลักษณะของ tuple เหมือนกับ row type ได้โดยมีรูปแบบของการสร้างตารางดังนี้

```
CREATE TABLE <relation_name> OF TYPE <row_type>
```

โดย <relation_name> เป็นชื่อของตารางที่ต้องการสร้างส่วน <row_type> คือชื่อของ row type ที่ได้ทำการประกาศไว้แล้ว โดยในการเข้าถึงข้อมูลที่เป็น row type ก็จะใช้วิธีอ้างแบบใช้เครื่องหมายจุด (dot notation) เหมือนกับการอ้างคอลัมน์ ในตารางกล่าวคือถ้าต้องการอ้างข้อมูลที่เป็น row type ก็จะต้อง

ใช้เครื่องจุดสองครั้งคือหนึ่งสำหรับอ้างคอลัมภ์และอีกหนึ่งสำหรับอ้างข้อมูลใน row type อีกทีแต่ในการอ้างข้อมูลใน row type จะต้องใช้ 2 จุดเช่นจากตัวอย่างที่ผ่านมาเราสามารถที่จะเข้าถึงข้อมูลได้ดังนี้

```
SELECT MovieStar.name,MovieStar.address..street
FROM MovieStar
WHERE MovieStar.address..city='Beverly Hills';
```

จากตัวอย่างข้างบนคือตัวอย่างของ งคิวรี่ที่ต้องการรายชื่อของดาราและถนนที่ดาราคนนั้นอาศัยอยู่โดยเอาเฉพาะดาราที่อาศัยในเมือง Beverly Hills เท่านั้น

3.1.3 Abstract data type (ADT) เป็นชนิดข้อมูลแบบออบเจ็กต์ ซึ่งชนิดข้อมูลนี้เราสามารถที่จะระบุให้มีแอททริบิวต์และการดำเนินการต่างๆ ซึ่งจะใช้สำหรับการทำเปรียบเทียบหรือเรียงข้อมูล (order ของ ADT) และใช้สำหรับการติดต่อกับภายนอก ซึ่งความแตกต่างระหว่าง ADT กับ row type ก็คือ row type ไม่มีคุณสมบัติในเรื่องของการซ่อนเร้นข้อมูลภายใน (encapsulation) โดยรูปแบบการทำ ADT ใน SQL3 สามารถที่จะทำได้ดังนี้

```
CREATE TYPE <type_name>
(
        list of attribute and their types
        optional declaration of = and <ฟังก์ชันfor the type
        declaration ofฟังก์ชัน(method) for the type
);
```

โดย <type_name> คือชื่อของ ADT ที่ต้องการประกาศ จากนั้นก็เป็นการประกาศแอททริบิวต์ต่างๆ และชนิดข้อมูลของแต่ละแอททริบิวต์และตามด้วยการประกาศฟังก์ชันที่ใช้สำหรับนำมาใช้ในการเปรียบเทียบซึ่งอนุญาติให้ประกาศได้สองเครื่องหมายคือ = และ < โดยในการประกาศจะใช้คำว่า EQUALS,LESS THAN และตามด้วยชื่อของฟังก์ชันที่เราสร้างขึ้นมาเพื่อสนับสนุนการเปรียบเทียบซึ่งการประกาศฟังก์ชันนี้เป็นส่วนที่สามารถจะเพิ่มเติม (option) ซึ่งถ้าเราไม่ได้สร้างฟังก์ชันนี้ขึ้นมาก็สามารถที่จะใช้คำว่า DEFAULT เพื่อให้ ADT ใช้ฟังก์ชันที่มีอยู่แล้วกับ ADT นี้ได้เลย แต่ถ้าไม่ต้องการให้ใช้เครื่องเหล่านี้กับ ADT ที่เรากำหนดก็ใช้คำว่า NONE แทน และส่วนสุดท้ายของการประกาศ ADT ก็คือส่วนของการประกาศฟังก์ชันต่างๆ ที่ต้องการสร้างขึ้นเพื่อใช้กับข้อมูลของ ADT ของที่ประกาศ

3.1.3.1 ชนิดข้อมูลแบบ Distinct เป็นการกำหนดชนิดของข้อมูลขึ้นมาใหม่โดยการใช้คุณสมบัติและโครงสร้างของข้อมูลชนิดอื่นที่มีอยู่แล้ว (source type) โดยใน SQL3 มาตรฐานนั้นได้กำหนดให้ชนิด

ข้อมูลแบบที่เรียกว่า distinct descriptor จะเป็น ADT ที่มีคอมโพเนนท์ต่างๆ เหมือนกัน descriptor ของชนิดข้อมูลต้นแบบหรือต้นฉบับ (source type) ที่เรากำหนดให้กับชนิดข้อมูลแบบ distinct

3.1.3.2 Encapsulation คือความสามารถที่จะกำหนดระดับการเข้าถึงของคอมโพเนนท์ต่างๆ ภายใน ADT ได้หลายระดับดังนี้ public,private และ protected

3.1.3.3 Observers และ mutators เป็นฟังก์ชันที่จะถูกกำหนดให้กับ ADT เมื่อมีการสร้าง ADT ขึ้นมาโดยที่ Observer ฟังก์ชันจะถูกกำหนดให้มีชื่อเดียวกับแอททริบิวต์ของ ADT และมีพารามิเตอร์ 1 ตัว ซึ่งเป็นชนิดเดียวกับ ADT ที่กำหนดโดยฟังก์ชันนี้จะใช้สำหรับส่งค่ากลับ (return)เป็นค่าของแอททริบิวต์นั้น ส่วนฟังก์ชัน mutators จะมีชื่อเดียวกับแอททริบิวต์เหมือนกันแต่จะมี 2 พารามิเตอร์โดยที่ตัวแรกจะเป็นชนิดเดียวกับ ADT ที่กำหนดและตัวที่สองจะเป็นชนิดข้อมูลของแอททริบิวต์นั้น ซึ่งฟังก์ชันนี้จะทำการกำหนดค่าใหม่ให้กับแอททริบิวต์โดยใช้ค่าจากพารามิเตอร์ตัวที่ 2 และจะทำการส่งค่านี้กลับมาให้ด้วย โดยระดับของการซ่อนเร้นของทั้งสองฟังก์ชันจะขึ้นอยู่กับระดับการกำหนดการซ่อนเร้น (encapsulate) ของแอททริบิวต์ใน ADT ของฟังก์ชันนั้นๆว่าอยู่ระดับใด

3.1.3..4 Constructors เป็นฟังก์ชันที่กำหนดให้ ADT โดยฟังก์ชัน constructor จะถูกกำหนดโดยอัตโนมัติโดยระบบฐานข้อมูลเพื่อใช้สร้างอินสแตนซ์ของ ADT โดยชื่อของ constructor จะมีชื่อเดียวกับ ADT ที่เรากำหนด

3.1.3.5 Subtypes และ supertype ของ ADT เป็นการทำถ่ายทอดคุณสมบัติของ ADT ที่สามารถที่จะให้มีการสืบทอดคุณสมบัติต่างๆของ ADT หนึ่งไปให้กับ ADT หนึ่งได้ โดยสมมุติเรากำหนดให้ ADT Ta จะเป็น subtype ของ ADT Tb ถ้าในส่วนของการประกาศ Ta ได้บ่งชี้ว่าตัวมันเป็น subtype ของ Tb แล้ว Ta จะถูกเรียกว่าเป็น direct subtype ของ Tb โดยที่ข้อมูลชนิด Ta จะเป็น subtype ของ Tb ก็ต่อเมื่อ

- Ta และ Tb เป็นชนิดข้อมูลที่เหมือนกัน
- Ta เป็น direct subtype ของ Tb หรือ
- ถ้ามีข้อมูลชนิด Tc อยู่ และ Ta เป็น direct subtype ของ Tc และ Tc เป็น subtype ของ Tb

ชนิดข้อมูล Tb จะถูกเรียกว่าเป็น supertype ของ Ta ถ้า Ta เป็น subtype ของ Tb ถ้า Ta เป็น direct subtype ของ Tb แล้ว Tb จะถูกเรียกว่าเป็น direct supertype ของ Ta สำหรับชนิดข้อมูลที่ไม่ได้เป็น subtype ของ type ใดๆ เราจะเรียกว่า maximal supertype

ถ้าให้ Ta เป็น maximal subtype และให้ T เป็น subtype ของ Ta กลุ่มของ subtype ของ Ta โดยรวมชนิดข้อมูล Ta ด้วย เราจะเรียกกลุ่มของชนิดข้อมูลนี้ว่าเป็น subtype family ของ T หรือ ของ Ta ส่วน leaf type คือชนิดข้อมูลใดๆที่ไม่มี subtype เลย

อินสแตนซ์ของ subtype ใดๆ จะเป็นอินสแตนซ์ของ supertype ของมันด้วย โดยแต่ละอินสแตนซ์จะมีคุณสมบัติเฉพาะเจาะจงที่ถูกกำหนดโดยชนิดข้อมูลนั้นๆ ซึ่งในการที่จะสร้างอินสแตนซ์ ขึ้นมาเราไม่จำเป็นที่จะใช้ชนิดข้อมูลที่ leaf type เท่านั้น เช่นสมมุติเรามีชนิดข้อมูล PERSON โดยมี STUDENT และ EMPLOYEE เป็น subtype และ STUDENT ยังมี UG_STUDENT และ PG_STUDENT เป็น subtype

อีกที ในการสร้างอินสแตนซ์เราอาจจะใช้ชนิดข้อมูล STUDENT ได้โดยไม่จำเป็นต้องเป็นชนิดข้อมูลที่ leaf type

ถ้า Ta เป็น subtype ของ Tb แล้วอินสแตนซ์ของ Ta สามารถที่จะถูกใช้แทนอินสแตนซ์ของ Tb เช่นเราสามารถที่จะกำหนดค่าของ Tb ด้วยค่าของ Ta หรืออาจจะทำการส่งค่า Ta ให้กับอากิวเมนต์ (argument) ของฟังก์ชันที่เป็นชนิดของ Tb ได้

ในการที่จะทำการประการชนิดข้อมูลให้เป็น subtype ของชนิดข้อมูลอื่นๆ นั้นจะต้องมีการประกาศหรือใช้คำสั่ง UNDER ก่อน โดยชนิดข้อมูลสามารถที่จะมี subtype ได้หลาย subtype เช่นเดียวกับ supertype ที่ชนิดข้อมูลหนึ่งๆ จะมีหลาย supertype ได้

3.1.3.6 Type templates เป็นชนิดข้อมูลที่ใช้ในการกำหนดกลุ่มของ ADT โดยในการกำหนด type template แต่ละตัวจะประกอบด้วยกลุ่มของ formal พารามิเตอร์และ body โดยในส่วนของพารามิเตอร์ จะใช้ในการกำหนดค่าหรือชนิดของข้อมูลที่ต้องใช้ในการนำ type template ไปใช้ในการสร้างชนิดข้อมูล (generate type (ADT type)) ส่วนของ body จะส่วนที่ใช้ในการกำหนดใน ADT

ตัวอย่าง การสร้าง ADT

```
CREATE TYPE AddressADT
(
        street    CHAR(50),
        city      CHAR(20),
        EQUAL addrEq,
        LESS    THAN  NONE
);
```

3.1.4 ชนิดข้อมูลแบบ Collection เป็นชนิดข้อมูลซึ่งประกอบด้วยสมาชิก (elements) ของข้อมูลอยู่ภายใน โดยสมาชิกของ collection จะต้องเป็นข้อมูลชนิดเดียวกันโดยทุกสมาชิกของ collection จะอยู่ในคอลัมน์เดียวกันหมด โดยข้อมูลชนิด collection แบ่งออกเป็น 3 ชนิดคือ set,multiset และ list โดยชนิดของข้อมูลใน collection อาจจะเป็นชนิดข้อมูลพื้นฐาน , ADT หรือ collection เองก็ได้โดยที่ชนิดข้อมูลแบบต่างๆ แสดงไว้ในตาราง 2.1

Collection data type	คุณสมบัติ
Set	เป็นชนิดของ collection type ที่ไม่มีลำดับของข้อมูลและข้อมูลภายในจะซ้ำกันไม่ได้
Multiset	มีลักษณะคล้ายกับชนิดข้อมูลแบบ set แต่สามารถที่จะมีข้อมูลที่ซ้ำกันได้
List	เป็น collection type ที่มีลำดับของข้อมูลและสามารถที่จะมีค่าข้อมูลซ้ำกันได้

ตาราง 2-1 แสดงชนิดของข้อมูล Collection แบบต่างๆ

3.1.5 ข้อมูลแบบใช้อ้างอิง (Reference) เป็นข้อมูลชนิดหนึ่งซึ่งมีประโยชน์ในการจัดการกับข้อมูล ซึ่งจะเปรียบเสมือนชนิดข้อมูลแบบตัวชี้ (pointer) ในภาษา C และ C++ นั่นเอง โดยมีวิธีการประการดังนี้

<component_name> REF (<type>);

โดยที่ <component_name> คือชื่อของคอมโพเนนท์ ที่ต้องการประกาศให้มีชนิดเป็น reference ส่วน <type> คือชนิดของข้อมูลที่ต้องการให้คอมโพเนนท์นี้ชี้ ตัวอย่างเช่น

bestMovie REF(MovieType);

ซึ่งมีความหมายว่า bestMovie เป็นคอมโพเนนท์ชนิด reference ที่ใช้ชี้ไปยังข้อมูลชนิด MovieType วิธีการเข้าถึงข้อมูลที่ bestMovie ชี้อยู่ทำได้ด้วยการระบุชื่อและตามด้วยสัญลักษณ์ -> แล้วตามด้วยชื่อของ คอมโพเนนท์ ที่อยู่ใน MovieType ได้เลย (เหมือนการอ้างถึงของข้อมูล Reference ในภาษา C++)

ข้อมูลแบบ reference นี้สามารถประยุกต์ใช้ได้กับการสร้างความสัมพันธ์แบบ many-to-many ได้ ในมาตรฐานของ SQL รุ่นเก่าๆ การสร้างความสัมพันธ์แบบ many-to-many สามารถทำได้โดยการแยกตารางออกมาหนึ่งตารางโดยที่ข้อมูลในตารางก็คือการจับคู่ระหว่าง key ของแต่ละ tuple ที่มีความสัมพันธ์กัน แต่สำหรับใน SQL3 แล้วยอมให้มีการใช้ข้อมูลแบบ reference แทนการใช้ key เพื่อใช้ในการอ้างถึงออบเจ็กต์ที่มีความสัมพันธ์กันได้โดยตรง

หากเราต้องการสร้างข้อมูลที่เก็บรายละเอียดของภาพยนต์แต่ละเรื่องและรายละเอียดของดาราแต่ละคน โดยที่หนังหนึ่งเรื่องจะประกอบด้วยดาราหลายคนและในขณะเดียวกัน ดาราหนึ่งคนก็สามารถแสดงหนังได้หลายเรื่อง จะเห็นว่าเป็นความสัมพันธ์แบบ Many-to-many เราสามารถนำข้อมูลชนิด reference มาใช้ได้ดังนี้

```
CREATE ROW TYPE MovieType
(
        Title     CHAR(30),
        Year      INTEGER,
        InColor   BIT(1)
);
CREATE ROW TYPE AddressType
(
        Street    CHAR(50),
        City      CHAR(20)
);

CREATE ROW TYPE StarType
(
        Name      CHAR(50),
        Address   AddressType
);

CREATE ROW TYPE StartInType
(
        Start     REF(StarType),
        Movie     REF(MovieType)
);

CREATE TABLE Movie OF TYPE MovieType;
CREATE TABLE MovieStar OF TYPE StarType;
CREATE TABLE StarIn OF TYPE StarInType;
```

จากตัวอย่างข้างต้นเป็นการสร้างความสัมพันธ์จากที่ได้กล่าวไว้ ซึ่งหากเราต้องการทราบรายชื่อของหนังทั้งหมดที่แสดงโดย Mel Gibson สามารถจะใช้คิวรี่ดังนี้

```
SELECT Movie->title
FROM StarIn
WHERE Star->name='Mel Gibson';
```

3.2 เปรียบเทียบ SQL3 กับ SQL92

นอกจากข้อดีต่างๆ ของภาษา SQL3 ที่ได้กล่าวไป SQL3 ยังมีข้อดีในการที่จะโค้ดง่ายกว่าภาษา SQL92 เนื่องจาก SQL3 มี "object navigation query" (การทำคิวรี่โดยอาศัย OID เพื่ออ้างถึงออบเจ็กต์อื่นๆ) ที่สามารถที่จะนำมาใช้ในการทำคิวรี่แทนการทำ "join query" ของภาษา SQL92 ดังตัวอย่างคิวรี่ต่อไปนี้

```
SELECT id,name,state,dno(major),name(major),building(major)
FROM Sutent
```

จากตัวอย่างคิวรี่ที่เห็นจะเห็นว่า id, name และ state เป็นแอททริบิวต์ของออบเจ็กต์ student ส่วน department number,department name และ building เป็นแอททริบิวต์ที่มาจากออบเจ็กต์ department โดยอาศัยความสัมพันธ์จาก major ที่อยู่ในออบเจ็กต์ student ในการที่จะเอาข้อมูลจากออบเจ็กต์ department อีกที่หนึ่ง ซึ่งจะเห็นว่าเป็นการแสดงข้อมูลของ department โดยไม่ต้องมีการอ้างออบเจ็กต์ department ตรงๆ ซึ่งถ้าเป็นการทำคิวรี่เดียวกับนี้โดยการใช้ภาษา SQL92 เราจะต้องทำการป้อนคิวรี่ดังนี้

```
SELECT s.id, s.name, s.state, d.dno, d.name, d.building
FROM Department d, Student s
WHERE s.majorDept=d.deptNo
UNION ALL
SELECT s.id, s.name, s.state, d.dno, d.name, d.building
FROM Department d, TA s
WHERE s.majorDept=d.deptNo
```

ซึ่งนอกจากนี้ข้อดีดังกล่าวแล้ว SQL3 ยังมีความสามารถที่จะเรียกฟังก์ชันที่ผู้ใช้เป็นคนสร้าง (user-derfined method) เช่นสมมุติเราต้องการที่จะทำการคิวรี่หาระยะทางระหว่าง staff member (โดยที่ staff เป็นออบเจ็กต์ที่อยู่ใน staff คลาส) กับ staff member ที่มี ID 6966.เราสามารถทำได้ดังนี้

SQL3	***SQL92***
SELECT distance(s1.place,s2.place) FROM Staff s1,Staff s2 WHERE s1.id=6966	SELECT SQRT((s1.latitude-s2.latitude)* (s1.latitude-s2.latitude)+ (s1.longtitude-s2.longtitude)* (s1.longtitude-s2.longtitude)) FROM Staff s1,Staff s2 WHERE s1.id=6966

ตาราง 2-2 เปรียบเทียบภาษา SQL3 กับ SQL92

3.3 SQL3 กับการเรียกตัวเอง (SQL3 & Recursion)

SQL3 มีความสามารถในการที่ทำคิวรี่ในลักษณะของการเรียกตัวเอง (Recusive) กล่าวคือมีการเรียกตัวเอง ซึ่งจากความสามารถดังกล่าวนี้เองจะมีประโยชน์มากในการช่วยลดจำนวนข้อมูลที่จะเก็บในตาราง เช่นสมมุติว่าเรามีตารางที่เก็บข้อมูลของสายการบินของ Untried Airlines(UA) และ Arcane Airlines(AA) ซึ่งมีเที่ยวบินไปเมืองต่างๆ ดังนี้ San Francisco,Denver,Dallas,Chicago และ New York โดยมีการเก็บข้อมูลในตาราง ซึ่งประกอบด้วยแอททริบิวต์ดังนี้ สายการบิน,เมืองต้นทาง,เมืองปลายทาง,เวลาออกและเวลาที่ถึง ถ้าเราต้องการทราบว่าเราสามารถที่จะเดินทางจากเมืองไหนไปอีกเมืองไหนได้บ้างจะเห็นว่าในการที่จะตอบปัญหาเช่นนี้จะต้องมีการเก็บข้อมูลว่าแต่ละเมืองสามารถที่จะเดินทางไปเมืองใดได้บ้างแต่ในกรณีที่เราใช้คิวรี่แบบเรียกตัวเองเราสามารถที่จะเก็บเฉพาะเมืองที่สามารถที่จะเดินถึงกันโดยตรงดังนี้

Airline	From	To	Depart	Arrives
UA	SF	DEN	9:30	12:30
AA	SF	DAL	9:00	14:30
UA	DEN	CHI	15:00	18:00
UA	DEN	DAL	14:00	17:00
AA	DAL	CHI	15:30	17:30
AA	DAL	NY	15:00	19:30
AA	CHI	NY	19:00	22:00
UA	CHI	NY	18:30	21:30

ตาราง 2-3 แสดงถึงตาราง Flights

ซึ่งจากคำถามข้างต้นเราสามารถที่จะเขียนแสดงในรูปของกฎ (Rule) คล้ายๆ กับการเขียนกฎในภาษา Prolog เพื่อที่จะทำการ resolution ได้ดังนี้

1. Reaches(x,y) <- Flights(a,x,y,d,r)
2. Reaches(x,y) <- Reaches(x,z) AND Reaches(z,y)

โดยที่กฎข้อที่ 1 เป็นการระบุให้ Reaches ประกอบด้วยคู่ลำดับของเมืองที่มีสายการบินเดินทางถึงกันได้โดยตรง ส่วนกฎข้อที่ 2. เป็นการหาเมืองต่างๆ ที่สามารถที่จะเดินทางถึงกันได้โดยอาศัยผลที่ได้จากกฎข้อ 1 อีกทีหนึ่ง ซึ่งจากกฎข้อที่ 1 เป็นการเริ่มหาข้อเท็จจริง (Fact) ต่างๆ มาจากตาราง Fligths เราเรียก Relation ที่เริ่มต้นนี้ว่า extensional database relation (EBR) และเรียก Relation ที่ได้จากการทำ resolution จะกฎข้อ 2 เรียกว่า intensional database relation (IBR)

ซึ่งจาก EBR และ IBR เราสามารถที่จะทำการแปลงเป็นภาษา SQL3 โดยใช้ คำสั่ง WITH ซึ่งจะได้เป็นคิวรี่ดังนี้

```
WITH RECURSIVE Reaches(frm,to) AS
        (SELECT frm, to FROM Flights)
UNION
        (SELECT R1.frm, R2.to
         FROM Reaches AS R1, Reaches AS R2
         WHERE R1.to=R2.frm)
SELECT * FROM Reaches;
```

ซึ่งในการคิดคิวรี่แบบเรียกตัวเองให้คิดถึงการทำงานของภาษา Prolog ในการที่จะทำการ resolution กฎต่างๆ ที่เราได้กำหนดให้จากดังที่ได้แสดงไว้ในตัวอย่างจากนั้นก็ทำการแปลงจากกฎที่เรากำหนดมาเป็นคำสั่ง SQL โดยใช้คำสั่ง WITH

บทที่ 4
ทฤษฎีและความรู้พื้นฐานเกี่ยวกับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (ORDBMS)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เป็นเทคโนโลยีใหม่ที่ใช้ในการออกแบบระบบฐานข้อมูลแบบที่รองรับชนิดข้อมูลหลายๆ แบบ (Universal database management systems) โดย ORDBMS เป็นระบบที่ได้รวมข้อดีต่างๆ ของระบบฐานข้อมูลแบบเชิงวัตถุ และระบบฐานข้อมูลแบบสัมพันธ์เข้าด้วยกัน ซึ่งในการที่จะทำให้ระบบฐานข้อมูลแบบสัมพันธ์เป็นแบบ ORDBMS จะต้องทำให้ระบบฐานข้อมูลนี้สนับสนุนในเรื่องของการซ่อนเร้นข้อมูล (Encapsulation) การสืบทอดคุณสมบัติ (Inheritance) และ การทำโพลิมอร์ฟีซึม (Polymorphism) ที่เป็นคุณสมบัติสำคัญๆ ของออบเจ็กต์ ซึ่งลักษณะของโครงสร้างหรือสถาปัตยกรรมของระบบฐานข้อมูลแบบใหม่นี้ที่มีอยู่กันในปัจจุบันนั้นสามารถจัดประเภทได้ดังนี้

1. แบบที่มีโค้ด plug-in ที่ใช้สำหรับสร้างฟังก์ชั่นต่างๆ หรือเรียกโปรแกรม (application) อื่นๆ ที่ จัดการกับออบเจ็กต์ได้
2. แบบเพิ่มส่วนของ APIs และ server subsystem เพื่อใช้ในการสนับสนุนหน้าที่ในส่วนของออบเจ็กต์
3. แบบที่มีการทำจำลอง (simulate) ส่วนของหน้าที่การทำงานแบบเชิงวัตถุสัมพันธ์ในส่วนของ middleware layer
4. แบบที่ทำการออกแบบตัว database engine ใหม่เพื่อให้สนับสนุนการทำงานในส่วนของออบเจ็กต์ ที่ database engine เลย
5. แบบที่เพิ่มส่วนของ object-oriented layer ที่สนับสนุน rich datatype บน relational database engine อีกทีหนึ่ง

จากการที่เป็นระบบฐานข้อมูลนี้รวมความสามารถของระบบฐานข้อมูลสัมพันธ์และระบบฐานข้อมูลเชิงวัตถุ ทำให้ระบบฐานข้อมูลแบบนี้สามารถที่จะมีแอททริบิวต์เป็นออบเจ็กต์ได้และสามารถที่จะทำให้ในคอลัมน์หนึ่งประกอบด้วยหลายๆ ออบเจ็กต์ ซึ่งจากลักษณะที่มีหลายๆ ออบเจ็กต์อยู่ในคอลัมน์เดียวกันนี้ดูเหมือนจะทำให้คอลัมน์นั้นไม่อะตอมมิกซ์ (non-atomic) แต่จริงแล้วการที่จะทำให้คอลัมน์ใดประกอบด้วยออบเจ็กต์หลายๆ ออบเจ็กต์นั้นจะต้องมีการกำหนดชนิดข้อมูลที่เป็นชนิดข้อมูลที่บรรจุข้อมูลอื่นๆ ไว้ภายใน (Collection data type) ได้และตัวชนิดข้อมูลนี้เองจะถูกมองเป็นออบเจ็กต์เดียวหรือเรียกว่าเป็นคอนเทรนเนอร์ (Container) ออบเจ็กต์และตัวคอนเทรนเนอร์นี้เองที่เราได้กำหนดให้เป็นชนิดข้อมูลของคอลัมน์ก็ทำให้คอลัมน์นี้สามารถที่จะบรรจุออบเจ็กต์หลายๆออบเจ็กต์ได้ตามคุณสมบัติของคอนเทรนเนอร์นั้นเอง

4.1 ส่วนประกอบของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์

ในการสนับสนุนการทำงานในส่วนของเชิงวัตถุสัมพันธ์ของในระบบ DBMS จะต้องมีการเพิ่มเติมหรือแก้ไขในส่วนต่างๆ ดังนี้

- Access method
- Executor
- Parallelism system
- Optimizer
- Parser

4.1.1 วิธีการเข้าถึง (Access methods)

ระบบฐานข้อมูลแบบสัมพันธ์ทั่วๆ ไปนั้นจะมีการสนับสนุนการเข้าถึงข้อมูลแบบ B-tree เพื่อใช้ในการทำ optimize keyed access method ของเรคคอร์ดซึ่งในระบบ ORDBMS นี้ในส่วนของการเข้าถึงข้อมูลนี้มีความจำเป็นที่จะต้องเพิ่มความสามารถในการสนับสนุนชนิดข้อมูลใหม่ๆ ที่มีในระบบฐานข้อมูลแบบนี้ ซึ่งแบบ B-tree ที่มีอยู่เดิมนี้จะมีการเก็บอินเด็กซ์เรคอร์ดไว้ในลักษณะที่ถูกกำหนดโดยตัวดำเนินการ (operator) < (น้อยกว่า) ที่กระทำกับข้อมูลที่ใช้อินเด็กซ์แบบนี้โดยในการกำหนดการทำงานของตัวดำเนินการนี้กับชนิดข้อมูลแต่ละแบบนี้จะถูกเก็บไว้ในส่วนข้อมูลเกี่ยวกับข้อมูล(meta data)ในรายการของระบบ (system catalogs)

ซึ่งลักษณะของข้อมูลชนิดที่มีขึ้นในระบบแบบ ORDBMS นี้แบบชนิดที่มีความซับซ้อน ในการที่จะค้นหาหรือจะเข้าถึงข้อมูล ไม่สามารถที่จะใช้วิธีการง่ายๆ แบบ B-tree ได้ ยกตัวอย่างเช่นชนิดของข้อมูลที่ใช้สำหรับจัดเก็บจุดต่างๆ ในแผนที่ซึ่งจะเห็นว่าจะต้องใช้การค้นหาที่มีลักษณะเป็นแบบ 2 มิติไม่สามารถที่จะอาศัยการทำ optimize ที่ใช้วิธีการแบบ 1 มิติเหมือนแบบ B-tree ได้ ดังนั้นจึงเกิดความคิดหรือวิธีการต่างๆ ที่เหมาะสมกว่า เช่นมีการใช้วิธีการแบบ R-tree,quad tree หรือแบบ grid file ซึ่งจากสาเหตุนี้จึงทำให้ระบบฐานข้อมูลแบบ ORDBMS จะต้องมี engine ที่สามารถสนับสนุนการเพิ่มเติมวิธีการเข้าถึงข้อมูลแบบต่างๆ ขึ้นมาได้ ซึ่งรวมไปถึงว่าสามารถที่จะสนับสนุนวิธีการเข้าถึงที่เขียนโดยเครื่องมือต่างๆ ของ ผลิตภัณฑ์ต่างๆ ที่มีอยู่ (third parties) ได้ ซึ่งเราเรียกความสามารถนี้ว่าเป็นความสามารถที่สนับสนุนการกำหนดวิธีการเข้าถึงข้อมูลโดยผู้ใช้เป็นคนกำหนด (user-define access method)

โดยในการสนับสนุนการกำหนดวิธีการเข้าถึง นี้ไม่จำเป็นต้องมีการทำโค้ด (hard-code) ลงไว้ใน DBMS กล่าวคืออาจจะสามารถที่จะให้มีการนิยาม (abstact) วิธีการเข้าถึงที่ต้องการโดยที่ DBMS สามารถที่จะเข้าใจนิยามที่เราต้องการได้

จากตัวอย่างของต่อไปเป็น Object-Relational Schema ที่เกี่ยวกับข้อมูลของลูกจ้างโดยประกอบด้วยข้อมูลดังนี้คือ ชื่อ อายุ เงินเดือน แผนก ตำแหน่งที่อยู่ของลูกจ้างโดยกำหนดให้เป็นชนิดข้อมูลที่เก็บตำแหน่ง และรูปภาพของลูกจ้างดังจะแสดงได้ดังนี้

```
CREATE EMP-OR (
        name = C12,
        age = int ,
        salary = int ,
        dept =C12,
        location = point,
        picture = image
);
```

จากตัวอย่างนี้ได้มีการกำหนดให้มีชนิดข้อมูลแบบจุด (point) ที่เป็นชนิดข้อมูลที่เก็บตำแหน่งที่อยู่ของลูกจ้างและชนิดข้อมูลรูปภาพ (image) เป็นรูปภาพของลูกจ้างซึ่งจาก schema นี้เราสามารถที่จะทำการคิวรี่ข้อมูลจากตารางนี้ดังนี้

```
SELECT name
FROM   EMP-OR
WHERE beard(picture) > 0.7 and
Age > 60 and location in circle ("10,10",5);
```

จากคิวรี่นี้เป็นการต้องการหาลูกจ้างที่มีเครา อายุมากกว่า 60 ปีและอาศัยอยู่ในบริเวณ 5 ไมค์จากจุดตำแหน่ง 10,10 โดยคิวรี่นี้จะใช้ตัวดำเนินการ (user-define operator (in)) และ ฟังก์ชันที่ผู้ใช้กำหนดขึ้นมา (user-define function (beard)) สำหรับกระทำกับชนิดของข้อมูลใหม่ที่เพิ่มเติมขึ้นมา

จากคิวรี่นี้เองจะเห็นว่ามีการพิจารณาหรือค้นหาลูกจ้างที่มีเครา ซึ่งจะเห็นว่าเราไม่สามารถที่จะใช้วิธีการแบบ B-tree ในการทำอินเด็กซ์บน pixel ของรูปภาพได้ ถ้าฟังก์ชันที่ชื่อ beard นี้ทำการคำนวณหรือวิเคราะห์ภาพแล้วทำการให้ค่าเป็นตัวเลขทศนิยมมาให้ ระบบฐานข้อมูลแบบ ORDBMS ดีๆ ก็จะต้องอนุญาตให้เราสามารถที่จะกำหนดให้มีการทำ index บนฟังก์ชันนี้ได้หรือกล่าวง่ายๆ ก็คือสามารถที่จะให้มีการทำอินเด็กซ์บนฟังก์ชันที่กำหนดขึ้นมาได้ที่กระทำกับแอททริบิวต์ที่เราต้องการได้นั้นเอง

4.1.2 ตัวจัดการ (Executor)

ตัวจัดการจะต้องมีความสามารถที่จะทำการแก้ปัญหา (solving) คิวรี่ที่รับมาได้ เช่น คิวรี่ EMP-OR จากตัวอย่างที่ผ่านมาที่ประกอบด้วยฟังก์ชันและชนิดข้อมูลที่ผู้ใช้กำหนดขึ้นมาเองซึ่งหมายถึงต้องมีส่วนที่เรียกว่าตารางของตัวจัดการ (table-driven executor) ที่สามารถจะทดสอบหรือตรวจสอบข้อมูลของข้อมูล (metadata) ใน รายการของระบบ (system catalogs) และทำการเรียกใช้ฟังก์ชันและตัวดำเนินการได้ถูกต้อง ดังนั้น ชนิดข้อมูลต่างๆ ที่เป็นแบบนี้จะต้องมีการเขียนโปรแกรมขึ้นมา (hard-coding datatype) ใน

SQL92 ที่ใช้กันในระบบฐานข้อมูลแบบสัมพันธ์ ตัวจัดการ (executor) ก็สามารถที่จะแก้ไขหรือแทนด้วยวิธีการนี้กล่าวคือด้วยโค้ดที่มีความยืดหยุ่นกว่า

มากไปกว่านั้น ตัวระบบฐานข้อมูลควรจะสามารถที่จะทำการยกเลิกหรือจะทำการเชื่อมโยง (relink) กับชนิดข้อมูล ฟังก์ชัน หรือ ตัวดำเนินการ ใหม่ๆ หรือไม่ก็ได้ซึ่งเราเรียกความสามารถดังกล่าวนี้ว่าการเชื่อมโยงและยกเลิกการเชื่อมโยงในขณะใดขณะหนึ่ง (dynamically link and unlink)

4.1.3 ระบบที่จัดการการทำงานแบบขนาน (Parallelism System)

ตัวของระบบฐานข้อมูลสัมพันธ์ (Relational engine) ส่วนมากจะสนับสนุนการทำงานแบบขนานกันภายใน (intraquery parallelism) ในกรณีที่ตารางของระบบฐานข้อมูลแบบสัมพันธ์ (relational table) มีการเก็บจัดข้อมูลเป็นแบบแยกกันหลายส่วน (multiple storage fragments) และมีการทำคิวรี่ ที่วิ่ง (run) ในลักษณะขนานกับในแต่ละส่วน (fragment) ในสภาวะของระบบที่เป็นแบบมัลติโปรเซสเซอร์ (multiprocessor) เพื่อลดเวลาของการตอบสนอง (response time) ของคิวรี่

ส่วนในระบบฐานข้อมูลแบบ ORDBMS นั้นในการที่จะทำงานแบบขนาน (parallelism) นั้นจะต้องมีการเพิ่มเติมความสามารถต่างๆ จากระบบฐานข้อมูลแบบสัมพันธ์ ยกตัวอย่างเช่น ในการทำงานของตัวฐานข้อมูลแบบสัมพันธ์ (relational engine) ที่สามารถจะทำการเรียก (execute) ส่วนที่เป็นการทำงานในลักษณะรวมกันของข้อมูล (aggregate) ในลักษณะการทำงานแบบขนานได้ เช่นในการหาผลรวมหรือค่าเฉลี่ย (sum, average) ในระบบฐานข้อมูลแบบสัมพันธ์จะทำการคำนวณหาผลรวมและทำการนับของแต่ละส่วนที่แยกกันเก็บ (stored fragment) จากนั้นก็จะทำการนำเอาผลที่ได้จากแต่ละส่วน (stored fragment) มาบวกกันและหารด้วยจำนวนที่นับได้ทั้งหมด ซึ่งสำหรับของระบบฐานข้อมูลแบบ ORDBMS จะต้องสามารถที่จะแก้ไขหรือแทนที่การทำงานแบบนี้ได้หรือสามารถที่จะให้ผู้ใช้กำหนดการทำงานแบบนี้ (user-defined aggregates) ได้ โดยจะเป็นตัวทำการกำหนดการทำงานให้

4.1.4 ตัวหาเส้นทาง (Optimizer)

ตัวหาเส้นทางของระบบฐานข้อมูลแบบสัมพันธ์จะมีการทำงานโดยอาศัยการทดสอบสับเซตของเส้นทางทั้งหมดโดยจะหาเส้นทางที่ใช้ทรัพยากรน้อยที่สุด ซึ่งโดยทั่วไปตัวหาเส้นทางของระบบฐานข้อมูลแบบสัมพันธ์นี้จะมีการเลือกเส้นทางอยู่ 2 แบบ

1. cost-based optimizer เป็นแบบที่ใช้สถิติในการตัดสินใจในการเลือกเส้นทาง
2. Rule-based optimizer เป็นแบบที่ไม่ใช้สถิติแต่ใช้กฎตายตัวซึ่งอาจจะขึ้นกับวิธีการเขียนคำสั่ง SQL

ซึ่งแบบที่ 1 นั้นค่า (cost) ต่างๆ นี้จะคิดจากตัวดำเนินการและฟังก์ชันที่กระทำกับชนิดข้อมูลนั้นๆ ส่วนในระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์นั้นการทำงานของตัวหาเส้นทางต้องมีความยืดหยุ่นกว่าคือจะต้องทำการดูชนิดข้อมูลซึ่งเป็นชนิดข้อมูลที่ซับซ้อน ตัวดำเนินการ และวิธีการเลือกข้อมูลจากข้อมูลต่างๆ เหล่านี้จากรายการของระบบ (System catalog) ซึ่งก็คือตัวหาเส้นทางนี้จะต้องสามารถที่จะทำการแปลงหรือดูโค้ด (hard code logic) ไปเป็นลักษณะของรูปแบบตาราง (table-driven) นอกจากนี้ในกรณีที่

มีคิวรี่ดังตัวอย่างข้างต้นที่มีการเรียกฟังก์ชั่น bread ซึ่งเป็นฟังก์ชันที่ต้องใช้ทรัพยากรในการดำเนินการมาก (expensive function) ตัวหาเส้นทางนี้จะพิจารณาเฉพาะจำนวนของหน้า (page) และจำนวนของเรคคอร์ด (record) ที่ต้องใช้ในการทดสอบรวมไปถึงจำนวนคำสั่งที่ใช้ในแต่ละฟังก์ชัน (CPU instruction) และจำนวนของพารามิเตอร์ที่ต้องใช้

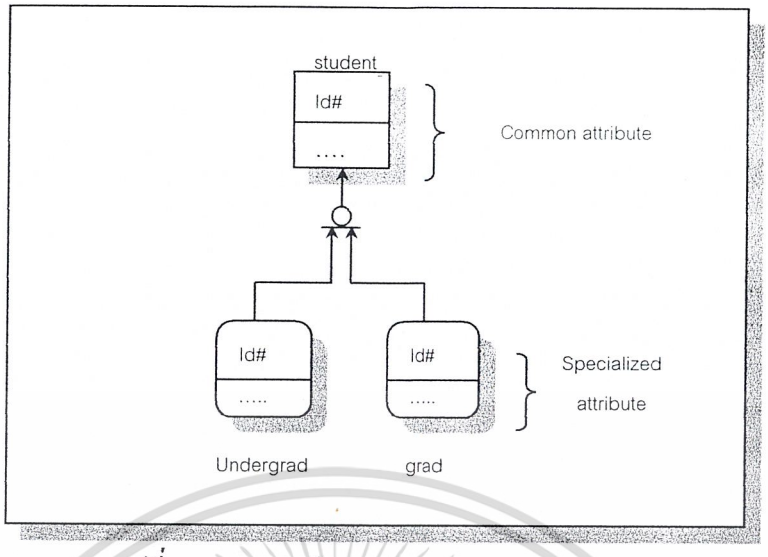
4.1.5 ตัวตรวจสอบไวยากรณ์ (Parser)

เป็นตัวที่คอยตรวจสอบไวยากรณ์ในการทำคิวรี่ในระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์จะมีการตรวจสอบโดยอาศัยตาราง (table-driven parser) คือจะมีการอ่านข้อมูล (metadata) จากรายการของระบบ (System catalog) นอกจากนี้ตัวตรวจสอบจะต้องรู้จักชนิดข้อมูลแบบออบเจ็กต์ที่ซับซ้อน (complex ออบเจ็กต์) เซต และข้อมูลแบบอ้างอิง (References) การจัดการกับการสืบทอด (inheritance)

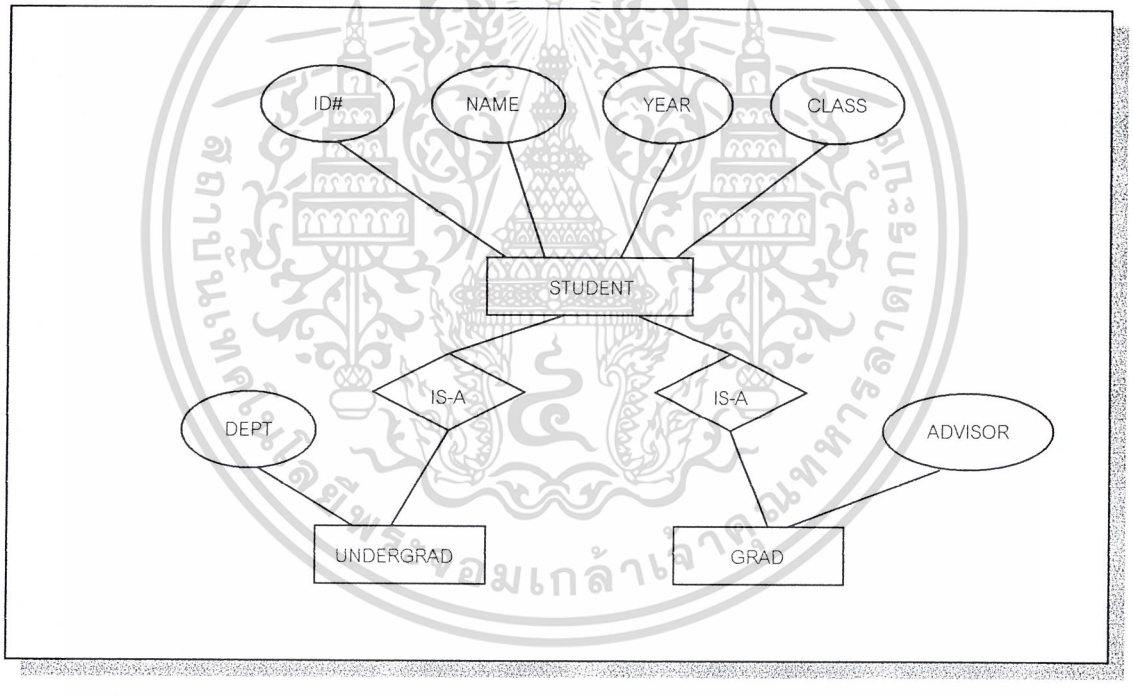
4.2 โมเดลที่ใช้ในการออกแบบ

สำหรับในการออกแบบฐานข้อมูลแบบนี้ก็ยังคงมองเหมือนแบบฐานข้อมูลสัมพันธ์ได้ โดยสำหรับโมเดลที่ใช้ในการออกแบบก็ยังสามารถที่จะใช้ ER หรือ ไนแอม (NIAM) ได้เพราะทั้งสองโมเดลนี้มีหลักการของการทำลำดับขั้นของชนิดข้อมูล (Subtype hierarchy) อยู่แล้วแต่ในการแปลงจากโมเดลเป็นตารางนั้นยังไม่มีหลักการที่แน่นอนอันเนื่องมาจากระบบฐานข้อมูลเดิมเป็นแบบสัมพันธ์ธรรมดาที่ยังไม่สนับสนุนหลักการดังกล่าว ทำให้ในการแปลงมาเป็นตารางของแต่ก่อนนั้นขึ้นอยู่กับงานซึ่งอาจจะมีการรวมเอาทั้งหมดอยู่ในตารางเดียว (ซุปเปอร์คลาสและซับคลาส) หรืออาจจะแบ่งให้มีตารางที่ประกอบด้วยแอตทริบิวต์ร่วมอยู่ตารางหนึ่งแล้วแยกตารางของแต่ละซับคลาสเป็นอีกตารางโดยจะประกอบด้วยแอตทริบิวต์ของแต่ละซับคลาสที่มีต่างหากออกมาก็ได้

โมเดล EER (Enhanced-Entity Relationship) เป็นโมเดลที่ได้เพิ่มแนวคิดเกี่ยวกับซับคลาส (Sub-class) และซุปเปอร์คลาส (Super-class) กับโมเดล ER ที่มีการกำหนดวิธีการในการแปลงจากโมเดลที่ชัดเจนกล่าวคือปกติในการแปลง ER จะมีขั้นตอนอยู่ 7 ขั้นตอน แต่สำหรับโมเดล EER จะเพิ่มขั้นตอนที่ 8 ซึ่งเป็นขั้นตอนที่จะนำแอตทริบิวต์ทั้งหมดจากซุปเปอร์คลาสมารวมกับแอตทริบิวต์ของซับคลาสดังตัวอย่างต่อไปนี้



รูปที่ 4-1 (ก) แสดงโมเดล ER (CASE TOOL)



รูปที่ 4-1 (ข) แสดงการแปลงจาก ER (CASE TOOL) มาเป็น EER

STUDENT

ID#	NAME	YEAR	CLASS

UNDERGRAD

ID#	NAME	YEAR	CLASS	DEPT

GRAD

ID#	NAME	YEAR	CLASS	ADVISOR

รูปที่ 4-1(ค) แสดงตัวอย่างการแปลง EER โมเดลเป็นรีเลชัน

บทที่ 5
ระบบฐานข้อมูลของอินฟอร์มิกซ์
INFORMIX UNIVERSAL SERVER (IUS)

IUS เป็นระบบฐานข้อมูลแบบ ORDBMS ที่มีการรวมความสามารถของ OODBMS และ RDBMS โดยภาษาที่ใช้กับระบบฐานข้อมูลนี้ได้มีการนำภาษา SQL3 มาใช้ดังนั้นการทำงานของ IUS จะต้องอ้างกับมาตรฐานของ SQL3 ที่ทาง ANSI และ ISOได้กำหนดไว้เช่นชนิดของข้อมูล,การทำคิวรี่เป็นต้น

5.1 ประเภทข้อมูลใน IUS

ในระบบฐานข้อมูลของ IUS ได้มีการแบ่งออกเป็น 3 ประเภทหลักๆคือ

- ชนิดข้อมูลพื้นฐาน (Built-in data type) เป็นชนิดข้อมูลที่มีมากับ IUS
- ชนิดข้อมูลที่สร้างขึ้นใหม่ (User-defined types) เป็นชนิดข้อมูลที่ระบบฐานข้อมูลอนุญาตให้ ผู้ใช้สามารถที่จะสร้างชนิดข้อมูลขึ้นมาใช้เอง ซึ่งแบ่งเป็น 2 ชนิดคือ
  - Opaque type
  - Distinct type
- ข้อมูลแบบ Complex เป็นชนิดของข้อมูลที่มีการกำหนดไว้ในมาตรฐาน SQL3 ซึ่งประกอบด้วย
  - Collection type (set, list, multiset)
  - Row types
  - Reference

5.1.1 ข้อมูลชนิดพื้นฐาน เป็นชนิดข้อมูลที่ทาง IUS มีให้มาซึ่งสามารถเทียบกับชนิดข้อมูลที่กำหนดในมาตรฐานของ SQL3 ได้ดังนี้

Data types		IUS
Character		CHAR,CHARACTER VARYING (VARCHAR), LVARCHAR
Numeric	Exact numeric	DECIMAL,MONEY,SMALLINT,INTEGER, INT8,SERIAL,SERIAL8
	Approximate numeric	SMALLFLOAT ,FLOAT

Data types		IUS
Large object	Simple large object	TEXT,BYTE
	Smart large object	CLOB,BLOB
Time		DATE,DATETIME,INTERVAL
Miscellaneous		BOOLEAN

ตาราง 5-1 แสดง built-in data type ของ IUS

Data Types		SQL3
Character string		CHARACTER, CHARACTER VARYING
Numeric	Exact numeric	NUMERIC,DECIMAL,INTEGER,SMALLINT
	Approximate numeric	FLOAT,REAL,DOUBLE PRECISION
Bit string		BIT,BIT VARYING
Large object string		BINARY LARGE OBJECT, CHARACTER LARGE OBJECT
Date time and Interval		DATE,TIME,TIMESTAMP,INTERVAL
Boolean		BOOLEAN

ตาราง 5-2 แสดงชนิดข้อมูลของ SQL3

5.1.2 ชนิดข้อมูลที่สร้างใหม่ (User defined type) เป็นชนิดข้อมูลที่ผู้ใช้สร้างขึ้นมาใหม่นอกเหนือจากที่ทาง IUS มีให้เพื่อความยืดหยุ่นในการเก็บข้อมูลและการจัดการ โดยทาง IUS ได้แบ่งข้อมูลชนิดนี้ออกเป็น 2 ชนิดคือ

- **Opaque type** เป็นชนิดข้อมูลที่มีลักษณะซ่อนเร้นข้อมูลภายในทั้งหมด (fully encapsulated) ซึ่งเป็นข้อมูลที่ไม่สามารถจะแบ่งแยกย่อยได้ (atomic data type) ที่เราสามารถที่จะกำหนดหรือสร้างไว้ในระบบฐานข้อมูลโดยที่ระบบฐานข้อมูลจะไม่รู้ถึงลักษณะของโครงสร้างภายในของข้อมูลชนิดนี้ว่าเป็นแบบใดเหมือนกับข้อมูลที่มากับระบบฐานข้อมูล (built-in data type) ดังนั้นจึงจำเป็นที่จะต้องมีฟังก์ชันต่างๆ ที่คอยจัดการและดำเนินการกับ opaque type

- **Distinct type** เป็นชนิดของข้อมูลที่โครงสร้างภายในเหมือนกับชนิดข้อมูลที่มีอยู่แล้วโดยที่จะมีชื่อและฟังก์ชันที่ถูกสร้างมาแตกต่างไปจากข้อมูลที่เป็นต้นแบบ (Source type) ตัวอย่างเช่นเราสร้างข้อมูลชนิด decnum ที่มีลักษณะของโครงสร้างภายในเหมือนกับข้อมูลชนิด real ดังนั้นฟังก์ชันทั้งหมดที่ใช้กับ real จะสามารถที่จะทำงานกับ decnum ได้ แต่อย่างไรก็ตามเราไม่สามารถที่จะทำการบวก ลบ หรือเปรียบเทียบข้อมูลระหว่าง decnum กับ real ได้ จำเป็นจะต้องทำผ่านฟังก์ชันที่มีหน้าที่แปลงชนิดข้อมูล(Casting

function เป็นฟังก์ชันที่ทำการ cast (เป็น mechanism ที่ทำการแปลงค่าจากข้อมูลชนิดหนึ่งไปเป็นข้อมูลอีกชนิดหนึ่ง) ซึ่งเราสามารถที่จะสร้างขึ้นมาเองได้นอกเหนือจากที่ระบบฐานข้อมูลมีให้) เพื่อทำการแปลงชนิดของข้อมูลก่อน

IUS	SQL3
Opaque data type	Abstract data type
Distinct data type	Distinct data type

ตาราง 5-3 แสดงการ Implement ชนิดข้อมูลของ IUS เทียบกับ SQL3

- ชนิดข้อมูลแบบ Opaque

ชนิดข้อมูลแบบ Opaque จะต้องประกอบด้วย

- โครงสร้างข้อมูล (Data structure)ในการเก็บข้อมูลโครงสร้างภายในซึ่งจะเขียนโดยภาษา C สามารถที่จะแบ่งออกเป็น 2 ประเภท
  - Fixed-length opaque type ใช้กับข้อมูลที่มีลักษณะขนาดคงที่เช่น numeric เป็นต้น
  - Verying-length opaque type ใช้กับข้อมูลแบบ multi-representation เช่นรูปภาพโดยได้กำหนดค่าขนาด (default maximum size) ของข้อมูลเป็น 2 Kbyte แต่เราสามารถที่จะกำหนดให้มีขนาดสูงสุดได้ถึง 32 Kbyte ซึ่งถ้าขนาดเกินกว่านี้ระบบฐานข้อมูลจะส่งค่าที่แสดงถึงข้อผิดพลาดมาให้
- ฟังก์ชันสนับสนุน (Support function) ต่างๆ ใช้สำหรับให้ระบบฐานข้อมูลทำการติดต่อกับข้อมูลแบบ opaque ซึ่งฟังก์ชันสนับสนุนนี้จะเขียนโดยภาษา C เรียกว่า external function โดยจะต้องทำการ compile function และทำการก็อปปี้ไปไว้ที่ share library บนระบบจัดการฐานข้อมูล จากนั้นจะต้องทำการรีจิสเตอร์ฟังก์ชันสนับสนุนเพื่อให้ระบบฐานข้อมูลรู้จัก
- ฟังก์ชันที่ใช้เรียกในภาษา SQL (Additional SQL -invoke routine) เป็นรูทีนที่เรียกใช้โดยใช้ในคำสั่ง SQL ซึ่งแบ่งตามประเภทได้ดังนี้
  - ฟังก์ชันภายใน (Built-in function) เช่น SUM,AVG,ABS
  - ฟังก์ชันที่ให้ผลลัพธ์เป็นค่าเดียว (Aggregate function) เช่น COUNT,MIN,MAX
  - ฟังก์ชันที่เป็นตัวดำเนินการ (Operator function) ซึ่งประกอบด้วย
    - ตัวดำเนินการทางคณิตศาสตร์ (Arthmetic operator) เช่น +,-,*,/
    - ตัวดำเนินการเกี่ยวกับข้อความ (Text operator) เช่น LIKE,MATCHE,||
    - ตัวดำเนินการใช้ในการเปรียบเทียบ (Comparing data)
      - ตัวดำเนินการแสดงความสัมพันธ์ (Relational operator) เช่น =,<,>,<>
      - ใช้เป็นเงื่อนไข (Condition) เช่น IS,IS NOT,BETWEEN

(หมายเหตุ : ซึ่งทั้งหมดนี้จะต้องเขียนโดยใช้ภาษา C ไม่สามารถที่จะใช้ภาษา SPL ได้)

ตัวอย่าง ในการสร้างชนิดข้อมูลแบบ Opaque ที่เรียกว่า circle ซึ่งเป็นแบบมีขนาดคงที่

```
typedef struct {
        double   x;
        double   y;
} point_t;

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

เป็นการกำหนดโครงสร้างภายในของ circle_t ซึ่งเป็นชนิดข้อมูลแบบ opaque ที่มีข้อมูลชนิด double มีขนาด 8 byte ดังนั้น ขนาดทั้งหมดของโครงสร้างภายในของ circle_t มีขนาด 24 byte ตัวโค้ดนี้เขียนโดยภาษา C ซึ่งจะถูกคอมไพล์ (compile) เป็นไฟล์ circle.so (สำหรับการคอมไพล์บนยูนิกซ์แต่ถ้าเป็นบน windows NT จะเป็นไฟล์ circle.bld) และจะถูกนำไปใส่ที่ shared library

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

ตารางที่ 5-4 *แสดงฟังก์ชันสนับสนุนต่างๆ ที่ต้องการรีจีสเตอร์ ของข้อมูลแบบ circle*

โดยที่แต่ละฟังก์ชันทำหน้าที่ดังนี้

- circle_input จะทำการ convert mi_lvarchar ที่เป็น external representation เป็น internal representation เป็น circle_t

- circle_output จะทำการ covert internal representation ของ circle_t ไปเป็น external representation เป็น mi_lvarchar
- circle_receive จะทำการ covert internal representation ของ client ไปเป็น internal representation ของ server
- circle_send จะทำการ covert internal representation ของ server ไปเป็น internal representation ของ client

เมื่อเราทำการกำหนดอินพุทและเอาท์พุทฟังก์ชันเสร็จแล้วลักษณะรูปแบบภายนอก (External format) ของ circle_t จะเป็นดังนี้

(	X	,	Y	,	Z	)

รูปที่ 5-1 แสดงรูปแบบภายนอกของ circle_t โดยที่ (x,y) เป็นจุดศูนย์กลางและ z เป็นรัศมี

การรีจีสเตอร์ circle สามารถที่จะใช้คำสั่ง SQL ทำการรีจิสเตอร์ชนิดข้อมูลและ input,output,send และ receive ได้ดังนี้

```
CREATE OPAQUE TYPE circle (INTERNALLENGTH=24);
-- เนื่องจาก x,y,z เป็น double ที่มีขนาด 8 byte

CREATE FUNCTION circle_int(txt LVARCHAR) RETURN circle
        EXTERNAL NAME '/usr/lib/circle.bld(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.bld(circle_output)'
        LANGUAGE C NOT VARIANT;
CREATE IMPLICIT CAST(circle AS LVARCHAR WITH circle_out);

CREATE FUNCTION circle_rev(cl_cir SENDREV) RETURNS circle
        EXTERNAL NAME '/usr/lib/circle.bld(circle_receive)'
        LANGUAGE C NOT VARIANT;
CREATE IMPLICIT CAST(SENDRECV AS circle WITH circle_rev);
```

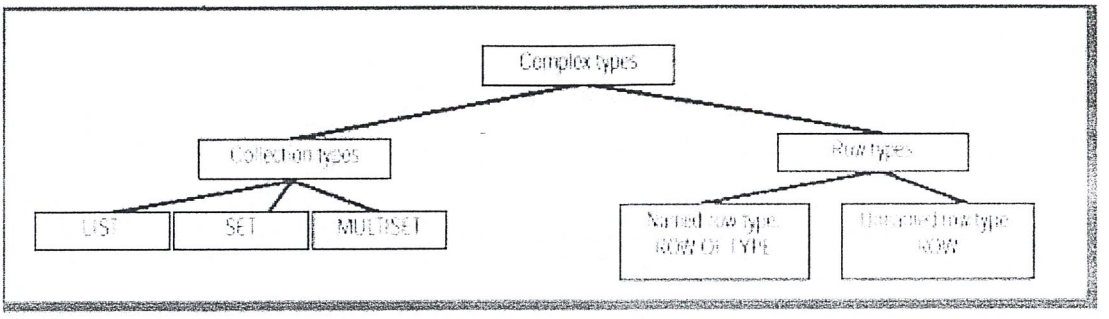
```
CREATE FUNCTION circle_snd(cir circle) RETURNS SENDRECV
        EXTERNAL NAME '/usr/lib/circle.bld(circle_send)'
        LANGUAGE C NOT VARIANT;
CREATE IMPLICIT CAST(circle AS LVARCHAR WITH circle_snd);
```

- ชนิดข้อมูลแบบ Distinct

ชนิดข้อมูลแบบ Distinct เป็นชนิดข้อมูลที่ผู้ใช้กำหนดหรือสร้างขึ้นที่มีการใช้โครงสร้างภายในของชนิดข้อมูลที่มีอยู่แล้ว ซึ่งอาจจะเป็นข้อมูลที่มากับระบบฐานข้อมูลเอง opaque distinct หรือ row type ก็ได้ โดยเมื่อเราทำการสร้างหรือประกาศข้อมูลแบบ distinct มาจะมีคุณสมบัติดังนี้

- ระบบฐานข้อมูลจะทำการสร้างการฟังก์ชันที่ทำการแปลงชนิดข้อมูลระหว่าง distinct กับข้อมูลที่เป็นต้นแบบ (parent type) ให้อัตโนมัติ ซึ่งเราสามารถที่จะดูได้ในรายการของระบบที่ชื่อ syscasts ได้
- ข้อมูลแบบ distinct จะได้รับการสืบทอดรูทีน (user-defined routine (UDRs)) และฟังก์ชันที่ทำการแปลงชนิดข้อมูลของข้อมูลที่เป็นต้นแบบ (แต่สำหรับข้อมูล distinct ของข้อมูลที่มากับระบบฐานข้อมูล (built-in type) จะไม่ทำการสืบทอด system-defined cast)
- เราสามารถที่จะรีจิสเตอร์ UDR และฟังก์ชันแปลงชนิดข้อมูลสำหรับข้อมูล distinct เพิ่มได้โดย
- ซ้ำกันที่มีอยู่แล้วที่ได้มาจากการสืบทอดมาจากข้อมูลที่เป็นต้นแบบ
- เพิ่มขึ้นมาใหม่ซึ่งตัวชนิดข้อมูลต้นแบบยังไม่มี
- ข้อมูลแบบ distinct จะต้องมีโครงสร้างภายในเหมือนกับข้อมูลที่เป็นแม่ (parent type) แต่ส่วนของโครงสร้างหรือรูปแบบภายนอก (external representation) อาจจะแตกต่างได้ ซึ่งถ้าเราต้องการที่จะสร้างข้อมูลแบบ distinct ที่มีรูปแบบภายนอกแตกต่างไป เราต้องทำการสร้าง UDR ที่ทำการแปลงระหว่างรูปแบบภายนอกกับรูปแบบโครงสร้างภายในและสร้าง ฟังก์ชันที่ใช้แปลงชนิดข้อมูลที่ใช้ระหว่างข้อมูลแบบ distinct กับ lvarchar (lvarchar เป็นชนิดของข้อมูลใน IUS ซึ่งถูกใช้เป็นรูปแบบภายนอก (external represention) ของ distinct และ opaque type) ขึ้นมาเอง

5.1.3 ข้อมูลแบบ Complex : เป็นข้อมูลที่ผู้ใช้สร้างที่สามารถจะประกอบไปด้วยข้อมูลชนิดต่างๆ ซึ่งคุณสมบัติที่สำคัญของชนิดข้อมูลนี้ก็คือการที่ง่ายที่จะเข้าถึงแต่ละส่วน (component) ของข้อมูลที่ไม่เหมือนกับข้อมูลชนิด built-in หรือแบบ opaque ที่มีลักษณะซ่อนเร้นกล่าวคือถ้าเราจะเข้าถึงข้อมูลของ opaque ก็จะต้องทำผ่านฟังก์ชันที่กำหนดไว้กับ opaque



รูปที่ 5-2 เป็นรูปแสดงถึง Complex types ที่มีอยู่ใน IUS

จากรูปที่ 5-2 จะเห็นว่าข้อมูลแบบ Complex ประกอบด้วย

- ข้อมูลแบบ Collection: เป็นกลุ่มของข้อมูลที่มีชนิดข้อมูลเดียวกัน โดยอนุญาตให้เราสามารถที่จะเก็บและจัดการกับ collection ของข้อมูลภายใน row เดียว ข้อมูลแบบ collection ประกอบ 2 ส่วนคือ type constructor และ element type โดยที่

- Type constructor ประกอบด้วย set, multiset และ list
  - SET :เป็นข้อมูลแบบ collection ที่ elements ไม่มีลำดับและไม่สามารถที่จะมีข้อมูลซ้ำกันได้ภายใน set ซึ่งในการกำหนด set จะต้องใช้ set constructor
  - LIST :เป็นข้อมูลแบบ collection ที่ elements มีลำดับและ element สามารถที่จะซ้ำกันได้ ในการกำหนด list จะต้องใช้ list constructor
  - MULTISET :เป็นข้อมูลแบบ collection ที่ไม่มีลำดับของ element และอนุญาตให้ข้อมูลซ้ำกันได้ ในการกำหนด multiset จะต้องใช้ multiset constructor
- Element type เป็นการระบุชนิดของข้อมูลของ element ซึ่ง element สามารถที่จะมีชนิดใดก็ได้นอกจาก SERIAL และ SERIAL8

- Row types เป็นชนิดข้อมูลที่มีลักษณะคล้ายกับข้อมูลแบบโครงสร้าง (structure) ในภาษา C โดยภายในจะประกอบด้วยฟิลด์ (fields) แต่ละฟิลด์จะมีชื่อและชนิดข้อมูลเป็นของตัวเองซึ่งฟิลด์ของ row type ก็เปรียบได้กันคอลัมน์ของตารางนั้นเองซึ่งจะเห็นว่า row type ของ IUS ก็เป็นชนิดของข้อมูลชนิดเดียวกัน row type ที่มีอยู่มาตรฐานของ SQL3 โดยที่ row type ใน IUS นั้นมีอยู่ด้วย 2 แบบคือ named row type และ unnamed row type

- Named row type :เป็นกลุ่มของฟิลด์ ที่การกำหนดภายใต้ชื่อชื่อหนึ่ง โดยฟิลด์นี้หมายถึงคอมโพเนนต์ของ row type โดยในการสร้าง named row type จะต้องมีชื่อไม่ซ้ำกับ named row type ที่มีในฐานข้อมูลอยู่ก่อนแล้ว เราจะมีการสร้าง named row type ก็ต่อเมื่อเราต้องการที่ทำงานที่ต้องการที่เข้าถึงคอมโพเนนท์ของข้อมูลได้เช่นเราอาจจะสร้าง named row type ที่เก็บ address ซึ่งสามารถที่จะเข้าถึงข้อมูลแต่ละคอมโพเนนท์ ของ address ได้เช่น ถนน เมือง จังหวัด เป็นต้น

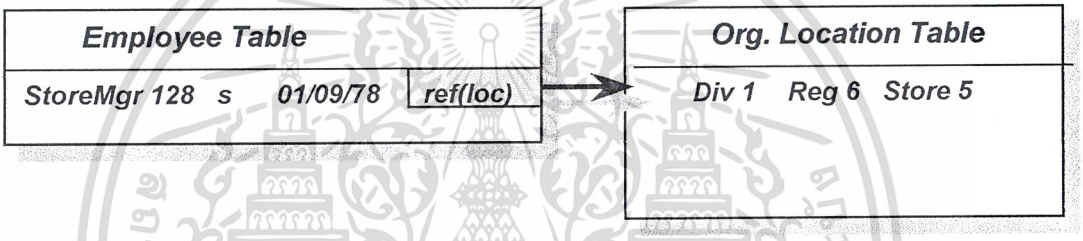
- Unnamed row type :เป็นกลุ่มของฟิลด์ที่กำหนดโดยใช้โครงสร้างของมันเองซึ่งไม่เหมือนกับ named row type เราสามารถที่จะกำหนด unnamed row type ให้กับตารางและใช้ในการกำหนดชนิดให้กับคอลัมน์ได้

- Reference: เป็นชนิดข้อมูลที่มีลักษณะคล้ายกับตัวชี้ (pointer) ในภาษา C และ C++ ในการอ้างถึงชนิดของข้อมูลที่กำหนดเมื่อตอนทำการสร้างข้อมูลแบบ Reference โดยข้อมูลแบบ Reference นี้จะมีลักษณะคล้ายกับ Reference ที่มีการกำหนด SQL3 มาตรฐาน ซึ่งการใช้ข้อมูลแบบ Reference จะใช้เมื่อ

1. มีการเปลี่ยนแปลงอินเด็กซ์บ่อย

2.มีการใช้คีย์ที่มีขนาดใหญ่และเป็นคีย์ที่มีความซับซ้อน

3. Complex row มีการชี้ไปที่หลายๆ ออบเจ็กต์โดยแต่ละอันก็ต้องการ key ที่มีขนาดใหญ่และเป็นคีย์ที่มีความซับซ้อน



รูปที่ 5-3 แสดงการใช้ชนิดข้อมูลแบบ Reference

ตัวอย่าง คำสั่ง SQL ในการใช้ข้อมูลแบบ reference

```
CREATE ROW TYPE employee_t
(
        NAME VARCHAR(50)
        Manager  REF (employee_t )
);
CREATE TABLE employee OF TYPE employee_t  WITH
LOGICAL ROWIDS;    ( REFerences require Logical RowIDs )

INSERT INTO employee (name)  VALUES ('Emily') ;
                ( Emily has no manager )

INSERT INTO employee VALUES('Sam'), SELECT &e
FROM employee e  WHERE e.name = 'Emily' ));
```

5.2 คุณสมบัติการสืบทอด (Inheritance)

IUS สนับสนุนการสืบทอดสำหรับข้อมูลชนิด named row type และ typed table ซึ่งการสืบทอดนี้เป็นกระบวนการที่อนุญาตให้ข้อมูลและตารางสามารถที่สืบทอดคุณสมบัติต่างๆ จากข้อมูลและตารางอื่น

5.2.1 Row type กับการสืบทอด

Named row type สามารถที่จะทำการสืบทอดสิ่งต่างๆ ดังนี้

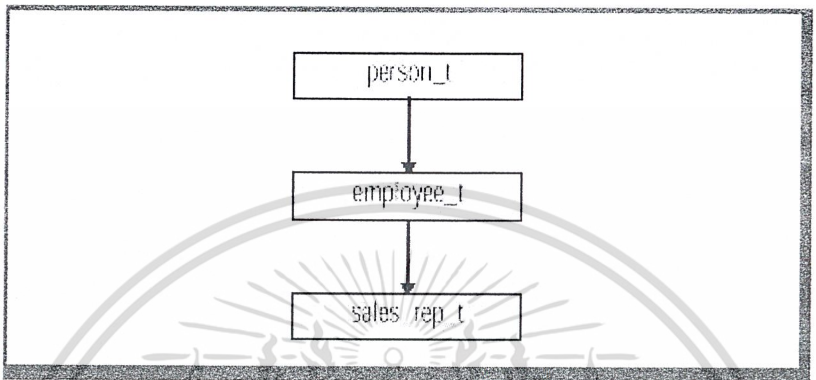
- โครงสร้างภายใน (representation) ได้แก่ฟิลด์ข้อมูลต่างๆ
- พฤติกรรม (behavior) ซึ่งได้แก่ routine,aggregate และ operators ต่างๆ)

ได้โดยจะเรียก row type ที่ทำการสืบทอดสมบัติต่างๆ ให้กับ row type อื่นๆ ว่า supertype และเรียก row type ที่ได้รับการสืบทอดว่า subtype โดยในการสืบทอดกันเป็นทอดๆ นั้นเราเรียกว่าลำดับชั้นของชนิดข้อมูล (type hierarchy) ซึ่ง IUS สนับสนุนการสืบทอดเฉพาะแบบ single inheritance กล่าวคือ 1 subtype จะมี 1 supertype ได้เท่านั้น

ตัวอย่าง คำสั่ง SQL ในการสืบทอด row type

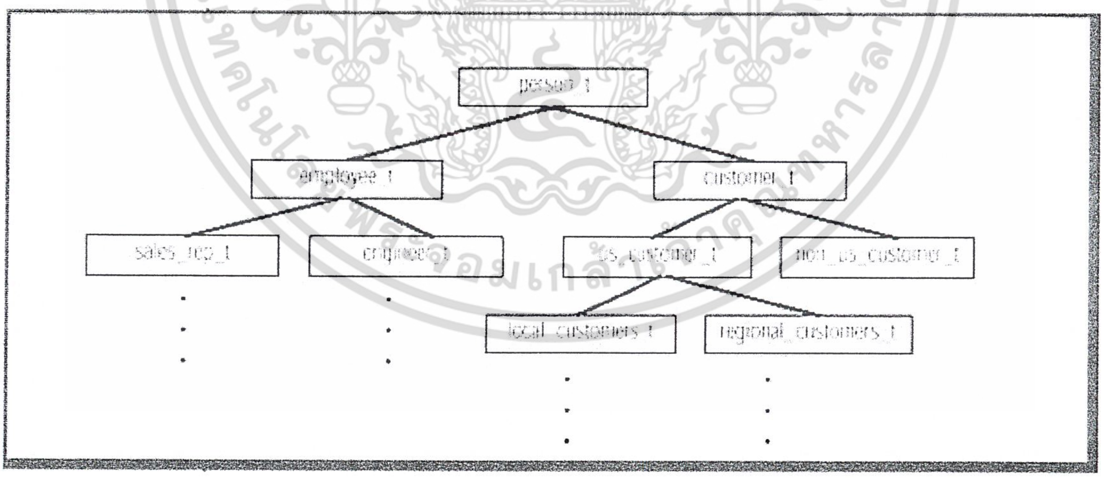
```
CREATE ROW TYPE person_t
(
        Name        VARCHAR(30) NOT NULL,
        Address     VARCHAR(20),
        City        VARCHAR(20),
        State       CHAR(2),
        Zip         INTEGER,
        Bdate       DATE
);

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;
```



รูปที่ 5-4 แสดงลำดับขั้นของชนิดข้อมูลที่ได้จาก SQL ในตัวอย่าง

จากรูปได้แสดงถึงลำดับขั้นของชนิดข้อมูลซึ่งจะเห็นว่าชนิดข้อมูล person_t เป็นซูเปอร์ไทป์ (supertype) ของ employee_t และ employee_t เป็นซับไทป์ (supertype) ของ sales_rep_t อีกที



รูปที่ 5-5 แสดงลำดับขั้นของชนิดข้อมูลในลักษณะแบบโครงสร้างต้นไม้ (Tree structure)

5.2.2 ข้อมูล Distinct กับการสืบทอด

ข้อมูลแบบ Distinct สามารถที่จะทำการสืบทอดสิ่งต่างๆจากชนิดข้อมูลที่เป็นต้นแบบหรือชนิดข้อมูลแม่ (parent type) ดังนี้

- ลักษณะโครงสร้างภายในและภายนอก (internal และ external representation)

- รูทีนที่ผู้ใช้สร้างขึ้น (users-defined routine (UDRs)) และฟังก์ชันที่ใช้ในการแปลงชนิดข้อมูล (casting function) ต่างๆ

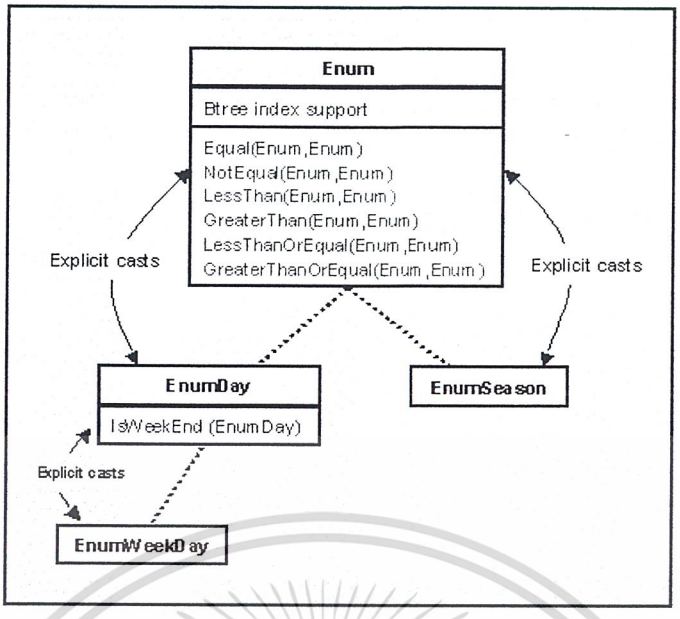
ซึ่งจะเห็นว่าข้อมูลแบบ Distinct จะทำการสืบทอดเมธทอดต่างๆจากชนิดข้อมูลแม่ซึ่งหมายความว่าข้อมูลแบบ distinct สามารถที่จะเรียกใช้เมธทอดของชนิดข้อมูลแม่ได้ โดยระบบฐานข้อมูล จะทำการ resolves (resolve เป็นกระบวนการที่ระบบฐานข้อมูบเลือกรูทีนที่จะใช้งานซึ่งจะกล่าวในรายละเอียดในหัวข้อ เรื่อง routine-resolution) UDRs หรือเมธทอดที่สืบทอดจากชนิดข้อมูลแม่ ณ.เวลาที่ run-time และนอกจากนี้ยังสามารถที่จะกำหนดเมธทอดของตัวเองได้อีกด้วย เช่น สมมุติว่ามีชนิดข้อมูลแบบ Enum ซึ่งเป็นข้อมูลแบบ opaque โดยเราจะกำหนดให้เป็นข้อมูลต้นแบบหรือชนิดข้อมูลแม่ของข้อมูลแบบ distinct ที่เราจะสร้างเป็นแบบ enumeration คือ EnumDay,EnumWeekDay และ EnumSeason โดยลักษณะโครงสร้างภายใน (internal representation) จะเหมือนกับของ Enum ซึ่งเป็น integer โดยที่รูปแบบภายนอก (external representation) แต่ละตัวจะเป็นข้อความที่อักษรตัวเล็กกับตัวใหญ่ถือเป็นคนละตัว (case-insensitive text string) ที่ระบุไว้ของแต่ละชนิดข้อมูล Enumeration ดังนี้

Domain

- *Enum* :one,two,three,four,five,six,seven,eight,nine,ten
- *EnumDay* :sunday,monday,tuesday,wednesday,thursday,friday,saturday
- *EnumWeekDay* :monday,tuesday,wednesday,thursday,friday
- *EnumSeason* :winter,spring,summer,fall

จากรูปที่ 5-6 ในหน้าถัดไปได้แสดงถึงลำดับของชนิดข้อมูล โดยที่ Enum ที่อยู่บนสุดของ ลำดับชั้นเป็นข้อมูลแบบ opaque ซึ่งมีฟังก์ชันอินพุท เอาท์พุท อินเด็กซ์แบบ B-tree และ UDR ที่ใช้สำหรับค้นหาข้อมูล ส่วน EnumDay และ EnumSeason เป็นข้อมูลแบบ distinct ของ Enum โดยมี EnumWeekDay ที่เป็นข้อมูลแบบ distinct ของ EnumDay อีกทีหนึ่ง

ทั้งสามข้อมูลแบบ Distinct นี้จะได้รับการสืบทอด UDR และอินเด็กซ์แบบ B-tree ทั้งหมดของ Enum โดยที่ EnumDay ได้มีการเพิ่ม UDR ชื่อ IsWeekend() ที่จะส่งค่ากลับเป็นค่า TRUE หรือ FALSE ว่าเป็นวันหยุดเสาร์หรืออาทิตย์หรือไม่ ซึ่งจะทำให้ EnumWeekDay ได้รับการถ่ายทอดรูทีนนี้ไปด้วย



รูปที่ 5-6 แสดงลำดับขั้นของชนิดข้อมูลของข้อมูลแบบ Enumeration ทั้ง 3 ชนิด

ในการใช้งานข้อมูลแบบ Distinct นี้ก็สามารถที่จะใช้เหมือนกับชนิดข้อมูลทั่วไป เช่น เราสามารถที่จะสร้างตารางโดยใช้ EnumDay ดังนี้

```
CREATE TABLE schedule
(
        Name            VARCHAR(50),
        DayOfWeek       EnumDay
        StartHour       INTEGER
        EndHour         INTEGER
);
```

5.2.3 ตารางกับการสืบทอด

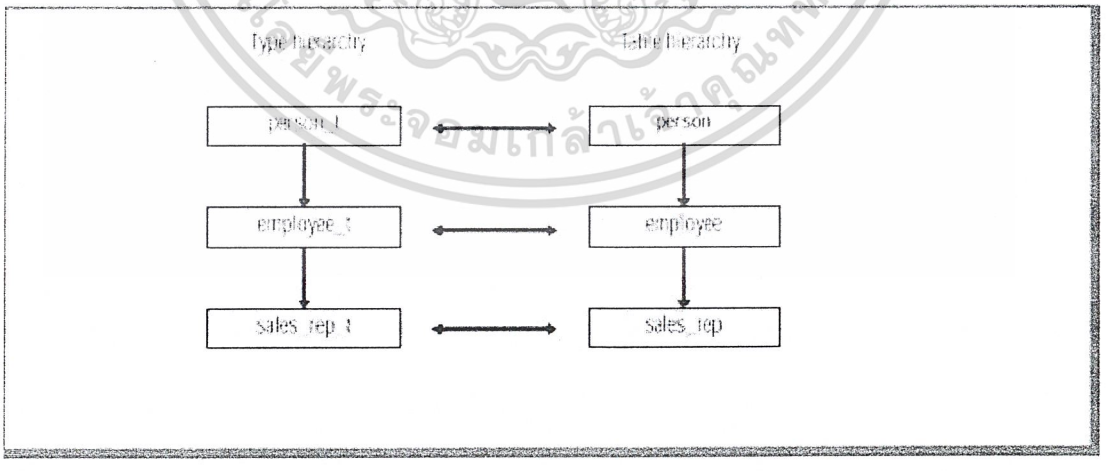
การสืบทอดคุณสมบัติต่างๆ ของตารางสามารถจะทำได้เฉพาะกับ Typed table เท่านั้น (คือ ตารางที่มีการกำหนด named row type ให้กับตาราง) ซึ่งการสืบทอดกับเป็นทอดๆ ของตารางเราเรียกว่าลำดับขั้นของตาราง (table hirarchy) โดยซับตาราง (subtable) จะได้รับการถ่ายทอดคุณสมบัติต่างๆ ต่างไปนี้จากซุปเปอร์ (supertable)

- คอลัมน์ ทั้งหมดที่อยู่ใน supertable
- กฎข้อบังคับ (Constraint) ต่างๆ
- การกำหนดการจัดเก็บข้อมูล(Storage options)
- อินเด็กซ์ (Indexes)

- Referential integrity
- ทริกเกอร์ (Triggers)
- วิธีการเข้าถึงข้อมูล (Access method)

ตัวอย่าง คำสั่ง SQL ในการทำการถ่ายทอดคุณสมบัติ

```
CREATE TABLE person OF TYPE person_t
(PRIMARY KEY (name))
FRAGMENT BY EXPRESSION
Name < 'n' IN dbspace1,
Name >= 'n'  IN dbspace2;

CREATE TABLE employee OF TYPE employee_t
(CHECK(salary > 34000))
UNDER person;

CREATE TABLE sales_rep OF TYPE sales_rep_t
LOCK MODE ROW
UNDER employee;
```



รูปที่ 5-7 *แสดงความสัมพันธ์ระหว่างลำดับขั้นของชนิดข้อมูลและลำดับขั้นของตาราง*

ด้วย Named row type ที่มีการอ้างถึงในการสร้างตารางในตัวอย่างจะใช้ named row type จากตัวอย่างที่แล้ว จากรูปที่ 5-7 เราจะเห็นความสัมพันธ์ระหว่างลำดับขั้นของชนิดข้อมูลกับลำดับขั้นของตาราง

ซึ่งจากลำดับขั้นของตาราง ตาราง employee และ sales_rep จะได้รับการสืบทอด primary key จาก person คือ name และการทำ fragmentation จากตาราง person และตาราง sales_rep จะได้รับการสืบทอดการตรวจสอบกฎข้อบังคับ (check constraint) จากตาราง employee และได้มีการเพิ่ม lock mode อีก ในลำดับขั้นของตาราง ตารางที่เป็นซับตารางสามารถที่จะกำหนดพฤติกรรม (behavior) ขึ้นมาเองโดยไม่ใช้พฤติกรรมที่ได้รับการสืบทอดมาได้

การแก้ไขพฤติกรรมของตารางในลำดับขั้นของตาราง (Table hierarchy) :เมื่อเราสร้างลำดับขั้นของตาราง เราไม่สามารถที่จะทำการแก้ไขคอลัมน์บนตารางที่มีอยู่ในลำดับขั้นของตารางได้แต่เราสามารถที่จะแก้ไขพฤติกรรมของตารางได้ ดังที่แสดงไว้ในตารางที่ 5-5 ดังนี้

Table behavior	Syntax	Comment
Constraint definition	ALTER TABLE	เราสามารถที่จะแก้ไขหรือยกเลิก constraint เฉพาะตารางที่ constraint นั้นถูก define คือเราไม่สามารถที่จะยกเลิกที่ระดับ subtable ได้
Indexes	CREATE INDEX ALTER INDEX	Index ของ subtable ที่ inherit มาไม่สามารถที่จะยกเลิกหรือแก้ไขได้แต่สามารถที่จะเพิ่ม index ที่ subtable ได้
Triggers	CREATE/DROP TRIGGER	ไม่สามารถ drop trigger ที่ inherit มาได้แต่สามารถที่จะกำหนด trigger ซ้ำกับที่ inherit มาได้

ตารางที่ 5-5 *แสดงถึง Table behavior ที่สามารถทำการแก้ไขได้*

ตัวอย่าง ในการแก้ไข Behavior ของ table hierarchy

```
CREATE TABLE person OF TYPE person_t
(PRIMARY KEY (name))
FRAGMENT BY EXPRESSION
name < 'n' IN dbspace1,
name >= 'n' IN dbspace2;

CREATE TABLE employee OF TYPE employee_t
(CHECK(salary > 34000))
FRAGMENT BY EXPRESSION
name < 'n' IN employ1,
name >= 'n'  IN employ2;
UNDER person;
```

```
CREATE TABLE sales_rep OF TYPE sales_rep_t
LOCK MODE ROW
UNDER employee;
```

การทำคิวรี่ในลำดับชั้นของตาราง สามารถที่จะทำการเลือก แก้ไขหรือทำการลบข้อมูล (select, update, delete) โดยจะมีผลทั้ง supertable และ subtable ด้วย (ส่วนคำสั่งแทรกข้อมูล (insert) จะกระทำกับเฉพาะตารางที่ระบุเท่านั้นจะไม่ทำกับ subtable ด้วย) เช่นเราทำการป้อนคำสั่ง select name บนตาราง person เราจะได้ result ที่อยู่ทั้งในตาราง person,employee และ sales_rep ด้วย

ตัวอย่าง คำสั่ง SQL ในการ select name จากตาราง person

```
SELECT name
FROM person
```

ในกรณีที่เราต้องการที่จะให้ Query กระทำกับเฉพาะตารางที่เราต้องการเราสามารถที่ใช้ keyword ONLY ในคำสั่ง SQL ได้ดังนี้

```
SELECT name
FROM person
ONLY(person)
```

ซึ่งจากความสามารถการสืบทอดกันในตารางนี้เองที่สนับสนุนหลักการของโมเดลของ EER ที่ได้กล่าวไว้ในเรื่องของโมเดลที่จะนำมาใช้กับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ในการที่จะแปลงโมเดลมาเป็นตาราง

5.3 โพลิมอร์ฟิซึม (Polymorphism)

โพลิมอร์ฟิซึม คือการสร้างรูทีนที่มีหน้าที่แตกต่างกันหรือมีหน้าที่เหมือนกันแต่มีอาร์กิวเมนต์ต่างกัน แต่ในการเรียกใช้งานสามารถใช้ชื่อเดียวกันเรียกว่า “One interface, multiple method “ เพื่อง่ายต่อการใช้งาน ซึ่ง IUS อนุญาติให้เราสามารถที่จะกำหนดรูทีนให้มีชื่อเดียวกันแต่มีพารามิเตอร์ที่ต่างกันซึ่งเรียกว่าการทำรูทีน overloading ซึ่งมีประโยชน์เมื่อ

- เรามีการสร้างรูทีนที่มีชื่อเหมือนกับฟังก์ชันภายในที่มากับ IUS แต่ใช้ดำเนินการกับข้อมูลที่ผู้ใช้สร้างหรือกำหนดขึ้นมาใหม่ (user-defined data type)
- เราสร้างลำดับขั้นของชนิดข้อมูลโดยที่ subtype มีการสืบทอดคุณสมบัติต่างจาก supertype รวมถึงรูทีนด้วย

- เราสร้างข้อมูลแบบ distinct ซึ่งมีโครงสร้างภายในของข้อมูลเหมือนกับชนิดข้อมูลต้นแบบที่มีอยู่แล้ว เพียงแต่มีชื่อของข้อมูลต่างกัน ซึ่งจะมีการสืบทอดรูทีนที่ใช้กับชนิดข้อมูลต้นแบบมาด้วยซึ่งเราสามารถจะทำ overload กับรูทีนเหล่านี้ได้

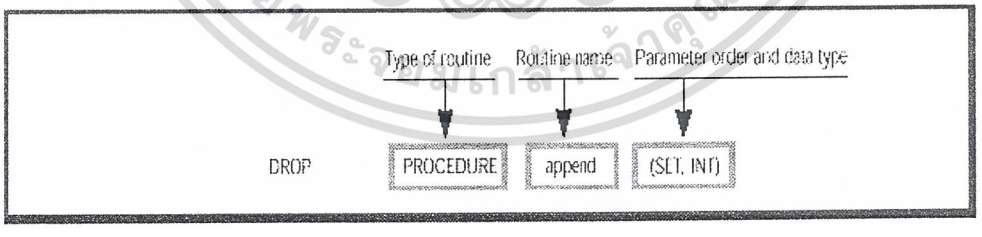
เมื่อมีการ Overload ของรูทีนเกิดขึ้น ตัวระบบฐานข้อมูลสามารถที่จะเรียกใช้งานรูทีนที่มีชื่อเหมือนกันได้ แต่อย่างไรก็ตามตัวระบบฐานข้อมูลไม่สามารถที่จะใช้ชื่อเพียงอย่างเดียวในการที่จะกำหนดว่ารูทีนที่จะทำงานได้

5.3.1 Routine Signature

ในการทำรูทีน Overloading ตัว ระบบฐานข้อมูลจะใช้ signature ของแต่ละรูทีนในการเลือกหรือกำหนดเพื่อการทำงานของรูทีนที่ถูกต้อง โดยรูทีน signature สามารถกำหนดจากข้อมูลดังต่อไปนี้

- ชนิดของรูทีน (ว่าเป็นโพรซีเยอร์หรือฟังก์ชัน)
- ชื่อของรูทีน
- จำนวนของพารามิเตอร์
- ชนิดของข้อมูลของพารามิเตอร์
- ลำดับต่าง ๆ ของพารามิเตอร์
- ชื่อของเจ้าของรูทีนนั้น (เฉพาะมาตรฐาน ANSI เท่านั้น)

จะเห็นว่าใน Signature ของรูทีนจะไม่มีการรวมเอาชนิดของข้อมูลที่ทำการส่งค่ากลับดังนั้นเราไม่ที่จะทำ overload ฟังก์ชัน โดยให้ชนิดของข้อมูลในพารามิเตอร์เหมือนกันแต่ต่างกันแค่ชนิดของข้อมูลที่จะทำการส่งค่ากลับเท่านั้นได้



รูปที่ 5-8 แสดง ข้อมูลที่ใช้ในการกำหนด Signature

5.3.2 Routine-Resolution

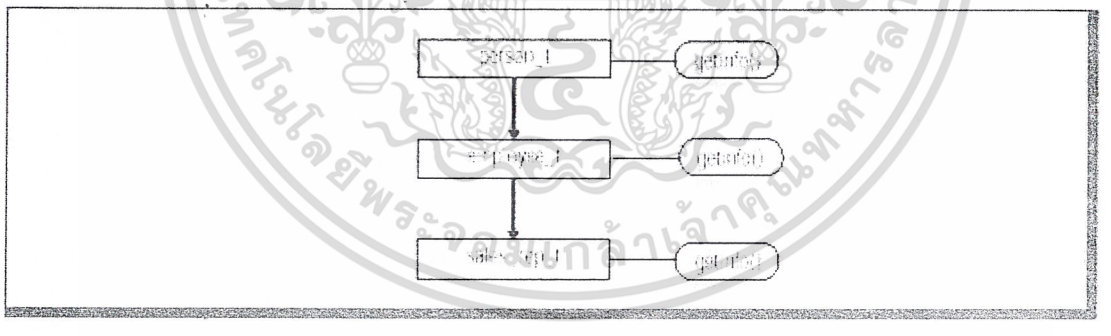
Routine Resolution เป็นกระบวนการในการที่ตัวระบบฐานข้อมูลจะกำหนดว่ารูทีนไหนที่จะถูกเรียกใช้งานเมื่อมีการทำ overload ของรูทีน โดยการเรียกผ่านรูทีน resolution โดยการทำงานของรูทีน resolution ก็คือ เมื่อผู้ใช้มีการเรียกใช้รูทีนขึ้น ตัวระบบฐานข้อมูลจะทำการค้นหา signature ที่ตรงกับชื่อ

ของรูทีนและ พารามิเตอร์ที่ผ่านของรูทีนนั้น ซึ่งถ้าตรงกับ signature ที่มีกำลังค้นหาอยู่ก็จะเรียกใช้งานรูทีนนั้นไป แต่ถ้าไม่ตรงก็จะทำการค้นหาในรูทีนอื่นๆ ถัดไป

เมื่อมีการผ่านพารามิเตอร์มาหลาย ๆ ตัว ระบบฐานข้อมูลจะทำการตรวจสอบพารามิเตอร์จากซ้ายไปขวา โดยถ้าในพารามิเตอร์ตัวแรกตรงกับพารามิเตอร์ในรูทีนมากกว่าหนึ่งรูทีนจะทำการตรวจสอบตัวถัดไปในแต่ละรูทีน โดยจะเลือกเอารูทีนแรกที่ตรงกับพารามิเตอร์ที่ผ่านมามากที่สุด

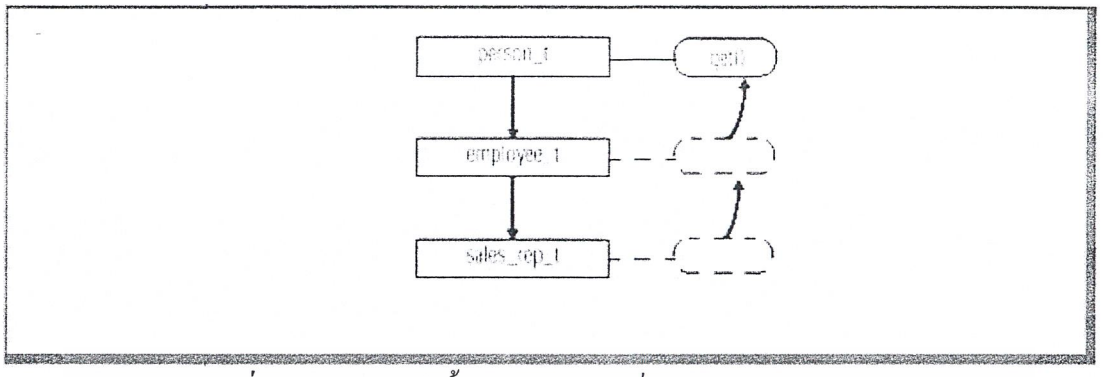
5.3.3 คุณสมบัติการทำ Overloading ของ row type

การทำ Overloading ของรูทีนหมายถึงความสามารถในการที่ทำการกำหนดชื่อให้กับรูทีนหลายๆ รูทีนให้มีชื่อเดียวกันและสามารถที่จะกระทำกับข้อมูลที่เป็นพารามิเตอร์ที่มีชนิดต่างกันได้ ในลำดับขั้นของชนิดข้อมูลนั้น subtype จะได้รับการสืบทอดรูทีนที่กระทำหรือกำหนดให้กับ supertype ในลำดับขั้นของชนิดข้อมูลอย่างไรก็ตามเราสามารถที่จะกำหนดรูทีนใหม่ที่ subtype ได้ซึ่งเรียกว่าเป็นการทำ overriding รูทีนที่มีชื่อเดียวกันรูทีนที่มีใน supertype เช่นสมมุติว่ามีรูทีน getinfo() เป็นรูทีนที่ทำการส่งค่า lastname และ birthdate ของอินสแตนซ์ของ person_t เมื่อเราสร้าง row type employee_t ขึ้นมาใหม่โดยให้ person_t เป็น supertype ซึ่งจะมีการรับรูทีน getinfo() มาด้วยซึ่งทำให้สามารถที่จะเรียกรูทีน getinfo() เพื่อส่งค่า lastname และ birthdate ของอินสแตนซ์ของ employee กลับได้ด้วยเช่นกัน ซึ่งในกรณีที่เราต้องการที่จะใช้รูทีน getinfo() ทำการส่งค่า lastname และ salary ด้วยเราก็สามารถที่จะประกาศรูทีนที่มีชื่อเดียวกันคือ getinfo() ขึ้นมาแต่ให้กระทำหรือดำเนินการที่ต่างกันกับของ supertype ได้ เราเรียกการทำ รูทีนที่มีชื่อเดียวกันแต่การทำงานต่างกันอย่างนี้ว่าการทำ overloading รูทีน



รูปที่ 5-9 แสดงตัวอย่างการทำรูทีน overloading ในลำดับขั้นของชนิดข้อมูล

จากรูปที่ 5-9 จะเห็นว่าแต่ละ Row type จะมีรูทีน getinfo() เป็นของตัวเองกล่าวคือมีการทำ overloading นั้นเอง



รูปที่ 5-10 แสดงลำดับขั้นของชนิดข้อมูลที่ไม่มีการทำ Overloading

จากรูปที่ 5-10 ได้แสดงให้เห็นว่า Row type employee และ sales_rep_t ไม่มีการทำ overloading ของรูทีนคือจะมีการใช้รูทีนของ person_t (เป็นรูทีนที่ได้รับการสืบทอดมาจาก person_t)

5.4 รูทีนที่ผู้ใช้กำหนดขึ้น (User-Defined Routine :UDR)

คือรูทีนที่เราสามารถที่จะกำหนดหรือสร้างขึ้นมาเองและทำการรีจิสเตอร์ลงในตารางรายการของระบบ (System catalog table) และสามารถที่จะเรียกใช้งานโดยคำสั่ง SQL หรือจากรูทีนอื่น ๆ ซึ่งเรามักจะสร้าง UDR เพื่อที่จะดำเนินการกับ complex และข้อมูลที่ผู้ใช้สร้างขึ้นโดย UDR สามารถที่จะเป็นได้ทั้งฟังก์ชันหรือโพรซีเยอร์โดยที่ฟังก์ชัน จะเป็นรูทีนที่อาจจะมีการผ่านพารามิเตอร์ไปให้หรือไม่ก็ได้ โดยฟังก์ชัน จะมีการส่งค่ากลับมาให้ซึ่งต่างจากโพรซีเยอร์ที่ไม่มีการส่งค่ากลับมา ซึ่งข้อแตกต่างกันระหว่างฟังก์ชันและโพรซีเยอร์อีกอย่างคือฟังก์ชัน สามารถที่จะเรียกใช้โดยคำสั่ง SQL ได้แต่โพรซีเยอร์ไม่สามารถจะเรียกใช้โดยคำสั่ง SQL ได้

โดย UDR ยังแบ่งออกเป็น 2 ประเภทคือ

- Stored Procedure Language (SPL) routine : เป็นรูทีนที่เขียนโดยใช้ภาษา SPL
- รูทีนภายนอก (External routine) :เป็นรูทีนที่ไม่ได้ใช้ภาษา SPL แต่ใช้ภาษาอื่น เช่น ภาษา C ซึ่งเราสามารถที่จะใช้รูทีนภายนอกทำงานกับ user-defined data type

ข้อดีของการทำ UDR เพื่อ

- เป็นการซ่อนคำสั่ง SQL หลายๆ คำสั่งเป็นการรวมเอาคำสั่งของ SQL ในหลาย ๆ คำสั่งที่มีความซับซ้อนสูงมาไว้ในคำสั่งเดียว(เป็น 1 routine) ซึ่งในการเรียกใช้ก็จะเรียกผ่านรูทีน (โพรซีเยอร์หรือฟังก์ชัน) ซึ่งจะมีประโยชน์อย่างมากในการทำงานแบบ client/server เช่น ในเรื่องของการบำรุงรักษา (maintain) เพราะสามารถที่จะแก้ไขรูทีนภายหลังได้โดยง่าย เพราะตัว UDR อยู่ที่ ระบบฐานข้อมูลที่เดียวไม่ได้อยู่ที่ฝั่ง client

- เป็นการขยายความสามารถในการทำงานกับชนิดข้อมูลพื้นฐานที่นอกเหนือจากรูทีนที่มีอยู่แล้ว
- เป็นการสนับสนุนชนิดของข้อมูลที่สร้างมาใหม่เนื่องจากชนิดของข้อมูลที่เราสร้างมาใหม่ตัว ระบบฐานข้อมูลไม่รู้จักมาก่อนจึงไม่มีรูทีนที่มาสนับสนุนการทำงาน
- ใช้สร้างการตอนสนองทริกเกอร์ (trigger action) ในหลาย ๆ โปรแกรมใช้งาน (application) โดยตัวทริกเกอร์คือคำสั่งการทำงานประเภทหนึ่งที่มีการทำงานแบบอัตโนมัติเมื่อมีเหตุการณ์ (event) ที่เป็นไปตามที่กำหนดไว้
- กำหนดสิทธิในการเข้าถึงข้อมูลสามารถทำได้โดยผู้บริหารระบบ (administrator) กำหนดสิทธิในการอ้างถึงรูทีนที่สามารถจะใช้กับข้อมูลที่สร้างขึ้นหรือที่มีอยู่แล้วได้นั่นเอง

5.5 วิธีการเข้าถึงข้อมูล (Access Method)

IUS สนับสนุนวิธีการเข้าถึง 2 แบบดังนี้

- Primary access method
- Secondary access methods

5.5.1 Primary access method: IUS สนับสนุนการใช้พื้นที่ภายนอก (external spaces (extspaces)) ซึ่งเป็นตัวจัดเก็บข้อมูล (storage spaces) ที่ระบบฐานข้อมูลไม่สามารถที่จะจัดการได้โดยตรง ซึ่งเราสามารถที่จะใช้พื้นที่ภายนอกเหมือนกับพื้นที่ที่เก็บข้อมูลที่อยู่ในระบบฐานข้อมูล (storage space) สำหรับเก็บตาราง ซึ่งในการเข้าถึงจะใช้ primary access method โดย primary access method ก็คือกลุ่มของฟังก์ชัน ที่ใช้ในการเข้าถึงและจัดการข้อมูลที่จัดเก็บไว้ภายนอกระบบ

IUS ได้นำเสนอตารางเสมือน (Virtual Table Interface (VTI)) มาใช้ช่วยในการทำ primary access method ให้ง่ายขึ้น โดยเราสามารถที่จะเข้าถึงข้อมูลต่อไปนี้โดยผ่าน primary access method

- ตารางในฐานข้อมูลของผู้ผลิตรายอื่น ๆ
- ข้อมูลที่ถูกเก็บใน Sequential files
- การบันทึกข้อมูลผ่าน network

เพราะว่า Primary access method สามารถที่จะถูกนำไปใช้ในโปรแกรมที่เป็น client ได้ ซึ่งโปรแกรมต่าง ๆ เหล่านี้ไม่ต้องการการเข้าถึงข้อมูลในระดับลึก ๆ (low-level database) ภายในระบบฐานข้อมูลซึ่ง primary access method สามารถรวมเอาข้อมูลทั้งหลายที่มีโครงสร้างและอยู่ตามที่ (site) ต่างๆ ที่แตกต่างกันมาไว้ภายในระบบฐานข้อมูลเชิงสัมพันธ์ เพียงระบบเดียวได้

VTI จะประกอบไปด้วยฟังก์ชัน ที่เป็นเมธทอดในการเข้าถึงต่าง ๆ เราสามารถที่ทำการเขียนและรีจิสเตอร์เมธทอดสำหรับใช้ในการเข้าถึงนี้ ซึ่งจะใช้ในการเข้าถึงข้อมูลที่อยู่ในฐานข้อมูลในลักษณะการควบคุมแบบรีโหมด (remote database) โดยในการทำรีจิสเตอร์จะต้องในคำสั่ง SQL ดังนี้


```
CREATE ACCESS_METHOD
```

เมื่อเราสร้างตารางเราจะต้องมีการระบุวิธีการเข้าถึงซึ่งจะถูกนำไปใช้เมื่อมีการใช้คำสั่ง SQL กับตารางนี้

5.5.2 Secondary Access Method

IUS สนับสนุนการใช้ secondary access methods 2 แบบคือ

- Generic B-trees
- R-trees

- **Generic B-Trees**

อินเด็กซ์แบบ B-Tree เป็นวิธีการทำอินเด็กซ์ที่ใช้กันในระบบฐานข้อมูลแบบ RDBMS ซึ่ง IUS จะใช้อินเด็กซ์แบบ B-tree กับ

- คอลัมน์ที่มีชนิดข้อมูลเป็นแบบข้อมูลพื้นฐาน (built-in data types)
- ชนิดข้อมูลที่ผู้ใช้สร้าง (User-defined data type) ที่มีลักษณะหนึ่งมิติ
- ค่าที่ส่งค่าจากฟังก์ชันที่ผู้ใช้สร้างที่เป็นชนิดข้อมูลพื้นฐานหรือชนิดที่ผู้ใช้กำหนดที่ไม่ใช่ข้อมูลแบบ Large object type

- **R-Trees**

IUS ได้มีการเสนอการทำอินเด็กซ์แบบใหม่นอกเหนือจากการทำอินเด็กซ์แบบ B-tree ที่เคยมีในระบบฐานข้อมูลแบบสัมพันธ์เดิมที่เรียกว่าอินเด็กซ์แบบ R-tree ซึ่งเป็น secondary access ที่เหมาะกับข้อมูลแบบ

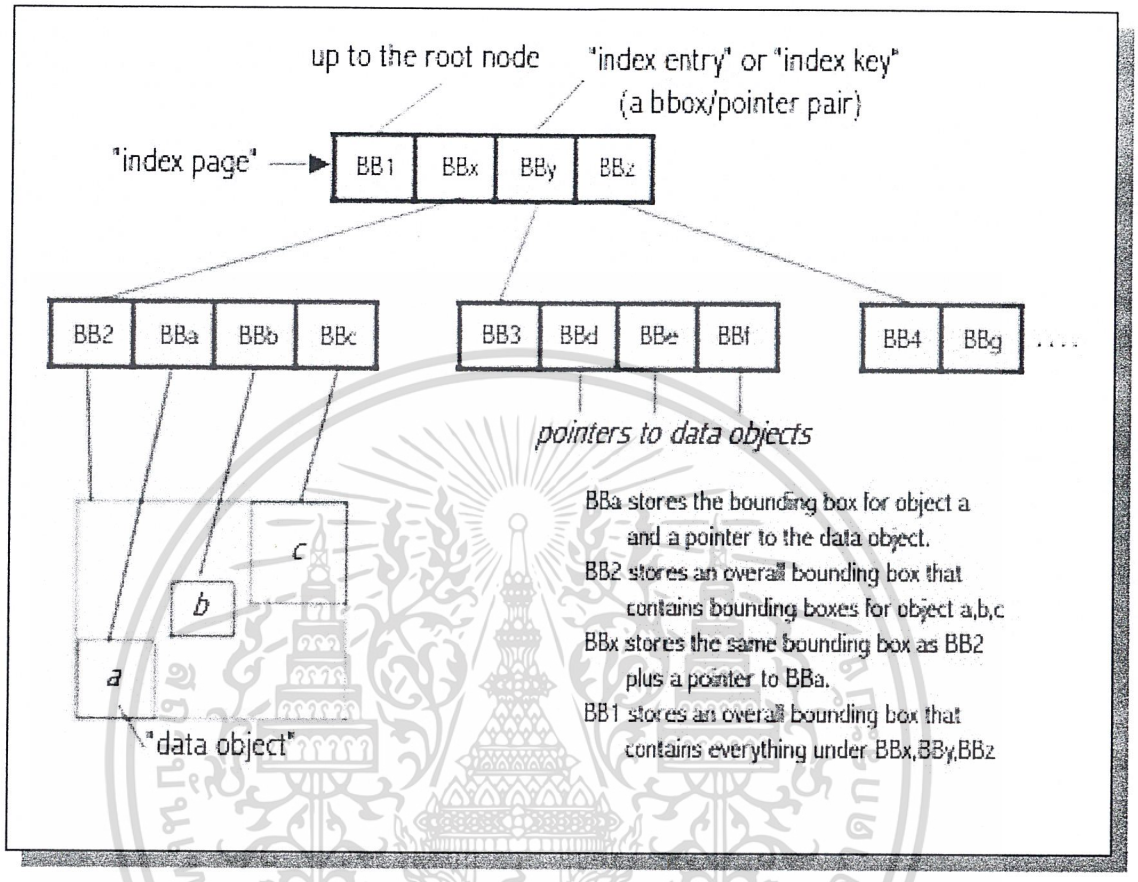
- หลายมิติ (Multi-dimension data)
- ข้อมูลที่มีลักษณะแบบ natural grouping คือมีการมองข้อมูลที่เราพิจารณาอยู่ว่าอยู่ใน (contain) หรือ เหลื่อม (overlap) กับค่าข้อมูลอื่นๆ หรือไม่
- ข้อมูลในลักษณะที่เป็นพื้นที่ (Spatial data) เช่น ต้องการค้นหาค่า latitude หรือ longitude ที่ได้มีการกำหนดโพลิกอน (polygon) หรือขอบเขตเอาไว้

ในการทำอินเด็กซ์แบบ R-treeบนชนิดข้อมูลที่กำหนดเอง UDT เริ่มแรกจะต้องทำการสร้าง UDR ที่ R-tree ต้องการใช้กับข้อมูลที่เราสร้างขึ้น จากนั้นก็จะต้องสร้างตัวดำเนินการ (operator class) ซึ่งก็คือชุดของ UDR หลายๆ รูทีนโดยตัวดำเนินการเหล่านี้ จะใช้เพื่อ

- ใช้ใน R-tree ในการเก็บและค้นหาค่าของข้อมูลชนิดที่เราสร้างขึ้น
- บอกตัวหาเส้นทาง (optimizer) ว่าสามารถที่จะใช้อินเด็กซ์ในคำสั่ง SQL อย่างไร

โดยการทำงานของ R-tree จะอาศัยหลักการที่เรียกว่า bounding box โดยลักษณะโครงสร้างของอินเด็กซ์จะเป็นต้นไม้เหมือนกับ B-tree แต่จะเป็นกลุ่มของขอบเขตของข้อมูล (bounding box) ซึ่งแต่ละอินเด็กซ์เพจจะประกอบด้วยหลายๆ อินเด็กซ์โดยอินเด็กซ์คีย์ (index key) จะเก็บค่าของขอบเขตข้อมูลโดยที่

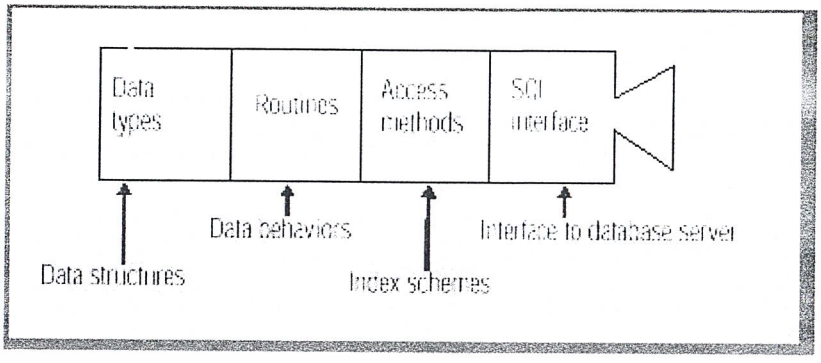
แต่ละ parent index page จะเก็บขอบเขตทั้งหมดที่อยู่ใน children index page โดยลักษณะโครงสร้างจะเป็นดังนี้



รูปที่ 5-11 แสดงโครงสร้างของ R-tree index

5.6 DataBlade โมดูล

DataBlade โมดูลคือกลุ่มของออบเจ็กต์และโค้ดที่จะเพิ่มเข้ามาในระบบฐานข้อมูลเพื่อเพิ่มหน้าที่การทำงานหรือความสามารถใหม่ๆ เข้าไป โดย DataBlade โมดูลจะทำให้ระบบฐานข้อมูลมีการสนับสนุนชนิดข้อมูลชนิดใหม่ ๆ ขึ้นมา โดยที่ชนิดข้อมูลชนิดนั้นอาจจะมีการใช้ชนิดข้อมูลพื้นฐาน, UDT เป็นต้น โดยเราอาจจะมอง DataBlade โมดูลเป็นเสมือนกับ object –oriented package คล้ายๆกับคลาส ในภาษา C++ ซึ่งเป็นข้อมูลชนิดพิเศษที่มีคุณสมบัติในการซ่อนเร้นตัวอย่างเช่น ข้อมูลชนิด image เป็นต้น

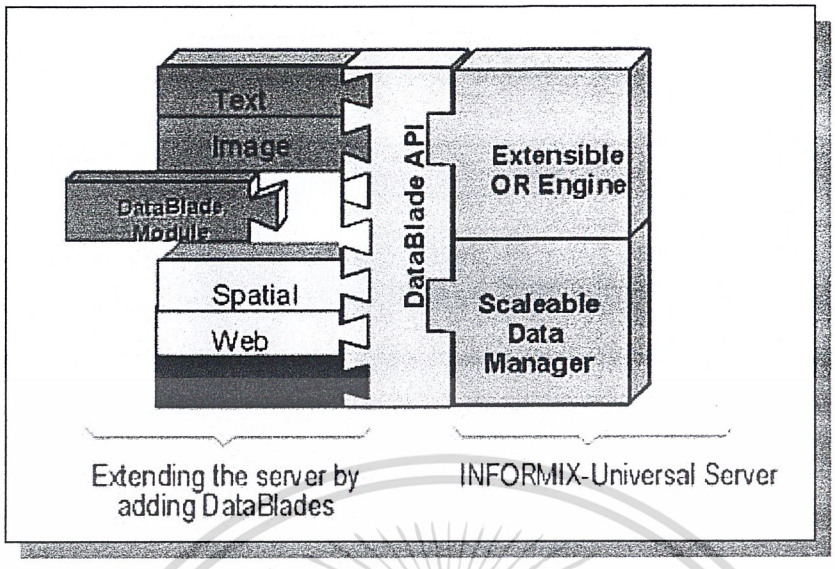


รูปที่ 5-12 แสดงคอมโพเนนต์ของ DataBlade โมดูล

DataBlade Module Component	Description
Data Types	Data type เป็น User-defined data type หรือ เป็น collection ของ User–defined data type
Routines	เป็น routines ที่ทำงานกับ data type ที่เรากำหนดขึ้นในส่วนของ data types component
Access Methods	Access methods ที่จะใช้ทำงานบน table และ index ที่จัดการโดย database server โดยผู้พัฒนา DataBlade module สามารถที่กำหนด index ที่จะใช้กับข้อมูลชนิดใหม่โดยอาจจะใช้ index ที่มีอยู่แล้วหรือจะเพิ่ม access method ของตนเอง
SQL Interface	SQL interface เป็น collection ของฟังก์ชัน ที่จะนำไปใช้กับคำสั่ง SQL

ตารางที่ 5-6 แสดงส่วนประกอบต่างๆ ของ DataBlade module

เราสามารถใช้ DataBlade โมดูลของอินฟอร์มิกซ์หรือจากผู้ผลิตรายอื่น ๆ หรือเราสามารถที่จะสร้าง DataBlade โมดูลเองได้ เพื่อนำไปใช้งานกับโปรแกรมต่างๆ โดย DataBlade โมดูลจะถูกสร้างโดยโปรแกรม Bladesmith โดยจะใช้ร่วมกับโปรแกรม BladePack และ BladeManager เพื่อทำการรีจิสเตอร์ DataBlade โมดูลลงระบบฐานข้อมูล



รูปที่ 5-13 แสดงการนำ DataBlade Module เพิ่มเข้าไปใน Server

5.7 ชนิดข้อมูลแบบ Large-Object

ชนิดข้อมูลแบบ Large-Object เป็นชนิดข้อมูลพื้นฐานที่มากับระบบฐานข้อมูลที่ IUS มีให้ ซึ่งมีอยู่ด้วยกัน 2 แบบ

- **แบบ Simple-Large object** มีขนาดได้ไม่เกิน 2^{32} byte และโดยในการใช้งานจริงๆ จะขึ้นอยู่กับขนาดของ disk ที่ใช้กับ simple large object มี 2 ชนิดด้วยกันคือ
  - **BYTE** ที่ใช้เก็บข้อมูลแบบ binary data
  - **TEXT** ที่ใช้เก็บข้อมูลแบบ text data

Simple large object ไม่สนับสนุนการทำการ random access บนข้อมูล ซึ่งหมายความว่าถ้าต้องการที่จะเข้าถึงข้อมูลก็จะต้องถึงเป็น BYTE หรือเป็น TEXT คือต้องเข้าถึงข้อมูลทั้งชุด

- **Smart-Large object** เป็นชนิดข้อมูลแบบ large object ที่สามารถจะทำการค้นหา อ่าน เขียน (seek, read, write) ข้อมูลภายในของออบเจ็กต์ได้ นอกจากนี้ยังทำการ recovery และ transaction rollback ได้อีกด้วย smart large object มีขนาดได้สูงสุดถึง 4 terabyte โดย Smart large object data type สามารถแบ่งออกเป็น 2 ชนิดคือ
  - **BLOB** ใช้จัดเก็บ binary data เช่น spreadsheet,program-load module,digitized voice pattern และ images เป็นต้น
  - **CLOB** ใช้จัดเก็บ text data เช่น ไฟล์ Postscript,HTML และ SGML เป็นต้น

โดยเราสามารถที่จะใช้ฟังก์ชัน เหล่านี้ในคำสั่ง SQL กับข้อมูลแบบ smart large object ได้

- **FILETOBLOB** ทำการก็อปปี้ไฟล์ข้อมูลไปไว้ที่คอลัมน์ที่เป็นชนิด BLOB (โดยที่ CLOB จะใช้คำสั่ง FILETOCLOB)
- **LOTOFILE** ทำการก็อปปี้ค่าใน BLOB หรือ CLOB ไปให้กับ operating system file
- **LOCOPY** ทำการ copy smart large object ที่มีอยู่แล้วไปให้กับ smart large object ใหม่

วิธีการกำหนดคอลัมน์ที่เป็นแบบ Smart large object ในตารางสามารถที่จะทำได้ดังนี้

- โดยการกำหนดคอลัมน์เป็นข้อมูลแบบ CLOB หรือ BLOB โดยตรง
- โดยสร้างข้อมูลแบบ opaque ซึ่งภายในประกอบด้วยข้อมูลแบบ smart large object

ในการสร้างและการเข้าถึงข้อมูลแบบนี้จะต้องทำผ่าน DataBlade API การกำหนดคอลัมน์ที่มีชนิดข้อมูลแบบ smart large object ในคอลัมน์ของตารางจะไม่ทำการเก็บตัวข้อมูลแบบ smart large object ไว้ในคอลัมน์ของตารางจริงๆ แต่จะเก็บ Lo-handle แทน ซึ่งจะเป็นตัวเก็บตำแหน่งของ smart large object อีกทีหนึ่ง

5.7.1 การใช้ Smart Large Object

ในการกำหนดคอลัมน์ในตารางหรือการกำหนดแอททริบิวต์ในข้อมูลแบบ Opaque ให้มีชนิดเป็น BLOB หรือ CLOB นั้น เราจะต้องมีการจองเนื้อที่สำหรับเก็บข้อมูลเหล่านี้ไว้ที่เรียกว่า sbspace ซึ่งในการใช้งานจริงตัว Lo-handle จะชี้มาที่ sbspace นี้ ส่วนในคำสั่ง SQL เราไม่สามารถที่จะใช้ BLOB และ CLOB ใน

- คำสั่งที่เกี่ยวกับคณิตศาสตร์หรือบูลีน (Arithmetic and Boolean expression)
- GROUP BY หรือ ORDER BY clause
- UNIQUE test
- อินเด็กซ์ที่เป็นแบบ B-tree (อย่างไรก็ตามใน DataBlade เราสามารถที่จะสร้างอินเด็กซ์บนคอลัมน์ที่เป็นแบบ CLOB ได้)

โดยในคำสั่ง SELECT เราสามารถที่จะ

- ระบุค่า NULL เป็นค่าเริ่มต้น (default) ตอนสร้างตารางโดยใช้ DEFAULT NULL clause
- ไม่อนุญาตให้เป็น NULL โดยใช้กฎบังคับ (constraint) NOT NULL ตอนสร้างตาราง
- ใช้ IS [NOT] NULL

ตัวอย่าง สมมุติเราจะสร้างตาราง inmate และ fbi_list ดังนี้

```
CREATE TABLE inmate
(
    id_num   INT,
    picture  BLOB,
```

```
        felony    CLOB
);

CREATE TABLE fbi_list
(
        id              INTEGER,
        mugshot    BLOB
) PUT mugshot IN (sbspace1);
```

คำสั่งต่อไปนี้แสดงการแทรกข้อมูลโดยใช้ FILETOBLOB() และ FILETOCLOB()ฟังก์ชัน เพื่อทำการแทรกข้อมูลในตาราง inmate

```
INSERT INTO inmate
VALUES (437, FILETOBLOB('datafile',client),FILETOCLOB('tmp/text','server'))
```

โดยอาร์กิวเมนต์แรกของ FILETOBLOB() และ FILETOCLOB() เป็นการระบุ path ของไฟล์ (source file) ที่จะทำการก็อปปี้ ไปที่ BLOB หรือ CLOB ส่วนอาร์กิวเมนต์ที่สองเป็นการระบุว่าไฟล์ (source file) อยู่ที่ฝั่งไหน (Server หรือ Client)

คำสั่งต่อไปเป็นการแสดงการแก้ไขโดยใช้ LOCOPY() ฟังก์ชัน ทำการก็อปปี้ BLOB จากคอลัมน์ mugshot ในตาราง fbi_list มาที่คอลัมน์ picture ของตาราง inmate

```
UPDATE  inmate(picture)
SET picture=LOCOPY(mugshot,'fbi_list','mugshot')
WHERE inmate.id_num = 437 and fbi_list.id = 669;
```

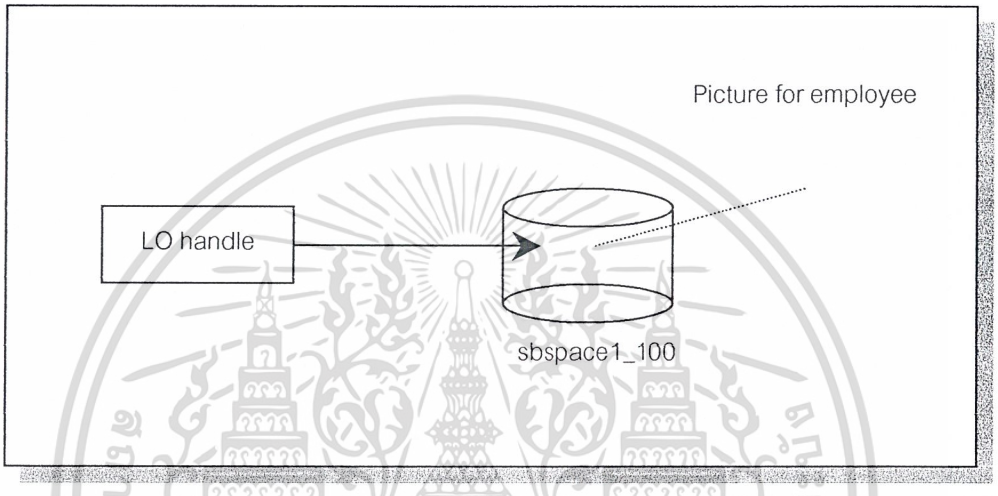
โดยอาร์กิวเมนต์แรกของฟังก์ชัน LOCOPY() ระบุถึงคอลัมน์ที่จะทำการ export Smart large object ส่วนอาร์กิวเมนต์ที่สองระบุถึงตาราง และอาร์กิวเมนต์ที่สามระบุคอลัมน์ที่จะใช้ storage characteristics ในการสร้าง large Object ใหม่

ส่วนคำสั่งต่อไปจะแสดงถึงคำสั่งในการเลือกข้อมูลที่ใช้ฟังก์ชัน LOTOFILE ที่ทำการก็อปปี้ ข้อมูลจากคอลัมน์ felony ไปไว้ที่ fenol_322.txt ที่อยู่ที่ฝั่ง client

```
SELECT id_num, LOTOFILE(felony,'fenol_322.txt','client)
FROM  inmate
WHERE id=322
```

5.7.2 การอ้างถึง Smart large object

ในการอ้างถึง Smart large object ไม่ว่าจะเป็นในคอลัมน์ที่มีการกำหนดชนิดข้อมูลเป็นแบบ BLOB หรือ CLOB โดยตรงหรือจะเป็นการอ้างในแอททริบิวต์ที่อยู่ข้อมูลแบบ opaque ก็ตามจะอ้างโดยผ่านส่วนที่เรียกว่า LO-Handle ซึ่งเป็นข้อมูลแบบ Opaque ชนิดหนึ่งที่จะถูกเก็บในคอลัมน์หรือในแอททริบิวต์ที่มีชนิดเป็น Smart large object โดยที่ LO-Handle จะชี้ไปที่ sbspace ซึ่งเป็นส่วนที่ทำการจัดเก็บ BLOB หรือ CLOB จริงๆ



รูปที่ 5-14 แสดงการอ้าง BLOB หรือ CLOB ผ่าน Lo-handle

โดย Sbspace เป็นตัวจัดเก็บ (logical storage) ซึ่งภายในจะประกอบด้วย 2 ส่วน

- ส่วนที่เก็บข้อมูล (metadata area) เป็นส่วนที่เก็บข้อมูลเกี่ยว smart large object ซึ่งประกอบด้วย
  - ข้อมูลภายใน (internal information) ที่ใช้สำหรับ smart large object optimizer ในการจัดการกับข้อมูล
  - คุณสมบัติของตัวจัดเก็บ (storage characteristic) ของ smart large object
  - ข้อมูลของสถานะ (status information) ของ smart large object
- user data area เป็นส่วนที่เก็บข้อมูลของ smart large object

❑ **ข้อมูลเกี่ยวกับ smart large object**

จะเก็บไว้ในส่วนของ Metadata ซึ่งประกอบด้วย

- คุณสมบัติของตัวจัดเก็บข้อมูล (storage characteristic)
- ข้อมูลเกี่ยวกับสถานะ (status information)

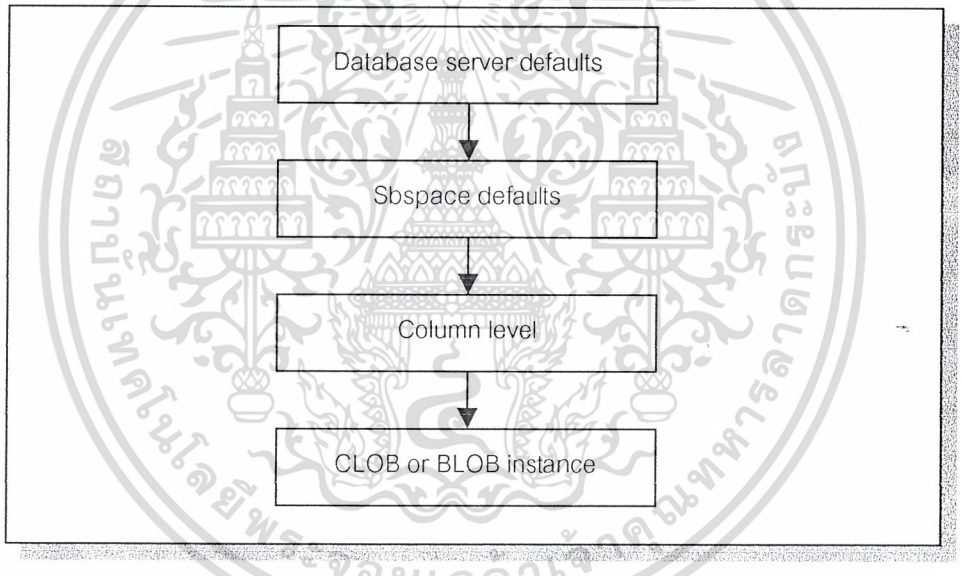
- คุณสมบัติของตัวจัดเก็บข้อมูล

จะประกอบด้วยข้อมูล

- ข้อมูลเกี่ยวกับ Disk-storage เป็นข้อมูลที่ใช้สำหรับจัดการกับข้อมูลบนดิสก์ (disk) เช่น allocation extent information,sizing information,location information
- ข้อมูลเกี่ยวกับแอททริบิวต์เป็นการกำหนดว่าต้องการให้มี option อะไรบ้างในการเข้าถึงข้อมูลซึ่งประกอบด้วย loggig indicator,last-access-time

ซึ่งในการระบุคุณสมบัติของการจัดเก็บทำได้หลายระดับ โดยสามารถระบุได้ตอน

- สร้าง sbspace โดยใช้ onspace utility
- สร้างตารางโดยใช้คีย์เวิร์ด PUT clause ในคำสั่ง CREATE TABLE
- สร้าง Smart large object โดยใช้ DataBlade API ฟังก์ชัน



รูปที่ 5-15 แสดง Storage-Characteristics Hierarchy

- ข้อมูลเกี่ยวกับสถานะ

เป็นข้อมูลที่เกี่ยวกับสถานะต่างๆของ Smart large object ที่จัดเก็บไว้ในส่วนของ meta data area ของ sbspace

5.7.3 การสร้างและการเข้าถึง Smart large object ผ่านทาง DataBlade API

ในอินฟอร์มิกซ์ได้มีส่วนที่ใช้ในการติดต่อ (Interface) ต่างๆ ที่ใช้กับ Smart large object โดยประกอบด้วยโครงสร้างข้อมูลที่ใช้เก็บข้อมูลต่างๆ เกี่ยวกับ Smart large object ดังนี้

Smart large object data sturcture	Data Type	Description
LO specitication	MI_LO_SPEC	เก็บ storage characteristics
LO handle	MI_LO_HANDLE	ใช้ระบุถึงตำแหน่ง
LO file descriptor	MI_LO_FD	ระบุถึงการ open Smart large object
LO status	MI_LO_STAT	เก็บ status ต่างๆ ของ Smart large object

ตาราง 5.7 *แสดงโครงสร้างข้อมูลที่ใช้เก็บข้อมูลต่างๆ เกี่ยวกับ Smart large object*

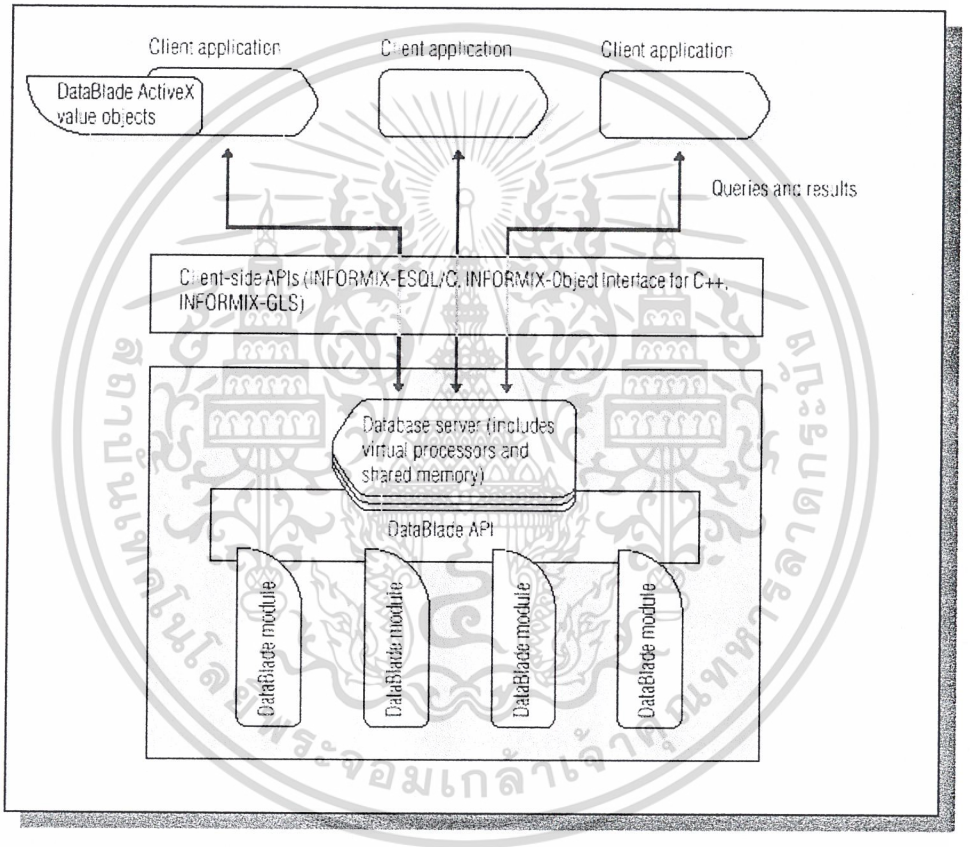
การดำเนินการที่กระทำกับ Smart large object ประกอบด้วย

- สร้าง (Create)
- การเข้าถึง (Access)
- การดูสถานะ (Obtain status)
- ลบ (Delete)

ซึ่งการเข้าถึงข้อมูลแบบ Smart large object โดยผ่านทาง API นี้จะกระทำโดยผ่านทางภาษา C ซึ่งจะต้องใช้โปรแกรม Microsoft Visual C++ ในการเขียน code และคอมไพล์โดยที่ API ต่างๆ เหล่านี้จะอยู่ใน header file (mili.h) ที่ทางอินฟอร์มิกซ์ได้ให้มา ดังนั้นก็จำเป็นต้องทำการรวม (include) ไฟล์นี้ไปด้วยในการใช้งาน API

บทที่ 6
การสร้าง DataBlade โมดูล

ในบทนี้จะกล่าวเกี่ยวกับการพัฒนา DataBlade โมดูลซึ่งถือว่าเป็นคุณลักษณะพิเศษที่ทาง IUS ได้อนุญาตให้เราสามารถที่จะสร้างหรือกำหนดหน้าที่การทำงานต่างๆ ที่โปรแกรมของเราต้องการใช้ที่นอกเหนือจากที่ทาง IUS มีให้ โดยจะเริ่มกล่าวจากการออกแบบ การสร้างและการนำไปใช้งานในระบบฐานข้อมูล



รูปที่ 6-1 Informix Universal Server Internal Architecture

6.1 การออกแบบ DataBlade โมดูล

ในการออกแบบ DataBlade โมดูลจะประกอบด้วยการออกแบบส่วนประกอบต่างๆ ของ DataBlade โมดูลดังนี้

- **Data Model** ซึ่งเป็นการกำหนดออบเจ็กต์และบริการ (service) ต่างๆ ที่อยู่ใน DataBlade โมดูลในระดับบน (High-level)
- **Data Type Design** เป็นการกำหนดชนิดของข้อมูลที่จะใช้

- **Query Language Interface** เป็นการกำหนดส่วนที่ใช้สำหรับ ผู้ใช้ฐานข้อมูลติดต่อกับ DataBlade โมดูล
- **Query Processing** เป็นการกำหนดการดำเนินการต่างๆของ DataBlade โมดูลภายในระบบฐานข้อมูล
- **Interoperability** เป็นกำหนดว่า DataBlade โมดูลที่จะสร้างใหม่นี้สามารถที่จะใช้ DataBlade โมดูลเก่าได้หรือไม่

6.1.1 โมเดลของข้อมูล (DataModel)

เป็นกำหนดในระดับบน (High-level definition) ที่จะต้องกำหนดว่าอะไรคือออบเจ็กต์และการดำเนินการที่จะต้องสร้างขึ้นมาโดยในการออกแบบจะคำนึงถึง

- data model จะต้องไม่ขึ้นอยู่กับโปรแกรมใช้งาน (application) และการติดต่อผู้ใช้ (user interface)
- ในการออกแบบจะต้องเน้นในเรื่องของบริการ (service) ที่ต้องการมากกว่าการที่จะนึกถึงว่าโปรแกรมใช้งานจะนำไปใช้งานอย่างไร
- ในการออกแบบโมเดลข้อมูลควรที่จะนำไปใช้งานได้โดยหลายๆ โปรแกรม client ตัวอย่าง เช่นเราต้องการที่จะสร้าง Circle DataBlade โมดูลซึ่งทำหน้าที่เก็บข้อมูล จัดการ และทำการคำนวณหาพื้นที่ของวงกลม ซึ่งจากความต้องการดังกล่าวเราอาจจะออกแบบให้ โมดูลนี้ประกอบด้วย
  - ข้อมูลชนิดวงกลม
  - การดำเนินการที่ทำหน้าที่ในการคำนวณหาพื้นที่และเปรียบเทียบระหว่างสองวงกลม

6.1.2 ออกแบบชนิดของข้อมูล (Data Type Design)

หลังจากที่ได้มีการออกแบบโมเดลของข้อมูลแล้วก็ถึงขั้นตอนการออกแบบชนิดของข้อมูลที่จะใช้ใน DataBlade โมดูลโดยในการออกแบบจะต้องคำนึงถึง

- ว่ามีความต้องการออบเจ็กต์ที่ประกอบด้วย element ภายในหรือไม่และลักษณะการเข้าถึงข้อมูลแต่ละ element เป็นอย่างไร
- แต่ละออบเจ็กต์มีขนาดใหญ่แค่ไหน

ซึ่งจากข้อแรกถ้าเราต้องการให้ภายในข้อมูลประกอบด้วยอีลิเมนท์ (Element) ต่างๆเราก็จะพิจารณาข้อมูลแบบ row type และแบบ opaque โดยเราจะใช้

- Row type ก็ต่อเมื่อเราต้องการให้ผู้ใช้สามารถที่จะเข้าถึงข้อมูลภายในหรืออีลิเมนท์ต่างๆ ได้โดยตรง
- Opaque type เมื่อต้องการซ่อนรายละเอียดและข้อมูลต่างๆ จากผู้ใช้

ซึ่งจากตัวอย่าง Circle DataBlade โมดูลถ้าสมมุติว่าเราไม่ต้องการให้ผู้ใช้ทำการเข้าถึงข้อมูลได้โดยตรง และเพื่อต้องการซ่อนรายละเอียดต่างๆ จากผู้ใช้ เราก็จะเลือกใช้ข้อมูลชนิด Opaque ซึ่งข้อดีอีกอย่างของการใช้ Opaque ก็คือเราสามารถที่จะเขียนรูทีนต่างๆ มาใช้กับข้อมูลโดยใช้ภาษา C ซึ่งจะทำให้ในการดำเนินการกับข้อมูลง่ายขึ้น

ส่วนในการพิจารณาขนาดของออบเจ็กต์นั้น เราจะต้องคำนึงเสนอว่าขนาดของแถว (Row) ในตารางฐานข้อมูลที่ทาง IUS กำหนดมาให้นั้นมีขนาดสูงสุดได้ 32 KB (ซึ่งเป็นขนาดรวมทุกคอลัมน์ในตาราง) ซึ่งถ้าเราต้องการที่จะเก็บข้อมูลที่มีขนาดใหญ่มาก เราก็สามารถที่จะใช้ข้อมูลชนิด smart large object แทน ซึ่งโดยทางลอจิคอล (logical) แล้วจะเก็บข้อมูลในคอลัมน์ของตารางเลยแต่ในทางฟิสิคอล (physical) จะถูกเก็บไว้ที่ sbspace ดังที่ได้กล่าวไว้แล้วก่อนหน้านี้

ซึ่งข้อดีของข้อมูลชนิดนี้คือ

- สามารถที่จะทำ random access ได้ (คือสามารถค้นหา อ่าน เขียน)
- สามารถที่จะทำการ recovery ได้เมื่อเกิด systen crash และสามารถทำ transaction isolation ได้
- มีขนาดไม่จำกัด
- สามารถที่จะสร้างและเก็บอินเด็กซ์ได้
- สามารถที่จะเข้าถึงและจัดการบนข้อมูลชนิดนี้โดยใช้ SQL,ESQL/C หรือ DataBlade API

ซึ่งสำหรับการเข้าถึงและจัดการ โดยใช้ SQL จะทำได้เฉพาะการเปรียบเทียบข้อมูล BLOB หรือ CLOB ว่าเท่ากันหรือไม่เท่านั้น ส่วนการดำเนินการอื่นๆจะต้องใช้ผ่านทางอินฟอร์มิกซ์ ESQL/C หรือ DataBlade API เท่านั้น

6.1.3 ส่วนที่ใช้ติดต่อกับภาษาคิวรี่ (Query Language Interface)

DataBlade โมดูลสามารถขยายความสามารถของ SQL โดยการกำหนดหรือสร้างชนิดของข้อมูลและรูทีนใหม่มาช่วยในการทำคิวรี่โดยภาษา SQL จะประกอบด้วยชุดคำสั่ง 2 ประเภทคือ Data Definition Language (DDL) เช่น คำสั่ง CREATE, ALTER และ DROP และ Data Manipulation Language (DML) เช่นคำสั่ง SELECT, INSERT, UPDATE และ DELETE ซึ่งปกติส่วนมากเราจะใช้คำสั่ง DML มากกว่า DDL ดังนั้นในการออกแบบส่วนที่ติดต่อกับภาษาที่ใช้คิวรี่จะต้องพิจารณา DML ในลักษณะที่เป็นนามธรรมหรือนิยามในรูปแบบดังนี้

SELECT *something* **FROM** *some table*

WHERE *some condition are satisfied*

หรือ

UPDATE *some table* **SET** *something*

WHERE *some condition are satisfied*

ซึ่งจากส่วนที่เป็นอักษรตัวเอียงแต่ละตัวจะมีหน้าที่ต่างกันใน DML คิวรี่โดย *some table* จะเรียกว่า "the from list" ซึ่งไม่จำเป็นต้องพิจารณาในการออกแบบ ส่วน *something* เป็นส่วนที่เรียกว่า "the target list" เป็นส่วนที่กำหนดคอลัมน์ที่จะทำการนำข้อมูลมาแสดง (retrives) หรือแก้ไข (update) ข้อมูล และ *some condition are satisfied* เป็นส่วนที่เรียกว่า "a qualification" เพราะว่าเป็นตัวกำหนดว่าแถวใดที่จะได้รับการดำเนินการ (operation) ซึ่งในการออกแบบ DataBlade โมดูลจะต้องพิจารณาว่าเราจะกำหนดให้ใช้รูทีนที่ไหนของคิวรี่ได้บ้างจากสองส่วนข้างต้นว่าจะอยู่ที่รายการปลายทาง (target list) หรือในส่วนกำหนดว่าแถวใดจะได้รับการดำเนินการโดยที่

- **รายการปลายทาง (The Target List)**

  รูทีนที่ใช้ในส่วนนี้มักจะเป็นรูทีนที่ใช้ในการคำนวณเพื่อช่วยลดจำนวนการถ่ายโอน (Transfer) ข้อมูลจากเซิร์ฟเวอร์มาที่ไคลเอนต์และเป็นการเพื่อประสิทธิภาพในการทำงานเนื่องจากรูทีนนี้จะอยู่ที่ฝั่งเซิร์ฟเวอร์ที่สามารถทำงานในลักษณะ parallelism และ sacalability

  ตัวอย่าง รูทีนที่ใช้ในส่วนของรายการปลายทางโดยสมมุติเรามีตารางที่ประกอบด้วยคอลัมน์ต่างๆ ดังนี้

```
CREATE TABLE circle_tab
(
        circle_id       SERIAL,
        color           VARCHAR(8),
        circle_col      mo_circle
);
```

  สมมุติเราต้องการที่จะแสดงข้อมูลข้อมูล id ของวงกลม circle และพื้นที่ของวงกลมเราสามารถที่เขียนคิวรี่ได้ดังนี้

```
SELECT circle_id, circle_col, area(circle_col)
FROM circle_tab;
```

- **การกำหนดจำนวนข้อมูลที่จะดำเนินการ (The Qualification)**

การกำหนดจำนวนข้อมูลที่จะดำเนินการใน SQL เป็นตัวที่ทำหน้าที่กรองเอาแถวของตารางที่เราไม่สนใจออกไป เพื่อไม่ต้องนำมาพิจารณาในส่วนของรายการปลายทางหรือรายการที่เราสนใจอีกที ดังนั้นส่วนนี้จะเป็นเครื่องมือที่มีประสิทธิภาพกว่าส่วน target list โดยทั่วไปประพจน์ (expression) ที่อยู่ในส่วนที่กำหนดการกระทำกับข้อมูล (qualification) จะเรียกว่านิพจน์ (predicate) โดยในกำหนดการกระทำกับข้อมูลสามารถจะประกอบด้วยหลายนิพจน์โดยอาศัยตัวดำเนินการทางตรรกศาสตร์ AND และ OR ซึ่งถ้ารูทีนของ DataBlade โมดูลนี้ใช้ในการกำหนดการกระทำกับข้อมูลจะทำหน้าที่กรองแถวในตารางที่จะส่ง

กลับมาที่ไคลเอนต์โดย IUS สามารถที่จะใช้ค่าของข้อมูลชนิดใหม่ในการกรองข้อมูลหรือแถวในตาราง ได้ซึ่งเป็นความสามารถที่ไม่มีในระบบฐานข้อมูลแบบสัมพันธ์ ในบางครั้งรูทีนของเราสามารถที่จะใช้ได้ทั้งในส่วนของ Target list และส่วนกำหนดการกระทำกับข้อมูล

ตัวอย่าง รูทีนที่ใช้ในส่วนของกำหนดการกระทำกับข้อมูล โดยสมมุติเราต้องเลือกวงกลมที่มีพื้นที่เท่ากับ 28.2743334 เราสามารถจะใช้คิวรี่ได้ดังนี้

```
SELECT circle_id,circle_col
FROM area(circle_col)= 28.2743334;
```

6.1.4 การปฏิบัติและการทำงานของคิวรี่ (Query Processing)

ในการพัฒนา DataBlade โมดูลเราจำเป็นต้องทราบหรือเข้าใจถึงการทำงานของคิวรี่ (query processing) และ การเรียกคำสั่ง SQL ให้ทำงานใน IUS ว่ามีกำหนดสภาวะแวดล้อมในการทำงานอย่างไรบ้าง โดยในหัวข้อนี้จะพิจารณาถึง

- **การประเมินนิพจน์ (Predicate Evaluation)**

นิพจน์ (Predicate) ที่ใช้ในส่วนนี้เป็นการกำหนดการทำงานกับข้อมูลเพราะว่าระบบฐานข้อมูลจะต้องทำการประเมินนิพจน์กับข้อมูลทุกแถวในตาราง เพื่อที่จะส่งแถวที่ตรงตามที่ได้ทำการกำหนดหรือระบุ (qualify) ไว้กลับมาให้

- **การพิจารณาการใช้ทรัพยากรของรูทีน (Expensive Routine)**

ในระบบฐานข้อมูลแบบ ORDBMS จะต้องมีการประเมินทรัพยากรต่างๆ ที่ฟังก์ชันนั้นๆ ใช้งาน (function costs) เนื่องจากบางรูทีนที่ผู้ใช้ได้สร้างขึ้นมามีลักษณะเป็นการคำนวณที่มีความซับซ้อนหรือจำเป็นต้องใช้เนื้อที่ในการทำงานมากเช่นเราต้องการหาว่าโพลิกอน (polygons) ที่เรากำหนดไปเหลื่อมกับโพลิกอนอื่นๆ หรือไม่ ซึ่งรูทีนลักษณะนี้จะมีผลต่อการทำงานของระบบมากเช่นเรานำไปใช้ในส่วนของการกำหนดการทำงานกับข้อมูลในตารางของ SQL ดังนั้นระบบจะต้องมีการประเมินหรือจัดลำดับของนิพจน์เหล่านี้ให้ดีคือควรจะให้นิพจน์ที่ใช้เวลาหรือเนื้อที่ในการทำงานน้อยก่อน ซึ่งในการกำหนดอาจจะดูจำนวนของแถวที่จะส่งกลับมาหรือปัจจัยอื่นๆ ร่วมกับ ซึ่งในการสร้างรูทีนใน DataBlade โมดูลเราสามารถที่จะใช้สูตรต่อไปนี้ช่วยในการประเมินได้

<line of code>+(<number of I/Os>*100)

ตัวอย่างเช่น ถ้าเรามีรูทีนที่มีขนาด 100 บรรทัดและต้องการใช้ 5 ดิสก์ I/O ในการทำงานเราจะได้ cost ดังนี้ 100+(5*100) หรือเท่ากับ 600 เมื่อเราได้ cost ดังกล่าวแล้วเราก็ต้องพิจารณาคำถามเหล่านี้

- รูทีนของเราใช้เวลามากไปหรือไม่
- รูทีนใช้เนื้อที่ของดิสก์หรือหน่วยความจำมากไปหรือไม่

- มีรูทีนอื่นที่เรียกใช้รูทีนนี้หรือไม่และถ้ามีจะมี cost เท่าไหร่
- มีรูทีนอื่นที่อยู่ใน โมดูลอื่นเรียกใช้รูทีนนี้หรือ ไม่และถ้ามีจะมี cost เท่าไหร่

- **การเรียงข้อมูล (Sorting)**

ถ้าเราต้องการที่จะเรียงข้อมูลของผลลัพธ์ที่เป็นข้อมูลที่เราสร้างขึ้นมาก็จำเป็นที่จะต้องสร้างรูทีนที่ทำหน้าที่ในการเปรียบเทียบข้อมูลชนิดนี้ เพื่อที่จะนำเป็นใช้ช่วยในการทำการเรียงข้อมูล

- **การจัดกลุ่มข้อมูล (Grouping)**

SQL อนุญาตให้เราสามารถที่จะทำการจัดกลุ่มของผลลัพธ์ที่ได้เรียกว่าการทำ grouping ซึ่งใน DataBlade โมดูลที่มีการกำหนดชนิดข้อมูลต่างๆ ขึ้นมาเราจะต้องพิจารณาว่าเราต้องการที่จะทำการจัดกลุ่มค่าของข้อมูลชนิดนี้หรือไม่ และถ้าเราต้องการเราจะใช้หลักเกณฑ์หรือจะใช้อะไรในการจัดกลุ่มค่าข้อมูลเพื่อที่จะได้ตรงตามความต้องการของผู้ใช้

- **การแปลงชนิดข้อมูล (Casts)**

ถ้า DataBlade โมดูลเรามีการเปรียบเทียบข้อมูลต่างชนิดกันเราจำเป็นต้องมีการกำหนดฟังก์ชันที่มีหน้าทีในแปลงชนิดของข้อมูลจากข้อมูลชนิดหนึ่งไปเป็นข้อมูลอีกชนิดหนึ่ง โดยที่รูทีนในการทำการแปลงชนิดข้อมูลนี้จะถูกใช้โดยตัวดำเนินการเกี่ยวกับคิวรี่ (query processing engine) ทั้งในลักษณะที่ทำการแปลงชนิดข้อมูลให้อัตโนมัติและแบบที่จะต้องระบุให้มีการแปลงชนิดข้อมูล (implicitly และ explicitly)

ซึ่งในลำดับชั้นของการสืบทอดคุณสมบัติ (Inheritance hierarchy) โดยทั่วไปแล้วซับคลาสจะสามารถที่จะทำการแปลงชนิดข้อมูลให้อัตโนมัติไปเป็นข้อมูลชนิดเดียวกับซูเปอร์คลาสได้แต่ในทางกลับกับเราสามารถที่จะทำเช่นนี้ได้เพราะซับคลาสได้มีการเพิ่มตัวแปรแบบอินสแตนซ์ (instance variable) ที่ไม่มีในตัวแม่ (parent) หรือในข้อมูลชนิด disinct ก็สามารถที่จะแปลงชนิดข้อมูลไปเป็นชนิดข้อมูลของข้อมูลที่เป็นต้นแบบได้

- **การเลือกเส้นทางเข้าถึงข้อมูล (Access Path Selection)**

ในการเลือกเส้นทางเข้าถึงข้อมูลที่ดีที่สุด ตัวเซิร์ฟเวอร์จะประเมินจาก Cost ต่างๆ เช่นจำนวนของดิสก์ I/O,จำนวนของ row return และ cost ต่างๆที่แต่ละรูทีนใช้ในการทำงานเป็นต้น

- *Unordered Row Processing*

เมื่อเราทำการออกแบบ DataBlade โมดูลเราไม่สามารถที่จะควบคุมการเรียกใช้งาน (execute) ได้ หมายถึงเราไม่สามารถที่จะรับประกันได้ว่าจะมีการเรียกรูทีนในคิวรี่ที่ไหนหรือมีลำดับที่เท่าไหร่ ดังนั้นเราต้องแน่ใจว่ารูทีนของเราไม่มีลักษณะการทำงานที่ต้องการค่าจากทำงานของส่วนอื่นก่อนหรือต้องมีลำดับที่แน่นอนถ้ารูทีนของเรามีลักษณะการทำงานแบบนี้เราจะต้องแก้ไขการทำงานเสียใหม่

- *Secondary Access Method*

การกำหนดวิธีการเข้าถึงข้อมูลลำดับที่สอง (Secondary access method) ใช้ในคิวรี่จะทำให้การทำงานของคิวรี่ที่ประสิทธิภาพมากขึ้นซึ่งในการสร้าง BataBlade โมดูลเราสามารถที่จะกำหนด index บนข้อมูลที่สร้างขึ้นได้ โดยในการกำหนดวิธีการทำอินเด็กซ์ว่าจะใช้ R-tree หรือ B-tree ก็ขึ้นอยู่กับลักษณะของชนิดข้อมูลว่าเป็นอย่างไร

- **การวางแผนสำหรับลักษณะของทรานเซคชั่น (Planning for Transaction Semantics)**

การสร้าง DataBlade โมดูลเราต้องคำนึงในเรื่องการทำงานของทรานเซคชั่นทั้งในเรื่องของคุณสมบัติของทรานเซคชั่นและในเรื่องของการทำงานที่มีลักษณะแบบคอนเคอเรนซี่ด้วยดังนั้นในการออกแบบเราต้องพิจารณาว่า

- จะมีอะไรเกิดขึ้นถ้ามีผู้ใช้มากกว่า 1 คนทำการเรียกใช้รูทีนเดียวกันบนตารางเดียวกันพร้อมกัน
- สามารถที่จะใช้ผลลัพธ์จากบัพเฟอร์ (buffer) มาใช้ได้อีกหรือไม่
- ถ้ามีคนใช้งานก่อนหน้านี้แล้วผลที่ได้จะคงอยู่หรือไม่

6.1.5 การใช้การดำเนินการที่มีอยู่แล้ว (Interoperability)

เป็นการพิจารณาว่า DataBlade โมดูลที่จะสร้างใหม่นี้จะสามารถใช้ข้อมูลที่กำหนดหรือสร้างไว้ (Universal data option) และสามารถที่จะใช้ DataBlade โมดูลเก่าดีหรือไม่ ซึ่งในหัวข้อนี้เราจะพิจารณาในส่วนของ

- Orthogonality
- Simply,Clean interface

Orthogonality

ในระบบที่เป็นแบบ Orthogonal เช่นระบบฐานข้อมูลแบบ ORDBMS ปกติมักจะประกอบด้วยการทำงานของหลายๆ ส่วนร่วมกัน เช่น DataBlade โมดูลแต่ละตัวก็จะนำเสนอการแก้ปัญหาเฉพาะอย่างเท่านั้นซึ่งหมายความว่าในส่วนที่อยู่นอกเหนือขอบเขตของปัญหาที่ต้องการใช้ใน DataBlade โมดูลนั้นก็จะใช้ DataBlade โมดูลอื่นๆ ที่แก้ปัญหาดังกล่าวมาช่วยในการทำงานของ DataBlade โมดูลนี้

Simply, Clean interface

ในการนำเสนอการติดต่อกับผู้ใช้ในลักษณะที่ง่ายหรือมีความชัดเจนนั้นเราสามารถทำได้โดยยึดหลักดังนี้คือ

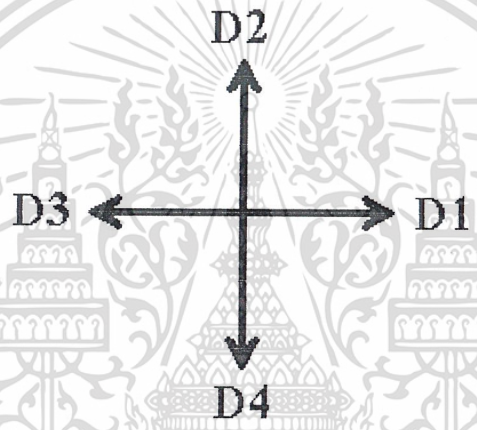
- กำหนดชื่อของรูทีนให้สื่อความหมายในตัวเอง (self-documenting)
- ใช้ข้อดีของการทำโพลิมอร์ฟิซึม (การที่มีหลายๆรูทีนที่มีชื่อเหมือนกันแต่รับอากิวเมนท์ต่างชนิดกัน)
- มีการกำหนดจำนวนของอากิวเมนท์ในแต่ละรูทีนให้มีจำนวนจำกัด เพื่อที่ผู้ใช้จำได้ง่ายว่าแต่ละรูทีนมีจำนวนอากิวเมนท์เท่าไหร่

- หลีกเลี่ยงการสร้างรูทีนที่มีลักษณะรับอากิวเมนท์ไปควบคุมการทำงานภายในของรูทีนเช่น มีการรับอากิวเมนท์หนึ่งถ้ามีค่าเป็น 0 ทำงานลักษณะหนึ่งค่าเป็น 1 ทำงานอีกลักษณะหนึ่ง เป็นต้น ซึ่งอากิวเมนท์ที่ว่านี้ไม่รวมถึงค่าข้อมูลที่ต้องการนำเป็นใช้งานภายในรูทีนซึ่งรูทีนลักษณะเช่นนี้จะทำใช้ผู้ใช้ไหม่งงได้

บทที่ 7

PQ Code

ขั้นตอนในการทำให้เครื่องคอมพิวเตอร์เกิดการเรียนรู้จำตัวอักษรนั้น อยู่บนพื้นฐานของทฤษฎีของการรู้จำตัวอักษรโดยการวิเคราะห์โครงสร้างของตัวอักษร ด้วยวิธีการ Feature Concentration ซึ่งมีรายละเอียดและสามารถแย่งออกได้เป็นขั้นตอนสำคัญๆได้ 3 ขั้นตอนคือ การสร้างรหัสเบื้องต้น (Initial Feature Extraction) การปรับรหัสพื้นตัวอักษร (Unification) การสร้างรหัสรวม (Concentration) โดยในแต่ละขั้นตอนต้องทำการพิจารณทุกๆจุดภาพตามการพิจารณาในแต่ละขั้นตอนโดยการพิจารณาตามทิศทางที่กำหนดไว้ดังแสดงในรูปที่ 7-1



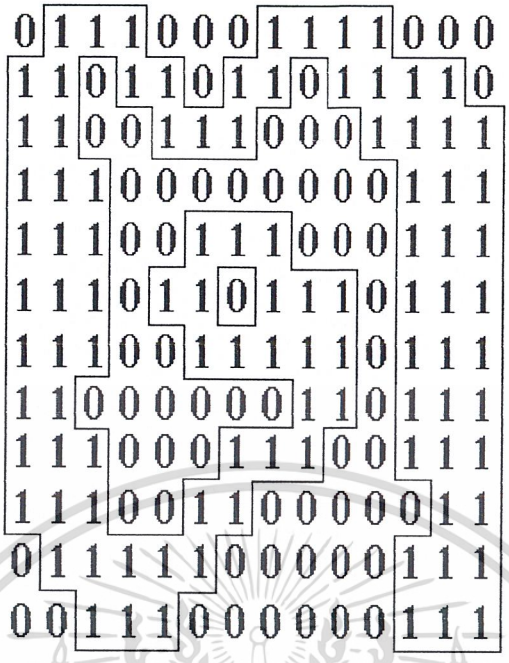
รูปที่ 7-1 การแสดงทิศทางในการเข้ารหัสข้อมูลทั้ง 4 ทิศทาง

7.1 การสร้างรหัสเบื้องต้น (Initial Feature Extraction)

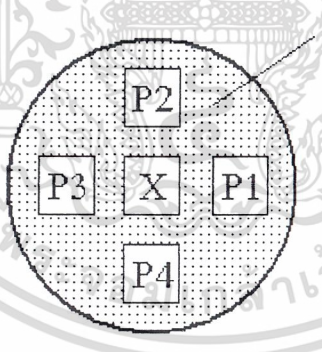
ในส่วนของข้อมูลภาพของตัวอักษรที่ได้ จากแฟ้มข้อมูลของเครื่องตรวจกวาดภาพ (Image Scanner) ซึ่งมีลักษณะข้อมูลเป็นไปตามเงื่อนไขและชนิดของการเก็บภาพ ดังนั้นจึงต้องมีกระบวนการในการเตรียมข้อมูลก่อนการประมวลผล (Pre-processing) เพื่อการเปลี่ยนแปลงข้อมูลจากแฟ้มข้อมูลเดิมให้อยู่ในรูปรหัสที่สามารถนำไปใช้ประมวลผลต่อได้โดยในเนื้อของตัวอักษรจะแทนด้วย 1 และในส่วนของพื้นตัวอักษรจะแทนด้วย 0 ดังแสดงในรูปที่ 7-2 แล้วพิจารณาการสร้างรหัสแทน โดยแยกออกเป็น 2 ส่วน คือ

7.1.1 รหัสเนื้อตัวอักษร (P-CODE)

เนื้อตัวอักษรซึ่งถูกแทนด้วย 1 ในรูปที่ 7-2 จะถูกแปลงต่อแทนด้วยรหัส P โดยการพิจารณาเฉพาะจุดที่แทนด้วยการที่เป็น 1 ตามวิธีการดังแสดงในรูปที่ 7-3

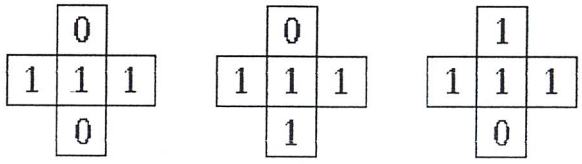


รูปที่ 7-2 แสดงตัวอย่างข้อมูล 1 ตัวอักษร

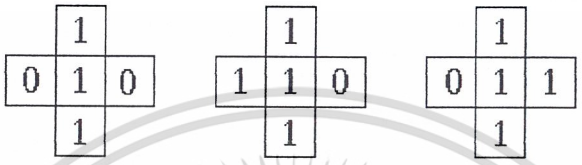


***รูปที่ 7-3** Initial Feature Extraction โดยการพิจารณาเนื้อตัวอักษร*

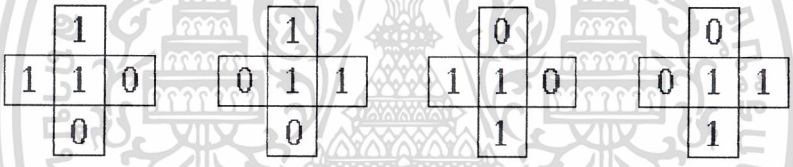
ถ้าให้จุด X เป็นจุดที่แทนด้วยรหัส 1 และจุดอื่นๆที่อยู่ติดกับจุด X ที่กำลังสนใจเป็น P1, P2, P3, P4 ซึ่งอยู่โดยรอบในทิศทางทั้ง 4 ลำดับ โดยเราสนใจกรณีที่ X ใดๆเป็นรหัส 1 เสมอจะพบว่ามีโอกาสเป็นไปได้ทั้งหมดหลายกรณีด้วยกันและสามารถแบ่งออกตามลักษณะการวางตัวของแนวเส้น (แนวทางเดินของรหัส 1) ได้เป็น 4 ลักษณะ ดังแสดงในรูปที่ 7-4



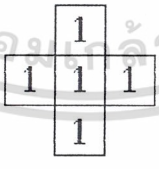
รูปที่ 7-4 (ก) แสดงการวางตัวในแนวนอน



รูปที่ 7-4 (ข) แสดงการวางตัวในแนวตั้ง



รูปที่ 7-4 (ค) แสดงการวางตัวในแนวทแยง



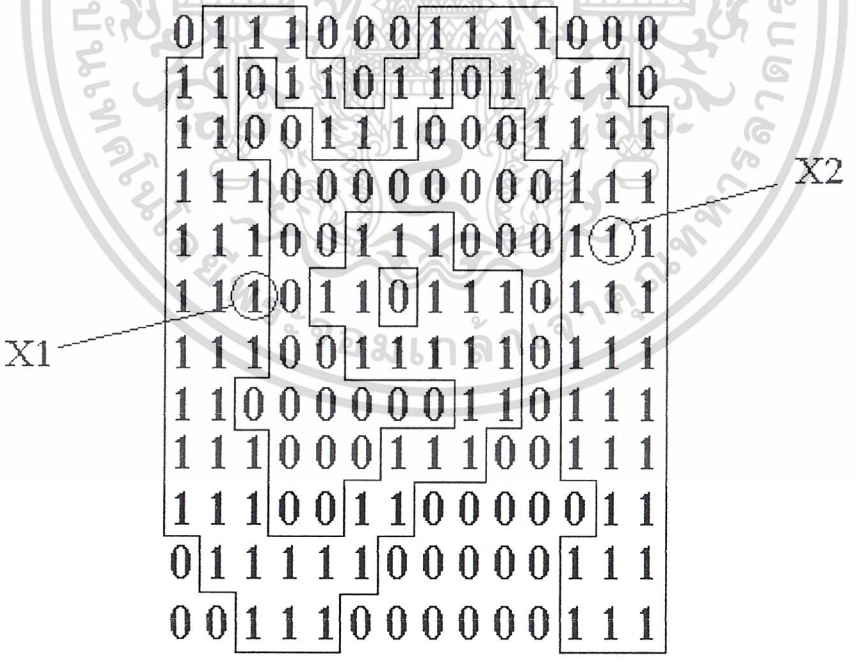
***รูปที่** 7-4 (ง) แสดงการวางตัวในลักษณะถูกล้อมรอบด้วย 1*

จากรูปที่ 7-4 จะเห็นได้ว่าแนวทางการวางตัวของรหัส 1 นั้นมีด้วยกัน 4 กรณีซึ่ง สามารถเขียนอธิบายในรูปของคณิตศาสตร์ได้ทั้ง 4 รูปแบบและได้กำหนดรหัสแทนการวางตัวของตัวของรหัส 1 ออกเป็น 4 ลักษณะคือ ถ้าวางตัวในแนวนอนแทนด้วย H ถ้าวางตัวในแนวตั้งแทนด้วย V ถ้าวางตัวในแนวทแยงใดๆให้แทนด้วย S และถ้าวางตัวในลักษณะที่ถูกล้อมรอบด้วย 1 ให้แทนด้วย I ซึ่งสามารถเขียนแสดงได้ในตารางที่ 7-1

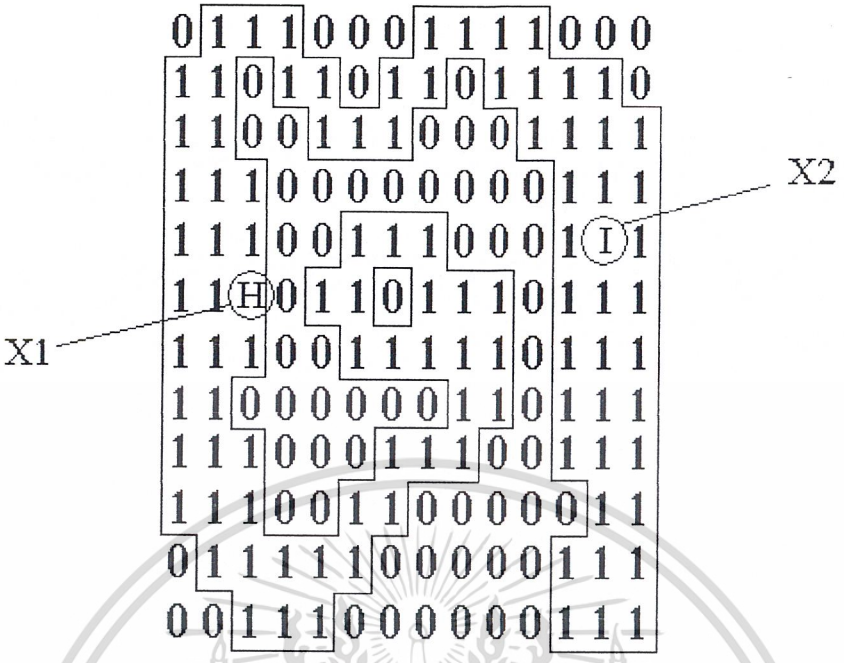
เงื่อนไข	รหัสแทน	ลักษณะเส้น
((1-P2*P4)*P1*P3)=1	H	วางตัวในแนวนอน
((1-P1*P3)*P2*P4)=1	V	วางตัวในแนวตั้ง
((1-P2*P4)*(1-P1*P3))=1	S	วางตัวในแนวทแยง
(P1*P2*P3*P4)=1	I	ถูกล้อมรอบภายใน

ตารางที่ 7-1 การกำหนดค่ารหัส P

จากรูปที่ 7-2 ถ้าเราแยกการพิจารณาโดยใช้เงื่อนไขตามตารางที่ 7-1 จะสามารถเขียนแสดงในรูปที่ 7-5(ก) โดยหากพิจารณาที่ตำแหน่ง X1 จะได้ค่าของของจุดที่อยู่ติดกับ X1 เป็น P1=0, P2=1, P3=1 และ P4=1 ตามลำดับซึ่งเมื่อนำมาคำนวณเพื่อหารหัสแทนตามในตารางที่ 7-1 จะสอดคล้องกับในบรรทัดที่ 1 โดยเมื่อแทนค่าแล้วจะทำให้สมการที่ใช้แทนรหัส H ได้เป็นจริงเพียงสมการเดียวจึงแทนรหัสที่ตำแหน่ง X1 ด้วยรหัส H และในทำนองเดียวกันเมื่อพิจารณาที่ตำแหน่ง X2 จะได้ค่าของจุดที่อยู่ติดกับจุด X2 เป็น P1=1, P2=1, P3=1 และ P4=1 ตามลำดับซึ่งเมื่อนำมาคำนวณเพื่อหารหัสแทนตามตารางที่ 7-1 จะสอดคล้องกับบรรทัดที่ 4 โดยเมื่อแทนค่าแล้วจะทำให้สมการที่แทนด้วยรหัส I เป็นจริงเพียงสมการเดียวนั้นจึงแทนรหัสที่ตำแหน่ง X2 ด้วยรหัส I



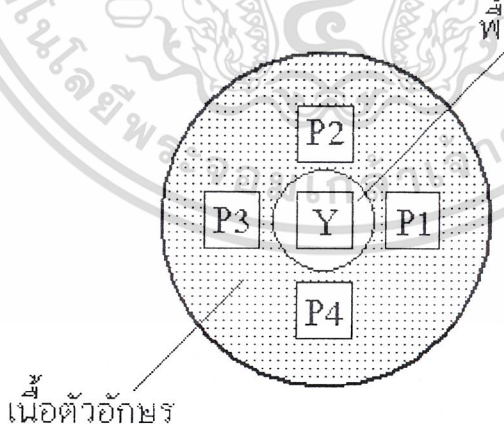
รูปที่ 7-5 (ก) แสดงการพิจารณารหัสเบื้องต้นจากเนื้อตัวอักษร



รูปที่ 7-5 (ข) แสดงการแทนรหัสเบื้องต้นจากเนื้อตัวอักษร

7.1.2 รหัสพื้นตัวอักษร (Q-code)

พื้นตัวอักษรซึ่งถูกแทนด้วย 0 ในรูปที่ 7-2 จะถูกแปลงด้วยการเข้ารหัส Q โดยการพิจารณาเฉพาะจุดที่เป็นรหัส 0 ตามวิธีการดังในรูปที่ 7-6



รูปที่ 7-6 Initial Feature Extraction โดยการพิจารณาพื้นตัวอักษร

ถ้าให้จุดที่ Y เป็นรหัสที่แทนด้วย 0 ซึ่งเป็นส่วนของพื้นตัวอักษรที่เรากำลังพิจารณาที่จะแปลงรหัสและพิจารณาลักษณะการวางตัวของจุด Y ว่าถูกล้อมรอบด้วยเส้นตรงหรือจุดที่เป็นเนื้อของตัวอักษรซึ่งแทนด้วยรหัส 1 หรือไม่ ในทิศทางทั้ง 4 จากจุดที่กำลังพิจารณา ซึ่งกำหนดให้เป็น P1, P2, P3, P4 ตาม

ลำดับ ถ้าพบรหัสแทนที่เป็น 1 ด้านใดก็ให้แทนค่า P ในทิศทางนั้นด้วย 1 ถ้าหากไม่พบรหัสที่เป็น 1 ก็ให้แทนค่าของ P ในทิศทางนั้นด้วย 0 โดยพิจารณาทั้ง 4 ทิศทางตามลำดับจะพบว่ามีโอกาสเป็นไปได้ที่จะมีการวางตัวของรหัสที่เป็น 1 จากทิศทางทั้ง 4 ด้วยกัน 16 กรณีซึ่งสามารถเขียนแสดงในตารางที่ 7-2 และแทนลักษณะการวางตัวในลักษณะต่างๆด้วยรหัสแทน Q0, Q1, Q2, ... QF ตามลำดับ

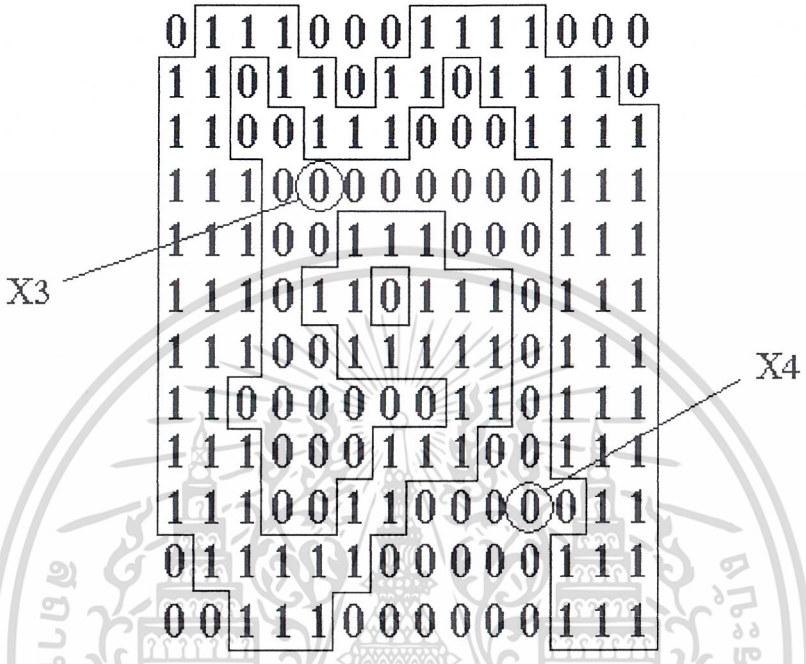
รหัส	ลักษณะแทน	รหัส	ลักษณะแทน
Q0	▪	Q8	▪
Q1	▪	Q9	▪
Q2	▪	QA	▪
Q3	▪	QB	▪
Q4	▪	QC	▪
Q5	▪	QD	▪
Q6	▪	QE	▪
Q7	▪	QF	▪

ตารางที่ 7-2 การกำหนดค่ารหัส Q

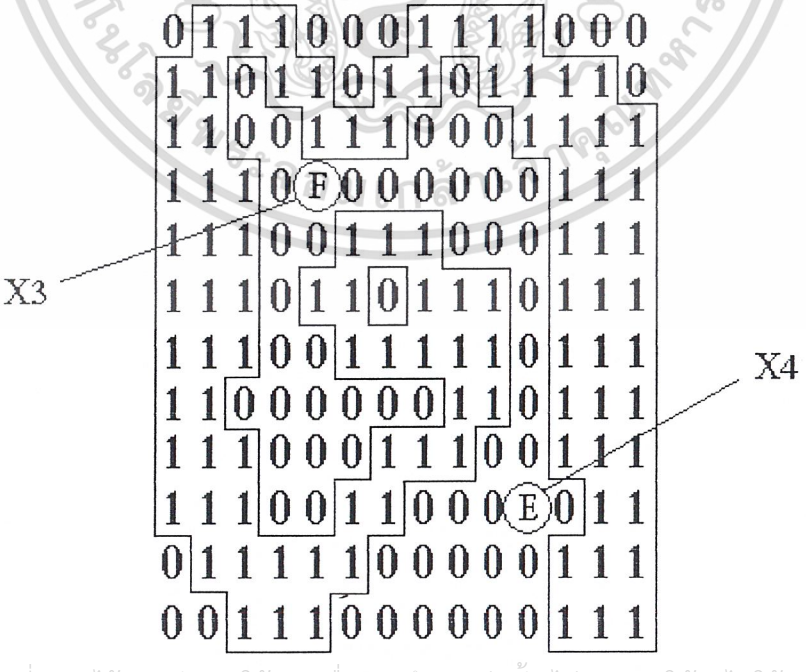
จากตารางที่ 7-2 จะเห็นได้ว่าโอกาสในการเกิดตามเงื่อนไขการพิจารณาในส่วนที่เป็นพื้นตัวอักษรจะเป็นไปได้ด้วยกัน 16 กรณี ซึ่งสามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์ โดยการกำหนดค่าถ่วงน้ำหนัก (Weight) ให้แต่ละแนวทางของการพิจารณาโดยให้มีแนวทางด้านล่างมีน้ำหนักเป็น 1 แนวทางด้านซ้ายมีน้ำหนักเป็น 2 แนวทางทางด้านบนมีน้ำหนักเป็น 4 แนวทางด้านขวามีน้ำหนักเป็น 8 ของจุดพื้นตัวอักษรที่มีรหัสแทนเป็น 0 ซึ่งเป็นรหัสของพื้นตัวอักษรที่กำลังพิจารณาแล้วนำค่าที่ได้จาการพิจารณาการวางตัวของจุดพื้นตัวอักษรที่ใช้แทนในการพิจารณาการสร้างรหัสเบื้องต้นด้วยการพิจารณาพื้นตัวอักษรจะได้ค่าเป็นไปได้ทั้งหมด 16 กรณี ซึ่งสามารถเขียนแสดงได้ด้วยเลขฐาน 16 ตั้งแต่ 0, 1, 2, ... F แต่จากการทดลองเมื่อผ่านข้อมูลในกระบวนการแยกแยะตัวอักษรแล้ว จะทำให้รหัสพื้นตัวอักษรที่ได้มีเพียงแค่ 9 กรณีเท่านั้น ยกเว้นกรณีของ 0, 1, 2, 4 และ 8 ด้วยข้อมูลที่ผ่านขั้นตอนของการแยกแยะตัวอักษรแล้วเท่านั้นจะต้องตัวแสดงส่วนที่เป็นเนื้อตัวอักษรติดด้านทั้ง 4 ของขอบตัวอักษรภาพแต่ละตัวดังแสดงในรูป

ที่ 7-2 รหัสพื้นของตัวอักษรจึงไม่เกิดกรณีที่ถูกล้อมรอบด้วยเส้นตรงหรือจุดที่เป็นรหัส 1 เพียงด้านเดียว หรือไม่ถูกล้อมรอบด้วยจุดที่เป็นรหัสเป็น 1 เลย ซึ่งสามารถแสดงการหารหัสพื้นตัวอักษรได้แสดงในรูป 5-7(ก) และ (ข)

K = {(8*P1)+(4*P2)+(2*P3)+(1*P4)}.....5-1



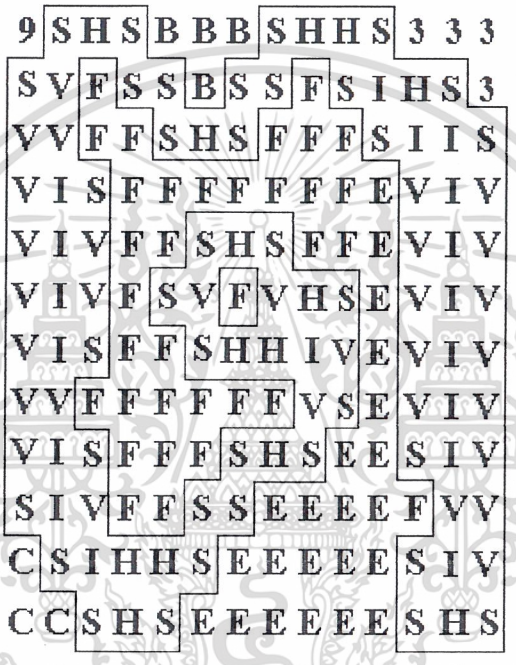
รูปที่ 7-7 (ก) แสดงการพิจารณารหัสเบื้องต้นจากพื้นตัวอักษร



รูปที่ 7-7 (ข) แสดงการแทนรหัสเบื้องต้นจากพื้นตัวอักษร

จากรูปหากพิจารณาที่จุด X3 และพิจารณาตามแนวทางทั้ง 4 เพื่อหาการถูกล้อมรอบด้วยเส้นที่มีรหัสเป็น 1 จะได้ค่าของ P1, P2, P3 และ P4 มีค่าเป็น 1, 1, 1 และ 1 ตามลำดับ และนำมาแทนค่าในสมการที่ 5-1 จะได้ค่าของ k=15 ซึ่งแทนด้วยรหัส F จะได้รหัสที่แทนในตำแหน่ง X3 นั้นด้วย F และในทำนองเดียวกันหากพิจารณาที่จุด X4 จะได้ค่าของ P1, P2, P3, P4 มีค่าเป็น 1, 1, 1, 0 ตามลำดับ และนำมาแทนค่าในสมการที่ 5-1 จะได้ค่า K= 14 ซึ่งแทนด้วยรหัส E จะได้รหัสแทนในตำแหน่ง X4 นั้นด้วย E

จากหัวข้อที่ 7.1.1 และ 7.1.2 เมื่อทำการพิจารณาทุกจุดของภาพทั้งในส่วนที่เป็นพื้นของตัวอักษรและส่วนที่เป็นเนื้อของตัวอักษรหรือเมื่อแทนค่าของรหัส P และรหัส Q ของทุกจุดของภาพแล้วจะสามารถแสดงดังในรูปที่ 7-8

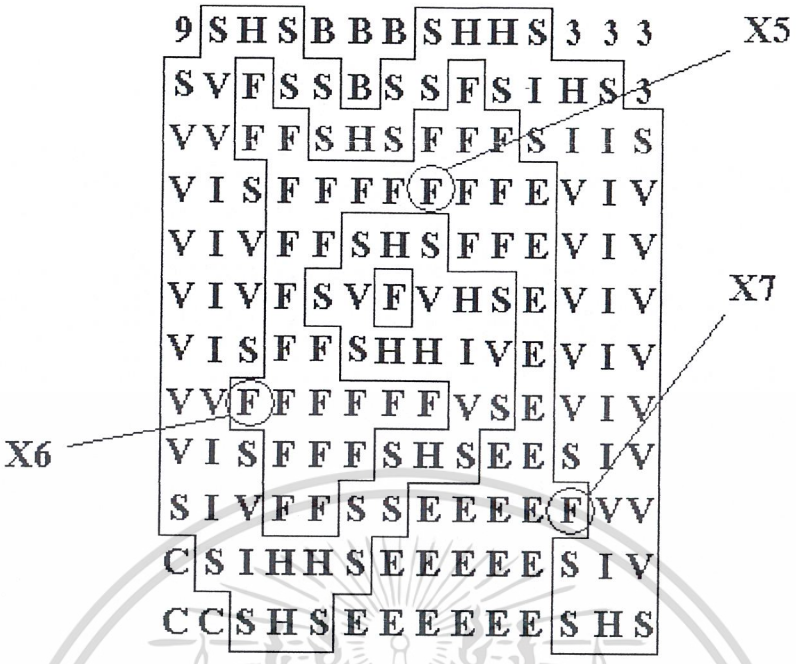


รูปที่ 7-8 แสดงข้อมูลที่ผ่านขั้นตอนการกำหนดรหัสเบื้องต้น

7.2 การปรับรหัสพื้นตัวอักษร (Unification)

จากขั้นตอนของการสร้างรหัสเบื้องต้นเป็นการพิจารณาแทนรหัสที่เป็น นื้อตัวอักษรและส่วนที่เป็นพื้นตัวอักษรแต่ปัญหาที่เกิดจากการทดลองนั้น ลักษณะของข้อมูลที่ได้จากกระบวนการเตรียมข้อมูลอยู่ในลักษณะที่ไม่สมบูรณ์และมีลักษณะที่เว้า แหว่ง ไม่แสดงลักษณะโดยรวมของตัวอักษรบริเวณนั้นๆ ดังแสดงในรูปที่ 7-9 จากรูปเมื่อพิจารณาจุด X5 จะเป็นได้ว่ารหัสที่คือ F แต่ลักษณะที่แสดงไม่ได้ถูกล้อมด้วยจุดที่มีรหัสเป็น 1 ในตำแหน่ง X5 จึงไม่แสดงลักษณะที่ถูกต้อง ซึ่งควรจะเป็นในลักษณะที่ถูกปิดด้วยเส้นที่มีรหัสเป็น 1 เพียงแค่ 3 ด้านเท่านั้น ในแบบรหัส E หากพิจารณาที่จุด X6, X7 ซึ่งทั้งสองจุดนี้แสดงลักษณะที่ไม่ถูกต้อง สาเหตุอาจเกิดมาจากความผิดพลาดจากกระบวนการเตรียมข้อมูล

จากรูปลักษณะดังกล่าวจึงต้องมีการเพิ่มกระบวนการในการปรับส่วนพื้นตัวอักษร(ส่วนที่เป็นรหัส Q) เพื่อแสดงลักษณะของโครงสร้างโดยรวมของบริเวณเดียวกันและปรับรหัสพื้นตัวอักษรที่เกิดจากข้อมูลไม่สมบูรณ์ (ผิดพลาดจากการเตรียมข้อมูล) ซึ่งอาศัยคุณสมบัติทางเรขาคณิตด้วยการนำค่ารหัส Q ที่



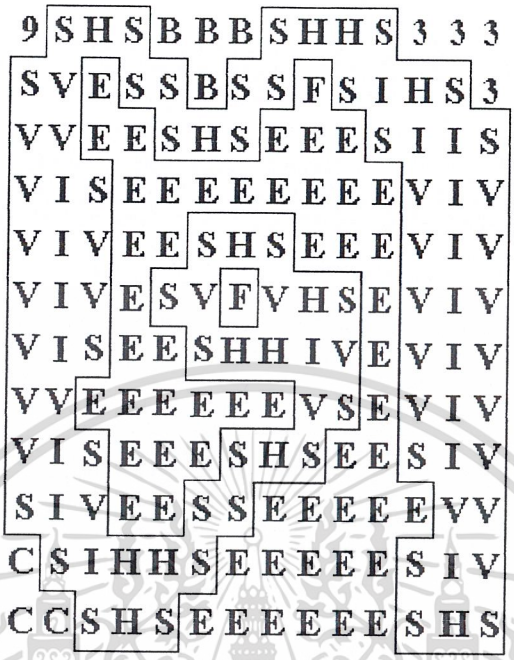
รูปที่ 7-9 แสดงข้อมูลที่มีลักษณะไม่สมบูรณ์

อยู่ติดกันมาพิจารณาปรับรหัสตามกฎเกณฑ์ของการปรับรหัส Q ในตารางที่ 7-3 โดยการพิจารณาจุดที่สนใจให้อยู่ในตารางที่แถวที่ 1 และให้จุดรอบข้างทั้ง 4 ทิศทางมีลักษณะในตารางแถวที่ 2 ซึ่งถ้าตรงกับลักษณะใดลักษณะหนึ่งก็ให้เปลี่ยนเป็นรหัสในตารางแถวที่ 1 เป็นรหัสในตารางแถวที่ 3 แทน

รหัสแถวที่ 1	รหัสแถวที่ 2	รหัสแถวที่ 3
F	C	C
F	9	9
F	E	E
F	D	D
F	B	B
F	7	7
F	5	5
E	C	C
D	C	C
D	9	9
B	9	9
7	6	6

ตารางที่ 7-3 กฎการปรับรหัสพื้นตัวอักษร

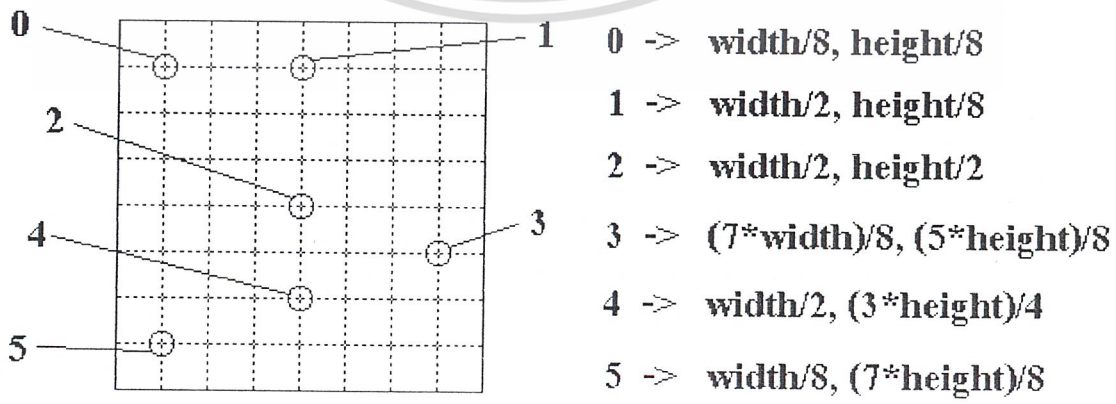
จากรูปที่ 7-8 เมื่อทำการปรับรหัสพื้นตัวอักษรตามเงื่อนไขจะได้ผลแสดงในรูปที่ 7-10



รูปที่ 7-10 แสดงข้อมูลที่ผ่านขั้นตอนการปรับรหัสพื้นตัวอักษร

7.3 การสร้างรหัสรวม (Concentration Code)

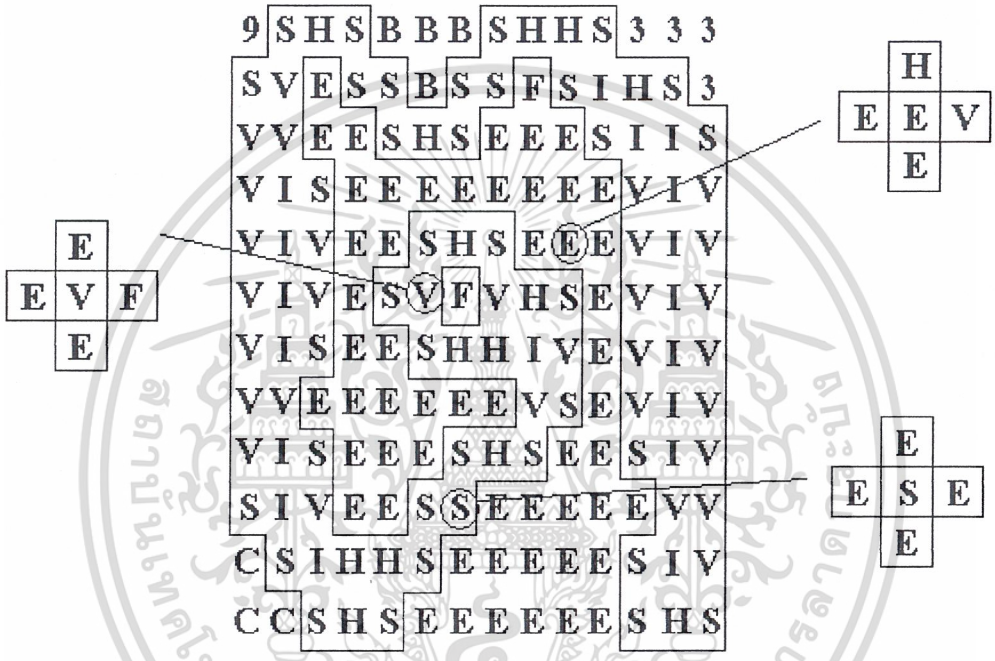
เมื่อผ่านกระบวนการในการพิจารณารหัสแทนส่วนที่เป็นเนื้อตัวอักษร และส่วนที่เป็นพื้นตัวอักษรแล้วการปรับรหัสส่วนที่เป็นพื้นตัวอักษรกระบวนการต่อไป เป็นกระบวนการนำเอารหัสเบื้องต้นที่ได้ไปพิจารณาเพื่อเก็บเป็นลักษณะเด่นของแต่ละตัวอักษรเพื่อใช้แทนตัวอักษรนั้นๆ ในการนำเอารหัสเบื้องต้นไปอธิบายอาจทำได้โดยการใช้ส่วนที่เป็นเฉพาะรหัสที่เป็นเนื้อตัวอักษรนั้นๆ หรือ เฉพาะส่วนที่เป็นรหัสพื้นส่วนใดส่วนหนึ่งใช้อธิบายลักษณะเด่นของแต่ละตัวอักษร ซึ่งจะเรียกว่ารหัสรวม



รูปที่ 7-11 แสดงตำแหน่งที่เลือกเพื่อเก็บค่ารหัสรวม

สำหรับโปรแกรมตัวอย่างที่ได้ลองทำในโครงงานนี้ได้เลือกที่จะเก็บรหัสรวมทั้งหมด 6 จุดโดยที่เลือกตำแหน่งตามรูปที่แสดงในรูปที่ 7-11 คือจะเก็บจุด 0, 1, 2, 3, 4, 5 ซึ่งได้แสดงวิธีการคำนวณตำแหน่งของจุดเหล่าต่างๆไว้ในภาพเดียวกันนั้นแล้ว ซึ่งค่าที่คำนวณก็คือตำแหน่ง (X, Y) ที่ต้องการเก็บค่ารหัสรวมสำหรับแต่ละจุดที่ถูกเลือกนั้น ค่ารหัสรวมจะเก็บดังในรูปที่ 7-12 คือจะเก็บค่าทั้งหมด 5 ค่าคือ Q0, Q1, Q2, Q3, Q4 โดยที่มาของแต่ละค่าอธิบายได้ดังนี้

ค่า Q0 คือค่ารหัส ณ ตำแหน่งของจุดที่ต้องการเก็บค่ารหัสรวมซึ่งอาจจะเป็นรหัสในส่วนของเนื้อตัวอักษร หรืออาจเป็นส่วนของพื้นของตัวอักษรก็ได้

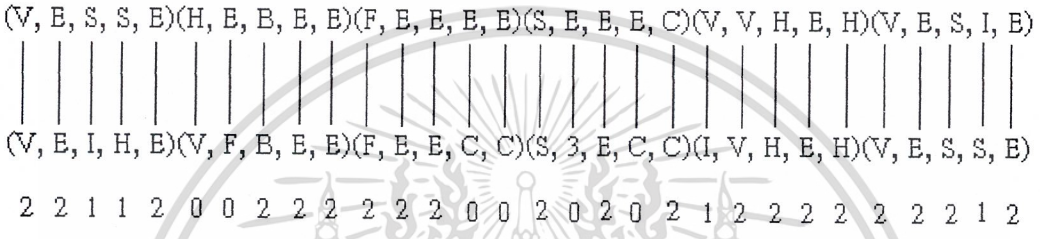


รูปที่ 7-12 แสดงค่าของรหัสรวม ณ ตำแหน่งต่างๆ

ค่า Q1, Q2, Q3, Q4 สามารถหาได้โดยการเริ่มต้นพิจารณาข้อมูลจากตำแหน่งที่ต้องการเก็บรหัสรวมแล้วค่อยๆย้ายตำแหน่งพิจารณาข้อมูลห่างออกไปจากจุดตำแหน่งเริ่มต้น โดยห่างไปทางทิศใดนั้นขึ้นกับว่าเป็นจุด Q ไหน ถ้าเป็น Q1 จะทำการพิจารณารหัสไปทางด้านขวา ถ้าเป็น Q2 จะทำการพิจารณาไปทางด้านบน ถ้าเป็น Q3 จะพิจารณาไปทางด้านซ้าย ส่วน Q4 จะพิจารณาไปทางด้านล่าง เมื่อทำการพิจารณาข้อมูลไปจนเจอรหัสที่เป็นส่วนเนื้อของตัวอักษร เมื่อไหร่ก็หมายความว่าหลังจากตำแหน่งนั้นเป็นต้นไปถ้าเจอรหัสที่เป็นส่วนของพื้นตัวอักษรก็จะนำค่านั้นเป็นค่า Q นั้นๆทันที แต่ถ้าในกรณีที่พบรหัสที่เป็นส่วนของตัวอักษรแล้วการพิจารณาข้อมูลไปถึงยังขอบของภาพเสียก่อนก็ให้เก็บรหัสของเนื้อตัวอักษร ณ ตำแหน่งนั้นๆเป็นค่า Q ได้เลย ส่วนกรณีที่ไม่พบรหัสที่เป็นเนื้อตัวอักษรเลยจนกระทั่งถึงขอบภาพ ก็จะเก็บรหัส X ให้เป็นค่า Q แทน

7.4 การให้คะแนนความเหมือนระหว่างสองข้อมูลลักษะเด่น

เมื่อมีชุดข้อมูลที่ใช้อธิบายลักษณะเด่นอยู่สองชุด สามารถเปรียบเทียบได้โดยการพิจารณาแต่ละรหัสว่าเหมือนกันหรือไม่ โดยมีหลักการง่ายๆว่าถ้าสองข้อมูลที่ทำการเปรียบเทียบนั้นมีรหัสเหมือนกันเลยก็จะให้คะแนนสำหรับรหันนั้นสองคะแนน สำหรับในกรณีที่เป็นการเปรียบเทียงระหว่างรหัส S เปรียบเทียบกับ I , V เปรียบเทียบกับ I หรือ H เปรียบเทียบกับ I ก็จะให้ 1 คะแนนสำหรับรหัสนั้น แล้วเมื่อทำการเปรียบเทียบกันครบทุกรหัสแล้วก็รวมคะแนนก็จะได้ผลลัพธ์ก็คือคะแนนความเหมือนของสองค่าลักษณะเด่นนั่นเองดังรูปที่ 7-13



คะแนนความคล้าย = 44

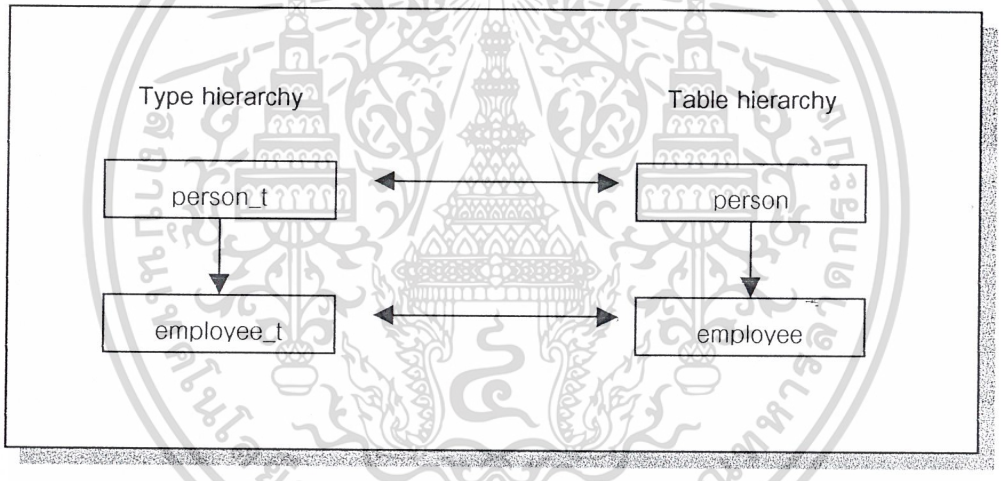
รูปที่ 7-13 แสดงการให้คะแนนความเหมือนระหว่างสองข้อมูลลักษณะเด่น

บทที่ 8
โปรแกรมสาธิตที่ 1 ทดสอบคุณสมบัติทางออบเจ็กต์

ในบทนี้จะเป็นการทดลองคุณสมบัติต่างๆ ของระบบฐานข้อมูลของอินฟอร์มิกซ์ที่เป็นระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ในเรื่องหลักการของออบเจ็กต์อันจะประกอบไปด้วยคุณสมบัติการสืบทอด การทำโพลิมอร์ฟิซึม การสร้างฟังก์ชันสำหรับออบเจ็กต์หรือที่เรียกว่าเมธทอดและการซ่อนข้อมูลภายในออบเจ็กต์โดยจะมีเนื้อหาดังนี้

8.1 ชนิดข้อมูลและตารางที่ใช้สำหรับการทดลอง

สำหรับโมเดลที่ได้มีการออกแบบไว้สำหรับทดสอบคุณสมบัติทางออบเจ็คประกอบด้วยชนิดของข้อมูลดังนี้



รูป 8.1 แสดงชนิดข้อมูลและตารางที่ใช้ในการทดสอบ

จากรูปที่ 8-1 แสดงให้เห็นว่าได้มีการใช้ชนิดข้อมูลเป็นแบบ name row type อันประกอบด้วย person_t และ employee_t โดยชนิดข้อมูล employee_t จะทำการสืบทอด (inherit) มาจากชนิดข้อมูลแบบ person_t นอกจากนี้ในรูปยังได้แสดงให้เห็นถึงตารางที่มีการใช้ชนิดข้อมูลจากลำดับชั้นของชนิดข้อมูล ซึ่งก็คือตาราง person และตาราง employee ที่มีการใช้ชนิดข้อมูลแบบ person_t และ employee_t ตามลำดับ ส่วนคำสั่ง SQL ที่ใช้สร้างชนิดข้อมูลและตารางในรูปสามารถที่จะแสดงได้ดังนี้

```
CREATE ROW TYPE person_t
(
    id          integer  not null ,
    name        varchar(30),
```

```
  age          integer
);
CREATE ROW TYPE employee_t
(
 salary        money,
 location      point,
 bonus         multiset(money not null)
) UNDER person_t;

CREATE TABLE person OF TYPE person_t
(PRIMARY KEY (id));

CREATE TABLE employee OF TYPE employee_t
UNDER person;
```

โดยจากคำสั่ง SQL ดังกล่าวนี้เราได้กำหนดให้ชนิดข้อมูล person_t ประกอบด้วย id ซึ่งในที่นี้ก็คือเลขประจำตัวประชาชน name คือชื่อของคนที่เราต้องการเก็บข้อมูลและ age เป็นอายุของคนคนนั้น ส่วนของ employee_t เป็นชนิดข้อมูลที่สืบทอดมาจาก person_t โดยได้เพิ่มส่วนของ salary ก็คือเงินเดือน location ก็คือตำแหน่งของที่อยู่ของลูกจ้างว่าอยู่ที่ตำแหน่งไหน โดยจะเป็นลักษณะของการเก็บเป็นพิกัด x และ y ซึ่งชนิดข้อมูลนี้เป็นข้อมูลแบบ opaque โดยจะมีเมธทอดที่ใช้หาระยะทางระหว่างจุดพิกัดต่างๆ ซึ่งรายละเอียดจะกล่าวในส่วนของการทำโพลิมอร์ฟีซึมอีกที ส่วนที่ employee_t ได้เพิ่มมาอีกตัวหนึ่งก็คือ bonus ซึ่งเป็นโบนัสที่พนักงานได้ทำเอาไว้โดยมีลักษณะของชนิดข้อมูลเป็นแบบ multiset เพราะเป็นข้อมูลที่สามารถจะมีค่าซ้ำกันได้และไม่มีลำดับ เมื่อเราได้ชนิดข้อมูลต่างๆ เสร็จเราก็ได้ทำการสร้างตารางที่สัมพันธ์กับชนิดข้อมูล person_t และ employee_t ดังคำสั่ง SQLที่แสดงไว้

8.2 ฟังก์ชันและเมธทอดต่างๆ ที่ใช้

สำหรับในการทดสอบในครั้งเราได้กำหนดให้มีฟังก์ชันต่างๆ สำหรับทำงานกับชนิดข้อมูลที่เรากำหนดไว้หรือที่เรียกว่าเมธทอดมีดังนี้

- เมธทอดของ point ประกอบด้วย
  - Distance เป็นเมธทอดที่เขียนด้วยภาษา C มีหน้าที่ในการคำนวณหาระยะทางระหว่างจุดสองจุด ซึ่งดังนั้นเราจะต้องทำการส่งพารามิเตอร์ให้กับเมธทอดนี้สองตัวซึ่งเป็นชนิดแบบ point ซึ่งลักษณะของเมธทอดนี้มีดังนี้

```
mi_double_precision *Distance (
        point *             pnt1,
        point *             pnt2,
        MI_FPARAM *         Gen_fparam      /* Standard info - see DBDK docs.*/
    )
    {
        mi_double_precision*        Gen_RetVal;   /* The return value. */
        MI_CONNECTION *             Gen_Con;      /* The connection handle. */
        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "Distance", ERRORMESG1, 10 );
        }
        /* Write to the trace file indicating that Distance has been called. */
        DBDK_TRACE_ENTER( "Distance" );
        Gen_RetVal = (mi_double_precision *)mi_alloc( sizeof( mi_double_precision ) );
        if( Gen_RetVal == 0){
                DBDK_TRACE_ERROR( "Distance", ERRORMESG2, 10 );
        }
        *Gen_RetVal = sqrt( (pnt1->locx - pnt2->locx) * (pnt1->locx - pnt2->locx) +
                        (pnt1->locy - pnt2->locy) * (pnt1->locy - pnt2->locy) );

        /* Write to the trace file indicating that Distance has successfully exited.*/
        DBDK_TRACE_EXIT( "Distance" );
        return Gen_RetVal;
    }
```

- Distance ซึ่งเป็นเมธทอดที่ทำหน้าที่เดียวกับ distance แบบแรกแต่จะรับพารามิเตอร์เพียงตัวเดียวโดยในการคำนวณหาค่าระยะห่างก็จะคำนวณหาระหว่าง point ที่เราผ่านพารามิเตอร์ไปให้กับค่าที่มีการกำหนดไว้ในเมธทอดนี้อยู่แล้ว ซึ่งเมธทอดตัวนี้จะเขียนโดยใช้ภาษา SPL (ซึ่งค่าที่กำหนดมานี้เราสมมุติว่าเป็นตำแหน่งของบริษัทที่พนักงานคนนั้นทำงานอยู่)

```
CREATE FUNCTION distance(p1 point)
RETURNING FLOAT;
DEFINE p2 point;
 LET p2='10 10';     -- set ค่า default เป็นตำแหน่งของบริษัท
RETURN distance(p1,p2);
END FUNCTION;
```

จากฟังก์ชัน Distance ของ opaque ที่แสดงจะเห็นว่าเป็นการทำฟังก์ชัน Overloading นั้นเอง

- เมธทอดของ person_t ประกอบด้วย
  - getname เป็นเมธทอดที่มีหน้าที่ในการส่งค่าชื่อของ person_t
  - getinfo เป็นเมธทอดที่มีหน้าที่ในการส่งค่าชื่อของ person_t เหมือนกับ getname แต่ที่ได้สร้างขึ้นมาก็เพื่อที่จะนำไปใช้ในการแสดงการทำ Overriding เมธทอด

```
CREATE FUNCTION getname(ps person_t)
RETURNING varchar(30);
        RETURN ps.name;
END FUNCTION;

CREATE FUNCTION getinfo(ps person_t)
RETURNING varchar(30);
        RETURN ps.name;
END FUNCTION;
```

- เมธทอดของ employee_t ประกอบด้วยเมธทอดต่างๆ ที่เป็นของ person_t เพราะ employee_t มีการสืบทอดมาจาก person_t โดย employee_t ได้มีการกำหนดให้มีเมธทอดที่เพิ่มขึ้นดังนี้
  - getinfo เป็นเมธทอดที่ทำการ Override กับเมธทอด getinfo ของ person_t ที่ employee_t ทำการสืบทอดมา

```
CREATE FUNCTION getinfo(ey employee_t)
RETURNING integer;
        RETURN ey.salary;
END FUNCTION;
```

- sumBonus เป็นเมธทอดที่มีหน้าที่ในการคำนวณหาค่าผลรวมทั้งหมดของโบนัสของพนักงานในแต่ละคน

```
CREATE FUNCTION sumBonus(key integer)
RETURNING money;
DEFINE vartexe  multiset (money not null);
DEFINE total,temp  money;
LET  total = 0;
SELECT bonus into vartexe FROM employee
WHERE id=key;
FOREACH cursor1 FOR
        SELECT * INTO temp FROM TABLE(vartexe)
        LET total = total +temp;
END FOREACH
RETURN total;
END FUNCTION;
```

จากโมเดลหรือ Schema ที่ได้กำหนดมานี้เราสามารถที่จะสรุปได้ว่าได้มีการใช้คุณสมบัติต่างๆ ทางออบเจ็คดังนี้

- การสืบทอด (Inheritance)
  - ชนิดข้อมูล ได้มีการสืบทอด row type โดยมี person_t เป็น parent ของ employee_t
  - ตาราง ได้มีการสืบทอดตาราง โดยมีตาราง person เป็น parent ของตาราง employee
- การทำโพลิมอร์ฟีซึม (polymorphism)
  - ทำ Overloading ของฟังก์ชัน Distance ที่เป็นเมธทอดของ point ที่เขียนโดยภาษา C และ SPL
  - ทำ Overriding โดยระหว่างฟังก์ชัน getinfo ของ person_t และ getinfo ของ employee_t

นอกจากนี้ยังมีการใช้ชนิดข้อมูลที่มีลักษณะเป็น Container class ซึ่งก็คือมีการใช้คอเลคชั่น (collection) ที่เป็นแบบ multiset นั้นเอง

8.3 ทดสอบโมเดลข้อมูล

จากชนิดข้อมูลและตารางที่ได้ออกแบบไว้เราสามารถที่จะนำมาทดสอบกับข้อมูลโดยทำการแทรกข้อมูลดังนี้

```
INSERT INTO employee
VALUES(257,'สุรชัย  เลิศบุญช่วยกุล',22,20000,'5 10',"multiset{'200','300','500'}" );
```

```
INSERT INTO employee
VALUES(233,'ชวลิต  ศิวบวรวัฒนา',22,20000,'15 10',"multiset{'400','300','500','800'}" );
INSERT INTO employee
VALUES(259,'อำนาจ  มีมงคล',23,20000,'35 17',"multiset{'400','500','800'}" );
INSERT INTO employee
VALUES(226,'กนกวุธ  ถนัดการ',22,20000,'25 13',"multiset{'400','300','500','800','300','400'}" );
INSERT INTO employee
VALUES(250,'สมชาย  เป็นคนดี',25,25000,'35 4',"multiset{'400','300','100','200'}" );
```

ซึ่งถ้าเราทำการเลือกค่าข้อมูลในตาราง Employee เราจะได้ข้อมูลออกมาดังนี้

Id	Name	age	salary	location	bonus
257	สุรชัย เลิศบุญช่วยกุล	22	$20000.00	5.000000 10.000000	multiset(3) of money
233	ชวลิต ศิวบวรวัฒนา	22	$20000.00	15.000000 10.000000	multiset(4) of money
259	อำนาจ มีมงคล	23	$20000.00	35.000000 17.000000	multiset(3) of money
226	กนกวุธ ถนัดการ	22	$20000.00	25.000000 13.000000	multiset(6) of money
250	สมชาย เป็นคนดี	25	$25000.00	35.000000 4.000000	multiset(4) of money

และจากตอนต้นที่ได้แสดงให้เห็นว่าตาราง employee ได้มีการสืบทอดมาจากตาราง person นั้น ถ้าเราทำการเลือกค่าข้อมูลต่างๆ จากตาราง person จะได้ผลดังนี้

Id	name	Age
257	สุรชัย เลิศบุญช่วยกุล	22
233	ชวลิต ศิวบวรวัฒนา	22
259	อำนาจ มีมงคล	23
226	กนกวุธ ถนัดการ	22
250	สมชาย เป็นคนดี	25

ซึ่งจะเห็นว่าเราไม่ได้ทำการแทรกข้อมูลให้กับตาราง person เลยแต่เมื่อเราทำการ select ค่าข้อมูลในตาราง person กับเป็นว่าเราได้ข้อมูลที่อยู่ในตาราง employee ที่เป็นเช่นนี้เป็นเพราะตาราง employee เป็น subtable ของตาราง person นั้นเอง ซึ่งจากตัวอย่างของโมเดลนี้เราสามารถที่จะทำคิวรี่ในลักษณะต่างๆ เพื่อต้องการทดสอบคุณสมบัติทางออบเจ็กต์ได้ดังนี้

ตัวอย่าง ของคิวรี่ต่างๆ

Q1: ต้องการหาพนักงานที่มีบ้านอยู่ในบริเวณเดียวกันในระยะ 10 กิโลเมตรเราสามารถที่จะทำการคิวรี่ได้ดังนี้

```
SELECT t1.name,t2.name,distance(t1.location,t2.location) distance
FROM employee t1,employee t2
WHERE distance(t1.location,t2.location)<=10 and t1.id < t2.id;
```

ซึ่งเราก็จะได้ผลลัพธ์ดังนี้

ผลลัพธ์

Name		name		distance
สุรชัย	เลิศบุญช่วยกุล	ชวลิต	ศิวบวรวัฒนา	10.0000000000000

Q2:ต้องการหาพนักงานอาศัยอยู่ใกล้กันบริษัทในบริเวณ 5 กิโลเมตร

```
SELECT name, distance(location)
FROM employee
WHERE distance(location) <= 5;
```

ผลลัพธ์

Name		distance
สุรชัย	เลิศบุญช่วยกุล	5.00000000000000
ชวลิต	ศิวบวรวัฒนา	5.00000000000000

Q3: ต้องการหาพนักงานที่อาศัยใกล้กับลูกค้าที่อยู่ที่ตำแหน่งพิกัด 12,10 ที่สุด

```
SELECT name,distance(location,'12 10'),location
FROM employee
WHERE distance(location,'12 10') <= ALL(SELECT distance(location,'12 10')
                                        FROM employee);
```

ผลลัพธ์

Name	distance	location
ชวลิต ศิวบวรวัฒนา	3.00000000000000	15.000000 10.000000

Q4: ต้องการทราบข้อมูลทั้งหมดที่ในตาราง person

```
SELECT *
FROM person
```

ผลลัพธ์

Id	name	Age
257	สุรชัย เลิศบุญช่วยกุล	22
233	ชวลิต ศิวบวรวัฒนา	22
259	อำนาจ มีมงคล	23
226	กนกวุธ ถนัดการ	22
250	สมชาย เป็นคนดี	25

จากคิวรี่นี่จะเห็นว่าเราได้ทำการเลือกข้อมูลจากตาราง person แต่ได้ข้อมูลจากตาราง employee ที่เป็นเช่นนี้ก็เพราะว่าตาราง person เป็น parent table ของ employee

Q5:ต้องการรายชื่อของพนักงานที่จำนวนผลรวมทั้งหมดของโบนัสมากกว่า 1000 บาท

```
SELECT name,sumBonus(id)
FROM employee
WHERE sumBonus(id) > 1000;
```

ผลลัพธ์

name	bonus
ชวลิต ศิวบวรวัฒนา	$2000.00
อำนาจ มีมงคล	$1700.00
กนกวุธ ถนัดการ	$2700.00

ต่อไปนี้เป็นตารางที่ใช้สำหรับการทดสอบการใช้ฟังก์ชันที่ทำ Overriding

```
CREATE TABLE test_override
(
        no      integer,
        detail  person_t
);

CREATE TABLE emp
(
        no      integer,
        detail  employee_t
);
```

```
INSERT INTO test_override
VALUES(1,row(257,'สุรชัย เลิศบุญช่วยกุล',25 )::person_t);
INSERT INTO emp
VALUES(1,row(257,'สุรชัย เลิศบุญช่วยกุล',25,2000,'10 10',"multiset{'23','34'}")::employee_t);
```

Q6: ต้องการชื่อของพนักงานที่อยู่ในตาราง test_override

```
SELECT getinfo(detail) name
FROM test_override;
```

ผลลัพธ์

name
สุรชัย เลิศบุญช่วยกุล

Q7:ต้องการทราบเงินเดือนของพนักงานที่อยู่ในตาราง emp

```
SELECT getinfo(detail)
FROM emp;
```

ผลลัพธ์

Salary
2000

จาก Q6 และ Q7 จะเห็นว่ามีการเรียกใช้ฟังก์ชันที่มีชื่อเหมือนกันแต่ผลลัพธ์ที่ได้มาแตกต่างกันทั้งนี้เนื่องจากได้มีการทำฟังก์ชัน Overriding นั้นเอง

8.4 สรุปผลการทดสอบ

ซึ่งจากกรณีศึกษาที่เราได้ทำการทดสอบในลักษณะต่างๆ เหล่านี้ทำให้เราสามารถที่จะวิเคราะห์ได้ว่าระบบฐานข้อมูลของอินฟอร์มิกซ์ในรุ่นที่เป็น Universal server เวอร์ชัน 9.14 บน NT นี้ที่พยายามที่จะตามมาตรฐานที่มีการกำหนดไว้สำหรับระบบฐานข้อมูลที่เป็นแบบเชิงวัตถุสัมพันธ์และมาตรฐานของภาษา SQL3 ว่าตรงตามที่กำหนดไว้มากน้อยแค่ไหน ซึ่งก็พอที่จะสรุปคุณสมบัติในด้านสนับสนุนในเรื่องของออบเจ็คได้ดังนี้

1. สนับสนุนการทำ Encapsulate ข้อมูลต่างๆ ภายในออบเจ็กต์โดยตัวชนิดข้อมูลที่เป็นแบบ row type จะไม่มีคุณสมบัตินี้อยู่แล้วทั้งของใน SQL3 มาตรฐานเองหรือในของอินฟอร์มิกซ์ก็ตามแต่สำหรับข้อมูลที่เป็นแบบ ADT ที่เป็นชนิดข้อมูลที่เป็นออบเจ็กต์อีกตัวหนึ่งที่มีการกำหนดในภาษา SQL3 นั้นสามารถที่จะระบุจากทำ Encapsulate ได้ทั้ง 3 แบบคือ public, private และ protect แต่สำหรับชนิดข้อมูลที่เป็นแบบ Opaque ที่เป็นชนิดข้อมูลที่ทางอินฟอร์มิกซ์พยายามที่จะอิมพลิเมนต์ชนิดข้อมูลแบบ ADT ใน SQL3 นั้นกลับว่าสามารถที่จะทำการซ่อนเร้นข้อมูลได้แค่แบบเดียวคือเป็นแบบ private ทั้งนี้อันเนื่องมาจากการที่เราจะกำหนดชนิดข้อมูลแบบ Opaque ได้นั้นจะต้องมีการเขียนโค้ดเป็นภาษาซีให้เป็นไลบอรีใช้กับตัวระบบฐานข้อมูลและต้องทำการรีจิสเตอร์ให้ระบบฐานข้อมูลนี้รู้จักชนิดข้อมูลนี้โดยตัวระบบฐานข้อมูลเองจะไม่รู้อะไรเกี่ยวกับชนิดข้อมูลนี้เลยนอกจากจะรู้ว่าข้อมูลนี้ต้องมีฟังก์ชันอะไรบ้างที่จะมาช่วยจัดการสิ่งต่างๆ ให้และจะรู้จักเฉพาะโครงสร้างภายนอก (external representation) ของข้อมูลนี้เท่านั้นทำให้เราไม่สามารถที่จะกำหนดให้การซ่อนเร้นข้อมูลเป็นแบบ public ได้ส่วนการกำหนดการซ่อนเร้นข้อมูลแบบ protect นั้นที่ไม่สามารถกระทำได้สืบเนื่องมาจากที่ชนิดข้อมูลแบบ Opaque ไม่สามารถที่จะทำการสืบทอดตัวเองให้กับ Opaque ตัวอื่นๆ ได้ทั้งนี้เพราะตามหลักการของการซ่อนเร้นข้อมูลที่เป็นแบบ protect แล้วก็มีไว้เพื่อที่จะให้ชนิดข้อมูลหรือคลาสที่ทำการสืบทอดได้สามารถที่จะอ้างข้อมูลที่กำหนดแบบนี้ แต่การที่ชนิดข้อมูลแบบ Opaque ไม่สามารถที่จะทำการสืบทอดคุณสมบัติต่างๆ (inheritance) ได้นั้นทางอินฟอร์มิกซ์เองได้มีการแนะนำให้อาศัยการทำชนิดข้อมูลที่เป็นแบบ Distinct แทนเพราะชนิดข้อมูลแบบนี้โดยตัวมันเองแล้วจะทำการคัดลอกคุณสมบัติและฟังก์ชันที่กระทำกับข้อมูลที่เป็นซอส (source data type) ซึ่งก็เปรียบเหมือนว่าได้มีการทำการสืบทอดกับชนิดข้อมูลนั้น

2. การสนับสนุนคุณสมบัติการสืบทอดระบบฐานข้อมูลของอินฟอร์มิกซ์นี้สนับสนุนการสืบทอดข้อมูลและตารางดังที่พอจะเห็นบ้างในตัวอย่างที่ผ่านมาข้างต้น โดยการสืบทอดข้อมูลนั้นสำหรับข้อมูล

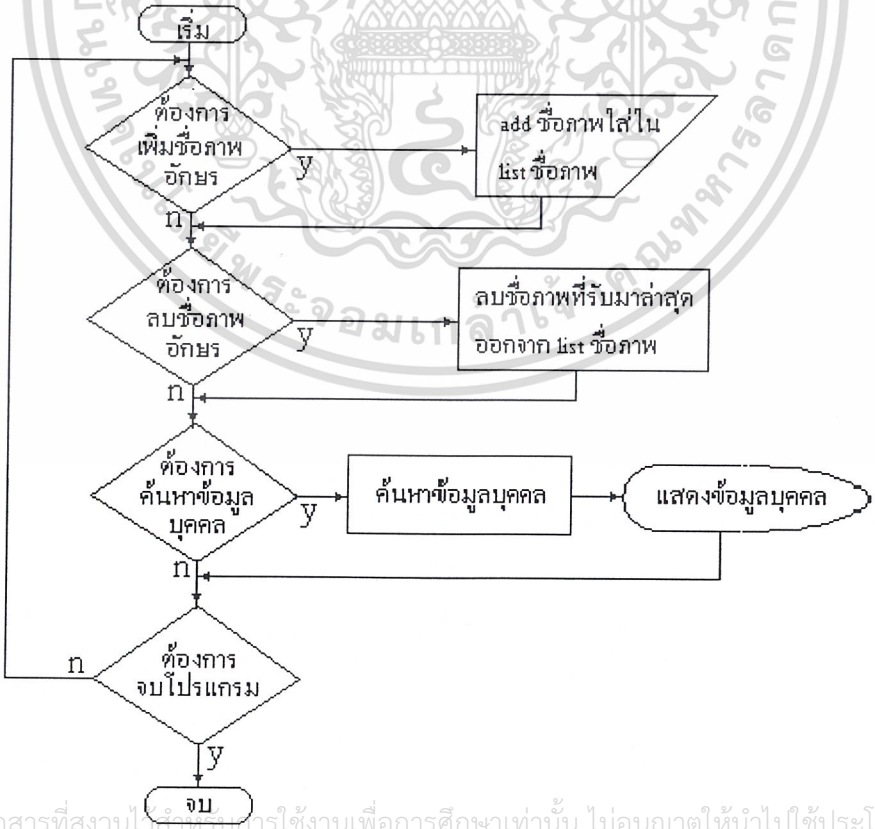
ของอินฟอร์มิกซ์สามารถที่จะสืบทอดได้เฉพาะข้อมูลที่เป็นแบบ row type ส่วน Opaque นั้นต้องอาศัยผ่านการสร้างชนิดข้อมูลที่เป็นแบบ distinct โดยสิ่งที่ได้ทำการสืบทอดไปนั้นก็ได้แก่โครงสร้างข้อมูลภายในและฟังก์ชันต่างๆ ที่กระทำกับข้อมูลที่เป็น parent type ส่วนการสืบทอดแบบตารางนั้นจะกระทำได้เฉพาะตารางที่เป็น Type Table ซึ่งก็คือตารางที่มีการกำหนดชนิดข้อมูลที่เป็นแบบ named row type ให้ซึ่งรายละเอียดเกี่ยวกับ Type table และ named row type นั้นได้กล่าวไปแล้วในส่วนของการสืบทอดของ informix ในบทต้นๆ ของงานวิจัยนี้

3. สนับสนุนการทำโพลิมอร์ฟีซึมระบบฐานข้อมูลของอินฟอร์มิกซ์สนับสนุนการทำโพลิมอร์ฟีซึมทั้งในรูปแบบของการทำ Overloading และ Overriding ของฟังก์ชันที่เป็นของ row type และ Opaque แต่สำหรับฟังก์ชันใดๆ ก็ตามที่ทำการโค้ดโดยใช้ภาษาซีนั้นในต้องสร้างฟังก์ชันที่ต้องการทำโพลิมอร์ฟีซึม นั้นจะต้องใช้ชื่อไม่เหมือนกันแล้วค่อยอาศัยในขั้นตอนการรีจิสเตอร์ให้มีชื่อเหมือนกันถึงแม้ว่าเราจะใช้ตัวโปรแกรม Bladesmith ช่วยสร้างก็ตามเพราะในขั้นตอนการแทรกฟังก์ชันที่มีชื่อเหมือนกับนั้นตัวโปรแกรมนี้จะไม่แจ้งข้อผิดพลาดใดๆ กับเราแต่เมื่อเราทำการให้โปรแกรมนี้สร้างโค้ดให้จะมีการแจ้งข้อพิดพลาดว่ามีการกำหนดชื่อของฟังก์ชันซ้ำ ซึ่งตัวอย่างการทำOverloading และ Overriding นั้นสามารถที่จะดูได้จากกรณีศึกษาข้างต้น

บทที่ 9
โปรแกรมสาธิตที่ 2 โปรแกรมเรียนรู้และจดจำอักษรภาษาไทย

ในบทนี้จะอธิบายรายละเอียดต่างๆของวิธีการสร้างโปรแกรมสาธิตซึ่งเป็นโปรแกรมที่ใช้ทดสอบการสนับสนุนชนิดข้อมูลที่มีความซับซ้อนสูงของระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์ โดยที่จะอธิบายทั้งส่วนของแบ็คเอนด์ (Back End) และส่วนของฟรอนท์เอนด์ (Front End) รวมไปถึงอธิบายถึงวิธีการทดลองใช้งานโปรแกรมสาธิตด้วย โปรแกรมสาธิตที่กล่าวถึงนี้คือโปรแกรมที่ทำหน้าที่ค้นหาข้อมูลของบุคคลที่เก็บอยู่ในระบบฐานข้อมูลซึ่งได้แก่ รหัสประจำตัว ชื่อ ที่อยู่ อายุ ฯลฯ โดยที่ข้อมูลที่จะนำไปใช้เพื่อการค้นหาข้อมูลของบุคคลที่ต้องการนั้นจะเป็นข้อมูลชื่อบุคคลซึ่งอยู่ในรูปแบบของภาพขาวดำที่อยู่ในรูปแบบ (Format) ของไฟล์นามสกุล BMP ข้อมูลอินพุทที่ต้องการคือรายการ (List) ชื่อของภาพตัวอักษรที่จะต้องเรียงลำดับให้ถูกต้องตามชื่อของบุคคลที่ต้องการค้นหานั้น แล้วโปรแกรมก็จะให้ผลลัพธ์ออกมาเป็นข้อมูลต่างๆของบุคคลที่ต้องการค้นหานั่นเอง โดยเบื้องหลังของการทำงานก็จะเป็นการนำเอาความรู้เรื่องการวิเคราะห์ลักษณะเด่นของตัวอักษรมาประยุกต์ ซึ่งหลักการวิเคราะห์ลักษณะเด่นของตัวอักษรนั้นก็ได้อธิบายไปแล้วในบทที่ผ่านมา

9.1 ขั้นตอนการสร้างโปรแกรมสาธิต



รูปที่ 9-1 Flow chart อธิบายการทำงานของโปรแกรมสาธิต

ต่อไปจะเป็นการอธิบายรายละเอียดของแต่ละขั้นตอนต่างๆในการสร้างโปรแกรมสาธิตรวมทั้งจะมีการกล่าวถึงปัญหาต่างๆที่เกิดขึ้นในขั้นตอนการทำงานและบอกถึงวิธีการแก้ไขของแต่ละปัญหาเหล่านั้นด้วย ก่อนอื่นต้องทำการกำหนดภาพรวมของการทำงานของโปรแกรมก่อน โดยพออธิบายเป็น Flow Chart ที่แสดงในรูปที่ 9-1

9.1.1 สร้างชนิดข้อมูลใหม่

เนื่องจากจุดประสงค์ของโครงการนี้เน้นที่จะศึกษาถึงคุณลักษณะทางด้าน ออปเจ็คของระบบจัดการฐานข้อมูล Informix Universal Server ดังนั้นเราจึงทดลองสร้างชนิดข้อมูลชนิดใหม่ขึ้นเพื่อใช้ในการเก็บข้อมูลที่เป็นลักษณะเด่นของตัวอักษรซึ่งก็คือชนิดข้อมูลที่สามารถเก็บข้อมูลที่เป็นรหัสรวมทั้งหกจุดที่ต้องการเก็บต่อตัวอักษรหนึ่งตัวนั่นเอง ข้อมูลชนิดใหม่ที่กำลังกล่าวถึงนี้เป็นข้อมูลที่เป็นแบบ Opaque Data Type ซึ่งเราได้เคยอธิบายไว้แล้วในบทต้นๆ

```
typedef struct
{
    mi_char         P[6][5];
    mi_char1        pad1[2];
    MI_LO_HANDLE    ImgBuffer;
}
OcrType;
```

จากที่เคยทราบแล้วว่าข้อมูลที่เป็น Opaque Data Type นั่นเป็นข้อมูลที่เป็นแบบสตรัคเจอร์ (structure) ของภาษาซี พิจารณาโครงสร้างได้จากด้านบน สำหรับชนิดข้อมูลที่เราสร้างขึ้นใหม่นี้ประกอบด้วย 2 ส่วนหลักๆคือ ส่วนแรกเป็นตัวแปรอาเรย์ 2 มิติ คือ P มีขนาดเท่ากับหกคูณห้า ใช้เก็บข้อมูลลักษณะเด่นของตัวอักษร ส่วนที่สองคือใช้สำหรับชี้ไปยังข้อมูลที่เป็นบัฟเฟอร์ (Buffer) ที่ใช้ในการเก็บข้อมูลของภาพตัวอักษรชั่วคราวที่ใช้ในขั้น ImgBuffer ตอนของการหาลักษณะเฉพาะของตัวอักษรในภาพนั้น ส่วนแอททริบิวท์ที่ไม่ได้กล่าวถึงนั้นเป็นแอททริบิวท์ที่โปรแกรม BladeSmith ทำสร้างขึ้นมาเองโดยที่เราไม่ได้กำหนดแต่อย่างใด ชนิดข้อมูลที่สร้างขึ้นใหม่นี้มีชื่อว่า OcrType

ในขั้นตอนที่เราใช้โปรแกรม BladeSmith ทำการสร้างชุดคำสั่งภาษาซีตามวิธีที่ได้อธิบายไว้ในภาคผนาก ค นอกจากโปรแกรม BladeSmith จะทำการสร้างฟังก์ชันสนับสนุน (Support Routine) ต่างๆให้กับข้อมูลชนิดใหม่ที่เราสร้างแล้ว เรายังได้ทำการกำหนดให้โปรแกรมทำการสร้างฟังก์ชันที่เราต้องการใช้เพิ่มเติมให้อีกด้วย

หลังจากได้ชุดคำสั่งมาเรียบร้อยแล้วก็ต้องนำเอาชุดคำสั่งเหล่านั้นมาทำการเพิ่มเติมและแก้ไขเพื่อให้การทำงานต่างๆตรงตามแบบที่เราต้องการ โดยเราต้องแก้ไขฟังก์ชัน OcrTypeInput ใหม่ ซึ่งเป็นฟังก์ชันที่จะถูกเรียกใช้งาน โดยระบบจัดการฐานข้อมูลในตอนที่มีการใช้คำสั่ง Insert กับข้อมูล OcrType

ซึ่งสิ่งที่เราต้องแก้ไขก็คือเพิ่มชุดคำสั่งในส่วนของการวิเคราะห์หาลักษณะเฉพาะของตัวอักษร นอกจากนั้นก็ต้องแก้ไขฟังก์ชัน OcrTypeOutput ด้วย ซึ่งจะเป็นฟังก์ชั่นที่จะถูกเรียกใช้งานโดยระบบจัดการฐานข้อมูลในตอนที่ใช้คำสั่ง Select กับข้อมูลชนิด OcrType สำหรับชุดคำสั่งที่ทำการแก้ไขแล้วมีดังนี้

```
OcrType *
OcrTypeInput
(
mi_lvarchar *        Gen_param1,              /* Pointer to the input text.      */
MI_FPARAM *          Gen_fparam               /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *      Gen_Con;        /* The current connection.          */
        gl_mchar_t *    Gen_InData;          /* Pointer to the input data.       */
        gl_mchar_t *    Gen_StartInData;     /* First value of Gen_InData.       */
        OcrType *       Gen_OutData;         /* Pointer to the output data.      */
        mi_integer      Gen_DataLen;         /* Length of the data in bytes.     */
        OcrType *       Gen_RetVal;          /* The return value.                */
        mi_integer        Gen_Index10; /* Array iterator */
        char              Gen_LOFile[FILENAME_MAX];/* Store the file name here.  */
        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 )
                { DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG1, 10 ); }
        DBDK_TRACE_ENTER( "OcrTypeInput" );
        Gen_InData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *) Gen_param1 );
        Gen_StartInData = Gen_InData;
        Gen_DataLen = mi_get_varlen( Gen_param1 );
        Gen_RetVal = (OcrType *)mi_alloc( sizeof( OcrType ) );
        if( Gen_RetVal == 0 )
                { DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG2, 10 ); }
        Gen_OutData = (OcrType *)Gen_RetVal;
        Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeInput", Gen_InData,
                                                mi_get_varlen( Gen_param1 ),
                                                sizeof(Gen_LOFile)-1,
                                                "%s %n",
```

```
                                        Gen_LOFile );
        Gen_LoadLOFromFile( Gen_Con, "OcrTypeInput", Gen_LOFile,
                        &Gen_OutData->ImgBuffer );
        Gen_OutData = Recorgnize( Gen_Con, Gen_OutData );
        DBDK_TRACE_EXIT( "OcrTypeInput" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

mi_lvarchar *
OcrTypeOutput
(
OcrType *           Gen_param1,     /* The UDT value.              */
MI_FPARAM *         Gen_fparam      /* Standard info - see DBDK docs.   */
)
{
        MI_CONNECTION *     Gen_Con;        /* The current connection.      */
        mi_integer    Gen_CharLen;    /* Estimate maximum length.     */
        OcrType *     Gen_InData;     /* Pointer to the input data.   */
        char *        Gen_OutData;    /* Pointer to the output data.  */
        mi_lvarchar *  Gen_RetVal;    /* The return result.           */
        mi_integer    Gen_DataLen;    /* The data length.             */
        mi_integer          Gen_Index10; /* Array iterator */
        char                Gen_LOFile[FILENAME_MAX];/* Store the file name here.   */
        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 )
                { DBDK_TRACE_ERROR( "OcrTypeOutput", ERRORMESG1, 10 ); }
        DBDK_TRACE_ENTER( "OcrTypeOutput" );
        Gen_InData = Gen_param1;
        Gen_CharLen     =   1           /* Leave room for the NULL terminator. */
                        + 6 * 5 * 2     /* Add the length for P. */
                        + 261;    /* Add the length for ImgBuffer. */
        Gen_RetVal = mi_new_var( Gen_CharLen );
        if( Gen_RetVal == 0 )
```

```
        { DBDK_TRACE_ERROR( "OcrTypeOutput", ERRORMESG2, 10 ); }
        Gen_OutData = mi_get_vardata( Gen_RetVal );
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; )
        {
                mi_integer      Gen_Index20;            /* Array Index.         */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; )
                {
                        sprintf(Gen_OutData,"%c ", Gen_InData->P[Gen_Index10][Gen_Index20] );
                        Gen_OutData += strlen( Gen_OutData );
                }
        }
        Gen_OutData += strlen( Gen_OutData );
        Gen_DataLen = (mi_integer)(Gen_OutData - mi_get_vardata( Gen_RetVal ));
        mi_set_varlen ( Gen_RetVal, Gen_DataLen );
        DBDK_TRACE_EXIT( "OcrTypeOutput" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}
```

อีกหนึ่งฟังก์ชันที่ต้องแก้ไขซึ่งเป็นฟังก์ชันสนับสนุนชนิดข้อมูลใหม่ที่เราสร้างขึ้นก็คือฟังก์ชันที่ชื่อ OcrTypeEqual ฟังก์ชันนี้เป็นฟังก์ชันที่จะถูกเรียกใช้งานในตอนที่มีการเปรียบเทียบระหว่างสองข้อมูลที่มีชนิดข้อมูลเป็น OcrType ดังที่เคยทราบมาแล้วว่าในส่วนของเซิร์ฟเวอร์นั่นไม่รู้จักลักษณะโครงสร้างภายของชนิดข้อมูลที่เป็น Opaque Type เราจึงต้องทำการแก้ไขเพิ่มเติมชุดข้อมูลในส่วนของของฟังก์ชันนี้เพื่อให้โปรแกรมทำงานได้ถูกต้องตามความต้องการของเรา ข้างล้างนี้คือฟังก์ชัน OcrTypeEqual

```
mi_boolean
OcrTypeEqual
(
OcrType *           Gen_param1,     /* The first UDT value to compare.   */
OcrType *           Gen_param2,     /* The second UDT value to compare.  */
MI_FPARAM *         Gen_fparam      /* Standard info - see DBDK docs.    */
)
{
        int i, j, count;
```

```
        OcrType * opaque;
        OcrType * opaque2;

        opaque = Gen_param1;
        opaque2= Gen_param2;
        for(count=i=0; i<6; i++)
                for(j=0; j<5; j++)
                        if(opaque->P[i][j]==opaque2->P[i][j]) count+=2;
                        else if( (opaque->P[i][j]=='S' && opaque2->P[i][j]=='I') ||
                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='S') ||
                                (opaque->P[i][j]=='V' && opaque2->P[i][j]=='I') ||
                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='V') ||
                                (opaque->P[i][j]=='H' && opaque2->P[i][j]=='I') ||
                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='H')
                                ) count++;
        return (count>=20)? 1 : 0;
}
```

หลังจากแก้ไขการทำงานของฟังก์ชันสนับสนุนต่างๆเรียบร้อยแล้ว เรายังได้เพิ่มเติมการทำงานในส่วนของฟังก์ชัน nearScore ซึ่งเป็นฟังก์ชันที่เรากำหนดให้โปรแกรม BladeSmith ทำการสร้างมาให้ในตอนที่สร้างชนิดข้อมูล OcrType สำหรับหน้าที่ของฟังก์ชันนี้ก็คือจะทำการนำเอาชื่อของภาพตัวอักษรซึ่งเป็นอินพุทที่ได้รับ มาทำการเปิดไฟล์แล้วทำการใช้วิธีการในการวิเคราะห์ลักษณะเด่นของตัวอักษรในภาพนั้นก่อนที่จะทำการเปรียบเทียบกับค่าลักษณะเด่นของตัวอักษรซึ่งเป็นอินพุทอีกตัวที่รับเข้ามาเพื่อเปรียบเทียบให้คะแนนความคล้ายของตัวอักษรนั้น แล้วส่งค่ากลับไปเป็นความเหมือน ซึ่งหลักการในการให้คะแนนความเหมือนนั้นก็ได้กล่าวไว้แล้วในบทที่ผ่านมา เมื่อเขียนเป็นฟังก์ชันก็จะเป็นดังนี้

```
mi_integer nearScore
(
OcrType *           opaque,
mi_lvarchar *       fileName,
MI_FPARAM *         Gen_fparam      /* Standard info - see DBDK docs.*/
)
{
        MI_CONNECTION * Gen_Con;        /* The connection handle.          */
```

```
OcrType * opaque2;
gl_mchar_t * Gen_InData;
mi_integer Gen_DataLen, count;
char Gen_LOFile[FILENAME_MAX];
int i, j;

mi_integer     Gen_RetVal;     /* The return value.              */
Gen_Con = mi_open( NULL, NULL, NULL );
if( Gen_Con == 0 )
        { DBDK_TRACE_ERROR( "nearScore", ERRORMESG1, 10 ); }
DBDK_TRACE_ENTER( "nearScore" );
Gen_InData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *) fileName );
Gen_DataLen = mi_get_varlen( fileName );
opaque2 = (OcrType *)mi_alloc( sizeof( OcrType ) );
if( opaque2 == 0 )
        { DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG2, 10 ); }
Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeInput", Gen_InData,
                                mi_get_varlen( fileName ),
                                sizeof(Gen_LOFile)-1,
                                "%s %n",
                                Gen_LOFile );
Gen_LoadLOFromFile( Gen_Con, "OcrTypeInput", Gen_LOFile,
        &opaque2->ImgBuffer );
opaque2 = Recorgnize( Gen_Con, opaque2 );
for(count=i=0; i<6; i++)
        for(j=0; j<5; j++)
                if(opaque->P[i][j]==opaque2->P[i][j]) count+=2;
                else if( (opaque->P[i][j]=='S' && opaque2->P[i][j]=='I') ||
                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='S') ||
                        (opaque->P[i][j]=='V' && opaque2->P[i][j]=='I') ||
                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='V') ||
                        (opaque->P[i][j]=='H' && opaque2->P[i][j]=='I') ||
                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='H')
                        ) count++;
```

```
        Gen_RetVal = count;
        DBDK_TRACE_EXIT( "nearScore" );
        return Gen_RetVal;
}
```

สำหรับชุดคำสั่งต่างๆแบบสมบูรณ์ที่เป็นชุดคำสั่งภาษาซีนั้น ได้แสดงไว้เรียบร้อยแล้วในส่วนของภาคผนวก ค

9.1.2 สร้างตารางและจัดเตรียมข้อมูลที่จำเป็น

สำหรับโปรแกรมสาธิตนี้เราได้สร้างตารางขึ้นมาเพื่อใช้ในการเก็บข้อมูลโดยสร้างทั้งหมด 5 ตารางคือ ascii_value, ascii_code, person_table, code_name_table และ temp_table โดยที่รายละเอียดภายในของแต่ละตารางสามารถดูได้จากคำสั่งที่ใช้ในการสร้างตารางดังต่อไปนี้

```
create row type ascii_value_type (
        ascii  int,
        charValue  char,
        stringValue  lvarchar
);
create table ascii_value of type ascii_value_type
(primary key (ascii));
```

ตาราง ascii_value_type ประกอบด้วย 3 แอททริบิวท์ (Attribute) คือ ascii มีชนิดข้อมูลเป็นเลขจำนวนเต็มใช้สำหรับเก็บค่ารหัสแอสกี้ของแต่ละตัวอักษรและแอททริบิวท์นี้ถูกกำหนดให้เป็นคีย์หลัก (primary key) ด้วย โดยที่แอททริบิวท์ที่ 2 คือ charValue มีชนิดเป็นตัวอักษรใช้เก็บตัวอักษรของรหัสแอสกี้นั้น และแอททริบิวท์สุดท้ายคือ stringValue มีชนิดข้อมูลเป็นแบบข้อความใช้เก็บตัวอักษรของรหัสแอสกี้นั้นโดยเก็บในรูปแบบของข้อความที่มีตัวอักษรเดียว

```
create row type ascii_code_type (
        ascii  int,
        font_number int,
        code  OcrType
);

create table ascii_code of type ascii_code_type
(primary key (ascii, font_number));
```

ตาราง ascii_code เป็นตารางที่ใช้เก็บข้อมูลเกี่ยวกับลักษณะเด่นของแต่ละตัวอักษร โดยที่แต่ละบรรทัดในตารางระบุข้อมูลได้ว่าตัวอักษรของรหัสแอสกี้ใด (เก็บในแอททริบิวท์ ascii ซึ่งมีชนิดข้อมูลเป็นเลขจำนวนเต็ม) ของรูปแบบตัวอักษรหมายเลขใด (เก็บในแอททริบิวท์ font_number ซึ่งมีชนิดข้อมูลเป็นเลขจำนวนเต็ม) มีลักษณะเด่นอย่างไร (เก็บในแอททริบิวท์ code ซึ่งมีชนิดข้อมูลเป็น OcrType)

```
create row type name_type (
        first  varchar(10),
        last  varchar(20)
);

create row type address_type (
        number  varchar(10),
        road  varchar(20),
        province  varchar(20)
);

create row type person_type (
        person_id  varchar(9),
        person_name  name_type,
        person_address  address_type,
        person_age  integer
);

create table person_table of type person_type
( primary key (person_id) );
```

ตาราง person_table ใช้สำหรับเก็บข้อมูลส่วนบุคคล แต่ละบรรทัดของตารางจะประกอบด้วยรหัสประจำตัวของแต่ละบุคคล (เก็บในแอททริบิวท์ person_id ซึ่งมีชนิดข้อมูลเป็นแบบข้อความ), ชื่อของบุคคล (เก็บในแอททริบิวท์ person_name มีชนิดข้อมูลเป็น name_type ซึ่งเป็นชนิดข้อมูลที่เป็นแบบ row type ประกอบด้วยสองแอททริบิวท์ย่อยคือ first และ last มีชนิดข้อมูลเป็นแบบข้อความที่ใช้เก็บชื่อและนามสกุลเป็นลำดับ), ที่อยู่ของบุคคล (เก็บในแอททริบิวท์ person_address มีชนิดข้อมูลเป็น address_type ซึ่งเป็นข้อมูลแบบ row Type ที่มีสามแอททริบิวท์ย่อยคือ number, road และ province มีชนิดข้อมูลเป็น

แบบข้อความ ใช้เก็บ เลขที่บ้าน, ถนน และ จังหวัด ตามลำดับ) และ อายุ (เก็บใน person_age ซึ่งมีชนิดข้อมูลเป็นตัวเลขจำนวนเต็ม)

```
create row type code_name_type (
        person_id  varchar(9),
        position  integer,
        code  OcrType
);

create table code_name_table of type code_name_type
( primary key (person_id, position) );
```

ตาราง code_name_table เป็นตารางที่ใช้สำหรับเก็บข้อมูลชื่อของบุคคล โดยเก็บในลักษณะของลำดับของลักษณะเด่นของตัวอักษรที่เรียงตามลำดับที่ถูกต้องตามชื่อของบุคคลนั้นๆ ข้อมูลจะบ่งบอกว่าข้อมูลลักษณะเด่นของตัวอักษร (เก็บในแอททริบิวท์ code ซึ่งมีชนิดข้อมูลเป็น OcrType) ตัวที่เท่าไหร่ของชื่อ (เก็บในแอททริบิวท์ position ซึ่งมีชนิดข้อมูลเป็นเลขจำนวนเต็ม) ของบุคคลที่มีรหัสอะไร (เก็บในแอททริบิวท์ person_id ซึ่งมีชนิดข้อมูลเป็นแบบข้อความ) มีค่าเป็นเท่าไร

```
create row type temp_type (
        position  integer,
        code  OcrType
);

create table temp_table of type temp_type
( primary key (position) );
```

ตาราง temp_table เป็นตารางที่ใช้เก็บข้อมูลชื่อของหนึ่งบุคคลเท่านั้น โดยที่จะเก็บลักษณะเด่นของตัวอักษรภายในชื่อ ข้อมูลของแต่ละบรรทัดจะระบุว่าตัวอักษรตำแหน่งที่เท่าไหร่ของบุคคลนั้น มีค่าลักษณะเด่นเป็นอย่างไร ข้อมูลเหล่านั้นจะเก็บในแอททริบิวท์ position (มีชนิดเป็นเลขจำนวนเต็ม) และ code (มีชนิดข้อมูลเป็น OcrType) ตามลำดับ

เพื่อให้เห็นภาพของตารางทั้งหมดที่กล่าวมาข้างต้นนั้นพิจารณาได้ดังรูปที่ 9-2

ascii_value

ascii	charValue	stringValue

ascii_code

ascii	code

person_table

person_id	person_name		person_address			person_age
	first	last	number	road	province	

code_name_table

person_id	position	code

temp_table

position	code

รูปที่ 9-2 ตารางต่างๆใช้เก็บข้อมูลที่จำเป็นสำหรับโปรแกรมสาธิต

9.1.3 สร้างฟรอนท์เอนด์และฟังก์ชันสนับสนุนต่างๆ

สำหรับในส่วนของฟรอนท์เอนด์ นั่น เราได้เลือกใช้โปรแกรม Visual Basic 5.0 ในการเขียนโปรแกรม เนื่องจากผลิตภัณฑ์ของ Informix Universal Server เป็นผลิตภัณฑ์แรก ของบริษัท Informix ที่มีคุณสมบัติเป็นระบบจัดการฐานข้อมูลแบบออปเจ็ครีเลชั่นนอล และในโครงงานนี้เราต้องการที่จะใช้คุณสมบัตินั้นในการสร้างโปรแกรมสาธิตแต่เนื่องจากโปรแกรม Visual Basic 5.0 ยังไม่มีคุณสมบัติที่สร้างมาเพื่อสนับสนุนคุณลักษณะทางด้านออปเจ็คของ Informix Universal Server ทางบริษัท Informix จึงแก้ไขโดยการสร้างสิ่งที่จะนำมาใช้เสริมความสามารถทางด้านออปเจ็คให้กับโปรแกรม Visual Basic เพื่อให้สามารถใช้งานร่วมกับ Informix Universal Server ได้ โดยสิ่งที่กำลังกล่าวถึงนี้มีชื่อเรียกว่า Data Director

จากที่กล่าวมาแล้วข้างต้นนั่นหมายความว่า ถ้าหากเราต้องการที่จะนำโปรแกรม Visual Basic เพื่อมาใช้ในการสร้างโปรแกรมในส่วนฟรอนท์เอนด์เพื่อใช้ติดต่อกับฐานข้อมูลของระบบจัดการฐานข้อมูล Informix Universal Server ณ ปัจจุบันนี้ จะต้องทำการติดตั้ง Data Director ให้กับโปรแกรม Visual Basic เสียก่อน ซึ่งตัว Data Director นั้นสามารถหาได้จากโฮมเพจของบริษัท Informix แล้วนำมาติดตั้งใช้งานได้ชั่วคราว แต่ถ้าต้องการให้ใช้งานได้ตลอดก็ต้องทำการติดต่อซื้อซอร์ฟแวร์นี้กับทางบริษัท สำหรับโปรแกรมสาธิตที่ได้ทำไปแล้วในโครงการนี้ก็ติดตั้งแบบชั่วคราวลงไปก็สามารถใช้งานได้

เมื่อติดตั้ง Data Director ให้กับโปรแกรม Visual Basic เรียบร้อยแล้ว ก็สามารถใช้ในการสร้างส่วนฟรอนท์เอนด์ได้แล้ว โดยใช้งานร่วมกับ Data Director สำหรับขั้นตอนคร่าวๆในการใช้งาน Data Director เพื่อติดต่อกับฐานข้อมูล พออธิบายได้ดังนี้

1. ทำการสร้างอิมพอร์ทโมเดลโดยเลือกที่เมนู Data Director แล้วเลือกต่อที่ Model แล้วเลือก Import หลังจากนั้นโปรแกรมก็จะให้เรากรอกรายละเอียดต่างๆเกี่ยวกับฐานข้อมูลที่ต้องการจะให้แอปพลิเคชันของเราทำการติดต่อด้วย ก็ให้กรอกให้เรียบร้อย พอเสร็จก็ให้บันทึกลงไฟล์

2. ภายในชุดคำสั่งของภาษาเบสิกที่เราเขียนนั้น จะต้องประกาศออปเจ็คต่างๆที่ต้องการใช้งาน สำหรับโปรแกรมสาธิตของโครงงานนี้ใช้ออปเจ็ค 5 ชนิดคือ ddoEnine, ddoProject, ddoDataGroup, ddoTable และ ddoModel ดังตัวอย่างชุดคำสั่งเป็นการประกาศออปเจ็คเพื่อใช้งาน

```
Dim oEngine1 As ddoEngine
Dim oProject1 As ddoProject
Dim oDataGroup1 As ddoDataGroup
Dim oVTable1 As ddoTable
Dim oModel1 As ddoModel
```

3. หลังจากนั้นก็ทำการใช้งานออปเจ็คต่างๆที่เราประกาศไว้นำมาใช้งานได้เลย ดังที่จะยกตัวอย่างต่อไปนี้จะเป็นใช้คำสั่ง SQL โดยส่งผ่านทางออปเจ็คของ DataDirector

```
Set oEngine1 = New ddoEngine
Set oProject1 = oEngine1.CreateProject
oProject1.Name = "OR_DBMS"
Set oModel1 = oEngine1.CreateModel("d:\data\vb\model1.mlt")
Set oDataGroup1 = oProject1.CreateDataGroup("ascii_value", "ascii_value",
"d:\data\vb\model1.mlt ")
oDataGroup1.Logon "informix", "inform123"
Set oVTable1 = oDataGroup1.ExecuteSQLCommand("select * from person", "oVTable1")
oDataGroup1.DeleteVirtualTable ("oVTable1")
oDataGroup1.Logoff
```

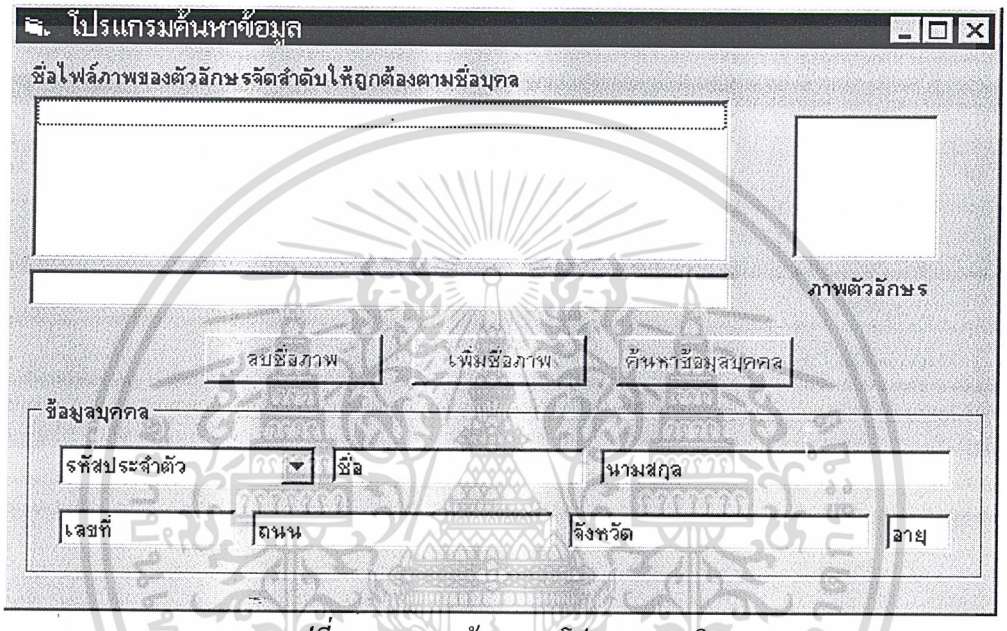
จากตัวอย่างชุดคำสั่งด้านบนคำสั่งแรกเป็นการสร้างออปเจ็ค ddoEngine เพื่อนำไปใช้ในการสร้างออปเจ็ค ddoProject ในคำสั่งที่สอง คำสั่งที่สามเป็นการตั้งชื่อให้กับโปรเจ็ค คำสั่งที่สี่เป็นการสร้างออปเจ็ค ddoModel โดยจะต้องระบุชื่อไฟล์ที่เราเคยตั้งไว้ในตอนอิมพอร์ทโมเดลในขั้นตอนที่สอง คำสั่งที่ห้าเป็นการสร้างออปเจ็ค ddoDataGroup เพื่อนำไปใช้งานในการกระทำคำสั่งภาษา SQL คำสั่งที่หกเป็นการลอกอินเข้าระบบฐานข้อมูล คำสั่งที่เจ็ดเป็นการส่งคำสั่ง SQL ไปทำงานเพื่อให้ได้ผลลัพธ์ของคำสั่งกลับมาที่ออปเจ็ค ddoTable หลังจากนั้นข้อมูลในออปเจ็ค ddoTable ก็สามารถนำไปใช้งานใดๆก็ได้ คำสั่งต่อไปเป็นการลบออปเจ็ค ddoTable ที่เคยสร้างไว้เมื่อไม่ต้องการใช้งานข้อมูลนั้นแล้ว ส่วนคำสั่งสุดท้ายก็คือการออกจากระบบฐานข้อมูล สำหรับรายละเอียดของการใช้งาน Data Director สามารถดูได้ในส่วนของภาคผนวกได้

ในส่วนที่มีการส่งคำสั่งภาษา SQL เพื่อให้ทำงานนั้น จะสามารถพิมพ์คำสั่งภาษา SQL ให้อยู่ในรูปของสตริงได้ในความยาวที่จำกัด ดังนั้นจากที่ได้ทดลองทำมาจึงพบว่าเกิดปัญหาในกรณีที่ต้องการทำคำ

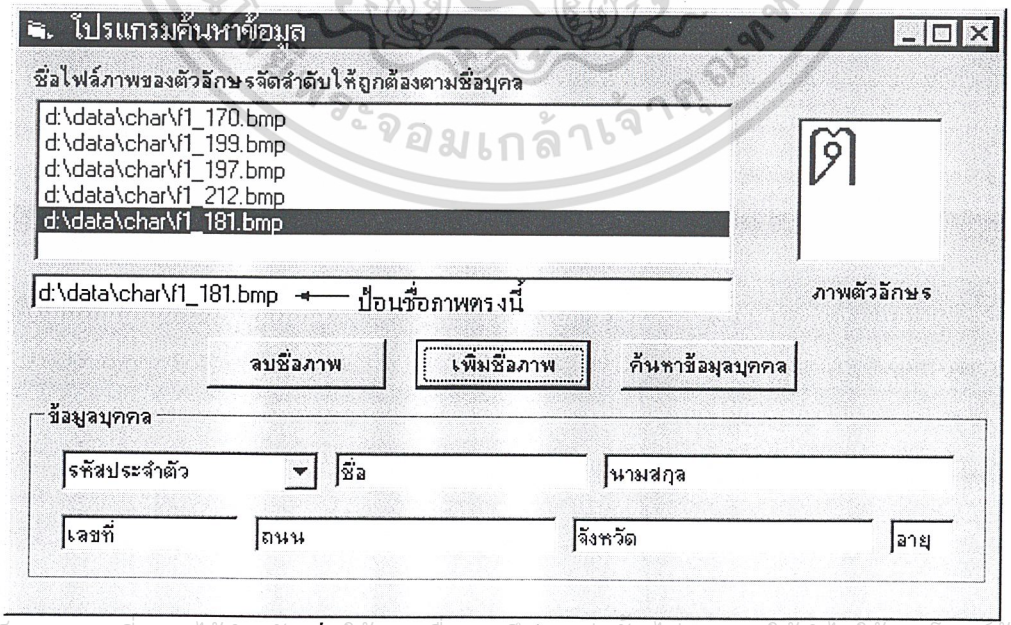
สั่งที่มีความยาวมากเกินกว่าข้อกำหนดของสตริง แต่เราสามารถจะเรียกใช้ฟังก์ชันของ SPL ก็แทนการพิมพ์คำสั่งทั้งหมดได้ ในโครงงานนี้ก็ได้สร้างฟังก์ชันโดยใช้ภาษา SPL เพื่อให้เรียกใช้งานได้ในความยาวไม่เกินข้อกำหนดของสตริง ทำให้สามารถแก้ไขปัญหาดังกล่าวได้

9.2 การทดลองใช้งานโปรแกรมสาธิต

ดังที่กล่าวนำไปแล้วว่าการทำงานของโปรแกรมสาธิตก็คือการค้นหาข้อมูลบุคคล โดยใช้ข้อมูลภาพของตัวอักษรของชื่อเพื่อเป็นข้อมูลนำ ไปค้นหาต่อไปจะอธิบายถึงวิธีการทดลองใช้งานโปรแกรมสาธิต เมื่อเริ่มเรียกใช้งานโปรแกรมก็จะปรากฏหน้าต่างดังรูปที่ 9-3



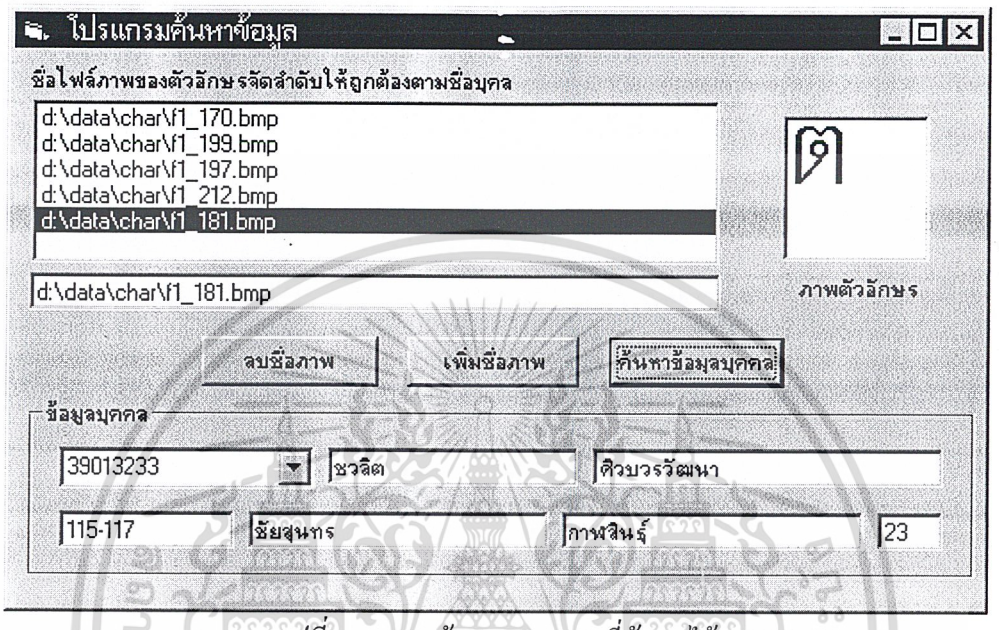
รูปที่ 9-3 แสดงหน้าตาของโปรแกรมสาธิต



รูปที่ 9-4 แสดงการป้อนชื่อภาพของตัวอักษร

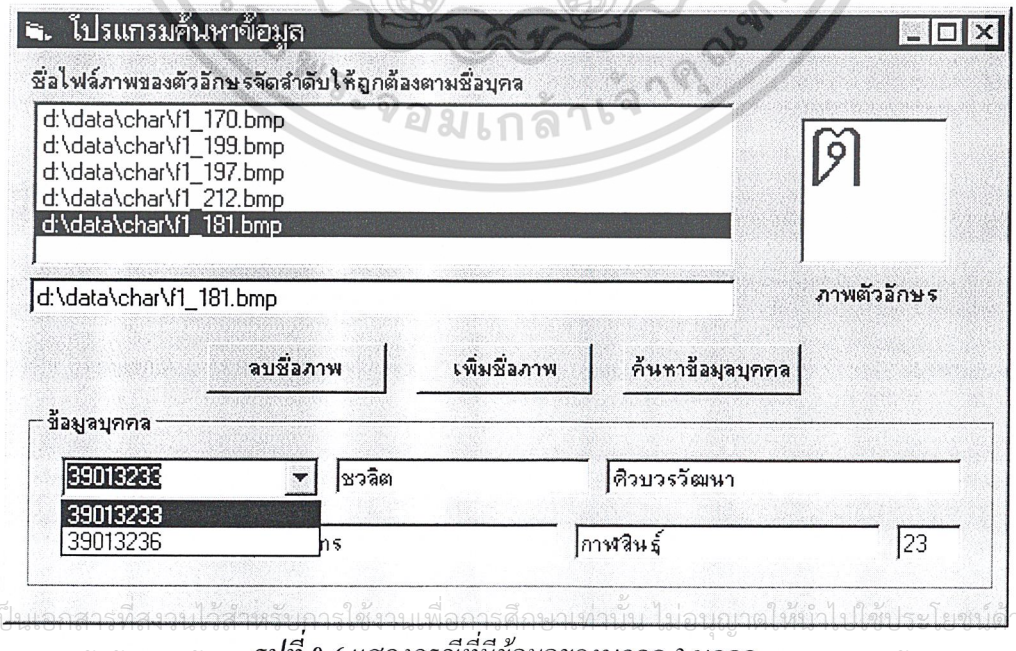
ต่อไปก็ทำการป้อนชื่อของภาพตัวอักษรที่ต้องการจะใช้เป็นข้อมูลในการค้นหาขอมูลบุคคลโดยต้องป้อนชื่อภาพตามลำดับที่ถูกต้องตามลำดับที่ตรงกับชื่อของบุคคลที่ต้องการ

เมื่อป้อนข้อมูลครบตามที่ต้องการแล้วก็กดที่ปุ่มค้นหาข้อมูลบุคคล หลังจากนั้นเบื้องหลังการทำงานของโปรแกรมก็จะทำการนำเอารูปภาพทั้งหมดที่รับเข้ามาไปใส่ลงในตาราง temp_table ที่เคยกล่าวไปแล้วตอนต้นของบทนี้ ในขั้นตอนดังกล่าวนี้เองภาพจะถูกเปลี่ยนเป็นค่าลักษณะเฉพาะเก็บไว้ในตาราง



รูปที่ 9-5 แสดงข้อมูลของบุคคลที่ค้นหาได้

หลังจากนั้นเราก็ทำการใช้คำสั่งภาษา SQL เพื่อทำการ join ตารางพร้อมทั้งกำหนดเงื่อนไขต่างๆ เพื่อให้ได้มาซึ่งข้อมูลของบุคคลที่เราต้องการ ดังแสดงไว้ในรูปที่ 9-5



รูปที่ 9-6 แสดงกรณีที่มีข้อมูลของบุคคล 2 บุคคล

ในกรณีที่ข้อมูลของบุคคลมีชื่อซ้ำกันมากกว่า 1 คน เราก็สามารถมารถเลือกดูข้อมูลของแต่ละคนได้โดยกดที่เครื่องหมายลูกศรที่อยู่หลังข้อมูลช่องรหัสประจำตัว แล้วก็กดเลือกไปที่รหัสประจำตัวของบุคคลที่เราต้องการก็จะดูข้อมูลของคนๆนั้นได้ ดังตัวอย่างในรูปที่ 9-6 เป็นตัวอย่างกรณีที่มีข้อมูลของบุคคล 2 บุคคล

บทที่ 10
สรุปและวิจารณ์

เนื่องจากจุดประสงค์ของการทำโครงงานนี้มีเป้าหมายที่จะศึกษาถึงคุณลักษณะทางด้านออปเจ็กต์ของผลิตภัณฑ์ซอฟต์แวร์ Informix Universal Server โดยที่เราเลือกเอางานทางด้านสาขาปัญญาประดิษฐ์มาบ่ระยุกต์เพื่อทดลองใช้งานความสามารถทางด้านออบเจ็กต์ดังที่กล่าวมา โดยเลือกเอาทฤษฎีที่เกี่ยวกับการรู้จำตัวอักษรมาประยุกต์ และเพื่อไม่ให้เสียเวลามากจนเกินไปกับการศึกษาทฤษฎีที่เกี่ยวกับการวิเคราะห์หาลักษณะเด่นของตัวอักษร โปรแกรมสาธิตนี้จึงเลือกใช้ทฤษฎีที่ไม่ยากจนเกินไปเพื่อนำมาวิเคราะห์หาลักษณะเด่นของตัวอักษรเพื่อใช้ในการเก็บข้อมูล ดังนั้นความสามารถทางการวิเคราะห์ลักษณะเด่นของตัวอักษรจึงอาจยังไม่มีประสิทธิภาพอย่างเต็มที่ แต่ก็อยู่ในระดับที่ทำให้ทราบแนวทางของการสร้างโปรแกรมลักษณะนี้ได้ แม้ว่าจะไม่ดีที่สุดแต่โปรแกรมก็สามารถแสดงให้เห็นถึงความสามารถในการรู้จำตัวอักษรได้เหมือนกัน แต่อาจมีข้อด้อยอยู่บ้างในกรณีที่ลักษณะรูปแบบของตัวอักษรนั้นมีความแปลกเกินไปหรือมีความไม่ชัดเจน

แต่ด้วยลักษณะของแอปพลิเคชันซึ่งเป็นการเรียงตัวอักษรให้ตรงตามชื่อบุคคล ทำให้โอกาสที่จะค้นหาข้อมูลมาจากชื่อที่ไม่ถูกต้องมีความน่าจะเป็นน้อยลงมาก ยกเว้นแต่ว่าจะมีชื่อที่มีลักษณะของตัวอักษรคล้ายกันเช่น กาญจนา และ ฉาญจนา อาจจะเป็นตัวอย่างของข้อมูลที่อาจทำให้เกิดข้อผิดพลาดได้ แต่จะเห็นว่าข้อมูลชื่อ ฉาญจนา ไม่น่าจะมีอยู่จริงในฐานข้อมูล ดังนั้นความผิดพลาดจึงลดลงได้ด้วยเหตุนี้เช่นกัน

สมมุติในกรณีที่มีบุคคลที่มีชื่อคล้ายกันในลักษณะของตัวอักษรทุกตัวจริงๆ โปรแกรมก็จะทำการนำข้อมูลของทุกบุคคลที่มีชื่อคล้ายกันนั้นออกมาทั้งหมด ซึ่งก็เป็นรายชื่อที่คัดมาให้น้อยลงมากแล้วจากรายชื่อทั้งหมดในฐานข้อมูล หลังจากนั้นก็เป็นหน้าที่ของคนใช้โปรแกรมที่จะเป็นคนเลือกดูข้อมูลเหล่านั้นด้วยตัวเอง จึงพอสรุปได้ว่าความคล้ายคลึงกันของชื่ออาจทำให้เกิดข้อผิดพลาดได้บ้างแต่เป็นข้อผิดพลาดที่ยอมรับได้ คือไม่ใช้ปัญหาใหญ่มากนัก

แต่กรณีที่จะเป็นปัญหาที่สุดก็คือในกรณีที่ข้อมูลของบุคคลที่เราต้องการนั้นมีอยู่ในฐานข้อมูลจริง แต่กลับไม่ถูกค้นพบได้โดยข้อมูลภาพที่รับเข้ามา กรณีนี้จะก่อให้เกิดปัญหามากกว่ากรณีแรก ปัญหาดังที่กล่าวนี้จะเกิดขึ้นในกรณีที่รูปแบบตัวอักษรมีความแตกต่างจากรูปแบบตัวอักษรที่เก็บอยู่ในฐานข้อมูลมากเกินไป ในกรณีนี้สามารถแก้ไขได้โดยการปรับค่าการยอมรับของความเหมือนให้ต่ำลง เช่นถ้าเคยยอมรับตัวอักษรที่มีคะแนนความเหมือนมากกว่า 30 คะแนนว่าเป็นตัวอักษรที่คล้ายกัน ก็อาจลดค่าลงเหลือเป็น 20 คะแนน การแก้ดังนี้อาจทำให้ข้อข้อมูลที่เป็นข้อมูลบุคคลมีผลลัพธ์มาให้ผู้ใช้โปรแกรมเลือกมากขึ้นซึ่งเป็นข้อเสีย แต่ช่วยแก้ข้อเสียในกรณีที่ข้อมูลที่ต้องการไม่ถูกค้นพบ ซึ่งถึงว่าเป็นข้อเสียที่เป็นปัญหามากกว่า

จะพบว่าปัญหาต่างๆที่กล่าวมานั้นเกิดจากที่ความแม่นยำในเรื่องของการวิเคราะห์ลักษณะเด่นของตัวอักษรยังต่ำอยู่ ดังนั้นทางแก้ที่ดีที่สุดจึงต้องเป็นการพัฒนาโปรแกรมในส่วนของการวิเคราะห์

ลักษณะเด่นของตัวอักษรให้ดีขึ้นโดยการนำเอาวิธีหรือทฤษฎีต่างที่ให้ผลลัพท์ความแม่นยำมากกว่านี้ เพื่อมาประยุกต์ให้เข้ากับงานนี้ และเมื่อความถูกต้องอยู่ในระดับที่น่าพอใจแล้วก็สามารถนำเอาหลักการที่ได้จากโครงการนี้ไปประยุกต์ให้เกิดแอปพลิเคชันที่ก่อให้เกิดประโยชน์มากยิ่งๆขึ้นไป

สำหรับผลสรุปเรื่องคุณลักษณะทางด้านออบเจ็กต์ของ Informix Universal Server นั้น ก็พอสรุปได้ว่ามีคุณลักษณะตรงกับทฤษฎีทางด้านออบเจ็กต์อยู่บ้าง ถึงแม้จะยังไม่เต็มที่ เรื่องของการทำ encapsulation นั้นทำได้ในระดับ private เท่านั้น คือชนิดข้อมูลชนิด Opaque Data Type จะเป็นชนิดข้อมูลที่ไม่สามารถทำการเข้าถึงได้โดยตรง แต่จะต้องทำการเข้าถึงโดยผ่านทางการเรียกใช้เมธทอดเท่านั้น ส่วนเรื่องของการสืบทอดคุณสมบัตินั้น ชนิดข้อมูลที่สามารถทำการสืบทอดได้ก็คือชนิดข้อมูล Row Type สามารถทำการสืบทอดเป็น Row Type ใหม่ได้ โดยที่สามารถเพิ่มส่วน Data และ ส่วนของเมธทอดให้กับ Row Type ชนิดใหม่ได้ และสุดท้ายคุณสมบัติโพลิมอร์ฟิซึมนั้น สามารถทำได้ทุกชนิดข้อมูล คือสามารถที่จะทำฟังก์ชันที่มีชื่อเหมือนกัน แต่ต่างหน้าที่การทำงานกันได้

สรุปข้อดีและข้อเสียระหว่าง RDBMS และ ORDBMS

หลังจากได้ทดลองใช้งานระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์แล้วก็พอที่จะสรุปข้อดีและข้อเสียของระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์เมื่อเปรียบเทียบกับระบบจัดการฐานข้อมูลแบบสัมพันธ์ โดยสรุปเป็นข้อๆได้ดังนี้

ข้อดี

1. สามารถเก็บซ่อนรายละเอียดของข้อมูลได้ เนื่องจากข้อมูลที่มีชนิดเป็น Opaque Type นั่นเป็นข้อมูลที่มีคุณสมบัติของการเอนแคปซูเลชัน ซึ่งทำให้เราสามารถที่จะทำการเก็บซ่อนรายละเอียดของข้อมูลที่เป็นข้อมูลที่เราไม่ต้องการให้ผู้ใช้สามารถเข้าถึงได้โดยตรง
2. ช่วยให้การใช้งานระบบเครือข่ายเพื่อการส่งข้อมูลสามารถทำได้โดยไม่ก่อให้เกิด traffic มากเกินไป กล่าวคือชนิดข้อมูลที่มีชนิดเป็น Opaque Type นั้น เราสามารถที่จะเขียนฟังก์ชันเพื่อใช้งานกับข้อมูลชนิดนี้โดยใช้ภาษาซีได้โดยฝังไว้ที่ฝั่งของเซิร์ฟเวอร์ โดยปรกติแล้วข้อดีดังที่กล่าวในข้อนี้เป็นข้อดีของการฝังฟังก์ชันการทำงานไว้ในส่วนของเซิร์ฟเวอร์แล้วทำการประมวณผลที่ฝั่งเซิร์ฟเวอร์จนเสร็จเรียบร้อย แล้วจึงส่งเฉพาะผลลัพธ์ที่ต้องการกับมาที่ฝั่งไคล์เอนท์ การทำเช่นนี้เราสามารถทำได้โดยการเขียนฟังก์ชันโดยใช้ภาษา SPL ก็ได้ แต่เนื่องจากการทำงานบางอย่างเช่นกระบวนการในการวิเคราะห์ตัวอักษรดังที่เราได้ลองทำเป็นโปรแกรมสาธิตนั้นก็เป็นตัวอย่างหนึ่งของการทำงานที่มีความซับซ้อนในระดับที่ค้อนข้างสูง ในกรณีดังกล่าวนี้ ภาษา SQL อาจจะยังไม่มีความเหมาะสมเท่าที่ควร แต่เนื่องจากข้อมูลชนิดที่เป็น Opaque Type อนุญาติให้เราสามารถเขียนฟังก์ชันเพื่อใช้งานกับข้อมูลชนิดนี้ได้โดยที่ใช้ได้ทั้งภาษา SQL อย่างเดิม และนอกจากนั้นเราสามารถเขียนโดยใช้ภาษาซีได้อีกด้วย จึงเป็นข้อได้เปรียบในจุดนี้

3. สะดวกต่อการนำกับมาใช้ใหม่ ข้อดีข้อนี้เป็นข้อดีของหลักการออบเจ็กต์โอเรนเตทด์อยู่แล้ว คือเมื่อเรามีชนิดข้อมูลใหม่ที่มีลักษณะเป็นออบเจ็กต์ที่มีส่วนของ data และ method เราก็สามารถนำเอาชนิดข้อมูลนั้นไปใช้งานได้อย่างสะดวกโดยที่ทั้งส่วน data และ method ก็จะมองเป็นกลุ่มก้อนเดียวกันจึงง่ายต่อการนำกับมาใช้งานใหม่

ข้อเสีย

1. การสร้างออบเจ็กต์เป็นเรื่องที่ค่อนข้างยุ่งยาก กล่าวคือชนิดข้อมูลชนิด Opaque Type ซึ่งเป็นข้อมูลที่มีลักษณะคล้ายกับหลักการของออบเจ็กต์นั้น มีข้อกำหนดในการสร้างที่ค่อนข้างยุ่งยาก คือจะต้องมีการสร้างฟังก์ชันสนับสนุนต่างๆให้กับชนิดข้อมูลที่มีชนิดเป็น Opaque Type ที่เราสร้างขึ้นใหม่ด้วย และฟังก์ชันต่างๆที่กล่าวถึงก็ต้องถูกเขียนโดยภาษาซี ซึ่งอาจจะเป็นปัญหาสำหรับผู้ที่ไม่คุ้นเคยได้

ในความเป็นจริงแล้วข้อเสียของ ORDBMS นั้น ไม่ถึงกับเป็นข้อเสียของ ORDBMS โดยตรงซะทีเดียว แต่มักเป็นข้อเสียของเรื่องออบเจ็คเสียมากกว่า ดังนั้นประเด็นของการเปรียบเทียบที่เรากล่าวมาเป็นการเปรียบเทียบระหว่าง RDBMS กับ ORDBMS ซึ่งที่จริงแล้ว ORDBMS ยังสามารถใช้งานได้ในทุกกรณีที่ RDBMS สามารถทำได้ แต่ว่ามีการเพิ่มความสามารถในส่วนของออบเจ็กต์เพิ่มขึ้นมาให้ ซึ่งถือว่าเป็นทางเลือกเสียมากกว่า ขึ้นกับว่าเราจะเอามาใช้งานหรือไม่เท่านั้นเอง สรุปแล้วจากการที่ลองใช้งานดูแล้วก็ถือว่าคุณสมบัติทางด้านออบเจ็กต์ที่มีเพิ่มขึ้นมานั้นบางทีอาจไม่มีประโยชน์ใดๆเลยกับบางแอปพริเคชั่น

แอปพริเคชั่นที่มีความเหมาะสมที่จะใช้คุณสมบัติทางด้านออบเจ็กต์ ควรจะเป็นแอปพริเคชันที่ต้องการใช้งานชนิดข้อมูลที่มีความซับซ้อน อย่างเช่นโปรแกรมสาธิตที่เรายกตัวอย่างนั้นมีการเก็บข้อมูลลักษณะเด่นของตัวอักษรก็เป็นตัวอย่างหนึ่งของชนิดข้อมูลที่มีความซับซ้อน แต่สำหรับข้อมูลธรรมดาๆทั่วไปแล้วการใช้คุณสมบัติเดิมของ RDBMS ก็น่าจะมีความเหมาะสมอยู่แล้ว

โดยสรุปแล้วยังไม่มีชนิดข้อมูลใดใน Informix Universal Server ที่มีคุณสมบัติตรงกับทฤษฎีทางด้านออบเจ็กต์ทั้งหมด แต่ทำได้ในระดับหนึ่งเท่านั้น คาดว่าแนวโน้มของการพัฒนาไปสู่อนาคตคงจะทำให้ระบบการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์เป็นระบบปฏิบัติการที่ทำให้ผู้ใช้สามารถเลือกใช้คุณลักษณะดีๆของระบบฐานข้อมูลแบบสัมพันธ์และแบบเชิงวัตถุมาใช้ได้ในระบบฐานข้อมูลแบบเชิงวัตถุสัมพันธ์เพียงระบบเดียว จึงเป็นระบบฐานข้อมูลแบบที่น่าสนใจต่อการทำการพัฒนาต่อไป

ภาคผนวก ก
วิธีการใช้งานเครื่องมือช่วยในการสร้าง DataBlade Modules

เครื่องมือช่วยที่ใช้สร้าง DataBlade Modules

สำหรับ Informix Universal Server ได้ให้เครื่องมือช่วยที่ใช้ในการสร้าง DataBlade Module โดยมี 3 โปรแกรมที่จะแนะนำต่อไปนี้

BladeSmith

เราใช้โปรแกรม BladeSmith เพื่อเริ่มต้นการสร้าง DataBlade Module อย่างเช่น ใช้ระบุว่าภายใน DataBlade Module จะประกอบด้วยอะไรบ้าง และใช้ในการสร้างไฟล์ต่างๆ รวมทั้ง source code ด้วย โปรแกรม BladeSmith จะแนะแนวทางเราเกี่ยวกับการกำหนดรายละเอียดให้กับ DataBlade Module ด้วย wizard pages โปรแกรมสามารถที่จะทำงานได้หลายอย่างแบบอัตโนมัติในขั้นตอนการสร้าง object เช่นการเขียนคำสั่งของ SQL ที่จำเป็นพื้นฐานสำหรับใช้กำหนด object ใน database

การใช้ BladeSmith นั้น ให้สร้าง project ขึ้นมาหนึ่งอันแล้วทำการ add เอาส่วนต่างๆ ที่ต้องการเข้ามา ซึ่งสิ่งเหล่านั้นได้แก่

- User-defined objects, aggregates, casts, errors, interfaces, routines และ data types
- Files ที่เป็นคำสั่งของ SQL หรือไฟล์ที่จำเป็นสำหรับ client
- Imported objects เช่น built-in data types และ interfaces จาก DataBlade Module อื่น

หลังจากทำการกำหนด User-defined objects, imported object และไฟล์ต่างที่เราต้องการรวมไว้ใน DataBlade Module เสร็จแล้ว ใช้ BladeSmith ทำการสร้างไฟล์ต่างๆ ที่จะใช้ในการ compile เป็น shared object file หรือ dynamic link library และไฟล์ต่างๆที่ใช้ในการจัดการกับ DataBlade Module ใน Informix Universal Server, ฟังก์ชันที่ใช้ในการทดสอบสิ่งที่เราได้เพิ่มเข้าไปได้แก่ ฟังก์ชัน ชนิดข้อมูล เป็นต้น นอกจากนี้ยังมีไฟล์ที่ใช้สำหรับทำ packaging file และส่วนต่างๆ ดังตารางนี้

Type of Generated File	Description
Source code	เราจะใช้ไฟล์เหล่านี้เพื่อสร้าง shared object หรือ dynamic link library files ต่างๆเหล่านี้ประกอบด้วย source code files, header files และ makefiles ที่เรา compile เป็น shared object file หรือ dynamic link library

SQL script	Files เหล่านี้จะบรรจุชุดคำสั่งของภาษา SQL ที่สนับสนุน DataBlade Modules ใน database system table file เหล่านี้ประกอบ 2 scripts คือ prepare script ที่ใช้อธิบายตัว DataBlade Module และ object script ที่ใช้อธิบาย DataBlade objects
Test	File เหล่านี้ใช้เพื่อตรวจสอบความถูกต้องของ user-defined routines, opaque data type, support routines และการ casts
Packaging	เราจะใช้ไฟล์เหล่านี้ร่วมกับโปรแกรม BladePack เพื่อสร้าง installation files

ตาราง แสดงส่วนต่างๆ ที่ถูกสร้างภายใน Project

BladePack

เป็นโปรแกรมที่ใช้สำหรับสร้าง Package ที่ใช้สำหรับติดตั้ง DataBlade Module โปรแกรมนี้จะต้องการใช้ packaging file ที่ถูกสร้างจากโปรแกรม BladeSmith สำหรับ packaging file จะอ้างถึง SQL script shared object file และไฟล์อื่นๆที่จำเป็นต่อ DataBlade Module

เราสามารถใช้โปรแกรม BladePack ทำงานเหล่านี้ได้

- ทำการ add เอาไฟล์ต่างๆให้กับ DataBlade Module ยกตัวอย่างเช่น เราสามารถรวมเอา documentation, on-line help และไฟล์ตัวอย่าง เข้าไปกับ DataBlade Module ได้
- รวมเอาหลายๆ BladePack projects ไว้ใน installation package ยกตัวอย่างเช่น เราอาจรวมเอาหลายๆ DataBlade Module ที่คล้ายๆกันเข้าไว้ใน installation package อันเดียวได้
- แบ่งไฟล์ออกเป็น separate components, subcomponents และ shared components ยกตัวอย่างเช่นเราสามารถจะทำการตั้งชื่อแต่ละ components ย่อยต่างๆ แล้วให้คนที่ทำการติดตั้งสามารถที่จะเลือกได้ว่าต้องการติดตั้ง components ย่อยตัวได้บ้าง
- รวมเอา custom installation routines ไว้

นอกจากที่กล่าวมาแล้วก็ยังสามารถทำงานอื่นๆได้อีก

BladeManager

โปรแกรมนี้จะใช้เพื่อทำการ register และ unregister ตัว DataBlade Module ใน database และเพื่อ install และ uninstall DataBlade module client files

หลังจากได้ทำการติดตั้ง DataBlade Module ลงบน Server แล้วเราจะต้องทำการรีจิสเตอร์ซึ่งเป็นขั้นตอนที่ทำให้เกิดการ add เอา DataBlade สู่ database system และ เกิดการสร้าง shared object หรือ dynamic link library ของ DataBlade ให้มีขึ้นใน server

โปรแกรมจะทำการตรวจสอบความสัมพันธ์ระหว่าง DataBlade Module ถ้าในกรณีที่มีการใช้ interface ที่มาจาก DataBlade Module ตัวอื่น โปรแกรมนี้จะทำการรีจิสเตอร์ให้กับ DataBlade Module ของเราก็ต่อเมื่อมั่นใจแล้วว่า interface ที่ DataBlade Module ต้องการนั้น ได้ถูกรีจิสเตอร์ให้กับ database ไว้ก่อนแล้ว

สำหรับในกรณีที่เราจะทำการ upgrade ตัว DataBlade Module ของโปรแกรมนี้ก็จะทำการ unregister ตัว DataBlade Module ตัวเดิมออกเองโดยอัตโนมัติ

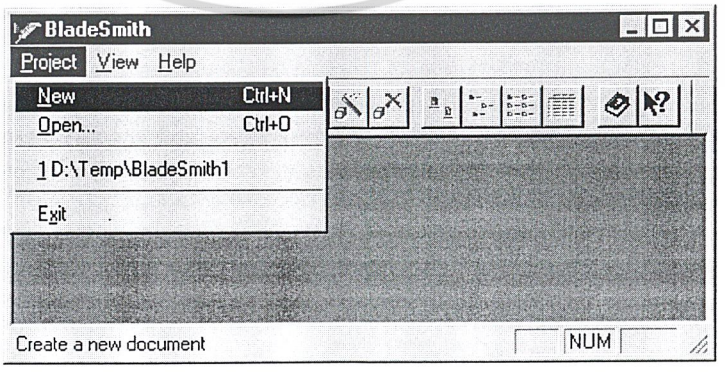
เราสามารถใช้โปรแกรมเพื่อทำการ unregister ตัว DataBlade Module ได้ โดยที่โปรแกรมจะตรวจด้วยว่า DataBlade Module ที่เราต้องการจะ unregister นั้นมีส่วนที่ DataBlade Module อื่นต้องการใช้งานหรือเปล่า ถ้าหากว่ามีโปรแกรมจะไม่ทำการ unregister เพราะจะเป็นการทำให้มีผลกระทบไปยัง DataBlade Module ตัวอื่นด้วย

การสร้าง DataBlade Modules ให้กับ database

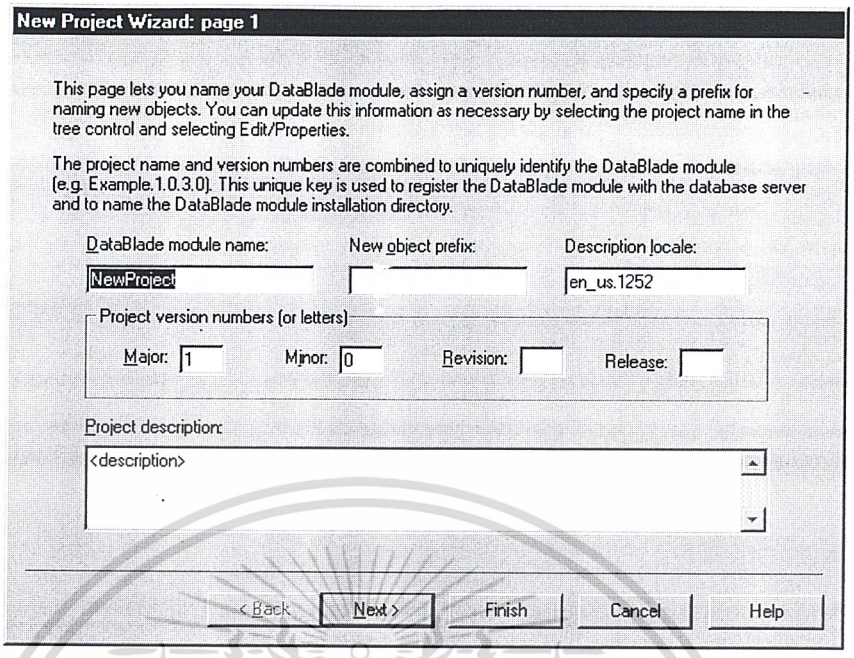
หัวข้อนี้จะอธิบายขั้นตอนและวิธีการในการสร้าง DataBlade Module โดยใช้โปรแกรม BladeSmith จากนั้นก็จะอธิบายการนำเอา DataBlade ที่สร้างโดยการเพิ่มโค้ดที่เราต้องการเสร็จแล้วมาทำเป็น package เพื่อใช้ในการติดตั้งโดยใช้โปรแกรม BladePack เป็นตัวสร้าง จากนั้นก็จะอธิบายการติดตั้ง DataBlade Module ให้กับ Server และอธิบายการใช้โปรแกรม BladeManager ในการรีจิสเตอร์ตัว DataBlade Module ให้กับ database ที่ต้องการใช้งาน DataBlade Module

การสร้าง DataBlade Module

เริ่มต้นด้วยการเรียกใช้งานโปรแกรม BladeSmith แล้วเลือกเมนู New ที่เป็นเมนูย่อยในเมนู Project ดังภาพ

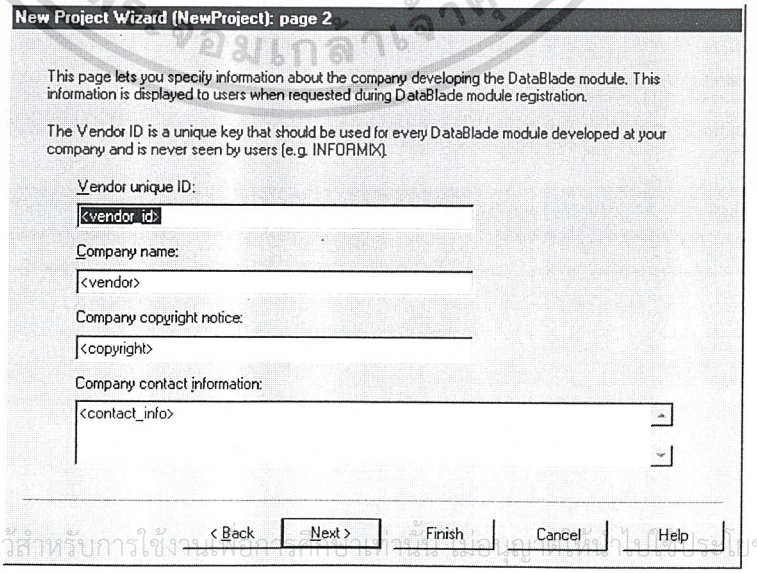


เมื่อเลือก New แล้วโปรแกรมก็จะมีแบบฟอร์มมาให้กรอกดังนี้

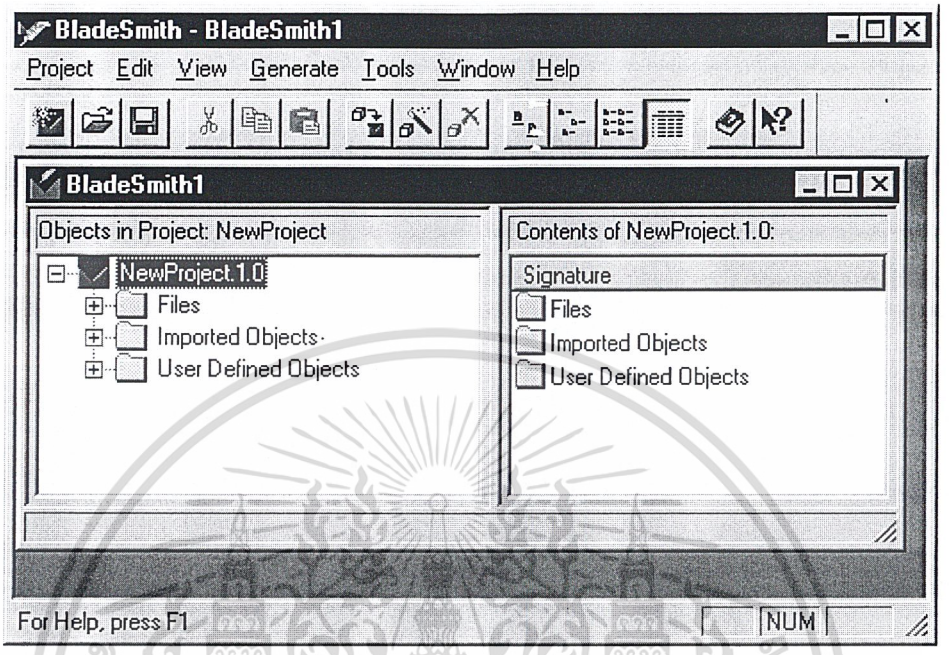


ช่อง DataBlade module name เป็นช่องมีไว้ให้ตั้งชื่อของ DataBlade Module ที่กำลังจะสร้าง ในกรณีที่ไม่ได้กรอกข้อมูลโปรแกรมก็จะมีค่า default ว่า NewProject ช่องที่ชื่อ New object prefix เป็นช่องที่โปรแกรมอนุญาติให้เราตั้งชื่อใดๆไว้ แล้วในภายหลังถ้าเราสร้างสิ่งต่างๆ เช่นชนิดข้อมูล หรืออาจจะเป็น routine ให้กับ DataBlade Module ของเรา โปรแกรมก็จะกำหนดให้มีชื่อขึ้นต้นตรงกับที่เราตั้งไว้เอง ส่วนช่อง Project description เป็นช่องที่ใช้อธิบายถึงตัว DataBlade Module ที่เรากำลังสร้างอยู่

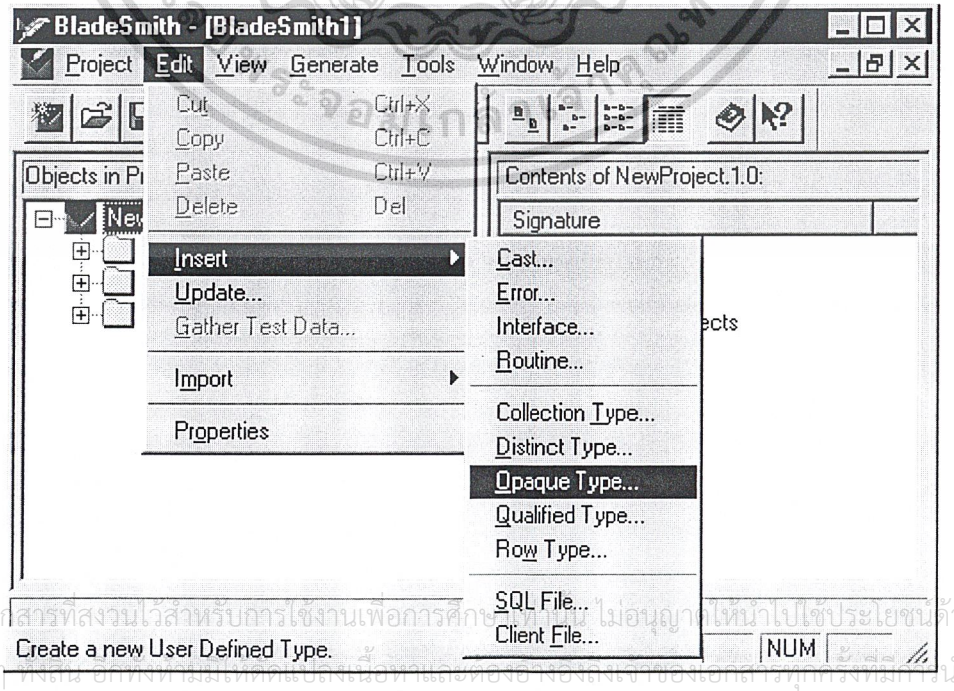
สำหรับช่อง New object prefix และช่อง Project description นั้น จะไม่กรอกก็ได้ เมื่อทำการกรอกข้อมูลเสร็จแล้วก็กด next เพื่อทำงานต่อไป



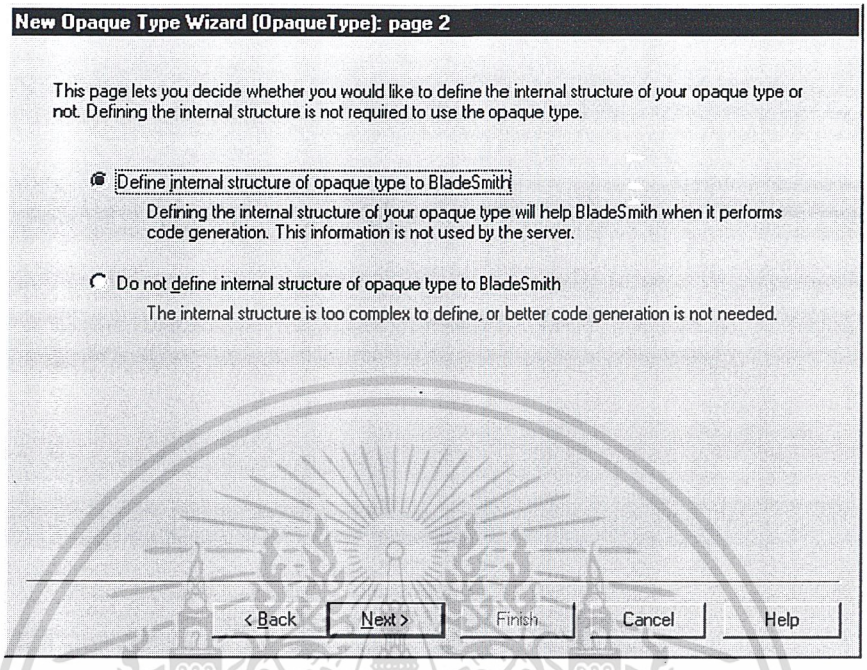
จากนั้นโปรแกรมก็จะให้กรอกข้อมูลที่เกี่ยวกับผู้สร้าง DataBlade Module ถ้าไม่ต้องการกรอกอะไรก็ได้ ให้กด next ข้ามไปเลย แล้วโปรแกรมก็จะให้เราดู script ภาษา SQL แล้วเราก็กดปุ่ม finish ได้เลย แล้วก็จะเห็นตัว project ใหม่ที่เราสร้างขึ้นดังในภาพต่อไปนี้



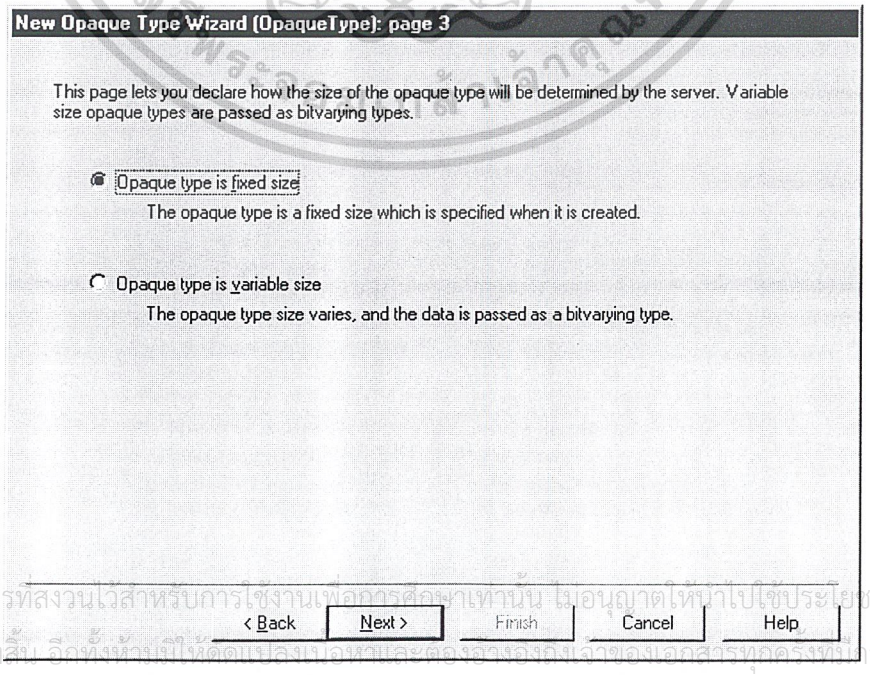
ที่นี้เราก็สามารถที่จะทำการสร้าง objects ต่างๆที่เราต้องการได้ เช่นยกตัวอย่างว่าถ้าหากเราต้องการจะสร้าง object ที่เป็น opaque type ก็ทำได้โดยกดที่เมนู Edit แล้วก็ไปเลือกเมนูย่อย Insert แล้วจะเห็นสิ่งต่างๆที่เราสามารถจะสร้างได้ ถ้าต้องการสร้าง opaque type ให้เลือกที่ opaque type ดังตัวอย่างในรูป



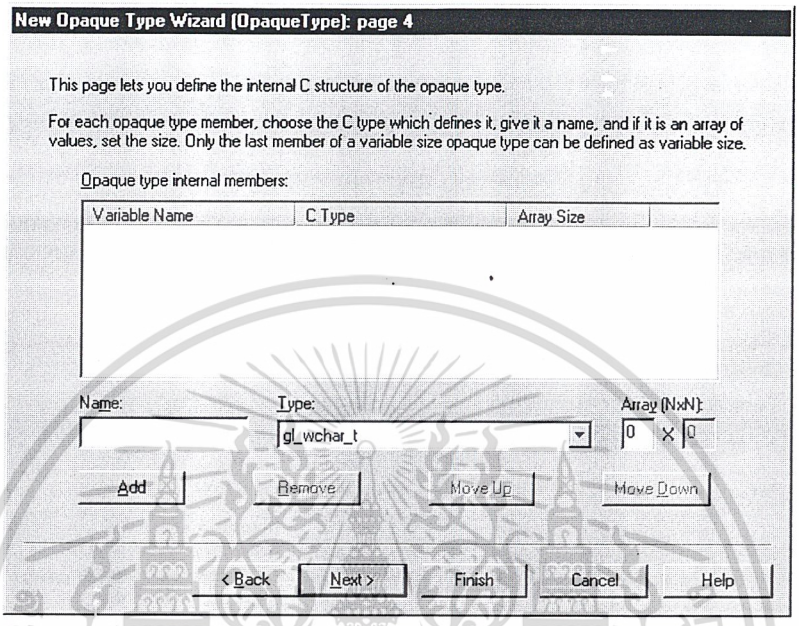
เมื่อเลือกแล้วโปรแกรมจะถามเพื่อให้ทำการตั้งชื่อของ Opaque type ที่ต้องการจะสร้าง เมื่อกรอกชื่อแล้วก็กด next เพื่อทำงานต่อ



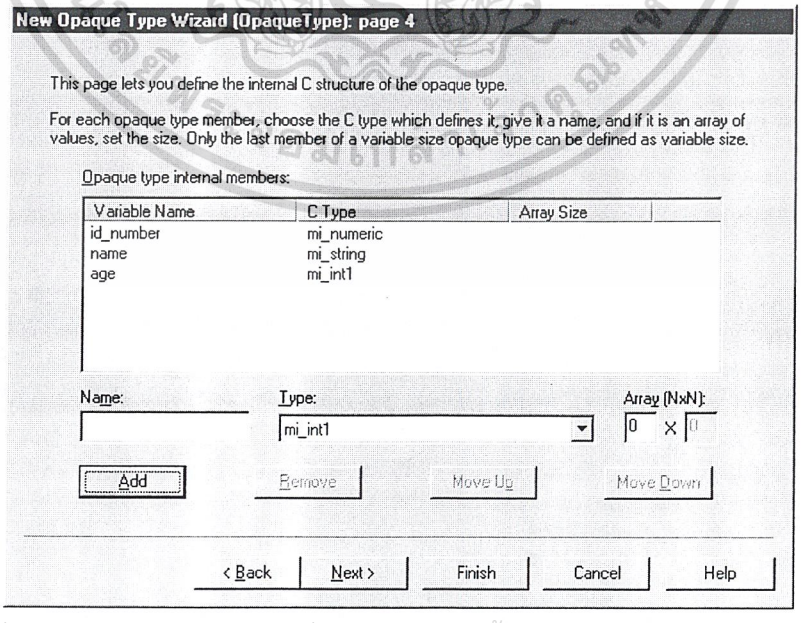
โปรแกรมจะให้เราเลือกว่าต้องการที่จะกำหนด internal structure โดยใช้โปรแกรมนี้หรือไม่ ถ้าหากต้องการก็เลือกด้านบน แต่หาก internal structure ที่ต้องการจะกำหนดนั้นมีความซับซ้อนเกินกว่าจะใช้โปรแกรมนี้กำหนดได้ ก็สามารถเลือกด้านล่าง และเราก็จะต้องไปทำการกำหนด internal structure เองภายหลัง สำหรับตัวอย่างนี้จะใช้โปรแกรม BladeSmith เป็นตัวกำหนดเลย โดยการกด Next เพื่อทำต่อไป



เนื่องจาก opaque type มีแบบที่มีขนาดแน่นอนกับแบบที่มีขนาดไม่แน่นอน จากกรอบด้านบน โปรแกรมจะให้เราเลือกว่าเราต้องการที่จะสร้าง opaque type ประเภทใด ถ้าเลือกแบบบนก็จะเป็น opaque type แบบที่มีขนาดแน่นอน แต่แบบล่างจะเป็น opaque type แบบที่ขนาดไม่แน่นอน เมื่อเลือกแบบที่ต้องการแล้วก็ให้กด next ต่อไป แล้วก็จะเข้าสู่กรอบที่ใช้ในการกำหนด internal structure ดังรูปข้างล่างนี้

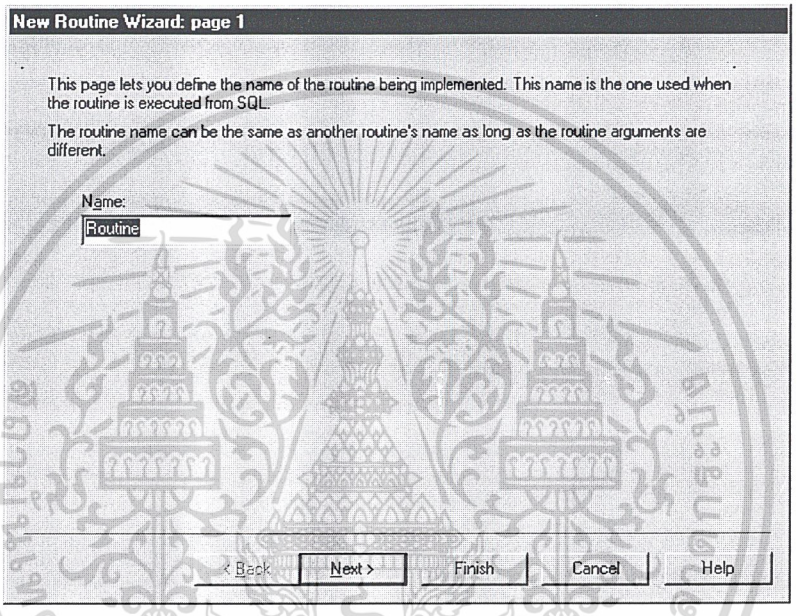


จากกรอบด้านบนเวลาที่จะเพิ่ม field ใหม่ให้กับ opaque type ทำได้โดยการกดปุ่ม Add แต่ก่อนหน้าที่จะ Add ก็ต้องทำการตั้งชื่อของ field นั้นก่อนตรงช่อง Name แล้วเลือกชนิดของข้อมูลสำหรับ field นั้น โดยกดลูกศรที่ตรง Type แล้วจะมีรายการให้เลือก สำหรับช่องที่เขียนว่า Array นั้นใช้สำหรับกำหนดชนิดข้อมูลให้เป็นแบบ Array

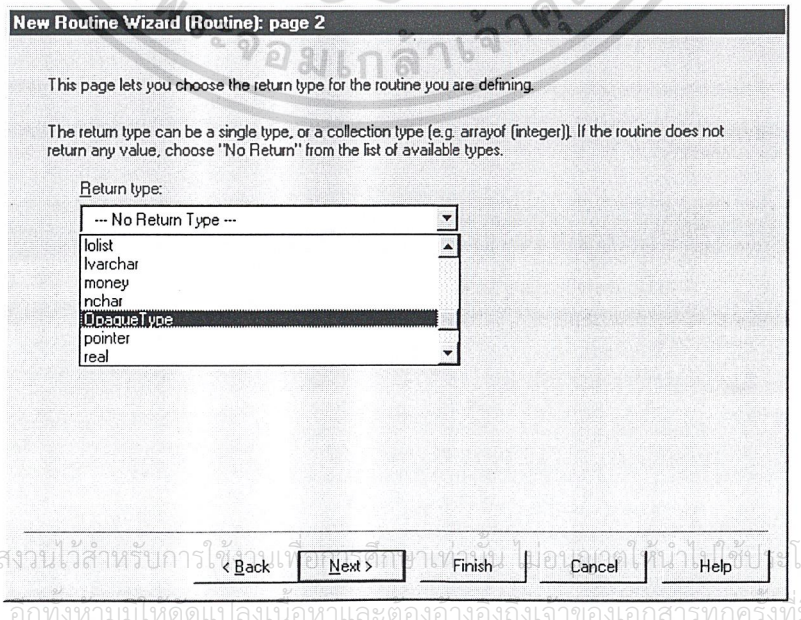


จากรูปที่แล้วเป็นตัวอย่างการกำหนด internal structure ให้มี 3 field คือ id_number มีชนิดข้อมูลเป็น mi_numeric , name มีชนิดเป็น mi_string และ age มีชนิดเป็น mi_int1 สำหรับ field ได้ที่ add ไปแล้ว แล้วต้องการจะยกเลิก field นั้น ทำได้โดยการเลือกชื่อ field นั้นแล้วกดปุ่ม Remove หรือถ้าต้องการเลื่อนขึ้นก็กดปุ่ม Move Up ถ้าต้องการเลื่อนลงกดปุ่ม Move Down เมื่อเพิ่มเติมแก้ไขจนพอใจแล้วก็กดปุ่ม Finish แล้วก็จะได้ opaque type ใหม่ขึ้นมา

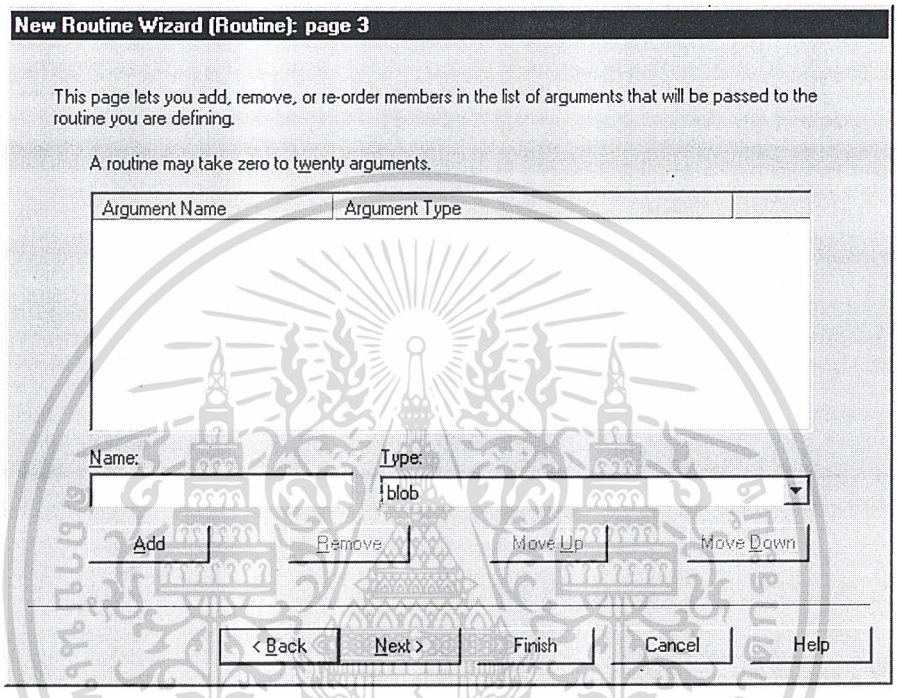
ทีนี้ก็สามารถสร้าง Routines เพื่อสนับสนุนตัว opaque ที่เราสร้างไปแล้วโดยการกดที่ Edit แล้วเลือก Insert คล้ายกับตอนที่เราจะสร้าง opaque type แต่ให้เลือกเป็น Routine แทน เมื่อเลือกแล้วก็จะมีกรอบนี้ขึ้มา



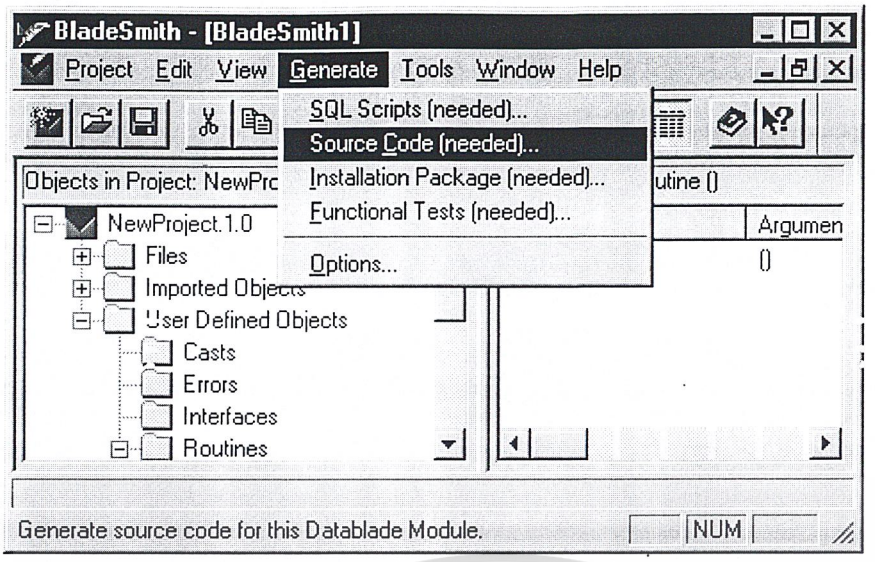
ให้ทำการป้อนชื่อของ Routine ที่ต้องการจะสร้าง แล้วกด Next เพื่อการสร้างต่อไป



กรอบนี้จะให้เราเลือกว่าต้องการให้ฟังก์ชั่นมีการส่งค่ากลับให้มีชนิดอะไร เมื่อกดปุ่มลูกศา แล้วก็จะมีชนิดข้อมูลมาให้เลือก และจะเห็นข้อมูล opaque type ที่เราได้สร้างไว้แล้วด้วย เมื่อเลือกชนิดข้อมูลเสร็จแล้ว ถ้าหาก Routine ที่เราต้องการสร้างเป็น Routine ที่ไม่มีการรับค่า argument ก็กด Finish ได้เลย แต่ถ้าฟังก์ชั่นมีการรับค่า argument ด้วยก็ให้กด next เพื่อทำการกำหนดรายละเอียดของ argument เหล่านั้น



สำหรับวิธีการกำหนด argument ทำเหมือนการกำหนด field ให้กับ opaque type ที่ได้เคยอธิบายไปแล้ว เมื่อกำหนดเรียบร้อยแล้วก็กด Finish ได้เลย แล้วพอเราทำการให้โปรแกรมสร้าง source files ต่างๆเสร็จแล้วก็เป็นหน้าที่ของเราที่ต้องไปปรับปรุงเพิ่มเติมรายละเอียดของ routine ที่ประกาศไว้เอาเอง โดยที่จะต้องเขียนเป็นภาษา C เพราะโปรแกรมจะสร้างเป็น source code ภาษา C ให้ สำหรับวิธีการที่จะสร้างไฟล์ต่างๆนั้นทำได้โดยเลือกที่เมนู Generate แล้วจะเห็นดังภาพ

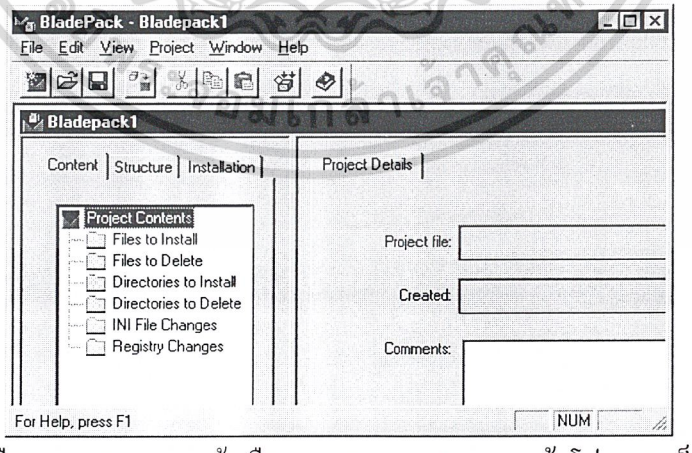


ก็ให้เลือกทีละหัวข้อจนครบ ในตอนที่เลือกอันแรกโปรแกรมจะถามให้ทำการบันทึกข้อมูลเสียก่อน ก็ทำการบันทึกแล้วก็จะได้ไฟล์ต่างที่โปรแกรมสร้างให้ก็จะอยู่ใน subdirectory ที่ชื่อ scripts, src, install และ functest ซึ่งจะอยู่ใน directory ที่เราทีการบันทึก project ไว้อีกที

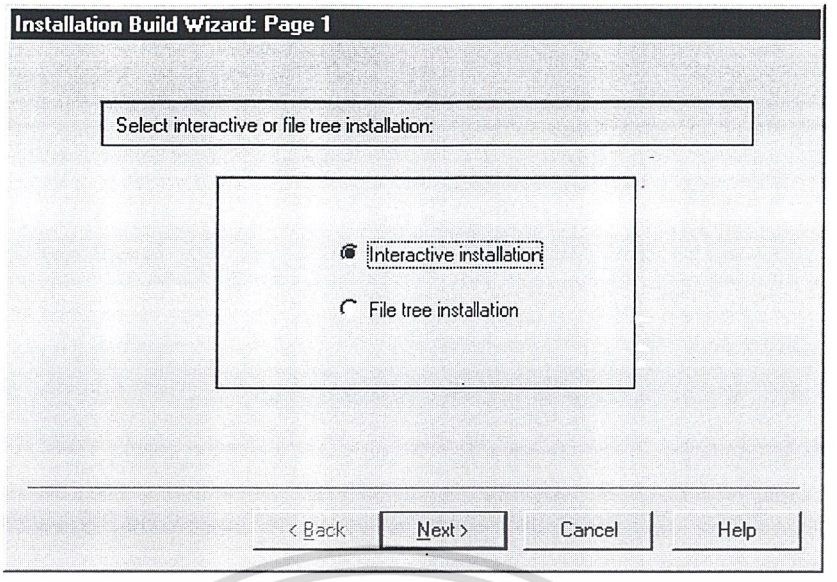
หลังจากนี้เราก็นำเอา source code ภาษา C ไปทำการ compile โดยใช้โปรแกรม Visual C เวอร์ชัน 5 โดยทำการ compile ให้เป็น library file แล้วก็เป็นอันว่าเราได้ DataBlade Module แล้วซึ่งในช่วงนี้เองที่เราจะทำการเพิ่มโค้ดที่ต้องการสำหรับ Module นี้เข้าไป

การทำ package installation

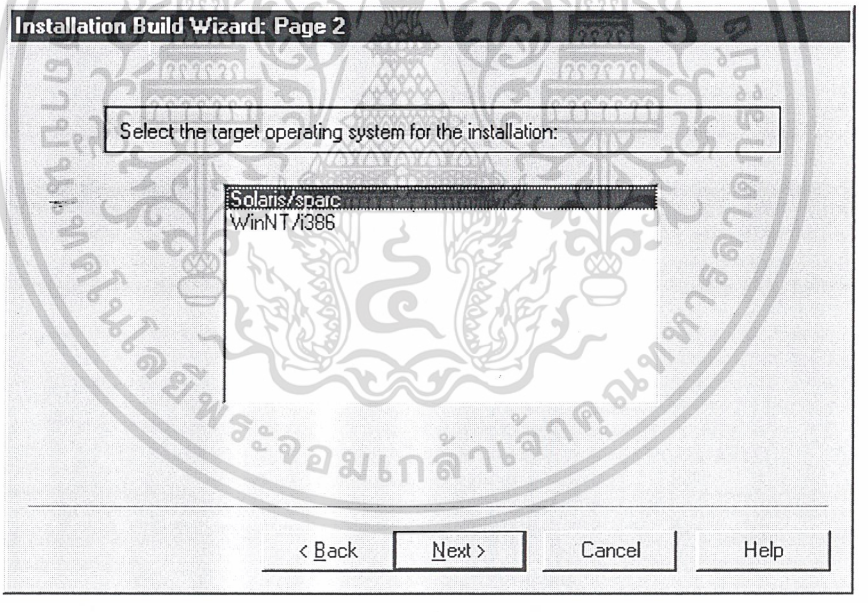
หลังจากสามารถสร้าง DataBlade Module ได้แล้ว ต่อไปก็จะสร้าง package installation โดยใช้โปรแกรม BladePack ที่สามารถเรียกใช้ได้โดยเลือกที่เมนู tools ที่อยู่ในโปรแกรม BladeSmith เมื่อเปิดโปรแกรมเสร็จแล้วจะได้ project ใหม่ดังนี้



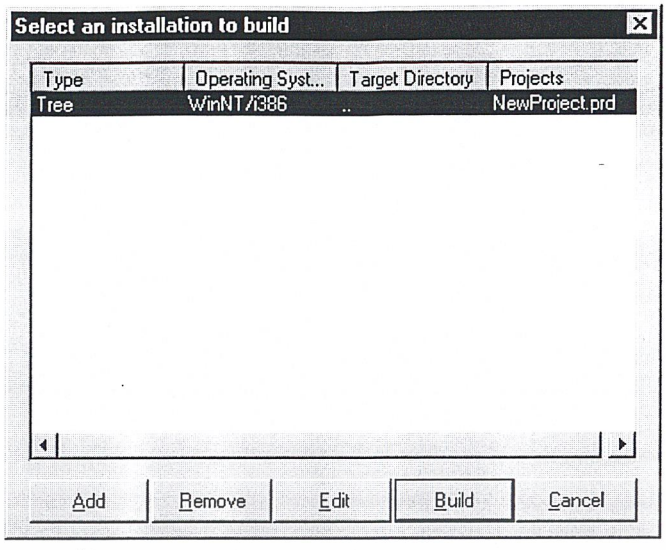
ต่อไปก็เลือกเมนู Project แล้วเลือก build installation แล้วโปรแกรมก็จะถามเพื่อให้เราบันทึก project ที่เราสร้างใหม่ใน BladePack ก็ทำการ save ไป แล้วโปรแกรมจะให้เลือกดังในภาพ



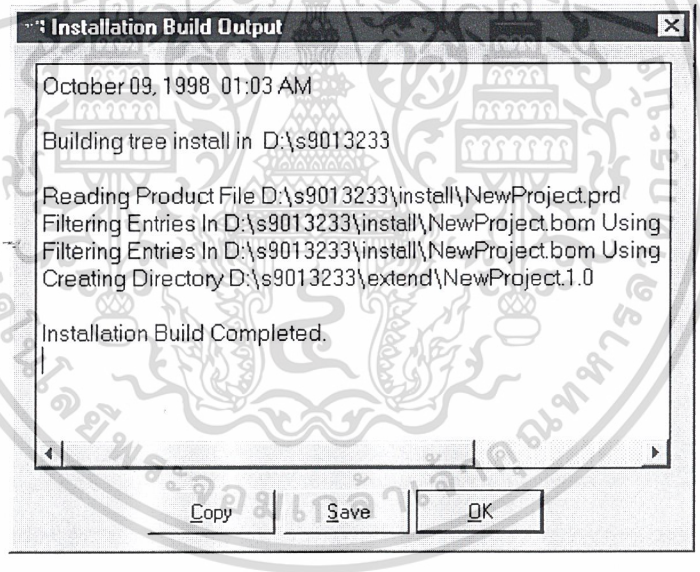
ให้เลือก File tree installation แล้วกด Next แล้วโปรแกรมก็จะให้เลือกต่อดังนี้



ก็ให้เลือก WinNT/i386 แล้วกด Next ต่อ แล้วโปรแกรมก็จะให้ป้อนชื่อของ directory ที่ต้องการจะให้ตัวติดตั้งไปอยู่ เมื่อป้อนเสร็จก็กด Next ต่อ จากนั้นก็กด Finish



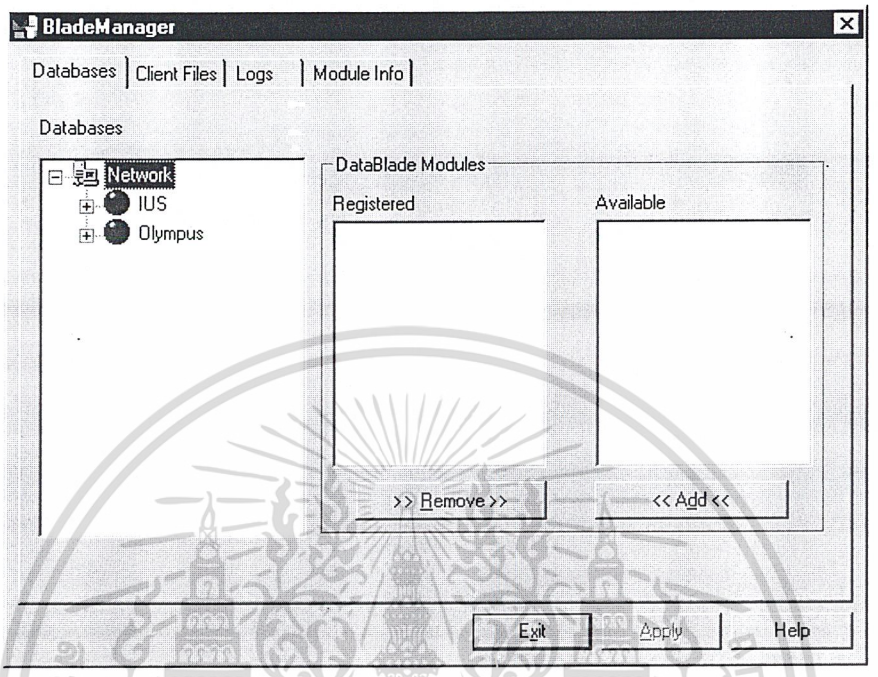
ให้กดปุ่ม Build เพื่อสร้างไฟล์ที่ต้องการ



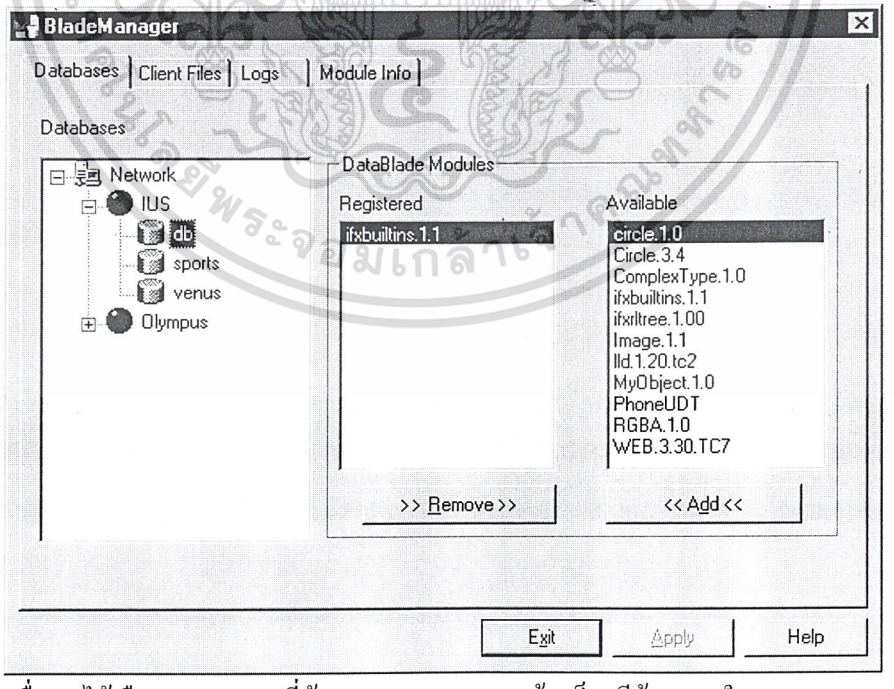
กดปุ่ม OK แล้วโปรแกรมก็จะกลับไปยังกรอบที่แล้วก็กด cancel ก็เป็นอันเสร็จสิ้นการสร้างตัวติดตั้ง DataBlade Module แล้ว ต่อไปก็ออกจากโปรแกรม BladePack ได้ จากนี้ก็ให้ copy เอาไฟล์ที่ได้จากขั้นตอนต่างๆที่ผ่านมาไปไว้ใน subdirectory ที่สร้างขึ้นใหม่โดยสร้างไว้ใน directory ของโปรแกรม informix ที่ชื่อว่า extends ก็เป็นอันเรียบร้อย

การรีจิสเตอร์ตัว DataBlade Module ให้กับ database ที่ต้องการ

สามารถทำได้โดยใช้โปรแกรม BladeManager เมื่อทำการเรียกใช้งานโปรแกรมแล้ว จะมีหน้าจอเป็นดังนี้



ให้ทำการเลือก database server ที่มี database ที่ต้องการจะทำการรีจิสเตอร์เช่นถ้าเลือก IUS โปรแกรมก็จะแสดงชื่อของ database ที่มีอยู่ใน server ที่ชื่อ IUS ดังภาพ



เมื่อเราได้เลือก database ที่ต้องการจะ resister แล้ว ก็จะมีข้อมูลมาในกรอบสองกรอบด้านขวา คือกรอบ Registered และกรอบ Available สำหรับกรอบแรกนั้นคือกรอบที่แสดงรายชื่อของ

DataBlade Module ที่ได้ทำการรีจิสเตอร์ไว้ใน database นี้แล้ว ส่วนอีกกรอบจะเป็นรายชื่อของ DataBlade Module ที่มีอยู่ใน database server ที่สามารถทำการรีจิสเตอร์ได้

เมื่อต้องการรีจิสเตอร์ก็ทำได้โดยการเลือก DataBlade Module ที่ต้องการจากกรอบ Available แล้วก็กดปุ่ม Add ที่อยู่ด้านล่างของกรอบ และเมื่อต้องการจะทำการ unregister ก็ทำได้โดยการเลือก DataBlade Module ที่ได้รีจิสเตอร์ไว้แล้วจากกรอบ Registered แล้วกดปุ่ม Remove หลังทำการรีจิสเตอร์และ unregister เรียบร้อยแล้ว ก็กดปุ่ม Apply แล้วก็ปุ่ม Exit ก็เป็นการเสร็จสิ้นขั้นตอนการทำ register และ unregister

การใช้งาน DataBlade Module

หลังจากที่เราได้สร้าง DataBlade Module แล้วเราก็จะได้ Module ที่ทำงานตามที่เราต้องการซึ่งอาจจะมีการสร้างชนิดข้อมูลใหม่ขึ้นมาโดยจะคล้ายกับ object เพราะจะมีทั้งข้อมูลและมีทั้งฟังก์ชันที่ใช้กับข้อมูลนั้น ดังนั้นลักษณะการใช้งานก็จะคล้ายกับว่าเป็น Package ที่รวมชนิดข้อมูลนั่นเอง เช่นสามารถจะนำไปสร้างตารางที่มีคอลัมน์ที่มีชนิดเป็น object ที่เราสร้างมาแล้วได้ เช่นถ้าเรามีข้อมูลชนิด mo_circle ซึ่งเป็นข้อมูลชนิด opaque ที่ใช้สำหรับเก็บข้อมูลของวงกลมโดยลักษณะโครงสร้างภายในจะประกอบด้วยจุดศูนย์กลางและรัศมี โดยได้ทำ routine ที่ใช้ในการหาพื้นที่ของวงกลมและ routine ที่ใช้ในการ return ค่าของรัศมี ซึ่งตัวอย่างต่อไปนี้จะยกตัวอย่างการสร้าง Opaque Data Type mo_circle ซึ่ง internal structure ของ mo_circle สามารถที่จะเขียนเป็นภาษา C++ ได้ดังนี้

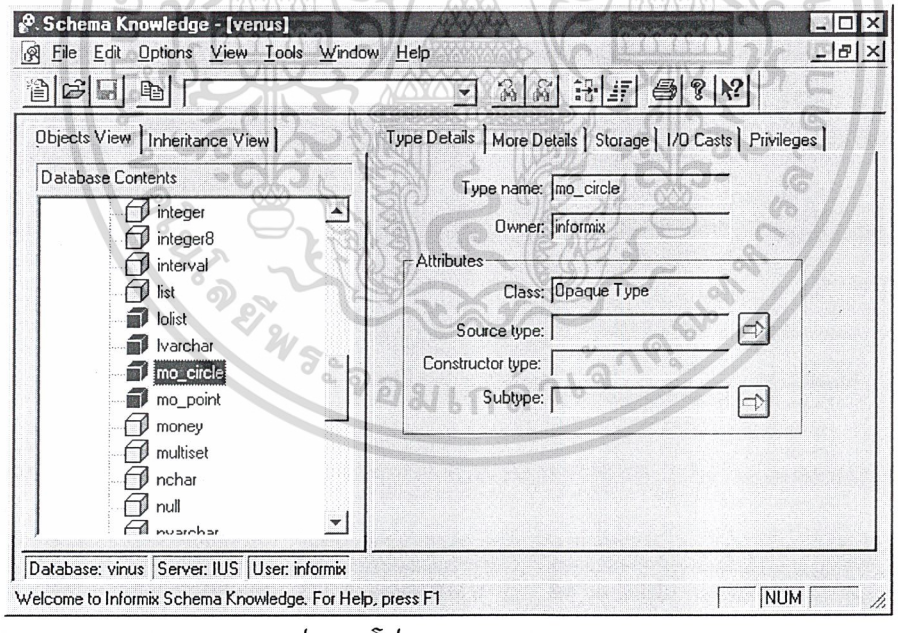
```
typedef struct {
        int x;
        int y;
} mo_point;
typedef struct{
        oint_t center;
        int radius;
} mo_circle     ;
```

ซึ่งตัว internal structure เราไม่ต้องทำการเขียน code เองแต่เราสามารถที่จะระบุใน BladeSmith ทำการ generate code ให้ซึ่งนอกจาก internal structure ที่จะต้องระบุแล้วเรายังสามารถที่จะระบุ Support function ต่างๆ ของ Opaque type ของเราเช่น function ที่ทำการแปลงจาก external structure เป็น internal structure(function input),function ที่ใช้ในการทำ compare opaque type (equal,noequal,compare ....) เป็นต้น ซึ่งตัว function เหล่านี้จะถูก generate มาให้โดย BladeSmith ส่วน function อื่นๆ เช่น radius ,area สำหรับ mo_circle เราจะต้องทำการ code เอง

เมื่อเราได้ code ภาษา C++ มาแล้วก็ทำการ compile เป็น library file จากนั้นก็ทำการรีจิสเตอร์ลง server เพื่อให้ server รู้จักกับ DataBlade Module ของเราเมื่อเราทำการรีจิสเตอร์เสร็จเราสามารถที่จะดูว่า data type ต่างๆ ที่เราได้สร้างอยู่ใน Database Server หรือยังโดยการใช้โปรแกรม Schema Knowledge ดู ซึ่งแสดงไว้ด้านล่าง

จาก Opaque type ที่ได้นี้เราสามารถที่จะนำมาใช้งานได้เหมือนกับ built-in data type ทั่วไปได้ เช่นเราทำการสร้างตารางดังนี้

```
CREATE TABLE circle_tab
(
        circle_id       SERIAL,
        color           VARCHAR(8),
        circle_col      mo_circle
);

CREATE INDEX circle_ix ON circle_tab(circle_col);
```



รูปแสดงโปรแกรม Schema Knowledge

ซึ่งเมื่อเราทำการสร้างตารางเสร็จแล้วเราก็สามารถที่จะทำการ query เหมือนกับทำกับตารางที่มี attribute ชนิดข้อมูลแบบ built-in data type ทั่วไปได้เช่น Query ในหน้าถัดไป

***Query 1:* ทำการ *list all fact* ที่มีวงกลมที่มีรัศมีและจุดศูนย์กลางเดียวกัน**

```
SELECT *
FROM circle_tab L1 ,circle_tab L2
WHERE L1.circle_col=L2.circle_col;
```

Query 2: ***ทำการ list all fact ของข้อมูลที่วงกลมรัศมีและจุดศูนย์กลางต่างกัน***

```
SELECT *
FROM circle_tab L1 ,circle_tab L2
WHERE L1.circle_col<>L2.circle_col;
```

ภาคผนวก ข

ชุดคำสั่งภาษาเบสิกของโปรแกรมสาธิต 2

```
/*---  test.frm  ---*/

VERSION 5.00
Begin VB.Form Form1
   Caption        =   "Form1"
   ClientHeight   =   4185
   ClientLeft     =   60
   ClientTop      =   345
   ClientWidth    =   7380
   LinkTopic      =   "Form1"
   ScaleHeight    =   4185
   ScaleWidth     =   7380
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox age
      Height         =   285
      Left           =   6600
      TabIndex       =   14
      Text           =   "อายุ"
      Top            =   3480
      Width          =   495
   End
   Begin VB.TextBox address_province
      Height         =   285
      Left           =   4200
      TabIndex       =   13
      Text           =   "จังหวัด"
      Top            =   3480
      Width          =   2295
   End
   Begin VB.TextBox address_road
      Height         =   285
      Left           =   1800
      TabIndex       =   12
      Text           =   "ถนน"
      Top            =   3480
      Width          =   2295
   End
   Begin VB.TextBox address_number
      Height         =   285
      Left           =   360
      TabIndex       =   11
```

```
    Text            =   "เลขที่"
    Top             =   3480
    Width           =   1335
  End
  Begin VB.TextBox last_name
    Height          =   285
    Left            =   4440
    TabIndex        =   10
    Text            =   "นามสกุล"
    Top             =   3000
    Width           =   2655
  End
  Begin VB.TextBox first_name
    Height          =   285
    Left            =   2400
    TabIndex        =   9
    Text            =   "ชื่อ"
    Top             =   3000
    Width           =   1935
  End
  Begin VB.ComboBox Combo3
    Height          =   315
    ItemData        =   "test.frx":0000
    Left            =   360
    List            =   "test.frx":0002
    TabIndex        =   8
    Text            =   "รหัสประจำตัว"
    Top             =   3000
    Width           =   1935
  End
  Begin VB.CommandButton Command4
    Caption         =   "ค้นหาข้อมูลบุคคล"
    Enabled         =   0   'False
    Height          =   375
    Left            =   4560
    TabIndex        =   7
    Top             =   2160
    Width           =   1335
  End
  Begin VB.PictureBox Picture1
    BackColor       =   &H80000009&
    FillStyle       =   0   'Solid
    Height          =   1095
    Left            =   5880
    ScaleHeight     =   1035
    ScaleWidth      =   1035
```

```
    TabIndex        =   5
    Top             =   480
    Width           =   1095
End
Begin VB.CommandButton Command3
    Caption         =   "เพิ่มชื่อภาพ"
    Enabled         =   0   'False
    Height          =   375
    Left            =   3000
    TabIndex        =   4
    Top             =   2160
    Width           =   1335
End
Begin VB.CommandButton Command2
    Caption         =   "ลบชื่อภาพ"
    Enabled         =   0   'False
    Height          =   375
    Left            =   1440
    TabIndex        =   3
    Top             =   2160
    Width           =   1335
End
Begin VB.TextBox Text2
    Height          =   285
    Left            =   120
    TabIndex        =   2
    Top             =   1680
    Width           =   5295
End
Begin VB.ListBox List1
    Height          =   1260
    ItemData        =   "test.frx":0004
    Left            =   120
    List            =   "test.frx":0006
    TabIndex        =   1
    Top             =   360
    Width           =   5295
End
Begin VB.Frame Frame1
    Caption         =   "ข้อมูลบุคคล"
    Height          =   1335
    Left            =   120
    TabIndex        =   15
    Top             =   2640
    Width           =   7215
End
```

```
    Begin VB.Label Label2
      Caption         =   "ภาพตัวอักษร"
      Height          =   255
      Left            =   6000
      TabIndex        =   6
      Top             =   1680
      Width           =   975
    End
    Begin VB.Label Label1
      Caption         =   "ชื่อไฟล์ภาพของตัวอักษรจัดลำดับให้ถูกต้องตามชื่อบุคล"
      Height          =   255
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   3735
    End
  End
  Attribute VB_Name = "Form1"
  Attribute VB_GlobalNameSpace = False
  Attribute VB_Creatable = False
  Attribute VB_PredeclaredId = True
  Attribute VB_Exposed = False
  Dim oEngine1 As ddoEngine
  Dim oProject1 As ddoProject
  Dim oDataGroup1 As ddoDataGroup
  Dim oVTable1 As ddoTable
  Dim oModel1 As ddoModel

  Private Sub Combo3_Click()
    Dim str1 As String
    str1 = "execute function getinfo(""" + Combo3.Text + """)"
    Set oVTable1 = oDataGroup1.ExecuteSQLCommand(str1, "oVTable1")
    oVTable1.FirstRecord
    first_name.Text = oVTable1.Columns(1).Value
    last_name.Text = oVTable1.Columns(2).Value
    address_number.Text = oVTable1.Columns(3).Value
    address_road.Text = oVTable1.Columns(4).Value
    address_province.Text = oVTable1.Columns(5).Value
    age.Text = oVTable1.Columns(6).Value
    oDataGroup1.DeleteVirtualTable ("oVTable1")
  End Sub

  Private Sub Command2_Click()
    Dim n
    If List1.ListCount > 0 Then
      n = List1.ListIndex
```

```
        List1.RemoveItem (List1.ListIndex)
        If List1.ListCount > 0 And n = 0 Then
            List1.ListIndex = 0
        Else: List1.ListIndex = n - 1
        End If
    End If
    If List1.ListCount < 1 Then
        Command2.Enabled = False
        Command4.Enabled = False
        Picture1.Picture = Nothing
    End If
End Sub

Private Sub Command3_Click()
    If Text2.Text <> "" And Dir(Text2.Text) Like "*.bmp" Then
        List1.AddItem Text2.Text
        List1.ListIndex = List1.ListCount - 1
        Picture1.Picture = LoadPicture(Text2.Text)
        Command2.Enabled = True
        Command4.Enabled = True
    End If
End Sub

Private Sub Command4_Click()
    Dim i, n, m As Integer
    Dim str1 As String
str1 = "delete from temp_table where position<100"
    oDataGroup1.ExecuteSQLCommand (str1)
    For i = 0 To List1.ListCount - 1
        List1.ListIndex = i
        str1 = "insert into temp_table values("
        str1 = str1 + Str(i + 1)
        str1 = str1 + ", """ + List1.Text + """)"
        oDataGroup1.ExecuteSQLCommand (str1)
    Next i
    str1 = "execute function persons()"
Set oVTable1 = oDataGroup1.ExecuteSQLCommand(str1, "oVTable1")
If Combo3.ListCount <> 0 Then Combo3.Clear
    oVTable1.FirstRecord
    While Not oVTable1.EOT
        Combo3.AddItem (oVTable1.Columns(1).Value)
        oVTable1.NextRecord
    Wend
    oDataGroup1.DeleteVirtualTable ("oVTable1")
    Combo3.ListIndex = 0
End Sub
```

```
Private Sub Form_Load()
    Set oEngine1 = New ddoEngine
    Set oProject1 = oEngine1.CreateProject
    oProject1.Name = "OR_DBMS"
    Set oModel1 = oEngine1.CreateModel("d:\data\vb\model1.mlt")
    Set oDataGroup1 = oProject1.CreateDataGroup("ascii_value", "ascii_value", "d:\data\vb\model1.mlt")
    oDataGroup1.Logon "informix", "inform123"
End Sub

Private Sub Form_Unload(Cancel As Integer)
    oDataGroup1.Logoff
End Sub

Private Sub List1_Click()
    Picture1.Picture = LoadPicture(List1.Text)
End Sub

Private Sub Text2_Change()
    If Text2.Text = "" Then
        Command3.Enabled = False
    Else: Command3.Enabled = True
    End If
End Sub
```

ภาคผนวก ค

ชุดคำสั่งภาษาซีของโปรแกรมสาธิต 2

```
/*---  ไฟล์ Ocr.h  ---*/

#ifndef HDR_Ocr_H
#define HDR_Ocr_H

#ifndef TRACE_DEBUG_Ocr
#define TRACE_DEBUG_Ocr 1
#endif

#ifndef DBDK_LOHSIZE
#define DBDK_LOHSIZE                    sizeof(MI_LO_HANDLE)
#define DBDK_LOBINFNSIZE DBDK_LOHSIZE
#endif

/* This data structure returned by LOhandles. */
typedef struct
{
        mi_integer      nlos; /* Number of large object handles. */
        MI_LO_HANDLE    los[1];      /* Valid large object handles.     */
} MI_LO_HANDLES;

#define LO_FN_MASK          "????????.lo"

#define ERRORMESG1          "UGEN1"
#define ERRORMESG2          "UGEN2"
#define ERRORMESG3          "UGEN3"
#define ERRORMESG4          "UGEN4"
#define ERRORMESG5          "UGEN5"
#define ERRORMESG6          "UGEN6"
#define ERRORMESG7          "UGEN7"
#define ERRORMESG8          "UGEN8"
#define ERRORMESG9          "UGEN9"
#define ERRORMESG10         "UGENA"
#define ERRORMESG11         "UGENB"
#define ERRORMESG12         "UGENC"
#define ERRORMESG13         "UGEND"
#define ERRORMESG14         "UGENE"
#define ERRORMESG15         "UGENF"
#define ERRORMESG16         "UGENG"
#define ERRORMESG17         "UGENH"
#define ERRORMESG18         "UGENI"
```

```
#define ERRORMESG19       "UGENJ"

/* Use DBDK_TRACE to direct trace messages to the trace file. */
#if TRACE_DEBUG_Ocr
#define DBDK_TRACE                  (1 << 16)
#else
#define DBDK_TRACE                  0
#endif

#define DBDK_TRACE_ERROR( Caller, ErrNo, ErrLevel )          \
                    Gen_Trace                                \
                    (                                        \
                       Gen_Con,                              \
                       Caller,                               \
                       __FILE__,                             \
                       __LINE__,                             \
                       ErrNo,                                \
                       "Ocr",                                \
                       ErrLevel,                             \
                       MI_SQL | DBDK_TRACE                   \
                    );

/* Print a message to the trace file. */
#if TRACE_DEBUG_Ocr

#define DBDK_TRACE_MSG( Caller, ErrNo, ErrLevel )            \
                    Gen_Trace                                \
                    (                                        \
                       Gen_Con,                              \
                       Caller,                               \
                       __FILE__,                             \
                       __LINE__,                             \
                       ErrNo,                                \
                       "Ocr",                                \
                       ErrLevel,                             \
                       DBDK_TRACE                            \
                    );

#else

#define DBDK_TRACE_MSG( Caller, ErrNo, ErrLevel )

#endif

/* These macros are used on entry to, and on exit from, a function. */
#define DBDK_TRACE_ENTER( Caller )  DBDK_TRACE_MSG( Caller, ERRORMESG13, 20 )
```

```
#define DBDK_TRACE_EXIT( Caller )   DBDK_TRACE_MSG( Caller, ERRORMESG14, 20 )
#define YEAR_TO_MONTH   1
#define DAY_TO_SECOND   2

/* Function prototypes. */
mi_integer Gen_nstrwords( gl_mchar_t *, mi_integer );
gl_mchar_t * Gen_sscanf
(
        MI_CONNECTION *         Gen_Con,
        char *                  Gen_Caller,
        gl_mchar_t *            Gen_InData,
        mi_integer              Gen_InDataLen,
        mi_integer              Gen_Width,
        char *                  Gen_Format,
        char *                  Gen_Result
);
void    Gen_LoadLOFromFile
(
        MI_CONNECTION *         Gen_Con,
        char *                  Gen_Caller,
        char *                  Gen_LOFile,
        MI_LO_HANDLE *          Gen_pLOh
);
void Gen_StoreLOToFile
(
        MI_CONNECTION *         Gen_Con,
        char *                  Gen_Caller,
        char *                  Gen_LOFile,
        MI_LO_HANDLE *          Gen_pLOh
);
void Gen_Trace
(
        MI_CONNECTION *         Gen_Con,
        char *                  Gen_Caller,
        char *                  Gen_FileName,
        mi_integer              Gen_LineNo,
        char *                  Gen_MsgNo,
        char *                  Gen_Class,
        mi_integer              Gen_Threshold,
        mi_integer              Gen_MsgType
);


typedef struct
{
    mi_char                     P[6][5];
```

```
    mi_char1                    pad1[2];
    MI_LO_HANDLE                ImgBuffer;
}
OcrType;
/* Warning: Do not modify. OcrType checksum: 0 */

/* extends my program */

typedef struct
 {
   mi_char *data;
   mi_integer width, height;
 } ImgBufferType;

void put(int X, int Y, char Data, ImgBufferType *buffer);
char get(int X, int Y, ImgBufferType *buffer);
void LoadLoToBuffer(MI_CONNECTION *Gen_Con, OcrType *Ocr, ImgBufferType *buffer);
void firstStep(MI_CONNECTION *Gen_Con, ImgBufferType *buffer);
void secondStep(MI_CONNECTION *Gen_Con, ImgBufferType *buffer);
void thirdStep(ImgBufferType *buffer);
void findConcen(OcrType *Ocr, int m, int x, int y, ImgBufferType *buffer);
void finalStep(ImgBufferType *buffer, OcrType *Ocr);
OcrType * Recorgnize(MI_CONNECTION *Gen_Con, OcrType *Ocr);

/* end of extends my program */

#endif
```

```
/*---  ไฟล์ OcrType.c  ---*/

#ifdef __cplusplus

extern "C"

{

#endif

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <windows.h>

#include <ifxgls.h>

#include <mi.h>

#include "Ocr.h"

OcrType *
OcrTypeAssign
(
OcrType *               Gen_param1,     /* The UDT value.                  */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.      */
        OcrType *               Gen_InData;     /* The input UDT value.         */

        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ) {
                DBDK_TRACE_ERROR( "OcrTypeAssign", ERRORMESG1, 10 );
        }

        DBDK_TRACE_ENTER( "OcrTypeAssign" );

        /* Point to the input data. */
        Gen_InData = Gen_param1;

        /* If the LO is valid ... */
```

```
        if( mi_lo_validate( Gen_Con, &Gen_InData->ImgBuffer ) == 0 ){
                /* ... increment its reference count. */
                mi_lo_increfcount( Gen_Con, &Gen_InData->ImgBuffer );
        }

        DBDK_TRACE_EXIT( "OcrTypeAssign" );
        /* Close the connection. */
        mi_close( Gen_Con );
        return Gen_InData;
}

void
OcrTypeDestroy
(
OcrType *               Gen_param1,     /* The UDT value.               */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.   */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.      */
        OcrType *               Gen_InData;     /* The input UDT value.         */

        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeDestroy", ERRORMESG1, 10 );
        }
        DBDK_TRACE_ENTER( "OcrTypeDestroy" );

        /* Point to the input data. */
        Gen_InData = Gen_param1;

        /* If the LO is valid ... */
        if( mi_lo_validate( Gen_Con, &Gen_InData->ImgBuffer ) == 0 ){
                /* ... decrement its reference count. */
                mi_lo_decrefcount( Gen_Con, &Gen_InData->ImgBuffer );
        }

        DBDK_TRACE_EXIT( "OcrTypeDestroy" );

        /* Close the connection. */
        mi_close( Gen_Con );
}
```

```
mi_sendrecv *
OcrTypeSend
(
OcrType *              Gen_param1,      /* The UDT value                    */
MI_FPARAM *            Gen_fparam       /* Standard info - see DBDK docs.   */
)
{
        MI_CONNECTION *        Gen_Con;        /* The current connection.          */
        OcrType *      Gen_InData;     /* Pointer to the UDT value.        */
        OcrType *      Gen_OutData;    /* Pointer to the packet data.      */
        mi_sendrecv *  Gen_RetVal;     /* The return value.                */
        mi_integer             Gen_Index10; /* Array iterator */
        char                   Gen_LOFile[FILENAME_MAX];/* Store the file name here.  */


        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeSend", ERRORMESG1, 10 );
        }

        DBDK_TRACE_ENTER( "OcrTypeSend" );

        /* Point to the input data. */
        Gen_InData = Gen_param1;

        /* Allocate a new return value. */
        Gen_RetVal = (mi_sendrecv *)mi_new_var( sizeof( OcrType ) );
        if( Gen_RetVal == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeSend", ERRORMESG2, 10 );
        }

        /* Point to the output data. */
        Gen_OutData = (OcrType *)mi_get_vardata( (mi_lvarchar *)Gen_RetVal );

        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer             Gen_Index20;            /* Array Index.                     */

                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                        /* Prepare the value for Gen_OutData->P[Gen_Index10][Gen_Index20]. */
                        mi_put_bytes( (mi_unsigned_char1 *)&Gen_OutData->P[Gen_Index10][Gen_Index20],
                        (char *)&Gen_InData->P[Gen_Index10][Gen_Index20], 1 );
```

```
}
}
        /* Prepare the value for Gen_OutData->ImgBuffer. */
        mi_put_bytes( (mi_unsigned_char1 *)&Gen_OutData->ImgBuffer, (char *) &Gen_InData->ImgBuffer,
DBDK_LOHSIZE );

        DBDK_TRACE_EXIT( "OcrTypeSend" );
        /* Close the connection. */
        mi_close( Gen_Con );

        /* Return the UDT for transmission. */
        return Gen_RetVal;
}

OcrType *
OcrTypeReceive
(
mi_sendrecv *           Gen_param1,     /* The UDT value.              */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.    */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.      */
        OcrType *               Gen_RetVal;     /* The return value.            */
        OcrType *               Gen_InData;     /* Packet data.                 */
        OcrType *               Gen_OutData;    /* Output UDT value.            */
        mi_integer              Gen_Index10; /* Array iterator */
        char                    Gen_LOFile[FILENAME_MAX];/* Store the file name here.   */

        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeReceive", ERRORMESG1, 10 );
        }

        DBDK_TRACE_ENTER( "OcrTypeReceive" );
        /* Point to the input data. */
        Gen_InData = (OcrType *)mi_get_vardata( (mi_lvarchar *)Gen_param1 );

        /* Allocate room for the UDT. */
        Gen_RetVal = (OcrType *)mi_alloc( sizeof( OcrType ) );
        if( Gen_RetVal == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeReceive", ERRORMESG2, 10 );
        }
```

```
        /* Point to the output data. */
        Gen_OutData = (OcrType *)Gen_RetVal;

        /* Copy the attribute value(s) from the transmission parcel. */

        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer          Gen_Index20;            /* Array Index.          */

                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                /* Prepare the value for Gen_OutData->P[Gen_Index10][Gen_Index20]. */
                mi_get_bytes( (mi_unsigned_char1 *)&Gen_InData->P[Gen_Index10][Gen_Index20], (char
                *)&Gen_OutData->P[Gen_Index10][Gen_Index20], 1 );
                }
        }
        /* Prepare the value for Gen_OutData->ImgBuffer. */
        mi_get_bytes( (mi_unsigned_char1 *)&Gen_InData->ImgBuffer, (char *)&Gen_OutData->ImgBuffer,
        DBDK_LOHSIZE );

        DBDK_TRACE_EXIT( "OcrTypeReceive" );

        /* Return the transmitted UDT value. */
        return Gen_RetVal;
}

OcrType *
OcrTypeImportBinary
(
mi_bitvarying *         Gen_param1,     /* The input value.          */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.    */
)
{
        MI_CONNECTION *     Gen_Con;        /* The current connection.       */
        OcrType *           Gen_RetVal;     /* The return result.            */
        OcrType *           Gen_InData;     /* The UDT input value.          */
        OcrType *           Gen_OutData;    /* The transfer data.            */
        mi_integer          Gen_Index10; /* Array iterator */
        char                Gen_LOFile[FILENAME_MAX];/* Store the file name here.   */

        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );
        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeImportBinary", ERRORMESG1, 10 );
        }
```

```
        DBDK_TRACE_ENTER( "OcrTypeImportBinary" );

        /* Point to the input data. */
        Gen_InData = (OcrType *)mi_get_vardata( (mi_lvarchar *)Gen_param1 );

        /* Allocate a new UDT for the return result. */
        Gen_RetVal = (OcrType *)mi_alloc( sizeof( OcrType ) );
        if( Gen_RetVal == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeImportBinary", ERRORMESG2, 10 );
        }

        Gen_OutData = (OcrType *)Gen_RetVal;

        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer          Gen_Index20;            /* Array Index.          */
                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                /* Prepare the value for Gen_OutData->P[Gen_Index10][Gen_Index20]. */
                mi_get_bytes( (mi_unsigned_char1 *)&Gen_InData->P[Gen_Index10][Gen_Index20], (char
                *)&Gen_OutData->P[Gen_Index10][Gen_Index20], 1 );
        }
        }
        /* Prepare the value for Gen_OutData->ImgBuffer. */
        mi_get_bytes( (mi_unsigned_char1 *)&Gen_InData->ImgBuffer, (char *) &Gen_OutData->ImgBuffer,
        DBDK_LOHSIZE );
        DBDK_TRACE_EXIT( "OcrTypeImportBinary" );
        /* Close the connection. */
        mi_close( Gen_Con );
        return Gen_RetVal;
}

mi_bitvarying *
OcrTypeExportBinary
(
OcrType *               Gen_param1,     /* The UDT value.              */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.      */
        mi_bitvarying *         Gen_RetVal;     /* The return value.            */
        mi_integer              Gen_Index10; /* Array iterator */
        OcrType *               Gen_InData;     /* The transfer data.           */
        OcrType *               Gen_OutData;    /* The output data.             */
        char                    Gen_LOFile[FILENAME_MAX];/* Store the file name here.  */
```

```
        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 ) {
                DBDK_TRACE_ERROR( "OcrTypeExportBinary", ERRORMESG1, 10 );
        }

        DBDK_TRACE_ENTER( "OcrTypeExportBinary" );

        /* Point to the input data. */
        Gen_InData = Gen_param1;
        /* Allocate the output parcel. */
        Gen_RetVal = (mi_bitvarying *)mi_new_var( sizeof( OcrType ) );
        if( Gen_RetVal == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeExportBinary", ERRORMESG2, 10 );
        }
        Gen_OutData = (OcrType *)mi_get_vardata( (mi_lvarchar *)Gen_RetVal );

        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer      Gen_Index20;            /* Array Index.                 */
                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                /* Prepare the value for Gen_OutData->P[Gen_Index10][Gen_Index20]. */
                mi_put_bytes( (mi_unsigned_char1 *)&Gen_OutData->P[Gen_Index10][Gen_Index20], (char
                *)&Gen_InData->P[Gen_Index10][Gen_Index20], 1 );
        }
        }
        /* Prepare the value for Gen_OutData->ImgBuffer. */
        mi_put_bytes( (mi_unsigned_char1 *)&Gen_OutData->ImgBuffer, (char *) &Gen_InData->ImgBuffer,
        DBDK_LOHSIZE );
        DBDK_TRACE_EXIT( "OcrTypeExportBinary" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

mi_integer
OcrTypeCompare
(
OcrType *               Gen_param1,     /* The first UDT value to compare.    */
OcrType *               Gen_param2,     /* The second UDT value to compare.   */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.     */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.          */
        mi_integer              Gen_cc;         /* Numeric attribute difference.    */
        mi_integer              Gen_Index10; /* Array iterator */
```

```
        OcrType *               Gen_Value1;     /* Pointer to the first value.      */
        OcrType *               Gen_Value2;     /* Pointer to the second value.     */

        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeCompare", ERRORMESG1, 10 );
        }
        DBDK_TRACE_ENTER( "OcrTypeCompare" );
        /* Point to the data values that are to be compared. */
        Gen_Value1 = Gen_param1;
        Gen_Value2 = Gen_param2;
        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer          Gen_Index20;            /* Array Index.             */
                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                        Gen_cc = Gen_Value1->P[Gen_Index10][Gen_Index20] - Gen_Value2->P
                        [Gen_Index10][Gen_Index20];
                        if( Gen_cc ){
                                return Gen_cc < 0 ? -1 : 1;
                        }
        }
        }
        Gen_cc = mi_lo_ptr_cmp( Gen_Con, &Gen_Value1->ImgBuffer, &Gen_Value2->ImgBuffer );
        if( Gen_cc ){
                return Gen_cc == 0 ? 0 : 1;
        }
        DBDK_TRACE_EXIT( "OcrTypeCompare" );
        mi_close( Gen_Con );
        return 0;
}

mi_boolean
OcrTypeEqual
(
OcrType *               Gen_param1,     /* The first UDT value to compare.  */
OcrType *               Gen_param2,     /* The second UDT value to compare. */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.   */
)
{
        int i, j, count;
        OcrType * opaque;
        OcrType * opaque2;
        opaque = Gen_param1;
        opaque2= Gen_param2;
```

```
        for(count=i=0; i<6; i++)
                for(j=0; j<5; j++)
                        if(opaque->P[i][j]==opaque2->P[i][j]) count+=2;
                        else if( (opaque->P[i][j]=='S' && opaque2->P[i][j]=='I') ||
                                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='S') ||
                                                (opaque->P[i][j]=='V' && opaque2->P[i][j]=='I') ||
                                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='V') ||
                                                (opaque->P[i][j]=='H' && opaque2->P[i][j]=='I') ||
                                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='H')
                                        ) count++;

        return (count>=20)? 1 : 0;
}

mi_boolean
OcrTypeNotEqual
(
OcrType *               Gen_param1,     /* The first UDT value to compare.   */
OcrType *               Gen_param2,     /* The second UDT value to compare.  */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.    */
)
{
        return (mi_boolean)(0 != OcrTypeCompare( Gen_param1,
                        Gen_param2, Gen_fparam ));
}

mi_bitvarying *
OcrTypeLOhandles
(
OcrType *               Gen_param1,     /* The UDT value.                    */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.    */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.       */
        OcrType *               Gen_InData;     /* The UDT value.                */
        MI_LO_HANDLES *         Gen_OutData;    /* Point to the output table.    */
        mi_integer              Gen_nLOs;       /* Number of LOs in UDT.         */
        mi_integer              Gen_LOno;       /* Index into the table.         */
        mi_bitvarying *         Gen_RetVal;     /* "Array" of LO handles.        */

        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeLOhandles", ERRORMESG1, 10 );
        }
        DBDK_TRACE_ENTER( "OcrTypeLOhandles" );
```

```
        /* Point to the input data. */
        Gen_InData = Gen_param1;
        Gen_LOno = Gen_nLOs = 0;

        if( mi_lo_validate( Gen_Con, &Gen_InData->ImgBuffer ) == 0 ){
                /* Add it to the count. */
                Gen_nLOs++;
        }

        if( Gen_nLOs ){
                Gen_RetVal = (mi_bitvarying *)mi_new_var(
                        sizeof(mi_integer) + Gen_nLOs * DBDK_LOHSIZE );
                if( Gen_RetVal == 0 ){
                        DBDK_TRACE_ERROR( "OcrTypeLOhandles", ERRORMESG2, 10 );
                }
                /* Point to the output LO table. */
                Gen_OutData = (MI_LO_HANDLES *)mi_get_vardata( (mi_lvarchar *)Gen_RetVal );
                /* If the LO is valid ... */
                if( mi_lo_validate( Gen_Con, &Gen_InData->ImgBuffer ) == 0 )
                {
                        /* Save it. */
                        memcpy( &Gen_OutData->los[Gen_LOno++],
                                &Gen_InData->ImgBuffer, DBDK_LOHSIZE );
                }
                /* Save the number of LOs in the return structure. */
                Gen_OutData->nlos = Gen_LOno;
                DBDK_TRACE_EXIT( "OcrTypeLOhandles" );
                mi_close( Gen_Con );
                return Gen_RetVal;
        }
        else {
                DBDK_TRACE_EXIT( "OcrTypeLOhandles" );
                mi_close( Gen_Con );
                /* Return failure. */
                return 0;
        }
}

OcrType *
OcrTypeImportText
(
mi_impexp *             Gen_param1,             /* The import text.             */
MI_FPARAM *             Gen_fparam              /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.      */
```

```
gl_mchar_t *        Gen_InData;             /* Pointer to the input data.       */
gl_mchar_t *        Gen_StartInData;        /* First value of Gen_InData.       */
OcrType *           Gen_OutData;            /* Pointer to the output data.      */
mi_integer          Gen_DataLen;            /* Length of the data in bytes.     */
OcrType *           Gen_RetVal;             /* The return value.                */
mi_integer          Gen_Index10;            /* Array iterator */
char                Gen_LOFile[FILENAME_MAX];/* Store the file name here.   */

/* Get the current connection handle. */
Gen_Con = mi_open( NULL, NULL, NULL );
if( Gen_Con == 0 ){
        DBDK_TRACE_ERROR( "OcrTypeImportText", ERRORMESG1, 10 );
}
DBDK_TRACE_ENTER( "OcrTypeImportText" );
/* Point to the input data. */
Gen_InData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *) Gen_param1 );
Gen_StartInData = Gen_InData;
/* Get the length of the input string. */
Gen_DataLen = mi_get_varlen( (mi_lvarchar *)Gen_param1 );
/* Allocate a new UDT for the return result. */
Gen_RetVal = (OcrType *)mi_alloc( sizeof( OcrType ) );
if( Gen_RetVal == 0 ){
        DBDK_TRACE_ERROR( "OcrTypeImportText", ERRORMESG2, 10 );
}
/* Point to the output data. */
Gen_OutData = (OcrType *)Gen_RetVal;
/* Perform the operation on each of the array elements. */
for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
        mi_integer          Gen_Index20;        /* Array Index.                 */
        /* Perform the operation on each of the array elements. */
        for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                /* Get the mi_char value for P[Gen_Index10][Gen_Index20]. */
                Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeImportText", Gen_InData,
                                Gen_DataLen - (Gen_InData - Gen_StartInData),
                        0, "%c %n", (char *)&Gen_OutData->P[Gen_Index10][Gen_Index20] );
        }
}
        /* Get the MI_LO_HANDLE data value for ImgBuffer. */
        Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeImportText", Gen_InData,
                                        mi_get_varlen( (mi_lvarchar *) Gen_param1 ),
                                        sizeof(Gen_LOFile)-1,
                                        "%s %n",
                                        Gen_LOFile );

        /* Load the large object from the file. */
        Gen_LoadLOFromFile( Gen_Con, "OcrTypeImportText", Gen_LOFile,
```

```
                    &Gen_OutData->ImgBuffer );

        DBDK_TRACE_EXIT( "OcrTypeImportText" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

mi_impexp *
OcrTypeExportText
(
OcrType *               Gen_param1,     /* The UDT value.               */
MI_FPARAM *             Gen_fparam      /* Standard info - see DBDK docs.   */
)
{
        MI_CONNECTION *         Gen_Con;        /* The current connection.          */
        mi_integer              Gen_CharLen;    /* Estimate maximum length.         */
        OcrType *               Gen_InData;     /* Pointer to the input data.       */
        char *                  Gen_OutData;    /* Pointer to the output data.      */
        mi_impexp *             Gen_RetVal;     /* The return result.               */
        mi_integer              Gen_Index10; /* Array iterator */
        mi_integer              Gen_DataLen;    /* The data length.                 */
        char                    Gen_LOFile[FILENAME_MAX];/* Store the file name here.   */

        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeExportText", ERRORMESG2, 10 );
        }

        /* Point to the output data. */
        Gen_OutData = mi_get_vardata( (mi_lvarchar *)Gen_RetVal );
        /* Format the attribute value into the output string. */
        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer              Gen_Index20;            /* Array Index.             */
                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                        /* Format the mi_char value for P[Gen_Index10][Gen_Index20]. */
                        sprintf( Gen_OutData, "%c ", Gen_InData->P[Gen_Index10][Gen_Index20] );
                        Gen_OutData += strlen( Gen_OutData );
                }
        }
        /* Save the large object to disk. */
        strcpy( Gen_LOFile, LO_FN_MASK );
        Gen_StoreLOToFile( Gen_Con, "OcrTypeExportText", Gen_LOFile, &Gen_InData->ImgBuffer );
        /* Construct the return value: the LO's file name. */
        sprintf( Gen_OutData, "%s", Gen_LOFile );
```

```
        /* Advance past the formatted data. */
        Gen_OutData += strlen( Gen_OutData );
        Gen_DataLen = (mi_integer)(Gen_OutData - mi_get_vardata( (mi_lvarchar *) Gen_RetVal ));
        mi_set_varlen ( (mi_lvarchar *) Gen_RetVal, Gen_DataLen );
        DBDK_TRACE_EXIT( "OcrTypeExportText" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

OcrType *
OcrTypeInput
(
mi_lvarchar *           Gen_param1,         /* Pointer to the input text.      */
MI_FPARAM *             Gen_fparam          /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *     Gen_Con;        /* The current connection.          */
        gl_mchar_t *        Gen_InData;     /* Pointer to the input data.       */
        gl_mchar_t *        Gen_StartInData;/* First value of Gen_InData.       */
        OcrType *           Gen_OutData;    /* Pointer to the output data.      */
        mi_integer          Gen_DataLen;    /* Length of the data in bytes.     */
        OcrType *           Gen_RetVal;     /* The return value.                */
        mi_integer          Gen_Index10; /* Array iterator */
        char                Gen_LOFile[FILENAME_MAX];/* Store the file name here. */

        Gen_Con = mi_open( NULL, NULL, NULL );
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG1, 10 );
        }
        DBDK_TRACE_ENTER( "OcrTypeInput" );
        /* Point to the input data. */
        Gen_InData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *) Gen_param1 );
        Gen_StartInData = Gen_InData;

        /* Get the length of the input string. */
        Gen_DataLen = mi_get_varlen( Gen_param1 );
        /* Allocate a new UDT for the return result. */
        Gen_RetVal = (OcrType *)mi_alloc( sizeof( OcrType ) );
        if( Gen_RetVal == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG2, 10 );
        }
        Gen_OutData = (OcrType *)Gen_RetVal;
        /* Get the MI_LO_HANDLE data value for ImgBuffer. */
        Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeInput", Gen_InData,
                                                mi_get_varlen( Gen_param1 ),
```

```
                                        sizeof(Gen_LOFile)-1,
                                        "%s %n",
                                        Gen_LOFile );

            /* Load the large object from the file. */
        Gen_LoadLOFromFile( Gen_Con, "OcrTypeInput", Gen_LOFile,
                        &Gen_OutData->ImgBuffer );
        /* Staring to recognize by PQ code algorithm */
        Gen_OutData = Recorgnize( Gen_Con, Gen_OutData );
        DBDK_TRACE_EXIT( "OcrTypeInput" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

/* Call when you select data with data type*/
mi_lvarchar *
OcrTypeOutput
(
OcrType *           Gen_param1,     /* The UDT value.              */
MI_FPARAM *         Gen_fparam      /* Standard info - see DBDK docs.  */
)
{
        MI_CONNECTION *     Gen_Con;        /* The current connection.      */
        mi_integer          Gen_CharLen;    /* Estimate maximum length.     */
        OcrType *           Gen_InData;     /* Pointer to the input data.   */
        char *              Gen_OutData;    /* Pointer to the output data.  */
        mi_lvarchar *       Gen_RetVal;     /* The return result.           */
        mi_integer          Gen_DataLen;    /* The data length.             */
        mi_integer          Gen_Index10; /* Array iterator */
        char                Gen_LOFile[FILENAME_MAX];/* Store the file name here.  */

        Gen_Con = mi_open( NULL, NULL, NULL );
        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeOutput", ERRORMESG1, 10 );
        }
        DBDK_TRACE_ENTER( "OcrTypeOutput" );
        Gen_InData = Gen_param1;
        /* Compute the maximum length of the text representation. */
        Gen_CharLen     =   1               /* Leave room for the NULL terminator. */
                        + 6 * 5 * 2         /* Add the length for P. */
                        + 261               /* Add the length for ImgBuffer. */
                            ;
        /* Allocate room for the output string. */
        Gen_RetVal = mi_new_var( Gen_CharLen );
        if( Gen_RetVal == 0 ){
```

```
                DBDK_TRACE_ERROR( "OcrTypeOutput", ERRORMESG2, 10 );
        }

        /* Point to the output data. */
        Gen_OutData = mi_get_vardata( Gen_RetVal );
        /* Format the attribute value into the output string. */
        /* Perform the operation on each of the array elements. */
        for( Gen_Index10 = -1; ++Gen_Index10 < 6; ){
                mi_integer      Gen_Index20;            /* Array Index.         */
                /* Perform the operation on each of the array elements. */
                for( Gen_Index20 = -1; ++Gen_Index20 < 5; ){
                        /* Format the mi_char value for P[Gen_Index10][Gen_Index20]. */
                        sprintf( Gen_OutData, "%c ", Gen_InData->P[Gen_Index10][Gen_Index20] );
                        Gen_OutData += strlen( Gen_OutData );
                }
        }
        /* Advance past the formatted data. */
        Gen_OutData += strlen( Gen_OutData );
        Gen_DataLen = (mi_integer)(Gen_OutData - mi_get_vardata( Gen_RetVal ));
        mi_set_varlen ( Gen_RetVal, Gen_DataLen );
        DBDK_TRACE_EXIT( "OcrTypeOutput" );
        mi_close( Gen_Con );
        return Gen_RetVal;
}

void put(int X, int Y, char Data, ImgBufferType *buffer) {
   *(buffer->data+(Y*buffer->width)+X) = Data;
 }

char get(int X, int Y, ImgBufferType *buffer) {
   return *(buffer->data+(Y*buffer->width)+X);
 }

void LoadLoToBuffer(MI_CONNECTION *Gen_Con, OcrType *Ocr, ImgBufferType *buffer) {
        BITMAPFILEHEADER        bf;
        BITMAPINFO              bi;
        MI_LO_FD                pt;
        mi_int8                 offset, seek_pos;
        int                     size, z, i, j, l, x1, y1, x2, y2, yy;
        unsigned char           temp, k, BytePerRow, ByteReaded;
        long                    x, y;
        char                    *tempBuffer;

   // get Header from Lo
   if( (pt=mi_lo_open(Gen_Con, &Ocr->ImgBuffer, MI_LO_RDONLY))==MI_ERROR)
    { DBDK_TRACE_ERROR( "LoadLoToBuffer","UGEN4" , 10 ); }
```

```
if(mi_lo_read(Gen_Con, pt, (char*)&bf, sizeof(BITMAPFILEHEADER))==MI_ERROR ||
        mi_lo_read(Gen_Con, pt, (char*)&bi, sizeof(BITMAPINFOHEADER))==MI_ERROR){
            DBDK_TRACE_ERROR( "LoadLoToBuffer","UGEN4", 10 );
    }

// set file point to starting point of data
    ifx_int8cvint(bf.bfOffBits, &offset);
if(mi_lo_seek(Gen_Con, pt, &offset, MI_LO_SEEK_SET, &seek_pos)==MI_ERROR)
    { DBDK_TRACE_ERROR( "LoadLoToBuffer", "UGEN4" , 10 ); }

// allocate memory for buffer
size = (bi.bmiHeader.biWidth* bi.bmiHeader.biHeight);
    if( ( tempBuffer = (char *)mi_alloc(size) ) == 0 )
    { DBDK_TRACE_ERROR( "LoadLoToBuffer", ERRORMESG2, 10 ); }

// get data from file and write to tempBuffer
BytePerRow = (unsigned char)(/*Header.ImageSize*/ bi.bmiHeader.biSizeImage/ bi.bmiHeader.biHeight);
ByteReaded = 0;
for(i=0, j=0 ; i<bi.bmiHeader.biHeight; i++){
    for(z=bi.bmiHeader.biWidth; z>0;){
        mi_lo_read(Gen_Con, pt, &temp, 1);
        ByteReaded++;
        for(k=128, l=0; k>0&&l<z; l++){
            tempBuffer[j++] = (char)( (temp&k) == 0 )? '1' : '0';
            if(k==1) k=0;
            else k/=2;
        }
        z-=8;
        if(z<=0){
            while( ByteReaded++ < BytePerRow ) mi_lo_read(Gen_Con, pt, &temp, 1);
            ByteReaded = 0;
        }
    }
}

// find x1, y1, x2, y2 that will be uses to flip character
// in step of transfering data to buffer
for(x=i=0; x<bi.bmiHeader.biWidth&&i==0; x++) // find x1 {
    for(y=0; y<bi.bmiHeader.biHeight&&i==0; y++){
        if( *(tempBuffer+(y* bi.bmiHeader.biWidth)+x)=='1' ) { x1=x; i=1; }
    }
}
for(i=0, x=bi.bmiHeader.biWidth-1; x>=0&&i==0; x--) // find x2    {
    for(y=0; y<bi.bmiHeader.biHeight&&i==0; y++) {
        if( *(tempBuffer+(y* bi.bmiHeader.biWidth)+x)=='1' ) { x2=x; i=1; }
    }
```

```
    }
    for(y=i=0; y<bi.bmiHeader.biHeight&&i==0; y++) // find y1   {
            for(x=0; x<bi.bmiHeader.biWidth&&i==0; x++) {
               if( *(tempBuffer+(y* bi.bmiHeader.biWidth)+x)=='1' ) { y1=y; i=1; }
            }
    }
    for(i=0, y=bi.bmiHeader.biHeight-1; y>=0&&i==0; y--) // find y2    {
            for(x=0; x<bi.bmiHeader.biWidth&&i==0; x++) {
               if( *(tempBuffer+(y* bi.bmiHeader.biWidth)+x)=='1' ) { y2=y; i=1; }
            }
    }

    // transfering data from tempBuffer to buffer
    buffer->width = x2-x1+1;
    buffer->height = y2-y1+1;
    size = buffer->width*buffer->height;
    if( (buffer->data = (char *)mi_alloc(size))==0 )
     { DBDK_TRACE_ERROR( "LoadLoToBuffer", ERRORMESG2, 10 ); }

    yy = buffer->height-1;
    for(z=0,y=0;y<bi.bmiHeader.biHeight;y++) {
            for(x=0;x<bi.bmiHeader.biWidth; x++,z++){
               if(x>=x1 && x<=x2 && y>=y1 && y<=y2) {
                     put(x-x1, yy, tempBuffer[z], buffer);
                     if( x-x1 >= buffer->width-1 ) yy--;
               }
            }
    }
    mi_free(tempBuffer);
    if( mi_lo_close(Gen_Con, pt)==MI_ERROR )
            { DBDK_TRACE_ERROR( "LoadLoToBuffer", "UGEN4" , 10 ); }
}

void firstStep(MI_CONNECTION *Gen_Con, ImgBufferType *buffer)
{
    char            *temp, *pt;
    int             x, y;

    if( (temp = (char *)mi_alloc(buffer->width*buffer->height))==0 )
     { DBDK_TRACE_ERROR( "LoadLoToBuffer", ERRORMESG2, 10 ); }
    for(y=0; y<buffer->height; y++) {
            for(x=0; x<buffer->width; x++){
               if( x>0 && x<buffer->width-1 && y>0 && y<buffer->height-1 && get(x,y,buffer)=='1' ) {
                     if( (get(x-1,y,buffer)=='1'&&get(x+1,y,buffer)=='1') &&
                        !(get(x,y-1,buffer)=='1'&&get(x,y+1,buffer)=='1')
                     ) *(temp+x+(y*buffer->width))='H';
```

```
                else if ( (get(x,y-1,buffer)=='1'&&get(x,y+1,buffer)=='1')&&
                          !(get(x-1,y,buffer)=='1'&&get(x+1,y,buffer)=='1')
                         ) *(temp+x+(y*buffer->width))='V';
                else if ( get(x-1,y,buffer)=='1'&&get(x+1,y,buffer)=='1'&&
                          get(x,y-1,buffer)=='1'&&get(x,y+1,buffer)=='1'
                         ) *(temp+x+(y*buffer->width))='I';
                else *(temp+x+(y*buffer->width))='S';
            }
            else if( get(x,y,buffer)=='1' )
            {
                if( x==0 || x==buffer->width-1 )
                {
                  if( (y==0||y==buffer->height-1) )
                   { *(temp+x+(y*buffer->width))='S'; }
                  else if(get(x,y-1,buffer)=='1'&&get(x,y+1,buffer)=='1')
                   { *(temp+x+(y*buffer->width))='V'; }
                  else
                   { *(temp+x+(y*buffer->width))='S'; }
                }
                else if( y==0 || y==buffer->height-1 )
                {
                  if( (x==0||x==buffer->width-1) )
                   { *(temp+x+(y*buffer->width))='S'; }
                  else if(get(x-1,y,buffer)=='1'&&get(x+1,y,buffer)=='1')
                   { *(temp+x+(y*buffer->width))='H'; }
                  else
                   { *(temp+x+(y*buffer->width))='S'; }
                }
            }
            else *(temp+x+(y*buffer->width))='0';
        }
    }
    pt = buffer->data;
    buffer->data = temp;
    mi_free(pt);
}

void secondStep(MI_CONNECTION *Gen_Con, ImgBufferType *buffer) {
    char            *temp, *pt;
    unsigned char   found, value, x, y;
    int             i;

    if( (temp = (char *)mi_alloc(buffer->width*buffer->height))==0 )
    { DBDK_TRACE_ERROR( "LoadLoToBuffer", ERRORMESG2, 10 ); }
    for(y=0; y<buffer->height; y++) {
        for(x=0; x<buffer->width; x++) {
```

```
            if(get(x,y,buffer)=='0') {
                value=0;
                // scan to buttom
                for(found=0, i=y+1; i<buffer->height && !found; i++)
                 { if(get(x,i,buffer)!='0') found=1; }
                if( found ) value++;
                // scan to left
                for(found=0, i=x-1; i>=0 && !found; i--)
                 { if(get(i,y,buffer)!='0') found=1; }
                if( found ) value+=2;
                // scan to top
                for(found=0, i=y-1; i>=0 && !found; i--)
                 { if(get(x,i,buffer)!='0') found=1; }
                if( found ) value+=4;
                // scan to right
                for(found=0, i=x+1; i<buffer->width && !found; i++)
                 { if(get(i,y,buffer)!='0') found=1; }
                if( found ) value+=8;
                // check value
                switch (value) {
                   case 0 : *(temp+x+(y*buffer->width))='0'; break;
                   case 1 : *(temp+x+(y*buffer->width))='1'; break;
                   case 2 : *(temp+x+(y*buffer->width))='2'; break;
                   case 3 : *(temp+x+(y*buffer->width))='3'; break;
                   case 4 : *(temp+x+(y*buffer->width))='4'; break;
                   case 5 : *(temp+x+(y*buffer->width))='5'; break;
                   case 6 : *(temp+x+(y*buffer->width))='6'; break;
                   case 7 : *(temp+x+(y*buffer->width))='7'; break;
                   case 8 : *(temp+x+(y*buffer->width))='8'; break;
                   case 9 : *(temp+x+(y*buffer->width))='9'; break;
                   case 10: *(temp+x+(y*buffer->width))='A'; break;
                   case 11: *(temp+x+(y*buffer->width))='B'; break;
                   case 12: *(temp+x+(y*buffer->width))='C'; break;
                   case 13: *(temp+x+(y*buffer->width))='D'; break;
                   case 14: *(temp+x+(y*buffer->width))='E'; break;
                   case 15: *(temp+x+(y*buffer->width))='F'; break;
                }
            }
            else *(temp+x+(y*buffer->width))=get(x,y,buffer);
        }
    }
    pt = buffer->data;
    buffer->data = temp;
    mi_free(pt);
}
```

```
void thirdStep(ImgBufferType *buffer) {
    unsigned char    inLoop=1;
    int              x, y;

    while(inLoop) {
        inLoop=0;
        for(y=1; y<buffer->height-2; y++){
            for(x=1; x<buffer->width-1; x++)  {
                if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='C'||get(x+1,y,buffer)=='C'||
                     get(x,y-1,buffer)=='C'||get(x,y+1,buffer)=='C')
                  ) { put(x, y, 'C',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='9'||get(x+1,y,buffer)=='9'||
                     get(x,y-1,buffer)=='9'||get(x,y+1,buffer)=='9')
                  ) { put(x, y, '9',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='E'||get(x+1,y,buffer)=='E'||
                     get(x,y-1,buffer)=='E'||get(x,y+1,buffer)=='E')
                  ) { put(x, y, 'E',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='D'||get(x+1,y,buffer)=='D'||
                     get(x,y-1,buffer)=='D'||get(x,y+1,buffer)=='D')
                  ) { put(x, y, 'D',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='B'||get(x+1,y,buffer)=='B'||
                     get(x,y-1,buffer)=='B'||get(x,y+1,buffer)=='B')
                  ) { put(x, y, 'B',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='7'||get(x+1,y,buffer)=='7'||
                     get(x,y-1,buffer)=='7'||get(x,y+1,buffer)=='7')
                  ) { put(x, y, '7',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='F' &&
                    (get(x-1,y,buffer)=='5'||get(x+1,y,buffer)=='5'||
                     get(x,y-1,buffer)=='5'||get(x,y+1,buffer)=='5')
                  ) { put(x, y, '5',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='E' &&
                    (get(x-1,y,buffer)=='C'||get(x+1,y,buffer)=='C'||
                     get(x,y-1,buffer)=='C'||get(x,y+1,buffer)=='C')
                  ) { put(x, y, 'C',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='D' &&
                    (get(x-1,y,buffer)=='C'||get(x+1,y,buffer)=='C'||
                     get(x,y-1,buffer)=='C'||get(x,y+1,buffer)=='C')
                  ) { put(x, y, 'C',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='D' &&
                    (get(x-1,y,buffer)=='9'||get(x+1,y,buffer)=='9'||
```

```
                    get(x,y-1,buffer)=='9'||get(x,y+1,buffer)=='9')
                  ) { put(x, y, '9',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='B' &&
                    (get(x-1,y,buffer)=='9'||get(x+1,y,buffer)=='9'||
                    get(x,y-1,buffer)=='9'||get(x,y+1,buffer)=='9')
                  ) { put(x, y, '9',buffer); inLoop=1; }
                else if( get(x,y,buffer)=='7' &&
                    (get(x-1,y,buffer)=='6'||get(x+1,y,buffer)=='6'||
                    get(x,y-1,buffer)=='6'||get(x,y+1,buffer)=='6')
                  ) { put(x, y, '6',buffer); inLoop=1; }
            }
        }
    }
}

void findConcen(OcrType *Ocr, int m, int x, int y, ImgBufferType *buffer) {
    char            inLoop, found, i;

    Ocr->P[m][0] = (mi_char)get(x, y, buffer);
        Ocr->P[m][1]=Ocr->P[m][2]=Ocr->P[m][3]=Ocr->P[m][4] = (mi_char)'X';
    // find p1
    for(i=x, inLoop=1, found=0; i<buffer->width && inLoop; i++){
        if( get(i, y, buffer)=='I' || get(i, y, buffer)=='V' ||
            get(i, y, buffer)=='H' || get(i, y, buffer)=='S' ) found=1;
        if( found==1 && get(i, y, buffer)!='I' && get(i, y, buffer)!='V' &&
            get(i, y, buffer)!='H' && get(i, y, buffer)!='S' ) { Ocr->P[m][1]=(mi_char)get(i, y, buffer); inLoop=0; }
        else if( found==1 && i==(buffer->width-1) ) { Ocr->P[m][1]=(mi_char)get(i, y, buffer); inLoop=0; }
    }
    // find p2
    for(i=y, inLoop=1, found=0; i>=0 && inLoop; i--)  {
        if( get(x, i, buffer)=='I' || get(x, i, buffer)=='V' ||
            get(x, i, buffer)=='H' || get(x, i, buffer)=='S' ) found=1;
        if( found==1 && get(x, i, buffer)!='I' && get(x, i, buffer)!='V' &&
            get(x, i, buffer)!='H' && get(x, i, buffer)!='S' ) { Ocr->P[m][2]=(mi_char)get(x, i, buffer); inLoop=0; }
        else if( found==1 && i==0 ) { Ocr->P[m][2]=(mi_char)get(x, i, buffer); inLoop=0; }
    }
    // find p3
    for(i=x, inLoop=1, found=0; i>=0 && inLoop; i--) {
        if( get(i, y, buffer)=='I' || get(i, y, buffer)=='V' ||
            get(i, y, buffer)=='H' || get(i, y, buffer)=='S' ) found=1;
        if( found==1 && get(i, y, buffer)!='I' && get(i, y, buffer)!='V' &&
            get(i, y, buffer)!='H' && get(i, y, buffer)!='S' ) { Ocr->P[m][3]=(mi_char)get(i, y, buffer); inLoop=0; }
        else if( found==1 && i==0 ) { Ocr->P[m][3]=(mi_char)get(i, y, buffer); inLoop=0; }
    }
    // find p4
    for(i=y, inLoop=1, found=0; i<buffer->height && inLoop; i++)  {
```

```
            if( get(x, i, buffer)=='I' || get(x, i, buffer)=='V' ||
                get(x, i, buffer)=='H' || get(x, i, buffer)=='S' ) found=1;
            if( found==1 && get(x, i, buffer)!='I' && get(x, i, buffer)!='V' &&
                get(x, i, buffer)!='H' && get(x, i, buffer)!='S' ) { Ocr->P[m][4]=(mi_char)get(x, i, buffer); inLoop=0; }
            else if( found==1 && i==(buffer->height-1) ) { Ocr->P[m][4]=(mi_char)get(x, i, buffer); inLoop=0; }
        }
    }

void finalStep(ImgBufferType *buffer, OcrType *Ocr) {
    findConcen(Ocr, 0, buffer->width/8, buffer->height/8, buffer);
    findConcen(Ocr, 1, buffer->width/2, buffer->height/8, buffer);
    findConcen(Ocr, 2, buffer->width/2, buffer->height/2, buffer);
    findConcen(Ocr, 3, buffer->width-(buffer->width/8), buffer->height-(3*(buffer->height/8)), buffer);
    findConcen(Ocr, 4, buffer->width/2, buffer->height-(buffer->height/4), buffer);
    findConcen(Ocr, 5, buffer->width/8, buffer->height-(buffer->height/8), buffer);
    mi_free(buffer->data);
}

OcrType * Recorgnize(MI_CONNECTION *Gen_Con, OcrType *Ocr){
    ImgBufferType               *buffer;

    if( ( buffer = (ImgBufferType *)mi_alloc( sizeof( ImgBufferType ) ) ) == 0 )
            { DBDK_TRACE_ERROR( "Recorgnize", ERRORMESG2, 10 ); }
            LoadLoToBuffer(Gen_Con, Ocr, buffer);
    firstStep(Gen_Con, buffer);
    secondStep(Gen_Con, buffer);
    thirdStep(buffer);
    finalStep(buffer, Ocr);
            return Ocr;
}

/* end of extends my program */

#ifdef __cplusplus

}

#endif
```

```
/*--- ไฟล์ support.c  ---*/

#ifdef __cplusplus

extern "C"
{

#endif

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <ifxgls.h>
#include <mi.h>

/* This is the project include file. */
#include "Ocr.h"

gl_mchar_t *
Gen_sscanf
(
MI_CONNECTION *         Gen_Con,        /* The database connection.        */
char *                  Gen_Caller,     /* Name of the calling function.   */
gl_mchar_t *            Gen_InData,     /* The input string data.          */
mi_integer              Gen_InDataLen,  /* The length of Gen_InData.       */
mi_integer              Gen_Width,      /* Max length if data is text.     */
char *                  Gen_Format,     /* The data's format.              */
char *                  Gen_Result      /* Place the result here.          */
)
{
        gl_mchar_t *    Gen_In;         /* Scanning ptr.                   */
        gl_mchar_t *    Gen_SaveIn;     /* Save Gen_In here.               */
        gl_mchar_t      Gen_NextChar;   /* Get the next char here.         */
        char *          Gen_InStart;    /* Gen_In before advancement.      */
        gl_mchar_t *    Gen_Out;        /* Place the GLS result here.      */
        mi_integer      Gen_ByteCount;  /* The number of bytes.            */

        Gen_In      =   Gen_InData;
        Gen_Out     =   (gl_mchar_t *)Gen_Result;
        Gen_ByteCount = 0;

        /* Scan past non-format characters. */
        while( *Gen_Format && *Gen_In ){
                while( Gen_In < Gen_InData + Gen_InDataLen ) {
                        /* Stop at the first non-blank. */
```

```
                if( !ifx_gl_ismblank( Gen_In, 4 ) ){
                        break;
                }

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );
        }

        /* Check for the format character. */
        if( *Gen_Format == '%' ){
                break;
        }

        if( *(char *)Gen_In != *Gen_Format ){
                goto parse_error;
        }

        /* Keep track of how far we've advanced. */
        Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

        /* Advance to the next character. */
        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

        /* Advance to the next character. */
        ++Gen_Format;
}

/* Check for no input data string. */
if( *(char *)Gen_In == '\0' ){
        goto parse_error;
}

/* The following code now handles the various format types. */
if( !strncmp( Gen_Format, "%s %n", 5 ) && Gen_Width ){
        if( '\"' != *Gen_In ){
                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG7, 10 );
        }
        /* Advance past the leading quote. */
        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        /* Copy while there's sufficient room. */
        while( Gen_Width ) {
                Gen_InStart = (char *)Gen_In;
```

```
                switch( *Gen_In ){
                        case '\"':
                                Gen_SaveIn = Gen_In;
                                Gen_NextChar = *ifx_gl_mbsnext( Gen_In, 4 );
                                Gen_In = Gen_SaveIn;
                                if( Gen_NextChar == '\"' ){
                                        *Gen_Out++ = '\"';
                                        /* Advance past the first quote. */
                                        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
                                        /* Advance past the second space. */
                                        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
                                        break;
                                }

                                *Gen_Out = (char)'\0';

                                /* Advance past the trailing quote. */
                                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                                /* Advance past the trailing space. */
                                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
                                return Gen_In;

                        case '\0':
                                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG9, 10 );
                        default:
                                *Gen_Out++ = *Gen_In;
                                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
                                Gen_Width -= (char *)Gen_In - Gen_InStart;
                                break;
                }
        }

        DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG10, 10 );
}

/* Check for an mi_integer. */
else if( !strncmp( Gen_Format, "%d %n", 5 ) ){
        mi_decimal   Gen_dec_number;  /* Store an mi_decimal value here. */
        mi_integer   Gen_IntWidth;    /* Used to compute the width.      */

        /* Convert the GLS string to a decimal value. */
        if( ifx_gl_convert_number( &Gen_dec_number, (char *)Gen_In, "%d" ) == -1 ){
                goto parse_error;
        }
```

```
        /* Convert the decimal to a 'C' integer. */
        if( dectoint( &Gen_dec_number, (int *)Gen_Result ) != 0 ){
                goto parse_error;
        }

        /* Insure the number is not too wide. */
        for( Gen_IntWidth = 0;
            Gen_In < Gen_InData + Gen_InDataLen &&
                        !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_IntWidth;
        }

        if( Gen_IntWidth > 11 ){
                goto parse_error;
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for a double value. */
else if( !strncmp( Gen_Format, "%lf %n", 6 ) )
{
        mi_decimal   Gen_dec_number;  /* Store an mi_decimal value here. */

        /* Convert the GLS string to a decimal value. */
        if( ifx_gl_convert_number( &Gen_dec_number, (char *)Gen_In, "%e" ) == -1 )
```

```
        {
                goto parse_error;
        }

        /* Convert the decimal to a 'C' double. */
        if( dectodbl( &Gen_dec_number, (double *)Gen_Result ) != 0 ){
                goto parse_error;
        }

        /* Scan past the number. */
        while( Gen_In < Gen_InData + Gen_InDataLen ){
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 6;
}

/* Check for a float value. */
else if( !strncmp( Gen_Format, "%f %n", 5 ) ){
        mi_double_precision Gen_dbl_number; /* Store an mi_float value here.   */
        mi_decimal          Gen_dec_number; /* Store an mi_decimal value here. */

        /* Convert the GLS string to a decimal value. */
        if( ifx_gl_convert_number( &Gen_dec_number, (char *)Gen_In, "%e" ) == -1 ){
                goto parse_error;
        }

        /* Convert the decimal to a 'C' float. */
        if( dectodbl( &Gen_dec_number, &Gen_dbl_number ) != 0 )
        {
```

```
                goto parse_error;
        }
        *(float *)Gen_Result = (float)Gen_dbl_number;

        /* Scan past the float value. */
        while( Gen_In < Gen_InData + Gen_InDataLen )   {
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for a long value. */
else if( !strncmp( Gen_Format, "%ld %n", 6 ) ){
        mi_decimal   Gen_dec_number;  /* Store an mi_decimal value here. */
        mi_integer   Gen_LongWidth;   /* Used to compute the width.      */

        /* Convert the GLS string to a decimal value. */
        if( ifx_gl_convert_number( &Gen_dec_number, (char *)Gen_In, "%d" ) == -1 ){
                goto parse_error;
        }

        /* Convert the decimal to a 'C' long. */
        if( dectolong( &Gen_dec_number, (long *)Gen_Result ) != 0 ){
                goto parse_error;
        }

        /* Insure the number is not too wide. */
        for( Gen_LongWidth = 0;
                Gen_In < Gen_InData + Gen_InDataLen &&
```

```
                !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'                  ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_LongWidth;
        }
        if( Gen_LongWidth > 11 ){
                goto parse_error;
        }

        /* Advance the format string. */
        Gen_Format += 6;
}

/* Check for an int8 value. */
else if( !strncmp( Gen_Format, "%8 %n", 5 ) ){
        gl_mchar_t * Gen_Int8Ptr;     /* Scanning pointer.          */
        gl_mchar_t * Gen_Int8Ptr2;    /* Scanning pointer.          */
        mi_integer   Gen_Int8Width;   /* Keep track of the length.  */
        mi_integer   Gen_Int8Ret;     /* ifx_int8cvasc return value.*/
        char         Gen_Int8Str[80]; /* Hold the mi_int8 string here. */

        for( Gen_Int8Ptr  = (gl_mchar_t *)Gen_Int8Str,
             Gen_Int8Ptr2 = Gen_In,
             Gen_Int8Width = 0;
                        Gen_In < Gen_InData + Gen_InDataLen &&
                        !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Keep track of how far we've advanced. */
```

```
            Gen_Int8Width += ifx_gl_mblen( Gen_In, 4 );

            /* Advance the input pointer. */
            Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
    }

    /* Copy the string and terminate. */
    memcpy( Gen_Int8Str, Gen_Int8Ptr2, Gen_Int8Width );
    Gen_Int8Str[Gen_Int8Width] = (char)'\0';

    /* Check for an empty string. */
    if( Gen_Int8Width == 0 ){
            goto parse_error;
    }

    /* Convert the mi_int8 value from text to internal format. */
    Gen_Int8Ret = ifx_int8cvasc( Gen_Int8Str, strlen( Gen_Int8Str ),
                            (ifx_int8_t *)Gen_Result );
    if( Gen_Int8Ret ){
            goto parse_error;
    }

    /* Scan past the int8 value. */
    while( Gen_In < Gen_InData + Gen_InDataLen )   {
            /* Stop at the first non-number. */
            if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                    *Gen_In == 'e'          ||
                    *Gen_In == 'E'          ||
                    *Gen_In == '.'          ||
                    *Gen_In == '+'          ||
                    *Gen_In == '-')         )
            {
                    break;
            }

            /* Keep track of how far we've advanced. */
            Gen_Int8Width += ifx_gl_mblen( Gen_In, 4 );

            /* Advance to the next character. */
            Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
    }

    /* Advance the format string. */
    Gen_Format += 5;
}
```

```
/* Check for an int1 value. */
else if( !strncmp( Gen_Format, "%1 %n", 5 ) ){
        int Gen_Int1Result;

        if( 1 != sscanf( (char *)Gen_In, "%d %n", &Gen_Int1Result, &Gen_ByteCount ) )  {
                goto parse_error;
        }

        /* Advance the input pointer. */
        Gen_In += Gen_ByteCount;

        if( Gen_Int1Result > 255 ){
                goto parse_error;
        }

        *(mi_int1 *)Gen_Result = Gen_Int1Result;

        /* Scan to trailing white space. */
        while( Gen_In < Gen_InData + Gen_InDataLen )  {
                /* Stop at the first non-blank. */
                if( !ifx_gl_ismblank( Gen_In, 4 ) ){
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }
        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_smallint value. */
else if( !strncmp( Gen_Format, "%2 %n", 5 ) ){
        int          Gen_SmallIntResult; /* The mi_smallint value.    */
        gl_mchar_t *  Gen_SmallIntPtr;    /* Scanning pointer.         */
        mi_integer    Gen_SmallIntWidth;  /* Keep track of the length. */
        mi_integer    Gen_ScanCount;      /* Number of bytes scanned.  */

        /* Use sscanf to scan the mi_smallint value. */
        if( 1 != sscanf( (char *)Gen_In, "%d %n", &Gen_SmallIntResult, &Gen_ScanCount ) ){
                goto parse_error;
        }
```

```
        /* Insure the number is not too wide. */
        for( Gen_SmallIntPtr  = Gen_In,
            Gen_SmallIntWidth = 0;
                        Gen_In < Gen_InData + Gen_InDataLen &&
                        !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance the input pointer. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
                /* Keep track of the length. */
                ++Gen_SmallIntWidth;
        }

        if( Gen_SmallIntWidth > 6 ){
                goto parse_error;
        }

        /* Validate the value. */
        if( Gen_SmallIntResult < -32767 || 32767 < Gen_SmallIntResult ){
                goto parse_error;
        }

        /* Save the temporary value back into the result. */
        *(mi_smallint *)Gen_Result = Gen_SmallIntResult;
        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_unsigned_smallint value. */
else if( !strncmp( Gen_Format, "%u2 %n", 5 ) ){
        int         Gen_USmallIntResult; /* The mi_smallint value.    */
        mi_integer  Gen_USmallIntWidth;  /* Keep track of the length. */
        mi_integer  Gen_ScanCount;       /* Number of bytes scanned.  */

        /* Use sscanf to scan the mi_unsigned_smallint value. */
        if( 1 != sscanf( (char *)Gen_In, "%d %n", &Gen_USmallIntResult, &Gen_ScanCount ) )
        {
                goto parse_error;
        }

        /* Insure the number is not too wide. */
        for( Gen_USmallIntWidth = 0;
                        Gen_In < Gen_InData + Gen_InDataLen &&
                        !ifx_gl_ismblank( Gen_In, 4 ); )
```

```
        {
                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance the input pointer. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_USmallIntWidth;
        }

        if( Gen_USmallIntWidth > 6 )
        {
                goto parse_error;
        }

        /* Validate the value. */
        if( Gen_USmallIntResult < 0 || 65535 < Gen_USmallIntResult)
        {
                goto parse_error;
        }

        /* Save the temporary value back into the result. */
        *(mi_unsigned_smallint *)Gen_Result = Gen_USmallIntResult;

        /* Advance the format string. */
        Gen_Format += 6;
}

else if( !strncmp( Gen_Format, "%b %n", 5 ) )
{
        if( *(char *)Gen_In == 'T' || *(char *)Gen_In == 't' )
        {
                *(char *)Gen_Result = 1;
        }
        else if( *(char *)Gen_In == 'F' || *(char *)Gen_In == 'f' )
        {
                *(char *)Gen_Result = 0;
        }
        else
        {
                goto parse_error;
        }

        Gen_ByteCount += sizeof(char);
        Gen_In += sizeof(char);
```

```
        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an unsigned int value. */
else if( !strncmp( Gen_Format, "%x %n", 5 ) ){
        mi_integer Gen_UIntWidth;  /* Used to compute the width.      */
        mi_decimal Gen_dec_number; /* Store an mi_decimal value here. */

        /* Convert the GLS string to a decimal value. */
        if( ifx_gl_convert_number( &Gen_dec_number, (char *)Gen_In, "%u" ) == -1 )
        {
                goto parse_error;
        }

        /* Convert the decimal to a 'C' unsigned integer. */
        if( dectoint( &Gen_dec_number, (int *)Gen_Result ) != 0 )
        {
                goto parse_error;
        }

        /* Scan past the number and insure the number is not too wide. */
        for( Gen_UIntWidth = 0;
             Gen_In < Gen_InData + Gen_InDataLen &&
                !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_UIntWidth;
```

```
        }

        if( Gen_UIntWidth > 11 )
        {
                goto parse_error;
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for a multibyte character of maximum length 4. */
else if( !strncmp( Gen_Format, "%W %n", 5 ) )
{
        /* Convert the multibyte character to gl_wchar_t. */
        Gen_ByteCount += ifx_gl_mbtowc( (gl_wchar_t *)Gen_Result, Gen_In, 4 );
        Gen_In        += ifx_gl_mbtowc( (gl_wchar_t *)Gen_Result, Gen_In, 4 );

        /* Scan to trailing white space. */
        while( Gen_In < Gen_InData + Gen_InDataLen )
        {
                /* Stop at the first non-blank. */
                if( !ifx_gl_ismblank( Gen_In, 4 ) )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for a multibyte character of max length 2 */
else if( !strncmp( Gen_Format, "%w %n", 5 ) )
{
        gl_wchar_t Gen_WideChar;
        int                Gen_NumBytes;

        /* Convert multibyte character to mi_wchar. */
        Gen_NumBytes = ifx_gl_mbtowc( (gl_wchar_t *) &Gen_WideChar, Gen_In, 2 );
```

```
        /* Check for an invalid character */
        if (Gen_NumBytes <= 0)
        {
                goto parse_error;
        }

        /* Advance the input data pointer. */
        Gen_ByteCount += Gen_NumBytes;
        Gen_In        += Gen_NumBytes;
        *((mi_wchar *)Gen_Result) = (mi_wchar) Gen_WideChar;

        /* Scan to trailing white space. */
        while( Gen_In < Gen_InData + Gen_InDataLen )  {
                /* Stop at the first non-blank. */
                if( !ifx_gl_ismblank( Gen_In, 4 ) ){
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );
                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_money value. */
else if( !strncmp( Gen_Format, "%m %n", 5 ) ){
        gl_mchar_t *   Gen_MoneyPtr;     /* Scanning pointer.              */
        gl_mchar_t *   Gen_MoneyPtr2;    /* Scanning pointer.              */
        mi_lvarchar *  Gen_MoneyLV;      /* mi_lvarchar ptr to Gen_MoneyStr. */
        mi_money *     Gen_MoneyVal;     /* The binary money value.        */
        mi_integer     Gen_MoneyWidth;   /* Keep track of the length.      */
        char           Gen_MoneyStr[80]; /* Hold the mi_money string here.  */

        for( Gen_MoneyPtr   = (gl_mchar_t *)Gen_MoneyStr,
             Gen_MoneyPtr2  = Gen_In,
             Gen_MoneyWidth = 0;
                        Gen_In < Gen_InData + Gen_InDataLen &&
                        !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Advance the input pointer. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
```

```
            /* Keep track of the length. */
            ++Gen_MoneyWidth;
}

/* Copy the string and terminate. */
ifx_gl_mbscpy( (gl_mchar_t *)Gen_MoneyStr, Gen_MoneyPtr2,
                                    (int)(Gen_In - Gen_MoneyPtr2) );
Gen_MoneyStr[Gen_In - Gen_MoneyPtr2] = (char)'\0';

/* Check for an empty string. */
if( Gen_MoneyWidth == 0 ){
            goto parse_error;
}

/* Convert the money string to mi_lvarchar. */
Gen_MoneyLV = mi_string_to_lvarchar( Gen_MoneyStr );

/* Check for an error in conversion. */
if( Gen_MoneyLV == 0 ){
            goto parse_error;
}

/* Convert the money string to internal format. */
Gen_MoneyVal = mi_money_to_binary( Gen_MoneyLV );

/* Check for an error in conversion. */
if( Gen_MoneyVal == NULL ) {
            goto parse_error;
}

/* Copy to the UDT data area. */
memcpy( Gen_Result, Gen_MoneyVal, sizeof(mi_money) );

/* ... and free the temporary money data values. */
mi_free( Gen_MoneyLV );
mi_free( Gen_MoneyVal );

/* Scan past the money value. */
while( Gen_In < Gen_InData + Gen_InDataLen )
{
            /* Stop at the first non-number. */
            if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
```

```
                    *Gen_In == '-')          )
            {
                    break;
            }

            /* Keep track of how far we've advanced. */
            Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

            /* Advance to the next character. */
            Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_numeric/mi_decimal value. */
else if( !strncmp( Gen_Format, "%N %n", 5 ) ){
        gl_mchar_t * Gen_DecPtr;    /* Scanning pointer.          */
        gl_mchar_t * Gen_DecPtr2;   /* Scanning pointer.          */
        mi_decimal   Gen_DecVal;    /* The binary decimal value.  */
        mi_integer   Gen_DecIndex;  /* Keep track of the length.  */
        mi_integer   Gen_DecRet;    /* deccvasc return value.     */
        char         Gen_DecStr[80]; /* Hold the mi_decimal string here. */

        for( Gen_DecPtr  = (gl_mchar_t *)Gen_DecStr,
          Gen_DecPtr2  = Gen_In,
          Gen_DecIndex = 0;
                    Gen_In < Gen_InData + Gen_InDataLen &&
                    !ifx_gl_ismblank( Gen_In, 4 ); )
        {
                /* Advance the input pointer. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_DecIndex;
                ++Gen_ByteCount;
        }

        /* Copy the string and terminate. */
        ifx_gl_mbscpy( (gl_mchar_t *)Gen_DecStr, Gen_DecPtr2,
                                (int)(Gen_In - Gen_DecPtr2) );
        Gen_DecStr[Gen_In - Gen_DecPtr2] = (char)'\0';

        /* Check for an empty string. */
        if( Gen_DecIndex == 0 )
```

```
        {
                goto parse_error;
        }

        /* Convert the mi_decimal value from text to internal format. */
        Gen_DecRet = deccvasc( Gen_DecStr, strlen( Gen_DecStr ), &Gen_DecVal );

        /* Check for an error in conversion. */
        if( Gen_DecRet )
        {
                goto parse_error;
        }

        /* Copy to the UDT data area. */
        memcpy( Gen_Result, &Gen_DecVal, sizeof(mi_decimal) );

        /* Scan past the decimal value. */
        while( Gen_In < Gen_InData + Gen_InDataLen ){
                /* Stop at the first non-number. */
                if( !(ifx_gl_ismdigit( Gen_In, 4 ) ||
                        *Gen_In == 'e'          ||
                        *Gen_In == 'E'          ||
                        *Gen_In == '.'          ||
                        *Gen_In == '+'          ||
                        *Gen_In == '-')         )
                {
                        break;
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_date value. */
else if( !strncmp( Gen_Format, "%D %n", 5 ) )      {
        gl_mchar_t * Gen_DatePtr;     /* Scanning pointer.              */
        gl_mchar_t * Gen_DatePtr2;    /* Scanning pointer.              */
        long         Gen_DateVal;     /* The binary date value.         */
        mi_integer   Gen_DateIndex;   /* Keep track of the length.      */
```

```
mi_integer   Gen_RstrdateRet; /* rstrdate return value.        */
char         Gen_DateStr[80]; /* Hold the mi_date string here. */
int          Gen_NumBytes;    /* Number of bytes.              */

for( Gen_DatePtr   = (gl_mchar_t *)Gen_DateStr,
     Gen_DatePtr2  = Gen_In,
     Gen_DateIndex = 0;
                    Gen_In < Gen_InData + Gen_InDataLen &&
                    !ifx_gl_ismblank( Gen_In, 4 ); )
{
          /* Advance the input pointer. */
          Gen_NumBytes = ifx_gl_mblen( Gen_In, 4 );
          Gen_In += Gen_NumBytes;

          /* Keep track of the length. */
          Gen_DateIndex += Gen_NumBytes;
          Gen_ByteCount += Gen_NumBytes;
}

/* Copy the string and terminate. */
ifx_gl_mbscpy( (gl_mchar_t *)Gen_DateStr, Gen_DatePtr2,
                              (int)(Gen_In - Gen_DatePtr2) );
Gen_DateStr[Gen_In - Gen_DatePtr2] = (char)'\0';

/* Check for an empty string. */
if( Gen_DateIndex == 0 ){
          goto parse_error;
}

/* Convert the date string to internal format. */
Gen_RstrdateRet = rstrdate( Gen_DateStr, &Gen_DateVal );

/* Check for an error in conversion. */
if( Gen_RstrdateRet ){
          goto parse_error;
}
/* Copy to the UDT data area. */
memcpy( Gen_Result, &Gen_DateVal, sizeof(mi_date) );

/* Scan past the date string. */
while( Gen_In < Gen_InData + Gen_InDataLen )   {
          /* Stop at the first non-number. */
          if( !ifx_gl_ismdigit( Gen_In, 4 ) )
          {
                    break;
          }
```

```
                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }
        /* Advance the format string. */
        Gen_Format += 5;
}

/* Check for an mi_datetime value. */
else if( !strncmp( Gen_Format, "%T %n", 5 ) ){
        gl_mchar_t * Gen_DTPtr;     /* Scanning pointer.          */
        gl_mchar_t * Gen_DTPtr2;    /* Scanning pointer.          */
        mi_datetime  Gen_DTVal;     /* The binary date value.     */
        mi_integer   Gen_DTIndex;   /* Keep track of the length.  */
        int          Gen_Blanks;    /* Number of blanks scanned.  */
        int          Gen_DtcvascRet; /* dtcvasc return value.     */
        char         Gen_DTStr[80]; /* Hold the mi_date string here. */

        for( Gen_DTPtr  = (gl_mchar_t *)Gen_DTStr,
             Gen_DTPtr2 = Gen_In,
             Gen_Blanks = 0,
             Gen_DTIndex = 0;
                        Gen_In < Gen_InData + Gen_InDataLen; )
        {
                /* Stop at the second blank. */
                if( ifx_gl_ismblank( Gen_In, 4 ) )
                {
                        if( ++Gen_Blanks == 2 )
                        {
                                break;
                        }
                }

                /* Advance the input pointer. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

                /* Keep track of the length. */
                ++Gen_DTIndex;
                ++Gen_ByteCount;
        }

        /* Copy the string and terminate. */
        ifx_gl_mbscpy( (gl_mchar_t *)Gen_DTStr, Gen_DTPtr2,
```

```
                                        (int)(Gen_In - Gen_DTPtr2) );
            Gen_DTStr[Gen_In - Gen_DTPtr2] = (char)'\0';
            /* Check for an empty string. */
            if( Gen_DTIndex == 0 ){
                    goto parse_error;
            }
            Gen_DTVal.dt_qual = ((mi_datetime *)Gen_Result)->dt_qual;
            /* Convert the mi_datetime value to internal format. */
            Gen_DtcvascRet = dtcvasc( Gen_DTStr, &Gen_DTVal );
            /* Check to insure that the conversion was successful. */
            if( Gen_DtcvascRet )            {
                    goto parse_error;
            }

            /* Copy to the UDT data area. */
            memcpy( Gen_Result, &Gen_DTVal, sizeof(mi_datetime) );

            /* Scan past the datetime string. */
            while( Gen_In < Gen_InData + Gen_InDataLen )   {
                    /* Stop at the first non-number. */
                    if( !ifx_gl_ismdigit( Gen_In, 4 ) )
                    {
                            break;
                    }

                    /* Keep track of how far we've advanced. */
                    Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                    /* Advance to the next character. */
                    Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
            }

            /* Advance the format string. */
            Gen_Format += 5;
    }

    /* Handle all other types using sscanf. */
    else
    {
            if( 1 != sscanf( (char *)Gen_In, Gen_Format, Gen_Out, &Gen_ByteCount ) )
            {
                    goto parse_error;
            }

            /* Advance past the data. */
            Gen_In += Gen_ByteCount;
```

```
            /* Advance the format string. */
            Gen_Format += 5;
}

/* Satisfy any remaining non-format characters in the format string. */
while( *Gen_Format && *Gen_In )
{
            /* Scan past trailing white space in the data. */
            while( Gen_In < Gen_InData + Gen_InDataLen )
            {
                        /* Stop at the first non-blank. */
                        if( !ifx_gl_ismblank( Gen_In, 4 ) )
                        {
                                    break;
                        }

                        /* Keep track of how far we've advanced. */
                        Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                        /* Advance to the next character. */
                        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
            }

            if( *(char *)Gen_In != *Gen_Format )
            {
                        goto parse_error;
            }

            /* Keep track of how far we've advanced. */
            Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

            /* Advance to the next character. */
            Gen_In = ifx_gl_mbsnext( Gen_In, 4 );

            /* Advance to the next character. */
            ++Gen_Format;
}

/* Scan past trailing white space in the data. */
while( Gen_In < Gen_InData + Gen_InDataLen )
{
            /* Stop at the first non-blank. */
            if( !ifx_gl_ismblank( Gen_In, 4 ) )
            {
                        break;
```

```
                }

                /* Keep track of how far we've advanced. */
                Gen_ByteCount += ifx_gl_mblen( Gen_In, 4 );

                /* Advance to the next character. */
                Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
        }

        return Gen_In;

parse_error:

        DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG11, 10 );

        /* not reached */
}

mi_integer
Gen_nstrwords
(
gl_mchar_t *        Gen_InData,     /* The data to scan.            */
mi_integer          Gen_InDataLen   /* Length of the data.          */
)
{
        mi_integer      Gen_WordCount;  /* The number of words present.  */
        gl_mchar_t *    Gen_In;         /* Scanning pointer.             */
        char *          Gen_DataCopy;   /* A NULL terminated copy.       */
        enum
        {
                DBDK_QUOTE,
                DBDK_ESCAPE,
                DBDK_WORD,
                DBDK_SPACE
        } Gen_state;    /* Parsing state.           */

        /* Copy the data and NULL terminate it. */
        Gen_DataCopy = (char *)mi_alloc(Gen_InDataLen + 1);
        memcpy(Gen_DataCopy, Gen_InData, Gen_InDataLen);
        Gen_DataCopy[Gen_InDataLen] = (char)'\0';
        Gen_InData = (gl_mchar_t *)Gen_DataCopy;

        /* Begin counting the words. */
        Gen_WordCount = 0;

        /* Set the initial state. */
```

```
Gen_state = DBDK_SPACE;

/* Point to the beginning of the input text. */
Gen_In = Gen_InData;

do
{
        switch ( Gen_state )
        {
                case DBDK_ESCAPE:
                        Gen_state = DBDK_QUOTE;
                        break;

                case DBDK_QUOTE:
                        if( '\"' == *Gen_In )
                        {
                                Gen_state = DBDK_SPACE;
                        }
                        break;

                case DBDK_WORD:
                        if( '\"' == *Gen_In )
                        {
                                Gen_WordCount++, Gen_state = DBDK_QUOTE;
                        }
                        else if( ifx_gl_ismspace( Gen_In, 4 ) )
                        {
                                Gen_state = DBDK_SPACE;
                        }
                        break;

                case DBDK_SPACE:
                        if( '\"' == *Gen_In )
                        {
                                Gen_WordCount++, Gen_state = DBDK_QUOTE;
                        }
                        else if( !ifx_gl_ismspace( Gen_In, 4 ) )
                        {
                                Gen_WordCount++, Gen_state = DBDK_WORD;
                        }
                        break;
        }

        /* Advance the pointer. */
        Gen_In = ifx_gl_mbsnext( Gen_In, 4 );
}
```

```
        while( *Gen_ln != '\0' );

        /* Free the copy of the data. */
        mi_free(Gen_DataCopy);

        return Gen_WordCount;
}

void
Gen_LoadLOFromFile
(
MI_CONNECTION *     Gen_Con,            /* The database connection.    */
char *              Gen_Caller,         /* Name of the calling routine. */
char *              Gen_LOFile,         /* LO file name.               */
MI_LO_HANDLE *      Gen_pLOh            /* Place the LO handle here.   */
)
{
        MI_LO_HANDLE *  Gen_LOh;        /* Place the LO handle here.   */
        MI_LO_FD        Gen_lo_fd;      /* LO descriptor.              */
        MI_LO_SPEC *    Gen_LOspec;     /* LO spec.                    */

        /* Prepare to read the large object from the file. */
        Gen_LOh    = NULL;
        Gen_LOspec = NULL;
        mi_lo_spec_init( Gen_Con, &Gen_LOspec );

        Gen_lo_fd  = mi_lo_from_file( Gen_Con, &Gen_LOh, Gen_LOFile,
                                      MI_LO_CLIENT_FILE, 0, -1, Gen_LOspec );

        /* Check to see if the LO was read successfully. */
        if( Gen_lo_fd >= 0 )
        {
                /* Success! - save the handle. */
                memcpy( Gen_pLOh, Gen_LOh, DBDK_LOHSIZE );

                /* ... and destroy the LO. */
                mi_free( (char *)Gen_LOh );

                /* Close the LO handle. */
                mi_lo_close( Gen_Con, Gen_lo_fd );

                /* ... and free the LO spec. */
                mi_lo_spec_free( Gen_Con, Gen_LOspec );
        }
        else
        {
```

```
                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG3, 10 );
        }
}

void
Gen_StoreLOToFile
(
MI_CONNECTION *     Gen_Con,            /* The database connection.    */
char *              Gen_Caller,         /* Name of the calling routine. */
char *              Gen_LOFile,         /* LO file name.               */
MI_LO_HANDLE *      Gen_pLOh            /* Store this LO               */
)
{
        MI_LO_FD    Gen_LOfd;           /* LO open descriptor.         */
        mi_integer  Gen_LOSize;         /* The LO file size in bytes.  */
        mi_integer  Gen_Valid;          /* Results of the validation.  */
        char *      Gen_OutFile;        /* The generated output file.  */

        /* Validate the LO handle before storing. */
        Gen_Valid = mi_lo_validate( Gen_Con, Gen_pLOh );

        if( Gen_Valid )
        {
                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG4, 10 );
        }

        /* Open the large object. */
        Gen_LOfd = mi_lo_open( Gen_Con, Gen_pLOh, MI_LO_RDWR );

        if( Gen_LOfd == MI_ERROR )
        {
                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG6, 10 );
        }

        Gen_OutFile = (char *)mi_lo_to_file( Gen_Con, Gen_pLOh, Gen_LOFile,
                                             MI_O_CLIENT_FILE | MI_O_RDWR,
                                             &Gen_LOSize);
        if( Gen_OutFile == NULL )
        {
                DBDK_TRACE_ERROR( Gen_Caller, ERRORMESG6, 10 );
        }

        /* Save the file name. */
        strcpy( Gen_LOFile, Gen_OutFile );

        /* Close the open large object. */
```

```
        mi_lo_close( Gen_Con, Gen_LOfd );
}

void
Gen_Trace
(
MI_CONNECTION *     Gen_Con,        /* The database connection.          */
char *              Gen_Caller,     /* Call originated from this routine. */
char *              Gen_FileName,   /* Call originated in this file.     */
mi_integer          Gen_LineNo,     /* Call originated on this line.     */
char *              Gen_MsgNo,      /* ERRORMESG number.                 */
char *              Gen_Class,      /* Tracing class.                    */
mi_integer          Gen_Threshold,  /* Tracing threshold.                */
mi_integer          Gen_MsgType     /* MI_SQL | MI_MESSAGE | DBDK_TRACE. */
)
{
        /* Route the message to the trace file?. */
        if( Gen_MsgType & DBDK_TRACE )
        {
                /* Write the message to the trace file. */
                GL_DPRINTF( Gen_Class,
                        Gen_Threshold,
                        (
                          Gen_MsgNo,
                          "FUNCTION%s", Gen_Caller,     /* Substitute the caller here.    */
                          "FILENAME%s", Gen_FileName,   /* Substitute the file name here. */
                          "LINENO%d",   Gen_LineNo,     /* Substitute the line info here. */
                          MI_LIST_END                   /* Terminate the list!!           */
                        )
                      );
        }

        /* Mask off the mi_db_error_raise flags. */
        Gen_MsgType &= 0xffff;

        /* Route the message back to the user? */
        if( Gen_MsgType )
        {
                /* If requested, also write the message to the user. */
                mi_db_error_raise( Gen_Con,             /* This is the connection handle. */
                                   Gen_MsgType,         /* Route to the user.             */
                                   Gen_MsgNo,           /* Print this message.            */
                                   "FUNCTION%s", Gen_Caller,   /* Substitute the caller here.    */
                                   (char *)NULL );      /* Terminator.                    */
        }
}
```

```
#ifdef __cplusplus

}

#endif
```

```
/*---  ไฟล์ udr.c  ---*/

#ifdef __cplusplus

extern "C"
{

#endif

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <ifxgls.h>
#include <mi.h>
#include "Ocr.h"

mi_integer nearScore
(
OcrType *                   opaque,
mi_lvarchar *               fileName,
MI_FPARAM *                 Gen_fparam      /* Standard info - see DBDK docs.*/
)
{
        MI_CONNECTION *         Gen_Con;        /* The connection handle.          */
        OcrType *               opaque2;
        gl_mchar_t *            Gen_InData;
        mi_integer              Gen_DataLen, count;
        char                    Gen_LOFile[FILENAME_MAX];
        int                     i, j;


        mi_integer              Gen_RetVal;     /* The return value.               */

        /* Get the current connection handle. */
        Gen_Con = mi_open( NULL, NULL, NULL );
        /* Verify that the connection has been established. */
        if( Gen_Con == 0 ) {
                DBDK_TRACE_ERROR( "nearScore", ERRORMESG1, 10 );
        }

        DBDK_TRACE_ENTER( "nearScore" );

        /* ------ {{Your_Code (PreserveSection) BEGIN ------ */
        /* Point to the input data. */
        Gen_InData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *) fileName );
```

```
        /* Get the length of the input string. */
        Gen_DataLen = mi_get_varlen( fileName );
        /* Allocate a new UDT for the return result. */
        opaque2 = (OcrType *)mi_alloc( sizeof( OcrType ) );
        if( opaque2 == 0 ){
                DBDK_TRACE_ERROR( "OcrTypeInput", ERRORMESG2, 10 );
        }

        /* Get the MI_LO_HANDLE data value for ImgBuffer. */
        Gen_InData = Gen_sscanf( Gen_Con, "OcrTypeInput", Gen_InData,
                                                mi_get_varlen( fileName ),
                                                sizeof(Gen_LOFile)-1,
                                                "%s %n",
                                                Gen_LOFile );

        /* Load the large object from the file. */
        Gen_LoadLOFromFile( Gen_Con, "OcrTypeInput", Gen_LOFile,
                &opaque2->ImgBuffer );
        opaque2 = Recorgnize( Gen_Con, opaque2 );

        for(count=i=0; i<6; i++)
                for(j=0; j<5; j++)
                        if(opaque->P[i][j]==opaque2->P[i][j]) count+=2;
                        else if( (opaque->P[i][j]=='S' && opaque2->P[i][j]=='I') ||
                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='S') ||
                                        (opaque->P[i][j]=='V' && opaque2->P[i][j]=='I') ||
                                (opaque->P[i][j]=='I' && opaque2->P[i][j]=='V') ||
                                        (opaque->P[i][j]=='H' && opaque2->P[i][j]=='I') ||
                                        (opaque->P[i][j]=='I' && opaque2->P[i][j]=='H')
                                ) count++;

        Gen_RetVal = count;
        DBDK_TRACE_EXIT( "nearScore" );
        /* Return the function's return value. */
        return Gen_RetVal;
}

#ifdef __cplusplus
}
#endif
```

ภาคผนวก ง

ชุดคำสั่งภาษา SQL ที่ใช้สร้างตารางของโปรแกรมสาธิต 2

```
/*--- create.sql ---*/
create row type ascii_value_type (
        ascii  int,
        charValue  char,
        stringValue  lvarchar
);

create table ascii_value of type ascii_value_type
(primary key (ascii));

create row type ascii_code_type (
        ascii  int,
        font_number int,
        code  OcrType
);

create table ascii_code of type ascii_code_type
(primary key (ascii, font_number));

/*--- person1.sql ---*/

create row type name_type (
        first  varchar(10),
        last  varchar(20)
);

create row type address_type (
        number  varchar(10),
        road  varchar(20),
        province  varchar(20)
);

create row type person_type (
        person_id  varchar(9),
        person_name  name_type,
        person_address  address_type,
        person_age  integer
);

create table person_table of type person_type
( primary key (person_id) );
```

```
/*---  person2.sql  ---*/

create row type code_name_type (
            person_id  varchar(9),
            position  integer,
            code  OcrType
);

create table code_name_table of type code_name_type
( primary key (person_id, position) );

/*---  person3.sql  ---*/

create row type temp_type (
            position  integer,
            code  OcrType
);

create table temp_table of type temp_type
( primary key (position) );
```

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จได้ด้วยดีเนื่องจากได้รับการแนะนำและการสนับสนุนจากบุคคลต่างๆ ซึ่งได้แก่ รศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาที่เป็นผู้ให้ความรู้พื้นฐานหลักการและแนวคิด อ. บัณฑิต พัสยา อ.พิทักษ์ ธรรมวาริน ที่ช่วยประสานงานต่างๆ ให้ คุณ บัณฑิต สื่อวิโรจนกุล ที่ให้ข้อมูล คุณ William W. White ที่ช่วยแนะนำเทคนิควิธีเกี่ยวกับอินฟอร์มิกซ์ และบุคคลสำคัญที่ผู้จัดทำไม่อาจจะลืมได้คือ บิดา มารดาที่ได้เลี้ยงดูอบรมให้กำลังใจเสมอมา สุดท้ายคือเพื่อนๆ และน้องๆ ที่เป็นกำลังใจแก่ผู้จัดทำเป็นอย่างดี ซึ่งผู้จัดทำขอขอบคุณทุกท่านมาไว้ ณ ที่นี้

ผู้จัดทำ

บรรณานุกรม

[1] Stephen K.Hunter:"Cutting to the Chase", Object magazine, August 1997

[2] Stonebraker, M., Object-Relational DBMS, "Component Requiring Modification to Create an Object-Relational Engine", Informix White Paper, November 1996

[3] Informix Inc., " INFORMIX-Universal Server Informix Guide to SQL: Tutorial, Version 9.1",March 1997

[4] Informix Inc., " Extending INFORMIX-Universal Server: Data Types, Version 9.1", March 1997

[5] Informix Inc., " DataBlade Developers Kit User's Guide, Version 3.6", July 1998

[6] Informix Inc., " INFORMIX-Universal Server DataBlade API Programmer's Manual, Version 9.12 ", March 1997

[7] Informix Inc., " INFORMIX-Universal Server Informix Guide to SQL: Syntax, Version 9.1", March 1997

[8] Jeffrey D. Ullman and Jennifer Widom, "Object-Oriented Query Languages", A First Course in Database Systems, Prentice-Hall International, Inc, 1997

[9] Grady Booch, "Object-Oriented Analysis and Design with Applications", Benjamin/Cummings Publishing Company, 1994

[10] ISO-ANSI, "Database Language SQL/Foundation (SQL3)", ISO-ANSI Working Draft, August 1994