ระบบฐานข้อมูลลายนิ้วมือ โดยใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

FINGERPRINT DATABASE USING OBJECT RELATIONAL DATABASE SYSTEM

นายปริญญา ดิษฐจร
นายสหศิลป์ ดรหมั่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2541

ระบบฐานข้อมูลลายนิ้วมือ โดยใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

FINGERPRINT DATABASE USING OBJECT RELATIONAL DATABASE SYSTEM

โดย

นายปริญญา ดิษฐจร

นายสหศิลป์ ดรหมั่น

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบฐานข้อมูลลายนิ้วมือ โดยใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

FINGERPRINT DATABASE USING OBJECT RELATIONAL DATABASE SYSTEM

ผู้จัดทำ

1. นาย ปริญญา คิษฐจร รหัสประจำตัว 39013241
2. นาย สหศิลป์ ดรหมั่น รหัสประจำตัว 39013253

อาจารย์ที่ปรึกษา

(รศ.ดร. ศุภมิตร จิตตะยโศธร)

ระบบฐานข้อมูลลายนิ้วมือ โดยใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

นาย ปริญญา ดิษฐจร 39013241
นาย สหศิลป์ ดรหมั่น 39013253
รศ.ดร.ศุภมิตร จิตตะยโสธร อาจารย์ที่ปรึกษา

บทคัดย่อ

ระบบฐานข้อมูลรีเลชั่นแนล (Relational Database) โดยใช้ภาษา เอส คิว แอล (SQL) ในการสอบถามข้อมูลจากฐานข้อมูลนั้นเป็นที่นิยมใช้อย่างกว้างขวาง ไม่ว่าจะเป็นการประยุกต์ใช้ในงานทางธุรกิจ งานทางด้านวิศวกรรม หรือทางด้านวิทยาศาสตร์ เป็นต้น แต่เมื่องานที่จำเป็นต้องใช้ข้อมูลที่มีชนิดของข้อมูลซับซ้อนสูง (Complex Data Type) ทำให้ ระฐานข้อมูลรีเลชั่นแนล ไม่สามารถที่จะตอบสนองความต้องการของงานประเภทนี้ได้

ระบบฐานข้อมูล เชิงวัตถุสัมพันธ์ (Object Relational Database) ก็เป็นเทคโนโลยีหนึ่งที่พัฒนามาจาก ระบบฐานข้อมูลรีเรชั่นแนล โดยเพิ่มเติมคุณสมบัติของ ออปเจ็ค (Object) ลงไป ซึ่งภาษา เอส คิว แอล ที่จะนำไปใช้จะเป็น เอส คิว แอล ระดับ 3 (SQL 3) และใน โครงงานชิ้นนี้ก็จะได้ใช้ระบบจัดการฐานข้อมูล อินฟอร์มิกซ์ เวอร์ชั่น 9.14 (INFORMIX UNIVERSAL SERVER Version 9.14) ที่มีความสามารถใช้งานกับ เอส คิว แอล 3 ได้เป็นอย่างดี

ในโครงงานชิ้นนี้ได้ใช้ระบบฐานข้อมูลลายนิ้วมือ ซึ่งมีความซับซ้อนของข้อมูลสูง เก็บลงในฐานข้อมูลประเภทออปเจ็ครีเลชั่นแนล โดยเก็บในลักษณะของออปเจ็ค รวมทั้งมีการเขียนเมทธอด (Methods) เพิ่มเติมกับชนิดของออปเจ็คนั้นอีกด้วย

ระบบฐานข้อมูลลายนิ้วมือสามารถตรวจสอบและยืนยันความความเป็นเจ้าของของลายนิ้วมือนั้นได้ ขั้นตอนต่าง ๆ ไม่ว่าจะเป็นการ ค้นหาลายนิ้วมือในฐานข้อมูลและการเพิ่มลายนิ้วมือลงในฐานข้อมูลได้ถูกอธิบายไว้ในโครงงานชิ้นนี้แล้ว โดยเริ่มตั้งแต่การรับรูปภาพลายนิ้วมือเข้ามา หาค่าเอกลักษณ์ (Feature) ทั้งหมดในลายนิ้วมือนั้น จากนั้นจึงทำการค้นหาหรือเพิ่มลงในฐานข้อมูลนั้นได้

Fingerprint Database using Object Relational Database system

Prinya Ditachorn

Sahasilp Dornmun

Assoc.Prof.Dr. Suphamit chittayasothon Advisor

ABTRACT

Relational Database System using SQL to query data from database was populated. In such that, Business Application, Engineering Application, Science Application , etc. have to used them ,but when we need to use high complex data type in some application ,Relational Database System is can not support for this categories.

Object-Relational Database System is the one technology to develop from Relational Database System, by additional Object-Oriented Concept to its. SQL that to used in these, is SQL3 and this project using Informix Universal Server version 9.14 that it has SQL3 implementation in good command.

Fingerprint Database is to used by this project, its have high complex data type and keep in Object-Relational Database System. Fingerprint Data Type is keep in Object that have method or function to operate with its.

Fingerprint Database System in this project can to verify and identify authentication of each fingerprint, and also each process that search fingerprint in database, insert fingerprint to database ,their process was described by this paper, they start at input fingerprint images and find feature of each fingerprint in all feature and then insert into or search from Fingerprint Database.

สารบัญ

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 วัตถุประสงค์และความเป็นมา

เนื่องจากว่าเทคโนโลยีของฐานข้อมูลได้มีการพัฒนาไปไกลมากและได้มีการพัฒนาอย่างไม่หยุดยั้ง ซึ่งระบบฐานข้อมูลในปัจจุบันที่นิยมกันอย่างแพร่หลายเป็นอย่างมากก็คือ ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) ซึ่งได้ใช้ภาษา SQL เพื่อทำการ Query กับฐานข้อมูลรวมทั้งมีการจัดการต่าง ๆ ที่เกี่ยวกับฐานข้อมูลโดยผ่านทางการใช้งานของภาษา SQL ซึ่งในปัจจุบัน ภาษา SQL ก็ได้มีการพัฒนาคุณสมบัติต่าง ๆ เพิ่มขึ้นมาด้วยเช่นกัน

และเทคโนโลยีอีกตัวหนึ่งที่มีการนำมาประยุกต์ใช้กันอย่างกว้างขวางเป็นอย่างมากในอุตสาหกรรมคอมพิวเตอร์ในทุก ๆ ด้านก็คือ แนวคิดในเชิงวัตถุ (Object-Oriented Concept) ซึ่งได้มีการประยุกต์ใช้กับโปรแกรมประยุกต์ (Application Program) ต่าง ๆ รวมไปถึงได้มีการผสมผสานเทคโนโลยีนี้เข้ากับระบบฐานข้อมูลที่เป็นที่นิยมกันอยู่ในปัจจุบัน นั่นก็เป็นที่มาของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ก็ได้มีการพัฒนาภาษา SQL ควบคู่ไปด้วย ซึ่งเป็นภาษา SQL 3 นั่นเองที่จะนำมาใช้กับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ การรวมกันของสองเทคโนโลยีนี้ก็เพื่อตอบสนองต่อความต้องการของ Application ต่าง ๆ ที่นับวันยิ่งมีความซับซ้อนมากยิ่งขึ้น ความซับซ้อนในที่นี้ก็คือความซับซ้อนของข้อมูลที่จะทำการเก็บลงในฐานข้อมูล ซึ่งในระบบฐานข้อมูลเชิงสัมพันธ์นั้นไม่สามารถตอบสนองได้

ลายนิ้วมือก็เป็นอีกตัวอย่างหนึ่งที่เมื่อนำมาประยุกต์ใช้งานกับฐานข้อมูลนั้นจะมีการเก็บของชนิดข้อมูลของลักษณะนี้ที่มีความซับซ้อนมาก ดังนั้นในการเก็บเราจึงเลือกใช้คุณสมบัติของ Object ที่มีอยู่ในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ เพื่อการเก็บคุณลักษณะต่าง ๆ ที่มีอยู่ในลายนิ้วมือหนึ่ง ๆ รวมทั้งได้มีการเขียน method หรือ ฟังก์ชั่นของ Object หนึ่ง ๆ ฝังอยู่ที่ตัวฐานข้อมูล ซึ่งก็จะตรงกับคุณสมบัติของ Object ด้วย

ในปัจจุบันระบบฐานข้อมูลลายนิ้วมือที่มีใช้กันอยู่ในประเทศไทยนั้นยังมีราคาสูงมาก เช่น ระบบฐานข้อมูลลายนิ้วมือที่สำนักงานตำรวจแห่งชาตินั้นมีมูลค่าถึง 500 กว่าล้านบาท เป็นต้น เราจึงได้ทำการใช้เทคโนโลยีของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์นี้สร้างระบบฐานข้อมูลลายนิ้วมือขึ้นมา

1.2 วิธีการดำเนินงาน

การทดลองของเราได้ใช้ระบบฐานข้อมูลของ Informix ที่มีคุณสมบัติสามารถที่จะใช้ภาษา SQL 3 ได้ ซึ่ง Version ที่ได้นำมาใช้งานคือ Informix Universal Server Version 9.14 ซึ่งได้มีคุณสมบัติของ Object เพิ่มเข้ามา และยังมีโมดูลที่ช่วยในการพัฒนาชนิดของข้อมูลและฟังก์ชั่นที่จะทำการเขียนอีกนั่น

คือ DataBlade Module ซึ่งเป็นโมดูลที่ Informix ได้ทำการพัฒนาขึ้นมาเพื่อใช้งานกับ Informix ใน Version นี้

หลักการทำงานของระบบฐานข้อมูลลายนิ้วมือได้ถูกอธิบายไว้ในปริญญานิพนธ์ในฉบับนี้แล้ว ซึ่งกระบวนการทำงานต่างๆ เริ่มตั้งแต่กระบวนการอินพุทของลายนิ้วมือ การทำ Image Processing การค้นหาเอกลักษณ์ของลายนิ้วมือ การ Insert ลงฐานข้อมูล การสร้างชนิดของข้อมูลที่เป็นลายนิ้วมือเพื่อทำการบันทึก การ Search หาลายนิ้วมือที่ต้องการ

แต่ในบทแรก ๆ นั้นจะเป็นการอธิบายถึงทฤษฎีพื้นฐานก่อน หลังจากนั้นจึงอธิบายถึงขั้นตอนในการสร้างต่าง ๆ ในโครงงานนี้เราได้ใช้ DataBlade Tools เพื่อใช้ในการพัฒนาชนิดของข้อมูล และใช้ Visual C++ V5.0 ในการเขียนฟังก์ชั่นที่ทำการฝังอยู่ในส่วนของ Server และได้ใช้ Visual Basic V5.0 ในการสร้างกระบวนการทาง Image Processing ซึ่งจะทำงานอยู่ในฝั่ง Client และได้ใช้ Data Director for VB version 3.5 ซึ่งเป็น tool อีกตัวหนึ่งของ Informix ที่ใช้ในการติดต่อกับ Server ก็คือมีทั้งการ Query และการ Insert ข้อมูลกับตัว Database Server

บทที่ 2

แนวความคิดของ Object - Oriented

Object oriented มีกำเนิดมาจากแนวความคิดของคำต่างๆ ดังต่อไปนี้

Objects

Class

Inheritance

Message passing

Polymorphism

2.1 Objects

ในมุมมองของระบบซอฟต์แวร์ที่ผ่านมาจะเกิดจากส่วนประกอบต่างๆ ของซอฟต์แวร์ มีการรวมกันของข้อมูลซึ่งจะใช้แทนข้อมูลข่าวสารบางอย่าง และเซตของโปรซีเยอร์ ต่างๆที่ใช้ในการจัดการกับดาต้านั้น การทำงานต่างๆในระบบซอฟต์แวร์จะเกิดขึ้นได้โดย การส่งผ่านข้อมูลบางอย่างไปให้โปรซีเยอร์ใช้ในการทำงาน ข้อมูลและโปรซีเยอร์จะเป็นส่วนประกอบที่แยกออกจากกัน ซึ่งเมื่อนำมารวมกันแล้วจะทำให้เกิดระบบซอฟต์แวร์ขึ้น

ในมุมมองของระบบซอฟต์แวร์แบบ Object oriented จะมีหน่วยๆ หนึ่ง (entity) ที่ถูกเรียกว่าเป็น object (วัตถุ) ซึ่งจะประกอบไปด้วยทั้ง ข้อมูล และ โปรซีเยอร์ Object สามารถที่จะถูกกระทำและถูกจัดการได้ อย่างไรก็ตาม Object สามารถที่จะอธิบายถึงการกระทำได้เช่นเดียวกันกับโปรซีเยอร์

2.2 Classes

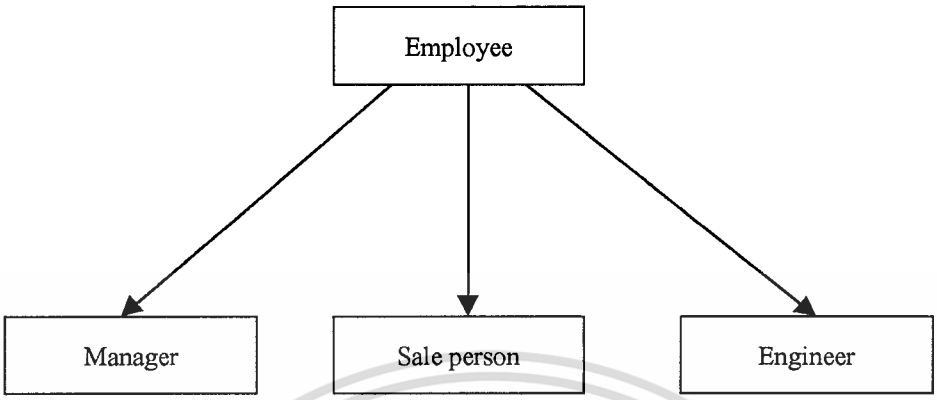
ในขณะที่ Object เป็นหน่วยพื้นฐานหน่วยหนึ่งที่ใช้อธิบายถึงลักษณะของโปรแกรม และ Object ทั้งหลายไม่ได้เป็นองค์ประกอบมูลฐานของระบบ ในการอธิบายลักษณะของ Object ที่เหมือนๆ กันสามารถทำได้ในแนวทางเดียวกัน การอธิบายลักษณะของ Object จะถูกเรียกว่าเป็น Class ทุกๆ Object จะเป็น Instance ของ Class

Class จะเป็นตัวกำหนด Method ต่างๆที่ถูกใช้โดย Object และจะประกาศชนิดของดาต้าต่างๆ ที่ Object สามารถที่จะใช้ได้ Class จะเป็นเหมือนพิมพ์เขียว (Blueprint) ที่ถูกใช้ในการสร้าง Object

2.3 Inheritance

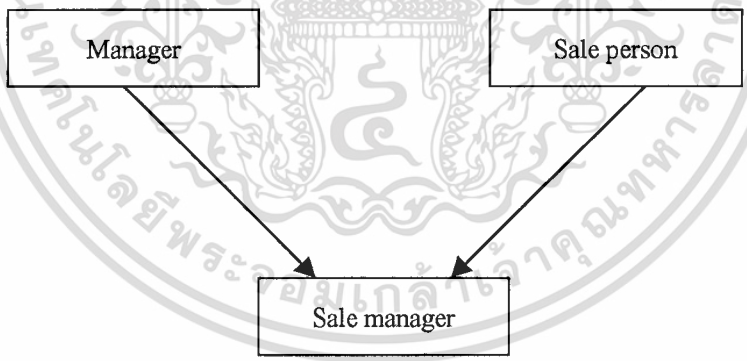
บ่อยครั้งที่ Object ดูคล้ายคลึงกัน แต่มันจะมีคุณสมบัติและ Method ที่ไม่เหมือนกัน ในสภาพการณ์เช่นนี้มีประโยชน์มากในการนำเอาความสามารถของคุณสมบัติการสืบทอดมาใช้ Inheritance จะ

เป็นกลไกในการจัดการเกี่ยวกับ Class และการใช้ Class ร่วมกัน โดยทั่วๆไป Inheritance จะมีรูปแบบที่มีชื่อเรียกว่า Single Inheritance และ Multiple Inheritance



รูปที่ 2.1 ตัวอย่างของ Single Inheritance

บริษัทจะมีพนักงาน พนักงานเป็นได้ทั้ง Manager, Sale person และ Engineer ทุกคนมีคุณสมบัติที่แน่นอนเหมือนกัน (เช่น หมายเลขพนักงาน, วันที่เข้าทำงาน เป็นต้น) ดังนั้นทั้ง Manager person และ Engineer สามารถที่จะสืบทอดคุณสมบัติทั่วๆไปเหล่านั้นได้จาก Class employee



รูปที่ 2.2 ตัวอย่างของ Multiple Inheritance

Multiple Inheritance อนุญาตให้ Class สามารถที่จะสืบทอดคุณสมบัติมาจาก Superclass มากกว่าหนึ่ง Superclass ได้ กฎเกณฑ์ในการที่จะกำหนด Multiple Inheritance จะต้องจัดการเกี่ยวกับ Conflict (ความขัดแย้ง) ต่างๆที่อาจจะเกิดขึ้นได้ เช่น ชื่อที่ Conflict กัน โดยที่ชื่อที่เหมือนกันต้องใช้แทนคุณลักษณะ หรือ Method ที่ต่างกันของสอง Parent ที่ต่างกัน และทั้ง คุณลักษณะ และ Method เหล่านี้ก็จะสืบทอดคุณสมบัติมายัง Child class ที่เหมือนกัน ในภาษาโปรแกรมมิ่งต้องแก้ปัญหา Conflict ชนิดนี้ได้

ในภาษาโปรแกรมมิ่งที่ต่างกันก็จะ Support Inheritance ในระดับที่ไม่เหมือนกัน เช่น การใช้ชื่อที่ต่างกัน ระดับการเขียนโปรแกรมในการ Support Inheritance มีความสำคัญในการพิจารณาเลือกภาษาโปรแกรมมิ่งเกี่ยวกับ Object oriented

2.4 Message Passing

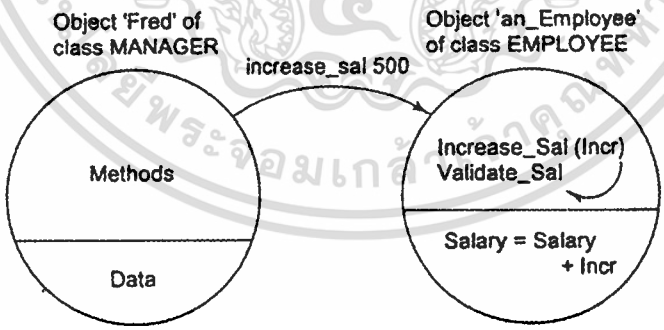
การติดต่อระหว่าง Object หนึ่งกับอีก Object หนึ่ง ทำได้โดยการส่งผ่าน Message โดยจะมีข้อมูลที่เป็นชื่อที่ใช้ในการกำหนดปลายทางและบางทีอาจจะมี Argument หรือ Parameter บางตัว ซึ่งๆก็คล้ายๆกับการเรียกใช้ฟังก์ชั่นหรือโปรซีเยอร์ของระบบซอฟต์แวร์แบบแต่ก่อน เมื่อ Object ได้รับ Message แล้วจะเป็นผลทำให้เกิดการเรียกใช้ Method ที่เหมาะสมภายใน Object นั้นๆ

การส่งผ่าน Message เหมือนกับการเรียกใช้ฟังก์ชั่นหรือโปรซีเยอร์ของระบบซอฟต์แวร์แบบแต่ก่อน แต่อย่างไรก็ดีในการส่งผ่าน Message ของระบบซอฟต์แวร์แบบ Object oriented จะมีความซับซ้อนมากกว่า ในการเรียกใช้โปรซีเยอร์ของระบบซอฟต์แวร์แบบแต่ก่อน เช่น ภาษา C หรือ Pascal ผู้ส่งจะต้องรู้ที่อยู่ของผู้รับ โดยที่การเรียกใช้โปรซีเยอร์หรือฟังก์ชั่นนั้นจะเป็นผลทำให้ Program counter กระโดดจากคำสั่งที่กำลังทำงานอยู่ไปยังตำแหน่งที่ถูกเรียก

ในระบบซอฟต์แวร์แบบ Object oriented การส่งผ่าน Message จาก Object หนึ่งไปยังอีก Object หนึ่งสามารถที่จะมีขั้นตอนต่างๆได้มากเพราะว่าเป็นไปไม่ได้ที่เราจะรู้ว่า ผู้ส่งทราบที่อยู่ของผู้รับหรือเปล่า ดังนั้นจะมีการอ้างถึง Dynamic binding

ตัวอย่างของ Message passing

An_Employee Increase_Sal 500



รูปที่ 2.3 Message Passing

Message นี้จะเกิดขึ้นจาก Method ๆ หนึ่งภายใน Object ที่มีชื่อว่า A_Manager (จาก Class Manager) Message นี้จะถูกส่งไปยัง Object ที่มีชื่อว่า An_Employee (เป็น พารามิเตอร์หนึ่งของ Method) ในที่นี้ Message จะมีชื่อว่า Increase_Sal 500 ซึ่งจะใช้เพิ่มเงินเดือนของพนักงานในบริษัทเป็น 500 (นี่คือ พารามิเตอร์ของ Method)

โดยทั่วไปแล้วชื่อของ Method จะตอบสนองต่อชื่อของ Message ในตัวอย่างข้างบนนี้ Method ที่ถูกกระทำจะชื่อ Increase_Sal ซึ่งมีความสัมพันธ์กับ Class ที่ใช้กำหนด Object An_Employee , Method นี้จะถูกกระทำโดยการ binding ชื่อ Message ไปยัง Method ของ Object

2.5 Polymorphism

เป็นความสามารถของ Object ในการที่จะจัดการกับ Object ที่ต่างชนิดกัน (คำว่า Polymorphic มีความหมายถึง มีรูปแบบได้หลายๆชนิด)

Polymorphism ทำให้ง่ายในการที่จะนำ Code ที่เหมือนกันใน Instance ที่ต่างกันกลับมาใช้ใหม่ได้ และสามารถทำการเปลี่ยนแปลงเล็กๆน้อยๆเพื่อทำให้ดีขึ้นได้

บทที่ 3

ชนิดข้อมูล Object ใน SQL3 (Tuple object in SQL3)

ภาษา SQL จะใช้กับ relation เป็นหลัก ใน SQL3 นั้นจะเพิ่มคุณสมบัติของ object เข้าไปแต่ยังคงความเป็น relation ไว้เป็นหลักเช่นเดิม ซึ่ง object ใน SQL3 ที่เพิ่มขึ้นมามีดังนี้

1. *Row objects* เป็น tuple โดยหลัก

2. *Abstract Data Types (ADT)* ใช้เป็น object ทั่ว ๆ ไป และสามารถใช้เป็น component ของ tuple ได้ด้วย

3.1 Row Types

ใน SQL3 มันสามารถกำหนดเป็น type ของ tuple ได้ และมีลักษณะคล้ายคลึงกับ class ของ object มาก ในการประกาศ row-type มีดังนี้

1. ใช้ keyword ว่า CREATE ROW TYPE
2. ให้ชื่อของ row-type
3. มี attribute list อยู่ในวงเล็บ และ type ของ attribute นั้น ๆ

รูปแบบของการประกาศ row-type จะมีลักษณะเช่นนี้

CREATE ROW TYPE (< component declarations >)

การประกาศ relation โดยใช้ row-type เราอาจจะประกาศได้มากกว่าหนึ่ง relation โดย tuple ก็คือ type ต่าง ๆ เหล่านั้น ซึ่งรูปแบบก็จะมีดังต่อไปนี้

CREATE TABLE (table name) OF TYPE (row-type name)

ในการ access component ต่าง ๆ ที่อยู่ใน row-type type นั้นเนื่องจากว่า component ใน SQL3 สามารถที่จะมี structure ภายในตัวมันเองได้ จึงจำเป็นที่จะต้องสามารถเข้าถึง component ของตัวมันเองได้ SQL3 จะใช้สัญลักษณ์ double-dot ในการอ้างอิงถึง เปรียบเสมือน single-dot ใน C

ใน Informix Universal Server ก็จะมี data type ที่เป็น row type ด้วยเหมือนกัน การใช้คำสั่ง SQL ก็จะเช่นเดียวกันกับใน SQL3 ซึ่งจะมีรายละเอียดเพิ่มเติมในส่วนต่อไป

```
CREATE ROW TYPE AddressType(
        Street    CHAR(50),
        City      CHAR(20)
```

```
);
CREATE ROW TYPE StarType(
        Name    CHAR(30),
        Address         AddressType
);
```

รูปที่ 3.1 ตัวอย่างการประกาศ Row Type

3.2 Abstract Data Types (ADT)

ใน row-type จะไม่มีคุณสมบัติ encapsulate ซึ่งจะต้องมีใน object ใน SQL3 เมื่อเทียบชนิดของ class ที่มีคุณสมบัติ encapsulate ด้วยจะใช้ตัว ADT ADT จะถูกนำไปใช้เป็น component ใน tuple และจะไม่ใช้เป็น tuple แต่โครงสร้างภายในจะมีลักษณะเช่นเดียวกับ tuple

รูปแบบของการ define ADT จะมีดังนี้

```
1)  CREATE TYPE <type name> (
2)          List of attribute and their types
3)          Optional declaration of = and < function for the type
4)          Declaration of function (method) for the type
5)  );
```

ในบรรทัดที่ 1) จะเป็น create statement และมีชื่อของ ADT นั้น ๆ บรรทัดที่ 2) แสดงถึง attribute name ต่าง ๆ และ type ของแต่ละ attribute โดยจะใช้ comma เป็นตัวแบ่ง บรรทัดที่ 3) จะเป็น optional ในการประกาศเพื่อใช้กับ operator ที่ใช้ในการ compare = และ < เช่นรูปแบบของการประกาศ equality function คือ

EQUALS <name of function implementing equality>

และ function < ก็มีลักษณะเช่นเดียวกัน เพียงแต่เปลี่ยน keyword คำว่า EQUALS เป็น LESS THAN แทน เราจะสังเกตเห็นว่า ใน comparison operator จะมีอีก ฉะนั้นเราจะใช้ operator ที่ define ไว้เหล่านี้เป็นตัวสร้างแทน ดังเช่น $\leq$ ก็คือ "= or <" และ > ก็คือ "not <" ในการ define ถึง = และ < จะถูกนำไปใช้ในWHERE clause เพื่อเปรียบเทียบค่าที่จะเป็นไปได้

บรรทัด 4) จะ define ถึง function ของ ADT (เปรียบเสมือน method ของ object) ใน SQL3 เตรียม built-in function ไว้สำหรับทุก ADT และ function ต่าง ๆ เหล่านี้ ไม่จำเป็นที่จะต้องประกาศออกมาหรือ ถูก define เอาไว้ function เหล่านั้น ก็คือ

1. constructor function ซึ่งจะ return new object ของ type นั้น ๆ โดย attribute ทั้งหมดจะถูกกำหนดค่าเริ่มต้นให้เป็น NULL ทั้งหมด เช่น ถ้า T เป็นชื่อของ ADT หนึ่ง T() จะเป็น constructor function

2. observer function จะมีหน้าที่ในการ return value ของแต่ละ attribute ใน ADT เช่น ถ้า A เป็นชื่อ attribute และ X เป็นตัวแปรที่มีค่าเป็น object ของ ADT A(X) ก็คือค่า attribute ของ A ใน object X เราสามารถใช้ X.A แทนได้ด้วยเหมือนกัน

3. mutator function ในแต่ละ attribute จะทำการ set ค่า ให้เป็นค่าใหม่ที่ต้องการ จะเห็นว่าถ้าจะให้มีความ encapsulate จำเป็นที่จะต้องมีการ กำหนด permission ในการ execute function ต่าง ๆ เหล่านี้ ใน SQL3 จะใช้ EXECUTE statement เพื่อกำหนดการทำงานของ function

function อื่น ๆ จะถูก define ไว้ภายในหรือภายนอกคำสั่ง CREATE TYPE ก็ได้ ถ้า function ที่ทำการประกาศภายนอก อาจจะต้องกลับมาประกาศภายในก็ได้เพราะจำเป็นต้องใช้ built-in function ต่าง ๆ เหล่านั้น

ใน Informix Universal Server จะใช้ Opaque Type ในการ represent เป็น ADT ใน SQL3 ซึ่งจะมีคุณสมบัติของ encapsulateด้วย ในการสร้าง Opaque ใน Informix Universal Server จะต้องสร้าง Support function ด้วย ซึ่งเปรียบเสมือนกับ built-in function ของ SQL3 และยังสามารถสร้าง function เพิ่มเติมขึ้นมาเพื่อนำไปเรียกใช้ใน SQL statement ได้ รายละเอียดต่าง ๆ ของ Opaque Type จะกล่าวถึงต่อไปในภายหลัง

```
CREATE TYPE AdressADT (
    Street          CHAR(50),
    City            CHAR(20),
    EQUALS          addrEq,
    LESS THAN addrLT
            Other functions could be declared here
);
```

รูปที่ 3.2 ตัวอย่างการประกาศ Abstract Data Type

3.3 การ Define Methods ของ ADT

หลังจากที่มีการสร้าง attribute ของ ADT แล้ว ก็จะมีส่วนที่เป็น function declaration รูปแบบที่ใช้ก็มีดังนี้

FUNCTION <name> (<argument>) RETURNS <type> ;

ในแต่ละ argument จะประกอบด้วย ชื่อตัวแปรและ type ของตัวแปร ซึ่งจะแบ่ง argument ด้วย comma

function มีสองแบบคือ internal และ external External function คือ function ที่เขียนด้วย host language และ signature ของแต่ละ function (เปรียบเสมือนกับ argument) ที่มีอยู่ในส่วนของการกำหนดของ ADT internal function จะถูกเขียนด้วยส่วนขยายเพิ่มเติมของภาษา SQL ซึ่ง option ในบางส่วนต่อไปนี้จะรวมอยู่ในส่วนขยายของทั้ง SQL2 และ query language บางส่วนของ SQL3

- := จะถูกใช้เป็น assignment operator
- การประกาศ local variable ใน function จะทำโดยประกาศชื่อของตัวแปร นำหน้าด้วย colon (:) และจะตามหลังด้วย type ของตัวแปร
- ใช้ dot operator ในการ access ถึง component ใน structure
- ค่า Boolean สามารถจะใช้ในประโยค WHERE clause ได้
- BEGIN และ END จะใช้เป็นตัวรวมคำสั่งหลาย ๆ คำสั่งไว้ใน body ของ function

ส่วน external function ที่ถูกเขียนด้วย host language จะต้องมีส่วนของ signature อยู่ด้วย ซึ่งจะต้องบอกถึงว่าใช้ภาษา host language ชนิดใดด้วยในการเขียน รูปแบบในการใช้งานมีดังนี้

DECLARE EXTERNAL <function name> <signature>
LANGUAGE <language name>

ใน Informix Universal Server จะใช้คำสั่ง CREATE FUNCTION statement ในการสร้าง function ขึ้นมา โดยสามารถกระทำได้ทั้งที่เป็น external และ internal ขึ้นอยู่กับการกำหนด option ต่าง ๆ ของ statement

internal function ที่สร้างโดย Informix Universal Server จะใช้ภาษา SPL (Store Procedure Language) ในการสร้าง ซึ่งจะมีทั้งการกำหนด return type ,กำหนด argument ต่าง ๆ และคำสั่งที่สามารถใช้ได้ใน SQL รวมทั้งคำสั่งที่เพิ่มขึ้นมา เช่น คำสั่งในการทำ loop,คำสั่งทางเงื่อนไข ส่วน external function จะถูกกำหนดโดย option ของ statement ที่กำหนดเป็น EXTERNAL และจะถูกเขียนด้วยภาษา host language เช่น ภาษา C เป็นต้น แต่อาจจะต้องจำกัดภาษาที่จะนำมาใช้ด้วย

```
FUNCTION AddressADT( :s CHAR(50), :c CHAR(20)
                RETURNS AddressADT;
        :a AddressADT;
        BEGIN
                :a := AddressADT();
                :a.street := :s;
                :a.city := :c;
                RETURN :a;
        END;
```

```
FUNCTION addrEq( :a1  AddressADT, :a2  AddressADT)
        RETURNS  BOOLEAN;
RETURN ( :a1.street = :a2.street  AND
        :a1.city = :a2.city);

FUNCTION addrLT( :a1  AddressADT, :a2  AddressADT)
        RETURNS  BOOLEAN;
RETURN (( :a1.city <  :a2.street ) OR
        ( :a1.city = :a2.city)  AND  :a1.street < :a2.street ));
```

รูปที่ 3.3 ตัวอย่างการประกาศ Method ของ ADT

บทที่ 4

ชนิดข้อมูลใน Informix Universal Server

Informix Universal server จะสนับสนุนการใช้งานชนิดของข้อมูล หรือ Data type อยู่ 3 ประเภท คือ

- Built-in Data type
- User-defined Data type
- Complex Data type

4.1 Built-in Data type

มีการใช้งานชนิดของข้อมูลต่างๆ ดังต่อไปนี้

4.1.1 Character data type

ชนิดของข้อมูลที่เป็นตัวอักษร (Character) ซึ่งประกอบไปด้วย : CHAR, CHARACTER VARYING (หรือVARCHAR) และ LVARCHAR

ชนิดของข้อมูลที่เป็นแบบ NCHAR และ NVARCHAR จะใช้เก็บข้อมูลที่เป็นตัวอักษรได้ด้วย

Universal server จะสนับสนุนภาษาโปรแกรมที่เป็นแบบ Global (Global language), ภาษาโปรแกรม Global language support (หรือ GLS) จะมีลักษณะพิเศษที่ให้ Universal server จัดการกับภาษา, กลุ่มของ Code ที่แตกต่างกันได้ GLS จะสนับสนุนการใช้งานในหัวข้อต่างๆ ดังต่อไปนี้

- จัดเรียงลำดับของตัวอักษร
- การกำหนดตัวอักษรตัวเล็ก, ตัวใหญ่
- อักษรที่ไม่ใช่รหัส แอสกี้ รวมทั้งตัวอักษรที่มีหลายๆ ไบท์
- แบบของข้อมูลที่เป็น numeric, monetary, date และ time

4.1.2 Numeric type

- ชนิดของข้อมูลที่เป็นตัวเลข (Numeric) ซึ่งประกอบไปด้วยชนิดข้อมูลที่เป็นตัวเลขจริงๆ ดังนี้ : --DECIMAL,MONEY, SMALLINT, INTERGER, INT8, SERIAL และ SERIAL8
- ชนิดข้อมูลที่เป็นค่าประมาณตัวเลขได้แก่ SMALLFLOAT และ FLOAT

4.1.3 Large-object data type

มีอยู่ 2 ชนิดคือ Simple-large-object type ซึ่งใช้เก็บ Text และ Byte และ Smart-large object type ซึ่งประกอบไปด้วย CLOB และ PLOB

4.1.4 Time data type

ประกอบไปด้วย DATE, DATETIME และ INTERVAL

4.1.5 Miscellaneous data type

ประกอบไปด้วย BOOLEAN

4.2 User-Define data type

มีการสร้างชนิดของข้อมูลที่กำหนดโดยผู้ใช้ขึ้นมาใหม่ได้ เพิ่มเติมจาก Built-in data type ที่มีใน Database server และมีความยืดหยุ่นมากในการจัดเก็บและจัดการกับชนิดของข้อมูลเหล่านั้น Universal server มีการแนะนำชนิดของข้อมูลใหม่, Opaque type และ Dinstinct type ดังนี้

4.2.1 Opaque type

เป็นชนิดของข้อมูลที่เป็นพื้นฐานที่เก็บค่าเพียงค่าเดียวและไม่สามารถแบ่งแยกส่วนประกอบโดย Database server ได้ Opaque ถูกสร้างมาจากโครงสร้างของภาษา C และ Routine ต่างๆที่ถูกเขียนในภาษา C จะยอมให้ Database server สนับสนุนการใช้งานชนิดของข้อมูลเหล่านั้น Routine ของภาษา C ซึ่งสนับสนุนการใช้งานชนิดข้อมูล Opaque type จะส่งข้อมูลที่เป็นโครงสร้างของ Opaque type นั้นมาที่ Database server เพื่อที่จะเก็บข้อมูลเหล่านั้นไว้ใน Database server โดยที่ Database server ไม่ได้มีการแปลงข้อมูลที่เป็นของสร้างเหล่านั้นแต่จะเก็บค่าข้อมูลที่เป็น Byte ต่อ Byte จากหน่วยความจำลงใน Database Routine ต่างๆที่สนับสนุนการใช้งาน Opaque type จะสร้าง Operation ต่างๆ ขึ้นมาทำงานกับข้อมูลของ Opaque type เช่น เปรียบเทียบ Instance 2 ตัวของ Data type, แปลงชนิดของข้อมูลจาก Opaque type เป็น Data type ชนิดอื่นๆ หรือแสดงค่าข้อมูล Instance ของ Data type Database server จะมีการเรียกใช้งาน Routine ต่างๆ ที่มีการสนับสนุนการทำงานของ Operation เหล่านั้น ซึ่งจะมีการส่ง โครงสร้างในรูปแบบของ C หรือ Pointer ไปที่ Database server ในรูปแบบของพารามิเตอร์ต่างๆ

4.2.2 Distinct type

Distinct type เป็นชนิดของข้อมูลที่มีรูปแบบการเก็บเหมือนกับ Opaque type แต่สิ่งที่แตกต่างกันคือ ชื่อของมันและไม่สามารถที่จะใช้แทน data source type ได้ เช่น มี distinct type ใหม่ชื่อ decnum ที่ถูกสร้างขึ้นคล้ายๆกันกับชนิดข้อมูลที่เป็นแบบ real ทุกๆ routine ที่ทำงานกับค่า real จะมี database server คอยจัดการทำสำเนาของค่า decnum ไว้ แต่อย่างไรก็ตามค่าของ decnum และ real ไม่สามารถที่จะ บวก, ลบ และ เปรียบเทียบกับค่าข้อมูลชนิดอื่นได้ นอกจากว่าจะมีการ casting ค่าข้อมูลค่าหนึ่งไปเป็นค่าข้อมูลอีกค่าหนึ่งอย่างชัดเจน Casting function จะมีหน้าที่ในการเปลี่ยน data type ชนิดหนึ่งไปเป็น data type อีกชนิดหนึ่ง

4.3 Complex data type

Complex data type ปกติจะประกอบไปด้วยชนิดของข้อมูลอื่นๆรวมกัน เช่น เราสามารถสร้าง Complex data type ที่ประกอบไปด้วย Built-in data type, Opaque data type, Distinct data type หรือ Complex data type อื่นๆ ได้ สิ่งที่แตกต่างที่สำคัญมากระหว่าง Complex data type และ User-defined

type คือ เราสามารถที่จะเข้าถึงข้อมูลในแต่ละส่วนประกอบของ Complex data type และจัดการกับส่วนประกอบเหล่านั้นโดย SQL statements ได้

4.3.1 Collection type

คือ กลุ่มของ element ที่มีชนิดข้อมูลที่เป็นชนิดเดียวกัน Collection type ให้เราเก็บข้อมูลและจัดการกับข้อมูลที่เป็น collection data ภายในแถวเดียวของตาราง Collection type ประกอบด้วยองค์ประกอบ 2 อย่างคือ

1. Type constructors ซึ่งประกอบไปด้วย set, multiset และ list

- Set คือ Collection data type ซึ่งทุกๆ element ใน set จะไม่มีลำดับ และจะไม่มีค่าข้อมูลภายใน set ที่เหมือนกัน

- List คือ Collection data type ซึ่งทุกๆ element ใน set จะมีลำดับ และ สามารถมีข้อมูลที่เหมือนกันได้

- MULTISET คือ Collection data type ซึ่งทุกๆ element ใน set ไม่มีลำดับ และสามารถมีข้อมูลที่เหมือนกันได้

2. Element type ซึ่งเป็นตัวกำหนดชนิดข้อมูลของ Collection type ทุกๆ element ของ Collection type สามารถมีชนิดข้อมูลได้ทุกอย่าง ยกเว้น data type ที่เป็น SERIAL และ SERIAL8

4.3.2 Row type

คือ การเรียงกันของ element ตั้งแต่ 1 element ขึ้นไป ซึ่งจะถูกเรียกว่าเป็น field แต่ละ field จะมีชื่อ field และชนิดข้อมูลของ field ทุกๆ field ของ Row type เปรียบได้เหมือนกันกับ columns ของ Table แต่มีสิ่งที่แตกต่างกันมาก คือ row type จะไม่มีค่า default บน field และไม่สามารถที่จะกำหนด กฎบังคับความถูกต้องบน field ได้ และเราไม่สามารถใช้ filed เหล่านั้นบน table. ได้ แต่จะใช้ได้กับ row type ได้เท่านั้น Row type ประกอบด้วย 2 ชนิด คือ

1. Name row type

Name row type เป็นกลุ่มของ field ที่ถูกกำหนดภายใต้ชื่อๆ เดียว Files ๆ หนึ่งจะอ้างถึงเป็นส่วนประกอบของ row type ครั้งแรกเมื่อมีการสร้าง name row type ชื่อที่เราตั้งให้ name row type จะเป็นชื่อที่มีชื่อเดียวใน database ซึ่งจะไม่ซ้ำกับใคร เราจะมีการสร้าง name row type เมื่อเรามีความต้องการ type ๆ หนึ่ง ซึ่งเก็บส่วนประกอบของข้อมูลที่เราต้องการที่จะ access เช่น เราต้องการสร้าง name row type ซึ่งเก็บรายละเอียดเกี่ยวกับ address ซึ่งผู้ใช้บางทีอาจจะต้องการ access ไปที่แต่ละองค์ประกอบของ address เช่น street, city, state และ postal code ดังนั้นเมื่อเราสร้าง name row type ชื่อ address type ผู้ใช้สามารถที่จะ access ไปที่แต่ละ filed ได้โดยตรง

ตัวอย่างของ name row type ที่ชื่อ person_t

```
CREATE ROW TYPE person_t
(       name        VARCHAR(30) NOT NULL,
        address     VARCHAR(20),
        city        VARCHAR(20),
        state       CHAR(2),
        zip         VARCHAR(9),
        bdate       DATE
);
```

รูปที่ 4.1 ตัวอย่างการประกาศ Row type ใน Informix

เป็นตัวอย่างของ name row type ชื่อ person_t ที่ประกอบไปด้วย 6 filed คือ : name, address, city, state, zip, และ bdate เราสามารถที่จะกำหนด data type ของ filed ต่างๆ เป็นข้อมูลชนิดอื่นได้ เช่น TEXT, BYTE เป็นต้น

2. Unnamed row type

Unnamed row type เป็นกลุ่มของ field ที่ถูกกำหนดโดยโครงสร้างของมัน ไม่เหมือนกันกับ name row type เราสามารถที่จะกำหนดให้ unnamed row type กับ table ได้ Unnamed row type ใช้กำหนด type ให้กับ column หรือ field ได้เท่านั้น

4.3.3 Inheritance properties

Universal server จะสนับสนุนการสืบทอดคุณสมบัติ (หรือ inheritance) ของข้อมูลที่เป็น name row type และ type table Inheritance คือขั้นตอนในการที่อนุญาติให้ type หรือ table สามารถที่จะได้รับคุณสมบัติทั้งหมดมาจาก type หรือ table อื่นๆได้ คุณสมบัติของ inheritance สามารถให้เราเปลี่ยนแปลงแก้ไขคุณสมบัติเดิมที่เราสืบทอดมาให้ดีขึ้นได้ และสามารถที่จะเพิ่มเติมคุณสมบัติใหม่ๆเข้าไปได้ Inheritance มีอยู่ 2 ชนิดคือ

- Type inheritance

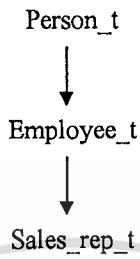
Type inheritance ใช้งานได้กับชนิดข้อมูลที่เป็น name row type ได้เท่านั้น เราสามารถที่จะใช้ inheritance ในกลุ่มของ name row type ในรูปแบบของลำดับชั้นของ name row type ได้ (type hierarchy) ซึ่งในแต่ละ subtype จะสืบทอดคุณสมบัติของ data field และ behavior ของ type ที่เป็น supper type ภายใต้รูปแบบที่ถูกกำหนดขึ้น Type hierarchy จะสนับสนุนข้อดีของการทำงานดังต่อไปนี้

- ช่วยเหลือการสร้าง data type ที่เป็นแบบ module
- รับประกันการนำกลับมาใช้ใหม่ขององค์ประกอบของ schema
- รับประกันได้ว่า field ของข้อมูลจะไม่หายจากการเกิดอุบัติเหตุ

- สามารถให้ type สืบทอดคุณสมบัติของ routine มาจาก type อื่นได้

Defineing a Type Hirerarchy

จากรูปแสดงลำดับชั้นของ type หรือที่เราเรียกว่า type hierarchy ได้ดังนี้คือ



รูปที่ 4.2 แสดงลำดับชั้นของ Type ใน Informix

Supperype ที่อยู่บนสุดของลำดับชั้นของ type จะประกอบไปด้วยทุกๆ field ซึ่งทุกๆ subtype ที่อยู่ภายใต้สามารถที่จะ inherit ค่าเหล่านั้นได้ จะต้องมีการสร้าง supertype ก่อนที่จะมีการสร้าง subtype ตัวอย่างต่อไปนี้เป็นการสร้าง suppertype ชื่อ person_t ของลำดับชั้นของ type

```
CREATE ROW TYPE person_t
(       name    VARCHAR(30) NOT NULL,
        address VARCHAR(20),
        city            VARCHAR(20),
        state           CHAR(2),
        zip             INTERGER,
        bdate           DATE
);
```

รูปที่ 4.3 การประกาศ Supper type ใน Informix

ในการสร้าง subtype จะกำหนด keyword ชื่อ under และชื่อของ suppertype ที่จะ inherit มา ตัวอย่างต่อไปนี้เป็นการสร้าง subtype ชื่อ employee_t ซึ่งสืบทอดคุณสมบัติของทุกๆ field ที่อยู่ใน person_t และจะมีการเพิ่มเติมคุณสมบัติใหม่ก็คือ field ชื่อ salary และ manager ซึ่งเป็น field ที่ไม่มีอยู่ใน person_t

```
CREATE ROW TYPE employee_t
(       salary          INTERGER,
        manager VARCHAR(30)
)
UNDER person_t;
```

รูปที่ 4.4 การประกาศ Sub type ใน Informix

- Table inheritance

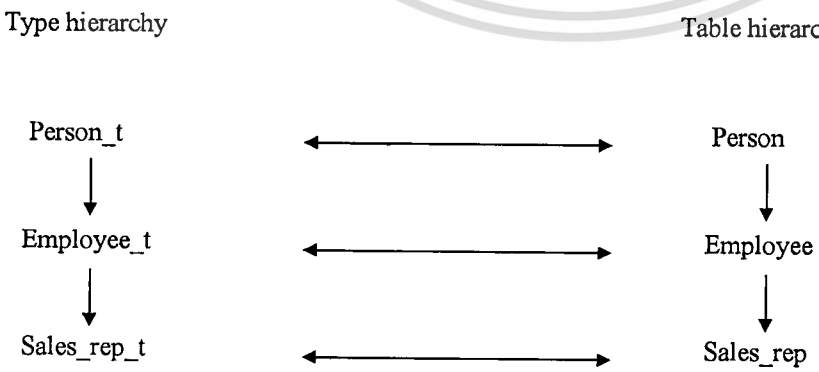
Table ที่ถูกกำหนดเป็น name row type เท่านั้นที่สนับสนุนการทำงานที่เป็น table inheritance ซึ่งอนุญาติให้สามารถสืบทอดคุณสมบัติ ทั้ง data และ behavior มาจาก suppertable ตามลำดับชั้นของ table hirarchy โดยที่ table hierachy เป็นการกำหนดความสัมพันธ์ระหว่าง table ที่ subtable จะสืบทอดคุณสมบัติของ supertable ข้อดีของ table inheritance คือ

- ช่วยเหลือการสร้างรูปแบบของ data ที่เป็น module
- รับประกันความถูกต้องของการนำกลับมาใช้ใหม่ขององค์ประกอบของ schema
- สามารถที่จะสร้างการสอบถามข้อมูลในบาง table หรือ ทุกๆ table ใน table hierarchy

ใน table hierarchy นั้น subtable สามารถสืบทอดคุณสมบัติต่างๆจาก supertable ได้ดังนี้

- ทุกๆ constraint definition (primary key, unique, และ referential constraints)
- storage option
- ทุกๆ trigger
-Access method

จากรูปแสดงความสัมพันธ์ระหว่าง Table และ Type hierarchy



รูปที่ 4.5 แสดงความสัมพันธ์ระหว่าง Table และ Type hirerachy

ตัวอย่างการสร้าง table

```
CREATE TABLE person OF TYPE person_t;

CREATE TABLE employee OF TYPE employee_t
UNDER person;

CREATE TABLE sales_rep OF TYPE sales_rep_t
UNDER employee;
```

รูปที่ 4.6 ตัวอย่างการประกาศ Table ใน Informix

เป็นการสร้าง Table person, employee และ sales_rep บน type person_t, employee_t และ sale_rep_t ตามลำดับ โดยที่ทุกๆ type ที่อยู่ใน type hierarchy จะสอดคล้องกับ table ที่มีอยู่ใน table hierarchy ยกตัวอย่างเช่น employee table จะสืบทอดคุณสมบัติมาจาก person table และ employee_t type จะสืบทอดคุณสมบัติของ type มาจาก person_t type และ ในทำนองเดียวกัน sales_rep table ที่สืบทอดคุณสมบัติมาจาก employee table ทำให้ sales_rep_t type สืบทอดคุณสมบัติมาจาก employee_t type เช่นเดียวกัน

4.4 User-define routines

User-define routine (UDR) เป็น routine ที่ถูกกำหนดและใส่เข้าไปใน system catalog table ของ database และเราสามารถที่จะเรียกใช้ routine เหล่านั้นได้โดย SQL statement หรือ routine อื่นๆ เราต้องมีการสร้าง user-define routine เพื่อมาทำงานกับ user-defined data type และ complex data type

UDR เป็นได้ทั้ง function หรือ procedure โดยทั่วไปแล้ว function เป็น routine ซึ่งรับค่าเป็น set ของ argument และส่งค่ากับเป็น set ของ value เราสามารถใช้ function ใน SQL expression ได้ ส่วน procedure จะรับค่าเป็น set ของ argument แต่จะไม่ส่งค่าใดๆกลับมาเลย เราไม่สามารถใช้ procedure ใน SQL expression ได้

Routine ของ UDR สามารถเป็นได้ดังนี้คือ

- Store procedure language (SPL) routine เป็น routine ที่ส่วนของ body ถูกเขียนด้วยภาษา SPL
- External routine เป็น routine ใดๆ ที่ส่วนของ body ถูกเขียนด้วยภาษาอื่นๆ มากกว่าภาษา SPL (เช่น ภาษา C) ปกติแล้วเราใช้ external routine ดำเนินการกระทำ operation ต่างๆ กับ user-defined data type

4.5 Access Methods

Universal Server จะมีวิธีการในการ Access อยู่สองวิธีการ

1. Primary access methods
2. Secondary access methods

4.5.1 Primary Access Methods

Universal Server จะมีการสนับสนุนพื้นที่การใช้งานจากภายนอกหรือ external spaces (extspaces) ซึ่งก็คือพื้นที่ในการเก็บข้อมูลที่ตัว Server ไม่สามารถเข้าไปจัดการข้อมูลได้โดยตรง เราสามารถระบุพื้นที่ภายนอกเป็นพื้นที่ในการเก็บ table ซึ่งก็จะเท่ากับว่าเราสร้าง primary access method ไว้ primary access method ก็คือกลุ่มฟังก์ชั่นการทำงานของ database server เพื่อการเข้าถึงข้อมูล (access) และการวางแบบแผนข้อมูล (manipulate) ซึ่งข้อมูลนั้นมีการเก็บไว้ภายนอกตัว database sever

Universal Server พัฒนา primary access method โดยนำ Virtual Table Interface (VTI) มาใช้เพื่อความสะดวกในการเข้าถึงข้อมูล เราสามารถจะเข้าถึงชนิดของข้อมูลใด ๆ ก็ได้ดังต่อไปนี้ โดยใช้วิธีการ primary access method

- Tables ในฐานข้อมูลของผู้ผลิตรายอื่น ๆ
- ข้อมูลที่ถูกเก็บใน Sequential files เช่น Tape Backup
- การบันทึกข้อมูลผ่าน network ในระยะไกล

เนื่องจากว่า primary access method สามารถที่จะถูกนำไปใช้ใน client applications ได้ ซึ่ง applications ต่าง ๆ เหล่านี้ไม่ต้องการ การเข้าถึงข้อมูลในระดับลึก ๆ ภายในโมดูลต่าง ๆ ของ database server เลย และการทำเช่นนี้จะเป็นการง่ายกว่าที่จะต้องมีการหาพื้นที่ในการเก็บข้อมูลใหม่ หรือการใช้วิธีสร้าง index เอง primary access method สามารถจะรวมเอาข้อมูลทั้งหลายที่ต่างกันในแต่ละโครงสร้างต่าง ๆ ทั้งหมดมาไว้ภายในระบบ object-relational เพียงระบบเดียวได้

VTI จะรวมเอา access method ต่าง ๆ มาไว้ในการเรียกใช้ เราจะเขียนและบันทึก (register) กลุ่มเป้าหมายของการทำงาน access-method ซึ่งจะเข้าถึงข้อมูลที่อยู่ในฐานข้อมูลในระยะไกลได้ ฟังก์ชั่นเหล่านี้จะถูกบันทึกลงใน Universal Server เมื่อเราใช้คำสั่ง SQL

CREATE ACCESS_METHOD

เมื่อเราสร้าง table เราก็จะกำหนด access method และถูกนำไปใช้เมื่อมีการใช้คำสั่ง SQL กับ table นั้น

4.5.2 Secondary Access Method

Universal Server นี้มีการใช้ secondary access methods ดังต่อไปนี้

- Generic B-trees
- R-trees

4.5.2.1 Generic B-Trees

B-Tree index จะรวบรวมข้อมูลเกี่ยวกับ index (index information) ทั้งหลายที่ใช้กับ database นั้น โดยจะมีการจัดเรียงแบบลำดับชั้นต้นไม้ลงมา (hierarchy of pages) Universal Server จะใช้ B-Tree index เก็บค่าต่าง ๆ ดังต่อไปนี้

- คอลัมน์ที่เป็น built-in data types ซึ่งก็ได้แก่ CHARACTER, DATETIME, INTEGER, FLOAT ฯลฯ (จะใช้ traditional B-Tree index)
- data type ที่เคยกำหนดโดยผู้ใช้ในลักษณะหนึ่งมิติ (Traditional one-dimensional user-defined data types) (จะใช้ generic B-Tree index)
- ค่าที่ returns โดยฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นมาเอง (User-defined function returns) (จะใช้ functional index) เราสามารถที่จะสร้าง functional index กับผลลัพธ์ที่ returns มาจากฟังก์ชั่นได้มากกว่าหนึ่งคอลัมน์ ซึ่งฟังก์ชั่นจะต้องเป็นฟังก์ชั่นที่ผู้ใช้กำหนดขึ้นมาเอง (User-defined function) เมื่อมีการสร้าง functional index Universal server จะคำนวณค่า key ของฟังก์ชั่นนั้น และทำการบันทึกลงใน index และสามารถที่จะหาตำแหน่งในการเก็บค่าที่ return มาจากฟังก์ชั่นลงใน index อย่างเหมาะสมได้ โดยจะไม่มีการทำงานของแต่ละฟังก์ชั่นในแต่ละคอลัมน์

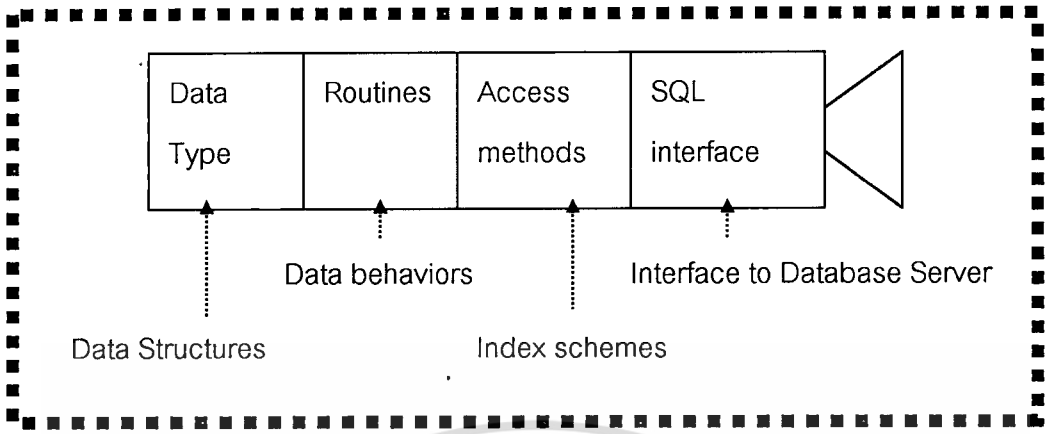
4.5.2.2 R-Trees

Universal server จะนำโครงสร้างของ index แบบใหม่ขึ้นมาเพื่อใช้กับข้อมูลในลักษณะสองมิติ, สามมิติ และอื่น ๆ ซึ่งเรียกว่า spatial data index ชนิดใหม่นี้จะเรียกว่า R-Tree index โดยเจ้า R-Tree index นี้จะใช้ concept ที่เรียกว่า bounding box ก็คือกลุ่มของค่า coordinates ที่จะบรรจุไว้มากกว่าหนึ่ง objects ขึ้นไป และ objects สามารถจะเกี่ยวข้องกับ bounding box ได้มากกว่าหนึ่งเช่นกัน

4.6 DataBlade Module

DataBlade module คือการรวมของ database objects และ code ที่เพิ่มเข้ามาใน database server โดยจะเพิ่มฟังก์ชั่นการทำงานต่าง ๆ เข้าไป DataBlade module จะทำให้ database server มีการสนับสนุน data types ชนิดใหม่ ๆ ขึ้นมา โดยที่ data type ชนิดนั้นมีการใช้ built-in data type ได้ เราอาจจะคิดได้ว่า DataBlade module เป็นเสมือน package ตัวหนึ่งใน object-oriented เปรียบได้กับ class ภายในภาษา C++ โดยจะเป็นข้อมูลชนิดพิเศษที่มีคุณสมบัติ encapsulate ข้อมูลแบบนี้ก็เช่น image เป็นต้น

ในรูปข้างล่างนี้จะแสดงส่วนประกอบของ DataBlade module



รูปที่ 4.7 แสดงส่วนประกอบของ DataBlade Module

DataBlade Module Component	Description
Data Types	Data type ก็คือ Use-defined data type หรือ การรวมกันของ User–defined data type ค่าของ user-defined data type จะมีการเก็บและตรวจสอบโดยใช้ query หรือการเรียก routine และมีการผ่านอาร์กิวเมนต์ให้กับ ฟังก์ชั่น การใช้ index เช่นเดียวกับ built-in data type ส่วนประกอบของ data type ถูกกำหนดให้เป็น data structure ใน Universal server
Routines	Routines สามารถที่จะ operate กับ data type ที่ถูกกำหนดขึ้นโดย developer เปรียบเสมือนกับ data type ชนิดใด ๆ ที่ database server รู้จัก รวมถึง data type จาก DataBlade module อื่นได้ด้วย ส่วนประกอบของ routine จะถูกกำหนดเป็น data behavior ใน Universal server
Access Methods	Access methods จะ operate บน table และ index ที่มีการจัดการโดย database server โดย developer สามารถที่จะใช้ index ของ data type ชนิดใหม่กับ access method ที่มีอยู่แล้วได้ หรือสร้าง access method เองได้ ในส่วนนี้จะถูกกำหนดให้เป็น index schemes ใน Universal server
SQL Interface	SQL interface ก็คือการรวมกันของฟังก์ชั่นที่ตรงตามมาตรฐานและนำออกไปใช้ในการให้บริการ query ได้ SQL interface จะควบคุม DataBlade module ให้แชร์การให้บริการกับ DataBlade module ตัวอื่นได้ด้วย ในส่วนนี้จะถูกกำหนดให้เป็น interface ใน Universal server

ตารางที่ 4.1 อธิบายส่วนประกอบของ DataBlade Module

ตารางที่ 4.1 จะอธิบายถึงส่วนประกอบทั้ง 4 ส่วนของ DataBlade module DataBlade module ก็คือ โมดูลของ software ที่เป็นมาตรฐานที่จะเพิ่มเข้าสู่ฐานข้อมูลเพื่อขยายความสามารถของตัวมันเองให้เพิ่มขึ้น เราสามารถใช้ DataBlade module ของ Informix หรือจากผู้ผลิตรายอื่น ๆ หรือเราสามารถที่จะสร้าง DataBlade module เองได้

4.7 Large Object Support

Universal server จะยอมให้มีการบันทึก object ขนาดใหญ่ได้หรือที่เรียกว่า large object รวมทั้งมีการ access และ manipulate ถึงรายการต่าง ๆ ที่ดึงมาจาก database server ได้ large object เป็น data object ที่ถูกบันทึกลงในคอลัมน์ของ table แบบ logically แต่ในทาง physically จะถูกบันทึกอย่างอิสระจากคอลัมน์นั้น โดยจะมีการบันทึกแยกออกมาจาก table เพราะจะต้องใช้พื้นที่ในการบันทึกที่มีขนาดใหญ่มาก ๆ Universal server จะมีการใช้ large object สองอย่างคือ smart large object และ simple large object

4.8 Smart Large Object

Smart Large Object จะยอมให้มีการ seek, การอ่าน, การเขียน ลงในส่วนต่าง ๆ หรือใน segment ของ object ได้ โดยจะมีส่วนประกอบเป็นชนิดของข้อมูลดังต่อไปนี้

- **Character Large Object (CLOB)** CLOB ก็คือ smart large object ที่มีการบันทึกเป็น text item อย่างเช่น PostScript, หรือ HTML file เป็นต้น CLOB สามารถจะบันทึก และดึงข้อมูลมาเป็นส่วน ๆ ได้ และก็จะมี database properties ต่าง ๆ ด้วยเช่น การ recovery และ transaction rollback เป็นต้น
- **Binary Large Object (BLOB)** BLOB มีการบันทึกข้อมูลชนิดใด ๆ ที่เป็น binary ซึ่งก็รวมทั้ง image ด้วย BLOB สามารถจะบันทึก และดึงข้อมูลมาเป็นส่วน ๆ ได้ และก็จะมี database properties ต่าง ๆ ด้วยเช่น การ recovery และ transaction rollback เป็นต้น

เราสามารถใช้ smart large object ในการเก็บข้อมูลที่ผู้ใช้กำหนดขึ้นมาเองได้ (user-defined data type) เช่น video และ audio clips, รูปภาพ, Document ข้อความต่าง ๆ ที่มีขนาดใหญ่ และ spatial object (แบบแปลน และแผนที่ ฯลฯ) เป็นต้น Universal server จะบันทึก object ขนาดใหญ่นี้ลงใน sbspaces ซึ่งก็คือ การรวมพื้นที่ในการเก็บข้อมูลที่จะจัดไว้อย่างเฉพาะเพื่อการบันทึก smart large object

เราสามารถที่จะสร้าง reference ไปสู่ object ขนาดใหญ่ และเก็บ reference นั้นเป็น เหมือน row ในฐานข้อมูลได้ด้วย ตัว object เองก็สามารถจะฝังตัวเองไว้ภายนอกฐานข้อมูลได้ ตัวอย่างเช่น การใช้ระบบ file (หรือ สามารถจะเป็นคอลัมน์ที่เป็น CLOB, BLOB ได้) reference จะเป็นตัวกำหนดชนิด หรือ access protocol ของ object และมีการชี้ไปยังตำแหน่งที่ตัวมันถูกบันทึกไว้ เราจะเข้าถึง object ขนาดใหญ่โดยผ่านทาง reference ของตัวมันเองไปยังฟังก์ชั่นการทำงานของ LOB Locator

4.9 Simple Large Objects

Simple Large Object คือประเภทของ large object ที่มีการขนาดได้ไม่เกิน 2^{31} bytes แต่ในทางปฏิบัติข้อจำกัดจะขึ้นอยู่กับขนาดของความจุของ disk Universal server จะมีการใช้อยู่สองอย่างคือ

- BYTE เก็บเป็น binary data
- TEXT เก็บเป็น text data

Simple large object จะต่างกับ smart large object ตรงที่ไม่สามารถเข้าถึงในลักษณะrandom access ได้ ถ้าเราจะต้องมีการขนถ่ายข้อมูลที่เป็นแบบ simple large object ระหว่าง client กับ database server จะต้องมีการขนถ่ายทั้งหมดที่เป็นค่าของ BYTE หรือ TEXT

บทที่ 5

ชนิดข้อมูล Opaque Data type

Opaque data type คือ ชนิดของข้อมูลที่เป็น atomic ซึ่งเรากำหนดขึ้นมาสำหรับใช้งานใน database โดยที่ database server จะรู้จักชื่อของ Opaque type แต่จะไม่รู้เกี่ยวกับโครงสร้างภายในของข้อมูลชนิดนี้ ไม่เหมือนกันกับชนิดของข้อมูลที่เป็นแบบ built-in data type ที่เรารู้รูปแบบของโครงสร้างภายในข้อมูล แต่สำหรับ Opaque type เป็นชนิดของข้อมูลที่ซ่อนรายละเอียดภายในเอาไว้ (encapsulate) ซึ่งทำให้ Universal server ไม่รู้รายละเอียดข้อมูลภายในของชนิดข้อมูลที่เป็นแบบ Opaque type

เมื่อเรากำหนดชนิดข้อมูลที่เป็นแบบ Opaque data type แล้ว เราต้องเพิ่มชนิดข้อมูลส่วนนี้เข้าไปในระบบข้อมูลภายใน database server ชนิดข้อมูล Opaque type ใหม่ที่เรากำหนดขึ้นนี้จะถูกใช้งานเหมือนกันกับชนิดของข้อมูลที่เป็น built-in data type ใน database server ในการที่จะกำหนดชนิดข้อมูล Opaque type ใน database server เราต้องมีการใช้รูปแบบของภาษา C ในการจัดเตรียมข้อมูลต่างๆ ดังต่อไปนี้

1. โครงสร้างการเก็บข้อมูลภายในของ Opaque data type
2. Function ที่ทำงานร่วมกับ Opaque type ที่ทำให้ database server สามารถติดต่อกับข้อมูลภายใน Opaque type ได้
3. Routine อื่นๆ ที่สามารถถูกเรียกใช้ได้โดย Support function หรือเรียกใช้งานโดย end user ในการทำงานกับ Opaque type ได้

5.1 Internal Structure

ในการสร้าง Opaque type ขั้นแรกเราต้องมีการกำหนดชนิดของข้อมูลที่ใช้ในการเก็บ data ภายใน Opaque type โครงสร้างของข้อมูลนี้จะถูกเรียกว่าเป็น *internal structure* ของ Opaque data type เพราะว่า internal structure จะเป็นตัวบอกว่าข้อมูลจะถูกเก็บบน disk ได้อย่างไร และ function ที่เราเขียนขึ้นเพื่อดำเนินการกับ Opaque type จะใช้งานกับ internal structure นี้ โดยที่ database server จะไม่เคยเห็น internal structure นี้เลย เราสามารถสร้าง internal structure ได้เหมือนกันกับการสร้างโครงสร้างข้อมูลในรูปแบบของภาษา C

เราสามารถที่จะกำหนดรูปแบบของ internal structure สำหรับชนิดข้อมูล Opaque type ได้ 2 รูปแบบดังนี้คือ

- A fixed-length opaque type
- A varying-length opaque type

5.1.1 A Fixed-length Opaque type

A fixed-length Opaque type คือ internal structure ที่มีขนาดสำหรับค่าข้อมูลต่างๆ ที่อยู่ภายใน Opaque type คงที่ และ fixed-length Opaque type จะมีประโยชน์สำหรับข้อมูลที่เราสามารถที่จะกำหนดขนาดให้คงที่ได้ เช่น ชนิดของข้อมูลที่เป็น numeric เป็นต้น เราจะมีการกำหนดขนาดของชนิดข้อมูล Opaque type ในขณะที่เรานำค่าข้อมูลชนิดนี้ไปเก็บไว้ใน database

5.1.2 A varying-Length Opaque type

A varying-length Opaque type คือ internal structure ที่มีขนาดของค่าข้อมูลที่แตกต่างกันสำหรับแต่ละชนิดของค่าข้อมูลภายใน Opaque type และมีประโยชน์มากสำหรับการเก็บค่าข้อมูลที่ใช้แทนสิ่งต่างๆมากมาย (multi representational data) เช่น image โดยที่ image จะมีขนาดเปลี่ยนไปจาก image หนึ่งไปยังอีก image หนึ่ง และแน่นอนถ้าเราต้องมีการเก็บค่าขนาดของข้อมูลที่เพิ่มขึ้นภายใน Opaque type ที่มีขนาดจำกัด เราอาจต้องมีการใช้ชนิดข้อมูลที่เป็นแบบ smart large object ช่วยในกรณีที่ขนาดของข้อมูลใหญ่มากๆเกินกว่าขนาดของ Opaque type

5.2 Support Functions

Support function ของชนิดข้อมูล Opaque type จะเป็นตัวบอกให้ database server ทราบว่าจะมีการติดต่อกับ Internal structure ของชนิดข้อมูล Opaque type ได้อย่างไร เนื่องจากชนิดข้อมูล Opaque type จะถูกซ่อนรายละเอียดเอาไว้ทำให้ database server ไม่สามารถที่จะรู้รายละเอียดของ Internal structure ได้ และ Support function ที่เราเขียนขึ้นจะเป็นตัวติดต่อกับ Internal structure นี้ โดยที่ database server จะมีการเรียกใช้ Support function เหล่านี้ในการดำเนินการโต้ตอบกับชนิดข้อมูล Opaque type

ตารางต่อไปนี้จะเป็นการสรุป Support function ของชนิดข้อมูล Opaque type

Function	Purpose
input	เปลี่ยนการแทนชนิดข้อมูล Opaque type จากภายนอกให้อยู่ในรูปแบบของการแทนที่ชนิดข้อมูลภายใน Opaque type
output	เปลี่ยนการแทนชนิดข้อมูล Opaque type จากภายนอกให้อยู่ในรูปแบบของการแทนที่ชนิดข้อมูลภายใน Opaque type
receive	เปลี่ยนการแทนชนิดข้อมูล Opaque type จากภายในบนเครื่อง Clientให้อยู่ในรูปแบบของการแทนที่ชนิดข้อมูลภายในของ Opaque type บน Server computer
send	เปลี่ยนการแทนชนิดข้อมูล Opaque type จากภายในบนเครื่อง Server computerให้อยู่ในรูปแบบของการแทนที่ชนิดข้อมูลภายในของ Opaque type บน เครื่อง Client
import	ดำเนินการในการใช้งาน Opaque type เมื่อมีการ import Opaque type มาจากข้างนอก
export	ดำเนินการในการใช้งาน Opaque type เมื่อมีการ export Opaque type ส่งไปยังภายนอก

importbinary	ดำเนินการในการใช้งาน Opaque type เมื่อมีการ import Opaque type ภายใน internal structure
exportbinary	ดำเนินการในการใช้งาน Opaque type เมื่อมีการ export Opaque type ภายใน internal structure
assign()	ดำเนินงานต่างๆบน Opaque type ก่อนที่จะเก็บลงบน disk
destroy()	ดำเนินงานต่างๆบน Opaque type ทีจำเป็นก่อนที่จะนำ row ของข้อมูลชนิดนี้ออกไป
lohandles()	Return list ของชนิดข้อมูล smart large object ที่ฝังอยู่ใน Opaque type
compare()	เปรียบเทียบค่าข้อมูล 2 ค่าของชนิดข้อมูล Opaque type ในระหว่างการเรียงลำดับ

ตารางที่ 5.1 แสดงถึง Support function ที่ใช้ใน Opaque data type

5.3 Additional SQL-Invoked Routines

Support function จะเป็นตัวจัดเตรียมการทำงานเบื้องต้นในการที่จะให้ database server มีการติดต่อกับชนิดข้อมูล Opaque type แต่อย่างไรก็ตามเราอาจต้องมีการเขียน Routine ต่างๆ เพิ่มขึ้นในการที่จะทำงานกับชนิดข้อมูล Opaque type ซึ่งจะมี function ต่างๆ ดังนี้

- Built-in function
- Aggregate function
- Operator function
- End-user routine

5.3.1 Built-in Functions

A built-in function คือ function ที่ถูกกำหนดขึ้นมาแล้วซึ่ง Universal server จะจัดเตรียมการใช้งานในรูปแบบของคำสั่งของ Sql โดยที่ Universal server จะ support การใช้งาน built-in function กับรูปแบบของ built-in data type ในการที่จะให้ built-in function ทำงานร่วมกับ Opaque data type เราต้องมีการเขียน function ขึ้นมาใหม่ที่จะสามารถรับ parameter ที่เป็นข้อมูลชนิด Opaque type ได้

5.3.2 Aggregrate Function

Aggregate function จะ return ค่าเพียงค่าเดียวจาก set ของการสอบถามข้อมูล Universal server จะสนับสนุนการใช้งาน Aggregate function กับชนิดข้อมูล Opaque type ดังต่อไปนี้

- Count
- Min
- Max

5.3.3 Operator Function

A operator function คือ function ที่ผู้ใช้กำหนดขึ้นที่สอดคล้องกับสัญลักษณ์ตัวดำเนินการทาง opretor ในการที่จะให้ operator function ทำงานร่วมกับ Opaque data type เราต้องมีการเขียน function ขึ้นมาใหม่ที่จะสามารถรับ parameter ที่เป็นข้อมูลชนิด Opaque type ได้

5.3.4 End-User Routines

Universal server อนุญาติให้เรากำหนด Sql-invoke function หรือ Procedure ที่ end-user สามารถเรียกใช้ในรูปแบบของคำสั่ง Sql ได้ End-user routine ต่างๆ นี้จะมีการจัดเตรียมการทำงานต่างๆ ที่เพิ่มขึ้นซึ่ง end-user จะต้องมีการทำงานกับชนิดข้อมูล Opaque type ตัวอย่างของ End-user routine มีดังนี้คือ

- เป็น function ที่ return ค่าเฉพาะภายในชนิดข้อมูล Opaque type เนื่องจากว่า ข้อมูลชนิด Opaque type ถูกซ่อนรายละเอียดเอาไว้มีเพียง end-user function เท่านั้นที่สามารถให้ user เข้าถึง field ข้อมูลของ internal structure ได้
- เป็น Casting functions

  Support function หลายๆ function จะสนับสนุนการใช้งาน casting function สำหรับ basic data type ซึ่ง database server ใช้งาน เราสามารถที่จะเขียน casting function ระหว่างชนิดข้อมูล Opaque type กับชนิดข้อมูลอื่นๆ (built-in, opaque, or complex) ของ database ได้
- เป็น function หรือ procedure ที่ทำงานร่วมกันกับชนิดข้อมูล Opaque type และ ถ้าการทำงานไหนเกิดบ่อยมากกับชนิดข้อมูล Opaque type เราต้องมีการเขียน end-user routine เพื่อที่จะดำเนินการกับงานนั้น

5.4 ขั้นตอนในการสร้าง Opaque data type

1. สร้างโครงสร้างภายในของข้อมูล (internal structure) ด้วยโครงสร้างของภาษา C สำหรับ Opaque data type
2. เขียน function ที่สนับสนุนการใช้งาน Opaque type ด้วยรูปแบบของภาษา C
3. นำ Opaque data type ใส่ลงไปใน database ด้วยคำสั่ง CREATE OPAQUE TYPE
4. นำ Function ที่สนับสนุนการใช้งาน Opaque type ใส่ลงใน database ด้วยคำสั่ง CREATE FUNCTION
5. จัดการวิธีการเข้าถึง Opaque data type และ Function ที่สนับสนุนการใช้งานของมันด้วยคำสั่ง GRANT
6. เขียน Function ของ Sql ที่สนับสนุนการใช้งานของ Opaque data type
7. จัดเตรียมวิธีการเข้าถึง Opaque data type แบบอื่นๆ

โดยที่การทำงานของแต่ละขั้นตอนโดยละเอียดมีดังต่อไปนี้ คือ

5.4.1 Creating the Internal structure

Internal sturcture ของ Opaque data type เป็นรูปแบบโครงสร้างของภาษา C โดยใช้รูปแบบของ C **typedefs** ใน internal structure ซึ่ง Datablade API จะมีการให้ใช้งาน field ที่มีขนาดไม่คงที่ขึ้นอยู่กับรูปแบบของแต่ละ field

Internal structure จะเป็นชื่อที่ไม่ซ้ำกันของ Opaque data type โดยที่ Informix จะแนะนำให้เราตั้งชื่อที่ไม่ซ้ำกันก่อนสำหรับ Opaque data type นั้น โดยที่เราสามารถจะคิดชื่อของแต่ละสมาชิกใน internal structure ได้ก่อน ข้อตกลงของการตั้งชื่อของ Opaque data type ตามแบบของ informix จะเพิ่ม string_t ต่อท้ายที่ชื่อของ structure เช่น ชนิดของข้อมูลเป็น Circle_t จะเป็นโครงสร้างที่เก็บค่าต่างๆ ของ Opaque type

เมื่อเรามีการสร้าง internal structure จะมีผลกระทบกับขนาดของโครงสร้างข้อมูลดังนี้

- ขนาดของ final structure ของ Opaque data type ใหม่
- Opaque type จะเก็บในหน่วยความจำได้อย่างไร
- Routine ที่ผู้ใช้กำหนดขึ้นจะส่งผ่านข้อมูลไปให้ Opaque type โดย database server ได้อย่างไร

Final structure size

ในการที่จะประหยัดพื้นที่ของ database เราจำเป็นต้องออกแบบ internal structure ให้กระทัดรัดที่สุดเท่าที่จะเป็นไปได้

เราสามารถที่จะกำหนดขนาดของ final size ของ internal structure ด้วยคำสั่ง INTERNAL LENGTH ในการสร้าง Opaque type ซึ่งสามารถทำได้ 2 วิธีดังนี้คือ

- กำหนดขนาดจริงๆ ของ internal structure ในรูปแบบที่เป็น byte ซึ่งจะเป็นการกำหนดขนาดที่คงที่ของ Opaque data type
- กำหนดขนาดของ internal structure ด้วยคำสั่ง VARIABLE ซึ่งจะเป็นการกำหนดขนาดที่เปลี่ยนแปลงได้ของ Opaque data type

A Fixed-Length Opaque Data Type

เมื่อเรากำหนดขนาดจริงๆของ internal structure จะเป็นการสร้าง Fixed-length opaque type ขนาดของ fixed-length opaque type ต้องเท่ากันกับค่าของฟังก์ชั่น **sizeof()** ในภาษา C ซึ่งจะส่งค่ากลับมายัง internal structure

ใน Compilers ส่วนมาก ฟังก์ชั่น sizeof() จะมีค่าประมาณ 4 byte เพื่อให้แน่ใจว่า pointer ที่ชี้ไปยัง Array ของ structure จะทำงานอย่างถูกต้อง อย่างไรก็ตามเราไม่ต้องการที่จะให้ค่าของ fixed-length ของ Opaque data type เพิ่มขึ้น

A Varying-Length Opaque Data Type

เมื่อเรามีการกำหนดคำสั่ง VARIABLE สำหรับ internal length แล้วจะเป็นการสร้าง Varying-length Opaque type โดยปกติแล้ว ความยาวสูงสุดของ Opaque type นี้คือ 2 killobyte และในการที่จะกำหนดขนาดของ varying-length Opaque type ที่ไม่เหมือนกันเราจะใช้คำสั่ง MAXLEN ช่วย เราสามารถกำหนดขนาดได้สูงถึง 32 killobyte เมื่อไหร่ก็ตามที่เรากำหนดขนาดโดยใช้คำสั่ง MAXLEN แล้ว Universal server สามารถที่จะจอง resource ต่างๆ สำหรับ Opaque type นี้ และถ้าขนาดของข้อมูลมีค่าเกินค่าของ MAXLEN จะมีการแจ้ง error ให้ทราบ

ตัวอย่างคำสั่งในการสร้าง Opaque type ที่ชื่อ var_type ที่เป็น varying-length Opaque type ซึ่งมีขนาดสูงสุด 4 kilobyte

```
CREATE OPAQUE TYPE var_type (INTERNALLENGTH=VARIABLE,
        MAXLEN=4096);
```

Memory Alignment

เมื่อ Universal server ส่งผ่านค่าข้อมูลไปที่ routine ที่กำหนดโดย user มันจะจัดข้อมูลของ Opaque type ไปอยู่ในของเขตของหน่วยความจำที่กำหนด ในการทำ Alignment นั้นขึ้นอยู่กับ C definition ของ Opaque type บนระบบ

เราสามารถที่จะกำหนดการทำ memory-alignment สำหรับ Opaque type ด้วยคำสั่ง ALLIGNMENT ของคำสั่งในการสร้าง Opaque type ดังตารางที่สรุปไว้ดังต่อไปนี้

Alligment value	Meaning	Purpose
1	Align structure on single-byte boundary	Structures that begin with 1-byte quantities
2	Align structure on 2-byte boundary	Structures that begin with 2-byte quantities such as mi_unsigned_smallint
4	Align structure on 4-byte boundary	Structures that begin with 4-byte quantities such as float or mi_unsigned_interger
8	Align structure on 8-byte boundary	Structures that contain members of the C double data type

ตารางที่ 5.2 แสดงค่าต่างๆ ในการทำ Memory Alignment

5.4.2 Writing The Support functions

Support function เป็น function ที่ผู้ใช้กำหนดขึ้นมา (user-defined function) ซึ่งถูกเขียนขึ้นมาในรูปแบบของภาษา C โดยบางครั้งอาจจะถูกเรียกว่าเป็น *external function* ครั้งแรกที่เรามีการเขียนฟังก์ชั่นนี้ขึ้นมา เราต้องมีการ compile แล้วนำฟังก์ชั่นที่ compile แล้วไปเก็บไว้ในส่วนที่เรียกว่าเป็น share libraries ใน server computer เราต้องมีการเตรียมเส้นทางของ share libraries เหล่านี้ในประโยคคำสั่ง CREATE FUNCTION เมื่อเราจะนำ support function เหล่านั้นไปเก็บไว้ใน database server และเราต้องแน่ใจได้ว่า database server สามารถเข้าถึง share libraries ได้ในขณะที่เวลา run time

5.4.3 Registering the Opaque Type with the Database

ครั้งแรกที่เรามีการสร้าง internal structure และ support function ของชนิดข้อมูลที่เป็น Opaque type แล้วเราจะใช้ Sql statement ต่อไปนี้ในการนำค่าต่างๆเหล่านั้นไปเก็บไว้ใน database

- CREATE OPAQUE TYPE statement ในการใช้งาน Opaque data type เหมือนเป็น data type ชนิดหนึ่ง
- CREATE FUNCTION statement ในการนำ support function ไปเก็บไว้ใน database
- CREATE CAST statement ในการที่จะใช้งาน support function เหมือนเป็น cast function

5.4.3.1 Registering the Opaque Data Type

CREATE OPAQUE TYPE statement จะเป็นการนำเอา Opaque type ไปเก็บไว้ใน database ซึ่งมีข้อมูลสำหรับ database ดังต่อไปนี้

- ชื่อ และผู้ที่เป็นเจ้าของ Opaque type

ชื่อของ Opaque type จะเป็นชื่อของ data type ที่ Sql statement ใช้ และมันจะไม่มีชื่อข้อมูลของ internal structure สำหรับ Opaque type

- The size of the opaque data type

เราจะกำหนดขนาดของ opaque type ด้วย modifier INTERNALLENGTH ซึ่งมันจะแสดงขนาดของ opaque type ว่าเป็นแบบ fixed-length หรือแบบ varying-length

- The values of the different opaque type modifier

CREATE OPAQUE TYPE statement สามารถที่จะกำหนด modifier สำหรับ opaque type ได้ดังนี้คือ : MAXLEN, PASSEDBYVALUE, CANNOTHASH และ ALIGNMENT และจะเก็บค่าข้อมูลเหล่านั้นใน ระบบตารางของ **sysxtdtypes** เมื่อไหร่ก็ตามที่มันเก็บค่าของ opaque type ไว้ใน sysxtdtypes แล้ว CREATE OPAQUE TYPE statement จะเป็นผลทำให้เกิดค่าที่ไม่ซ้ำกันซึ่งจะเรียกว่า *extended identifier*

5.4.3.2 Registering Support Functions

ใช้รูปแบบของคำสั่ง CREATE FUNCTION statement ในการที่จะนำ support function ไปเก็บไว้ใน database เนื่องจาก support function จะถูกเขียนด้วย external language เช่น ภาษา C โดยที่ CREATE FUNCTION statement มี syntax ของคำสั่งดังต่อไปนี้

CREATE FUNCTION *func_name(parameter_list)*
RETURN *ret_type*
EXTERNAL NAME *'pathname'*
LANGUAGE C NOT VARIANT

Sql statement นี้จะจัดเตรียมข้อมูลต่างๆให้ database ดังต่อไปนี้

- name ที่ชื่อ *func_name* และ owner ของ support function
- optional specific name สำหรับ support function
- data type ของ parameter, *patameter_list* และ return value, *ret_type* ของ support function
- location, *pathname* ของ source code ของ support function
- language ที่ใช้เขียน support function เช่น C language
- routine modifier ที่แสดงว่า support function จะไม่ return ค่าผลลัพธ์ที่ไม่ตรงกับ arguments : NOT VARIANT

5.4.4 Granting Privileges for an Opaque Data type

ครั้งแรกเมื่อมีการสร้างชนิดข้อมูล Opaque type และนำไปเก็บไว้ใน database ใช้รูปแบบของคำสั่ง GRANT ในการที่จะกำหนดสิทธิ์ในการใช้งานของข้อมูลชนิดนี้ดังต่อไปนี้

- Privileges on the use of the opaque data type
- Privileges on the support functions of the opaque data type

5.4.4.1 Privileges on the Opaque Data Type

ในการที่จะสร้าง Opaque type ใหม่ใน database เราต้องมีสิทธิ์ในการในใช้งานทรัพยากรต่างๆ ภายใน database คำสั่ง CREATE OPATUE TYPE จะสร้าง ชนิดข้อมูล Opaque type ใหม่ โดยที่สิทธิ์ในการใช้งานจะเป็นของ owner และ DBA ในการใช้งานชนิดข้อมูล Opaque type เราต้องมีสิทธิ์ในการใช้งาน (privilege) , Owner สามารถที่จะให้สิทธิ์ในการใช้งานแก่ user คนอื่นๆโดยใช้รูปแบบของคำสั่ง GRANT USAGE ON TYPE

database server จะมีการตรวจสอบสิทธิ์ในการใช้งานเมื่อไหร่ก็ตามที่ชื่อของ Opaque type ปรากฏอยู่ในรูปแบบคำสั่งของ Sql แต่ database server จะไม่ตรวจสอบสิทธิ์ในการใช้งานของคำสั่ง Sql ต่อไปนี้

- access column ของชนิดข้อมูล Opaque type สิทธิ์ในการ Select, Insert, Update และ Delete จะเป็นตัวกำหนดในการ access colums
- invoke user define routine ที่ใช้ชนิดข้อมูล Opaque type เป็น argument ในการที่จะ excute routine ต่างๆ สิทธิ์ในการ excute จะเป็นตัวกำหนดระดับในการ access user define routine

ตัวอย่างของคำสั่ง GRANT ในการกำหนดสิทธิ์ในการใช้งานบนชนิดข้อมูล circle Opaque type แก่ user ชื่อ dexter:

```
GRANT USAGE ON TYPE circle TO dexter
```

5.4.4.2 Privileges on the Support Functions

ในการที่จะนำ support function ไปเก็บไว้ใน database เราต้องมีสิทธิ์ในการในใช้งานทรัพยากรต่างๆ ภายใน database คำสั่ง CREATE FUNCTION จะสร้าง support function ใหม่ โดยที่สิทธิ์ในการใช้งานจะเป็นของ owner และ DBA ในการใช้งาน support function นั้นเราต้องมีสิทธิ์ในการใช้งาน (privilege)

5.4.5 Create SQL-Invoked Function

SQL-invoked function คือ ฟังก์ชั่นที่กำหนดขึ้นโดยผู้ใช้ (end-user) โดยที่ผู้ใช้สามารถที่จะรียกใช้ในรูปแบบของ SQL statement ได้ เราต้องเขียน SQL-invoked function เพิ่มเติมเพื่อใช้ทำงานกับ opaque type ได้ในรูปแบบดังต่อไปนี้

- เขียน arithmetic หรือ built-in function ขึ้นมาใหม่ เพื่อที่จะสนับสนุนการทำงานเกี่ยวกับ arithmetic operation และ built-in function บน opaque type
- เขียน relational-operator function ขึ้นมาใหม่ เพื่อที่จะสนับสนุน operation ในการเปรียบเทียบบน opaque type
- เขียน end-user routine ขึ้นมาใหม่ เพื่อสนับสนุนการทำงานที่เพิ่มเติมนอกเหนือจากนี้บน opaque type
- เขียน casting function ขึ้นมาใหม่ เพื่อสนับสนุนการ convert ข้อมูลที่นอกเหนือจากนี้ทั้งไปและกลับ บน opaque type

- Operating on Data

Universal server จะสนับสนุนการทำงานของ Sql-invoked function ซึ่งอนุญาติให้เราดำเนินการกับข้อมูลในรูปแบบของคำสั่ง Sql ดังต่อไปนี้คือ Arithmetic and text operator functions

- Built-in functions
- Aggregate functions

- Operator function for Opaque types

Universal server จะสนับสนุนการทำงานชนิดของ operator ในรูปแบบของคำสั่ง Sql ดังต่อไปนี้คือ

- Arithmetic operator ซึ่งปกติแล้วจะดำเนินการกับค่าข้อมูลที่เป็นตัวเลข
- Text operator จะดำเนินการกับข้อมูลที่เป็นตัวอักษร

Universal server จะเตรียม operator function ต่างๆ เกี่ยวกับ arithmetic operator และ text operator โดยที่ operator function เหล่านั้นจะทำงานกับ built-in data type เราต้องมีการเขียน operator function เหล่านั้นขึ้นมาใหม่ในการที่จะทำงานกับ new Opaque data type โดยที่ในกรณีที่เราเขียน operator function นั้นขึ้นมาใหม่ เราต้องแน่ใจได้ว่าเราทำตามกฎเกณฑ์ต่างๆ ดังต่อไปนี้คือ

1. ชื่อของ operator function ต้องตรงกับชื่อตามที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตามชื่อของ operator function ไม่ค่อยมีความรู้สึกต่างกันมากนักเช่น ชื่อของ function plus() เหมือนกันกับชื่อของ function Plus()
2. Operator function จะต้องจัดการกับจำนวนของ parameter ได้อย่างถูกต้อง
3. Operator function จะต้อง return ค่ากลับอย่างถูกต้องและเหมาะสม

- Built-in Function for Opaque Data Type

Universal server ได้มีการจัดเตรียม Sql-invoke function พิเศษขึ้นซึ่งเราเรียกว่า built-in function ในการดำเนินการทางด้านคณิตศาสตร์เบื้องต้น โดยที่ built-in function ที่ Universal server มีให้นั้นจะทำงานกับชนิดข้อมูลที่เป็น built-in data type และในการที่จะทำงานกับชนิดข้อมูล Opaque data type เราต้องมีการเขียน built-in function ใหม่ขึ้นมา และในกรณีที่เราเขียน built-in function ใหม่ขึ้นมา เราต้องแน่ใจได้ว่าเราทำตามกฎเกณฑ์ต่างๆ ดังต่อไปนี้คือ ชื่อของ built-in function ต้องตรงกับชื่อตามที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตามชื่อของ built-in function ไม่ค่อยมีความรู้สึกแตกต่างกันมากนักเช่น ชื่อของ function abs() จะเหมือนกันกับชื่อของ function Abs()

1. Built-in function จะต้องมีการเขียนทับกันได้
2. Built-in function จะต้องจัดการกับจำนวนของ parameter ได้อย่างถูกต้อง
3. Built-in function จะต้อง return ค่ากลับอย่างถูกต้องและเหมาะสม

- Aggregrate Function for Opaque Data Type

Universal server จะสนับสนุนการทำงานของ aggregrate function บนชนิดข้อมูล Opaque data type ดังต่อไปนี้

- COUNT
- MIN
- MAX

- Comparing Data

Universal server จะสนับสนุนการทำงานของ function ต่างๆที่อนุญาติให้เราสามารถที่จะเปรียบเทียบข้อมูลในรูปแบบของคำสั่ง Sql ดังนี้

- SQL operators is a conditional clause
- Relational operator function

- Condition for Opaque Data Types

Universal server จะสนับสนุนการทำงานในรูปแบบที่เป็นเงื่อนไขกับชนิดข้อมูล Opaque type ในรูปแบบของคำสั่ง Sql ดังนี้

- The IS and IS NOT operators
- The IN operator ถ้ามีการพบ function equal()
- The BETWEEN operator ถ้ามีการพบ function compare()

- Relational Operators for Opaque Data Type

Universal server จะจัดเตรียม operator function ที่สนับสนุนการทำงานของ relational operator โดยที่ relational operator ที่ universal server จัดเตรียมไว้ให้จะทำงานกับ built-in data type เราสามารถที่จะสร้าง relational operator ขึ้นมาใหม่ในการทำงานชนิดข้อมูล Opaque type ได้และ ถ้าเรามีการสร้างรุ่นของ relational operator ขึ้นมาใหม่ เราต้องแน่ใจได้ว่าเราได้ทำตามกฎเกณฑ์ดังต่อไปนี้

1. ชื่อของ relational operator function ต้องตรงกับชื่อตามที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตามชื่อของ operator function ไม่ค่อยมีความรู้สึกต่างกันมากนักเช่น ชื่อของ function equal() เหมือนกันกับชื่อของ function Equal()
2. Relational operator function จะต้องมี parameter 2 ตัวซึ่งเป็นชนิดข้อมูล Opaque type

3. Relational operator function จะต้องเป็น boolean function และจะ return ค่าเป็น boolean

เราต้องมีการกำหนด function equal() ในการจัดการกับชนิดข้อมูล Opaque type ถ้าเราต้องการอนุญาติให้ column ของชนิดข้อมูลนี้ทำงานต่างๆดังนี้

- constranied as UNIQUE or PRIMARY KEY
- compared ith the equal(=) operator in an expression
- used with IN operator in a condition

- Opaque Type Sorting

function compare() คือ invoke-function ซึ่งสามารถที่จะเรียงลำดับของข้อมูลที่ต้องการได้ Universal server จใช้ function compare() ในการ excute รูปแบบของคำสั่ง SELECT ดังต่อไปนี้

- The ORDER BY clause
- The UNIQUE and DISTINCT keywords
- The UNION keyword

database server จะใช้ compare function ในการใช้งาน BETWEEN operator ในคำสั่งที่เป็นเงื่อนไขของ Sql

Universal server จะจัดเตรียม compare function ที่ทำงานกับ built-in data type เราสามารถที่จะสร้าง compare function ขึ้นมาใหม่ในการทำงานชนิดข้อมูล Opaque type ได้และ ถ้าเรามีการสร้างรุ่นของ compare function ขึ้นมาใหม่ เราต้องแน่ใจได้ว่าเราได้ทำตามกฎเกณฑ์ดังต่อไปนี้

1. ชื่อของ compare() function ต้องตรงกับชื่อตามที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตามชื่อของ compare() function ไม่ค่อยมีความรู้สึกต่างกันมากนักเช่น ชื่อของ function compare() เหมือนกันกับชื่อของ function Compare()
2. compare function จะต้อง return ค่า interger ที่แสดงผลลัพธ์ของการเปรียบเทียบดังต่อไปนี้

- <.0 จะแสดงว่า argument แรกน้อยกว่า (<) argument ที่สอง
- 0 จะแสดงว่า argument แรกน้อยเท่ากันกับ (=) argument ที่สอง
- > 0 จะแสดงว่า argument แรกมากกว่า (>) argument ที่สอง

- Create End-User Routine

Universal server อนุญาติให้เรากำหนด Sql-invoke function และ procedure ต่างๆ ที่ end - user สามารถใช้งานได้ในรูปแบบของคำสั่ง Sql ได้ end-user routine เป็น function ที่ทำเพิ่มเติมในกรณีที่ end-user ต้องการทำงานกับชนิดข้อมูล Opaque type ขั้นตอนในการสร้าง end-user routine มีดังต่อไปนี้

- เขียน end-user routine ในรูปแบบของภาษา C หรือ SPL
- นำ end-user routine ไปเก็บไว้ใน database ด้วยคำสั่ง CREATE FUNCTION หรือนำ end-user procedure ไปเก็บไว้ใน database ด้วยคำสั่ง CREATE PROCEDURE
- กำหนดสิทธิ์ (Grant) ในการใช้งาน end-user routine ต่างๆ ให้แก่ user

เราต้องมีการเขียน end-user function ในการทำงานเหมือนกันกับ casting function สำหรับชนิดข้อมูล Opaque type ในการสร้าง casting function เราจะใช้คำสั่ง CREATE FUNCTION ในการนำ end-user function ไปเก็บไว้ใน database และคำสั่ง CREAE CAST ในการนำ casting function ไปเก็บไว้ใน database

5.4.6 Customizing Use of Access Methods

Universal Server จะสนับสนุนการสร้าง B-tree secondary access method และจะมีการกำหนดรูปแบบในการใช้งานด้วย โดยปกติแล้วคำสั่ง CREATE INDEX (โดยไม่มี USING clause) จะสร้าง generic B-tree index บน column หรือบน function ที่ผู้ใช้กำหนดกำหนดขึ้นมาเอง

เมื่อเรามีการสร้าง Opaque data type เราต้องแน่ใจว่า secondary access method มีอยู่บนชนิดข้อมูล Opaque type ใหม่ พิจารณาปัจจัยต่างๆต่อไปนี้เกี่ยวกับ secondary access method สำหรับชนิดข้อมูล Opaque type

- Generic B-tree สนับสนุนการใช้งานชนิดข้อมูล Opaque type ไหม ?
- ถ้าชนิดข้อมูล Opaque type คือ spatial เราสามารถที่จะใช้ R-tree index ได้ไหม ?
- มี secondary access method อื่นไหม ที่ทำ index ได้ดีกว่าบนชนิดข้อมูล Opaque type

ในการสร้าง index สำหรับ secondary access method บน column ของชนิดข้อมูล Opaque type แล้ว database server จะมีการค้นหา operator class ที่เกี่ยวข้องกับ secondary access method, Operator class นี้จะเป็นตัวกำหนดการทำงาน (strategy functions) บนชนิดข้อมูล Opaque type ซึ่ง secondary access method สามารถใช้งาน support function ได้

5.4.6.1 Using the Genreic B-Tree

Generic B-tree secondary access method จะมี operator class ปกติคือ btree_ops ซึ่งเป็น operator-class function ที่จัดการเกี่ยวกับ index สำหรับชนิดข้อมูล built-in data type และมีหน้าที่ต่างๆ ดังนี้

- เรียงลำดับข้อมูลตามลำดับของ lexiographical ถ้าชนิดข้อมูล Opaque type ไม่เป็นลำดับที่เป็น logical แล้วเราสามารถที่จะกำหนด operator-class functions สำหรับชนิดข้อมูล Opaque type ในการจัดการเกี่ยวกับลำดับตามที่เราต้องการได้

- เปรียบเทียบค่าข้อมูล 2 ค่าของ one-dimensional value ในกรณีที่ชนิดข้อมูล Opaque type มีค่าข้อมูลมากกว่า 1 ค่า เราสามารถที่จะกำหนด operator-class function ในการเปรียบเทียบค่าข้อมูล 2 ค่าของข้อมูลชนิดนี้ได้ ถ้าเราไม่สามารถกำหนด one-dimensional value ของชนิดข้อมูล Opaque type ได้แล้วเราก็จะไม่สามารถใช้งาน B-tree index เป็น secondary access method ได้

5.4.6.2 Indexing Spatial Data

วิธีการที่ Generic B-tree secondary method ใช้ในการจัดเรียงลำดับข้อมูลจะมีประโยชน์สำหรับ one-dimensional data, btree_ps จะเป็น operator-class function ปกติของ B-tree operator class ซึ่งจะเรียงลำดับข้อมูล one-dimension ภายใน B-tree index ในกรณีที่ข้อมูลของ Opaque type ของเราเป็น multidemensional แล้วเราต้องมีการใช้ secondary access method ในการจัดเรียงข้อมูลสำหรับของ multidimensional data ด้วย

R-tree secondary method มีประโยชน์มากสำหรับข้อมูลที่เป็น spatial หรือ multidimensional เช่น map และ diagram ถ้า database ของเรามีการสร้างรูปแบบการทำงานเป็นแบบ R-tree เราสามารถที่จะกำหนด R-tree index บน column ของชนิดข้อมูล spatial ได้

5.4.7 Other Operation On Opaque Data type

ในส่วนนี้จะอธิบายถึงการดำเนินงานต่างๆ ที่เราสามารถทำกับชนิดข้อมูล Opaque data type ได้

- เราสามารถจะเข้าถึงข้อมูล Opaque type จาก application program ได้อย่างไร
- เราสามารถที่จะยกเลิกการใช้งานชนิดข้อมูล Opaque type จาก database ได้อย่างไร

5.4.7.1 Accessing an Opaque Data Type

ครั้งแรกที่เรามีการสร้างชนิดข้อมูล Opaque type เราต้องนำข้อมูลชนิดนี้ไปเก็บไว้ใน database ต่อไปนี้เป็น program ที่สามารถติดต่อเข้าไปที่ database เพื่อใช้งานกับชนิดข้อมูล Opaque type

- An INOFORMIX-ESQL/C application จะใช้รูปแบบคำสั่งของ Sql และตัวแปรต่างๆ คือ lvarchar,fixed binary หรือ var binary
- A external routine (user-routine ที่ถูกเขียนด้วยภาษา C) ที่ใช้ Datablade API
- An SPL routine (user-define routine ที่ถูกเขียนด้วยภาษา store procedure)

5.4.7.2 Dropping an Opaque Data Type

เราไม่สามารถที่จะยกเลิกการทำงานของชนิดข้อมูล Opaque type ได้ถ้าข้อมูลยังเกี่ยวข้องอยู่กับ database ดังนั้นในการที่จะยกเลิกการทำงานของชนิดข้อมูล Opaque type เราต้องทำในลักษณะการ reverse กับการที่เรานำข้อมูลชนิด Opaque type ไปเก็บไว้ใน database ดังต่อไปนี้

1. Remove หรือเปลี่ยนแปลงชนิดข้อมูล Opaque type ที่อยู่บน column ใน database ให้เป็นชนิดข้อมูลแบบอื่น
2. ใช้คำสั่ง INVOKE ในการยกเลิกสิทธิ์ในการใช้งานบนชนิดข้อมูล Opaque type ซึ่งก่อนนั้นจะให้สิทธิ์ด้วยคำสั่ง USAGE ON TYPE
3. ใช้คำสั่ง INVOKE ในการยกเลิกสิทธิ์ในการ excute function หรือ routine ของข้อมูล Opaque type ซึ่งก่อนนั้นจะให้สิทธิ์ในการ excute function หรือ routine ด้วยคำสั่ง EXCUTE ON FUNCTION หรือ EXCUTE ON ROUTINE
4. ใช้คำสั่ง DROP CAST ในการยกเลิกการใช้งานของ casting function บนชนิดข้อมูล Opaque type
5. ใช้คำสั่ง DROP FUNCTION หรือ DROP ROUTINE ในการยกเลิกการใช้งาน support function ของชนิดข้อมูล Opaque type ใน database ที่เรากำลังใช้งานอยู่
6. ใช้คำสั่ง DROP TYPE ในการยกเลิกการใช้งานชนิดข้อมูล Opaque type ใน database ที่เรากำลังใช้งานอยู่

บทที่ 6

การใช้งาน Datablade Module

Datablade module เป็นที่เก็บรวบรวม Object ต่างๆรวมทั้ง Code ที่ Informix Universal Server ไม่มีมาให้โดยจะมีการเพิ่มเติม function ใหม่ๆเข้าไป Universal Server สามารถที่จัดการกับ Datablade modeul ในการใช้งานในระดับต่างๆ ของ Data type ใหม่เหมือนกันกับการใช้งาน Built-in function

โดยทั่วไปแล้ว Datablade module จะสนับสนุนการทำงานกับโปรแกรมที่ใช้งานโดยเฉพาะ เช่น โปรแกรมจัดการเกี่ยวกับข้อมูลที่เป็นแผนที่ (Spatial data management) , การทำ Index เกี่ยวกับเอกสาร (Document indexing) , และการเรียกค้นข้อมูล (Retrieval) Datablade module หลายๆ ตัวสามารถถูกใช้งานกับโปรแกรมๆ เดียวได้ เช่น โปรแกรมบันทึกข้อมูลทางแพทย์ (Medical records application) ที่จะเก็บข้อมูลมาจากโรงพยาบาลหลายๆแห่งที่แตกต่างกัน จะใช้ Informix spatial databalde module ในการเก็บข้อมูลและใช้ในการสอบถามที่ตั้งของโรงพยาบาลต่างๆ เป็นต้น

6.1 Datablade Module Component

Datablade module จะประกอบไปด้วยสิ่งต่างๆ ดังต่อไปนี้

- **Extended data type** เป็น data type ชนิดใหม่ที่เพิ่มเติมมาจาก Database server ซึ่งได้แก่ Collection data type, Row data type, Distinct data type และ Opaque data type
- **Routines** ฟังก์ชั่นต่างๆ ของ C หรือ Pascal ที่จะถูกเรียกใช้งานจาก Database server ในการใช้งานกับชนิดข้อมูล Opaque type หรือ Client application
- **Casts** เป็น routine ที่ใช้เปลี่ยนแปลงค่าข้อมูลระหว่าง data type
- **Interfaces** เป็น Unique IDs ที่ให้กับ Databalde module หรือ Subsets ของชนิดข้อมูลและ routine ที่อยู่ใน Datablade module โดยที่ Datablade module จะขึ้นอยู่กับ Databalde module อื่นๆที่ interface ด้วยและ Blademanager จะต้องแน่ใจว่าทุกๆ Datablade module ที่ interface กันนั้นจะถูกนำเข้าไปใน Database server ทั้งหมด
- **Qualified data typed** เป็นชนิดข้อมูลที่เป็น built-in data type ของ Universal server เช่น CHARACTER และ DECIMAL ซึ่งสามารถที่จะเพิ่มเติมข้อกำหนดต่างๆ เข้าไปได้
- **Tables and indexs** จะเป็น Database table และ index ในการใช้งานกับชนิดข้อมูลใหม่ หรือ ใช้งานกับ Client application
- **Error codes and error message** จะเป็นข้อความที่บอกข้อผิดพลาด (Error) ที่ถูกใช้งานใน Datablade module
- **Client code** เป็น code ที่อนุญาติให้เครื่อง Client สามารถที่จะเข้าถึงส่วนประกอบอื่นๆ ของ Datablade module ได้

6.2 Developing Databalde modules

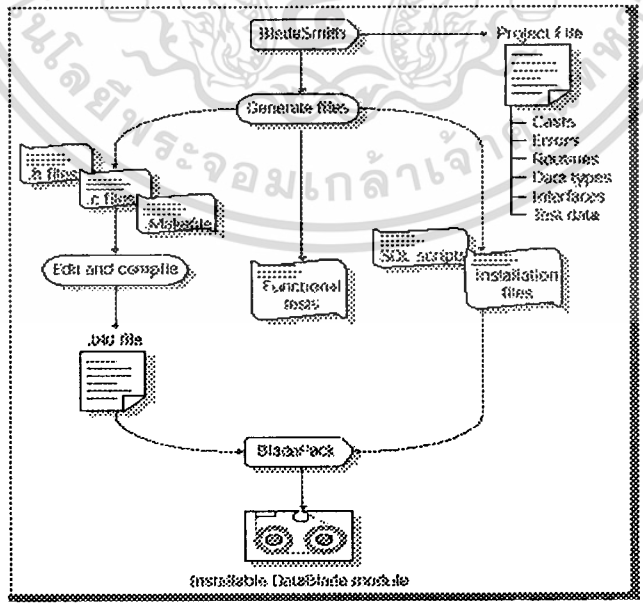
ในการที่จะสร้าง Datablade module จะเกี่ยวข้องกับการกำหนดชนิดข้อมูลต่างๆ และ Routine ต่างๆ ที่จะใช้งานกับชนิดข้อมูลนั้น , การเขียน Supporting code สำหรับชนิดข้อมูลและ Client application , การทดสอบ (Testing), การตรวจหาข้อผิดพลาด (Debugging), การทำเอกสาร (Documenting), และ การรวม Module ต่างๆเข้าไว้ด้วยกัน (Packing)

Datablade Developers Kits จะรวมเครื่องมือหรือ Tools ต่างๆที่ช่วยในการสร้าง Datablade module โดยที่เครื่องมือต่างๆนั้นจะสร้างงานหลายๆอย่างให้เองโดยอัตโนมัติ จะมีการจัดเตรียม Product ต่างๆสำหรับผู้ใช้ที่มีความถูกต้องและมี User interface ที่ใช้งานง่าย โดยที่ Datablade module เหล่านั้นสามารถรับประกันได้ว่าจะใช้งานได้ใน Informix Universal server

Bladesmith จะเป็นเครื่องมือในการสร้าง Project ที่จะจัดการกับรูปแบบของ Datablade module เราสามารถที่จะใช้ Bladesimth ในการสร้าง Project หลังจากนั้นแล้วจะมีการกำหนด Object ต่างๆ ที่เป็นของ Databalde module เช่น Data type, Routine และ Blasesmith จะสร้าง Source code ในรูปแบบของภาษา C, สร้าง header files, makefiles, functional test files, SQL scripts, message และ packing files ต่างๆเหล่านี้ให้

Bladepack จะเป็นเครื่องมือที่สร้าง Datablade module ในขั้นสุดท้าย โดยที่จะอนุญาติให้เรากำหนดข้อมูลของ Databalde module รวมทั้งกำหนด directory ที่ใช้ในการเก็บ Datablade module

Blademanager เป็น Utility ที่มีมาพร้อมกับ Universal server และ Datablade Developers Kits, มันเป็นเครื่องมือสำหรับ Database administrations ในการที่จะนำ Datablade module (Register)ไปเก็บไว้ใน Database หรือ นำออกมาจาก Database (Unregister)



รูปที่ 6.1 Datablade Module Development process

6.3 Using BladeSmith

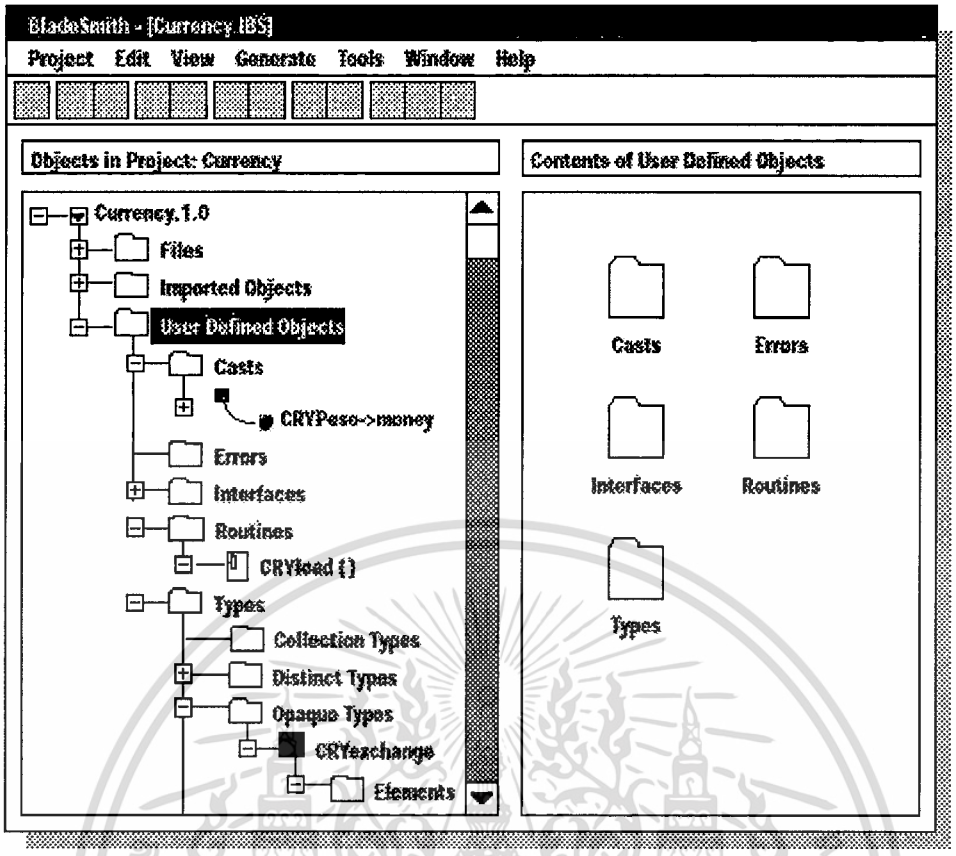
Blademith จะเป็นตัวช่วย Datablade module ในการพัฒนา Project และจะจัดเตรียมเสมือนเป็นการแทน Object ต่างๆ ภายใน Datablade module และอนุญาติให้เราเพิ่มเติม Object และแก้ไขคุณสมบัติต่างๆของ Object ใน Datablade moule ได้ หลังจากได้มีการกำหนด Object ต่างที่เป็น Project ด้วย Bladesmith แล้ว Bladesmith จะมีการสร้าง Source code files, SQL script, functional test, และการติดตั้ง Packaging fles ให้

ก่อนที่จะมีการเริ่มต้นใช้งาน Bladesmith เราควรมีการออกแบบ Datablade module ต่างๆ เสร็จสิ้นเรียบร้อยแล้วก่อน โดยที่เราจะมีการเขียนข้อกำหนดต่างๆที่เป็นภาพรวมว่า Datablade moule ของเราทำอะไรได้บ้าง และจะมีการออกแบบข้อกำหนดต่างๆที่เป็นรายละเอียดที่จะใช้ในการสร้าง Datablade module นี้ขึ้นมา

Bladesmith จะอนุญาติให้เรากำหนด Object ต่างๆ สำหรับ Databalde module ได้ดังนี้คือ

- Casts
- Interfaces
- Errors
- Routines
- Collection data type
- Distinct data type
- Qualified data type
- Opaque data type
- SQL files
- Client files

Bladesmith จะใช้ wizard ในการที่จะเลือกข้อมูลต่างๆที่ใช้ในการกำหนด Object โดยที่ wizard จะมีประโยชน์มากในกาอธิบายถึงข้อมูลต่างๆที่เราป้อนเข้าไป หลังจากที่เรามีการกำหนด Object ต่างๆ ใน Datablade module แล้วเราจะใช้ Bladesmith ในการสร้าง Source code, SQL script, test files และ installation files



รูปที่ 6.2 แสดง Bladesmith windows

Bladesmith project window จะแบ่งเป็น 2 ส่วนคือ ช่องด้านซ้าย ซึ่งจะเรียกว่า *Project view* โดยจะแสดง Object ต่างๆในรูปแบบเป็นลำดับชั้นของ tree ภายใน Project นั้น โดยที่อาจจะเป็น Folder ของ files, imported object และ user-define object ส่วนช่องด้านขวาจะเรียกว่า *Item view* ที่จะแสดงข้อมูลต่างๆของ Object ที่ถูกเลือกใน Project view

ขั้นตอนในการใช้งาน BladeSmith

ในส่วนนี้จะอธิบายถึงขั้นตอนในการใช้ Bladesmith ในการสร้าง Datablade module ที่เราออกแบบไว้แล้ว โดยที่จะมีขั้นตอนที่ใช้กันโดยทั่วไปดังต่อไปนี้

- *Setup a project environment*

เป็นการกำหนด directory ต่างๆที่ใช้ในการเก็บแต่ละ Datablade module โดยที่ตามปกติแล้ว Bladesmith จะเก็บชื่อของ Datablade module object ใน Project files ที่มีส่วนขยายเป็น **.ibs** และ Bladesmith จะมีการสร้าง Source code, SQL script, functional test และ installation package files ใน directory ที่มีความสัมพันธ์กับ Project file นั้น โดยปกติแล้ว Bladesmith จะสร้าง sub directory ชื่อ **src, script, function** และ **install** ภายใต้ directory ที่ใช้เก็บ Project files

- *Import Datablade module objects*

เป็นการนำเอา interface และ data type มาจาก Datablade module project files อื่นๆได้ โดยที่ interface และ data type ที่เราอ้างถึงใน Datablade module ของเราจะต้องปรากฏอยู่ใน Import Object folder ในลำดับชั้นของ Project และ Bladesmith จะแสดงรายชื่อของ data type ต่างๆ ที่อยู่ใน Universal server ที่เราสามารถที่จะนำเอามาใช้ได้

- *Defining New Datablade module objects*

เป็นการใช้งาน Bladesmith ในการที่จะเพิ่ม Object ต่างๆเข้าไปใน Datablade module รวมทั้งสามารภที่จะเปลี่ยนแปลงคุณสมบัติต่างๆของ Object ได้

- *Adding Test Data*

เป็นการเพิ่มเติมข้อมูลต่างๆที่จะใช้ Test สำหรับชนิดข้อมูลที่เป็น Opaque type, user-define routine และ Cast type

- *Generating Files*

เมื่อมีการสร้าง files แล้ว Bladesmith จะสร้าง files ที่อธิบายถึงรายละเอียดต่างๆของ Object ที่เรากำหนดใน Project ตัวอย่าง ของ files ต่างๆ เช่น SQL script ที่จะใช้โดย Blade manager ในการที่จะนำ Datablade module ไปเก็บไว้ใน database , C source files ซึ่งประกอบไปด้วย code ของ function ต่างๆ ที่กำหนดใน Project เป็นต้น

- *Regenerating Files*

เราสามารถที่จะเปลี่ยนแปลง files ต่างๆ ที่ถูกสร้างโดย Bladesmith ได้ ตัวอย่าง เช่น ในกรณีที่เรามีการเพิ่ม Cast เข้าไปใหม่ใน Datablade module เราจะต้องมีการ Regenerate SQL script ใหม่ รวมทั้งถ้า Cast นั้นประกอบไปด้วย Support routine เราจะต้องมีการ Regenerate source file นั้นใหม่ด้วย

6.4 BladePack

BladePack คือโปรแกรมที่จะทำการสร้างตัว Installation Package เพื่อนำไป Registration ลงใน Informix Universal โดยสามารถใช้ได้ทั้งบน UNIX Platform และ Microsoft Windows NT BladePack สามารถที่จะสร้างเป็น directory tree (คือโครงสร้างของ tree ที่เป็น directory และบรรจุ files ที่จะทำการ install อยู่) หรือว่าจะทำการสร้างเป็น installation ใน interactive mode (เป็นลักษณะ wizard ในการ install Datablade)

ใน UNIX Platform ตัว installation interactive mode จะใช้ shell scripts ที่มีชื่อว่า install กับ uninstall ส่วนใน Microsoft Windows จะใช้ Files setup program กับ uninstall program โดยจะต้องสร้างร่วมกับโปรแกรม InstallShield ในที่นี้เราจะทำการสร้างในส่วนเฉพาะ directory tree ก็พอซึ่งจะนำไปใช้ใน Microsoft Windows ซึ่งไม่จำเป็นที่จะต้องมีโปรแกรม InstallShield

files ต่าง ๆ ใน installation package จะถูกแบ่งเป็น components (directory),sub-components (subdirectory),sub-subcomponents (sub-subdirectory) โดยที่คุณจะต้องกำหนดอย่างน้อยหนึ่ง component เพื่อใช้ในการสร้าง installation package

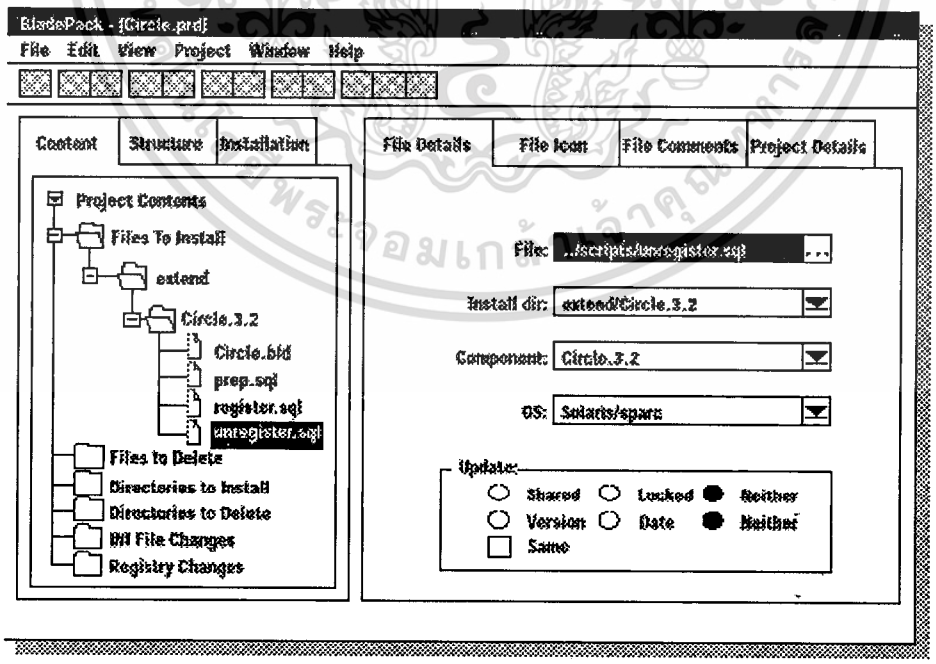
ในการใช้ BladePack ในการจัดโครงสร้างของ files ของ project ที่เราสร้างขึ้น โดยในแต่ละ project จะถูกควบคุมโดย product file (*project_name*.prd) , componet ต่าง ๆ ที่จะถูกบรรจุลงใน package (*project_name*.cmp) ,bill of materials file (*project_name*.bom),string file (*project_name*.str)

ถ้าสร้าง DataBlade module ด้วย BladeSmith ใน menu **Generate Installation Package** จะทำการสร้าง file ต่าง ๆ เหล่านี้ไว้หมดแล้ว

BladePack Windows

BladePack Windows จะถูกแบ่งเป็นสองหน้าต่าง ส่วนทางด้านซ้ายมือจะเรียกว่า project view ซึ่งจะแสดงโครงสร้างต่าง ๆ ของ installation package ทางด้านขวามือจะเรียกว่า item view จะแสดงรายละเอียดต่าง ๆ ของ object ที่ถูกเลือกใน project view

เราใช้ project view ในการเพิ่ม object ลงใน installation package และการจัดโครงสร้างของ installation package เราใช้ item view ในการใส่รายละเอียดของ object ใน installation package ดังรูป



รูปที่ 6.3 แสดง BladePack windows

การใช้งาน BladePack

ขั้นแรกจะทำการ Open File ที่ได้สร้างมาจาก BladeSmith นามสกุล file คือ .prd และทำการสร้าง package ขึ้นมา โดยเลือก menu **Project -> Build Installation** จากนั้นก็จะมี wizard ของการ Build installation ก็จะมีขั้นตอนต่าง ๆ ดังนี้

- Installation type : interactive or file tree ให้ทำการเลือกเป็น file tree
- Platform ที่จะทำการ install ลง ให้เลือกเป็น Windows NT
- Target directory ที่จะสร้างตัว install
- ชื่อ file .prd ต่าง ๆ ที่จะทำการรวมกันใน installation package

BladePack จะทำการสร้าง directory ตามที่ระบุไว้แล้วทำการ copy files ต่าง ๆ ลงใน tree ถ้าสร้างเป็น interactive mode ใน Windows Environment จะต้องมีการเรียกใช้ InstallShield และจะทำการสร้าง directory CD-ROM และ สร้าง disk image ขึ้นมา แต่ใน UNIX จะทำการสร้าง **install** และ **uninstall** shell scripts

ในที่นี้เราจะทำการสร้างเป็น file tree ซึ่งก็จะทำการสร้าง directory ตามที่ระบุไว้ในโครงสร้างของ installation package และจะ copy file ต่าง ๆ ตามที่ระบุไว้ด้วย โดยที่ไม่ต้องใช้โปรแกรม InstallShield ช่วยเลย

6.5 การ Install และการ Register DataBlade Module

การ register ลง Informix Universal สามารถแบ่งเป็น 2 ขั้นตอนหลัก ๆ ดังนี้

1. ทำการ install DataBlade Module ลงใน Informix Universal server
2. ใช้ BladeManager ในการ Register ลงฐานข้อมูล

6.5.1 การ Install DataBlade Module บน Windows NT

ขั้นตอนดังต่อไปนี้คือการ install ลง Informix Universal Server ที่ทำงานอยู่บน Windows NT Platform

1. เข้า login ในกลุ่มของ **Informix-Admin**
2. ทำการ copy files ต่าง ๆ ลงใน Subdirectory ของตัว DBMS เอง โดยจะเป็น Subdirectory ที่ชื่อว่า **\extend\datablade.version** ในส่วนของ datablade.version คือชื่อ directory ที่เป็นชื่อของ DataBlade module ซึ่งจริง ๆ แล้วในส่วนนี้ BladePack จะทำการสร้างให้เรียบร้อยแล้ว เราเพียงแต่ copy directory นั้น ลงใน Subdirectory **%INFORMIXDIR%\extend** ก็พอ ตัวอย่างเช่น DataBlade module ที่มีชื่อว่า Sapatial DataBlade module version 2.2 จะถูก copy ลงใน Subdirectory **%INFORMIXDIR%\extend\spatial.2.20.UC1** เป็นต้น

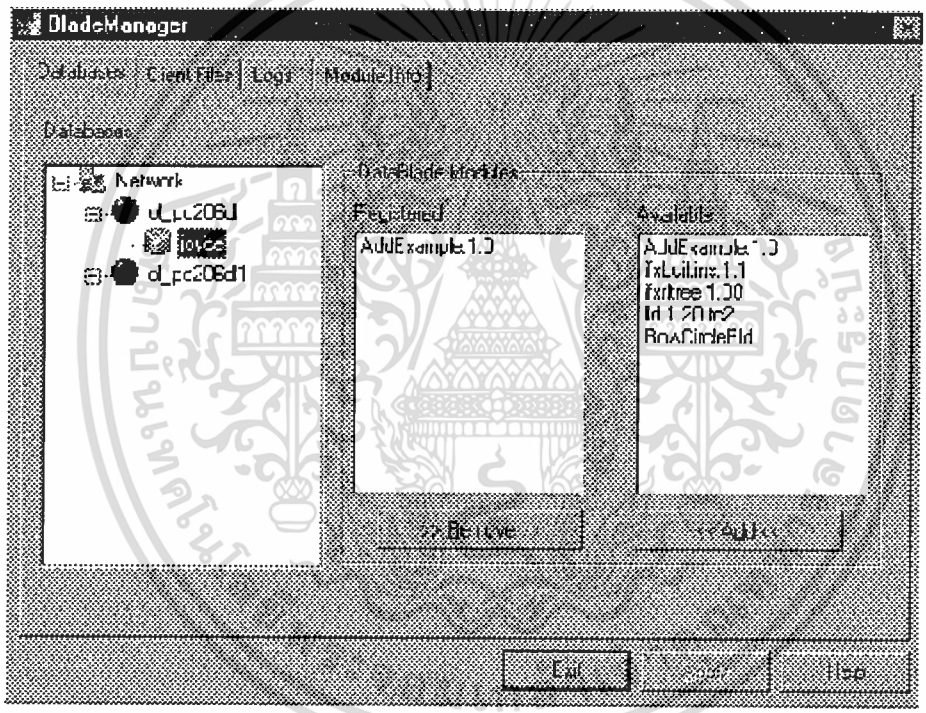
3. ทำการ set attribute ทุก ๆ files ที่ทำการ copy ลงใน directory นั้นให้เป็น read-only ทั้งหมด

หมายเหตุ - %INFORMIXDIR% ก็คือ root directory ของตัว DBMS นั้น ๆ

- ถ้าเป็น interactive mode สามารถสั่ง setup ได้เลย แล้วปฏิบัติตามคำสั่งต่าง ๆ ที่ปรากฏใน wizard install

6.5.2 การใช้งาน BladeManager Register DataBlade module

ก่อนอื่นเราจะต้องมีโปรแกรม BladeManager เสียก่อน ซึ่ง BladeManager เป็นส่วนหนึ่งของ DataBlade Developer Kits ดังเช่น BladeSmith กับ BladePack จากนั้นเราจะทำการเรียกโปรแกรมนั้นขึ้นมาใช้งานซึ่งจะมีลักษณะหน้าตา Windows ดังนี้



รูปที่ 6.4 แสดง BladeManager windows

ในการ Manage ตัว DataBlade module นั้นจะแบ่งเป็น 2 ขั้นตอนต่าง ๆ ดังต่อไปนี้

- Connect to Database
- Register or Unregistered DataBlade module

6.5.2.1 Connect to Database

หลังจากที่มีการ install DataBlade module ลงใน database server แล้ว เราจะทำการ register Datablade module ลงในแต่ละ database แล้วแต่ว่าจะต้องการใน module ไหน ซึ่งจะต้องมีการ

connect ไปที่ database เสียก่อน ในการ connect เราจะต้องมี resource permissions ของการ connect ไปที่ database ด้วย ซึ่งจะต้องให้ Admin เป็นผู้เพิ่มให้ก็ได้

การ connect

1. ให้ดูที่ Database list box ใน Database tab ทำการ click expander button ใน network icon และ database server icon
2. click ชื่อฐานข้อมูลที่จะทำการ connect ด้วย ในบางครั้งอาจจะต้องมีการป้อน user name กับ password ด้วย

หลังจากที่ทำการ connect แล้วจะปรากฏชื่อ DataBlade module ต่าง ๆ ที่ได้ทำการ register ไว้แล้ว กับในส่วนที่จะต้องนำไป register อีก

เมื่อ BladeManager ได้ connect กับฐานข้อมูลแล้ว จะทำการตรวจสอบ DataBlade module ที่ได้ทำการ install ไว้แล้วและจะทำการสร้าง log file ขึ้นมา ในระหว่างที่ทำการตรวจสอบ DataBlade module ที่ตรวจพบจะปรากฏในส่วนของ DataBlade module box ซึ่งถ้าการตรวจนั้น fail จะไม่ปรากฏชื่อใด ๆ เลยใน **Available** list box เราจะต้องทำการตรวจสอบใน log file อีกที(ให้ดูในส่วนของ View Log file)

6.5.2.2 Register DataBlade module

เมื่อ BladeManager จะทำการ Register DataBlade module มันจะทำการ execute กลุ่มคำสั่ง SQL เพื่อเป็นการ register แต่ละ database object โดยจะเป็นคำสั่ง SQL CREATE เราจะต้องมี resource permission ในการ register DataBlade module ลงใน database ด้วย

การ register DataBlade module

1. ใน **Database** tab ให้เราทำการเลือก database ที่จะทำการ register
2. ใน **Available** list box เลือก DataBlade module ที่จะทำการ register
3. click **Add** เพื่อเป็นการ move DataBlade module จาก **Available** list box ไป **Registered** list box
4. click **Apply** เพื่อเป็นการ register DataBlade module ลงใน database

ถ้าเกิดการ register fail ตัว BladeManager จะ return ถึงลำดับก่อนหลังของการ register ด้วย เนื่องจากว่าในบาง DataBlade module จำเป็นต้องพึ่ง DataBlade module ตัวอื่นด้วย ฉะนั้นจำเป็นต้องมีการ register ตามลำดับก่อนหลังตามการใช้งาน DataBlade module ในแต่ละตัว การจะดูว่าคำสั่ง SQL ที่ fail สามารถดูได้ที่ log file (ให้ดูในส่วนของ View Log file)

6.5.2.3 Unregistered DataBlade Module

ถ้าจะทำการ Unregistered DataBlade Module BladeManager จะทำการ remove ในแต่ละส่วนประกอบออกจากฐานข้อมูลโดยใช้คำสั่ง SQL DROP

การ Unregistered DataBlade module

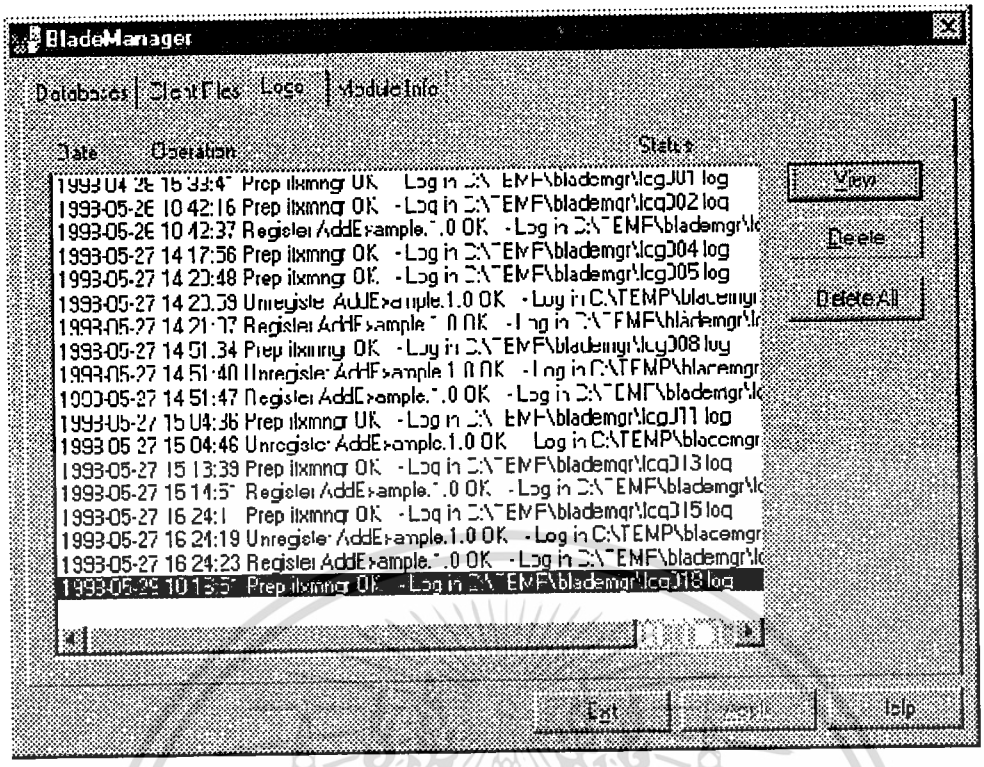
1. ใน **Database** tab ให้เราทำการเลือก database ที่จะทำการ Unregistered
2. ใน **Registered** list box เลือก DataBlade module ที่จะทำการ Unregistered
3. click **Remove** เพื่อเป็นการ move DataBlade module จาก **Registered** list box ไป **Available** list box
4. click **Apply** เพื่อเป็นการ Unregistered DataBlade module ออกจาก database

ถ้าหากมีการ fail เกิดขึ้น BladeManager จะ return ถึงลำดับก่อนหลังในการที่จะ Unregistered ออก และสามารถดูคำสั่ง SQL ที่ fail ได้จาก log file (ให้ดูในส่วนของ View Log file)

6.5.2.4 View Log File

BladeManager จะทำการสร้าง log file เมื่อมีการตรวจสอบฐานข้อมูลเพื่อการ register , การ register ,การ Unregistered ซึ่งถ้ามีการทำงานไหน fail ขึ้นมา คำสั่ง SQLที่ fail จะถูกระบุใน log file ด้วย

log files จะมีหมายเลข log file ที่ต่อเนื่องกันและจะระบุถึง time stamp ไว้ด้วย เราสามารถที่จะลบ log file เพื่อให้มี free space มากขึ้นได้



รูปที่ 6.5 แสดง Log file ของ DataBlade Module

การ view log file

1. ใน **Logs** tab ให้เลือก log file ที่ต้องการ
2. click **View**

การลบ log file

1. ใน **Logs** tab ให้เลือก log file ที่ต้องการ
2. click **Delete**

บทที่ 7
การหาค่าเอกลักษณ์ของลายนิ้วมือ (Minutia Extraction)

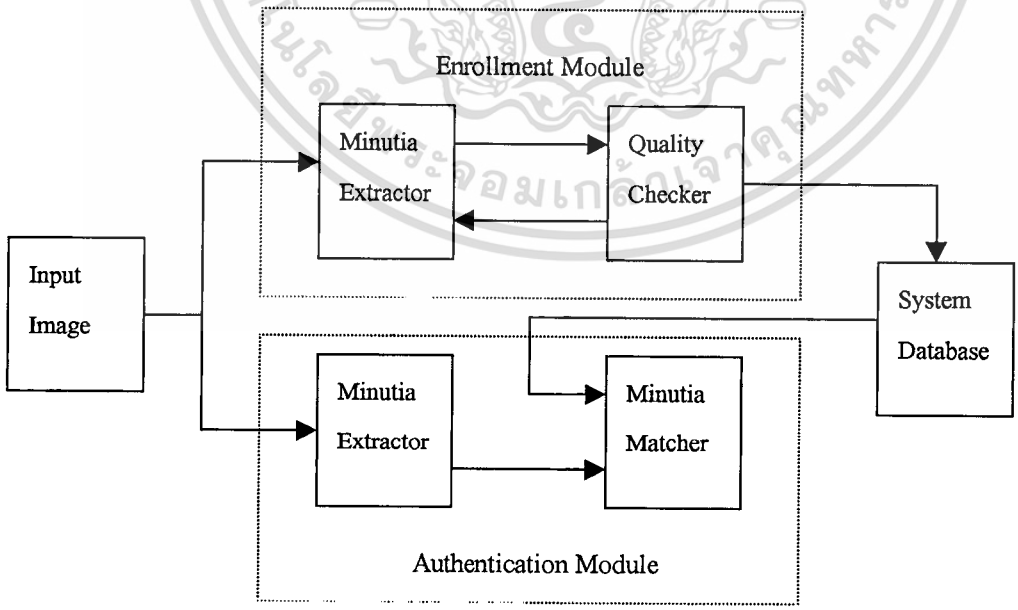
7.1 ความรู้เบื้องต้นเกี่ยวกับลายนิ้วมือ

ตามปกติแล้วที่ผิวหนังบริเวณปลายนิ้วมือ จะประกอบไปด้วยลายเส้น 2 ชนิดเรียงสลับกันไป เส้นหนึ่งเรียกว่า เส้นนูน (ridge) อีกเส้นหนึ่งเรียกว่าเส้นร่อง หรือ รอยร่อง (furrows) อยู่ตลอดไปทั่วลายนิ้วมือ โดยที่ลักษณะที่สำคัญของลายนิ้วมือ (minutia) ที่เราใช้ในการจำแนกบุคคลต่างๆ ให้ผิดแผกไปจากคนอื่นนั้นมีอยู่ด้วยกันหลายชนิด เช่น ridge ending , ridge bifurcation และ ridge crossover เป็นต้น

ลักษณะที่สำคัญ หรือ minutia ที่เราใช้วิเคราะห์ในระบบฐานข้อมูลลายนิ้วมือในที่นี้ก็คือ ในส่วนของ ridge ending และ ridge bifurcation

รูปที่ 7.1 แสดงลักษณะทั่วไปของ ridge ending และ ridge bifurcation

โดยทั่วไปแล้วในระบบการยืนยันและตรวจสอบความถูกต้องด้วยลายนิ้วมือ นั้นจะประกอบไปด้วยส่วนประกอบที่สำคัญอยู่ 2 ส่วนคือ Enrollment Module และ Authentication Module ดังรูปที่ 7.2



รูปที่ 7.2 ระบบการยืนยันความถูกต้องด้วยลายนิ้วมือ

7.2 Enrollment Module

เป็นส่วนที่ใช้ในการป้อนข้อมูลที่เป็นลายนิ้วมือเข้าไปเก็บไว้ในฐานข้อมูลลายนิ้วมือ (system database) โดยที่จะประกอบไปด้วย 2 ส่วนคือ

- Minutia Extractor เป็นส่วนที่ใช้ในการ extract หรือ สกัดเอาค่า minutia ออกมาจากภาพลายนิ้วมือที่ป้อนเข้ามา โดยในที่นี้ minutia ที่เราสนใจ คือ ridge ending และ ridge bifurcation
- Quality Checker หรืออาจเรียกได้ว่าเป็นการทำ post processing ก็ได้ ก็คือเป็นการนำค่า minutia ที่ได้มาทำให้ดีขึ้น ก่อนที่จะนำไปเก็บไว้ในฐานข้อมูลลายนิ้วมือ

7.3 Authentication Module

เป็นส่วนที่ใช้ในการตรวจสอบและยืนยันความถูกต้องว่าบุคคล ๆ นั้นมีสิทธิ์ที่จะเข้าไปใช้งานในระบบได้หรือไม่ หรือพูดอีกอย่างหนึ่งก็คือว่า เป็นการตรวจสอบลายนิ้วมือที่เข้ามากับลายนิ้วมือที่มีอยู่ในฐานข้อมูลว่าตรงกันหรือไม่ เพื่อที่จะอนุญาติให้บุคคลนั้นมีสิทธิ์ที่จะกระทำการใดๆ ต่อไปได้ ซึ่งจะประกอบไปด้วย 2 ส่วนคือ

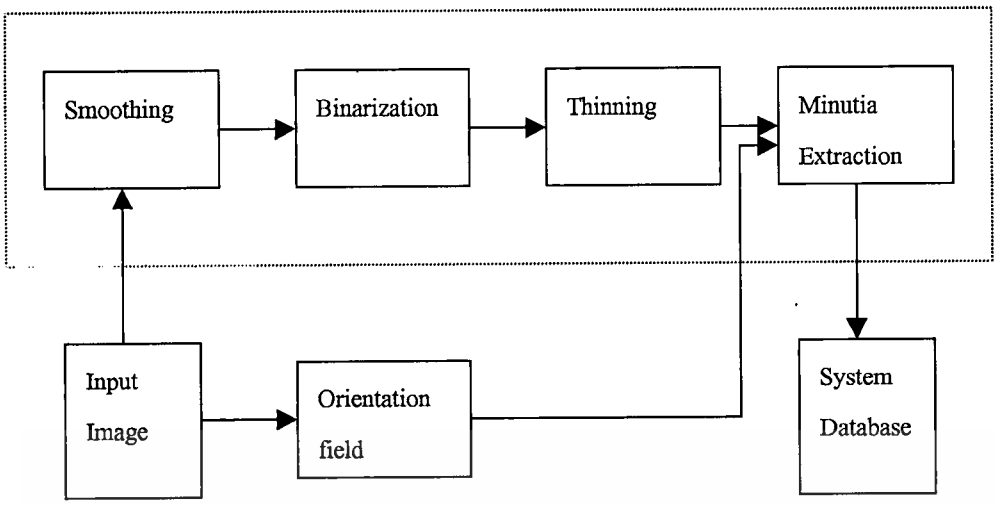
- Minutia Extractor เหมือนกันกับในส่วนของ enrollment module ก็คือเป็นส่วนที่ใช้ในการ extract หรือ สกัดเอาค่า minutia ออกมาจากภาพลายนิ้วมือที่ป้อนเข้ามา โดยในที่นี้ minutia ที่เราสนใจ คือ ridge ending และ ridge bifurcation เพื่อที่จะใช้ในการทำ matching ต่อไป
- Minutia Matcher เป็นส่วนที่ใช้ในการตรวจสอบว่า minutia ที่เข้ามากับ minutia ที่มีอยู่ในฐานข้อมูลลายนิ้วมือตรงกันหรือไม่

7.4 Minutia Extractor

จากที่ได้กล่าวมาแล้วจะเห็นได้ว่าในส่วนของ minutia extractor นั้นจะเป็นส่วนที่ใช้ในการ extract หรือการสกัดเอาค่าข้อมูลที่เป็น minutia ออกมาจากภาพลายนิ้วมือที่ป้อนเข้ามา โดยที่ในส่วนของ minutia ที่เราสนใจก็คือ ridge ending และ ridge bifurcation และค่าของ minutia ที่ได้จะอยู่ในรูปของค่าพิกัด (x,y,0) ของแต่ละ minutia ของภาพลายนิ้วมือที่ป้อนเข้ามานั้น หรือจะพูดอีกอย่างหนึ่งว่า จะได้ set ของ minutia ที่อยู่ในรูปของ (x,y,0) ของแต่ละภาพลายนิ้วมือที่ป้อนเข้ามาเพื่อที่จะนำค่า set เหล่านั้นไปเก็บไว้ในฐานข้อมูลลายนิ้วมือ เพื่อที่จะใช้ในการทำ matching ต่อไป

ในการที่จะ extract ให้ได้ค่า minutia นั้นจะต้องผ่านขั้นตอนและกระบวนการต่างๆ ดังต่อไปนี้

- Smoothing
- Binarization
- Thinning
- Orientation
- Extraction



รูปที่ 7.3 แสดงขั้นตอนในการทำ Minutia Extractor

7.4.1 Smoothing

เป็นวิธีที่ใช้ในการกำจัด noise ที่ปนมากับภาพลายนิ้วมือ ที่เป็นภาพ grey – levels วิธีการต่างๆ ที่มีใช้กันนั้นจะมีคุณสมบัติเป็น low pass filter ซึ่งในที่นี้จะใช้วิธีการของ average filtering โดยหลักการนี้จะใช้ 3x3 windows ภายใน windows จะประกอบไปด้วยจุดต่างๆ ตั้งแต่จุด P0 – P7 ดังรูปที่ 7.2 โดยที่เราจะมีจุด P0 เป็นจุดที่ใช้ในการพิจารณาถึง จุด P0 นี้จะเปลี่ยนแบบวนรอบไปจนครบทุกจุดภาพ ขณะที่แต่ละจุดภาพมีตำแหน่งเป็น P0 ก็จะมีการหาค่า P0 นั้นใหม่ด้วย สมาการ 7.1

$$P0 = \frac{1}{9}\sum_{i=0}^{8}(Pi) \quad \text{..... (7.1)}$$

P1	P2	P3
P7	P0	P4
P7	P6	P5

*รูปที่ 7.4 แสดงตำแหน่งของ 3*3 windows*

7.4.2 Binarization

การทำ binarization เป็นขั้นตอนในการทำภาพ grey – levels ให้เป็นภาพ ขาวดำโดยอาศัยค่าจาก histrogram เพื่อกำหนดจุด threshold ในการแปลงเป็น ภาพขาวดำ ดังสมาการที่ 7.2

$$g(x,y) = \begin{cases} 1; f(x,y) > T \\ 0; f(x,y) <= T \end{cases} \quad \text{.....(7.2)}$$

f(x,y) คือ ค่าของ grey levels ของตำแหน่ง x,y

T คือ ค่า threshold

g(x,y) คือ ค่าใหม่ที่ได้ของจุด (x,y)

7.4.2.1 การหาค่า Threshold

การหาค่า threshold อัตโนมัติด้วยวิธีการของ minimum cross – entropy thresholding จากในอดีตการเลือกค่า threshold นั้นยังคงต้องใช้มนุษย์เป็นคนทำ กล่าวคือมนุษย์จะเป็นผู้สังเกตุดูภาพที่ได้จากการทำ binarization ว่าค่า threshold ค่าไหนที่ให้รายละเอียดต่างๆของภาพได้ชัดเจนที่สุด ก็จะเลือกค่านั้นไปใช้งาน เป็นที่น่าสังเกตุได้ว่ามนุษย์ในยุคต่อๆ มาเริ่มตระหนักถึงวิธีการดังกล่าวนั้นใช้เวลามาก อันเนื่องมาจากข้อมูลที่เป็นลายนิ้วมือเริ่มมีจำนวนมากขึ้นเรื่อยๆ อีกทั้งยังมีความผิดพลาดในการเลือกค่า threshold เอง เช่น ปัญหาจากภาพลายนิ้วมือหนึ่ง ที่คนสองคนต่างให้ค่า threshold คนละค่าแล้วบอกว่าของตนเป็นค่าที่ถูกต้อง หรือคนคนเดียวที่ให้ค่า threshold มากกว่าหนึ่งค่าซึ่งตัวเขาเองไม่รู้ว่าจะเลือกค่าไหนดี ฉะนั้นจึงเป็นการดีถ้ามีการหาค่า threshold อย่างอัตโนมัติ ในที่นี้จะเสนอวิธีการหาด้วยวิธีการของ minimum cross – entropy thresholding โดยมีหลักการพื้นฐานที่ควรทราบมีดังนี้

$$D(P,Q) = \sum_{i=1}^{n} (Pi) \log \frac{Pi}{Qi} \quad \text{.....(7.3)}$$

D(P,Q) คือ distance ระหว่าง การแจกแจง 2 เหตุการณ์ความน่าจะเป็น

P คือ เป็นเซตของความน่าจะเป็นของเหตุการณ์แรก

ซึ่ง P = {p1,p2, . . .,Pn}

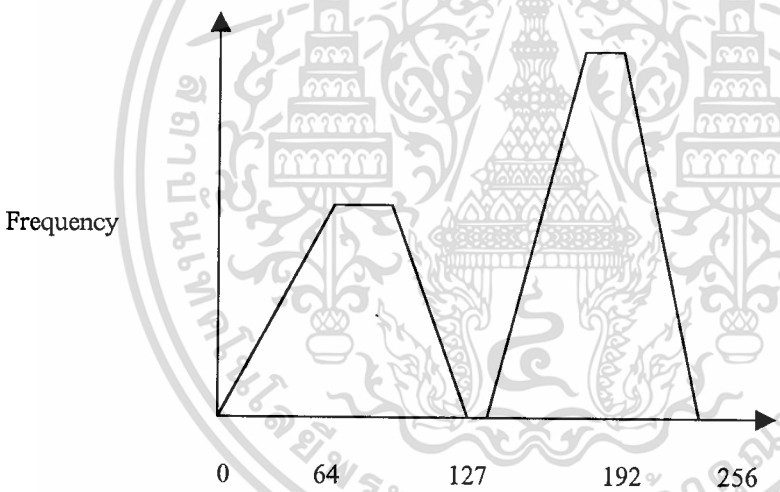
Q คือ เป็นเซตของความน่าจะเป็นของเหตุการณ์ที่สอง

ซึ่ง Q = {q1,q2, . . .,Qn}

เนื่องจาก $\sum_{i=1}^{n}(Pi) = \sum_{i=1}^{n}(Qi) = 1$, จากอสมาการของ Gibbs ($-\sum Pi \log Pi \leq -\sum Pi \log Qi$) จะพิสูจน์ได้ว่า $D(P,Q) \geq 0$, ถ้า $Pi \neq Qi$, ทุกๆ ค่า i แล้ว D(P,Q) ก็จะไม่มีคุณสมบัติของความสัมมาตร คือ $D(P,Q) \neq D(Q,P)$ ในงานประยุกต์บางอย่างจะมีการใช้งานในรูปแบบของ symmetric version ดังนี้คือ

$$Ds(P,Q) = D(P,Q) + D(Q,P) \quad \text{.....(7.4)}$$

มีส่วนที่ต้องทราบต่อไปคือ กำหนดให้ $F = [f(x,y)]_{MxN}$ เป็นภาพหนึ่งทาง digital ซึ่งมีระดับของ grey – level L พร้อมด้วย $f(x,y) \in \{0,1,\ldots,L\}$ กำหนดต่อไปให้ hi เป็นความถี่ของ grey - level ณ. ระดับที่ i และกำหนดให้ $Pi = \frac{hi}{(MxN)}$, ซึ่ง Pi คือ ความน่าจะเป็นของระดับ grey – level ที่ i นั้นๆ MxN เป็นขนาดของภาพ



รูปที่ 7.5 แสดง Synthetic histogram

จากรูป Synthetic Histogram ข้างบนนี้ กำหนดให้ S เป็นค่า Threshold และมีข้อสันนิษฐานเบื้องต้นที่ควรทราบคือ จากช่วง $[0-S]$ ใน grey – levels นั้นจะเป็นส่วนของ object และช่วง $[(S+1)-L]$ นั้นจะเป็นส่วนของ background ซึ่งในที่นี้ L = 255 (256 grey – levels) ในบางทฤษฎีจะใช้ช่วง $[0-(S-1)]$ เป็น object และ $[S-L]$ เป็น background แทนซึ่งก็ไม่ผิดเช่นกัน เมื่อทราบหลักการพื้นฐานแล้วเราก็มาเริ่มด้วยหลักการที่ใช้จริงดังนี้ เริ่มจากสมาการ

$$Ds(P,Q) = \sum Pi \log \frac{Pi}{Qi} + \sum Qi \log \frac{Qi}{Pi} \quad \text{.....(7.5)}$$

กำหนดให้ S เป็นค่า threshold สำหรับการทำ segmentation ดังนั้นถ้าเรามีการแจกแจงความน่าจะเป็นนี้กับ object ใน grey – level ก็จะกำหนดได้ดังนี้ $P_O = \{ \overset{O}{p}_1, \overset{O}{p}_2, \ldots, \overset{O}{p}_S \}$ และการแจกแจงความน่าจะเป็นกับ background เป็น $P_B = \{ \overset{B}{p}_{s+1}, \overset{B}{p}_{s+2}, \ldots, \overset{B}{p}_L \}$ ที่ซึ่ง

$$\overset{O}{p}_i = \frac{hi}{Ps} \text{ , } i = 1, 2, \ldots, S \text{ ; } Ps = \sum_{i=1}^{S} hi \quad \ldots\ldots\ldots\ldots\ldots(7.6)$$

$$\overset{B}{p}_i = \frac{hi}{MN - Ps} \text{ , } i = S+1, S+2, \ldots, L \quad \ldots\ldots\ldots\ldots\ldots(7.7)$$

ข้อสังเกต $\sum_{i=1}^{S} \overset{O}{p}_i = \sum_{i=S+1}^{L} \overset{B}{p}_i = 1$ ใน model Q ก็เช่นเดียวกันจะมี $Q_O = \{ \overset{O}{q}_1, \overset{O}{q}_2, \ldots, \overset{O}{q}_S \}$ สำหรับ object และ $Q_B = \{ \overset{B}{q}_{S+1}, \overset{B}{q}_{S+2}, \ldots, \overset{B}{q}_L \}$ สำหรับ background เมื่อถึงตอนนี้เราก็จะกำหนด cross entropy ของขอบเขต object ได้เป็น

$$D_O(S) = D_S(P_O, Q_O) \quad \ldots\ldots\ldots\ldots\ldots(7.7)$$

และ cross entropy ของขอบเขต background ได้เป็น

$$D_B(S) = D_S(P_B, Q_B) \quad \ldots\ldots\ldots\ldots\ldots(7.9)$$

สุดท้ายก็จะหา total cross entropy ดังสมาการต่อไปนี้

$$D(s) = D_O(s) + D_B(s) \quad \ldots\ldots\ldots\ldots\ldots(7.10)$$

$$= \sum_{i=1}^{S} \overset{O}{p}_i \log \left[\frac{\overset{O}{p}_i}{\overset{O}{q}_i} \right] + \sum_{i=1}^{S} \overset{O}{q}_i \log \left[\frac{\overset{O}{q}_i}{\overset{O}{p}_i} \right] +$$

$$\sum_{i=S+1}^{L} \overset{B}{p}_i \log \left[\frac{\overset{B}{p}_i}{\overset{B}{q}_i} \right] + \sum_{i=S+1}^{L} \overset{B}{q}_i \log \left[\frac{\overset{B}{q}_i}{\overset{B}{p}_i} \right]$$

จากสมาการดังกล่าวนี้เราจะต้องหาค่าน้อยที่สุดของ D(s) แล้วพิจารณาค่า S ให้เป็นค่า threshold นั้น ปัญหาจากสมาการที่ (3.4) นั้นเรายังไม่ทราบวิธีการหา Q_B และ Q_O เลย ดังนั้นขั้นตอนต่อไปจะศึกษาเกี่ยวกับวิธีการหาค่าดังกล่าวโดยใช้ model ของการแจกแจงความน่าจะเป็นของ Poisson

$$q_i^O = \frac{e^{-\lambda_O}}{i!}\lambda_O^i,\ i = 1, 2, \ldots, S; \ldots\ldots\ldots\ldots\ldots(7.11)$$

$$q_i^B = \frac{e^{-\lambda_B}}{i!}\lambda_B^i,\ i = S+1, S+2, \ldots, L; \ldots\ldots\ldots\ldots\ldots(7.12)$$

ซึ่งค่าของ λ_O และ λ_B หาได้ดังนี้

$$\lambda_O = \frac{\sum_{i=1}^{S} ihi}{\sum_{i=1}^{S} hi} \ldots\ldots\ldots\ldots\ldots(7.13)$$

$$\lambda_B = \frac{\sum_{i=S+1}^{L} ihi}{\sum_{i=S+1}^{L} hi} \ldots\ldots\ldots\ldots\ldots(7.14)$$

เมื่อสรุปหลักการต่างๆ มาเขียน Algorithm ได้ดังนี้

Begin

Mindiv = High – value; {กำหนดให้ Mindiv เป็นค่ามากที่สุดเท่าที่เป็นไปได้}

For $2 \le S \le L-1$ do

Begin

คำนวณหาค่า p_i^O, i = 1,2, . . . , S โดยใช้สมาการที่ (7.6)

คำนวณหาค่า p_i^B, i = S+1,S+2, . . . , L โดยใช้สมาการที่ (7.7)

คำนวณหาค่า λ_O, โดยใช้สมาการที่ (7.13)

คำนวณหาค่า λ_B, โดยใช้สมาการที่ (7.14)

คำนวณหาค่า q_i^O, i = 1,2, . . . , S โดยใช้สมาการที่ (7.11)

คำนวณหาค่า q_i^B, i = 1,2, . . . , S โดยใช้สมาการที่ (7.12)

```
                คำนวณหาค่า D(s)  โดยใช้สมาการ (7.10)
                if (Mindiv > D(s))
                Begin
                        Mindiv = D(s) ;
                        Threshold = S ;
                End
        End
End.
```

7.4.3 Thinning

เป็นวิธีการที่นำภาพ binary ที่ได้มาทำให้ภาพนั้นเป็นภาพโครงร่าง มีเพียงจุดเดียว (median axis) ซึ่งภาพ thinning ที่ได้นั้นจะเป็นภาพที่จะนำไปทำการ extract หาค่า minutia ต่อไป วิธีการ thinning ที่เราใช้นี้จะใช้วิธีการของ Zhang and Suen's method

โดยเราจะใช้ windows ขนาด 3x3 ที่ใช้เป็น เทมเพลตดังรูปที่ 7.6 เพื่อใช้ในการทำคอนโวลูชั่น โดยเราจะพิจารณาที่ตำแหน่ง P0 และ P0 เป็นจุดภาพสีขาว (1) โดยเราจะทำการตรวจสอบว่าจุดภาพที่ตำแหน่ง P0 นั้นจะเปลี่ยนเป็นจุดดำ (0) หรือไม่ โดยที่เราจะใช้เงื่อนไข 2 ส่วนในการพิจารณาคือ

P1	P2	P3
P7	P0	P4
P7	P6	P5

รูปที่ 7.6 แสดงตำแหน่ง เทมเพลต ของ Zhang and Suen

<u>ส่วนที่ 1</u>

1. A(P0) = 1
2. 1 < B(P0) < 7
3. P2 * P4 * P6 = 0
4. P4 * P6 * P7 = 0

<u>ส่วนที่ 2</u>

1. A(P0) = 1
2. 1 < B(P0) < 7
3. P2 * P4 * P7 = 0
4. P2 * P6 * P7 = 0

A(P0) = นับจำนวนจุดภาพที่มีคู่ 01 ในทิศทางเดียวกัน

B(P0) = จำนวนจุดภาพที่เป็น 1

โดยที่ถ้าเงื่อนไขทั้ง 4 ข้อในแต่ละส่วนเป็นจริงทั้งหมด จุดภาพนั้นจะถูกเปลี่ยนจาก ภาพสีขาว (1) เป็นภาพสีดำ (0) โดยจะทำการตรวจสอบจนไม่มีการเปลี่ยนแปลง เป็นการสิ้นสุดการทำ thinning ด้วยวิธีการของ Zhang and Suen

7.4.4 Orientation

เป็นวิธีการหาค่าของมุม ของแต่ละ minutia ต่างๆ สำหรับภาพลายนิ้วมือที่เข้ามา โดยในที่นี้เราจะใช้วิธีการของ least mean square orientation estimation algorithm ซึ่งประกอบไปด้วยขั้นตอนต่างๆ ดังนี้ คือ

- แบ่งภาพลายนิ้วมือที่เข้ามา หรือ image ออกเป็น block ที่มีขนาดเป็น w x w (16 x 16)
- หาค่า gradiant ของภาพลายนิ้วมือที่เข้ามาหรือ image Curlx(i,j) และ Curly(i,j) ของแต่ละ pixel ของภาพที่ตำแหน่ง (i,j) โดยในที่นี้เราใช้ Mask ที่เป็น Sobel operator ช่วย
- หาค่าของ loacal orientation หรือหาค่าของมุม ของแต่ละ block center ที่ตำแหน่ง pixel (i,j) โดยใช้สมาการดังต่อไปนี้

$$Vx(i,j) = \sum_{u=i-(w/2)}^{i+(w/2)} \sum_{v=j-(w/2)}^{j+(w/2)} 2Curlx(u,v) * Curly(u,v) \quad \text{.....(7.15)}$$

$$Vy(i,j) = \sum_{u=i-(w/2)}^{i+(w/2)} \sum_{v=j-(w/2)}^{j+(w/2)} Curlx^{2}(u,v) - Curly^{2}(u,v) \quad \text{.....(7.16)}$$

ดังนั้น

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\frac{Vy(i,j)}{Vx(i,j)} \quad \text{.....(7.17)}$$

จะเห็นได้ว่าค่า ที่ได้จะเป็นค่า least square estimate ของ local ridge orientation ที่ block center ที่จุด pixel (i,j)

7.4.5 Extraction

เป็นการ extract หรือ สกัดเอาค่า minutia ออกจารภาพที่ได้จากการทำ thinning ในที่นี้ minutia ที่เราสนใจก็คือ ในส่วนของ ridge ending และ ridge bifurcation โดยที่เราจะใช้ Mask หรือ windows ขนาด 3 x3 ช่วย ในที่นี้เราจะพิจารณาที่จุด P0 โดยใช้สมาการดังต่อไปนี้

$$CN = \sum_{i=1}^{8} |P(i+1) - P(i)| \quad \text{.....(7.17)}$$

P1	P2	P3
P7	P0	P4
P7	P6	P5

*รูปที่ 7.7 แสดง windows ขนาด 3*3 ที่ใช้ในการทำ extract*

CN คือ ค่า condition number

ถ้า CN = 2 และ จุด P0 มีค่าเป็น 1 แล้วจุด P0 นั้นจะเป็นจุด ridge ending

ถ้า CN = 6 และ จุด P0 มีค่าเป็น 1 แล้วจุด P0 นั้นจะเป็นจุด ridge bifurcation

บทที่ 8

การจับคู่ของลายนิ้วมือ (Minutia Matching)

ระบบตรวจสอบลายนิ้วมือโดยอัตโนมัติ (Automatic Fingerprint Identification System :AFIS) จะประกอบไปด้วยส่วนประมวลผลหลาย ๆ ส่วน และเนื่องจากว่าในลายนิ้วมือหนึ่ง ๆ อาจจะประกอบด้วย Ridge ending และ Ridge bifurcation หลาย ๆ ตัว และอาจจะรวมถึงข้อมูลที่ผิดพลาดรวมอยู่ด้วย หรือน้อยส์ (noise) ของลายนิ้วมือนั่นเอง เพราะฉะนั้นจึงเป็นโอกาสยากมากที่จะตรวจพบในแบบ หนึ่งต่อหนึ่ง (one-to-one) ได้ ซึ่งจริง ๆ แล้วระบบในเชิงพานิชย์จะทำการหาลายนิ้วมือที่มีโอกาสเป็นไปได้ที่จะใช่ลายนิ้วมือจริงที่ต้องการ โดยปกติจะเลือกเอามา 10 ลายนิ้วมือแรกที่มีโอกาสสูงที่สุด และหลังจากนั้นจะทำการเลือกโดยผู้เชี่ยวชาญที่เป็นมนุษย์อีกที

ในการ matching ของลายนิ้วมือ สามารถจะทำการกลั่นกรองข้อมูลที่อยู่ในฐานข้อมูลก่อนที่จะทำการคำนวณค่าคะแนนของลายนิ้วมือที่ได้จากการป้อนอินพุทกับลายนิ้วมือที่อยู่ในฐานข้อมูลได้ โดยข้อมูลที่จะทำการกลั่นกรองจะได้จากการป้อนอินพุทของผู้ใช้เอง ซึ่งข้อมูลเหล่านี้ เช่น ประวัติส่วนตัว, รหัสประจำตัว ฯลฯ หรืออาจพูดโดยสรุปว่าจะทำการตรวจสอบเงื่อนไขที่เป็นข้อความหรือตัวเลขที่มีอยู่ก่อน เพราะจะเป็นการลดจำนวนของลายนิ้วมือที่จะ return มาจากฐานข้อมูล ผลลัพธ์ที่ได้จากในขั้นตอนแรกนี้ก็จะนำไปจัด Class ของลายนิ้วมือ (Classification Fingerprint) โดยจะเลือกเอาเฉพาะลายนิ้วมือที่อยู่ใน Class เดียวกับลายนิ้วมือที่เป็นอินพุท จากนั้นในขั้นตอนสุดท้ายก็จะทำการคำนวณหาค่าคะแนนของการเทียบกันของลายนิ้วมือที่อยู่ในฐานข้อมูลกับลายนิ้วมือที่ถูกป้อนเข้ามาเป็นอินพุท

ในตัวงานที่ทางคณะผู้จัดทำได้สร้างขึ้นมานั้นจะไม่มีในส่วนของการจัด Class ของลายนิ้วมือ ซึ่งก็คือจะทำการคำนวณหาค่าคะแนนกับทุก row ที่อยู่ในฐานข้อมูลจากนั้นจึงนำมาเลือกเอา 10 อันดับแรกที่มีค่าที่ดีที่สุด โดยถ้าจะทำการพัฒนาต่อไปสามารถเพิ่มในส่วนของการจัด Class ของลายนิ้วมือได้ (Classification Fingerprint)

8.1 การ Matching ของลายนิ้วมือ

เมื่อทำการ Query ฐานข้อมูล ก็จะทำการ Return ลายนิ้วมือ 10 อันดับแรกที่มีค่าคะแนนมากที่สุด โดยจะทำการ Return มาเป็นเซ็ท โดยขั้นตอนในการ Matching สามารถแบ่งออกเป็นสามขั้นตอนย่อยได้ คือ

1) Registration
2) Minutiae pairing
3) การคำนวณหาค่าคะแนน (Computation matching score)

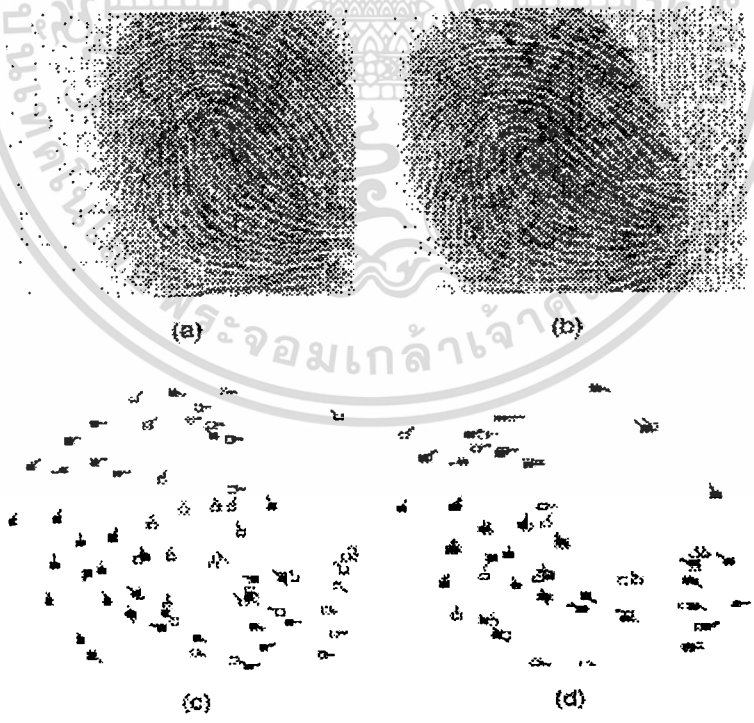
8.1.1 Registration

เราจะใช้หลักการของ Hough Transform[10] ในการนำมาหมุนจุดต่าง ๆ ที่อยู่บนลายนิ้วมือเพื่อเทียบหาค่าที่ดีที่สุดในการทาบลายนิ้วมือ

เราจะทำการแบ่งลายนิ้วมือออกเป็นสองส่วนหลัก ๆ คือ ลายนิ้วมือที่เป็นอินพุทเข้ามาและลายนิ้วมือที่อยู่ในฐานข้อมูล โดยจะแบ่งออกเป็นสองเซ็ทหลัก ๆ คือเซ็ท P และเซ็ท Q ซึ่งจะใช้สัญลักษณ์ดังต่อไปนี้

$$P = \{(p_x^1, p_y^1, \alpha^1),\ldots,(p_x^P, p_y^P, \alpha^P)\},$$
$$Q = \{(q_x^1, q_y^1, \beta^1),\ldots,(q_x^Q, q_y^Q, \beta^Q)\}$$

โดยให้ $|P| = P$ และ $|Q| = Q$ (p_x^i, p_y^i, α^i) คือสาม feature หลัก ๆ (ตำแหน่ง x,y ในลายนิ้วมือกับค่าของ orientation) ที่มีอยู่ในเซ็ท P และในลายนิ้วมืออันที่สองเราก็ให้เป็นเช่นเดียวกัน และเราให้ลายนิ้วมืออันที่สองคือลายนิ้วมือที่เป็นอินพุทเข้ามา สมมติให้ลายนิ้วมือเซ็ทQ เป็น version ที่ได้ทำการ rotate ,shift ,scale แล้วกับลายนิ้วมือเซ็ท P ซึ่งเมื่อนำมาทาบกันก็จะให้จุดที่ตรงกันมากที่สุด เพราะฉะนั้นในการทาบลายนิ้วมือหน้าที่ของขั้นตอนนี้คือจะหาค่าที่จะทำการ transform เซ็ททั้งสองให้ตรงกันมากที่สุด นั่นคือจะหาค่าที่จะทำการ rotate ,shift, scale ที่จะทำให้มีจุดตรงกันมากที่สุด ดังรูปที่จะแสดงดังต่อไปนี้



รูปที่ 8.1 แสดงการทาบกันของลายนิ้วมือ โดยรูป (a) , (b) คือรูปของลายนิ้วมือเดียวกัน
รูป (c) คือก่อนการทำ Registration, (d) คือหลังจากทำ Registration แล้ว

โดยปกติ Hough Transform จะเก็บสถิติที่เกี่ยวกับการหมุนในแต่ละจุดว่ามีจุดตรงกันเท่าไหร่ ซึ่งอาจจะทำให้ต้องใช้หน่วยความจำขนาดใหญ่เพื่อการเก็บตัวเลขเหล่านี้ รูปแบบของการแปลงจะเป็นดังนี้

$$F_{s,\theta,\Delta x,\Delta y}\begin{pmatrix}x\\y\end{pmatrix} = s\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}$$

ให้ s,θ และ (Δx, Δy) ก็คือค่า scale , rotation และ shift พารามิเตอร์ตามลำดับโดยให้พารามิเตอร์แต่ละตัวมีขอบเขตดังนี้

$$s \in \{s_1,\ldots,s_K\}, \theta \in \{\theta_1,\ldots,\theta_L\}, \Delta x \in \{\Delta x_1,\ldots,\Delta x_M\}$$

และ

$$\Delta y \in \{\Delta y_1,\ldots,\Delta y_N\}$$

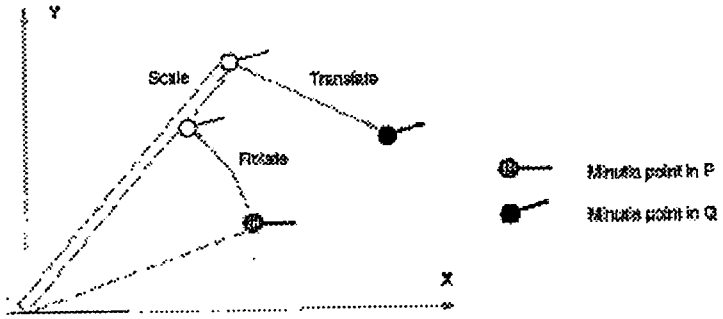
ค่าสถิติที่ได้ทำการคำนวณจากการ transform ในแต่ละครั้งจะถูกบันทึกรวมอยู่ใน array accumulator A โดยที่ A(k,l,m,n) คือตัวคอยบันทึกเหตุการณ์ต่าง ๆ ที่ได้ทำการ transform ในแต่ละ $F_{s,\theta,\Delta x,\Delta y}$

เราจะกำหนดให้ในแต่ละคู่ (**p** , **q**) โดยที่ **p** = (p_x^i,p_y^i) เป็นจุดในเซ็ท$\boldsymbol{P}$ และ **q**=(q_x^i,q_y^i) เป็นจุดในเซ็ท$\dot{\boldsymbol{Q}}$ และจะทำการคำนวณหาค่าการ transform เพื่อ map จาก **p** ไป **q** ทั้งหมดที่จะเป็นไปได้ และจะทำการเพิ่มว่า ณ ที่การ transform นี้ได้มีการนำไปคำนวณเพิ่มขึ้นอีกหนึ่งครั้ง โดยจะทำการบันทึกลงใน array A

ในแต่ละค่า (s_k, θ_l) ก็จะให้ค่า (Δx, Δy) เสมอ สมการที่จะใช้ในการหาค่าคือ

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \mathbf{q} - s_k\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\mathbf{p}$$

และลักษณะของการคำนวณในแบบจุดต่อจุดจะเป็นดังรูปข้างล่างนี้



รูปที่ 8.2 แสดงถึงการคำนวณในลักษณะจุดต่อจุด

เมื่อเราทำการคำนวณและบันทึกลงใน A แล้ว ก็จะนำมาเลือกเอาค่าที่มีค่าความถี่ที่มากที่สุด เพื่อที่จะได้นำเอาพารามิเตอร์ ต่างๆ มาทำการ transform ลายนิ้วมือที่อยู่ในสมการของ Hough ข้างต้นนั่นเอง อัลกอริทึมที่จะทำการ Registration จะแสดงดังรูปข้างล่างนี้

```
PROCEDURE Hough
  A(k, l, m, n) := 0,
    k = 1, ..., K;
    l = 1, ..., L;
    m = 1, ..., M;
    n = 1, ..., N
  FOR (p_x, p_y, α) ∈ P DO
    FOR (q_x, q_y, β) ∈ Q DO
      FOR θ ∈ {θ_1, ..., θ_L} DO
        IF α + θ = β THEN
          FOR s ∈ {s_1, ..., s_L} DO
(Δx, Δy)^T := (q_x, q_y)^T − s_k [ cos θ_l   sin θ_l ; −sin θ_l   cos θ_l ] × (p_x, p_y)^T
            Add evidence for F_{s_k, θ_l, Δx, Δy}
          END FOR
        END IF
      END FOR
    END FOR
  END FOR
  Result := arg max_{k,l,m,n} A(k, l, m, n)
END PROCEDURE
```

รูปที่ 8.3 อัลกอลิทึมในการ Registration

8.1.2 Minutia Pairing

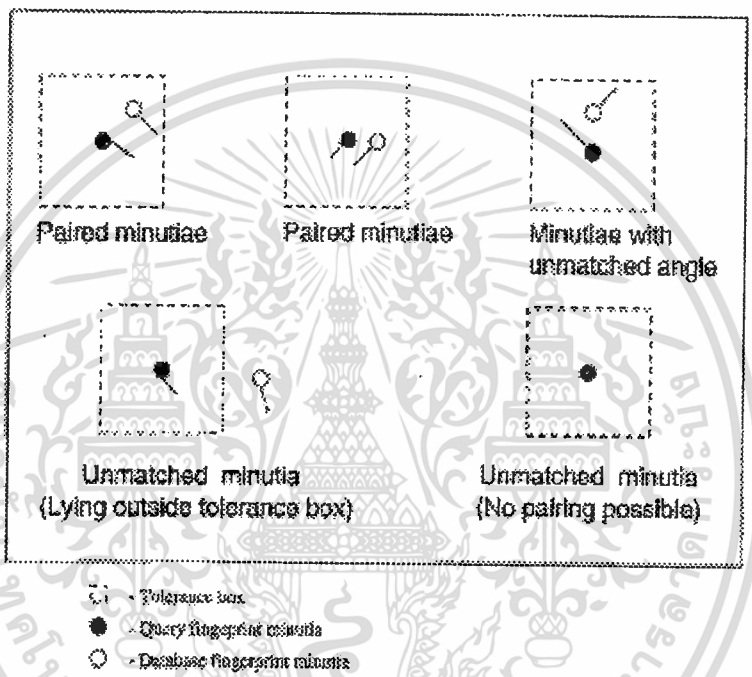
หลังจากที่มีการทำ Registration แล้วเราก็จะได้เซ็ทมาจำนวนสองเซ็ทคือ เซ็ทที่ได้ทำการแปลงมาเรียบร้อยแล้วและอีกเซ็ทหนึ่งก็คือเซ็ทที่คงเหลืออยู่อีกตัวหนึ่งในสมการ ต่อไปเราก็จะทำการจับคู่ตัว minutia โดยที่ minutia จะจับคู่กันได้หรือ match กันนั่นเอง ก็คือจะต้องมีค่า feature (x, y, θ) เท่ากัน

หรือมีค่าที่ใกล้เคียงกันที่อยู่ในขอบเขตที่เรากำหนดซึ่งยังสามารถยอมรับได้ว่ามีค่าที่เท่ากัน เพราะฉะนั้น เมื่อเรานำมาพิจารณาจากรูปด้านล่างนี้ก็จะมีอยู่สามกรณีที่มีโอกาสจะเกิดจะแสดงดังต่อไปนี้

กรณีที่ 1) คือค่า feature (x, y, θ) ของลายนิ้วมือในฐานข้อมูลเท่ากันกับค่า feature (x, y, θ) ของลายนิ้วมือที่อินพุทเข้ามา

กรณีที่ 2) คือค่า feature (x, y, θ) มีค่า x ,y ที่เท่ากันเท่านั้น

กรณีที่ 3) คือ ไม่มีค่าไหนที่เท่ากันเลย



รูปที่ 8.4 แสดงถึงการจับคู่ (match) ของ feature แต่ละ feature

เราจะถือว่าในกรณีแรกเท่านั้นที่สามารถจับคู่ หรือ match กันได้

8.1.3 การคำนวณหาค่าคะแนน (Computation matching score)

ในการจับคู่นั้นเราจะต้องมีการบันทึกไว้ด้วยว่ามีการจับคู่ได้ทั้งหมดกี่คู่แล้ว และเราจะต้องทำการหาหรือจับคู่เฉพาะกรอบของทั้งสองลายนิ้วมือที่เป็น intersection กันเท่านั้น

กรอบที่ intersection กันก็คือ กรอบหรือขอบเขตของแต่ละลายนิ้วมือที่เป็นพารามิเตอร์อยู่ เราจะเรียกว่า Bounding box ซึ่งเราจะทำการหาหลังจาก Registration แล้วก็ได้ จากนั้นจึงนำมาหาค่าขอบเขตที่จะพิจารณาเพื่อทำการ match feature ที่อยู่ในขอบเขตที่เราหามาได้ ขอบเขตที่ว่านี้คือขอบเขตหรือกรอบที่เป็นส่วนที่ขอบเขตของลายนิ้วมือทั้งสองมีการ intersection กัน

เมื่อเราหากรอบที่ intersection แล้วก็จะทำการนับจำนวน feature ของแต่ละลายนิ้วมือจากนั้นจึงนำมาเข้าสูตรดังต่อไปนี้

$$MS(q,r) = \frac{m^{r} * m^{r}}{n_r^b * n_q^b}$$

m^r คือ จำนวนที่มีการ match กัน

n_r^b คือ จำนวน feature ของลายนิ้วมือในฐานข้อมูล ที่อยู่ในขอบเขตที่มีการ intersection กัน

n_q^b คือ จำนวน feature ของลายนิ้วมืออินพุท ที่อยู่ในขอบเขตที่มีการ intersection กัน

สรุปอัลกอริทึมในการ matching ลายนิ้วมือจะแสดงไว้ดังรูปข้างล่างนี้

Input: A set of n_q minutiae points in the query fingerprint f^q and the rolled fingerprint database $f^D = \{f^r\}_{r=1}^{N}$. Let the rth database fingerprint have n_r minutiae points $f^r(f_1^r, f_2^r, \ldots f_{n_r}^r)$.

Output: A list of top 10 records from the database with matching score greater than a threshold T.

Begin

FOR r = 1 to N DO

1. Register the database fingerprint with respect to the query fingerprint.
2. Compute the common bounding box for the query and reference fingerprints. Let the query print have n_q^b and reference print have n_r^b minutiae in this box.
3. Set the number of paired minutiae for the rth database fingerprint m^r to zero.

   FOR i = 1 to n_q^b DO

   Compute the tolerance vector for the ith minutiae points in the rth database fingerprint feature vector f_i^r.

   If it can be paired with a query minutiae, then increment m^r and mark the query minutia paired. A paired query minutiae will not be paired again.

   END FOR
4. Compute the matching score (MS (q, r)):

$$MS(q,r) = \frac{m^r * m^r}{n_r^b * n_q^b}.$$

5. Update a list of top 10 scoring database fingerprints.

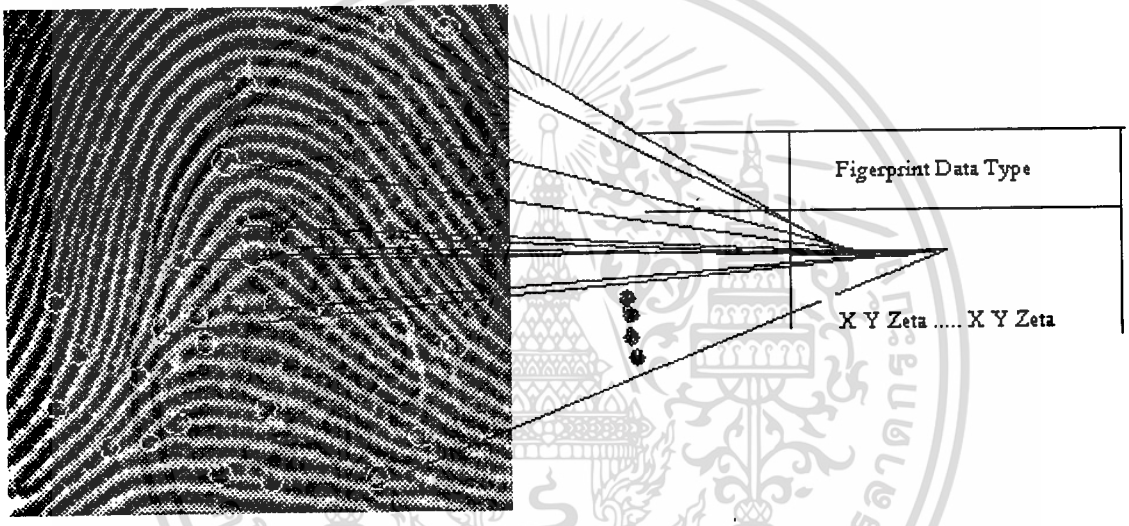
END FOR

END

รูปที่ 8.5 สรุปอัลกอริทึมที่ใช้ในการ Matching

บทที่ 9
ชนิดข้อมูลลายนิ้วมือ (Fingerprint data type)

ในการสร้าง Data type ชนิดหนึ่งที่ยังไม่มีใครสร้างมาก่อนเลยนั้น อาจจะต้องสร้างจาก User-defined type ของ Informix ซึ่งเราก็ได้พูดไว้แล้วในตอนต้น Fingerprint data type ในที่นี้ ก็คือการสร้าง Opaque type ชนิด varying-length นั่นเอง เพื่อทำการเก็บ feature ของแต่ละรูปลายนิ้วมือ ซึ่งในแต่ละรูปลายนิ้วมือจะมี feature ไม่เท่ากันทุกรูปลายนิ้วมือ จึงต้องทำการเก็บในแบบ varying-length Opaque type โดยลายละเอียดที่จะทำการเก็บมีดังต่อไปนี้



รูปที่ 9.1 ลักษณะการเก็บลงในฐานข้อมูล

9.1 การสร้าง Internal structure ของ Fingerprint data type

เนื่องจากว่า feature ของลายนิ้วที่ได้ทำการ Extraction ออกมาแล้วนั้น ในแต่ละจุดของ feature จะมีค่า x,y,θ โดยที่ค่า x,y เป็นตำแหน่ง x,y ในจุดภาพของรูปลายนิ้วมือซึ่งเป็นจุดที่มีลักษณะ ending หรือ bifurcation และค่า θ จะเก็บทิศทางของจุดเหล่านั้นว่ามีทิศทางเป็นอย่างไร เพื่อที่จะทำการ matching อีกต่อไป

```
/* BladeSmith 3.40.TXXXXX typedef pnt_t */
typedef struct
{
mi_integer          X;
mi_integer          Y;
mi_integer          Zeta;
}
pnt_t;
/* Warning: Do not modify. pnt_t checksum: 0 */

/* BladeSmith 3.40.TXXXXX typedef fing_t */
typedef struct
{
pnt_t               data[1];
}
fing_t;
```

รูปที่ 9.2 แสดงโครงสร้างภายในของชนิดข้อมูลลายนิ้วมือ

จะเห็นว่าเราไม่ได้เก็บชนิดของ feature ในแต่ละ feature หรือก็คือตัวที่บ่งบอกว่าเป็น ending หรือ bifurcation เหตุผลที่ไม่ได้เก็บเนื่องจากการเพิ่ม filed ขึ้นมาหนึ่งชนิด ซึ่งเป็นตัวบอกว่า feature นั้นมีชนิดเป็นอะไร ไม่ได้ช่วยให้การ matching เร็วขึ้นสักเท่าไรนัก อีกทั้งในการเก็บนั้นในแต่ละรูปมีหลายจุด และในฐานข้อมูลก็มีหลาย ๆ row ทำให้มีการสิ้นเปลืองเนื้อที่ในการเก็บโดยใช้เหตุ แต่ว่าจะมีเพิ่มไว้ก็ไม่เสียหายนัก

โครงสร้างภายในของ Fingerprint data type ที่ชื่อ fing_t จะประกอบไปด้วย mi_integer สาม filed โดยได้มีการอ้างมาจาก pnt_t ซึ่งเป็น type ที่แทน feature ในแต่ละ feature ของในรูปภาพลายนิ้วมือ ตัว type mi_integer เทียบได้กับ type integer ของฐานข้อมูล mi_integer นั้นเป็น type ของ DataBlade API

นอกจากนี้เรายังได้สร้าง Data Type เพื่อไว้ใช้ในการดีบักการทำงานของการ Query เพื่อทำการ Search หาลายนิ้วมือในฐานข้อมูล Data Type ตัวนี้คือ HoughIdx ซึ่งจะประกอบไปด้วย mi_integer 4 fields โดยในแต่ละ field คือ S, Z, X, Y เพื่อใช้ในการ return ค่าจากการคำนวณของ Hough Transforms ลักษณะการใช้งานก็เช่นเดียวกับการใช้งานของ fing_t และ pnt_t

9.2 การเขียน Support function ของ Fingerprint data type

ตารางต่อไปนี้จะแสดง Support function เบื้องต้นที่ทำงานร่วมกับชนิดข้อมูล fing_t

Support Function	Function Signature
Input	Fing_t * fing_tInput(extrnl_format) Mi_lvarchar *extrnl_format;
Output	mi_lvarchar *fing_tOutput(intrnl_format) fing_t * intrnl_format;
Receive	Fing_t * fing_tReceive(client_intrnl_format) Mi_sendrec * client_intrnl_format;
Send	mi_sendrecv *fing_tSend(srvr_intrnl_format) fing_t * srvr_intrnl_format;
Import	Fing_t * fing_tImport(extrnl_bcopy_format) Mi_impexp * extrnl_bcopy_format;
Export	mi_impexp * fing_tExport(intrnl_bcopy_format) fing_t * intrnl_bcopy_format;
Importbinary	Fing_t * fing_tImbin(client_intrl_bcopy_format) Mi_impexpbin * client_intrnl_bcopy_format;
Exportbinary	mi_impexpbin * fing_tExpbin(srvr_intrnl_bcopy_format) fing_t * srvr_intrnl_bcopy_format;

รูปที่ 9.3 แสดง support function ของชนิดข้อมูลลายนิ้วมือ

Code จริง ๆ ของชนิดข้อมูล fing_t จะถูกคอมไพล์ด้วย Visual C++ และจะถูกเก็บไว้ใน share library ที่อยู่ใน subdirectory extend ใน Database root Directory โดยจะมีชื่อว่า Fingerprint.bld

ฟังก์ชั่น input และ output ของชนิดข้อมูล fing_t จะถูกกำหนดให้มีรูปแบบภายนอก (external representation) ดังนี้

'X Y Zeta X Y Zeta ... X Y Zeta'

ฟังก์ชั่น input หรือ fing_tInput จะยอมรับค่าข้อมูลที่เป็น string ดังแสดงไว้ข้างบนนี้และจะทำการแปลง string ชุดนั้นเป็นชนิดข้อมูล fing_t เพื่อที่จะทำการส่งผ่านไปให้ database server ทำการเก็บลงฐานข้อมูลอีกที และในฟังก์ชั่น output หรือ fing_tOutput ก็เช่นเดียวกันแต่ว่าจะกระทำในทางตรงกันข้ามคือจะนำค่าในรูปแบบของชนิดข้อมูล fing_t ที่เก็บอยู่ในฐานข้อมูลออกมาแสดงผลเป็น string โดยมีรูปแบบตามที่แสดงไว้ข้างบนนี้

ส่วนฟังก์ชั่น send, receive ก็จะทำเช่นเดียวกันแต่จะทำการแปลงจาก internal structure ของ client กับ server เท่านั้นเอง

ในส่วนของ Support Function ต่าง ๆ เหล่านี้ถ้าใช้ DataBlade Developr Kit 3.4 เมื่อเราทำการกำหนดคุณสมบัติต่าง ๆ ไว้ให้หมดแล้ว ก็จะทำการ generate source code ไว้ให้เลยซึ่งจะเป็นการสะดวกมากในการพัฒนา เพราะฉะนั้นเราจึงสามารถกำหนดฟังก์ชั่นต่างๆ เผื่อไว้ใช้งานในอนาคตได้

9.3 Register ลงในฐานข้อมูล

ในการ Register ลงในฐานข้อมูลคือการสร้าง Opaque type นั่นเอง และเป็นการสร้าง Function ลงในฐานข้อมูล

การสร้าง Opaque type ลงในฐานข้อมูลจะใช้คำสั่ง CREATE OPAQUE TYPE โดยจะกำหนด Parameter ให้เป็น varying-length

```
create opaque type fing_t(
        internallength = variable,
        maxlen = 6144,
        alignment = 4
);
```

รูปที่ 9.4 แสดงการสร้างชนิดข้อมูล Opaque type

โดยคำสั่งของการสร้าง Opaque Type มีรายละเอียดคือ เป็นชนิดข้อมูล Opaque Type แบบ Varying-length ที่มีขนาดสูงสุด 6144 bytes และมีขอบเขตในการ alignment 4 bytes

ส่วน Function จะใช้คำสั่ง CREATE FUNCTION ในการสร้าง Function ซึ่งในขั้นตอนนี้จะเป็นการสร้าง Support function ต่าง ๆ ที่จะต้องใช้กับ Opaque type ที่เราได้ทำการสร้างขึ้นไว้แล้ว และในส่วนนี้นี่เองจะมีการสร้าง Cast อีกด้วย โดย Cast จะทำหน้าที่ในการแปลงข้อมูลชนิดหนึ่งไปเป็นข้อมูลอีกชนิดหนึ่ง การสร้างจะใช้คำสั่ง CREATE CAST

```
create function fing_tIn (lvarchar)
returns fing_t
external name "$INFORMIXDIR/extend/Fingerprint.1.7.3p.41/Fingerprint.bld
(fing_tInput)"
language c;

create implicit cast ( lvarchar as fing_t  with  fing_tIn );

create function fing_tExpB (fing_t)
returns impexpbin
external name "$INFORMIXDIR/extend/Fingerprint.1.7.3p.41/Fingerprint.bld
(fing_tExportBinary)"
language c;

create cast  ( fing_t as impexpbin with  fing_tExpB );
```

รูปที่ 9.5 แสดงการสร้าง Cast function

รายละเอียดเพิ่มเติมสามารถหาดูได้จาก file objects.sql ที่อยู่ใน subdirectory extend

9.4 ทำการ Grant สิทธิ์ให้ผู้อื่นได้ใช้งาน

ในคำสั่ง CREATE OPAQUE TYPE จะให้สิทธิ์กับผู้สร้างเท่านั้น เพราะฉะนั้นจึงต้องมีการให้สิทธิ์กับผู้อื่นให้ได้ใช้งานกัน

```
grant usage on type fing_t to public;
grant execute on function fing_tIn (lvarchar) to public;
grant execute on function fing_tOut (fing_t) to public;
.
grant execute on function fing_tExpB (fing_t) to public;
```

รูปที่ 9.6 แสดงการให้สิทธิ์ผู้อื่นในการใช้งาน Opaque type

9.5 ทำการสร้างฟังก์ชั่นที่จะเรียกใช้จาก SQL Statement

ฟังก์ชั่นที่เราได้ทำการสร้างขึ้นและสามารถทำการเรียกได้ใช้นั้นแบ่งออกเป็นสองประเภท ประเภทแรกคือฟังก์ชั่นที่ใช้ในการ Matching ส่วนในระเภทที่สองคือฟังก์ชั่นที่ใช้ในการช่วยดีบัก ในฟังก์ชั่นประเภทที่สองนี้เราจะไม่ขอกล่าวถึง

การสร้างฟังก์ชั่น Matching นั้น ในที่นี้ก็คือการสร้าง User-defined function นั่นเอง เราได้สร้างฟังก์ชั่นที่จะถูกเรียกใช้งานจาก SQL Statement อยู่ 1 ฟังก์ชั่นคือ ฟังก์ชั่น score() และ score2() เพื่อใช้ในการคำนวณค่า matching เพื่อนำไปคัดเลือกเอา 10 อันดับแรกที่ดีที่สุดต่อไป

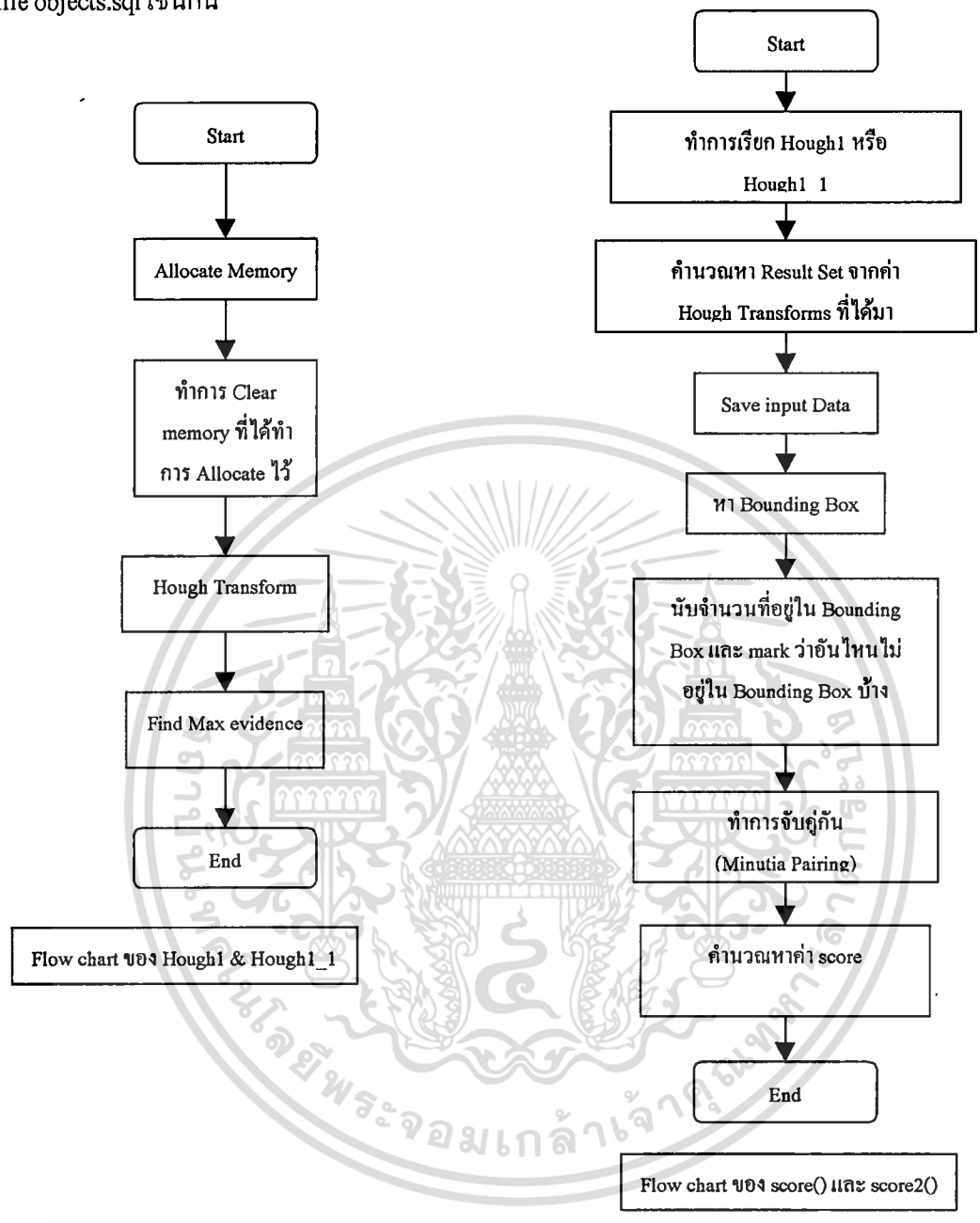
หน้าที่ของทั้งสองฟังก์ชั่นนั้นเหมือนกันอีกทั้งยังให้ผลลัพธ์ที่เหมือนกันอีกด้วย แต่ในสิ่งที่แตกต่างกันคือ ฟังก์ชั่น score() นั้นจะใช้หน่วยความจำที่น้อยกว่าแต่จะมีการประมวลผลที่ช้ากว่าฟังก์ชั่น score2() แต่ในฟังก์ชั่น score2() ก็จะมีการใช้หน่วยความจำที่มากกว่า score() อยู่มากนัก ซึ่งถ้าเราจะใช้ฟังก์ชั่น score2() ขอแนะนำว่า server ที่จะทำการประมวลผลนั้นให้มี RAM ติดตั้งอยู่อย่างน้อย 64 เมกะไบต์ โดยฟังก์ชั่น score() จะเรียกใช้ routine Hough1_1() ซึ่งเป็น routine ที่ใช้ในการทำ Hough Transforms ส่วนฟังก์ชั่น score2() จะเรียกใช้ routine Hough1() แทน ซึ่งนี้ก็เป็นส่วนหนึ่งที่แตกต่างกันอีกเช่นกัน

รูปแบบในการสร้างคือ

```
create function score2 (fing_t,fing_t)
returns double precision
external name
"$INFORMIXDIR/extend/Fingerprint.1.7.3p.41/Fingerprint.bld(score2)"
language c;
grant execute on function score2 (fing_t,fing_t) to public;
```

รูปที่ 9.7 แสดงการสร้างฟังก์ชั่น Score

เราจะต้องทำการให้สิทธิ์แก่ผู้อื่นให้ได้ใช้งานอีกด้วย และรายเอียดต่าง ๆ ก็สามารถหาดูได้จาก file objects.sql เช่นกัน



รูปที่ 9.8 แสดง Flow chart ของฟังก์ชั่นต่างๆ ที่ใช้งาน

9.6 การใช้งาน Fingerprint data type ที่สร้างขึ้น

การนำไปใช้งานสามารถทำได้เช่นเดียวกับ Built-in data type ตัวอื่น ๆ แต่อาจจะต่างกันตรงที่ในบางคำสั่งที่ไม่ได้มีการสร้างเอาไว้จะไม่สามารถถูกเรียกใช้งานได้ ตัวอย่างต่อไปนี้เป็นเพียงรูปแบบการใช้งานในเบื้องต้นเท่านั้น

```
CREATE TABLE Fingerprint (Fing_Col  fing_t);
INSERT INTO Fingerprint VALUES ('12 35 64 36 78 96 ... 25 68 94');
INSERT INTO Fingerprint VALUES  ('58 96 42 15 92 35 ... 45 89 63');
SELECT score2(f1.Fing_Col,f2.Fing_Col)
FROM Fingerprint f1,Fingerprint f2
ORDER BY 1
```

รูปที่ 9.9 แสดงการใช้งานฟังก์ชั่น Score

รูปแบบในการ input ผ่านทาง INSERT INTO Statement จะใช้ในลักษณะ

'X Y Zeta X Y Zeta ... X Y Zeta'

จะเป็น String ที่จะทำการผ่านไปให้ Support function Input/Output ทำการแปลงเป็น Internal format ให้ฐานข้อมูลได้ทำการบันทึก แต่ที่สำคัญคือ String ชุดนั้นจะต้องมีค่าไม่เกิน 256 Bytes (คือมองในแง่ของ String ที่มี Character ไม่เกิน 256 ตัว) ซึ่งถ้าเราต้องการจะ input เกินค่าที่เรากำหนดจะต้องสร้าง Module ในฝั่ง Client เพื่อทำการ Insert ลงในฐานข้อมูลโดยเฉพาะ และในการ Insert ในวิธีนี้จะเป็นการเรียกใช้ Support function Send/Receive ตามที่เราได้สร้างเอาไว้

ในการใช้งานจริงนั้นเราจะสร้าง Table ที่ชื่อ Fingerprint เพื่อทำการเก็บข้อมูลของลายนิ้วมือที่เราจะทำการ search หาเอาไว้โดยจะสร้างในลักษณะดังนี้

```
CREATE TABLE Fingerprint (
        Fid       integer,
        Picture   BYTE,
        Fing      fing_t
)
```

และจะยังมีการสร้าง Table เพื่อไว้ในการ Query อีกชื่อ for_query โดยมีรูปแบบดังนี้

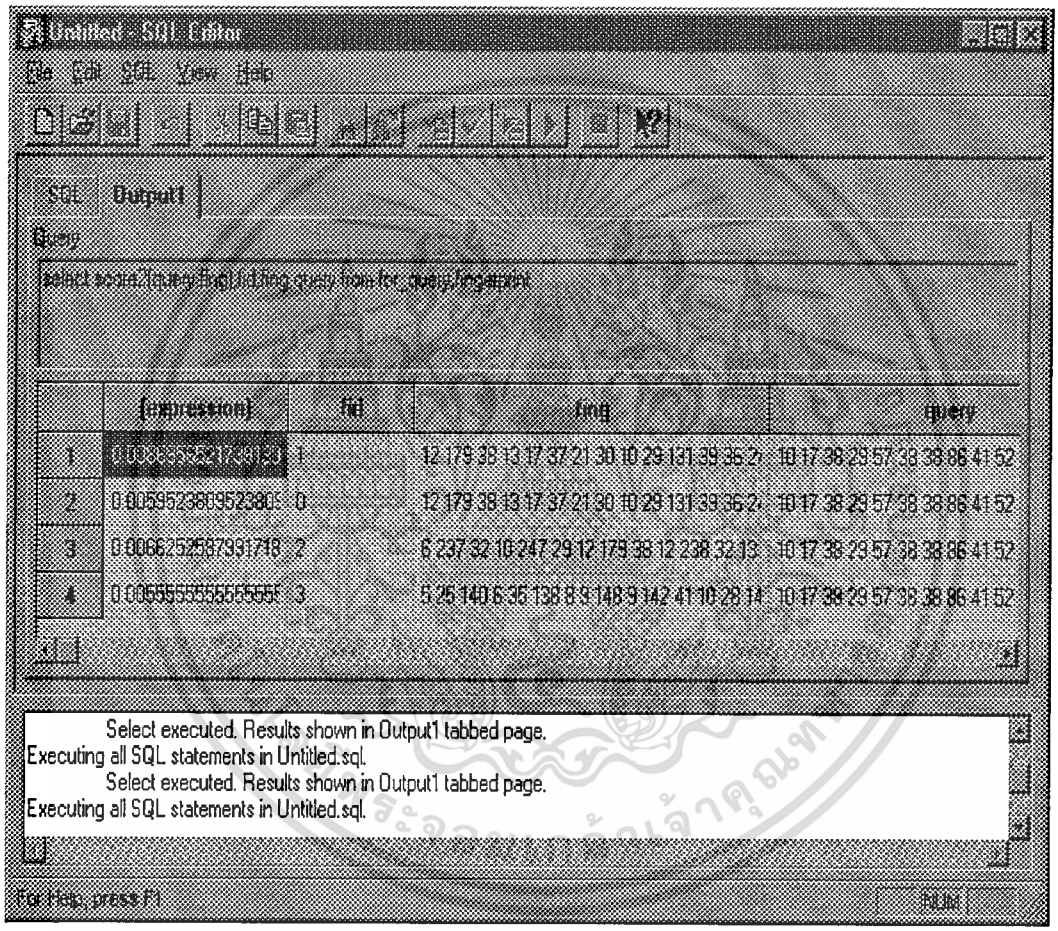
```
CREATE TABLE for_query (
        Qid       integer,
        Query     fing_t
)
```

Table for_query นี้จะมีไว้เพื่อการ search หาลายนิ้วมือที่ต้องการ โดยจะทำการ insert input รูปลายนิ้วมือที่ต้องการหาลงใน for_query table โดยมีข้อแม้ว่าจะต้องมีเพียง row เดียวเนื่องจากว่าข้อจำกัด

ของ tool ที่ใช้อยู่จึงต้องทำในลักษณะนี้ จากนั้นจึงเรียกใช้ฟังก์ชั่น score() หรือ score2() ได้ โดยลักษณะการเรียกใช้มีดังนี้

```
Select score2(query, fing), fid, picture
From for_query, fingerprint
Order by 1 desc
```

จากนั้นจึงนำผลลัพธ์ที่ได้มาแสดง



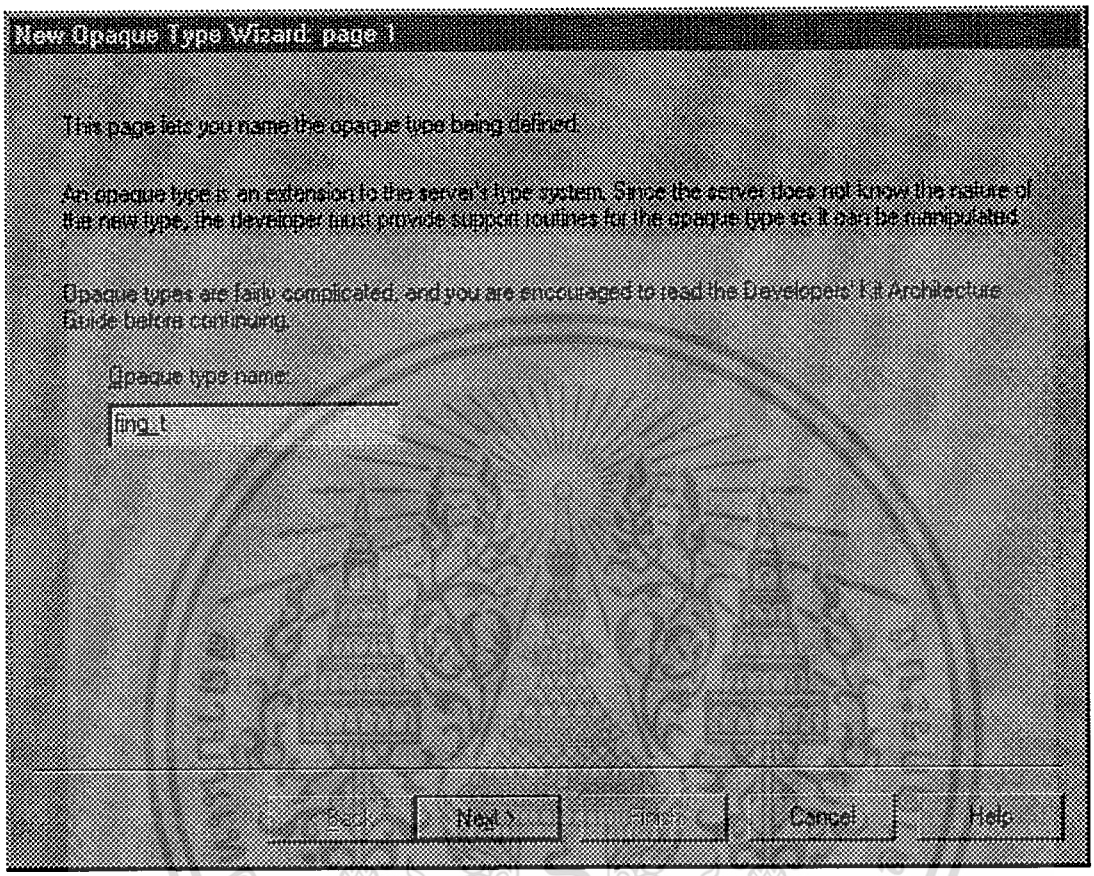
รูปที่ 9.10 แสดงถึงผลลัพธ์ที่ได้จากการ Query

9.7 การใช้ DataBlade เพื่อสร้าง Fingerprint Data Type

การใช้ DataBlade Developer Kit ในการสร้างจะทำการเรียกใช้โปรแกรม BladeSmith, BladePack, BladeManager ตามลำดับ โดยในที่นี้จะแสดงถึงการเรียกใช้งานโปรแกรม BladeSmith เพื่อสร้าง Fingerprint Data Type ส่วนการเรียกใช้โปรแกรม BladePack ซึ่งจะมีไว้เพื่อการ Copy files ต่าง ๆ ที่จำเป็นในการ Register ลงในฐานข้อมูล และโปรแกรม BladeManager ที่มีไว้เพื่อการ register

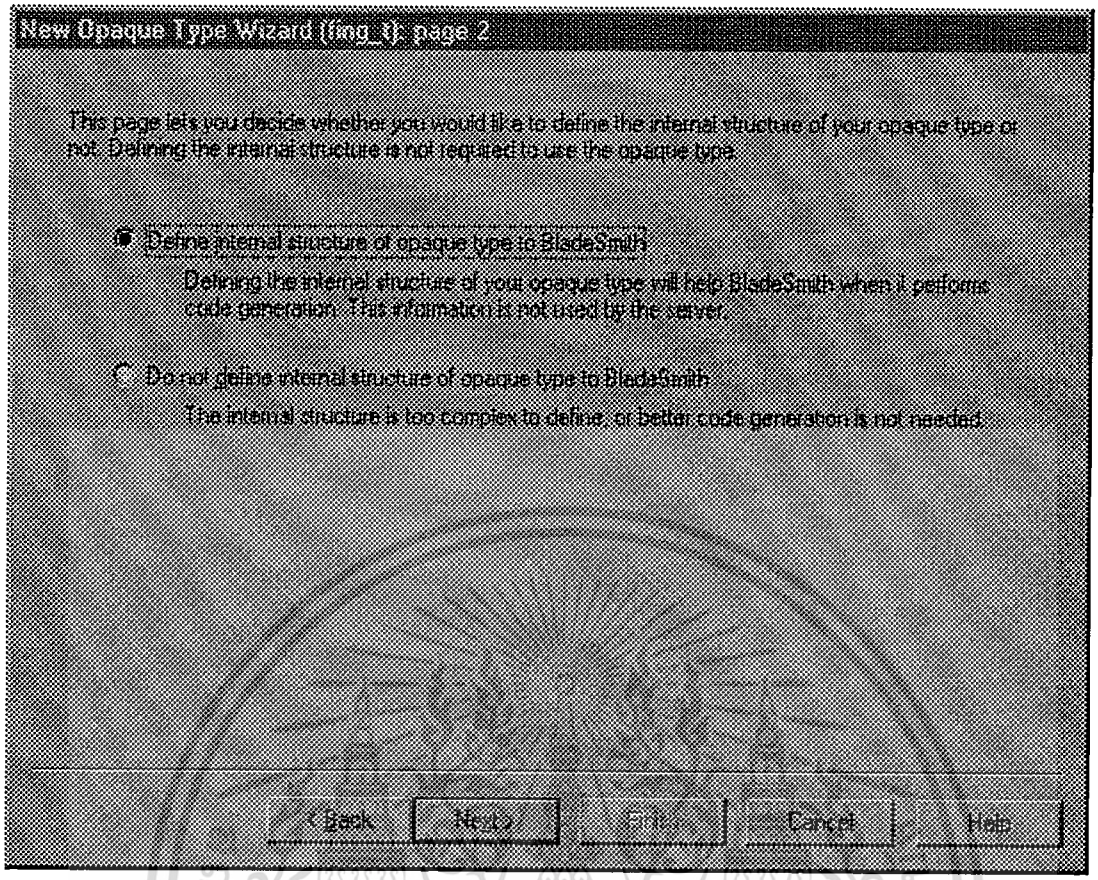
DataBlade Module ลงในฐานข้อมูล จะไม่ขอกล่าวถึง ซึ่งจริง ๆ แล้วสามารถหาอ่านจากคู่มือของ Informix ได้

ลำดับขั้นตอนของการใช้งานของ BladeSmith เพื่อสร้าง Fingerprint Data Type มีดังนี้



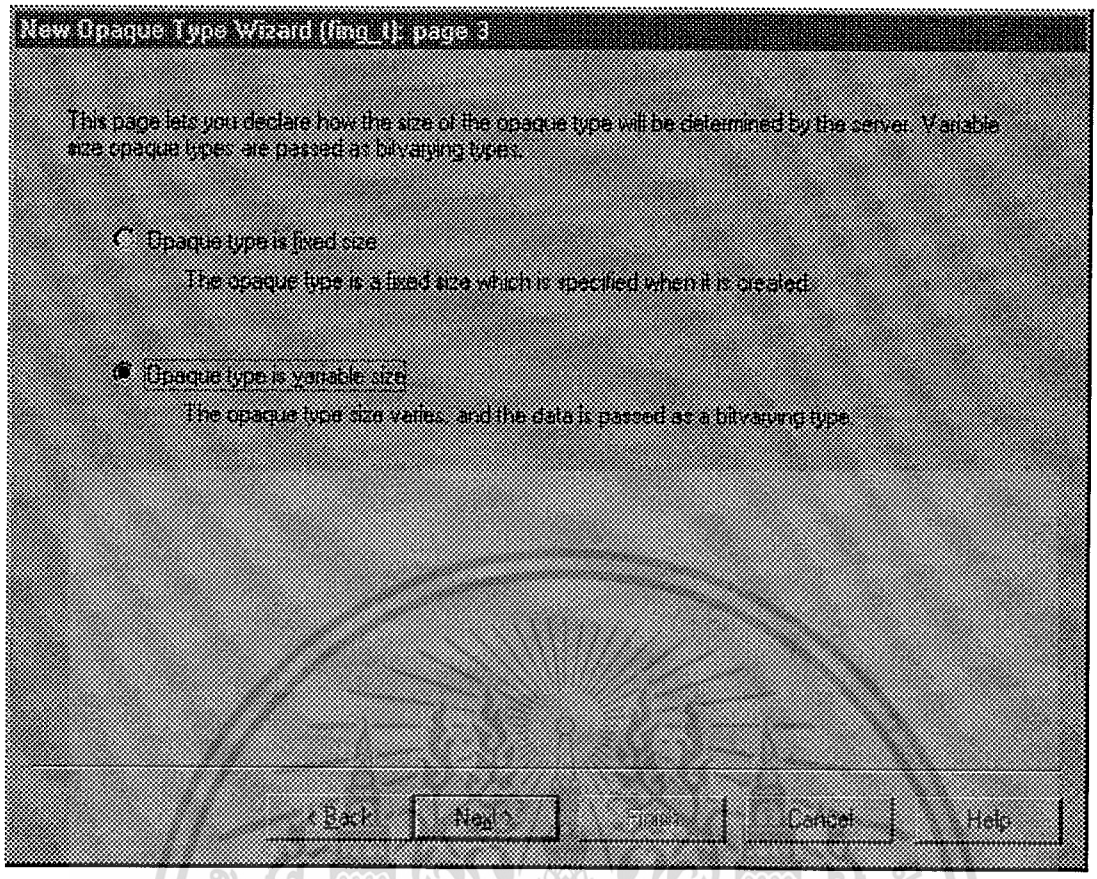
รูปที่ 9.11 แสดงหน้าแรกของการสร้าง fing_t

ในขั้นตอนแรกจะเป็นการกำหนดชื่อของ Fingerprint Data Type โดยเราจะกำหนดให้เป็นชื่อ fing_t



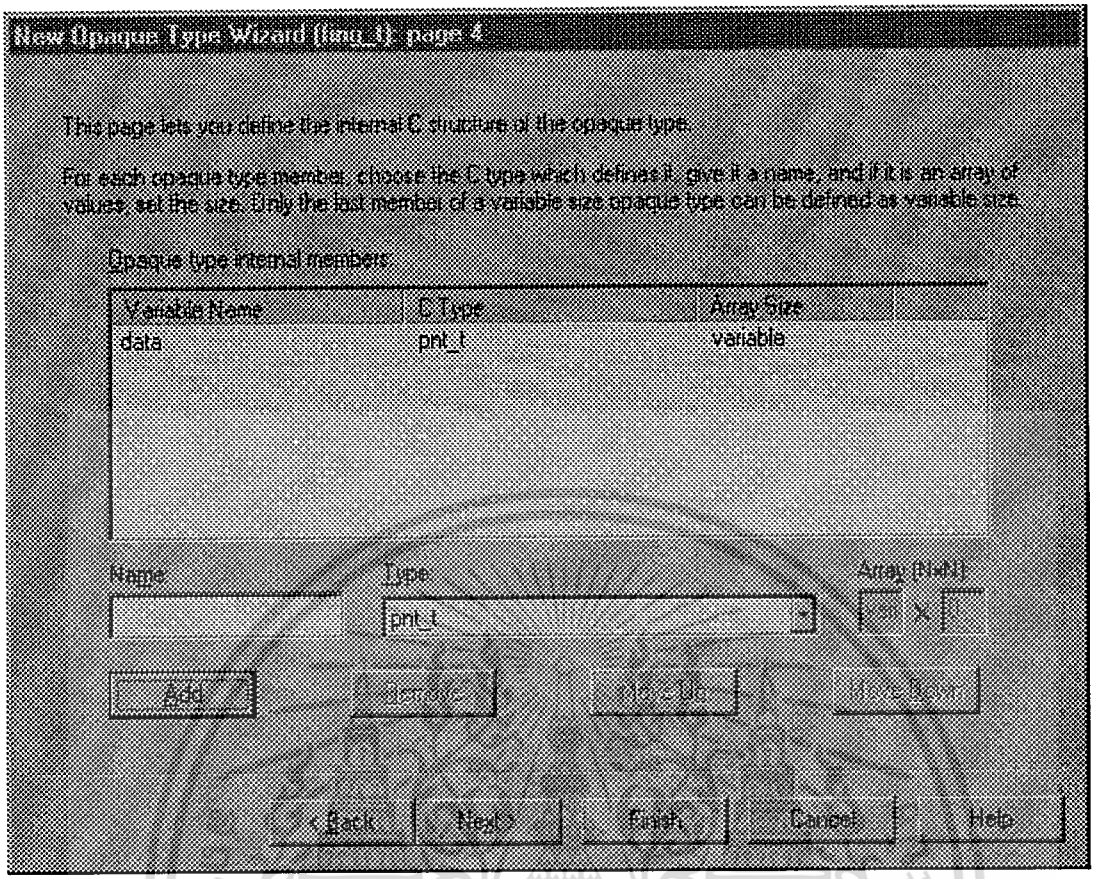
รูปที่ 9.12 เป็นการถามว่าจะมีการกำหนดโครงสร้างของ Opaque Type ที่จะทำการสร้างขึ้นหรือไม่

ในขั้นตอนที่สองจะเป็นการถามว่าจะมีการกำหนดโครงสร้างภายใน Opaque Type หรือไม่



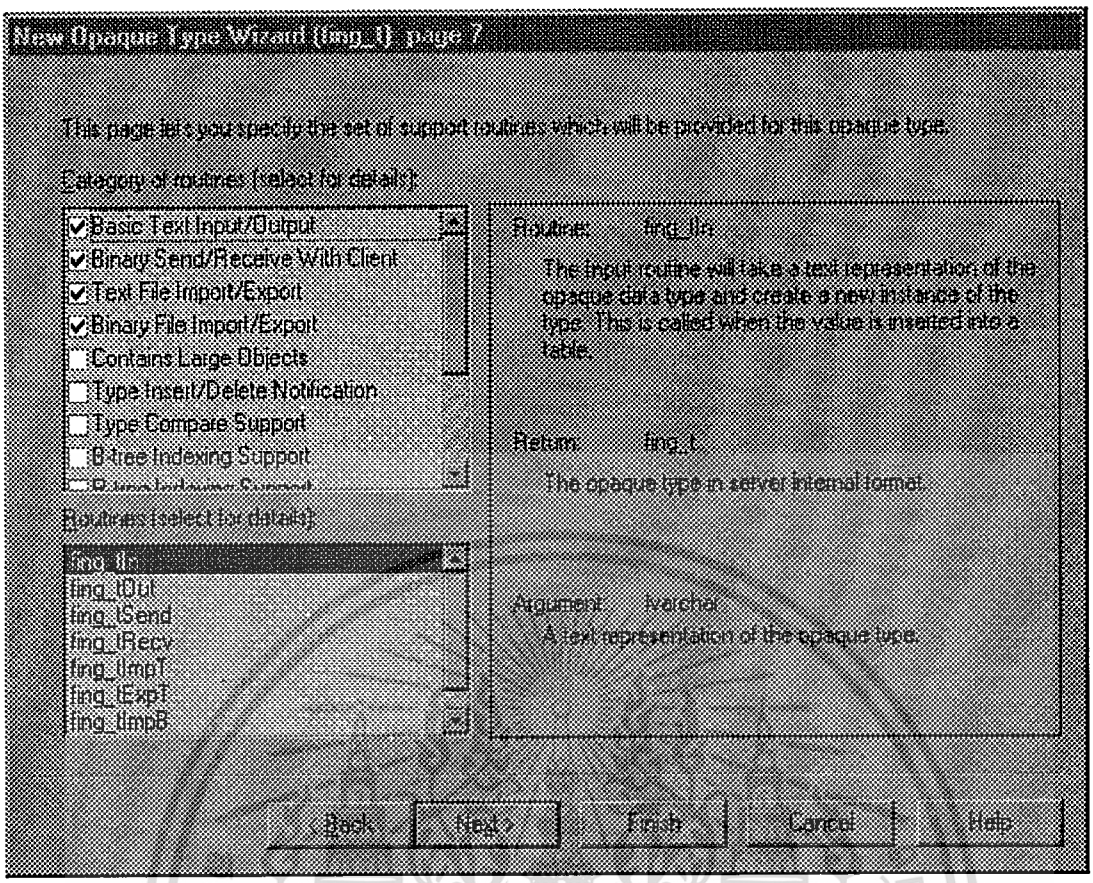
รูปที่ 9.13 การกำหนดรูปแบบของ Opaque Type

ในขั้นตอนที่สามจะเป็นการกำหนดว่า Opaque Type ที่เราจะทำการสร้างขึ้นมีชนิดเป็น Varying-Length Opaque Type



รูปที่ 9.14 กำหนดว่าโครงสร้างภายในจะมีชนิดเป็นอะไรบ้าง

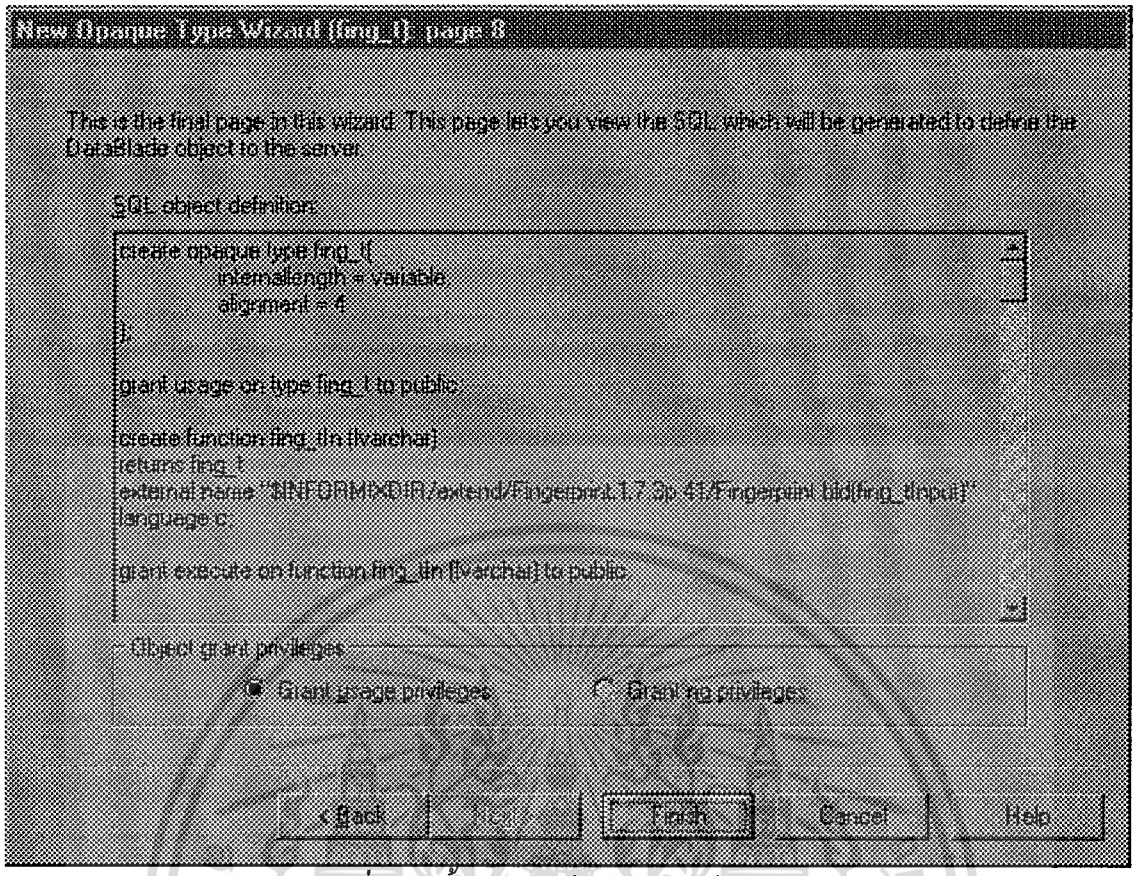
ในขั้นตอนนี้จะเป็นการกำหนดว่าจะมีโครงสร้างภายในเป็นชนิดอะไรบ้าง ซึ่งตัว BladeSmith เองจะเป็นตัวสร้าง code ขึ้นมาเองโดยอัตโนมัติ โดยที่เราไม่ต้องเข้าไปแก้ไขเลย



รูปที่ 9.15 เป็นการกำหนด Support Function ที่จะทำการใช้กับ fing_t ที่เราสร้างขึ้น

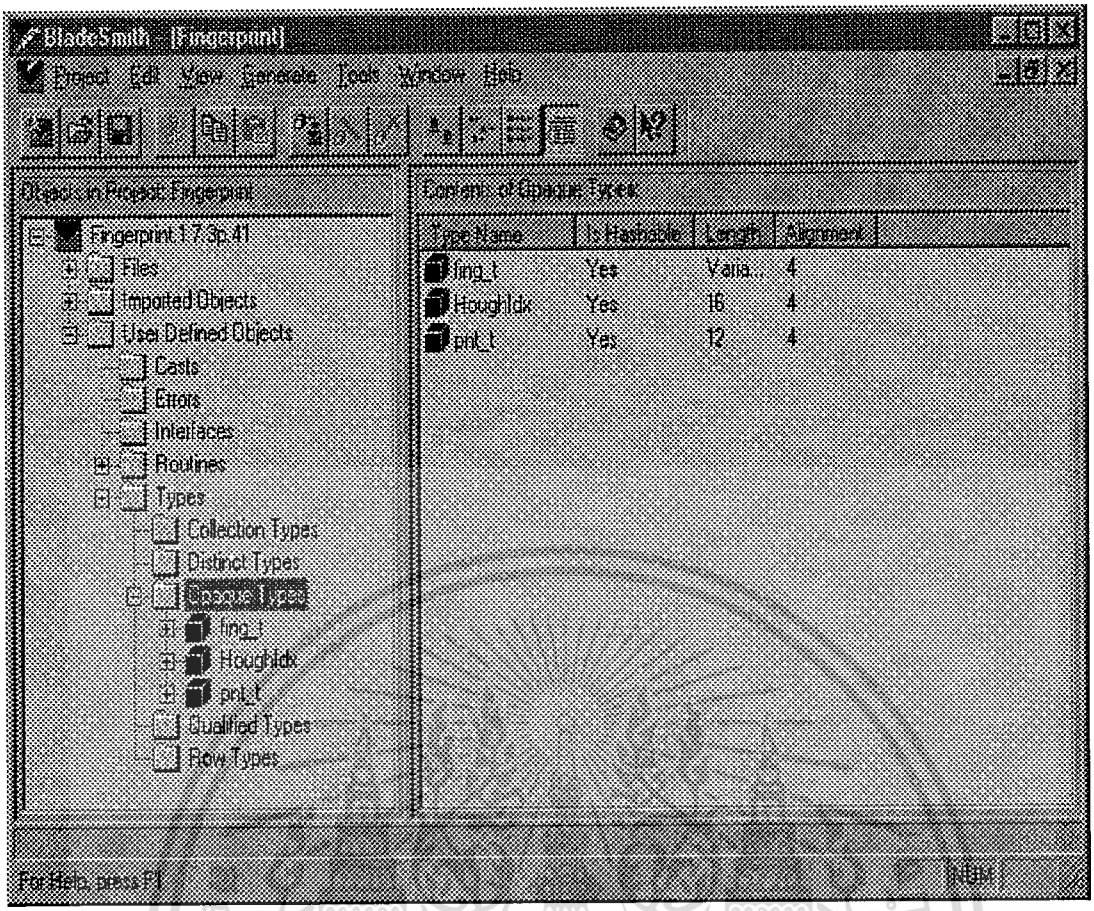
ขั้นตอนของการกำหนด Support Function จะทำการกำหนดในฟังก์ชั่นต่าง ๆ ที่เราจำเป็นต้องมี และอาจจะต้องมีการใช้งานต่อไปในอนาคต ซึ่งตัว BladeSmith เองจะทำการสร้าง Code ให้เองโดยอัตโนมัติ แต่ในบางครั้งจะทำการสร้างเฉพาะโครงร่างเอาไว้เพื่อให้เราสามารถเพิ่มเติมเองทีหลังก็ได้ ซึ่งในลักษณะนี้จะมี field ภายในโครงสร้างเป็นชนิดข้อมูลของ large object

การสร้างนี้ BladeSmith จะกำหนดพารามิเตอร์ต่าง ๆ ให้เองโดยอัตโนมัติซึ่งเราจะต้องไม่เข้าไปแก้ไขในส่วนนี้เนื่องจากจะมีผลในตอน register ลงในฐานข้อมูล



รูปที่ 9.16 ขั้นตอนสุดท้ายของการสร้าง fing_t

ในขั้นตอนสุดท้ายนี้จะแสดงถึงคำสั่ง SQL ต่าง ๆ ที่จะทำการ Register ลงในฐานข้อมูลซึ่งเราสามารถที่จะกำหนดถึงการ grant สิทธ์ให้ผู้อื่นใช้งานได้อีกด้วย



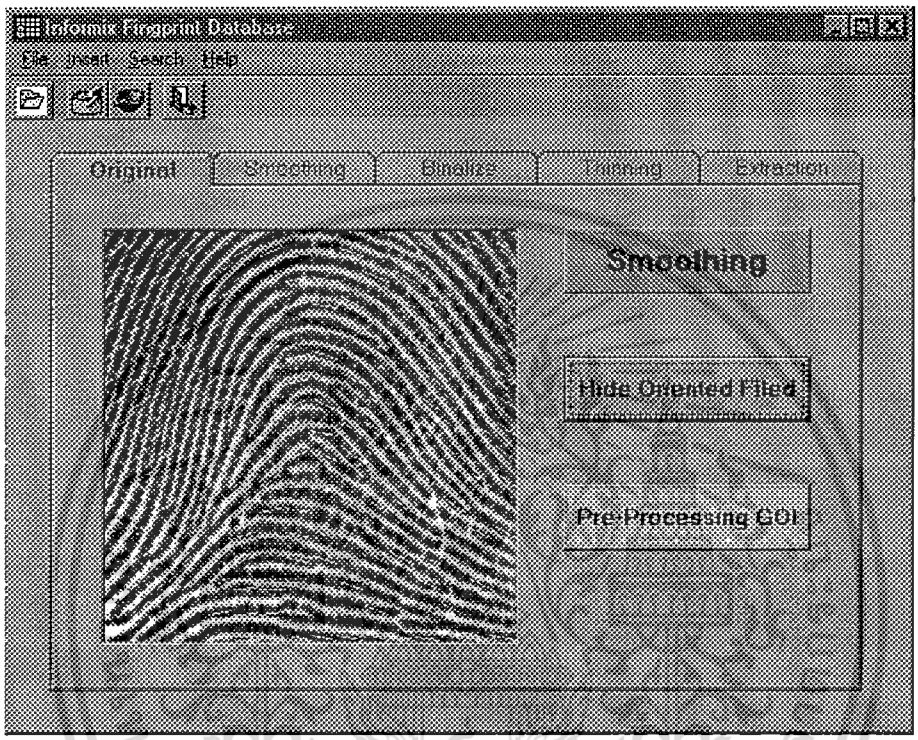
รูปที่ 9.17 หน้าจอของ BladeSmith ที่ทำการสร้าง Data Type ต่าง ๆ แล้ว

เมื่อเราทำการสร้างเสร็จหมดแล้วก็จะทำการกลับเข้ามาอยู่ในหน้าจอหลัก ซึ่งต่อจากนี้ไปเราสามารถสร้างตัว SQL , Code ต่าง ๆ ที่จะนำไป Register ลงในฐานข้อมูลและนำไปคอมไพล์ต่อได้อีก โดยเราจะเลือกเมนู Generate เพื่อสร้างสิ่งต่าง ๆ ตามที่ได้พูดมาแล้ว

บทที่ 10

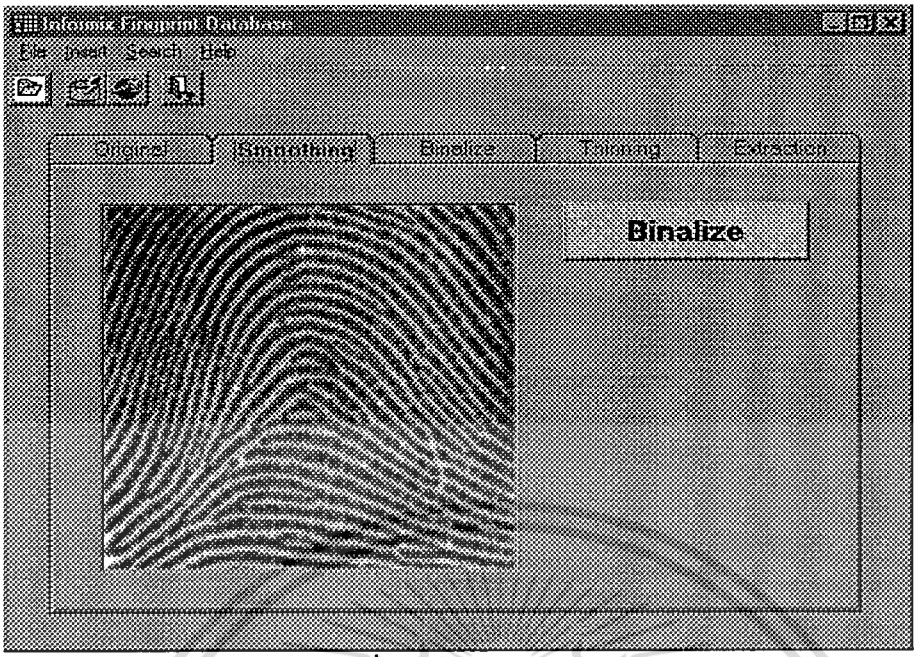
ผลการทดลอง

ผลการทดลองที่ได้เป็นดังรูปดังต่อไปนี้



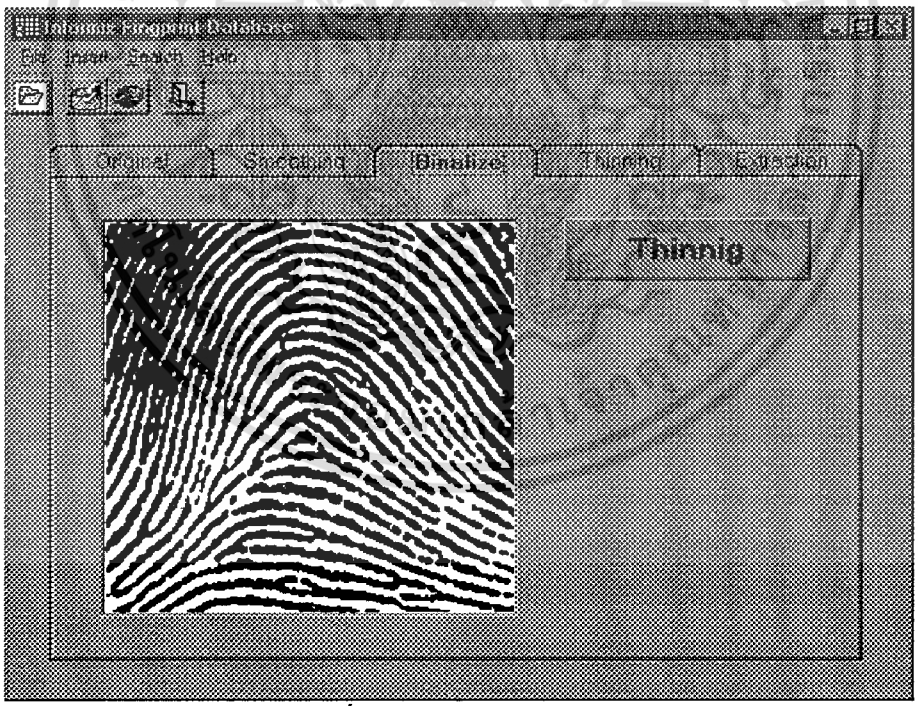
รูปที่ 10.1 การทำ Oriented field

การ Oriented Field เพื่อหามุมของลายนิ้วมือ หรือหาทิศทางของลายนิ้วมือเพื่อนำไปใช้ในการ matching จากรูปที่ได้จะเป็นการทิศทางโดยรวมทั้งหมดของลายนิ้วมือ โดยวิธีในการหาจะทำการแบ่งภาพลายนิ้วมือออกเป็นบล็อก ๆ แล้วทำการหาทิศทางของในแต่ละบล็อก ซึ่งสูตรที่ใช้และขั้นตอนต่าง ๆ ได้อธิบายผ่านมาแล้ว เพราะฉะนั้นจะไม่ขอกล่าวซ้ำในที่นี้อีก ในขั้นตอนต่อไปจะเป็นการทำ smoothing ภาพ



รูปที่ 10.2 การทำ Smoothing ภาพ

หลังจากได้มีการทำ Oriented Field แล้วก็จะมาทำ smoothing ภาพเพื่อที่จะทำการลด noise ต่าง ๆ ที่เกิดขึ้นกับภาพ

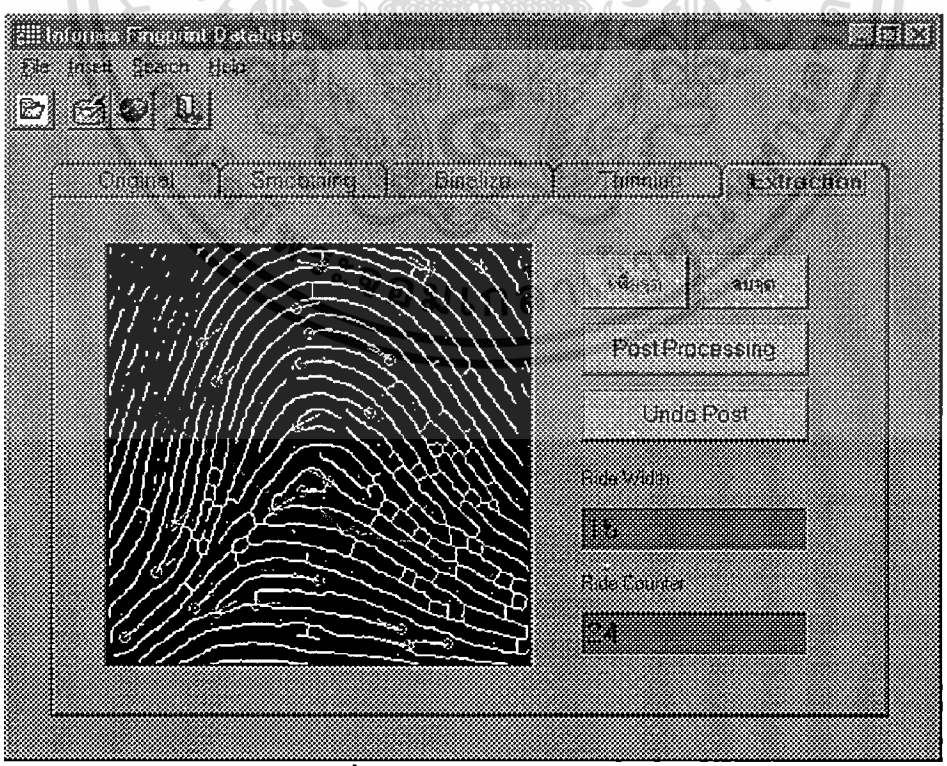


รูปที่ 10.3 การทำ Binarize ภาพ

การทำ Binarize คือการปรับระดับภาพจาก Gray level เป็นเพียงภาพขาวดำเท่านั้น

รูปที่ 10.4 การทำ Thinning ภาพ

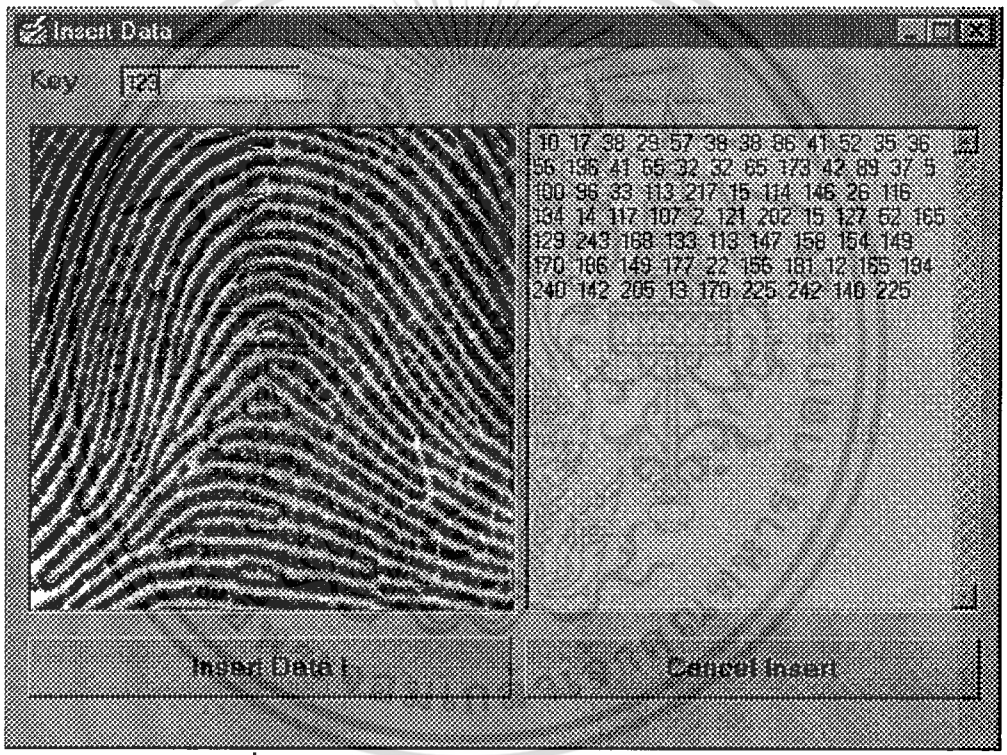
เมื่อทำการ Binarize ภาพเสร็จแล้วก็จะทำการหาโครงร่างของภาพ หรือการทำ Thinning ภาพนั่นเอง ซึ่งผลที่ได้จะเป็นเพียงเส้นของลายนิ้วมือที่มีความกว้างเพียงแค่ 1 พิกเซลเท่านั้นและผลลัพธ์ที่ได้ยังคงเป็นภาพขาวดำอยู่ด้วย



รูปที่ 10.5 การทำ Extraction ภาพ

เมื่อทำการ Thinning เสร็จแล้วก็จะทำการหาค่า feature ของลายนิ้วมือ นั่นก็คือการทำ Extract ภาพ ผลจากการ Extract ที่ได้จะเป็นค่า feature ของลายนิ้วมือในแต่ละจุด ในขั้นตอนนี้สามารถจะทำกระบวนการ Quality Checker ได้ด้วย ซึ่งก็ได้แก่การ Post-Processing ภาพเพื่อลด feature ต่าง ๆ ที่เป็นค่า error โดยในกระบวนการของเรานั้นจะทำการกำหนดความกว้างของ Ridge หรือกำหนดระยะห่างระหว่างเส้นลายนิ้วมือ โดยกำหนดเป็นค่าเฉลี่ยของทั้งลายนิ้วมือ และยังสามารถเพิ่มและลดจุดได้ตามต้องการ โดยอาศัยผู้ชำนาญที่เป็นมนุษย์มาทำการแก้ไข แต่มีข้อแม้ว่าจุดที่ทำการเพิ่มและลดจุดจะต้องเป็นจุดที่ได้เคยทำการ Extract เจอตั้งแต่ตอนแรกแล้วเท่านั้นเอง

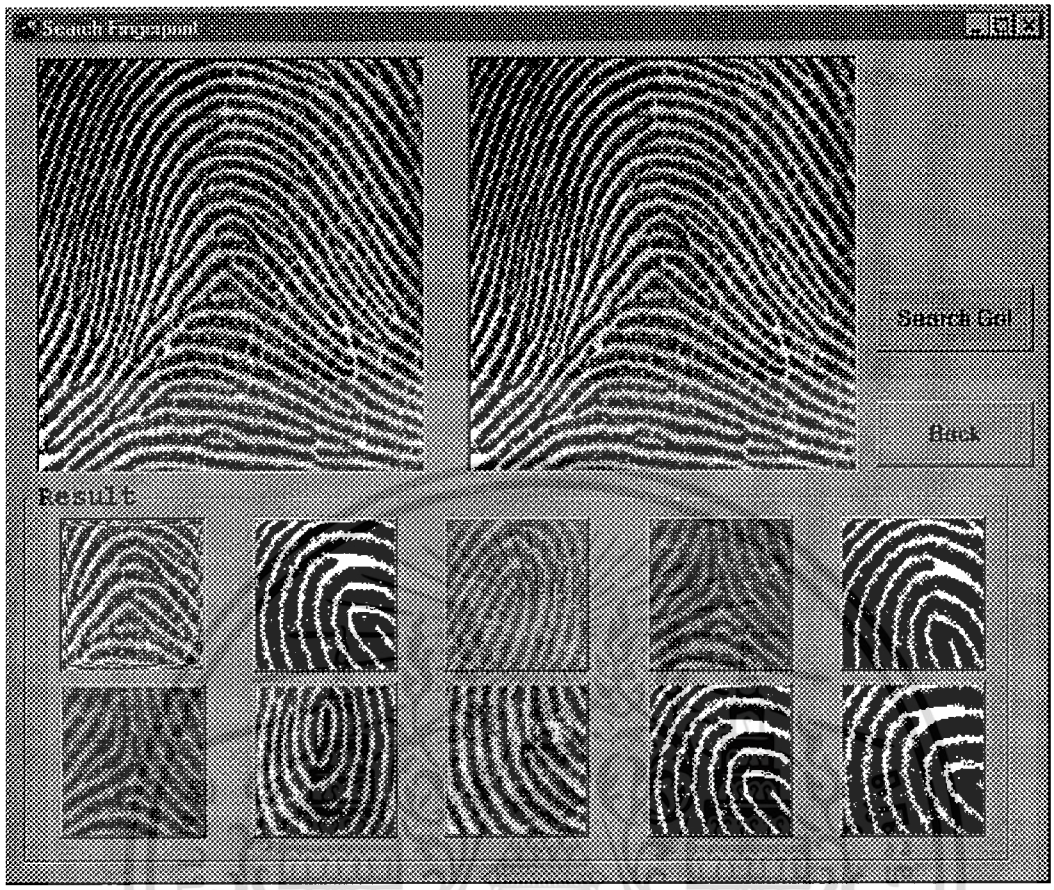
ผลที่ได้จากการ Extract สามารถนำไป insert ลงฐานข้อมูลหรือจะทำการ search หาลายนิ้วมือที่ต้องการได้



รูปที่ 10.6 แสดงการ Insert ข้อมูลลงในฐานข้อมูล

ในหน้าจอของการ insert จะต้องมีการกำหนด key เอาไว้ด้วยเพื่อที่จะนำค่า key นี้ไปใช้ในการ join กับตารางอื่น ๆ เพื่อหารายละเอียดเพิ่มเติมต่อไปได้อีก

เราสามารถตรวจลายละเอียดก่อนที่จะทำการ insert ได้ ผลลัพธ์ที่ได้จากการ Extract จะเป็นค่า integer ที่แสดงไว้ในหน้าต่างฝั่งขวามือ



รูปที่ 10.7 ผลจากการ Search ที่ได้จากฐานข้อมูล

ผลลัพธ์จากการ search คือ 10 อันดับแรกที่มีค่าที่ดีที่สุด โดยจะแสดงใน option button ที่อยู่ด้านล่างของ form เราสามารถจะ click เลือกลงไปได้เลย รูปลายนิ้วมือที่เป็น Result ก็จะปรากฏอยู่ในกรอบภาพฝั่งขวาบนของ form

บทสรุป

จากการทดลองและจากการค้นคว้าของทางคณะผู้จัดทำ ทำให้ได้เรียนรู้ถึงการใช้งานของ Object ที่อยู่ในฐานข้อมูลเชิงวัตถุสัมพันธ์ และรู้ถึงข้อดีข้อเสียของการใช้งานในแบบ Object โดยข้อดีก็คือมีความยืดหยุ่นสูงในการใช้งาน เนื่องจากว่าเป็นชนิดข้อมูลที่สามารถกำหนดขึ้นใช้งานเองได้ แต่ว่าข้อเสียที่ได้คือ หา tools ต่าง ๆ ที่จะนำมาใช้งานด้วยนั้นยาก เพราะว่าเป็นชนิดของข้อมูลที่ทำการสร้างขึ้นตามความต้องการของผู้ใช้เอง ซึ่งจะทำให้ tools ทั่วไปไม่สามารถที่จะเข้าใจชนิดของข้อมูลนั้นได้ จึงนับเป็นข้อเสียอย่างหนึ่งที่ทำให้การพัฒนาได้ค่อนข้างจะลำบาก และล่าช้า

ประการสุดท้ายคือ ได้รับรู้ถึงกระบวนการ การทำงานของระบบฐานข้อมูลลายนิ้วมือเริ่มตั้งแต่กระบวนการอินพุทของภาพลายนิ้วมือ ทำ Image Processing และหาค่าเอกลักษณ์ในภาพลายนิ้วมือนั้น จากนั้นจึงทำการ Insert ข้อมูล หรือจะทำการเรียกใช้ method เพื่อทำการ search หาลายนิ้วมือที่ต้องการได้

ปัญหาและอุปสรรค

ปัญหาที่เกิดขึ้นส่วนใหญ่จะมาจากการใช้ tools เพื่อการทำงานชิ้นนี้เนื่องจากว่ายังมี tools ที่สามารถรองรับเทคโนโลยีตรงนี้อยู่น้อย และประการสุดท้ายคือข้อมูลในสาขานี้ค่อนข้างหายาก และเนื้อหาต่าง ๆ โดยเฉพาะในส่วนของ Image Processing ค่อนข้างจะเข้าใจยากเนื่องจากพื้นฐานตรงส่วนนี้ทางคณะผู้จัดทำยังมีความรู้อยู่น้อยมากจึงทำได้เท่าที่หาข้อมูลและสามารถประยุกต์ใช้งานได้เท่านั้น

แนวทางในการพัฒนาต่อ

เราสามารถที่จะพัฒนาต่อในส่วนของ Image Processing ให้ดียิ่งขึ้นไปโดยสามารถที่จะ Recovery ลายนิ้วมือที่เสียหายขึ้นมาได้ หรือก็คือการทำ Enchantment Fingerprint Images นั่นเอง ซึ่งจะส่งผลถึงการ Extract เพื่อให้ค่า feature ที่ถูกต้องมากยิ่งขึ้น

ในส่วนของการ matching เราสามารถเพิ่มคุณสมบัติของการ Classification เพื่อการจัดกลุ่มลายนิ้วมือก่อนที่จะมีการ search จริงซึ่งจะทำให้มีการ search ที่เร็วขึ้น และยังนำค่าที่ได้บางส่วนของการ Classification มาใช้ในการ Post-Processing ได้อีกด้วย

ทางคณะผู้จัดทำหวังไว้ว่าโครงงานชิ้นนี้จะสามารถเป็นฐาน หรือเป็นประโยชน์เพื่อใช้ในการพัฒนาต่อไปยิ่งขึ้นไปอีก เพราะเนื่องจากในปัจจุบันระบบฐานข้อมูลลายนิ้วมือที่ซื้อขายกันในเชิงธุรกิจมีมูลค่าอย่างต่ำคือ 500 ล้านบาท (อ้างอิงจากสำนักงานตำรวจแห่งชาติ) ซึ่งมีเพียงคอมพิวเตอร์เพียงสองเครื่องนอกนั้นจะเป็นค่าซอฟท์แวร์ทั้งสิ้น

บรรณานุกรม

1) Norman J. Landis :"C for Pascal Programmers" 1989:ISBN 0-673-39917-6

2) Rafael C. Gonzalez, Richard E. Woods:"Digital Image processing":Addison-Wesly Publishing Company ISBN 0-201-60078-1

3) P.A. Maragos and R.W.Schafer : "Morphological skeleton representation and coding of binary image" , IEEE Trans.on Acoustics Speech and Signal Processing, Vol. ASSP-34 , No.5 October 1986 , pp. 1228-1244

4) B.K. Jang and R . T. Chin , "Analysis of thinning algorithms using mathematical morphology" , IEEE Trans.Patt.Anal.Mach.Intell., Vol. 12 , No.6 , June 1990, pp.541-551

5) A.D. Brink, "Thresholding of digital images using two dimensiopnal entropies" , Pattern Recognition , Vol. 25 , No. 8 , 1992 , pp. 803-808

6) Nikhil R. Pal , "On minimum cross entropy thresholding" , Pattern Recognition , Vol. 29 , No. 4 , 1996 , pp. 575-580

7) A. Jain, L. Hong, and R. Bolle. "On-line fingerprint verification" , IEEE Trans. Pattern Anal. And Machine Intell. , 19(4) : 302-314, 1997.

8) M. Kawagoe and A. Tojo. "Fingerprint pattern classification" Pattern Recognition , 17(3) : 295-303, 1984.

9) L.Hong, A.K. Jain , Sharath Pankanti , and Ruud Bolle , "Fingerprint Enhancement" , to appear In Proc. 1st WACV, Sarasota, FL, Dec., 1996

10) Nalini K. Ratha , Kalle Karu, Shaoyun Chen, and Anil K. Jain " A Real-Time Matching System for Large Fingerprint Databases" IEEE Trans. Pattern Anal And Machine Intell , 18(8) , August 1996

18) ธันวา ศรีประโมง , " การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม " มหานครเทคโนโลยีมหานคร , 739 หน้า , 2538

19) Michael Halvorson, "Step by Step Microsoft Visual Basic 5" : A division of Microsoft Corperation ISBN 1-57231-435-4

20) INFORMIX – Universal Server Informix Guide to SQL : Syntax, Version 9.1

21) DataBlade Developers Kit User's Guide, Version 3.4

22) INFORMIX – Universal Server DataBlade API Programmer's Manual, Version 9.12

ภาคผนวก

```
/*
** Title:          Fingerprint.h
** SCCSid:         %W% %E% %U%
** CCid:           %W% %E% %U%
** Author:
** Created:        Mar 06, 1999 09:14:08 AM
** Description:    Generated header file.
** Comments:       Generated for project Fingerprint.1.7.3p.41.
*/

/*
**      Special Note: This file should not be modified.
**      No merging is performed on this header file. It
**      is regenerated each time the project is written.
*/

#ifndef HDR_Fingerprint_H
#define HDR_Fingerprint_H

/*
** Configure tracing by setting TRACE_DEBUG_Fingerprint
** to 0 to completely disable tracing or 1 to enable
** tracing. This define may be set from the compiler
** command line by using the -DTRACE_DEBUG_Fingerprint=0 flag.
*/
#ifndef TRACE_DEBUG_Fingerprint
#define TRACE_DEBUG_Fingerprint 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;

/*
**      Large object file name mask.  '?' is a wild-card that
**      is filled in when a large object is written to disk.
*/
#define LO_FN_MASK    "????????.lo"

/*      Error messages
**
**      English versions of these error messages are automatically
**      added to the syserrors table as part of your DataBlade module
**      registration. If you do not like the default messages, you
**      can create new errors and change these defines to use your
**      new codes. You can not, however, change the text of the default
**      messages because they are shared by other DataBlade modules.
*/
#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_Fingerprint
#define DBDK_TRACE              (1 << 16)
#else
#define DBDK_TRACE              0
#endif

/*
** Print a message to the trace file and for  the user.
** N.B.: This macro uses Gen_Con.  Your  function  must
**      declare Gen_Con as MI_CONNECTION * and  either
**      open the connection or set it to NULL.
*/
#define DBDK_TRACE_ERROR( Caller, ErrNo, ErrLevel )     \
                Gen_Trace                       \
                (                       \
                  Gen_Con,              \
                  Caller,               \
                  __FILE__,             \
                  __LINE__,             \
                  ErrNo,                \
                  "Fingerprint",                        \
                  ErrLevel,             \
                  MI_SQL | DBDK_TRACE     \
                );
```

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

/*
** Print a message to the trace file.
** N.B.: This macro uses Gen_Con.  Your  function  must
**      declare Gen_Con as MI_CONNECTION * and  either
**      open the connection or set it to NULL.
*/
#define DBDK_TRACE_MSG( Caller, ErrNo, ErrLevel )      \
                Gen_Trace               \
                (                       \
                  Gen_Con,              \
                  Caller,               \
                  __FILE__,             \
                  __LINE__,             \
                  ErrNo,                \
                  "Fingerprint",                \
                  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 )

/*
**          Interval types.
```

```
*/
#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
);

/* BladeSmith 3.40.TXXXXX typedef pnt_t */
typedef struct
{
        mi_integer          X;
        mi_integer          Y;
        mi_integer          Zeta;
}
pnt_t;
/* Warning: Do not modify. pnt_t checksum: 0 */

/* BladeSmith 3.40.TXXXXX typedef fing_t */
typedef struct
{
        pnt_t               data[1];
}
fing_t;
/* Warning: Do not modify. fing_t checksum: 0 */

/* BladeSmith 3.40.TXXXXX typedef HoughIdx */
typedef struct
{
        mi_integer          S;
        mi_integer          Z;
        mi_integer          X;
        mi_integer          Y;
}
```

```
HoughIdx;
/* Warning: Do not modify. HoughIdx checksum: 0 */


#endif
```

```
/*****************************************************************
**
** Function name:
**
**        score
**
** Description:
**
** Special Comments:
**
**        Entrypoint for the SQL routine score (fing_t,fing_t) returns double precision.
**
** Parameters:
**
** Return value:
**
**        mi_double_precision *
**
** History:
**
**        Mar 06, 1999 - Generated by BladeSmith Version 3.40.TXXXXX.
**
** Identification:
**
**        Warning: Do not remove or modify this comment:
**          score FunctionId: bafa51d0-c0a0-11d2-806f-204c4f4f5020
**
*****************************************************************
*/
mi_double_precision *
score
(
mi_bitvarying *   query,
mi_bitvarying *   reference,
```

```
MI_FPARAM *  Gen_fparam  /* Standard info - see DBDK docs. */
)
{
        mi_double_precision *Gen_RetVal;  /* The return value. */
        MI_CONNECTION                   *Gen_Con;       /* The connection handle. */
        fing_t                          *R_set,*Q_set,*P_set;
        mi_integer                      R_nitems,Q_nitems,R,Q;
        mi_integer                      Xtop,Ytop,Xbottom,Ybottom;
        mi_integer                      QXtop,QYtop,QXbottom,QYbottom;
        mi_integer                      RXtop,RYtop,RXbottom,RYbottom;
        mi_integer                      Count,zeta_d,Qinbox,Rinbox;
        HoughIdx                        *Hidx;
        mi_double_precision     Sk,zeta_l,Xm,Yn,Px,Py,score_fing;

        /* 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( "score", ERRORMESG1, 10 );
        DBDK_TRACE_ENTER( "score" );
        Gen_RetVal = (mi_double_precision *)mi_alloc( sizeof( mi_double_precision * ) );
        if( Gen_RetVal == 0)
        {
                DBDK_TRACE_ERROR( "score", ERRORMESG2, 10 );
                return (mi_double_precision*)NULL;
        }
        /* Step 1: registration */
        Hidx = Hough1_1(query,reference,Gen_fparam);
        if(Hidx == (HoughIdx*)NULL)
        {
                mi_db_error_raise(Gen_Con,MI_EXCEPTION,"Hough Index Routine return
ERROR!");
                return (mi_double_precision *)NULL;
```

```
}
Sk = (Hidx->S*1.0)/10;
zeta_d = (Hidx->Z);
zeta_l = zeta_d*PI/180;
Xm = (Hidx->X)*1.0;
Yn = (Hidx->Y)*1.0;

/* Compute number of reference fingerprint (fingerprint in database) */
R_nitems = (mi_get_varlen((mi_bitvarying *)reference)-sizeof(fing_t)+ sizeof( R_set->data ))
        / sizeof( R_set->data );

/* compute number items of query fingerprint*/
Q_nitems = (mi_get_varlen((mi_bitvarying *)query)-sizeof(fing_t)+ sizeof( Q_set->data ))
        / sizeof( Q_set->data );

/* Step 1.1 : Compute result fingerprint  from Hough */

/* Point to where the return value is to be placed. */
P_set=(fing_t*)mi_get_vardata((mi_lvarchar*)reference);
R_set=(fing_t*)mi_zalloc(sizeof(fing_t)*R_nitems);
for(R=0;R<R_nitems;R++)
{
        Px=(P_set->data[R].X)*1.0;
        Py=(P_set->data[R].Y)*1.0;
        R_set->data[R].X = (mi_integer)(Sk*(Px*cos(zeta_l)+Py*sin(zeta_l))+Xm);

        R_set->data[R].Y = (mi_integer)(Sk*(Py*cos(zeta_l)-Px*sin(zeta_l))+Yn);
        R_set->data[R].Zeta = P_set->data[R].Zeta+zeta_d;
        /* Zeta is in 0<=Zeta<180 */
        if (R_set->data[R].Zeta >= 180)
                R_set->data[R].Zeta -= 180;
        else if (R_set->data[R].Zeta <0)
                R_set->data[R].Zeta +=180;
}
```

```
/* The Result from Hough transform,its may be negative in some items */

/* Step 1.2 :save data to fingerprint variable */

P_set=(fing_t*)mi_get_vardata((mi_lvarchar*)query);
Q_set=(fing_t*)mi_zalloc(Q_nitems*sizeof(fing_t));
for(Q=0;Q<Q_nitems;Q++)
{
        Q_set->data[Q].X=P_set->data[Q].X;
        Q_set->data[Q].Y=P_set->data[Q].Y;
        Q_set->data[Q].Zeta=P_set->data[Q].Zeta;
}

/* Step 2: find Bounding box in each fingerprint and intersect Bounding box*/
QXtop=QYtop=MAX_PIC;
QXbottom=QYbottom=0;
for(Q=0;Q<Q_nitems;Q++)
{
        if ((Q_set->data[Q].X<QXtop) && (Q_set->data[Q].X>=0)) QXtop = Q_set->data
[Q].X;
        if ((Q_set->data[Q].Y<QYtop) && (Q_set->data[Q].Y>=0)) QYtop = Q_set->data
[Q].Y;
        if ((Q_set->data[Q].X>QXbottom) && (Q_set->data[Q].X>=0))QXbottom = Q_set->
data[Q].X;
        if ((Q_set->data[Q].Y>QYbottom) && (Q_set->data[Q].Y>=0)) QYbottom = Q_set-
>data[Q].Y;
}
RXtop=RYtop=MAX_PIC;
RXbottom=RYbottom=0;
for(R=0;R<R_nitems;R++)
{
        if ((R_set->data[R].X<RXtop) && (R_set->data[R].X>=0)) RXtop = R_set->data
[R].X;
```

```
                if ((R_set->data[R].Y<RYtop) && (R_set->data[R].Y>=0)) RYtop = R_set->data
[R].Y;
                if ((R_set->data[R].X>RXbottom) && (R_set->data[R].X>=0))RXbottom = R_set->
data[R].X;
                if ((R_set->data[R].Y>RYbottom) && (R_set->data[R].Y>=0)) RYbottom = R_set->
data[R].Y;
        }
        /* find Boundind box */
        Xtop = (QXtop > RXtop) ? QXtop : RXtop;
        Ytop = (QYtop > RYtop) ? QYtop : RYtop;
        Xbottom = (QXbottom > RXbottom) ? RXbottom : QXbottom;
        Ybottom = (QYbottom > RYbottom) ? RYbottom : QYbottom;

        /* Step 2.1: mark features not in Bounding box */
        /* if Q features not in Bounding box ,mark it not to compare */
        /* if in Bounding Box increase Qinbox */
        Qinbox=0;
        for(Q=0;Q<Q_nitems;Q++)
                if ((Q_set->data[Q].X >= Xtop) && (Q_set->data[Q].X <= Xbottom) &&
                        (Q_set->data[Q].Y >= Ytop) && (Q_set->data[Q].Y <= Ybottom))
                {
                        Qinbox++;
                }
                else
                {
                        Q_set->data[Q].X = (mi_integer) MARK;
                        Q_set->data[Q].Y = (mi_integer) MARK;
                }

        /* if R features in Bounding box ,mark it not to compare */
        /* increase Rinbox */
        Rinbox =0;
        for(R=0;R<R_nitems;R++)
                if ((R_set->data[R].X >= Xtop) && (R_set->data[R].X <= Xbottom) &&
```

```
                (R_set->data[R].Y >= Ytop) && (R_set->data[R].Y <= Ybottom))
        {
                Rinbox++;
        }
        else
        {
                R_set->data[R].X = (mi_integer) MARK;
                R_set->data[R].Y = (mi_integer) MARK;
        }

/* Step 3: pair each feature of fingerprint and compute Score for each paired */
/* if 'Oriented filed' is in Zeta +5 and -5 angle degree */
Count = 0;
score_fing = 0.0;
if( (Qinbox != 0) && (Rinbox != 0) )
{
        for (Q=0;Q<Q_nitems;Q++)
                for (R=0;R<R_nitems;R++)
                        if(     (R_set->data[R].Zeta >=0) &&
                                (R_set->data[R].X >=0) &&
                                (R_set->data[R].Y >=0) &&
                                (Q_set->data[Q].Zeta >=0) &&
                                (Q_set->data[Q].X >=0) &&
                                (Q_set->data[Q].Y >=0) &&
                                (R_set->data[R].Zeta <= (Q_set->data[Q].Zeta+5)) &&
                                (R_set->data[R].Zeta >= (Q_set->data[Q].Zeta-5)) &&

                                (R_set->data[R].X >= (Q_set->data[Q].X-BOUND)) &&
                                (R_set->data[R].X <= (Q_set->data[Q].X+BOUND)) &&
                                (R_set->data[R].Y >= (Q_set->data[Q].Y-BOUND)) &&
                                (R_set->data[R].Y <= (Q_set->data[Q].Y+BOUND)) )
                                {
                                        Count++;
                                        /* mark not paired it again */
```

```
                                                    R_set->data[R].X = (mi_integer) MARK;
                                                    R_set->data[R].Y = (mi_integer) MARK;
                                                    R_set->data[R].Zeta = (mi_integer) MARK;
                                                }
                        score_fing = (Count*Count*1.0)/(Qinbox*Rinbox*1.0);
                }
                mi_free(R_set);
                mi_free(Q_set);
                *Gen_RetVal = score_fing*1.0;
                DBDK_TRACE_EXIT( "score" );
                return Gen_RetVal;
}
/* Warning: Do not modify. score checksum: 815827793  */

/*****************************************************************
**
** Function name:
**
**          score2
**
** Description:
**
** Special Comments:
**
**          Entrypoint for the SQL routine score2 (fing_t,fing_t) returns double precision.
**
** Parameters:
**
** Return value:
**
**          mi_double_precision *
**
** History:
**
```

```
**        Mar 06, 1999 - Generated by BladeSmith Version 3.40.TXXXXX.
**
** Identification:
**
**        Warning: Do not remove or modify this comment:
**          score2 FunctionId: 203c7d20-d36a-11d2-b042-204c4f4f5020
**
*****************************************************************
*/
mi_double_precision *
score2
(
mi_bitvarying *   Arg1,
mi_bitvarying *   Arg2,
MI_FPARAM *   Gen_fparam  /* Standard info - see DBDK docs. */
)
{
        mi_double_precision *Gen_RetVal;  /* The return value. */
        MI_CONNECTION                   *Gen_Con;       /* The connection handle. */
        fing_t                          *R_set,*Q_set,*P_set;
        mi_integer                      R_nitems,Q_nitems,R,Q;
        mi_integer                      Xtop,Ytop,Xbottom,Ybottom;
        mi_integer                      QXtop,QYtop,QXbottom,QYbottom;
        mi_integer                      RXtop,RYtop,RXbottom,RYbottom;
        mi_integer                      Count,zeta_d,Qinbox,Rinbox;
        HoughIdx                        *Hidx;
        mi_double_precision     Sk,zeta_l,Xm,Yn,Px,Py,score_fing;

        /* 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( "score2", ERRORMESG1, 10 );
```

```
DBDK_TRACE_ENTER( "score2" );
Gen_RetVal = (mi_double_precision *)mi_alloc( sizeof( mi_double_precision * ) );
if( Gen_RetVal == 0)
{
        DBDK_TRACE_ERROR( "score2", ERRORMESG2, 10 );
        return (mi_double_precision*)NULL;
}
/* Step 1: registration */
Hidx = Hough1(Arg1,Arg2,Gen_fparam);
if(Hidx == (HoughIdx*)NULL)
{
        mi_db_error_raise(Gen_Con,MI_EXCEPTION,"Hough Index Routine return
ERROR!");
        return (mi_double_precision *)NULL;
}
Sk = (Hidx->S*1.0)/10;
zeta_d = (Hidx->Z);
zeta_1 = zeta_d*PI/180;
Xm = (Hidx->X)*1.0;
Yn = (Hidx->Y)*1.0;

/* Compute number of Arg2 fingerprint (fingerprint in database) */
R_nitems = (mi_get_varlen((mi_bitvarying *)Arg2)-sizeof(fing_t)+ sizeof( R_set->data ))
        / sizeof( R_set->data );

/* compute number items of Arg1 fingerprint*/
Q_nitems = (mi_get_varlen((mi_bitvarying *)Arg1)-sizeof(fing_t)+ sizeof( Q_set->data ))
        / sizeof( Q_set->data );

/* Step 1.1 : Compute result fingerprint, from Hough */

/* Point to where the return value is to be placed. */
P_set=(fing_t*)mi_get_vardata((mi_lvarchar*)Arg2);
R_set=(fing_t*)mi_zalloc(sizeof(fing_t)*R_nitems);
```

```
for(R=0;R<R_nitems;R++)
{
        Px=(P_set->data[R].X)*1.0;
        Py=(P_set->data[R].Y)*1.0;
        R_set->data[R].X = (mi_integer)(Sk*(Px*cos(zeta_l)+Py*sin(zeta_l))+Xm);

        R_set->data[R].Y = (mi_integer)(Sk*(Py*cos(zeta_l)-Px*sin(zeta_l))+Yn);
        R_set->data[R].Zeta = P_set->data[R].Zeta+zeta_d;
        /* Zeta is in 0<=Zeta<180 */
        if (R_set->data[R].Zeta >= 180)
                R_set->data[R].Zeta -= 180;
        else if (R_set->data[R].Zeta <0)
                R_set->data[R].Zeta +=180;
}
/* The Result from Hough transform,its may be negative in some items */

/* Step 1.2 :save data to fingerprint variable */

P_set=(fing_t*)mi_get_vardata((mi_lvarchar*)Arg1);
Q_set=(fing_t*)mi_zalloc(Q_nitems*sizeof(fing_t));
for(Q=0;Q<Q_nitems;Q++)
{
        Q_set->data[Q].X=P_set->data[Q].X;
        Q_set->data[Q].Y=P_set->data[Q].Y;
        Q_set->data[Q].Zeta=P_set->data[Q].Zeta;
}

/* Step 2: find Bounding box in each fingerprint and intersect Bounding box*/
QXtop=QYtop=MAX_PIC;
QXbottom=QYbottom=0;
for(Q=0;Q<Q_nitems;Q++)
{
        if ((Q_set->data[Q].X<QXtop) && (Q_set->data[Q].X>=0)) QXtop = Q_set->data
[Q].X;
```

```
                if ((Q_set->data[Q].Y<QYtop) && (Q_set->data[Q].Y>=0)) QYtop = Q_set->data
[Q].Y;
                if ((Q_set->data[Q].X>QXbottom) && (Q_set->data[Q].X>=0))QXbottom = Q_set->
data[Q].X;
                if ((Q_set->data[Q].Y>QYbottom) && (Q_set->data[Q].Y>=0)) QYbottom = Q_set-
>data[Q].Y;
        }
        RXtop=RYtop=MAX_PIC;
        RXbottom=RYbottom=0;
        for(R=0;R<R_nitems;R++)
        {
                if ((R_set->data[R].X<RXtop) && (R_set->data[R].X>=0)) RXtop = R_set->data
[R].X;
                if ((R_set->data[R].Y<RYtop) && (R_set->data[R].Y>=0)) RYtop = R_set->data
[R].Y;
                if ((R_set->data[R].X>RXbottom) && (R_set->data[R].X>=0))RXbottom = R_set->
data[R].X;
                if ((R_set->data[R].Y>RYbottom) && (R_set->data[R].Y>=0)) RYbottom = R_set->
data[R].Y;
        }
        /* find Boundind box */
        Xtop = (QXtop > RXtop) ? QXtop : RXtop;
        Ytop = (QYtop > RYtop) ? QYtop : RYtop;
        Xbottom = (QXbottom > RXbottom) ? RXbottom : QXbottom;
        Ybottom = (QYbottom > RYbottom) ? RYbottom : QYbottom;

        /* Step 2.1: mark features not in Bounding box */
        /* if Q features not in Bounding box ,mark it not to compare */
        /* if in Bounding Box increase Qinbox */
        Qinbox=0;
        for(Q=0;Q<Q_nitems;Q++)
                if ((Q_set->data[Q].X >= Xtop) && (Q_set->data[Q].X <= Xbottom) &&
                        (Q_set->data[Q].Y >= Ytop) && (Q_set->data[Q].Y <= Ybottom))
                {
```

```
                Qinbox++;
        }
        else
        {
                Q_set->data[Q].X = (mi_integer) MARK;
                Q_set->data[Q].Y = (mi_integer) MARK;
        }

/* if R features in Bounding box ,mark it not to compare */
/* increase Rinbox */
Rinbox =0;
for(R=0;R<R_nitems;R++)
        if ((R_set->data[R].X >= Xtop) && (R_set->data[R].X <= Xbottom) &&
                (R_set->data[R].Y >= Ytop) && (R_set->data[R].Y <= Ybottom))
        {
                Rinbox++;
        }
        else
        {
                R_set->data[R].X = (mi_integer) MARK;
                R_set->data[R].Y = (mi_integer) MARK;
        }

/* Step 3: pair each feature of fingerprint and compute Score for each paired */
/* if 'Oriented filed' is in Zeta +5 and -5 angle degree */
Count = 0;
score_fing = 0.0;
if( (Qinbox != 0) && (Rinbox != 0) )
{
        for (Q=0;Q<Q_nitems;Q++)
                for (R=0;R<R_nitems;R++)
                        if(     (R_set->data[R].Zeta >=0) &&
                                (R_set->data[R].X >=0) &&
                                (R_set->data[R].Y >=0) &&
```

```
                                        (Q_set->data[Q].Zeta >=0) &&
                                        (Q_set->data[Q].X >=0) &&
                                        (Q_set->data[Q].Y >=0) &&
                                        (R_set->data[R].Zeta <= (Q_set->data[Q].Zeta+5)) &&
                                        (R_set->data[R].Zeta >= (Q_set->data[Q].Zeta-5)) &&

                                        (R_set->data[R].X >= (Q_set->data[Q].X-BOUND)) &&
                                        (R_set->data[R].X <= (Q_set->data[Q].X+BOUND)) &&
                                        (R_set->data[R].Y >= (Q_set->data[Q].Y-BOUND)) &&
                                        (R_set->data[R].Y <= (Q_set->data[Q].Y+BOUND)) )
                                        {
                                                Count++;
                                                /* mark not paired it again */
                                                R_set->data[R].X = (mi_integer) MARK;
                                                R_set->data[R].Y = (mi_integer) MARK;
                                                R_set->data[R].Zeta = (mi_integer) MARK;
                                        }
                        score_fing = (Count*Count*1.0)/(Qinbox*Rinbox*1.0);
                }
                mi_free(R_set);
                mi_free(Q_set);
                *Gen_RetVal = score_fing*1.0;
                DBDK_TRACE_EXIT( "score" );
                return Gen_RetVal;
}
/* Warning: Do not modify. score2 checksum: 716255067  */
/*****************************************************************
**
** Function name:
**
**          Hough1
**
** Description:
**
```

```
** Special Comments:
**
**        Entrypoint for the SQL routine Hough1 (fing_t,fing_t) returns HoughIdx.
**
** Parameters:
**
** Return value:
**
**        HoughIdx *
**
** History:
**
**        Mar 06, 1999 - Generated by BladeSmith Version 3.40.TXXXXX.
**
** Identification:
**
**        Warning: Do not remove or modify this comment:
**          Hough1 FunctionId: 23468d10-cdf5-11d2-b02a-204c4f4f5020
**
*****************************************************************
*/
HoughIdx *
Hough1
(
mi_bitvarying *   Arg1,
mi_bitvarying *   Arg2,
MI_FPARAM *   Gen_fparam  /* Standard info - see DBDK docs. */
)
{
        MI_CONNECTION * Gen_Con;        /* The connection handle.          */
        HoughIdx *     Get_RetVal;      /* The return value.               */
        fing_t                  *Pset,*Qset;
        mi_int1                 ****A;                          /* array A[Sk][zeat_l][Xm]
[Yn] */
```

```
mi_integer              S,zeta_d,Count,Px,Py,Qx,Qy,Xm,Yn;
mi_integer              Q,P,Q_nitems,P_nitems;
mi_boolean              ZZ;
mi_double_precision                     zeta_l,Sk,CosZ,SinZ;



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

/* Verify that the connection has been established. */
if( Gen_Con == 0 )
{
        /*
        ** Opening the current connection has failed
        ** so issue the following message and quit.
        **
        **      "Connection has failed in Hough1."
        */
        DBDK_TRACE_ERROR( "Hough1", ERRORMESG1, 10 );

        /* not reached */
}

/*
** Write to the trace file indicating
** that Hough1 has been called.
*/
DBDK_TRACE_ENTER( "Hough1" );
/* Compute the number of variable length query items*/
Q_nitems =
        (mi_get_varlen( (mi_bitvarying *)Arg1 ) - sizeof( fing_t ) + sizeof( Qset->data ))
        / sizeof( Qset->data );
/* Compute the number of variable length reference items*/
P_nitems =
```

```
                (mi_get_varlen( (mi_bitvarying *)Arg2 ) - sizeof( fing_t ) + sizeof( Pset->data ))
                / sizeof( Pset->data );
        /* Get input data to Q set and P set */
        Qset = (fing_t*)mi_get_vardata(Arg1);
        Pset = (fing_t*)mi_get_vardata(Arg2);
        /* Allocate */
        A=(mi_int1****)mi_fp_funcstate(Gen_fparam);
        if (A==(mi_int1****)NULL)
        {
                A=(mi_int1****)mi_dalloc(MAX_S*sizeof(mi_int1*),PER_COMMAND);
                if ( A == (mi_int1****)NULL )
                {
                        mi_db_error_raise(Gen_Con,MI_EXCEPTION,"memory failed in Sk axis");
                        return (HoughIdx*)NULL;
                }
                for(S=0;S<MAX_S;S++)
                {
        *(A+S)=(mi_int1***)mi_dalloc(MAX_Z*sizeof(mi_int1*),PER_COMMAND);
                        if( *(A+S) == (mi_int1***)NULL )
                        {
                                mi_db_error_raise(Gen_Con,MI_EXCEPTION,"memory failed in
Zeta axis");
                                return (HoughIdx*)NULL;
                        }
                }
                for(S=0;S<MAX_S;S++)
                        for(zeta_d=0;zeta_d<MAX_Z;zeta_d++)
                        {
                                *(*(A+S)+zeta_d)=(mi_int1**)mi_dalloc(MAX_X*sizeof
(mi_int1*),PER_COMMAND);
                                if( *(*(A+S)+zeta_d) == (mi_int1**)NULL )
                                {
```

```
                                        mi_db_error_raise(Gen_Con,MI_EXCEPTION,"memory
failed in Xm axis");
                                        return (HoughIdx*)NULL;
                                }
                        }
                for(S=0;S<MAX_S;S++)
                        for(zeta_d=0;zeta_d<MAX_Z;zeta_d++)
                                for(Xm=0;Xm<MAX_X;Xm++)
                                {
                                        *(*(*(A+S)+zeta_d)+Xm)=(mi_int1*)mi_dalloc
(MAX_Y*sizeof(mi_int1),PER_COMMAND);
                                        if( *(*(*(A+S)+zeta_d)+Xm) == (mi_int1*)NULL )
                                        {

        mi_db_error_raise(Gen_Con,MI_EXCEPTION,"memory failed in Yn axis");
                                                return (HoughIdx*)NULL;
                                        }
                                }
                        mi_fp_setfuncstate(Gen_fparam,(void****)A);
        }
        /* Hough */
        /* Step 1: Clear array all */
        for(S=0;S<MAX_S;S++)
                for(zeta_d=0;zeta_d<MAX_Z;zeta_d++)
                        for(Xm=0;Xm<MAX_X;Xm++)
                                for(Yn=0;Yn<MAX_Y;Yn++)
                                        *(*(*(*(A+S)+zeta_d)+Xm)+Yn) = 0;
        /* Step2: Compute and add evidence */
        for(P=0;P<P_nitems;P++)
                for(Q=0;Q<Q_nitems;Q++)
                {
                        ZZ = MI_FALSE;
                        zeta_d=0;
                        while ((ZZ==MI_FALSE)&&(zeta_d<MAX_Z))
```

```
                {
                        Count = zeta_d-(mi_integer)(MAX_Z-1)/2; /* -45 .. 45*/
                        if(Pset->data[P].Zeta+Count==Qset->data[Q].Zeta)
                        {
                                ZZ=MI_TRUE;
                                zeta_l = Count*PI/180;          /* degree to radians */
                                CosZ=cos(zeta_l);
                                SinZ=sin(zeta_l);
                                Px = Pset->data[P].X;
                                Py = Pset->data[P].Y;
                                Qx = Qset->data[Q].X;
                                Qy = Qset->data[Q].Y;
                                for(S=0;S<MAX_S;S++)            /* 0.5 ... 1.5 */
                                {
                                        Sk = ((S*1.0)+(MAX_S-1)/2)/10;
                                        Xm = Qx-(mi_integer)(Sk*(Px*CosZ+Py*SinZ));
                                        Yn = Qy-(mi_integer)(Sk*(Py*CosZ-Px*SinZ));
                                        Xm += (mi_integer)(MAX_X/2);    /* -64 ... +63
*/
                                        Yn += (mi_integer)(MAX_Y/2);    /* -64 ... +63
*/
                                        if ((Xm>=0)&&(Xm<MAX_X)&&(Yn>=0)&&
(Yn<MAX_Y))
                                                (*(*(*(*(A+S)+zeta_d)+Xm)+Yn))++;
                                }
                        }
                        else zeta_d++;
                }
        }
        /* Step 3:find max evidence of array A */
        Get_RetVal = (HoughIdx*) mi_zalloc(sizeof(HoughIdx));
        if (Get_RetVal == (HoughIdx*)NULL)
        {
```

```
                mi_db_error_raise(Gen_Con,MI_EXCEPTION,"Hough memory failed in Result
allocate");
                return (HoughIdx*)NULL;
        }

        Count =0;
        for(S=0;S<MAX_S;S++)
                for(zeta_d=0;zeta_d<MAX_Z;zeta_d++)
                        for(Xm=0;Xm<MAX_X;Xm++)
                                for(Yn=0;Yn<MAX_Y;Yn++)
                                        if( *(*(*(*(A+S)+zeta_d)+Xm)+Yn) > Count)
                                        {
                                                Get_RetVal->S = S+(mi_integer)(MAX_S-1)/2;
                                                Get_RetVal->Z = zeta_d-(mi_integer)(MAX_Z-
1)/2;
                                                Get_RetVal->X = Xm-(mi_integer)(MAX_X/2);
                                                Get_RetVal->Y = Yn-(mi_integer)(MAX_Y/2);
                                                Count= *(*(*(*(A+S)+zeta_d)+Xm)+Yn);
                                        }

        /*
        ** Write to the trace file indicating
        ** that Hough1 has successfully exited.
        */
        DBDK_TRACE_EXIT( "Hough1" );

        return Get_RetVal;
}
/* Warning: Do not modify. Hough1 checksum: 566424054 */

/*****************************************************************
**
** Function name:
**
```

```
**        Hough1_1
**
** Description:
**
** Special Comments:
**
**        Entrypoint for the SQL routine Hough1_1 (fing_t,fing_t) returns HoughIdx.
**
** Parameters:
**
** Return value:
**
**        HoughIdx *
**
** History:
**
**        Mar 06, 1999 - Generated by BladeSmith Version 3.40.TXXXXX.
**
** Identification:
**
**        Warning: Do not remove or modify this comment:
**          Hough1_1 FunctionId: 42671e00-d055-11d2-b035-204c4f4f5020
**
*****************************************************************
*/
HoughIdx *
Hough1_1
(
mi_bitvarying *   Arg1,
mi_bitvarying *   Arg2,
MI_FPARAM *   Gen_fparam  /* Standard info - see DBDK docs. */
)
{
        MI_CONNECTION * Gen_Con;        /* The connection handle.        */
```

```
        HoughIdx *    Gen_RetVal;      /* The return value.              */
        mi_integer                Q_vl_nitems;          /* number items of query fingerprint*/
        mi_integer                P_vl_nitems;          /* number items of reference
fingerprint*/
        mi_int1          **A;                           /* keep evidence for Xm and Ym in
Hough Algorithm */
        mi_integer                S,zeta_d,minS,maxS,minZ,maxZ,Q,P,Xm,Yn,Count,Px,Py,Qx,Qy;
        mi_double_precision  zeta_l,Sk;
        fing_t                    *Q_set,*P_set;        /* Pointer to the input data */

        /* struct to keep static evidence at S,zeta_l,Xm,Yn */
        struct stat {
                mi_integer        x;
                mi_integer  y;
                mi_integer        evidence;
        } Ev[SCALE+1][ROTATE+1]; /* because of in C array reference is 0..n-1*/


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

        /* Verify that the connection has been established. */
        if( Gen_Con == 0 )
        {
                /*
                ** Opening the current connection has failed
                ** so issue the following message and quit.
                **
                **        "Connection has failed in Hough1_1."
                */
                DBDK_TRACE_ERROR( "Hough1_1", ERRORMESG1, 10 );

                /* not reached */
        }
```

```
/*
** Write to the trace file indicating
** that Hough1_1 has been called.
*/
DBDK_TRACE_ENTER( "Hough1_1" );
Q_vl_nitems =
        (mi_get_varlen( (mi_bitvarying *)Arg1 ) - sizeof( fing_t ) + sizeof( Q_set->data ))
        / sizeof( Q_set->data );
/* Compute the number of variable length reference intems*/
P_vl_nitems =
        (mi_get_varlen( (mi_bitvarying *)Arg2 ) - sizeof( fing_t ) + sizeof( P_set->data ))
        / sizeof( P_set->data );
/* Get input data to Q set and R set */
Q_set = (fing_t*)mi_get_vardata(Arg1);
P_set = (fing_t*)mi_get_vardata(Arg2);
/* Get address from fp_funcstate */
A=(mi_int1**)mi_fp_funcstate(Gen_fparam);

if (A==(mi_int1**) NULL)                    /* Not Allocate or first Call */
{
        A=(mi_int1**)mi_dalloc(MAX_X*sizeof(mi_int1*),PER_COMMAND);
        if (A==(mi_int1**)NULL)
        {
                mi_db_error_raise(Gen_Con,MI_EXCEPTION,"A in Xm axis has failed to
allocate!");
                return (mi_integer) NULL;
        }
        for (Xm=0;Xm<=MAX_X;Xm++)
        {

*(A+Xm)=(mi_int1*)mi_dalloc(MAX_Y*sizeof(mi_int1),PER_COMMAND);
                if (*(A+Xm)==(mi_int1*)NULL)
                {
```

```
                                        mi_db_error_raise(Gen_Con,MI_EXCEPTION,"A in Yn axis failed
to allocate!");
                                        return (mi_integer) NULL;
                                }
                        }
                        mi_fp_setfuncstate(Gen_fparam,(void**)A); /* save address to fp_funcstate */
                }
                /* Hough Procedure */
                /* Step 1:Initialize all variable */
                minS=(mi_integer)SCALE/2;
                maxS=(mi_integer)SCALE+minS;  /* Sk=S/10 = 0.5 ... 1.5  SCALE =10*/
                minZ=(mi_integer)(-1)*(ROTATE/2);
                maxZ=(mi_integer)(ROTATE/2);   /* Zeta = -90 ... 90 ROTATE=180*/
                /* Setp 2: Compute and add evidence */
                for(S=minS;S<=maxS;S++)
                {
                        for(zeta_d=minZ;zeta_d<=maxZ;zeta_d++)
                        {
                                for(Xm=0;Xm<MAX_X;Xm++)
                                        for(Yn=0;Yn<MAX_Y;Yn++)
                                                *(*(A+Xm)+Yn) = 0;      /*clear data in Xm,Yn index to
zero */
                                for(P=0;P<P_vl_nitems;P++)        /* (Rx,Ry,Alpha) in reference set */
                                        for(Q=0;Q<Q_vl_nitems;Q++) /* (Qx,Qy,Beta) in query set */
                                                if(P_set->data[P].Zeta+zeta_d==Q_set->data[Q].Zeta)  /*
Alpha+Zeta=Beta*/
                                                {
                                                        zeta_l=zeta_d*PI/180;                /* degree to
radians */
                                                        Px = P_set->data[P].X;
                                                        Py = P_set->data[P].Y;
                                                        Qx = Q_set->data[Q].X;
                                                        Qy = Q_set->data[Q].Y;
                                                        Sk = (S*1.0)/10;
```

```
                                        Xm = Qx-(mi_integer)(Sk*(Px*cos
(zeta_l)+Py*sin(zeta_l)));
                                        Yn = Qy-(mi_integer)(Sk*(Py*cos(zeta_l)-Px*sin
(zeta_l)));
                                        Xm += (mi_integer)(MAX_X/2);    /* -64 ... +63
*/
                                        Yn += (mi_integer)(MAX_Y/2);    /* -64 ... +63
*/
                                        if ((Xm>=0)&&(Xm<MAX_X)&&(Yn>=0)&&
(Yn<MAX_Y))
                                                (*(*(A+Xm)+Yn))++;
                                }
                        Ev[S-minS][zeta_d-minZ].evidence = 0;
                        /* find MAX evidence for Xm,Yn*/
                        for(Xm=0;Xm<MAX_X;Xm++)
                                for(Yn=0;Yn<MAX_Y;Yn++)
                                        if(*(*(A+Xm)+Yn) > Ev[S-minS][zeta_d-minZ].evidence)
                                        {
                                                Ev[S-minS][zeta_d-minZ].evidence = *(*
(A+Xm)+Yn);
                                                Ev[S-minS][zeta_d-minZ].x=Xm;
                                                Ev[S-minS][zeta_d-minZ].y=Yn;
                                        }

                }
        }
        /* Step 3:fine max evidence */
        Gen_RetVal = (HoughIdx*) mi_zalloc(sizeof(HoughIdx));
        if (Gen_RetVal == (HoughIdx*)NULL)
        {
                mi_db_error_raise(Gen_Con,MI_EXCEPTION,"Hough memory failed in Result
allocate");
                return (HoughIdx*)NULL;
        }
```

```
        /* find MAX evidence for S,Zeta_l */
        Count =0;
        for(S=minS;S<=maxS;S++)
                for(zeta_d=minZ;zeta_d<=maxZ;zeta_d++)
                        if(Ev[S-minS][zeta_d-minZ].evidence > Count)
                        {
                                Gen_RetVal->X = (Ev[S-minS][zeta_d-minZ].x-MAX_X/2);
                                Gen_RetVal->Y = (Ev[S-minS][zeta_d-minZ].y-MAX_Y/2);
                                Gen_RetVal->S = S;
                                Gen_RetVal->Z = zeta_d;
                                Count=Ev[S-minS][zeta_d-minZ].evidence;
                        }

        /*
        ** Write to the trace file indicating
        ** that Hough1_1 has successfully exited.
        */
        DBDK_TRACE_EXIT( "Hough1_1" );

        return Gen_RetVal;
}
/* Warning: Do not modify. Hough1_1 checksum: 170060365  */
```

```
' Class cImageProcessc
Option Explicit

Private Type SAFEARRAYBOUND
    cElements As Long
    lLbound As Long
End Type

Private Type SAFEARRAY1D
    cDims As Integer
    fFeatures As Integer
    cbElements As Long
    cLocks As Long
    pvData As Long
    Bounds(0 To 0) As SAFEARRAYBOUND
End Type

Private Type SAFEARRAY2D
    cDims As Integer
    fFeatures As Integer
    cbElements As Long
    cLocks As Long
    pvData As Long
    Bounds(0 To 1) As SAFEARRAYBOUND
End Type

Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As Any) As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (pDst As Any, pSrc As
Any, ByVal ByteLen As Long)
Private Type BITMAP
    bmType As Long
    bmWidth As Long
    bmHeight As Long
    bmWidthBytes As Long
```

```
    bmPlanes As Integer
    bmBitsPixel As Integer
    bmBits As Long
End Type

Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As
Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As
Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
Private Const SRCCOPY = &HCC0020 ' (DWORD) dest = source

Private Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Long,
ByVal nCount As Long, lpObject As Any) As Long
Private Declare Function timeGetTime Lib "winmm.dll" () As Long


Public Enum EFilterTypes
    eSmooth
    eBinary
    eThin
    eExtract
    eOriented
End Enum

Public Enum eFilterError
    eeFilterErrorBase = vbObjectError Or 1048 Or &H500
End Enum

Private m_eFilterType As EFilterTypes
Private hist(256) As Long
Private Const pi As Double = 3.14159265358979
Private Const L As Integer = 256
Private Const initial As Integer = 2
Private Const dynamics As Integer = 1
Private Const statics As Integer = 0
```

```
Public xMax, yMax As Long
Private curl_x(256, 256) As Long
Private curl_y(256, 256) As Long
Private curl_x_2(256, 256) As Long
Private curl_y_2(256, 256) As Long
Private zeta(16, 16) As Integer
Private extracted(256, 256) As Integer
Private Save_extract(256, 256) As Integer
Private thined(256, 256) As Integer
Private Const EP As Integer = 123
Private Const BP As Integer = 456
Public RideWidth As Integer

Public Property Let FilterType(ByVal eType As EFilterTypes)
  m_eFilterType = eType
End Property

' select the case of image processing
Public Function ProcessImage( _
  ByRef picImage As PictureBox) As Boolean

  Select Case m_eFilterType
  Case eSmooth
    ProcessImage = SmoothFilter(picImage)

  Case eThin
    ProcessImage = ThinFilter(picImage)

  Case eBinary
    ProcessImage = BinaryFilter(picImage)

  Case eExtract
    ProcessImage = ExtractFilter(picImage)
```

```
    Case eOriented
        ProcessImage = OrientedFilter(picImage)
    End Select
End Function

Public Function Feature(ByVal x As Integer, ByVal y As Integer) As Integer
    Feature = extracted(x, y)
End Function

Public Function OrientedField(ByVal x As Integer, ByVal y As Integer) As Integer
    OrientedField = zeta(x, y)
End Function

'---------- Gradiant image ----------
Private Sub gradiant(ByRef pict() As Byte)
Dim x, y As Integer
    For x = 1 To xMax - 1
        For y = 1 To yMax - 1
        'curl_y(x, y) = (pict(x + 1, y + 1) + (2 * pict(x + 1, y)) + pict(x + 1, y - 1) _
        '          - pict(x - 1, y + 1) - (2 * pict(x - 1, y)) - pict(x - 1, y - 1))
        'curl_x(x, y) = (pict(x + 1, y + 1) + (2 * pict(x, y + 1)) + pict(x - 1, y + 1) _
        '          - pict(x - 1, y - 1) - (2 * pict(x, y - 1)) - pict(x + 1, y - 1))
        curl_y(x, y) = pict(x - 1, y - 1) + (2 * pict(x, y - 1)) + pict(x + 1, y - 1) _
                    - pict(x + 1, y + 1) - (2 * pict(x, y + 1)) - pict(x - 1, y + 1)
        curl_x(x, y) = pict(x - 1, y + 1) + (2 * pict(x - 1, y)) + pict(x - 1, y - 1) _
                    - pict(x + 1, y + 1) - (2 * pict(x + 1, y)) - pict(x + 1, y - 1)
        Next y
    Next x
' use sobel operator 3*3 mask
End Sub

Private Sub gradiant2()
Dim i, j As Integer
For i = 0 To xMax
```

```
    For j = 0 To yMax
    curl_x_2(i, j) = curl_x(i + 1, j) - curl_x(i, j)
    curl_y_2(i, j) = curl_y(i, j + 1) - curl_y(i, j)
    Next j
  Next i
  End Sub

  Private Function block(ByVal x As Integer, ByVal y As Integer) As Integer
  Dim sum1, sum2, temp1, temp2 As Double
  Dim i, j As Integer

  sum1 = 0: sum2 = 0
    For i = (x - 1) * 16 To x * 16
      For j = (y - 1) * 16 To y * 16
        sum1 = sum1 + (2 * curl_x(i, j) * curl_y(i, j))
        sum2 = sum2 + (curl_x_2(i, j) - curl_y_2(i, j) ^ 2)
      Next j
    Next i

  'case divide by 0
  If sum2 <> 0 Then
    temp1 = 0.5 * Atn(sum1 / sum2)  ' -90 .... 90
    temp2 = temp1 * 180 / pi ' convert radiant to degree
    If temp2 < 0 Then
      block = 180 + Int(temp2) ' return integer degree value
    Else
      block = Int(temp2)
    End If
  Else
    block = 90
  End If
  End Function
  Private Sub orientation()
  Dim i, j As Integer
```

```
    For i = 1 To 16
        For j = 1 To 16
        zeta(i, j) = block(i, j) 'find x,y location * 8
        Next j
    Next i
End Sub

Public Function OrientedFilter(ByRef picImage As PictureBox) As Boolean
' these are used to address the pixel using matrices
Dim pict() As Byte
Dim sa As SAFEARRAY2D, bmp As BITMAP
Dim x As Long, y As Long
Dim i As Long, j As Long
Dim fraction As Long, sum As Long
    xMax = 0: yMax = 0
    ' get bitmap info
    GetObjectAPI picImage.Picture, Len(bmp), bmp 'dest

    ' exit if not a supported bitmap
    If bmp.bmBitsPixel <> 8 Then
        MsgBox " 8-bit bitmaps only", vbCritical
        Exit Function
    End If

    ' have the local matrix point to bitmap pixels
    With sa
        .cbElements = 1
        .cDims = 2
        .Bounds(0).lLbound = 0
        .Bounds(0).cElements = bmp.bmHeight
        .Bounds(1).lLbound = 0
        .Bounds(1).cElements = bmp.bmWidthBytes
        .pvData = bmp.bmBits
    End With
```

```
    CopyMemory ByVal VarPtrArray(pict), VarPtr(sa), 4
    yMax = UBound(pict, 2)
    xMax = UBound(pict, 1)

    Call gradiant(pict)
    Call gradiant2
    Call orientation

    CopyMemory ByVal VarPtrArray(pict), 0&, 4
    OrientedFilter = True
End Function

'---------- Use for Smooth image ----------
Private Function Top(ByVal y As Long) As Long
    If (y < 0) Then
        Top = 1
    Else
        Top = 0
    End If
End Function
Private Function bottom(ByVal y As Long) As Long
    If (y > yMax) Then
        bottom = 1
    Else
        bottom = 0
    End If
End Function
Private Function Right(ByVal x As Long) As Long
    If (x > xMax) Then
        Right = 1
    Else
        Right = 0
    End If
End Function
```

```
Private Function Left(ByVal x As Long) As Long
  If (x < 0) Then
    Left = 1
  Else
    Left = 0
  End If
End Function
Private Function X_sum(ByVal i As Integer, ByVal x As Long, ByVal y As Long, ByRef pict() As
Byte) As Long
Dim xx As Long
Select Case i
  Case 0
    X_sum = Val(pict(x, y))

  Case 1
  If (Right(x + 1)) Then        'R(x+1)=1
    X_sum = 255
  Else
    X_sum = Val(pict(x + 1, y))
  End If
  Case 2
  If (Right(x + 1) Or Top(y - 1)) Then
    X_sum = 255
  Else
    X_sum = Val(pict(x + 1, y - 1))
  End If
  Case 3
  If (Top(y - 1)) Then
    X_sum = 255
  Else
    X_sum = Val(pict(x, y - 1))
  End If
  Case 4
  If (Top(y - 1) Or Left(x - 1)) Then
```

```
        X_sum = 255
    Else
        X_sum = Val(pict(x - 1, y - 1))
    End If
    Case 5
    If (Left(x - 1)) Then
        X_sum = 255
    Else
        X_sum = Val(pict(x - 1, y))
    End If
    Case 6
    If (Left(x - 1) Or bottom(y + 1)) Then
        X_sum = 255
    Else
        X_sum = Val(pict(x - 1, y + 1))
    End If
    Case 7
    If (bottom(y + 1)) Then
        X_sum = 255
    Else
        X_sum = Val(pict(x, y + 1))
    End If
    Case 8
    If (bottom(y + 1) Or Right(x + 1)) Then
        X_sum = 255
    Else
        X_sum = Val(pict(x + 1, y + 1))
    End If
End Select

End Function

Public Function SmoothFilter( _
    ByRef picImage As PictureBox) As Boolean
```

```
' these are used to address the pixel using matrices
Dim pict() As Byte
Dim sa As SAFEARRAY2D, bmp As BITMAP
Dim x As Long, y As Long
Dim i As Long, j As Long
Dim fraction As Long, sum As Long
  xMax = 0: yMax = 0
  ' get bitmap info
  GetObjectAPI picImage.Picture, Len(bmp), bmp 'dest

  ' exit if not a supported bitmap
  If bmp.bmBitsPixel <> 8 Then
    MsgBox " 8-bit bitmaps only", vbCritical
    Exit Function
  End If

  ' have the local matrix point to bitmap pixels
  With sa
    .cbElements = 1
    .cDims = 2
    .Bounds(0).lLbound = 0
    .Bounds(0).cElements = bmp.bmHeight
    .Bounds(1).lLbound = 0
    .Bounds(1).cElements = bmp.bmWidthBytes
    .pvData = bmp.bmBits
  End With
  CopyMemory ByVal VarPtrArray(pict), VarPtr(sa), 4

  ' Do filter on pict into pict2

  yMax = UBound(pict, 2)
  xMax = UBound(pict, 1)
  For x = 0 To xMax
    For y = 0 To yMax
```

```
    sum = 0: fraction = 0
    sum = sum + X_sum(0, x, y, pict)
    sum = sum + X_sum(1, x, y, pict)
    sum = sum + X_sum(2, x, y, pict)
    sum = sum + X_sum(3, x, y, pict)
    sum = sum + X_sum(4, x, y, pict)
    sum = sum + X_sum(5, x, y, pict)
    sum = sum + X_sum(6, x, y, pict)
    sum = sum + X_sum(7, x, y, pict)
    sum = sum + X_sum(8, x, y, pict)

    fraction = sum \ 9
    pict(x, y) = Str(fraction)

    Next y
  Next x

  ' clear the temporary array descriptor
  ' without destroying the local temporary array
  CopyMemory ByVal VarPtrArray(pict), 0&, 4
  SmoothFilter = True

End Function

'---------- Use for Binary image ----------

Private Sub read_frequency(ByRef pict() As Byte, ByRef hist() As Long, ByVal L As Integer)
Dim x, y As Integer
Dim i, j As Integer
  For i = 0 To L - 1
    hist(i) = 0
  Next i

  For x = 0 To xMax
```

```
    For y = 0 To yMax
      i = Val(pict(x, y))
      hist(i) = hist(i) + 1
    Next y
  Next x

End Sub
Static Function Log10(ByVal x As Double) As Double
Log10 = (Log(x) / Log(10#))
End Function
Private Function Ps(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, status As
Integer) _
  As Double
Static sum As Double
Dim i As Integer

If (status = initial) Then
  i = lower_bound
  sum = h(i)
  Ps = 0
  Exit Function
Else
  If status = dynamics Then
    i = S
    sum = sum + h(i)
    Ps = 0
    Exit Function
  End If
End If
Ps = sum
End Function
Private Function iPs(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, status As
Integer) _
  As Double
```

```
Static sum As Double
Dim i As Integer

If (status = initial) Then
   i = lower_bound
   sum = lower_bound * h(i)
   iPs = 0
   Exit Function
Else
   If status = dynamics Then
      i = S
      sum = sum + (S * h(i))
      iPs = 0
      Exit Function
   End If
End If
iPs = sum
End Function
Private Function Ps_inv(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, status
As Integer) _
   As Double
Static sum As Double
Dim i, j As Integer
Dim xMax, yMax As Double
xMax = 256: yMax = 256
If (status = initial) Then
   i = lower_bound
   sum = (xMax * yMax) - h(i)
   Ps_inv = 0
   Exit Function
Else
   If status = dynamics Then
      i = S
      sum = sum - h(i)
```

```
    Ps_inv = 0
    Exit Function
  End If
End If
Ps_inv = sum
End Function
Private Function iPs_inv(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, _
ByVal upper_bound As Integer, ByVal status As Integer) As Double

Static sum As Double
Dim i As Integer

If (status = initial) Then
  sum = 0
  For i = lower_bound To upper_bound
    sum = sum + (i * h(i))
  Next i
  sum = sum - (lower_bound * h(lower_bound))
  iPs_inv = 0
  Exit Function
Else
  If status = dynamics Then
    i = S
    sum = sum - (S * h(i))
    iPs_inv = 0
    Exit Function
  End If
End If
iPs_inv = sum
End Function

Private Function POi(ByRef h() As Long, ByVal i As Integer, ByVal S As Integer, ByVal lower_bound
As Integer, _
ByVal status As Integer) As Double
```

```
Static buffer As Double
Dim temp As Double

If (status = initial) Then
   buffer = Ps(h, S, lower_bound, statics)
Else
   temp = h(i) / buffer
   POi = temp
   Exit Function
End If
POi = 0
End Function
Private Function PBi(ByRef h() As Long, ByVal i As Integer, ByVal S As Integer, ByVal lower_bound
As Integer, _
ByVal status As Integer) As Double
Static buffer As Double
Dim temp As Double

If (status = initial) Then
   buffer = Ps_inv(h, S, lower_bound, statics)
Else
   temp = h(i) / buffer
   PBi = temp
   Exit Function
End If
PBi = 0
End Function
Private Function lambdaO(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer,
ByVal status As Integer) _
   As Double
Static buffer As Double
Dim temp1, temp2 As Double

If (status = initial) Then
```

```
    temp1 = iPs(h, S, lower_bound, statics)
    temp2 = Ps(h, S, lower_bound, statics)
    buffer = temp1 / temp2
Else
    lambdaO = buffer
    Exit Function
End If
    lambdaO = 0
End Function
Private Function lambdaB(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, _
ByVal upper_bound As Integer, ByVal status As Integer) As Double

Static buffer As Double
Dim temp1, temp2 As Double

If (status = initial) Then
    temp1 = iPs_inv(h, S, lower_bound, upper_bound, statics)
    temp2 = Ps_inv(h, S, lower_bound, statics)
    buffer = temp1 / temp2
Else
    lambdaB = buffer
    Exit Function
End If
    lambdaB = 0
End Function

Private Function qOi(ByRef h() As Long, ByVal i As Integer, ByVal S As Integer, ByVal lower_bound
As Integer, _
ByVal status As Integer) As Double

Static lambdaObuffer As Double
Dim product, temp1 As Double
Dim j As Integer
```

```
If (status = initial) Then
   lambdaObuffer = lambdaO(h, S, lower_bound, statics)
   qOi = 0
   Exit Function
Else
   If i = 0 Then
      temp1 = Exp(-1 * lambdaObuffer)
      qOi = temp1
      Exit Function
   Else
      product = Exp(-1 * lambdaObuffer)
      For j = 1 To i
      product = product * (lambdaObuffer / j)
      Next j
   End If
End If
qOi = product
End Function

Private Function qBi(ByRef h() As Long, ByVal i As Integer, ByVal S As Integer, ByVal lower_bound
As Integer, _
ByVal upper_bound As Integer, ByVal status As Integer) As Double

Static lambdaBbuffer As Double
Dim product, temp1 As Double
Dim j As Integer

If (status = initial) Then
   lambdaBbuffer = lambdaB(h, S, lower_bound, upper_bound, statics)
   qBi = 0
   Exit Function
Else
   If i = 0 Then
      temp1 = Exp(-1 * lambdaBbuffer)
```

```
        qBi = temp1
        Exit Function
    Else
        product = Exp(-1 * lambdaBbuffer)
        For j = 1 To i
        product = product * (lambdaBbuffer / j)
        Next j
    End If
End If
qBi = product
End Function
Private Function DO_O(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer) _
    As Double

Dim buffer1, buffer2 As Double
Dim sum1, sum2 As Double
Dim i As Integer
Dim temp1, temp2 As Double

sum1 = 0: sum2 = 0:
temp1 = POi(h, i, S, lower_bound, initial)
temp2 = qOi(h, i, S, lower_bound, initial)

For i = lower_bound To S
    buffer1 = POi(h, i, S, lower_bound, statics)
    buffer2 = qOi(h, i, S, lower_bound, statics)
    If (buffer1 <> 0) And (buffer2 <> 0) Then
        If (buffer1 / buffer2 = 0) Or (buffer2 / buffer1 = 0) Then
        Else
        temp1 = Log10(buffer1 / buffer2)
        temp2 = Log10(buffer2 / buffer1)
        sum1 = sum1 + (buffer1 * temp1)
        sum2 = sum2 + (buffer2 * temp2)
        End If
```

```
    End If
Next i
DO_O = sum1 + sum2

End Function
Private Function DB_B(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, _
ByVal upper_bound As Integer) As Double

Dim buffer1, buffer2 As Double
Dim sum1, sum2 As Double
Dim i As Integer
Dim temp1, temp2 As Double

sum1 = 0: sum2 = 0:
temp1 = PBi(h, i, S, lower_bound, initial)
temp2 = qBi(h, i, S, lower_bound, upper_bound, initial)

For i = S + 1 To upper_bound
   buffer1 = PBi(h, i, S, lower_bound, statics)
   buffer2 = qBi(h, i, S, lower_bound, upper_bound, statics)
   If (buffer1 <> 0) And (buffer2 <> 0) Then
      If (buffer1 / buffer2 = 0) Or (buffer2 / buffer1 = 0) Then
      Else
      temp1 = Log10(buffer1 / buffer2)
      temp2 = Log10(buffer2 / buffer1)
      sum1 = sum1 + (buffer1 * temp1)
      sum2 = sum2 + (buffer2 * temp2)
      End If
   End If
Next i
DB_B = sum1 + sum2

End Function
Private Function DD(ByRef h() As Long, ByVal S As Integer, ByVal lower_bound As Integer, _
```

```
ByVal upper_bound As Integer) As Double

Dim temp As Double

temp = DO_O(h, S, lower_bound) + DB_B(h, S, lower_bound, upper_bound)
DD = temp
End Function

Private Function thresholding(ByRef h() As Long, ByVal L As Integer) As Integer
Dim i, upper_bound, lower_bound As Integer
Dim S, threshold As Integer
Dim mindiv, buffer As Double
Dim temp1, temp2, temp3, temp4 As Double
Dim temp5, temp6 As Double

i = 0: mindiv = 1000
Do While h(i) = 0
   i = i + 1
Loop
lower_bound = i
i = L - 1
Do While h(i) = 0
   i = i - 1
Loop
upper_bound = i
'initial value


temp1 = Ps(h, S, lower_bound, initial)
temp2 = iPs(h, S, lower_bound, initial)
temp3 = Ps_inv(h, S, lower_bound, initial)
temp4 = iPs_inv(h, S, lower_bound, upper_bound, initial)

For S = lower_bound + 1 To upper_bound - 1
```

```
    temp1 = Ps(h, S, lower_bound, dynamics)
    temp2 = iPs(h, S, lower_bound, dynamics)
    temp3 = Ps_inv(h, S, lower_bound, dynamics)
    temp4 = iPs_inv(h, S, lower_bound, upper_bound, dynamics)
    temp5 = lambdaO(h, S, lower_bound, initial)
    temp6 = lambdaB(h, S, lower_bound, upper_bound, initial)
    buffer = DD(h, S, lower_bound, upper_bound)
   'If (mindiv > buffer) Then
      mindiv = buffer
      threshold = S
    End If
Next S

thresholding = threshold
End Function
Public Function BinaryFilter( _
      ByRef picImage As PictureBox) As Boolean
' these are used to address the pixel using matrices
Dim pict() As Byte
Dim temp_image(256, 256) As Byte
Dim sa As SAFEARRAY2D, bmp As BITMAP
Dim i, j As Integer
Dim x, y As Integer
Dim threshold As Integer
    xMax = 0: yMax = 0
    ' get bitmap info
    GetObjectAPI picImage.Picture, Len(bmp), bmp 'dest

    ' exit if not a supported bitmap
    If bmp.bmBitsPixel <> 8 Then
      MsgBox " 8-bit bitmaps only", vbCritical
      Exit Function
    End If
```

```
    ' have the local matrix point to bitmap pixels
    With sa
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = bmp.bmHeight
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = bmp.bmWidthBytes
      .pvData = bmp.bmBits
    End With
    CopyMemory ByVal VarPtrArray(pict), VarPtr(sa), 4

    yMax = UBound(pict, 2)
    xMax = UBound(pict, 1)
    Call read_frequency(pict, hist, L)
    threshold = thresholding(hist, L)

'Do Binary 254 = white, 0 = black
    For x = 0 To xMax
      For y = 0 To yMax
      If Val(pict(x, y)) > threshold Then
        pict(x, y) = 254
      Else
        pict(x, y) = 0
      End If
      Next y
    Next x

    ' clear the temporary array descriptor
    ' without destroying the local temporary array
    CopyMemory ByVal VarPtrArray(pict), 0&, 4
    BinaryFilter = True

  End Function
```

```
'---------- use for Thinning image ----------

Private Function restorage(ByRef pict() As Byte, ByRef temp_image() As Byte) As Integer
Dim x, y, z As Integer
z = 1
For x = 0 To xMax
  For y = 0 To yMax
  If temp_image(x, y) <> 254 Then
    If pict(x, y) = 254 Then
    pict(x, y) = 0
    z = 0
    End If
  End If
  Next y
Next x

restorage = z

End Function

Private Function condition_1(ByVal x As Integer, ByVal y As Integer, ByRef pict() As Byte) As Integer
' input positon of x,y of image
' output result of condition_1 thinning
Dim a, b, c, d As Long
a = 0: b = 0: c = 0: d = 0
'condition 1 A()
If (pict(x - 1, y - 1) = 0) And (pict(x, y - 1) = 254) Then a = a + 1
If (pict(x, y - 1) = 0) And (pict(x + 1, y - 1) = 254) Then a = a + 1
If (pict(x + 1, y - 1) = 0) And (pict(x + 1, y) = 254) Then a = a + 1
If (pict(x + 1, y) = 0) And (pict(x + 1, y + 1) = 254) Then a = a + 1
If (pict(x + 1, y + 1) = 0) And (pict(x, y + 1) = 254) Then a = a + 1
If (pict(x, y + 1) = 0) And (pict(x - 1, y + 1) = 254) Then a = a + 1
If (pict(x - 1, y + 1) = 0) And (pict(x - 1, y) = 254) Then a = a + 1
```

```
If (pict(x - 1, y) = 0) And (pict(x - 1, y - 1) = 254) Then a = a + 1

'condition 2 B()

'FrmImage.Print pict(x - 1, y - 1)
b = Val(pict(x - 1, y - 1)) + Val(pict(x, y - 1)) + Val(pict(x + 1, y - 1)) + Val(pict(x + 1, y)) _
 + Val(pict(x + 1, y + 1)) + Val(pict(x, y + 1)) + Val(pict(x - 1, y + 1)) + Val(pict(x - 1, y))

'condition 3 //P2*P4*P6
c = Val(pict(x, y - 1)) * Val(pict(x + 1, y)) * Val(pict(x, y + 1))

'condition 4 //P4*P6*P8
d = Val(pict(x + 1, y)) * Val(pict(x, y + 1)) * Val(pict(x - 1, y))

'decision
  If (a = 1) And (254 < b) And (b < 2032) And (c = 0) And (d = 0) Then
    condition_1 = 0 'delete
  Else
    condition_1 = 254 'undelete
  End If

End Function

Private Function condition_2(ByVal x As Integer, ByVal y As Integer, ByRef pict() As Byte) As Integer
' input positon of x,y of image
' output result of condition_2 thinning
Dim a, b, c, d As Long
a = 0: b = 0: c = 0: d = 0
'condition 1 A()
If (pict(x - 1, y - 1) = 0) And (pict(x, y - 1) = 254) Then a = a + 1
If (pict(x, y - 1) = 0) And (pict(x + 1, y - 1) = 254) Then a = a + 1
If (pict(x + 1, y - 1) = 0) And (pict(x + 1, y) = 254) Then a = a + 1
If (pict(x + 1, y) = 0) And (pict(x + 1, y + 1) = 254) Then a = a + 1
If (pict(x + 1, y + 1) = 0) And (pict(x, y + 1) = 254) Then a = a + 1
```

```
If (pict(x, y + 1) = 0) And (pict(x - 1, y + 1) = 254) Then a = a + 1
If (pict(x - 1, y + 1) = 0) And (pict(x - 1, y) = 254) Then a = a + 1
If (pict(x - 1, y) = 0) And (pict(x - 1, y - 1) = 254) Then a = a + 1

'condition 2 B()
b = Val(pict(x - 1, y - 1)) + Val(pict(x, y - 1)) + Val(pict(x + 1, y - 1)) + Val(pict(x + 1, y)) _
  + Val(pict(x + 1, y + 1)) + Val(pict(x, y + 1)) + Val(pict(x - 1, y + 1)) + Val(pict(x - 1, y))

'condition 3 //P2*P4*P8
c = Val(pict(x, y - 1)) * Val(pict(x + 1, y)) * Val(pict(x - 1, y))

'condition 4 //P2*P6*P8
d = Val(pict(x, y - 1)) * Val(pict(x, y + 1)) * Val(pict(x - 1, y))

'decision
   If (a = 1) And (254 < b) And (b < 2032) And (c = 0) And (d = 0) Then
      condition_2 = 0 'delete
   Else
      condition_2 = 254 'undelete
   End If

End Function


Public Function ThinFilter( _
      ByRef picImage As PictureBox) As Boolean
' these are used to address the pixel using matrices
Dim pict() As Byte
Dim temp_image(256, 256) As Byte
Dim sa As SAFEARRAY2D, bmp As BITMAP
Dim fraction As Long, sum As Long
Dim a, b As Integer
Dim x, y As Integer
```

```
xMax = 0: yMax = 0
' get bitmap info
GetObjectAPI picImage.Picture, Len(bmp), bmp 'dest

' exit if not a supported bitmap
If bmp.bmBitsPixel <> 8 Then
   MsgBox " 8-bit bitmaps only", vbCritical
   Exit Function
End If

' have the local matrix point to bitmap pixels
With sa
   .cbElements = 1
   .cDims = 2
   .Bounds(0).lLbound = 0
   .Bounds(0).cElements = bmp.bmHeight
   .Bounds(1).lLbound = 0
   .Bounds(1).cElements = bmp.bmWidthBytes
   .pvData = bmp.bmBits
End With
CopyMemory ByVal VarPtrArray(pict), VarPtr(sa), 4

' Do Thinning 254 = white, 0 = black

yMax = UBound(pict, 2)
xMax = UBound(pict, 1)
' set top and buttom to 0
For x = 0 To xMax
  pict(x, 0) = 0
  pict(x, yMax) = 0
Next x
' set left and right to 0
For y = 0 To yMax
  pict(0, y) = 0
```

```
    pict(xMax, y) = 0
  Next y

  a = 0: b = 0

  Do While (a = 0) And (b = 0)
    For x = 1 To xMax
      For y = 1 To yMax
      If pict(x, y) = 254 Then
          temp_image(x, y) = 254
          temp_image(x, y) = Str(condition_1(x, y, pict))
        End If
        Next y
    Next x

  a = restorage(pict, temp_image)

     For x = 1 To xMax
       For y = 1 To yMax
       If pict(x, y) = 254 Then
          temp_image(x, y) = 254
          temp_image(x, y) = Str(condition_2(x, y, pict))
        End If
        Next y
    Next x

  b = restorage(pict, temp_image)

  Loop
  ' End while loop

  ' clear the temporary array descriptor
  ' without destroying the local temporary array
  CopyMemory ByVal VarPtrArray(pict), 0&, 4
```

```
    ThinFilter = True

End Function

'---------- Extracting image ----------
Public Sub Post_Pro(ByVal rule As Integer)
Dim x, y As Integer
Dim NumX_block, NumY_block, j, i, EPCount, BPCount As Integer

'Rule:1 เอาจุด BP และ EP ที่อยู่ใน block ริมขอบรูปออกก่อน
If rule = 1 Then
   For x = 0 To xMax
      For y = 0 To 4
         extracted(x, y) = 0
      Next y
   Next x
   For x = 0 To xMax
      For y = yMax - 4 To yMax
         extracted(x, y) = 0
      Next y
   Next x
   For x = 0 To 4
      For y = 0 To yMax
         extracted(x, y) = 0
      Next y
   Next x
   For x = xMax - 4 To xMax
      For y = 0 To yMax
         extracted(x, y) = 0
      Next y
   Next x
'End If
```

```
'If RideWidth >= 5 And rule <> 1 Then
'Rule 2: EP อยู่ในขอบเขต RideWidth ที่ กำหนด ให้ลบออก ให้หมด
ElseIf rule = 2 Then
  For x = RideWidth To xMax - RideWidth
    For y = RideWidth To yMax - RideWidth
    EPCount = 0
      If extracted(x, y) = EP Then
        For i = x To x + (RideWidth)
          For j = y + 1 To y + (RideWidth)
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x To x - (RideWidth) Step -1
          For j = y - 1 To y - (RideWidth) Step -1
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x - 1 To x - (RideWidth) Step -1
          For j = y To y + (RideWidth)
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i
```

```
            For i = x + 1 To x + (RideWidth)
              For j = y To y - (RideWidth) Step -1
                If extracted(i, j) = EP Then
                  EPCount = EPCount + 1
                  extracted(i, j) = 0
                End If
              Next j
            Next i

            If EPCount > 0 Then
              extracted(x, y) = 0
            End If
          End If
        Next y
      Next x
'End If

'Rule 3:ถ้า BP อยู่ในขอบเขต RideWidth เดียวกัน ก็ให้ลบ BP นั้นทิ้งออกไป
ElseIf rule = 3 Then
  For x = RideWidth To xMax - RideWidth
    For y = RideWidth To yMax - RideWidth
    BPCount = 0
      If extracted(x, y) = BP Then
        For i = x To x + (RideWidth)
          For j = y + 1 To y + (RideWidth)
            If BP = extracted(i, j) Then
              BPCount = BPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x To x - (RideWidth) Step -1
          For j = y - 1 To y - (RideWidth) Step -1
```

```
            If BP = extracted(i, j) Then
              BPCount = BPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x - 1 To x - (RideWidth) Step -1
          For j = y To y + (RideWidth)
            If BP = extracted(i, j) Then
              BPCount = BPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x + 1 To x + (RideWidth)
          For j = y To y - (RideWidth) Step -1
            If BP = extracted(i, j) Then
              BPCount = BPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        If BPCount > 0 Then
          extracted(x, y) = 0
        End If
      End If
    Next y
  Next x
'End If
```

'Rule 4:จุด BP และ EP ใด ๆ ที่อยู่ในขอบเขตเดียวกัน ก็ให้ตัดทิ้งไปทั้งคู่

```
ElseIf rule = 4 Then
  For x = RideWidth To xMax - RideWidth
    For y = RideWidth To yMax - RideWidth
    EPCount = 0
      If extracted(x, y) = BP Then
        For i = x To x + (RideWidth)
          For j = y + 1 To y + (RideWidth)
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x To x - (RideWidth) Step -1
          For j = y - 1 To y - (RideWidth) Step -1
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x - 1 To x - (RideWidth) Step -1
          For j = y To y + (RideWidth)
            If extracted(i, j) = EP Then
              EPCount = EPCount + 1
              extracted(i, j) = 0
            End If
          Next j
        Next i

        For i = x + 1 To x + (RideWidth)
          For j = y To y - (RideWidth) Step -1
```

```
                If extracted(i, j) = EP Then
                    EPCount = EPCount + 1
                    extracted(i, j) = 0
                End If
            Next j
        Next i

        If EPCount > 0 Then
            extracted(x, y) = 0
        End If
    End If
  Next y
Next x
End If
'Else
'  EPCount = MsgBox("ค่าต้องมากกว่าหรือเท่ากับ 5 ขึ้นไป", vbInformation, "ค่าผิด")
'End If
End Sub

Private Sub extraction(ByRef bit_thin() As Integer)
Dim x, y As Integer
Dim Cn, sum As Integer
'Dim ending(256) As Byte
'Dim temp1, temp2 As Integer
'temp1 = 0: temp2 = 0
For x = 1 To xMax - 1
  For y = 1 To yMax - 1
    If bit_thin(x, y) = 1 Then
    sum = 0
    sum = sum _
      + Abs(bit_thin(x + 1, y + 1) - bit_thin(x + 1, y)) _
      + Abs(bit_thin(x + 1, y) - bit_thin(x + 1, y - 1)) _
      + Abs(bit_thin(x + 1, y - 1) - bit_thin(x, y - 1)) _
      + Abs(bit_thin(x, y - 1) - bit_thin(x - 1, y - 1)) _
```

```
        + Abs(bit_thin(x - 1, y - 1) - bit_thin(x - 1, y)) _
        + Abs(bit_thin(x - 1, y) - bit_thin(x - 1, y + 1)) _
        + Abs(bit_thin(x - 1, y + 1) - bit_thin(x, y + 1)) _
        + Abs(bit_thin(x, y + 1) - bit_thin(x + 1, y + 1))

      If sum = 2 Then
        extracted(x, y) = EP
      ElseIf sum = 6 Then
        extracted(x, y) = BP
      Else
        extracted(x, y) = 0
      End If
      End If
    Next y
Next x
End Sub

Public Function ExtractFilter( _
      ByRef picImage As PictureBox) As Boolean
'these are used to address the pixel using matrices
Dim pict() As Byte
Dim sa As SAFEARRAY2D, bmp As BITMAP
Dim x, y As Integer
Dim i, j As Integer

   xMax = 0: yMax = 0
   ' get bitmap info
   GetObjectAPI picImage.Picture, Len(bmp), bmp 'dest

   ' exit if not a supported bitmap
   If bmp.bmBitsPixel <> 8 Then
      MsgBox " 8-bit bitmaps only", vbCritical
      Exit Function
   End If
```

```
' have the local matrix point to bitmap pixels
With sa
   .cbElements = 1
   .cDims = 2
   .Bounds(0).lLbound = 0
   .Bounds(0).cElements = bmp.bmHeight
   .Bounds(1).lLbound = 0
   .Bounds(1).cElements = bmp.bmWidthBytes
   .pvData = bmp.bmBits
End With
CopyMemory ByVal VarPtrArray(pict), VarPtr(sa), 4

yMax = UBound(pict, 2)
xMax = UBound(pict, 1)
' Do Extract 254 = white, 0 = black
' Convert image from 254 to 1
For x = 0 To xMax
   For y = 0 To yMax
   If pict(x, y) = 254 Then
      thined(x, y) = 1
   Else
      thined(x, y) = 0
   End If
   Next y
Next x

' Do extract algorithm
Call extraction(thined)
Call Post_Pro(1)
For x = 0 To 256
   For y = 0 To 256
      Save_extract(x, y) = extracted(x, y)
   Next y
```

```
    Next x
    ' clear the temporary array descriptor
    ' without destroying the local temporary array
    CopyMemory ByVal VarPtrArray(pict), 0&, 4
    ExtractFilter = True
End Function


Public Function RideCount() As Integer
Dim counter As Integer
Dim i, j As Integer
counter = 0
   For i = 0 To xMax
      For j = 0 To yMax
         If (extracted(i, j) = EP) Or (extracted(i, j) = BP) Then
            counter = counter + 1
         End If
      Next j
   Next i
   RideCount = counter
End Function

Public Function AddFeature(ByVal x As Integer, ByVal y As Integer) As Boolean
Dim i, j, Xtmp, Ytmp As Integer
Dim Found As Boolean
Dim xNearest, yNearest As Integer
Dim counterX, counterY As Integer

xNearest = 7       ' for error +- 7 pixel
yNearest = 7
Found = False
Xtmp = x - 7
Ytmp = y - 7
For counterX = 0 To 14
```

```
For counterY = 0 To 14
  i = Xtmp + counterX
  j = Ytmp + counterY
  If i >= 0 And j >= 0 And i <= xMax And j <= yMax Then
    If i <> x Or j <> y Then
      If Save_extract(i, j) = EP Or Save_extract(i, j) = BP Then
        Found = True
        If Abs(i - x) <= xNearest Or Abs(j - y) <= yNearest Then ' for Nearest x,y point
          xNearest = i
          yNearest = j
        End If
      End If
    End If
  End If
Next counterY
Next counterX
If Found Then
  extracted(xNearest, yNearest) = Save_extract(xNearest, yNearest)
End If
AddFeature = Found
End Function

Public Function SubFeature(ByVal x As Integer, ByVal y As Integer) As Boolean
Dim i, j, Xtmp, Ytmp As Integer
Dim Found As Boolean
Dim xNearest, yNearest As Integer
Dim counterX, counterY As Integer

xNearest = 7      ' for error +- 7 pixel
yNearest = 7
Found = False
Xtmp = x - 7
Ytmp = y - 7
For counterX = 0 To 14
```

```
For counterY = 0 To 14
i = Xtmp + counterX
j = Ytmp + counterY
  If i >= 0 And j >= 0 And i <= xMax And j <= yMax Then
    If i <> x Or j <> y Then
      If extracted(i, j) = EP Or extracted(i, j) = BP Then
        Found = True
        If Abs(i - x) <= xNearest Or Abs(j - y) <= yNearest Then ' for Nearest x,y point
          xNearest = i
          yNearest = j
        End If
      End If
    End If
  End If
Next counterY
Next counterX
If Found Then
    extracted(xNearest, yNearest) = 0
End If
SubFeature = Found
End Function

Public Sub Undoextracted()
Dim i, j As Integer
  For i = 0 To xMax
    For j = 0 To yMax
      extracted(i, j) = Save_extract(i, j)
    Next j
  Next i
End Sub
```