การศึกษาระบบฐานข้อมูลเชิงวัตถุ

Studying the Object-Oriented Database System

โดย

นางสาวนันทวัน นาคอร่าม

รหัส 38626073

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2540

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การศึกษาระบบฐานข้อมูลเชิงวัตถุ
นักศึกษา	นางสาว นันทวัน นาคอร่าม
อาจารย์ที่ปรึกษา	รศ.ดร.ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
พ.ศ.	2540

บทคัดย่อ

ระบบฐานข้อมูลเชิงวัตถุเป็นโครงสร้างข้อมูลชั้นสูงรูปแบบหนึ่งที่เหมาะสมกับการใช้งานข้อมูลเชิงซับซ้อน เช่น ข้อมูลประเภทรูปภาพ หรือ ข้อมูลที่มีขนาดใหญ่ เป็นต้น ซึ่งแนวทางการในการศึกษาระบบฐานข้อมูลเชิงวัตถุ จะเน้นการศึกษาทางด้านโครงสร้างข้อมูล และสถาปัตยกรรมของระบบฐานข้อมูล พร้อมทั้งศึกษาตัวอย่างการใช้งานระบบการจัดการฐานข้อมูลเชิงวัตถุ ซึ่งในที่นี้คือการเลือกศึกษาระบบจัดการฐานข้อมูลเชิงวัตถุ POSTGRESQL และออกแบบระบบงานตัวอย่าง คือ ระบบงานห้องสมุด ทั้งด้านการออกแบบ และการสร้างฟังก์ชันในการทำงาน คือ ฟังก์ชันการสืบค้นข้อมูลทั้งด้านข้อมูลสารนิเทศ และประวัติการยืม-คืน ของสมาชิก และการเพิ่มรายการสารนิเทศใหม่เข้าไปในระบบผ่านทางเครือข่ายอินเตอร์เน็ทได้

Title	Studying the Object-Oriented Database
Student	Miss Nantawan Nakaram
Advisor	Assoc.Prof. Dr. Suphamit Chittayasothorn
Level of Study	Master of Science in Information Technology
Major	Information Science
Year	1970

ABSTRACTS

Object-Oriented Database System is one of the high-level database models that appropriate with complex data, e.g., Image or large data. This project we will address in the study of it's Architecture and implement the " Library System on the Internet " by using PostgreSQL ; Object-Oriented Database product . The functions of this system that we want to present are composed of searching, managing the history of circulation and New Entry of data in system.

กิตติกรรมประกาศ

โครงงานชิ้นนี้สำเร็จลงได้ด้วยดีเพราะ ได้รับความกรุณา และกำลังใจที่บุพการีมีให้ตลอดช่วงเวลาในการศึกษา และการพัฒนาระบบงาน ขอขอบพระคุณที่ท่านคอยช่วยเหลือในทุกๆ ด้าน

ขอขอบพระคุณสำหรับการให้คำปรึกษาอย่างดียิ่งจาก รศ.ดร.ศุภมิตร จิตตะยโศธร พร้อมทั้งเวลาที่ท่านสละให้ในการเตรียมงาน และการให้คำแนะนำต่างๆ

ขอขอบคุณท่านต่างๆ ที่มีส่วนในการช่วยงานทุกท่าน คือ

เจ้าหน้าที่ของคณะเทคโนโลยีสารสนเทศที่ช่วยในการให้ข้อมูลต่างๆ

คณะเทคโนโลยีสารสนเทศที่ช่วยในการใช้อุปกรณ์ต่างๆ ของคณะฯ ในการทำโครงงาน

และเพื่อนๆ ที่คอยช่วยเหลือซึ่งกันและกันในด้านต่างๆ คือ เกษรา เวียงวะลัย ในการให้ที่พักอาศัยในช่วงพัฒนาระบบงาน, ฉลองชัย จึงประเสริฐพร ที่คอยช่วยเหลือในทุกๆ ด้านตั้งแต่เริ่มต้นการทำงานจนถึงส่วนสุดท้ายของระบบงาน, กมลพรรณ ทองพูล, ปราลี มณีรัตน์ และจิรพร ควรชัยตระกูล ที่ช่วยเป็นกำลังใจ และร่วมแก้ไขปัญหาต่างๆ ที่เกิดขึ้น พร้อมทั้งเพื่อนๆ ที่เรียนร่วมกันมาทุกท่าน และน้องๆ รุ่นสองที่คอยเป็นห่วงเป็นใย

นางสาว นันทวัน นาคอร่าม

สารบัญ

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

บทที่ 1

บทนำ

ความเป็นมาและความสำคัญของระบบ

ระบบการจัดการฐานข้อมูลเชิงวัตถุ(ODBMS) ได้รับความนิยมมาตั้งแต่ปี 1980 แต่ปัญหาของ ODBMS ในช่วงแรกๆ ที่ค้นพบ คือยังไม่เป็นระบบฐานข้อมูลที่สมบูรณ์ ยังขาดความสามารถในการสำรอง และกู้คืนข้อมูล, แบบจำลองข้อมูล (Data Model) ยังมีความสับสนในบางส่วน, ภาษาที่ใช้ยังไม่มีมาตรฐานที่แน่นอน, มีปัญหาเกี่ยวกับการสืบค้นข้อมูล นอกจากนั้นตัว ODBMS เองยังขาดคุณสมบัติในด้านความยืดหยุ่น เนื่องด้วยการใช้พื้นที่ในหน่วยความจำสูง

ผู้ผลิตหลายรายที่ได้ใช้ ODBMS จำเป็นต้องเลิกราการใช้งานไปเนื่องจากประสิทธิภาพในการทำงานที่ล่าช้า ทำให้จำนวนผู้ใช้งาน ODBMS ลดลง แต่ปัจจุบันการพัฒนาของ ODBMS ได้เปลี่ยนแปลงไปจากเดิม จากทัศนคติและแนวความคิดเดิมที่มีต่อ ODBMS คือ

1. ODBMS ยังขาดทฤษฎีพื้นฐานรองรับ RDBMS นั้นเป็นแบบจำลองที่มีทฤษฎีทางคณิตศาสตร์มารองรับ และยังมีเครื่องมือที่ใช้ในการออกแบบฐานข้อมูลด้วยในขณะที่ ODBMS ไม่มีสิ่งที่สนับสนุนการทำงานทั้งสองอย่าง แต่ปัจจุบันเครื่องมือที่สนับสนุนในการออกแบบของ ODBMS คือ Jasmine ของ Computer -Associates
2. ODBMS ไม่มีภาษามาตรฐานมารองรับการทำงาน
3. ODBMS ไม่มีความยืดหยุ่น ซึ่งความจริงแล้วปัญหาข้อนี้เป็นเพียงปัญหาในตอนเริ่มแรกของการพัฒนาระบบ แต่ปัจจุบัน ODBMS มีให้เลือกใช้งานหลายระดับ และมีคุณสมบัติทางด้านการจัดการความปลอดภัยของข้อมูล, การสำรองข้อมูล, และการกู้คืนข้อมูล เช่นเดียวกับระบบฐานข้อมูลเชิงสัมพันธ์

ปัจจุบันระบบคอมพิวเตอร์ได้ถูกพัฒนาให้เป็นระบบสื่อสารข้อมูลที่มีการเชื่อมโยงเป็นเครือข่าย ระบบฐานข้อมูลจึงมีการพัฒนาในรูปแบบของออบเจ็กต์เพื่อสนับสนุนการทำงานแบบมัลติมีเดีย การสร้างฐานข้อมูลมัลติมีเดียส่วนใหญ่จะให้ความสำคัญกับการจัดการข้อมูลประเภทต่างๆ คือ วีดีโอ, อิมเมจ, เสียง, ข้อมูลเท็กซ์, และกราฟิก ซึ่งการประยุกต์ใช้งานกับฐานข้อมูลแบบออบ

เจ็กต์นั้นจะสนับสนุนการทำงานนี้ เพราะการที่ผู้ใช้สามารถกำหนดชนิดของข้อมูลใหม่ตามที่ต้องการได้ จะทำให้การใช้งาน หรือเรียกข้อมูลประเภทมัลติมีเดียมาใช้งานง่าย และสะดวกต่อการใช้งานมากขึ้น

วัตถุประสงค์ของงานวิจัย

ในการพัฒนาระบบงานวิจัยนี้มีวัตถุประสงค์ดังนี้

1. ศึกษาทฤษฎี และหลักการทำงานของระบบฐานข้อมูลเชิงวัตถุ
2. ศึกษาตัวอย่างระบบฐานข้อมูลเชิงวัตถุในด้านต่างๆ คือ
   - โครงสร้างทางด้านสถาปัตยกรรม
   - รูปแบบการใช้งานโปรแกรมประยุกต์กับระบบฐานข้อมูลเชิงวัตถุ
   - วิธีการใช้งานของระบบการจัดการฐานข้อมูลเชิงวัตถุ
3. การสร้างตัวอย่างโปรแกรมประยุกต์ พร้อมทั้งออกแบบโครงสร้างการทำงานให้อยู่ในรูปของระบบฐานข้อมูลเชิงวัตถุ
4. ประยุกต์การใช้งานตัวอย่างโปรแกรมประยุกต์บนระบบเครือข่าย

ทฤษฎี หรือแนวความคิดที่เกี่ยวข้องกับงานวิจัย

การศึกษาระบบการจำลองแบบ (model) ของฐานข้อมูล (database) ประกอบไปด้วยหลายรูปแบบ เช่น ER model ซึ่งเป็นแบบจำลองพื้นฐานข้อมูลเชิงสัมพันธ์ (Relational Database) รวมทั้งแบบจำลองที่เป็นแบบเครือข่าย และแบบจำลองลำดับขั้น เป็นต้น การพัฒนาทางด้านแบบจำลองข้อมูล และระบบต่างๆ ได้รับความสำเร็จในการนำไปประยุกต์ใช้กับเทคโนโลยีทางด้านฐานข้อมูลไม่ว่าจะเป็น ER model ที่นำไปสู่การพัฒนาแบบ EER model และแบบจำลองข้อมูลอื่นๆ เช่น functional, nested relational, structure และ semantic model

อย่างไรก็ตามการประยุกต์ใช้กับ applicaiton ที่มีความซับซ้อนยังก่อให้เกิดข้อบกพร่องอยู่ เช่นก่ารออกแบบทางด้านวิศวกรรม และโรงงาน (CAD/CAM and CIM) การจัดการฐานข้อมูลที่เป็น image และ graphic, ฐานข้อมูลทางด้านมัลติมีเดีย และการกำหนดรูปแบบการเข้าถึงข้อมูลร่วม (multiple database system) ซึ่ง applications ที่กล่าวถึงเหล่านี้มีลักษณะการทำงานที่ต่างไปจากเดิม ดังนั้นจึงมีการนำ object-oriented database มาประยุกต์ใช้งานกับ application ที่มีความซับซ้อนมากๆ โดยที่สามารถจัดเก็บข้อมูลที่อยู่ในรูปแบบรูปภาพ และข้อมูลที่มีขนาดใหญ่มากๆ ได้อย่างง่ายดาย เพราะ object-oriented database มีความยืดหยุ่นในการจัดการกับความต้องการต่างๆ ของผู้ใช้ได้โดยปราศจากข้อจำกัดในด้านประเภทของข้อมูล (data type) และภาษา

ที่ใช้ ซึ่งลักษณะเด่นของ object-oriented นี้ ทำให้ผู้ใช้สามารถกำหนดทั้งโครงสร้าง และเครื่องมือในการทำงานกับ complex objects ได้

ประโยชน์ที่คาดว่าจะได้รับ

ในงานวิจัยนี้ได้ทำการศึกษาโครงสร้างด้านต่างๆ ของระบบฐานข้อมูลเชิงวัตถุ และตัวอย่างการใช้งานระบบจัดการฐานข้อมูลเชิงวัตถุ ดังนั้นศาดว่าประโยชน์ที่จะได้รับคือ

1. ทราบความแตกต่างในด้านโครงสร้างของภาษาที่ใช้ในการทำงานระหว่างระบบฐานข้อมูลเชิงวัตถุ และระบบฐานข้อมูลเชิงสัมพันธ์
2. เรียนรู้โครงสร้างการทำงานของระบบฐานข้อมูลเชิงวัตถุ
3. สามารถสร้างโปรแกรมประยุกต์ที่ใช้งานกับระบบฐานข้อมูลเชิงวัตถุได้ และเรียนรู้การควบคุมการทำงานระบบฐานข้อมูลเชิงวัตถุ

ขอบเขตงานวิจัย

ศึกษาระบบฐานข้อมูลเชิงวัตถุในด้านต่างๆ พร้อมทั้งนำโปรแกรมระบบจัดการฐานข้อมูลมาทำการศึกษาการใช้งาน และทำการสร้างโปรแกรมประยุกต์ขึ้นมา โดยทำการออกแบบให้มีโครงสร้างการทำงานสนับสนุนระบบฐานข้อมูลเชิงวัตถุ ซึ่งตัวอย่างของโครงงานคือ โครงงานการค้นหาข้อมูลสารนิเทศของระบบงานห้องสมุด ซึ่งแนวทางในการทำงานคือ เพื่อศึกษาวิธีการออกแบบระบบให้เป็นระบบฐานข้อมูลเชิงวัตถุ แล้วสร้างฟังก์ชันการตัวอย่างการทำงาน ซึ่งในระบบนี้ก็คือการสร้างฟังก์ชันในการสืบหาข้อมูล และตัวอย่างฟังก์ชันในการเพิ่มข้อมูลรายการสิ่งพิมพ์ใหม่เข้าไปในระบบ โดยประยุกต์ให้ระบบทำงานอยู่บนเครือข่ายอินเตอร์เน็ท เพื่อให้ผู้ใช้ทั้งที่เป็นสมาชิกและผู้ใช้ที่ไม่ใช่สมาชิกเข้ามาทำการค้นหาข้อมูล หรือรายการต่างๆ ของระบบงานห้องสมุดผ่านทางระบบเครือข่ายอินเตอร์เน็ทได้ หรือการที่สมาชิกสามารถทำการสืบค้นประวัติการยืม-คืนจากระบบห้องสมุดได้

เนื้อหาในวิทยานิพนธ์

เนื้อหางานวิจัยในบทแรกนี้จะกล่าวถึงความสำคัญของระบบงาน และวัตถุประสงค์ในการทำงานวิจัย บทที่ 2 กล่าวถึงระบบฐานข้อมูลเชิงวัตถุ ซึ่งจะกล่าวถึงโครงสร้างต่างๆ ของระบบฐานข้อมูลเชิงวัตถุ ในบทที่ 3 จะกล่าวถึงทฤษฎี และหลักการทำงานของระบบจัดการฐานข้อมูลเชิงวัตถุ PostgreSQL บทที่ 4 กล่าวถึงโครงสร้าง และรูปแบบการใช้งานของ PostgreSQL บทที่ 5 กล่าวถึงโปรแกรมประยุกต์ และวิธีการในการออกแบบของระบบฐานข้อมูลห้องสมุด และในบทที่ 6

จะกล่าวถึงบทสรุป และแนวทางในการพัฒนาระบบต่อไป พร้อมทั้งข้อจำกัดต่างๆ ที่ใช้ในการพัฒนาระบบเพื่อให้ได้ระบบที่สมบูรณ์ตามที่สามารถใช้งานได้จริง

บทที่ 2

ระบบฐานข้อมูลเชิงวัตถุ

การศึกษาในบทนี้จะทำการศึกษาเกี่ยวกับหลักการ และทฤษฎีที่เกี่ยวกับการพัฒนาระบบฐานข้อมูลเชิงวัตถุในด้านต่างๆ

Object-Oriented Database System

Object-Oriented Database คือ การนำวิธีการจัดการข้อมูลเชิงวัตถุมาพิจารณาประกอบร่วมกับความสามารถต่างๆ ของฐานข้อมูล เพื่อนำไปประยุกต์สู่ระบบการจัดการฐานข้อมูลในรูปแบบของวัตถุที่มีความเป็นอิสระต่อกันในการทำงาน

Object-Oriented Data Model

ปัญหาหลักของ Object-Oriented programming ก็คือ การที่ไม่มีการกำหนดมาตรฐานและกฎเกณฑ์ที่เกี่ยวกับ Object-Oriented ที่แน่นอน ซึ่งแตกต่างกับ Relational data model ที่มีการกำหนดรูปแบบที่แน่นอนส่งผลให้สามารถกำหนดโครงสร้างสำหรับภาษา SQL ได้และมีรูปแบบในการนำเสนอข้อมูลในรูปของตาราง (table)

โครงสร้างข้อมูล และลักษณะพื้นฐานของฐานข้อมูลเชิงวัตถุ [3]

<u>โครงสร้างข้อมูลเชิงวัตถุ ประกอบด้วย</u>

OBJECT รูปแบบพื้นฐานของข้อมูลเชิงวัตถุ คือ วัตถุ ซึ่งวัตถุ คือ สิ่งใดๆ ที่มีอยู่จริง และสามารถจับต้องได้ หรือสิ่งใดๆ ที่สามารถแสดงลักษณะใดๆ ได้ โดยที่ลักษณะเหล่านั้นเป็นคุณสมบัติเฉพาะของแต่ละวัตถุที่สามารถแปรเปลี่ยนไปมาได้ตลอดการคงอยู่ของวัตถุเหล่านั้น ตามหลักความคิดพื้นฐานทางด้านฐานข้อมูลนั้น วัตถุก็คือ แนวความคิดของ entity นั่นเอง

วัตถุนั้นจะแสดงคุณสมบัติในการซ่อนข้อมูล, การกำหนดชื่อ และโอเปอเรชั่น (operation) เป็นต้น วัตถุแต่ละวัตถุสามารถติดต่อสื่อสารซึ่งกัน โดยผ่านวิธีการที่เรียกว่า "การส่งผ่านข้อความ" (message passing) ที่เป็นตัวกระตุ้นการตอบสนองโดยผ่านโอเปอเรชั่นที่ซ่อนอยู่ในตัวของวัตถุที่

เป็นผู้รับ นั่นคือ ข้อมูลที่จัดเก็บอยู่ภายในวัตถุผู้รับจะถูกเข้าถึง และดำเนินการโดยโอเปอเรชั่นของวัตถุนั้นเอง

วัตถุจะประกอบด้วยส่วนของการเชื่อมต่อ และส่วนของการพัฒนาการ (Implementation) ซึ่งการพัฒนาการนั้นจะประกอบด้วยส่วนของโครงสร้างข้อมูล (Data Structure) และโอเปอเรชั่นโค๊ด ซึ่งโครงสร้างข้อมูลจะถูกซ่อนจากภายนอก โดยได้รับการเข้าถึง และจัดการโดยตัวเอง สำหรับโอเปอเรชั่นโค๊ดนั้นเมื่อมีการเปลี่ยนแปลงจะปราศจากผลกระทบใดๆ กับส่วนที่เชื่อมต่อกัน

CLASS คือ กลุ่มของวัตถุโดยแบ่งตามลักษณะเฉพาะ และการใช้งาน หรือวัตถุที่มีคุณสมบัติพื้นฐานเหมือนกัน ซึ่งข้อมูลเชิงวัตถุที่เก็บอยู่ภายในคลาสจะเรียกว่า instance ของคลาส โดยที่ instance จะเป็นส่วนขยายของคลาส และถูกเก็บอยู่ในส่วนของฐานข้อมูล ดังนั้นข้อมูลต่างๆ ที่เป็นตัวกำหนดคลาสจะถูกเข้าถึง และถ่ายทอดผ่านทางวัตถุ instance และ class method

- *Instance method* คือ โอเปอเรชั่นที่ถูกประยุกต์ใช้งานกับ instance แต่ละตัวในคลาสที่กำหนด
- *Class method* คือ วิธี (method) ที่ถูกประยุกต์ใช้กับส่วนของคลาสทั้งหมด เช่น Total_Employees ที่ใช้คำนวณหาจำนวนพนักงานทั้งหมด

ATTRIBUTES คือ โครงสร้างของฟิลด์ภายในทูเปิลที่ให้สื่อทางด้านความหมายของแต่ละทูเปิล สำหรับแอททริบิวท์จะถูกแปลงไปเป็น atomic object หรือกลุ่มของ atomic object ซึ่งแอททริบิวท์จะแบ่งออกเป็น 2 ประเภท คือ

- *Class Attributes* คือ แอททริบิวท์ของคลาสนั้นๆ ที่ถูกเข้าถึงโดยตัวคลาสเอง, instance และ subclass ของตัวคลาส
- *Instance Attributes* คือ แอททริบิวท์ที่เข้าถึงโดย instance ของคลาสที่กำหนดเท่านั้น ซึ่งจะคล้ายคลึงกับแอททริบิวท์ในโครงสร้างอื่นๆ โดยที่ instance attributes นั้นจะมีค่าตายตัวของมันเอง (Default instance attributes) หรือ อาจจะมีค่าสากลที่ประยุกต์ใช้กับแต่ละ instance ของคลาสได้ (Shared instance attributes)

RELATIONSHIP คือ ฟิลด์ของทูเปิล ซึ่ง relationship จะแมพทูเปิลของออบเจ็กต์ กับทูเปิลของออบเจ็กต์ตัวอื่นๆ หรือกลุ่มของทูเปิลออบเจ็กต์

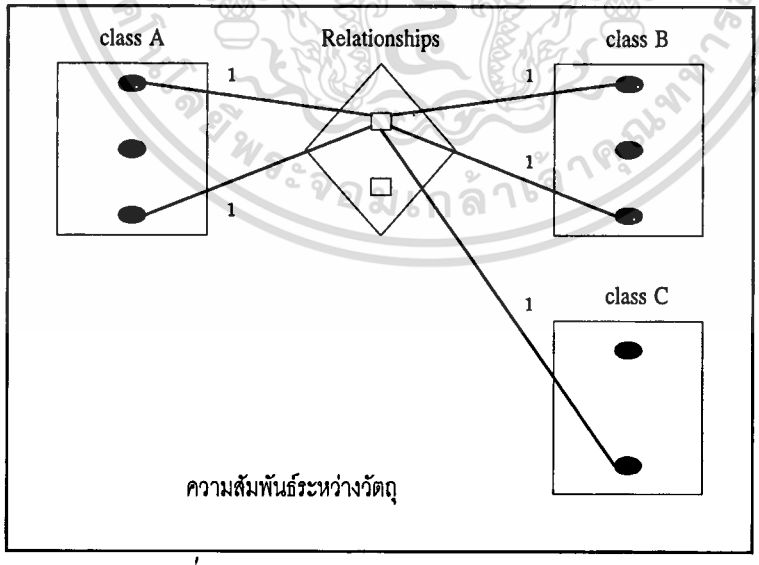
Relationship สามารถแบ่งออกได้เป็น 2 ชนิดหลักคือ ความสัมพันธ์เชิงลำดับขั้น (Hierarchical Relationship) และความสัมพันธ์แบบไม่มีกฎเกณฑ์ (Non-Hierarchical Relationship)

- *Hierarchical Relationship* คือ ความสัมพันธ์ที่เชื่อมระหว่าง 2 กลุ่ม ในรูปแบบเชิงลำดับขั้น
- *Non-Hierarchical Relationship* คือ ความสัมพันธ์ในรูปแบบนี้ไม่สามารถเสนอออกมาในรูปของความสัมพันธ์เชิงลำดับขั้นได้

Hierarchical Relationship สามารถแบ่งได้ 4 ชนิด คือ

- Generalization
- Classification
- Aggregation
- Association

Non-Hierarchical Relationship จะแสดงได้อย่างชัดเจนในรูปแบบของ Object relationship mapping นั่นคือ ความสัมพันธ์ในรูปแบบของ 1-1, 1-M, M-1 และ M-N



รูปที่ 1 ความสัมพันธ์ระหว่างวัตถุ

OPERATIONS ถูกกำหนดขึ้นมาสำหรับคลาส และสืบทอดผ่านทาง instance โดยแบ่งออกได้ 3 ทางดังนี้

- type specific vs generic
- predefined vs user-defined

Type specific operation คือ โอเปอเรชั่นที่ถูกประยุกต์ใช้กับ atomic value ในขณะที่ generic operation คือ โอเปอเรชั่นที่ประยุกต์ใช้กับ composite values

Predefined operation คือ โอเปอเรชั่นที่ถูกติดตั้งโดยนักพัฒนาระบบฐานข้อมูล ประยุกต์ใช้กับ atomic value บางชนิด User-defined operation คือ โอเปอเรชั่นที่ถูกนิยามโดยผู้ใช้

ลักษณะพื้นฐานของฐานข้อมูลเชิงวัตถุ ประกอบด้วย

COMPLEX OBJECTS (general and part-of) แต่ละทูเปิลนั้นประกอบด้วยแอททริบิวท์และแต่ละแอททริบิวท์มีค่าที่ประกอบด้วย composite element อื่นๆ

Composite objects ที่รู้จักกันดี คือ part-of composite object ซึ่งเป็นการอ้างถึงวัตถุตัวอื่นๆ หรือที่เรียกว่า Composite reference และโอเปอเรชั่นสำหรับ composite object คือ

- compose/decompose a composite objects
- copy a composite objects

USER-DEFINDED OPERATIONS ลักษณะที่แตกต่างของภาษาเชิงวัตถุ คือ การจัดเตรียมให้กับผู้ใช้ในการกำหนดคลาส และความสัมพันธ์ของโอเปอเรชั่น ซึ่งลักษณะนี้จะเป็นประโยชน์สำหรับคลาสได้

โครงสร้างการจัดการเชิงวัตถุนั้น โอเปอเรชั่นสามารถกำหนดโดยผู้ใช้ ซึ่งโอเปอเรชั่นเหล่านั้น คือ

- วิธีการต่างๆ ที่ซ่อนอยู่ภายใต้วัตถุ
- ผู้ใช้ได้กำหนดโอเปอเรเตอร์ขึ้นมาสำหรับ user-define type

OBJECT IDENTITY จะสนับสนุนในด้านความถูกต้องของข้อมูล นั่นคือ การเปลี่ยนแปลงใดๆ ที่เกิดขึ้นกับค่า หรือโครงสร้าง จะไม่ส่งผลกระทบกับตัวกำหนดวัตถุ ซึ่งเป็นตัวแสดงว่าข้อมูลเป็นอิสระต่อกัน

INHERIHANCE คือ การสืบทอด หรือกระบวนการในการจัดการคลาส เป็นการสร้างส่วนของคลาสใหม่ขึ้นมา โดยมีพื้นฐานมาจากคลาสเดิม แต่จะมี data หรือ member function ที่พิเศษเป็นของตัวเองเพิ่มขึ้นมาจากคลาสเดิม ซึ่งก่อให้เกิดการแตกสายพันธ์ของวัตถุเป็นวัตถุใหม่ที่มีคุณสมบัติของวัตถุเดิมซ่อนอยู่ไม่มากก็น้อย

- *Class และ Instance* การขยายความของคลาสจะนำเสนอโดยกลุ่มของ instance ที่มีคุณสมบัติเดียวกัน นั่นคือ ความสัมพันธ์ของคลาสแม่ (Superclass) และคลาสลูก (Subclass) จะถูกนำเสนอออกมา โดยผ่านทาง instance
- *Attributes* ซึ่งคลาสลูกจะสืบทอดแอททริบิวท์ต่างๆ จากคลาสแม่โดยตรง นั่นคือถ้ามีการเปลี่ยนแปลงใดๆ ที่คลาสแม่แล้ว คลาสลูกจะได้รับการเพิ่มแอททริบิวท์นั้นโดยตรง การสืบทอดแอททริบิวท์ทั้งชื่อ, ข้อจำกัด และความสัมพันธ์ต่างๆ ภายในแต่ละแอททริบิวท์ ซึ่งข้อจำกัดที่คลาสลูกได้รับนั้นอาจจะเป็นเพียงส่วนหนึ่งของคลาสแม่ หรือได้รับการถ่ายทอดมาทั้งหมด ซึ่งจะรวมทั้งค่า default value ด้วย
- *Relationships* การสืบทอดคุณสมบัติความสัมพันธ์จะรวมในส่วนของชื่อ, ชนิดของความสัมพันธ์, ข้อจำกัดต่างๆ และความสัมพันธ์ที่เกี่ยวข้องกับคลาสอื่นๆ
- *Operations* โอเปอเรชั่นจะถูกสืบทอดโดยคลาสลูก ซึ่งจะเกี่ยวข้องกับ
    1. คลาสลูกสามารถเพิ่มโอเปอเรชั่นเข้าไปได้ โดยอัตโนมัติ
    2. โอเปอเรชั่นของคลาสลูกสามารถเป็นส่วนขยายของคลาสแม่ได้

OBJECT PERSISTENCY เป็นทั้งคุณสมบัติพื้นฐาน และลักษณะเด่นของระบบฐานข้อมูลเชิงวัตถุ Persistency คือ ความสามารถของวัตถุที่จะสืบทอดชีวิตอยู่ภายในระบบในขณะที่ทำการประมวลผลโปรแกรม ดังรูปการต่อไปนี้

- Persistent Object จะไม่ถูกทำลายเมื่อโปรแกรมได้ถูกกำหนดจุดสิ้นสุดไว้ ซึ่งจะเกี่ยวข้องกับ internal garbage collection
- Persistent object จะประกอบด้วยชื่อที่เป็นสากล ซึ่งสามารถถูกอ้างถึงได้โดยโปรแกรมอื่นๆ

ผู้ใช้สามารถกำหนดคลาสของวัตถุที่เป็น persistent ในบางภาษาได้โดยการกำหนดในรูปแบบท้องถิ่นได้เท่าเทียมกับการกำหนดที่เป็นสากล

COMPUTATIONAL COMPLETENES ระบบจะมีการคำนวณที่สมบูรณ์ ถ้าระบบนั้นกำหนดการใช้ภาษาที่อนุญาติให้มีการเรียกใช้ฟังก์ชัน หรืออัลกอริทึมที่โค๊ดด้วย Data Manipulation Language (DML) ของระบบ

VERSION การเปลี่ยนแปลงของวัตถุจะเปลี่ยนรูปวัตถุจากสถานะนึงไปยังอีกสถานะอื่นๆ ซึ่ง version ก็คือกระบวนการในการเก็บผลลัพธ์ของวัตถุที่มีการเปลี่ยนแปลงในแต่ละสถานะอย่างรวดเร็ว คลาสที่อนุญาติให้มีการทำ version เรียกว่า version class และทุกๆ instance ที่อยู่ภายใต้คลาสนั้นจะเรียกว่า version instance ก็คือ การทำ version ให้กับคลาสที่มีความสัมพันธ์กัน การกำหนดให้คลาสเป็น version class จะต้องทำการกำหนดก่อน version instance ซึ่งการกำหนด version instance จะเกิดภายใต้ 3 สถานะคือ

- *Transient state* จะเกิดขึ้นเมื่อมีการเปลี่ยนแปลง และการลบแสดงออกมา ซึ่ง version instance ที่พิจารณานี้จะไม่คงทนเมื่ออยู่ในสถานะของ transient state
- *Working state* จะเกิดขึ้นเมื่อ version instance นั้นคงทน และไม่สามารถเปลี่ยนแปลงได้ แต่อย่างไรก็ตามการลบก็ยังสามารถกระทำได้
- *Released state* คือ version instance ที่กำหนดขึ้นในสถานะสุดท้าย ซึ่งจะมีความแข็งแรง (robust) และไม่สามารถทำการเปลี่ยนแปลงใดๆ ได้ทั้งการปรับปรุง และการลบวัตถุ

Version Instance ถูกนำเสนอจากสถานะนึงไปอีกสถานะนึง นั่นคือจาก transient ไปยัง working และไปสู่ released state และการนำเสนอการใช้งาน version instance กระทำได้ 2 วิธีการดังนี้

- การอ้างอิง version instance โดยตรงโดยการใช้ Object identifier
- การอ้างอิงกับ generic object และเรียกใช้งาน version ล่าสุด

ENCAPSULATION คือ การรวมกันของโครงสร้างของข้อมูลกับฟังก์ชันที่เกี่ยวข้อง หรือเรียกใช้ข้อมูลนั้น เกิดเป็นวัตถุตัวใหม่ที่สามารถทำการซ่อนข้อมูลจากภายนอกได้ ผลจากการนำโครงสร้าง และฟังก์ชันที่ใช้งานข้อมูลนั้นมารวมกันสร้างความสัมพันธ์ให้กันและกัน จะทำให้ข้อมูลมีความมั่นคงขึ้น ผลก็คือ ไม่มีการเข้าถึงข้อมูลโดยตรงอีกต่อไป นั่นคือเป็นการทำ Data Hiding

ซึ่งข้อดีของการทำ Encapsulation คือ เราสามารถแบ่งระดับของการเข้าถึงข้อมูลได้ด้วยตัว methods ของวัตถุเอง

POLYMORPHISM คือ การที่คลาสหนึ่งๆ สามารถแปรเปลี่ยนไปได้หลายรูปแบบ ขึ้นกับสภาพแวดล้อม หรือสถานการณ์ในขณะนั้น ความสามารถอีกอย่างหนึ่งของ Polymorphism ก็คือ สามารถกำหนดการดำเนินการใหม่ให้กับตัวดำเนินการ หรือแม้แต่ฟังก์ชันได้ เรียกว่าการทำ Overloading คือ การเรียกใช้ฟังก์ชัน หรือตัวดำเนินการเดิมให้ไปทำงานอีกอย่างหนึ่งได้

ข้อดีของรูปแบบข้อมูลเชิงวัตถุ [1]

Extensibility

สำหรับ user-defined types และ overloaded operations นั้น ส่วนของคลาส และการจัดการความสัมพันธ์จะสามารถปรับเปลี่ยน หรือเพิ่มได้ตามความต้องการ การเปลี่ยนแปลง หรือการเพิ่มนี้จะถูกจัดการให้กับคลาสที่มีความสัมพันธ์กันโดยอัตโนมัติ นั่นคือคลาส และโอเปอเรชันใหม่ๆ สามารถรวมเข้าไปในระบบ โดยปราศจากผลกระทบกับวัตถุอื่นๆ ภายในระบบ

Information Hiding

การซ่อนข้อมูลวัตถุจะอนุญาติให้ข้อมูล และโอเปอเรชันถูกกำหนดอยู่ภายในโครงสร้างข้อมูลเดี่ยว (Single model) นั่นคือ ข้อมูลจะถูกกำหนดเฉพาะที่ (local) และถูกซ่อนอยู่ภายในวัตถุเอง เมื่อมีการเชื่อมต่อเกิดขึ้นจะมีการแสดงออกมาโดยอัตโนมัติ

Flexibility of Type definition

ความสามารถที่อนุญาติให้ผู้ใช้กำหนดคลาส และโอเปอเรชันใหม่จะให้ความยืดหยุ่น และการควบคุมให้กับผู้ใช้ ซึ่งเป็นประโยชน์กับการประยุกต์ใช้ฐานข้อมูลใหม่ๆ ในการปรับเปลี่ยนข้อมูล

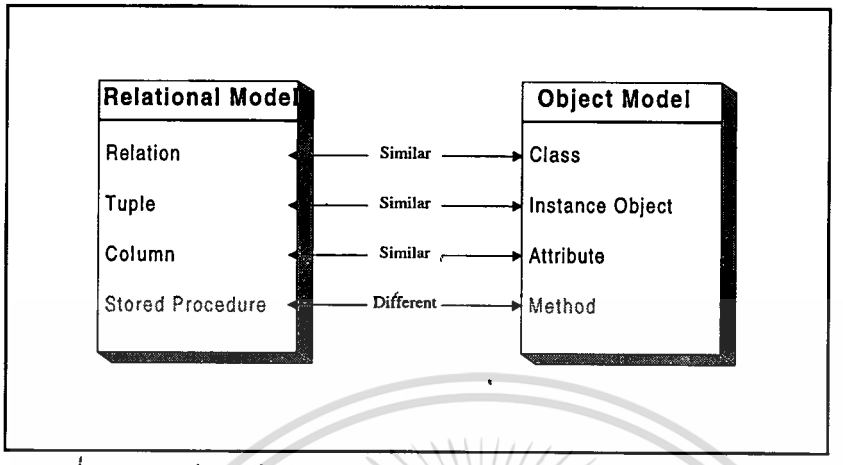
Code reusability

จากคุณสมบัติการถ่ายทอดจากคลาสแม่ไปยังคลาสลูก ทำให้ไม่ต้องมีการกำหนดวิธีการและโอเปอเรชันขึ้นใหม่ในแต่ละคลาส ซึ่งจะช่วยลดความซ้ำซ้อน และการทำงานที่ไม่ควบคู่กัน

การเปรียบเทียบระหว่าง Object-Oriented model **และ**Relational data model [2]

ข้อเปรียบเทียบของ Relational data model และ Object-Oriented data model สามารถสรุปได้ดังนี้ใน 2 ทิศทาง คือ

1. ความเหมือน และแตกต่างทางด้านรูปแบบ สามารถแสดงออกมาได้ดังรูป



รูปที่ 2 การเปรียบเทียบระหว่างฐานข้อมูลเชิงวัตถุ และเชิงสัมพันธ์

2. ความเหมือน และแตกต่างทางด้านแนวความคิด

- Data Type
- Data Integrity
- Data Manipulation

Data Type

ลักษณะของชนิดข้อมูลที่ใช้กับระบบฐานข้อมูลเชิงสัมพันธ์จะเป็นลักษณะที่เรียกว่า Built-In data type เช่น int, real, string เป็นต้น ซึ่งลักษณะของชนิดข้อมูลเหล่านี้ระบบจะเป็นผู้ทำการกำหนดให้ แต่สำหรับระบบฐานข้อมูลเชิงวัตถุนั้นชนิดของข้อมูลที่สามารถใช้งานได้จะเป็นทั้ง Built-In data type ชนิดของข้อมูลที่ผู้ใช้สามารถทำการกำหนดขึ้นมาเองได้ เช่น complex number เป็นต้น

Data Integrity

Relational system การทำงานลักษณะที่เรียกว่า referential integrity โดยการใช้การสื่อความหมายในการอ้างถึงโดยผ่าน foreign key แต่ในทางตรงกันข้ามระบบ OODB คลาสจะเป็นตัวบรรยายถึง data abstraction และรวมไปถึงการกำหนด specific ให้กับโอเปอเรชันต่างๆ ที่จะถูกประยุกต์ใช้กับ instances ของคลาส ซึ่งการกำหนดเช่นนี้จะทำให้ข้อมูลมีความเป็นอิสระต่อกันมากขึ้น นั่นคือเมื่อมีการเปลี่ยนแปลง หรือมีการพัฒนาระบบจะไม่มีส่งผลกระทบต่อคลาส หรือกระบวนการทำงานอื่นๆ ซึ่งส่งผลให้ข้อมูลมีความถูกต้อง (data integrity)

Data Manipulation

การจัดการข้อมูลสามารถเปลี่ยนรูปไปอยู่ในรูปแบบของ query processing system ได้ ซึ่งระบบนี้จะต้องให้ความสะดวกต่อการสร้าง, การลบ, การปรับเปลี่ยน และการเข้าถึงข้อมูลที่จำเป็นได้ สำหรับระบบ relational DB นั้น query language จะให้แนวความคิดในการจัดการข้อมูลที่ต่างกันไป

SQL2 นั้นจะมีพื้นฐานมาจาก relational calculus สำหรับการจัดการข้อมูลใน OODB นั้นกระทำโดยการเชื่อมต่อคลาส และผ่านโครงสร้างของภาษาโปรแกรมที่ล้อมรอบด้วยการส่งไปยังคลาส นั่นคือการส่งข้อความ (message) ไปยังวัตถุที่ทำงานด้วย ซึ่ง sql ที่ทำงานกับ OODB คือ SQL3 ซึ่งสนับสนุนการทำงานในส่วนของ Object ด้วย

บทที่ 3

POSTGRESQL

3.1 บทนำ

PostgreSQL ได้รับการพัฒนามาจาก Postgres (version 4.2)[5] ภาษาที่ใช้ในการพัฒนาโปรแกรมคือ ANSI C โดยทำการเปลี่ยนแปลงจากเดิมประมาณ 25% ในการเปลี่ยนแปลงโครงสร้างภายในเพื่อเพิ่มประสิทธิภาพในการทำงานและความง่ายในการปรับเปลี่ยนในอนาคต สิ่งที่ได้รับการพัฒนาขึ้นจากเดิมคือ

- ภาษาที่ใช้ในการสอบถามเกี่ยวกับข้อมูลต่างๆ จากเดิมคือ POSTQUEL ได้รับการเปลี่ยนแปลงเป็น SQL มีการสนับสนุนการใช้คำสั่ง GROUP BY และทำการพัฒนาในส่วนของ Aggregation เพิ่มขึ้น
- สนับสนุนการใช้งานในส่วนของ Interface ในการใช้ GNU ในการทำงานในรูปของ PSQL
- เพิ่มชุดคำสั่งของการทำงานในส่วนของ Front-end, libpgtcl ที่สนับสนุนการทำงานในส่วนของการเชื่อมต่อกับ Clients ซึ่งจะมี shell ที่เป็นส่วนของคำสั่งต่างๆ ที่ใช้ในการเชื่อมต่อเข้ากับส่วนของ Postgresql backend

รูปแบบการทำงานของ Postgres จะอยู่ในรูปของ "process-per-user" โดยใช้แบบจำลองของ client/server ซึ่งจะประกอบด้วยโปรแกรมการจัดการต่างๆ ของ UNIX คือ

- Supervisory daemon process (Postmaster)
- User's frontend application
- Backend database servers

Postmaster จะทำหน้าที่ในการจัดการชุดของคำสั่งทางฐานข้อมูลบน host ซึ่งชุดของคำสั่งเหล่านั้นเรียกว่า installation หรือ site การที่ Frontend applications จะทำการติดต่อเพื่อเข้าไปขอใช้งานฐานข้อมูลจะเรียกโปรแกรมในส่วนของ LIBPQ library ขึ้นมาใช้งาน หลังจากนั้น library

จะส่งคำร้องขอของผู้ใช้ผ่านเครือข่ายไปยัง Postmaster ซึ่งเป็นการเริ่มต้นการติดต่อในส่วนของ backend process แล้วจึงทำการเชื่อมต่อไปยัง server ตัวใหม่ จากขบวนการที่เกิดขึ้นส่วนของ frontend process และ backend server จะทำการติดต่อสื่อสารโดยปราศจากการขัดขวางจากส่วนของ Postmaster ดังนั้นการทำงานของ postmaster จะทำงานอยู่ตลอดเวลา, ทำการรอการส่งคำร้องขอ, ในขณะที่ส่วนของ frontend และ backend process จะถูกส่งเข้ามาและรอรับผลลัพท์ที่ได้จากการประมวลผล LIBPQ library จะอนุญาติให้ frontend 1 ตัว ทำการเชื่อมต่อกับหลายๆ backend process แต่อย่างไรก็ตามส่วนของ frontend application จะยังคงเป็นแบบ single-threaded process ซึ่งการเชื่อมต่อในลักษณะ multithread frontend/backend process ยังไม่สามารถใช้งานได้ในส่วนของ LIBPQ

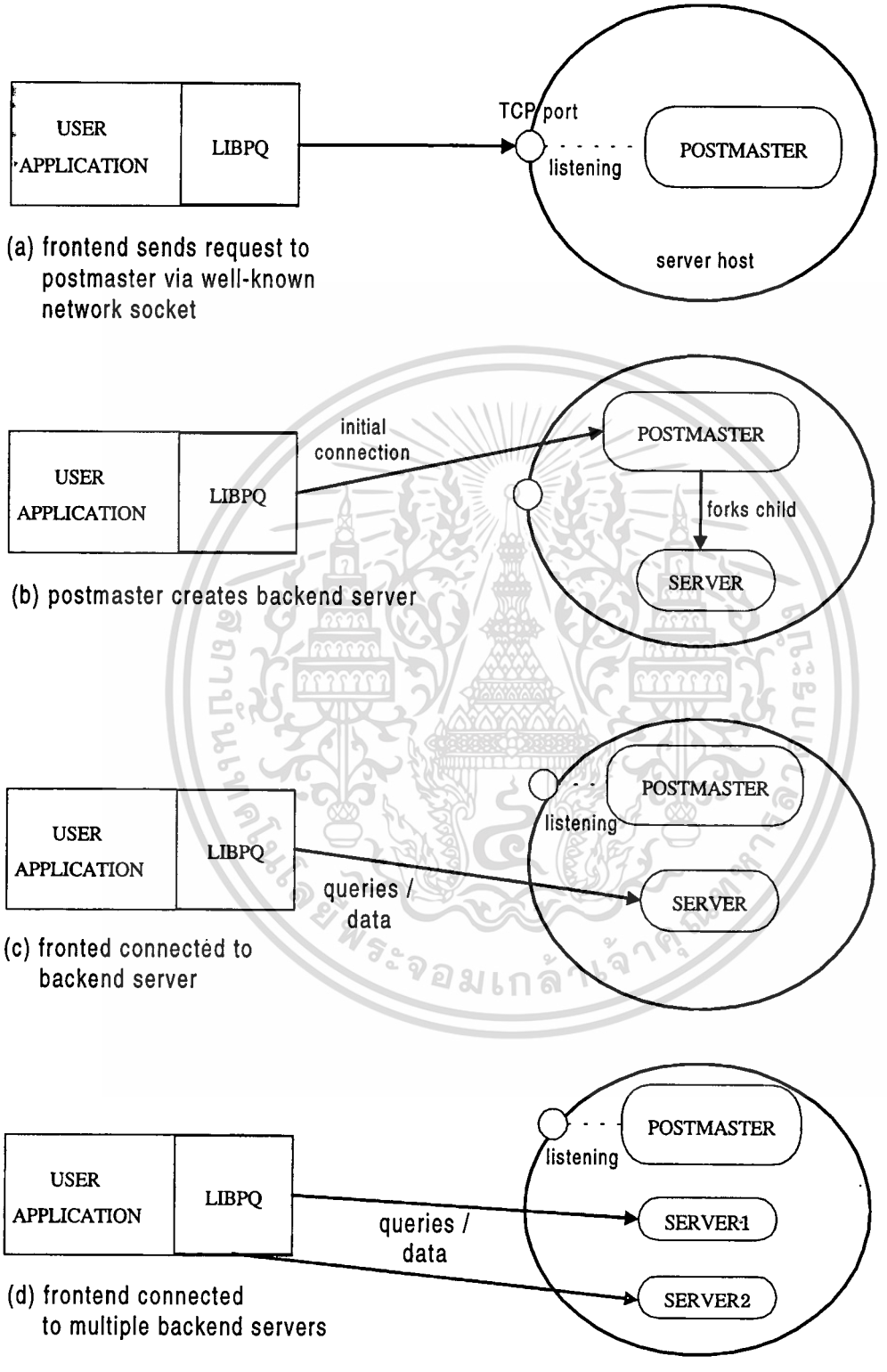
จากโครงสร้างการทำงานของ Postgres นั้นแสดงว่า การทำงานในส่วนของ postmaster และ backend จะทำงานอยู่ในเครื่องเดียวกัน (database server) ในขณะที่ส่วนของ frontend application นั้นในที่ใดๆ ก็ได้

3.2 General Overview

เราสามารถแบ่งสภาพแวดล้อมการทำงานของ Postgres ได้ 3 ส่วนดังนี้ คือ

- Client Applications
- The Postmaster
- Postgres Server Backends

Client Applications คือ โปรแกรมที่ทำหน้าส่ง และรับข้อมูลจากตัว Postgres Backend โดยทั่วๆ ไป Client Applications คือ Applications ที่ทำหน้าที่ในส่วนติดต่อกับผู้ใช้งาน ซึ่ง Postgres จะสบับสนุนการทำงานในรูปแบบเชิงสถาปัตยกรรมแบบ Client/Server เมื่อตัว clients สามารถทำงานบนเครื่องอื่นๆ ที่แยกจากส่วนของ database backend ได้ โดยที่ Client Applications จะทำหน้าที่ในการติดต่อสื่อสารกับส่วนของ Database Server โดยทาง Postgres Client Protocol



รูปที่ 3 โครงสร้างการทำงานของ POSTMASTER

The Postmaster จะทำหน้าที่ในการติดต่อทางด้านเครือข่ายให้กับระบบทั้งในด้านการเริ่มต้นการติดตั้งการเชื่อมต่อ และขบวนการต่างๆ ในการเชื่อมต่อ The Postmaster จะรอรับสัญญาณร้องขอการเชื่อมต่อจากตัว client ที่ส่งผ่านมาทางพอร์ท จากนั้นจะจำลองการทำงานของ Postgres ที่ส่วนของ server backend หลังจากเสร็จขบวนการในการเชื่อมต่อแล้ว Postmaster จะไม่เข้ามาเกี่ยวข้องกับการทำงานอีก แต่มันจะกลับไปรอรับสัญญาณที่ส่งเข้ามาใหม่ ซึ่งในการทำงานในส่วนที่เหลือ client จะเป็นผู้ทำหน้าที่ติดต่อโดยตรงกับส่วนของ backend

เมื่อ Postmaster เริ่มต้นการทำงาน มันจะเริ่มการกำหนดพื้นที่ในหน่วยความจำและเซมาฟอร์ที่ถูกใช้สำหรับ Postgres backend สำหรับการทำ locking โดยที่ Postmaster และ backend จะต้องทำงานอยู่บนเครื่องเดียวกันเท่านั้น

Postgres Server Backends (backend) คือ ส่วนที่ทำหน้าที่ในการจัดการทางด้านฐานข้อมูลโดยตรง เมื่อตัวมันถูกจำลองให้มีการทำงานเกิดขึ้นโดยส่วนของ Postmaster แล้ว มันจึงพร้อมที่จะรับคำร้องขอในการทำงานกับส่วนของข้อมูลจาก client และส่งในส่วนของผลลัพธ์กลับไปที่ client applications ส่วนของ client application สามารถเชื่อมต่อกับส่วนของ backend ได้หลายส่วนพร้อมๆ กัน แต่โดยปกติแล้วส่วนของ backend นั้นจะทำการเชื่อมต่อกับส่วนของ client application ได้เพียงส่วนเดียวในเวลาใดเวลาหนึ่งเท่านั้น

3.3 Postgres Main Modules

ส่วนของ Postgres backend นั้นประกอบด้วยส่วนของการทำงานหลายส่วน แต่ละโมดูลจะส่งผลการทำงานกับส่วนกลางของระบบ ซึ่งในการทำงานของ Postgres นั้นแต่ละส่วนยังมีการทำงานที่ขึ้นตรงต่อกันซึ่งในส่วนนี้ยังเป็นข้อเสียของ Postgres95 อยู่

ส่วนของ backend ประกอบด้วยโมดูลหลักๆ ดังนี้

- Parser
- Query Rewrite Rule System
- Planner / Optimizer
- Executor
- Access Methods
- Storage Manager
- Utilities

3.4 Internal Data Structures

Postgres โพรซีเดอร์ ส่วนมากจะทำงานกับกลุ่มโครงสร้างข้อมูลภายใน ที่เรียกว่า โหนด (nodes) โหนดของระบบจะถูกกำหนดอยู่ภายใต้ไดเรคเทอรี่ /src/backend/nodes ในรูปของ pseudo-object-oriented

Postgres Types system [4]

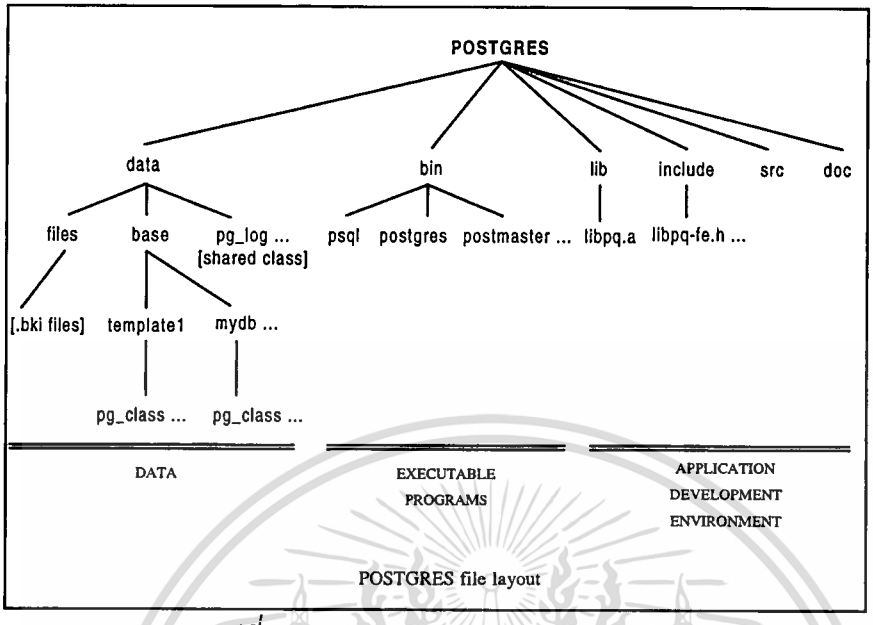
Postgres Types system สามาถแบ่งออกได้เป็น 2 ประเภท คือ

- Base types เช่น int4 ที่สามารถพัฒนาผ่านการใช้งานภาษา C ได้ หรือที่เรียกกันว่า Abstract Data Types ซึ่ง postgres จะทำงานกับ types เหล่านั้นผ่านทาง methods ที่กำหนดโดยผู้ใช้ และจะเข้าใจพฤติกรรมของข้อมูลจากการที่ผู้ใช้เป็นผู้กำหนดเท่านั้น
- Composite types จะถูกสร้างขึ้นมาเมื่อผู้ใช้ทำการสร้างคลาสขึ้นมา Postgres จะทำการเก็บ types เหล่านี้ได้เพียงทางเดียว คือ ภายในไฟล์ที่เก็บอินสแตนท์ของคลาส แต่ผู้ใช้สามารถมองผ่านข้อมูลโดยการใช้ภาษาในการสอบถามข้อมูลเท่านั้น

3.5 Function Manager

Postgres Function Manager จะทำหน้าที่คอยจัดการคำร้องขอที่มาจากหลายๆ ชนิดของฟังก์ชัน ความยืดหยุ่นในการทำงานของ Postgres คือ ความสามารถในการใช้งานของผู้ใช้ในการกำหนด และเพิ่มหน้าที่หรือฟังก์ชันในการทำงานใหม่ๆ ได้เอง Function Manager จะจัดการทำงานกับทั้งฟังก์ชันที่ผู้ใช้เป็นผู้กำหนดเอง และฟังก์ชันที่ฝังอยู่ภายในระบบ ซึ่งฟังก์ชันที่ใช้งานใน Postgres จะแบ่งได้เป็น 2 กลุ่ม คือ ฟังก์ชันที่ฝังติดมากับระบบ และฟังก์ชันที่เป็นภาษา C หรือภาษา SQL แต่ละฟังก์ชันจะถูกนำเสนอโดยการใช้ Object ID หรือที่เรียกว่า Procedure ID

ฟังก์ชันที่ถูกฝังอยู่ภายในระบบ (built-in functions) คือ ชุดของกลุ่มฟังก์ชันภาษา C ที่ใช้ในการแปลงภาษาโปรแกรม



รูปที่ 4 POSTGRES File Layout

บทที่ 4

การใช้งาน POSTGRESQL

4.1 Embedded SQL and Precompiler in Postgres

Embedded SQL มีข้อดีบางอย่างที่นอกเหนือการใช้งาน SQL โดยทั่วไป คือ การที่มันสามารถจัดการการเคลื่อนย้ายข้อมูลเข้าและออกโดยผ่านตัวแปรที่ใช้ในภาษา C จะเป็นแบบ embedd SQL จะใช้มาตรฐานของ ANSI ในการทำงาน preprocessor ที่ใช้กับ Embedded SQL บนภาษา C สำหรับ Postgres95 คือ Ecpg

ในการเขียนโปรแกรมภาษา C ที่มีการใช้งาน SQL ด้วยนั้นการกำหนดค่าของตัวแปรที่ใช้งานจำเป็นจะต้องมีการใช้คำสั่งพิเศษเพื่อเป็นการกำหนดตำแหน่งการเริ่มต้นการใช้งาน ก่อนที่จะมีการคอมไพล์โปรแกรมจะต้องมีการรันไฟล์ผ่านทาง Embedded SQL C Processor และทำการแปลงคำสั่งที่ใช้ภาษา SQL ให้อยู่ในรูปของฟังก์ชันที่มีการผ่านค่าตัวแปรในรูปของอาร์กิวเมนต์เมื่อมีการเรียกใช้งาน ซึ่งตัวแปรทั้ง 2 ตัวที่มีการใช้งาน คือ ตัวแปรที่ใช้ในการเก็บค่าอินพุท และตัวแปรที่ใช้ในการเก็บค่าเอาท์พุท

เมื่อถึงเวลาคอมไพล์ และลิงค์โปรแกรม จะต้องทำการลิงค์ไปยังไลบรารี่ที่เก็บฟังก์ชันต่างๆ ในการใช้งาน ฟังก์ชันต่างๆ เหล่านี้จะรับค่าจากอาร์กิวเมนต์ และทำงานในส่วนของ SQLโดยผ่านการเชื่อมต่อ (pq) และใส่ค่าผลลัพธ์ที่ได้กลับไปยังในส่วนของอาร์กิวเมนต์ที่ใช้ในการรับค่าผลลัพธ์ และเมื่อใดที่ทำการรันโปรแกรม ในส่วนของ SQL จะถูกส่งไปยังในส่วนของฐานข้อมูลเพื่อไปดึงข้อมูลขึ้นมา และทำงานในส่วนอื่นต่อไป

Preprocessor ที่ใช้ในงานในระบบ คือ ECPG หลังจากที่ทำการอินสตอลล์ระบบ มันจะรับค่าผ่านทางอาร์กิวเมนท์ 2 ตัว คือ iname=filename และ oname=filename ซึ่งทั้งสองตัวจะต้องมีการแสดงค่า หรือไม่ก็แสดงข้อผิดพลาดที่เกิดขึ้นได้

Library ที่ใช้งานในระบบ คือ libecpg.a ซึ่งไลบรารี่จะเรียกใช้ pq library ในการติดต่อสื่อสารไปยัง Postgres server คำสั่งหรือสัญลักษณ์พิเศษที่ใช้ คือ EXEC ที่เป็นตัวเรียกคำสั่งต่างๆ ของ SQL ในการทำงาน ซึ่งการใช้งานคำสั่ง SQL จะอยู่ระหว่างเครื่องหมาย “exec sql” และเครื่องหมาย “;”

4.2 SQL3

ลักษณะการทำงานของ SQL3 จะแตกต่างไปจากการใช้งานในส่วนของ SQL92 ตรงที่จะสนับสนุนการทำงานในส่วนของ Object-Oriented หลายส่วน เช่น Inheritance, Time Interval, Create Function, Create Type, Create Operators เป็นต้น

Inheritance

การสร้าง class 2 class ที่ สัมพันธ์ กันระหว่าง Superclass และ Subclass ซึ่งจะมีการถ่ายทอดกันในส่วนต่างๆ เช่น attributes, relationship เป็นต้น ซึ่งเราสามารถกระทำได้ดังนี้

```
CREATE TABLE cities (
        name            text,
        population      float,
        altitude        int
);
CREATE TABLE capitals (
        state           char2
) INHERITS (cities);
```

ซึ่งในกรณีนี้ class capitals จะสืบทอดในส่วนของ attributes ต่างๆ ของ parents มาทั้งหมด ซึ่งก็คือ class cities ซึ่งตัวอย่างการใช้งานมีดังนี้

```
SELECT name, altitude
FROM cities
WHERE altitude > 500;
```

name	altitude
Las Vegas	2174
Mariposa	1953

แต่ถ้าในกรณีที่ต้องการค้นหาข้อมูลที่ class อื่นๆ ที่เป็น subclass เราสามารถกำหนดได้โดยใช้คำสั่งดังนี้

```
SELECT c.name, c.altitude
FROM cities* c
WHERE c.altitude > 500;
```

name	altitude
Las Vegas	2174
Mariposa	1953
Madison	845

Time Travel

Postgres สนับสนุนการสอบถามเกี่ยวกับข้อมูลที่อยู่ในรูปของ historical ได้ ซึ่งในตัวอย่างที่ทำงานนี้เราจะทำการสอบถามข้อมูลที่เกี่ยวกับ population ของ Mariposa city ดังนี้

```
SELECT * FROM cities WHERE name = 'Mariposa';
```

name	population	altitude
Mariposa	1320	1953

ซึ่ง Postgres จะทำการค้นหาข้อมูลที่เป็นข้อมูลปัจจุบัน และจะสามารถหาช่วงเวลาที่ได้มีการเปลี่ยนแปลงของข้อมูลได้ ดังนี้

```
SELECT name, population
FROM cities['epoch','now']
WHERE name = 'Mariposa';
```

ซึ่ง "epoch" นั้นจะเป็นตัวบ่งชี้ว่าให้ทำการแสดงตั้งแต่จุดเริ่มต้นเวลาที่ทำการกำหนด และถ้าหากเคยทำการประมวลผลการทำงานไปแล้วผลที่ได้จะเป็นดังนี้

name	population
Mariposa	1200
Mariposa	1320

เมื่อข้อมูลในส่วนแรกที่แสดงเป็นข้อมูลที่ได้เคยทำการประมวลผลไปแล้ว และข้อมูลที่แสดงในส่วนหลังเป็นข้อมูลที่แสดงอยู่ในปัจจุบัน

System Catalogs

Postgres สามารถขยายระบบได้โดยง่าย เพราะส่วนของโอเปอเรชั่นนั้น คือ catalog-driven ซึ่งจะเหมือนกับในระบบฐานข้อมูลเชิงสัมพันธ์ที่มี system catalog ทำการเก็บในส่วนรายละเอียดของฐานข้อมูล เช่น ตาราง, คอลัมน์, ... (ในบางระบบเรียกว่า data dictionary) ส่วนของ system catalog จะแสดงอยู่ในรูปของคลาสเหมือนกับส่วนอื่นๆ แต่ข้อแตกต่างหลักของ system catalog ในระบบฐานข้อมูลเชิงสัมพันธ์ และ postgres คือ ใน Postgres จะเก็บข้อมูลมากกว่าไม่ใช่แค่ข้อมูลที่เกี่ยวกับตารางและคอลัมน์เท่านั้น แต่ยังเก็บรายละเอียดในส่วนของชนิดของข้อมูล, หน้าที่การทำงาน, วิธีการเข้าถึงข้อมูล, และส่วนอื่นๆ อีก คลาสเหล่านี้สามารถดัดแปลงได้เองโดยผู้ใช้ ซึ่งจะแตกต่างกับระบบฐานข้อมูลโดยทั่วๆ ไปที่การดัดแปลง หรือขยายระบบนั้นสามารถกระทำได้เฉพาะผู้ค้าของ DBMS เท่านั้น

ส่วนของ system catalogs ที่สำคัญมีรายละเอียดดังนี้

Catalog Name	Description
pg_aggregate	aggregates and aggregate functiond
pg_am	access methods
pg_amop	access method operators
pg_amproc	access method support functions
pg_attribute	class attributes
pg_class	classes
pg_database	database
pg_defaults	(not used)
pg_demon	(not used)
pg_group	user groups
pg_hosts	host based access control (not used)
pg_index	secondary indics
pg_inheritproc	(not used)
pg_inherits	class inheritance hierarchy

pg_ipl	inheritance precedence list
pg_language	programming language recognized by the system
pg_listener	asynchronous notification
pg_log	transaction status log
pg_magic	(not used)
pg_opclass	operator classes
pg_operator	operator
pg_parg	(not used)
pg_proc	procedures (both C and SQL)
pg_rewrite	rewrite rule system
pg_server	(not used)
pg_statistic	high/low attribute keys (currently not filled)
pg_time	transaction commit times
pg_type	types (both base and complex)
pg_user	database users
pg_variable	oid counter
pg_version	version maintenance

ตารางที่ 1 System Catalog

EXTENDING SQL : FUNCTIONS

Create Function by use Query Language (SQL)

- SQL Function on Base Types

  With no argument

  CREATE FUNCTION one() RETURNS int4

  AS ' SELECT 1 as RESULT ' LANGUAGE 'sql';

```
SELECT one() AS answer;
```

answer
1

With argument

```
CREATE FUNCTION add_em(int4, int4) RETURNS int4
        AS ' SELECT $1 + $2;' LANGUAGE 'sql';

SELECT add_em(1, 2) AS answer;
```

answer
3

- **SQL Function on Composite Types**

ในกรณีที่เราต้องการทำงานกับข้อมูลที่อยู่ในตาราง เช่น ตาราง EMP เราไม่ได้กำหนดว่า attribute ตัวใดที่เราต้องการ แต่เป็นการทำงานกับ attributes ของ argument ตัวนั้น เช่น ตัวอย่างการทำงานที่เพิ่มอัตราเงินเดือนเป็นสองเท่า

```
CREATE FUNCTION double_salary(EMP) RETURNS int4
        AS ' SELECT $1.salary * 2 AS salary;' LANGUAGE 'sql';
SELECT name, double_salary(EMP) AS dream
FROM EMP
WHERE EMP.dept = 'toy';
```

name	dream
Sam	2400

การกำหนดฟังก์ชันนั้นจะมีข้อจำกัดหลายประการ ซึ่งจะกล่าวดังต่อไปนี้

- รายการที่จะทำการแสดงนั้นจะต้องเหมือนกับที่กำหนดไว้ในส่วนคำสั่งของการ CREATE TABLE
- จะต้องระวังในการเลือกใช้ส่วนขยายของคำสั่ง เพราะถ้าเกิดข้อผิดพลาดระบบจะแสดงข้อความดังนี้

  WARN::function declared to return type EMP does not retrieve (EMP.*)

- เมื่อใดที่มีการเรียกใช้ฟังก์ชันที่มีการส่งคืนผลลัพธ์ที่เป็นค่าคงที่ เราจะไม่สามารถทำการเรียกข้อมูลที่เป็น instance ได้ ซึ่งเราจะต้อง project ในส่วนของ attribute ออกมาจาก instance หรือทำการเรียกส่งค่าในส่วนของ instance ไปยังฟังก์ชันการทำงานอื่นๆ

  SELECT name(new_emp()) AS nobody;

nobody
None

- จากเหตุผลที่ว่าทำไมเราจะต้องทำการ project ค่าผ่านการเรียกใช้ฟังก์ชัน ก็คือ การที่เครื่องไม่เข้าใจความหมายของตัว (dot) ที่ฝังอยู่ในฟังก์ชัน

  SELECT new_emp().name AS nobody;

  WARN:parser: ssntax error at or near "."

Create Function by use Programming Language

- Programming Language Functions on Base Types

  Base type ที่ใช้งานกันอยู่จะมีอยู่ 3 รูปแบบ คือ

  1. pass by value, fixed-length
  2. pass by reference, fixed-length
  3. pass by reference, variable-length

สำหรับ By-value type นั้นควรจะมีความยาว 1, 2 หรือ 4 ไบท์ ซึ่งตัว Postgres นั้นจะทำการส่งผ่านค่าข้อมูลที่เป็นตัวเลขโดยการผ่านค่าไป ควรระวังในเรื่องการกำหนดประเภทต่างๆ ว่ามีขนาดที่ตรงกันหรือไม่กับที่ระบบสามารถรองรับการทำงานได้ เช่น การกำหนดเป็น long นั้นจะเป็น

อันตรายเพราะว่าบางระบบกำหนดให้เป็น 4 ไบท์ แต่บางระบบกำหนดให้เป็น 8 ไบท์ ในขณะที่เลขจำนวนเต็มในระบบยูนิกซ์นั้นคือ 4 ไบท์ ซึ่งสามารถกำหนดได้ดังนี้

```
/* 4-byte integer, passed by value */
typedef int int4;
```

แต่สำหรับการกำหนดที่เป็นแบบ fixed-length นั้นจะมีการผ่านค่าทาง by-reference เช่น การกำหนดให้เป็น char16 คือ

```
/* 16-byte structure, passed by reference */
typedef struct {
        char data[16];
} char16;
```

และสำหรับ variable-length จะต้องมีการส่งผ่านค่าการทำงานในลักษณะของ by-reference ซึ่งในส่วนนี้การกำหนดค่าจะต้องเริ่มต้นด้วยการกำหนดให้ความกว้างของฟิลด์เป็นจำนวนเต็มแบบ 4 ไบท์ และข้อมูลในทุกๆ ตัวจะถูกจัดเก็บอยู่ภายในที่สามารถทำการโหลดเข้าสู่ส่วนของหน่วยความจำได้ตามความกว้างของฟิลด์

```
typedef struct {
        int4 length;
        char data[16];
} text;
```

ตัวอย่างการใช้ฟังก์ชันภายในภาษา C คือ

```
#include <string.h>
#include "postgres.h"   /* for char16, etc. */
#include "utils/palloc.h"   /* for alloc */

int
add_one(int arg)
{
```

```
        return(arg + 1);
}

char16 *
concat16(char16 *arg1, char16 *arg2)
{
        char16 *new_c16 = (char16 *) palloc(sizeof(char16));
        memset((void *) new_c16, 0, sizeof(char16));
        (void) strncpy(new_c16, arg1, 16);
        retrn(char16 *)(strncat(new_c16, arg12, 16));
}

text *
copytext (text *t)
{
        text *new_t = (text *) palloc(VARSIZE(t));
        memset(new_t, 0, VARSIZE(t));
        VARSIZE(new_t) = VARSIZE(t);

        memcpy((void *) VARDATA(new_t),
                (void *) VARDATA(t),
                VARSIZE(t) -VARHDRSZ);
        return(new_t);
}
```

- **Programming Language Functions on Base Types**

instance ที่เป็น composite type นั้นบางทีอาจจะประกอบด้วยฟิลด์ที่มีค่าเป็น null ได้ และในส่วนของ composite type นั้นบางทีจะได้รับการสืบทอดผ่านมาจากคลาสที่เหนือกว่า ซึ่ง

อาจจะมีฟิลด์ที่ต่างกันไป ในส่วนของ Postgres นั้นจะทำงานกับชุดของข้อมูลที่เป็น instance โดยการผ่านค่าไปทางฟังก์ชันในรูปของทูเปิล ดังนี้

```
SELECT name, c_overpaid(EMP, 1500) AS overpaid
FROM EMP
WHERE name = 'Bill' or name = 'Sam';
```

ซึ่งในส่วนของการทำงานกับภาษา c คือ

```
#include "postgres.h" /* for char16, etc. */
#include "libpq.-fe.h" /* for tuple */
bool
c_overpaid(TUPLE t, /* the current instance of EMP */
        int4 limit)
{
    bool isnull = false;
    int4 salary;

    salary = (int4) GetAttributeByName(t, "salary", &isnull);
    if (isnull)
        return (flase);
    return (salary > limit);
}
```

Create Function by User-Defined Type

ในการกำหนด type นั้นจะต้องมีส่วนของฟังก์ชันที่ใช้ในการรับค่า และแสดงผลลัพธ์ ซึ่งฟังก์ชันเหล่านี้จะต้องทำการกำหนดทั้งในส่วนหน่วยความจำด้วย เช่นตัวอย่างของการกำหนดฟังก์ชันที่เป็น complex type ซึ่งนำเสนอในเรื่องของ complex number

```
typedef struct Complex {
    double      x;
    double      y;
} complex;
```

ซึ่งหลักในการกำหนดฟังก์ชันนั้นจะต้องระลึกถึงในส่วนดังนี้

- การกำหนดในภาษา C นั้นจะต้องมีการกำหนดให้มีความสมบูรณ์

```
complex *
complex_in (char *str)
{
        double x, y;
        Complex *result;

        if (sscanf(str, "(%1f, %1f)", &x, &y) != 2) {
                elog(WARN, "Complex_in: error  in parsing return
                NULL;
        }
        result = (Complex *)palloc(sizeof(Complex));
        result->x = x;
        result->y = y;
        return(result);
}
```

Output Function :

```
char *
complex_out(Complex *complex)
{
        char *result;

        if (complex == NULL)
                return(NULL);
        result = (char *) palloc(60);
```

```
        sprintf(result, "(%g,%g)",complex->x, complex->y);
        return(result);
    }
```

- การกำหนดทั้งในส่วนฟังก์ชันของการป้อนข้อมูล และส่วนของการแสดงผลลัพธ์ นั้นควรจะทำการแยกการทำงานออกจากกัน เพราะอาจจะเกิดปัญหาขึ้นในกรณีที่มีการเปิดไฟล์ที่ใช้ในการรับข้อมูล และแสดงผลลัพธ์ขึ้น

เมื่อต้องการกำหนดฟังก์ชันในการทำงานจะต้องทำการกำหนดดังนี้

```
CREATE FUNCTION complex_in(opaque)
        RETURNS complex
        AS '/usr/local/postgres95/tutorial/obj/complex.so'
        LANGUAGE 'c';

CREATE FUNCTION complex_out(opaque)
        RETURNS opaque
        AS '/usr/local/postgres95/tutorial/obj/complex.so'
        LANGUAGE 'c';

CREATE TYPE complex (
        internallength = 16,
        input = complex_in,
        output = complex_out
);
```

Create Operators

ส่วนการทำงานของ Postgres จะสนับสนุนการใช้ left unary, right unary และ binary operators ซึ่งในส่วนของ operators เหล่านี้สามารถนำมาใช้งานได้ใหม่ได้ หรือใช้งานในกรณีที่

เป็น overloaded ได้ การสร้าง operators ใหม่ขึ้นมาสำหรับการใช้งานกับ complex numbers นั้น กระทำได้ดังนี้

```
CREATE FUNCTION complex_add(complex, complex)
        RETURNS complex
        AS '$PWD/obj/complex.so'
        LANGUAGE 'c';

CREATE OPERATOR + (
        leftarg = complex,
        rightarg = complex,
        procedure = complex_add,
        commutator = +
);
```

การใช้งานฟังก์ชันที่กำหนดขึ้นมานั้นสามารถกระทำได้ดังนี้

```
SELECT (a+b) AS c FROM test_complex;
```

c
(5.2, 6.05)
(133.42,144.95)

4.3 LIBPQ

LIBPQ คือ โปรแกรมประยุกต์ที่ใช้ในการเชื่อมต่อกับ Postgres ซึ่ง LIBPQ คือ กลุ่มของไลบรารีรูทีน ที่อนุญาติให้โปรแกรมในส่วนของไคลเอนต์ทำการส่งผ่านการสอบถามข้อมูลไปยัง Postgres backend server และทำการรับค่าผลลัพธ์ที่ได้จากการสอบถามข้อมูลอีกที

ฟรอนท์เอนต์โปรแกรมที่ใช้งานในส่วนของ LIBPQ จะต้องมีการเรียกใช้งานไลบรารีโดยการกำหนดในส่วนหัวของโปรแกรมเป็น libpq-fe.h และทำการเชื่อมต่อกับไลบรารี libpq ด้วย

1. การควบคุม และการกำหนดการเริ่มต้นการใช้งาน (Control and Initialization)
   - PGHOST sets the default name server
   - PGOPTIONS sets additional runtime options for the POSTGRES backend
   - PGPORT sets the default port for the communicating with POSTGRES backend
   - PGTTY sets the file or tty on which debugging messages from the backend server are displayed.
   - PGDATABASE sets the default POSTGRES database name.

2. ฟังก์ชันที่ใช้ในการเชื่อมต่อกับฐานข้อมูล (Database Connection Functions) ประกอบด้วยรูทีนต่างในการเชื่อมต่อกับส่วนของ backend ดังนี้
   - PQsetdb เพื่อทำการสร้างการเชื่อมต่อใหม่ไปยัง backend

```
Pgconn *PQsetdb(char *pghost,
    char *pgport,
    char *pgoptions,
    char *pgtty,
    char *dbname);
```

ถ้าอาร์กิวเมนท์ทุกตัวมีค่าว่าง ดังนั้นจะมีการตรวจสอบค่าในตัวแปรของสภาพแวดล้อมของระบบ แต่ถ้าไม่มีการกำหนดค่าในตัวแปรของสภาพแวดล้อมเอาไว้ ดังนั้นจะถูกกำหนดให้ใช้ค่าดีฟอล์ทโดยอัตโนมัติ

PQsetdb จะส่งกลับค่าตัวชี้ของ Pgconn เสมอ และคำสั่ง PQstatus จะถูกเรียกใช้เพื่อให้แน่ใจว่าในการติดต่อสื่อสารได้ถูกกระทำก่อนที่จะมีการส่งการร้องขอการสอบถามข้อมูลผ่านทางการเชื่อมต่อ

- PQdb จะส่งกลับชื่อของฐานข้อมูลที่ใช้ในการติดต่อ

  char *PQdb(PGconn *conn)

- PQhost จะส่งกลับชื่อของโฮสต์ที่ใช้ในการเชื่อมต่อ

  char *PQhost(PGconn *conn)

- PQoptions จะส่งกลับค่าของออพชันที่ใช้ในการเชื่อมต่อ

  char *PQoptions(PGconn *conn)

- PQport จะส่งค่าพอร์ทที่ใช้ในการเชื่อมต่อ

  char *PQport(PGconn *conn)

- PQtty จะส่งกลับค่าของ pgtty ที่ใช้ในการเชื่อมต่อ

  char *PQtty(PGconn *conn)

- PQstatus จะส่งกลับค่าของสถานะของการเชื่อมต่อ ซึ่งสถานะนั้นอาจจะเป็น CONNECTION_OK หรือ CONNECTION_BAD

  ConnStatusType *PQstatus(PGconn *conn)

- PQerrorMassage จะส่งกลับค่าของความผิดพลาดที่เกี่ยวข้องกับการเชื่อมต่อ

  char *PQerrorMessage(PGconn *conn);

- PQfinish ปิดการเชื่อมต่อกับส่วนของ backend รวมทั้งการปล่อยการจองเนื้อที่ในหน่วยความจำที่ถูกใช้โดย PGconn Structure และการเรียกใช้พอยต์เตอร์ของ PGconn ก็ไม่ควรมีการอ้างถึงหลังจากที่ฟังก์ชัน PQfinish ทำงานด้วย

void PQfinish(PGconn *conn)

- PQreset ทำการเริ่มต้นการสร้างการเชื่อมต่อกับส่วนของ backend ใหม่ ซึ่งฟังก์ชันนี้จะทำการปิดการเชื่อมต่อเดิมของ IPC socket และพยายามสร้างการเชื่อมต่อใหม่ไปยัง backend ตัวเดิม

  void PQreset(PGconn *conn)

3. ฟังก์ชันที่ใช้ในการประมวลผลการสอบถามข้อมูล (Query Execution Functions)

- Pqexec ส่งการสอบถามข้อมูลไปยัง POSTGRES และส่งกลับค่าพอยน์เตอร์ถ้าการสอบถามประสพความสำเร็จ หรือค่า NULL ถ้าส่งกลับค่า NULL จะเรียกใช้ฟังก์ชัน PQerrorMessage ในการแสดงข้อผิดพลาด

  PGresult *PQexec(PGconn *conn,
  char *query);

- PQresultStatus ส่งกลับค่าของสถานะของการสอบถามข้อมูล ซึ่ง PQresultStatus จะส่งกลับในค่าใดค่าหนึ่งดังต่อไปนี้

  PGRES_EMPTY_QUERY,
  PGRES_COMMAND_OK,
  PGRES_TUPLES_OK,
  PGRES_COPY_OUT,
  PGRES_COPY_IN,
  PGRES_BAD_RESPONSE,
  PGRES_NONFATAL_ERROR,
  PGRES_FATAL_ERROR

ถ้าผลลัพธ์ที่ได้เป็น PGRES_TUPLES_OK จึงจะมีการเรียกใช้รูทีนดังต่อไปนี้

- Pqntuples ส่งค่าจำนวนของทูเปิลที่ใช้ในการสอบถามข้อมูล

  int PQntuples(PGresult *res);

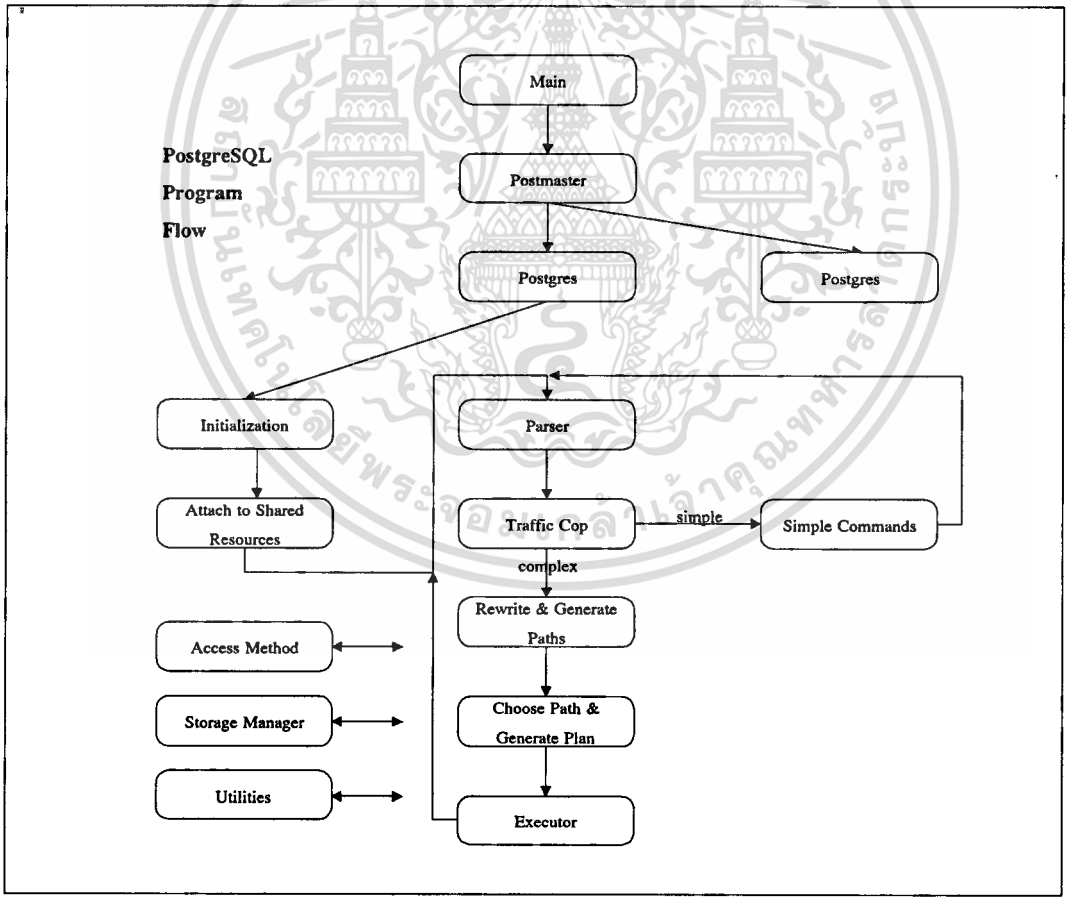
- Pqnfields ส่งค่าจำนวนของฟิลด์ที่ใช้ในการสอบถามข้อมูล

  int PQnfields(PGresult *res);

- Pqfname ส่งชื่อของแต่ละฟิลด์ที่เกี่ยวข้องกับฟิลด์ที่เป็นอินเด็กซ์ โดยกำหนดให้ฟิดล์ที่เป็นอินเด็กซ์เริ่มต้นการทำงานที่ 0

  char *PQfname(PGresult *res,

  int field_index);



รูปที่ 5 POSTGRES Program Flow

บทที่ 5

โปรแกรมประยุกต์ใช้งาน

5.1 ระบบงานห้องสมุด

ผู้มีสิทธิ์ใช้งานระบบห้องสมุด

1. นักศึกษาปัจจุบันที่ลงทะเบียนเรียนในภาคการศึกษานั้นๆ
2. อาจารย์ ข้าราชการ และลูกจ้างประจำของสถาบัน
3. อาจารย์พิเศษ

บริการสิ่งพิมพ์ของห้องสมุด

1. สิ่งพิมพ์ที่เกี่ยวข้องกับพระบาทสมเด็จพระจอมเกล้าเจ้าอยู่หัว
2. วารสารและหนังสือพิมพ์ฉบับปัจจุบัน
3. วารสารฉบับล่วงเวลา
4. หนังสือตำรา สารคดี
5. หนังสืออ้างอิง
6. วิทยานิพนธ์ (ระดับปริญญาโท/เอก)
7. ปริญญานิพนธ์ (ระดับปริญญาตรี)
8. สิ่งพิมพ์รัฐบาล
9. ราชกิจจานุเบกษา
10. มาตรฐานผลิตภัณฑ์อุตสาหกรรม
11. หนังสืออนุสรณ์งานศพ

ระเบียบการยืมสิ่งพิมพ์

1. หนังสือทั่วไป

ประเภทสมาชิก	จำนวนเล่มที่ยืม	จำนวนวัน
นักศึกษาปริญญาตรี	3	7
นักศึกษาปริญญาโท/เอก	5	14
อาจารย์	5	30
อาจารย์พิเศษ	5	14
ข้าราชการ และลูกจ้างประจำ	5	14

ตารางที่ 2 ประเภทของสมาชิก และสิทธิ์ในการยืม-คืน

2. หนังสือจอง หรือหนังสือสำรอง

   ยืมได้ครั้งละ 1 เล่ม ต่อ 1 คืน

3. วารสารล่วงเวลา (ฉบับปลีก)

   นักศึกษาปริญญาตรี, โท, เอก รวมทั้งอาจารย์ ข้าราชการ และลูกจ้างประจำยืมได้ 3 ฉบับ ต่อ 3 วัน

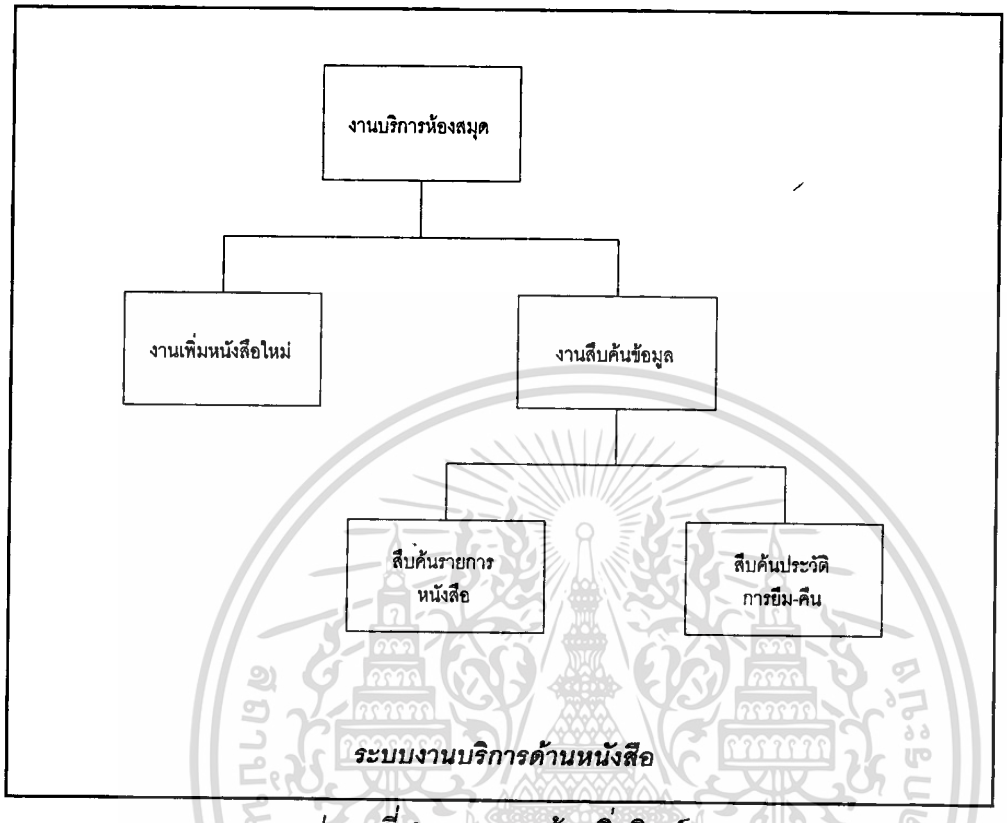
4. สิ่งพิมพ์ประเภทอื่นให้ใช้ภายในสำนักหอสมุด

ข้อบังคับในการยืมสิ่งพิมพ์

1. นักศึกษา อาจารย์ ข้าราชการ และลูกจ้างประจำ จะไม่มีสิทธิ์ยืมหนังสือเล่มต่อไปถ้ามีหนังสือค้างส่ง
2. หนังสือเล่มเดียวกัน ครั้งที่พิมพ์เดียวกัน ยืมได้ครั้งละไม่เกิน 1 เล่ม
3. ยืมหนังสือเล่มเดียวกัน ครั้งที่พิมพ์เดียวกัน สามารถยืมติดต่อกันได้ไม่เกิน 3 ครั้ง

5.2 การวิเคราะห์ระบบงานห้องสมุด

ระบบงานบริการในการสืบค้นข้อมูลในระบบห้องสมุดจะแสดงดังต่อไปนี้



รูปภาพที่ 6 ระบบงานด้านสิ่งพิมพ์

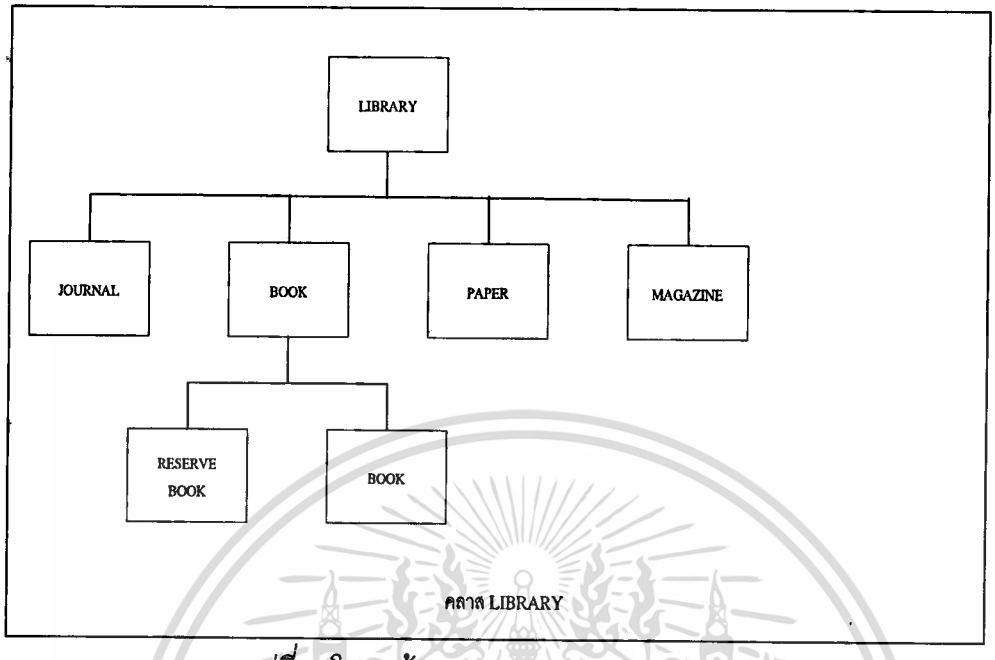
5.2.1 การแบ่งโครงสร้างข้อมูลภายในระบบงานห้องสมุด

1. คลาส LIBRARY
2. คลาส USER
3. คลาส CIRCULATION

ซึ่งแต่ละคลาสจะมีโครงสร้าง และมีรายละเอียดต่างๆ ดังนี้

คลาส LIBRARY คือ คลาสของสิ่งพิมพ์ต่างๆ ทั้งหมดภายในระบบห้องสมุด ซึ่งเราสามารถแบ่งคลาสของห้องสมุดออกได้เป็น subclass ย่อยๆ ดังนี้

- BOOK
- MAGAZINE/ JOURNAL
- PAPER



รูปที่ 7 โครงสร้างของ LIBRARY

คลาส USERS คือ คลาสของผู้ใช้งานทั้งหมดที่สามารถเข้ามาทำการสืบค้นข้อมูลภายในระบบได้ ซึ่งจะแบ่งออกได้เป็น subclass ย่อยๆ ดังนี้

- OUTSIDE USER
- MEMBER

คลาส CIRCULATON ซึ่งเป็นคลาสที่เกี่ยวกับรายการยืม-คืน หนังสือของสมาชิกระบบห้องสมุด

รายละเอียดของชนิดของข้อมูลที่เก็บอยู่ภายในแต่ละคลาสจะแสดงดังในตารางต่างๆ ต่อไปนี้

Table LIB

Field Name	Field Type	Field Length
serial_no	varchar	8
lib_type	varchar	1
title	varchar	40
borr_flag	varchar	1
date_issue	date	8
rec_now	int4	4
s_key	text	var
year_pub	varchar	4
topic	text	var

ตารางที่ 3 LIB

Table Book

Field Name	Field Type	Field Length
serial_no	varchar	8
lib_type	varchar	1
title	varchar	30
borr_flag	varchar	1
date_issue	date	8
rec_now	int4	4
s_key	text	var
year_pub	varchar	4
publish	varchar	40
edition	varchar	2
isbn	varchar	20
author	text	var
call_no	varchar	20
topic	text	var

ตารางที่ 4 BOOK

Table Magazine

Field Name	Field Type	Field Length
serial_no	varchar	8
lib_type	varchar	1
title	varchar	40
borr_flag	varchar	1
date_issue	date	8
rec_now	int4	4
s_key	text	var
year_pub	varchar	4
vol_no	varchar	4
number	varchar	4
category	text	var
issn	varchar	15
author	text	var
topic	text	var

ตารางที่ 5 MAGAZINE

Table Paper

Field Name	Field Type	Field Length
serial_no	varchar	8
lib_type	varchar	1
title	varchar	30
borr_flag	varchar	1
date_issue	date	8
rec_now	int4	4
s_key	text	var
year_pub	varchar	4
category	text	var
vol_no	varchar	4
topic	text	var

ตารางที่ 6 PAPER

Table Member

Field Name	Field Type	Field Length
memberid	varhcar	8
mem_name	varchar	40
mem_type	varchar	2
mem_stat	varchar	2
limit	int4	4
loan	int4	4

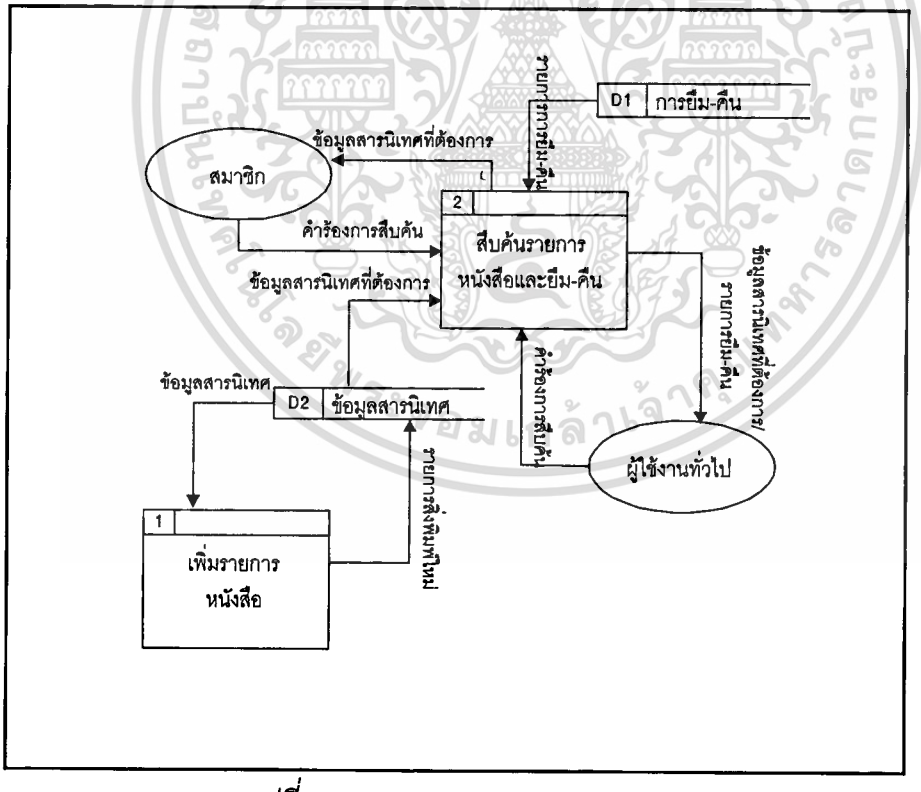
ตารางที่ 7 MEMBER

Table Circulation

Field Name	Field Type	Field Length
mem_id	varchar	8
serial_no	varchar	8
borr_date	date	8
req_date	date	8

ตารางที่ 8 CIRCULATION

5.2.2 DATA FLOW DIAGRAM



รูปที่ 8 DATA FLOW DIAGRAM

5.3 ความรู้ทั่วไปเกี่ยวกับเครือข่ายเวิร์ลไวด์เว็บ

ระบบเวิร์ลไวด์เว็บทำงานโดยอาศัยหลักการจัดการรูปแบบของข้อมูลสารสนเทศ (Information) แบบไฮเปอร์เท็กซ์ (Hypertext) ซึ่งเป็นการโยงใยความสัมพันธ์ระหว่างเอกสารที่มีความสัมพันธ์กัน ในการเชื่อมโยงจะใช้หลักการของ URL (Uniform Resource Locators) ซึ่งจะระบุการอ้างอิงถึงตัวข้อมูล จะมีส่วนประกอบหลักๆ 2 ส่วนคือ ส่วนของ โปรโตคอล (Protocol) และสถานที่ (Location) ที่เป็นแหล่งข้อมูลนั้นๆ

ตัวอย่างเช่น

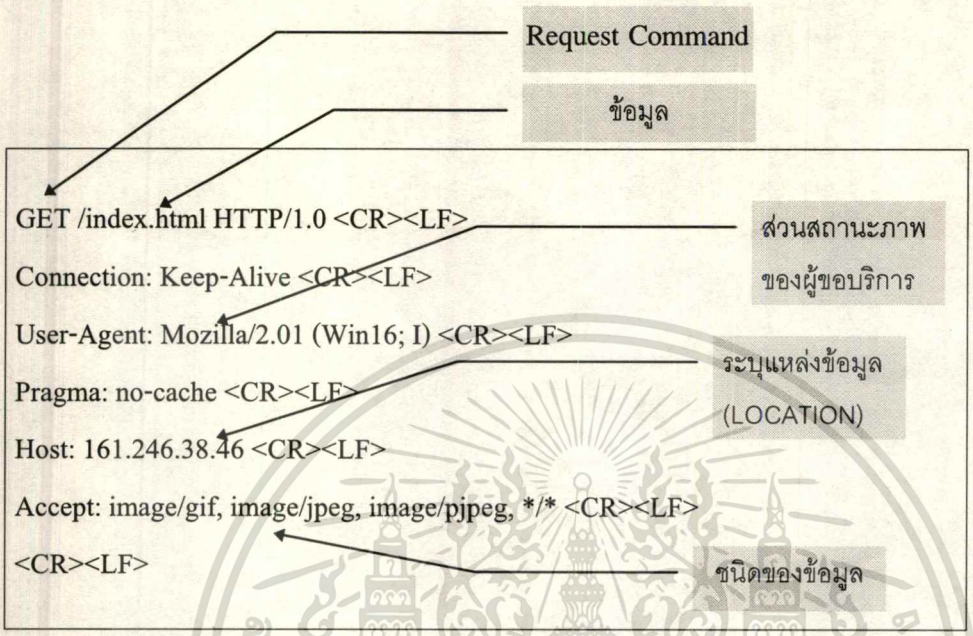
URL	Protocol	Location
http://www.kmitl.ac.th	HTTP	www.kmitl.ac.th
ftp://chaokhun.kmitl.ac.th	FTP	chaokhun.kmitl.ac.th
ftp:s8626072@kmitl.ac.th	FTP	kmitl.ac.th

หลักการทำงานของระบบเครือข่ายเวิร์ลไวด์เว็บ จะอาศัยโปรโตคอลที่ชื่อ HTTP (HyperText Transfer Protocol) ซึ่งเป็นโปรโตคอลแบบทรานส์แอคชั่น โอเรียนเต็ด (Tranaction Oriented) โดยทำงานบนพื้นฐานของโปรโตคอล TCP กล่าวคือจะเชื่อมต่อโดยอาศัย TCP ในการเชื่อมต่อระหว่าง ผู้ใช้บริการ (Client) กับ ผู้ให้บริการ (Server) ในการเชื่อมต่อแต่ละครั้งสามารถทำงานได้หลายทรานเอคชั่น และแต่ละทรานเอคชั่นจะถูกยกเลิกโดยผู้ให้บริการเมื่อทรานเอคชั่นนั้นเสร็จสมบูรณ์ โดยลักษณะข้อมูลที่สนับสนุนเป็นลักษณะแบบไฮเปอร์มีเดีย (Hypermedia) จึงทำให้ระบบมีความหลากหลายในตัวข้อมูลไม่ว่าจะเป็นข้อมูลภาพ (Graphics) , ข้อความ (Text) รวมถึงข้อมูลประเภทเสียงและภาพเคลื่อนไหว

การทำงานของระบบเครือข่ายเวิร์ลไวด์เว็บ

ระบบเครือข่ายเวิร์ลไวด์เว็บทำงานในลักษณะของ Client / Server โดยผู้ใช้ จะใช้งานระบบผ่านโปรแกรมประเภทบราวเซอร์ (Browser) ติดต่อไปยังผู้ให้บริการ โดยที่ฝั่งผู้ให้บริการจะมีการ ติดตั้งโปรแกรม Web Server โดยปกติ HTTPD (HyperText Transfer Protocol Deamon) จะทำงานที่ PORT 80 แต่สามารถเปลี่ยนแปลงได้แล้วแต่จะกำหนด

ตัวอย่าง การส่งขอบริการ (Request) ของผู้ขอบริการ (Client)



จากตัวอย่างการร้องขอบริการข้างต้นจะเป็นขอแฟ้มชื่อ Index.html ที่ Host Address เท่ากับ 161.246.38.46 โดยข้อมูลที่ผู้ให้บริการตอบกลับมาจะแสดงดังส่วนต่อไป

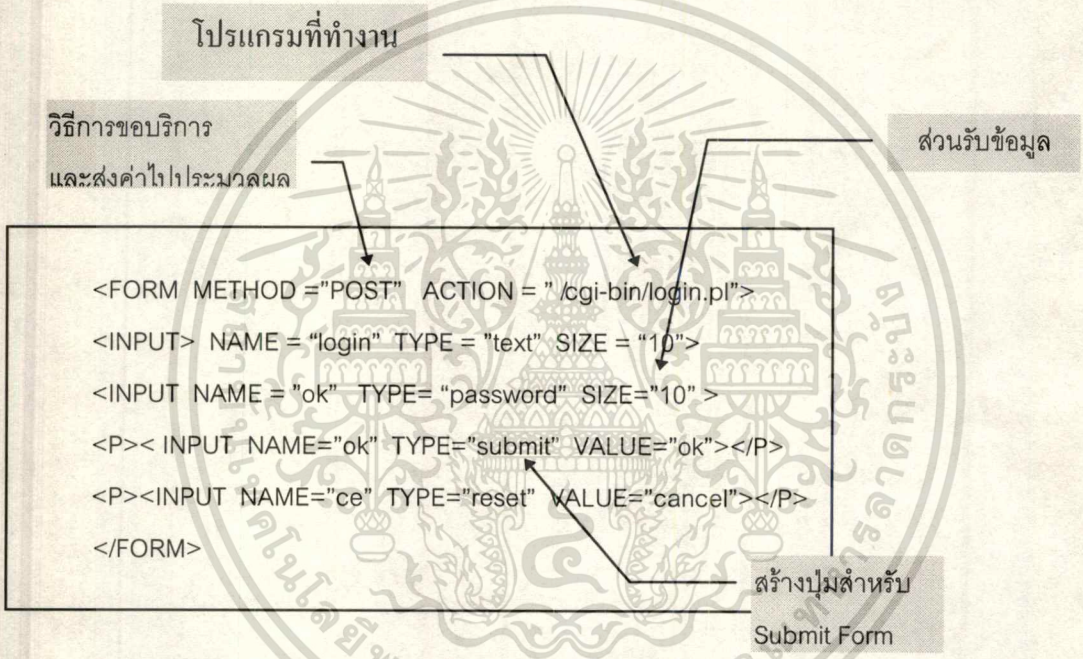
การพัฒนาโปรแกรมประยุกต์บนระบบเว็บโดยหลักการ CGI

ระบบเวิร์ลไวด์เว็บได้มีการกำหนดวิธีการสำหรับการพัฒนาแอพพลิเคชั่นหรือโปรแกรมที่ใช้งานบนระบบ โดยใช้หลักการที่มีชื่อว่า CGI หรือ CGI Web Application ซึ่งได้ระบุถึงวิธีการพัฒนาโปรแกรมโดยอาศัย HTML ในการติดต่อหรือขอบริการจากผู้ให้บริการ ซึ่งโปรแกรมที่ให้ทำงานนั้นจะถูกเก็บบนส่วนของผู้ให้บริการ สำหรับ HTML จะใช้ Tag Form ในการรับค่าจากผู้ใช้งานผ่าน Browser แล้วส่งค่าดังกล่าวไปทำการประมวลผล

นอกจากหลักการของ CGI แล้วยังมีหลักการที่เรียกว่า ISAPI (Internet Server Application Program Interface) ซึ่งจะมีวิธีการคล้ายๆ กัน แต่ ISAPI ส่วนมากจะมีความเฉพาะเจาะจงในส่วนประกอบหรือวิธีการพัฒนามากกว่าทั้งนี้เพราะส่วนมากแล้ววิธีการของ ISAPI มักจะขึ้นกับองค์กรหรือเจ้าของผลิตภัณฑ์นั้นๆ เช่น ตัวฐานข้อมูลและ Web Server ต้องเป็นยี่ห้อเดียวกันจึงจะทำงานได้ดังนั้นการติดต่อ ฐานข้อมูล จะไม่ใช่เรื่องยากในการพัฒนารวมไปถึงวิธีการด้านการรักษาความปลอดภัยด้านฐานข้อมูลด้วย

การทำงานของ CGI

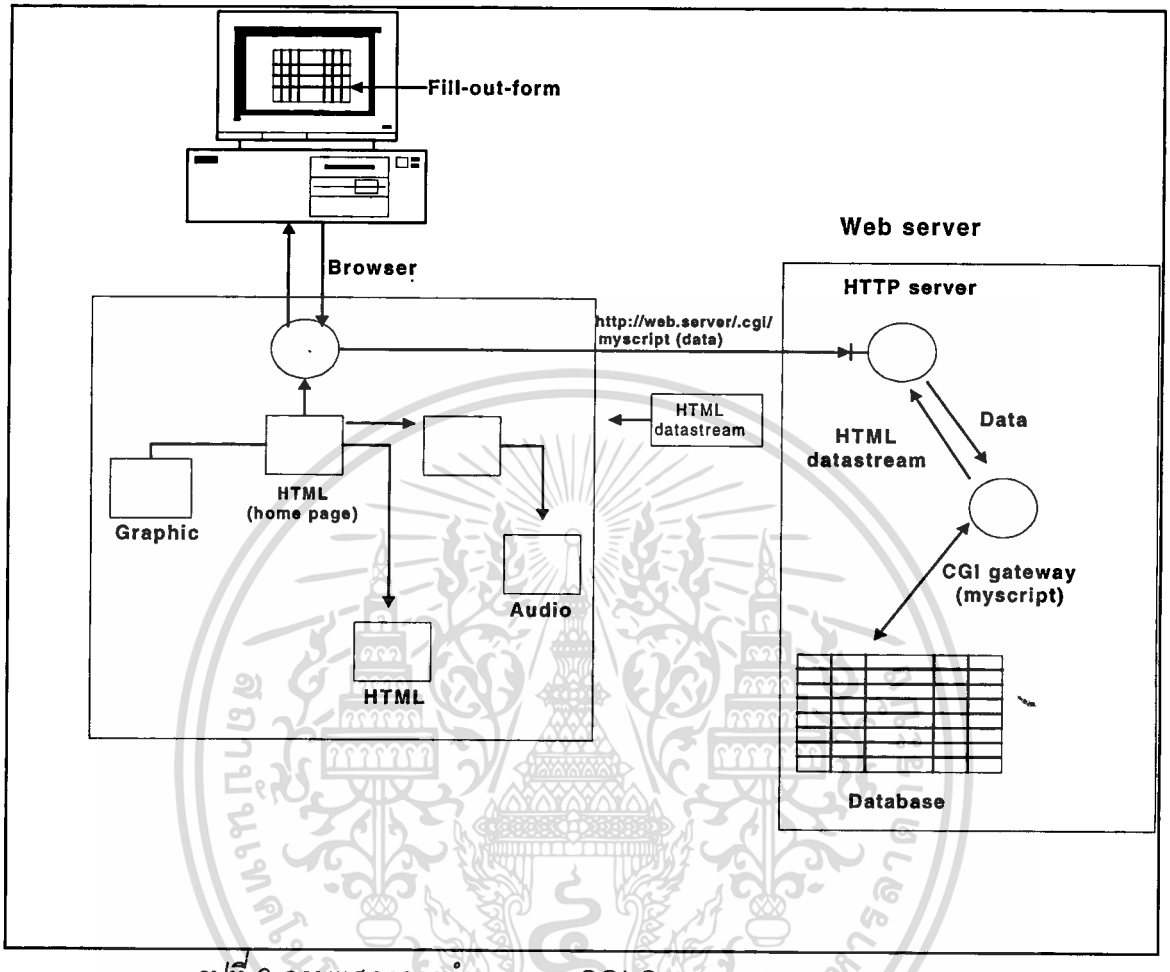
ระบบเครือข่ายเวิร์ลไวด์เว็บมีการสนับสนุนให้สามารถพัฒนาโปรแกรมประยุกต์ (Application) ได้โดยสามารถใช้งานติดต่อใช้งานได้ในลักษณะโต้ตอบ (Interactive) ระหว่างผู้ใชและผู้ให้บริการโดยอาศัยหลักการที่เรียกว่า CGI (Common Gateway Interface) และความสามารถของ HTML (HyperText Markup Language) ซึ่งนอกจาก HTML จะเป็นภาษาที่เราใช้ในการพัฒนาเอกสารในรูปแบบ HyperText แล้ว HTML ยังระบุถึงวิธีการพัฒนา Web Application ตามหลักการของ CGI โดยใช้ Tag Form ใน HTML



ตัวอย่างการแสดงการใช้งาน Tag Form (HTML)

การทำงานของ CGI นั้น CGI จะเป็นตัวเชื่อมในการทำงานระหว่าง Web Server กับ Server Environment โดย Web Server จะเรียกใช้โปรแกรมที่ติดตั้งอยู่บนฝั่งผู้ให้บริการ

โดยหลักการของ CGI จะทำให้ผู้ใช้บริการ (Client) สามารถติดต่อส่วนอื่นๆ ที่ Server ซึ่งผู้ใช้บริการไม่สามารถติดต่อได้โดยตรง เช่น Database ด้วยเหตุนี้ทำให้สามารถพัฒนาแอพพลิเคชั่นบนเครือข่ายเวิร์ลไวด์เว็บได้ สำหรับภาษาที่ในการพัฒนาโปรแกรมนั้นจะขึ้นอยู่กับชนิด (Plat-form) ของ Web Server และ ระบบปฏิบัติการ (O.S.) เช่น ระบบปฏิบัติ Unix มักนิยมใช้ จะเป็น Perl ส่วน Database จะนิยมใช้ MSQL(Minisql) ในการพัฒนาแอพพลิเคชั่น แต่สำหรับโครงงานนี้เลือกใช้ Perl ในการพัฒนาโปรแกรม และเลือกใช้ Postgresql เป็นฐานข้อมูล



รูปที่ 9 ภาพแสดงการทำงานของ CGI Gateway

บทที่ 6

บทสรุปและแนวทางการพัฒนาระบบในอนาคต

PostgreSQL เป็นตัวอย่างหนึ่งของระบบจัดการฐานข้อมูลเชิงวัตถุ ซึ่ง PostgreSQL ในระบบนั้นทำงานอยู่ภายใต้การทำงานของแพทฟอร์ม Intel Pentium ที่ใช้ตัว Operating system คือ LINUX ดังนั้นรูปแบบของคำสั่งต่างๆ ที่ทำงานกับระบบฐานข้อมูลเชิงวัตถุจะเป็นคำสั่งพื้นฐานของระบบ UNIX ดังนั้นผู้ที่ทำหน้าที่ดูแลระบบฐานข้อมูล PostgreSQL จะต้องมีความรู้พื้นฐานทางด้านคำสั่ง หรือการใช้งานของระบบ UNIX ด้วย

ลักษณะของโปรแกรมประยุกต์ที่ใช้ในระบบเป็นตัวอย่างของการประยุกต์การออกแบบระบบจากเดิมที่เป็นระบบฐานข้อมูลเชิงสัมพันธ์เป็นระบบฐานข้อมูลเชิงวัตถุ ดังนั้นรูปแบบของโครงสร้างข้อมูลจะแตกต่างกัน รวมทั้งภาษาที่ใช้ในการสอบถาม หรือทำงานกับข้อมูลก็จะมีลักษณะการใช้งานที่แตกต่างกันด้วย นั่นคือจะใช้ SQL3 ในการสอบถาม หรือทำงานร่วมกับข้อมูล แต่โครงสร้างของรูปแบบเดิมจะยังคงสภาพการใช้งานอยู่ ซึ่งผู้ใช้ที่ไม่มีความรู้เกี่ยวกับ SQL3 นั้นก็สามารถใช้งานระบบได้ถ้ามีความรู้เกี่ยวกับการใช้งาน SQL-92 อยู่

แนวทางการพัฒนาระบบในอนาคต คือ การเพิ่มหน้าที่การทำงานต่างๆ ของระบบให้สมบูรณ์ครบถ้วน ทั้งด้านระบบการยืม-คืน เอกสารต่างๆ ผ่านทางระบบอินเตอร์เน็ทได้ หรือการนำลักษณะการทำงานของระบบฐานข้อมูลเชิงวัตถุมาออกแบบระบบให้อยู่ในรูปที่สนับสนุนการทำงานของห้องสมุดเต็มที่ เช่น การนำเสนอในรูปของมัลติมีเดีย เป็นต้น

บรรณานุกรม

[1] Hughes,John G. Object-Oriented Database. NewYork:Prentice Hall,1991.

[2] Maier D. and J. Stenia Research in Object-Oriented Database System: Combining Language and Database Advances in an Object-Oriented Development. Morgan Kaufman,1990

[3] Petr,Kroha. Objects Oriented Database System. New York:MacGraw-Hill, 1993.

[4] Stonebraker,Michael, Eric Hanson and Chin-Heng Hong.Postgres Data Model:Http://www.postgresql.org/

[5] Stonebraker,Michael and Lawrence A. Rowe. The Design of Postgres: Http://www.berkeley.edu/

ภาคผนวก

ภาคผนวก ก

1. Postgres Setup

Hardware and Software Requirement

PLATFORMS	OPERATING SYSTEM
IBM on AIX 3.2.5	aix
DEC Alpha AXP on OSF/12.0	alpha
NetBSD, FreeBSD	BSD44_derived
BSD/OS 2.0, 2.01, 2.1	bsdi
DG/UX 5.4R3.10	dgux
HP PA-RISC on HP-UX 9.0	hpux
i386 Solaris	i386_solaris
SGI MIPS on IRIX 5.3	irix5
Intel x86 on Linux 2.0 and Linux ELF	linux
SPARC on Linux ELF	
SUN SPARC on Solaris 2.4	sparc_solaris
SUN SPARC on SunOS 4.1.3	sunos4
Intel x86 on Intel SVR4	svr4
DEC MIPS on Ultrix 4.4	ultrix4

Installation Guide

การ Install PostgreSQL v6.1 มีขั้นตอนการทำงานดังนี้

1. อ่านรายละเอียดในการ install เกี่ยวกับ platforms ต่างๆ ที่สนับสนุนการทำงานของ PostgreSQL และขั้นตอนในการ install จากแฟ้มข้อมูลชื่อ INSTALL
2. สร้าง account postgres ขึ้นมาถ้ายังไม่มี
3. Login เข้าไปใน account postgres

4. ตรวจสอบเนื้อที่ของฮาร์ดดิสก์ ซึ่งจะต้องใช้เนื้อที่ประมาณ 17 Mbytes สำหรับ /usr/src/pgsql และประมาณ 2 Mbytes สำหรับ /usr/local/pgsql ในระหว่างที่ทำการ installed และทดสอบการทำงาน ควรจะเก็บเนื้อที่ไว้ประมาณ 20 Mbytes หรือมากกว่าภายในพาท /usr/local และอีกประมาณ 25 เมกะไบท์เพื่อใช้ในการเก็บฐานข้อมูลทั้งหมด ซึ่งสามารถตรวจสอบเนื้อที่ภายในฮาร์ดดิสก์ได้โดยการใช้คำสั่ง "df -k"

5. ถ้ายังไม่มีโปรแกรมที่ใช้ในการ install (Suorce Codes) สามารถไปดึงมาได้จากไซท์ของ ftp://ftp.postgresql.org/pub/postgresql-v6.1.tar.gz ผ่านทางระบบอินเตอร์เน็ทได้ แล้วนำมาเก็บไว้ยังโฮมไดเร็คทอรี

6. บางแพลทฟอร์มจะต้องมีการใช้เฟล็กซ์ ซึ่งถ้าระบบที่ใช้อยู่ใช้เฟล็กซ์ควรทำการตรวจสอบก่อนว่าใช้เวอร์ชันตรงกับที่ต้องการหรือไม่ โดยการใช้คำสั่ง

   flex –version

   แต่ถ้าไม่พบคำสั่งเฟล็กซ์แสดงว่าไม่จำเป็นที่จะต้องใช้เฟล็กซ์ แต่ถ้าเวอร์ชันที่ใช้อยู่เป็น 2.5.2 หรือ 2.5.4 หรือ เป็นเวอร์ชันที่สูงกว่านั้นแสดงว่าตรงกับที่ต้องการใช้งาน แต่ถ้าเป็นเวอร์ชัน 2.5.3 หรือ เป็นเวอร์ชันก่อนหน้า 2.5.2 จะต้องทำการปรับปรุงเวอร์ชันของเฟล็กซ์ใหม่ ซึ่งสามารถไปดึงเอามาใช้งานได้จากไซท์ ftp://prep.ai.mit.edu/pub/gnu/flex-2.5.4.tar.gz

   การ install flex กระทำดังนี้

   ```
   cd
   gunzip -c flex-2.5.4.tar.gz | tar xvf -
   cd flex-2,5.4
   configure --prefix=/usr
   make
   make check
   ```

   #จะต้องเป็น root เท่านั้นที่จะสามารถทำคำสั่งต่อไปนี้ได้

   ```
   make install
   cd
   rm -rf flex-2.5.4
   ```

7. ถ้าทำการปรับปรุงจากระบบที่มีอยู่ ให้ทำการสำรองข้อมูลที่มีอยู่ทั้งหมด

```
cd
gunzip -c postgresql-v6.1.tar.gz | tar xvf - src/bin/pg_dump/pg_dumpall
chmod a+x src/bin/pg_dump/pg_dumpall src/bin/pg_dump/pg_dumpall >
db.out
rm -rf src
```

ข้อควรระวังในขณะที่ทำการสำรองข้อมูล คือ จะต้องระวังไม่ให้เกิดการเปลี่ยนแปลงกับข้อมูลในขณะนั้น ซึ่งทางที่ดีควรจะหยุดการทำงานในส่วนของ postmaster ก่อนแล้วกำหนดสิทธิ์ให้กับไฟล์ /usr/local/pgsql/data/pg_hba.conf เพื่อให้คุณสามารถใช้งานได้เพียงผู้เดียวเท่านั้น แล้วค่อยเริ่มการทำงานในส่วนของ postmaster ใหม่

8. ถ้าเป็นการปรับปรุงระบบใหม่ให้ทำลายการทำงานของ postmaster ก่อนโดยใช้คำสั่ง

```
ps -ax | grep postmaster
```

9. ถ้าเป็นการปรับปรุงระบบใหม่ให้ทำการเคลื่อนย้ายไดเร็คทอรี่เดิมออกไปก่อน แต่ถ้าเหลือพื้นที่น้อยให้ทำการสำรอง และลบไดเร็คทอรี่นั้นออกไปก่อน โดยการจัดเก็บส่วนของข้อมูลเก่าลงไปในส่วนของ /usr/local/pgsql/data และจัดเก็บไฟล์ลงในส่วนของ /usr/local/pgsql/data/pg_hba.conf
โดยทำตามขั้นตอนดังนี้

```
su
cd /usr/src
mv pgsql pgsql_6_0
cd /usr/local
mv pgsql pgsql_6_0
exit
```

10. ทำการสร้างไดเร็คทอรีที่ใช้ในการ install ดังนี้

```
su
cd /usr/src
mkdir pgsql
chown postgres:postgres pgsql
cd /usr/local
```

```
mkdir pgsql
chown postgres:postgres pgsql
exit
```

11. ทำการ unzip และ untar ไฟล์ที่เป็นตัวใหม่ โดย

```
cd /usr/src/pgsql
gunzip -c ~/postgresql-v6.1.tar.gz | tar xvf -
```

12. ทำการปรับขยายส่วนของโปรแกรมต้นฉบับให้กับระบบ โดยการกำหนดเส้นทางที่ใช้ในการเก็บตัวโปรแกรมต้นฉบับ และเส้นทางที่ใช้ในการเก็บส่วนของการประมวลผล

```
cd /usr/src/pgsql
./configure
```

ซึ่งโปรแกรมที่ใช้ในการปรับขยายจะแสดงรายการของต้นแบบโปรแกรม (template files) เพื่อให้ทำการเลือก ซึ่งอาจจะต้องใช้เวลาระยะหนึ่งในการเลือกและกำหนดค่า หรือกดแป้น Enter เพื่อกำหนดให้เป็นค่า default และจะต้องทำการป้อนข้อมูลบางส่วนเข้าไปในการตอบคำถามที่ต้องใช้ในการกำหนดการปรับขยายระบบ ซึ่งผู้ใช้สามารถจะข้ามในส่วนของการตอบคำถามได้โดยการกำหนดพารามิเตอร์ลงไปในส่วนท้ายของคำสั่งได้ พาราเตอร์ที่สามารถใช้ได้มีดังนี้ คือ

--prifix=BASEDIR เลือกไดเร็คทอรีพื้นฐานที่ใช้ในการกำหนดการปรับขยายของการ install PostgreSQL

--enable-hba ยอมให้มีการรองรับส่วนพื้นฐานของ Host

--disable-hba ไม่ยอมให้มีการรองรับส่วนพื้นฐานของ Host

--enable-locale Enables USE_LOCALE

--disable-locale Disables USE_LOCALE

--enable-cassert Enables ASSERT_CHECKING

--disable-cassert Disables ASSERT_CHECKING

--with-template=TEMPLATE การใช้ต้นแบบโปรแกรม ซึ่ง TEMPLATE คือ ส่วนของต้นแบบโปรแกรมที่ถูกกำหนดให้อยู่ในไดเร็คทอรีของ src/template

(ถ้าส่วนของ configure script ไม่สามารถค้นหาส่วนของต้นแบบโปรแกรมได้ มันจะถามผู้ใช้อีกครั้งหนึ่ง)

--with-pgport=PORT การกำหนด port ที่ postmaster ใช้ในการประมวลผลในการติดต่อการสื่อสารกับส่วนอื่นๆ ซึ่ง default port ที่ใช้ คือ 5432

ตัวอย่างการใช้งาน เช่น การเขียน script ให้ใช้งานบนเครื่อง Sparc Solaris 2.5 ที่ใช้เส้นทาง /opt/postgres ในการ installed คือ

```
% ./configure --prefix=/opt/postgres --with-template=sparc_solaris-gcc
                --with-pgport=5432 --enable-hba --disable-locale
```

13. ทำการ compile ส่วนของโปรแกรม ดังนี้

```
cd /usr/src/pgsql/src
gmake all > & make.log &
tail -f make.log
```

ถ้าในส่วนของ compiler นั้นล้มเหลวในการทำงานจากส่วนของเฟล็กซ์ไม่สามารถค้นหาได้ ให้ทำการ install ในส่วนของเฟล็กซ์ใหม่ตามที่ได้กล่าวมาแล้ว จากนั้นให้เปลี่ยนไดเร็คทอรีกลับมาที่เดิม แล้วใช้คำสั่ง “make clean” หลังจากนั้นให้ทำการ compile โปรแกรมใหม่อีกครั้ง

14. ทำการ install โปรแกรม ดังนี้

```
cd /usr/src/pgsql/src
gmake install >& make.install.log &
   tail -f make.install.log
```

15. ถ้าเป็นไปได้ ให้กำหนดให้ UNIX ทราบด้วยว่าควรจะไปค้นหาในส่วนของ shared libraries ได้ที่ไหน ถ้าใช้ Linux-ELF ให้เลือกทำใน 1 ข้อจากที่กล่าวถึงต่อไปนี้

a) ผู้ใช้ที่เป็น root ให้แก้ไขไฟล์ /etc/ld.so.conf แล้วเพิ่มบรรทัดนี้เข้าไป

   /usr/local/pgsql/lib

   แล้วทำการเรียกการทำงานคำสั่ง /sbin/ldconfig

b) ใน bash shell ให้พิมพ์

   export LD_LIBRARY_PATH=/usr/local/pgsql/lib

c) ใน csh shell ให้พิมพ์

setenv LD_LIBRARY_PATH /usr/local/pgsql/lib

สำหรับคำสั่งที่ใช้ดังกล่าวจะแปรตามการใช้งานของ os ในแต่ละรุ่น และถ้าในขณะที่ทำการสร้างฐานข้อมูลขึ้นมาแล้วได้รับข้อความว่า "pg_id: can't load library 'libpq.so'" นั่นคือคำสั่งที่ใช้ในข้อข้างบนเป็นส่วนจำเป็นที่จะต้องกระทำตาม แล้วจึงทำการสร้างฐานข้อมูลขึ้นมาใหม่

16.เตรียมในส่วนของ account postgres เพื่อให้ใช้งาน PostgreSQL ได้ โดยการแก้ไขในส่วนของ login shell ในไฟล์ ~/.bash_profile

```
PATH=$PATH:/usr/local/pgsql/bin
MANPATH=/usr/local/pgsql/man
PGLIB=/usr/local/pgsql/lib
PGDATA=/usr/local/pgsql/data
export PATH MANPATH PGLIB PGDATA
```

เพื่อให้แน่ใจว่าได้มีการกำหนดตรงตามกับที่กล่าวมาข้างบน สามารถตรวจสอบได้โดยการใช้คำสั่ง source ~/.bash_profile

17.สร้างฐานข้อมูล ซึ่งขั้นตอนการทำงานนี้ห้ามกระทำโดย ROOT

initdb

18.กำหนดสิทธ์ที่ใช้ในการเข้าถึงข้อมูลในระบบฐานข้อมูล โดยการแก้ไขในไฟล์ /usr/local/pgsql/data/pg_hba.conf ถ้าเป็นการเปลี่ยนจากระบบเดิมไปสู่ระบบใหม่ให้ทำการคัดลอกไฟล์จากระบบเดิมมาเก็บไว้ในระบบฐานข้อมูลใหม่

19.ขั้นตอนนี้เป็นส่วนของการทดสอบระบบ ถ้าไม่ต้องการทดสอบสามารถข้ามไปทำงานในส่วนของข้อ 22 ได้เลย

ขั้นตอนในการทดสอบระบบให้อ่านรายละเอียดจากไฟล์

/usr/src/pgsql/src/test/regress/README

คำสั่งที่ใช้ในการรัน และการแปลงระบบ ในส่วนถัดไปเป็นข้อสรุปการทำงานดังนี้
เริ่มต้นการทำงานของ postmaster เพื่อเตรียมทดสอบโปรแกรม โดยการเริ่มต้นการกำหนดช่วงเวลาซึ่งจะใช้มาตรฐานของ Berkeley, California ในบางระบบจะทำขั้นตอนนี้โดยการกำหนดสภาวะแวดล้อมใน TZ. ถ้าใช้ bash shell จะพิมพ์

```
export TZ=PST8PDT
```

หลังจากนั้นให้เริ่มต้นการทำงานของ postmaster โดยการรันเป็น background process โดยการพิมพ์

```
cd
nohup postmaster > regress.log 2>& 1 &
```

รันการทำงานของ postmaster โดย postgres super user account **ห้ามรันผ่าน root** account

20.รัน regression tests พิมพ์

```
cd /usr/src/pgsql/src/test/regress
gmake clean
gmake all runtest
```

ถ้าเป็นการทำงานในครั้งแรกของการใช้งาน postgres ไม่จำเป็นต้องใช้คำสั่ง “gmake clean” การทดสอบที่เกิดขึ้นนั้นอาจจะมีบางส่วนที่พลาดไปเป็นเรื่องปกติ ซึ่งอาจจะเกิดมาจากการกำหนดช่วงเวลาที่แตกต่างกันได้ และสามารถเปรียบเทียบผลจากการรันได้ในส่วนของ ./results กับส่วนของไฟล์ที่เก็บอยู่ในไดเร็คทอรีของ ./expected
หลังจากทำการรันแล้ว พิมพ์

```
destroydb regression
cd /usr/src/pgsql/src/test/regress
gmake clean
```

21.หยุดการทำงานของ postmaster โดยการใช้คำสั่งที่กล่าวไปแล้วในขั้นตอนที่ 8 และกำหนดในส่วนของช่วงเวลาใหม่ ถ้ากำหนดช่วงเวลาโดยการกำหนดสภาพแวดล้อมจะแก้ไขได้โดยการปิดเครื่อง และ login เข้ามาใหม่โดยการใช้ postgres account

22.รัน postmaster deamon ให้ทำงาน พิมพ์

```
cd
nohup postmaster > server.log 2>&1 &
```

ข้อสังเกตุ จะต้องทำการรัน postmaster ด้วย postgres super user account ห้ามรัน postmaster ด้วย ROOT ACCOUNT

23.การจัดการให้ postmaster เริ่มการทำงานโดยอัตโนมัติเมื่อมีการบู๊ทเครื่องคอมพิวเตอร์ นั่นคือเมื่อไรก็ตามที่เริ่มต้นการทำงานของ postmaster จะต้องไม่ใช้ root account ต่อไปจะกล่าวถึงวิธีการกำหนดในแต่ละแพลตฟอร์ม

a) แก้ไขไฟล์ rc.local บน NetBSDหรือ rc2.d บน SPARC Solaris 2.5.1 โดยการเพิ่มในส่วนของคำสั่งดังต่อไปนี้

```
su postgres -c "/usr/local/pgsql/bin/postmaster -S -D /usr/local/data"
```

b) สำหรับ RedHat v4.0 Linux แก้ไขในไฟล์ /etc/inittab แล้วเพิ่มในส่วนต่อไปนี้

```
pg:2345:respawn:/bin/su - postgres -c
"/usr/local/pgsql/bin/postmaster -D/usr/local/pgsql/data >>
/usr/local/pgsql/server.log 2>&1" /dev/null
```

c) สำหรับ FreeBSD 2.2-RELEASE ให้ทำการแก้ไขในส่วนของไฟล์ /usr/local/etc/rc.d/pgsql.sh ดังนี้

```
#!/bin/sh
[ -x /usr/local/pgsql/bin/postmaster ] && {
    su -l pgsql -c 'exec /usr/local/pgsql/bin/postmaster
    -D/usr/local/pgsql/data
    -S -o -F > /usr/local/pgsql/errlog' &
  echo -n ' pgsql'
}
```

หลังจากนั้นให้ทำการสร้าง chmod 755 และ chown root:bin

d) สำหรับ RedHat Linux สร้างไฟล์ /etc/rc.d/init.d/postgres.init ในการเก็บส่วนดังนี้

```
su -c "cd ~postgres; nohup /usr/local/pgsql/bin/postmaster
    -D/usr/local/pgsql/data > server.log 2>&1 &" postgres
```

ต่อมาให้พิมพ์

```
cd /etc/rc3.d
ln -s ../init.d/postgres.init S1000postgres
```

(จากตัวอย่างที่แสดงนี้ยังไม่ได้รับการทดสอบ)

23a).การทำงานในขั้นตอนนี้สำหรับการบำรุงรักษาระบบ โดยทำตามขั้นตอนต่างๆ ที่กำหนดให้ดังนี้

a) รัน SQL vacuum ซึ่งจะเป็นการล้างฐานข้อมูลทั้งหมด
b) สำรองระบบเอาไว้ แต่ในขณะนั้นจะต้องไม่มีผู้ใดใช้งานระบบอยู่
c) หยุด และเริ่มการทำงานของ postmaster ใหม่

จากขั้นตอนการทำงานที่กล่าวมาทั้งหมดนั้นสามารถเขียนให้อยู่ในรูปของ script file ได้เพื่อให้ระบบสามารถทำงานได้ตลอดเวลาทั้งกลางวัน และกลางคืน ซึ่งการทำงานในส่วนนี้จะดูรายละเอียดได้จากส่วนของ man page

24).ถ้าทำการปรับปรุงระบบจากระบบเดิม ให้ทำการ install ฐานข้อมูลเดิมลงไป พิมพ์

```
cd
psql -e template1 < db.out
```

25).ถ้าเป็นผู้ใช้ใหม่สำหรับ postgres สามารถทดลองการทำงานต่างๆ ได้ดังจะได้กล่าวต่อไป

26).ทำการ clean up ได้ดังนี้

```
rm -rf /usr/src/pgsql_6_0
rm -rf /usr/local/pgsql_6_0
rm ~/postgresql-v6.1.tar.gz
```

2. Administering Postgres

งานที่ผู้ควบคุมระบบงานจะต้องกระทำเมื่อมีการเริ่มต้นระบบการทำงาน Postgres จะแบ่งออกเป็น 2 ประเภท คือ

- Frequent Tasks
- Infrequent Tasks

1. Frequent Tasks

ขบวนการในการทำงานที่ผู้ควบคุมระบบงานควรจะต้องทำความคุ้นเคยในการจัดการเกี่ยวกับระบบ Postgres มีดังนี้

1.1 *Starting the Postmaster*

ถ้าการลงโปรแกรม (Installation) ไม่สมบูรณ์ตรงตามขั้นตอนต่างๆ ที่กล่าวถึงในส่วนของการลงโปรแกรมระบบ Postgres แล้วคุณอาจจะต้องทำความเข้าใจขั้นตอนการทำงานต่างๆ ก่อนการเริ่มต้นการทำงานของ Postmaster ดังนี้

- ต้องเข้าใจเกี่ยวกับสถานที่ที่ได้ทำการลงโปรแกรมเอาไว้ กำหนดในส่วนของสภาวะแวดล้อมในการใช้งาน และส่วนอื่นๆ
- คุณจะต้องเริ่มต้นการทำงานในส่วนของ Postmaster ด้วยการใช้งาน user-id ที่เป็นเจ้าของในส่วนของฐานข้อมูล ในกรณีส่วนใหญ่ควรจะใช้ user-id ที่เป็น Postgres ถ้าเริ่มต้นการทำงาน Postmaster ด้วย User-id ที่ถูกต้อง ในส่วนของ backend server ที่ถูกเริ่มต้นการทำงานด้วย Postmaster จะไม่ทำงานตามที่ได้กำหนดไว้
- สร้างความแน่ใจว่าเส้นทางที่ใช้ในการลงโปรแกรมนั้น คือเส้นทางที่ตรงกับ /usr/local/postgres95/bin อยู่ในส่วนของ shell command เรียบร้อยแล้ว เพราะว่าการทำงานของ Postmaster จะใช้เส้นทาง PATH ในการค้นหาส่วนของ Postgres Commands
- อย่าลืมที่จะกำหนดในส่วนของสภาวะแวดล้อมของระบบโดยผ่านทาง PGDATA ลงในส่วนไดเรคทอรี่ที่ Postgres database ได้ถูกลงเอาไว้ (อธิบายไว้ในส่วนของการลงโปรแกรมแล้ว)
- ถ้าเริ่มต้นการทำงานในส่วนของ Postmaster ด้วยออพชันที่ไม่เป็นมาตรฐาน เช่น การใช้ TCP port number ที่แตกต่างออกไป จะต้องทำการบอกให้ผู้ใช้ทุกคนรู้ว่าจะต้องทำการกำหนด port ให้ตรงกับที่กำหนดไว้ใน PGPORT อย่างถูกต้อง

1.2 *Shutting Down the Postmaster*

ถ้าต้องการหยุดการทำงานของขบวนการ Postmaster สามารถกระทำได้โดยการใช้คำสั่งของระบบ UNIX คือ kill(1) ซึ่งผู้ใช้บางคนอาจจะใช้ออพชัน -9 หรือ -KILL ซึ่งในส่วนนี้ไม่ใช่เรื่องสำคัญ และไม่ขอแนะนำให้ใช้งาน ในขณะที่ Postmaster ไม่สามารถทำการปล่อยการใช้งานร่วมของทรัพยากรต่างๆ, ขบวนการต่างๆ ที่เป็นส่วนลูกนั้นไม่สามารถหยุดการทำงานได้อย่างถูกต้อง เป็นต้น

1.3 *Adding and Removing Users*

การใช้คำสั่งในการสร้างผู้ใช้งานระบบใหม่จะใช้คำสั่ง Createuser และการยกเลิกการใช้งานของผู้ใช้เหล่านั้นจะใช้คำสั่ง Destroyuser ในการกำหนดให้ผู้ใช้มีสิทธิ์ หรือหมดสิทธิ์ในการใช้งานระบบ Postgres

1.4 *Periodic Upkeep*

Vacuum command ที่ใช้งานควรจะทำการประมวลผลกับส่วนของแต่ละฐานข้อมูลเป็นระยะๆ คำสั่งนี้จะใช้ในการลบส่วนของข้อมูล (instance) และที่สำคัญจะทำการปรับปรุงในเชิงสถิติของระบบเกี่ยวกับขนาดของแต่ละคลาส ถ้าข้อมูลเชิงสถิตินี้เกี่ยวกับการหมดอายุ และไม่ถูกต้องแล้ว Postgres Query Optimizer จะเป็นผู้ตัดสินใจว่าควรเลือกข้อมูลในส่วนใด ดังนั้นจึงขอแนะนำให้ทำการประมวลผลโดยการใช้คำสั่ง Vacuum ทุกๆ คืน หรือ ในระยะเวลาใกล้เคียง

Frequent Backups ควรกระทำการสำรองข้อมูลในฐานข้อมูลเป็นระยะๆ โดยการใช้คำสั่ง Copy หรือ คำสั่งของระบบยูนิกซ์ คือ dump หรือ tar คุณอาจจะคิดว่าทำไมจะต้องทำการสำรองข้อมูล และจะต้องทำอะไรบ้างในการกู้คืนเมื่อระบบเกิดล้มไป สิ่งที่สำคัญของระบบ Postgres คือ “no overwrite” ในส่วนของการจัดการเนื้อที่เก็บข้อมูลและนั่นคือ “no log” ดังนั้นในส่วนของ log นั้นจะเก็บข้อมูลเฉพาะที่เป็น abort/commit data เท่านั้น นั่นคือข้อมูลที่เก็บอยู่ไม่เพียงพอต่อการกู้คืนของระบบ และที่สำคัญถ้าข้อมูลทีเก็บอยู่ภายในระบบเป็นข้อมูลที่มีการแบ่งการใช้งาน เช่น pg_database เป็นต้น

1.5 *Tuning*

เมื่อใดก็ตามที่ผู้ใช้ทำการใช้งานกับข้อมูลที่เก็บอยู่ ประสิทธิภาพการใช้งานของระบบจะตกลง เพราะว่า Postgres ไม่ใช่ DBMS ที่มีความเร็วที่สุด ดังนั้นจะต้องมีการปรับระบบซึ่งจะต้องอาศัยประสพการณ์ในการทำงานของผู้ใช้ระบบด้วย

- กำหนดส่วนของ indices ให้กับ attributes ที่ใช้ในการกำหนดคุณสมบัติของระบบ เช่น ถ้ามีการใช้คำสั่ง

  SELECT * from EMP where salary < 5000

  ดังนั้น B-tree index ที่ใช้ใน salary นั้นจะมีความสำคัญมาก แต่ถ้าเป็นการ scan เช่น

  SELECT * from EMP where salary = 5000

  ควรจะใช้ hash index ที่ salary ซึ่งจะทำให้ประสิทธิภาพสูงกว่า การใช้งานโดยผ่าน indices จะมีความเร็วสูงกว่า sequential scans
- ควรจะรัน vacuum command มากครั้ง ซึ่งจะช่วยให้ query optimizer ช่วยในการตัดสินใจในการทำงานที่ดีได้
- เมื่อมีการทำการ join หลายๆ คลาสพร้อมกันใน query พยายามเขียน join clause ให้อยู่ในรูปของ "chained" เช่น

  where A.a = B.b and B.b = C.c and ...
- ถ้าต้องการเปรียบเทียบผลการทำงานใน query นั้นสามารถเรียกใช้ออพชัน -d เมื่อทำการรัน postmaster และ -t เมื่อทำการรัน monitor รูปแบบที่ออกมานั้นจะยากต่อการดู แต่สามารถบอกได้ว่ามีการแสดงการทำงานเกี่ยวกับ index อย่างไร

2. Infrequent Tasks

ขบวนการในการทำงานของผู้ควบคุมระบบนั้นจะมีงานประเภทอื่นที่จะต้องคำนึงถึง ดังต่อไปนี้

2.1 Cleaning Up After Crashes

Postgres server และ Postmaster จะรันแยกออกเป็น 2 โพรเซสแยกกันซึ่งในแต่ละโพรเซสนั้นอาจจะหยุดทำงานแยกกัน หรืออาจเกิดมาจากทั้งสองโพรเซสร่วมกัน ซึ่งโพรซีเดอร์ที่ใช้

ในการแก้ไขนั้นก็จะแตกต่างกันออกไป

ข้อความที่พบบ่อยเมื่อระบบนั้นเกิดการหยุดทำงาน คือ

FATAL: no response from backend: detected in …

นั่นคืออาจจะมีความหมายดังนี้ : อาจจะมี bug เกิดขึ้นในระบบ หรือ อาจจะเกิด bug ขึ้นที่ตัวโปรแกรมของผู้ใช้ที่ทำการโหลดเข้าสู่ระบบ Postgres คุณควรจะทำการ restart แอพพลิเคชัน และกู้คืนใหม่ แต่ควรจะต้องพิจารณาในส่วนดังต่อไปนี้

- Postgres มักจะทำการ dump core file ลงในไดเรคทอรีของฐานข้อมูล คือ

  /usr/local/postgres95/data/base/<database>/core

  ลงในส่วนของ server ถ้าไม่ต้องการ debug ปัญหาที่เกิดขึ้น สามารถลบในส่วนนี้ออกไปได้ ซึ่งจะช่วยลดเนื้อที่ในการใช้งานระบบลงไปได้ประมาณ 10MB

- ถ้า backend ส่วนใดส่วนหนึ่งหยุดการทำงานขึ้นโดยที่ไม่สามารถควบคุมได้นั้น Postmaster จะต้องค้นหาสาเหตุ, ทำลายโพรเซสที่กำลังทำงานอยู่ทั้งหมด และเริ่มต้นการทำงานใหม่ในส่วนของ backend ถ้าส่วนของ server หยุดการทำงานคุณจะได้รับข้อความแสดงว่า "no response" แต่ถ้าส่วนของ server ถูกทำลายเพราะเกิดจาก server อื่นนั้น คุณจะได้รับข้อความดังนี้

  " I have been signalled by the postmaster. Some backend process has died unexpectedly and possibly corrupted shared memory. The current transaction was aborted, and I am going to exit. Please resend the last query. – The postgres backend."

- บางเวลาสถานะในการใช้งานจะไม่สมบูรณ์ ส่วนของ Frontend application จะแสดงข้อความดังนี้

  WARN: cannot write block 34 of myclass [mydb] blind

  ในกรณีนี้ให้ทำการทำลายการทำงานในส่วนของ Postmaster แล้วทำการ restart ใหม่

- ถ้าระบบหยุดการทำงานในขณะที่ระบบกำลังทำการปรับปรุงในส่วนของ system catalog นั้นส่วนของ B-tree ที่ถูกกำหนดใน catalog จะถูกหยุดการทำงานด้วย และในส่วนของการ Queries ทั้งหมดจะหยุดการทำงานด้วย ดังนั้นให้ลองใช้ในส่วนของคำสั่ง reindexdb ใหม่ แต่ถ้าส่วนของ reindexdb นั้นสำเร็จแต่ส่วนอื่นๆ

ไม่ทำงาน จะต้องทำการกลับไปดึงข้อมูลในส่วนของ system catalog ที่ทำการสำรองไว้นำมาใช้งานใหม่ เพราะปัญหาที่เกิดขึ้นนั้นเป็นปัญหาในส่วนอื่น

การทำงานในส่วนของ Postmaster นั้นจะไม่ค่อยหยุดการทำงานเท่าไรนัก แต่ก็เป็นไปได้ในบางโอกาส จากที่กล่าวมาคือโอกาสในการเกิดปัญหาขึ้นระหว่างที่ทำการ reinitialize ทรัพยากรที่ทำการแบ่งการใช้งานอยู่

คุณอาจจะใช้คำสั่ง ipcclean ถ้าระบบเกิดข้อผิดพลาดมาจาก Postmaster หยุดการทำงาน โดยการใช้ user-id ที่เป็น postgres ทำการจัดการเนื้อที่ในหน่วยความจำใหม่ และถ้ามี postmaster ที่ทำงานอยู่หลายตัวบนเครื่องเดียวกัน คุณควรที่จะทำการทำลายทุกตัวเหล่านั้นก่อนที่จะทำงานกับคำสั่ง ipcclean

2.1 Moving Database Directories

โดยพื้นฐานทั่วไป ระบบฐานข้อมูล Postgres นั้นจะถูกจัดเก็บแยกกันในแต่ละส่วนของสับไดเรคทอรี่ภายใต้ไดเรคทอรี่พื้นฐาน /usr/local/postgres95/data/base แต่ในบางเวลาคุณจะพบว่าถ้าต้องการที่จะย้ายส่วนของข้อมูลไปยังที่อื่นๆ ถ้าต้องการที่จะย้ายระบบฐานข้อมูลทั้งหมดไปยังสถานที่ใหม่ จะต้องทำดังนี้

- ทำลายการทำงานในส่วนของ postmaster ก่อน
- copy ข้อมูลทั้งหมดในไดเรคทอรี่ไปยังไดเรคทอรี่ใหม่ โดย

  % cp -rp /usr/local/postgres95/data/new/place/data
- ทำการ reset ในส่วนสภาวะแวดล้อมของตัวแปรของ PGDATA โดยใช้

  # using csh or tcsh ...

  % setenv PGDATA /new/place/data

  # using sh, ksh or bash

  % PGDATA=/new/place/data; export PGDATA
- restart ในส่วนของ Postmaster ใหม่โดย

  % postmaster &
- หลังจากที่ทำการัน query ในบางส่วนแล้ว และแน่ใจว่าฐานข้อมูลที่เคลื่อนย้ายมาใหม่นั้นสามารถทำงานได้ คุณสามารถทำการลบในส่วนของข้อมูลเดิมได้ โดย

% rm -rf /usr/local/postgres95/data

ในการที่จะลงโปรแกรมลงในฐานข้อมูลเดียวนั้น กระทำดังนี้

- สร้างฐานข้อมูลใหม่ โดยการใช้คำสั่ง createdb โดยในขั้นตอนต่อไปจะกำหนดให้ใช้ชื่อฐานข้อมูล คือ foo
- ทำลายการทำงานในส่วนของ Postmaster
- Copy จากไดเรคทอรี่ /usr/local/postgres95/data/base/foo ไปยังไดเรคทอรีปลายทาง โดยในขณะที่ทำงานจะต้องใช้ user-id Postgres เสมอ

  % cp -rp /usr/local/postgres95/data/base/foo /new/place/foo
- ทำการเคลื่อนย้ายในส่วนของ ไดเรคทอรี่ /usr/local/postgres95/data/base/foo : ออกไป

  % rm -rf /usr/local/postgres95/data/base/foo
- สร้างสัญญลักษณ์ที่ใช้ในการเชื่อมโยงไปยังตำแหน่งใหม่ คือ จาก /usr/local/postgres95/data/base ไปยังไดเรคทอรี่ใหม่ดังนี้

  % ln -s /new/place/foo /usr/local/postgres95/data/base/foo
- ทำการ restart การทำงานของ Postmaster ใหม่

2.2 Updating Database

เมื่อไรก็ตามที่ต้องการปรับปรุงระบบงานของ Postgres ไปเป็นเวอร์ชันใหม่ จะต้องทำดังนี้

- Extensions (เช่น user-defined types, functions, aggregates, etc.) จะต้องถูก reload ใหม่โดยการทำ re-executing ในส่วนของคำสั่ง CREATE ของ SQL
- Data จะต้องถูก dump มาจากคลาสเดิมลงไปใน ASCII file (โดยการใช้คำสั่ง copy) แล้วคลาสใหม่จะถูกสร้างขึ้นมาในฐานข้อมูลใหม่เลย (โดยการใช้คำสั่ง CREATE TABLE) และข้อมูลจะถูกโหลดมาจาก ASCII file
- Rules และ view จะต้องถูกโหลดโดย re-executing คำสั่ง CREATE หลายๆ ตัว

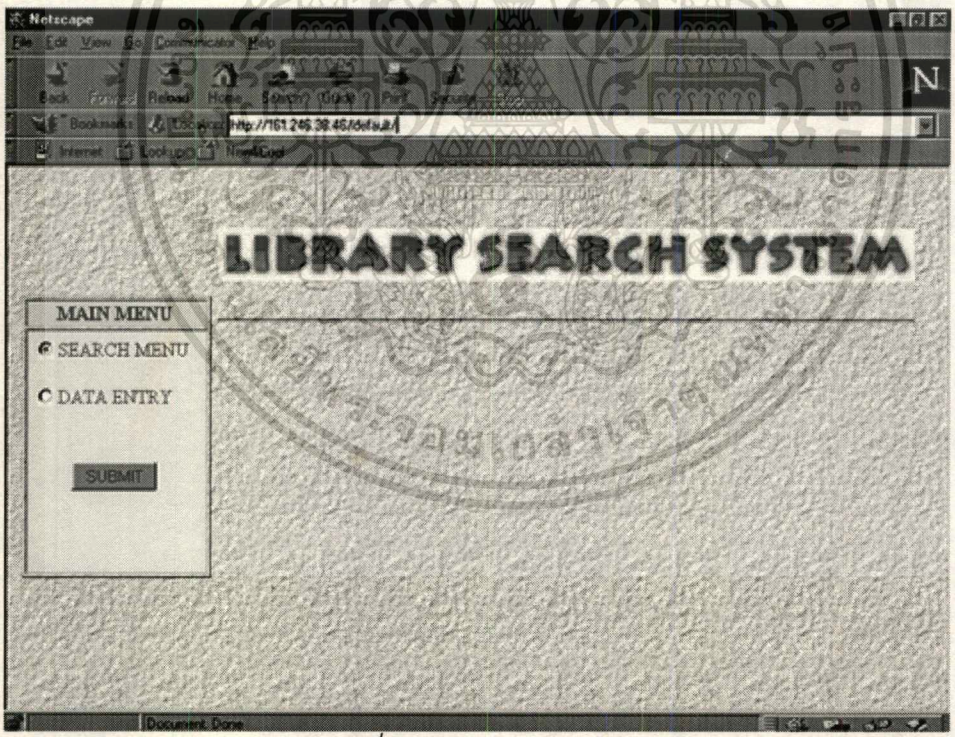
ภาคผนวก ข

รูปแบบระบบงานการสืบหาข้อมูลของระบบงานห้องสมุด

การทำงานภายในระบบงานห้องสมุดแบ่งออกได้ 2 หน้าที่การทำงานดังนี้

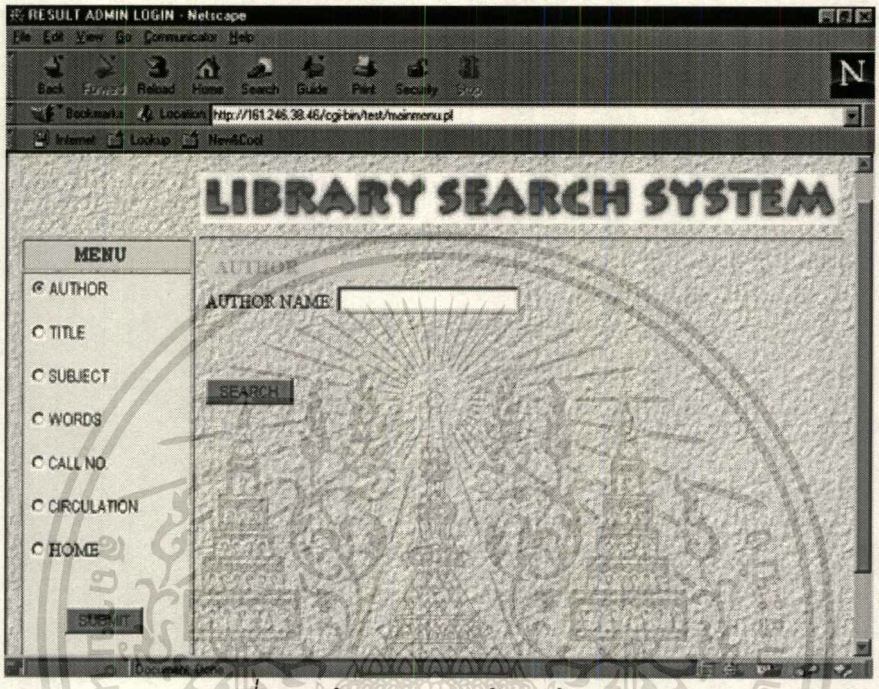
- ลักษณะการสืบหาข้อมูล และ
- ลักษณะการป้อนข้อมูลใหม่เข้าสู่ระบบ

โดยที่การทำงานในส่วนของหน้าจอหลักจะแสดงดังรูป



รูปที่ 9 หน้าจอเมนูหลัก

1. เมื่อเลือกระบบการทำงานในส่วนของการค้นหาข้อมูล (SEARCH MENU) จะแสดงการทำงานดังรูปต่อไปนี้

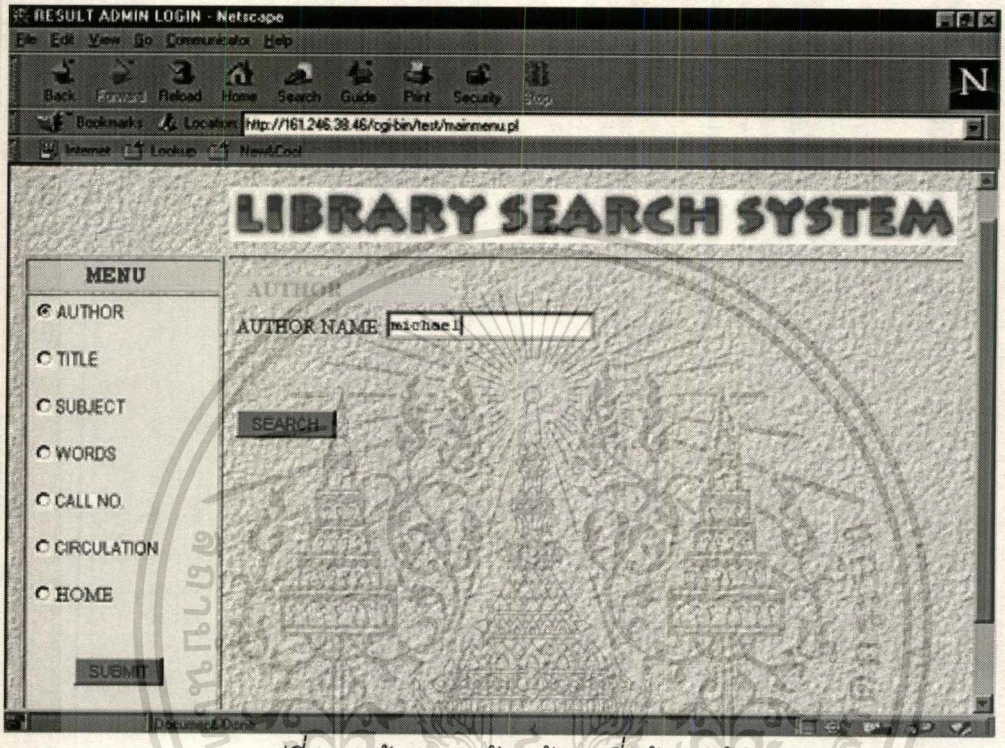


รูปที่ 10 หน้าจอเมนูการค้นหาข้อมูล

ในส่วนของการค้นหาข้อมูลเราสามารถทำการค้นหาได้ใน 6 กรณี คือ

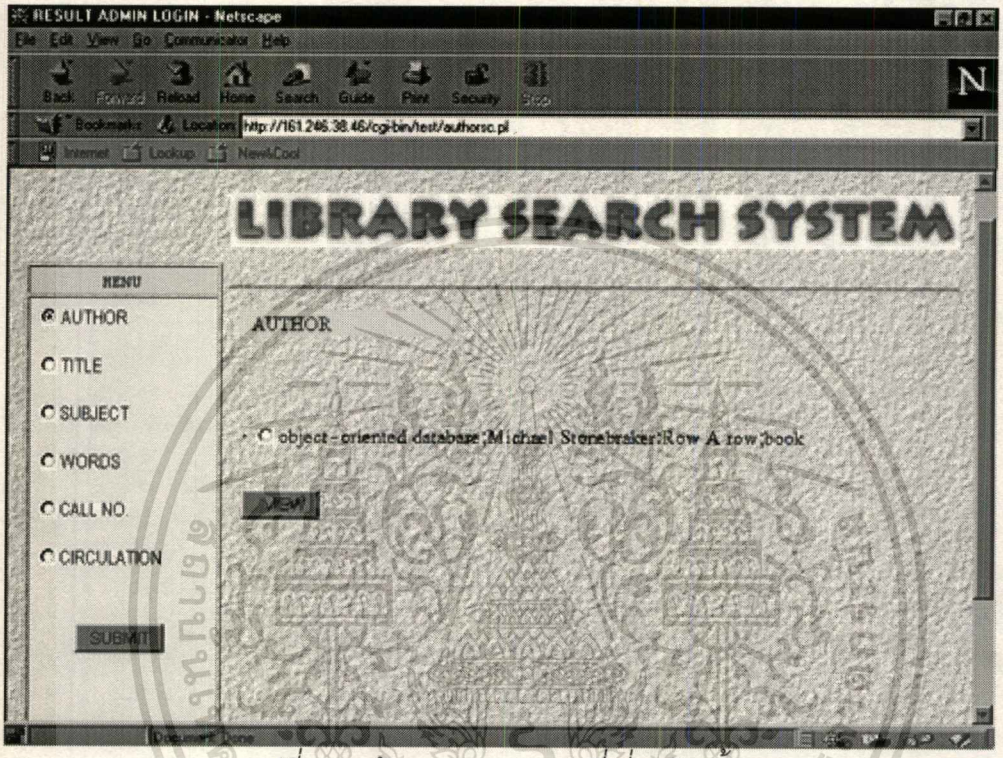
- การค้นหาผ่านทาง Author name
- การค้นหาผ่านทาง Title name
- การค้นหาผ่านทาง Subject name
- การค้นหาผ่านทาง Words
- การค้นหาผ่านทาง Call No.
- การค้นหารายละเอียดการยืม - คืน

ซึ่งตัวอย่างที่แสดงนี้จะเป็นการค้นหาผ่านทาง Author Name ดังรูป



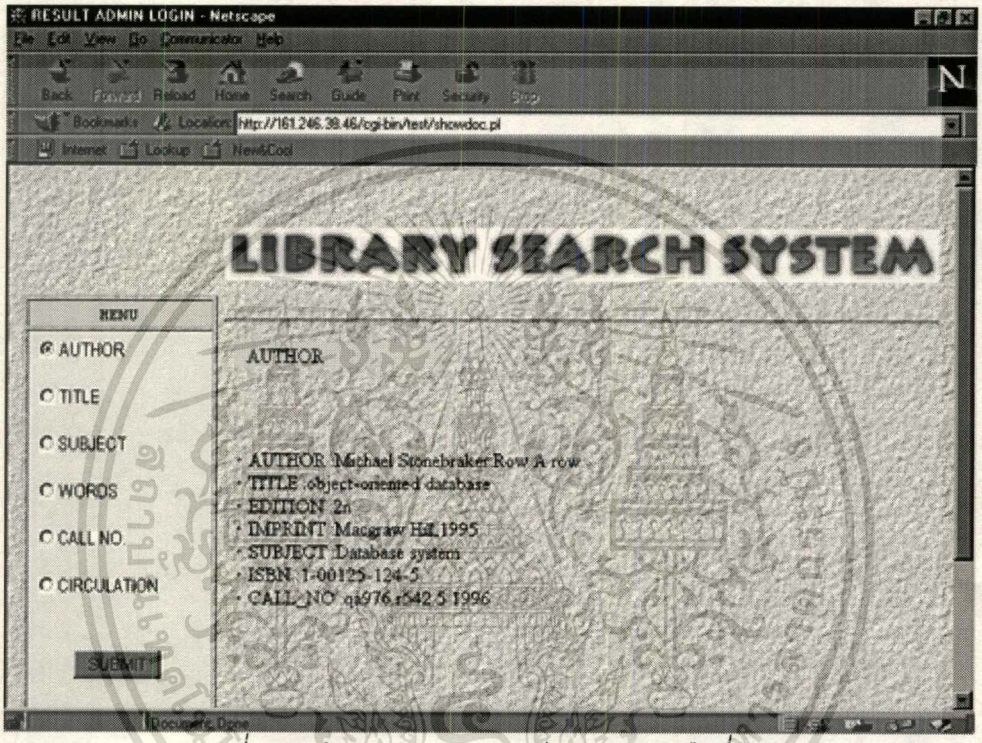
รูปที่ 11 หน้าจอการป้อนข้อมูลที่ต้องการค้นหา

หลังจากที่ผู้ใช้ได้ทำการป้อนข้อมูลในส่วนที่ต้องการค้นหา คือ ชื่อของผู้แต่ง ถ้าค้นหาเจอจะแสดงรายละเอียดดังนี้



รูปที่ 12 หน้าจอแสดงรายการที่เกี่ยวข้องทั้งหมด

ถ้าผู้ใช้ต้องการทำดูรายละเอียดต่างๆ เกี่ยวกับหนังสือที่ต้องการให้ทำการเลือก แล้วจะแสดงรายละเอียดที่ต้องการ ซึ่งจะแสดงดังต่อไปนี้



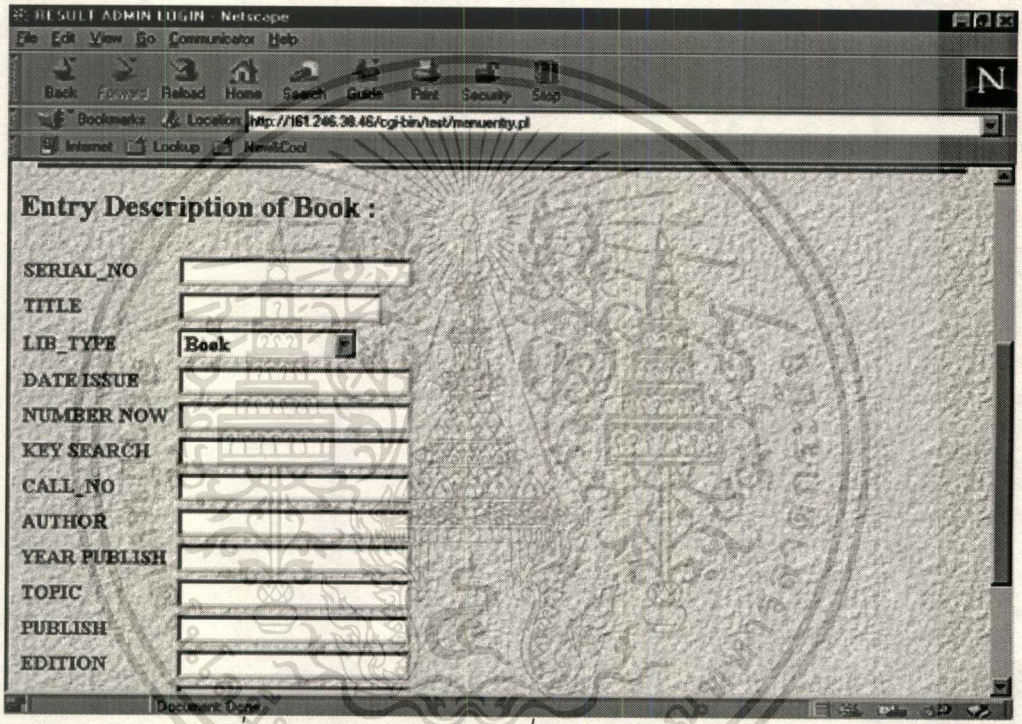
รูปที่ 13 หน้าจอแสดงรายละเอียดของหนังสือที่ต้องการ

ซึ่งการค้นหาในส่วนอื่นๆ นั้นจะมีลักษณะการทำงานที่คล้ายคลึงกันกับในกรณีที่ทำการค้นหาผ่านทาง ชื่อของผู้แต่ง หรือ Author Name

2. เมื่อเลือกระบบการทำงานในส่วนของการป้อนรายละเอียดหนังสือใหม่เข้าสู่ระบบ สามารถกระทำได้ดังนี้ ซึ่งหน้าจอที่แสดงเป็นดังนี้

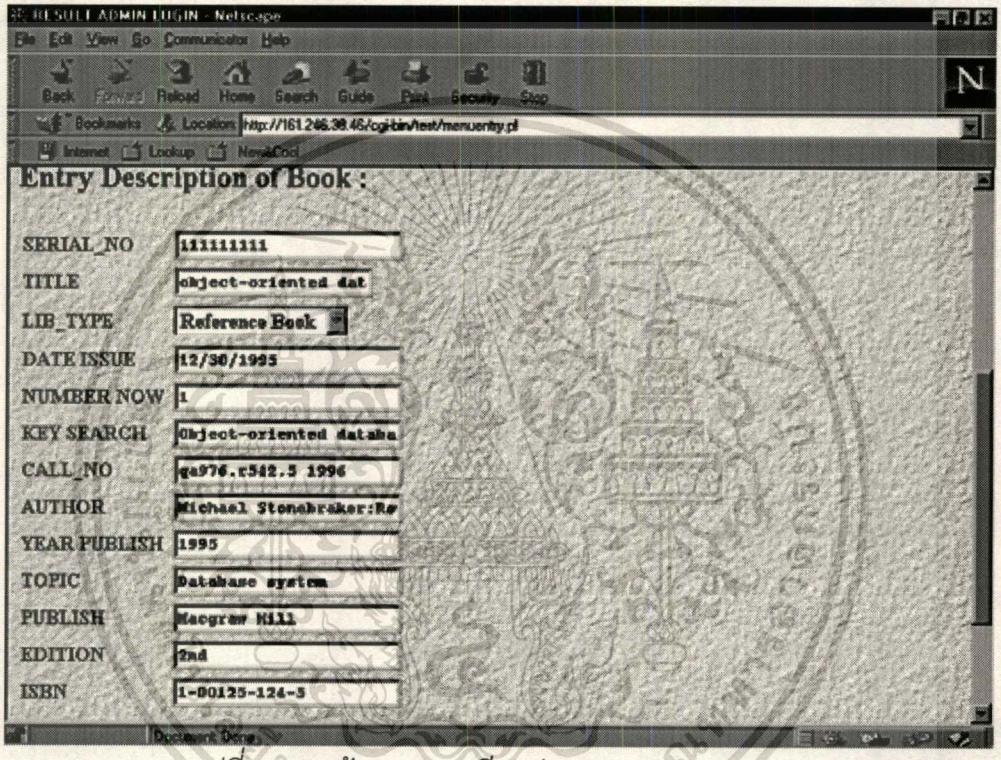
รูปที่ 14 หน้าจอเมนูการเลือกรายการป้อนหนังสือใหม่

ในกรณีนี้ผู้ที่ทำการป้อนข้อมูลสามารถเลือกได้ว่าต้องการป้อนข้อมูลหนังสือ ประเภทใด ระหว่าง หนังสือทั่วไป, Magazine/Journal หรือ Paper ซึ่งถ้าผู้ใช้ต้องการป้อนรายละเอียดของหนังสือเล่มใด ก็ให้ทำการเลือกไปยังประเภทของหนังสือนั้น และกดปุ่มยอมรับการทำงานดังแสดงในรูป



รูปที่ 15 แสดงรายการข้อมูลที่ต้องทำการป้อน

เมื่อผู้ใช้ทำการกรอกรายละเอียดต่างๆ ครบทุกช่องแล้วให้กดปุ่ม summit เพื่อเป็นการยอมรับการป้อนข้อมูลเข้าสู่ระบบ ดังแสดงในรูปแล้วเมนูการทำงานจะกลับมาอยู่ที่การทำงานในส่วนของการรอรับการป้อนข้อมูลใหม่อีกรอบ



รูปที่ 16 การป้อนรายละเอียดต่างๆ

ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาวนันทวัน นาคอร่าม
วันเดือนปีเกิด	24 ตุลาคม 2516
สถานที่เกิด	จังหวัดพระนครศรีอยุธยา
วุฒิการศึกษาระดับปริญญาตรี	วท.บ.(วิทยาการคอมพิวเตอร์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2538
ประสพการณ์การทำงาน	2538 บริษัทไทยซัมมิทออโต้พาร์ท จำกัด
	2539 เจ้าหน้าที่ศูนย์บริการคอมพิวเตอร์
	มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ
อาชีพปัจจุบัน	อาจารย์พิเศษประจำคณะวิทยาศาสตร์
	มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ