ภาษาเอสคิวแอลเชิงวัตถุสัมพันธ์สำหรับ PostgreSQL

POSTGRESQL OBJECT-RELATIONAL SQL

ณัฐวิชช์ อนุเมธางกูร
ณัฐวุฒิ กวีวัจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ภาษาเอสคิวแอลเชิงวัตถุสัมพันธ์สำหรับ PostgreSQL

PostgreSQL Object-Relational SQL

ผู้จัดทำ

1. นายณัฐวิชช์ อนุเมธางกูร รหัสนักศึกษา 47010238
2. นายณัฐวุฒิ กวีวัจน์ รหัสนักศึกษา 47010239

อาจารย์ที่ปรึกษา

(รศ. ดร. ศุภมิตร จิตตะยโศธร)

ภาษาเอสคิวแอลเชิงวัตถุสัมพันธ์สำหรับ PostgreSQL

นายณัฐวิชช์ อนุเมธางกูร 47010238
นายณัฐวุฒิ กวีวัจน์ 47010239
รศ. ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2550

บทคัดย่อ

ปัจจุบันได้มีเทคโนโลยีระบบฐานข้อมูลแบบใหม่ ที่ได้นำแนวความคิด และข้อดีของระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) และระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database System) เข้ามาประยุกต์ร่วมกันเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database System) ซึ่งกำลังได้รับความสนใจ และมีแนวโน้มที่จะได้รับความนิยมอย่างสูงในอนาคต ทำให้บริษัทผู้พัฒนาซอฟท์แวร์ระบบการจัดการฐานข้อมูลหลายแห่งได้เพิ่มเติมความสามารถในการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์นี้ให้กับผลิตภัณฑ์ของตนเอง โดยอ้างอิงตามมาตรฐาน SQL3 ที่ถูกกำหนดขึ้นโดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI : American National Standards Institute) ในโครงงานนี้จะได้ศึกษาทฤษฎีและสถาปัตยกรรมเบื้องต้นของมาตรฐานระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

PostgreSQL Object-Relational SQL

Mr. Nuttawit	Anumathangkul	47010238
Mr. Nuttawoot	Kaveewat	47010239
Assoc.Prof. Dr. Supamit Chittayasothon		Advisor

Academic Year 2007

ABSTRACT

Nowadays, the new database technology, Object-Relational database, is emerged by the combination of Object database and Relational database concepts. This trend is interested by DBMS vendors to improve their own products to support the use of object datatype. Although the ANSI SQL 3 standard is incomplete, the improvement of some vendors was based on their own technology. In case of this project, is studied about the object extension which is major volume of SQL 3 standard by using PostgreSQL for educating Object-Relational database.

กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้สำเร็จได้ก็เนื่องด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.ศุภมิตร จิตตะยโศธร ที่คอยให้คำแนะชี้แนะช่วยแก้ไขปัญหาตลอดจนให้ความรู้และประสบการณ์ที่เป็นประโยชน์แก่ข้าพเจ้า ซึ่งท่านต้องสละเวลาอันมีค่ามาตรวจสอบความคืบหน้า อีกทั้งท่านยังเป็นแรงแรงผลักดันที่สำคัญที่ทำให้วิทยานิพนธ์เล่มนี้เสร็จสมบูรณ์ในที่สุด

ขอกราบขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกๆ ท่านที่ได้กรุณาประสิทธิ์ประสาทวิชาความรู้กับข้าพเจ้า

ขอขอบคุณทางภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้สนับสนุนปัจจัยและสถานที่ในการทำโครงงาน รวมถึงแหล่งค้นหาความรู้ต่างๆที่จัดให้

สุดท้ายนี้ ข้าพเจ้าขอขอบคุณ คุณพ่อ คุณแม่ และทุกคนในครอบครัว หากไม่ได้รับการสนับสนุนจากพวกท่าน ข้าพเจ้าคงไม่สามารถมาถึง ณ จุดนี้ได้

ณัฐวิชช์ อนุเมธางกูร
ณัฐวุฒิ กวีวัจน์

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมาของโครงงาน

ปัจจุบันเทคโนโลยีสารสนเทศได้เข้ามามีบทบาทสำคัญในการดำเนินกิจกรรมและระบบงานต่างๆ ของแต่ละองค์กร ซึ่งการจะทำให้ระบบสารสนเทศเหล่านั้นมีประสิทธิภาพเพียงพอที่จะสนับสนุนระบบงานได้นั้นจำเป็นจะต้องมีการจัดการข้อมูลอย่างมีประสิทธิภาพ ด้วยสาเหตุเหล่านี้จึงเป็นที่มาของการพัฒนาระบบการจัดการฐานข้อมูลขึ้น โดยในอดีตที่ผ่านมาได้มีการค้นคว้าและพัฒนาระบบฐานข้อมูลขึ้นหลายรูปแบบและหลายแนวความคิด อย่างไรก็ตามระบบฐานข้อมูลที่ได้รับความนิยมและแพร่หลายสูงสุดในการใช้งาน คือ ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) ในขณะเดียวกันหลักการคิดและมุมมองข้อมูลในรูปแบบเชิงวัตถุทำให้การพัฒนาโปรแกรมส่วนใหญ่ในยุคปัจจุบันล้วนแต่ยึดหลักการโปรแกรมเชิงวัตถุมากขึ้น จนหลักการดังกล่าวได้เริ่มเข้ามามีบทบาทสำคัญในการพัฒนาระบบฐานข้อมูลเชิงวัตถุ (Object-Database) ซึ่งจะสามารถรองรับการใช้งานข้อมูลที่มีความซับซ้อนได้มากกว่าฐานข้อมูลเชิงสัมพันธ์ ดังนั้นจึงได้มีการพัฒนาเทคโนโลยีระบบฐานข้อมูลแบบใหม่ที่ได้นำแนวความคิดและข้อดีของระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) และระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database System) เข้ามาประยุกต์ร่วมกันเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database System) ซึ่งกำลังได้รับความสนใจและมีแนวโน้มที่จะได้รับความนิยมอย่างสูงในอนาคต ทำให้บริษัทผู้พัฒนาซอฟท์แวร์ระบบการจัดการฐานข้อมูลหลายแห่งได้เพิ่มเติมความสามารถในการจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์นี้ให้กับผลิตภัณฑ์ของตนเองโดยอ้างอิงตามมาตรฐาน SQL3 ที่ถูกกำหนดขึ้นโดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI: American National Standards Institute) ในโครงงานนี้จะได้อ้างถึงทฤษฎีและสถาปัตยกรรมของมาตรฐานระบบฐานข้อมูลเชิงวัตถุสัมพันธ์พร้อมทั้งทำการศึกษากับผลิตภัณฑ์ซอฟท์แวร์ระบบการจัดกาฐานข้อมูลที่สนับสนุนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยได้เลือกใช้ระบบจัดการฐานข้อมูล PostgreSQL ที่ถือว่าเป็นผลิตภัณฑ์ระบบการจัดการฐานข้อมูล Open Source ที่มีประสิทธิภาพและได้รับความนิยมอย่างสูง

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database)

1.2.2 เพื่อศึกษาซอฟต์แวร์ที่ใช้จัดการระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

1.2.3 เพื่อศึกษาความก้าวหน้าของฐานข้อมูลเชิงวัตถุสัมพันธ์

1.2.4 เพื่อทดสอบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) ตามแนวความคิดเชิงวัตถุสัมพันธ์โดยใช้ซอฟต์แวร์ที่ใช้จัดการระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

1.2.5 เพื่อศึกษาการออกแบบและทำการพัฒนาโปรแกรมประยุกต์ที่เหมาะสมกับโครงสร้างข้อมูลเชิงวัตถุสัมพันธ์

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้เกี่ยวกับแนวคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์

1.3.2 ได้รับความรู้เกี่ยวกับการใช้ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

1.3.3 สามารถวิเคราะห์และออกแบบฐานข้อมูลเชิงวัตถุสัมพันธ์ได้

1.3.4 ได้รับความรู้เกี่ยวกับเทคโนโลยีใหม่ๆ ของฐานข้อมูลเชิงวัตถุสัมพันธ์

1.3.5 สามารถนำระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL มาประยุกต์ใช้งานตามแนวความคิดของของฐานข้อมูลเชิงวัตถุสัมพันธ์ได้

1.4 ขอบเขตของโครงงาน

ศึกษาแนวคิดเชิงวัตถุสัมพันธ์ ศึกษาฟังก์ชันการทำงานและความสามารถต่างๆ ที่มีมาให้กับระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL และศึกษาโปรแกรมประยุกต์ที่สนับสนุนระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL เช่น psql, Phppgadmin, pgadmin3 แล้วทดสอบสิ่งที่ศึกษามาทั้งหมดด้วยการติดต่อกับฐานข้อมูลเชิงวัตถุสัมพันธ์ด้วยโปรแกรมประยุกต์ที่สนับสนุนระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยเริ่มจากการเก็บข้อมูลที่มีลักษณะเป็น Atomic data type แล้วพัฒนาขึ้นโดยเก็บข้อมูลที่มีความซับซ้อน (Complex Data Types) จากนั้นก็ทำการสร้างชนิดของข้อมูลที่ผู้ใช้สร้างขึ้นเองได้ (User Defined Type) พร้อมทั้งสร้างฟังก์ชัน (User-Defined Function), โอเปอร์เรเตอร์ เพื่อรองรับกับชนิดของข้อมูลที่ผู้ใช้สร้างขึ้นตามแนวความคิดเชิงวัตถุ

1.5 วิธีการดำเนินการงาน

1.5.1 ศึกษาคุณลักษณะ คุณสมบัติ และการออกแบบฐานข้อมูลเชิงสัมพันธ์

1.5.2 ศึกษาคุณลักษณะ คุณสมบัติ และการออกแบบฐานข้อมูลเชิงวัตถุ

1.5.3 ศึกษาซอฟต์แวร์ที่สนับสนุนระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

1.5.4 ศึกษาฟังก์ชันการทำงานและความสามารถต่างๆ ที่มีมาให้กับระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

1.5.5 ทำการออกแบบและพัฒนาโปรแกรมประยุกต์ที่เหมาะสมกับโครงสร้างข้อมูลเชิงวัตถุสัมพันธ์

1.6 ส่วนประกอบของรายงาน

โครงงานฉบับนี้ประกอบไปด้วยเนื้อหาทั้งหมด 8 บท ซึ่งสามารถแบ่งตามเนื้อหาได้ดังนี้

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน, วัตถุประสงค์ของโครงงาน, ขอบเขตในการดำเนินงาน, วิธีการดำเนินงาน, ประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 กล่าวถึงแนวความคิดของฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

บทที่ 3 กล่าวถึงแนวความคิดของฐานข้อมูลเชิงวัตถุ (Object Database)

บทที่ 4 กล่าวถึงแนวความคิดของฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database)

บทที่ 5 แนะนำระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ PostgreSQL

บทที่ 6 กล่าวถึงการออกแบบโปรแกรมประยุกต์ที่มีการใช้งานฐานข้อมูล

บทที่ 7 กล่าวถึงการออกแบบโปรแกรมประยุกต์ที่มีการใช้งานฐานข้อมูลสัมพันธ์และฐานข้อมูลเชิงวัตถุสัมพันธ์ พร้อมทั้งอธิบายการใช้งานโปรแกรมประยุกต์ที่ได้ทำการออกแบบขึ้น

บทที่ 8 จะเป็นการสรุปผลโครงงาน พร้อมทั้งแนวทางในการพัฒนาและประยุกต์ใช้

บทที่ 2
ฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

พื้นฐานของเทคโนโลยีระบบฐานข้อมูลเชิงสัมพันธ์ได้เกิดขึ้นจากทฤษฎีทางคณิตศาสตร์ โดย E.F. Codd ซึ่งต่อมาบริษัท IBM และบริษัทต่างๆ ได้นำมาพัฒนาเป็นผลิตภัณฑ์ของตน ในภายหลังได้มีการกำหนดมาตรฐานของระบบฐานข้อมูลเชิงสัมพันธ์ โดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI: American National Standards Institute) และรับรองโดยองค์การมาตรฐานนานาชาติ (ISO: International Standards Organization) เรียกว่า มาตรฐาน SQL โดยจะมีหมายเลขเวอร์ชันของมาตรฐานกำกับอยู่ด้านท้าย ตัวอย่างเช่น SQL 2 ซึ่งถือว่าเป็นมาตรฐานล่าสุดที่ได้เสร็จสมบูรณ์แล้ว เป็นต้น

2.1 โครงสร้างข้อมูล

โมเดลเชิงสัมพันธ์ใช้รูปแบบของความสัมพันธ์ที่ชัดเจนมาเป็นตัวกำหนดลักษณะของข้อมูล โมเดลเชิงสัมพันธ์ได้ถูกนำไปใช้อย่างแพร่หลายในระบบการประมวลผลข้อมูล โครงสร้างข้อมูลของระบบฐานข้อมูลเชิงสัมพันธ์จะจัดเก็บข้อมูลเป็น Relation ที่สามารถแสดงได้ในรูปแบบของตาราง (Table) ดังนั้น

- ชื่อของตารางคือชื่อของความสัมพันธ์
- แต่ละคอลัมน์ของตารางความสัมพันธ์ เรียกว่า แอททริบิว (Attribute) ของความสัมพันธ์ ซึ่งแต่ละคุณสมบัติจะมีชื่อเรียกและค่าของแอททริบิวที่แตกต่างกัน ค่าและขอบเขตของ ข้อมูลของแอททริบิวเรียกว่าโดเมน (Domain) เช่น ความสัมพันธ์ "player" ประกอบ ด้วยแอททริบิว 5 อย่าง คือ ชื่อ ตำแหน่ง อายุ ส่วนสูง น้ำหนัก
- แต่ละแถวของตารางความสัมพันธ์ เรียกว่า แถว หรือ ทูเพิล (Tuple) ของความสัมพันธ์ จากที่กล่าวมาข้างต้น สรุปโดยใช้หลักการทางคณิตศาสตร์ได้ว่า
  - โดเมน D คือ ชุดของข้อมูลที่มีค่าเฉพาะเจาะจง และมีความหมายเป็นอย่างใดอย่างหนึ่ง
  - แอททริบิว A คือ ตัวแทนของโดเมน เขียนได้เป็น dom(A) เมื่อกล่าวถึงแอททริบิว A จะหมายความถึงชุดของข้อมูลชุดหนึ่งที่เป็นที่รู้กันว่ามีขอบเขตเพียงไร

- รูปแบบของความสัมพันธ์ R ซึ่งสามารถเขียนได้เป็น R(A1, A2, ..., An) คือ ชุดของแอททริบิว โดยที่เมื่อกล่าวถึง R จะหมายถึง {A1, A2, ..., An} โดยปริยาย
- ดังนั้น ความสัมพันธ์ r คือ กลุ่มของแถวที่เป็น subset ของกลุ่มของแอทริบิว dom(A1) x dom(A2) x ... x dom(An)
- หนึ่งแถวในความสัมพันธ์ r จึงเขียนได้เป็น t(A1, A2, ..., An) และมีความหมายเป็น subset หนึ่งของ dom(A1) x dom(A2) x ... x dom(An)

รูปแบบทางคณิตศาสตร์ของความสัมพันธ์ดังข้างต้นได้ถูกนำไปใช้ในการอธิบายรูปแบบของความสัมพันธ์ซึ่งเป็นการกำหนดชนิดและความหมายของข้อมูลในฐานข้อมูลสิ่งที่มักทำให้เกิดความสับสนก็คือ รูปแบบของความสัมพันธ์ (Relational Schema) และตัวความสัมพันธ์ (Relational Instance) รูปแบบของความสัมพันธ์เป็นการกำหนดลักษณะโครงสร้างของความสัมพันธ์ แต่ความสัมพันธ์เป็นข้อมูลที่เกิดความสัมพันธ์กันของแอททริบิว

คุณสมบัติของความสัมพันธ์

- ลำดับของแถวและคอลัมน์ไม่ทำให้ความหมายของข้อมูลเปลี่ยนไป
- แต่ละแถวในความสัมพันธ์จะเป็นเอกลักษณ์ คือจะไม่มีสองแถวที่ซ้ำกัน
- แอททริบิวทุกตัวจะเป็น atomic เท่านั้น ไม่มีแอททริบิวที่เป็น Multivalve หรือ Composite
- ดีกรีของความสัมพันธ์ คือ จำนวนแอททริบิวที่มีในความสัมพันธ์นั้น

คุณสมบัติหนึ่งที่สำคัญของความสัมพันธ์ก็คือ ความเป็นเอกลักษณ์ (Uniqueness Property) สิ่งที่ใช้กำหนดความเป็นเอกลักษณ์ของแถวในความสัมพันธ์ เรียกว่า คีย์ (Key)

ฐานข้อมูลหนึ่งๆ จะมีข้อมูลอยู่มากมาย ยิ่งฐานข้อมูลมีขนาดใหญ่ขึ้นก็จะมีข้อมูลจำนวนมากขึ้นเป็นเงาตามตัวข้อมูลเหล่านี้อาจมีค่าแตกต่างกัน คล้ายกัน หรือแม้กระทั่งเหมือนกัน ทำให้การแยกแยะโดยอาศัยเพียงตัวข้อมูลอย่างเดียวทำได้อย่างยากลำบาก ดังนั้นจึงมีการกำหนดค่า Key ประจำข้อมูลเพื่อทำให้การแยกแยะข้อมูลในฐานข้อมูลเป็นไปอย่างถูกต้อง

คีย์มีหลายประเภท ได้แก่ คีย์หลัก, Secondary Key, Foreign vey, Candidate Key, Super Key ดังจะได้กล่าวในรายละเอียดต่อไป

1. คีย์หลัก (Primary Key)

คีย์หลัก คือ Key หลักที่ใช้ในการอ้างถึง Entity ในฐานข้อมูล การเลือกคีย์หลักสามารถเลือกได้จาก Record ใดๆ ก็ได้ที่ไม่มีโอกาสซ้ำซ้อนกันบนฐานข้อมูลนั้น

คีย์หลักเป็นข้อมูลสำคัญที่จะทำให้การเข้าถึงข้อมูลบนฐานข้อมูลเป็นไปได้อย่างรวดเร็ว ดังนั้นผู้ใช้จึงควรกำหนดคีย์หลักให้ชัดเจนตั้งแต่ขั้นตอนออกแบบฐานข้อมูล หากไม่มีข้อมูลใดเลยในฐานข้อมูลที่เหมาะที่จะเป็นคีย์หลักก็ควรที่จะกำหนด Record ใหม่สำหรับให้เป็นคีย์หลักโดยเฉพาะ

2. คีย์รอง (Secondary Key)

คีย์สำรอง คือ คีย์เดี่ยวหรือคีย์ผสม (Single or Composite Key) ซึ่งเมื่อใช้ในการค้นหาข้อมูลจากความสัมพันธ์จะได้มากกว่าหนึ่งเรคคอร์ด ต่างจากคีย์หลักที่ทำให้ข้อมูลในตารางไม่ซ้ำกัน ดังนั้นคีย์รองจึงไม่จำเป็นจะต้องเป็นเอกลักษณ์

3. คีย์นอก (Foreign key)

คีย์นอก คือ คีย์เดี่ยวหรือคีย์ผสม ซึ่งปรากฏเป็นคีย์ทั่วไปของความสัมพันธ์หนึ่งแต่ไปปรากฏเป็นอีกคีย์หลักในอีกความสัมพันธ์หนึ่ง คีย์นอกเป็นอีกคีย์หนึ่งที่มีความสำคัญมากในฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเป็นตัวที่ใช้สร้างการเชื่อมต่อระหว่างความสัมพันธ์ การเปลี่ยนแปลงค่าของคีย์นอกจะต้องอาศัยความระมัดระวังเป็นอย่างมาก เนื่องจากจะมีผลกระทบโดยตรงต่อข้อมูลในความสัมพันธ์อื่นที่มีการอ้างอิงถึงคีย์นอกตัวนี้ จึงมีกฎและเงื่อนไขที่บังคับใช้เพื่อทำให้ข้อมูลมีความถูกต้องอยู่เสมอ

4. ซุปเปอร์คีย์ (Superkey)

คือกลุ่มของแอททริบิวที่สามารถนำไปใช้ในการค้นหาข้อมูลที่เป็นเอกลักษณ์ได้

5. คีย์แข่งขัน (Candidate key)

คีย์แข่งขัน ก็คือ ซุปเปอร์คีย์และไม่มีกลุ่มย่อยของคีย์ใดในคีย์แข่งขันที่จะสามารถเป็นซุปเปอร์คีย์ได้อีก

สำหรับความสัมพันธ์ หรือ Relationship ระหว่างข้อมูลแต่ละตารางจะถูกแสดงโดยใช้การจัดเก็บ Primary Key จากอีกตารางหนึ่งเป็น Foreign Key ในอีกตารางหนึ่งและต้องสร้างตารางเสริมพิเศษเพื่อจัดการกับความสัมพันธ์ของข้อมูลแบบ Many-to-Many ระหว่างตาราง

2.2 ภาษาที่ใช้ในการจัดการฐานข้อมูล

ภาษา SQL (สามารถอ่านออกเสียงได้ 2 แบบ คือ "เอสคิวแอล" (SQL) หรือ "ซีเควล" (Sequel)) ย่อมาจาก Structured Query Language หรือภาษาในการสอบถามข้อมูล เป็นภาษาทางด้านฐานข้อมูลที่สามารถสร้างและปฏิบัติการกับฐานข้อมูลแบบสัมพันธ์ (Relational Database)

โดยเฉพาะ และ เป็นภาษาที่มีลักษณะคล้ายกับภาษาอังกฤษ ภาษา SQLถูกพัฒนาขึ้นจากแนวคิดของ Relational Calculus และ Relational Algebra เป็นหลัก ภาษา SQL เริ่มพัฒนาครั้งแรกโดย Almaden Research Center ของบริษัท IBM โดยมีชื่อเริ่มแรกว่า "ซีเควล" (Sequel) ต่อมาได้เปลี่ยนชื่อเป็น"เอสคิวแอล" (SQL) หลังจากนั้นภาษา SQLได้ถูกนำมาพัฒนาโดยผู้ผลิตซอฟแวร์ด้านระบบจัดการฐานข้อมูลเชิงสัมพันธ์จนเป็นที่นิยมกันอย่างแพร่หลายในปัจจุบันโดยผู้ผลิตแต่ละรายก็พยายามที่จะพัฒนาระบบจัดการฐานข้อมูลของตนให้มีลักษณะเด่นเฉพาะขึ้นมา ทำให้รูปแบบการใช้คำสั่ง SQL มีรูปแบบที่แตกต่างกันไปบ้าง เช่น ORACLE ACCESS SQL Base ของ Sybase INGRES หรือ SQL Server ของ Microsoft เป็นต้น ดังนั้นในปี ค.ศ. 1986 ทางด้าน American National Standards Institute (ANSI) จึงได้กำหนดมาตรฐานของ SQL ขึ้น อย่างไรก็ดี โปรแกรมฐานข้อมูลที่ขายในท้องตลาด ได้ขยาย SQL ออกไปจนเกินข้อกำหนดของ ANSI โดยเพิ่มคุณสมบัติอื่นๆ ที่คิดว่าเป็นประโยชน์เข้าไปอีกแต่โดยหลักทั่วไปแล้วก็ยังปฏิบัติตามมาตรฐานของ ANSI ในการอธิบายคำสั่งต่างๆของภาษา SQL

1. ประเภทของคำสั่งของภาษา SQL

ภาษา SQL เป็นภาษาที่ใช้งานได้ตั้งแต่ระดับเครื่องคอมพิวเตอร์ส่วนบุคคลพีซีไปจนถึงระดับเมนเฟรม ประเภทของคำสั่งในภาษา SQL (The Subdivision of SQL) แบ่งออกเป็น 3 ประเภท คือ

1. ภาษาสำหรับการนิยามข้อมูล (Data Definition Language: DDL) ประกอบด้วยคำสั่งที่ใช้ในการกำหนดโครงสร้างข้อมูลว่ามีคอลัมน์อะไร แต่ละคอลัมน์เก็บข้อมูลประเภทใด รวมถึงการเพิ่มคอลัมน์ การกำหนดดัชนี การกำหนดวิวหรือตารางเสมือนของผู้ใช้ เป็นต้น

2. ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language: DML) ประกอบด้วยคำสั่งที่ใช้ในการเรียกใช้ข้อมูล การเปลี่ยนแปลงข้อมูล การเพิ่มหรือลบข้อมูล เป็นต้น

3. ภาษาควบคุม (Data Control Language: DCL): ประกอบด้วยคำสั่งที่ใช้ในการควบคุม การเกิดภาวะพร้อมกัน หรือการป้องกันการเกิดเหตุการณ์ที่ผู้ใช้หลายคนเรียกใช้ข้อมูลพร้อมกัน และคำสั่งที่เกี่ยวข้องกับการควบคุมความปลอดภัยของข้อมูลด้วยการกำหนดสิทธิของผู้ใช้ที่แตกต่างกัน เป็นต้น

2. ชนิดของข้อมูลที่ใช้ในภาษา SQL

ในภาษา SQL การบรรจุข้อมูลลงในคอลัมน์ต่าง ๆ ของตารางจะต้องกำหนดชนิดของข้อมูล (Data Type) ให้แต่ละคอลัมน์ ชนิดของข้อมูลนี้จะแสดงชนิดของค่าที่อยู่ในคอลัมน์ ค่าทุกค่าในคอลัมน์ที่กำหนดจะต้องเป็นชนิดเดียวกัน เช่น ในตารางลูกค้าคอลัมน์ที่เป็นรายชื่อลูกค้า จะต้อง

เป็นตัวหนังสือ ในขณะที่คอลัมน์จำนวนเงินที่ลูกค้าซื้อสินค้าเป็นตัวเลขชนิดของข้อมูลของแต่ละคอลัมน์จะขึ้นกับลักษณะของข้อมูลแต่ละคอลัมน์ ซึ่งแบ่งได้ดังนี้ชนิดข้อมูลพื้นฐานในภาษา SQL ดังนี้

1. ตัวหนังสือ (character) ในภาษา SQL จะใช้

- ตัวหนังสือแบบความยาวคงที่ (Fixed-Length Character) จะใช้ char (n) หรือ character(n) แทนประเภทของข้อมูลที่เป็นตัวหนังสือใดๆที่มีความยาวของข้อมูลคงที่โดยมีความยาว n ตัวหนังสือประเภทนี้จะมีการจองเนื้อที่ตามความยาวที่คงที่ตามที่กำหนดไว้ ชนิดของข้อมูลประเภทนี้จะเก็บความยาวของข้อมูลได้มากที่สุดได้ 255 ตัวอักษร

- ตัวหนังสือแบบความยาวไม่คงที่ (Variable-Length Character) จะใช้ varchar (n) แทนประเภทของข้อมูลที่เป็นตัวหนังสือใดๆที่มีความยาวของข้อมูลไม่คงที่ โดยมีความยาว n ตัวหนังสือประเภทนี้จะมีการจองเนื้อที่ตามความยาวของข้อมูล ชนิดของข้อมูลประเภทนี้จะเก็บความยาวของข้อมูลได้มากที่สุดได้ 4000 ตัวอักษร

2. จำนวนเลข (numeric)

- จำนวนเลขที่มีจุดทศนิยม (Decimal) ในภาษา SQL จะใช้ dec(m,n) หรือ decimal(m,n) เป็นประเภทข้อมูลที่เป็นจำนวนเลขที่มีจุดทศนิยมโดย m คือจำนวนตัวเลขทั้งหมด (รวมจุดทศนิยม) และ n คือจำนวนตัวเลขหลังจุดทศนิยม

- จำนวนเลขที่ไม่มีจุดทศนิยมในภาษา SQL จะใช้ int หรือ Integer เป็นเลขจำนวนเต็มบวกหรือลบขนาดใหญ่ เป็นตัวเลข 10 หลัก ที่มีค่าตั้งแต่ –2,147,483,648 ถึง +2,147,483,647 และในภาษา SQL จะใช้ smallint เป็นประเภทข้อมูลที่เป็นเลขจำนวนเต็มบวกหรือลบขนาดเล็ก เป็นตัวเลข 5 หลัก ที่มีค่าตั้งแต่ –32,768 ถึง + 32,767 ตัวเลขจำนวนเต็มประเภทนี้จะมีการจองเนื้อที่น้อยกว่าแบบ Integer

- เลขจำนวนจริง ในภาษา SQL อาจใช้ number(n)แทนจำนวนเลขที่ไม่มีจุดทศนิยมและจำนวนเลขที่มีจุดทศนิยม

2.3 ข้อมูลในลักษณะอื่นๆ

- วันที่และเวลา (Date/Time) เป็นชนิดวันที่หรือเวลาในภาษา SQL จะใช้ date เป็นข้อมูลวันที่ ซึ่งจะมีหลายรูปแบบให้เลือกใช้ เช่น yyyy-mm-dd (1999-10-31) dd.mm.yyyy(31. 10.1999) หรือ dd/mm/yyyy (31/10/1999)

3. ลักษณะการใช้งานของภาษา SQL

ภาษา SQL เป็นส่วนประกอบหนึ่งของ DBMS มักพบใน DBMS เชิงสัมพันธ์หลายตัวและเป็นที่นิยมใช้ในปัจจุบัน ภาษา SQL ง่ายต่อการเรียนรู้ การใช้งานในภาษา SQLแบ่งเป็น 2 ลักษณะ คือ

1. ภาษา SQL ที่โต้ตอบได้ ใช้เพื่อปฏิบัติงานกับฐานข้อมูลโดยตรง เป็นการใช้คำสั่งภาษา SQL สั่งงานบนจอภาพ โดยเรียกดูข้อมูลได้โดยตรงในขณะที่ทำงานเพื่อให้ได้ผลลัพธ์ที่นำไปใช้ได้

2. ภาษา SQL ที่ฝังในโปรแกรม เป็นภาษา SQL ที่ประกอบด้วยคำสั่งต่าง ๆ ของภาษา SQL ที่ใส่ไว้ในโปรแกรมที่ส่วนมากแล้วเขียนด้วยภาษาอื่น เช่น โคบอล ปาสคาล ภาษาซี ลักษณะของคำสั่ง SQL จะแตกต่างจากภาษาอื่นๆ ในแง่ที่ว่า SQL ไม่มีคำสั่งที่เกี่ยวกับการควบคุม (Control Statement) เหมือนภาษาอื่น เช่น if..then...else for...do หรือ loop หรือ while ทำให้มีข้อจำกัดในการเขียนชุดคำสั่งงาน การใช้ภาษา SQL ฝังในโปรแกรมอื่นจะทำให้ภาษา SQL มีความสามารถและมีประสิทธิภาพมากยิ่งขึ้น ผลลัพธ์ของคำสั่งที่เกิดจากภาษา SQL ที่ฝังในโปรแกรมจะถูกส่งผ่านไปให้กับตัวแปรหรือพารามิเตอร์ที่ใช้ โดยโปรแกรมที่ภาษา SQL ไปฝังตัวอยู่

ทั้งภาษา SQL ที่โต้ตอบได้และภาษา SQL ที่ฝังในโปรแกรมจะมีลักษณะของคำสั่งที่ใช้งานเหมือนกัน จะต่างกันแต่เพียงภาษา SQL ที่ฝังในโปรแกรมจะมีวิธีการเชื่อมโยงกับภาษาอื่น ๆ

ในการ Retrieve จะอาศัยค่าของข้อมูลที่ถูกจัดเก็บอยู่ใน Field ของแต่ละ Record และสามารถสนับสนุนการทำ Query ได้ทุกระดับ ตั้งแต่ Query ในตารางข้อมูลง่ายๆ จากเพียงตารางเดียวจนถึงแบบที่มีความซับซ้อนจากหลายตารางที่ต้องอาศัยการใช้การ Join, การทำ Nesting Query หรือวิธีอื่นๆ

2.3 กลไกที่ใช้ในการเข้าถึงข้อมูล

การดำเนินการทั้งหมดในการเข้าถึงข้อมูลจะใช้ค่าของข้อมูลที่ถูกจัดเก็บไว้ใน Field ของแต่ละ Record โดยที่ไม่สามารถสนับสนุนการอ้างอิงจาก Record หนึ่งไปยังอีก Record หนึ่งได้และผลการปฏิบัติ Query จะเสร็จสิ้นภายใต้การควบคุมของ Cursor ที่คอยอนุญาตให้เกิดการปฏิบัติการต่างๆ กับข้อมูลได้ และได้ครั้งละ 1 Record เท่านั้น ซึ่งกลไกนี้จะใช้ทั้งในการทำ Query หรือการเปลี่ยนแปลงแก้ไขข้อมูล

2.4 ลักษณะเด่นและข้อจำกัดของการจัดการฐานข้อมูลแบบสัมพันธ์

ลักษณะเด่น

1. เหมาะกับงานที่เลือกดูข้อมูลแบบมีเงื่อนไขหลายคีย์ฟิลด์ข้อมูล

2. ป้องกันข้อมูลถูกทำลายหรือแก้ไขได้ดี เนื่องจากโครงสร้างแบบสัมพันธ์นี้ผู้ใช้จะไม่ทราบว่าการเก็บข้อมูลในฐานข้อมูลอย่างแท้จริงเป็นอย่างไร จึงสามารถป้องกันข้อมูลถูกทำลายหรือถูกแก้ไขได้ดี

3. การเลือกดูข้อมูลทำได้ง่าย มีความซับซ้อนของข้อมูลระหว่างแฟ้มต่างๆ น้อยมาก อาจมีการฝึกฝนเพียงเล็กน้อยก็สามารถใช้ทำงานได้

ข้อจำกัด

1. มีการแก้ไขปรับปรุงแฟ้มข้อมูลได้ยากเพราะผู้ใช้จะไม่ทราบการเก็บข้อมูลในฐานข้อมูลอย่างแท้จริงเป็นอย่างไร

2. มีค่าใช้จ่ายของระบบสูงมากเพราะเมื่อมีการประมวลผลคือ การอ่าน เพิ่มเติม ปรับปรุง หรือยกเลิกระบบจะต้องทำการสร้างตารางขึ้นมาใหม่ ทั้งที่ในแฟ้มข้อมูลที่แท้จริงอาจจะมีการเปลี่ยนแปลงเพียงเล็กน้อย แต่ต้องมาปรับแต่งตารางใหม่ให้ผู้ใช้แฟ้มข้อมูลนั้นถูกใช้ในรูปของตารางที่ดูง่ายสำหรับผู้ใช้

บทที่ 3

ฐานข้อมูลเชิงวัตถุ (Object-Oriented Database)

ในทุกวันนี้ Application ส่วนมากมักใช้ Relational Database Management System (RDBMS) เป็น Data Store ขณะที่ใช้ Object-Oriented Programming Language ในการพัฒนา ดังนั้นจึงเป็นสาเหตุให้เกิดความไม่มีประสิทธิภาพเพราะ Object ต้องถูกทำการแปลงเป็น Row ก่อนลงฐานข้อมูล ซึ่งถ้าใช้ Object-Oriented Database สามารถแก้ปัญหาในจุดนี้ได้

ฐานข้อมูลเชิงวัตถุ เกิดขึ้นจากหลักการด้านภาษาสำหรับการเขียนโปรแกรมเชิงวัตถุ และความสามารถด้านฐานข้อมูลเข้าด้วยกัน ผลลัพธ์ที่ได้คือทำให้เกิดความเข้ากันได้ดีระหว่างโครงสร้างข้อมูลภายในฐานข้อมูล กับโครงสร้างข้อมูลที่ใช้ในการพัฒนาโปรแกรม ซึ่งอำนวยความสะดวกในการใช้ภาษาสำหรับการเขียนโปรแกรมเชิงวัตถุต่างๆ โดยหลักการ Object-Oriented Programming เช่น Encapsulation, Polymorphism และ Inheritance เป็นต้น ส่วน Database Management เช่น ACID Properties (Atomicity, Consistency, Isolation and Durability) เป็นต้น

ฐานข้อมูลเชิงวัตถุออกแบบมาเพื่อนำเสนอ Complex Object ซึ่งนำไปสู่การพัฒนา Object-Oriented (OO) Systems โดยใน OODBMS สามารถเก็บ Object ซึ่งใน Object นั้นประกอบไปด้วย 1 Atomic Type หรือ มากกว่าหนึ่ง และ Object อื่นๆ ได้โดย ในระบบฐานข้อมูลเชิงวัตถุนี้ระบบจะกำหนด OID ให้ออบเจ็กต์แต่ละตัวอัตโนมัติเมื่อออบเจ็กต์ถูกสร้าง และค่า OID ของแต่ละออบเจ็กต์จะไม่ซ้ำกันไปตลอดการใช้งาน นอกจากนี้ในตัวออบเจ็กต์แต่ละออบเจ็กต์ยังสามารถจัดเก็บ OID ของ Object ตัวอื่นเพื่อใช้ในการอ้างอิงถึง Object ตัวอื่นได้ ซึ่งจะเป็นประโยชน์ในการสร้างความสัมพันธ์ระหว่าง Object และในฐานข้อมูลเชิงวัตถุยังมีแนวความคิดเกี่ยวกับ Abstract Data Types คือ ไม่สามารถอ้างถึงข้อมูลได้โดยตรง แต่หากจำเป็นต้องการอ่านหรือเขียนข้อมูลบางส่วนจะต้องทำผ่านทาง Operation ที่กำหนดไว้เท่านั้น ชนิดข้อมูลที่มีลักษณะดังกล่าวนี้จะถูกเรียกว่า Abstract Data Types ซึ่งเป็นแนวคิดเชิงวัตถุ

3.1 แนวความคิดเชิงวัตถุ

แนวความคิดเชิงเชิงวัตถุ จะประกอบด้วยองค์ประกอบเหล่านี้

• OID

Object Identity หรือ Object Identifier (OID) เป็นสิ่งที่ใช้ในการระบุหรือใช้ในการอ้างอิงถึงออบเจ็กต์ ซึ่งในแต่ละออบเจ็กต์จะมีค่า OID ไม่ซ้ำกัน (Unique) เมื่อออบเจ็กต์ถูกสร้างขึ้นระบบจะทำการกำหนดค่า OID ให้ และไม่สามารถเปลี่ยนแปลงค่า OID ภายหลังได้ และเมื่อออบเจ็กต์นั้น

ถูกลบออกจากระบบ OID ของออบเจ็กต์นั้นก็จะถูกลบไปด้วยไม่มีการนำค่า OID ของออบเจ็กต์ที่ถูกลบไปแล้วกลับมาใช้ใหม่

• คุณสมบัติ (Property)

คือข้อมูลของออบเจ็กต์ เป็นได้ทั้งข้อมูลชนิดพื้นฐานโดยทั่วไป ประกอบด้วย Integer, Float และ String เป็นต้น ส่วนข้อมูลที่ซับซ้อนจะประกอบด้วยการนำเอาข้อมูลพื้นฐานมารวมเป็นโครงสร้าง ได้แก่ Array หรืออาจจะเป็นข้อมูลที่เป็น ออบเจ็กต์ก็ได้

• เมธทอด (Method)

ออบเจ็กต์ที่เก็บจะถูกเรียกใช้งานผ่านทาง เมธทอดเท่านั้น ตามคุณสมบัติเอนแคปซูเลชั่น เมธทอดจะถูกเรียกใช้เพื่อเรียกดูหรือเปลี่ยนแปลงค่าคุณสมบัติของออบเจ็กต์ เมธทอดจะประกอบด้วย ชื่อเมธทอด , รูปแบบในการเรียกใช้, และการทำงานของเมธทอดนั้น

• เมสเซจ (Message)

ออบเจ็กต์ที่เก็บอยู่ในระบบฐานข้อมูลเชิงวัตถุจะมีวิธีการใช้งานโดยผ่านทาง เมธทอด โดยเมธทอดจะประกอบด้วย ชื่อ, รูปแบบในการใช้ และส่วนที่เป็นตัวเนื้อของเมธทอดในการเรียกใช้เมธทอดนั้นต้องทำการส่งเมสเซจไปยังออบเจ็กต์ กล่าวคือ เมสเซจที่ถูกส่งไปยังออบเจ็กต์จะเป็นตัวสั่งให้เมธทอด ในออบเจ็กต์ทำงานตามรูปแบบการทำงานในส่วนตัวเนื้อของเมธทอด โดยเมสเซจที่ส่งจะต้องระบุออบเจ็กต์ที่จะรับเมสเซจนั้น ชื่อเมธทอด และพารามิเตอร์ที่เกี่ยวข้องทั้งหมด

• คลาส (Class)

กลุ่มของออบเจ็กต์ที่มีโครงสร้างภายในเหมือนกัน ตอบสนองต่อเมสเซจเดียวกัน โดยที่มีรายละเอียดของคุณสมบัติและรายละเอียดของเมธทอดที่เหมือนกันชนิดตัวแปรชนิดเดียวกัน ในบางกรณีอาจพบว่าเราสามารถให้ออบเจ็กต์ตัวหนึ่ง ๆ เป็นตัวแทนของคลาสมากกว่าหนึ่งคลาส มากกว่าหนึ่งคลาสได้ โดยการสืบทอดคุณสมบัติ ซึ่งจะอธิบายต่อไป แต่อย่างไรก็ตามออบเจ็กต์ตัวนั้นจะถูกสร้างขึ้นมาจากคลาสหลักเพียงหนึ่งคลาสเท่านั้น

• เอนแคปซูเลชั่น (Encapsulation)

เอนแคปซูเลชั่นเป็นการซ่อนรายละเอียด คุณสมบัติ (Property) และเมธทอด ของคลาสไว้ ข้อมูลภายในออบเจ็กต์จะถูกปิดกั้นไว้ไม่ให้มองเห็นจากภายนอกคลาส โดยเมื่อสิ่งต่าง ๆ เมื่ออยู่ภายนอกคลาสจะติดต่อกับคลาสได้ต้องติดต่อผ่านทางเมธทอดที่คลาสนั้นเตรียมไว้ให้เท่านั้น ทำให้ข้อมูลมีความปลอดภัยมากขึ้น การซ่อนรายละเอียดของคลาสมีหลายระดับ ในบ้างรายละเอียดอาจเปิดเผยให้ภายนอกสามารถมองเห็นและเรียกใช้งานได้โดยตรง แต่ในบางรายละเอียดต้องการปกปิดไม่ให้ภายนอกมองเห็น ขึ้นกับความจำเป็นในการใช้งาน การปกปิดดังกล่าวสามารถแบ่งได้ 3 ระดับ คือ

1) Private เมื่อถูกกำหนดให้เป็นแบบ Private ข้อมูลภายในของคลาสจะไม่ถูกเปิดเผยแก่ภายนอก และไม่สามารถเข้าถึงได้โดยตรงจากภายนอก แต่สามารถเข้าถึงได้จากเมธทอดที่อยู่ภายในคลาสนั้นเท่านั้น การเรียกใช้จากภายนอกจริงต้องเรียกผ่านเมธทอดเท่านั้น

2) Protect เป็นการประกาศข้อมูลแบบให้คลาสลูกเรียกใช้ได้ มีผลให้เฉพาะเมธทอดในคลาสนั้นและคลาสที่สืบทอด (Sub Class) เท่านั้นที่สามารถอ้างอิงถึงข้อมูลภายในคลาสได้

3) Public ข้อมูลภายในคลาสจะถูกเปิดเผยและสามารถถูกเข้าถึงโดยตรงจากภายนอก นั่นคือไม่มีการปดปิดข้อมูลใด ๆ ทั้งสิ้น

• การสืบทอดคุณสมบัติ (Inheritance)

ในระบบฐานข้อมูลเชิงวัตถุ มีแนวคิดการสืบทอดคุณสมบัติ โดย ออบเจ็กต์จะสืบทอดคุณสมบัติ (Property) และเมธทอด มาจากคลาสที่มีอยู่แล้ว (Super Class) โดยสามารถเพิ่มคุณสมบัติพิเศษบางอย่างให้กับคลาสเดิมเพื่อให้เกิดคลาสใหม่ที่พิเศษกว่าคลาสเดิม จากคุณสมบัตินี้ทำให้ได้ประโยชน์ทางด้านการนำกลับมาใช้ใหม่ และเพิ่มคุณสมบัติเข้าไป โดยที่ไม่ต้องเขียนคลาสขึ้นมาใหม่ทั้งหมด แบ่งออกเป็น 2 แบบคือ

Single Inheritance เป็นการสืบทอดมาจากคลาสแม่เพียงคลาสเดียว

Multiple Inheritance เป็นการสืบทอดมาจากคลาสแม่หลาย ๆ คลาส

• พอลิมอร์ฟิซึม (Polymorphism)

การที่เมธทอดสามารถใช้ชื่อเดียวกันในหลาย ๆ คลาส โดยยอมให้มีการส่งเมสเซจเดียวไปยังออบเจ็กต์หลายตัว และออบเจ็กต์แต่ละตัวมีการตอบสนองต่อเมซเสจนั้นแตกต่างกัน ทำให้เราไม่ต้องใช้เมสเซจที่แตกต่างกันตามแต่ละออบเจ็กต์ ช่วยให้ชนิดของเมสเซจที่ส่งเหลือเพียงชนิดเดียว แบ่งออกเป็น 2 ลักษณะใหญ่คือ

1. แบบ Overloading คือ การที่เมธทอดภายในคลาสเดียวกันมีชื่อเหมือนกัน แต่มีรูปแบบการใช้งานที่แตกต่างกัน

2. แบบ Overriding คือ คลาสลูกมีเมธทอดที่มีชื่อเหมือนกับเมธทอดที่สืบทอดมาจากคลาสแม่โดยได้ทำการแก้ไขการทำงานภายในเมธทอดให้มีความเจาะจงมากขึ้น เมธทอดนี้จะถูกเรียกใช้แทนเมธทอดที่สืบทอดมาจากคลาสแม่

3.2 โครงสร้างข้อมูล

จากแนวคิดการออกแบบระบบเชิงวัตถุนำไปสู่การพัฒนาฐานข้อมูลเชิงวัตถุ โดยการใช้งานข้อมูลในฐานข้อมูลเชิงวัตถุจะอยู่ในรูปแบบของคลาส ซึ่งเป็นแนวความคิดตามทฤษฎีของ Object-

Orientedโดยมีลักษณะการเก็บข้อมูลและการเรียกใช้ข้อมูลแตกต่างต่างจากระบบฐานข้อมูลเชิงสัมพันธ์ คือจะมองข้อมูลที่ถูกจัดเก็บในรูปแบบของออบเจ็กต์ ซึ่งออบเจ็กต์จะประกอบด้วย

คุณสมบัติ (Property) และเมธทอด ซึ่งจะมีการกำหนดค่าที่ใช้ในการอ้างอิงถึงออบเจ็กต์ (Object Identifier: OID) เอาไว้ นอกจากนี้ยังสนับสนุนการทำงานแบบ Encapsulation และ Inheritance ด้วย

3.3 ภาษาที่ใช้ในการจัดการฐานข้อมูล

1) ระบบฐานข้อมูลเชิงวัตถุจะใช้ภาษาสำหรับการเขียนโปรแกรมเชิงวัตถุ ทั้งในการจัดการฐานข้อมูล และการพัฒนาโปรแกรม เช่น C++, Java เนื่องจากแอพพลิเคชั่น และออบเจ็กต์ที่เก็บข้อมูลในระบบฐานข้อมูลชนิดนี้มีความสัมพันธ์กันโดยตรง ดังนั้นการทำ กำหนดข้อมูล, การจัดการกับข้อมูล และการ Query จึงสามารถทำได้ด้วยภาษาสำหรับการโปรแกรมเชิงวัตถุ คือ กระทำผ่านออบเจ็กต์ซึ่งเมื่อมีการทำการเปลี่ยนแปลงค่าใดๆ ในออบเจ็กต์แล้วจะมีผลต่อข้อมูลในฐานข้อมูล มีข้อดีคือ

- ทำให้ไม่ต้องใช้ คำสั่ง SQL ซึ่งมีความยุ่งยากซับซ้อนกว่า เนื่องจากสามารถทำการเปลี่ยนแปลง, แก้ไข, ค้นหาข้อมูลได้โดยผ่านทางออบเจ็กต์
- ซ่อนความยุ่งยากในการเขียนโปรแกรมแอพพลิเคชั่น เนื่องจากการใช้คุณสมบัติเอนแคปซูเลชั่นของออบเจ็กต์ ผู้ใช้แค่ใช้เมธทอดที่ผู้ออกแบบระบบฐานข้อมูลได้เตรียมไว้ให้โดยไม่ต้องทราบว่าข้างในออบเจ็กต์เป็นยังไง

2) ตามมาตรฐาน ODMG-93 ได้มีการกำหนดภาษา ขึ้นมาอีกภาษาหนึ่งคือ OQL (Object Query Language) ซึ่งใช้ในการเข้าถึงข้อมูลแบบวัตถุซึ่งจะมีความแตกต่างกันออกไปจาก SQL ตามมาตรฐาน SQL 2 เพราะในมาตรฐาน SQL 2 ไม่มีความสามารถในการเข้าถึงข้อมูลแบบวัตถุ อย่างไรก็ดีในมาตรฐานใหม่ของ SQL 3 จะได้มีบางส่วนที่จะได้เพิ่มความสามารถเหล่านี้ลงไป

3.4 กลไกที่ใช้ในการเข้าถึงข้อมูล

การสร้างและแก้ไขข้อมูลในระบบฐานข้อมูลเชิงวัตถุจะใช้การเข้าถึงโดยตรงจากภาษาสำหรับการเขียนโปรแกรมเชิงวัดถุ ในระบบฐานข้อมูลเชิงวัตถุนี้ระบบจะกำหนด OID ให้ออบเจ็กต์แต่ละตัวอัตโนมัติเมื่อออบเจ็กต์ถูกสร้าง และค่า OID ของแต่ละออบเจ็กต์จะไม่ซ้ำกันไปตลอดการใช้งาน นอกจากนี้ในตัวออบเจ็กต์แต่ละออบเจ็กต์ยังสามารถจัดเก็บ OID ของออบเจ็กต์ตัวอื่น เพื่อใช้ในการอ้างอิงถึงออบเจ็กต์ตัวอื่นได้ ซึ่งจะเป็นประโยชน์ในการสร้างความสัมพันธ์ระหว่างออบเจ็กต์

3.5 เปรียบเทียบ OODBMS กับ RDBMS

มีแนวคิดของ Relational Database Model ที่มีความคล้ายคลึงกับแนวคิดของ Object Database Model อย่างเช่น

- Relation หรือ Table ใน Relational Database นั้นเหมือนกับ Class ใน Object Database
- Tuple ใน Relational Database นั้นเหมือนกับ Instance ของ Class แต่แตกต่างตรงที่ Tuple มีแต่ Attribute แต่ไม่มี Method ส่วน Instance ของ Class นั้นมี Method
- Column ใน Tuple คล้ายคลึงกับ Class Attribute แต่ Column สามารถเก็บได้เฉพาะ Primitive Data Types ส่วน Class Attribute เก็บได้ทุกๆ Type ที่ต้องการ
- ส่วน Primary Key ใน Relational Database คล้ายคลึงกับ OID ของ Object Database

ตาราง 3.1 เปรียบเทียบ OODBMS กับ RDBMS

RDBMS	OORDBMS
Relational Schema	Class
Tuple	Instance of Class
Column	Class Attribute
OID	Primary Key

บทที่ 4
ฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database)

การพัฒนาโปรแกรมในยุคปัจจุบัน มีแนวโน้มไปทางการพัฒนาโปรแกรมเชิงวัตถุมากขึ้น ทำให้เกิดความต้องการในการใช้ข้อมูลที่มีความซับซ้อนมากขึ้นตามไปด้วย ซึ่งระบบฐานข้อมูลเชิงสัมพันธ์แบบเดิมไม่สามารถรองรับได้เต็มที่ จึงได้เกิดแนวคิดที่จะมีการเพิ่มความสามารถในการจัดการข้อมูลในรูปแบบ Object ให้แก่ ระบบฐานข้อมูลเชิงสัมพันธ์ ระบบฐานข้อมูลใหม่นี้เรียกว่า ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ซึ่งได้รับความสนใจเป็นอย่างมากเป็นอย่างมาก จนทำให้เกิดแนวคิด และวิธีการที่หลากหลายในการพัฒนาระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ขึ้น โดยเฉพาะเมื่อ Michael Stonebraker ได้ร่วมกับทีมวิจัยของคนพัฒนาระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ตัวอย่างขึ้น ชื่อ Postgres

ต่อมาในภายหลัง ANSI จึงได้เริ่มดำ เนินการเพื่อเพิ่มมาตรฐานด้าน Object-Oriented ให้แก่มาตรฐาน SQL ที่มีอยู่ โดยการดำเนินการดังกล่าวอยู่ในความรับผิดชอบของทีมพัฒนาชุด X3H2 ของ ANSI ซึ่งได้ใช้วิธีการเพิ่ม Object Extension โดยจะเป็นการเพิ่มชนิดข้อมูลที่เป็น Object เข้าไปเป็นส่วนหนึ่งในระบบฐานข้อมูลเชิงสัมพันธ์อย่างแท้จริงให้แก่ระบบฐานข้อมูลเชิงสัมพันธ์ โดยตรงเป็นแนวทางในการพัฒนามาตรฐานจาก SQL2 ไปสู่ SQL3

4.1 โครงสร้างข้อมูล

ดังที่กล่าวมาแล้วข้างต้นว่าการพัฒนามาตรฐาน SQL3 เพื่อรองรับการใช้งานข้อมูลประเภท Object ของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ใช้วิธีการเพิ่ม Object Extension ให้แก่ระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งการดำเนินการในส่วนนี้ถือเป็นหัวใจของมาตรฐาน SQL 3 โดยการเพิ่ม Object Extension ดังกล่าวคือ การเพิ่มชนิดข้อมูลแบบใหม่ให้แก่ระบบฐานข้อมูลเชิงสัมพันธ์ โดยที่โครงสร้างการจัดเก็บข้อมูลยังถูกแสดงอยู่ในรูปแบบของตารางเช่นเดิม ชนิดข้อมูลแบบใหม่นี้ถูกเรียกว่า Abstract Data Type (ADT) ซึ่งจะมีลักษณะคล้าย Class ในหลักการโปรแกรมเชิงวัตถุ กล่าวคือ ADT จะถูกใช้เพื่อรวมกลุ่มของข้อมูลที่มีความสัมพันธ์กันทั้งในส่วนของคุณสมบัติของข้อมูล และสถานการณ์ที่จะเกิดขึ้นกับข้อมูลเหล่านั้น โดยที่ชนิดข้อมูลแบบ ADT นี้ จะสนับสนุนคุณสมบัติ Encapsulate หรือการมีความสัมพันธ์กับ ADT กลุ่มอื่นในรูปแบบของ Subtype-Supertype ทั้งยังมีคุณสมบัติ Inherit ในการถ่ายทอดคุณสมบัติของ Attribute มาจาก ADT ที่เป็น Supertype ของตนด้วย นอกจากนี้แล้วยังสามารถกำหนด Operation และ Function ที่มีความสัมพันธ์กับข้อมูล เพื่อใช้ในการทำ Index, Store และ Retrieve Record ข้อมูลตามคุณลักษณะพิเศษของข้อมูลแต่ละประเภทได้ เช่น ข้อมูลที่เป็น Multimedia ต่างๆ ที่จะต้องมีวิธีการ Retrieve ที่

แตกต่างออกไปจากปกติเป็นต้น ซึ่ง Operation และ Function ของ ADT เหล่านี้สามารถนำมาใช้ในการทำ Query ข้อมูลได้

สำหรับ Type ที่ ADT สนับสนุนมีทั้ง Row Type (หรือชนิดของ Row ที่อยู่ใน Table), Distinct Type (หรือการสร้าง User-Defined Type จาก Type ต่างที่ระบบมีอยู่) และ การสร้าง Type Template ขณะเดียวกันยังมีการเตรียมเครื่องมือภายในระบบเอาไว้สนับสนุนการสร้าง Type ในรูปแบบ Collection Type จำพวก LIST หรือ SET อีกด้วย ซึ่งจะเห็นได้ว่า ADT จะสามารถรองรับการใช้งานข้อมูลที่มีโครงสร้างซับซ้อนมากให้ดีกว่าเดิมได้ เช่น การสร้าง Collection ของ Row ขึ้นมาแทนการทำ Nested Table ที่มีความยุ่งยากเป็นต้น

4.2 ภาษาที่ใช้ในการจัดการข้อมูล

เพื่อสนับสนุนการใช้งานข้อมูล ADT ในระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ จึงได้มีการเพิ่มเติมความสามารถในการเข้าถึงข้อมูลดังกล่าวให้แก่ภาษา SQL ซึ่งอาจเรียกส่วนที่เพิ่มเติมใหม่นี้ว่า ObjectSQL ประเด็นสำคัญของส่วนที่เพิ่มเติมใหม่นี้คือการสนับสนุนการใช้งานข้อมูลในรูปแบบ Object ที่รวมการจัดการข้อมูลแบบ Nested Object, set-valued Attribute, ความสามารถในการใช้ Operation หรือ Function ของ Object ในการทำ Query กับข้อมูล ADT ได้ โดยยังยึดเอา SQL เป็นภาษาหลักในการทำ Data Definition, Data Manipulation และการทำ Query สำหรับผลของการทำ Query นั้น ข้อมูลจะยังถูกแสดงในรูปแบบของตารางอยู่ อย่างไรก็ดีในอนาคตได้มีการวางแผนที่จะพัฒนามาตรฐานใหม่เป็น SQL 4 ที่สามารถแสดงผลการทำ Query ให้อยู่ในรูปของ Object ด้วยการผสมผสานเทคนิคของ OQL เข้ามา หรือการใช้เทคนิคที่ใกล้เคียง

นอกจากนี้มาตรฐาน SQL 3 ยังมีส่วนของการเพิ่ม Procedural Extension (SQL/PSM) ซึ่งการดำเนินการดังกล่าวได้ครอบคลุมไปถึงการพัฒนารูปแบบคำสั่ง SQL ที่ใช้ในการทำ Stored Procedure และ User-Defined Function เพื่อใช้งานร่วมกับฐานข้อมูลด้วย กล่าวคือ Procedure และ Function เหล่านี้จะสามารถสร้างขึ้นได้จากภาษา SQL เอง ซึ่งมีการพัฒนารูปแบบคำสั่งใหม่ที่ใกล้เคียงกับภาษาสำหรับการโปรแกรมทั่วไปในยุคปัจจุบัน ได้แก่ความสามารถในการใช้คำสั่งเงื่อนไขประเภท IF/THEN/ELSE, การใช้ LOOP แบบ WHILE และ DO/UNTIL หรือการใช้ CASE เป็นต้น ทั้งยังให้ Programmer สามารถสร้าง Function Library ด้วยภาษาอื่นเช่น C, Fortran, COBOL, ADA หรือ PASCAL เป็นต้น เพื่อใช้งานร่วมกับภาษา SQL ที่มีอยู่ได้ ซึ่งจะเพิ่มขีดความสามารถในการทำ Query ได้ดียิ่งขึ้น

อีกส่วนหนึ่งที่มีความสำคัญในมาตรฐาน SQL 3 ก็คือ Relational Extension ซึ่งจะทำให้โปรแกรมเมอร์สามารถสร้าง Recursive Query, สนับสนุน Query Expression ที่สามารถใช้งานแบบ Share กันได้ รวมไปถึงการพัฒนา Cursor และ View ให้ดีขึ้นจากมาตรฐานเดิม ซึ่งจะสามารถ

สนับสนุนระบบการทำงานแบบ Active Database ในส่วนของการทำ Trigger และ Integrity Constrain ได้ ซึ่งเราพอจะสรุปคุณสมบัติที่เพิ่มขึ้นมาใน SQL3 ได้ดังนี้

- Base Type Extension
- Complex Objects
- Inheritance
- Production Rule System

4.3 กลไกที่ใช้ในการเข้าถึงข้อมูล

ระบบฐานข้อมูลแบบสัมพันธ์ทั่วๆ ไปนั้นจะมีการสนับสนุนการเข้าถึงข้อมูลแบบ B-tree เพื่อใช้ในการทำ Optimize Keyed Access Method ของเรคคอร์ด ซึ่งในระบบ ORDBMS ส่วนของการเข้าถึงข้อมูลนี้มีความจำเป็นที่ต้องเพิ่มความสามารถในการสนับสนุนชนิดของข้อมูลใหม่ๆ ที่มีในระบบฐานข้อมูลแบบนี้ ซึ่งแบบ B-tree ที่มีอยู่เดิมนี้มีการเก็บอินเด็กซ์เรคอร์ดไว้ในลักษณะที่ถูกกำหนดโดยตัวดำเนินการ(Operator) < "น้อยกว่า" ที่กระทำอยู่กับข้อมูลที่ใช้อินเด็กซ์แบบนี้ โดยในการกำหนดการทำงานของตัวดำเนินการนี้กับชนิดข้อมูลแต่ละแบบจะถูกเก็บไว้ในส่วนข้อมูลเกี่ยวกับข้อมูล (Meta Data) ในรายการของระบบ (System Catalogs)

ซึ่งลักษณะของข้อมูลชนิดที่มีขึ้นในระบบแบบ ORDBMS อาจเป็นแบบที่มีความซับซ้อนในการที่จะค้นหาหรือจะเข้าถึงข้อมูลอาจไม่สามารถที่จะใช้วิธีการแบบ B-tree ได้ ยกตัวอย่างเช่น ชนิดข้อมูลสำหรับเก็บเก็บจุดต่างๆ บนแผนที่ซึ่งจะเห็นว่าต้องใช้การค้นหาในลักษณะ 2 มิติ ไม่สามารถที่จะใช้วิธี Optimize ที่ใช้วิธีการแบบ 1 มิติเหมือนแบบ B-tree ได้ ดังนั้นจึงมีการคิดวิธีการต่างๆ ที่เหมาะสมกว่า เช่นมีการใช้วิธีการแบบ R-tree, Quad tree หรือแบบ grid file ซึ่งจากสาเหตุนี้จึงทำให้ระบบฐานข้อมูลแบบ ORDBMS จะต้องมี Engine ที่สามารถสนับสนุนการเพิ่มเติมวิธีการเข้าถึงข้อมูลต่างๆ ขึ้นมาได้ ซึ่งรวมไปถึงความสามารถที่จะสนับสนุนวิธีการเข้าถึงข้อมูลที่ถูกเขียนโดยเครื่องมือของผลิตภัณฑ์ต่างๆ ที่มีอยู่ (Third Parties) ได้ ซึ่งเราเรียกความสามารถนี้ว่าความสามารถที่สนับสนุนการเข้าถึงข้อมูลโดยผู้ใช้เป็นคนกำหนด (User-Define Access Method)

โดยการสนับสนุนการกำหนดวิธีการเข้าถึงนี้ไม่จำเป็นต้องมีการเขียนโปรแกรมลงไว้ใน DBMS กล่าวคืออาจสามารถให้มีการนิยาม (Abstract) วิธีการเข้าถึงที่ต้องการโดยที่ DBMS สามารถเข้าใจนิยามที่เราต้องการได้

เนื่องจากภาษาหลักที่ใช้ในการจัดการฐานข้อมูลยังคงเป็น SQL และระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ได้ถูกพัฒนาขึ้นมาจากระบบฐานข้อมูลเชิงสัมพันธ์ ดังนั้นกลไกในการเข้าถึงข้อมูลส่วนใหญ่จะยังคงเป็นเช่นเดียวกับระบบฐานข้อมูลเชิงสัมพันธ์อยู่ ส่วนการเพิ่มความสามารถในการเข้าถึงข้อมูลประเภท ADT หรือการเข้าถึงข้อมูลในรูปแบบ Object นั้นยังมีปัญหาเมื่อต้องการเข้าถึง

โดยตรงจากภาษาการโปรแกรมเชิงวัตถุ ทำให้ยังจำเป็นที่จะต้องมีการแปลงรูปให้กลับมาอยู่ในลักษณะตารางเสียก่อน

บทที่ 5

PostgreSQL

5.1 PostgreSQL คืออะไร

PostgreSQL คือ ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database Management System หรือ ORDBMS) ซึ่งปรับปรุงมาจากต้นแบบระบบฐานข้อมูล POSTGRES 4.2 ของมหาวิทยาลัยแคลิฟอเนีย ที่ เบอร์กเลย์ ภายใต้ความควบคุมของ Professor Michael Stonebreaker โดยได้รับเงินวิจัยสนับสนุนจาก the Defense Advanced Research Project Agency (DARPA), the Army Research Office (ARO), the National Science Foundation (NSF) และ ESL, Inc

PostgreSQL เป็นโปรแกรมรหัสเปิด (Open-Source) สามารถนำไปใช้งานได้โดยไม่มีค่าใช้จ่าย และสนับสนุนมาตรฐาน SQL: 2003 และมีความสามารถต่างๆ เพิ่มขึ้นมาอีกมาก

POSTGRES ได้นำเอาแนวคิดของ Object-Relational มาใช้ ซึ่งปัจจุบันก็เป็นต้นแบบของระบบฐานข้อมูลที่นำมาทำเพื่อการค้ามากมาย และเป็นธรรมดาที่ Relational Database Management Systems (RDBMS) จะมีการสนับสนุนรูปแบบของข้อมูลที่ประกอบไปด้วยกลุ่มของข้อมูลที่มีความสัมพันธ์กัน แต่ PostgreSQL ได้มีการเพิ่มเติมสิ่งที่น่าสนใจเพิ่มขึ้นทำใช้ผู้ใช้งานสามารถจัดการการขยายระบบของตนเองได้ง่ายขึ้นดังนี้

- Inheritance
- Data Types
- Functions

นอกจากนี้ PostgreSQL ยังได้มีการเพิ่มเติมความสามารถให้ระบบมีประสิทธิภาพและมีความยืดหยุ่นเพิ่มขึ้นหลายอย่าง เช่น

- Constraints
- Triggers
- Rules
- Transactional Integrity

ซึ่งความสามารถเหล่านั้นนำไปให้ PostgreSQL ถือว่าเป็นฐานข้อมูลประเภท Object-Relational

5.2 ความเป็นมา PostgreSQL

5.2.1 กำเนิด PostgreSQL

PostgreSQL มีต้นต่อมาจากโครงการ University Ingres ตั้งแต่ปี ค.ศ. 1977 ภายใต้ความควบคุมการวิจัยของ Professor Michael Stonebreaker ซึ่งต้นแบบของ Ingres ได้นำไปปรับปรุงเป็นเชิงพาณิชย์โดย Relational Technologies/Ingres Corporation (ปัจจุบันเป็นผลิตภัณฑ์ของ Computer Associates ภายใต้ชื่อ CA-Ingres II)

ต่อมาในปี ค.ศ. 1986 Professor Michael Stonebreaker ได้เล็งเห็นว่า ระบบฐานข้อมูลเชิงสัมพันธ์ในขณะนั้น ไม่เพียงพอในการรองรับระบบงานด้านฐานข้อมูลที่ซับซ้อนในอนาคตได้ ซึ่งต้องการความสามารถพิเศษเพิ่มเติมแบ่งออกเป็น 3 หัวข้อใหญ่ๆ คือ

1. **ระบบจัดการฐานข้อมูล (Database Management System)** ต้องการองค์ความรู้และสถาปัตยกรรมโครงสร้างใหม่ในการจัดการข้อมูลให้มีประสิทธิภาพยิ่งขึ้น
2. **ระบบจัดการฐานความรู้ (Knowledge-Based Management System)** เป็นโครงสร้างใหม่เพื่อสร้างฐานความรู้ ซึ่งเห็นได้ทั่วไปในการจัดการกฎเกณฑ์และข้อกำหนดทางธุรกิจ
3. **ระบบจัดการวัตถุ (Object Management System)** เป็นโมเดลใหม่ที่ต้องการขยายต่อเพื่อช่วยเสริมให้ระบบฐานข้อมูลเชิงสัมพันธ์สามารถรองรับระบบงานที่ต้องการประเภทข้อมูลและโมเดลเชิงวัตถุระบบงานที่ต้องการใช้โมเดลเชิงวัตถุและระบบจัดการฐานข้อมูลได้แก่ งาน ประเภท CAD-CAM หรือ multimedia เป็นต้น องค์ความรู้เพิ่มเติมเพื่อนำมาใช้สนับสนุน ความสามารถดังกล่าวได้แก่ Inheritance, user-defined data types และ functions เป็นต้น

แนวคิดต่างๆ เหล่านี้ถูกนำมาวิจัยและตีพิมพ์ในวารสารเชิงวิชาการต่างๆ เป็นจำนวนมากในปี ค.ศ. 1986 ซึ่งในระบบจัดการฐานข้อมูลเชิงพาณิชย์ในปัจจุบันก็ได้รับแนวคิดเหล่านี้มาใช้เช่นกัน

5.2.2 ต้นแบบ PostgreSQL

ต้นแบบระบบฐานข้อมูลตัวแรกในโครงการนี้ใช้เริ่มแรกว่า POSTGRES ต้นแบบตัวแรกถูกเขียนด้วยภาษา LISP ซึ่งทำงานได้ช้ามาก หลังจากนั้นจึงเขียนใหม่ด้วยภาษา C ต้นแบบเริ่มใช้งานได้เมื่อปี ค.ศ. 1987 และได้ถูกเปิดตัวครั้งแรกในงานประชุมวิจัย ACM-SIGMOD ในปีเดียวกัน นับจากนั้นมาแนวความคิดใหม่ต่างๆ ได้ถูกเพิ่มเติม พร้อมทั้งสิ่งที่ล้าสมัยได้ถูกรื้อทิ้งและได้รับการออกแบบพัฒนาใหม่มาตลอดเวลา

ในระหว่างนั้น POSTGRES ถูกนำไปใช้เป็นระบบจัดการฐานข้อมูลในระบบงานที่ใช้งานจริงไม่ว่าจะเป็นระบบวิเคราะห์การเงิน ระบบตรวจวัดสมรรถภาพเครื่องยนต์อากาศยาน ระบบติดตามทางดาราศาสตร์ ระบบฐานข้อมูลการแพทย์ หรือระบบงานเชิงภูมิศาสตร์ นอกจากถูกนำไปใช้ในมหาวิทยาลัยและในงานการศึกษาแล้ว ในปี ค.ศ. 1992 ยังถูกนำไปใช้เป็นระบบจัดการฐานข้อมูลหลักของโครงการ Sequoia 2000 ของ NASA ซึ่งใช้เก็บข้อมูลเกี่ยวกับการเปลี่ยนแปลงของโลกไม่ว่าจะเป็นภูมิอากาศ ระดับน้ำ รังสี และอื่นๆ โดยมีจำนวนข้อมูลประมาณ 2 terabytes/วัน การที่ถูกนำไปใช้อย่างมากมายนี้เอง ทำให้จำนวนผู้ใช้เพิ่มขึ้นอย่างรวดเร็ว

โดยจุดประสงค์ของโครงการนั้น เพียงเพื่อใช้เป็นต้นแบบในการทดสอบหลักการและทฤษฎีที่คิดค้น แต่ได้รับความนิยมอย่างมากทำให้ต้องให้การสนับสนุนแก่ผู้ใช้งานจนเกิดเป็นภาระแก่ผู้พัฒนา ทำให้จุดประสงค์ของโครงการถูกเบี่ยงเบนไป อีกทั้งการเพิ่มเติมความสามารถต่างๆ ลงไปในซอฟต์แวร์ ทำให้ POSTGRES มีขนาดใหญ่ขึ้นมากจนยากที่จะควบคุม ด้วยเหตุผลต่างๆ ดังกล่าว ทำให้โครงการ POSTGRES ได้สิ้นสุดลงอย่างเป็นทางการเมื่อปี ค.ศ. 1993 ที่เวอร์ชัน 4.2 อย่างน่าเสียดาย

5.2.3 Postgres

ในปี ค.ศ. 1994 นักศึกษา 2 คนของโครงการ POSTGRES คือ Andrew Yu และ Jolly Chen ได้นำ POSTGRES มารื้อใหม่หมด ซึ่งทั้งสองได้แก้ไขข้อบกพร่องและเพิ่มเติมข้อดีต่างๆ ให้แก่ซอฟต์แวร์ดังนี้

- ตัดทอน Source Code ส่วนที่ซ้ำซ้อนออกไป โดยใช้ ANSI C ทั้งหมดเพื่อประโยชน์ในการพอร์ตข้ามระบบ
- เปลี่ยนภาษาในการสืบค้นข้อมูลมาตรฐาน SQL แทนที่ภาษาในการการสืบค้นเดิมที่ใช้ Postquel
- ปรับปรุงสมรรถภาพให้ทำงานได้เร็วขึ้น 30-50%
- เพิ่มเติมเครื่องมือต่างๆ เพื่อสนับสนุนในการใช้งาน เช่น Tcl/Tk Interface

รวมทั้งปรับแต่งระบบต่างๆ และเปิดให้ Download ผ่านทาง Web Site ในรูปแบบของ Open-Source ซอฟต์แวร์ภายใต้ชื่อรหัสโครงการใหม่ Postgres95

5.2.4 PostgreSQL

ในปี ค.ศ. 1996 ชื่อ Postgres95 ถูกเปลี่ยนใหม่เป็น PostgreSQL โดยเริ่มต้นที่เวอร์ชัน 6.0 ด้วยเหตุผลหลังจากเพิ่มเติมความสามารถในภาษาสืบค้นข้อมูล SQL เพื่อให้เทียบเท่ากับมาตรฐาน SQL-92 ลงในระบบฐานข้อมูล Postgres95

ในระยะเริ่มต้นนั้นโครงการต้องการอาสาสมัคร (ที่เป็นมืออาชีพ) โดยไม่เน้นที่จำนวนแต่ต้องการเวลาทุ่มเทมากกว่าจำนวนมาก เริ่มแรกนอกจาก Jolly Chen ยังมีผู้พัฒนาที่ทุ่มเทให้กับโครงการอีก 4 คน คือ Marc Fournier ชาว Canada, Vadim Mikheev ชาว Russia, Thomas Lockhart และ Bruce Momjian ชาว American ในเริ่มต้นเป็นการแก้ไขข้อผิดพลาดต่างๆ ที่มีอยู่ตามที่ได้รับแจ้งมา มีการจำแนกประเภทของข้อผิดพลาดเพื่อจัดลำดับในการแก้ไข บางอย่างสามารถแก้ไขได้ง่าย บางอย่างจำเป็นต้องใช้ความรู้ในการวิจัยเพิ่มเติม อย่างไรก็ตาม ในการปรับปรุงนั้นเน้นอยู่ที่ความน่าเชื่อถือของระบบ เนื่องจากงานฐานข้อมูลเป็นงานที่ระเอียดอ่อน ระบบงานที่ทำงานภายใต้ระบบฐานข้อมูลไม่เหมือนงานประเภทอื่น เช่น โปรแกรมจัดการเอกสารหรือเกม ที่ระบบหยุดทำงานแล้วเริ่มใหม่ได้โดยไม่สนใจงานที่ทำมา

PostgreSQL มีการออกรีลีสใหม่เสมอทุกๆ 3-5 เดือน โดยใช้เวลาประมาณ 3 เดือนในการพัฒนา อีกประมาณ 1 เดือนในการทดสอบ และหลังจากที่ประกาศออกไปอาจต้องใช้เวลาอีกหลายอาทิตย์ในการเก็บตกข้อผิดพลาด ในเวลาต่อมาได้มีผู้สนใจเข้ามาร่วมต่อเติมความสามารถให้กับ PostgreSQL เพิ่มมากขึ้น เพื่อช่วยให้นักพัฒนาทำงานร่วมกันอย่างมีประสิทธิภาพ เป้าหมายหลักสิ่งหนึ่งคือ การให้ความกระจ่างในรายละเอียดเทคโนโลยีภายในของ PostgreSQL จึงได้มีการจัดทำเอกสารทางเทคนิคต่างๆ ขึ้นเพื่อให้ผู้ที่สนใจได้ศึกษาทำให้การแก้ไขข้อผิดพลาดและการเพิ่มเติมความสารถใหม่ๆ ให้กับระบบทำได้อย่างมีประสิทธิภาพ

อย่างไรก็ตาม การที่มีผู้พัฒนามากมายร่วมกันทำงาน ก็ก่อให้เกิดปัญหาของความเป็นรูปแบบเดียวกันในการพัฒนา คณะทำงานได้พัฒนาเครื่องมือในการจัดโครงสร้างโปรแกรม (Source Tree) ให้อยู่ในรูปแบบมาตรฐานที่กำหนด พัฒนาเครื่องมือในการค้นหาโมดูลที่ไม่ได้ถูกเรียกใช้งาน เครื่องมือเหล่านี้ถูกนำมาใช้เพื่อจัดระเบียบและทำความสะอาดโปรแกรมก่อนออกเป็นรีลีสใหม่ทุกครั้ง

ในปัจจุบัน นั้นผู้พัฒนาจำนวนมากร่วมกันพัฒนาเพิ่มเติมความสามารถให้กับ PostgreSQL ส่วนของฐานผู้ใช้งานก็ขยายเพิ่มขึ้น โดยเฉพาะอย่างยิ่งเมื่อ Red Hat Linux ได้นำ PostgreSQL มาบรรจุเป็นส่วนหนึ่งในแพ็กเกจของตน อีกทั้งได้มีการตั้งบริษัทเพื่อให้บริการสนับสนุนการใช้งานและให้คำปรึกษาทางเทคนิคอีกด้วย

อาจกล่าวได้ว่า PostgreSQL ได้มีการพัฒนามาอย่างต่อเนื่องตลอดเวลาความสามารถหลักที่ได้เพิ่มเติมลงในระบบฐานข้อมูลนับจาก Postgres95 คือ

- เปลี่ยนระบบล็อกข้อมูลแบบ Table-Level Locking ด้วยระบบ Multi-Version Concurrency Control ที่ให้ผู้ใช้งานสามารถ อ่านข้อมูลที่ถูกต้องได้ในขณะที่ผู้ใช้งานคนอื่นสามารถเขียนข้อมูลอยู่ อีกทั้งทำให้สามารถทำการสำรองข้อมูลแม้ฐานข้อมูลกำลังถูกใช้งานอยู่

• เพิ่มความสารถในระบบฐานข้อมูลไม่ว่าจะเป็น Subselect สำหรับ SQL, ค่าตั้งต้น Defaults, Integrity Constraints และ Triggers

• ความสามารถในภาษาสืบค้นข้อมูล SQL เพื่อให้มีความสามารถใกล้เคียงกับ SQL-92

• Built-in Data Type ไม่ว่าจะเป็น Date/Time หรือ Geometric Data Type สำหรับระบบฐานข้อมูลภูมิศาสตร์

• ปรับแต่งความเร็วในการประมวลผลให้เร็วขึ้นประมาณ 20-40% และเวลาสำหรับ Server Start Up เร็วขึ้น 80% ตั้งแต่เวอร์ชัน 6.0

จากเวอร์ชัน 6.0 เป็นต้นมาจนถึงปัจจุบันมาถึงที่เวอร์ชัน 8.2.4 (กันยายน 2550) PostgreSQL ได้รับการปรับปรุงเพิ่มเติมมาตลอด

5.3 สถาปัตยกรรมการทำงานของ PostgreSQL

สถาปัตยกรรมมาตรฐานสำหรับระบบที่ทำงานบนระบบเครือข่าย คือ สถาปัตยกรรมที่เรียกว่า Client-Server ซึ่งในสถาปัตยกรรมนี้ประกอบด้วย

• เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Server ซึ่งจะมี Process ทำงานอยู่ ซึ่งจากนี้ไปจะขอเรียกว่า Server Process

• เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Client ซึ่งจะมี Process ทำงานอยู่ ซึ่งจากนี้ไปจะขอเรียกว่า Client Process

Server Process จะทำหน้าที่รอรับการติดต่อจาก Client Process และทำงานให้บริการแก่ Client Process โดยทั่วไป Server Process ยังแบ่งตามลักษณะช่วงเวลาที่ Client Process ติดต่อกับ Server Process ซึ่งสามารถออกเป็น 2 ประเภท คือ

• Iterative Server ใช้สำหรับที่สามารถประมาณช่วงเวลาได้ ตัวอย่างเช่น httpd process ใน Web Server ซึ่งให้บริการช่วงสั้นๆ ในการส่งข้อมูล Web Page ให้แก่ Web Browser

• Concurrent Server ใช้กับระบบที่ไม่สามารถประมาณช่วงเวลานั้นได้ Client อาจติดต่อเป็นช่วงเวลาสั้นๆ หรือนานมากในการขอบริการ ระบบจัดการฐานข้อมูลทำงานในลักษณะนี้

PostgreSQL ทำงานภายใต้โมเดลของสถาปัตยกรรมแบบ Server Process โดย Server Process ทำงานในลักษณะ Concurrent Server ซึ่งมีขั้นตอนดังต่อไปนี้

Server Process ที่ชื่อว่า Postmaster ถูกเรียกให้เริ่มต้นทำงานบนคอมพิวเตอร์ที่ทำหน้าที่เป็น Server ทำการเปิดช่องการติดต่อและแจ้งกับระบบปฏิบัติการที่จะรับคำขอบริการจาก Client ที่ Port ที่กำหนดไว้ โดยทั่วไป Postmaster กำหนด Port เริ่มต้นไว้ที่หมายเลข 5432

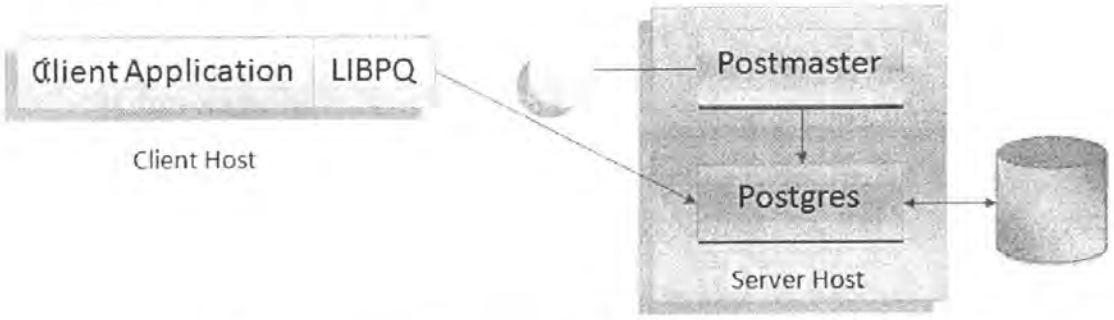
Postmaster จะเข้าสู่สภาวะ Sleep เพื่อรอการเรียกจาก Client Process โดยที่ Client Process จากเครื่องคอมพิวเตอร์ลูกข่าย (หรือแม้จาก Server เองก็ได้) ติดต่อขอรับบริการจากไลบรารี LIBPQ มาที่ Port ที่กำหนดไว้ ดังแสดงในรูป

รูปที่ 5.1 แสดง Client Process ขอบริการ Postmaster

เมื่อ Postmaster ได้รับการติดต่อจาก Client Process แล้วจะทำการสร้าง Process ลูก คือ postgres เพื่อให้บริการแทนตัวเองดังแสดงในรูป

รูปที่ 5.2 แสดง Postmaster สร้าง Postgres Process

หลังจากนั้น Postmaster จะรอรับการขอบริการจาก Client Process อื่นต่อไป ในขณะที่ postgres process เมื่อให้บริการจนจบ ก็จะปิดตัวเองดังแสดงในรูป



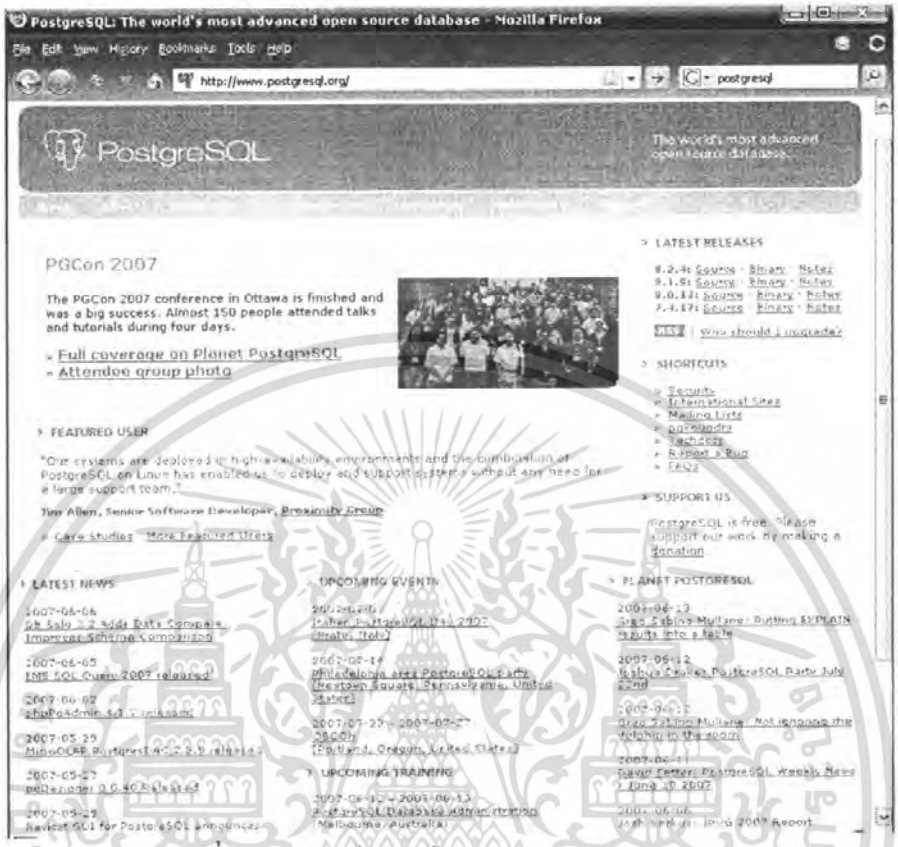
รูปที่ 5.3 แสดง Postgres Process ให้บริการแก่ Client Process

PostgreSQL ทำงานในลักษณะ per-user process โดยทั้ง postmaster และ postgres process ทำงานด้วย user-id ของ postgres super user (postgres super user ไม่จำเป็นต้องเป็น user ที่ชื่อ postgres แต่เป็น user ที่ติดตั้ง PostgreSQL)

รูปที่ 5.4 แสดงการขอบริการ Server ผ่านอินเตอร์เฟส JDBC,ODBC

LIBPQ เป็นไลบรารีมาตรฐานสำหรับ Client Program (ที่เขียนด้วย C Program) ในการติดต่อกับ PostgreSQL Server อย่างไรก็ตาม เราสามารถที่จะติดตั้งซอฟต์แวร์มาตรฐานอื่น เช่น ODBC หรือ JDBC ดังแสดงในรูป เพื่อให้ Client Program ต่างๆที่ใช้อินเตอร์เฟสมาตรฐานเหล่านี้สามารถติดต่อกับ PostgreSQL ได้เช่นกัน

5.4 ข้อมูลเพิ่มเติมอื่นๆ



รูปที่ 5.5 แสดงเว็บไซต์ของ PostgreSQL

Web Site อย่างเป็นทางการของ PostgreSQL อยู่ที่ http://www/postgresql.org โดยมี Mirror Sites ให้บริการกระจายอยู่มากกว่า 10 ประเทศ รูปที่ 5.1 แสดงหน้าจอ Web Site ของ www.postgresql.org ซึ่งเราสามารถค้นหาข้อมูลต่างๆ เกี่ยวกับ PostgreSQL ไม่ว่าจะเป็น Source Code หรือ Binary โปรแกรมบนแพลตฟอร์มต่างๆ ที่จัดเตรียมไว้ให้ เอกสารต่างๆ รวมทั้งข่าวคราวความเคลื่อนไหวล่าสุด

บทที่ 6

การออกแบบ Application

ในบทนี้จะกล่าวถึงการนำสิ่งต่างๆ ที่ได้ศึกษามาทำการสร้างโปรแกรมประยุกต์ซึ่งได้เลือกการเก็บวงจรดิจิตอลมาเป็นกรณีศึกษา

6.1 Requirement และ Specification ของ Application

6.1.1 Server

โปรแกรมประยุกต์จะทำการอ่านข้อมูลเพื่อจัดการทุกอย่างจากฝั่งเซิร์ฟเวอร์ โดยจะทำการสร้าง Store Procedure ซึ่งเขียนโดยใช้ภาษา Pl/pgSQL ซึ่งเป็นภาษาที่มีความสามารถของ SQL และเพิ่มความสามารถในการทำการคอนโทรลการทำงานลงไป (Flow Control) เช่น การทำลูป (Looping) หรือ การทำทางเลือก (Branching) เป็นตัวตรวจสอบและจัดการกับข้อมูลให้แก่Client

6.1.2 Application Programming Interface (API)

การติดต่อระหว่างโปรแกรมประยุกต์กับฐานข้อมูลจะติดต่อผ่านทาง Open Database Connectivity (ODBC) ซึ่งให้ API method ในการติดต่อกับระบบจัดการฐานข้อมูล PostgreSQL กับภาษาที่ใช้ในการพัฒนา C# ทำให้การติดต่อกับฐานข้อมูลทำได้โดยง่าย ลดการเขียนโค๊ดลงไปได้มาก

6.1.3 Client

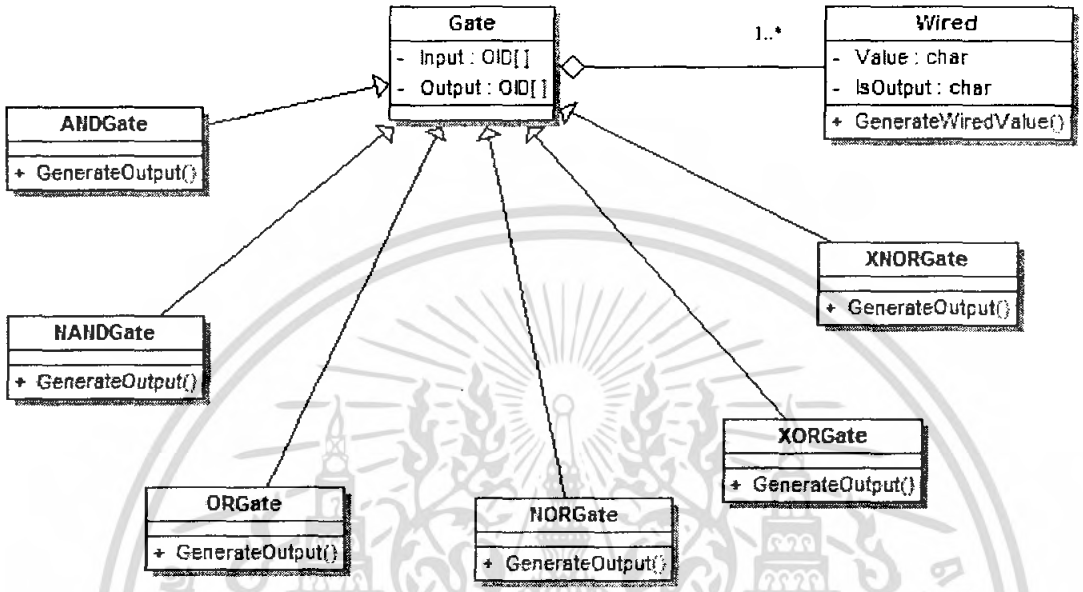
การทำงานของโปรแกรมประยุกต์สามารถสร้างวงจรเกตอย่างง่าย โดยเมื่อทำการสร้างวงจรเสร็จแล้วจะต้องทำการกำหนดด้วยว่าอะไรคืออินพุทและเอาท์พุทของระบบ จากนั้นตรวจสอบความถูกต้องของวงจรว่าสามารทำการ Simulate ได้หรือไม่ก่อนที่จะบันทึกข้อมูลของวงจรลงฐานข้อมูล ถ้าวงจรถูกต้องก็สามารถทำการ Simulate วงจรเพื่อหาเอาท์พุดออกมาได้

สรุปความสามารถของโปรแกรมประยุกต์

1. สามารถสร้างวงจรดิจิตอลอย่างง่ายซึ่งประกอบไปด้วยเกตชนิดต่างๆ ได้
2. สามารถทำการ Simulate การทำงานของวงจรได้

6.2 Analysis และ Design

6.2.1 การออกแบบฐานข้อมูล



รูปที่ 6.1 แสดงการออกแบบฐานข้อมูลเชิงวัตถุสัมพันธ์

การจัดเก็บวงจรดิจิตอลประกอบไปด้วยเกตต่างๆ เป็นไปตาม Class Diagram ดังภาพ ซึ่งประกอบไปด้วยคลาสต่างๆ ดังต่อไปนี้

• GATE เป็นคลาสที่ประกอบไปด้วย Attribute ดังต่อไปนี้

1. INPUT เป็น Array ของ OID ใช้แสดง OID ของเส้นวงจรที่ต่อเข้ามาที่อินพุทของ เกต

2. OUTPUT เป็น Array ของ OID ใช้แสดง OID ของเส้นวงจรที่ต่อเข้ามาที่เอาท์พุทของเกต

จากแนวคิดในการออกแบบเชิงวัตถุ เกตแต่ละชนิดจะมีคุณสมบัติที่เหมือนกันจะแตกต่างกันที่วิธีในการหาค่าของเอาท์พุทซึ่งเกตแต่ละตัวจะมีวิธีเฉพาะของตัวเอง จากเหตุผลดังกล่าวจึงทำให้การออกแบบให้เกตแต่ละชนิดทำการรับการถ่ายทอดคุณสมบัติจากคลาสแม่หรือซูเปอร์คลาส (Class GATE) ในการถ่ายทอดคุณสมบัติพื้นฐานมาให้แก่คลาสลูกหรือซับคลาส ซึ่งคือคลาสเกตแต่ละชนิด ทำการเขียน Method ที่ใช้ในการหาค่าเอาท์พุทเฉพาะแต่ละชนิดของเกต โดยในจาก PostgreSQL นั้นสนับสนุนการถ่ายทอกคุณสมบัตรในกรณีของ Table

• เกตประเภทต่างๆ ANDGATE, ORGATE, NANDGATE, NORGATE, XORGATE, XNORGATE เป็นคลาสที่ต้องเพิ่มส่วน Method ที่ทำการหาค่าเอาท์พุทเพิ่มขึ้นมาซึ่งเกตแต่ละชนิดจะต้องทำการถ่ายทอดคุณสมบัติจากคลาสเกต

• WIRED เป็นคลาสที่ทำการเก็บเส้นวงจรที่ทำการเชื่อมต่อแต่ละเกต โดยคลาส WIRED ประกอบไปด้วย Attribute ดังต่อไปนี้

1. VALUE เป็น Character เก็บค่าสถานะของเส้นวงจรนั้นว่าเป็น LOW หรือ HIGH

- H คือ สถานะของเส้นวงจรนี้เป็น HIGH
- N คือ เส้นวงจรนี้ยังไม่ถูกการกำหนดสถานะ
- L คือ สถานะของเส้นวงจรนี้เป็น LOW

2. ISOUTPUT เป็น Character ที่ทำการเก็บค่าว่าเส้นวงจรนั้นเป็นอินพุทหรือเอาท์พุทของระบบ ดังแสดงดังต่อไปนี้

- I คือ เส้นวงจรนี้เป็นอินพุทของระบบ
- N คือ เส้นวงจรนี้ไม่เป็นทั้งอินพุทและเอาท์พุทของระบบ
- O คือ เส้นวงจรนี้เป็นเอาท์พุทของระบบ

6.2.2 ภาษา SQL ที่ใช้การออกแบบฐานข้อมูล

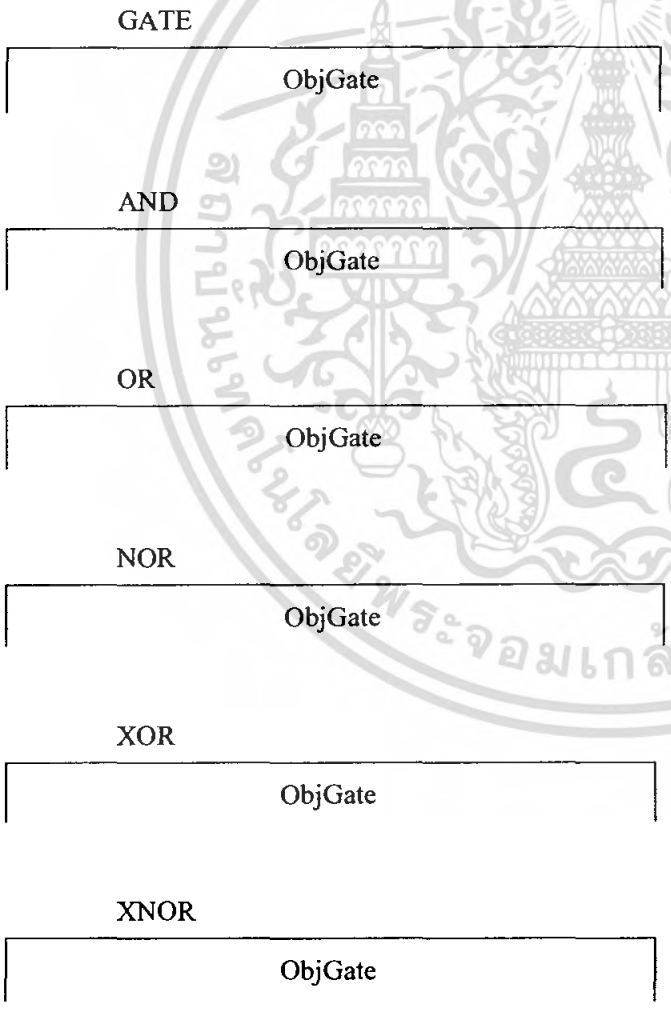
```
CREATE TYPE MyGate AS (
    Input    Integer[],
    Output   Integer[]
);

CREATE TYPE MyWired AS (
    Value          char,
    IsOutput       char
);

CREATE TABLE GATE (
    ObjGate        MyGate
);
```

```
CREATE TABLE WIRED (
    ObjWired        MyWired
);

CREATE TABLE ANDGATE () INHERITS (GATE);
CREATE TABLE NANDGATE () INHERITS (GATE);
CREATE TABLE ORGATE () INHERITS (GATE);
CREATE TABLE NORGATE () INHERITS (GATE);
CREATE TABLE XORGATE () INHERITS (GATE);
CREATE TABLE XNORGATE () INHERITS (GATE);
```



• GATE เป็นรีเลชั่นที่ประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

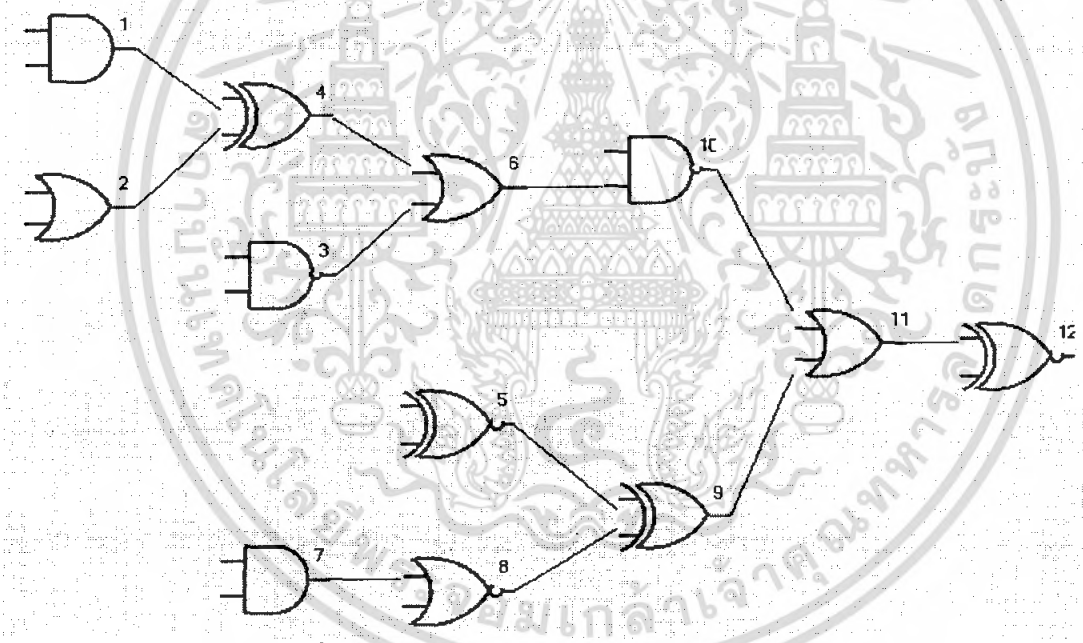
ObjGate มีชนิดเป็น MyGate ซึ่งเป็น Composite Type ซึ่งประกอบไปด้วยฟิลด์เนมตามแอททริบิวท์ของคลาสเกต

• ANDGATE, ORGATE, NANDGATE, NORGATE, XORGATE, XNORGATE เป็นรีเลชั่นที่ได้รับการถ่ายทอดคุณสมบัติมาจากรีเลชั่นเกต โดยที่แอททริบิวท์นั้นประกอบไปด้วยแอททริบิวท์ที่ได้รับการถ่ายทอดจากรีเลชั่นเกต

• WIRED เป็นรีเลชั่นที่ทำการเก็บเส้นวงจรที่ทำการเชื่อมต่อแต่ละเกต โดยประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

• ObjWired มีชนิดเป็น MyWired ซึ่งเป็น Composite Type ซึ่งประกอบไปด้วยฟิลด์เนมตามแอททริบิวท์ของคลาส WIRED

ตัวอย่างการจัดเก็บวงจรดิจิตอลบนฐานข้อมูล



รูปที่ 6.2 แสดงตัวอย่างวงจรดิจิตอลที่ทำการจัดเก็บ

AND

ตาราง 6.1 แสดงการจัดเก็บ AND Gate

OID	ObjGate
92235	(“{92236,92237}”,{92238})
92255	(“{92256,92257}”,{92258})

(2 rows)

NAND

ตาราง 6.2 แสดงการจัดเก็บ NAND Gate

OID	ObjGate
92243	("{92244,92245}",{92246})
92264	("{92265,92254}",{92266})

(2 rows)

OR

ตาราง 6.3 แสดงการจัดเก็บ OR Gate

OID	ObjGate
92239	("{92240,92241}",{92242})
92253	("{92248,92246}",{92254})
92267	("{92266,92263}",{92268})

(3 rows)

NOR

ตาราง 6.4 แสดงการจัดเก็บ NOR Gate

OID	ObjGate
92259	("{92258,92260}",{92261})

(1 rows)

XOR

ตาราง 6.5 แสดงการจัดเก็บ XOR Gate

OID	ObjGate
92247	("{92238,92242}",{92252})
92262	("{92252,92261}",{92263})

(2 rows)

XNOR

ตาราง 6.6 แสดงการจัดเก็บ XNOR Gate

OID	ObjGate
92249	("{92250,92251}",{92252})
92269	("{92268,92270}",{92271})

(2 rows)

Gate

ตาราง 6.7 แสดงการจัดเก็บ Gate

OID	ObjGate
92235	(“{92236,92237}”,{92238})
92255	(“{92256,92257}”,{92258})
92243	(“{92244,92245}”,{92246})
92264	(“{92265,92254}”,{92266})
92239	(“{92240,92241}”,{92242})
92253	(“{92248,92246}”,{92254})
92267	(“{92266,92263}”,{92268})
92259	(“{92258,92260}”,{92261})
92247	(“{92238,92242}”,{92252})
92262	(“{92252,92261}”,{92263})
92249	(“{92250,92251}”,{92252})
92269	(“{92268,92270}”,{92271})

(12 rows)

WIRED

ตาราง 6.8 แสดงการจัดเก็บ Wired

OID	ObjWired
92236	(H , I)
92237	(H , I)
92240	(L , I)
92241	(H , I)
92244	(L , I)
92245	(L , I)
92250	(H , I)
92251	(H , I)
92256	(H , I)
92257	(L , I)
92260	(L , I)

92265	(H , I)
92270	(L , I)
92238	(H , N)
92258	(L , N)
92246	(H , N)
92266	(L , N)
92242	(H , N)
92254	(H , N)
92268	(L , N)
92261	(H , N)
92248	(L , N)
92263	(L , N)
92252	(H , N)
92221	(H , O)

(25 rows)

6.2.3 การออกแบบโปรแกรมประยุกต์

การทำงานของโปรแกรมประยุกต์จะทำงานในลักษณะของ Client-Server โดยการทำงานที่ฝั่ง Client จะเน้นที่การสร้างวงจรดิจิตอลแสดงผลข้อมูล มีรูทีนที่ทำการจัดเก็บข้อมูลลงฐานข้อมูล เมื่อต้องการทำการจำลองการทำงานจะทำการเรียกรูทีนที่ฝั่งเซิร์ฟเวอร์โดยส่งคำสั่งไป โดยกำหนดให้การจัดการทั้งหมดอยู่ที่ฝั่งเซิร์ฟเวอร์

เมื่อเริ่มเข้าสู่การทำงานของโปรแกรมประยุกต์ผู้ใช้จะต้องสร้างโปรเจคใหม่ จากนั้นทำการวาดวงจรที่ต้องการ โดยเริ่มจากการวาดเกตแล้วลากเส้นวงจรเชื่อมต่อแต่ละเกต โดยในการลากเส้นวงจรจะต้องมีการกำหนดหมายเลขเกมและขาของเกตที่เกตที่ต้องการจะเชื่อมต่อ

หลังจากวาดลายวงจรเสร็จแล้วผู้ใช้ทำการจำลองการทำงานระบบจะทำการตรวจสอบความถูกต้องของวงจรว่าวงจรที่สร้างขึ้นสามารถคำนวณหาค่าได้หรือไม่ โดยถ้าวงจรถูกต้องก็จะทำการบันทึกวงจรลงฐานข้อมูล โดยจะมีโมดูลของโปรแกรมประยุกต์ทำหน้าที่ในการนำข้อมูลลงไปเก็บบนฐานข้อมูล จากนั้น

โปรแกรมประยุกต์ก็จะทำการคำนวณหาค่าเอาท์พุทซึ่งเป็นหน้าที่ฝั่งเซิร์ฟเวอร์เมื่อคำนวณค่าเสร็จก็จะส่งค่าเอาท์พุทคืนมาให้โปรแกรมประยุกต์นำไปแสดงผลต่อไป

6.2.4 การออกแบบการจัดการดาต้าเบสเซิร์ฟเวอร์

การทำงานที่ฝั่งดาต้าเบสเซิร์ฟเวอร์นอกจากการจัดเก็บข้อมูลแล้วที่ฝั่งดาต้าเบสเซิร์ฟเวอร์จะมีการฝังรูทีนให้ฝั่ง Client มาเรียกใช้ซึ่งเรียกรูทีนดังกล่าวว่า Stored Procedure ของ PostgreSQL ซึ่งถูกเขียนด้วยภาษา SQL และ Pl/pgSQL ซึ่งเป็นภาษาที่ติดตั้งมาพร้อมกับ PostgreSQLซึ่งมีความสามารถของ SQL และเพิ่มความสามารถในการทำการคอนโทรลการทำงานลงไป (Flow Control) เช่น การทำลูป (Looping) หรือ การทำทางเลือก (Branching) โดยที่ฝั่งเซิร์ฟเวอร์จะประกอบไปด้วยรูทีนต่างๆ ดังต่อไปนี้

ชื่อรูทีน	InputGate
หน้าที่	ทำการ Insert Gate ลงฐานข้อมูล
ถูกเรียกใช้เมื่อ	Client จำลองการทำงาน ก่อนที่จะคำนวณหาค่าเอาท์พุท
รายละเอียด	จะทำการ Insert Gate ลงฐานข้อมูลทีละObjectพร้อมทั้ง Return เลข OID ของRow ที่ถูก insert ล่าสุด

ชื่อรูทีน	InputWired
หน้าที่	ทำการ Insert Wired ลงฐานข้อมูล
ถูกเรียกใช้เมื่อ	Client จำลองการทำงาน ก่อนที่จะคำนวณหาค่าเอาท์พุท
รายละเอียด	จะทำการ Insert Wired ลงฐานข้อมูลทีละ Object ตามพารามิเตอร์ที่ส่งเข้ามาซึ่งประกอบไปด้วยสถานะของสายวงจรและเอาท์พุทของสายวงจร พร้อมทั้ง return เลข OID ของ Row ที่ถูก insert ล่าสุด

ชื่อรูทีน	GenOutput< and|or|nand|nor|xor|xnor >
หน้าที่	ทำการสร้างค่าเอาท์พุทจากอินพุทที่เป็นพารามิเตอร์ที่ส่งเข้ามา
ถูกเรียกใช้เมื่อ	ถูกเรียกใช้ภายในรูทีน GenFinalOutput ขณะทำการคำนวณหาค่าเอาท์พุท
รายละเอียด	ทำการตรวจสอบค่าอินพุทและคำนวณตามหลักของเกตแต่ละชนิดแล้วคืนค่า 'H' หรือ 'L'

ชื่อรูทีน	GenFinalOutput
หน้าที่	ทำการคำนวณหาค่าเอาท์พุททั้งหมดทั้งโปรเจค
ถูกเรียกใช้เมื่อ	ฝั่งClient ทำการจำลองการทำงานของวงจรเพื่อดูค่าเอาท์พุท

รายละเอียด	วนลูปทำการเรียก SPL GenOutput< and|or|nand|nor|xor|xnor > ของแต่ละเกตและจะหยุดลูปเมื่อทำการหาค่าเอาท์พุทครบทุกเกตหรือ CheckFinalOutput คืนค่าเป็นศูนย์

ชื่อรูทีน	GetWiredValue
หน้าที่	ทำการหาค่าสถานะของเส้นวงจร
ถูกเรียกใช้เมื่อ	ถูกเรียกใช้ภายในรูทีน GenOutput< and|or|nand|nor|xor|xnor >
รายละเอียด	ทำการหาค่าสถานะของเส้นวงจรโดยตรวจสอบจากเลข OID ซึ่งเป็นพารามิเตอร์ที่ส่งเข้ามา

ชื่อรูทีน	CheckFinalOutput
หน้าที่	ตรวจสอบว่าเกตทุกเกตในโปรเจคทำการหาค่าเอาท์พุทครบทุกเกตแล้วหรือไม่
ถูกเรียกใช้เมื่อ	ถูกเรียกใช้ภายในรูทีน GenFinalOutput ขณะทำการคำนวณหาค่าเอาท์พุท
รายละเอียด	ทำการตรวจสอบจากเส้นวงจรโดยทำการนับเส้นวงจรที่ยังไม่ได้คำนวณหาค่าเอาท์พุทซึ่งสถานะของเส้นจะเท่ากับ 'N' แล้วคืนค่าจำนวนเส้นวงจรที่ยังไม่ได้คำนวณหาค่าเอาท์พุท นั้นคือถ้าคืนค่าเป็นศูนย์แสดงว่าเกตทุกเกตในโปรเจคทำการหาค่าเอาท์พุทครบทุกเกต

บทที่ 7
โปรแกรมประยุกต์ Logic Simulator

ในบทนี้จะกล่าวถึงการจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงสัมพันธ์เปรียบเทียบการใช้งานฐานข้อมูลเชิงวัตถุสัมพันธ์ รายละเอียดของตัวโปรแกรมประยุกต์ที่เราได้ทำการสร้าง และวิธีการใช้งานโปรแกรมประยุกต์

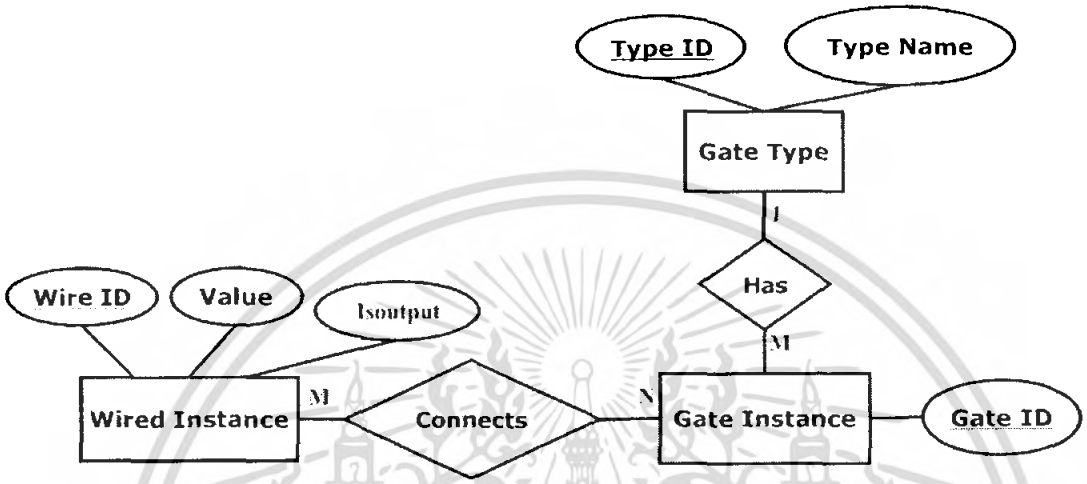
7.1 การพัฒนาการจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงสัมพันธ์

จากการที่ฐานข้อมูลเชิงสัมพันธ์ไม่สนับสนุนการจัดการเกี่ยวกับ Object ในแนวคิดการเขียนโปรแกรมเชิงวัตถุ ดังนั้นการจัดเก็บ Object หนึ่งต้องแตกออกเป็นหลายๆ Row กระจายอยู่ในหลายๆ Relation ซึ่งการจัดเก็บซึ่งการจัดเก็บด้วยวิธีดังกล่าวนั้นอาจจะทำให้เกิดข้อเสีย ดังนี้

1. ยากต่อการทำความเข้าใจของผู้ใช้งาน เนื่องจากการที่ต้องมีการกระจายไปในหลายๆ Relation ทำให้ข้อมูลอาจอยู่รูปที่ทำความเข้าใจได้ยากและข้อมูลที่แสดงอาจไม่สามารถสื่อความหมายให้เข้าใจได้หากดูจาก Relation เพียง Relation เดียว
2. การทำ Operation ใดๆ กับ Object ที่ทำการจัดเก็บอยู่ไม่สามารถทำได้ในครั้งเดียวต้องมีการเรียกใช้งานคำสั่งหลายคำสั่งเพราะข้อมูลได้มีการกระจายไปในหลาย Relation ทำให้การนำเอาข้อมูลมารวมกันเพื่อการทำ Operation เกิดความยุ่งยาก
3. ในฐานข้อมูลเชิงสัมพันธ์นั้นไม่สามารถเก็บข้อมูลประเภท Object ได้ดังนั้นถ้าหากต้องการทำการจัดเก็บข้อมูลประเภท Object ต้องทำการแปลง Attribute ของ Object ลงในคอลัมน์เป็นไปได้ว่าทำให้ตารางมีขนาดใหญ่และอาจจะเต็มไปด้วยค่า NULL หรือถ้าทำการ Normalize ตารางโดยใช้ Foreign Key นั้นทำให้การคายรี่ทุกครั้งต้องทำการ Join ตาราง ดังนั้นการใช้โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์จึงเหมาะสมและง่ายกว่าที่ใช้จัดการกับข้อมูลประเภท Object มากกว่าโมเดลฐานข้อมูลเชิงสัมพันธ์
4. ข้อมูลบางครั้งมีลักษณะที่เป็นลำดับชั้น เช่นกรณีของลูกจ้างกับนายจ้าง โดยถ้าในโมเดลฐานข้อมูลเชิงสัมพันธ์จำเป็นต้อง Foreign Key หรือคอลัมน์ที่เป็น Identifier แต่ถ้าในโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์สามารถใช้ความสัมพันธ์ของซุปเปอร์คลาสกับซับคลาส
5. ในโมเดลฐานข้อมูลเชิงสัมพันธ์ผู้ใช้ต้องกำหนด Primary Key เองเพื่อสร้างความแตกต่างในแต่ละทูเปิล แต่ในโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ไม่จำเป็นต้อง

ดูแล Primary Key เพราะ ORDBMS จะควบคุม Primary key ให้อัตโนมัติผ่านทาง OID

7.1.1 ตัวอย่างการจัดเก็บโดยใช้ฐานข้อมูลเชิงสัมพันธ์



รูปที่ 7.1 แสดงฐานข้อมูลเชิงสัมพันธ์ที่ทำการออกแบบ

Gate Instance

P.K	F.K
GID	GateType

GateType

P.K	
TypeID	TypeName

Wired Instance

P.K	F.K	F.K		
WID	GID1	GID2	Value	Isoutput

- Gate Instance เป็นรีเลชั่นที่ประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

    1. GID เป็น Integer ใช้แสดงหมายเลขของเกต และเป็น Primary Key ของรีเลชั่น

    2. GateType เป็น String ใช้แสดงชนิดของเกต

- Wired Instance เป็นรีเลชั่นที่ทำการเก็บเส้นวงจรที่ทำการเชื่อมต่อแต่ละเกต โดยตาราง Wired Instance ประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

    1. WID เป็น Integer ใช้แสดงหมายเลขของเส้นวงจร เป็น Primary Key ของรีเลชั่น

    2. GID1 เป็น Integer ใช้แสดงหมายเลขของเกตตัวที่หนึ่งซึ่งต่ออยู่กับเส้นวงจรเส้นดังกล่าว

    3. GID2 เป็น Integer ใช้แสตงหมายเลขของเกตตัวที่สองซึ่งต่ออยู่กับเส้นวงจรเส้นดังกล่าว

    4. Value เป็น Character เก็บค่าสถานะของเส้นวงจรนั้นว่าเป็น LOW หรือ HIGH

        - H คือ สถานะของเส้นวงจรนี้เป็น HIGH
        - N คือ เส้นวงจรนี้ยังไม่ถูกการกำหนดสถานะ
        - L คือ สถานะของเส้นวงจรนี้เป็น LOW

    5. Isoutput เป็น Character ที่ทำการเก็บค่าว่าเส้นวงจรนั้นเป็นอินพุทหรือเอาท์พุทของระบบ ดังแสดงดังต่อไปนี้

        - I คือ เส้นวงจรนี้เป็นอินพุทของระบบ
        - N คือ เส้นวงจรนี้ไม่เป็นทั้งอินพุทและเอาท์พุทของระบบ
        - O คือ เส้นวงจรนี้เป็นเอาท์พุทของระบบ

7.2 การพัฒนาการจัดเก็บวงจรดิจิตอลโดยใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์

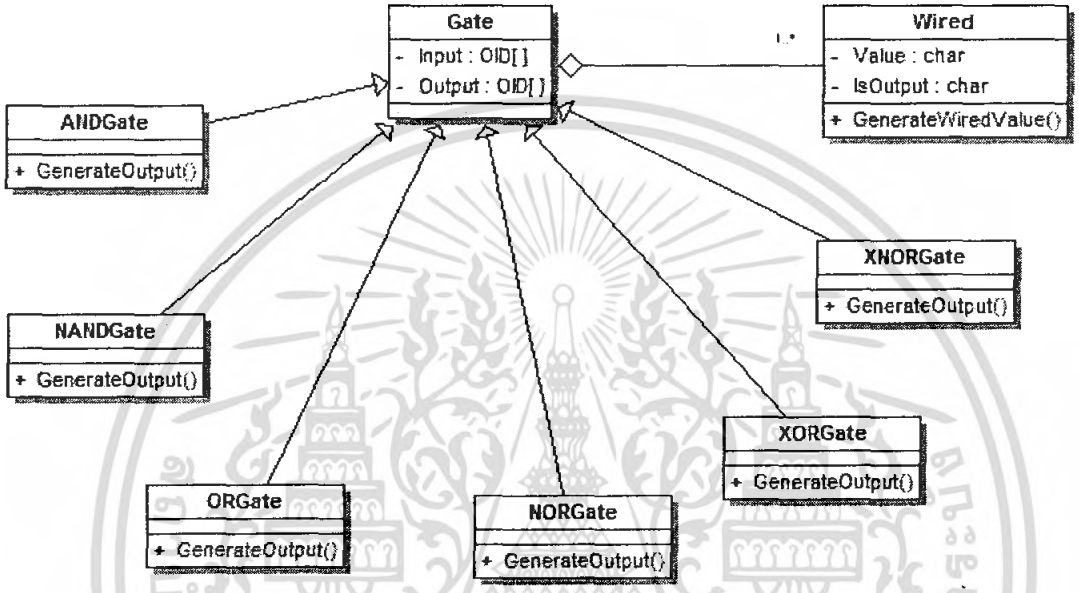
จากปัญหาต่างๆในการจัดเก็บข้อมูลโดยใช้โมเดลฐานข้อมูลเชิงสัมพันธ์แสดงให้เห็นว่าการจัดเก็บวงจรดิจิตอลไม่เหมาะสมกับการใช้งานร่วมกับฐานข้อมูลเชิงสัมพันธ์ ในโครงงานจึงได้นำเอาโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์มาแก้ปัญหาเกี่ยวกับการใช้งานข้อมูลที่มีความซับซ้อนซึ่งทำให้

1. มองข้อมูลที่จัดเก็บเป็น Object ไม่ต้องทำการแตกออกเป็นหลายๆ Row ทำให้ง่ายต่อการทำความเข้าใจของผู้ใช้งาน

2. การทำ Operation ใดๆ กับ Object ทำได้ในคำสั่งเดียวโดยการสร้าง Method ให้กับ Object นั้น

3. ลดความซ้ำซ้อนในการจัดเก็บข้อมูลทำให้การดูแลความถูกต้องของฐานข้อมูลทำได้ง่ายกว่า

7.2.1 ตัวอย่างการจัดเก็บโดยใช้ฐานข้อมูลเชิงวัตถุสัมพันธ์



รูปที่ 7.2 แสดงการออกแบบฐานข้อมูลเชิงวัตถุสัมพันธ์

GATE

ObjGate

ANDGATE

ObjGate

ORGATE

ObjGate

NORGATE

ObjGate

XORGATE

ObjGate

XNORGATE

ObjGate

• GATE เป็นรีเลชั่นที่ประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

ObjGate มีชนิดเป็น MyGate ซึ่งเป็น Composite Type ซึ่งประกอบไปด้วยฟิลด์เนมตามแอททริบิวท์ของคลาสเกต

• ANDGATE, ORGATE, NANDGATE, NORGATE, XORGATE, XNORGATE เป็นรีเลชั่นที่ได้รับการถ่ายทอดคุณสมบัติมาจากรีเลชั่นเกต โดยที่แอททริบิวท์นั้นประกอบไปด้วยแอททริบิวท์ที่ได้รับการถ่ายทอดจากรีเลชั่นเกต

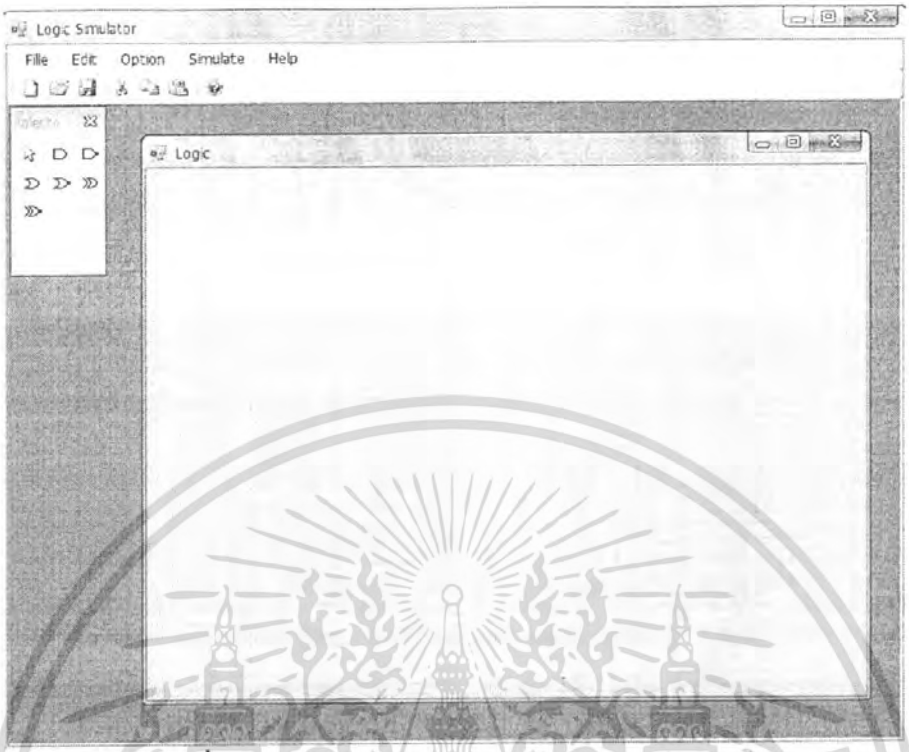
• WIRED เป็นรีเลชั่นที่ทำการเก็บเส้นวงจรที่ทำการเชื่อมต่อแต่ละเกต โดยประกอบไปด้วยแอททริบิวท์ ดังต่อไปนี้

• ObjWired มีชนิดเป็น MyWired ซึ่งเป็น Composite Type ซึ่งประกอบไปด้วยฟิลด์เนมตามแอททริบิวท์ของคลาส WIRED

7.3 การใช้งานโปรแกรมประยุกต์

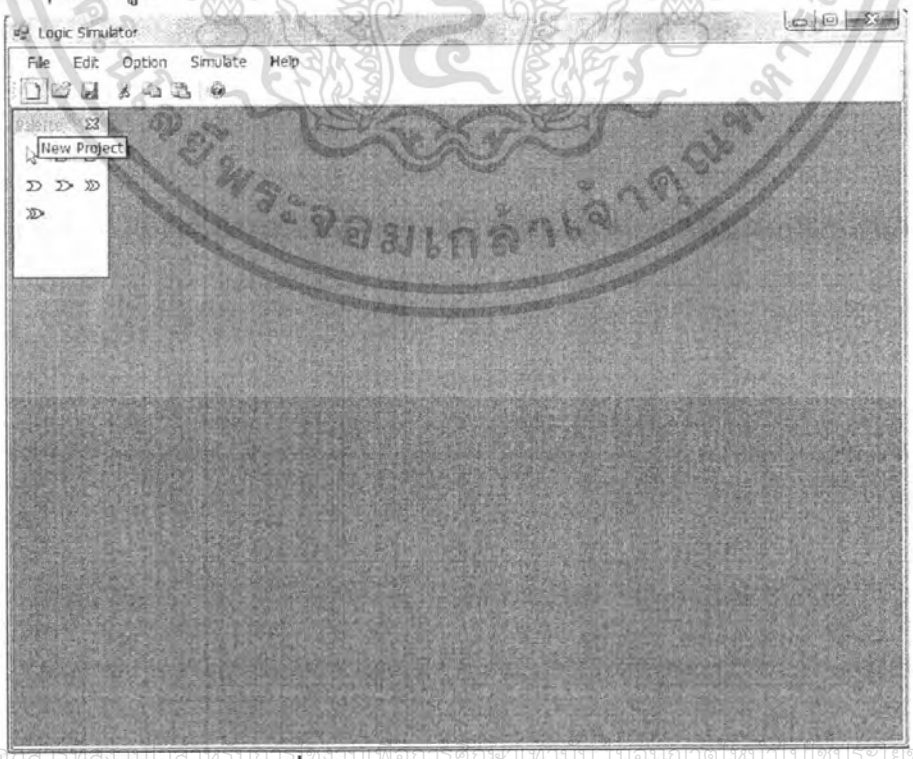
โปรแกรมประยุกต์จะทำการจำลองการทำงานวงจรดิจิตอลที่สร้างขึ้น โดยมีส่วนประกอบหลักที่สำคัญอยู่ 2 ส่วน ได้แก่

1. ส่วนเลย์เอาท์ เป็นส่วนที่ใช้ในการวาดภาพ
2. ส่วนคอมโพเนนท์ เป็นส่วนที่แสดงคอมโพเนนท์ต่างๆ ที่สามารถนำมาใช้สร้างวงจรได้

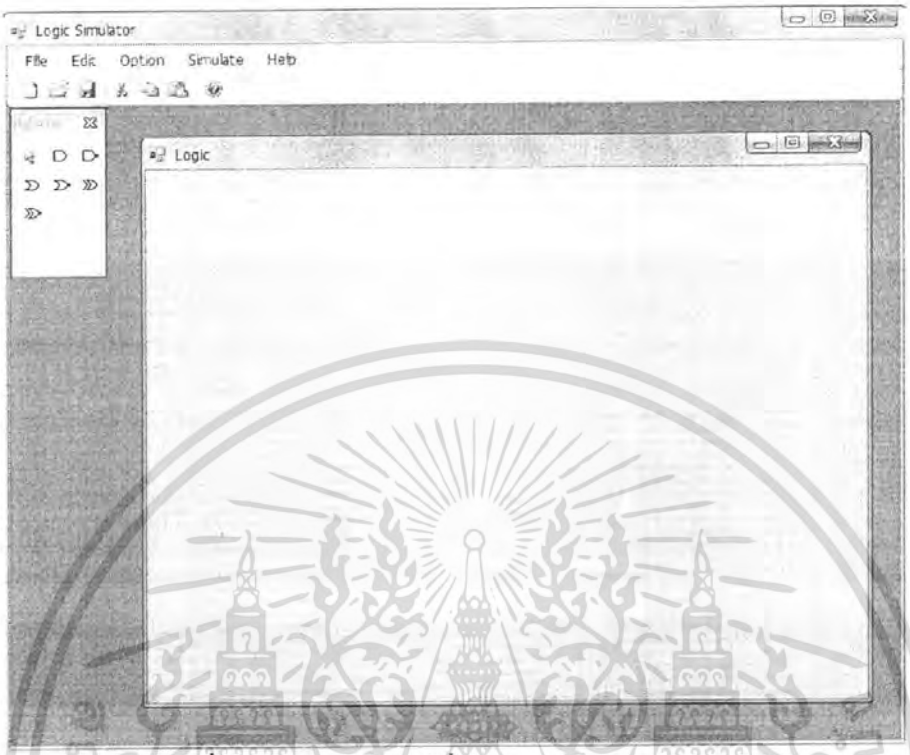


รูปที่ 7.3 แสดงส่วนประกอบของโปรแกรมประยุกต์

ในการเลือกคอมโพเนนท์ที่ต้องการทำได้โดยการคลิกบนคอมโพเนนท์ที่ต้องการคอมโพเนนท์ที่เลือกจะปรากฏบนเลย์เอาท์พร้อมหมายเลขประจำเกตให้สามารถเลือกไปวางบนตำแหน่งต่างๆได้ ดังรูป



รูปที่ 7.4 แสดงการสร้างโปรเจคใหม่



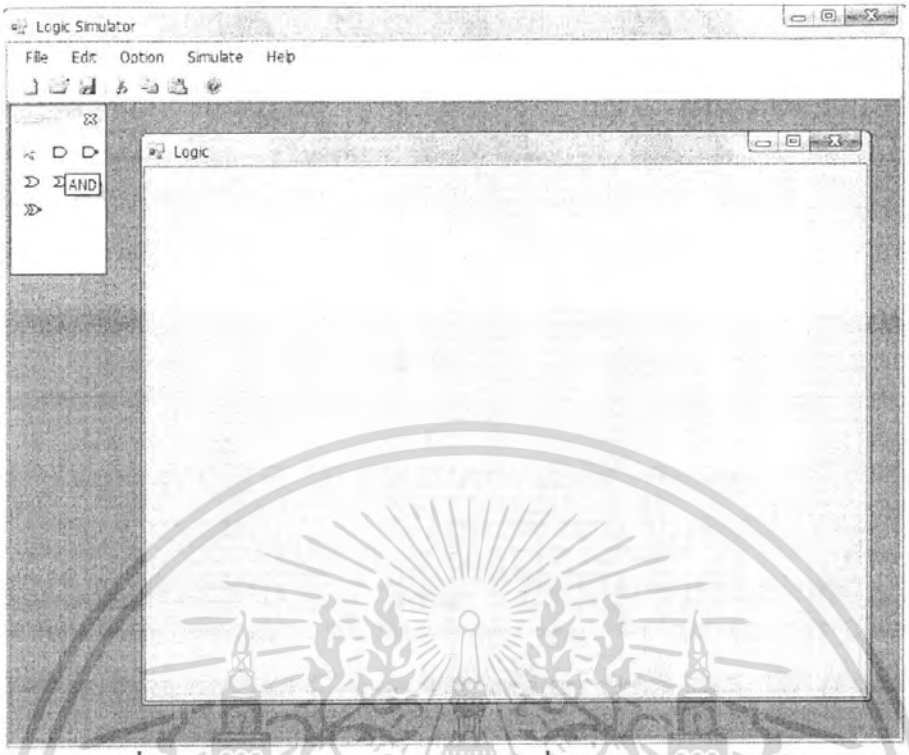
รูปที่ 7.5 แสดงเลย์เอาท์เมื่อทำการสร้างโปรเจคใหม่

ในการเลือกคอมโพเนนท์ที่ต้องการทำได้โดยการคลิกบนคอมโพเนนท์ที่ต้องการคอมโพเนนท์ที่เลือกจะปรากฏบนเลย์เอาท์พร้อมหมายเลขประจำเกตให้สามารถเลือกไปวางบนตำแหน่งต่างๆได้ ดังรูป

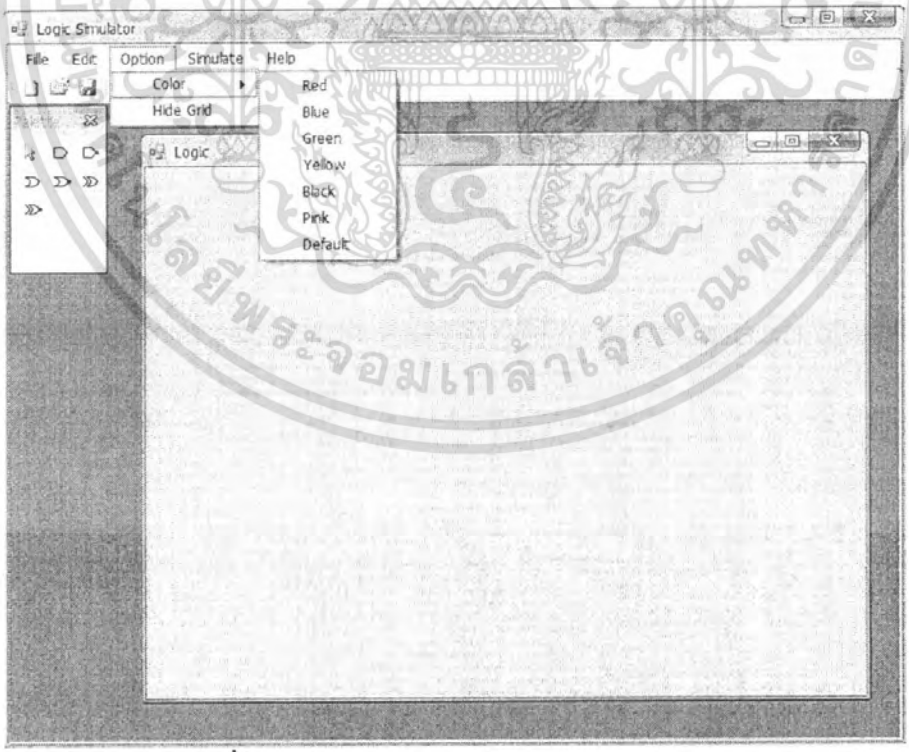
การเชื่อมต่อระหว่างแต่ละคอมโพเนนท์ทำได้โดยการคลิกเมาส์ที่ปุ่มขวาบนคอมโพเนนท์ที่ต้องการจะเชื่อมต่อจะขึ้นคอนเทกซ์เมนูให้เลือก AddLine ดังรูป จากนั้นจะขึ้นฟอร์มขึ้นมาประกอบไปด้วยคอมโบบอกซ์ 2 ส่วนคือ

1. GateID คือ เกตปลายทางที่ต้องการเชื่อมต่อ
2. LegID คือ ขาของเกตปลายทางที่ต้องการเชื่อมต่อ

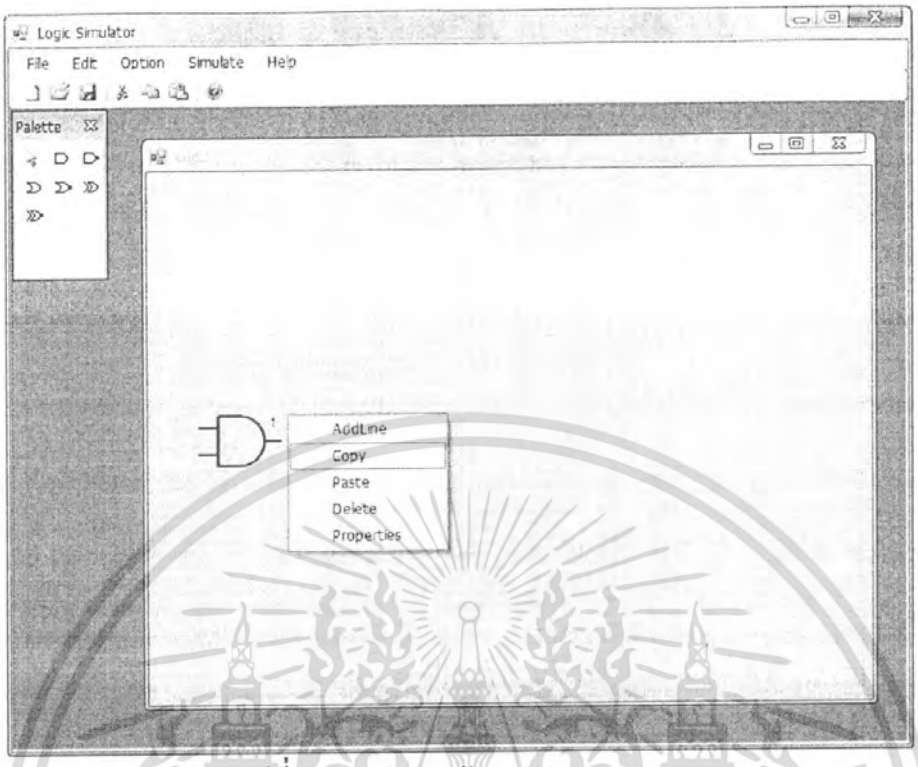
ในกรณีที่ผู้ใช้ต้องการยกเลิกการเชื่อมต่อที่ขาเอาท์พุทของเกตทำได้โดยคลิกเมาส์ที่ปุ่มขวาบนคอมโพเนนท์เกตตัวที่ต้องการแล้วเลือก AddLine จากนั้นที่ช่องคอมโบบอกซ์ GateID เลือก Not Connect เพื่อยกเลิกการเชื่อมต่อที่ขาเอาท์พุทของเกต ดังรูป ส่วนถ้าผู้ใช้ต้องการลบคอมโพเนนท์ทำได้โดยคลิกเมาส์ที่ปุ่มขวาบนคอมโพเนนท์เกตตัวที่ต้องการแล้วเลือก Delete ดังรูป



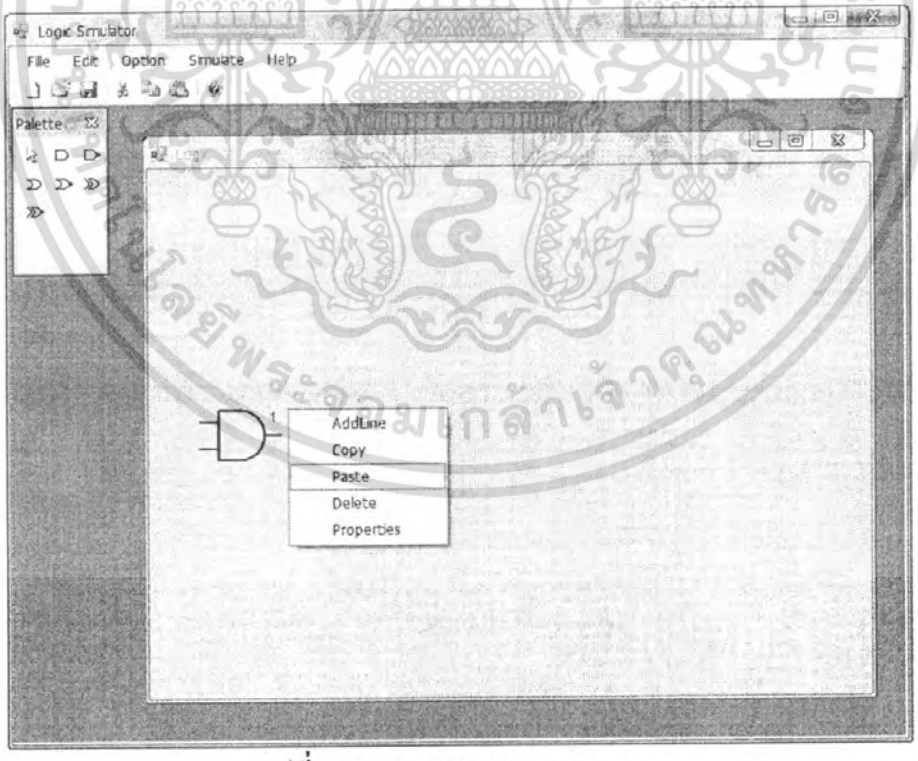
รูปที่ 7.6 แสดงการเลือกคอมโพเนนท์ เพื่อทำการวาดวงจรลอจิก



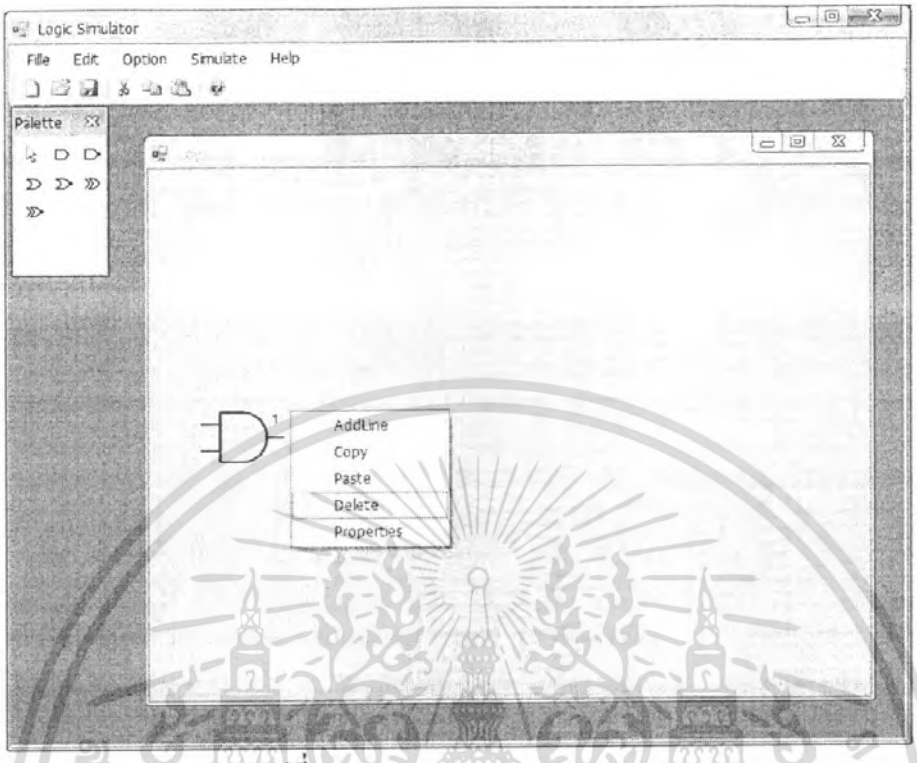
รูปที่ 7.7 แสดงการกำหนดลักษณะการแสดงผล



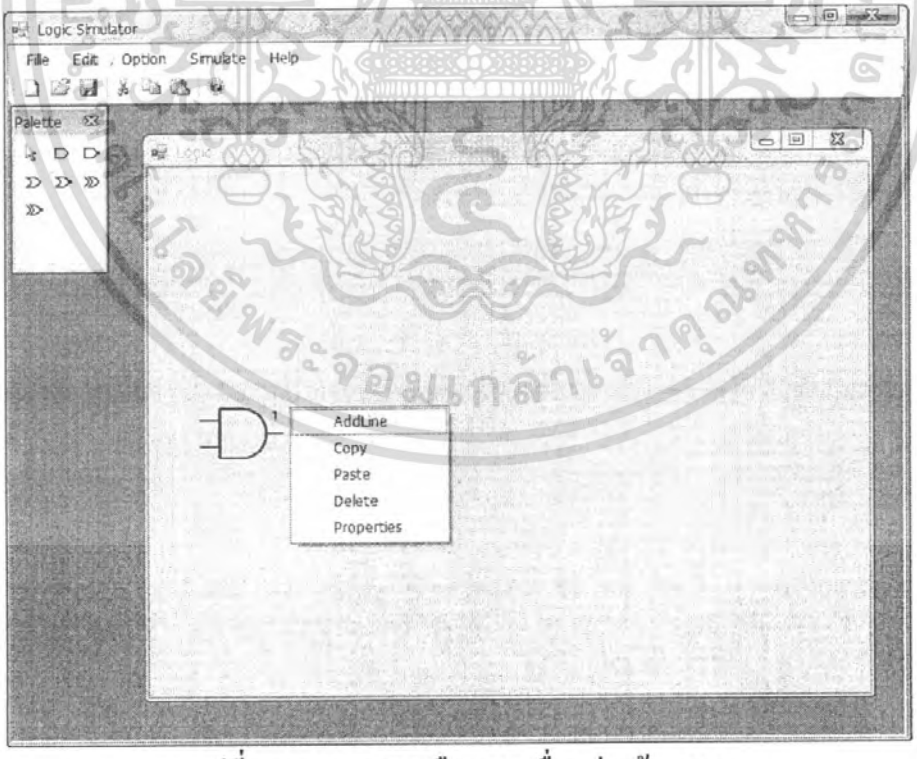
รูปที่ 7.8 แสดงการคัดลอก Component



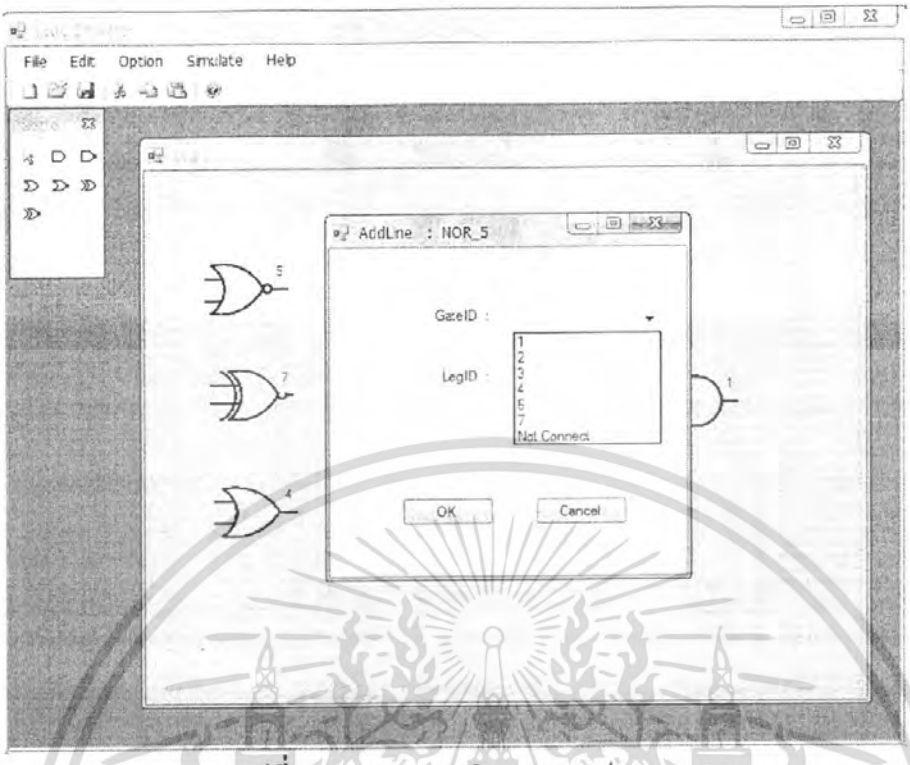
รูปที่ 7.9 แสดงการวาง Component



รูปที่ 7.10 แสดงการลบ Component



รูปที่ 7.11 แสดงการเลือกการเชื่อมต่อเส้นวงจร

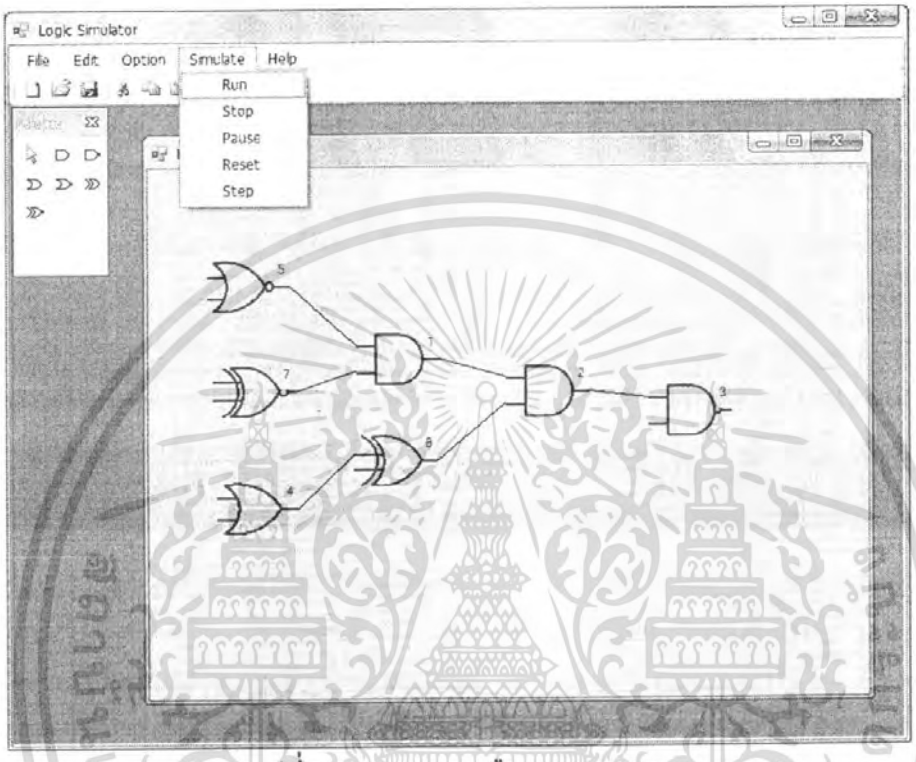


รูปที่ 7.12 แสดงการกำหนดเกตปลายทาง

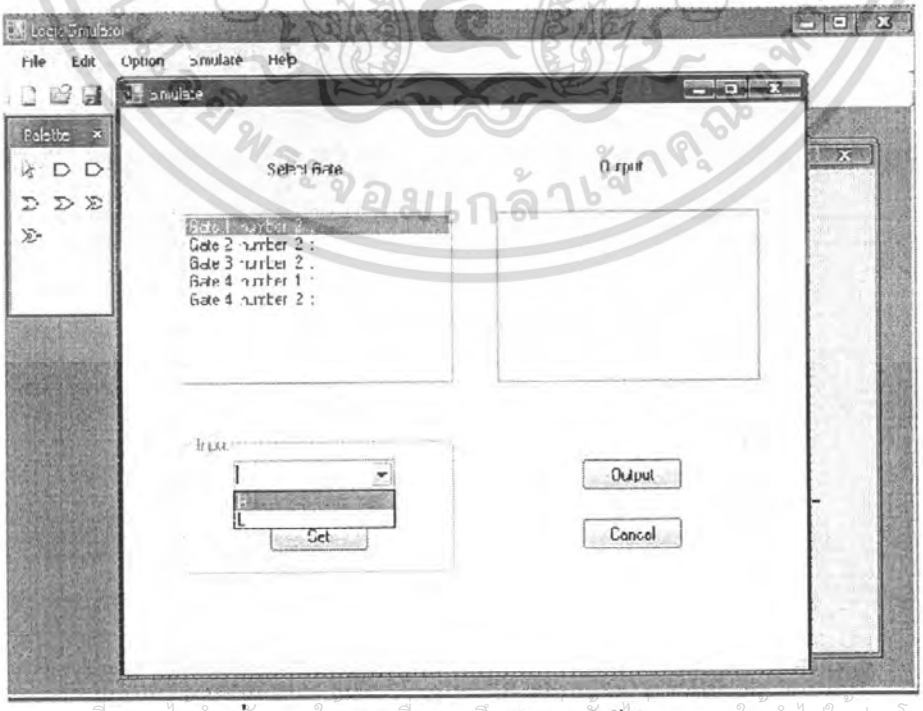
รูปที่ 7.13 แสดงการเลือกขาของเกตปลายทางที่ต้องการเชื่อมต่อ

หลังจากสร้างวงจรเสร็จเมื่อจะทำการ Simulate การทำงานของโปรแกรมทำได้โดยเลือก Simulate->Run ดังรูป ซึ่งจะแสดงแบบฟอร์มให้สามารถใส่ค่าอินพุทระบบขาต่างๆ โดยเราสามารถเลือกใส่ค่า Low-High ได้แล้วทำการเลือกปุ่ม Set เป็นการกำหนดค่าให้ขานั้นและเมื่อเรียบร้อยก็

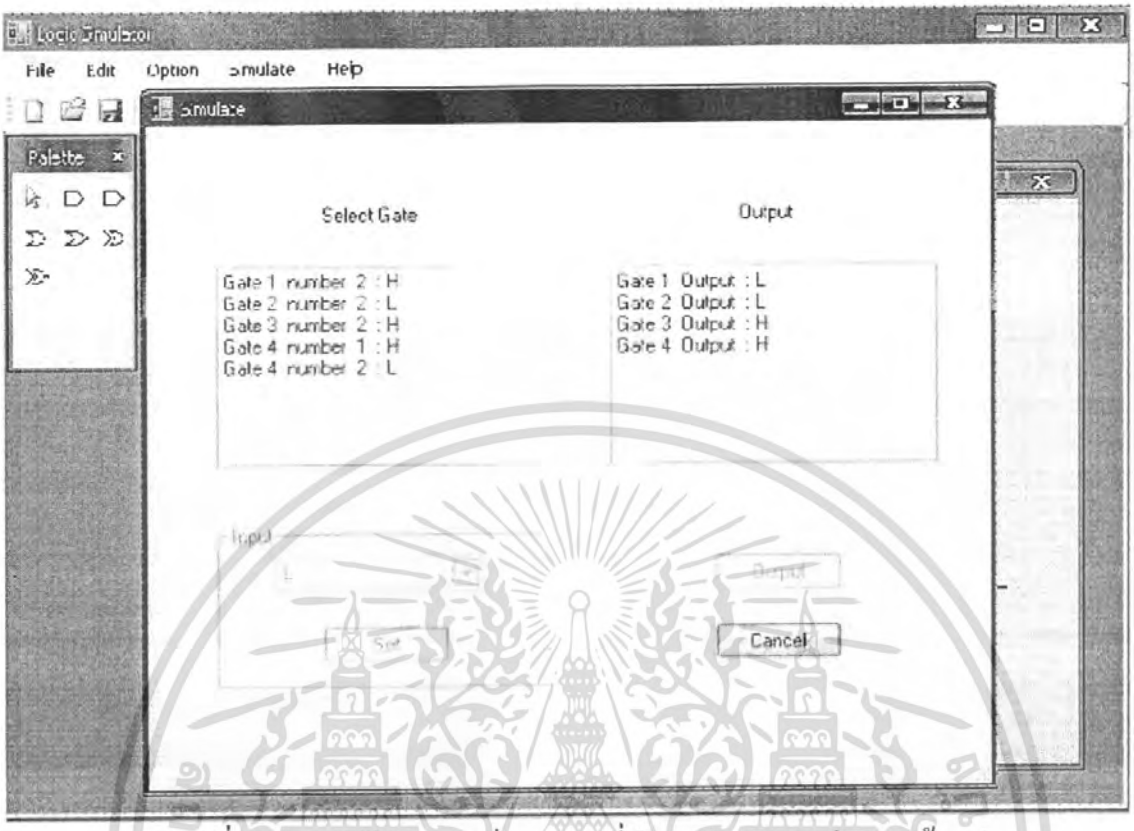
เลือกปุ่ม Output ซึ่งถ้าผู้ใช้ใส่อินพุทของระบบไม่ครบโปรแกรมจะแจ้งเตือนกลับมาที่ผู้ใช้เพื่อให้ใส่อินพุทของระบบให้ครบ เมื่อใส่อินพุทของระบบครบแล้ว โปรแกรมก็จะทำการคำนวณค่าเอาท์พุทโดยจะแสดงค่าเอาท์พุทของแต่ละเกตออกมาในช่องเอาท์พุท ดังรูป



รูปที่ 7.14 แสดงการเลือกการ Simulate



รูปที่ 7.15 แสดงการกำหนดอินพุทให้กับระบบ



รูปที่ 7.16 แสดงอินพุทเอาท์พุทระบบที่ทำการจำลองการทำงานแล้ว

บทที่ 8
บทสรุปและวิจารณ์

ในบทนี้จะกล่าวถึงการทำงานทั้งหมดในภาพรวมผลที่ได้จากงานวิจัยนี้ แนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

8.1 สรุปผลโครงงาน

ในโครงงานได้ทำการศึกษาหลักการของโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ศึกษาภาษาฐานข้อมูลตามมาตรฐาน SQL3 ว่ามีความสามารถอะไรเพิ่มขึ้นมาในการใช้งานร่วมกับโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ แล้วศึกษาการออกแบบและการใช้งานฐานข้อมูลเชิงวัตถุสัมพันธ์เพื่อประยุกต์ใช้ในงานต่างๆ

หลังจากมีความรู้และความเข้าใจโมเดลฐานข้อมูลแล้วได้ทำการศึกษาระบบจัดการฐานข้อมูล (Database Management System: DBMS) ที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ในโครงงานได้ใช้ PostgreSQL version 8.2.4 เป็นระบบจัดการฐานข้อมูลที่สามารถจัดเก็บข้อมูลได้ทั้งที่เป็นโมเดลฐานข้อมูลเชิงสัมพันธ์และโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ ศึกษาและทดลองใช้งานชนิดข้อมูลชนิดต่างๆที่สนับสนุนโมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ที่ระบบจัดการฐานข้อมูลมีให้และศึกษาการใช้งานของระบบจัดการฐานข้อมูลเทียบกับมาตรฐาน SQL3

หลังจากมีความคุ้นเคยกับการใช้งานระบบจัดการฐานข้อมูลก็ได้ทำการทดลองสร้างโปรแกรมประยุกต์ ที่ทำการเก็บข้อมูลที่มีความซับซ้อนซึ่งได้ทำการเลือกการจัดเก็บวงจรดิจิตอลซึ่งประกอบไปด้วยเกทชนิดต่างๆ โดยใช้คุณลักษณะของฐานข้อมูลเชิงวัตถุสัมพันธ์ในการจัดเก็บข้อมูล โดยในการพัฒนาโปรแกรมประยุกต์เริ่มจากการออกแบบฐานข้อมูลโดยยึดแนวคิดการออกแบบตามคุณลักษณะของฐานข้อมูลเชิงวัตถุสัมพันธ์แล้วทำการสร้างรูทีนที่เขียนด้วย Stored Procedure Language ด้วยภาษา Pl/pgSQL ซึ่งเป็นภาษาที่มีมากับ PostgreSQL พร้อมทั้งเป็นภาษาที่มีความสามารถของภาษา SQL แล้วเพิ่มส่วนในการควบคุมการทำงานเช่นการวนลูปและการตัดสินใจเพิ่มเข้าไป โดยรูทีนดังกล่าวจะมีไว้ให้ไคลเอนท์เรียกเพื่อจัดการกับฐานข้อมูล

การติดต่อระหว่างไคลเอนท์กับดาต้าเบสเซิร์ฟเวอร์ไม่ได้ติดต่อการโดยตรงแต่ทำการติดต่อผ่านทาง Open Database Connectivity (ODBC) ซึ่งให้ API method ในการติดต่อกับระบบจัดการฐานข้อมูล PostgreSQL ทำให้การติดต่อระหว่างไคลเอนท์กับดาต้าเบสเซิร์ฟเวอร์ทำได้โดยง่าย

8.2 แนวทางในการพัฒนาและแนวทางในการประยุกต์ใช้

จากการที่โปรแกรมประยุกต์ที่ได้ทำการพัฒนาโดยใช้โมเดลฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งสนับสนุนการเก็บข้อมูลที่มีความซับซ้อน เช่น Object อีกทั้งในการออกแบบโปรแกรมประยุกต์และออกแบบฐานข้อมูลได้พยายามออกแบบโดยแนวคิดเชิงวัตถุ การพัฒนาต่อจึงอาจทำได้โดยการพัฒนาสร้าง Componentใหม่ซึ่งประกอบไปด้วยเกตพื้นฐาน และสามารถทำการสร้าง Project ได้มากกว่าหนึ่ง Project พร้อมทั้งสามารถทำการบันทึกและโหลด Project ที่ต้องการได้

วัตถุประสงค์การใช้งานโปรแกรมประยุกต์นี้คือการพัฒนาเสมือนเป็นลอจิกเทรนเนอร์ ให้ผู้ที่ทำการออกแบบวงจรดิจิตอลสามารถนำมาสร้างและทดลองวงจรที่ต้องการจะสร้าง ดูผลเอาท์พุทในกรณีต่างๆซึ่งเมื่อโปรแกรมประยุกต์นี้เสร็จสามารถสามารถทำตามวัตถุประสงค์ดังกล่าวได้ และหากมีการพัฒนาต่อก็จะสามารถเป็นโปรแกรมที่ช่วยทำการจำลองการทำงานวงจรดิจิตอลได้อย่างสมบูรณ์

บรรณานุกรม

บทความ การศึกษาระบบฐานข้อมูลเชิงวัตถุสัมพันธ์, [Online].

Available: http://www.rtafa.ac.th

ดร.วิสุทธิ์ แซ่ตั้ง.2549.**Open Source DBMS: PostgreSQL**. สำนักพิมพ์ ส.ส.ท.

Chen, Y.S. 2007. **Object-Relational Concepts**. [Online].

Available : http://www.cs.nthu.edu.tw/~yishin/Courses/ISA6120/Handout/Session6%20-20Handout.pdf.

Elmasri, R. and Navathe, S.B. 2007. **Fundamentals of Database Systems.** 5th ed. Boston. Addison Wesley.

Memon, N. 2007. **Object-Relational Database Systems.** [Online].

Available : http://cs.aaue.dk/~**nasrullah**/F7S/db/Fall06/lecture_notes/week41_2b.pdf.

Oracle. 2007. **Oracle Database Application Developer's Guide – Object-Relational Feature 10g Release 2 (10.2)**. [Online].

Available : http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14260/toc.htm.

Stonebraker, M. and Brown, P. 1999. **Object-Relational DBMSs Tracking the Next Great Wave.** San Francisco : Morgan Kaufmann.

Manual PostgreSQL 8.2.4, [Online].

Available: http://www.postgresql.org/docs/manuals/

Abraham Silberschatz, Henry F. Korth, S. Sudarshan. **Database System Concepts**. McGraw-Hill

John C. Worsley and Joshua D. Drake.1999. **Practical PostgreSQL (O'Reilly Unix)**. Command Prompt Inc.

ภาคผนวก ก.

การติดตั้ง PostgreSQL for Windows

ระบบปฏิบัติการที่รองรับ

- Windows 2000 32 bit
- Windows XP 32 bit
- Windows 2003 32 bit

สำหรับระบบปฏิบัติการที่เป็น 64 bit ยังไม่ได้ทำการทดสอบ

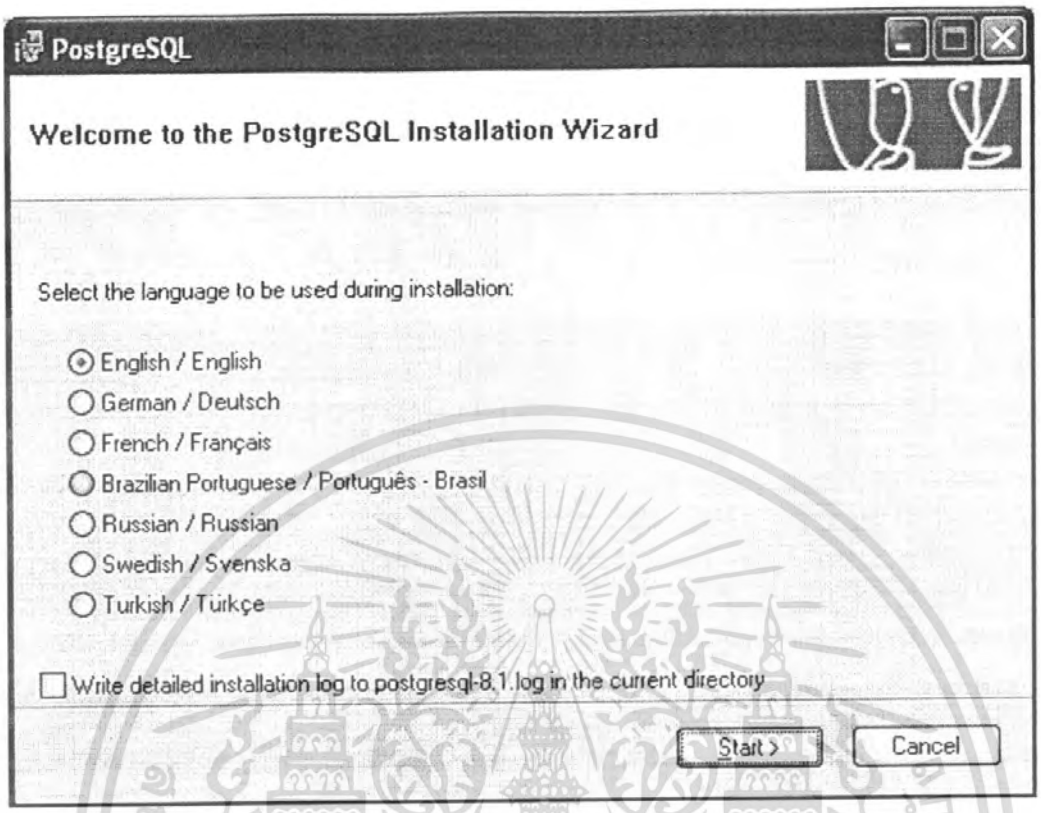
ขั้นตอนการติดตั้ง

ขั้นตอนการติดตั้งจะเป็นวิซาร์ด เหมือนกันโปรแกรมอื่นๆที่ติดตั้งภายใต้ระบบปฏิบัติการวินโดว์เวอร์ชั่นที่จะทำการติดตั้งเป็นเวอร์ชั่นล่าสุดขณะทำการทดสอบ คือเวอร์ชั่น 8.1.2 โดยไฟล์ติดตั้งหลังจากดาวน์โหลดมาแล้ว จะประกอบด้วย

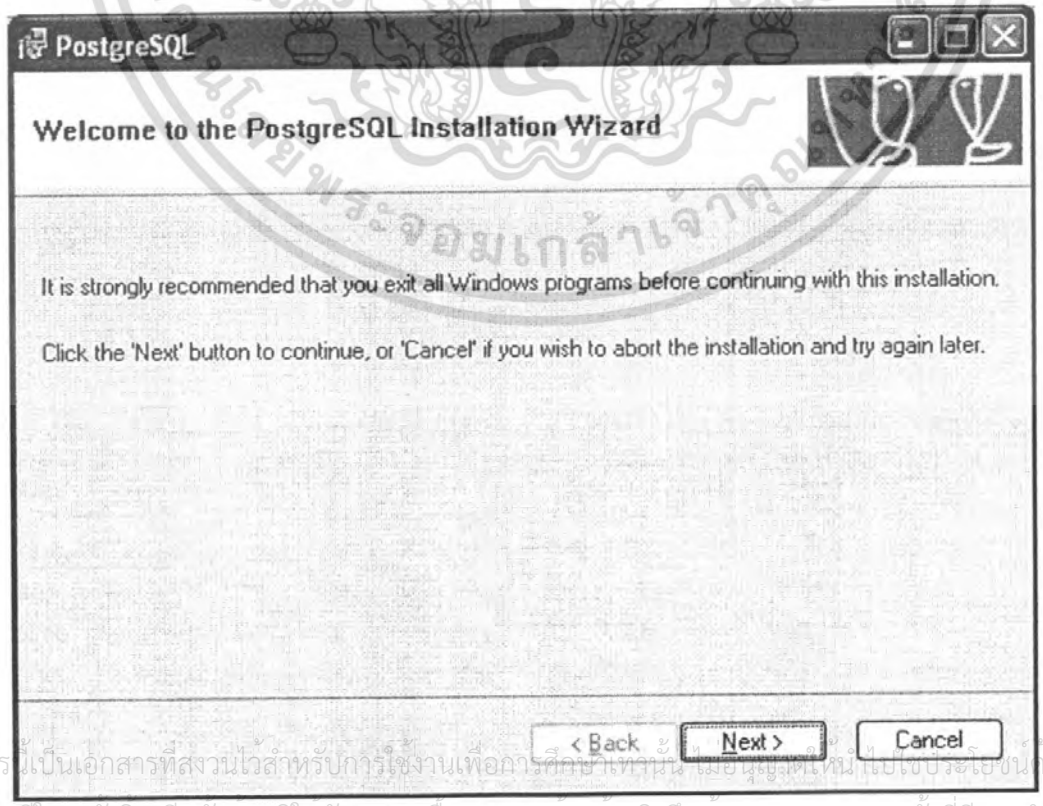
o postgresql-8.1-int.msi ไฟล์ข้อมูลติดตั้งหลัก
o postgresql-8.1.msi ไฟล์ติดตั้งให้สั่งรันไฟล์ตัวนี้หากต้องการติดตั้งใหม่
o readme.txt ไฟล์แนะนำ
o upgrade.bat ให้รันไฟล์นี้หากต้องการ upgrade จากเวอร์ชั่นเก่า

ในขั้นตอนนี้จะทำการติดตั้งใหม่โดยเครื่องที่ยังไม่เคยติดตั้ง PostgreSQL มาก่อน ให้ดับเบิ้ลคลิกที่ไฟล์ **postgresql-8.1.msi**

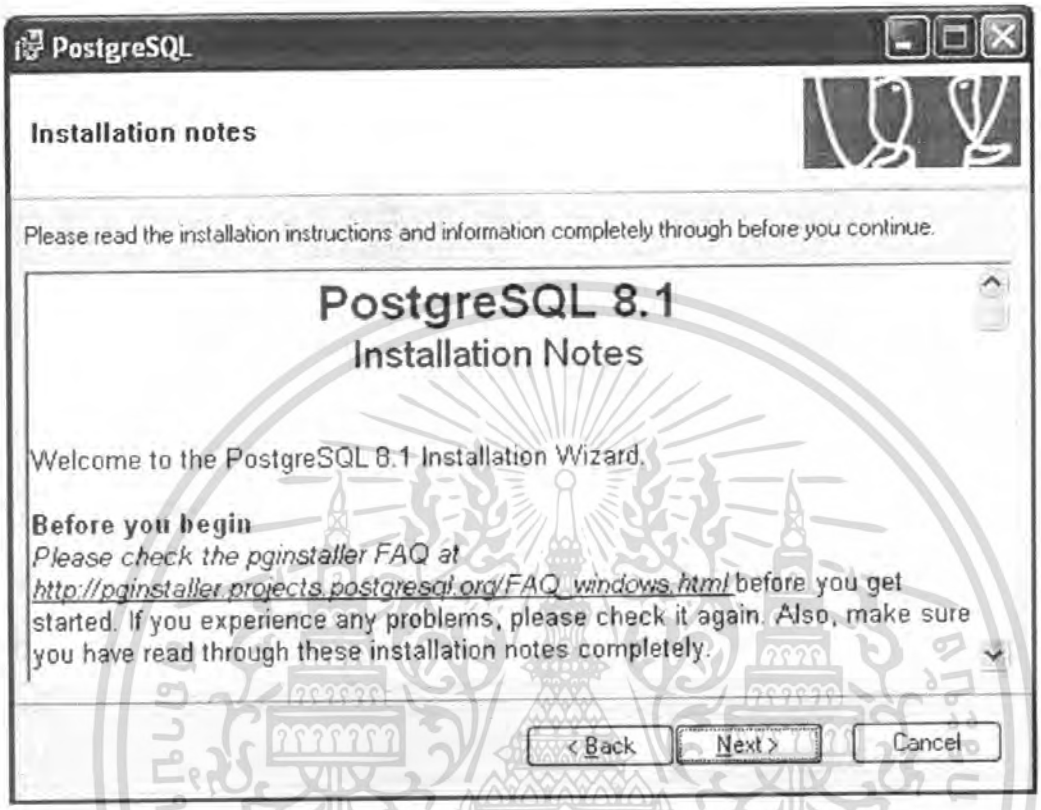
1. เลือกภาษา เป็นการเลือกภาษาที่วิซาร์ดใช้แสดงในการติดตั้งเท่านั้น ไม่เกี่ยวข้องกับภาษาที่ใช้แสดงในตัวโปรแกรมหลังการติดตั้ง ที่ด้านล่างของหน้าต่างผู้ติดตั้งสามารถเลือกเพื่อให้โปรแกรมติดตั้ง สร้างไฟล์เก็บข้อมูลการติดตั้งไว้ได้ รวมถึงข้อมูลและ password ของ service user และ database superuser เมื่อเลือกครบแล้วให้กด Start



2. หน้าต่างคำแนะนำเบื้องต้น แนะนำให้ปิดโปรแกรมอื่นก่อนทำการติดตั้ง เพื่อลดปัญหาการขัดแย้งกัน แนะนำให้ทำตาม หลังจากนั้นให้กด Next



3. หน้าต่างข้อความต้อนรับ และคำแนะนำ ให้อ่านข้อความแล้วกด Next

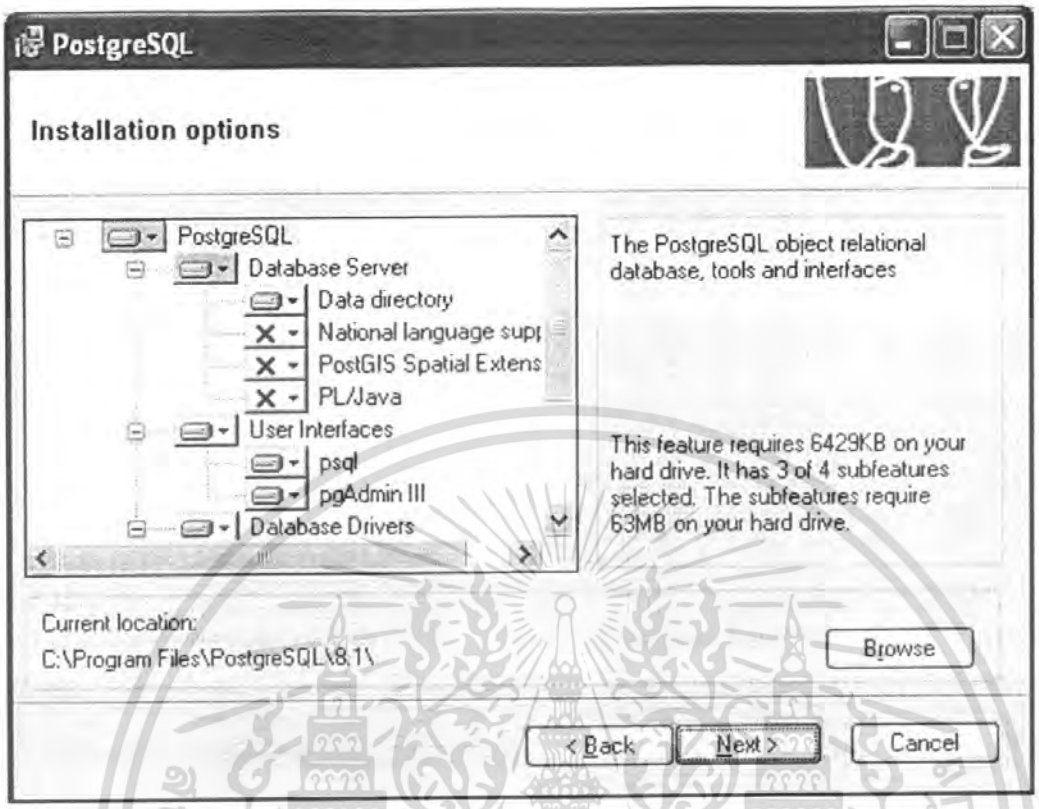


4. เลือกส่วนประกอบต่างๆของโปรแกรมที่จะทำการติดตั้ง

ส่วนประกอบของ Database Server สามารถติดตั้งได้เพียงบน NT Based Platforms (WindowsNT, 2000, XP และ 2003) เท่านั้น.

ในส่วนของ Data Directory ควรติดตั้งบนพาร์ติชั่นที่เป็น NTFS เท่านั้น เนื่องจาก PostgreSQL ต้องการใช้ลักษณะเฉพาะบางอย่างของระบบไฟล์ NTFS หากไปติดตั้งบนพาร์ติชั่นชนิดอื่น ผู้ติดตั้งจะต้องไปรันไฟล์ initdb.exe เพื่อทำการ initial database หลังทำการติดตั้งโปรแกรมเสร็จ

ผู้ติดตั้งสามารถเลือกตำแหน่งติดตั้งตัวโปรแกรมและส่วนประกอบได้โดยอิสระ โดยคลิกเลือกส่วนประกอบแล้วกด Browse เพื่อกำหนดตำแหน่งติดตั้ง หรือเลือกทั้งโปรแกรมให้ติดตั้งในที่เดียวกันโดยเลือกที่ PostgreSQL แล้วกด Browse เพื่อเลือกตำแหน่งติดตั้ง ในที่นี้เราจะใช้ค่าที่โปรแกรมกำหนดให้เพื่อไม่ให้ยุ่งยากในการจดจำ เสร็จแล้วกด Next เพื่อดำเนินการต่อไป



5. ติดตั้งให้ทำงานเป็นเซอร์วิส

หน้าต่างนี้จะเป็นการกำหนดให้ PostgreSQL ติดตั้งเป็น Service บน Windows ซึ่งจะมีข้อดีคือ ตัวโปรแกรมจะสามารถทำงานขึ้นมาได้โดยอัตโนมัติ และไม่ต้อง Logon Windows เพื่อเข้าไป Start โปรแกรม ทั้งนี้จะต้องมีการกำหนด Account บน Windows ที่จะใช้เพื่อ Start Service ด้วย กำหนดค่าต่างๆเสร็จแล้วกด Next เพื่อดำเนินการต่อไป

ในการกำหนด Account name "postgres" ถ้าเป็นการติดตั้งครั้งแรก ในเครื่องที่ใช้งานไม่มี user ชื่อ postgres มาก่อน จะมีหน้าต่างขึ้นมาให้ ยืนยันการเพิ่ม user ให้ตอบ **Yes** และจากนั้นจะมีหน้าต่างขึ้นมาแจ้งว่า รหัสผ่าน (password) ที่ตั้งขึ้นง่ายไปโปรแกรมจะสร้างรหัสผ่าน (password) ให้ใหม่ ให้ตอบ **No**

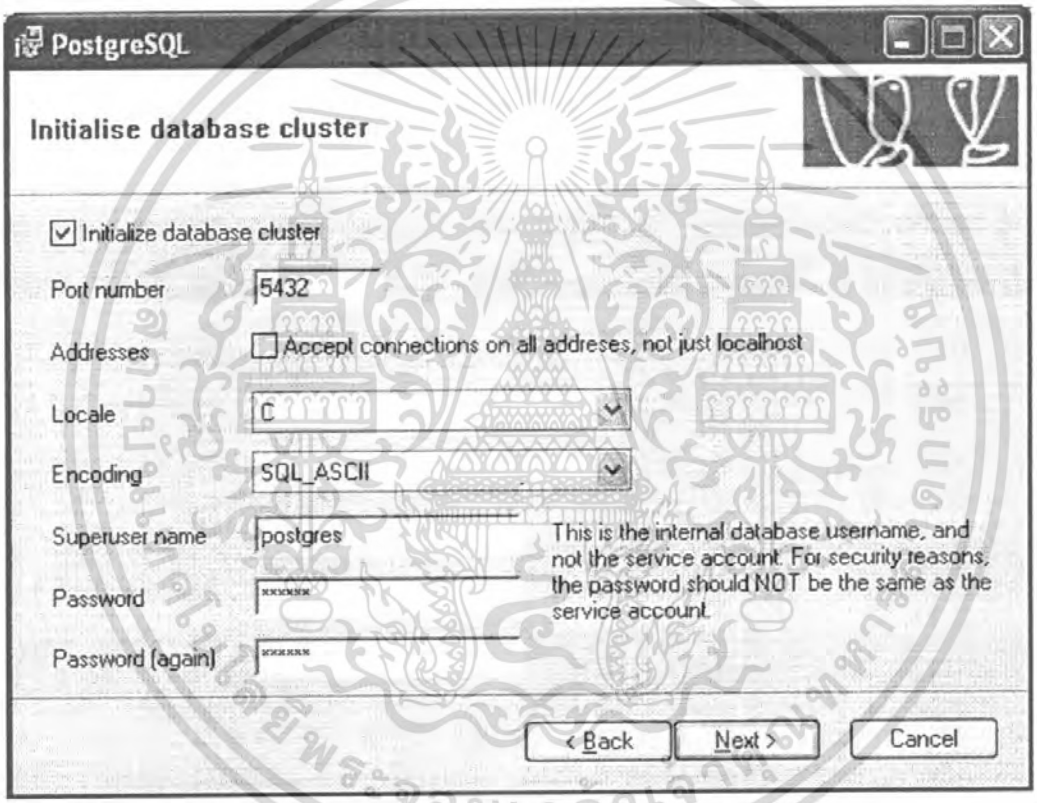
6. Initdb

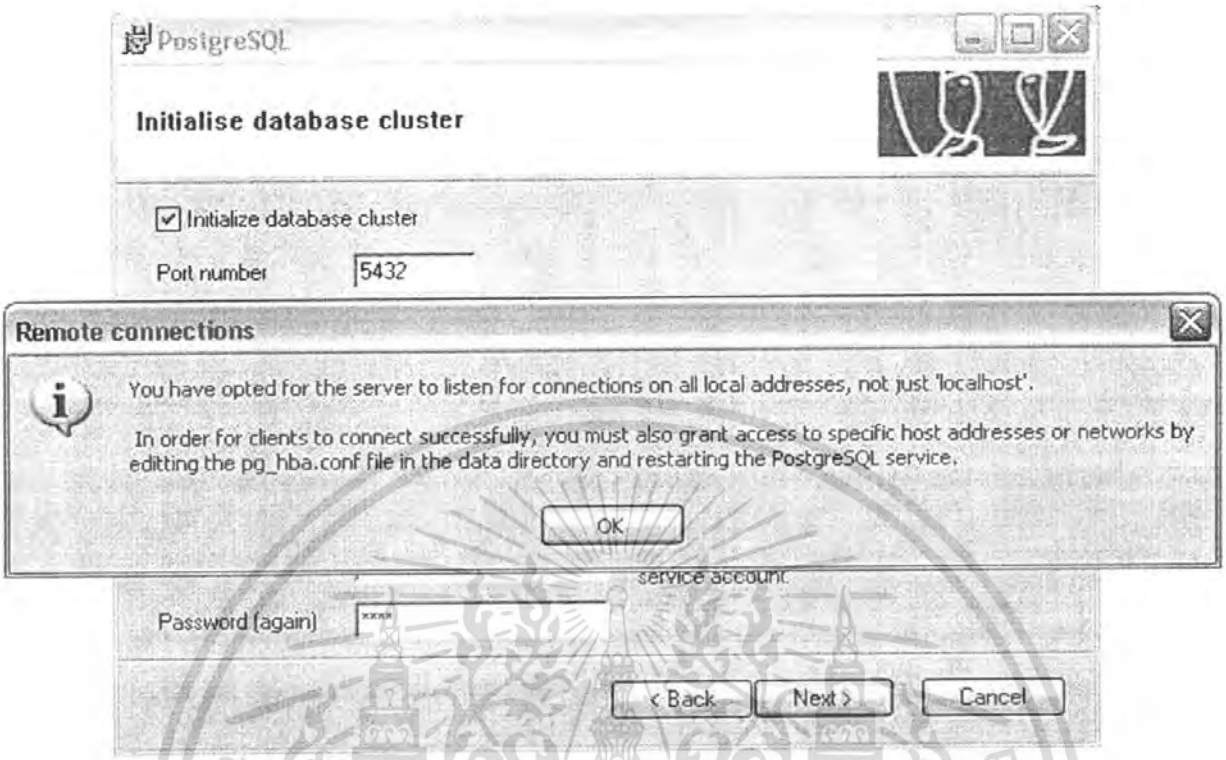
หน้าต่างนี้จะไม่แสดงขึ้นมาหากไม่ได้เลือกติดตั้งให้ทำงานเป็น service (ขั้นตอนที่ 6) ในขั้นตอนนี้เป็นการเลือกว่าต้องการจะสร้างพื้นที่เพื่อจัดเก็บฐานข้อมูล (database cluster) หรือไม่ ถ้าต้องการให้เลือก character set และ encoding และกำหนด ข้อมูลในการ login ค่าด้าเบสของ superuser สำหรับ Port number ที่จะให้ server ทำงานสามารถกำหนดเป็นหมายเลขอะไรก็ได้แต่ต้องไม่ซ้ำกับที่ windows เปิดใช้อยู่ และ option Accept

connections on all IP addresses, not just localhost จะเป็นการกำหนดให้เครื่อง client อื่นสามารถเข้าใช้งาน database บน serverได้

อย่างไรก็ตาม ผู้ติดตั้งจะยังต้องทำการเปิดอนุญาตให้เครื่อง Client เข้าใช้งาน server ด้วยตนเองในไฟล์ pg_hba.conf

ถ้าผู้ติดตั้งไม่เลือกที่จะทำการ Initialize database ในขั้นตอนนี้ ก็สามารถดำเนินการได้หลังจากติดตั้งโปรแกรมสำเร็จ โดย run ไฟล์ initdb.exe





7. กำหนดภาษาชุดคำสั่ง (Procedural languages)

เลือกภาษาชุดคำสั่ง(Procedural languages) ที่ต้องการใช้ใน template1 จริงๆแล้วทุกชุดภาษาคำสั่ง PL จะถูกติดตั้งลงไปทั้งหมด ในขั้นตอนนี้จึงเป็นเพียงการเปิดใช้งานว่าจะให้ทำงานด้วยชุด PL ชุดใดเป็นค่าเริ่มต้น

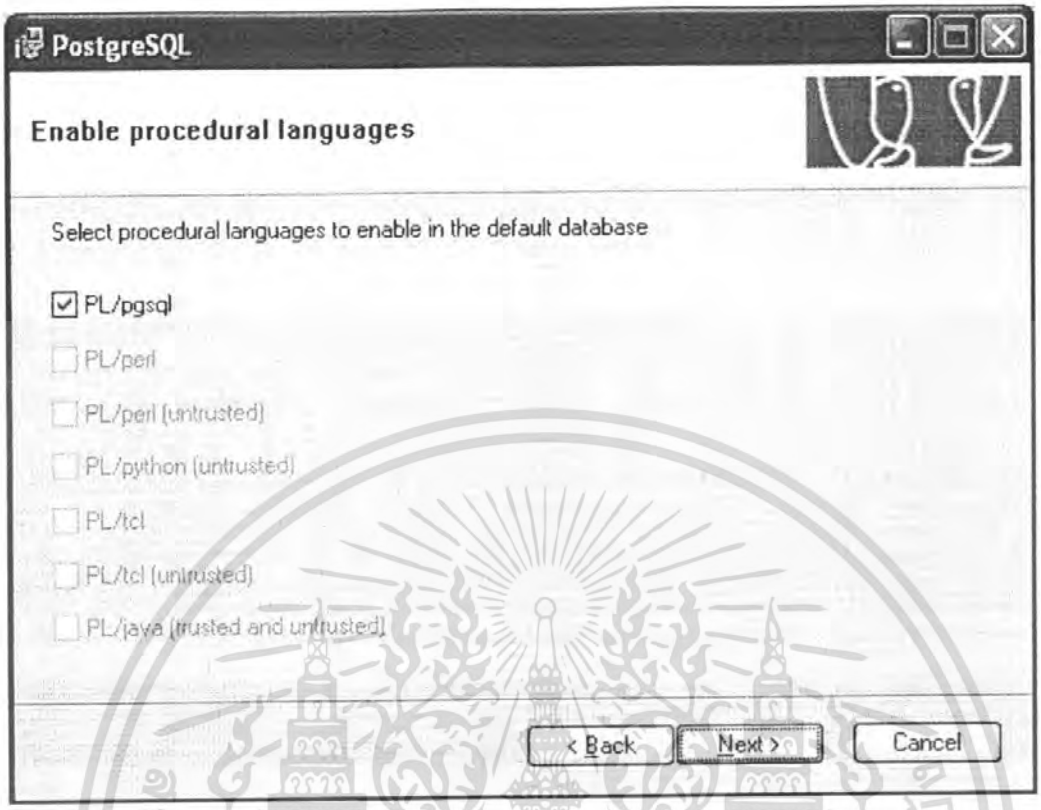
หน้าต่างนี้จะแสดงขึ้นมาเฉพาะที่มีการเลือกติดตั้ง PostgreSQL เป็น service และเลือกที่จะกำหนดค่าเริ่มต้นของ database cluster ผู้ติดตั้งสามารถเลือกได้เฉพาะภาษาที่มี runtime ที่ถูกต้องติดตั้งอยู่เท่านั้น ยกตัวอย่างเช่น

PL/Perl จะต้องมี ActiveState Perl 5.8 ติดตั้งอยู่

PL/python จะต้องมี Python 2.3 ติดตั้งอยู่

PL/tcl จะต้องมี ActiveState Tcl 8.4 ติดตั้งอยู่

PL/java จะต้องมี Sun Java Runtime Environment ติดตั้งอยู่



8. เลือก Contrib modules

เลือก contrib modules ที่จะใช้งานใน template1 ทุกโมดูลจะถูกติดตั้งตามปกติ แต่โมดูลที่ถูกเลือกในขั้นตอนนี้เท่านั้นจะถูกกำหนดเป็นค่าเริ่มต้นของฐานข้อมูล

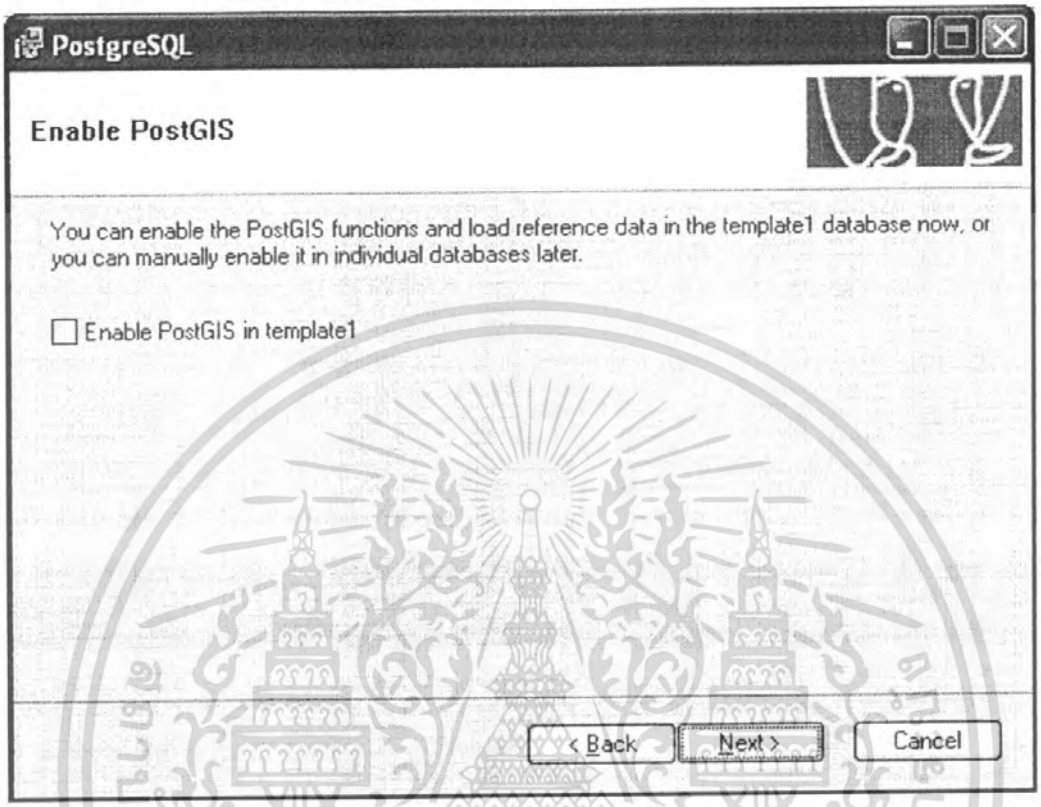
หน้าต่างนี้จะแสดงขึ้นมาเฉพาะที่มีการเลือกติดตั้ง PostgreSQL เป็น service และเลือกที่จะกำหนดค่าเริ่มต้นของ database cluster

ข้อสังเกต : โมดูล Admin81 จะถูกติดตั้งเป็นค่าเริ่มต้น เพราะว่า pgAdmin จำเป็นต้องใช้เพื่อให้เพิ่มเติมความสามารถในการให้บริการ เนื่องจากโมดูลนี้ถูกกำหนดใน template ดังนั้นจึงถูกติดเป็นค่าเริ่มให้กับฐานข้อมูลที่ถูกสร้างขึ้นมาใหม่ทุกตัว ถ้าผู้ติดตั้งไม่ต้องการสามารถเลือกไม่ติดตั้งได้ แต่จะไม่ได้ใช้บริการเพิ่มเติมที่โมดูลนี้มีอยู่เสร็จแล้วกด Next เพื่อดำเนินการต่อไป

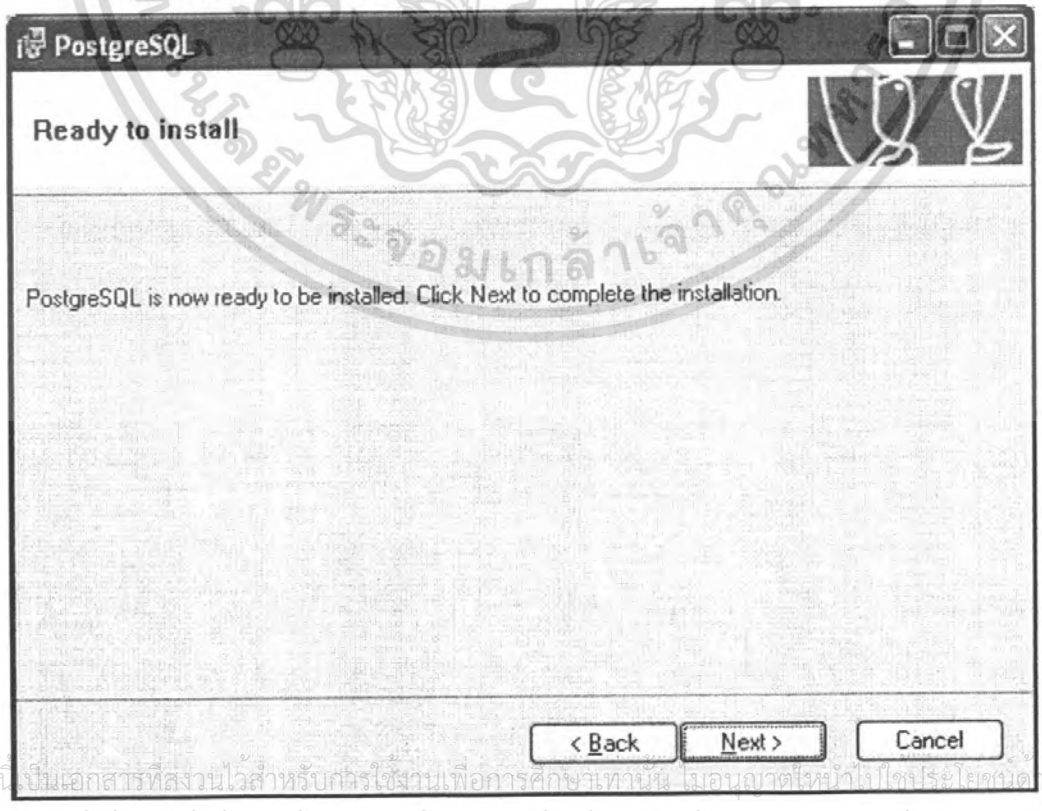
9. เปิดใช้งาน PostGIS

เลือกว่าจะเปิดใช้งาน PostGIS ใน template1 หรือไม่ มีผลกับทุกฐานข้อมูลที่ถูกสร้างขึ้นมาใหม่ ถ้าไม่เลือกตรงนี้แล้วต้องการใช้งานในภายหลัง สามารถไปเลือกใช้ในแต่

ละฐานข้อมูลได้ในภายหลัง เสร็จแล้วกด Next เพื่อดำเนินการต่อไป หน้าต่างนี้จะแสดงขึ้นมาเฉพาะที่มีการเลือกติดตั้ง PostGIS Spatial Extensions ในขั้นตอนที่ 4



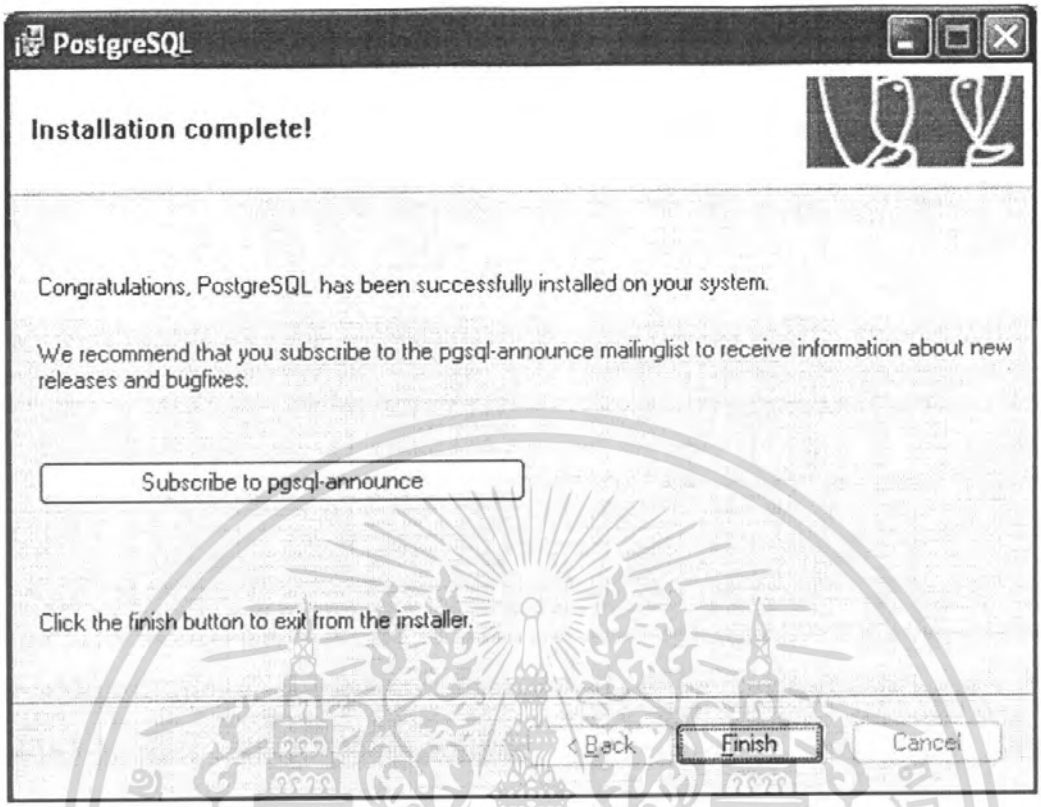
10. คลิก Next เพื่อดำเนินการติดตั้ง



11. แสดงความก้าวหน้าในการติดตั้ง

จะแสดงแถบความคืบหน้าในการติดตั้ง ใน windows บางเวอร์ชั่นเช่น Windows XP รุ่นก่อน Service Pack2 และ Windows 2003 รุ่นก่อน Service Pack1 จะมีหน้าต่าง Command Prompt เปิดขึ้นมาในขั้นตอน "Initializing database cluster" อย่าปิดหน้าต่างนี้ ให้ปล่อยทิ้งไว้ ซึ่งมันจะถูกปิดไปเองหลังจากการติดตั้งเสร็จสิ้น

12. จบการติดตั้ง ผู้ติดตั้งสามารถไปสมัครเป็นสมาชิกเพื่อรับข่าวสารความเคลื่อนไหวเกี่ยวกับ หลังจากนี้ถ้าผู้ติดตั้งต้องการเพิ่มหรือถอดถอนความสามารถบางอย่างให้ใช้ Add/remove programs จาก Control Panel.



ภาคผนวก ข.

การติดตั้ง PostgreSQL บนเซิร์ฟเวอร์ลินุกซ์

เนื้อหาส่วนนี้จะอธิบายถึงวิธีการติดตั้ง PostgreSQL บนเครื่องเซิร์ฟเวอร์ที่ได้ติดตั้งระบบปฏิบัติการลินุกซ์ Fedora Core 4 ไว้เรียบร้อยแล้วเท่านั้น

การติดตั้ง PostgreSQL บนเซิร์ฟเวอร์ลินุกซ์ (Fedora Core 4)

เนื้อหาส่วนนี้จะอธิบายถึงวิธีการติดตั้ง PostgreSQL บนเครื่องเซิร์ฟเวอร์ที่ได้ติดตั้งระบบปฏิบัติการลินุกซ์ Fedora Core 4 ไว้เรียบร้อยแล้วเท่านั้น

ระบบปฏิบัติการที่รองรับ

- Fedora Core 4 (32 บิต และ 64 บิต)

ขั้นตอนการติดตั้ง

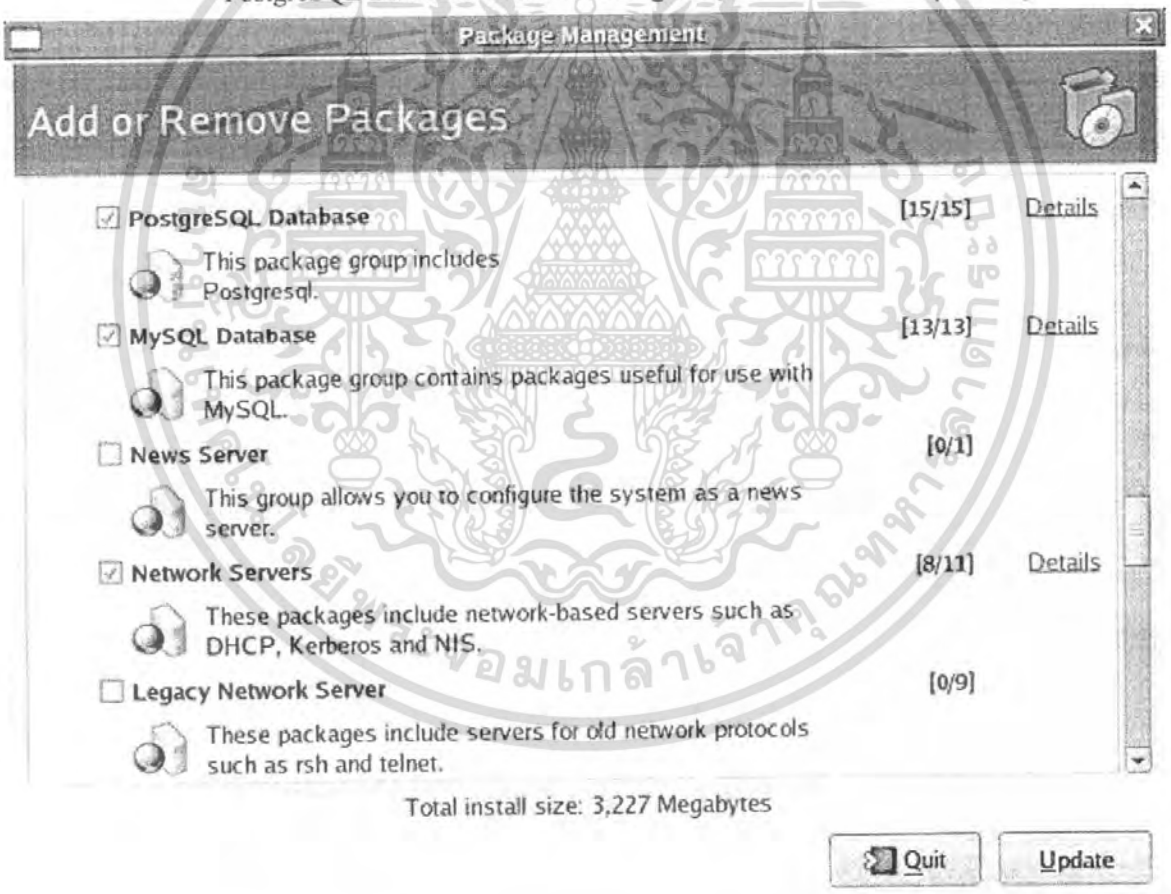
วิธีการติดตั้งจะแบ่งเป็น 2 วิธีการ วิธีการติดตั้งในโหมดวินโดว์ (X-Window) เหมาะสำหรับเซิร์ฟเวอร์ลินุกซ์ที่สามารถใช้งานโหมดวินโดว์ (X-Window) ได้ และวิธีการติดตั้งโหมดคำสั่ง (Command Line) เหมาะสำหรับเซิร์ฟเวอร์ลินุกซ์ที่ไม่สามารถใช้งานโหมดวินโดว์ได้

กรณีติดตั้งในโหมดวินโดว์ (X-Window)

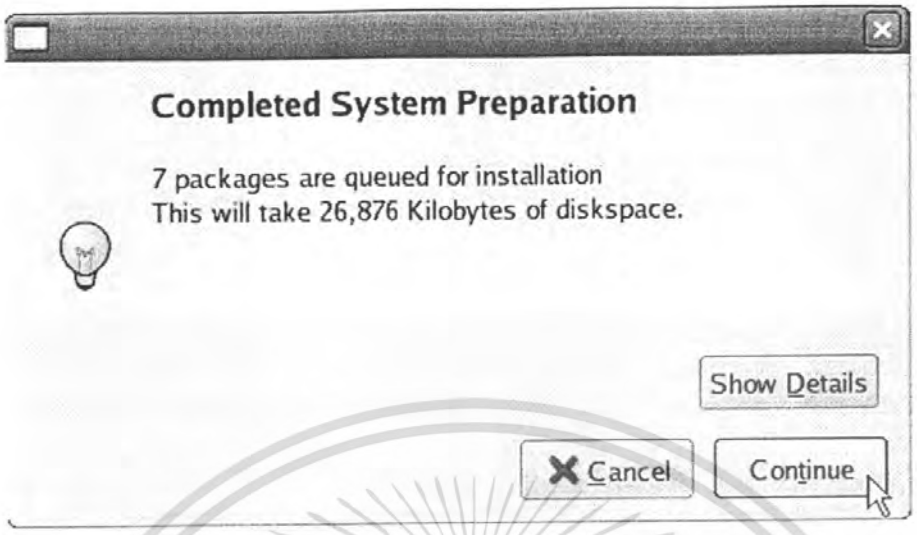
การติดตั้งในโหมดวินโดว์ (X-Window) เป็นวิธีที่สะดวกที่สุด โดยทั่วไปจะเป็นการสั่งงานผ่านการคลิกเมนูต่างๆ ดังขั้นตอนต่อไปนี้

1. หน้าจอเดสก์ทอปของวินโดว์ คลิกเลือกเมนู Desktop > System Settings > Add/Remove Applications ดังรูป

2. หน้าต่าง Package Management แสดงแพ็กเกจที่ติดตั้งไว้แล้ว และสามารถเลือกแพ็กเกจเพิ่มเติมได้ด้วย ในที่นี้ให้คลิกทำเครื่องหมายถูก (/) หน้ากลุ่มโปรแกรม PostgreSQL Database เพื่อติดตั้ง PostgreSQL บนเซิร์ฟเวอร์ลินุกซ์ ดังรูป

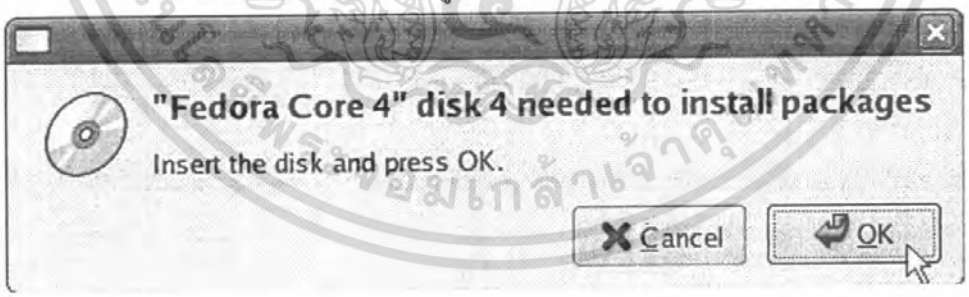


3. หน้าต่าง Completed System Preparation จะสรุปจำนวนแพ็กเกจที่จะติดตั้ง ให้คลิก Continue เพื่อดำเนินการต่อ ดังรูป

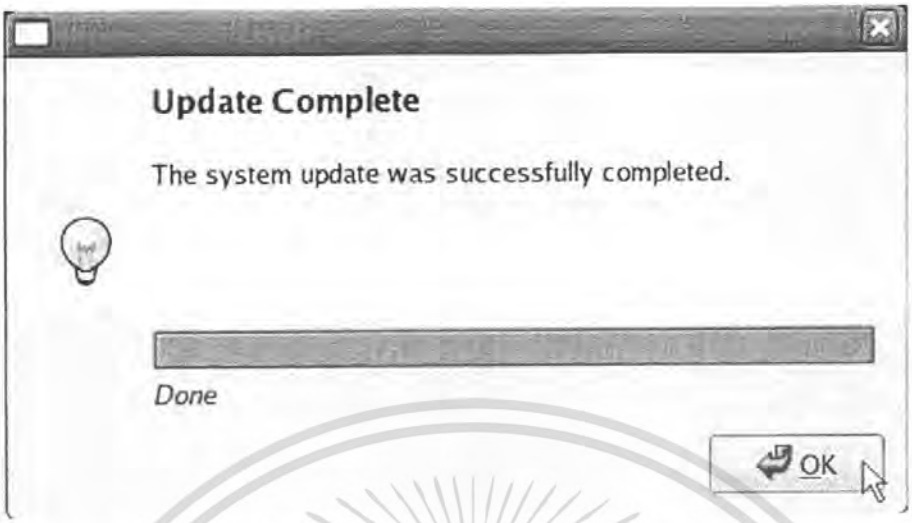


4. ลินุกซ์จะเรียกใช้แผ่นซีดีชุดติดตั้งลินุกซ์แผ่น # 1 ให้นำแผ่น CD ดังกล่าวเข้าในไดรว์ซีดี แล้วคลิก OK ดังรูป

5. ลินุกซ์จะเรียกใช้แผ่นซีดีชุดติดตั้งลินุกซ์แผ่น # 4 ให้นำแผ่น CD ดังกล่าวเข้าในไดรว์ซีดี แล้วคลิก OK ดังรูป



6. เมื่อปรากฏหน้าต่าง Update Complete ดังรูป (6) แสดงว่าติดตั้ง PostgreSQL เรียบร้อยแล้ว



กรณีติดตั้งในโหมดคำสั่ง (Command Line)

การติดตั้งวิธีการนี้จะเป็นการสั่งงานผ่านโปรแกรมยูทิลิตี้ Red Hat Package Manager (rpm) โดยจะเรียกใช้ไฟล์แพ็กเกจ rpm ของ PostgreSQL จากแผ่นติดตั้งลินุกซ์ วิธีการนี้เหมาะสำหรับการติดตั้งบนเซิร์ฟเวอร์ลินุกซ์ที่ไม่สามารถใช้งานโหมดวินโดว์ (X-Window) เช่น Secure Shell (SSH) เป็นต้น ขั้นตอนการดำเนินการก็มีดังต่อไปนี้

1. ก่อนการติดตั้งให้ตรวจสอบก่อนว่ามีการติดตั้ง PostgreSQL ไว้บนเซิร์ฟเวอร์หรือไม่ หากติดตั้งไว้ หากต้องการติดตั้งเวอร์ชั่นที่ใหม่กว่า ต้องลบเวอร์ชั่นเก่าออกก่อน หรือ ติดตั้งแบบอัพเกรดก็ได้ # rpm -qa | grep -I ^postgres postgresql-8.0.3-1 | postgresql-libs-8.0.3-1 |---> รายชื่อแพ็กเกจ PostgreSQL ที่มีการติดตั้งไว้แล้ว (ถ้ามี) postgresql-server-8.0.3-1 | หากมีการติดตั้ง PostgreSQL ไว้แล้ว จะปรากฏรายการชื่อแพ็กเกจแสดงขึ้นมา หากไม่มีรายการชื่อแพ็กเกจแสดงขึ้นมา ก็แสดงว่าไม่มี PostgreSQL ติดตั้งในเซิร์ฟเวอร์
2. นำแผ่นซีดีลินุกซ์ **แผ่น # 1** เข้าไดรว์ซีดี แล้วเรียกใช้อุปกรณ์ซีดีโดยการ mount # mount /media/cdrecorder
3. ติดตั้งแพ็กเกจไฟล์ rpm postgresql-libs-8.0.3-1 ซึ่งเป็นชุดไฟล์ไลบรารี่ของ PostgreSQL **(ซีดีลินุกซ์ แผ่น # 1)** # rpm -i /media/cdrecorder/Fedora/RPMS/postgresql-libs-8.0.3-1.i386.rpm
4. เลิกใช้อุปกรณ์ซีดีโดยการ unmount # umount /media/cdrecorder
5. นำแผ่นซีดีลินุกซ์ **แผ่น # 4** เข้าไดรว์ซีดี แล้วเรียกใช้อุปกรณ์ซีดีโดยการ mount # mount /media/cdrecorder

6. ติดตั้งแพ็กเกจไฟล์ rpm postgresql-8.0.3-1 ซึ่งเป็นชุดไฟล์สำหรับโปรแกรมไคลเอนต์และไลบรารี่ต่างๆ **(ซีดีลินุกซ์ แผ่น # 4)** # rpm -i /media/cdrecorder/Fedora/RPMS/postgresql-8.0.3-1.i386.rpm
7. ติดตั้งแพ็กเกจไฟล์ rpm postgresql-server-8.0.3-1 ซึ่งเป็นชุดไฟล์สำหรับติดตั้ง PostgreSQL เป็นเซิร์ฟเวอร์ **(ซีดีลินุกซ์ แผ่น # 4)** # rpm -i /media/cdrecorder/Fedora/RPMS/postgresql-server-8.0.3-1.i386.rpm
8. เลิกใช้อุปกรณ์ซีดีโดยการ unmount ก็เป็นอันเสร็จขั้นตอนการติดตั้ง PostgreSQL # umount /media/cdrecorder

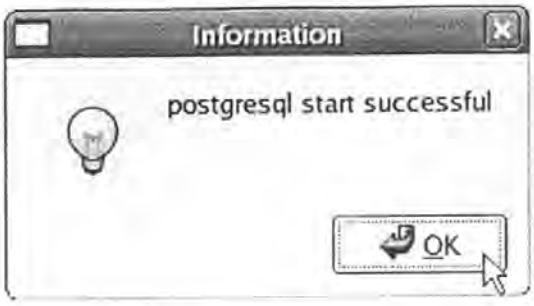
กำหนดการทำงานของเซอร์วิส PostgreSQL

ขั้นตอนต่อไปนี้จะเป็นการกำหนดให้ PostgreSQL ทำงานแบบเซอร์วิส ซึ่งจะเริ่มต้นทำงานโดยอัตโนมัติทุกครั้งที่มีการเปิดเซิร์ฟเวอร์ลินุกซ์ สามารถกำหนดได้ทั้งในโหมดวินโดว์และโหมดคำสั่ง

ขั้นตอนสำหรับโหมดวินโดว์ (X-Window)

1. กำหนดให้ PostgreSQL เริ่มต้นทำงานทุกครั้งที่มีการเปิดเครื่องเซิร์ฟเวอร์ลินุกซ์ โดยการกำหนดผ่านเซอร์วิสของ PostgreSQL ให้คลิกไปที่เมนู Desktop > System Settings > Server Settings > Services ดังรูป

2. คลิกปุ่ม Start ดังรูป (8) เพื่อสั่งให้เซอร์วิสของ PostgreSQL ทำงานในครั้งนี้ทันที หลังจากเซอร์วิสเริ่มต้นทำงานจะปรากฏข้อความ ดังรูป (9) ก็เป็นอันจบขั้นตอนการติดตั้ง PostgreSQL



3. หากต้องการหยุดการทำงาน หรือกำหนดให้เซอร์วิส ของ PostgreSQL ไม่ทำงานเมื่อเปิดเซิร์ฟเวอร์ ก็สามารถทำได้เช่นเดียวกัน โดยการคลิกปุ่ม Stop และ คลิกเอาเครื่องหมายถูก (/) หน้าชื่อเซอร์วิส postgresql ออกไป หลังจากนั้นให้คลิกปุ่ม Save

ขั้นตอนสำหรับโหมดคำสั่ง (Command Line)

1. กำหนดให้ **เพิ่มเซอร์วิส** ของ PostgreSQL เข้าในรายการของเซิร์ฟเวอร์ลินุกซ์ # chkconfig --add postgresql

2. กำหนดให้เซอร์วิสของ PostgreSQL **ทำงาน** ทุกครั้งที่เปิดเซิร์ฟเวอร์ลินุกซ์ # chkconfig postgresql on

3. หากต้องการ **เริ่มต้นการทำงาน** เซอร์วิสของ PostgreSQL ในขณะใดขณะหนึ่งทันที ให้ส่งค่าพารามิเตอร์ **start** ให้เซอร์วิส # service postgresql start