แบบจำลองการใช้ภาษา SQL

SQL GENERATOR

ณัฐพร นิมมานสถิตย์
มาศินีย์ มานะวรชัยสกุล
สุโรจนา โพธิฆานนนท์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

SQL GENERATOR

NATTAPORN NIMMANSATIT
MASINEE MANAVORACHAISKUL
SUROJJANA PHOTHICHARNNONT

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHEMETICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2000

หัวข้อปัญหาพิเศษ	แบบจำลองการใช้ภาษา SQL	
ชื่อนักศึกษา	นางสาวณัฐพร นิมมานสถิตย์	40056026
	นางสาวมาศินีย์ มานะวรชัยสกุล	40056061
	นางสาวสุโรจนา โพธิฌานนนท์	40056105
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2543

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ดร. กิตติมา เมฆาบัญชากิจ	
กรรมการ	รองศาสตราจารย์ภัคคินี ชิตสกุล	
กรรมการและอาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปัญหาพิเศษ	แบบจำลองการใช้ภาษา SQL	
ชื่อนักศึกษา	นางสาวณัฐพร นิมมานสถิตย์	40056026
	นางสาวมาศินีย์ มานะวรชัยสกุล	40056061
	นางสาวสุโรจนา โพธิฌานนนท์	40056105
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2543	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

บทคัดย่อ

เนื่องจากโลกยุคปัจจุบันเป็นยุคของข้อมูลข่าวสาร ซึ่งการที่จะใช้งานข้อมูลข่าวสารได้อย่างมีประสิทธิภาพสูงสุด ย่อมหลีกเลี่ยงจากการใช้งานคอมพิวเตอร์ไม่ได้อย่างแน่นอน จึงได้มีการนำข้อมูลในองค์กรที่มีความเกี่ยวข้องกันมารวมไว้อย่างเป็นระบบในที่เดียวกัน ที่เรียกกันว่าระบบฐานข้อมูล โดยที่ผู้ใช้ฐานข้อมูลแต่ละคนจะมองข้อมูลนี้ในแง่มุม หรือวิวที่แตกต่างกันไปตามจุดประสงค์ของการประยุกต์ใช้งาน และมุ่งหวังให้การเรียกใช้ข้อมูลในงานของเขามีประสิทธิภาพมากที่สุด จึงได้มีภาษาที่ถูกออกแบบมาโดยเฉพาะ เพื่อใช้กับงานการจัดการข้อมูลที่อยู่ในฐานข้อมูล ซึ่งภาษาในการเรียกใช้ข้อมูลที่ได้รับความนิยมสูงสุดในปัจจุบันคือ ภาษา SQL (Structured Query Language)

ดังนั้นแบบจำลองการใช้ภาษา SQL จึงได้ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้แก่ผู้ใช้ในการเขียนภาษา SQL โดยที่ไม่ต้องทำการเขียนภาษา SQL เองทั้งหมด เป็นเครื่องมือในการช่วยเขียนภาษา SQL ที่มีรูปแบบการใช้งานที่ง่าย รวมถึงผู้ใช้งานสามารถใช้งานได้โดยผ่านทางอินเทอร์เน็ต

Special Project Title	SQL Generator	
Students	Miss. Nattaporn Nimmansatit	40056026
	Miss. Masinee Manavorachaiskul	40056061
	Miss. Surojjana Phothicharnnont	40056105
Degree	Bachelor's Degree of Science	
Department	Mathematics and Computer Sciences , Faculty of Science	
Programme	Computer Sciences	
Academic Year	2000	
Special Project Advisor	Lecturer Wisan Tangwongcharoen	

ABSTRACT

Because of the world today , also called the age of information technology demanded the most efficient way to managed information as well as possible . Therefore computers had became the most essential tools since . As a consequence , the information that concerned by any organization , was systematically gather up together which was called database system . Database system could be used by different users and different tasks depend on how the user wanted . Because of this , a particular computer language was designed in order to increase the efficiency of database system using . The most popular language today is SQL (Structured Query Language) .

The main objective of the SQL GENERATOR was developed to help SQL developer as a assistant tool with a easy to use interfaces and works through the internet .

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องแบบจำลองการใช้ภาษา SQL สามารถสำเร็จลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์วิสันต์ ตั้งวงษ์เจริญ อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาในการแก้ปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษนี้สำเร็จด้วยดี รวมทั้งเพื่อนๆ ทุกคนที่ให้คำปรึกษาและความช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2544

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหาพิเศษ

เนื่องจากในปัจจุบันการติดต่อกับ Database ส่วนใหญ่จะใช้ ภาษา SQL (Structure Query Language) เป็นตัวติดต่อ แต่เนื่องจากผู้ใช้บางคนอาจไม่มีความชำนาญในการเขียนคำสั่งต่าง ๆ หรือมีความสับสนในการเขียนฟังก์ชั่นพิเศษ และต้องการให้ผู้ใช้ได้รับความสะดวกในการใช้มากขึ้น โดยที่ไม่จำเป็นต้องทำการเขียนภาษาSQL ทั้งหมด จึงได้มีการจัดทำเครื่องมือในการช่วยเขียนภาษา SQL ขึ้นมา

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษาปัญหาพิเศษ

1.2.1 เพื่อต้องการให้ผู้ที่ไม่มีความชำนาญในการใช้ภาษา SQL สามารถเข้าใจการทำงานได้มากขึ้น และสามารถใช้งานได้อย่างมีประสิทธิภาพ ได้รับประโยชน์จากภาษา SQL มากที่สุด

1.2.2 เพื่อต้องการให้ผู้ที่มีความสนใจในภาษา SQL สามารถศึกษาถึงการใช้งานได้ และมีความเข้าใจใน syntax ต่าง ๆ

1.2.3 เพื่อช่วยลดการจดจำฟังก์ชันพิเศษ และการพิมพ์คำสั่งต่าง ๆ ที่อาจเกิดการผิดพลาดได้

1.2.4 เพื่อต้องการให้ผู้ใช้งานสามารถเข้าถึงฐานข้อมูลจากที่ใดก็ได้ โดยผ่านทางอินเทอร์เน็ต

1.2.5 เพื่อศึกษาเทคโนโลยีของ Java Servlet ซึ่งนำมาใช้ในการติดต่อกับฐานข้อมูล

1.3 สมมติฐานของการศึกษาปัญหาพิเศษ

เป็นการต้องการให้ผู้ที่ต้องการติดต่อกับ Database ที่ใช้ภาษา SQL มีความสะดวกในการใช้งาน และได้รับประโยชน์อย่างสูงสุด รวมถึงให้ผู้สนใจที่ต้องการศึกษาสามารถทำความเข้าใจกับภาษา SQL ได้ง่ายขึ้น โดยช่วยลดการจดจำฟังก์ชันพิเศษต่าง ๆ

1.4 ขอบเขตของปัญหาพิเศษ

เป็นการเขียนเครื่องมือในการช่วยให้สามารถติดต่อกับ Database ได้ โดยใช้คำสั่ง SQL เป็นตัวติดต่อ โดยที่ประกอบไปด้วยฟังก์ชันต่าง ๆ และมีตารางให้ทำการเลือกได้ โดยช่วยลดปัญหาด้านการจดจำคำสั่ง ซึ่งสามารถ ทำการศึกษา และติดต่อกับ Database โดยผ่านระบบอินเทอร์เน็ตได้

1.5 ขั้นตอนในการศึกษาและดำเนินงาน

1.5.1 ศึกษารวบรวมข้อมูล

1.5.1.1 ศึกษาหลักการ และทฤษฎีต่าง ๆ ที่ใช้ในการพัฒนาระบบ

1.5.1.2 ศึกษาและฝึกการใช้ software และเขียนโปรแกรม

1.5.2 ออกแบบระบบและฐานข้อมูล

1.5.2.1 ออกแบบฐานข้อมูล (Data Model Design)

1.5.2.2 ออกแบบหน้าจอส่วนที่ติดต่อกับผู้ใช้ (User Interface Design)

1.5.3 พัฒนาระบบ

1.5.3.1 สร้างตารางฐานข้อมูล (Database Table Implementation)

1.5.3.2 สร้างหน้าจอ Web (Web Interface Implementation)

1.5.3.3 เขียนโปรแกรมติดต่อระหว่างหน้าจอกับฐานข้อมูล (Web Database programming)

1.5.4 ทดสอบและแก้ไขระบบ

1.5.5 จัดทำเอกสาร

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับการพัฒนาระบบ

2.1 ภาษา SQL : Structured Query Language

คือภาษากลางที่ใช้ในการติดต่อกับฐานข้อมูลเชิงสัมพันธ์ (Relational Database) คุณสมบัติสำคัญของภาษานี้ คือสามารถใช้เรียกดูข้อมูลตามที่เราระบุในความต้องการได้ สามารถใช้ในการเปลี่ยนแปลงเพิ่มเติม และลบข้อมูลออกจากระบบ นอกจากนี้ยังใช้ในการสร้างระบบฐานข้อมูลด้วย

จริง ๆ แล้ว SQL เป็นกึ่ง ๆ ภาษา คือถ้าเรานำ SQL ไปเทียบกับ Programming language ทั่วไปแล้วจะพบว่ามีจุดแตกต่างกันดังนี้

- SQL จะจัดการข้อมูลทีละกลุ่ม ๆ (เซ็ต) ไม่ใช่ทีละรายการ
- แต่ละคำสั่ง SQL เป็นอิสระแก่กัน และสมบูรณ์ในตัวมันเอง
- แต่ละประโยคใน SQL จะมุ่งเน้นไปที่ว่าคุณต้องการทำอะไรกับข้อมูล (What) เช่น ปรับปรุงข้อมูลในตาราง ... โดยมีเงื่อนไขต่อไปนี้... โดยไม่ต้องไปคำนึงว่ามันทำงานอย่างไร ดึงข้อมูลขึ้นมาด้วยวิธีไหน แก้ไขกลับไปอย่างไร (How) เรื่องเหล่านี้เป็นหน้าที่ของระบบฐานข้อมูล (RDBMS) ดูแล
- SQL ไม่มี Flow-control statement อย่างที่พบกันใน Programming language ทั่ว ๆ ไป เช่น Loopstructure

2.1.1 โครงสร้างทางไวยากรณ์ภาษา SQL : SQL Grammar

โครงสร้างทางไวยากรณ์ภาษา SQL มีลักษณะพิเศษจากภาษาอื่นๆ ดังต่อไปนี้

- ทุกคำสั่ง SQL จะต้องปิดท้ายด้วย ; เสมอ
- คำสั่ง SQL คำสั่งหนึ่ง ๆ สามารถเขียนแยกเป็นหลาย ๆ บรรทัดได้ และก็แนะนำให้แยกออกเป็นหลาย ๆ บรรทัดเพื่อให้อ่านได้ง่าย
- ภาษา SQL ตัวใหญ่หรือตัวเล็กถือว่าเหมือนกัน ไม่ว่าจะเป็นคียเวริด์ เช่น CODE หรือ SELECT , ชื่อตาราง , ชื่อ Column แต่ค่าใน Column ตัวใหญ่ตัวเล็กแตกต่างกัน เช่น สมมติว่าเราเก็บชื่อสินค้าเป็นภาษาอังกฤษเป็นตัวพิมพ์ใหญ่ แล้วเราหาสินค้าที่ชื่อขึ้นต้นด้วยตัว a เราก็จะไม่เจอข้อมูลที่ต้องการ

2.1.2 รูปแบบคำสั่งภาษา SQL : SQL Command

รูปแบบคำสั่งแบ่งออกเป็นกลุ่มๆได้ ดังนี้

- Data Manipulation Language (DML) เป็นคำสั่งที่ใช้ในการเรียกดู , เพิ่ม , แก้ไข หรือลบข้อมูล SELECT , INSERT , UPDATE , DELETE
- Data Definition Language (DDL) เป็นคำสั่งในการสร้าง , แก้ไข , ลบ Schema object ต่าง ๆ ได้แก่ CREATE , DROP , ALTER , TRUNCATE และ คำสั่งกำหนดสิทธิ์ในการเข้าถึงข้อมูล เช่น GRANT , REVOKE

2.1.2.1 **การเรียกดูข้อมูล**

ประโยคคำสั่งสำหรับการเรียกดูข้อมูลได้แก่ คำสั่ง SELECT ซึ่งมีรูปแบบของคำสั่งดังนี้

SELECT... FROM... WHERE... GROUP BY... HAVING... ORDER BY...

ซึ่งผู้ใช้สามารถระบุส่วนต่างๆในคำสั่งได้ดังนี้

ส่วนหลัง SELECT ใช้ระบุ คอลัมน์หรือกลุ่มของคอลัมน์ที่เราต้องการดูข้อมูล

ส่วนหลัง FROM ใช้ระบุ ชื่อของตารางที่เราต้องการ

ส่วนหลัง WHERE ใช้ระบุ เงื่อนไขของข้อมูลที่เราสนใจ ซึ่งอาจจะมีหรือไม่มีก็ได้

ส่วนหลัง GROUP BY ใช้ระบุ คอลัมน์เป็นการจับกลุ่มข้อมูลที่เราสนใจเข้าไว้ด้วยกัน ซึ่งจะมีหรือไม่มีก็ได้

ส่วนหลัง HAVING ใช้ระบุ เงื่อนไขที่เป็นฟังก์ชันต่างๆเปรียบเสมือนกับการใช้ WHERE สำหรับกรณีที่ มีการจัดกลุ่มนั่นเอง ซึ่งอาจจะมีหรือไม่มีก็ได้

ส่วนหลัง ORDER BY ใช้ระบุ วิธีการเรียงลำดับการแสดงผลข้อมูล ซึ่งอาจจะมีหรือไม่มีก็ได้

ตัวอย่างง่ายๆของประโยคคำสั่งก็คือ

```
SELECT  *
FROM    ชื่อตาราง
```

ซึ่งจะทำการแสดงข้อมูลทั้งหมดจากตารางที่ระบุออกมา จะสังเกตได้ว่าเราใส่เครื่องหมาย * ไว้ข้างหลัง SELECT หมายความว่าให้เรียกดูทุกคอลัมน์นั่นเอง

ถ้าต้องการระบุเงื่อนไขของข้อมูลเฉพาะแถวของข้อมูลที่ต้องการเรียกดู กระทำโดยการใส่เงื่อนไขไว้ข้างหลังคำว่า WHERE โดยมีรูปแบบดังนี้

```
SELECT  *
FROM    ชื่อตาราง
```

WHERE เงื่อนไข

ซึ่งอาจจะมีเงื่อนไขมากกว่า 1 เงื่อนไขก็ได้โดยการเชื่อมด้วย AND หรือ OR

ถ้าเราสนใจจะดูข้อมูลเฉพาะเพียงบางคอลัมน์ก็สามารถทำได้ โดยการระบุชื่อคอลัมน์ไว้ข้างหลัง SELECT โดยคั่นระหว่างชื่อคอลัมน์ด้วยเครื่องหมาย "," มีรูปแบบดังนี้

SELECT ชื่อคอลัมน์ , ชื่อคอลัมน์ ,ชื่อคอลัมน์
FROM ชื่อตาราง
WHERE เงื่อนไข

ถ้าเราต้องการจะจับกลุ่มข้อมูลที่เราสนใจเข้าด้วยกัน เพื่อกระทำฟังก์ชันบางอย่างกับแต่ละกลุ่มก็สามารถทำได้ โดยใช้ GROUP BY ตัวอย่างเช่น ถ้าเราต้องการคำนวณหาจำนวนเงินที่จะได้จากการสั่งทั้งหมด โดยต้องการให้แยกจำนวนเงินตามรหัสการสั่ง ก็จำเป็นต้องใช้ GROUP BY เข้ามาช่วย โดยใส่ชื่อคอลัมน์ของกลุ่มที่เราจะจัดเข้าด้วยกัน มีรูปแบบดังนี้

SELECT ...
FROM ...
WHERE ...
GROUP BY ชื่อคอลัมน์หรือกลุ่มของชื่อคอลัมน์

ส่วนการใช้ HAVING จะเป็นการระบุเงื่อนไขในการจัดกลุ่ม ซึ่งคล้าย ๆ กับการใช้ WHERE ต่างกันที่ WHERE เป็นเงื่อนไขของข้อมูลเฉพาะแถว ดังนั้นการใช้ HAVING ก็เป็นการใช้ฟังก์ชันกรองกลุ่มอีกครั้งนั่นเอง มีรูปแบบดังนี้

SELECT ...
FROM ...
WHERE ...
GROUP BY ...
HAVING เงื่อนไขที่เป็นฟังก์ชัน

นอกจากนี้ เรายังสามารถกำหนดให้มีการเรียงลำดับผลลัพธ์ได้ตามค่าที่อยู่ในคอลัมน์ที่ระบุ โดยการใช้ ORDER BY ซึ่งสามารถเรียงจากน้อยไปมาก (ASC) หรือเรียงจากมากไปน้อย (DESC) ก็ได้ ด้วยการใส่ชื่อคอลัมน์ที่ต้องการให้เรียงลำดับ หรือ ใส่เป็นตำแหน่งของคอลัมน์ในตารางที่ทำการ SELECT ซึ่งนับจากซ้ายไปขวา และถ้าในกรณีที่มีข้อมูลมากกว่า 1 แถวที่มีค่าเหมือนกัน ก็จะเรียงลำดับตามคอลัมน์ถัดมาที่เราระบุไว้ในคำสั่ง ORDER BY โดยมีรูปแบบดังนี้

กรณีเรียงจากน้อยไปมาก จะใส่แต่คอลัมน์เท่านั้น ไม่ต้องระบุ ASC รูปแบบคือ

ORDER BY ชื่อคอลัมน์ หรือ ตำแหน่งของคอลัมน์ , ชื่อคอลัมน์ หรือ ตำแหน่งของคอลัมน์

กรณีเรียงจากมากไปน้อย นิยมใส่เป็นตำแหน่งของคอลัมน์ แล้วต้องระบุ DESC ด้วย รูปแบบคือ

ORDER BY ตำแหน่งของคอลัมน์ DESC , ตำแหน่งของคอลัมน์

จากที่ผ่านมาเป็นการเรียกดูข้อมูลจากตารางเพียงตารางเดียวเท่านั้น ใน SQL อนุญาตให้เรียกดูข้อมูลจากตารางมากกว่า 1 ตารางได้ มี 2 วิธี ได้แก่ การดูแบบย่อย (Subquery) และ การดูแบบรวมกัน (Join method)

วิธีการแรก การดูแบบย่อย หรือ การดูแบบซ้อนกัน (Nested query) เป็นการใส่ประโยคการเรียกดูซ้อนเข้าไปในประโยคการเรียกดูอีกประโยคหนึ่ง โดยมีรูปแบบคือ

```
SELECT    ....
FROM      ....
WHERE     ชื่อคอลัมน์ IN
(SELECT   ...)
```

วิธีการทำงาน จะเริ่มทำงานจากข้างในก่อน สิ่งที่น่าสังเกตจากรูปแบบคือ การใช้ IN แทนเครื่องหมาย "-" ในส่วนของประโยคหลัก ทั้งนี้ก็เพราะเราอาจได้ผลลัพธ์จากการเรียกดูประโยคที่อยู่ข้างในมามากกว่า 1 แถว ดังนั้นจึงใช้คำว่า IN แต่ถ้าผู้ใช้แน่ใจว่า ผลลัพธ์จากประโยค SELECT ที่ซ้อนอยู่ข้างในจะมีเพียง 1 ค่าเสมอก็สามารถใช้เครื่องหมาย "-" หรือ ">" หรือ "<" ทำนองนี้ได้

วิธีการที่ 2 คือวิธีการดูข้อมูลที่อยู่ร่วมกัน ซึ่งจะทำการเรียกดูข้อมูลจากตารางได้มากกว่า 1 ตาราง โดยเลือกดูเฉพาะแถวที่มีค่าข้อมูลบางส่วนร่วมกันอยู่ มีรูปแบบคือ

```
SELECT    ชื่อคอลัมน์
FROM      ชื่อตารางที่1 , ชื่อตารางที่ 2
WHERE     ชื่อตารางที่ 1 . ชื่อคอลัมน์ = ชื่อตารางที่ 2 . ชื่อคอลัมน์
```

2.1.2.2 **ฟังก์ชันต่างๆ** (Built-in Function)

จะช่วยให้เกิดความคล่องตัวมากขึ้นในการเรียกดูข้อมูล ได้แก่ฟังก์ชันต่อไปนี้

```
COUNT     ใช้ในการนับจำนวน
SUM       รวมค่าในคอลัมน์ที่กำหนด
AVG       แสดงค่าเฉลี่ยของคอลัมน์ที่กำหนด
```

MAX แสดงค่ามากที่สุดของข้อมูลในคอลัมน์ที่กำหนด

MIN แสดงค่าน้อยที่สุดของข้อมูลในคอลัมน์ที่กำหนด

โดยมีรูปแบบการออกคำสั่งคือ

SELECT ชื่อฟังก์ชัน (ชื่อคอลัมน์)

FROM ชื่อตาราง

WHERE เงื่อนไข

นอกจากนี้ยังมีคำว่า DISTINCT ถ้าเราใส่คำว่า DISTINCT ลงในคำสั่งด้วยแล้ว การทำงานของคำสั่งจะทำเฉพาะกับข้อมูลที่มีค่าไม่ซ้ำกันเลยเท่านั้น ซึ่งในแง่ของ MAX และ MIN แล้ว การใช้ DISTINCT จะไม่มีผลใด ๆ แต่ที่จะเห็นผลกระทบมากที่สุด ก็คงได้แก่ การนับด้วยฟังก์ชัน COUNT

ส่วนกรณีที่ค่าของข้อมูลที่เกี่ยวข้องเป็นค่าว่าง (Null) ฟังก์ชันเหล่านี้ก็จะทำการข้ามข้อมูลเหล่านี้ไปให้โดยอัตโนมัติ โดยไม่นำมาใช้ในการประมวลด้วยเลย ยกเว้นการใช้ฟังก์ชัน COUNT (*) ซึ่งจะทำการนับทุกแถวแม้ว่าข้อมูลในคอลัมน์นั้นจะเป็นค่าว่างก็ตาม

2.1.2.3 **การใช้ข้อมูล**

คำสั่งสำหรับการใช้ข้อมูล ได้แก่ กลุ่มคำสั่งที่ใช้ในการเพิ่มเติม , แก้ไข และลบข้อมูลออกจากตาราง

1) การเพิ่มเติมข้อมูล : คำสั่ง INSERT

คำสั่ง INSERT ใช้ในการเพิ่มเติมข้อมูลแถวใหม่ลงในตาราง โดยอาจจะระบุข้อมูลไว้ในคำสั่งเลย หรือดึงข้อมูลออกมาจากตารางอื่นก็ได้ โดยบังคับไว้ว่า ข้อมูลที่จะเติมลงไปนี้จะต้องมีข้อมูลในแต่ละคอลัมน์ตรงตามคอลัมน์ในตารางที่จะเพิ่มเติมลงไปด้วย ซึ่งมีรูปแบบดังนี้

INSERT INTO ชื่อตาราง

VALUES (ค่าที่ต้องการใส่ในคอลัมน์ต่าง ๆ) หรือ

INSERT INTO ชื่อตาราง : SELECT ...

โดยทั่ว ๆ ไปแล้ว คำสั่ง INSERT มักจะใช้ในการเพิ่มเติมข้อมูลที่มีจำนวนไม่มากนัก ส่วนการเพิ่มเติมข้อมูลลงไปในตารางหนึ่ง โดยดึงเอาข้อมูลบางส่วนมาจากอีกตารางหนึ่งก็สามารถทำได้ โดยการใช้คำสั่ง INSERT ควบคู่กับคำสั่ง SELECT ตัวอย่างเช่น ถ้าเรามีตารางชื่อ CUST 1 ซึ่งประกอบด้วย Attribute ดังนี้

CUST 1 (NUMBER , NAME)

โดยที่ NUMBER และ NAME มีชนิดและโดเมนเช่นเดียวกับ CUSTOMER_NUMBER และ NAME ในตาราง CUSTOMER การออกคำสั่งคือ

```
INSERT  INTO  CUST 1 :
SELECT   CUSTOMER_NUMBER , NAME
FROM     CUSTOMER
WHERE   CREDIT_LIMIT  >  5000
```

จะเป็นการดึงเอารหัส และชื่อของลูกค้าที่มีวงเงินเครดิตมากกว่า 5000 บาท เก็บไปใส่ในตาราง CUST 1

2) การแก้ไขข้อมูล : คำสั่ง UPDATE

คำสั่ง UPDATE ใช้ในการแปลี่ยนแปลงแก้ไขค่าของคอลัมน์ที่ต้องการเฉพาะในแถวที่มีเงื่อนไขสอดคล้องกับที่ระบุไว้ โดยเราสามารถแก้ไขได้มากกว่า 1 คอลัมน์ในคำสั่งเดียวกัน รูปแบบของคำสั่งคือ

```
UPDATE   ชื่อตาราง
SET        ชื่อคอลัมน์ =  ค่าข้อมูล
WHERE    เงื่อนไข
```

3) การลบข้อมูล : คำสั่ง DELETE

คำสั่ง DELETE ใช้ในการลบแถวของข้อมูลที่สอดคล้องกับเงื่อนไขที่ระบุออกจากตาราง โดยมีรูปแบบการออกคำสั่งคือ

```
DELETE   ชื่อตาราง
WHERE    เงื่อนไข
```

ซึ่งถ้าผู้ใช้ไม่ใส่ส่วนของ WHERE เอาไว้ก็จะมีผลทำให้ลบข้อมูลทุกแถวออกจากตารางที่ระบุ

2.1.2.4 **การนิยามข้อมูล**

คำสั่งสำหรับการนิยามข้อมูลเพื่อสร้าง หรือลบตาราง ดัชนี และวิวออกจากระบบ ซึ่งเป็นที่แน่นอนว่า คำสั่งในการนิยามตารางจะต้องเป็นคำสั่งแรกก่อนที่เราจะจัดเก็บข้อมูล หรือทำอะไรอื่น ๆ กับข้อมูลในตารางนั้น ๆ ได้

1) คำสั่งสำหรับนิยามตาราง : คำสั่ง CREATE TABLE

การสร้างตารางในระบบฐานข้อมูลกระทำได้โดยการใช้คำสั่ง CREATE TABLE ซึ่งในคำสั่งนี้เราต้องใส่ชื่อของตารางพร้อมทั้งชื่อของคอลัมน์ ชนิด และขนาดของข้อมูลของแต่ละคอลัมน์โดยเราไม่สามารถกำหนดข้อบังคับ เช่นโดเมนของข้อมูลในแต่ละคอลัมน์ได้ มีรูปแบบการออกคำสั่งดังนี้

CREATE TABLE ชื่อตาราง
(ชื่อคอลัมน์ที่ 1 data-type-1 ,
ชื่อคอลัมน์ที่ 2 data-type-2 ...)

และเมื่อใดก็ตามที่พิจารณาตารางแล้วเห็นว่า ตารางนั้นไม่มีความจำเป็นสำหรับระบบอีกต่อไป เราก็สามารถลบตารางนี้ออกจากระบบได้โดยการใช้คำสั่ง DELETE ลบข้อมูลออกจากตารางก่อน จากนั้นจึงใช้คำสั่ง

DROP TABLE ชื่อตาราง

เป็นการลบตารางออกจากระบบ

2) คำสั่งสำหรับนิยามวิวของผู้ใช้ : คำสั่ง CREATE VIEW

ผู้ใช้ระบบฐานข้อมูลแต่ละคนสามารถมองเห็นระบบในลักษณะที่แตกต่างกัน เราเรียกเค้าร่างของระบบที่ผู้ใช้แต่ละคนเห็นว่า "วิว" ในการนิยามวิวขึ้นมานี้ สามารถระบุได้ว่าให้วิวประกอบด้วยคอลัมน์ใดบ้าง โดยคอลัมน์เหล่านี้อาจจะมาจากตารางได้มากกว่า 1 ตาราง นอกจากนี้ยังสามารถระบุได้ว่าให้วิวที่นิยามอยู่นี้ ประกอบด้วยแถวที่สอดคล้องกับเงื่อนไขที่กำหนดเท่านั้น เมื่อวิวได้ถูกสร้างขึ้นมาแล้ว ผู้ที่มีสิทธิใช้วิวก็จะมองเสมือนหนึ่งว่าวิวเหล่านี้ก็คือตารางนั่นเอง และสามารถเรียกดูข้อมูลจากวิวนี้ได้เต็มที่ รูปแบบในการสร้างวิวคือ

CREATE VIEW ชื่อวิว
[(ชื่อคอลัมน์ 1 , ชื่อคอลัมน์ 2 , ...)]
AS SELECT ประโยคคำสั่ง SELECT

ซึ่งจะเป็นการสร้างตารางขึ้นมาใหม่ตามชื่อที่ระบุ อันจะประกอบด้วยคอลัมน์และแถวของข้อมูลตามที่ได้จากประโยค SELECT และผู้ใช้ที่มีอำนาจสามารถออกคำสั่งยกเลิกการใช้วิวใด ๆ ได้ด้วยคำสั่ง

DROP VIEW ชื่อวิว

จะเห็นได้ว่า วิว เป็นแค่เพียงตารางสมมติที่ไม่มีตัวตนจริง คำสั่ง และการกระทำทุกอย่างที่เกิดกับวิว จะถูกแปลให้ทำโดยตรงต่อตารางหลัก ดังนั้นประโยชน์จากวิวในการใช้งานจริง ส่วนใหญ่ได้แก่ การเรียกดูข้อมูล

3) คำสั่งสำหรับเพิ่มคอลัมน์ใหม่ : คำสั่ง ALTER TABLE

คำสั่งนี้ใช้ในการเพิ่มเติมฟิลด์ (หรือคอลัมน์) ใหม่เข้าไปในตารางได้โดยคอลัมน์ใหม่จะมีค่าเริ่มต้นเป็นค่าว่าง รูปแบบของคำสั่งคือ

ALTER TABLE ชื่อตาราง
ADD ชื่อคอลัมน์ ชนิด และ ขนาดของข้อมูล

4) คำสั่งสำหรับสร้างดัชนี : คำสั่ง CREATE INDEX

ในการเรียกใช้ข้อมูล การระบุดัชนีที่เหมาะสม จะช่วยให้การค้นหาข้อมูลเป็นไปอย่างรวดเร็วยิ่งขึ้น การใช้งานของคำสั่ง CREATE INDEX นี้ทำให้เราสามารถสร้างดัชนี สำหรับตารางใด ๆ ได้โดยดัชนีที่สร้างขึ้นมานี้อาจประกอบด้วยคอลัมน์มากกว่า 1 คอลัมน์ ก็ได้ รูปแบบของคำสั่งคือ

CREATE INDEX ชื่อดัชนี

ON ชื่อตาราง (ชื่อคอลัมน์ 1 [, ชื่อคอลัมน์ 2] ...)

นอกเหนือจากประโยชน์ในการเพิ่มความเร็วแล้ว การสร้างดัชนียังสามารถใช้ในการควบคุมค่าของข้อมูลได้ในระดับหนึ่ง คือ ถ้าเราต้องการสร้างหลักประกันว่าไม่ต้องการให้ข้อมูลในตารางมีค่าของคอลัมน์ใดซ้ำกันมากกว่า 1 แถว ก็สามารถควบคุมได้โดยเพิ่มคำว่า UNIQUE เข้าไปในคำสั่ง มีรูปแบบคือ

CREATE UNIQUE INDEX ชื่อดัชนี

ON ...

และถ้าดัชนีตัวใดไม่เป็นที่ต้องการใช้อีกต่อไป ก็สามารถยกเลิกได้โดยใช้คำสั่ง

DROP INDEX ชื่อดัชนี

2.1.2.5 การกำหนดสิทธิในการเข้าถึงข้อมูล

เป็นการกำหนดว่าจะให้ผู้ใช้คนใดมีสิทธิประการใดในการใช้ตารางเหล่านี้

1) คำสั่งสำหรับการให้สิทธิ : คำสั่ง GRANT

คือการให้สิทธิซึ่งเราสามารถระบุละเอียดลงไปว่า ให้สิทธิในการใช้ตารางหรือวิวใด มีรูปแบบคำสั่งดังนี้

GRANT Object_Priv ON Object TO User หรือ Role หรือ Public

โดยที่ Object_Privilege ได้แก่ Select , Insert , Delete , Update , Alter , Index , References , Execute , All

Object ได้แก่ Table , view , sequence , synonym , procedure , function , package , snapshots

นอกจากนี้การให้สิทธิสามารถที่จะให้กันต่อได้ (GRANT OPTION) ตัวอย่าง เช่น นาย ก ให้สิทธิแก่ นาย ข ในการใช้ข้อมูล จากตารางใดก็ตามพร้อมด้วย GRANT OPTION ก็หมายความว่าในตอนนี้ นาย ข สามารถให้สิทธิในการใช้ข้อมูลในตารางนี้แก่ นาย ค ต่อไป และถ้าให้พร้อมด้วย GRANT OPTION อีก นาย ค ก็สามารถให้สิทธิแก่คนอื่นต่อไปได้อีก มีรูปแบบคือ

GRANT ... ON ... TO ... WITH GRANT OPTION ;

2) คำสั่งสำหรับการยกเลิกสิทธิ : คำสั่ง REVOKE

เมื่อผู้ใช้คนหนึ่งได้ให้สิทธิแก่ผู้ใช้อีกคนหนึ่ง ผู้ใช้คนแรกก็สามารถยกเลิกสิทธิของผู้ใช้คนที่ 2 ได้ด้วยคำสั่ง REVOKE โดยการระบุสิทธิที่ต้องการยกเลิก ชื่อตาราง และบัญชีชื่อผู้ใช้ที่ถูกยกเลิกสิทธิ มีรูปแบบคำสั่งดังนี้

REVOKE Object_Priv ON Object FROM User หรือ Role หรือ Public

ในกรณีที่ได้รับการ GRANT OPTION มา การยกเลิกนี้จะกระทำกันเป็นทอด ๆ คือถ้าผู้ใช้คนแรกถูกยกเลิกสิทธิ คนต่อ ๆ มา ก็จะหมดสิทธิในการใช้ไปด้วย

2.2 จาวาสคริปต์ (Java Script)

2.2.1 JavaScript คืออะไร

JavaScript เป็นภาษาสคริปต์แบบใหม่ที่พัฒนาโดย Netscape เราสามารถสร้าง web-page ที่มีการโต้ตอบกับผู้ใช้ได้อย่างง่ายดาย เป็นภาษาที่ผู้เขียน HTML สามารถเรียนรู้ได้ง่าย สามารถใช้ จาวาสคริปต์เพิ่มเติมความสามารถของเว็บเพจให้ดูมีชีวิตชีวา มีการโต้ตอบกับผู้ใช้ หรือแม้กระทั่งการสร้างเกม

2.2.2 การใช้งาน JavaScript

ต้องมี browser ที่สามารถใช้ JavaScript ได้ ยกตัวอย่างเช่น Netscape Navigator (ตั้งแต่เวอร์ชัน 2.0) หรือ Microsoft Internet Explorer (MSIE ตั้งแต่ เวอร์ชัน 3.0) ซึ่งเป็น browser ที่นิยมใช้กันทั่วไป JavaScript สามารถทำงานบน browser ทั้งสองได้ แต่จำเป็นต้องมีพื้นฐานเกี่ยวกับ HTML ก่อน เพราะคำสั่ง Java Script ต้องอยู่ภายในคำสั่ง SCRIPT ของ HTML เท่านั้น มีวิธีในการทำดังตัวอย่าง:

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
 <script language="JavaScript">
   document.write("This is JavaScript!")
 </script>
<br>
Back in HTML again.
```

```
</body>
</html>
```

ดูทั่วไปมันก็เหมือนกับ HTML ปรกติ แต่ที่แตกต่างออกไปก็คือส่วนนี้:

```
<script language="JavaScript">
   document.write("This is JavaScript!")
 </script>
```

ส่วนนี้คือส่วนที่เป็น JavaScript ซึ่งเราสามารถเซฟร่วมกับไฟล์ HTML ปรกติ และ จะต้องทำงานบน browser ที่รู้จัก JavaScript เท่านั้น ผลลัพท์ของโค้ดข้างต้น(ถ้า browser ของคุณสามารถใช้ JavaScript ได้ คุณจะเห็นผลลัพท์ 3 บรรทัดดังต่อไปนี้)

```
This is a normal HTML document.
This is JavaScript!
Back in HTML again.
```

ทุกอย่างที่อยู่ระหว่าง <script> และ </script> จะถูกแปลความ(interpreted) ว่าเป็นโค้ด JavaScript การใช้ document.write() ซึ่งเป็นคำสั่งสำคัญที่สุดอันหนึ่ง ของการโปรแกรม JavaScript โดย document.write() ใช้ในการ เขียนอะไรก็ตามแต่ความต้องการของเราออกมาทางเอกสารจริง(ในที่นี้เอกสาร คือไฟล์ HTML) ซึ่งในตัวอย่างข้างต้นก็คือโปรแกรมขนาดเล็กที่สามารถเขียน คำว่า This is JavaScript! บนเอกสาร HTML

ถ้า browser ที่ไม่สนับสนุน JavaScript ก็จะไม่รู้จักแท็ก <script> มันจะไม่สนใจ แท็กนั้นและจะแสดงโค้ดของโปรแกรมออกมาเหมือนกับตัวอักษรปรกติ นั่นหมายความว่า ผู้ใช้จะเห็นโค้ดของ JavaScript ออกทาง browser วิธีในการแก้สำหรับ browser รุ่นเก่า โดยเราจะใช้ comment ของ HTML ในการซ่อนมัน คือใช้แทกนี้ <!-- --> โค้ดที่ปรับปรุงใหม่จากตัวอย่างข้างต้นก็จะเป็นลักษณะดังนี้:

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
 <script language="JavaScript">
```

```
<!-- hide from old browsers

  document.write("This is JavaScript!")

// -->
</script>
<br>
Back in HTML again.
</body>
</html>
```

ผลลัพท์ของมันก็จะเป็นในลักษณะนี้

This is a normal HTML document.
Back in HTML again.

ถ้าเราไม่ใส่ comment ผลลัพท์จาก browser ที่ไม่สนับสนุน JavaScript จะได้ผลดังนี้

This is a normal HTML document.
document.write("This is JavaScript!")
Back in HTML again.

คุณไม่สามารถซ่อนโค้ดของ JavaScript ได้อย่างสมบูรณ์แบบ ที่ทำนั้นเป็นเพียงป้องกันไม่ให้โค้ดของโปรแกรมแสดงออกทางหน้าจอของ browser รุ่นเก่าที่ไม่รู้จัก JavaScript แต่ผู้ใช้ทั่วไปยังคงสามารถเห็นโค้ดของโปรแกรมผ่าน 'View document source' ได้อยู่ดี ดังนั้นไม่มีทางที่จะซ่อนโค้ดของโปรแกรมได้

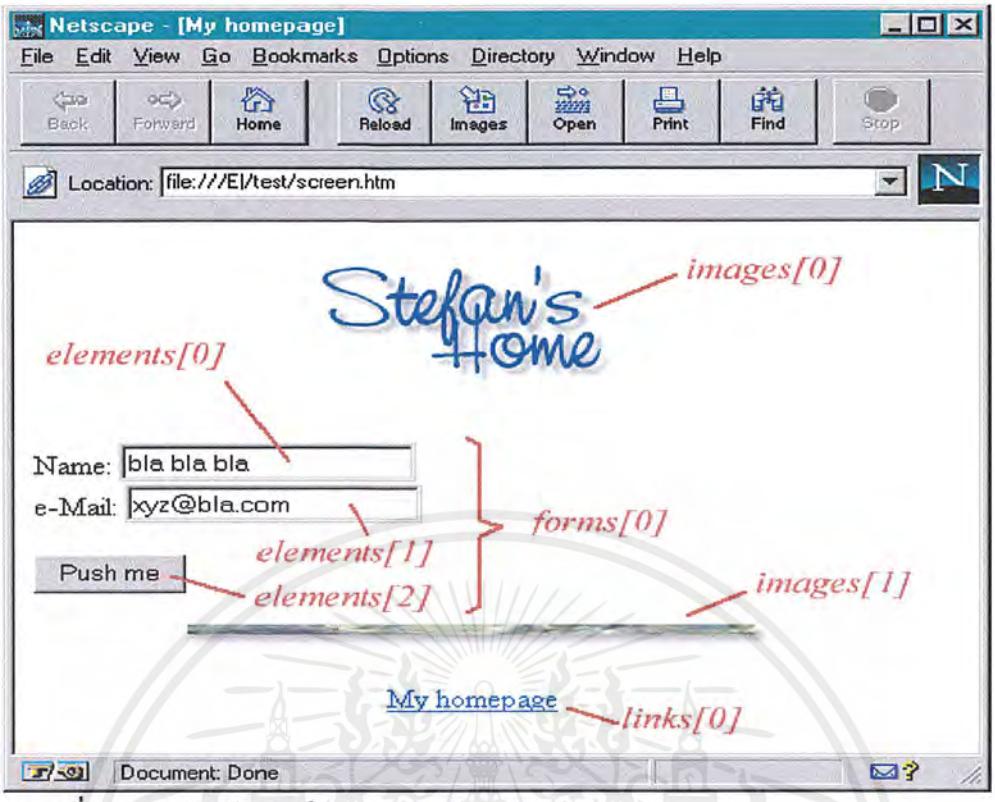
2.2.3 การจัดระดับความสัมพันธ์ของ JavaScript

JavaScript จะทำการจัดส่วนประกอบต่างๆของ web-page ให้อยู่ในลักษณะของความสัมพันธ์เกี่ยวเนื่องกัน ทุกๆส่วนประกอบจะถูกมองเป็นออปเจ็ก(หรือวัตถุ) ทุกออปเจ็กจะประกอบไปด้วย property (คุณสมบัติ) และ method (เมทอด ลักษณะเหมือนฟังก์ชัน) ถ้าเราแก้ไข property ก็จะเปลี่ยนคุณลักษณะของออปเจ็ก หรือถ้าเราต้องการให้ออปเจ็กทำฟังก์ชันซึ่งเป็นฟังก์ชันเฉพาะของออปเจ็กนั้น เราก็ต้องเรียกเมทอดของออปเจ็ก ซึ่งทั้งคุณสมบัติหรือเมทอดมันจะขึ้นกับชนิดของออปเจ็ก

ดังนั้นเราจำเป็นต้องเข้าใจความสัมพันธ์ของแต่ละออปเจ็กก่อนที่จะเรียกใช้มัน ซึ่งสามารถเข้าใจการทำงานของมันได้อย่างรวดเร็วโดย ดูจากตัวอย่างของโค้ด HTML ต่อไปนี้

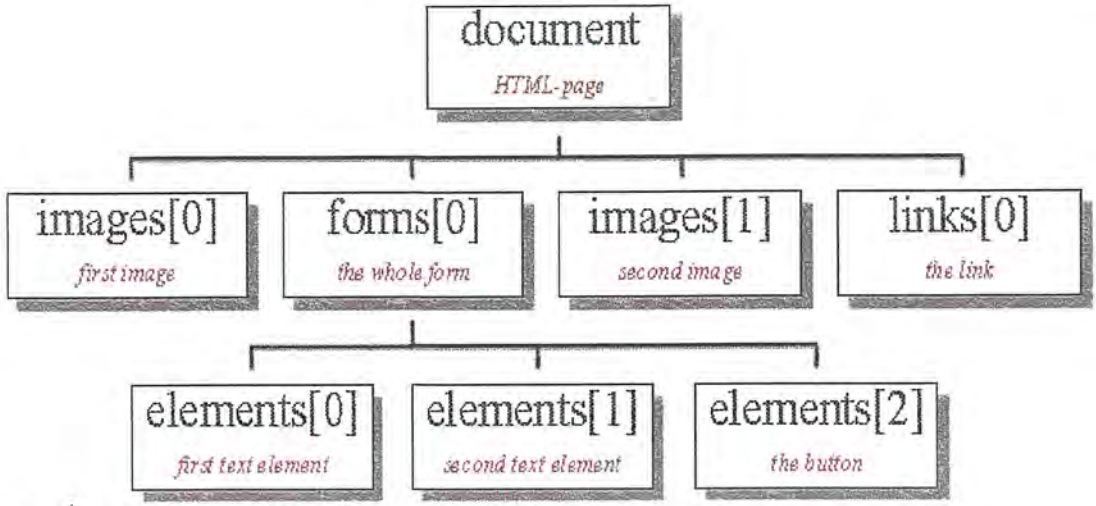
```
<html>
<head>
<title>My homepage
</head>
<body bgcolor=#ffffff>
<center>
<img src="home.gif" name="pic1" width=200 height=100>
</center>
<p>
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
e-Mail:
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me" name="myButton" onClick="alert('Yo')">
</form>
<p>
<center>
<img src="ruler.gif" name="pic4" width=300 height=15>
<p>
<a href="http://rummelplatz.uni-mannheim.de/~skoch/">My homepage</a>
</center>
</body>
</html>
```

จะได้ผลของหน้าของเพจนี้ดังภาพที่ 2.1



ภาพที่ 2.1 ผลลัพธ์ของหน้าจอ My homepage

จะเห็นได้ว่ามีภาพอยู่หนึ่งภาพ ,หนึ่งlink , หนึ่ง form ที่มีสอง field(ช่อง) กับอีกหนึ่งปุ่ม จากมุมมองของ JavaScript วินโดว์ของ browser จะเป็นออปเจ็กชื่อ window โดยออปเจ็ก window จะมี statusbar(แถบสถานะ) เป็นส่วนประกอบ อันหนึ่งในวินโดว์เรา สามารถโหลดเอกสารของ HTML(หรือเป็นไฟล์ชนิดอื่น แต่ในที่นี้จะพูดเฉพาะ ไฟล์ประเภท HTML) ภายในหน้าต่างที่เป็นเวบเพจจะเป็นออปเจ็ก document นั่นหมายความว่า ออปเจ็ก document จะเป็นตัวแทนของเอกสาร HTML ที่ถูกโหลดในขณะนั้น ออปเจ็ก document เป็นออปเจ็กที่สำคัญมากใน Java Script เพราะเราจะนำมาใช้ซ้ำแล้วซ้ำอีก ส่วน property(คุณสมบัติ) ของออปเจ็ก document ยกตัวอย่างได้แก่สี background สิ่งที่ลืมไม่ได้ก็คือ ออปเจ็ก HTML ทุกตัวเป็น property ของ ออปเจ็ก document ตัวอย่างเช่น link , form ฯลฯ ภาพต่อไปจะเป็นการแสดงความสัมพันธ์ของออปเจ็กในตัวอย่างที่ผ่านมา:



ภาพที่ 2.2 แสดงความสัมพันธ์ของออปเจ็กในโปรแกรม My homepage

ถ้าต้องการข้อมูลของแต่ละออปเจ็กและวิธีในการจัดการมัน ก็ต้องทราบวิธีในการเข้าถึงออปเจ็กต่างๆเสียก่อน ซึ่งจะเห็นได้จากสายสัมพันธ์ของ ออปเจ็กที่จัดเป็นชั้นๆ การที่จะอ้างถึงภาพแรกในเพจ HTML ได้อย่างไรนั้น ทำได้โดยดูที่แผนภาพของความสัมพันธ์โดยเริ่มจากด้านบนสุด จากออปเจ็กตัวแรกที่ชื่อ document ภาพแรกของเพจจะ ถูกนำเสนอโดยออปเจ็ก images[0] นั่นหมายความว่าสามารถเข้าถึงออปเจ็กภาพนี้บน JavaScript ได้โดยอ้างดังนี้

document.images[0]

ถ้าสมมุติว่า อยากรู้ว่าจะเข้าถึงส่วนประกอบแรกของ form ก็ต้องการทราบก่อนว่าจะเข้าถึงออปเจ็กอย่างไร ขั้นตอนแรกให้เริ่มจากส่วนบนสุดของแผนภาพความสัมพันธ์ แล้วไล่ตามเส้นทางจะพบออปเจ็กต่างๆ ตามเส้นทาง ให้เราทำการรวบรวมรายชื่อของออปเจ็กที่ผ่านไว้ด้วยกัน ดังนั้นการที่จะเข้าถึงช่องแรกของ form จะอ้างดังนี้ document.forms[0].elements[0] ซึ่งต้องทราบ property และ method ที่ออปเจ็กมีให้เรียกใช้ จะเห็นได้ว่าออปเจ็กตัวนี้มี property ที่ชื่อ value อยู่ เราสามารถอ่านค่าที่อยู่ในออปเจ็กตัวนี้ได้ โดยดูจากโค้ดตัวอย่าง:

```
name= document.forms[0].elements[0].value;
```

ข้อความจะถูกเก็บไว้ในตัวแปร name แล้วสามารถนำมันไปใช้งานได้ อย่างเช่น เราสามารถสร้างป๊อปอัปวินโดว์อย่างนี้ได้ alert("Hi " + name) ถ้าป้อนอินพุต 'Stefan' คำสั่ง alert("Hi " + name) จะสร้าง ป๊อปอัปวินโดว์ที่มีคำว่า 'Hi Stefan' ออกมา

ถ้าเขียนเพจที่มีขนาดใหญ่มากๆ อาจจะสับสนกับตัวเลขของออปเจ็กแต่ละตัวบนหน้าจอ อย่างเช่น document.forms[3].elements[17] หรือ document.forms[2].elements[18] ซึ่งอาจจะลืมไปแล้วว่ามันคืออะไร วิธีแก้ไขทำได้โดยตั้งชื่อที่ไม่ซ้ำกัน และควรมีความหมายชัดเจนให้ออปเจ็กแต่ละตัว โดยทำตามตัวตัวอย่างข้างล่างนี้:

```
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
...
```

นั่นหมายความว่า forms[0] ถูกเปลี่ยนชื่อเป็น myForm และ elements[0] สามารถใช้ชื่อว่า name แทนได้ (ดังที่ได้กำหนดไว้ใน property ของแท็ก <input>) ดังนั้นแทนที่เราจะเขียน

name= document.forms[0].elements[0].value;

เราสามารถเขียนได้ดังนี้

name= document.myForm.name.value;

วิธีนี้ทำให้เพจที่มีขนาดใหญ่ที่มีออปเจ็กหลายตัวสามารถจัดการได้ง่ายยิ่งขึ้น (ระวังไว้อย่างหนึ่งว่าไม่สามารถเขียน myform แทน myForm ได้ มันไม่ใช่ตัวเดียวกัน) property หลายตัวของ JavaScript ไม่ได้เข้มงวดกับการเขียนหรืออ่านค่ามากนัก คุณสามารถกำหนดค่าให้กับ property ของออปเจ็กได้โดยตรง

นอกจากออปเจ็กวินโดว์และ document แล้วยังมีออปเจ็กที่สำคัญอีกตัวหนึ่งก็คือออปเจ็ก location มันเป็นออปเจ็กที่ใช้แทนตำแหน่งของเพจที่ถูกโหลด ดังนั้นถ้าคุณต้องการโหลดเพจ http://www.xyz.com/page.html ด้วยออปเจ็ก location.href สามารถทำได้โดยกำหนดค่าให้กับ location.href ได้โดยตรง ด้านล่างนี้เป็นตัวอย่างของการสร้างปุ่มเพื่อใช้ในการโหลดเพจใหม่ขึ้นมา

```
<form>
<input type=button value="Yahoo"
 onClick="location.href='http://www.yahoo.com'; ">
</form>
```

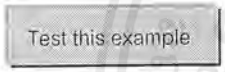
2.2.4 อีเว็นท์

Event และ event handlers เป็นส่วนสำคัญของการเขียน JavaScript ส่วนมากแล้ว Event ก็คือการกระทำอะไรก็แล้วแต่ของผู้ใช้กับ webpage ถ้าผู้ใช้ทำการกดเมาส์หนึ่งครั้งจะเกิด event Click ถ้าตัวชี้ของเมาส์เคลื่อนผ่าน link ก็จะเกิด even MouseOver ถ้าเราต้องการให้โปรแกรมของเราตอบสนองต่อ event เราต้องสร้าง event-handler ซึ่งหมายถึงการเขียนโปรแกรมเพื่อสนอง event นั้น ๆ การที่ปุ่มสามารถทำให้วินโดว์ป๊อปอัปออกมา เมื่อเรากดมัน นั่นหมาย

ความว่าวินโดว์ที่ป๊อปอัปออกมาตอบสนองต่อ event Click ส่วน event-handler ที่ใช้จัดการกับ event นั้นเรียกว่า onClick มันจะเป็นตัวบอกให้ Computer ทราบว่ามันควรจะทำอย่างไรเมื่อเกิด event นั้น

การเขียนโปรแกรมแบบในลักษณะที่ตอบสนองกับเหตุการณ์(event)ที่เกิดขึ้นเรียกว่า การเขียนโปรแกรมแบบ event driven ซึ่งเป็นลักษณะเฉพาะตัวของการเขียนโปรแกรมบนวินโดว์ ก็คือจะมีตารางของเหตุการณ์ที่จะเกิดขึ้นกับวินโดว์นั้นๆ บางเหตุการณ์ก็จะมีโปรแกรมที่สนอง event นั้นๆอยู่แล้ว บางทีก็ไม่มี หน้าที่ของเราก็ไปเขียนโปรแกรมสำหรับ event ที่เราจะตอบสนองนั้นแล้วนำไป แทนที่(handdle บางที่เรียกว่า hook)โปรแกรมที่มีอยู่เดิม (ถ้ามี) เวลาเกิด event ที่เราทำการ handdle แล้วมันก็จะไปเรียกโปรแกรมของเราขึ้นมา ตัวอย่างข้างล่างจะแสดงให้เห็น ของ event-handler ของ onClick:

```
<form>
<input type="button" value="Click me" onClick="alert('Yo')">
</form>
```



จะเห็นบางสิ่งผิดปรกติเล็กน้อย คือ โค้ดของปุ่ม ที่สร้างจากฟอร์ม มีส่วนที่เพิ่มจากปรกติคือ onClick="alert('Yo')" ข้างในแท็ก <input> เป็น การกำหนดเหตุการที่จะเกิดขึ้นถ้ามีการกดปุ่ม ดังนั้นถ้ามี evern Click เข้ามาคอมพิวเตอร์ก็จะทำงานคำสั่ง alert('Yo') นี่เป็นโค้ด JavaScript อันหนึ่ง คำสั่ง alert() จะเป็นการสร้างวินโดว์ป๊อปอัปขึ้นมา ในเครื่องหมายคำพูด เราต้องใส่ข้อความลงไป ในที่นี้ก็คือคำว่า *'Yo'* ซึ่งจะเป็นตัวอักษรที่ปรากฏ ในป๊อปอัปวินโดว์

สคริปต์จะทำการสร้างหน้าต่างที่มีคำนี้ออกมาเมื่อมีการกดปุ่ม อย่างหนึ่งที่อาจทำให้สับสนก็คือคำสั่ง document.write() เราจะใส่มันระหว่าง เครื่องหมายฟันหนู (double quotes) แต่คำสั่ง alert() เราจะใช้มันระหว่าง เครื่องหมายหยดน้ำฝน (single quotes) ซึ่งโดยทั่วไปสามารถใช้ได้ทั้ง สองอย่าง แต่ในตัวอย่างนี้ เราเขียน onClick="alert('Yo')" เราใช้เครื่องหมายทั้งสองแบบ เพราะว่า ถ้าคุณเขียน onClick="alert("Yo")" คอมพิวเตอร์จะงงว่าส่วนไหนเป็น event-handler ของ onClick หรือส่วนไหนไม่ใช่ คุณจะใช้ double quote มาก่อน single quote หรือจะทำกลับกันก็ได้ นั่นหมายความว่าคุณสามารถเขียน onClick='alert("Yo")' ก็ได้ผลลัพท์เหมือนกัน

2.2.5 การสร้างเฟรม

วินโดว์ของ browser สามารถถูกแบ่งออกเป็นหลายๆเฟรม ความหมายของเฟรม ก็คือ พื้นที่สี่เหลี่ยมที่อยู่ในวินโดว์ของ browser แต่ละเฟรมจะแสดงเอกสารของมันเอง (ส่วนใหญ่ก็คือ

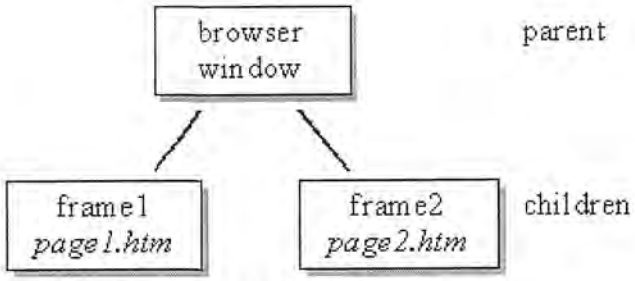
เอกสาร HTML นั่นเอง) ยกตัวอย่างเช่นคุณสามารถสร้างเฟรมสองเฟรม โดยที่เฟรมหนึ่งโหลดเพจของ Netscape และอีกเฟรมหนึ่ง โหลดเพจของ Microsoft การสร้างเฟรม จะต้องใช้สองแท็กก็คือ: <frameset> และ <frame> การสร้างเพจ HTML ที่มีสองเฟรมจะมีหน้าตาดังนี้:

```
<html>
<frameset rows="50%,50%">
 <frame src="page1.htm" name="frame1">
 <frame src="page2.htm" name="frame2">
</frameset>
</html>
```

นี้เป็นการสร้างสองเฟรมในหนึ่งหน้า ถ้าสังเกตุจะเห็น property rows ในแท็ก <frameset> นั่นหมายความว่าจะแบ่งหน้าออกเป็นสองหน้าในแนวของแถว (แบ่งตามแนวนอน) โดยที่แถวบนจะเกิดจากไฟล์ page1.htm ส่วนแถวล่างจะเกิดจากไฟล์ page2.htm ในแต่ละ frame สามารถตั้งชื่อให้กับมันได้ โดยใส่ไว้ที่ property ที่ชื่อ name ในแท็ก <frame> ซึ่งมันจะทำให้การเข้าถึงออปเจ็กทำได้ง่ายยิ่งขึ้น

```
<frameset cols="50%,50%">
 <frameset rows="50%,50%">
  <frame src="cell.htm">
  <frame src="cell.htm">
 </frameset>
 <frameset rows="33%,33%,33%">
  <frame src="cell.htm">
  <frame src="cell.htm">
  <frame src="cell.htm">
 </frameset>
</frameset>
```

จากที่ได้สร้างสองเฟรมขึ้นมาดังตัวอย่างข้างบน แผนภาพข้างล่างเป็นความสัมพันธ์ของเฟรมในตัวอย่างนี้:



ภาพที่ 2.3 แสดงความสัมพันธ์ของเฟรม

ส่วนบนสุดจะเป็นวินโดว์ของ browser ซึ่งมันจะถูกแบ่งออกเป็นสองเฟรม window จะเป็นแม่(parent) ของเฟรมทั้งสอง และเฟรมทั้งสองก็จะเป็นลูก ซึ่งจะตั้งชื่อที่ไม่เหมือนกันให้แก่เฟรมทั้งสองคือ frame1 และ frame2 เพื่อที่จะใช้ในการอ้างในการส่งข้อมูลระหว่างเฟรม

เราจะใช้ Script ในกรณีเมื่อมีการคลิกที่ link ของเฟรมแรกแต่ต้องการให้ เพจสามารถโหลดไปที่หน้าสองแทนที่จะเป็นหน้าเดียวกัน ยกตัวอย่างเช่นเราจะใช้เฟรม เพื่อเป็น menubar (หรือ navigationbar) โดยจะมีเฟรมหนึ่งอยู่กับที่แต่มี link ชี้ไปหลายๆที่ จะมาดูความสัมพันธ์ที่เป็นไปได้สามกรณีดังนี้:

- วินโดว์แม่ต้องการติดต่อกับเฟรมลูก
- เฟรมลูกต้องการติดต่อกับวินโดว์แม่
- เฟรมลูกต้องการติดต่อกับเฟรมลูก

ในมุมมองจากวินโดว์แม่จะมอง frame1 และ frame2 ได้โดยตรง ดังนั้นถ้าเราต้องการเขียน Script ติดต่อจากวินโดว์แม่กับเฟรมลูก ก็เพียงอ้างชื่อของเฟรม ดังตัวอย่างข้างล่างเราสามารถเขียน:

```
frame2.document.write("A message from the parent window.");
```

ในบางครั้งเราจำเป็นต้องติดต่อจากเฟรมลูกไปหาวินโดว์แม่ อย่างเช่นเราต้องการเอาเฟรมออก โดยให้วินโดว์ทำการโหลดหน้าใหม่ขึ้นมาแทนที่ เพจที่มีเฟรมเดิมอยู่เราสามารถติดต่อกับวินโดว์แม่โดยอ้างผ่าน parent โดยจะทำการกำหนด URL ให้กับ ออปเจ็ก location.href โดยทำการอ้างดังนี้: ตัวอย่างจะทำการกำหนด URL ให้กับวินโดว์แม่ใหม่

```
parent.location.href= "http://...";
```

แต่ที่ใช้งานกันประจำก็คือการติดต่อระหว่าง frame ลูกด้วยกันเอง ดังนั้น การติดต่อจาก frame1 ไปที่ frame2 โดยใช้คำสั่งจาก page1.htm ซึ่งเป็นของ frame1 เราจะทำได้อย่างไร เนื่องจาก ในแผนภาพจะเห็นได้ว่า ไม่มีเส้นทางที่จะติดต่อโดยตรงระหว่างเฟรมทั้งสอง เราไม่สามารถอ้างชื่อ

frame2 ได้โดยตรงจาก frame1 เพราะ frame1 ไม่ได้รู้จัก frame2 เลย แต่ในมุมมองของวินโดว์แม่จะรู้จัก frame ที่สองในชื่อ frame2 ดังนั้นจะต้องทำการอ้างตามลำดับดังนี้เพื่อที่จะทำการเข้าถึงออปเจ็ก document ของเฟรมที่สอง:

```
parent.frame2.document.write("Hi, this is frame1 calling.");
```

2.2.6 ฟอร์ม

2.2.6.1 การตรวจสอบตัวอักษร

ในบางครั้งต้องการบังคับอินพุตที่ป้อนเข้ามา ให้มีตัวอักษรและตัวเลขที่กำหนด อย่างเช่นหมายเลขโทรศัพท์ มีเพียงตัวเลขเท่านั้นที่สามารถป้อนเข้ามาได้ (ในกรณีนี้เราสมมุติว่าหมายเลข โทรศัพท์มีเพียงตัวเลขอย่างเดียว) แต่โดยทั่วไปจะทำการป้อนตัวอักษรพิเศษเข้าไปด้วย ยกตัวอย่างเช่น 01234-56789, 01234/56789 หรือ 01234 56789 (มีช่องว่างตรงกลาง) เราต้องให้ผู้ใช้สามารถป้อนตัวอักษรพิเศษได้ด้วย ซึ่งจะสามารถเพิ่มความสามารถในการตรวจสอบ ตัวอักษรพิเศษและตัวเลขให้กับโปรแกรม ดังตัวอย่างต่อไปนี้


```
<html>
<head>
<script language="JavaScript">
<!-- hide

function check(input) {
  var ok = true;

  for (var i = 0; i < input.length; i++) {
    var chr = input.charAt(i);
    var found = false;
    for (var j = 1; j < 13; j++) {// I change from check.length -> 13
      if (chr == check[j]) found = true;
    }
    if (!found) ok = false;
  }
```

```
 return ok;
}

function test(input) {

 if (!check(input, "1", "2", "3", "4",
      "5", "6", "7", "8", "9", "0", "/", "-", " ")) {

   alert("Input not ok.");
 }
 else {
   alert("Input ok!");
 }
}
// -->
</script>
</head>

<body>
<form>
Telephone:
<input type="text" name="telephone" value="">
<input type="button" value="Check"
 onClick="test(this.form.telephone.value)">
</form>
</body>
</html>
```

ฟังก์ชัน test() จะทำการตรวจสอบว่าค่าที่ป้อนเข้ามาถูกต้องหรือไม่

2.2.6.2 การส่งค่าที่อยู่ในฟอร์ม

อย่างที่ง่ายที่สุดคือการส่งไปเป็น e-mail ซึ่งวิธีนี้ใช้กันแพร่หลายมาก แต่ถ้าต้องการให้ server สามารถจัดการได้เราก็ต้องใช้ CGI (Common Gateway Interface)มาช่วยด้วย ซึ่งมันจะยอมให้จัดการอินพุตของฟอร์ม โดยอัตโนมัติ เช่นเราต้องการให้ server ทำการสร้างฐานข้อมูล

ของลูกค้าที่ป้อนเข้ามา ก็จะใช้ฟอร์มช่วยในการหาข้อมูลจาก database มันจะตอบสนองทันทีโดยไม่ต้องให้ผู้ดูแล server ทำการป้อนข้อมูลเอง ซึ่ง JavaScript ไม่สามารถทำเช่นนี้ได้

คุณไม่สามารถเขียน guestbook ได้เพราะว่า JavaScript ไม่สามารถเขียนไฟล์บน server ได้ คุณต้องใช้ CGI อย่างเดียว ตัวอย่างข้างล่างนี้เป็น HTML ปรกติ ไม่มี JavaScript แต่ถ้าต้องการตรวจข้อความในฟอร์ม ควรนำ JavaScript มาช่วยด้วย

```
<form method=post action="mailto:your.address@goes.here" enctype="text/plain">
Do you like this page?
  <input name="choice" type="radio" value="1">Not at all.<br>
  <input name="choice" type="radio" value="2" CHECKED>Waste of time.<br>
  <input name="choice" type="radio" value="3">Worst site of the Net.<br>
  <input name="submit" type="submit" value="Send">
</form>
```

ในส่วนที่ enctype="text/plain" หมายความว่าจะทำการส่งตัวอักษรเป็นตัวหนังสือ ล้วนๆ โดยไม่ต้องมีในรหัสหรือแทกพิเศษให้กับตัวอักษร ทำให้จดหมายอ่านได้สะดวก ถ้าต้องการตรวจสอบจดหมายก่อนที่จะส่ง ก็ทำได้โดยไปดักที่ even onSubmit ทำดังตัวอย่าง:

```
function validate() {
  // check if input ok
  // ...
  if (inputOK) return true
    else return false;
} ...
<form ... onSubmit="return validate()">
...
```

โปรแกรมนี้จะทำการตรวจสอบข้อมูลที่ป้อนเข้าไปในฟอร์มถ้าผิดจะไม่ส่ง

2.3 จาวาเซิร์ฟเลท (Java Servlet)

2.3.1 Servlet คืออะไร

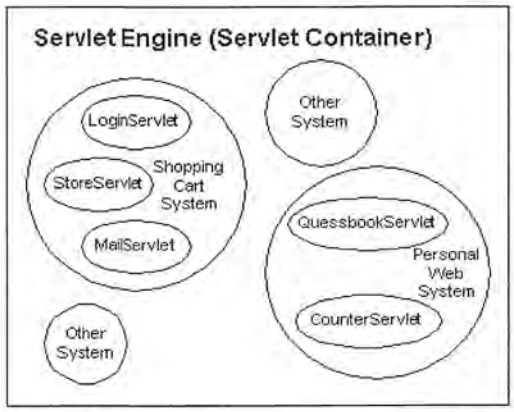
Servlet คือ Java Program ที่ถูกสร้างมาให้ทำงานได้บน HTTP Server ซึ่ง Servlet เป็น Server Side Application แบบหนึ่งซึ่งอ้างอิงคอนเซ็ปมาจาก CGI ข้อดีของ Servlet ที่อยู่เหนือ

CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือจาวานั่นเอง จาวาเป็นภาษาที่ใช้คอนเซ็ปของ Object Oriented ในการเขียน หลายคนที่เกี่ยวข้องกับการเขียนโปรแกรมสำหรับโปรเจคใหญ่ ๆ จะทราบดีว่า Object Oriented สามารถลดความซับซ้อนของโครงสร้างโปรแกรมรวมไปถึงการอำนวยความสะดวกในการ reuse ส่วนของโปรแกรมที่เขียนไว้แล้วเพียงไร นอกจากนี้จาวายังเป็นภาษาที่เป็นลักษณะแบบ platform independent ซึ่งจะช่วยให้เราสามารถที่จะทำการพัฒนาระบบโดยใช้ Environment อะไรก็ได้ซึ่งโดยทั่วไปมักนิยมใช้ Window Environment โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมารันบน Unix Environment เพื่อเพิ่มความเสถียรภาพของโปรแกรม

นอกจากนี้ Servlet ยังมีความเร็วที่สูงกว่า CGI เพราะ Servlet ใช้หลักการของ thread โดยจะทำการสร้าง 1 thread ต่อหนึ่ง request ที่มาจาก client ซึ่งในทางกลับกัน CGI จะทำการสร้าง 1 process ต่อหนึ่ง request ซึ่งจะทำให้เปลืองทรัพยากรมากกว่าและ process ในการรันก็จะช้ากว่าด้วย ท้ายที่สุดจุดเด่นที่สำคัญของ Servlet ก็คือ API (Application Programming Interface) โดยระบบที่ทำการพัฒนาโดยใช้คอนเซ็ปของ Servlet จะสามารถเรียกใช้ API ที่ทางจาวามีมาให้ (javax.servlet.*, javax.servlet.http.*) ซึ่งจะช่วยทำให้การพัฒนาระบบดังกล่าวง่ายและเร็วยิ่งขึ้น

2.3.2 ความหมายของ Servlet Engine

ในการรันระบบ (หมายถึง Zone (Apache JServ) หรือ Web Application (Tomcat) ก็ได้) ที่เขียนขึ้นโดยใช้หลักการของ Servlet เราจะต้องนำระบบดังกล่าวมาบรรจุอยู่ในสิ่ง ๆ หนึ่งที่เรียกว่า Servlet Engine ให้นึกถึง Servlet Engine คล้าย ๆ กับกล่อง ๆ หนึ่งที่ใส่ลูกปิงปองไว้หลายลูก โดยลูกปิงปองแต่ละลูกก็คือระบบ ๆ หนึ่งนั่นเอง หลายคนอาจสงสัยทำไมถึงใช้คำว่าระบบ โดยทั่วไป Server Side Application หนึ่ง ๆ ที่ถูกเขียนขึ้นโดยใช้ Servlet API จะถูกเรียกว่า Servlet ในหนึ่งระบบอาจประกอบด้วย Servlet หลายอัน ยกตัวอย่างเช่น ระบบที่เกี่ยวกับ Shopping Cart อาจจะประกอบด้วย Servlet ที่ทำหน้าที่ในการเช็คล็อกอิน , Servlet ที่ทำหน้าที่ในการเก็บข้อมูลสินค้า , Servlet ที่ทำหน้าที่ในการส่งเมล์กลับไปยังลูกค้าเพื่อบอกว่าได้ทำการส่งของไปให้แล้ว เป็นต้น ดังนั้นถ้ามองโดยรวมแล้ว Servlet Engine ก็คือที่รวมของระบบตั้งแต่หนึ่งระบบถึงหลายระบบ โดยแต่ละระบบจะประกอบด้วย Servlet หนึ่งอันหรือมากกว่า ดังภาพที่ 2.4



ภาพที่ 2.4 แสดง Servlet Engine and its Servlets

Servlet Engine เป็นเพียงกล่อง ๆ หนึ่งที่ใช้บรรจุและรันกลุ่มของ Servlet เท่านั้น ในการที่จะทำการติดต่อสื่อสารกับ Client ตัว Servlet Engine นี้จะต้องทำงานร่วมกัน Web Server ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับ Client อีกทีหนึ่ง เมื่อใดก็ตามที่มี request ส่งมาจาก Client ถ้า request นั้นเจาะจงมาที่ตัว Servlet ทาง Web Server ก็จะทำการ forward ตัว request นั้นมาให้ Servlet Engine ซึ่งทาง Servlet Engine ก็จะทำการเรียก Servlet ที่ Client ต้องการขึ้นมาทำการประมวลผล request นั้น โดยท้ายสุด Servlet จะส่งผลกลับไปให้ Servlet Engine , Servlet Engine ก็จะ forward ผลที่ได้กลับไปให้ Web Server ซึ่ง Web Server ก็จะส่งผลกลับไปให้ Client

Sevlet Engine อาจจะเป็นส่วนที่ติดมากับ Web Server อยู่แล้วยกตัวอย่างเช่น Servlet Engine ที่อยู่ใน Netscape Enterprise Server , IBM WebSphere หรืออาจจะเป็นส่วนที่เป็น Add-on ให้กับ Web Server ก็ได้เช่น Apache Jserv , Tomcat , JRun หรือแม้กระทั่งเป็นส่วนหนึ่งที่อยู่ใน Web Application เช่น BEA Weblogic เป็นต้น ทั้งนี้การเลือกใช้ Servlet Engine แต่ละชนิดก็มักขึ้นอยู่กับปัจจัยหลายอย่างเช่น ความสะดวกในการรวมระบบที่จะสร้างขึ้นมาใหม่กับระบบที่มีอยู่แล้ว, งบประมาณที่มีอยู่สำหรับโครงการหรืออาจจะรวมไปถึงทักษะและประสบการณ์ส่วนตัวของนักพัฒนาแต่ละคน

2.3.3 การทำงานของ Servlet

สมมุติว่าเรามี interface อันหนึ่งชื่อ Servlet โดย interface นี้ประกอบไปด้วยฟังก์ชันสองฟังก์ชันดัง code snippet ข้างล่างนี้

```
interface Servlet {
 public void init();
   public void service(); }
```

ถ้าเราต้องการสร้างคลาส ๆ หนึ่งชื่อ MyServlet โดยคลาสดังกล่าว implement Servlet interface คลาส MyServlet อาจจะเป็นอย่างข้างล่างนี้

```
class MyServlet implements Servlet {
  public void init() {
    System.out.println("Calling MyServlet init()...");
  }

  public void service() {
    System.out.println("I'm MyServlet");
  }
}
```

เราจะสังเกตเห็นว่าคลาส MyServlet ทำการ implement ฟังก์ชั่น init() และ service() ซึ่งเป็นฟังก์ชันที่ถูก define อยู่ใน Servlet interface

Servlet Engine ใช้ประโยชน์จากคอนเซ็ปของ interface ในการโหลด Servlet ต่าง ๆ ในช่วง Runtime โดย Servlet ต่าง ๆ จะถูกโหลดและ cast กลับไปอยู่ในรูปของ interface แทน ถ้าดูจากตัวอย่างของเราหลังจาก MyServlet ถูก cast กลับไปเป็น instance ของ Servlet interface แล้วทาง Servlet Engine ก็จะสามารถเรียกฟังก์ชันอะไรก็ได้ที่ถูก define อยู่ใน Servlet interface (ในที่นี้ก็คือ init() และ service())

ตัวอย่าง code ง่าย ๆ ที่ Servlet Engine ใช้ในการโหลด Servlet ในช่วง Runtime อาจจะเป็นอย่างข้างล่างนี้

```
public class SimpleServletEngine {
  public static void main(String args[]) throws Exception {
   if (args.length <= 0) {
    System.out.println("Usage: java SimpleServletEngine javaClassName");
    System.exit(0);
   }
   System.err.println("loading class " + args[0] + "...");
   Class cls = Class.forName(args[0]);
   Servlet servlet = (Servlet) cls.newInstance(); // casting
   servlet.init();
```

```
    servlet.service();
  }
}
```

ตัว Servlet ที่ Servlet Engine โหลดจะถูกอ้างอิงมาจาก argument ที่ชื่อ javaClassName โดย arguement นี้จะถูกส่งผ่านไปให้ Class.forName() เพื่อเรียก Class object ที่จะใช้สร้าง Servlet instance ขึ้นมาโดยใช้คำสั่ง Class.newInstance() ซึ่งหลังจากนั้น Servlet instance ที่ได้จะถูก cast กลับให้กลายเป็น instance ของ Servlet interface เพื่อ Servlet Engine จะได้ใช้ในการเรียกฟังก์ชั่น init() และ service()

อาจพูดง่าย ๆ ว่าหลักการทำงานของ Servlet Engine ก็คือการโหลดคลาสใดคลาสหนึ่งขึ้นมาโดยคลาสนั้นจะต้อง implement ตัว interface ที่ชื่อ Servlet ซึ่งทาง Servlet Engine ทราบอยู่แล้วว่ามีฟังก์ชันอะไรประกอบอยู่ใน Servlet interface บ้างโดยถ้า Service Engine ทำการ cast คลาสที่ implement Servlet interface (ในที่นี้ก็คือ MyServlet) กลับไปเป็น instace ของ Servlet interface แล้วตัว Servlet Engine ก็จะสามารถเรียกฟังก์ชันที่ชื่อ init() และ service() ที่ define อยู่ใน Servlet interface จากคลาสดังกล่าวได้

ในกรณีที่เราไม่ต้องการ implement ฟังก์ชันทุกฟังก์ชันที่ถูก define อยู่ใน Servlet interface เราก็อาจจะสร้างคลาสที่เป็น abstract ขึ้นมาคลาสหนึ่งก่อนโดยคลาสนี้จะทำการ implement ฟังก์ชันบางฟังก์ชันที่อยู่ใน Servlet interface แล้วให้ subclass ของ abstract คลาสนี้ทำการ implement ฟังก์ชันที่เหลือ ยกตัวอย่างเช่น

```
abstract class HttpServlet implements Servlet {
  public void init() {
   System.out.println("Calling HttpServlet init()...");
  }
 // not implemented yet
  public abstract void service();
}
```

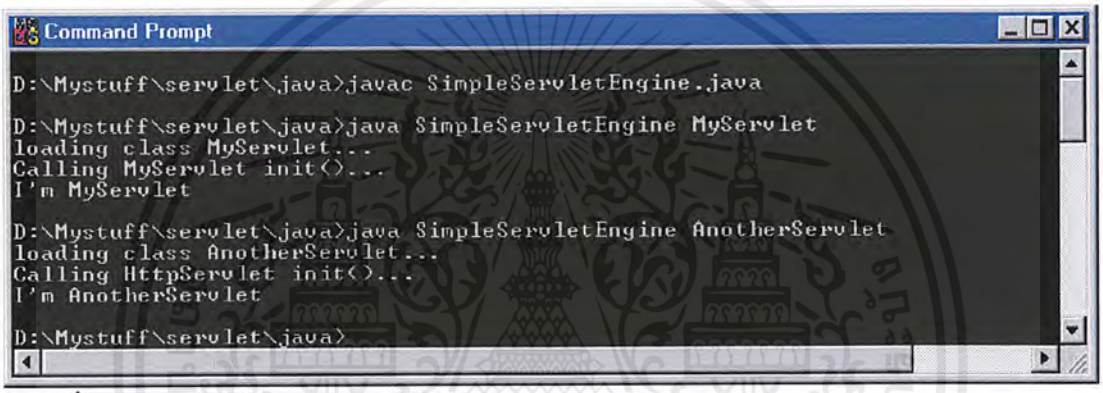
ถ้าเราต้องการสร้าง Servlet ขึ้นมาโดยจะ implement แค่ฟังก์ชัน service() อย่างเดียว เราก็เพียง subclass คลาส HttpServlet เท่านั้นโดยคลาส Servlet อาจจะเป็นดังตัวอย่างข้างล่างนี้

```
class AnotherServlet extends HttpServlet {
  public void service() {
```

```
    System.out.println("I'm AnotherServlet");
  }
}
```

เราจะสังเกตเห็นว่าคลาส AnotherServlet จะไม่ต้องทำการ implement ฟังก์ชัน init() เพราะตัว parent คลาสของมันซึ่งก็คือ HttpServlet ได้ทำการ implement ไปแล้ว

ในการโหลดคลาส AnotherServlet เข้าไปยัง Servlet Engine ก็ยังใช้หลักการเดียวกันกับคลาส MyServlet ข้างต้นคือโหลดคลาส AnotherServlet แล้ว cast กลับให้กลายเป็น instance ของ Servlet interface ซึ่งผลจากการรันโปรแกรม SimpleServletEngine โดยโหลด Servlet (หรือคลาส)ที่ชื่อ MyServlet และ AnotherServlet ก็จะออกมาเป็นดังรูป



ภาพที่ 2.5 แสดงการโหลด Servlet ต่าง ๆ ของ SimpleServletEngine

จุดสำคัญของ Servlet จริง ๆ อยู่ที่ interface ที่ชื่อ javax.servlet.Servlet (javax คือ standard extention package ของ Sun ที่อยู่นอกเหนือจาก core API ที่อยู่ใน JDK ตัว Servlet ที่เราพูดถึงกันอยู่นี้ก็เป็นหนึ่งใน standard extension ของ Sun เช่นเดียวกัน ดังนั้น API ทุกอย่างที่เกี่ยวกับ Servlet ก็จะเป็น javax.servlet.XXX.YYY.ZZZ) โดยทุก Servlet ที่ถูกเขียนขึ้นจะต้องทำการ implement ตัว interface นี้ไม่ทางตรงก็ทางอ้อม (ทางตรงก็คือการ implement ตัว interface นี้เลย ส่วนทางอ้อมก็คือการให้ Servlet ทำการ subclass คลาสบางคลาสที่ได้ทำการ implement ตัว interface นี้ไว้แล้ว) เหตุผลว่าทำไมเราต้อง implement ตัว interface นี้เพราะว่าเมื่อไรก็ตามที่มี request จาก client เข้ามายัง Servlet Engine ตัว Servlet Engine จะทำการหา Servlet ที่ request ดังกล่าวอ้างถึง หลังจากนั้น Servlet Engine จะทำการเรียกฟังก์ชันต่าง ๆ ที่อยู่ใน Servlet เพื่อทำการประมวลผล request ของ client โดยฟังก์ชันที่ Servlet Engine จะทำการเรียกก็คือฟังก์ชันที่ Servlet ได้ทำการ implement ซึ่งเป็นฟังก์ชันที่ถูก define อยู่ใน javax.servlet.Servlet interface นั่นเอง

การที่ Servlet Engine เรียกฟังก์ชันที่ถูก define อยู่ใน interface นี้ เหตุผลง่าย ๆ ก็คือ Servlet เป็นส่วนที่ถูกโหลดเข้าไปใน Servlet Engine ในช่วง Runtime ตัว Servlet Engine เองไม่สามารถทราบได้ว่า Servlet ต่าง ๆ มีฟังก์ชันอะไรประกอบอยู่บ้างนอกเสียจากว่า Servlet นั้นได้ทำการ implement ฟังก์ชันที่เป็นมาตรฐานที่ Servlet Engine รับรู้

อย่างไรก็ตามเราสามารถให้ Servlet Engine เรียกฟังก์ชันอื่น ๆ ที่อยู่ใน Servlet ได้ซึ่งวิธีการก็คือการใส่ฟังก์ชันดังกล่าวเข้าไปในส่วน implementation ของฟังก์ชันต่าง ๆ ที่ถูก define อยู่ใน javax.servletServlet interface นั่นเอง

ดังนั้นหลักการง่าย ๆ ในการสร้าง Servlet ก็คือการสร้างคลาสขึ้นมาคลาสหนึ่งโดยคลาสนั้นจะต้องทำการ implement ตัว interface ที่ชื่อ javax.servlet.Servlet เพียงเท่านี้เราก็ได้ Servlet เป็นของตัวเองแล้ว และเพื่อจะอำนวยความสะดวกให้กับนักพัฒนา ทางจาวาจึงได้มีการสร้างคลาสพื้นฐานที่ได้ทำการ implement ตัว javax.servlet.Servlet interface ขึ้นมาสองคลาส คือ คลาส javax.servlet.GenericServlet และคลาส javax.servlet.http.HttpServlet (ซึ่งเป็น subclass ของ javax.servlet.GenericServlet อีกทีหนึ่ง) ดังนั้นสิ่งที่นักพัฒนาจะต้องทำก็คือ การ subclass คลาสใดคลาสหนึ่งในสองคลาสนี้แล้ว override ฟังก์ชันที่ต้องการซึ่งโดยทั่วไปก็คือฟังก์ชันที่ชื่อ service() นั่นเอง

2.3.4 การเขียน Servlet

โปรแกรมแรกที่เรามักจะเขียนก็คือโปรแกรมที่พิมพ์คำว่า Hello World ออกมาดังนั้นในการเขียน Servlet เราก็จะทำเช่นเดียวกัน โดยโปรแกรม Hello World ที่เขียนแบบ Servlet จะเป็นดังตัวอย่างข้างล่างนี้ (HelloWorld.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends GenericServlet {

  public void service(ServletRequest request, ServletResponse response)
    throws IOException, ServletException
    {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
```

```
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

คลาส HelloWorld เป็นตัวอย่างพื้นฐานที่อธิบายหลักการทำงานของ Servlet คลาสนี้เป็นคลาสที่subclass คลาส GenericServlet โดยทำการ override ฟังก์ชัน public void service (ServletRequest request , ServletResponse response)... ซึ่งเป็นฟังก์ชันที่อยู่ใน Servlet interface ฟังก์ชันนี้เป็นฟังก์ชันที่สำคัญที่สุดสำหรับทุก Servlet เพราะว่าเมื่อใดก็ตามที่มี request เข้ามายัง Servlet ฟังก์ชัน service ที่อยู่ใน Servlet ดังกล่าวจะถูกเรียกโดย Servlet Engine เพื่อทำการประมวลผล request ที่เข้ามาเสมอ

GenericServlet เป็น abstract คลาสที่ทำการ implement ตัว Servlet interface โดยทิ้งฟังก์ชัน service ให้เป็น abstract (ไม่มีส่วนที่เป็น implementation) ซึ่งทางคลาสที่เป็น subclass ของ GenericServlet จะต้องมีหน้าที่ในการ implement ฟังก์ชันนี้ยกตัวอย่างเช่นคลาส HelloWorld เป็นต้น

มาดู code ที่อยู่ใน HelloWorld กันบ้าง ในสามบรรทัดแรกเราจะเห็นว่าจะมีสาม package ที่ถูก import เข้ามาซึ่งก็คือ java.io.* , javax.servlet.* และ javax.servlet.http.* เราเรียก package java.io เข้ามาเพราะว่าเราต้องการใช้คลาส IOException และ คลาส PrintWriter ส่วน package javax.servlet และ javax.servlet.http เราจะใช้สำหรับเรียกคลาสที่เกี่ยวข้องกับ Servlet ทั้งหมด และมาดู code ที่อยู่ในฟังก์ชัน service บ้าง ตัวฟังก์ชัน service จะมี argument อยู่ด้วยกันสองตัวคือ ServletRequest และ ServletResponse โดยเมื่อใดก็ตามที่ฟังก์ชัน service ถูกเรียกโดย Servlet Engine ตัว Servlet Engine จะทำการส่งผ่านออปเจ็กมายัง argument เหล่านี้ซึ่งก็คือ ServletRequest และ ServletResponse oblect นั่นเอง ServletResponse object เป็นตัวที่เก็บรายละเอียดของ Request ที่ส่งมาโดย client ทั้งหมดโดย Servlet สามารถใช้ object นี้เพื่ออ่าน parameters ต่าง ๆ ที่อยู่ใน Request เข้ามาประมวลผลได้ หลังจากที่ Servlet ทำการประมวลผลเสร็จแล้ว Servlet ก็สามารถส่งผลที่ได้กลับออกไปให้ Client ได้โดยใช้

ServletResponse object อย่างไรก็ตามฟังก์ชัน service ที่อยู่ใน Hello World Servlet ของเราไม่ได้มีการประมวลผลใด ๆ เพียงแต่จะส่ง HTML Stream ที่เขียนคำว่า "Hello World" กลับไปให้ Client เท่านั้น

ในการส่งผลต่าง ๆ กลับไปให้ Client ขั้นตอนแรกที่ Servlet ต้องทำคือการเซ็ต Header โดยในตัวอย่างของเรา Header ที่ถูกส่งกลับไปก็คือ Content Type (MIME Type) Header ซึ่งเป็นตัวบอก Client ว่า Stream ที่กำลังจะได้รับเป็น Stream ชนิดไหน (ให้นึกถึง MIME Type เป็นลักษณะคล้าย ๆ กับไฟล์ extension) ยกตัวอย่างเช่น text/html จะบอกว่า Stream ที่กำลังส่งออกไปเป็น HTML Stream หรือ image/gif จะบอกว่า Stream ที่ส่งออกไปเป็น Stream ของ gif image เป็นต้น ตัว Servlet สามารถทำการเซ็ต Content Type ได้โดยใช้ฟังก์ชัน setContentType () ซึ่งเป็นฟังก์ชันที่อยู่ใน ServletResponse นั่นเอง ในการส่ง Stream ต่าง ๆ ไปให้ Client ตัว Servlet สามารถส่ง Stream ออกไปได้สองแบบคือ text Stream และ byte Stream ในกรณีนี้ จะส่ง Stream ออกไปเป็นแบบ text Stream ดังนั้นคลาสที่เราจะใช้ในการเขียนลงไปที่ ServletResponse ก็คือคลาส PrintWriter นั่นเอง

ในการเรียกใช้คลาส PrintWriter สำหรับเขียนลงไปที่ ServletResponse เราก็สามารถทำได้ง่าย ๆ โดยการเรียกฟังก์ชัน ServletResponse.getWriter() ซึ่งจะ return ตัว PrintWriter instance ที่จะใช้ในการเขียน output ต่าง ๆ ลงไปยังตัว ServletResponse ที่เราใช้สำหรับเรียก instance ดังกล่าวขึ้นมา ท้ายสุด HelloWorld Servlet จะใช้ PrintWriter ทำการพิมพ์ text ต่าง ๆ ที่ใช้ในการสร้าง HTML Output Stream ที่เขียนคำว่า "Hello World" ออกไปให้ Client

2.3.5 การรัน Servlet

Servlet เป็นคอนเซ็ปหนึ่งในจาวาที่เราเขียนขึ้นเพื่อใช้งานสำหรับ Server Side Application ถ้าใครเคยเขียน application โดยใช้จาวามาก่อนจะเห็นว่าตัว entry point ของ application จะเป็น static ฟังก์ชันที่ชื่อ main() ตัว Servlet เองจะมีลักษณะคล้าย ๆ Applet ตรงที่ว่าในการรัน ฟังก์ชันที่ใช้เป็น entry point จะไม่ใช่ฟังก์ชัน main() แต่จะกลายเป็นฟังก์ชันอื่นที่ถูก define ไว้ใช้สำหรับตัวมันเองโดยเฉพาะยกตัวอย่างเช่นใน Applet ฟังก์ชันที่ถูกใช้เป็น entry point ก็จะเป็นฟังก์ชันพวก init() , start() ซึ่งสำหรับ Servlet แล้วฟังก์ชันที่ถูกใช้เป็น entry point ก็คือ service() ซึ่งก็คือฟังก์ชันที่ถูก define อยู่ใน javax.servlet.Servlet interface นั่นเอง ในการรัน Sevlet สิ่งที่เราต้องการมีดังต่อไปนี้คือ

- JDK สำหรับคนที่ใช้ Window , Sun หรือ Linux

- Servlet API Package and Servlet Engine

  Package นี้เป็นตัวที่เก็บ Servlet API ทั้งหมดไว้ซึ่งก็คือ javax.servlet.* และ javax.servlet.http.* package

Servlet Engine ที่จะยกตัวอย่างในการรัน Servlet จะเป็น Servlet Engine ที่มาจาก Apache Group ซึ่งถือว่าเป็น Servlet Engine และ JSP (Java Server Pages) Reference Implementation ของ Sun โดยมี code name ที่ชื่อว่า Tomcat

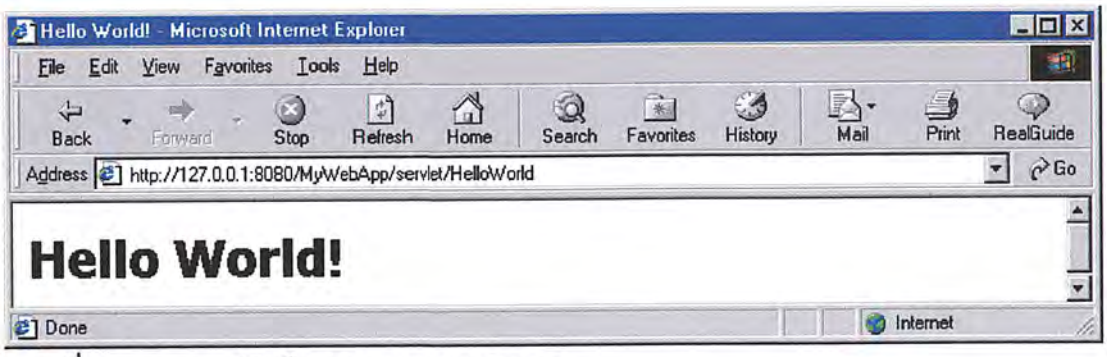
สมมุติว่าเราต้องการรัน HelloWorld Servlet ของเราที่ Context path ที่ชื่อ MyWebApp โดยชี้ไปที่ไดเรคทรอรี่ที่ชื่อ D:\MyWebApplication\ เราก็สามารถทำการแอท Context path นี้เข้าไปใน Tomcat ได้โดยการเปิดไฟล์ที่ชื่อ server.xml ซึ่งจะอยู่ในไดเรคทรอรี่ conf ของ Tomcat Home directory เสร็จแล้วให้ทำการเพิ่มบรรทัดข้างล่างนี้เข้าไปในส่วนของของ <Context path> tag

```
<Context path="/MyWebApp" docBase="D:\MyWebApplication\" debug="0"
reloadable="true"/>
```

จากนั้นให้ทำการสร้าง subdirectory ที่ชื่อ WEB-INF ขึ้นมาใต้ D:\MyWebApplication\ ไดเรคทรอรี่ เสร็จแล้วให้สร้างอีก subdirectory ที่ชื่อ classes ขึ้นมาใต้ WEB-INF อีกทีหนึ่งโดย subdirectory classes นี้จะเป็นที่ ๆ เราใช้เก็บคลาส Servlet ทั้งหมด ซึ่งท้ายสุดไฟล์ HelloWorld.java ก็จะอยู่ใน directory stucture ดังนี้

D:\MyWebApplication\WEB-INF\classes\HelloWorld.java

จากนั้นให้ทำการ compile ไฟล์ HelloWorld เหมือนปกติ โดยถ้าคอมไฟล์นี้ไม่ได้ ให้เช็คดูว่า Servlet API ถูกรวมอยู่ใน CLASSPATH แล้วหรือยัง เสร็จแล้วให้ทำการ start ตัว Tomcat แล้ว พิมพ์ http://127.0.0.1:8080/MyWebApp/servlet/HelloWorld ก็จะได้ผลดังรูปข้างล่าง



ภาพที่ 2.6 แสดงผลลัพธ์ของ HelloWorld Servlet

ตัวอย่างที่สอง ตัวอย่างนี้เป็น HelloWorld อีกแบบแต่แทนที่คลาส Servlet ของเราจะทำการ subclass คลาส GenericServlet เราจะทำการ subclass คลาส HttpServlet ซึ่งเป็น subclass ของคลาส GenericServlet อีกทีหนึ่งแทน คลาส GenericServlet และคลาส HttpServlet จะต่างกันตรงที่ว่าคลาส GenericServlet จะเป็นคลาสที่ไม่เจาะจงโปรโตคอลใดโปรโตคอลหนึ่งซึ่งเหมาะสำหรับเป็น base class สำหรับ Servlet ที่ต้องการ implement ตัว Servlet ที่ใช้สำหรับโปรโตคอลจำเพาะเจาะจงขึ้นมาเองยกตัวอย่างเช่น Servlet สำหรับ RMI, Servlet สำหรับ IIOP (CORBA) หรือแม้กระทั่ง Servlet สำหรับ Http โปรโตคอลซึ่งก็คือ HttpServlet ที่เรากำลังพูดถึงอยู่นั่นเอง คลาส HelloWorld แบบที่สองของเราก็จะเป็นอย่างด้านล่างนี้

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld2 extends HttpServlet {
public void service(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<body>");
out.println("<head>");
out.println("<title>Hello World2</title>");
out.println("</head>");
out.println("<body>");
```

```
out.println("<h1>Hello World2</h1>");
out.println("</body>");
out.println("</html>");
}
}
```

ถ้าทำการรันก็จะได้ผลดังนี้

ภาพที่ 2.7 แสดงผลลัพธ์ของ HelloWorld2 Servlet

จะสังเกตเห็นว่าฟังก์ชันที่เราทำการ override ก็คือฟังก์ชัน service แต่ต่างกันตรงที่ว่าฟังก์ชันนี้ไม่ใช่ฟังก์ชัน service(ServletRequest req , ServletResponse res)แต่เป็น service (HttpServletRequest req , HttpServletResponse res) แทนซึ่งเป็นฟังก์ชันที่เจาะจงสำหรับ Http โปรโตคอลโดยเฉพาะ ซึ่ง code ที่อยู่ใน HttpServlet จะเห็นว่าเป็นลักษณะคล้าย ๆ ด้านล่างนี้

```
public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
HttpServletRequest request;
HttpServletResponse response;

try {
request = (HttpServletRequest) req;
response = (HttpServletResponse) res;
} catch (ClassCastException e) {
throw new ServletException("non-HTTP request or response");
}
```

```
service(request, response); // service(HttpServletRequest res, HttpServletResponse res)
...
}
```

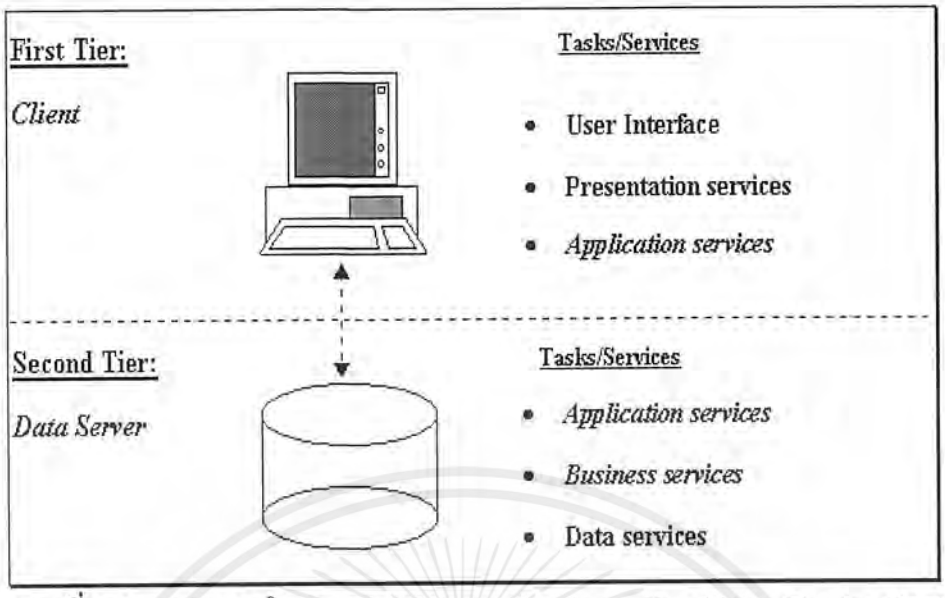
จะเห็นว่าคลาส HttpServlet ก็ยังคงทำการ implement ฟังก์ชัน service (ServletRequest req ,...) ซึ่งเป็นฟังก์ชันที่อยู่ใน javax.servlet.Servlet interface อยู่เพียงแต่ว่าเมื่อใดก็ตามที่มีการเรียกฟังก์ชันนี้โดย Servlet Engine ทาง HttpServlet จะทำการ cast ตัว ServletRequest และ ServletResponse ให้กลายเป็น HttpServletRequest และ HttpServletReponse ตามลำดับ หลังจากนั้นจะส่งผ่าน object สองตัวนี้เข้าไปยัง service (request , response) ซึ่งก็คือฟังก์ชัน service(HttpServletRequest req , HttpServletResponse res) อีกต่อหนึ่งโดยโค๊ดของเราที่พิพม์คำว่า HelloWorld ที่อยู่ในฟังก์ชันนี้ก็จะถูกเรียกโดย Servlet Engine ในท้ายสุด

2.4 การทำงานแบบ Client/Server

สถาปัตยกรรมของ application แบบ Client / Server จะแบ่งการประมวลผลออกเป็น 2 โปรแกรม โดยทั่วไปจะทำงานบนเครื่องสองเครื่องขึ้นไป application ที่ทำงานกับฐานข้อมูลแบบ Client / Server จะรับผิดชอบการเก็บข้อมูล , การประมวลผลข้อมูล , การโอนย้ายข้อมูล และไปแสดงผลที่อื่น เครื่อง Server จะเก็บรวบรวมข้อมูลไว้ ส่วนเครื่อง Client จะประมวลผลข้อมูลที่ได้มา หรือสร้างเป็นข้อมูลใหม่ วิธีการทำงานโดยใช้ สถาปัตยกรรม แบบ Client / Server นี้ทำให้สามารถติดต่อใช้งานข้อมูลได้จากผู้ใช้หลายแห่ง

2.4.1 การทำงาน Client/Server แบบ Two - Tier (Two - Tier Application)

รูปแบบธรรมดาทั่วไปของสถาปัตยกรรม Client / Server เป็น two - tier ซึ่งมาจากการแบ่งการทำงานของ application ออกเป็นส่วน Client / Server ยอมรับการติดต่อจากหลาย ๆ ที่เข้าสู่ส่วนให้บริการซึ่งเก็บข้อมูลไว้ ส่วนแสดงผลจะอยู่ที่ Client และส่วนเก็บรวบรวมข้อมูล จะอยู่ที่ Server application ทั่วไป ส่วนใหญ่บนอินเตอร์เนต เช่น e-mail , telnet , ftp , gopher หรือ web เป็น application แบบ 2 ระดับซึ่งทำงานโดยไม่ต้อง ประมวลผลข้อมูลขนาดใหญ่ โดยทั่วไปจะทำงานติดต่อใช้ข้อมูลภายใน



ภาพที่ 2.8 แสดงการทำงาน Client/Server แบบ Two - Tier (Two - Tier Application)

ข้อดี ของ application แบบ Two – Tier คือ เป็น application ง่าย ๆ ธรรมดา ที่ไม่ต้องการ การดูแลบำรุงรักษามาก สามารถพิจารณาเลือกใช้ว่าเหมาะกับ application แบบ two - tier หรือไม่ ควรขึ้นกับเงื่อนไขดังนี้

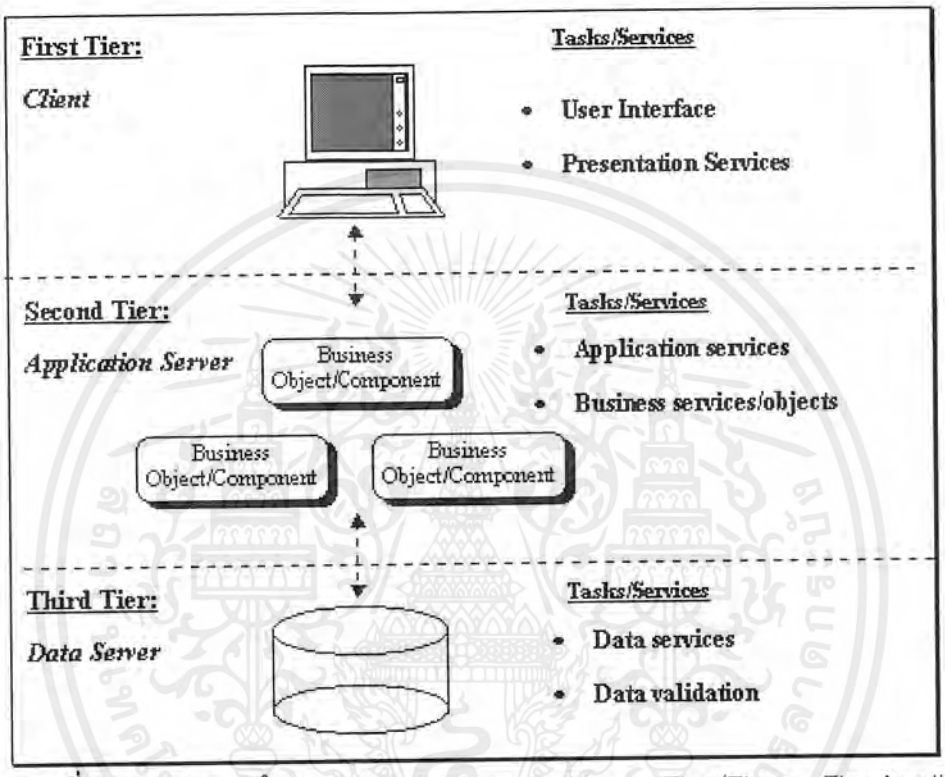
- เป็น application ที่ใช้ฐานข้อมูลเดี่ยว
- ฐานข้อมูลบรรจุอยู่ภายใน CPU เครื่องเดียว
- ฐานข้อมูลมีขนาดเดิมไม่เปลี่ยนแปลงบ่อย ๆ
- user base ไม่มีการเปลี่ยนแปลงบ่อย
- requirement ไม่มีการเปลี่ยนแปลง หรือ เปลี่ยนแปลงน้อยมาก
- application ที่สมบูรณ์แล้ว ไม่จำเป็นต้องดูแลบำรุงรักษามาก

ข้อเสียของ Two – Tier เนื่องจากความต้องการของผู้ใช้เพิ่มมากขึ้น ดังนั้นความซับซ้อนของ application จึงต้องมากตามไปด้วย จากการที่ Client มีประสิทธิภาพ และมีความซับซ้อนขึ้นเรื่อย ๆ ในขณะที่ Server มีขนาดเล็กลงเพื่อให้ราคาถูกลง และความสามารถในการจัดการกับฐานข้อมูลที่ซับซ้อนต่ำลง เช่น ในปัจจุบันเครื่องคอมพิวเตอร์เมนเฟรมได้ถูกเปลี่ยนมาใช้เครื่องคอมพิวเตอร์ขนาดเล็กจำนวนมากมาทำงานแทน และงานบางส่วนจะถูกผลักภาระไปที่ Client เพื่อเป็นการลดค่าใช้จ่าย แต่การทำเช่นนี้ทำให้เกิดปัญหา " fat client " Client ที่มีปัญหา fat client นี้เกิดจากการที่ Client ไม่สามารถรองรับขนาดของข้อมูล และงานของผู้ใช้ที่มีจำนวนมากขึ้นได้ เพราะว่างานของ Client ไม่ได้มีแค่แสดงข้อมูลให้เห็นเท่านั้น แต่ยังมีการดึงข้อมูลอื่น ๆ จำนวนมากที่ไม่เกี่ยวข้องเลยกับงานนั้น ๆ มาด้วย และในกรณีที่มีการเปลี่ยนแปลงฟังก์ชันการทำงาน

บางส่วน ผู้ใช้จำเป็นต้องมีการเปลี่ยนแปลง , ทดสอบ และแจกจ่ายโปรแกรมในส่วนของ Client ที่ได้ปรับปรุงแล้ว ไปยัง Client ทุกเครื่อง

2.4.2 การทำงาน Client/Server แบบ Three – Tier (Three - Tier Application)

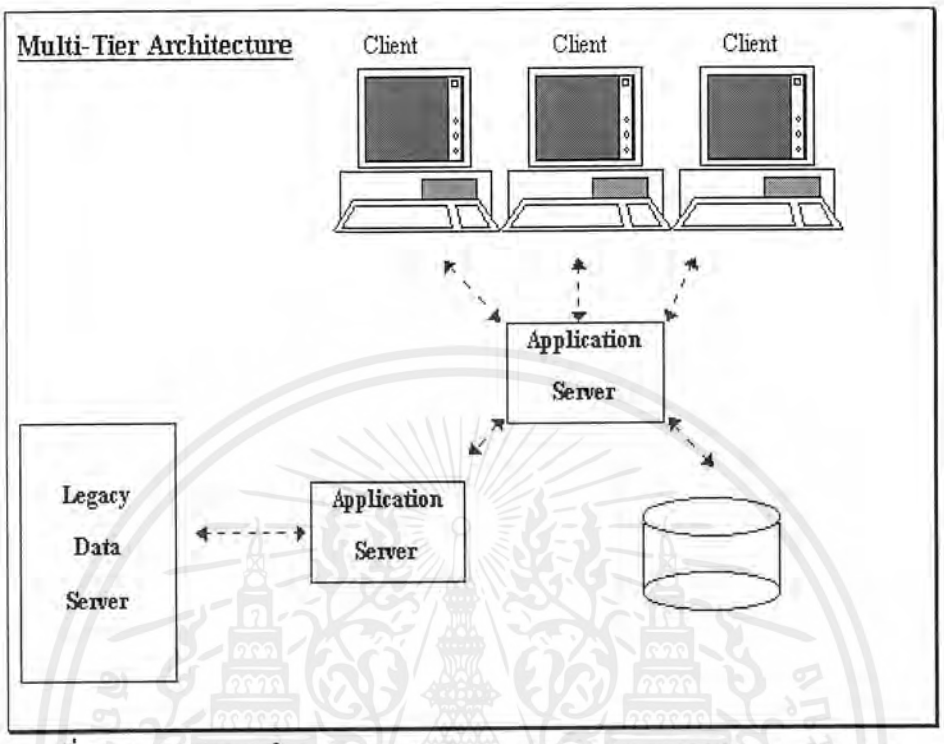
เพื่อแก้ปัญหาของ two - tier จึงเพิ่มจาก 2 tier เป็น 3 tier ดังภาพที่ 2.9



ภาพที่ 2.9 แสดงการทำงาน Client/Server แบบ Three – Tier (Three - Tier Application)

โดยในแบบ two – tier เดิม Client จะติดต่อโดยตรงกับฐานข้อมูล หากมีการเปลี่ยนแปลงใด ๆ เกิดขึ้นในฐานข้อมูล การแสดงผลทางด้าน Client จำเป็นต้องเปลี่ยนแปลงตามไปด้วยในการแก้ปัญหานี้เราจะเพิ่ม tier ใหม่เข้ามาขั้นระหว่าง Client และ Server โดย Client จะติดต่อกับ Server โดยผ่านทาง object ที่อยู่บน middle - tier จากนั้น middle - tier จะติดต่อกับ Server โดย Client จะเห็นเฉพาะ object ใน middle - tier เท่านั้นการเปลี่ยนแปลงใด ๆ จะต้องทำผ่าน middle tier เท่านั้น

2.4.3 **การทำงาน Client/Server แบบ Multi – Tier (Multi - Tiered Application)**
มีรูปแบบการทำงานดังรูปที่ 2.10



ภาพที่ 2.10 แสดงการทำงาน Client/Server แบบ Multi – Tier (Multi - Tiered Application)

โปรแกรม application โดยทั่วไปที่ใช้งานอยู่ จะประกอบด้วยส่วนที่ติดต่อกับผู้ใช้ (user Interface) สำหรับแสดงผล และเก็บรวบรวมข้อมูลเข้ามา กลุ่มของฟังก์ชั่นต่าง ๆ ที่ทำหน้าที่ประมวลผลข้อมูล และแบ่งงานต่าง ๆ รวมถึง วิธีการเก็บรักษาข้อมูล ถึงแม้ว่าฟังก์ชั่นที่ใช้ในการเก็บรักษาข้อมูล โดยทั่วไปจะทำงานอยู่ภายใต้ Server ของฐานข้อมูลส่วนกลาง บางครั้งเราเรียกรูปแบบลักษณะการทำงานแบบนี้ว่า เป็น model applicationแบบ 2 ระดับ (two-tier application model) , ซึ่งโปรแกรม application แบบเก่า จะเป็นโปรแกรมเดียวซึ่งทำงานบนเครื่องของผู้ใช้ เนื่องจากโปรแกรม application ที่ทำงานเดี่ยว ๆ นั้นมีขนาดใหญ่มากจึงพัฒนาได้ช้าและบำรุงรักษามากอีกทั้งยังใช้เนื้อที่ harddisk สูงมาก เพียงแค่มีการเปลี่ยนแปลงเล็กน้อย ก็จะต้องมีการเขียนโปรแกรมทับลงไปใหม่ , คอมไพล์ใหม่ และเนื่องจากโปรแกรม application เหล่านี้ เขียนขึ้นมาเพื่อใช้งานกับระบบที่มีลักษณะต่างกันจึงไม่สามารถที่จะเปลี่ยนไปใช้งานบนระบบที่แตกต่างไปได้วิธีการแก้ปัญหาดังกล่าวมาทำได้โดยการแบ่ง โปรแกรม application เดี่ยว ๆ นี้ ออกเป็น module ย่อย ๆ ที่ทำงานร่วมกัน การแยกส่วนที่ติดต่อกับผู้ใช้ออกมาจากฟังก์ชั่นอื่น ๆ ในโปรแกรม application ทำให้เราสามารถสร้างโปรแกรม Client เล็ก ๆ ซึ่งไม่ซับซ้อน และไม่ต้องทำงานมาก

เกินไป บนเครื่องของผู้ใช้โดยใน module นี้จะใช้ harddisk บนเครื่องผู้ใช้น้อยกว่า และสามารถพัฒนาบำรุงรักษา ได้ง่ายกว่า ตัวอย่างเช่น ส่วนที่ติดต่อกับผู้ใช้สามารถเปลี่ยนแปลงได้โดยไม่ต้องเขียนโปรแกรม application ใหม่ทั้งหมดในทางทฤษฎี ส่วน module ที่ติดต่อกับผู้ใช้นี้สามารถเขียนได้โดยใช้ ภาษาที่ต่างกันเช่น จาวา (Java) จึงทำให้ใช้ได้บนเครื่องที่แตกต่างกัน Client module ออกแบบโดยใช้ Java applet จึงไม่ต้องการเนื้อที่บน harddisk เพื่อการติดตั้ง

บทที่ 3
วิธีดำเนินการวิจัย

3.1 ความต้องการทางด้านฮาร์ดแวร์

3.1.1 เว็บเซิร์ฟเวอร์

ก่อนที่จะพัฒนาระบบงาน จะต้องให้ความสำคัญกับประสิทธิภาพของฮาร์ดแวร์และการเตรียมความพร้อมเว็บเซิร์ฟเวอร์ เพื่อให้เว็บเซิร์ฟเวอร์ทำหน้าที่ของมันอย่างที่เราต้องการ เนื่องจากเว็บเซิร์ฟเวอร์ทำหน้าที่ให้บริการต่างๆ แก่เครื่องไคลเอนต์

ตารางที่ 3.1 รายการฮาร์ดแวร์ที่จำเป็นต้องใช้สำหรับเว็บเซิร์ฟเวอร์

รายการฮาร์ดแวร์	รายละเอียด	คำแนะนำ
Processor	Pentium II 400 ขึ้นไป	
Hard Disk	มีขนาดมากกว่า 3 GB	ควรมีขนาด 5 GB เป็นอย่างต่ำ
RAM	อย่างต่ำ 64 MB	
CD-ROM	ต้องมี	
Network Interface Card	ต้องมี	

3.1.2 ไคลเอนต์

เราไม่จำเป็นต้องให้ความสำคัญกับประสิทธิภาพของฮาร์ดแวร์สำหรับเครื่องที่ทำหน้าที่เป็นไคลเอนต์ เพราะการทำงานส่วนใหญ่จะอยู่ที่เครื่องเซิร์ฟเวอร์ ถ้าเครื่องเซิร์ฟเวอร์มีประสิทธิภาพดี ก็จะทำให้การทำงานมีประสิทธิภาพดีตามไปด้วย และเครื่องที่ทำหน้าที่เป็นไคลเอนต์จะต้องเป็นเครื่องที่สามารถเชื่อมต่อเข้าสู่เครือข่ายอินเตอร์เน็ตได้ด้วย

3.2 ความต้องการทางด้านซอฟต์แวร์

3.2.1 เว็บเซิร์ฟเวอร์

ซอฟท์แวร์ที่ใช้สำหรับเครื่องที่ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ มีดังนี้

- ระบบปฏิบัติการ เช่น Windows NT Server 4.0 , Windows 2000 Server
- เว็บเซิร์ฟเวอร์ ได้แก่ซอฟต์แวร์ Web Server ทั่วไป เช่น Java Web Server 2.0
- Internet Tools เช่น Microsoft Internet Explorer 5.0-5.5 เป็นต้น

- โปรแกรมที่จำเป็นต้องใช้ คือ Jsdk 2.1 , Jdk 1.2.2
- Database Server คือ Oracle 8i

3.2.2 ไคลเอนต์

ซอฟท์แวร์ที่ใช้สำหรับเครื่องที่ทำหน้าที่เป็นไคล์เอนต์ มีดังนี้

- ระบบปฏิบัติการ เช่น Windows 98 , Windows 2000 เป็นต้น
- Internet Tools เช่น Microsoft Internet Explorer 5.0-5.5 เป็นต้น

3.3 ขั้นตอนการพัฒนาโปรแกรม

หัวข้อนี้จะอธิบายถึงขั้นตอนในการสร้างเครื่องมือที่ใช้ในการพัฒนาโปรแกรม และเพื่อให้การพัฒนาโปรแกรมนี้ทำได้โดยสะดวกขึ้น ก็จะจำลองเครื่องที่ใช้ให้มีลักษณะเป็นเครื่องเซิร์ฟเวอร์ ซึ่งมีขั้นตอนดังต่อไปนี้

1. ติดตั้ง Jdk 1.2.2 , Jsdk 2.1 และเว็บบราวเซอร์ ลงในเครื่องที่จะใช้ในการพัฒนาโปรแกรม
2. เมื่อทำการติดตั้งซอฟต์แวร์เรียบร้อยแล้ว ต้องทำการ SET PATH และ SET CLASSPATH ในไฟล์ AUTOEXEC.BATดังนี้

   SET PATH = c:\jdk 1.2.2 \bin;

   SET CLASSPATH = c:\jsdk 2.1\servlet.jar;c:\jdk 1.2.2\lib\tools.jar;
3. เราใช้ภาษาจาวาในการพัฒนาซึ่งจะเขียนโดยใช้เอดิเตอร์ตัวใดก็ได้ แต่ในที่นี้ใช้ NotePad , EditPad
4. โปรแกรมซึ่งเขียนโดยภาษาจาวาที่ได้ ให้บันทึกเป็นนามสกุล java
5. จากนั้น คอมไพล์โปรแกรมที่มีนามสกุล java ที่ MS-DOS Prompt ด้วยคำสั่ง javac ตามด้วยชื่อfile.java จะได้ file.class เช่น CreateTable.java จะได้ CreateTable.class
6. นำ file.class ที่ได้ไปไว้ใน c:\jsdk 2.1\examples\web-inf\servlets
7. เปิด Web Server โดยพิมพ์คำสั่ง c:\jsdk 2.1\startserver บน MS-DOS Prompt
8. ทดลองรันโปรแกรมที่เว็บบราวเซอร์ โดยใช้แอดเดรส

   http://127.0.0.1:8080/examples/servlet/ชื่อfile

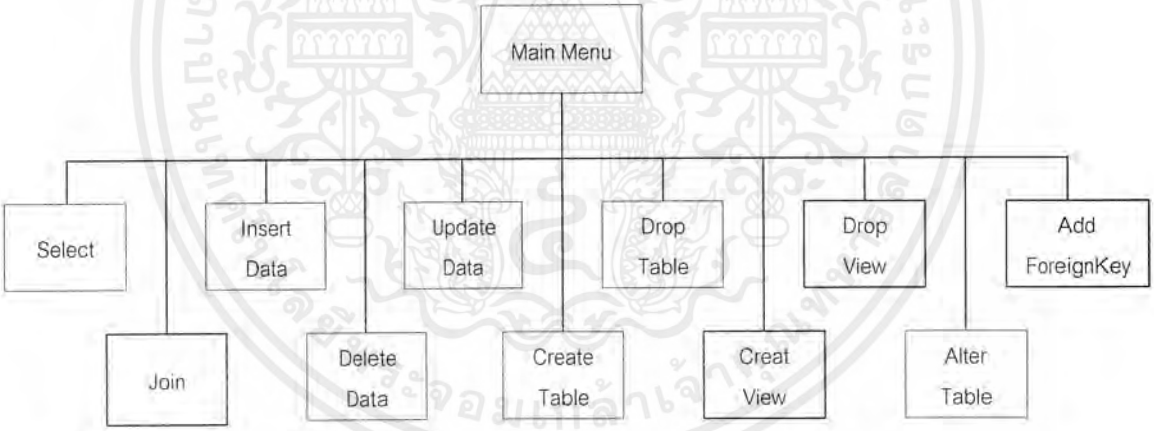
   เช่น จะทดลองรันโปรแกรมชื่อ DropTable ทำได้โดยใช้ http://127.0.0.1:8080/examples/servlet/DropTable
9. เมื่อทดสอบโปรแกรมเสร็จแล้วให้ปิด Web Server โดยพิมพ์คำสั่ง c:\jsdk 2.1\stopserver

วิธีนี้เป็นการรันโปรแกรมแบบเสมือนว่าเครื่องที่ใช้ทำการพัฒนาโปรแกรมนี้เป็นเครื่องเซิร์ฟเวอร์ แต่ในความเป็นจริงจะต้อง Upload โปรแกรมขึ้นไปไว้ที่เครื่องเว็บเซิร์ฟเวอร์ก่อน และเมื่อจะใช้งานจะต้องใช้เครื่องที่ทำการเชื่อมต่ออินเตอร์เน็ตเปิดแอดเดรสที่เก็บโปรแกรมขึ้นมา แล้วการทำงานต่างๆ ก็จะทำโดยอัตโนมัติ โดยที่มีโปรแกรมจัดการให้เรียบร้อย

3.4 แผนภาพโครงสร้างการทำงานของระบบ

ภาพที่ 3.1 แสดงโครงสร้างการทำงานของระบบ

3.5 แผนภาพหน้าจอของระบบ



ภาพที่ 3.2 แสดง Menu Diagram

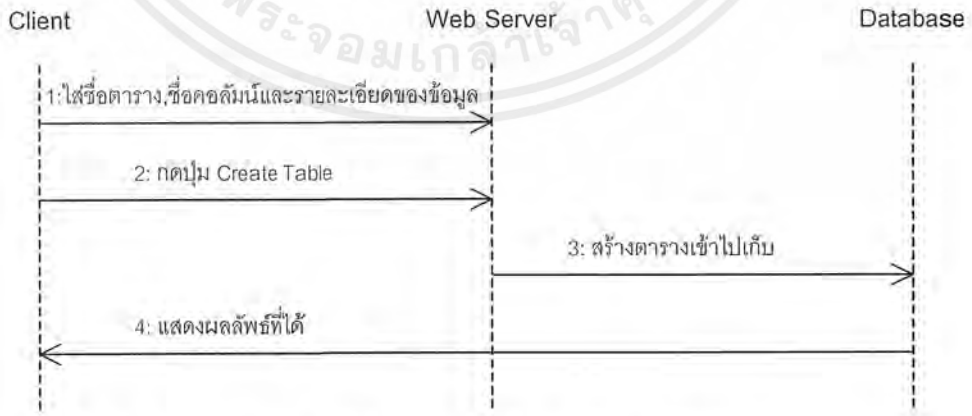
จากแผนผังแสดงถึงหน้าที่การทำงานหลัก โดยหน้าจอแรกของโปรแกรมคือ Main Menu มีรายละเอียดของฟังก์ชันต่างๆดังนี้

- Select เป็นคำสั่งสำหรับการเรียกดูข้อมูล โดยเลือกคอลัมน์ที่ต้องการดูข้อมูล
- Join เป็นคำสั่งสำหรับการดูข้อมูลที่อยู่ร่วมกัน ซึ่งจะทำการเรียกดูข้อมูลจากตารางได้มากกว่า 1 ตาราง โดยเลือกดูเฉพาะแถวที่มีค่าข้อมูลบางส่วนร่วมกันอยู่

- Insert Data เป็นคำสั่งที่ใช้เพิ่มเติมข้อมูลเข้าไปในฐานข้อมูล (เป็นการเพิ่มเรกคอร์ดในตาราง)
- Delete Data เป็นคำสั่งที่ใช้ลบแถวของข้อมูลที่สอดคล้องกับเงื่อนไขที่ระบุออกจากตาราง แต่ถ้าไม่ใส่เงื่อนไขก็จะเป็นการลบรายการทั้งหมดออกจากตารางที่ถูกเลือก
- Update Data เป็นคำสั่งเปลี่ยนแปลงแก้ไขข้อมูลใน table ซึ่งการเปลี่ยนแปลงแก้ไขควรระบุเงื่อนไข ถ้าไม่ระบุเงื่อนไขอะไรเลย การปรับปรุงจะกระทำกับทุกรายการ
- Create Table เป็นคำสั่งที่ใช้ในการสร้างตารางในระบบฐานข้อมูล
- Drop Table เป็นคำสั่งที่ใช้ลบตารางออกจากระบบ
- Create View ซึ่ง "วิว"คือเค้าร่างของระบบที่ผู้ใช้แต่ละคนเห็น จึงสามารถระบุได้ว่าให้วิวประกอบด้วยคอลัมน์ใดบ้าง เมื่อวิวได้ถูกสร้างขึ้นมาแล้ว ผู้ที่มีสิทธิใช้วิวก็จะมองเสมือนหนึ่งว่าวิวเหล่านี้ก็คือตารางนั่นเอง
- Drop View เป็นคำสั่งสำหรับการยกเลิกใช้วิวใด ๆ
- Alter Table เป็นคำสั่งที่ใช้ในการเพิ่มเติมฟิลด์ (คอลัมน์) ใหม่เข้าไปในตาราง โดยคอลัมน์ใหม่จะมีค่าเริ่มต้นเป็นค่าว่าง
- Add ForeignKey เป็นหน้าจอของคำสั่งที่ใช้ในการสร้างคีย์รองในตาราง โดยคีย์รองนั้นจะต้องเป็นคีย์หลักหรือต้องเป็นค่าไม่ซ้ำ (Unique)

ลักษณะการทำงานในระบบ ของหน้าจอต่างๆ ใน Main Menu มีตัวอย่างดังนี้

3.5.1 ลักษณะการทำงานของหน้าจอ Create Table



ภาพที่ 3.3 แสดงแผนภาพการทำงานของหน้าจอ Create Table

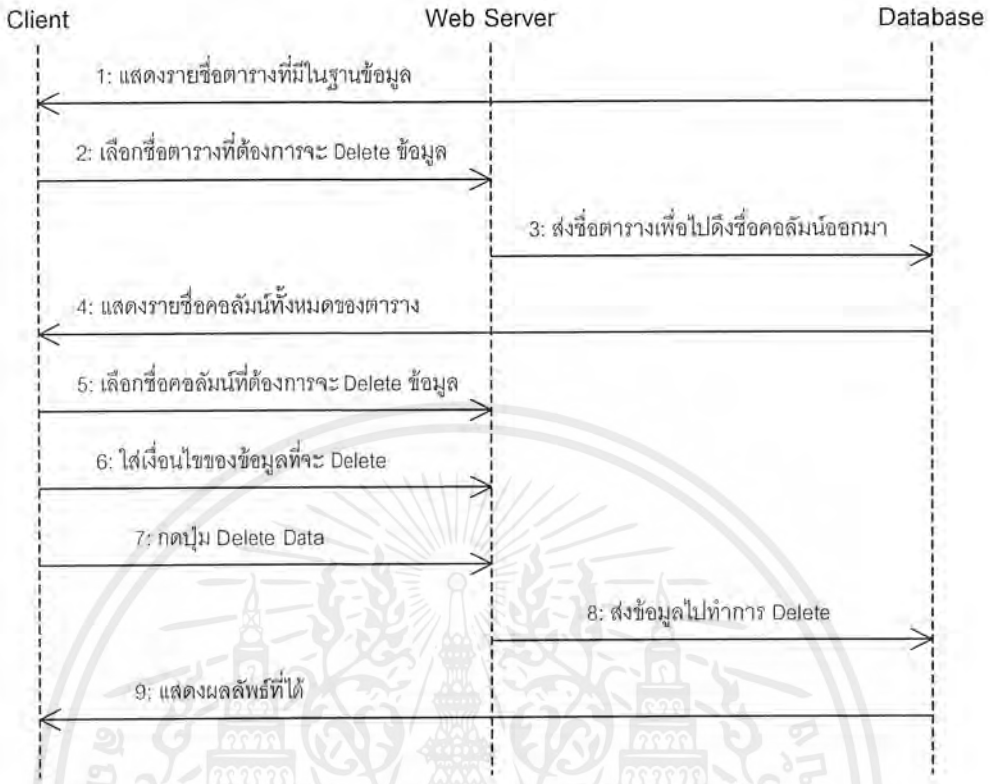
จากหน้าจอ Create Table ซึ่งเป็นไฟล์ Servlets ชื่อ CreateTable.class เมื่อใส่ชื่อตาราง , ชื่อคอลัมน์ และรายละเอียดของข้อมูลในตารางเรียบร้อยแล้ว และกดปุ่ม Create Table จะมีไฟล์ Servlets อีกไฟล์หนึ่งคือ sCreateTable .class มารองรับการทำงานแล้วทำการสร้างตารางเข้าไปในฐานข้อมูล หลังจากนั้นก็จะส่งผลลัพธ์ที่ได้มาแสดงที่หน้าจอ

3.5.2 ลักษณะการทำงานของหน้าจอ Insert Data

ภาพที่ 3.4 แสดงแผนภาพการทำงานของหน้าจอ Insert Data

จากหน้าจอ Insert Data ซึ่งเป็นไฟล์ Servlets ชื่อ Insert.class จะไปดึงชื่อตารางทั้งหมดจากฐานข้อมูลขึ้นมาให้เลือกว่าจะทำการ Insert ข้อมูลเข้าไปในตารางใด เมื่อเลือกตารางแล้วจะไปยังส่วนของ Fieldname ซึ่งจะรับชื่อตารางมาแล้วไปดึงชื่อคอลัมน์ทั้งหมดของตารางนั้นจากฐานข้อมูลมาแสดงที่หน้าจอ เมื่อเลือกคอลัมน์ที่ต้องการจะ Insert ข้อมูล และใส่รายละเอียดเรียบร้อยแล้ว เมื่อกดปุ่ม Insert Data จะมีไฟล์ Servlets ที่ชื่อ sInsert.class มารองรับการทำงานและจัดการเกี่ยวกับการ Insert ข้อมูลเข้าไปในฐานข้อมูล หลังจากนั้นก็จะส่งผลลัพธ์ที่ได้มาแสดงที่หน้าจอ

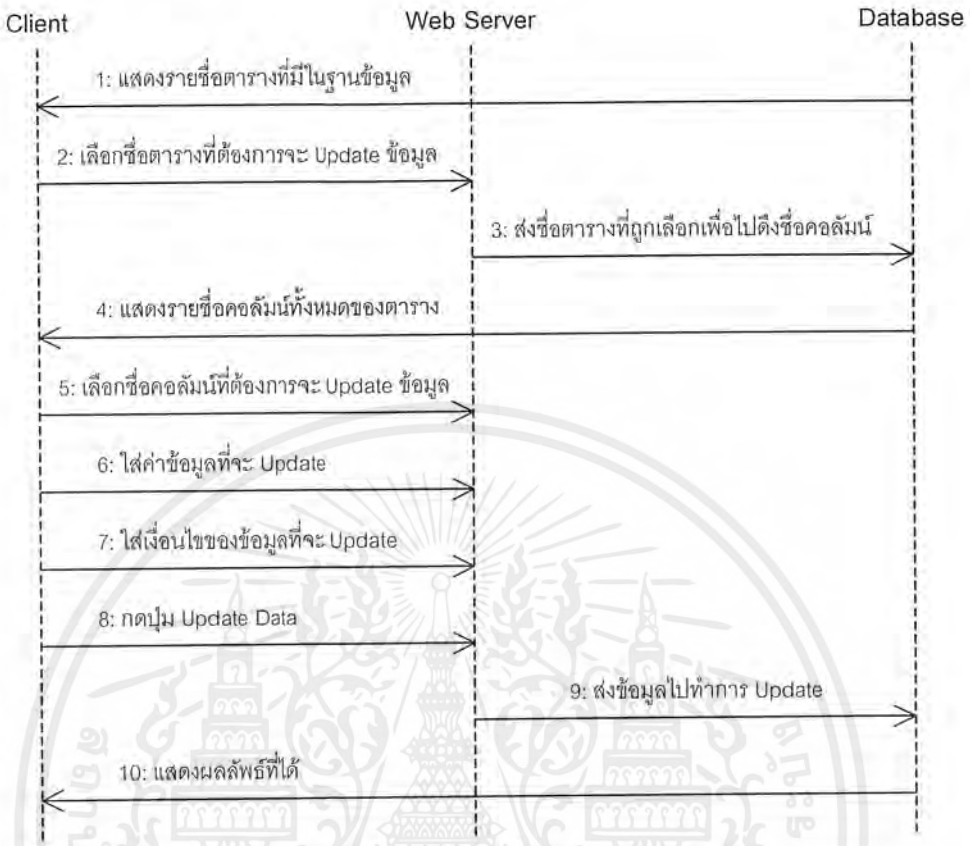
3.5.3 ลักษณะการทำงานของหน้าจอ Delete Data



ภาพที่ 3.5 แสดงแผนภาพการทำงานของหน้าจอ Delete Data

จากหน้าจอ Delete Data ซึ่งเป็นไฟล์ Servlets ชื่อ Delete.class จะไปดึงชื่อตารางทั้งหมดจากฐานข้อมูลขึ้นมาให้เลือกว่าจะทำการ Delete ข้อมูลในตารางใด เมื่อเลือกตารางได้แล้วจะไปยังส่วนของ Fieldname ซึ่งจะรับชื่อตารางมาแล้วไปดึงชื่อคอลัมน์ทั้งหมดของตารางนั้นจากฐานข้อมูลมาแสดงที่หน้าจอ เมื่อเลือกคอลัมน์ที่ต้องการจะ Delete ข้อมูล และใส่เงื่อนไขเรียบร้อยแล้ว เมื่อกดปุ่ม Delete Data จะมีไฟล์ Servlets ที่ชื่อ sDelete.class มารองรับการทำงานและจัดการเกี่ยวกับการ Delete ข้อมูลในฐานข้อมูล หลังจากนั้นก็จะส่งผลลัพธ์ที่ได้มาแสดงที่หน้าจอ

3.5.4 ลักษณะการทำงานของหน้าจอ Update Data



ภาพที่ 3.6 แสดงแผนภาพการทำงานของหน้าจอ Update Data

จากหน้าจอ Update Data ซึ่งเป็นไฟล์ Servlets ชื่อ Update.class จะไปดึงชื่อตารางทั้งหมดจากฐานข้อมูลขึ้นมาให้เลือกว่าจะทำการ Update ข้อมูลในตารางใด เมื่อเลือกตารางได้แล้วจะไปยังส่วนของ Fieldname ซึ่งจะรับชื่อตารางมาแล้วไปดึงชื่อคอลัมน์ทั้งหมดของตารางนั้นจากฐานข้อมูลมาแสดงที่หน้าจอ เมื่อเลือกคอลัมน์ที่ต้องการจะ Update ข้อมูล ใส่ค่าข้อมูลใหม่ และใส่เงื่อนไขเรียบร้อยแล้ว เมื่อกดปุ่ม Update Data จะมีไฟล์ Servlets ที่ชื่อ sUpdate.class มารองรับการทำงานและจัดการเกี่ยวกับการ Update ข้อมูลในฐานข้อมูล หลังจากนั้นก็จะส่งผลลัพธ์ที่ได้มาแสดงที่หน้าจอ

3.6 สิ่งที่ได้จากการพัฒนาโปรแกรม

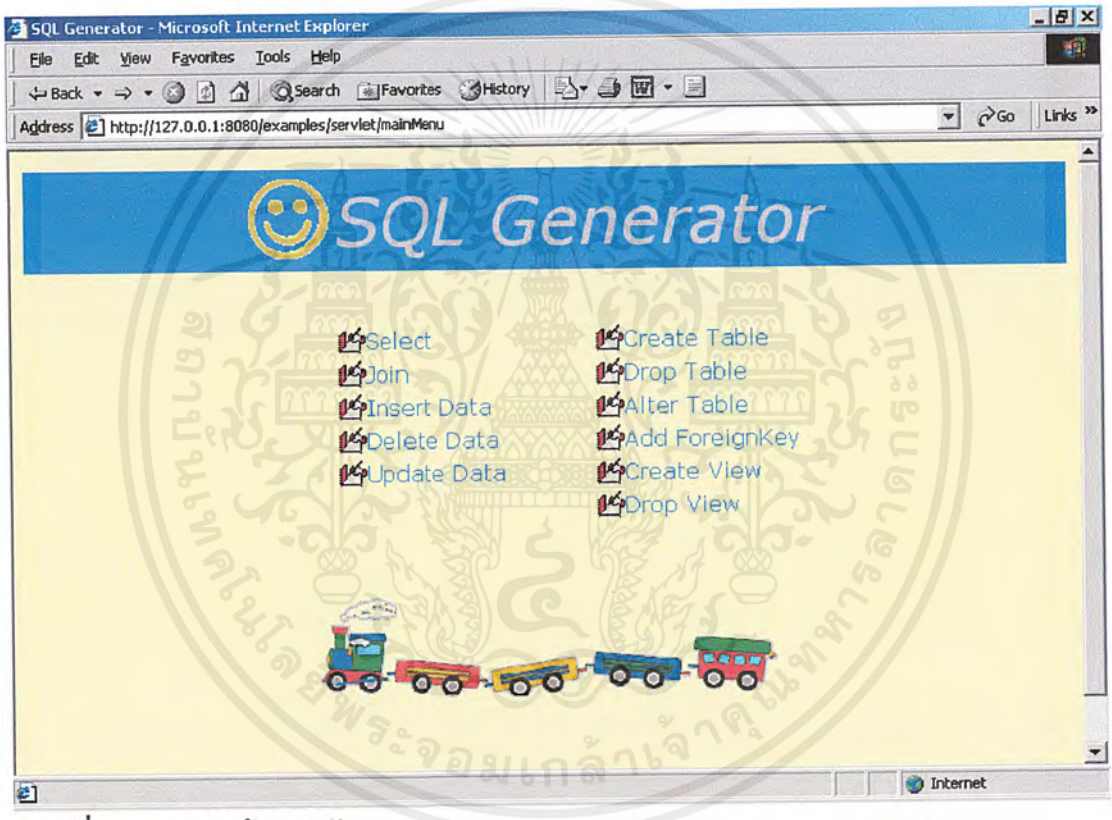
เมื่อสร้างแบบจำลองการใช้ภาษา SQL นี้เรียบร้อยแล้ว จะทำให้ได้เว็บแอปพลิเคชันที่สามารถสร้างตาราง เข้าไปในฐานข้อมูล ทำการ เพิ่ม ลบ เปลี่ยนแปลงข้อมูลที่อยู่ในฐานข้อมูล และยังสามารถดึงข้อมูลออกมาแสดงได้อีกด้วย

บทที่ 4
ผลการศึกษาและดำเนินงาน

จากการศึกษาและดำเนินงานพัฒนาแบบจำลองการใช้ภาษา SQL ได้ผลออกมาเป็นหน้าจอการทำงานต่างๆ ดังนี้

4.1 หน้าจอการทำงานหลัก (Main Menu)

เมื่อเริ่มต้นการใช้งาน ก็จะได้หน้าจอหลักดังภาพที่ 4.1 เป็นหน้าจอแรกที่จะพบเมื่อใช้ระบบ



ภาพที่ 4.1 แสดงหน้าจอหลัก

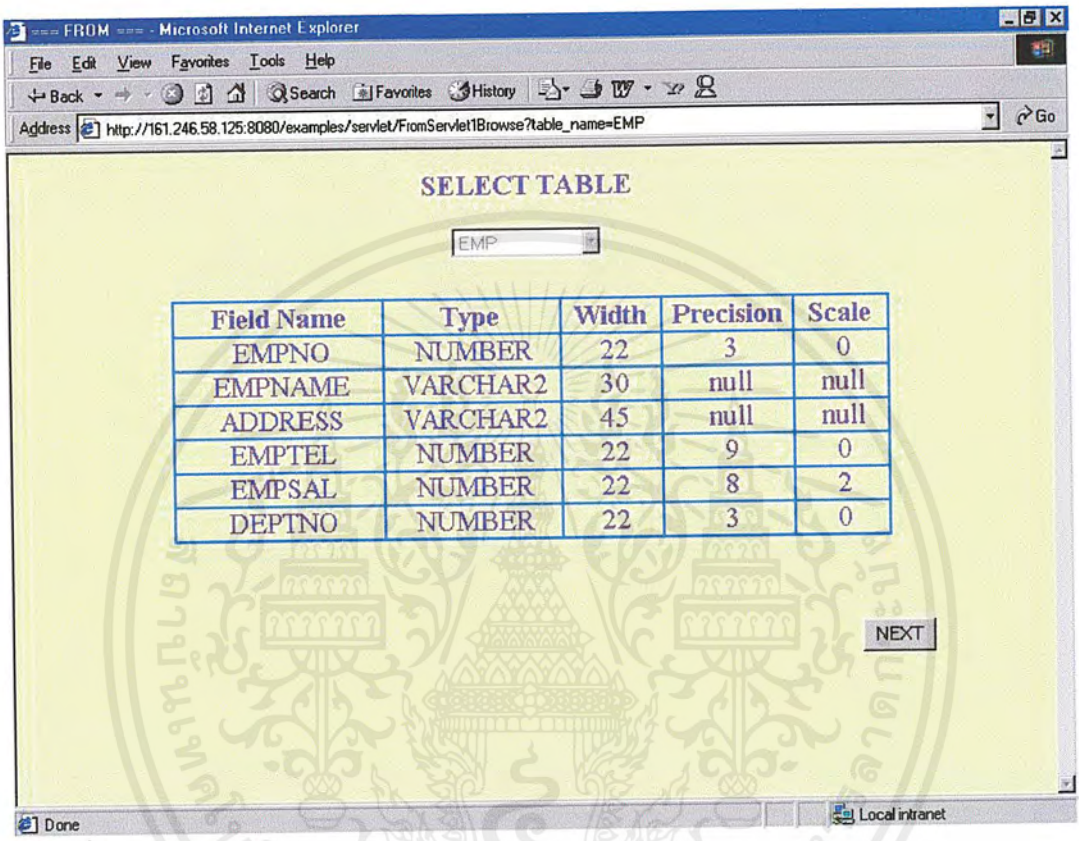
หน้าจอนี้สามารถเลือกฟังก์ชันในการจัดการกับ Data ,View และ Table โดยมีอยู่ 11 ฟังก์ชัน คือ Select , Join , Insert Data, Delete Data , Update Data , Create Table , Drop Table , Alter Table , Add ForeignKey , Create View และ Drop View โดยผู้ใช้ต้องการเลือกใช้คำสั่งใดให้คลิกเลือกที่ข้อความนั้นๆ

4.2 ฟังก์ชันการทำงาน Select

เป็นคำสั่งสำหรับการเรียกดูข้อมูล

4.2.1 หน้าจอ From

เมื่อผู้ใช้คลิกเลือกฟังก์ชัน Select ก็จะปรากฏหน้าจอ From ดังภาพที่ 4.2

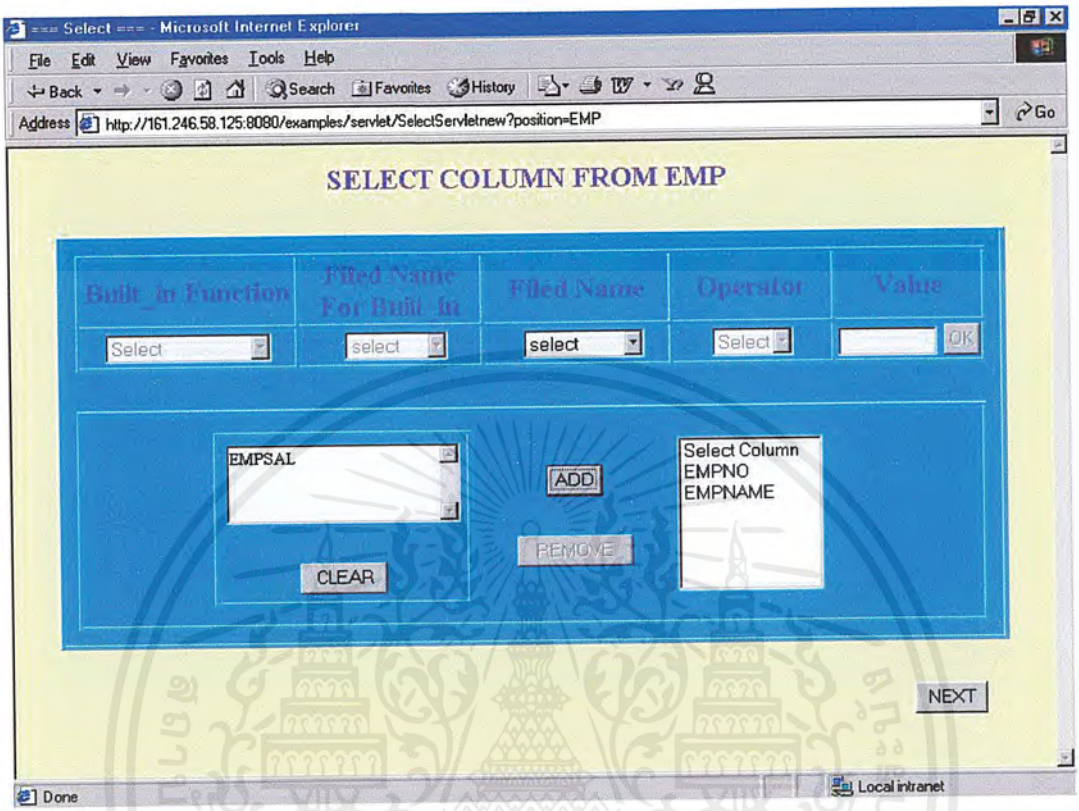


ภาพที่ 4.2 แสดงหน้าจอ From

เป็นหน้าจอสำหรับให้เลือก Table โดยเมื่อเลือก Table แล้วจะมีรายละเอียดของ Table นั้นแสดงให้เห็น และถ้าต้องการ Table ที่จะใช้งานให้กดปุ่ม NEXT เพื่อไปยังหน้าถัดไป

4.2.2 หน้าจอ Select

เมื่อเลือก Table จากหน้าจอ From เรียบร้อยแล้ว จะมาสู่หน้าจอนี้ดังภาพที่ 4.3



ภาพที่ 4.3 แสดงหน้าจอ Select

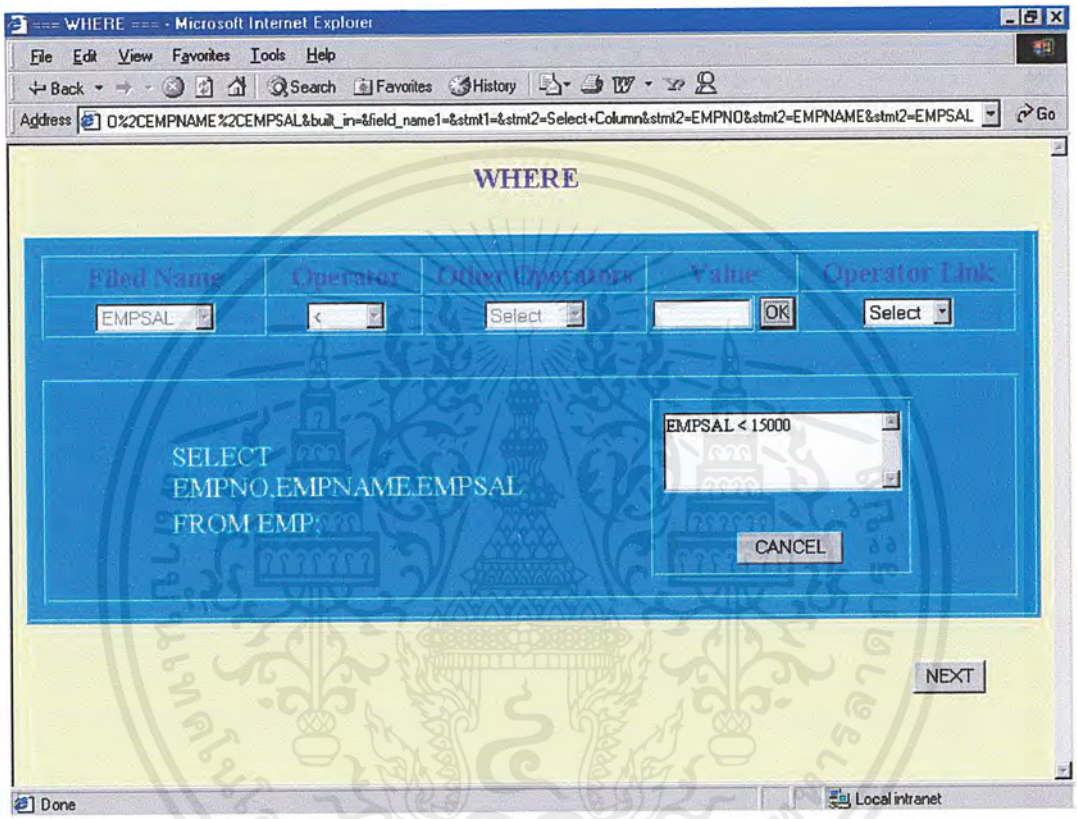
เป็นหน้าจอสำหรับเลือกคอลัมน์ ที่ต้องการดูข้อมูล มีรายละเอียดของหน้าจอดังนี้

- Built_in Function ใช้สำหรับต้องการดูข้อมูลในเชิงคำนวณและการนับ โดยคลิกเลือกฟังก์ชันที่ต้องการ
- Field Name For Built_in ใช้สำหรับคลิกเลือกชื่อคอลัมน์ที่ต้องการดูข้อมูลในเชิงคำนวณและการนับ
- Field Name ใช้สำหรับคลิกเลือกชื่อคอลัมน์ ที่ต้องการดูข้อมูล
- Operator ใช้สำหรับเลือกเครื่องหมายต่างๆ
- Value ให้พิมพ์ค่าคงที่ลงไปในช่อง แล้วกดปุ่ม O.K.
- Text Area ด้านซ้ายจะแสดงรูปแบบของคำสั่งที่กำลังเลือก
- ปุ่ม CLEAR เป็นปุ่มยกเลิกคำสั่งที่ได้เลือกมานี้
- List ด้านขวาจะแสดงรูปแบบของคำสั่งที่ถูกเลือก
- ปุ่ม Add เป็นปุ่มที่ใช้เพิ่มรายการคำสั่งจากซ้ายมือไปรวมไว้ที่ขวามือ

- ปุ่ม Remove เป็นปุ่มที่ใช้ย้ายรายการคำสั่งจากขวามือออก
- ปุ่ม NEXT ทำการกดปุ่มเพื่อไปหน้าต่อไป

4.2.3 หน้าจอ Where

เป็นหน้าจอสำหรับระบุเงื่อนไขของข้อมูลเฉพาะแถวที่ต้องการเรียกดู มีหน้าจอดังภาพที่ 4.4



ภาพที่ 4.4 แสดงหน้าจอ Where

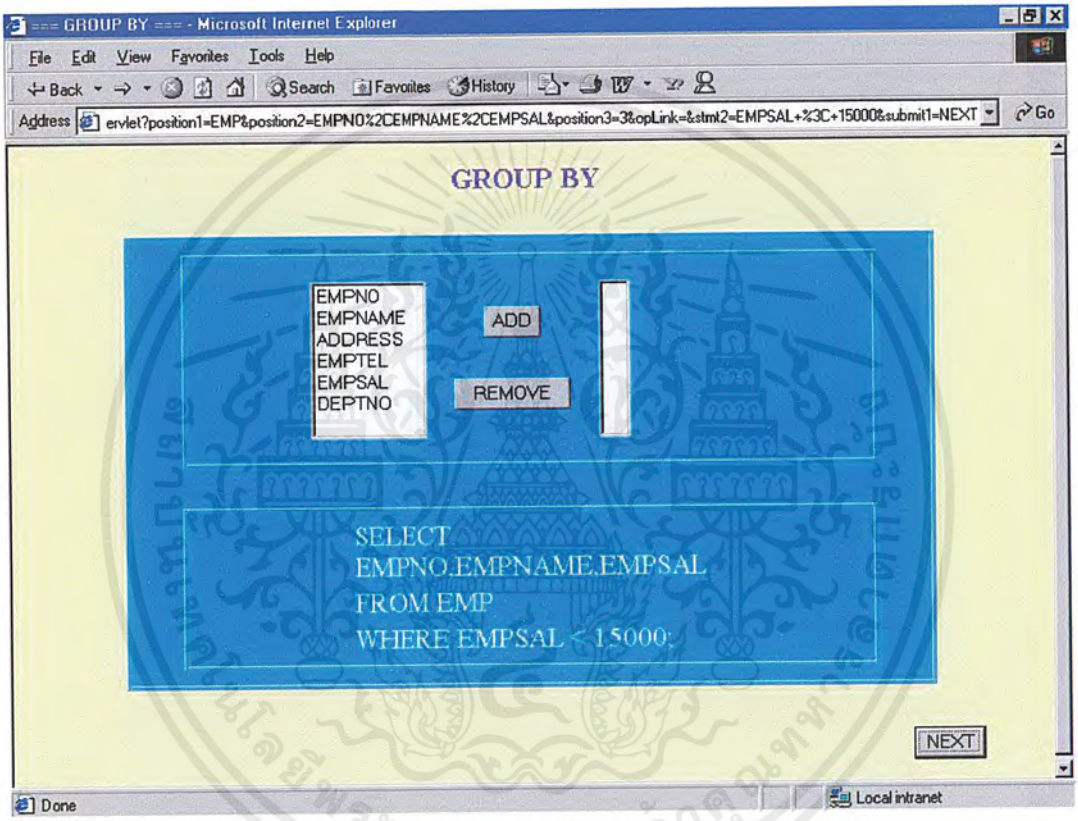
มีรายละเอียดของหน้าจอดังนี้

- Field Name ใช้สำหรับเลือกชื่อคอลัมน์ ที่ต้องการระบุเงื่อนไข
- Operator ใช้สำหรับเลือกเครื่องหมายที่ใช้ในการเปรียบเทียบ เช่น Operation "="
- Other operators ใช้สำหรับเลือกเครื่องหมายเปรียบเทียบที่นอกเหนือไปจากช่อง Operator เช่น Operation "In" , Operation "Is Null"
- Value ให้พิมพ์ค่าคงที่ลงไปในช่อง แล้วกดปุ่ม O.K.
- Operation link เป็นตัวเชื่อมในกรณีที่มีหลายเงื่อนไข
- ข้อความด้านซ้าย-ล่างจะแสดงรูปแบบคำสั่งที่ได้เลือกตั้งแต่หน้าจอ From จนถึงหน้าจอก่อนหน้านี้

- Text Area ด้านขวาจะแสดงรูปแบบของคำสั่งที่กำลังเลือก
- ปุ่ม CANCEL เป็นปุ่มยกเลิกเงื่อนไขที่ได้เลือกมานี้
- ปุ่ม NEXT ทำการกดปุ่มเพื่อไปหน้าต่อไป

4.2.4 **หน้าจอ** Group by

เป็นหน้าจอสำหรับระบุคำสั่งในการจับกลุ่มข้อมูลที่เหมือนกันเอาไว้ด้วยกัน แล้วกระทำฟังก์ชันบางอย่างตามกลุ่มเหล่านั้น มีหน้าจอดังภาพที่ 4.5



ภาพที่ 4.5 แสดงหน้าจอ Group by

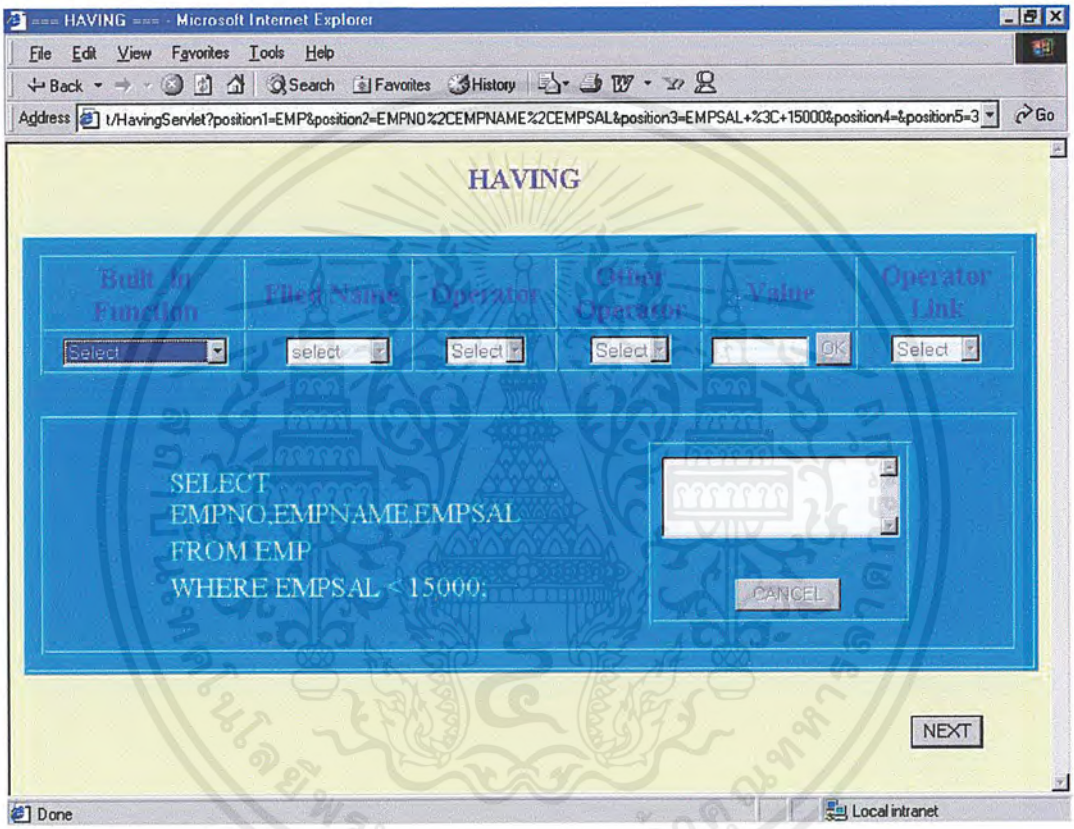
คำสั่ง Group by จะถูกใช้เมื่อในส่วนของคำสั่ง Select มี Fieldname ที่ต้องการจะจัดกลุ่ม และมี Built_in Function เพราะ Fieldname ในGroup by นี้ จะต้องเป็นชื่อเดียวกันกับของ Select เพื่อยืนยันว่าจะ return ค่า 1 ค่า จริงๆ จากที่ได้แบ่งกลุ่มแน่นอนแล้ว ซึ่งมีรายละเอียดของหน้าจอดังนี้

- List ด้านซ้ายจะระบุ Fieldname ไว้อยู่แล้ว ให้คลิกเลือก Fieldname ที่ต้องการ
- ปุ่ม Add ใช้สำหรับเพิ่ม Fieldname ที่ถูกเลือกไปใส่ไว้ใน List ด้านขวา
- ปุ่ม Remove ใช้สำหรับย้าย Fieldname ใน List ด้านขวาออกไป โดยคลิกเลือกที่ Fieldname นั้น แล้วกดปุ่ม REMOVE

- ข้อความด้านล่างจะแสดงรูปแบบของคำสั่งที่ได้เลือกตั้งแต่หน้าจอ From จนถึงหน้าจอก่อนหน้านี้
- ปุ่ม NEXT ทำการกดปุ่มเพื่อไปหน้าต่อไป

4.2.5 หน้าจอ Having

เป็นหน้าจอของการกำหนดเงื่อนไขให้มีการเปรียบเทียบโดยใช้ฟังก์ชันในการจัดกลุ่ม ซึ่งเงื่อนไขเหล่านั้นใช้ใน Where ไม่ได้ มีหน้าจอดังภาพที่ 4.6



ภาพที่ 4.6 แสดงหน้าจอ Having

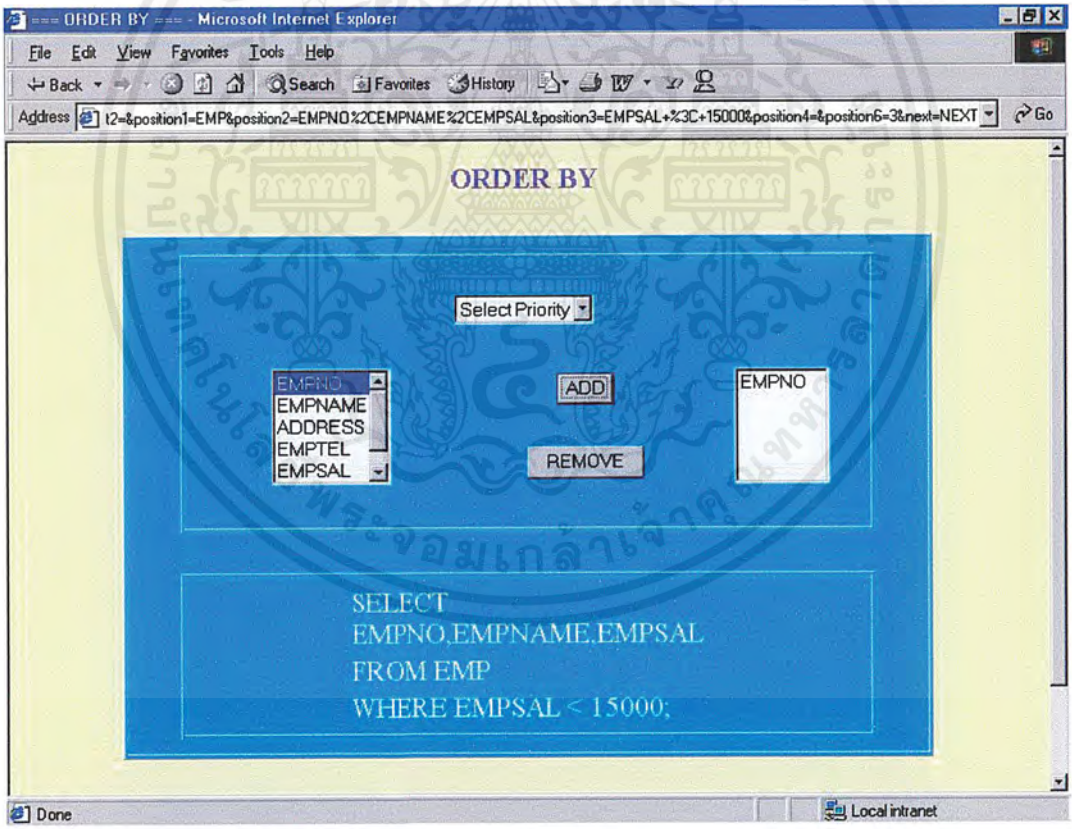
มีรายละเอียดของหน้าจอดังนี้

- Built_in Function ใช้สำหรับข้อมูลในเชิงคำนวณและการนับ โดยคลิกเลือกฟังก์ชันที่ต้องการ
- Field Name ใช้สำหรับเลือกชื่อคอลัมน์ ที่ต้องการจัดกลุ่ม
- Operator ใช้สำหรับเลือกเครื่องหมายเปรียบเทียบต่างๆเช่น "=",">"
- Other operators ใช้สำหรับเลือกเครื่องหมายในการคำนวนเช่น "+","-"
- Value ให้พิมพ์ค่าคงที่ลงไปในช่อง แล้วกดปุ่ม O.K.

- Operation link เป็นตัวเชื่อมในกรณีที่มีหลายเงื่อนไข
- ข้อความด้านซ้าย-ล่างจะแสดงรูปแบบคำสั่งที่ได้เลือกตั้งแต่หน้าจอ From จนถึงหน้าจอก่อนหน้านี้
- Text Area ด้านขวาจะแสดงรูปแบบของคำสั่งที่กำลังเลือก
- ปุ่ม CANCEL เป็นปุ่มยกเลิกเงื่อนไขที่ได้เลือกมานี้
- ปุ่ม NEXT ทำการกดปุ่มเพื่อไปหน้าต่อไป

4.2.6 **หน้าจอ** Order by

เป็นหน้าจอของคำสั่งการเรียงลำดับผลที่ได้จากคำสั่งสืบค้น ตามชื่อ Fieldname ที่ระบุอยู่หลังคำสั่ง Order by โดย Fieldname เหล่านั้น จะต้องถูกเลือกขึ้นมาแสดงด้วย (มาจากหลังคำสั่ง Select) โดยปกติการเรียงลำดับจะเรียงคอลัมน์ที่ระบุก่อนหลังตามลำดับ มีหน้าจอดังภาพที่ 4.7



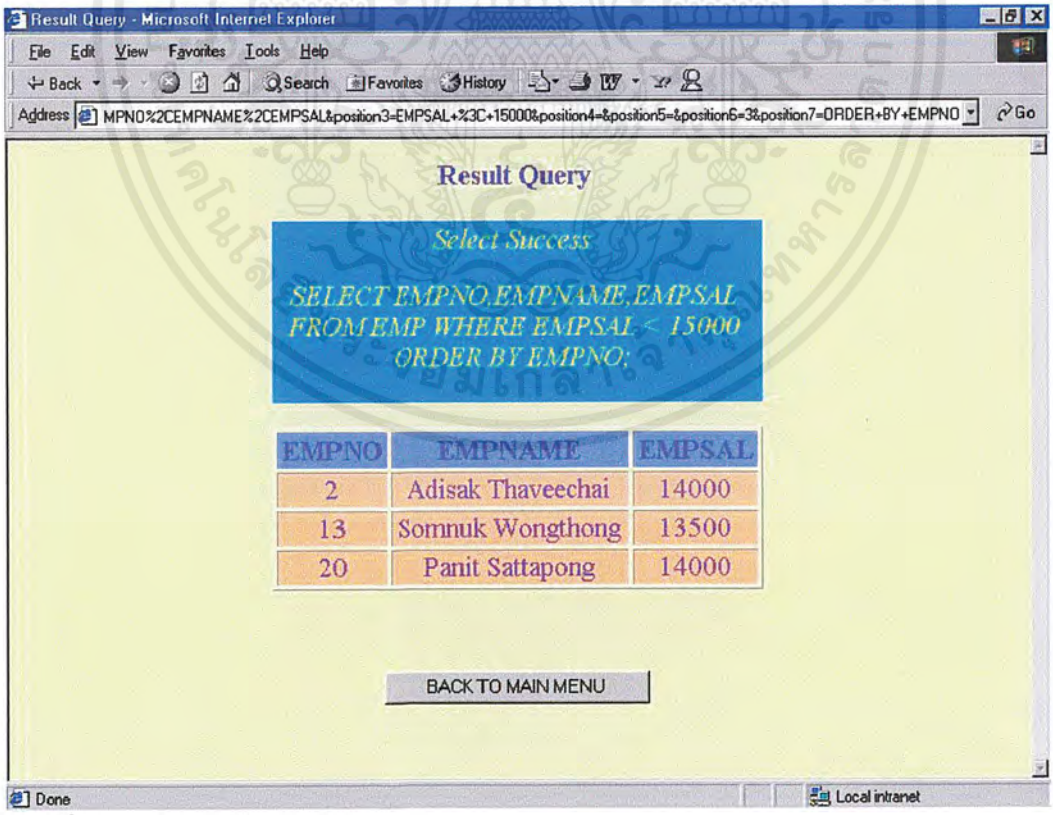
ภาพที่ 4.7 แสดงหน้าจอ Order by

มีรายละเอียดของหน้าจอดังนี้

- Select Priority ให้คลิกเลือกว่าต้องการให้ผลลัพธ์เรียงลำดับอย่างไร ถ้าต้องการเรียงจากน้อยไปหามาก ก็เลือก ascending แต่หากต้องการเรียงจากมากไปหาน้อย ก็เลือก descending
- List ด้านซ้ายจะระบุ Fieldname ไว้อยู่แล้ว ให้คลิกเลือก Fieldname ที่ต้องการ
- ปุ่ม Add ใช้สำหรับเพิ่ม Fieldname ที่ถูกเลือกไปใส่ไว้ใน List ด้านขวา
- ปุ่ม Remove ใช้สำหรับย้าย Fieldname ที่ถูกเลือกใน List ด้านขวาออกไป
- ข้อความด้านล่างจะแสดงรูปแบบของคำสั่งที่ได้เลือกตั้งแต่หน้าจอ From จนถึงหน้าจอก่อนหน้านี้
- ปุ่ม PROCESS QUERY ใช้สำหรับส่งคำสั่ง Select ทั้งหมดไปที่ฐานข้อมูลแล้วจะได้ผลลัพธ์กลับมา

4.2.7 หน้าจอ Query

เป็นหน้าจอแสดงผลลัพธ์ที่ได้ จากการส่งคำสั่ง Select ไปที่ฐานข้อมูล มีหน้าจอดังภาพที่ 4.8



ภาพที่ 4.8 แสดงหน้าจอ Query

มีรายละเอียดของหน้าจอ ดังนี้

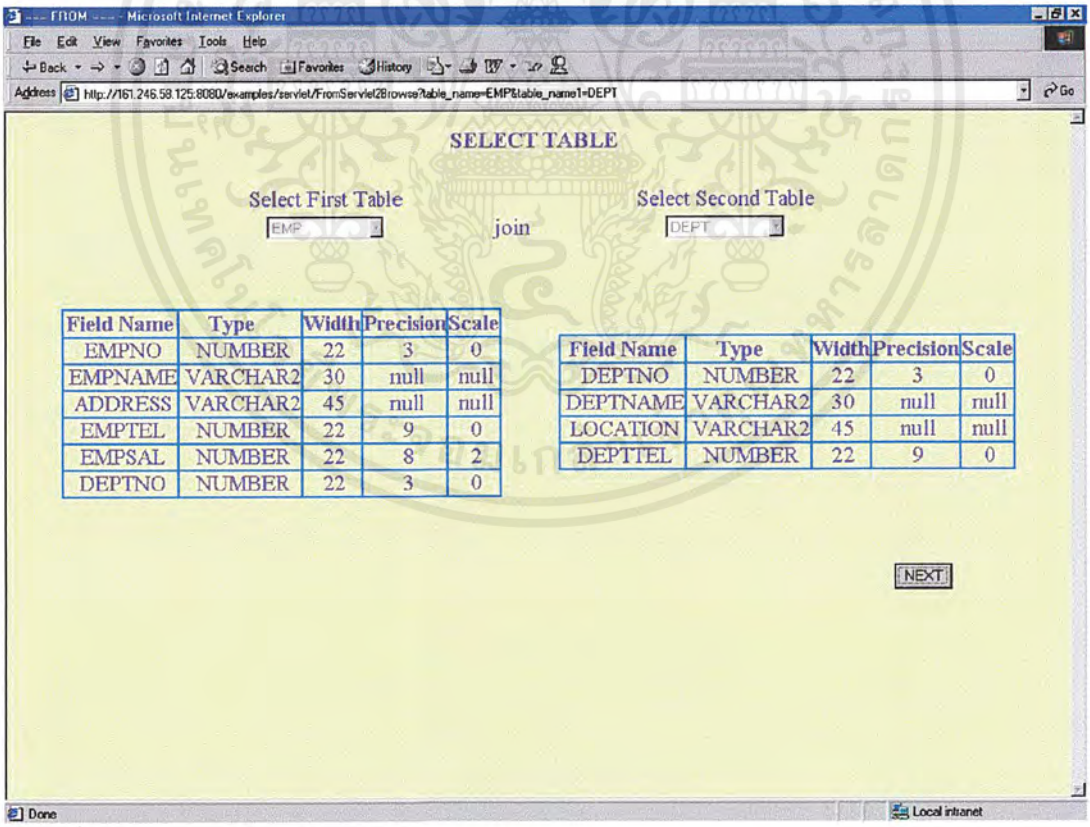
- ส่วนบนของหน้าจอเป็นรูปแบบคำสั่ง Select ที่ได้ส่งไป Query
- ส่วนด้านล่างเป็นผลลัพธ์ที่ได้จากการ Query
- ปุ่ม BACK TO MAIN MENU จะอยู่ด้านล่างสุด กดปุ่มเพื่อกลับไปหน้า MENU แล้วเลือกฟังก์ชันการทำงานตามแต่ต้องการ

4.3 ฟังก์ชันการทำงาน Join Table

เป็นคำสั่งสำหรับการดูข้อมูลที่อยู่ร่วมกัน ซึ่งจะทำการเรียกดูข้อมูลจากตารางได้มากกว่า 1 ตาราง โดยเลือกดูเฉพาะแถวที่มีค่าข้อมูลบางส่วนร่วมกันอยู่

4.3.1 หน้าจอ Select Table

เป็นหน้าจอสำหรับเลือกตารางที่ต้องการจะดูข้อมูล โดยในตารางทั้ง 2 ที่จะทำการ Join จะต้องมี Fieldname ที่เหมือนกันอยู่ 1 ตัว คือชื่อเดียวกัน และ Data type ตรงกัน มีหน้าจอดังภาพที่ 4.9



ภาพที่ 4.9 แสดงหน้าจอ Select Table ของการ Join

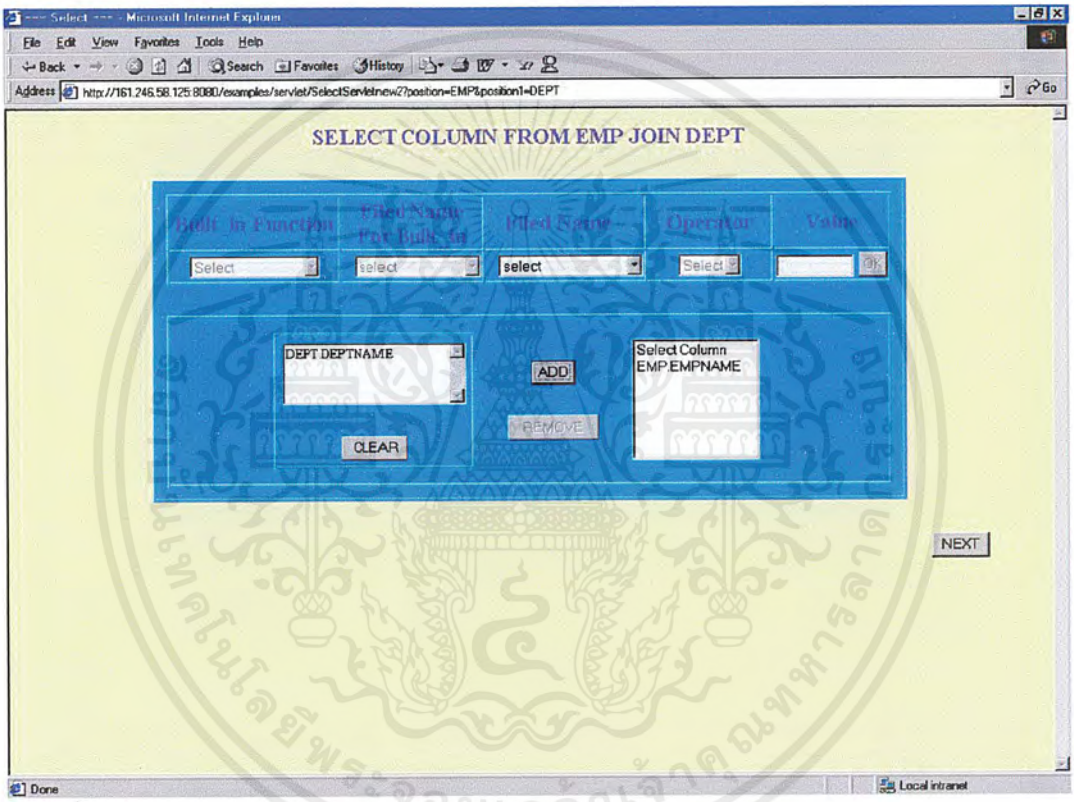
มีรายละเอียดของหน้าจอดังนี้

- Select First Table ให้ทำการเลือกตารางที่ 1 ที่ต้องการจะดูข้อมูล
- Select Second Table ให้ทำการเลือกตารางที่ 2 เพื่อมา Join กับตารางที่ 1

เมื่อเลือก Table ครบแล้วจะมีรายละเอียดของ Table นั้นแสดงให้เห็น และถ้าต้องการ Table ที่จะใช้งานให้กดปุ่ม NEXT เพื่อไปยังหน้าถัดไป

4.3.2 หน้าจอ Select Column

เมื่อเลือก Table ที่ต้องการเรียบร้อยแล้ว จะมาสู่หน้าจอนี้ดังภาพที่ 4.10

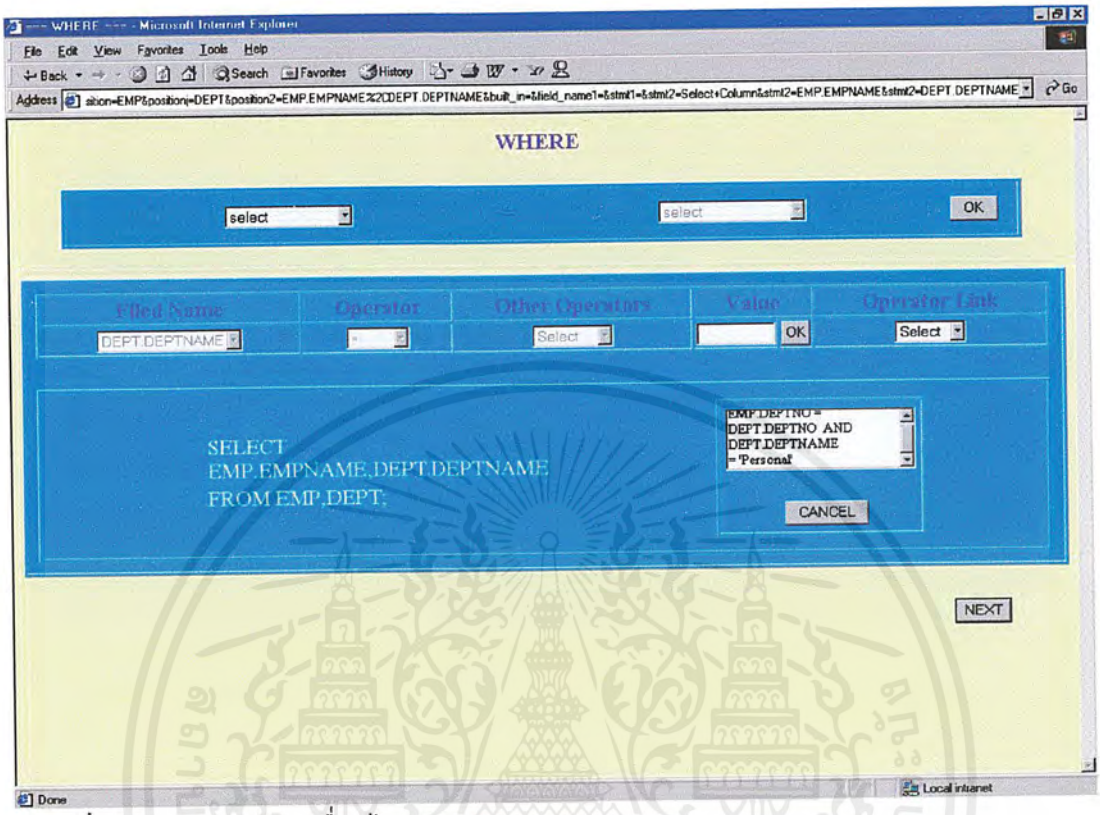


ภาพที่ 4.10 แสดงหน้าจอ Select Column

เป็นหน้าจอสำหรับเลือก Fieldname ที่ต้องการจะดูข้อมูล โดยมีรูปแบบหน้าจอเหมือนกับหน้าจอ Select ของการเรียกดูข้อมูลจากหนึ่งตาราง จะมีส่วนต่างกันเล็กน้อย คือ ชื่อของ Fieldname จะเป็นชื่อ Table ต่อด้วยจุด แล้วตามด้วยชื่อ Fieldname

4.3.3 หน้าจอ Where

เป็นการกำหนดเงื่อนไขในการ Join มีหน้าจอดังภาพที่ 4.11



ภาพที่ 4.11 แสดงหน้าจอเงื่อนไขการ Join

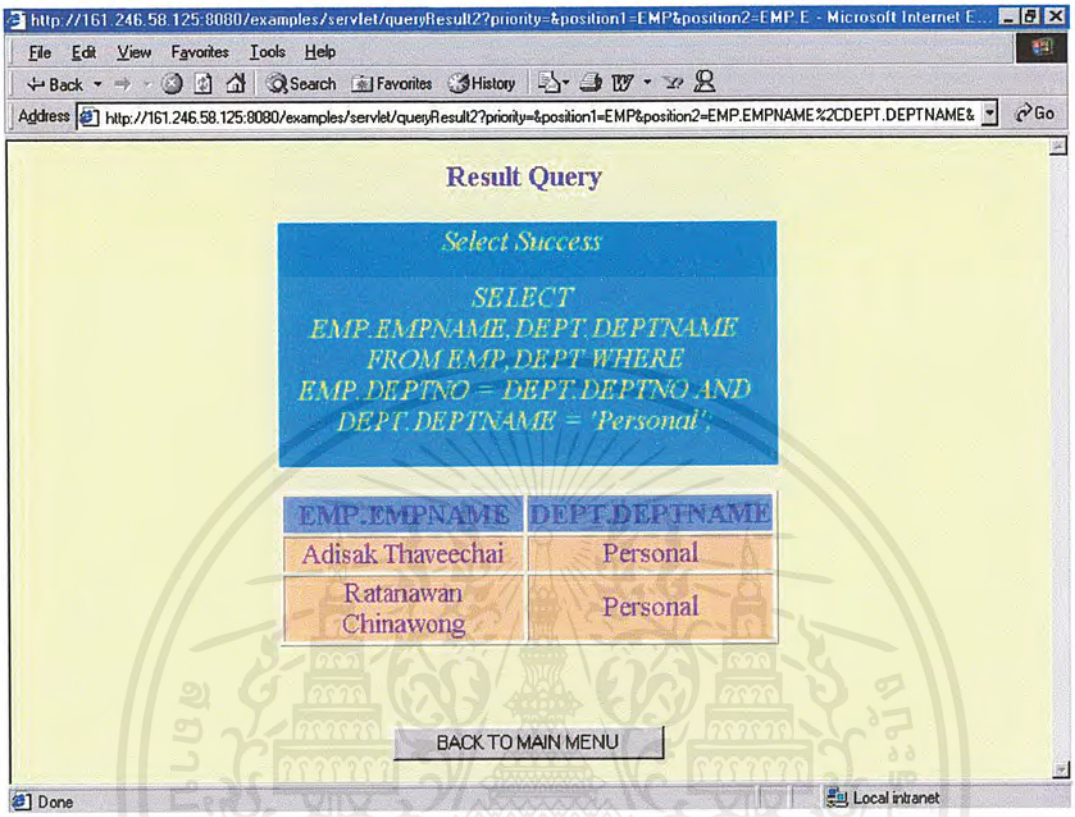
มีรายละเอียดของหน้าจอดังนี้

- ส่วนบนเป็นการนำ Fieldname ที่เหมือนกันของทั้ง 2 ตาราง มา Join กัน เพื่อเป็นการกำหนดเงื่อนไขในการ Join
- ปุ่ม O.K จะทำการกดปุ่มนี้ ถ้าต้องการกำหนดเงื่อนไขในการค้นหาเพิ่มเติม
- ส่วนของ Where Condition ส่วนนี้เป็นการกำหนดเงื่อนไขในการค้นหาข้อมูล ซึ่งมีรูปแบบเหมือนกับการกำหนดเงื่อนไขการเรียกค้นข้อมูลจากหนึ่งตาราง จะมีส่วนต่างกันเล็กน้อย คือ ชื่อของ Fieldname จะเป็นชื่อ Table ต่อด้วยจุด แล้วตามด้วยชื่อ Fieldname

ส่วนหน้าจอของการ Group by , Having และ Order by ของการ Join Table นี้ มีรูปแบบการใช้งานเหมือนกับการใช้งานในกรณีเรียกค้นข้อมูลจากหนึ่งตาราง

4.3.4 หน้าจอ Query

เป็นหน้าจอแสดงผลลัพธ์ที่ได้ จากการส่งคำสั่งไปที่ฐานข้อมูล มีหน้าจอดังภาพที่ 4.12



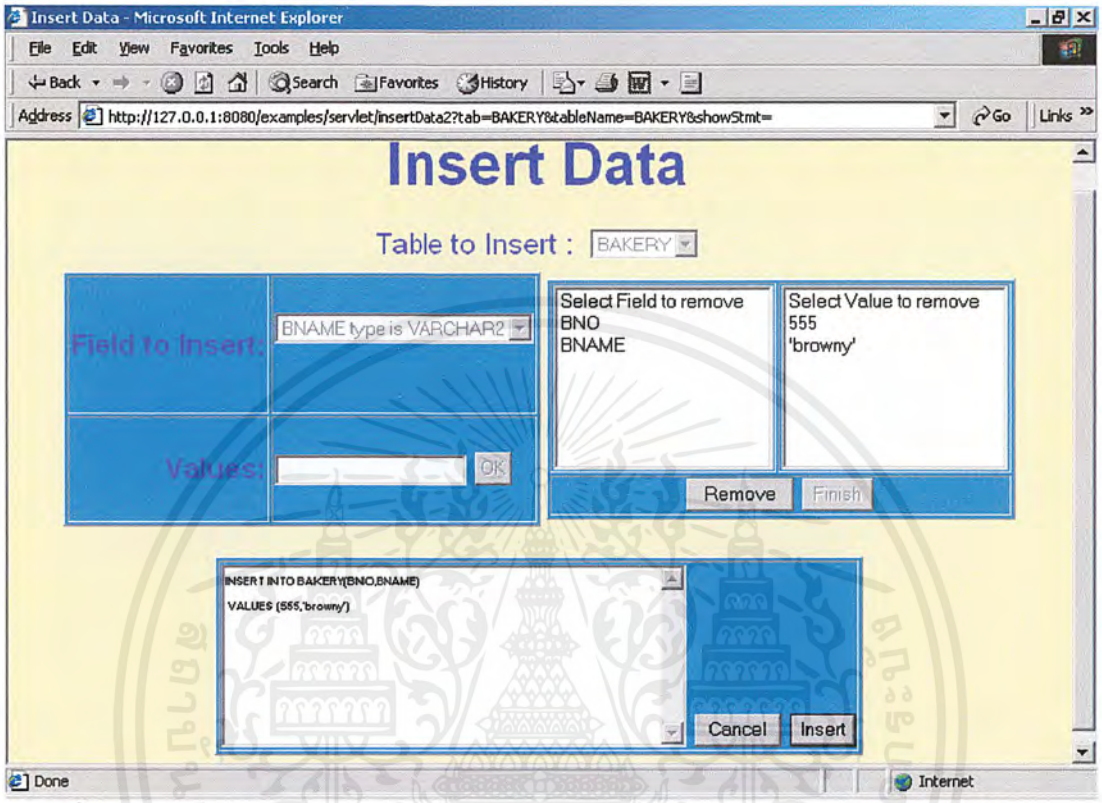
ภาพที่ 4.12 แสดงหน้าจอ Query

มีรายละเอียดของหน้าจอ ดังนี้

- ส่วนบนของหน้าจอเป็นรูปแบบคำสั่งที่ได้ส่งไป Query
- ส่วนด้านล่างเป็นผลลัพธ์ที่ได้จากการ Query
- ปุ่ม BACK TO MAIN MENU จะอยู่ด้านล่างสุด กดปุ่มเพื่อกลับไปหน้า MENU แล้วเลือกฟังก์ชันการทำงานตามแต่ต้องการ

4.4 หน้าจอ Insert Data

เป็นหน้าจอที่ใช้เพิ่มเติมข้อมูลเข้าไปในฐานข้อมูล (เป็นการเพิ่มแรกคอร์ดในตาราง) โดยจัดเก็บไว้ใน table มีหน้าจอดังภาพที่ 4.13



ภาพที่ 4.13 แสดงหน้าจอ Insert Data

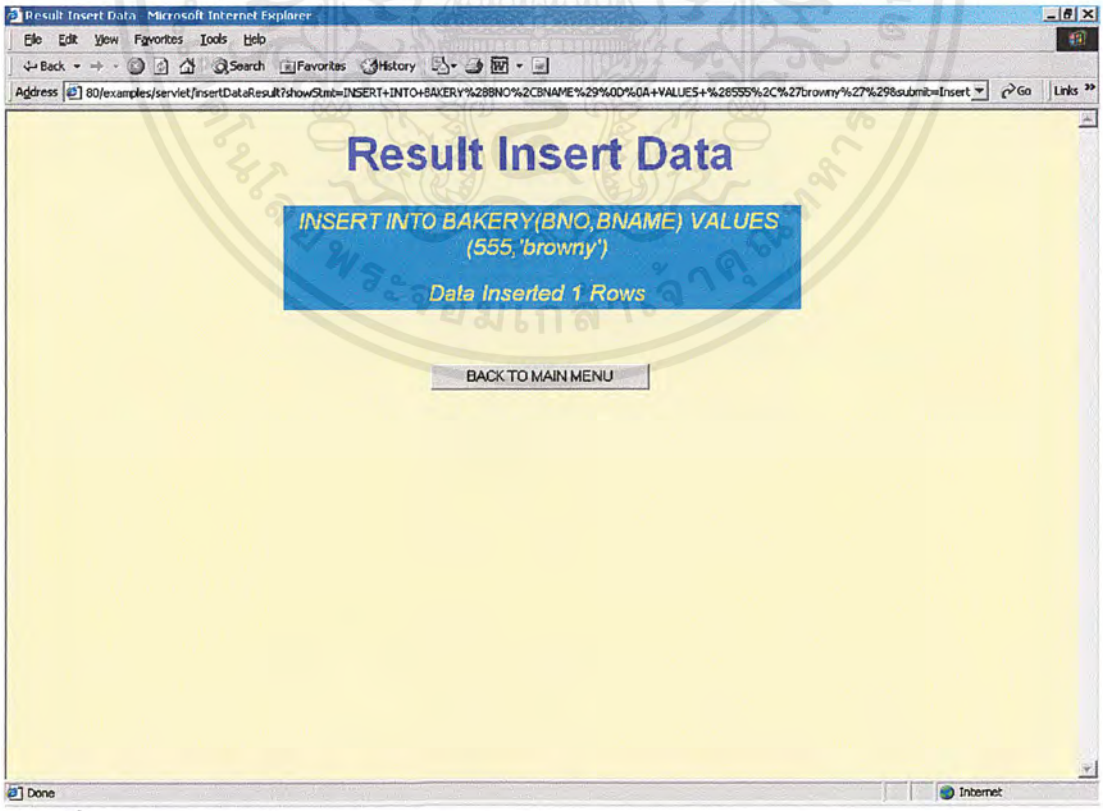
มีรายละเอียดของหน้าจอดังนี้

- Table to Insert ใช้สำหรับเลือกชื่อตารางที่ต้องการจะเพิ่มเติมข้อมูล
- Field to Insert ใช้สำหรับเลือกชื่อคอลัมน์ ที่ต้องการจะเพิ่มเติมข้อมูล โดยคลิกเลือก Fieldname ที่ต้องการจนกระทั่งครบ
- Values ใส่ค่าข้อมูลที่จะเพิ่มตามลำดับของ Field ที่ได้เลือกไว้ แล้วกดปุ่ม O.K ซึ่งสามารถใส่เพิ่มได้จนเท่ากับจำนวน Field ที่เลือก Insert
- List ด้านซ้ายจะแสดงชื่อคอลัมน์ที่ถูกเลือกจากช่อง Field to Insert
- List ด้านขวาจะแสดงรายการข้อมูลที่ได้ใส่ลงไปในช่อง Value ซึ่งค่า Value ต้องสัมพันธ์กับ FieldName

- ปุ่ม Remove เป็นปุ่มที่ใช้สำหรับลบชื่อคอลัมน์ที่เลือกมาแล้วออกไปจาก List ด้านซ้าย หรือลบรายการข้อมูลที่ใส่ลงไปออกจาก List ด้านขวา โดยคลิกเลือกที่ชื่อ หรือ ค่าที่ต้องการจะ Remove แล้วกดปุ่ม Remove
- ปุ่ม Finish เป็นปุ่มที่เมื่อเลือกชื่อคอลัมน์ และ ใส่รายการข้อมูลที่จะ Insert เรียบร้อยแล้ว ก็จะกดปุ่มนี้เพื่อเป็นการตอบตกลงว่าเลือกเสร็จเรียบร้อยแล้ว และจะแสดง Command ใน Text Area ข้างล่าง
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Insert โดยจะแสดงให้เห็นหลังจากกดปุ่ม Finish แล้ว
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Insert ที่ได้เลือกมานั้นทั้งหมด
- ปุ่ม Insert เป็นปุ่มสำหรับส่งคำสั่ง Insert นี้ไปยังฐานข้อมูล

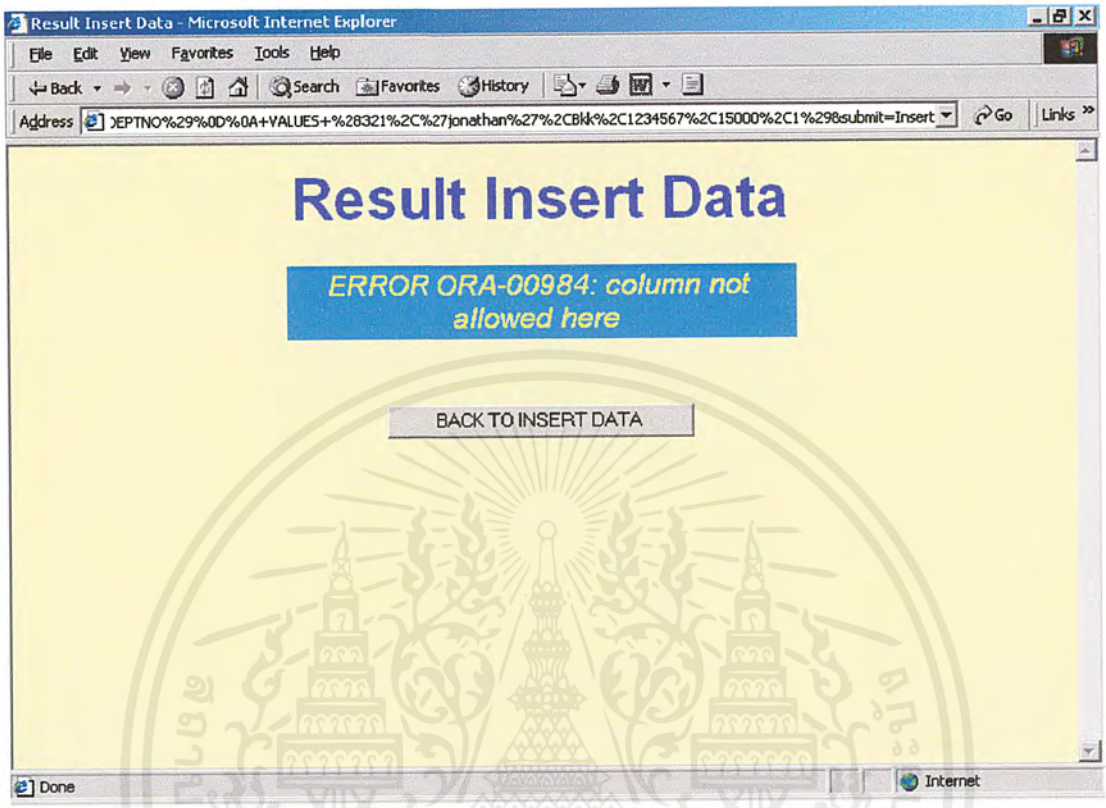
4.4.1 หน้าจอ Result Insert Data

เป็นหน้าจอที่แสดงผลหลังจากเพิ่มเติมข้อมูลเข้าไปในฐานข้อมูล ถ้า Insert สำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงจำนวนเรกคอร์ดหรือ Rows ที่ได้ทำการ Insert ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.14



ภาพที่ 4.14 แสดงหน้าจอ Result Insert Data

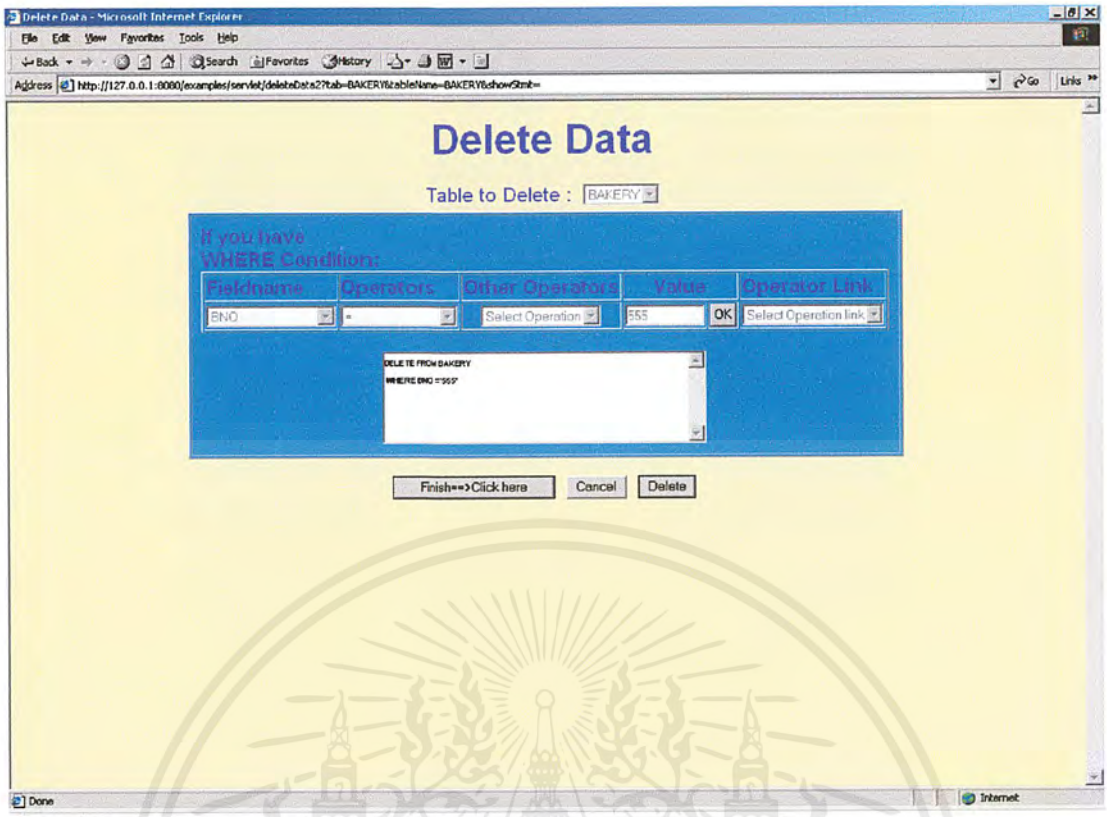
ถ้า Insert ไม่สำเร็จจะมีข้อความแสดงผลว่าทำไมถึงเกิดข้อผิดพลาดขึ้น ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.15



ภาพที่ 4.15 แสดงหน้าจอ Result Insert Data Error

4.5 หน้าจอ Delete Data

เป็นหน้าจอของคำสั่งลบรายการออกจากตาราง ใช้ลบแถวของข้อมูลที่สอดคล้องกับเงื่อนไขที่ระบุออกจากตาราง แต่ถ้าไม่ระบุเงื่อนไข ก็จะเป็นการลบรายการข้อมูลทั้งหมดในตารางนั้น มีหน้าจอดังภาพที่ 4.16



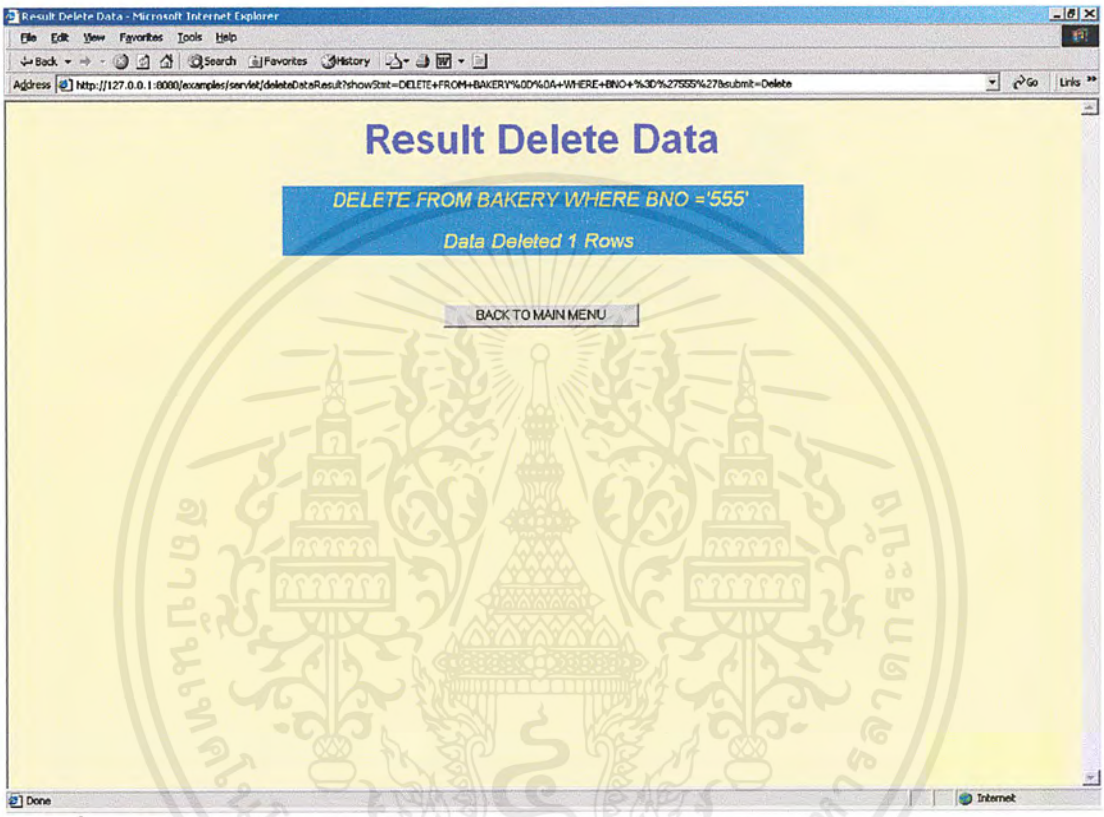
ภาพที่ 4.16 แสดงหน้าจอ Delete Data

มีรายละเอียดของหน้าจอดังนี้

- Table to Delete ใช้สำหรับเลือกชื่อตารางที่ต้องการจะลบข้อมูล
- Where Condition เป็นส่วนที่จะระบุเงื่อนไขของข้อมูลที่ต้องการจะลบ
- Fieldname ใช้สำหรับเลือกชื่อคอลัมน์ ที่ต้องการระบุเงื่อนไข
- Operator ใช้สำหรับเลือกเครื่องหมายที่ใช้ในการเปรียบเทียบ
- Other operators ใช้สำหรับเลือกเครื่องหมายที่ใช้เปรียบเทียบที่นอกเหนือไปจากช่อง Operator
- Value ให้พิมพ์ค่าคงที่ลงไปในช่อง แล้วกดปุ่ม O.K.
- Operation link เป็นตัวเชื่อมในกรณีที่มีหลายเงื่อนไข
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Delete โดยจะแสดงให้เห็นหลังจากกดปุ่ม Finish Chick here
- ปุ่ม Finish Chick here เป็นปุ่มสำหรับตอบตกลงว่าได้ทำการเลือกรายการที่จะ Delete เรียบร้อยแล้ว
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Delete ที่ได้เลือกมานั้นทั้งหมด
- ปุ่ม Delete เป็นปุ่มสำหรับส่งคำสั่ง Delete นี้ไปยังฐานข้อมูล

4.5.1 **หน้าจอ** Result Delete Data

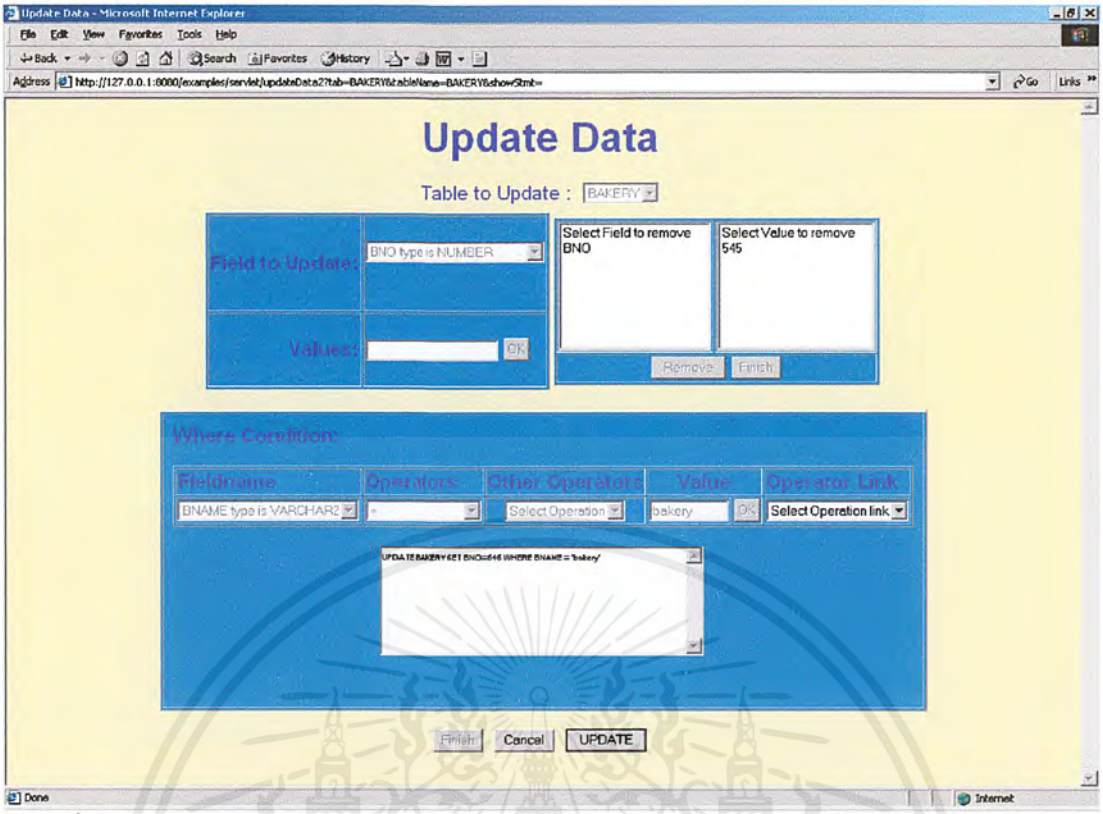
เป็นหน้าจอที่แสดงผลหลังจากลบข้อมูลในฐานข้อมูล เมื่อลบข้อมูลสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงจำนวนเรกคอร์ดหรือ Rows ที่ได้ทำการลบข้อมูล ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.17



ภาพที่ 4.17 แสดงหน้าจอ Result Delete Data

4.6 **หน้าจอ** Update Data

เป็นหน้าจอของคำสั่งเปลี่ยนแปลงแก้ไขข้อมูลใน table ซึ่งการเปลี่ยนแปลงแก้ไขควรระบุเงื่อนไข ถ้าไม่ระบุเงื่อนไขอะไรเลย การปรับปรุงจะทำกับทุกรายการ มีหน้าจอดังภาพที่ 4.18



ภาพที่ 4.18 แสดงหน้าจอ Update Data

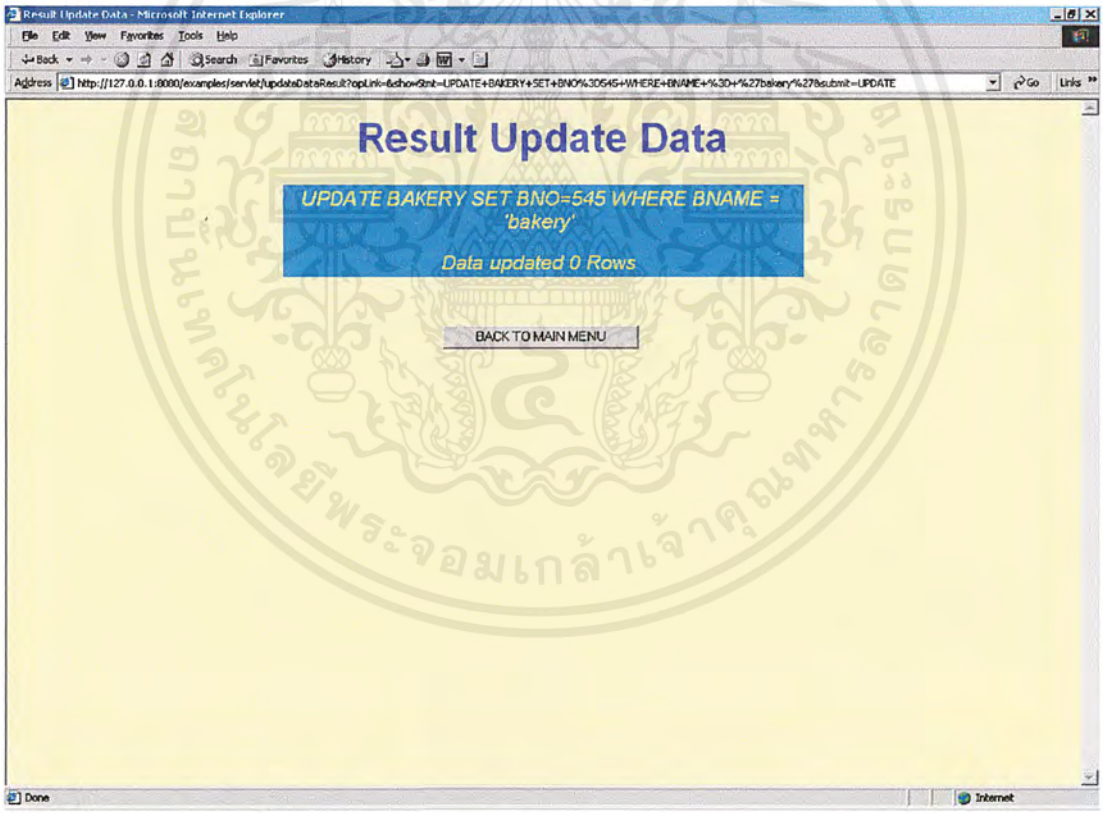
มีรายละเอียดของหน้าจอดังนี้

- Table to Update ใช้สำหรับเลือกชื่อตารางที่ต้องการจะเปลี่ยนแปลงแก้ไขข้อมูล
- Field to Update ใช้สำหรับเลือกชื่อคอลัมน์ ที่ต้องการจะเปลี่ยนแปลงแก้ไขข้อมูล โดยคลิกเลือก Fieldname ที่ต้องการจนกระทั่งครบ
- Values ใส่ค่าข้อมูลที่จะเปลี่ยนแปลงแก้ไขตามลำดับของ Field ที่ได้เลือกไว้ จนกระทั่งครบแล้วกดปุ่ม O.K
- List ด้านซ้าย จะแสดงชื่อคอลัมน์ที่ถูกเลือกจากช่อง Field to Update
- List ด้านขวาจะแสดงรายการข้อมูลที่ได้ใส่ลงไปในช่อง Value
- ปุ่ม Remove เป็นปุ่มที่ใช้สำหรับลบชื่อคอลัมน์ที่เลือกมาแล้วออกไปจาก List ด้านซ้าย หรือลบรายการข้อมูลที่ใส่ลงไปออกจาก List ด้านขวา โดยคลิกเลือกที่ชื่อ หรือ ค่าที่ต้องการจะ Remove แล้วกดปุ่ม Remove
- ปุ่ม Finish เป็นปุ่มที่เมื่อเลือกชื่อคอลัมน์ และ ใส่รายการข้อมูลที่จะ Update เรียบร้อยแล้ว ก็จะกดปุ่มนี้เพื่อเป็นการตอบตกลงว่าเลือกเรียบร้อยแล้ว และต้องการระบุเงื่อนไขในการ Update ข้อมูล

- Where Condition มีรูปแบบเหมือนกับ หน้าจอ Delete ในส่วนของการระบุเงื่อนไขในการ Update ข้อมูล
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Update โดยจะแสดงให้เห็นเมื่อกดปุ่ม Finish
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Update ที่ได้เลือกมานั้นทั้งหมด
- ปุ่ม Update เป็นปุ่มสำหรับส่งคำสั่ง Update นี้ไปยังฐานข้อมูล

4.6.1 **หน้าจอ** Result Update Data

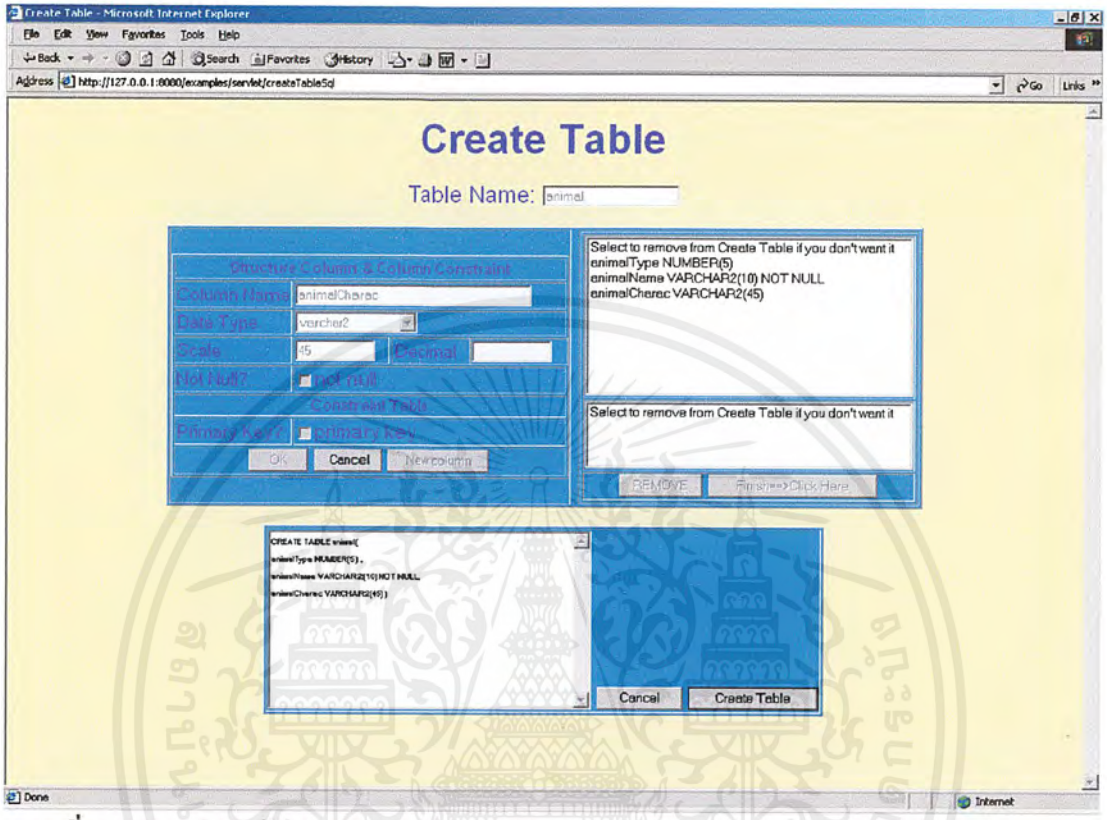
เป็นหน้าจอที่แสดงผลหลังจากเปลี่ยนแปลงแก้ไขข้อมูลในฐานข้อมูล เมื่อเปลี่ยนแปลงแก้ไขข้อมูลสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงจำนวนเรกคอร์ดหรือ Rows ที่ได้ทำการเปลี่ยนแปลงแก้ไขข้อมูล ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.19



ภาพที่ 4.19 แสดงหน้าจอ Result Update Data

4.7 หน้าจอ Create Table

เป็นหน้าจอของคำสั่งในการสร้างตารางขึ้นมาใหม่ในระบบฐานข้อมูล มีหน้าจอดังภาพที่ 4.20



ภาพที่ 4.20 แสดงหน้าจอ Create Table

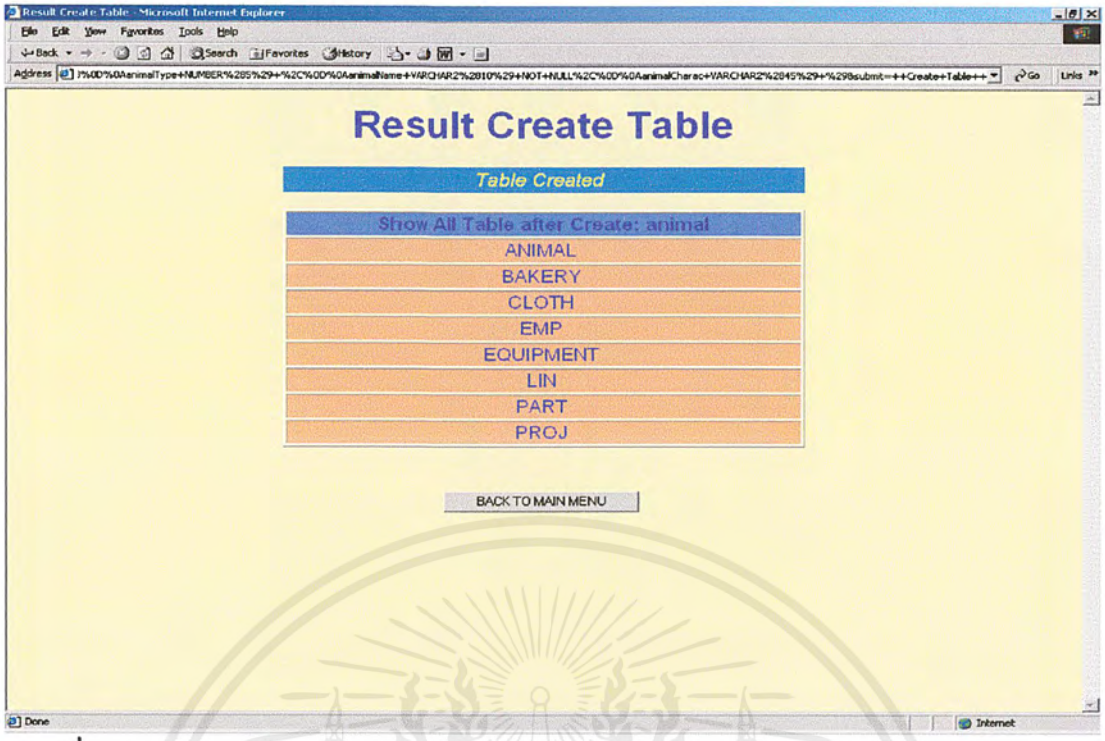
มีรายละเอียดของหน้าจอดังนี้

- Table name กำหนดชื่อตาราง โดยใส่ชื่อตารางที่จะสร้าง
- Structure Column & Column Constraint เป็นส่วนของการกำหนดโครงสร้างให้แต่ละคอลัมน์
- Column name กำหนดชื่อคอลัมน์ โดยใส่ชื่อ Fieldname
- Data type เป็นการกำหนดชนิดของข้อมูลในคอลัมน์นั้น โดยทำการคลิกเลือกชนิดของข้อมูล
- Scale เป็นการกำหนดขนาด หรือความยาวของข้อมูลที่คอลัมน์นี้สามารถรองรับได้ โดยใส่ตัวเลขลงไป
- Decimal เป็นการกำหนดจุดทศนิยม ในกรณีที่ เลือก Data type เป็น number โดยใส่ตัวเลขลงไป

- Not null เป็นการกำหนดให้คอลัมน์นี้ต้องมีการใส่ค่าข้อมูล ห้ามเป็นค่าว่าง โดยคลิกเลือกที่ check box แต่ถ้าไม่คลิกเลือก คอลัมน์นั้นก็จะมีสถานะเป็น null
- Primary key เป็นการกำหนดให้คอลัมน์นี้เป็นคีย์หลัก โดยคลิกที่ check box แต่ถ้าคอลัมน์นั้นไม่ได้เป็นคีย์หลัก ก็ไม่ต้องคลิกเลือก
- ปุ่ม O.K กดปุ่มเพื่อเสร็จสิ้นการกำหนดโครงสร้างของคอลัมน์ 1 คอลัมน์
- ปุ่ม Cancel กดปุ่มเพื่อยกเลิกการกำหนดโครงสร้างของคอลัมน์
- ปุ่ม New column กดปุ่มนี้เมื่อต้องการเพิ่มคอลัมน์อีก
- List ด้านขวา-บน เป็นส่วนที่แสดงถึงรูปแบบโครงสร้างของคอลัมน์ที่ได้กำหนดไว้ โดยจะแสดงให้เห็นเมื่อกดปุ่ม O.K
- List ด้านขวา-ล่าง เป็นส่วนที่แสดงถึงรูปแบบโครงสร้างของคอลัมน์ที่ถูกกำหนดให้เป็นคีย์หลัก
- ปุ่ม Remove เป็นปุ่มที่ใช้สำหรับลบคอลัมน์ที่ได้กำหนดโครงสร้างแล้วออกไปจาก List ด้านขวา-บน โดยคลิกเลือกที่รายการที่ต้องการจะ Remove แล้วกดปุ่ม Remove
- ปุ่ม Finish Click Here เป็นปุ่มสำหรับตกลงว่าได้ทำการกำหนดโครงสร้างคอลัมน์ที่จะ Create ครบเรียบร้อยแล้ว
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Create Table โดยจะแสดงให้เห็นหลังจากกดปุ่ม Finish Chick here
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Create ที่ได้กำหนดมานั้นทั้งหมด
- ปุ่ม Create Table เป็นปุ่มสำหรับส่งคำสั่ง Create Table นี้ไปยังฐานข้อมูล

4.7.1 **หน้าจอ** Result Create Table

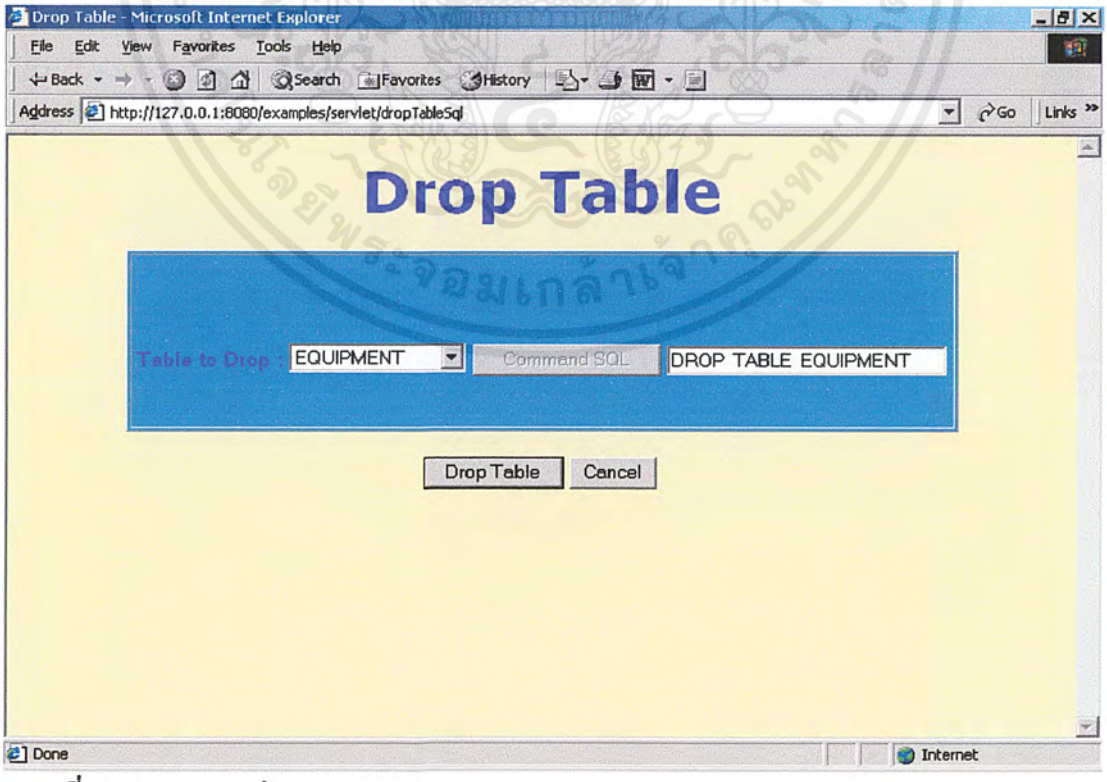
เป็นหน้าจอที่แสดงผลหลังจากสร้างตารางในฐานข้อมูล เมื่อสร้างตารางสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงตารางทั้งหมดหลังจากที่ได้สร้างตารางใหม่ ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.21



ภาพที่ 4.21 แสดงหน้าจอ Result Create Table

4.8 หน้าจอ Drop Table

เป็นหน้าจอของคำสั่งที่ใช้ลบตารางออกจากระบบ มีหน้าจอดังภาพที่ 4.22



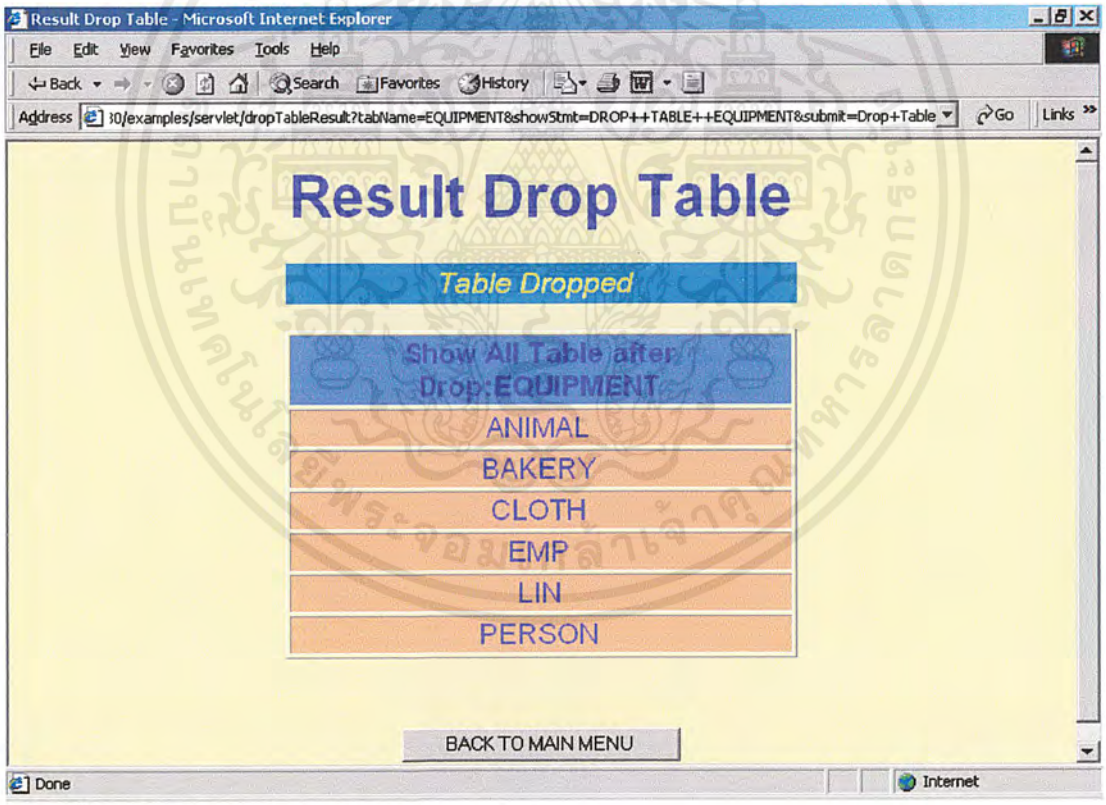
ภาพที่ 4.22 แสดงหน้าจอ Drop Table

มีรายละเอียดของหน้าจอดังนี้

- Select Table to Drop ใช้สำหรับเลือกชื่อตารางที่ต้องการจะลบ
- ปุ่ม Show command SQL เป็นปุ่มกดเพื่อให้แสดงรูปแบบของคำสั่ง Drop Table ที่ Text Area ด้านขวา
- ปุ่ม Drop Table เป็นปุ่มสำหรับส่งคำสั่ง Drop Table นี้ไปยังฐานข้อมูล
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Drop Table ที่ได้เลือกมานั้นทั้งหมด

4.8.1 หน้าจอ Result Drop Table

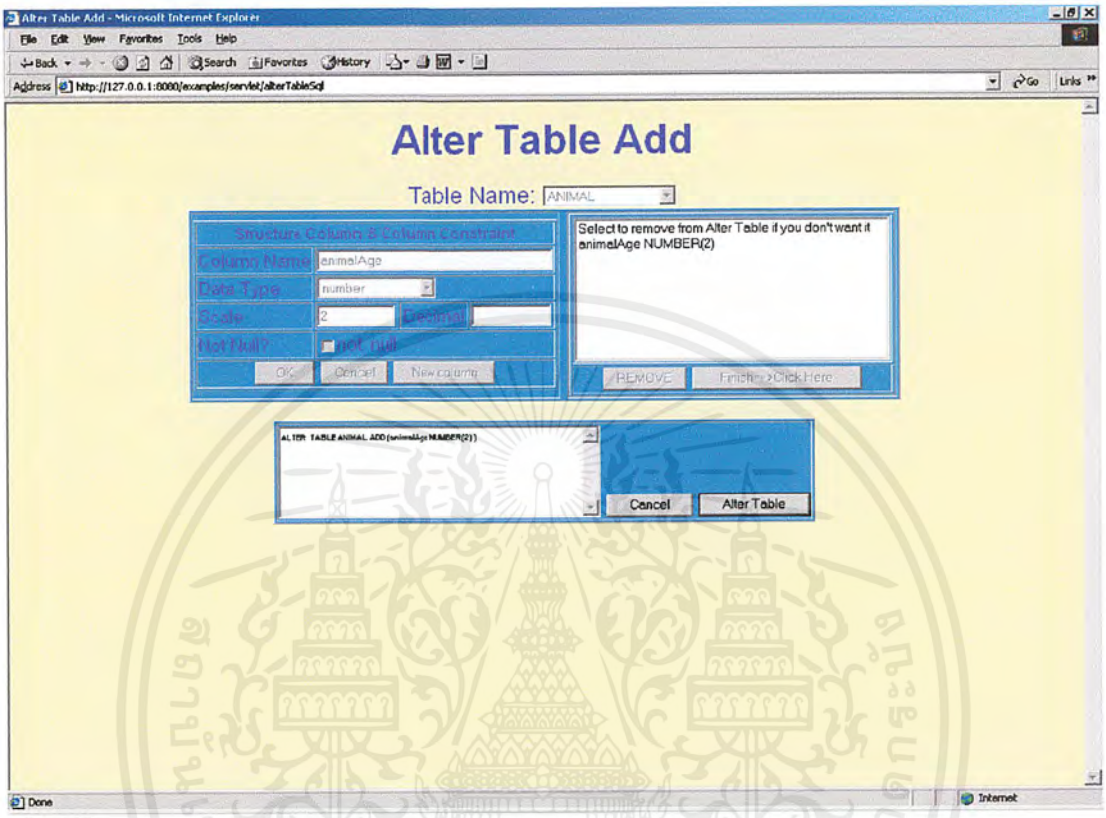
เป็นหน้าจอที่แสดงผลหลังจากลบตารางออกจากฐานข้อมูล เมื่อลบตารางออกจากฐานข้อมูลสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และตารางทั้งหมดหลังจากที่ได้ลบตาราง ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.23



ภาพที่ 4.23 แสดงหน้าจอ Result Drop Table

4.9 หน้าจอ Alter Table

เป็นหน้าจอของคำสั่งที่ใช้ในการเพิ่มเติมฟิลด์ (คอลัมน์) ใหม่เข้าไปในตาราง โดยคอลัมน์ใหม่จะมีค่าเริ่มต้นเป็นค่าว่าง มีหน้าจอดังภาพที่ 4.24



ภาพที่ 4.24 แสดงหน้าจอ Alter Table

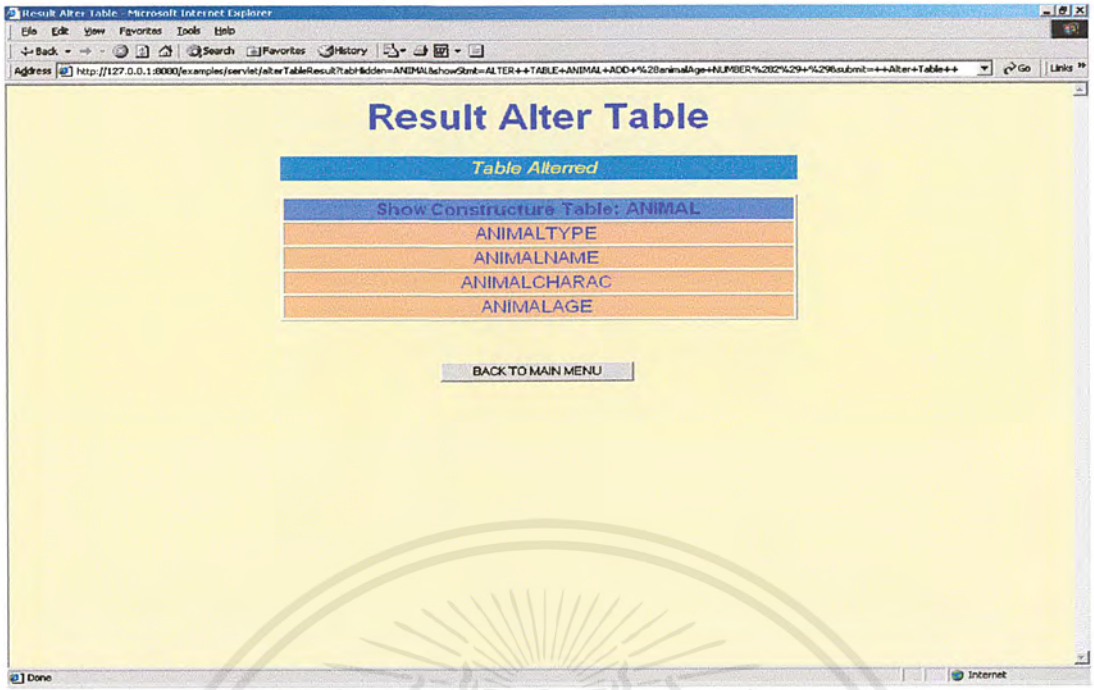
มีรายละเอียดของหน้าจอดังนี้

- Table name กำหนดชื่อตาราง โดยใส่ชื่อตารางที่จะสร้าง
- Structure Column & Column Constraint เป็นส่วนของการกำหนดโครงสร้างให้แต่ละคอลัมน์
- Column name กำหนดชื่อคอลัมน์ โดยใส่ชื่อ Fieldname
- Data type เป็นการกำหนดชนิดของข้อมูลในคอลัมน์นั้น โดยทำการคลิกเลือกชนิดของข้อมูล
- Scale เป็นการกำหนดขนาด หรือความยาวของข้อมูลที่คอลัมน์นี้สามารถรองรับได้ โดยใส่ตัวเลขลงไป
- Decimal เป็นการกำหนดจุดทศนิยม ในกรณีที่ เลือก Data type เป็น number หรือ char โดยใส่ตัวเลขลงไป

- Not null เป็นการกำหนดให้คอลัมน์นี้ต้องมีการใส่ค่าข้อมูล ห้ามเป็นค่าว่าง โดยคลิกเลือกที่ check box แต่ถ้าไม่คลิกเลือก คอลัมน์นั้นก็จะมีสถานะเป็น null
- ปุ่ม O.K กดปุ่มเพื่อเสร็จสิ้นการกำหนดโครงสร้างของคอลัมน์ 1 คอลัมน์
- ปุ่ม Cancel กดปุ่มเพื่อยกเลิกการกำหนดโครงสร้างของคอลัมน์
- ปุ่ม New column กดปุ่มนี้เมื่อต้องการเพิ่มคอลัมน์อีก
- List ด้านขวา เป็นส่วนที่แสดงถึงรูปแบบโครงสร้างของคอลัมน์ที่ได้กำหนดไว้ โดยจะแสดงให้เห็นเมื่อกดปุ่ม O.K
- ปุ่ม Remove เป็นปุ่มที่ใช้สำหรับลบคอลัมน์ที่ได้กำหนดโครงสร้างแล้วออกไปจาก List ด้านขวา-บน โดยคลิกเลือกที่รายการที่ต้องการจะ Remove แล้วกดปุ่ม Remove
- ปุ่ม Finish Click Here เป็นปุ่มสำหรับตกลงว่าได้ทำการกำหนดโครงสร้างคอลัมน์ที่จะเพิ่ม ครบเรียบร้อยแล้ว
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Alter โดยจะแสดงให้เห็นหลังจากกดปุ่ม Finish Chick here
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Alter ที่ได้กำหนดมานั้นทั้งหมด
- ปุ่ม Alter Table เป็นปุ่มสำหรับส่งคำสั่ง Alter Table นี้ไปยังฐานข้อมูล

4.9.1 **หน้าจอ** Result Alter Table

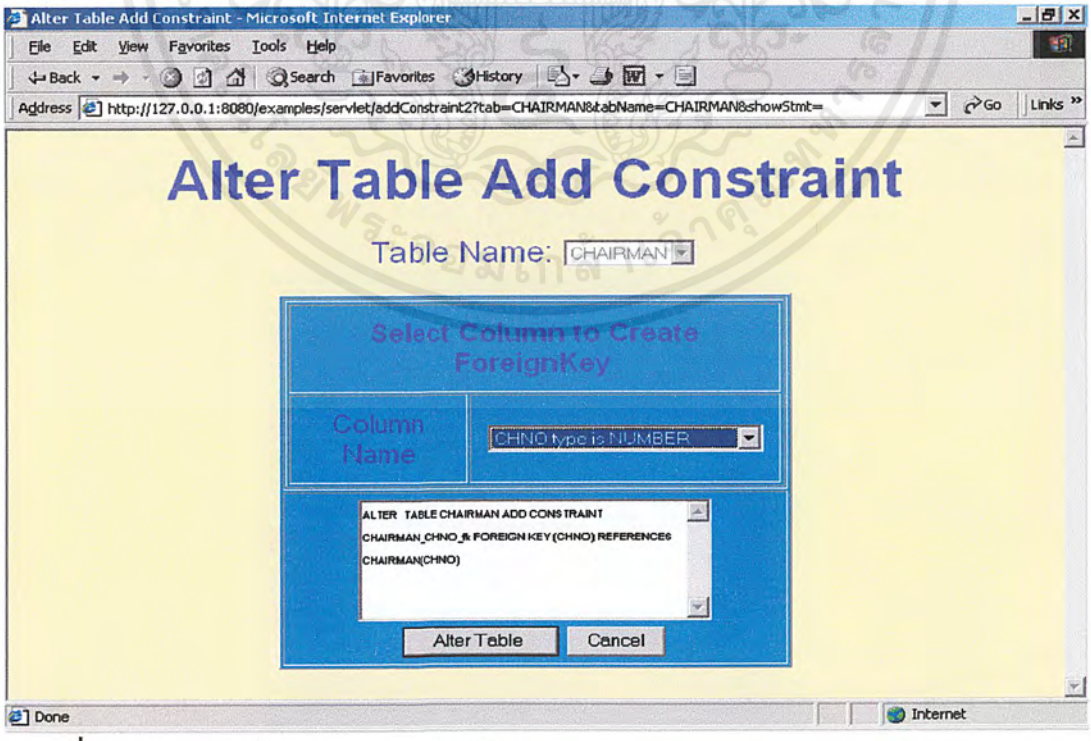
เป็นหน้าจอที่แสดงผลหลังจากเพิ่มเติมฟิลด์ (คอลัมน์) ใหม่เข้าไปในตาราง เมื่อเพิ่มเติมฟิลด์ (คอลัมน์) ใหม่เข้าไปในตารางสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงฟิลด์ (คอลัมน์) ทั้งหมดหลังจากที่ได้เพิ่มเติมฟิลด์ (คอลัมน์) ใหม่เข้าไปในตาราง ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.25



ภาพที่ 4.25 แสดงหน้าจอ Result Alter Table

4.10 หน้าจอ Add ForeignKey

เป็นหน้าจอของคำสั่งที่ใช้ในการสร้างคีย์รองในตาราง โดยคีย์รองนั้นจะต้องเป็นคีย์หลักหรือต้องเป็นค่าไม่ซ้ำ (Unique) มีหน้าจอดังภาพที่ 4.26



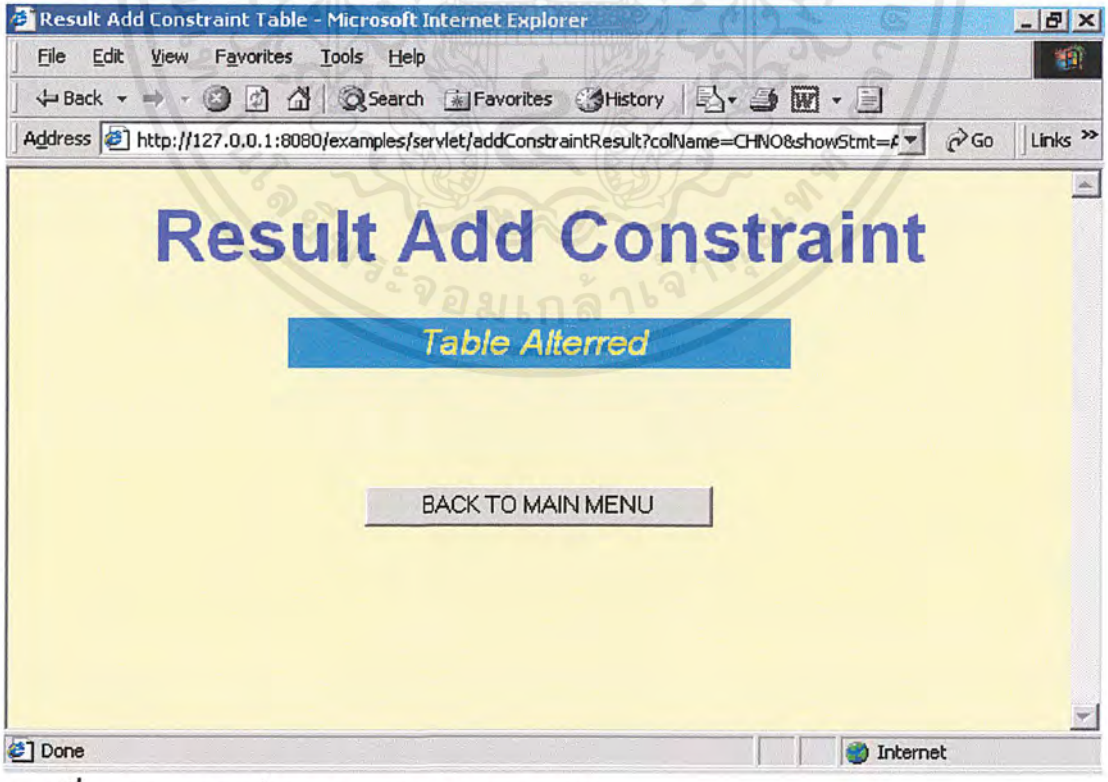
ภาพที่ 4.26 แสดงหน้าจอ Add ForeignKey

มีรายละเอียดของหน้าจอดังนี้

- Table name ให้เลือกชื่อตารางที่จะสร้างคีย์รอง
- Select Column to Create ForeignKey เป็นส่วนของการเลือกคอลัมน์ที่จะนำมาสร้างเป็นคีย์รอง
- Column name ให้เลือกชื่อคอลัมน์ที่จะนำมาสร้างเป็นคีย์รอง
- Text Area ด้านล่างจะแสดงรูปแบบของคำสั่ง Alter Table Add Constraint โดยจะแสดงให้เห็นหลังจากเลือกคอลัมน์ที่จะนำมาสร้างเป็นคีย์รอง
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Alter Table Add Constraint ที่ได้กำหนดมานั้นทั้งหมด
- ปุ่ม Alter Table เป็นปุ่มสำหรับส่งคำสั่ง Alter Table Add Constraint นี้ไปยังฐานข้อมูล

4.10.1 **หน้าจอ** Result Add ForeignKey

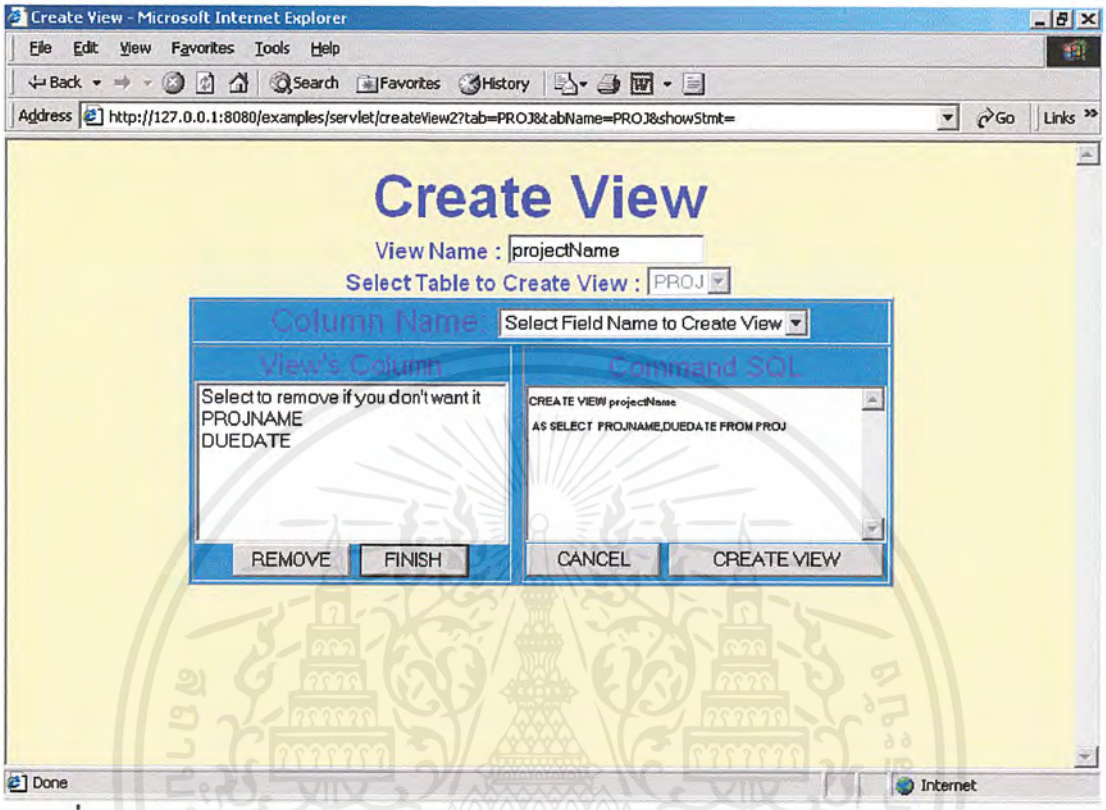
เป็นหน้าจอที่แสดงผลหลังจากสร้างคีย์รองในตาราง เมื่อสร้างคีย์รองในตารางสำเร็จ จะแสดงข้อความ Table Altered ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.27



ภาพที่ 4.27 แสดงหน้าจอ Result Add Constraint

4.11 หน้าจอ Create View

เป็นหน้าจอของคำสั่งที่ใช้ในการสร้างวิวในฐานข้อมูล มีหน้าจอดังภาพที่ 4.28



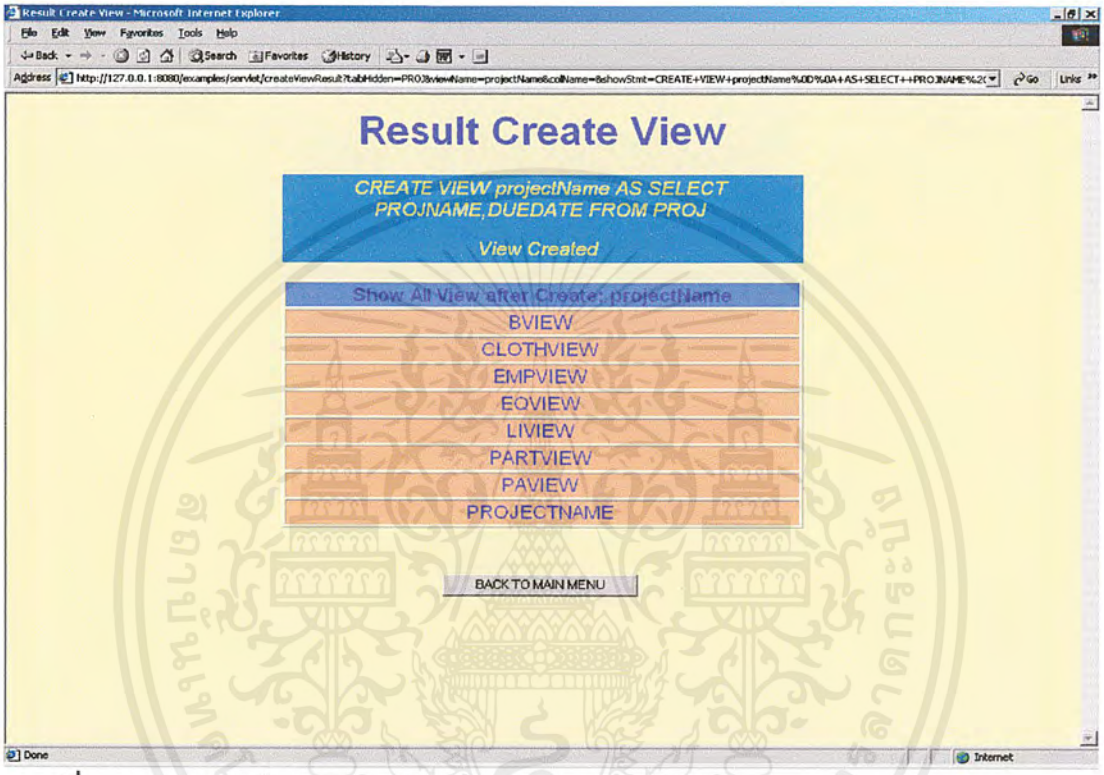
ภาพที่ 4.28 แสดงหน้าจอ Create Veiw

มีรายละเอียดของหน้าจอดังนี้

- View name ให้ใส่ชื่อวิวที่จะสร้าง
- Select Table to Create View ให้เลือกตารางที่จะสร้างวิว
- Column Name ให้เลือกคอลัมน์ที่จะสร้างวิว
- List ทางด้านซ้าย จะแสดงคอลัมน์ที่ถูกนำไปสร้างเป็นคอลัมน์ในวิว
- ปุ่ม REMOVE สามารถที่จะเลือกรายการคอลัมน์ใน List ทางด้านซ้ายออกไปได้ ถ้าไม่ต้องการ
- ปุ่ม FINISH เมื่อเลือกคอลัมน์ได้เสร็จแล้วจึงกดปุ่มนี้
- List ทางขวามือ จะแสดงคำสั่ง SQL COMMAND หลังจากกดปุ่ม FINISH
- ปุ่ม CANCEL เมื่อกดปุ่มนี้จะทำการลบ SQL COMMAND ออกจาก List ทางขวามือ และทำการยกเลิกรายการคอลัมน์ที่ได้ทำการเลือกไว้ทั้งหมด
- ปุ่ม CREATE VIEW เป็นปุ่มสำหรับส่งคำสั่ง CREATE VIEW นี้ไปยังฐานข้อมูล

4.11.1 **หน้าจอ** Result Create View

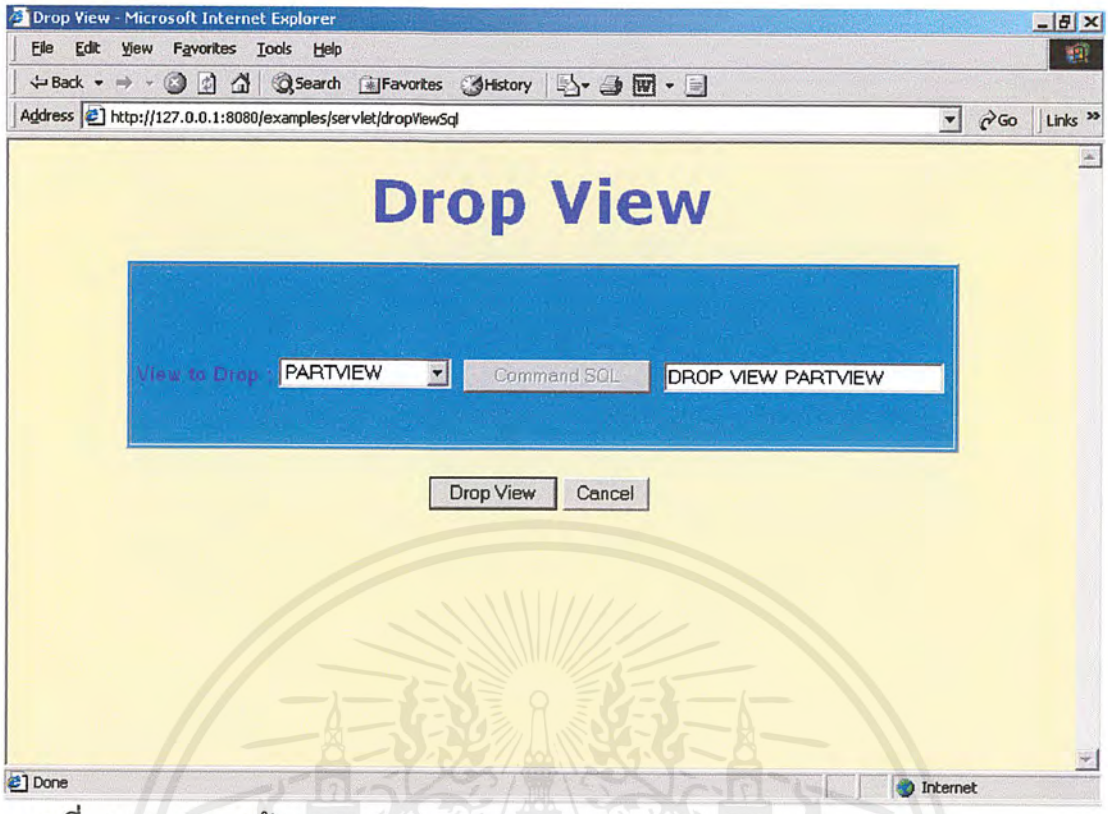
เป็นหน้าจอที่แสดงผลหลังจากสร้างวิวในฐานข้อมูล เมื่อสร้างวิวในฐานข้อมูลสำเร็จ จะแสดงคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และแสดงวิวทั้งหมดหลังจากที่ได้สร้างวิวใหม่เข้าไปในฐานข้อมูล ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.29



ภาพที่ 4.29 แสดงหน้าจอ Result Create View

4.12 หน้าจอ Drop View

เป็นหน้าจอของคำสั่งที่ใช้ลบวิวออกจากฐานข้อมูล มีหน้าจอดังภาพที่ 4.30



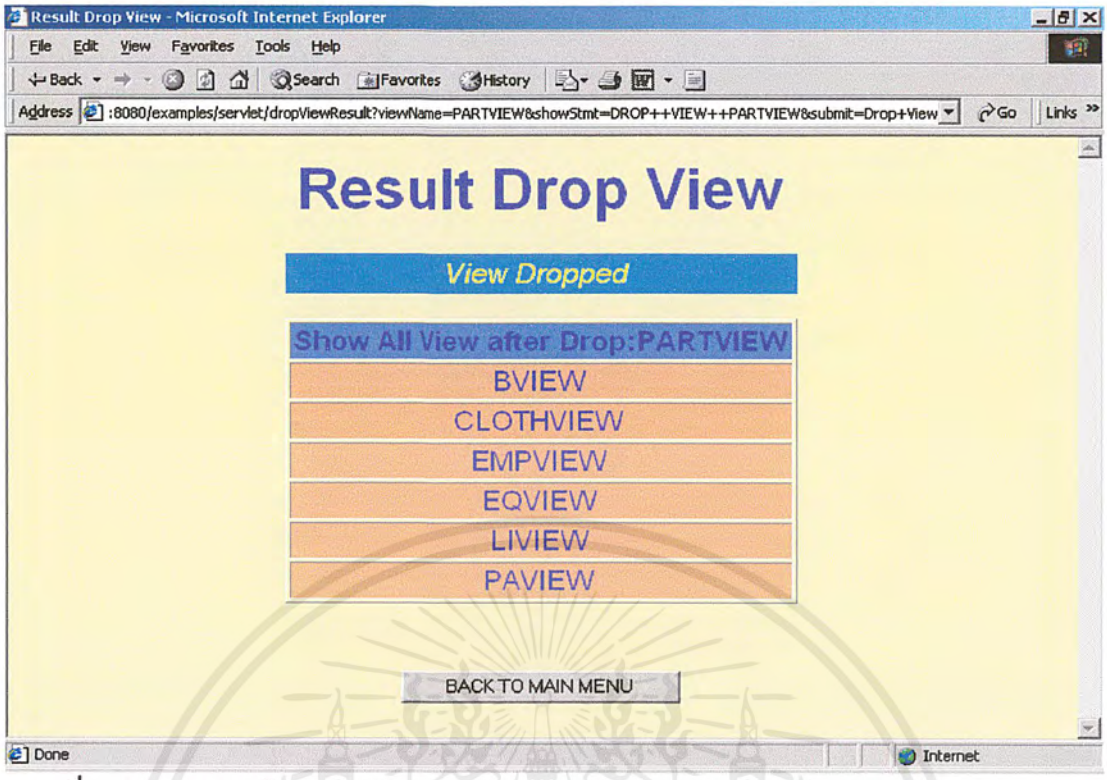
ภาพที่ 4.30 แสดงหน้าจอ Drop View

มีรายละเอียดของหน้าจอดังนี้

- View to Drop ใช้สำหรับเลือกชื่อวิวที่ต้องการจะลบ
- ปุ่ม Command SQL เป็นปุ่มกดเพื่อให้แสดงรูปแบบของคำสั่ง Drop View ที่ Text Area ด้านขวา
- ปุ่ม Drop View เป็นปุ่มสำหรับส่งคำสั่ง Drop View นี้ไปยังฐานข้อมูล
- ปุ่ม Cancel เป็นปุ่มสำหรับยกเลิกคำสั่ง Drop Table ที่ได้เลือกมานั้นทั้งหมด

4.12.1 หน้าจอ Result Drop View

เป็นหน้าจอที่แสดงผลหลังจากลบวิวออกจากฐานข้อมูล เมื่อลบวิวออกจากฐานข้อมูลสำเร็จจะมีคำสั่ง SQL ที่ได้ Generate ไว้จากหน้าจอก่อนหน้านี้ และวิวทั้งหมดหลังจากที่ได้ลบวิว ส่วนด้านล่างของหน้าจอจะมีปุ่มที่สามารถกลับไป Main Menu ได้ มีหน้าจอดังภาพที่ 4.31



ภาพที่ 4.31 แสดงหน้าจอ Result Drop View

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการศึกษา

คณะผู้พัฒนาได้พัฒนาโปรแกรมแบบจำลองการใช้ภาษา SQL ซึ่งใช้ JavaScript , Java Servlet และHTML เป็นส่วนติดต่อกับผู้ใช้ และใช้ Java Servlet และ JDBC ในการประมวลผลและติดต่อกับฐานข้อมูลโปรแกรมแบบจำลองการใช้ภาษา SQL ทีพัฒนาขึ้นนี้ นับว่าเป็นเครื่องมือสำหรับใช้งานบนเครือข่ายอินเตอร์เน็ต ซึ่งจะช่วยให้ผู้ใช้งานสามารถเข้าใจการใช้งานได้อย่างสะดวก รวดเร็ว และสามารถทำการเรียนรู้คำสั่ง SQL Command ได้ โดยที่ไม่จำเป็นต้องมีความรู้ทางด้าน SQL มากนัก

โปรแกรมแบบจำลองการใช้ภาษา SQL ที่พัฒนาขึ้นมาสามารถรองรับการทำงานในหน้าที่หลักๆ ดังนี้

1. การเรียกดูข้อมูล
2. การสร้างตารางเพิ่มในฐานข้อมูล
3. การเพิ่มข้อมูลในตาราง
4. การลบข้อมูลออกจากตาราง
5. การแก้ไขข้อมูลในตาราง
6. การลบตารางออกจากฐานข้อมูล
7. การเพิ่มคอลัมน์ในตาราง

5.2 อุปสรรคในการพัฒนา

การพัฒนาโปรแกรมแบบจำลองการใช้ภาษา SQL คณะผู้พัฒนาได้เลือกใช้ Hyper Text Markup Language ((HTML) , JavaScript , Java Servlet และ Java Database Connectivity (JDBC) เป็นเครื่องมือในการพัฒนา ดังนั้นผู้พัฒนาจึงจำเป็นต้องทำการศึกษาค้นคว้าการทำงานของเครื่องมือต่างๆ เหล่านั้นอย่างลึกซึ้ง และเนื่องจากแหล่งข้อมูลของ Java Servlet และ JDBC ที่เป็นภาษาไทยมีอยู่ไม่มากนักซึ่งเป็นอุปสรรคในการพัฒนา ทำให้คณะผู้จัดทำต้องใช้เวลานานในการศึกษาและค้นคว้าข้อมูล

นอกจากนี้คำสั่งในภาษา SQL นั้นมีความซับซ้อนมาก ทำให้แบบจำลองการใช้ภาษา SQL นี้ไม่สามารถตรวจสอบความถูกต้องทางไวยากรณ์ได้คลอบคลุมทั้งหมด

5.3 ข้อเสนอแนะ

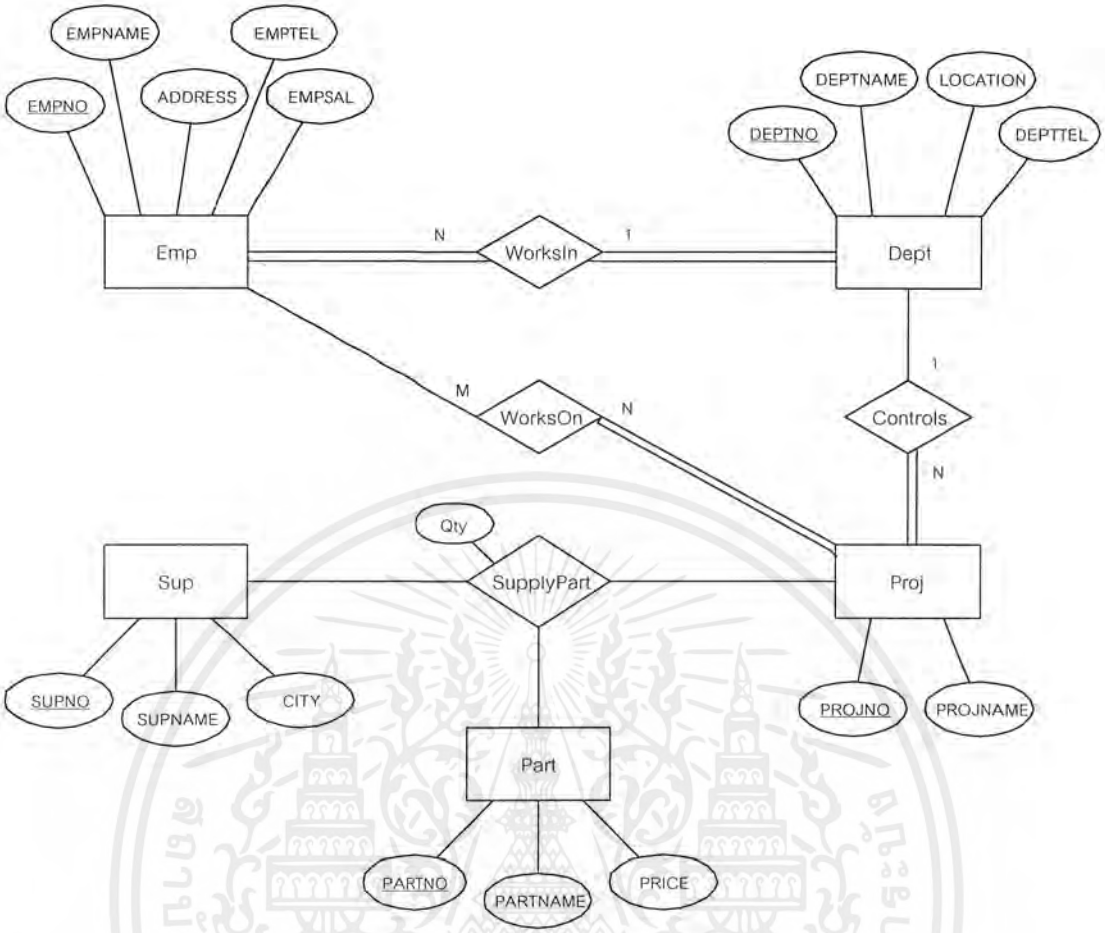
ในการพัฒนาโปรแกรมแบบจำลองการใช้ภาษา SQL นี้ ผู้ใช้โปรแกรมควรมีความรู้ในเรื่องต่อไปนี้คือ

1. ผู้ใช้โปรแกรมควรทราบรายละเอียดในการติดต่อกับฐานข้อมูลในเครื่อง Server ได้แก่ หมายเลข IP Aderss , Port
2. ผู้ใช้โปรแกรมควรทราบรายละเอียดในฐานข้อมูล ได้แก่ รายละเอียดของตารางทั้งหมดในฐานข้อมูล และรายละเอียดของชื่อคอลัมน์ทั้งหมดของแต่ละตารางในฐานข้อมูล
3. ผู้ใช้โปรแกรมควรมีความรู้ในการเขียน SQL Command เบื้องต้นพอสมควร
4. ควรมีความเข้มงวดในการควบคุมการใช้งานของผู้ใช้
5. การเพิ่มประสิทธิภาพของโปรแกรมเพื่อให้มีการตอบสนองของระบบที่รวดเร็ว

อย่างไรก็ตาม โปรแกรมนี้ยังมีข้อจำกัด บางประการ คือ ยังไม่สามารถทำการรองรับ SQL Command ได้ทุกคำสั่ง ซึ่งจะไม่สามารถทำงานตามคำสั่งที่มีความซับซ้อนตามความต้องการของผู้ใช้ได้

ภาคผนวก ก.

รายละเอียดการออกแบบฐานข้อมูลที่ใช้ในระบบ



ภาพที่ ก.1 แสดง ER-Diagram ของฐานข้อมูลที่ใช้ในระบบ

จากการออกแบบฐานข้อมูลโดยใช้ ER-Diagram สามารถพิจารณาถึงความสัมพันธ์ต่างๆ และนำมาสร้างเป็นตารางได้ทั้งหมดดังนี้

ตารางที่ ก.1 แสดงรายชื่อของตารางทั้งหมดในระบบ

Field	Table Name	Description
1	Emp	ตารางข้อมูลรายละเอียดของพนักงานในบริษัท
2	Dept	ตารางข้อมูลรายละเอียดของแผนกในบริษัท
3	Proj	ตารางข้อมูลรายละเอียดของโครงการในบริษัท
4	Sup	ตารางข้อมูลรายละเอียดของผู้ผลิต
5	Part	ตารางข้อมูลรายละเอียดของสินค้า
6	SupplyPart	ตารางข้อมูลรายละเอียดของสินค้าที่ใช้ในโครงการ
7	WorksOn	ตารางข้อมูลรายละเอียดของพนักงานที่รับผิดชอบโครงการ

รายละเอียดตารางฐานข้อมูลต่างๆมีดังนี้

ตารางที่ ก.2 ชื่อตาราง Emp แสดงข้อมูลรายละเอียดของพนักงานในบริษัท

Fieldname	Type	Width	Decimal	Description	Key	Comment
EMPNO	Number	3	0	รหัสพนักงาน	P.K	
EMPNAME	Varchar 2	30	0	ชื่อและนามสกุลพนักงาน		
ADDRESS	Varchar 2	45	0	ที่อยู่ของพนักงาน		
EMPTEL	Number	9	0	เบอร์โทรศัพท์ของพนักงาน		
EMPSAL	Number	8	2	เงินเดือนของพนักงาน		
DEPTNO	Number	3	0	รหัสแผนกที่พนักงานสังกัด		F.K

ตารางที่ ก.3 ชื่อตาราง Dept แสดงข้อมูลรายละเอียดของแผนกในบริษัท

Fieldname	Type	Width	Decimal	Description	Key	Comment
DEPTNO	Number	3	0	รหัสแผนก	P.K	
DEPTNAME	Varchar 2	30	0	ชื่อแผนก		
LOCATION	Varchar 2	45	0	ที่ตั้งของแผนก		
DEPTTEL	Number	9	0	เบอร์โทรศัพท์ของแผนก		

ตารางที่ ก.4 ชื่อตาราง Proj แสดงข้อมูลรายละเอียดของโครงการในบริษัท

Fieldname	Type	Width	Decimal	Description	Key	Comment
PROJNO	Number	3	0	รหัสของโครงการ	P.K	
PROJNAME	Varchar 2	30	0	ชื่อโครงการ		
DEPTNO	Number	3	0	รหัสแผนก		F.K

ตารางที่ ก.5 ชื่อตาราง Sup แสดงข้อมูลรายละเอียดของผู้ผลิต

Fieldname	Type	Width	Decimal	Description	Key	Comment
SUPNO	Number	3	0	รหัสของผู้ผลิต	P.K	
SUPNAME	Varchar 2	30	0	ชื่อของผู้ผลิต		
CITY	Varchar 2	45	0	ชื่อเมืองที่เป็นที่ตั้งของผู้ผลิต		

ตารางที่ ก.6 ชื่อตาราง Part แสดงข้อมูลรายละเอียดของสินค้า

Fieldname	Type	Width	Decimal	Description	Key	Comment
PARTNO	Number	3	0	รหัสของสินค้า	P.K	
PARTNAME	Varchar 2	30	0	ชื่อของสินค้า		
PRICE	Number	8	2	ราคาของสินค้า		

ตารางที่ ก.7 ชื่อตาราง SupplyPart แสดงข้อมูลรายละเอียดของสินค้าที่ใช้ในโครงการ

Fieldname	Type	Width	Decimal	Description	Key	Comment
SUPNO	Number	3	0	รหัสของผู้ผลิต	P.K	F.K
PARTNO	Number	3	0	รหัสของสินค้า	P.K	F.K
PROJNO	Number	3	0	รหัสของโครงการ	P.K	F.K
QTY	Number	7	0	จำนวนสินค้าที่ใช้ในโครงการ		

ตารางที่ ก.8 ชื่อตาราง WorksOn แสดงข้อมูลรายละเอียดของพนักงานที่รับผิดชอบโครงการ

Fieldname	Type	Width	Decimal	Description	Key	Comment
EMPNONO	Number	3	0	รหัสพนักงาน	P.K	F.K
PROJNO	Number	3	0	รหัสโครงการ	P.K	F.K

บรรณานุกรม

คอกซ์, โทมาส บี. 2542. **คู่มือออราเคิล 8.** แปลโดย พิชัย จันทร์จรัสทอง. กรุงเทพฯ : แมคกรอ – ฮิล.

จิตเกษม พัฒนาศิริ. ม.ป.ป. **เสริมแต่ง home page ให้มีชีวิตด้วย JavaScript.** กรุงเทพฯ : Witty Group.

ดวงแก้ว สวามิภักดิ์. 2540. **ระบบฐานข้อมูล.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

วันชัย แซ่เตีย และสิทธิชัย ประสานวงศ์. 2543. **สร้าง Dynamic Web Pages ด้วย JavaScript.** กรุงเทพฯ : ซอฟท์เพรส.

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2543. **คู่มือการเขียนรายงานปัญหาพิเศษ.** กรุงเทพฯ : คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

สุปราณี ธีรไกรศรี. 2542. **HTML 4 Visual guide.** กรุงเทพฯ : Provision.

Ashton Hobbs. 1997. **Teach yourself Database Programming with JDBC in 21 days.** Indianana : Sams. Net Publishing.

Falkenberg, E.D. and Nyssen, G.M. n.d. **Introduction To IBM SQL Covering SQL/DS Release 2.** n.p.

Karl Moss. 1999. **Java Servlet.** New York : Mc graw-Hill.