โครงงานการสร้างระบบเชื่อมต่อกับอินทราเน็ตสำหรับการค้นสื่อสารสนเทศห้องสมุด

The Development of Intranet Interconnections for Library Information Retrieval

นายเสถียร ชลประเสริฐสุข
นางสาวพรทิพย์ ไชยบำรุง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2541

โครงงานการสร้างระบบเชื่อมต่อกับอินทราเน็ตสำหรับการค้นสื่อสารสนเทศห้องสมุด

The Development of Intranet Interconnections for Library Information Retrieval

โดย

นาย เสถียร ชลประเสริฐสุข

นางสาว พรทิพย์ ไชยบำรุง

อาจารย์ที่ปรึกษา

ผศ.ดร. เอื้อน ปิ่นเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โครงงานการสร้างระบบเชื่อมต่ออินทราเน็ตสำหรับการค้นสื่อสารสนเทศห้องสมุด

The Development of Intranet Interconnections for Library Information Retrieval

ผู้จัดทำ

1. นายเสถียร ชลประเสริฐสุข 38014603
2. นางสาวพรทิพย์ ไชยบำรุง 38014325

อาจารย์ที่ปรึกษา

(ผศ.ดร. เอื้อน ปิ่นเงิน)

โครงงานการสร้างระบบเชื่อมต่อกับอินทราเน็ตสำหรับการค้นสื่อสารสนเทศห้องสมุด

นายเสถียร ชลประเสริฐสุข 38014603
นางสาวพรทิพย์ ไชยบำรุง 38014325
ผศ.ดร. เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เรียบเรียงขึ้นตามรายละเอียดของโครงงานการสร้างระบบเชื่อมต่อกับอินทราเน็ตสำหรับการสื่อสารสนเทศห้องสมุด ซึ่งเป็นโครงงานที่จัดทำขึ้นเพื่อพัฒนาโปรแกรมที่ใช้ในการค้นหาข้อมูลของห้องสมุดบนเครือข่ายคอมพิวเตอร์ โดยมีขอบเขตเพื่อใช้กันภายในองค์กร อันจะมีส่วนทำให้เกิดความสะดวกและรวดเร็วในการค้นหาข้อมูลของห้องสมุด ซึ่งแบ่งออกเป็น 2 ส่วน โดยส่วนแรกได้แก่ ส่วนที่จะช่วยในการติดต่อระหว่างผู้ใช้กับเซิร์ฟเวอร์ โดยผ่านโปรแกรมซีจีไอ ที่มีหน้าที่ในการรับและส่งข้อมูลกันระหว่างผู้ใช้กับเซิร์ฟเวอร์ และส่วนที่สองได้แก่ ส่วนจัดเก็บข้อมูลและส่วนค้นหาข้อมูลซึ่งถูกเก็บอยู่ในฐานข้อมูล โดยจะทำการค้นหาข้อมูลตามเงื่อนไขต่างๆ ที่ผู้ใช้เป็นผู้กำหนดขึ้นจากข้อมูลที่ถูกเก็บอยู่ในฐานข้อมูล

The Development of Intranet Interconnections for Library Information Retrieval

Satean Chontrasurtsuk

Porntip Chaibumrung

Asst. Prof. Dr. Ouen Pin-ngern Advisor

ABSTRACT

This thesis covers all information about The Development of Intranet Interconnections for Library Information Retrieval that has a purpose to develop program to search an information of media in the library that use Intranet technology. The project has 2 main parts. The first part is communication between user and server , receiving user request and retrieving response from server . The second part is Database and Query Information , searching data in database from any condition that user request .

กิตติกรรมประกาศ

คณะผู้จัดทำขอแสดงความนับถือ และขอบพระคุณผู้มีพระคุณทุกท่านที่ทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับคำแนะนำรวมทั้งความช่วยเหลือจากบุคคลต่อไปนี้

ผศ.ดร.เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา

บิดา มารดา รุ่นพี่ และ เพื่อนๆ ที่คอยให้ความช่วยเหลือ รวมทั้ง กำลังใจตลอดมา

คณะผู้จัดทำ

นายเสถียร ชลประเสริฐสุข

นางสาวพรทิพย์ ไชยบำรุง

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1. ความสำคัญ และที่มา

ในปัจจุบันเทคโนโลยีทางด้านอินเตอร์เน็ต ได้เป็นที่รู้จัก และนิยมใช้อย่างแพร่หลายเกือบทั่วโลก พร้อมทั้งมีแนวโน้มอัตราการเจริญเติบโตเพิ่มขึ้นเรื่อยๆ อินเตอร์เน็ตกล่าวได้ว่าเป็นแหล่งรวมข้อมูลขนาดใหญ่ และวิธีการใช้งานก็มีการพัฒนาที่จะให้สะดวกต่อผู้ใช้งานมากที่สุด ด้วยความที่อินเตอร์เน็ตเจริญเติบโตอย่างรวดเร็ว ทุกๆคนมีความต้องการที่จะใช้งาน ด้วยเหตุนี้แม้แต่ภายในองค์กรขนาดเล็กก็มีความต้องการที่จะใช้งาน หรือพัฒนาระบบอินเตอร์เน็ตขนาดเล็กขึ้นใช้ภายในองค์กร เพื่อประโยชน์ทางด้านการแบ่งปันข้อมูล หรือการติดต่อสื่อสารที่รวดเร็วเป็นการภายใน เพราะจะเป็นการลดภาระที่จะต้องเช่าคู่สายในการเชื่อมต่ออินเตอร์เน็ตนอกองค์กร

ห้องสมุดถือว่าเป็นแหล่งรวมของข้อมูลที่สำคัญ สำหรับทุกๆหน่วยงาน ถ้าข้อมูลต่างๆภายในองค์กรสามารถให้ทุกคนในองค์กรแบ่งปันกันใช้ได้ก็จะก่อให้เกิดประโยชน์สูงสุด และเพื่อเป็นการประหยัดงบประมาณ จึงได้พัฒนาแอปพลิเคชั่นชันนี้ขึ้นมา

1.2. วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาวิถีทางที่จะนำเอาข้อมูลที่อยู่ภายในฐานข้อมูลของออร์ราเคิ้ลออกมา โดยการเข้าถึงข้อมูลด้วยการใช้บราวซ์เซอร์

1.2.2 ศึกษาวิธีการสร้างสคริปต์ในการเขียนการติดต่อระบบฐานข้อมูล โดยผ่านดีบีดี ออร์ราเพิลว่าจะต้องมีการเขียนอย่างไรบนแพลฟอร์มของเซิฟเวอร์ที่มีอยู่

1.3. ขอบเขตของงานวิจัย

งานวิจัยนี้จะสร้างระบบของการติดต่อฐานข้อมูลของออร์ราเคิ้ลที่มีอยู่แล้ว ให้สามารถเข้าถึงได้โดยการเรียกใช้จากบราวซ์เซอร์ และเรียกใช้จากที่ใดๆก็ได้ภายในองค์กร โดยจะสร้างระบบที่ทำงานบนข้อกำหนดต่อไปนี้

- Operating System: Sun O.S. v.5.6
- Database: Oracle v.7.1.4.1.0
- HTTP Server: Apache Server
- Database Driver: DBD::Oracle v.0.59
- Database Interface: DBI::Oraperl v.1.06
- Script Language: Perl v.5

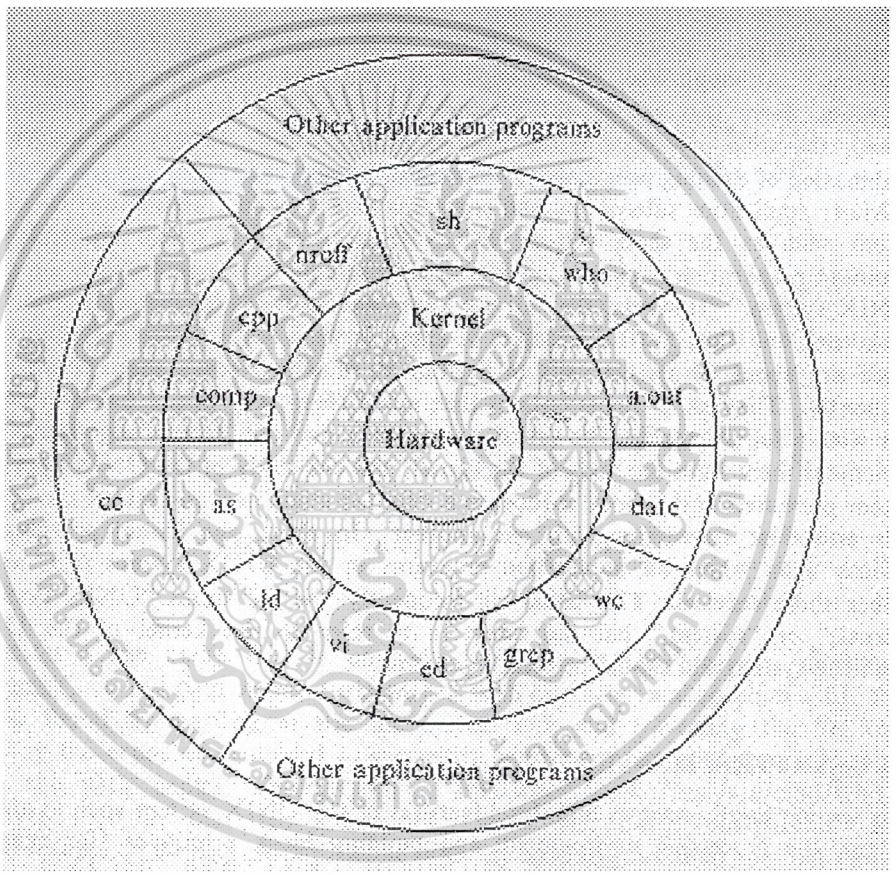
1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลักๆคือ หลักการการทำงานของระบบปฏิบัติการยูนิกซ์ ระบบการทำงานของเวอร์ไวด์เว็บที่ใช้ในการติดต่อกับฐานข้อมูลของออร์ราเคิ้ล และวิธีการในการจัดตั้งการเชื่อมต่อฐานข้อมูล โดยการใช้ดีบีดี(DBD) ศึกษาการเขียนภาษาเพิล ซึ่งมีรายละเอียดดังในบทที่ 2, 3, 4 และ 5 จากนั้นก็จะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบสถาปัตยกรรมของระบบ และทำการติดตั้งซึ่งมีรายละเอียดในส่วนของ การดำเนินงานของโครงงาน

บทที่ 2
ความรู้เบื้องต้นเกี่ยวกับยูนิกซ์ (UNIX)

ยูนิกซ์ เป็นระบบที่เริ่มได้รับความนิยมตั้งแต่ปี 1969 สามารถทำงานได้บนเครื่องตั้งแต่ ระดับไมโครคอมพิวเตอร์ จนถึงระดับเมนเฟรม ซึ่งตัวระบบจะแบ่งออกได้เป็น 2 ส่วน ส่วนแรกจะประกอบโปรแกรมและบริการ (service) ต่างๆ และส่วนที่สองได้แก่ ส่วนระบบปฏิบัติการ (Operation System) ซึ่งจะสนับสนุนการทำงานของโปรแกรมและบริการ

2.1โครงสร้างของระบบ



รูปที่ 2.1 สถาปัตยกรรมของระบบยูนิกซ์

จากรูป 2.1 แสดงถึงสถาปัตยกรรมของระบบ ฮาร์ดแวร์จะอยู่ตรงกลางของรูป โดยจะมีระบบปฏิบัติการเป็นตัวติดต่อโดยตรงกับฮาร์ดแวร์ เพื่อจัดการกับบริการและโปรแกรมทั่วไป ซึ่งสามารถมองระบบยูนิกซ์เป็นระบบที่มีการแบ่งออกเป็นเลเยอร์ (Layer) โดยในส่วนของระบบปฏิบัติการ โดยทั่วไปจะถูกเรียกว่า ซิสเท็มเคอร์เนล (system kernel) หรือเรียกสั้นๆว่า เคอร์เนล ซึ่งจะเป็นส่วนที่ติดต่อกับโปรแกรมของผู้ใช้ และเนื่องจากการที่โปรแกรมไม่ได้ขึ้นอยู่กับฮาร์ดแวร์โดยตรง มันจึงง่ายในการที่จะย้ายโปรแกรมไปมาในการใช้

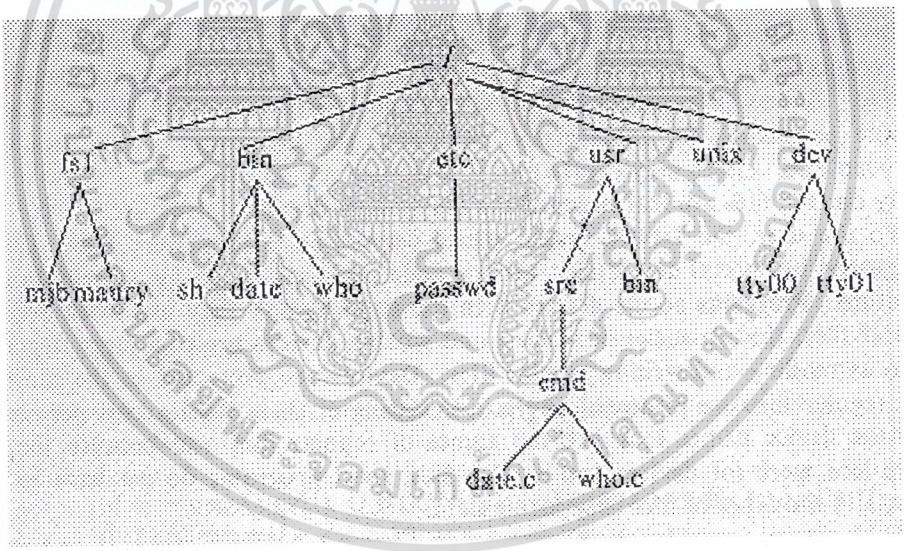
ระหว่างเครื่องคอมพิวเตอร์ที่มีฮาร์ดแวร์ต่างกัน บนระบบยูนิกซ์เหมือนกันได้ (ถ้าโปรแกรมนั้นไม่ได้มีการเฉพาะเจาะจงว่าต้องใช้ฮาร์ดแวร์พิเศษใดๆเพิ่มเติม)

โปรแกรมจำพวก เชลล์และอิดิเตอร์ (ed และ vi) ซึ่งถูกแสดงอยู่ในเลเยอร์รอบนอก จะมีการติดต่อกันกับเคอร์เนล ซึ่งจะถูกเรียกว่า ซิสเท็มคอล (System Call) โดย ซิสเท็มคอลจะเรียกให้เคอร์เนล ทำกระบวนการต่างๆ เพื่อเรียกโปรแกรมและแลกเปลี่ยนข้อมูลระหว่างเคอร์เนลและโปรแกรม ในหลายๆโปรแกรมที่ใช้กับคอนฟิกกูเลชั่นพื้นฐานของระบบ จะถูกเรียกว่า คอมมานด์ (Command)

2.2 ระบบไฟล์

ระบบไฟล์ของยูนิกซ์ มีลักษณะดังนี้

- เป็นโครงสร้างแบบลำดับขั้น
- ประกอบด้วยไฟล์ของข้อมูล
- ความสามารถที่จะลบหรือสร้างไฟล์
- การเจริญเติบโตของไฟล์แบบไดนามิค (Dynamic growth of files)
- การป้องกันไฟล์ข้อมูล
- การปฏิบัติต่ออุปกรณ์ต่อเชื่อมในลักษณะเช่นเดียวกับเป็นไฟล์



รูปที่ 2.2 โครงสร้างต้นไม้ของระบบไฟล์แบบพื้นฐาน

ระบบไฟล์ของยูนิกซ์ จะถูกจัดการในรูปแบบของโครงสร้างต้นไม้ ดังแสดงในรูปที่ 2.2 โดยจะมีโหนดที่เป็นรากเพียงโหนดเดียวและเรียกว่า รูท (root : โดยจะถูกเขียนเป็น "/") ทุกๆโหนดที่ไม่ใช่ใบของโครงสร้างระบบไฟล์นี้ จะถูกเรียกว่า ไดเรกทอรี่ของไฟล์ และไฟล์ที่อยู่ตรงใบของต้นไม้ จะเป็นได้ทั้ง ไดเรกทอรี่, ไฟล์ หรือไฟล์อุปกรณ์พิเศษต่างๆ ชื่อของไฟล์จะถูกกำหนดโดยชื่อเส้นทางเข้าถึง (path name) ซึ่งจะระบุว่าตำแหน่งของไฟล์จะถูกเก็บไว้ในตำแหน่งใดของโครงสร้างลำดับขั้นของระบบไฟล์

ชื่อเส้นทางเข้าถึง จะเป็นลำดับของคอมโพเนนท์ของชื่อที่แบ่งแยกด้วยอักขระ สแลช ("/") โดยคอมโพเนนท์จะเป็นลำดับของอักขระที่บ่งบอกถึงชื่อไฟล์ ที่ถูกบรรจุอยู่ในคอมโพเนนท์ (ไดเรกทอรี่) ก่อนหน้านั้น ชื่อของชื่อเส้นทางเข้าถึงจะเริ่มต้นด้วยอักขระสแลชและไฟล์ที่จะถูกพบได้ที่ส่วนของรูท ไปเรื่อยๆตามโครงสร้างต้นไม้ของไฟล์ ไปตามกิ่งก้านต่างๆ จนกว่าจะไปถึงชื่อของคอมโพเนนท์ ของชื่อเส้นทางเข้าถึง ตัวอย่างเช่น ชื่อเส้นทางเข้าถึง "/etc/passwd" , "/bin/who" และ "/usr/bin/cmd/who.c" ที่จะบ่งบอกถึงไฟล์ในโครงสร้างต้นไม้ดังแสดงในรูปที่ X2 โดย "/bin/passwd" และ "/usr/src/date.c" เป็นชื่อผิดเนื่องจากไม่ได้เป็นไปตามโครงสร้างต้นไม้ โดยชื่อเส้นทางเข้าถึงไม่จำเป็นจะต้องไปเริ่มต้นจากที่รูทเสมอไป แต่สามารถเริ่มจากไดเรกทอรี่ปัจจุบันที่อยู่ได้ โดยเริ่มต้นด้วยอักขระสแลชในชื่อเส้นทางเข้าถึง เช่น ถ้าเริ่มจากไดเรกทอรี่ "/dev" และ "tty01" คือชื่อเส้นทางเข้าถึงของไฟล์ที่ต้องการ ดังนั้นชื่อเส้นทางเข้าถึงเต็ม คือ "/dev/tty01"

โปรแกรมในระบบยูนิกซ์ จะไม่มีความรู้ในรูปแบบภายในที่เคอร์เนลใช้ในการเก็บไฟล์ข้อมูล โดยจะปฏิบัติต่อข้อมูลในลักษณะของสายข้อมูลแบบไบต์ที่ไม่มีรูปแบบ (unformatted stream) โดยโปรแกรมจะตีความสายข้อมูลแบบไบต์ที่มันได้มา ตามแบบวิธีที่มันต้องการ แต่จะไม่รู้เลยว่าระบบปฏิบัติการเก็บข้อมูลอย่างไร ดังนั้นรูปแบบในการเข้าถึงข้อมูลภายในไฟล์จะถูกกำหนดโดยระบบ และจะมีการกำหนดให้กับทุกๆโปรแกรม

ไดเรกทอรี่จะถูกปฏิบัติเช่นเดียวกับไฟล์ กล่าวคือ ระบบจะปฏิบัติต่อข้อมูลภายในไดเรกทอรี่เป็นสายข้อมูลแบบไบต์เช่นกัน แต่ข้อมูลจะประกอบด้วยชื่อของไฟล์ที่อยู่ในไดเรกทอรี่ โดยสามารถใช้ " ls" (ซึ่งมาจาก list the names and attributes of files) ในการดูไฟล์ในไดเรกทอรี่

การอนุญาตการเข้าถึง (permissions) ของไฟล์ จะถูกควบคุมโดยการอนุญาตเพื่อที่จะมีการเข้าถึง (access permission) ของไฟล์ การอนุญาตเพื่อที่จะมีการเข้าถึงจะแบ่งออกเป็น การอนุญาตในการอ่าน,เขียน และดำเนินการ(execute) และยังแบ่งออกเป็นอีก 3 ระดับชั้น คือ ระดับชั้นของเจ้าของไฟล์เอง , ระดับชั้นของคนในกลุ่มเดียวกับเจ้าของไฟล์ และ ระดับชั้นของบุคคลใดๆ

สำหรับผู้ใช้แล้ว ยูนิกซ์จะปฏิบัติต่ออุปกรณ์ต่างๆในรูปแบบของไฟล์ อุปกรณ์จะถูกกำหนดให้อยู่ในรูปของไฟล์อุปกรณ์พิเศษ โดยมีการจองตำแหน่งของโหนดในส่วนของไดเรกทอรี่ของตัวระบบไฟล์โปรแกรมสามารถเข้าถึงอุปกรณ์ต่างๆได้ โดยใช้ลักษณะเดียวกับการเข้าถึงไฟล์ธรรมดาทั่วไป วิธีการป้องกันการใช้อุปกรณ์จะใช้วิธีการเดียวกับไฟล์

2.3 สภาพแวดล้อมการประมวลผล

โปรแกรม คือ ไฟล์ที่สามารถดำเนินการได้ และ โพรเซส คือ ส่วนของโปรแกรมในการดำเนินการ โดยหลายๆโพรเซสสามารถดำเนินการพร้อมๆกันได้บนระบบยูนิกซ์ ซึ่งคุณสมบัตินี้ในบางครั้งจะถูกเรียกว่า มัลติโปรแกรมมิ่ง (multiprogramming) หรือ มัลติแทคกิ้ง (multitasking) โดยไม่มีการกำหนดการจำกัดจำนวนผู้ใช้ และหลายๆโปรแกรม สามารถอยู่พร้อมกันในระบบได้ หลายๆซิสเต็มคอล จะอนุญาตให้มีการสร้างโพรเซสใหม่,เลิกโพรเซสเดิม,ซิงโครไนซ์ขั้นตอนของการดำเนินการโพรเซส และควบคุมผลตอบรับที่ได้มาจากเหตุการณ์ต่างๆ การเรียกใช้ซิสเต็มคอลและการดำเนินการโพรเซสจะเป็นอิสระต่อกัน

เชลล์ (shell) เป็นตัวตีความคำสั่ง ที่ผู้ใช้ดำเนินการหลังจากที่มีการล็อกเข้าสู่ระบบ โดยเชลล์จะตีความคำในบรรทัดของคำสั่ง เช่นเดียวกับเป็นชื่อคำสั่ง

2.4 การใช้งานของผู้ใช้

ผู้ใช้แต่ละคน จะได้รับไดเรกทอรี่ส่วนตัวไว้ใช้งาน โดยภายในไดเรกทอรี่นี้ ผู้ใช้จะสามารถทำอะไรก็ได้ในนี้ ไม่ว่าจะเป็นการสร้างไฟล์,ลบไฟล์

2.5 การแบ่งประเภทของผู้ใช้

ผู้ใช้สามารถแบ่งออกเป็นประเภทใหญ่ ได้ 2 ประเภท คือ

1. รูท (root) เป็นผู้ดูแลระบบ สามารถทำการใดๆ ก็ได้ บนระบบ ไม่ว่าจะเป็นการติดตั้งโปรแกรมใหม่ การตั้งค่าต่างๆให้กับระบบ การเพิ่มหรือลบผู้ใช้ การลบโพรเซส ฯลฯ
2. ผู้ใช้ทั่วไป จะมีสิทธิใช้งานทั่วๆไป ตามกลุ่มที่รูทกำหนดไว้

2.6 การแบ่งผู้ใช้ออกเป็นกลุ่ม (group)

การแบ่งผู้ใช้ออกเป็นกลุ่ม จะมีการแบ่งเป็นกลุ่มโดยที่แต่ละกลุ่มก็จะได้สิทธิในการทำงานที่แตกต่างกันไป แล้วแต่รูทจะเป็นคนกำหนดขอบเขตในการอนุญาตให้ใช้งาน โดยผู้ใช้คนหนึ่งอาจจะมีกลุ่มอยู่มากกว่า 1 กลุ่มก็ได้

บทที่ 3
หลักการพื้นฐานของเวิลด์ไวด์เว็บ, เอชทีเอ็มแอล และ ซีจีไอ

3.1 หลักการพื้นฐานของเวิลด์ไวด์เว็บ(World Wide Web)

เวอร์ไวด์เว็บ (World Wide Web) หรือที่เราเรียกกันสั้นๆ ว่า เว็บ (Web) คือเป็นที่เก็บรวบรวมข้อมูลต่างๆ ที่อยู่ในคอมพิวเตอร์เครื่องต่างๆ ที่เชื่อมต่อกันเป็นเครือข่ายทั่วโลก ข้อมูลที่เก็บสามารถเป็นข้อมูลชนิดใดๆ ก็ได้ เช่น อักขระ, รูปภาพ, ออร์ดิโอ หรือ แม้แต่วิดีโอ ความแตกต่างระหว่าง เว็บ และลักษณะด้านอื่นๆ ของอินเตอร์เน็ตคือ เว็บ ใช้วิธีการเชื่อมต่อที่เรียกว่า "ไฮเปอร์ลิงค์ (hyperlinks)" .เมื่อการเชื่อมต่อเหล่านี้อยู่บนพื้นฐานของอักขระจึงเรียกว่า "ไฮเปอร์เท็กซ์ (hypertext)".

ความรู้เบื้องต้นที่เกี่ยวกับการทำงานของ เว็บ

1) เอชทีเอ็มแอล (HTML)

เอชทีเอ็มแอล เป็นคำย่อมาจาก ไฮเปอร์ เท็กซ์ มาร์คอัพ แลงเกวจ (Hyper Text Markup language) เป็นภาษาที่ใช้ในการทำการเชื่อมต่อ ไฮเปอร์เท็กซ์ และเอกสารสำคัญอื่นๆ

2) ยูอาร์แอล (URL)

ยูอาร์แอล เป็นคำย่อมาจาก ยูนิฟอร์ม รีซอร์ส โลเคเตอร์ (Uniform Resource Locator) เป็นตัวที่บ่งบอกที่ตั้งของทุกๆรีซอร์ส (resource) บนอินเตอร์เน็ต และโปรโตคอลเพื่อส่งมันออกมา

ตัวอย่างเช่น

"http://www.cs.uh.edu/~zsyu/me.html"

ยูอาร์แอล จะประกอบด้วย 3 ส่วน :

1. โปรโตคอลที่ใช้ในการสื่อสาร ในตัวอย่างนี้คือ "http" ซึ่งเป็นไฮเปอร์เท็กซ์ ทรานสเฟอร์ โปรโตคอล (HyperText Transfer Protocol)
2. โดเมนเนม (Domain Name) หรือชื่อของเซิร์ฟเวอร์ (Server) ในตัวอย่างนี้คือ www.cs.uh.edu
3. ชื่อพาร์ธ (path) ของไฟล์ ในตัวอย่างนี้คือ "/~zsyu/me.html"

3) เว็บบราวเซอร์ (Web Browser)

เว็บบราวเซอร์เป็นโปรแกรมที่รวบรวมความสามารถต่างๆ ที่ใช้ในการดึงข้อมูลในรูปแบบต่างๆ ภายใต้ตัวเชื่อมต่อ (Interface) เพียงตัวเดียว ซึ่งทำให้มันง่ายต่อการใช้งานและดึงข้อมูล เว็บบราวซ์เซอร์ ที่รู้จักกันดีและใช้กันอย่างแพร่หลาย เช่น เน็ตสเคป (Netscape), ไมโครซอฟต์ อินเตอร์เน็ต เอ็กซ์โพลเรอร์ (Microsoft Internet Explorer) และโมเสอิค (Mosaic)

4) เว็บเซิร์ฟเวอร์ (Web Server)

เว็บเซิร์ฟเวอร์ เป็นโปรแกรมที่ทำงาน (run) บนคอมพิวเตอร์ที่ถูกเชื่อมต่อกับอินเตอร์เน็ต เว็บเซิร์ฟเวอร์ จะเป็นตัวที่ดูการเชื่อมต่อของอินเตอร์เน็ตและรอคอยการร้องขอจาก เว็บบราวซ์เซอร์ เพื่อร้องขอเอกสาร เอชทีเอ็มแอล เมื่อตัว เว็บเซิร์ฟเวอร์ ได้รับการร้องขอแล้วมันจะไปทำการค้นหาเอกสาร เอชทีเอ็มแอล และส่งกลับไปที่ บราวซ์เซอร์ที่ร้องขอมา

เว็บเซิร์ฟเวอร์ จะให้การทำงานหลัก 4 อย่าง :

- การให้บริการเว็บเพจ(Web page)
- การทำงานโปรแกรมเกทเวย์และส่งเอาท์พุทกลับคืนไป
- ควบคุมการเข้าถึง เซิร์ฟเวอร์
- การเฝ้าดูและการเข้าถึง เซิร์ฟเวอร์

5) ระบบไคล์เอนท์เซิร์ฟเวอร์ (Client/Server) และโปรโตคอลต่างๆ

เว็บ ถูกสร้างบนรูปแบบ ไคล์เอนท์เซิร์ฟเวอร์ ที่ต้องมีการร้องขอและการตอบรับ เซิร์ฟเวอร์ ก็มีลักษณะเหมือนคลังเก็บข้อมูลทั่วๆ ไปที่รอคอยการส่งสัญญาณร้องขอจากตัวไคล์เอนท์. ไคล์เอนท์ ก็เปรียบเหมือนลูกค้า/ผู้ใช้ที่ต้องการเข้ามาค้นหาข้อมูลที่เก็บอยู่ในตัวคลังข้อมูล

ไคล์เอนท์ และ เซิร์ฟเวอร์ จะติดต่อสื่อสารกันด้วยโปรโตคอลเดียวกัน.โปรโตคอลคือกฎต่างๆ ที่ใช้เป็นข้อตกลงในการทำการติดต่อสื่อสารซึ่งกันและกัน.โปรโตคอลสำหรับเว็บ คือ เอชทีทีพี (HTTP)

เวิลด์ไวด์เว็บ (World-Wide Web) เป็นเทคโนโลยีที่นิยมมากบนอินเตอร์เน็ตซึ่ง เวิลด์ไวด์เว็บ นี้เป็นกราฟฟิกยูสเซอร์อินเทอร์เฟซ (Graphic User Interface :GUI) ทำให้ผู้ใช้งานไม่ต้องมีความรู้อะไรมาก และง่ายต่อการใช้งาน. เวิลด์ไวด์เว็บ จะถูกเขียนในรูปของไฮเปอร์เท็กซ์ มาร์คอัพ แลงเกวจ (Hyper Text Markup Language (HTML)) และใช้ไฮเปอร์ เท็กซ์ ทรานสเฟอร์ โปรโตคอล (Hyper Text Transfer Protocol (HTTP)) เป็นโปรโตคอลในการสื่อสาร

เมื่อกล่าวถึงการสื่อสารบนเน็ตเวิร์ค(Network) และการทำงานของโปโตคอลระดับเน็ตเวิร์ค ส่วนมากจะมีขั้นตอนการทำงานที่ประกอบด้วย

- Connection การเชื่อมต่อ
- Request การร้องขอ
- Response การตอบสนอง
- Close การยกเลิกการเชื่อมต่อ

3.2. ภาษาเอชทีเอ็มแอล (HTML)

การเขียนแอปพลิเคชั่น (Application) โดยใช้ภาษามาตรฐานเอชทีเอ็มแอล หรือที่มีชื่อเรียกเต็มว่า ไฮเปอร์ เท็กซ์ มาร์คอัพ แลงเกวจ (Hyper Text Markup Language). ทำให้สามารถเขียนแอปพลิเคชั่น (Application) ที่สามารถเข้าถึง (Access) จากเครื่องคอมพิวเตอร์ชนิดใดก็ได้. ซึ่ง เอชทีเอ็มแอล เป็นภาษาที่มีลักษณะที่ง่ายต่อการทำความเข้าใจ.ส่วนประกอบต่างๆ ที่จำเป็นในการสร้างแอปพลิเคชั่นก็สามารถหาได้ทั่วไป อีกทั้งการเปลี่ยนแปลงยังสามารถทำได้ง่ายและรวดเร็ว

เอชทีเอ็มแอล ไม่จำเป็นต้องใช้ตัวแปลภาษา(Compiler)ราคาแพง โดยสามารถสร้างและอ่านได้โดยใช้เท็กซ์ เอดิเตอร์ (Text editor) ธรรมดา ที่สำคัญคือ ความเปลี่ยนแปลงใดๆ ก็ตามที่จะเกิดกับมาตรฐาน เอชทีเอ็มแอล จะเป็นไปในลักษณะของการเพิ่มฟังก์ชัน (function) เข้าไปเท่านั้น ซึ่งก็คือแอปพลิเคชั่นที่เขียนขึ้นเพื่อใช้ในเวอร์ชั่นเก่า จะสามารถทำงานได้อย่างถูกต้องภายใต้เวอร์ชั่นใหม่ แต่มันก็มีข้อจำกัดที่ต้องคำนึงถึงก็คือ เอชทีเอ็มแอล ไม่ใช่ภาษาสำหรับการเขียนโปรแกรม(Programming)ที่แท้จริง ซึ่งมันจะไม่สามารถทำแอปพลิเคชั่นแบบกราฟฟิก โดยไม่ต้องใช้เวลามาก และต้องการใช้เพียงฟังก์ชั่น(Function)พื้นฐานเท่านั้น

เอชทีเอ็มแอล ยังมีความสามารถในการเป็นเกทเวย์ที่จะติดต่อกับโปรแกรมอื่น โดยคอมมอน เกทเวย์ อินเตอร์เฟซ (Common Gateway Interface (CGI)) จะทำให้สามารถเรียก โปรแกรมภายนอก (External Program) จากภายใน เอชทีเอ็มแอล ได้และยังได้ผลตอบสนองจากโปรแกรมอื่นด้วย

แอปพลิเคชั่นของเอชทีเอ็มแอล (HTML Application) จะถูกรันโดยกราฟฟิกฟอนต์เอ็นด์ (Graphical Font End) ที่เรียกว่า "เว็บบราวซ์เซอร์"(Web browser) หรือที่เรียกกันสั้นๆว่า บราวเซอร์(browser).โดยบราวเซอร์จะอ่านรหัสของเอชทีเอ็มแอล(HTML code) และแปลงมันให้อยู่ในรูปของแอปพลิเคชั่นที่มีลักษณะเป็นแบบกราฟฟิก ซึ่งสามารถใช้เม้าส์(Mouse)ได้ โดยข้อมูลอาจจะใช้การใส่เข้าไปในฟิลด์(field), การเลือกจากลิสต์(List)หรือการเลือกจากปุ่ม โดยใช้การกดปุ่มเพื่อส่งข้อมูลไปทำงาน

ในทุกๆ ชนิดของคอมพิวเตอร์จะมี บราวเซอร์ ของตัวมันเอง. โปรแกรม เอชทีเอ็มแอล ที่เขียนบนเครื่องชนิดหนึ่งจะสามารถไปรันบน บราวซ์เซอร์ ตัวอื่นๆ ได้ด้วย ตัวอย่างของ บราวซ์เซอร์ ได้แก่ เน็ตสเคป (Netscape), อินเตอร์เน็ต เอ็กซ์โพลเรอร์ (Internet Explorer), โมเสอิค (Mosaic (บราวซ์เซอร์ ตัวแรก)) และลีนุกซ์ (Lynx (บราวซ์เซอร์ แบบ text)) และตัวอื่นๆ ซึ่งยอมให้ผู้ใช้เลือก บราวซ์เซอร์ ตัวใดก็ได้ที่ต้องการใช้

ซึ่งเราสามารถควบคุมการเข้าถึงฐานข้อมูล (Database)ได้ ถ้าโปรแกรมอยู่บนเครื่องเดียวกับเซิร์ฟเวอร์ฐานข้อมูล(Database Server) โปรแกรม เอชทีเอ็มแอล จะถูกส่งไปยัง บราวซ์เซอร์ โดย เซิร์ฟเวอร์ ที่ถูกออกแบบเพื่อส่งโปรแกรมของเอชทีเอ็มแอลโดย เซิร์ฟเวอร์ จะสามารถยอมรับหรือปฏิเสธการเข้าถึงข้อมูลได้

คำสั่ง ใน เอชทีเอ็มแอล ถูกเรียกว่า 'อีลีเมนท์ (elements)' สามารถแบ่งออกได้เป็น 2 ประเภท

1. กำหนดการแสดงผลของเอกสารในส่วน บอดี้ โดยบราวเซอร์
2. กำหนดข้อมูลเกี่ยวกับเอกสารเช่นความสัมพันธ์กับเอกสารอื่นหรือหัวข้อของเอกสาร

3.2.1.อีลีเมนท์ในเอกสารแบบเอ็ชทีเอ็มแอล

คำสั่งที่สามารถทำงานได้โดย เอชทีเอ็มแอล ถูกเรียกว่า เอชทีเอ็มแอล อีลีเมนท์ หรือเรียกย่อๆ ว่า แท็ก (Tag) ใช้ในการกำหนดรูปแบบของข้อมูล ซึ่งมีรูปแบบพื้นฐาน ดังนี้

- เป็นแท็กที่อยู่ในตำแหน่งเริ่มต้นของบล็อคของข้อมูลที่ต้องการกำหนดรูปแบบ โดยมีชื่อของอีลีเมนท์อยู่ระหว่างปีกกา (Angle brackets) “< >”

- เป็นแท็กที่อยู่ในตำแหน่งสิ้นสุดของบล็อคของข้อมูล

- เอ็มตี้อีลีเมนท์ (Empty Elements)

บางอีลีเมนท์สามารถเป็นอีลีเมนท์ว่างได้ นั่นคือ ไม่ส่งผลกระทบกับบล็อคของข้อมูลอีลีเมนท์ ชนิดนี้ไม่จำเป็นต้องใช้แท็กปิดท้ายบล็อคของข้อมูล

- อัพเพอร์และโลว์เวอร์เคสอีลีเมนท์ (UpperandLowerCase Element)

การเขียน อีลีเมนท์เนม สามารถใช้ได้ทั้งตัวใหญ่และตัวเล็ก ซึ่งให้ผลเหมือนกัน เช่นการขีดเส้นในแนวนอนสามารถเขียนได้ ดังนี้

, <Em> หรือ <EM>.

อีลีเมนท์ หลายชนิดสามารถมีอาร์กิวเมนท์ เพื่อใช้ในการผ่านค่าพารามิเตอร์ต่างๆ ไปยังอีลีเมนท์ได้ อาร์กิวเมนท์ เหล่านี้ถูกเรียกว่า แอททริบิวของอีลีเมนท์ (Attributes of the element)

ตัวอย่าง

<a HREF=http://www.somewhere.ca/file.html> marked text .

จากตัวอย่าง a เป็น Element name ซึ่งปิดท้ายด้วย Tag

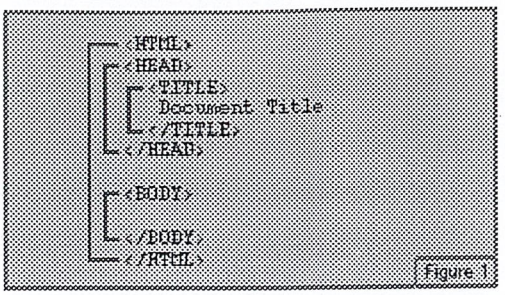
HREF เป็นแพททริบิวซึ่งใช้กำหนดค่า

3.2.2. โครงสร้างของเอกสารแบบเอ็ชทีเอ็มแอล

เอกสารแบบเอ็ชทีเอ็มแอล ประกอบด้วย 2 ส่วน คือเฮดและบอดี้

- เฮด ประกอบด้วยข้อมูลเกี่ยวกับเอกสาร
- บอดี้ ประกอบด้วยส่วนเนื้อข้อมูลที่จะแสดง

ดังแสดงในรูปที่ 3.1 แสดงถึงการวางโครงสร้างของเอกสารเอชทีเอ็มแอล



รูปที่ 3.1 แสดงโครงสร้างของภาษาเอ็ชทีเอ็ม (HTM)

3.2.3.การกำหนดลักษณะของเอกสารแบบเอ็ชทีเอ็มแอล

เมื่อ เอชทีเอ็มแอล บราวเซอร์ (Mosaic, TkWWW, Lynx, ...) ได้รับไฟล์ มันต้องรู้ว่าจะทำอย่างไรกับไฟล์เหล่านั้น วิธีการที่ง่ายที่สุดก็คือ การดูไฟล์เนม เอ็กซ์เทนชั่น (filename extension) นั่นคือ เอชทีเอ็มแอล ไฟล์จะมีส่วนขยาย (extension)เป็น เอ็ชทีเอ็มแอล (html) สำหรับยูนิกซ์และเป็นเอ็ชทีเอ็ม (htm) สำหรับดอส ซึ่งมีความสามารถกำหนดได้ไม่เกิน 3 ตัว

เอ็กซ์เท็นชั่น (Extension) มาตรฐานที่ใช้กันทั่วๆ ไป

- .html สำหรับยูนิกซ์ หรือ .htm สำหรับดอส
- .txt or .text
- .gif
- .xbm
- .xpm
- .jpeg
- .mpeg
- .au.
- .Z

บราวเซอร์ใช้ เอ็มไอเอ็มอีไทพ์ (Multipurpose Internet Mail Extension) ของ เอกสาร ในการกำหนดว่า ไฟล์ชนิดใด ต้องใช้วิธีการใดในการแสดงผล เช่น .GIF ต้องใช้โปรแกรมดูรูปภาพ (Image viewer) เป็นต้น เอชทีทีพีเซิร์ฟเวอร์จะเพิ่มไมน์ไทพ์ (MINE Type) เข้าไปในส่วนเฮดเดอร์ ของทุกไฟล์ที่ถูกร้องขอให้กับบราวเซอร์เพื่อให้บราวเซอร์ รู้ชนิดของไฟล์และวิธีที่จะจัดการกับไฟล์เหล่านี้

รายละเอียดของอีลีมนท์ต่างๆ

3.2.3.1. เฮด (HEAD)

ในส่วนเฮดนี้จะประกอบไปด้วยข้อมูลเกี่ยวกับ เอกสารแต่ไม่มีการแสดงผลเหมือนกัน กับในส่วนบอดี้

มาร์คอัพ (Mark-up) อีลีเมนท์ที่สามารถใช้ในส่วนเฮดนี้ ได้แก่

- TITLE
- ISINDEX
- NEXTID
- LINK
- BASE

3.2.3.2. ไตเติ้ล (TITLE)

ส่วน ไตเติ้ล ของเอกสาร ถูกกำหนดโดย ไตเติ้ล อีลีเมนท์ในส่วนเฮด ซึ่งมีได้เพียง ไตเติ้ล เดียวในแต่ละเอกสาร มักใช้เป็นลาเบล (Label) ของวินโดว์ในการแสดงผล การกำหนด ไตเติ้ล ของเอกสาร ควรจะมีความหมาย และไม่ยาวนัก (ควรจะน้อยกว่า 64 ตัวอักษร)

3.2.3.3. ไอเอสอินเด็กซ์ (ISINDEX)

อีลีเมนท์ นี้จะบอกตัวแสดงผลว่าเอกสาร นี้เป็นเอกสารดรรชนี (index document) นั่นคือ ในการแสดงหรือค้นเอกสารสามารถทำการค้นหาข้อมูลโดยตามคีย์เวิร์ดที่กำหนดได้ โดยการเติมเครื่องหมาย "?" ตามหลังยูอาร์แอลนั้นและถ้าหาก คีย์เวิร์ดมีหลายคำ ให้ใช้เครื่องหมาย "+" ในการแยกคำ เช่น

ยูอาร์แอลที่สามารถค้นหาข้อมูลได้

http://www.here.ca/cgi-bin/srch?instructional

การค้นหาคำว่า 'instructional' ทำได้โดยใช้เปลี่ยนแปลงยูอาร์แอลเป็น

http://www.here.ca/cgi-bin/srch?instructional

โดยปกติ ISINDEX อีลีเมนท์นี้ จะถูกสร้างโดยเซิร์ฟเวอร์ ไซด์ สคริปต์ (server-side script) โดยอัตโนมัติ สำหรับเซิร์ฟเวอร์ที่สามารถทำการค้นหาได้เท่านั้น ดังนั้นจึงไม่ควรใส่อีลีเมนท์นี้เข้าไปเองในเอกสาร

3.2.3.4. NEXTID

แท็ก (Tag) นี้เป็นแอททริบิวเดี่ยวอยู่ในส่วนเฮดของเอกสาร ถูกออกแบบมาเพื่อใช้สำหรับ ออโตเมติค ไฮเปอร์เท็กซ์ เอดิเตอร์ (Automatic HyperText Editors)

Old anchor ids จะไม่ถูกนำกลับมาใช้ใหม่เมื่อมีการเปลี่ยนแปลงเอกสารขณะที่ เอกสารอื่นอ้างอิงถึง เอกสารเหล่านี้

3.2.3.5. ลิงค์ (LINK)

ลิงค์ (LINK) อีลีเมนท์ใช้สำหรับบอกความสัมพันธ์ระหว่างเอกสารกับเอกสาร หรือ ออปเจค อื่นในปัจจุบันไม่ค่อยใช้กันแล้ว

ลิงค์ (LINK) อีลีเมนท์เป็นอีลีเมนท์ว่างมี แอททริบิวต่างๆ ดังนี้

- HREF กำหนดเอกสารหรือส่วนหนึ่งของเอกสารที่ต้องการติดต่อ
- NAME สำหรับตั้งชื่อลิงค์ (LINK) เพื่อเป็นปลายทางสำหรับเอกสารอื่น
- REL อธิบายความสัมพันธ์ที่ถูกกำหนดโดยการเชื่อมต่อนี้
- REV คล้ายๆ กับ REL แต่เป็นความสัมพันธ์ที่ตรงข้ามกัน
- URN กำหนดชื่อของยูนิฟอร์ม (Uniform) รีซอร์สเนม
- TITLE ใช้เป็น ไตเติ้ล ของ HREF แอททริบิวของลิงค์ (LINK) อีลีเมนท์
- METHOD อธิบายเอ็ชทีทีพี (HTTP) เมธอททีออปเจค อ้างอิงถึงใน HREF ของลิงค์ (LINK) อีลีเมนท์

3.2.3.6. เบส (BASE)

อีลีเมนท์นี้ใช้สำหรับเก็บออคิจินอบ ยูอาร์แอลของเอกสาร ซึ่งจะทำให้สามารถย้าย เอกสารไปยังไดเรคทอรีหรือไซท์อื่น มีแอททริบิวเดียว คือ HREF ใช้สำหรับเก็บ เบสยูอาร์แอลของเอกสาร เพื่อใช้กับ พาร์เชี่ยล (Partial) ยูอาร์แอลภายในเอกสาร โดยใช้เบส (Base) ยูอาร์แอลเป็นจุดเริ่มต้น

3.2.3.7. เมต้า (META)

อีลีเมนท์ที่ใช้ในการกำหนดเมต้าอินฟอร์เมชั่น (Information about Information) ภายในเอกสารโดยมีแอททริบิวต่างๆ ดังนี้

- HTTP-EQUIV แอททริบิวนี้ใช้ติดต่อระหว่างเมต้า (META) อีลีเมนท์นี้ เพื่อตอบสนองต่อโปรโตคอลเฉพาะ ซึ่งถูกสร้างโดยเอ็ชทีทีพี (HTTP) เซิร์ฟเวอร์ที่เก็บเอกสาร
- เนม (NAME) กำหนดชื่อของอินฟอร์เมชั่นในเอกสารไม่เหมือนกับ ไตเติ้ล เนื่องจาก “meta name” เป็นการแบ่งประเภทของอินฟอร์เมชั่น
- CONTENT “meta name” สำหรับคอนเท็นต์ (content) ที่เกี่ยวข้องกับชื่อที่กำหนดโดย เนม (NAME) แอททริบิวหรือการตอบสนองที่ถูกกำหนดโดย HTTP-EQUIV

3.2.3.8 บอดี้ (BODY)

บอดี้อีลิเมนท์ประกอบด้วย อินฟอร์เมชั่นเกี่ยวกับเอกสาร และมาร์คอัพ (mark-up) อีลีเมนท์ต่างๆ ที่ใช้ในการกำหนดรูปแบบของตัวอักษร, รูปภาพ และอื่นๆ

3.2.3.9 เซคชั่น เฮดดิ้ง (เอ็ช1, เอ็ช2, …, เอ็ช6) (Section Headings (H1, H2, …, H6))

เอชทีเอ็มแอล ยอมให้มีเฮดดิง ถึง 6 ระดับ โดยใช้แท็ก เอ็ช1, เอ็ช2, ..., เอ็ช6 (Tag H1, H2, ..., H6) ในการใช้ไม่จำเป็นต้องเรียงตามลำดับ แต่เพื่อความเข้าใจที่ตรงกัน จึงควรที่จะใช้ <H1> เพื่อเป็นเฮดดิงที่สำคัญที่สุดและใช้แท็ก (Tag) อื่น กับเฮดดิ้งที่สำคัญรองลงมา

3.2.3.10. มาร์คกิ้ง พารากราฟ กับ เอ็ชทีเอ็มแอล (Marking Paragraphs with HTML)

อีลีเมนท์ <P> ใช้สำหรับแบ่งพารากราฟ และใช้เมื่อต้องการแบ่ง Text 2 Blocks ออกเป็น 2 พารากราฟ ซึ่งในการใช้ต้องทำเครื่องหมายทุกอีลีเมนท์และ ข้อความทุกอย่างที่ต้องการให้อยู่ในพารากราฟเดียวกัน

3.2.3.11. ไลน์ เบรก (Line Breaks)

อีลีเมนท์
 ใช้สำหรับกำหนดให้ข้อความที่ตามหลัง Tag ขึ้นบรรทัดใหม่

3.2.3.12 IMG (In-line Images) Element

อีลีเมนท์นี้ใช้สำหรับให้กราฟฟิค บราวเซอร์ (Graphical browser) แสดงรูปภาพในตำแหน่งที่แท็ก (Tag) ปรากฏ มีชนิดเป็นอีลีเมนท์ว่างจึงไม่จำเป็นปิดท้ายด้วย </IMG> ประกอบด้วยแอททริบิวต่างๆ ดังนี้

- SRC="image_url" ใช้กำหนดยูอาร์แอลของอิมเมจ (image)
- ALIGN=BOTTOM (MIDDLE or TOP) ใช้บอกตำแหน่งของรูปภาพในการแสดง
- ALT="alternative text" ใช้กำหนดข้อความที่แสดงผลสำหรับ บราวเซอร์ ที่ไม่สามารถแสดงผลแบบกราฟฟิคได้
- ISMAP ใช้สำหรับกำหนดให้รูปภาพเป็น active image map นั่นคือ สามารถให้ผู้ใช้คลิกเมาส์ไปในตำแหน่งที่แตกต่างกันสามารถให้ผลตอบสนองที่แตกต่างกัน

3.2.3.13. ไฮเปอร์เท็กซ์ แองเคอร์ (Hypertext Anchors)

แองเคอร์ (Anchor) คือ การกำหนดให้ข้อความหรือออปเจคชนิดต่างๆ ที่อยู่ระหว่าง <A> และ </A> ใช้สำหรับไฮเปอร์เท็กซ์ ลิงค์ แองเคอร์ แอททริบิวท์ (Hypertext link.Anchor Attributes) เหมือนกับ ลิงค์ แอท ทริบิวท์ (LINK Attributes)

- HREF (link to object) *
- NAME (link from object) *
- REL (relationship between objects)
- REV (relationship between objects)
- URN (URN for the document)
- TITLE (TITLE of document)
- METHOD (how to link)

3.2.3.14. ลิสต์ (Lists)

เอชทีเอ็มแอล สนับสนุนการสร้างลิสต์โดยมีอีลีเมนท์ต่างๆ ให้เลือกใช้ ซึ่งแบ่งออกเป็น 2 ประเภท คือกลอซเซรี่ ลิสต์ (Glossary Lists) : <DL>, เรกิวลาร์ ลิสต์ (Regular lists) : <UL>, <OL>, <MENU> and <DIR>.

3.2.3.15. เส้นแบ่งแนวนอน (Horizontal Ruled Line)

HR อีลีเมนท์ใช้สำหรับวาดเส้นในแนวนอน ใช้ในการแบ่งบล็อคข้อมูล จัดเป็นประเภท อีลีเมนท์ว่างไม่จำเป็นต้องมี </HR> ปิดท้าย

3.2.3.16. อักขระพิเศษในเอชทีเอ็มแอล (Special Characters in HTML)

สัญลักษณ์พิเศษบางตัวถูกใช้โดย เอชทีเอ็มแอล ไปตามวัตถุประสงค์ที่ต่างกัน ดังนั้นเมื่อผู้ใช้ต้องการให้แสดงสัญลักษณ์พิเศษบนบราวเซอร์จึงมี เอนติตี้ของตัวอักษรพิเศษเพื่อใช้ในการแสดงสัญลักษณ์พิเศษเหล่านั้น ซึ่งเอนติตี้จะประกอบด้วย 3 ส่วน คือ

- ขึ้นต้นด้วย Ampersand "&"
- ตัวเลขหรือข้อความของตัวอักษรที่ต้องการแสดง
- ลงท้ายด้วยเซมิโคลอน (Semicolon) ";"

3.2.3.17. รูปแบบของตัวอักขระ (Character Emphasis Modes)

เอชทีเอ็มแอล ยอมให้มีการกำหนดรูปแบบของตัวอักษรได้ เช่น ตัวหนา, ตัวเอียง เป็นต้น ซึ่งแบ่งเป็น 2 วิธี คือ

- **ลอจิคอล สไตล์ (Logical Styles)**

EM	ตัวเอียง
STRONG	ตัวหนา
CODE	ขนาดตายตัว
SAMP	ลำดับของตัวอักษร
KBD	Text1 ที่ถูก mark จะเหมือนกับอินพุต จาก คีย์บอร์ด
VAR	ชื่อตัวแปร
DFN	กำหนดค่า
CITE	คำที่ใช้อ้าง มักจะเป็นตัวเอียง

- **ฟิสิกคอลสไตล์ (Physical Styles)**

  TT ใช้รูปแบบตัวอักษรเหมือนพิมพ์ดีด ขนาดคงที่

  B ตัวหนา

  I ตัวเอียง

  U ขีดเส้นใต้

3.2.3.18. รูปแบบอักษร (Special Text Formatting Modes)

เอชทีเอ็มแอล มีรูปแบบการแสดงผลข้อความในระดับลอจิคอล (logical) อยู่ 3 แบบ แบ่งตามลักษณะการแสดงผลของบราวเซอร์ คือ

3.2.3.19. PRE (Preformatted Text)

แสดงข้อความที่อยู่ระหว่างแท็ก (Tag) <PRE> และ </PRE> ให้มีรูปแบบแน่นอนตามที่กำหนด มีออปชั่นแอททริบิว คือ WIDTH (default=80) ใช้กำหนดจำนวนตัวอักษรมากที่สุด ที่สามารถแสดงได้ใน 1 บรรทัด

3.2.3.20. BLOCKQUOTE

ใช้สำหรับกำหนดให้บราวเซอร์จัดการเรียงข้อมูลที่อยู่ระหว่างแท็ก (Tag) <BLOCKQUOTE> และ </BLOCKQUOTE> ตามหน้าจอแสดงผล

3.2.3.21. แอดเดรส (ADDRESS)

ใช้สำหรับข้อมูลเกี่ยวกับที่อยู่, ซิกเนเชอร์ (Signatures), ออเทอร์ชิพ (authorship) เป็นต้น

3.2.3.22. เทเบิ้ล (TABLES)

การสร้างตารางสามารถทำได้โดยใช้เทเบิ้ล อีลีเมนท์ (TABLE element) ซึ่งมีแอททริบิวต่างๆ ดังนี้

- ALIGN ใช้ในการกำหนดตำแหน่งของตาราง ซึ่งมีค่าที่เป็นไปได้ ดังนี้ BLEEDLEFT. LEFT, CENTER, RIGHT, BLEEDRIGHT, JUSTIFY
- BORDER ใช้กำหนดขนาดของเส้นขอบของตาราง
- WIDTH ใช้กำหนดความกว้างของตาราง สามารถกำหนดเป็น % ของความกว้างของส่วนที่ใช้ในการแสดงผลได้
- COLSPEC ใช้กำหนดตำแหน่งของไอเทมในคอลัมน์
- CAPTION ใช้สำหรับตั้งชื่อตาราง มีแอททริบิว คืออะไลน์ (Align) ใช้สำหรับกำหนดตำแหน่งของชื่อตาราง ซึ่งมีค่าต่างๆ
- ทีเอ็ช หรือเทเบิ้ล เฮดเดอร์ (TH (Table Header)) และ ทีดี หรือเทเบิ้ล ดาต้า (TD (Table Data)) ใช้กำหนดชนิดของข้อมูลที่อยู่ในตารางโดย TH จะเป็นตัว

หนา ส่วน TD จะเป็นตัวอักษรธรรมดา ทั้ง 2 อีลีเมนท์นี้มีแอททริบิวที่เหมือนกัน

3.2.3.23. ฟอร์ม อีลีเมนท์ สำหรับฟิลล์เอาท์ ฟอร์ม (FORM Element for fill-out Forms)

ฟอร์ม (FORMS) เป็นรูปแบบที่ทำให้ผู้ใช้สามารถติดต่อกับเอ็ชทีทีพี เซิร์ฟเวอร์ (HTTP Server) ได้มากกว่าการใช้ไอเอสอินเด็กซ์ (ISINDEX) โดยฟอร์ม อีลีเมนท์ (FORM element) จะสร้างฟิลล์เอาท์ฟอร์ม (fill-out Form) ที่ผู้ใช้สามารถกรอกข้อมูลลงไปในช่องที่กำหนดมา เพื่อส่งต่อไปให้ ซีจีไอบินสคริปต์บนเอ็ชทีทีพี เซิร์ฟเวอร์ (HTTP Server)

อีลีเมนท์ที่ใช้ในฟอร์มประกอบด้วย

1).ฟอร์ม (FORM)

ใช้สำหรับกำหนดวิธีการรับข้อมูล และวิธีการจัดการกับอินพุตที่ได้มาเพื่อให้ได้ผลลัพธ์ ที่ต้องการมีแอททริบิวดังนี้

- แอ็คชั่น (ACTION) ใช้กำหนดยูอาร์แอลของโปรแกรม หรือสคริปต์ที่จะใช้ในการกระทำงาน ถ้าไม่กำหนดส่วนนี้เบส ยูอาร์แอล (Base URL)จะถูกเรียกใช้
- เมธอท (METHOD) เป็นการกำหนดเอ็ชทีทีพี เมธอท (HTTP Method) ที่ใช้ในการส่งข้อมูลจากฟอร์มไปยังเซิร์ฟเวอร์ซึ่งมีได้ 2 แบบ คือ
    1. GET (default) ข้อมูลจากฟอร์มจะถูกต่อท้ายยูอาร์แอล (หลังเครื่องหมายคำถาม "?") เหมือนกัน ISINDEX queries
    2. โพสต์ (POST) ข้อมูลจากฟอร์มจะถูกส่งไปยังเซิร์ฟเวอร์ในส่วน บอดี้ของข่าวสาร การที่จะใช้วิธีนี้ต้องแน่ใจว่าเอ็ชทีทีพี เซิร์ฟเวอร์ (HTTP Server) ที่ใช้สนับสนุนวิธีนี้ด้วย

2).อินพุท (INPUT)

ใช้สำหรับรวบรวมข้อมูลจากผู้ใช้ มี แอททริบิวต่างๆ ดังนี้

- อะไลน์ (ALIGN) ใช้กับอิมเมจไทพ์ (IMAGE TYPE) ในการกำหนดตำแหน่งของ image ที่สัมพันธ์กับเท็กซ์ (text (TOP, MIDDLE หรือ BOTTOM))
- เช็ค (CHECKED) ใช้ร่วมกับเช็คบ็อกซ์ (CHECKBOX) และเรดิโอ บัททอน (RADIO button) ในการเลือกในตอนเริ่มต้น ถ้าหากไม่กำหนดจะไม่มีตัวใดถูกเลือก
- MAXLENGTH กำหนดจำนวนตัวเลขและตัวอักษรมากที่สุดที่ผู้ใช้สามารถเติมได้โดยปกติจะไม่ถูกจำกัด
- เนม (NAME) ชื่อของตัวแปรที่ใช้ในการส่งข้อมูล
- ไซส์ (SIZE) กำหนดความกว้างของช่องที่ใช้ป้อนข้อมูล
- เอสอาร์ซี (SRC) ใช้กำหนดยูอาร์แอลของอิมเมจ ไฟล์ (Image File)เมื่อไทพ์ถูกกำหนดเป็นแบบอิมเมจไทพ์ (image TYPE) กำหนดรูปแบบของฟิลด์ที่ให้ผู้ใช้ป้อนข้อมูล

- เช็คบ็อกซ์ (CHECKBOX) สามารถให้ค่าได้หลายค่า
- ฮิดเด้น (HIDDEN) สำหรับค่าที่ถูกกำหนด โดยฟอร์มโดยไม่ต้องรับอินพุตจากผู้ใช้
- อิมเมจ (IMAGE) ใช้รูปภาพเป็นตัวรับอินพุตจากผู้ใช้โดยส่งค่า x, y ที่ถูกคลิกโดย เมาส์
- พาสเวิร์ด (PASSWORD) เป็นฟิลด์สำหรับป้อนข้อมูลแบบเท็กซ์แต่ไม่แสดงผล
- เรดิโอ (RADIO) ใช้สำหรับเก็บข้อมูลที่สามารถเลือกได้เพียงค่าเดียวในแต่ละครั้ง
- รีเซ็ต (RESET) ใช้เคลียร์ค่าให้กลับเป็นค่ามาตรฐาน
- ซับมิท (SUBMIT) เป็นปุ่มที่ใช้สำหรับการส่งข้อมูลในฟอร์มใช้ค่า (VALUE) ในการกำหนดข้อความที่แสดงบนปุ่ม
- เท็กซ์ (TEXT) ใช้สำหรับซิงเกิ้ล ไลน์ของเท็กซ์ (single-line of text)
- เท็กซ์ (TEXT) เป็นช่องให้เติมข้อมูลแบบเท็กซ์บรรทัดเดียว
- VALUE ใช้กำหนดค่าเริ่มต้นของฟิลด์หรือ กำหนดค่าของฟิลด์เมื่อถูกเลือก
- SELECT อีลีเมนท์จะแสดงทางเลือกสำหรับให้ผู้ใช้เลือกค่าใดค่าหนึ่งจากหลายๆ ค่า โดยใช้ออปชั่นเป็นตัวกำหนดค่าต่างๆ เหล่านั้น มีแอททริบิวต่างๆ ดังนี้
- NAME ชื่อที่ใช้ในการส่งข้อมูลที่มาผู้ใช้เลือก
- MULTIPLE โดยปกติผู้ใช้สามารถเลือกได้เพียงอย่างเดียว แต่ MULTIPLE แอททริบิวนี้ จะทำให้ผู้ใช้สามารถเลือกได้มากกว่าหนึ่งอย่าง
- ไซส์ (SIZE) กำหนดจำนวนของไอเทมที่ผู้ใช้สามารถมองเห็นได้ ถ้าไอเทมมีมากกว่าที่กำหนดการแสดงผลจะอยู่ในรูปของลิสต์

3). ออปชั่น (OPTION)

อีลีเมนท์นี้จะปรากฏภายในซีเล็ค อีลีเมนท์ (SELECT element) และใช้ในการแสดงตัวเลือกต่างๆ ภายในซีเล็ค อีลีเมนท์ (SELECT element) มีแอททริบิวต่างๆ ดังนี้

- SELECTED กำหนดให้ออปชั่นถูกเลือกในตอนเริ่มต้น
- VALUE ใช้กำหนดค่าที่จะถูกส่งกลับไปยังเซิร์ฟเวอร์ของแต่ละออปชั่น ถ้าไม่มี แอททริบิวนี้ ค่าที่จะถูกส่งกลับไปคือค่าที่ถูกกำหนดโดยออปชั่นแอททริบิว

4). เท็กซ์แอเรีย (TEXTAREA)

อีลีเมนท์ นี้จะใช้ในการรวบรวมข้อมูลชนิด เท็กซ์ที่มีหลายบรรทัดจากผู้ใช้มีสครอลบาร์ (Scroll Bar) ทำให้สามารถเลื่อน ไปยังตำแหน่งต่างๆ ได้เท็กซ์แอเรีย (TEXTAREA) มีแอททริบิวต่างๆ ดังนี้

- NAME ชื่อที่จะใช้ในการส่งค่ากลับ
- ROWS จำนวนของแถวที่แสดงผล
- COLS จำนวนของคอลัมน์ที่แสดงผล

3.3. ความรู้เบื้องต้นเกี่ยวกับซีจีไอ (CGI)

3.3.1 ความหมายของซีจีไอ (CGI)

ซีจีไอ (CGI) คือ ส่วนหนึ่งของ เว็บเซิร์ฟเวอร์ ที่ใช้ในการติดต่อสื่อสารกับโปรแกรมอื่นๆ ที่ทำงานอยู่บน เซิร์ฟเวอร์นั้น. ซีจีไอ (CGI) เป็นเพียงโปรแกรมที่เขียนขึ้นแล้วฝังอยู่ที่ฝั่งเซิร์ฟเวอร์

3.3.2 ส่วนประกอบของซีจีไอ (CGI)

❖ เอ็นไวโรเม็นท์ วาริเอเบิ้ล(Environment variables) ที่ใช้ในการเขียนโปรแกรมเกทเวย์(Gateway Programming)

AUTH_TYPE	:เป็นตัวระบุโปรโตคอล(Protocol) การเข้ามากับแวลิเดท ยูสเซอร์(validate user) มันจะถูกเซ็ตเมื่อเซิร์ฟเวอร์ สนับสนุนการเข้ามาของยูสเซอร์ (user)
CONTENT_LENGTY	:ความยาวของข้อมูลซึ่งถูกให้โดย ไคล์เอนท์
CONTENT_TYPE	:ระบุชนิดของข้อมูลสำหรับคิวรี่ (query) ซึ่งถูก แอทแทช (attach) ข้อมูล (เช่น เอ็ชทีทีพี (HTTP) โพสต์ (POST) และพุท (PUT))
GATEWAY_INTERFACE	:สเป็ค รีวิชั่น ซีจีไอ (Spec revision CGI) ของ เซิฟร์เวอร์ ซึ่งอยู่ในรูป ซีจีไอ/รีวิชั่น (CGI/revision)
PATH_INFO	:พาธ อินฟอร์เมชั่น (Path information) ที่ถูกให้โดยยูสเซอร์ รีเควส (user request)
PATH_TRANSLATED	:ทรานสเลท เวอร์ชั่น (Translate version) ของพาธ อินโฟ (PATH_INFO) โดย พาร์ธ ประกอบด้วย virtual-to-physical Mapping to it
QUERY_STRING	:เป็นสตริงของข้อมูลที่มาจากการ submit ในแบบฟอร์ม
REMOTE_ADDR	:เป็นไอพี แอดเดรส (IP address) ของเครื่อง ไคล์เอนท์
REMOTE_HOST	:เป็นไอพี แอดเดรส (IP address) ของโฮสต์ (host)
REMOTE_USER	:เป็นออเทนติเคท (authenticate) ของยูสเซอร์ ไอดี (user ID) ผู้ร้องขอสคริปต์ (script) ตัวแปรนี้จะถูกใช้เมื่อ เซิฟร์เวอร์ สนับสนุนกับออเทนติเคชั่น (authentication) และออเทน ติเคชั่น (authentication) ถูกใช้สำหรับ access script

3.3.3 หลักการทำงานของซีจีไอ (CGI) กับแบบสอบถามของ เว็บเพจ

ก่อนที่เราจะเริ่มเขียนซีจีไอ (CGI) มี 3 หลักการทำงานพื้นฐานที่เราควรทำความเข้าใจให้ถ่องแท้เสียก่อนคือ

1. **ไคล์เอนท์ สามารถส่งข้อมูลออกมาได้อย่างไร**

มี 2 วิธีที่ ไคล์เอนท์ สามารถส่งข้อมูลออกมาผ่าน เซิฟร์เวอร์ ไปที่เกทเวย์ (Gateway) คือ โดยผ่าน ยูอาร์แอล หรือในรูปแบบของ METHOD=POST. การส่งผ่าน ยูอาร์แอล คือส่งผ่านรูปแบบไอเอสอินเด็กซ์ (ISINDEX).

สำหรบการส่งผ่านแบบ METHOD=POST จะส่งโดยผ่านสแตนดาร์ด อินพุท (standard input(stdin)) ตัวแปรเอ็นไวโรเม็นท์ ของซีจีไอ (CGI) คือ

CONTENT_LENGTH ซึ่งจะเป็นตัวกำหนดจำนวนของอักขระที่กำลังส่ง

CONTENT_TYPE ซึ่งจะเป็นตัวกำหนด application/x-www-form-urlencoded

สำหรับการส่งผ่านด้วย ยูอาร์แอล จะอยู่ในรูป "?[Field]=[value]+[field]=[value]..." เช่น

http://www.some.box/cgi-bin/name.pl?FirstName=Bill&SecondName=Elmer

ซึ่งจะมีค่าเหมือนการที่ บราวซ์เซอร์ ส่งข้อมูลไปที่ เซิร์ฟเวอร์ โดยผ่าน METHOD=GET. ยูอาร์แอล ที่อยู่ในรูปแบบที่ส่งข้อมูลมาตามหลังเครื่องหมายคำถามดังเช่น ?[data] นี้จะไม่สามารถแสดงอักขระ "=" ได้ แต่จะแสดงในรูปที่ถูกเข้ารหัสแล้วคือ "%3D" ปรากฏอยู่ใน ยูอาร์แอล.ของ เซิฟร์เวอร์ ก็จะรักษาสิ่งนี้เหมือน ไอเอสอินเด็กซ์ คิวรี่ (ISINDEX query) ตัวอย่างเช่น

http://www.hydra.com/cgi-bin/sams/nothing.pl?20

http://www.hydra.com/cgi-bin/sams/nothing.pl?chapter%3D20

ไอเอสอินเด็กซ์ คิวรี่ (ISINDEX query) เป็นคิวรี่ (query) ชนิดเดียวที่ส่งผ่านข้อมูลโดยผ่านคอมมานด์ไลน์ (command line). ข้อมูลไอเอสอินเด็กซ์ (ISINDEX) ไม่ได้ถูกเข้ารหัสโดย เซิร์ฟเวอร์ ก่อนที่จะถูกส่งผ่านไปที่โปรแกรมเกทเวย์. ยูอาร์แอล ที่มีเส้นทางที่แสดงที่เก็บข้อมูลแบบพิเศษ (extra path data) ด้วยวิธีนี้จะอยู่ตามหลังชื่อของโปรแกรมเกทเวย์ทันที ข้อมูลจะถูกต่อท้ายในรูปแบบของเส้นทางที่แสดงที่เก็บข้อมูล(data path)ดังนี้

http://www.hydra.com/cgi-bin/sams/nothing.pl/Bill/Elmer/

แต่ละครั้งที่ เซิร์ฟเวอร์ ติดต่อกับโปรแกรมเกทเวย์ (Gateway) มันจะใส่ทุกๆ สิ่งที่ตามมาในรูป PATH_INFO ดังนั้นเราจะได้ PATH_INFO=Bill/Elmer/

ก่อนที่ เซิร์ฟเวอร์ จะส่งผ่านข้อมูลก็จะมีการเข้ารหัส

สำหรับเอ็กเซ็ปชั่น (Exception) ของไอเอสอินเด็กซ์ (ISINDEX) ข้อมูลจะถูกเข้ารหัสก่อนโดย เซิร์ฟเวอร์. โดยที่สเปซ (space) จะถูกเปลี่ยนเป็นเครื่องหมายบวก (+) อักขระก็จะถูกเปลี่ยนเป็น %[hex equivalent] และ field ที่อยู่ในแบบฟอร์มก็จะเชื่อมต่อกันด้วยเครื่องหมาย & ตัวอย่างเช่น ถ้าแบบฟอร์มเป็นดังนี้

Field1<INPUT NAME=FIELD1>Filed2<INPUT NAME=FIELD2>

ถ้าข้อมูลที่อยู่ใน Field1 และ 2 คือ 1 !@#$%และ 2^&*_+| ดังนั้นข้อมูลที่ถูกเอ็นโค้ด (encode) แล้วจะได้ดังนี้

FIELD1=1+%21@%23%24%25&FIELD2=2+5E%26*%28%29_%2B%7C

2. เซิร์ฟเวอร์ ส่งข้อมูลไปให้โปรแกรมเกทเวย์ (Gateway) อย่างไร

ขณะนี้ เซิร์ฟเวอร์ ได้รับข้อมูลแล้ว มีอยู่ 3 วิธีที่จะส่งข้อมูลนี้ไปให้โปรแกรมเกทเวย์ (Gateway)

- ผ่าน stdin ของโปรแกรมเกทเวย์ (gateway). ถ้า REQUEST_METHOD=POST ลำดับแรก เซิร์ฟเวอร์ จะเอ็นโค้ด (encode) ข้อมูลดังที่กล่าวมาแล้วในข้างต้น ต่อจากนั้นก็จะส่งข้อมูลที่เอ็นโค้ด (encode) แล้วไปที่เกทเวย์ (gateway) โดย stdin
- แต่ถ้า REQUEST_METHOD=GET เซิร์ฟเวอร์ จะรับข้อมูลมาแบบไอเอสอินเด็กซ์ คิวรี่ (ISINDEX query). เซิร์ฟเวอร์ จะส่งข้อมูลไปบนโปรแกรมเกทเวย์ (gateway) เหมือนกับคอมมานด์ ไลน์ อาร์กิวเมนท์ (command line argument) โดยปราศจากการเอ็นโค้ด (encode) ข้อมูล
- ผ่านเอ็นไวโรเมนท์ วาไรเอเบิ้ล (environment variable) ของ เซิร์ฟเวอร์. วาไรเอเบิ้ล (Variable) ใดๆ จะเซตโดย ไคล์เอนท์ ก็จะถูกส่งผ่านไปโดย เซิร์ฟเวอร์ ไปที่โปรแกรมเกทเวย์ (Gateway)

3. **เอาท์พุทของเกทเวย์ (Gateway Output)**

เอาท์พุทของเกทเวย์ จะเริ่มต้นด้วยเฮดเดอร์ (Header) ที่ เซิร์ฟเวอร์ เข้าใจซึ่ง ณ เวลานี้มีเฮดเดอร์ (header) อยู่ 3 เฮดเดอร์ (header) ที่ เซิร์ฟเวอร์ เข้าใจ

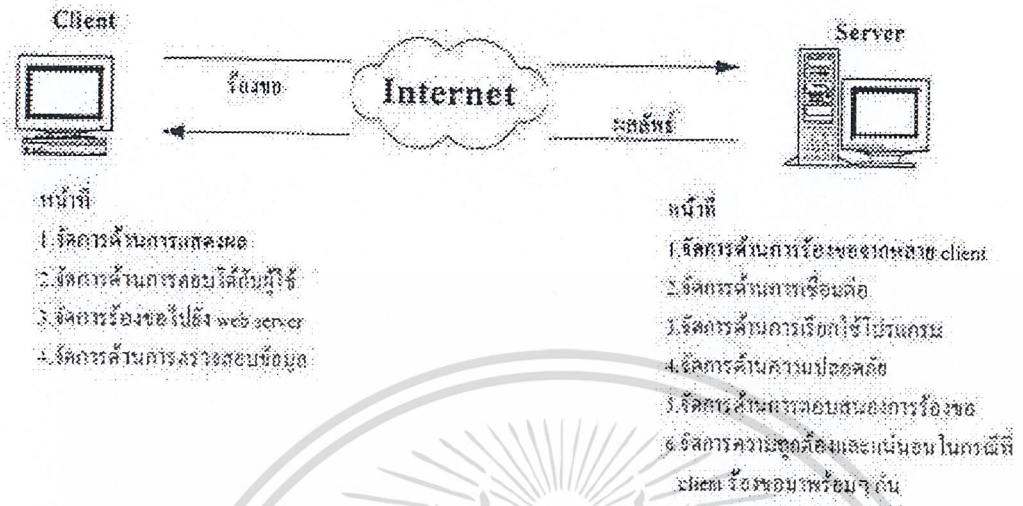
- Content-type:[type]/[subtype] ซึ่งส่วนใหญ่ที่ใช้กันคือ Content-type: text/html
- Location:[URL]
- Status:[message string] เช่น Status: 305 Document moved

3.4. ความรู้เบื้องต้นเกี่ยวกับระบบ ไคล์เอนท์เซิร์ฟเวอร์

การทำงานของระบบแบบ ไคล์เอนท์เซิร์ฟเวอร์ จะมีการทำงาน โดยมีการแบ่งโพรเสส (Process) วิ่งบนส่วนต่างๆ. โดยจะแบ่งโปรแกรมออกเป็น 2 ส่วน ส่วนหนึ่งจะทำงานอยู่ที่ฝั่ง ไคล์เอนท์ และอีกส่วนจะทำงานอยู่ที่ฝั่ง เซิร์ฟเวอร์

3.4.1 ระบบจัดการฐานข้อมูลแบบไคลเอ็นท์/เซิร์ฟเวอร์

รูปแสดงการทำงานของระบบไคลเอ็นท์/ เซิร์ฟเวอร์ แสดงดังในรูปที่ 3.2



รูปที่ 3.2 แสดงการทำงานของ แบบจำลองไคลเอ็นท์/เซิร์ฟเวอร์

ระบบจัดการฐานข้อมูลนับว่าเป็นระบบโปรแกรมจัดการประเภทแรกๆ ของระบบคอมพิวเตอร์ที่มีประสิทธิภาพการทำงานอย่างมหาศาล. ต่อมาเมื่อระบบเครือข่ายคอมพิวเตอร์มีการพัฒนาและเป็นที่ยอมรับจึงได้มีการนำระบบจัดการฐานข้อมูลมาใช้ในรูปแบบของ ไคล์เอนท์เซิฟร์เวอร์

- **ส่วน เซิร์ฟเวอร์**

เป็นส่วนทำงานหลักของระบบจัดการฐานข้อมูล คือ มีหน้าที่ในการเก็บข้อมูลและดึงข้อมูลจากอุปกรณ์จัดเก็บข้อมูล ซึ่งส่วนเซิร์ฟเวอร์ จะสามารถทำหน้าที่ดังกล่าวได้สองวิธี คือ

1. ทำการติดต่อกับอุปกรณ์จัดเก็บข้อมูล โดยการทำงานร่วมกันระหว่างส่วนเซิร์ฟเวอร์ ของระบบจัดการฐานข้อมูลและระบบปฏิบัติการ

2. ทำการติดต่อกับอุปกรณ์จัดเก็บข้อมูลโดยการทำงานของส่วน เซิร์ฟเวอร์ ของระบบจัดการฐานข้อมูลเอง

การที่ระบบจัดการฐานข้อมูลทำการติดต่อกับอุปกรณ์จัดเก็บข้อมูลโดยการทำงานของส่วนเซิร์ฟเวอร์เอง จะทำให้สามารถทำการ จัดเก็บและการดึงข้อมูลได้อย่างมีประสิทธิภาพสูงสุด

นอกจากหน้าที่ดังกล่าวแล้ว ส่วนเซิร์ฟเวอร์ ยังมีหน้าที่ในการ จัดการด้านความปลอดภัยของข้อมูล, ความถูกต้องของข้อมูล และการกู้ข้อมูล ตลอดจนจัดการอำนวยความสะดวกให้กับส่วน ไคล์เอนท์ ในการติดต่อกับฐานข้อมูล โดยผ่านคำสั่งที่เรียกว่า *คำสั่งเอสคิวแอล(SQL)*

- **ส่วน ไคล์เอนท์**

ส่วน ไคล์เอนท์ หมายถึงตัวโปรแกรมที่ทำการติดต่อและเข้าถึงข้อมูลในระบบจัดการฐานข้อมูลโปรแกรมโดยใช้*ภาษาเอสคิวแอล(SQL)* เพื่อทำการเข้าถึงข้อมูลในระบบจัดการฐานข้อมูล การพัฒนาโปรแกรมในส่วนนี้บางครั้งจำเป็นอย่างยิ่งที่โปรแกรมดังกล่าวจะต้องสามารถนำไปใช้ได้ในทุกๆ ระบบ ซึ่งสามารถพิจารณาส่วนประกอบเป็นสองส่วนคือ

1. ส่วนติดต่อกับผู้ใช้
2. ส่วนติดต่อกับระบบจัดการฐานข้อมูล

ส่วนติดต่อกับผู้ใช้นั้นสามารถแก้ปัญหาดังกล่าวได้โดยอาศัยคุณสมบัติของ เวิลด์ไวด์เว็บ โดยใช้ภาษา เอชทีเอ็มแอล สำหรับส่วนติดต่อกับระบบจัดการฐานข้อมูลนั้นปัญหาดังกล่าวสามารถแก้ไขได้โดยการเพิ่มตัวกลางระหว่างส่วน ไคล์เอนท์ และส่วน เซิร์ฟเวอร์ โดยตัวกลางดังกล่าวจะทำหน้าที่ในการสื่อสารกับส่วน ไคล์เอนท์ และไดรเวอร์ (driver) ของระบบจัดการฐานข้อมูล ทำให้เกิดการติดต่อสื่อสารระหว่างส่วน ไคล์เอนท์ กับส่วน เซิร์ฟเวอร์ ที่เป็นมาตรฐาน

การใช้ตัวกลางในการติดต่อสื่อสารระหว่างส่วน ไคล์เอนท์ กับส่วน เซิฟร์เวอร์ นั้นทำให้เกิดการล่าช้าในการสื่อสารเนื่องจากเกิดการรับส่งข้อมูลหลายลำดับขั้นนั้น นอกจากนั้นการใช้ตัวกลางในการติดต่อสื่อสารระหว่างส่วน ไคล์เอนท์ กับส่วน เซิฟร์เวอร์ นั้นยังทำให้ไม่สามารถใช้งานส่วน เซิฟร์เวอร์ ของระบบจัดการฐานข้อมูลได้อย่างเต็มที่อีกด้วย

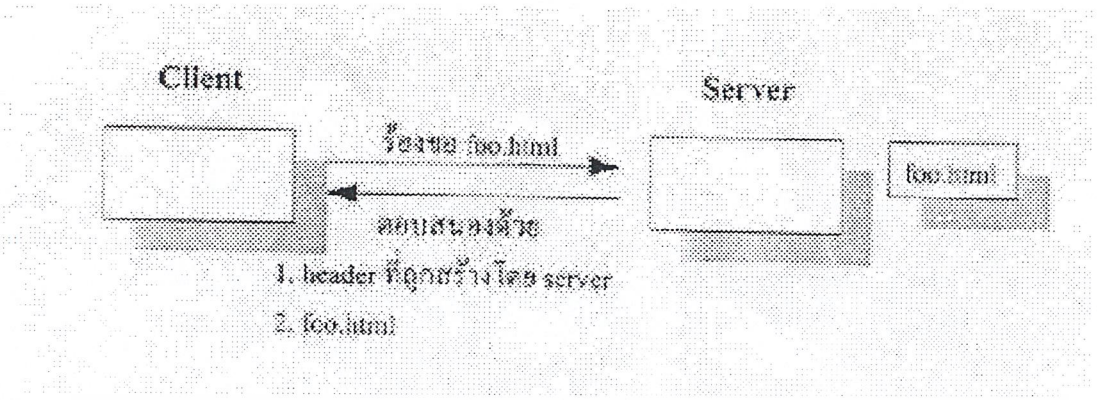
3.4.2 การเข้าถึงข้อมูลโดยผ่านเวิลด์ไวด์เว็บแบบไคลเอ็นท์/เซิร์ฟเวอร์

จากนิยามของ ไคล์เอนท์เซิฟร์เวอร์ เมื่อนำมาประยุกต์เข้ากับ เวอร์ไวด์เว็บ จะเห็นได้ว่าโปรแกรมที่วิ่งอยู่ที่ ไคล์เอนท์ คือ เว็บบราวซ์เซอร์ และโปรแกรมที่วิ่งอยู่ที่ เซิฟร์เวอร์ คือ เว็บเซิฟร์เวอร์ หรือ HTTP Server

เมื่อ ไคล์เอนท์ ร้องขอเอกสาร เอชทีเอ็มแอล มายัง เซิฟร์เวอร์. เซิฟร์เวอร์ จะทำการหาเอกสารนั้นในระบบแฟ้มข้อมูลของตนเอง แล้วทำการสร้างเฮดเดอร์ของเอกสาร เอชทีเอ็มแอล นั้นแล้วส่งผ่านเครือข่ายไปยังไคล์เอนท์ พร้อมกับเอกสาร เอชทีเอ็มแอล

การทำงานของระบบไคลเอ็นท์/ เซิร์ฟเวอร์ ถ้าจำแนกตามการเรียกใช้โปรแกรมจากภายนอก สามารถจำแนกได้เป็น 2 แบบ

1. แบบที่ไม่มีการเรียกใช้โปรแกรมภายนอก ดังแสดงในรูปที่ 3.3
2. แบบที่มีการเรียกใช้โปรแกรมภายนอก ดังแสดงในรูปที่ 3.4



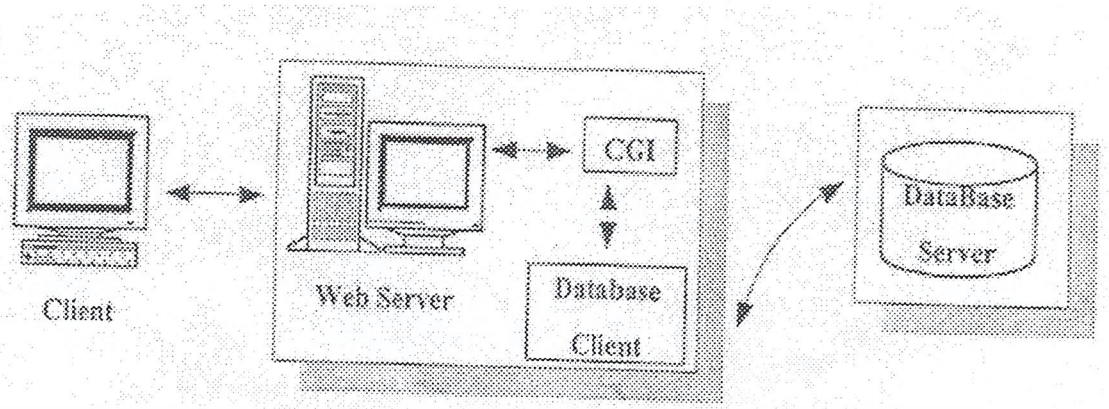
รูปที่ 3.3 แสดงการทำงานของไคลเอ็นท์/เซิร์ฟเวอร์ แบบที่ไม่มีการเรียกใช้โปรแกรมภายนอก

ในกรณีที่ ไคล์เอนท์ ร้องขอโปรแกรมภายนอก หรือโปรแกรมที่ไม่ใช่ส่วนของ เซิฟร์เวอร์ แต่อยู่ในเครื่อง เซิฟร์เวอร์ กรณีดังกล่าวนี้จำเป็นต้องใช้โปรแกรมชนิดหนึ่งที่อยู่ใน เซิฟร์เวอร์ ที่เรียกว่า ***โปรแกรมซีจีไอ*** (Common Gateway Interface :CGI) เป็นตัวเชื่อมต่อกับโปรแกรมภายนอกเหล่านั้นเพื่อให้ เซิฟร์เวอร์ จัดการ แล้วทำการจัดรูปแบบของผลลัพธ์ ตลอดจนสร้าง header เพื่อให้ เซิฟร์เวอร์ ส่งผ่านระบบเครือข่ายไปยังไคล์เอนท์ ต่อไป ดังแสดงในรูปที่ 3.4ข้างล่างนี้

รูปที่ 3.4 แสดงการทำงานของไคลเอ็นท์/เซิร์ฟเวอร์ แบบที่มีการเรียกใช้โปรแกรมภายนอก

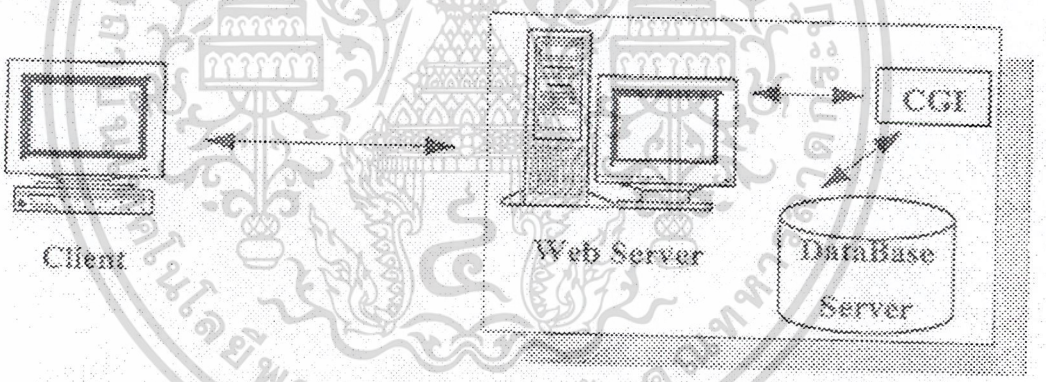
ระบบฐานข้อมูลที่ผ่าน เว็บ แบบ ไคล์เอนท์เซิฟร์เวอร์ นี้สามารถแบ่งออกเป็น 2 ประเภท คือ ฐานข้อมูลที่ผ่าน Network และฐานข้อมูลที่ไม่ผ่าน Network (ซึ่งดูที่การติดต่อระหว่าง CGI กับ Database Server)

- ฐานข้อมูลที่ผ่าน Network - CGI และ Database Server จะต้องติดต่อกันผ่าน Network เหมาะกับการที่ต้องการใช้งานฐานข้อมูลขนาดใหญ่ ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 แสดงระบบฐานข้อมูลที่ผ่านเน็ตเวิร์ค

- ฐานข้อมูลที่ไม่ผ่าน Network - CGI และ Database Server อยู่บนเครื่องเดียวกันเหมาะกับฐานข้อมูลขนาดเล็ก ดังแสดงในรูปที่ 3.6

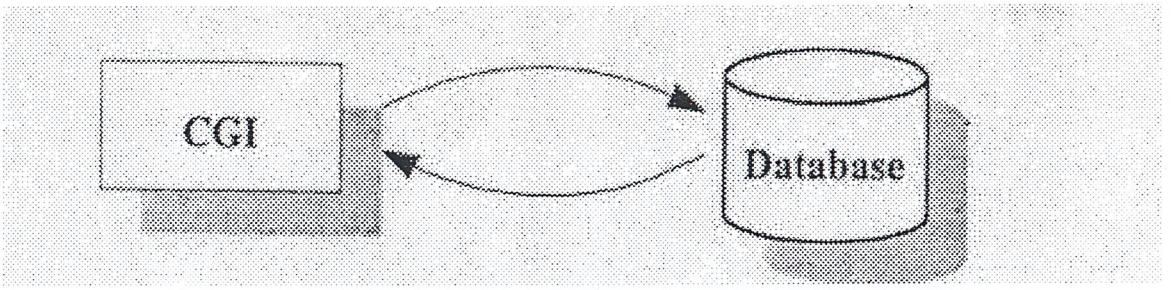


รูปที่ 3.6 แสดงระบบฐานข้อมูลที่ไม่ผ่านเน็ตเวิร์ค

3.5.การติดต่อระหว่างซีจีไอกับฐานข้อมูล(CGI and Database)

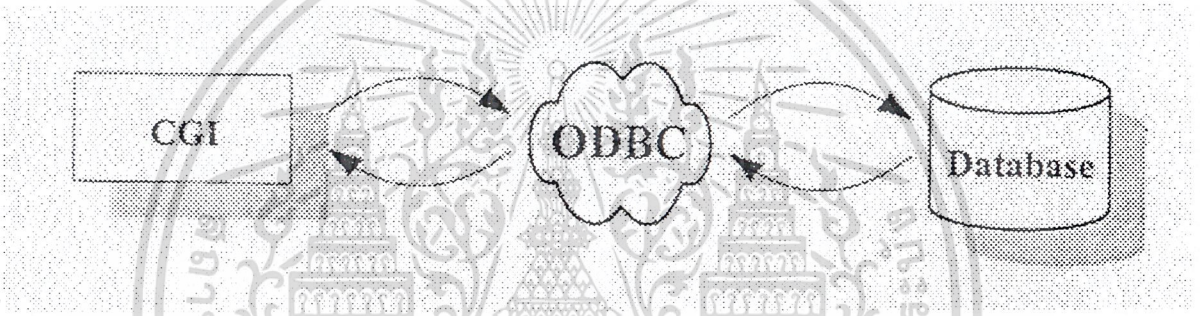
การติดต่อกับฐานข้อมูล ระหว่าง CGI และ Database สามารถทำได้ 2 วิธี คือ

1. เรียกใช้งานโดยตรง เช่นในกรณีของ Visual basic จะมี Microsoft Access database engine ซึ่งสามารถเรียกใช้งานได้เลย ดังแสดงในรูปที่ 3.7

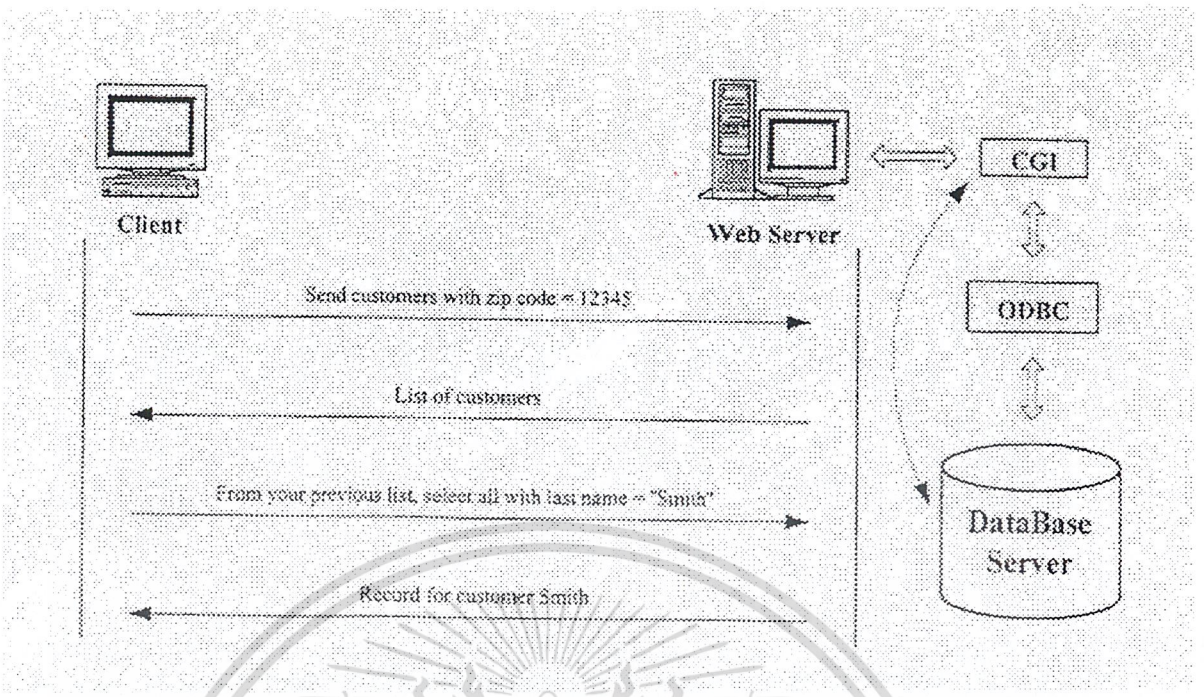


รูป 3.7 แสดงการเรียกใช้งานฐานข้อมูลโดยตรง

2. เรียกใช้งานโดยผ่าน ODBC (Open Database Connectivity) แสดงในรูปที่ 3.8



รูปที่ 3.8 แสดงการเรียกใช้งานฐานข้อมูลโดยผ่านโอดีบีซี



รูปที่ 3.9 แสดงการจำลองการทำงานจริงของการติดต่อฐานข้อมูลโดยผ่าน เว็บ

บทที่ 4

หลักการของ DBI และ DBD

4.1. ความรู้เบื้องต้นเกี่ยวกับภาษาเอสคิวแอล (SQL)

Structured Query Language (SQL) คือ ภาษามาตรฐานที่ใช้ในการจัดการกับข้อมูลในรีเลชั่นนอลดาต้าเบส (relational database) และเป็นภาษาที่มีการใช้มาเป็นเวลาช้านานแล้ว. เอสคิวแอลถูกสร้างมาเพื่อเป็นภาษามาตรฐานที่ใช้ในการจัดการข้อมูลในฐานข้อมูล.ซึ่งไม่ใช้ภาษาดั้งเดิม (traditional)เช่น ภาษาซี (C)หรือภาษาปาสคาล (Pascal)ซึ่งเป็นภาษาที่มีคีย์เวิร์ดจำนวนมาก และวากยสัมพันธ์เพื่อช่วยในการสร้าง (creation) และการจัดการ (manipulation) คอมโพเน็นต์ (component)

เอสคิวแอล เป็นภาษาที่เขียนด้วยเท็กซ์ (text) ที่ดีและเหมาะกับหลักการของฐานข้อมูล.ดังนั้นจึงมีเครื่องมือที่เป็นอินเตอร์เฟสกับผู้ใช้แบบกราฟฟิกที่ใช้ในการเขียนเอสคิวแอลโค้ด

การทำคิวรี (Perform a Query)

เป็นการค้นหาฐานข้อมูล โดยขึ้นกับกรณีต่างๆ รวมทั้งโครงสร้าง ดัชนีและ ชนิดของคอลัมน์ ฐานข้อมูลส่วนใหญ่จะมีการทำออบติไมซ์ (optimization) เพื่อทำการแปลคำสั่งที่มีการประมวลผลอย่างมีประสิทธิภาพมากที่สุด การทำคิวรีมีวากยสัมพันธ์ดังต่อไปนี้

SELECT [all | distance] field list
[INTO table name]
[FROM table name]
[WHERE predicate]
[GROUP BY expression]
[ORDER BY column(s)]
[WHERE predicate]

where ใช้ในการระบุขอบเขตในการเลือก โดยใช้รูปแบบดังนี้

* ชื่อฟิลด์โอเปอเรทกับนิพจน์
* ชื่อฟิลด์โอเปอเรทกับชื่อฟิลด์

โอเปอเรชั่นต่างๆ

= เป็นการเช็คความเท่ากัน
!= เป็นการเช็คความไม่เท่ากัน
> เป็นการเช็คความมากกว่า
< เป็นการเช็คความน้อยกว่า

[not] between เป็นการเช็คว่าฝั่งซ้ายมืออยู่ในช่วงฝั่งขวามือหรือไม่

[not] in เป็นการเช็คว่า

4.2.DBI

DBI เป็นตัวช่วยสำหรับ การออกแบบและติดตั้งตัวเชื่อมต่อกับฐานข้อมูล(database-independent interface).สำหรับการเชื่อมต่อกับฐานข้อมูล.โดยต้องอาศัยผู้ที่มีความรู้อย่างลึกซึ้งทางด้านเทคโนโลยีฐานข้อมูล

Perl นิยมนำมาใช้อย่างมากในการเขียนโปรแกรม CGI(Common Gateway Interface)และก็เลี่ยงไม่ได้ที่จะต้องมีการติดต่อกับฐานข้อมูล.

ทำไมถึงต้องใช้ DBI?

- Database Independence

หลักการทำงานของ DBI จะเกี่ยวข้องกับฐานข้อมูล โดยทำหน้าที่เป็นตัวเชื่อมต่อกับฐานข้อมูล โดยการ

ทำ

- Connection และ Disconnection
- การเปิด และ ปิด cursors
- การดึงข้อมูลออกมา หรือ นำข้อมูลเข้าไปโดยมีลำดับการทำงานดังนี้

1. โหลด DBI driver.
2. ติดต่อกับฐานข้อมูลโดยต้องผ่าน DBD.
3. เปิด cursor เพื่อเก็บคำสั่ง SQL.
4. ดึงผลลัพธ์
5. ปิด cursor
6. ปิดการเชื่อมต่อ
7. Exit

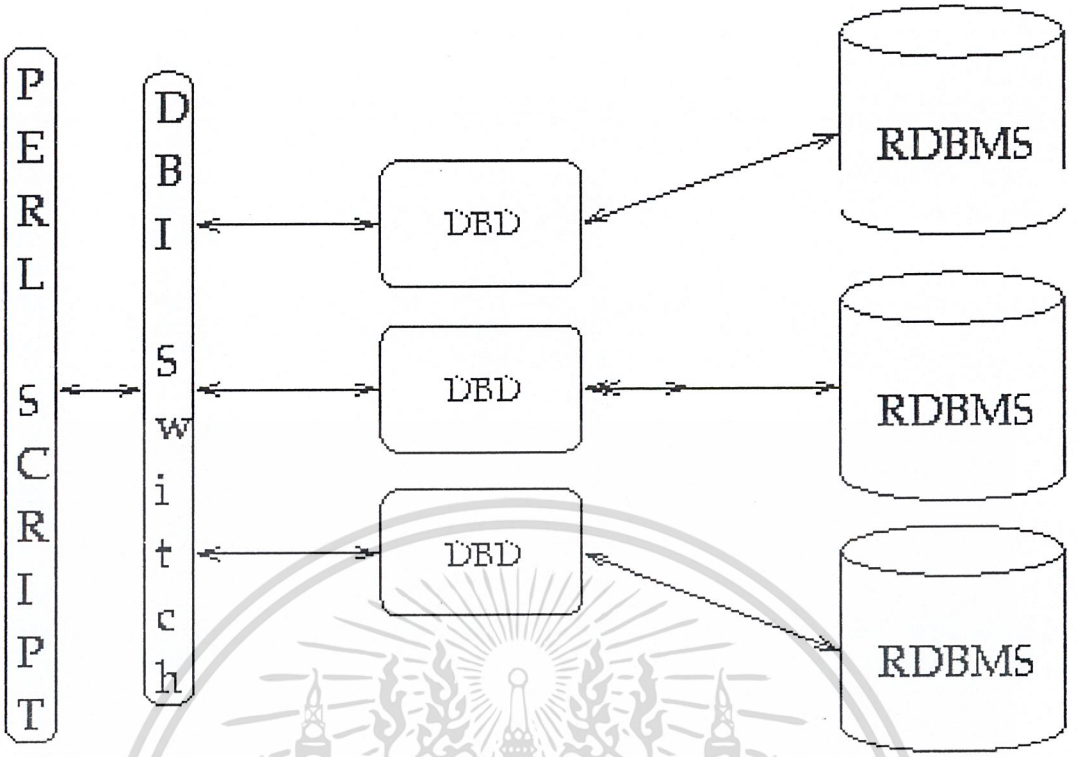
ระหว่างฐานข้อมูลแต่ละชนิดก็จะมีรายละเอียดไม่เหมือนกัน เช่น ชนิดของข้อมูลของ field ต่างๆ ที่ถูกส่งออกมา เป็นต้น แต่ก็มีหลักการอย่างเดียวกัน DBI ได้จัดให้การรวมที่ซึ่งสามารถเขียน code ในแบบทั่วไป ในการ access ในลักษณะที่ไม่เป็นมาตรฐานได้เมื่อต้องการ

หลังจากที่เราดึงข้อมูลจากฐานข้อมูลได้แล้ว DBI จะเก็บข้อมูลไปใส่ไว้ในตัวแปร scalar ซึ่งเป็นชนิดข้อมูลที่ perl สามารถจัดการได้อย่างดี และ DBI สามารถติดต่อได้มากกว่าหนึ่งฐานข้อมูลที่แตกต่างกันภายในโปรแกรม perl โปรแกรมเดียว DBI จะ run โดยตรงใน platform หลายๆ platform มากกว่า ODBC และง่ายกว่า ODBC

4.3.ลักษณะโครงสร้างของ DBI

โครงสร้างของ DBI(แสดงในรูปที่ 4.1) มีลักษณะที่สละสลวยโดยการวางหลักการ หรือ แนวคิดของตัวเชื่อมต่อไว้อย่างดี.

DBI interface เป็นคำที่ถูกนำมาอธิบายทั้ง specification ของตัวเชื่อมต่ออย่างเช่น Method ต่างๆที่เราเขียนในโปรแกรม และ มอดูลของซอฟต์แวร์ ซึ่งเป็นส่วนประกอบของระบบ



รูปที่ 4.1 แสดงโครงสร้างของการติดต่อระหว่าง DBI และ DBD

<u>คำศัพท์ที่ควรรู้เบื้องต้น(TERM)</u>

API (Application Perl – script Interface) :

เป็นตัวที่ DBI เรียกตัวเชื่อมต่อและตัวแปรกับ perl scripts. API ถูกติดตั้งเป็นเหมือนเป็นส่วนขยายของ Perl

Switch :

Code ที่จัดให้ DBI API และส่งเรียกฟังก์ชั่น DBI กับ Driver ที่เหมาะสม Switch ยังเป็นตัวที่ทำหน้าที่ dynamic loading ของ Driver ต่างๆ, การตรวจสอบข้อผิดพลาด/handling และหน้าที่อื่นๆ ทั่วๆ ไป

Driver :

Driver เป็นตัวสนับสนุน Engine (database) ที่ให้มาซึ่งมีอยู่ด้วยกันหลายชนิด ภายใน driver จะเก็บ ฟังก์ชั่นของ DBI ซึ่งใช้ฟังก์ชั่นติดต่อ privatae interface กับ Engine ภายในตัวของ Switch เองจะมี Driver interface และสามารถดูแลเหมือนเป็น driver ของ Application ได้อย่างสะดวก

Engine :

"Engine" จริงๆ ซึ่งถูกติดต่อผ่าน Driver เช่น Oracle, Ingros, Sysbase เป็นต้น สังเกตุว่า Engine ต่างๆ ไม่จำเป็นต้องเป็นฐานข้อมูล (database) อย่างเช่น DNS, X.500, SNMP

DRI (Driver Interface) :

เป็นตัวเชื่อมต่อ (interface) ที่ถูกจัดให้โดย Switch ซึ่งถูกใช้ติดต่อสื่อสารกับ Driver เฉพาะตัวติดตั้ง

Driver เท่านั้นที่ต้องการนำมาพิจารณากับตัวเชื่อมต่อนี้ การเขียน Driver ต่างๆ ของ Dbperl จะทำงาน

เกี่ยวกับ Switch และ DRI

Handle :

Handle คือตัวอ้างอิงที่ให้ค่ากลับเมื่อมีฟังก์ชั่น DBI เรียกใช้งานโดยสามารถส่งค่ากลับไปที่ DBI ฟังก์ชั่น

เพื่ออ้างอิงกับ object เช่น Driver, การติดต่อของฐานข้อมูล หรือคำสั่ง SQL

Library :

แพ็คเกจของ perl ของฟังก์ชั่นอรรถประโยชน์ (utility functions) เพื่อให้การทำงานทั่วๆ ไปกับผู้ใช้ DBI

Bundle :

ชุดของ DBI และ Driver จริงๆ ซึ่งถูกสร้างและ/หรือ ถูก installed บนระบบซึ่งปฏิบัติการ (execute) perl

script

TBD :

ส่วนของ specification ซึ่งยังไม่สมบูรณ์

4.4.กฎการเชื่อมต่อทั่วไป

1) ข้อมูลส่วนใหญ่จะถูกส่งกลับไปที่ perl script เป็นตัวสตริงของ perl ซึ่งยอมให้แสดงภายในของเขตของ Engine/Adaptor (โดยปราศจากความแน่นอน) ที่ถูก handle โดยปราศจากการสูญเสียความถูกต้อง ระวังการแปลงของ perl และ Engine/Adaptor ต่างๆ อาจจะไม่รักษาความถูกต้องเดียวกัน
2) วันที่และเวลาจะถูกส่งกลับเป็นสตริงของอักขระในรูปแบบเดิมของ Engine ที่ติดต่ออยู่ ฟังก์ชั่นต่างๆ ถูกให้กับการแปลงค่ารูปแบบดั้งเดิม (native format) เป็น integer ของจำนวนของวินาทีตั้งแต่ 1 มกราคม ค.ศ. 1970 ฟังก์ชั่นการแปลงอื่นๆ อาจจะถูกกำหนดทีหลัง
3) Perl สนับสนุนข้อมูลชนิดไบนารีในสตริงของ perl และ Switch จะเป็นตัวส่งข้อมูลไบนารีไปให้กับ Adapter และรับจาก Adaptor โดยไม่มีการเปลี่ยนแปลงค่าใดๆ มันขึ้นอยู่กับตัวสร้าง Adaptor ที่จะตัดสินว่าจะ handle อย่างไร

4) คำสั่ง SQL หลายๆ คำสั่งอาจไม่ถูกรวมอยู่ในคำสั่ง handle คำสั่งเดียวเช่น $sh ข้อจำกัดนี้จะเกิดขึ้นกับผู้ใช้ Sybase เป็นส่วนใหญ่
5) การอ่าน record แบบไม่เป็นลำดับจะไม่ถูกสนับสนุนในเวอร์ชั่นของ Dbperl API นี้ Record ต่างๆ สามารถถูกดึงออกมาตามลำดับเท่านั้นซึ่งฐานข้อมูล (database) จะส่งมันกลับและมันจะถูกดึงออกมาแต่ละครั้งแล้วก็จะถูกละไป
6) การ updates และ deletes ยังไม่ถูกสนับสนุนโดย Dbperl เวอร์ชั่นนี้
7) ตัวสร้าง Adaptor แต่ละตัวจะให้กลไกการทำงานที่เป็นประโยชน์อย่างอิสระ

4.5. Database VS DBM

Database (อาจเป็น ODBMS หรือ RDBMS) เป็นตัวที่เก็บข้อมูลและส่งข้อมูลเมื่อมีต้องการดูข้อมูลที่ถูกเก็บไว้ภายใน แต่เดิม UNIX จะถูกให้ "database" กับไฟล์พื้นฐาน (file-based) คือ DBM หรือ Database Manager system ระบบนี้จะเป็นตัวให้ความสามารถที่จะจัดเก็บข้อมูลในไฟล์ต่างๆ และผลิตข้อมูล

- **File Locking**

ระบบ DBM จะไม่ยินยอมให้ความสามารถ file locking ที่ robust แต่ละตัวหรือความสามารถสำหรับการแก้ปัญหาที่อาจเกิดขึ้นระหว่างการเขียนที่เป็นแบบ unsynchronized

- **Arbitrary Data Structure**

สิ่งที่สำคัญส่วนใหญ่ระบบ DBM จะยินยอมเฉพาะโครงสร้างข้อมูลที่ถูกระบุ key และ value เท่านั้น value สามารถเป็น object ที่ซับซ้อนแต่ key จะต้องเป็นหนึ่งเดียว (unique) เท่านั้น

แต่อย่างไรก็ตามระบบ DBM ยังคงให้ฟังก์ชั่นที่เป็นประโยชน์สำหรับผู้ใช้ มอดูลต่างๆ ของ perl ที่เข้าถึงระบบ DBM ตอนนี้ได้ถูกรวมอยู่ใน Perl แล้ว เช่น GDBM_File

The Modules

Perl5 มีความสามารถที่จะเพิ่มมอดูลภายนอกเข้ากับตัว interpreter ของ perl ความสามารถนี้ทำได้โดยการ compile และ link มอดูลกับ interpreter ของ perl หรือโดย Dynaloading (Dynamically loading) module เข้าไปขณะทำการรัน interpreter

จริงๆ แล้ว DBI จะกระทำตัวเป็นเสมือนท่อเชื่อมต่อสำหรับมอดูล DBD DBD ต่างๆ สร้าง method ต่างๆ กำหนดใน DBI ทั้งหมดเช่น connect ( ) แต่สร้างมันในวิธีทาง database – specific Method ต่างๆ ของ database_independent ถูกกำหนดโดย DBI DBD code จะถูกเขียนและดูแลโดย shell และผู้ขาย database แต่ละแบบประกอบด้วย Oracle, Informix, mSQL, Ingres และ Sybase.

มี 2 วิธีที่จะใช้ในการโหลด driver กับ DBI แต่วิธีที่เหมาะสมคือการใช้ method DBI -> connect ( ) ระบุ driver ที่คุณต้องการจะโหลด หลังจากโหลด driver ของ database แล้วพยายามติดต่อกับฐานข้อมูล ซึ่งจะเป็นการเรียกค่ามาเก็บใน handle ของ database ($dbh)

ตัวอย่าง code :

```
#! /usr/bin/perl    -W
use DBI ;
$dbh = DBI -> connect ('connection_string', 'username', 'password', 'databasename');
if (!defined $dbh) {
die "Cannot do \$dbh -> connect : $DBI :: errstrin";
}
```

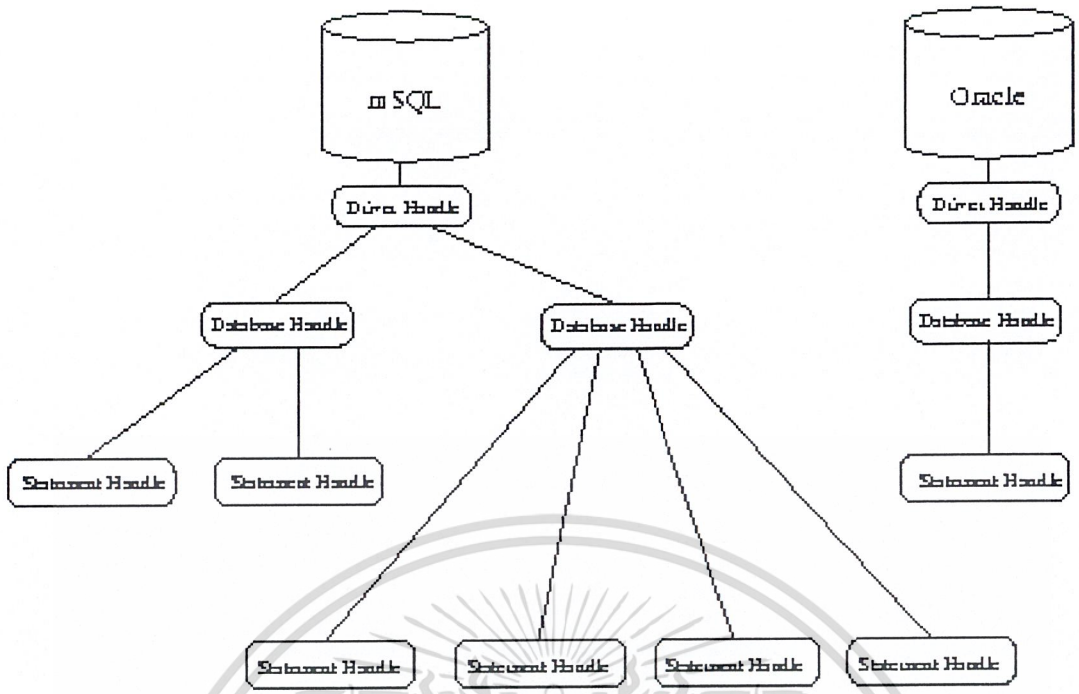
อีกวิธีหนึ่งที่ใช้ได้คือการเรียกใช้ method ภายในโดยการเรียก install_driver ( ) กับ DBI ซึ่งจะทำการ load driver และเก็บไว้ใน driver handle

ตัวอย่าง code :

```
#! /usr/bin/perl    -W
use DBI;
$drh = DBI -> install_driver ('databasename');
if (!defined $drh) {
        die "Cannot load driver : $!\n" ;
}
```

- **Handles**

Handle ต่างๆ เป็น object ของ perl ที่ถูกส่งกลับโดย method ต่างๆ ของ DBI ซึ่งโปรแกรมเมอร์สามารถใช้เข้าถึงข้อมูลที่ระดับ Abstracted layer หลายระดับ Handle ต่างๆ ที่ถูกใช้โดย DBI แสดงได้ดังในรูปที่ 4.2

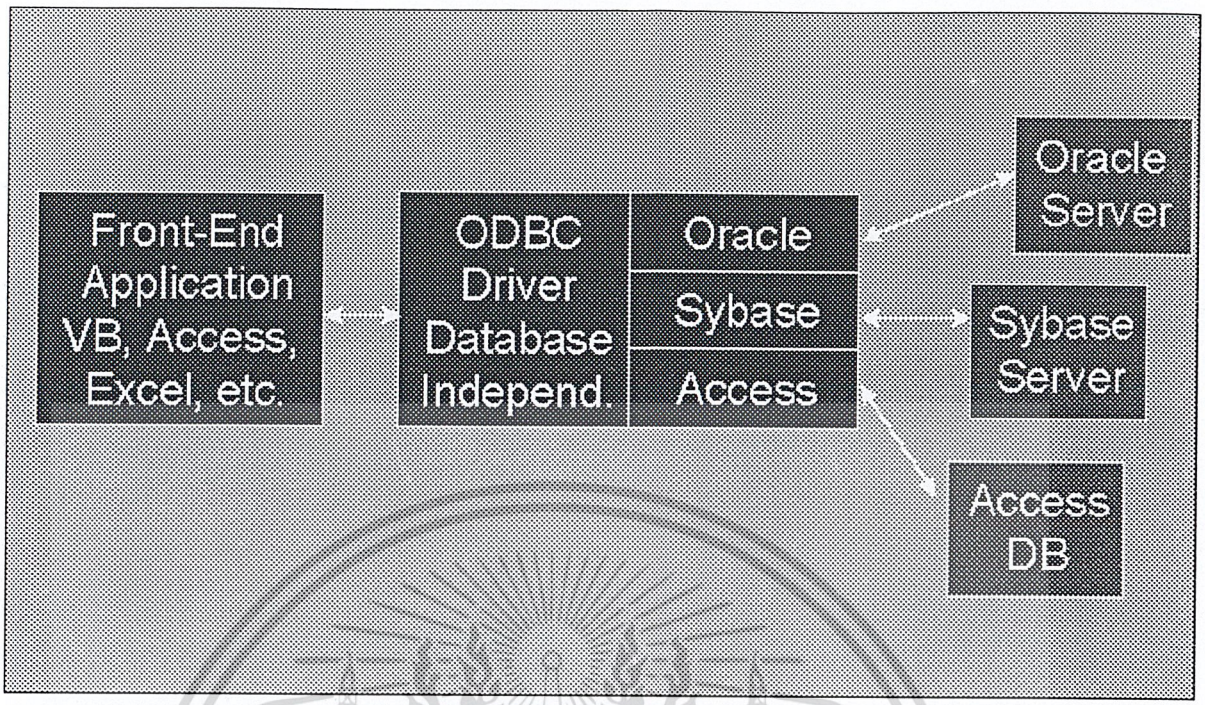


รูปที่ 4.2 แสดงลักษณะของ Handle

- **Driver Handle**

หรือ drh จะชี้ไปที่ชนิดของ database Database แต่ละชนิดจะมี driver 1 ตัวต่อ 1 database ถ้าเราติดต่อกับ database หลายๆ ตัว Driver handle จะไม่ติดต่อกับตัว database หรือแสดงการทำงานของ database แต่อย่างใด มันเพียงแค่เป็นตัวต่อระหว่างการเรียกของ database API ระดับต่ำกับ method ของ DBI

สำหรับการเชื่อมต่อกับฐานข้อมูลหลายรูปแบบ แสดงไว้ในรูปที่ 4.3



รูปที่ 4.3 แสดงการเชื่อมต่อกับฐานข้อมูลหลายรูปแบบ

- **Database Handle**

หรือ dbh รวบรวมการเชื่อมต่อ/การติดต่อเพียงหนึ่งเดียวให้กับ database ที่ให้มาผ่าน driver handle ซึ่งสามารถมีจำนวนของ database handle ต่อ driver handle เท่าไรก็ได้ เช่น เรามี script ที่คัดลอกข้อมูลจาก database ตัวหนึ่งไปยังอีกตัวหนึ่ง ดังนั้นเราจะมี driver handle เพียงแค่ 1 ตัว และ 2 database handle ซึ่งทำการติดต่อผ่าน driver handle

- **Statement Handle**

หรือ sth รวบรวมคำสั่งต่างๆ ให้กับฐานข้อมูล (database) ผ่าน database handle, statement handle สามารถมีจำนวนของ handle ต่อ database handle เท่าไรก็ได้ ตัวอย่างเช่น ถ้าเรามี 2 ตารางในฐานข้อมูลของเราตัวหนึ่งเก็บข้อมูลและอีกตัวเก็บข้อมูลเก่าที่คัดลอกไว้ และเรามีโปรแกรมที่ refresh ข้อมูลเก่าที่คัดลอกไว้จากข้อมูลดั้งเดิม เราสามารถใช้ 2 statement handle ตัวหนึ่งสำหรับ SELECT ข้อมูลจากตารางแรก และอีกตัวสำหรับ UPDATE ข้อมูลในตารางที่ 2. Statement handle เหล่านี้จะทำงานแบบ asynchronous

4.6. Emulation Layers

DBD::Oracle ได้สร้าง Oraperl emulation layer ซึ่งเป็น layer ของซอฟต์แวร์ที่เป็นตัวแปลงการเรียกของ Oraperl API ไปใน method ของ DBI ผลที่สุด code ของ Oraperl ที่มีอยู่แล้วจะสามารถทำงาน

ได้ตลอดโดยใช้ DBI และ DBD::Oracle ซึ่งหมายความว่าคุณสามารถเขียน code ใหม่โดยใช้ DBI interface ระหว่างการ maintain code ของ Oraperl หรือการย้าย code ไปที่ DBI

4.7. จุดประสงค์ของ DBI

จุดประสงค์ของ DBI API (Application Programming Interface) คือการกำหนดและสร้าง/ติดตั้งตัวเชื่อมต่อที่ใช้ร่วมกันได้ระหว่าง application ต่างๆ กับตัว database engine DBI จะยินยอมให้สร้าง script ที่เรียกใช้ฐานข้อมูลโดยไม่ต้องคำนึงถึง engine ที่กำลังใช้อยู่ สิ่งสำคัญคือ จำไว้ว่า DBI เป็นเพียงแค่ตัวเชื่อมต่อ (Interface) ตัวหนึ่ง, layer บางๆ ที่ยึดติดระหว่าง application และ driver ของฐานข้อมูล 1 ตัวหรือมากกว่า, มันเป็น driver ที่ทำงานจริงๆ DBI จะจัดให้มาตรฐานการเชื่อมต่อและ framework สำหรับ driver ต่างๆ ให้สามารถทำงานได้ภายในตัว DBI

แนวทางหลักการ:

1. ให้การเชื่อมต่อที่มีประสิทธิภาพ และนำมาใช้ประโยชน์ได้อย่างดี
2. Application พื้นฐานต่างๆ สามารถเรียกใช้ได้ง่าย
3. มีความยืดหยุ่นที่จะจัดการสิ่งที่ผิดปกติให้เหมาะสมหรือสิทธิของการทำงานและแม้แต่ข้อมูลที่ไม่ใช่ SQL ได้อย่างมีประสิทธิภาพ
4. ตรงตามมาตรฐานหรือคาดว่าจะใช้ได้ตามมาตรฐาน โดยเฉพาะ X/Open & SQL Access Group SQL และมาตรฐานของ CLI
5. สามารถที่จะสร้าง perl script ของ database_independent แต่ไม่จำกัดที่ฟังก์ชั่นที่ใช้ร่วมกันในระดับต่ำสุด
6. สนับสนุน dynamic loading ของ driver ต่างๆ ของฐานข้อมูล
7. สนับสนุนการเข้าถึงฐานข้อมูลหลายๆ engine ได้พร้อมๆ กัน
8. สามารถหาได้อย่างอิสระ (ไม่ต้องซื้อ/ไม่ใช่ทางการค้า)

4.8. ออราเพิล(Oraperl)

Oraperl เป็น “C” API ที่จะใช้ในการติดต่อกับ Oracle RDBMS ซึ่งได้ถูกพัฒนาขึ้นมา โดยจะมี 8 function หลักๆ ที่จะ access กับ Oracle database ผ่านทาง Oraperl ดังนี้

- ora_login
- ora_open
- ora_bind
- ora_fetch
- ora_close
- ora_commit
- ora_rollback
- ora_logoff

1. **ora_login**

ใช้ในการ connect database ซึ่งมีรูปแบบดังนี้

$dba = &ora_login ($database, $user, $password); โดย ora_login ต้องการ database name, Oracle userid และ Oracle password และจะได้รับ handle ในการเปิด database กลับมา

2. **ora_open**

ใช้เปิด cursor สำหรับรับ SQL statement ซึ่งมีรูปแบบดังนี้ :

$csr = &ora_open ($dba, $statement);

โดย ora_open ใช้ handle ที่ได้จาก ora_login และสร้าง cursor สำหรับ SQL statement โดย SQL statement สามารถเป็น Oracle SQL statement ใดก็ได้ที่ถูกต้อง และยังสามารถใช้ bind variable ได้ด้วย ตัวอย่างเช่น :

$csr = &ora_open ($dba, "select firstname from employee where id = 1234");

หรือถ้าใช้ bind variable จะเป็น :

$csr = $ora_open ($dba, "select firstname from employee where id = :1");

แต่การจะใช้ bind variable มีความจำเป็นที่จะต้องเรียกใช้ ora_bind ด้วย

3. **ora_bind**

จะถูกใช้เมื่อ ora_open มีการเรียกใช้ bind variable เพื่อที่จะบอกว่าค่าไหนที่จะถูกใช้ โดยมีรูปแบบดังนี้ :

$numrows = &ora_bind ($csr, $var1, $var2, …, $varN);

ซึ่งตัวอย่างการใช้ bind ในการ insert ทำได้ดังนี้ :

$csr = &ora_open ($dba, "insert into employee (id, firstname, lastname) values (:1, :2, :3)");

$numrows + &ora_bind ($csr, 1, "Michael", "Marolda");

4. **ora_fetch**

เมื่อ cursor ที่ได้มาจาก ora_open สำหรับ select statement

ora_fetch ถูกใช้เพื่อ retrieve ข้อมูลที่เกี่ยวข้องกับผลของ select statement ซึ่งมีรูปแบบดังนี้ :

@data = &ora_fetch ($csr);

ora_fetch จะ return array ของ value ที่เกี่ยวข้องกับ attribute name list ใน select statement เมื่อ fetch ได้รับผลลัพธ์ทั้งหมด undefined value จะถูกส่งกลับมา โดยตัวอย่างต่อไปจะเป็นการแสดงชื่อลูกค้าใน database ซึ่งมีรูปแบบดังนี้ :

$csr = &ora_open ($dba, "select id, firstname from employee");

while (($id, $firstname) = &ora_fetch ($csr)) {

```
        print "customer id = $id, customer name = $firstname\n";
    }
```

โดย ora_fetch ยังสามารถแสดงจำนวนของ field ของ Query ที่ return กลับมาได้ดังนี้

```
$nfields = &ora_fetch ($csr);
```

ซึ่งจะได้ค่า 2 ออกมา

5. ora_close

ใช้ในการ close และปล่อย resource ต่างๆ ของ cursor ซึ่งมีรูปแบบดังนี้ :

```
&ora_close ($csr);
```

6. ora_commit

ใช้ในการ commit การเปลี่ยนแปลง มีรูปแบบดังนี้ :

```
&ora_commit ($dba);
```

7. ora_rollback

ใช้เมื่อต้องการ rollback การเปลี่ยนแปลงใดที่เกิดขึ้นตั้งแต่การ commit ครั้งสุดท้าย มีรูปแบบดังนี้ :

```
&ora_rollback ($dba);
```

8. ora_logoff

ใช้ในการ free resource ทั้งหมด ที่ใช้ในการติดต่อกับ database มีรูปแบบคือ :

```
&ora_logoff ($dba);
```

บทที่ 5
ความรู้เบื้องต้นเกี่ยวกับภาษา Perl

5.1. บทนำ

Perl ได้ถูกพัฒนาขึ้นโดย UNIX Programmer ชื่อ Larry Wall ซึ่งงานของเขาต้องใช้ text file หลายๆ file เพื่ออ้างถึงการใช้งานพร้อมๆกัน ซึ่ง tool เดิมที่มีอยู่ใน UNIX ไม่สามารถอำนวยความสะดวกให้เขาได้มากนัก เขาจึงพัฒนาภาษา Perl ขึ้นมาโดยรวมเอาความสามารถต่างๆของหลายๆ Tool มาใช้โดยเฉพาะอย่างยิ่ง sed และ awk และยังได้เขียน interpreter สำหรับ Perl (perl)ขึ้นมา โดย Perl จะสามารถทำหลายๆอย่างที่ sed, awk และ UNIX Shell script ทำได้และสามารถทำได้ดีกว่าอีกด้วย

Perl เป็น freeware นั่นคือ source และ document ทั้งหมดสามารถ copy, compile, print หรือเผยแพร่โดยวิธีใดก็ได้ โปรแกรมที่ถูกเขียนโดย Perl จะเป็นสิทธิของผู้เขียนที่จะนำไปทำอย่างไรก็ได้

Perl เป็นคำย่อของ Practical Extraction and Report Language เป็นภาษาที่ถูกออกแบบมาสำหรับผู้ดูแลระบบ UNIX โดยได้รวมคุณลักษณะต่างๆ ของ C shell, Unix shell, sed และ awk Perl ได้กลายเป็นภาษาสำหรับการจัดการไฟล์ต่างๆ, การทำงานเกี่ยวกับการจัดการและงานของผู้ดูแลระบบที่เกี่ยวกับการแก้ไขความสามารถของการทำ pattern matching และดัดแปลงวากยสัมพันธ์ (syntax). แต่ Perl เป็นที่รู้จักกันอย่างแพร่หลายเมื่อถูกนำมาใช้ในการเขียน CGI

Perl เป็นภาษาที่ถูกแปรทีละบรรทัด ดังนั้นบ่อยครั้งที่เราเรียกโปรแกรม Perl ว่า "Perl script"

ตัวอย่างโปรแกรมที่เขียนด้วยภาษา Perl

โปรแกรมพื้นฐานที่ง่ายที่สุดคือ โปรแกรม "hello.pl"

```
#!/usr/local/bin/perl
#
# This is the famous Hello, world example
#
print 'Hello world.';          #Print the word
```

5.2. ส่วนประกอบต่างๆ ของโปรแกรม

1) บรรทัดแรก :

```
#! /usr/local/bin/perl
```

ในโปรแกรม Perl ส่วนใหญ่บรรทัดแรกจะเริ่มต้นด้วย "#!" (โดยเฉพาะสำหรับโปรแกรม CGI) ในบรรทัดนี้จะบอกคอมพิวเตอร์เพื่อหาตัวแปรภาษา (interpreter) ของ Perl และสั่งให้ตัวแปรภาษาทำการ execute โปรแกรม เราจะต้องระบุ พาร์ธ เต็มของระบบที่จะทำให้เข้าถึงที่เก็บตัวแปรภาษาของ Perl แต่ถ้าจะทำการรันโปรแกรมที่ command line ก็ไม่จำเป็นต้องมีบรรทัดแรกนี้เพียงแค่ใช้คำสั่ง

```
perl     hello.pl
```

ถ้าเราไม่รู้ว่า Perl ถูกเก็บอยู่ที่ directory ไหนเราสามารถใช้คำสั่ง

which perl

เพื่อทำการค้นหา directory ที่เก็บตัวแปรภาษาของ perl

2) คำอธิบาย (comment)

ถ้าต้องการจะพิมพ์คำอธิบายใดๆ ให้เริ่มต้นด้วย "#" ส่วนต่างๆ ที่ตามหลังเครื่องหมายนี้จนจบบรรทัดจะไม่ถูกแปรโดยตัวแปรภาษาของ perl

3) คำสั่ง (statement)

ส่วนใดๆ ของโปรแกรมที่ต้องการให้ตัวแปรภาษาทำการแปร จะลงท้ายด้วยเครื่องหมาย ";"

5.3. ตัวแปรและชนิดข้อมูล

Perl มีตัวแปรต่างๆ 3 ชนิดด้วยกันคือ

1). Scalar Variables

เป็นตัวแปรที่เก็บข้อมูลชนิดตัวเลข หรือสตริง (ชุดของอักขระ) ตัวแปรสเกลาร์จะขึ้นต้นด้วยเครื่องหมายดอลลาร์ไซยน์ "$" ตัวอย่างต่อไปนี้เป็นตัวอย่างของตัวแปรชนิดสเกลาร์

```
$str1="www";              #assign string "www" to scalar variable $str1
$str2="cgi";              #assign string "cgi" to scalar variable $str2
$str3=$str1.$str2, #concatenate two strings so $str3="wwwcgi"
$int=5;                   #assign integer 5 to scalar variable $int
$int=3+2;                 #here $int=5
$int=3*4;                 #here $int=12
$int=5;$int++;            #here $int=5+2=6
$int=5;$int+=4;           #here $int=5+4=9
```

2). Array (อาร์เรย์)

ตัวแปรชนิดอาร์เรย์จะขึ้นต้นด้วยเครื่องหมาย "@" แต่เวลาที่จะระบุสมาชิดในอาร์เรย์คุณสามารถกำหนดมันโดยการขึ้นต้นด้วยเครื่องหมาย "$" ตัวอย่างของตัวแปรอาร์เรย์เช่น

```
@name1=("viky", "jeff"); #assign "viky", "jeff" two strings to array @name1
@name2=@name1;           #now @name2=("viky", "jeff")
@name3=("john", @name1); #now @name3=("john", "viky", "jeff")
($one,@name4)=@name3;#now @one="john", @name4=("viky", "jeff")
@name1=();               #now @name1 becomes an empty array
@int=(2, 6,7, 8, 9);     #Five numbers are assigned to array @int
$x=@int;                 #assign an array to an scalar variable will return the
                         #number of elements in the array, so $x=5
```

```
$x=$#int;                    # $# is a special variable which will return the index of the
                             last element in an array, so $x=4
($x)=@int                    #$x is equal to the first element of array, so $x=2
$b=$int[0];                  #$b is equal to the first element in array, so $b=2
$c=@int[0];                  #$c is also equal to the first element in array, so $c=2
                             # $int[0] works the same way as @int[0]
$int[0]=5;                   #assign 5 to be the first element of array, so @int=
                             (5,6,7,8,9)
$int[0,1]=[1,3]              #assign 1 to the first element, 3 to be the second element,
                             so @int=(1,3,7,8,9)
@int[0,1]=@int[1,0]          #the first two element exchange, now @int=(3,1,7,8,9)
@data=@int[0,1]          #here @data=(1,3)
$int[5]=10                   #assigns the sixth element to @int so @int=(1,3,7,8,9,10)
```

3). Associate Array

บางครั้งเรียกว่า "hashes" Associate Array เป็นอาร์เรย์ที่ไม่ได้ถูกชี้ (index) โดยตัวเลข แต่จะใช้คำของตัวอักษร โดยทั่วไป สมาชิกของ Associate Array จะถูกอ้างโดยใช้รูปแบบ "key" และ "value" key จะเป็นตัวชี้ในการเข้าไปที่เก็บข้อมูล "value" Associate Array จะเขียนขึ้นต้นด้วยเครื่องหมาย "%"

รูปแบบของ Associate Array คือ

%arrayname = (key1, value1, key2, value2, key3, value3, ...)

สำหรับแต่ละ key ใน associate array จะมีค่า value ที่ตรงกับแต่ละ key ที่อ้างถึง value นั้นๆ

- การเพิ่ม หรือ การเปลี่ยนค่า แต่ละคู่สมาชิกใน associate array

```
$arrayname {key} = value
```

ในการเพิ่มคู่สมาชิก key_value ใน associate array ลำดับแรกขึ้นต้นชื่อของ array ด้วยเครื่องหมาย "$" และใส่ค่า key อยู่ภายในเครื่องหมายปีกกา { } จากนั้นจึงกำหนดค่า value ให้กับ key ถ้า key นั้นมีอยู่แล้วใน array ค่า value ของ key นั้นก็จะถูกเปลี่ยนแปลง

+ การรับค่า value ของ key

```
$value = $arrayname {key}
```

- การลบคู่สมาชิก key_value ใช้คำสั่ง

```
delete $arrayname{key}
```

ซึ่งคำสั่ง delete นี้เป็นฟังก์ชั่นที่ให้มาโดย Perl

ต่อไปนี้เป็นตัวอย่างของ associate array

```
%name=('john', 'biology', 'jeff', 'physics');          #suppose john teaches biology and jeff teaches
                                                       physics
$a=$name{'john'};                                      #here $a='Biology'
$b=$name{'jeff'};                                      #here $b='physics'
$name{'bill'}='math';                                  #add a new pair of element : key is bill, its value
                                                       is math.
                                                       #now %name=('john', 'biology', jeff', 'physics',
                                                       'bill', 'math')
$name{'bill'}='English';                               # Because we have already have a key 'bill' in the
                                                       associate array,
                                                       # so the 'bill' value change to 'English'
                                                       # now %name=('john', 'biology', 'jeff', 'physics',
                                                       'bill', 'English')
delete $name{'jeff'};                                  # now  #now %name=('john', 'biology', 'jeff',
                                                       'physics', 'bill', 'English')
@X=%name                                               # associate array assign to array @X
%name=@X                                               # array assign to associate array.
```

5.4. String (สตริง)

ในการอ้างถึงสตริงใน Perl ใช้อักขระในการอ้างถึง 3 ประเภท:

1. '(apostophe)

อักษรที่อยู่ระหว่างเครื่องหมาย apostrophe จะถูกแปรเป็นตามตัวอักษร (literally) ไม่ต้องมีตัวแปรในการเขียนเครื่องหมายนี้ลงไปในสตริงจะต้อง

```
$string1 = 'chocolate';
$string2 = 'I love $string1';
# The value of $string2 is "I love $string1"
```

2. "(double quotes)

เครื่องหมาย double quote จะแทรกอยู่ระหว่างตัวแปรดังนี้

```
$string1 = "chocolate";
$string2 = "I love $string1";
#Now the value of $string2 is "I love chocolate"
```

3. ‘(backtick)

Backtick โดยปกติจะเป็นการอ้างถึงคำสั่งของ UNIX shell ผลของการ execute จะถูกส่งกลับเหมือนเป็นค่าของสตริง Backtick จะอยู่ระหว่างตัวแปร และในการเขียน backtick ลงไปในสตริงจะต้องนำหน้าด้วยเครื่องหมาย "\"

```
$document = "/home/document.txt";
$WordCount = WC -w $document;
# $wordCount get the number of words in the "/home/document.txt" file.
# here is a UNIX command which counts the number of lines, words or characters of a life
```

5.5. โอเปอร์เรเตอร์(Operator)

5.5.1. โอเปอร์เรเตอร์ทางการคำนวณ(Arithmetic operators)

ตัวอย่างการใช้งานของโอเปอร์เรเตอร์ทางการคำนวณแสดงในตารางที่ 5.1 ดังนี้

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
+	$x + $y	Addition
-	$x - $y	Subtraction
*	$x * $y	Multiplication
/	$x / $y	Division
%	$x % $y	Modulus
**	$x ** $y	Exponentiation

ตารางที่ 5.1 แสดงโอเปอร์เรเตอร์ทางการคำนวณ

5.5.2. โอเปอร์เรเตอร์ทางตรรกะ(Logical operators)

ตัวอย่างการใช้งานของโอเปอร์เรเตอร์ทางตรรกะแสดงในตารางที่ 5.2 ดังนี้

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
&&	$x && $y	ถ้า $x เป็นจริง แปรค่า $y ว่าเป็นจริงหรือไม่ ถ้า $y ก็เป็นจริง ส่งค่าจริง(true)กลับ ถ้า $x เป็นเท็จ ส่งค่าเท็จกลับ
||	$x || $y	ถ้า $x เป็นจริง ส่งค่าจริงกลับ ถ้า $x เป็นเท็จ แปรค่า $y ถ้า $y เป็นจริง ส่งค่าจริงกลับ มิฉะนั้นส่งค่าเท็จกลับ
!	! $x	ถ้า $x เป็นจริง ส่งค่าเท็จกลับ ถ้า $x เป็นเท็จ ส่งค่าจริงกลับ

ตารางที่ 5.2 แสดงโอเปอร์เรเตอร์ทางตรรกะ

5.5.3. โอเปอร์เรเตอร์ทางด้ารการกำหนดค่า(Assignment operators)

ตัวอย่างการใช้งานของโอเปอร์เรเตอร์ทางด้านการกำหนดค่าแสดงในตารางที่ 5.3 ดังนี้

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
=	$var = 5;	Assign 5 to *$var*.
+=	$var += 3;	Add 3 to *$var* and assign result to *$var*.
- =	$var -= 2;	Subtract 2 from *$var* and assign result to *$var*.
.=	$str.= “ing”;	Contatenate “ing” to $str and assign result to *$str*.
*=	$var *=4;	Multiply *$var* by 4 and assign result to *$var*.
/=	$var /= 2;	Divide *$var* by 2 and assign result to *$var*.
**=	$var **= 2;	Square *$var* and assign result to *$var*.
%=	$var %= 2;	Divide *$var* by 2 and assign remainder to *$var*.
X=	$str x= 20;	Repeat value of *$str* 20 times and assign result to *$str*.
<<=	$var <<= 1;	Left-shift bits in *$var* one position and assign result to *$var*.
>>=	$var>>= 2;	Right-shift bits in *$var* two positions and assigh result to *$var*.
&=	$var &= 1;	One is Bitwise-ANDed to *$var* and the result is assigned to *$var*.
|=	$var != 2;	Two is Bitwise-Ored to *$var* and the result is assighed to *$var*.
^=	$var ^= 2;	Two is Bitwise-ExclusiveORed to *$var* and the result is assigned to *$var*.

ตารางที่ 5.3 แสดงโอเปอร์เรเตอร์ทางด้ารการกำหนดค่า

5.5.4. โอเปอร์เรเตอร์ทางด้านการเปรียบเทียบสตริง(Relational operators used in string context)

ตัวอย่างการใช้งานของโอเปอร์เรเตอร์ทางด้านการเปรียบเทียบสตริงแสดงในตารางที่ 5.4 ดังนี้

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
Gt	$str1 gt $str2	$str1 is greater than $str2
Ge	$str1 ge $str2	$str1 is greater than or equal to $str2
Lt	$str1 lt $str2	$str1 is less than $str2
Le	$str1 le $str2	$str1 is less than or equal to $str2
Eq	$str1 eq $str2	$str1 is equal to $str2

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
Ne	$str1 ne $str2	$str1 is not equal to $str2
Cmp	$str1 cmp $str2	$str1 is compared to $str2, with signed return

ตารางที่ 5.4 แสดงโอเปอร์เรเตอร์ทางด้านการเปรียบเทียบสตริง

5.5.5. โอเปอร์เรเตอร์ทางด้านการเปรียบเทียบค่าทางตัวเลข(Relational operators used in numeric context)

แสดงตัวอย่างดังในตารางที่ 5.5 ดังนี้

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
>	$x > $y	$x is greater than $y
>=	$x >= $y	$x is greater than or equal to $y
<	$x < $y	$x is less than $y
<=	$x <= $y	$x is less than or equal to $y
==	$num1 == $num2	$num1 is equal to $num2
!=	$num1 != $num2	$num1 is not equal to $num2
<=>	$num1 <=> $num2	$num1 is compared to $num2 with signed return; 1 if $num1 is greater than $num2, 0 if $num1 is equal to $num2, and –1 if $num1 is less than $num2

ตารางที่ 5.5 แสดงโอเปอร์เรเตอร์ทางด้านการเปรียบเทียบค่าทางตัวเลข

5.6. คำสั่งเงื่อนไข(Condition Statement) และคำสั่งการวนซ้ำ(Loop)

5.6.1. if

เป็นคำสั่งเงื่อนไขที่ยินยอมให้ทดสอบเอ็กซ์เพรสชั่น (Expression) เพื่อการตัดสินใจ. เอ็กซ์เพรสชั่นจะอยู่ภายในเครื่องหมายวงเล็บ "( )" และเพิร์ล (Perl) จะตีความหมายของเอ็กซ์เพรสชั่นในขอบเขตของสตริงถ้าสตริงไม่เป็น null เอ็กซ์เพรสชั่นจะเป็นจริง ถ้าเป็น null เอ็กซ์เพรสชั่นจะเป็นเท็จ. ถ้าเอ็กซ์เพรสชั่นเป็นค่าของตัวเลข มันจะถูกแปลงเป็นสตริงก่อนถูกนำมาตรวจสอบ. ถ้าเอ็กซ์เพรสชั่นถูกทดสอบได้ค่าเป็นจริง คำสั่งในบล็อคถัดมาจะถูกทำงาน แต่ถ้าเอ็กซ์เพรสชั่นเป็นเท็จ เพิร์ลจะทำการข้ามไปทำคำสั่งถัดไปในสคริปต์ (script)

<u>รูปแบบคำสั่ง if</u> ประกอบด้วยคีย์เวิร์ด if ตามด้วยเอ็กซ์เพรสชั่นของเงื่อนไขแล้วตามด้วยบล็อคของคำสั่งภายในเครื่องหมายปีกกา ({ }) แต่ละคำสั่งภายในบล็อคจะถูกแบ่งแยกด้วยเครื่องหมายเซมิโคลอน (;) เช่น

if (Expression) {Block}

if (Expression) {Block} else {Block}

if (Expression) {Block} elsif (Expression) {Block} ... else {Block}

รูปแบบคำสั่ง if/else อีกรูปแบบหนึ่งของคำสั่ง if ก็คือ if/else. รูปแบบคำสั่ง if/else นี้จะให้มีทางเลือกของการตัดสินใจ 2 ทาง. ถ้าเอ็กซ์เพรสชั่นของเงื่อนไขแรกที่ตามหลังคีย์เวิร์ด if เป็นจริง บล็อคคำสั่งที่ตามหลัง ifจะถูกทำงาน. ไม่เช่นนั้น ถ้าเอ็กซ์เพรสชั่นของเงื่อนไขที่ตามหลัง if เป็นเท็จ คำสั่งที่อยู่ตามหลังคีย์เวิร์ด elseจะถูกทำงานแทน

if (Expression)

{Block}

else

{Block}

รูปแบบคำสั่ง if/elsif/else ยังมีอีกรูปแบบหนึ่งของคำสั่ง if ก็คือคำสั่ง if/else/elsif. รูปแบบคำสั่งนั้นทำให้มีทางเลือกของการตัดสินใจหลายทาง. ถ้าเอ็กซ์เพรสชั่นของเงื่อนไขแรกที่ตามหลังคีย์เวิร์ด if เป็นจริง บล็อคคำสั่งที่ตามหลัง if จะถูกทำงาน ไม่เช่นนั้นคำสั่ง elsif ก็จะถูกตรวจสอบ. ถ้าเอ็กซ์เพรสชั่นของเงื่อนไขที่ตามหลังคำสั่ง elsif แรกเป็นเท็จ elsif ตัวถัดมาก็จะถูกตรวจสอบแทน และถ้าเอ็กซ์เพรสชั่นของเงื่อนไขที่ตามหลัง elsif ทั้งหมดเป็นเท็จ. บล็อคคำสั่งที่อยู่ตามหลัง else ก็จะถูกทำงานแทน

if (Expression1)

{Block}

elsif (Expression2)

{Block}

elsif (Expression3)

{Block}

else

{Block}

5.6.2. unless

รูปแบบคำสั่ง unless คล้ายกับคำสั่ง if เพียงแต่ว่าจะทำคำสั่งที่อยู่ตามหลังคีย์เวิร์ด unless ก็ต่อเมื่อเงื่อนไขที่เอ็กซ์เพรสชั่นที่อยู่หลัง unless เป็นเท็จ

unless/else และ unless/elsif ก็กระทำตนเหมือนกับ if/else และ if/elsif

unless (Expression) {Block}

unless (Expression) {Block} else {Block}

unless (Expression) {Block} elsif (Expression) {Block} ... else {Block}

5.6.3. while Loop

คำสั่ง while จะทำงานบล็อคคำสั่งจนกว่าเอ็กซ์เพรสชั่นของการควบคุมที่ตามหลังคีย์เวิร์ด while จะเป็นเท็จเอ็กซ์เพรสชั่นจะเป็นจริง ถ้ามันไม่เป็นศูนย์หรือไม่เป็น null เช่น while (1) จะเป็นจริงเสมอและ Loop จะถูกทำงานตลอด. เอ็กซ์เพรสชั่นจะเป็นเท็จถ้ามันถูกตีค่าเป็นศูนย์ (null) เช่น while (0)

<u>รูปแบบคำสั่ง while</u>

while (Expression) {Block}

5.6.4. until Loop

คำสั่ง until จะทำงานบล็อคคำสั่งตราบเท่าที่เอ็กซ์เพรสชั่นของการควบคุมที่ตามหลัง until เป็นเท็จหรือศูนย์. เมื่อใดก็ตามที่เอ็กซ์เพรสชั่นถูกตีค่าเป็นจริงจะออกจากรูปการทำงานทันที

5.6.5. for Loop

คีย์เวิร์ด for จะถูกตามด้วย 3 เอ็กซ์เพรสชั่น ซึ่งคั่นจากกันด้วยเครื่องหมายเซมิโคลอน (;) และอยู่ภายในเครื่องหมายวงเล็บ "( )". เอ็กซ์เพรสชั่นแรกจะใช้ในการกำหนดค่าตัวแปรเริ่มต้น. เอ็กซ์เพรสชั่นที่สองจะใช้การทดสอบการทำงานของลูปว่าจะหยุดเมื่อใด และเอ็กซ์เพรสชั่นตัวที่สามจะเป็นตัวเปลี่ยนแปลงตัวแปรของรูป

<u>รูปแบบคำสั่ง for</u>

for (Expression1 ; Expression2 ; Expression3) {Block}

5.6.6. foreach Loop

จะทำงานบนแต่ละอีลีเมนท์ (element)ที่อยู่ภายในเครื่องหมายวงเล็บ "( )" ซึ่งเก็บเป็นตัวแปรอาร์เรย์ และจะกำหนดค่าของแต่ละอิลีเม้นท์ของอาร์เรย์ให้กับตัวแปรสเกลาร์จนกระทั่งอาร์เรย์ว่าง(หมด)

<u>รูปแบบคำสั่ง foreach</u>

foreach VARIABLE (ARRAY)

{Block}

5.6.7. Loop Control

5.6.7.1. Label สามารถใช้ในการควบคุมทิศทางของลูป. แต่ด้วยตัวของลาเบลเองมันจะไม่ทำอะไร มันจะถูกใช้กับตัว modifier ของตัวควบคุมลูป โดยมีรูปแบบดังนี้

LABEL: while (Expression) {Block}

LABEL: while (Expression) {Block} continue {Block}

LABEL: for (Expression; Expression / Expression)

{Block}

LABEL: foreach Variable (Array) {Block}

LABEL: {Block} continue {Block}

สำหรับการควบคุมทิศทางของลูป จะใช้คำสั่งต่อไปนี้ ซึ่งอาจจะถูกใช้อยู่ภายในบล็อคคำสั่ง

nextn

next LABEL

last

last LABEL

redo

redo LABEL

goto LABEL

คำสั่ง next จะสั่งให้กลับไปที่จุดเริ่มต้นของลูปอีกครั้งตามที่กำหนดและไปตีความเอ็กซ์เพรสชั่นอีกครั้ง

คำสั่ง last จะเป็นการหยุดและออกจากบล็อคคำสั่งเหมือนกับคำสั่ง break ในภาษา C

คำสั่ง redo จะให้กลับไปเริ่มทำคำสั่งของบล็อคคำสั่งอีกครั้งโดยไม่ต้องตีความเอ็กซ์เพรสชั่นอีกครั้ง

คำสั่ง continue จะถูกทำงานก่อนเอ็กซ์เพรสชั่นเงื่อนไขคือจะถูกตีความอีกครั้ง

คำสั่ง goto จะสั่งให้กระโดดไปทำงานตามที่กำหนด

Label สามารถกำหนดไว้ที่ใดก็ได้ในสคริปต์ แต่มันจะไม่ทำงานเมื่ออยู่ข้างในคำสั่ง do หรือภายใน subroutine

5.6.7.2. Labeled Block โดยไม่มีลูป Block ก็เหมือนกับลูปที่ถูกทำงานเพียงครั้งเดียว. Block สามารถกำหนดเป็น label ได้. คำสั่ง redo จะทำให้เริ่มต้นทำงานที่ตอนต้นของบล็อค (Block)โดยไม่ต้องมาตีความเอ็กซ์เพรสชั่นอีกครั้ง

เช่น

```
ATTEMT : {
        .
    ชุดคำสั่ง
        .
        .
    redo ATTEMPT unless (Expression) ;
}
```

5.6.7.3 ลูปซ้อนลูปและลาเบล (Nested Loops and Labels) ลูปที่อยู่ภายในลูปอีกอันหนึ่งเราเรียกว่า "Nested Loop". ลูปที่อยู่ข้างนอกจะถูกกำหนดค่าเริ่มต้นและถูกตรวจสอบ แล้วลูปที่อยู่ข้างในจึงถูกทำงานตาม. ลูปที่อยู่ข้างในจะมีการเคลื่อนที่เร็วกว่าลูปที่อยู่ข้างนอก. ลูปสามารถซ้อนกันกี่ระดับก็ได้ตามที่เราต้องการ. โดยปกติถ้าเราใช้คำสั่งควบคุมลูป อย่างเช่น next และ last การควบคุม.

การควบคุมจะขึ้นอยู่กับลูปที่อยู่ในสุด ด้วยการใช้ลาเบลนำหน้าลูปจะทำให้เราสามารถควบคุมทิศทางของโปรแกรม ซึ่งเราอาจจะใช้คำสั่ง last, next และ redo เช่น

```
OUT : while (1) {
        <Program continues here>
        MID : while (1) {
                If ( <expression is true> ) {last OUT;}
                <Program continue here>
                INNER : while (1) {
                        If ( <expression is true> ) {next OUT;}
                        <Program continues here>
                }
        }
}
print "Out of all loops.\n" ;
```

5.7. File I/O

ในภาษา Perl ข้อมูลสามารถอ่านจากไฟล์อินพุทมาตรฐาน (STDIN) และเขียนไปที่ไฟล์เอาท์พุทมาตรฐาน (STDOUT) หรือไฟล์ error มาตรฐาน (STDERR) ดังเช่นตัวอย่างต่อไปนี้

```
$inputline = <STDIN>
```

อ่านข้อมูลจากอินพุทไฟล์มาทีละบรรทัดและ

```
print (STDOUT, "This is the content to write:);
```

ทำการเขียนข้อมูลสตริงไปที่เอาท์พุทไฟล์

<u>การเข้าถึงไฟล์ทั่วๆ ไป</u>

1. เปิดไฟล์โดยการเรียกใช้ฟังก์ชั่น open เช่น

```
open (MYFILE, "/home/pub/myfile");
```

MYFILE คือตัวแปรไฟล์เป็นตัวแปรตัวแรก ตัวแปรทั้งสองเป็นชื่อ พาร์ธ ของไฟล์ โดยปกติ Perl จะสมมติว่าเราต้องการจะทำการไฟล์ที่เราทำการเปิด สำหรับการเปิดไฟล์เพื่อการเขียนให้ใส่เครื่องหมาย ">" ไว้หน้าชื่อไฟล์ ดังนี้

```
open (MYFILE, ">/home/pub/myfile");
```

เมื่อคุณเปิดไฟล์เพื่อจะทำการเขียน ข้อมูลเดิมในไฟล์จะถูกทำลาย ถ้าต้องการจะเพิ่มข้อมูลเข้าไปในข้อมูลเดิมที่มีอยู่แล้วให้ใส่เครื่องหมาย ">>" ไว้หน้าชื่อไฟล์ ดังเช่น

```
open (MYFILE, ">>/home/pub/myfile")
```

2. เขียนหรืออ่านไฟล์โดยทำเช่นเดียวกับการเขียนหรืออ่าน STDIN และ STDOUT

$inputline = <MYFILE>

อ่านข้อมูลจากไฟล์ทีละบรรทัดและ

5.8. Subroutines

การประกาศสับรูทีน (subroutine) สามารถประกาศไว้ที่ส่วนใดๆ ก็ได้ภายในสคริปต์และสับรูทีนนี้จะสามารถมองเห็นหรือเรียกใช้จากส่วนใดก็ได้แม้แต่ไฟล์อื่น. แต่เมื่อจะมีการเรียกใช้สับรูทีนจากไฟล์อื่น เราจะต้องทำการโหลดสับรูทีนนั้นเข้ามาในสคริปต์โดยการใช้คำสั่ง do, require หรือ use

<u>รูปแบบการประกาศสับรูทีน</u>

sub subroutine_name

{Block คำสั่งเพิร์ล}

<u>รูปแบบการเรียกใช้สับรูทีน</u>

do subroutine_name ;

&subroutine_name ;

subroutine_name ( ) ;

subroutine_name ;

<u>รูปแบบการเรียกใช้สับรูทีนด้วยพารามิเตอร์</u>

&subroutine_name (parameter1, parameter2, ...)

subroutine_name (parameter1, parameter2, ...)

อาร์กิวเมนท์ที่ถูกส่งเข้าไปในสับรูทีนจะถูกเก็บอยู่ในตัวแปรอาร์เรย์ @_

ถ้าเราต้องการจะตรวจสอบค่าสับรูทีนว่าได้ถูกประกาศไว้แล้วหรือยัง สามารถตรวจสอบได้โดยการเรียกใช้ฟังก์ชั่น defined

ค่าสับรูทีนส่งกลับออกมาจะเป็นค่าที่ถูกกระทำล่าสุดก่อนออกจากสับรูทีน

5.9 โมดูล (Module)

โมดูล ก็คือแพ็คเกจที่สามารถนำกลับมาใช้ได้อีกครั้ง ซึ่งโดยปกติจะถูกกำหนดอยู่ในไลบรารี่. โมดูลจะเป็นไฟล์ที่มีส่วนขยายต่อท้ายไฟล์ด้วย ".pm"

<u>การเรียกใช้โมดูล</u> มีรูปแบบดังนี้

use Module ;

use Module (list) ; เป็นการเรียกใช้หลายๆ โมดูล

use Directory :: Module ; เรียกใช้โมดูลที่ซ่อนอยู่ใต้ directory ในไลบรารี่

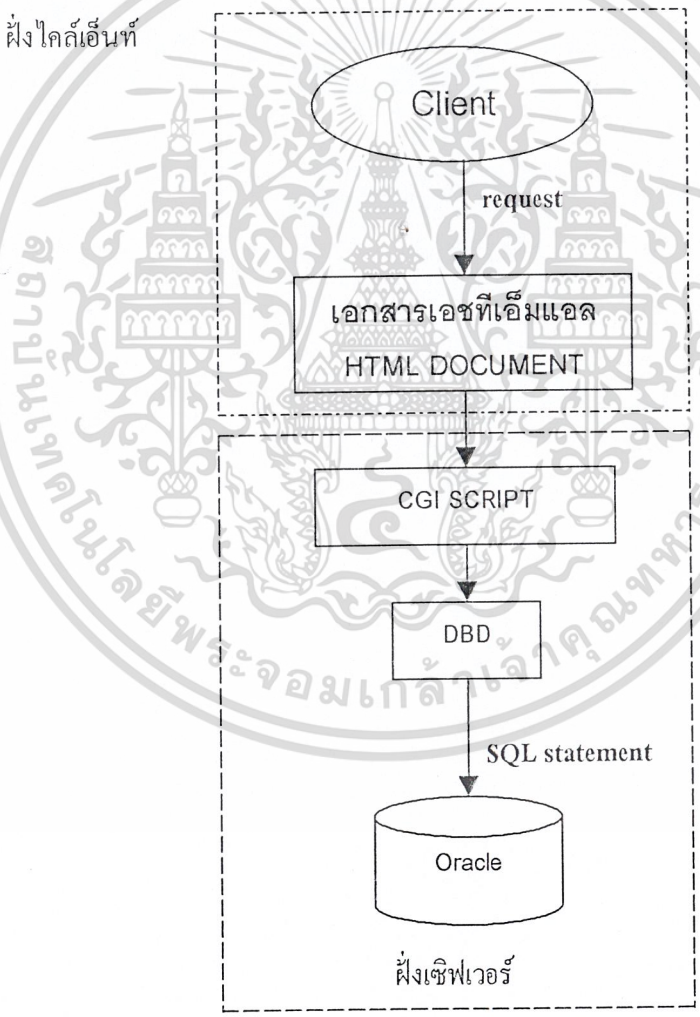
use pragma (list) ; เรียกใช้โดยใช้ progma

บทที่ 6
การดำเนินงานโครงงาน

6.1 ขั้นตอนการศึกษาความเป็นไปได้ของโครงงาน

ทำการศึกษาวิธีการเชื่อมต่อกันระหว่างเซิร์ฟเวอร์กับฐานข้อมูลที่มีอยู่ โดยได้ข้อสรุปว่า จะทำโดยใช้ภาษาเพิร์ล และสามารถทำงานได้บนระบบยูนิกซ์ รวมถึงมีโมดูลมาตรฐานที่ถูกเขียนขึ้นมาเพื่อใช้ในการติดต่อกับฐานข้อมูลใดๆก็ได้ ซึ่งได้แก่ ดีบีไอ (DBI : DataBase Interconnection) และมีโมดูลที่เพิ่มเข้าไปเพื่อระบุชนิดของฐานข้อมูลเป็นการเฉพาะเจาะจง ซึ่งได้แก่ ดีบีดี (DBD : DataBase Driver) ซึ่งในที่นี้ใช้ ดีบีดี : ออราเคิล ซึ่งสามารถหาดาว์นโหลดได้จากอินเทอร์เนต

6.2 ขั้นตอนการออกแบบระบบ



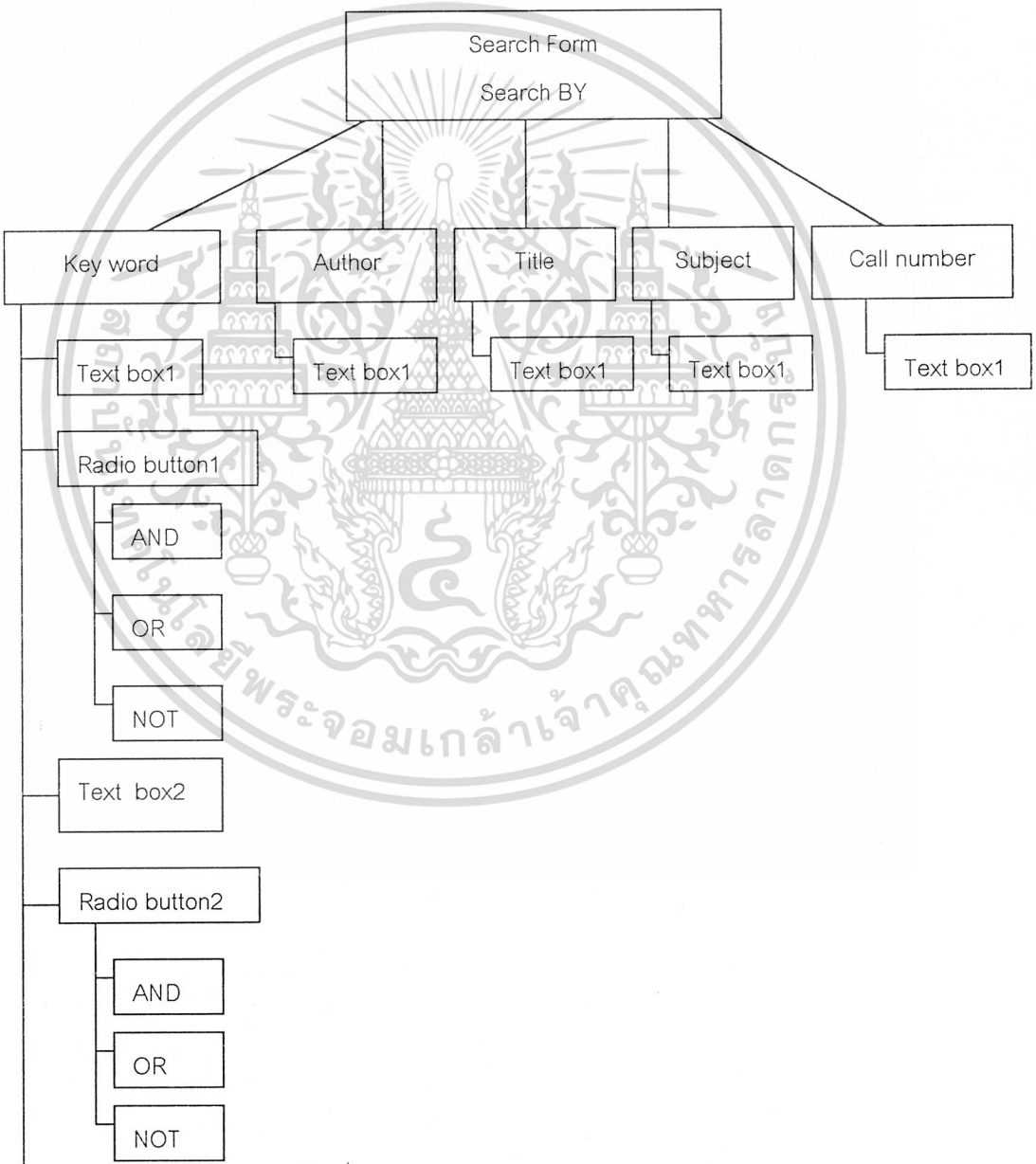
รูปที่ 6.1 แสดงการทำงานโดยรวมของระบบ

จากรูปที่ 6.1เมื่อไคล์เอ็นท์ส่งสัณญาณร้องขอเอกสารเอชทีเอ็มแอลมาที่ฝั่งเซิฟเวอร์ เซิฟเวอร์จะส่งเอกสารเอชทีเอ็มแอลกลับไปให้ที่ไคล์เอ็นท์ หลังจากที่ผู้ใช้กรอกข้อมูลที่ต้องการค้นหาแล้วกดปุ่ม submit ในแบบฟอร์มของเว็บเพจนั้นแล้ว ที่เซิฟเวอร์จะไปเรียกไฟล์ซีจีไอสคริปต์ขึ้นมาประมวลผลข้อมูลที่ได้รับจากฝั่งไคล์เอ็นท์

ภายในสคริปต์จะมีคำสั่งที่ไปเรียกใช้งานดีบีดี เพื่อทำการติดต่อเข้าไปที่ฐานข้อมูล แล้วทำการคิวรีข้อมูลตามที่ต้องการ จากนั้นก็ส่งข้อมูลที่ได้กลับไปที่ฝั่งไคล์เอ็นท์อีกครั้งหนึ่ง

- **การออกแบบส่วนประกอบของแบบฟอร์ม**

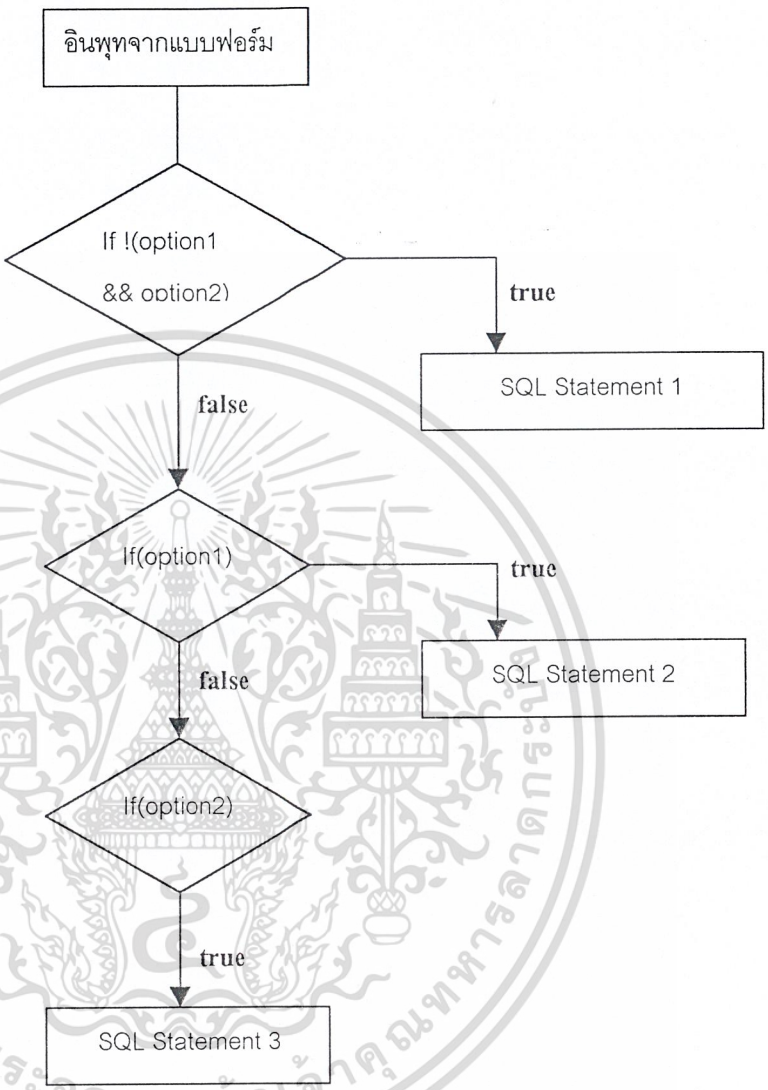
ความสัมพันธ์ของส่วนประกอบต่างในแบบฟอร์มแสดงดังในรูปที่ 6.2



รูปที่ 6.2 แสดงไดอะแกรมของส่วนประกอบของแบบฟอร์ม

- Flow-chart diagram

Flow-chart diagram ของขั้นตอนการพิจารณาการเรียกใช้งานคำสั่งเอสคิวแอลแสดงดังในรูปที่ 6.3



รูปที่ 6.3 แสดง Flow-chart diagram ของ SQL statement

6.3. ขั้นตอนทดสอบ และติดตั้ง

เนื่องจากความไม่สะดวกในหลายๆด้าน ในการที่จะไปทดลองการทำงาน ณ สถานที่จริงที่จะนำโครงงานนี้ไปประยุกต์ใช้งาน จึงจำเป็นที่จะต้องมีการจำลองระบบ โดยระบบที่นำมาสร้างเป็นเครื่องทดสอบมีรายละเอียดในการทดลองดังนี้

6.3.1.ทดสอบการติดตั้งภาษาเพิร์ล

โดยมีขั้นตอนการติดตั้งดังนี้

- sh Configure เป็นการคอมไพล์โปรแกรมและตรวจสอบค่าคอนฟิกูเลชั่นต่างๆของระบบ ไลบรารีที่ต้องการใช้

- make เป็นคำสั่งที่ช่วยเรียกตัวคอมไพล์โปรแกรม โดยนำค่าที่ได้จาก config.sh ที่ได้มาจากขั้นตอนแรกมาใช้
- make test ทำการทดสอบการติดตั้ง
- make install ทำการติดตั้งโปรแกรมจริง

6.3.2.ทดสอบการติดตั้ง อาปาเช่เว็ปเซิร์ฟเวอร์ (apache web server)

โดยมีขั้นตอนการติดตั้งดังนี้

- configure
- make
- make install

และทำการตั้งค่าคอนฟิกูเลชั่นต่างๆของเว็ปเซิร์ฟเวอร์ โดยแบ่งไฟล์ที่เกี่ยวกับการคอนฟิกูเลชั่นค่าต่างๆ ดังนี้

- access.conf เป็นไฟล์ที่กำหนดบริการที่มีการอนุญาตให้ใช้
- httpd.conf เป็นคอนฟิกูเลชั่นหลักของเซิร์ฟเวอร์
- srm.conf เป็นไฟล์ที่กำหนดค่าต่างๆ เช่นเส้นทางการเข้าถึงเอกสาร,ผลของการร้องขอใช้บริการ,ผลลัพธ์ได้ถูกแสดงผลในรูปแบบใด

6.3.3.ขั้นตอนการติดตั้งโปรแกรม

หลังจากได้ทำการทดสอบการติดตั้งจนมั่นใจแล้ว จึงเริ่มทำการติดตั้งโปรแกรมต่างๆที่เกี่ยวข้องลงไปบนระบบจริง ดังมีขั้นตอนดังต่อไปนี้

6.3.3.1 ติดตั้ง gzip ซึ่งเป็นโปรแกรมจัดการบีบอัดข้อมูลในรูปของไฟล์สกุล gz เพื่อใช้ในการขยายไฟล์ที่ใช้ในการติดตั้งโปรแกรมต่างๆ ที่ถูกบีบอัดเอาไว้มาใช้ โดยมีขั้นตอนดังนี้

- sh configure
- make
- make check
- make install

6.3.3.2.ติดตั้งดีบีไอ โดยมีขั้นตอนดังนี้

- perl Makefile.Pl
- make
- make test
- make install

6.3.3.3.ติดตั้งดีบีดี โดยมีขั้นตอนการติดตั้งดังนี้

- perl Makefile.PL
- make
- make test
- make install

6.4. ขั้นตอนการเขียนโปรแกรมเพื่อทำการค้นหาข้อมูลผ่านฐานข้อมูลผ่านทางเว็ปบราวเซอร์

Description of Files:

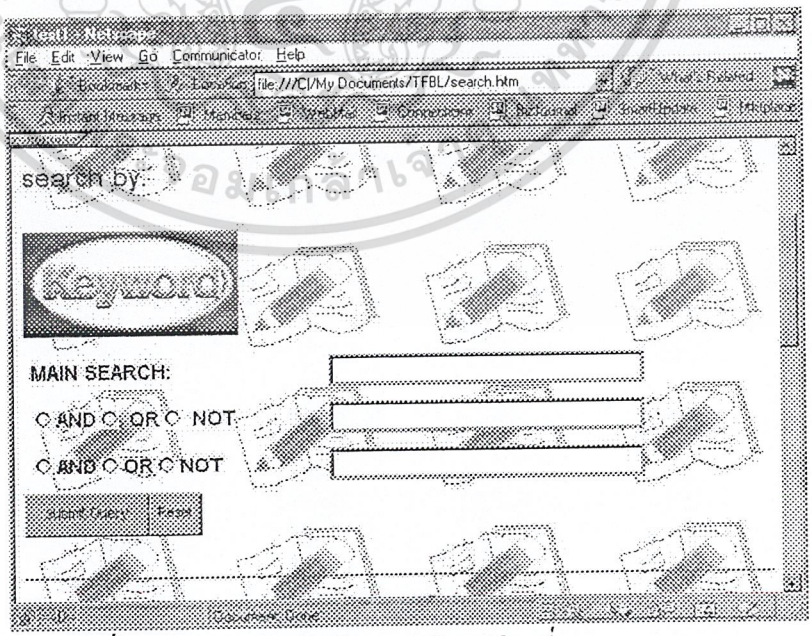
NAME	DESCRIPTION
1.search.html	file ของ เอชทีเอ็มแอล ที่เป็นตัวแสดง เว็บ ของการสืบค้นข้อมูลเริ่มแรก
2.cgi.pl	เป็น Perl script ที่ถูกเรียกใช้โดย search.html แล้วทำการแปลงข้อมูลที่รับเข้ามาจาก ไคล์เอนท์ จากนั้นเก็บข้อมูลที่ต้องการค้นหาไว้ในตัวแปร แล้วทำการติดต่อไปยัง Database โดยผ่าน DBI:oraperl.ข้อมูลที่ค้นได้จากฐานข้อมูลก็จะถูกส่งกลับไปที่ เว็บเพจ เริ่มแรก

- การเก็บไฟล์ข้อมูลของเอกสารเอชทีเอ็มแอลให้เก็บอยู่ในพาร์ธ
  - Path of HTML files: /opt/home/www/virtua.1/english
- การเก็บไฟล์ข้อมูลของสคริปต์เพิร์ลให้เก็บอยู่ในพาร์ธ
  - Path of CGI files: /opt/home/www/virtua.1/code

6.5. ผลการทำงาน

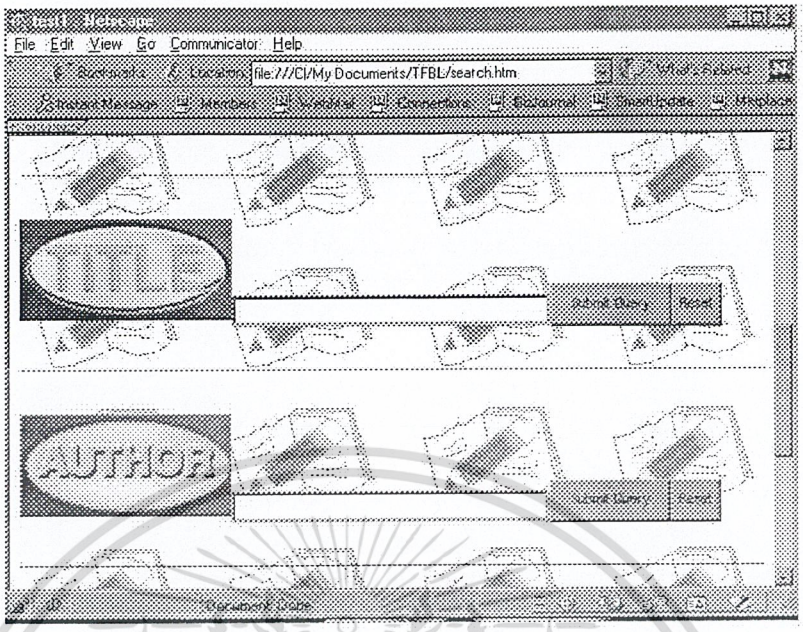
ส่วนแสดงหน้าจอของบราวซ์เซอร์

- หน้าจอส่วนที่ 1 แสดงดังในรูป 6.4



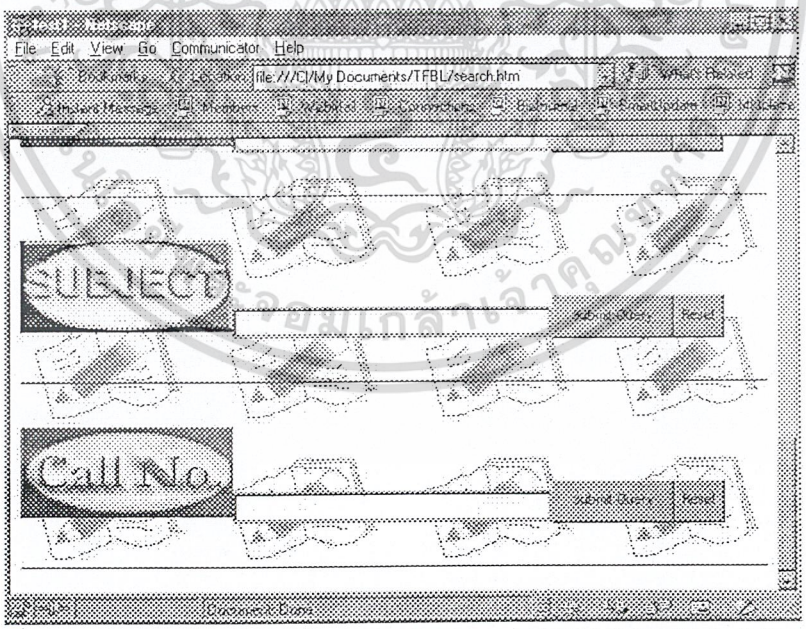
รูปที่ 6.4 แสดงผลลัพธ์หน้าจอการค้นหาส่วนที่ 1

- หน้าจอส่วนที่ 2 แสดงดังในรูปที่ 6.5



รูปที่ 6.5 แสดงผลลัพธ์หน้าจอการค้นหาส่วนที่ 2

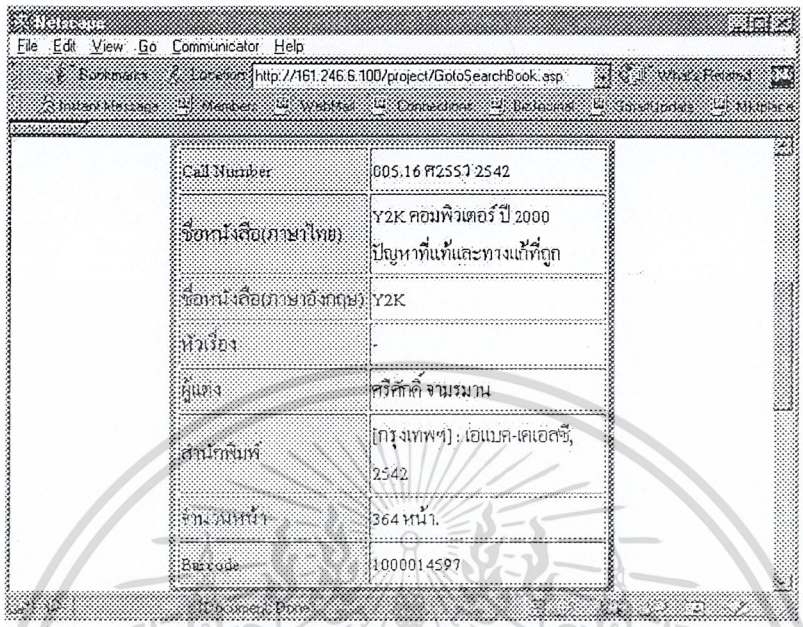
- หน้าจอส่วนที่ 3 แสดงดังในรูปที่ 6.6



รูปที่ 6.6 แสดงผลลัพธ์หน้าจอการค้นหาส่วนที่ 3

- หน้าจอแสดงผลลัพธ์

แสดงดังในรูปที่ 6.7



รูปที่ 6.7 แสดงหน้าจอแสดงผลลัพธ์ที่ได้จากการสืบค้น

บทที่ 7
บทสรุปและวิจารณ์

7.1 บทสรุป

จากการดำเนินงานของโครงงานนี้สามารถดำเนินการตามวัตถุประสงค์ที่ ปัญหาส่วนใหญ่ที่เกิดขึ้นกับโครงงานนี้ เป็นปัญหาทางด้านการติดต่อกับตัวแทนจำหน่ายของผลิตภัณฑ์ ซึ่งต้องประวิงเวลาในการรอบางไฟล์ที่เราต้องทำการขอไปที่ฝ่ายตัวแทนจำหน่ายของผลิตภัณฑ์นั้น และยังมีปัญหาทางด้านการติดตั้งดีบีดี (DBD)ที่ต้องนำมาใช้ให้ถูกต้องกับแพลทฟอร์มของเซิฟเวอร์แต่ละแบบโดย ณ ขณะนี้มีดีบีดีที่ให้การสนับสนุนกับแต่ละแพลทฟอร์มดังนี้

- DBD::Adabase
- DBD::Altera
- DBD::CSV
- DBD::DB2
- DBD::Fulcrum
- DBD::Illusta
- DBD::Informix
- DBD::Ingres
- DBD::ODBC
- DBD::Oracle
- DBD::Pg
- DBD::Solid
- DBD::Sybase
- DBD::Xbase
- DBD::pNET

ซึ่งสามารถดาวน์โหลดได้จากหลายๆเว็บไซท์เช่น

http://www.areana.co.uk/technologia/DBI/doc/tpj5/index.html

หรือจะเข้าไปดูแหล่งข้อมูลข่าวสารของเทคโนโลยีของภาษาเพิลที่ให้การสนับสนุน เกี่ยวกับการทำการเชื่อมต่อเว็บเพจกับฐานข้อมูล จาก

http://www.perl.com/

รวมทั้งแหล่งรวมมอดูลที่ให้การสนับสนุนบนแพลทฟอร์มของบริษัทซัน ได้จาก

http://www.sunfreeware.com/

โครงงานนี้สามารถทำการเพิ่มเติม ระบบยืม-คืน และ การแสดงผลของรูปภาพของหนังสือที่ได้ทำการสแกนเก็บไว้ในอีกฐานข้อมูลหนึ่ง รวมทั้งกาปรับแต่งทางด้านกราฟฟิกของหน้าจอแสดงผลลัพธ์ของโปรแกรมลงในสคริปต์นี้ได้เลยในส่วนของซีจีไอสคริปต์

7.2 บทวิจารณ์

การพัฒนาแอพพลิเคชั่น เว็บดาต้าเบส(Web Database) มีทางเลือกในการพัฒนาได้หลายวิถีทาง ขึ้นอยู่กับขอบเขตของลักษณะโครงงานที่เราจะทำ หรือถูกกำหนดให้ทำตามขอบเขตที่กำหนด เช่น

ถ้าเรามีตัวฐานข้อมูลเป็น Access และเซิร์ฟเวอร์ใช้ระบบปฏิบัติการ วินโดว์เอ็นที(Windows NT) เราสามารถเลือกพัฒนาโดยใช้ ASP(Active Server Page)

หากเราใช้ระบบปฏิบัติการอื่นๆ ก่อนที่จะเลือกพัฒนาด้วยวิธีใดนั้นควรจะพิจารณาถึงการเข้ากันได้กับขอบเขตของรีซอสต่างๆที่เรามีอยู่ รวมทั้งแนวทางที่จะพัฒนาต่อไปในอนาคต เทคโนโลยีใหม่ที่ถูกพัฒนาขึ้นมาใช้มีทั้งข้อดีและข้อเสีย หากเราสามารถเลือกพัฒนาด้วยวิถีทางที่สามารถหาที่ปรึกษาได้ง่ายก็จะเป็นการดีเพื่อการประหยัดเวลาในการแก้ปัญหา

ภาคผนวก

❑ Source Code ของเอกสาร HTML

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=x-user-defined">
  <meta name="Author" content="oiy">
  <meta name="GENERATOR" content="Mozilla/4.5 [en] (Win95; I) [Netscape]">
  <title>test1</title>
</head>
<body background="bground.gif">
<center><img SRC="logo.gif" height=100 width=200></center>
<hr NOSHADE WIDTH="100%">
<br><b><font face="Garamond"><font color="#990000"><font size=+3>   
Please type words that you want to search followed by topic in the box
below each option.ONLY!! One topic you can search by.</font></font></font></b>
<br><form METHOD=GET ACTION="/cgi-bin/cgi.pl"><input type=hidden name=type value="keyword">
<img SRC="key.GIF" ALT="KeyWord.gif" height=81 width=165>
<table COLS=2 WIDTH="90%" >
<tr>
<td WIDTH="200"><b><font color="#660000"><font size=+2>MAIN SEARCH:</font></font></b></td>
<td><input TYPE="TEXT" NAME="text1" VALUE=" " SIZE=30></td>
</tr>

<tr>
<td><input TYPE=Radio NAME="choice1" VALUE="AND"><b><font color="#660000"><font size=+2>AND
<input TYPE=Radio Name="choice1" Value="OR">OR
<input TYPE=Radio Name="choice1" Value="NOT"> NOT</font></font></b></td>
<td><input TYPE=TEXT Name="text2" VALUE" " SIZE="30"></td>
</tr>

<tr>
<td><input TYPE=Radio Name="choice2" value="AND"><b><font color="#660000"><font size=+2>AND
<input TYPE=RADIO NAME="choice2" VALUE="OR">OR
```

```
<input TYPE=RADIO NAME="choice2" VALUE"NOT">NOT</font></font></b></td>
<td><input TYPE=TEXT NAME="text3" VALUE=" " SIZE="30"></td>
</tr>
</table>
<input TYPE=SUBMIT NAME="SEARCH"><input TYPE=RESET NAME="CLEAR" >
</form>
<hr SIZE=1 WIDTH="100%">

<p><form METHOD=GET Action="/cgi-bin/cgi.pl"><input TYPE=hidden NAME="type" VALUE="TITLE">
<img SRC="title.gif" height=78 width=164><input TYPE="TEXT" NAME="text1" VALUE=" " SIZE=30>
<input TYPE=SUBMIT NAME="SEARCH"><input TYPE=RESET NAME="CLEAR" ></form>
<hr SIZE=1 WIDTH="100%">


<p><form METHOD=GET Action="/cgi-bin/cgi.pl">
<input TYPE=hidden NAME="type" VALUE="AUTHER"><img SRC="auth.gif" BORDER=0 height=79
width=163><input TYPE="TEXT" NAME="text1" VALUE=" " SIZE=30><input TYPE=SUBMIT
NAME="SEARCH"><input TYPE=RESET NAME="CLEAR" ></form>
<hr SIZE=1 WIDTH="100%">

<p><form METHOD=GET Action="/cgi-bin/cgi.pl">
<input TYPE=hidden NAME="type" VALUE="SUBJECT"><img SRC="sub.gif" height=71 width=163><input
TYPE="TEXT" NAME="text1" VALUE=" " SIZE=30><input TYPE=SUBMIT NAME="SEARCH"><input
TYPE=RESET NAME="CLEAR" ></form>
<hr SIZE=1 WIDTH="100%">

<p><form METHOD=GET Action="/cgi-bin/cgi.pl">
<input TYPE=hidden NAME="type" VALUE="CALL_NUM"><img SRC="call_num.gif" height=70
width=163><input TYPE="TEXT" NAME="text1" VALUE=" " SIZE=30><input TYPE=SUBMIT
NAME="SEARCH"><input TYPE=RESET NAME="CLEAR" ></form>
<hr SIZE=1 WIDTH="100%">
</body>
</html>
#----------
```

Variable Description:(in HTML file)

NAME	DESCRIPTION
1).Hidden Input: Name:"type"	กลุ่มตัวเลือกของหัวข้อของข้อมูลที่ต้องการสืบค้น ประกอบด้วย
▪ AUTHER	สืบค้นตามชื่อผู้แต่ง
▪ TITLE	สืบค้นตามชื่อเรื่อง
▪ KEYWORD	สืบค้นตามคำ
▪ CALL_NUM	สืบค้นตามหมายเลขเรียกหนังสือ
▪ SUBJECT	สืบค้นตามชื่อหัวข้อเรื่อง
2).INPUT Type:Text Name:"text1"	เก็บคำที่ต้องการค้นหาลำดับแรก
3).INPUT Type:Text Name:"text2"	เก็บคำที่ต้องการค้นหาลำดับที่สอง
4).INPUT Type:Text Name:"text3"	เก็บคำที่ต้องการค้นหาลำดับที่สาม
5).Radio button Name:"choice1"	เก็บ option ของการ combine ทาง logical
6). Radio button Name:"choice2"	ก็บ option ของการ combine ทาง logical
7). INPUT Type:Text Name:"text3"	เก็บคำที่ต้องการค้นหาลำดับที่สาม
8). INPUT Type=hidden name="type" value=" "	เป็นการส่งข้อมูลโดยไม่ต้องการแสดงให้ผู้ใช้เห็น โดยในตัวอย่างนี้จะเป็นการส่งค่าตัวเลือกที่จะทำการสืบค้นตามแต่ละหัวข้อ

ขั้นตอนการใช้งาน

หลังจากโหลด เว็บเพจ หรือ เอกสารเอชทีเอ็มแอลมาจากเซิร์ฟเวอร์เรียบร้อยแล้ว ผู้ใช้งานจะได้รับแบบฟอร์มที่ใช้ในการสืบค้นข้อมูลตามแต่ละหัวข้อที่ต้องการค้นหา แต่ในการใช้งานให้เลือกที่จะค้นหาเพียงแค่หัวข้อเดียวเท่านั้นเช่น จะสืบค้นตามหัวข้อ "ชื่อผู้แต่ง" ก็ให้พิมพ์คำที่ต้องการค้นหาภายในช่องว่างที่อยู่ด้านล่างของหัวข้อ ชื่อผู้แต่ง

เฉพาะหัวข้อการสืบค้นตามคำสำคัญจะมีลักษณะพิเศษคือ จะมีช่องกรอกข้อความมาให้ 3 ช่อง และตัวเลือก radio button มาให้ 2 ชุดเช่น ต้องการสืบค้นคำว่า "Perl" และ "Programming" ให้ทำการพิมพ์คำว่า Perl ในช่องกรอกข้อความที่ 1 หรือ Main Search แล้วคลิก radio button ที่ตัวเลือก AND จากนั้นก็พิมพ์คำว่า Programming ลงในช่องกรอกข้อความที่ 2 เสร็จแล้วคลิกปุ่ม Submit Query

- **Source Code Of Perl script**

```
1:      #!/usr/local/bin/perl
2:      sub form_method{
        $method=$ENV{'REQUEST_METHOD'};
        }
        if(&form_method eq 'POST'){
        read(STDIN,$temp,$ENV{'CONTENT_LENGTH'});
        }
        else{
        $temp=$ENV{'QUERY_STRING'};
        }
3:      @pairs=split(/&/,$temp);
4:      foreach $item(@pairs){
5:      ($key,$content)=split(/=/,$item,2);
6:      $content=~tr/+/ /;
7:      $content=~ s/%(..)/pack("c",hex($1))/ge;
8:      $fields{$key}=$content;
        }

        ##########Assign value to variable#################

9:      $topic=$fields{'type'};
10:     $word1=$fields{'text1'};
11:     $word2=$fields{'text2'};
12:     $word3=$fields{'text3'};
13:     $option1=$fields{'choice1'};
14:     $option2=$fields{'choice2'};

        ######### Check null value of each Option of the radio button ##########

15:     if((!defined $option1)&&(!defined $option2)){
            $statement="select *  from book where $topic like '%$word1%'";
        }
```

```
16:   if(defined $option1){
        $statement="select *  from book where $topic like '%$word1%' $option1 $topic like
      '%$word2%'";
      }
17:   if(defined $option2){
       $statement="select *  from book where $topic like '%$word1%' $option1 $topic like '%$word2%'
      $option2 $topic like '%$word3%'";
      }

      ############Database Connection Section#################
18:   $ENV{ORACLE_HOME} = '/usr/oracle/app/oracle/product/7.3.4';
19:   $ENV{ORACLE_SID} = 'SUNT';

20:   $dbname = $ARGV[0] || '';      # if '' it'll use TWO_TASK/ORACLE_SID
21:   $dbuser = $ENV{ORACLE_USERID} || 'scott/tiger';

22:   use Oraperl;
      #load Module Oraperl

      ######## login to database ######################
23:   $dba= &ora_login($dbname,$dbuser,'');
24:   $csr= &ora_open($dba,$statement);

      #########Print result to WEB#########################
25:   print"Content-type: text/html\n\n";
      print"<body bgcolor=\"#ffffff\">\n";
      print"<h1>This is your result</h1>\n";
      print"Your choice is: $topic <br>\n";
      print"Your word that want to search: $word1<br>\n";
26:   while(@result=&ora_fetch($csr)){
      @barcodeid=$result[5];
      @barid=splice(@barcodeid,0,2);
      $check=shift @barid;
      if($check==0)
```

```
{ unshift(@barid,B); }
elsif($check==1)
{ unshift(@barid,J); }
elsif($check==2)
{ unshift(@barid,V); }
elsif($check==3)
{ unshift(@barid,T); }
else($check==4)
{ unshift(@barid,C); }
print"<table>\n";
print"<tr>\n";
print"<td>$result[0] </td>\n";
print"<td>$result[1] </td>\n";
print"<td>$result[2] </td>\n";
print"<td>$result[3] </td>\n";
print"<td>$result[4] </td>\n";
print"<td><a href="http://1.2.27.22/image/@barid.gif">Image</a></td>\n";
print"</tr>\n";
print"</table>\n";
}
27:  &ora_close($csr);
28:  &ora_logoff($dba);
```

#----------

ขั้นตอนการทำงาน

1: เป็นคำสั่งที่สั่งให้ไปเรียกตัวแปรภาษาของภาษาเพิร์ลตามเส้นทางที่ระบุ

2: ประกาศสับรูทีน form_method ซึ่งทำหน้าที่ในการพิจารณาว่าเมธอดใน form tag มีการส่งค่ามาเป็นแบบ

ใดโดยพิจารณาจากตัวแปร SENV{'REQUEST_METHOD'}

ถ้า method=POST : จะให้ทำการอ่านอินพุทจากตัวแปร SENV{'CONTENT_LENGTH'} แล้วไปเก็บไว้

ในตัวแปร $temp

ถ้า method=GET : จะทำการอ่านอินพุทจากตัวแปร $ENV{'QUERY_STRING'} ไปเก็บในตัวแปร $temp

3: เป็นการตัดแบ่งข้อมูลที่เชื่อมกันอยู่ด้วยเครื่องหมาย & ของแต่ละคู่ข้อมูลที่ส่งมาจากแบบฟอร์ม โดยการ

เรียกใช้ฟังก์ชั่น split จากนั้นก็เก็บข้อมูลที่ถูกแบ่งแยกแล้วลงในตัวแปรอาร์เรย์ @pairs

4: foreach $item(@pairs) เป็นการพิจารณาค่าของข้อมูลในแต่ละตัวแปรอาร์เรย์ @pairs มาเก็บในตัวแปร

สเกล่าร์ $item

5: เป็นคำสั่งที่ให้ทำบนแต่ละข้อมูลของตัวแปร $item

($key,$content)=split(/=/,$item,2);

ทำการตัดแบ่งข้อมูลโดยใช้เครื่องหมายเท่ากับเป็นตัวตัดแบ่งออกจากตัวแปร $item ไปเก็บลงในตัวแปร

$key และ $content

6: ในข้อมูลของตัวแปร $content ให้ทำการแปลงเครื่องหมาย + ให้เป็นช่องว่าง

7: ในข้อมูลของตัวแปร $content ให้ทำการแปลงอักขระที่อยู่ในรูปเลขฐานสิบหกเป็นอักขระปกติ

8: เก็บข้อมูลลงในตัวแปร Assoiate array %fields โดยมีคีย์ตาม $key และมีวารู(value)ตรงตาม $content

จบการทำงานตามคำสั่ง foreach

15: if((!defined $option1)&&(!defined $option2))

เป็นการตรวจสอบว่ามีการส่งข้อมูลจากการคลิก radio button มาหรือไม่ถ้าไม่มีมาจากทั้งสองชุดข้อมูล

ของ radio button(option1 และ option2)ให้ทำตามคำสั่งในบล็อกที่อยู่ถัดมา

18: $ENV{ORACLE_HOME} เก็บพาร์ธที่จะอ้างไปถึงที่เก็บข้อมูลของฐานข้อมูล

19: $ENV{ORACLE_SID} = 'SUNT' เป็นการกำหนดค่า SID ซึ่งค่านี้จะถูกเซตไว้ใน Oracle Database

21: $dbuser = $ENV{ORACLE_USERID} || 'scott/tiger'; กำหนดค่า login name และ password สำหรับเข้า
ถึงในฐานข้อมูล

22: ทำการโหลดโมดูล Oraperl มาใช้งานภายในสคริปต์

23: ทำการล็อกอินเข้าสู่ Oracle โดยเรียกใช้ฟังก์ชัน &ora_login

24: เปิด cursor ของฐานข้อมูลตาม SQL Statement ที่กำหนด

25: ส่งข้อมูลกลับมาให้ฝั่ง ไคล์เอนท์ โดยทุกครั้งต้องเริ่มต้นส่วนของข้อมูลที่จะส่งกลับด้วย การส่งข้อมูล Content-type: [รูปแบบของข้อมูลที่ส่งกลับตามมาตรฐาน MIME type] แล้วตามด้วย newline 2 ครั้งติดกัน

26: while(@result=&ora_fetch($csr))
ในแต่ละ cursor ที่เปิดโดย $csr ทำการเรียกใช้ฟังก์ชัน &ora_fetch เพื่อดึงข้อมูลออกมาเก็บไว้ในตัวแปร
อาร์เรย์ @result และพิมพ์ข้อมูลที่ได้ส่งมาให้ที่ฝั่ง ไคล์เอนท์ ในรูปแบบตาราง

27: ทำการปิด cursor

28: ปิดการเชื่อมต่อกับฐานข้อมูล