บทที่ ๒

เอกสารประกอบการสอนรายวิชา

204204 การออกแบบและพัฒนาฐานข้อมูล

ภาคการศึกษาที่ 2 ปีการศึกษา 2551

สถิตย์โชค โพธิ์สอาด

สาขาวิชาเทคโนโลยีสารสนเทศ

สำนักวิชาเทคโนโลยีสังคม

มหาวิทยาลัยเทคโนโลยีสุรนารี

กันยายน 2551

บทที่ 7 SQL: ภาษานิยามข้อมูล (Data Definition Language)

วัตถุประสงค์

- สามารถอธิบายความหมายของ SQL ได้
- สามารถอธิบายวิวัฒนาการของ SQL ได้
- สามารถอธิบายการชนิดข้อมูลมาตรฐานที่ SQL รองรับ
- สามารถอธิบายวัตถุประสงค์ของการกำหนดเงื่อนไขบังคับบูรณภาพด้วย SQL ได้
- สามารถสร้าง แก้ไข และลบตารางข้อมูลด้วย SQL ได้
- สามารถสร้างเงื่อนไขบังคับบูรณภาพด้วย SQL ได้
- สามารถสร้าง แก้ไข และลบดัชนีได้
- สามารถสร้าง แก้ไข และลบวิวด้วย SQL ได้

คำสำคัญ: SQL (structured query language); ภาษานิยามข้อมูล (data definition language); CREATE DATABASE; CREATE TABLE; ALTER TABLE; DROP TABLE; CREATE INDEX; DROP INDEX; CREATE VIEW; DROP VIEW

7.1 บทนำ

ในบทนี้เป็นหัวใจของการพัฒนาระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เป็นการอธิบายถึงการใช้เครื่องมือหลักในการจัดการระบบจัดการฐานข้อมูลเชิงสัมพันธ์ซึ่งได้แก่คำสั่ง SQL SQL เป็นภาษามาตรฐานที่ใช้กับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ในบทนี้อธิบายถึงกลุ่มของคำสั่ง SQL ที่ใช้ในการนิยามข้อมูลหรือ DDL เป็นสำคัญ โดยเริ่มจากการอธิบายถึงวัตถุประสงค์ของ SQL อย่างกระชับและอธิบายวิวัฒนาการของ SQL ในหัวข้อที่ 7.2.1 และ 7.2.2 จากนั้นเป็นการ แนะนำชนิดของข้อมูลมาตรฐานที่ใช้กับ SQL ในหัวข้อ 7.3 จากนั้นเป็นการอธิบายและแสดงตัวอย่างการใช้งาน DDL ในหัวข้อ 7.4 โดยเริ่มจากการสร้างฐานข้อมูลและตารางข้อมูลโดยใช้คำสั่ง CREATE ในหัวข้อที่ 7.4.1 และ 7.4.2 การแก้ไขโครงสร้างของตารางด้วยคำสั่ง ALTER ในหัวข้อ 7.4.3 และการลบฐานข้อมูล/ตารางข้อมูลด้วยคำสั่ง DROP ในหัวข้อที่ 7.4.4

การนิยามโครงสร้างของข้อมูลในระบบจัดการฐานข้อมูลนั้น เราจำเป็นต้องกำหนดข้อกำหนดบังคับเพื่อให้ข้อมูลมีความถูกต้องหรือมีบูรณภาพซึ่งอาศัย SQL อาศัย CONSTRAINTS ในการกำหนดเงื่อนไขบังคับต่างๆ ดังได้อธิบายไว้ในหัวข้อ 7.5 สำหรับหัวข้อที่ 7.6 เป็นการสร้างดรรชีเพื่อกำหนดรูปแบบเชิงกายภาพของฐานข้อมูลเพื่อเพิ่มประสิทธิภาพของการแสดงและจัดการข้อมูล และหัวข้อที่ 7.7 เป็นการอธิบายการสร้างและจัดการวิว

7.2 แนวคิดทั่วไปเกี่ยวกับ SQL

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์อาศัยภาษามาตรฐานสำหรับดำเนินการเกี่ยวกับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เพื่อความสะดวกและประสิทธิภาพในการดำเนินการกับฐานข้อมูล SQL (structured query language) เป็นภาษาที่ได้รับการยอมรับอย่างแพร่หลายมากที่สุดในการดำเนินกับฐานข้อมูลเชิงสัมพันธ์ ซึ่ง SQL นี้อาศัยพื้นฐานของการดำเนินการพีชคณิตเชิงสัมพันธ์และแคลคูลัสเชิงสัมพันธ์ ในการดำเนินการกับฐานข้อมูล SQL ใช้สำหรับแสดงและจัดการข้อมูล SQL แบ่งออกเป็น 2 ประเภทใหญ่ๆ ได้แก่ภาษาที่ใช้ในการนิยามข้อมูล (data definition language—DDL) เป็นกลุ่มของคำสั่ง SQL ที่ใช้ในการนิยมโครงสร้างของฐานข้อมูลต่างๆ เช่นการกำหนดโครงสร้างของฐานข้อมูล การกำหนดโครงสร้างของตารางข้อมูลต่างๆ ในบทนี้อธิบายถึงกลุ่มคำสั่ง DDL นี้ คำสั่ง SQL อีกประเภทหนึ่งได้แก่ภาษาที่ใช้ในการจัดการข้อมูล (data manipulation language—DML) เป็นกลุ่มของคำสั่งที่ใช้ในการเพิ่ม แก้ไข และลบข้อมูลใดๆ ในฐานข้อมูล ซึ่งฐานข้อมูลนั้นๆ ถูกสร้างขึ้นด้วย DDL ก่อนแล้ว ทั้งนี้ยังมีกลุ่มของคำสั่ง SQL อีกกลุ่มหนึ่งซึ่งสามารถแยกออกมาเป็นประเภทเฉพาะจาก DDL ได้แก่คำสั่งควบคุม (control language) ใช้สำหรับสร้างและกำหนดสิทธิของผู้ใช้งานฐานข้อมูล

7.2.1 วัตถุประสงค์

SQL มีวัตถุประสงค์เพื่อใช้ในการแสดง และจัดการข้อมูลในฐานข้อมูลเชิงสัมพันธ์ เพื่อการใช้ฐานข้อมูลเชิงสัมพันธ์อย่างเป็นมาตรฐานและมีประสิทธิภาพสูงสุด SQL เป็นภาษาที่สามารถจัดการฐานข้อมูลเชิงสัมพันธ์ได้อย่างเต็มรูปแบบเป็นไปตามกฎ 12 ข้อของ E.F. Codd

7.2.2 วิวัฒนาการ

SQL ถือกำเนิดขึ้นพร้อมๆ กับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ซึ่งได้แก่การพัฒนาและประยุกต์แนวคิดของแบบจำลองเชิงสัมพันธ์ขึ้นมาเป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ในช่วงทศวรรษที่ 1970 ใน System R ของบริษัท IBM สำหรับ SQL ในยุคแรกๆ ที่ใช้จัดการกับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ System R นั้นมีชื่อเรียกว่า SEQUEL ย่อมาจาก Structured English Query Language สำหรับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ Ingres อีกระบบหนึ่งที่พัฒนาในช่วงไล่เลี่ยกันที่ University of California, Berkeley นั้นใช้ภาษา QUEL แต่ในท้ายที่สุดได้หันมาใช้ SQL เช่นกันในการใช้งานจริง

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่เริ่มใช้ในเชิงพาณิชย์ในปี ค.ศ. 1979 ใช้ SQL เป็นภาษาที่ใช้ในการจัดการฐานข้อมูล เป็นของบริษัท Oracle ซึ่งแต่เดิมใช้ชื่อว่าบริษัทว่า Relational Software, Inc. ต่อมามีการจำหน่ายซอฟต์แวร์สำหรับจัดการฐานข้อมูลเชิงสัมพันธ์โดยใช้ SQL อย่างแพร่หลาย

เนื่องจากมีการนำเสนอผลิตภัณฑ์จากผู้ผลิตระบบจัดการฐานข้อมูลเชิงสัมพันธ์หลายราย จึงจำเป็นต้องมีการกำหนดมาตรฐานของรูปแบบของคำสั่ง SQL เพื่อความเป็นเอกภาพในการใช้งาน โดย American National Standard Institute—ANSI ได้กำหนดมาตรฐานของ SQL ขึ้นเป็นครั้งแรกในปี ค.ศ. 1986 และต่อมากลายเป็นมาตรฐานในระดับนานาชาติโดย International Standard Organization—ISO ในปี ค.ศ. 1987 รูปแบบของคำสั่ง SQL ในปี ค.ศ. 1986 นี้เรียกว่า SQL-86 แม้ว่าจะมีมาตรฐานกำหนดไว้เพื่อให้ผู้ผลิตระบบจัดการฐานข้อมูลแต่ละรายพัฒนาเพื่อความเข้ากันได้ ผู้ผลิตแต่ละรายได้พัฒนาส่วนเพิ่มเติมจาก SQL ในผลิตภัณฑ์ของตนเองเพื่อเพิ่มประสิทธิภาพและความสามารถในการทำงานของระบบจัดการฐานข้อมูลในสูงขึ้น ในปี ค.ศ. 1992 จึงได้มีการกำหนดรูปแบบของคำสั่ง SQL ที่เพิ่มความสามารถในการจัดการฐานข้อมูลเพิ่มขึ้นให้เป็นมาตรฐาน โดยรู้จักกันในชื่อ SQL-92 หรือ SQL2 เป็นการปรับปรุง SQL ครั้งใหญ่ โดย SQL2 ครอบคลุมการจัดการระบบฐานข้อมูลได้ครบถ้วนและสมบูรณ์กว่า SQL-86 อาจกล่าวว่า SQL2 เป็นมาตรฐานที่ต่ำที่สุดที่ระบบจัดการฐานข้อมูลปัจจุบันจะต้องรองรับ

มีการกำหนดมาตรฐานครั้งสำคัญอีกครั้งหนึ่งในปี ค.ศ. 1999 โดย SQL:1999 หรือ SQL3 ที่กำหนดขึ้นใหม่นี้เป็นการปรับปรุงครั้งสำคัญ SQL3 นั้นยังคงรองรับการทำงานหลายๆ อย่างที่ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ในปัจจุบันสามารถทำได้ ซึ่งคุณสมบัติสำคัญๆ ที่สามารถดำเนินการได้ใน SQL3 นี้ได้แก่การเพิ่มความหลากหลายของข้อคำถาม การสร้าง triggers การเขียนคำสั่ง SQL ให้อยู่ในรูปของภาษาเชิงกระบวนการและโครงสร้าง (มีคำสั่งในการเลือกทำและทำซ้ำเหมือนในภาษาคอมพิวเตอร์ระดับสูง เช่น if..else และ for ในภาษา C) และการรองรับความสามารถบางประการของฐานข้อมูลเชิงวัตถุ

นอกจากนี้ยังมีมาตรฐาน SQL:2003, SQL:2006 และ SQL:2008 ที่เพิ่มความสามารถของ SQL ไปตามยุคสมัย เช่นการใช้งาน XML (Extensible Mark-up Language) และการกำหนดมาตรฐานโครงสร้างของคำสั่งเชิงกระบวนการให้ชัดเจนและมีความสามารถมากขึ้น

7.3 ชนิดของข้อมูล

SQL เป็นคำสั่งที่ใช้โดยตรงกับข้อมูล ซึ่งข้อมูลที่ใช้กันโดยทั่วไปสามารถแบ่งออกเป็น 5 ประเภทใหญ่ๆ ได้แก่

1) ตัวอักษร และ ข้อความ
2) ตัวเลขต่างๆ ที่สามารถนำมาใช้ได้ในการคำนวณ เช่น จำนวนนับ เงินตรา หรือจำนวนจริง เป็นต้น
3) เวลา และ วันที่
4) ตรรกะ เช่น จริง/ไม่จริง ถูก/ผิด เปิด/ปิด เป็นต้น
5) วัตถุ (object) อื่นๆ เช่น ไฟล์เอกสารต่างๆ รูปภาพ ไฟล์เสียง เป็นต้น

ประเภทของข้อมูลที่เราใช้งานนั้นระบบจัดการฐานข้อมูลมีความจำเป็นต้องแทนด้วยรูปแบบที่สามารถจัดการได้ด้วยระบบคอมพิวเตอร์ ระบบจัดการฐานข้อมูลของผู้ผลิตแต่ละรายจัดการข้อมูลในลักษณะต่างๆ กัน จึงมีความจำเป็นที่จะต้องกำหนดมาตรฐานของข้อมูลที่ SQL รองรับ โดยชนิดของฐานข้อมูล SQL:1999 หรือ SQL3 เป็นชนิดข้อมูลที่ระบบจัดการฐานข้อมูลส่วนใหญ่รองรับในปัจจุบัน ซึ่งมีรายการดังต่อไปนี้

ชนิดข้อมูล	ขอบเขตของค่าที่รองรับ	คำอธิบาย
BIT	0, 1	ค่าบิต 0 และ 1 เท่านั้น ซึ่งเป็นชุดของบิตเรียงต่อกันได้
BLOB	ชุดของ 0, 1	ค่าบิต 0 และ 1 เท่านั้น ซึ่งเป็นชุดของบิตเรียงต่อกันขนาดใหญ่มาก—Binary Large Object โดยปกติเราใช้เก็บข้อมูลไฟล์ต่างๆ เช่นรูปภาพ ไฟล์เอกสาร ไฟล์วิดีโอ และไฟล์เสียง เป็นต้น
CHAR	ชุดของตัวอักษร ASCII	ตัวอักษรหรือข้อความ เป็นข้อมูลทั่วๆ ไปที่เรามักจัดเก็บในฐานข้อมูล
CLOB	ชุดของตัวอักษรขนาดใหญ่	ตัวอักษรหรือข้อความ ซึ่งตัวอักษรประเภทนี้สามารถแทนรูปแบบของตัวอักษรได้มากกว่ารหัส ASCII รวมถึงภาษาของตัวอักษร และจัดเก็บข้อความขนาดยาวมาก—Character Large Object
VARCHAR*	ชุดของตัวอักษร ASCII มีความยาวแปรผัน	ตัวอักษรหรือข้อความ โดยความยาวของข้อความแปรผันได้ตามข้อมูลจริง แต่จะจัดเก็บค่าได้ยาวที่สุดตามขนาดของคอลัมน์ที่กำหนด
INTEGER	จำนวนเต็มขนาด 32 บิต	จำนวนเต็มขนาด 32 บิตสำหรับแต่ละคอลัมน์
SMALLINT	จำนวนเต็มขนาด 16 บิต	จำนวนเต็มขนาด 16 บิตสำหรับแต่ละคอลัมน์
DECIMAL/ NUMERIC	ตัวเลขทศนิยม	ตัวเลขจำนวนเต็มและทศนิยม
FLOAT	จำนวนจริงขนาดสั้นที่สุด 24 บิต	FLOAT คือจำนวนจริงใดๆ

	และยาวที่สุด 53 บิต	
REAL	จำนวนจริงขนาด 24 บิต	REAL เป็นคำที่ใช้แทน FLOAT(24) ซึ่งเป็นจำนวนจริงขนาดสั้นที่สุด
DOUBLE	จำนวนจริงขนาดยาวที่สุด 53 บิต	DOUBLE เป็นคำที่ใช้แทน FLOAT(53) ซึ่งเป็นจำนวนจริงขนาดยาวที่สุด
DATE	วันที่	ใช้จัดเก็บวันที่ ปีใน SQL จัดเก็บในรูปแบบ YYYY จึงไม่มีปัญหากับปี 2000 และ SQL ต้องสามารถคำนวณปีอธิกสุรทินหรือปีที่มีเดือนกุมภาพันธ์ 29 วันได้อย่างถูกต้อง
TIME	เวลา	เวลาใน SQL บรรจุข้อมูลวินาทีกำกับไว้เสมอรวมทั้งอาจมีส่วนของวินาทีและโซนของเวลา
TIMESTAMP	วันที่ และ เวลา	การนำวันที่และเวลามาเขียนต่อกัน
INTERVAL	ระยะห่างระหว่างวันที่ เวลา หรือ TIMESTAMP 2 ค่า	ระยะห่างระหว่างวันที่ เวลา หรือ TIMESTAMP 2 ค่า แต่จะต้องป้อนค่าตาม syntax ที่กำหนด

*VARCHAR ไม่ได้รองรับด้วย SQL:1999 แต่เป็นชนิดของข้อมูลที่สำคัญและใช้สำหรับสายอักขระส่วนใหญ่ในปัจจุบัน จึงนำมาแสดงไว้ สิ่งสำคัญที่ VARCHAR ต่างจาก CHAR เกี่ยวข้องกับการแสดงผล การแสงดผลของ CHAR นั้น SQL จะไม่ทราบว่าข้อความที่บรรจุอยู่ในคอลัมน์โดยแท้จริงนั้นยาวเท่าใด ผลลัพธ์ที่ได้จาก SQL ในการแสดงผลข้อมู CHAR นั้นจะมีความยาวคงที่ กล่าวคือแม้ข้อความจะสั้นกว่าขนาดของคอลัมน์ที่ได้ประกาศไว้ หลังจากข้อความจะมีช่องว่างเป็นขนาดเท่ากับจำนวนตัวอักษรที่เหลืออยู่ของคอลัมน์ ซึ่งต่างจาก VARCHAR ที่จะแสดงข้อความพอดีกับขนาดของข้อมูลจริง การจัดการข้อมูล CHAR นั้นใช้ทรัพยากรน้อยกว่า VARCHAR ดังนั้นในกรณีที่ข้อมูลมีความยาวที่แน่นอนจึงควรใช้ CHAR

ในทางปฏิบัติแล้วระบบจัดการฐานข้อมูลเชิงสัมพันธ์ของผู้ผลิตแต่ละรายจะรองรับชนิดของข้อมูลต่างกันออกไปอีก ตามแต่จะรายจะเพิ่มความสามารถและประสิทธิภาพของระบบจัดการฐานข้อมูลของตนเอง เช่น อาจมีการรองรับชนิดข้อมูลประเภท CURRENCY สำหรับรองรับข้อมูลทางด้านการเงินซึ่งสามารถจัดเก็บรูปแบบการแสดงผลตลอดจนสกุลเงินได้ หรือการรองรับตัวเลขที่มีขนาดใหญ่มากด้วยชนิดของข้อมูลพิเศษอื่นๆ เป็นต้น

7.4 การสร้างและจัดการสคีมา

การสร้างและจัดการสคีมาเป็นการจัดการเฉพาะโครงสร้างของฐานข้อมูล เช่นการสร้างและการลบฐานข้อมูลด้วยคำสั่ง CREATE DATABASE และ DROP DATABASE ในหัวข้อ 7.4.1 การสร้างและการลบตารางข้อมูลด้วยคำสั่ง CREATE TABLE และ DROP TABLE ในหัวข้อ 7.4.2 และ 7.4.4 ตามลำดับ การสร้างการเปลี่ยนแปลงโครงสร้างตารางด้วยคำสั่ง ALTER TABLE ในหัวข้อ 7.4.3 เป็นคำสั่ง SQL ในกลุ่มนิยามข้อมูล—DDL ในกรณีที่ไม่ได้ระบุเป็นอย่างอื่น คำสั่ง SQL ที่แสดงอ้างอิงรูปแบบของคำสั่ง SQL-92 ซึ่งเป็นมาตรฐานของ DDL และ DML ที่

ใช้กันในปัจจุบัน ตัวอย่างที่จะกล่าวถึงในหัวข้อถัดไป ใช้โครงสร้างของฐานข้อมูล university ซึ่งอ้างอิงถึงบางตารางของสคีมาต่อไปนี้

Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Faculty (*fid:* string, *fname:* string, *sal:* real)

Courses (*cid:* string, *cname:* string, *credits:* integer)

Rooms (*rno:* integer, *address:* string, *capacity:* integer)

Enrolled (*sid:* string, *cid:* string, *grade:* float)

Teaches (*fid:* string, *cid:* string)

Meets_In (*cid:* string, *rno*: integer, *time:* string)

7.4.1 การสร้างฐานข้อมูล

คำสั่งในการสร้างตารางอาจจะต่างกันออกไปสำหรับระบบจัดการฐานข้อมูลของผู้ผลิตแต่ละรายเนื่องจากการจัดการไฟล์ทางกายภาพไม่เหมือนกัน ในที่นี้ใช้รูปแบบอย่างง่ายของระบบจัดการฐานข้อมูล Microsoft SQL Server ซึ่งใช้ในปฏิบัติการ

```
CREATE DATABASE <database_name>
ON PRIMARY
(
        NAME= '<data_name >',
        FILENAME='<data_file_name>'
        [, SIZE=<data_file_size>]
        [, MAXSIZE=<data_file_max_size>]
        [, FILEGROWTH=<data_file_growth_step>]
)
LOG ON
(
        NAME= '<log_name >',
        FILENAME='<log_file_name>'
        [, SIZE=<log_file_size>]
        [, MAXSIZE=<log_file_max_size>]
        [, FILEGROWTH=<log_file_growth_step>]
);
```

CREATE DATABASE	คำสั่งสำหรับสร้างฐานข้อมูล
database_name	ชื่อของฐานข้อมูลซึ่งจะใช้อ้างอิงต่อไปในระบบจัดการฐานข้อมูล
ON PRIMARY	อุปกรณ์จัดเก็บข้อมูล
NAME	ชื่อของวัตถุในฐานข้อมูล ในที่นี้คือชื่อของไฟล์ที่จัดเก็บข้อมูลและไฟล์ log
FILENAME	ชื่อของไฟล์ที่จัดเก็บวัตถุฐานข้อมูลนั้นๆ โดยเป็นชื่อทางกายภาพของไฟล์ที่จัดเก็บในระบบปฏิบัติการ มีนามสกุลเป็น mdf (ไฟล์ข้อมูล) และ ldf (ไฟล์ log)
SIZE	ขนาดเริ่มต้นของไฟล์วัตถุในฐานข้อมูล มีหน่วยเป็น MB
MAXSIZE	ขนาดที่ใหญ่ที่สุดที่อนุญาตให้ไฟล์ของวัตถุในฐานข้อมูลสามารถขยายได้ มีหน่วยเป็น MB
FILEGROWTH	ขนาดของการขยายไฟล์วัตถุในฐานข้อมูลแต่ละครั้ง มีหน่วยเป็น MB
LOG ON	เป็นการระบุการจัดเก็บ log หรือการบันทึกการทำงานของฐานข้อมูล log นี้มีประโยชน์สำหรับการตรวจสอบการทำงานตลอดจนการจัดการธุรกรรมและการกู้คืนข้อมูลซึ่งจะได้อธิบายในบทที่ 9

ตัวอย่าง

สร้างฐานข้อมูล univsersity โดยไฟล์ที่ใช้จัดเก็บข้อมูลชื่อ university.mdf และไฟล์ Log ชื่อ university.ldf

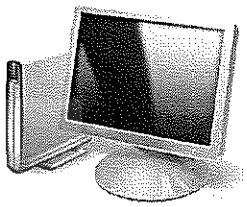
```
CREATE DATABASE university
ON PRIMARY
(
    NAME='university_DAT',
    FILENAME='C:\SQL\university.mdf',
    SIZE=4,
    MAXSIZE=20,
    FILEGROWTH=1
)
LOG ON
(
    NAME= 'university_LOG',
    FILENAME='C:\SQL\university.ldf'
```

```
        SIZE=2,
        MAXSIZE=5,
        FILEGROWTH=1
);
```

สามารถแสดงแผนภาพประกอบคำอธิบายการประมวลผลคำสั่ง SQL ในการสร้างฐานข้อมูลดังนี้

ก่อนการสร้างฐานข้อมูล

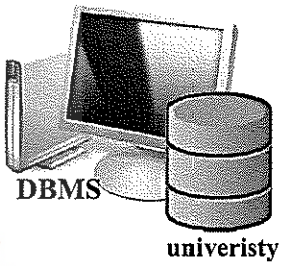
ก่อนการสร้างฐานข้อมูล เครื่องคอมพิวเตอร์ที่ติดตั้งระบบจัดการฐานข้อมูลยังไม่มีฐานข้อมูล



university

สร้างฐานข้อมูล

```
CREATE DATABASE university
ON PRIMARY
(
        NAME='university_DAT',
        FILENAME='C:\SQL\university.mdf',
        SIZE=4,
        ...
);
```



ชื่อไฟล์ข้อมูล: university.mdf

ชื่อไฟล์ Log: university.ldf

จากตัวอย่างเป็นการสร้างฐานข้อมูลชื่อ university โดยไฟล์ที่จะจัดเก็บข้อมูลต่างๆ ของผู้ใช้จะถูกจัดเก็บไว้ในไฟล์ university.mdf ซึ่งเป็นรูปแบบของ Microsoft SQL Server เท่านั้น รวมถึงไฟล์ที่ใช้จัดเก็บบันทึกรายการกิจกรรมของฐานข้อมูลหรือ Log ในชื่อ university.ldf ทั้ง 2 ไฟล์ถูกวางไว้ที่ "C:\SQL" ของเครื่องคอมพิวเตอร์ที่ระบบจัดการฐานข้อมูลควบคุมอยู่ ระบบจัดการฐานข้อมูลจะทำการประมวลคำสั่ง SQL ที่ใช้ในการสร้างฐานข้อมูลจากนั้นสร้างวัตถุที่เกี่ยวข้อง การสร้างฐานข้อมูลเป็นการกำหนดโครงสร้างของฐานข้อมูลเท่านั้น จึงเป็นการนิยามโครงสร้างของข้อมูล ยังไม่ได้มีการสร้างตารางหรือจัดเก็บข้อมูลที่เราใช้ใดๆ ลงในฐานข้อมูล ในภาพตัวอย่างใช้วัตถุทรงกระบอกแทนฐานข้อมูล คำสั่งสร้างฐานข้อมูลแตกต่างกันออกไปตามระบบจัดการฐานข้อมูลและเป็นคำสั่งที่ใช้งานไม่บ่อยนักเนื่องจากระบบสารสนเทศแต่ละระบบมักใช้งานฐานข้อมูลเพียงฐานข้อมูลเดียวหรือมีจำนวนไม่มาก แต่เรียกใช้งานข้อมูลจากหลายๆ ตาราง

การลบฐานข้อมูลสามารถทำได้โดยการเรียกใช้คำสั่ง DROP DATABASE ดังนี้

```
DROP DATABASE <database_name>;
```

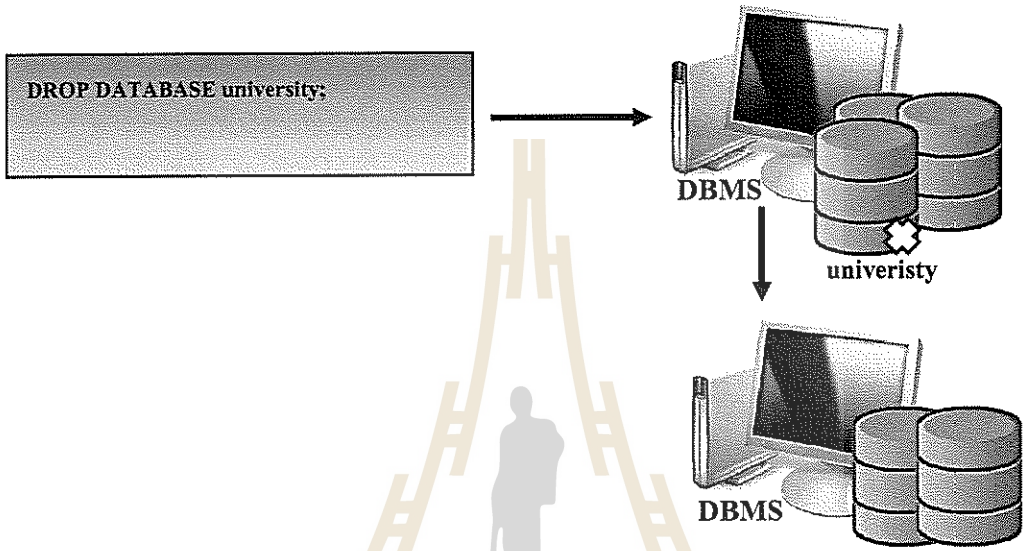
มหาวิทยาลัยเทคโนโลยีสุรนารี

ตัวอย่าง

ลบฐานข้อมูล university

DROP DATABASE university;

สามารถแสดงแผนภาพประกอบคำอธิบายการประมวลผลคำสั่ง SQL ในการลบฐานข้อมูลดังนี้



จากตัวอย่าง ระบบคอมพิวเตอร์อาจมีฐานข้อมูลบรรจุอยู่มากกว่า 1 ฐานข้อมูล ในภาพจำลองนี้จำลองฐานข้อมูลเป็น 3 ฐานข้อมูล ซึ่ง 1 ในนั้นได้แก่ฐานข้อมูล University เมื่อระบบจัดการฐานข้อมูลประมวลผลคำสั่งลบฐานข้อมูลแล้ว ฐานข้อมูล University จะถูกลบทิ้ง คงเหลืออีกเพียง 2 ฐานข้อมูล

7.4.2 การสร้างตาราง

คำสั่งสร้างตารางเป็นคำสั่งสำคัญสำหรับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเอนทิตีและความสัมพันธ์ต้องถูกแทนด้วยตาราง รูปแบบอย่างง่ายของคำสั่งที่ใช้ในการสร้างตารางแสดงได้ดังนี้

```
CREATE TABLE <table_name>
(
        <column_name data_type>
        [,<column_name data_type>]…
);
```

โดย

CREATE TABLE	คือคำสั่งที่ใช้ในการสร้างตารางในฐานข้อมูล
table_name	ชื่อของตารางที่ต้องการสร้าง
column_name	ชื่อของคอลัมน์ในตาราง

data_type ชนิดของข้อมูลที่สัมพันธ์กับคอลัมน์ที่สร้างขึ้นในตาราง

ตารางหนึ่งตารางสามารถมีหลายคอลัมน์ได้โดยการแสดงรายการคอลัมน์และชนิดของคอลัมน์ ขั้นคอลัมน์แต่ละคอลัมน์ด้วยเครื่องหมายจุลภาค "," สามารถเขียนรูปแบบคำสั่งสร้างตารางที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

สร้าง ตาราง <ชื่อตาราง>

(

<ชื่อคอลัมน์ ประเภทข้อมูล>

[,<ชื่อคอลัมน์ ประเภทข้อมูล>]...

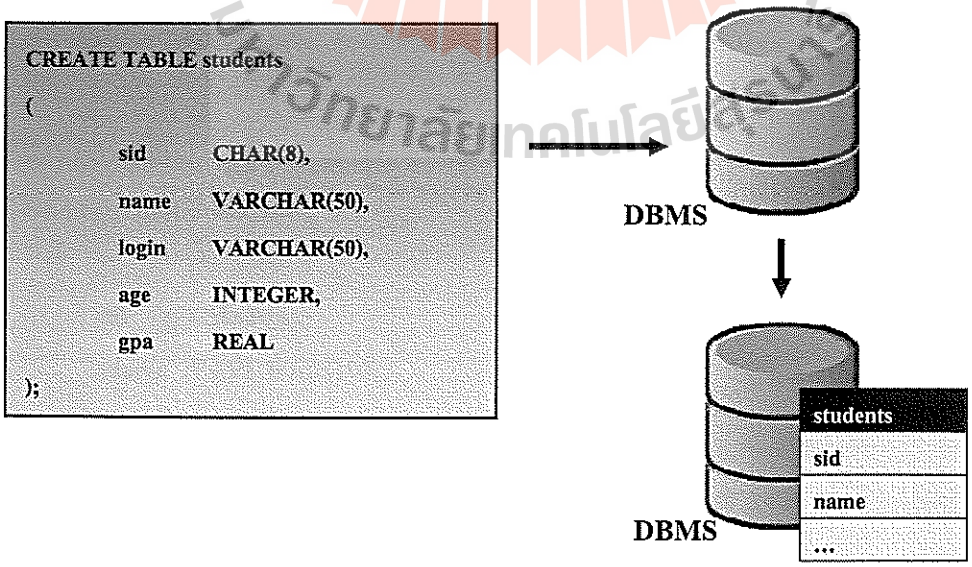
);

ตัวอย่าง

สร้างตารางนักศึกษา ซึ่งมีสคีมาดังนี้ Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

```
CREATE TABLE students
(
        sid     CHAR(8),
        name    VARCHAR(50),
        login   VARCHAR(50),
        age     INTEGER,
        gpa     REAL
);
```

สามารถแสดงแผนภาพประกอบคำอธิบายการประมวลผลคำสั่ง SQL ในการสร้างตารางข้อมูลดังนี้



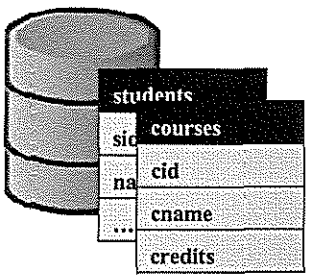
จากตัวอย่างเป็นสร้างตาราง student ประกอบไปด้วยคอลัมน์ 5 คอลัมน์ได้แก่ sid, name, login, age และ gpa ซึ่งคอลัมน์ 3 คอลัมน์แรกนั้นมีชนิดของข้อมูลเป็น string ซึ่งในระบบจัดการฐานข้อมูลโดยทั่วไปใช้ฐานข้อมูล CHAR และ VARCHAR ในการจัดการกับข้อมูลสายอักขระ—string นี้ สำหรับชนิดของข้อมูลที่สามารถกำหนดขนาดของคอลัมน์ได้นั้น สามารถกำหนดไว้ในเครื่องหมายวงเล็บ ( ) หลังชนิดของข้อมูล จากตัวอย่าง sid เป็นชื่อของคอลัมน์แทนรหัสนักศึกษามีชนิดของข้อมูลเป็น CHAR ความยาว 8 ตัวอักษร ซึ่งสามารถจัดเก็บข้อมูลได้ความยาวสูงสุด 8 ตัวอักษร และความยาวมีค่าคงที่เช่น 'B5075666' ในกรณีที่รหัสนักศึกษามีค่าสั้นกว่า 8 ตัวอักษร ระบบจัดการฐานข้อมูลจะแทนค่าตัวอักษรที่ว่างอยู่ด้วยช่องว่าง เช่น ข้อมูล 'B507' ในคอลัมนี CHAR ระบบจัดการฐานข้อมูลจะจัดการและแสดงในรูป "B507 " ซึ่งช่องว่างนั้นมีขนาด 4 ตัวอักษรพอดี สำหรับ name และ login เป็นชื่อของคอลัมน์แทนชื่อนักศึกษาและชื่อสำหรับการเข้าใช้ระบบตามลำดับ login ทั้ง 2 คอลัมน์นี้จัดเก็บข้อมูลชนิดตัวอักษรความยาวแปรผัน VARCHAR ความยาวสูงสุด 50 ตัวอักษร เช่น 'สมชาย' ข้อมูลชนิด VARCHAR นี้ระบบจัดการฐานข้อมูลจะแสดงผลลัพธ์ที่มีความยาวเท่ากับความยาวของข้อมูลจริงที่บรรจุอยู่ในคอลัมน์ไม่ใช่ตามความยาวของขนาดคอลัมน์ดังที่ CHAR แสดง ความยาวสูงสุดของสายอักขระผู้สร้างตารางต้องระบุเองโดยกำหนดจากการวิเคราะห์ให้สอดคล้องกับการใช้งานจริง age และ gpa เป็นข้อมูลชนิดจำนวนเต็มและจำนวนจริง แทนอายุและเกรดเฉลี่ยของนักศึกษาตามลำดับ

จากแผนภาพ ระบบจัดการฐานข้อมูล—DBMS เป็นผู้ประมวลผลคำสั่ง SQL ซึ่งในแผนภาพได้ละ DBMS ไว้แสดงแต่เพียงฐานข้อมูล university แทนด้วยทรงกระบอก ควรระลึกไว้เสมอว่าฐานข้อมูลไม่สามารถเข้าถึงได้โดยตรงแต่ต้องใช้งานด้วยคำสั่ง SQL ผ่านทาง DBMS เสมอเพื่อคงประโยชน์และความปลอดภัยสูงสุดของระบบจัดการฐานข้อมูล เมื่อหน่วยประมวลคำสั่ง SQL และหน่วยจัดการข้อมูลอื่นๆ ใน DBMS ประมวลผลสำเร็จ จะเกิดโครงสร้างของตารางข้อมูลขึ้นในฐานข้อมูลที่ DBMS จัดการอยู่ในปัจจุบัน ซึ่งในที่นี้คือเกิดตาราง students ในฐานข้อมูล university นั่นเอง โดยตาราง students นี้ยังไม่มีข้อมูลใดๆ มีเพียงชื่อของตาราง คอลัมน์ ชนิดของคอลัมน์ซึ่งเปรียบเสมือนมีเพียงโครงสร้างข้อมูลรอการจัดเก็บข้อมูลต่อไป ตัวอย่างต่อไปนี้แสดงการสร้างตารางอื่นๆ เพิ่มเติม

ตัวอย่าง

สร้างตารางรายวิชาจากสคีมา Courses (*cid:* string, *cname:* string, *credits:* integer)

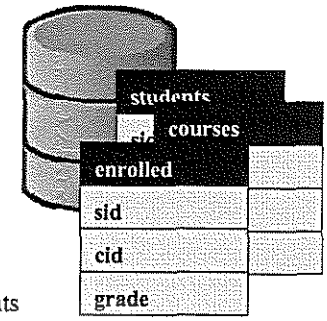
```
CREATE TABLE courses
(
        cid       CHAR(6),
        cname     VARCHAR(100),
        credits   INTEGER
);
```



ตัวอย่าง

สร้างตารางการลงทะเบียนจากสคีมา Enrolled (*sid:* string, *cid:* string, *grade:* float)

```
CREATE TABLE enrolled
(
        sid     CHAR(8),
        cid     CHAR(6),
        grade   FLOAT
);
```



ทั้ง 2 ตัวอย่างหลังเป็นสร้างตารางเพิ่มจากเดิมที่มีอยู่เพียงตาราง students

7.4.3 การแก้ไขตาราง

เมื่อมีการสร้างตารางจากข้อกำหนดของระบบหรือโมเดลเชิงตรรกะของระบบแล้วเป็นเรื่องปกติที่ข้อกำหนดของระบบอาจมีการเปลี่ยนแปลง เช่น ระบบลงทะเบียนต้องสามารถระบุสังกัดของนักศึกษาได้ว่าสังกัดสาขาใด สคีมาของตารางนักศึกษาจึงต้องจัดเก็บรหัสของสาขาวิชาด้วย เป็นต้น หรืออาจมีการปรับเปลี่ยนโครงสร้างของคอลัมน์ด้วยเหตุผลอื่นๆ การแก้ไขโครงสร้างของตารางจึงเป็นเรื่องปกติ คำสั่งที่ใช้ในการแก้ไขโครงสร้างตารางมีรูปแบบอย่างง่ายดังนี้

```
ALTER TABLE <table_name>
<alter_commands> <column_name> [data_type];
```

โดย

ALTER TABLE	คือคำสั่งที่ใช้ในการแก้ไขตาราง
table_name	ชื่อของตารางที่ต้องการแก้ไข จะต้องเป็นตารางที่มีอยู่แล้วในฐานข้อมูล
alter_commands	เป็นคำสั่งในการแก้ไขตาราง ได้แก่ ADD, ALTER และ DROP สำหรับเพิ่ม แก้ไข และลบคอลัมน์หรือวัตถุใดๆ ในตารางข้อมูล
column_name	ชื่อของคอลัมน์ในตารางที่ต้องการแก้ไข
data_type	ชนิดของข้อมูลใหม่ที่สัมพันธ์กับคอลัมน์ที่แก้ไข

สามารถเขียนรูปแบบคำสั่งแก้ไขตารางที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

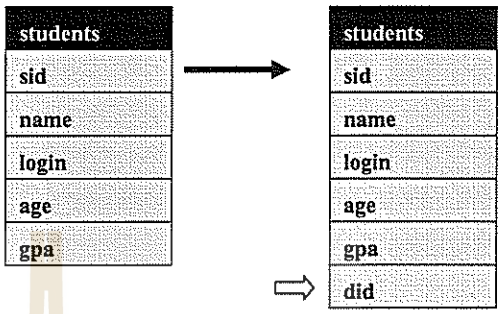
แก้ไข ตาราง <ชื่อตาราง>

<คำสั่งแก้ไขข้อมูล> <ชื่อคอลัมน์> [<ประเภทข้อมูล>];

ตัวอย่าง ADD COLUMN

แก้ไขตารางนักศึกษา โดยการเพิ่มคอลัมน์รหัสสาขา did เป็นข้อมูลชนิดสายอักขระความยาว 3 ตัวอักษร

```
ALTER TABLE students
ADD did CHAR (3);
```

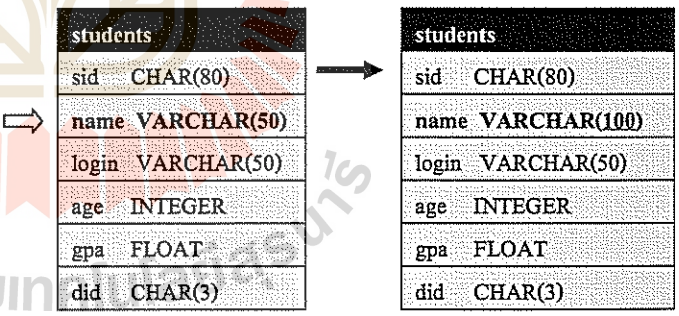


จากตัวอย่าง ส่วนของคำสั่ง **ALTER TALBE students** เป็นการระบุให้ทำการแก้ไขโครงสร้างของตารางนักศึกษา และ **ADD** ใน **(ADD did CHAR(3))** ทำการเพิ่มคอลัมน์ที่ชื่อว่า did เป็นชนิด CHAR ความยาว 3 ตัวอักษร คำสั่ง ADD ยังใช้ในการเพิ่มวัตถุอื่นๆ นอกจากคอลัมน์ โดยจะได้กล่าวถึงในหัวข้อต่อๆ ไป

ตัวอย่าง ALTER COLUMN

แก้ไขตารางนักศึกษา โดยการแก้ไขคอลัมน์ชื่อนักศึกษา name ให้เป็นข้อมูลชนิด VARCHAR ที่รองรับสายอักขระได้ยาวถึง 100 ตัวอักษร จากเดิม 50 ตัวอักษร

```
ALTER TABLE students
ALTER COLUMN name VARCHAR(100);
```

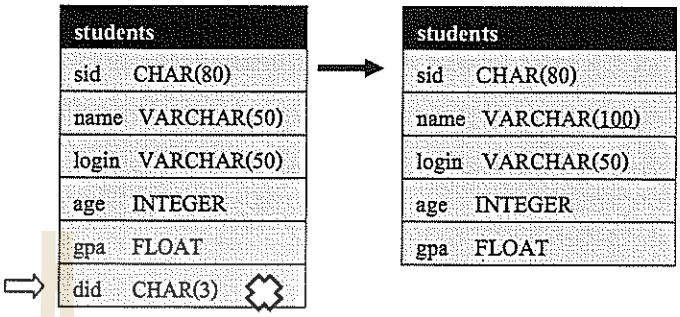


จากตัวอย่าง ส่วนของคำสั่ง **ALTER TALBE students** เป็นการระบุให้ทำการแก้ไขโครงสร้างของตารางนักศึกษา และ **ALTER COLUMN** ใน **(ALTER COLUMN name VARCHAR(100))** ทำการแก้ไขคอลัมน์ที่ชื่อว่า name ให้เป็นชนิด VARCHAR ความยาว 100 ตัวอักษร เช่นเดียวกันกับคำสั่ง ADD คำสั่ง ALTER ยังสมารถใช้ในการแก้ไขวัตถุอื่นๆ นอกจากคอลัมน์ โดยจะได้กล่าวถึงในหัวข้อต่อๆ ไป

ตัวอย่าง DROP COLUMN

แก้ไขตารางนักศึกษา โดยการลบคอลัมน์รหัสสาขา did ทิ้งเสีย

```
ALTER TABLE students
DROP did;
```



จากตัวอย่าง ส่วนของคำสั่ง **ALTER TALBE students** เป็นการระบุให้ทำการแก้ไขโครงสร้างของตารางนักศึกษา และ **DROP** ใน (DROP did) ทำการลบคอลัมน์ที่ชื่อว่า did ทิ้งไป

7.4.4 การลบตาราง

คำสั่งสร้างตารางเป็นคำสั่งสำคัญสำหรับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเอนทิตีและความสัมพันธ์ต้องถูกแทนด้วยตาราง รูปแบบอย่างง่ายของคำสั่งที่ใช้ในการสร้างตารางแสดงได้ดังนี้

DROP <column_name>;

โดย

DROP คือคำสั่งที่ใช้ในการลบตารางออกจากฐานข้อมูล

column_name ชื่อของคอลัมน์ที่ต้องการลบในตาราง

สามารถเขียนรูปแบบคำสั่งลบตารางที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

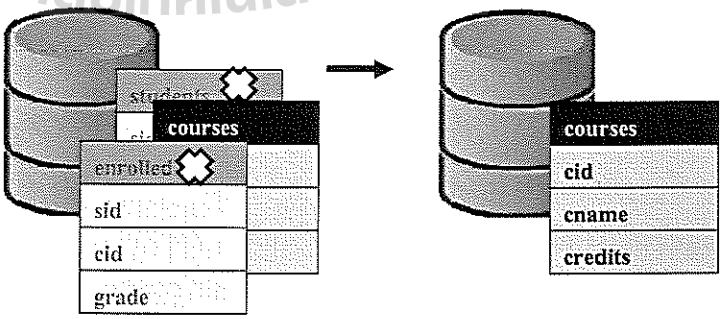
ลบ <ชื่อตาราง>;

ตัวอย่าง

ลบตาราง students และ enrolled

DROP students;

DROP enrolled;



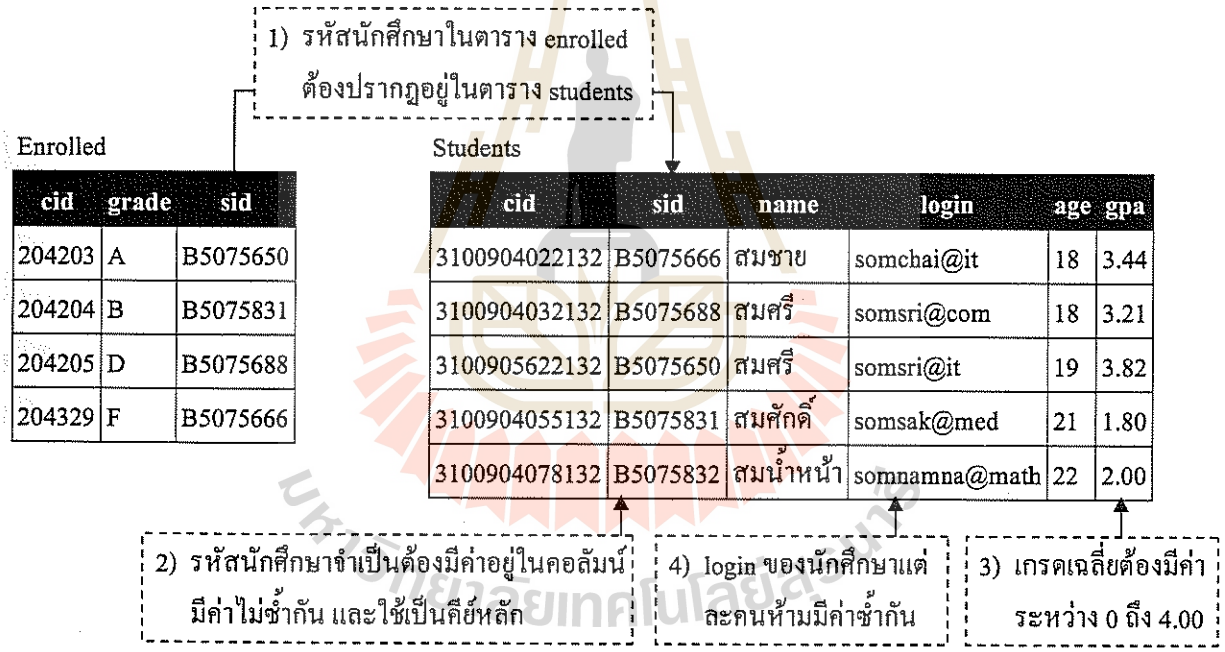
จากตัวอย่างเป็นการลบตาราง students และ enrolled ทีละตาราง คงเหลือเฉพาะตาราง courses ดังภาพ การ DROP ตารางจะทำให้ข้อมูลในตารางหายไปด้วยในกรณีที่มีข้อมูลบรรจุอยู่ในตาราง คำสั่ง DROP สามารถใช้ลบวัตถุอื่นๆ ของฐานข้อมูลได้ เช่น เงื่อนไขบังคับบูรณภาพ—Integrity Constraints ต่างๆ

7.5 บูรณภาพของข้อมูล Data Integrity

บูรณภาพของข้อมูลคือความถูกต้องของข้อมูลทั้งฐานข้อมูลดังได้อธิบายในหัวข้อ 3.4 ในบทที่ 3 และ 4.3 ในบทที่ 4 และกล่าวถึงอีกหลายครั้งเนื่องจากมีความสำคัญและเป็นประโยชน์ที่สำคัญที่ได้รับจากระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ในที่นี้จะนำตัวอย่างในหัวข้อ 4.3 จากบทที่ 4 มาแสดงอีกครั้งโดยย่อ ซึ่งสามารถอ้างอิงได้จากหัวข้อที่เกี่ยวข้องกับบูรณภาพของข้อมูลในบทที่ 3 และ 4 โดยใช้สคีมาและกรณีตัวอย่างข้อมูลของตาราง students และ enrolled ต่อไปนี้ (ตาราง students จากหัวข้อ 4.3 ในบทที่ 4 มี 5 คอลัมน์)

Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Enrolled (*sid:* string, *cid:* string, *grade:* float)



cid	grade	sid
204203	A	B5075650
204204	B	B5075831
204205	D	B5075688
204329	F	B5075666

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

จากตัวอย่าง ข้อมูลที่บรรจุในฐานข้อมูลจะมีความถูกต้องหรือมีบูรณภาพได้จะต้องเป็นไปตามข้อกำหนดของฐานข้อมูลเพื่อคงความถูกต้องของฐานข้อมูล เช่น หมายเลข 1) รหัสนักศึกษาในตาราง enrolled ต้องปรากฏอยู่ในตาราง students การกำหนดดังนี้เพื่อป้องกันความผิดพลาดของการเพิ่มระเบียนในตาราง enrolled หากมีการเพิ่มข้อมูลโดยบรรจุระเบียนการลงทะเบียนเป็นรหัสนักศึกษาที่ไม่ปรากฏอยู่ในตารางนักศึกษา เป็นการเพิ่มข้อมูลที่ไม่ถูกต้อง ทำให้ข้อมูลขาดบูรณภาพ สำหรับหมายเลข 2), 3) และ 4) ต่างก็เป็นข้อกำหนดอื่นๆ เพื่อให้ข้อมูลมีความถูกต้องตรงกับการ

ใช้งาน ข้อกำหนดที่ได้ยกตัวอย่างนี้เป็นเพียงส่วนหนึ่งของข้อกำหนดที่จำเป็นต้องระบุไว้ในตาราง students และ enrolled ดังแสดง

ข้อกำหนดที่ได้ยกตัวอย่างแต่ละข้ออาจแยกออกเป็นข้อกำหนดย่อยๆ ได้ ซึ่งเราจะต้องกำหนดเงื่อนไขในการแก้ไขข้อมูลเพื่อให้ข้อมูลมีความถูกต้องตรงตามข้อกำหนด เงื่อนไขต่างๆ เหล่านี้เรียกว่าเงื่อนไขบังคับบูรณภาพ (integrity constraints) เงื่อนไขบังคับบูรณภาพเหล่านี้จัดอยู่ในประเภทต่างๆ เช่น เงื่อนไขบังคับคีย์ เงื่อนไขบังคับคีย์หลัก เงื่อนไขบังคับการมีส่วนร่วม เงื่อนไขบังคับคีย์ภายนอก เงื่อนไขบังคับทั่วไป เป็นต้น เราสามารถกำหนดเงื่อนไขบังคับเหล่านี้ให้กับโครงสร้างข้อมูลในฐานข้อมูลด้วย SQL ได้ด้วยเงื่อนไขบังคับต่อไปนี้

PRIMARY KEY	เงื่อนไขบังคับคีย์หลัก เป็นการบังคับว่าค่าในคอลัมน์ที่กำหนดนั้นเป็นคีย์หลัก ซึ่งจะต้องมีค่าไม่ซ้ำกัน และไม่เป็นค่าว่างด้วย
NOT NULL	เงื่อนไขบังคับค่าว่าง เป็นการบังคับว่าคอลัมน์นั้นๆ ห้ามเป็นค่าว่าง
DEFAULT	เงื่อนไขบังคับค่าโดยปริยาย เป็นการกำหนดค่าโดยปริยายหากไม่มีการระบุค่าของคอลัมน์นั้นๆ
UNIQUE KEY	เงื่อนไขบังคับคีย์ เป็นการบังคับว่าค่าในคอลัมน์ที่กำหนดต้องมีค่าไม่ซ้ำกัน
FORIGN KEY	เงื่อนไขบังคับเชิงอ้างอิง เป็นการบังคับว่าค่าในคอลัมน์ของตารางหนึ่งจะต้องปรากฏอยู่ในค่าในคอลัมน์ของอีกตารางหนึ่ง
CHECK	เงื่อนไขบังคับตรวจสอบ เป็นการบังคับว่าค่าในคอลัมน์จะต้องเท่ากับค่าหรืออยู่ในช่วงค่าที่กำหนด

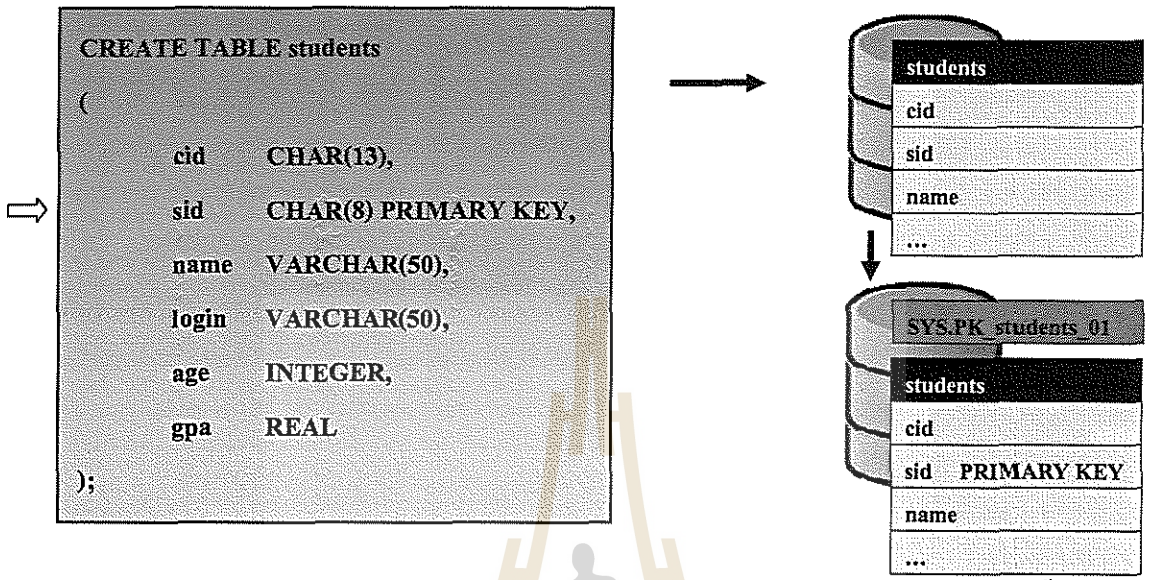
เงื่อนไขบังคับบูรณภาพที่ได้จากการออกแบบฐานข้อมูลในระดับแนวคิดและระดับตรรกะนั้นจะต้องใช้เงื่อนไขบังคับของ SQL ที่เหมาะสมในการประยุกต์กับฐานข้อมูลจริง ในบางกรณีอาจจะต้องใช้หลายเงื่อนไขบังคับประกอบกัน การกำหนดเงื่อนไขบังคับแต่ละเงื่อนไขมีรายละเอียดปลีกย่อยพอสมควร ให้พิจารณารูปแบบการใช้งานจากตัวอย่างต่อไปนี้

ตัวอย่าง PRIMARY KEY CONSTRAINT

การกำหนดเงื่อนไขบังคับคีย์หลัก โดยกำหนดให้คอลัมน์ sid ในตาราง students เป็นคีย์หลัก

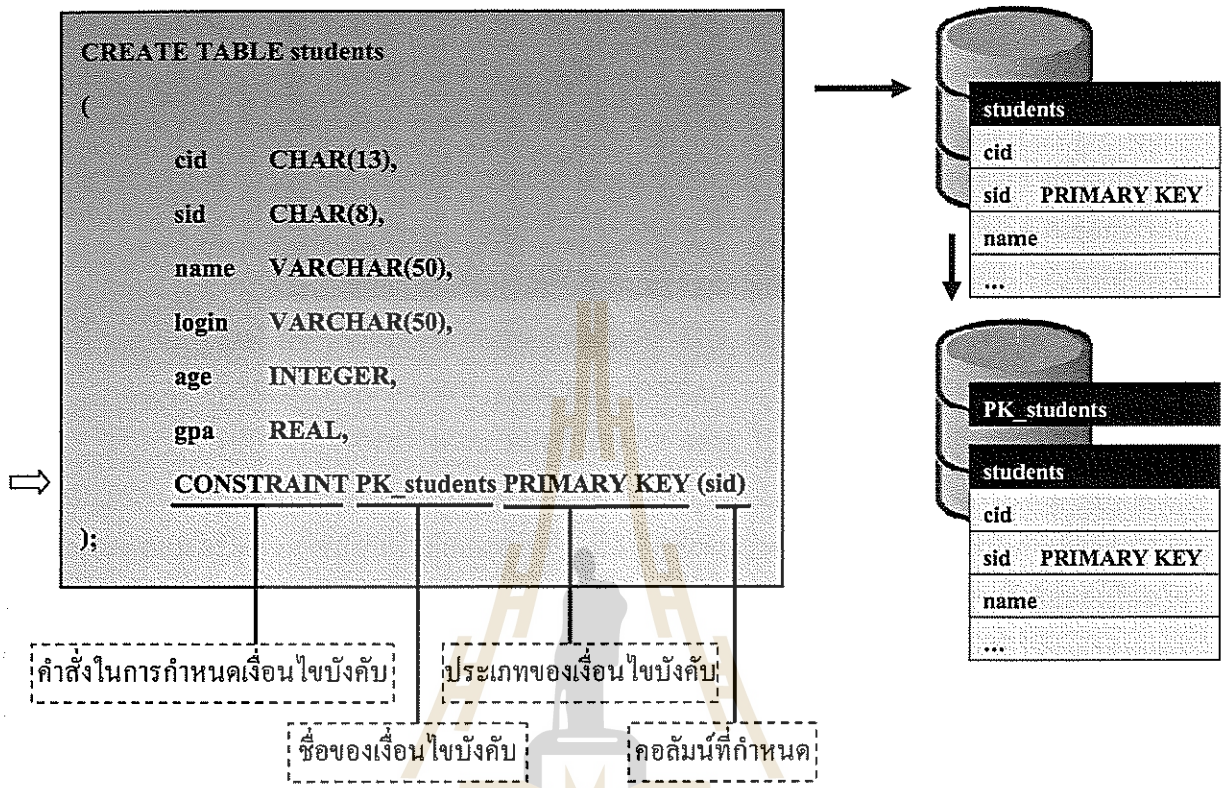
```
CREATE TABLE students       (
        cid     CHAR(13),
        sid     CHAR(8) PRIMARY KEY,
        name    VARCHAR(50),
        login   VARCHAR(50),
        age     INTEGER,
        gpa     REAL
);
```

สามารถแสดงแผนภาพประกอบคำอธิบายการประมวลผลคำสั่ง SQL ในการสร้างเงื่อนไขบังคับดังนี้



จากตัวอย่าง การกำหนดเงื่อนไขบังคับดังกล่าวทำให้คอลัมน์ sid ไม่สามารถเป็นค่าว่างและมีค่าซ้ำกันไม่ได้ ซึ่งตรงกับหมายเลข 1) ในกรณีตัวอย่างเงื่อนไขบังคับบูรณภาพ ** ที่ได้ยกตัวอย่างก่อนหน้า การกำหนดเงื่อนไขบังคับที่แสดงจำเป็นต้องกำหนดพร้อมๆ กับการสร้างตารางข้อมูลโดยคำสั่ง CREATE TABLE โดยวางประเภทของเงื่อนไขบังคับ PRIMARY KEY ไว้หลังจากการกำหนดชนิดข้อมูลของคอลัมน์ที่ระบุให้เป็นคีย์หลักในที่นี้คือ sid CHAR(8) ระบบจัดการฐานข้อมูลจะกำหนดโครงสร้างในฐานข้อมูลให้คอลัมน์ sid เป็นคีย์หลัก ซึ่งต่อไปนี้การเพิ่มระเบียนข้อมูลลงในตาราง students แต่ละระเบียนนั้น ค่าของคอลัมน์ sid แต่ละระเบียนจะเป็นค่าว่างและมีค่าซ้ำกันไม่ได้

เราสามารถกำหนดเงื่อนไขบังคับพร้อมๆ กับการสร้างตารางข้อมูลในอีกรูปแบบหนึ่งดังนี้



จากตัวอย่าง การกำหนดเงื่อนไขบังคับไม่จำเป็นต้องระบุประเภทของเงื่อนไขบังคับไว้ด้านหลังคอลัมน์ที่ต้องการกำหนดเงื่อนไขบังคับก็ได้ มีรูปแบบดังแสดง โดยเพิ่มบรรทัดใหม่หลังจากระบุโครงสร้างคอลัมน์ของตารางข้อมูลเรียบร้อยแล้ว การกำหนดเงื่อนไขบังคับรูปแบบนี้มีข้อดีคือผู้จัดการฐานข้อมูลสามารถจัดการเงื่อนไขบังคับได้สะดวกกว่าในแบบแรกเนื่องจากมีการกำหนดชื่อของเงื่อนไขบังคับนั้นๆ (ระบบจัดการฐานข้อมูลจะกำหนดชื่อของเงื่อนไขบังคับให้ในกรณีที่ไม่ได้กำหนดชื่อด้วยตนเอง) เราสามารถใช้รูปแบบที่คล้ายคลึงกันนี้ในการสร้างเงื่อนไขบังคับประเภทอื่นๆ ได้อีก จึงสามารถพิจารณารูปแบบการใช้งานการสร้างเงื่อนไขบังคับนี้เพื่อประยุกต์ใช้กับการสร้างเงื่อนไขบังคับแบบอื่นๆ โดย

CONSTRAINT คำสั่งในการกำหนดเงื่อนไขบังคับ เป็นคำสั่งที่ต้องคงไว้สำหรับการสร้างเงื่อนไขบังคับทุกๆ ประเภท

PK_students ชื่อของเงื่อนไขบังคับ ในที่นี้คือชื่อของเงื่อนไขบังคับคีย์หลักของตารางนักศึกษา จากแผนภาพด้านบนแสดงให้เห็นว่าโดยแท้จริงแล้วเงื่อนไขบังคับคีย์หลักนี้ถูกสร้างขึ้นเป็นวัตถุหนึ่งในฐานข้อมูล ซึ่งวัตถุนี้จะถูกเรียกใช้งานจากระบบจัดการ

ฐานข้อมูลเพื่อตรวจสอบว่าค่าที่เพิ่มลงในตารางและคอลัมน์ที่เงื่อนไขบังคับนั้นถูกกำหนดขึ้นเป็นไปตามเงื่อนไขหรือไม่

PRIMARY KEY ประเภทของเงื่อนไขบังคับ สามารถแทนด้วย NOT NULL, DEFAULT, UNIQUE KEY, FOREIGN KEY หรือ CHECK ขึ้นอยู่กับประเภทของเงื่อนไขบังคับที่ต้องการสร้าง

sid คอลัมน์ที่กำหนด เป็นคอลัมน์ที่เงื่อนไขบังคับที่สร้างขึ้นนั้นๆ ตรวจสอบค่า สำหรับเงื่อนไขบังคับบางประเภทอาจมีรูปแบบที่ซับซ้อนกว่านี้ ซึ่งนอกการระบุคอลัมน์ที่เงื่อนไขบังคับนั้นตรวจสอบแล้ว ต้องระบุตารางและคอลัมน์อื่นๆ ที่เกี่ยวข้องด้วย เช่น FOREIGN KEY CONSTRAINT หรืออาจต้องระบุเงื่อนไขเพิ่มเติม เช่น CHECK CONSTRAINT เป็นต้น ซึ่งสามารถศึกษาได้จากตัวอย่างการสร้างเงื่อนไขบังคับในตัวอย่างต่อๆ ไป

เนื่องจากการสร้างเงื่อนไขบังคับในลักษณะนี้มีข้อดีที่สำคัญคืออนุญาตให้เราตั้งชื่อเงื่อนไขบังคับเพื่อจัดการในภายหลังได้ ข้อควรคำนึงถึงที่สำคัญประการหนึ่งของการสร้างเงื่อนไขบังคับในลักษณะนี้ ได้แก่การตั้งชื่อของเงื่อนไขบังคับนั้นนิยมตั้งตามมาตรฐานของการตั้งชื่อเงื่อนไขบังคับ เพื่อความสะดวกในการจัดการ โดยเรามักใช้มาตรฐานการตั้งชื่อเงื่อนไขบังคับที่คล้ายคลึงกันดังนี้

PK_TableName

NN_TableName_ColumnName

DF_ TableName _ ColumnName

UK_ TableName _ ColumnName

FK_ReferencedTable_ReferencingTable

CK_TableName

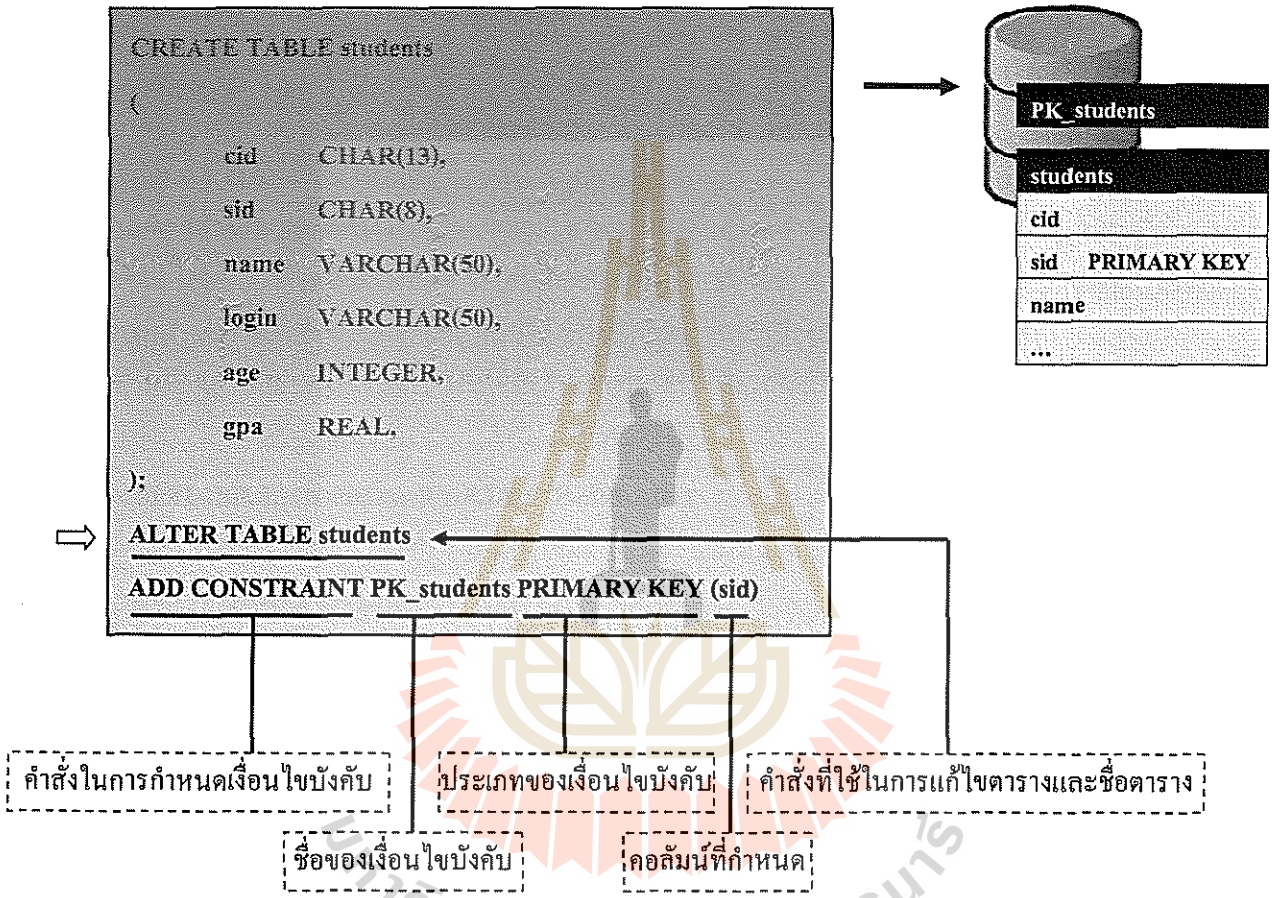
ตัวย่อ 2 ตัวแรกแทนเงื่อนไขบังคับ PRIMARY KEY, NOT NULL, DEFAULT, UNIQUE KEY, FOREIGN KEY และ CHECK ตามลำดับ

TableName และ ColumnName แทนชื่อของตารางและคอลัมน์ที่เงื่อนไขบังคับนั้นตรวจสอบ

สำหรับเงื่อนไขบังคับ FOREIGN KEY นั้น ReferencedTable คือชื่อของตารางหลักและ ReferencingTable คือชื่อของตารางที่อ้างอิงค่าจากตารางหลัก

ตัวย่อ 2 ตัวแรกมักเขียนเป็นตัวใหญ่ ตัวอักษรขึ้นต้นคำสามารถเขียนเป็นตัวใหญ่ได้ แล้วแต่มาตรฐานการตั้งชื่อตัวแปรที่ใช้ อย่างไรก็ตามชื่อของเงื่อนไขบังคับสามารถเขียนเป็นตัวเล็กทั้งหมดก็ได้

อย่างไรก็ตาม ในกรณีที่เรามีความจำเป็นต้องสร้างหรือแก้ไขเงื่อนไขบังคับบูรณภาพใดๆ หลังจากที่ได้สร้างตารางนั้นๆ ไปแล้ว นอกจากนี้ยังอาจมีการเพิ่มข้อมูลในตารางแล้ว ทำให้ไม่มีความสะดวกในการลบตารางนั้นๆ ทิ้งเพื่อสร้างตารางใหม่พร้อมๆ กับเงื่อนไขบังคับบูรณภาพ เงื่อนไขบังคับสามารถสร้างได้ด้วยคำสั่ง ALTER TABLE ดังนี้



จากตัวอย่าง คำสั่งที่ใช้ในการเพิ่มเงื่อนไขบังคับคือคำสั่งตั้งแต่ ALTER TABLE เป็นต้นไปเท่านั้น สำหรับคำสั่ง CREATE TABLE ในด้านบนนำมาแสดงเพื่อให้เห็นโครงสร้างของตารางแต่เพียงอย่างเดียว เราใช้การสร้างเงื่อนไขบังคับในลักษณะนี้กรณีที่ตารางข้อมูลถูกสร้างเรียบร้อยและอาจบรรจุข้อมูลในตารางแล้ว โดย

ALTER TABLE	คำสั่งที่ใช้ในการแก้ไขตาราง จำเป็นต้องระบุไว้เพื่อให้ระบบจัดการฐานข้อมูลแก้ไขตารางที่ระบุ สำหรับทำการเพิ่มเงื่อนไขบังคับที่ผูกติดกับตารางนั้นๆ
students	ชื่อตาราง คือตารางที่ต้องการเพิ่มเงื่อนไขบังคับ ในตัวอย่างคือตารางนักศึกษา
ADD CONSTRAINT	คำสั่งในการกำหนดเงื่อนไขบังคับ เป็นคำสั่งที่ต้องคงไว้สำหรับการสร้างเงื่อนไขบังคับทุกๆ ประเภท

PK_students	ชื่อของเงื่อนไขบังคับ ในที่นี้คือชื่อของเงื่อนไขบังคับคีย์หลักของตารางนักศึกษา
PRIMARY KEY	ประเภทของเงื่อนไขบังคับ สามารถแทนด้วย NOT NULL, DEFAULT, UNIQUE KEY, FOREIGN KEY หรือ CHECK ขึ้นอยู่กับประเภทของเงื่อนไขบังคับที่ต้องการสร้าง
sid	คอลัมน์ที่กำหนด เป็นคอลัมน์ที่เงื่อนไขบังคับที่สร้างขึ้นนั้นๆ ตรวจสอบค่า สำหรับเงื่อนไขบังคับบางประเภทอาจมีรูปแบบที่ซับซ้อนกว่านี้ขึ้นอยู่กับประเภทของเงื่อนไขบังคับที่สร้าง ซึ่งสามารถศึกษาได้จากตัวอย่างการสร้างเงื่อนไขบังคับในตัวอย่างต่อๆ ไป

รูปแบบการสร้างเงื่อนไขบังคับทั้ง 3 รูปแบบได้แก่ 1) การกำหนดไว้หลังคอลัมน์พร้อมๆ กับการสร้างตาราง 2) การกำหนดไว้ท้ายการสร้างตาราง และ 3) การสร้างเงื่อนไขบังคับภายหลังจากการสร้างตาราง เป็นรูปแบบที่สามารถนำมาใช้สร้างเงื่อนไขบังคับประเภทอื่นๆ ได้เช่นเดียวกัน ซึ่งการเลือกใช้รูปแบบใดนั้นขึ้นอยู่กับลักษณะการใช้งานความเหมาะสม และรายละเอียดปลีกย่อยที่อาจไม่ได้กล่าวถึงของการสร้างเงื่อนไขบังคับประเภทต่างๆ

ตัวอย่างต่อไปนี้แสดงการสร้างเงื่อนไขบังคับประเภทต่างๆ โดยเลือกใช้รูปแบบที่เหมาะสม

ตัวอย่าง MULTI-COLUMN PRIMARY KEY CONSTRAINT

การกำหนดเงื่อนไขบังคับคีย์หลัก โดยกำหนดให้คอลัมน์ sid และ cid เป็นคีย์หลัก ในตาราง enrolled

```
CREATE TABLE enrolled
(
        cid     CHAR(13),
        sid     CHAR(8),
        grade   FLOAT
        CONSTRAINT PK_enrolled PRIMARY KEY (cid, sid)
);
```

จากตัวอย่างเราสามารถกำหนดให้คีย์หลักสร้างจากคอลัมน์มากกว่า 1 คอลัมน์ได้ ซึ่งเราเรียกว่า Composite Primary Key โดยการกำหนดรายการคอลัมน์ที่รวมกันเป็นคีย์หลักไว้ดังแสดงคือ PRIMARY KEY (cid, sid) ทำให้ระเบียน 2 ระเบียนหรือมากกว่า ไม่สามารถมีค่า cid และ sid ที่เหมือนกันทั้ง 2 คอลัมน์พร้อมกัน ในกรณีนี้เป็นการกำหนดไม่ให้นักศึกษาลงทะเบียนรายวิชาที่ตนเองลงทะเบียนไปแล้ว (เป็นเพียงตัวอย่างเท่านั้นอาจไม่ตรงตามหลักการลงทะเบียนทั่วไป) เราไม่สามารถสร้างเงื่อนไขบังคับนี้โดยใช้รูปแบบการเขียน PRIMARY KEY ไว้หลังคอลัมน์ cid และ sid ได้ในกรณีนี้เนื่องจาก PRIMARY KEY ต้องมีเพียงชุดเดียวเท่านั้น

ตัวอย่าง NOT NULL CONSTRAINT

การกำหนดเงื่อนไขบังคับค่าว่าง โดยกำหนดให้คอลัมน์ cid และ name ในตาราง student ห้ามเป็นค่าว่าง sid เป็นคีย์หลัก

```
CREATE TABLE students
(
        cid     CHAR(13) NOT NULL,
        sid     CHAR(8) PRIMARY KEY,
        name    VARCHAR(50) NOT NULL,
        login   VARCHAR(50),
        age     INTEGER,
        gpa     REAL
);
```

จากตัวอย่างเงื่อนไขบังคับแต่ละประเภทสามารถสร้างพร้อมๆ กันได้ PRIMARY KEY CONSTRAINT สามารถสร้างได้ด้วยรูปแบบการนำมาเขียนไว้ท้ายการสร้างตารางในขณะที่ NOT NULL CONSTRAINT นิยมเขียนในลักษณะนี้ เนื่องจากรายละเอียดปลีกย่อยของ CONSTRAINT ที่เกี่ยวข้องกับค่าโดยปริยาย ซึ่งไม่ได้กล่าวถึงในที่นี้

เงื่อนไขบังคับดังกล่าวทำให้การเพิ่มระเบียนใหม่ในตารางนักศึกษาห้ามปล่อยให้ค่า cid และ name รวมถึง sid เป็นค่าว่าง

ตัวอย่าง DEFAULT, UNIQUE, CHECK AND OTHER CONSTRAINTS

การกำหนดเงื่อนไขบังคับต่างๆ

```
CREATE TABLE students
(
        cid     CHAR(13) UNIQUE KEY,
        sid     CHAR(8) PRIMARY KEY,
        name    VARCHAR(50) NOT NULL,
        login   VARCHAR(50),
        age     INTEGER DEFAULT 18,
        gpa     REAL,
        CONSTRAINT CK_students_gpa CHECK (gpa BETWEEN 0 AND 4)
);
```

จากตัวอย่างมีการกำหนดเงื่อนไขบังคับกับคอลัมน์แต่ละคอลัมน์ดังนี้

cid ห้ามไม่ให้ค่าในคอลัมน์นี้ซ้ำกันระหว่างระเบียน

sid เป็นคีย์หลัก

name ห้ามเป็นค่าว่าง

login ไม่ได้ระบุเงื่อนไขบังคับใดๆ สามารถเป็นค่าว่างได้กรณีไม่ระบุค่า แต่ยังคงต้องเป็นข้อมูลชนิด VARCHAR ความยาวไม่เกิน 50 เท่านั้น

age ค่าโดยปริยายกรณีไม่ระบุ จะมีค่าเท่ากับ 18

gpa ตรวจสอบค่าจะต้องอยู่ในช่วง 0.00 ถึง 4.00 เท่านั้น

รูปแบบการกำหนดเงื่อนไขบังคับสามารถเลือกได้ตามความเหมาะสม

ตัวอย่าง FOREIGN KEY CONSTRAINT

การกำหนดเงื่อนไขบังคับคีย์ภายนอก กำหนดให้ค่ารหัสนักศึกษา sid ในตารางลงทะเบียน enrolled ต้องอ้างอิงถึงถึง sid ในตาราง student เท่านั้น

```
CREATE TABLE enrolled
(
        cid     CHAR(13) NOT NULL,
        sid     CHAR(8) NOT NULL,
        grade   FLOAT,
        CONSTRAINT FK_student_enrolled FOREIGN KEY (sid) REFERENCES students (sid)
);
```

จากตัวอย่างเงื่อนไขบังคับคีย์ภายนอกสร้างจาก **CONSTRAINT FK_student_enrolled FOREIGN KEY (sid) REFERENCES students (sid)** โดย

CONSTRAINT เป็นคำสั่งที่ใช้ในการสร้างเงื่อนไขบังคับคีย์ภายนอก

FK_student_enrolled เป็นชื่อของเงื่อนไขบังคับ

FOREIGN KEY คือประเภทของเงื่อนไขบังคับ

(sid) คือชื่อของคอลัมน์ในตารางปัจจุบันซึ่งค่าในตารางนี้อ้างอิงถึงตารางหลัก ข้อมูลที่จะบรรจุในคอลัมน์นี้ต้องปรากฏอยู่ในระเบียนใดๆ ของตารางหลักและคอลัมน์ที่กำหนด

REFERENCES เป็นส่วนของคำสั่งสร้างเงื่อนไขบังคับคีย์ภายนอก วางไว้ระหว่างวัตถุอ้างอิงทั้ง 2

students (sid) คือชื่อของตาราง (students) และชื่อของคอลัมน์ (sid) ในตารางหลัก

ในที่นี้การเพิ่มข้อมูลในตารางการลงทะเบียน รหัสนักศึกษาที่จะเพิ่มจะต้องปรากฏอยู่ในตาราง students ด้วย เพื่อป้องกันความผิดพลาดของการกรอกรหัสนักศึกษาผิดในตารางลงทะเบียน

ตัวอย่าง DROP CONSTRAINT

การลบเงื่อนไขบังคับ—CONSTRAINT คำสั่งต่อไปนี้แสดงการลบเงื่อนไขบังคับ PRIMARY KEY ในตาราง students ที่ได้สร้างไว้ก่อนหน้า

```
DROP CONSTRAINT PK_students;
```

เงื่อนไขบังคับ PK_student จะถูกลบทิ้ง คอลัมน์ sid จะไม่ได้รับการตรวจสอบค่าให้เป็นไปตามเงื่อนไขคีย์หลักอีกต่อไป กล่าวคือค่าในคอลัมน์ sid สามารถเป็นค่าว่างและอาจมีค่าซ้ำกันได้ภายในตารางเดียวกัน รูปแบบการลบเงื่อนไขบังคับดังกล่าวใช้กับเงื่อนบังคับทุกประเภท โดยเปลี่ยนเพียงชื่อของเงื่อนไขบังคับ PK_students เป็นเงื่อนไขบังคับที่ต้องการลบ

การกำหนดเงื่อนไขบังคับบูรณภาพ หรือ integrity constraints ต่างๆ เป็นการกำหนดนิยามของโครงสร้างของฐานข้อมูลให้มีความสามารถในการตรวจสอบการแก้ไขข้อมูลใดที่ข้อมูลในฐานข้อมูลต้องมีสภาพถูกต้องสอดคล้องกับการใช้งานจริงอยู่ตลอดเวลา นับเป็นประโยชน์สำคัญที่ได้รับจากการใช้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ SQL ในการสร้างเงื่อนไขบังคับบูรณภาพได้แก่คำสั่ง CREATE CONSTRAINT ซื่อเงื่อนไขบังคับบูรณภาพต่างๆ ตามข้อกำหนดของฐานข้อมูล และจากการออกแบบฐานข้อมูล สามารถรองรับได้ด้วย CONSTRAINT: PRIMARY KEY, NOT NULL, DEFAULT, UNIQUE KEY, FOREIGN KEY และ CHECK

SQL อนุญาตให้ผู้พัฒนาระบบฐานข้อมูลสร้างกลไกการรักษาบูรณาภาพของข้อมูลที่ซับซ้อนมากกว่าการใช้ CONSTRAINT เช่นการใช้ TRIGGERS ซึ่งจะได้กล่าวถึงในบทที่ 8 สำหรับในหัวข้อถัดไปเป็นการอธิบายการสร้างดรรชนี ซึ่งเป็นความสามารถอีกอันหนึ่งของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่ SQL รองรับได้

7.6 ดรรชนี Index

ดรรชนีหรือ index เป็นกลไกในการเข้าถึงข้อมูลระเบียนใดๆ ได้รวดเร็วขึ้น เป็นความสามารถของ SQL ในการจัดการฐานข้อมูลระดับกายภาพเพื่อเพิ่มประสิทธิภาพของการใช้งานฐานข้อมูล หัวข้อต่อไปนี้อธิบายถึงหลักการและแนวคิดเกี่ยวกับ index ตลอดจนการจัดการ index ด้วยคำสั่ง SQL

7.6.1 แนวคิดทั่วไปเกี่ยวกับ Index

พิจารณาข้อมูลตารางนักศึกษาที่ได้อ้างถึงในบทนี้ ถ้าข้อคำถามที่ได้รับการใช้งานบ่อยๆ คือ “แสดงรายชื่อนักศึกษาเรียงตามลำดับตัวอักษรตามพจนานุกรม” ระบบจัดการฐานข้อมูลจะมีกลวิธีใดในการจัดเรียงข้อมูลเพื่อแสดงผลข้อมูลให้เรียงตามลำดับ หากข้อมูลในตารางของนักศึกษาในฐานข้อมูลนั้นไม่ได้เรียงตามลำดับอยู่ ซึ่งอาจจะ

เรียงตามรหัสนักศึกษาหรือไม่ก็ได้ เนื่องจากข้อมูลที่เพิ่มเข้ามาใหม่มักต่อท้ายข้อมูลเดิม ข้อมูลในตารางที่จัดเก็บในอุปกรณ์จัดเก็บจึงมักไม่ได้เรียงลำดับตามตัวอักษร

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

name
สมชาย
สมศรี
สมศรี
สมศักดิ์
สมน้ำหน้า

จากตัวอย่างนี้ระบบจัดการฐานข้อมูลจึงต้องนำรายชื่อนักศึกษาที่ได้รับจากการประมวลผลการแสดงข้อมูลในรูปด้านขวา มาเรียงลำดับอีกทีหนึ่ง

การเรียงลำดับนั้นเป็นการประมวลผลที่ต้องใช้เวลาพอสมควร เพื่อให้เข้าใจถึงหลักการประมวลผลการเรียงลำดับข้อมูล เราสามารถพิจารณาวิธีการเรียงลำดับข้อมูลรายชื่อนักศึกษาด้วยวิธีการเรียงลำดับข้อมูลต่อไปนี้

รอบที่ 1	สมชาย	สมศรี	สมศรี	สมศักดิ์	สมน้ำหน้า
รอบที่ 2	สมชาย	สมศรี	สมศรี	สมศักดิ์	สมน้ำหน้า
อบที่ 3	สมชาย	สมศรี	สมศรี	สมศักดิ์	สมน้ำหน้า
รอบที่ 4	สมชาย	สมศรี	สมศรี	สมศักดิ์	สมน้ำหน้า
สิ้นสุด	สมชาย	สมน้ำหน้า	สมศรี	สมศรี	สมศักดิ์

name
สมชาย
สมน้ำหน้า
สมศรี
สมศรี
สมศักดิ์

การเรียงลำดับที่ได้แสดงนั้นมีหลักการคร่าวๆ คือการเปรียบเทียบข้อมูลในลำดับแรกกับลำดับถัดไป และให้ทำการเรียงลำดับข้อมูลนั้น จากนั้นนำข้อมูลลำดับถัดไปมาเปรียบเทียบกับข้อมูลที่ได้เรียงลำดับไว้แล้วว่าข้อมูลใหม่ที่นำมาเปรียบเทียบควรอยู่ในตำแหน่งใดของรายการที่ได้เรียงไว้แล้ว จากนั้นนำข้อมูลตัวถัดไปมาเปรียบเทียบอีกไปเรื่อยๆ ซึ่งสามารถอธิบายได้ตามขั้นตอนที่แสดงด้านบนดังนี้

รอบที่ 1 พิจารณาข้อมูลรายชื่อนักศึกษาเริ่มต้นที่ไม่ได้เรียงกันตามลำดับตัวอักษร ในบรรทัด “รอบที่ 1” ซึ่งยังไม่ได้ทำการเรียงลำดับข้อมูลใดๆ ให้ทำการเปรียบเทียบข้อมูลในลำดับแรกกับลำดับถัดมาคือ “สมชาย” และ “สมศรี” ว่าควรจะเรียงลำดับตามตัวอักษรอย่างไร เราพบว่าข้อมูลเรียงตามลำดับตัวอักษรอยู่แล้วให้ทำการเปรียบเทียบในรอบถัดไป

รอบที่ 2 หมายถึงได้มีการเปรียบเทียบข้อมูลแล้วจำนวน 1 คู่ใน 2 ลำดับแรก จากนั้นทำการเปรียบเทียบในรอบที่ 2 โดยการนำรายชื่อในลำดับที่ 3 คือ “สมศรี” มาเปรียบเทียบกับรายการ “สมชาย

สมศรี" ว่ารายชื่อที่นำมาเปรียบเทียบใหม่ควรอยู่ที่ใด โดยจะต้องทำการเปรียบเทียบข้อมูลตั้งแต่ตัวแรกคือเปรียบเทียบกับ "สมชาย" ไปจนกระทั่งพบตำแหน่งที่สามารถวางข้อมูลได้คือหลัง "สมศรี" คนแรก

รอบที่ 3 หลังจากนั้นให้นำรายชื่อถัดไปได้แก่ "สมศักดิ์" มาเปรียบเทียบกับรายการที่ได้เรียบลำดับแล้ว "สมชาย สมศรี สมศรี" เพื่อหาตำแหน่งว่างของ "สมศักดิ์" โดยต้องเริ่มต้นเปรียบเทียบกับ "สมชาย" ไปจนกระทั่งพบว่าต้องวางไว้หลัง "สมศรี" คนที่ 2

รอบที่ 4 หลังจากรอบที่ 3 คือรอบที่ 4 ซึ่งเป็นรอบสุดท้ายตามจำนวนข้อมูลที่มี นำ "สมน้ำหน้า" มาเปรียบเทียบกับรายการ "สมชาย สมศรี สมศรี สมศักดิ์" โดยต้องเริ่มเปรียบเทียบกับ "สมชาย" ไปจนกระทั่งพบว่าต้องวางไว้ระหว่าง "สมชาย" และ "สมศรี"

เราพบว่าการเรียงลำดับที่ได้ยกตัวอย่างนั้น การต้องมีการเปรียบเทียบข้อมูลเป็นจำนวนรอบเท่าๆ กับจำนวนข้อมูลที่มีอยู่ และแต่ละรอบก็จะต้องทำการเปรียบเทียบข้อมูลเท่าๆ กับจำนวนข้อมูลที่มีอยู่เช่นกัน ทำให้จะต้องมีการเปรียบเทียบค่าถึง จำนวนข้อมูลยกกำลังสอง หรือถ้าข้อมูลมีจำนวน n ระเบียน วิธีการเรียงลำดับนี้จะต้องทำการเปรียบเทียบข้อมูลเป็นจำนวนประมาณ n^2 ครั้ง ซึ่งในการใช้งานจริงนั้นข้อมูลนักศึกษาอาจมีมากถึงกว่า 10,000 ระเบียน เวลาที่ใช้ในการเปรียบเทียบข้อมูลนักศึกษา $10{,}000^2$ ครั้งที่ยังไม่รวมถึงเวลาในการอ่านข้อมูล การปรับลำดับของข้อมูลนั้น ใช้เวลาในการประมวลผลอย่างมาก

วิธีการเรียงลำดับข้อมูลที่ได้ยกตัวอย่างอาศัยอัลกอริธึมในการเรียงลำดับอย่างง่าย ถึงแม้ว่าเราจะใช้อัลกอริธึมในการเรียงลำดับที่มีประสิทธิภาพมากกว่านี้ เราก็สามารถทำได้ดีที่สุดที่ $n \log_2 n$ ซึ่งยังคงเป็นการประมวลผลที่ใช้เวลามากสำหรับของมูลขนาดใหญ่ หากระบบจัดการฐานข้อมูลไม่มีกลไกในการจัดการที่ดีพอนั้น ข้อคำถามบางคำถามที่ควรจะใช้เวลาไม่กี่วินาทีอาจใช้เวลาเป็นชั่วโมงในการประมวลผล และหากข้อคำถามในการแสดงข้อมูลแบบเรียงลำดับนี้จะต้องกระทำบ่อยครั้งในแต่ละวันหรือในแต่ละชั่วโมงหรือแม้แต่นาที ระบบจัดการฐานข้อมูลอาจมีประสิทธิภาพลดลงอีกเนื่องจากต้องยุ่งอยู่กับการประมวลผลใหม่ตลอดเวลา

ปัญหาการใช้ความมานะอย่างมากในการเรียงลำดับข้อมูลนี้เป็นเพียงตัวอย่างหนึ่งของความจำเป็นในการที่ระบบจัดการฐานข้อมูลต้องมีกลไกในการแก้ปัญหาการจัดเก็บของข้อมูลให้สอดคล้องกับการใช้งาน ในความเป็นจริงแล้วยังมีปัญหาที่สำคัญอื่นๆ อีก เช่นการระบุตำแหน่งทางกายภาพบนอุปกรณ์จัดเก็บข้อมูลของระบบจัดการฐานข้อมูลว่าข้อมูลระเบียนใดๆ อยู่ที่ตำแหน่งใดของอุปกรณ์จัดเก็บในการเข้าถึงได้รวดเร็ว

Index เป็นกลไกที่ใช้ช่วยให้การเข้าถึงข้อมูลในลักษณะที่ได้กล่าวมารวดเร็วขึ้น โดยมีหลักการทำงานง่ายๆ คือให้ทำการสร้างข้อมูลที่เรียงลำดับไว้อยู่แล้วแยกต่างหากจากตาราง เช่น ให้สร้าง index ของคอลัมน์ชื่อนักศึกษา และ index ของคอลัมน์รหัสนักศึกษา

Index บน
คอลัมน์ name

Index บน
คอลัมน์ sid

Students

name	sid
สมชาย	B5075666
สมน้ำหน้า	B5075832
สมศรี	B5075688
สมศรี	B5075650
สมศักดิ์	B5075831

sid	address
B5075650	3
B5075666	1
B5075688	2
B5075831	4
B5075832	5

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

พิจารณาการสร้างผลลัพธ์จากข้อคำถามต่อไปนี้ โดยอาศัย index

"แสดงรายการข้อมูลนักศึกษาประกอบด้วยรหัสนักศึกษา ชื่อ และเกรดเฉลี่ยเรียงตามลำดับรหัสนักศึกษา" การสร้างรายการข้อมูลนักศึกษาดังกล่าวเริ่มจากตาราง index บนคอลัมน์ sid ซึ่งประกอบด้วยรหัสนักศึกษา sid และตำแหน่งที่อยู่ของระเบียนเชิงตรรกะ ดังแสดงในตารางข้อมูล กระบวนการแสดงผลเริ่มจากการเข้าถึงข้อมูลแรกได้แก่ นักศึกษารหัส "B5075650" ซึ่งอยู่ในตำแหน่งที่ 3 จึงได้ผลลัพธ์เป็นระเบียนแรก คือนักศึกษา "สมศรี" จากนั้นพิจารณานักศึกษาคนถัดมาใน index ได้แก่นักศึกษา "B5075666" ซึ่งอยู่ในลำดับที่ 1 ของตารางข้อมูล (ดังแสดงตามแผนภาพด้านบนด้วยเส้นทึบ) เมื่อดำเนินการอย่างนี้เรื่อยไปจนได้ผลลัพธ์ตามข้อคำถามซึ่งแสดงได้ดังนี้

sid	name	gpa
B5075650	สมศรี	3.82

→

sid	name	gpa
B5075650	สมศรี	3.82
B5075666	สมชาย	3.44

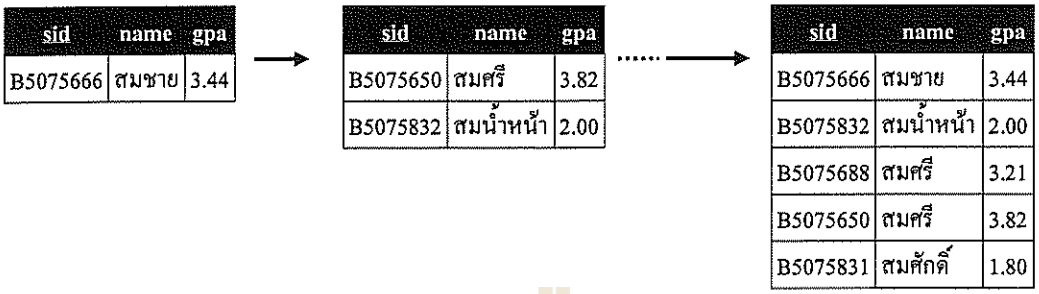
.....→

sid	name	gpa
B5075650	สมศรี	3.82
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

การกระทำในลักษณะดังกล่าวเปรียบเสมือนกับดัชนีในหนังสือซึ่งแสดงรายการของคำสำคัญตามลำดับตัวอักษร ควบคู่กับหน้าของหนังสือที่คำนั้นๆ ปรากฏ เช่นเดียวกับการสร้าง index ในระบบจัดการฐานข้อมูลดังได้แสดงนั่นเอง ที่ข้อมูล index เรียงลำดับในคอลัมน์ใดๆ ควบคู่กับตำแหน่งที่อยู่ในตารางข้อมูลจริง ซึ่งกลไกดังกล่าวจะช่วยให้การเข้าถึงข้อมูลนั่นรวดเร็วขึ้นมาก โดยเฉพาะการจัดการกับข้อมูลจำนวนมหาศาล

พิจารณาการสร้างผลลัพธ์จากข้อคำถามเพิ่มเติมต่อไปนี้ โดยอาศัย index

"แสดงรายการข้อมูลนักศึกษาประกอบด้วยรหัสนักศึกษา ชื่อ และเกรดเฉลี่ยเรียงตามลำดับ**ชื่อนักศึกษา**"



sid	name	gpa
B5075666	สมชาย	3.44

sid	name	gpa
B5075650	สมศรี	3.82
B5075832	สมน้ำหน้า	2.00

sid	name	gpa
B5075666	สมชาย	3.44
B5075832	สมน้ำหน้า	2.00
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80

จากภาพ ให้ใช้ภาพ index ประกอบการอธิบาย โดยการแสดงข้อมูลเรียงลำดับตามตัวอักษรของชื่อนั้นเริ่มจากตาราง index ของ name นักศึกษา "สมชาย" เป็นนักศึกษาคนแรก ซึ่งมีรหัสนักศึกษา "B5075666" เราสามารถค้นหาตำแหน่งที่อยู่ของระเบียนในตารางข้อมูลได้อย่างรวดเร็วเนื่องจากข้อมูลใน index บนคอลัมน์ sid เรียงลำดับไว้เช่นกัน (ดังแสดงตามเส้นประ) จากนั้นระบบจัดการฐานข้อมูลจะสามารถเข้าถึงระเบียนข้อมูลได้ทันทีด้วยตำแหน่งของระเบียนใน index บนคอลัมน์ sid ดังเช่นตัวอย่างที่แล้ว เมื่อดำเนินการอย่างนี้เรื่อยไปจะได้ผลลัพธ์ที่เรียงตามรายชื่อ การดำเนินการเช่นนี้รวดเร็วกว่าการต้องเรียงลำดับทุกครั้งในการแสดงข้อมูลเป็นอย่างมาก

ขอให้พิจารณาว่า การแสดงข้อมูลตามข้อคำถามด้วย index ทั้ง 2 กรณีต่างกัน กล่าวคือ สำหรับ sid นั้นข้อมูลใน index ชี้ไปยังตำแหน่งในตารางข้อมูล แต่ index บนคอลัมน์ name ข้อมูลชี้ไปยัง index บนคอลัมน์ sid อีกทีหนึ่ง ทั้งนี้เนื่องจากข้อมูลที่บรรจุใหม่อาจจะมีการเปลี่ยนตำแหน่งในตารางได้ตลอดเวลา ดังนั้นการสร้าง index ที่อ้างอิงกับตำแหน่งที่อยู่ในตารางข้อมูลเพียงหนึ่งชุดจะทำให้ระบบจัดการฐานข้อมูลมีประสิทธิภาพมากกว่า กล่าวคือ ระบบจัดการฐานข้อมูลไม่จำเป็นต้องปรับปรุง index ทุกๆ index ที่สร้างขึ้น เพียงแต่ปรับปรุง index หลักที่อ้างอิงถึงตำแหน่งที่อยู่ของระเบียนเท่านั้น ซึ่งโดยทั่วไปแล้วระบบจัดการฐานข้อมูลจะทำการจัดเก็บข้อมูลโดยอาศัยกลไก index บนคอลัมน์ที่เป็น PRIMARY KEY นั่นเอง การที่การดำเนินการต่างๆ ในตารางจำเป็นต้องอาศัย PRIMARY KEY นี้เอง การเลือก PRIMARY KEY ในกรณีที่มี candidate key จึงต้องคำนึงถึงความกระชับของ PRIMARY KEY ด้วย จากตัวอย่างนี้เราสามารถสร้าง index ได้บนคอลัมน์ใดๆ ก็ได้ที่ต้องการเพื่อเพิ่มประสิทธิภาพการแสดงผลข้อมูล โดย index ที่สร้างขึ้นจะอ้างอิงกับ PRIMARY KEY

อย่างไรก็ตามการสร้าง index ไม่ได้มีแต่ข้อดีเสมอไป เนื่องจาก index ที่สร้างขึ้นจะต้องถูกปรับปรุงให้ถูกต้องอยู่เสมอ เช่นเมื่อมีการเพิ่มข้อมูลนักศึกษาใหม่ index บนคอลัมน์รายชื่อนักศึกษาก็จะต้องมีการจัดการใหม่เพื่อให้เรียงลำดับตามรายชื่อนักศึกษาเช่นเดิม ซึ่งก็เป็นการใช้ทรัพยากรของระบบที่รองรับระบบจัดการฐานข้อมูลส่วนหนึ่ง ถ้าเราสร้าง index ที่เกินความจำเป็น เช่นสร้าง index บนคอลัมน์ gpa หรือ age เพิ่มขึ้นอีก ทุกครั้งที่มีการเพิ่มหรือแก้ไขระเบียนนักศึกษา ระบบจัดการฐานข้อมูลตัวปรับปรุง index ของ gpa และ age เพิ่มขึ้นอีก เป็นการเพิ่มภาระให้กับระบบ นอกจากนี้ข้อคำถามที่มีการเข้าถึงข้อมูลโดยอาศัยกลไก index บนคอลัมน์ gpa และ age อาจมีไม่บ่อยครั้ง จึงเป็น

การไม่คุ้มค่าที่จะสร้าง index โดยไม่จำเป็น ทั้งนี้นักวิเคราะห์ระบบ—SA และผู้ดูแลระบบ—DBA มีหน้าที่วิเคราะห์การสร้าง index ที่เหมาะสมตามข้อกำหนดการใช้งานหรือจากสถิติการใช้งานจริง

ในทางปฏิบัติแล้ว ระบบจัดการฐานข้อมูลจะใช้เทคนิคที่มีประสิทธิภาพในการจัดการกับ index มากกว่านี้ โดยไม่ได้ทำ index ทุกระเบียน แต่จะอาศัยกลุ่มของระเบียนที่รวมกันเรียกว่า page โดยในแต่ละ page จะมีระเบียนของข้อมูลที่อาจเรียงกันตาม index หลัก ซึ่งมักจะเรียงกันตาม PRIMARY KEY เมื่อจะทำการเข้าถึงข้อมูลระเบียนใดๆ ระบบจัดการฐานข้อมูลจะค้นหาว่าระเบียนนั้นๆ อยู่ใน page ใด แล้วทำการเข้าถึงข้อมูลที่ page นั้นเพื่อเข้าถึงข้อมูลในระเบียนที่ต้องการเข้าถึงอีกทีหนึ่ง นอกจากนี้ยังมีเทคนิคและรายละเอียดปลีกย่อยอื่นๆ ของการจัดการฐานข้อมูลในระดับกายภาพซึ่งไม่ได้ลงรายละเอียดในเอกสารเล่มนี้

SQL รองรับแนวคิด index นี้ โดยอำนวยความสะดวกให้ผู้จัดการระบบฐานข้อมูลสร้างและจัดการ index ได้ด้วยคำสั่งที่ไม่ซับซ้อนเพื่อให้ระบบจัดการฐานข้อมูลอาศัยโครงสร้าง index ที่ได้นิยามไว้นี้ในการจัดการกับข้อมูล

7.6.2 การสร้าง Index

การสร้าง index ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

CREATE INDEX <index_name> ON <table_name> (<column_name>);

โดย

CREATE INDEX	คือคำสั่งที่ใช้ในการสร้างดัชนี
index_name	ชื่อของ index เพื่อใช้ในการจัดการในภายหลัง
ON	ประกอบคำสั่ง CREATE INDEX เพื่อระบุว่าสร้าง index "บน" คอลัมน์
table_name	ชื่อตารางที่ต้องการสร้าง index
column_name	ชื่อของคอลัมน์สำหรับสร้าง index

สามารถเขียนรูปแบบคำสั่งสร้าง index ที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

สร้าง ดัชนี <ชื่อดัชนี> บน <ชื่อตาราง>(<ชื่อคอลัมน์>);

ตัวอย่าง

การสร้าง index บนคอลัมน์ name ในตาราง students

CREATE INDEX idx_students_name ON students(name);

จากตัวอย่าง idx_students_name เป็นชื่อของ index ซึ่งเราจะนำมาใช้จัดการในภายหลัง คำสั่งดังกล่าวสร้างโครงสร้าง index ที่ระบุไว้ การเข้าถึงข้อมูลที่อ้างถึงชื่อนักศึกษาและอาศัยกลไกการทำงานของ index จะมีความรวดเร็วขึ้น

7.6.3 การลบ Index

การสร้าง index ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

DROP INDEX <index_name>;

โดย

DROP INDEX คือคำสั่งที่ใช้ในการลบดัชนี

index_name ชื่อของ index ที่ต้องการลบ

สามารถเขียนรูปแบบคำสั่งลบ index ที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

ลบ ดัชนี <ชื่อดัชนี>;

ตัวอย่าง

การลบ index บนคอลัมน์ name ในตาราง students ที่ได้สร้างไว้

DROP INDEX idx_students_name;

จากตัวอย่าง idx_students_name เป็นชื่อของ index ซึ่งเราจะนำมาใช้จัดการในภายหลัง คำสั่งดังกล่าวสร้างโครงสร้าง index ที่ระบุไว้ การเข้าถึงข้อมูลที่อ้างถึงชื่อนักศึกษาและอาศัยกลไกการทำงานของ index จะมีความรวดเร็วขึ้น

7.7 วิว View

วิว เป็นตารางข้อมูลเสมือนที่ไม่ได้มีข้อมูลอยู่จริง สามารถเปรียบได้กับหน้ากากที่ครอบตารางความสัมพันธ์ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เพื่อให้ได้ตารางของข้อมูลที่ตรงกับการใช้งาน เป็นการกำหนดสคีมาของฐานข้อมูลระดับภายนอก สำหรับประโยชน์และการจัดการวิวจะได้อธิบายในหัวข้อนี้

7.7.1 แนวคิดทั่วไปเกี่ยวกับ View

วิวมีประโยชน์หลักๆ 3 ประการได้แก่การสร้างตารางเสมือนเพื่อเพิ่มความสะดวกในการเรียกใช้งานข้อมูล การสร้างตารางเสมือนเพื่อคงความเป็นอิสระของข้อมูล และเรื่องเกี่ยวกับความปลอดภัยของข้อมูล

ในหัวข้อที่ 1.6.3 ได้กล่าวถึงการใช้งานวิวเพื่อสร้างตารางเสมือน ซึ่งเป็นการใช้งานประโยชน์หลักทั้ง 3 ประการของวิว โดยเฉพาะในเรื่องการคงความเป็นอิสระของข้อมูล นอกจากตัวอย่างในหัวข้อที่ 1.6.3 แล้ว ในที่นี้เราจะพิจารณาการใช้งานวิวด้วยกรณีตัวอย่างข้อมูลของตาราง 3 ตารางดังนี้

Faculty

fid	fname	sal
001	แพนเค้ก	40,000
007	สถิตย์	50,000
069	พอลล่า	30,000

Teaches

fid	cid
001	204205
007	204203
007	204204
007	204329
069	202102
069	202103

Courses

cid	cname	credits
202102	Information Technology I	3
202103	Information Technology II	3
204203	Software Design and Development	4
204204	Database Design and Development	4
204205	Multimedia Design and Development	4
204329	Web Technology	3

ในกรณีที่ข้อมูลที่จะต้องใช้บ่อยๆ คือการแสดงรายชื่อของอาจารย์ผู้สอนกับรายวิชาที่สอน เราสามารถสร้างวิวที่แสดงข้อมูลจากทั้ง 3 ตารางได้ดังนี้

Faculty_Teaches_Course

fid	fname	cid	cname
001	แพนเค้ก	204205	Multimedia Design and Development
007	สถิตย์	204203	Software Design and Development
007	สถิตย์	204204	Database Design and Development
007	สถิตย์	204329	Web Technology
069	พอลล่า	202102	Information Technology I
069	พอลล่า	202103	Information Technology II

ซึ่งตารางข้อมูลนี้สามารถสร้างให้อยู่ในรูปของตารางเสมือนเพื่อเรียกใช้งานได้บ่อยครั้ง โดยทำการสร้างวิว faculty_teaces_courses ขึ้นมาเพื่อความสะดวกได้

นอกจากนี้ข้อมูลเงินเดือน sal ของอาจารย์ยังเป็นข้อมูลที่เป็นความลับกับบุคคลทั่วไป แต่เนื่องจากฐานข้อมูลของมหาวิทยาลัยนี้จะต้องใช้ร่วมกันทั้งมหาวิทยาลัย เราสามารถสร้างวิว faculty_public ซึ่งไม่มีข้อมูลเงินเดือนของคณาจารย์เป็นหน้ากากครอบตาราง faculty ไว้ เพื่อให้ระบบลงทะเบียนสามารถเข้าถึงได้เฉพาะวิว faculty_public ในขณะที่ระบบเงินเดือนของมหาวิทยาลัยสามารถเข้าถึงตาราง faculty ได้เพื่อจัดการเกี่ยวกับเงินเดือนของคณาจารย์

Faculty_Public

fid	fname
001	แพนเค้ก
007	สถิตย์
069	พอลล่า

7.7.2 การสร้าง View

การสร้าง view ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

CREATE VIEW <view_name> AS <select_statement>;

โดย

CREATE VIEW	คือคำสั่งที่ใช้ในการสร้างวิว
view_name	ชื่อของวิว เพื่อใช้ในการจัดการในภายหลัง
AS	ประกอบคำสั่ง CREATE VIEW เพื่อระบุว่าสร้าง VIEW “ด้วย” คำสั่ง SELECT ที่กำหนด

select_statement รายการคำสั่ง SELECT ของ SQL ที่ใช้ในการสร้างวิว

สามารถเขียนรูปแบบคำสั่งสร้างวิวที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

สร้าง วิว <ชื่อวิว> บน <คำสั่ง SELECT>;

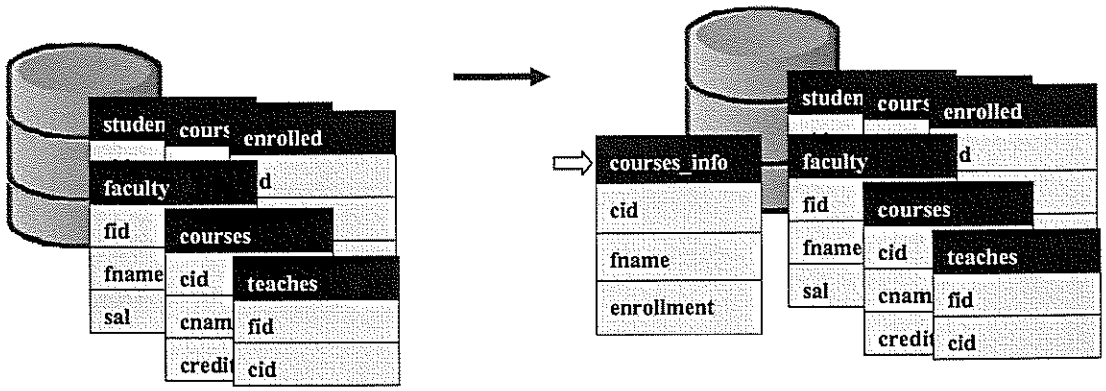
เนื่องจากวิวคือตารางเสมือน การสร้างวิวจึงใช้คำสั่ง SELECT ในการเลือกข้อมูลเพื่อสร้างตารางเสมือนจากตารางข้อมูลที่มีอยู่แล้ว เราเรียกวิวว่าตารางเหมือน—virtual table และตารางข้อมูลจริงในฐานข้อมูลว่าตารางหลัก—base table วัตถุที่เกิดขึ้นจากการสร้างวิวเป็นเพียงโครงสร้างของตารางเสมือน ในการใช้งานวิวนั้นเราใช้คำสั่ง SQL ในการแสดงข้อมูล แก้ไขข้อมูล และลบข้อมูลผ่านทางวิวเสมือนเป็นการกระทำกับตารางข้อมูลจริงตารางหนึ่งโดยระบบจัดการฐานข้อมูลมีกลไกในการจัดการข้อมูลให้มีผลต่อตารางหลักผ่านทางวิว ซึ่งเป็นกฎข้อหนึ่งในกฎ 12 ข้อของ E.F.Codd นั่นเอง สำหรับคำสั่ง SELECT นั้นเป็นคำสั่งในกลุ่มคำสั่งที่ใช้ในการจัดการข้อมูล—DML ของ SQL ที่จะได้อธิบายในบทถัดไป ในที่นี้จึงเป็นการศึกษาถึงรูปแบบการใช้คำสั่งในการสร้างวิวเท่านั้น

ตัวอย่าง

สร้างวิว Course_info (*cid:* string, *fname:* string, *enrollment:* integer) ซึ่งประกอบด้วยคอลัมน์รหัสวิชา cid ชื่ออาจารย์ fname และจำนวนผู้ลงทะเบียน enrollment ซึ่งจะต้องใช้ข้อมูลจากตาราง teaches, faculty และ enrolled ประกอบกันเพื่อแสดงรายการการลงทะเบียนของแต่ละรายวิชา โดยแสดงรหัสวิชา ชื่ออาจารย์ผู้สอน และจำนวนนักศึกษาที่ลงทะเบียน

```
CREATE VIEW courses_info
AS
SELECT e.cid, fname, COUNT(*) AS enrollment
FROM teaches t, faculty t, enrolled e
WHERE t.fid = f.fid
AND t.cid = e.cid
GROUP BY e.cid;
```

จากตัวอย่าง ผลลัพธ์ที่ได้จากกาดำเนินการตามคำสั่ง SQL สร้างวิว คือการกำหนดโครงสร้างของวิวว่าประกอบด้วยคอลัมน์อะไรบ้าง และข้อมูลของแต่ละคอลัมน์ นำมาจากตารางใด เสมือนการสร้างสคีมาของตารางโดยปกติทั่วไปดังแผนภาพต่อไปนี้



คำสั่งสร้างวิวนั้นไม่ได้แสดงข้อมูลใดๆ การเรียกใช้วิวนั้นเราดำเนินการด้วยการใช้คำสั่ง SQL ดังเช่นที่กระทำกับตารางโดยทั่วไป ในกรณีนี้คือตาราง courses_info ซึ่งเป็นวิวที่ได้สร้างขึ้น ยกตัวอย่างการแสดงข้อมูลทั้งหมดจากวิวด้วย SQL ต่อไปนี้

SELECT * FROM courses_info ORDER BY cid

ได้ผลลัพธ์ดังตารางต่อไปนี้

Courses_info

cid	fname	enrollment
204203	สถิตย์	1
204204	สถิตย์	2
204205	แพนเค้ก	1
204329	สถิตย์	2

ตัวอย่างการสร้างวิว Courses_info นี้ นอกจากจะใช้ประโยชน์ในการใช้วิวเพื่อเพิ่มความสะดวกในการเรียกใช้งานข้อมูลที่ใช้บ่อยๆ ผ่านทางวิวที่สร้างขึ้นจาก SQL ที่มีความซับซ้อนแล้ว วิวดังกล่าวยังคงความเป็นอิสระของข้อมูลจากฐานข้อมูลระดับตรรกะ และป้องกันข้อผิดพลาดที่อาจจะเกิดขึ้นจากความซ้ำซ้อนของข้อมูลคอลัมน์ enrollment ซึ่งสามารถศึกษารายละเอียดได้ในบทที่ 1 หัวข้อที่ 1.6.3 ความเป็นอิสระของข้อมูล

ตัวอย่าง

สร้างวิว Faculty_Teaches_Courses (*fid:* string, *fname:* string, *cid:* string, *cname:* string)

CREATE VIEW faculty_teaches_courses

AS

SELECT f.fid, fname, c.cid, cname

FROM faculty f, teaches t, courses c

WHERE f.fid = t.tid AND t.cid = c.cid;

จากตัวอย่าง วิวดังกล่าวประกอบด้วยคอลัมน์ fid, fname, cid และ cname ซึ่งเป็นรายการข้อมูลที่เรียกใช้บ่อยครั้งว่าอาจารย์ท่านใด สอนรายวิชาใดบ้าง โดยข้อมูลสร้างจากตาราง faculty, teaches และ courses เราสามารถยกตัวอย่างการใช้วิวดังกล่าวโดยการแสดงข้อมูลทั้งหมดด้วยคำสั่ง

SELECT * FROM faculty_teaches_courses

ซึ่งผลลัพธ์แสดงได้ตามรูปที่ 1**

ตัวอย่าง

สร้างวิว Faculty_Public (*fid:* string, *fname:* string)

CREATE VIEW faculty_public

AS

SELECT fid, fname

FROM faculty;

จากตัวอย่าง วิวดังกล่าวประกอบด้วยคอลัมน์รหัสอาจารย์ fid และชื่ออาจารย์ fname เท่านั้น ไม่มีคอลัมน์เงินเดือน sal ของอาจารย์ การพัฒนาระบบลงทะเบียนสามารถอ้างอิงถึงเฉพาะวิว faculty_public นี้ได้ โดยเก็บตารางหลัก faculty ที่มีข้อมูลเงินเดือนสำหรับระบบสารสนเทศเงินเดือนของมหาวิทยาลัยเท่านั้นที่สามารถเข้าถึงได้ การเรียกใช้ข้อมูลสามารถยกตัวอย่างได้ด้วยคำสั่ง

SELECT * FROM faculty_public

ได้ผลลัพธ์ตามตารางข้อมูลในรูปที่ 1**

วิวต่างๆ ที่ได้สร้างขึ้นมานั้นสามารถรองรับคำสั่ง SQL ที่ใช้ในการแก้ไขและลบข้อมูลอีกด้วย สำหรับคำสั่งที่ใช้ในการแก้ไขและลบข้อมูลได้อธิบายไว้ในบทถัดไป

7.7.3 การลบ View

การลบ view ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

DROP VIEW <view_name>;

โดย

DROP VIEW คือคำสั่งที่ใช้ในการลบวิว

view_name ชื่อของวิวที่ต้องการลบ

สามารถเขียนรูปแบบคำสั่งลบวิวที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

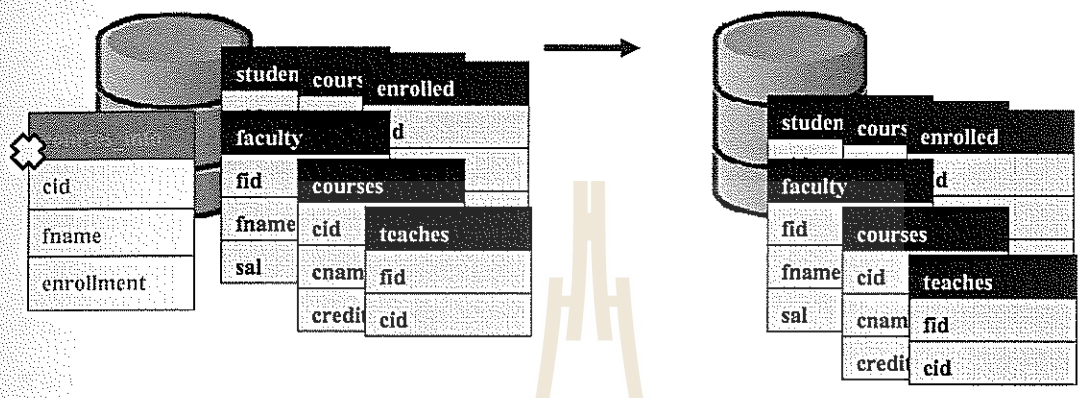
ลบ วิว <ชื่อวิว>;

ตัวอย่าง

การลบวิว Courses_info ที่ได้สร้างไว้

DROP INDEX idx_students_name;

การลบวิวเปรียบเสมือนการลบสคีมาของตารางโดยปกติทั่วไปดังแผนภาพต่อไปนี้



การลบวิวนั้นต่างจากการลบตารางเนื่องจากการลบวิวเป็นเพียงการลบสคีมาหรือโครงสร้างของตารางรางเสมือนเท่านั้น ข้อมูลที่บรรจุอยู่ในตารางหลักที่เกี่ยวข้องกับวิวยังคงอยู่ไม่ได้ถูกลบไปด้วย175

7.8 แบบฝึกหัดท้ายบท

ให้นักศึกษาเขียนคำสั่ง SQL สำหรับสร้างตาราง กำหนดเงื่อนไขบูรณภาพ ตลอดจนสร้างอินเด็กซ์และวิวของโครงงานเช่า-ฉัน-สิ

บทที่ 8 SQL: ภาษาจัดการข้อมูล (Data Manipulation Language)

วัตถุประสงค์

- สามารถอธิบายและแสดงข้อมูลจากฐานข้อมูลด้วยคำสั่ง SELECT ได้
- สามารถอธิบายสร้างคำสั่ง SQL ที่ประกอบด้วย
  - WHERE clause สำหรับแสดงแถวของข้อมูลที่เป็นไปตามเงื่อนไข
  - เรียงลำดับผลลัพธ์จากข้อคำถามด้วยคำสั่ง ORDER BY
  - ใช้งานฟังก์ชันสำหรับข้อมูลเป็นชุด (aggregate function) ด้วยคำสั่ง GROUP BY
  - สร้างข้อคำถามย่อย (subquery)
  - เชื่อมตาราง (join tables)
  - ดำเนินการเกี่ยวกับเซ็ต (UNION, INTERSECTION, EXCEPT)
- สามารถเพิ่มข้อมูลด้วยคำสั่ง INSERT ได้
- สามารถแก้ไขข้อมูลด้วยคำสั่ง UPDATE ได้
- สามารถลบข้อมูลด้วยคำสั่ง DELETE ได้

คำสำคัญ: SELECT; WHERE; ORDER BY; GROUP BY; UNION; INTERSECTION; EXCEPT; INSERT; UPDATE; DELETE

8.1 บทนำ

8.2 SQL Commands

คำสั่ง SQL ที่ใช้ในการจัดการข้อมูล มีวัตถุประสงค์หลักเพื่อใช้ในการค้นคืนและแสดงข้อมูลตามข้อคำถามจากฐานข้อมูล ตลอดจนเพิ่มและแก้ไขข้อมูลในฐานข้อมูล การดำเนินการของคำสั่ง SQL นั้นจะประกอบไปด้วยการดำเนินการ selection, projection และ join ตามพีชคณิตเชิงสัมพันธ์ ในคำสั่ง SQL 1 คำสั่ง การใช้งาน SQL กรณีแรกคือการแสดงข้อมูลแบบต่างๆ มีรูปแบบของคำสั่งดังนี้

```
SELECT      [DISTINCT | ALL] {* | [column_expression [AS new_name]] [,...]}
FROM        table_name [alias] [, ...]
[WHERE      condition]
[GROUP BY   column_list] [HAVING condition]
[ORDER BY   column_list]
```

โดย

column_expression คือ ชื่อของคอลัมน์ในตารางข้อมูลซึ่งอาจมีหลายๆ คอลัมน์ก็ได้หรือนิพจน์อื่นๆ ของคอลัมน์ตลอดจนการกำหนดชื่อใหม่ให้กับคอลัมน์

table_name คือ ชื่อของตารางความสัมพันธ์หรือวิวที่จะทำการค้นคืนข้อมูล

สำหรับลำดับในการดำเนินการของคำสั่ง SQL เรียงตามลำดับดังนี้

1. **FROM** เป็นการกำหนดตารางความสัมพันธ์หรือวิวที่เกี่ยวข้องในคำสั่ง
2. **WHERE** เป็นการระบุเงื่อนไขในการแสดงระเบียนข้อมูลที่ตรงตามเงื่อนไข ซึ่งเป็นการดำเนินการ selection และ join ในพีชคณิตเชิงสัมพันธ์
3. **GROUP BY** เป็นการจัดกลุ่มของระเบียนที่มีข้อมูลในคอลัมน์เหมือนกันตามที่ระบุ
4. **HAVING** เป็นการกรองกลุ่มของข้อมูลให้เป็นไปตามเงื่อนไขที่ระบุ
5. **SELECT** เป็นการเลือกแสดงคอลัมน์ ซึ่งเป็นการดำเนินการ projection ในพีชคณิตเชิงสัมพันธ์
6. **ORDER BY** เป็นการกำหนดรูปแบบการแสดงข้อมูลให้เรียงลำดับตามคอลัมน์ที่กำหนด

ลำดับของคำสั่งไม่สามารถเปลี่ยนได้ และต้องเป็นไปตามไวกรณ์ที่กำหนด SELECT และ FROM เป็นคำที่จะต้องปรากฏอยู่ในคำสั่ง SQL เสมอ ผลลัพธ์ที่ได้จากคำสั่ง SQL คือตารางของข้อมูลผลลัพธ์ หัวข้อต่อไปนี้เป็นการอธิบายและยกตัวอย่างการใช้งานคำสั่ง SQL แบบต่างๆ โดยจะใช้สคีมาตัวอย่างของฐานข้อมูลมหาวิทยาลัยที่ได้กล่าวถึงบ่อยๆ ดังต่อไปนี้

Students (*cid:* string, *sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Faculty (*fid:* string, *fname:* string, *sal:* real)

Courses (*cid:* string, *cname:* string, *credits:* integer)

Rooms (*rno:* integer, *address:* string, *capacity:* integer)

Enrolled (*sid:* string, *cid:* string, *grade:* float)

Teaches (*fid:* string, *cid:* string)

Meets_In (*cid:* string, *rno*: integer, *time:* string)

8.2.1 ข้อคำถามอย่างง่าย (Simple Queries)

- **แสดงข้อมูลทุกระเบียน**

ตัวอย่าง

แสดงข้อมูลทุกระเบียน ทุกคอลัมน์

คำสั่ง SQL สำหรับแสดงข้อมูลทุกระเบียนและทุกคอลัมน์ จะไม่มีการใช้ WHERE ในคำสั่ง

```
SELECT cid, sid, name, login, age, gpa
FROM students;
```

หรือใช้เครื่องหมาย "*" แทนการแสดงข้อมูลทุกคอลัมน์

```
SELECT *
FROM students;
```

จากตัวอย่างเป็นการแสดงข้อมูลทุกๆ คอลัมน์ โดยการระบุชื่อคอลัมน์ทั้งหมดหรือใช้เครื่องหมาย "*" แทน ซึ่งเป็นการแสดงข้อมูลทุกระเบียนและทุกคอลัมน์จากตารางนักศึกษาได้ผลลัพธ์ดังนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ตัวอย่าง

แสดงข้อมูลทุกระเบียน แต่แสดงเฉพาะบางคอลัมน์

คำสั่ง SQL สำหรับแสดงข้อมูลทุกระเบียน จะไม่มีการใช้ WHERE ในคำสั่ง สำหรับการระบุคอลัมน์นั้นจะทำการระบุหลัง SELECT

```
SELECT sid, name, gpa
FROM students
```

จากตัวอย่างเป็นการแสดงข้อมูลทุกๆ ระเบียน สำหรับคอลัมน์นั้นแสดงเพียงคอลัมน์ รหัสนักศึกษา ชื่อนักศึกษา และเกรดเฉลี่ยเท่านั้น ได้ผลลัพธ์ดังนี้

Students

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82
B5075831	สมศักดิ์	1.80
B5075832	สมน้ำหน้า	2.00

ซึ่งผลลัพธ์ที่แสดงไม่ได้มีการเรียงลำดับตามคอลัมน์ใดๆ หรือในกรณีทั่วไประบบจัดการฐานข้อมูลจะแสดงข้อมูลเรียงตามลำดับของคีย์หลัก

ตัวอย่าง

การแสดงข้อมูลที่ไม่ซ้ำกันโดยการใช้ DISTINCT

แสดงรหัสนักศึกษาทั้งหมดที่ได้ลงทะเบียนเรียน

SELECT sid

FROM enrolled;

แสดงรหัสนักศึกษาทั้งหมดที่เคยลงทะเบียนเรียน โดยรหัสนักศึกษาที่ซ้ำกันแสดงเพียงครั้งเดียวเท่านั้นโดยการใช้ DISTINCT

SELECT DISTINCT sid

FROM enrolled;

ได้ผลลัพธ์ดังนี้ ตารางซ้ายมือแสดงข้อมูลทั้งหมดในตาราง Enrolled ในขณะที่ตารางกลางแสดงข้อมูลตามคำสั่ง SQL ที่ไม่ได้ใช้ DISTINCT และตารางทางขวาแสดงข้อมูลที่ได้จากการใช้ DISTINCT ทั้งนี้จะพบว่า รหัสนักศึกษา B5075688 จะแสดงเพียง 1 ระเบียนเท่านั้น

Enrolled

cid	grade	sid
204203	4	B5075650
204204	3	B5075831
204205	1	B5075688
204329	0	B5075666
204204	4	B5075688

ข้อมูลทั้งหมดในตาราง Enrolled

Enrolled

sid
B5075650
B5075831
B5075688
B5075666
B5075688

รหัสนักศึกษาทั้งหมดในการลงทะเบียน

Enrolled

sid
B5075650
B5075831
B5075688
B5075666

รหัสนักศึกษาที่เคยลงทะเบียนโดยไม่แสดงรหัสนักศึกษาที่ซ้ำกัน

ตัวอย่าง

การแสดงข้อมูลโดยมีการคำนวณ

แสดงรายการรหัสนักศึกษา รายชื่อนักศึกษา และผลการเรียนของนักศึกษาเป็นร้อยละจากเกรดเฉลี่ย

```
SELECT sid, name, gpa*100/4
FROM students;
```

ข้อมูลที่ได้จากคำสั่ง SQL ดังกล่าวปรากฏในคอลัมน์หรือฟิลด์สุดท้ายเรียกว่าฟิลด์ที่ได้จากการคำนวณ (calculated filed or computed field or derived field) ในตัวอย่างเป็นการนำเอาเกรดเฉลี่ยในรูปแบบเต็ม 4.00 มาคำนวนให้เป็นร้อยละ โดยการคูณด้วย 100 และหารด้วย 4 ซึ่งฟิลด์ที่ได้จากการคำนวณนี้จะไม่มีชื่อฟิลด์ ผลลัพธ์มักจะเป็นลำดับที่ของคอลัมน์ ในกรณีนี้คือ col3 ซึ่งเราสามารถกำหนดชื่อของคอลัมน์ที่ได้จากการคำนวณโดยการใช้ AS ดังนี้

```
SELECT sid, name, gpa*100/4 AS percentGrade
FROM students;
```

ซึ่งการคำนวณนั้นสามารถใช้เครื่องหมาย บวก ลบ คูณ หาร วงเล็บ ซึ่งการคำนวณนอกจากจะคำนวณจากคอลัมน์เดียวแล้ว ยังสามารถคำนวณได้จากหลายคอลัมน์ โดยค่าของคอลัมน์จะต้องเป็นตัวเลขเท่านั้น จึงจะสามารถคำนวณได้ คำสั่ง SQL ตามตัวอย่าง สร้างตารางผลลัพธ์ดังนี้

Students

sid	name	percentGrade
B5075666	สมชาย	86.00
B5075688	สมศรี	80.25
B5075650	สมศรี	95.50
B5075831	สมศักดิ์	45.00
B5075832	สมน้ำหน้า	50.00

- **การแสดงข้อมูลบางระเบียน Row selection (WHERE clause)**

เป็นการแสดงข้อมูลจากตารางข้อมูลเฉพาะระเบียนที่ตรงตามเงื่อนไข โดยสามารถกำหนดเงื่อนไขในรูปแบบของนิพจน์ต่างๆ ประกอบการใช้ WHERE ซึ่งรูปแบบของเงื่อนไขนั้นอาจมีลักษณะดังนี้

- การเปรียบเทียบค่าของนิพจน์หนึ่งกับค่าของอีกนิพจน์หนึ่ง
- การทดสอบค่าของนิพจน์หนึ่งแบบเป็นช่วงกับช่วงค่าที่ระบุ
- การทดสอบว่าค่าที่ระบุเป็นสมาชิกหรือมีค่าเท่ากับสมาชิกในเซ็ตที่ระบุไว้
- การตรวจความเหมือนกันของรูปแบบสายอักขระตามรูปแบบที่กำหนด
- การทดสอบว่าคอลัมน์นั้นๆ มีค่าว่าง หรือไม่มีค่าว่าง

การดำเนินการ WHERE นั้นเป็นลักษณะของการดำเนินการ selection ในพีชคณิตเชิงสัมพันธ์

ตัวอย่าง

เงื่อนไขการแสดงข้อมูลแบบเปรียบเทียบ

แสดงข้อมูลนักศึกษาทั้งหมดที่มีเกรดเฉลี่ยมากกว่า 2.00

SELECT cid, sid, name, login, age, gpa

FROM students

WHERE gpa > 2;

คำสั่ง SQL ตัวอย่างแสดงข้อมูลนักศึกษาทุกคอลัมน์จากตารางนักศึกษา ในส่วนของช่วงคำสั่ง WHERE นั้นกำหนดเงื่อนไข gpa > 2 หมายถึงให้แสดงข้อมูลเฉพาะระเบียนที่ค่าในคอลัมน์ gpa มีค่ามากกว่า 2 ได้ผลลัพธ์ดังนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82

เครื่องหมายที่สามารถใช้ในนิพจน์สำหรับเงื่อนไขการแสดงข้อมูลได้แก่

เท่ากับ	=
ไม่เท่ากับ	<> หรือ !=
น้อยกว่า	<
มากกว่า	>
น้อยกว่าหรือเท่ากับ	<=
มากกว่าหรือเท่ากับ	>=
และ	AND
หรือ	OR
ไม่	NOT
วงเล็บเปิดและปิด	( )

การดำเนินการในนิพจน์จะกระทำจากซ้ายไปขวาโดยมีลำดับการดำเนินการดังนี้

( ), NOTs, ANDs, และ ORs ตามลำดับ

ตัวอย่าง

เงื่อนไขการแสดงข้อมูลแบบเปรียบเทียบหลายการเปรียบเทียบ

แสดงข้อมูลของนักศึกษาทั้งหมดที่มีอายุ 18 หรือ 19 ปี

```
SELECT *
FROM students
WHERE age = 18 OR age = 19;
```

จากตัวอย่างเป็นการแสดงข้อมูลทุกคอลัมน์จากตารางนักศึกษา โดยกำหนดเงื่อนไขแบบเปรียบเทียบ แสดงเฉพาะข้อมูลของนักศึกาที่มีอายุ 18 หรือ 19 ปี โดยใช้ OR ในการเชื่อมการเปรียบเทียบทั้ง 2 ผลลัพธ์จึงเป็นระเบียนของนักศึกษาที่มีอายุ 18 หรือ 19 ปี ดังต่อไปนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82

ตัวอย่าง

เงื่อนไขในการแสดงข้อมูลแบบเป็นช่วง (BETWEEN/NOT BETWEEN)

แสดงข้อมูลนักศึกษาที่มีเกรดเฉลี่ยระหว่าง 3.25 ถึง 3.49 ซึ่งเป็นการแสดงนักศึกษาที่มีระดับคะแนนเฉลี่ยอยู่ในช่วงเกียรตินิยมอันดับ 2 ซึ่งในตัวอย่างนี้จะใช้ BETWEEN ซึ่งค่าระหว่างในที่นี้รวมค่าขอบเขตล่างและบนของช่วงด้วย

```
SELECT sid, name, gpa
FROM student
WHERE gpa BETWEEN 3.25 AND 3.49;
```

เราสามารถแสดงข้อมูลโดยกำหนดช่วงแบบกลับกันคือแสดงข้อมูลนักศึกษาที่ไม่มีเกรดเฉลี่ยอยู่ในช่วงที่กำหนดโดยใช้ **NOT BETWEEN** ได้ สำหรับตัวอย่างด้านบนนั้นตรงกับการใช้ SQL แบบเปรียบเทียบ 2 การเปรียบเทียบดังนี้

```
SELECT sid, name, gpa
FROM student
WHERE gpa > = 3.25 AND gpa < = 3.49;
```

อย่างไรก็ตามจะพบว่าการใช้ BETWEEN นั้นดูง่ายกว่า ซึ่งได้ผลลัพธ์ดังนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44

ตัวอย่าง

เงื่อนไขการแสดงข้อมูลโดยตรวจสอบว่าเป็นสมาชิกในเซ็ตที่กำหนดหรือไม่ (IN/NOT IN)

แสดงรหัสนักศึกษา พร้อมทั้งรายวิชา และเกรด สำหรับการลงทะเบียนที่ได้เกรด A และ B

```
SELECT sid, cid, grade
FROM enrolled
WHERE grade IN (4, 3);
```

เราสามารถแสดงข้อมูลโดยกำหนดเงื่อนไขแบบกลับกันคือแสดงข้อมูลนักศึกษาที่ไม่มีเกรดเฉลี่ยอยู่ในเซ็ตที่กำหนดโดยใช้ **NOT IN** ได้ สำหรับตัวอย่างด้านบนนั้นตรงกับการใช้ SQL แบบเปรียบเทียบ 2 การเปรียบเทียบดังนี้

```
SELECT sid, cid, grade
FROM enrolled
WHERE grade = 4 OR grade = 3;
```

อย่างไรก็ตามจะพบว่าการใช้ IN นั้นดูง่ายกว่า ซึ่งได้ผลลัพธ์ดังนี้

Enrolled

sid	cid	grade
B5075650	204203	4
B5075831	204204	3
B5075688	204204	4

ตัวอย่าง

เงื่อนไขการแสดงข้อมูลแบบตรวจความเหมือนกันของรูปแบบสายอักขระตามรูปแบบที่กำหนด (LIKE/NOT LIKE)

แสดงรายวิชาทั้งหมดที่มีคำว่า "Development" อยู่ในชื่อวิชาจากตารางรายวิชาต่อไปนี้

Courses

cid	cname	credits
202102	Information Technology I	3
202103	Information Technology II	3
204203	Software Design and Development	4
204204	Database Design and Development	4
204205	Multimedia Design and Development	4
204329	Web Technology	3

สำหรับการแสดงข้อมูลโดยการตรวจสอบความเหมือนกันของรูปแบบสายอักขระนั้นเราจะใช้ LIKE ประกอบกับเครื่องหมายต่างๆ ที่ใช้แทนรูปแบบต่างๆ ของสายอักขระ ได้แก่

% หรือเครื่องหมายร้อยละ แทนอักขระใดๆ ก็ได้มีความยาว 0 ตัวอักษรหรือมากกว่า (wildcard)

_ หรือเครื่องหมายเส้นใต้ แทนอักขระใดๆ ก็ได้มีความยาว 1 ตัวอักษรพอดี

นอกจากตัวอักษรพิเศษทั้ง 2 ตัวแล้ว ตัวอักษรอื่นๆ แทนตัวอักษรของมันเอง

พิจารณาตัวอย่างส่วนของ SQL ที่ใช้ในการตรวจสอบรูปแบบสายอักขระต่อไปนี้

cname LIKE 'I%'

หมายความว่ารายวิชาใดๆ ที่ชื่อวิชาขึ้นต้นด้วยตัวอักษร I จากนั้นจะตามด้วยตัวอักษรใดๆ ยาวกี่ตัวก็ได้ หรือไม่มีตัวอักษรตามหลังตัว I ก็ได้

cname LIKE 'I_ _ _'

หมายความว่ารายวิชาใดๆ ที่ชื่อวิชาขึ้นต้นด้วยตัวอักษร I จากนั้นตามด้วยตัวอักษรใดๆ มีความยาวต่อมาจากตัว I 3 ตัวอักษรพอดี ไม่มากหรือน้อยไปกว่านั้น ขอให้สังเกตว่าเครื่องหมายขีดเส้นใต้ทั้ง 3 ไม่มีช่องว่างใดๆ ขั้นระหว่างขีด

cname LIKE '%y'

หมายความว่ารายวิชาใดๆ ที่ชื่อวิชาลงท้ายด้วยตัวอักษร y ซึ่งก่อนหน้าตัวอักษร y เป็นตัวอักษรใดๆ มีความยาวเท่าใดก็ได้ หรือไม่มีตัวอักษรหน้าตัว y เลยก็ได้ แต่ขอให้สังเกตว่าจะต้องไม่มีตัวอักษรตามหลังจากอักษร y มิเช่นนั้นจะไม่เข้าเงื่อนไข

cname LIKE '%Development%'

หมายความว่ารายวิชาใดๆ ที่ชื่อวิชามีคำว่า Development อยู่ในส่วนใดส่วนหนึ่ง

cname NOT LIKE 'I%'

สังเกตว่าในกรณีนี้เราไม่ได้ใช้ LIKE เท่านั้น แต่มีการเพิ่ม NOT จึงหมายความว่ารายวิชาใดๆ ที่ชื่อวิชาไม่ได้ขึ้นต้นด้วยตัวอักษร I

เนื่องจากเครื่องหมาย % และ _ เป็นเครื่องหมายพิเศษ ในบางกรณีเราอาจต้องการเปรียบเทียบรูปแบบของสายอักขระที่มีเครื่องหมาย % หรือ _ เช่น เปรียบเทียบค่าของคอลัมน์กับค่า '15%' เราใช้ ESCAPE ประกอบเครื่องหมายอื่นในการแทนเครื่องหมายพิเศษ โดยส่วนของ SQL ต่อไปนี้แสดงการเปรียบเทียบค่าในคอลัมน์กับสายอักขระ '15%'

LIKE '15#%' ESCAPE '#'

ตัวอย่างที่กล่าวมาเป็นเพียงส่วนของ SQL ซึ่งเป็นเงื่อนไขในการเปรียบเทียบรูปแบบของสายอักขระ SQL ต่อไปนี้เป็นตัวอย่างของ SQL เต็มรูปแบบในการแสดงข้อมูลรายวิชาทั้งหมดที่มีคำว่า Development อยู่ในส่วนหนึ่งส่วนใดของชื่อวิชา

SELECT cid, cname, credits

FROM courses

WHERE cname LIKE '%Development%'

ได้ผลลัพธ์ดังนี้

Courses

cid	cname	credits
204203	Software Design and Development	4
204204	Database Design and Development	4
204205	Multimedia Design and Development	4

หมายเหตุ ในระบบจัดการฐานข้อมูลบางระบบอาจใช้เครื่องหมายอื่นๆ ในการแทนรูปแบบพิเศษของการตรวจสอบรูปแบบสายอักขระ เช่น ระบบจัดการฐานข้อมูล Microsoft Access ใช้เครื่องหมาย "*" และ "?" แทนเครื่องหมาย "%" และ "_" ตามลำดับ

ตัวอย่าง

เงื่อนไขการแสดงข้อมูลที่เป็นค่าว่าง (IS NULL/IS NOT NULL)

พิจารณาตารางข้อมูลการลงทะเบียนของนักศึกษาต่อไปนี้

Enrolled

cid	grade	sid
204203	4	B5075650
204204	3	B5075831
204205	1	B5075688
204329	0	B5075666
204204	4	B5075688
204329	*NULL*	B5075831

ในกรณีนี้เกรดของนักศึกษารหัส B5075831 ที่ลงทะเบียนเรียนรายวิชา 204329 ในระเบียนสุดท้ายนั้น ยังไม่ปรากฏว่าได้เกรดใดในการเรียน ค่าจึงเป็นค่าว่าง หรือค่า NULL ซึ่งหมายความว่าไม่มีค่าใดๆ เลยในคอลัมน์หรือยังไม่ทราบค่า ในกรณีนี้การเปรียบเทียบค่าในคอลัมน์นั้นๆ กับค่าอื่นๆ จึงไม่สามารถกระทำได้ เช่น ให้แสดงข้อมูลการลงทะเบียนเรียนของนักศึกษา B5075831 ที่ยังไม่ได้รับเกรด ลองพิจารณาการเปรียบเทียบเพื่อแสดงข้อมูลดังกล่าวจากส่วนของ SQL ต่อไปนี้

sid = 'B5075831' AND grade = ''

หรือ

sid = 'B5075831' AND grade <> 3

ส่วนของ SQL ทั้งสองไม่สามารถแสดงข้อมูลได้ โดยในกรณีที่ 1 '' จะไม่เท่ากับค่าว่างหรือ NULL และในกรณีที่ 2 เป็นการสื่อความหมายที่ไม่ถูกต้อง เพราะต่อไปนักศึกษาคนดังกล่าวอาจได้เกรดจากวิชาอื่นที่ไม่ใช่ 3 จะทำให้การแสดงข้อมูลไม่ถูกต้อง การทดสอบโดยใช้ 0 ยิ่งไม่ถูกต้องเพราะ 0 เป็นเกรดหนึ่งของผลการเรียน การทดสอบค่าว่างเราใช้ **IS NULL** ดังนี้

```
SELECT *
FROM courses
WHERE sid = 'B5075831' AND grade IS NULL;
```

เป็นการแสดงข้อมูลทั้งหมดในการลงทะเบียนแต่ละระเบียนของนักศึกษารหัส B5075831 ที่ยังไม่ทราบเกรด ซึ่งได้ผลลัพธ์ดังนี้

Enrolled

cid	grade	sid
204329	*NULL*	**B5075831**

8.2.2 การเรียงลำดับผลลัพธ์ (ORDER BY)

ผลลัพธ์ที่ได้จากการแสดงข้อมูลด้วยคำสั่ง SQL นั้นปกติจะไม่ได้เรียงตามลำดับใดๆ ตามข้อมูลในคอลัมน์ใดๆ เลย เว้นแต่ในบางระบบจัดการฐานข้อมูลที่ข้อมูลจะเรียงตามคีย์หลักของตารางนั้นๆ เราสามารถกำหนดให้การแสดงผลลัพธ์เรียงตามลำดับของข้อมูลในคอลัมน์ใดๆ ได้ ซึ่งสามารถที่จะเรียงลำดับจากน้อยไปมาก หรือมากไปน้อยก็ได้ โดยใช้ **ASC** และ **DESC** ตามลำดับ นอกจากนี้แล้วเราสามารถกำหนดให้ข้อมูลที่แสดงเรียงลำดับตามคอลัมน์ที่มีอยู่ในระเบียนแต่ไม่ได้ปรากฏอยู่ในผลลัพธ์ก็ได้ การเรียนลำดับนั้นเป็นส่วนหลังสุดของคำสั่ง SQL

ตัวอย่าง

จากตารางข้อมูลนักศึกษาต่อไปนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

แสดงรหัสนักศึกษาทั้งหมดเรียงตามลำดับเกรดเฉลี่ยจากมากไปน้อย

SELECT sid

FROM students

ORDER BY gpa DESC;

จากตัวอย่าง ORDER BY gpa เป็นส่วนที่กำหนดการเรียนลำดับของผลลัพธ์ DESC เป็นการกำหนดรูปแบบการเรียนลำดับตาม gpa จากมากไปน้อย มาจากคำว่า descending ซึ่งตรงข้ามกับการใช้รูปแบบ ASC ที่มาจากคำว่า ascending ซึ่งเป็นการเรียนลำดับจากน้อยไปมาก จากตัวอย่างได้ผลลัพธ์ดังนี้

Students

sid	gpa
B5075650	3.82
B5075666	3.44
B5075688	3.21
B5075832	2.00
B5075831	1.80

ผลลัพธ์ที่ได้จากตัวอย่างเป็นตารางข้อมูลที่มีเพียงคอลัมน์เดียว แสดงเฉพาะรหัสนักศึกษา สำหรับคอลัมน์ที่เป็นสีเทานั้นไม่ได้ปรากฏในผลลัพธ์แต่แสดงในที่นี้เพื่อให้เห็นว่าผลลัพธ์ที่แสดงเรียงตามลำดับเกรดเฉลี่ยของนักศึกษาแต่ละคนจากมากไปหาน้อย โดยเกรดเฉลี่ยไม่จำเป็นต้องปรากฏในผลลัพธ์ก็ได้

ตัวอย่าง

การเรียงลำดับข้อมูลโดยกำหนดค่าในคอลัมน์ที่ต้องการเรียงลำดับหลายคอลัมน์

แสดงข้อมูลนักศึกษาทั้งหมดเรียงตามชื่อนักศึกษาตามด้วยอายุจากมากไปหาน้อย

SELECT *

FROM students

ORDER BY name, age DESC;

ได้ผลลัพธ์ดังนี้

Students

cid	sid	name	login	age	gpa
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
3100904022132	B5075666	สมชาย	somchai@it	18	3.44

จากผลลัพธ์ ข้อมูลจะเรียงตามชื่อนักศึกษา จากมากไปน้อย ซึ่งสมศักดิ์จะมาเป็นลำดับสุดท้ายตามพจนานุกรม ในขณะที่สมชายจะมาก่อน จากข้อมูลที่มีทั้งหมด อย่างไรก็ตามบ่อยครั้งที่มักมีค่าที่สามารถซ้ำกันได้ในคอลัมน์ เราสามารถกำหนดเพิ่มเติมได้อีกว่าในกรณีที่ค่าในคอลัมน์ที่ใช้เรียงลำดับซ้ำกัน ต่อไปจะให้เรียงลำดับตามคอลัมน์ใด ในตัวอย่างนี้เรากำหนดให้ใช้อายุเป็นคอลัมน์ถัดมาที่ใช้ในการเรียงลำดับ ซึ่งสังเกตได้จากสมศรีที่มี 2 คน สมศรีที่มีอายุ 19 ปีมาก่อนสมศรีที่มีอายุ 18 ปี

8.2.3 การดำเนินการเป็นชุด (SQL Aggregate Functions)

การดำเนินการเป็นชุดในคำสั่ง SQL อาศัยฟังก์ชันต่างๆ ในลักษณะเดียวกันกับการดำเนินการเป็นชุดในพีชคณิตเชิงสัมพันธ์ ซึ่ง International Standard Organization (ISO) ได้กำหนดฟังก์ชันมาตรฐานไว้ดังนี้

COUNT—นับจำนวนของค่าในคอลัมน์	AVG—หาค่าเฉลี่ยของค่าในคอลัมน์
SUM—หาผลรวมของค่าในคอลัมน์	MIN/MAX—หาค่าต่ำสุด/สูงสุดของคอลัมน์

โดยมีข้อกำหนดเพิ่มเติมดังนี้

- ฟังก์ชันการดำเนินการแบบชุดทั้งหมดสามารถดำเนินการได้กับคอลัมน์ 1 คอลัมน์เท่านั้น
- สำหรับฟังก์ชัน **COUNT, MIN, MAX** ไม่จำเป็นต้องดำเนินการกับค่าที่เป็นตัวเลขก็ได้
- การใช้ **COUNT(*)** นั้น เป็นการนับระเบียนทุกระเบียน ซึ่งรวมระเบียนที่มีค่า NULL ซึ่งโดยปกติการใช้ COUNT กับคอลัมน์ใดๆ จะไม่นับระเบียนที่คอลัมน์ที่กำหนดมีค่าเป็น NULL ซึ่ง COUNT(*) นั้นโดยแท้จริงแล้วคือการนับระเบียนของคอลัมน์ใดๆ แต่ละระเบียนจะต้องมีอย่างน้อย 1 คอลัมน์ที่ไม่มีค่าเป็น NULL จึงทำให้ COUNT(*) นับระเบียนที่มีค่า NULL ด้วย
- เราสามารถใช้ **DISTINCT** ประกอบฟังก์ชัน โดยวาง DISTINCT ไว้ก่อนหน้าชื่อคอลัมน์ที่จะดำเนินการเพื่อขจัดระเบียนที่มีค่าซ้ำซ้อนออกก่อน
- เราสามารถใช้ **ALL** ประกอบฟังก์ชัน เพื่อให้ฟังก์ชันดำเนินการกับระเบียนทุกระเบียนแม้จะมีค่าซ้ำกัน ซึ่งปกติฟังก์ชันดำเนินการเสมือนมี ALL อยู่แล้ว
- DISTINCT ไม่มีผลกับฟังก์ชัน MIN และ MAX แต่ผลมีกับ SUM และ AVG
- DISTINCT ใช้ได้เพียง 1 ครั้งในฟังก์ชันดำเนินการเป็นชุด
- ฟังก์ชันการดำเนินการเป็นชุดใช้ได้กับคำสั่ง **SELECT** และส่วนของคำสั่ง **HAVING** เท่านั้น

ซึ่งสามารถยกตัวอย่างการใช้งานฟังก์ชันต่างๆ ดังด้านล่างด้วยข้อมูลของนักศึกษาดังตารางต่อไปนี้

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

Enrolled

cid	grade	sid
204203	4	**B5075650**
204204	3	**B5075831**
204205	1	**B5075688**
204329	0	**B5075666**
204204	4	**B5075688**
204329	*NULL*	**B5075831**

Rooms

rno	address	capacity
1101	Lecture Building	30
1102	Lecture Building	30
1201	Lecture Building	120
2101	Lecture Building	300
3101	Lecture Building	300
4101	Lecture Building	1500

ตัวอย่าง

การนับระเบียนทั้งหมด **COUNT(*)**

SELECT COUNT(*) AS myCount FROM students;

เป็นการนับจำนวนนักศึกษาทั้งหมดในตารางได้ผลลัพธ์ดังนี้

Students

myCount
5

ตัวอย่าง

การนับระเบียนที่มีค่าไม่ซ้ำกัน **COUNT(DISTINCT)**

SELECT COUNT(cid) AS enrolledCount, COUNT(DISTINCT cid) AS courseCount

FROM enrolled;

COUNT(cid) เป็นการนับจำนวนรายวิชาทั้งหมดที่ทราบค่า และ COUNT(DISTINCT cid) เป็นการนับจำนวนรายวิชาทั้งหมดที่มีเคยมีนักศึกษาลงทะเบียน โดยตัดรหัสวิชาที่ซ้ำกันให้เหลือเพียง 1 ระเบียนในการนับ จากตารางการลงทะเบียน ได้ผลลัพธ์ดังนี้

Students

enrolledCount	courseCount
6	4

ผลลัพธ์ที่ได้เป็นตาราง 2 คอลัมน์ ซึ่ง enrolledCount เกิดจาก COUNT(cid) เป็นการนับรายวิชาทั้งหมดแม้จะซ้ำกันส่วน courseCount เกิดจาก COUNT(DISTINCT cid) ได้ค่าเพียง 4 เนื่องจากรายวิชา 204204 และ 204329 ปรากฏอยู่ในตารางการลงทะเบียนรายวิชาละ 2 ระเบียน เมื่อตัดระเบียนที่ซ้ำกันออกจากการใช้ DISTINCT จึงเหลือเพียง 4 ระเบียน

ตัวอย่าง

การนับและการหาผลรวม **COUNT** และ **SUM**

SELECT COUNT(*) AS roomCount, SUM(capacity) AS seatCount

FROM rooms

จากตัวอย่าง COUNT(*) เป็นการนับจำนวนระเบียนที่มีอยู่ในตารางห้องซึ่งผลลัพธ์จะปรากฏอยู่ในคอลัมน์ roomCount ซึ่งเป็นการนับจำนวนห้องนั่นเอง ส่วน SUM(capacity) เป็นการหาผลรวมของความจุทุกๆ ห้อง ได้เป็นที่นั่งรวมทั้งหมด ดังผลลัพธ์ต่อไปนี้

Students

roomCount	seatCount
6	2280

ตัวอย่าง

การหาค่าน้อยที่สุด มากที่สุด และค่าเฉลี่ย **MIN, MAX** และ **AVG**

SELECT MIN(gpa) AS minGpa, MAX(gpa) AS maxGpa, AVG(gpa) AS avgGpa

FROM students;

ได้ผลลัพธ์ดังนี้

Students

minGpa	maxGpa	avgGpa
1.80	3.82	2.854

8.2.4 การจัดกลุ่ม (Grouping)

ข้อคำถามแบบกลุ่มนั้นเราจะใช้ **GROUP BY** ในการจัดกลุ่ม โดยระบุกลุ่มของการดำเนินการตามคอลัมน์ ซึ่งค่าที่ได้จากการดำเนินการจะเป็นค่าเดียวสำหรับแต่ละกลุ่ม

ตัวอย่าง

การใช้ GROUP BY

แสดงรหัสวิชาและจำนวนผู้ลงทะเบียนในแต่ละรายวิชา

SELECT cid, COUNT(cid) AS studentCount

FROM enrolled

GROUP BY cid;

จากตัวอย่างมีการใช้ฟังก์ชันการดำเนินการแบบกลุ่มได้แก่ COUNT ซึ่งเป็นการนับจำนวนรหัสวิชาจากตารางลงทะเบียน ในกรณีนี้มีการใช้ GROUP BY cid เป็นการนับจำนวนรายวิชาแล้วแสดงผลลัพธ์แยกเป็นผลรวมของจำนวนระเบียนของแต่ละรายวิชา ได้ผลลัพธ์ดังนี้

Enrolled

cid	studentCount
204203	1
204204	2
204205	1
204329	2

การบังคับการจัดกลุ่ม (Restricing groupings—HAVING clause)

ในการนีที่มีการแสดงข้อมูลโดยการจับกลุ่มของข้อมูล เราสามารถกรองกลุ่มของข้อมูลโดยกำหนดเงื่อนไขที่แต่ละกลุ่มจะต้องคุณสมบัติตามที่กำหนดได้ โดยใช้ **HAVING** ซึ่งมีลักษณะคล้ายคลึงกันกับการใช้ **WHERE** เพื่อกรองแถวของข้อมูล แต่ **HAVING** นี้จะใช้กับ **GROUP BY** ในการกรองกลุ่มของข้อมูล

ในส่วนของประโยค HAVING นี้จะต้องมีการใช้คำสั่งการดำเนินการแบบเป็นชุด (aggregate function) อย่างน้อย 1 คำสั่ง มิเช่นนั้นจะไม่ได้เป็นการกรองข้อมูลแบบเป็นกลุ่ม หากเป็นเช่นนั้นเงื่อนไขการค้นหาและกรองข้อมูลใน HAVING ต้องย้ายไปอยู่ในส่วนของประโยค WHERE เพื่อกรองระเบียนข้อมูล และให้ระลึกว่าเราไม่ใช้การดำเนินการแบบเป็นชุดในส่วนของประโยค WHERE

ตัวอย่าง

การใช้ **HAVING**

SELECT age, MAX(gpa) AS maxGpa

FROM students

GROUP BY age HAVING MAX(gpa) >= 3.25;

จากตัวอย่างเป็นการแสดงข้อมูลนักศึกษาโดยแสดงอายุกับคะแนนสูงสุดของนักศึกษาจำแนกตามอายุนักศึกษาโดยการจำแนกตามอายุนั้นได้แก่การใช้ GROUP BY age นอกจากนี้ยังระบุด้วยว่าแสดงเฉพาะกลุ่มอายุที่มีคะแนนสูงสุดมากกว่าหรือเท่ากับ 3.25 ซึ่งได้แก่การใช้ HAVING MAX(gpa) >= 3.25 ได้ผลลัพธ์ดังนี้

Students

age	gpa
18	3.44
19	3.82

8.2.5 ข้อคำถามรอง (Subqueries)

ข้อคำถามรองหรือข้อคำถามย่อยเป็นการใช้คำสั่ง SELECT ซ้อนคำสั่ง SELECT อีกทีหนึ่งเพื่อให้คำสั่ง SELECT หลักสมบูรณ์ โดยเราเรียกคำสั่ง SELECT ที่เป็นคำสั่งรองอยู่ภายใต้คำสั่ง SELECT อื่นอีกทีหนึ่งว่า **Inner SELECT** หรือ **SubSELECT** ซึ่งจะนำไปทำให้คำสั่งหลักหรือคำสั่ง SELECT ที่อยู่ภายนอกสมบูรณ์ซึ่งเราเรียกคำสั่ง SELECT หลักนี้ว่า **Outer SELECT**

สำหรับการใช้งานคำสั่ง SELECT ย่อยนี้เราจะใช้ประกอบ WHERE และ HAVING ใน Outer SELECT ซึ่งเราจะเรียกประโยค WHERE และ HAVING ที่บรรจุ Inner SELECT นี้ว่าข้อคำถามรอง (subqury หรือ nested query) ซึ่งสามารถใช้ได้ในคำสั่ง **INSERT, UPDATE** และ **DELETE** ได้เช่นเดียวกัน

ซึ่งข้อคำถามย่อยนี้มีหลายประเภทจำแนกตามรูปแบบของผลลัพธ์ของข้อคำถามย่อยนั้น ดังนี้

1) **Scalar subquery** ผลลัพธ์ที่ได้เป็นค่าสเกลาร์ เป็นค่าเดี่ยว อยู่ในตาราง 1 แถวและ 1 คอลัมน์เท่านั้น
2) **Row subquery** ผลลัพธ์ที่ได้เป็นลักษระของแถว 1 แถว มีหลายคอลัมน์
3) **Table subquery** ผลลัพธ์ที่ได้เป็นตาราง 1 หรือหลายคอลัมน์ หลายแถว เช่นการนำไปใช้ประโยชน์ในการใช้ IN

ตัวอย่าง

ข้อคำถามย่อยโดยเปรียบเทียบความเท่ากัน

```
SELECT sid, grade
FROM  enrolled
WHERE cid = (SELECT cid
        FROM courses
        WHERE cname = 'Database Design and Development'
```

จากตัวอย่างเป็นการแสดงข้อมูลรหัสนักศึกษาและเกรดที่ได้ของรายวิชาที่มีค่าเท่ากับข้อคำถามย่อย ซึ่งข้อคำถามย่อยในที่นี้คือการค้นหารหัสวิชาที่มีชื่อวิชาว่า 'Database Design and Development' ตัวอย่างข้างต้นจึงมีค่าเท่ากับการใช้คำสั่ง

```
SELECT sid, grade
FROM enrolled
WHERE cid = '204204'
```

ได้ผลลัพธ์ดังนี้

Enrolled

sid	grade
B5075831	3
B5075688	4

เราสามารถมองผลลัพธ์ที่ได้จากการใช้ข้อคำถามย่อยว่าเป็นตารางชั่วคราวก็ได้ ซึ่งนอกจากการเปรียบความความเท่ากันโดยใช้เครื่องหมาย "=" แล้ว เรายังสามารถใช้เครื่องหมายเปรียบเทียบอื่นๆ ได้แก่ <, >, <=, >=, <> ได้ ขอให้ระลึกไว้เสมอว่าข้อคำถามย่อยใช้ในประโยค WHERE หรือ HAVING เท่านั้น โดยอาศัยเครื่องหมายวงเล็บ "(...)" เพื่อครอบข้อคำถามย่อยไว้

ตัวอย่าง

การใช้ข้อคำถามย่อยกับฟังก์ชันการดำเนินการแบบเป็นชุด

SELECT sid, name, gpa - (SELECT AVG(gpa) FROM students) AS gpaDiff

FROM students

WHERE gpa > (SELECT AVG(gpa) FROM students);

จากตัวอย่างเป็นการแสดงข้อมูลรหัสนักศึกษา ชื่อนักศึกษา และ gpa – (SELECT AVG(gpa) FROM students) คือการแสดงข้อมูลผลต่างระหว่าง gpa ของนักศึกษาแต่ละคนกับ gpa เฉลี่ยของนักศึกษาทั้งหมด ทั้งนี้ในประโยค WHERE เป็นการระบุว่าให้แสดงเฉพาะข้อมูลของนักศึกษาที่มี gpa มากกว่า gpa เฉลี่ยของนักศึกษาทั้งหมด ซึ่งมีค่าเท่ากับการใช้คำสั่ง SQL ต่อไปนี้

SELECT sid, name, gpa – 2.854 AS gpaDiff

FROM students

WHERE gpa > 2.854;

ได้ผลลัพธ์ดังต่อไปนี้

Students

sid	name	gpaDiff
B5075666	สมชาย	0.586
B5075688	สมศรี	0.356
B5075650	สมศรี	0.966

เกณฑ์ในการใช้ข้อคำถามย่อยสามารถสรุปเป็นกฎ 4 ข้อได้ดังนี้

1) ORDER BY จะไม่ใช้ในข้อคำถามย่อย แต่จะใช้ได้กับ Outer SELECT ที่อยู่ชั้นนอกสุดเท่านั้น
2) การใช้ข้อคำถามย่อยเพื่อให้ Outer SELECT เปรียบเทียบรายการของข้อมูลที่เกิดจาก Inner SELECT นั้น Inner SELECT จะต้องมีการแสดงข้อมูลอย่างน้อย 1 คอลัมน์หรือ 1 นิพจน์ ยกเว้นการตรวจสอบโดยการใช้ EXISTS

3) การอ้างถึงคอลัมมน์ใดๆ ในข้อคำถามย่อยจะเป็นการอ้างถึงคอลัมน์จากตารางใน FROM ของข้อคำถามย่อยนั้นเอง กรณีที่ต้องการอ้างถึงคอลัมน์จากตารางของ Outer FROM ต้องมีการกำหนดชื่อคอลัมน์ (qualifying column name) ใน Outer FROM เพื่อนำมาอ้างถึงในข้อคำถามย่อย

4) ข้อคำถามย่อยจะต้องอยู่ทางด้านขวาของตัวดำเนินเปรียบเทียบเสมอ ยกตัวอย่างเช่น

SELECT sid, name, gpa - (SELECT AVG(gpa) FROM students) AS gpaDiff

FROM students

WHERE (SELECT AVG(gpa) FROM students) < gpa;

เป็นการสร้างคำสั่ง SQL ที่ไม่ถูกต้องเนื่องจากข้อคำถามย่อยใน WHERE นั้นอยู่ทางด้านซ้ายของเครื่องหมายเปรียบเทียบ "<"

ตัวอย่าง

การใช้ข้อคำถามย่อยกับ IN (Nested IN)

SELECT sid, name, gpa

FROM students

WHERE sid IN(SELECT sid

FROM enrolled

WHERE cid = '204204')

จากตัวอย่างเป็นการแสดงข้อมูลรหัสนักศึกษา ชื่อนักศึกษา และเกรดเฉลี่ยของนักศึกษาที่ลงทะเบียนเรียนรายวิชา 204204 ซึ่งในส่วนของ WHERE sid IN หมายถึงการแสดงข้อมูลของนักศึกษาที่มีรหัสนักศึกษาปรากฏอยู่ในรายการของการ SELECT ในข้อคำถามย่อย สำหรับข้อคำถามย่อย (SELECT sid FROM students WHERE cid = '204204') เป็นการแสดงรายการรหัสนักศึกษาทั้งหมดที่ลงทะเบียนรายวิชา 204204 ได้ผลลัพธ์ดังนี้

Students

sid	name	gpa
B5075688	สมศรี	3.21
B5075831	สมศักดิ์	1.80

คำสั่งที่ได้ยกตัวอย่างสามารถเขียนให้อยู่ในรูปของการ JOIN กันของตารางได้ดังนี้

SELECT students.sid, name, gpa

FROM students, enrolled

WHERE students.sid = enrolled.sid

AND cid = '204204'

ซึ่งจะได้อธิบายในหัวข้อที่ 8.2.7 ข้อคำถามหลายตาราง

8.2.6 มีบ้าง และ ทั้งหมด (ANY and ALL)

ANY และ ALL เป็นการใช้งานข้อคำถามย่อยเพื่อกำหนดเงื่อนไขในการแสดงข้อมูล โดยคำสั่ง SQL ที่มีการใช้ ANY หรือ ALL จะแสดงข้อมูลเฉพาะแถวที่มีค่าของคอลัมน์เป็นไปตามเงื่อนไขที่กำหนดในทำนองเดียวกับการใช้เครื่องหมายเปรียบเทียบค่า แต่ ANY และ ALL นำมาใช้ประกอบการทดสอบค่าของคอลัมน์หนึ่ง กับชุดของค่าที่นำมาเปรียบเทียบกับค่าทางด้านซ้ายของเครื่องหมายเปรียบเทียบ โดยค่าของนิพจน์จะเป็นจริงสำหรับ ANY ก็ต่อเมื่อ การผลทดสอบโดยเครื่องหมายเปรียบเทียบระหว่างค่าทางซ้าย กับแต่ละค่าในชุดข้อมูลทางด้านขวาของเครื่องหมายเปรียบเทียบเป็นจริงอย่างน้อยหนึ่งค่า ANY สามารถใช้คำว่า SOME แทนได้ สำหรับ ALL ค่าของนิพจน์จะเป็นจริงก็ต่อเมื่อการเปรียบเทียบทุกๆ ค่าระหว่างค่าทางซ้ายของเครื่องหมายเปรียบเทียบกับทุกค่าในชุดของข้อมูลทางขวาของเครื่องหมายเปรียบเทียบเป็นจริงทั้งหมดทุกกรณี

ANY และ ALL ใช้กับข้อคำถามย่อยแสดงข้อมูลผลลัพธ์เป็นข้อมูล 1 คอลัมน์เท่านั้น เนื่องจากคำสั่ง SQL ที่ใช้งาน ANY และ ALL มีความซับซ้อน จึงจำเป็นต้องอาศัยตัวอย่างเพื่อแสดงการใช้งานดังนี้

ตัวอย่าง

แสดงรหัสนักศึกษา ชื่อนักศึกษา และเกรดเฉลี่ย สำหรับนักศึกษาที่มีเกรดเฉลี่ยสูงกว่าหรือเท่ากับเกรดของนักศึกษาอย่างน้อย 1 คนที่ลงทะเบียนเรียนรายวิชา 204204

```
SELECT sid, name, gpa
FROM students
WHERE gpa >= SOME (SELECT grade
        FROM enrolled
        WHERE cid = '204204');
```

จากตัวอย่าง คำสั่ง SQL ในส่วน **SELECT sid, name, gpa FROM students WHERE …** เป็นข้อคำถามหลักแสดงข้อมูลรหัสนักศึกษา ชื่อนักศึกษา และเกรดเฉลี่ยจากตารางนักศึกษา ซึ่งเป็นไปตามเงื่อนไขของ **WHERE gpa > SOME (…)** สำหรับเงื่อนไขดังกล่าวเป็นการทดสอบว่าเกรดเฉลี่ยของนักศึกษานั้นมากกว่าหรือเท่ากับค่าใดๆ ในชุดข้อมูลด้านขวาของเครื่องหมาย ">=" อย่างน้อย 1 ค่าหรือไม่ ซึ่งค่าทางด้านขวาของเครื่องหมายมากกว่าหรือเท่ากับนั้นคือเกรดของรายวิชา 204204 (ขอให้สังเกตว่าเกรดของรายวิชา 204204 นั้นเป็นเกรดที่ได้จากการลงทะเบียนเรียนรายวิชานั้นโดยเฉพาะ ไม่ใช่เกรดเฉลี่ย) ได้มาจากข้อคำถามย่อย **SELECT grade FROM enrolled WHERE cid = '204204'** ลองพิจารณาดูกรณีตัวอย่างข้อมูลดังแสดงด้านล่าง ผลลัพธ์ของข้อคำถามย่อยนี้เป็นการแสดงเกรดหรือระดับคะแนนตัวเลขทั้งหมดของผู้ลงทะเบียนเรียนรายวิชา 204204 ซึ่งมี 2 เกรดได้แก่ 3 และ 4 ซึ่งเป็นเกรดของนักศึกษารหัส B5075831 และ B5075688 ตามลำดับ

จากนั้นเรานำค่าระดับคะแนน {3, 4} (ซึ่งเป็นข้อมูล 1 คอลัมน์เท่านั้น เป็นไปตามรูปแบบของการใช้ ANY/SOME) มาเปรียบเทียบกับเกรดเฉลี่ยของนักศึกษาแต่ละคน เสมือนมีการตรวจสอบด้วยคำสั่ง SQL ดังนี้

WHERE gpa > 3 OR WHERE gpa > 4

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

Enrolled

cid	grade	sid
204203	4	B5075650
204204	3	B5075831
204205	1	B5075688
204329	0	B5075666
204204	4	B5075688
204329	*NULL*	B5075831

Results

grade
3
4

ดังนั้นเมื่อเกรดเฉลี่ยของนักศึกษาคนใด มากกว่าหรือเท่ากับระดับคะแนนของนักศึกษาในรายวิชา 204204 อย่างน้อยหนึ่งคน ข้อมูลของนักศึกษาคนนั้นๆ จะแสดงในผลลัพธ์ จากกรณีตัวอย่างข้อมูล สมชาย 1 คน และสมศรีอีก 2 คนรวมเป็น 3 คนนี้ต่างก็มีเกรดเฉลี่ยสูงกว่าระดับคะแนนที่นักศึกษาคนใดๆ ได้รับจากรายวิชา 204204 นั่นคืออย่างน้อยก็มากกว่านักศึกษาที่ได้เกรด 3 เนื่องจากทั้ง 3 คนมีเกรดเฉลี่ยสูงกว่า 3 ทั้งสิ้น ถึงแม้จะไม่ได้มากกว่าหรือเท่าเกรด 4 เงื่อนไขที่ใช้ SOME นี้ก็ยังเป็นจริง ในขณะที่สมศักดิ์และสมน้ำหน้าไม่ได้มีเกรดเฉลี่ยสูงกว่าหรือเท่ากับเกรดใดๆ ของรายวิชา 204204 เลย จึงได้ผลลัพธ์ดังนี้

Student

sid	name	gpa
B5075666	สมชาย	3.44
B5075688	สมศรี	3.21
B5075650	สมศรี	3.82

จากตัวอย่างเราอาจนำไปใช้ประโยชน์เช่นรายวิชา 204204 เป็นรายวิชาที่นักศึกษามักจะได้เกรดสูง หากนักศึกษาคนใดมีเกรดเฉลี่ยสูงกว่านักศึกษาคนใดๆ ที่ได้เกรดจากรายวิชานี้จะปรากฏรายชื่อในผลลัพธ์ อาจนำรายชื่อไปใช้ในการคัดเลือกนักศึกษาเพื่อส่งนักศึกษาเข้าแข่งขันการตอบปัญหาทางวิชาการ เป็นต้น

ตัวอย่าง

แสดงรหัสนักศึกษา ชื่อนักศึกษา และเกรดเฉลี่ย สำหรับนักศึกษาที่มีเกรดเฉลี่ยสูงกว่าหรือเท่ากับเกรดของนักศึกษาทุกคนที่ลงทะเบียนเรียนรายวิชา 204204

```
SELECT sid, name, gpa
FROM students
WHERE gpa >= ALL (SELECT grade
        FROM enrolled
        WHERE cid = '204204');
```

ตัวอย่างนี้สามารถอธิบายได้ในทำนองเดียวกันกับตัวอย่างการใช้ ANY/SOME แต่แตกต่างกันตรงที่เงื่อนไขของข้อคำถามย่อย **WHERE gpa >= ALL (…)** จะเป็นจริงก็ต่อเมื่อค่าทางซ้ายมือของเครื่องหมาย ">=" จะต้องมากกว่าหรือเท่ากับค่าทุกๆ ค่าในชุดข้อมูลทางด้านขวาของเครื่องหมาย ">=" ซึ่งจะต้องมากกว่า 3 และ 4 ซึ่งการใช้ข้อคำถามย่อยดังกล่าวเสมือนแสดงข้อมูลจากคำสั่ง SQL ดังนี้

```
WHERE gpa >= 3 AND gpa >= 4
```

หากลองพิจารณาดูเกรดเฉลี่ยของนักศึกษาทั้งหมดจะพบว่ามีบางส่วนที่มีเกรดเฉลี่ยมากกว่า 3 แต่มีมีใครเลยที่มีเกรดเฉลี่ยสูงกว่าหรือเท่ากับ 4 ดังนั้นจึงไม่ปรากฏผลลัพธ์เป็นระเบียนข้อมูลของนักศึกษาคนใดเลย เราอาจใช้ประโยชน์จากคำสั่ง SQL ดังกล่าวเพื่อคัดเลือกนักศึกษาที่มีเกรดเฉลี่ยสูงมากๆ เพื่อนำมาติวรายวิชาดังกล่าว เป็นต้น

8.2.7 Multi-Table Queries

โดยปกติแล้วข้อมูลที่นำมาใช้ประโยชน์จากระบบจัดการฐานข้อมูลเชิงสัมพันธ์มักจะเป็นข้อมูลที่เกิดจากตารางข้อมูลหลายๆ ตาราง เช่น นักศึกษาคนใด ได้เกรดเท่าไร สำหรับรายวิชาที่ระบุ โดยแสดงรายละเอียดชื่อนักศึกษาและชื่อวิชา ซึ่งจะต้องอาศัยข้อมูลจากตารางนักศึกษา ตารางการลงทะเบียน และตารางรายวิชาประกอบกัน การแสดงข้อมูลจากตารางหลายๆ ตารางนี้เป็นการดำเนินการโดยปกติของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ เนื่องจากข้อมูลแต่ละเอนทิตีมีความสัมพันธ์กัน ตลอดจนข้อมูลที่ซ้ำซ้อนกัน เราได้แยกออกเป็นตารางๆ ซึ่งเป็นการดำเนินการที่ได้จากการทำให้เป็นรูปแบบบรรทัดฐาน

การแสดงข้อมูลจากตารางหลายๆ ตารางนั้นสามารถทำได้โดยอาศัยการดำเนินการ JOIN ตารางเช่นเดียวกันกับการดำเนินการ JOIN ในพีชคณิตเชิงสัมพันธ์

Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

Faculty (*fid:* string, *fname:* string, *sal:* real)

Courses (*cid:* string, *cname:* string, *credits:* integer)

Rooms (*rno:* integer, *address:* string, *capacity:* integer)

Enrolled (*sid:* string, *cid:* string, *grade:* float)

Teaches (*fid:* string, *cid:* string)

Meets_In (*cid:* string, *rno*: integer, *time:* string)

ตัวอย่าง การเชื่อมตารางอย่างง่าย (Simple JOIN)

แสดงรหัสอาจารย์ ชื่ออาจารย์ และรหัสรายวิชาที่อาจารย์แต่ละคนสอน

```
SELECT faculty.fid, fname,cid
FROM faculty, teaches
WHERE faculty.fid = teaches.fid;
```

จากตัวอย่างเป็นการแสดงข้อมูล 3 คอลัมน์จากราตารางอาจารย์และการสอน สำหรับ **WHERE faculty.fid = teaches.fid** นั้นเป็นการระบุคอลัมน์ที่เชื่อมตารางทั้งสองเข้าด้วยกันนั่นเอง ซึ่งระบบจัดการฐานข้อมูลจะนำแถวใดๆ ในตารางหลังมาเชื่อมต่อกับแถวในตารางแรกในกรณีที่มีค่าในคอลัมน์ที่ระบุเท่ากัน ในกรณีนี้คือคอลัมน์ fid แสดงการเชื่อมตารางกัน และได้ผลลัพธ์ดังนี้

Faculty

fid	fname	sal
001	แพนเค้ก	40,000
007	สถิตย์	50,000
069	พอลล่า	30,000

Teaches

fid	cid
001	204205
007	204203
007	204204
007	204329
069	202102
069	202103

ผลลัพธ์

fid	fname	cid
001	แพนเค้ก	204205
007	สถิตย์	204203
007	สถิตย์	204204
007	สถิตย์	204329
069	พอลล่า	202102
069	พอลล่า	202103

สำหรับคำสั่ง SQL ที่แสดงข้างต้น เมื่อเราทำการเชื่อมตาราง คอลัมน์ fid ที่ใช้เป็นตัวเชื่อมตารางจะมีค่าซ้ำกัน คือค่า fid จากตาราง faculty และ fid จากตาราง teaches ซึ่งทำให้เกิดการกำกวมขึ้น เราจึงมีความจำเป็นต้องระบุว่าจะแสดง fid จากตารางใด ดังคำสั่งตัวอย่าง **SELECT faculty.fid, ...** ในบางกรณีตารางที่นำมาเชื่อมกันอาจมีชื่อของคอลัมน์ที่ซ้ำกัน แม้จะไม่ได้เป็นคอลัมน์ที่ใช้ในการเชื่อมตารางก็ตาม การระบุตารางเจ้าของคอลัมน์ก็มีความจำเป็นเช่นเดียวกัน ซึ่งเราสามารถเขียนคำสั่ง SQL ตามตัวอย่างให้กระชับขึ้นอีก ได้ดังนี้

```
SELECT f.fid, fname,cid
FROM faculty f, teaches t
WHERE f.fid = t.fid;
```

โดยการแทนตาราง faculty ด้วยการใช้ตัวแปรอื่นๆ ในการแทนตาราง ในกรณีนี้คือการแทน faculty ด้วย f และ teaches ด้วย t จากนั้นเราสามารถแทน faculty และ teaches ที่ปรากฏอยู่ในคำสั่ง SQL ด้วย f และ t ได้ตามลำดับ นอกจากนี้เราสามารถใช้ JOIN ในการดำเนินการเชื่อมตารางได้แทนคำสั่ง SQL ข้างต้น ซึ่งให้ผลลัพธ์เช่นเดียวกันดังนี้

```
SELECT f.fid, fname,cid
FROM faculty f JOIN teaches t ON f.fid = t.fid;
```

และเนื่องจากคอลัมน์ที่เชื่อมกันนั้นเป็นคอลัมน์ที่มีชื่อเหมือนกัน เราสามารถเขียน SQL ได้ในอีกรูปแบบหนึ่งดังนี้

```
SELECT f.fid, fname,cid
FROM faculty f JOIN teaches t USING fid;
```

ในกรณีที่การเชื่อมตารางใช้คอลัมน์ที่เป็นชื่อเดียวกันและเปรียบเทียบค่าเท่ากัน เป็นการดำเนินการในลักษณะของ NATURAL JOIN ในพีชคณิตเชิงสัมพันธ์นั่นเอง ซึ่งสามารถเขียนในรูปของคำสั่ง SQL ดังนี้

```
SELECT faculty.fid, fname,cid
FROM faculty NATURAL JOIN teaches;
```

ตัวอย่าง การเชื่อมตารางและเรียงลำดับข้อมูลในผลลัพธ์

แสดงรหัสอาจารย์ ชื่ออาจารย์ และรหัสรายวิชาที่อาจารย์แต่ละคนสอน โดยเรียงลำดับตามรหัสวิชา

```
SELECT faculty.fid, fname,cid
FROM faculty, teaches
WHERE faculty.fid = teaches.fid
ORDER BY t.cid;
```

เราใช้ **ORDER BY** ในการเรียงลำดับซึ่ง ORDER BY จะต้องวางไว้หลังจาก WHERE เช่นเดียวกันกับการใช้ ORDER BY โดยทั่วไป โดยหากมีการกำกวมในชื่อคอลัมน์ที่ใช้ในการเรียงลำดับ ต้องระบุเจ้าของตารางของคอลัมน์นั้นๆ ด้วย

ผลลัพธ์

fid	fname	cid
069	พอลล่า	202102
069	พอลล่า	202103
007	สถิตย์	204203
007	สถิตย์	204204
001	แพนเค้ก	204205
007	สถิตย์	204329

ตัวอย่าง การเชื่อมตาราง 3 ตาราง (Three-table JOIN)

แสดงรหัสอาจารย์ ชื่ออาจารย์ รหัสวิชา และชื่อวิชาที่อาจารย์แต่ละคนสอน

```
SELECT f.fid, fname, c.cid, cname
FROM faculty f, teaches t, courses c
WHERE f.fid = t.tid AND t.cid = c.cid
ORDER BY f.fid, c.cid;
```

จากตัวอย่างคำสั่ง SQL แสดงข้อมูลจากตาราง 3 ตาราง โดยเชื่อมตาราง Faculty, Teaches และ Courses เข้าด้วยกัน ลองพิจารณาข้อมูลแถวแรกของตาราง Faculty อาจารย์แพนเค้กมีรหัสอาจารย์เป็น 001 ซึ่งเมื่อพิจารณาในตาราง Teaches เราพบว่ารหัสอาจารย์ 001 สอนรายวิชา 204205 และเมื่อนำค่ารหัสวิชา 204205 มาเปรียบเทียบในตาราง Courses พบว่าเป็นรายวิชา Multimedia Design and Development จึงได้ข้อมูลครบทั้งหมดที่ต้องการแสดงสำหรับอาจารย์คนแรก ซึ่งการเชื่อมตารางและผลลัพธ์ทั้งหมดแสดงได้ดังต่อไปนี้

Faculty

fid	fname	sal
001	แพนเค้ก	40,000
007	สถิตย์	50,000
069	พอลล่า	30,000

Teaches

fid	cid
001	204205
007	204203
007	204204
007	204329
069	202102
069	202103

Courses

cid	cname	credits
202102	Information Technology I	3
202103	Information Technology II	3
204203	Software Design and Development	4
204204	Database Design and Development	4
204205	Multimedia Design and Development	4
204329	Web Technology	3

ผลลัพธ์

fid	fname	cid	cname
001	แพนเค้ก	204205	Multimedia Design and Development
007	สถิตย์	204203	Software Design and Development
007	สถิตย์	204204	Database Design and Development
007	สถิตย์	204329	Web Technology
069	พอลล่า	202102	Information Technology I
069	พอลล่า	202103	Information Technology II

เราสามารถใช้คำสั่ง **JOIN** ในการแสดงข้อมูลในคำสั่ง SQL ได้ในทำนองเดียวกันกับการเชื่อมตารางสองตารางดังนี้

```
SELECT f.fid, fname, c.cid, cname
FROM (faculty f JOIN teaches t USING fid) AS ft JOIN courses c USING cid;
```

ตัวอย่าง Multiple grouping columns

```
SELECT s.branchNo, s.staffNo, COUNT(*) AS myCount
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo
ORDER BY s.branchNo, s.staffNo;
```

หลักการทำงานของ JOIN

วิธีการคำนวณบนฐานข้อมูลเชิงสัมพันธ์เพื่อให้ได้มาซึ่งผลลัพธ์ของการดำเนินการ JOIN นั้นอาศัยพื้นฐานของผลลัพธ์จากการดำเนินการหาผลคูณคาร์ทีเซียน จากนั้นอาศัยการดำเนินการอื่นๆ อีก กล่าวได้ว่าผลลัพธ์ที่เกิดจากการดำเนินการ JOIN เป็นซับเซตของผลลัพธ์ที่ได้จากการดำเนินการหาผลคูณคาร์ทีเซียน เราใช้ **CROSS JOIN** สำหรับดำเนินการหาผลคูณคาร์ทีเซียนของตารางด้วยคำสั่ง SQL ซึ่งมีรูปแบบคำสั่งดังนี้

```
SELECT [DISTINCT | ALL] {* | column_list}
FROM table_name1 CROSS JOIN table_name2
```

การใช้ **CROSS JOIN** ในรูปแบบที่ได้แสดงคือการ **SELECT** ข้อมูลจากตารางข้อมูลหลายๆ ตารางโดยไม่มีการกรองด้วย **WHERE** จากรูปแบบการใช้ที่ได้แสดง เราสามารถใช้ร่วมกับ WHERE และ ORDER BY ได้เช่นเดียวกับการ JOIN โดยทั่วไป

ขั้นตอนการคำนวณของการดำเนินการ JOIN สามารถอธิบายได้ดังนี้

1) คำนวนหาผลคูณคาร์ทีเซียนจากตารางที่ระบุใน **FROM**
2) ในกรณีที่มีการใช้ **WHERE** ให้กรองเฉพาะทูเพิลที่ตรงตามเงื่อนไข ซึ่งนั่นคือการดำเนินการ **restriction** ในพีชคณิตเชิงสัมพันธ์
3) สำหรับแถวที่เหลืออยู่จากการดำเนินการ ทำการสร้างข้อมูลตามรายการใน **SELECT** และสร้างแถวทีละแถว
4) ถ้ามีการใช้ **SELECT DISTINCT** ให้กำจัด แถวที่ซ้ำกันออก การดำเนินการในข้อ 3 และ 4 คือ การดำเนินการ **projection** บน **selection**
5) เรียงลำดับตามคอลัมน์ที่กำหนดในกรณีมีการใช้ **ORDER BY**

OUTER JOIN

โดยปกติแล้วผลลัพธ์จากการ JOIN ระหว่างตารางจะแสดงเฉพาะข้อมูลที่สามารถเชื่อมกันได้ ในกรณีที่ทูเพิลใดๆ ในตารางทางซ้ายไม่สามารถเชื่อมกับทูเพิลใดๆ ในตารางทางขวาได้เลย ทูเพิลนั้นจะไม่ปรากฏในผลลัพธ์ ในทำนองเดียวกันกับทูเพิลทางขวาที่ไม่สามารถเชื่อมกับทูเพิลใดๆ ของตารางทางซ้าย ทูเพิลนั้นจะไม่ปรากฏในผลลัพธ์ เราเรียกการ JOIN ในลักษณะนี้ว่า INNER JOIN

พิจารณาตัวอย่างตารางข้อมูลนักศึกษาและตารางสาขาวิชา โดย did ในตารางนักศึกษาซึ่งได้แก่รหัสสาขาวิชาที่นักศึกษาสังกัด อ้างถึง did ซึ่งได้แก่รหัสประจำสาขาวิชาในตารางสาขาวิชา ซึ่งข้อมูลที่ยกตัวอย่างนี้สามารถเชื่อมกันได้ด้วย did โครงสร้างตารางนักศึกษาที่เพิ่มคอลัมน์ did นี้เป็นตัวอย่างประกอบสำหรับหัวข้อนี้เท่านั้น สำหรับคอลัมน์ institute เป็นคอลัมน์แสดงสำนักวิชาที่สาขาวิชานั้นๆ สังกัดอยู่ ในกรณีนี้ตาราง Department ยังไม่ได้อยู่ในรูปแบบบรรทัดฐานเนื่องจากสำนักวิชานั้นซ้ำๆ กัน หรือ Department อาจเป็จ View ก็ได้

Students

cid	sid	name	login	age	gpa	*did*
3100904022132	B5075666	สมชาย	somchai@it	18	3.44	204
3100904032132	B5075688	สมศรี	somsri@com	18	3.21	408
3100905622132	B5075650	สมศรี	somsri@it	19	3.82	204
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80	103
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00	NULL

Department

did	dname	dphone	institute
D102	สาขาวิชาฟิสิกส์	4187	สำนักวิชาวิทยาศาสตร์
D204	สาขาวิชาเทคโนโลยีสารสนเทศ	4273	สำนักวิชาเทคโนโลยีสังคม
D205	สาขาวิชาเทคโนโลยีการจัดการ	4267	สำนักวิชาเทคโนโลยีสังคม
D423	สาขาวิชาวิศวกรรมคอมพิวเตอร์	4422	สำนักวิชาวิศวกรรมคอมพิวเตอร์
D609	สาขาวิชาแพทย์ศาสตร์	3917	สำนักวิชาแพทย์ศาสตร์

จากข้อมูลตัวอย่าง เราสามารถแสดงข้อมูลนักศึกษาพร้อมกับชื่อสาขาวิชาที่นักศึกษาสังกัดได้ด้วย SQL และผลลัพธ์ดังต่อไปนี้

```
SELECT s.*, d.dname
FROM student s JOIN department d ON s.did = d.did
```

cid	sid	name	login	age	gpa	did	dname
3100904022132	B5075666	สมชาย	somchai@it	18	3.44	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904032132	B5075688	สมศรี	somsri@com	18	3.21	D423	สาขาวิชาวิศวกรรมคอมพิวเตอร์
3100905622132	B5075650	สมศรี	somsri@it	19	3.82	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80	D609	สาขาวิชาแพทย์ศาสตร์

จากตัวอย่างดังกล่าว ผลลัพธ์ที่ได้เป็นไปในลักษณะการดำเนินการ JOIN เช่นเดียวกับตัวอย่างอื่นๆ โดยผลลัพธ์ไม่ปรากฏระเบียนของนักศึกษา "สมน้ำหน้า" เนื่องจากนักศึกษาคนดังกล่าวยังไม่มีสาขาวิชาสังกัด นอกจากนี้สาขาวิชาบางสาขาวิชาที่ยังไม่มีนักศึกษาสังกัดก็ไม่ได้แสดงในผลลัพธ์ เช่นสาขาวิชาฟิสิกส์ เป็นต้น

ในบางกรณีผู้ใช้ข้อมูลมีความประสงค์ที่จะแสดงข้อมูลทั้งหมดในตารางทางซ้ายของการ JOIN แม้ว่าจะไม่มีข้อมูลทูเพิลใดๆ จากตารางทางขวาที่สามารถเชื่อมข้อมูลกันได้ เช่นต้องการแสดงข้อมูลของนักศึกษาทั้งหมด พร้อมทั้งชื่อสาขาวิชาที่สังกัด รวมทั้งนักศึกษาที่ยังไม่ได้สังกัดสาขาวิชาด้วย หรือมีความประสงค์จะแสดงข้อมูลจากตารางทางขวาทั้งหมดแม้จะไม่มีข้อมูลจากตารางทางซ้ายที่เชื่อมกันได้ เช่นต้องการแสดงข้อมูลของสาขาวิชาทั้งหมดพร้อมทั้งรายชื่อนักศึกษากรณีที่มีนักศึกษาสังกัด หรือแม้แต่ผู้ใช้มีความประสงค์แสดงข้อมูลจากทั้งสองตาราง โดยแสดงทั้งข้อมูลที่ทูเพิลสามารถเชื่อมกันได้ และไม่สามารถเชื่อมกันได้ทั้งจากตารางซ้ายและขวา ซึ่งได้แก่การดำเนินการ OUTER JOIN ในพีชคณิตเชิงสัมพันธ์ ซึ่งสามารถเขียนให้อยู่ในรูปของคำสั่ง SQL ได้ดังนี้

ตัวอย่าง Left OUTER JOIN

แสดงข้อมูลนักศึกษาทั้งหมดพร้อมทั้งชื่อสาขาวิชาที่นักศึกษาคนนั้นๆ สังกัด รวมถึงแสดงข้อมูลนักศึกษาที่ไม่มีสังกัดสาขาวิชาด้วย

```
SELECT s.*, d.dname
FROM student s LEFT JOIN department d ON s.did = d.did
```

cid	sid	name	login	age	gpa	did	dname
3100904022132	B5075666	สมชาย	somchai@it	18	3.44	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904032132	B5075688	สมศรี	somsri@com	18	3.21	D423	สำนักวิชาวิศวกรรมคอมพิวเตอร์
3100905622132	B5075650	สมศรี	somsri@it	19	3.82	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80	D609	สาขาวิชาแพทย์ศาสตร์
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00	NULL	NULL

ผลลัพธ์ที่ได้จากการใช้ **LEFT JOIN** เป็นการแสดงข้อมูลของตารางทางซ้ายทั้งหมด จึงได้ชื่อว่า LEFT JOIN สำหรับนักศึกษา "สมน้ำหน้า" ที่ยังไม่มีสาขาวิชาสังกัดนั้น ผลลัพธ์ที่เกิดจากตารางทางขวา ซึ่งได้แก่ตาราง Department จะมีค่าเป็น NULL

ตัวอย่าง Right OUTER JOIN

แสดงข้อมูลนักศึกษาทั้งหมดพร้อมทั้งชื่อสาขาวิชาที่นักศึกษาคนนั้นๆ สังกัด รวมถึงแสดงรายชื่อสาขาวิชาแม้สาขาวิชานั้นๆ จะไม่มีนักศึกษาในสังกัด

```
SELECT s.*, d.dname
FROM student s RIGHT JOIN department d ON s.did = d.did
```

cid	sid	name	login	age	gpa	did	dname
3100904022132	B5075666	สมชาย	somchai@it	18	3.44	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904032132	B5075688	สมศรี	somsri@com	18	3.21	D423	สำนักวิชาวิศวกรรมคอมพิวเตอร์
3100905622132	B5075650	สมศรี	somsri@it	19	3.82	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80	D609	สาขาวิชาแพทย์ศาสตร์
NULL	NULL	NULL	NULL	NULL	NULL	NULL	สาขาวิชาฟิสิกส์
NULL	NULL	NULL	NULL	NULL	NULL	NULL	สาขาวิชาเทคโนโลยีการจัดการ

ตัวอย่าง Full OUTER JOIN

แสดงข้อมูลนักศึกษาทั้งหมดพร้อมทั้งชื่อสาขาวิชาที่นักศึกษาคนนั้นๆ สังกัด ทั้งนี้รวมถึงแสดงข้อมูลนักศึกษาที่ไม่มีสังกัดสาขาวิชาด้วย และรายชื่อสาขาวิชาแม้สาขาวิชานั้นๆ จะไม่มีนักศึกษาในสังกัด

```
SELECT s.*, d.dname
FROM student s FULL JOIN department d ON s.did = d.did
```

cid	sid	name	login	age	gpa	did	dname
3100904022132	B5075666	สมชาย	somchai@it	18	3.44	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904032132	B5075688	สมศรี	somsri@com	18	3.21	D423	สาขาวิชาวิศวกรรมคอมพิวเตอร์
3100905622132	B5075650	สมศรี	somsri@it	19	3.82	D204	สาขาวิชาเทคโนโลยีสารสนเทศ
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80	D609	สาขาวิชาแพทย์ศาสตร์
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	สาขาวิชาฟิสิกส์
NULL	NULL	NULL	NULL	NULL	NULL	NULL	สาขาวิชาเทคโนโลยีการจัดการ

8.2.8 มีอยู่ และ ไม่มีอยู่ (EXISTS and NOT EXISTS)

EXISTS ใช้กับข้อคำถามย่อยสำหรับการตรวจสอบเงื่อนไขการมีอยู่อย่างน้อย 1 ระเบียนข้อมูล โดยข้อคำถามย่อย EXISTS จะมีค่าเป็นจริง ก็ต่อเมื่อข้อคำถามย่อย SELECT สามารถแสดงผลลัพธ์ได้อย่างน้อย 1 ระเบียน และในทางตรงกันข้าม ข้อคำถามย่อย **NOT EXISTS** จะมีค่าเป็นจริง ก็ต่อเมื่อข้อคำถามย่อย SELECT ไม่สามารถแสดงผลลัพธ์ได้เลยแม้แต่ระเบียนเดียว

การตรวจสอบการมีอยู่อย่างน้อย 1 ระเบียนหรือไม่มีเลยนั้นตรวจสอบเฉพาะจำนวนระเบียน เราจึงใช้ SELECT * FROM ในข้อคำถามย่อย เนื่องจากไม่สนใจในจำนวนหรือค่าของคอลัมน์ใดๆ

ตัวอย่าง

แสดงข้อมูลนักศึกษาที่ลงทะเบียนเรียนรายวิชา 204204

```
SELECT sid, name
FROM students s
WHERE EXISTS (SELECT *
        FROM enrolled e
        WHERE s.sid = e.sid AND cid = '204204');
```

คำสั่ง SQL ข้างต้นจะตรวจสอบข้อมูลของนักศึกษาที่ปรากฏในตาราง Enrolled และมีการลงทะเบียนเรียนวิชา 204204 ซึ่งหากพบ ข้อคำถามย่อยก็จะมีค่าเป็นจริง หรือ **true** สำหรับระเบียนของนักศึกษาแต่ละระเบียนในข้อคำถามหลักที่ทำการตรวจสอบ ในลักษณะของ SQL ดังนี้

```
SELECT sid, name
FROM students s
WHERE true;
```

ระเบียนของนักศึกษาในข้อคำถามหลักคนใด ที่ข้อคำถามย่อย EXISTS มีค่าเป็น true ข้อมูลจะถูกแสดง ซึ่งการใช้คำสั่ง SQL ในลักษณะดังกล่าวให้ผลลัพธ์เช่นเดียวกันกับการเชื่อมตารางในรูปแบบของการ JOIN ดังนี้

```
SELECT sid, name
FROM students s, enrolled e
WHERE s.sid = e.sid AND cid = '204204';
```

8.2.9 Combining Result Tables (UNION, INTERSCET, EXCEPT)

การใช้ UNION, INTERSECT และ EXCEPT เป็นการดำเนินการแบบเซ็ต อาจเลือกใช้ในกรณีที่คำสั่ง SQL มีความซับซ้อน สามารถแยกคำสั่ง SQL ออกจากกันเป็น 2 ชุดและนำผลลัพธ์มารวมหรือมากระทำต่อกันต่อไป ตัวอย่างต่อไปนี้แสดงวิธีการใช้คำสั่ง UNION, INTERSECT และ EXCEPT อย่างง่าย

ตัวอย่าง UNION

แสดงข้อมูลนักศึกษาที่ลงทะเบียนเรียนรายวิชา 204204 และ 204203

```
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204203')
UNION
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204204')
```

จากตัวอย่าง คำสั่ง SELECT ในส่วนแรกแสดงข้อมูลจากตารางนักศึกษา โดยที่ WHERE ทำการเปรียบเทียบ sid กับ sid ที่ได้จากข้อคำถามย่อยในการแสดงรหัสนักศึกษา sid จากตาราง enrolled ตามรหัสรายวิชาที่ระบุไว้ ซึ่งตัวอย่างเป็นการแสดงข้อมูลนักศึกษาที่ลงทะเบียนรายวิชา 2040204 จากนั้น นำมารวมกับผลลัพธ์ที่ได้ในการแสดงผลคำสั่ง SELECT อีกชุดหนึ่งหลัง UNION ผลลัพธ์ที่ได้คือข้อมูลนักศึกษาที่ลงทะเบียนในทั้ง 2 รายวิชา ให้สังเกตการใช้เครื่องหมายวงเล็บ "( )" ครอบชุดคำสั่ง SELECT ทั้ง 2 โดยให้ระลึกว่าผลลัพธ์ที่สามารถนำมา UNION กันได้ต้องเข้ากันได้ กล่าวคือมีจำนวนและลำดับของคอลัมน์เท่ากัน

ตัวอย่าง INTERSECT

แสดงข้อมูลนักศึกษาที่ลงทะเบียนเรียนทั้งรายวิชา 204204 และ 204203

```
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204203')
INTERSECT
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204204')
```

จากตัวอย่างสามารถอธิบายได้ในทำนองเดียวกันกับตัวอย่างการใช้ UNION ต่างกันตรงที่ผลลัพธ์ที่ได้คือข้อมูลของนักศึกษาที่ลงทะเบียนทั้ง 2 รายวิชา นักศึกษาที่ลงทะเบียนรายวิชาใดวิชาหนึ่งเท่านั้นจะไม่ปรากฏในผลลัพธ์

ตัวอย่าง EXCEPT

แสดงข้อมูลนักศึกษาที่ลงทะเบียนเรียนรายวิชา 204203 แต่ต้องไม่ลงทะเบียนเรียนรายวิชา 204204 ด้วย

```
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204203')
EXCEPT
(SELECT sid, name, gpa
FROM students
WHERE sid IN (SELECT sid
        FROM enrolled
        WHERE cid = '204204')
```

จากตัวอย่างสามารถอธิบายได้ในทำนองเดียวกันกับตัวอย่างการใช้ UNION และ INTERSECTION แต่การใช้ EXCEPT นั้นตรงตามความหมายของคำสั่งคือการยกเว้น โดยการแสดงผลลัพธ์จากคำสั่ง SELECT ในชุดแรก ยกเว้นระเบียนที่ปรากฏในผลลัพธ์ของ SELECT ในชุดที่สองซึ่งอยู่หลังจากคำสั่ง EXCEPT การใช้งาน EXCEPT นี้ตรงกันกับการดำเนินการ DIFFERENCE ในพีชคณิตเชิงสัมพันธ์

8.3 การเพิ่ม การแก้ไข และการลบข้อมูล (Insertion, Update and Deletion)

นอกจากคำสั่ง SELECT ที่ใช้ในการแสดงข้อมูลแล้ว คำสั่ง SQL ในกลุ่มของ DML ยังได้แก่คำสั่งที่ใช้ในการเพิ่ม การแก้ไข และการลบข้อมูล ซึ่งได้แก่คำสั่ง INSERT, UPDATE และ DELETE ตามลำดับ ทั้งนี้คำสั่ง SELECT ใช้สำหรับแสดงข้อมูลที่บรรจุอยู่ในฐานข้อมูลอยู่แล้ว และเป็นเพียงการแสดงข้อมูลเท่านั้น ข้อมูลที่บรรจุอยู่ในฐานข้อมูลจะไม่มีการเปลี่ยนแปลงใดๆ คำสั่ง SQL ในหัวข้อนี้เป็นคำสั่งที่ใช้ในการเปลี่ยนแปลงข้อมูลดังต่อไปนี้

8.3.1 การเพิ่มข้อมูล (Data Insertions)

การเพิ่มข้อมูลเข้าไปในตาราง ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

```
INSERT INTO <table_name> [(column_list)]
VALUES (data_value_list)
```

โดย

INSERT INTO	คือคำสั่งที่ใช้ในการเพิ่มข้อมูล
table_name	ชื่อของตารางที่ต้องการแทรกข้อมูลเข้าไป
column_list	คือรายการของคอลัมน์ข้อมูลที่ต้องการเพิ่มข้อมูล ในกรณีที่เพิ่มข้อมูลหลายๆ คอลัมน์ ให้ใช้เครื่องหมายจุลภาค "," คั่นระหว่างคอลัมน์
VALUES	ประกอบคำสั่ง INSERT INTO เพื่อระบุว่าแทรกข้อมูลด้วย "ค่า" ที่กำหนด
data_value_list	รายการของค่าข้อมูลที่ต้องการเพิ่ม ในกรณีที่เพิ่มข้อมูลหลายๆ คอลัมน์ ให้ใช้เครื่องหมายจุลภาค "," ขั้นระหว่างค่า โดยค่าจะต้องสัมพันธ์กับ column_list หรือโครงสร้างของตารางข้อมูลที่จะทำการเพิ่มข้อมูล

สามารถเขียนรูปแบบคำสั่งเพิ่มข้อมูล ที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

เพิ่ม เข้าไปยัง <ชื่อตาราง> [(รายการของคอลัมน์)]

ด้วยค่า (รายการของค่า);

นอกจากนี้การแทรกค่าเข้าไปในตารางข้อมูลยังสามารถแทรกค่าได้โดยการนำค่าที่ได้จากคำสั่ง SELECT เพิ่มลงในฐานข้อมูลด้วยรูปแบบดังนี้

INSERT INTO <table_name> [(column_list)]

<select_statement>

โดย

select_statement	คือคำสั่ง SQL ที่ใช้ในการสร้างข้อมูลจากตารางข้อมูลต่างๆ เพื่อนำมาแทรกในตารางที่กำหนดในคำสั่ง INSERT INTO

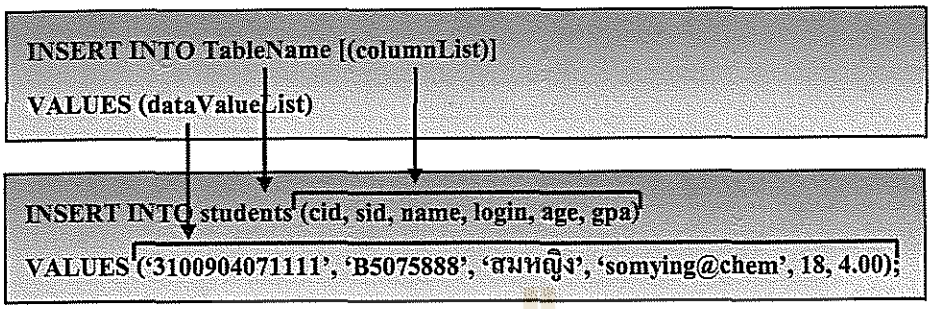
ตัวอย่าง

เพิ่มข้อมูลเข้าไปในตารางนักศึกษา 1 ระเบียน โดยกำหนดค่าให้ทุกคอลัมน์

INSERT INTO students (cid, sid, name, login, age, gpa)

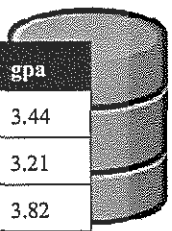
VALUES ('3100904071111', 'B5075888', 'สมหญิง', 'somying@chem', 18, 4.00);

จากตัวอย่าง ส่วนของคำสั่ง **INSERT INTO students** เป็นการกำหนดให้เพิ่มข้อมูลที่ระบุไว้ลงในตาราง students สำหรับส่วน **(cid, sid, name, login, age, gpa)** นั้นเป็นการระบุรายการคอลัมน์ที่ต้องการเพิ่มข้อมูล หมายความว่าในบางครั้งตารางข้อมูลอาจมีจำนวนคอลัมน์มากกว่าที่ระบุก็ได้ แต่เราต้องการเพิ่มข้อมูลไม่ครบทุกคอลัมน์ ในตัวอย่างนี้เป็นการเพิ่มข้อมูลจำนวน 6 คอลัมน์ซึ่งเท่ากับจำนวนคอลัมน์ที่มีอยู่ในตาราง students ในส่วนของ **VALUES ('3100904071111', 'B5075888', 'สมหญิง', 'somying@chem', 18, 4.00)** เป็นการระบุค่าที่ต้องการเพิ่ม แสดงได้ตามภาพต่อไปนี้



ผลลัพธ์ที่ได้จากการประมวลผลคำสั่งการเพิ่มข้อมูลดังตัวอย่างคือการเพิ่มข้อมูลลงในตาราง students ด้วยระเบียนใหม่ 1 ระเบียน ดังนี้

Students



cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
⇨ 3100904071111	B5075888	สมหญิง	somying@chem	18	4.00

ตัวอย่าง

การเพิ่มข้อมูลโดยไม่ระบุรายการคอลัมน์ที่ต้องการเพิ่มข้อมูล

INSERT INTO students

VALUES ('3100904071111', 'B5 075888', 'สมหญิง', 'somying@chem', 18, 4.00);

จากตัวอย่าง คำสั่ง INSERT INTO นี้ไม่มีรายการกำกับคอลัมน์ที่ต้องการเพิ่มข้อมูลในตารางหลังชื่อตารางซึ่งสามารถกระทำได้ในกรณีที่รายการของค่าที่จะแทรกมีจำนวนและลำดับตรงกันกับโครงสร้างของตารางที่เพิ่มข้อมูล แสดงได้ดังนี้

```
INSERT INTO students
VALUES ('3100904071111', 'B5075888', 'สมหญิง', 'somying@chem', 18,  4.00);
```

Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
⇨ 3100904071111	B5075888	สมหญิง	somying@chem	18	4.00

การแทรกข้อมูลดังกล่าวอาจจะสะดวกแต่อาจก่อให้เกิดปัญหาขึ้นได้ภายหลัง เช่น ในกรณีที่มีการแก้ไขสคีมาของตารางข้อมูลดังภาพต่อไปนี้

```
INSERT INTO students
VALUES ('3100904071111', 'B5075888', 'สมหญิง', 'somying@chem', 18, 4.00);
```

Students

cid	Sid	name	login	did	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	D204	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	D423	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	D204	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	D609	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	NULL	22	2.00
⇨ 3100904071111	B5075888	สมหญิง	somying@chem	18	4.00	?

จากรูป ตารางข้อมูล students อาจมีการเพิ่มคอลัมน์ did เพื่อระบุรหัสสาขาที่นักศึกษาแต่ละคนสังกัด คำสั่ง SQL ที่แสดงจึงไม่สามารถใช้ได้กับตารางดังกล่าว พิจารณาค่าในลำดับที่ 5 และ 6 ซึ่งได้แก่ค่า age และ gpa ตามสคีมาเดิมของตาราง ซึ่งคอลัมน์ทั้ง 2 ถูกเลื่อนลำดับไปด้วยคอลัมน์ did การเพิ่มข้อมูล "18" ลงในคอลัมน์ did และ "4.00" ลง

ในคอลัมน์ age จึงไม่ถูกต้อง ดังนั้นเราจึงควรระบุรายการของคอลัมน์ ไว้ในคำสั่ง INSERT ด้วย ซึ่งเป็นแนวปฏิบัติที่พึงกระทำ

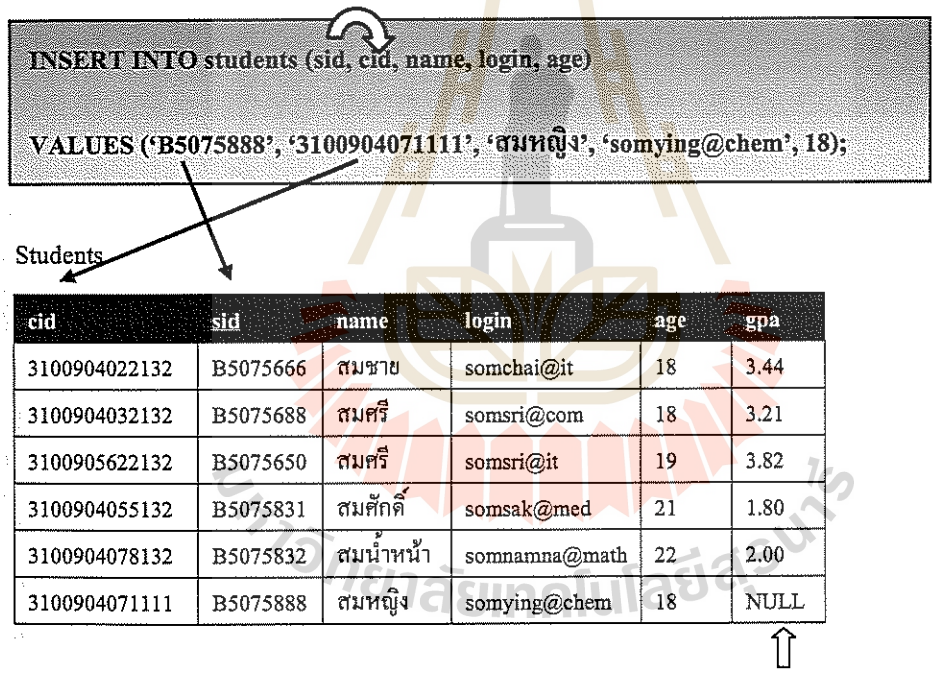
ตัวอย่าง

การเพิ่มข้อมูลโดยระบุรายการของคอลัมน์ไม่เรียงตามที่ปรากฏในสคีมาของตาราง

INSERT INTO students (sid, cid, name, login, age, gpa)

VALUES ('B5075888', '3100904071111', 'สมหญิง', 'somying@chem', 18, gpa);

จากตัวอย่าง ให้สังเกตรายการของคอลัมน์ ในลำดับที่ 1 และ 2 ของรายการในคำสั่ง INSERT INTO นั้น สลับกันกับลำดับที่ของคอลัมน์ในตาราง การกระทำเช่นนี้สามารถกระทำได้ แต่ค่าที่ระบุในรายการของค่าหลัง VALUES นั้นจะต้องสอดคล้องกันกับคอลัมน์ในรายการคอลัมน์ ในตัวอย่างค่า 'B5075888' ซึ่งเป็นค่าแรกในรายการเพิ่มข้อมูลสัมพันธ์กันกับคอลัมน์ sid และค่า '3100904071111' ในลำดับถัดมาก็สัมพันธ์กันกับคอลัมน์ cid ทำให้เราสามารถใช้คำสั่งดังกล่าวในการเพิ่มข้อมูลได้อย่างถูกต้อง ซึ่งสามารถแสดงด้วยภาพได้ดังนี้



cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
3100904071111	B5075888	สมหญิง	somying@chem	18	NULL

นอกจากนั้น ในตัวอย่างนี้ รายการของคอลัมน์มีเพียง 5 รายการเท่านั้น โดยคำสั่งไม่ได้ระบุค่าของ gpa ซึ่งสามารถกระทำได้ในกรณีที่เราอนุญาตให้ค่าในคอลัมน์ gpa สามารถเป็นค่าว่างได้ สำหรับในกรณีนี้หากไม่ได้กำหนดค่าโดยปริยายไว้ ค่า gpa จะมีค่าเป็นค่าว่างหรือ NULL

อย่างไรก็ตาม เราไม่สามารถละเว้นการกำหนดค่าในการเพิ่มข้อมูลสำหรับคอลัมน์ที่ไม่อนุญาตให้เป็นค่าว่างหรือได้กำหนดค่าโดยปริยายไว้ ดังนี้

INSERT INTO students (name, login, age, gpa)

VALUES ('สมหญิง', 'somying@chem', 18, 4.00);

Students

cid	sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00
NULL	NULL	สมหญิง	somying@chem	18	4.00

เนื่องจาก sid เป็น PRIMARY KEY ของตาราง เราจึงต้องระบุค่า sid ทุกครั้งในการเพิ่มข้อมูลในตาราง students คำสั่ง INSERT INTO ที่ไม่ได้ระบุค่าของ sid นี้จึงไม่สามารถใช้งานได้

ตัวอย่าง

การเพิ่มข้อมูลโดยใช้ผลลัพธ์ที่ได้จากคำสั่ง SELECT

แทรกข้อมูลระเบียนใหม่ลงในตาราง student_hall_of_fame ด้วยข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยสูงกว่า 3.25 และเป็นนักศึกษาที่รหัสนักศึกษาขึ้นต้นด้วย B50

INSERT INTO student_hall_of_fame (cid, sid, name, age, gpa)

SELECT cid, sid, name, age, gpa

FROM students

WHERE gpa >= 3.25 AND sid LIKE 'B50*'

จากตัวอย่าง ส่วนของคำสั่ง **INSERT INTO student_hall_of_fame (cid, sid, name, age, gpa)** เป็นการระบุตารางและคอลัมน์ข้อมูลที่จะทำการเพิ่มข้อมูล ซึ่งมีรูปแบบเหมือนคำสั่ง INSERT ทั่วๆ ไป สำหรับในส่วนที่ 2 นั้นแทนด้วยข้อคำถาม SELECT ซึ่งสามารถใช้คำสั่ง SELECT ได้ทุกรูปแบบ ผลลัพธ์ที่ได้ต้องสัมพันธ์กันกับรายการคอลัมน์ในการเพิ่มข้อมูล ในกรณีนี้ข้อคำถามเป็นการเลือกข้อมูลจากตาราง students สำหรับนักศึกษาที่รหัสนักศึกาขึ้นต้นด้วย B50 และมีเกรดเฉลี่ยมากกว่าหรือเท่ากับ 3.25 เพื่อคัดลอกข้อมูลของนักศึกาเหล่านี้เข้าไปอยู่ในตาราง student_hall_of_fame ซึ่งตารางดังกล่าวเป็นตารางที่ใช้เก็บนักศึกษาที่จะได้รับการเชิดชูเกียรต์นั่นเอง หากตาราง student_hall_of_fame นี้ยังไม่มีข้อมูลบรรจุอย่าง จะได้ผลลัพธ์ของตัวอย่างนี้ดังนี้

Student_Hall_Of_Fame

cid	sid	name	age	gpa
3100904022132	B5075666	สมชาย	18	3.44
3100905622132	B5075650	สมศรี	19	3.82

8.3.2 การแก้ไขข้อมูล (Data Updates)

การแก้ไขข้อมูลในตาราง ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

UPDATE <table_name>

SET column_name1 = data_value1 [, column_name2 = data_value2...]

[WHERE search_condition]

โดย

UPDATE	คือคำสั่งที่ใช้ในการแก้ไขข้อมูล
table_name	ชื่อของตารางที่ต้องการแก้ข้อมูล
SET	ประกอบคำสั่ง UPDATE เพื่อระบุการ "กำหนด" ค่าตามที่ระบุ
column_name	คือชื่อของคอลัมน์ข้อมูลที่ต้องการแก้ไข
=	เครื่องหมายประกอบคำสั่งแก้ไขข้อมูล หมายถึงการกำหนดค่า
data_value	คือค่าใหม่ของข้อมูลที่ต้องการแก้ไข

คำสั่ง UPDATE หนึ่งคำสั่งสามารถแก้ไขข้อมูลพร้อมๆ กันได้ทีละหลายๆ คอลัมน์ โดยการใช้เครื่องหมายจุลภาค "," ขั้นชุดของการกำหนดค่า โดยคำสั่ง UPDATE จะแก้ไขค่าคอลัมน์ที่กำหนดของทุกระเบียน แต่ถ้ามีการกำหนด search_condition ด้วย WHERE นั้น ระบบจัดการฐานข้อมูลจะแก้ไขเฉพาะระเบียนที่เป็นไปตามเงื่อนไขใน WHERE เท่านั้น

สามารถเขียนรูปแบบคำสั่งแก้ไขข้อมูล ที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

แก้ไข <ชื่อตาราง>

กำหนดให้ ชื่อคอลัมน์1 = ค่าใหม่1 [, ชื่อคอลัมน์2 = ค่าใหม่2]

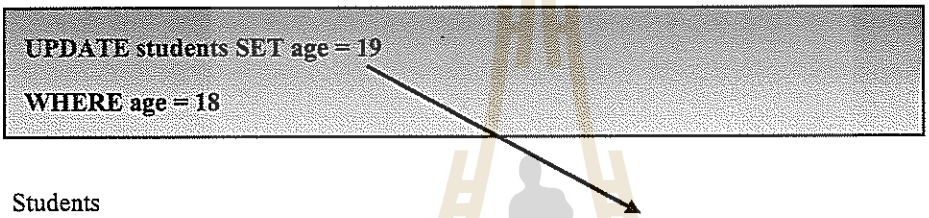
[โดยที่ เงื่อนไข];

ตัวอย่าง

แก้ไขอายุนักศึกษาในตาราง students โดยเปลี่ยนข้อมูลอายุของนักศึกษาที่มีอายุ 18 ปี ให้เป็นอายุ 19 ปี

```
UPDATE students
SET age = 19
WHERE age = 18
```

จากตัวอย่าง ระบบจัดการฐานข้อมูลจะประมวลผลคำสั่งโดยการค้นหานักศึกษาที่มีอายุ 18 ปีตามเงื่อนไข **WHERE age = 18** ในตาราง students จากนั้นจะทำการแก้ไขค่าในคอลัมน์ age ให้เป็น 19 ปี ตามส่วนของคำสั่ง **SET age = 19** แสดงได้ด้วยภาพต่อไปนี้



Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ได้ผลลัพธ์ดังนี้

Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	19	3.44
3100904032132	B5075688	สมศรี	somsri@com	19	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	21	2.00

พึงระลึกว่าโดยตารางข้อมูลไม่ควรเก็บค่าในลักษณะอายุ แต่ควรจัดเก็บเป็นวันเดือนปีเกิด

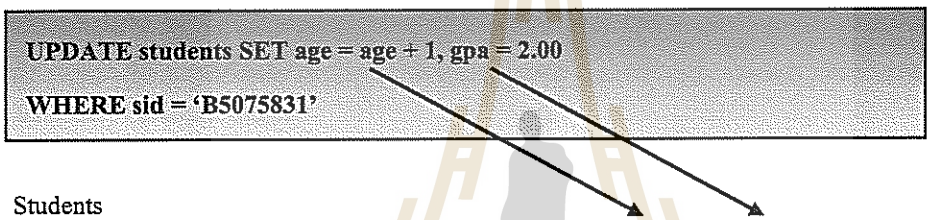
ตัวอย่าง

แก้ไขอายุนักศึกษาในตาราง students โดยเปลี่ยนข้อมูลของนักศึกษาที่มีรหัสนักศึกษาเท่ากับ B5075831 โดยแก้ไขอายุให้เพิ่มขึ้น 1 ปี และกำหนดให้ gpa มีค่าเท่ากับ 2.00

UPDATE students SET age = age + 1, gpa = 2.00

WHERE sid = 'B5075831'

จากตัวอย่างเป็นการแก้ไขค่าของนักศึกษารหัส B5075831 จากเงื่อนไข **WHERE sid = 'B5075831'** โดยเป็นการแก้ไขค่าพร้อมๆ กัน 2 คอลัมน์ ได้แก่คอลัมน์ age และ gpa ซึ่งการแก้ไขค่านั้นเราสามารถกำหนดค่าใหม่ให้เป็นค่าที่ได้จากการคำนวณได้ เช่น **age = age + 1** ระบบจัดการฐานข้อมูลจะทำการค้นหานักศึกษารหัส B5075831 ก่อน จากนั้นจึงทำการแก้ไขข้อมูลคอลัมน์ age และ gpa ตามค่าที่กำหนดหรือค่าที่ได้จากการคำนวณ ซึ่งแสดงได้ด้วยภาพต่อไปนี้



Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	**B5075831**	สมศักดิ์	somsak@med	**21**	**1.80**
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ได้ผลลัพธ์ดังนี้

Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	19	3.44
3100904032132	B5075688	สมศรี	somsri@com	19	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	22	2.00
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	21	2.00

8.3.3 การลบข้อมูล (Data Deletions)

การลบข้อมูลออกจากตาราง ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

DELETE FROM <table_name>

[WHERE search_condition]

โดย

DELETE FROM คือคำสั่งที่ใช้ในการลบข้อมูลออกจากตาราง

table_name ชื่อของตารางที่ต้องการลบระเบียนใดๆ

คำสั่ง DELETE จะทำการลบข้อมูลออกเป็นระเบียนๆ ซึ่งระเบียนใดที่ตรงกับเงื่อนไขที่กำหนดใน search_condition ด้วย WHERE นั้น จะถูกลบทิ้ง พึงสังเกตว่าคำสั่ง DELETE FROM นั้นไม่ได้ลบข้อมูลออกเป็นคอลัมน์ๆ

สามารถเขียนรูปแบบคำสั่งลบข้อมูล ที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

ลบ ออกจาก <ชื่อตาราง>

[โดยที่ เงื่อนไข];

ตัวอย่าง

ลบข้อมูลของนักศึกษารหัส B5075831

DELETE FROM students

WHERE sid = 'B5075831'

ระบบจัดการฐานข้อมูลจะทำการค้นหานักศึกษารหัส B5075831 จากนั้นจะทำการลบข้อมูลนักศึกษาคนดังกล่าวออกทั้งระเบียน

Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	18	3.44
3100904032132	B5075688	สมศรี	somsri@com	18	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904055132	B5075831	สมศักดิ์	somsak@med	21	1.80
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	22	2.00

ได้ผลลัพธ์ดังนี้

Students

cid	Sid	name	login	age	gpa
3100904022132	B5075666	สมชาย	somchai@it	19	3.44
3100904032132	B5075688	สมศรี	somsri@com	19	3.21
3100905622132	B5075650	สมศรี	somsri@it	19	3.82
3100904078132	B5075832	สมน้ำหน้า	somnamna@math	21	2.00

การเพิ่ม แก้ไข และลบข้อมูล นั้นต้องเป็นไปตามเงื่อนไขบังคับบูรณภาพของข้อมูลด้วย มิเช่นนั้นจะไม่สามารถดำเนินการตามคำสั่งได้

8.4 Stored Procedures and Triggers

Stored procedures และ triggers เป็นความสามารถของ SQL ที่ทำให้การใช้งานระบบจัดการฐานข้อมูลมีประสิทธิภาพมากขึ้น โดย stored procedures เป็นรูปแบบของคำสั่ง SQL ที่มีการทำงานเป็นขั้นตอนต่อเนื่อง ทำซ้ำ หรือเลือกทำได้เช่นเดียวกันกับภาษาคอมพิวเตอร์แบบ procedural language หรือภาษาเชิงกระบวนการ เช่นภาษาซี เป็นต้น แต่อยู่ในรูปของ SQL สำหรับ triggers นั้นเป็นความสามารถในการกำหนดให้ SQL ทำงานอย่างหนึ่งอย่างใดเมื่อมีเหตุการณ์บางอย่างเกิดขึ้น ทั้งในเอกสารเล่มนี้จะแนะนำ stored procedures และ triggers ในเบื้องต้นเพื่อให้เข้าใจแนวคิดและการใช้งานอย่างง่ายเท่านั้น การใช้งาน stored procedures และ triggers สามารถศึกษาได้ในหัวข้อชั้นสูงของระบบจัดการฐานข้อมูลต่อไป

8.4.1 Stored Procedures

คำสั่ง SQL จะมีประโยชน์เพิ่มขึ้นอีกมากหากเราสามารถนำคำสั่งทั้งกลุ่มของ DDL และ DML มาพัฒนาระบบโดยเขียนคำสั่งเป็นขั้นตอน มีกระบวนการ และโครงสร้าง เช่นเดียวกับคำสั่งเชิงกระบวนการ และโครงสร้างอื่นๆ แนวคิดของ stored procedures นี้ยังเกี่ยวข้องกับการจัดการธุรกรรมซึ่งจะกล่าวถึงในบทถัดไปและ triggers ในหัวข้อถัดไป

ในภาษาเชิงโครงสร้าง (structural language) เช่นภาษาซี เราสามารถรับค่าจากผู้ใช้ เปรียบเทียบค่าเพื่อเลือกทำหรือทำซ้ำ เพื่อแสดงผลลัพธ์ (output) ที่ถูกต้องตามข้อมูลนำเข้า (input) ได้ ความสามารถในการรองรับการเขียนโปรแกรมเชิงกระบวนการและโครงสร้างใน SQL นี้ไม่ได้มีมาพร้อมกับ SQL ในรุ่นแรกๆ จะเป็นความสามารถพิเศษที่ผู้ผลิตระบบจัดการฐานข้อมูลแต่ละรายเพิ่มเข้ามาให้ได้เปรียบคู่แข่งและเพิ่มประสิทธิภาพการใช้งานฐานข้อมูล อีกทั้งการพัฒนาระบบฐานข้อมูลไม่ได้จำเป็นที่จะต้องเขียนคำสั่ง SQL เชิงกระบวนการ ทำให้มาตรฐานของคำสั่ง SQL เชิงกระบวนการนั้นถูกกำหนดได้ช้า syntax หรือไวยกรณ์ของ SQL เชิงกระบวนการของผู้ผลิตแต่ละรายในปัจจุบันจึงค่อนข้างหลากหลาย

ใน Oracle เราเรียกความสามารถเชิงกระบวนการของ SQL ว่า PL/SQL (Procedural Language/SQL) ใน Microsoft SQL Server เรียกว่า Transac-SQL (T-SQL) สำหรับใน MySQL นั้นเรียกว่า SQL/PSM (SQL/Procedural Stored Modules) ซึ่งตรงกับมาตรฐานที่กำหนดโดย ISO ในปี 2003—SQL:2003

เราสามารถใช้ SQL เชิงกระบวนการในการสร้างกระบวนการเพื่อรอบรับความต้องการของผู้ใช้ได้ตั้งแต่การดำเนินการกับข้อมูลง่ายๆ จนถึงระดับที่มีความซับซ้อนมากๆ ได้ ในบทที่ 10 ฐานข้อมูลบนเว็บได้มีการอธิบายถึงสถาปัตยกรรม 3-Tier ของซอฟต์แวร์ ซึ่งโดยปกติ ส่วนของโปรแกรมที่รองรับความต้องการของผู้ใช้จะแยกออกจากระดับฐานข้อมูล การใช้ SQL เชิงกระบวนการนั้นเป็นการฝังส่วนของโปรแกรมที่รองรับความต้องการของผู้ใช้ลงไปใน

ฐานข้อมูลซึ่งอาจเพิ่มความเร็วในการทำงานของระบบแต่จะลดความเป็นอิสระระหว่างส่วนของแอพลิเคชันกับฐานข้อมูลลง ซึ่งการพิจารณาใช้ SQL เชิงกระบวนการนั้นขึ้นอยู่กับความเหมาะสมในการใช้งาน

Stored procedures เป็นการเขียน SQL เชิงกระบวนการและจัดเก็บไว้เป็นโปรแกรมที่ทำงานบนระบบจัดการฐานข้อมูล stored procedures ที่สร้างขึ้นสามารถถูกเรียกใช้งานได้เรื่อยๆ นอกจาก stored procedures จะประมวลผลตามสถานะของข้อมูลในฐานข้อมูลขณะนั้นๆ แล้ว stored procedures ยังสามารถรับค่าที่ผู้ใช้กำหนด รวมถึงสร้างผลลัพธ์หลากหลายรูปแบบได้ ในที่นี้จะแนะนำ stored procedure ในเบื้องต้นเท่านั้น โดยจะกล่าวถึงส่วนของคำสั่งในการสร้าง stored procedure เฉพาะเบื้องตนและตัวอย่างพื้นฐาน ซึ่งสามารถนำรูปแบบไปประยุกต์ใช้ได้ในโอกาสต่อไป การสร้าง stored procedures มีรูปแบบอย่างง่ายดังนี้

```
CREATE      [OR REPLACE] PROC [owner,] <procedure_name> [;number]
            [<@parameter_name> <datatype> [, ...]]
[WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}]
[FOR REPLICATION]
AS sql_statements;
```

โดย

CREATE OR REPLACE PROC	เป็นคำสั่งที่ใช้ในการสร้าง stored procedure ในกรณีที่ระบุ OR REPLACE ด้วย ระบบจัดการฐานข้อมูลจะแทนที่ stored procedure ที่มีอยู่เดิมในกรณีที่ชื่อซ้ำกัน
<procedure_name>	คือชื่อของ stored procedure
<@parameter_name> <data_type>	คือชื่อและชนิดของของพารามิเตอร์ ทั้งนี้ stored procedures สามารถรับค่าพารามิเตอร์จากผู้ใช้ได้เพื่อประกอบการประมวลผล ดังแสดงในตัวอย่างต่อไป
AS	ประกอบคำสั่ง CREATE PROC เพื่อระบุว่าสร้าง PROC "ด้วย" SQL เชิงกระบวนการที่กำหนด
sql_statements	คำสั่ง SQL ซึ่งอาจจะเป็น SQL ง่ายๆ หรือ SQL เชิงกระบวนการเต็มรูปแบบ

ตัวอย่าง

สร้าง stored procedures สำหรับแสดงข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับ 3.25

```
CREATE PROC sp_show_gpa1
AS
SELECT sid, name, gpa
FROM students
```

WHERE gpa >= 3.25;

จากตัวอย่าง เป็นการสร้าง stored procedure โดยใช้ชื่อว่า sp_show_gpa1 ด้วยคำสั่ง SQL ที่แสดงข้อมูล sid, name และ gpa ของนักศึกษาที่มีเกรดเฉลี่ยมากกว่า 3.25 การสร้าง stored procedure นั้นเป็นการสร้างวัตถุสำหรับประมวลผลเก็บไว้ในฐานข้อมูลเท่านั้นแต่ยังไม่ได้มีประมวลผลตามคำสั่งใน stored procedure จนกว่าจะมีการเรียกใช้งานจริง ซึ่งการเรียกใช้งาน stored procedure นั้นต้องใช้คำสั่ง EXEC เพื่อเรียกใช้งาน ตามด้วยชื่อของ stored procedure ดังนี้

EXEC sp_show_gpa1;

ได้ผลลัพธ์ดังนี้

Students

sid	name	gpa
B5075666	สมชาย	3.44
B5075650	สมศรี	3.82

ตัวอย่าง ** ใส่ IF..ELSE

สร้าง stored procedure ที่สามารถรับพารามิเตอร์ได้ เป็น stored procedure สำหรับแสดงข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับค่าพารามิเตอร์ที่ระบุ

CREATE PROC sp_show_gpa2

@gpa FLOAT

AS

SELECT sid, name, gpa

FROM students

WHERE gpa >= @gpa;

จากตัวอย่างเป็นการสร้าง stored procedures โดยใช้ชื่อว่า sp_show_gpa2 ในบรรทัดถัดมาหลังจากชื่อของ stored procedure เป็นการประกาศรายการของพารามิเตอร์ที่สามารถนำไปใช้ในการประมวลผลซึ่งสามารถมีได้หลายพารามิเตอร์ ในตัวอย่างนี้ stored procedures รับพารามิเตอร์เพียงพารามิเตอร์เดียว ชื่อว่า @gpa มีชนิดข้อมูลเป็น FLOAT ชื่อของพารามิเตอร์จะต้องใช้เครื่องหมาย "@" นำหน้าเสมอ ทั้งในขณะประกาศพารามิเตอร์ และการอ้างถึงในขณะใช้ค่าพารามิเตอร์ใน stored procedures

Stored procedures ดังกล่าวใช้ในการแสดงข้อมูลของนักศึกษา ประกอบด้วย sid, name, gpa ดังเช่นตัวอย่างที่แล้ว แต่ในตัวอย่างนี้ จะแสดงข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับค่าที่กำหนดเป็นพารามิเตอร์ขณะเรียกใช้ stored procedures นี้ จากส่วนของคำสั่ง **WHERE gpa >= @gpa** เห็นได้ว่า เป็นการเปรียบเทียบค่า gpa กับค่าพารามิเตอร์ @gpa การเรียกใช้ stored procedures ให้ใช้คำสั่ง EXEC ตามด้วยชื่อ stored procedures เช่นเดียวกับการเรียกใช้งาน stored procedures โดยทั่วไป พร้อมทั้งระบุค่าพารามิเตอร์หลังจากชื่อของ stored procedures ดังนี้

EXEC sp_show_gpa2 3.25;

เป็นการแสดงข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับ 3.25 ซึ่งผลลัพธ์ที่ได้เป็นตารางข้อมูลเหมือนกันกับตัวอย่างที่แล้ว

ตัวอย่างอื่นๆ

EXEC sp_show_gpa2 3.50;

EXEC sp_show_gpa2 2.00;

EXEC sp_show_gpa2 1.80;

แสดงตัวอย่างการเรียกใช้ stored procedures sp_show_gap2 ในการแสดงข้อมูลนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับ 3.50, 2.00 และ 1.80 ตามลำดับ

เราสามารถดัดแปลงการสร้าง stored procedures ให้สามารถรับค่าพารามิเตอร์มากกว่า 1 ค่า พร้อมทั้งใช้งานได้อย่างถูกต้อง เพียงประกาศตัวแปรเพิ่มในรายการพารามิเตอร์ และกำหนดค่าพารามิเตอร์เพิ่มเมื่อเรียกใช้งาน stored procedures โดยใช้เครื่องหมายจุลภาค "," ขั้นระหว่างชุดพารามิเตอร์

8.4.2 Triggers

Trigger เป็นกลไกของระบบจัดการฐานข้อมูลที่เพิ่มเข้ามาเพื่อช่วยให้ระบบจัดการฐานข้อมูลทำงานบางอย่างโดยอัตโนมัติเมื่อเกิดเหตุการณ์ที่เรากำหนดไว้ได้ เช่น ให้นำข้อมูลเก่าการการเปลี่ยนแปลงไปจัดเก็บไว้ในตารางสำรองข้อมูล หรือให้ตรวจสอบค่าต่างๆ เป็นต้น ความสามารถที่ trigger ทำได้อาจเป็นในระดับง่ายๆ ไม่ซับซ้อน โดยอาศัย SQL ที่ไม่ซับซ้อน จนถึงการทำงานที่ซับซ้อนโดยการใช้ SQL เชิงกระบวนการและการใช้ stored procedures

เราอาจใช้ trigger ในการกำหนดเงื่อนไขบังคับบูรณภาพที่ซับซ้อนก็ได้ แต่เราจะหลีกเลี่ยงการใช้ trigger สำหรับการดำเนินการดังกล่าวถ้า CHECK CONSTRAINTS สามารถตรวจสอบได้เนื่องจากการจัดการ CONSTRAINTS ที่มากับระบบจัดการฐานข้อมูลมีประสิทธิภาพมากกว่า

รูปแบบของคำสั่ง trigger และการใช้งานรวมกับ SQL เชิงกระบวนการรวมถึง stored procedures นั้นแตกต่างกันไปพอสมควรตามระบบจัดการฐานข้อมูลของผู้ผลิตแต่ละราย ในที่นี้อธิบายการใช้งาน trigger ที่อิงมาตรฐาน SQL:2003 เป็นส่วนใหญ่ โดยจะได้ยกตัวอย่างการใช้งาน trigger สำหรับสร้างตารางตรวจสอบเส้นทาง (audit trail) ที่เป็นการดำเนินการอันหนึ่งที่เรามักนิยมใช้ประโยชน์จาก trigger

การสร้าง trigger ใช้คำสั่งที่มีรูปแบบอย่างง่ายดังนี้

CREATE [OR REPLACE] TRIGGER <trigger_name>
{BEFORE|AFTER} {INSERT|DELETE|UPDATE} ON <table_name>
FOR EACH ROW
<trigger_statement>

โดย

CREATE OR REPLACE TRIGGER	เป็นคำสั่งที่ใช้ในการสร้าง trigger ในกรณีที่ระบุ OR REPLACE ด้วย ระบบจัดการฐานข้อมูลจะแทนที่ trigger ที่มีอยู่เดิมในกรณีที่ชื่อซ้ำกัน
<procedure_name>	คือชื่อของ trigger
BEFORE|AFTER	ผู้สร้าง trigger สามารถระบุได้ว่าจะให้ trigger ทำงานก่อนและ/หรือหลังจากการเกิดเหตุการณ์ใดๆ
INSERT|DELETE|UPDATE	ระบุเหตุการณ์หรือการดำเนินการใดๆ ที่กำหนดให้ trigger ทำงาน
ON	ประกอบคำสั่ง CREATE TRIGGER เพื่อระบุว่าสร้าง TRIGGER "บน" ตารางใด
FOR EACH ROW	ประกอบคำสั่ง CREAT TIRGGER เพื่อระบว่าสร้าง TRIGGER "สำหรับแต่ละแถว"
sql_statements	คำสั่ง SQL ซึ่งอาจจะเป็น SQL ง่ายๆ หรือ SQL เชิงกระบวนการเต็มรูปแบบ

ตัวอย่าง

สร้าง stored procedures สำหรับแสดงข้อมูลของนักศึกษาที่มีเกรดเฉลี่ยมากกว่าหรือเท่ากับ 3.25

```
CREATE TRIGGER tr_student_audit_insert
ON students
BEFORE INSERT
FOR EACH ROW
BEGIN
        INSERT INTO student_audit (sid, action, user, timestamp, gpa)
        VALUES (NEW.sid, 'I', CURRENT_USER(), NOW(), NEW.gpa)
END;
```

จากตัวอย่าง เป็นการสร้าง trigger โดยใช้ชื่อว่า tr_student_audit_insert ด้วยคำสั่ง SQL ที่ทำงานบนตาราง students โดยจะทำงานทุกครั้งก่อนการ INSERT ข้อมูลลงในตาราง การทำงานของ trigger เรียกว่าการ fire trigger ซึ่งสิ่งที่ trigger ที่ได้ยกตัวอย่างดำเนินการได้แก่การเพิ่มระเบียนลงในตาราง student_audit ซึ่งเป็นตารางตรวจสอบการเปลี่ยนแปลงข้อมูลของตาราง students ด้วยข้อมูล ดังนี้

sid	คือรหัสนักศึกษา ในที่นี้กำหนดค่าเท่ากับ NEW.sid ซึ่ง NEW ในที่นี้คือการระบุถึงค่าใหม่พร้อมกำกับด้วยชื่อคอลัมน์ โดยที่ trigger สำหรับการดำเนินการ UPDATE นั้นจะมีค่าทั้ง

OLD และ NEW ที่สามารถอ้างอิงได้ ในขณะที่ trigger สำหรับการดำเนินการ DELETE นั้นมีเพียงค่า OLD ที่สามารถอ้างอิงได้

action คือกิจกรรมหรือการดำเนินการที่ทำให้ trigger ทำงาน ในที่นี้ระบุเป็นค่าคงที่ 'I' หมายถึง INSERT

user ชื่อของผู้ใช้ฐานข้อมูลขณะนั้นๆ โดยปกติระบบสารสนเทศใดๆ จะกำหนดชื่อผู้ใช้ฐานข้อมูลโดยเฉพาะ ในที่นี้ใช้ค่าที่เรียกจากฟังก์ชัน CURRENT_USER() ซึ่งจะแต่งต่างกันออกไปสำหรับแต่ละระบบจัดการฐานข้อมูล การจัดการผู้ใช้กล่าวถึงเพิ่งเติมในบทที่ 9

timestamp เวลาที่ trigger นั้นๆ ทำงาน เพื่อใช้ในการตรวจสอบลำดับเหตุการณ์ที่เกิดขึ้น

gpa ค่าเกรดเฉลี่ยที่เพิ่มเข้ามาในฐานข้อมูล โดยใช้ค่า NEW.gpa หมายถึงค่า gpa ที่เพิ่งเพิ่มเข้ามาในฐานข้อมูล เราเก็บค่า gpa ไว้ตรวจสอบเนื่องจาก gpa เป็นข้อมูลที่มีความสำคัญ และเพื่อตรวจสอบเจตนาของการแก้ไขข้อมูลที่อาจเกิดจากการบุกรุกระบบได้

สำหรับ **BEGIN** และ **END** นั้นยกตัวอย่างเพื่อแนะนำการเขียน SQL ในรูปแบบของธุรกรรม กรณีที่มีชุดของ SQL หลายคำสั่ง ซึ่งเป็น syntax ของ SQL เชิงกระบวนการ หมายถึงเริ่มและสิ้นสุดชุดกระบวนการ

หลังจากการสร้าง trigger แล้วเสร็จ ระบบจะไม่ได้ดำเนินการอื่นใดนอกจากสร้างวัตถุ trigger ไว้ในฐานข้อมูล ซึ่งวัตถุ trigger นี้จะทำงานก็ต่อเมื่อเกิดเหตุการณ์หรือกิจกรรมตามที่กำหนดไว้ ในกรณีนี้ trigger tr_student_audit_insert จะทำงานเมื่อมีการเรียกใช้คำสั่ง INSERT กับตาราง students ดังตัวอย่างต่อไปนี้จึงจะทำให้ trigger student_audit_insert ทำงาน

```
INSERT INTO students (cid, sid, name, login, age, gpa)
VALUES ('3100904078132, 'B5075832, 'สมน้ำหน้า', 'somnamna@math', 21, 2.00)
```

เราสามารถสร้าง trigger สำหรับกิจกรรม UPDATE และ DELTE บนตาราง students ได้อีก อาจสร้างแยกเป็น trigger ต่างหาก หรือสร้างรวมกันแล้วตรวจสอบว่า trigger นั้นเกิดจากกิจกรรมใด ตารางต่อไปนี้เป็นตัวอย่างของข้อมูลตารางตรวจสอบ student_audit หลังจากการดำเนินการกับตาราง students ด้วยการดำเนินการมาแล้วระยะหนึ่ง

Students_Audit

audit_id	sid	action	user	timestamp	gpa
234	B5075688	U	enrollment	11/11/08 2:22:22 AM	3.21
235	B5075650	D	enrollment	11/11/08 3:33:33 PM	3.82
236	B5075832	U	enrollment	11/11/08 4:44:44 PM	2.00
237	B5075832	I	enrollment	11/11/08 1:11:11 AM	2.00

คำสั่ง INSERT INTO students ที่ได้ยกตัวอย่างสำหรับการทำให้ trigger tr_student_audit_insert ดำเนินการนั้น นอกจากจะทำให้ข้อมูลใหม่เพิ่มลงในตาราง students แล้ว ยังมีข้อมูลสำหรับการตรวจสอบเพิ่มขึ้นในระเบียนสุดท้ายของตาราง student_audit ซึ่งมี audit_id เท่ากับ 237 ดังแสดง คอลัมน์ audit_id เป็น PRIMARY KEY ที่ค่าสามารถเพิ่มเองได้โดยอัตโนมัติ ตาราง student_audit ต้องถูกสร้างขึ้นมาก่อนที่ trigger จะทำงาน

ตาราง student_audit นี้เป็นตารางที่ใช้ในการบันทึกกิจกรรมที่เกิดขึ้นกับตาราง students เพื่อประโยชน์ในการสำรองข้อมูล การตรวจสอบความผิดพลาดของระบบแอพพลิเคชันว่ามีการคำนวณเกรดผิดพลาดหรือมีตรรกะของโปรแกรมผิดพลาดหรือไม่ และประโยชน์ประการสำคัญคือการตรวจสอบการแก้ไขที่ไม่ได้รับอนุญาต เป็นต้น เราเรียกตารางการตรวจสอบลักษณะนี้ว่าตารางตรวจสอบ (audit table) สำหรับการตรวจสอบเส้นทาง (audit trail) หรือตรวจสอบกิจกรรมใดๆ มันนิยมสร้างขึ้นเพื่อรองรับตารางข้อมูลที่มีความสำคัญ

8.5 แบบฝึกหัดท้ายบท

ให้นักศึกษาเขียนคำสั่ง SQL เพื่อรองรับ Transaction Requirements แต่ละข้อของโครงงาน เช่า-ฉัน-สิ

บทที่ 9 การจัดการธุรกรรมและความมั่นคงปลอดภัยของข้อมูล

วัตถุประสงค์

- สามารถอธิบายความหมายของธุรกรรมได้
- สามารถอธิบายความหมายของภาวะการเกิดพร้อมกัน
- สามารถอธิบายปัญหาที่เกิดจากภาวะการเกิดพร้อมกันของธุรกรรม
- สามารถอธิบายวิธีแก้ปัญหาที่เกิดจากภาวะการเกิดพร้อมกันของธุรกรรมได้
- สามารถอธิบายเกี่ยวกับการคืนสภาพข้อมูล
- สามารถอธิบายเกี่ยวกับความมั่งคงปลอดภัยของข้อมูลได้

คำสำคัญ: ธุรกรรม (transaction); ภาวะการเกิดพร้อมกัน (concurrency); serializbility; เทคนิคการปิดกั้น (lock technique); การปิดตาย (dead lock); การคือสภาพข้อมูล (data recovery); การลงบันทึกระบบ (system log); ความมั่งคงปลอดภัยของฐานข้อมูล (database security)

บทที่ 9 การจัดการธุรกรรมและความมั่นคงปลอดภัยของข้อมูล

9.1 บทนำ

หลังจากที่ได้ทำการออกแบบ และพัฒนาระบบฐานข้อมูล ตลอดจนนำระบบฐานข้อมูลมาใช้แล้ว การดูแลและจัดการฐานข้อมูลเป็นสิ่งที่จะต้องปฏิบัติต่อไป ซึ่งสิ่งที่ควรคำนึงถึงในการจัดการระบบฐานข้อมูลได้แก่ความน่าเชื่อถือของข้อมูล และความมั่นคงปลอดภัยของข้อมูล โดยเฉพาะในกรณีที่มีใช้งานฐานข้อมูลพร้อมกันหลายๆ คนหรือในกรณีที่ระบบทำงานไม่ปกติหรือเกิดความล้มเหลวของระบบ ซึ่งในบทนี้จะกล่าวถึงการจัดการภาวะการเกิดพร้อมกัน (concurrency control) การกู้ข้อมูล (data recovery) และความมั่นคงปลอดภัยของข้อมูล (data security)

การจะจัดการภาวะการเกิดพร้อมกันจะต้องเข้าใจถึงลักษณะการใช้งานที่ก่อให้เกิดปัญหาขึ้น โดยจะต้องเข้าใจถึงธุรกรรม (transaction) และ serializability จากนั้นจะสามารถเข้าใจถึงการใช้เทคนิคต่างๆ ในการแก้ปัญหาภาวะการเกิดพร้อมกันได้ โดยเฉพาะการใช้เทคนิคการปิดกั้น (lock technique) จะการจัดการกับปัญหาที่อาจจะตามมาอีกในการใช้เทคนิคต่างๆ ซึ่งได้อธิบายในหัวข้อ 9.2 สำหรับหัวข้อ 9.3 เป็นการอธิบายกลไกของการกู้ข้อมูลของระบบจัดการฐานข้อมูลในกรณีที่ระบบทำงานไม่ปกติ และหัวข้อ 9.4 อธิบายถึงการรักษาความมั่นคงปลอดภัยของข้อมูลด้วยวิธีต่างๆ เพื่อให้ข้อมูลปลอดภัยจากภัยคุกคามต่างๆ

9.2 การจัดการภาวะการเกิดพร้อมกัน

การรองรับการทำงานของผู้ใช้งานพร้อมๆ กันหลายคนอย่างมีประสิทธิภาพเป็นข้อดีที่สำคัญข้อหนึ่งของระบบจัดการฐานข้อมูล อย่างไรก็ตามการใช้งานฐานข้อมูลพร้อมๆ กันโดยผู้ใช้งานหลายคนก่อให้เกิดปัญหาที่จะต้องจัดการ โดยเฉพาะการใช้งานข้อมูลบางข้อมูร่วมกันและทำให้ข้อมูลที่ผู้ใช้อีกคนหนึ่งใช้อยู่นั้นเป็นข้อมูลที่เป็นปัจจุบัน ในที่นี้จะเป็นการอธิบายถึงกลไกอันเป็นพื้นฐานของการจัดการภาวะการเกิดพร้อมกัน ซึ่งระบบจัดการฐานข้อมูลอาศัยกลไกนี้ในการจัดการกับภาวะการเกิดพร้อมกันแล้วระดับหนึ่ง ซึ่งผู้พัฒนาระบบสารสนเทศที่ใช้งานฐานข้อมูลต้องพัฒนาระบบให้สอดรับกับกลไกในการจัดการภาวะการเกิดพร้อมกันนี้ ในหัวข้อนี้จะกล่าวถึงแนวคิดที่เกี่ยวข้องกับการจัดการภาวะการเกิดพร้อมกันได้แก่ ธุรกรรม (transaction) ความจำเป็นในกาแก้ปัญหาภาวะการเกิดพร้อมกัน serializability และเทคนิคทั่วไปที่ใช้ในการจัดการกับภาวะการเกิดพร้อมกัน

9.2.1 ธุรกรรม (Transaction)

ธุรกรรม หรือ รายการ(เปลี่ยนแปลง) ในที่นี้คือธุรกรรมฐานข้อมูล (database transaction) หมายถึงคำสั่ง หรือชุดของคำสั่งที่กระทำกับฐานข้อมูลเพื่อให้บรรลุกิจกรรมใดๆ ตามวัตถุประสงค์ โดยกำหนดว่าคำสั่งย่อยๆ ต้องดำเนินการให้เสร็จสิ้นเพื่อธุรกรรม 1 ธุรกรรม และการดำเนินธุรกรรมของผู้ใช้แต่ละคนพร้อมกันนั้นเป็นอิสระต่อกันตลอดจนเสมือนกับผู้ใช้แต่ละคนใช้งานฐานข้อมูลอยู่เพียงคนเดียว การจัดการธุรกรรมในฐานข้อมูลนั้นมีวัตถุประสงค์ 2 ข้อใหญ่ๆ ได้แก่

1) เพื่อสร้างกลไกในการกู้คืนข้อมูล ในกรณีที่ธุรกรรมกำลังดำเนินการยังไม่เสร็จสิ้นไม่ว่าจะด้วยเหตุกาณ์ใดๆ ข้อมูลบางส่วนได้ถูกเปลี่ยนแปลงแต่ยังไม่ครบถ้วน จะต้องจัดการปรับสภาพข้อมูลในฐานข้อมูลให้ถูกต้อง ไม่ว่าจะเป็นการยกเลิกการแก้ไขที่ได้กระทำมา หรือดำเนินต่อให้เสร็จสิ้น ฯลฯ
2) เพื่อสร้างความเป็นอิสระแยกจากกันของผู้ใช้งานฐานข้อมูลหลายๆ คนพร้อมกัน ไม่มีการแก้ไขข้อมูลซ้ำกันจนเป็นเหตุให้เกิดความผิดพลาด

ซึ่งวัตถุประสงค์ทั้ง 2 ข้อนี้ของการจัดการธุรกรรมคือใจความสำคัญของบทนี้ ธุรกรรมของฐานข้อมูลจะต้องมีคุณสมบัติที่เรียกสั้นๆ ว่า ACID กล่าวคือการไม่แบ่งแยก (atomicity) ความสอดคล้องต้องกันของข้อมูล/ความต้องกัน (consistency) ความเป็นอิสระแยกจากกัน (isolation) และการคงอยู่อย่างถาวร/ความคงทน (durability)

ธุรกรรมมีลักษณะ "ทั้งหมดหรือไม่ทำเลย (all-or-nothing)" หมายความว่าธุรกรรมต้องทำสำเร็จทั้งหมด ไม่เช่นนั้นก็เสมือนกับว่าไม่มีการดำเนินคำสั่งย่อยใดๆ เลย ข้อมูลที่กระทำโดยคำสั่งย่อยๆ ทั้งหมดต้องเป็นไปตามข้อกำหนดบูรณภาพเพื่อความถูกต้องของฐานข้อมูลและข้อมูลต้องสอดคล้องต้องกันในสภาพที่ควรจะเป็นในบริบทการใช้งาน ธุรกรรมแต่ละธุรกรรมต้องเป็นอิสระแยกจากกันและข้อมูลที่เปลี่ยนแปลงจากการดำเนินการที่เสร็จสิ้นของธุรกรรมต้องคงอยู่ในอุปกรณ์จัดเก็บต่อไป

คำสั่งย่อยๆ ในธุรกรรมได้แก่คำสั่งที่ดำเนินการกับฐานข้อมูล เช่น คำสั่งในการแสดงข้อมูล และคำสั่งในการแก้ไขข้อมูล สำหรับคำสั่งในการแสดงข้อมูลจากฐานข้อมูลนั้นได้แก่คำสั่ง SELECT ส่วนคำสั่งที่ใช้ในการแก้ไข้ข้อมูลนั้นได้แก่คำสั่ง INSERT ที่ใช้ในการเพิ่มข้อมูล คำสั่ง UPDATE ที่ใช้ในการปรับปรุงแก้ไขข้อมูล และคำสั่ง DELETE ในการลบระเบียนข้อมูล

พิจารณาตัวอย่างธุรกรรมการโอนเงิน นายสมชายจากบัญชีหมายเลข 0352000001 โอนเงินจำนวน 1,000 บาท ไปให้กับนางสมศรี บัญชีหมายเลข 0352000002 ในที่นี้ balance คือชื่อฟิลด์ของข้อมูลยอดเงินในบัญชี และ bank_account คือตารางข้อมูลบัญชี ดังนี้

```
BEGIN TRANSACTION
EXEC SQL UPDATE bank account
        SET balance = balance - 1000
        WHERE account_no = '0352000001';
EXEC SQL UPDATE bank account
        SET balance = balance + 1000
        WHERE account_no = '0352000002';
COMMIT;
```

จากตัวอย่าง จะเห็นได้ว่าธุรกรรมการโอนเงินนี้ประกอบด้วยคำสั่ง SQL ที่ใช้ในการแก้ไขข้อมูลหลักๆ 2 คำสั่ง ได้แก่การ UPDATE ข้อมูลของบัญชี 2 บัญชี ซึ่งธุรกรรมนี้จะสำเร็จลงได้ยอดเงินในบัญชีแรกจะต้องถูกหักออก และยอดเงินในบัญชีที่ 2 จะต้องเพิ่มขึ้นให้สำเร็จเรียบร้อยจึงจะถูว่าธุรกรรมสมบูรณ์ ในกรณีที่คำสั่งใดๆ ไม่สำเร็จ ข้อมูลทั้งหมดจะต้องอยู่ในสภาพก่อนเริ่มธุรกรรม คำสั่ง BEGIN TRANSACTION เป็นคำสั่งกำหนดจุดเริ่มต้นของธุรกรรม และคำสั่ง COMMIT เป็นคำสั่งในการยอมรับธุรกรรมและทำให้ผลของการเปลี่ยนแปลงข้อมูลคงอยู่

นอกจากนี้แล้วชุดคำสั่ง SQL แต่ละคำสั่งยังประกอบไปด้วยกระบวนการย่อยๆ ที่เกี่ยวข้องกับอุปกรณ์ฮาร์ดแวร์และกลไกของระบบจัดการฐานข้อมูล ก่อนที่การดำเนินการตามคำสั่งนั้นๆ จะสำเร็จลุล่วงซึ่งสามารถอธิบายได้ดังนี้

คำสั่งที่เกี่ยวข้องกับการอ่านข้อมูลนั้นมีขั้นตอนดังนี้

<u>การอ่านข้อมูล</u> กำหนดให้การอ่านข้อมูล X (ข้อมูลสมมติ) เขียนแทนด้วย READ(X)

1. ค้นหาตำแหน่งที่อยู่ของ block ข้อมูลที่มีข้อมูล X ซึ่งเป็นตำแหน่งบนจานบันทึกข้อมูล เพื่อหาข้อมูล X
2. คัดลอกข้อมูล block ดังกล่าวมาเก็บไว้ใน buffer ในหน่วยความจำหลักก่อนนำไปประมวลผล
3. คัดลอกข้อมูล X จาก buffer ซึ่งเป็นที่พักข้อมูลไปประมวลผลยังชุดคำสั่งที่ใช้งานข้อมูล X

<u>การเขียนข้อมูล</u> กำหนดให้การเขียนข้อมูล X (ข้อมูลสมมติ) เขียนแทนด้วย WRITE(X)

1. ค้นหาตำแหน่งที่อยู่ของ block ข้อมูลที่มีข้อมูล X ซึ่งเป็นตำแหน่งบนจานบันทึกข้อมูล เพื่อหาข้อมูล X
2. คัดลอกข้อมูล block ดังกล่าวมาเก็บไว้ใน buffer ในหน่วยความจำหลักเพื่อรอการแก้ไขข้อมูล
3. คัดลอกข้อมูล X จากผลลัพธ์ที่ได้จากการประมวลผลด้วยชุดคำสั่งไปเขียนลงยัง buffer ซึ่งเป็นที่พักข้อมูล
4. เขียนข้อมูล block ที่ได้ทำการปรับปรุงนี้จาก buffer ลงยังจานบันทึกข้อมูล

ขั้นตอนที่ 4 ซึ่งเป็นขั้นตอนสุดท้ายของการเขียนข้อมูลนั้นถือว่าเป็นการดำเนินการเสร็จสิ้นของธุรกรรม เนื่องจากข้อมูลที่แก้ไขจะคงอยู่ ดังนั้นเมื่อขั้นตอนนี้สำเร็จ และไม่มีคำสั่งอื่นๆ อีกจึงจะถึงว่าธุรกรรมนั้นๆ เสร็จสมบูรณ์ ระบบจัดการฐานข้อมูลจะดำเนินการยอมรับ (COMMIT) ว่าธุรกรรมนั้นๆ สำเร็จลงแล้ว แต่หากการดำเนินธุรกรรมไม่มีเหตุขัดข้องก่อนการ COMMIT การดำเนินการย่อยๆ ที่ดำเนินการไปแล้วทั้งหมดจะถูก ROLLBACK หรือย้อนกลับเพื่อให้ข้อมูลมีสภาพดังเช่นก่อนเริ่มธุรกรรม

การจัดการธุรกรรมมีความจำเป็นเนื่องจากข้อมูลที่ถูกใช้งานจากธุรกรรมหลายๆ ธุรกรรมพร้อมกัน ตลอดจนความขัดข้องของระบบจะทำให้ข้อมูลไม่มีความต้องกัน (inconsistency) ได้ ดังที่จะได้อธิบายต่อไปนี้

9.2.2 ความจำเป็นในการแก้ปัญหาภาวะการเกิดพร้อมกัน

การใช้งานระบบฐานข้อมูลโดยผู้ใช้หลายๆ คนพร้อมกัน อาจก่อให้เกิดปัญหาขึ้นได้ พิจารณาตัวอย่างธุรกรรมและกรณีตัวอย่างที่ก่อให้เกิดปัญหาจากภาวะการเกิดพร้อมกัน ดังนี้

นายสมชายและนางสมศรีเป็นเจ้าของบัญชีธนาคารร่วมกัน มีเงินฝากในบัญชี 10,000 บาท นายสมชายและนางสมศรีทำธุรกรรมพร้อมๆ กัน โดยนายสมชายทำการฝากเงินเข้าบัญชีเป็นจำนวน 2,000 บาท นางสมศรีถอนเงินออกจากบัญชี 1,000 บาท แสดงเป็นขั้นตอนการทำงานย่อยๆ ของธุรกรรมทั้งสองด้วยโค้ดจำลอง (pseudo code) ได้ดังนี้

ธุรกรรมที่ 1 (T1) นายสมชายฝากเงิน 2,000 บาท สมมติให้ยอดเงินในบัญชีคือข้อมูล X

```
T1:    READ(X)
       UPDATE X = X + 2000
       WRITE(X)
```

ธุรกรรมที่ 2 (T2) นางสมศรีถอนเงิน 1,000 บาท

```
T2:     READ(X)
        UPDATE X = X - 1000
        WRITE(X)
```

ระบบจัดการฐานข้อมูลรองรับการทำงานในกรณีที่ธุรกรรมทั้ง 2 ทำงานพร้อมกัน หลักการทำงานของการรองรับธุรกรรมหลายๆ ธุรกรรมพร้อมกันนั้นคล้ายคลึงกันกับหลักการทำงานหลายงาน (multi-tasking) ของระบบปฏิบัติการ เช่นระบบปฏิบัติการ Microsoft Windows ที่ผู้ใช้สามารถพิมพ์เอกสารด้วย Microsoft Word พร้อมๆ กับการฟังเพลงจากโปรแกรม Media Player และรอเว็บเพจที่กำลังโหลดอยู่ด้วยโปรแกรม Internet Explorer หลักการทำงานของระบบปฏิบัติการที่ทำงานหลายๆ อย่างบนเครื่องคอมพิวเตอร์เครื่องเดียวพร้อมๆ กันนั้นอาศัยการสลับการทำงานไปมาของหน่วยประมวลผลให้ประมวลผลโปรแกรมทีละโปรแกรมด้วยช่วงเวลาสั้นๆ ซึ่งกระบวนการสลับงานเพื่อประมวลผลโปรแกรมแต่ละโปรแกรมเกิดขึ้นอย่างรวดเร็วมากและวนไปเรื่อยๆ บนเครื่องคอมพิวเตอร์ ในขณะที่หน่วยประมวลผลทำการประมวลผลโปรแกรมๆ หนึ่งอยู่และหยุดพักการประมวลผล เพื่อไปประมวลผลอีกโปรแกรมหนึ่งเราเรียกว่าการ interleave แล้ววนกลับมาประมวลผลโปรแกรมที่ค้างไว้ หลักการนี้เองเป็นหลักการเดียวกันกับการประมวลผลธุรกรรมหลายๆ ธุรกรรมพร้อมๆ กันของระบบจัดการฐานข้อมูล โดยระบบจัดการฐานข้อมูลจะทำการ interleave เพื่อประมวลผลหลายๆ ธุรกรรมอย่างมีประสิทธิภาพ โดยเฉพาะเมื่อระบบกำลังรอการจัดการข้อมูลกับอุปกรณ์จัดเก็บที่ใช้เวลานาน ผู้ใช้แต่ละคนไม่จำเป็นต้องรอการทำงานเป็นลำดับตั้งแต่ลำดับแรกจนถึงคนสุดท้าย ซึ่งใช้เวลาเกินความจำเป็นในการรอการติดต่อกับอุปกรณ์จัดเก็บ ตัวอย่างนี้เป็นข้อดีตัวอย่างหนึ่งของการให้บริการธุรกรรมพร้อมๆ กันของฐานข้อมูล

พิจารณากรณีตัวอย่างการจัดการธุรกรรมฝากและถอนเงิน T1 และ T2 ของนายสมชายและนางสมศรีด้วยหลักการการให้บริการธุรกรรมพร้อมๆ กัน ดังนี้

สมมติให้ธุรกรรม T1 และ T2 เกิดขึ้นพร้อมๆ กันและการประมวลผลรายการทั้ง 2 เป็นแบบสลับกันและเกิดการ interleave ผลลัพธ์ที่ได้จากกรณีตัวอย่างนี้ข้อมูลยอดเงินไม่ถูกต้อง ซึ่งเกิดจากเหตุผลหลักคือการอ่านค่าของธุรกรรม T2 ก่อนที่ข้อมูลที่ได้รับการแก้ไขในธุรกรรม T1 จะบันทึกลงในฐานข้อมูล

กรณีตัวอย่างเกิดขึ้นโดยธุรกรรม T1 ของนายสมชายทำการฝากเงินเริ่มอ่านข้อมูลยอดเงิน READ (X) จากฐานข้อมูล มีค่าเท่ากับ 10,000 จากนั้นระบบทำการคำนวณเพิ่มยอดเงินในบัญชี UPDATE X = X + 2,000 ทำให้ยอดเงินในขณะนี้มีค่าเท่ากับ 12,000 แต่เป็นยอดเงินที่อยู่ในหน่วยความจำของเครื่องคอมพิวเตอร์ ยังไม่ได้ปรับปรุงลงในจานบันทึกข้อมูลและยังไม่ได้ COMMIT ธุรกรรม แต่ระหว่างที่ระบบกำลังคำนวณยอดเงินใหม่นั้น ระบบจัดการฐานข้อมูลทำการ interleave เพื่อประมวลผลธุรกรรม T2 ที่เข้ามาพร้อมๆ กัน ณ เวลา t2 เพื่ออ่านข้อมูลยอดเงิน READ (X) ได้ค่ายอดเงิน 10,000 บาทซึ่งเป็นยอดเงินจากฐานข้อมูล ถึงขณะนี้ ไม่ว่าธุรกรรมใดจะสำเร็จก่อนหรือหลังจะเกิดความผิดพลาดของข้อมูลขึ้น ในกรณีนี้ T2 ได้รับการประมวลผลเสร็จสิ้น โดยยอดเงินในบัญชีถูกหักออก 1,000 บาท UPDATE X = X – 1,000 เหลือ 9,000 และถูกเขียนลงในฐานข้อมูล WRITE (X) และ COMMIT จากนั้นที่เวลา t3 ระบบ

ทำการประมวลผล T1 ที่เหลือต่อ โดยทำการเขียนข้อมูลยอดเงิน 12,000 ที่คำนวณค้างไว้ลงในฐานข้อมูลและ COMMIT ทำให้มียอดเงินเท่ากับ 12,000

เวลา	ธุรกรรม T1	ธุรกรรม T2	ยอดเงิน (X)
t1	BEGIN TRANSACTION		10000
	READ (X) (10000)		10000
⇨	UPDATE X = X + 2000 (**12000**)		**10000**
t2		BEGIN TRANSACTION	10000
		⇨ READ (X) (10000)	10000
		UPDATE X = X – 1000 (9000)	10000
		⇨ WRITE (X) (9000)	9000
		COMMIT	**9000**
t3 ⇨	WRITE (X) (12000)		**12000**
	COMMIT		12000

รูป**

หากลองคำนวณอย่างง่ายจะพบว่าการฝากเงิน 2,000 และถอนเงิน 1,000 บาท ยอดเงินในบัญชีควรจะมีค่าเท่ากับ 11,000 บาท แต่ในกรณีนี้มีเงิน 12,000 บาทในบัญชี เกินมา 1,000 บาท ซึ่งอาจจะเป็นที่ชื่นชอบของนายสมชายและสมศรี แต่ธนาคารจะเกิดความเสียหาย ในทางกลับกันถ้าเกิดธุรกรรมไม่ได้เกิดขึ้นตามลำดับในกรณีตัวอย่างนี้แต่เป็นกรณีที่ T2 เป็นฝ่ายเขียนข้อมูลลงฐานข้อมูลทีหลัง เงินในบัญชีจะเหลือเพียง 9,000 บาทเท่านั้น

ตัวอย่างนี้เองเป็นการอธิบายถึงความสำคัญของการจัดการภาวะการเกิดพร้อมกันเพื่อไม่ให้ข้อมูลผิดพลาด ซึ่งปัญหาที่เกิดจากภาวะการเกิดพร้อมกันที่ไม่ได้จัดการอย่างเหมาะสมแบ่งเป็น 3 ลักษณะดังนี้

1) ปัญหาข้อมูลที่ถูกปรับปรุงถูกเขียนทับ

ปัญหานี้มีลักษณะดังตัวอย่าง T1 และ T2 การฝากและถอนเงินของสมชายและสมศรี

2) ปัญหาการอ้างอิงข้อมูลที่ไม่ได้ COMMIT

ยกตัวอย่าง

กรณีตัวอย่างเกิดขึ้นโดยธุรกรรม T1 ของนายสมชายทำการฝากเงินเริ่มอ่านข้อมูลยอดเงิน READ (X) จากฐานข้อมูล มีค่าเท่ากับ 10,000 จากนั้นระบบทำการคำนวณเพิ่มยอดเงินในบัญชี UPDATE X = X + 2,000 ทำให้ยอดเงินในขณะนี้มีค่าเท่ากับ 12,000 ธุรกรรม T1 ทำงานต่อเนื่องไปจนมีการปรับปรุงข้อมูลลงในจานบันทึกข้อมูลสำเร็จ อย่างไรก็ตาม ระบบจัดการฐานข้อมูลยังไม่ได้ทำการยอมรับหรือ COMMIT ธุรกรรม ซึ่งในขณะที่จะทำการ COMMIT ธุรกรรม T1 นั้น ระบบจัดการฐานข้อมูลทำการ interleave เพื่อประมวลผลธุรกรรม T2 ที่เข้ามาพร้อมๆ กัน ณ เวลา t2 เพื่ออ่านข้อมูลยอดเงิน READ (X) ได้ค่ายอดเงิน 11,000 บาทซึ่งเป็นยอดเงินจากฐานข้อมูล ถึงขณะนี้ดูเหมือนว่าธุรกรรมทั้ง 2 จะประมวลผลได้ถูกต้อง อย่างไรก็ตาม ณ เวลา t3 ระบบจัดการฐานข้อมูลเกิดการสลับการดำเนินการ

ประมวลผลธุรกรรม interleave กลับไปยังธุรกรรม T1 แต่เกิดข้อผิดพลาดที่ธุรกรรม T1 ขึ้น ถึงแม้จะเกิดการเขียนข้อมูลลงไปในจานบันทึกข้อมูลแล้วแต่ยังไม่ได้มีการ COMMIT สำหรับข้อผิดพลาดที่เกิดขึ้นนั้นเป็นอะไรไม่สำคัญ อาจจะเป็นการผิดพลาดที่ตัวโปรแกรมเอง หรือการจัดการหน่วยความจำใดๆ ทำให้ระบบจัดการฐานข้อมูล ROLL BACK คือการเลิกทำ และกลับไปคืนสภาพข้อมูลยอดบัญชีก่อนเริ่มธุรกรรม T1 คือยอดเงิน 10,000 บาท ต่อมาระบบจัดการฐานข้อมูลประมวลผลธุรกรรม T2 ที่เหลืออยู่ โดยใช้ข้อมูลที่ดำเนินการค้างไว้ได้แก่ยอดเงิน 12,000 บาท ซึ่งยอดเงินนี้เองเป็นยอดเงินที่อ้างอิงมาจากธุรกรรม T1 ก่อนหน้าการ ROLL BACK เมื่อคำนวณเสร็จสิ้นและปรับปรุงลงในฐานข้อมูลได้ยอดเงินสุดท้ายเป็น 11,000 บาท โดยแสดงดังตารางการดำเนินธุรกรรมต่อไปนี้

เวลา	ธุรกรรม T1	ธุรกรรม T2	ยอดเงิน (X)
t1	BEGIN TRANSACTION		10000
	READ (X) (10000)		10000
	UPDATE X = X + 1000 (11000)		10000
⇨	WRITE (X) (12000)		**12000**
t2		BEGIN TRANSACTION	12000
		⇨ READ (X) (12000)	12000
t3⇨	Transaction failed.		**10000**
	ROLL BACK (เกิดข้อผิดพลาดขึ้น กลับไปใช้ค่าก่อนหน้าธุรกรรมคือ 10000)		
t4		⇨ UPDATE X = X – 1000 (11000)	11000
		WRITE (X) (11000)	**11000**
		COMMIT	11000

รูป**

เนื่องจากธุรกรรมการฝากเงินของนายสมชายไม่สำเร็จ จึงเสมือนมีการดำเนินการธุรกรรมเพียงธุรกรรมเดียวคือธุรกรรมถอนเงินของสมศรี หากลองพิจารณาการถอนเงิน 1,000 บาทจากบัญชีที่มียอดเงิน 10,000 บาท ยอดเงินที่เหลือจากการถอนควรจะเป็น 9,000 บาท แต่กรณีตัวอย่างยอดเงินคงเหลือคือ 11,000 เกินมา 2,000 บาท ซึ่งเกิดจากการอ้างอิงถึงยอดเงินที่มีการฝากแต่ไม่ได้ COMMIT ด้วย ตัวอย่างนี้แสดงให้เห็นถึงความผิดพลาดของข้อมูลที่เกิดจากการอ้างอิงถึงข้อมูลที่ไม่ได้ COMMIT และมีการ ROLL BACK สำหรับกรณีที่ยกตัวอย่างนี้เกิดความผิดพลาดข้อมูลในแบบข้อที่ 1 ที่ได้กล่าวมาแล้วด้วย คือการเขียนทับข้อมูลที่มีการแก้ไข (ข้อมูล ROLL BACK) ความผิดพลาดในลักษณะนี้สามารถเกิดขึ้นได้โดยไม่เกิดร่วมกับการผิดพลาดตามลักษณะความผิดพลาดในข้อ 1 ได้ในกรณีที่ T2 มีการดำเนินการเขียนข้อมูลที่ได้จากการคำนวณค่า X ไว้ที่อื่นโดยไม่แก้ไขข้อมูล X

3) ปัญหาการวิเคราะห์ข้อมูลผิดพลาด

บัญชีเงินฝากที่ได้ใช้อธิบายมาก่อนหน้าเป็นบัญชีเงินฝากออมทรัพย์ กำหนดให้นายสมชายและนางสมศรีมีบัญชีเงินฝากประจำร่วมกันอีกบัญชีหนึ่ง สมมติว่ามีธุรกรรม T3 คือนายสมชายโอนเงินจำนวน 1,000 บาทจากบัญชีออมทรัพย์ที่มีเงินอยู่ 10,000 บาท เข้าไปยังบัญชีเงินฝากประจำที่มีเงินอยู่ 50,000 บาท ธุรกรรม T3 นี้จะดำเนินการสำเร็จก็ต่อเมื่อมีการหักบัญชีจากบัญชีออมทรัพย์ และเพิ่มยอดในบัญชีฝากประจำโดยสมบูรณ์ถือเป็น 1 ธุรกรรม โดยธุรกรรม T3 เกิดขึ้นพร้อมๆ กับธุรกรรม T4 ที่นางสมศรีขอดูยอดเงินทั้งหมดที่มีในธนาคาร ซึ่งรวมถึงในบัญชีเงินฝากออมทรัพย์และบัญชีเงินฝากประจำ กำหนดให้เกิดกรณีตัวอย่างของการดำเนินธุรกรรมดังนี้

เวลา	ธุรกรรม T3	ธุรกรรม T4	ยอดเงิน (X)	ยอดเงิน (Y)	รวม (SUM)
t1	BEGIN TRANSACTION		10000	50000	60000
	READ (X) (10000)		10000	50000	
	UPDATE X = X - 1000 (9000)		10000	50000	
⇨	WRITE (X) (9000)		**9000**	50000	
t2		BEGIN TRANSACTION	9000	50000	
		⇨ READ (X) (9000)	9000	50000	
		READ (Y) (50000)	9000	50000	
		SUM = X + Y (14000)	9000	50000	59000
		เกิดความผิดพลาดในการวิเคราะห์ข้อมูล			
t3	READ (Y) (50000)		9000	50000	
	UPDATE Y = Y + 1000 (51000)		9000	50000	
⇨	WRITE (Y) (51000)		9000	**51000**	
	COMMIT		9000	51000	60000

ข้อผิดพลาดที่เกิดขึ้นเกิดจากการที่ธุรกรรมการแสดงยอดทรัพย์สินรวมอ่านยอดคงเหลือระหว่างที่ยอดเงินในบัญชีออมทรัพย์ถูกหักออก แต่ยังไม่ได้เพิ่มในบัญชีเงินฝากประจำ

เมื่อยอดเงินในบัญชีเงินฝากออมทรัพย์แทนด้วย X และยอดเงินในบัญชีเงินฝากประจำแทนด้วย Y ธุรกรรมเริ่มจากการที่ธุรกรรม T3 อ่านยอดบัญชีเงินฝากออมทรัพย์ READ (X) ได้ค่า 10,000 จากนั้นทำการหักยอดออก 1,000 บาท UPDATE X = X – 1,000 เพื่อที่จะโอนไปยังบัญชีเงินฝากประจำ ยอดเงินฝากที่เหลือในบัญชีเงินฝากประจำได้ถูกปรับปรุงในฐานข้อมูล WRITE (X) ซึ่งเป็นจำนวนเงิน 9,000 บาทที่เหลืออยู่ ในตอนนี้เองที่ธุรกรรม T4 ที่ดำเนินการพร้อมๆ กัน โดยนางสมศรีขอดูยอดสินทรัพย์รวมจากบัญชีทั้ง 2 โดยอ่านยอดเงินคงเหลือในบัญชีเงินฝากออมทรัพย์ READ (X) ได้ 9,000 และเงินฝากประจำ READ (Y) ได้ 50,000 บาท เหตุการณ์นี้เองที่ทำให้เกิดความผิดพลาดขึ้น

เนื่องจากธุรกรรมการโอนเงินยังไม่เสร็จสิ้น เงินที่ต้องเพิ่มในยอดของบัญชีเงินฝากประจำจำนวน 1,000 บาทนั้นยังไม่ได้ดำเนินการเนื่องจากเกิด interleave ระบบคำนวณหายอดสินทรัพย์รวม SUM ได้ 59,000

ซึ่งในความเป็นจริงแล้วสินทรัพย์ในบัญชีเงินฝากออมทรัพย์ 10,000 บาท และเงินฝากประจำ 50,000 บาท ควรจะรวมกันได้ 60,000 บาทอยู่ตลอดเวลา ดังนั้นผลรวม SUM ในธุรกรรม T4 ที่ได้ 59,000 จึงเป็นการผิดพลาดที่เกิดจากภาวะการเกิดพร้อมกันในลักษณะความผิดพลาดในการวิเคราะห์ข้อมูล

การแก้ไขปัญหาที่เกิดจากภาวะการเกิดพร้อมกันให้คำนึงถึงผลลัพธ์ที่มีลักษณะ serializability ดังนี้

9.2.3 Serializability

Serializabitity คือการจัดการภาวะการเกิดพร้อมกันให้มีผลลัพธ์เหมือนกับการที่ธุรกรรมแต่ละธุรกรรมทำงานต่อกัน โดยธุรกรรมหนึ่งดำเนินการเสร็จสิ้นก่อน แล้วต่อด้วยธุรกรรมอีกธุรกรรมหนึ่ง ถ้าการจัดการธุรกรรมสามารถทำให้ผลลัพธ์ออกมาได้ดังนี้ทำให้มั่นใจได้ว่าข้อมูลจะมีความถูกต้อง เนื่องจากการที่ธุรกรรมหนึ่งดำเนินการเสร็จสิ้นแล้วต่อด้วยการดำเนินการอีกธุรกรรมหนึ่งก็จะไม่ต่างกับการที่ระบบจัดการฐานข้อมูลอนุญาตให้ผู้ใช้งานใช้งานได้ทีละคน จึงจะไม่เกิดปัญหาความผิดพลาดที่เกิดจากภาวะการเกิดขึ้นพร้อมกันอย่างแน่นอน ซึ่ง serial ก็มาจากคำว่าต่อกัน/อนุกรมนั่นเอง

การจัดการธุรกรรมที่เกิดขึ้นพร้อมกันแต่ยังคงมีลักษณะของ serializability นั้นทำให้เกิดพิธีการหรือขั้นตอนในการจัดการกับภาวะการเกิดขึ้นพร้อมกัน (concurrency control protocol) ขึ้น เพื่อจัดลำดับการทำงานของธุรกรม โดยอาศัยหลักดังนี้

1) ถ้าธุรกรรม 2 ธุรกรรมเป็นการอ่านข้อมูล ธุรกรรมทั้ง 2 จะไม่มีผลกระทบใดๆ กับฐานข้อมูล จึงไม่ต้องมีการจัดลำดับธุรกรรม แม้ว่าธุรกรรมทั้ง 2 จะใช้ข้อมูลร่วมกัน (read only)
2) ถ้าธุรกรรม 2 ธุรกรรมเป็นการอ่านหรือเขียนข้อมูล แต่ธุรกรรมทั้ง 2 ไม่ได้ใช้ข้อมูลใดๆ ร่วมกันเลย ไม่ว่าจะอ่านหรือเขียน ธุรกรรมทั้ง 2 จึงไม่มีผลกระทบต่อกัน ไม่ต้องมีการจัดลำดับธุรกรรม (read-write on different items)
3) ถ้าธุรกรรมหนึ่งมีการดำเนินการเขียนข้อมูลแล้ว ไม่ว่าอีกธุรกรรมจะทำการอ่านหรือเขียนข้อมูลบนข้อมูลเดียวกัน อาจก่อนให้เกิดปัญหาข้อมูลผิดพลาดหรือขัดแย้งกัน ภาวะการเกิดพร้อมกันของธุรกรรมในลักษณะนี้มีความจำเป็นที่เราจะต้องจัดลำดับธุรกรรมก่อน-หลัง (read-write on the same item)

การจัดการกับปัญหาภาวะการเกิดพร้อมกันสามารถอธิบายได้ตามหัวข้อต่อไป

9.2.4 การจัดการปัญหาภาวะการเกิดพร้อมกันด้วยเทคนิคการปิดกั้น

การจัดการกับปัญหาภาวะการเกิดพร้อมกันมีหลายวิธี วิธีที่นิยมใช้กันมากคือเทคนิคที่เรียกว่าเทคนิคการปิดกั้น (lock technique) หลักการทำงานของเทคนิคการปิดกั้นนี้อาศัยการป้องกันข้อมูลที่มีการใช้งานร่วมกัน โดยไม่อนุญาตให้ธุรกรรมหรือผู้ใช้คนอื่นเข้าถึงข้อมูลได้ในระหว่างที่มีการเปลี่ยนแปลงข้อมูลในธุรกรรมหนึ่ง โดยธุรกรรมอื่นจะ

เข้าถึงข้อมูลได้ก็ต่อเมื่อธุรกรรมที่ดำเนินการกับฐานข้อมูลอยู่แล้วเสร็จ โดยจะเป็นการกำหนดลำดับการทำงาน ก่อน-หลังของขั้นตอนย่อยๆ ของธุรกรรม หัวข้อต่อไปนี้อธิบายถึงประเภทของการปิดกั้น ระดับของการปิดกั้นข้อมูล แล้วอธิบายถึงวิธีการทำงานของเทคนิคการปิดกั้นนี้

9.2.4.1 ประเภทของการปิดกั้น

การปิดกั้นข้อมูลด้วยเทคนิคการปิดกั้นแบ่งออกเป็น 2 ประเภท ได้แก่

1) **การปิดกั้นแบบร่วม (Shared Lock หรือ S Lock หรือ Read Lock)**
เป็นการปิดกั้นข้อมูลที่ยินยอมให้ธุรกรรมต่างๆ สามารถเข้าถึงและใช้งานข้อมูลร่วมกันโดยพร้อมๆ กัน โดยอนุญาตให้ธุรกรรมต่างๆ อ่านข้อมูลเท่านั้น ไม่อนุญาตให้ธุรกรรมใดธุรกรรมหนึ่งเปลี่ยนแปลงข้อมูลที่ถูกปิดกั้นนั้น

2) **การปิดกั้นแบบเฉพาะ (Exclusive Lock หรือ X Lock หรือ Write Lock)**
เป็นการปิดกั้นข้อมูลที่มีเพียงธุรกรรมเดียวเท่านั้นสามารถเข้าใช้ข้อมูลได้ ธุรกรรมอื่นๆ จะไม่สามารถเข้าถึงข้อมูลได้เลย แม้แต่จะทำการอ่านข้อมูลที่ถูกปิดกั้นแบบเฉพาะนั้นๆ ก็ไม่สามารถทำได้ โดยการปิดกั้นแบบ X Lock นี้จะใช้ในธุรกรรมที่มีการแก้ไขข้อมูล

9.2.4.2 ระดับของการปิดกั้น (Levels of Locking หรือ Graularity of Data Items)

ระดับของการปิดกั้นหมายถึงขอบเขตของการปิดกั้นในมุมมองของระดับชั้นของข้อมูล ซึ่งโดยทั่วไประดับของการปิดกั้นในระบบจัดการฐานข้อมูลแบ่งเป็นระดับได้ดังนี้

1) **การปิดกั้นฐานข้อมูล (Database Lock)**
เป็นการปิดกั้นข้อมูลทั้งฐานข้อมูล ซึ่งจะไม่อนุญาตให้ธุรกรรมอื่นๆ แก้ไขข้อมูลหรืออ่านข้อมูลใดๆ ในฐานข้อมูลเลยตามแต่ประเภทของการปิดกั้น การปิดกั้นในลักษณะนี้มักไม่เกิดขึ้นบ่อย อาจจะต้องใช้ในกรณีการสำรองข้อมูลที่สถานะข้อมูลทั้งฐานข้อมูลต้องสอดคล้องต้องกัน

2) **การปิดกั้นตาราง (Table Lock)**
เป็นการปิดกั้นข้อมูลทั้งตารางข้อมูล ส่วนใหญ่แล้วจะกระทำในกรณีที่มีการแก้ไขโครงสร้างของตาราง เช่น เพิ่ม แก้ไข หรือลดคอลัมน์

3) **การปิดกั้น Page หรือ Block หรือ Database Space (Page/Block/Database Space Lock)**
ระบบจัดการฐานข้อมูลจัดการข้อมูลในลักษณะกลุ่มของข้อมูลด้วย Page หรือ Block หรือ Database Space ที่มีประสิทธิภาพมากกว่าการจัดการทีละแถวย่อยๆ ในตารางข้อมูลหนึ่งตารางอาจประกอบด้วยกลุ่มข้อมูลหลายๆ Page การปิดกั้น Page นี้ ระบบจัดการฐานข้อมูลจะปิดกั้นข้อมูลทั้ง Page ซึ่งประกอบด้วยข้อมูลหลายแถว การปิดกั้นระดับ Page นี้เป็นการดำเนินการที่กระทำบ่อยๆ ของระบบจัดการฐานข้อมูลแทนการปิดกั้นในระดับแถว โดยปิดกั้นทั้ง Page ที่มีแถวข้อมูลที่จะทำการปิดกั้น เนื่องจากจะทำได้รวดเร็วกว่า

4) **การปิดกั้นแถว (Row Lock หรือ Record Lock)**

เป็นการปิดกั้นเฉพาะแถวข้อมูลหรือระเบียนข้อมูลที่กำหนด ระเบียนอื่นๆ จะไม่ได้รับผลกระทบจากการปิดกั้น

5) **การปิดกั้นคอลัมน์ (Column Lock)**

เป็นการปิดกั้นเฉพาะคอลัมน์ คอลัมน์อื่นๆ จะไม่ถูกปิดกั้น การใช้งานการปิดกั้นในลักษณะนี้จะใช้ในการปิดกั้นคอลัมน์ที่มีการแก้ไขข้อมูลบ่อยครั้ง เช่น คอลัมน์จำนวนสินค้าคงคลังในตารางสินค้า ซึ่งจะถูกแก้ไขอยู่ตลอดเวลาที่มีธุรกรรมการซื้อขาย แต่คอลัมน์ชื่อสินค้า คำอธิบายสินค้า และอื่นๆไม่ได้รับการแก้ไขบ่อย

การปิดกั้นข้อมูลด้วยรูปแบบต่างๆ ในระบบจัดการฐานข้อมูลสามารถดำเนินการได้ด้วยคำสั่ง SQL ซึ่งมีรูปแบบดังนี้

```
LOCK TABLE <table_name> IN <SHARE | EXCLUSIVE> MODE;
```

การจะใช้การปิดกั้นรูปแบบใดนั้นต้องคำนึงถึงวัตถุประสงค์และผลกระทบที่ผู้พัฒนาระบบฐานข้อมูลและผู้ดูแลฐานข้อมูล (DBA) ต้องพิจารณา

ตัวอย่าง

การปิดกั้นตารางนักศึกษา โดยใช้การปิดกั้นประเภท S Lock ซึ่งอนุญาตให้ธุรกรรมต่างๆ อ่านข้อมูลจากตารางนักศึกษาได้แต่ไม่สามารถแก้ไขได้ แสดงด้วย SQL ดังนี้

```
LOCK TABLE students IN SHARE MODE;
```

9.2.4.3 วิธีการดำเนินการปิดกั้น

วิธีดำเนินการปิดกั้นข้อมูลอาศัยหลักการปิดกั้นแบบ 2 ช่วง (2 Phase Locking—2PL) คือ

1) **ช่วงขยาย (Growing Phase)**

เป็นการกำหนดการปิดกั้นข้อมูลที่จะต้องทำการเข้าถึง โดยข้อมูลที่จะได้รับการแก้ไขให้ปิดกั้นแบบ X-LOCK ส่วนข้อมูลที่อ่านเพียงอย่างเดียวไม่ได้รับการแก้ไขในธุรกรรมให้ใช้ S-LOCK ระยะนี้จะไม่มีการปลดการปิดกั้นใดๆ

2) **ช่วงถดถอย (Shring Phase)**

หลังจากการดำเนินคำสั่งย่อยๆ ของธุรกรรมเสร็จสิ้นเรียบร้อยแล้ว ช่วงนี้เป็นช่วงของการปลดการปิดกั้นข้อมูลที่ได้ปิดกั้น ในช่วงนี้จะไม่มีการปิดกั้นข้อมูลเพิ่มอีก

อาศัยตัวอย่างธุรกรรม T3 และ T4 จากหัวข้อ 9.2.2** ที่นายสมชายทำการโอนเงินจำนวน 1,000 บาทระหว่างบัญชีเงินฝากออมทรัพย์และบัญชีเงินฝากประจำ และธุรกรรมนางสมศรีแสดงยอดทรัพย์สินรวมของทั้ง 2 บัญชี

ธุรกรรม T5 และT6 เป็นธุรกรรมที่ปรับเปลี่ยนจากธุรกรรม T3 และ T4 โดยการใช้เทคนิคการปิดกั้นป้องกันปัญหาที่อาจจะเกิดขึ้นจากภาวะการเกิดพร้อมกันของทั้ง 2 ธุรกรรม ดังนี้

ธุรกรรมที่ 5 (T5) นายสมชายโอนเงินจากบัญชีฝากออมพรัพทย์ (X) ไปยังบัญชีฝากประจำ (Y)

```
T5:   X-LOCK(X)
      READ(X)
      UPDATE X = X - 1000
      WRITE(X)
      X-LOCK(Y)
      READ(Y)
      UPDATE Y = Y + 1000
      WRITE(Y)
      UNLOCK(X)
      UNLOCK(Y)
```

ธุรกรรมที่ 6 (T6) นางสมศรีแสดงยอดสินทรัพย์รวม 2 บัญชี

```
T6:   S-LOCK(X)
      READ(X)
      S-LOCK(Y)
      READ(Y)
      SUM = X + Y
      UNLOCK(X)
      UNLOCK(Y)
```

ต่อไปนี้เป็นกรณีตัวอย่างการดำเนินธุรกรรม T5 และ T6 ที่เกิดขึ้นในเวลาใกล้เคียงกัน และเกิดการ interleave ในการประมวลผลของระบบจัดการฐานข้อมูลระหว่างธุรกรรมทั้ง 2

เวลา	ธุรกรรม T5	ธุรกรรม T6	ยอดเงิน (X)	ยอดเงิน (Y)	รวม (SUM)
t1	BEGIN TRANSACTION		10000	50000	60000
t2	X-LOCK (X)		10000	50000	
t3	READ (X) (10000)		10000	50000	
t4	UPDATE X = X – 1000 (9000)		10000	50000	
t5	WRITE (X) (9000)		9000	50000	59000
t6		BEGIN TRANSACTION	9000	50000	
t7		S-LOCK (X)	9000	50000	
t8	LOCK (Y)	WAIT	9000	50000	
t9	READ (Y) (50000)	WAIT	9000	50000	
t10	UPDATE Y = Y + 1000 (51000)	WAIT	9000	50000	
t11	WRITE (Y) (51000)	WAIT	9000	51000	
t12	COMMIT	WAIT	9000	51000	
t13		READ (X) (9000)	9000	51000	
t14		S-LOCK (Y)	9000	51000	
t15		READ (Y) (10000)	9000	51000	
t16		SUM = X + Y	9000	51000	60000
t17		COMMIT	9000	51000	

รูป**

ณ เวลา t2 มีการ LOCK ข้อมูล X แบบ X-LOCK ไว้ด้วยธุรกรรม T5 ซึ่งข้อมูลอาจมีการเปลี่ยนแปลงของข้อมูล จึงไม่อนุญาตให้ธุรกรรมอื่นเข้าถึงข้อมูล X หรือยอดบัญชีเงินฝากออมทรัพย์ของนายสมชายและนางสมศรีได้ ที่เวลา t5 ข้อมูลของยอดบัญชีเงินฝากออมทรัพย์เหลือเพียง 9,000 บาท ในขณะนี้เองหาธุรกรรม T6 อ่านข้อมูล X และ Y ในกรณีไม่มีการปิดกั้นข้อมูล ผลรวมของยอดเงินในบัญชีจะผิดพลาด แต่ในตัวอย่างนี้มีการใช้เทคนิคการปิดกั้น พิจารณาที่เวลา t6 มีเกิดการ interleave ของระบบจัดการฐานข้อมูลไปประมวลผลธุรกรรม T6

ณ เวลา t7 เป็นตัวอย่างสำคัญที่แสดงว่าธุรกรรม T6 ไม่สามารถอ่านข้อมูลยอดเงินในบัญชีเงินฝากได้ โดยไม่สามารถปิดกั้นข้อมูล X เพื่ออ่านได้ เนื่องจากธุรกรรม T5 ปิดกั้นข้อมูล X ไว้ตั้งแต่เวลา t2 ธุรกรรม T6 จึงไม่มีทางเลือกและต้องรอจนกว่าธุรกรรม T5 จะปลดการปิดกั้นข้อมูล X เสียก่อน ธุรกรรม T6 จึงจะสามารถดำเนินต่อไปได้ ธุรกรรม T6 สามารถดำเนินการต่อได้ที่เวลา t13 หลังจากที่ธุรกรรม T5 ปลดการปิดกั้นข้อมูลแล้ว

จากตัวอย่างการใช้เทคนิคการปิดกั้นดังกล่าวด้วยหลักการ 2PL นั้นเป็นการแก้ปัญหาความผิดพลาดของข้อมูลที่จะเกิดขึ้นจากภาวะการเกิดพร้อมกันของธุรกรรมได้ อย่างไรก็ตามเทคนิคการปิดกันแบบ 2PL นี้ ก่อให้เกิดผลกระทบข้างเคียงที่ไม่พึงประสงค์ ดังนี้

9.2.4.4 การปิดตาย (Dead Lock)

การปิดตายเป็นสภาวะที่ธุรกรรมต่างรอกันเพื่อใช้ข้อมูลที่อีกธุรกรรมหนึ่งปิดกั้นไว้ ซึ่งมีลักษณะของการดำเนินธุรกรรมดังนี้

เวลา	ธุรกรรม Ta	ธุรกรรม Tb
t1	ปิดกั้น (X)	
t2		ปิดกั้น (Y)
t3	ปิดกั้น (Y) เพื่อใช้งานข้อมูล แต่ไม่สามารถปิดกั้นได้	
t4		ปิดกั้น (X) เพื่อใช้งานข้อมูล ไม่แต่สามารถปิดกั้นได้
t5	รอ (Y)	รอ (X)
t6	รอ (Y)	รอ (X)
t7	รอ (Y)	รอ (X)

จากลักษณะการดำเนินธุรกรรมดังกล่าว ทั้งธุรกรรม Ta และ Tb ต่างก็ใช้ข้อมูลหรือระเบียน X และ Y เช่นกัน แต่ปรากฏว่า Ta สามารถปิดกั้นข้อมูล X ได้ก่อน แต่ยังไม่ทันได้ปิดกั้นข้อมูล Y ในขณะที่ธุรกรรม Tb สามารถปิดกั้นข้อมูล Y ได้ก่อน ธุรกรรมทั้ง 2 พยายามปิดกั้นข้อมูลอีกข้อมูลหนึ่งจนต้องรอไปเรื่อยๆ การรอนี้จะไม่มีทางสิ้นสุดได้ และธุรกรรมทั้ง 2 จะไม่สำเร็จ ปัญหาที่เกิดขึ้นนี้เรียกว่าปัญหาการปิดตายของธุรกรรม (transaction dead lock)

พิจารณากรณีตัวอย่างการเกิดการปิดตายของธุรกรรม T5 และ T6 ของนายสมชายและนางสมศรี โดยมีการปรับเปลี่ยนลำดับคำสั่งย่อยของการดำเนินธุรกรรมเล็กน้อยเพื่อความสะดวกในการยกตัวอย่าง โดยธุรกรรม T6 ทำการปิดกั้นข้อมูล Y ก่อน X ดังนี้

เวลา	ธุรกรรม T5	ธุรกรรม T6
t1	BEGIN TRANSACTION	
t2	X-LOCK (X)	
t3	READ (X)	
t4	UPDATE X = X – 1000	
t5	WRITE (X)	
t6		BEGIN TRANSACTION
t7		S-LOCK (Y)
		READ (Y)
t8		S-LOCK (X)
t9	LOCK (Y)	WAIT
t10	WAIT	WAIT
t11	WAIT	WAIT
t12	WAIT	WAIT

รูป

วิธีที่จะจัดการกับปัญหาการปิดตายนั้นทำได้ 2 วิธีใหญ่ๆ คือการป้องกันไว้ล่วงหน้า กับการแก้ปัญหาในกรณีที่เกิดขึ้นแล้ว

1) **การป้องกันการปิดตาย**

วิธีการป้องนการปิดตายทำได้โดยการ ปิดกั้นข้อมูลล่วงหน้าทุกข้อมูลที่จะใช้ให้หมดเสียก่อนที่จะดำเนินการใดๆ กับข้อมูลไม่ว่าจะเป็นการอ่าน แก้ไข หรือเขียน และจะต้องให้ธุรกรรมนั้นๆ ปิดกั้นข้อมูลให้เสร็จเสียก่อน วิธีการนี้ทำได้ค่อนข้างยากเนื่องจากจะต้องทราบข้อมูลทุกข้อมูลที่จะใช้งานทั้งธุรกรรม

2) **การแก้ปัญหาหลังจากเกิดการปิดตาย**

วิธีการแก้ปัญหาสามารถทำได้หลายวิธี เช่นการใช้ Wait-for-Graph (WFG) ซึ่งเป็นการเก็บข้อมูลว่าธุรกรรมใดกำลังปิดกั้นข้อมูลตัวใดอยู่ จากนั้นอาศัยการสร้างรูปกราฟความสัมพันธ์ของธุรกรรม เมื่อใดก็ตามที่รูปกราฟเป็นรูปปิดวงกลม เช่น ธุรกรรม Ta รอข้อมูล Y จากธุรกรรม Tb และธุรกรรม Tb รอข้อมูล X จากธุรกรรม Ta ก็จะเกิดเป็นกราฟวงกลม (cyclic graph) วนกลับมาที่เดิม ทำให้สามารถตัดสินใจให้ธุรกรรมใดดำเนินต่อไปก่อนได้

รูป**

9.2.5 เทคนิคอื่นๆ ที่ใช้ในการแก้ปัญหาภาวะการเกิดพร้อมกัน

ยังมีเทคนิคอื่นๆ อีกที่สามารถแก้ปัญหาภาวะการเกิดพร้อมกันได้นอกจากเทคนิคการปิดกั้น เช่น เทคนิคการทำเวอร์ชัน เทคนิค optimistic หรือเทคนิคผสมผสานกัน เป็นต้น เทคนิคอื่นๆ ที่นิยมใช้กันนั้นสามารถอธิบายได้โดยง่าย กล่าวคือ การดำเนินคำสั่งย่อยๆ ของธุรกรรมต่างๆ จะไม่มีการปิดกั้นข้อมูล แต่จะปล่อยให้ธุรกรรมดำเนินไปพร้อมๆ กัน และอาศัยกลไกสำคัญคือการจดบันทึกเวลาเริ่มต้น และเวลาที่ข้อมูลแต่ละข้อมูลถูกปรับปรุง ในกรณีที่ข้อมูลๆ หนึ่งที่กำลังจะถูกเรียกใช้ กลับถูกธุรกรรมอีกธุรกรรมหนึ่งแก้ไขข้อมูลไปแล้ว ธุรกรรมอีกธุรกรรมจะทราบว่าเกิดการแย่งข้อมูลกันขึ้นโดยดูจากเวลาที่ธุรกรรมเริ่มต้น และเวลาที่ข้อมูลที่กำลังจะถูกเรียกใช้ได้รับการแก้ไข เมื่อเกิดเหตุการณ์ดังกล่าว ธุรกรรมที่มาทีหลังจะ ROLL BACK ซึ่งการ ROLL BACK สามารถเกิดขึ้นได้ในขณะที่ธุรกรรมตรวจพบการเกิดภาวะการเกิดพร้อมกันหรือก่อนทำการ COMMIT ในขั้นตอนสุดท้าย ขึ้นอยู่กับเทคนิคที่ใช้

กลไกที่รองรับการป้องกันและแก้ไขข้อผิดพลาดของข้อมูลที่เกิดจากปัญหาการเกิดภาวะพร้อมกันนี้มีการประยุกต์ใช้ในระบบฐานข้อมูลต่างๆ กันไปตามผู้ผลิตแต่ละราย นอกจากนี้การสร้างธุรกรรมต่างๆ ผู้พัฒนาระบบควรเลือกใช้เทคนิคในการควบคุมการเกิดภาวะพร้อมกันให้สอดรับกับกลไกที่ระบบจัดการฐานข้อมูลมีมาให้

9.3 การคืนสภาพข้อมูล (Data Recovery)

ในกรณีที่ระบบจัดการฐานข้อมูลดำเนินการคำสั่งย่อยๆ ในธุรกรรมยังไม่แล้วเสร็จ มีการเปลี่ยนแปลงข้อมูลบางส่วนแล้วระบบเกิดการขัดข้อง ข้อมูลที่อยู่ในฐานข้อมูลอาจมีสภาพไม่สอดคล้องต้องกันได้ เช่นการโอนเงินจากบัญชีหนึ่งไปยังอีกบัญชีหนึ่ง หากธุรกรรมได้ทำการหักยอดบัญชีต้นทางไปแล้ว แต่ระบบยังไม่ได้ทำการเพิ่มยอดที่บัญชีปลายทาง แต่เกิดความผิดพลาดของระบบเสียก่อน เช่นไฟดับ หรือเครื่องล้มเหลว (crash) เป็นต้น ในที่นี้เราต้องสร้างกลไกในการรองรับธุรกรรมเพื่อที่จะกำหนดว่าเมื่อระบบจัดการฐานข้อมูลกลับมาทำงานได้ตามปกติแล้วจะจัดการอย่างไรกับธุรกรรมที่ยังทำค้างไว้ ซึ่งเรียกว่าการคืนสภาพข้อมูล (data recovery)

วิธีการคืนสภาพข้อมูลที่เกิดจากการขัดข้องของระบบจากสาเหตุต่างๆ นั้น อาศัยหลักการที่คล้ายคลึงกันแต่ยังมีรายละเอียดที่แตกต่างกันบ้าง ขึ้นอยู่กับสาเหตุของความขัดข้องนั้นๆ ในหัวข้อนี้จะได้อธิบายถึงสาเหตุของความขัดข้องประเภทต่างๆ และวิธีการที่ใช้ในการคืนสภาพข้อมูลจากสาเหตุแต่ละประเภท

9.3.1 ประเภทของความขัดข้อง

สาเหตุที่ทำให้เกิดความขัดข้องของธุรกรรมแบ่งเป็น 3 ประเภทใหญ่ๆ ได้แก่

1) **ความขัดข้องที่เกิดจากระบบคอมพิวเตอร์ล้มเหลว (System Crash or Failure)**
เป็นความผิดพลาดของฮาร์ดแวร์หรือซอฟต์แวร์ที่ทำให้เกิดความผิดพลาดระหว่างการดำเนินการธุรกรรมใดๆ เช่นการเขียนทับกันของข้อมูลในหน่วยความจำของเครื่อง ข้อมูลบางส่วนในหน่วยความจำหายไป หรือไฟดับและไม่มีอุปกรณ์สำรองไฟ เป็นต้น

2) **ความขัดข้องที่เกิดจากข้อผิดพลาดของธุรกรรม (Transaction Error)**
เป็นความผิดพลาดที่เกิดจากคำสั่งในธุรกรรมเอง เช่นความผิดพลาดที่เกิดจากการไม่ได้ตรวจสอบชนิดของข้อมูลนำเข้าเสียก่อน มีการนำตัวอักษรมาคำนวณเหมือนตัวเลข หรือการหารค่าด้วย 0 เป็นต้น

3) **ความขัดข้องที่เกิดจากอุปกรณ์จัดเก็บข้อมูลหรือสื่อผิดพลาด (Storage or Media Error)**
เป็นความผิดพลาดที่เกิดจากอุปกรณ์จัดเก็บ เช่นฮาร์ดดิสก์/จานบันทึกข้อมูล ได้รับความเสียหาย เขียนอ่านข้อมูลไม่ได้บนผิวจานบันทึกบางช่วงที่จัดเก็บฐานข้อมูล เป็นต้น

9.3.2 วิธีการคืนสภาพข้อมูล

วิธีการคืนสภาพข้อมูลขึ้นอยู่กับสาเหตุของการเกิดการขัดข้องแต่ละประเภท ดังนี้

1) **การคืนสภาพข้อมูลจากความขัดข้องที่เกิดจากระบบคอมพิวเตอร์ล้มเหลว (Data Recovery from System Crash or Failure)**

ความล้มเหลวของระบบอาจทำให้ธุรกรรมเปลี่ยนแปลงข้อมูลในอุปกรณ์จัดเก็บไปแล้วบางส่วน แต่คำสั่งย่อยที่เหลือในธุรกรรมยังไม่แล้วเสร็จ แต่ข้อมูลที่ใช้ในธุรกรรม ตลอดจนสถานะของธุรกรรมในหน่วยความจำสูญหายไปแล้ว

เทคนิคที่เรานิยมใช้ในการจัดการกับธุรกรรมที่ขัดข้อง ได้แก่เทคนิคที่เรียกว่าการลงบันทึกระบบ/ลงบันทึกรายการ (system log) ซึ่งสิ่งที่บันทึกนี้เป็นค่าต่างๆ เกี่ยวกับระบบ เช่น กิจกรรม/คำสั่ง/การดำเนินการกับข้อมูล ข้อมูลในแต่ละขั้นตอนของธุรกรรม หรือความสำเร็จ/ล้มเหลวของขั้นตอนแต่ละขั้นตอน เป็นต้น

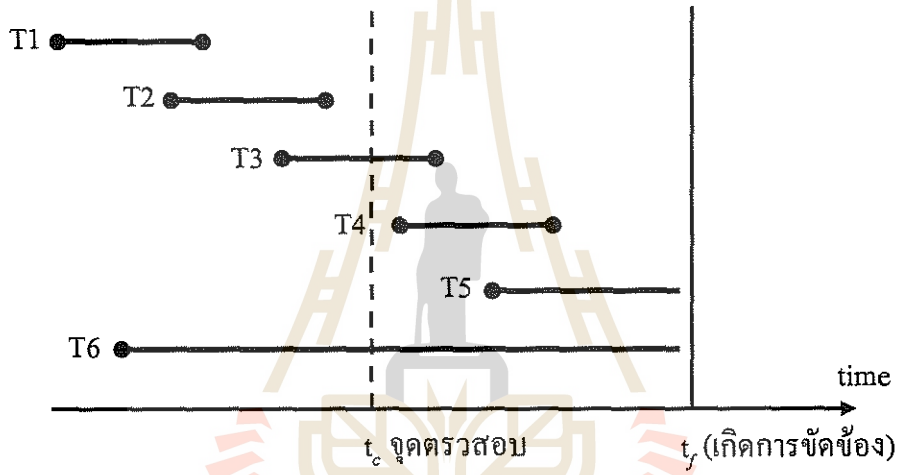
<u>การลงบันทึกระบบ (System Log)</u>

เทคนิคการลงบันทึกระบบนี้มีหลักการทำงานง่ายๆ ในแนวคิดกล่าวคือระบบจัดการฐานข้อมูลจะทำการเริ่มลงบันทึกตั้งแต่เริ่มธุรกรรม บันทึกกิจกรรมที่กระทำกับกับข้อมูล และที่สำคัญคือการบันทึกข้อมูลแต่ละข้อมูลที่ถูกแก้ไขทั้งก่อนและหลังการแก้ไขในฐานข้อมูล การใช้เทคนิค log เพื่อการคืนสภาพข้อมูลมีหลักการง่ายๆ คือเมื่อมีการล้มเหลวของระบบ log จะมีข้อมูลของธุรกรรมที่ยังดำเนินการไม่เสร็จสิ้น ก็ให้ทำการ ROLL BACK ธุรกรรมที่ยังไม่เสร็จสิ้นทั้งหมด ส่วนการคืนสภาพจากข้อมูลสำรอง ก็นำข้อมูลสำรองขึ้นมาใช้ แล้วดำเนินธุรกรรมตาม log ที่เกิดขึ้นหลังจากการสำรองข้อมูลนั้น ในทางเทคนิคแล้ว การประยุกต์ใช้ log ยังอาศัยกลไกอีกกลไกหนึ่งที่เรียกว่าการทำจุดตรวจสอบ (checkpoiting) จุดตรวจสอบนี้เป็นกลไกในการจับคู่ (synchronize) log กับสถานะของระบบจัดการฐานข้อมูลให้สัมพันธ์กัน ขณะสร้างจุดตรวจสอบ ระบบจัดการฐานข้อมูลจะบันทึกข้อมูลที่อยู่ในหน่วยความจำหลักของเครื่องไว้ด้วย เพื่อให้เป็นการแน่ใจว่า log ที่บันทึกไว้ ณ จุดตรวจสอบมีความสัมพันธ์กับสถานะจริงๆ ของฐานข้อมูล การกระทำเช่นนี้คล้ายๆ กับการสร้าง hibernate file ของคอมพิวเตอร์สมัยใหม่ ที่เมื่อทำการ hibernate แล้ว เครื่องจะเขียนข้อมูลในหน่วยความจำหลักทั้งหมดลงฮาร์ดดิสก์แล้วปิดเครื่อง เมื่อเปิดเครื่องขึ้นมาข้อมูลหน่วยความจำที่เขียนไว้จะถูกอ่านเก็บไว้ที่หน่วยความจำหลักเพื่อให้เครื่องพร้อมใช้งานได้ทันทีต่อจากสถานะเดิม เสมือนเปิดเครื่องคอมพิวเตอร์อยู่ตลอดเวลา

หลักการของการคืนสภาพข้อมูลโดยการใช้ log มีอยู่ว่า

- ธุรกรรมใดที่ยังดำเนินการไม่สำเร็จก่อนที่จะเกิดการขัดข้องขึ้น แน่นอนว่าให้ยกเลิกธุรกรรมนั้น (UNDO/ROLL BACK)
- ธุรกรรมใดที่ดำเนินการสำเร็จ (COMMIT) ก่อนเวลาสำหรับจุดตรวจสอบ จากนั้นเกิดการขัดข้องขึ้น ธุรกรรมนั้นไม่ต้องทำใหม่ (ไม่ต้อง REDO) เนื่องจากเสมือนเราได้ยอมรับธุรกรรมนั้นแล้ว
- ธุรกรรมใดที่เริ่มดำเนินการก่อนจุดตรวจสอบ แต่สำเร็จ (COMMIT) หลังจากจุดตรวจสอบ จากนั้นเกิดการขัดข้องขึ้น ธุรกรรมประเภทนี้ต้องยกเลิกและทำใหม่ (ROLLBACK & REDO)

พิจารณาภาพจำลองการดำเนินธุรกรรมดังนี้



แผนภาพดังกล่าวมีลักษณะคล้ายคลึงกับ Gantt Chart ซึ่งเป็นการแสดงว่าธุรกรรมแต่ละธุรกรรมเริ่มต้นที่เวลาใด ดำเนินธุรกรรม และสิ้นสุดที่เวลาใด โดยธุรกรรมเริ่มจากเวลาที่อยู่ทางซ้ายของแผนภาพ t_c คือเวลาที่มีการสร้างจุดตรวจสอบ t_f คือเวลาที่เกิดการขัดข้องของธุรกรรม

ธุรกรรม T1 และ ธุรกรรม T2 เกิดขึ้นและสำเร็จเรียบร้อยแล้ว ที่จุดตรวจสอบมีการบันทึกข้อมูลสถานะของระบบเก็บไว้ เมื่อเกิดการขัดข้องขึ้นที่เวลา t_f ธุรกรรมทั้ง 2 จึงไม่จำเป็นต้องทำใหม่หรือไม่จำเป็นต้องดำเนินการใดๆ กับธุรกรรมทั้ง 2

ธุรกรรม T3 นั้น แม้ว่าจะเริ่มก่อนจุดตรวจสอบ แต่จะมีสถานะเช่นเดียวกับธุรกรรม T4 ที่ COMMIT แล้ว แม้ว่าธุรกรรมทั้ง 2 สำเร็จแล้วแต่ธุรกรรมทั้ง 2 นี้ มีรายละเอียดปลีกย่อยที่จะต้องดำเนินการ กล่าวคือธุรกรรมต่างๆ เกิดขึ้นพร้อมๆ กัน เราไม่ทราบว่าธุรกรรม T3 และ T4 ได้แก้ไขข้อมูลอะไรในหน่วยความจำหลักไว้บ้าง ซึ่งอาจมีผลกระทบกับธุรกรรมที่กำลังดำเนินการอยู่จึงเป็นที่มาของการสร้างจุดตรวจสอบเก็บสถานะของระบบจัดการฐานข้อมูล

วิธีการจัดการกับธุรกรรมทั้ง 2 หลังจากแก้ปัญหาการขัดข้องของระบบได้แล้วก็คือ ระบบจัดการฐานข้อมูลจะทำการหาจุดอ้างอิงที่สามารถคืนสภาพได้ ซึ่งก็คือจุดตรวจสอบ t_c นั่นเอง ธุรกรรม T3 และ T4 ควรดำเนินการใหม่อีกครั้งเพื่อให้ระบบมีสถานะเหมือนกับระบบที่ทำงานอยู่ ณ จุดตรวจสอบ โดยระบบจะทำการยกเลิก ROLL BACK และทำซ้ำ REDO ธุรกรรม T3 และ T4 (สามารถเปรียบเทียบได้กับการ hibernate ของเครื่องคอมพิวเตอร์ที่ได้อธิบายไว้ข้างต้น กิจกรรมใดที่ดำเนินอยู่ควรทำให้สำเร็จเสมือนเครื่องคอมพิวเตอร์ไม่ได้ปิดหรือไม่ได้ขัดข้อง)

ธุรกรรม T5 และ T6 ยังไม่แล้วเสร็จเมื่อเกิดการขัดข้องของเครื่อง เราจึงไม่ทราบว่าธุรกรรมดังกล่าวจะสำเร็จหรือไม่หากไม่เกิดการขัดข้อง อาจเกิด ROLL BACK จากสาเหตุต่างๆ ธุรกรรมทั้ง 2 จึงจะต้องถูก ROLL BACK

2) การคืนสภาพข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของธุรกรรม (Data Recover from Transaction Error)

การคืนสภาพข้อมูลจากความขัดข้องที่เกิดจากข้อผิดพลาดของธุรกรรมนี้ไม่ค่อยซับซ้อน กล่าวคือธุรกรรมใดก็ตามที่ยังดำเนินการไม่เสร็จสิ้น และยังไม่ได้ยอมรับ (COMMIT) แต่เกิดข้อผิดพลาดขึ้นเสียก่อน ไม่ว่าจะด้วยสาเหตุใดก็ตาม เช่น การ ROLL BACK จากปัญหาภาวะการเกิดพร้อมกัน เป็นต้น แต่ความขัดข้องนี้ระบบจัดการฐานข้อมูลยังทำงานต่อไปได้ ระบบจัดการฐานข้อมูลจะทำการคืนสภาพโดยการเลิกทำหรือย้อนกลับ (UNDO or ROLL BACK) ให้ข้อมูลอยู่ในสภาพก่อนหน้าที่จะมีธุรกรรม การจะทำซ้ำธุรกรรมนั้นหรือไม่ขึ้นอยู่กับข้อกำหนดในการใช้งานระบบ

3) การคืนสภาพข้อมูลจากความขัดข้องที่เกิดจากอุปกรณ์จัดเก็บข้อมูลหรือสื่อผิดพลาด (Data Recovery from Storage or Media Error)

ความขัดข้องประเภทนี้มักจะมีความเสียหายของข้อมูลมากกว่า 2 ประเภทแรก และมีความซับซ้อนในการกู้คืนสภาพข้อมูล กล่าวคือเราไม่สามารถที่จะดำเนินการเพียงนำ log ของธุรกรรมมาดำเนินการทำซ้ำหรือยกเลิกขั้นตอนใดๆ เท่านั้น เพราะข้อมูลบางส่วนที่อยู่ในอุปกรณ์จัดเก็บสูญหายไปแล้ว วิธีการกู้คืนสภาพข้อมูลทำได้โดยการนำข้อมูลที่ได้ทำการสำรอง (backup) ไว้ มาใช้งานจากนั้นทำการคืนสภาพข้อมูลสำรองให้มีค่าของข้อมูลใกล้เคียงกับข้อมูลล่าสุดมากที่สุดโดยการดำเนินการตาม log ที่ได้บันทึกไว้ตั้งแต่สำรองข้อมูล จะเห็นว่า log ของระบบจัดการฐานข้อมูลมีความสำคัญมาก ไฟล์ log นี้มักจะสร้างไว้หลายชุด และจัดเก็บไว้หลายๆ ที่เพื่อกระจายความเสี่ยงต่อภัยคุกคามต่างๆ

9.4 ความมั่นคงปลอดภัยของข้อมูล

ความมั่งคงปลอดภัยของข้อมูลเป็นเรื่องที่เกี่ยวกับการสร้างกลไกในการป้องกันข้อมูลจากภัยคุกคามต่างๆ ที่ไม่เพียงแต่จะก่อให้เกิดความเสียหายต่อตัวข้อมูลเอง แต่รวมถึงเป็นความเสียหายที่เป็นผลพวงที่เกิดกับข้อมูลจากภัยคุกคามนั้นๆ วัตถุประสงค์ของการป้องกันภัยคุกคามก็เพื่อรักษาความถูกต้องหรือความมีบูรณาภาพของข้อมูล (integrity) เพื่อคงไว้ซึ่งความลับของข้อมูลที่ปกปิด (confidential) และเพื่อให้ข้อมูลมีอยู่ให้ใช้งานเมื่อต้องการ (availability) ในหัวข้อนี้จึงจะได้กล่าวถึงภัยคุกคามที่สำคัญ และวิธีการแต่ละวิธีที่จะป้องกันภัยคุกคามได้อย่างมีประสิทธิภาพ

9.4.1 ภัยคุกคาม

ภัยคุกคามคือการกระทำใดๆ ที่เกิดจากความตั้งใจหรืออาจไม่ได้ตั้งใจ รวมถึงเหตุการณ์ที่เป็นภัยต่างๆ ที่เกิดขึ้นแล้วสร้างความเสียหายให้กับข้อมูล และแน่นอน ย่อมต้องมีผลเสียหายต่อเจ้าของข้อมูลและผู้ใช้ข้อมูล ต่อไปนี้จะเป็นการอธิบายภัยคุกคามรูปแบบต่างๆ ซึ่งเมื่อทราบถึงภัยคุกคามแล้วเราจะสามารถดำเนินการป้องกันภัยคุกคามที่อาจจะเกิดขึ้นได้

ภัยคุกคามมีหลากหลายรูปแบบมาก และมีรูปแบบใหม่ๆ เกิดขึ้นอยู่พอสมควร เราสามารถจัดกลุ่มภายคุกคามออกได้หลายรูปแบบ ในที่นี้จะแจกแจงภัยคุกคามออกโดยอาศัยสาเหตุของการเกิดภัยคุกคาม ได้แก่ มนุษย์ ฮาร์ดแวร์ ซอฟต์แวร์ และ ภัยพิบัติ ดังนี้

1) มนุษย์

เป็นภัยคุกคามที่เกิดจากมนุษย์ ทั้งที่ตั้งใจและไม่ได้ตั้งใจ ภัยคุกคามจากมนุษย์ที่สำคัญได้แก่

- การเข้าใช้ระบบโดยบัญชีผู้ใช้ของผู้อื่น
  หมายถึงการที่ผู้ไม่ประสงค์ดีทำการวิธีการใดๆ ในการได้มาซึ่งบัญชีในการเข้าใช้ระบบของผู้อื่นพร้อมรหัสผ่าน เช่นได้จากการแอบสังเกต การเจาะระบบ (hack) การขโมยข้อมูลที่บันทึกไว้ การได้บัญชีมาโดยบังเอิญ การใช้ซอฟต์แวร์ช่วยในการขโมยบัญชีผู้ใช้ ฯลฯ ซึ่งสามารถทำให้บุคคลที่ไม่สมควรได้รับอนุญาตเข้าถึงข้อมูลได้ ซึ่งความเสียหายที่เกิดขึ้นของภัยคุกคามประเภทนี้นั้นอาจมากมายเกินกว่าจะประเมินได้ เพราะเจ้าของบัญชีอาจจะไม่ทราบเลยก็ได้ว่าข้อมูลถูกผู้อื่นใช้งาน สูญเสียความเป็นส่วนตัว ข้อมูลอาจนำไปใช้เพื่อความได้เปรียบของคู่แข่ง หรือแม้แต่ถูกทำให้เสียหาย
- การเจาะระบบ (hack)
  การเจาะระบบคือความพยายามใดๆ ที่มีเจตนาจะเข้าถึงข้อมูลหรือเป็นผู้ใช้ของระบบให้ได้ โดยเน้นที่กิจกรรมการดำเนินการทางซอฟต์แวร์ที่จะได้มาซึ่งช่องทาง บัญชีผู้ใช้ ฯลฯ ในการเข้าถึงข้อมูล การเจาะระบบสามารถใช้วิธีการได้หลายวิธีได้แก่การอาศัยช่องว่างของระบบเพื่อหาทำเข้าควบคุมระบบ การเดาบัญชีและรหัสผู้ใช้ การใช้ซอฟต์แวร์ในการดักจับข้อมูลบัญชีผู้ใช้ การใช้ซอฟต์แวร์จำพวกโทรจัน (Trojan) ในการสร้างช่องทางการเข้าโจมตีระบบเป็นต้น ผู้ที่เจาะระบบสำเร็จสามารถดำเนินกิจกรรมใดๆ ตามที่ต้องการในระดับที่เจาะระบบได้
- การใช้งานเกินสิทธิของตัวเองโดยบังเอิญ
  ในบางกรณีผู้ดูแลระบบกำหนดสิทธิของผู้ใช้ไว้ผิดพลาด เช่น ผู้ใช้บางคนสามารถเข้าถึงข้อมูลตารางพนักงานได้แต่ผู้ดูแลและผู้พัฒนาระบบไม่ได้กำหนดสิทธิป้องกันไม่ให้เข้าถึงข้อมูลเงินเดือนของพนักงานคนอื่น เป็นต้น
- การลักลอบดักจับข้อมูลในเครือข่ายการสื่อสาร

เป็นการใช้ซอฟต์แวร์คอยดักจับข้อมูล หรือแม้แต่นำอุปกรณ์เชื่อมต่อเข้ากับเครือข่ายเพื่อดักจับข้อมูลต่างๆ ในเครือข่ายคอมพิวเตอร์ ข้อมูลที่ได้อาจจะเป็นข้อมูลที่สำคัญ รวมถึงอาจเป็นข้อมูลในการเข้าใช้ระบบ

- การเข้าถึงระบบทางกายภาพ
  หมายถึงการเข้าถึงตัวฮาร์ดแวร์ของระบบสารสนเทศ ฐานข้อมูล หรือระบบเครือข่ายในทางกายภาพ อาจมีการทำลายอุปกรณ์เหล่านี้ให้เสียหาย ขโมยอุปกรณ์ หรือความพยายามที่จะเจาะระบบ
- การโจมตีเพื่อให้ระบบไม่สามารถให้บริการได้
  ระบบฐานข้อมูลควรที่จะสามารถให้บริการได้อยู่ตลอดเวลาที่ต้องการ แต่ผู้ไม่ประสงค์ดีอาจจะอาศัยการร้องขอข้อมูลจากระบบมากเกินความสามารถที่ระบบจะรองรับได้ ทำให้ระบบไม่สามารถให้บริการได้ในที่สุด

2) ฮาร์ดแวร์

เป็นภัยคุกคามที่เกิดจากความเสียหายที่เกิดจากตัวอุปกรณ์ทางด้านกายภาพ

- ระบบคอมพิวเตอร์ ระบบการสื่อสาร หรืออุปกรณ์จัดเก็บเสียหาย
  เป็นเรื่องปกติที่อุปกรณ์ฮาร์ดแวร์ใช้งานแล้วจะเสื่อสภาพตามอายุการใช้งาน หรือความผิดปกติของอุปกรณ์จากการผลิต การขนส่ง หรือสภาพแวดล้อมและกิจกรรมอื่นๆ ทำให้อุปกรณ์เสียหาย ทำให้ข้อมูลสูญหาย ระบบไม่สามารถใช้งานได้ หรือเกิดช่องว่างของความมั่นคงปลอดภัยของระบบ

3) ซอฟต์แวร์

ภัยคุกคามที่เกิดจากซอฟต์แวร์นั้น โดยแท้จริงแล้วก็เป็นผลจากการกระทำของมนุษย์ที่เป็นผู้พัฒนาซอฟต์แวร์เช่นเดียวกัน ภัยคุกคามที่เกิดจากซอฟต์แวร์ที่สำคัญได้แก่

- ไวรัสคอมพิวเตอร์ หรือโปรแกรมประสงค์ร้าย (computer virus, malware, worm, etc.)
  เป็นซอฟต์แวร์ที่อาศัยวิธีต่างๆ ในการแพร่กระจายไปยังระบบคอมพิวเตอร์อื่นๆ แล้วสร้างความเสียหายให้กับข้อมูล ลบและเปลี่ยนแปลงข้อมูลโดยไม่ได้รับอนุญาต ตลอดจนสร้างช่องทางในการเข้าใช้ระบบจากผู้ไม่ประสงค์ดี
- ความผิดพลาดของตัวระบบสารสนเทศ หรือ ระบบจัดการฐานข้อมูลเอง
  ได้แก่ความผิดพลาดในการดำเนินงานของระบบ เช่นการเขียนและอ่านข้อมูลจากหน่วยความจำหลักที่ไม่มีข้อมูลอยู่ หรือความผิดพลาดจากการพัฒนาโค้ดโปรแกรมทำให้การจัดการข้อมูลผิดพลาดและระบบความมั่นคงปลอดภัยมีช่องโหว่
- ความล้มเหลวของระบบรักษาความมั่นคงปลอดภัย

ซอฟต์แวร์หลายประเภท รวมถึงหน่วยย่อยในระบบจัดการฐานข้อมูลบางหน่วยถูกออกแบบมาสำหรับรักษาความมั่นคงปลอดภัยของข้อมูล เช่น หน่วยที่ใช้ควบคุมการเข้าใช้ระบบ หรือซอฟต์แวร์ป้องกันการเจาะระบบ เกิดทำงานล้มเหลว ทำให้ผู้ไม่ประสงค์ดีเข้าโจมตีระบบและข้อมูลได้

4) ภัยพิบัติ

เป็นภัยคุกคามที่เกิดขึ้นจากอุบัติเหตุหรือการกระทำที่ไม่อาจควบคุมได้

- ไฟฟ้าดับหรือไฟกระชาก (power outage or power surge)
  ทำให้ธุรกรรมขัดจังหวะ หรืออุปกรณ์เสียหาย
- ไฟไหม้
  ซึ่งอาจจะเกิดจากการวางเพลิง ไฟฟ้าลัดวงจร ฯลฯ ทำให้อุปกรณ์ฮาร์ดแวร์และข้อมูลเสียหาย
- ภัยธรรมชาติ
  น้ำท่วม แผ่นดินไหว ฯลฯ ซึ่งภัยธรรมชาติบางประเภทยากที่จะควบคุม
- ภัยจากการก่อการร้าย
  ภัยจากการก่อการร้าย และภัยสงครามที่ไม่ได้สร้างความเสียหายให้กับข้อมูลเท่านั้น

ภัยคุกคามที่กล่าวมาทั้งหมดสามารถสร้างความเสียหายให้กับการใช้ข้อมูลได้มากน้อยต่างกัน แต่จะมีผลอย่างหนึ่งอย่างใดที่จะทำให้ไม่เกิดความมั่นคงปลอดภัยของข้อมูล ได้แก่การรั่วไหลของข้อมูล ข้อมูลไม่มีความเป็นบูรณภาพ ข้อมูลผิดจากที่ควรจะเป็น หรือข้อมูลเสียหาย และการไม่สามารถใช้ระบบฐานข้อมูลได้ ซึ่งเราสามารถป้องกันภัยคุกคามต่างๆ ได้ดังนี้

9.4.2 การป้องกันภัยคุกคาม

วิธีรักษาความมั่งคงปลอดภัยของระบบฐานข้อมูลที่นิยมปฏิบัติได้แก่

1) การให้สิทธิในการเข้าใช้ข้อมูลและยืนยันบุคคลผู้ใช้ (Authorization and Authentication)

การให้สิทธิในการเข้าใช้ข้อมูล (authorization) คือการให้สิทธิกับผู้ใช้ที่กำหนดให้มีสิทธิในการเข้าใช้ข้อมูลหรือวัตถุใดๆ ในฐานข้อมูล เช่นการสร้างบัญชีผู้ใช้ระบบ ทั้งนี้ยังต้องอาศัยกลไกการยืนยันบุคคลผู้ใช้เมื่อจะใช้สิทธิที่ได้รับขณะเข้าใช้ระบบที่เรียกว่าการยืนยันบุคคลผู้ใช้ (authentication) เช่น การล็อกอินเข้าใช้ระบบของผู้ใช้โดยตรวจสอบชื่อผู้ใช้กับรหัสผ่านนั่นเอง

2) การควบคุมการเข้าถึง (Access Controls)

นอกจากการให้สิทธิในการเข้าใช้ข้อมูลแล้ว เรายังสามารถกำหนดระดับของข้อมูล และกิจกรรมที่กระทำกับข้อมูลนั้นได้โดยการควบคุมการเข้าถึง เช่น การกำหนดให้ฝ่ายบุคคลสามารถอ่าน แก้ไข และเพิ่มข้อมูลของพนักงานได้ในขณะที่ผู้ใช้ทั่วไปสามารถแสดงข้อมูลได้เพียงอย่างเดียวเท่านั้น เราสามารถกำหนดสิทธิในการดำเนินกิจกรรมของ

ผู้ใช้ฐานข้อมูลบนตารางแต่ละตารางได้ โดยระบบจัดการฐานข้อมูลเชิงสัมพันธ์รองรับการกำหนดสิทธิด้วยคำสั่ง SQL GRANT (ให้สิทธิ) และ REVOKE (ยกเลิกหรือเพิกถอนสิทธิ) ซึ่งมีรูปแบบอย่างง่ายของคำสั่งดังนี้

การให้สิทธิ

GRANT <SELECT | INSERT | UPDATE | DELETE>

TO <user_name> ON <table_name>

โดย

GRANT	คือคำสั่งที่ใช้ในการให้สิทธิ
<SELECT|INSERT…>	คือกิจกรรมที่ต้องการจะให้สิทธิแก่ผู้ใช้ เราสามารถให้สิทธิหลายๆ สิทธิพร้อมๆ กันได้โดยใช้เครื่องหมายจุลภาค “,” ขั้น กิจกรรมที่ต้องการจะให้สิทธิ
TO	ประกอบคำสั่ง GRANT เพื่อระบุว่าให้สิทธิ “แก่” ผู้ใช้คนใด
user_name	ชื่อผู้ใช้ฐานข้อมูลที่สร้างขึ้นไว้แล้ว
ON	ประกอบคำสั่ง GRANT เพื่อระบุว่าให้สิทธิ “บน” ตารางใด
table_name	ชื่อตารางผู้ใช้ได้รับสิทธิให้ทำกิจกรรมนั้นๆ

สามารถเขียนรูปแบบคำสั่งการให้สิทธิที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

ให้สิทธิ <แสดง|แทรก|แก้ไข|ลบ> แก่ ชื่อผู้ใช้ บน ชื่อตาราง

ตัวอย่าง การให้สิทธิ

ผู้ดูแลระบบสร้างบัญชีผู้ใช้สำหรับข้อมูลทะเบียนนักศึกษาได้แก่ registra การให้สิทธิ registra ในการ SELECT, INSERT, UPDATE และ DELETE ข้อมูลบนตารางลงทะเบียน Enrolled ได้ด้วยคำสั่งดังนี้

GRANT SELECT, INSERT, UPDATE, DELETE TO registra ON enrolled

การเพิกถอนสิทธิ

REVOKE <SELECT | INSERT | UPDATE | DELETE>

FROM <user_name> ON <table_name>

โดย

REVOKE	คือคำสั่งที่ใช้ในการเพิกถอนสิทธิ
<SELECT|INSERT…>	คือกิจกรรมที่ต้องการจะให้สิทธิแก่ผู้ใช้ เราสามารถให้สิทธิหลายๆ สิทธิพร้อมๆ กันได้โดยใช้เครื่องหมายจุลภาค “,” ขั้น กิจกรรมที่ต้องการจะให้สิทธิ
FROM	ประกอบคำสั่ง REVOKE เพื่อระบุว่าเพิกถอนสิทธิ “จาก” ผู้ใช้คนใด
user_name	ชื่อผู้ใช้ฐานข้อมูลที่สร้างขึ้นไว้แล้ว
ON	ประกอบคำสั่ง REVOKE เพื่อระบุว่าให้สิทธิ “บน” ตารางใด
table_name	ชื่อตารางผู้ใช้ได้รับสิทธิให้ทำกิจกรรมนั้นๆ

สามารถเขียนรูปแบบคำสั่งการให้สิทธิที่สามารถเข้าใจได้ง่ายขึ้นเทียบกับภาษาไทยได้ดังนี้

เพิกถอนสิทธิ <แสดง|แทรก|แก้ไข|ลบ> จาก ชื่อผู้ใช้ บน ชื่อตาราง

ตัวอย่าง การเพิกถอนสิทธิ

การเพิกถอนสิทธิ registra ในการ INSERT, UPDATE และ DELETE ข้อมูลบนตารางคณาจารย์ Faculty ใช้คำสั่ง

REVOKE INSERT, UPDATE, DELETE FROM registra ON faculty

คำสั่ง GRANT และ REVOKE คือคำสั่งในกลุ่มที่ใช้นิยามข้อมูล แต่สามารถแยกออกมาเป็นกลุ่มเฉพาะนอกเหนือจาก DDL และ DML ได้ เรียกว่าภาษาควบคุม (control language) กิจกรรมที่สามารถให้และเพิกถอนสิทธิได้อาจมีมากกว่านี้ต่างกันไปตามระบบจัดการฐานข้อมูล เช่น INDEX คือการให้สิทธิในการสร้างดัชนีของตาราง

3) การใช้วิว Views

วิวคือการสร้างตารางข้อมูลจำลองเสมือนหน้าต่างครอบตารางข้อมูลที่มีอยู่จริง เช่นการสร้างตาราง faculty_public ซึ่งเป็นตารางข้อมูลคณาจารย์ที่ไม่มีคอลัมน์เงินเดือน เป็นการรักษาความลับของข้อมูลที่ควรปกปิด วิวได้กล่าวโดยละเอียดถึงการใช้งานในหัวข้อ 7.7 ในบทที่ 7

4) การสำรองและคืนสภาพข้อมูล Backup and Recovery

การสำรองข้อมูลคือการทำสำเนาของข้อมูลไว้เป็นประจำตามระยะเวลาที่กำหนด เช่น ทุกๆ วัน หรือทุกๆ สัปดาห์ เป็นต้น ซึ่งข้อมูลที่ทำสำเนาได้แก่ตัวฐานข้อมูล log หรือแม้แต่ตัวโปรแกรมประยุกต์ที่ใช้กับระบบฐานข้อมูล การสำรองข้อมูลควรจัดเก็บสำเนาไว้ในอุปกรณ์จัดเก็บอีกชุดหนึ่ง ทั้งนี้สามารถทำสำเนาไว้ได้หลายสำเนา และจัดเก็บไว้ในสถานที่ต่างๆ กันก็ได้ การสำรองข้อมูลนี้สามารถ

5) การเข้ารหัส Encryption

การเข้ารหัสหมายถึงการทำให้ข้อมูลอยู่ในรูปแบบที่อ่านเข้าใจไม่ได้ และไม่มีโปรแกรมใดๆ จะถอดรหัสให้ข้อมูลกลับมาอยู่ในรูปแบบของข้อมูลจริงก่อนการถอดรหัสนอกเสียจากว่าจะใช้คีย์ในการถอดรหัส เนื่องจากระบบจัดการฐานข้อมูลอาจจัดเก็บข้อมูลที่มีความคำสัญเป็นอย่างมากต่อบริษัท เช่นข้อมูลบัตรเครดิตของลูกค้า (ซึ่งปัจจุบันไม่ค่อยได้จัดเก็บไว้เต็มรูปแบบ) หรือข้อมูลความลับทางการค้าต่างๆ ฯลฯ ข้อมูลเหล่านี้จะมีความปลอดภัยระดับหนึ่งในกรณีที่มีผู้เจาะเข้ามาในระบบและทำการขโมยข้อมูล จะไม่สามารถนำข้อมูลไปใช้ได้โดยง่าย การเข้ารหัสสามารถทำได้ทั้งการเข้ารหัสบนฐานข้อมูลและการเข้ารหัสระหว่างการติดต่อสื่อสารในระบบ

6) การใช้เงื่อนไขบังคับบูรณภาพ

การใช้เงื่อนไขบังคับบูรณภาพเป็นการกำหนดมาตรการของการรักษาความมั่นคงปลอดภัยโดยตรงต่อความมีบูรณภาพของข้อมูล เงื่อนไขบูรณภาพอธิบายไว้อย่างละเอียดในบทที่ 4 หัวข้อ 4.3

7) การใช้ RAID

RAID (Redundant Array of Independent Disks) คือการใช้ฮาร์ดดิสก์ที่เป็นอิสระต่อกันหลายๆ ตัวเพื่อจัดเก็บข้อมูล ซึ่ง RAID นั้นมีหลายระดับ เช่น RAID0 ข้อมูลที่ถูกจัดเก็บและอ่านจากฮาร์ดดิสก์จะถูกแบ่งออกเป็นช่วงๆ ในปริมาณเท่าๆ กัน ข้อมูลแต่ละช่วงจะถูกจัดเก็บลงในฮาร์ดดิสก์พร้อมๆ กัน และอ่านขึ้นมาพร้อมๆ กันขนานกันไปทำให้การจัดการข้อมูลเป็นไปอย่างรวดเร็ว RAID1 เป็นการใช้ RAID ที่มีการสร้างข้อมูลเหมือนกันจำนวน 2 ชุด ซึ่งลักษณะของการใช้ RAID แบบนี้เองที่จะสามารถเพิ่มความมั่นคงปลอดภัยของข้อมูลได้ในกรณีที่ฮาร์ดดิสก์ตัวหนึ่งเกิดเสียหายเรายังมีข้อมูลอีกชุดหนึ่งไว้ใช้งานได้ทันที อย่างไรก็ตาม RAID1 นี้จะต้องใช้ฮาร์ดดิสก์ที่คุสิ้นเปลืองเนื่องจากจะต้องใช้ฮาร์ดแวร์เป็น 2 เท่า จึงมีการคิดค้น RAID แบบอื่นๆ อีกที่สามารถลดจำนวนอุปกรณ์จัดเก็บลงได้ แต่อาจแลกมาด้วยประสิทธิภาพที่ลดลงเพราะต้องสูญเสียไปกับการประมวลผล RAID ที่นิยมใช้ได้แก่ RAID0, RAID1 และ RAID5

8) การป้องกันทางกายภาพ

การป้องกันทางกายภาพเป็นการป้องกันอุปกรณ์ฮาร์ดแวร์ต่างๆ จากภัยคุกคามที่จะเข้าถึงตัวฮาร์ดแวร์ได้โดยตรง เช่น การควบคุมบุคคลเข้า-ออกจากห้องที่เป็นที่อยู่เครื่องคอมพิวเตอร์ที่ให้บริการฐานข้อมูล การใช้ประตูที่แน่นหนา การใช้ระบบแสกนลายนิ้วมือในการเข้าห้องของระบบ ตรวจสอบอุปกรณ์การเชื่อมต่อเพื่อป้องกันดักฟังสัญญาณ การป้องกันระบบคอมพิวเตอร์จากน้ำท่วม ไฟไฟม้ ความร้อน เป็นต้น

นอกจากนี้ยังมีมาตรการอื่นๆ อีก เช่นการใช้โปรแกรมป้องกันภัยคุกคามโดยตรง การใช้อุปกรณ์สำรองไฟฟ้าฯลฯ ซึ่งผู้ดูแลควรตรวจสอบมาตรการรักษาความมั่นคงปลอดภัยของข้อมูลให้ได้รับการปฏิบัติอยู่เสมอ

9.5 แบบฝึกหัดท้ายบท

ให้นักศึกษาใช้วัตถุประสงค์ของบทเป็นแบบฝึกหัดท้ายบท

บทที่ 10 ฐานข้อมูลบนเว็บ

วัตถุประสงค์

- สามารถอธิบายเกี่ยวกับแนวคิดของอินเตอร์เน็ตและเวิลด์ไวด์เว็บได้
- สามารถอธิบายวิวัฒนาการของเว็บได้
- สามารถอธิบายแนวคิดเกี่ยวกับเอกสาร HTML
- สามารถสร้างหน้าเว็บด้วย HTML ได้
- สามารถอธิบายเกี่ยวกับสถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บได้
- สามารถประยุกต์ใช้ PHP ในการพัฒนาโปรแกรมประยุกต์บนเว็บได้
- สามารถอธิบายเกี่ยวกับสถาปัตยกรรมของฐานข้อมูลบนเว็บได้
- สามารถประยุกต์ใช้ PHP และ MySQL ในการพัฒนาฐานข้อมูลบนเว็บได้

คำสำคัญ: อินเตอร์เน็ต (Internet); เวิลด์ไวด์เว็บ (World-Wide Web—WWW); TCP/IP (Transmission Control Protocol/Internet Protocol); HTTP (HyperText Transfer Protocol); HTML (HyperText Mark-up Language); HTML element; HTML tag; HTTP REQUEST; HTTP RESPONSE; CSS (Cascading Style Sheet); JavaScripts; two-tier software architecture; three-tier software architecture; CGI (Common Gateway Interface); โปรแกรมประยุกต์บนเว็บ (web application); PHP (PHP HyperText Preprocessor); ฐานข้อมูลบนเว็บ (Web database)

10.1 บทนำ

เวิลด์ไวด์เว็บ (World-Wide Web) หรือเว็บเป็นเครือข่ายของระบบสารสนเทศที่ได้รับความนิยมและทรงอิทธิพลมากที่สุดในปัจจุบัน เว็บเป็นบริการอันหนึ่งบนเครือข่ายอินเตอร์เน็ต **จำนวนผู้ใช้งานอินเตอร์เน็ตนั้นเพิ่มขึ้นอย่างรวดเร็วจากเพียง 16 ล้านคนในปี ค.ศ. 1995 มาเป็น 1.5 พันล้านคนในปี ค.ศ. 2008 คิดเป็น 23.3% ของจำนวนประชากรโลก 6 พันล้านคน เว็บเป็นเครื่องมือที่ใช้ในการกระจายข้อมูลและสารสนเทศประเภทและรูปแบบต่างๆ ไม่ว่าจะเป็นข้อความ ภาพ เสียง และภาพเคลื่อนไหว

แพลตฟอร์มเว็บนี้สามารถนำมาประยุกต์รวมกับระบบจัดการฐานข้อมูล เพื่อประโยชน์ในการใช้งานข้อมูลจากฐานข้อมูลที่ทรงพลัง ไม่ว่าจะเป็นการกระจายข้อมูลจากฐานข้อมูลไปบนเครือข่ายอินเตอร์ที่มีความแพร่หลายมากที่สุดนี้ หรือการใช้งานเว็บเพื่อประโยชน์ภายในหน่วยงานใดๆ แพลตฟอร์มเว็บสามารถนำมาพัฒนาระบบสารสนเทศร่วมกับระบบจัดการฐานข้อมูลได้อย่างรวดเร็วและมีประสิทธิภาพ หลายๆ หน่วยงานได้นำการใช้ระบบสารสนเทศบนเว็บร่วมกับระบบจัดการฐานข้อมูลนี้มาใช้อย่างแพร่หลาย

ในบทนี้จะได้กล่าวถึงการใช้งานฐานข้อมูลบนเว็บ ซึ่งเป็นการอธิบายและยกตัวอย่างการพัฒนาระบบสารสนเทศบนเว็บที่อาศัยข้อมูลจากระบบจัดการฐานข้อมูล โดยอธิบายถึงแนวคิดพื้นฐานที่เกี่ยวข้องตลอดจนแสดงตัวอย่างการประยุกต์ระบบที่สามารถในไปใช้งานได้จริง เริ่มจากการแนะนำอินเตอร์เน็ตและเวิลด์ไวด์เว็บในหัวข้อถัดไป หลังจากนั้นอธิบายถึงสถาปัตยกรรมซอต์แวร์ที่เป็นพื้นฐานของการพัฒนาโปรแกรมประยุกต์บนเว็บในหัวข้อ 10.3 หัวข้อ 10.4 กล่าวถึง HyperText Markup Language เป็นหลัก ซึ่งเป็นองค์ประกอบสำคัญของเว็บ และหัวข้อ 10.5 อธิบายถึงหัวใจของการสร้างโปรแกรมประยุกต์บนเว็บและการใช้งานฐานข้อมูลบนเว็บด้วย PHP และ MySQL

ในบทนี้จะมีการยกตัวอย่างการออกแบบและพัฒนาฐานข้อมูลจากกรณีตัวอย่าง ร้านค้าออนไลน์ 7-Elephant โดยจะกล่าวถึงในประเด็นสำคัญของการออกแบบและพัฒนาฐานข้อมูล ตลอดจนการพัฒนาโปรแกรมประยุกต์บนเว็บจากกรณีศึกษาการพัฒนาระบบร้านค้าออนไลน์ขายโทรศัพท์มือถือในหัวข้อที่ 10.6

10.2 อินเตอร์เน็ตและเวิลด์ไวด์เว็บ

10.2.1 วิวัฒนาการของอินเตอร์เน็ตและเวิลด์ไวด์เว็บ

เครือข่ายอินเตอร์เน็ตที่ใช้งานกันอย่างแพร่หลายในปัจจุบันสามารถรองรับบริการต่างๆ บนอินเตอร์เน็ตได้มากมาย เช่นการรับส่งจดหมายอิเล็คทรอนิกส์ (Electronic-mail—e-mail) บริการโอนย้ายข้อมูล (File Transfer Protocol—FTP) กลุ่มข่าว (Newsgroup) เป็นต้น นั้นถือกำเนิดขึ้นในช่วงปลาย ค.ศ. 1960s จากโครงการทดลองของกระทรวงกลาโหมของประเทศสหรัฐอเมริกาที่มีชื่อว่า ARPANET (Advanced Research Projects Agency NETwork) โดยมีแนวคิดในการสร้างเครือข่ายคอมพิวเตอร์ที่ยังสามารถเชื่อมต่อกันได้ในกรณีที่จุดเชื่อมโยงใดๆ ขัดข้อง โดยเฉพาะการล้มเหลวของเครือข่ายอย่างร้ายแรงที่อาจเกิดจากการโจมตี เช่นระเบิดนิวเคลียร์ แต่เครื่องคอมพิวเตอร์อื่นๆ ยังสามารถเชื่อมโยงกันได้

ในปี ค.ศ. 1982 TCP/IP (Transmission Control Protocol and Internet Protocol) ได้รับการกำหนดให้เป็นมาตรฐานในการสื่อสารข้อมูลบนเครือข่าย ARPANET ซึ่งหน้าที่สำคัญของ TCP/IP นั้นได้แก่การค้นหาสร้างทางการเชื่อมโยงเครื่องคอมพิวเตอร์ตลอดจนควบคุมการส่งข้อมูลของเครื่องคอมพิวเตอร์ในเครือข่ายให้เป็นไปอย่างถูกต้องตรบถ้วน และมีประสิทธิภาพ หลังจากนั้นได้มีการสร้างบริการต่างๆ เพื่อใช้งานการส่งผ่านข้อมูลด้วยมาตรฐาน TCP/IP เช่นการโอนย้ายข้อมูลด้วย FTP (File Transfer Protocol) การส่งอีเมล์ด้วย SMTP (Simple Mail Transfer Protocol) การใช้งานเครื่องระยะไกลด้วย Telnet (Telecommunication Network) การกำหนดและค้นหาเครื่องคอมพิวเตอร์ด้วยชื่อโดยใช้ DNS (Domain Name Service) และรับอีเมล์ด้วย POP (Post Office Protocol) เป็นต้น

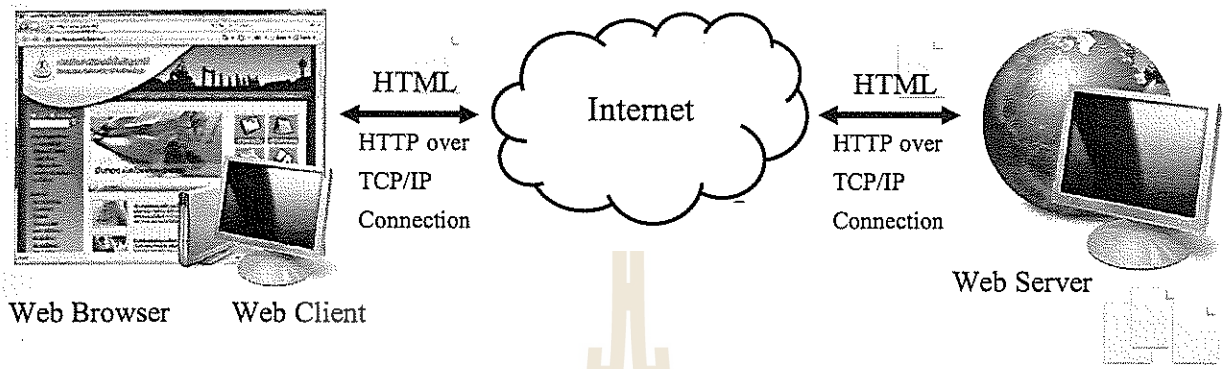
หลังจากนั้นได้มีการนำ TCP/IP ไปประยุกต์ใช้ในการเชื่อมโยงเครื่องข่ายคอมพิวเตอร์ของบริษัทขนาดใหญ่และมหาวิทยาลัยต่างๆ เพื่องานวิจัยในชื่อ NSFNET (National Science Foundation NETwork) ซึ่งต่อมาได้กลายเป็นแกนหลักของเครือข่ายอินเตอร์เน็ต (Internet backbone) ในปี ค.ศ. 1995 ในที่สุด

สำหรับเว็บนั้นเป็นบริการบนเครือข่ายอินเตอร์เน็ตรูปแบบหนึ่ง ถือกำเนิดขึ้นในปี ค.ศ. 1989 โดยสถาบันวิจัย CERN (European Organization for Nuclear Research) ซึ่งเป็นองค์กรที่มุ่งศึกษาฟิสิกส์พื้นฐาน เว็บเป็นเครื่องมือที่พัฒนาขึ้นเพื่อแลกเปลี่ยนข้อมูลระหว่างนักวิจัย การแสดงหน้าเว็บอาศัยโปรแกรมที่ชื่อว่าเว็บบราวเซอร์ (web browser) โดยเว็บบราวเซอร์ที่ได้รับความนิยมมากที่สุดในช่วงแรกได้แก่ Mosaic ที่พัฒนาขึ้นโดยสถาบัน NCSA (National Center for Supercomputing Application) แห่ง University of Illinois at Urbana-Champaign ซึ่งเว็บบราวเซอร์ดังกล่าวได้ถูกพัฒนาให้ใช้ในเชิงพาณิชย์ในชื่อ Netscape ที่รู้จักกันดีในยุคนั้น ซึ่งเว็บบราวเซอร์นี่เองที่ได้รับการยอมรับว่าเป็นเครื่องมืออันหนึ่งที่ทำให้การใช้งานอินเตอร์เน็ตแพร่หลายอย่างรวดเร็ว เว็บบราวเซอร์ได้ถูกพัฒนาให้มีความสามารถเพิ่มมากขึ้นเรื่อยๆ ซึ่งสามารถแสดงข้อความรูปแบบต่างๆ ภาพ เสียง และสื่อประสมต่างๆ บนเว็บได้ ผู้ใช้สามารถเชื่อมโยงไปยังหน้าเว็บอื่นๆ ได้ด้วย hyperlink นอกจากนี้เว็บบราวเซอร์ยังรองรับการทำงานของโปรแกรมที่ซับซ้อนอื่นๆ ซึ่งเรามักเรียกโปรแกรมหรือระบบสารสนเทศที่โต้ตอบกับผู้ใช้ผ่านทางเว็บว่าโปรแกรมประยุกต์บนเว็บ (Web application) ในหัวข้อถัดไปเป็นการอธิบายหลักการทำงานของสถาปัตยกรรมของเว็บ ซึ่งการเข้าใจในสถาปัตยกรรมของเว็บนี้เป็นพื้นฐานของการพัฒนาฐานข้อมูลบนเว็บ

10.2.2 สถาปัตยกรรมฐานข้อมูลบนเว็บ

การบริการเว็บบนเครือข่ายอินเตอร์เน็ตนั้น เป็นการบริการที่ใช้มาตรฐานของการสื่อสารที่เรียกว่า HTTP (HyperText Transfer Protocol) เพื่อควบคุมการสื่อสารระหว่างเครื่องให้บริการและเครื่องรับบริการ ซึ่ง HTTP นี้อาศัย TCP/IP ในการจัดการการสื่อสารข้อมูลในเครือข่ายคอมพิวเตอร์อีกทีหนึ่ง เช่นเดียวกับบริการบน TCP/IP อื่นๆ เช่น TCP, Telnet และ SMTP เป็นต้น การบริการเว็บนี้ต้องอาศัยเครื่องที่ให้บริการเว็บหรือเว็บเซิร์ฟเวอร์ (Web server) ซึ่งจัดเก็บข้อมูลที่จะส่งผ่านไปบนเครือข่าย และเครื่องผู้รับบริการเว็บคือเครื่องคอมพิวเตอร์โดยทั่วไปที่เชื่อมต่อเข้ากับเครือข่าย การเรียกดูและแสดงหน้าเว็บนั้นผู้ใช้ต้องอาศัยเว็บบราวเซอร์ (Web browser) เพื่อแสดงข้อมูลที่ส่งมาจากเว็บเซิร์ฟเวอร์ ข้อมูลและสารสนเทศที่สื่อสารผ่าน HTTP ในบริการเว็บนี้เป็นข้อมูลที่มีลักษณะเฉพาะ ซึ่งอาศัยภาษา

HTML (HyperText Markup Language) ในการจัดการรูปแบบ สถาปัตกรรมของเว็บสามารถแสดงได้ดังแผนภาพต่อไปนี้



จากภาพเอกสารหน้าเว็บและไฟล์ต่างที่ต้องการเผยแพร่ถูกบรรจุอยู่บนเว็บเซิร์ฟเวอร์ (ด้านขวามือ) ผู้ใช้ซึ่งต้องการจะเรียกดูข้อมูลหน้าเว็บใดๆ จะใช้โปรแกรมเว็บบราวเซอร์หรือเว็บไคลเอ็นท์ (Web client—หมายถึงเครื่องหรือโปรแกรมที่ใช้งานเว็บ) เพื่อส่งการร้องขอเอกสารที่ต้องการไปยังเว็บเซิร์ฟเวอร์ ในทางเทคนิคเว็บเซิร์ฟเวอร์จะรอรับการร้องขอการใช้บริการเว็บจากเครื่องคอมพิวเตอร์ใดๆ โดยคอยตรวจรับการร้องขอด้วยพอร์ต (port) 80 ซึ่งเป็พอร์ตโดยทั่วไปที่ให้บริการเว็บ พอร์ตนี้มีลักษณะการทำงานคล้ายๆ กับประตูหมายเลขต่างๆ ที่จะเป็นช่องทางในการสื่อสารระหว่างเครื่องคอมพิวเตอร์ ซึ่งเป็นหลักการทำงานของ TCP/IP เว็บเซิร์ฟเวอร์จะส่งข้อมูลเอกสาร HTML ประกอบไฟล์ต่างๆ ด้วย HTTP บนการจัดการการสื่อสารข้อมูล TCP/IP ผ่านทางเครือข่ายอินเตอร์เน็ต เอกสาร HTML จะถูกประมวลผลและแสดงตามรูปแบบที่กำหนด สถาปัตยกรรมเว็บนี้เป็นพื้นฐานที่จะใช้อ้างอิงถึงต่อไปในการใช้งานฐานข้อมูลบนเว็บ

เราอาศัยสถาปัตยกรรมเว็บนี้เป็นพื้นฐานของการใช้งานระบบฐานข้อมูลบนเว็บ นอกจากแนวคิดเกี่ยวกับสถาปัตยกรรมของเว็บแล้ว การพัฒนาโปรแกรมประยุกต์เพื่อเรียกใช้งานฐานข้อมูลยังอาศัยแนวคิดที่สำคัญอื่นๆ อีก ได้แก่ สถาปัตยกรรมซอฟต์แวร์แบบต่างๆ และรูปแบบของเอกสาร HTML ซึ่งจะได้กล่าวถึงในหัวข้อต่อไปนี้

10.3 สถาปัตยกรรมฐานข้อมูลบนเว็บ

การใช้งานฐานข้อมูลบนเว็บอาศัยการใช้งานผ่านโปรแกรมประยุกต์บนเว็บ (Web application) เป็นหลัก โดยระบบจัดการฐานข้อมูลและตัวฐานข้อมูลเป็นองค์ประกอบหนึ่งของโปรแกรมประยุกต์บนเว็บ หัวข้อต่อไปนี้อธิบายถึงสถาปัตยกรรมซอฟต์แวร์แบบทวิภาค ไตรภาคี และโปรแกรมประยุกต์บนเว็บ ซึ่งเป็นแนวคิดสำคัญในการพัฒนาฐานข้อมูลบนเว็บ

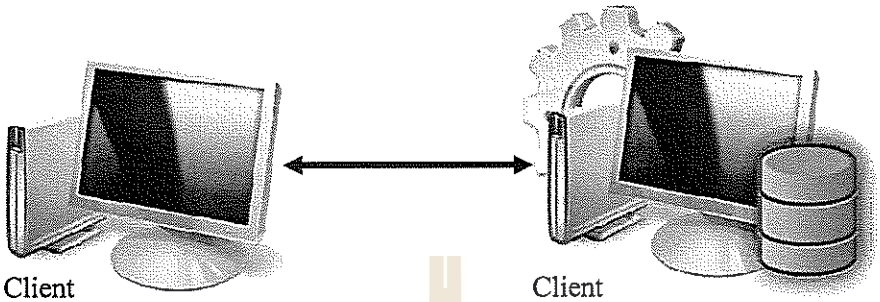
10.3.1 สถาปัตยกรรมซอฟต์แวร์แบบทวิภาคี (Two-Tier Software Architecture)

สถาปัตยกรรมซอฟต์แวร์แบบทวิภาคี (two-tier software architecture) และไตรภาคี (three-tier software architecture) เป็นรูปแบบการทำงานของซอฟต์แวร์จำแนกตามองค์ประกอบของระบบซอฟต์แวร์ ก่อนการใช้งานซอฟต์แวร์ในลักษระ two-tier นั้น ซอฟต์แวร์จะทำงานเบ็ดเสร็จอยู่บนเครื่องคอมพิวเตอร์เพียงเครื่องเดียว เช่น โปรแกรมประมวลผลคำ (word processor) โปรแกรมกระดานคำนวณ (spread sheet) หรือระบบจุดชำระสินค้าที่จัดเก็บข้อมูลอยู่บนเครื่องคอมพิวเตอร์เพียง 1 เครื่อง ต่อมาเครื่องคอมพิวเตอร์สามารถเชื่อมต่อกันได้ด้วยระบบเครือข่ายคอมพิวเตอร์ จึงมีการพัฒนาซอฟต์แวร์ที่สามารถเชื่อมยังกันได้โดยเฉพาะระบบสารสนเทศที่ใช้ข้อมูลร่วมกัน เช่น ระบบจุดชำระสินค้า เราสามารถแยกระหว่างหน้าจอการชำระสินค้าและฐานข้อมูลไว้บนเครื่องคอมพิวเตอร์ต่างเครื่องกันโดยเครื่องกัน โดยจุดชำระสินค้าจะแสดงหน้าจอรายการสินค้าที่ซื้อ โดยจะสอบถามราคาจากเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งที่บรรจุข้อมูลราคาสินค้า การจัดเก็บรายการขายและการลดจำนวนสินค้าคงคลังจะถูกกระทำบนเครื่องที่บรรจุฐานข้อมูล จุดชำระสินค้ามีได้หลายจุด แต่จะเชื่อมต่อเข้ากับฐานข้อมูลเดียวกัน ลักษณะการทำงานของซอฟต์แวร์ที่ส่วนติดต่อกับผู้ใช้แยกออกจากส่วนจัดเก็บข้อมูล และทำงานในลักษณะกระจายตัวดังกล่าวเป็นสถาปัตยกรรมซอฟต์แวร์แบบ two-tier การทำงานของเว็บโดยทั่วไปก็เป็นลักษณะการทำงานด้วยสถาปัตยกรรม two-tier

สถาปัตยกรรมแบบ two-tier นั้นประกอบด้วยองค์ประกอบหลัก 2 ส่วนได้แก่เครื่องผู้ให้บริการหรือเซิร์ฟเวอร์ (server) และเครื่องผู้ใช้บริการ/ไคลเอนท์ (client) โดยไคลเอนท์มักทำหน้าที่หลักๆ 2 ประการ ได้แก่ 1) โต้ตอบกับผู้ใช้โดยการแสดงส่วนติดต่อกับผู้ใช้เพื่อรับค่า และแสดงผลที่ได้จากการประมวลผล 2) ติดต่อกับเซิร์ฟเวอร์เพื่อรับและส่งค่าจากผู้ใช้เพื่อการประมวลผลและบรรจุข้อมูลลงในฐานข้อมูลของเซิร์ฟเวอร์ สำหรับเซิร์ฟเวอร์นั้นมีหน้าที่หลักๆ 2 ประการเช่นกัน ได้แก่ 1) ประมวลผลตามคำสั่งของผู้ใช้ผ่านทางเครื่องไคลเอนท์ 2) เรียกใช้ เพิ่ม แก้ไข และลดข้อมูลในฐานข้อมูล (ในกรณีที่ระบบใช้ฐานข้อมูลเพื่อจัดเก็บข้อมูล) ตามการประมวลผล ระบบสารสนเทศส่วนใหญ่จะใช้ไคลเอนท์เพื่อการโต้ตอบกับผู้ใช้และตรวจสอบความถูกต้องของรูปแบบข้อมูลที่ป้อน ส่วนการประมวลผลตามข้อกำหนดของระบบนั้นมักจะเกิดขึ้นบนฝั่งเซิร์ฟเวอร์ เนื่องจากการประมาวลมักมีการเชื่อมต่อกับฐานข้อมูลและเครื่องเซิร์ฟเวอร์ก็มักจะมีประสิทธิภาพในการทำงานสูง อย่างไรก็ตาม การใช้งานในบางลักษณะเครื่องไคลเอนท์อาจรับหน้าที่ในการประมวลเสียเอง เครื่องแม่ข่ายถูกใช้งานในการจัดเก็บข้อมูลเท่านั้น

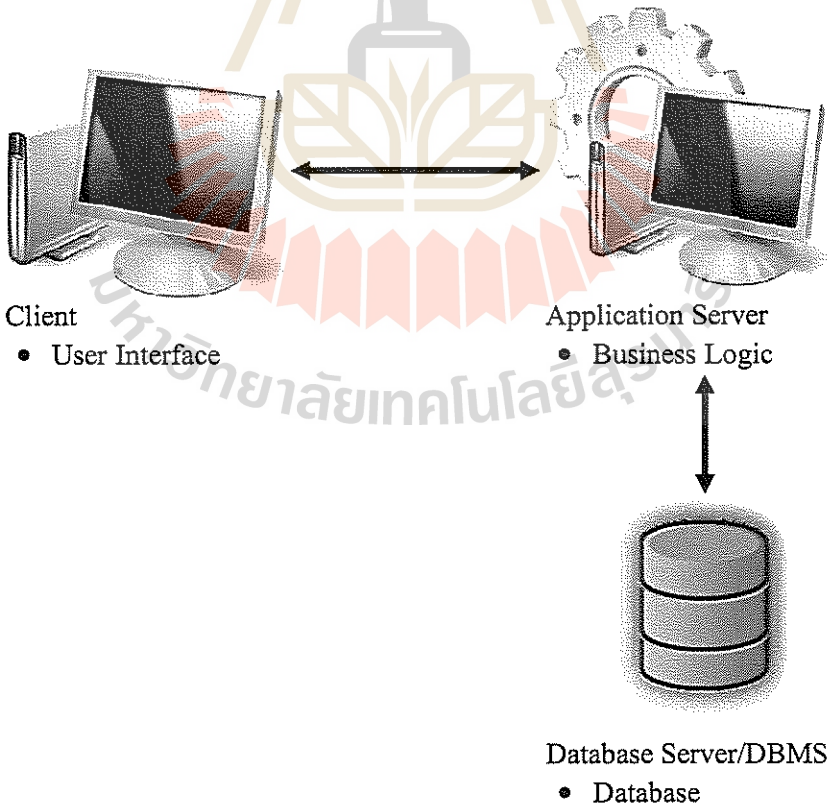
ขั้นตอนการประมวลผลไม่ว่าจะบนเครื่องไคลเอนท์หรือเซิร์ฟเวอร์เพื่อรองรับการใช้งานของผู้ใช้สามารถเรียกด้วยคำเฉพาะว่าตรรกะทางธุรกิจ (business logic) เช่น ระบบโอนเงินของธนาคารจากบัญชีหนึ่งไปยังอีกบัญชีหนึ่งนั้นจะต้องตรวจสอบก่อนว่า 1) หมายเลขบัญชีที่ถอนและหมายเลขบัญชีที่โอนมีอยู่จริง 2) จำนวนเงินที่โอนจะต้องไม่เกินจำนวนเงินที่อยู่ในบัญชี 3) หักยอดเงินออกจากบัญชีต้นทาง 4) เพิ่มยอดเงินในบัญชีปลายทาง เป็นต้น business logic นี้สามารถประยุกต์ไว้ได้ทั้งบนฝั่งไคลเอนท์และเซิร์ฟเวอร์ ในบางกรณีเราสามารถฝัง business logic ไว้ในระบบจัดการฐานข้อมูลก็ได้ ทั้งนี้การประยุกต์ business logic ในองค์ประกอบใดนั้นแล้วแต่ความเหมาะสม

สถาปัตยกรรมซอฟต์แวร์แบบ two-tier สามารถแสดงได้ดังภาพต่อไปนี้



10.3.2 สถาปัตยกรรมซอฟต์แวร์แบบไตรภาคี (Three-Tier Software Architecture)

โปรแกรมประยุกต์บนเว็บอาศัยสถาปัตยกรรมซอฟต์แวร์แบบ three-tier ซึ่งต่างจากสถาปัตยกรรมซอต์ฟแวร์แบบ two-tier ตรงที่ business logic จะแยกตัวออกจาก tier อื่นๆ ซึ่งแสดงได้ด้วยแผนภาพดังนี้



การพัฒนาซอฟต์แวร์โดยอ้างอิงสถาปัตยกรรมแบบ three-tier นี้ ผู้พัฒนาโดยส่วนใหญ่จะพยายามประยุกต์ business logic ให้แยกตัวออกมาจาก tier อื่นๆ โดยเฉพาะออกมาจากตัวฐานข้อมูล (stored procedure ต่างๆ) เพื่อประสิทธิภาพในการใช้งานให้มีมากที่สุด กล่าวคือเราสามารถแยกเครื่องคอมพิวเตอร์ที่รองรับการจัดการฐานข้อมูลออกจากเครื่องคอมพิวเตอร์ที่บริการโปรแกรมประยุกต์ (application server) ได้ การกระทำดังกล่าวจะเป็นการแบ่งเบาภาระการทำงานของแต่ละ tier นอกจากประสิทธิภาพที่เพิ่มขึ้นแล้ว ประโยชน์ที่เห็นเด่นชัดจากสถาปัตยกรรมซอฟต์แวร์แบบ three-tier นี้ได้แก่ความเป็นอิสระระหว่าง tier ต่างๆ ของระบบบ เช่น ภาษาหลักที่ใช้ในการพัฒนาโปรแกรมประยุกต์นั้นไม่จำเป็นต้องอ้างอิงระบบจัดการฐานข้อมูลของผู้ผลิตรายใดรายหนึ่งโดยเฉพาะ ในกรณีที่มีความจำเป็นต้องเปลี่ยนระบบจัดการฐานข้อมูล business logic ยังคงอยู่ดังเดิมใน application server และสามารถใช้งานใหม่ได้ทันทีที่การปรับเปลี่ยนระบบแล้วเสร็จ

10.3.3 สถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บ (Web Applicaiton Architecture)

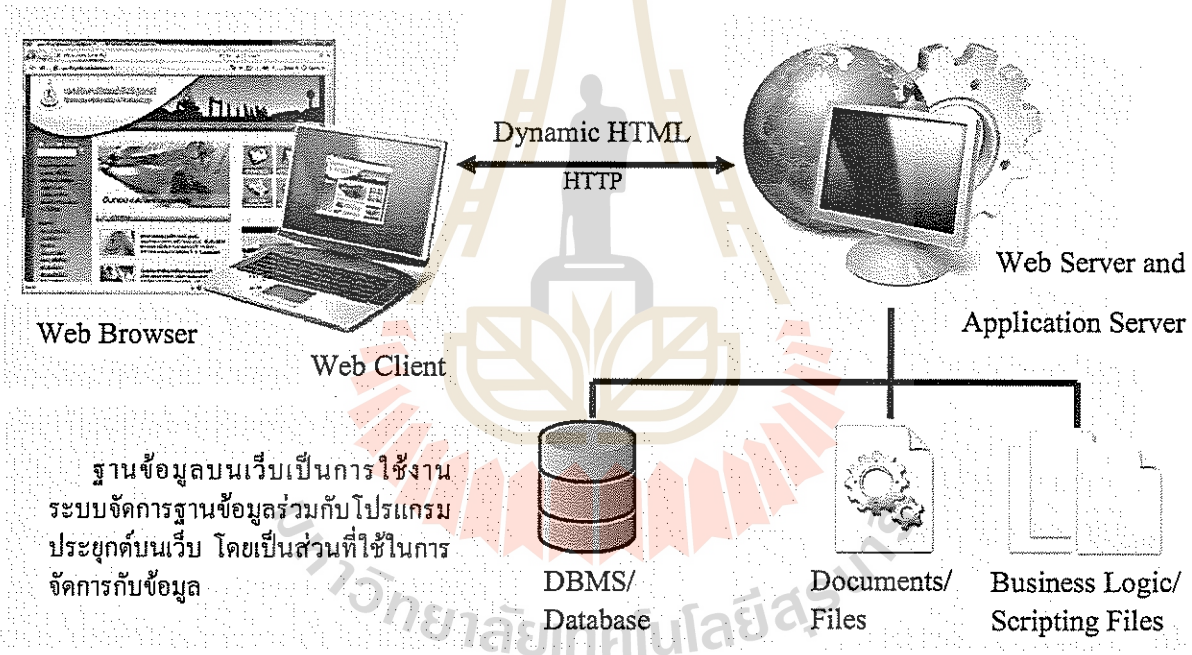
นอกจากเว็บจะทำให้การใช้งานเครือข่ายอินเตอร์เน็ตได้รับความนิยมอย่างแพร่หลาย สถาปัตยกรรมของเว็บเองยังช่วยอำนวยความสะดวกในการสร้างระบบสารสนเทศที่ใช้งานผ่านเครือข่ายได้อย่างรวดเร็วและมีมาตรฐานมากขึ้น หากย้อนไปพิจารณาถึงสถาปัตยกรรมซอฟต์แวร์แบบ tree-tier ในส่วนของการติดต่อกับผู้ใช้ ผู้พัฒนาระบบมีความจำเป็นต้องพัฒนาซอฟต์แวร์ที่อยู่บนเครื่องไคลเอนท์สำหรับติดต่อกับผู้ใช้และสื่อสารกับเซิร์ฟเวอร์ด้วยตนเอง ซึ่งการพัฒนาโปรแกรมในลักษณะดังกล่าวมีความยากลำบากและมีข้อเสียหลายประการ เช่น

1) การพัฒนาโปรแกรมในลักษณะการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง ได้แก่เครื่องผู้รับบริการและเครื่องผู้ให้บริการ (client-server) นั้นเป็นงานที่ยาก เนื่องจากระบบจะมีความซับซ้อนกว่าการพัฒนาโปรแกรมที่ทำงานบนเครื่องๆ เดียว(โดยทั่วไป) โดยเฉพาะการจัดการการเชื่อมต่อ และการโต้ตอบทางเทคนิคระหว่างเครื่อง
2) โปรแกรมที่พัฒนาสำหรับระบบปฏิบัติการใด จะสามารถใช้ได้เฉพาะบนระบบปฏิบัติการนั้นเท่านั้น เช่น โปรแกรมไคลเอนท์ที่พัฒนาสำหรับระบบปฏิบัติการ Windows จะไม่สามารถนำไปใช้บนระบบปฏิบัติการ Apple หรือระบบปฏิบัติการในตระกูล UNIX ได้
3) สภาพแวดล้อมของเครื่องไคลเอนท์แต่ละเครื่องไม่เหมือนกัน อาจก่อให้เกิดปัญหาโปรแกรมไม่สามารถใช้งานได้ และเสียเวลาในการแก้ไข
4) การติดตั้งโปรแกรมเวอร์ชันใหม่ที่มีความสามารถมากขึ้น ทั้งนี้อาจเพื่อรองรับข้อกำหนดของระบบใหม่สามารถสร้างปัญหาในการจัดการระบบได้หลายกรณี ไม่ว่างเป็นจำนวนเครื่องไคลเอนท์ที่มาก การไม่ติดตั้งเวอร์ชันใหม่จนเกิดปัญหากับระบบ หรือการออกเวอร์ชันใหม่บ่อยเกินไป เป็นต้น

ข้อเสียทั้ง 4 ข้อนี้เป็นเพียงตัวอย่างเพียงส่วนเดียวเท่านั้นของอุปสรรคในการพัฒนาและใช้งานระบบสารสนเทศในลักษณะ three-tier ซึ่งการนำสถาปัตยกรรมของเว็บมาบูรณาการในการสร้างโปรแกรมประยุกต์สามารถกำจัดปัญหาดังกล่าวให้หมดไปได้ โดยอาศัยแนวคิดในการสร้างส่วนติดต่อกับผู้ใช้ผ่านทางเว็บ กล่าวคือ ซอฟต์แวร์

หรือระบบสารสนเทศที่ได้พัฒนาขึ้น จะโต้ตอบกับผู้ใช้ผ่านทางเว็บ เมื่อผู้ใช้ป้อนข้อมูลใดๆ บนหน้าเว็บแล้วสั่งให้โปรแกรมประมวลผล เว็บบราวเซอร์จะส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์ จากนั้นเว็บเซิร์ฟเวอร์สามารถประสานกับ application server เพื่อประมวลผล หลังจากหรือในขณะที่ระบบประมวลผลนั้น ผลลัพธ์จะถูกสร้างขึ้นในรูปแบบของเอกสาร HTML แล้วส่งกลับไปยังผู้ใช้ผ่านทางเว็บบราวเซอร์เช่นเดิม ซึ่งการโต้ตอบกับผู้ใช้ผ่านทางเว็บนี้ทำให้ผู้ใช้สามารถใช้งานบนระบบปฏิบัติการใดก็ได้ เนื่องจาก HTML ไม่ยึดติดกับระบบปฏิบัติการและรูปแบบของอุปกรณ์แสดงผลแต่ถูกจัดการด้วยเว็บบราวเซอร์ การให้บริการเว็บก็เป็นมาตรฐานที่สามารถประยุกต์ได้ทันที ลดความซับซ้อนในการพัฒนาระบบลงเป็นอย่างมาก อีกทั้งเว็บบราวเซอร์เองยังมีความสามารถที่หลากหลายอยู่ในตัว สามารถโต้ตอบกับผู้ใช้ได้อย่างมีประสิทธิภาพโดยไม่ต้องการเครื่องคอมพิวเตอร์ที่มีความสามารถในการประมวลผลสูงนัก และระบบที่พัฒนาขึ้นสามารถใช้งานจากที่ใดๆ ก็ได้ที่เชื่อมต่อกับเครือข่ายอินเตอร์เน็ต หรือใช้ภายในองค์กรได้อย่างสะดวกด้วย TCP/IP บนเครือข่ายท้องถิ่น

สถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บสามารถแสดงได้ด้วยแผนภาพดังนี้



จากรูป สถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บอ้างอิงพื้นฐานของสถาปัตยกรรมซอฟต์แวร์แบบ three-tier ซึ่งประกอบด้วยส่วนโต้ตอบกับผู้ใช้ด้วยเว็บ ซึ่งเราสามารถเรียกส่วนนี้ได้ว่าชั้นของการนำเสนอ (presentation layer) ส่วนหรือชั้นของโปรแกรมประยุกต์ (application layer) และส่วนของฐานข้อมูล ในแผนภาพ ส่วนของการโต้ตอบกับผู้ใช้ได้แก่เว็บบราวเซอร์ ส่วนของโปรแกรมประยุกต์มักจะผสานเข้ากับเว็บเซิร์ฟเวอร์ซึ่งเป็นที่จัดเก็บเอกสาร HTML และไฟล์ต่างๆ ตลอดจนโปรแกรมประยุกต์ที่เขียนขึ้นโดยภาษาสคริปต์ (scripting language) ต่างๆ

สำหรับในส่วนของฐานข้อมูลนั้นสามารถแยกออกมาจัดการในเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง หรือจะอยู่รวมกับ application server ก็ได

เราสามารถยกตัวอย่างการใช้งาน Google ซึ่งเป็นโปรแกรมประยุกต์บนเว็บที่ใช้งานกันมากที่สุดโปรแกรมหนึ่งสำหรับการค้นหาเว็บไซต์ตามคำค้นเพื่ออธิบายการทำงานของสถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บดังนี้ ส่วนของการโต้ตอบกับผู้ใช้นั้นเราใช้เว็บบราวเซอร์ในการโต้ตอบ โดยเริ่มจากการที่ผู้ใช้ร้องขอหน้าเว็บที่ต้องการโดยระบุ URL (Universal Resource Locator) บนเว็บบราวเซอร์ http://www.google.com จากนั้นเว็บเบราวเซอร์จะทำการส่งการร้องของไปยังเว็บเซิร์ฟเวอร์ด้วย HTTP REQUEST เว็บไซต์ Google จะทำการส่งเว็บหน้าแรกมายังเว็บบราวเซอร์ หน้าเว็บนี้เป็นเอกสาร HTML ที่ส่งมาด้วยวัตถุ HTTP RESPONSE เมื่อเว็บบราวเซอร์ได้รับเอกสาร HTML แล้วจะทำการประมวลผลโดยการจัดหน้าจอ (render) เพื่อแสดงผลบนเว็บบราวเซอร์ด้วยรูปแบบที่กำหนดไว้ในเอกสาร HTML ในช่วงนี้ยังไม่ได้มีการใช้งานฐานข้อมูลหรือการประมวลผลใดๆ เอกสาร HTML ซึ่งเป็นหน้าแรกของ Google นี้ไม่ได้ถูกสร้างขึ้นจากการประมวลผลใดๆ เป็นเอกสาร HTML ที่เป็นไฟล์เก็บไว้บนเซิร์ฟเวอร์อยู่แล้ว เราเรียกเอกสาร HTML ประเภทนี้ว่า HTML ที่มีเนื้อหาคงที่ (static HTML)

จากนั้นผู้ใช้ทำการกรอกคำที่ต้องการค้นหา เช่น “web database” และคลิกที่ปุ่มค้นหา “Search” เว็บบราวเซอร์จะทำการส่งการร้องขอ HTTP REQUEST ไปยัง Google อีกครั้งหนึ่ง ซึ่งคำค้น “web database” นี้จะถูกส่งไปพร้อมกันเพื่อให้ Google ประมวลผล ในขั้นตอนนี้ Google จะอาศัย business logic สำหรับค้นหาเว็บไซต์ด้วยโปรแกรมที่ Google พัฒนาขึ้นใน application server และประสานกับระบบจัดการฐานข้อมูลที่เก็บข้อมูลเว็บไซต์ไว้เป็นจำนวนมาก application server หรือถ้าจะกล่าวให้เฉพาะเจาะจงคือ application engine จะทำการประมวลผลเพื่อให้ได้มาซึ่งผลลัพธ์คือเว็บที่ตรงตามคำค้น จากนั้นผลลัพธ์จะถูกสร้างขึ้นในรูปแบบของ HTML บนเซิร์ฟเวอร์ เพื่อส่งกลับมายังเครื่องผู้ใช้และ render โดยเว็บบราวเซอร์เพื่อแสดงผลในที่สุด ผลลัพธ์ HTML ที่สร้างขึ้นทันทีจากการประมวลผลนี้จะไม่ได้ถูกจัดเก็บไว้บนเครื่องเซิร์ฟเวอร์ เราเรียก HTML ประเภทนี้ว่า dynamic HTML

การอธิบายการทำงานของโปรแกรมประยุกต์บนเว็บด้วย Google ที่ได้ยกตัวอย่างนั้นอาจมีขั้นตอนที่ซับซ้อนกว่านี้ในทางปฏิบัติ แต่โดยภาพรวมนั้นมีแนวคิดดังที่กล่าว และโปรแกรมประยุกต์บนเว็บทั่วๆ ไปก็มีหลักการทำงานเช่นเดียวกันซึ่งจะได้กล่าวถึงในรายละเอียดต่อไป

จากสถาปัตยกรรมของโปรแกรมประยุกต์ที่ได้อธิบายนั้น การพัฒนาโปรแกรมประยุกต์บนเว็บจึงต้องเข้าใจถึงวิธีการสร้างและจัดการเอกสาร HTML และการประยุกต์ business logic ด้วย scripting language บน application server รวมถึงวิธีการทำงานร่วมกับฐานข้อมูล ซึ่งในส่วนของการจัดการกับฐานข้อมูลนั้นเราอาศัย SQL ที่ได้ศึกษาไปนั่นเองในการใช้งานฐานข้อมูล ซึ่ง HTML นั้นได้อธิบายในหัวข้อ 10.4 และการพัฒนาโปรแกรมประยุกต์ด้วย PHP อธิบายในหัวข้อ 10.5 ซึ่งเป็นหัวข้อ 2 หัวข้อที่จะกล่าวถึงต่อไปนี้

10.4 HTML

HTML ย่อมาจาก HyperText Mark-up Language เป็นภาษาๆ หนึ่งที่ใช้ในการกำหนดรูปแบบและคุณสมบัติให้กับองค์ประกอบของหน้าเว็บ เอกสาร HTML เป็นเอกสารตัวหนังสือธรรมดา (plain text) ประกอบไปด้วย

องค์ประกอบต่างๆ (HTML elements) ซึ่งองค์ประกอบแต่ละองค์ประกอบจะถูกกำกับ (mark-up) ไว้ด้วย tag ถึงคุณสมบัติและรูปแบบตามที่กำหนด ซึ่ง HTML tag นี้ รวมกับองค์ประกอบของหน้าเว็บ เช่น ข้อความ ภาพ และสื่อต่างๆ จะถูกประมวลผลเพื่อแสดง (render) ให้อยู่ในรูปแบบตามที่กำหนดในเอกสารHTML ด้วยเว็บบราวเซอร์ องค์กรที่เป็นผู้กำหนดมาตรฐานของ HTML ได้แก่ World Wide Web Consortium (W3C) http://www.w3c.org

เอกสาร HTML รวมกับไฟล์ต่างๆ ที่องค์ประกอบของหน้าเว็บนี้เองที่จะถูกส่งจากเว็บเซิร์ฟเวอร์ไปยังผู้ใช้บริการเว็บ โดยดั้งเดิมเอกสาร HTML ที่ส่งไปยังผู้ใช้ถูกสร้างขึ้นและเก็บไว้บนเว็บเซิร์ฟเวอร์ดังเช่นไฟล์ทั่วๆ ไป แต่สำหรับการใช้บริการโปรแกรมประยุกต์บนเว็บนั้น เอกสาร HTML ที่ผู้ใช้แต่ละคนได้รับจะถูกสร้างขึ้นเฉพาะสำหรับการร้องขอแต่ละการร้องขอ ไม่ว่าจะเป็นการสร้างเอกสาร HTML แบบดั้งเดิมหรือการสร้างขึ้นทันทีทันใดสำหรับโปรแกรมประยุกต์บนเว็บนั้นความรู้เกี่ยวกับ HTML tag เป็นสิ่งจำเป็นสำหรับผู้พัฒนา ในหัวข้อต่อไปจะเป็นการอธิบายถึงการสร้างหน้าเว็บด้วย HTML tag

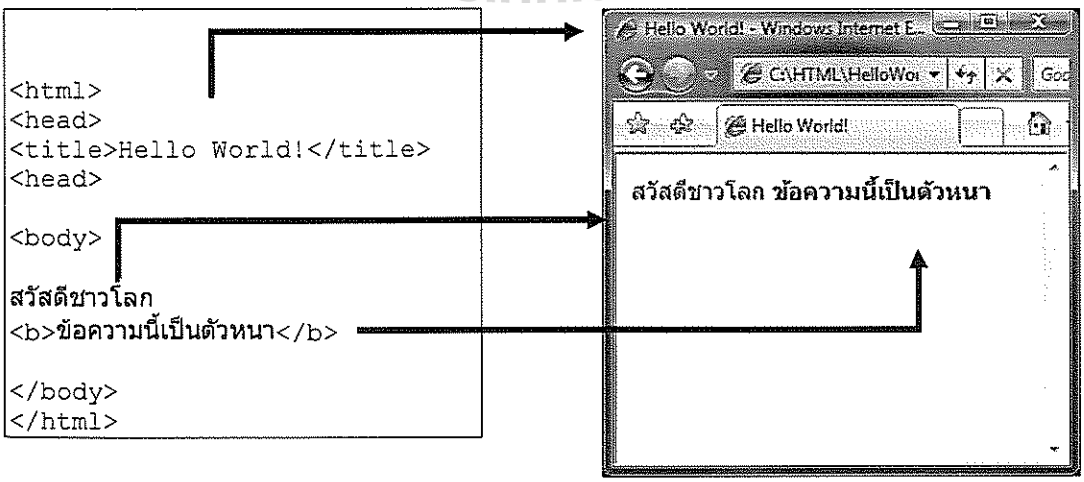
10.4.1 HTML Tags

HTML tag ที่ใช้กำหนดคุณสมบัติและรูปแบบองค์ประกอบต่างๆ ของหน้าเว็บ ประกอบไปด้วยคำหลัก (keyword) ที่บรรจุอยู่ในเครื่องหมายน้อยกว่า "<" และ เครื่องหมายมากกว่า ">" เช่น <html> ซึ่งโดยปกติแล้ว HTML tag จะใช้เป็นคู่ เช่น และ เพื่อกำกับองค์ประกอบที่อยู่ระหว่าง tag ทั้งสอง เราเรียก tag ที่มาก่อนกว่า tag เริ่มต้น (start tag) และ tag ที่อยู่ท้ายองค์ประกอบว่า tag สิ้นสุด (end tag) หรือบางครั้งเรียกว่า opening tag และ closing tag ตามลำดับ

ปัจจุบันมาตรฐานการสร้างเอกสาร HTML นิยมอ้างอิงถึง HTML เวอร์ชัน 4.01 ซึ่งเป็น HTML ที่อ้างอิงรูปแบบของ XML ที่เราเรียกว่า XHTML ซึ่งมีรูปแบบที่รัดกุมขึ้น เช่น HTML tag จะต้องเขียนด้วยตัวเล็ก หรือ tag ที่เป็น tag เดี่ยวจะต้องมีเครื่องหมายทับ "/" เพื่อปิด เช่น
 เป็นต้น ซึ่งในที่นี้จะขอกล่าวถึง HTML โดยภาพรวมไม่ได้อ้างอิงถึง XHTML อย่างเคร่งครัด

ตัวอย่าง ตัวอย่างเอกสาร HTML อย่างง่าย

ตัวอย่างของเอกสาร HTML แสดงทางด้านซ้าย และการแสดงตัวอย่าง HTML ด้วยเว็บบราวเซอร์



จากตัวอย่าง

<html> เป็น tag ที่ระบุว่าเอกสารนี้เป็นเป็นเอกสาร HTML ให้เว็บบราวเซอร์ render แบบ HTML เนื่องจากเว็บบราวเซอร์สามารถแสดงเอกสารได้หลายชนิด การระบุดังกล่าวจะทำให้เว็บบราวเซอร์ทำงานอย่างถูกต้อง

<head> เป็น tag ที่ระบุส่วนหัวของเอกสาร HTML มักใช้ระบุคุณสมบัติต่างๆ ของเอกสารทั้งเอกสารซึ่งข้อมูลในส่วนนี้จะไม่ปรากฏอยู่ในบริเวณที่ใช้แสดงเนื้อหาของเว็บ

<title> เป็น tag ที่ระบุ Title ของเอกสารหน้าเว็บ จะปรากฏอยู่บน Title Bar ดังตัวอย่าง "Hello World!"

<body> ระบุว่าองค์ประกอบของ HTML ภายในส่วน <body> นี้เป็นส่วนของตัวเนื้อหา ให้แสดงในบริเวณส่วนแสดงเนื้อหาของเว็บบราวเซอร์

 เป็น tag จัดรูปแบบข้อความเป็นตัวหนาดังแสดง

สังเกตได้ว่าในเอกสาร HTML ข้อความในเอกสาร "สวัสดีชาวโลก" และ "ข้อความนี้เป็นตัวหนา" แยกออกเป็น 2 บรรทัด แต่ในผลลัพธ์ที่แสดงด้วยเว็บบราวเซอร์นั้นแสดงอยู่ในบรรทัดเดียวกัน เนื่องจากเว็บบราวเซอร์จะไม่คำนึงถึงการขึ้นบรรทัดใหม่ โดยจะพิจารณาว่าองค์ประกอบของ HTML ในเอกสาร HTML นั้นเขียนต่อเนื่องกัน ในกรณีที่เราต้องการให้มีการขึ้นบรรทัดใหม่เมื่อแสดงผลด้วยเว็บบราวเซอร์ เราใช้
 ในการขึ้นบรรทัดใหม่

ต่อไปนี้จะเป็นอธิบายถึงการใช้งาน HTML Tag ที่สำคัญๆ ได้แก่ Tag ที่ใช้ในการจัดรูปแบบตัวอักษร การแทรกองค์ประกอบพื้นฐาน และการสร้างตาราง

<u>การจัดรูปแบบตัวอักษร</u>

ตัวอย่าง การจัดรูปแบบตัวอักษรของ HTML

พิจารณาตัวอย่าง HTML เกี่ยวกับการจัดรูปแบบตัวอักษร ประกอบผลลัพธ์ที่แสดงด้วยเว็บบราวเซอร์ต่อไปนี้

```
<html>
<head>
<title>HTML</title>
<head>

<body>

<h1>หัวเรื่อง</h1>
<h2>หัวเรื่อง</h2>
<h3>หัวเรื่อง</h3>

<p style="font-family:times new roman">
  Times New Roman Fonts
</p>

<p style="font-size:30px">
 ขนาดตัวอักษร 30 พิเซล
</p>

<p style="color:green">
 สีตัวอักษรเขียว
</p>

<p style="font-family:angsana new;
font-size:40px;color:#0000FF">
 รูปแบบ ขนาด และสี
</p>

<p style="text-align:center">
  จัดกึ่งกลาง
</p>

</body>
</html>
```

จากตัวอย่าง สามารถอธิบายการใช้งาน tag ต่างประกอบแผนภาพดังนี้

<h1> คือ tag หัวเรื่อง โดย <h1> คือหัวเรื่องระดับที่ 1 ซึ่งมีถึง <h6>

<p> คือ tag ครอบข้อความให้เป็นหนึ่งย่อหน้า

style เป็นการใช้ style โดยระบุคุณสมบัติของข้อความ ปัจจุบันแนะนำให้ใช้แทน tag <font> โดยระบุแอทริบิวต์ที่ต้องการกำหนดตามด้วยเครื่องหมาย colon ":" ตามด้วยค่าที่กำหนด

"font-family:times new roman" ระบุชนิดของ font เป็น Times New Roman

"font-size:30px" ระบุขนาดของตัวอักษรเป็นขนาด 30 พิกเซล

"color:green" ระบุสีของตัวอักษร เป็นสีเขียว

"color:#0000FF" ระบุสีของตัวอักษรโดยใช้รหัสสี ในที่นี้คือสีน้ำเงิน

"text-align:center" ระบุรูปแบบการจัดวางข้อความกึ่งกลางหน้าเว็บ

แอทริบิวต์ของ style สามารถกำหนดพร้อมๆ กันหลายๆ แอทริบิวต์ในองค์ประกอบ HTML หนึ่งองค์ประกอบได้ดังตัวอย่าง style นี้ใช้ควบคู่กับองค์ประกอบ HTML ใดๆ โดยฝังไว้ภายในเอกสาร HTML หรืออาจแยกออกมาเป็น style sheet ก็ได้ ซึ่งจะได้อธิบายในหัวข้อ CSS

องค์ประกอบพื้นฐาน

ตัวอย่าง การแทรกองค์ประกอบพื้นฐาน

พิจารณาตัวอย่าง HTML แสดงการแทรกองค์ประกอบพื้นฐานของหน้าเว็บ ประกอบผลลัพธ์ที่แสดงด้วยเว็บบราวเซอร์ต่อไปนี้

```
<html><!--Header Part-->
<head><title>HTML</title><head>
<body><!--Body Part-->

<img src="images/phone.jpg" />
<br />
<b>การแทรกรูปภาพ<b><br />
<hr />

การใส่ลิงค์ไปยัง<br />
<a href="http://www.sut.ac.th">
 มหาวิทยาลัยเทคโนโลยีสุรนารี</a>
<br />

การใส่ลิสต์แบบเรียงลำดับ
<ol>
  <li>รายการที่ 1</li>
  <li>รายการที่ 2</li>
</ol>

การใส่ลิสต์แบบไม่เรียงลำดับ
<ul>
  <li>รายการที่ 1</li>
  <li>รายการที่ 2</li>
</ul>

</body>
</html>
```

จากตัวอย่าง สามารถอธิบายการใช้งาน tag ต่างประกอบแผนภาพดังนี้

<!-- ... --> เป็นการหมายเหตุหรือ comment ในโค้ด HTML เท่านั้น ไม่แสดงผลทางหน้าเว็บ

<img> เป็น tag ที่ใช้แทรกรูปภาพ ในที่นี้ หมายถึงที่อยู่ของรูปภาพอยู่ในโฟลเดอร์ images ไฟล์ภาพชื่อว่า phone.jpg สำหรับที่อยู่ของโฟลเดอร์นั้นอ้างอิงกับที่อยู่ปัจจุบันของเอกสาร HTML ในที่นี้ img มาจากคำว่า image ส่วน src มาจากคำว่า source ให้สังเกตเครื่องหมาย "/" ที่ท้าย tag เนื่องจาก tag <img> เป็น tag เดี่ยว จึงควรใส่ "/" ตามข้อกำหนด XHTML

 เป็นการขึ้นบรรทัดใหม่

<a> เป็น tag ที่สำคัญที่สุด tag หนึ่งของ HTML ก็ว่าได้ โดยเฉพาะการใช้ในกรณีตัวอย่างเพื่อการเชื่อมโยงเอกสาร มหาวิทยาลัยเทคโนโลยีสุรนารี นั้นเป็นการครอบข้อความ "มหาวิทยาลัยเทคโนโลยีสุรนารี" ไว้ด้วย tag <a> นั่นหมายถึงการระบุ hyperlink เชื่อมโยงไปยังเอกสารหรือเว็บไซต์ที่กำหนดเมื่อคลิกที่คำว่า "มหาวิทยาลัยเทคโนโลยีสุรนารี" โดยค่าที่กำกับไว้สำหรับแอทริบิวต์ href คือค่าของเอกสารปลายทางที่จะเชื่อมโยง ในที่นี้คือ "http://www.sut.ac.th" ซึ่งเป็นเว็บไซต์ของมหาวิทยาลัยเทคโนโลยีสุรนารี การใส่ hyperlink สามารถอ้างอิงถึงเอกสารภายในเว็บไซต์ตนเองโดยไม่จำเป็นต้องอ้างอิงถึงเว็บไซต์อื่นได้ โดยระบุตำแหน่งของไฟล์ภายในเว็บเซิร์ฟเวอร์ด้วยการอ้างอิงตำแหน่งกับตำแหน่งไฟล์ HTML ปัจจุบัน

 สร้างรายการแบบมีลำดับ (ordered-list) ซึ่งสมาชิกของรายการที่ระบุไว้จะถูกนำหน้าด้วยหมายเลขหรือตัวอักษาที่เรียงลำดับ

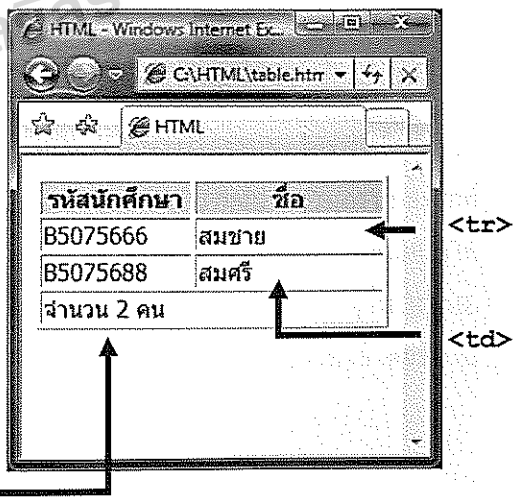
 สร้างรายการแบบไม่มีลำดับ (unordered-list) ซึ่งสมาชิกของรายการที่ระบุไว้จะถูกนำหน้าด้วยเครื่องหมายนำหน้ารายการแต่ละรายการ

 ใช้ระบุสมาชิกของรายการ <list item> ไม่ว่าจะเป็นรายการมีและไม่มีลำดับ

ตาราง

ตัวอย่าง การสร้างตารางด้วย HTML

```
<table border="1" width="100%">
  <tr>
    <th width="100"
    bgcolor="lightblue">
      รหัสนักศึกษา</th>
    <th bgcolor="lightblue">ชื่อ</th>
  </tr>
    <td>B5075666</td>
    <td>สมชาย</td>
  <tr>
    <td>B5075688</td>
    <td>สมศรี</td>
  </tr>
  <tr>
    <td colspan="2">จำนวน 2 คน</td>
```



```
  </tr>
</table>
```

จากตัวอย่าง

<table> คือ tag ที่ใช้สร้างตารางนั่นเอง โดยต่อไปจะเป็นการระบุสมาชิกในตาราง อันประกอบไปด้วยแถวและคอลัมน์ สังเกตตัวอย่างมีการยกตัวอย่างการกำหนดแอทริบิวต์

border="1" หมายถึงเส้นแบ่งตารางมีความหนา 1 พิกเซล

width="100%" ความกว้างของตารางเท่ากับร้อยละ 100 ของหน้าจอเว็บ

<tr> กำหนดแถวของตาราง องค์ประกอบใดๆ ที่อยู่ภายใต้ tag นี้จะเป็นแถวของตาราง

<td> กำหนดสมาชิกในแถว หรือกำหนดคอลัมน์ในแถวนั่นเอง

<th> เป็นการกำหนดสมาชิกในแถวประเภทหนึ่ง แต่เป็นเซลพิเศษคือเป็นหัวตาราง เพื่อการจัดรูปแบบของตารางที่สะดวก ข้อมูลที่ปรากฏในตารางจะต้องอยู่ภายใต้ tag <td> หรือ <th> นี้ ไม่เช่นนั้นข้อมูลจะแสดงอยู่ภายนอกตาราง ในตัวอย่างมีการกำหนดแอทริบิวต์ของ <td> และ <th>

width="100" ความกว้างของคอลัมน์ 100 พิกเซล สำหรับคอลัมน์ที่ไม่ได้กำหนดความกว้าง ความกว้างของคอลัมน์จะใช้ความกว้างที่เหลืออยู่ของความกว้างตาราง หรืออ้างอิงกับความยาวของเนื้อหาในคอลัมน์

bgcolor="lightblue" กำหนดสีพื้นหลังของเซลให้เป็นสีฟ้าอ่อน

colspan="2" หมายถึงการขยายเซลให้ครอบคลุม 2 คอลัมน์ โดยตารางจะเริ่มสร้างเซลจากสมาชิก <td> แต่ละตัว ตำแหน่งของคอลัมน์ในแถวถัดมาจะอ้างอิงถึงคอลัมน์จากแถวบน ในกรณีที่คอลัมน์ในแต่ละแถวมีจำนวนไม่เท่ากันต้องใช้ colspan ในการขยายขนาดของคอลัมน์ โดยขยายได้เป็นคอลัมน์ๆ คล้ายๆ กับการผสานตารางในโปรแกรมทั่วไป นอกจาก colspan แล้ว rowspan ก็เป็นการกำหนดอาณาเขตของเซลในทางแนวตั้งซึ่งมีหลักคล้ายคลึงกัน

ตัวอย่างการใช้งาน HTML tag ที่สำคัญๆ ที่กล่าวมานั้นเป็นเพียงการแสดงตัวอย่างพื้นฐานของการนำไปใช้งานต่อไป สำหรับการนำไปประยุกต์ใช้จริงเพื่อสร้างหน้าเว็บที่เป็นรูปธรรมมากกว่านี้ และการพัฒนาโปรแกรมประยุกต์บนเว็บนั้นได้แสดงด้วยตัวอย่างการสร้างเว็บไซต์ e-commerce ในหัวข้อกรณีศึกษา นอกจาก HTML tag สำคัญๆ ที่ได้กล่าวไปแล้ว HTML tag ยังมีอีกเป็นจำนวนมาก ซึ่งตารางต่อไปนี้** ได้สรุป Tag เกือบทั้งหมดตามมาตรฐาน HTML 4.01 โดย tag ที่ไม่ได้แสดงในที่นี้ได้แก่ tag ที่เกี่ยวข้องกับ frame และการจัดรูปแบบเฉพาะที่ไม่ค่อยได้ใช้โดยทั่วไป

Tag พื้นฐาน

<!DOCTYPE>	ระบุประเภทของเอกสาร
<html>	ระบุว่าเป็นเอกสาร HTML
<body>	ระบุองค์ประกอบส่วนของเนื้อหา
<h1> to <h6>	รูปแบบหัวข้อระดับ 1 ถึง 6
<p>	รูปแบบย่อหน้า
 	ขึ้นบรรทัดใหม่
<hr>	ขีดเส้นแนวนอน
<!--...-->	หมายเหตุ

รูปแบบของตัวอักษร

<b>	ตัวหนา
<font>	ปัจจุบันไม่ใช้แล้ว เปลี่ยนเป็น style
<i>	ตัวเอียง
	ตัวเน้น
<big>	ตัวใหญ่
	คำสำคัญ
<small>	ตัวเล็ก
<sup>	รูปแบบยก
<sub>	รูปแบบห้อย
<bdo>	ทิศทางของตัวอักษร
<pre>	ไม่กำหนดรูปแบบ

ลิงค์เชื่อมโยง

<a>	กำหนดชื่อของส่วนใดๆ ในเอกสาร
<link>	กำหนดแหล่งข้อมูลอ้างอิง

รายการ

	รายการไม่เรียงลำดับ
	รายการเรียงลำดับ
	ระบุรายการแต่ละรายการ
<dl>	รายการแบบนิยาม
<dt>	รายการแต่ละรายการในรายการแบบนิยาม
<dd>	คำอธิบายแต่ละรายการในรายการแบบนิยาม

รูปภาพ

<img>	กำหนดองค์ประกอบรูปภาพ
<map>	สร้างแผนที่ในรูปภาพ
<area>	กำหนดบริเวณในแผนที่ในรูปภาพ

ตาราง

<table>	กำหนดองค์ประกอบตาราง
<caption>	หัวข้อตาราง
<th>	หัวข้อคอลัมน์
<tr>	กำหนดองค์ประกอบแถว
<td>	กำหนดองค์ประกอบเซล
<thead>	ส่วนหัวตาราง
<tbody>	ส่วนเนื้อหาของตาราง
<tfoot>	ส่วนท้ายตาราง
<col>	กำหนดคุณสมบัติ
<colgroup>	จัดกลุ่มคอลัมน์

รูปแบบตัวอักษร

<style>	กำหนดรูปแบบตัวอักษร
	กำหนดส่วนขององค์ประกอบในเอกสาร
	กำหนดส่วนขององค์ประกอบในเอกสาร

คุณสมบัติของเอกสาร HTML

<head>	ส่วนหัวข้อเอกสาร
<title>	ชื่อของเอกสาร
<meta>	คุณสมบัติอื่นๆ ของเอกสาร
<base>	URL ฐาน

ฟอร์ม

<form>	กำหนด
<input>	กำหนดองค์ประกอบการรับข้อมูล
<textarea>	บริเวณกรอกข้อความตัวอักษร
<button>	ปุ่มคลิก
<select>	ตัวเลือก
<optgroup>	กลุ่มลัวเลือกอย่างหนึ่งอย่างใด
<option>	กำหนดตัวเลือกแต่ละตัว
<label>	กำหนดชื่อขององค์ประกอบในการรับ
<fieldset>	จัดกลุ่มขององค์ประกอบรับข้อมูล
<legend>	กำหนดชื่อของ fieldset

การเขียนโปรแกรม

<script>	กำหนดองค์ประกอบสร้างสคริปต์
<noscript>	ข้อความแสดงกรณีใช้สคริปต์ไม่ได้
<object>	ระบุวัตถุในเอกสาร

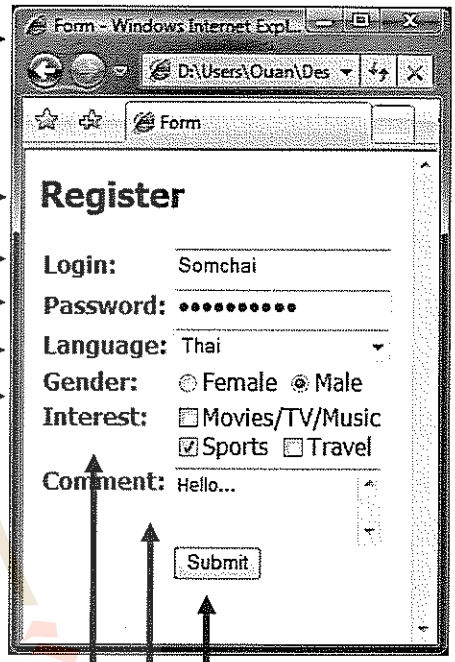
<param> กำหนดพารามิเตอร์สำหรับวัตถุ

10.4.2 HTML Forms

HTML Form เป็นส่วนติดต่อกับผู้ใช้ที่ใช้รับข้อมูลลักษณะต่างๆ พิจารณาตัวอย่างหน้าเว็บที่มีองค์ประกอบของฟอร์มดังต่อไปนี้

ตัวอย่าง HTML Form

```
<html>
<head><title>Form</title><!--JavaScript--></head>
<body>
<h2><b>Register</b></h2>
<form name="form1" method="post"
  action="register.php">
<table>
  <tr>
    <td><b>Login:</b></td>
    <td><input type="text" name="login"></td>
  </tr>
  <tr>
    <td><b>Password:</b></td>
    <td><input type="password"
          name="password"></td>
  </tr>
  <tr>
    <td><b>Language:</b></td>
    <td>
      <select name="language">
        <option value="ch-CH" />Chinese
        <option value="en-US" />English
        <option value="th-TH" />Thai
       </select>
    </td>
  </tr>
  <tr>
    <td><b>Gender:</b></td>
    <td>
      <input type="radio" name="sex" value="f">Female
      <input type="radio" name="sex" value="m">Male
    </td>
  </tr>
  <tr>
    <td><b>Interest:</b></td>
    <td>
      <input type="checkbox" name="checkbox1"
        value="Entertainment">Movies/TV/Music
      <input type="checkbox" name="checkbox2"
        value="Sports">Sports
      <input type="checkbox" name="checkbox3"
        value="Travel">Travel
   </td>
  </tr>
  <tr>
    <td valign="top"><b>Comment:</b></td>
    <td>
      <textarea rows="3"></textarea><br />
      <!--Submit Button-->
      <input type="submit" value="Submit" />
   </td>
  </tr>
```



```
</table></form></body></html>
```

จากตัวอย่าง

<input type="text">	somchai	คือกล่องข้อความ (text box) นั่นเอง
<input type="password">	••••••••	คือกล่องสำหรับกรอกรหัส นิยมให้กรอกยืนยันอีกครั้ง
<select> และ <option>	Thai	คือตัวเลือกแบบรายการ drop-down โดย option ใช้กำหนดสมาชิกในรายการ
<input type="radio">	Female Male	คือตัวเลือกอย่างใดอย่างหนึ่ง (radio button)
<input type="checkbox">	Sports Travel	คือตัวเลือกได้หลายรายการ (check box)
<input type="textarea">	Hello...	คือบริเวณข้อความ (text area) สำหรับกรอกข้อความที่มีความยาวมาก
<input type="submit">	Submit	คือปุ่มส่งแบบฟอร์ม นอกจากนี้ยังมีปุ่มแบบ button และ reset ซึ่งเป็นปุ่มทั่วไปและปุ่มล้างข้อมูลอีกด้วย

การกำหนดแอทริบิวต์ต่างๆ สำหรับแต่ละ input ให้เหมาะสมสามารถศึกษาได้จากตัวอย่าง HTML สำหรับการนำไปใช้นั้น โดยปกติค่าในฟอร์มจะต้องถูกส่งไปประมวลผลยังเว็บเซิร์ฟเวอร์ ข้อมูลในฟอร์มจะถูกส่งไปพร้อมกับวัตถุการร้องขอ HTTP REQUEST ไปยังวัตถุปลายทางที่ต้องการให้ประมวลผล ในที่นี้เมื่อฟอร์มถูก Sumit วัตถุที่จะประมวลผลนั้นคือไฟล์ "register.php" ซึ่งระบุไว้ที่แอทริบิวต์ "action" ใน tag <form> ในบรรทัดที่ 5 ของโค้ดตัวอย่าง <form name="form1" method="post" **action="register.php"**> ซึ่งจะเห็นว่าฟอร์มเป็นเพียงส่วนติดต่อกับผู้ใช้เท่านั้น สำหรับการประมวลผล ฟอร์มต้องอาศัยโปรแกรมประยุกต์ที่เขียนขึ้นบนเว็บเซิร์ฟเวอร์ ซึ่งในที่นี้จะใช้ PHP ในการจัดการข้อมูลต่อไป

นอกจาก HTML แล้ว การพัฒนาเว็บที่สมบูรณ์ยังประกอบด้วยการประยุกต์ HTML ร่วมกับมาตรฐานการจัดรูปแบบ CSS และการเขียนโปรแกรมในฝั่งไคลเอนท์ซึ่งได้แก่ JavaScript ซึ่งเป็นภาษาที่นิยมใช้ในฝั่งเว็บบราวเซอร์ อย่างไรก็ตาม ทั้ง CSS และ JavaScript นั้นมีรายละเอียดปลีกย่อยพอสมควรและไม่ได้เป็นส่วนสำคัญของการประยุกต์ใช้ฐานข้อมูลบนเว็บ จึงแนะนำเพียงเบื้องต้น และจะยกตัวอย่างเพื่อใช้งานจริงในหัวข้อ กรณีศึกษาต่อไป

10.4.3 CSS

CCS ย่อมาจาก Cascading Style Sheet ใช้สำหรับกำหนดรูปแบบของเนื้อหาในเอกสาร HTML โดยมีแนวคิดในการแยกรูปแบบออกจากตัวเนื้อหาของหน้าเว็บ ซึ่งจะมีประโยชน์ในการกำหนดรูปแบบใหม่ได้ทันทีโดยไม่ต้องแก้ไขหน้าเว็บ แต่แก้ไขที่ตัวรูปแบบ โดยรูปแบบจะบรรจุอยู่ใน Style Sheet แล้วผนวกเข้ากับเอกสาร HTML สำหรับแสดงผล CSS นี้สามารถกำหนดได้ทั้งรูปแบบตัวอักษร และการจัดวางของหน้าเว็บ ใน HTML 4.01 ซึ่งเป็นเวอร์ชันล่าสุดแนะนำให้ยกเลิกการใช้ Tag <font> ของ HTML โดยเปลี่ยนมาใช้ <style> แทน และการจัดวางหน้าเว็บให้เลี่ยงใช้ <table> แต่ให้เปลี่ยนมาใช้ CSS แทนเช่นกัน โดย <table> ยังคงไว้สำหรับการสร้างองค์ประกอบที่เป็นตารางข้อมูล

CSS มีรูปแบบคล้ายคลึงกันกับการใช้ <style> ในตัวอย่างที่ **

```
<p style="font-family:angsana new;
     font-size:40px;
     color:#0000FF">
     รูปแบบ ขนาด และสี
</p>
```

แต่ตัวอย่างนี้เป็นการฝัง style เข้ากับเนื้อหา การใช้งาน CSS นั้นเราจะแยกรูปแบบออกมา โดยกำหนดรูปแบบขององค์ประกอบต่างๆ ของเว็บ เช่น <p> ให้มีรูปแบบตามที่กำหนด <h1> ก็ให้มีรูปแบบอื่นๆ เป็นต้น ซึ่งการสร้าง CSS นั้นเกินขอบเขตที่จะกล่าวถึงรายละเอียดในที่นี้

10.4.4 JavaScripts

สถาปัตยกรรมโปรแกรมประยุกต์บนเว็บที่ได้กล่าวถึงในหัวข้อ 10.3.3 นั้นแสดงโดยเน้นถึงการประมวลผลที่เกิดขึ้นบนเว็บเซิร์ฟเวอร์เป็นส่วนใหญ่ แต่โปรแกรมประยุกต์บนเว็บสามารถเน้นที่การประมวลผลที่เกิดขึ้นบนเว็บบราวเซอร์ก็ได้ โดยเฉพาะในปัจจุบันที่การสื่อสารบนเครือข่ายอินเตอร์เน็ตมีความเร็วสูง โปรแกรมประยุกต์ที่นิยมใช้กันโดยทั่วไป เช่น โปรแกรมประมวลผลคำ โปรแกรมกระดานคำนวณ สามารถพัฒนาขึ้นด้วยภาษาคอมพิวเตอร์ที่เรียกว่าสคริปต์ ใช้งานบนเว็บบราวเซอร์ได้อย่างมีประสิทธิภาพ

อย่างไรก็ตามโดยปกติเมื่อกล่าวถึง JavaScripts จะหมายถึงสคริปต์ที่พัฒนาขึ้นเพื่อประมวลผลเบื้องต้นกับข้อมูลในฝั่งผู้ใช้ รวมถึงการโต้ตอบกับผู้ใช้ในรูปแบบที่หลากหลายมากขึ้นกว่าการใช้เพียง HTML สำหรับกิจกรรมหลักๆ ที่ JavaScript มักถูกใช้งานได้แก่การตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้ป้อนให้เป็นไปตามเงื่อนไข เช่น TextBox บาง TextBox จำเป็นต้องกรอกข้อมูล ห้ามปล่อยว่างไว้ หรือค่าที่กรอกจะต้องเป็นเพียงตัวเลขเท่านั้น เป็นต้น

ด้วยความสามารถในการสร้างระบบการตรวจสอบย่อยๆ จนถึงโปรแกรมขนาดใหญ่ที่มีความซับซ้อนนั้น JavaScript จึงมีรายละเอียดปลีกย่อยมากที่ไม่อาจกล่าวถึงรายละเอียดได้ในที่นี้ JavaScript ไม่ได้มีความเกี่ยวข้องกับการพัฒนาฐานข้อมูลบนเว็บโดยตรง อย่างไรก็ตามได้มีการยกตัวอย่างการใช้งาน JavaScript เป็นตัวอย่างง่ายๆ ดังด้านล่าง และนอกจากนี้ยังมีการใช้ JavaScript ประกอบกรณีศึกษาเพื่อให้โปรแกรมประยุกต์บนเว็บที่พัฒนาขึ้นมีความสามารถโดยสมบูรณ์ในกรณีศึกษา 7-Elephant ซึ่งสามารถศึกษาได้จากหัวข้อ 10.**

ตัวอย่าง JavaScript อย่างง่าย

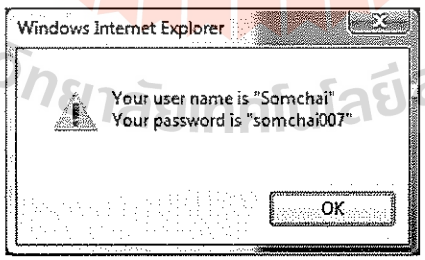
ตัวอย่างอย่างง่ายนี้เป็นการแสดงข้อมูลจากฟอร์มในตัวอย่าง ** โดยแสดงข้อมูลชื่อ login และ password เราแทรกฟังก์ชัน JavaScript ที่ทำงานให้เราใน HTML ภายใต้ tag <script> โดยให้โค้ดช่วงที่เหลือมีค่าดังเดิม ดังนี้

```
<html>
<head><title>Form</title>
<script type="text/javascript">
function show() {
  alert ("Your user name is \"" +
         document.forms["form1"].elements["login"].value + "\"\n" +
         "Your password is \"" +
         document.forms["form1"].elements["password"].value + "\"\n");
}
</script>
</head>
Follow by codes in example ** ...
```

จากตัวอย่างเป็นการสร้างฟังก์ชันโดย JavaScript ชื่อว่าฟังก์ชัน show() ซึ่งสิ่งที่จะได้จากฟังก์ชันคือการแสดงข้อมูลจากกล่อง "login" และกล่อง "password" ด้วยคำสั่ง alert() ฟังก์ชันดังกล่าวจะถูกเรียกใช้งานได้ก็ต่อเมื่อมีการเรียกใช้งาน ยกตัวอย่างการเรียกใช้งานฟังก์ชัน show() โดยการปรับเปลี่ยนโค้ดในส่วนของปุ่ม "Submit" ในตัวอย่าง ** เป็น

```
<!--Submit Button-->
<input type="button" value="Submit" onclick="show()" />
```

จากปุ่มที่มี type เป็น "submit" นั้นเราเปลี่ยนเป็น "button" และเพิ่มการจัดการเหตุการณ์เมื่อผู้ใช้คลิกปุ่มหรือ "onclick" ให้เรียกฟังก์ชัน show() ขึ้นมาทำงาน เว็บบราวเซอร์จะเรียกใช้งานฟังก์ชัน show() ที่เขียนด้วย JavaScript แสดงเป็นหน้าต่างข้อความ ดังตัวอย่างผลลัพธ์ต่อไปนี้



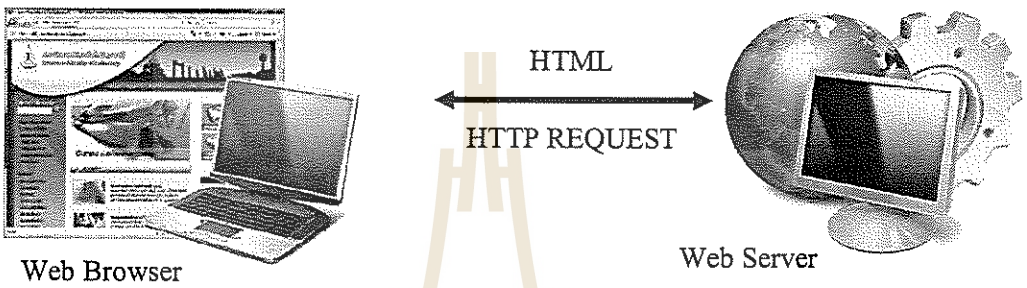
10.5 การพัฒนาโปรแกรมประยุกต์บนเว็บด้วย PHP

ให้หัวข้อที่แล้วเป็นการอธิบายถึง HTML ซึ่งเป็นชั้นของการนำเสนอ (presentation layer) ในสถาปัตยของโปรแกรมประยุกต์บนเว็บที่ได้แสดงในรูป ** ตามหัวข้อที่ 10.3.3 ในหัวข้อนี้จะกล่าวถึงหัวใจสำคัญของการพัฒนาโปรแกรมประยุกต์บนเว็บซึ่งเป็นส่วนของแอพลิเคชันเซิร์ฟเวอร์ในสถาปัตยกรรมของโรปแกรมประยุกต์บนเว็บ ในที่นี้

ได้อธิบายถึงการใช้ PHP ซึ่งเป็นภาษาที่ใช้ในการพัฒนาโปรแกรมประยุกต์แบบ open source ที่ได้รับความนิยมอย่างแพร่หลาย โดยจะได้กล่าวถึงแนวคิดและวิวัฒนาการของการให้บริการโปรแกรมประยุกต์บนเว็บจนกระทั่งเป็น PHP

10.5.1 CGI: The Common Gateway Interface และ Application Servers

หลักการสำคัญในการทำงานของโปรแกรมประยุกต์บนเว็บนั้นได้แก่การประมวลผลตามการร้องขอ (request) ของผู้ใช้แต่ละการร้องขอเพื่อสร้างผลลัพธ์ตอบสนอง (response) ไปยังผู้ใช้ ซึ่งผลลัพธ์อยู่ในรูปแบบของ HTML แนวคิดพื้นฐานนี้เป็นแนวคิดสำคัญในการสร้างโปรแกรมประยุกต์บนเว็บ

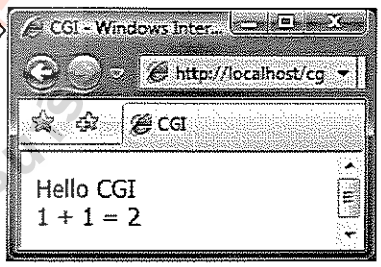


โปรแกรมประยุกต์บนเว็บในยุคแรกๆ เรียกว่า CGI หรือ Common Gateway Interface ซึ่งผู้ใช้สามารถเรียกใช้งานโปรแกรมต่างๆ ผ่านทางเว็บโดยไม่ต้องทราบว่าโปรแกรมมีหลักการทำงานอย่างไร เพียงป้อนข้อมูลนำเข้าที่ครบถ้วนเท่านั้น โปรแกรม CGI ในยุคแรกพัฒนาขึ้นด้วยภาษาคอมพิวเตอร์ทั่วไป โดยเฉพาะภาษา C เราสามารถยกตัวอย่างให้เห็นภาพของหลักการทำงานด้วยส่วนของโปรแกรมต่อไปนี้

ตัวอย่าง การทำงานของ CGI

```
printf("<html>\n");
printf("<head>\n");
printf("   <title>CGI</title>\n");
printf("</head>\n");
printf("<body>\n");
printf("   Hello CGI<br />\n");
printf("   1 + 1 = %d\n", 1+1);
printf("</body>");
printf("</html>");
```

```
<html>
<head>
  <title>CGI<title>
<head>
<body>
  Hello CGI<br />
  1 + 1 = 2
</body></html>
```



จากตัวอย่าง โปรแกรม CGI ที่เขียนขึ้นด้วยภาษา Cทางด้านซ้ายจะถูกเรียกให้ประมวลผลโดยเว็บเซิร์ฟเวอร์เมื่อมีการร้องขอ ผลลัพธ์ที่ได้จากการประมวลผลเป็นการพิมพ์ค่าธรรมดาด้วยคำสั่ง printf() คำสั่ง printf() ใช้สำหรับแสดงผลออกทางหน่วยแสดงผลที่กำหนดไว้ ซึ่งเราสามารถกำหนดให้หน่วยแสดงผลเป็นพอร์ต 80 ซึ่งหมายถึงการส่งข้อมูลออกทางพอร์ตที่ให้บริการเว็บได้ ในตัวอย่างนั้นมีการประมวลผลซึ่งสร้าง HTML ขึ้นอย่างอัตโนมัติที่คำสั่งการคำนวณ printf(“ 1 + 1 = %d\n”, 1+1); ซึ่ง 2 เป็นผลลัพธ์ที่ได้จากการคำนวณจะแทนที่ %d ตามรูปแบบของคำสั่งภาษา C สำหรับ \n นั้นเป็นการขึ้นบรรทัดใหม่ สังเกตจากผลลัพธ์ HTML ที่อยู่ในกรอบข้อความตรงกลางว่า HTML tag แต่ละ tag จะขึ้นบรรทัดใหม่ด้วย \n ยกเว้น </body></html> ที่อยู่ในบรรทัดเดียวกัน ไม่ได้ใช้ \n เพื่อขึ้นบรรทัดใหม่

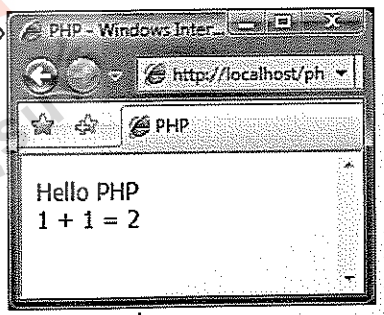
จากนั้นผลลัพธ์ที่ได้จะถูกส่งออกไปยังเว็บบราวเซอร์ปลายทาง เว็บบราวเซอร์ทำการ render HTML และได้ผลลัพธ์ดังภาพด้านขวา

เราสามารถแทนที่โปรแกรมข้างต้นได้ด้วยโค้ดขนาดใหญ่ที่สลับซับซ้อนเพื่อการประมวลผลใดๆ ตามที่ผู้ใช้ต้องการรวมถึงการรับข้อมูลจากผู้ใช้ การเชื่อมต่อกับฐานข้อมูลต่างๆ ได้ นั่นเป็นจุดกำเนิดของการพัฒนาโปรแกรมประยุกต์บนเว็บ นักพัฒนาพบว่าภาษาในสมัยก่อนไม่ได้ออกแบบมาสำหรับการประมวลผลในลักษณะการใช้งานบนเว็บมากนัก ดังสังเกตได้จากการที่ผู้พัฒนาจำเป็นต้องใช้คำสั่ง printf() อยู่ตลอดเวลา รวมถึงการจัดการกับข้อความโดยเฉพาะการดำเนินการพื้นฐานที่ต้องกระทำเป็นอย่างมากในเอกสาร HTML ได้แก่การนำข้อความมาต่อกันนั้นไม่สามารถทำได้โดยสะดวกด้วยภาษาที่มีอยู่ Perl เป็นภาษาในยุคถัดมาที่การพัฒนาโปรแกรมประยุกต์บนเว็บเริ่มได้รับความนิยม ซึ่ง Perl นั้นมีข้อดีในการจัดการข้อความโดยสะดวก อย่างไรก็ตาม Perl ยังต้องอาศัยคำสั่ง print ในการแสดงผลข้อมูล โดยในที่สุดได้มีความพยายามพัฒนาภาษาต่างๆ ที่เหมาะสมสำหรับการพัฒนาโปรแกรมประยุกต์บนเว็บ ได้แก่ ASP (Active Server Page), JSP (Java Server Page) และ PHP เป็นต้น ภาษาที่ใช้สำหรับสร้างโปรแกรมประยุกต์บนเว็บในปัจจุบันมักมีรูปแบบที่ฝังอยู่ในเอกสาร HTML และจะประมวลผลเมื่อมีคำสั่งในการประมวลผล เราเรียกภาษาเหล่านี้ว่าภาษาสคริปต์บนฝั่งเซิร์ฟเวอร์ (server-sided scripts) ซึ่งในที่นี้ PHP เป็นภาษาที่จะได้ใช้ในการอธิบายและยกตัวอย่างเพื่อพัฒนาโปรแกรมประยุกต์บนเว็บ PHP เป็นภาษาที่ได้รับความนิยมมากที่สุดภาษาหนึ่งเนื่องจากความง่ายในการใช้งานบนเว็บ การเชื่อมต่อกับฐานข้อมูลที่สะดวก และโดยเฉพาะอย่างยิ่งเป็น open source ที่สามารถใช้ได้ฟรี รูปแบบของ PHP ที่สำคัญที่แตกต่างจากภาษาที่ใช้พัฒนา CGI ในยุคแรกคือการแทรกตัวอยู่ในเอกสาร HTML สำหรับเนื้อหาในส่วนของ HTML นั้นไม่จำเป็นต้องมีการประมวลผล แต่เมื่อมีคำสั่ง PHP อยู่ที่ใดในเอกสาร ระบบจะทำการประมวลผล สามารถยกตัวอย่างให้เห็นภาพของหลักการทำงานของ PHP ด้วยส่วนของโปรแกรมต่อไปนี้

ตัวอย่าง PHP

```
<html>
<head>
  <title>PHP</title>
</head>
<body>
  <?php
    echo "Hello PHP<br />\n";
    echo "1 + 1 = " . (1+1);
  ?>
</body>
</html>
```

```
<html>
<head>
  <title>PHP<title>
<head>
<body>
  Hello PHP<br />
  1 + 1 = 2
</body>
</html>
```



จากตัวอย่างเป็นเอกสาร HTML โดยทั่วไป แต่มีส่วนของคำสั่ง PHP ซึ่งฝังอยู่ในส่วนที่คล้าย Tag “<?php ... ?>” ลองพิจารณาคำสั่ง echo "Hello PHP
\n"; เป็นการแสดงผลธรรมดาคล้ายคำสั่ง printf() และมีการคำนวณในคำสั่ง echo "1 + 1 = " . (1+1);

หัวข้อต่อไปนี้อธิบายถึงการทำงาน PHP ในรายละเอียด อย่างไรก็ตาม PHP มีรายละเอียดปลีกย่อยมากพอสมควร ในที่นี้จะแนะนำคำสั่ง PHP ต่างๆ ที่จำเป็นในการพัฒนาฐานข้อมูลบนเว็บเบื้องต้นซึ่งจะเป็นพื้นฐานในการประยุกต์และศึกษาเพิ่มเติมเพื่อพัฒนาการใช้งานในขั้นสูงต่อไป

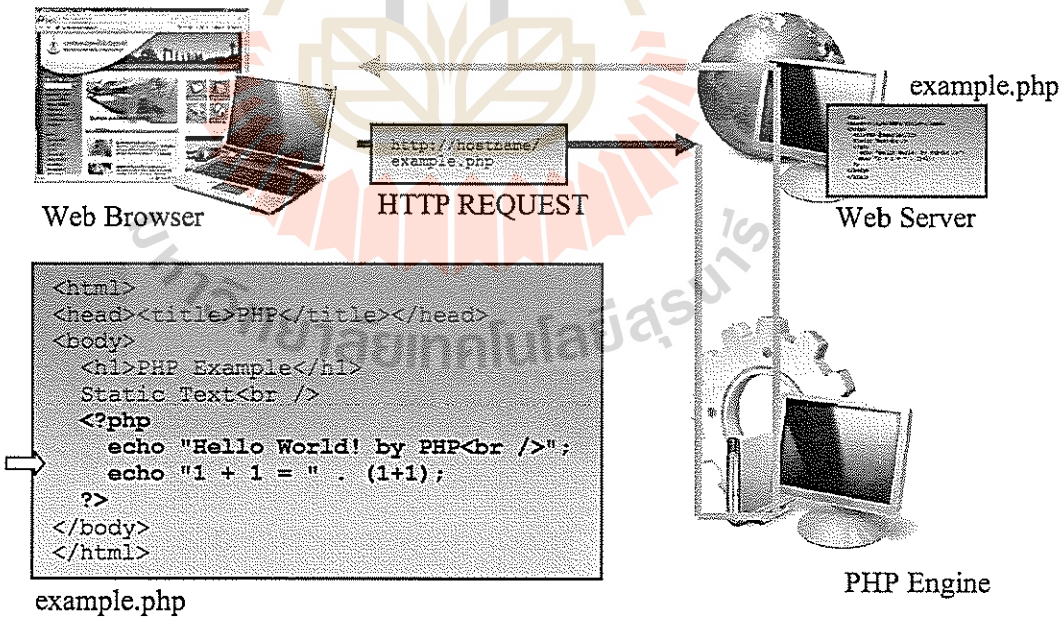
10.5.2 PHP

PHP ย่อมาจาก PHP Hypertext Preprocessor เป็นภาษาสคริปต์บนเซิร์ฟเวอร์สำหรับสร้างโปรแกรมประยุกต์บนเว็บที่ได้รับความนิยมอย่างสูง โดยจะฝังตัวอยู่บนเว็บเซิร์ฟเวอร์ มี syntax ใกล้เคียงกับภาษา C ตัวอย่างของไฟล์ PHP มีรูปแบบดังต่อไปนี้

ตัวอย่าง PHP

```
<html>
<head><title>PHP</title></head>
<body>
  <h1>PHP Example</h1>
  Static Text<br />
  <?php
    echo "Hello World! by PHP<br />";
          echo "1 + 1 = " . (1+1);
  ?>
</body>
</html>
```

สามารถอธิบายหลักการทำงานของภาษาได้ด้วยแผนภาพตัวอย่างที่เชื่อมโยงกับสถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บที่ได้อธิบายในหัวข้อ ** ดังต่อไปนี้

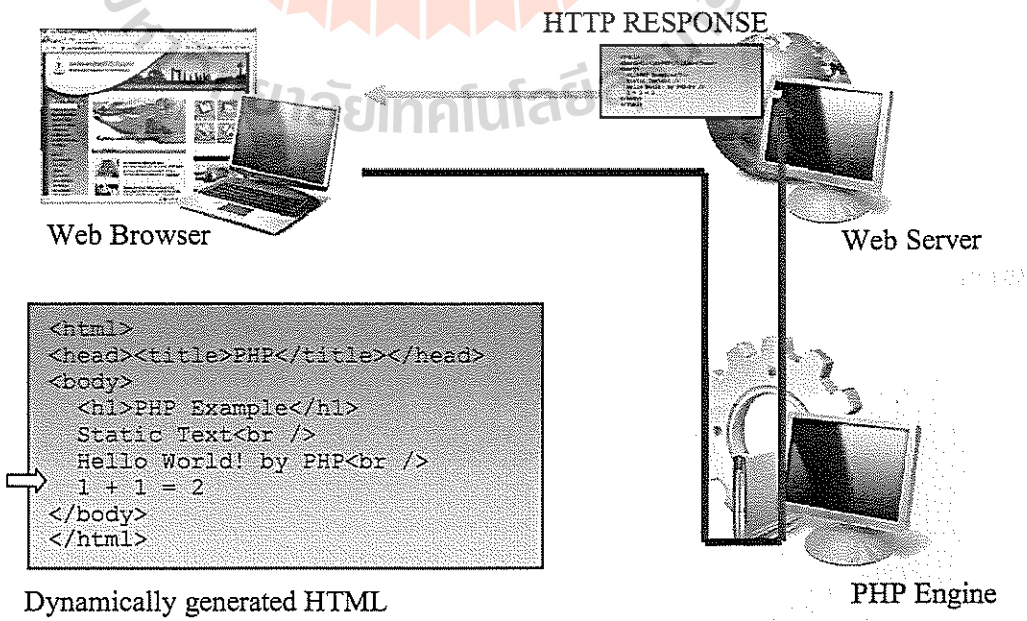


การทำงานของ PHP ต่างจาก HTML ตรงที่ PHP ต้องได้รับการประมวลผลก่อนในกรณีที่มีคำสั่ง PHP อยู่ในเอกสาร PHP จึงต้องทำงานผ่านเว็บเซิร์ฟเวอร์ จากตัวอย่าง ไฟล์ PHP นี้วางอยู่บนเว็บเซิร์ฟเวอร์ ในตัวอย่างนี้ให้ชื่อไฟล์ว่า example.php สังเกตว่าไฟล์ PHP นั้นใช้นามสกุล .php ขั้นตอนการทำงานเริ่มต้นด้วยผู้ใช้เว็บทำการร้องขอไฟล์

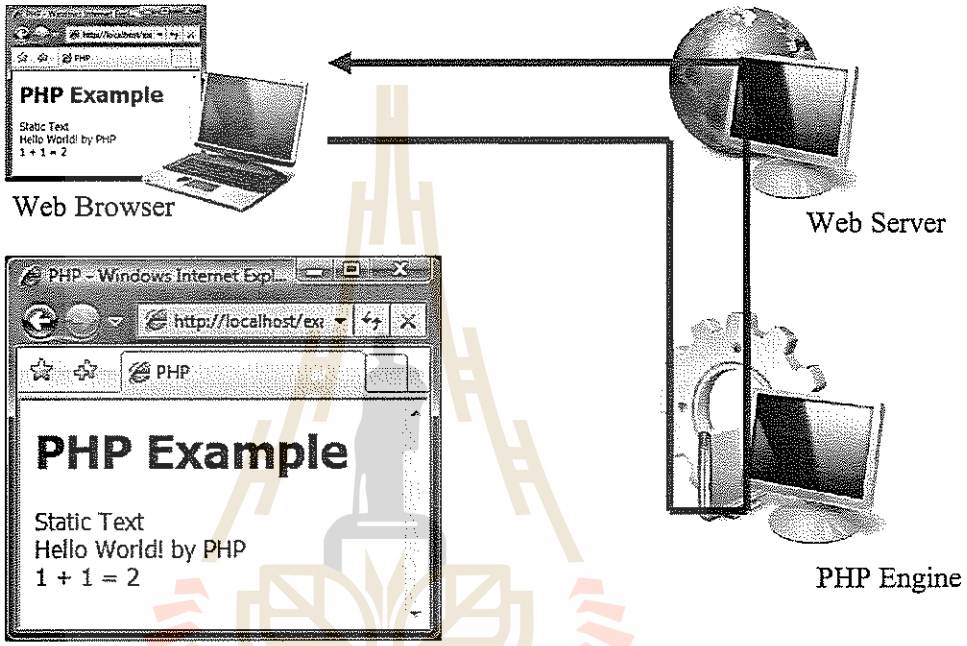
ดังกล่าวผ่านทาง URL ของเว็บบราวเซอร์ เช่น http://localhost/example.php เว็บบราวเซอร์จะทำการสร้างวัตถุการร้องขอ HTTP REQUEST เพื่อส่งไปยังเครื่อง localhost (ชื่อเครื่องสมมติสำหรับตัวอย่าง ซึ่ง localhost แท้ที่จริงเป็นคำเฉพาะในการเข้าถึงเครื่องของตนเอง) วัตถุการร้องขอนี้นอกจากจะประกอบไปด้วยไฟล์ปลายทางที่ต้องการ ได้แก่ example.php แล้ว ยังประกอบไปด้วยข้อมูลต่างที่ใช้ในการสื่อสารและประมวลผล เช่น เครื่องคอมพิวเตอร์ปลายทางที่ร้องขอข้อมูล หมายเลข IP ของเครื่องต้นทางที่ร้องขอ เพื่อให้เว็บเซิร์ฟเวอร์ส่งข้อมูลกลับมาได้ ตลอดจนข้อมูลที่ผู้ใช้ส่งไปประมวลผลยังเครื่องเซิร์ฟเวอร์ เช่นข้อมูลที่กรอกในแบบฟอร์มเป็นต้น วัตถุการร้องขอนี้จะถูกส่งผ่านเครือข่ายในองค์กรหรือเครือข่ายอินเตอร์เน็ตเพื่อไปยังเครื่องปลายทางที่ทำการร้องขอข้อมูล

เมื่อเว็บบราวเซอร์ได้รับการร้องขอจากเว็บไคลเอนท์โดยการรอฟังการร้องขอด้วยพอร์ตที่ให้บริการเว็บ (พอร์ต 80 โดยทั่วไป) เว็บบราวเซอร์จะส่งวัตถุที่ร้องขอมากลับไปยังเครื่องต้นทาง ในที่นี้ไฟล์ example.php เป็นวัตถุที่จะส่งกลับไป สำหรับข้อมูลในส่วนแรกของไฟล์ example.php ที่เป็นข้อมูล HTML นั้นสามารถส่งกลับไปได้ในทันที แต่เมื่อเว็บเซิร์ฟเวอร์กำลังสร้างวัตถุการตอบรับ HTTP RESPONSE ที่ประกอบไปด้วยผลลัพธ์นั้น ไฟล์ example.php มีการฝังโค้ด PHP อยู่ภายในเครื่องหมาย <?php หรือ <? และเครื่องหมายสิ้นสุดโค้ด PHP ได้แก่ ?> โค้ดในส่วนนี้จะไม่ถูกส่งกลับไปยังเว็บบราวเซอร์โดยทันทีแต่จะได้รับการประมวลผลก่อน โดยเว็บเซิร์ฟเวอร์จะอาศัย PHP engine เพื่อช่วยในการประมวลผลคำสั่ง PHP นั้นๆ PHP engine นี้จะต้องถูกติดตั้งไว้บนเว็บเซิร์ฟเวอร์ก่อน จากแผนภาพ PHP engine นี้คือแอพลิเคชันเซิร์ฟเวอร์นั่นเอง ซึ่งโดยปกติจะติดตั้งและทำงานร่วมกันกับเว็บเซิร์ฟเวอร์บนเครื่องๆ เดียวกัน

คำสั่ง echo "Hello World! by PHP
"; เป็นการแสดงผลลัพธ์ข้อความธรรมดา "Hello World! by PHP
" ซึ่งผลลัพธ์นั้นจะถูกบรรจุอยู่ในวัตถุตอบสนอง RESPONSE ที่จะส่งกลับไปยังผู้ร้องขอหน้าเว็บ ซึ่งผลลัพธ์จะแทรกอยู่ในตำแหน่งตามที่แทรกคำสั่ง PHP ไว้ สำหรับคำสั่ง echo "1 + 1 = " . (1+1); เป็นการแสดงข้อความภายในเครื่องหมายอัญประกาศ "..." เช่นเดียวกับคำสั่งก่อนหน้า แต่คำสั่งหลังนี้มีการประมวลผล โดยเครื่องหมายจุด "." ใช้สำหรับดำเนินการนำข้อความมาต่อกันใน PHP ข้อความ "1 + 1 =" จึงตามด้วยผลลัพธ์ที่ได้จากการคำนวณ (1+1) ซึ่งปรากฏผลลัพธ์พร้อมแผนภาพประกอบคำอธิบายต่อไปนี้



ผลลัพธ์ที่ได้จากการประมวลผลไฟล์ PHP คือวัตถุตอบรับที่สร้างขึ้นทันทีทันใด วัตถุตอบรับที่ถูกสร้างขึ้นเสร็จสมบูรณ์ หรือในขณะที่ถูกสร้างนั้น จะถูกส่งกลับไปยังเครื่องที่ทำการร้องขอในรูปแบบของเอกสาร HTML ประกอบไฟล์ต่างๆ ที่เกี่ยวข้อง จากนั้นเว็บบราวเซอร์ปลายทางจะทำการ render และแสดงผลลัพธ์ซึ่งสามารถแสดงได้ดังนี้



หลักการทำงานของ PHP นี้เป็นหลักการที่คล้ายคลึงกับหลักการทำงานของภาษาพัฒนาโปรแกรมประยุกต์บนเว็บอื่นๆ โดยทั่วไป หัวข้อต่อไปนี้อธิบาย syntax พื้นฐานข้อ PHP และประยุกต์ใช้งานฐานข้อมูลบนเว็บด้วย PHP ในที่สุด

10.5.2.1 พื้นฐานของคำสั่ง PHP

PHP ฝังอยู่ในเอกสาร HTML ภายในเครื่องหมาย <?php ซึ่งบอกจุดเริ่มต้นของคำสั่ง PHP เราสามารถใช้เครื่องหมาย <? โดยไม่มีคำว่า php ก็ได้ และใช้เครื่องหมาย ?> บอกจุดสิ้นสุดของกลุ่มคำสั่ง คำสั่งแต่ละคำสั่งแยกออกจากกันด้วยเครื่องหมาย semi colon ";" PHP สามารถแทรกไว้ที่ใดของเอกสาร HTML ก็ได้ syntax ของ PHP มีความคล้ายคลึงกับ syntax ของภาษา C เป็นอย่างมาก ซึ่งจะสังเกตได้จากแนะนำรูปแบบคำส่ง PHP ต่อไป

PHP ใช้เครื่องหมาย // และ /* ... */ สำหรับหมายเหตุ (comment) โค้ดเช่นเดียวกับภาษา C

ตัวอย่าง การใช้ Comment ใน PHP

```
<?php
  // This is a comment.
  /* This is a multiline
     comment. */
?>
```

จากตัวอย่างเป็นการหมายเหตุในโค้ด จะไม่มีผลลัพธ์ใดๆ ปรากฏ

การแสดงผลลัพธ์ PHP ใช้คำสั่ง echo ในการแสดงผลลัพธ์ ข้อความจะถูกแสดงในวัตถุตอบรับไปยังเว็บบราวเซอร์ เพื่อให้เว็บบราวเซอร์ทำการ render อีกทีหนึ่ง

ตัวอย่าง การแสดงผลลัพธ์ ด้วย echo

```
<?php
  echo "Hello World!";
?>
```

จากตัวอย่าง ผลลัพธ์ที่ได้คือคำว่า Hello World! เท่านั้น มีลักษณะดังตัวอย่าง**

10.5.2.2 ตัวแปร

ตัวแปรกใน PHP ใช้สำหรับจัดเก็บค่าต่างๆ เช่นตัวอักษร/ข้อความ/สายอักขระ/สตริง (string) ตัวเลข หรืออาร์เรย์ เป็นต้น ซึ่งตัวแปรสามารถกำหนดค่าใหม่ได้เรื่อยๆ การตั้งชื่อตัวแปรของ PHP มีหลักคล้ายคลึงกับในภาษา C มีข้อกำหนดดังนี้

- ต้องเริ่มต้นด้วยตัวอักษรภาษาอังกฤษ หรือเครื่องหมายขีดเส้นใต้ "_"
- ชื่อภายในชื่อตัวแปรยังสามารถมีตัวเลขได้
- การใช้งานตัวแปรทุกครั้งต้องมีเครื่องหมาย $ นำหน้าเสมอ ไม่ว่ากรณีใดๆ ทั้งสิ้น นักพัฒนา PHP เริ่มต้นมักจะลืมเครื่องหมายดังกล่าว
- ตัวแปรใน PHP เป็นตัวแปรชนิด loosely-typed กล่าวคือไม่กำหนดชนิดของตัวแปรที่ตายตัว ตัวแปรจะปรับเปลี่ยนชนิดไปตามข้อมูลที่กำหนด ซึ่งเป็นข้อดีอีกอย่างหนึ่งของ PHP
- ไม่จำเป็นต้องมีการประกาศตัวแปร สามารถเรียกใช้งานได้ทันที

  การกำหนดค่าให้กับตัวแปรใช้เครื่องหมาย "=" ในการกำหนดค่า

ตัวอย่าง การใช้งานตัวแปร

```
<?php
  $txt="Hello World!";
  echo $txt;
?>
```

จากตัวอย่าง $txt เป็นตัวแปร ได้ผลลัพธ์ HTML ดังนี้

```
Hello World!
```

10.5.2.3 ตัวดำเนินการ

ตัวดำเนินการทั้งหมดนอกเหนือจากตัวดำเนินการสตริงทำงานเช่นเดียวกับในภาษา C ดังนี้

ตัวดำเนินการกับสตริง (String Operator)

	เครื่องหมายจุด เป็นตัวดำเนินการเพียงตัวเดียวเท่านั้นที่ดำเนินการกับสตริง เป็นการจำสตริงมาต่อกัน (concatenate)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

+, - , *, และ /	สำหรับการบวก ลบ คูณ และหาร ตามลำดับ
%	สำหรับการหารเพื่อหาค่าเศษของการหาร(modulus)
++ และ --	สำหรับเพิ่มค่า และลดค่าตัวถูกดำเนินการทีละ 1 เช่น 1++ มีค่าเท่ากับ 2

ตัวดำเนินการกำหนดค่า (Assignment Operators)

=	กำหนดค่าตัวถูกดำเนินการทางซ้ายให้เท่ากับค่าของตัวถูกดำเนินการทางขวา
+=, -=, *=, /= และ %=	เป็นการเขียนแบบย่อของการดำเนินการระหว่างตัวถูกดำเนินการทางซ้ายและทางขวา จากนั้นนำค่าผลลัพธ์มาเก็บไว้ที่ตัวถูกดำเนินการทางซ้าย เช่น x+=y มีค่าเท่ากับ x=x+y

ตัวดำเนินการเปรียบเทียบ (Compare Operators)

== และ !=	เท่ากับหรือไม่ และ ไม่เท่ากับหรือไม่ เช่น 1==1 ได้ค่าเป็นจริง 1!=1 ได้ค่าเป็นเท็จ
> และ <	มากกว่าหรือไม่ และ น้อยกว่าหรือไม่
>= และ <=	มากกว่าหรือเท่ากับหรือไม่ และ น้อยกว่าหรือเท่ากับหรือไม่

ตัวดำเนินการทางตรรกะ (Logic Operators)

&&	และ (and) เช่น กำหนด x และ y ให้มีค่าเท่ากับ 1 (x==y && x!=y) ได้ค่าเป็นเท็จ เนื่องจากมีนิพจน์หนึ่งที่เป็นเท็จ
||	หรือ (or)
!	นิเสธ (not)

ตัวอย่างการใช้งานแสดงตัวดำเนินการบางส่วนแสดงประกอบกับการใช้คำสั่ง PHP อื่นๆ ในหัวข้อถัดไป

10.5.2.4 คำสั่งควบคุม

คำสั่งควบคุมหมายถึงคำสั่งที่มีการตัดสินใจและเลือกกระบวนการถัดไปที่จะดำเนินการ แบ่งเป็น 2 ชนิดใหญ่ๆ ได้แก่คำสั่งควบคุมชนิดเลือกทำ และทำซ้ำ สำหรับคำสั่งควบคุมชนิดเลือกทำที่จะอธิบายได้แก่คำสั่ง if...else และ elseif คำสั่งควบคุมชนิดทำซ้ำหรือ loop ได้แก่ while, do...while, for และ foreach

คำสั่ง if...else

เป็นคำสั่งควบคุมชนิดเลือกทำ ถ้า...แล้ว มี syntax ดังนี้

```
if (<condition>)
        <statement (if the condition is true)>;
else
        <statement (if the condifiton is false)>;
```

โดย

if	คือคำสั่ง "ถ้า" เพื่อตรวจสอบเงื่อนไขและระบุรูปแบบการควบคุม
condition	คือเงื่อนไขที่จะตรวจสอบ
else	คือคำสั่ง "มิฉะนั้น" เป็นคำสั่งประกอบ "ถ้า" สำหรับดำเนินการกรณีเงื่อนไขไม่เป็นจริง
statement	คือชุดคำสั่งที่จะให้ดำเนินการ ชุดคำสั่งที่จะให้ดำเนินการในกรณีที่เงื่อนเป็นจริงจะอยู่หลังจาก "if" และชุดคำสั่งที่จะให้ทำในกรณีเงื่อนไขไม่เป็นจริงอยู่หลัง else

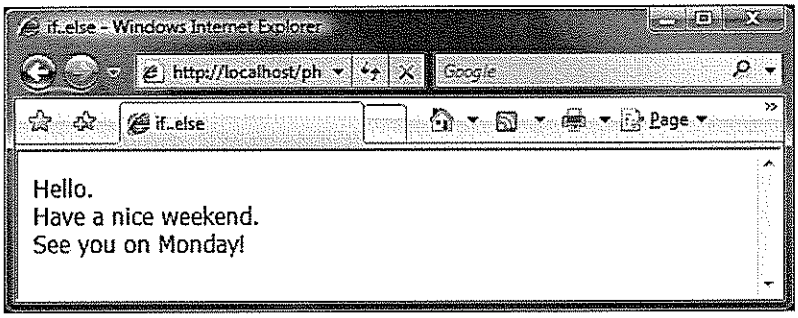
คำสั่ง if ไม่จำเป็นต้องมี else ก็ได้ แต่คำสั่ง else จำเป็นต้องใช้ร่วมกับคำสั่ง if นอกจากนี้เรายังสามารถใช้คำสั่ง elseif สำหรับตรวจสอบเงื่อนไขอีกครั้งในกรณีที่เงื่อนไขก่อนหน้าไม่เป็นจริง ซึ่งสามารถแสดงตัวอย่างการใช้งานได้ดังนี้

ตัวอย่าง if...else

ตัวอย่างต่อไปนี้เป็นตัวอย่างการตรวจสอบวันปัจจุบัน หากเป็นวันศุกร์ จะแสดงข้อความ "Hello. Have a nice weekend. See you on Monday!" มิเช่นนั้นแสดงข้อความ "Too bad. It's not Friday"

```
<?php
  $d=date("D");
  if ($d=="Fri")
  {
    echo "Hello.<br />";
    echo "Have a nice weekend.<br />";
    echo "See you on Monday!";
  } else {
      echo "Too bad. It's not Friday";
  }
?>
```

จากตัวอย่าง $d เป็นตัวแปรมีค่าเท่ากับ date("D") ซึ่งฟังก์ชัน date() เป็นฟังก์ชันแสดงข้อมูลวันและเวลาปัจจุบัน โดย ("D") เป็นการจัดรูปแบบของข้อมูลวันเวลาปัจจุบันให้แสดงเพียงวันของสัปดาห์เท่านั้น คำสั่ง if($d=="Fri") คือการตรวจสอบ ถ้าตัวแปร $d หรือวันปัจจุบันเป็นวันศุกร์ จะแสดงข้อความที่ตามหลัง if แต่หากไม่ใช่วันศุกร์ จะแสดงข้อความที่ตามหลัง else สมมติว่าวันนี้เป็นวันศุกร์ ผลลัพธ์ที่ได้เป็นดังนี้



ตัวอย่าง elseif

ตัวอย่างต่อไปนี้ตรวจสอบวันปัจจุบัน ในกรณีที่เป็นวันเสาร์ จะแสดงข้อความ "Have a nice weekend!" แต่หากไม่ใช่จะมีการตรวจสอบด้วยเงื่อนไขอีกครั้งด้วยคำสั่ง elseif ตรวจสอบว่าเป็นวันอาทิตย์หรือไม่ ถ้าใช่จะแสดงข้อความ "Have a nice Sunday!" แต่หากไม่ใช่ทั้ง 2 วัน จะแสดงข้อความ "Oh, Working!"

```
<?php
        $d=date("D");
        if ($d=="Sat")
            echo "Have a nice weekend!";
        elseif ($d=="Sun")
            echo "Have a nice Sunday!";
        else echo "Oh, Working!";
?>
```

สมมติวันปัจจุบันคือวันศุกร์ จะแสดงข้อความ "Oh, Working!" ให้สังเกตว่าคำสั่งที่จะให้ดำเนินการนั้นไม่จำเป็นต้องอยู่ภายในเครื่องหมายวงเล็บปีกการ "{ }" ก็ได้ ในกรณีที่คำสั่งที่จะให้ดำเนินการมีเพียงบรรทัดเดียว แต่ถ้าคำสั่งที่จะให้ดำเนินการตามเงื่อนไข หรือแม้แต่การทำซ้ำที่จะได้กล่าวถึงต่อไป จะต้องอยู่ภายในเครื่องหมายวงเล็บปีกกาเพื่อจัดกลุ่มให้เป็นชุดคำสั่งที่จะต้องดำเนินการเป็นชุด

คำสั่ง while และ do…while

เป็นคำสั่งควบคุมชนิดทำซ้ำ ซึ่งจะทำชุดคำสั่งที่กำหนดจนกระทั่งเงื่อนไขที่ตรวจสอบไม่เป็นจริง มี syntax ดังนี้

```
while (<condition>)
        <statement (if the condition is true)>;
```

โดย

while	คือคำสั่ง "ในขณะที่" เพื่อตรวจสอบเงื่อนไขและระบุชนิดการควบคุม
condition	คือเงื่อนไขที่จะตรวจสอบ
statement	คือชุดคำสั่งที่จะให้ดำเนินการในขณะที่เงื่อนไขเป็นจริง

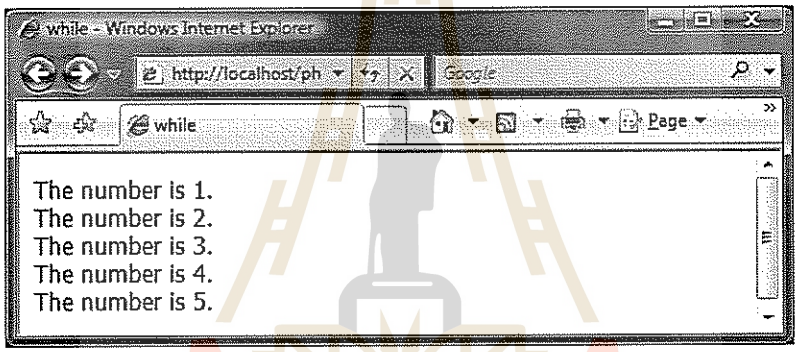
นอกจากนี้ยังมีคำสั่ง do…while ซึ่งมีรูปแบบคล้ายคลึงกันกับคำสั่ง while ต่างกันเพียงการตรวจสอบเงื่อนไขนั้นจะกระทำหลังจากที่ได้ดำเนินการคำสั่งที่จะให้ทำซ้ำแล้วอย่างน้อยหนึ่งครั้งโดยการใช้คำสั่ง do ประกอบ

ตัวอย่าง while

ตัวอย่างต่อไปนี้เป็นการแสดงข้อความ "The number is" แล้วตามด้วยหมายเลขลำดับของการทำซ้ำ

```
<?php
        $i=1;
        while($i <= 5)
        {
            echo "The number is " . $i . ".<br />";
            $i++;
        }
?>
```

จากตัวอย่าง $i คือตัวแปรกำหนดค่าเริ่มต้นเป็น 1 เมื่อเริ่มประมวลผลคำสั่ง while เงื่อนไข $i <= 5 เป็นการตรวจสอบว่าค่า $i น้อยกว่าหรือเท่ากับ 5 หรือไม่ ถ้าใช่จะแสดงข้อความพร้อมหมายค่า $i จากนั้นจะทำการเพิ่มค่า $i ขึ้นอีก 1 ด้วยคำสั่ง $i++ แล้วทำซ้ำโดยการตรวจสอบเงื่อนไข $i <= 5 อีกครั้ง ทำเช่นนี้ไปเรื่อยๆ จนกระทั่งเงื่อนไขไม่เป็นจริงจึงหลุดออกจากการทำซ้ำ แสดงตัวอย่างผลลัพธ์ได้ดังนี้



ตัวอย่าง do...while

ตัวอย่างต่อไปนี้มีลักษณะเช่นเดียวกันกับตัวอย่างการใช้ while ก่อนหน้า เพียงแต่เปลี่ยนเป็นการใช้ do...while

```
<?php
        $i=1;
        do
        {
            echo "The number is " . $i . ".<br />";
            $i++;
        } while ($i<=5);
?>
```

คำสั่ง while ถูกย้ายไปหลังกลุ่มคำสั่งที่จะทำซ้ำโดยใช้คำสั่ง do เพื่อระบุขอบเขตการเริ่มต้นของคำสั่งที่จะทำซ้ำ ดังนั้น ไม่ว่าเงื่อนไขจะเป็นจริงหรือไม่จริง ชุดคำสั่งที่จะทำซ้ำต้องถูกดำเนินการอย่างน้อย 1 ครั้ง ได้ผลลัพธ์เช่นเดียวกันกับตัวอย่าง while ก่อนหน้า

<u>คำสั่ง for</u>

คำสั่ง for เป็นคำสั่งควบคุมชนิดทำซ้ำ มี syntax ดังนี้

for (<variable initialization>; <condition>; <variable alteration>)

<statement (if the condition is true)>;

โดย

for คือคำสั่ง "สำหรับ" กำหนดการทำซ้ำ

condition คือเงื่อนไขที่ใช้ในการตรวจสอบ

initialize คือการกำหนดค่าเริ่มต้นของตัวแปรที่จะใช้เพิ่ม ลด หรือเปลี่ยนแปลงค่า สำหรับใช้ในเงื่อนไขตรวสอบ

alteration คือการเปลี่ยนแปลงของตัวแปร เช่นการเพิ่ม ลด หรือเปลี่ยนแปลงอื่นๆ

statement คือชุดคำสั่งที่จะให้ดำเนินการในขณะที่เงื่อนไขเป็นจริง

for เป็นคำสั่งควบคุมที่ใช้ในการทำซ้ำเช่นเดียวกับคำสั่ง while และ do...while แต่มีรูปแบบที่ต่างกันออกไป มักใช้กับการทำซ้ำที่ทราบจำนวนรอบในการทำซ้ำ

ตัวอย่าง for

ตัวอย่างต่อไปนี้เป็นการแสดงข้อความ "The number is" แล้วตามด้วยหมายเลขลำดับของการทำซ้ำ

```
<?php
        for ($i=1; $i<=5; $i++)
        {
           echo "The number is " . $i . ".<br />";
        }
?>
```

จากตัวอย่าง for คือการกำหนดการเริ่มต้นของคำสั่งทำซ้ำ $i คือตัวแปรกำหนดค่าเริ่มต้นเป็น 1 ด้วยคำสั่ง %i=1 เงื่อนไข $i <= 5 เป็นการตรวจสอบว่าค่า $i น้อยกว่าหรือเท่ากับ 5 หรือไม่ ถ้าใช่จะแสดงข้อความพร้อมหมายค่า $i จากนั้นจะทำการเพิ่มค่า $i ขึ้นอีก 1 ด้วยคำสั่ง $i++ แล้วทำซ้ำโดยการตรวจสอบเงื่อนไข $i <= 5 อีกครั้ง ทำเช่นนี้ไปเรื่อยๆ จนกระทั่งเงื่อนไขไม่เป็นจริงจึงหลุดออกจากการทำซ้ำ ได้ผลลัพธ์เช่นเดียวกันกับในตัวอย่าง while

<u>คำสั่ง foreach</u>

คำสั่ง foreach เป็นคำสั่งควบคุมชนิดทำซ้ำ มี syntax ดังนี้

foreach (<collection> as <collection_item>)

<statement (refer to collection_item)>;

โดย

foreach คือคำสั่ง "สำหรับแต่ละ"

collection คือชุดของข้อมูลที่จะนำมาดำเนินการทำซ้ำ

as ประกอบคำสั่ง foreach เพื่อกำหนดข้อมูลแต่ละข้อมูลในชุดข้อมูล "เป็น" ตัวแปรชื่ออะไร

statement คือชุดคำสั่งที่จะให้ดำเนินการทำซ้ำ

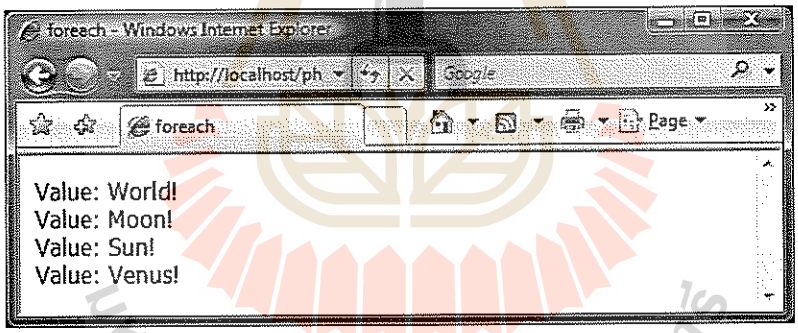
คำสั่ง foreach นี้เป็นคำสั่งทำซ้ำที่มีลักษณะพิเศษไม่เหมือนกับคำสั่งควบคุมทั่วไป โดยที่คำสั่ง foreach นี้ไม่มีการตรวจสอบเงื่อนไข แต่เป็นการทำซ้ำให้ครบจำนวนครั้งเท่ากับจำนวนข้อมูลสมาชิกในชุดข้อมูลที่จะทำซ้ำ อีกทั้งในแต่ละรอบของการทำซ้ำ จะเป็นการนำค่าในชุดข้อมูลทีละค่ามาดำเนินการในแต่ละรอบ จนครบจำนวนรอบและได้ดำเนินการกับข้อมูลทีละข้อมูลจนหมด

ตัวอย่าง foreach

ตัวอย่างต่อไปนี้เป็นการแสดงข้อความ "Value: " จากนั้นตามด้วยค่าดวงดาวที่เป็นสมาชิกในอาเรย์ $arr_solar

```
<?php
        $arr solar = array("World", "Moon", "Sun", "Venus");
        foreach ($arr solar as $arr item)
        {
           echo "Value: " . $arr item . "<br />";
        }
?>
```

จากตัวอย่าง $arr_solar เป็นตัวแปรชนิดอาเรย์ (ชนิดของข้อมูลแปรเปลี่ยนไปตามรูปแบบของข้อมูลที่บรรจุ) ซึ่งได้แก่อาเรย์ของชื่อดวงดาว World, Moon, Sun และ Venus คำสั่ง foreach เป็นการเริ่มคำสั่งทำซ้ำ $arr_item เป็นตัวแปรที่ใช้อ้างอิงถึงข้อมูลชื่อดวงดาวแต่ละดวงในอาร์เรย์ $arr_solar คำสั่ง foreach ($arr_solar as $arr_item) เปรียบเสมือนการสั่งงานว่า สำหรับแต่ละข้อมูล $arr_item ให้ดำเนินการตามคำสั่งทำซ้ำ ซึ่งคำสั่งทำซ้ำได้แก่การแสดงข้อความพร้อมชื่อดวงดาวแต่ละดวง แสดงผลลัพธ์ได้ดังนี้



10.5.2.5 การจัดการฟอร์ม

องค์ประกอบของการใช้งานโปรแกรมประยุกต์บนเว็บที่สำคัญคือการโต้ตอบกับผู้ใช้ การรับข้อมูลจากผู้ใช้อาศัย HTML Forms ที่ได้อธิบายในหัวข้อ 10.4.2 ข้อมูลที่ได้รับจากฟอร์มจะถูกส่งมายังเว็บเซิร์ฟเวอร์พร้อมกับ HTTP REQUEST ในหัวข้อนี้จะได้อธิบายถึงวิธีการในการนำข้อมูลจากฟอร์มมาใช้ประมวลผลด้วย PHP ซึ่งได้แก่การใช้ตัวแปร $_POST, $_GET และ $_REQUEST

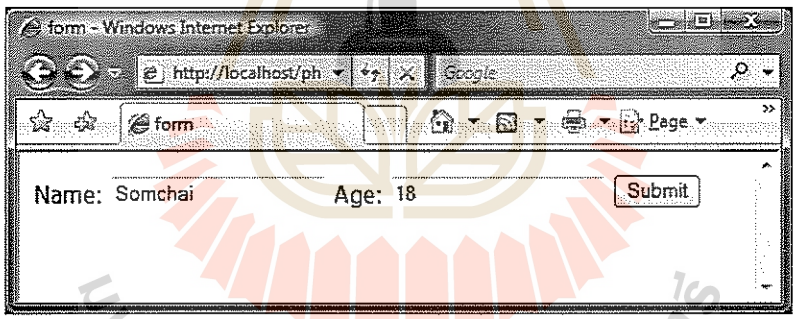
การใช้งาน $_POST

ตัวแปร $_POST เป็นตัวแปรพิเศษใน PHP ซึ่งมีวิธีการใช้ที่ง่าย เพียงอ้างถึงชื่อขององค์ประกอบของฟอร์มจากหน้าที่รับค่า เราก็จะสามารถนำค่ามาใช้งานได้ทันที เช่น ในหน้า form.php มีกล่องข้อความที่ชื่อ name หน้า form กำหนด action ให้ทำกับหน้า show.php การเรียกใช้ค่าที่กรอกในกล่องข้อความ “name” สามารถทำได้อย่างง่ายดายโดยใช้ตัวแปร $_POST[“name”] ดังตัวอย่างต่อไปนี้

form.php

```
<html> <head><title>form</title></head>
<body><form name="form1" method="post" action="show.php">
        Name: <input type="text" name="name" />
        Age: <input type="text" name="age" />
        <input type="submit" value="Submit" />
</form>
</body>
</html>
```

จากตัวอย่าง method=“post” ซึ่งเป็นแอทริบิวต์หนึ่งของฟอร์ม “form1” เป็นการกำหนดวิธีการส่งค่าไปยังเว็บเซิร์ฟเวอร์รูปแบบหนึ่ง ซึ่งหมายถึงการห่อหุ้มค่าของฟอร์มไว้ วิธีการส่งค่าไปยังเว็บเซิร์ฟเวอร์อีกวิธีหนึ่งได้แก่ GET ซึ่งจะได้อธิบายต่อไป action=“show.php” หมายถึงเมื่อทำการส่งแบบฟอร์มแล้วให้ไปประมวลผลยังเครื่องเซิร์ฟเวอร์ด้วยไฟล์ show.php ชื่อของกล่องข้อความทั้ง name และ age จะใช้ในการอ้างถึงเพื่อนำข้อมูลไปประมวลผล form.php ยังไม่มีสคริปต์ฝั่งเซิร์ฟเวอร์ แสดงผลที่ได้ด้วยเว็บบราวเซอร์ดังนี้



จากตัวอย่าง หน้าจอแสดงการกรอกข้อมูลตัวอย่าง “Somchai” ในกล่องข้อความ “name” และ “18” ในกล่องข้อความ “age” เมื่อคลิกปุ่ม “Submit” เว็บบราวเซอร์จะทำการส่งการร้องของไปยังหน้า show.php ซึ่งวัตถุร้องขอนี้บรรจุข้อมูลในฟอร์มไปด้วย การนำข้อมูลจากฟอร์มไปใช้สามารถยกตัวอย่างได้ด้วยหน้า show.php ดังนี้

show.php

```
<html> <head><title>form</title></head>
<body>
Hello <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

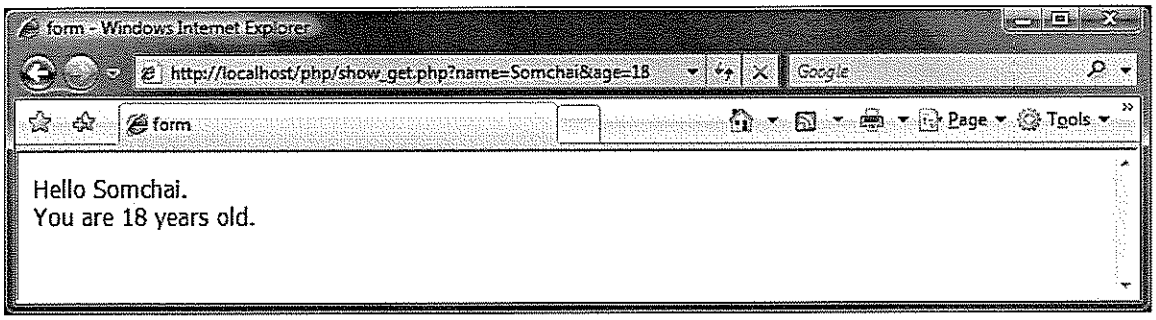
จากตัวอย่าง <?php echo $_POST["name"]; ?> คือการแสดงข้อความโดยอ้างถึงตัวแปร $_POST[“name”] ซึ่ง “name” ก็คือกล่องข้อความ “name” จากหน้า form.php นั่นเอง เช่นเดียวกันกับการใช้ $_POST[“age”] ในบรรทัดถัดมา แสดงตัวอย่างผลลัพธ์ได้ดังนี้

ยังมีวิธีการอีกวิธีหนึ่งในการส่งข้อมูลไปประมวลผลยังเว็บเซิร์ฟเวอร์ได้โดยไม่ต้องอาศัยฟอร์มในการส่งข้อมูล สามารถทำได้โดยการส่งข้อมูลไปพร้อมกับ URL โดยกำหนดค่าให้กับตัวแปรที่ต้องการส่งค่าต่อท้าย URL ในรูปแบบ http://<web_server_address>/<target_object>?<variable_name>=<value>[&<variable_name>=<value>...] เช่น http://localhost/show_get.php?**name=Somchai&age=18** เป็นการส่งข้อมูลตัวแปร name และ age โดยกำหนดค่าให้เป็น “Somchai” และ “18” ตามลำดับ หน้าที่ทำการประมวลผลได้แก่ show_get.php บางเครื่อง localhost วิธีการส่งค่าไปเช่นนี้ วัตถุเป้าหมายที่ทำการร้องขอจะนำข้อมูลจากตัวแปรไปใช้ได้โดยอาศัยวัตถุ $_GET[“<ชื่อตัวแปร>”] เช่น $_GET[“name”] และ $_GET[“age”] สำหรับการนำค่าของตัวแปร “name” และ “age” ไปใช้ โค้ดที่แสดงตัวอย่างการนำค่าใน URL ไปใช้แสดงได้ด้วยหน้า show_get.php ดังนี้

show_get.php

```
<html> <head><title>form</title></head>
<body>
Hello <?php echo $_GET["name"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old.
</body>
</html>
```

จากตัวอย่าง การแสดงข้อมูลจาก URL ด้วยวิธี GET นี้สามารถทำได้ง่ายๆ เพียงอ้างถึงตัวแปรด้วย $_GET[“name”] และ $_GET[“age”] เพื่อนำข้อมูลมาแสดง สมมติว่าผู้ใช้ป้อน URL ในช่องกรอก address ของเว็บดังนี้ http://localhost/show_get.php?name=Somchai&age=18 เป็นการเรียกแสดงผลหน้า show_get.php พร้อมทั้งส่งค่าตัวแปรไปด้วย ได้ผลลัพธ์ดังนี้



การรับข้อมูลของหน้า PHP ด้วยวิธี GET นี้ยังสามารถรับข้อมูลจากฟอร์มได้เช่นเดียวกัน แต่ต้องกำหนดแอทริบิวต์ "method" ของฟอร์มให้เป็น "GET" เช่น <form name="form1" **method="GET"** action="show_get.php" ซึ่งสามารถนำไปประยุกต์ใช้กับตัวอย่างที่ ** ได้

การรับข้อมูล-ส่งข้อมูลของฟอร์มด้วยวิธี GET นี้มักถูกมองว่ามีความปลอดภัยน้อยกว่าการใช้วิธี POST แต่มีความสะดวกและความหลากหลายในการประยุกต์ใช้มากกว่า

นอกจากนี้ ใน PHP ยังมีตัวแปรอีกตัวแปรหนึ่งให้เรียกใช้ คือตัวแปร $_REQUEST ซึ่งสามารถแสดงข้อมูลจากวัตถุการร้องขอที่ส่งข้อมูลมาได้ทั้งแบบ POST และ GET เช่น

```
Hello <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```

เป็นการแสดงค่า name และ age ซึ่งส่งค่ามาจากฟอร์มด้วย method POST หรือ GET ก็ได้ อีกทั้งยังแสดงค่าจาก URL ก็ได้

การจัดการข้อมูลฟอร์มของ HTML ยังสัมพันธ์โดยตรงกับสิ่งที่เราเรียกว่าการจัดการ session เนื่องจากการเชื่อมต่อระหว่างเว็บบราวเซอร์กับเว็บเซิร์ฟเวอร์เป็นแบบไม่คงการเชื่อมต่อไว้ (connectionless) หมายความว่าหลังเว็บบราวเซอร์ส่งข้อมูลการตอบกลับซึ่งเป็นผลลัพธ์มายังเว็บบราวเซอร์แล้ว การเชื่อมต่อจะถูกตัดขาด การสื่อสารกันระหว่างเว็บบราวเซอร์และเว็บเซิร์ฟเวอร์อีกครั้งนั้นจะต้องมีกระบวนการที่ทำให้เว็บเซิร์ฟเวอร์ทราบสถานะการทำงานของโปรแกรมประยุกต์ เช่น ผู้ใช้ที่เพิ่งส่งการร้องขอเข้ามาทำการล็อกอินเข้าสู่ระบบหรือยัง เป็นต้น ซึ่งต้องอาศัยกลไกในการจัดการ session ซึ่งได้แก่การใช้งาน $_SESSION และ $_COOKIES โดยในที่นี้จะไม่กล่าวถึงในรายละเอียดแต่จะแสดงตัวอย่างการใช้งานประกอบกรณีศึกษา 7-Elephant ในหัวข้อที่ **

10.5.2.6 PHP และฐานข้อมูล

การพัฒนาฐานข้อมูลบนเว็บสามารถทำได้โดยอาศัยแนวคิดและเครื่องมือต่างๆ ที่ได้อธิบายมาทั้งหมด ได้แก่ HTML, PHP และการจัดการฟอร์มมาประสานเข้ากับระบบจัดการฐานข้อมูลโดยใช้ชุดคำสั่งของ PHP ที่ใช้ในการทำงานกับระบบจัดการฐานข้อมูล ในหัวข้อนี้จะได้อธิบายถึงการประยุกต์ใช้ PHP กับระบบจัดการฐานข้อมูล MySQL ซึ่งเป็นระบบจัดการฐานข้อมูลที่นิยมใช้กับ PHP เนื่องจากเป็น open source เช่นเดียวกัน และออกแบบให้ทำงานเข้ากันได้เป็นอย่างดี อย่างไรก็ตาม PHP สามารถใช้ร่วมกับระบบจัดการฐานข้อมูลของผู้ผลิตรายอื่นได้

ขั้นตอนของการใช้งานฐานข้อมูลบนเว็บเชิงเทคนิคแบ่งเป็น 4 ขั้นตอนใหญ่ๆ ได้แก่ การเชื่อมต่อ การดำเนินการคำสั่ง SQL การใช้งานข้อมูล และการปิดการเชื่อมต่อ

1) **การเชื่อมต่อ**

การเชื่อมต่อหมายถึงการเชื่อมต่อระหว่าง application engine ซึ่งในที่นี้คือ PHP engine ไปยังฐานข้อมูลที่ต้องการเรียกใช้ผ่านทางระบบจัดการฐานข้อมูล ซึ่งมักประกอบด้วยขั้นตอนย่อยๆ 2 ขั้นตอนได้แก่ 1) การระบุที่อยู่หรือชื่อของเครื่องคอมพิวเตอร์ที่บรรจุฐานข้อมูลตลอดจนระบุตัวฐานข้อมูลที่ต้องการเรียกใช้ 2) เปิดการเชื่อมต่อไปยังฐานข้อมูลที่ระบุ ซึ่งในบางครั้ง เราอาจจะเปิดการเชื่อมต่อเข้าไปยังเครื่องคอมพิวเตอร์ที่ให้บริการฐานข้อมูลก่อนแล้วจริงระบุฐานข้อมูล ขึ้นอยู่กับภาษาที่ใช้ในการพัฒนาโปรแกรมประยุกต์บนเว็บ

2) **การดำเนินการคำสั่ง SQL**

หลังจากเปิดการเชื่อมต่อระหว่าง application engine และ DBMS แล้ว เราสามารถดำเนินการกับฐานข้อมูลใดๆ โดยการใช้คำสั่ง SQL ที่ได้อธิบายในบทที่ 7 และ 8 นั่นเอง การดำเนินการคำสั่ง SQL สามารถแบ่งเป็นขั้นตอนย่อยได้ 2 ขั้นตอนคือ 1) การกำหนดคำสั่ง SQL และ 2) การสั่งคำสั่งไปยัง DBMS ซึ่งทั้ง 2 ขั้นตอนสามารถรวมกันได้

3) **การใช้งานข้อมูล**

ในกรณีที่คำสั่ง SQL ที่กระทำกับฐานข้อมูลคือคำสั่งแสดงข้อมูล—SELECT เราสามารถนำผลลัพธ์ที่ได้มาใช้ในการประมวลผล สร้างวัตถุการตอบรับหรือเอกสาร HTML สำหรับแสดงผล ส่งกลับไปยังผู้ใช้ได้ ในกรณีที่เป็นคำสั่งจำพวก INSERT, UPDATE หรือ DELETE ขั้นตอนนี้ก็จะไม่จำเป็นต้องดำเนินการ

ข้อมูลที่ได้จากระบบจัดการฐานข้อมูลมักจะอยู่ในรูปของตารางที่ได้จากคำสั่ง SELECT จะถูกส่งมาเก็บไว้ยังตัวแปรจัดเก็บข้อมูลใน PHP engine ซึ่งเราสามารถใช้งานตารางนี้ได้ต่อไป

4) **การปิดการเชื่อมต่อ**

หลังจากการดำเนินการใช้งานฐานข้อมูลเสร็จสิ้น application engine ต้องทำการปิดการเชื่อมต่อที่เชื่อมไปยัง DBMS การปิดการเชื่อมต่อเป็นสิ่งสำคัญ เนื่องจากการเชื่อมต่อกับฐานข้อมูลจำเป็นต้องสร้างกลไกต่างๆ ในการประสานงานกับระบบจัดการฐานข้อมูลซึ่งอาศัยทรัพยากรของเครื่องให้บริการพอสมควร เมื่อใช้งานฐานข้อมูลเสร็จจึงจะต้องปิดการเชื่อมต่อเพื่อคืนทรัพยากรให้ระบบ

ตัวอย่างต่อไปนี้เป็นตัวอย่างประกอบคำอธิบายการประยุกต์ใช้ PHP ร่วมกับ MySQL สำหรับใช้งานฐานข้อมูลบนเว็บ ซึ่งแต่ละตัวอย่างแสดงการตัวอย่างการดำเนินการคำสั่ง SELECT, INSERT, UPDATE และ DELETE โดยใช้ฐานข้อมูลมหาวิทยาลัย (university) และตารางนักศึกษา (student) ซึ่งสร้างไว้แล้วในฐานข้อมูลโดยใช้สคีมาของระบบฐานข้อมูลมหาวิทยาลัยเช่นเดียวกับบทอื่นๆ ดังต่อไปนี้

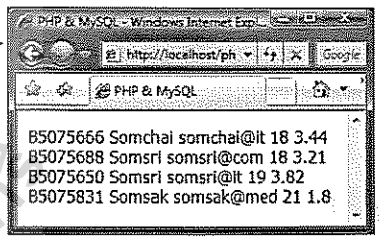
ตัวอย่าง PHP & MySQL: SELECT

ตัวอย่างต่อไปนี้เป็นการแสดงข้อมูลจากฐานข้อมูลมหาวิทยาลัย (university) ตารางนักศึกษา (students) ด้วย PHP ไฟล์ **select.php** ดังนี้

```
1  <html> <head><title>form</title></head>
2  <body>
3  <?
4     // Connect to the database
5     $con = mysql_connect("localhost","root","password");
6     if (!$con) die('Cannot connect: ' . mysql_error());
7     mysql_select_db("university", $con);
8
9     // Execute query
10    $result = mysql_query("SELECT * FROM students");
11    if(!$result) die('SQL error: ' . mysql_error());
12
13    // Utilize data
14    while($row = mysql_fetch_array($result)) {
15       echo $row['sid'] . " " .
16          $row['name'] . " " .
17          $row['login'] . " " .
18          $row['age'] . " " .
19          $row['gpa'] . " <br />" ;
20    }
21
22    // Close the connection
23    mysql_close($con);
24 ?>
25 </body>
26 </html>
```

cid	sid	name	age	gpa
310...	B5075666	Somchai	...	3.44
310...	B5075688	Somsri	...	3.21
310...	B5075650	Somsri	...	3.82
310...	B5075831	Somsak	...	1.80

```
<html><head><title>PHP & MySQL
</title></head>
<body>
B5075666 Somchai ... 18 3.44<br />
B5075688 Somsri ... 18 3.21<br />
B5075650 Somsri ...  19 3.82<br />
B5075831 Somsak ... 21 1.8<br />
</body>
</html>
```



จากตัวอย่าง เนื่องจากโค้ดมีความซับซ้อนพอสมควรจึงใส่เลขของบรรทัดเพื่อการอ้างอิง การใช้งานฐานข้อมูลบนเว็บนั้นประกอบด้วย 4 ขั้นตอนดังได้อธิบายโดยเริ่มจาก

1) การเชื่อมต่อ แสดงได้ด้วยคำสั่งของโปรแกรมในบรรทัดที่ 5

$con = mysql_connect("localhost","root","password"); คำสั่ง mysql_connect() คือคำสั่งที่ใช้สำหรับเปิดการเชื่อมต่อฐานข้อมูล โดยให้ระบุชื่อเครื่องปลายทาง ชื่อผู้ใช้ฐานข้อมูล และรหัสผ่าน ซึ่งในตัวอย่างนี้ได้แก่ “localhost”, “root” และ “password” ตามลำดับ ซึ่งโดยปกติแล้วเราควรจะสร้างบัญชีผู้ใช้ใหม่ที่ไม่ใช่ root ซึ่ง root เป็นชื่อบัญชีผู้มีสิทธิสูงสุดของระบบ เราสามารถอ้างอิงถึงการเชื่อมต่อนี้หลังจากที่เปิดการเชื่อมต่อแล้วด้วยตัวแปร $con เราจะดำเนินการต่างๆ ผ่านตัวแปร $con เสมือนดำเนิการกับระบบจัดการฐานข้อมูล บรรทัดที่ 6 และ 7 เป็นการตรวจสอบความสำเร็จของการเชื่อมต่อ

2) การดำเนินการคำสั่ง SQL $result = mysql_query("SELECT * FROM students"); ในบรรทัดที่ 10

เป็นการใช้งานฐานข้อมูล ซึ่งคำสั่ง mysql_query() นั่นเองที่เป็นคำสั่งที่ใช้ในการส่งคำสั่ง SQL ไปยังระบบจัดการฐานข้อมูลเพื่อดำเนินการใดๆ เพียงส่งคำสั่ง SQL ในรูปแบบพารามิเตอร์ ผลลัพธ์ที่ได้จากการดำเนินการตาม

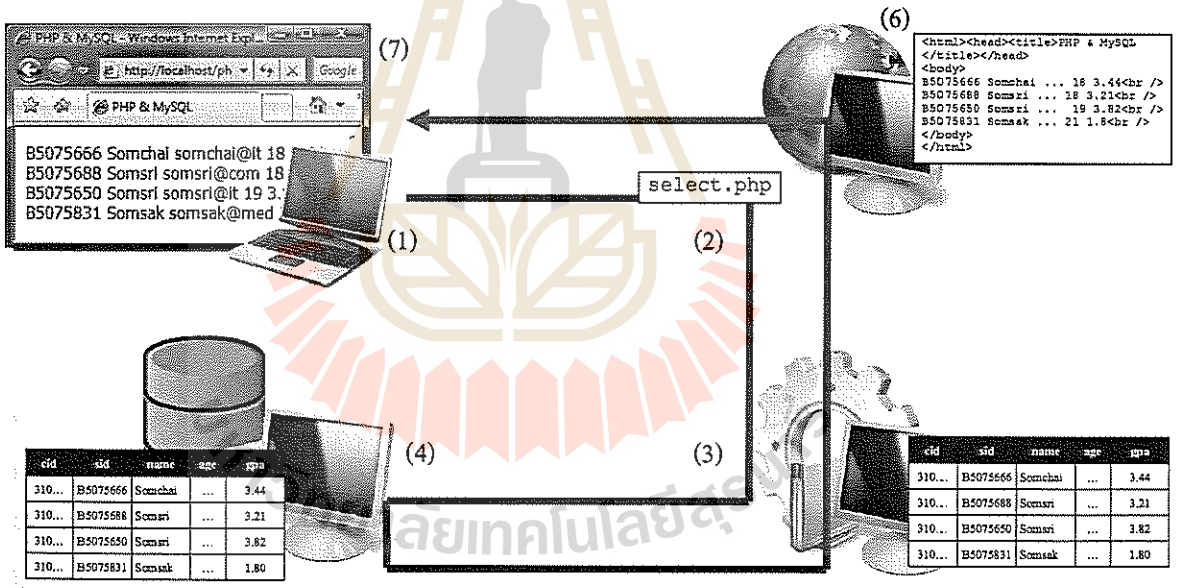
คำสั่ง SQL จะนำมาเก็บไว้ในตัวแปร $result ซึ่งในที่นี้ตัวแปร $result จะมีชนิดของข้อมูลเป็นตารางข้อมูลและผลสำเร็จของการดำเนินการคำสั่ง SQL บรรทัดที่ 11 เป็นการตรวจสอบความสำเร็จของการดำเนินการตามคำสั่ง

3) การใช้งานข้อมูล

เป็นขั้นตอนที่เป็นวัตถุประสงค์หลักของการใช้งานฐานข้อมูลบนเว็บ (ในกรณีแสดงข้อมูล) ของมูลตารางที่ได้มาจากระบบจัดการฐานข้อมูลจะสามารถนำมาใช้ได้โดยการนำค่ามาใช้ทีละแถวๆ คำสั่ง $row = mysql_fetch_array($result) ในบรรทัดที่ 14 เป็นการอ่านค่าข้อมูลมา 1 แถวจากตารางข้อมูลที่เก็บไว้ใน $result ซึ่งในที่นี้คือตารางข้อมูลนักศึกษา การใช้งานคำสั่ง mysql_fetch_array แต่ละครั้งจะแสดงข้อมูลแถวบนสุดแล้วเลื่อนตำแหน่งตัวชี้ไปยังแถวถัดลงมา $result จึงจัดเก็บข้อมูลเป็นแถวๆ และเราสามารถอ้างถึงข้อมูลในแถวได้โดยการระบุชื่อหรือลำดับที่ของคอลัมน์ เช่น result["sid"] หรือ result['sid'] เป็นการอ้างถึงข้อมูลรหัสนักศึกษา คำสั่งบรรทัดที่ 15-19 เป็นการสร้าง HTML เพื่อส่งไปยังเว็บบราวเซอร์ คำสั่งบรรทัดที่ 14 เป็นคำสั่งทำซ้ำในการแสดงข้อมูลซึ่งจะสร้าง HTML จนกว่าแถวข้อมูลนักศึกษาจะถูกนำมาสร้างผลลัพธ์จนหมด ซึ่งมีข้อมูล 4 ระเบียน

4) การปิดการเชื่อมต่อ และสุดท้าย คำสั่ง mysql_close($con) ในบรรทัดที่ 23

เป็นการปิดการเชื่อมต่อซึ่งจะต้องปฏิบัติเพื่อคืนทรัพยากร เช่น หน่วยความจำต่างๆ ให้กับระบบ



จากแผนภาพ เป็นการแสดงขั้นตอนของการใช้งานฐานข้อมูลบนเว็บด้วยมุมมองภาพรวมของสถาปัตยกรรมของโปรแกรมประยุกต์บนเว็บที่ทำงานร่วมกับฐานข้อมูลบนเว็บโดยสมบูรณ์ ขั้นตอนที่ (1) เป็นการเรียกหน้าเว็บที่ต้องการของผู้ใช้ ในที่นี้คือ select.php ซึ่งเว็บบราวเซอร์จะส่งการร้องขอไปยังเว็บบราวเซอร์เพื่อให้ select.php (2) ประมวลผล ในไฟล์ select.php มีสคริปต์ PHP หมายเลข (3) จึงเป็นการประมวลผลด้วย PHP engine ซึ่งคำสั่ง PHP ใน select.php มีการใช้งานฐานข้อมูล PHP engine จึงติดต่อกับระบบจัดการฐานข้อมูล (4) และได้ผลลัพธ์เป็นตารางข้อมูลกลับมายัง PHP engine (5) PHP ทำการสร้าง HTML ผลลัพธ์จากตารางข้อมูลที่ได้มา (6) แล้วส่งกลับมายังผู้ใช้ (7)

ตัวอย่าง Web Database: INSERT

การแทรกข้อมูลนักศึกษาคนใหม่ลงในฐานข้อมูล

insert.php

```
<html> <head><title>PHP & MySQL</title></head>
<body>
<?
   // Connect to the database
   $con = mysql connect("localhost","root","password");
   if (!$con) die('Could not connect: ' . mysql error());
   mysql select db("university", $con);

   // Execute query
   $result = mysql_query("INSERT INTO students (cid, sid, name, login, age,
   gpa) VALUES ('3100904078132', 'B5075832', 'Somnamna', 'somnamna@math', 22,
   2.00)");

   if (!$result) die('Could not connect: ' . mysql error());

   //
   echo "Insert record successfully.";

   // Close the connection
   mysql close($con);
?>
</body>
</html>
```

จากตัวอย่างมีการติดต่อกับฐานข้อมูล ซึ่งมีขั้นตอนใหญ่ๆ 3 ขั้นตอนคล้ายคลึงกันกับการ SELECT ข้อมูล ได้แก่การเชื่อมต่อฐานข้อมูล การดำเนินการคำสั่ง SQL สำหรับขั้นตอนการใช้งานข้อมูลนั้นไม่มีเนื่องจากเป็นการนำข้อมูลเข้าไปยังระบบจัดการฐานข้อมูล และสุดท้ายเป็นการปิดการเชื่อมต่อ โดยคำสั่งที่เน้นตัวหนา $result = mysql_query("INSERT INTO students (cid, sid, name, login, age, gpa) VALUES ('3100904078132', 'B5075832', 'Somnamna', 'somnamna@math', 22, 2.00)"); เป็นคำสั่งหลักที่กระทำกับฐานข้อมูล สำหรับโค้ดในส่วนอื่นๆ เป็นคำสั่งสำหรับจัดการกับกลไกการใช้งานฐานข้อมูลบนเว็บ ตัวอย่างนี้เป็นตัวอย่างแสดงการดำเนินการคำสั่ง SQL INSERT อย่างง่ายเท่านั้น การเรียกใช้งาน insert.php จะเป็นการทำให้คำสั่งแทรกข้อมูลนี้ทำงาน จะเป็นเพียงการนำข้อมูลของ “Somnamna” เพิ่มลงในตารางนักศึกษาเท่านั้น ซึ่งโดยปกติเราจะใช้งานคำสั่งในลักษณะการ INSERT, UPDATE และ DELETE ร่วมกับฟอร์มซึ่งจะได้แสดงในตัวอย่างถัดไป

ตัวอย่าง Web Database: INSERT

แบบฟอร์มการเพิ่มข้อมูลนักศึกษาใหม่ แสดงได้ด้วยการเรียกใช้งานฟอร์ม add_student_form.php จากนั้นฟอร์มดังกล่าวส่งข้อมูลไปประมวลผลด้วยไฟล์ add_student.php แสดงตัวอย่างโค้ดดังนี้

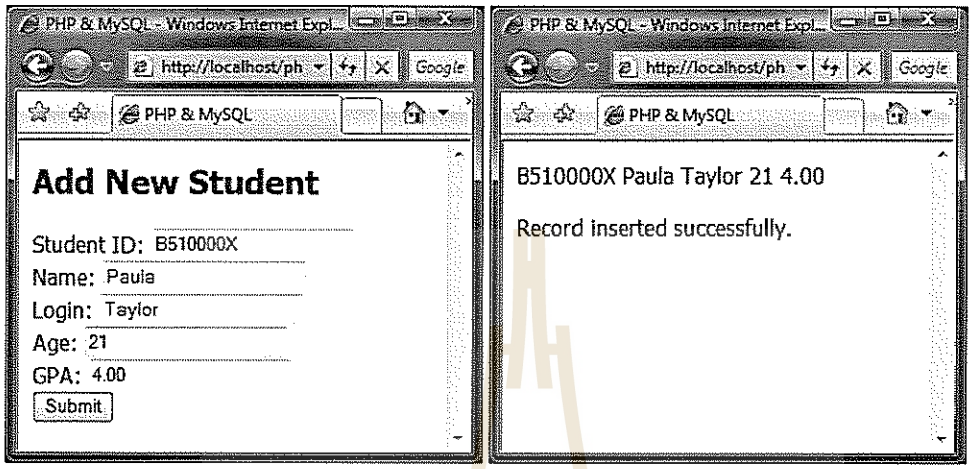
add_student_form.php

```
<html> <head><title>PHP & MySQL</title></head>
<body>
<form name="form1" method="post" action="add student.php" >
        Student ID: <input type="text" name="sid" /><br />
        Name: <input type="text" name="name" /><br />
        Login: <input type="text" name="login" /><br />
        Age: <input type="text" name="age" /><br />
        GPA: <input type="text" name="gpa" /><br />
        <input type="submit" value="Submit" />
</form>
</body>
</html>
```

add_student.php

```
<html> <head><title>PHP & MySQL</title></head>
<body>
<?
   // Connect to the database
   $con = mysql connect("localhost","root","password");
   if (!$con) die('Could not connect: ' . mysql error());
   mysql select db("university", $con);

   // Execute query
   // INSERT INTO students (sid, name, login, age, gpa)
   // VALUES ('B5000000', 'Name', 'Login', 18, 2.00);
   $result = mysql query("INSERT INTO students (sid, name, login, age, gpa)
      VALUES ('" .
      $ POST['sid'] . "', '" .
      $ POST['name'] . "', '" .
      $ POST['login'] . "', " .
      $_POST['age'] . ", " .
      $ POST['gpa'] . ")");
   if (!$result) die('Could not connect: ' . mysql error());

   // Display form data. (Not using data from the database)
   echo $ POST['sid'] . " " .
      $ POST['name'] . " " .
      $ POST['login'] . " " .
      $_POST['age'] . " " .
      $ POST['gpa'] . "<br />" .
      "<br />Insert record successfully.";

   // Close the connection
   mysql close($con);
?>
</body>
</html>
```

ตัวอย่างการเพิ่มข้อมูลนักศึกษาด้วยหน้าเว็บ add_student_form.php (ซ้าย) และผลลัพธ์ add_student.php ข้อมูลนักศึกษา "Paula" จะเข้าบรรจุอยู่ในฐานข้อมูล (ตัวอย่างนี้ไม่มีการเพิ่มข้อมูลหมายเลขบัตรประจำตัวประชาชน)



ตัวอย่าง Web Database: UPDATE

การปรับปรุง gpa โดยกำหนดรหัสนักศึกษาที่ต้องการแก้ไข gpa ตัวอย่างฟอร์มแสดงในด้านท้ายของบท

update_gpa_form.php

```
<html> <head><title>PHP & MySQL</title></head><body>
<form name="form1" method="post" action="update_gpa.php" >
   Student ID: <input type="text" name="sid" /><br />
   Enter GPA: <input type="text" name="gpa" /><br />
   <input type="submit" value="Submit" />
</form></body></html>
```

update_gpa.php

```
<html> <head><title>PHP & MySQL</title></head><body>
<?
   // Connect to the database
   $con = mysql_connect("localhost","root","password");
   if (!$con) die('Could not connect: ' . mysql_error());
   mysql_select_db("university", $con);

   // Execute query
   // UPDATE students SET gpa = 2.00 WHERE sid = 'B5000000'
   // VALUES ('B5000000', 'Name', 'Login', 18, 2.00);
   $sql_statement = "UPDATE students SET gpa = " .
      $_POST['gpa'] .
      " WHERE sid = '" .
      $_POST['sid'] . "'";
      echo $sql_statement;
   $result = mysql_query($sql_statement);
   if (!$result) die('Could not connect: ' . mysql_error());

   //
   // Display form data.
   echo ("Record updated successfully.");

   // Close the connection
   mysql_close($con);
?></body></html>
```

ตัวอย่าง Web Database: DELETE

ลบข้อมูลนักศึกษา โดยกำหนดรหัสนักศึกษาที่ต้องการลบ ตัวอย่างฟอร์มแสดงในด้านท้ายของบท

delete_student_form.php

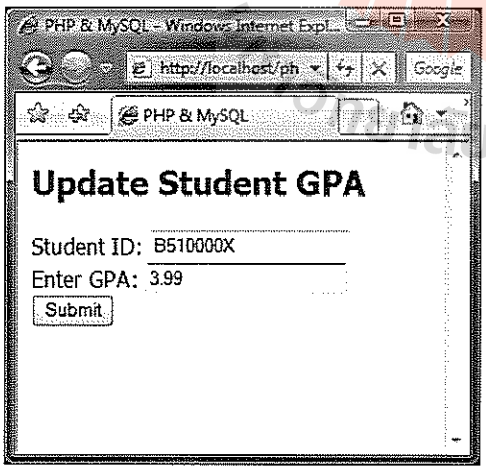
```
<html>
<body>
<form name="form1" method="post" action="delete_student.php" >
   <h2>Delete Student</h2>
   Enter Student ID: <input type="text" name="sid" /><br />
   <input type="submit" value="Submit" />
</form>
</body>
</html>
```

delete_student.php

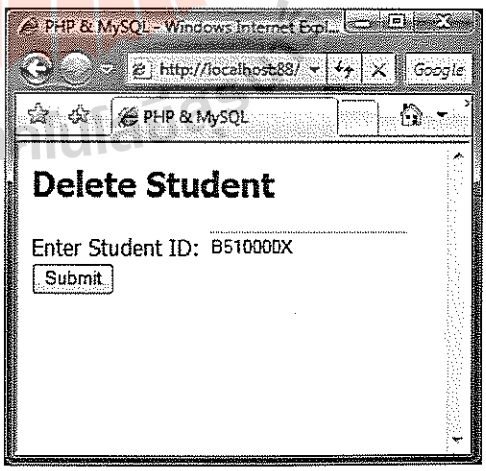
```
<html>
<body>
<?
   // Connect to the database
   $con = mysql_connect("localhost","root","password");
   if (!$con) die('Could not connect: ' . mysql error());
   mysql_select_db("university", $con);

   // Execute query
   // DELETE FROM students WHERE sid = 'B5000000'
   $sql_statement = "DELETE FROM students WHERE sid = '" .
      $ POST['sid'] . "'";
   $result = mysql_query($sql_statement);
   if (!$result) die('Could not connect: ' . mysql error());

   // Display result (No data used.)
   echo ("Record deleted successfully.");

   // Close the connection
   mysql_close($con);
?>
</body>
</html>
```

ตัวอย่างหน้าจอการแก้ไข gpa และลบข้อมูลนักศึกษา

update_gpa_form.php



delete_student.php



10.6 แบบฝึกหัดท้ายบท

ให้นักศึกษาฝึกปฏิบัติสร้างโปรแกรมประยุกต์บนเว็บ 7-Elephant ในภาคผนวก ให้ใช้งานได้

ภาคผนวก

ภาคผนวก ก

โครงงาน

"เช่า-ฉัน-สิ"

Rent-Me-Please

วัตถุประสงค์

เพื่อประยุกต์ความรู้ และพัฒนาทักษะด้านการออกแบบ พัฒนาและจัดการฐานข้อมูลสำหรับใช้งานจริงให้ลึกซึ้ง เพื่อเชื่อมโยงกับการนำไปใช้จริงในงานต่างๆ

คำสั่ง

ให้จับกลุ่มๆ ละ 5 คน เพื่อออกแบบและพัฒนาฐานข้อมูลสำหรับระบบสารสนเทศเพื่อการจัดการธุรกิจการให้เช่ารถยนต์ ระยะเวลาในการพัฒนา 3 เดือน ซึ่งมีความต้องการของผู้ใช้และข้อกำหนดดังนี้

"เช่า-ฉัน-สิ"

เช่า-ฉัน-สิ เป็นบริษัทให้บริการเช่ารถซึ่งเป็นธุรกิจที่กำลังเติบโตสูงเนื่องจากนโยบายส่งเสริมการท่องเที่ยว ทั้งของชาวไทยและต่างประเทศ เช่า-ฉัน-สิ ต้องการพัฒนาระบบสารสนเทศเพื่ออำนวยความสะดวกในการให้บริการเช่ารถ จึงมีความจำเป็นต้องออกแบบฐานข้อมูลโดยมีข้อกำหนดหรือความต้องการของระบบดังนี้

Data Requirements

เช่า-ฉัน-สิ เป็นบริษัทให้บริการเช่ารถยนต์โดยให้บริการและมีสำนักงานอยู่ทั่วประเทศ โดยสำนักงานมี 2 ประเภท ได้แก่สำนักงานทำการที่รับบริการจองรถและเปลี่ยนแปลงรายละเอียดการจองเท่านั้น และสำนักงานที่ทั้งรับจองรถและให้ผู้เช่ามารับและคืนรถ ข้อมูลของสำนักงานได้แก่หมายเลขสำนักงาน ชื่อของสำนักงาน ที่ตั้งของสำนักงาน ประเภทของสำนักงาน จำนวนรถที่ให้บริการ ฯลฯ โดยสถานที่ตั้งสำนักงานส่วนใหญ่จะเป็นในตัวเมือง และท่าอากาศยาน

พนักงานของบริษัทจะเป็นผู้รับจองการเช่าและรายละเอียดการเช่าทางโทรศัพท์ ข้อมูลของพนักงานประกอบด้วยหมายเลขพนักงาน ชื่อ-นามสกุล ที่อยู่ หมายเลขโทรศัพท์ และเงินเดือน การจองรถยนต์ การรับกุญแจ และการคืนรถยนต์ทุกครั้งจะต้องมีการจัดเก็บข้อมูลพนักงานผู้ให้บริการด้วย

ลูกค้าที่ประสงค์จะเช่ารถ จะต้องถูกบันทึกประวัติในการจองครั้งแรก และข้อมูลจะถูกใช้ในครั้งต่อไป ซึ่งข้อมูลของลูกค้าประกอบด้วยหมายเลขประจำตัวประชาชนหรือหมายเลขหนังสือเดินทาง ชื่อ-นามสกุล ที่อยู่ หมายเลขโทรศัพท์ ฯลฯ โดยลูกค้าสามารถกรอกข้อมูลของตัวเองผ่านทางอินเตอร์เน็ตได้

การบริการเช่ารถ เมื่อลูกค้าประสงค์จะเช่ารถ ลูกค้าจะทำการโทรศัพท์หรือเลือกจากทางอินเตอร์เน็ต เพื่อระบุประเภทของรถ ยี่ห้อหรือผู้ผลิต สี ขนาด วันที่และเวลารับรถ วันที่และเวลาส่งคืนรถ ชนิดของการประกันภัยรถยนต์

สำนักงานที่จะรับและคืนรถ หลังจากนั้นเมื่อถึงวันที่รับรถ ผู้เช่าจะต้องมารับรถด้วยตนเองพร้อมกับบัตรประจำตัวที่มีรูปถ่ายที่สามารถระบุได้ว่าเป็นผู้เช่าจริง ผู้เช่าจะต้องจ่ายค่าเช่ารถยนต์ซึ่งรวมทั้งค่าประกันภัยต่างๆ ก่อนนำรถไปใช้ หลังจากใช้งานเสร็จผู้เช่าจะต้องเติมน้ำมันให้เต็มถังก่อนส่งคืนรถยนต์ หรือลูกค้าที่ประสงค์จะคืนรถยนต์โดยมีน้ำมันเท่าใดก็ได้ ลูกค้าสามารถซื้อน้ำมันกับบริษัทล่วงหน้า คิดราคาน้ำมันลิตรละ 25 บาท ตามความจุของถังน้ำมันรถแต่ละคัน โดยการจ่ายไว้ก่อน หลังจากนั้นจะคืนรถยนต์โดยมีน้ำมันเท่าใดก็ได้ นอกจากนี้ หากมีการเสียหายของรถยนต์ ลูกค้าไม่จำเป็นต้องรับผิดชอบเนื่องจากลูกค้าจะต้องซื้อประกันภัยความเสียหายของรถยนต์ บุคคลที่ 3 และประกันการสูญหายเป็นอย่างต่ำ โดยประกันภัยความเสียหายและสูญหายจะคิดตามราคาของรถ ลูกค้าสามารถคืนรถที่สำนักงานใดก็ได้ โดยระบุสถานที่ไว้ก่อน ลูกค้าที่คืนรถยนต์ช้าจะถูกปรับชั่วโมงละ 10% ของค่าเช่าต่อวัน ลูกค้าที่คืนรถช้าเกิน 6 ชั่วโมง จะถูกคิดค่าเช่าเพิ่ม 1 วัน ลูกค้าจะต้องชำระค่าเช่าทั้งหมดก่อนนำรถไปใช้ รวมกับค่ามัดจำ 10,000 บาทสำหรับค่ากุญแจ ฯลฯ

รถยนต์ที่ให้บริการมีหลากหลายประเภทได้แก่ รถเก๋งขนาดต่ำกว่าขนาดเล็ก (subcompact) รถเก๋งขนาดเล็ก (compact) รถเก๋งขนาดกลาง (mid-size) รถเก๋งขนาดใหญ่ (full-size) รถ SUV ขนาดต่าง รถกระบะ และรถประเภทพิเศษเช่นรถตู้ รถแวน รถเปิดประทุน นอกจากประเภทของรถยนต์ดังกล่าวรถยนต์แต่ละคันจะแตกต่างกันที่ผู้ผลิต ปีที่ผลิต ปีที่บริษัทซื้อมาให้บริการ สี จำนวนที่นั่งประเภทของการเกียร์ รายละเอียดกรรมธรรม์ประกันภัยรายปี

รถยนต์แต่ละคันจะมีตารางการบำรุงรักษา โดยพนักงานที่เป็นช่างจะทำการตรวจสอบรถยนต์ตามตารางที่ตั้งไว้ ในกรณีที่มีอุปกรณ์ต้องซ่อม หรือเกิดอุบัติเหตุ จะต้องมีการจดบันทึกประวัติการซ่อมรถกับอู่ซ่อมรถในแต่ละครั้ง รายละเอียดการซ่อม รวมถึงค่าใช้จ่ายที่เกิดขึ้น

ประกันภัยและประกับอุบัติเหตุรถยนต์เป็นชุดประกันภัยที่บริษัทใช้บริการจากบริษัท AIG และนำเสนอให้ลูกค้าเลือกซื้อ ซึ่งกรมธรรม์แตกต่างกันที่วงเงินความเสียหายที่รับผิดชอบต่อบุคคลที่ 3

ในกรณีที่ผู้เช่ารถต้องการรับรถที่อาจไม่มีอยู่ให้บริการในสถานที่ๆ ระบุเพื่อรับรถ ระบบจะสามารถระบุรถที่ผู้ใช้ต้องการที่อยู่ในสำนักงานที่ใกล้เคียงที่สุด ทั้งนี้สำนักงานที่มีรถอยู่ จะต้องให้พนักงานขับรถขับรถดังกล่าวมาส่งยังสำนักงานที่ลูกค้าจะรับรถยนต์ก่อนเวลาที่ลูกค้าจะมารับรถ การขับรถส่งข้ามสำนักงานแต่ละครั้งจะต้องทำการจดบันทึกวันที่ ค่าใช้จ่าย พนักงานขับรถ ฯลฯ

บริษัทจะทำการตลาดโดยการเสนอโปรโมชั่นพิเศษ ในลักษณะการลดราคา โดยข้อมูลการลดราคาที่จัดเก็บได้แก่ชื่อของโปรโมชั่น ช่วงเวลาที่ลดราคา ชนิดของรถยนต์ที่ทำการลดราคา เงื่อนไขของการลดราคาได้แก่จำนวนวันที่เช่า เช่น 3 วัน 1 อาทิตย์ โดยการลดราคาสามารถระบุได้เป็น 2 รูปแบบได้แก่ร้อยละของราคาที่ลดหรือราคาพิเศษที่คำนวณไว้แล้ว

Transactional Requirements

1. พนักงานสามารถเพิ่มเติม แก้ไขข้อมูลพนักงานได้
2. พนักงานหรือลูกค้าสามารถเพิ่มเติม แก้ไขข้อมูลของลูกค้าได้
3. พนักงานสามารถเพิ่มเติม แก้ไขข้อมูลรถยนต์ที่ให้บริการเช่าได้
4. พนักงานหรือระบบสามารถบันทึกรายการการใช้บริการของลูกค้าแต่ละครั้งได้ ซึ่งครอบคลุมการจอง การรับรถ และการคืนรถ
5. พนักงานสามารถบันทึก แก้ไขข้อมูลของกรมธรรม์ประกับอุบัติเหตุได้
6. พนักงานสามารถบันทึก ตารางการบำรุงรักษารถยนต์แต่ละคันได้
7. พนักงานสามารถบันทึกการบำรุงรักษาและการซ่อมรถยนต์แต่ละคันได้
8. พนักงานสามารถบันทึกการส่งมอบรถข้ามสำนักงานได้
9. พนักงานสามารถเพิ่มเติม แก้ไขโปรโมชั่นได้
10. ออกรายงานต่างๆ ที่เป็นประโยชน์ในการทำการตลาดและจัดการธุรกิจที่เหมาะสมได้

ภาคผนวก ข

กรณีศึกษาร้านมือถือออนไลน์ 7-Elephant
การพัฒนาฐานข้อมูลบนเว็บด้วย PHP และ MySQL

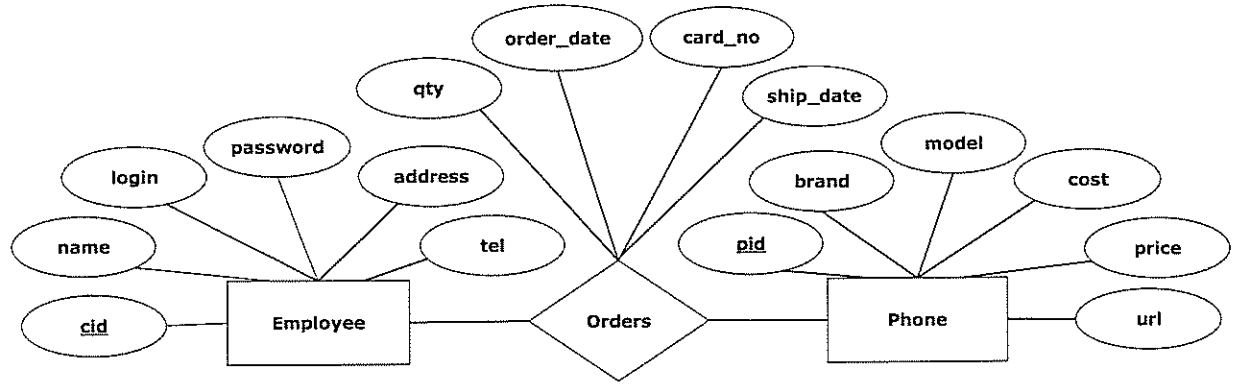
ร้านมือถือออนไลน์ 7-Elephant

ตัวอย่างร้าน 7-Elephant นี้แสดงการออกแบบ ERD สำหรับร้านมือถือออนไลน์ ซึ่งแสดงเฉพาะ ERD ที่ได้มาจากความต้องการของผู้ใช้ ในภาคผนวก เราได้ใช้ตัวอย่างร้าน 7-Elephant นี้เป็นกรณีศึกษาในการออกพัฒนาระบบฐานข้อมูล โดยแสดงขั้นตอนที่สำคัญ ได้แก่การออกแบบ ERD และพัฒนาระบบด้วย PHP, MySQL และคำสั่ง SQL

เจ้าของร้าน 7-Elephant เล่าถึงขั้นตอนการทำงานของระบบที่ต้องการได้โดยละเอียด (ซึ่งโดยทั่วไปแล้วการระบุความต้องการของผู้ใช้นั้นต้องอาศัยทักษะของผู้รวบรวมและวิเคราะห์ความต้องการของผู้ใช้ในการรวบรวมข้อมูลและซักถามความต้องการจากผู้ใช้ซึ่งใช้เวลาพอสมควร) ดังนี้

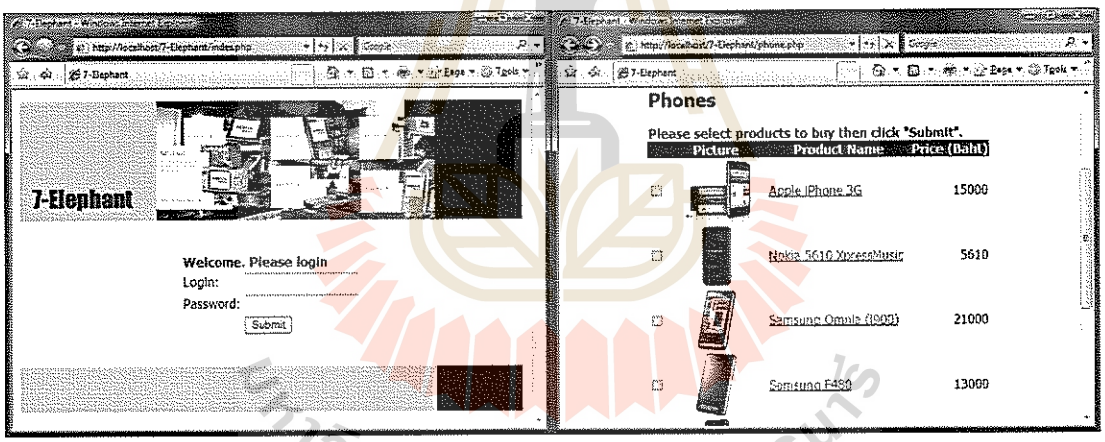
"7-Elephant ต้องการพัฒนาระบบร้านค้าออนไลน์เพื่อให้ลูกค้าแสดงแคตตาล็อกมือถือและสั่งซื้อมือถือบนอินเตอร์เน็ตได้ ปัจจุบันลูกค้าสั่งซื้อสินค้าทางโทรศัพท์ ลูกค้าส่วนใหญ่เป็นลูกค้าประจำของร้าน เป็นลูกค้าระดับบริษัทขนาดกลางและใหญ่ ทางร้านจัดเก็บข้อมูลของลูกค้าไว้ได้แก่ ชื่อลูกค้า (name) ที่อยู่ (address) และหมายเลขโทรศัพท์ (tel) จะทำการสั่งซื้อสินค้าโดยบอกยี่ห้อและรุ่นของโทรศัพท์ ตลอดจนจำนวนที่ต้องการสั่งซื้อ โดยส่วนใหญ่เอาไปให้พนักงานในบริษัทใช้ในกรณีออกติดต่อกับลูกค้านอกสำนักงาน ลูกค้ามักชำระค่าสินค้าด้วยบัตรเครดิต จากนั้นร้าน 7-Elephant จะทำการเตรียมสินค้าให้ครบตามจำนวนที่ผู้ใช้สั่งซื้อเพื่อจัดส่งสินค้า ในกรณีที่สินค้ามีไม่พอกับยอดที่สั่ง ทางร้านจะสั่งสินค้ามาเพิ่มจนกว่าจะได้ครบตามกำหนดแล้วค่อยจัดส่งสินค้าไปในคราวเดียวกันสำหรับการสั่งซื้อแต่ละครั้ง ในแคตตาล็อกจะมีข้อมูลของมือถือทั้งหมดที่ร้านจำหน่าย โดยข้อมูลของมือถือ (Phone) ได้แก่ ยี่ห้อ (brand) รุ่น (model) ราคาทุน (cost) ราคาขาย (price) และ URL ของเว็บไซต์อย่างเป็นทางการของมือถือแต่ละรุ่น (url) เพื่อให้ลูกค้าสามารถดูรายละเอียดข้อกำหนดและความสามารถของมือถือแต่ละรุ่น เพื่อความสะดวกในการจัดการสินค้า ร้านค้ายังกำหนดรหัสประจำมือถือแต่ละรุ่น (pid) เนื่องจากรุ่นของมือถือแต่ละยี่ห้ออาจตั้งชื่อมาซ้ำกันได้ ลูกค้าที่จะสั่งซื้อสินค้าทางอินเตอร์เน็ตต้องมีชื่อผู้ใช้ (login) และรหัสผ่าน (password) ที่สัมพันธ์กับลูกค้า ลูกค้าที่ยังไม่มีข้อมูลกับทางร้านจะต้องโทรศัพท์มาเพื่อบันทึกข้อมูลลูกค้าและสร้างบัญชีผู้ใช้ก่อน ทางร้านจะกำหนดให้ลูกค้าแต่ละคนมีหมายเลขประจำลูกค้า (cid) สำหรับจัดเก็บข้อมูลการสั่งซื้อ หลังจากที่ลูกค้าเข้าใช้ระบบ แสดงสินค้าจากแคตตาล็อกและสั่งซื้อสินค้า"

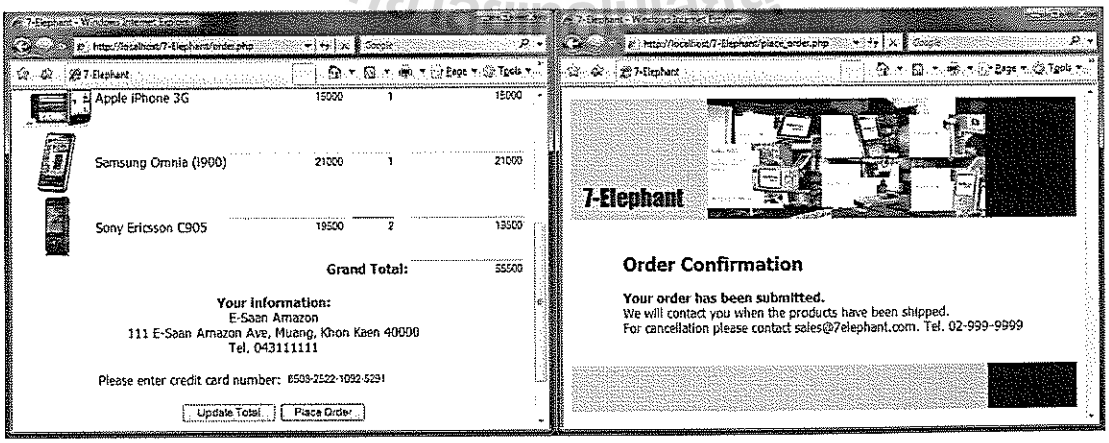
ERD เบื้องต้นของระบบร้านค้าออนไลน์สามารถวิเคราะห์และออกแบบได้ดังนี้

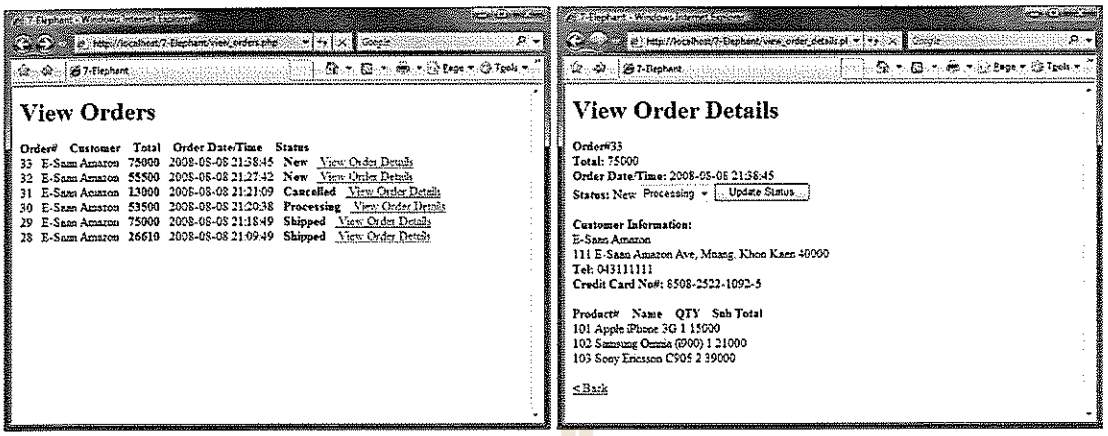


แม้ว่าระบบจะรองรับการเข้าใช้เว็บไซต์ ตลอดจนสั่งซื้อและจัดส่งมือถือได้ อย่างไรก็ตามหากพิจารณาเงื่อนไขที่เจ้าของร้านยังไม่ได้กล่าวถึง เช่น การสั่งซื้อมือถือรุ่นเดิมอีกครั้งในวันหลังจะไม่สามารถกระทำได้ ซึ่งจะต้องสอบถามถึงประเด็นดังกล่าวกับผู้ใช้งานระบบอีกครั้งหนึ่งเพื่อปรับการออกแบบ ERD

** employee ผิด







View Orders

Order#	Customer	Total	Order Date/Time	Status	
33	E-Sann Amaron	75000	2008-05-08 21:38:45	New	View Order Details
32	E-Sann Amaron	55500	2008-05-08 21:27:42	New	View Order Details
31	E-Sann Amaron	13000	2008-05-08 21:21:09	Cancelled	View Order Details
30	E-Sann Amaron	53500	2008-05-08 21:20:38	Processing	View Order Details
29	E-Sann Amaron	75000	2008-05-08 21:18:49	Shipped	View Order Details
28	E-Sann Amaron	26610	2008-05-08 21:09:49	Shipped	View Order Details

View Order Details

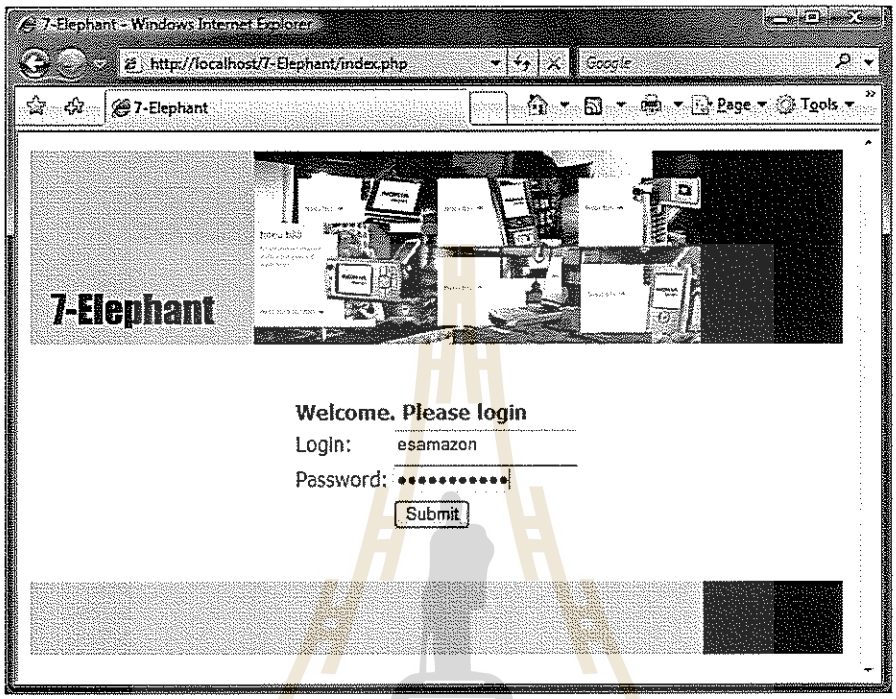
Order#33
Total: 75000
Order Date/Time: 2008-05-08 21:38:45
Status: New Processing Update Status

Customer Information:
E-Sann Amaron
111 E-Sann Amaron Ave, Muang, Khon Kaen 40000
Tel: 043111111
Credit Card No#: 6508-2522-1092-5

Product# Name QTY Sub Total
101 Apple iPhone 3G 1 15000
102 Samsung Omnia i900) 1 21000
103 Sony Ericsson C905 2 39000

<< Back

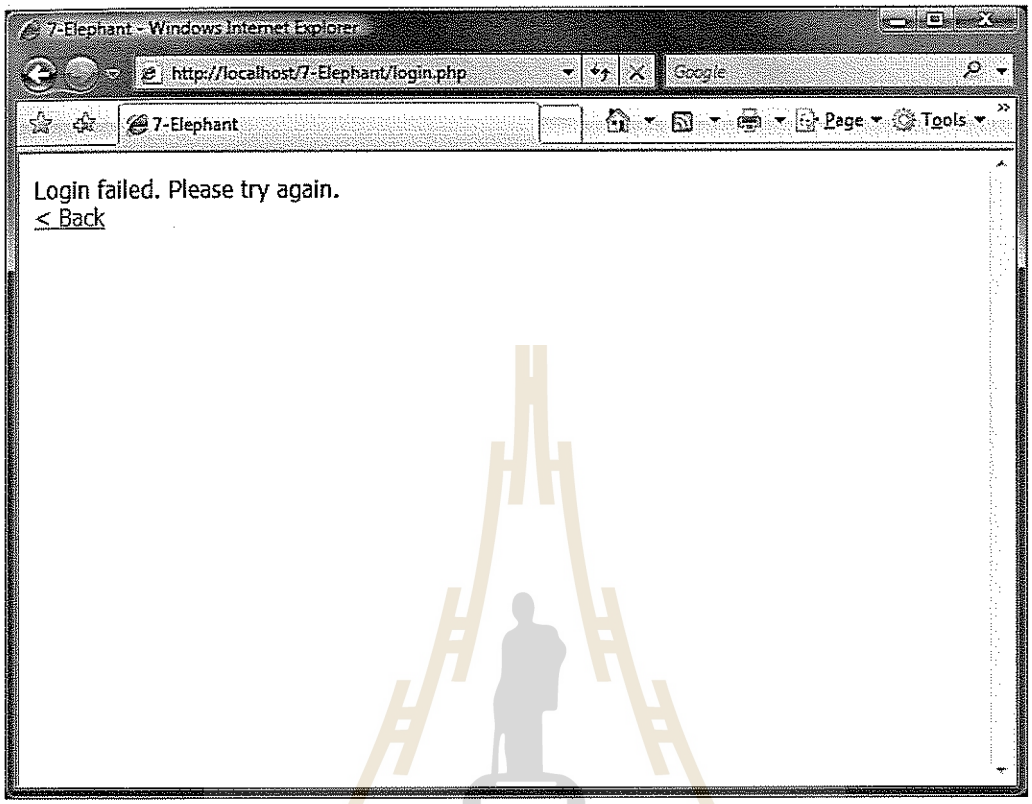
1. **หน้าเริ่มต้น**



index.php

```
1  <html>
2  <head><title>7-Elephant</title></head>
3  <body style="text-align:center">
4  <form name="login" method="post" action="login.php">
5     <img src="images/phonebanner.jpg">
6     <br /><br />
7
8     <table border="0">
9        <tr>
10         <td colspan="2"><b>Welcome. Please login</b></td>
11       </tr>
12       <tr>
13         <td>Login:</td>
14         <td><input type="text" name="login"></td>
15       </tr>
16       <tr>
17         <td>Password:</td>
18         <td><input type="password" name="password"></td>
19       </tr>
20       <tr>
21         <td> </td>
22         <td><input type="submit" value="Submit"></td>
23       </tr>
24    </table>
25    <br /><br />
26    <img src="images/phonefooter.jpg">
27 </form>
28 </body>
29 </html>
```

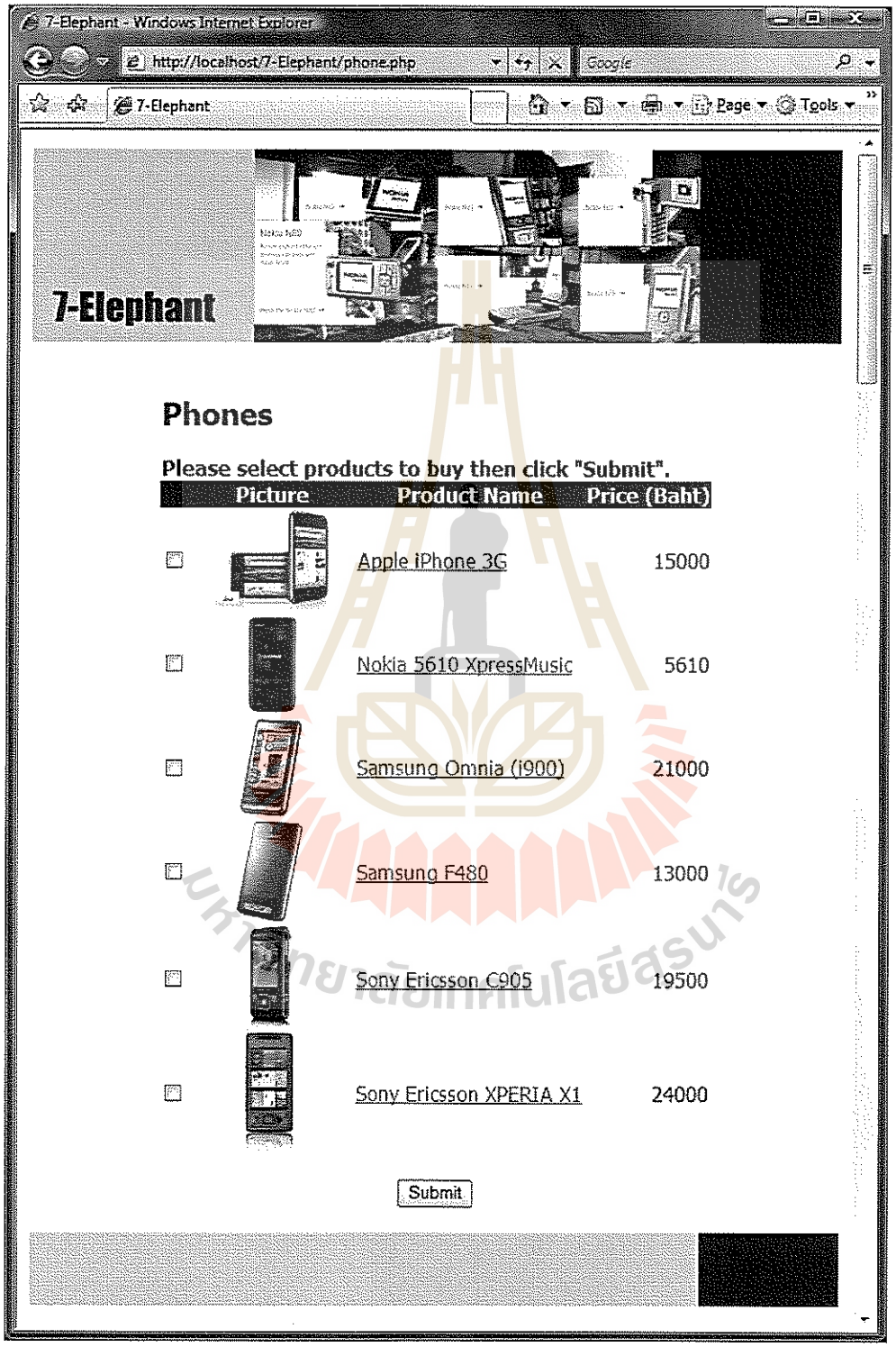
2. หน้าตรวจสอบการล็อกอินเข้าสู่ระบบ

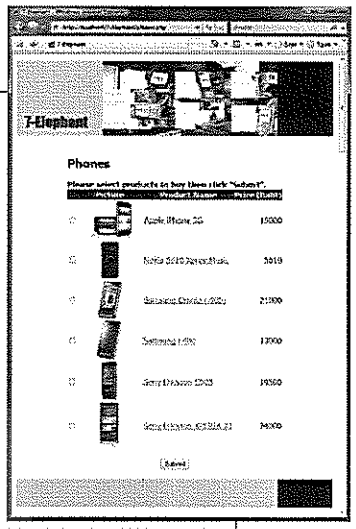


login.php

```
1  <?
2     @session start();
3     $con = mysql connect("localhost","root","password");
4     if (!$con) die('Could not connect: ' . mysql error());
5
6     mysql select db("seven elephant", $con);
7     $sql statement = "SELECT * FROM customer WHERE login = '" .
8        $_POST['login'] . "' AND password = '" . $_POST['password'] . "'";
9     $result = mysql query($sql statement);
10    if (!$result) die('Error query: ' . mysql error());
11    //echo $sql statement;
12
13    while($row = mysql fetch array($result)) {
14       $customer id = $row['custoemr id'];
15       mysql close($con);
16       $ SESSION['customer id'] = $customer id;
17       //echo $customer_id;
18       header("Location: phone.php");
19    }
20
21    mysql close($con);
22    echo "Login failed. Please try again.<br /><a href=\"index.php\">
23       &lt; Back</a>";
24 ?>
```

3. <u>หน้าแสดงและเลือกซื้อสินค้า</u>

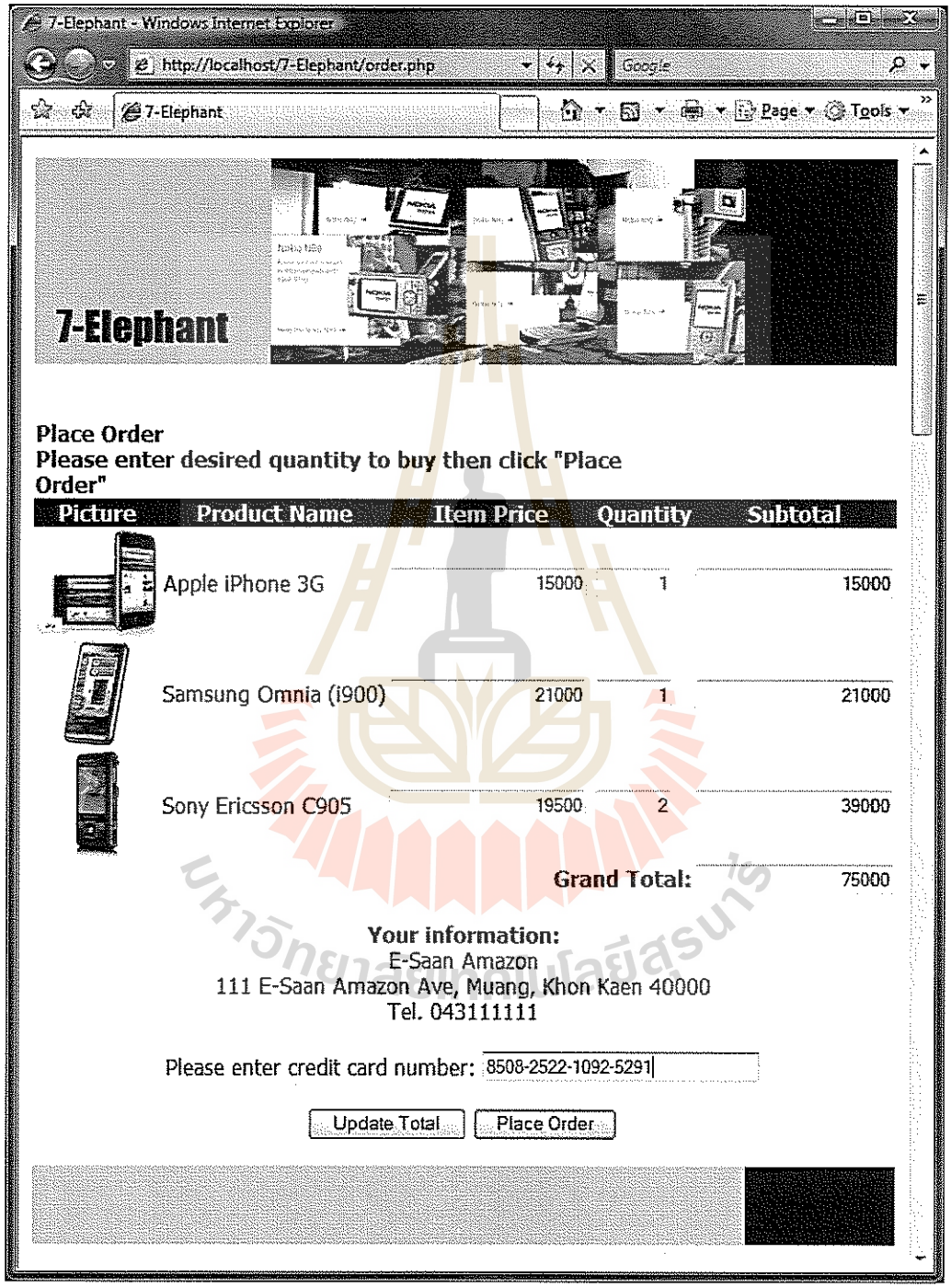




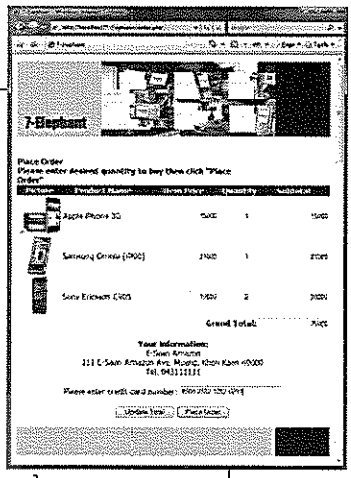
phones.php

```
<? session_start() ?>
<html>
<head>
<title>7-Elephant</title>
<head>

<body style="text-align:center">
<form id="product" method="POST" action="order.php">
   <img src="images/phonebanner.jpg">
   <br /><br />

   <table border="0" cellspacing="0">
      <tr>
         <td colspan="4"><b><h2>Phones</h2>Please select
         products to buy then click "Submit".</b></td>
      </tr>
      <tr>
         <th style="color:white;background:black"> </th>
         <th style="color:white;background:black" width="130">Picture</th>
         <th style="color:white;background:black">Product Name</th>
         <th style="color:white;background:black">Price (Baht)</th>
      </tr>

      <?
      $_SESSION['customer_id'] = "101";
      $con = mysql_connect("localhost","root","password");
      if (!$con) die('Could not connect: ' . mysql_error());

      mysql_select_db("seven_elephant", $con);
      $result = mysql_query("SELECT * FROM phone ORDER BY brand");
      if (!$result) die('Error query: ' . mysql_error());

      while($row = mysql_fetch_array($result)) {
         echo "<tr>";
         echo "<td><input type=\"checkbox\" name=\"buy[]\" value=\"" .
            $row['product_id'] . "\" /></td>";
         echo "<td align=\"center\"><img src=\"images/" . $row['product_id'] .
            ".jpg\" /></td>";
         echo "<td><a href=\"" .$row['url'] . "\" target=\" blank\">" .
            $row['brand'] . " " . $row['model'] . "</a></td>";
         echo "<td align=\"right\">" . $row['price'] . "</td>";
         echo "</tr>";
        }
      mysql_close($con);
      ?>
   </table>
   <br />
   <input type="submit" value="Submit" />
   <br /><br />
   <img src="images/phonefooter.jpg">
</form>
</body>
</html>
```

4. <u>หน้ากำหนดรายละเอียดการสั่งซื้อสินค้า</u>

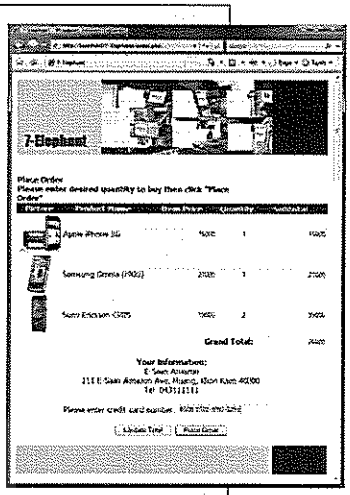


order.php

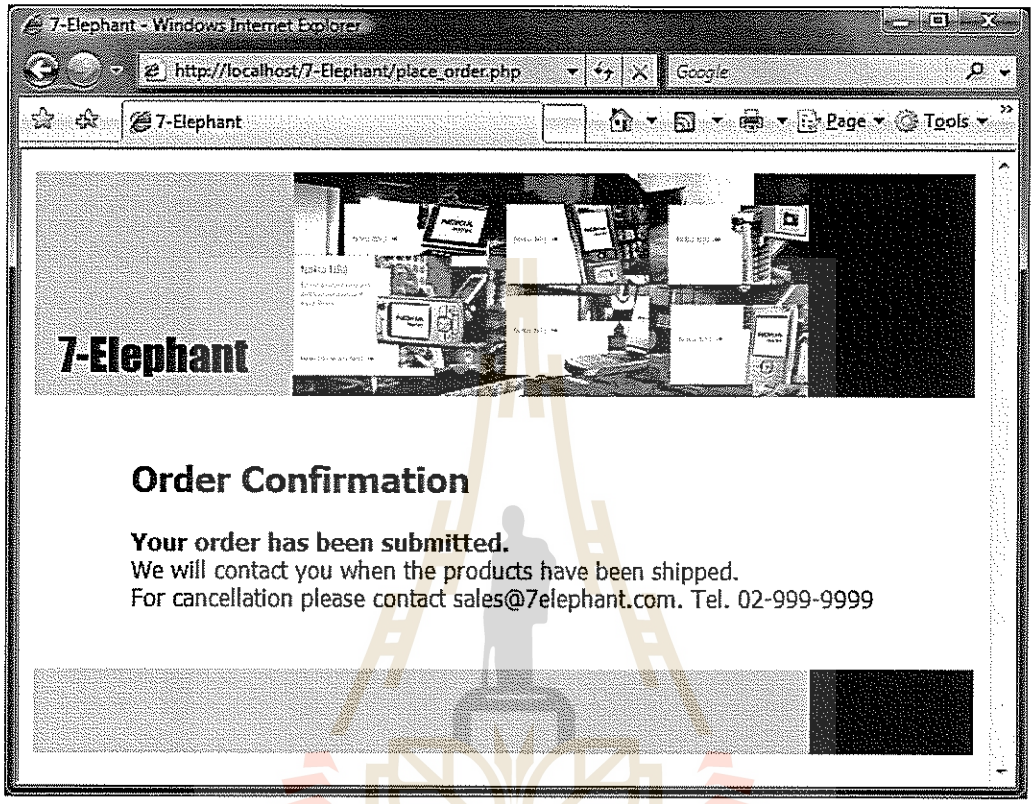



```
<? session_start() ?>
<html>
<head>
<script type="text/javascript">
function updateTotal() {
   var i = 0;
   var price = 0;
   var amount = 0;
   var name = "";
   var sub_total = 0;
   var grand_total = 0;

   for(i = 0;
      i < document.forms["product"].elements[
         "order_amount"].value; i++) {
      price = document.forms["product"].elements["price" + i].value;
      amount = document.forms["product"].elements["amount" + i].value;
      sub_total = price * amount;
      document.forms["product"].elements["sub_total" + i].value = sub_total;
      grand_total += sub_total;
   }
   document.forms["product"].elements["grand_total"].value = grand_total;
}

function jump() {
   document.forms["product"].elements["submit"].focus();
}
</script>
<title>7-Elephant</title>
<head>

<body style="text-align:center">
<form id="product" method="POST" action="place_order.php">
   <img src="images/phonebanner.jpg">
   <br /><br />

   <table border="0" cellspacing="0">
      <tr>
         <td colspan="4"><b>Place Order<br />
         Please enter desired quantity to buy then click
         "Place Order"</b></td>
      </tr>
      <tr>
         <th style="color:white;background:black">Picture</th>
         <th style="color:white;background:black">Product Name</th>
         <th style="color:white;background:black">Item Price</th>
         <th style="color:white;background:black">Quantity</th>
         <th style="color:white;background:black">Subtotal</th>
      </tr>

      <?
      $product_list = implode(",", $_POST["buy"]);

      $con = mysql_connect("localhost","root","password");
      if (!$con) die('Could not connect: ' . mysql_error());

      mysql_select_db("seven_elephant", $con);
      $result = mysql_query("SELECT * FROM phone WHERE product_id IN
         (" . $product_list . ")");
      if (!$result) die('Error query: ' . mysql_error());

      $i = 0;
```

```
63          $grand total = 0;
64          while($row = mysql fetch array($result)) {
65             echo "<tr>
66                <input type=\"hidden\" name=\"product id" . $i .
67                "\" value=\"" . $row['product_id'] . "\" />";
68             echo "<td align=\"center\"><img src=\"images/" .
69                $row['product id'] . ".jpg\" /></td>";
70             echo "<td>" . $row['brand'] . " " . $row['model'] .
71                "</td>";
72             echo "<td><input type=\"text\"
73                style=\"text-align:right\" onfocus=\"jump()\"
74                name=\"price" . $i . "\" value=\"" .
75                $row['price'] . "\" /></td>";
76             echo "<td><input type=\"text\"
77                style=\"text-align:right\" size=\"5\"
78                name=\"amount" . $i . "\" value=\"1\" /></td>";
79             echo "<td><input type=\"text\"
80                style=\"text-align:right\" onfocus=\"jump()\"
81                name=\"sub total" . $i . "\" value=\"" .
82                $row['price'] . "\" /></td>";
83             echo "</tr>";
84
85             $i++;
86             $grand total += $row['price'];
87          }
88
89          echo "<input type=\"hidden\" name=\"order amount\" value=\"" .
90             $i . "\" />";
91
92          ?>
93          <tr>
94             <td colspan="4" align="right"><b>Grand Total:</b></td>
95             <td><input type="text" style="text-align:right"
96                onfocus="jump()" name="grand_total" value="
97                <? echo $grand total; ?>" /></td>
98          </tr>
99       </table>
100      <br />
101      <b>Your information:</b><br />
102      <?
103         $result = mysql query("SELECT * FROM customer WHERE customer id = " .
104            $ SESSION['customer id']);
105         if (!$con) die('Could not connect: ' . mysql error());
106
107         $row = mysql fetch array($result);
108         echo $row['name'] . "<br />" . $row['address'] .
109            "<br />Tel." . $row['phone no'];
110         mysql close($con);
111       ?>
112      <br />
113      <br />
114      Please enter credit card number:
115      <input type="text" name="card no" size="30" />
116      <br /><br />
117      <input type="button" name="update total"
118         value="Update Total" onClick="updateTotal()" />
119      <input type="submit" name="submit" value="Place Order" />
120      <br /><br />
121      <img src="images/phonefooter.jpg">
122   </form>
123   </body>
124   </html>
```



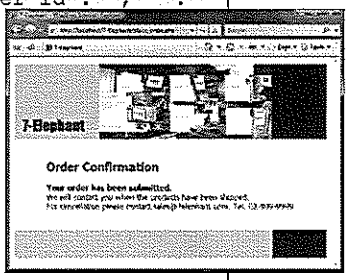
5. <u>หน้าดำเนินการสั่งซื้อสินค้าและยืนยันการสั่งซื้อ</u>



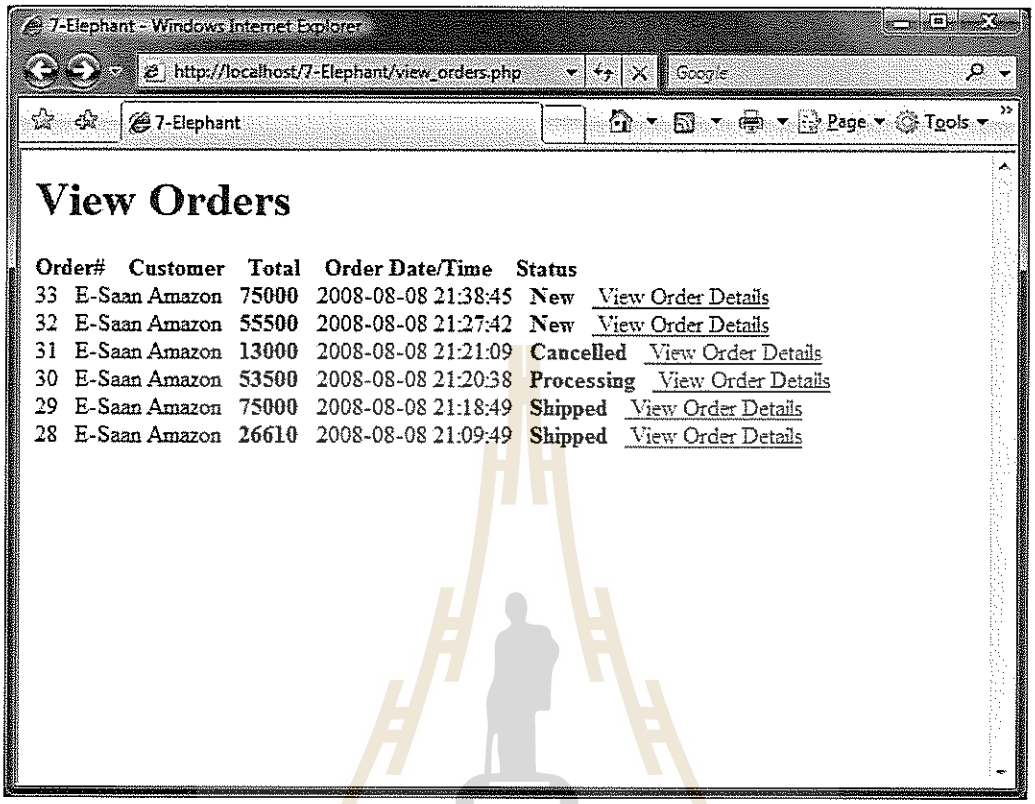
place_order.php

```
<?
   session start();
   $con = mysql connect("localhost","root","password");
   if (!$con) die('Could not connect: ' . mysql error());

   mysql_select_db("seven_elephant", $con);
   $sql statement = "INSERT INTO orders (order id, customer id, grand total,
      card no, order date, order status) VALUES (null, "
      $ SESSION["customer id"] ."," . $ POST["grand total"] ."," '" .
      $ POST["card no"] . "', '" . date("Y-m-d H:i:s") . "', 'New')";
   $result = mysql query($sql statement);
   if (!$result) die('Query error: ' . mysql error());

   $sql statement = "SELECT MAX(order id) AS order id FROM orders WHERE
      customer id = " . $ SESSION["customer id"];
   $result = mysql query($sql statement);
   if (!$result) die('Query error: ' . mysql error());

   $row = mysql fetch array($result);
   $order id = $row['order id'];
   $order amount = $ POST["order amount"];
   $i =0;
   for($i = 0; $i < $order amount; $i++) {
     $product id = "product id" . $i;
     $amount = "amount" . $i;
     $sub total = "sub_total" . $i;
```

```
      $sql statement = "INSERT INTO order details (order details id, order id,
         product id, amount, sub total) VALUES (null, " . $order id . ", " .
         $ POST[$product id] . ", " . $ POST[$amount] . ", " .
30       $ POST[$sub total] . ")";
31    $result = mysql_query($sql_statement);
32    if (!$result) die('Query error: ' . mysql error());
33  }
34  mysql close($con);
35 ?>
36
37 <html>
38 <head>
39 <title>7-Elephant</title>
40 </head>
41 <body style="text-align:center">
42   <img src="images/phonebanner.jpg">
43   <br /><br />
44
45   <table border="0" cellspacing="0">
46     <tr>
47       <td colspan="4"><b><h2>Order Confirmation</h2>
48       Your order has been submitted.</b><br />
49       We will contact you when the products have been shipped.<br />
50       For cancellation please contact sales@7elephant.com.
51       Tel. 02-999-9999</td>
52     </tr>
53   </table>
54   <br /><br />
55   <img src="images/phonefooter.jpg">
56 </body>
57 </html>
```



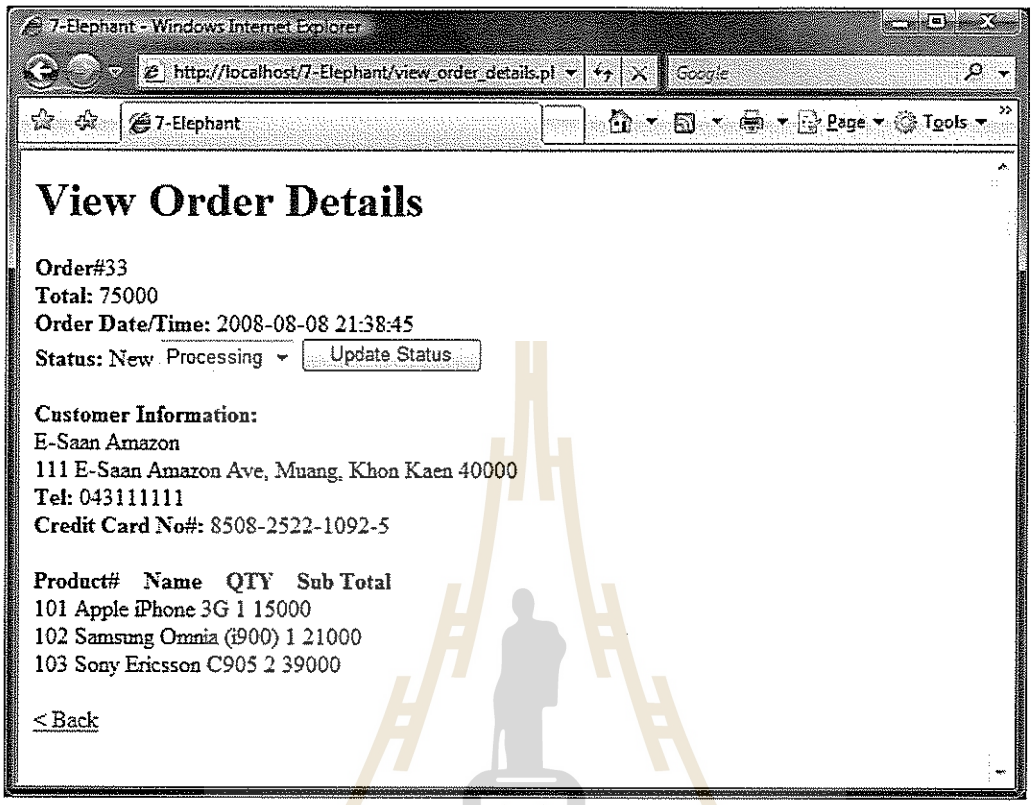
6. หน้าแสดงรายการสั่งซื้อ



view_orders.php

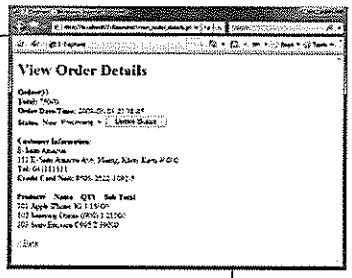
```
<html>
<head><title>7-Elephant</title><head>
<body>
<h1>View Orders</h1><b>
Order#     Customer   
Total     Order Date/Time   
Status     </b><br />
<?
   $con = mysql connect("localhost","root","password");
   if (!$con) die('Cannot connect: ' . mysql error());

   mysql select db("seven elephant", $con);
   $result = mysql query("SELECT order id, name, grand total, order date,
      order status FROM orders, customer WHERE orders.customer id =
      customer.customer id");
   if (!$con) die('Could not connect: ' . mysql_error());

   while($row = mysql fetch array($result)) {
     echo $row['order id'] . "   " .
        $row['name'] . "   <b>" .
        $row['grand total'] . "</b>   " .
        $row['order date'] . "   <b>" .
        $row['order status'] . "</b>   ";
     echo "<a href=\"view order details.php?order id=" . $row['order id'] .
        "\">View Order Details</a><br />";
     }
   mysql close($con);
?>
</body></html>
```

7. หน้าแสดงรายละเอียดการสั่งซื้อ



view_order_details.php

```
1  <html>
2  <head><title>7-Elephant</title><head>
3  <body>
4  <h1>View Order Details</h1>
5  <form name="form1" method="post" action="update status.php">
6  <?
7     $con = mysql connect("localhost","root","password");
8     if (!$con) die('Could not connect: ' . mysql_error());
9
10    mysql select db("seven elephant", $con);
11    $result = mysql query("SELECT * FROM orders, customer WHERE
12       orders.customer id = customer.customer id AND order id = " .
13       $ GET['order id']);
14    if (!$con) die('Could not connect: ' . mysql error());
15
16    $row = mysql_fetch_array($result);
17     echo "<b>Order#</b>" . $row['order id'] .
18       "<br /><b>Total: </b>" . $row['grand total'] .
19       "<br /><b>Order Date/Time: </b>" . $row['order date'] .
20       "<br /><b>Status: </b>" . $row['order status'] . " ";
21 ?>
22 <select name="status">
23    <option value="" />
24    <option value="Processing" />Processing
25    <option value="Shipped" />Shipped
26 </select>
27 <input type="submit" value="Update Status">
```




```
<?
   echo "<br /><br /><b>Customer Information:</b><br />" .
      $row['name'] . "<br />" .
      $row['address'] . " " . "<br /><b>Tel: </b>" .
      $row['phone no'] . "<br /><b>Credit Card No#: </b>" .
      $row['card_no'];
?>
<b><br /><br />
Product#      Name   
QTY      Sub Total   
</b><br />
<?
   $result = mysql query("SELECT * FROM order details, phone
      WHERE order details.product id = phone.product id AND order id = " .
      $_GET['order_id']);
   if (!$result) die('Could not connect: ' . mysql error());


   while($row = mysql fetch array($result)) {
      echo $row['product id'] . " " . $row['brand'] . " " . $row['model'] .
      " " . $row['amount'] . " " . $row['sub total'] . " <br />" ;
   }

   mysql close($con);
?>
<br />
<a href="view_orders.php">&lt Back</a>
</body>
</html>
```

จากกรณีศึกษาร้านมือถือออนไลน์ 7-Elephant เป็นตัวอย่างแสดงให้เห็นถึงการประยุกต์ใช้ความรู้ตั้งแต่แนวคิดพื้นฐานเกี่ยวกับฐานข้อมูล การออกแบบ และการพัฒนาฐานข้อมูลเพื่อใช้งานได้จริง ระบบในตัวอย่างเป็นระบบที่มีขนาดเล็ก ความต้องการของผู้ใช้และข้อกำหนดของระบบไม่มากนัก ในการปฏิบัติงานจริงนั้นระบบฐานข้อมูลอาจมีขนาดใหญ่หลายสิบตาราง มีความต้องการของผู้ใช้จำนวนมาก ซึ่งการออกแบบและพัฒนาจะซับซ้อนกว่านี้ อย่างไรก็ตามเราสามารถประยุกต์ใช้หลักการต่างๆ ที่ได้อธิบายมาทั้งหมดนี้ในเป็นฐานในการต่อยอดและพัฒนาระบบฐานข้อมูลและระบบสารสนเทศใจชีวิตจริง ให้ใช้งานได้อย่างมีประสทธิภาพต่อไป