เอกสารประกอบการใช้ SQL เบื้องต้น

(การออกแบบฐานข้อมูลระดับกายภาพ : Physical Database Design)

สุนี โชติดิลก

เอกสารฉบับนี้เป็นส่วนหนึ่งของรายวิชา ระบบฐานข้อมูล

คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยราชภัฏพระนคร

สารบัญ

ภาษาเอสคิวแอล
(Structured Query Language)

ภาษาสอบถามเชิงโครงสร้าง (Structured Query Language: SQL, อ่านอย่างย่อว่า *เอสคิวแอล, ซีเควล* (Sequel), *ซีควล*) เป็นภาษาสอบถามข้อมูลที่นิยมมากที่สุดในการปฏิบัติการกับฐานข้อมูลแบบสัมพันธ์ (relational database) โดยเฉพาะ และเป็นภาษาที่มีลักษณะคล้ายกับภาษาอังกฤษ เพื่อสร้าง แก้ไขและเรียกใช้ฐานข้อมูล โดยใช้มาตรฐานของแอนซี (ANSI) และ ไอเอสโอ (ISO) ภาษา SQL ถูกพัฒนาขึ้นจากแนวคิดของ relational calculus และ relational algebra เป็นหลัก เริ่มพัฒนาครั้งแรกโดย almaden research center ของ บริษัท IBM โดยมีชื่อเริ่มแรกว่า "ซีเควล" (Sequel) ต่อมาได้เปลี่ยนชื่อเป็น "เอสคิวแอล" (SQL) ภาษา SQL ได้ถูกนำมาพัฒนาโดยผู้ผลิตซอฟแวร์ด้านระบบจัดการฐานข้อมูลเชิงสัมพันธ์จนเป็นที่นิยมกันอย่างแพร่หลายในปัจจุบัน โดยผู้ผลิตแต่ละรายก็พยายามที่จะพัฒนาระบบจัดการฐานข้อมูลของตนให้มีลักษณะเด่นเฉพาะขึ้นมา ทำให้รูปแบบการใช้คำสั่ง SQL มีรูปแบบที่แตกต่างกันไปบ้าง เช่น ORACLE ACCESS SQL Base ของ Sybase INGRES หรือ SQL Server ของ Microsoft เป็นต้น ดังนั้นในปี ค.ศ.1986 สถาบัน มาตรฐานแห่งชาติของสหรัฐอเมริกา (American National Standards Institute : ANSI) ได้กำหนดมาตรฐาน SQL ขึ้น โปรแกรมจัดการฐานข้อมูลที่ขายในท้องตลาดได้ขยายความสามารถของ SQLให้มีความสามารถมากกว่าข้อกำหนดของ ANSI โดยเพิ่มคุณสมบัติต่าง ๆ ที่คิดว่าเป็นประโยชน์ แต่โดยหลักทั่วไปแล้วก็ยังปฏิบัติตามมาตรฐานของ ANSI คู่มือเล่มนี้จะกล่าวถึงคำสั่งที่เป็นรูปแบบมาตรฐานของภาษา SQL โดยทั่วไป

ลักษณะการใช้งานของภาษา SQL

ภาษา SQL เป็นส่วนประกอบหนึ่งของโปรแกรมระบบจัดการฐานข้อมูล (DBMS) มักพบใน DBMS เชิงสัมพันธ์หลายตัวและเป็นที่นิยมใช้ในปัจจุบัน ภาษา SQL ง่ายต่อการเรียนรู้ การใช้งานในภาษา SQL แบ่งเป็น 2 ลักษณะ คือ ภาษา SQL ที่โต้ตอบได้ (interactive SQL) และภาษา SQL ที่ฝังในโปรแกรม (embedded SQL) มีรายละเอียดพอสังเขปดังนี้

1. ภาษา SQL ที่โต้ตอบได้ ใช้เพื่อปฏิบัติงานกับฐานข้อมูลโดยตรง เป็นการใช้คำสั่งภาษา SQL สั่งงานบนจอภาพ โดยเรียกดูข้อมูลได้โดยตรงในขณะที่ทำงาน เพื่อให้ได้ผลลัพธ์ที่นำไปใช้ได้ ตัวอย่างเช่น ต้องการเรียกดูข้อมูลในคอลัมน์ SALENAME และ SALECOM จากตาราง SALESTAB จะใช้คำสั่งของภาษา SQL ดังนี้

```
SELECT SALENAME, SALECOM
  FROM SALESTAB;
```

โดยตาราง SALESTAB มีรายละเอียดของตารางดังนี้

SALENO	SALENAME	ADDRESS	SALECOM
1001	KANJANA	Rayong	0.15
1002	BENJAWAN	Bangkok	0.12
1004	KOMCHAN	Nonthaburi	0.14
1008	MITREE	Chianrai	0.13
1003	SUTI	Bangkok	0.11

ผลของคำสั่งจะแสดงผลลัพธ์ดังนี้ทันที

SALENAME	SALECOM
KOMCHAN	0.15
BENJAWAN	0.12
KANJANA	0.14
MITREE	0.13
SUTI	0.11

2. **ภาษา SQL ที่ฝังในโปรแกรม** เป็นภาษา SQL ที่ประกอบด้วยคำสั่งต่าง ๆ ของภาษา SQL ที่เขียนในโปรแกรมภาษาอื่น เช่น โคบอล ปาสคาล ภาษาซี เป็นต้น ลักษณะของคำสั่งภาษา SQL จะแตกต่างจากภาษาอื่นๆ คือภาษา SQL ไม่มีคำสั่งที่เกี่ยวกับการควบคุม (control statement) เหมือนภาษาอื่น เช่น if.....then.....else for.....do หรือคำสั่ง loop หรือคำสั่ง while ทำให้มีข้อจำกัดในการเขียนชุดคำสั่งดังนั้นการเขียนคำสั่งภาษา SQL ฝังในโปรแกรมอื่นจะทำให้ภาษา SQL มีความสามารถและมีประสิทธิภาพมากยิ่งขึ้น ผลลัพธ์ของคำสั่งที่เกิดจากภาษา SQL ที่ฝังในโปรแกรมจะถูกส่งผ่านไปให้กับตัวแปรหรือพารามิเตอร์ที่ใช้ เช่น

```
begin
        readin(id-num, salesperson, loc, comm);
        EXEC SQL INSERT INTO SALESTAB
        VALUES(:id-num,:salesperson,:loc,:comm);
end;
```

จากตัวอย่างข้างต้น ถ้าใช้คำสั่ง

```
INSERT INTO SALESTAB
        VALUES (:id-num,:salesperson,: loc, :comm);
```

เพียงอย่างเดียว จะทำให้คำสั่งนี้ใส่ค่า id-num salesperson loc comm ใส่ค่าได้เพียงครั้งเดียว แต่เมื่อนำคำสั่งนี้มาใส่ไว้ในภาษาปาสคาลข้างต้นจะทำให้คำสั่งดังกล่าวมีความสามารถสูงขึ้นคือคำสั่งนี้จะสามารถทำงานซ้ำ(loop) โดยใส่ค่าต่างๆ ลงในตัวแปรเพื่อให้ทำซ้ำกันหลายๆ ครั้ง โดยจากตัวอย่างส่วนของโปรแกรมภาษาปาสคาลจะกำหนดลูปวนซึ่งจะอ่านค่าจากแฟ้มข้อมูลแล้วเก็บค่านั้นไว้ในตัวแปร id-num, salesperson, loc, comm ของตารางSALESTAB การอ่านค่าแล้วเก็บค่าไว้ในตัวแปรจะทำซ้ำจนกระทั่งข้อมูลหมดจากแฟ้มข้อมูล ทั้งสองรูปแบบคือ ภาษา SQL ที่โต้ตอบได้และภาษา SQL ที่ฝังในโปรแกรมจะมีลักษณะของคำสั่งที่ใช้งานเหมือนกัน จะต่างกันแต่เพียงภาษา SQL ที่ฝังในโปรแกรมจะมีวิธีการเชื่อมโยงกับภาษาอื่น

ประเภทของคำสั่งของภาษา SQL

ภาษา SQL เป็นภาษาที่ใช้งานได้ตั้งแต่ระดับเครื่องคอมพิวเตอร์ส่วนบุคคลพีซีไปจนถึงระดับเมนเฟรม ประเภทของคำสั่งในภาษา SQL แบ่งออกเป็น 3 ประเภท คือ

1) ภาษานิยามข้อมูล (Data Definition Language: DDL) ประกอบด้วยคำสั่งที่ใช้ในการกำหนดโครงสร้างข้อมูลในตาราง (ตารางที่ผ่านการนอร์มัลไลซ์ในรูป 3NF) โดยระบุว่าตารางประกอบด้วยคอลัมน์ชื่ออะไร แต่ละคอลัมน์เก็บข้อมูลประเภทใด คำสั่งการเพิ่มคอลัมน์ คำสั่งการกำหนดดัชนี คำสั่งการกำหนดวิวหรือตารางเสมือนของผู้ใช้ ภาษานิยามข้อมูลได้แก่ คำสั่ง CREATE TABLE, CREATE INDEX, DROP INDEX, ALTER TABLE, DROP TABLE

2) ภาษาจัดการข้อมูล (Data Manipulation Language: DML) ประกอบด้วยคำสั่งที่ใช้ในการเรียกใช้ข้อมูล คำสั่งที่ใช้เปลี่ยนแปลงข้อมูล คำสั่งที่ใช้เพิ่มหรือลบข้อมูล ภาษาจัดการข้อมูลได้แก่ คำสั่ง SELECT, UPDATE, DELETE, INSERT

3) ภาษาควบคุมข้อมูล (Data Control Language: DCL) ประกอบด้วยคำสั่งที่ใช้ในการควบคุม การเกิดภาวะพร้อมกัน หรือการป้องกันการเกิดเหตุการณ์ที่ผู้ใช้หลายคนเรียกใช้ข้อมูลพร้อมกัน และคำสั่งที่เกี่ยวข้องกับการควบคุมความปลอดภัยของข้อมูลด้วยการกำหนดสิทธิของผู้ใช้ที่แตกต่างกัน เป็นต้น ภาษาควบคุมข้อมูลได้แก่ คำสั่ง GRANT, REVOKE

ชนิดของข้อมูลที่ใช้ในภาษา SQL

ชนิดข้อมูล	คำอธิบาย
Char(size)	ข้อมูลชนิด String ใช้เก็บข้อความที่มีความยาวคงที่
Varchar(size)	ข้อมูลชนิด String ใช้เก็บข้อความที่มีความยาวไม่เกิน size ที่กำหนด
Date	เก็บข้อมูลวันเดือนปี
Time	เก็บข้อมูลเวลา
Float(n)	ตัวเลขทศนิยม
Smallint	เก็บตัวเลขจำนวนเต็ม 2 byte
Integer	เก็บตัวเลขจำนวนเต็ม 4 byte

หมายเหตุ ชนิดข้อมูลในภาษา SQL จะแตกต่างกันขึ้นอยู่กับระบบฐานข้อมูลนั้น ๆ

ตารางข้อมูลที่บรรจุในฐานข้อมูลเชิงสัมพันธ์ เป็นตาราง 2 มิติ ประกอบด้วย แถว (rows) และคอลัมน์ (columns) เช่น ตารางพนักงานขาย (SALESTAB)

พนักงานขาย

SALENO	SALENAME	ADDRESS	SALECOM
1001	KOMCHAN	Rayong	0.15
1002	BENJAWAN	Bangkok	0.12
1004	KANJANA	Nonthaburi	0.14
1008	MITREE	Chianrai	0.13
1003	SUTI	Bangkok	0.11

คำอธิบายของคอลัมน์ต่าง ๆ ในตารางพนักงานขาย

คอลัมน์	ชนิดข้อมูล	รายละเอียด
SALENO	Integer	เลขประจำตัวพนักงานขาย
SALENAME	Char(30)	ชื่อพนักงานขาย
ADDRESS	Char(100)	ที่อยู่ของพนักงานขาย
SALECOM	Decimal	ค่าคอมมิชชั่นของพนักงานขายตามคำสั่งซื้อ

เมื่อต้องการนำตารางที่ผ่านกระบวนการออกแบบฐานข้อมูลระดับตรรกะ หรือกล่าวอีกลักษณะหนึ่งคือตารางที่อยู่ในรูปแบบนอร์มัลระดับที่ 3 (3NF) แล้ว เมื่อต้องการนำตารางมาเก็บในระบบฐานข้อมูลจะต้องดำเนินการออกแบบฐานข้อมูลระดับกายภาพ (Physical Database Design) โดยผ่านโปรแกรมจัดการฐานข้อมูล หรือเขียนคำสั่งในภาษา SQL ทั้ง 3 ประเภท ได้แก่ คำสั่งภาษาสำหรับการนิยามข้อมูล (DDL) คำสั่งภาษาสำหรับการจัดการข้อมูล (DML) และคำสั่งภาษาควบคุม (DCL)

คำสั่งภาษาสำหรับการนิยามข้อมูล(DDL) เป็นภาษาที่ใช้นิยามโครงสร้างของฐานข้อมูล เพื่อทำการสร้างเปลี่ยนแปลงหรือยกเลิกโครงสร้างของฐานข้อมูลตามที่ได้ออกแบบไว้ โครงสร้างของฐานข้อมูลสามารถเรียกได้อีกอย่างว่า สคีมา (sehema) ประกอบด้วยคำสั่ง CREATE TABLE, CREATE INDEX, DROP INDEX, ALTER TABLE, DROP TABLE ซึ่งมีรายละเอียดดังนี้

1. **คำสั่ง CREATE TABLE** เป็นคำสั่งที่ใช้ในการสร้างตาราง จะกำหนดชื่อตารางและกำหนดลักษณะข้อมูลเป็นคอลัมน์ต่างๆ ซึ่งได้จากการนอร์มัลไลซ์ให้อยู่ในรูป 3NF มาแล้ว (การออกแบบในระดับตรรกะ) และกำหนดชนิดของข้อมูลของแต่ละคอลัมน์ คำสั่งการสร้างตารางมีรูปแบบไวยากรณ์ดังต่อไปนี้

```
CREATE TABLE<table name>
(<column name><>[<size>]{[constraint<constraint_name>]constraint_type}
[,<column name>data type>[<size>},.....]);
```

CREATE TABLE เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการสร้างตาราง
table name ชื่อตารางที่ต้องการสร้าง
column name ชื่อของคอลัมน์แต่ละคอลัมน์
data type ชนิดข้อมูลของคอลัมน์นั้น ๆ
constraint ข้อกำหนดของคอลัมน์
constraint_name ชื่อของข้อกำหนดที่ต้องการสร้างให้กับคอลัมน์
constraint_type ประเภทของข้อกำหนด

ตัวอย่างที่ 1 การสร้างตารางพนักงานขาย

```
CREATE TABLE SALESTAB
        (SALENO integer,
        SALENAME char(10),
        ADDRESS char(10));
```

จากคำสั่งจะทำให้ได้ตารางพนักงานขายที่มีคอลัมน์ SALENO มีชนิดข้อมูลเป็น integer คอลัมน์ SALENAME มีชนิดข้อมูลเป็น char มีความยาม 10 ตัวอักษร คอลัมน์ ADDRESS มีชนิดข้อมูลเป็น char มีความยาว 10 ตัวอักษร

ผลของคำสั่งการสร้างตารางจะได้ตารางพนักงานขายที่ยังไม่มีข้อมูลใด ๆ เป็นเพียงแต่โครงของตารางเท่านั้นดังนี้

SALENO	SALENAME	ADDRESS	SALECOM

ข้อกำหนดในการสร้างตาราง

การสร้างตารางมีกฎเกณฑ์ข้อจำกัด (constraints) ในการป้อนข้อมูลลงในคอลัมน์ต่างๆ ของตารางซึ่งเป็นการควบคุมความถูกต้องสมบูรณ์ (integrity) ที่จัดเก็บในฐานข้อมูลให้มีความถูกต้องตามที่กำหนดไว้หรือตามที่ควรจะเป็น การกำหนดข้อจำกัดทำให้ข้อมูลมีความน่าเชื่อถือมากยิ่งขึ้น การกำหนดข้อจำกัดมีดังนี้

1. การกำหนดไม่ให้คอลัมน์ที่เป็นคีย์หลัก (primary key) เป็นค่าว่าง (NOT NULL) หรือคอลัมน์ที่เป็นนันคีย์อื่นๆ เป็นค่าว่าง โดยระบุคำสั่ง "NOT NULL" เช่น ต้องการให้ลูกค้าทุกคนในตารางลูกค้าต้องมีข้อมูลชื่อ โดยทั่วไปการสร้างตารางถ้าในคอลัมน์ไม่ระบุคำว่า "NOT NULL" คอลัมน์นั้นจะถูกระบุให้เป็นค่า NULL โดยปริยาย (DEFAULT) นั่นคือ คอลัมน์นั้นสามารถมีค่าว่างได้ (NULL)

ตัวอย่างที่ 2 สมมติว่าต้องการกำหนดให้ตารางพนักงานขายในคอลัมน์ SALENO และคอลัมน์ SALENAME ไม่ให้เป็นค่าว่าง (NOT NULL) จะสามารถสร้างตารางพนักงานด้วยคำสั่งดังนี้

```
CREATE TABLE SALESTAB
        (SALENO integer NOT NULL,
        SALENAME char(10) NOT NULL,
        ADDRESS char(10));
```

2. การกำหนดไม่ให้มีค่าซ้ำกัน (UNIQUE) ใช้กับการสร้างตารางโดยกำหนดให้คอลัมน์นั้นทั้งตารางไม่ให้มีค่าซ้ำกัน โดยใช้คำว่า "UNIQUE" เช่น คอลัมน์รหัสพนักงานที่เป็นคีย์หลัก และไม่ต้องการให้มีค่าซ้ำ จะใช้คำว่า UNIQUE เป็นการระบุข้อจำกัดนี้

ตัวอย่างที่ 3 สมมติว่าต้องการกำหนดให้ตารางพนักงานขายในคอลัมน์ SALENO และคอลัมน์ SALENAME ไม่ให้เป็นค่าว่าง (NOT NULL) และไม่ให้มีค่าซ้ำกัน จะสามารถสร้างตารางพนักงานด้วยคำสั่งดังนี้

```
CREATE TABLE SALESPEOPLE
        (SALENO integer NOT NULL UNIQUE,
        SALENAME char(10) NOT NULL UNIQUE,
        ADDRESS char(10),
        SALECOM decimal);
```

ข้อสังเกต UNIQUE มักจะนำไปใช้กับข้อมูลที่ไม่สามารถซ้ำกันได้ เช่น ชื่อสารเคมี, ชื่อเว็บไซต์ เป็นต้น

3. การกำหนดคีย์หลัก (primary key) สามารถกำหนดได้ 2 วิธีคือ

3.1 การกำหนดให้คอลัมน์เดียวเป็นคีย์หลัก (single key)

ตัวอย่างที่ 4 สมมติว่าต้องสร้างตารางพนักงานที่กำหนดให้คอลัมน์ SALENO เป็นคีย์หลัก (primary key) โดยไม่ให้มีค่าซ้ำกันและคอลัมน์ SALENAME ไม่ให้เป็นค่าว่าง (NOT NULL) และไม่ให้มีค่าซ้ำกัน จะสามารถสร้างตารางพนักงานด้วยคำสั่งดังนี้

```
CREATE  TABLE SALESPEOPLE
        (SALENO integer NOT NULL UNIQUE PRIMARY KEY,
        SALENAME char(10) NOT NULL UNIQUE,
        ADDRESS char(10));
```

3.2 การกำหนดให้คอลัมน์มากกว่า 1 คอลัมน์เป็นคีย์หลัก(composite key) เพราะในบางครั้งการอ้างอิงคีย์หลักอาจต้องใช้คอลัมน์มากกว่า 1 คอลัมน์เป็นคีย์หลัก

ตัวอย่างที่ 5 สมมติว่าต้องสร้างตาราง NAMEFIELD โดยกำหนดให้คอลัมน์ FIRSTNAME และคอลัมน์ LASTNAME เป็นคีย์หลัก (primary key) จะสามารถสร้างตาราง NAMEFIELD ด้วยคำสั่งดังนี้

```
CREATE TABLE NAMEFIELD
        (FIRSTNAME char(10) NOT NULL,
        LASTNAME char(10) NOT NULL UNIQUE,
        CITY char(10),
        PRIMARY KEY (FIRSTNAME,LASTNAME));
```

3.3 การกำหนดคีย์นอก (foreign key) ซึ่งเป็นคอลัมน์ของตารางหนึ่งที่ใช้เชื่อมโยงหรืออ้างอิงข้อมูลกับอีกตารางหนึ่งที่มีชื่อคอลัมน์เดียวกัน เช่น ลูกค้าในตารางลูกค้า แต่ละคนมีคอลัมน์ SALENO ซึ่งเป็นคอลัมภ์ที่อยู่ในตารางพนักงานขาย

หากกำหนดคีย์นอกเป็นข้อจำกัดในระดับคอลัมน์จะใช้คำสั่ง REFERENCE ต่อท้ายประเภทและขนาดของคอลัมน์ที่เป็นคีย์นอก

CUSNO	CUSNAME	SALENO
2001	Arlee	1001
2002	Tanachote	1003
2003	Tawatchai	1002
2004	Amporni	1002
2006	Surasit	1001
2008	Jintana	1007
2007	Siriwan	1004

ตารางลูกค้า (CUSTOMERTAB)

SALENO	SALENAME	SALECOM
1001	Chaiwat	0.12
1002	Mitree	0.13
1004	Beejawan	0.11
1007	Kanjana	0.15
1003	Ternjai	0.10

ตารางพนักงานขาย (SALESTAB)

จากตารางข้างต้นแสดงการอ้างอิงข้อมูลคอลัมน์ที่มีชื่อเดียวกันคือ คอลัมน์ SALENO ในตารางลูกค้าและตารางพนักงานขาย

ตัวอย่างที่ 6 ถ้าต้องการสร้างตารางลูกค้า (CUSTOMERSTAB) โดยกำหนดให้คอลัมน์ CUSNO เป็น PRIMARY KEY และคอลัมน์ SALENO เป็นคีย์นอก (foreign key) ที่ใช้เชื่อมโยงหรืออ้างอิงข้อมูลกับตารางพนักงานขาย (SALESTAB) โดยใช้คำสั่งดังนี้

```
CREATE TABLE CUSTOMERSTAB
        (CUSNO integer NOT NULL PRIMARY KEY,
        CUSNAME char(10),
        ADDRESS char(10),
        SALENO integer,
        FOREIGN KEY (SALENO) REFERENCES SALESTAB(SALENO));
```

หรือ

```
CREATE TABLE CUSTOMERSTAB
        (CUSNO integer NOT NULL PRIMARY KEY,
        CUSNAME char(10),
        ADDRESS char(10),
        SALENO integer, REFERENCES SALESTAB(SALENO));
```

การกำหนดคีย์นอก (foreign key) เป็นคอลัมน์ของตารางหนึ่งที่ใช้เชื่อมโยงหรืออ้างอิงข้อมูลกับอีกตารางหนึ่งจะทำให้การปรับปรุงตารางมีผลต่อตารางอ้างอิง คำสั่งที่มีผลต่อการปรับปรุงตาราง ได้แก่คำสั่ง "CASCADES" และ "RESTRICTED" ทั้ง 2 คำสั่งนี้ใช้เพื่อควบคุมความถูกต้องครบถ้วนสมบูรณ์ในการอ้างอิงข้อมูล (referential integrity) ดังตัวอย่างต่อไปนี้

```
CREATE TABLE CUSTOMERSTAB
        (CUSNO integer NOT NULL PRIMARY KEY,
        CUSNAME char(10),
        ADDRESS char(10),
        SALENO integer, REFERENCES SALESTAB(SALENO)
        UPDATE OF SALESTAB CASCADES,
        DELETE OF SALESTAB RESTRICTED);
```

ผลของคำสั่งนี้จะทำให้เมื่อต้องการปรับปรุงตารางที่สร้างขึ้นหรือตารางที่อ้างอิงถึง จะต้องมีเงื่อนไขในการปรับปรุง โดย

1. คำสั่ง UPDATE OF SALESTAB CASCADES จะทำให้เมื่อมีการปรับปรุงคอลัมน์ SALENO ในตาราลูกค้าจะทำให้คอลัมน์ SALENO ในตารางพนักงานขายถูกปรับปรุงไปด้วย
2. คำสั่ง DELETE OF SALESTAB RESTRICTED จะทำให้เมื่อต้องการลบคอลัมน์ SALENO ในตารางลูกค้าจะไม่สามารถลบได้ ถ้าคอลัมน์ SALENO ในตารางพนักงานขายยังมีข้อมูลอยู่

CUSNO	CUSNAME	SALENO
2001	Arlee	1001
2002	Tanachote	1003
2003	Tawatchai	1002
2004	Amporni	1002
2006	Surasit	1001
2008	Jintana	1008
2007	Siriwan	1004

ตารางลูกค้า (CUSTOMERTAB)

SALENO	SALENAME	SALECOM
1001	KANJANA	0.15
1002	BENJAWAN	0.12
1004	KOMCHAN	0.14
1008	MITREE	0.13
1003	SUTI	0.11

ตารางพนักงานขาย (SALESTAB)

จากตัวอย่างสมมติว่าต้องการลบ KANJANA ออกจากตารางพนักงานขาย (SALESTAB) คำสั่งนี้ก็จะไม่เป็นที่ยอมรับ นอกเสียจากเปลี่ยนค่าคอลัมน์ SALENO ของลูกค้าชื่อ Arlee และ Surasit ไม่เป็น SALENO ของพนักงานขายผู้อื่น หรือกล่าวอีกอย่างหนึ่งว่าจะเปลี่ยนค่า SALENO ของ KANJANA เป็น 1009 แล้ว ในตารางลูกค้า Arlee และ Surasit ก็จะเปลี่ยนค่า SALENO ของทั้งสองคนนั้นตามไปด้วยโดยอัตโนมัติ

2. คำสั่ง DROP TABLE เป็นคำสั่งการลบโครงสร้างตารางออกจากระบบฐานข้อมูล มีรูปแบบดังนี้

DROP TABLE <table name>[CASCADE CONSTRAINTS];

DROP TABLE	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการลบโครงสร้างตาราง
table name	ชื่อตารางที่ต้องการลบ
CASCADE CONSTRAINTS	ระบบจัดการฐานข้อมูลจะทำการลบข้อจำกัดต่าง ๆ (constraint) ที่มีการอ้างถึงตารางทิ้งไปให้ด้วยทั้งหมด

ตัวอย่างที่ 7 ถ้าต้องการลบตารางพนักงานขาย (SALESTAB) จะใช้คำสั่งดังนี้

```
DROP TABLE SALESTAB;
```

ผลของคำสั่งการลบโครงสร้างตาราง จะทำให้ข้อมูลถูกลบไปด้วยจะทำให้ดัชนี (index) ทุกตัวและตารางเสมือนหรือวิวที่สร้างขึ้นสำหรับตาราง SALESTAB นี้ จะถูกลบไปพร้อม ๆ กันด้วย เมื่อมีการใช้คำสั่งลบโครงสร้างตารางเกิดขึ้นก่อนที่จะลบโครงสร้างตารางข้อมูล DBMS จะเตือนผู้ใช้ถึงผลที่เกิดจากการลบโครงสร้างตาราง

3. คำสั่ง ALTER TABLE เป็นคำสั่งที่ใช้ในการเปลี่ยนแปลงโครงสร้างตาราง เช่น ต้องการเพิ่มหรือลบบางคอลัมน์ที่เป็นโครงสร้างของตารางออก หรือต้องการเปลี่ยนประเภทข้อมูลของคอลัมน์ ซึ่งในกรณีที่ตารางมีข้อมูลและกำหนดโครงสร้างไปแล้ว การแก้ไขโครงสร้างข้อมูลอาจมีผลกระทบกับข้อมูลที่มีอยู่ รูปแบบของคำสั่ง ALTER TABLE มี 2 แบบ คือ

3.1 ALTER TABLE ที่ใช้ในการเพิ่มคอลัมน์

3.2 ALTER TABLE ที่ใช้ในการเปลี่ยนชื่อคอลัมน์

คำสั่ง ALTER TABLE เป็นคำสั่งที่ใช้ในการแก้ไขปรับปรุงโครงสร้างตาราง เมื่อจำเป็นที่ต้องปรับปรุงจากโครงสร้างเดิมตามที่ได้กำหนดไว้ตั้งแต่สร้างตารางในครั้งแรก คำสั่ง ALTER TABLE มีรูปแบบดังนี้

ALTER TABLE<table name>

Database update(<column_name>data type[SIZE]);

ALTER TABLE	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเปลี่ยนแปลงโครงสร้างตาราง
table name	ชื่อตารางที่จะเปลี่ยนแปลง
Database update	คำสั่งการเปลี่ยนแปลง
column_name	ชื่อคอลัมน์
data type[SIZE]	ชนิดข้อมูลและขนาดของข้อมูล

ตัวอย่างที่ 8 ถ้าต้องการเปลี่ยนแปลงโครงสร้างตารางโดยการเพิ่มคอลัมน์ลงไปบนโครงสร้างตารางเดิมจะใช้คำสั่งดังนี้

ALTER TABLE SALESPEOPLE ADD SALESTAB_FAX CHAR(15);

ผลของคำสั่งจะทำให้ตารางพนักงานขายมีคอลัมน์ SALESTAB_FAX ที่มีชนิดข้อมูลเป็น char มีความยาว 15 ตัวอักษรเพิ่มขึ้น

ข้อแตกต่างระหว่างคีย์หลักและดัชนี

คีย์หลัก ได้แก่ คอลัมน์ 1 คอลัมน์หรือหลายคอลัมน์ที่ทำให้แต่ละแถวในตารางข้อมูลมีค่าของข้อมูลที่ไม่ซ้ำกัน (unique) เช่น คอลัมน์รหัสลูกค้า (CUSNO) ของตารางลูกค้า (COSTOMERTAB) ซึ่งใช้แทนรหัสลูกค้าแต่ละคนจะมีรหัสประจำตัวไม่ซ้ำกัน คีย์หลักเป็นพื้นฐานในการเชื่อมโยงกันระหว่างตารางและควบคุมความถูกต้องครบถ้วนสมบูรณ์ (integrity) ในการตรวจสอบความซ้ำกันของข้อมูลระหว่างที่ทำการป้อนข้อมูลหรือกำหนดข้อมูลใหม่ให้กับตาราง ส่วนดัชนีเป็นการสร้างโดยการเลือกคอลัมน์ใดคอลัมน์หนึ่งหรือหลายคอลัมน์จากตารางขึ้นมาเป็นดัชนี โดยในตารางหนึ่ง ๆ สามารถมีดัชนีได้หลายดัชนี คอลัมน์ที่จะเลือกเป็นดัชนีควรจะมีค่าของข้อมูลไม่ซ้ำกัน (unique) ในแต่ละแถว ถ้าเลือกคอลัมน์ที่เป็นดัชนีที่สามารถมีค่าว่างได้จะทำให้ DBMS ไม่สามารถนำดัชนีที่เป็นค่าว่างนั้นไปค้นหาข้อมูลในฐานข้อมูลได้ การสร้างดัชนีก็เพื่อเป็นตัวนำทางในการสืบค้นข้อมูลให้เร็วขึ้น

4. คำสั่ง CREATE INDEX เป็นคำสั่งที่มีส่วนสำคัญมากต่อฐานข้อมูลเชิงสัมพันธ์ คือ ช่วยเพิ่มความสามารถในการค้นหาข้อมูลได้เร็วยิ่งขึ้น โดยดัชนีที่ถูกสร้างขึ้นในแต่ละแถวจะถูกเก็บเป็นตารางแยกจากตารางข้อมูล ซึ่งเป็นความสะดวกในการค้นหาข้อมูลในแต่ละแถว DBMS สามารถทำการค้นหาข้อมูลในตารางดัชนี เมื่อพบดัชนีที่ต้องการจะชี้ไปยังตารางข้อมูลนั้นๆ ซึ่งถ้าตารางข้อมูลใดไม่มีการสร้างดัชนีไว้การค้นหาข้อมูลจะทำการค้นหาแบบเรียงลำดับจากแถวแรกจนถึงแถวสุดท้าย นอกจากนี้ดัชนียังช่วยในการตรวจสอบและควบคุมไม่ให้มีข้อมูลเดียวกันหลายแถวซ้ำกันในตารางได้อย่างอัตโนมัติ โดยดัชนีสามารถช่วยให้ผู้ใช้ค้นหาข้อมูลแต่ละแถวตามที่กำหนดเฉพาะเจาะจงตามต้องการได้โดยอัตโนมัติ

ในการค้นหาข้อมูลใดข้อมูลหนึ่งในตารางข้อมูล ถ้าไม่มีดัชนีในการค้นหาจะทำให้เสียเวลาในการค้นหาพอสมควร ดัชนีจะเป็นตัวนำทางในการค้นหา เมื่อสร้างดัชนีจากคอลัมน์หนึ่งของตาราง DMBS จะเก็บ

คอลัมน์นั้นเรียงลำดับที่เหมาะสมนั้นไว้ เช่น ตารางลูกค้ามีข้อมูลป้อนไว้หลายรายการ และต้องการค้นหาข้อมูลของลูกค้าหมายเลข 2999 ถ้าไม่ได้เรียงลำดับข้อมูลตามหมายเลขลูกค้าไว้ โปรแกรมจะต้องค้นหาไปที่ละแถวทั้งตารางเพื่อหาข้อมูลลูกค้าที่มีค่าหมายเลข 2999 ในคอลัมน์ CUSNO อย่างไรก็ดีถ้ามีดัชนีอยู่ในคอลัมน์ CUSNO โปรแกรมก็จะตรงไปที่หมายเลข 2999 ในดัชนีเลย การสร้างดัชนีจะทำให้การค้นหาข้อมูลเร็วขึ้น แต่การสร้างดัชนีก็เปลืองพื้นที่ในหน่วยความจำ

การสร้างดัชนีมีรูปแบบคำสั่งของการสร้างดังนี้

CREATE INDEX<index name>
ON<table name>(<column>name>[,<column name>]...);

CREATE INDEX	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการสร้างดัชนี
index name	ชื่อดัชนี
table name	ชื่อตารางที่จะสร้างดัชนี

ตัวอย่างที่ 9 ถ้าตารางลูกค้าเป็นตารางที่พนักงานขายอ้างถึงบ่อยที่สุดเพื่อถามหาลูกค้าของตนเองแล้ว ก็ควรสร้างดัชนีขึ้นในคอลัมน์พนักงาน (SALENO) ของตารางลูกค้า จะใช้คำสั่งดังนี้

CREATE UNIQUE INDEX CLIENTGROUP ON CUSTOMERSTAB(SALENO);

จากคำสั่ง "UNIQUE" เป็นการระบุว่าดัชนี (index) ที่สร้างขึ้นในคอลัมน์ CLIENTGROUP ซึ่งในคอลัมน์นี้จะมีค่าที่ซ้ำกันไม่ได้ และจะใช้คอลัมน์ SALENO เป็นข้อมูลในการค้นหา ผลของคำสั่งตารางลูกค้าจะมี CLIENTGROUP เป็นดัชนี (index) ในการค้นหาข้อมูล โดยเรียงลำดับตามข้อมูลในคอลัมน์ SALENO ของตารางลูกค้า index คอลัมน์ CLIENTGROUP ที่สร้างขึ้นนี้จะไม่ถูกเก็บไว้ในตารางลูกค้า แต่จะถูกเก็บไว้แยกต่างหากในหน่วยความจำของเครื่องคอมพิวเตอร์

คำสั่งบันทึกข้อมูล ปรับปรุงข้อมูล ลบข้อมูลและเรียกข้อมูลอย่างง่าย :

ในระบบฐานข้อมูล การบันทึกข้อมูล การปรับปรุงข้อมูลและการลบข้อมูลถือเป็นกิจกรรมสำคัญ ภาษา SQL มีคำสั่งสำหรับการจัดการข้อมูล (DML) ซึ่งเป็นภาษาที่ใช้จัดการข้อมูลในตาราง ตัวอย่างของคำสั่งในภาษาสำหรับการจัดการข้อมูล จะเป็นคำสั่งเกี่ยวกับการปรับปรุงข้อมูล ได้แก่ การเพิ่มข้อมูล (INSERT) การปรับปรุง (UPDATE) และการลบข้อมูล (DELETE) คำสั่งทั้ง 3 นี้ เมื่อดำเนินการในภาษา SQL จะไม่แสดงผลลัพธ์ออกมาทางหน้าจอ แต่ผลของคำสั่งจะมีผลต่อข้อมูล ผู้ใช้สามารถดูผลของการใช้คำสั่งในการเพิ่มข้อมูล การปรับปรุงข้อมูล และการลบข้อมูล โดยใช้คำสั่งการเรียกค้นข้อมูล (SELECT) ซึ่งมีรายละเอียดดังนี้

1. คำสั่ง INSERT เป็นคำสั่งการเพิ่มข้อมูลในตารางมีอยู่ 2 รูปแบบคือ การเพิ่มข้อมูลเข้าไปทีละแถว และการเพิ่มข้อมูลโดยการดึงกลุ่มข้อมูลด้วยคำสั่งค้นหาข้อมูล

1.1 **คำสั่งการเพิ่มข้อมูลทีละแถวโดยระบุข้อมูลที่จะ** INSERT เข้าไปโดยตรง รูปแบบของคำสั่งเป็นดังนี้

INSERT INTO <table name>[(column 1, column 2,...)]
VALUE(<value1,value2,...>);

INSERT INTO	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเพิ่มข้อมูล
table name	ชื่อตารางที่จะเพิ่มข้อมูล
column 1, column 2,...	คอลัมน์ที่ต้องการเพิ่มข้อมูล
value1,value2,	ค่าข้อมูลของแต่ละคอลัมน์ที่ต้องการเพิ่ม

ตัวอย่างที่ 10 ถ้าต้องการจะใส่ข้อมูลทุกคอลัมน์ลงในตารางลูกค้า

```
INSERT INTO SALESTAB
VALUES(1001, "KANJANA", "Rayong",0.15);
```

ผลของคำสั่งนี้ จะมีข้อมูลปรากฏในทุกคอลัมน์ในตารางพนักงานขายดังนี้

SALENO	SALENAME	ADDRESS	SALECOM
1001	KANJANA	Rayong	0.15

ตัวอย่างที่ 11 ถ้าต้องการจะใส่ข้อมูลบางคอลัมน์ เช่น ชื่อเมือง Bangkok ชื่อลูกค้า Arlee และหมายเลขลูกค้า 2001 ลงในตารางลูกค้า ใช้คำสั่งดังนี้

```
INSERT INTO CUSTOMERSTAB(ADDRESS,CUSNAME,CUSNO)
    VALUES('Bangkok','Arlee',2001);
```

ผลของคำสั่งในตารางลูกค้า จะทำให้คอลัมน์ ADDRESS มีค่าเป็น Bangkok คอลัมน์ CUSNAME จะมีค่าเป็น Arlee คอลัมน์ CUSNO จะมีค่าเป็น 2001 ดังนี้

CUSNO	CUSNAME	SALENO	RATING	SALENO
2001	Arlee	1001		

จะเห็นว่าไม่ได้ใส่ค่าในคอลัมน์ RATING และ SALENO ไว้ ดังนั้นทั้งสองคอลัมน์นี้จะมีค่าเป็น NULL โดยอัตโนมัติ

1.2 คำสั่งการเพิ่มข้อมูลโดยการดึงกลุ่มข้อมูลด้วยคำสั่งค้นหาข้อมูล ในภาษา SQL สามารถใช้คำสั่ง INSERT ในการนำค่าหรือหาค่าจากตารางหนึ่งแล้วไปใส่ไว้ในอีกตารางหนึ่งได้ โดยได้ค่านั้นมาจากการสอบถามข้อมูล รูปแบบเป็นดังนี้

```
INSERT INTO <table name>[(column 1, column 2,...)]
    SELECT statement;
```

INSERT INTO	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเพิ่มข้อมูล
table name	ชื่อตารางที่จะเพิ่มข้อมูล
SELECT statement	ประโยคคำสั่ง SELECT ที่ต้องการข้อมูลอีกตารางหนึ่ง

ตัวอย่างที่ 12 ถ้าต้องการใส่ข้อมูลพนักงานลงในตาราง BANGKOKSTAFF โดยข้อมูลที่จะใส่ลงไปนั้นได้มาจากตารางพนักงานขายที่อาศัยอยู่ใน "Bangkok"

```
INSERT INTO BANGKOKSTAFF
    SELECT *
        FROM SALESTAB
        WHERE ADDRESS = 'Bangkok';
```

ผลของคำสั่งนี้จะทำให้ได้ข้อมูลพนักงานที่อยู่ในเมือง Bangkok (ADDRESS = 'Bangkok') ทั้งหมดไปใส่ไว้ในตาราง BANGKOKSTAFF โดยตาราง BANGKOKSTAFF ได้ถูกสร้างไว้แล้วด้วยคำสั่ง CREATE TABLE ในการสร้างตาราง BANGKOKSTAFF จะต้องสร้างให้มี 4 คอลัมน์และมีชนิดข้อมูลตรงกับคอลัมน์ของตารางพนักงานขาย (โดยไม่จำเป็นต้องมีชื่อคอลัมน์เหมือนกัน)

ตารางพนักงานขาย

SALENO	SALENAME	ADDRESS	SALECOM
1001	KANJANA	Rayong	0.15
1002	BENJAWAN	Bangkok	0.12
1004	KOMCHAN	Nonthaburi	0.14
1008	MITREE	Chianrai	0.13
1003	SUTI	Bangkok	0.11

ตาราง BANGKOKSTAFF ที่พนักงานขายอยู่ในเมือง Bangkok

SALENO	SALENAME	ADDRESS	SALECOM
1004	BENJAWAN	Bangkok	0.12
1003	SUTI	Bangkok	0.11

2. คำสั่ง UPDATE เป็นคำสั่งที่ใช้ปรับปรุงแก้ไขข้อมูลในตาราง ซึ่งการปรับปรุงข้อมูลอาจมีมากกว่า 1 คอลัมน์ในแถวทุกแถวที่มีเงื่อนไขสอดคล้องกับที่ระบุไว้หลักคำว่า WHERE

รูปแบบของคำสั่งปรับปรุงแถวข้อมูลมีดังนี้

```
UPDATE <table name>SET<column 1>
        [, column 2,...]=<expression |sunquery> [WHERE<condition>];
```

UPDATE	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการปรับปรุงข้อมูล
table name	ชื่อตารางที่ต้องการปรับปรุง
SET<column>	ชื่อคอลัมน์ที่ต้องการปรับปรุง
expression	ค่าข้อมูลที่ต้องการปรับปรุง
WHERE<condition>	เงื่อนไขในการปรับปรุง

ตัวอย่าง ที่ 13 ถ้าต้องการเปลี่ยนค่า RATING ของลูกค้าทั้งหมดในตารางลูกค้าให้เป็น 200 จะต้องป้อนคำสั่งดังนี้

```
UPDATE CUSTOMERSTAB
    SET RATING = 200;
```

ผลของคำสั่งจะทำให้คอลัมน์ RATING ของตารางลูกค้ามีค่าเป็น 200 ทุกแถวและเมื่อเข้าไปดูข้อมูลในตารางลูกค้าจะปรากฏข้อมูลดังนี้

ตารางลูกค้า (CUSTOMERSTAB)

CUSNO	CUSNAME	ADDRESS	RATING	SALENO
2001	Arlee	Bangkok	100	1001
2002	Tanachote	Pratum	200	1003
2003	Tawatchai	Puket	200	1002
2004	Amporni	Ubon	300	1002
2006	Surasit	Bangkok	100	1001
2008	Jintana	Puket	300	1007
2007	Siriwan	Pratum	100	1004

ตารางลูกค้าที่มีค่า RATING=200

CUSNO	CUSNAME	ADDRESS	RATING	SALENO
2001	Arlee	Bangkok	200	1001
2002	Tanachote	Pratum	200	1003
2003	Tawatchai	Puket	200	1002
2004	Amporni	Ubon	200	1002
2006	Surasit	Bangkok	200	1001
2008	Jintana	Puket	200	1007
2007	Siriwan	Pratum	200	1004

หากต้องการจะเปลี่ยนเฉพาะแถวใดแถวหนึ่งเท่านั้นก็สามารถทำได้ตามตัวอย่างดังนี้

ตัวอย่างที่ 14 ถ้าต้องการเปลี่ยนค่า RATING ให้กับลูกค้าทั้งหมด ทีมีหมายเลขประจำตัวพนักงานขาย (SALENO) เป็น 1001 ให้มีค่า RATING เป็น 200 จะต้องป้อนคำสั่งดังนี้

```
UPDATE CUSTOMERSTAB
    SET RATING = 200
    WHERE SALENO = 1001;
```

ผลของคำสั่งจะทำให้ตารางลูกค้าเดิมเปลี่ยนเป็นตารางใหม่ ในตารางใหม่นี้ลูกค้าทั้งหมดที่มีหมายเลขประจำตัวพนักงานขายเป็น 1001 จะมีค่า RATING เป็น 200 ดังนี้

ตารางลูกค้า (CUSTOMERSTAB)

CUSNO	CUSNAME	ADDRESS	RATING	SALENO
2001	Arlee	Bangkok	100	1001
2002	Tanachote	Pratum	200	1003
2003	Tawatchai	Puket	200	1002
2004	Amporni	Ubon	300	1002
2006	Surasit	Bangkok	100	1001
2008	Jintana	Puket	300	1007
2007	Siriwan	Pratum	100	1004

ตารางลูกค้าที่หมายเลขประจำตัวพนักงานขายเป็น 1001 ให้มีค่า RATING=200

CUSNO	CUSNAME	ADDRESS	RATING	SALENO
2001	Arlee	Bangkok	200	1001
2002	Tanachote	Pratum	200	1003
2003	Tawatchai	Puket	200	1002
2004	Amporni	Ubon	300	1002
2006	Surasit	Bangkok	200	1001
2008	Jintana	Puket	300	1007
2007	Siriwan	Pratum	100	1004

3. คำสั่ง DELETE เป็นคำสั่งที่ใช้ในการลบแถวข้อมูลทุกแถวที่มีในตารางตามเงื่อนไขที่สอดคล้องกับที่ระบุไว้หลัง WHERE มีรูปแบบดังนี้

DELETE FROM<table name>
[WHERE<condition>]:

DELETE FROM	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการลบข้อมูล
table name	ชื่อตารางที่ต้องการลบข้อมูล
WHERE<condition>	เงื่อไขในการลบข้อมูล

<u>ตัวอย่างที่ 15</u> ถ้าต้องการลบแบบมีเงื่อนไข เช่น ต้องการลบพนักงานขายชื่อSUTI ซึ่งมีหมายเลขพนักงาน (SALENO) = 1003 ออกจากตารางจะใช้คำสั่งดังนี้

```
DELETE FROM SALESTAB
    WHERE SALENO = 1003;
```

ผลของคำสั่งจากตารางพนักงานขายเดิมจะทำให้ได้ตารางใหม่ดังนี้

ตารางพนักงานขาย (SALESTAB)

SALENO	SALENAME	ADDRESS	SALECOM
1001	KANJANA	Rayong	0.15
1002	BENJAWAN	Bangkok	0.12
1004	KOMCHAN	Nonthaburi	0.14
1008	MITREE	Chianrai	0.13
1003	SUTI	Bangkok	0.11

SALENO	SALENAME	ADDRESS	SALECOM
1001	KANJANA	Rayong	0.15
1002	BENJAWAN	Bangkok	0.12
1004	KOMCHAN	Nonthaburi	0.14
1008	MITREE	Chianrai	0.13

โดยปกติแล้วการลบข้อมูลจะกระทำการลบเพียงบางแถวของตารางเท่านั้น การลบแถวต่าง ๆ ออกจากตารางด้วยคำสั่งในการปรับปรุงคือคำสั่ง DELETE คำสั่งนี้จะลบแถวทั้งแถวแต่ไม่สามารถลบค่าเพียงคอลัมน์ใดคอลัมน์หนึ่ง

ตัวอย่างที่ 16 ถ้าต้องการลบรายละเอียดทั้งหมดของตารางพนักงานขายจะต้องป้อนคสั่งต่อไปนี้

```
DELETE FROM SALESTAB;
```

ในตารางพนักงานขายก็จะว่างไม่มีค่าใด ๆ อยู่แต่ตารางยังปรากฏอยู่ ถ้าต้องการตารางออกไปจะใช้คำสั่ง DROP TABLE

ตัวอย่างที่ 17 ถ้าต้องการลบตาราง SALESTAB ออกจากฐานข้อมูลใช้คำสั่งดังนี้

```
DROP TABLE SALESTAB;
```

4. คำสั่ง SELECT เป็นคำสั่งการสอบถามข้อมูลหรือ “Query” โดย DBMS จะนำข้อมูลจากตารางในฐานข้อมูลมาแสดงทางจอภาพ การเรียกค้นข้อมูลจะเป็นไปตามเงื่อนไขที่ผู้ใช้ข้อมูลระบุ ในที่นี้จะนำเสนอ 3 รูปแบบดังนี้

1) การเรียกค้นดูทุกคอลัมน์ในตาราง มีรูปแบบดังนี้

```
SELECT *
        FROM<table name>;
```

SELECT *	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเรียกค้นข้อมูลทุกคอลัมน์
FROM	เป็นการกำหนดว่าให้เรียกดูข้อมูลได้จากตารางใดบ้าง
table name	ชื่อตารางที่ต้องการเรียกค้นข้อมูล

การเรียกดูข้อมูลสามารถเรียกดูได้มากกว่า 1 คอลัมน์ขึ้นไป โดยถ้ามีมากกว่า 1 คอลัมน์ แต่ละคอลัมน์จะต้องคั่นด้วยเครื่องหมายคอมม่า (,) และถ้าต้องการดูทุกคอลัมน์จะใช้เครื่องหมายดอกจัน (*) หลัก SELECT การใช้คำสั่ง SELECT จะใช้ควบคู่กับคำสั่ง FROM เสมอในการเลือกตาราง

การใช้คำสั่ง SELECT ในการเรียกค้นข้อมูลทุกคอลัมน์ในตารางจะใช้เครื่องหมายดอกจัน (*) ตามหลังคำสั่ง SELECT

2) การเรียกค้นข้อมูลเฉพาะคอลัมน์ใดๆ ในตารางและการเปลี่ยนลำดับคอลัมน์ การใช้คำสั่งมีรูปแบบดังนี้

```
SELECT <column 1, column 2,....>
        FROM<table name>;
```

SELECT	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเรียกค้นข้อมูลทุกคอลัมน์
column 1, column 2,...	คอลัมน์ที่ต้องการเรียกค้น
FROM	เป็นการกำหนดว่าให้เรียกดูข้อมูลได้จากตารางใดบ้าง
table name	ชื่อตารางที่ต้องการเรียกค้นข้อมูล

3) การใช้คำสั่ง SELECT กับ WHERE

```
SELECT <column 1, column 2,....>
        FROM<table name>
        [WHERE<condition>];
```

SELECT	เป็นคำสั่งที่ต้องมีทุกครั้งที่ต้องการเรียกค้นข้อมูลทุกคอลัมน์
column 1, column 2,...	คอลัมน์ที่ต้องการเรียกค้น
FROM	เป็นการกำหนดว่าให้เรียกดูข้อมูลได้จากตารางใดบ้าง
table name	ชื่อตารางที่ต้องการเรียกค้นข้อมูล
WHERE<condition>	ส่วนของคำสั่งที่บอกเงื่อนไขที่จะใช้ในการค้นหาข้อมูล

การใช้ WHERE ในคำสั่ง SELECT จะช่วยให้สามารถสืบค้นข้อมูลได้อย่างเจาะจงมากกว่า เช่น ถ้าใช้เฉพาะ SELECT อย่างเดียวจะได้ข้อมูลทั้งหมด ดังตัวอย่างที่ 18

ตัวอย่างที่ 18 ตาราง BIKES

NAME	FRAMESIZE	COMPOSITION	MIEBSRIDDENS	TYPE
TREK 2300	22.5	CARBON FIBER	3500	RACING
BURLEY	22	STEEL	2000	TANDEM
GIANT	19	STEEL	1500	COMMUTER
FUJI	20	STEEL	500	TOURING
CANNONDATE	22.5	ALUMINUM	3000	RACING

ถ้าต้องการจะดูเฉพาะข้อมูลของ "BURLEY" เท่านั้นเราจะต้องใช้คำสั่ง WHERE ดังนี้

```
SELECT * FROM BIKES
        WHERE NAME='BURLEY'
```

ผลลัพธ์

NAME	FRAMESIZE	COMPOSITION	MIEBSRIDDENS	TYPE
BURLEY	22	STEEL	2000	TANDEM

ตัวอย่าง คำสั่ง SELECT ในหลายรูปแบบ

(หมายเหตุ: ใช้คู่กับตัวอย่างงานฐานข้อมูล STUDENT)

กำหนดตาราง STUDENT TEACHER CLASS STUDENT_SUBJECT (วิทยากรกำหนดให้)

คำสั่ง SELECT

1. จงแสดงชื่อวิชา รหัสวิชา และหน่วยกิตของวิชาที่เปิดสอนอยู่ทั้งหมด

```
SELECT name, subject, credit
        FROM subject;
```

2. ในสถานศึกษาแห่งนี้ชั้นเรียนอยู่กี่ชั้น และชื่อว่าอะไรบ้าง

```
SELECT class
        FROM class
```

คำสั่ง ORDER BY

3. แสดงข้อมูลจากตาราง Teacher โดยเรียงลำดับตามชื่ออาจารย์ (น้อยไปหามาก) และรหัสวิชา (มากไปหาน้อย)

```
SELECT subjected, teacher
        FROM teacher
        ORDER BY teacher, subjected DESC;
```

คำสั่ง WHERE

4. แสดงข้อมูลของนักศึกษาทั้งหมดที่อยู่ห้อง A

```
SELECT *
        FROM student
        WHERE class='A';
```

คำสั่งฟังก์ชันที่ใช้คำนวณ

คำสั่ง AVG

5. หาคะแนนเฉลี่ยของนักศึกษาที่เรียนวิชาการออกแบบเว็บไซต์

```
SELECT AVG(score) AS ค่าเฉลี่ยของคะแนน
        FROM student_subject
        WHERE subjectid='I002';
```

คำสั่ง MAX หรือ MIN

6. หาคะแนนต่ำสุด และคะแนนเฉลี่ยของนักศึกษาที่เรียนวิชาการออกแบบเว็บไซต์

```
SELECT MIN(subjected), SUM(score)/COUNT(*)
        FROM student_subject
        WHERE subjectid='I002';
```

คำสั่ง SUM และ COUNT

7. หาคะแนนเฉลี่ยของนักศึกษาที่เรียนวิชา I002 (วิชาการออกแบบเว็บไซต์) โดยที่ไม่ใช้ Function AVG

```
SELECT SUM(score)/COUNT(*) AS คะแนนเฉลี่ย
        FROM student_subject
        WHERE subjectid='I002';
```

คำสั่ง โอเปอเรเตอร์เปรียบเทียบ =, <>, >, >=, < และ <=

8. แสดงข้อมูลรหัสนักศึกษา, รหัสวิชา, เกรด และคะแนนของนักศึกษาทุกคนที่ได้คะแนนมากกว่า 80 คะแนน

```
SELECT studentid, subjected, grade, score
        FROM student_subject
        WHERE score>80;
```

คำสั่ง โอเปอเรเตอร์ทางคณิตศาสตร์ + - * /

9. จงหาค่ากึ่งกลางพิสัยของคะแนนในวิชา P001

```
SELECT MIN(subjected), (MAX(score)+MIN(score))/2
        FROM student_subject
        WHERE subjectid='P001';
```

คำสั่ง โอเปอเรเตอร์ทางตรรกะ AND, OR, UNION

คำสั่ง AND

10. แสดงรหัสนักศึกษา รหัสวิชา และคะแนนของนักศึกษาที่ได้คะแนนระหว่าง 60-70

```
SELECT studentid, subjected, score
        FROM student_subject
        WHERE score>=60 AND score<=70;
```

คำสั่ง UNION

11. แสดงรหัสและชื่อนักศึกษาที่อยู่ห้อง C ร่วมกับรหัสและชื่อวิชาที่มี 2 หน่วยกิต

```
SELECT studentid, name
        FROM student
        WHERE class= 'C'
        UNION
            SELECT studentid, name
                FROM subject
                WHERE credit=2;
```

คำสั่งโอเปอเรเตอร์ที่มีลักษณะเฉพาะตัวไม่สามารถจัดกลุ่มได้ DISTINCT, IN, LIKE, BETWEEN...AND, NOT, IS, NULL, EXISTS, ANY, ALL

คำสั่ง IS NOT NULL

12. หาค่าเฉลี่ยของคะแนนในวิชา P001 โดยไม่ใช้ฟังก์ชั่น AVG

```
SELECT SUM(score)/COUNT(*) AS ค่าเฉลี่ยของคะแนน
        FROM student_subject
        WHERE subjectid='P001' AND score IS NOT NULL;
```

คำสั่ง DISTINCT

13. แสดงรายชื่อชมรมทั้งหมดที่มีนักศึกษาสังกัดอยู่

```
SELECT DISTINCT club
        FROM club;
```

คำสั่ง BETWEEN...AND

14. แสดงรหัสนักศึกษา รหัสวิชา และคะแนนของนักศึกษาที่ได้คะแนนระหว่าง 60-70

```
SELECT studentid, subjected, score
        FROM student_subject
        WHERE score BETWEEN 60 AND 70;
```

คำสั่ง LIKE

15. แสดงชื่อนักศึกษาที่ขึ้นต้นด้วยอักษร ก และอยู่ห้อง A, B หรือ C

```
SELECT name, class
        FROM student
        WHERE name LIKE 'ก%' AND class IN ('A','B','C');
```

คำสั่งคิวรีข้อมูลเป็นกลุ่มด้วย GROUP BY

16. หาจำนวนนักศึกษาในแต่ละชั้นเรียน

```
SELECT class, COUNT(*) AS จำนวนนักศึกษา
        FROM student
        GROUP BY class;
```

17. หาจำนวนนักศึกษาโดยแยกตามรหัสวิชาและเกรดที่ได้

```
SELECT subjected, grade, COUNT(*)
        FROM student_subjcet
        WHERE grade IS NOT NULL
        GROUP BY subjcetid, grade;
```

18. แสดงรายชื่อชมรมทั้งหมดที่นักศึกษาสังกัดอยู่

```
SELECT club
        FROM club
        GROUP BY club;
```

คำสั่งเลือกกลุ่มข้อมูลโดยใช้ HAVING กำหนดเงื่อนไข

19. แสดงรหัสวิชาและจำนวนอาจารย์ผู้สอนในวิชาที่มีอาจารย์ผู้สอนตั้งแต่ 2 คนขึ้นไป

```
SELECT subjected, COUNT(*)
        FROM teacher
        GROUP BY subjcetid
        HAVE COUNT(*)>=2;
```

ค้นข้อมูลทีละสองตารางขึ้นไป

คำสั่งการคิวรีข้อมูลหลายตารางผ่านการ JOIN

20. แสดงรหัสวิชา ชื่อวิชา และอาจารย์ผู้สอนในวิชานั้น ๆ ทั้งหมด

```
SELECT subject.subjectid, name, teacher
        FROM subject, teacher
        WHERE subject.subjectid=teacher.subjectid;
```

คำสั่งการ JOIN ตารางและกำหนดเงื่อนไขเพื่อเลือกข้อมูลบางแถว

21. อาจารย์ที่ปรึกษาของนักศึกษาชื่อ "ปักษา ระเริงชล" มีชื่อว่าอะไร

```
SELECT name, class.class, advisor
        FROM class, student
        WHERE student.class=class.class AND name= 'ปักษา ระเริงชล';
```

คำสั่งกำหนดชื่อเล่นเพื่อใช้อ้างถึงตาราง

22. อาจารย์ "มานะ มีวินัย" ใช้หนังสือ "Hotmail & MSN" สอนวิชาใด

```
SELECT subjectid, a.teacher, b.textbook
        FROM teacher AS a,teacher_textbook AS b
        WHERE a.teacher-b.teacher AND b.teacher='มานะ มีวินัย'
            AND textbook='Hotmail & MSN';
```

คำสั่งแสดงแถวที่ไม่ตรงกับเงื่อนไขด้วยวิธี Outer Jon (LEFT JOIN หรือ RIGHT JOIN)

23. แสดงรหัสวิชาและชื่อวิชาทั้งหมดพร้อมทั้งชื่ออาจารย์ผู้สอนของวิชานั้น ๆ ด้วย (ถ้ามี)

```
SELECT a.subjectid, name, teacher
        FROM subject AS a LEFT JOIN teacher AS b on a.subjectid=b.subjectid;
```

คำสั่ง SELECT ซ้อนกันหลายระดับ (Subqueries)

24. แสดงรหัสนักศึกษาของนักศึกษาซึ่งอยู่ในชมรมที่มีจำนวนสมาชิกเท่ากับ 3 คน

```
SELECT studentid,club
        FROM club
        WHERE club IN
        (SELECT club
            FROM club
            GROUP BY club
            HAVING COUNT(*)=3);
```

คำสั่งการใช้ ANY (ใช้ร่วมกับ = > <)

25. รายชื่ออาจารย์ที่สอนวิชาเดียวกับอาจารย์สิริ ศรีสมร

```
SELECT subjected,teacher
        FROM teacher
        WHERE subjected=ANY
        (SELECT subjectid
                FROM teacher
                  WHERE teacher='สิริ ศรีสมร');
```

26. หาชั้นเรียนที่มีจำนวนนักศึกษามากที่สุด

```
SELECT class, COUNT(*)
        FROM student
        GROUP BY class
        HAVING COUNT(*) >= ALL
        (SELECT COUNT(*)
                FROM student
                GROUP BY class);
```

27. นักศึกษาที่ได้คะแนนวิชา I001 เป็นอันดับที่ 2

```
SELECT studentid,score
        FROM student_subject
        WHERE subjectid='I001' AND score=
        (SELECT MAX(score)
                FROM student_subject
                WHERE subjectid='I001' AND score<
                (SELECT MAX(score)
                        FROM student_subjcet
                        WHERE subjectid='I001'));
```

28. นักศึกษาคนใดได้คะแนน P001 เท่ากันบ้าง

```
SELECT studentid, subjectid, score
        FROM student_subject AS s1
        WHERE subjectid='P001' AND score=ANY
        (SELECT score
                FROM student_subject AS s2
                WHERE subjectid='P001' AND s1.studented<>s2.studentid)
                ORDER BY score;
```

29. นักศึกษาที่ได้คะแนนวิชา I001 เป็นอันดับที่ 2

```
SELECT studentid, score
        FROM student_subject AS s1
        WHERE subjectid='I001' AND 1=
        (SELECT COUNT(*)
                FROM student_subject AS s2
                WHERE subjectid='i001' AND s1.score<s2.score);
```

30. นักศึกษาคนใดบ้างที่ไม่มีชมรมสังกัด

```
SELECT *
        FROM student AS s
        WHERE NOT EXISTS
        (SELECT studentid
                FROM club AS c
                WHERE s.studentid=c.studentid);
```

คำสั่งที่ใช้ควบคุมระบบฐานข้อมูล (Data Control Language : DCL)

คำสั่งในกลุ่ม Data Control Language (DCL) เป็นคำสั่งที่เกี่ยวข้องกับการกำหนดสิทธิของผู้ใช้ในการเข้าถึงทรัพยากรของระบบฐานข้อมูล (เช่น ตาราง) โดยผู้ดูแลระบบฐานข้อมูลจะใช้คำสั่งในกลุ่มนี้กำหนดสิทธิให้กับผู้ใช้แต่ละคน หรือผู้ใช้จะกำหนดสิทธิเพื่ออนุญาตให้ผู้อื่นมาใช้ทรัพยากรที่ตนเองเป็นเจ้าของก็ได้ คำสั่งที่ใช้กำหนดสิทธิในการใช้งานมี 2 คำสั่ง ได้แก่

คำสั่ง GRANT เป็นคำสั่งที่ใช้กำหนดสิทธิให้กับผู้ใช้คนอื่นเพื่อให้สามารถใช้งานทรัพยากรที่จำเป็นได้

คำสั่ง REVOKE เป็นคำสั่งที่ใช้ยกเลิกหรือเรียกคืนสิทธิที่เคยให้ไว้ ทำให้ผู้ใช้ที่ถูกยกเลิกสิทธิไม่สามารถใช้งานทรัพยากรได้อีกต่อไป

หมายเหตุ คำสั่ง GRANT และ REVOKE จะใช้ได้ในระบบฐานข้อมูลที่รองรับผู้ใช้งานหลายคนเท่านั้น สำหรับระบบฐานข้อมูลเล็ก ๆ ที่ใช้บนเครื่องคอมพิวเตอร์ทั่วๆ ไป เช่น MS ACCESS จะไม่สามารถใช้งานคำสั่งกลุ่มนี้ได้

ยังมีคำสั่งที่ใช้ควบคุมระบบฐานข้อมูลอยู่อีกกลุ่มหนึ่ง คำสั่งในกลุ่มนี้เกี่ยวข้องกับการทำงานของ Transaction ซึ่งมีลักษณะเฉพาะก็คือ การทำงานใด ๆ ที่อยู่ใน Transaction เดียวกันถ้ากระทำสำเร็จจะต้องสำเร็จทั้งหมด ไม่เช่นนั้นก็ต้องยกเลิกการกระทำที่เกิดขึ้นทั้งหมดเช่นกัน

คำสั่งที่ใช้ควบคุมการทำงาน Transaction มี 2 คำสั่ง ดังนี้

คำสั่ง COMMIT เป็นคำสั่งที่ใช้ยืนยันการทำงานหลักจากที่ Transaction หนึ่ง ๆ ทำงานเสร็จแล้ว มีผลให้เกิดการเปลี่ยนแปลงขึ้นจริง ๆ ในฐานข้อมูล

คำสั่ง ROLLBACK ใช้ในกรณีตรงข้ามกับคำสั่ง COMMIT โดยถ้า Transaction ใดทำงานไม่สำเร็จเราก็จะใช้คำสั่ง ROLLBACK เพื่อยกเลิกการกระทำทั้งหมดที่เกิดขึ้นแทน

แบบฝึกหัดท้ายบท

1-5 CREATE, INSERT, UPDATE, DELETE, 6-20 SELECT

1. จงสร้างตาราง ดังต่อไปนี้

Studentid	Name	Birthday	class

สร้างตาราง Student และกำหนด Primary Key

Class	advisor

สร้างตาราง Class และกำหนด Primary Key

Studentid	club

สร้างตาราง club และกำหนด primary Key และ Foreign Key

Studentid	hobby

สร้างตาราง hobby และกำหนด Primary Key และ Foreign Key

Subjectid	Name	Credit

สร้างตาราง subject และกำหนด Primary Key

Subjectid	teacher

สร้างตาราง teacher และกำหนด Primary Key และ Foreign Key

Subjectid	textbook

สร้างตาราง textbook และกำหนด Primary Key

Teacher	textbook

สร้างตาราง teacher_textbook และกำหนด Primary Key และ Foreign Key

Studentid	Subjectid	Grade	Score	Term

สร้างตาราง student_subject และกำหนด Primary Key และ Foreign Key

2. จง Update ข้อมูลเมื่อมีการเปิดสอนวิชาการเขียนโปรแกรมเบื้องต้นขึ้นมา โดยวิชาดังกล่าวมีรหัสวิชาเป็น P001 และมีหน่วยกิต
3. ถ้านักศึกษาที่มีรหัส 4600001 ลงทะเบียนเรียนวิชาการเขียนโปรแกรมเบื้องต้นในภาคเรียนที่ 1 ปีการศึกษา 2546 จงเก็บข้อมูลดังกล่าวเข้าไปในฐานข้อมูลให้ถูกต้อง
4. จงกำหนดให้นักศึกษาชื่อ "กานต์ มงคลลักขณา" เรียนอยู่ห้อง B
5. จงลบข้อมูลในตาราง student ออกทั้งหมดโดยไม่ต้องลบตาราง student ตามไปด้วย
6. จงแสดงรายชื่อของอาจารย์และวิชาที่อาจารย์คนนั้นเป็นผู้สอน
7. แสดงรายชื่อนักศึกษาและวันเกิดเรียงตามอายุจากมากไปหาน้อย
8. จงหารายชื่อหนังสือที่ใช้เรียนวิชา P001
9. แสดงรายชื่องานอดิเรกและจำนวนนักศึกษาที่มีงานอดิเรกนั้น เรียงลำดับตามจำนวนนักศึกษาจากมากไปหาน้อย
10. วิชาใดที่มีนักศึกษาได้คะแนนต่างกันมากกว่า 4 เท่า
11. มีอาจารย์คนใดบ้างที่เลือกใช้หนังสือ "เทคนิคการดูแลกล้อง" ประกอบการสอน
12. จงหานักศึกษาที่มีนามสกุล "วารีวานิช"
13. จงแสดงชื่อนักศึกษา และวันเกิดของนักศึกษาที่เกิดในเดือนมกราคม ค.ศ.1984
14. อาจารย์ที่ปรึกษาของห้อง D คือใคร

15. แสดงรายชื่อหนังสือที่ใช้เรียนในวิชา P002 และ S001
16. จงหาว่ามีนักศึกษาคนใดบ้างที่ได้คะแนนน้อยกว่า 30 คะแนนในวิชา I001, I002 และ P001
17. แสดงข้อมูลของนักศึกษาทุกคนในวิชาที่ยังไม่ได้สอบปลายภาค
18. หารายชื่อชมรมที่มีสมาชิกเท่ากับ 3 คน
19. จงแสดงรายละเอียดของวิชาที่สอนเกี่ยวกับศาสตร์คอมพิวเตอร์มาทั้งหมด (ไม่รวมวิชาที่สอนวิธีการใช้งานโปรแกรม)
20. จงหาคะแนนเฉลี่ยของนักศึกษาที่เรียนวิชาการออกแบบเว็บไซต์