ฐานข้อมูล MySQL

MySQL จัดอยู่ในกลุ่มของระบบบริหารจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS : Relational Database Management System) ชนิดหนึ่ง เป็นที่นิยมใช้กันมากในปัจจุบัน เหมาะกับองค์กรขนาดกลางที่มีข้อมูลไม่มากนัก ในบทนี้จะกล่าวถึงคำสอบถามหรือคำสั่ง (Query) พื้นฐานของ MySQL ที่ใช้งานบ่อยครั้ง เช่น 1) การจัดการฐานข้อมูล 2) การจัดการตาราง และ 3) การจัดการกับขอบเขตข้อมูล เป็นต้น ซึ่งทั้ง 3 ส่วนนี้เป็นองค์ประกอบที่สำคัญของฐานข้อมูล ในส่วนของขอบเขตข้อมูลก็จำเป็นต้องกำหนดชนิดของขอบเขตข้อมูลให้ถูกต้องกับลักษณะการใช้งานรวมถึงการกำหนดคุณสมบัติอื่นๆ จะเห็นได้ว่ามีรายละเอียดปลีกย่อยจำนวนมากที่จำเป็นต้องกำหนดให้ถูกต้องในแต่ละส่วน ทำให้ผู้ใช้งานอาจจะไม่สะดวกในการใช้งาน ดังนั้นในปัจจุบันก็มีการพัฒนาโปรแกรมเพื่ออำนวยความสะดวกสำหรับการบริหารจัดการฐานข้อมูล MySQL ก็คือ โปรแกรม phpMyAdmin

1 คำสอบถามหรือคำสั่งพื้นฐานของ MySQL

การศึกษา MySQL ควรเริ่มต้นจากการใช้คำสอบถามหรือคำสั่งพื้นฐานให้เกิดความคุ้นเคยเสียก่อน แม้ในการใช้งานจริงจะส่งคำสอบถามหรือคำสั่งผ่านฟังก์ชันต่างๆ ของภาษา PHP ก็ตาม แต่หากไม่มีพื้นฐานในการใช้งานกับฐานข้อมูลโดยตรงจะทำให้ไม่เข้าใจขั้นตอน และหลักการทำงานที่แท้จริง ลักษณะคำสอบถามหรือคำสั่งของ MySQL จะใช้รูปแบบคำสอบถามหรือคำสั่งตามมาตรฐานของภาษา SQL (Structure Query Language)

1.1 SHOW DATABASES;

SHOW DATABASES เป็นคำสอบถามเพื่อเรียกดูรายชื่อฐานข้อมูลทั้งหมดที่มีอยู่ภายในระบบฐานข้อมูล MySQL มีตัวอย่างดังนี้

ตัวอย่างที่ 1 การใช้คำสอบถามเพื่อเรียกดูรายชื่อฐานข้อมูลทั้งหมด

```
mysql> SHOW DATABASES;
        ----------------------------------------
        |        Database                |
        ----------------------------------------
        |        mysql                   |
        |        test                    |
        |        parinya_db              |
```

1.2 USE;

USE เป็นคำสั่งที่ใช้สำหรับเลือกใช้งานฐานข้อมูล มีตัวอย่างดังนี้

รูปแบบ

```
USE ชื่อฐานข้อมูล;
```

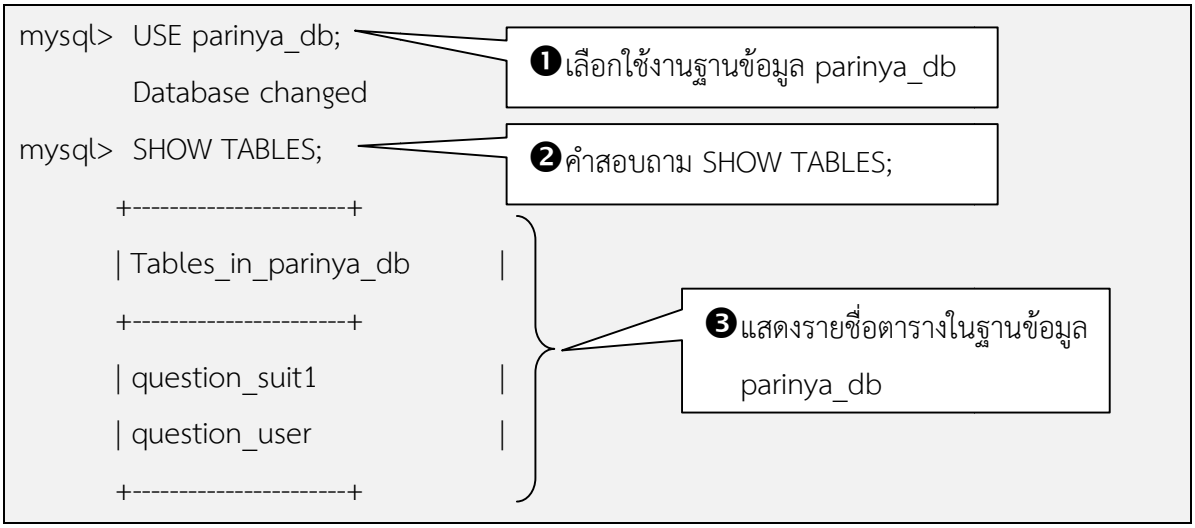
ตัวอย่างที่ 2 ตัวอย่างการกาเลือกใช้งานฐานข้อมูลชื่อ parinya_db

```
mysql>USE parinya_db;
```

1.3 SHOW TABLES;

SHOW TABLES เป็นคำสอบถามเพื่อเรียกดูรายชื่อตารางทั้งหมดที่มีอยู่ในฐานข้อมูลที่ถูกเลือกไว้ก่อนหน้านี้ มีตัวอย่างดังนี้

ตัวอย่างที่ 3 การใช้คำสอบถามคำสอบถาม SHOW TABLES;



1.4 DESCRIBE หรือ DESCDESC

DESCRIBE หรือ DESC เป็นคำสอบถามที่ใช้สำหรับดูโครงสร้างของขอบเขตข้อมูล (Field) และข้อมูล ที่มีอยู่ในตารางที่ระบุ มีรูปแบบการใช้งานดังนี้

รูปแบบ

```
DESCRIBE ชื่อตาราง;
หรือ
DESC ชื่อตาราง;
```

ตัวอย่างที่ 4 การใช้การใช้ คำสอบถาม DESCRIBE หรือ DESC

```
mysql> DESCRIBE question_user;
+------------+------------+------+-----+---------+----------------+
| Field      | Type       | Null | Key | Default | Extra          |
+------------+------------+------+-----+---------+----------------+
| id         | int(11)    | NO   | PRI | NULL    | auto_increment |
| user_id    | varchar(30)| NO   |     | NULL    |                |
| password   | varchar(30)| NO   |     | NULL    |                |
| before_name| varchar(20)| YES  |     | NULL    |                |
| name       | varchar(60)| YES  |     | NULL    |                |
| lastname   | varchar(60)| YES  |     | NULL    |                |
| privilege  | varchar(1) | NO   |     | 3       |                |
+------------+------------+------+-----+---------+----------------+
```

13.1.5 DROP

DROP เป็นคำสั่งที่ใช้สำหรับลบฐานข้อมูล หรือตาราง มีรูปแบบดังนี้

รูปแบบ

```
DROP ชื่อตาราง; หรือ ชื่อฐานข้อมูล;
```

หากลบฐานข้อมูลออกจะทำให้ทุกๆ ตารางที่อยู่ในฐานข้อมูลนั้นๆ ถูกลบออกไปด้วย ตัวอย่างดังนี้

ตัวอย่างที่ 13.5 การใช้คำสั่ง DROP เพื่อลบฐานข้อมูล parinya_db

```
mysql>DROP parinya_db;
```

13.1.6 QUIT หรือ EXIT

QUIT หรือ EXIT เป็นคำสั่งที่ใช้สำหรับออกจากโปรแกรม MySQL ตัวอย่างดังนี้

ตัวอย่างที่ 13.6 การใช้คำสั่ง QUIT หรือ EXIT เพื่อออกจากระบบฐานข้อมูล MySQL

```
mysql>EXIT; หรือ QUIT
```

2 องค์ประกอบของฐานข้อมูลานข้อมูล

โดยทั่วไปแล้ว องค์ประกอบหลักของระบบฐานข้อมูล MySQL จะมีลักษณะดังต่อไปนี้

2.1 ฐานข้อมูล (Database) ในการที่จะจัดเก็บข้อมูลใน MySQL ได้ จะต้องเริ่มต้นที่การสร้าง "ฐานข้อมูล" ก่อน ฐานข้อมูลนี้อาจเปรียบได้กับโฟลเดอร์ที่อยู่ในไดเรกทอรี

13.2.2 ตาราง (Table) คือ ตาราง มีลักษณะการแยกจัดเก็บข้อมูล ในแต่ละเรื่องออกจากกัน เช่น ตารางพนักงาน ตารางสินค้า ตารางลูกค้า เป็นต้น ทั้งนี้ตารางอาจเปรียบได้กับไฟล์ต่างๆ ที่อยู่ในโฟลเดอร์นั่นเอง และฐานข้อมูลหนึ่งๆ จะมีตารางจำนวนเท่าไหร่ก็ได้ (ขึ้นอยู่กับการวิเคราะห์ความสัมพันธ์ของข้อมูลภายในระบบที่จะพัฒนา) เช่นเดียวกับโฟลเดอร์หนึ่งๆ จะมีไฟล์อยู่จำนวนเท่าไรก็ได้

13.2.3 ขอบเขตข้อมูล ภายในตารางจะประกอบไปด้วยขอบเขตข้อมูลต่างๆ เพื่อกำหนดว่าตารางนั้นจะเก็บข้อมูลอะไรบ้าง โดยปกติแล้วข้อมูลที่จัดเก็บในตารางเดียวกัน จะต้องมีความเกี่ยวข้องอย่างใดอย่างหนึ่งต่อกัน เช่น หากเป็นตารางที่จัดเก็บข้อมูลของพนักงาน ก็อาจประกอบไปด้วยขอบเขตข้อมูลรหัส ชื่อ ที่อยู่ ตำแหน่ง และเงินเดือน เป็นต้น นอกจากนี้แล้ว แต่ละขอบเขตข้อมูลจะต้องระบุชนิดข้อมูลให้สัมพันธ์กับข้อมูลที่จะจัดเก็บในขอบเขตข้อมูลนั้นด้วย

3 ชนิดข้อมูลสำหรับกำหนดให้กับสำหรับกำหนดให้กับ ขอบเขตข้อมูล

เนื่องจากขอบเขตข้อมูลใช้สำหรับเก็บข้อมูลในแต่ละเรื่อง โดยข้อมูลเหล่านี้อาจจะมีหลายชนิดแตกต่างกันออกไป เช่น ชื่อ หรือ ที่อยู่เป็นข้อมูลชนิดข้อความ และเงินเดือนมักจะเป็นชนิดตัวเลข และถ้าเป็นวันเกิดก็มักจะเป็นข้อมูลชนิดวันเวลา เป็นต้น ทั้งนี้เนื่องจากวิธีในการจัดการกับข้อมูลเหล่านี้ จะต่างกันออกไปตามลักษณะของข้อมูล ดังนั้นทุกขอบเขตข้อมูลของตาราง จะต้องกำหนดชนิดข้อมูลให้กับขอบเขตข้อมูลเสมอ สำหรับ MySQL ได้แบ่งชนิดข้อมูลของขอบเขตข้อมูล ดังนี้

3.1 ข้อมูลชนิดตัวเลข

ข้อมูลชนิดตัวเลขนั้นมีอยู่หลายชนิดตามขนาดของตัวเลข และในแต่ละชนิดสามารถกำหนดเพิ่มเติมเป็นจำนวนบวกหรือลบ (SIGNED) หรือเป็นได้เฉพาะจำนวนบวกอย่าง**เดียว** (UNSIGNED) ดังตารางต่อไปนี้

ตารางที่ 1 แสดงชนิดข้อมูลของ MySQL ชนิดตัวเลข

แสดงชนิดข้อมูลของ	ชนิด ไบต์		ค่าต่ำสุด	ค่าสูงสุด
TINYINT	1	SIGNED	-128	127
		UNSIGNED	0	255
SMALLINT	2	SIGNED	-32768	32767
		UNSIGNED	0	65535
MEDLUMINT	3	SIGNED	-8388608	8388067
		UNSIGNED	0	16777215
INT	4	SIGNED	-2147483648	2147483647
		UNSIGNED	0	4294967295
BIGINT	8	SIGNED	-9223372036854775808	9223372036854775807
		UNSIGNED	0	18446744073709551615

ตารางที่ 1 (ต่อ)

ชนิด	ไบต์	รองรับ	ค่าต่ำสุด	ค่าสูงสุด
FLOAT	4	SIGNED	-3.402823466E+38	-1.1754966E-38
		UNSIGNED	1.7976931348632157E+38	3.402823466+38
DOUBLE	8	SIGNED	-1.7976931348623157E+308	-2.2250738585072014E-308
		UNSIGNED	2.225073858072014E-308	1.7976931348623157E+308

3.2 ข้อมูลชนิดข้อคข้อความ ข้อมูลชนิดข้อคข้อความ

ที่กำหนดได้ ดังตารางที่ 13.2 ทั้งนี้ข้อมูลชนิด CHAR และ VARCHAR จะต้องระบุความยาวสูงสุดของข้อความลงไปด้วย กำหนดได้ไม่เกิน 255 เช่น CHAR(50) หรือ VARCHAR(250) เป็นต้น ส่วนชนิดอื่นๆ นอกจากนี้ ไม่จำเป็นต้องระบุค่าความยาวของข้อความ

ตารางที่ 2 แสดงชนิดข้อมูลของ แสดงชนิดข้อมูลของ MySQL ชนิดข้อความ

ชนิด	ค่าสูงสุด
CHAR (length)	255
VARCHAR (length)	255
YINYTEXT	255
TEXT	65535
MEDIUMTEXT	16777215
LONGTEXT	4294967295

13.3.3 ข้อมูลชนิด BLOB (Binary Large Object)

ข้อมูลชนิด BLOB มักใช้กับข้อมูลประเภทรูปภาพ หรือมัลติมีเดีย มีลักษณะเป็นข้อมูลแบบไบนารี มีรายละเอียดดังตาราง ที่ 3

ตารางที่ 3 แสดงชนิดข้อมูลของ แสดงชนิดข้อมูลของ MySQL ชนิด BLOB

ชนิด	ค่าสูงสุด
TINYBLOB	255
BLOB	65535
MEDIIUMBLOB	16777215
LONGBLOB	4294967295

3.4 ข้อมูลชนิด SET SETและ ENUM

ข้อมูลชนิด SET และ ENUM คือ กลุ่มของข้อมูลที่ทราบค่าที่แน่นอนแล้วจำนวนหนึ่ง ข้อมูลที่จะกำหนดให้กับขอบเขตข้อมูลชนิดนี้ได้ต้องเป็นข้อมูลที่เป็นสมาชิกของ SET หรือ ENUM เท่านั้น

โดยข้อมูลชนิด SET จะมีสมาชิกได้ไม่เกิน 64 ตัว และ ENUM มีสมาชิกได้ไม่เกิน 65,535 ตัว รูปแบบของขอบเขตข้อมูลประเภทนี้ ดังตารางที่ 4

ตารางที่ **ตารางที่ 4** แสดงชนิดข้อมูลของ MySQL ชนิด SET และ ENUM

ชนิด แสดงชนิดข้อมูลของ	รูปแบบ	ค่าสูงสุด
SET	SET ("member1", "member2",..., "member64")	64
ENUM	ENUM ("member1", "member2",..., "member65535")	65535

13.3.5 ข้อมูลชนิดวันเวลา

โดยปกติข้อมูลชนิดวันเวลาของ MySQL จะอยู่ในรูปแบบต่อไปนี้

YYYY-MM-DD หรือ Year-Month-Date

เช่น วันที่ 20 เดือนตุลาคม ปี ค.ศ. 2010 จะต้องกำหนดเป็น 2010-10-20

ตารางที่ 5 แสดงชนิดข้อมูลของ แสดงชนิดข้อมูลของ MySQL ชนิดวันเวลา

ชนิด	รายละเอียด	ค่าสูงสุด
DATE	เฉพาะข้อมูลวันเดือนปี	1000 01 01 ถึง 9999 12 31
TIME	เฉพาะข้อมูลเวลา	-838:59:59 ถึง 838:59:59
DATETIME	ทั้งวันเดือนปีและเวลา	1000-01-01 00:00:00 ถึง 9999-12-31 23-59-59

4 แอททริบิวต์สำหรับกำหนดให้กับขอบเขตข้อมูล

แอททริบิวต์ (Attribute) หรือแฟล็ก (Flag) คือ ข้อกำหนดเพิ่มเติมนอกเหนือจากชนิดข้อมูลที่กำหนดให้กับขอบเขตข้อมูล แอททริบิวต์จะช่วยให้จัดการข้อมูลได้สะดวกขึ้น และบางแอททริบิวต์ก็ยังช่วยตรวจสอบความถูกต้องของข้อมูลได้ด้วย แต่ต้องกำหนดให้สัมพันธ์กับชนิดข้อมูลของขอบเขตข้อมูล แอททริบิวต์ของขอบเขตข้อมูล ดังตารางที่ 13.6

ตารางที่ 6 แสดงแอททริบิวต์สำหรับกำหนดให้กับขอบเขตข้อมูล ใน MySQL

แอททริบิวต์	รายละเอียด
NOT NULL	ห้ามขอบเขตข้อมูลเป็นค่าว่าง (NULL) คือ จะต้องใส่ข้อมูลให้กับขอบเขตข้อมูลเสมอ นิยมใช้กำหนดเพื่อควบคุมความถูกต้องของตาราง
BINARY	ใช้กับข้อมูลชนิด CHAR หรือ VARCHAR โดยปกติแล้วการจัดเรียงข้อมูลชนิด CHAR หรือ VARCHAR จะเป็นแบบ case-sensitive (ตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่มีความหมายแตกต่างกัน) แต่หากระบุ "แอททริบิวต์" เป็น BINARY การสืบค้นจะไม่คำนึงว่าจะเป็นตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่

ตารางที่ 6 (ต่อ)

แอททริบิวต์	รายละเอียด
AUTO_INCREMENT	กำหนดให้ใส่ตัวเลขลงไปในขอบเขตข้อมูลแบบอัตโนมัติ ตัวเลขที่ใส่เข้าไปโดยปกติจะเริ่มจาก 1 แล้วเพิ่มค่าครั้งละ 1 ไปเรื่อยๆ ค่าที่โปรแกรมใส่ให้จะไม่สามารถเปลี่ยนแปลงแก้ไขได้ ทั้งนี้ขอบเขตข้อมูลที่จะกำหนดให้เป็นแบบ AUTO_INCREMEN ได้ ต้องกำหนดชนิดข้อมูลให้เป็นแบบเลขจำนวนเต็ม เช่น INT เป็นต้น และห้ามใช้ร่วมกับแอททริบิวต์ NULL แต่ส่วนมากนิยมใช้ร่วมกับ NOT NULL
DEFAULT	เป็นการกำหนดค่าให้กับขอบเขตข้อมูล ด้วยค่าใดค่าหนึ่งเอาไว้ล่วงหน้า มักเป็นค่าที่ใช้กันบ่อยๆ เช่น พนักงานส่วนใหญ่เป็นเพศหญิง ดังนั้นจึงอาจกำหนดค่า DEFAULT ให้แก่ขอบเขตข้อมูลเพศ เป็น 'หญิง' เป็นต้น
INDEX	ใช้ในการจัดเรียงลำดับข้อมูลจากน้อยไปหามาก หรือมากไปหาน้อย ข้อมูลที่เรียงลำดับจะช่วยให้ค้นหารวดเร็วขึ้น สามารถจัดเรียงลำดับข้อมูลได้มากกว่า 1 ขอบเขตข้อมูล เมื่อสร้าง INDEX โปรแกรมฐานข้อมูลจะสร้างไฟล์ INDEX แยกออกไปต่างหาก เนื่องจากเรื่อง INDEX ถือว่าเป็นเรื่องสำคัญอย่างหนึ่งของฐานข้อมูล
UNIQUE	เป็น INDEX ในอีกลักษณะหนึ่ง โดยมีข้อกำหนดที่สำคัญ คือ ห้ามมีข้อมูลซ้ำกันในขอบเขตข้อมูลเดียวกัน ใช้กับข้อมูลที่คาดว่าข้อมูลในแต่ละแถวจะไม่มีทางซ้ำกันได้เลย เช่น รหัสพนักงาน อีเมล์ เป็นต้น และถ้าใส่ข้อมูลซ้ำกับที่มีอยู่แล้ว โปรแกรมจะไม่ยอมรับข้อมูลนั้น ดังนั้นแอททริบิวต์ UNIQUE สามารถนำมาช่วยในการตรวจสอบความถูกต้องได้อีกทางหนึ่ง
PRIMARY KEY	เป็น INDEX ในอีกลักษณะหนึ่ง โดย PRIMARY KEY คือ ขอบเขตข้อมูลที่จะใช้เป็นขอบเขตข้อมูลหลักของตาราง แต่ PRIMARY KEY มีข้อกำหนดที่สำคัญคือขอบเขตข้อมูลที่จะใช้เป็น PRIMARY KEY ข้อมูลในขอบเขตข้อมูลนั้นจะต้องไม่ซ้ำกัน โดยทั่วไปทุกๆ ตารางจะต้องมีขอบเขตข้อมูลอย่างน้อย 1 ขอบเขตข้อมูล ที่กำหนดเป็น PRIMARY KEY

5 การสร้างฐานข้อมูล

ขั้นตอนการสร้างฐานข้อมูลเป็นขั้นตอนแรกสุดก่อนเริ่มขั้นตอนต่อๆ ไป การสร้างฐานข้อมูลมีรูปแบบดังนี้

รูปแบบ

```
CREATE DATABASE (IF NOT EXISTS) ชื่อฐานข้อมูล;
```

หมายเหตุ

IF NOT EXISTS ใช้ป้องกันการสร้างชื่อฐานข้อมูลซ้ำกัน ความหมายคือ ระบบจะไม่สร้างฐานข้อมูลใหม่ให้ หากชื่อฐานข้อมูลใหม่ไปตรงกับชื่อฐานข้อมูลเดิมที่มีอยู่ (จะระบุหรือไม่ก็ได้)

ตัวอย่างที่ 7 การใช้ CREATE DATCREATE DATABASE

เพื่อสร้างฐานข้อมูลใหม่ดังนี้ mysql> CREATE DATABASE guestbook;guestbook;

จากตัวอย่างที่ 13.7 เป็นการสร้างฐานข้อมูลชื่อ guestbook โดยฐานข้อมูลที่สร้างขึ้นนี้ใช้เพื่อเก็บตารางต่างๆ ที่เกี่ยวข้อง สามารถตรวจสอบได้ว่าฐานข้อมูลได้ถูกสร้างแล้วหรือไม่ โดยใช้คำสอบถาม SHOW DATABASES ก่อนหน้าที่ได้อธิบายไปแล้ว หรือใช้ IF NOT EXISTS ร่วมด้วย

6 การสร้างตาราง

การสร้างตารางใหม่จะใช้คำสั่งเพื่อกำหนดโครงสร้างของตาราง ตารางใหม่ที่จะสร้างนี้ จะถูกสร้างไว้ในฐานข้อมูลที่ได้เลือกใช้ไว้ก่อนหน้านี้ โดยมีกฎการตั้งชื่อตารางที่สำคัญ คือ อนุญาตให้ตั้งชื่อที่มีความยาวได้ไม่เกิน 64 ตัวอักษร และห้ามใช้อักขระ "/" หรือ "." ในชื่อตาราง เพราะอาจเกิดข้อผิดพลาดในการอ้างถึงตาราง ตารางจะมีโครงสร้างประกอบด้วย ชื่อตาราง ชื่อขอบเขตข้อมูล ชนิดข้อมูล ขนาด และแอททริบิวต์เพิ่มเติมอื่นๆ มีรายละเอียดดังนี้

ตารางที่ 7 ตัวอย่างโครงสร้างตารางตาราง user

ขอบเขตข้อมูล	ชนิด	ว่างเปล่า (null)	ค่าปริยาย	คีย์	หมายเหตุ
name	VARCHAR(40)	-	-	-	ชื่อผู้ใช้งาน
address	VARCHAR(40)	-	-	-	ที่อยู่
email	VARCHAR(40)	-	-	-	อีเมล์
url	VARCHAR(40)	-	-	-	ที่อยู่ URL
comment	TEXT	-	-	-	หมายเหตุ

ตัวอย่างที่ 7 ตัวอย่างการสร้างการสร้าง ตาราง user ในฐานข้อมูล guestbook

```
mysql>  USE guestbook;
Database changed
mysql>  CREATE TABLE user (
    ->    name VARCHAR (40),
    ->    address VARCHAR (40),
    ->    email  VARCHAR (40),
    ->    url VARCHAR (40),
    ->    comment TEXT);
```

จากตัวอย่างที่ 7 แสดง ตัวอย่างการสร้างตาราง user ในฐานข้อมูล guestbook ประกอบด้วย 5 ขอบเขตข้อมูล **ประกอบด้วย** name, address, email, url และ comment นอกจากนี้ตารางยังสามารถสร้างได้หลายวิธีและสามารถกำหนดแอททริบิวต์ให้กับขอบเขตข้อมูลได้มากกว่า 1 แอททริบิวต์ มีตัวอย่างดังนี้

ตารางที่ 8 ตัวอย่างโครงสร้างโครงสร้าง ตาราง employees

ขอบเขตข้อมูล	ชนิด	ว่างเปล่า (null)	ค่าปริยาย	คีย์	หมายเหตุ
emp_id	INT	NOT NULL	-	PK	รหัสพนักงาน
firstname	VARCHAR(30)	NOT NULL	-	-	ชื่อพนักงาน
lastname	VARCHAR(30)	NOT NULL	-	-	นามสกุลพนักงาน
position	VARCHAR(20)	-	-	-	ตำแหน่งงาน
salary	MEDIUMINT	-	-	-	เงินเดือน (UNSIGNED)
address	TEXT	-	-	-	ที่อยู่
email	VARCHAR(40)	NOT NULL	-	-	อีเมล์ (UNIQUE)
phone	VARCHAR(30)	NOT NULL	-	-	เบอร์โทรศัพท์ (UNIQUE)

ตัวอย่างที่ 8 การสร้างตารางและกำหนดแอททริบิวต์ ขอบเขตข้อมูล**มากกว่า** 1 แอททริบิวต์

```
mysql> CREATE TABLE emplCREATE TABLE employees
(    ->  emp_id      INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
     ->  firstname   VARCHAR (30) NOT  NULL,
     ->  lastname    VARCHAR (30) NOT  NULL,
     ->  position    VARCHAR (20),
     ->  salary      MEDIUMINT UNSIGNED,
     ->  address     TEXT,
     ->  email       VARCHAR (40) NOT NULL UNIQUE,
     ->  phone       VARCHAR (30) NOT NULL UNIQUE) ;
```

6.1 การกำหนด PRIMARY KEYPRIMARY KEY

PRIMARY KEY คือ ขอบเขตข้อมูลที่ใช้เป็นขอบเขตข้อมูลหลักของตาราง โดยมีข้อกำหนดที่สำคัญ คือ ขอบเขตข้อมูลที่จะใช้เป็น PRIMARY KEY ต้องไม่มีข้อมูลที่ซ้ำกันเลย แต่กรณีที่ขอบเขตข้อมูลที่ต้องการใช้เป็น PRIMARY KEY มีโอกาสที่ข้อมูลจะซ้ำกัน ก็สามารถเลือกมากกว่า 1 ขอบเขตข้อมูล มาเป็น PRIMARY KEY ร่วมกันได้ เช่น สมมติตารางพนักงานประกอบด้วยขอบเขตข้อมูล "firstname" และ "lastname" หากคาดว่าชื่อของพนักงานอาจซ้ำกันได้ ดังนั้นจึงไม่สามารถใช้ขอบเขตข้อมูล "firstname" เป็น PRIMARY KEY เพียงขอบเขตข้อมูลเดียวได้ จึงอาจเลือกใช้ทั้ง firstname และ lastname เป็น PRIMARY KEY ร่วมกัน

กรณีที่ต้องใช้หลายขอบเขตข้อมูลเป็น PRIMARY KEY ร่วมกัน จึงไม่สามารถระบุแอททริบิวต์ PRIMARY KEY แบบแยกขอบเขตข้อมูลได้ จะต้องกำหนดในรูปแบบของฟังก์ชันรูปแบบดังนี้

รูปแบบ

```
PRIMARY KEY (column1, column2,...)
```

โดยการระบุต่อจากขอบเขตข้อมูลสุดท้ายของตารางตามรูปแบบที่ได้กล่าวมาแล้ว เช่น

ตัวอย่างที่ 9 การระบุ PRIMARY KEY ให้กับขอบเขตข้อมูลของตาราง employees

```
mysql>CREATE TABLE employees (
    -> firstname   VARCHAR (30) NOT NULL,
    -> lastname    VARCHAR (30) NOT NULL,
    -> position    VARCHAR (20),
    -> salary      MEDIUMINT UNSIGNED,
    -> address     TEXT,
    -> email       VARCHAR (40) NOT NULL UNIQUE,
    -> phone       VARCHAR (30) NOT NULL UNIQUE
    -> PRIMARY KEY (firstname, lastname));
```

จากตัวอย่างที่ 13.9 แสดงตัวอย่างการสร้างตาราง โดยระบุ PRIMARY KEY ให้กับขอบเขตข้อมูลของตาราง employees มากกว่า 1 ขอบเขตข้อมูล จากตัวอย่างกำหนดให้ขอบเขตข้อมูล firstname และ lastname เป็น PRIMARY KEY

13.6.2 ขอบเขตข้อมูลแบบ UNIQUE

จากข้อกำหนดของ UNIQUE คือ ข้อมูลในขอบเขตข้อมูลนั้นจะต้องไม่ซ้ำกันเลย สามารถสร้างขอบเขตข้อมูลแบบ UNIQUE ได้หลายขอบเขตข้อมูล และแต่ละขอบเขตข้อมูลจะเป็นอิสระจากกัน (ต่างจาก PRIMARY KEY ที่จะนำขอบเขตข้อมูลที่เป็น PRIMARY KEY มาพิจารณาร่วมกัน) นั่นคือ ไม่ว่าจะสร้างขอบเขตข้อมูลแบบ UNIQUE ขึ้นมากี่ขอบเขตข้อมูลก็ตาม ขอบเขตข้อมูลเหล่านั้น ในขอบเขตข้อมูลเดียวกันต้องไม่มีข้อมูลที่ซ้ำกันเลย ดังนั้นการสร้าง UNIQUE แบบระบุแยกขอบเขตข้อมูล หรือจะสร้างรวมด้วยฟังก์ชัน UNIQUE ( ) ก็ให้ผลที่ไม่แตกต่างกัน (เปลี่ยนแปลงคำสั่งจากตัวอย่างที่ 13.9 โดยใช้ร่วมกับฟังก์ชัน UNIQUE ( )) ตัวอย่างดังนี้

ตัวอย่างที่ 10 การระบุ UNIQUE ให้กับขอบเขตข้อมูลของตาราง employees

```
mysql>CREATE TABLE employees (
    -> firstname   VARCHAR (30) NOT NULL,
    -> lastname    VARCHAR (30) NOT NULL,
    -> position    VARCHAR (20),
    -> salary      MEDIUMINT UNSIGNED,
```

```
    -> address      TEXT,
    -> email        VARCHAR (40) NOT NULL,
    -> phone        VARCHAR (30) NOT NULL,
    -> UNIQUE (email, phone));
```

6.3 การสร้าง INDEXINDEX

INDEX เป็นการจัดเรียงลำดับข้อมูลเพื่อให้สามารถค้นหาข้อมูลได้รวดเร็วยิ่งขึ้นโดย MySQL จะสร้างไฟล์ INDEX แยกจากข้อมูลไว้ต่างหาก การเรียงลำดับข้อมูลสามารถเรียงได้ทั้งแบบจากน้อยไปหามาก หรือถ้าเป็นตัวอักษรคือ เรียงจาก A ไป Z โดยเขียนแทนด้วย ASC และเรียงจากมากไปหาน้อย หรือจาก Z ไป A เขียนแทนด้วย DESC สำหรับค่าโดยปริยาย คือ ASC ทั้งนี้สามารถสร้าง INDEX ให้กับขอบเขตข้อมูลใดขอบเขตข้อมูลหนึ่ง หรือกำหนดให้กับหลายๆ ขอบเขตข้อมูลได้ เช่นเดียวกับ UNIQUE หรือ PRIMARY KEY ด้วยการกำหนดแบบฟังก์ชัน ดังตัวอย่างที่ 13.11

ตัวอย่างที่ 11 การระบุ INDEX ให้กับขอบเขตข้อมูลของตาราง employees

```
mysql>CREATE TABLE employees (
    -> firstname    VARCHAR (30) NOT NULL,
    -> lastname     VARCHAR (30) NOT NULL,
    -> position     VARCHAR (20),
    -> salary       MEDIUMINT UNSIGNED,
    -> address      TEXT,
    -> email        VARCHAR (40) NOT NULL,
    -> phone        VARCHAR (30) NOT NULL,
    -> INDEX (email, phone));
```

6.4 การกำหนดหลายๆ แอททริบิวต์แบบฟังก์ชัน

หากต้องการกำหนดหลายๆ แอททริบิวต์แบบฟังก์ชัน ก็ให้วางต่อท้ายขอบเขตข้อมูลสุดท้าย โดยเรียงลำดับอย่างไรก็ได้ ตัวอย่างดังนี้

ตัวอย่างที่ 12 การกำหนดหลายๆ แอททริบิวต์แบบฟังก์ชัน

```
mysql>CHEATE TABLE employees (
    -> emp_id       INT NOT NULL ,
    -> firstname    VARCHAR (30) ,
    -> lastname     VARCHAR (30) ,
    -> position     VARCHAR (20),
    -> salary       MEDIUMINT UNSIGNED,
```

```
    -> address      TEXT,
    -> email        VARCHAR (40) NOT NULL,
    -> phone        VARCHAR (30) NOT NULL,
    -> PRIMARY KEY (emp_id) , INDEX (firstname, lastname) , UNIQUE (email, phone));
```

6.5 การสร้างตารางใหม่ใหม่จากตารางเดิมที่มีอยู่

การสร้างตาราง อาจสร้างโดยการคัดลอกมาจากตารางที่มีอยู่ก่อนหน้า ถ้าคัดลอกมาทั้งหมดก็จะทำให้ตารางใหม่มีลักษณะเหมือนกับตารางต้นแบบทุกประการ หรืออาจเลือกเพียงบางขอบเขตข้อมูลก็ได้ รูปแบบการคัดลอกตารางดังนี้

รูปแบบ

```
CREATE    TABLE ชื่อตารางใหม่
          SELECT ขอบเขตข้อมูล_1, ขอบเขตข้อมูล_2, ..., ขอบเขตข้อมูล_N
          FROM ชื่อตารางเดิม
          [WHERE  เงื่อนไข]
```

SELECT...FROM เป็นคำสอบถามสำหรับการคัดเลือกขอบเขตข้อมูลที่ต้องการ หากต้องการคัดลอกทุกขอบเขตข้อมูล อาจใช้เครื่องหมาย * แทนการระบุชื่อขอบเขตข้อมูล ตัวอย่างดังนี้

ตัวอย่างที่ 13 การใช้คำสั่งคำสั่ง CREATE TABLE ร่วมกับคำสอบถาม SELECT...FROM

```
mysql>CREATE TABLE employee2 SELECT * FROM employees;
```

ตาราง employees

emp_id	firstname	lastname	position	salary	address	email	phone
100	Parinya	Noidonprai	Manager	45000	Suratthani	pa@sru.ac.th	077355466
200	Jaidee	Ruktum	Programmer	17000	Suratthani	ja@sru.ac.th	077355123
300	Dennis	Robert	Programmer	19000	Suratthani	ha@sru.ac.th	077355443
400	John	Mackay	Finance	18000	Chumphon	jo@sru.ac.th	077355889

CREATE TABLE employee2 SELECT * FROM employees;

ตาราง employees2

emp_id	firstname	lastname	position	salary	address	email	phone
100	Parinya	Noidonprai	Manager	45000	Suratthani	pa@sru.ac.th	077355466
200	Jaidee	Ruktum	Programmer	17000	Suratthani	ja@sru.ac.th	077355123
300	Dennis	Robert	Programmer	19000	Suratthani	ha@sru.ac.th	077355443
400	John	Mackay	Finance	18000	Chumphon	jo@sru.ac.th	077355889

ภาพที่ 13.1 แสดงลักษณะการทำงานและผลลัพธ์เมื่อใช้ CREATE TABLE ร่วมกับ SELECT...FROM

จากตัวอย่างที่ 13 เป็นการใช้คำสั่ง CREATE TABLE ร่วมกับคำสอบถาม SELECT...FROM เพื่อสร้างตารางใหม่ ชื่อ employee2 โดยคัดเลือกขอบเขตข้อมูลและข้อมูลทั้งหมดจากตาราง employees แสดงผลลัพธ์ที่ได้ ดังภาพที่ 13.1

หากต้องการคัดเลือกเพียงบางขอบเขตข้อมูล ก็ให้ระบุชื่อขอบเขตข้อมูลลงไป ดังนี้

```
mysql>CREATE TABLE employee2 SELECT firstname, lastname FROM employees;
```

ปกติหากไม่ระบุเงื่อนไข ถ้าตารางต้นแบบมีข้อมูลอยู่แล้ว ข้อมูลนั้นก็จะถูกคัดลอกมาทั้งหมดทุกแถว แต่สามารถกำหนดเงื่อนไขเพื่อคัดลอกเฉพาะข้อมูลที่ต้องการได้ เช่น

ตัวอย่างที่ 14 ใช้การใช้ CREATE TABLE ร่วมกับ SELECT...FROM

```
แบบมีเงื่อนไแบบมีเงื่อนไข mysql> CREATE TABLE empCREATE TABLE em
    -> SELECT firstname, lastname, position FROM employees
    -> WHERE position = "Programmer";
```

ตาราง employees

emp_id	firstname	lastname	position	salary	address	email	phone
100	Parinya	Noidonprai	Manager	45000	Suratthani	pa@sru.ac.th	077355466
200	Jaidee	Ruktum	Programmer	17000	Suratthani	ja@sru.ac.th	077355123
300	Dennis	Robert	Programmer	19000	Suratthani	ha@sru.ac.th	077355443
400	John	Mackay	Finance	18000	Chumphon	jo@sru.ac.th	077355889

```
CREATE TABLE emp SELECT firstname, lastname, position FROM employees WHERE
position = "Programmer";
```

ตาราง emp

firstname	lastname	position
Jaidee	Ruktum	Programmer
Dennis	Robert	Programmer

ภาพที่ 13.2 แสดงลักษณะการทำงานและผลลัพธ์เมื่อใช้การใช้ CREATE TABLE ร่วมกับ SELECT ...FROM แบบมีเงื่อนไข

7 การแทรก ปรับปรุง ลบ และ การเรียกดูข้อมูลในตาราง

13.7.1 การแทรกข้อมูลในตารางด้วย INSERT

การแทรกข้อมูลลงในตารางด้วย INSERT มีรูปแบบดังนี้

รูปแบบ

```
INSERT INTO ชื่อตาราง [ขอบเขตข้อมูล1, ขอบเขตข้อมูล2,... )] VALUES (ค่า1, ค่า2, ...) ;
```

ชื่อขอบเขตข้อมูล จะกำหนดหรือไม่ก็ได้ โดยหากกำหนด จะต้องระบุค่าเรียงตามลำดับการจัดเรียงของขอบเขตข้อมูลหากไม่กำหนด จะต้องระบุค่า ให้ครบทุกขอบเขตข้อมูล และเรียงตามลำดับการจัดเรียงของขอบเขตข้อมูลในตาราง

ตัวอย่างที่ 15 การแทรกข้อมูลในตารางด้วย INSERT

```
mysql>INSERT INTO employees VALUES (123, "John", "john@example.com");
```

จากรูปแบบของ INSERT ที่กล่าวมานี้ สามารถใส่ข้อมูลได้เพียงครั้งละ 1 แถว หากจะแทรกข้อมูลมากกว่า 1 แถว ก็ต้องใช้ INSERT ซ้ำๆ ในทุกแถว อาจจะดูข้อมูลไม่สะดวกนัก แต่ใน MySQL ยังมีวิธีที่สามารถแทรกข้อมูลพร้อมกันครั้งละหลายๆ แถว โดย INSERT เพียงครั้งเดียว มีรูปแบบดังนี้

```
INSERT INTO ชื่อตาราง VALUES
(ค่าขอบเขตข้อมูล_1_ของแถวที่_1,ค่าของขอบเขตข้อมูล_2_ของแถวที่_1,...) ,
(ค่าขอบเขตข้อมูล_1_ของแถวที่_2,ค่าของขอบเขตข้อมูล_2_ของแถวที่_2,...) ,
....
....
....
(ค่าขอบเขตข้อมูล_N_ของแถวที่_N,ค่าของขอบเขตข้อมูล_N_ของแถวที่_N,...);
```

ตัวอย่างที่ 16 การแทรกข้อการแทรกข้อมูลในตารางด้วย INSERT แบบหลายระเบียน**ในครั้งเดียว**

```
mysql>INSERT INTO employees VALUES
    -> (123, "John", "Lee", "Finance", "20000", "Suratthani", "jo@sru.ac.th" , "077355890") ,
    -> (456, "Jim", "Murray", "Staff", "17000", "Suratthani", "jim@sru.ac.th" , "077355000") ,
    -> (789, "Jane", "Nuch", "Staff", "19000", "Chumphon", "jan@sru.ac.th" , "077355339");
```

7.2 การปรับปรุงข้อมูลข้อมูล ในตารางด้วย UPDATE

การปรับปรุงข้อมูลในตารางด้วย UPDATE สามารถทำการปรับปรุงข้อมูลได้หลายขอบเขตข้อมูล และหลายระเบียนภายในคำสั่ง 1 คำสั่ง ทั้งนี้ขึ้นอยู่กับเงื่อนไข ซึ่งกำหนดไว้ใน WHERE การปรับปรุงข้อมูลลงในตารางจะต้อง โดยใช้ UPDATE มีรูปแบบโดยทั่วไปดังนี้

รูปแบบ

```
UPDATE ชื่อตาราง SET ขอบเขตข้อมูล1=การคำนวณหรือค่าใหม่ที่ต้องการกำหนด1,
ขอบเขตข้อมูล2=การคำนวณหรือค่าใหม่ที่ต้องการกำหนด2,
... ,
... ,
```

```
ขอบเขตข้อมูลN=การคำนวณหรือค่าใหม่ที่ต้องการกำหนดN
[WHERE เงื่อนไข] [LIMIT กำหนดแถวเริ่มต้นและจำนวนแถวที่ต้องการปรับปรุง];
```

ตัวอย่างที่ 17 การปรับปรุงข้อมูลในตาราง employees ด้วย UPDATE

```
mysql>  UPDATE employees SET address='Chumphon',
   ->     position='President'
   ->     WHERE emp_id=001;
```

จากตัวอย่างเป็นการปรับปรุงข้อมูลในตาราง employees โดยกำหนดให้ขอบเขตข้อมูล address มีค่าเท่ากับ 'Chumphon' และ position มีค่าเท่ากับ 'President' โดยมีเงื่อนไขว่าจะปรับปรุงข้อมูลก็ต่อเมื่อ ขอบเขตข้อมูล emp_id มีค่าเท่ากับ 001 เท่านั้น

7.3 การลบข้อมูลในตารางด้วย ในตารางด้วย DELETE

การลบข้อมูลในตารางด้วย DELETE สามารถทำการลบได้หลายระเบียนภายใน หรือระเบียนเดียว ทั้งนี้ขึ้นอยู่กับการกำหนดเงื่อนไข WHERE การลบข้อมูลในตาราง โดยใช้ DELETE มีรูปแบบโดยทั่วไปดังนี้

รูปแบบ

```
DELETE FROM ชื่อตาราง [WHERE เงื่อนไข] [LIMIT แถวเริ่มต้นและจำนวนแถวที่จะลบ] ;
```

ตัวอย่างที่ 18 การลบข้อมูลในตารางด้วย

```
DELETE FROM employees WHERE emp_id
```

= 004; จากตัวอย่างเป็นการลบข้อมูลในตารางด้วย DELETE ในตาราง "employees" โดยมีเงื่อนไขการลบ คือ ขอบเขตข้อมูล "emp_id" จะต้องมีค่าเท่ากับ 004 จึงจะลบข้อมูล

7.4 การใช้คำสอบถาม เพื่อเรียกดูข้อมูลในตารางด้วย SELECT

การใช้คำสอบถามเพื่อเรียกดูข้อมูลในตารางด้วย SELECT สามารถระบุเงื่อนไขต่างๆ ได้ การใช้คำสอบถาม SELECT สามารถแบ่งตามลักษณะการใช้งาน มีรูปแบบดังนี้

1) การใช้คำสอบถามเพื่อเรียกดูข้อมูลในตารางด้วย SELECT อย่างง่าย

รูปแบบ

```
SELECT ขอบเขตข้อมูล FROM ชื่อตาราง;
```

ขอบเขตข้อมูล คือ ขอบเขตข้อมูลที่ต้องการสอบถาม หากมีหลายขอบเขตข้อมูลจะคั่นด้วยเครื่องหมาย Comma หากต้องการสอบถามข้อมูลทุกขอบเขตข้อมูล สามารถใช้เครื่องหมาย * แทน

ตัวอย่างที่ 19 การใช้ คำสอบถามเพื่อเรียกดูข้อมูลทั้งหมด จากตาราง employees s

```
mysql>SELECT * FROM employees;
```

ผลลัพธ์

```
-----------------------------------------------------------------------------------------------------------------
| emp_id | firstname | lastname   | position   | salary | address    | email         | phone      |
-----------------------------------------------------------------------------------------------------------------
|    100 | Parinya   | Noidonprai | Manager    |  45000 | Suratthani | pa@sru.ac.th | 077355466 |
|    200 | Jaidee    | Ruktum     | Programmer |  17000 | Suratthani | ja@sru.ac.th  | 077355123 |
|    300 | Dennis    | Robert     | Programmer |  19000 | Suratthani | ha@sru.ac.th | 077355443 |
|    400 | John      | Mackay     | Finance    |  18000 | Chumphon| jo@sru.ac.th  | 077355889 |
-----------------------------------------------------------------------------------------------------------------
4 rows in set (0.00 sec)
```

จากตัวอย่างที่ 13.19 เป็นการใช้คำสอบถามเพื่อเรียกดูข้อมูลทั้งหมดจากตาราง employees ผลลัพธ์ที่ได้ มี 4 แถว

ตัวอย่างที่ 20 การใช้คำสอบถาม เพื่อเรียกดูขอบเขตข้อมูลบางรายการ

```
mysql>SELECT emp_id, firstname, position, salary FROM employees;
```

ผลลัพธ์

```
-------------------------------------------------------
| emp_id  | firstname  | position     | salary    |
-------------------------------------------------------
|    100  | Parinya    | Manager      |  45000    |
|    200  | Jaidee     | Programmer   |  17000    |
|    300  | Dennis     | Programmer   |  19000    |
|    400  | John       | Finance      |  18000    |
-------------------------------------------------------
4 rows in set (0.00 sec)
```

2) การใช้คำสอบถามเพื่อเรียกดูข้อมูลในตารางด้วย SELECT แบบมีเงื่อนไข

รูปแบบ

```
SELECT ขอบเขตข้อมูล FROM ชื่อตาราง
[WHERE เงื่อนไข]
[GROUP BY รายชื่อขอบเขตข้อมูลที่ใช้ในการกำหนดกลุ่ม]
[HAVING เงื่อนไขที่ใช้ร่วมกับ GROUP BY]
[ORDER BY ชนิดของการเรียงลำดับ]
[LIMIT แถวเริ่มต้นและจำนวนแถวที่จะดูข้อมูล] ;
```

หมายเหตุ

WHERE ใช้สอบถามข้อมูลแบบมีเงื่อนไข เป็นการระบุค่าเฉพาะข้อมูลที่ต้องการสอบถาม อาจใช้เงื่อนไขเพื่อดึงข้อมูลบางแถวจากตารางก็ได้ โดยภายใน WHERE ประกอบด้วยส่วนสำคัญ ได้แก่ ชื่อขอบเขตข้อมูล ได้แก่ ชื่อขอบเขตข้อมูลโอเปอร์เรเตอร์ และข้อมูลเฉพาะที่ต้องการแสดงเป็นเงื่อนไขของขอบเขตข้อมูลที่ระบุ

ตารางที่ 9 แสดงโอเปอร์เรเตอร์ ที่ใช้งานร่วมกับ WHERE

โอเปอร์เรเตอร์	ชื่อ	ตัวอย่าง
=	เท่ากับ	column = 8000
>	มากกว่า	column > 66.00
<	น้อยกว่า	column < 66.00
>=	มากกว่าหรือเท่ากับ	column >= 66.00
<=	น้อยกว่าหรือเท่ากับ	column <= 66.00
!= หรือ <>	ไม่เท่ากับ	column !=0
IS NULL	ค่าว่าง	column IS NULL
IS NOT NULL	ไม่ใช่ค่าว่าง	column IS NOT NULL
BETWEEN	ค่าที่อยู่ระหว่าง	column BETWEEN 0 AND 66.00
IN	ค่าที่อยู่ในขอบเขตข้อมูลที่กำหนด	column IN ("mai", "mon")
NOT IN	ค่าที่ไม่ได้อยู่ในขอบเขตข้อมูลที่กำหนด	column NOT IN ("mai", "mon")
LIKE	เหมือนกับรูปแบบที่กำหนด	column LIKE ("mai%")
NOT LIKE	ไม่เหมือนกับรูปแบบที่กำหนด	column NOT LIKE ("mai%")
REGEXP	Regular Expression	column REGEXP

ตัวอย่างที่ 21 การใช้ คำสอบถามเพื่อเรียกดูข้อมูลในตารางด้วย SELECT แบบมีเงื่อนไข

```
mysql>SELECT * FROM employees WHERE address = "Suratthani";
```

ผลลัพธ์

```
-----------------------------------------------------------------------------------------------------------------
| emp_id   | firstname   | lastname   | position    | salary | address    | email          | phone       |
-----------------------------------------------------------------------------------------------------------------
|   100    | Parinya     | Noidonprai | Manager     |  45000 | Suratthani | pa@sru.ac.th   | 077355466   |
|   200    | Jaidee      | Ruktum     | Programmer  |  17000 | Suratthani | ja@sru.ac.th   | 077355123   |
|   300    | Dennis      | Robert     | Programmer  |  19000 | Suratthani | ha@sru.ac.th   | 077355443   |
```

ตัวอย่างที่ 22 การใช้คำคำสอบถาม เพื่อเรียกดูข้อมูลในตารางด้วย SELECT **แบบมีเงื่อนไข** เป็นช่วงของค่า

```
mysql>SELECT * FROM employees WHERE salary BETWEEN 15000 AND 20000;
```

ผลลัพธ์

```
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
| emp_id | firstname | lastname | position  | salary | address   | email       | phone     |
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
|    200 | Jaidee    | Ruktum   | Programmer|  17000 | Suratthani| ja@sru.ac.th|077355123  |
|    300 | Dennis    | Robert   | Programmer|  19000 | Suratthani| ha@sru.ac.th|077355443  |
|    400 | John      | Mackay   | Finance   |  18000 | Chumphon  | jo@sru.ac.th|077355889  |
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
3 rows in set (0.00 sec)
```

ตัวอย่างที่ 23 การใช้คำสอบถาม ด้วย LIKE เพื่อเรียกดูข้อมูลในตารางด้วย SELECT แบบกำหนดเงื่อนไข

```
mysql>SELECT * FROM employees WHERE lastname LIKE 'R%';
```

ผลลัพธ์

```
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
| emp_id | firstname | lastname | position  | salary | address   | email       | phone     |
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
|    200 | Jaidee    | Ruktum   | Programmer|  17000 | Suratthani| ja@sru.ac.th| 077355123 |
|    300 | Dennis    | Robert   | Programmer|  19000 | Suratthani| ha@sru.ac.th| 077355443 |
+--------+-----------+----------+-----------+--------+-----------+-------------+-----------+
2 rows in set (0.00 sec)
```

2.1) การใช้ GROUP BY เพื่อค้นหาข้อมูลแล้วแยกเป็นกลุ่มตามเงื่อนไขที่กำหนด สามารถใช้ร่วมกับฟังก์ชัน Aggregate เพื่อหาค่าทางสถิติของกลุ่มข้อมูล มีรายละเอียดดังนี้

ตารางที่ 10 แสดงฟังก์ชัน Aggregate สำหรับใช้ร่วมกับ GROUP BY

ฟังก์ชัน	คำอธิบาย
AVG (column)	หาค่าเฉลี่ยของขอบเขตข้อมูล
COUNT (item)	นับจำนวนรายการ
MIN (column) / MAX (column)	หาค่าต่ำสุด / ค่าสูงสุด ในขอบเขตข้อมูล
STD (column) / STDDEV (column)	หาค่า Standard Deviation (ส่วนเบี่ยงเบนมาตรฐาน) ในขอบเขตข้อมูล

ตารางที่ 10 (ต่อ)

ฟังก์ชัน	คำอธิบาย
SUM (column)	หาค่าผลรวมของขอบเขตข้อมูล

ตัวอย่างที่ 24 การใช้ GROUP BY และฟังก์ชัน COUNT ( )

```
mysql>SELECT address, COUNT (address) FROM employees GROUP BY address;
```

จากตัวอย่างที่ 13.24 เป็นการสอบถามข้อมูล โดยใช้ SELECT เพื่อแสดงขอบเขตข้อมูล "address" และใช้ฟังก์ชัน COUNT ( ) นับขอบเขตข้อมูล "address" จากนั้นใช้ GROUP BY เพื่อหาค่าของข้อมูลตามกลุ่มของขอบเขตข้อมูล "address"

ผลลัพธ์

```
-----------------------------------------------
| address        | COUNT(address)  |
-----------------------------------------------
| Chumphon   |            1          |
| Suratthani   |            3          |
-----------------------------------------------
2 rows in set (0.01 sec)
```

2.2) การใช้ ORDER BY ร่วมกับ SELECT เพื่อเลือกข้อมูลโดยการเรียงลำดับตามเงื่อนไขที่กำหนดจากน้อยไปมาก (ASC: Ascending) หรือมากไปน้อย (DESC: Descending) มีตัวอย่างดังนี้

ตัวอย่างที่ 25 การใช้ ORDER BY ร่วมกับ SELECT

```
mysql>SELECT * FROM employees ORDER BY salary DESC;
```

จากตัวอย่างเป็นการใช้ SELECT โดยเรียงลำดับค่าในขอบเขตข้อมูล "salary" จากมากไปน้อย (DESC)

ผลลัพธ์

```
-----------------------------------------------------------------------------------------------------------------
| emp_id   | firstname   | lastname   | position     | salary  | address     | email          |phone      |
-----------------------------------------------------------------------------------------------------------------
|   100     | Parinya     | Noidonprai | Manager     |  45000 | Suratthani  | pa@sru.ac.th  | 077355466 |
|   300     | Dennis      | Robert     | Programmer|  19000 | Suratthani  | ha@sru.ac.th  | 077355443 |
|   400     | John        | Mackay     | Finance     |  18000 | Chumphon | jo@sru.ac.th  | 077355889 |
|   200     | Jaidee      | Ruktum     | Programmer|  17000 | Suratthani  | ja@sru.ac.th  | 077355123 |
-----------------------------------------------------------------------------------------------------------------
```

2.3) การใช้ LIMIT ร่วมกับ SELECT ใช้เพื่อระบุแนวเริ่มต้นและจำนวนแถวที่จะแสดง โดยจะต้องส่งค่าพารามิเตอร์ 2 ค่า คือ ตำแหน่งของแถวเริ่มต้นและจำนวนแถว มีตัวอย่างดังนี้

ตัวอย่างที่ 26 การใช้ LIMIT ร่วมกับ SELECT

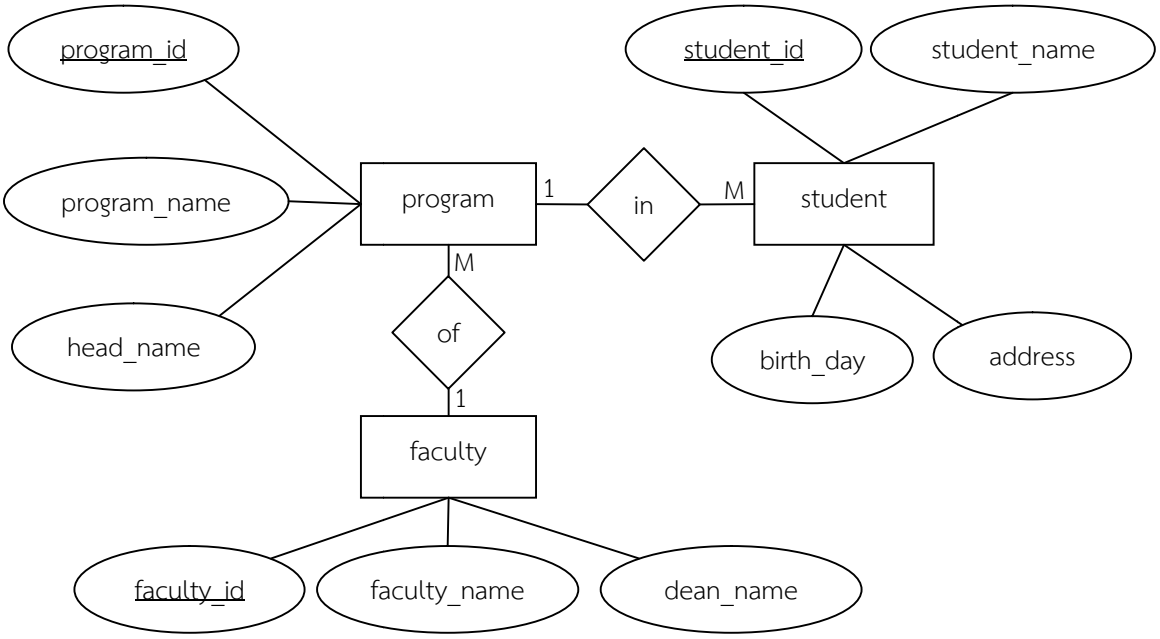
```
mysql>SELECT * FROM employees LIMIT 2, 2;
```

จากตัวอย่างเป็นการใช้ SELECT ร่วมกับ LIMIT โดยเริ่มต้นจากแถวที่ 3 (เริ่มนับแถวแรก คือ 0) แสดงทั้งหมด 2 แถว

ผลลัพธ์

```
-----------------------------------------------------------------------------------------------------------------
| emp_id   | firstname   | lastname   | position    | salary  | address     | email          |phone      |
-----------------------------------------------------------------------------------------------------------------
|   300    | Dennis      | Robert     | Programmer|  19000 | Suratthani  | ha@sru.ac.th  | 077355443 |
|   400    | John        | Mackay     | Finance     |  18000 | Chumphon | jo@sru.ac.th  | 077355889 |
-----------------------------------------------------------------------------------------------------------------
4 rows in set (0.00 sec)
```

3) การใช้คำสอบถามเพื่อเรียกดูข้อมูลที่มาจากตารางมากกว่า 1 ตาราง (ตารางที่มีความสัมพันธ์กัน โดยใช้ขอบเขตข้อมูลสำหรับการอ้างอิง) ด้วย SELECT แบบมีเงื่อนไข ดังนี้



ภาพที่ 3 แสดงความสัมพันธ์ของข้อมูล ในรูปแบบ ER-Diagram สำหรับเก็บข้อมูลนักศึกษา

จากภาพที่ 13.3 แสดงความสัมพันธ์ของข้อมูล ในรูปแบบ ER-Diagram สำหรับเก็บข้อมูลนักศึกษา หลังจากขั้นตอนนี้จะต้องถอดความสัมพันธ์เพื่อหาแอททริบิวต์ของความสัมพันธ์ (ไม่อธิบายส่วนของการถอดความสัมพันธ์) แล้วนำไปเขียนเป็นพจนานุกรมข้อมูล (Data Dictionary) เพื่อ

ความสะดวกสำหรับการพัฒนาฐานข้อมูล จากภาพที่ 13.3 สามารถ สร้างตารางความสัมพันธ์ได้ 3 ตาราง ประกอบด้วย 1) ตารางข้อมูลนักศึกษา (student) 2) ตารางข้อมูลสาขาวิชา (program) และ 3) ตารางคณะ (faculty) มีรายละเอียดของตารางความสัมพันธ์ดังนี้

ตารางที่ 11 ตัวอย่างตารางข้อมูลนักศึกษา

ขอบเขตข้อมูล	ชนิดชนิด	ว่างเปล่า (null)	ค่าปริยาย	คีย์	หมายเหตุ
student_id	VARCHAR(5)	Not	-	PK	รหัสนักศึกษา
student_name	VARCHAR(40)	Not	-	-	ชื่อ-สกุล
birth_day	DATE	-	-	-	วันเดือนปีเกิด
address	VARCHAR(40)	-	-	-	ที่อยู่
program_id	VARCHAR(3)	Not	-	FK	รหัสสาขาวิชา

ตารางที่ 12 ตัวอย่างตารางข้อมูลสาขาวิชา

ขอบเขตข้อมูล	ชนิด	ว่างเปล่า (null)	ค่าปริยาย	คีย์	หมายเหตุ
program_id	VARCHAR(3)	Not	-	PK	รหัสสาขาวิชา
program_name	VARCHAR(40)	Not	-	-	ชื่อสาขาวิชา
head_name	VARCHAR(40)	-	-	-	ชื่อหัวหน้าสาขาวิชา
faculty_id	VARCHAR(3)	Not	-	FK	รหัสคณะ

ตารางที่ 13 ตัวอย่างตารางข้อมูลคณะ

ขอบเขตข้อมูล	ชนิด	ว่างเปล่า (null)	ค่าปริยาย	คีย์	หมายเหตุ
faculty_id	VARCHAR(3)	Not	-	PK	รหัสคณะ
faculty_name	VARCHAR(40)	Not	-	-	ชื่อคณะ
dean_name	VARCHAR(40)	-	-	-	ชื่อคณบดี

ตารางที่ 14 แแสดงตัวอย่างข้อมูลในตาราง ข้อมูลนักศึกษา

student_id	student_name	birth_day	address	program_id
57001	Southern Thailand	1990/01/02	Suratthani	101
57002	Siriwatthan Jaidee	1990/05/18	Suratthani	101
57003	Rakthum Surat	1991/08/20	Chumphon	102
57004	Arun Gnamthong	1989/09/11	Suratthani	103
57005	Thongchai Swee	1990/08/12	Chumphon	104

ตารางที่ 15 แสดงตัวอย่างข้อมูลในตาราง ข้อมูลสาขาวิชา

program_id	program_name	head_name	faculty_id
101	Computer Science	Parinya Noidonprai	A01
102	Business	Aurai Somwang	A02
103	Biology	Siwat Hengchaiyo	A01
104	Human Develop	Siriwan Suwanmani	A03

ตารางที่ 16 แสดงตัวอย่างข้อมูลในตาราง ข้อมูลคณะ

faculty_id	faculty_name	dean_name
A01	Science and Techlogy	Somporn Thojai
A02	Management Science	Anirut Gnamsuwan
A03	Humanity Science	Sawai Suwanpradit

การใช้คำสอบถามเพื่อเรียกดูข้อมูลที่มาจากตารางมากกว่า 1 ตาราง (ตารางที่มีความสัมพันธ์กัน โดยใช้ขอบเขตข้อมูลสำหรับการอ้างอิง) ด้วย SELECT แบบมีเงื่อนไข อธิบายวิธีการดังนี้

กรณีศึกษาที่ 1 ต้องการดูข้อมูลนักศึกษาทั้งหมดในสาขาวิชา Computer Science สามารถใช้คำสอบถาม SELECT แบบมีเงื่อนไข ได้ดังนี้

ตัวอย่างที่ 27 การใช้ คำสอบถามเพื่อเรียกดูข้อมูลที่มาจากหลายตารางด้วย SELECT กรณีศึกษาที่ 1

```
mysql>  SELECT `student`.`student_id`,
        `student`.`student_name`,
        `student`.`program_id`,
        `program`.`program_id`,
        `program`.`program_name`
        FROM student, program
        WHERE `student`.`program_id`=`program`.`program_id` AND
        `program`.`program_id`='101'
        ORDER BY `student`.`program_id` ASC;
```

ผลลัพธ์

```
+------------ +------------------------ +------------ +------------ +------------------------ +
| student_id  | student_name            | program_id | program_id | program_name            |
+------------ +------------------------ +------------ +------------ +------------------------ +
| 57001       | Southern Thailand       | 101        | 101        | Computer Science        |
```

```
| 57002      | Siriwatthan Jaidee | 101          | 101          | Computer Science |
| 57005      | SThongchai Swee    | 101          | 101          | Computer Science |
```

กรณีศึกษาที่ 2 ต้องการดูข้อมูลนักศึกษาทั้งหมดในคณะ Science and Techlogy สามารถใช้คำสอบถาม SELECT แบบมีเงื่อนไข ได้ดังนี้

ตัวอย่างที่ 28 การใช้คำสอบถามเพื่อเรียกดูข้อมูลที่มาจากหลายตารางด้วย SELECT กรณีศึกษาที่ 2

```
mysql> SELECT `student`.`student_id`,
       `student`.`student_name`,
       `student`.`program_id`,
       `program`.`program_id`,
       `program`.`program_name`,
       `program`.`faculty_id`,
       `faculty`.`faculty_id`,
       `faculty`.`faculty_name`
       FROM student, program, faculty
       WHERE `student`.`program_id`=`program`.`program_id`
       AND `program`.`faculty_id`=`faculty`.`faculty_id`
       AND `faculty`.`faculty_id`='A01'
       ORDER BY `student`.`program_id` ASC;
```

ผลลัพธ์

```
+------------+-----------------+------------+------------+-----------------+------------+------------+-----------------+
| student_id | student_name    | program_id | program_id | program_name    | faculty_id | faculty_id | faculty_name    |
+------------+-----------------+------------+------------+-----------------+------------+------------+-----------------+
| 57001      | Southern Thailand | 101      | 101        | Computer Science | A01        | A01        | Science and Techlogy |
| 57002      | Siriwatthan Jaidee | 101     | 101        | Computer Science | A01        | A01        | Science and Techlogy |
| 57005      | SThongchai Swee | 101        | 101        | Computer Science | A01        | A01        | Science and Techlogy |
| 57004      | Arun Gnamthong  | 103        | 103        | Biology         | A01        | A01        | Science and Techlogy |
+------------+-----------------+------------+------------+-----------------+------------+------------+-----------------+
4 rows in set (0.00 sec)
```