



# ระบบจัดการฐานข้อมูลทางธุรกิจ

## Chapter 6 Structure Query Language



Aj. Pakawat T.Siriwattana

# ภาษา SQL

2

- ภาษา SQL สามารถอ่านได้สองแบบคือ เอสคิวเอล หรือ ซีเคอล (SEQUEL : Structure English QUERy Language, 1974 IBM)
- เป็นภาษามาตรฐานที่ใช้ในจัดการฐานข้อมูลเชิงสัมพันธ์
- มีรูปแบบโครงสร้างภาษาที่ง่ายต่อการเรียนรู้และการใช้งาน
- รูปแบบภาษาเป็นภาษาอังกฤษ



# Data Type

3

ชนิดข้อมูล	คำอธิบาย
CHAR(M)	สำหรับเก็บข้อมูลประเภทตัวอักษร ขนาดตามค่า M แต่ไม่เกิน 255 ตัวอักษร
VARCHAR(M)	สำหรับเก็บข้อมูลประเภทตัวอักษร ขนาดตามค่า M แต่ไม่เกิน 255 ตัวอักษร หมายเหตุ การเก็บข้อมูลสั้นๆ เช่น ชื่อ นามสกุล ฯลฯ
TINYTEXT	เป็น case-insensitive สามารถเก็บข้อมูล ได้ 255 ตัวอักษร
TINYBLOB	เป็น case-sensitive สำหรับการเรียงและเปรียบเทียบ สามารถเก็บข้อมูล ได้ 255 ตัวอักษร
TEXT	เป็น case-insensitive เก็บข้อมูลประเภทตัวอักษร เช่นเดียวกับ TINYTEXT แต่สามารถเก็บได้สูงสุดคือ 65,535 ตัวอักษร หมายเหตุ ข้อมูลจำนวนมาก ๆ
BLOB	เป็น case-sensitive สำหรับการเรียงและเปรียบเทียบ สามารถเก็บข้อมูล ได้สูงสุดคือ 65,535 ตัวอักษร
SET('value1','value2',...)	จะใช้เลขฐานสองเก็บค่าตัวเลือก โดยรายการแรกมีค่า 1, 2, 4, 8, 16, 32,... ไปเรื่อยๆ ถ้าเลือกรายการไหนก็เอามาบวกกันจะได้ค่าที่จะเก็บบันทึก หมายเหตุ การเลือกแบบเช็คบ็อกซ์ เพื่อให้ผู้ตอบเลือกได้มากกว่า 1 ตัวเลือก SET เก็บค่ารายการได้ 64 ตัวเลือก



# Data Type

4

MySQL Data Types		
Type	Size	Description
INT[Length]	4 bytes	สำหรับเก็บข้อมูลชนิดตัวเลข จำนวนเต็มไม่มีทศนิยม signed คือ -2147483648 ถึง 2147483647 unsigned คือ 0 ถึง 4294967295
FLOAT	4 bytes	A small number with a floating decimal point.
DECIMAL[Length, Decimals]	Length + 1 or Length + 2 byte	A DOUBLE stored as a string, allowing for a fixed decimal point
BIGINT[Length]	8 bytes	Range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 unsigned.



# Data Type

5

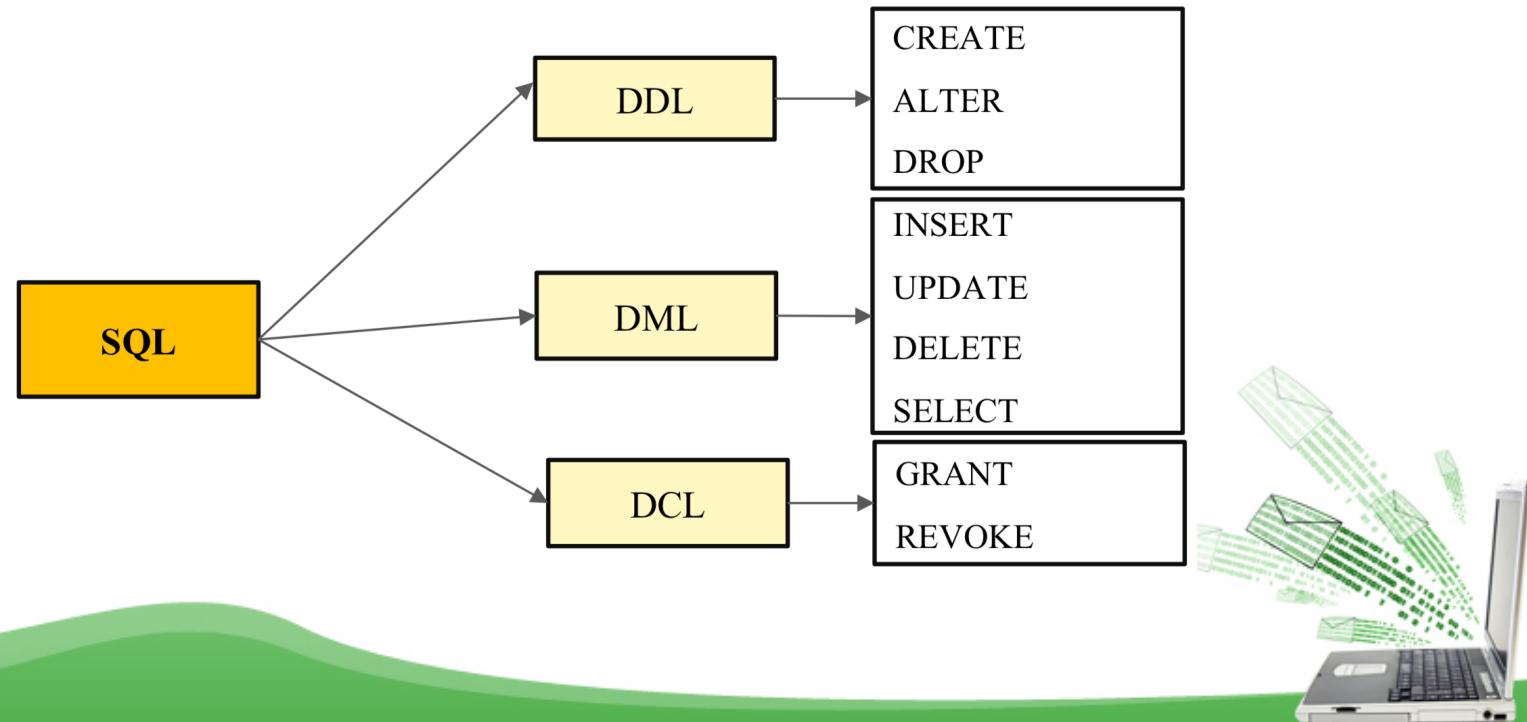
ชนิดข้อมูล	คำอธิบาย
DATE	สำหรับเก็บข้อมูลประเภทวันที่ โดยเก็บได้จาก 1 มกราคม ค.ศ. 1000 ถึง 31 ธันวาคม ค.ศ. 9999 โดยจะแสดงผลในรูปแบบ YYYY-MM-DD
TIME	สำหรับเก็บข้อมูลประเภทเวลา โดยจะแสดงผล ออกมานั่นรูปแบบ HH:MM:SS
DATETIME	สำหรับเก็บข้อมูลประเภทวันที่ และเวลา โดยรูปแบบการแสดงผล จะเป็น YYYY-MM-DD HH:MM:SS
TIMESTAMP[(M)]	สำหรับเก็บข้อมูลประเภทวันที่ และเวลาเช่นกัน แต่จะเก็บในรูปแบบของ YYYYMMDDHHMMSS หรือ YMMDDHHMMSS หรือ YYYYMMDD หรือ YYMMDD และแต่ ว่าจะระบุค่า M เป็น 14, 12, 8 หรือ 6 ตามลำดับ
YEAR[(2/4)]	สำหรับเก็บข้อมูลประเภทปี ในรูปแบบ YYYY หรือ YY และแต่จะเลือกโดย หากเลือกเป็น 4 หลัก จะเก็บค่าได้ตั้งแต่ ค.ศ. 1901 ถึง 2155 แต่ หากเป็น 2 หลัก จะเก็บตั้งแต่ ค.ศ. 1970 ถึง 2069



# Types of SQL language commands

6

- ภาษาสำหรับการนิยามข้อมูล (Data Definition Language : DDL)
- ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language : DML)
- ภาษาควบคุม (Data Control Language : DCL)



# ແນະນຳ DBMS - MYSQL

7

The screenshot shows the phpMyAdmin interface running on a local server at 127.0.0.1. The left sidebar lists databases: New, ac\_24b, icecream, information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main content area has three panels: 'General settings' (Server connection collation: utf8mb4\_unicode\_ci), 'Appearance settings' (Language: English, Theme: pmahomme, Font size: 82%), and 'Database server' (Server: 127.0.0.1 via TCP/IP, Server type: MariaDB, Server version: 10.1.25-MariaDB - mariadb.org binary distribution, Protocol version: 10, User: root@localhost, Server charset: UTF-8 Unicode (utf8)). Below these is a 'Web server' panel listing Apache/2.4.26 (Win32) OpenSSL/1.0.2I PHP/5.6.31, Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - \$Id: 76b08b24596e12d4553bd41fc93cc\$, PHP extension: mysqli curl mbstring, and PHP version: 5.6.31.

localhost / 127.0.0.1 | ph... X

localhost/phpmyadmin/

## phpMyAdmin

Recent Favorites

- New
- ac\_24b
- icecream
- information\_schema
- mysql
- performance\_schema
- phpmyadmin
- test

### General settings

Server connection collation:

utf8mb4\_unicode\_ci

### Appearance settings

Language: English

Theme: pmahomme

Font size: 82%

More settings

### Database server

- Server: 127.0.0.1 via TCP/IP
- Server type: MariaDB
- Server version: 10.1.25-MariaDB - mariadb.org binary distribution
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

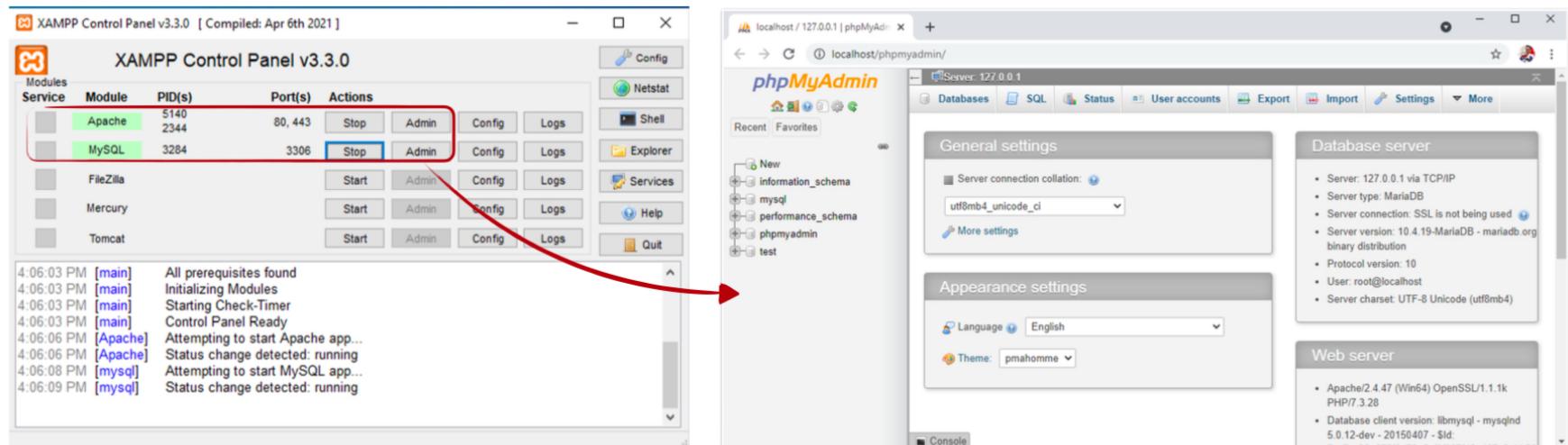
### Web server

- Apache/2.4.26 (Win32)  
OpenSSL/1.0.2I PHP/5.6.31
- Database client version: libmysql - mysqlnd 5.0.11-dev - 20120503 - \$Id: 76b08b24596e12d4553bd41fc93cc\$
- PHP extension: mysqli curl mbstring
- PHP version: 5.6.31

# ขั้นตอนการเข้าใช้งาน

8

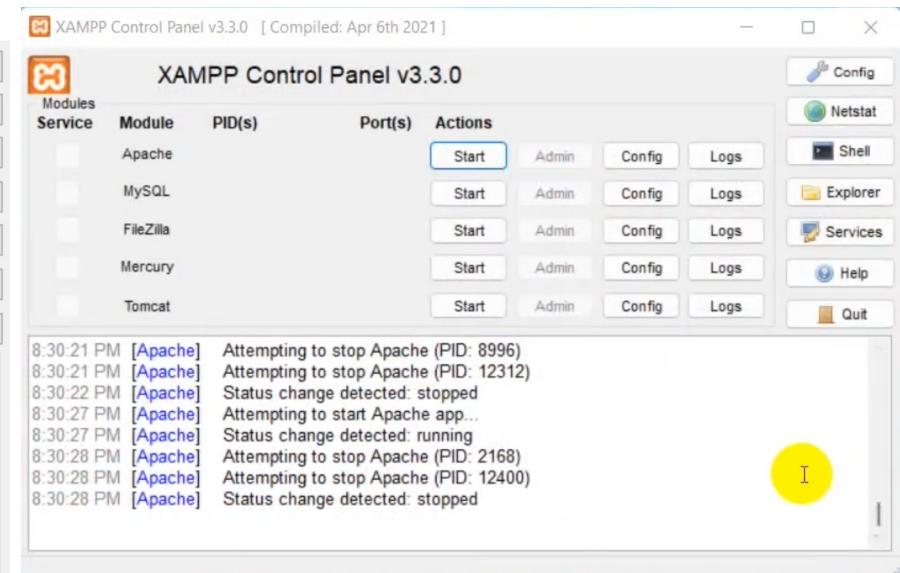
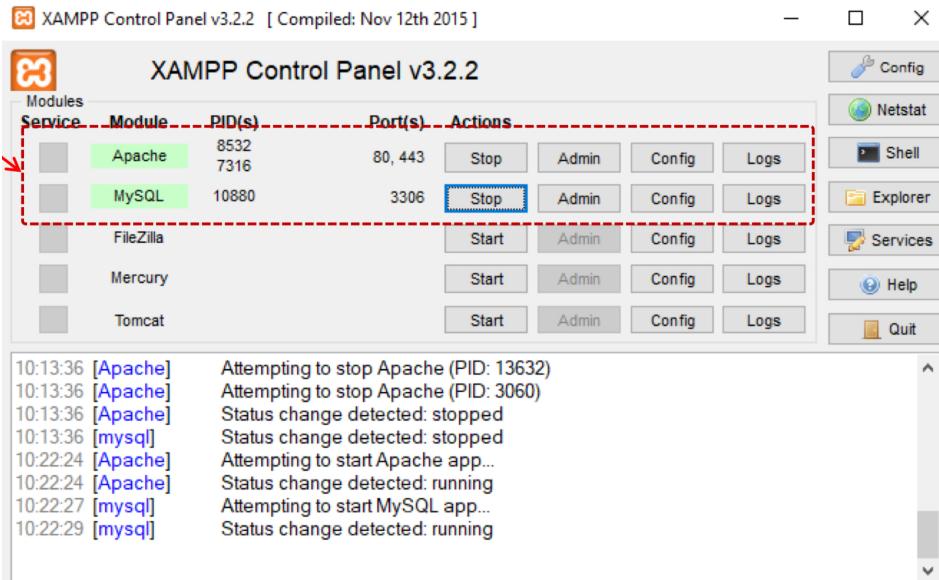
- เปิดโปรแกรม XAMPP จากนั้นคลิก Start Appche และ MySQL
- คลิกที่ปุ่ม Admin ที่แถบของ MySQL เพื่อเปิดโปรแกรมจัดการฐานข้อมูล



# ขั้นตอนการเข้าใช้งาน

9

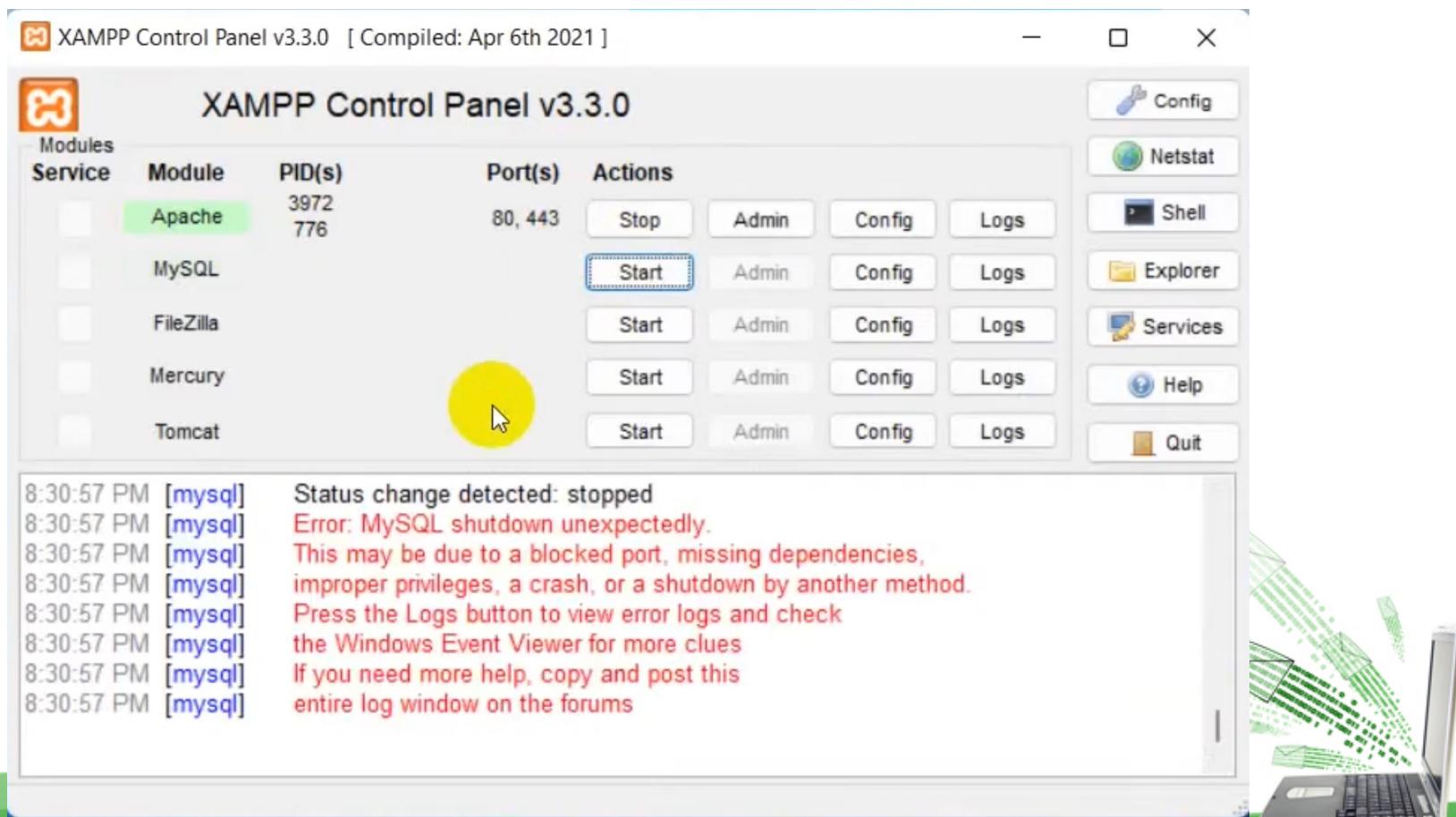
- เมื่อหยุดการใช้งานทุกครั้งให้กด STOP หากลืมกด STOP จะทำให้ MySQL ERROR



# ขั้นตอนการเข้าใช้งาน

10

- เมื่อหยุดการใช้งานทุกครั้งให้กด STOP หากลืมกด STOP จะทำให้ MySQL ERROR



# Data Definition Language : DDL

11

คำสั่ง		รายละเอียด
CREATE	DATABASE	สร้างฐานข้อมูล
	TABLE	สร้างโครงสร้างตารางข้อมูล
	INDEX	สร้างดัชนีของตารางข้อมูล
	VIEW	สร้างวิวข้อมูลหรือตารางข้อมูลเสมือน (Virtual Table) ของผู้ใช้
DROP	DATABASE	ยกเลิกฐานข้อมูลออกจากระบบ
	TABLE	ยกเลิกโครงสร้างตารางข้อมูล
	INDEX	ยกเลิกดัชนีของตารางข้อมูล
	VIEW	ยกเลิกโครงสร้างวิวข้อมูลหรือตารางข้อมูลเสมือนของผู้ใช้
ALTER	TABLE	เปลี่ยนแปลงโครงสร้างตารางข้อมูล



# ชื่อฐานข้อมูลคือ db\_icecream

12

Table: MenuItems

Field Name	Field Type	Size	PK/FK	Description
ItemID	Int		PK	รหัสเมนู
ItemType	varchar	50		ประเภทเมนู
ItemName	Varchar	200		ชื่อเมนู
Toppings	Varchar	300		เมนูที่อปปิงเพิมเติม
Description	Varchar	500		หมายเหตุ

Table: ItemPrice

Field Name	Field Type	Size	PK/FK	Description
ItemID	Int		PK, FK	รหัสเมนู
Size	Varchar	15		ชื่อขนาดของเมนู
Price	Int			ราคา



# ชื่อฐานข้อมูลคือ db\_icecream

13

Table: Orders

Field Name	Field Type	Size	PK/FK	Description
OrderId	Int		PK	รหัสการสั่งซื้อ
OrderDate	Date			วันที่สั่งซื้อ
CustomerName	Varchar	100		ชื่อลูกค้า
CustomerAddress	Varchar	300		ที่อยู่ลูกค้า
DeliveryCharge	Int			ค่าจัดส่ง
TotalValue	Int			รวมค่าใช้จ่าย

Table: OrderItems

Field Name	Field Type	Size	PK/FK	Description
OrderId	Int		PK,FK	รหัสการสั่งซื้อ
ItemId	Int		PK,FK	รหัสเมนู
Size	Varchar	15		ชื่อขนาดของเมนู
Quantity	Int			จำนวนสั่งซื้อ



# CREATE Commands

14

## ❖ CREATE Database : สร้างฐานข้อมูล

### ■ Syntax

```
CREATE Database Database_name [CHARACTER SET  
    charset_name COLLATE collation name];
```

โดยที่

***charset\_name*** คือ ชุดรูปแบบตัวอักษรที่ใช้ในระบบฐานข้อมูล เช่น utf-8 คือ รองรับชุดอักขระเดิมหลายภาษารวมถึงภาษาไทย ส่วน tis-620 รองรับชุดอักขระภาษาไทยและภาษาอังกฤษเป็นหลัก

***Collation\_name*** คือ การจัดเรียงอักขระที่ใช้ในระบบฐานข้อมูลเพื่อให้สอดคล้องกับข้อมูลหรือภาษาที่จัดเก็บไว้ในระบบฐานข้อมูล เช่น utf8\_Unicode\_ci จะเป็นการจัดเรียงอักขระตามภาษาท้องถิ่น



# CREATE Commands

15

```
CREATE DATABASE de_icecream  
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
1 CREATE DATABASE db_icecream  
2 CHARACTER SET utf8 COLLATE utf8_general_ci;
```

ผลลัพธ์จากชุดคำสั่งจะได้ระบบฐานข้อมูลชื่อ db\_icecream รองรับภาษาไทยกับภาษาอังกฤษ

The screenshot shows the phpMyAdmin interface. On the left, the database tree lists several databases, with 'db\_icecream' highlighted by a red box. In the main area, a SQL query has been entered:

```
CREATE DATABASE db_icecream CHARACTER SET utf8 COLLATE utf8_general_ci;
```

The output of the query is displayed in a green box with a checkmark icon:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0143 seconds.)

Below the query, there are three buttons: [Edit inline], [Edit], and [Create PHP code]. A yellow warning box at the bottom states:

Error: #1046 No database selected

# CREATE Commands

16

## ❖ CREATE TABLE : สร้างตาราง

### ■ Syntax

CREATE TABLE *table\_name*

(

*column1 Datatype( ) Constraint,*

*column2 Datatype( ) Constraint,*

.....

**PRIMARY KEY (*Field\_name*) ,**

**FOREIGN KEY (*Field\_name*) REFERENCES *table\_name*(*Field\_name*)**

);



# CREATE Commands

17

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A red box highlights the following SQL code:

```
1 CREATE TABLE MenuItems
2     (ItemID      integer      NOT NULL,
3      ItemType    varchar(10)   NOT NULL,
4      ItemName    varchar(50)   NOT NULL,
5      Toppings    varchar(150)  ,
6      Description varchar(300)  ,
7      PRIMARY KEY (ItemID));
8
```

Create table MenuItems  
(ItemID integer Not null,  
ItemType varchar(10) Not null,  
ItemName varchar(50) Not Null,  
Toppings Varchar(150),  
Description varchar(300),  
Primary Key(ItemID)  
);

ผลลัพธ์ใช้คำสั่ง DESCRIBE TABLENAME;

The screenshot shows the MySQL Workbench interface after executing a DESCRIBE query. A red box highlights the query:

```
1 DESCRIBE MenuItems;
```

The results show the table structure:

Field	Type	Null	Key	Default	Extra
ItemID	int(11)	NO	PRI	NULL	
ItemType	varchar(50)	NO		NULL	
ItemName	varchar(200)	NO		NULL	
Toppings	varchar(300)	NO		NULL	
Description	varchar(500)	NO		NULL	

The screenshot shows the MySQL Workbench interface with the Structure tab selected. A red box highlights the table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	temID	int(11)	utf8_unicode_ci		No	None			Change  Drop  More
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			Change  Drop  More
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			Change  Drop  More
4	Toppings	varchar(300)	utf8_unicode_ci		Yes	NULL			Change  Drop  More
5	Description	varchar(500)	utf8_unicode_ci		Yes	NULL			Change  Drop  More

# CREATE Commands

18

ตาราง ItemPrice

Field Name	Field Type	Size	PK/FK	Description
ItemID	Int		PK, FK	รหัสเมนู
Size	Varchar	15		ชื่อขนาดของเมนู
Price	Int			ราคา

```
Create Table ItemPrice(  
    ItemID integer,  
    Price integer,  
    Size Varchar(15),  
    PRIMARY KEY(ItemID),  
    FOREIGN KEY(ItemID) REFERENCES MenuItems(ItemID)  
)
```



# CREATE Commands

19

ตาราง Orders

Field Name	Field Type	Size	PK/FK	Description
OrderId	Int		PK	รหัสการสั่งซื้อ
OrderDate	Date			วันที่สั่งซื้อ
CustomerName	Varchar	100		ชื่อลูกค้า
CustomerAddress	Varchar	300		ที่อยู่ลูกค้า
DeliveryCharge	Int			ค่าจัดส่ง
TotalValue	Int			รวมค่าใช้จ่าย

ตาราง OrderItems

Field Name	Field Type	Size	PK/FK	Description
OrderId	Int		PK,FK	รหัสการสั่งซื้อ
ItemId	Int		PK,FK	รหัสเมนู
Size	Varchar	15		ชื่อขนาดของเมนู
Quantity	Int			จำนวนสั่งซื้อ



# Example: Create Table

20

Table Name      Data type      Constraint  
CREATE TABLE EMPLOYEE (  
    ID            CHAR(5)      NOT NULL ,  
    NAME          VARCHAR(35)   NOT NULL ,  
    ADDRESS       VARCHAR(15)   NOT NULL ,  
    PHONE         CHAR(8)      NOT NULL ,  
    E\_MAIL        CHAR(1)      NOT NULL ,  
    PRIMARY KEY(ID)  
);

Set ID to be PK

คำสั่ง SQL นี้จะกำหนดให้คอลัมน์ 'ID' เป็นคีย์หลัก (Primary Key) ในขณะที่กำลังสร้างตารางพนักงาน (Employee)



# Example: Create Table

21

Table: PRODUCT

No	Field Name	Type	Size	PK	Description
1	P_CODE	Varchar	10	PK	รหัสสินค้า
2	P_NAME	Varchar	35		ชื่อสินค้า
3	P_INDATE	Date	4		วันที่ผลิต
4	P_ONHAND	Smallint	8		สินค้าคงเหลือ
5	P_MIN	Smallint	1		จำนวนต่ำสุด
6	P_PRICE	Number			ราคา
7	P_DISCOUNT	Number			ส่วนลด
8	ID	Char	5	FK	รหัสพนักงาน



# SQL FOREIGN KEY Constraint

22

- Foreign Key (คีย์นอก) คือตัวเข้ามที่ใช้สำหรับจับคู่หรือเข้ามโยงข้อมูลระหว่าง 2 ตารางเข้าด้วยกัน

```
CREATE TABLE PRODUCT (
    P_CODE          VARCHAR(10) ,
    P_NAME          VARCHAR(35)      NOT NULL ,
    P_INDATE        DATE           NOT NULL ,
    P_ONHAND        SMALLINT       NOT NULL ,
    P_MIN           SMALLINT       NOT NULL ,
    P_PRICE          INT            NOT NULL ,
    P_DISCOUNT      INT            NOT NULL ,
    ID              CHAR(5) ,
    PRIMARY KEY (P_CODE) ,
    FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID)
);
```



# SQL FOREIGN KEY Constraint

23

- Foreign Key (คีย์นอก) คือตัวเข้ามที่ใช้สำหรับจับคู่หรือเข้ามโยงข้อมูลระหว่าง 2 ตารางเข้าด้วยกัน

```
CREATE TABLE PRODUCT (
    P_CODE          VARCHAR(10) ,
    P_NAME          VARCHAR(35)      NOT NULL ,
    P_INDATE        DATE           NOT NULL ,
    P_ONHAND        SMALLINT       NOT NULL ,
    P_MIN           SMALLINT       NOT NULL ,
    P_PRICE          INT            NOT NULL ,
    P_DISCOUNT      INT            NOT NULL ,
    ID              CHAR(5) ,
    PRIMARY KEY (P_CODE) ,
    FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID)
);
```

Set ID to be FK



# SQL CREATE INDEX Statement

24

การสร้างดัชนีของตาราง

- เป็นการสร้างดัชนีให้กับตาราง
- การสร้างดัชนีจะช่วยเพิ่มให้เพิ่มประสิทธิภาพในการคิรีข้อมูล



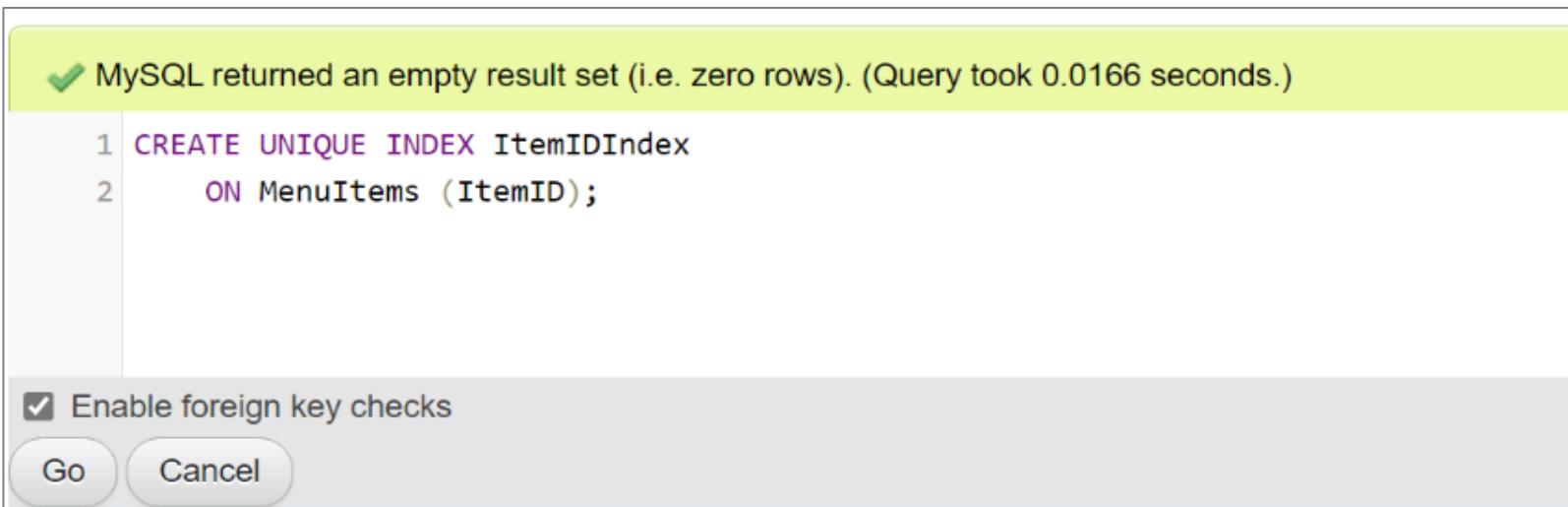
# SQL CREATE INDEX Statement

25

## ❖ CREATE INDEX Syntax

```
CREATE UNIQUE INDEX index_name
    ON table_name (column1, column2, ...);
```

## ❖ CREATE INDEX Example



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0166 seconds.)

```
1 CREATE UNIQUE INDEX ItemIDIndex
2     ON MenuItems (ItemID);
```

Enable foreign key checks

Go Cancel



# SQL CREATE INDEX Statement

26

ผลลัพธ์ที่ได้

The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected. In the 'Table structure' view, the 'ItemID' column is highlighted with a red box. Below the table, the 'Indexes' section is expanded, showing two entries. The first entry is for the primary key 'PRIMARY'. The second entry, which is the newly created index, is highlighted with a red box and has the name 'ItemIDIndex'.

Table structure:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)	utf8_unicode_ci		No	None			Change  Drop  More
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			Change  Drop  More
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			Change  Drop  More
4	Toppings	varchar(300)	utf8_unicode_ci		Yes	NULL			Change  Drop  More
5	Description	varchar(500)	utf8_unicode_ci		Yes	NULL			Change  Drop  More

Indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit  Rename  Drop	PRIMARY	BTREE	Yes	No	ItemID	0	A	No	
Edit  Rename  Drop	ItemIDIndex	BTREE	No	No	ItemID	0	A	No	

# DROP INDEX Statement

27

- ❖ The DROP INDEX statement is used to delete an index in a table
- ❖ SQL DROP VIEW Syntax

`DROP INDEX index_name ON table_name;`

Example

`DROP INDEX ItemIDIndex ON MenuItems;`



# SQL CREATE VIEW Statement

28

In SQL, a view is a virtual table based on the result-set of an SQL statement.

## CREATE VIEW Syntax

```
CREATE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0080 seconds.)

```
1 CREATE VIEW ViewMenuItems AS  
2     (SELECT ItemID, ItemType, ItemName FROM MenuItems);
```

Enable foreign key checks

Go

Cancel

1

2

3

4

5

phpMyAdmin

Server: 127.0.0.1 > Database: db\_icecream > Table: menuitems

Browse Structure SQL Search Insert Export Import Privileges Operations More

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0113 seconds.)

```
CREATE VIEW ViewMenuItems AS (SELECT ItemID, ItemType, ItemName FROM MenuItems);
```

Enable foreign key checks

Go Cancel

[Edit inline] [Edit] [Create PHP code]

1

2

3

4

5

Browse Structure SQL Search Insert Export

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0021 seconds.)

```
SELECT * FROM `viewmenuitems`
```

Profiling

ItemID	ItemType	ItemName
--------	----------	----------

# SQL CREATE VIEW Statement

30

SQL CREATE VIEW Examples

```
CREATE VIEW ViewEmp;  
As (select ID, NAME, PHONE  
      from EMPLOYEE);
```

```
CREATE TABLE EMPLOYEE (  
    ID          CHAR(5)      NOT NULL ,  
    NAME        VARCHAR(35)   NOT NULL ,  
    ADDRESS     VARCHAR(15)   NOT NULL ,  
    PHONE       CHAR(8)      NOT NULL ,  
    E-MAIL      CHAR(1)      NOT NULL ,  
    PRIMARY KEY(ID)  
);
```



# SQL Dropping a View

31

- ❖ A view is deleted with the DROP VIEW command.
- ❖ SQL DROP VIEW Syntax

`DROP VIEW view_name;`

- ❖ Example

`DROP VIEW ViewEmp;`



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0036 seconds.)

1 `DROP VIEW viewmenuitems;`

32

Enable foreign key checks

Go Cancel

The screenshot shows the phpMyAdmin interface for the database 'db\_icecream'. A red box labeled '1' highlights the 'Views' node under the 'Tables' section in the left sidebar. A red box labeled '2' highlights the 'Structure' tab in the top navigation bar. A red box labeled '3' highlights the 'Drop' button for the 'viewmenuitems' view in the 'Action' column of the main table. A red box labeled '4' highlights the 'OK' button in a confirmation dialog box.

2

1

3

4

Structure

SQL Search Query Export Import Operations Privileges Routines More

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
viewmenuitems	Browse Structure Search Insert Edit Drop	~0	View	InnoDB	utf8_unicode_ci	0 B

Confirm

Do you really want to execute "DROP VIEW `viewmenuitems`"?

Enable foreign key checks

OK Cancel

# SQL ALTER TABLE Statement

33

- ALTER TABLE : เปลี่ยนแปลงตาราง
- Changes the Structure of a table
  - Syntax

ALTER TABLE *table\_name*

([command to Change] [column\_name] [datatype]);

- Command to Change

- Add คือ การเพิ่มคอลัมน์
- Modify คือ เปลี่ยนแปลงขนาดของคอลัมน์
- Drop คือ ลบคอลัมน์
- Change คือ เปลี่ยนชื่อคอลัมน์



# ตัวอย่างการใช้คำสั่ง

34

การแก้ไขโครงสร้างตารางด้วยการเพิ่ม colum

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0036 seconds.)

```
1 ALTER TABLE MenuItems
2   ADD Picture varchar(150);
```

Enable foreign key checks

Go Cancel

Browse Structure SQL Search Insert Export Import Privileges Open

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)	utf8_unicode_ci		No	None			Change  Drop  More
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			Change  Drop  More
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			Change  Drop  More
4	Toppings	varchar(300)	utf8_unicode_ci		No	None			Change  Drop  More
5	Description	varchar(500)	utf8_unicode_ci		No	None			Change  Drop  More
6	Picture	varchar(150)	utf8_unicode_ci		Yes	NULL			Change  Drop  More

ItemID	ItemType	ItemName	Toppings	Description	Picture

คำสั่ง ALTER TABLE !!แบบ ADD



# ตัวอย่างการใช้คำสั่ง

35

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0125 seconds.)

```
1 ALTER TABLE MenuItems  
2 MODIFY COLUMN Picture varchar(200);
```

Enable foreign key checks

Go Cancel

Browse Structure SQL Search Insert Export Import Privileges Open

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action	
1	ItemID	int(11)			No	None			Change	Drop
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			Change	Drop
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			Change	Drop
4	Toppings	varchar(300)	utf8_unicode_ci		No	None			Change	Drop
5	Description	varchar(500)	utf8_unicode_ci		No	None			Change	Drop
6	Picture	varchar(200)	utf8_unicode_ci		Yes	NULL			Change	Drop

การแก้ไขขนาดข้อมูลของคอลัมน์



# ตัวอย่างการใช้คำสั่ง

36

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0196 seconds.)

```
1 ALTER TABLE MenuItems
2 CHANGE Picture PictrueFilePath varchar(200);
```

Enable foreign key checks

Go Cancel

การแก้ไขชื่อคอลัมน์ของตาราง

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)			No	None			Change  Drop  More
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			Change  Drop  More
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			Change  Drop  More
4	Toppings	varchar(300)	utf8_unicode_ci		No	None			Change  Drop  More
5	Description	varchar(500)	utf8_unicode_ci		No	None			Change  Drop  More
6	PictrueFilePath	varchar(200)	utf8_unicode_ci		Yes	NULL			Change  Drop  More

## การลบคอลัมน์ของตาราง

37

ItemID	ItemType	ItemName	Toppings	Description	PictureFilePath

คำสั่ง ALTER TABLE แบบ DROP 

ItemID	ItemType	ItemName	Toppings	Description	

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0185 seconds.)

```
1 ALTER TABLE MenuItems  
2   DROP PictureFilePath;
```

Enable foreign key checks

Go Cancel

Browse Structure SQL Search Insert Export Import Privileges Open

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)			No	None			 Change  Drop 
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			 Change  Drop 
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			 Change  Drop 
4	Toppings	varchar(300)	utf8_unicode_ci		No	None			 Change  Drop 
5	Description	varchar(500)	utf8_unicode_ci		No	None			 Change  Drop 



# การเพิ่มคอลัมน์ตารางด้วยตัวช่วยเหลือในโปรแกรม

38

The screenshot shows the phpMyAdmin interface for the db\_icecream database. The menuitems table is selected. A red box highlights the 'Structure' tab (labeled 2). In the bottom search bar, 'Add 1 column(s) after Description' is entered (labeled 3). The 'Picture' column is being defined with a VARCHAR type and length of 150 (labeled 4). Finally, the 'Save' button is highlighted (labeled 5).

Server: 127.0.0.1 Database: db\_icecream » Table: menuitems

Browse Structure SQL Search Insert Export Import Privileges

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)			No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
4	Toppings	varchar(300)	utf8_unicode_ci		Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/>
5	Description	varchar(500)	utf8_unicode_ci		Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/>

Add 1 column(s) after Description Go

Name Type Length/Values Default Collation Attributes Null Index

Picture VARCHAR 150 None

Structure

Online transaction Preview SQL Save

# การแก้ไขโครงสร้างโครงสร้างตารางข้อมูล

39

The screenshot shows the phpMyAdmin interface for managing a MySQL database named 'db\_icecream'. The 'menuitems' table is selected. The 'Structure' tab is active, indicated by a red circle labeled '2'. A red circle labeled '1' highlights the 'menuitems' table in the left sidebar. A red circle labeled '3' highlights the 'Change' button for the 'Description' column. A red circle labeled '4' highlights the 'PictureFilePath' column in the detailed view below. A red circle labeled '5' highlights the 'Save' button at the bottom of the edit form.

Server: 127.0.0.1 Database: db\_icecream Table: menuitems

Browse Structure SQL Search Insert Export Import Privileges

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ItemID	int(11)			No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
2	ItemType	varchar(50)	utf8_unicode_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
3	ItemName	varchar(200)	utf8_unicode_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/>
4	Toppings	varchar(300)	utf8_unicode_ci		Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/>
5	Description	varchar(500)	utf8_unicode_ci		Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/>

Check all With selected: Browse Change Drop Primary Unique

Index Spatial Fulltext Add to central columns Remove from central columns

Print Propose table structure Track table Move columns Normalize

Add 1 column(s) after Description Go

Browse Structure SQL Search Insert Export Import Privileges Operations More

Name	Type	Length/Values	Default	Collation	Attributes	Null
PictureFilePath	VARCHAR	200	None	utf8_unicode_ci		

Pick from Central Columns

Structure

Online transaction Preview SQL Save

# สรุปการใช้ AFTER TABLE

40

## 1. การเพิ่มคอลัมน์ใหม่ (ADD)

- ALTER TABLE EMPLOYEE  
ADD POSITION VARCHAR (30)

## 2. การแก้ไขรายละเอียดคอลัมน์เดิม (MODIFY)

- ALTER TABLE EMPLOYEE  
Modify Column PHONE VARCHAR (10) ;
- Alter table EMPLOYEE  
Modify Column E\_MAIL Char (30) ;

## 3. การลบคอลัมน์ (DROP)

- ALTER TABLE EMPLOYEE  
DROP POSITION;



# คำสั่ง SQL : DDL

41

- **DROP TABLE** : ลบตาราง

- Syntax

- DROP TABLE *Table\_name*;

- Example

- DROP TABLE PRODUCT;

- DROP TABLE EMPLOYEE;

- DROP TABLE PROJECT;



# ภาษาจัดการข้อมูล (Data Manipulation Language: DML)

42

คำสั่ง	ความหมาย
INSERT	เพิ่มແລວข้อมูลใหม่เข้าไปในตาราง
UPDATE	ปรับปรุงແລວข้อมูลในตาราง
DELETE	ลบແລວข้อมูลในตาราง
SELECT	สอบถามข้อมูลจากตารางข้อมูล



# Data Manipulation Language: DML

43

- **INSERT :** เพิ่มแถว (Row) ลงใน Table มี 2 รูปแบบ

## 1. INSERT INTO Syntax

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

The screenshot shows a MySQL Workbench interface. At the top, a green bar displays the message "1 row inserted. (Query took 0.0078 seconds.)". Below this, the SQL query is displayed in a code editor:

```
1 | INSERT INTO MenuItems
2 |     VALUES(1,
3 |             'ไอศกรีมเด็ก',
4 |             'สตรอเบอร์รี่พาราไดซ์',
5 |             'สตรอเบอร์รี่และแผ่นช็อกโกแล็ต',
6 |             'ไอศกรีมเด็ก 4 รสชาติ ประกอบด้วย วนิลลา สตรอเบอร์รี่ชีสเค้ก คุกกี้แอนด์ครีมและวาร์ฟ');
```

At the bottom of the interface, there is a checkbox labeled "Enable foreign key checks" which is checked. Below the checkbox are two buttons: "Go" and "Cancel".



# Data Manipulation Language: DML

44

## 2. INSERT INTO Syntax

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

1 row inserted. (Query took 0.0094 seconds.)

```
1 INSERT INTO MenuItems (ItemID,
2                         ItemType,
3                         ItemName,
4                         Toppings,
5                         Description)
6 VALUES(1,
7        'ไอศกรีมเค้ก',
8        'สตรอเบอร์รี่พาราไดซ์',
9        'สตรอเบอร์รี่และแพนเค้กโกโก้แล็ต',
10       'ไอศกรีมเค้ก 4 รสชาติ  pragobudai  วนิลลา สตรอเบอร์รี่ชีสเค้ก  คุกเก้แอนด์ครีมแลชาร์');
```

Enable foreign key checks

Go  Cancel



ตัวอย่างการใช้คำสั่ง เมื่อต้องการเพิ่มข้อมูลหนึ่งແລກเข้าไปในตาราง members  
สามารถทำได้ด้วยคำสั่งดังนี้

45

**INSERT INTO** MenuItems(**ItemID**, **ItemType**, **ItemName**,**Topping**, **Description**)

**VALUES** (1,’ไอศกรีมเค้ก’,’สตรอเบอร์รี่พาราไดซ์’,’สตรอเบอร์รี่และแพ่นช็อกโกแล็ต’,’ไอศกรีม  
เค้ก 4 รสชาติ ประกอบด้วย วนิลลา สตรอเบอร์รี่ชีสเค้ก คุกกี้แอนด์ครีมและเวรี่’);

หรือ

**INSERT INTO** MenuItems

**VALUES**(1,’ไอศกรีมเค้ก’,’สตรอเบอร์รี่พาราไดซ์’,’สตรอเบอร์รี่และแพ่นช็อกโกแล็ต’,’ไอศกรีม  
เค้ก 4 รสชาติ ประกอบด้วย วนิลลา สตรอเบอร์รี่ชีสเค้ก คุกกี้แอนด์ครีมและเวรี่’);

ตาราง MenuItems

<b>ItemID</b>	<b>ItemType</b>	<b>ItemName</b>	<b>Toppings</b>	<b>Description</b>
1	ไอศกรีมเค้ก	สตรอเบอร์รี่ พาราไดซ์	สตรอเบอร์รี่และแพ่นช็อกโกแล็ต	ไอศกรีมเค้ก 4 รสชาติ ประกอบด้วย วนิลลา, สตรอเบอร์รี่ชีสเค้ก, คุกกี้แอนด์ครีม และเวรี่



# การเพิ่มข้อมูลที่หลากหลายแล้วหรือการเพิ่มข้อมูลแบบกลุ่ม มีรูปแบบคำสั่งดังนี้

**INSERT INTO <ชื่อตาราง> [(คอลัมน์ 1, คอลัมน์ 2,...คอลัมน์ N)]  
(คำสั่งสืบคันข้อมูลด้วยคำสั่ง SELECT)**

ตัวอย่างการใช้คำสั่ง

**INSERT INTO MenuItemsBackup  
(SELECT \* FROM MenuItems);**



# ผลลัพธ์

ตาราง MenuItemsBackup

ItemID	ItemType	ItemName	Toppings	Description
1	ไอศกรีมเก๊ก	สครอเบอร์รี่ พาราไคซ์	สครอเบอร์รี่และแผ่นช็อกโกแล็ต	ไอศกรีมเก๊ก 4 รสชาติ ประกอบด้วย วนิลลา, สครอเบอร์รี่ ชีสเก๊ก, คุกเก้แอนด์ครีม และวีร์ชั่นช็อกโกแล็ต
2	ไอศกรีมความหวานรักษ์ช็อกโกแลตทรัฟเฟิล		แผ่นช็อกโกแล็ต	ไอศกรีมดาวรักษ์ช็อกโกแล็ตเข้มข้น ไดร์สช็อกโกแล็ตเต็มๆ เข้มข้นมาก พร้อมชั้นช็อกโกแล็ตทรัฟเฟิล
3	ไอศกรีมความหวานคุกเก้แอนด์ครีม		คุกเก้โอริโอ	ไอศกรีมรสวนิลลาผสมคุกเก้โอริโอ
4	ไอศกรีมเก๊ก	อัลติเมทช็อกโกแลต	แผ่นช็อกโกแลตมาร์บิลและช็อว์ร์ชีสแองสต์	เก๊กไอศกรีม 2 รสชาติ ประกอบด้วย ไอศกรีมหวานนิลลาประปานด้วยไอศกรีมรสสต็อกกี้ชูว์ช็อกโกแลตบนและล่าง วางบนสปอนเจ้กช็อกโกแลตราดด้วยช็อกโกแลตซอฟฟ์ฟิล์



ตาราง MenuItems

ItemID	ItemType	ItemName	Toppings	Description
1	ไอศกรีมเก๊ก	สครอเบอร์รี่ พาราไคซ์	สครอเบอร์รี่และแผ่นช็อกโกแล็ต	ไอศกรีมเก๊ก 4 รสชาติ ประกอบด้วย วนิลลา, สครอเบอร์รี่ ชีสเก๊ก, คุกเก้แอนด์ครีม และวีร์ชั่นช็อกโกแล็ต
2	ไอศกรีมความหวานรักษ์ช็อกโกแลตทรัฟเฟิล		แผ่นช็อกโกแล็ต	ไอศกรีมดาวรักษ์ช็อกโกแล็ตเข้มข้น ไดร์สช็อกโกแล็ตเต็มๆ เข้มข้นมาก พร้อมชั้นช็อกโกแล็ตทรัฟเฟิล
3	ไอศกรีมความหวานคุกเก้แอนด์ครีม		คุกเก้โอริโอ	ไอศกรีมรสวนิลลาผสมคุกเก้โอริโอ
4	ไอศกรีมเก๊ก	อัลติเมทช็อกโกแลต	แผ่นช็อกโกแลตมาร์บิลและช็อว์ร์ชีสแองสต์	เก๊กไอศกรีม 2 รสชาติ ประกอบด้วย ไอศกรีมหวานนิลลาประปานด้วยไอศกรีมรสสต็อกกี้ชูว์ช็อกโกแลตบนและล่าง วางบนสปอนเจ้กช็อกโกแลตราดด้วยช็อกโกแลตซอฟฟ์ฟิล์



- `INSERT INTO EMPLOYEE(ID,NAME,ADDRESS,PHONE)`

`VALUES ('V0004', 'กมล', 'นครราชสีมา', '0991115511');`

- `INSERT INTO EMPLOYEE(ID,ADDRESS,PHONE, NAME)`

`VALUES ('V0005', 'ชัยภูมิ', '0862214444', 'ดวงกมล');`

ID	NAME	ADDRESS	PHONE	E_MAIL
V0004	กมล	นครราชสีมา	0991115511	
V0005	ดวงกมล	ชัยภูมิ	0862214444	



# Ex. INSERT MenuItem

49

ItemID	ItemType	ItemName	Toppings	Description
2	ไอศกรีมครัวท์	ดาร์คช็อคโกแลตทรัฟเฟิล	แผ่นช็อคโกแล็ต	ไอศกรีมดาร์คช็อคโกแล็ต เข้มข้น ได้รสชาติช็อคโกแล็ต เต็ม ๆ เข้มข้นมาก พร้อม ชิ้นช็อคโกแล็ตทรัฟเฟิล
3	ไอศกรีมครัวท์	คุกเก้ แอนครีม	คุกเก้โอริโอ	ไอศกรีมรสวนิลลาผสาน คุกเก้โอริโอ
4	ไอศกรีมเค้ก	อัลมิเมทช์ช็อคโกแลต	แผ่นช็อคโกแลต ตามาร์เบิลและชอร์วี่ส์ แดงสด	เค้กไอศกรีม 2 รสชาติ ประกอบด้วย ไอศกรีมวา นิลลาประกับไอศกรีมรส สตั๊กเก้ ชูวี่ช็อคโกแล็ตบน และล่าง



# Ex. INSERT

50

## PRODUCT

P_CODE	P_NAME	P_INDATE	P_ONHAND	P_IMN	P_PRICE	P_DISCOUNT	ID
1	นามา	2022-1-12	30	5	6	0	V0001
2	น้ำดื่ม	2022-1-2	120	10	12	0	V0001
3	เลย	2022-1-15	320	20	20	0	V0002



# SQL UPDATE Statement

51

- UPDATE : แก้ไขข้อมูลในตาราง

- UPDATE Syntax

UPDATE *Table\_name*

SET *column\_name1*= New Data, *column\_name2*= New  
Data

WHERE *condition*;



Ex. เมื่อต้องการปรับปรุงข้อมูลที่คอลัมน์ `ItemName` จากค่า ‘ดาวร์คช็อกโกแลต ทรัฟเฟิล’ ให้มีค่าเป็น ‘สติคกี้ ชูวี่ ช็อกโกแลต’ ที่คอลัมน์ `ItemID` มีค่าเป็น 2 จากตาราง `MenuItems` สามารถทำได้ด้วยคำสั่งดังนี้

52

## UPDATE `MenuItems`

`SET ItemName = 'สติคกี้ ชูวี่ ช็อกโกแลต'`

`WHERE ItemID = 2;`

ผลลัพธ์

ตาราง `MenuItems`

ItemID	ItemType	ItemName	Toppings	Description
2	ไอศกรีมหวาน	ดาวร์คช็อกโกแลต ทรัฟเฟิล	แผ่นช็อกโกแลต	ไอศกรีมดาวร์คช็อกโกแลตเต็มช้อน ไดร์ฟช็อกโกแลตเต็มๆ เบิ่มชั้นมาก พร้อมชั้นช็อกโกแลตราฟเฟิล

คำสั่ง WHERE

คำสั่ง UPDATE

ItemID	ItemType	ItemName	Toppings	Description
2	ไอศกรีมหวาน	สติคกี้ ชูวี่ ช็อกโกแลต	แผ่นช็อกโกแลต	ไอศกรีมดาวร์คช็อกโกแลตเต็มช้อน ไดร์ฟช็อกโกแลตเต็มๆ เบิ่มชั้นมาก พร้อมชั้นช็อกโกแลตราฟเฟิล



# SQL UPDATE Statement

53

- Example

- UPDATE PRODUCT

```
SET P_MIN = 10
```

```
WHERE P_CODE = '1';
```

- แก้ไขข้อมูลจุดสั่งซื้อ(P\_MIN) โดยให้แก้ไขเฉพาะสินค้าที่มีรหัสสินค้า (P\_CODE) เป็น 1 ให้เป็น 10



# SQL UPDATE Statement

54

- Example

- UPDATE PRODUCT

```
SET P_NAME = 'มันฝรั่งทอดกรอบเลย์'  
WHERE P_CODE = '3';
```

P_CODE	P_NAME	P_INDATE	P_ONHAND	P_IMN	P_PRICE	P_DISCOUNT	ID
1	นามา	2022-1-12	30	5	6	0	V0001
2	น้ำดื่ม	2022-1-2	120	10	12	0	V0001
3	มันฝรั่งทอดกรอบเลย์	2022-1-15	320	20	20	0	V0002



# SQL UPDATE Statement

55

UPDATE EMPLOYEE  
SET Name = “สมใจ”

EMPLOYEE

ID	NAME	ADDRESS	PHONE
V0004	กมล	นครราชสีมา	0991115511
V0005	ดวงกมล	ชัยภูมิ	0862214444

EMPLOYEE

ID	NAME	ADDRESS	PHONE
V0004	สมใจ	นครราชสีมา	0991115511
V0005	สมใจ	ชัยภูมิ	0862214444

คำสั่งจะ ทำให้ชื่อในช่อง Name ของ ทุกคนทุกແກ່ວ ให้เป็น สมใจ ทั้งหมดเลย



# SQL UPDATE Statement

56

UPDATE EMPLOYEE

SET Name = “สมใจ”

WHERE ID = ‘V0005’

EMPLOYEE

ID	NAME	ADDRESS	PHONE
V0004	กมล	นครราชสีมา	0991115511
V0005	ดวงกมล	ชัยภูมิ	0862214444

EMPLOYEE

ID	NAME	ADDRESS	PHONE
V0004	กมล	นครราชสีมา	0991115511
V0005	สมใจ	ชัยภูมิ	0862214444



ถ้าใช้ WHERE ID จะเปลี่ยนผลลัพธ์ตาม ID ตาม ID คนนั้น ๆ



# SQL DELETE Statement

57

- DELETE : ลบแถว (Row) ออกจากตาราง
- DELETE Syntax

DELETE FROM ชื่อตาราง  
WHERE เงื่อนไข;



# SQL DELETE Statement

58

- Example

**DELETE FROM MenuItem**

**WHERE ItemID=2;**

ItemID	ItemType	ItemName	Toppings	Description
1	ไอศกรีมเก้ก	สครอเบอร์รี่ พาราไคซ์	สครอเบอร์รี่และแพนช็อกโก้แล็ค	ไอศกรีมเก้ก 4 รสชาติ ประกอบด้วย วนิลลา, สครอเบอร์รี่ ชีสเค้ก, คุกเก้แอนด์ครีม และเวร์ชั่นช็อกโกแล็คทารฟเฟิล
2	ไอศกรีมควาוח่า	คาร์ช็อกโก้แล็ค ทารฟเฟิล	แพนช็อกโก้แล็ค	ไอศกรีมคาร์ช็อกโก้แล็คเข้มข้น ได้รับชื่อช็อกโกแล็คเต็มๆ เนื้อมาก พร้อมชิ้นช็อกโกแล็คทารฟเฟิล
3	ไอศกรีมราดากุ๊กช็อกโก้เก้วว์	กุ๊กช็อกโก้	กุ๊กช็อกโก้	ไอศกรีมราดากุ๊กช็อกโก้เก้วว์
4	ไอศกรีมเก้ก	อัลติเมทช็อกโก้แล็ค	แพนช็อกโก้แล็คส์แคนดี้สค์	เก้กไอศกรีม 2 รสชาติ ประกอบด้วย ไอศกรีมวนิลลาประยุบค์ด้วยไอศกรีมรสสดตึ๊กกะซูวี่ช็อกโกแล็คบันและล่าง หวานนสปอนจ์เก้กช็อกโก้แล็ค ราดด้วยช็อกโกแล็ค ซอชา ฟีฟี่

คำสั่ง WHERE

คำสั่ง DELETE

ItemID	ItemType	ItemName	Toppings	Description
1	ไอศกรีมเก้ก	สครอเบอร์รี่ พาราไคซ์	สครอเบอร์รี่และแพนช็อกโก้แล็ค	ไอศกรีมเก้ก 4 รสชาติ ประกอบด้วย วนิลลา, สครอเบอร์รี่ ชีสเค้ก, คุกเก้แอนด์ครีม และเวร์ชั่นช็อกโกแล็คทารฟเฟิล
3	ไอศกรีมควาוח่า	กุ๊กช็อกโก้เก้วว์	กุ๊กช็อกโก้	ไอศกรีมราดากุ๊กช็อกโก้เก้วว์
4	ไอศกรีมเก้ก	อัลติเมทช็อกโก้แล็ค	แพนช็อกโก้แล็คมาრ์เบิลและช็อว์ว์ส์ส์แคนดี้สค์	เก้กไอศกรีม 2 รสชาติ ประกอบด้วย ไอศกรีมวนิลลาประยุบค์ด้วยไอศกรีมรสสดตึ๊กกะซูวี่ช็อกโกแล็คบันและล่าง หวานนสปอนจ์เก้กช็อกโก้แล็ค ราดด้วยช็อกโกแล็ค ซอชา ฟีฟี่



# SQL DELETE Statement

59

- **DELETE** : ลบแถว (Row) ออกจากตาราง

- Example

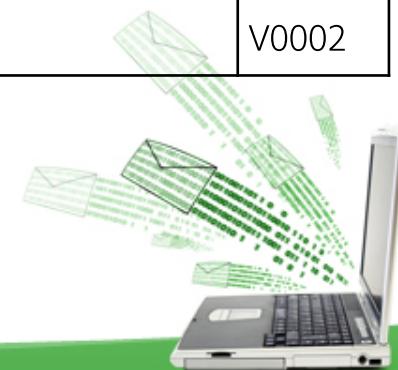
DELETE FROM PRODUCT

WHERE P\_CODE = '1';

DELETE FROM PRODUCT

WHERE P\_MIN = 10;

P_CODE	P_NAME	P_INDATE	P_ONHAND	P_IMN	P_PRICE	P_DISCOUNT	ID
1	มาม่า	2022-1-12	30	5	6	0	V0001
2	น้ำดื่ม	2022-1-2	120	10	12	0	V0001
3	เลียร์	2022-1-15	320	20	20	0	V0002



# SQL DELETE Statement

60

```
DELETE  
FROM EMPLOYEE  
WHERE ID = "V004"
```

Employees

ID	NAME
V001	ไบรชัน
V002	จันทร์ดี
V003	นภา
V004	กมล

Employees

ID	NAME
V001	ไบรชัน
V002	จันทร์ดี
V003	นภา



# SQL DELETE Statement

61

```
DELETE  
FROM EMPLOYEE
```

Employees

ID	NAME
01	ไบรซั่น
02	จันทร์ดี
03	นภา
04	กมล

Employees

ID	NAME



## 2. Data Manipulation Language : DML

62

คำสั่ง SQL ที่ใช้สำหรับการ จัดการหรือจัดการแก้ไขข้อมูล (Manipulate) ที่อยู่ในฐานข้อมูล จะถูกจัดอยู่ในกลุ่มที่เรียกว่า DML หรือ Data Manipulation Language ซึ่งกลุ่มนี้ถือเป็นกลุ่มคำสั่งที่เราต้องใช้งานบ่อยที่สุดในการเขียนโปรแกรม



## 2. Data Manipulation Language : DML

63

### Examples of DML

- ❖ SELECT – ใช้สำหรับคิวรี / เรียกดูข้อมูลในฐานข้อมูล
- ❖ INSERT – ให้สำหรับเพิ่มข้อมูลลงในตาราง
- ❖ UPDATE – ใช้สำหรับอัปเดตข้อมูลในตาราง
- ❖ DELETE – ใช้สำหรับลบข้อมูลในตาราง



## 2. Data Manipulation Language : DML

64

### Tb\_Employees

No	Field Name	Type	Size	PK/FK
1	Emp_ID	Char	3	PK
2	First_name	Varchar	30	
3	Last_name	Varchar	30	
4	Address	Varchar	50	
5	Salary	Int		

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร	5000
004	สมหวัง	ทุกเรื่อง	พะเยา	7800



# DML คำสั่ง SELECT

65

- **SELECT**

The SELECT Statement in SQL is used to retrieve or fetch data from a database

## Syntax

```
SELECT Column1, Column2,...,ColumnN  
FROM Table_name;
```

```
Select First_name, Last_name  
From Employees
```



# DML คำสั่ง SELECT

66

โดยเนพาะการดึง คอลัมน์ที่ต้องการ

USERS

ID	Gender	city	Name	Height	Weight
A01	M	BKK	Jack	170	52
A02	F	TPE	Lila	162	60
A03	M	NY	Tom	180	70
A04	F	BKK	Jahe	165	58
A05	F	NY	Marry	155	48

มีข้อมูลแบบนี้

USERS

Name
Jack
Lila
Tom
Jahe
Marry

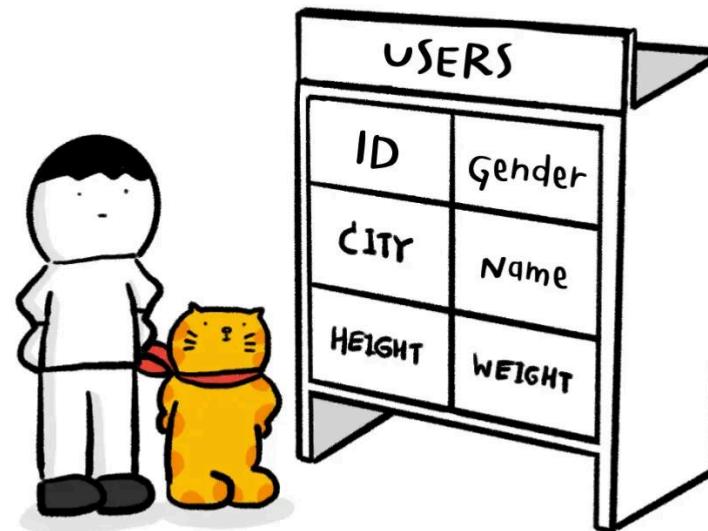
แต่อย่างเดียวมาแค่นี้



# DML คำสั่ง SELECT

67

การดึงคอลัมน์ออกมา  
ทำได้ด้วยคำสั่ง **SELECT**



เนื้อหาเดินไปบนบิบของจากชั้นวาง



# DML คำสั่ง SELECT

68

ลูตร **SELECT** มีน้ำตาดังนี้

คอลัมน์ที่ต้องการ (มีหลายคอลัมน์ได้)

```
SELECT column1, column2, ...
FROM table_name ;
```

ชื่อตาราง  
ที่ต้องการดึงข้อมูล

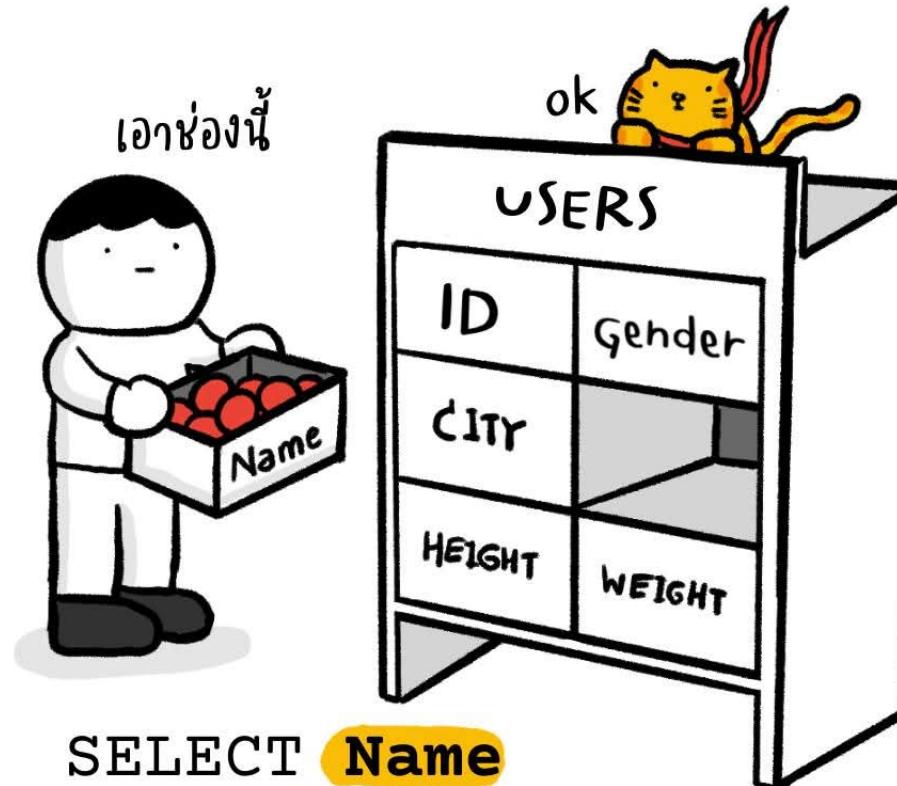
อย่าลืมปิดห้าย  
ด้วยเครื่องหมาย ;



# DML คำสั่ง SELECT

69

เมื่อใช้คำสั่ง **SELECT** ก็เหมือนเรา  
เลือกนับของจากช่องที่ต้องการ



# DML คำสั่ง SELECT

70

```
SELECT Name  
FROM USERS;
```

ผลลัพธ์ที่ได้ น้ำหมาจะเป็นแบบนี้



NAME
Jack
Lila
Tom
Jane
Marry



# DML คำสั่ง SELECT

71

```
SELECT Name , Height  
FROM USERS;
```

ผลลัพธ์ที่ได้ น้ำใจจะเป็นแบบนี้



USERS	
Name	Height
Jack	170
Lila	162
Tom	180
Jane	165
Marry	155



# DML คำสั่ง SELECT

72

TIPS

พิมพ์ \* คือการเลือกทุก colum นั่น

SELECT \*  
FROM **USERS**;

ผลลัพธ์ที่ได้ หน้าตาจะเป็นแบบนี้

↓

USERS					
ID	Gender	city	Name	Height	Weight
A01	M	BKK	Jack	170	52
A02	F	TPE	Lila	162	60
A03	M	NY	Tom	180	70
A04	F	BKK	Jane	165	58
A05	F	NY	Matt	155	48



# DML คำสั่ง SELECT

73

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

❖ Ex.

SELECT First\_name, Last\_name FROM Employees ;

First_name	Last_name
สมชาย	ชาตรี
สมหญิง	งามแท้



# DML คำสั่ง SELECT

74

- To fetch the entire table or all the fields in the table

## Syntax

```
SELECT *
FROM Table_name;
```



# DML คำสั่ง SELECT

75

- To fetch the entire table or all the fields in the table

Ex.

```
SELECT *
FROM Employees;
```



# DML คำสั่ง SELECT

76

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

- ❖ EX. This query will return all the fields in Employee table

```
SELECT * FROM Employees ;
```

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์



# DML คำสั่ง SELECT การใช้ WHERE Clause

77

การใช้ WHERE เงื่อนไขที่สร้างขึ้นเพื่อกรองกรองข้อมูลแต่ละแถวที่ต้องการ

## ❖ SQL | WHERE Clause

### • Syntax

```
SELECT Column1, Column2,...,ColumnN
      FROM Table_name
      WHERE condition ;
```



# DML คำสั่ง SELECT การใช้ WHERE Clause

78

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

- EX. To fetch field First\_name Last\_name Address where address in อุตรดิตถ์

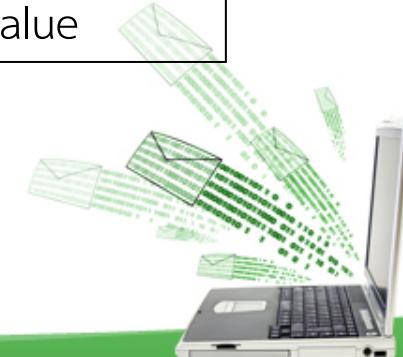
```
SELECT First_name, Last_name, Address  
FROM Employees  
WHERE Address = 'อุตรดิตถ์';
```

Address คอลัมน์ที่เป็นเงื่อนไข

= คือ operator

อุตรดิตถ์ คือ value

First_name	Last_name	Address
สมหญิง	งามแท้	อุตรดิตถ์



# DML - ตัวดำเนินการ ในการใช้ WHERE

79

Operator	Description	ภาษาไทย
>	Greater Than	มากกว่า
>=	Greater than or Equal to	มากกว่าหรือเท่ากับ
<	Less Than	น้อยกว่า
<=	Less than or Equal to	น้อยกว่าหรือเท่ากับ
=	Equal to	เท่ากับ/เป็นหรือตรงกับเงื่อนไข
<>	Not Equal to	ไม่เท่ากับเงื่อนไข



# DML - ตัวดำเนินการ ในการใช้ WHERE

80

- แสดงข้อมูล ชื่อ สกุล ที่อยู่ เงื่อนไข เฉพาะ ที่อยู่ ใน จ. เชียงใหม่
- แสดง ชื่อ สกุล ที่อยู่ เงินเดือน เงื่อนไข เฉพาะเงินเดือนตั้งแต่ 7000 บาท

```
Select first_name, Last_name, address, salary  
From employees  
Where salary >= 7000
```

- แสดง ทุกฟิลด์ เงื่อนไข เงินเดือนน้อยกว่า 10000 บาท (ลองทำดู)

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร	5000
004	สมหวัง	ทุกเรื่อง	พะ夷า	7800



# DML - ตัวดำเนินการ ในการใช้ WHERE

81

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

- EX. To fetch field First\_name Last\_name Address where address in อุตรดิตถ์ หรือ เชียงใหม่

```
SELECT First_name,Last_name,Address  
FROM Employees  
WHERE Address = 'อุตรดิตถ์' or Address = 'เชียงใหม่';
```

Address คอลัมน์ที่เป็นเงื่อนไข

= คือ operator

อุตรดิตถ์ คือ value

First_name	Last_name	Address
สมหญิง	งามแท้	อุตรดิตถ์



# DML - ตัวดำเนินการ ในการใช้ WHERE

82

Operator	Description
AND	ตรวจสอบเงื่อนไขต้องเป็นจริงทั้งคู่
OR	ตรวจสอบเงื่อนไข เป็นจริงอย่างใดอย่างหนึ่งก็ได้
Not	ตรวจสอบเงื่อนไขต้องไม่เป็นหรือต้องไม่ใช่



# DML - ตัวดำเนินการ ในการใช้ WHERE

83

Not T is F

Not F is T

T	and	T	T
T	and	F	F
F	and	T	F
F	and	F	F

T	or	T	T
T	or	F	T
F	or	T	T
F	or	F	F



# DML - Try It ลองทำดู

84

- ขอดู ทุกฟิลด์ เงื่อนไข คือ อยู่ใน เชียงใหม่ และ มีเงินเดือนมากกว่า 7000
- ขอดู ทุกฟิลด์ เงื่อนไข คือ อยู่ใน เชียงใหม่หรือพะเยา และมีเงินเดือนน้อยกว่า 10000 บาท



# DML - Special operators

85

Operator	Description
BETWEEN ...AND...	อยู่ระหว่างค่าสองค่า (รวมหัวและท้ายด้วย)
IN(set)	ตรงกับค่าใดค่าหนึ่งที่อยู่ในตาราง ที่ระบุไว้
LIKE	ตรงกับรูปของตัวอักษร
IS NULL	เป็นค่าว่าง ไม่มีข้อมูล

```
select emp_name, last_name, salary  
from employee  
where salary in (20000, 50000)
```

เลือกแสดงชื่อชื่อ นามสกุล และเงินเดือน จากตารางพนักงาน โดยเลือกเฉพาะคนที่  
มีเงินเดือนเท่ากับ 20,000 หรือ 50,000 บาท เท่านั้น



# DML - AND

86

```
SELECT First_name, Last_name, Address, Salary  
FROM Employees  
WHERE Address = 'อุตรดิตถ์' AND Salary > 5000;
```

*Result*

First_name	Last_name	Address	Salary
สมหญิง	งามแท้	อุตรดิตถ์	6000



# DML - OR

87

ใช้ต่อเมื่อต้องการข้อมูลที่มีเงื่อนไขเดียวกันเป็นจริง

```
SELECT First_name, Last_name, Address, Salary  
FROM Employees  
WHERE Address = 'อุตรดิตถ์' OR Salary > 5000;
```

*Result*

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000



# การใช้ Between...AND....

88

- ใช้ต่อเมื่อต้องการข้อมูลลักษณะที่เป็นช่วง และตัวดำเนินการนี้มักจะร่วมใช้กับตัวดำเนินการ AND  
เสมอ <กำหนดเงื่อนไขให้กับ Where โดยเลือกช่วงข้อมูลที่สนใจ>

## •Systax

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name Between Value1 AND value2 ;
```



Lower limit



Upper limit



# ตัวอย่างการใช้ Between...AND....

- ๘๙ ตัวอย่าง แสดงชื่อ และเงินเดือนของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่มีเงินเดือน 6000 ถึง 15000

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000

```
SELECT First_name, Salary  
FROM Employees  
WHERE Salary between 6000 AND 15000;
```

First_name	Salary
สมชาย	15000
สมหญิง	6000



# ตัวอย่างการใช้ NOT Between...AND....

90

การใช้ Not ร่วมกับ Between....AND... เป็นการระบุเงื่อนไขว่า ข้อมูลต้องไม่อยู่ในช่วงที่กำหนด

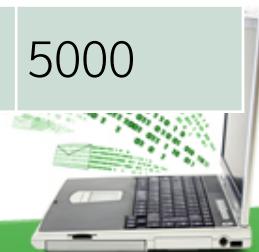
Employees

	Emp_ID	First_name	Last_name	Address	Salary
	001	สมชาย	ชาตรี	เชียงใหม่	15000
	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร่	5000

```
SELECT First_name, Salary  
FROM Employees  
WHERE Salary Not between 6000 AND 15000;
```

ตัวอย่าง แสดงชื่อ และเงินเดือนของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่มีเงินเดือน ไม่อยู่ในช่วง 6000 ถึง 15000

First_name	Salary
สมใจ	5000



# การใช้ LIKE Condition

91

คำสั่ง LIKE หรือ LIKE Operator เป็นคำสั่งที่จะถูกนำไปใช้ต่อจากคำสั่ง WHERE ในภาษา SQL เพื่อค้นหาข้อมูลตามรูปแบบที่เราต้องการ สำหรับคำสั่ง LIKE จะมี Wildcards ( สัญลักษณ์ ) อยู่ 2 ตัวที่มักจะใช้งานร่วมกัน คือ

- % Percent เครื่องหมายเปอร์เซ็นต์ แทนค่าเท่ากับอักษรหนึ่งตัว หรือมากกว่า
- \_ Underscore เครื่องหมายขีดเส้นใต้ แทนค่าเท่ากับอักษรหนึ่งตัวเท่านั้น

## ■ Systax

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name Like pattern ;
```



# การใช้ LIKE Condition

92

Format	Ex.	Description	Result
'%value'	'%S'	ตัวอักษรด้านหน้าเป็นอะไรก็ได้ ตัวสุดท้ายต้องเป็น S	Sandnes , Louis
'value%	'N%'	ข้อความนั้นต้องขึ้นด้วยตัว N	Nerissa
'%value%	'%a%'	ข้อความต้องมี a ประกอบอยู่ในข้อความ	Baer, Louisa
'_ value%	'_ a%'	ในข้อความ ตัวอักษรตัวแรกเป็นอะไรก็ได้ ขอให้ตัวอักษรตัวที่ 2 เป็น a	Baida,Nayer
'%value__'	'%a__'	ในข้อความ ตัวอักษรสองตัวหลังจะเป็นอะไรก็ได้ แต่ตัวที่สาม (นับจากด้านหลัง) ต้องเป็น a	Rajs, Hall



# DML - LIKE Condition

93

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name Like pattern ;
```

The following SQL statement selects all customers with a CustomerName that Like pattern

```
select First_name,Last_name
from Employee
where First_name like 'A%'
```



# DML - NOT LIKE Condition

94

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name NOT Like pattern ;
```

The following SQL statement selects all customers with a CustomerName that Like pattern

```
select First_name,Last_name
from Employee
where First_name Not like ‘%’
```



# การใช้ IN

95

IN เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (TABLE) โดยเลือกเฉพาะค่าที่กำหนด

## Syntax

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column IN (value1,value2,...)
```



# การใช้ IN

96

Employees

	Emp_ID	First_name	Last_name	Address	Salary
	001	สมชาย	ชาตรี	เชียงใหม่	15000
	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร่	5000
	004	สมหวัง	ทุกเรื่อง	พะเยา	7800

❖ Ex.

```
SELECT Emp_ID, First_name  
FROM Employees  
WHERE Emp_ID IN ('001', '002','004')
```

Emp_ID	First_name
001	สมชาย
002	สมหญิง
004	สมหวัง



# การใช้ NOT IN Condition

97

Employees

	Emp_ID	First_name	Last_name	Address	Salary
❖	001	สมชาย	ชาตรี	เชียงใหม่	15000
	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร่	5000
	004	สมหวัง	ทุกเรื่อง	พะเยา	7800

```
SELECT Emp_ID, First_name  
FROM Employees  
WHERE Emp_ID Not IN ('001', '002','004')
```

Emp_ID	First_name
003	สมใจ



# DML - IS NULL Condition

98

คำสั่ง IS NULL ใช้สำหรับค้นหาแล้วที่มีค่าว่าง (NULL) ในคอลัมน์ที่ระบุ โดยค่า NULL หมายถึง ไม่มีข้อมูล หรือ ไม่ทราบค่า

**IS NULL** Test for nulls with the IS NULL operator

Syntax

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name IS NULL
```



# DML - IS NULL Condition

99

Employees

	Emp_ID	First_name	Last_name	Address	Salary
	001	สมชาย	ชาตรี	เชียงใหม่	15000
❖	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร	5000
	004	สมหวัง	ทุกเรื่อง	พะเยา	7800

```
SELECT First_name, Address  
FROM Employees  
WHERE Address IS NULL
```

First_name	Address
สมใจ	
สมหวัง	



# DML - IS NOT NULL Condition

100

Test for Not nulls with the IS NOT NULL operator

Syntax

```
SELECT Column1, Column2,...,ColumnN
FROM Table_name
WHERE Column_name IS NOT NULL
```



# DML - IS NOT NULL Condition

101

Employees

	Emp_ID	First_name	Last_name	Address	Salary
	001	สมชาย	ชาตรี	เชียงใหม่	15000
	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร่	5000
	004	สมหวัง	ทุกเรื่อง	พะเยา	7800

```
SELECT First_name, Address  
FROM Employees  
WHERE Address IS NOT NULL
```

First_name	Address
สมชาย	เชียงใหม่
สมหญิง	อุตรดิตถ์



# การใช้ DISTINCT

102

DISTINCT ใช้แสดงข้อมูล หากซ้ำกันจะแสดงเพียงครั้งเดียว

## Syntax

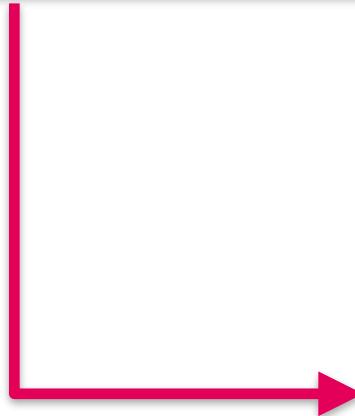
```
SELECT DISTINCT ชื่อคอลัมน์  
FROM ชื่อตาราง
```



# ตัวอย่างการใช้ DISTINCT

103

■ SELECT DISTINCT V\_CODE  
FROM PRODUCT;



V_CODE
V0001
V0003
V0004
V0006
V0008
► V0011



# ตัวอย่างการใช้ DISTINCT

104

■ SELECT DISTINCT V\_CODE  
FROM PRODUCT;

P_CODE	P_NAME	V_CODE
P01	Mouse	V0001
P02	Keyboard	V0001
P03	Monitor	V0001
P04	Laptop	V0004
P05	USB HUB	V0006
P06	Cable	V0003
P07	Webcam	V0008
P08	Headset	V0011
P09	Speaker	V0006



V_CODE
V0001
V0003
V0004
V0006
V0008
V0011

V_CODE
V0001
V0003
V0004
V0006
V0008
V0011



# ตัวอย่างการใช้ DISTINCT

105

**SELECT DISTINCT Amphur  
FROM Personal**

Personal

FirstName	LastName	Amphur
มานะ	พากเพียร	เมือง
อดทน	ตั้งใจเรียน	เมือง
มานี	หมันเพียร	ปง

ผลลัพธ์

Amphur
เมือง
ปง



# การใช้ ORDER BY

106

ORDER BY คือ การจัดกลุ่มข้อมูล (Group) การเรียงลำดับข้อมูล (Order) เช่น **มากไปหาน้อย (DESC)** หรือ **น้อยไปมาก (ASC)**

```
SELECT columnlist
      FROM tablelist
      [WHERE conditionlist]
      [ORDER BY Columnlist [
          ASC | DESC ]];
```



# ตัวอย่างการใช้ ORDER BY

107

```
SELECT *
FROM Address
ORDER BY ZipCode DESC
```

Address

Amphur	ZipCode
เมืองพะเยา	56000
จุน	56002
ปง	56001



ผลลัพธ์

Amphur	ZipCode
จุน	56002
ปง	56001
เมืองพะเยา	56000



# ตัวอย่างการใช้ ORDER BY

108

Select Id, Name, Addr, Curr\_Bal

From customer order by Curr\_Bal desc ;

<u>Id</u>	<u>Name</u>	<u>Addr</u>	<u>Curr_Bal</u>
197	วรชาติ สีคล้ำ	อยุธยา	500000
110	ศรี สุขพานิช	กรุงเทพฯ	200000
217	อนันต์ บุญญาณุพงศ์	กรุงเทพฯ	200000
309	สุภาวดี เพชรสุข	ระยอง	150000
100	โสภา สีคล้ำ	กรุงเทพฯ	100000
...	...	...	...



# ตัวอย่างการใช้ ORDER BY

109

• Ex.

- ```
SELECT P_CODE, P_DESCRIP, P_INDATE, P_PRICE  
      FROM PRODUCT  
  ORDER BY P_PRICE;
```
- ```
SELECT P_CODE, P_DESCRIP, P_INDATE, P_PRICE  
      FROM PRODUCT  
  ORDER BY P_PRICE DESC;
```



# ตัวอย่างการใช้ ORDER BY

110

- ```
SELECT P_DESCRIP, V_CODE, P_INDATE, P_PRICE
      FROM PRODUCT
      ORDER BY V_CODE ASC, P_PRICE DESC;
```
- ```
SELECT P_DESCRIP, V_CODE, P_PRICE
      FROM PRODUCT
     WHERE P_PRICE <= 500
      ORDER BY V_CODE ASC, P_PRICE DESC;
```



# การใช้ ALIAS

111

ALIAS เป็นคำสั่งที่ใช้ระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดย ALIAS คือการสร้างชื่อจำลองขึ้นมาใหม่ โดยสามารถจำลองได้ทั้งชื่อ Field และชื่อ Table

```
SELECT columnlist
      FROM tablelist
      [WHERE conditionlist]
      [ORDER BY Columnlist [
          ASC | DESC ]];
```



# การใช้ ALIAS

112

## Syntax

```
SELECT Column1 AS New_Column_name, Column 2,...  
FROM Table_name
```

```
SELECT Emp_ID as รหัสพนักงาน, first_name ชื่อ, last_name สกุล FROM employees
```

```
SELECT Emp_ID as รหัสพนักงาน, first_name ชื่อ, last_name สกุล  
FROM employees  
WHERE ชื่อ LIKE 'สมชาย'
```



# ตัวอย่าง การใช้ ALIAS

113

Employees

	Emp_ID	First_name	Last_name	Address	Salary
	001	สมชาย	ชาตรี	เชียงใหม่	15000
	002	สมหญิง	งามแท้	อุตรดิตถ์	6000
	003	สมใจ	สุขสม	แพร'	5000

```
SELECT Emp_ID AS ID, First_name  
FROM Employees  
WHERE Salary between 6000 AND 15000;
```

ID	First_name
001	สมชาย
002	สมหญิง
003	สมใจ



# การใช้ นิพจน์ทางคณิตศาสตร์

114

การนำตัวดำเนินการพื้นฐาน มาใช้ในการคำนวณค่าตัวเลขภายในคำสั่ง SQL โดยเฉพาะอย่างยิ่ง กับ SELECT , UPDATE หรือ WHERE เพื่อหาผลรวม ผลต่าง ค่าเฉลี่ย หรือ เปอร์เซนต์

Arithmetic	Description
+	Add
-	Subtract
*	Multiply
/	Divide

$$5 + 10 * 2 = 25$$

$$(5 + 20) / 2 * 5 - 10 =$$



# ตัวอย่างการใช้ นิพจน์ทางคณิตศาสตร์

115

- `SELECT P_DESCRIP, P_ONHAND, P_PRICE,  
P_ONHAND*P_PRICE  
FROM PRODUCT;`



	P_DESCRIP	P_ONHAND	P_PRICE	Expr1003
▶	หัวพนัก	8	1090.99	8727.92
	ใบเลื่อย 7.25 นิ้ว	32	140.99	4511.68
	ใบเลื่อย 9 นิ้ว	18	170.49	3068.82
	แผ่นเหล็กหนา 1/4 นิ้ว	23	390.95	8991.85
	แผ่นเหล็กหนา 1/2 นิ้ว	23	430.99	9912.77
	เลื่อยจีกซอร์ 12 นิ้ว	8	1090.92	8727.36
	เลื่อยจีกซอร์ 8 นิ้ว	6	990.87	5945.22
	ส่วนไรัสาย	12	380.95	4571.4
	ตัวน	23	90.95	2091.85
	ตัวน 12 ปอนต์	8	140.4	1123.2
	ตะไบ	43	40.99	1762.57
	เลื่อย Hicut 16 นิ้ว	11	2560.99	28170.89
	ห่อ PVC 3.5 นิ้ว	188	50.87	9563.56
	สกรู 1.25 นิ้ว	172	60.99	10490.28
	สกรู 2.5 นิ้ว	237	80.45	19066.65
	ตาข่ายเหล็ก 4 x 8 ฟุต	18	1190.95	21437.1
*				



# ตัวอย่างการใช้ Ex. Using Column Aliases

116

- ```
SELECT P_DESCRIP, P_ONHAND, P_PRICE,  
P_ONHAND*P_PRICE AS TOTAL_VALUE  
FROM PRODUCT;
```
- ```
SELECT P_CODE, P_INDATE, P_INDATE + 90 AS  
EXPIRE_DATE  
FROM PRODUCT;
```



# การใช้ ตัวดำเนินการ Function

117

## • Course

- COUNT ใช้สำหรับนับค่า
- MIN หาค่าต่ำสุด
- MAX หาค่าสูงสุด
- SUM หาผลรวม
- AVG หาค่าเฉลี่ย



# ตัวอย่างการใช้ ตัวดำเนินการ Function

118

- COUNT : ใช้สำหรับการนับค่า

- Ex.

- SELECT COUNT(\*)  
FROM VENDOR;

Output 5

V_code	name
1001	นครราชสีมา
1002	บุรีรัมย์
1001	นครราชสีมา
1004	ชัยภูมิ



# ตัวอย่างการใช้ ตัวดำเนินการ Function

119

• Ex.

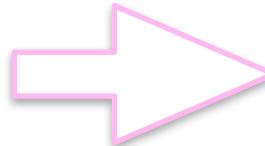
- `SELECT COUNT(*)`

`FROM`

`(SELECT DISTINCT V_CODE FROM PRODUCT  
WHERE V_CODE IS NOT NULL);`

Output 3

V_code	name
1001	นครราชสีมา
1002	บุรีรัมย์
1001	นครราชสีมา
1004	ชัยภูมิ



V_code
1001
1002
1004



# ตัวอย่างการใช้ ตัวดำเนินการ Function

120

• Ex.

- `SELECT COUNT(*)`

- `FROM`

- `(SELECT DISTINCT V_CODE FROM PRODUCT  
WHERE V_CODE IS NOT NULL  
AND P_PRICE < 100);`



# ตัวคำแนะนำ FUNCTION MAX

121

- MAX : หาค่าสูงสุด

- Ex.

- SELECT MAX(P\_PRICE)

- FROM PRODUCT;

- Output 120

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



P_price
120



# ตัวคำแนะนำการ FUNCTION MAX

122

## • MAX : หาค่าสูงสุด

```
• SELECT v_CODE, name, P_PRICE  
      FROM PRODUCT  
 WHERE P_PRICE = (SELECT MAX(P_PRICE)  
      FROM PRODUCT);
```

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



V_code	name	P_price
1002	ข้าวผัดทะเล	120



# ตัวคำแนะนำการ FUNCTION MIN

123

- MIN : หาค่าต่ำสุด

- Ex.

- SELECT MIN(P\_PRICE)

- FROM PRODUCT;

- Output 70

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



P_price
70



# ตัวคำแนะนำการ FUNCTION MIN

124

## • MIN : หาค่าต่ำสุด

```
• SELECT v_CODE, name, P_PRICE  
      FROM PRODUCT  
     WHERE P_PRICE = (SELECT MIN(P_PRICE)  
      FROM PRODUCT);
```

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



V_code	name	P_price
1004	ผัดไทย	70



# ตัวดำเนินการ FUNCTION SUM

125

## • SUM : หาผลรวม

- SELECT SUM(P\_ONHAND \* P\_PRICE) AS TOTAL\_VALUE  
FROM PRODUCT;
- SELECT SUM(P\_PRICE) FROM PRODUCT;

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



P_price
350



# ตัวดำเนินการ FUNCTION SUM

126

- SUM : หาผลรวม

```
SELECT Name, SUM(Amount)  
FROM Income
```

INCOME

Name	Amount
ANT	5500
BAT	4500
ANT	7100

ผลลัพธ์

Name	SUM(Amount)
ANT	17100
BAT	17100
ANT	17100



# ตัวดำเนินการ FUNCTION AVG

127

## • AVG : หาค่าเฉลี่ย

• SELECT AVG(P\_PRICE)

FROM PRODUCT;

V_code	name	P_price
1001	กระเพรา	80
1002	ข้าวผัดทะเล	120
1001	กระเพรา	80
1004	ผัดไทย	70



P_price
87.5



# ตัวดำเนินการ FUNCTION AVG

128

## • AVG : หาค่าเฉลี่ย

```
• SELECT P_DESCRIFT, P_ONHAND, P_PRICE  
      FROM PRODUCT WHERE P_PRICE > (SELECT AVG(P_PRICE)  
      FROM PRODUCT) ORDER BY P_PRICE DESC;
```

P_CODE	P_DESCRIFT	P_ONHAND	P_PRICE
1001	กระเพราไก่	10	80
1002	ข้าวผัดทะเล	5	120
1003	ต้มยำกุ้ง	3	150
1004	ผัดไทย	12	70
1005	ข้าวไข่เจียว	20	50



P_DESCRIFT	P_ONHAND	P_PRICE
ต้มยำกุ้ง	3	150
ข้าวผัดทะเล	5	120



# การใช้ GROUP BY และ HAVING

129

คำสั่งใน SQL ใช้ร่วมกันเพื่อจัดกลุ่มข้อมูล (Grouping) และกรองกลุ่มข้อมูล (Filtering Grouped Data)

- GROUP BY ใช้เพื่อจัดกลุ่มข้อมูลที่เหมือนกันในคอลัมน์ที่ระบุ เช่น แบ่งกลุ่มตามหมวดหมู่สินค้า
- HAVING ใช้กรองกลุ่มข้อมูล โดยเป็นเงื่อนไขที่ใช้กับค่าที่รวมแล้ว เช่น ต้องการดูเฉพาะรายการสินค้าที่มียอดขายเฉลี่ยมากกว่า 1500
- คำสั่งที่มักใช้ร่วมกัน SELECT, FROM, WHERE (กรองข้อมูลก่อนจัดกลุ่ม) GROUP BY, HAVING (กรองข้อมูลหลังจัดกลุ่ม), ORDER BY

```
SELECT columnlist
FROM tablelist
      [WHERE conditionlist]
      [GROUP BY columnlist]
      [HAVING conditionlist]
      [ORDER BY columnlist [ASC | DESC]];
```



# ตัวอย่างการใช้ GROUP BY

130

ตัวอย่าง

```
SELECT Name, SUM(Amount)  
FROM Income  
GROUP BY Name
```

INCOME

Name	Amount
ANT	5500
BAT	4500
ANT	7100

ผลลัพธ์

Name	SUM(Amount)
ANT	12600
BAT	4500



# ตัวอย่างการใช้ GROUP BY และ HAVING

131

ตัวอย่าง

```
SELECT Name, SUM(Amount)  
FROM Income  
GROUP BY Name  
HAVING NAME = "ANT"
```

INCOME

Name	Amount
ANT	5500
BAT	4500
ANT	7100

ผลลัพธ์

Name	SUM(Amount)
ANT	12600



# ตัวอย่างการใช้ GROUP BY และ HAVING

132

ตัวอย่าง

```
SELECT Name, SUM(Amount)  
FROM Income  
GROUP BY Name  
HAVING NAME = "ANT"
```

INCOME

Name	Amount
ANT	5500
BAT	4500
ANT	7100

ผลลัพธ์

Name	SUM(Amount)
ANT	12600



# ตัวอย่างการใช้ GROUP BY และ HAVING

133

## ตัวอย่าง

หาผลรวมของ (ราคาสินค้า \* จำนวนคงเหลือ) โดยจัดกลุ่มตามรหัสผู้ผลิต  
และไม่เอาแถวที่ไม่มีผู้ผลิต

```
• SELECT V_CODE, SUM(P_PRICE * P_ONHAND)  
      FROM PRODUCT  
     WHERE V_CODE <> NULL  
   GROUP BY V_CODE;
```

PRODUCT

P_CODE	P_DESCRIP	P_ONHAND	P_PRICE	V_CODE
1001	ผงซักฟอก	100	50	V01
1002	น้ำยาปรับผ้านุ่ม	50	80	V01
1003	สบู่	200	20	V02
1004	ยาสีฟัน	30	120	V02
1005	แปรงสีฟัน	10	45	NULL



V_CODE	SUM
V01	9000
V02	7600



# ตัวอย่างการใช้ GROUP BY และ HAVING

134

## ตัวอย่าง

หาค่าเฉลี่ยของราคางานค้า โดยจัดกลุ่มตามรหัสผู้ผลิต และไม่เอาแถวที่ไม่มีผู้ผลิต

```
• SELECT V_CODE, AVG(P_PRICE)  
  FROM PRODUCT  
 WHERE V_CODE <> NULL  
 GROUP BY V_CODE;
```

PRODUCT

P_CODE	P_DESCRIP	P_ONHAND	P_PRICE	V_CODE
1001	ผงซักฟอก	100	50	V01
1002	น้ำยาปรับผ้านุ่ม	50	80	V01
1003	สบู่	200	20	V02
1004	ยาสีฟัน	30	120	V02
1005	แปรงสีฟัน	10	45	NULL



V_CODE	AVG
V01	65
V02	70



# ตัวอย่างการใช้ GROUP BY และ HAVING

135

## ตัวอย่าง

หากค่าเฉลี่ยของราคาสินค้า โดยจัดกลุ่มตามรหัสผู้ผลิต และไม่เอาพวกที่ไม่มีผู้ผลิต  
เงื่อนไข "ต้องเป็นกลุ่มที่มีค่าเฉลี่ยราคาน้อยกว่า 100 "

```
• SELECT V_CODE, AVG(P_PRICE)  
      FROM PRODUCT  
      WHERE V_CODE <> NULL  
      GROUP BY V_CODE  
      HAVING AVG(P_PRICE) < 100;
```

PRODUCT

P_CODE	P_DESCRIP	P_ONHAND	P_PRICE	V_CODE
1001	ผงซักฟอก	100	50	V01
1002	น้ำยาปรับผ้านุ่ม	50	80	V01
1003	สบู่	200	20	V02
1004	ยาสีฟัน	30	120	V02
1005	แปรงสีฟัน	10	45	NULL



V_CODE	AVG
V01	65
V02	70



# ตัวอย่างการใช้ GROUP BY และ HAVING

136

## ตัวอย่าง

หาผลรวมของ (ราคาสินค้า \* จำนวนคงเหลือ) โดยจัดกลุ่มตามรหัสผู้ผลิต  
และไม่เอาแถวที่ไม่มีผู้ผลิต เรียงลำดับจากมูลค่ามากไปน้อย

```
• SELECT V_CODE,  
        SUM(P_ONHAND * P_PRICE) AS TOTAL_PRICE  
    FROM PRODUCT  
   WHERE V_CODE <> NULL  
GROUP BY V_CODE  
ORDER BY SUM(P_ONHAND * P_PRICE) DESC;
```

PRODUCT

P_CODE	P_DESCRIP	P_ONHAND	P_PRICE	V_CODE
1001	ผงซักฟอก	100	50	V01
1002	น้ำยาปรับผ้านุ่ม	50	80	V01
1003	สบู่	200	20	V02
1004	ยาสีฟัน	30	120	V02
1005	แปรงสีฟัน	10	45	NULL



V_CODE	AVG
V01	9000
V02	7600



# ตัวอย่างการใช้ GROUP BY และ HAVING

137

## ตัวอย่าง

หาผลรวมของ (ราคาสินค้า \* จำนวนคงเหลือ) โดยจัดกลุ่มตามรหัสผู้ผลิต  
และไม่เอาแถวที่ไม่มีผู้ผลิต เรียงลำดับจากมูลค่ามากไปน้อย

```
• SELECT V_CODE,  
        SUM(P_ONHAND * P_PRICE) AS TOTAL_PRICE  
    FROM PRODUCT  
   WHERE V_CODE <> NULL  
GROUP BY V_CODE  
ORDER BY SUM(P_ONHAND * P_PRICE) DESC;
```

PRODUCT

P_CODE	P_DESCRIP	P_ONHAND	P_PRICE	V_CODE
1001	ผงซักฟอก	100	50	V01
1002	น้ำยาปรับผ้านุ่ม	50	80	V01
1003	สบู่	200	20	V02
1004	ยาสีฟัน	30	120	V02
1005	แปรงสีฟัน	10	45	NULL



V_CODE	AVG
V01	9000
V02	7600



# การดึงข้อมูลหลายตาราง Join Table

138

Department

Code_dep	Depname
is	Information System

Teacher

Tec_id	Tec_name	Code_dep
t01	yoyo	Is
t02	paipai	Is

Course

C-id	C_name	Tec_id
111	IT	T01
112	Programming	T02

Student

s_id	s_name	C_id
6401	somchai	111
6401	wanna	112



# การดึงข้อมูลหลายตาราง Join Table

139

Department

Code_dep	Depname
is	Information System

Select teacher.tec\_name,student.s\_name  
From teacher natural join course  
natural join student

Teacher

Tec_id	Tec_name	Code_dep
t01	yoyo	Is
t02	paipai	Is

Course

C-id	C_name	Tec_id
111	IT	T01
112	Programming	T02

Select d.depname, t.tec\_name,s.s\_name  
From department d join teacher t  
using(code\_dep)  
join course c using( tea\_id)  
join student using(c\_id)

Student

s_id	s_name	C_id
6401	somchai	111
6401	wanna	112



# คำสั่ง SQL สำหรับตัวดำเนินการ JOIN

140

เป็นคำสั่ง SQL ขั้นสูง

- การดำเนินการ inner join ด้วยคำสั่ง NATURAL JOIN
- การดำเนินการ inner join ด้วยคำสั่ง USING
- การดำเนินการ inner join ด้วยคำสั่ง ON



# Joining Tables Using SQL:1999 Syntax

141

เป็นคำสั่ง SQL ขั้นสูง

- การดำเนินการ inner join ด้วยคำสั่ง NATURAL JOIN
- การดำเนินการ inner join ด้วยคำสั่ง USING
- การดำเนินการ inner join ด้วยคำสั่ง ON

```
SELECT table1.column, table2.column
FROM   table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

# ลองทำดู ฐานข้อมูลชื่อ campus

142

เพื่อใช้ในการเรียน inner join

*Campusworld*

Campus_ID	Campus_Name	Location
101	Main Campus	Bangkok
102	IT Center	Chonburi

*Room*

Room_ID	Room_Name	Building_Code
R101	Network Lab	B03
R202	Library	B01

*Building*

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

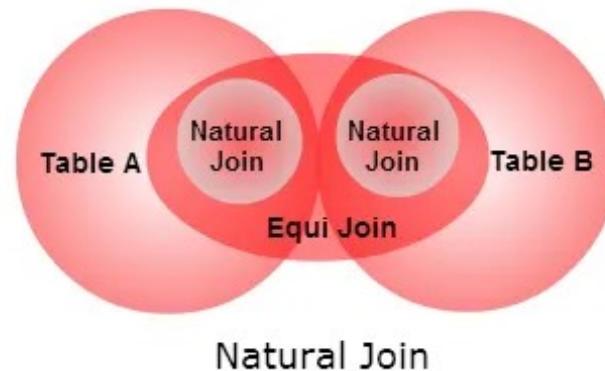


# การ inner join ด้วยคำสั่ง NATURAL JOIN

143

- ค้นหาแอทริบิวท์มีชื่อเหมือนกันทั้งสองตาราง
- ทำการ SELECT เฉพาะแผลข้อมูลจากทั้งสองตารางที่มีค่าข้อมูลที่ปรากฏในแอทริบิวท์มีชื่อตรงกันและเหมือนกัน
- ทำการลบแอทริบิวท์เหมือนกันออกแอทริบิวหนึ่งเพื่อลดความซ้ำซ้อนของแอทริบิว

```
SELECT COLUMN-LIST  
FROM table1 NATURAL JOIN table2;
```



# ตัวอย่างการใช้ ด้วยคำสั่ง NATURAL JOIN

144

## ตัวอย่าง

แสดง ชื่อวิทยาเขต ชื่ออาคาร ที่ตั้ง โดยดึงจากตาราง Campusworld และ Building

```
SELECT Campus_Name, Build_Name, Location  
FROM Campusworld NATURAL JOIN Building
```

Campusworld

Campus_ID	Campus_Name	Location
101	Main Campus	Bangkok
102	IT Center	Chonburi

Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

ผลลัพธ์

Campus_Name	Build_Name	Location
Main Campus	Engineering Hall	Bangkok
Main Campus	Science Lab	Bangkok
IT Center	Information Bldg	Chonburi



# ตัวอย่างการใช้ ด้วยคำสั่ง NATURAL JOIN

145

```
SELECT d.department_id, d.department_name,  
       l.location_id, l.city  
FROM   departments as d  
NATURAL JOIN locations l ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford



# การ inner join ด้วยคำสั่ง USING

146

- แก้ปัญหาเมื่อชื่อคอลัมน์ตรงกันแต่ Data Type ต่างกัน: ปกติถ้าใช้ NATURAL JOIN ระบบจะพยายามจับคู่ทุกคอลัมน์ที่มีชื่อเหมือนกันให้โดยอัตโนมัติ แต่ถ้าคอลัมน์เหล่านั้นมีชนิดข้อมูล (Data Type) ไม่ตรงกันจะเกิด Error เราจึงต้องเปลี่ยนมาใช้ USING เพื่อ "เจาะจง" เลือกเฉพาะคอลัมน์ที่เราต้องการใช้ทำ Equijoin (การเชื่อมด้วยค่าที่เท่ากัน)
- ใช้เพื่อเลือกเฉพาะบางคอลัมน์: ในกรณีที่มีหลายคอลัมน์ชื่อเหมือนกัน แต่เราอยากระบุต้องการจะใช้มันต่อ ก็โดยอ้างอิงแค่ "คอลัมน์เดียว" ให้ใช้ USING เพื่อรบุชื่อคอลัมน์นั้นลงไป
- ห้ามใส่ชื่อตารางหรือ Alias (ชื่อย่อ): ในคอลัมน์ที่ระบุหลัง USING ห้าม เขียนชื่อตารางกำกับหน้าชื่อคอลัมน์เด็ดขาด (เช่น ใช้ USING (department\_id) ได้เลย ห้ามเขียน USING (employees.department\_id))
- ใช้ร่วมกับ NATURAL JOIN ไม่ได้: คำสั่ง NATURAL JOIN และ USING เป็นคำสั่งประเภท "เลือกอย่างใดอย่างหนึ่ง" (Mutually Exclusive) ห้ามใช้พร้อมกันใน SQL Statement เดียวกัน

**SELECT COLUMN-LIST  
FROM table1 JOIN table2 USING (common-column);**



# การ inner join ด้วยคำสั่ง USING

147

Joining Column Names

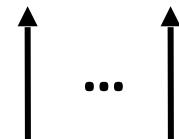
EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

Foreign key



Primary key



# ตัวอย่างการใช้ ด้วยคำสั่ง USING

148

## ตัวอย่าง

แสดง รหัสวิทยาเขต, ชื่อวิทยาเขต และ ชื่ออพาร์ต เงื่อนไข แสดงตามรหัสวิทยาเขต  
ที่เหมือนกันทั้งสองตาราง

```
SELECT Campus_Name, Build_Name, Location  
FROM Campusworld JOIN Building USING(Campus_ID);
```

Campusworld

Campus_ID	Campus_Name	Location
101	Main Campus	Bangkok
102	IT Center	Chonburi

Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

ผลลัพธ์

Campus_ID	Campus_Name	Build_Name
101	Main Campus	Engineering Hall
101	Main Campus	Science Lab
102	IT Center	Information Bldg



# ตัวอย่างการใช้ ด้วยคำสั่ง USING

149

```
SELECT employees.employee_id, employees.last_name,  
       departments.location_id, departments.department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
124	Mourgos	1500	50
141	Rajs	1500	50
142	Davies	1500	50
144	Vargas	1500	50
143	Matos	1500	50

19 rows selected.



# การจัดการกับคอลัมน์ที่ซ้ำซ้อน

150

- **ใช้ชื่อตารางนำหน้า (Table Prefixes):** เพื่อเจาะจงว่าคอลัมน์นั้นมาจากตารางไหน ในกรณีที่ชื่อคอลัมน์ปรากฏอยู่ในหลายตาราง
- **ช่วยเพิ่มประสิทธิภาพ (Improve Performance):** การระบุชื่อตารางนำหน้าช่วยให้ฐานข้อมูลไม่ต้องเสียเวลาเดาหรือค้นหาว่าคอลัมน์นั้นมาจากตารางใด ทำให้ Query ทำงานได้เร็วขึ้น
- **ใช้การตั้งชื่อเล่นคอลัมน์ (Column Aliases):** เพื่อช่วยแยกแยะคอลัมน์ที่มีชื่อเหมือนกันแต่มาจากคลัสเตอร์ต่างกันในผลลัพธ์ (เช่น emp.name AS Employee\_Name, dept.name AS Department\_Name)
- **ข้อห้ามสำคัญสำหรับ USING:** ห้ามใช้ Alias (ชื่อย่อ) หรือชื่อตารางนำหน้าคอลัมน์ที่ถูกระบุไว้ใน USING clause รวมถึงคอลัมน์นั้นที่ไปปรากฏอยู่ในส่วนอื่นๆ ของคำสั่ง SQL นั้นด้วย



# การใช้ Aliases

151

- Simplify Queries ช่วยให้คำสั่งดูง่ายขึ้น ไม่ต้องพิมพ์ชื่อตารางยาว ๆ ซ้ำไปซ้ำมา
- Improve Performance ช่วยให้ฐานข้อมูลเข้าใจได้ทันทีว่าคอลัมน์นั้นมาจากตารางไหน ลดภาระในการประมวลผล

```
SELECT alias1.column_name, alias2.column_name
FROM table_name_1 alias1
JOIN table_name_2 alias 2
USING (common_column);
```

หมายเหตุ การตั้งชื่อเล่น (Alias) ทำได้โดยการเว้นวรคหลังชื่อตาราง และตามด้วยตัวอักษรที่ต้องการ (เช่น employee e)



# ตัวอย่างการใช้ Aliases

152

## ตัวอย่าง

แสดง ชื่อวิทยาเขต และชื่อตึก ทั้งหมด

### Campusworld

Campus_ID	Campus_Name	Location
101	Main Campus	Bangkok
102	IT Center	Chonburi

### Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

```
SELECT c.Campus_Name, b.Build_name  
FROM Campusworld c  
JOIN Building b  
USING (Campus_ID);
```

ผลลัพธ์

Campus_Name	Build_Name
Main Campus	Engineering Hall
Main Campus	Science Lab
IT Center	Information Bldg



# ตัวอย่างการใช้ Aliases

153

- Use table aliases to simplify queries.
- Use table aliases to improve performance.

```
SELECT e.employee_id, e.last_name,  
       d.location_id, department_id  
FROM   employees e JOIN departments d  
USING (department_id) ;
```

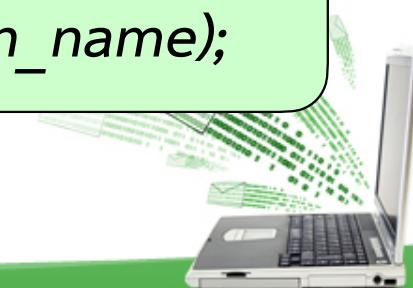


# การ inner join ด้วยคำสั่ง ON

154

- **Arbitrary conditions:** ใช้ระบุเงื่อนไขการเชื่อมโยงได้อย่างอิสระ หรือเลือกคอลัมน์ที่จะ Join ได้ตามต้องการ
- **Separation of conditions:** แยกเงื่อนไขการ Join ออกจากเงื่อนไขการค้นหา (Search conditions) อื่นๆ อย่างชัดเจน
- **Code readability:** ช่วยให้โค้ดอ่านง่ายและเข้าใจได้ทันทีว่าตารางไหนเชื่อมกันด้วยคอลัมน์ใด
- **Self-Joins:** สามารถใช้เชื่อมโยงข้อมูลภายในตารางเดียวกันเองได้ (เช่น การหาชื่อหัวหน้าจากตารางพนักงานเดิม)

```
SELECT alias1.column_name, alias2.column_name  
FROM table_name_1 alias1  
JOIN table_name_2 alias 2  
ON (alias1.column_name = alias2.column_name);
```



# ตัวอย่างการใช้ ด้วยคำสั่ง ON

155

## ตัวอย่าง

แสดง ชื่อตึก และชื่อห้อง เงื่อนไข ชื่อห้องที่อยู่ในตึกนั้น ๆ

### Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

### Room

Room_ID	Room_Name	Building_Code
R101	Network Lab	B03
R202	Library	B01

```
SELECT b.Build_Name, r.Room_Name  
FROM Building b  
JOIN Room r  
ON (b.Build_ID = r.Building_Code);
```

ผลลัพธ์



Build_Name	Room_Name
Information Bldg	Network Lab
Engineering Hall	Library



# ตัวอย่างการใช้ ด้วยคำสั่ง ON

156

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

19 rows selected.



# Self-Joins Using the ON Clause

157

**EMPLOYEES (WORKER)**

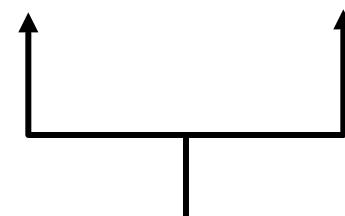
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

...



**MANAGER\_ID in the WORKER table is equal to  
EMPLOYEE\_ID in the MANAGER table.**



# Self-Joins Using the ON Clause

158

```
SELECT e.last_name emp, m.last_name mgr  
FROM   employees e JOIN employees m  
ON     (e.manager_id = m.employee_id);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

19 rows selected.



# การใช้ join เพื่อเชื่อมโยงข้อมูลที่มีความซับซ้อน

159

แบ่งออกเป็น 3 หัวข้อ

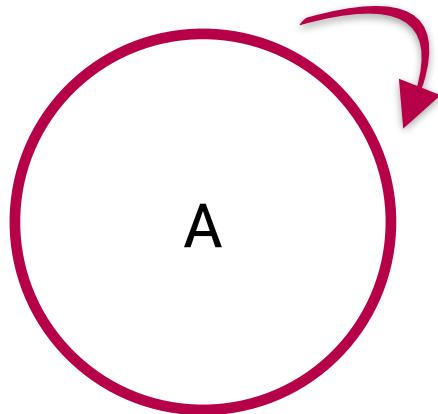
- Self-Joins
- Three-Way Joins
- Non Equijoins



# การใช้ Self-Joins

160

การนำตารางหนึ่งตารางมาเชื่อมตัวมันเอง มักใช้ในกรณีที่ข้อมูลในแถวหนึ่งมีความสัมพันธ์กับอีกแถวหนึ่งในตารางเดียวกัน



```
SELECT alias1.column, alias2.column  
FROM table_name alias1  
JOIN table_name alias2  
ON (alias1.column_id = alias2.reference_id);
```



# ตัวอย่างการใช้ Self-Join

161

## ตัวอย่าง

แสดง ชื่อตึก และชื่อตึกหลัก ที่ตึกนั้นสังกัดอยู่

### Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

```
SELECT b.Build_Name AS "Sub Building",
       m.Build_Name AS "Main Building"
  FROM Building b
 JOIN Building m
 WHERE b.Main_Build_ID = m.Build_ID;
```

### Room

Room_ID	Room_Name	Building_Code
R101	Network Lab	B03
R202	Library	B01

ผลลัพธ์

Sub Building	Main Building
Network Lab	Information Bldg
Library	Engineering Hall



# ตัวอย่างการใช้ Self-Join

162

Applying Additional Conditions to a Join

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

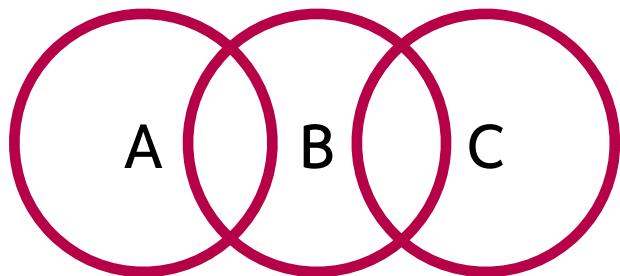
EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500



# การใช้ Three-Way Join

163

การดึงข้อมูลจาก 3 ตารางขึ้นมาแสดงผลพร้อมกันในคำสั่งเดียว โดยใช้คำสั่ง JOIN...ON ต่อ กัน เป็นทอด ๆ



```
SELECT a.column, b.column, c.coloum  
FROM table_A a  
JOIN table_B b ON (a.key = b.key)  
JOIN table_C c ON (b.key = c.key);
```



# ตัวอย่างการใช้ Three-Way Join

164

## ตัวอย่าง

แสดง ชื่อวิทยาเขต ชื่อตึก และชื่อห้องที่อยู่ในวิทยาเขตนั้น ๆ ทั้งหมด

### Building

Build_ID	Build_Name	Campus_ID
B01	Engineering Hall	101
B02	Science Lab	101
B03	Information Bldg	102

### Room

Room_ID	Room_Name	Building_Code
R101	Network Lab	B03
R202	Library	B01

```
SELECT c.Campus_Name, b.Build_Name,  
       r.Room_Name  
FROM   Campusworld c  
JOIN   Building b ON (c.Campus_ID = b.Campus_ID)  
JOIN   Room r ON (b.Build_ID = r.Builidng_Code);
```

### Campusworld

Campus_ID	Campus_Name	Location
101	Main Campus	Bangkok
102	IT Center	Chonburi



# ตัวอย่างการใช้ Three-Way Join

165

## ตัวอย่าง

แสดง ชื่อวิทยาเขต ชื่อตึก และชื่อห้องที่อยู่ในวิทยาเขตนั้น ๆ ทั้งหมด

```
SELECT c.Campus_Name, b.Build_Name,  
       r.Room_Name  
FROM   Campusworld c  
JOIN   Building b ON (c.Campus_ID = b.Campus_ID)  
JOIN   Room r ON (b.Build_ID = r.Builiding_Code);
```



# ตัวอย่างการใช้ Three-Way Join

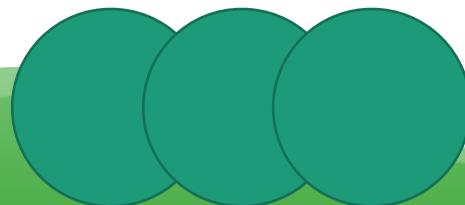
166

```
SELECT employee_id, city, department_name  
FROM employees e  
JOIN departments d  
ON d.department_id = e.department_id  
JOIN locations l  
ON d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

19 rows selected.

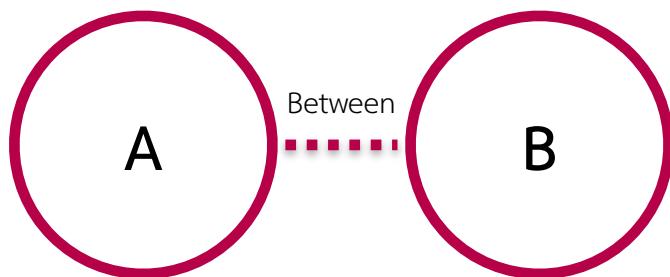
Creating Three-Way Joins with the  
ON Clause



# การใช้ Non-Equiijoins

167

การเชื่อมโยงด้วยเงื่อนไขที่ไม่ใช่เครื่องหมายเท่ากับ ในปกติมัก JOIN ด้วยเครื่องหมาย = ( เช่น ID ตรงกัน ) แต่ Non-Equiijoins จะใช้เงื่อนไขอื่น เช่น การเปรียบเทียบช่วงข้อมูล (Between), มากกว่า หรือน้อยกว่า



```
SELECT a.column, b.column  
FROM table_A a  
JOIN table_B b  
ON (a.column BETWEEN b.low_limit AND  
b.hight_limit) ;
```



# ตัวอย่างการใช้ Non-Equijoins

168

การนำ เงินเดือนพนักงาน ไปตรวจสอบว่าอยู่ในช่วง เกรดเงินเดือน ไหนในตารางหลัก

**EMPLOYEES**

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

20 rows selected.

**JOB\_GRADES**

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

←  
**Salary in the EMPLOYEES  
table must be between  
lowest salary and highest  
salary in the JOB\_GRADES  
table.**

