SQL: Data Manipulation Language (DML)

By Kanokwan Atchariyachanvanich

Faculty of Information Technology

KMITL

Database System Concepts

OUTLINE ก่อนสอบกลางภาค

Date	SQL
9, 11 JAN 2023	• Lab Introduction • Introduction to DBLearn (SQL tool)
16, 18 JAN 2023	LAB 1 - Creating and Managing Tables (DDL)
23, 25 JAN 2023	LAB 2 - Including Constraints (DDL)
30 JAN, 1 FEB 2023	LAB 3 - Manipulating Data (DML)
6, 8 FEB 2023	LAB 4 - SQL SELECT Statements • Writing Basic
13, 15 FEB 2023	LAB 5 - SQL SELECT Statements • Restricting and Sorting Data
20, 22 FEB 2023	- ทวนก่อนสอบ Quiz 1
27 FEB, 1 MAR 2023	**Quiz 1: LAB 1 – LAB 5**

SQL Statement

Type	SQL Statement
Data Manipulation Language (DML) ภาษาสำหรับจัดการข้อมูล คือส่วนของประโยค SQL ที่อนุญาตให้คุณควบคุมหรือจัดการข้อมูล	**SELECT** **INSERT** **UPDATE** **DELETE** **MERGE** **CALL** **EXPLAIN PLAN** **LOCK TABLE**
Data Definition Language (DDL) อธิบายส่วนของ SQL ที่อนุญาตให้สร้าง, เปลี่ยน, และทำลายอ็อบเจ็กต์ฐานข้อมูล อ็อบเจ็กต์ฐานข้อมูลเหล่านี้รวมถึงแบบแผน, ตาราง, มุมมอง ,ลำดับ, แคตาล็อก, ดัชนี, และ alias	**CREATE** **ALTER** **DROP** **RENAME** **ANALYZE** **AUDIT** **COMMENT** **ASSOCIATE STATISTICS** **DISASSOCIATE STATISTICS** **FLASHBACK** **GRANT** **NOAUDIT** **PURGE** **REVOKE** **TRUNCATE** **UNDROP**
Transaction Control จัดการ transaction จากการเปลี่ยนแปลงที่เกิดจาก DML	**COMMIT** **ROLLBACK** **SAVEPOINT** **SET TRANSACTION**

Objective

- After completing this lesson, you should be able to do the following:
  - Describe each data manipulation language (DML) statement
  - Insert rows into a table
  - Update rows in a table
  - Delete rows from a table

DATA MANIPULATION LANGUAGE

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table
- A *transaction* consists of a collection of DML statements that form a logical unit of work.

ADDING A NEW ROW TO A TABLE

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400

New row

280	IT Planning	101	1700

Insert new row into the DEPARTMENTS table

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700
210	IT Support		1700
220	NOC		1700
230	IT Helpdesk		1700
240	Government Sales		1700
250	Retail Sales		1700
260	Recruiting		1700
270	Payroll		1700
280	IT Planning	101	1700

INSERT STATEMENT SYNTAX

- Add new rows to a table by using the INSERT statement:

```
INSERT   INTO   table_name   (column1,  column2, …. ) ]
VALUES          (value1, value2, …..) ;
```

  - *table* is the name of the table
  - *column* is the name of the column in the table to populate
  - *value* is the corresponding value for the column

1. INSERTING A NEW ROW (ไม่ระบุคอลัมน์)

- ไม่แสดงคอลัมน์ในบรรทัด INSERT clause
- ใส่ค่าทุกตัวให้ตรงตามลำดับคอลัมน์ที่กำหนดในตาราง
- คอลัมน์ที่อนุญาตให้ใส่ Null value ได้ ไม่จำเป็นต้องระบุค่าในแถว
- ล้อมตัวอักษรและวันที่ด้วย single quotation marks.

```
INSERT   INTO  regions
VALUES         (5, 'South Africa') ;
```

Field	Type	Null	Key
region_id	int(11) unsigned	NO	PRI
region_name	varchar(25)	YES	

1. INSERTING MULTIPLE ROWS (ไม่ระบุคอลัมน์)

- ไม่แสดงคอลัมน์ในบรรทัด INSERT clause
- ใส่ค่าทุกตัวให้ตรงตามลำดับคอลัมน์ที่กำหนดในตาราง
- คอลัมน์ที่อนุญาตให้ใส่ Null value ได้ ไม่จำเป็นต้องระบุค่าในแถว
- ล้อมตัวอักษรและวันที่ด้วย single quotation marks.

```
INSERT   INTO  regions
VALUES         (5, 'South Africa'),
               (6, 'Australia and Oceania') ;
```

2. INSERTING A NEW ROW (ระบุคอลัมน์)

- แสดงคอลัมน์ในบรรทัด INSERT clause ที่ต้องการเพิ่มค่าในคอลัมน์นั้น
- ใส่แถวใหม่ที่มีค่าในแต่ละคอลัมน์ตามลำดับคอลัมน์ที่กำหนดในตาราง
- ล้อมตัวอักษรและวันที่ด้วย single quotation marks.

```
INSERT   INTO    locations(location_id, city, country_id)
VALUES           (3300, 'Bangkok', 'TH') ;
INSERT   INTO    locations(location_id, city, country_id)
VALUES           (3400, 'Kyoto', 'JP') ;
```

Field	Type	Null	Key
location_id	int(11) unsigned	NO	PRI
street_address	varchar(40)	YES	
postal_code	varchar(12)	YES	
city	varchar(30)	NO	
state_province	varchar(25)	YES	
country_id	char(2)	NO	MUL

2. INSERTING A NEW ROW (ระบุคอลัมน์)

```
INSERT   INTO    countries(country_id, country_name, region_id)
VALUES           ('TH', 'Thailand', 3) ;
```

ก่อนที่จะเพิ่มข้อมูล Bangkok ในตาราง locations จะต้องเพิ่ม TH ในตาราง countries ก่อน

```
INSERT   INTO    locations(location_id, city, country_id)
VALUES           (3300, 'Bangkok', 'TH') ;
```

Countries (Parent table)

Field	Type	Null	Key
country_id	char(2)	NO	PRI
country_name	varchar(40)	YES	
region_id	int(11) unsigned	NO	MUL

Locations (Child table)

Field	Type	Null	Key
location_id	int(11) unsigned	NO	PRI
street_address	varchar(40)	YES	
postal_code	varchar(12)	YES	
city	varchar(30)	NO	
state_province	varchar(25)	YES	
country_id	char(2)	NO	MUL

Exercise 1: INSERT INTO

- จงเขียน SQL เพิ่มข้อมูลให้กับตาราง locations และตารางอื่นที่เกี่ยวข้องตามข้อมูลนี้

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
3500	Sukhumvit	21000	Muang	Rayong	TH

```
INSERT   INTO  countries
VALUES         ('TH', 'Thailand', 3) ;

INSERT   INTO  locations
VALUES         (3500, 'Sukhumvit', 21000, 'Muang', 'Rayong', 'TH') ;
```

Exercise 2: INSERT INTO

- จงเขียน SQL เพิ่มข้อมูลให้กับตาราง locations และตารางอื่นที่เกี่ยวข้องตามข้อมูลนี้

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
3600			Taipei		TW

```
INSERT   INTO  countries (country_id,country_name,region_id)
VALUES         ('TW', 'Taiwan', 3) ;

INSERT   INTO  locations(location_id, country_id,city)
VALUES         (3600, 'TW', 'Taipei',) ;
```

Exercise 3: INSERT INTO

- จงเขียน SQL เพิ่มข้อมูลให้กับตาราง locations และตารางอื่นที่เกี่ยวข้องตามข้อมูลนี้

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
3700			Shanghai		CN

```
INSERT   INTO  locations(location_id, city, country_id)
VALUES         (3700, 'Shanghai', 'CN') ;
```

3. INSERTING ROWS WITH NULL VALUES

- ไม่กำหนดคอลัมน์ แต่ใส่ค่าในคอลัมน์ต่างๆ และ NULL ในคอลัมน์ที่ต้องการไม่ใส่ค่า

```
INSERT   INTO  locations
VALUES         (3700, NULL, NULL, 'Xian', NULL,'CN' ) ;
```

Field	Type	Null	Key
location_id	int(11) unsigned	NO	PRI
street_address	varchar(40)	YES	
postal_code	varchar(12)	YES	
city	varchar(30)	NO	
state_province	varchar(25)	YES	
country_id	char(2)	NO	MUL

Exercise 4: INSERT INTO

- จงเพิ่มข้อมูลให้กับตาราง locations ตามข้อมูลนี้ แบบไม่ระบุชื่อคอลัมน์

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
3900			Harbin		CN

```
INSERT   INTO  locations
VALUES         (3900, NULL,NULL, 'Harbin', NULL, 'CN' ) ;
```

COMMON ERRORS ต้องระวัง

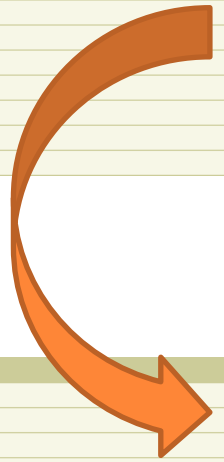
Common errors that can occur during user input:

1. ใน NOT NULL column เกิด error เนื่องจากไม่ใส่ค่าใดๆ ในคำสั่ง INSERT ซึ่งคอลัมน์นั้นห้ามเป็นค่า null
2. ใน column ที่กำหนด uniqueness constraint ใส่ค่าที่ซ้ำกับค่าที่มีอยู่เดิม
3. ละเมิด Foreign key constraint
4. ชนิดของข้อมูลที่ใส่ ไม่ตรงกับชนิดข้อมูลของ column นั้น
5. ค่าที่ใส่มีขนาดใหญ่เกินกว่าที่คอลัมน์นั้นกำหนดไว้

CHANGING DATA IN A TABLE

SALES_REPS

REPS_ID	NAME	SALARY	COMMISSION_PCT
150	Tucker	10000	.3
151	Bernstein	9500	.25
152	Hall	9000	.25
153	Olsen	8000	.2
154	Cambrault	7500	.2
155	Tuvault	7000	.15
156	King	10000	.35
157	Sully	9500	.35



Update rows in the SALES_REPS tables

REPS_ID	NAME	SALARY	COMMISSION_PCT
150	Tucker	10000	.3
151	Bernstein	9500	.3
152	Hall	9000	.3
153	Olsen	8000	.2
154	Cambrault	7500	.2
155	Tuvault	7000	.15
156	King	10000	.35
157	Sully	9500	.35

UPDATE STATEMENT SYNTAX

- แก้ไขแถวในตารางด้วย UPDATE statement:

```
UPDATE    table
SET       column = value  [,  column = value, …]
[WHERE    condition ] ;
```

- Update more than one row at a time (if required).
  - *table* is the name of the table
  - *column* is the name of the column in the table to populate
  - *value* is the corresponding value or subquery for the column
  - *condition* identifies the rows to be updated and is composed of column names, expressions, constants, subqueries, and comparison operators

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not qual to

....WHERE hire_date= '1998-05-04'

....WHERE salary>=6000

....WHERE last_name='Smite'

UPDATING ROWS IN A TABLE

- แบบที่ 1: แก้ไข 1 แถวหรือมากกว่า 1 แถว ด้วย WHERE clause

```
UPDATE    employees
SET       salary = 10000
WHERE     employee_id = 101 ;
```

- แบบที่ 2: แก้ไขทุกแถวในตาราง ไม่จำเป็นต้องใช้ WHERE clause

```
UPDATE    employees
SET       commission_pct = .3 ;
```

Exercise 5: UPDATE

- จงแก้ไขข้อมูลในตาราง locations เป็นตามข้อมูลใหม่นี้

Column	Old	New
location_id	1000	1000
postal_code	00989	10100

```
UPDATE    locations
SET       postal_code = 10100
WHERE     location_id = 1000 ;
```

Exercise 6: UPDATE

- จงแก้ไขข้อมูลในตาราง jobs เป็นตามข้อมูลใหม่นี้

Column	Old	New
min_Salary	4000-5000	6000

```
UPDATE    jobs
SET       min_salary = 6000
WHERE     min_salary >= 4000 and min_salary <=5000 ;
```

REMOVING A ROW FROM A TABLE

SALES_REPS

REPS_ID	NAME	SALARY	COMMISSION_PCT
173	Kumar	6100	.1
174	Abel	11000	.3
175	Hutton	8800	.25
176	Taylor	8600	.2
177	Livingston	8400	.2
179	Johnson	6200	.1
202	Fay	6000	
203	Mavris	6500	
204	Baer	10000	

Delete a Row from a SALES_REPS Table

REPS_ID	NAME	SALARY	COMMISSION_PCT
173	Kumar	6100	.1
174	Abel	11000	.3
175	Hutton	8800	.25
176	Taylor	8600	.2
177	Livingston	8400	.2
179	Johnson	6200	.1
202	Fay	6000	
203	Mavris	6500	

DELETE STATEMENT SYNTAX

- ลบแถวที่อยู่ในตารางด้วย DELETE statement:

```
DELETE   FROM      table
  [WHERE                  condition] ;
```

  - *table* is the table name
  - *condition* identifies the rows to be deleted and is composed of column names, expressions, constants, subqueries, and comparison operators

DELETING ROWS FROM A TABLE

- แบบที่ 1: ลบเฉพาะแถวที่ต้องการ ด้วย WHERE clause
- ตัวอย่าง : ลบแถวที่มี reps_id 204 ออกจากตาราง sales_reps

```
DELETE   FROM     sales_reps
WHERE    reps_id = 204 ;
```

- ตัวอย่าง : ลบแถวที่มี salary ตั้งแต่ 7000 ถึง 7500 ออกจากตาราง

```
DELETE   FROM      sales_reps
WHERE      salary >=7000 and salary <= 7500;
```

DELETING ROWS FROM A TABLE (CONT.)

- แบบที่ 2: ลบทุกแถวในตาราง โดยการห้ามใช้ WHERE clause

```
DELETE   FROM    sales_reps  ;
```

- Note: ถึงแม้ไม่มีค่าในคอลัมน์ใดๆ เลย แต่ยังมี Table structure เหลืออยู่

DELETING ROWS: INTEGRITY CONSTRAINT ERROR

```
DELETE   FROM    regions ;
```

```
Cannot delete or update a parent row: a foreign key constraint fails
(`g39336`.`countries`, CONSTRAINT `countries_regions_region_id`
FOREIGN KEY (`region_id`) REFERENCES `regions` (`region_id`))
```

NOTE: คุณไม่สามารถลบแถวที่มี primary key ซึ่งใช้เป็น foreign key ในตารางอื่นได้

Exercise 7: DELETE

- จงลบข้อมูลในตาราง sales_reps โดยลบเฉพาะแถวที่มีพนักงานเงินเดือนมากกว่า 5000

```
DELETE   FROM     sales_reps
WHERE      salary > 5000;
```

Exercise 8: INSERT

- จงเขียน SQL statementเพิ่มข้อมูลให้กับตาราง lab_location ตามข้อมูลนี้ (แบบระบุคอลัมน์)

LOCATION_ID	LOCATION_NAME
002	Rayong
003	Ranong

```
INSERT INTO lab_location(location_id,location_name)

VALUES (002,'Rayong'),(003,'Ranong');
```

Exercise 9: INSERT

- จงเขียน SQL statement เพื่อเพิ่มข้อมูลให้กับตาราง lab_emp ตามข้อมูลนี้แบบไม่ระบุชื่อคอลัมน์

ID	LAST_NAME	FIRST_NAME	SALARY	LOCATION_ID
004	Woo	Woody	15000	001
005	Sun	Peng		

```
INSERT INTO lab_emp

VALUES

(004,'Woo','Woody',15000,001),(005,'Sun','Peng',NULL,NULL);
```

FOREIGN KEY CONSTRAINT (REFERENTIAL INTEGRITY CONSTRAINT)

- **FOREIGN KEY:** นิยามคอลัมน์ในตารางลูก
  - Insert Constraint: Value cannot be inserted in CHILD Table if the value is not lying in MASTER Table
  - Delete Constraint: Value cannot be deleted from MASTER Table if the value is lying in CHILD Table
- **ON DELETE CASCADE:** ลบ dependent rows ในตารางลูกเมื่อแถวในตารางแม่ถูกลบ
- ถ้ามี ON DELETE CASCADE
  - เมื่อสั่งลบแถวข้อมูลในตารางแม่ ด้วยคำสั่ง “delete from ชื่อตาราง;” จะลบแถวในตารางลูกทั้งหมด
  - เมื่อสั่งลบแถวข้อมูลในตารางแม่ ด้วยคำสั่ง “delete from ชื่อตาราง where ;” จะลบแถวในตารางลูกนั้นที่ตรงตามเงื่อนไขเท่านั้น
- ถ้าไม่มี ON DELETE CASCADE เมื่อสั่งลบแถวข้อมูลด้วยคำสั่ง “delete from ชื่อตาราง;” มันจะฟ้องเตือน FK ดังนั้นต้อง
  - ทำการลบ FK ในตารางแม่ และใช้คำสั่ง “delete from ชื่อตาราง” เพื่อลบแถวในตารางแม่ แต่จะไม่กระทบข้อมูลในตารางลูก แต่ไม่สมเหตุสมผล

ถ้ามี ON DELETE CASCADE

```
CREATE TABLE Student (
    sno INT PRIMARY KEY,
    sname VARCHAR(20),
    age INT
);
CREATE TABLE Course (
    cno INT PRIMARY KEY,
    cname VARCHAR(20)
);
CREATE TABLE Enroll (
    sno INT,
    cno INT,
    jdate date,
    FOREIGN KEY(sno) REFERENCES Student(sno) ON DELETE CASCADE,
    FOREIGN KEY(cno) REFERENCES Course(cno) ON DELETE CASCADE
);
```

```
INSERT INTO Student(sno, sname,age)
 VALUES(1,'Ankit',17),
      (2,'Ramya',18),
      (3,'Ram',16),
      (4,'Mai', 18);
INSERT INTO Course(cno, cname)
 VALUES(101,'c'),
      (102,'c++'),
      (103,'DBMS');
INSERT INTO Enroll(sno,cno,jdate)
 VALUES(1, 101, '2021-06-05'),
      (1, 102, '2021-06-07'),
      (2, 103, '2021-06-05');
```

ถ้ามี ON DELETE CASCADE (ต่อ)

เมื่อใช้คำสั่ง Delete from student

where sno=1;

จะได้ผลลัพธ์คือ แถวข้อมูล sno=1 ในตาราง student ถูกลบ

sno	sname	age
2	Ramya	18
3	Ram	16
4	Mai	18

และ แถวข้อมูลในตาราง enroll ที่มีค่าใน sno=1 ถูกลบไปด้วย

sno	cno	jdate
2	103	2021-06-05

ถ้าไม่มี ON DELETE CASCADE

เมื่อใช้คำสั่ง Delete from student

where sno=1;

จะได้ผลลัพธ์คือ

```
Cannot delete or update a parent row: a foreign key constraint fails (`g36632`.`enroll`, CONSTRAINT `enroll_ibfk_1`
FOREIGN KEY (`sno`) REFERENCES `student` (`sno`))
```

ดังนั้นต้อง ทำการลบ FK ในตารางแม่ และใช้คำสั่ง “delete from ชื่อตาราง” เพื่อลบแถวในตารางแม่ แต่จะไม่กระทบข้อมูลในตารางลูก แต่ไม่สมเหตุสมผล

student

sno	sname	age
2	Ramya	18
3	Ram	16
4	Mai	18

enroll

sno	cno	jdate
1	101	2021-06-05
1	102	2021-06-07
2	103	2021-06-05

FOREIGN KEY CONSTRAINT (REFERENTIAL INTEGRITY CONSTRAINT)

- **FOREIGN KEY:** นิยามคอลัมน์ในตารางลูก
- **ON DELETE SET NULL:** แปลงค่าที่อยู่ใน foreign key เป็นค่า null เมื่อแถวในตารางแม่ถูกลบ
- ถ้ามี ON DELETE SET NULL
  - เมื่อสั่งลบแถวข้อมูลในตารางแม่ ด้วยคำสั่ง “delete from ชื่อตาราง;” จะเปลี่ยนค่าของ foreign key ในตารางลูกทั้งหมดเป็น ค่า NULL
  - เมื่อสั่งลบแถวข้อมูลในตารางแม่ ด้วยคำสั่ง “delete from ชื่อตาราง where ;”จะเปลี่ยนค่าของ foreign key ในตารางลูกที่ตรงตามเงื่อนไขเท่านั้นให้เป็น ค่า NULL

ถ้ามี ON DELETE SET NULL

```
CREATE TABLE Student (
    sno INT PRIMARY KEY,
    sname VARCHAR(20),
    age INT
);
CREATE TABLE Course (
    cno INT PRIMARY KEY,
    cname VARCHAR(20)
);
CREATE TABLE Enroll (
    sno INT,
    cno INT,
    jdate date,
    FOREIGN KEY(sno) REFERENCES Student(sno) ON DELETE SET NULL,
    FOREIGN KEY(cno) REFERENCES Course(cno) ON DELETE SET NULL
);
```

```
INSERT INTO Student(sno, sname,age)
 VALUES(1,'Ankit',17),
      (2,'Ramya',18),
      (3,'Ram',16),
      (4,'Mai', 18);
INSERT INTO Course(cno, cname)
 VALUES(101,'c'),
      (102,'c++'),
      (103,'DBMS');
INSERT INTO Enroll(sno,cno,jdate)
 VALUES(1, 101, '2021-06-05'),
      (1, 102, '2021-06-07'),
      (2, 103, '2021-06-05');
```

ถ้ามี ON DELETE SET NULL (ต่อ)

เมื่อใช้คำสั่ง Delete from student

where sno=1;

จะได้ผลลัพธ์คือ แถวข้อมูล sno=1 ในตาราง student ถูกลบ

sno	sname	age
2	Ramya	18
3	Ram	16
4	Mai	18

และ แถวข้อมูลในตาราง enroll ที่มีค่าใน sno=1 เปลี่ยนเป็น NULL

sno	cno	jdate
NULL	101	2021-06-05
NULL	102	2021-06-07
2	103	2021-06-05

Thank you
For your attention