บทที่ 7
ภาษาในการจัดการฐานข้อมูล

ภาษาฐานข้อมูลสามารถช่วยลดแรงงานผู้ใช้ ไม่ว่าจะเป็นงานด้านการเรียนรู้การใช้งานชุดคำสั่ง รวมถึงงานคิวรีข้อมูลขั้นสูงเพื่อเรียกดูข้อมูลที่มีความซับซ้อนได้ และการแปลงข้อมูลดิบให้เป็นสารสนเทศ โดยทั่วไปภาษาที่ใช้งานในฐานข้อมูลจะต้องสามารถสนับสนุนงานด้านต่างๆ ได้แก่ การสร้างฐานข้อมูลและโครงสร้างของรีเลชั่น การสนับสนุนงานด้านการจัดการข้อมูลพื้นฐาน ซึ่งประกอบด้วย การเพิ่ม การปรับปรุง และการลบข้อมูลออกจากรีเลชั่น และการสนับสนุนงานคิวรีข้อมูล (Query)

ภาษาที่ใช้ในการสืบค้นข้อมูลแบบมีโครงสร้าง

ภาษาที่ใช้ในการสืบค้นข้อมูลแบบมีโครงสร้าง (Structural Query Language : SQL) จัดเป็นภาษามาตรฐานบนระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งเป็นภาษาที่สามารถนำไปใช้งานได้ในคอมพิวเตอร์หลายระดับด้วยกัน ไม่ว่าจะเป็นเมนเฟรมคอมพิวเตอร์จนถึงไมโครคอมพิวเตอร์

ภาษา SQL ถูกพัฒนาขึ้นจากแนวคิดทางคณิตศาสตร์ คือ Relational Algebra และ Relation Calculus ซึ่งเป็นไปตามแนวคิดของเทคโนโลยีฐานข้อมูลเชิงสัมพันธ์ที่ E.F. Codd เป็นผู้คิดค้นขึ้นเมื่อปี ค.ศ. 1970 และต่อมาทางบริษัท IBM ได้เริ่มพัฒนางานวิจัยปี ค.ศ. 1974 โดยใช้ชื่อว่า “Structured English Query Language” หรือ SEQUEL (อ่านว่า ซี-ควอล) จากนั้นจึงได้ปรับปรุงเวอร์ชันเป็น SEQUEL/2 เมื่อปี ค.ศ. 1976 และต่อมาก็ได้เปลี่ยนชื่อมาเป็น SQL (S-Q-L) อันเนื่องมาจากคำย่อเดิมนั้นไปซ้ำกับผลิตภัณฑ์ทางการค้าของเจ้าอื่นที่ใช้มาก่อน ดังนั้นในปัจจุบันอาจจะได้ยินชื่อจากคนบางกลุ่มที่มักเรียกชุดคำสั่ง “SEQUEL” ซึ่งหมายถึง “SQL” นั่นเอง

หลังจากปี ค.ศ.1970 เป็นต้นมา ระบบฐานข้อมูล ORACLE ที่ถูกพัฒนาโดยบริษัท ORACLE Corporation และถือเป็นก้าวแรกของการพัฒนาระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ในเชิงพาณิชย์ที่ตั้งอยู่บนพื้นฐานของ SQL

เมื่อมีผลิตภัณฑ์ DBMS จากผู้ผลิตต่างๆ มากขึ้นจึงทำให้เกิด SQL หลายรูปแบบ ดังนั้นในราวปี ค.ศ. 1982 ทาง American National Standards Institute (ANSI) จึงได้คิดค้นและดำเนินการร่างมาตรฐานชุดคำสั่ง SQL ขึ้นมาเพื่อให้ผู้ผลิตรายต่างๆ สร้างชุดคำสั่งดังกล่าวให้อยู่ภายใต้มาตรฐานเดียวกัน อย่างไรก็ตามในปัจจุบัน แต่ละค่ายต่างก็มีการเพิ่มคุณภาพสมบัติพิเศษเพิ่มเติมเพื่อให้ผลิตภัณฑ์ของตนเองมีประสิทธิภาพสูงขึ้น เพื่อนำมาใช้เป็นจุดขายในเชิงการตลาด แต่ทั้งนี้โดยหลักการแล้วชุดคำสั่งกล่าวก็ยังคงตั้งอยู่บนพื้นฐานที่ทาง ANSI บัญญัติไว้ โดยปัจจุบันมีผลิตภัณฑ์ระบบจัดการฐานข้อมูลต่างๆ เช่น ORACLE, DB2, Informix, MS-SQL, MS-Access รวมทั้ง MS-Visual FoxPro เป็นต้น (Eric & Jim, 2012, p. 193)

การใช้งานคำสั่งภาษาที่ใช้ในการสืบค้นข้อมูลแบบมีโครงสร้าง

การใช้งานคำสั่ง SQL สามารถนำไปใช้งานได้ 2 ลักษณะคือ แบบโต้ตอบ (Interactive SQL) และแบบฝังในตัวโปรแกรม (Embedded SQL) ซึ่งจะอธิบายดังต่อไปนี้

1. แบบโต้ตอบ

แบบโต้ตอบ (Interactive SQL) เป็นการปฏิบัติการที่ผู้ใช้สามารถเขียนคำสั่งภาษา SQL โต้ตอบกันบนจอภาพ เพื่อเรียกดูข้อมูลในขณะทำงานได้ทันที ตัวอย่างเช่น ต้องการแสดงรายชื่อพนักงานที่สังกัดอยู่สาขา B003 ก็สามารถใช้คำสั่งได้ดังนี้

```
SELECT *
FROM Employee
WHERE branchNo =  'B003' ;
```

อย่างไรก็ตาม การปฏิบัติการโดยผู้ใช้ที่สามารถโต้ตอบกับข้อมูลในฐานข้อมูล ผู้ใช้เหล่านี้จำเป็นต้องมีความรู้ในระดับเบื้องต้นที่สามารถเขียนคำสั่งเพื่อจัดการกับข้อมูลที่ต้องการได้ สิ่งเหล่านี้ทำให้เกิดความคล่องตัวในกรณีที่ผู้ใช้อาจจำเป็นต้องเรียกดูข้อมูลแบบเฉพาะกิจที่ต้องการ ทั้งนี้ผู้บริหารฐานข้อมูลจำเป็นต้องจำกัดสิทธิการใช้งานในการเข้าถึงเพื่อความปลอดภัยในฐานข้อมูล เช่น อาจให้สิทธิ์กับผู้ใช้บางคนที่สามารถโต้ตอบกับระบบคำสั่ง SQL ได้ รวมถึงมีสิทธิ์ในการเข้าถึงฐานข้อมูลบางส่วนเท่านั้นและใช้งานได้เพียงการเรียกดูข้อมูลเท่านั้น ไม่สามารถปรับปรุงหรือแก้ไขข้อมูลใดๆ ได้ เป็นต้น

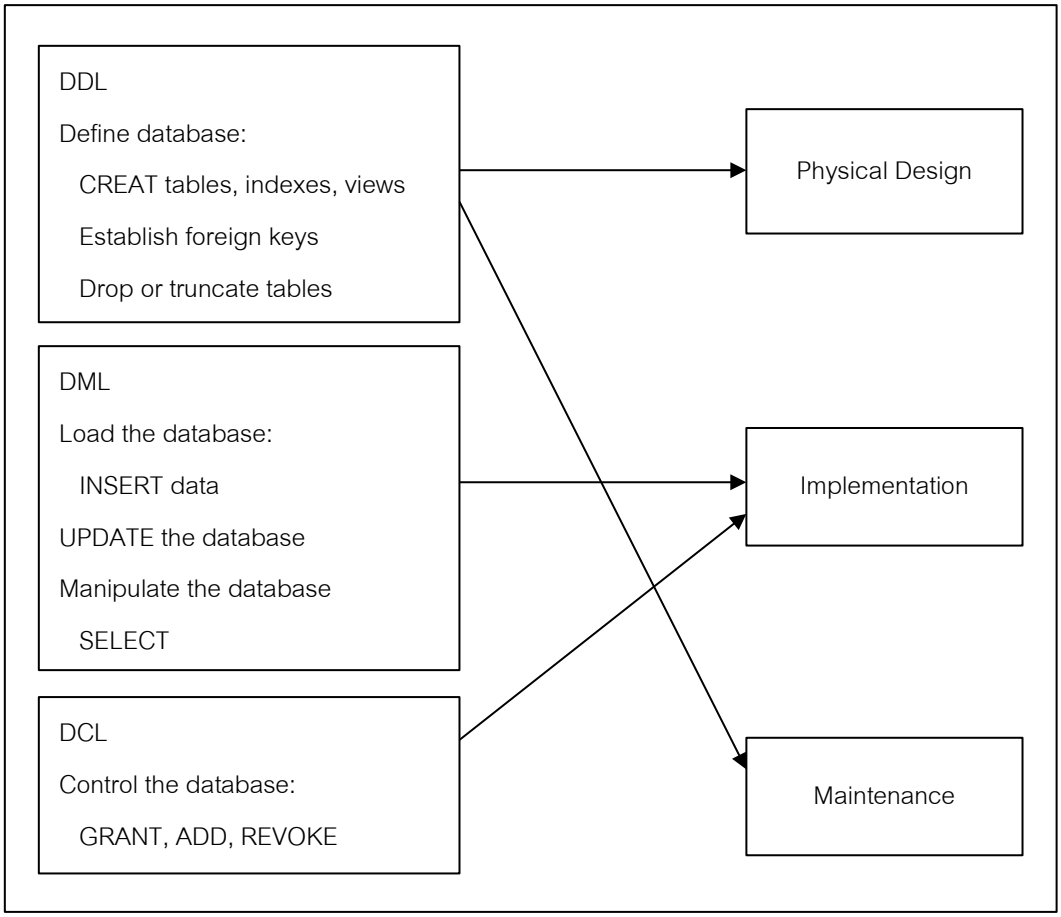
2. แบบฝังในตัวโปรแกรม

แบบฝังในตัวโปรแกรม (Embedded SQL) เป็นการนำคำสั่งภาษา SQL ไปใช้งานร่วมกับโปรแกรมภาษาระดับสูง ซึ่งปัจจุบันมีภาษาโปรแกรมหลายภาษาด้วยกันที่สนับสนุนคำสั่ง SQL ในการปฏิบัติการกับฐานข้อมูล เช่น ภาษาโคบอล ภาษาซี ภาษาปาสคาล ในการ

ปฏิบัติการลักษณะนี้จำเป็นต้องพึ่งพาโปรแกรมเมอร์ในการเขียนคำสั่ง SQL ด้วยการฝังไว้ในตัวโปรแกรม ทั้งนี้เพื่อลดข้อจำกัดบางอย่างที่ SQL ทำไม่ได้ เช่น คำสั่งเพื่อการควบคุมซึ่งได้แก่ คำสั่งลูป (Loop), DO…WHILE รวมถึงการสร้างเงื่อนไขที่มีความซับซ้อน เป็นต้น การนำคำสั่ง SQL ฝังในตัวโปรแกรมเป็นการเพิ่มประสิทธิภาพการทำงานของโปรแกรมให้สูงขึ้น

ประเภทคำสั่งของภาษาที่ใช้ในการสืบค้นข้อมูลแบบมีโครงสร้าง

คำสั่ง SQL แบ่งออกเป็น 3 ประเภท ดังภาพที่ 7.1 ซึ่งสามารถอธิบายได้ดังนี้



ภาพที่ 7.1 DDL, DML, DCL และกระบวนการพัฒนาฐานข้อมูล

ที่มา : Mohankumar & Robert (2015, p.150)

1. **ภาษานิยามข้อมูล** (Data Definition Language : DDL) ภาษา DDL ประกอบด้วยกลุ่มคำสั่งที่ใช้สำหรับสำหรับสร้างตาราง แก้ไขตาราง และลบตาราง กล่าวคือกลุ่มคำสั่งที่ใช้ในการสร้างฐานข้อมูลด้วยการกำหนดโครงสร้างข้อมูลว่ามีคอลัมน์หรือหรือแอตทริบิวต์ใดบ้าง มีชนิดข้อมูล (data type) เป็นชนิดใด รวมทั้งการจัดการด้านการเพิ่ม แก้ไข และลบแอตทริบิวต์ต่างๆ ในรีเลชั่น รวมถึงการสร้างลำดับดัชนี ให้กับรีเลชั่น อย่างไรก็ตาม ชุดคำสั่ง DDL ดังกล่าวผู้บริหารฐานข้อมูลมักจะกำหนดสิทธิ์การใช้งานชุดคำสั่งเหล่านี้ให้กับผู้ดูแล หรือผู้ที่ได้รับมอบหมายให้สามารถจัดการ ได้เท่านั้น เนื่องจากเป็นชุดคำสั่งสำคัญที่ส่งผลถึงการเปลี่ยนแปลงโครงสร้างในฐานข้อมูล

2. **ภาษาจัดการข้อมูล** (Data Manipulation Language : DML) ภาษา DML จัดเป็นกลุ่มคำสั่งที่ถือเป็นแกนสำคัญของภาษา SQL โดยกลุ่มคำสั่งเหล่านี้จะใช้เพื่อการอัปเดต เพิ่ม ปรับปรุง และเรียกดูข้อมูลในฐานข้อมูล ซึ่งคำสั่งดังกล่าวอาจเขียนในรูปแบบของ Interactive SQL หรือ Embedded SQL

3. **ภาษาควบคุมข้อมูล** (Data Control Language : DCL) ภาษา DCL เป็นกลุ่มคำสั่งที่ช่วยให้ผู้บริหารฐานข้อมูลสามารถควบคุมฐานข้อมูล ซึ่งประกอบด้วยคำสั่งที่ใช้สำหรับการอนุญาต หรือ ยกเลิกสิทธิ์ในการเข้าถึงฐานข้อมูล ซึ่งเป็นกระบวนการป้องกันความปลอดภัยต่อฐานข้อมูล (Mohankumar & Robert, 2015, p.148)

อย่างไรก็ตาม DBMS ของแต่ละผลิตภัณฑ์จะมีการกำหนดชนิดข้อมูล (data types) ที่แตกต่างกันออกไป เช่น Numeric, String, Date/Time เป็นต้น ทั้งนี้ในบางครั้งอาจมีชนิดข้อมูลแบบ Graphics และ Image โดยแต่ละผลิตภัณฑ์อาจใช้ชื่อแทนความหมายแตกต่างกัน แต่เป็นชนิดข้อมูลเดียวกัน รวมทั้งบาง DBMS อาจไม่มีชนิดข้อมูลบางชนิดที่มีอยู่ใน DBMS อื่นๆ ก็เป็นได้ ตัวอย่างชนิดข้อมูลของ MS-Access แสดงในตารางที่ 7.1 และตัวอย่างชนิดข้อมูลบางส่วนของ ORACLE แสดงในตารางที่ 7.2 (Guy, 2015, p. 157)

ตารางที่ 7.1 ตัวอย่าง Data Types ของ MS-Access

ที่มา : Guy (2015, p. 157)

Data Type	Use	Size
Text	Text or text/numbers. Also numbers that do not require calculations, such as telephone numbers.	Up to 255 characters
Memo	Lengthy text and numbers, such as notes or descriptions	Up to 65,536 characters
Number	Numeric data to be used for mathematical calculations, involving money (use Currency type). Corresponds to the SQL exact numeric and approximate numeric data type.	1,2,4 or 8 bytes (16 bytes for Replication ID)
Date/Time	Dates and times	8 bytes
Currency	Currency Values. Use the currency data type to prevent rounding off during calculations	8 bytes
Autonumber	Unique sequential (incrementing by 1) or random numbers automatically inserted when a record is added.	4 bytes (16 bytes for Replication ID)
Yes/No	Fields that will contain only one of two values such as Yes/No, True*False On/Off. Corresponds to the SQL Bit data type	1 bit
OLE Object	Objects (such as Microsoft Word documents, Microsoft Excel spreadsheets, picture, sounds, or other binary data),created in other programs using the OLE protocol, which can be linked to, or embedded in a Microsoft Office Access table	Up to 1 gigabyte
Hyperlink	Field that will store hyperlinks.	Up to 64,000 characters
Lookup Wizard	Creates a field that allows the user to choose a value from another table or from a list of values using a combo box choosing this option in the data type list starts a wizard to define this	Same size are the primary key that forms the lookup field (typically 4 bytes)

ตารางที่ 7.2 ตัวอย่าง Data Types บางส่วนของ ORACLE

ที่มา : Guy (2015, p. 158)

Data Type	Use	Size
char (size)	Stores fixed-length data (default=1)	Up to 2000 bytes
nchar (size)	Unicode data types that store Unicode character data. Same as char data type, except the maximum length is determined by the character set of the database	Up to 4000 bytes
varchar2 (size)	Stores variable length character data.	
nvarchar2 (size)	Same as varchar2 with the same caveat as for nchar data type.	Up to 2000 bytes
varchar	Currently the same as char. However, use of varchar2 is recommended as varchar might become a separate data type with different comparison semantics in a later release	
number (1,d)	Stores fixed-point or floating-point numbers, where 1 stands for length and d stands for the number of decimal digits. For example, number(5,2) could contain nothing larger than 999.99 without and error.	+1.0E-103… +9.99E125 (up to 38 significant digits)
decimal (1,d) or numeric (1,d)	Same as number. Provided for compatibility with SQL standard	
integer, int or smallint	Provided for compatibility with SQL standard. Convertied to number (38)	
data	Stores dates	Up to 4 gigabytes
blob	A binary large object.	Up to 4 gigabytes
clob	A character large object.	Up to 2000 bytes
raw (size)	Raw binary data, such as a sequence of graphics characters or a digitized picture.	

ภาษานิยามข้อมูล

ภาษานิยามข้อมูลหรือ Data Definition Language : DDL ใน SQL อนุญาตให้ฐานข้อมูลที่ประกอบด้วย SCHEMAS, DOMAINS, TABLES, VIEWS และ INDEXS สามารถถูกสร้างขึ้นหรือถูกทำลายได้โดยมาตรฐานของ SQL ในกลุ่มคำสั่ง DDL ประกอบด้วยคำสั่งต่อไปนี้

CREATE SCHEMA		DROP SCHEMA
CREATE DOMAIN	ALTER DOMAIN	DROP DOMAIN
CRATE TABLE	ALTER TABLE	DROP TABLE
CREATE VIEW		DROP VIEW

โดยกลุ่มคำสั่งข้างต้นเกี่ยวข้องกับการสร้าง การเปลี่ยน และการทำลายโครงสร้างของสคีมา และนอกจากนี้ก็ยังมีชุดคำสั่งเกี่ยวกับการจัดการดัชนี ซึ่งทั้ง 2 คำสั่งต่อไปนี้ถึงแม้ว่าจะไม่ได้บรรจุอยู่ในมาตรฐาน SQL ก็ตาม แต่ DBMS ส่วนใหญ่มักจะบรรจุชุดคำสั่งทั้งสองนี้สำหรับการจัดการเกี่ยวกับการเรียงลำดับดัชนี (Orin, Neil & Peter, 2012, p. 242)

CREATE INDEX

DROP INDEX

1. การสร้างฐานข้อมูล

ในการสร้างฐานข้อมูลจะมีขั้นตอนที่แตกต่างกันตามแต่ละผลิตภัณฑ์ของ DBMS ตัวอย่างเช่น ในระบบมัลติยูสเซอร์ ผู้ที่มีสิทธิ์ในการสร้างฐานข้อมูลปกติจะถูกสงวนไว้ให้กับผู้บริหารฐานข้อมูลเป็นผู้จัดการเท่านั้น ในขณะที่ระบบผู้ใช้คนเดียวก็จะมีรูปแบบที่แตกต่างกันออกไปโดยผู้ใช้ทั่วไปก็สามารถสร้างขึ้นเองตามความต้องการได้ มาตรฐาน ISO มิได้ระบุไว้ในข้อกำหนดว่าจะต้องสร้างฐานข้อมูลอย่างไร ดังนั้นผลิตภัณฑ์ DBMS แต่ละค่ายจึงมีวิธีการสร้างฐานข้อมูลที่แตกต่างกันออกไป

ตามมาตรฐาน ISO ได้ระบุไว้ว่า รีเลชั่นและฐานข้อมูลอื่นๆ จะคงชีพอยู่ในสภาพแวดล้อม ซึ่งประกอบไปด้วยหนึ่งแคตาล็อก (Catalog) หรือมากกว่า แต่ละแคตาล็อกจะประกอบด้วยกลุ่มของสคีมา (Schema) กล่าวคือ ทุกออบเจ็กต์ในฐานข้อมูลจะถูกอธิบายโดยสคีมา สำหรับออบเจ็กต์ก็สามารถเป็นได้ทั้งตาราง วิวหรือโดเมน และทุกๆ ออบเจ็กต์ในสคีมาจะมีเจ้าของเดียวกัน และสามารถนำไปแชร์ใช้งานร่วมกันได้ตามจำนวนผู้ใช้ที่กำหนดค่าดีฟอลต์ไว้ (Louis & Stacia, 2017, p. 153)

การสร้างสคีมาในรูปแบบอย่างง่ายมีดังนี้

CREATE SCHEMA [Name ‘ AUTHORIZATION CreatorIdentifier]

เช่น หากผู้สร้างสคีมา Test นี้คือ Somchai ก็สามารถเขียนเป็นชุดคำสั่งดังนี้

CREATE SCHEMA Test AUTHORIZATION Somchai

ในขณะเดียวกัน สคีมาที่สร้างขึ้นก็สามารถลบออกไปได้ด้วยคำสั่งดังต่อไปนี้

DROP SCHEMA Test

1.1 การสร้างตาราง

หลังจากมีการสร้างฐานข้อมูลแล้ว ต่อไปจึงสามารถสร้างตาราง ที่บรรจุอยู่ในฐานข้อมูลได้ด้วยคำสั่ง CREATE TABLE ซึ่งมีรูปแบบประโยคคำสั่งดังนี้

```
CREATE TABLE TableName
        { (columnName datatype [NOT NULL] [UNIQUE]
        [DEFAULT defaultOption] [CHECK (searchCondition)] [,…]}
        [PRIMARY KEY (listOfColumns),]
        { [UNIQUE (listOfColumns)] [,…]}
        { [FOREING KEY (listOfForeignKeyColumns)
        REFERENCES ParentTableName [(listOfCandidateKeyColumns)]
            [MATCH {PARTIAL | FULL}
             [ON UPDATE referentialAction]
             [ON DELETE referentialAction]] [,…]}
        { [CHECK (searchCondition)] [,…]})
```

รูปแบบของชุดคำสั่ง CREATE TABLE สามารถที่จะทำการกำหนดความคงสภาพ (integrity) และข้อบังคับ (constraints) ต่างๆ ลงไปได้ อย่างไรก็ตาม ประโยคคำสั่งข้างต้นจะนำไปใช้งานบน DBMS ที่สนับสนุนการทำงานดังกล่าวได้เท่านั้น

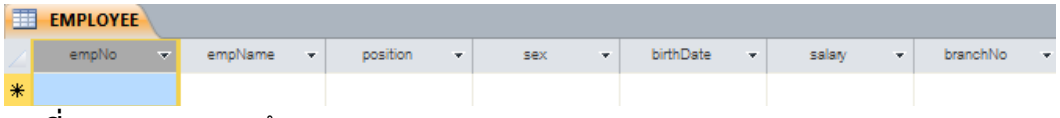
ตัวอย่างการสร้างตาราง EMPLOYEE ด้วย MS-Access ผ่านชุดคำสั่ง DDL ดังต่อไปนี้

```
CREATE   TABLE EMPLOYEE (
         empNo        TEXT (4)      NOT  NULL,
         empName      TEXT (30)     NOT  NULL,
         position     TEXT (20)     NOT  NULL,
         sex          TEXT (1),
         birthdate    DATE,
```

salary CURRENCY NOT NULL,

branchNo TEXT (4) NOT NULL);

ผลจากการสร้างตารางจะได้ตารางเปล่าขึ้นมาที่พร้อมใช้งานสำหรับการบันทึกข้อมูลต่อไป ดังนี้



ภาพที่ 7.2 ผลจากการสร้างตาราง

1.2 การเปลี่ยนแปลงโครงสร้างตาราง

มาตรฐาน ISO ได้กำหนดชุดคำสั่ง ALTER TABLE เพื่อเปลี่ยนโครงสร้างของตารางที่เคยสร้างขึ้น โดยคำสั่ง ALTER TABLE สามารถจัดการกับทางเลือกทั้ง 6 ต่อไปนี้

1.2.1 การเพิ่มคอลัมน์ใหม่เข้าไปในตาราง

1.2.2 การลบคอลัมน์ออกจากตาราง

1.2.3 การเพิ่มข้อบังคับ (constraints) ใหม่ให้กับตาราง

1.2.4 การลบข้อบังคับออกจากตาราง

1.2.5 การกำหนดค่าปกติ (default)

1.2.6 การยกเลิกปกติให้กับคอลัมน์

รูปแบบการใช้งานของประโยคคำสั่ง ALTER TABLE แสดงได้ดังนี้

ALTER TABLE TableName

[ADD [COLUMN] columnName dataType **[NOT NULL] [UNIQUE]**

[DEFAULT defaultOption**] [CHECK** (searchCondition)]]

[DROP [COLUMN] columnName **[RESTRICT | CASCADE]]**

[ADD [CONSTRAINT [ConstraintName]] tableConstraintDefinition]

[DROP CONSTRAINT ConstraintName [RESTRICT | CASCADE]]

[ALTER [COLUMN] SET DEFAULT defaultOption

[ALTER [COLUMN] DROP DEFAULT]

ตัวอย่างการใช้งานชุดคำสั่ง ALTER TABLE สำหรับการยกเลิกค่าปกติของแอตทริบิวต์ position และการกำหนดค่าปกติเป็น 'F' ให้กับแอตทริบิวต์ sex ในตาราง EMPLOYEE ซึ่งสามารถทำได้ดังนี้

ALTER TABLE Employee

ALTER position DROP DEFAULT;

ALTER TABLE Employee

ALTER sex SET DEFAULT 'F';

1.3 การลบตาราง

ในกรณีที่มีการสร้างตารางต่างๆ ขึ้นมาใช้งาน แล้วพบว่ามีบางตารางไม่ได้ถูกใช้งาน หรือเป็นตารางที่ซับซ้อน ก็สามารถลบตารางที่ซ้ำซ้อนเหล่านี้ออกไปจากฐานข้อมูลได้ด้วยการใช้คำสั่ง DROP TABLE ซึ่งมีรูปแบบประโยคคำสั่งดังนี้

DROP TABLE TableName [RESTRICT | CASCADE]

ตัวอย่างการลบตาราง PROPERTY_FOR_RENT สามารถทำได้ดังนี้

DROP TABLE Property_For_Rent

1.4 การสร้างดัชนี

การสร้างดัชนี ให้กับตารางจะช่วยเพิ่มประสิทธิภาพในการคิวรีข้อมูล โดยสามารถสร้างดัชนีเพื่อเรียงลำดับคีย์จากน้อยไปมาก หรือมากไปน้อยก็ได้ รวมถึงสร้างดัชนีได้มากกว่าหนึ่งคอลัมน์ และสิ่งสำคัญคือ คำสั่ง INDEX จะนำไปใช้งานได้เฉพาะกับตารางเท่านั้น ทั้งนี้ตารางที่ถูกต้องกำหนดขึ้นด้วยคำสั่งวิว (Views) จะไม่สามารถสร้างดัชนีได้ รูปแบบประโยคคำสั่งการสร้างดัชนีแสดงได้ดังนี้

CRETE [UNIQEX] INDEX IndexName

ON TableName (columName [ASC | DESC] [,...])

ตัวอย่างแสดงการสร้างดัชนีชื่อ EmpNoIdx บนคอลัมน์ empNo ในตาราง EMPLOYEE และสร้างดัชนีชื่อ propertyNoIdx บนคอลัมน์ properyNo ในตาราง PROPERTY_FOR_RENT

CREATE UNIQUE INDEX EmpNoIdx ON Employee (empNo);

CREATE UNIQUE INDEX PropertyNoIdx ON Property_For_Rent

(propertyNo);

ตัวอย่างแสดงการสร้างดัชนี RentIdx บนคอลัมน์ city และ rent ในตาราง

CREATE INDEX RentIdx ON Property_For_Rent (city, rent);

1.5 การลบดัชนี

เมื่อดัชนีได้ถูกสร้างขึ้นเพื่อใช้งานแล้วไม่ต้องการใช้งานดัชนีนี้อีกต่อไป สามารถดำเนินการลบดัชนีออกจากฐานข้อมูลนี้ได้ด้วยคำสั่ง DROP INDEX ซึ่งมีรูปแบบประโยคคำสั่งดังนี้

DROP INDEX IndexName

ตัวอย่างเช่น

DROP INDEX RentIdx

1.6 การสร้างวิว

การสร้างวิว (CREATE VIEW) เป็นการสร้างตารางสมมติขึ้นมา เพื่อนำไปให้กับผู้ใช้บางกลุ่ม โดยให้ผู้ใช้เหล่านั้นสามารถเรียกดูข้อมูลที่ต้องการจากวิวที่เรากำหนดให้เท่านั้น รูปแบบประโยคคำสั่งเขียนได้ดังนี้

CREATE VIEW ViewName [(newColumnName [, …])]

AS subselect **[WITH [CASCADED | LOCAL] CHECK OPTION]**

ตัวอย่างการสร้างวิวให้กับผู้จัดการสาขา B003 เพื่อเรียกดูข้อมูลพนักงานที่สังกัดอยู่ในสาขานี้ สามารถเขียนคำสั่งได้ดังนี้

```
CREATE VIEW Manager3Emp
AS      SELECT *
        FROM   Employee
        WHERE  branchNo = ‘ B003’ ;
```

ตัวอย่างข้างต้นเป็นการสร้างวิวชื่อ Manager3Emp โดยวิวนี้ทาง DBA ได้สร้างให้กับผู้จัดการสาขา B003 เพื่อนำไปใช้งาน ดังนั้นผู้จัดการสาขา B003 จึงสามารถเรียกดูรายชื่อพนักงานทั้งหมดผ่านวิวดังกล่าวได้ (เฉพาะสาขา B003) ซึ่งจะพบว่าหากไม่ได้กำหนดวิวดังกล่าวขึ้นมาให้กับผู้จัดการสาขา B003 โดยปล่อยให้เข้าถึงตาราง EMPLOYEE ได้โดยตรง ก็จะทำให้ผู้จัดการสาขา B003 สามารถเห็นข้อมูลของพนักงานของแต่ละสาขาได้ทั้งหมด ดังนั้นการสร้างวิวถือเป็นการสร้างข้อจำกัดในการเรียกดูข้อมูล และถือเป็นระบบความปลอดภัยชนิดหนึ่งที่ DBA สามารถกลั่นกรองเฉพาะข้อมูลบางส่วนให้กับผู้ใช้บางกลุ่ม โดยผู้จัดการสาขา B003 สามารถเรียกดูข้อมูลพนักงานทั้งหมดของตนด้วยคำสั่งดังนี้

```
SELECT * FROM Manager3Emp;
```

ผลลัพธ์ที่ได้จากคำสั่ง SELECT เพื่อเรียกดูข้อมูลแสดงในภาพที่ 7.3

empNo	empName	position	sex	birthdate	salary	branchNo
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย	F	10/09/2515	12000	B003
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	M	24/03/2520	18000	B003
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	F	03/06/2511	24000	B003

ภาพที่ 7.3 การใช้คำสั่ง SELECT เพื่อเรียกดูข้อมูลผ่านวิวที่สร้างขึ้น

1.7 การลบวิว

เมื่อต้องการยกเลิกการใช้งานวิวที่เคยสร้างขึ้นก็สามารถลบออกได้ด้วยประโยคคำสั่ง DROP VIEW ซึ่งมีรูปแบบดังนี้

DROP VIEW ViewName [RESTRICT | CASADE]

DROP VIEW Manager3Emp

ภาษาจัดการข้อมูล

ภาษาจัดการข้อมูลหรือ Data Manipulation Language : DML ใน SQL ประกอบด้วยกลุ่มคำสั่งต่างๆ ดังต่อไป

SELECT ใช้สำหรับคิวรีหรือเรียกดูข้อมูลในฐานข้อมูล

INSERT ใช้สำหรับเพิ่มข้อมูลในตาราง

UPDATE ใช้สำหรับการอัปเดตข้อมูลในตาราง

DELETE ใช้สำหรับลบข้อมูลในตาราง

ประโยคคำสั่ง SELECT เป็นคำสั่งที่ใช้สำหรับการเรียกดูข้อมูลในฐานข้อมูล การเรียกดูข้อมูลในฐานข้อมูลนั้น สามารถสร้างชุดคำสั่งได้ตั้งแต่ระดับพื้นฐานทั่วไปจนถึงระดับสูงที่มีความซับซ้อน (พนิดา พานิชกุล และ ณัฐพงษ์ วารีประเสริฐ, 2552, น. 207)

ต่อไปนี้จะเป็นตัวอย่างการใช้คำสั่ง SELECT ด้วยการใช้ข้อมูลจากฐานข้อมูล DreamHome ในบางรีเลชั่น ซึ่งประกอบด้วยรีเลชั่นทั้ง 6 โดยจะอธิบายความหมายของแต่ละรีเลชั่น และแต่ละแอตทริบิวต์ดังรายละเอียดต่อไปนี้

BRANCH	(branchNo, street, city, postcode)
สาขา	(รหัสสาขา, ถนน, จังหวัด, รหัสไปรษณีย์)
EMPLOYEE	(empNo, empName, position, sex birthdate, salary, branchNo)
พนักงาน	(รหัสพนักงาน, ชื่อ, ตำแหน่ง, เพศ, วันเกิด, เงินเดือน, รหัสสาขา)
PROPERTY_FOR_RENT	(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, empNo, branchNo)
บ้านพักที่ปล่อยเช่า	(รหัสบ้านเช่า, ถนน, จังหวัด, รหัสไปรษณีย์, ชนิดบ้านเช่า, จำนวนห้อง, ค่าเช่าต่อวัน, รหัสเจ้าของบ้านเช่า, รหัสพนักงานที่ดูแล, รหัสสาขา)

CLIENT	(clientNo, clientName, telNo, prefType, mexRent)
ลูกค้า	(รหัสลูกค้า, ชื่อลูกค้า, เบอร์โทรศัพท์, ประเภทบ้านเช่าที่ต้องการ, ราคาสูงสุดของค่าเช่าบ้านที่ลูกค้ายอมรับได้)
PRIVATE_OWNER	(ownerNo, ownerName, address, telNo)
เจ้าของบ้าน	(รหัสเจ้าของบ้าน, ชื่อเจ้าของ, ที่อยู่, เบอร์โทรศัพท์)
VIEWING	(clientNo, propertyNo, viewDate, comment)
ข้อมูลการเยี่ยมชมบ้านของลูกค้า	(รหัสลูกค้า, รหัสบ้านเช่า, วันที่เข้าเยมชม, คำแนะนำติชม)

โดยแต่ละรีเลชั่นประกอบด้วยข้อมูลที่บรรจุไว้ดังรายละเอียดต่อไปนี้

BRANCH

branchNo	street	city	postcode
B005	21 ถ.ห้วยแก้ว	เชียงใหม่	50300
B007	56 ถ.พหลโยธิน	พิษณุโลก	65150
B003	143 ถ.วิภาวดีรังสิต	กรุงเทพฯ	10110
B004	22 ถ.สหมิตร	เชียงราย	57000
B002	11 ถ.พหลโยธิน	เชียงใหม่	50300

EMPLOYEE

empNo	empName	Position	sex	birthDate	salary	branchNo
SL21	ชูชัย สุขศรี	ผู้จัดการ	M	01/10/2508	30000	005
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย	F	10/09/2515	12000	003
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	M	24/03/2520	18000	003
SA09	ปิยฉัตร เอี่ยมสุข	ผู้ช่วย	F	19/02/2521	9000	007
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	F	03/06/2511	24000	003
SL41	ลัดดา วงค์ดี	ผู้ช่วย	F	13/06/2524	9000	005

PROPERTY_FOR_RENT

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	EmpNo	branchNo
PG14	14 ถ.พหลโยธิน	พิษณุโลก	65000	บ้าน	6	650	CO46	SA09	007
PG94	19 ถ.ลำห้วย	เชียงใหม่	50310	แฟลต	4	400	CO87	SL41	005
PG04	6 ถ.วิภาวดีรังสิต	กรุงเทพฯ	10200	แฟลต	3	350	CO40		003
PG36	2 ถ.ประชาอุทิศ	กรุงเทพฯ	10160	แฟลต	3	375	CO93	SG37	003
PG21	18 ถ.พญาไท	กรุงเทพฯ	10400	บ้าน	5	600	CO87	SG37	003
PG16	5 ถ.พญาไท	กรุงเทพฯ	10400	แฟลต	4	450	CO93	SG14	003

CLIENT

clientNo	ClientName	telNo	prefType	maxRent
CR76	ยงยุทธ ธนเลิศ	02-5568891	แฟลต	425
CR56	สิราณี พรมจรรย์	02-4456328	แฟลต	350
CR74	ศรีสมร หิรัญพงศ์	02-4505568	บ้าน	750
CR62	ตะวัน ส่งศรีสุข	081-7655588	แฟลต	600

PRIVATE_OWNER

ownerNo	ownerName	address	telNo
13	นิรมล พัฒนา	พิษณุโลก	081-4445568
87	ฉัตรชัย ชุนศรี	กรุงเทพฯ	02-5369980
40	กานดา ใจมา	กรุงเทพฯ	02-4868891
93	สุขใจ แซ่ลี้	กรุงเทพฯ	02-5678811

VIEWING

clientNo	propertyNo	viewDate	comment
CR56	PA14	04/06/2550	พื้นที่ขนาดเล็ก
CR76	PG04	20/02/2550	ระยะทางไกลเกิน
CR56	PG04	26/25/2550	
CR62	PA16	14/05/2550	ไม่มีห้องครัว
CR56	PG36	28/04/2550	

1. ประโยคคำสั่ง SELECT

ประโยคคำสั่ง SELECT เป็นคำสั่งที่ใช้สำหรับเรียกดูข้อมูลจากตารางในฐานข้อมูล โดยเป็นประโยคคำสั่งที่นำมาใช้งานมากที่สุดและมีรูปแบบการใช้งานอยู่หลายรูปแบบด้วยกัน ซึ่งสามารถเรียกดูข้อมูลพร้อมเงื่อนไขประกอบ รวมทั้งการเรียกดูข้อมูลจากในหลายๆ ตาราง รูปแบบการใช้งานคำสั่งมีดังนี้ (Connolly & Begg, 2005, pp. 118-119)

```
SELECT      [DISTINCT | ALL] {* | columnexpression [as newName] ] [ , . . . .] }
FROM        tableName  [ alias ]  [ , . . . ]
    [WHERE        condition]
    [GROUP  BY    columnlist ] [HAVING  condition]
    [ORDER BY     columnlist  ]
```

โดยที่

columnExpression	เป็นการแทนค่าชื่อคอลัมน์หรือนิพจน์ expression
TableName	คือชื่อของตารางที่ต้องการเข้าถึงเพื่อวิวข้อมูล
Alias	คือชื่อย่อของตารางที่ใช้แทน TableName

สำหรับลำดับการประมวลผลในชุดคำสั่ง SELECT มีดังนี้

FROM	กำหนดตารางที่ต้องการใช้งาน
WHERE	เงื่อนไขที่สร้างขึ้นเพื่อนกลั่นกรองข้อมูลแต่ละแถวที่ต้องการ
GROUP BY	จัดกลุ่มในแถวที่มีค่าคอลัมน์เดียวกัน
HAVING	กรองกลุ่มเนื้อหาในบางเงื่อนไขจาก GROUP BY
ORDER BY	กำหนดให้เรียงลำดับผลลัพธ์ โดยที่ ASC คือการเรียงลำดับจากน้อยไปหามาก DESC คือการเรียงลำดับจากมากไปหาน้อย (ค่าปกติเมื่อไม่ได้กำหนด จะเป็นการเรียงลำดับจากน้อยไปมาก)

ตัวอย่างที่ 1 การเรียกดูข้อมูลทุกคอลัมน์ ทุกแถว

จงแสดงรายการทั้งหมดของพนักงาน

```
SELECT empNo, empName, position, sex, birthdate, salary, branchNo
FROM  Employee ;
```

หรือ

SELECT *

FROM Employee ;

empNo	empName	position	sex	birthDate	salary	branchNo
SA09	ปิยะฉัตร เอี่ยมสุข	ผู้ช่วย	F	19/22/2521	9000	B007
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	F	3/6/2511	24000	B003
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	M	24/3/2520	18000	B003
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย	F	10/9/2515	12000	B003
SG21	ชูชัย สุขศรี	ผู้จัดการ	M	1/10/2508	30000	B005
SL41	ลัดดา วงศ์ดี	ผู้ช่วย	F	13/6.2524	9000	B005

ภาพที่ 7.4 ผลลัพธ์ที่ได้จากตัวอย่างที่ 1

ตัวอย่างที่ 2 การเรียกดูข้อมูลบางคอลัมน์ และทุกๆ แถว

จงแสดงข้อมูลเงินเดือนของพนักงาน โดยแสดงเฉพาะรหัสพนักงาน ชื่อพนักงาน และเงินเดือน

SELECT empNo, empName, salary

FROM Employee ;

empNo	empName	salary
SA09	ปิยะฉัตร เอี่ยมสุข	9000
SG05	พรรัตน์ ธนะศิลป์	24000
SG14	สมศักดิ์ แซ่ตั้ง	18000
SG37	ศิริทัตร มณีจันทร์	12000
SL21	ชูชัย สุขศรี	30000
SL41	ลัดดา วงค์ดี	9000

ภาพที่ 7.5 ผลลัพธ์ที่ได้จากตัวอย่างที่ 2

ตัวอย่างที่ 3 การใช้ DISTINCT เพื่อขจัดข้อมูลซ้ำๆ ออกไป

จงแสดงรหัสบ้านเช่าทั้งหมด ในตารางการเยี่ยมชมของลูกค้า

กรณีไม่ได้ระบุ DISTINCT

SELECT propertyNo

FROM Viewing ;

propertyNo
PA14
PG04
PG36
PA16
PG04

ภาพที่ 7.6 ผลลัพธ์ที่ได้จากตัวอย่างที่ 3 : กรณีไม่ได้ระบุ DISTINCT

กรณีระบุ DISTINCT

```
SELECT DISTINCT propertyNo
FROM  Viewing ;
```

propertyNo
PA14
PG04
PG36

ภาพที่ 7.7 ผลลัพธ์ที่ได้จากตัวอย่างที่ 3 : กรณีระบุ DISTINCT

ตัวอย่างที่ 4 การนำแอตทริบิวต์มาคำนวณ

จงคำนวณเงินเดือนพนักงานใหม่ ด้วยการเพิ่มจากฐานเงินเดือน 5%

```
SELECT empNo, emName, Salary*1.05 AS newSalary
FROM Employee ;
```

empNo	empName	newSalary
SA09	ปิยะฉัตร เอี่ยมสุข	9450
SG05	พรรัตน์ ธนะศิลป์	25200
SG14	สมศักดิ์ แซ่ตั้ง	18900
SG37	ศิริทัตร มณีจันทร์	12600
SL21	ชูชัย สุขศรี	31500
SL41	ลัดดา วงค์ดี	9450

ภาพที่ 7.8 ผลลัพธ์ที่ได้จากตัวอย่างที่ 4

ตัวอย่างที่ 5 การเรียกดูข้อมูลแบบมีเงื่อนไข

จงแสดงข้อมูลพนักงานที่มีเงินเดือนมากกว่า 10000 บาท

```
SELECT empNo, empName, position, salary
FROM Employee
WHERE salary>10000;
```

empNo	empName	position	salary
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย	12000
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	18000
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	24000

ภาพที่ 7.9 ผลลัพธ์ที่ได้จากตัวอย่างที่ 5 กรณีกำหนดเงื่อนไขเงินเดือน > 10000

เครื่องหมายเปรียบเทียบประกอบด้วยโอเปอเรเตอร์ต่างๆ ดังต่อไปนี้

=	เท่ากับ	<=	น้อยกว่าหรือเท่ากับ
<	น้อยกว่า	>=	มากกว่าหรือเท่ากับ
>	มากกว่า	<> , !=	ไม่เท่ากับ

นอกจากเงื่อนไขดังกล่าวแล้ว ยังสามารถใช้โอเปอเรเตอร์ทางตรรกะ เช่น AND,OR และ NOT เข้าร่วมได้

จงแสดงข้อมูลของสาขา เฉพาะสาขาที่ตั้งอยู่ในจังหวัดเชียงใหม่ หรือกรุงเทพฯ

```
SELECT *
FROM Branch
WHERE city = 'เชียงใหม่' OR city= 'กรุงเทพ' ;
```

branchNo	street	city	postcode
B005	21 ถ.ห้วยแก้ว	เชียงใหม่	50300
B003	143 ถ.วิภาวดีรังสิต	กรุงเทพฯ	10110
B002	11 ถ.พหลโยธิน	เชียงใหม่	50300

ภาพที่ 7.10 ผลลัพธ์ที่ได้ กรณีกำหนดเงื่อนไขพร้อมโอเปอเรเตอร์ทางตรรกะ OR

ตัวอย่างที่ 6 การใช้โอเปอเรเตอร์ BETWEEN/NOT BETWEEN ในการตรวจสอบค่าที่อยู่ภายในช่วงที่กำหนด

จงแสดงข้อมูลพนักงานที่มีเงินเดือนอยู่ในช่วง 20000 และ 30000 บาท

```
SELECT empNo, empName, position, salary
FROM Employee
WHERE salary BETWEEN 20000 AND 30000;
```

empNo	empName	position	salary
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	24000

ภาพที่ 7.11 ผลลัพธ์ที่ได้จากตัวอย่างที่ 6

ตัวอย่างที่ 7 การใช้โอเปอเรเตอร์ IN/NOT IN เพื่อตรวจสอบสมาชิกภายในกลุ่ม

จงแสดงข้อมูลพนักงานที่อยู่ในกลุ่มของผู้จัดการและซูเปอร์ไวเซอร์

```
SELECT empNo, empName, position
FROM Employee
WHERE position IN ('ผู้จัดการ','ซูเปอร์ไวเซอร์') ;
```

empNo	empName	position
SL21	ชูชัย สุขศรี	ผู้จัดการ
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ

ภาพที่ 7.12 ผลลัพธ์ที่ได้จากตัวอย่างที่ 7

ตัวอย่างที่ 8 การใช้โอเปอเรเตอร์ LIKE/NOT LIKE

LIKE และ NOT LIKE เป็นโอเปอเรเตอร์ที่ใช้ในการค้นหาข้อมูลชนิดตัวอักษรเท่านั้น โดยอาจยังไม่ทราบค่าที่แน่นอนของข้อมูลทั้งหมดที่ต้องการค้นหา ดังนั้นจึงมีการใช้เครื่องหมาย Wild Card ช่วย ซึ่งประกอบด้วยสัญลักษณ์ % และ _ โดยที่

% แทนตัวอักษรหลายตัว เช่น empNo LIKE ‘SG%’ หมายถึงค้นหารหัสพนักงานที่ขึ้นต้นด้วยอักษร SG ทั้งหมด

_ แทนตัวอักษร 1 ตัว เช่น empNo LIKE ‘_a%’ หมายถึงให้ค้นหาชื่อพนักงานที่ขึ้นอักษรตัวแรกเป็นตัวใดก็ได้หนึ่งตัว ส่วนตัวอักษรลำดับถัดไปต้องเป็นตัว a และตัวถัดไปที่เหลือจะเป็นอักษรใดก็ได้

อย่างไรก็ตามใน MS-Access จะใช้สัญลักษณ์ * แทนสัญลักษณ์ % และใช้สัญลักษณ์ ? แทน _

จงแสดงข้อมูลเจ้าของบ้านเช่าที่อยู่ในจังหวัดกรุงเทพฯ

```
SELECT ownerNo, ownerName,address,telNo
FROM Private_Owner
WHERE address LIKE ‘*กรุงเทพฯ*’ ;
```

ownerNo	ownerName	address	telNo
CO87	ฉัตรชัย ชุนศรี	กรุงเทพฯ	025369980
CO40	กานดา ใจมา	กรุงเทพฯ	024868894
CO93	สุขใจ แซ่ลี้	กรุงเทพฯ	025678811

ภาพที่ 7.13 ผลลัพธ์ที่ได้จากตัวอย่างที่ 8

ตัวอย่างที่ 9 การใช้ NULL ตรวจสอบข้อมูล

จงแสดงข้อมูลลูกค้าที่เข้ามาเยี่ยมชมบ้านเช่ารหัส PG04 เฉพาะข้อมูลที่ไม่มีรายการการติชมจากลูกค้า

```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo=’PG04’ AND comment IS NULL ;
```

clientNo	viewDate
CR56	26/5/2550

ภาพที่ 7.14 ผลลัพธ์ที่ได้จากตัวอย่างที่ 9

ตัวอย่างที่ 10 การเรียงลำดับข้อมูลคอลัมน์เดียว (Single-Column Ordering)

จงแสดงข้อมูลพนักงานด้วยการเรียงลำดับตามเงินเดือนจากมากไปน้อย

```
SELECT empNo, empName, salary
FROM Employee
ORDER BY salary DESC ;
```

empNo	empName	salary
SL21	ชูชัย สุขศรี	30000
SL05	พรรัตน์ ธนะศิลป์	24000
SL14	ศิริทัตร มณีจันทร์	18000
SL37	สมศักดิ์ แซ่ตั้ง	12000
SL41	ลัดดา วงค์ดี	9000
SL09	ปิยะฉัตร เอี่ยมสุข	9000

ภาพที่ 7.15 ผลลัพธ์ที่ได้จากตัวอย่างที่ 10

ตัวอย่างที่ 11 การเรียงลำดับข้อมูลแบบหลายคอลัมน์ (Multiple-Column Ordering)

จงแสดงข้อมูลบ้านเช่าด้วยการเรียงลำดับตามชนิดของบ้านเช่าและค่าเช่าบ้านต่อวัน โดยเรียงจากมากไปน้อย

```
SELECT propertyNo, type, rooms, rent
FROM Property_For_Rent
ORDER BY type, rent  DESC ;
```

propertyNo	type	rooms	rent
PA14	บ้าน	6	650
PG21	บ้าน	5	600
PG16	แฟลต	4	450
PL94	แฟลต	4	400
PG36	แฟลต	3	375
PG04	แฟลต	3	350

ภาพที่ 7.16 ผลลัพธ์ที่ได้จากตัวอย่างที่ 11

2. ฟังก์ชันการรวมใน SQL

จากตัวอย่างข้างต้น ได้มีการเรียนรู้ ถึงการเรียกดูข้อมูลที่ต้องการตามแต่ละแถวและคอลัมน์ที่กำหนดจากเงื่อนไขต่างๆ แต่ในบางครั้งเราต้องการหาผลรวมข้อมูล ซึ่งเครื่องจะต้องคำนวณยอดรวมทั้งสิ้นจากต้นแฟ้มไปจนถึงท้ายแฟ้ม มาตรฐาน IOS ได้กำหนดฟังก์ชันการรวมไว้ 5 ฟังก์ชันด้วยกันคือ

COUNT เป็นฟังก์ชันการนับจำนวน

SUM เป็นฟังก์ชันหาผลรวม

AVG เป็นฟังก์ชันหาค่าเฉลี่ย

MIN เป็นฟังก์ชันหาค่าต่ำสุด

MAX เป็นฟังก์ชันหาค่าสูงสุด

ตัวอย่างที่ 12 การใช้ฟังก์ชัน COUNT(*)

ต้องการทราบจำนวนบ้านเช่าที่มีค่าเช่ามากกว่า 350 บาทต่อวัน

```
SELECT COUNT(*) AS myCount
FROM Property_For_Rent
WHERE rent>350 ;
```

myCount
5

ภาพที่ 7.17 ผลลัพธ์ที่ได้จากตัวอย่างที่ 12

ตัวอย่างที่ 13 การใช้ฟังก์ชัน COUNT ร่วมกับฟังก์ชัน SUM

ต้องการทราบว่ามีผู้จัดการอยู่กี่คน และยอดรวมเงินเดือนของผู้จัดการทั้งหมดคือเท่าไหร่

```
SELECT COUNT(empNo) AS myCount, SUM(salary) AS mySUM
FROM Employee
WHERE position='ผู้จัดการ' ;
```

myCount	mySum
2	5400

ภาพที่ 7.18 ผลลัพธ์ที่ได้จากตัวอย่างที่ 13

ตัวอย่างที่ 14 การใช้ฟังก์ชัน MIN, MAX และ AVG

ต้องการทราบยอดเงินเดือนพนักงานที่ต่ำที่สุดและสูงที่สุด รวมถึงค่าเฉลี่ยเงินเดือนของพนักงานทั้งหมด

```
SELECT MIN(salary) AS myMin, MAX(salary) AS myMax,AVG(salary) AS myAvg
FROM Employee ;
```

myMin	myMAX	myAvg
9000	30000	17000

ภาพที่ 7.19 ผลลัพธ์ที่ได้จากตัวอย่างที่ 14

ตัวอย่างที่ 15 การใช้ประโยค GROUP BY

จากฟังก์ชั่นการรวมที่ผ่านมา เป็นการคิวรีเพื่อหายอดรวม (Total) ของผลลัพธ์ และจะได้ค่าสรุปเพียงค่าเดียวเท่านั้น แต่ในบางครั้งเราอาจมีความต้องการผลสรุปที่เป็นลักษณะแบบ Subtotal เพื่อจัดกลุ่มแยกประเภทตามค่าของแต่ละคอลัมน์ ซึ่งสามารถทำได้ด้วยการใช้ GROUP BY

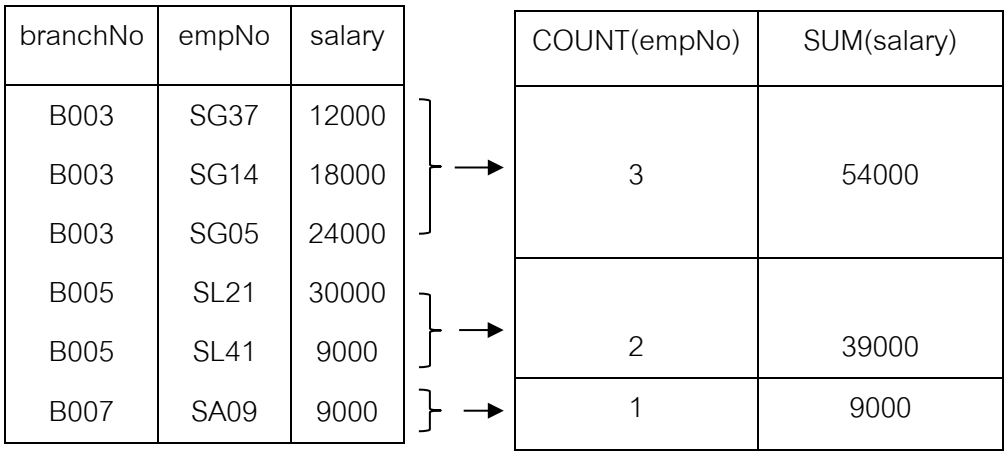
จงแสดงยอดรวมพนักงานที่ทำงานอยู่ในแต่ละสาขา พร้อมเงินเดือนรวมที่ต้องจ่ายให้แต่ละสาขา

```
SELECT branchNo, COUNT(empNo) AS myCount, SUM(salarary) AS mySum
FROM Employee
GROUP BY branchNo
ORDER BY branchNo ;
```

branchNo	myCount	mySum
B003	3	54000
B005	2	39000
B007	1	9000

ภาพที่ 7.20 ผลลัพธ์ที่ได้จากตัวอย่างที่ 15

ผลลัพธ์ที่ได้จากภาพที่ 7.20 จะพบว่าแต่ละกลุ่มนั้น SQL จะมีวิธีหาผลรวมของจำนวนพนักงาน และคำนวณผลรวมของเงินเดือนด้วยการแบ่งกลุ่มพนักงานตามแต่ละสาขา ซึ่งเป็นไปตามหลักการดังภาพที่ 7.21 ต่อไปนี้



branchNo	empNo	salary
B003	SG37	12000
B003	SG14	18000
B003	SG05	24000
B005	SL21	30000
B005	SL41	9000
B007	SA09	9000

COUNT(empNo)	SUM(salary)
3	54000
2	39000
1	9000

ภาพที่ 7.21 การหาผลรวมของจำนวนพนักงานและผลรวมของเงินเดือน

ตัวอย่างที่ 16 การใช้ประโยค HAVING

ประโยค HAVING ถูกออกแบบมาเพื่อใช้งานร่วมกับประโยค GROUP BY โดยประโยค HAVING นี้จะทำการแสดงข้อมูลที่ผ่านการจัดกลุ่มด้วย GROUP BY แล้วตามเงื่อนไขที่กำหนดไว้ใน HAVING

แสดงยอดพนักงานที่ทำงานในแต่ละสาขา และคำนวณหาเงินเดือนที่ต้องจ่ายทั้งสิ้นในแต่ละสาขาด้วยการแสดงเฉพาะสาขาที่มีพนักงานสังกัดอยู่มากกว่า 1 คนเท่านั้น

```
SELECT branchNo, COUNT(empNo) AS myCount, SUM(salary) AS mySum
FROM Employee
GROUP BY branchNo
HAVING COUNT(empNo) >1
ORDER BY branchNo ;
```

branchNo	myCount	mySum
B003	3	54000
B005	2	39000

ภาพที่ 7.22 ผลลัพธ์ที่ได้จากตัวอย่างที่ 16

ตัวอย่างที่ 17 การใช้ Subqueries

ในการเรียกดูข้อมูลด้วยคำสั่ง SELECT นั้น สามารถใช้ชุดคำสั่ง SQL อย่างง่ายเพื่อเรียกดูข้อมูลที่ต้องการ แต่ในบางครั้งอาจจำเป็นต้องใช้ชุดคำสั่งที่มีความซับซ้อนยิ่งขึ้นไปอีกในการเรียกดูข้อมูล ซึ่งเรียกว่า Subqueries หรือ Nested Query

จงแสดงข้อมูลพนักงานที่ทำงานสังกัดสาขาที่ตั้งอยู่ที่ 143 ถ.วิภาวดีรังสิต

```
SELECT empNo, empName, position
FROM Employee
WHERE branchNo =    (SELECT branchNo
                     FROM Branch
                     WHERE street = '143 ถ.วิภาวดีรังสิต') ;
```

คำสั่งภายใน

โดยคำสั่งภายในจะถูกประมวลผลก่อน

empNo	empName	Position
SG37	ศิริทัตร มณีจันทร์	ผู้ช่วย
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอรี
SG05	พรรัตน์ นะศิลป์	ผู้จัดการ

ภาพที่ 7.23 ผลลัพธ์ที่ได้จากตัวอย่างที่ 17

เนื่องจากการค้นหารหัสสาขาที่ตั้งอยู่เลขที่ "143 ถ.วิภาวดีรังสิต" คือ B003 การเรียกดูข้อมูลพนักงานทั้งหมดที่ทำงานอยู่ในสาขา B003 ดังกล่าวพบว่า ผลลัพธ์ที่ได้จะเหมือนกับคำสั่งต่อไปนี้

```
SELECT empNo , empName , position
FROM Employee
WHERE branchNo = 'B003' ;
```

ตัวอย่างที่ 18 การใช้ Subqueries ร่วมกันฟังก์ชันการรวม

จงแสดงข้อมูลพนักงานที่มีเงินเดือนสูงกว่าเงินเดือนเฉลี่ย พร้อมทั้งแสดงส่วนต่างของเงินเดือนว่าแต่ละคนมีเงินเดือนเฉลี่ยเท่าไร

```
SELECT empNo, empName, Position,
                Salary – ( SELECT AVG (salary) FROM Employee)  AS salDiff
FROM Employee
WHERE salary > (SELECT AVG (salary) FROM Employee) ;
```

empNo	empName	position	salDiff
SL21	ชูชัย สุขศรี	ผู้จัดการ	13000
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	1000
SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	7000

ภาพที่ 7.24 ผลลัพธ์ที่ได้จากตัวอย่างที่ 18

ตัวอย่างที่ 19 การใช้ Nested Subqueries ด้วยการใช้โอเปอเรเตอร์ IN

จงแสดงข้อมูลบ้านเช่าที่ถูกดูแลโดยพนักงานที่ทำงานอยู่ในสาขาที่ตั้งอยู่เลขที่ 143 ถ. วิภาวดีรังสิต

```
SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM Property_For_Rent
WHERE empNo IN ( SELECT empNo
                        FROM Employee
                        WHERE branchNo = (SELECT branchNo
                                                FROM Branch
                                                WHERE street = '143 ถ.วิภาวดีรังสิต') ) ;
```

ในทำนองเดียวกัน การประมวลผลชุดคำสั่งจะกระทำชุดคำสั่งภายในสุดก่อน แล้วทยอยออกไปด้านนอกสุด โดยชุดทำสั่งดังตัวอย่างที่ 19 นี้ จะมีชุดคำสั่งที่อยู่ส่วนในสุด ส่วนกลาง และส่วนนอกสุด โดยลำดับแรกจะคิวรีข้อมูลด้วยการค้นหารหัสสาขาที่ตั้งอยู่เลขที่ “143 ถ.วิภาวดีรังสิต” ก่อน ต่อมาจึงคิวรีข้อมูลด้วยการค้นหาพนักงานที่ทำงานอยู่ในรหัสสาขานั้น (ที่ได้จากส่วนในสุด) โดยผลลัพธ์ที่ได้จะพาข้อมูลหลายแถวด้วยกัน ดังนั้นจึงไม่สามารถใช้เครื่องหมายเปรียบเทียบ = ตรงชุดคำสั่งส่วนนอกสุดได้ ในที่นี้จึงใช้ IN แทนและที่ชุดคำสั่งส่วนนอกสุดก็จะคิวรีข้อมูลบ้านเช่าที่ถูกดูแลโดยพนักงานแต่ละคนที่ถูกระบุในคิวรีที่ได้จากชุดคำสั่งส่วนกลาง โดยผลลัพธ์ที่ได้ก็จะเป็นได้ดังภาพที่ 7.25

propertyNo	Street	city	postcode	type	rooms	rent
PG36	2.ถ.ประชาอุทิศ	กรุงเทพฯ	10160	แฟลต	3	375
PG21	18 ถ.พญาไท	กรุงเทพฯ	10400	บ้าน	5	600
PG16	5 ถ. พญาไท	กรุงเทพฯ	10400	แฟลต	4	450

ภาพที่ 7.25 ผลลัพธ์ที่ได้จากตัวอย่างที่ 19

การใช้ ANY และ ALL คำว่า **'ANY'** และ **'ALL'** สามารถนำไปใช้ร่วมกับซับคิวรี (Subqueries) ได้ โดยหากซับคิวรีมีการใช้ ALL เงื่อนไขจะเป็นจริงก็ต่อเมื่อ ถ้าข้อมูลทุกค่ามีค่าตรงกันทั้งหมดในซับคิวรี หากซับคิวรีใช้ ANY เงื่อนไขจะเป็นจริงก็ต่อเมื่อมีข้อมูลบางส่วนที่มีค่าตรงกัน (หนึ่งหรือมากกว่า) ในซับคิวรี และหากซับคิวรีมีผลลัพธ์ว่างเปล่า เงื่อนไข ALL จะรีเทิร์นค่าจริง ในขณะที่เงื่อนไข ANY จะรีเทิร์นค่าเท็จกลับไป สำหรับมาตรฐาน ISO ได้กำหนดการใช้คำว่า **SOME** แทนคำว่า ANY

ตัวอย่างที่ 20 การใช้ SOME

จงค้นหาข้อมูลพนักงานว่ามีคนใดบ้างที่มีเงินเดือนสูงกว่าเงินเดือนคนอื่นอย่างน้อย 1 คน เฉพาะในสาขา B003

```
SELECT empNo, position, salary
FROM Employee
WHERE salary > SOME ( SELECT salary
                      FROM Employee
                      WHERE branchNo = 'B003') ;
```

empNo	empName	position	salary
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000
SG14	สมศักดิ์ แซ่ตั้ง	ซูเปอร์ไวเซอร์	18000
SG	พรรัตน์ ธนะศิลป์	ผู้จัดการ	24000

ภาพที่ 7.26 ผลลัพธ์ที่ได้จากตัวอย่างที่ 20

ตัวอย่างที่ 21 การใช้ ALL

จงค้นหาข้อมูลพนักงานว่ามีคนใดบ้างที่มีเงินเดือนสูงกว่าเงินเดือนพนักงานทุกคนที่อยู่ในสาขา B003

```
SELECT empNo, position, salary
FROM Employee
WHERE salary >  ALL ( SELECT salary
                      FROM Employee
                      WHERE branchNo = 'B003') ;
```

empNo	empName	position	salary
SL21	ชูชัย สุขศรี	ผู้จัดการ	30000

ภาพที่ 7.27 ผลลัพธ์ที่ได้จากตัวอย่างที่ 21

การคิวรีหลายตาราง (Multi-table Queries) จากตัวอย่างทั้งหมดที่ผ่านมา ผลลัพธ์ของคอลัมน์ที่ได้ จากการประมวลผลชุดคำสั่ง SQL จะได้มาจากตารางเดียว ซึ่งในการใช้งานจริงๆ แล้ว อาจจะยังไม่สามารถตอบสนองได้อย่างเพียงพอ ดังนั้นในที่นี้จะเป็นการรวมคอลัมน์จากหลายๆ ตารางไปเก็บไว้ในตารางที่เป็นผลลัพธ์ด้วยการใช้โอเปอเรชั่น Join

โอเปอเรชั่น Join ใน SQL จะทำการรวมข้อมูลสารสนเทศของสองตารางเข้าด้วยกัน ด้วยการจับคู่ความสัมพันธ์ระหว่างแถวจากตารางทั้งสอง โดยคู่ของแถวที่ได้จากการ Join ตารางจะได้มาจากการจับคู่คอลัมน์ที่ตรงกันของตารางทั้งสองนั่นเอง

ตัวอย่างที่ 22 วิธีการ Join อย่างง่าย

จงแสดงรายชื่อของลูกค้าทั้งหมดที่เข้าเยี่ยมชมบ้านเช่า พร้อมคำติชม

```
SELECT c.clientNo, clientName , v.propertyNo , comment
FROM Client AS c, Viewing AS v
WHERE c.clientNo=v.clientNO ;
```

clientNO	clientName	propertyNo	comment
CR76	ยงยุทธ ธนเลิศ	PG04	ระยะทางไกลเกิน
CR56	สิราณี พรมจรรย์	PA14	พื้นที่ขนาดเล็ก
CR56	สิราณี พรมจรรย์	PG04	
CR56	สิราณี พรมจรรย์	PG36	
CR62	ตะวัน ส่งศรีสุข	PA14	ไม่มีห้องครัว

ภาพที่ 7.28 ผลลัพธ์ที่ได้จากตัวอย่างที่ 22

จะพบว่าผลลัพธ์ของตารางที่ได้จะมีการนำข้อมูลจากตาราง CLIENT และ VIEWING มีรวมกันโดยให้ดูจากชุดคำสั่ง SQL ข้างต้นตรงหลัง FROM จะมีการระบุตารางที่ใช้งานมากกว่าหนึ่งตาราง (Client AS c, Viewing AS v) โดยคำที่ระบุหลัง AS จะใช้แทนชื่อตารางจริง ทั้งนี้ก็เพื่อสะดวกในการอ้างอิงหรือสะดวกต่อการสร้างเงื่อนไขนั่นเอง

ตัวอย่างที่ 23 การเรียงลำดับผลลัพธ์ที่ได้จากการ Join

ในแต่ละสาขาให้แสดงรหัสและชื่อพนักงานที่ดูแลบ้างเช่า รวมถึงรายละเอียดบ้านเช่าที่ถูกดูแล โดยให้เรียงลำดับข้อมูลตามรหัสสาขา รหัสพนักงาน และรหัสบ้านเช่า

```
SELECT e.branchNo , e.empNO , e.empName , p.propertyNo , p.street
FROM Employee AS e, Property_For_Rent AS p
WHERE e.empNo=p.empNo
ORDER BY e.branchNo , propertyNo ;
```

branchNo	empNo	empName	propertyNo	street
B003	SG14	สมศักดิ์ แซ่ตั้ง	PG16	5 ถ.พญาไท
B003	SG37	ศิริทัตร มณีจันทร์	PG21	18 ถ.พญาไท
B003	SG37	ศิริทัตร มณีจันทร์	PG36	2 ถ.ประชาอุทิศ
B005	SL41	ลัดดา วงศ์ดี	PL94	14 ถ.ลำห้วย
B007	SA09	ปิยะฉัตร เอี่ยมสุข	PA14	19 ถ.พหลโยธิน

ภาพที่ 7.29 ผลลัพธ์ที่ได้จากตัวอย่างที่ 23

ตัวอย่างที่ 24 การ Join สามตาราง

ในแต่ละสาขา ให้แสดงรหัสและชื่อพนักงานที่ดูแลบ้านเช่า พร้อมทั้งที่ตั้งจังหวัดของแต่ละสาขา และ รายละเอียดบ้านเช่าที่ถูกดูแลโดยพนักงาน

```
SELECT b.branchNo , b.city , e.empNo , e.empName , p.propertyNo , p.street
FROM Branch AS b, Employee AS e, Property_For_Rent AS p
WHERE b.branchNo=e.branchNo AND e.empNo=p.empNo
ORDER BY b.branchNo , e.empNo, propertyNo ;
```

branchNo	city	empNo	empName	propertyNo	street
B003	กรุงเทพ	SG14	สมศักดิ์ แซ่ตั้ง	PG16	5 ถ.พญาไท
B003	กรุงเทพ	SG37	ศิริทัตร มณีจันทร์	PG21	18 ถ.พญาไท
B003	กรุงเทพ	SG37	ศิริทัตร มณีจันทร์	PG26	2 ถ.ประชาอุทิต
B005	เชียงใหม่	SL41	ลัดดา วงค์ดี	PL94	14 ถ.ลำห้วย
B007	พิษณุโลก	SA09	ปิยะฉัตร เอี่ยมสุข	PA14	19 ถ.พหลโยธิน

ภาพที่ 7.30 ผลลัพธ์ที่ได้จากตัวอย่างที่ 24

ตัวอย่างที่ 25 การรวมกลุ่มแบบ Multiple Grouping Columns

จงหายอดรวมของบ้านเช่าที่ดูแลโดยพนักงานแต่ละคน

```
SELECT e.branchNo, e.empNo , COUNT(*) AS myCount
FROM Employee AS e, Property_For_Rent AS P
WHERE e.empNo=p.empNo
GROUP BY e.branchNo, e.empNo
ORDER BY e.branchNo, e.empNo ;
```

branchNo	empNo	myCount
B003	SG14	1
B003	SG27	2
B005	SL41	1
B007	SA09	1

ภาพที่ 7.31 ผลลัพธ์ที่ได้จากตัวอย่างที่ 25

ตัวอย่างที่ 26 การใช้ UNION

สำหรับ Union ในที่นี้ก็คือโอเปอเรชั่น Union (U) ใน Relational Algebra นั่นเอง ซึ่งเป็นการเชื่อมความสัมพันธ์ด้วยการนำรีเลชั่น R และรีเลชั่น S มายูเนียนกัน ผลลัพธ์ที่ได้ก็คือจำนวนทูเพิลทั้งหมดในรีเลชั่น R และจำนวนทูเพิลทั้งหมดในรีเลชั่น S มารวมกันอยู่ในรีเลชั่นใหม่ โดยทูเพิลที่ซ้ำซ้อนจะถูกขจัดออกไป

จงแสดงรายชื่อจังหวัดทั้งหมดที่มีอยู่ในแต่ละสาขาหรือบ้านเช่า

```
(SELECT city
 FROM Branch
 WHERE city IS NOT NULL)
 UNION (SELECT city
 FROM Property_For_Rent
 WHERE city IS NOT NULL) ;
```

city
กรุงเทพฯ
เชียงราย
เชียงใหม่
พิษณุโลก

ภาพที่ 7.32 ผลลัพธ์ที่ได้จากตัวอย่างที่ 26

ตัวอย่างที่ 27 การใช้ INTERSECT

INTERSECT เป็นการนำรีเลชั่นที่ประกอบไปด้วยกลุ่มของทูเพิลทั้งหมดที่มีอยู่ทั้งใน R และ S

จงแสดงรายชื่อจังหวัดทั้งหมดที่มีอยู่ในสาขาและบ้านเช่า

```
(SELECT city
 FROM Branch
 INTERSECT
(SELECT city
 FROM Property_For_Rent ) ;
```

อย่างไรก็ตาม ใน MS-Access ไม่ได้มีคำสั่ง INTERSECT มาให้ใช้โดยตรงแต่สามารถ DISTINCT และ EXISTS แทนได้ ดังชุดคำสั่งต่อไปนี้

```
SELECT DISTINCT b.city
FROM Branch AS b, Property_For_Rent  AS p
WHERE b.city=p.city ;
```

หรือ

```
SELECT DISTINT city
FROM Branch AS b
WHERE EXISTS (SELECT *
FROM Property_For_Rent AS p
WHERE b.city = p.city) ;
```

city
กรุงเทพฯ
เชียงใหม่
พิษณุโลก

ภาพที่ 7.33 ผลลัพธ์ที่ได้จากตัวอย่างที่ 27

ตัวอย่างที่ 28 การใช้ EXCEPT

EXCEPT เป็นโอเปอเรชั่นชนิดหนึ่ง ซึ่งผลลัพธ์ที่ได้ในรีเลชั่นใหม่จะได้ทูเพิลเฉพาะที่อยู่ในรีเลชั่น R แต่ไม่อยู่ในรีเลชั่น S อย่างไรก็ตาม สำหรับ DBMS บางผลิตภัณฑ์จะใช้ EXCEPT แทน DIFFERENCE ในขณะที่ MS-Access จะไม่สนับสนุนคำสั่งดังกล่าว แต่จะใช้ NOT IN หรือ NOT EXISTS แทน

จงแสดงรายชื่อจังหวัดที่มีอยู่ในสาขา แต่ไม่อยู่ในบ้านเช่า

```
(SELECT city
 FROM Branch)
 EXCEPT
(SELECT city
 FROM Property_For_Rent) ;
```

สำหรับใน MS-Access สามารถประยุกต์การเขียนชุดคำสั่งเหล่านี้ได้ดังนี้

```
SELECT DISTINT city
FROM Branch AS b
WHERE NOT EXISTS
        (SELECT *
        FROM Property_For_Rent AS p
        WHERE b.city = p.city) ;
```

3. การอัปเดตฐานข้อมูล

หลังจากได้เรียนรู้ถึงวิธีการใช้ประโยคคำสั่ง SELECT ผ่านมาแล้ว ในลำดับต่อไปจะกล่าวถึงภาษาจัดการข้อมูล ได้แก่ คำสั่ง INSERT, UPDATE และ DELETE

ตัวอย่างที่ 29 การใช้คำสั่ง INSERT. . . VALUES

รูปแบบการใช้ประโยคคำสั่ง INSERT โดยทั่วไปคือ

INSERT INTO TableName [(columnList)]

VALUES (dataValueList)

จงแทรกข้อมูลของสาขาใหม่ในตาราง BRANCH

INSERT INTO Branch

VALUES (‘B008’ , ’40 ถ.เลียบคลองทวีวัฒนา’, ‘กรุงเทพ’, ‘10160’);

จากการแทรกข้อมูลข้างต้นจะต้องลำดับข้อมูลในแต่ละคอลัมน์ให้ตรงกับโครงสร้าง แต่หากต้องการให้บางคอลัมน์ไม่ต้องบรรจุค่าให้แทนค่า NULL ลงไปเพื่อคงลำดับของข้อมูลในคอลัมน์ให้ถูกต้องตามโครงสร้าง ดังชุดคำสั่งต่อไปนี้ระบุค่า Street เป็นค่า NULL

INSERT INTO Branch

VALUES (‘B009’, NULL, ‘เชียงราย’, ‘57000’);

หรืออาจระบุชื่อคอมลัมน์และค่าข้อมูลลงไป ดังชุดคำสั่งต่อไปนี้

INSERT INTO Branch (branchNo, city, postcode)

VALUES (‘B010’, ‘กรุงเทพ’, ‘ 10400’);

ตัวอย่างที่ 30 การใช้คำสั่ง INSERT. . . SELECT

ชุดคำสั่ง INSERT ยังสามารถนำไปใช้เพื่อการคัดลอกข้อมูลจากตารางหนึ่งหรือมากกว่าเพื่อไปเก็บไว้ในอีกตารางหนึ่ง โดยมีรูปแบบประโยคคำสั่งดังนี้

INSERT INTO TableName [(columnList)]

SELECT . . .

โดยที่ TableName และ ColumnList จะต้องถูกต้องถูกสร้างเป็นตารางเปล่าขึ้นมาก่อน

จงคัดลอกข้อมูลพนักงานที่มีตำแหน่งเป็นผู้จัดการทั้งหมดไปไว้ในตาราง EMPLOYEE_MANAGER

```
INSERT INTO Employee_Manager
SELECT
FROM  Employee
WHERE position = 'ผู้จัดการ' ;
```

EMPLOYEE_MANAGER : Table							
	empNo	empName	postion	sex	birthdate	salary	branchNo
	SG05	พรรัตน์ ธนะศิลป์	ผู้จัดการ	F	3/6/2511	24000	B003
	SL21	ชูชัย สุขศรี	ผู้จัดการ	M	1/10/2508	30000	B005
						0	

ภาพที่ 7.34 ตาราง EMPLOYEE_MANAGER ที่มีข้อมูลเฉพาะพนักงานที่เป็นผู้จัดการ

ตัวอย่างที่ 31 การใช้คำสั่ง UPDATE

ชุดคำสั่ง UPDATE จะนำไปใช้สำหรับปรับปรุงค่าข้อมูลในตาราง โดยมีรูปแบบประโยคคำสั่งดังนี้

```
UPDATE TableName
SET columnName1=dataValue [, columnName2=dataValue2 . . .]
[WHERE seachCondition]
```

จงปรับเงินเดือนพนักงานทั้งหมดเพิ่มขึ้น 3% จากฐานเงินเดือน

```
UPDATE Employee
SET salary = salary * 1.03;
```

ตัวอย่างที่ 32 การใช้คำสั่ง UPDATE พร้อมระบุเงื่อนไข

จงปรับเงินเดือนพนักงานทั้งหมดที่มีตำแหน่งเป็นผู้จัดการเพิ่มขึ้น 5% จากฐานเงินเดือน

```
UPDATE Employee
SET salary = salary*1.05
WHERE position = 'ผู้จัดการ';
```

ตัวอย่างที่ 33 การใช้คำสั่ง UPDATE เพื่ออัปเดตข้อมูลหลายคอลัมน์

จงปรับตำแหน่งพนักงานรหัส SG14 ชื่อสมศักดิ์ แซ่ตั้ง เป็นผู้จัดการสาขา และเปลี่ยนเงินเดือนจากเดิมเป็น 20,000 บาท

```
UPDATE Employee
SET postion = 'ผู้จัดการ', salary = 20000
WHERE empNo = SG14 ' ;
```

ตัวอย่างที่ 34 การใช้คำสั่ง DELETE เพื่อลบแถวที่ต้องการ

ชุดคำสั่ง DELETE นำไปใช้เพื่อลบแถวในตารางที่ต้องการ โดยมีรูปแบบประโยคคำสั่งดังนี้

```
DELETE FROM TableName
[WHERE searchCondition]
```

จงลบรหัสบ้านเช่ารหัส PG04 ทั้งหมด ออกจากตาราง VIEWING

```
DELETE FROM Viewing
WHERE propertyNo = 'PG04' ;
```

ตัวอย่างที่ 35 การใช้คำสั่ง DELETE เพื่อลบข้อมูลทั้งหมดในตาราง

จงลบข้อมูลทั้งหมดออกจากตาราง VIEWING

```
DELETE FROM Viewing;
```

จะเห็นได้ว่า หากไม่มีการระบุเงื่อนไข WHERE จะหมายถึงการลบข้อมูลทั้งหมดออกจากตารางโดยยังเหลือโครงสร้างไว้เหมือนเดิม เพื่อให้สามารถนำไปใช้จัดเก็บข้อมูลเพิ่มเติมต่อไป

สรุป

ในบทนี้ได้กล่าวถึงภาษาในการจัดการฐานข้อมูล SQL เป็นภาษาที่ใช้ในการสืบค้นข้อมูลแบบมีโครงสร้างและเป็นภาษามาตรฐานบนระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งถูกนำไปใช้งานในคอมพิวเตอร์หลายระดับ เช่น เมนเฟรมคอมพิวเตอร์ ไมโครคอมพิวเตอร์ เป็นต้น ภาษาฐานข้อมูลต้องสามารถสนับสนุนงานด้านการสร้างฐานข้อมูลและโครงสร้างรีเลชั่น การจัดการข้อมูลพื้นฐานได้แก่ การเพิ่ม การปรับปรุง และการลบข้อมูล รวมถึงงานคิวรีข้อมูล การใช้งาน

คำสั่งภาษา SQL มี 2 ชนิดคือ แบบโต้ตอบ และแบบฝังในตัวโปรแกรม คำสั่ง SQL มี 3 ประเภท ได้แก่ ภาษานิยามข้อมูล ภาษาจัดการข้อมูล และภาษาควบคุมข้อมูล ภาษานิยามข้อมูลใน SQL ประกอบด้วย สคีมา โดเมน ตาราง วิว และดัชนี ซึ่งสามารถถูกสร้างขึ้น หรือถูกทำลายได้ ภาษาจัดการข้อมูลประกอบด้วยกลุ่มคำสั่ง SELECT, INSERT, UPDATE และ DELETE สำหรับภาษาควบคุมข้อมูล ซึ่งเกี่ยวข้องกับการควบคุมระบบความปลอดภัยภายในฐานข้อมูล ประกอบด้วยคำสั่ง GRANT และ REVOKE จะนำไปอธิบายในบทที่ 10 ในหัวข้อการกำหนดสิทธิ์การใช้งานของผู้ใช้

แบบฝึกหัดทบทวน

จากข้อมูลบางส่วนในฐานข้อมูลทะเบียนและวัดผลของวิทยาลัยแห่งหนึ่ง ประกอบด้วยข้อมูลที่บันทึกอยู่ในแต่ละตารางดังต่อไปนี้

STUDENT

stdCode	stdName	address	sex	facultyCode	majorCode	gpa
50420001	นงคราญ	กรุงเทพฯ	F	BA	ACC	3.15
50420002	สมควร	กรุงเทพฯ	M	IT	ICT	3.11
50420010	ศรีสมร	ปทุมธานี	F	IT	ICT	2.58
50420018	นิรุตน์	อยุธยา	M	BA	ACC	3.75
50420019	จงรัก	ลพบุรี	M	BA	MN	3.80
50420020	สมชาย	ระยอง	M	BA	MN	2.15
51410002	สมฤทัย	ชลบุรี	F	IT	ICT	2.00
51410004	มณีรัตน์	เชียงราย	F	BA	MN	3.05
51410005	ประภาคาร	เชียงใหม่	M	IT	ICT	2.85
51410006	ปิยะนุช	กรุงเทพฯ	F	IT	ITM	3.51
51410007	ลัดดา	ลำปาง	F	BA	MK	3.55
51410008	พงษ์ธร	ลำปาง	M	BA	MK	2.92
51410011	สัญญา	อุตรดิตย์	M	IT	ITM	3.68
51410012	ลิขิต	กรุงเทพฯ	M	BA	ACC	3.58
51410013	สมชาติ	ลำปาง	M	IT	ITM	3.04
51410021	อุมาพร	เชียงใหม่	F	BA	ACC	3.57

MAJOR

facultyCode	majorCode	majorName
BA	ACC	สาขาวิชาการบัญชี
BA	MK	สาขาวิชาการตลาด
BA	MN	สาชาวิชาการจัดการ
IT	ICT	สาชาวิชาเทคโนโลยีสารสนเทศและการสื่อสาร
IT	ITM	สาขาวิชาการจัดการระบบสารสนเทศ

FACULTY

facultyCode	facultyName
BA	คณะบริหารธุรกิจ
IT	คณะเทคโนโลยีสารสนเทศ

ITEM_TEACHING

lectCode	sbjCode	Section
001	B10000	1
001	B10001	1
005	IT0001	1
005	IT0001	2
006	IT0005	1
006	IT0009	1

LECTURER

lectCode	name	facultyCode	majorCode
001	อ.ฟ้าใส	BA	ACC
002	อ.งามแข	BA	ACC
005	อ.โอรส	IT	ICT
006	อ.ทรงศรี	IT	ITM

SUBJECT

sbjCode	sbjName	credit
B100000	การบัญชีเบื้องต้น	3
B100001	การบัญชีขั้นกลาง	3
B100002	พฤติกรรมองค์การ	3
B100003	เศรษฐศาสตร์เบื้องต้น	3
B100004	เศรษฐศาสตร์มหภาค	3
B100005	การตลาดและการบริการ	3
B100006	การจัดการช่องทางจำหน่าย	3
B100007	ระบบบริหารงานบุคคล	3
IT0001	เทคโนโลยีสารสนเทศเบื้องต้น	3
IT0002	การโปรแกรมคอมพิวเตอร์1	3
IT0003	การโปรแกรมคอมพิวเตอร์2	3
IT0004	ระบบฐานข้อมูล	3
IT0005	การวิเคราะห์และออกแบบระบบ	3
IT0006	เครือข่ายคอมพิวเตอร์	3
IT0007	โครงสร้างข้อมูล	3
IT0008	การบริหารศูนย์คอมพิวเตอร์	3
IT0009	ระบบสารสนเทศเพื่อการจัดการ	3

ITEM_REGISTERED

Semester	stdCode	sbjCode	Section
2551/2	50420001	B10000	1
2551/2	50420001	IT0009	1
2551/2	50420010	IT0001	1
2551/2	50420010	IT0009	1
2551/2	50420018	B10000	1
2551/2	50420018	IT0009	1
2551/2	50420019	B10002	1
2551/2	50420019	B10003	2
2551/2	50420019	B10007	1
2551/2	50420020	B10007	2
2551/2	51410002	IT0004	1
2551/2	51410002	IT0007	1
2551/2	51410007	B10005	1
2551/2	51410007	B10006	1
2551/2	51410007	IT0005	1
2551/2	51410010	IT0001	1
2551/2	51410010	IT0002	1
2551/2	51410011	IT0004	1
2551/2	51410011	IT0007	1
2551/2	51410012	B10000	1
2551/2	51410012	B10001	1
2551/2	51410012	IT0009	1
2551/2	54140013	IT0005	1
2551/2	54140013	IT0006	2
2551/2	54140013	IT0009	1

จงเขียนชุดคำสั่ง SQL ในการตอบโจทย์ต่อไปนี้ พร้อมแสดงผลลัพธ์

1. จงแสดงข้อมูลนักศึกษาทั้งหมด โดยจัดเรียงตามคณะและสาขา
2. ต้องการทราบว่าวิทยาลัยประกอบไปด้วยกี่คณะ กี่สาขาวิชา
3. จงแสดงยอดรวมนักศึกษาทั้งสิ้นในแต่ละคณะ

4. อยากทราบว่า ณ ขณะนี้นักศึกษาได้ลงทะเบียนเรียนวิชาใดบ้าง
5. ต้องการทราบยอดลงทะเบียนเรียนของแต่ละวิชา
6. ต้องการทราบรายชื่อนักศึกษาที่ยังไม่ได้ลงทะเบียนเรียน
7. แสดงรายวิชาที่สอนของอาจารย์แต่ละคน
8. ต้องการทราบว่านักศึกษาของวิทยาลัยมาจากจังหวัดใดบ้าง แต่ละจังหวัดมีนักศึกษาอยู่กี่คน
9. อยากทราบว่าวิทยาลัยมีนักศึกษาเพศชายกี่คน นักศึกษาเพศหญิงกี่คน
10. ต้องการทราบค่าเฉลี่ย GPA ของนักศึกษาทั้งหมด แยกตามคณะและสาขาต่างๆ รวมถึงยอดจำนวนนักศึกษาในแต่ละสาขาวิชา

เอกสารอ้างอิง

พนิดา พานิชกุล และ ณัฐพงษ์ วารีประเสริฐ. (2552). *การออกแบบ พัฒนา และดูแลระบบฐานข้อมูล* (พิมพ์ครั้งที่ 1). กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

Connolly, T., & Begg, C. (2005). *Database System: A Practical Approach to Design, Implementation, and Management* (9th ed.). England: Pearson Education.

Eric, R., & Jim, R. W. (2012). *Seven Databases in Seven Weeks : A Guide to Modern Databases and the NoSQL Movement.* USA: The Pragmatic Programmers.

Guy, H. (2015). *Next Generation Databases 2015 : NoSQL and Big Data*. Germany: aPress.

Louis, D., & Stacia, V. (2017). *Exam Ref 70-762 Developing SQL Databases.* USA: Microsoft Press, U.S.

Mohankumar, S., & Robert, C. (2015). *DB2 10.1/10.5 for Linux, Unix, & Windows Database Administration*. USA: MC Press, LLC.

Orin, T., Neil, H. B., T., & Peter, W. (2012). *Administering Microsoft SQL Server 2012 Databases: Training Kit (Exam 70-462)* (1st ed.). USA: Microsoft Press.