

Abstract geometric lines in the top-left corner of the page, consisting of several overlapping, irregular polygons and lines in black and dark gray.

SQL FUNDAMENTALS

Oracle Database 11g

By Yodesaya Timprom (Mon)

AGENDA

SQL language and What is PL/SQL?

- Introduction, Course Requirement & Prerequisite

Database and What is Database Management System (DBMS)

- Relational Database Management System (RDBMS)
- How to Create a Database and PL/SQL Elements
- Select, Delete a Database
- SQL Naming Convention

Data Types in SQL

- Data Types included Numeric, String and etc.

Tables in SQL (Data Definition Language (DDL))

- Create Tables (included PK,FK and GTT) and Delete a Table.
- Insert Data to Tables and Retrieve Data from Tables,

AGENDA

Data Definition Language (DDL)

- CREATE, ALTER and DROP AS DYNAMIC SQL
- Comments, NULL or NOT NULL, Constraint, IDENTITY, etc.

Data Query Language (DQL)

- SELECT Statement, DISTINCT, TOP Clause , WHERE Clause and JOIN TABLES

Data Manipulation Language (DML)

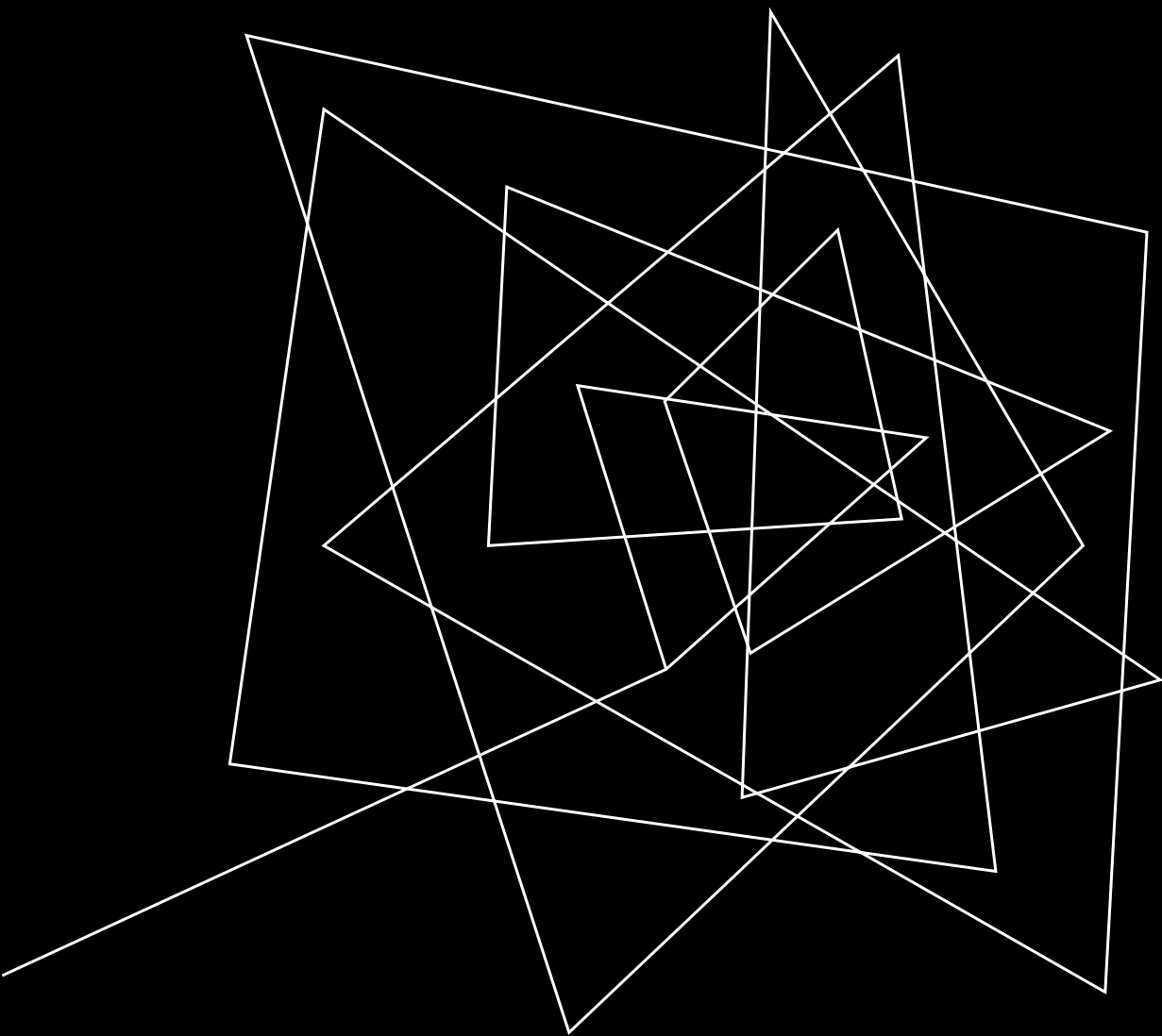
- UPDATE and DELETE command, Cursors

String Functions

- Such as LEFT, RIGHT, LEN, UPPER, LOWER, SUBSTRING and etc.

Sorting Records

- Order By, OFFSET & FETCH
- % Percentage Wildcard and Underscore Wildcard

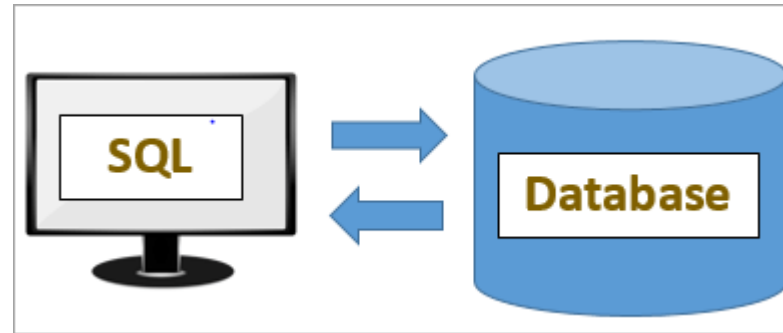


PART I

SQL LANGUAGE

SQL ย่อมาจาก **structured query language** คือภาษาที่ใช้ในการเขียนโปรแกรม เพื่อจัดการกับฐานข้อมูลโดยเฉพาะ เป็นภาษามาตรฐานบนระบบฐานข้อมูลเชิงสัมพันธ์ (RDBMS) และเป็นระบบเปิด (**open system**) หมายถึงเราสามารถใส่คำสั่ง **sql** กับฐานข้อมูลชนิดใดก็ได้ และ คำสั่งกับงานเดียวกันเมื่อสั่งงานผ่านระบบฐานข้อมูลที่แตกต่างกันจะได้ผลลัพธ์เหมือนกัน ทำให้เราสามารถเลือกใช้ฐานข้อมูล ชนิดใดก็ได้โดยไม่ติดขัดกับฐานข้อมูลใดฐานข้อมูลหนึ่ง โดยใช้มาตรฐานของภาษาเป็นรูปแบบ แอนซี (ANSI) และ ไอเอสโอ (ISO).

โดยภาษา **programming** ที่ออกแบบมาเพื่อทำการจัดการข้อมูล (**Manipulation**), ค้นหาข้อมูล (**Searching**), ปรับปรุง (**Improvement**), เปลี่ยนแปลง (**Adjustment**), เพิ่ม (**Adding/Insertion**) และ ลบ (**Deleting**) ข้อมูลนั่นเอง ซึ่งข้อมูลจะถูกเก็บอยู่ในฐานข้อมูลในรูปแบบตาราง (**Tables**) หรือ **Structured Data / Schema-based** ที่มีลักษณะเป็นคอลัมน์และแถว (**Columns and Rows**).



- ข้อดี
- 1) เป็นเทคโนโลยีที่มีการพัฒนาต่อเนื่องมานาน ทำให้มีความสามารถรอบด้าน โดยถูกออกแบบมาให้เป็น **General Purpose** รองรับการทำงานได้หลากหลาย
 - 2) สามารถทำงานร่วมกับ **Hardware** แบบเดิมๆ ได้ รวมถึงสามารถทำงานร่วมกับ **Application** และ **Programming Language** ได้หลากหลาย
 - 3) มีเครื่องมือสนับสนุนการใช้งานให้พร้อม ทั้งสำหรับการเสริมความทนทาน, การเพิ่มความปลอดภัย, การบริหารจัดการ และการดูแลรักษา

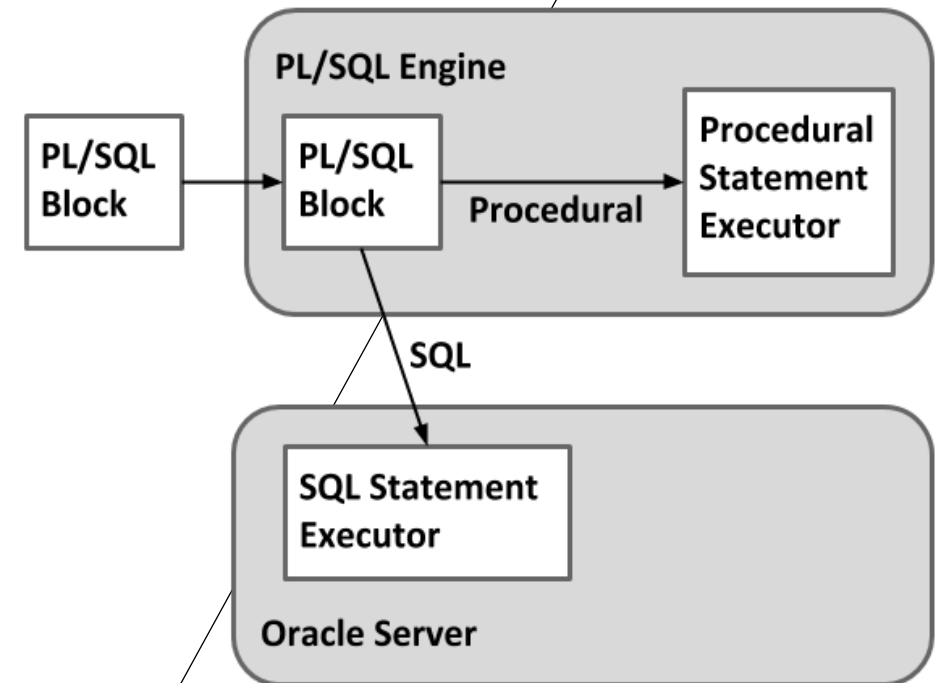
- ข้อเสีย
- 1) ส่วนใหญ่จะไม่สามารถทำ **Scale-out** ได้อย่างยืดหยุ่นเท่ากับเทคโนโลยีฐานข้อมูลอื่นๆ ทำให้การออกแบบ **SQL** สำหรับรองรับผู้ใช้งานจำนวนมากในระบบใหญ่ๆ นั้นถือว่าค่อนข้างยาก
 - 2) การที่ต้องระบุ **Schema** ชัดเจนก็ทำให้การเพิ่มฟิลด์ของข้อมูลนั้นทำได้ลำบาก (ถึงแม้จะใช้ **JSON** ได้แต่ก็ไม่ยืดหยุ่นเท่าการใช้ **NoSQL** เต็มๆ อยู่ดี)
 - 3) การถูกออกแบบมาเป็น **General Purpose** นั้นก็ทำให้มีประสิทธิภาพสู้กับพวก **NoSQL** หรือ **NewSQL** ในงานเฉพาะทางบางอย่างไม่ได้ รวมทั้งการทำ **Performance Tuning** นั้นต้องอาศัยความรู้เป็นอย่างมาก

WHAT IS PL/SQL

Oracle PL/SQL เป็นส่วนขยายของภาษา SQL ที่รวมพลังการจัดการข้อมูลของ SQL เข้ากับพลังการประมวลผลของ procedural language (ภาษาขั้นตอน) เพื่อสร้างการสืบค้น SQL ที่ทรงพลังเป็นพิเศษ PL/SQL ช่วยให้เรามั่นใจถึงการประมวลผลคำสั่ง SQL ที่ราบรื่นโดยการเพิ่มความปลอดภัย ความเบาสะดวก และความทนทานของฐานข้อมูล. เราสามารถแบ่งปัญหาที่ซับซ้อนออกเป็นโปรแกรมย่อยที่เข้าใจได้ง่าย ซึ่งเราสามารถใช้ซ้ำได้ในหลายแอปพลิเคชัน เป็นต้น

ความแตกต่างระหว่าง SQL และ PL/SQL

SQL	PL/SQL
SQL is a single query that is used to perform DML and DDL operations.	PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
It is declarative, that defines what need to be done, rather than how things need to be done.	PL/SQL is procedural that defines how the things needs to be done.
Execute as a single statement.	Execute as a whole block.
Mainly used to manipulate data.	Mainly used to create an application.
Interaction with a Database server.	No interaction with the database server.
Cannot contain PL/SQL code in it.	It is an extension of SQL, so that it can contain SQL inside it.

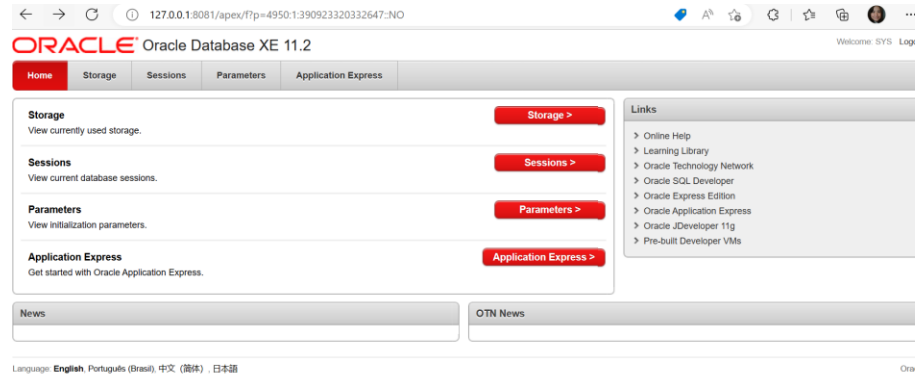


PL/SQL Architecture

SQL IN ORACLE (PL/SQL)

Course Requirement & Prerequisite

- Oracle Database XE 11.2D



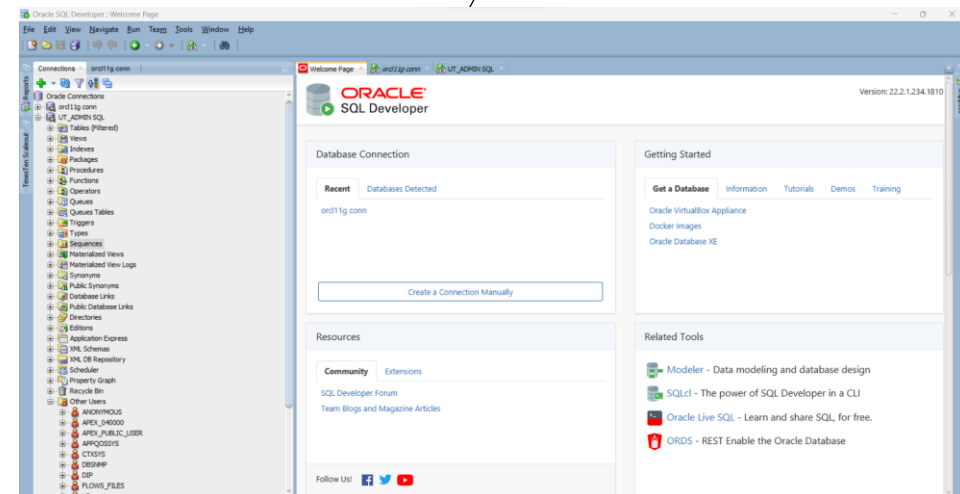
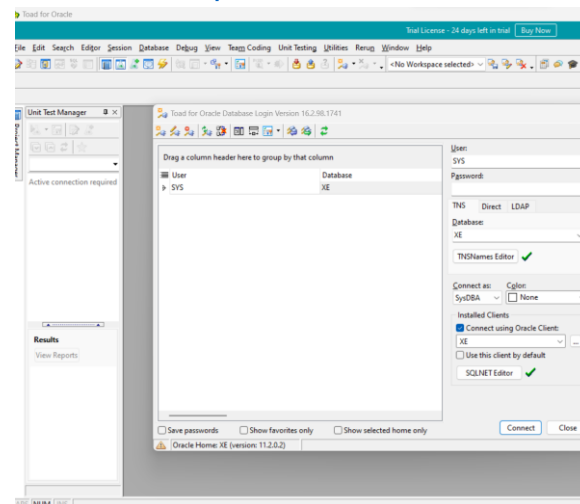
Login

Username

Password

Login as a database user which has been granted the DBA database role (for example, SYSTEM).

- Data Management Tools such as Toad for Oracle ([Best Oracle Developer and Administrator Database Tools | Free Trial \(quest.com\)](http://quest.com)) หรือ [Oracle SQL Developer](http://www.oracle.com/technetwork/developer-tools/sql-developer/index-007464.html)



ORACLE SIDS VS. ORACLE SERVICE NAMES

SID คือ Oracle SID เป็นชื่อเฉพาะที่ระบุอินสแตนซ์/ฐานข้อมูลของผู้ใช้งาน โดยเฉพาะ ในขณะที่ชื่อบริการก่อนนามแฝง TNS ที่ให้เมื่อเราเชื่อมต่อกับฐานข้อมูลของเราจากระยะไกล และชื่อบริการนี้จะถูกบันทึกไว้ในไฟล์ tnsnames.ora บนไคลเอนต์ของเรา และชื่อบริการนี้ สามารถเหมือนกับ SID และเรายังสามารถตั้งชื่ออื่น ๆ ที่เราต้องการได้อีกด้วย

Service name หรือ SERVICE_NAME เป็นคุณลักษณะใหม่ตั้งแต่ oracle 8i เป็นต้นไป ซึ่งฐานข้อมูลสามารถลงทะเบียนตัวเองกับ listener ได้ หากฐานข้อมูลลงทะเบียนด้วยวิธีนี้ เราสามารถใช้พารามิเตอร์ SERVICE_NAME ใน tnsnames.ora หรือใช้ SID ใน tnsnames.ora ได้

ตัว SERVICE_NAMES มันจะต้องระบุชื่ออย่างน้อยหนึ่งชื่อสำหรับบริการฐานข้อมูลที่อินสแตนซ์นั้นเชื่อมต่อ เราสามารถระบุชื่อบริการได้หลายชื่อเพื่อแยกความแตกต่างระหว่างการใช้ฐานข้อมูลเดียวกัน

วิธีการค้นหา "SID" และ "Service Name" ใน oracle

1) Get Oracle "SERVICE NAMES" through SQL commands เชื่อมต่อกับเซิร์ฟเวอร์เป็น "system" โดยใช้ SID:

```
select value from v$parameter where name like '%service_name%';
```

หรือ

```
show parameter service_name;
```

หรือ

```
select sys_context('userenv','service_name') from dual;
```

2) Get Oracle Instance Name

```
SELECT sys_context('userenv','instance_name') FROM dual;
```

3) Get Oracle SID

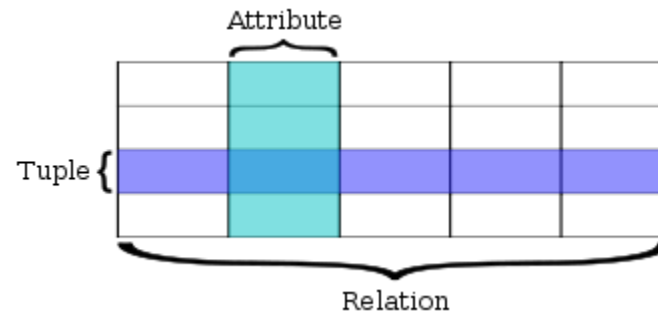
```
SELECT sys_context('USERENV', 'SID') FROM DUAL;
```

4) Get Oracle Database Name

```
select ora_database_name from dual;
```


RDBMS (RELATIONAL DATABASE MANAGEMENT SYSTEM)

เป็นระบบจัดการฐานข้อมูลที่นิยมใช้กันมานานแล้วตั้งแต่ยุค 1970 มีความเสถียรมากเหมาะสำหรับการเก็บข้อมูลที่มีจุดประสงค์และแยกประเภทชัดเจน ใช้ภาษา SQL ในการ Query และ Maintain Database มีการเก็บข้อมูลในรูปแบบ Tables (ตาราง) มีองค์ประกอบเป็น Rows และ Columns (มองภาพคล้าย ๆ ตารางของ Microsoft Excel) สาเหตุที่เก็บข้อมูลเป็น Tables (ตาราง) เนื่องจากช่วยให้ง่ายต่อการเข้าถึงข้อมูลของกันและกัน หรือเรียกอีกชื่อว่า Relation ก็ได้รับ เนื่องจากเป็นการนำข้อมูลมาเชื่อมต่อกัน มีความสัมพันธ์กัน



องค์ประกอบของ Tables จะประกอบไปด้วย

1. Row (แถว หรือ แนวนอน) เรียกอีกชื่อว่า **Tuple** คือ ข้อมูล
2. Column (สดมภ์ หรือ แนวตั้ง) เรียกอีกชื่อว่า **Attribute** คือ การระบุชนิดของข้อมูลนั้น ๆ เช่น ที่อยู่, วัน เดือน ปีเกิด
3. Table (ตาราง) เรียกอีกชื่อว่า **Relation** คือ ชุดของข้อมูล (Record) ที่แบ่งชนิด (Attribute) เรียบร้อยแล้ว (Rows & Columns)
4. View เรียกอีกชื่อว่า **Query** คือ การรายงานข้อมูลจาก RDBMS โดยจะเรียกดูจาก Record จาก Row ใดก็ได้

โดยในการเก็บข้อมูล สังเกตได้ว่าการวางโครงสร้างเป็นลักษณะแบบตาราง ข้อมูลใหม่ที่เพิ่มเข้ามาก็จะไปสร้าง Row ขึ้นมาใหม่ต่อทางด้านล่าง ซึ่งการเพิ่มเข้ามาของข้อมูลจะมีลักษณะเป็นแบบการเพิ่มบรรทัดนั่นเอง

การเก็บข้อมูลของ RDBMS จึงมีความเป็นระเบียบมาก สามารถหาข้อมูลที่เกี่ยวข้องกันได้ง่ายเนื่องจากข้อมูลมีการผูกกันแบบชัดเจน แต่ตารางที่ว่าจะถูก Fix Column มาให้แล้ว ทำให้เพิ่มข้อมูลได้เฉพาะเท่าที่มี Fields อยู่

ที่นี้ผู้คิดค้นก็มีความกังวลว่าในการใช้งานหากมีคนใส่ข้อมูลที่ไม่ตรงกับความต้องการที่แท้จริงของ Field นั้น ๆ และเวลานำข้อมูลของแต่ละ Table มาเชื่อมต่อกัน (JOINS) จึงต้องมีการกำหนด **Constraints** ขึ้นมา

COMPARISON BETWEEN SQL VS NOSQL

สรุปความแตกต่างระหว่าง RDBMS กับ NoSQL

	RDBMS	NoSQL
ชนิด	Relational	Non-Relational
รูปแบบของชุดข้อมูล	เป็นแบบโครงสร้างที่เก็บอยู่ใน Table	ไม่เป็นโครงสร้าง เก็บในรูปแบบ JSON (Text) หรือแบบอื่น ๆ
การ Scale	Vertical (เพิ่ม Spec Server)	Horizontal (เพิ่มจำนวน Server)
Schema	เปลี่ยนแปลงไม่ได้	เปลี่ยนแปลงได้ ค่อนข้างยืดหยุ่น
ตัวอย่าง Brand ในตลาด	Oracle, MySQL, Microsoft SQL Server, PostgreSQL	MongoDB, CouchDB, Redis, DynamoDB, Cassandra, HBase, Neo4j, Neptune

ข้อดีและข้อเสียของ SQL และ NoSQL

ใช้ RDBMS ในกรณีที่	ใช้ NoSQL ในกรณีที่
Workload คงที่ คาดเดาได้ และต้องการพื้นที่ปานกลางถึงมาก	มีปริมาณ Workload มหาศาลที่ต้อง Scale ได้เยอะ
ข้อมูลมีรูปแบบที่คาดเดาได้ มีโครงสร้างชัดเจน	ข้อมูลเป็นแบบ Dynamic มีการเปลี่ยนแปลงบ่อย
ข้อมูลต้องมี Relation เชื่อมถึงกันและกัน	ข้อมูลไม่จำเป็นต้อง Relation กันแบบซับซ้อน
ข้อมูลที่ยืนยันต้องเป็นไปตาม Condition	ต้องการเขียนไว้ ๆ Condition สำคัญรองลงมา ไม่ได้เน้นมาก
ข้อมูลค่อนข้าง Complex ต้องมีการ Query และ Report ได้	ข้อมูลค่อนข้าง Simple
ต้องการให้มี User ในการควบคุม	ต้องการให้ข้อมูลกระจายให้เข้าถึงกันได้ทุกส่วนใน Environment
จะ Deploy ใส่ Hardware ขนาดใหญ่ หรือของตัวเอง (On-Premise)	จะ Deploy บน Cloud (On-Cloud)

PL/SQL ELEMENTS ...

ด้วยประสิทธิภาพที่ PL/SQL ช่วยส่งชุดคำสั่ง (block of statements) ไปยังฐานข้อมูล ซึ่งช่วยลดการรับส่งข้อมูลระหว่างแอปพลิเคชันและฐานข้อมูลได้อย่างมาก มาดู Basic Syntax Of PL SQL นั้นประกอบด้วย

- 1. Declaration:** This section begins with the DECLARE keyword. It is not considered as the required one and has variables, subprograms, and so on.
- 2. Executable Commands:** This section begins with BEGIN and END keywords respectively. It is considered a required one and contains PL/SQL statements. It consists of at least one executable line of code.
- 3. Exception Handling:** This section begins with the keyword EXCEPTION. It comprises the types of exceptions that the code will handle.
- 4. Begin:** This is the keyword used for pointing to the execution block. It is required in a PL/SQL code where actual business logic is described.
- 5. End:** This is the keyword used to determine the end of the block of code.

Main Features of PL/SQL มีดังนี้

- [Error Handling](#)
- [Blocks](#)
- [Variables and Constants](#)
- [Subprograms](#)
- [Packages](#)
- [Triggers](#)
- [Input and Output](#)
- [Data Abstraction](#)
- [Control Statements](#)
- [Conditional Compilation](#)
- [Processing a Query Result Set One Row at a Time](#)

```
1 DECLARE
2     msg varchar (40) := 'Software Testing Help - PL/SQL series';
3 BEGIN
4     dbms_output.put_line(msg);
5 END;
6 /
7
```

Results	Explain	Describe	Saved SQL	History
Software Testing Help - PL/SQL series				
Statement processed.				
0.00 seconds				

SELECT, DELETE A DATABASE

การ select database ใน Oracle database เราสามารถใช้ command เพื่อทำการระบุ database ที่เราต้องการจะเลือกใช้ หรือใช้ในการสลับเลือกใช้ database ไปยังอันอื่นๆด้วย

Syntax ใช้ดังนี้

```
USE [DATABASE];
```

ตัวอย่างการใช้งาน

```
USE ut_admin_sql;
```

หรือจะเป็นการเลือกและสลับ database ใน database management tool อย่างเช่น Toad, Oracle SQL developer

Noted : คู่มือในส่วนของการใช้งาน [Connect to Oracle Database Server using SQL*Plus](#)

การ delete database ใน Oracle database (DROP DATABASE) เพื่อลบฐานข้อมูลเป้าหมายก่อนดำเนินการใดๆ ต้องทราบตำแหน่งของไฟล์ข้อมูล ไฟล์ควบคุม และไฟล์บันทึกการทำซ้ำแบบออนไลน์ (datafile, controlfile, and online redo log files) ใช้แบบสอบถามตามนี้ก่อนเพื่อจะได้รับรายละเอียดของที่อยู่ database file จาก command ดังนี้

```
select name from v$database; <<--- check database name
```

```
select name from v$datafile;
```

```
select name from v$controlfile;
```

```
select member from v$logfile;
```

จากนั้นสั่ง Shutdown the database และเริ่มฐานข้อมูลในโหมด Exclusive จนถึงขั้นตอนเเมนต์เพื่อเตรียมเริ่มกระบวนการด้วย startup mount exclusive restrict; ก่อน drop ฐานข้อมูล oracle ในตอนท้าย drop database;

หรือจะทำขั้นตอนเหล่านี้ผ่านทาง database management tool อย่างเช่น Toad, Oracle SQL developer ก็ทำได้ง่ายด้วยเช่นเดียวกันซึ่งในขณะที่จะ drop database ต้องไม่หลงเหลือ session การใช้งานใดๆกับ database นั้นเสียก่อนจึงจะลบได้สำเร็จ

SQL NAMING CONVENTION

การตั้งชื่อแบบแผนใน PL/SQL

ในโปรแกรมที่ซับซ้อนเราอาจต้องใช้ **identifiers** จำนวนมาก รวมถึงตัวแปร เคอร์เซอร์ ด้วยเพื่อหลีกเลี่ยงความสับสนและเพื่อเพิ่มความสามารถในการอ่านของโปรแกรม เราจึงต้องปฏิบัติตามหลักการตั้งชื่อให้เหมาะสม หลักการตั้งชื่อที่ใช้กันทั่วไปใน PL/SQL: ควรใช้อักษรตัวแรกเพื่อระบุ **declared level** ยกตัวอย่างของอักษรตัวแรกที่ใช้กันดังนี้

'P' – ตัวแปรถูกประกาศที่ระดับพารามิเตอร์

'L' – ประกาศตัวแปรที่ local block

'G' – ตัวแปรถูกประกาศใน global level

ตัวอักษรตัวที่สองระบุประเภทของ **identifier** ด้านล่างนี้คือประเภทของ **identifier** ที่ใช้กันทั่วไปและ **naming code** ดังนี้

- 'C' – Cursor Identifier
- 'V' – Varchar and char datatype
- 'N' – Number datatype
- 'R' – Record type
- 'T' – Table type

ด้านล่างนี้เป็นตัวอย่างบางส่วนของหลักการตั้งชื่อที่เหมาะสมใน PL/SQL:

Lv_name – local level variable of varchar/char datatype

Pc_num – parameter level cursor identifier

Gn_user_id – Global level variable of numerical data type

DATA TYPES IN SQL

ทุกค่าคงที่ของ PL/SQL, ตัวแปร, พารามิเตอร์ และ function return value ต้องมีชนิดข้อมูลที่กำหนดรูปแบบการจัดเก็บและค่าตัวแปรที่ถูกต้อง แบ่งเป็น 4 กลุ่มหลักและที่ใช้เป็นประจำเป็นประเภท SCALAR และ LOB

S.No	Category & Description
1	Scalar Single values with no internal components, such as a NUMBER , DATE , or BOOLEAN .
2	Large Object (LOB) Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.
3	Composite Data items that have internal components that can be accessed individually. For example, collections and records.
4	Reference Pointers to other data items.

ประเภทข้อมูล SCALAR ของ PL/SQL และประเภทย่อยย่อยภายใต้หมวดหมู่ต่อไปนี้:

S.No	Date Type & Description
1	Numeric Numeric values on which arithmetic operations are performed.
2	Character Alphanumeric values that represent single characters or strings of characters.
3	Boolean Logical values on which logical operations are performed.
4	Datetime Dates and times.

TABLES IN SQL .. CREATE TABLES AND PRIMARY KEY (DDL)

ตัว syntax ที่ใช้สร้างคำสั่งสำหรับการ CREATE TABLE ใน Oracle/PLSQL คือ:

```
CREATE TABLE table_name  
(  
    column1 datatype [ NULL | NOT NULL ],  
    column2 datatype [ NULL | NOT NULL ],  
    ...  
    column_n datatype [ NULL | NOT NULL ]  
);
```

ตัวอย่างเช่น :

```
CREATE TABLE customers  
( customer_id number(10) NOT NULL,  
  customer_name varchar2(50) NOT NULL,  
  city varchar2(50)  
);
```

```
CREATE TABLE customers  
( customer_id number(10) NOT NULL,  
  customer_name varchar2(50) NOT NULL,  
  city varchar2(50),  
  CONSTRAINT customers_pk PRIMARY KEY  
  (customer_id)  
);
```

ในการสร้าง Primary Keys ตัว syntax ที่ใช้สร้างคำสั่ง คือ:

```
ALTER TABLE table_name  
ADD CONSTRAINT constraint_name PRIMARY KEY (column1, column2, ... column_n);
```

ในการ Drop Primary Key ตัว syntax ที่ใช้สร้างคำสั่ง คือ:

```
ALTER TABLE table_name  
DROP CONSTRAINT constraint_name;
```

TABLES IN SQL .. CREATE GLOBAL TEMPORARY TABLES (DDL)

ตัว syntax ที่ใช้สร้างคำสั่งสำหรับการ Global Temp Table (GTT) หรือ DECLARE GLOBAL TEMPORARY TABLE statement ใน Oracle/PLSQL คือ:

ตัวอย่างเช่น :

```
DECLARE GLOBAL TEMPORARY TABLE table-Name  
  { column-definition [ , column-definition ] * }  
  [ ON COMMIT {DELETE | PRESERVE} ROWS ]  
  NOT LOGGED [ON ROLLBACK DELETE ROWS]
```

```
CREATE GLOBAL TEMPORARY TABLE table-Name  
( column_id number(10) NOT NULL,  
  column_name varchar2(50) NOT NULL,  
  .....  
) ON COMMIT DELETE ROWS  
  ON COMMIT PRESERVE ROWS;
```

Global Temporary Tables including Undo and Redo

แม้ว่าข้อมูลใน GTT จะถูกเขียนลงในพื้นที่ตารางชั่วคราว (temporary tablespace) แต่การเลิกทำที่เกี่ยวข้องยังคงเขียนลงในพื้นที่เลิกทำตารางตามปกติ (normal undo tablespace) ซึ่งได้รับการป้องกันด้วยการทำซ้ำ ดังนั้นการใช้ GTT จะไม่ลดการเลิกทำและการทำซ้ำ (undo and redo) ที่เกี่ยวข้องกับการป้องกันการเลิกทำพื้นที่ตาราง (undo tablespace)

```
CREATE GLOBAL TEMPORARY TABLE my_temp_table (  
  id NUMBER,  
  description VARCHAR2(20)  
) ON COMMIT PRESERVE ROWS;
```

```
-- Insert and commit, then check contents of GTT.  
INSERT INTO my_temp_table VALUES (1, 'ONE');  
COMMIT;
```

```
DROP TABLE my_temp_table PURGE;
```

เพิ่มเติม [Global Temporary Tables](#)

TABLES IN SQL .. FOREIGN KEY (DDL)

ตัว syntax ที่ใช้สร้างคำสั่งสำหรับการ Foreign Key ใน Oracle/PLSQL คือ:

[Foreign Keys](#)

[Foreign Keys with cascade delete](#)

[Foreign Keys with "set null on delete"](#)

ตัวอย่างเช่น :

```
CREATE TABLE products (  
  product_id numeric(10) not null,  
  supplier_id numeric(10) not null,  
  CONSTRAINT fk_supplier  
    FOREIGN KEY (supplier_id)  
    REFERENCES supplier(supplier_id)  
);
```

```
CREATE TABLE products(  
  product_id numeric(10) not null,  
  supplier_id numeric(10) not null,  
  supplier_name varchar2(50) not null,  
  CONSTRAINT fk_supplier_comp  
    FOREIGN KEY (supplier_id, supplier_name)  
    REFERENCES supplier(supplier_id, supplier_name)  
);
```

ในการ Drop a foreign key ตัว syntax ที่ใช้สร้างคำสั่ง คือ:

```
ALTER TABLE table_name
```

```
DROP CONSTRAINT constraint_name;
```

Disable/Enable Foreign Keys

[Disable a foreign key](#)

[Enable a foreign key](#)

```
CREATE TABLE table_name  
(  
  column1 datatype null/not null,  
  column2 datatype null/not null,  
  ...  
  CONSTRAINT fk_column  
    FOREIGN KEY (column1, column2, ... column_n)  
    REFERENCES parent_table (column1, column2, ... column_n)  
);
```

TABLES IN SQL .. DELETE/DROP A TABLE (DDL)

ในการ CREATE TABLE ใน oracle plsql มีด้วยกันหลายประเภท

Oracle alternative to SQL CREATE TABLE IF NOT EXISTS

Create a new table from another table using CREATE TABLE AS SELECT รูปแบบ: CREATE TABLE table_name AS (SELECT select_query);

Create and insert data into a temporary table

Syntax :

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
(startdate DATE,
enddate DATE,
operation CHAR(20))
ON COMMIT DELETE ROWS;
```

Oracle External Table ตารางภายนอกคือตารางที่มีข้อมูลมาจากไฟล์แฟลต (flat files) ที่เก็บไว้บนฐานข้อมูล สามารถแยกวิเคราะห์ไฟล์รูปแบบใดก็ได้ที่ SQL*Loader รองรับ ตารางภายนอกมีประโยชน์ในกระบวนการ ETL ของคลังข้อมูล เนื่องจากข้อมูลไม่จำเป็นต้องทำ staged table (หรือ data stage) และสามารถสืบค้นแบบ parallel ได้ (สามารถสืบค้นได้ไปพร้อมๆกันด้วยนั่นเอง)

ในการสร้าง DROP TABLE Statement ตัว syntax ที่ใช้สร้างคำสั่ง คือ:

```
DROP TABLE [schema_name].table_name
[ CASCADE CONSTRAINTS ]
[ PURGE ];
```

ตัวอย่างเช่น :

```
DROP TABLE brands;
```

```
DROP TABLE brands CASCADE CONSTRAINTS;
```

```
DROP TABLE cars purge;
```

เพิ่มเติม การลบ column จาก Table แบบหลาย Column Syntax ดังนี้

```
ALTER TABLE table_name
DROP (column_name1, column_name2,...);
```

เพิ่มเติม [Drop Tables](#)

TABLES IN SQL .. INSERT DATA TO TABLES, RETRIEVE DATA FROM TABLES

The INSERT Statement Syntax ดังนี้ :

```
INSERT INTO table_name (list_of_columns)
VALUES (list_of_values);
```

Noted : การ INSERT ยังมีวิธีเพิ่มเติมคือ [INSERT ALL](#) , [INSERT INTO SELECT](#) , [SELECT INTO](#)

The UPDATE Statement

```
UPDATE table_name
SET column_name = value [, column_name = value]...
[ WHERE condition ];
```

Noted : Transaction Control Statements จะใช้ SAVEPOINT, COMMIT และ ROLLBACK

Triggers

ทริกเกอร์คือหน่วย PL/SQL ที่จัดเก็บไว้ในฐานข้อมูล และ (หากอยู่ในสถานะเปิดใช้งาน) จะดำเนินการโดยอัตโนมัติ ("เริ่มทำงาน หรือ fires") เพื่อตอบสนองต่อเหตุการณ์ที่ระบุ หากต้องการสร้างทริกเกอร์ ให้ใช้อินเทอร์เฟซแบบกราฟิกของ SQL Developer หรือคำสั่ง DDL CREATE TRIGGER และ หากต้องการเปลี่ยนทริกเกอร์ ให้ใช้เครื่องมือแก้ไขของ SQL Developer หรือคำสั่ง DDL CREATE TRIGGER ด้วยคำสั่งย่อย OR REPLACE clause

การ **Retrieve Data from Tables** (เพิ่มเติม [Selecting Table Data](#))

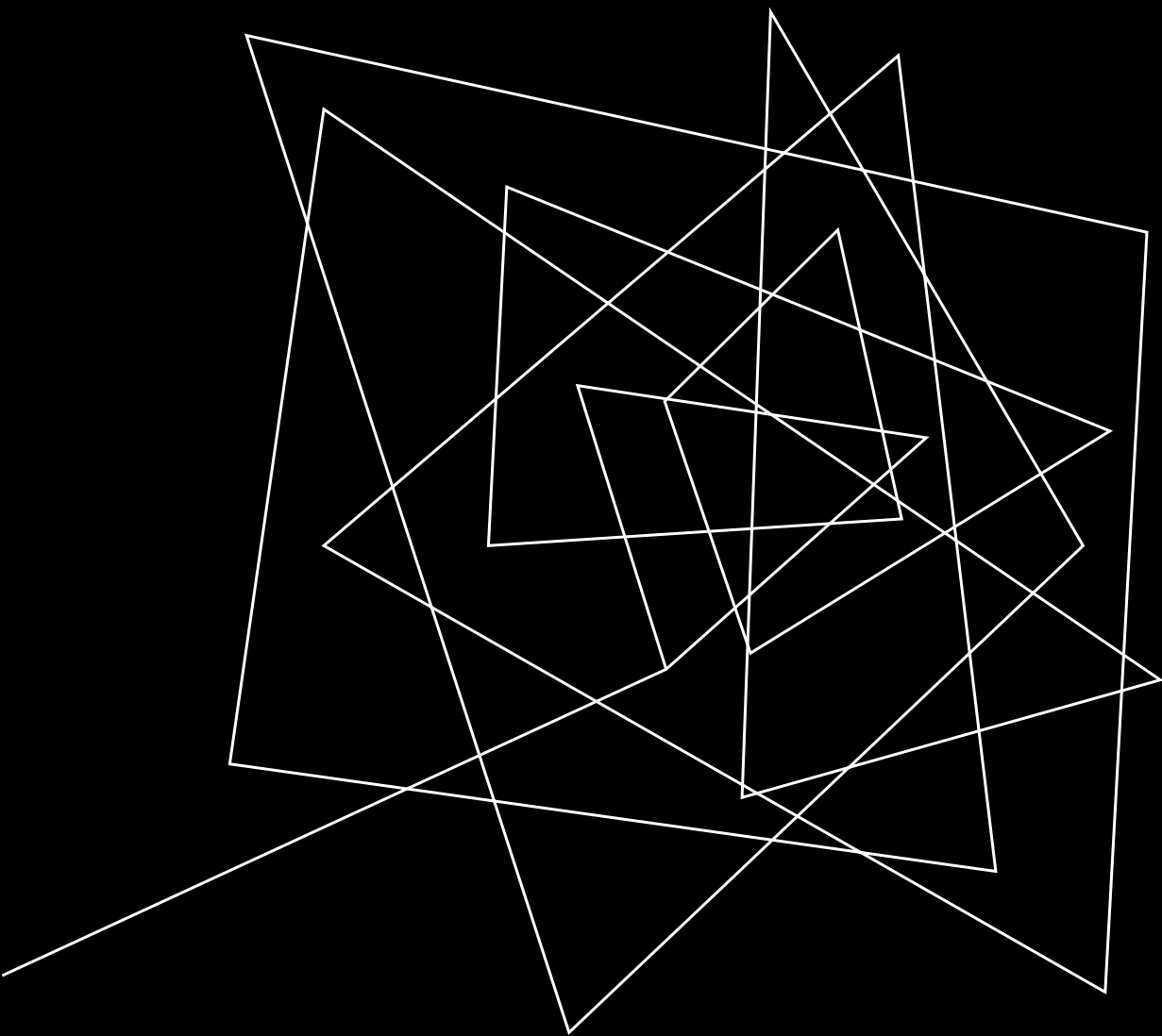
Selecting Data that Satisfies Specified Conditions, Specifying Conditions with Regular Expressions

Selecting Data from Multiple Tables (Joining Multiple Tables)

Using Operators and Functions in Queries, Using the Concatenation Operator in Queries, Using Datetime Functions in Queries

Using Conversion Functions in Queries, Using Aggregate Functions in Queries, Using CASE Expressions in Queries

Using the DECODE Function in Queries



PART II

DDL.. CREATE, ALTER AND DROP AS DYNAMIC SQL

ด้วยความสามารถของ Native Dynamic SQL เราสามารถดำเนินการคำสั่ง DDL และ DML ในบล็อก PL/SQL ได้ การดำเนินการคำสั่งนิยามข้อมูล SQL (เช่น CREATE) คำสั่งควบคุมข้อมูล (เช่น GRANT) หรือ คำสั่งควบคุมเซสชัน (เช่น ALTER SESSION) ใน PL/SQL คำสั่งดังกล่าวไม่สามารถดำเนินการแบบคงที่ได้และต้องการความยืดหยุ่นมากขึ้นยกตัวอย่างเช่น เมื่อต้องการเลื่อนตัวเลือกของออบเจกต์ schema ออกไปจนกว่าจะถึงเวลาทำงาน หรือ เมื่อต้องการให้โปรแกรมของเราสร้างเงื่อนไขการค้นหาค้นหาที่แตกต่างกันสำหรับส่วนคำสั่ง WHERE ในส่วนคำสั่ง SELECT โปรแกรมที่ซับซ้อนกว่าอาจเลือกจาก various SQL operations, clauses เป็นต้น เราจะใช้แพ็คเกจ DBMS_SQL เพื่อดำเนินการคำสั่ง SQL แบบไดนามิกก็ได้แต่ประสิทธิภาพไม่ดีเท่า Native Dynamic SQL เนื่องจาก DBMS_SQL ไม่ support objects และ collections.

วิธีการสร้าง Dynamic DDL แบบ Native Dynamic SQL

- 1) เตรียมคำสั่ง DDL โดยการทดสอบ compile statement ที่ต้องการเอาไว้ให้เรียบร้อย
- 2) เอามาเขียนเป็น เขียนโปรแกรม Native Dynamic SQL

```
CREATE OR REPLACE
PROCEDURE DynamicCursor (p_parameter IN VARCHAR2) IS
  TYPE cur_typ IS REF CURSOR;
  c_cursor cur_typ;
  l_query VARCHAR2(1000);
  l_text VARCHAR2(100);
BEGIN

  IF Length(p_parameter) > 10 THEN
    l_query := 'SELECT "The parameter (" || :parameter || ") is too long" AS text FROM dual';
  ELSE
    l_query := 'SELECT first_name || " " || last_name FROM users WHERE user_type =
:parameter';
  END IF;

  OPEN c_cursor FOR l_query USING p_parameter;
  LOOP
    FETCH c_cursor INTO l_text;
    EXIT WHEN c_cursor%NOTFOUND;
    -- process row here
  END LOOP;
  CLOSE c_cursor;
END;
```

```
ALTER TABLE tut_83 ADD tut_date DATE;
```

```
DROP TABLE tut_83
```

```
SET SERVEROUTPUT ON;
DECLARE
ddl_qry VARCHAR2(50);
BEGIN
ddl_qry := 'ALTER TABLE tut_83
            ADD tut_date DATE';
EXECUTE IMMEDIATE ddl_qry;
END;
```

```
SET SERVEROUTPUT ON;
DECLARE
ddl_qry VARCHAR2 (100);
BEGIN
ddl_qry := 'DROP TABLE tut_83';
EXECUTE IMMEDIATE ddl_qry;
END;
```

Dynamic Cursors

บางครั้งเราอาจไม่ทราบ definition ของ cursor จนกว่าจะรันไทม์อย่างเช่นการเข้าถึง object ที่ไม่มีอยู่ในปัจจุบันควร compile ในสคิมาใด ๆ เนื่องจากไม่มีการอ้างอิงโดยตรงไปยัง database objects จนกว่าจะรันไทม์

DROP COLUMN AND NATIVE DYNAMIC SQL VS DBMS_SQL

DROP Column commands มีด้วยกัน 2 แบบคือแบบ physical delete (ลบออกถาวร) และ logical delete ซึ่งจะตั้งค่าคอลัมน์นั้นว่าไม่ได้ใช้ และเราสามารถลบได้ในภายหลัง มาดูแบบ logical กันการใช้งานดังนี้

To set a column as unused syntax: `ALTER TABLE table_name SET UNUSED COLUMN column_name;`

หลังจากดำเนินการคำสั่งนี้ คอลัมน์จะไม่สามารถเข้าถึงได้อีกต่อไป ดังนั้นจะไม่ปรากฏในมุมมองข้อมูลหรือผลการสืบค้น ข้อจำกัดและดัชนีทั้งหมดสำหรับคอลัมน์นั้นจะถูกลบออกด้วย แต่คอลัมน์นั้นยังคงจัดเก็บไว้ในไครฟ์จริง สิ่งนี้มี

ประโยชน์อย่างหนึ่งเมื่อเราเริ่รับและไม่ต้องการรอคิวรี DROP ที่ยาวนานให้เสร็จสิ้น หากต้องการลบคอลัมน์ทั้งหมดและเพิ่มพื้นที่เก็บข้อมูลจริง ให้รันคำสั่งนี้: `ALTER TABLE table_name DROP UNUSED COLUMNS;` จากตัวอย่างเช่น `ALTER TABLE employees DROP COLUMN salary;` จะสามารถใช้ดังตารางได้

```
ALTER TABLE employees
SET UNUSED COLUMN salary;
```

```
ALTER TABLE table_name
DROP UNUSED COLUMNS;
```

Native Dynamic SQL

- Easy to use and concise.
- PL/SQL interpreter has built in support for Native Dynamic SQL so it is more efficient than DBMS_SQL.
- Supports user defined types.
- Supports FETCH INTO record types
- Not supported in client site code.
- Does not support DESCRIBE_COLUMNS
- Does not support bulk Dynamic SQL, but it can be faked by placing all statements in a PL/SQL block.
- Only supports Single row Updates/Deletes with RETURNING clause.
- Does not support SQL statements bigger than 32K
- Parse required for every execution

DBMS_SQL

- Often long-winded and awkward.
- DBMS_SQL uses a Procedural API so it is generally slower than Native Dynamic SQL.
- Does not support user defined types.
- Does not support FETCH INTO record types
- Supported in client side code.
- Supports DESCRIBE_COLUMNS
- Supports bulk Dynamic SQL.
- Supports Single and Multiple row Updates/Deletes with RETURNING clause.
- Does support SQL statements bigger than 32K
- Parse once, execute many possible

COMMENTS, NULL OR NOT NULL, CONSTRAINT, IDENTITY

COMMENTS

รูปแบบการใช้ **comment** สำหรับ 1 บรรทัดเราสามารถใช้ **double hyphen (- -)** ที่ใดก็ได้ในบรรทัดและขยายไปยังจุดสิ้นสุดของบรรทัด และเมื่อต้องการใช้หลายบรรทัดใช้ขึ้นต้นด้วยเครื่องหมายทับ (**/***) และลงท้ายด้วยเครื่องหมาย (***/**) สามารถขยายได้หลายบรรทัด เช่น -- UPDATE dept SET loc = my_loc WHERE deptno = my_deptno;

NULL or NOT NULL

เราสามารถใช้มันได้กับ **Condition** ที่เป็น SQL statement หรือใน block of PLSQL code และ **Operator** ตัวอย่างใน condition :

```
DELETE FROM customers WHERE status IS NOT NULL; หรือ SELECT * FROM orders WHERE salesman_id IS NULL ORDER BY order_date DESC;
```

นอกจากนี้ใน oracle ยังมี [NULL-Related Functions](#) ประกอบด้วย NVL, DECODE, NVL2, COALESCE, NULLIF, LNNVL, NANVL เป็นต้น

CONSTRAINT clause

เป็นข้อจำกัดคือกฎสำหรับตารางและคอลัมน์ของตารางที่จำกัดวิธีการและข้อมูลที่จะสามารถแทรก (INSERT) อัปเดต (UPDATE) หรือลบ (DELETE) ได้ อย่างเช่น **NOT NULL constraint** หมายความว่าคอลัมน์ต้องมีค่า โดยต้องไม่เป็น **unknown** หรือ **blank**. อีกประเภทคือ **UNIQUE** ใช้เพื่อระบุว่าค่าในคอลัมน์ต้องไม่ซ้ำกัน ทั้งยังรวมถึงการตั้งการกำหนด PK และ FK นอกจากนี้ยังสามารถใช้ **CHECK** เพื่อใช้ระบุกฎสำหรับค่าในคอลัมน์ ดูเพิ่มเติมการใช้ [Check Constraints](#) กับ Drop/Enable/Disable a Check Constraint

```
CREATE TABLE suppliers (  
  supplier_id numeric(4),  
  supplier_name varchar2(50),  
  CONSTRAINT check_supplier_name CHECK (supplier_name = upper(supplier_name)) );
```

IDENTITY

เป็นคุณสมบัติที่เพิ่มเติมเข้ามาเริ่มตั้งแต่ **Oracle 12c** นำเสนอวิธีใหม่ที่เรากำหนดคอลัมน์เอกลักษณ์สำหรับตาราง ซึ่งคล้ายกับคอลัมน์ **AUTO_INCREMENT** ใน MySQL หรือคอลัมน์ **IDENTITY** ใน SQL Server โดยคอลัมน์เอกลักษณ์มีประโยชน์มากสำหรับคอลัมน์หลักตัวแทน เมื่อเราแทรกแถวใหม่ในคอลัมน์เอกลักษณ์ Oracle จะสร้างและแทรกค่าตามลำดับลงในคอลัมน์โดยอัตโนมัติ Syntax ดังนี้:

IDENTITY [(seed , increment)]

ตัวอย่างการใช้งาน :

ดูเพิ่มเติม [Identity Columns in Oracle Database 12c Release 1 \(12.1\)](#)

```
CREATE TABLE EMP  
(  
  id_number int IDENTITY(1,1),  
  nick_name varchar (20)  
)
```

```
INSERT EMP VALUES ('Sachin')  
  select * from EMP  
INSERT EMP VALUES ('Amit')  
  select * from EMP
```

DATA QUERY LANGUAGE (DQL)

การใช้คำสั่ง SELECT หรือ subquery เพื่อดึงข้อมูลจากตาราง (table), object tables, views, object views, materialized views, analytic views หรือ hierarchies

การใช้ with_clause เพื่อกำหนด :

PL/SQL procedures และ functions (ใช้คำสั่ง plsql_declarations)

Subquery blocks (โดยใช้ subquery_factoring_clause หรือ subav_factoring_clause หรือทั้งสองอย่าง)

นอกจากนี้ยังมีการใช้ search_clause, cycle_clause, subav_factoring_clause, subav_clause, hierarchies_clause, filter_clauses, attr_dim_alias, add_calcs_clause, calc_meas_clause เป็นต้น เพิ่มเติมดูที่ [SELECT](#)

DISTINCT | UNIQUE การระบุสิ่งนี้ เมื่อต้องการให้ฐานข้อมูลส่งคืนสำเนาเพียงชุดเดียวของแถวที่ซ้ำกันแต่ละชุดที่เลือก คำหลักทั้งสองนี้มีความหมายเหมือนกัน แถวที่ซ้ำกันคือแถวที่มีค่าตรงกันสำหรับแต่ละนิพจน์ในรายการที่เลือก

ALL การระบุสิ่งนี้ เมื่อต้องการให้ฐานข้อมูลส่งคืนแถวทั้งหมดที่เลือก รวมถึงสำเนาทั้งหมดของรายการที่ซ้ำกัน คำเริ่มต้นคือทั้งหมด

FROM Clause ใช้ระบุวัตถุที่เลือกจากข้อมูล

pivot_clause การระบุสิ่งนี้ เมื่อต้องการให้เขียนแบบสอบถามข้ามตารางที่หมุนแถวเป็นคอลัมน์, aggregating data ใน process of the rotation ผลลัพธ์ของการดำเนินการ Pivot โดยทั่วไปจะมีจำนวนคอลัมน์และแถวน้อยกว่าชุดข้อมูลเริ่มต้น (และยังมี unpivot_clause ด้วยเช่นกัน)

join_clause การระบุสิ่งนี้ เพื่อระบุตารางที่เป็นส่วนหนึ่งของการรวมที่จะเลือกข้อมูล inner_cross_join_clause ให้เราระบุการรวมภายในหรือข้ามได้ ส่วน outer_join_clause เป็นการให้เราระบุการรวมภายนอก และ cross_outer_apply_clause ให้เราระบุรูปแบบของ ANSI CROSS JOIN หรือ ANSI LEFT OUTER JOIN ที่มีการสนับสนุนความสัมพันธ์ด้านซ้าย

group_by_clause การระบุสิ่งนี้ เพื่อต้องการให้ฐานข้อมูลจัดกลุ่มแถวที่เลือกตามค่าของ expr(s) สำหรับแต่ละแถว และส่งคืนข้อมูลสรุปแถวเดียวสำหรับแต่ละกลุ่ม หากส่วนคำสั่งนี้มีส่วนขยาย CUBE หรือ ROLLUP ฐานข้อมูลจะสร้างการจัดกลุ่มแบบ superaggregate นอกเหนือจากการจัดกลุ่มแบบปกติ

Expressions in the GROUP BY clause สามารถมีคอลัมน์ใดๆ ของตาราง views หรือแม้แต่ materialized views ในส่วนคำสั่ง FROM โดยไม่ต้องคำนึงว่าคอลัมน์จะปรากฏในรายการที่เลือกหรือไม่

ส่วน GROUP BY clause จะจัดกลุ่มแถว แต่ไม่ได้รับประกันลำดับของชุดผลลัพธ์ หากต้องการจัดกลุ่มให้ใช้คำสั่ง ORDER BY ร่วมด้วย

WHERE Clause ใช้เพื่อกรองผลลัพธ์จากคำสั่ง SELECT, INSERT, UPDATE หรือ DELETE นั่นเอง ดูเพิ่มเติม : [Oracle / PLSQL: WHERE Clause](#)

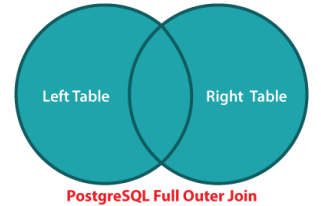
PLSQL : JOIN TABLE

การเชื่อมแบบ FULL JOIN : ทำการรวมแบบเต็มของ **2** Tables จะรวมผลลัพธ์ของการรวมด้านซ้ายและการรวมด้านขวาทั้งหมด ถ้าแถวในตารางที่รวมไม่ตรงกันผลที่ได้มันจะ set ค่า NULL สำหรับทุกคอลัมน์ของตารางที่ไม่มีแถวที่ตรงกัน ถ้าแถวจากตารางหนึ่งตรงกับแถวในอีกตารางหนึ่ง แถวผลลัพธ์จะมีคอลัมน์ที่สร้างจากคอลัมน์ของแถวจากทั้งสองตาราง

ตัวอย่างเช่น : `SELECT * FROM A FULL [OUTER] JOIN B on A.id = B.id;` --(OUTER keyword is optional)

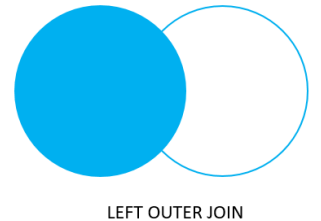
Noted: เป็นการ JOIN แบบที่เอา data ของทั้ง **2** Tables มาโชว์ที่ผลลัพธ์ทั้งหมดไม่มีข้อยกเว้น

และในบาง select query หากเราใช้ LEFT JOIN UNION RIGHT JOIN แบบไม่มี Where clauses จะได้ค่าเท่าๆกันกับ FULL JOIN ได้เช่นกัน



การเชื่อมแบบ LEFT OUTER JOIN : เป็นผลลัพธ์รวมโดยเลือกข้อมูลจากตารางด้านซ้ายหากเท่ากันกับตารางด้านขวา สำหรับแต่ละแถวในตารางด้านซ้ายตามเงื่อนไข โดยเติมคอลัมน์ที่มาจากตารางด้านขวาด้วย NULL

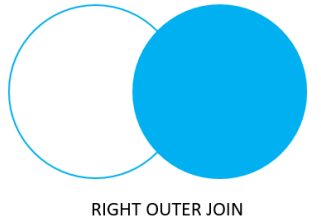
ตัวอย่างเช่น : `SELECT f.film_id, title, inventory_id FROM film f LEFT JOIN inventory i USING (film_id)`
`WHERE i.film_id IS NULL ORDER BY title;`



การเชื่อมแบบ RIGHT OUTER JOIN : เลือกข้อมูลจากตารางด้านขวาหากเท่ากันกับตารางด้านซ้าย ประกอบด้วยคอลัมน์จากทั้งสองตารางและรวมแถวใหม่ในชุดผลลัพธ์และเติมคอลัมน์จากตารางด้านซ้ายด้วย NULL (จะเลือกแถวทั้งหมดจากตารางด้านขวาไม่ว่าจะมีแถวที่ตรงกันจากตารางด้านซ้ายหรือไม่)

ตัวอย่างเช่น : `SELECT review, title FROM films RIGHT JOIN film_reviews using (film_id) WHERE title IS NULL;`

Noted: ให้ระวังการนำตัวเลขมากระทำทางคณิตศาสตร์ด้วยเครื่องหมาย +, -, / หรืออื่นๆกับค่า NULL ในผลลัพธ์กับตารางจะได้ค่าออกมาเป็น NULL ควรใช้ Coalesce() function มาช่วยในการแปลง NULL เป็น **0** ก่อนนำไปคำนวณ



การเชื่อมภายในตารางเดียวกัน (SELF JOIN หรือ INNER JOIN) : เป็นการรวมปกติที่รวมตารางเข้ากับตัวเองมักจะใช้เพื่อสืบค้นข้อมูลแบบลำดับชั้นหรือเพื่อเปรียบเทียบแถวภายในตารางเดียวกันโดยจะต้องระบุตารางเดียวกัน **2** ครั้งโดยใช้ table aliases ที่แตกต่างกันใน query (การทำตารางเสมือนด้วยตารางเดิมเป็นตารางใหม่เพื่อ JOIN กัน)

ตัวอย่างเช่น : `SELECT e.first_name || ' ' || e.last_name employee, m.first_name || ' ' || m.last_name manager FROM employee e`
`INNER JOIN employee m ON m.employee_id = e.manager_id ORDER BY manager;`

PLSQL : JOIN TABLE (cont.)

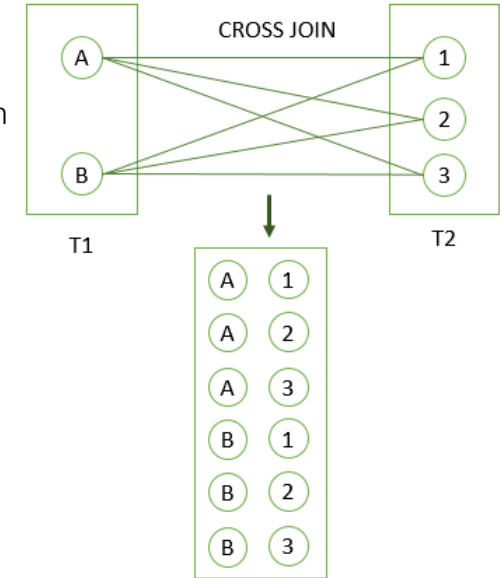
การเชื่อมแบบการทำงานของ CROSS JOIN : เป็นการเชื่อมแบบไขว้มองแบบ Table ว่าเป็น **2** set นำมาทำได้เป็นผลลัพธ์แบบผลคูณ Cartesian แตกต่างจากส่วนคำสั่งการรวมอื่นๆ เช่น LEFT JOIN, RIGHT JOIN หรือ INNER JOIN เพราะ CROSS JOIN ไม่มี join predicate (คือ ON table1.field_key = table2.field_key)

รูปแบบ : `SELECT column-lists FROM Table1 CROSS JOIN Table2;`

หรือ `SELECT [column_list]* FROM Table1, Table2;`

หรือ `SELECT * FROM Table1 INNER JOIN Table2 ON true;`

ตัวอย่างเช่น : `SELECT * FROM "EMPLOYMENT" CROSS JOIN "DEPARTMENT";`
`Select (*) from T1 CROSS JOIN T2`



Noted: เราสามารถประยุกต์ใช้ CROSS JOIN ในการข้อมูลแบบการจัดการแข่งขันพบกันหมด (round-robin tournament) ได้ด้วย

ข้อควรระวังในการใช้ CROSS JOIN กับตารางที่มี JOIN อื่นอย่าง INNER JOIN รวมอยู่ใน **1** ชุด query ตัวอย่างเช่น เงื่อนไข FROM T1 CROSS JOIN T2 INNER JOIN T3 ON ไม่เหมือนกับเงื่อนไข FROM T1, T2 INNER JOIN T3 ON เนื่องจากเงื่อนไขสามารถอ้างอิง T1 ในกรณีแรกได้ แต่ไม่ใช่กรณีที่สอง เป็นต้น

DATA MANIPULATION LANGUAGE (DML)

เราสามารถจัดการข้อมูลในฐานข้อมูลได้โดยใช้คำสั่ง DML โดยสามารถออกคำสั่ง DML เช่น INSERT, UPDATE, DELETE และ MERGE โดยไม่มีข้อจำกัดใน PL/SQL การล๊อคแถว (และการล๊อคตาราง) ถูกนำออกโดยการรวมคำสั่ง COMMIT หรือ ROLLBACK ไว้ในโค้ด PL/SQL

- The INSERT statement adds new rows to the table.
- The UPDATE statement modifies existing rows in the table.
- The DELETE statement removes rows from the table.
- The MERGE statement selects rows from one table to update or insert into another table. The decision whether to update or insert into the target table is based on a condition in the ON clause.
- ROLLBACK Restores the database to the condition before the changes were made
- COMMIT Permanently saves the changes

Noted: MERGE เป็นคำสั่งที่กำหนดขึ้น นั่นคือ เราไม่สามารถอัปเดตแถวเดียวกันของตารางเป้าหมายหลายครั้งในคำสั่ง MERGE เดียวกันได้ เราต้องมีสิทธิ์ INSERT และ UPDATE object ในตารางเป้าหมายและสิทธิ์ SELECT บนตารางต้นฉบับ

เมื่อเราส่งคำสั่ง SQL INSERT, UPDATE, DELETE, MERGE หรือ SELECT โดยตรงในโค้ด PL/SQL คอมไพเลอร์ PL/SQL จะเปลี่ยนตัวแปรในส่วนคำสั่ง WHERE และ VALUES เป็น bind variables ([Static SQL Statements](#)) โดย Oracle Database สามารถใช้คำสั่ง SQL เหล่านี้ซ้ำได้ทุกครั้งที่เราใช้โค้ดเดียวกัน ซึ่งโดยปกติ PL/SQL จะไม่สร้าง bind variables ในกรณีที่เรานำมาใช้ Dynamic SQL ซึ่งเป็นวิธีการเขียนโปรแกรมสำหรับการสร้างและรันคำสั่ง SQL ณ เวลาทำงานนั้นๆ ใช้กับการเขียนโปรแกรมที่ต้องเรียกใช้คำสั่งภาษานิยามฐานข้อมูล (DDL) หรือเมื่อไม่ทราบเวลาที่เรานำคอมไพล์ทั้งหมดของคำสั่ง SQL หรือจำนวน หรือประเภทข้อมูลของตัวแปรอินพุตและเอาต์พุต ([Dynamic SQL](#))

คอร์เซอร์ใน PL/SQL Oracle สร้างพื้นที่หน่วยความจำที่เรียกว่า context area สำหรับการประมวลผลคำสั่ง SQL ซึ่งมีข้อมูลทั้งหมดที่จำเป็นสำหรับการประมวลผลคำสั่ง ตัวอย่างเช่น จำนวนแถวที่ประมวลผล เป็นต้น คอร์เซอร์เป็นตัวชี้ไปยังพื้นที่บริบท PL/SQL ควบคุมพื้นที่บริบทผ่านคอร์เซอร์เก็บแถว (หนึ่งแถวขึ้นไป) ที่ส่งคืนโดยคำสั่ง SQL ชุดของแถวที่คอร์เซอร์ค้างไว้เรียกว่าชุดที่ใช้งานอยู่ เราสามารถดึงข้อมูลคอร์เซอร์เพื่อให้สามารถอ้างอิงในโปรแกรมเพื่อดึงข้อมูลและประมวลผลแถวที่ส่งคืนโดยคำสั่ง SQL ได้ทีละแถว คอร์เซอร์มีสองประเภท ได้แก่ : (ดูเพิ่มเติม [Working with cursors and dynamic queries in PL/SQL](#))

- Implicit cursors
- Explicit cursors

STRING FUNCTIONS

unction	Example	Result	Purpose
ASCII	ASCII('A')	65	Returns an ASCII code value of a character.
CHR	CHR('65')	'A'	Converts a numeric value to its corresponding ASCII character.
CONCAT	CONCAT('A','BC')	'ABC'	Concatenate two strings and return the combined string.
CONVERT	CONVERT('Ä Ê Í', 'US7ASCII', 'WE8ISO8859P1')	'A E I'	Convert a character string from one character set to another.
DUMP	DUMP('A')	Typ=96 Len=1: 65	Return a string value (VARCHAR2) that includes the datatype code, length measured in bytes, and internal representation of a specified expression.
INITCAP	INITCAP('hi there')	'Hi There'	Converts the first character in each word in a specified string to uppercase and the rest to lowercase.
INSTR	INSTR('This is a playlist', 'is')	3	Search for a substring and return the location of the substring in a string
LENGTH	LENGTH('ABC')	3	Return the number of characters (or length) of a specified string
LOWER	LOWER('Abc')	'abc'	Return a string with all characters converted to lowercase.
LPAD	LPAD('ABC',5,'*')	'**ABC'	Return a string that is left-padded with the specified characters to a certain length.
LTRIM	LTRIM(' ABC ')	'ABC '	Remove spaces or other specified characters in a set from the left end of a string.
REGEXP_COUNT	REGEXP_COUNT('1 2 3 abc','\d')	3	Return the number of times a pattern occurs in a string.
REGEXP_INSTR	REGEXP_INSTR('Y2K problem','\d+')	2	Return the position of a pattern in a string.
REGEXP_LIKE	REGEXP_LIKE('Year of 2017','\d+')	true	Match a string based on a regular expression pattern.
REGEXP_REPLACE	REGEXP_REPLACE('Year of 2017','\d+', 'Dragon')	'Year of Dragon'	Replace substring in a string by a new substring using a regular expression.
REGEXP_SUBSTR	REGEXP_SUBSTR('Number 10', '\d+')	10	Extract substrings from a string using a pattern of a regular expression.
REPLACE	REPLACE('JACK AND JOND','J','BL');	'BLACK AND BLOND'	Replace all occurrences of a substring by another substring in a string.
RPAD	RPAD('ABC',5,'*')	'ABC**'	Return a string that is right-padded with the specified characters to a certain length.
RTRIM	RTRIM(' ABC ')	' ABC'	Remove all spaces or specified character in a set from the right end of a string.
SOUNDEX	SOUNDEX('sea')	'S000'	Return a phonetic representation of a specified string.
SUBSTR	SUBSTR('Oracle Substring', 1, 6)	'Oracle'	Extract a substring from a string.
TRANSLATE	TRANSLATE('12345', '143', 'bx')	'b2x5'	Replace all occurrences of characters by other characters in a string.
TRIM	TRIM(' ABC ')	'ABC'	Remove the space character or other specified characters either from the start or end of a string.
UPPER	UPPER('Abc')	'ABC'	Convert all characters in a specified string to uppercase.

SORTING RECORDS AND OFFSET & FETCH

ใน Oracle ตารางจะจัดเก็บแถวในลำดับที่ไม่ได้ระบุ โดยไม่คำนึงถึงลำดับที่แถวถูกแทรกลงในฐานข้อมูล ในการสืบค้นแถวในลำดับจากน้อยไปหามากหรือจากมากไปน้อยตามคอลัมน์ เราต้องแจ้ง Oracle Database อย่างชัดเจนว่าต้องการทำเช่นนั้น ตัวอย่างเช่น เราอาจต้องการแสดงรายชื่อลูกค้าทั้งหมดตามชื่อของพวกเขาตามตัวอักษร หรือแสดงลูกค้าทั้งหมดตามลำดับวงเงินสินเชื่อต่ำสุดถึงสูงสุด ต้องการจัดเรียงข้อมูล ให้เพิ่มคำสั่ง ORDER BY ลงในคำสั่ง SELECT Syntax ดังนี้:

```
SELECT
    column_1, column_2, column_3, ...
FROM table_name
ORDER BY
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST],
    column_1 [ASC | DESC] [NULLS FIRST | NULLS LAST], ...
```

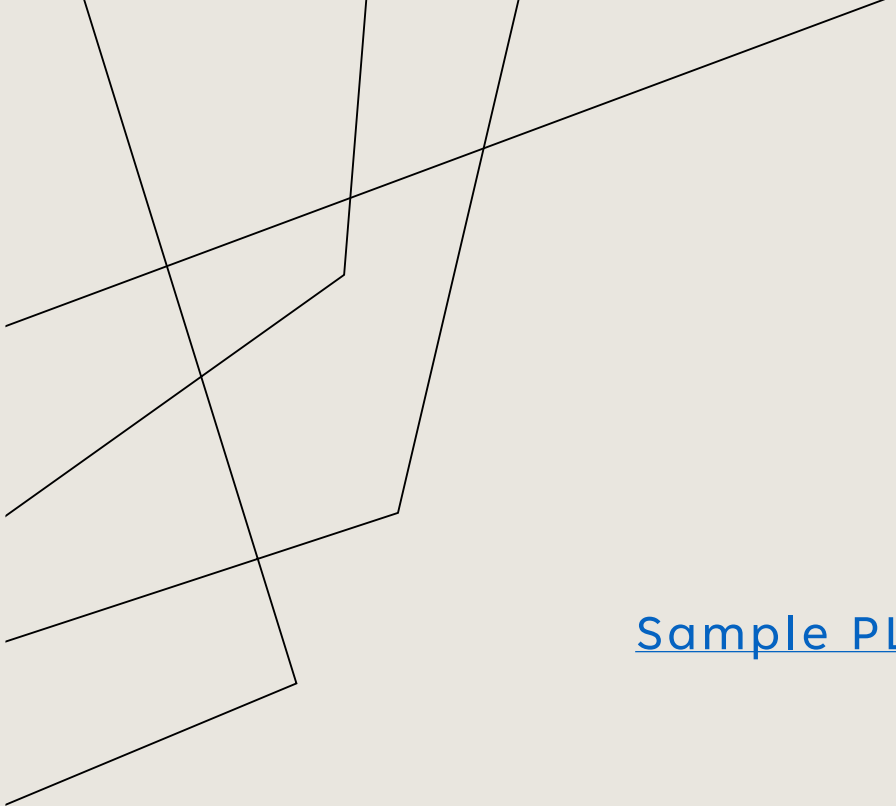
ตามค่าเริ่มต้น ส่วนคำสั่ง ORDER BY จะเรียงลำดับแถวจากน้อยไปมาก ไม่ว่าจะระบุ ASC หรือไม่ก็ตาม (default) ถ้าต้องการเรียงลำดับแถวจากมากไปหาน้อยเราใช้ DESC อย่างชัดเจน NULLS FIRST วางค่า NULL ก่อนค่าที่ไม่ใช่ NULL และ NULLS LAST ไล่ค่า NULL หลังค่าที่ไม่ใช่ NULL และ คำสั่ง ORDER BY จะเป็นประโยชน์สุดท้ายในคำสั่ง SELECT เสมอ

OFFSET ส่วนคำสั่ง OFFSET ใช้เพื่อระบุจำนวน N ของผลลัพธ์การค้นหาลำดับที่ควรข้ามไป (ไม่ส่งคืนไปยังแอปพลิเคชัน) N จำนวน โดยนิพจน์ที่อาจเป็นจำนวนเต็มเดียวตามตัวอักษร หรือตัวแปรภายนอกตัวเดียว หรือนิพจน์ใดๆ ที่สร้างจากตัวแปรภายนอกและส่งคืนจำนวนเต็มเดียวที่ไม่เป็นลบ Syntax of Offset & Fetch:

```
ORDER BY order_by_expression
    [ COLLATE collation_name ]
    [ ASC | DESC ]
    [ ,...n ]
[ < offset_fetch > ]

<offset_fetch> ::=
{ OFFSET { integer_constant | offset_row_count_expression } { ROW | ROWS }
  [ FETCH { FIRST | NEXT } { integer_constant | fetch_row_count_expression } { ROW | ROWS } ONLY
  ]
}
```

ส่วนของ [Wildcard Characters](#) และ [Oracle / PLSQL: LIKE Condition](#)



[Sample PL/SQL Programs](#)

[Overview of PL/SQL](#)

THANK YOU

Get to know SQL, NoSQL, and NewSQL, three alternatives to today's database technologies >>

<https://www.techtalkthai.com/introduce-sql-nosql-and-newsql-as-choices-of-database-technology/>

Oracle 11g Articles >>

<https://oracle-base.com/articles/11g/articles-11g>

Oracle Database Online Documentation 11g Release 2 (11.2) Database Administration >>

https://docs.oracle.com/cd/E11882_01/nav/portal_4.htm

Email : howtotailscompany@contoso.com

Github : <https://github.com/howtotailscompany/Oracle-PL-SQL>