

SQL Introduction

SQL 簡介與使用





Summary

- ◆ SQL 全名是結構化查詢語言 (Structured Query Language) ，是用於資料庫中的標準數據查詢語言。
- ◆ SQL 是目前關聯式資料庫系統所使用查詢語法的標準，使用者可以應用 SQL 語法對資料庫系統進行資料的存取、編輯、刪除及管理…等動作。





Outline

- ◆ 認識SQL語法
- ◆ 定義資料庫物件語法
- ◆ 查詢資料庫資料的內容
- ◆ MySQL常用函式
- ◆ 新增、更新與刪除資料
- ◆ 多資料表關聯查詢



SQL 簡介

- ◆ SQL，可以唸字母 S. Q. L. 或 sequel，結構化查詢語言，是用來建立、操作及存取關聯式資料庫 (Relational Database) 的語言。對於 SQL 的標準化作業，主要是由 ANSI 與 ISO 這兩個組織所推動。
- ◆ SQL 是目前關聯式資料庫系統所使用查詢語法的標準，使用者可以應用 SQL 語法對資料庫系統進行資料的存取、編輯、刪除及管理…等動作。
- ◆ SQL 語法的內容是利用簡單的英文語句所構成。不僅在應用上簡單，判讀維護上也相當容易。





認識SQL語法 - 功能關鍵字與指令

- ◆ SQL 語法是由一系列的指令所組成的，最後採用一個「;」分號來結尾，其中可以任意斷行。
- ◆ 每個指令中會包含一些功能關鍵字，如 SELECT、FROM、WHERE、UPDATE 或 DELETE 等，它們在 SQL 語法中都有其代表的功能或是特殊的意義，而且是不區分英文大小寫的。
- ◆ 除了關鍵字，指令中也會包含符號、引號所包圍的字串或是變數，甚至是特殊用途的符號。
- ◆ 每個單字間以空白分隔，也可以使用分行來斷句。



認識SQL語法 - 命名的保留字

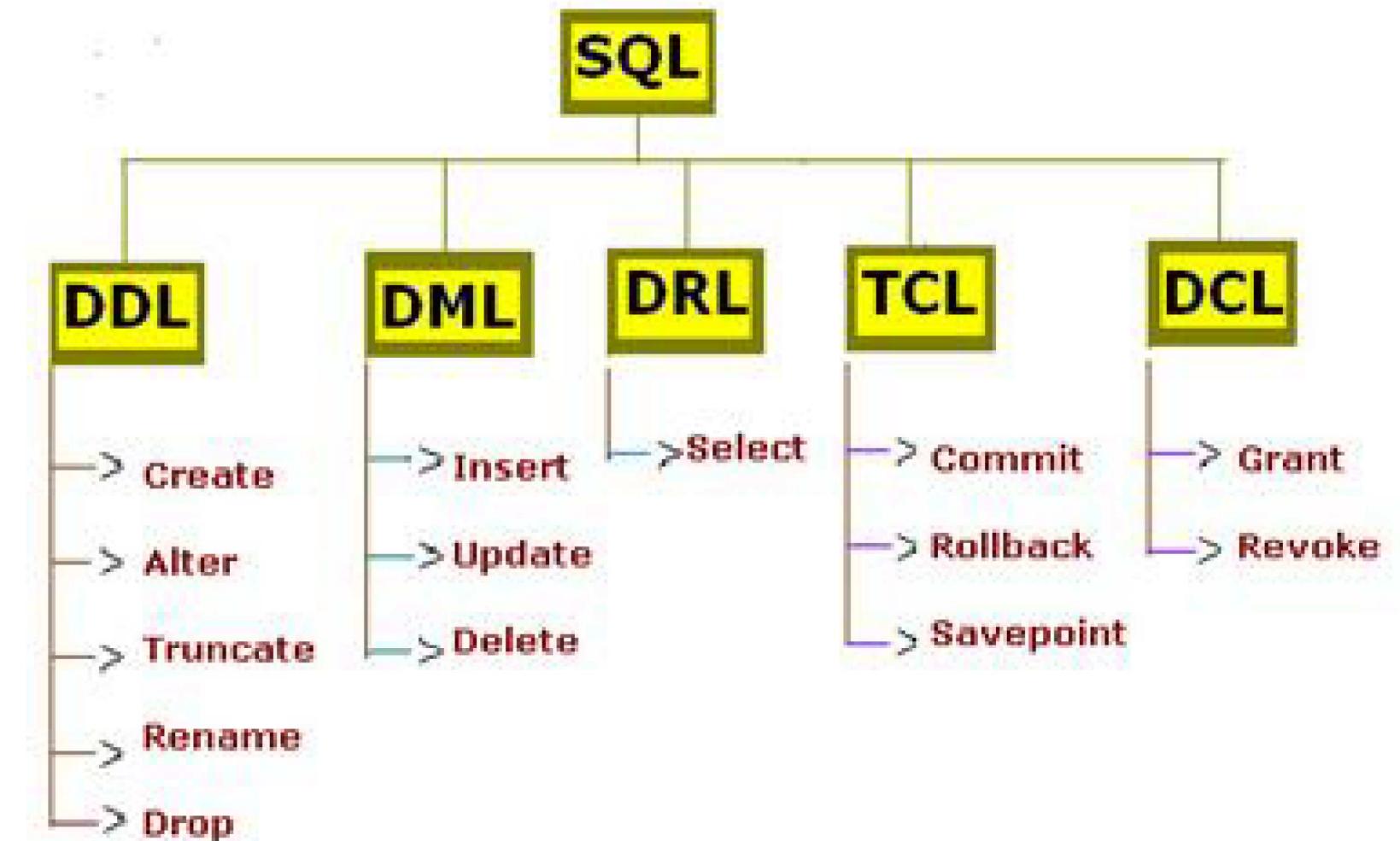
- ◆ 在 SQL 語法中也有保留字的顧慮，舉凡語法的關鍵字、SQL 函式名稱、資料型別的名稱因為都有其代表的意義，所以不能直接做為資料庫、資料表、欄位…等項目的名稱，以免造成程式在運作時的混淆。
- ◆ 但是實務上還是很容易因為對於保留字不熟悉而造成命名上的錯誤，所以建議您在 SQL 指令中只要使用到資料表欄位名稱時，可以利用「`」將名稱括住，就可輕易解決這個問題。



SQL 語法分類

◆SQL 語法可大致分為幾個類別，以下分別說明。

1. Data definition 資料定義 (DDL)
2. Data manipulation 資料處理 (DML)
3. Data Queries 資料查詢 (DQL)
4. Data control 資料控制 (DCL)
5. Transaction controls 交易控制 (TCL)

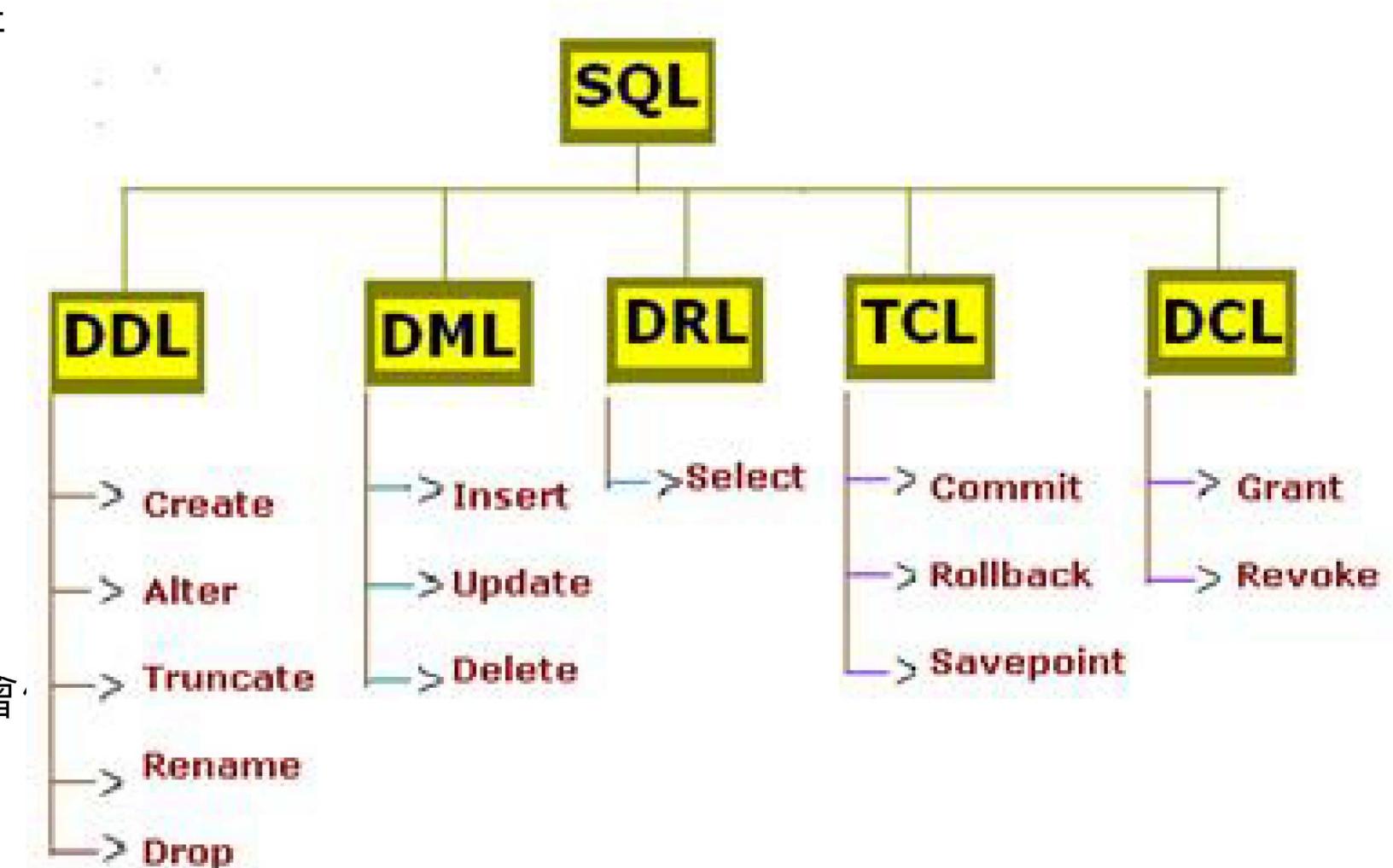


SQL 語法分類 - DDL

1. Data Definition Language 資料定義語言 (DDL)

- ◆ 用來定義資料庫、資料表、檢視表、索引、預存程序、觸發程序、函數等資料庫物件。常見的指令有：
- CREATE：主要為建立資料庫、資料表的物件
- ALTER：主要為變更資料庫、資料表的物件
- DROP：主要為刪除資料庫、資料表的物件
- Rename：更改資料庫或資料表的名字
- Truncate：移除資料表中的所有資料列，但會

資料表結構

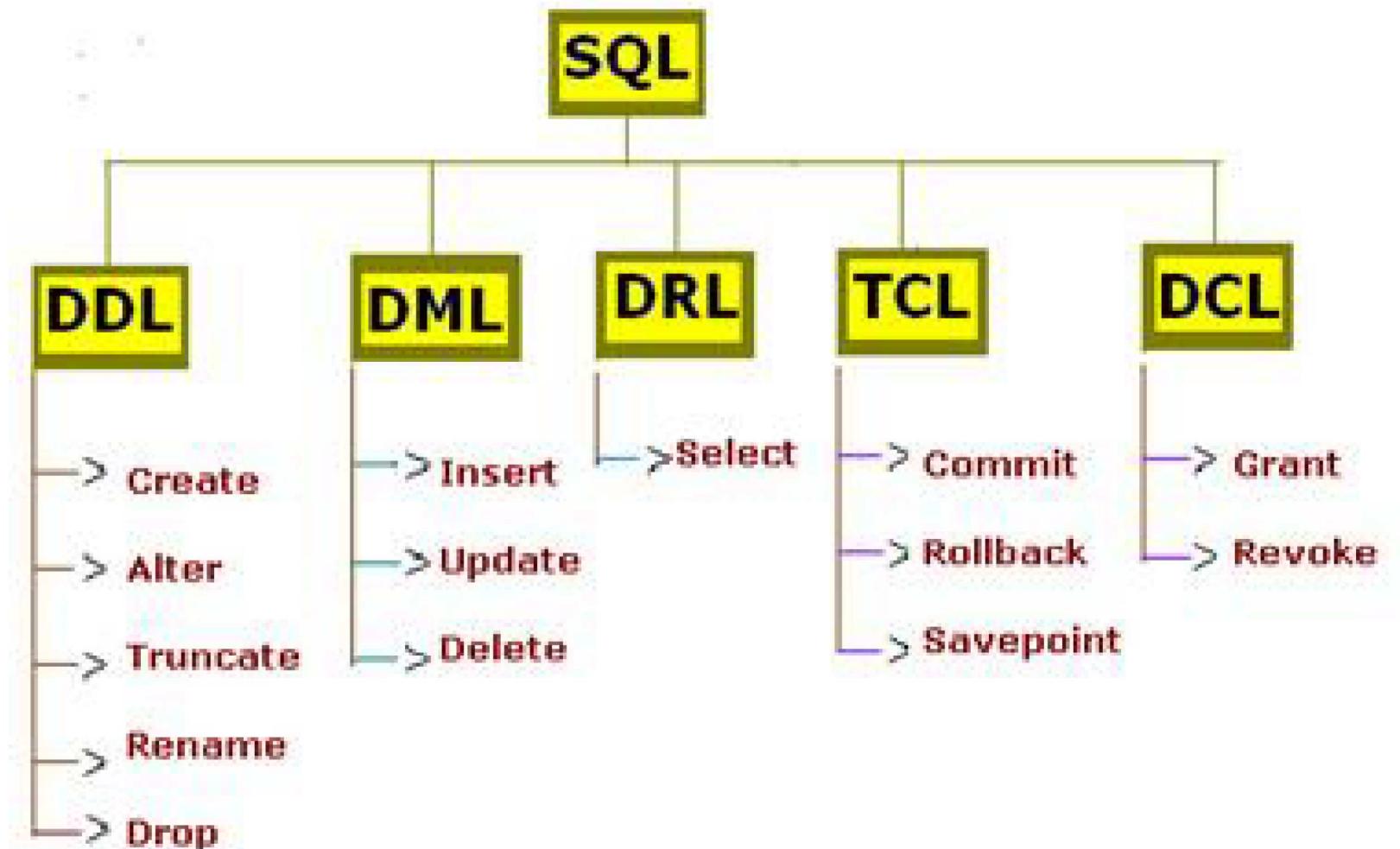


SQL 語法分類 - DML

2. Data Manipulation Language 資料處理語言 (DML)

◆ 用來處理資料表裡的資料，常見的指令有：

- INSERT：新增資料到資料表中
- UPDATE：修改資料表中的資料
- DELETE：刪除資料表中的資料

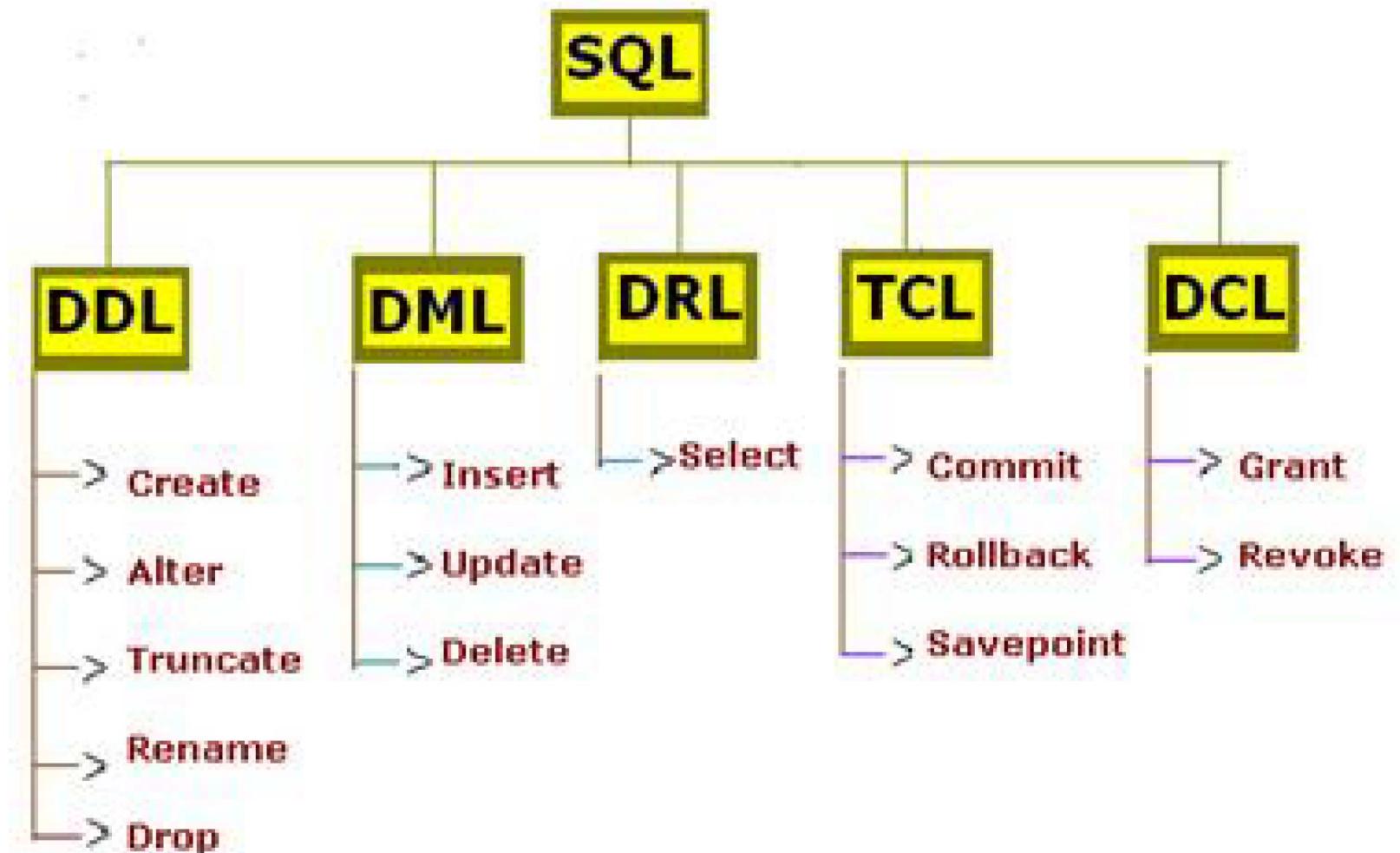


SQL 語法分類 - DQL/DRL

3. Data Queries Language 資料查詢語言 (DQL)

- ◆ DQL 也稱為DRL : Data Retrieval Language ; 有些書籍或網站會將其歸納在DML中。
- ◆ 用來查詢資料表裡的資料，常見的指令有：

- SELECT : 選取資料庫中的資料



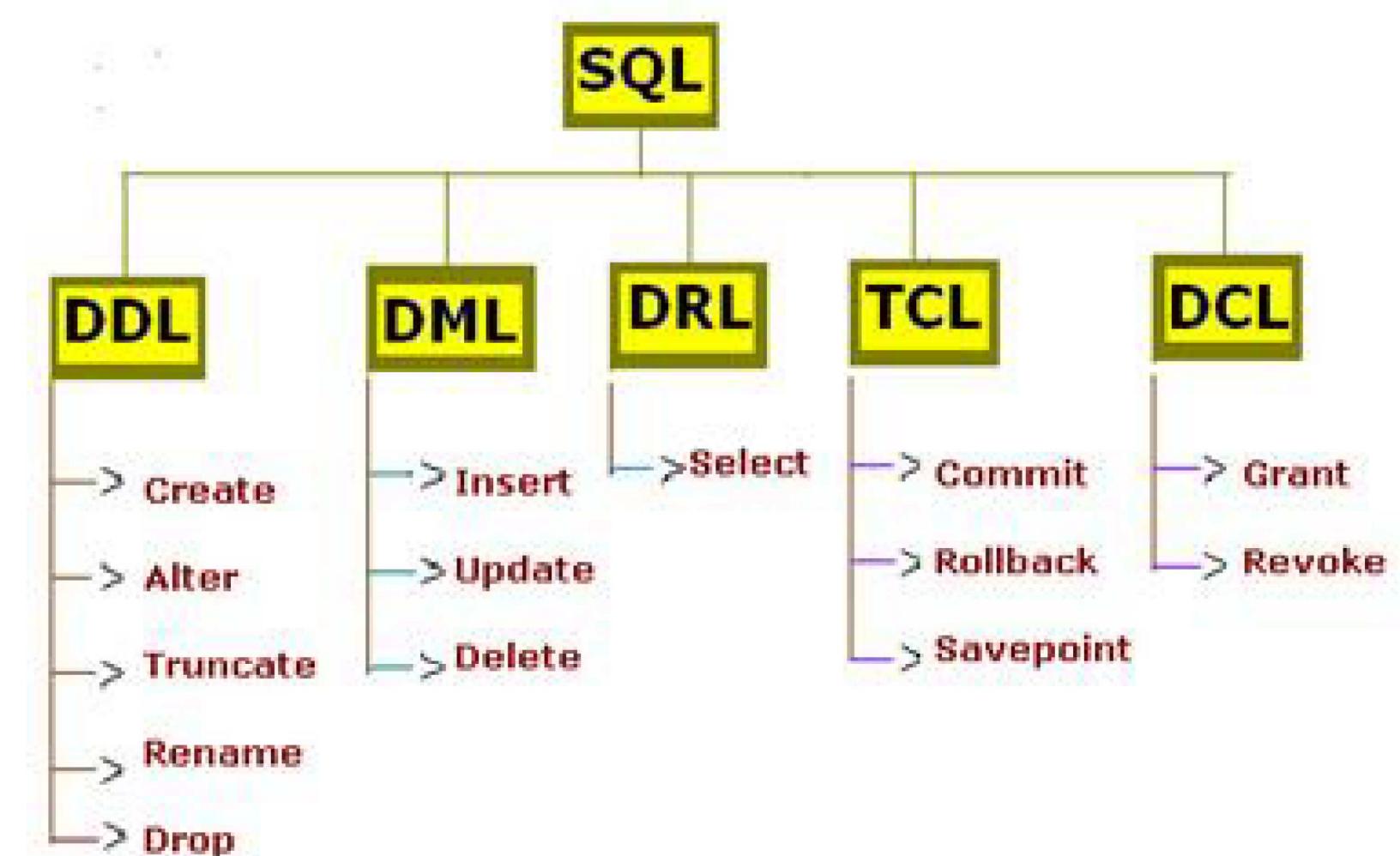
SQL 語法分類 - DCL

4. Data Control Language 資料控制語言 (DCL)

◆ 提供管理資料庫或資料表授權使用之命令，可用來控制資料表、檢視表的存取權限，常見的指令有：

➤ GRANT：賦予使用者使用權限

➤ REVOKE：取消使用者的使用權限



SQL 語法分類 - TCL

5. Transaction Controls 交易控制語言 (TCL)

◆ 交易是單一工作單元。如果交易成功，便會確定交易期間所修改的所有資料，且會成為資料庫中永久的內容。如果交易發現錯誤，必須取消或回復，便會清除所有的資料修改。

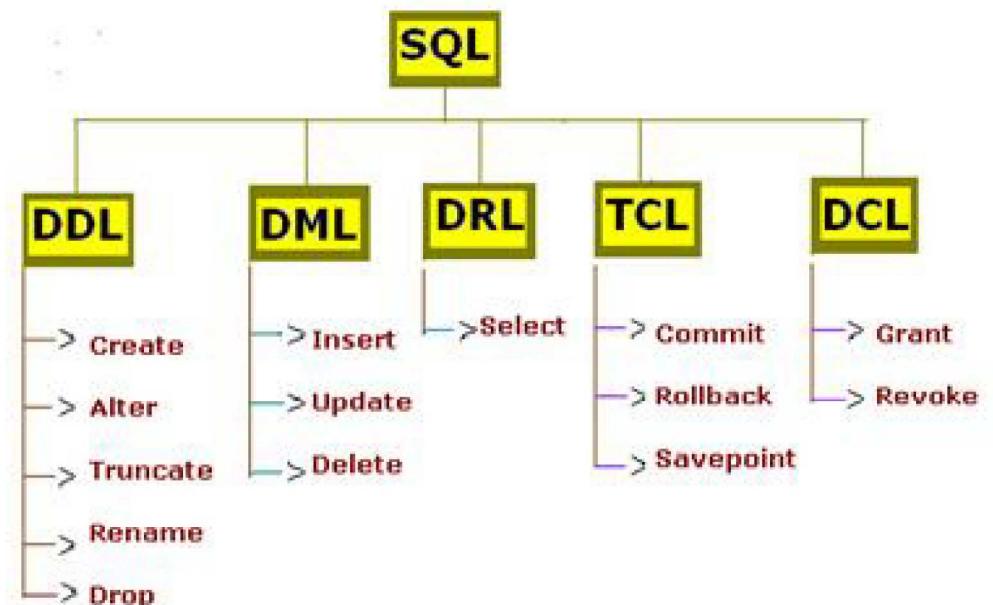
◆ 交易控制語言命令主要用在多人使用的環境

➤ COMMIT：完成交易作業

➤ ROLLBACK：交易作業異常，將已變動的資料回復到交易開始的狀態

➤ SAVEPOINT：儲存還原點，以後可以將資料還原(ROLLBACK)到為上一次COMMIT的時間點

➤ SET TRANSACTION：設定交易選項（如隔離）





SQL 語法概觀 - 資料表

- ◆ 首先，讓我們先來了解組成 SQL 語法的元素有什麼。

資料表 (Database Tables)

- ◆ 資料庫中最重要的物件就是資料表 (table)，資料庫由一個或一個以上的資料表所構成，每個資料表名稱在資料庫中都是唯一的，資料表中每一直行 (column) 稱之為欄位，每個欄位都有其資料型態 (data type)，由不同欄位所組成的橫列 (row)，稱之為記錄 (record)，我們舉一個叫做 "customers" 的資料表作為例子：此資料表共包含 2 筆記錄，5 個欄位 (C_Id, Name, City, Address, Phone)
- ◆ 資料表名稱有區分大小寫 (case-sensitive)，但某些資料庫在 Windows 作業系統中是不分大小寫的，而為了方便維護最好統一你的命名方式。

C_Id	Name	City	Address	Phone
1	張一	台北市	XX路100號	02-12345678
2	王二	新竹縣	YY路200號	03-12345678



SQL 敘述句 (Statements)

- ◆ SQL 語言是由命令 (commands) 、子句 (clauses) 、運算子 (operators) 及 函數 (functions) 所組成的敘述句，而我們利用 SQL 敘述句來跟資料庫溝通、下達指令。
- ◆ 通常一個 SQL 敘述句由一段命令句開始描述您要對資料庫要求的動作，接著可能會接著條件語句，最後以分號 「;」 結束，但有些資料庫並不強制一定要加上結尾分號。

SELECT * FROM customer ;

- ◆ 以上的例子便是一個SQL語法(敘述句)，表示從 customer 資料表取出所有的資料。
- ◆ SQL 語句不區分大小寫 (case-insensitive)，如 select * from customer; 和上面的例子是相同的結果。



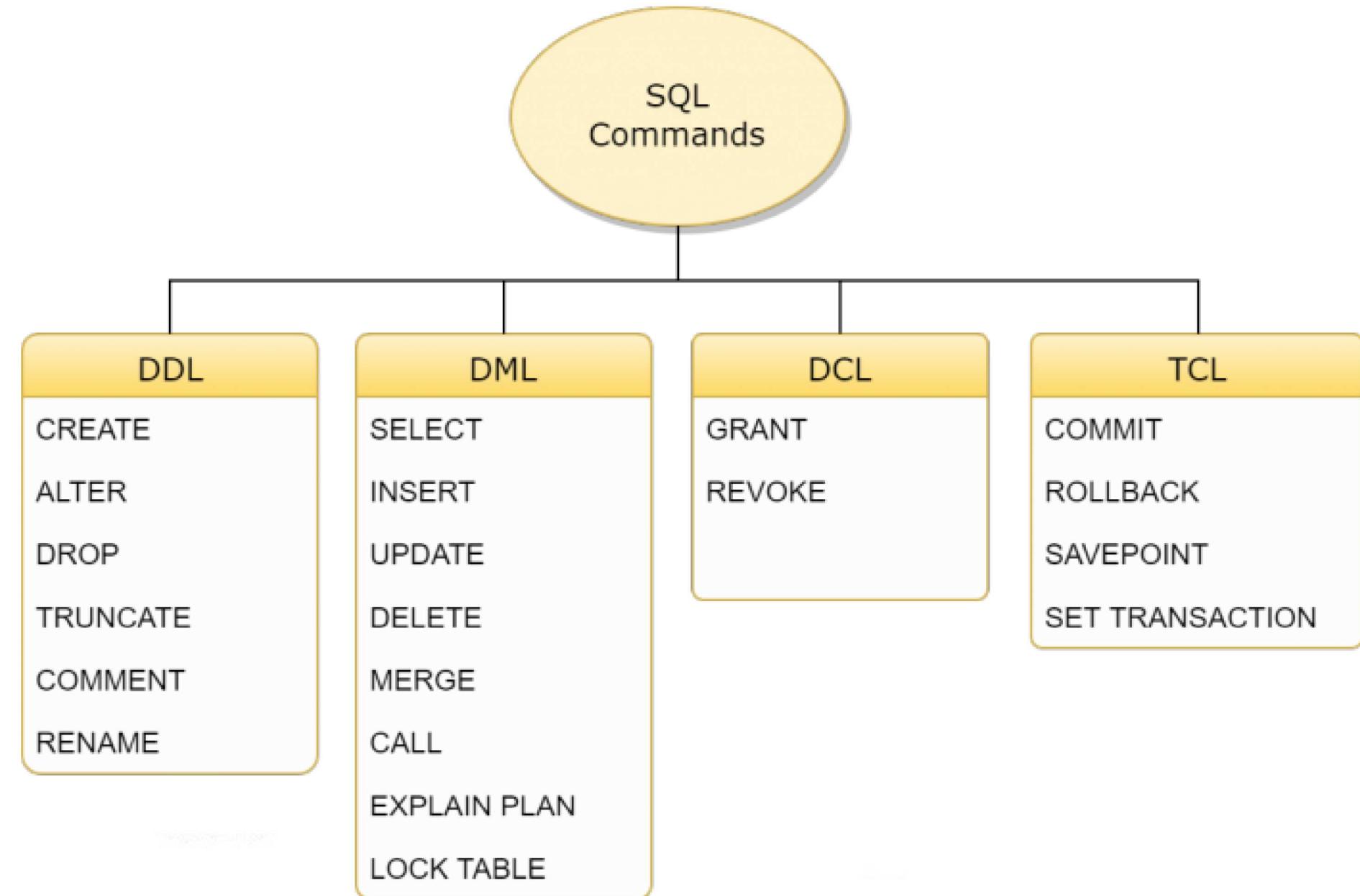
SQL 敘述句的書寫習慣

- ◆ 將所有的執行語句中的 SQL 關鍵字大寫是一個良好的 SQL 書寫習慣，這會幫助你更輕易的去檢視你的 SQL 語法。
- ◆ 此外，你可以將較長的 SQL 語句拆成多行書寫，這會讓你比較容易閱讀！

```
SELECT column_one, column_two  
FROM table_name  
WHERE table_id = 123;
```

什麼是命令 (Commands)

- ◆ 建立新的資料庫、資料表、欄位及、索引等，或建立查詢表、排序、過濾資料、查詢、修改、新增及刪除資料等動作。
(即 CREATE、DROP、ALTER、SELECT、INSERT、UPDATE、DELETE 等命令)



什麼是運算子 (Operators)

- ◆ 運算子用來幫助 SQL 語句處理數值、字串或進行邏輯運算及比較條件。

Operator	Description	Example
=	Equal to	$(x=y)$ is not true
!=	Equal or not	$(x!=y)$ is true
< >	Not equal to	$(x<>y)$ is true
>	Greater than	$(x>y)$ is not true
<	Less than	$(x<="" td="">>$
>=	Greater than or equal to	$(x>=y)$ is not true
<=	Less than or equal to	$(x<=y)$ is true
!<	Not less than	$(x!<="" td="">>$
!>	Not greater than	$(x!>y)$ is true

運算子	說明
LIKE	包含，只需子字串即符合條件
BETWEEN/AND	在一個範圍之內
IN	屬於清單其中之一
IS NULL	是否是空值
EXISTS	檢查括號中 SQL 指令查詢的記錄是否存在
NOT	非，可以否定運算式的結果
AND	且，需要連接的 2 個運算子都會真，才是真
OR	或，只需其中一個運算子為真，即為真

什麼是函數 (Functions)

◆ SQL 語言內建許多函數可以直接在 SQL 語句裡面使用，像是取得某欄位加總後數值、取得某欄位內最大或最小值等等。

- [SQL Functions](#)
- [SQL AVG\(\)](#)
- [SQL COUNT\(\)](#)
- [SQL MAX\(\)](#)
- [SQL MIN\(\)](#)
- [SQL SUM\(\)](#)
- [SQL GROUP BY](#)
- [SQL HAVING](#)
- [SQL ASCII\(\)](#)
- [SQL CHAR\(\)](#)
- [SQL CONCAT\(\)](#)
- [SQL LENGTH\(\)](#)
- [SQL REPLACE\(\)](#)
- [SQL UCASE\(\)](#)
- [SQL LCASE\(\)](#)
- [SQL MID\(\)](#)
- [SQL ABS\(\)](#)
- [SQL CEIL\(\)](#)
- [SQL FLOOR\(\)](#)
- [SQL POWER\(\)](#)
- [SQL ROUND\(\)](#)
- [SQL SQRT\(\)](#)
- [SQL PI\(\)](#)
- [SQL EXP\(\)](#)
- [SQL LOG\(\)](#)
- [SQL 三角函數](#)
- [SQL TRIM\(\)](#)



資料定義 DDL - 建立新資料庫

◆ SQL 語法在應用上對於 DDL (Data Definition Language)：定義資料庫物件使用的語法是很基礎而重要的，其中重要的功能關鍵字有：

- CREATE：建立資料庫的物件。
- ALTER：變更資料庫的物件。
- DROP：刪除資料庫的物件。



資料定義 DDL - 建立新資料庫

◆ CREATE : 建立資料庫及資料表

1. 建立資料庫

- CREATE 是 SQL 指令中建立資料庫或資料表的關鍵字，新增資料庫的語法如下：

CREATE DATABASE [IF NOT EXISTS] 資料庫名稱

[DEFAULT] CHARACTER SET [=] 字元集

[DEFAULT] COLLATE [=] 編碼

- 在語法中以「[]」左右括號所包含的內容表示可以不填，但是「[IF NOT EXISTS]」表示如果沒有該資料庫時才執行建立資料庫的動作。



資料定義 DDL - 建立新資料庫

◆ CREATE DATABASE 敘述句 (SQL CREATE DATABASE Statement)

◆ CREATE DATABASE 是我們用來建立一個新資料庫的語法。

CREATE DATABASE 語法 (Syntax)

CREATE DATABASE database_name;

◆ 例如，我們想建立一個叫做 xyz 的資料庫：

CREATE DATABASE xyz;

◆ 這樣就可以建立完成。接著我們就可以新增資料表到此資料庫



資料定義 DDL - 建立新資料庫

- ◆ **CREATE** : 建立資料庫及資料表

2. 建立資料表

- 新增資料表的動作更為頻繁而重要，其語法如下：

CREATE TABLE [IF NOT EXISTS] 資料表名稱

(欄位名稱 資料類別 [資料屬性])

[, 欄位名稱 資料類別 [資料屬性]]...);

- 可以使用 CREATE TABLE 的指令來建立資料表，其中包含了定義及新增資料表欄位的內容。也就是新增資料表的過程，可以一併新增資料表中的欄位，讓整個資料表能夠完整。



資料定義 DDL - 建立新資料表

◆ CREATE TABLE 敘述句 (SQL CREATE TABLE Statement)

◆ CREATE TABLE 是我們在資料庫中用來建立一個新資料表的語法。

CREATE TABLE 語法 (Syntax)

```
CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type, ... );
```

◆ `data_type` 用來指定該欄位資料儲存的資料型別，要注意的是，不同的資料庫其型別會有所差異。



資料定義 DDL - 建立新資料表

◆ CREATE TABLE 敘述句 (SQL CREATE TABLE Statement)

◆ 假設我們現在想建立一個 "customers" 資料表，其中包含這幾個欄位 - C_Id, Name, Address, Phone :

```
CREATE TABLE customers (
    C_Id INT,
    Name varchar(50),
    Address varchar(255),
    Phone varchar(20) );
```

◆ 新建資料表看起來會像是這個樣子：

C_Id	Name	Address	Phone

◆ 接著，我們就可以用 INSERT INTO 來加入資料。



資料定義 DDL - 變更資料庫及資料表內容

◆ ALTER : 變更資料庫及資料表內容

➤ 資料庫或是資料表在新增後，可以使用 ALTER 語法進行資料庫的修改或資料表的調整(新增、修改與刪除)。

1. 修改資料庫：ALTER DATABASE 語法可以修改資料庫的字元集及編碼方式。

ALTER DATABASE 指令可以修改存在的資料庫結構，基本語法如下：

ALTER DATABASE 資料庫名稱

[DEFAULT] CHARACTER SET [=] 字元集

[DEFAULT] COLLATE [=] 編碼



資料定義 DDL - 變更資料庫及資料表內容

◆ ALTER : 變更資料庫及資料表內容

2. 新增資料表欄位 : ALTER TABLE 語法也可以在已存在的資料表中新增資料表欄位，其語法如下：

ALTER TABLE 資料表名稱

ADD 欄位名稱 資料類別 [資料屬性]

[,ADD 欄位名稱 資料類別 [資料屬性]]...)

[FIRST | AFTER 欄位名稱];

➤ 新增的欄位預設是放置在所有欄位之後，您也可以利用「FIRST」將欄位新增在所有欄位之前，或是用「AFTER 欄位名稱」放在指定欄位之後。



資料定義 DDL - 更改資料表結構

◆ ALTER : 變更資料庫及資料表內容

- ALTER TABLE 是用來對已存在的資料表結構作更改，語法如下：

ALTER TABLE table_name ...;

- 假設現在我們已經建立好一個 "customers" 資料表：

C_Id	Name	Address	Phone
------	------	---------	-------

- 接著，我們要增加欄位 (ADD COLUMN)，語法如下：

ALTER TABLE table_name ADD column_name datatype;

- 舉例：如果我們想增加一個 Discount 欄位，語法如下：

ALTER TABLE customers ADD Discount VARCHAR(10);



資料定義 DDL - 變更資料庫及資料表內容

◆ ALTER : 變更資料庫及資料表內容

3. 修改資料表欄位：ALTER TABLE 語法可以修改存在的資料表結構，基本語法如下：

ALTER TABLE 資料表名稱

CHANGE 原欄位名稱 新欄位名稱 資料類別 [資料屬性]

[, 原欄位名稱 新欄位名稱 資料類別 [資料屬性]]...);



資料定義 DDL - 變更資料庫及資料表內容

◆ ALTER : 變更資料庫及資料表內容

4. 刪除資料表欄位：ALTER TABLE 語法可以在已存在的資料表中刪除資料表欄位，其語法如下：

ALTER TABLE 資料表名稱

DROP 欄位名稱；



資料定義 DDL - 更改資料表結構

◆ ALTER : 變更資料庫及資料表內容

- 我們要更改欄位資料型別 (ALTER COLUMN TYPE) , 語法如下 :

ALTER TABLE table_name ALTER COLUMN column_name datatype;

- 例如 , 更改 Discount 欄位的資料型別 , 語法如下 :

ALTER TABLE customers ALTER COLUMN Discount DECIMAL(18, 2);

- 接著 , 我們要刪除欄位 (DROP COLUMN) , 語法如下 :

ALTER TABLE table_name DROP COLUMN column_name;

- 例如 , 刪除 Discount 欄位 , 語法如下 :

ALTER TABLE customers DROP COLUMN Discount;



資料定義 DDL - 變更資料庫及資料表內容

◆ DROP : 刪除資料庫及資料表內容

- 資料庫或是資料表在新增後，可以使用 DROP 語法進行修改。
- 刪除資料庫與資料表的語法相似，刪除資料庫的語法如下，這個刪除資料庫的動作會連同儲存在資料庫中的所有物件，如資料表都一同刪除。

```
DROP DATABASE [IF EXISTS] 資料庫名稱；
```

- 刪除資料表的語法如下，這個刪除資料表的動作會連同儲存在資料表中的所有資料都一同刪除。

```
DROP TABLE [IF EXISTS] 資料表名稱；
```



資料定義 DDL - 刪除資料表或資料庫

◆ DROP : 刪除資料庫及資料表內容

- 我們可以用 DROP DATABASE / DROP TABLE / TRUNCATE TABLE 來刪除資料庫或資料表。

DROP TABLE / TRUNCATE TABLE / DROP DATABASE Statement

- 刪除資料庫 (DROP DATABASE)

`DROP DATABASE database_name;`

- 刪除資料表 (DROP TABLE) : 完全刪除整個資料表。

`DROP TABLE table_name;`

- 僅刪除資料表內容，但保留結構 (TRUNCATE TABLE) : 資料表還在，只是資料清空了。

`TRUNCATE TABLE table_name;`



資料定義 DDL - SQL Constraint 限制

- ◆ Constraint 是指有條件地限制哪些資料才可以被存入資料表中，也就是對欄位作約束。這些限制可以在建立資料表時 CREATE TABLE 指定條件，或是之後使用 ALTER TABLE 指定。
- ◆ SQL 有以下類型的限制條件，以下將逐一說明
 - NOT NULL 非空值限制
 - UNIQUE 唯一限制
 - PRIMARY KEY 主鍵限制
 - FOREIGN KEY 外鍵限制
 - CHECK 檢查限制
 - DEFAULT 預設值限制



資料定義 DDL - NOT NULL 限制

◆ NOT NULL 限制 (SQL NOT NULL Constraint)

- NOT NULL 用來限制該欄位不能接受空值，而在預設的情況下，一個欄位是允許接受空值的。
- 當某欄位限制為 NOT NULL 時，則在新增資料時該欄位一定要有值。
- 例如，我們建立一張 customer 資料表，並限制其 C_Id 及 Name 欄位值不能是空值：

```
CREATE TABLE customer (
    C_Id INT NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20)
);
```



資料定義 DDL - UNIQUE 唯一限制

◆ UNIQUE 唯一限制 (SQL UNIQUE Constraint)

- UNIQUE 用來保證欄位在資料表中的唯一性，約束資料表中的欄位不能有重複的資料。
- 一個資料表可有多個 UNIQUE 欄位，此外 UNIQUE 欄位中可以接受 NULL 值。
- 假設我們要對「customer」資料表限制「C_Id」欄位不能有重複值存在：在建立資料表時 CREATE TABLE...可以有以下寫法：

```
CREATE TABLE customer (
    C_Id INT NOT NULL UNIQUE,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20)
);
```

```
CREATE TABLE customer (
    C_Id INT NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20),
    UNIQUE (C_Id) );
```



資料定義 DDL - UNIQUE 唯一限制

◆ UNIQUE 唯一限制 (SQL UNIQUE Constraint)

- 更改資料表限制 ALTER TABLE...

```
ALTER TABLE customer ADD UNIQUE (C_Id);
```

- 替唯一鍵命名與多欄位的唯一限制：

```
ALTER TABLE customer  
ADD CONSTRAINT u_Customer_Id UNIQUE (C_Id, Name);
```



資料定義 DDL - UNIQUE 唯一限制

◆ UNIQUE 唯一限制 (SQL UNIQUE Constraint)

- 移除資料表限制 ALTER TABLE...
- MySQL :

```
ALTER TABLE customer DROP INDEX u_Customer_Id;
```

- SQL Server / Oracle / MS Access :

```
ALTER TABLE customer DROP CONSTRAINT u_Customer_Id;
```



資料定義 DDL - PRIMARY KEY 主鍵限制

◆ PRIMARY KEY 主鍵限制 (SQL PRIMARY KEY Constraint)

- PRIMARY KEY 用來保證欄位在資料表中的唯一性，主鍵欄位中的每一筆資料在資料表中都必需是獨一無二的。
- PRIMARY KEY 有點類似 UNIQUE 加上 NOT NULL。一個資料表中只能有一個 PRIMARY KEY，但是可以有多個 UNIQUE。
- 假設我們要將 customer 資料表中的 C_Id 欄位設為主鍵：在建立資料表時 CREATE TABLE...，可以有以下寫法：

```
CREATE TABLE customer (
    C_Id INT NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20)
);
```

```
CREATE TABLE customer (
    C_Id INT NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20),
    PRIMARY KEY (C_Id)
);
```



資料定義 DDL - PRIMARY KEY 主鍵限制

◆ PRIMARY KEY 主鍵限制 (SQL PRIMARY KEY Constraint)

- 當主鍵包含多個欄位時，我們稱之為組合鍵 (Composite Key)。
- 我們限制 C_Id 及 Name 這兩個欄位為主鍵，CONSTRAINT 後面接著的即是此主鍵的名稱。
- 以下是我們替主鍵命名與多欄位的組合鍵 (Composite Primary Keys)：

```
CREATE TABLE customer (
    C_Id INT NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20),
    CONSTRAINT pk_Customer_Id PRIMARY KEY (C_Id, Name)
);
```



資料定義 DDL - FOREIGN KEY 外鍵限制

- ◆ FOREIGN KEY 外鍵限制 (SQL FOREIGN KEY Constraint)
- ◆ 讓我們簡單了解一下什麼是外鍵：這是一個客戶資料表 customers

C_Id	Name	City	Address	Phone
1	張一	台北市	XX路100號	02-12345678
2	王二	新竹縣	YY路200號	03-12345678
3	李三	高雄縣	ZZ路300號	07-12345678

- ◆ 而這是客戶訂單的資料表 orders

O_Id	Order_No	C_Id
1	2572	2
2	7375	3
3	7520	1

- ◆ 在這裡我們會想有一個限制，就是在客戶訂單資料表中的客戶，都一定要在 customers 資料表中存在。所以我們需要在 orders 資料表中設定一個外鍵，再將此外鍵指向 customers 資料表中的主鍵，以確定所有在 orders 資料表中的客戶都存在於 customers 資料表中，才不會有任何幽靈訂單的出現！



資料定義 DDL - FOREIGN KEY 外鍵限制

◆ FOREIGN KEY 外鍵限制 (SQL FOREIGN KEY Constraint)

- 外鍵是一個 (或多個) 指向其它資料表中主鍵的欄位，它限制欄位值只能來自另一個資料表的主鍵欄位，用來確定資料的參考完整性 (Referential Integrity)。
- 如果想在 MySQL 資料庫中使用外鍵限制，必需讓資料表使用 InnoDB 儲存引擎。

C_Id	Name	City	Address	Phone
1	張一	台北市	XX路100號	02-12345678
2	王二	新竹縣	YY路200號	03-12345678
3	李三	高雄縣	ZZ路300號	07-12345678

O_Id	Order_No	C_Id
1	2572	2
2	7375	3
3	7520	1



資料定義 DDL - FOREIGN KEY 外鍵限制

◆ FOREIGN KEY Constraint

- 假設我們要將 customer 資料表中的 C_Id 欄位設為外鍵：在建立資料表時 CREATE TABLE...

```
CREATE TABLE orders (
    O_Id INT NOT NULL,
    Order_No INT NOT NULL,
    C_Id INT, PRIMARY KEY (O_Id),
    FOREIGN KEY (C_Id) REFERENCES customers(C_Id)
);
```



資料定義 DDL - FOREIGN KEY 外鍵限制

◆ FOREIGN KEY Constraint

- 替外鍵命名與多欄位的外鍵：

```
CREATE TABLE orders (
    O_Id INT NOT NULL PRIMARY KEY,
    Order_No INT NOT NULL,
    C_Id INT, CONSTRAINT fk_Cusomer_Id FOREIGN KEY (C_Id) REFERENCES customers(C_Id)
);
```

- 我們限制 C_Id 為外鍵，CONSTRAINT 後面接著的即是此外鍵的名稱，另一個重點是記得 customers 資料表中需將 C_Id 設為主鍵。



資料定義 DDL - DEFAULT 預設值限制

◆ DEFAULT 預設值限制 (SQL DEFAULT Constraint)

- DEFAULT 限制用來設定欄位的預設值。當你在 INSERT 資料時若該欄位沒指定值則會採用預設值。
- 假設我們要設定 customer 資料表中的 Address 欄位預設值為 "未知"：在建立資料表同時 CREATE TABLE...

```
CREATE TABLE customer (
    C_Id INT NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(255) DEFAULT '未知',
    Phone VARCHAR(20)
);
```



資料定義 DDL - AUTO INCREMENT 欄位

◆ AUTO INCREMENT 欄位 (SQL AUTO INCREMENT column)

- AUTO INCREMENT 欄位會自動遞增資料行的值，因為每次新增資料時欄位值都會自動遞增也就是說 AUTO INCREMENT 欄位值會是唯一的，該欄位用途就像是一個識別碼或流水號，而 AUTO INCREMENT 常與 Primary Key 一起搭配使用。
- 設定 AUTO INCREMENT 欄位

```
CREATE TABLE customers (
    C_Id INT AUTO_INCREMENT,
    Name varchar(50),
    Address varchar(255),
    Phone varchar(20),
    PRIMARY KEY (C_Id)
);
```



資料定義 DDL - AUTO INCREMENT 欄位

◆ AUTO INCREMENT 欄位 (SQL AUTO INCREMENT column)

- MySQL 語法使用 AUTO_INCREMENT 這關鍵字。注意要將 AUTO_INCREMENT 欄位指定為 PRIMARY KEY，否則會有錯誤！
- 新增一筆資料：C_Id 欄位不需要指定值，MySQL 預設會由1開始逐列自動遞增 (2, 3, 4...)。不過你也可以替 AUTO_INCREMENT 欄位指定一個起始值，語法如下：

ALTER TABLE table_name AUTO_INCREMENT=起始數字;

```
CREATE TABLE customers (
    C_Id INT AUTO_INCREMENT,
    Name varchar(50),
    Address varchar(255),
    Phone varchar(20),
    PRIMARY KEY (C_Id)
);
```

```
INSERT INTO customers (Name, Address, Phone)
VALUES ('姓名XXX', '地址XXX', '電話XXX');
```



資料定義 DDL - View 檢視表、視圖 (SQL View)

- ◆ View 是藉由 SQL SELECT 查詢動態組合生成的資料表 (亦即 View 是由查詢得到的結果集組合而成的資料表)。View 內的資料紀錄是由其它實際存在的資料表中產生的，它就像是一個虛擬資料表，實際上資料庫 (或說是硬碟) 裡面是不存在這一個資料表的 (只存在此 View 的相關定義)，但是我們使用上卻有如實際存在的資料表 - 所有的 SQL 查詢語法都可以操作在此 View 上。
- ◆ 資料表是一種實體結構 (physical structure)，而 View 是一種虛擬結構 (virtual structure)。



資料定義 DDL - View 檢視表、視圖 (SQL View)

View 的特性

- ◆ 加強資料庫的安全性，View 可以將實體資料表結構隱藏起來，同時限制使用者只可以檢視及使用哪些資料表欄位。
- ◆ 檢視表是唯讀的，亦即外部使用者無法直接透過 View 去修改內部資料。
- ◆ 將複雜的 SQL 查詢包裝在 View 中，可以簡化查詢的複雜度。
- ◆ 當資料表結構有變更時，只需要更改 View 的設定，不需更改程式。



資料定義 DDL - View 檢視表、視圖 (SQL View)

建立 View (SQL CREATE VIEW)

```
CREATE VIEW view_name [(column_list)] AS  
SELECT column_name(s) F  
ROM table_name  
WHERE condition;
```

- ◆ 例如以下例子，一個 View 可以由查詢實體資料表而建立，亦可以查詢其它已存在的 View 而建立。

CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;



資料定義 DDL - View 檢視表、視圖 (SQL View)

更新 View (SQL CREATE OR REPLACE VIEW)

- ◆ 如果加上 OR REPLACE 子句的意思就是若同名的 View 已經存在就覆蓋取代它。如果 View 不存在，我們可以把 CREATE OR REPLACE VIEW 看做是如同 CREATE VIEW；而如果 View 已存在，我們可以把 CREATE OR REPLACE VIEW 看做是 ALTER VIEW。

```
CREATE OR REPLACE VIEW view_name [(column_list)] AS  
SELECT column_name(s)  
FROM table_name  
WHERE condition;
```



資料定義 DDL - View 檢視表、視圖 (SQL View)

更新 View (SQL CREATE OR REPLACE VIEW)

- ◆ 如果加上 OR REPLACE 子句的意思就是若同名的 View 已經存在就覆蓋取代它。如果 View 不存在，我們可以把 CREATE OR REPLACE VIEW 看做是如同 CREATE VIEW；而如果 View 已存在，我們可以把 CREATE OR REPLACE VIEW 看做是 ALTER VIEW。

```
CREATE OR REPLACE VIEW view_name [(column_list)] AS  
SELECT column_name(s)  
FROM table_name  
WHERE condition;
```

刪除 View (SQL DROP VIEW)

```
DROP VIEW view_name;
```



資料定義 DDL - View 檢視表、視圖 (SQL View)

View 使用實例 (Example)

- ◆ 假設這是一個產品訂單資料表 p_orders：我們可以建立一個方便查詢各產品售出總額的 View：

```
CREATE VIEW view_p_sum (Product, P_SUM) AS  
SELECT Product, Price*Quantity FROM p_orders GROUP BY Product;
```

Product	Price	Quantity
LCD	4000	100
CPU	5000	200

- ◆ 接著，你就可以像操作一般資料表：

```
SELECT * FROM view_p_sum
```

Product	P_SUM
CPU	1000000
LCD	400000

Q & A