MySQL-SQL

李偉銘

大綱

- SQL的基本觀念
- DQL
  - select敘述
    - select子句
    - where子句
    - order by子句
    - group by子句
    - having子句
    - from子句
    - limit子句
    - 子查詢
    - 子句順序
- DML
  - insert敘述
  - delete敘述
  - update敘述
- TCL
  - set autocommit敘述
  - start transaction敘述
  - commit敘述
  - rollback敘述
  - savepoint敘述
- DDL
  - create table敘述
    - 欄位定義
    - 約束定義
    - 其他
  - drop table敘述
  - alter table敘述
    - 針對欄位
    - 針對約束
    - 針對資料表
  - truncate table敘述
- DCL
  - grant敘述
  - revoke敘述

SQL的基本觀念 (1/3)

- SQL語言
  - 分成5種子語言(DML、TCL、DQL、DDL、DCL)
  - 每個子語言又分成數個敘述(Statement)
  - EX. DML有3個敘述:insert、delete、update
  - 每個敘述又分成多個子句(Clause)
  - EX. select敘述有7個子句:select、from、where、group by、having、order by、limit

SQL的基本觀念 (2/3)

- 撰寫SQL敘述的好習慣
  - 指令預設不區分大小寫，但應保持一致風格
  - 敘述用分號(;)結尾
  - 可寫成多行，每個子句換新行
  - 用縮排增加可讀性
- 講義中SQL語句的顏色
  - 綠色: 註解
  - **深紫色: 關鍵字**
  - 咖啡色: 識別字。EX.表格名、欄位名
  - 粉紅色: 函數
  - 藍色: 字串

```sql
-- 範例-撰寫敘述的好習慣
select
    e.DEPTNO,
    d.DNAME,
    COUNT(*)
from
    EMP e
    left join DEPT d
        on e.DEPTNO = d.DEPTNO
where
    e.MGR is not null
    and d.LOC in('DALLAS', 'CHICAGO')
group by
    e.DEPTNO, d.DNAME
having
    COUNT(*) > 2
order by
    COUNT(*) desc
limit
    3;
```

SQL的基本觀念 (3/3)

- SQL中的註解
  - 單行註解

```
# 註解文字
-- 註解文字
```

  - 多行註解

```
/*
    註
    解
    文
    字
*/
```

```
-- 攏長的SQL閱讀性極差

# 即使是自己撰寫的SQL也容易看不懂

/*
    在SQL前加上註解，
    在學習階段有很大的幫助
*/
```

DQL

- 簡述
  - Data Query Language，資料查詢語言
  - 用以查詢Table內的資料
  - 僅查詢，不會改變資料
- 敘述
  - select: 查詢Table的資料

- select敘述的子句
  - select: 顯示的欄位
  - from: 資料的來源
  - where: 過濾條件
  - group by: 資料分組的依據
  - having: 分組欄位的過濾條件
  - order by: 查詢結果的排序方式
  - limit: 查詢結果的忽略/限制筆數

select敘述 (1/2)

- 語法

```
select
    欄位名1,
    ..,
    欄位名N
from
    表格名
```

*註: select子句和from子句是select敘述的必要子句!

- 說明
  - 欄位名: 欲顯示的Column名，可用星號(*)表示全部Column
  - 表格名: 欲查詢的Table名

select敘述 (2/2)

- 範例

```
-- 範例-select敘述-簡單查詢
select
    *
from
    EMP
```

select敘述-select子句 (1/8)

- 語法

```
select
    {欄位名|運算式|字面常數} [[as] 別名]
    ..,
from
    表格名
```

- 說明
  - 運算式(Expression): 任何合法的運算式皆可，算術運算、字串運算、函數呼叫..等等，EX. 32 + 1、ENAME + 'william'、NOW()
  - 字面常數(Literal): 直接指定的一個值，EX. 'william'
  - 別名(Alias): 自訂的名稱，若含有中文字或其他特殊字時，需要引號包含。as關鍵字可省略

select敘述-select子句 (2/8)

- 範例

```
-- 範例-select敘述-select子句1
select
    ENAME as '姓名',
    SAL + IFNULL(COMM, 0) '收入',
    NOW() as '統計點'
from
    EMP
```

select敘述-select子句 (3/8)

- 語法

```
select all|distinct|distinctrow
    ..
from
    表格名
```

- 說明
  - all: 表示列出全部的資料，不管重複與否，預設即為all，可省略
  - distinct: 表示只列出不同的資料，去除重複資料
  - distinctrow: distinct的同義字

select敘述-select子句 (4/8)

- 範例

```
-- 範例-select敘述-select子句2
-- 列出所有職稱。重複職稱不會顯示多次
select distinct
    JOB
from
    EMP
```

select敘述-select子句 (5/8)

- 空值(null)
  - 意義為沒有值，不等於0，不等於空字串，不等於空白字串，也不等於自己
  - insert資料時，若無指定某欄位之值時，就會放入null

- 遇判斷式: 用is或<=>判斷
  - null = null → null
  - null != null → null
  - null is null → true
  - null <=> null → true

- 遇運算式: 結果都會是null
  - null + null → null
  - null + 1 → null
  - 1 + null → null

select敘述-select子句 (6/8)

- 字串串接
  - MySQL的+運算子雖可用在字串上，但意義非串接，而是試著將字串轉數字，然後再做加法運算
  - 字串串接需使用函數CONCAT()
    - CONCAT('willi', 'am', 's') → 'williams'
    - CONCAT('willi', 'am', null) → null

select敘述-select子句 (7/8)

- case運算式
  - 又分成2種用法: 列舉式、條件式
  - 列舉式(Simple Case)
    - 說明: 用於確切值(列舉值)

```
case 欄位名 | 運算式 | 字面常數
    when 列舉值1 then 結果1
    ..
    [when 列舉值N then 結果N]
    [else 結果]
end
```

```
-- 範例-select敘述-case when 1
select
    case LOC
        when 'NEW YORK' then 'NY'
        when 'DALLAS' then 'DAL'
        when 'CHICAGO' then 'CHI'
        when 'BOSTON' then 'BSN'
        else 'UNKNOWN'
    end as ABBR   -- ←一般會加上別名
from DEPT
```

select敘述-select子句 (8/8)

- case運算式
  - 條件式(Searched Case)
    - 說明: 常用於範圍

```
case
    when 條件1 then 結果1
    ..
    [when 條件N then 結果N]
    [else 結果]
end
```

```
-- 範例-select敘述-case when 2
select
    case
        when SAL >= 700 and SAL <= 1200
            then 'E'
        when SAL >= 1201 and SAL <= 1400
            then 'D'
        when SAL >= 1401 and SAL <= 2000
            then 'C'
        when SAL >= 2001 and SAL <= 3000
            then 'B'
        when SAL >= 3001 then 'A'
    end as SAL_GRADE -- ←一般會加上別名
from EMP
```

select敘述-Exercise01

01.撰寫一select敘述，查詢Table: DEPT，列出所有資料

02.撰寫一select敘述，查詢Table: EMP，列出所有員工的員工姓名(ENAME)、職稱(JOB)、到職日(HIREDATE)、及員工編號(EMPNO)，員工編號需顯示在第一欄

03.撰寫一select敘述，查詢Table: EMP，列出所有到職日(HIREDATE) ，同樣的到職日不重複顯示

04.續第02題，將select敘述加上別名..

ENAME→EmployeeName

JOB→Title

HIREDATE→HireDate

EMPNO→EmployeeNo

05.撰寫一select敘述，查詢Table: EMP，列出員工姓名(ENAME)串接職稱(JOB)，中間用逗號和空白隔開(', ')，並加上別名 NAMEandTITLE

select敘述-where子句 (1/6)

- 語法

```
select
      ..
from
      表格名
where
      條件
```

- 說明
  - 條件(Conditions): 過濾資料的條件，可以是多個任何結果為真(true)、假(false)、空值(null)的運算式，EX. 比較運算式、邏輯運算式、SQL特定運算式

select敘述-where子句 (2/6)

- 範例

```
-- 範例-select敘述-where子句1
-- 列出部門編號20且獎金小於500的員工
select
    *
from
    EMP
where
    DEPTNO = 20
    and COMM < 500
```

select敘述-where子句 (3/6)

- 比較運算子(Comparison Operators)
  - 說明: 比較兩個值的大小
  - *註: (預設) 字串會比較其內碼，但不區分大小寫

Comparison Operators	
Operator	**Meaning**
=	等於(Equal to)
>	大於(Greater than)
>=	大於或等於(Greater than or equal to)
<	小於(Less than)
<=	小於或等於(Less than or equal to)
<>	不等於(Not equal to)
!=	

select敘述-where子句 (4/6)

- 邏輯運算子(Logical Operators)
  - 說明: 將多個條件合成一個結果

Logical Operators	
Operator	**Meaning**
and	且(Returns TRUE if **both** conditions are TRUE)
or	或(Returns TRUE if **either** conditions is TRUE)
not	反向(Returns TRUE if condition is FALSE)

select敘述-where子句 (5/6)

- SQL特定運算子
  - 說明: SQL定義的強大運算子

SQL Own Operators		
Operator	**Meaning**	**Remark**
between 值1 and 值2	介於或等於(Between two values(inclusive))	等同於 >= 值1 and <= 值2
in(值1, .., 值N)	任一(Match any of a list of values)	等同於 =值1 or .. or =值N
like	模糊比對(Match a character pattern)	支援萬用字元，如下頁表
is null	是null(Is a null value)	等同於 <=> null， 但不等同於 = null
rlike	支援Regular Expression的比對	請參考Regular Expression

select敘述-where子句 (6/6)

- like支援的萬用字元

like Wildcard Character		
Character	**Meaning**	**Example**
%	任何含有零或多個字元的字串	ENAME like '%w%'→ 含有w ENAME like 'w%' → w開頭 ENAME like '%w' → w結尾
_	任何單一字元	ENAME like '_w_' → 第2個字元為w，且ENAME長度為3 ENAME like '_w' → 第2個字元為w，且ENAME長度為2 ENAME like 'w_1%' → w開頭，第3字元為1
escape	指定跳脫字元 用來跳脫上述字元	TEXT like '100/%%' escape '/' → 100%開頭

select敘述-order by子句 (1/2)

- 語法

```
select ..
from ..
..
order by
    欄位名1|別名1|欄位序1|運算式1 [asc|desc],
    ..,
    欄位名N|別名N|欄位序N|運算式N [asc|desc]
```

*註
1.order by子句撰寫於limit子句之前
2.null值排序會被視為最小

- 說明
  - 欄位名: 任何從from子句可找到的欄位名。select子句未指定的欄位亦可使用
  - 別名: select子句指定的別名
  - 欄位序: select子句指定的欄位順序
  - 運算式: 任何合法運算式皆可， EX. SAL + IFNULL(COMM, 0)
  - asc/desc: 排序的方向(遞增/遞減)，預設為asc

select敘述-order by子句 (2/2)

- 範例

```
-- 範例-select敘述-order by子句1
select
    EMPNO,
    ENAME as NAME
from
    EMP
order by
    1 desc,                          -- 欄位序1，即EMPNO (遞減)
    NAME desc,                       -- 別名 (遞減)
    SAL + IFNULL(COMM, 0)            -- 運算式，且使用select子句未指定欄位
```

select敘述-Exercise02

請撰寫出select敘述..

01.列出薪資不介於1000到2000元的員工之姓名和薪資

02.列出到職日為1981年的員工之姓名、職稱、到職日，並依到職日遞減排序

03.列出薪資超過2000元 且 部門編號為10或30 的員工之姓名、薪資，並依序取別名為 "EMPLOYEE_NAME"、"SALARY"

04.列出有獎金(獎金 不是null，也不是0)的員工之姓名、薪資、獎金，並排序，排序依據為薪資加上獎金

05.列出員工姓名最後一個字是"S"的員工之姓名、職稱

06.列出職稱為CLERK、SALESMAN，且薪資不等於1100、1300、1500的員工之姓名、職稱、薪資

07.列出獎金大於薪資1.05倍的員工之姓名、薪資、獎金

select敘述-group by子句 (1/13)

- 語法

```
select ..
from ..
[where ..]
group by
    欄位名1|欄位序1|運算式1,
    ..,
    欄位名N|欄位序N|運算式N,
    [with rollup]
```

- 說明
  - 欄位名: select子句指定的非聚合函數欄位
  - 欄位序: select子句指定的非聚合函數欄位順序
  - 運算式: select子句指定的非聚合函數運算式
  - with rollup: 在後面加上一筆總計或小計(多個分組依據時)的資料

select敘述-group by子句 (2/13)

- 範例

```
-- 範例-select敘述-group by子句1
-- 每個部門，每數量的獎金，各有多少人領取
select
    DEPTNO,
    IFNULL(COMM, 0) as COMM,
    COUNT(*) as COUNT
from
    EMP
group by
    DEPTNO,          -- 欄位名
    2                -- 欄位序2，即IFNULL(COMM, 0)
```

*註: group by後到底應該寫甚麼?

select子句所指定的欄位，去掉聚合函數欄位，其餘都複製到group by子句。

多數狀況下只要這樣寫即可!

EX. (上例)select子句指定了3個欄位，第1、2欄位非聚合函數: DEPTNO、IFNULL(COMM, 0)，

直接複製至group by子句即可

select敘述-group by子句 (3/13)

- 內建函數(Functions) (1/2)
  - 說明

    MySQL定義了許多副程式，供我們使用，稱之為內建函數
  - 以用到資料列數量分類
    - *註: 以單次運算，用到的資料列分類
    - 單一資料列函數(Single-Row Functions)

      每筆資料列各自執行一次，可用在select、where、group by、order by等子句
    - 多重資料列函數(Multiple-Row Functions)

      多筆資料列一起執行一次，可用在select 、group by 、having、order by等子句

select敘述-group by子句 (4/13)

- 內建函數(Functions) (2/2)
  - 以用途分類
    - 字串函數(String Functions)
    - 數值函數(Numeric Functions)
    - 日期時間函數(Date and Time Functions)
    - 資料型態轉換函數(Conversion Functions)
    - 通用函數(General Functions)
    - 群組/聚合函數(Group/Aggregate Functions) → 多重資料列函數

select敘述-group by子句 (5/13)

- 字串函數(String Functions) (1/2)

String Functions		
Function	**Meaning**	**Example**
LENGTH(str)	回傳str占用的**Byte數**。 中文占N Bytes，N取決於編碼 EX. UTF-8的一中文字占用3Bytes	LENGTH('a') -- 1 LENGTH('李') -- 3 (UTF-8)
CHAR_LENGTH(str)	回傳str的**字元數**	CHAR_LENGTH('a') -- 1 CHAR_LENGTH('李') -- 1 CHAR_LENGTH('william李') -- 8
LCASE(str)|LOWER(str)	回傳str**轉小寫**的結果	
UCASE(str)|UPPER(str)	回傳str**轉大寫**的結果	
ASCII(char)	回傳char對應的**ASCII碼** 若參數為多個字元，則只處理第一個字元	ASCII('a') -- 97
CONCAT(str1, .., strN)	回傳所有參數**串接**的結果	CONCAT('a', 'b', 'c') -- abc CONCAT('a', 'b', null) -- null
FIELD(str, str1, .., strN)	回傳str在之後的參數(str1, .., strN)中，**出現的位置** 從1開始，找不到則回傳0	FIELD('b', 'a', 'b', 'c') -- 2
INSERT(str, pos, len, newStr)	回傳str從pos開始len個字元，用newStr**取代**的結果	INSERT('abcde', 2, 3, '#') -- a#e
LEFT(str, len)	回傳str**最左**len個字元	LEFT('abcde', 3) -- abc
RIGHT(str, len)	回傳str**最右**len個字元	RIGHT('abcde', 3) -- cde
REVERSE(str)	回傳字串**反轉**的結果	REVERSE('abcde') -- edcba
LPAD(str, len, padStr)	回傳用padStr**左補**str至len個字元的結果 回傳結果長度只會有len個字元	LPAD('abcde', 7, '#') -- ##abcde LPAD('abcde', 4, '#') -- abcd
RPAD(str, len, padStr)	回傳用padStr**右補**str至len個字元的結果 回傳結果長度只會有len個字元	RPAD('abcde', 7, '#') -- abcde## RPAD('abcde', 4, '#') -- abcd

select敘述-group by子句 (6/13)

- 字串函數(String Functions) (2/2)

String Functions		
Function	**Meaning**	**Example**
SUBSTRING(str, pos[, len]) SUBSTRING(str from pos[for len])	回傳str從pos開始取len個字元的**子字串** 若無指定len則取到最後一字元 pos可指定負值，表示從右邊開始算	SUBSTRING('abcde', 3) -- cde SUBSTRING('abcde' from 3 for 2) -- cd SUBSTRING('abcde', -3, 1) -- c
REPEAT(str, count)	回傳str**重複**count次的結果	REPEAT('ba', 3) -- bababa
SPACE(count)	回傳count個**空白**	CHAR_LENGTH(SPACE(5)) -- 5
INSTR(str, subStr) LOCATE(subStr, str[, pos])	回傳subStr在str的第pos開始，第1次**出現的位置** 若無指定pos則從最前面開始找	INSTR('william', 'l') -- 3 LOCATE('l', 'william') -- 3 LOCATE('l', 'william', 4) -- 4
REPLACE(str, oldStr, newStr)	回傳str內用newStr**取代所有**oldStr的結果	REPLACE('william', 'l', 'L') -- wiLLiam
LTRIM(str)	回傳str**去除左**側空白的結果	LTRIM(' a') --a
RTRIM(str)	回傳str**去除右**側空白的結果	RTRIM('a ') --a
TRIM([BOTH|LEADING|TRAILING[remStr] FROM] str) TRIM([remStr FROM] str)	回傳str**去除**remStr的結果 BOTH: 前後都去除remStr(預設) LEADING: 只去除左側remStr TRAILING: 只去除右側remStr remStr: 欲去除的字串，預設為空白	TRIM(' a ') --a TRIM(TRAILING '#' from '#a#') -- #a
STRCMP(str1, str2)	回傳str1與str2**比較**結果 0: str1 = str2 1: str1 > str2 -1: str1 < str2	STRCMP('a', 'a') -- 0 STRCMP('a', 'b') -- -1 STRCMP('b', 'a') -- 1

select敘述-group by子句 (7/13)

- 數值函數(Numeric Functions)

Numeric Functions		
Function	**Meaning**	**Example**
ROUND(X[, D])	回傳X**四捨五入**到小數點第D位的結果 D預設為0	ROUND(10.5) -- 11 ROUND(10.5, 1) -- 10.5 ROUND(10.55, 1) -- 10.6
TRUNCATE(X, D)	回傳X**無條件捨去**至小數點第D位的結果	TRUNCATE(123.456, 0) -- 123 TRUNCATE(123.456, 1) -- 123.4 TRUNCATE(-123.456, 2) -- -123.45
MOD(N, M)	回N除以M的**餘數**	MOD(3, 2) -- 1 MOD(-10, 3) -- -1
CEIL(X)	回傳不小於X的**最小整數**	CEIL(123.456) -- 124 CEIL(-123.456) -- -123
FLOOR(X)	回傳不大於X的**最大整數**	FLOOR(123.456) -- 123 FLOOR(-123.456) -- -124
POWER(X, Y)	回傳X的Y**次方**	POWER(2, 4) -- 16 POWER(-2, 5) -- -32
SQRT(X)	回傳X**平方根**。X >= 0	SQRT(4) -- 2，SQRT(-1) -- null
ABS(X)	回傳X**絕對值**	ABS(-1) --1
SIGN(X)	回傳**符號**代表數字 0: 零，1: 正，-1: 負	SIGN(-1) -- -1，SIGN(1) -- 1 SIGN(0) -- 0
RAND([N])	回傳**偽亂數**。0<= 亂數 < 1 若有指定N，每次的RAND(N)都會取得一樣的亂數	RAND() RAND(3)
PI()	回傳**圓周率**	PI()

select敘述-group by子句 (8/13)

- 日期時間函數(Date and Time Functions) (1/3)

Date and Time Functions		
Function	**Meaning**	**Example**
CURDATE()	回傳**現在日期**	
CURTIME([fsp])	回傳**現在時間** fsp: 0~6，指定至1/fsp秒	
CURRENT_TIMESTAMP CURRENT_TIMESTAMP([fsp]) NOW()	回傳**現在日期時間** fsp: 0~6，指定顯示至1/fsp秒	CURRENT_TIMESTAMP -- 2020-05-04 10:08:29 CURRENT_TIMESTAMP(6) -- 2020-05-04 10:08:29.038902 NOW() -- 2020-05-04 10:08:29
UTC_DATE UTC_DATE()	回傳**現在UTC日期**	
UTC_TIME([fsp])	回傳**現在UTC時間** fsp: 0~6，指定至1/fsp秒	
UTC_TIMESTAMP UTC_TIMESTAMP([fsp])	回傳**現在UTC日期時間** fsp: 0~6，指定至1/fsp秒	
YEAR(date)	回傳date的**年**	YEAR('2020-05-04 10:08:29') -- 2020
MONTH(date)	回傳date的**月**	MONTH('2020-05-04 10:08:29') -- 5
DAY(date)	回傳date的**日**	DAY('2020-05-04 10:08:29') -- 4
HOUR(time)	回傳time的**時**	HOUR('2020-05-04 10:08:29') -- 10
MINUTE(time)	回傳time的**分**	MINUTE('2020-05-04 10:08:29') -- 8
SECOND(time)	回傳time的**秒**	SECOND('2020-05-04 10:08:29') -- 29

select敘述-group by子句 (9/13)

- 日期時間函數(Date and Time Functions) (2/3)

Date and Time Functions		
Function	**Meaning**	**Example**
TIME(expr)	回傳expr的**時間**	TIME('2020-05-04 10:08:29') -- 10:08:29 TIME('10:08:29') -- 10:08:29
MICROSECOND(expr)	回傳expr的**微秒**	MICROSECOND('2020-05-04 10:08:29.038902') -- -- 38902
EXTRACT(unit from date)	回傳date的**unit部分** unit: 欲取出的部分，請參考右側連結	https://dev.mysql.com/doc/refman/8.0/en/expressions.html#temporal-intervals
DAYNAME(date)	回傳date的**星期之英文**全名	DAYNAME('2020-05-04 10:08:29') -- Monday
MONTHNAME(date)	回傳date的**月份之英文**全名	MONTHNAME('2020-05-04 10:08:29') -- May
DAYOFWEEK(date)	回傳date是**本周的第幾天** 1: 星期日	DAYOFWEEK('2020-05-04 10:08:29') -- 2
DAYOFMONTH(date)	回傳date是**本月的第幾天**	DAYOFMONTH('2020-05-04 10:08:29') -- 4
DAYOFYEAR(date)	回傳date是**本年的第幾天**	DAYOFYEAR('2020-05-04 10:08:29') -- 125
WEEK(date[, mode]) WEEKOFYEAR(date)	回傳以mode模式表示date是**本年的第幾周** mode: 模式，請參考右側連線 WEEKOFYEAR(date)等同WEEK(date, 3)	https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_week
WEEKDAY(date)	回傳date的**星期索引** 0: 星期一	WEEKDAY('2020-05-04 10:08:29') -- 0
YEARWEEK(date[, mode])	回傳以mode模式表示date的**年度跟本年的第幾周** mode: 同WEEK()的mode	YEARWEEK('2020-05-04 10:08:29') -- 202018

select敘述-group by子句 (10/13)

- 日期時間函數(Date and Time Functions) (3/3)

Date/Time Functions		
Function	**Meaning**	**Example**
DATEDIFF(expr1, expr2)	回傳expr1與expr2**相差的日數** 只有日期部分會被拿來運算	DATEDIFF('2020-05-04', '2020-05-03 23:59:59') -- 1
ADDDATE(expr, days) ADDDATE(date, INTERVAL expr unit) DATE_ADD(date, INTERVAL expr unit)	1.回傳expr**加上**days**天**的結果 2、3.回傳date**加上unit單位的量**之結果 unit: 請參考右側連結	https://dev.mysql.com/doc/refman/8.0/en/expressions.html#temporal-intervals
SUBDATE(expr, days) SUBDATE(date, INTERVAL expr unit) DATE_SUB(date, INTERVAL expr unit)	1.回傳expr**減去**days**天**的結果 2、3.回傳date**減去unit單位的量**之結果 unit: ADDDATE()的unit	SUBDATE('2020-05-04', 1) -- 2020-05-03 SUBDATE('2020-05-04', interval 3 day) -- 2020-05-01
ADDTIME(expr1, time)	回傳expr1**加上時間量**time之結果	ADDTIME('2020-05-04 01:00:00', '12:34:56') -- 2020-05-04 13:34:56
SUBTIME(expr1, expr2)	回傳expr1**減去時間量**time之結果	SUBTIME('2020-05-04 12:34:56', '01:00:00') -- 2020-05-04 11:34:56
TIMEDIFF(expr1, expr2)	回傳expr1減去expr2的**時間差**之結果 expr1、expr2必須同時是time或date-time	TIMEDIFF('13:34:56', '12:34:56') -- 01:00:00
TIMESTAMP(expr1, expr2) TIMESTAMPADD(unit, interval, datetime_expr)	1.回傳expr1**加上**expr2的**日期時間**之結果 2.回傳datetime_expr**加上interval個unit單位**之結果 unit: MICROSECOND、SECOND、MINUTE、HOUR、DAY、WEEK、MONTH、QUARTER、YEAR	TIMESTAMP('2020-05-04 12:00:01', '12:00:00') -- 2020-05-05 00:00:01 TIMESTAMPADD(WEEK, 1, '2020-05-04') -- 2020-05-11
TIMESTAMPDIFF(unit, datetime_expr1, datetime_expr2)	回傳datetime_expr2**減去**datetime_expr1的日期時間之結果 datetime_expr1、datetime_expr2必須同時是日期或日期時間 unit: 同TIMESTAMPADD()的unit	TIMESTAMPDIFF(day,'2020-05-01','2020-05-04 12:34:56') -- 3
PERIOD_DIFF(P1, P2)	回傳P1**減去**P2的**月份數** P1、P2: 可以是數字或字串型態，格式:YYYYMM、YYMM	PERIOD_DIFF('202002', '2001') -- 1

select敘述-group by子句 (11/13)

- 資料型態轉換函數(Conversion Functions)

Conversion Functions		
Function	**Meaning**	**Example**
CAST(expr as type)	回傳expr**轉成type型態**之結果 type: BINARY、CHAR、DATETIME、DATE、TIME、SIGNED [INTEGER]、UNSIGNED [INTEGER]	CAST('2020-05-04' as DATE)
CONVERT(expr, type) CONVERT(expr USING transcoding_name)	1.回傳expr**轉成type型態**之結果 2.回傳expr**轉成transcoding_name編碼**的結果 type: 同CAST()的type 與CAST()功能相同，但建議使用CONVERT() (多數DBMS有此函數)	CONVERT('2020-05-04', DATE) CONVERT('william' USING utf8mb4)
TIMESTAMP(expr)	回傳expr**轉成日期時間**的結果	TIMESTAMP('2020-05-04') -- 2020-05-04 00:00:00

select敘述-group by子句 (12/13)

- 通用函數(General Functions)
  - 系統資訊函數(Information Functions)

Information Functions	
Function	**Meaning**
USER()	回傳當前**使用者**
VERSION()	回傳當前**資料庫版本**
DATABASE()	回傳當前**資料庫**
CONNECTION_ID()	回傳當前**連線的Conection ID** Conection ID依連入順序，由系統指定
CHARSET(str)	回傳str所屬的**字元集**

  - 流程控制函數(Control Flow Functions)

Control Flow Functions		
Function	**Meaning**	**Example**
IFNULL(expr1, expr2)	若**expr1等於null**，回傳expr2，否則回傳expr1	IFNULL('William', 'Guest') -- William IFNULL(null, 'Guest') -- Guest
IF(expr1, expr2, expr3)	若**expr1為true**，回傳expr2，否則回傳expr3 true的定義: 不等於0 且 不等於null	IF(2 > 1, 'Greater', 'Less or Equal')
NULLIF(expr1, expr2)	若**expr1等於expr2**，回傳null，否則回傳expr1	NULLIF(1, 1) -- null NULLIF(1, 2) -- 1

select敘述-group by子句 (13/13)

- 群組/聚合函數(Group/Aggregate Functions)
  - *註: 以下函數皆可與group by子句一起使用

Group/Aggregate Functions		
Function	**Meaning**	**Example**
COUNT(*)	回傳資料**總筆數**	
COUNT(expr)	回傳expr**不為null的筆數**	COUNT(ENAME)
COUNT(distinct expr)	回傳expr**去除重複後的筆數**	COUNT(distinct JOB)
MAX(expr)	回傳expr的**最大值**	MAX(SAL)
MIN(expr)	回傳expr的**最小值**	MIN(COMM)
SUM(expr)	回傳expr的**總和值**	SUM(SAL)
AVG(expr)	回傳expr的**平均值**	AVG(SAL)
AVG(distinct expr)	回傳expr**去除重複後的平均值**	AVG(distinct SAL)

select敘述-Exercise03

請撰寫出select敘述..

01.請列出公司每年需發出薪資總和 (不含獎金)

02.請列出公司平均月薪

03.請列出各部門編號、部門每月發出薪資總和，並依部門編號遞增排序

04.請列出職稱、該職稱平均薪資、該職稱人數

05.請列出部門編號、部門最低薪資、部門最高薪資

06.請列出到職年、到職年當年人數

select敘述-having子句 (1/2)

- 語法

```
select ..
from ..
[where ..]
group by ..
having
     條件
```

- 說明
  - 條件(Conditions): 同where子句一樣是過濾資料的條件，不過是針對群組/聚合函數運算出的結果做過濾，EX. COUNT(*) > 10

select敘述-having子句 (2/2)

- 範例

```
-- 範例-select敘述-having子句1
-- 各部門人數，但只顯示人數超過3人的部門
select DEPTNO, COUNT(*) as COUNT
from EMP
group by DEPTNO
having
    COUNT(*) > 3
```

select敘述-Exercise04

請撰寫出select敘述..

01.請列出部門編號、部門平均薪資，只顯示部門平均薪資大於2500的部門

02.請列出到職年、到職年當年人數，只顯示到職年當年人數等於1的資料

03.請列出各部門編號、部門每月發出薪資總和，只顯示部門每月發出薪資總和小於10000的部門，並依部門編號遞減排序

04.請列出職稱、該職稱平均薪資、該職稱人數，只顯示職稱平均薪資大於2000且職稱人數小於2的資料

05.請列出部門編號、部門最低薪資、部門最高薪資，且過濾掉最低薪資大於1200的資料

select敘述-from子句 (1/8)

- 簡述
  - from子句用以指定資料的來源
  - 若只有一個來源，寫法極為單純，但若來源有多個，需用join on語法撰寫
- 語法

```
select ..
from
    資料來源1 [[as] 別名1]
    [連結方式] join 資料來源2 [[as] 別名2]
        [on 條件]
    ..
```

select敘述-from子句 (2/8)

- 說明
  - 資料來源: select敘述的資料來源，可以是Table、View、子查詢
  - 別名: 資料來源的別名
  - 條件: 連結的依據條件

- 連結方式

- 交叉連結(Cross Join): 無條件連結
- 內部連結(Inner Join)
  - 自然連結(Natural Join): 依2資料來源 的 相同的欄位名連結
  - 相等連結(Equal Join): on語法後有等於運算子
  - 不相等連結(Non-Equal Join): on語法後無等於運算子

- 外部連結(Outer Join)
  - 左側連結(Left Join): 依左側資料來源為主
  - 右側連結(Right Join): 依右側資料來源為主
  - 完全連結(Full Join): Left Join的結果 加上 Right Join的結果 (MySQL不支援)
- 自我連結(Self Join): 2個資料來源是同一個物件或子查詢

select敘述-from子句 (3/8)

- 交叉連結

(Cross Join)

```
-- 範例-select敘述-from子句-cross join
-- Cross Join: 無條件連結
-- 連結後的資料筆數: EMP的筆數 * DEPT的筆數
-- 應用: 產生測試資料
select *
from
    EMP
    cross join DEPT
```

- 自然連結

(Natural Join)

```
-- 範例-select敘述-from子句-natural join
-- Natural Join: 相同的欄位名連結
-- Table EMP、DEPT各有一欄位 DEPTNO，依此連結
select *
from
    EMP
    natural join DEPT
```

select敘述-from子句 (4/8)

- 相等連結(Equal Join)

```
-- 範例-select敘述-from子句-equal join
-- Equal Join: on語法後有等於運算子
-- 若有一DEPTNO之值只在Table EMP或DEPT出現，
-- 連結後該筆資料不會出現!!
select
    e.ENAME,
    d.DNAME
from
    EMP e
    join DEPT d
        on e.DEPTNO = d.DEPTNO
```

select敘述-from子句 (5/8)

- 不相等連結(Non-Equal Join)

```
-- 範例-select敘述-from子句-non-equal join
-- Non-Equal Join: on語法後無等於運算子
-- 用Table EMP跟SAL_LEVEL(薪水等級表)連結
-- 連結條件為 員工薪水 介於 最低薪水 和 最高薪水
select
    e.ENAME,
    e.SAL,
    sl.LEVEL
from
    EMP e
    join SAL_LEVEL sl
        on e.SAL between sl.SAL_MIN
            and sl.SAL_MAX
```

select敘述-from子句 (6/8)

- 左側連結(Left Join)

```
-- 範例-select敘述-from子句-left join
-- Left Join: 依左側資料來源為主
-- 若有一DEPTNO之值只在Table EMP出現，
-- 連結後該筆資料仍然會出現，只是DNAME會是null
select
    e.ENAME,
    d.DNAME
from
    DEPT d
    left join EMP e
        on d.DEPTNO = e.DEPTNO
```

select敘述-from子句 (7/8)

- 右側連結(Right Join)

```
-- 範例-select敘述-from子句-right join
-- Right Join: 依右側資料來源為主
-- 同左側連結邏輯
-- 只是換成依寫在join右側的Table為主
-- 以下SQL意義同前一範例
select
    e.ENAME,
    d.DNAME
from
    EMP e
    right join DEPT d
        on e.DEPTNO = d.DEPTNO
```

select敘述-from子句 (8/8)

- 自我連結(Self Join)

```
-- 範例-select敘述-from子句-self join
-- Self Join:資料來源1 跟 資料來源2 是 同一個物件或子查詢
-- 列出員工姓名 和 主管姓名
select
    e1.ENAME,
    e2.ENAME as MGR_NAME
from
    EMP e1
    left join EMP e2
        on e1.MGR = e2.EMPNO
```

select敘述-Exercise05

請撰寫出select敘述..

01.請列出所有員工的員工編號、姓名、職稱、部門編號及部門名稱

02.請列出所有部門編號為30 且 職稱為"SALESMAN"之部門名稱、員工姓名、獎金

03.請列出薪水等級為"B"的員工之員工編號、員工姓名、薪資

04.請列出部門編號、部門名稱及部門人數

05.請列出每個主管之姓名、直屬下屬人數、直屬下屬平均薪資，並依 直屬下屬人數遞減、主管姓名遞增 排序

select敘述-limit子句 (1/2)

- 語法

```
select ..
from ..
[where ..]
[group by ..]
[having]
[order by ..]
limit
    [略過筆數,] 顯示筆數
```

*註

1.寫在所有子句的最後

2.控制顯示筆數有時稱為"分頁"

3.只有少數DBMS有支援limit子句，MySQL是其中之一

4.沒支援limit子句的DBMS有其他寫法, EX..

Oracle: ROWNUM隱含欄位或是offset子句

MS. SQL Server: top (寫在select子句中)

- 說明
  - 略過筆數(Offset): (從最前面開始算) 欲跳過的資料筆數，需>=0，預設為0
  - 顯示筆數(Count): 欲顯示的資料筆數，需>=0

select敘述-limit子句 (2/2)

- 範例

```
-- 範例-select敘述-limit子句1
-- 顯示全公司薪資前三高的員工，但過濾掉老闆
select ENAME, SAL
from EMP
order by SAL desc
limit
    1, 3
```

```
-- 範例-select敘述-limit子句2
-- 產生流水號
-- (續上例) 在最前面新增一欄位"名次"(流水號)
set @seq := 0;
select
    (@seq := @seq + 1) as '名次',
    ENAME, SAL
from EMP
order by SAL desc
limit 1, 3;
```

select敘述-子查詢 (1/9)

- 說明
  - 有時候需要先透過select敘述取得一個結果，再用此結果去找出我們要的資料
  - 例如 列出薪資低於平均薪資的員工..
    - 步驟1: 找出平均薪資
    - 步驟2: 用步驟1找出的平均薪資過濾員工資料
  - 要將以上兩步驟寫成一個敘述，就必須用子查詢(SubQuery)
- 範例

```
-- 範例-子查詢1-簡單子查詢
-- *註: 灰底部分為子查詢，其他部分為主查詢
select
    *
from
    EMP
where
    SAL < (select AVG(SAL) from EMP)
```

select敘述-子查詢 (2/9)

- 種類
  - 依查詢結果分類
    - 單欄單筆 (又稱純量(Scalar)) (如前頁範例)
    - 單欄多筆 (又稱多值(Multi-valued))
    - 多欄單筆
    - 多欄多筆 (又稱表格值(Table-valued))
  - 依執行方式分類
    - 自主子查詢(Self-contained subquery): 可獨立執行，不受主查詢影響 (如前頁範例)
    - 相關子查詢(Correlated subquery): 不可獨立執行，需依賴主查詢的資料

*註: 效能極差，應盡量改用Join取代

select敘述-子查詢 (3/9)

- 使用細節
  - 使用位置
    - select敘述的select子句(較少使用)，from子句和where、having子句
    - DML的敘述(insert、delete、update)
  - 區塊化: 需用圓括號包含
  - 可用比較運算子
    - 單筆: >、>=、<、<=、=、<>、!=
    - 多筆: in、any(任一)、all(所有)

select敘述-子查詢 (4/9)

- 單欄多筆子查詢

```
-- 範例-子查詢2-單欄多筆子查詢
-- 列出在NEW YORK上班的員工
select *
from EMP
where
    DEPTNO in (
        select
            DEPTNO
        from
            DEPT
        where
            LOC = 'NEW YORK')
```

select敘述-子查詢 (5/9)

- 多欄單筆子查詢

```
-- 範例-子查詢3-多欄單筆子查詢
-- 列出跟員工編號7499的員工，同職稱、同主管的員工
-- *註: 黃底部分稱為多重欄位條件(Multiple Column Condition)
--      MySQL有支援
select *
from EMP
where
      (JOB, MGR) = (
          select
              JOB,
              MGR
          from
              EMP
          where
              EMPNO = 7499)
```

select敘述-子查詢 (6/9)

- 多欄多筆子查詢

```
-- 範例-子查詢4-多欄多筆子查詢
-- 列出職稱跟薪水有重複的員工
select *
from EMP
where
      (JOB, SAL) in (
         select
              JOB,
              SAL
         from
              EMP
         group by
              JOB,
              SAL
         having
              COUNT(*) > 1)
```

select敘述-子查詢 (7/9)

- 相關子查詢(Correlated subquery)

```
-- 範例-子查詢5-相關子查詢
-- Correlated subquery: 不可獨立執行，需依賴主查詢的資料
-- 列出各部門薪資最高的員工
-- *註: 黃底為依賴主查詢部分
select DEPTNO, ENAME, SAL
from
    EMP t1
where
    SAL = (
        select MAX(SAL)
        from EMP
        where DEPTNO = t1.DEPTNO)
order by
    DEPTNO
```

select敘述-子查詢 (8/9)

- exists運算子 (1/2)
  - 說明
    - 用於得知子查詢結果有無資料，所以運算結果也只會是true(有)或false(無)
    - 一般會配合相關子查詢使用
    - 若用自主子查詢，主查詢的結果只會是沒有資料或全部資料
  - 語法

```
[not] exists (子查詢)
```

select敘述-子查詢 (9/9)

- exists運算子 (2/2)
  - 範例

```
-- 範例-子查詢6-exists運算子
-- 列出沒有員工的部門
select *
from DEPT d
where
    not exists (
        select *
        from EMP
        where DEPTNO = d.DEPTNO)
```

select敘述-子句順序

- 撰寫順序
  1. select
  2. from
  3. where
  4. group by
  5. having
  6. order by
  7. limit

- 執行順序
  1. from
  2. where
  3. group by
  4. having
  5. select
  6. order by
  7. limit

*註

1.子句執行的順序，

跟我們撰寫的順序差異很大

2.此順序也默默解釋了很多，

select敘述上的限制

3.記得這順序才能弄懂select敘述

select敘述-Exercise06

請撰寫出select敘述..

01.請列出薪資比所有SALESMAN還低的員工

02.請列出到職年(到職日之年)最早的兩年，那兩年到職的員工，並依到職日排序

03.請列出主管的主管是KING 的員工

04.請列出部門內員工薪資有3種薪資等級之部門名稱、部門所在地

05.請列出跟員工姓名最後一字元是S的員工同部門，該部門薪資最低的員工之部門名稱、姓名、薪資

DML

- 簡述
  - Data Manipulation Language，資料操作語言
  - 用以新增、刪除、修改Table內的資料
  - 執行完後可能會改變資料
- 敘述
  - insert: 新增資料至Table
  - delete: 刪除Table內的資料
  - update: 修改Table內的資料

DML-insert敘述 (1/4)

- 語法

```
insert into
      資料表名[(欄位名, ..)]
values(值, ..)
      [, (值, ..)]
```

- 說明
  - 資料表名: 欲新增資料的Table名
  - 欄位名: 欲新增資料的Column名。若省略不寫，表示全部Column。依序與後面的值對應
  - 值: 各Column的值，依序與前面的欄位名對應，值也須與對應欄位名的資料型態相同或相容

DML-insert敘述 (2/4)

- 範例

```
-- 範例-insert敘述1-新增一筆資料
insert into EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values(9999, 'William', 'Engineer', 7566, NOW(), 5500, 0, 20);
```

```
-- 範例-insert敘述2-新增多筆資料
insert into EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
values(9999, 'William', 'Engineer', 7566, NOW(), 5500, 0, 20),
      (8888, 'Vera', 'Engineer', 7566, NOW(), 4500, 0, 20);
```

DML-insert敘述 (3/4)

- 變化型 (1/2)
  - 簡述

    有時候我們想要新增的資料來自別的Table，此時可以用子查詢來完成

  - 語法

```
insert into 資料表名[(欄位名, ..)]
子查詢
```

  - 說明
    - 子查詢:欲新增的資料來源。欄位數需與insert描述的欄位數一樣，

      且資料型態亦須與對應欄位名的資料型態相同或相容

DML-insert敘述 (4/4)

- 變化型 (2/2)
  - 範例

```
-- 範例-insert敘述3-變化型寫法
insert into EMP(EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)
select 7777, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO
from EMP
where EMPNO = 9999
```

DML-delete敘述 (1/3)

- 語法

```
delete from 資料表名
where 條件
```

- 說明
  - 資料表名:欲刪除資料的Table名
  - 條件:欲刪除的資料之條件。實務上一定會寫條件

- 範例

```
-- 範例-delete敘述1-標準寫法
-- 刪除EMPNO為7777的那一筆資料
delete from EMP where EMPNO = 7777;
```

DML-delete敘述 (2/3)

- 變化型 (1/2)
  - 簡述

    有時候我們想要刪除資料的條件來自別的Table，此時可以用子查詢來完成

  - 語法

```
delete from 資料表名
where 子查詢
```

  - 說明
    - 子查詢:刪除資料的條件

DML-delete敘述 (3/3)

- 變化型 (2/2)
  - 範例

```
-- 範例-delete敘述2-變化型寫法
-- 刪除在部門名稱為'SALES'工作的所有員工
delete from EMP where DEPTNO = (
    select DEPTNO
    from DEPT
    where DNAME = 'SALES')
```

DML-update敘述 (1/4)

- 語法

```
update 資料表名
set
      欄位名1 = 值1,
      欄位名2 = 值2,
      ..
where
      條件
```

- 說明
  - 資料表名:欲修改資料的Table名
  - 欄位名: 欲修改資料的Column名
  - 值:欲修改(更新)的值
  - 條件:欲修改的資料之條件，若不寫則會修改全部資料。實務上大多會寫條件

DML-update敘述 (2/4)

- 範例

```
-- 範例-update敘述1-基本寫法
-- 修改員工編號為9999的員工，員工姓名改為William Lee、薪水加500
update
    EMP
set
    ENAME = 'William Lee',
    SAL = SAL + 500
where
    EMPNO = 9999;
```

DML-update敘述 (3/4)

- 變化型 (1/2)
  - 簡述

    有時候我們想要修改資料的值或條件來自別的Table，此時可以用子查詢來完成

  - 語法

```
update 資料表名 set 欄位名 = 子查詢 where 條件
```

```
update 資料表名 set 欄位名 = 值 where 子查詢
```

  - 說明
    - 子查詢

      用在set子句:資料值的來源

      用在where子句:條件來源

DML-update敘述 (4/4)

- 變化型 (2/2)
  - 範例

```
-- 範例-update敘述2-變化型寫法-用在set子句
-- 將部門編號10的所有員工，調至ACCOUNTING部門
update EMP
set DEPTNO = (
        select DEPTNO
        from DEPT
        where DNAME = 'ACCOUNTING')
where DEPTNO = 10;
```

```
-- 範例-update敘述3-變化型寫法-用在where子句
-- 銷售部的所有員工加薪500
update EMP
set SAL = SAL + 500
where DEPTNO = (
    select DEPTNO
    from DEPT
    where DNAME = 'SALES')
```

DML-Exercise07

01.請新增以下資料至資料表DEPT

50, 'Software', 'Taipei'

02.請新增以下資料至資料表EMP的欄位 EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO

9999, 'William', 'PG', null, NOW(), 2500, 50

8888, 'Lee', 'PM', null, NOW(), 3500, 50

03.請修改資料表EMP的資料..

員工8888的主管改為7839

員工9999的主管改為8888

提示: 可配合case運算式，將2個敘述合併成1個敘述！

04.請刪除員工編號為8888的員工之資料

05.請修改資料表EMP的資料..

TCL

- 簡述
  - Transaction Control Language，交易控制語言
  - 用以控制資料庫中的交易
  - 交易內容是由多個SQL敘述所組成
  - 多個DML敘述在商業邏輯上，常控制成: 要就全部執行成功，否則就全部執行失敗(或不執行)
  - 成功時送交(Commit)，失敗時還原(Rollback)
  - MySQL只有InnoDB引擎支援交易控制
  - 交易控制開始時，DML敘述會產生列鎖定(Row Locking)

TCL

- 敘述
  - set autocommit: 啟用/停用交易控制模式。是全局設定，之後的所有敘述都會受影響
  - start transaction: 開啟單一交易控制。只有當前範圍內敘述會受影響
  - commit: 送交，配合start transaction使用時，會結束交易
  - rollback: 還原，配合start transaction使用時，會結束交易
  - savepoint: 設定一個儲存點。rollback可指定還原至儲存點

TCL-set autocommit敘述 (1/2)

- 語法

```
set autocommit = 值
```

- 說明
  - 值
    - 0 / off:啟用交易控制模式
      1 / on:停用交易控制模式(預設)

*註
1.變數autocommit的意義是自動送交，所以設為0或off才是啟用交易控制(手動控制)
2.set autocommit敘述只會影響當前連線，其他連線不會受影響
3.通常交易控制結束後，會馬上改回1或on，保持自動送交
4.可用右邊敘述查詢目前變數autocommit之值

```
select @@autocommit
```

TCL-set autocommit敘述 (2/2)

- 範例

```
-- 範例-set autocommit敘述1
-- 啟用交易控制模式
set autocommit = 0;
-- 查詢變數autocommit之值
select @@autocommit;
```

```
-- 範例-set autocommit敘述2
-- 停用交易控制模式
set autocommit = 1;
```

TCL-start transaction敘述

- 語法

```
start transaction
```

*註
1.配合commit/rollback使用
2.從start transaction至commit/rollback稱為一個單一交易
3.執行到commit/rollback時，就會結束此單一交易

- 範例

```
-- 範例-start transactio敘述
-- 開啟單一交易控制
start transaction;
```

TCL-commit敘述

- 語法

```
commit
```

```
*註
    1.需先執行set autocommit = 0或start transaction，commit敘述才有意義
    2.配合start transaction使用時，同時會結束交易
```

- 範例

```
-- 範例-commit敘述
-- 送交
commit;
```

TCL-rollback敘述

- 語法

```
rollback [to 儲存點識別名]
```

- 說明

  - 儲存點識別名

    欲還原的儲存點

    此儲存點是經由savepoint敘述所設定

*註
1.需先執行set autocommit = 0或start transaction，rollback敘述才有意義
2.配合start transaction使用時，同時會結束交易

- 範例

```
-- 範例-rollback敘述
-- 還原
rollback;
```

TCL-savepoint敘述

- 語法

```
savepoint 儲存點識別名
```

- 說明

*註: 配合rollback使用才有意義

  - 儲存點識別名

    儲存點的識別名，自訂但不可重複，且區分大小寫

    rollback敘述用此名稱還原至此點

- 範例

```
-- 範例-savepoint敘述
-- 設定一個儲存點，名為savepoint01
savepoint savepoint01;
```

TCL-完整範例

```
-- 將員工7499的獎金，分給員工7521 300元
-- 2個update敘述要就一起成功，否則就一起失敗
-- 方式一: 使用交易控制模式
set autocommit = off;
    update EMP set COMM = COMM - 300 where EMPNO = 7499;
    update EMP set COMM = COMM + 300 where EMPNO = 7521;
commit;
-- 設回預設值
set autocommit = on;

-- 方式二: 使用單一交易控制
start transaction;
    update EMP set COMM = COMM - 300 where EMPNO = 7499;
    update EMP set COMM = COMM + 300 where EMPNO = 7521;
commit;
```

TCL-完整範例-使用儲存點

```
-- 依序刪除部門編號10、20、30的所有員工
-- 每次刪除都加入一個儲存點
start transaction;
    delete from EMP where DEPTNO = 10;
    savepoint p1;
    delete from EMP where DEPTNO = 20;
    savepoint p2;
    delete from EMP where DEPTNO = 30;
rollback to p2;

/* rollback to p2: 還原至儲存點p2，表示還原了刪除部門編號30所有員工
                   部門編號10、20的所有員工還是會被刪除
   若改成rollback to p1，則只會刪除部門編號10的所有員工
*/
```

TCL-Exercise08

01.請啟用交易控制模式，執行以下動作..

刪除除了老闆以外的所有員工之資料

查詢確認是否已刪除

還原

02.請開啟單一交易控制，執行以下動作..

修改除了老闆以外的所有員工，獎金+1000、薪水+15%

查詢確認是否已修改

送交

03.今天公司空降了一位主管ERIC，員工編號: 6666，職稱: MANAGER，主管: 7839，

薪資: 3500，部門編號: 50

另外原本就在職的2位員工7499、7844調至部門編號50，且主管改為6666

請開啟單一交易控制，將上述動作在一個交易內完成

DDL

- 簡述
  - Data Definition Language，資料定義語言
  - 用以維護資料庫內的物件
  - 資料庫常見物件: Table、View、Index..等
- 敘述
  - create: 新建物件
  - drop: 移除物件
  - alter: 修改物件
  - truncate: 截斷物件

*註: 物件在講義內皆以table為例

DDL-create table敘述

- 語法

```
create table 資料表名(各欄位定義 和 各約束定義) [comment '資料表註解']
```

- 說明
  - 資料表名: 新建Table的名稱
  - 欄位定義: 各Column的描述，名稱、資料型態和長度、接受null否、預設值、註解..等
  - 約束定義: Table的PK、UK、FK..等。約束定義亦可寫在欄位定義段落，但通常會分開

*註: "欄位定義"段落也稱為Column Level，"約束定義"段落則稱為Table Level

DDL-create table敘述-欄位定義 (1/2)

- 語法

```
欄位名 資料型態(長度) [not null] [default 預設值] [comment '註解']
```

- 說明
  - 欄位名: 欄位的名稱
  - 資料型態: 欄位的資料型態
  - 長度: 欄位的長度
  - not null: 欄位不可放入null值
  - 預設值: (執行insert敘述時) 若未指定欄位值，自動放入的值
  - 註解: 欄位的註解

DDL-create table敘述-欄位定義 (2/2)

- 範例

```
-- 範例-欄位定義
create table MEMBER(
    ID int not null comment '編號',
    USERNAME varchar(50) not null comment '帳號',
    PASSWORD varchar(50) not null comment '密碼',
    NICKNAME varchar(50) not null comment '暱稱',
    CREATE_TIME datetime not null comment '建立日期時間',
    PASS bit(1) not null,
     ..約束定義
);
```

DDL-create table敘述-約束定義 (1/13)

- 主要鍵(Primary Key)
  - 寫在欄位定義段落(Column Level)
    - 語法

```
..欄位定義 primary key
```

    - 說明
      - 欄位定義: 寫在某個欄位定義之後，此欄位就會被設定成主要鍵
  - 寫在約束定義段落(Table Level)
    - 語法

```
primary key (欄位名1, .., 欄位名N)
```

    - 說明
      - 欄位名: 欲被設定成主要鍵的欄位名稱

*註

1.一個資料表，主要鍵只能有一個

2.主要鍵欄位不接受null值!

DDL-create table敘述-約束定義 (2/13)

- 範例

```
-- 範例-約束定義-建立主要鍵-寫在欄位定義段落(Column Level)
create table MEMBER(
    ID int not null comment '編號' primary key,
    ..略
);


-- 範例-約束定義-建立主要鍵-寫在約束定義段落(Table Level)
create table MEMBER(
    ID int not null comment '編號',
    ..略,
    primary key (ID)
);
```

DDL-create table敘述-約束定義 (3/13)

- 唯一鍵(Unique Key)
  - 寫在欄位定義段落(Column Level)
    - 語法

```
..欄位定義 unique
```

    - 說明
      - 欄位定義: 寫在某個欄位定義之後，此欄位就會被設定成唯一鍵
  - 寫在約束定義段落(Table Level)
    - 語法

```
unique key [唯一鍵名] (欄位名1, .., 欄位名N)
```

    - 說明
      - 唯一鍵名: 自訂的唯一鍵名稱。命名習慣:UK_資料表名_欄位名
      - 欄位名: 欲被設定成唯一鍵的欄位名稱

*註

1.一個資料表，唯一鍵可以有多個

2.唯一鍵欄位可以接受null值!

DDL-create table敘述-約束定義 (4/13)

- 範例

```
-- 範例-約束定義-建立唯一鍵-寫在欄位定義段落(Column Level)
create table MEMBER(
    USERNAME varchar(50) not null comment '帳號' unique,
    ..略
);


-- 範例-約束定義-建立唯一鍵-寫在約束定義段落(Table Level)
create table MEMBER(
    USERNAME varchar(50) not null comment '帳號',
    ..略,
    unique key UK_MEMBER_USERNAME(USERNAME)
);
```

DDL-create table敘述-約束定義 (5/13)

- 外來鍵(Foreign Key) (1/2)
  - 寫在欄位定義段落(Column Level)
    - 語法

```
..欄位定義 references 被參考資料表名(被參考欄位名)
```

    - 說明
      - 欄位定義: 寫在某個欄位定義之後，此欄位就會被設定成外來鍵
      - 被參考資料表名: 欲參考的資料表名稱
      - 被參考欄位名: 欲參考的欄位名稱

*註: MySQL中，在Column Level定義外來鍵，語法不會錯誤，但不會有任何作用
(測試版本: 8.0.19)

DDL-create table敘述-約束定義 (6/13)

- 外來鍵(Foreign Key) (2/2)
  - 寫在約束定義段落(Table Level)
    - 語法

*註: 被參考的欄位必須有索引(Index)

```
[constraint 外來鍵名] foreign key(欄位名1, .., 欄位名N)
references 被參考資料表名(被參考欄位名1, ..,被參考欄位名N)
```

    - 說明
      - 外來鍵名: 自訂的外來鍵名稱。命名習慣:FK_資料表名_欄位名
      - 欄位名: 欲被設定成外來鍵的欄位名稱
      - 被參考資料表名: 被參考的資料表名稱
      - 被參考欄位名: 被參考的欄位名稱

DDL-create table敘述-約束定義 (7/13)

- 範例

```
-- 範例-約束定義-建立外來鍵-寫在欄位定義段落(Column Level)
create table EMP(
    DEPTNO int not null references DEPT (DEPTNO),
    ..略
);

-- 範例-約束定義-建立外來鍵-寫在約束定義段落(Table Level)
create table EMP(
    DEPTNO int not null,
    ..略,
    constraint FK_EMP_DEPTNO foreign key (DEPTNO)
    references DEPT (DEPTNO)
);
```

DDL-create table敘述-約束定義 (8/13)

- on Update / on Delete
  - 外來鍵的進階設定
  - 當被參考端的被參考欄位之值，修改/刪除時的對應動作
  - 寫在欄位定義段落(Column Level)

```
..欄位定義 references 參考資料表名(參考欄位名) on update 動作
```

```
..欄位定義 references 參考資料表名(參考欄位名) on delete 動作
```

  - 寫在約束定義段落(Table Level)

```
..外來鍵定義 on update 動作
```

```
..外來鍵定義 on delete 動作
```

DDL-create table敘述-約束定義 (9/13)

- on Update / on Delete (續)
  - 說明
    - 動作: 對應動作，可以是以下4種之一
      - restrict(禁止): 被參考端禁止修改/刪除。預設的動作
      - cascade(連動): 參考端的資料列會連動被修改/刪除
      - set null (設為空值): 將參考端的值設成null
      - no action: 原意為無動作。但在MySQL中等同restrict
      - https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html

DDL-create table敘述-約束定義 (10/13)

- 範例

```
-- 範例-約束定義-建立外來鍵-On Update/ On Delete
create table EMP(
    DEPTNO int not null,
    ..略,
    constraint FK_EMP_DEPTNO foreign key (DEPTNO)
    references DEPT (DEPTNO) on update cascade
);
```

DDL-create table敘述-約束定義 (11/13)

- 檢查(Check) (1/2)
  - 說明
    - MySQL 8.0.16開始支援
    - 在MySQL較舊的版本中使用，不會有語法錯誤，但無任何作用!
  - 寫在欄位定義段落(Column Level)
    - 語法

```
..欄位定義 check(條件) [[not] enforced]
```

    - 說明
      - 欄位定義: 寫在某個欄位定義之後，就只能檢查此欄位
      - 條件: 欲檢查的條件，不符合時，無法儲存資料
      - [not] enforced: 是否強制，即啟用/停用。預設為啟用

DDL-create table敘述-約束定義 (12/13)

- 檢查(Check) (2/2)
  - 寫在約束定義段落(Table Level)
    - 語法

```
[constraint 檢查名] check(條件) [[not] enforced]
```

    - 說明
      - 檢查名: 自訂的檢查名稱。命名習慣:CK_資料表名_欄位名
      - 條件: 欲檢查的條件，不符合時，無法儲存資料
      - [not] enforced: 是否強制，即啟用/停用。預設為啟用
  - 加上not enforced的意義
    - 可先加入Check約束，但設成停用
    - 之後再使用alter table alter check敘述，啟用Check約束。語法在後面章節說明

DDL-create table敘述-約束定義 (13/13)

- 範例

```
-- 範例-約束定義-建立Check約束-寫在欄位定義段落(Column Level)
create table EMP(
    SAL decimal(7,2) comment '薪資' check(SAL > 0),
    ..略
);

-- 範例-約束定義-建立Check約束-寫在約束定義段落(Table Level)
create table EMP(
    SAL decimal(7,2) comment '薪資',
    ..略,
    constraint CK_EMP_SAL check (SAL > 0)
);
```

DDL-create table敘述-完整範例

```
create table EMP (
    EMPNO int not null comment '編號(主鍵)' primary key,
    ENAME varchar(10) comment '姓名',
    JOB varchar(9) comment '職稱',
    MGR int comment '主管編號',
    HIREDATE datetime comment '到職日',
    SAL decimal(7,2) comment '薪資',
    COMM decimal(7,2) comment '獎金',
    DEPTNO int not null comment '部門編號(外來鍵)',
    unique key UK_EMP_ENAME (ENAME),
    constraint FK_EMP_DEPTNO foreign key (DEPTNO) references DEPT (DEPTNO)
        on delete restrict
        on update cascade,
    constraint CK_EMP_SAL check (SAL > 0)
) comment '員工資料表';
```

DDL-create table敘述-其他 (1/4)

- auto_increment
  - 說明
    - 用在欄位定義上
    - 大部分數值資料型態可使用
    - 設定此欄位從1開始，由系統自動編號
    - insert資料時，應跳過此欄位
  - 語法

```
..(接在某欄位定義之後) auto_increment
```

DDL-create table敘述-其他 (2/4)

- 範例

```
-- 範例-欄位定義-auto_increment
create table MEMBER(
    ID int not null auto_increment comment '編號',
    ..略,
);
```

DDL-create table敘述-其他 (3/4)

- 變化型-複製某資料表的結構和資料
  - 語法

```
create table 資料表名 [as] 子查詢
```

  - 說明
    - 資料表名: 新建Table的名稱
    - select敘述: 欲複製的結構和資料
  - 範例

```
-- 範例-create table as select敘述
create table EMP20
as
select * from EMP where DEPTNO = 20;
```

DDL-create table敘述-其他 (4/4)

- 變化型-複製某資料表的結構
  - 語法

```
create table 資料表名 like 欲複製的資料表名
```

  - 說明
    - 資料表名: 新建Table的名稱
    - 欲複製的資料表名: 會複製此Table的結構
  - 範例

```
-- 範例-create table like敘述
create table EMP_BAK like EMP;
```

DDL-drop table敘述

- 語法

```
drop table 資料表名
```

- 說明
  - 資料表名: 欲移除的Table名稱

- 範例

```
-- 範例-drop table敘述
drop table EMP;
```

DDL-alter table敘述

- 語法

```
alter table 資料表名 修改定義1 [,修改定義2, .., 修改定義N]
```

- 說明
  - 資料表名: 欲修改Table的名稱
  - 修改定義: 欲修改的細節
  - 可針對以下對象，進行以下動作

    對象: 欄位(column)、約束(constraint)、資料表(table)

    動作: 加入(add)、移除(drop)、修改(alter、modify、change)

DDL-alter table敘述-針對欄位 (1/7)

- 加入一個欄位
  - 語法

```
add [column] 欄位定義 [first | after 前一個欄位名]
```

  - 說明
    - 欄位定義: 欲加入欄位的定義。語法同create table敘述的欄位定義
    - 前一個欄位名: 位置上前一個欄位的名稱，用以控制加入的位置

      EX. after EMPNO，表示加入的欄位會在EMPNO之後
  - 範例

```
-- 範例-alter table敘述-加入欄位
alter table EMP
add column GENDER char(1) after ENAME;
```

DDL-alter table敘述-針對欄位 (2/7)

- 加入多個欄位
  - 語法

```
add [column] (欄位定義1, .., 欄位定義N)
```

  - 說明
    - 欄位定義: 欲加入欄位的定義。語法同create table敘述的欄位定義
  - 範例

```
-- 範例-alter table敘述-加入多個欄位
alter table EMP
add column (EMAIL varchar(50), BIRTHDAY datetime);
```

DDL-alter table敘述-針對欄位 (3/7)

- 移除欄位
  - 語法

```
drop [column] 欄位名
```

  - 說明
    - 欄位名: 欲移除的欄位名稱
  - 範例

```
-- 範例-alter table敘述-移除欄位
alter table EMP
drop column BIRTHDAY;
```

DDL-alter table敘述-針對欄位 (4/7)

- 修改欄位定義
  - 語法

```
modify [column] 欄位定義 [first | after 前一個欄位名]
```

  - 說明
    - 欄位定義: 欲修改欄位的定義。語法同create table敘述的欄位定義
    - 前一個欄位名: 位置上前一個欄位的名稱，用以控制加入的位置

      EX. after EMPNO，表示修改的欄位會在EMPNO之後
  - 範例

```
-- 範例-alter table敘述-修改欄位定義
alter table EMP
modify column EMAIL varchar(100) not null default '' after ENAME;
```

DDL-alter table敘述-針對欄位 (5/7)

- 修改欄位名稱/資料型態(長度)
  - 語法

```
change [column] 欄位舊名 欄位新名 資料型態(長度)
```

  - 說明
    - 欄位舊名: 欲修改的欄位舊名稱
    - 欄位新名: 欲修改的欄位新名稱
    - 資料型態(長度): 欄位的型態和長度。語法同create table敘述的欄位定義
  - 範例

```
-- 範例-alter table敘述-修改欄位名稱
alter table EMP
change column EMAIL `E-MAIL` varchar(200);
```

DDL-alter table敘述-針對欄位 (6/7)

- 修改欄位預設值
  - 語法

```
alter [column] 欄位名{set default 預設值 | drop default}
```

  - 說明
    - 欄位名: 欲修改的欄位名稱
    - 預設值: 欲設定的預設值
    - drop default: 移除預設值

*註: 使用運算式當預設值

若要將預設值改為運算式，則須使用小括號將值包含，

EX.((NOW()))

DDL-alter table敘述-針對欄位 (7/7)

- 範例

```
-- 範例-alter table敘述-修改欄位預設值

-- 將欄位EMAIL的預設值改成'@'
alter table EMP
alter column EMAIL set default '@';

-- 將欄位EMAIL的預設值移除
alter table EMP
alter column EMAIL drop default;

-- 將欄位HIREDATE的預設值改成NOW()
alter table EMP
alter column HIREDATE set default (NOW());
```

DDL-alter table敘述-針對約束 (1/6)

- 加入約束
  - 語法

```
add [constraint 鍵名] 約束定義
```

  - 說明
    - 欄位名: 自訂的鍵名稱
    - 約束定義: 欲新稱的約束定義。語法同create table敘述的約束定義(Table Level寫法)

約束定義-主要鍵、約束定義-唯一鍵、約束定義-外來鍵、約束定義-檢查

DDL-alter table敘述-針對約束 (2/6)

- 範例

```
-- 範例-alter table敘述-加入約束

-- 加入主要鍵
alter table EMP add primary key (EMPNO);

-- 加入唯一鍵
alter table EMP add constraint UK_EMP_ENAME unique key (ENAME);

-- 加入外來鍵
alter table EMP
add constraint FK_EMP_DEPTNO foreign key (DEPTNO) references DEPT(DEPTNO);

-- 加入Check約束
alter table EMP add constraint CK_EMP_SAL check (SAL > 0);
```

DDL-alter table敘述-針對約束 (3/6)

- 移除約束
  - 語法

```
drop {primary key | constraint 約束名}
```

  - 說明
    - primary key: 移除主要鍵時使用。主要鍵只會有一個，所以不用指定鍵名
    - 約束名: 欲移除的約束名

DDL-alter table敘述-針對約束 (4/6)

- 範例

```
-- 範例-alter table敘述-移除約束

-- 移除主要鍵
alter table EMP drop primary key;

-- 移除唯一鍵
alter table EMP drop constraint UK_EMP_ENAME;

-- 移除外來鍵
alter table EMP drop constraint FK_EMP_DEPTNO;

-- 移除Check約束
alter table EMP drop constraint CK_EMP_SAL;
```

DDL-alter table敘述-針對約束 (5/6)

- 啟用/停用Check約束
  - 語法

```
alter constraint 鍵名 [not] enforced
```

  - 說明
    - 鍵名: 欲啟用/停用的Check約束鍵名

*註: 在create table敘述-約束定義章節有提過，可先加入Check約束，並設成停用

需要時，就可以使用此語法來啟用Check約束!

DDL-alter table敘述-針對約束 (6/6)

- 範例

```
-- 範例-alter table敘述-啟用/停用Check約束

-- 新建資料表EMP時，同時建立Check約束，但設成停用
create table EMP (
    ..略,
    SAL decimal(7,2) comment '薪資',
    ..略,
    constraint CK_EMP_SAL check (SAL > 0) not enforced
);

-- 啟用Check約束
alter table EMP alter constraint CK_EMP_SAL enforced;
```

DDL-alter table敘述-針對資料表 (1/2)

- 修改資料表名稱
  - 語法

```
rename [to] 新資料表名
```

  - 說明
    - 新資料表名: 資料表的新名稱
  - 範例

```
-- 範例-alter table敘述-修改資料表名稱
alter table EMP rename to EMPLOYEE;
```

DDL-alter table敘述-針對資料表 (2/2)

- 修改資料表儲存引擎
  - 語法

```
engine 儲存引擎名
```

  - 說明
    - 儲存引擎名: 新儲存引擎名稱。可使用以下敘述查詢有哪些儲存引擎可使用

```
show engines
```

  - 範例

```
-- 範例-alter table敘述-修改資料表儲存引擎
alter table EMP engine INNODB;
```

DDL-truncate table敘述

- 語法

```
truncate [table] 資料表名
```

- 說明
  - 資料表名: 欲截斷的Table名稱

- 範例

```
-- 範例-truncate table敘述
truncate table EMP;
```

*註
- 會徹底的清理資料表
- 底層動作
  1. 移除資料表(drop table)
  2. 新建資料表(create table)
- 資料會被刪除
- auto_increment的值會回到1

DDL-Exercise09 (1/2)

01.請新建 會員資料表MEMBER，欄位描述如下..

Column Name	Data Type	Length	Default	PK?	auto_increment	Comment
ID	int			Y	Y	編號
USERNAME	varchar	50				帳號
PASSWORD	varchar	50				密碼
PASS	bit	1	0			帳號開通記號
CREATE_DATE	datetime					建立日期
LAST_UPDATE_DATE	datetime					

02.請對資料表MEMBER的欄位PASSWORD之後加入一欄位..

Column Name	Data Type	Length	Default	PK?	auto_increment	Comment
NICKNAME	varchar	50				暱稱

03.請修改資料表MEMBER的欄位

USERNAME、PASSWORD: not null、長度100
LAST_UPDATE_DATE: 預設值NOW()、註解"最後修改日期"

04.請將資料表MEMBER的欄位USERNAME設為唯一鍵

05.請將資料表MEMBER的欄位USERNAME的唯一鍵約束移除

DDL-Exercise09 (2/2)

06.請將資料表MEMBER的欄位CREATE_DATE移除

07.請新建資料表 MEMBER2、MEMBER3，2個資料表的結構跟資料須與資料表MEMBER一樣

08.請將資料表MEMBER2改名為MEMBER_HIS

09.請將資料表MEMBER3截斷(truncate)

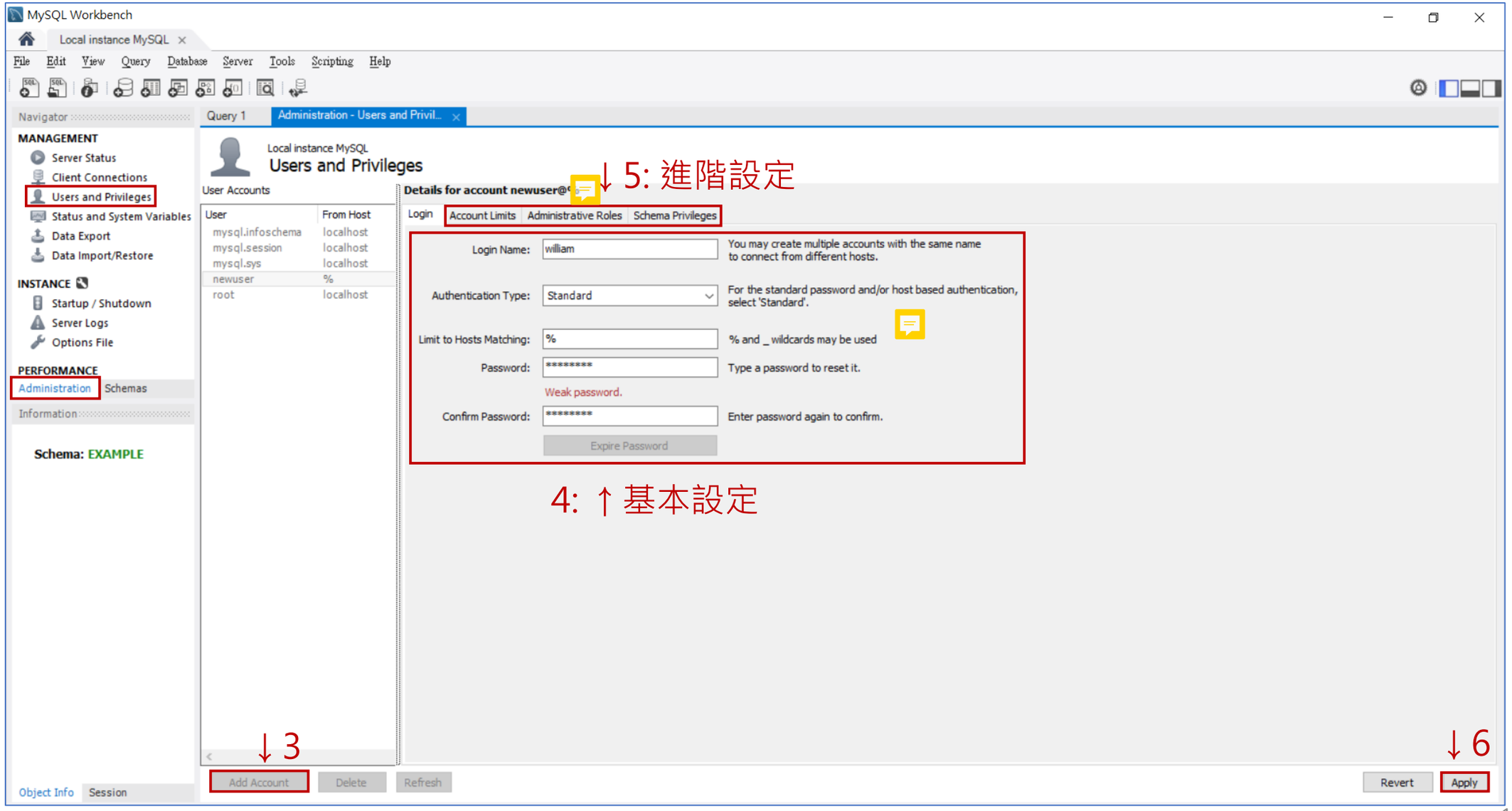
10.請將資料表MEMBER3的儲存引擎改為MyISAM

11.請將資料表MEMBER3移除

DCL

- 簡述
  - Data Control Language，資料控制語言
  - 用以控制使用者(User)對資料庫各物件的權限
- 使用者帳號(User Account)
  - 說明: 連線時所輸入的使用者名(Username)，是使用者帳號的一部分
  - 組成: 使用者名@使用者IP或主機名
    - EX. william@192.168.43.5
    - 使用者william可以從IP:192.168.43.5連線
    - 另可用%表示不限定某段IP，EX. william@192.168.43.%、william@%
  - Workbench: 可依下頁方式新增使用者帳號

DCL-使用Workbench新建使用者帳號



DCL-grant敘述 (1/2)

- 語法

```
grant 權限1, ..,權限N
on [資料庫.]資料表
to 使用者帳號1, .., 使用者帳號N
[with grant option]
```

- 說明
  - 權限: 欲授予給使用者的權限。請參考 https://dev.mysql.com/doc/refman/8.0/en/grant.html
  - 資料庫/資料表: 可操作的對象。可用星號(*)表示全部

    EX. EXAMPLE.* 表示資料庫EXAMPLE底下的全部資料表
  - 使用者帳號: 得到權限的使用者帳號，EX. william@192.168.43.5
  - with grant option: 授權給其他使用者的權限

DCL-grant敘述 (2/2)

- 範例

```
-- 範例-grant敘述
-- 授予使用者william所有權限，包含授權給其他使用者的權限
grant all
on *.*
to 'william'@'192.168.43.5'
with grant option;
```

DCL-revoke敘述 (1/2)

- 語法

```
revoke 權限1, ..,權限N
on [資料庫.]資料表
from 使用者帳號1, .., 使用者帳號N
```

```
-- 僅用來撤銷授權給其他使用者的權限
revoke grant option
on [資料庫.]資料表
from 使用者帳號1, .., 使用者帳號N
```

- 說明
  - 權限: 欲從使用者移除的權限。請參考 https://dev.mysql.com/doc/refman/8.0/en/grant.html
  - 資料庫/資料表: 撤銷操作的物件。可用星號(*)表示全部

    EX. EXAMPLE.* 表示資料庫EXAMPLE底下的全部資料表
  - 使用者帳號: 被撤銷權限的使用者帳號，EX. william@192.168.43.5

DCL-revoke敘述 (2/2)

- 範例

```
-- 範例-revoke敘述1
-- 從使用者william撤銷所有操作物件的權限
revoke all
on *.*
from 'william'@'192.168.43.5';
```

```
-- 範例-revoke敘述2
-- 從使用者william撤銷授權給其他使用者的權限
revoke grant option
on *.*
from 'lee'@'%';
```

DCL-Exercise10

01.請利用Workbench建立一使用者帳號 Username: william，Host: %，Password:

P@ssw0rd (請擷圖)

02.請授予使用者william 對資料庫EXAMPLE底下所有資料表的所有權限

03.請撤銷william 對資料庫EXAMPLE底下所有資料表的所有權限