資料庫系統(大五下)--期末專題--作業2

基本資料

任課老師：江季翰老師

組別：第14組

組員：

40843245 黃奕捷

目錄表

[使用情況 1](#_Toc1551204334)

[使用案例 2](#_Toc747825510)

[系統功能說明 6](#_Toc500450327)

[試算系統 6](#_Toc1000008442)

[試算綜合所得稅 (當年度) 6](#_Toc1670393711)

[使用者系統 6](#_Toc34080548)

[訪客 (guest) 6](#_Toc455393064)

[一般使用者 (user) 6](#_Toc1083345850)

[管理使用者 (administrator) 6](#_Toc854823700)

[行政流程系統 8](#_Toc114545353)

[申請繳納綜合所得稅 (當年度) 8](#_Toc626415940)

[列印繳納綜合所得稅 (當年度) 8](#_Toc927652142)

[資料預處理和後處理的相關功能 8](#_Toc233729043)

[資料匯入和匯出 8](#_Toc720129533)

[匯入資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度) 8](#_Toc24748398)

[匯出資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度) 9](#_Toc1871139727)

[匯入資料 (納稅義務人綜合所得稅之申請單) (當年度) 9](#_Toc1178652630)

[匯出資料 (納稅義務人綜合所得稅之申請單) (當年度) 9](#_Toc1653996131)

[匯入資料 (納稅義務人試算綜合所得稅所產生的結果之相關資料) (當年度) 9](#_Toc1255739361)

[匯入資料 (納稅義務人試算綜合所得稅所產生的結果之相關資料) (當年度) 9](#_Toc1671887492)

[匯入資料 (納稅義務人之個資) (當年度) 10](#_Toc610461666)

[匯出資料 (納稅義務人之個資) (當年度) 10](#_Toc537662577)

[匯入資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度) 10](#_Toc444984515)

[匯出資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度) 10](#_Toc1148351346)

[設定環境變數系統 10](#_Toc1449761997)

[匯入環境變數 11](#_Toc1419742381)

[匯出環境變數 11](#_Toc751167054)

[查詢環境變數 11](#_Toc224071191)

[正則表達式 11](#_Toc1991634629)

[資料表之完整性限制 14](#_Toc116662092)

[使用者帳戶之基本資料之資料表 14](#_Toc767863669)

[登入次數之資料表 15](#_Toc1708453114)

[驗證次數之資料表 15](#_Toc1649345150)

[納稅義務人之綜合所得稅申請單之基本資料之資料表 16](#_Toc815905015)

[納稅義務人之綜合所得稅申請單之相關資料之資料表 17](#_Toc1702724668)

[log之相關資料之資料表 19](#_Toc483034390)

[操作log之相關資料之資料表 19](#_Toc1186470484)

[驗證log之相關資料之資料表 20](#_Toc1730175503)

[登入和登出log之相關資料之資料表 20](#_Toc369101295)

[修改帳戶資料log之相關資料之資料表 21](#_Toc74672275)

[匯入和匯出納稅義務人之申請單log之相關資料之資料表 21](#_Toc620474543)

[ER-model 22](#_Toc223524010)

[圖片 23](#_Toc242364569)

# 使用情況

我們將會在這節簡單地討論稅務計算系統的使用案例(相關數額以2025年度為準)。這裡只討論免稅額和標準扣除額。至於其他相關數額之計算，請參考參考資料區附的連結裡的內容或財政部入口網裡的內容。

部分一：

免稅額之計算。

備註：

針對每個人依其免稅額扣減其綜合所得淨額。依據這個敘述，若採用戶合併申報所得稅，則有N個人之戶最多會有N個人會因免稅額而扣除其綜合所得淨額。

情況一：

符合下列其中之一條件的人若滿70歲，則該人的綜合所得稅之免稅額為145500元。

* 納稅義務人
* 其配偶
* 受納稅義務人扶養之直系尊親屬

情況二：

符合上述其中之一條件的人若未滿70歲，則該人的綜合所得稅之免稅額為97000元。

情況三：

不符合上述所有條件的人，則免稅額為0。

部分二：

標準扣除額之計算。

情況一：

若納稅義務人為單身，則標準扣除額為131000元。

情況二：

若納稅義務人有配偶，則標準扣除額為262000元。

備註：

此部分的情況二只適用於夫妻合併申報綜合所得稅。例如：若夫妻採用分開申報綜合所得稅 (也就是，夫申報夫的，妻申報妻的。)，則要適用情況一。

參考資料：

[2024 所得稅級距表整理！綜所稅免稅額、扣除額、累進差額級距調整前後比較 - Money101](https://www.money101.com.tw/blog/%E6%89%80%E5%BE%97%E7%A8%85-%E5%85%8D%E7%A8%85%E9%A1%8D-%E6%89%A3%E9%99%A4%E9%A1%8D-%E7%A8%85%E7%8E%87-%E7%B4%9A%E8%B7%9D)

# 使用案例

在此小節，我們將會隨便取幾個例子來說。

案例：

案例一：

案例一之一：

題目：

在114年度，納稅義務人甲沒有配偶也沒有扶養直系尊親屬也沒有子女。甲的相關資訊如下。

甲未滿70歲。甲在大同公司工作，每個月可以領取70000 元的薪水。可惜的是，甲當年未從大同公司獲取到股利。且甲未有大同公司的股票 (也就是，持股比例 = 0)。同時

，大同公司為提供福利給該公司的員工，也未提供特別扣除額。

甲當年花了30000元投自己保全民健保。此外，甲還花了200元投保了強制汽車責任險。甲當年沒有投保其他保險。

假設該戶沒有其他收入、其他費用、其他特別扣除額和其他應記載之會計分錄。

若甲採用戶合併申報綜得所得稅。則該戶要繳納多少綜合所得稅?

我的答案：

首先：

先計算免稅額。

因為甲未滿70歲，甲的免稅額 = 97000元。

此外，因為甲沒有配偶，也沒有受扶養直系尊親屬，所以該戶的只有甲受到免稅額的減除。(因為其他人不符合免稅額的減除之適用對象。)

因此，該戶的免稅額 = 97000元。

第二：

計算標準扣除額。

因為甲沒有配偶，則標準扣除額 = 131000元。

第三：

在當年度。

甲的薪資所得收入 = 70000 \* 12 = 840000元

薪資所得之必要費用 = min( 0 , 840000 \* 3% ) = 0

薪資所得特別扣除額 = min( 840000 – 0 , 218000 ) = 218000

第四：

因為該戶沒有其他所得收入，所以該戶的綜合所得收入 = 840000

第五：

因為在當年度該戶沒有應記載之會計分錄，這意味著該戶沒有其他合法交易，該戶也不會有其他的資產、其他負債、其他收益(收入和利益)和費損(費用、支出和損益)。所以，該戶不會有其他的特別扣除額之項目。也不會有其他可列舉扣除額的項目。

因此，我們可以求出。

特別扣除額 = 218000 (備註：只有薪資所得特別扣除額這個項目。)

可列舉扣除額 = 0

站在節稅的角度，在可列舉扣除額和標準扣除額中(備註：只能二選一。)，納稅義務人會選擇數額最高的選項。此題，選擇標準扣除額。

扣除額 = 特別扣除額 + max (可列舉扣除額 , 標準扣除額)

=> 扣除額 = 218000 + max( 0, 131000 ) = 218000 + 131000 = 349000

第六：

計算綜合所得總額。

綜合所得總額 = 840000 (備註：只有甲的薪資所得收入。)

第七：

計算綜合所得淨額。

綜合所得淨額 = 綜合所得總額 - 免稅額 - 扣除額

=> 綜合所得淨額 = 840000 - 97000- 349000 = 394000

第八：

根據當年度的綜合所得淨額之級距表，可知該戶的綜合所得稅之稅率為5%。

所以，該戶的綜合所得稅額的計算方式如下。

綜合所得之課徵所得 = 綜合所得淨額

=> 綜合所得之課徵所得 = 394000

綜合所得稅額 = 綜合所得之課徵所得 \* 綜合所得之課徵稅率

=> 綜合所得稅額 = 394000 \* 5% = 19700

綜合所得之應納稅額 = 綜合所得稅額

=> 綜合所得稅額 = 19700

最後，我們知道該戶要繳納綜合所得稅 19700元新台幣。

案例一之二：

題目：

承續案例一之一的題目。

但甲有未滿70歲之配偶。該配偶沒有薪資所得收入或其他收入。若該戶採用夫妻合併報綜合所得稅，則該戶應繳納多少綜合所得稅?

我的答案：

首先：

先計算免稅額。

因為甲未滿70歲，甲的免稅額 = 97000元。

因為甲的配偶未滿70歲，甲的配偶的免稅額 = 97000元。

此外，因為甲沒有受扶養直系尊親屬，所以該戶的只有甲和甲的配偶受到免稅額的減除。(因為其他人不符合免稅額的減除之適用對象。)

因此，該戶的免稅額 = 97000 + 97000 = 194000 元。

第二：

計算標準扣除額。

因為甲有配偶，則標準扣除額 = 262000 元。

第三：

在當年度。

甲的薪資所得收入 = 70000 \* 12 = 840000元

薪資所得之必要費用 = min( 0 , 840000 \* 3% ) = 0

薪資所得特別扣除額 = min( 840000 – 0 , 218000 ) = 218000

第四：

因為該戶沒有其他所得收入，所以該戶的綜合所得收入 = 840000

第五：

因為在當年度該戶沒有應記載之會計分錄，這意味著該戶沒有其他合法交易，該戶也不會有其他的資產、其他負債、其他收益(收入和利益)和費損(費用、支出和損益)。所以，該戶不會有其他的特別扣除額之項目。也不會有其他可列舉扣除額的項目。

因此，我們可以求出。

特別扣除額 = 218000 (備註：只有薪資所得特別扣除額這個項目。)

可列舉扣除額 = 0

站在節稅的角度，在可列舉扣除額和標準扣除額中(備註：只能二選一。)，納稅義務人會選擇數額最高的選項。此題，選擇標準扣除額。

扣除額 = 特別扣除額 + max (可列舉扣除額 , 標準扣除額)

=> 扣除額 = 218000 + max( 0, 262000 ) = 218000 + 262000 = 480000

第六：

計算綜合所得總額。

綜合所得總額 = 840000 (備註：只有甲的薪資所得收入。)

第七：

計算綜合所得淨額。

綜合所得淨額 = 綜合所得總額 - 免稅額 - 扣除額

=> 綜合所得淨額 = 840000 – 194000 - 480000 = 166000

第八：

根據當年度的綜合所得淨額之級距表，可知該戶的綜合所得稅之稅率為5%。

所以，該戶的綜合所得稅額的計算方式如下。

綜合所得之課徵所得 = 綜合所得淨額

=> 綜合所得之課徵所得 = 166000

綜合所得稅額 = 綜合所得之課徵所得 \* 綜合所得之課徵稅率

=> 綜合所得稅額 = 166000 \* 5% = 8300

綜合所得之應納稅額 = 綜合所得稅額

=> 綜合所得稅額 = 8300

最後，我們知道該戶要繳納綜合所得稅 8300 元新台幣。

參考資料：

[2024 所得稅級距表整理！綜所稅免稅額、扣除額、累進差額級距調整前後比較 - Money101](https://www.money101.com.tw/blog/%E6%89%80%E5%BE%97%E7%A8%85-%E5%85%8D%E7%A8%85%E9%A1%8D-%E6%89%A3%E9%99%A4%E9%A1%8D-%E7%A8%85%E7%8E%87-%E7%B4%9A%E8%B7%9D)

# 系統功能說明

我們預計開發一款試算綜合所得稅系統，其預計的主要功能如下。

## 試算系統

### 試算綜合所得稅 (當年度)

納稅義務人可以用此系統在申報期間外試算，以進行實務練習。

## 使用者系統

使用者系統專門處理跟使用者之相關資料。其中這裡的使用者包括訪客、一般使用者和管理使用者。

### 訪客 (guest)

若以訪客身分登入，則只有訪客的權限，只能進行稅務試算。

不過，不用帳號和密碼即可以訪客身分登入。

### 一般使用者 (user)

若以一般使用者身分登入，則只有訪客和一般使用者的權限，在此系統進行訪客身分的操作和基本操作(包括稅務計算和繳稅等)。不能進行帳戶管理和帳號查詢。

然而，為了資訊安全，必須要

(1)

帳號和密碼及驗證碼認證才可以一般使用者身分登入。

(2)

一般使用者每一個動作會被記錄在log檔。

(3)

不過為了資訊安全，一般使用者無法看到使用者的密碼，當然也不能在後台直接修改密碼。若忘記密碼，則只能按忘記密碼，然後完成忘記密碼之流程。

### 管理使用者 (administrator)

若以管理使用者登入，則有更高的權限，有訪客、一般使用者和管理者的權限，在此系統可以進行訪客身分的操作和一般使用者身分的操作，也可以進行帳戶管理和帳號查詢。

然而，為了資訊安全，當權限很高時，需要在登入時設立許多的限制且修改時會有詳細記錄，還有其他。下面說明了管理使用者的限制。

需要帳號、密碼、驗證碼認證及圖靈驗證才可登入。其中圖靈驗證跟Github的圖靈驗證一模一樣。會出現兩道題目，每題中，會出現兩張圖案，需要將最右邊的圖案裡的食指指向跟在最左邊的圖案裡的食指同一個方向然後按"驗證"按鈕，當回答符合解答時，才會完成一道題目。當連續回答答對2次時，則會驗證成功，並把回答正確和回答錯誤次數歸零。反之當回答錯誤達到2次，則會驗證失敗，並上鎖30分鐘，之後需要重新輸入帳號才能再次進行驗證。此外，不管驗證結果如何(驗證失敗或驗證成功)，都會在登入成功後，將相關資料(包括已登入的帳戶)存到log檔，以便管理使用者方便查看登入的相關足跡，此為資訊安全中重要的一環。

(2)

管理使用者每一個動作也會被記錄在log檔。

(3)

不過為了資訊安全，管理使用者無法看到使用者的密碼，當然也不能在後台直接修改密碼。

|  |  |  |  |
| --- | --- | --- | --- |
| 身分 | 訪客 | 一般使用者 | 管理使用者 |
| 權限優先權 | 最低 | 第二高 | 最高 |
| 權限 (採累積列舉) | * 試算稅務 | * 綜合所得稅捐計算 * 申報綜合所得稅 * 列印綜合所得稅之申請單 * 匯入資料 * 匯出資料 | * 處理一般使用者之帳戶 * 查詢處理一般使用者之帳戶 |
| 登入? | 不 | 需要 | 需要 |
| 帳戶之相關資訊 | 無(因不須登入) | * 帳號 * 密碼 | * 帳號 * 密碼 |
| 登入時驗證 | 無 | 驗證碼 | * 驗證碼 * 圖靈驗證 |

|  |
| --- |
| 個人之基本資料 |
| 姓名  出生日期  身分證字號 |

|  |
| --- |
| 納稅義務人之綜合所得稅申請單之基本資料 |
| 稅捐年度  納稅義務人的個人之基本資料  納稅義務人是否有配偶?  受納稅義務人扶養的對象和其相關之基本資料 |

|  |
| --- |
| 使用者的帳戶之基本資料 |
| 帳號  密碼  log檔 |

|  |
| --- |
| log檔 |
| 使用者的每一個動作，包括：   * 驗證次數(驗證失敗次數、驗證成功次數) * 重設密碼 * 登入和登出 * 使用者之其他操作 |

## 行政流程系統

納稅義務人須以一般使用者身分登入帳戶。

### 申請繳納綜合所得稅 (當年度)

納稅義務人可以根據納稅義務人當年度之相關資料製作一份綜合所得稅之申請表。以供納稅義務人方便填寫當年度綜合所得稅之申請表。

### 列印繳納綜合所得稅 (當年度)

納稅義務人可以根據納稅義務人當年度之綜合所得稅之申請表，列印之。

不過，納稅義務人須登入帳戶。

## 資料預處理和後處理的相關功能

### 資料匯入和匯出

納稅義務人在此系統可以匯入和匯出資料，藉此大大增加資料的可移植性。

不過，納稅義務人須登入帳戶。以便辨識身分。

### 匯入資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度)

納稅義務人可以匯入計算綜合所得稅會用到的資料。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯出資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度)

納稅義務人可以匯出計算綜合所得稅會用到的資料。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯入資料 (納稅義務人綜合所得稅之申請單) (當年度)

納稅義務人可以匯入計算綜合所得稅會用到的資料。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯出資料 (納稅義務人綜合所得稅之申請單) (當年度)

納稅義務人可以匯入其綜合所得稅之申請單。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯入資料 (納稅義務人試算綜合所得稅所產生的結果之相關資料) (當年度)

納稅義務人可以匯入其綜合所得稅之相關結果。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯入資料 (納稅義務人試算綜合所得稅所產生的結果之相關資料) (當年度)

納稅義務人可以匯出其綜合所得稅之相關結果。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯入資料 (納稅義務人之個資) (當年度)

納稅義務人可以匯入其個資。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯出資料 (納稅義務人之個資) (當年度)

納稅義務人可以匯出其個資。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯入資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度)

納稅義務人可以匯入其用來計算綜合所得稅之相關資料。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

### 匯出資料 (納稅義務人為了試算綜合所得稅之相關資料) (當年度)

納稅義務人可以匯出其用來計算綜合所得稅之相關資料。

檔案編輯器：

至少可以用excel編輯。

副檔名：

.xlsx。

## 設定環境變數系統

專門設定和查詢此系統之環境變數。這樣可以讓管理者更方便維護和管理系統。

注意：

需要管理者權限才可以進入此系統。

環境變數：

詳見，系統需求分析節裡正則表達式的identifier。

### 匯入環境變數

可以從裝置匯入環境變數。

支援檔案格式：

.json

### 匯出環境變數

可以匯出環境變數至裝置。

支援檔案格式：

.json

### 查詢環境變數

可以查看已經匯入環境變數之檔案。

支援檔案格式：

.json

# 正則表達式

自訂之正則表達式的identifier：

# 正號。

<positive\_sign> := “+” ;

# 負號。

<negative\_sign> := “-”;

# underscore。

<underscore> := “\_”;

# 數字字元。

<digit> := [0-9];

# 數字字元 (不包含0)。

<nonzero\_digit> := [0-9];

# 數字零。

<zero> := 0 ;

# 數字一。

<one> := 1 ;

# 數字二。

<two> := 2 ;

# 大寫英文字母。

<uppercase\_letter> := [A-Z];

# 小寫英文字母。

<lowercase\_letter> := [a-z];

# 英文字母。

<letter> := (<uppercase\_letter> | <lowercase\_letter>);

# 英文字母或數字字元。

<alpha\_digit> := (<letter> | <digit>);

# 正整數(不含多餘的前置0，如：0123)。

<positive\_integer> := <plus\_sign>?<nonzero\_digit><digit>\* ;

# 非負整數 (不含多餘的前置0，如：0123)。

<nonnegative\_integer> := (<zero>|<positive\_integer>) ;

# 負整數。

<negative\_integer> := <negative\_sign><positive\_integer> ;

# identifier (in programming language) (not start with underscore)

<identifier> := <letter>(<alpha\_digit>|<underscore>)\* ;

# AC。西元日期。格式： yyyy/mm/dd。

# 注意：月份和日份要在有效範圍內。每年有12個月。每個月有28到31天。

<AC\_date> := <digit>{4}/<digit>{2}/<digit>{2}

# 時間(二十四小時制)。格式：hh:mm:ss。

<time> := <digit>{2}:<digit>{2}:<digit>{2} ;

# 日期時間。

# 格式：yyyyy/mm/dd hh:mm:ss。

# 其中最左邊的兩個mm代表月份，最右邊的兩個則代表分鐘。

<datetime> := <AC\_date> ” ” <time> ;

# ending part of private email.

<private\_email\_account\_endingPart> := “gmail.com”;

# ending part of NFU school email.

<NFU\_email\_account\_endingPart> := “gm.nfu.edu.tw”;

備註：

(1)

這份報告裡的正規表達式採用MSDS .NET上的正則表達式(詳見下面網址)。不過，為了更方便地書寫和閱讀我在裡面定義了幾個符號，如下表。

|  |  |  |
| --- | --- | --- |
| 符號 | 英文 | 說明 |
| ; | semicolon | 表示一個敘述的結尾，像程式語言C一樣。 |
| “” | Pair of double quotation | 一對雙引號應恰好包含了一個字串，例如："123"表示字串”123”，跟數字一百二十三不等價因為型別不同。 |
| # | Number sign | 表示單行註解。跟Python一樣。 |
| ‘’’  ‘’’ | Pair of three single quotation | 一對的連續三個單引號表示多行註解。一對的連續三個單引號中的所有字元會被註解。跟Python一樣。注意：三個單引號一定要連續(不能有任何字元)，且一定要有一對連續三個單引號。 |

複習：

我們快速複習一下正則表達式。

|  |  |  |
| --- | --- | --- |
| quantifier | 英文 | 說明 |
| \* | Asterisk sign | 表示零個、一個或多個重複的字元。  例如：”123”\*，將會配對字串含有零個、一個或多個”123”，"123123"會被配對，”123”也會被配對，””也會被配對。 |
| + | Plus sign | 表示至少一個或多個重複的字元。  例如：”123”+，將會配對字串含有一個或多個”123”，"123123"會被配對，”123”也會被配對，””不會被配對。 |
| ? | Question mark | 表示零個或一個重複的字元。  例如：”123”?，只會配對字串含有零個或一個”123”，這裡只有""和"123”會被配對。 |
| - | Dash sign | 放在[]裡，表示連續的範圍。  例如：[a-z]將會配對a到z(小寫字母) |
| {n} |  | 只會配對到恰好重複n次。  注意：  n>=0 , n 為整數。  例如：1{3}只會配對”111”。 |
| {n,m} |  | 只會配對到重複n到m次。  注意：  n>=0 , m>=0 , m>=n ,n和m為整數。  例如：1{3,5}只會配對”111”、”1111”和”11111”。 |

|  |  |  |
| --- | --- | --- |
| Alternation construct | 英文 | 說明 |
| | |  | 表示或。  例如：  th(e|is)，只會配對”the”和”this”。 |

還有更多。詳見，下方的網址。

詳見：

有關Regular expression Language (正則表達式語言)，詳見MSDS (微軟官方文件) 的內容。如下：

[Regular Expression Language - Quick Reference - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference)

# 資料表之完整性限制

## 使用者帳戶之基本資料之資料表

使用者的帳號：

應該跟電子郵件帳號一樣，必須為字串類型，且恰好有一個@和@之後只能接gm.nfu.edu.tw.com或gmail.com。

因為它一定為該使用者的電子郵件帳號。(只支援私人的和NFU校用的電子郵件)。

正則表達式：

# email account.

<email\_account> := <alpha\_digit>+@(<private\_email\_account\_endingPart>| <NFU\_email\_account\_endingPart>);

## 登入次數之資料表

登入次數之相關設定。

連續嘗試登入次數上限：

應該為一個非負整數。

指示可以連續嘗試登入失敗次數。當登入失敗時

達到連續登入失敗次數達到連續嘗試登入次數上限就會鎖住系統"連續登入失敗次數之上鎖時間"，也就是<max\_consecutive\_failed\_login\_locking\_time>分鐘。

詳見，下方註解。

連續登入失敗次數之上鎖時間：

應該為一個非負整數。

指示因連續嘗試登入次數大於等於其上限，致使鎖該裝置的登入系統，解鎖的時間。

詳見，下方註解。

正則表達式：

# 連續嘗試登入失敗次數上限。

# 只要連續登入失敗次數達到連續嘗試登入次數上限就會鎖 該裝置的登入系統。

# 若要設定連續嘗試登入次數上限為無限大，則設定上鎖後馬上解鎖，透過設定

# <max\_consecutive\_failed\_login\_locking\_time> = 0 ;

<max\_consecutive\_failed\_login> := <positive\_integer>

# 連續登入失敗次數之上鎖時間。

# 因連續嘗試登入次數大於等於其上限，致使鎖該裝置的登入系統，解鎖的時間。

# 當它被設為零。代表只會上鎖零分鐘。也就是上鎖後馬上解鎖。

# 單位：分鐘。

<max\_consecutive\_failed\_login\_locking\_time> := <nonnegative\_integer>

## 驗證次數之資料表

驗證次數之相關設定。

驗證次數失敗上限：

應該為一個非負整數。

當在完成驗證前，驗證嘗試失敗次數達到其上限時，則會視為登入失敗。此時，驗證嘗試失敗次數會被清除為零。不過，連續登入嘗試失敗次數會遞增 <max\_failed\_identification> (也就是，<max\_consecutive\_failed\_login> += <max\_failed\_identification> ; ) 。當驗證步驟成功後，它會被清除為零 。

驗證題目數：

應該為一個非負整數。

指示在驗證步驟中要回答對幾道題目，才會完成驗證步驟。

驗證失敗之登入嘗試次數增加數。

應該為一個正整數。

指示當驗證失敗時，驗證失敗之登入嘗試次數會增加多少。

正則表達式：

# 驗證次數失敗上限。

# 注意：當它被設定為零時，代表會自動跳過驗證這個步驟。

# 注意：不是完成一道題目就將它清除為零，只有當完成所有題目時，才會被視為驗

# 證成功，此時才會將它清除為零。

<max\_failed\_identification> := <nonnegative\_integer> ;

# 驗證題目數。

# 注意：當它被設定為零時，代表會自動跳過驗證步驟。

<number\_of\_identification> := <nonnegative\_integer> ;

# 驗證失敗之登入嘗試次數增加數。

# 只要每次登入失敗時， 它就會被增加 <max\_failed\_identification>。

# (也就是，<max\_consecutive\_failed\_login> += <max\_failed\_identification> ; )

# 注意：當它被設定為大於等於<max\_failed\_identification>時，只要每次登入失敗時，

# 它就會被增加 <max\_failed\_identification> 因而觸發上鎖機制。

<max\_failed\_identification> := <positive\_integer> ;

## 納稅義務人之綜合所得稅申請單之基本資料之資料表

納稅義務人之綜合所得稅申請單上的基本資料。

英文名稱：

應該為字元長度介於5到20之間的英文字母字元或數字字元。

出生日期：

應該為西元日期。且不能大於等於今天。

身分證字號：

應符合戶政事務所之身分證字號之訂定規範。也就是，應該包含10個字元，其中2第一個字母為大寫英文字母，第二個字母為數字一或二，第三到第十字母皆為數字字元。當然還有其他規範，不過，有點複雜，在此月略過。

正則表達式：

# 英文名稱。

<english\_name> := <alpha\_digit>{5,20}

# 出生日期。

<birth\_date> := <AC\_date>

# 性別編號。

<gender\_id> := ( <one>|<two> ) ;

# 身分證字號。

<identification\_id> := <uppercase\_letter> <gender\_id><digit>{8} ;

# 配偶狀態。

<spouse\_id> := ( <one>|<two> );

## 納稅義務人之綜合所得稅申請單之相關資料之資料表

納稅義務人之綜合所得稅申請單上的相關資料，用來決定綜得所得之應納稅額。

詳見，所得稅法第14條之規範。

薪資所得：

應為一個非負整數。當數字為零時，則代表該人沒有薪資所得。

每年的薪資所得收入。

單位：年，新台幣。以下的單位也跟這個項目一樣。

營利所得：

應為一個非負整數。當數字為零時，則代表該人沒有營利所得。

因營利事業的所得。

執行業務所得：

應為一個非負整數。當數字為零時，則代表該人沒有執行業務所得。

利息所得：

應為一個非負整數。當數字為零時，則代表該人沒有利息所得。

租賃所得：

應為一個非負整數。當數字為零時，則代表該人沒有租賃所得。

自力耕作、漁、牧、林、礦之所得：

應為一個非負整數。當數字為零時，則代表該人沒有自力耕作、漁、牧、林、礦

之所得。

財產交易所得：

應為一個非負整數。當數字為零時，則代表該人沒有交易財產之所得。

競技、競賽及機會中獎之獎金或給與：

應為一個非負整數。當數字為零時，則代表該人沒有競技、競賽及機會中獎之獎金或給與。

退職所得：

應為一個非負整數。當數字為零時，則代表該人沒有退職所得。

退休金。

其他所得：

應為一個非負整數。當數字為零時，則代表該人沒有其他所得。

正則表達式：

# 薪資所得。

<salary\_earning> := <nonnegative\_integer> ;

# 營利所得。

<operating\_earning> := <nonnegative\_integer> ;

# 執行業務所得。

<working\_earning> := <nonnegative\_integer> ;

# 利息所得。

<interest\_earning> := <nonnegative\_integer> ;

# 租賃所得。

<rent\_earning> := <nonnegative\_integer> ;

# 自力耕作、漁、牧、林、礦之所得。

# yield 為農產品的英文。同義詞：produce。

<yield\_earning> := <nonnegative\_integer> ;

# 財產交易所得。

<assets\_trading\_earning> := <nonnegative\_integer> ;

# 競技、競賽及機會中獎之獎金或給與。

<reward\_earning> := <nonnegative\_integer> ;

# 退職所得。

<retirement\_earning> := <nonnegative\_integer> ;

# 其他所得。

<other\_earning> := <nonnegative\_integer> ;

## log之相關資料之資料表

### 操作log之相關資料之資料表

一個專門儲存使用者登入後的操作之資料表。

操作ID：

應該為一個正整數。

指示使用者所進行的操作。

操作成功時間：

應該為一個日期時間。

指示使用者成功地進行的操作之相對應的時間。

正則表達式：

# 操作ID。

<operation\_id> := <positive\_integer> ;

# 操作成功時間。

<operation\_success\_datetime> := <datetime> ;

### 驗證log之相關資料之資料表

一個專門儲存驗證步驟中的相關資料的資料表。

驗證成功時間：

應該為一個日期時間。

指示該使用者的驗證時間，更精確的說，指示使用者進入驗證步驟的時間。

正則表達式：

# 驗證成功時間。

<verifiication\_success\_datetime> := <date\_time> ;

### 登入和登出log之相關資料之資料表

一個專門儲存使用者登入和登出的相關資料的資料表。

登入成功時間：

應該為一個日期時間。

指示該使用者的登入時間，更精確的說，指示使用者登入成功的時間。

登入失敗時間：

應該為一個日期時間。

指示該使用者登入失敗的時間。

登出成功時間：

應該為一個日期時間。

指示該使用者的登出時間，更精確的說，指示使用者登出成功的時間。

登出失敗時間：

應該為一個日期時間。

指示該使用者登出失敗的時間。

正則表達式：

# 登入成功時間。

<login\_success\_datetime> := <datetime> ;

# 登入失敗時間。

<login\_failed\_datetime> := <datetime> ;

# 登出成功時間。

<logout\_success\_datetime> := <datetime> ;

# 登出失敗時間。

<logout\_failed\_datetime> := <datetime> ;

### 修改帳戶資料log之相關資料之資料表

一個專門儲存使用者修改其相關資料的資料表。

修改密碼成功時間：

應該為一個日期時間。

若且唯若當使用者成功修改密碼時，該時間會被記錄。

修改密碼失敗時間：

應該為一個日期時間。

若且唯若當使用者修改密碼失敗時，該時間會被記錄。

正則表達式：

# 修改密碼成功時間

<change\_password\_success\_datetime> := <datetime> ;

# 修改密碼失敗時間

<change\_password\_failed\_datetime> := <datetime> ;

### 匯入和匯出納稅義務人之申請單log之相關資料之資料表

一個專門儲存匯入和匯出有關納稅義務人上的納稅申請單上的相關資料之資料表(備註：申請單上包含納稅義務人之稅捐，當稅捐被計算完時。)。因為只儲存有關納稅義務人，所以只有在納稅系統中，相關資料才有可能會被儲存。在稅務試算系統(備註：該系統只有試算稅捐之功能，沒有繳稅之功能。)中，相關資料一定不會被儲存。

匯入申請單資料成功時間：

應該為一個日期時間。

若且唯若當使用者成功匯入其申請上的資料時，該時間會被記錄。

匯入申請單資料失敗時間：

應該為一個日期時間。

若且唯若當使用者失敗匯入其申請上的資料時，該時間會被記錄。

匯出申請單資料成功時間：

應該為一個日期時間。

若且唯若當使用者成功匯出其申請上的資料時，該時間會被記錄。

匯出申請單資料失敗時間：

應該為一個日期時間。

若且唯若當使用者失敗匯出其申請上的資料時，該時間會被記錄。

正則表達式：

# 匯入申請單資料成功時間。

<import\_form\_success\_datetime> := <datetime> ;

# 匯入申請單資料失敗時間。

<import\_form\_failed\_datetime> := <datetime> ;

# 匯出申請單資料成功時間。

<export\_form\_success\_datetime> := <datetime> ;

# 匯出申請單資料失敗時間。

<export\_form\_failed\_datetime> := <datetime> ;

# ER-model

備註：

這份文件的ER model皆採用我自己定義的規範 -- ERMDF。

ERMDF：

ERMDF為combination of Entity Relationship Model and DataFlow 的縮寫。是我自己發明的規範。我將ERM和Dataflow組合成該規範。在此規範中，符號皆採用課堂投影片裡的ERM的符號和Dataflow的符號，還有額外新增幾個符號。符號如下：

|  |  |  |
| --- | --- | --- |
| 符號形狀和特徵 | 自己定義或重新定義 | 說明 |
| 長方形 | 否 | 強實體 |
| 長方形(內有圈圈) | 否 | 弱實體(可當多值屬性) |
| 橢圓形 | 否 | 單值屬性 |
| 橢圓形(內有圈圈) | 否 | 衍生屬性 (單值) |
| 菱形 | 否 | 關係 |
| 圓形(小小的) | 是(自己定義) | 連接至其他實物。  詳見，下方解釋。 |
| 長方形(裡有兩條豎限) | 否 | 流程圖中的程序。 |

基數：

我採用課堂投影片裡的ERM的例子來表示基數。

連續三個句點表示範圍。例如：0...1，可以從0個到1個。

符號形狀和特徵之詳細解釋：

1. 圓形(小小的)

我把流程圖的圓形(小小的)符號延伸出新的符號，表示連接至其他實物。圓形內的文字為標籤，用來識別文字要連接到哪個實物。

以此文件裡的實際例子為例。

在” 資料庫系統(大五下)--期末專題--作業1--ERM\_\_user”檔案裡，有一個關係連接著標籤為"a1"的圓形，同時，在另外一個檔案"資料庫系統(大五下)--期末專題--作業1--ERM\_\_main "裡，有一個強實體連接著標籤為”a1”的圓形。代表該強實體連接至該關係。這表示著擁有administration permission即可以讀取general user info。

名詞解釋：

實物：

為一個實體或屬性。

實體：

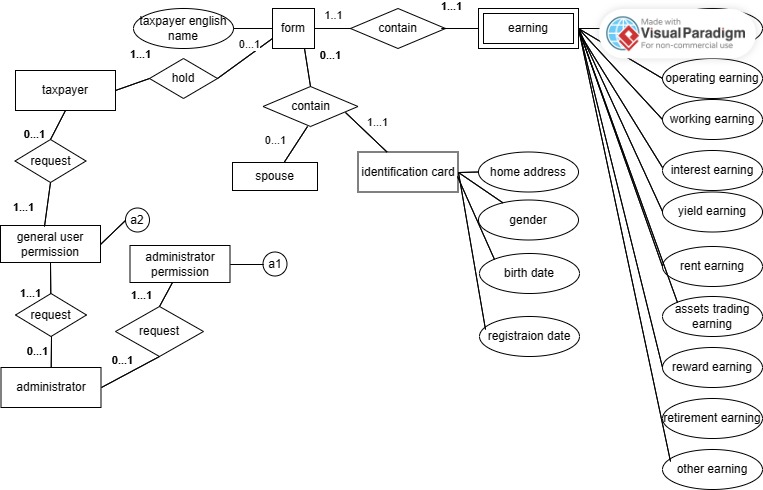
為一個強實體或弱實體。

屬性：

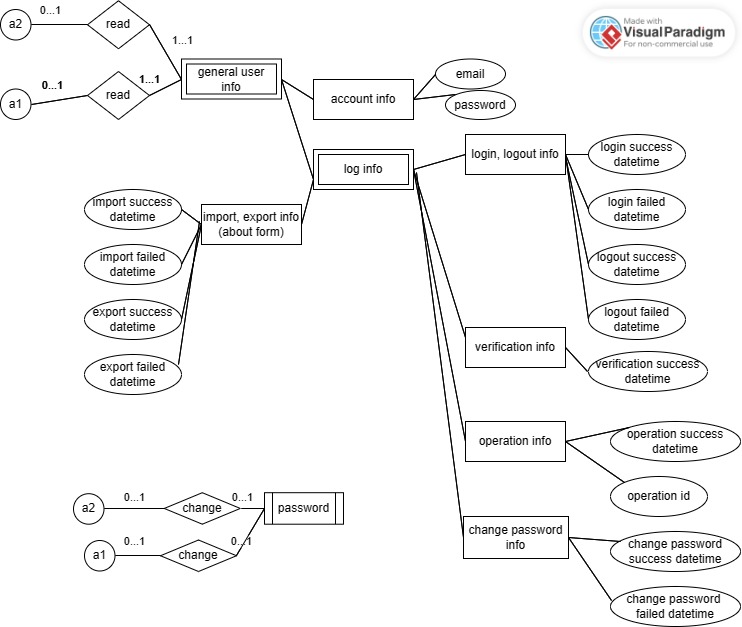
為一個單值屬性或多值屬性。

圖片

檔名：資料庫系統(大五下)--期末專題--作業1--ERM\_\_main 。



檔名：資料庫系統(大五下)--期末專題--作業1--ERM\_\_user。



# 資料庫 tax

## SQL syntax

### Create

CREATE DATABASE tax;

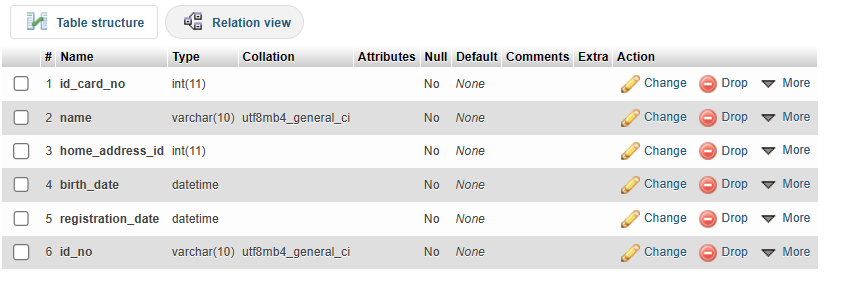
### Select

SELECT DATABASE tax;

## 綱要

### id\_card

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE id\_card(

id\_card\_no INT NOT NULL UNIQUE AUTO\_INCREMENT,

name VARCHAR(10) NOT NULL UNIQUE,

home\_address\_no INT NOT NULL UNIQUE CHECK(home\_address\_no > 0 ),

birth\_date DATETIME NOT NULL UNIQUE,

registration\_date DATETIME NOT NULL UNIQUE,

id\_no INT NOT NULL UNIQUE CHECK(id\_no > 0 )

);

###### CREATE TRIGGER

CREATE TRIGGER `data\_validation` BEFORE INSERT ON `id\_card`

FOR EACH ROW BEGIN

IF

NEW.birth\_date > NOW() OR

NEW.registration\_date > NOW() OR

NEW.birth\_date > NEW.registration\_date

THEN

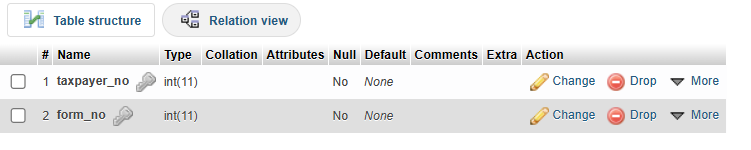
SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid data!';

END IF;

END

### taxpayer

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE taxpayer(

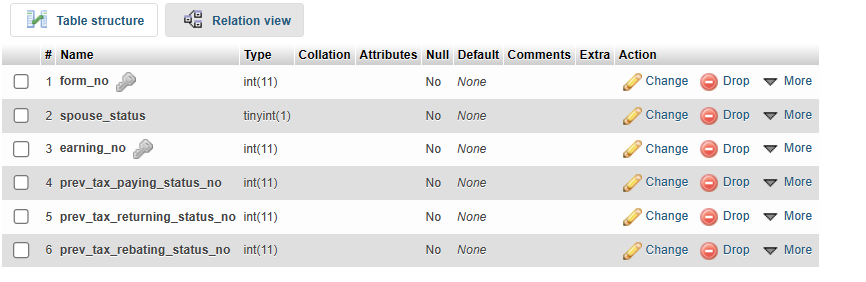
taxpayer\_no INT NOT NULL UNIQUE CHECK(taxpayer\_no > 0 ),

form\_no INT NOT NULL UNIQUE CHECK(form\_no > 0 )

);

### form

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE form(

form\_no INT NOT NULL UNIQUE CHECK(form\_no > 0),

spouse\_status BOOLEAN NOT NULL,

earning\_no INT NOT NULL UNIQUE CHECK(earning\_no > 0 ),

prev\_tax\_paying\_status\_nos INT NOT NULL CHECK(prev\_tax\_paying\_status\_nos > -5 AND prev\_tax\_paying\_status\_nos < 5 ),

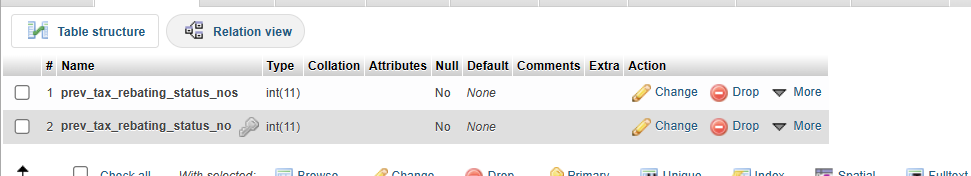
prev\_tax\_returning\_status\_nos INT NOT NULL CHECK(prev\_tax\_returning\_status\_nos > -5 AND prev\_tax\_returning\_status\_nos < 5 ),

prev\_tax\_rebating\_status\_nos INT NOT NULL CHECK(prev\_tax\_rebating\_status\_nos > -5 AND prev\_tax\_rebating\_status\_nos < 5 )

);

### prev\_tax\_rebating\_list

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE prev\_tax\_rebating\_list(

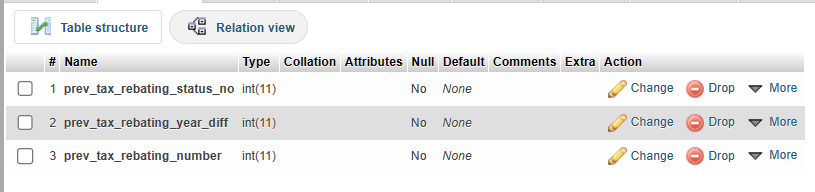
prev\_tax\_rebating\_status\_nos INT NOT NULL CHECK(prev\_tax\_rebating\_status\_nos > 0 ),

prev\_tax\_rebating\_status\_no INT NOT NULL UNIQUE CHECK(prev\_tax\_rebating\_status\_no > 0 )

);

### prev\_tax\_rebating

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE prev\_tax\_rebating(

prev\_tax\_rebating\_status\_no INT NOT NULL CHECK(prev\_tax\_rebating\_status\_no > 0 ),

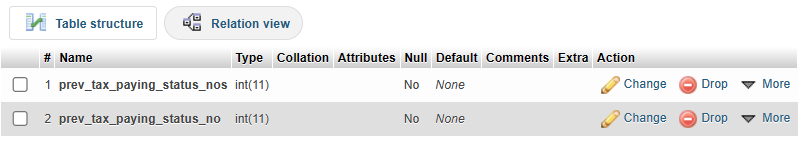
prev\_tax\_rebating\_year\_diff INT NOT NULL CHECK(prev\_tax\_rebating\_year\_diff > 0 AND prev\_tax\_rebating\_year\_diff <= 3 ),

prev\_tax\_rebating\_number INT NOT NULL CHECK(prev\_tax\_rebating\_number > 0 )

);

### prev\_tax\_paying\_list

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE prev\_tax\_paying\_list(

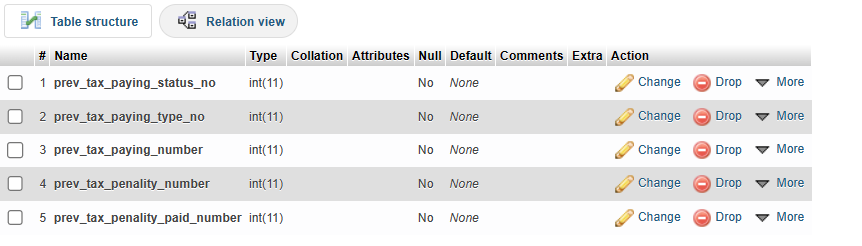
prev\_tax\_paying\_status\_nos INT NOT NULL CHECK(prev\_tax\_paying\_status\_nos > 0 ),

prev\_tax\_paying\_status\_no INT NOT NULL CHECK(prev\_tax\_paying\_status\_no > 0 )

);

### prev\_tax\_paying

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE prev\_tax\_paying(

prev\_tax\_paying\_status\_no INT NOT NULL CHECK(prev\_tax\_paying\_status\_no > 0 ),

prev\_tax\_paying\_type\_no INT NOT NULL CHECK(prev\_tax\_paying\_type\_no > 0 ),

prev\_tax\_paying\_number INT NOT NULL CHECK(prev\_tax\_paying\_number > 0 ),

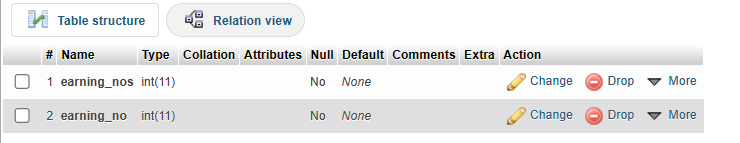
prev\_tax\_penality\_number INT NOT NULL CHECK(prev\_tax\_penality\_number > 0 ),

prev\_tax\_penality\_paid\_number INT NOT NULL CHECK(prev\_tax\_penality\_paid\_number > 0 )

);

### earning\_no\_list

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE earning\_no\_list(

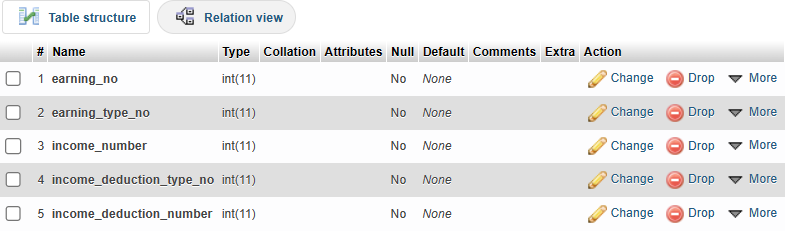
earning\_nos INT NOT NULL CHECK(earning\_nos > 0 ),

earning\_no INT NOT NULL CHECK(earning\_no > 0 )

);

### earning\_no

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE earning\_no(

earning\_no INT NOT NULL CHECK(earning\_no > 0 ),

earning\_type\_no INT NOT NULL CHECK(earning\_type\_no > 0 ),

income\_number INT NOT NULL CHECK(income\_number >= 0 ),

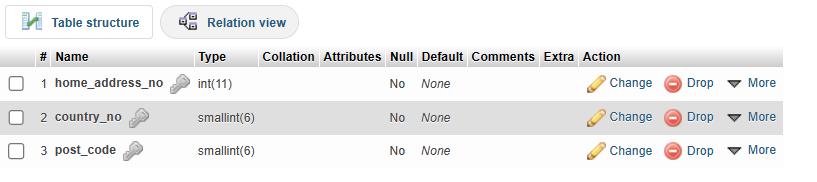
income\_deduction\_type\_no INT NOT NULL CHECK(income\_deduction\_type\_no > 0 ),

income\_deduction\_number INT NOT NULL CHECK(income\_deduction\_number >= 0 )

);

### home\_address\_no

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE home\_address\_no(

home\_address\_no INT NOT NULL UNIQUE CHECK(home\_address\_no > 0 ),

country\_no SMALLINT NOT NULL UNIQUE CHECK(country\_no > 0 AND country\_no < 1000),

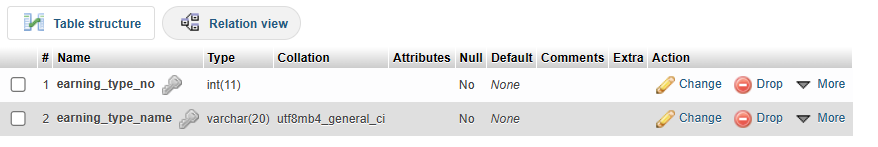
post\_code SMALLINT NOT NULL UNIQUE CHECK(post\_code > 0 AND post\_code < 1000)

);

## 對照表

### earning\_type\_mapping

#### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE earning\_type\_mapping(

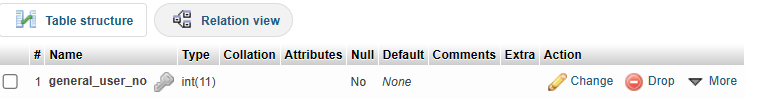
earning\_type\_no INT NOT NULL UNIQUE CHECK(earning\_type\_no > 0 ),

earning\_type\_name VARCHAR(20) NOT NULL UNIQUE

);

## 綱要 (只針對使用者或其權限)

### general\_user



#### table structure

##### SQL syntax

###### CREATE TABLE

CREATE TABLE general\_user( general\_user\_no INT NOT NULL UNIQUE CHECK(general\_user\_no > 0 ) );

### administrator

#### Table structure



##### SQL syntax

###### CREATE TABLE

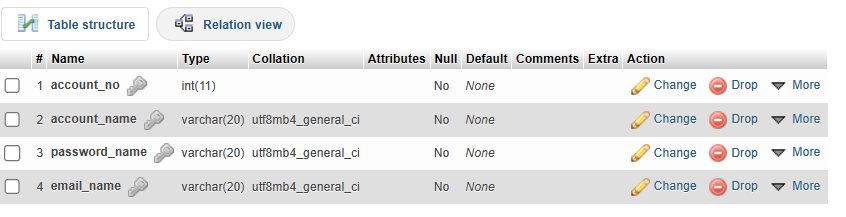
CREATE TABLE `administrator` (

`administrator\_no` int(11) NOT NULL UNIQUE CHECK (`administrator\_no` > 0),

)

### administrator

#### Table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE account\_list(

account\_no INT NOT NULL UNIQUE CHECK(account\_no > 0 ),

account\_name VARCHAR(20) NOT NULL UNIQUE CHECK(account\_name != CHAR(0x10) ),

password\_name VARCHAR(20) NOT NULL UNIQUE CHECK(password\_name != CHAR(0x10) AND LENGTH(password\_name) >=8 ),

email\_name VARCHAR(20) NOT NULL UNIQUE CHECK(email\_name != CHAR(0x10) AND LENGTH(email\_name) >=8 )

);

###### CREATE TRIGGER

CREATE TRIGGER `email\_name\_validation` BEFORE INSERT ON `account\_list`

FOR EACH ROW BEGIN

IF NOT(NEW.email\_name LIKE '%@gmail.com' OR

NEW.email\_name LIKE '%@gm.nfu.edu.tw')

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid email!';

END IF;

END

CREATE TRIGGER `try\_login\_result` BEFORE INSERT ON `account\_list`

FOR EACH ROW BEGIN

IF NEW.account\_name = NEW.password\_name

THEN

UPDATE try\_login SET try\_login.try\_login\_times = 0;

UPDATE try\_login SET try\_login.prev\_success\_login\_time = NOW();

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'login fail!';

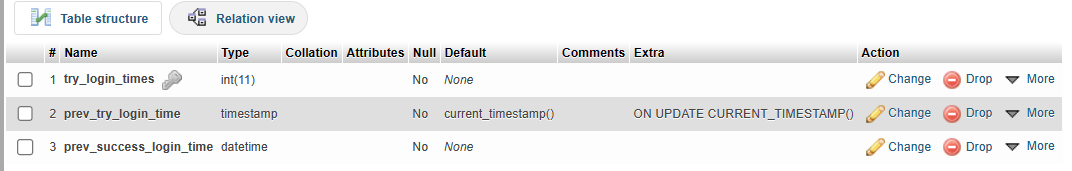
UPDATE try\_login SET try\_login.prev\_try\_login\_time = try\_login.prev\_try\_login\_time + 1;

END IF;

END

### try\_login

#### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `try\_login` (

`try\_login\_times` int(11) NOT NULL,

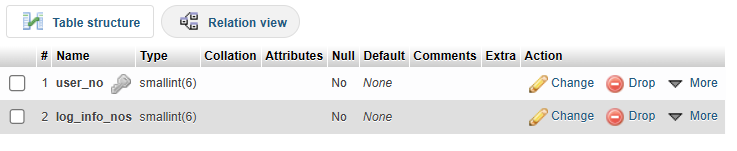
`prev\_try\_login\_time` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp(),

`prev\_success\_login\_time` datetime NOT NULL

)

### log\_info\_list

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE log\_info\_list(

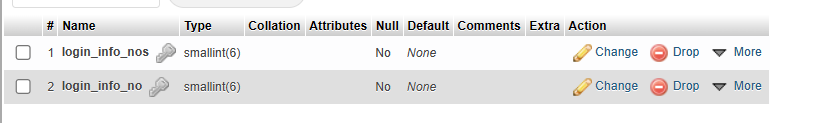
user\_no SMALLINT NOT NULL UNIQUE CHECK(user\_no > 0 ),

log\_info\_nos SMALLINT NOT NULL CHECK(log\_info\_nos > 0 )

);

### log\_info\_nos

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `login\_info\_nos` (

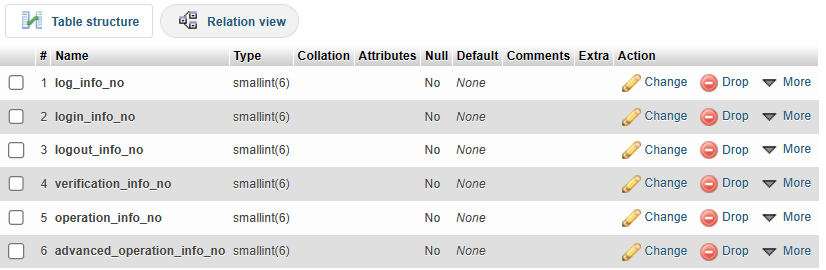
`login\_info\_nos` smallint(6) NOT NULL UNIQUE CHECK (`login\_info\_nos` > 0),

`login\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`login\_info\_no` > 0)

)

### log\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `log\_info` (

`log\_info\_no` smallint(6) NOT NULL,

`login\_info\_no` smallint(6) NOT NULL CHECK (`login\_info\_no` > 0),

`logout\_info\_no` smallint(6) NOT NULL CHECK (`logout\_info\_no` > 0),

`verification\_info\_no` smallint(6) NOT NULL CHECK (`verification\_info\_no` > 0),

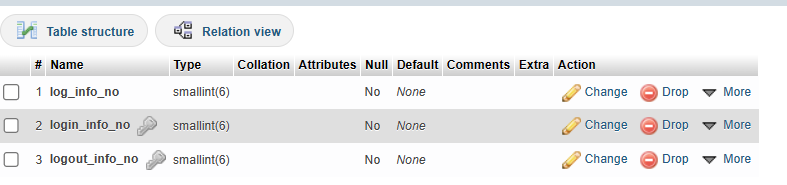
`operation\_info\_no` smallint(6) NOT NULL CHECK (`operation\_info\_no` > 0),

`advanced\_operation\_info\_no` smallint(6) NOT NULL CHECK (`advanced\_operation\_info\_no` > 0)

)

### login\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `login\_info` (

`login\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`login\_info\_no` > 0),

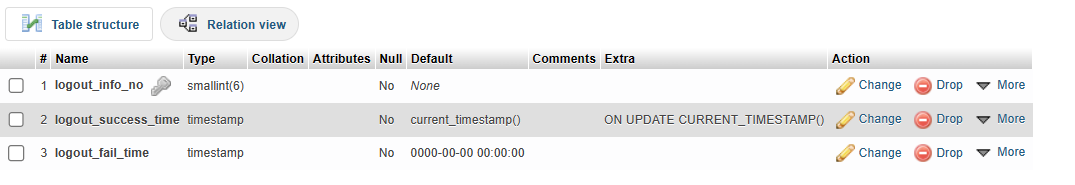
`login\_success\_time` timestamp NOT NULL,

`login\_fail\_time` timestamp NOT NULL

)

### logout\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `logout\_info` (

`logout\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`logout\_info\_no` > 0),

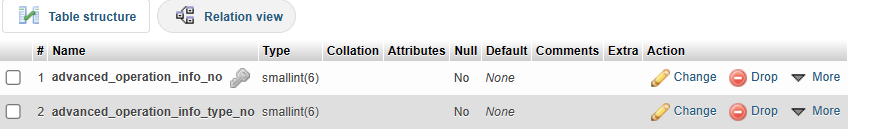
`logout\_success\_time` timestamp NOT NULL,

`logout\_fail\_time` timestamp NOT NULL

)

### advanced\_operation\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `advanced\_operation\_info` (

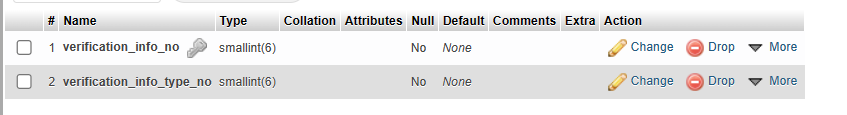
`advanced\_operation\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`advanced\_operation\_info\_no` > 0),

`advanced\_operation\_info\_type\_no` smallint(6) NOT NULL CHECK (`advanced\_operation\_info\_type\_no` > 0)

)

### verification\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `verification\_info` (

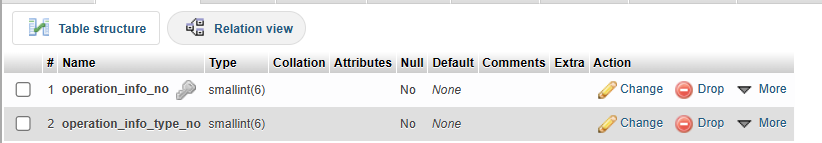
`verification\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`verification\_info\_no` > 0),

`verification\_info\_type\_no` smallint(6) NOT NULL CHECK (`verification\_info\_type\_no` > 0)

)

### verification\_info

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `operation\_info` (

`operation\_info\_no` smallint(6) NOT NULL UNIQUE CHECK (`operation\_info\_no` > 0),

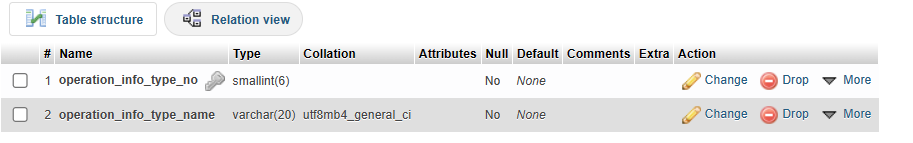
`operation\_info\_type\_no` smallint(6) NOT NULL CHECK (`operation\_info\_type\_no` > 0)

)

## 對照表

### operation\_info\_mapping

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `operation\_info\_mapping` (

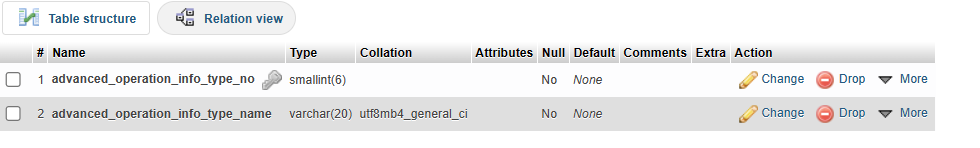
`operation\_info\_type\_no` smallint(6) NOT NULL UNIQUE CHECK (`operation\_info\_type\_no` > 0),

`operation\_info\_type\_name` VARCHAR(20) NOT NULL CHECK (`operation\_info\_type\_name` != CHAR(0x10) )

)

### advanced\_operation\_info\_mapping

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `advanced\_operation\_info\_mapping` (

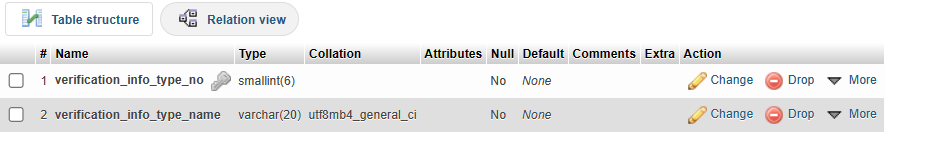
`advanced\_operation\_info\_type\_no` smallint(6) NOT NULL UNIQUE CHECK (`advanced\_operation\_info\_type\_no` > 0),

`advanced\_operation\_info\_type\_name` VARCHAR(20) NOT NULL CHECK (`advanced\_operation\_info\_type\_name` != CHAR(0x10) )

)

### verification\_info\_mapping

### table structure



##### SQL syntax

###### CREATE TABLE

CREATE TABLE `verification\_operation\_info\_mapping` (

`verification\_info\_type\_no` smallint(6) NOT NULL UNIQUE CHECK (`verification\_info\_type\_no` > 0),

`verification\_info\_type\_name` VARCHAR(20) NOT NULL CHECK (`verification\_info\_type\_name` != CHAR(0x10) )

)

# 工作分配表

## 第一次作業

|  |  |
| --- | --- |
| 組員 | 任務分配 |
| 黃奕捷 | 想題目、想報告內容、畫ER Model、撰寫報告(所有部分)、報告審核和報告校正。 |
| 另外兩個組員 | 報告再次審核和ER Model再次審核和意見提供。 |
| 報告再次審核和ER Model再次審核和意見提供。 |

## 第二次作業(之一)

|  |  |
| --- | --- |
| 組員 | 任務分配 |
| 黃奕捷 | 創建有關專題的資料庫和表格、將SQL語法貼上至報告等。 |
| 另外兩個組員 | 報告再次審核。 |
| 報告再次審核。 |