**資料庫管理-期末專案**

**指導教授 : 柯士文教授**

**助教: 楊文瀚、陳威良、林筱芙**

**Hashed :** [**https://www.dailycred.com/article/bcrypt-calculator**](https://www.dailycred.com/article/bcrypt-calculator)

**salt :** [**https://www.grc.com/passwords.htm**](https://www.grc.com/passwords.htm)

➢ 作業請繳交以下檔案**(檔名請務必都寫對)**

**ER Diagram (PDF 檔)**

(檔名:db\_學號.pdf ,例如:db\_108000222.pdf)

**ER Model 匯出檔 (.mwb 檔)**

(檔名:db\_學號.pdf ,例如:db\_108000222.mwb)

**DataBase 的 schema (.sql 檔)**

(檔名:db\_學號.sql, 例如:db\_108000222.sql)

● 請將自己試想為 DBA(Database Administrator),依照以下需求,設計資料庫中的

**ER-Model,tables 與 Stored Procedures**。

● 以下為資料庫中的 stored procedures 內容與描述,總共會有 12 個 stored procedures(其中

兩個為加分題)。**請依照以下 stored procedures 的名字做相同大小寫的命名**。**(若命名不一 樣,直接不予給分)**

### 加分題為[7] , [12] 而[7]這個加分題是指可以選作[6] or [7] 其中一個繼續做下去

### [1].sp\_GetStation

**透過 sp\_GetStation來得到Station table中的全國各高鐵站的 ID 以及名字。**

input: 無  
 output: number of rows in the result set (int)

Result set:

|  |  |
| --- | --- |
| station\_id | station\_name |
| 1 | 台北站 |
| 2 | 桃園站 |

### [2]. sp\_RegisterUser

**透過 sp\_RegisterUser( 會員的信箱,email,姓氏,名字,生日,salt ,加密後的 password )將Input輸入後於User table進行會員註冊。並顯示此次新增的Row number。**

input: email(varchar(100)),firstname(varchar(100)), lastname(varchar(100)), Dob(date),

salt(char(64)), hashedPwd(**char(200)**)

output: affected\_row\_num (int(100)) →被影響的列，為了確認store procedure有效的執行

**註:userID-->auto\_increment**

### [3]. sp\_ResetNewPassword

**透過 sp\_ResetNewPassword(userID,加密後的 password) 可以更改會員的password**

input: userID(int(10)), **salt (char(64))**, hashedPwd(**char(200)**)

output: affected\_row\_num (int(100)) →被影響的列，為了確認store procedure有效的執行

### [4]. sp\_UserLogin

**透過 sp\_UserLogin(會員的電子信箱,雜湊後的 password),若成功則進入系統。**

input: email (varchar(200)) ,hashedPwd(**char(200)**)

output : row\_num(int(100)) →select出來的列

### [5].sp\_DeleteUserByUserID

**透過 sp\_DeleteUserByUserID(會員 ID),進行刪除此會員。要先檢查ticket table 有無此user的資料，若有的話，就不能刪除此會員，反之則可。**

input: user\_id(int(10))

output :affected\_row\_num(int(100)) → 被影響的列，為了確認store procedure有效的執行

### [6]. sp\_QueryTicket

### **透過 sp\_QueryTicket (出發時間,起始站,終點站),可以獲得多個符合篩選條件的車次，起訖站名，以及出發時間，抵達時間。而price的計算可以用station的location\_marker相減的絕對值來獲得，要注意的是要去看該日期該車次是否還有座位。**

**(舉例:台北的location\_marker=1,台中的location\_marker=8,台北到台中的price計算方式為:(8-1)\*100=700)**

input: departure\_time(datetime),departure\_station(int),arrival\_station(varchar(100))

output : num\_row(int(100)) →select出來的列數

Result set:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| train\_id | departure\_station | arrival\_station | departure\_time | arrival\_time | price |
| 3 | 1 | 4 | 2020/01/01 10:00 | 2020/01/01 11:30 | 700 |
| 18 | 1 | 3 | 2020/01/01 13:00 | 2020/01/01 13:30 | 300 |

### [7]. sp\_QueryTicketWithSeatManagement(加分題)

**透過 sp\_QueryTicketWithSeatManagement (出發時間,起始站,終點站),可以獲得多個符合篩選條件的車次及座位，出發時間以及抵達時間 。與[6] sp\_QueryTicket的差別在於座位管理機制，同台車同個座位，在不同站若無人購買就可以被售出，也就是在[7]Query車票時就安排座位，產生ticket。**

**(舉例:今天有人從台北坐到台中就下車，她下車後的那個位置就可以被賣出，但在[6]的題目我們假設同個車次的所有位子只能被賣出一次)**

input:departure\_time(datetime),departure\_station(int),arrival\_station(int)

output:num\_row(int(100)) →select出來的列數

Result set:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| train\_id | departure\_station | arrival\_station | seat\_id | departure\_time | arrival\_time | price |
| 3 | 1 | 4 | 1A | 7:00 | 08:00 | 700 |

### [8]. sp\_BookTicket 要考量到加分題的問題

**透過 sp\_bookTicket (會員 ID,車次 ID) ,將此加入到ticket table中,同時紀錄訂票時間(book\_time)。 並且，訂完票後會給他一個 ticket\_id，pay\_time指的是付款時間，若還未付款就不會有值，座位是隨機分配(非加分題)，(加分題)用戶在[7]選擇座位。**

input: user\_id(int(10)) , train\_id (int(11)), departure\_station(varchar(100)),arrival\_station(varchar(100)),seat\_id(varchar(64)), train\_date date

p.s price要用time\_marker算。若做加分題可以input seat\_id,非加分題不可input seat\_id。

output: num\_row(int(100)) →select出來的列數

Result set:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ticket\_id | user\_id | train\_id | departure\_station | arrival\_station | seat\_id | book\_time | train\_date |
| 222 | 8 | 3 | 1 | 4 | 1A | 2019/12/25 20:00 | 2019/12/25 |

### [9]. sp\_GetBookTicketByUser

**透過 sp\_GetBookTicket(會員 ID),可以獲得其使用者還未付款的訂票紀錄(要去判斷pay\_time是否為空)。**

input: userID (int(10))

output :num\_row(int(100)) →select出來的列數

Result set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ticket\_id | userID | train\_id | seat\_id | book\_time |
| 222 | 8 | 3 | 1A | 2019/12/25 20:00 |
| 238 | 5 | 18 | 2A | 2019/12/25 12:00 |

### [10]. sp\_PayTicket(join table)

**先用ticket table的book\_time判斷商業邏輯(3天內包含當天、發車前30分鐘不能付款)，若符合商業邏輯才可付款。執行 sp\_PayTicket(會員 ID,車票 ID)要將ticket table中的paytime新增付款時間。**

**最後必須利用ticket\_id讓 train table& ticket table進行 join table的動作，並輸出以下欄位**

input: user\_id(int(10)) , ticket\_id(int(11))

output: affected\_row\_num(int(100)) → 被影響的列，為了確認store procedure有效的執行

Result set:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ticket\_id | user\_id | train\_id | departure\_station | arrival\_station | seat\_id | book\_time | departure\_time | arrival\_time | price | pay\_time |
| 8 | 4 | 3 | 1 | 4 | 1A | 2020/12/25 20:00 | 2020/01/01 07:00 | 2020/01/01 07:30 | 700 | 2020/12/28 10:00 |

### [11]. sp\_Delete\_bookticket

**透過 sp\_Delete\_book\_ticket(user\_ID,使用者訂票的ticket\_ID),將該車票取消訂購，從Ticket table中刪除。要注意的是刪除過後的座位(seat\_id)是可以再次被訂購的。已被付款的票不能被刪除**

input: user\_id(int(10)) , ticket\_id (int(11))

output :affected\_row\_num(int(100)) → 被影響的列，為了確認store procedure有效的執行

### [12]. sp\_TrainManagement (加分題)

**管理者身份，透過 sp\_TrainManagement，針對某車班更動發車時間、起始站、最終站，(只能更動14天後的車班)。若起始站、最終站有更動，要修改為對應的價錢。**

**提示: 新增一班車次，可以透過on\_date跟off\_date去控制。**

**(舉例：若要train\_id=1的車班在2020/06/07後更改為8點從台北發車，則該班次的off\_data＝2020/06/07，表示該車班在2020/06/07後取消，同時新增車班train\_id=7，並新增車班的on\_data＝2020/06/07，表示在2020/06/07後開始該車班。需注意arrival\_time、departure\_time的時間要由station table的time\_marker推算，此方式等於是預計關掉一班次，再預計新增一班次)**

input:

train\_id(20),off\_date(date),departure\_time(datetime),departure\_station(varchar(100)),arrival\_station(varchar(100)),on\_date(date)

output:affected\_row\_num(int(100)) → 被影響的列，為了確認store procedure有效的執

p.s. input內的 train\_id,off\_date 是輸入要關掉的班次資訊，其餘是輸入新增的班次的欄位。（on\_date:建立車班的日期,off\_date:結束車班的日期）

Result set:

train\_id:新增的train\_id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| train\_id | departure\_time | arrival\_time | departure\_station | arrival\_station | on\_date | off\_date |
| 5 | 2020/02/02 07:20 | 2020/02/02 07:30 | 2 | 3 | 2020/02/02 | 2020/02/03 |