

# Digital Online Music System

## Database Management Project

Group 9

B06705019 資管三 黃勛哲

B06705021 資管三 許亦佑

B06705024 資管三 郭宇軒

B06705034 資管三 吳禹辰

---

### 動機

我們是一群喜歡聽音樂的大學生，但在使用 **kkbox**，普遍大學生都有用音樂 **APP** 聽音樂的習慣，不論是在讀書、通勤或是慢跑的時候，有著美妙的樂音陪伴，總是維持著快樂的心情。而現在流行的音樂平台諸如 **KKBOX**、**Spotify** 或 **Street Voice**...等雖然功能都已經相當完善，但是在為獨立創作者的平台優化上感覺還稍嫌不足，所以我們希望可以設計一個客製化功能為特色的音樂平台，讓使用者們能夠分享自己的 **Playlist**，以及上傳自己所演唱或是製作的歌曲，並且分享給其他使用者聆聽。

### 系統架構介紹

#### Functional requirement

# User (UID, UName, membership, register\_time)

create\_user(Uname, UID, register\_time)

set\_Uname(UID, Uname)

set\_user\_membership(UID, membership)

have\_playlist(PID, UID)

delete\_user(UID)

#Listen (UID, ID, Time, Date)

create\_listen(UID, ID, Time, Date)

#Song (ID, Name, release time, genre, SID)

create\_song(ID, Name, release time, genre, SID)

set\_song\_Genre(ID, Genre):

like\_song( ID, like\_song)

#Singer(SID, SName, like\_singer, UID, is\_Band)

create\_singer(SName, SID, UID)

like\_singer( ID, like\_singer)

#In\_Band(SID, B\_SID)

create in\_band(SID, B\_SID)

join\_Band(SID, B\_SID)

#Album(AID, AName, publish\_time)

create\_album(AID, AName, publish\_time)

#PlayList(PName, PID, creator)

create\_playlist(PName, PID, creator)

delete\_playlist(PID):

#Have\_playlist(PID, UID)  
having\_playlist(PID, UID)  
drop\_playlist(PID, UID)

#Son\_inPlaylist(ID, PID)  
add\_song(ID, PID)  
drop\_song(ID, PID)

#Board(Board\_ID, Board\_ID)  
create\_board(Board\_ID, Board\_Name)

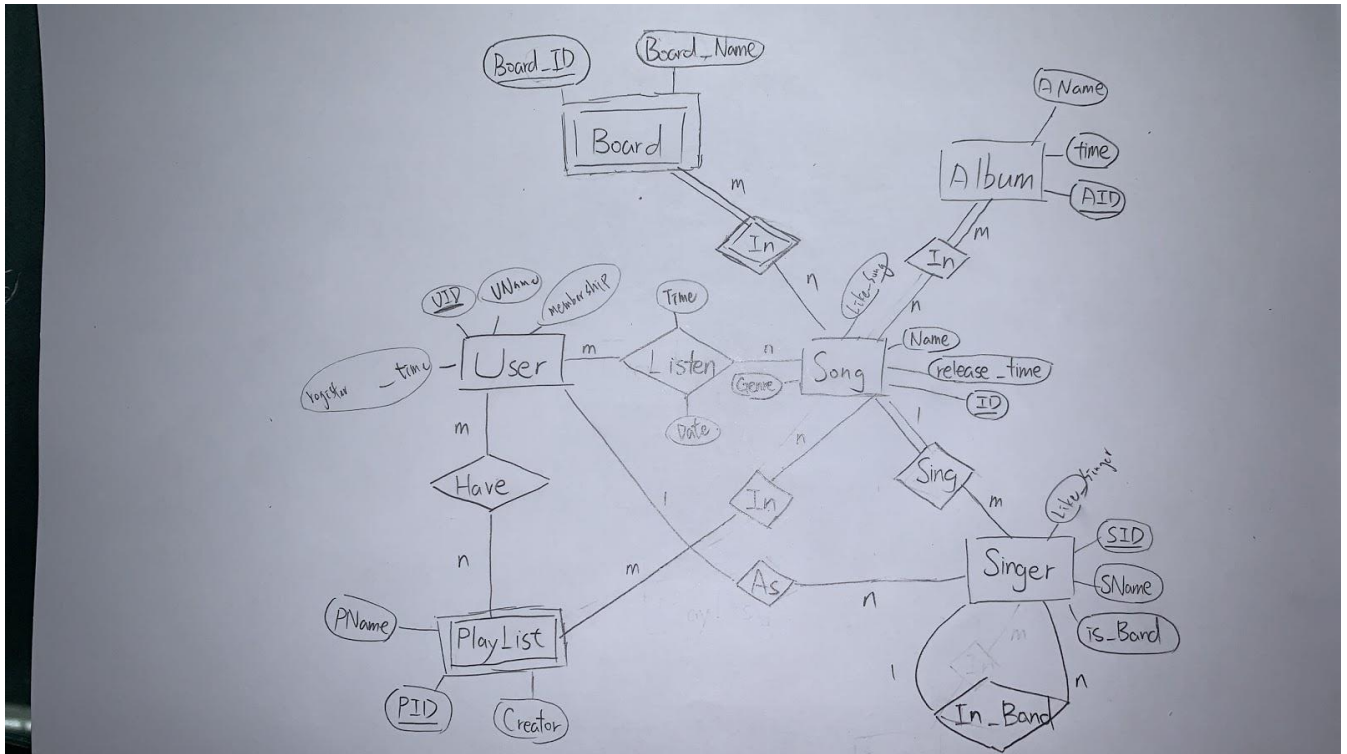
#Son\_inAlbum(ID, AID)  
add\_song(ID, AID)

#Son\_inBoard(ID, BID)  
add\_song(ID, BID)

#### Data requirement

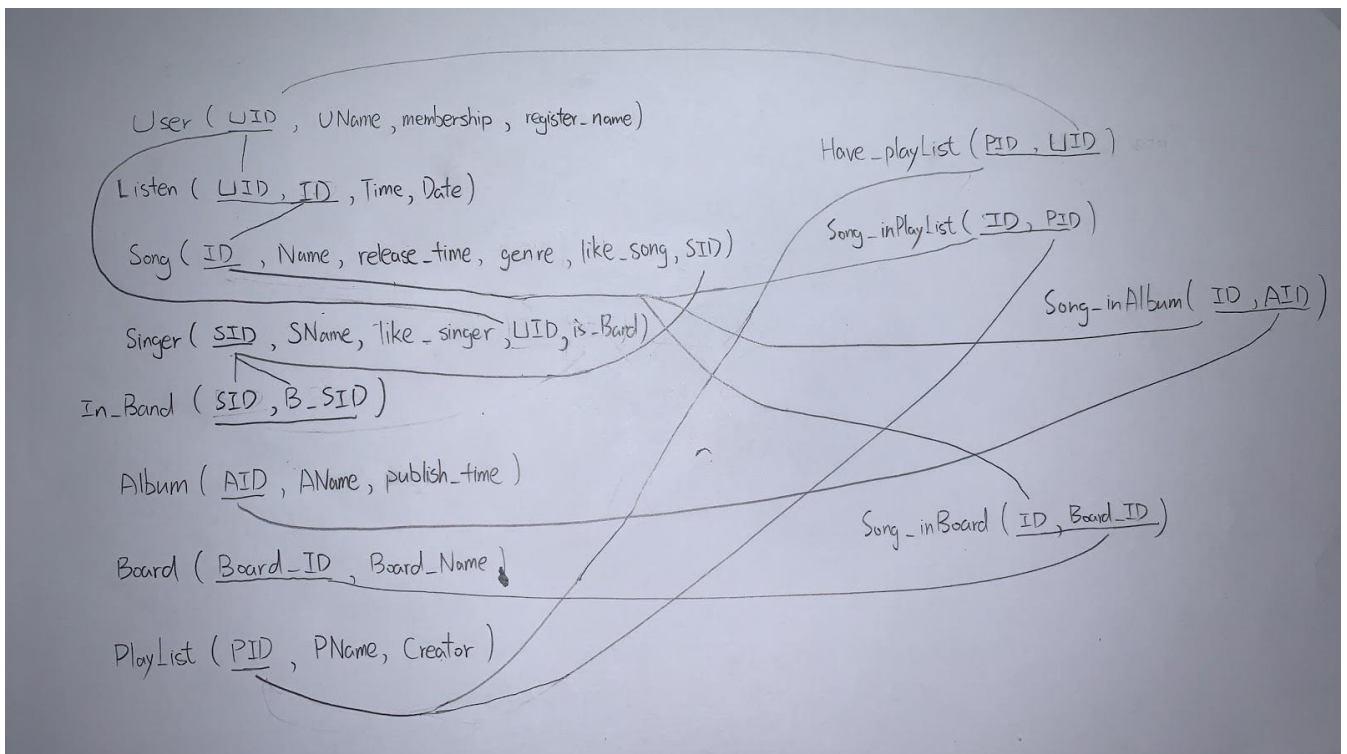
每個 user 都有一個唯一的 uid、user name、會員等級、註冊時間。每位 user 可以有 multiple playlist，同時可以把 playlist 分享給其他的 user，Playlist 中也會以 PID、PName、Creator 紀錄。User 可以 listen to song，listen 會紀錄 user 使用的日期跟時間。

每一首 Song 都會有唯一的 ID、Genre、Name、Release\_time、Like\_song，同時 Song 也會在 Album 中(in)，Album 中則會紀錄 AName、time、AID。Song 也會 related 到 Singer，Singer 會紀錄 SID、SName、Like\_singer、is\_band 等資料，其中 is\_band 為 boolean attribute，如果為 True 時，則會有 Band relation 可以紀錄此歌手為哪一樂團成員及查看此樂團成員。Singer 也可以 related 到 User(User as Singer)，代表每一位平台使用者除了收聽其他人的作品外，也可以發布自己的作品讓人聆聽。Board 則會 related 到 Song 再依據不同的類別挑選內容物並以 Board\_ID、Board\_Name 紀錄。



ER Diagram

DB schema



## 系統功能介紹

系統介面： r Shiny 套件製作出使用者介面  
後端:SQL workbench 資料庫  
再利用 rstudio 連結後端資料庫及使用者介面

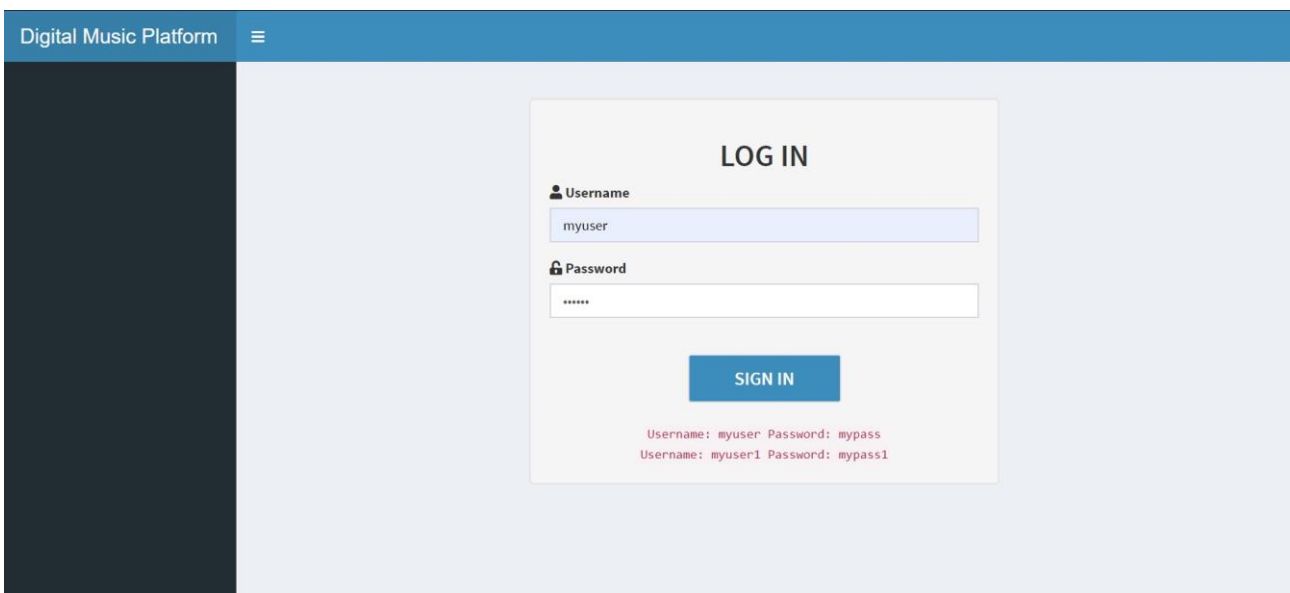
使用者介面：

- 1.使用者登入畫面
- 2.Insert 新增音樂
- 3.Select 搜尋 album 即可獲得所有在專輯內資料
- 6.建立自己的 playlist

一開始先進入登入畫面，登入後每一個使用者可以去瀏覽音樂資料庫(song)，也可以藉由不同的專輯、不同的排行榜來瀏覽音樂。

如果有使用者想要上傳自己發表的新單曲，則可以在新增(Insert)時新增在 song 中，也可以加上專輯名稱加入到已知的專輯當中。

每個使用者可以在介面中瀏覽其他使用者新增的歌曲，或是資料庫中原有的資料，決定要不要加入到自己的 playlist 中。



The screenshot shows a web application titled "Digital Music Platform" with a blue header bar. On the left, there is a dark blue sidebar. The main content area is light blue and contains a white login form titled "LOG IN". The form has two input fields: "Username" with the value "myuser" and "Password" with masked characters. Below the fields is a blue "SIGN IN" button. At the bottom of the form, there are two lines of text: "Username: myuser Password: mypass" and "Username: myuser1 Password: mypass1".

登入畫面：

Digital Music Platform
Logout
Main Page

Columns in song to show:
☒ ID
☒ name
☒ release\_time
☒ genre
☒ SID
☒ Sname
☒ Board\_ID
☒ AID

Click the column header to sort a column.

Display 5 records by default.

Find IDInsert playlistInsert SongShow AlbumInsert song to playlist

Show 5 entriesSearch:

	ID	name	release_time	genre	SID	Sname	Board_ID	AID
1	1001	Happy	2020-01-10 00:00:00	a	1	Leo	100000	101
2	1002	Red	2017-01-10 00:00:00	b	2	Taylor	100000	201
3	1003	Sucker	2018-05-14 00:00:00	c	3	Jonus	100000	301
4	1004	Joe	2020-01-15 18:47:58	a	1	Leo		101
5	1005	dick	2020-01-15 18:53:04	a	1	Leo		101

Showing 1 to 5 of 11 entriesPrevious123Next

Enter song name:  
new song

新增歌曲成功畫面：

Digital Music Platform
Logout
Main Page

Columns in song to show:
☒ ID
☒ name
☒ release\_time
☒ genre
☒ SID
☒ Sname
☒ Board\_ID
☒ AID

Click the column header to sort a column.

Display 5 records by default.

Find IDInsert playlistInsert SongShow AlbumInsert song to playlist

Show 10 entriesSearch:

	ID	name	release_time	genre	SID	Sname	Board_ID	AID
1	1001	Happy	2020-01-10 00:00:00	a	1	Leo	100000	101

Showing 1 to 1 of 1 entriesPrevious1Next

Enter song name:  
Happy

search

利用歌曲名稱找尋歌曲 ID 畫面：

Function:

select\_al:從 album 中搜尋所有在 album 的歌曲

```

19 # 從專輯名找歌曲
20 select_al<- function(x){
21   # "select name from song where AID = (select AID from albums where Aname = 'Suck')"
22   dbGetQuery(connect, str_c("select name from song where AID = (select AID from albums where Aname = '", "''", sep = x))
23 }
24 # 加入新歌輸入歌名、類型、SID、AID
25 insert_song <- function(name, genre, SID, AID){
26   song = dbGetQuery(connect, "select * from song")
27   dbSendQuery(connect, str_c("insert into song(ID, name, release_time, genre, SID, Sname, AID) values ('",
28     as.character(max(song$ID)+1),"',", name,"',", Sys.time(),"',", genre,"',", SID,"',",
29     dbGetQuery(connect, str_c("select Sname from Singer where SID =", SID)),",", AID,"')"))
30   song = dbGetQuery(connect, "select * from song")
31 }

```

insert\_song: 加入新的歌曲

```

# 新增playlist
insert_playlist<-function(pname, UID){
  playlist = dbGetQuery(connect, "select * from playlist")
  dbSendQuery(connect, str_c("insert into playlist(PID, Pname, UID) values ('", as.character(max(playlist$PID)+1),"',",
    pname,"',", UID,"')"))
  playlist = dbGetQuery(connect, "select * from playlist")
}

```

insert\_playlist: 使用者新增一個新的 playlist

```

#把歌加進playlist
insert_song_to_playlist <- function(PID, ID){
  dbSendQuery(connect, str_c("insert into song_in_list(PID, ID) values ('", PID,"',", ID,"')"))
}

```

insert\_song\_to\_playlist: 從 song 資料庫中新增歌曲到自己資料庫中

find\_id: 當需要新增歌曲資料時，先查詢需要的歌曲 ID（同時也增加修改 playlist 的難易

```

#查詢歌的id
find_id <- function(songName){
  data.frame(ans)
  for(i in 1:nrow(song))
  {
    if(song[2,i] == songName)
    {
      cbind(song[,i])
    }
  }
}

```

度，避免使用者誤刪或誤植）

find\_AID:當需要新增歌曲資料時，先查詢需要的專輯 ID（同時也增加修改 playlist 的難易

```
#查詢album的id
find_AID <- function(albumName){
  ans <- data.frame()
  for(i in 1:nrow(albums))
  {
    temp <- as.data.frame(albums[i,])
    if(albums[i,2] == albumName)
    {
      ans <- rbind(ans, temp)
    }
  }
  return(ans)
}
```

度，避免使用者誤刪或誤植）

資料庫樣貌：

根據 ER diagram 建立 table，設立 primary key 以及 foreign key 等等。

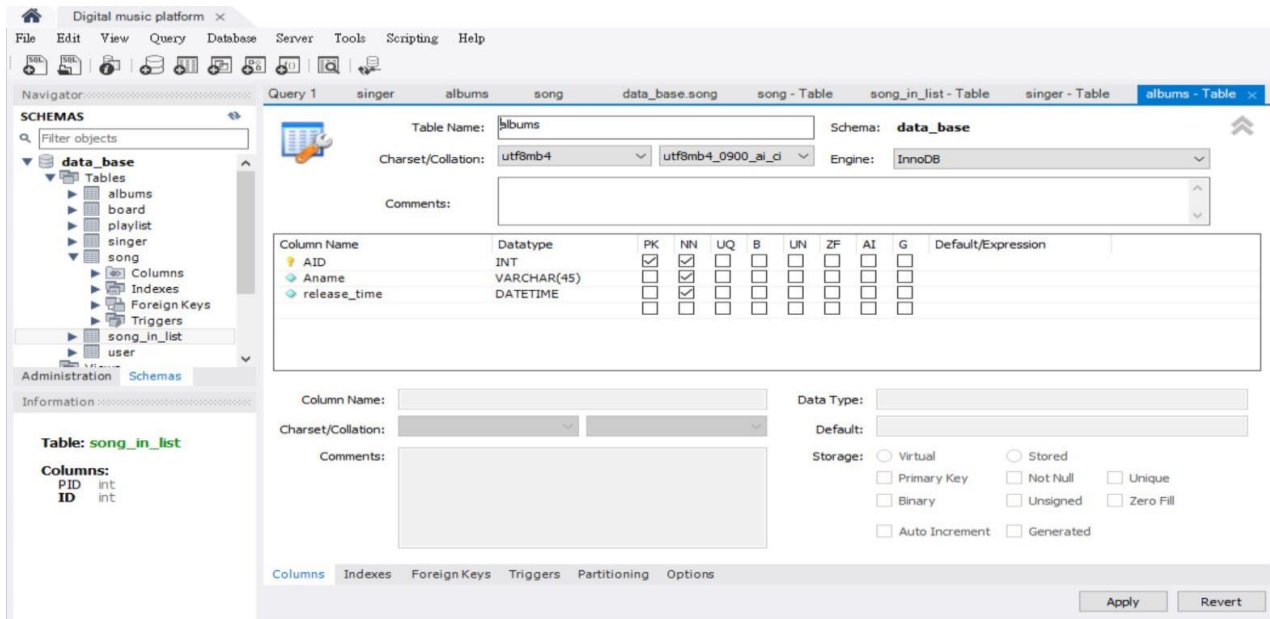
例如：table Song 得 Primary Key 為 ID, SID、AID 則為 foreign key，連結到的 table 則分別為

The screenshot shows a database management interface for a table named 'song'. The table is located in the 'data\_base' schema. The charset/collation is set to 'utf8mb4' and 'utf8mb4\_0900\_ai\_ci', and the engine is 'InnoDB'. Below the table configuration, there are two tables showing foreign key relationships.

Foreign Key Name	Referenced Table
AID	`data_base`.`albums`
SID	`data_base`.`singer`

Column	Referenced Column
<input type="checkbox"/> ID	
<input type="checkbox"/> name	
<input type="checkbox"/> release_time	
<input type="checkbox"/> genre	
<input checked="" type="checkbox"/> SID	SID
<input type="checkbox"/> Sname	
<input type="checkbox"/> Board_ID	
<input type="checkbox"/> AID	

singer 和 albums，如圖所示。



## 未來展望

1. 希望可以增加更多功能，比如讓不同使用者間分享自己的 **playlist**，或是增加 **playlist** 的使用權限，讓更多人可以一同修改使用。
2. 可以更加優化使用者介面，加入更多擴充功能。
3. 定時更新排行榜資料，根據一定時間區間中，使用者聆聽歌曲的上榜數。
4. 新進歌曲告示榜：將在一說間內剛上線的音樂縣市於此，幫助使用者得知音樂新發佈消息。

## 心得

B06705034 吳禹辰

在這個 **project** 中我負責的是前端和串接前端還有後端的工作和一些後端，其實我一開始以為這份工作會很難做而且很麻煩，因為前端和後端畢竟是由不同的人完成，格式、**coding** 習慣的不同會導致銜接上的困難，不過因為在寫後端的過程中其實是大家一起互相討論一起作業的，所以後來在串接我寫的前端和後端時，並沒有原本想像的那麼困難。這次專案中學到最多的部分大概就是串接前後端的過程，這份工作其實在每個專案中都扮演著極其重要的一個齒輪的腳色，為了使整組構造能順利運作，中間的齒輪必須要去和他連接的其他組件、齒輪完美相接，就如串接的工作，要適切的了解後端的需求和前端的介面，再把他們



完美的結合成最終的資料庫系統專案。這次的專案讓我團體合作專案的能力有所提升，不再是簡單的個人專案，而是較為複雜的多人專案。

#### B06705019 黃勛哲

動機，可以說是單純的力量，伏爾泰曾講過，在理想的最美好的世界中一切都是為最美好的目的而設。這讓我的思緒清晰了，在這種不可避免的衝突下，我們必須解決這個問題。當前最急迫的事，想必就是釐清疑惑了。探討期末報告時，如果發現非常複雜，那麼想必不簡單，我們一般認為，抓住了問題的關鍵，其他一切則會迎刃而解。看看別人，再想想自己，會發現問題的核心其實就在你身旁，在這次的期末報告中，我寫的是用 **MYSQL** 建資料庫的部分，而這部分是由我和另一位同學共同完成的，其實在過程中並不一定都那麼順利，我們兩個在作業中期曾為了資料庫的 **relation** 方式的意見不同有了些爭執，不過在靜下心來好好的談過後，不但不計前嫌，接下來我們合作的速度也大為提升，這次的專案讓我學到最多的大概就是如何化解團隊專案中的爭執了。

#### B06705024 郭宇軒

所謂期末報告，關鍵是期末報告需要如何解讀。本人也是經過了深思熟慮，在每個日日夜夜思考這個問題，想必大家都能了解期末報告的重要性。在這種不可避免的衝突下，我們必須解決這個問題。在這次專案中，我和另一位組員一起負責後端的部分，實際應用上課教到的創建和搜尋資料庫的函式還有創建各關連表間的關聯真的是挺有趣的，可惜的是跟這次的過程中和其他的組員發生了一些摩擦，不過最後也是和平化解了，曾有人說“持以坦白的態度，出以誠懇的目的”正如吃苦藥治病一樣。會這麼說是有理由的，雖然有點爭執但大家都是出自於同樣的初衷，即想要做好專案的心情，所以以誠相待後，小小的摩擦便能迎刃而解，同心協力完成更好的專案，總的來說，期末專案讓我知道怎麼學以致用以及團隊合作的模式。

#### B06705021 許亦佑

總而言之，當你搞懂後就會明白了。資料庫期末專案因何而發生？在這種不可避免的衝突下，我們必須解決這個問題。想必各位已經看出了其中的端倪。有一句發人省思的話，春天不播種，夏天就不會生長，秋天就不能收割，冬天就不能品嚐。但願諸位理解後能從中有所成長。但可惜的是，在最後大家期末考完要開始準備資料庫專案的時候，我正好和系羽前往台中打大資盃，所以只好讓我來最一些文本處理了，我覺得研究資料庫的那些關聯，還有指來指去的告種鍵，實在是相當有趣！所以覺得蠻可惜沒有參與到資料庫專案的建立，只有剛開始時有參與討論專案主題，希望自己以後可以在事前就先把所有的事件都排上行事曆，就不會發生這種所有事情都擠在一起的悲劇了。