午餐系統及點餐資料分析

[文件副標題]

[壹、研究動機 3](#_Toc531644022)

[貳、研究目的 3](#_Toc531644023)

[參、實驗器材及設備 3](#_Toc531644024)

[肆、研究過程及方式 4](#_Toc531644025)

[一、資料庫 4](#_Toc531644026)

[1. 點單查詢 5](#_Toc531644027)

[2. Procedure優化 5](#_Toc531644028)

[二、後端 5](#_Toc531644029)

[(一)、處理流程 6](#_Toc531644030)

[(三)、效能阻塞 6](#_Toc531644031)

[1. session 阻塞 6](#_Toc531644032)

[2. sql效能低下 7](#_Toc531644033)

[3. 硬碟存取阻塞 7](#_Toc531644034)

[(四)、安全性 7](#_Toc531644035)

[1. 密碼安全性 7](#_Toc531644036)

[2. SQL注入 7](#_Toc531644037)

[3. XSS 7](#_Toc531644038)

[4. 行政安全性 8](#_Toc531644039)

[5. 資料庫死結 8](#_Toc531644040)

[三、前端 9](#_Toc531644041)

[(一)、iOS前端 9](#_Toc531644042)

[1.呼叫後端 9](#_Toc531644043)

[2.Layout圖 9](#_Toc531644044)

[3.錯誤分析 9](#_Toc531644045)

[4.推播功能 9](#_Toc531644046)

[(二)、網頁前端 9](#_Toc531644047)

[1. ajax 9](#_Toc531644048)

[2. 分類樹 9](#_Toc531644049)

[3. 半耦合式後端 9](#_Toc531644050)

[(三)、輔助外掛 9](#_Toc531644051)

[1. 當日會計報表 9](#_Toc531644052)

[2. 新生資料匯入軟體 9](#_Toc531644053)

[(四)、前後端交換介面 9](#_Toc531644054)

[1. 點單資料結構 9](#_Toc531644055)

[2. 編碼 10](#_Toc531644056)

[四、預測模型 10](#_Toc531644057)

[(一)、資料結構 10](#_Toc531644058)

[1. 點餐序列 10](#_Toc531644059)

[2. 點餐圖 11](#_Toc531644060)

[3. 點餐矩陣 11](#_Toc531644061)

[(一)、宏觀模型 11](#_Toc531644062)

[(二)、微觀模型 11](#_Toc531644063)

[(三)、模型應用 11](#_Toc531644064)

[(二)、宏觀模型 11](#_Toc531644065)

[伍、目前研究結果 13](#_Toc531644066)

[一、系統特色 13](#_Toc531644067)

[二、分析實例 13](#_Toc531644068)

[三、實際案例 13](#_Toc531644069)

[陸、參考資料及其他 14](#_Toc531644070)

# 壹、研究動機

建立一個可供廠商、消費者使用的平台

# 貳、研究目的

可追蹤特定目標族群

可預測明日之點餐數量

與產品經理討論並開發所需功能

# 參、實驗器材及設備

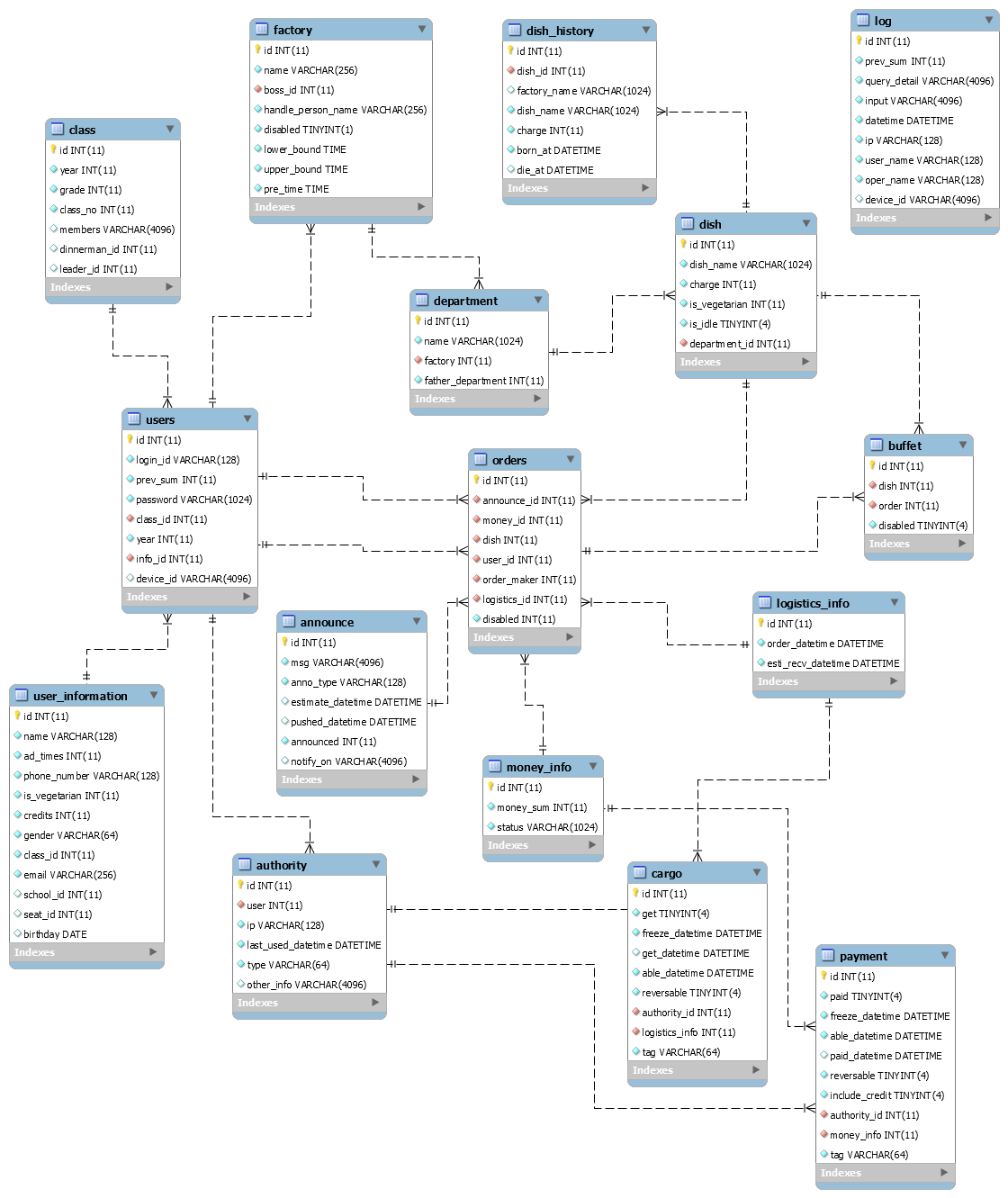
|  |  |  |
| --- | --- | --- |
| 電腦類 | 行政類 | 其他類 |
|  |  |  |
| Firebase | 校務會議支持 |  |
| Dynamic DNS |  |  |
| 伺服器電腦 |  |  |
|  |  |  |
|  |  |  |

# 肆、研究過程及方式

　　午餐系統後端係由php作為後台，mysql作為資料庫，並且輸出到前後端交換介面；凡是能夠操作前後端交換介面者，都可以稱呼為一個前台，包括網頁版前台、iOS前台、Android前台，都可稱為一個前台。

## 一、資料庫

　　下列為午餐系統的資料結構模型，每一條線代表一個關聯性連接。



#### 1. 點單查詢

由於點單的變動率高，直接從資料庫擷取資料較為合適，後台可以根據傳入參數，使用這六種篩選語句*(syntax)*組合成所需的篩選條件。

|  |  |
| --- | --- |
| 時間下界 | AND (? < LO.esti\_recv\_datetime) |
| 時間上界 | AND (? > LO.esti\_recv\_datetime) |
| 針對特定使用者搜尋 | AND (? = U.id) |
| 針對某間廠商進行搜尋 | AND (? = F.id) |
| 針對特定列查詢 | AND (? = O.id) |
| 針對班級查詢 | AND ((SELECT U.class\_id FROM users AS U WHERE U.id = ?) = U.class\_id) |

在下達sql指令時，可以使用代稱*(alias)*來方便撰寫sql語句，以下是篩選語句的解釋。

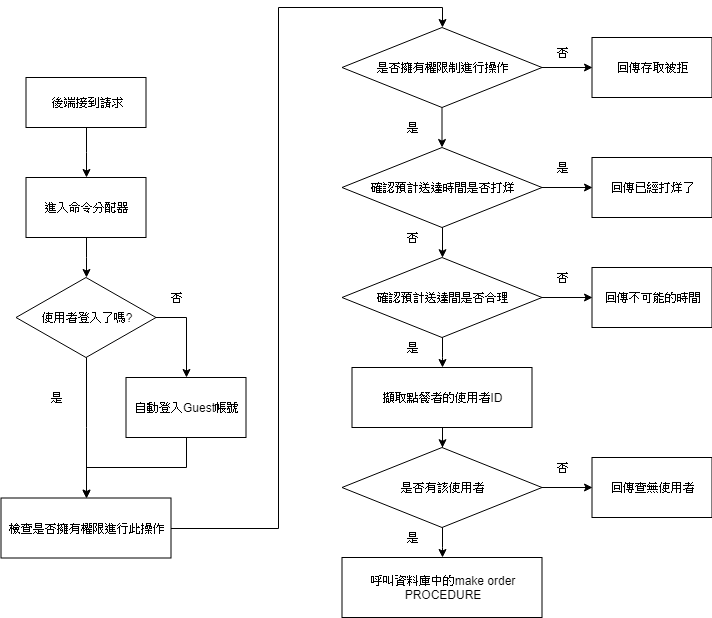
|  |  |  |
| --- | --- | --- |
| 中文名稱 | 英文全名 | 英文縮寫 |
| 使用者 | User | U |
| 物流資訊 | Logistic info | LO |
| 廠商 | Factory | F |
| 點單 | Order | O |
| 預計送達時間 | Estimate receive datetime | Esti recv datetime |

#### 2. Procedure優化

對於單語句*(syntax)*的sql操作，直接在php中呼叫資料庫即可；對於多語句*(syntax)*的sql操作，則包裝成一個Procedure，方便處理死結回溯，也不必與資料庫伺服器多次連線。

## 二、後端

### (一)、處理流程

****

　　以進行點餐操作為例，系統會先進行嚴密的輸入審查，確認是否無誤後，再將資料寫入資料庫。對於大部分的處理程序，皆與點餐操作相似，先進入命令分配器，再進行參數檢查，最後寫入、讀取資料庫。

### (二)、效能阻塞

#### 1. session 阻塞

　　php為了保證執行緒安全，同一個session同時只能給一個request使用。在每個request都會長期持有session的狀況下，同時送出大量request，就無法受益於平行處理；如果每個request都只會短期持有session，不必等待其他request繼續占用session，就可以同步進行操作。

　　session的資料中，大多都是常駐性資料，只有在登入時才會寫入，其他行為皆不會寫入session。為了受益於平行處理，系統經過命令分配器之後，就會決定是否已經可以釋放session了，使得其他request可以更快開始進行處理。

#### 2. sql效能低下

　　如果需要搜尋在「六月九號之後，八月七號之前」這個區間內的點單，若點單的時間有加上索引，我們可以使用二分搜尋法在*O(log N)*的時間內搜尋出資料；在沒有加上索引的情況下，系統沒辦法進行二分搜尋法，則需要*O(N)*搜尋整張資料表。

#### 3. 硬碟存取阻塞

　　硬碟的存取速度絕對低於記憶體的存取速度，如果每次使用常駐資料時，都向資料庫要求一次資料，則每次使用常駐資料都需要啟用硬碟一次。系統將資料先快取於session，需要使用常駐資料時直接從session調用資料，就不必再啟動硬碟一次了。

### (三)、安全性

針對後端安全，我們進行了以下幾種保護措施。

#### 1. 密碼安全性

對於所有工作人員的密碼，皆為六個字元以上的英數混和字串；並且將所有的登入失敗寫入紀錄檔中，若有異常登入資料，立刻通知該使用者。

#### 2. SQL注入

不以舊版的mysql模組操作資料庫，而以新版的mysqli操作資料庫；並且將所有的statement進行prepare後bind\_param，不直接在statement中寫入值，如下列虛擬碼所示。

|  |  |
| --- | --- |
| 具有sql注入風險 | 不具有sql注入風險 |
| $sql=”select id from orders where id = $uid”  $database->execute($sql) | $sql=”select id from orders where id = ?”  $database->prepare($sql)  $database->bind\_param(“1 or True”)  $database->execute() |

#### 3. XSS

在check\_valid 模組中，嚴格限制了每個輸入參數，凡是參數不符合規定，則立刻丟出例外狀況，並寫入記錄檔中。

#### 4. 行政安全性

向學生以及工作人員呼籲必須更改密碼，避免有他人惡意登入。我們選擇在學校的朝會、文字廣播系統上向學生呼籲必須更改密碼；因工作人員人數較少，我們一一協助工作人員更改自己的密碼。

#### 5. 資料庫死結

　　在單線程的測試環境下，很少會遇到資料庫死結，而在系統真正運行的時候，常會遇到不可預知的死結。一個Procedure包裝了多個語句(Syntax)，若是在尚未執行完Procedure前，發生了死結，則可能會有不可預知的後果。

我們決定針對容易發生死結的Procedure加上start transcation、rollback、commit，若是在Procedure尚未結束前發生死結，則回溯(Rollback)整個Procedure進行的操作。

## 三、前端

### (一)、iOS前端

iOS前端為專屬開發給蘋果使用者的操作介面，由於網頁版前端較不美觀，我們開發了專屬於蘋果使用者的前端，該前端符合iOS三大設計原則：「**清晰、遵從和深度」。**

 **以下為iOS前端的登入畫面，我們可以見到，，符合**

#### 1.呼叫後端

#### 2.Layout圖

#### 3.錯誤分析

#### 4.推播功能

### (二)、網頁前端

#### 1. ajax

#### 2. 分類樹

#### 3. 半耦合式後端

### (三)、輔助外掛

#### 1. 當日會計報表

#### 2. 新生資料匯入軟體

### (四)、前後端交換介面

前後端交換介面確保後端與前端的分工，任何能夠操作命令介面，與使用者互動，並且能夠擷取回傳資料，不論語言、平台撰寫成的程式，皆可被視為一個前端；任何能夠根據命令回傳出相對應的程式，不論語言、不論平台，也皆可被視為一個後端。

#### 輸出資料結構

原始輸出為JSON字串，我們將JSON轉換成表格，以利理解，以下表格為精簡後的點單資料結構。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | |  |  | | --- | --- | | **id** | 22410 | | **user** | |  |  | | --- | --- | | **id** | 11184 | | **name** | 白翔云 | | **seat\_no** | 20904 | | | **dish** | |  |  | | --- | --- | |  | | | **dish\_name** | 卡拉雞 雞塊 三副菜 | | **dish\_id** | 1 | | **dish\_cost** | 55 | | | **recv\_date** | 2018-11-11 12:00:00 | |

#### 命令介面

以下表格為精簡後的命令界面，「-」代表沒有參數。

|  |  |  |
| --- | --- | --- |
| Login | Id=[使用者帳號] | Password=[密碼] |
| Show dish | - | - |
| Select self | Esti\_start=[時間上界] | Esti\_start=[時間上界] |
| Set payment | Oid=[點餐單號] | Target=[是否繳款] |
| Make order | Did=[餐點號碼] | Esti\_recv=[預計送達時間] |

#### 3. JSON與編碼

採用UTF-8編碼，包含BOM Header，對於JSON中包含的中文字串不予加以編碼，對於「”」字元更改為「\”」，對於「\」字元更改為「\\」。

## 四、預測模型

### (一)、資料結構

原始的點餐資料可以視為下列表格，「-」代表沒有點餐。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 甲生 | 乙生 | 丙生 | 丁生 | 戊生 |
| 08/07 | 韓式拌飯 | 烤肉飯 | - | 烤肉飯 | 素便當 |
| 08/08 | - | 烤肉飯 | 韓式拌飯 | - | 素便當 |
| 08/09 | 素便當 | 韓式拌飯 | 素便當 | 烤肉飯 | 素便當 |
| 08/10 | 烤肉飯 | 烤肉飯 | - | 韓式拌飯 | 素便當 |
| 08/11 | 韓式拌飯 | 烤肉飯 | 韓式拌飯 | 韓式拌飯 | 素便當 |

以下根據所需的原料進行分類，由上往下進行篩選，例如「韓式拌飯」符合第一行的Regex篩選，故屬於調味飯類；「烤肉飯」不符合第一行的Regex篩選，而符合第三行的Regex篩選，故屬於便當類。

|  |  |
| --- | --- |
| 調味飯類 | Regex: "(焗)|(拌飯)" |
| 麵類 | Regex: "(烏龍)|(麵)|(湯)" |
| 便當類 | Regex: "(副菜)|(飯)" |
| 小吃類 | Regex: "((餃)|(蔥抓餅)|(鍋貼)|(板條))" |
| 鍋類 | Regex: "(鍋)|(粥)" |
| 其他 | 不滿足以上篩選者 |

#### 1. 點餐序列

將甲生的點餐資料提取出來，並且加以分類，再填上是否有點餐，即為點餐序列。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 甲生 | 08/07 | 08/08 | 08/09 | 08/10 | 08/11 |
| 餐點類別 | 調味飯類 | - | 其他 | 便當類 | 調味飯類 |
| 是否點餐 | True | False | True | True | True |

#### 2. 點餐圖

根據原點餐序列，刪去沒有點餐的那一天，即如下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 甲生 | 08/07 | 08/09 | 08/10 | 08/11 |
| 餐點類別 | 調味飯類 | 其他 | 便當類 | 調味飯類 |
| 是否點餐 | True | True | True | True |

將每一天的轉移視為圖上的一個邊，每一個分類視為圖上的一個節點，則可得到一個有向圖，我們稱呼為點餐圖。

若將每一個人的點餐序列寫入同一張點餐圖，則可得到類似下面的有向圖。

#### 3. 點餐矩陣

### (一)、宏觀模型

|  |  |
| --- | --- |
| 電腦進行的操作 | 現實世界的意義 |
| 從點餐圖上的任意一個點出發 | 第一天先隨便點一道菜 |
| 經由任意一條邊 | 經過了一天 |
| 到達另外圖上的另外一個點 | 第二天點了另外一道餐 |
| 經過一個自環 | 經過了一天，決定要吃同一道菜 |
| 到了同一個點 | 第三天點了同一道菜 |

### (二)、微觀模型

方便細部追蹤一個群體

### (三)、模型應用

便當的原料

廠商只需要準備多少資料

# 伍、目前研究結果

## 一、系統

## 二、分析實例

#### (一)、宏觀模型預測實例

#### (二)、微觀模型預測實例

#### (三)、預估明日應準備多少原料

# 陸、參考資料及其他