國 立 中 央 大 學 資 訊 管 理 學 系

系統 分析 與 設 計

- 第三次作業繳交第二,三,四章全部,
 - ◆ 第二章須包含全部資料庫表格,
 - ◆ 第三章須包含所有 java classes
 - ◆ 第四章包含兩個使用案例就好了

其它章節自己參考就好,請勿當作業繳交,以免老師跟助教需要處理額外紙張

(12/17 繳交本作業):

系統軟體設計規格書

第 17 組

資管三 A 110403016 劉育嘉

資管三 A 110403011 王述函

資管三B 109403047 秦 瓏

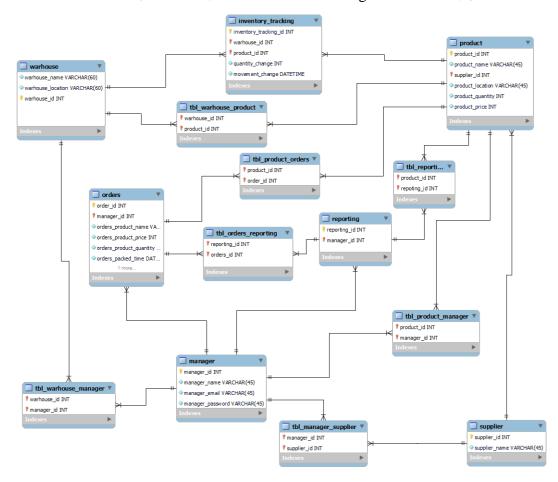
資管三B 110403042 邱奕輝

資管三B 110403567 陳美靜

指導教授: 許智誠 教授

第2章 資料庫設計

設計階段之資料庫,根據分析文件之實體關係圖(Entity-Relation Diagram),進行確認並依據其規劃資料庫之資料表。在您提供的 SQL 腳本中,共有 14 個實體(即資料表),以及 14 個關係(即外鍵約束),下圖為設計階段之 ER 圖,亦可使用資料庫綱要圖(Schema Diagram)進行取代:



er model(圖 1)

根據上圖進行資料表之設計,以下將逐一說明資料庫每張資料表之欄位

1. 管理員模組

表 1: 管理員資料表 (manager) 之資料結構

Key	名稱	類型	預設 值	空 值	自動 增加	編碼
P.K	id	Int(11)	無	否	V	
	name	Varchar(45)	無	否		utf8mb4_0900_ai_ci
	email	Varchar(45)	無	否		utf8mb4_0900_ai_ci
	password	Varchar(45)	無	否		utf8mb4_0900_ai_ci

✓ id: 為自動增加作為管理者編號,不可更動由資料庫系統自動產 生。

✓ name: 用於紀錄管理者名稱。

✓ email: 用於紀錄該名管理者的 email。

✓ password: 用於記錄該名管理者的密碼。

2. 倉庫模組

表 2: 倉庫資料表 (warehouse) 之資料結構

Key	名稱	類型	預設 值	空 值	自動 增加	編碼
P.K	id	Int(11)	無	否	V	
	name	Varchar(45)	無	否		utf8mb4_0900_ai_ci
	location	Varchar(45)	無	否		utf8mb4_0900_ai_ci

✓ id: 為自動增加作為倉庫編號,不可更動由資料庫系統自動產生。

✓ name: 用於紀錄倉庫名稱。

✓ location: 用於紀錄該倉庫的位置。

3. 供應商模組

表 3: 供應商資料表 (supplier) 之資料結構

Key	名稱	類型	預設 值	空值	自動 增加	編碼
P.K	id	Int(11)	無	否	V	
	name	Varchar(45)	無	否		utf8mb4_0900_ai_ci

✓ id: 為自動增加作為供應商編號,不可更動由資料庫系統自動產 生。

✓ name: 用於紀錄供應商名稱。

4. 產品模組

表 4: 管理員資料表 (manager) 之資料結構

Key	名稱	類型	預設 值	空值	自動 增加	編碼
P.K	id	Int(11)	無	否	V	
	name	Varchar(45)	無	否		utf8mb4_0900_ai_ci
F.K	Supplier_id	Int(11)	無	否		
	quantity	Int(11)	無	否		utf8mb4_0900_ai_ci
	price	Int(11)	無	否		utf8mb4_0900_ai_ci
	location	Varchar(45)	無	否		utf8mb4_0900_ai_ci

✓ id: 為自動增加作為管理者編號,不可更動由資料庫系統自動產生。

✓ name: 用於紀錄產品名稱。

✓ supplier_id: 作為外鍵得到供應商資料。

✓ location: 用於記錄該產品存放的位置。

✓ quantity:紀錄該產品的數量。

✓ price:紀錄該產品的價格。

5. 庫存追蹤模組

表 5: 庫存追蹤資料表 (inventory_tracking) 之資料結構

Key	名稱	類型	預 設 值	空值	自動增加	編碼
P.K	id	Int(11)	無	否	V	
F.K	Product_id	Int(11)	無	否		
F.K	warhouse_id	Int(11)	無	否		
	Quantity_change	Int(11)	無	否		utf8mb4_0900_ai_ci
	Movement_time	Date_time	無	否		utf8mb4_0900_ai_ci

✓ id: 為自動增加作為管理者編號,不可更動由資料庫系統自動產生。

✓ product_id: 作為外來鍵得到產品資料。

✓ warhouse_id: 作為外來鍵得到倉庫資料。

✓ movement_time: 紀錄該產品數量變化的時間。

✓ quantity_change:紀錄該產品數量上的變化。

6. 訂單模組

表 6: 訂單資料表 (orders) 之資料結構

Key	名稱	類型	預設值	空值	自動增加	編碼
P.K	id	Int(11)	無	否	V	
F.K	manager_id	Int(11)	無	否		
	Orders_product_name	Varchar(45)	無	否		
	orders_product_quantity	Int(11)	無	否		utf8mb4_0900_ai_ci
	orders_statement	Varchar(45)	無	否		utf8mb4_0900_ai_ci
	orders_packed_time	Date_time	無	否		utf8mb4_0900_ai_ci

- ✓ id: 為自動增加作為訂單編號,不可更動由資料庫系統自動產生。
- ✓ manager_id: 作為外來鍵得到管理者資料。
- ✓ Orders_product_name: 紀錄訂單中要求的產品名稱。
- ✓ Orders_product_quantity: 紀錄訂單中要求的產品數量。
- ✓ Orders_statement:紀錄訂單得狀態。
- ✓ Orders_packed_time:紀錄訂單打包時間。

7. 報告模組

表 7: 供應商資料表(supplier)之資料結構

Key	名稱	類型	預設 值	空 值	自動 增加	編碼
P.K	id	Int(11)	無	引	V	
	Manager_id	Int(11)	無	否		

- ✓ id: 為自動增加作為報告編號,不可更動由資料庫系統自動產生。
- ✓ manager_id: 作為外來鍵得到管理者資料。

第3章 類別圖

下圖(圖、圖、圖)係依據倉儲管理系統的分析模型和建立的互動圖,以及實體關係圖(Entity-Relation Diagram)所繪製之設計階段之類別圖(Class Diagram),用於描述系統的類別集合,包含其中之屬性,與類別之間的關係。

本階段之類別圖屬於細部(detail)之設計圖,與上一份文件分析階段之類別圖需要有詳細之變數型態、所擁有之方法,依據這些設計原則,本類別圖之說明如下所列:類別圖除包含與資料庫相對應之物件外,亦包含相關之控制物件(controller)、DBMgr 與各功能相對應資料庫操作類別(例如:

MemberHelper) 和相對應之類別工具(JsonReader)。

	DBMgr	ManagerHelper
uest_string: String est: HttpServletReguset	+ JDBC_DRIVER: Sring + DB_URL: String	- mh: ManagerHelper - conn: Connection - pres: PreparedStatement
nReader(request: HttpServletRequset)	+ USER: String + USER: String + PASS: String	- ManagerHelper() + getHelper(): ManagerHelper + getAll(): JSONObject
Parameter(key:String): String Array(): JSONArray pipect(): JSONObject ponse(resp_string, response:HttpServietResponse) ponse(resp.JSONObject, response:HttpServietResponse)	+ DBMgr() + getConnection(): Connection + close(stm:Statement, conn:Connection) + close(rs:ResultSet, stm:Statement, conn: C	+ checkiDuplicate(m: Manager): boolea + create(m: Manager): JSONObject + deleteByID(id: int): JSONObject + update(m: Manager): JSONObject
oonse(status_code:int, resp:JSONObject, response:HttpServietRes		
SupplierHelper	- wh: WarehouseHelper	ehouseHelper
- sh:SupplierHelper - conn:Connection - pres:PreparedStatement	- conn: Connection - pres: PreparedStatement - WarehouseHelper() + oetHelper(): WarehouseHelper	
	- Contract of the contract of	
- SupplierHelper() + getHelper():SupplierHelper + sp_deletesupplier(id:int):JSONObject	+ getAll():JSONObject	house_location: String, p_manager_id: int): JSONObject se_name: String, p_warhouse_location: String): JSONObject
+ getHelper():SupplierHelper	+ getAll():JSONObject	
+ getHelper().SupplierHelper + sp_deletesupplier(id:int).JSONObject + getAll().JSONObject	+ getAll()_ISONObject + sp_updatewarhouse(p_warhouse_id: int, p_warhouse - ph:Proc - conn:Ci	se_name: String, p_warhouse_location: String): JSONObjer ProductHelper uctHelper
+ getHelper():SupplierHelper + sp_deletesupplier(id.int):JSONObject + getAll():JSONObject + sp_addsupplier(s:Supplier):JSONObject + sp_updatesupplier(s:Supplier):JSONObject	+ getAll()_ISONObject + sp_updatewarhouse(p_warhouse_id: int, p_warhouse - ph:Proc - conn:C pres:Pr	ProductHelper uotHelper sparedStatement
+ getHelper().SupplierHelper + sp_deletesupplier(id-int).JSONObject + getAll().JSONObject + sp_addsupplier(s:Supplier).JSONObject + sp_updatesupplier(s:Supplier).JSONObject + sp_selectsupplier().JSONArray	+ getAl()_JSONObject + sp_updatewarhouse(p_warhouse_lid-int, p_warhouse - ph:Proc - con::C - pres:Pr	ProductHelper uotHelper sparedStatement

圖 2: 類別圖(1/3)

Manager

- id: int

- name: String - email: String - password: String - mh: ManagerHelper

+ Manager(name: String, email: String, password: String)

+ Manager(id: int, name: String, email: String, password: String)

+ getID(): int

+ getName(): String + getEmail(): String

+ getPassword(): String + update(): JSONObject + getData(): JSONObject

+ Manager(email: String, password: String)

+ setID(id: int)

圖 3: 類別圖 (2/3)

OrderController

- serialVersionUID: long ph: ProductHelper
- oh: OrderHelper
- + doPost(request: HttpServletRequest, response: HttpServletResponse)
- + doGet(request: HttpServletRequest, response: HttpServletResponse) + doDelete(request: HttpServletRequest, response: HttpServletResponse
- + doPut(request: HttpServletRequest, response: HttpServletResponse)

ProductController

- SerialVersionUID:long
- ph:ProductHelper
- + ProductController()
- + doPost(request:HttpServletRequest, response:HttpServletResponse)
- + doGet(request: HttpServletRequest, response: HttpServletResponse)
- + doDelete(request:HttpServletRequest, response:HttpServletResponse)
- + doPut(request:HttpServletRequest, response:HttpServletResponse)

ManagerController

- serialVersionUID: long mh: ManagerHelper
- + doPost(request: HttpServletRequest, response: HttpServletResponse)
 + doGet(request: HttpServletRequest, response: HttpServletResponse)
 + doDelete(request: HttpServletRequest, response: HttpServletResponse)
- + doPut(request; HttpServletRequest, response; HttpServletResponse)

SupplierController

- SerialVersionUID:long
- sh:SuppliertHelper
- + SupplierController()
- + doPost(request:HttpServletRequest, response:HttpServletResponse)
- + doPut(request:HttpServletRequest, response:HttpServletResponse)
- + doDelete(request:HttpServletRequest, response:HttpServletResponse)
- + doGet(request:HttpServletRequest, response:HttpServletResponse)

WarehouseController

- serialVersionUID: long
- wh: WarehouseHelper
- + doPost(request: HttpServletRequest, response: HttpServletResponse)
 + doGet(request: HttpServletRequest, response: HttpServletResponse)
 + doDelete(request: HttpServletRequest, response: HttpServletResponse)
- + doPut(request: HttpServletRequest, response: HttpServletResponse)

圖 4: 類別圖 (3/3)

第4章 系統循序圖

本章節主要依照第一份文件需求所產生之使用案例圖(use case)與第二份 文件分析之邏輯階段活動圖與強韌圖為基礎,進行設計階段之循序圖設計,將 每個使用案例進行闡述。於此階段,需要有明確之類別(class)名稱與呼叫之 方法(method)與傳入之變數名稱與型態等細部設計之內容。

4.1 使用案例圖

依據第一份文件針對專案之需求進行確定, 本倉儲管理系統預計共有 1 位動作者與 24 個使用案例,並依照不同之模組區分成不同子系統共計七個子系統,其中包含以下: ①倉庫子系統、②產品管理子系統、③庫存追蹤子系統、〇4 訂單處理子系統、⑤應商管理子系統、⑥報告子系統、⑦管理員管理子系統,下圖(圖 1)為本系統之使用案例圖:

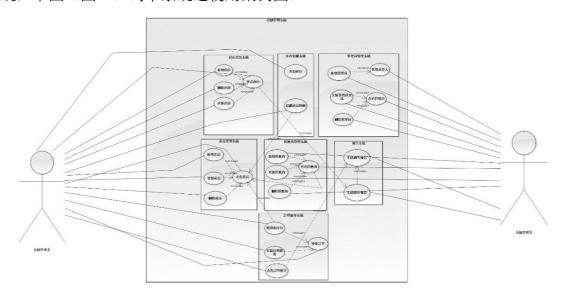


圖 5: 使用案例圖

4.2 Use Case 實做之循序圖

4.2.1 流程編號 1.0: 會員模組

在 Use Case 1.0 中所使用到的功能(包含:新增倉庫、查看倉庫、更新倉庫、刪除倉庫等),以下詳細說明各功能。

- 1. Use Case 1.1: 新增倉庫
 - 一般使用者可以新增新的倉庫,包括倉庫名稱、地址、容量等基本資訊。
- 2. Use Case 1.2: 查看倉庫
 - 一般使用者可以查看現有倉庫的詳細資訊,包括倉庫名稱、地址、容量等。
- 3. Use Case 1.3: 更新倉庫
 - 一般使用者可以對現有倉庫的資訊進行更新,例如修改倉庫名稱、地址或容量。
- 4. Use Case 1.4: 刪除倉庫 管理員可以刪除倉庫資訊。

4.2.1.1 Sequence Diagram—Use Case 1.1 新增管理員

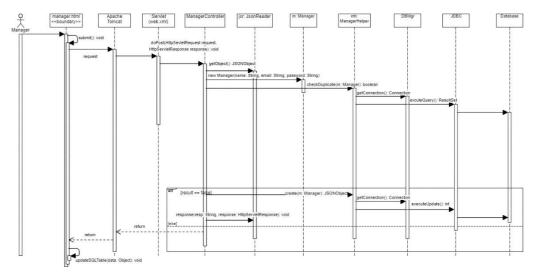


圖 6: 商業流程編號 1.1 新增管理員循序圖

- 1. 管理員完成「7.1 管理員登入」後,進入管理員介面(manager.html)。
- 2. 當管理員完成表單之填寫並通過前端之資料驗證後,透過 JavaScript 之 submit()送出 POST 請求。
- 3. 後端以 ManagerController 之 doPost()進行處理,以 JsonReader 取回 request 之參數,使用 ManagerHelper 物件之 checkDuplicate()方法檢查該管理員是 否重複,若無則以 create()新增該名管理員資料至資料庫當中。
- 4. 若重複則直接回傳重複之錯誤訊息。
- 5. 回傳新增之結果後,若新增成功則透過 JavaScript 之 updateSQLTable()更新 SQL 表格內之內容。

表 8: 商業流程編號 1.1 新增管理員 Business Exception List

http status code/message	發生之 method	說明
400/新增帳號失敗,帳號重複!	submit()	信箱重複,新增失敗

4.2.1.2 Sequence Diagram—Use Case 2.3 刪除產品

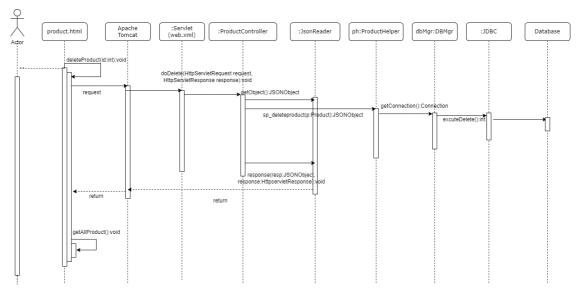


圖 7: 商業流程編號 2.3 刪除產品循序圖

- 6. 管理員完成「7.1 管理員登入」後,點擊進入產品管理介面 (product.html)。
- 7. 針對所欲刪除之產品點擊刪除之按鈕, JavaScript 之 deleteProduct()發送 DELETE 之請求。
- 8. 後端以 ProductController 之 doDelete()進行處理,以 JsonReader 取得該參數後,透過 ProductHelper 物件的 sp_deleteproduct(id:int)方法將資料庫之該產品進行刪除。
- 9. 若刪除成功則透過 JavaScript 之 getAllProduct ()再次執行商業流程編號「2.4 檢視產品」之循序圖過程。