資料庫管理（113-1）
期末專案計劃書

B11705004 周子馨、B11705027 陳承妤、B11705033 江睿宸

December 10, 2024

GitHub 專案連結 展示影片連結

1 系統分析

隨著航空業的發展，飛機製造商如波音面臨越來越複雜的生產需求。波音的製造流程涉及全球數百家供應商及成千上萬個零件，要求每個零件必須精確符合飛機設計規範，並且在生產流程中能夠準時到達生產線。因此，波音需要一個強大的存貨管理系統，不僅能精確追蹤零件庫存數量，還能有效地支援供應商管理與整體生產流程，確保供應鏈到最終生產的每個環節都能高效運作，這正是「Boeing’s Components Record System（波音零件記錄系統）」所要滿足的核心。

從製造方面分析，波音的飛機系列涵蓋了多種類型，包括波音 707、717、720、727、737、747、757、767、777 和 787 系列。每個系列還包括不同的機型，如 787 系列的 787-8、787-9、787-10，各種機型的組成零件各有不同。以 787 系列為例，787-9 與 787-10 的機身皆使用碳纖維複合材料，但機身長度不同，而兩者經濟艙座椅的設計雖相似，但整體座位配置因載客量不同而有所區別。因此，在設計與生產的每個階段都需要精準掌握各機型的零件要求，以避免錯誤、減少成本並確保生產進度。

從供應商分析，波音無法自行生產所有零件，而是依賴全球範圍內的供應商網絡。例如，在 787 Dreamliner 的生產中，波音與數百家供應商合作，這些供應商為其提供機身、引擎、內部座椅等核心零件。由於供應商來自世界各地，波音面臨如何整合零件並將它們順利輸送至最終組裝線的挑戰。此過程中，供應商需嚴格遵循波音的設計規範，以符合精密的尺寸和質量標準。為達到標準化的效果，波音對於每個機型的零件大多數採取固定供應商的策略，以降低供應鏈變數並保持生產穩定性。另外，波音會綜合評估供應商的歷年交貨品質、成本、可靠性、交貨速度等因素，並考慮到是否與該供應商有持股關係。透過這些標準，波音的高層可以有系統地評分並選擇最合適的供應商，以降低生產成本並提高供應鏈效率。

從內部分工分析，決策過程和分工同樣嚴格。員工層級涵蓋了從 CEO、高層主管到各工廠的組裝員工。CEO 和高層主管負責決策某機型的設計方案、零件組成與供應商選擇等關鍵任務；而組裝員工則負責實際生產流程中的執行與支援工作，包括下訂單、確認訂單進度、檢查零件的到貨狀況等。在這種多層次的管理體系中，每位員工的職責明

確，並根據不同的職務角色，具備相應的系統操作權限。

「Boeing’s Components Record System」作為波音零件管理的核心系統，記錄製造相關的零件流動、人事資訊以及供應商狀況，權限劃分的部分有 User 跟 Admin。User 組裝員工可利用系統來完成日常的例行性工作，例如新增訂單、記錄訂單狀態（如「未出貨」、「運送中」、「已到貨」、「退貨」和「刪除」）、查詢零件和供應商資訊，並查看某工廠負責生產的零件類型等。這些功能可幫助組裝員工即時掌握每個零件的生產和配送狀況，從而減少生產過程中的意外延遲。Admin 角色由公司高層成員擔任，擁有更高的系統操作權限。他們可以修改零件資訊、更新供應商資訊、對供應商的交貨品質等進行打分，以供未來評估。

1.1 系統功能

1.1.1 關於 Boeing’s Components Record System 的相關設定

在零件記錄的部分，每個零件都有自己的 ID，而不同零件（Inventory）有可能屬於同一個物件（Part），像是 787-9 機型的引擎跟廠商乙進貨、787-10 機型的引擎跟廠商甲進貨，雖然 Inventory 的 ID 不同，但是同屬一個物件（Part），因此適用的檢查標準相同。每個零件會由多個子零件組成，子零件可能是外部跟供應商進貨，抑或是由內部提供。

在內部運作的部分，波音有好幾個製造工廠，而每個工廠都有一個負責人，一個人可能負責好幾個工廠。員工包含 CEO、公司高層、組裝員工，每個組裝員工都有一個監督主管，一位主管會監督好幾個組裝員工，而 CEO 不受任何人監督。

在訂單記錄的部分，只要是公司內部的員工，都有權限新增跟修改訂單，新增訂單時，會記錄該訂單是由哪名員工下單，之後的收貨也是由該員工負責。訂單狀態包含未出貨、運送中、到貨、刪除、退貨，而另外會有一格不限字數的備註，可記錄到貨後的檢測成果、退貨原因等。一旦新增訂單，該訂單的狀態會初始設定為未出貨，實際到貨時間會是空值。

1.1.2 給 User 的功能

在本系統中，User 可以執行以下功能：

1. 新增訂單：組裝員工可根據當時的製造需求下訂單，訂單會記錄訂單 ID、下單時間、應到貨時間、實際到貨時間、數量、貨品狀態（未出貨、運送中、已到貨、退貨和刪除）、檢查回饋、該訂單負責員工 ID（使用者本人）、下訂的零件 ID。一開始下單的實際到貨時間為空值，貨品狀態為未出貨，檢查回饋為空值。

2. 修改訂單狀態：組裝員工如果要刪除該訂單，可將貨品狀態改成刪除，因此還是保留該記錄。等產品確定出貨，可將貨品狀態改成運送中。到貨後，可將貨品狀態改成已到貨，會記錄實際到貨時間，而且會進行貨品檢查，可把細節記錄在檢查回饋裡。如果檢查過後發現有問題，退貨後可將貨品狀態改成退貨。

3. 查詢零件資訊：組裝員工可使用零件 ID 查詢相關資訊，包括零件名稱、零件狀態（現在有無使用）、該零件適用檢查標準。

4. 查詢工廠資訊：組裝員工可查詢公司內部工廠相關資訊，包括工廠位置、工廠聯絡電話、工廠負責人等。

5. 供應商資訊：組裝員工可查詢有跟公司合作的供應商資訊，方便在下訂貨品的時候做決定。

1.1.3 給 Admin 的功能

在本系統中，Admin 可以執行以下功能：

1. 新增訂單：公司高層可根據當時的製造需求下訂單，訂單會記錄訂單 ID、下單時間、應到貨時間、實際到貨時間、數量、貨品狀態（未出貨、運送中、已到貨、退貨和刪除）、檢查回饋、該訂單負責員工 ID（使用者本人）、下訂的零件 ID。一開始下單的實際到貨時間為空值，貨品狀態為未出貨，檢查回饋為空值。

2. 修改訂單狀態：公司高層如果要刪除該訂單，可將貨品狀態改成刪除，因此還是保留該記錄。等產品確定出貨，可將貨品狀態改成運送中。到貨後，可將貨品狀態改成已到貨，會記錄實際到貨時間，而且會進行貨品檢查，可把細節記錄在檢查回饋裡。如果檢查過後發現有問題，退貨後可將貨品狀態改成退貨。

3. 新增、查詢、修改零件資訊：公司高層除了與組裝員工一樣能夠查詢，還有權限更動零件資訊，有新的零件使用會新增零件資訊。如果有零件在現有設計中都不會用到，則在該零件狀態加上註記。

4. 新增、查詢、修改供應商資訊：公司高層可根據需要新增供應商資料，記錄供應商的 ID、名稱、地區、聯絡資訊等。若供應商資訊有更新，公司高層也可進行修改。

5. 新增、查詢、修改評分標準：公司高層可以為供應商設立評分標準，並且根據過往合作經驗調整評分標準的項目，例如交貨準時性、零件品質、售後服務等，但記錄中只會有最終分數，且會記錄是在哪年評分該供應商的零件，每次評分都會有一位負責人，其員工 ID 會被記錄。

2 系統設計

2.1 ER Diagram

圖 1 是「Boeing's Components Record System」的 ER Diagram，在這個 ERD 中共有六個實體（entity），分別是 `INVENTORY`、`FACTORY`、`SUPPLIER`、`EMPLOYEE`、`ORDER`、`PART`，以及九個關係（relationship）。其中員工在入職時會有專屬員工 ID 及密碼作為登入帳號，可以檢視及新增部分內容。如果該員工同時做為其他員工的上級，他可以作為 Admin 擁有更多權限如刪除、修改。

`INVENTORY` 記錄每個飛機會需要使用到的零件，包含小至螺絲、螺帽，大至機翼、貨艙等。由不同廠家生產的零件都會有不一樣的編號，且同一零件只會由唯一一家廠商生產。每個零件都需要紀錄其狀態及他是哪一種物件（part），每一種物件會有一個統一的規格（例如：廠商甲和廠商乙都有生產飛機座椅，則兩者會記錄為兩種 `INVENTORY`，但屬於一樣的 `PART`）。每種零件都可能由內部工廠製成或外部公司簽約合作製成。而每個

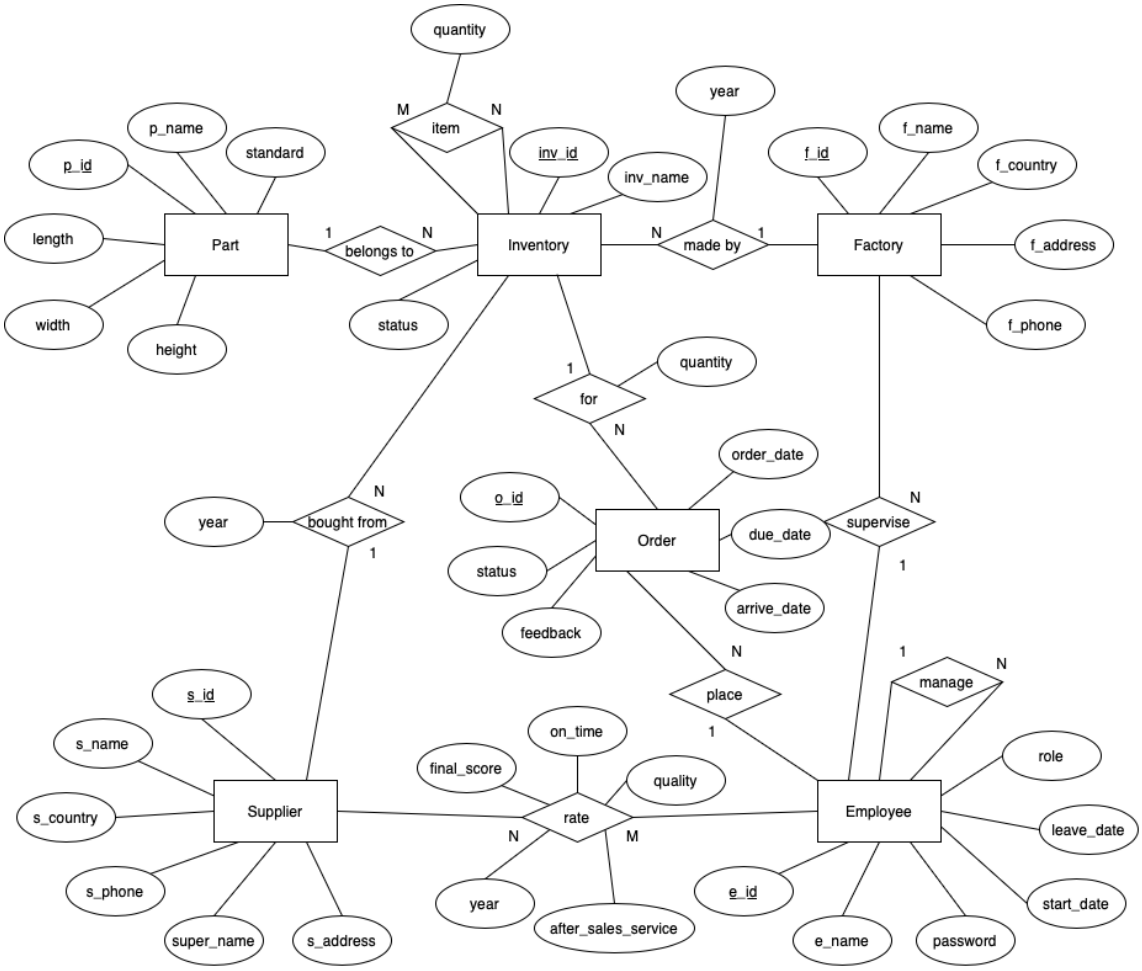

Figure 1: 「Boeing's Component Record System」的 ER Diagram

零件都有可能是由其他子零件組合而成，ITEM 會記錄相關的關係。

每間合作公司和工廠我們都會記錄相關的資料如地址及電話等，其中外部公司和內部工廠都會有一位對接負責人，而內部的負責人會是 EMPLOYEE 之一。每年我們都會給外部合作的公司評分，根據評分結果可能更換合作公司，並且內部可能因為成本問題開或關閉工廠，因此我們會記錄每個個部件由誰製作及開始的年份。

ORDER 會記錄訂單紀錄，包括訂單編號、下訂的員工、款式、數量、日期、截止日及實際到貨日等資料，其中狀態則會記錄這批貨品的狀態，包括未出貨、運送中、已到貨、退貨和刪除，到貨後的檢查細節則會記錄在回饋中。由於每款零件都由唯一一家廠商製造，廠家在下訂單時便會根據需求下訂該家廠商的該款零件，不再另外記錄下訂對象。

2.2 Relational Database Schema Diagram

我們可以將圖 1 的 ER Diagram 轉換成圖 2 的 Database Schema，一共由十個關聯（relation）組成，分別是 SUPPLIER、RATE、PART、FACTORY、EMPLOYEE、ORDER、INVENTORY、INVENT_BY_FACTORY、

INVENT_BY_SUPPLIER、ITEM。

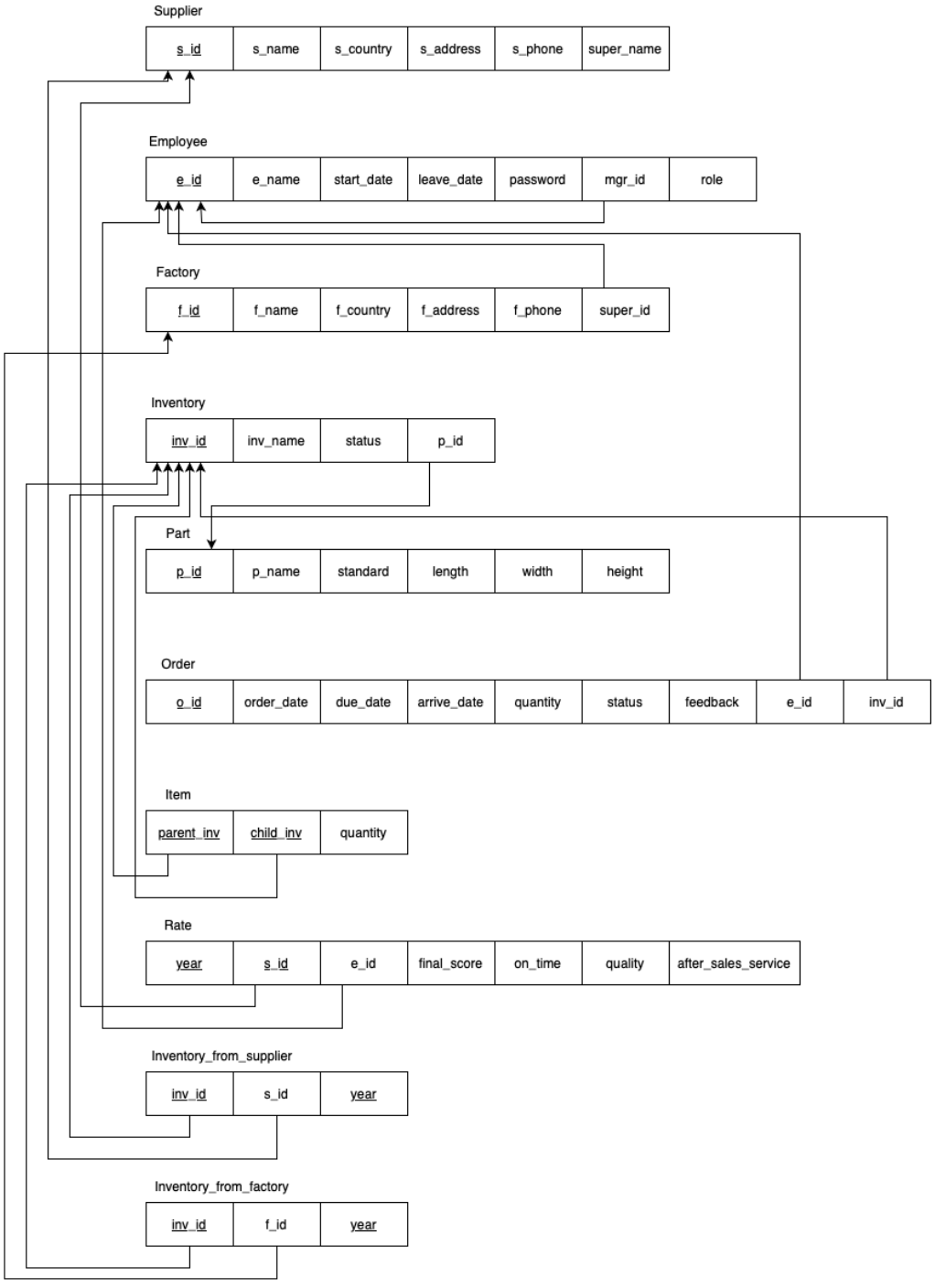

Figure 2:「Boeing's Component Record System」的 Relational Database Schema Diagram

INVENTORY 是指一個獨立零件，包含 p_id 作為外部鍵指向 PART，代表它屬於哪一種類別。而 ITEM 由多對多 INVENTORY 獨立而來，兩個外部鍵都指向 INVENTORY。

INVENT_BY_SUPPLIER、INVENT_BY_FACTORY 分別記錄 INVENTORY 是由誰製造，分別

指向 SUPPLIER 的主鍵 s_id 與 FACTORY 的主鍵 f_id。

RATE 這個關聯是由多對多之 RATE 關係型態產生而來，該關聯的主鍵由兩個外部鍵組合而成，分別作為參考到連結該關係的兩個實體 EMPLOYEE 的主鍵 e_id 以及 SUPPLIER 的主鍵 s_id。

2.3 Data Dictionary

「Boeing's Component Record System」的資料表共有圖 2 所示的十個，各個資料表的欄位相關資訊依序呈現在表 1 到表 10。

Column Name	Meaning	Data Type	Key	Constraint	Domain
s_id	供應商編號	bigint	PK	Not Null	
s_name	供應商名稱	varchar (40)		Not Null	
s_country	供應商國家	varchar (20)		Not Null	
s_address	供應商地址	varchar (100)		Not Null	
s_phone	供應商電話	varchar (20)		Not Null	
super_name	負責人姓名	varchar (40)		Not Null	

Table 1: 資料表 SUPPLIER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
e_id	員工編號	varchar (20)	PK	Not Null	
e_name	員工姓名	varchar (40)		Not Null	
start_date	開始工作日期	date		Not Null	
leave_date	結束工作日期	date			
password	登入密碼	varchar (50)		Not Null	
mgr_id	主管編號	varchar (20)	FK: EMPLOYEE(e_id)	Not Null	
role	員工角色	varchar (5)		Not Null	Admin, User
Referential triggers		On Delete	On Update		
mgr_id: EMPLOYEE(e_id)		Cascade	Cascade		

Table 2: 資料表 EMPLOYEE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
f_id	工廠編號	bigint	PK	Not Null	
f_name	工廠名稱	varchar (40)		Not Null	
f_country	工廠國家	varchar (20)		Not Null	
f_address	工廠地址	varchar (100)		Not Null	
f_phone	工廠電話	varchar (20)		Not Null	
super_id	負責人編號	varchar (20)	FK: EMPLOYEE(e_id)	Not Null	
Referential triggers		On Delete	On Update		
super_id: EMPLOYEE(e_id)		Cascade	Cascade		

Table 3: 資料表 FACTORY 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
inv_id	零件編號	bigint	PK	Not Null	
inv_name	零件名稱	varchar (40)		Not Null	
status	零件狀態	varchar (10)		Not Null	{Using, Not Using}
p_id	物件編號	bigint	FK: PART(p_id)	Not Null	
Referential triggers		On Delete	On Update		
p_id: PART(p_id)		Cascade	Cascade		

Table 4: 資料表 INVENTORY 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
p_id	物件編號	bigint	PK	Not Null	
p_name	物件名稱	varchar (30)		Not Null	
standard	物件標準	varchar (1000)		Not Null	
length	物件長度	float		Not Null	(毫米)
width	物件寬度	float		Not Null	(毫米)
height	物件高度	float		Not Null	(毫米)

Table 5: 資料表 PART 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
o_id	訂單編號	bigint	PK	Not Null	
order_date	訂購日期	date		Not Null	
due_date	截止日期	date		Not Null	
arrive_date	抵達日期	date			
quantity	訂購數量	int		Not Null	
status	訂單狀態	varchar (20)		Not Null	{Not Yet Shipped, In Transit, Delivered, Returned, Deleted}
feedback	零件狀態回饋	varchar (1000)			
e_id	下訂員工編號	varchar(20)	FK: EMPLOYEE(e_id)	Not Null	
inv_id	下訂零件編號	bigint	FK: INVENTORY(inv_id)	Not Null	
Referential triggers		On Delete	On Update		
e_id: EMPLOYEE(e_id)		Cascade	Cascade		
inv_id: INVENTORY(inv_id)		Cascade	Cascade		

Table 6: 資料表 ORDER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
parent_inv	母體零件編號	bigint	PK, FK: INVENTORY(inv_id)	Not Null	
child_inv	子零件編號	bigint	PK, FK: INVENTORY(inv_id)	Not Null	
quantity	包含子零件數量	int	PK, FK: INVENTORY(inv_id)	Not Null	
Referential triggers		On Delete	On Update		
parent_inv: INVENTORY(inv_id)		Cascade	Cascade		
child_inv: INVENTORY(inv_id)		Cascade	Cascade		

Table 7: 資料表 ITEM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
s_id	被評分供應商編號	bigint	PK, FK: SUPPLIER(s_id)	Not Null	
year	評分年度	int	PK	Not Null	
on time	準時性	int		Not Null	
quality	交貨品質	int		Not Null	
after-sales service	售後服務	int		Not Null	
final score	最終加權分數	float		Not Null	
e_id	評分員工編號	varchar(20)	FK: EMPLOYEE(e_id)	Not Null	
Referential triggers		On Delete	On Update		
s_id: SUPPLIER(s_id)		Cascade	Cascade		
e_id: EMPLOYEE(e_id)		Cascade	Cascade		

Table 8: 資料表 RATE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
inv_id	零件編號	bigint	PK, FK: INVENTORY(inv_id)	Not Null	
s_id	供應商編號	bigint	FK: SUPPLIER(s_id)	Not Null	
year	提供年度	int	PK	Not Null	
Referential triggers		On Delete	On Update		
inv_id: INVENTORY(inv_id)		Cascade	Cascade		
s_id: SUPPLIER(s_id)		Cascade	Cascade		

Table 9: 資料表 INVENTORY_FROM_SUPPLIER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
inv_id	零件編號	bigint	PK, FK: INVENTORY(inv_id)	Not Null	
f_id	工廠編號	bigint	FK: FACTORY(f_id)	Not Null	
year	提供年度	int	PK	Not Null	
Referential triggers		On Delete	On Update		
inv_id: INVENTORY(inv_id)		Cascade	Cascade		
f_id: FACTORY(f_id)		Cascade	Cascade		

Table 10: 資料表 INVENTORY_FROM_FACTORY 的欄位資訊

2.4 正規化分析

關聯式資料庫設計完成後，能經由正規化（normalization）改進關聯式綱要（relational schema），以下將說明這次的「Boeing's Components Record System」符合第一正規式（1NF）到第四正規式（4NF）。

首先，1NF 要求每個關聯的屬性都是 single-valued 且 simple。在我們的題目設定中，各個屬性原先就只會對應到一個值，因此 schema 滿足 1NF。

其次，2NF 成立的條件為關聯中每個非鍵屬性（nonprime arrtibute）皆非部分相依任意候選鍵（partially depend on any candidate key）。2.2 小節中展示的綱要圖符合此項條件。

接著，若滿足 3NF 則表示關聯中的任意非鍵屬性都沒有遞移相依（transitively dependency），而我們的 `SUPPLIER` 中的 s_country、s_address、s_phone 以及 `FACTORY` 的 f_country、f_address、f_phone 都有遞移相依，因為可以從地址或電話號碼得知國家。

至於 BNCF 更嚴格地規定功能相依性的箭頭左方必須為超級鍵（super key）。我們的 schema 與這規定相符。

最後，4NF 要求關聯內不能存在多值相依（multi-valued dependency），我們的綱要圖也沒有這項問題。綜上所述，Boeing's Components Record System 的 relational schema 符合正規化條件。

3 系統實作

3.1 資料庫建置方式及資料來源說明

`SUPPLIER` 的資料是由 Faker 套件搭上各國家的權重（把歐美國家權重調重）以及國際電話號碼生成，共 3000 筆資料。`EMPLOYEE` 的資料一樣是搭配 Faker 套件在加上日期條件生成，因為此 Table 涵蓋了從 1997 年（波音公司創立）至今的組裝員工資料，在生成測資時，有讓較多年前的員工是離職狀態，password 為了符合真實情況有匯入字典來搭配亂數數字，共 100002 筆資料。`FACTORY` 的資料是用 Chatgpt 生成的工廠 list 搭配字母，一樣有按照波音設廠比例去調控工廠國家分布，共有 150 筆資料。

`PART` 是先分大零件 50 種、中零件 100 種、小零件 200 種，接著用亂數決定要加什麼形容詞，像是 Fast Clamp，共有 1984 筆資料。`INVENTORY` 則是在 `PART` 生成時順便再加有無延伸 inventory，像是 Fast Clamp A、Fast Clamp B、Fast Clamp C，共有 5824 筆資料。`ITEM` 代表子母零件關係，所以在分配時，會是飛機本身由大零件組成，大零件由中零件組成、中零件由小零件組成，數量則是大的零件會包含較多零件，小的零件會包含較少零件，共有 1121034 筆資料。`INVENTORY_FROM_SUPPLIER`、`INVENTORY_FROM_FACTORY` 是一起生成，記錄 1997 年至今的所有零件及其對應供應商，因為波音主要還是依靠供應商，所以它的 inventory 大多會是 supplier 提供，共有 145432 筆資料，剩餘的則是自

己製造抑或是再製品，而供應年分會盡量保持不變，很小機率會換成另一家廠商，共有 17612 筆資料。

ORDER 記錄從 1997 年至今的所有訂單記錄，在 2024 年前的訂單都會是被完成、刪除、退回三種狀態，只有在 2024 年後的會出現未出貨、運送中的狀態，交貨真實日期會在期限前後五天，如果是未出貨、運送中，交貨真實日期會是空值，共有 207087 筆資料。RATE 則是 1997 至今的供應商評分記錄，最終分數會是其他三種分數的加權所得，評分員工都會是 Admin，共有 75810 筆資料。

3.2 重要功能及對應的 SQL 指令

第 1.1 節中我們有提到部分 User 和 Admin 可操作的功能，包含動作：新增、查詢、列舉、更改，與對象員工、供應商、工廠、訂單、評分、零件等。在第 3.2.1 和第 3.2.2 小節中將列出特定情境下，執行這些功能需要的（一或數個）SQL 指令。

3.2.1 給 User 的功能

1. 新增訂單：若要實現此功能，假設情境為「使用者代號 e_id『AA0123456789』想訂購零件編號 inv_id 為『1』的零件，數量 quantity 設為 10 個，下訂日期 order_date 是『2024-01-01』，訂單截止日期 due_date 為『2024-03-01』，訂單狀態 status 初始值為 Not Yet Shipped。」對應的 SQL 指令如下。系統會在 order 的資料表新增一筆訂單的資料，並自動生成訂單編號 o_id。

```
INSERT INTO "Order" (order_date, due_date, quantity, status,
                     e_id, inv_id)
VALUES (2024-01-01, 2024-03-01, 10, Not Yet Shipped,
        AA0123456789, 1)
RETURNING o_id;
```

Listing 1: 新增訂單 SQL 指令

2. 更新訂單：若要實現此功能，假設情境為「使用者想更改編號 o_id 為『1』的訂單，他能選擇想要更改的欄位 item，可能值為狀態 status，回饋 feedback 或抵達日期 arrive_date，如例他能將 status 改為 Delieverd」對應的 SQL 指令如下。系統會把新的值 new_value 存入該欄位中，此外，也會檢查欲更改的欄位是否能被更改。

```
UPDATE "Order"
SET status = Delieverd
WHERE o_id = 1;
```

Listing 2: 更新訂單 SQL 指令

3. 查詢訂單：若要實現此功能，假設情境為「使用者想查詢特定編號 o_id 為『1』的訂單」對應的 SQL 指令如下。系統會執行該查詢並回傳該筆訂單的所有資訊。

```
SELECT *
```

```
FROM "Order"
WHERE o_id = 1
```

Listing 3: 查詢訂單 SQL 指令

4. 列舉訂單：若要實現此功能，假設情境為「使用者想列出特定條件的訂單，如訂購日期 order_date 在『2023-01-01』，或是截止日期 due_date 在『2023-06-30』，或是抵達日期 arrive_date 在『2023-06-01』，抑或是訂單狀態 status 為『delievered』的訂單」對應的其中一種 SQL 指令如下。系統會回傳所有符合條件的訂單及其所有相關資訊。

```
SELECT *
FROM "Order"
WHERE order_date >= '2023-01-01'
```

Listing 4: 列舉訂單 SQL 指令

5. 查詢零件歷史訂單：若要實現此功能，假設情境為「使用者想查詢零件編號 inv_id 為『1』的歷史訂單記錄」對應的其中一種 SQL 指令如下。系統會回傳該零件過去的所有訂單，包含其訂單編號、訂購日期、截止日期等等。

```
SELECT *
FROM "Order"
WHERE inv_id = 1
```

Listing 5: 查詢零件歷史訂單 SQL 指令

6. 列舉零件：若要實現此功能，假設情境為「使用者想列舉特定名稱 inv_name 為『big seat』的零件」對應的 SQL 指令如下，系統會回傳所有與名稱相同之零件的所有資訊。

```
SELECT *
FROM inventory
WHERE inv_name = big seat
```

Listing 6: 列舉零件 SQL 指令

7. 查詢零件資訊：若要實現此功能，假設情境為「使用者想查詢特定零件編號 inv_id 為『1』的零件，除了基本資訊外，還要知道它屬於哪個部件 part 以及來自哪個供應商 supplier 或是工廠 factory」對應的 SQL 指令如下。系統會回傳上述使用者想知道的資訊，由於不一定每個零件會同時被供應商和工廠提供，我們需要 Left Join 確保任意欲查詢的零件都能被搜尋到。

```
SELECT *
FROM inventory AS i
JOIN part AS p ON i.p_id = p.p_id
LEFT JOIN inventory_from_factory as iff on i.inv_id =
    iff.inv_id
```

```
LEFT JOIN inventory_from_supplier as ifs on i.inv_id =
    ifs.inv_id
WHERE i.inv_id = 1
```

Listing 7: 查詢零件資訊 SQL 指令

8. 查詢供應商資訊：若要實現此功能，假設情境為「使用者想查詢供應商名稱 s_name 為『Bryan LLC』的相關資訊」對應的 SQL 指令如下，系統會回傳所有該供應商的資訊，如地址、電話、負責人等等。

```
SELECT *
FROM supplier
WHERE s_name = Bryan LLC
```

Listing 8: 查詢供應商資訊 SQL 指令

9. 查詢工廠資訊：若要實現此功能，假設情境為「使用者想查詢工廠名稱 f_name 為『Stellar Dynamics A』的相關資訊」對應的 SQL 指令如下，系統會回傳所有該工廠的資訊，如地址、電話、負責人等等。

```
SELECT *
FROM factory
WHERE f_name = Stellar Dynamics A
```

Listing 9: 查詢工廠資訊 SQL 指令

10. 列舉評分：若要實現此功能，假設情境為「使用者想查詢某廠商編號 s_id 為『1』在零件提供表現上的歷年評分」對應的 SQL 指令如下。系統會回傳評分分數 score、年度 year，以及評分員工的編號 e_id。

```
SELECT score, year, e_id
FROM rate
WHERE s_id = 1
```

Listing 10: 列舉評分 SQL 指令

11. 搜尋子母零件：若要實現此功能，假設情境為「使用者想查詢某子零件編號 p_id 為『1』的母零件有哪些及其數量，或是母零件編號 p_id 為『2』的子零件有哪些及其數量，同時，他能得知某子零件在某母零件的使用個數 quantity」查詢子零件的對應 SQL 指令如下，系統會回傳欲查詢之零件的相對應子或母零件編號。

```
SELECT *, COUNT(*) OVER () AS total_count
FROM item
WHERE p\_id = 1
```

Listing 11: 搜尋子母零件 SQL 指令

12. 更新密碼：若要實現此功能，假設情境為「使用者編號 e_id 為『EK848641033』的人員想更改登入密碼 password 成『123456』」對應的 SQL 指令如下，系統讀入新值 new_value 後，會在資料庫進行更新。

```
UPDATE "employee"
SET password = 123456
WHERE e_id = EK848641033;
```

Listing 12: 更新密碼 SQL 指令

3.2.2 給 Admin 的功能

Admin 能執行所有上述 User 的功能以及下列幾種功能。

1. 註冊員工：若要實現此功能，假設情境為「主管想要新增某位員工，將其名字 e_name『Kelly Ortberg』、開始工作日期 start_date『2010-10-15』、密碼 password『gumphion8904』、主管 mgr_id『XP717711713』以及角色 role『Admin』各自輸入值，員工編號 e_id 『EK848641033』為隨機生成，並遵守前兩位為英文字母，後十位為數字的規則」對應的 SQL 指令如下。系統會先隨機生成員工編號，將連同編號把其他新的資訊存入資料庫的表 employee。

```
SELECT 'EM' || lpad(nextval('employee_id_seq')::text, 10,
    '0');
INSERT INTO employee (e_id, e_name, start_date, password,
    mgr_id, role)
VALUES (EK848641033, Kelly Ortberg, 2010-10-15, gumphion8904,
    XP717711713, Admin)
```

Listing 13: 註冊員工 SQL 指令

2. 更改員工：若要實現此功能，假設情境為「主管想要變更員工編號 e_id 為『EK848641033』的資料，能更改他的名字 name、開始工作日期 start_date、密碼 password 及主管 mgr_id，例如他想將該員工的主管變更成編號『OF918970618』的職員」對應的 SQL 指令如下。系統將把主管輸入的新值更新至指定行中。

```
UPDATE employee
SET mgr_id = OF918970618
WHERE e_id = EK848641033;
```

Listing 14: 更改員工 SQL 指令

3. 列舉員工：若要實現此功能，假設情境為「主管想要列舉出開始工作日期 start_date 為『2010-10-15』之後的員工，或是主管編號 mgr_id 為『XP717711713』的所有員工」前者條件對應的 SQL 指令如下。系統將回傳符合條件的員工，至少一個條件存在即可。

```
SELECT *
FROM employee
```

```
WHERE start_date >= '2010-10-15'
```

Listing 15: 列舉員工 SQL 指令

4. 查詢員工：若要實現此功能，假設情境為「主管想查詢員工編號 e_id 為『EK848641033』的相關資料」對應的 SQL 指令如下。系統將回傳員工的所有資料。

```
SELECT *
FROM employee
WHERE e_id = EK848641033
```

Listing 16: 查詢員工 SQL 指令

5. 新增供應商：若要實現此功能，假設情境為「主管想新增某家供應商，他會輸入其名稱 s_name『Bryan LLC』、所在國家 s_country『Vietnam』、詳細地址 s_address『693 Patton Wells Suite 871 North Coryfort, MP 22185』、聯絡電話 s_phone『+84 0011908440』以及負責人姓名 super_name『Jonathan Duncan』」對應的 SQL 指令如下。系統將依已存在的供應商編號 s_id 自動生成下一個序列性的編號。

```
INSERT INTO supplier (s_name, s_country, s_address, s_phone,
    super_name)
VALUES (Bryan LLC, Vietnam, 693 Patton Wells Suite 871 North
    Coryfort, MP 22185, +84 0011908440, Jonathan Duncan)
RETURNING s_id
```

Listing 17: 新增供應商 SQL 指令

6. 更改供應商：若要實現此功能，假設情境為「主管想更改供應商編號 s_id『1』的資訊，他能更改的欄位為名稱 s_name、國家 s_country、地址 s_address 和負責人姓名 super_name，例如他想更新負責人姓名為『Kayla Mason』」對應的 SQL 指令如下。系統將把主管輸入的新值更新至資料庫相應欄位。

```
UPDATE supplier
SET s\_name = Kayla Mason
WHERE s_id = 1;
```

Listing 18: 更改供應商 SQL 指令

7. 新增評分：若要實現此功能，假設情境為「員工編號 e_id 為『AA0123456789』的主管，想為供應商編號 s_id 為『1』，在年份 year 為『2024』打上準時性分數 on time『80』、交貨品質分數 quality『80』、售後服務分數 after_sales_service『80』以及最終加權分數 final_score『80』」對應的 SQL 指令如下。系統將把這四項分數存進相對應的被評分供應商，且會紀錄該評分員工的身份。

```
INSERT INTO rate (s_id, year, on_time, quality,
    after_sales_service, final_score, e_id)
VALUES (1, 2024, 80, 80, 80, 80, AA0123456789)
```

Listing 19: 新增評分 SQL 指令

8. 更改評分：若要實現此功能，假設情境為「某主管想更改年度 year 為『2024』，供應商編號 s_id 為『1』的評分，他能選擇更改最終分數 final_score 或是評分員工編號 e_id，例如他想更改最終分數變成 90」對應的 SQL 指令如下。系統將把新資訊更新至資料表 RATE 的相對應欄位。

```
UPDATE rate
SET final_score = 90
WHERE year = 2024 AND s_id = 1;
```

Listing 20: 更改評分 SQL 指令

9. 新增零件：若要實現此功能，假設情境為「主管想新增某個零件，名稱 inv_name 為『big seat』，狀態 status 為 not using，對應的部件編號 p_id 為『1』」對應的 SQL 指令如下。系統為新增該零件至資料庫，並依據已存在的編號 inv_id 自動生成下一個序號。

```
INSERT INTO inventory (inv_name, status, p_id)
VALUES (big seat, not using, 1)
RETURNING inv_id
```

Listing 21: 新增零件 SQL 指令

10. 更新部件：若要實現此功能，假設情境為「主管想更新部件編號 p_id『1』的資訊，他能更改其名稱 p_name、標準 standard、長度 length、寬度 width 以及高度 height，例如他想長度改為 20」對應的 SQL 指令如下。系統會把變更的數值存入指定行。

```
UPDATE part
SET length = 20
WHERE p_id = 1;
```

Listing 22: 更新部件 SQL 指令

11. 新增子母零件：若要實現此功能，假設情境為「主管想新增某個子零件編號 child_inv 為『1』，在母零件編號 parent_inv『2』中會使用數量 quantity 『10』個」對應的 SQL 指令如下。系統為新增該子母零件關係至資料庫。

```
INSERT INTO item (parent_inv, child_inv, quantity)
VALUES (2, 1, 10)
```

Listing 23: 新增子母零件 SQL 指令

3.3 SQL 指令效能優化與索引建立分析

在 find history 功能中，我們需要在二十萬筆 order 中搜尋 inv_id 的歷史紀錄，在沒有建立 index 時會花費較久。

	QUERY PLAN text
1	Gather (cost=1000.00..13418.08 rows=35 width=405) (actual time=0.967..220.495 rows=47 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on "Order" (cost=0.00..12414.58 rows=15 width=405) (actual time=0.066..13.824 rows=16 loop...
5	Filter: (inv_id = 1)
6	Rows Removed by Filter: 69014
7	Planning Time: 0.167 ms
8	Execution Time: 220.545 ms

Figure 3: No index

	QUERY PLAN text
1	Bitmap Heap Scan on "Order" (cost=4.57..139.17 rows=35 width=405) (actual time=0.034..0.096 rows=47 loops=...
2	Recheck Cond: (inv_id = 1)
3	Heap Blocks: exact=46
4	-> Bitmap Index Scan on idx_order (cost=0.00..4.56 rows=35 width=0) (actual time=0.015..0.015 rows=47 loops...
5	Index Cond: (inv_id = 1)
6	Planning Time: 0.123 ms
7	Execution Time: 0.120 ms

Figure 4: Index on inv_id

在重複執行後，可以得到未建立 index 時每次執行時間大約在 250ms 左右，而有建立 index 大約在 0.1ms 左右。可以發現當我們對 inv_id 建立 index 時，速度有顯著提升。可能原因是因為 order 本身是按照 o_id 排序，因此在查找 inv_id 時，需要用平行序列掃描，整個 Scan 一次，而在我們的查找情境下，也就是篩選條件較精確且匹配的資料頁較多狀況下，建立 index 後，使用索引是較快的方法。

3.4 並行控制

在我們的系統中，登入頁面後可以選擇更改訂單狀態 update order status，我們認為這個階段需要併行控制。更改訂單狀態的流程是先輸入欲更改的訂單編號，接著選擇要更改的項目，再輸入新的值。這一項操作需要加鎖，因為可更改任一筆訂單，若同時有其他人也想對該訂單做更新，兩人新輸入的值可能沒辦法同時都被更新到資料庫。因此我們選擇在決定執行這項功能、輸入完指定編號的訂單後就加鎖，防止其他人在他做更改時也來修正訂單狀態，並在行為完成後釋放鎖，這個做法能確保訂單在完成修改前都是上鎖狀態。我們使用 SQL 指令 FOR UPDATE 執行加鎖。

4 分工資訊

周子馨：各項功能的 SQL 指令、相對應後端程式及 html 撰寫。
江睿宸：database 建立、前後端架構生成、系統測試、錄影片。
陳承妤：資料生成、前端設計修改、錄影片。

5 專案心得

周子馨：這個專案整體都非常困難，尤其是前端、後端、SQL 指令都要互相連接的部分，不過因為有助教提供的範本，對如何撰寫這些程式碼才有一些些頭緒。雖然花了很多時間在這作業上，但它真的是一個讓我們練習前後端、寫出完整系統很好的機會，不僅學習到寫 html、實現想操作的功能，過程中還需解決 query 回傳格式、尋找資料庫失敗等問題，總之是份很累又收穫很多的專案。

江睿宸：剛開始寫架構的時候覺得應該不會太困難，就是一樣的東西稍微改 query 就好了，殊不知開始測試的時後就一直有問題，一方面一直想到要新增功能，一方面回傳又一直跳 error，就這樣改了又改好幾個小時就過去了，不得不被逼著熬了幾個夜。在這次寫專案的過程中，了解到 Github 以及版本控制的重要性，看來還是需要花時間研究一下 GitHub 正確的用法。

陳承妤：一開始跟周子馨研究助教的專案範例架構怎麼寫，後來發現如果要用前端設計的話，一開始就要定好，否則之後擴充功能只會越來越麻煩，然後因為自己是提案人，對真實測資要長怎樣有想法，所以擔任生資料庫資料的工作，但發現各種表的連結關係，還有要讓測資真實必須設定更複雜的條件，有時想一想會自己卡住或是寫出較沒效率的生資料方法，好在後來還是有研究出來。前端的部分，有花時間調顯示方式，還有 CSS Style 跟 Script 裡回傳結果的 HTML 架構。感覺自己花了很多時間在處理繁雜的事，或許是自己的能力還不夠、邏輯不好，希望之後事情能處理得更有效率。