

資料庫設計期末專題

- 訂單 / 進銷存系統
- 組員：第20組
- 楊竣安411631053、陳奕嘉411631384

目錄

系統目標

功能構想

ER 圖

正規化與設計原則

資料表語法與結構

進階 SQL 功能

測試與執行結果組員分工

系統目標

此專題在設計一套 訂單 / 進銷存資料庫系統，協助企業有效管理商品的進貨、銷售與庫存狀況。系統將以關聯式資料庫為基礎，實作完整的表格設計、資料正規化、SQL 查詢及進階功能（如視圖、儲存程序、觸發器等），以支援企業日常營運中的資料處理與分析需求。

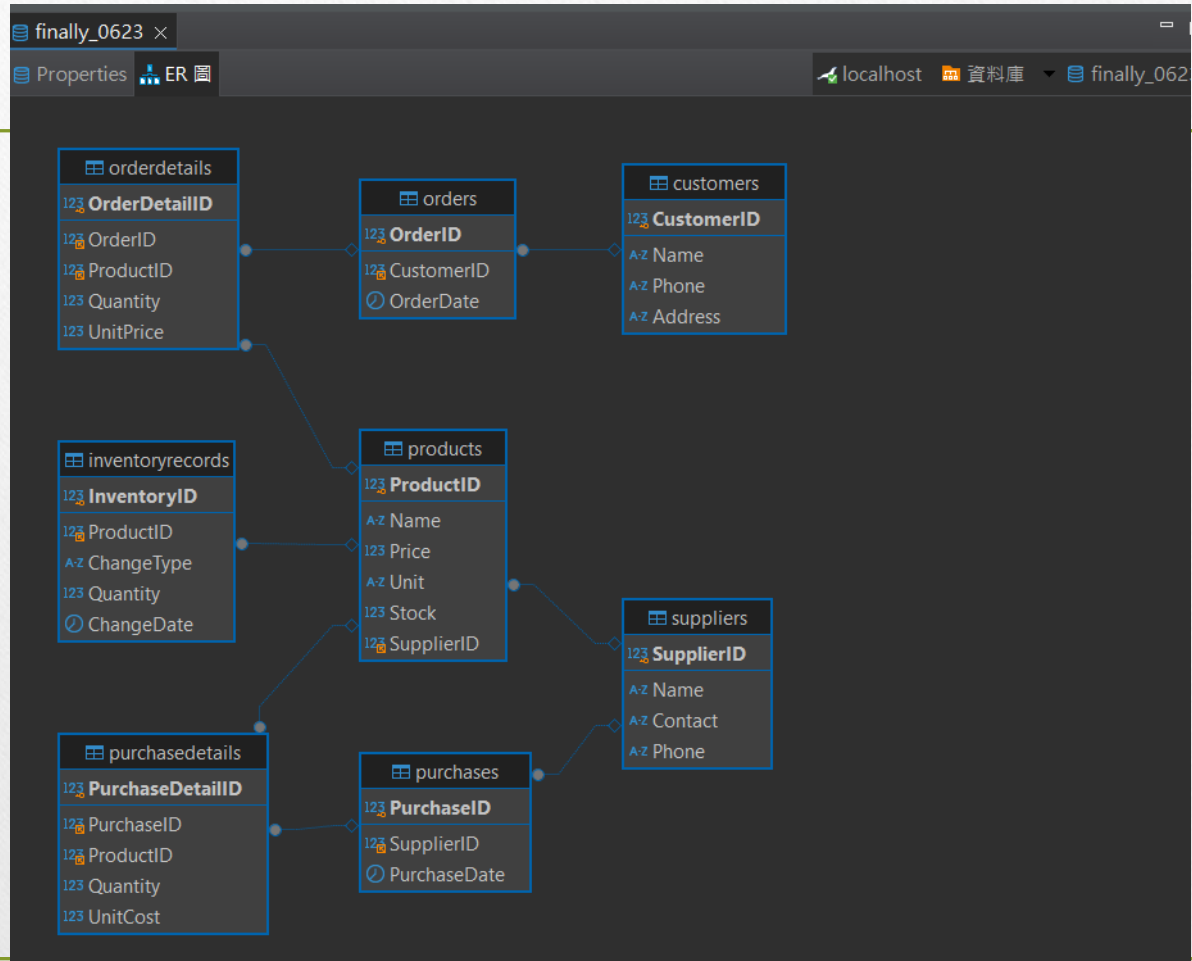
功能構想

1. 客戶與供應商管理客戶：記錄姓名、電話、地址等資訊。供應商：記錄名稱、聯絡人等聯絡資訊。
2. 商品管理:商品名稱、單價、單位、初始庫存量、供應商來源等。可從供應商進貨並調整庫存。
3. 訂單管理:建立訂單並對應客戶。每筆訂單包含一筆或多筆明細（商品、數量、單價）。自動計算總金額，並更新庫存記錄。
4. 建立進貨單並對應供應商。每筆進貨記錄包含多項商品數量與成本價。自動記錄庫存進帳資訊。

功能構想

5. 庫存異動管理:每筆進貨或銷售皆會產生一筆「IN / OUT」的異動記錄。可查詢每項商品的總進貨量、出貨量與即時庫存。
6. 查詢與分析報表功能查詢每位客戶的訂單與總金額。商品銷售總量排行。商品與供應商關係一覽。商品目前庫存計算與安全存量警示。各商品歷史異動記錄 (IN/OUT 細節) 。
7. 進階 SQL 應用功能使用 View 建立熱門商品檢視表。撰寫 Stored Procedure 實作訂單處理邏輯。使用 Trigger 實作自動記錄庫存異動。使用 Transaction 確保訂單資料一致性。

ER圖



正規化與設計原則

第一正規形式 (1NF)

每個欄位都是

不可再分的原子值。

Name、Phone、Address

都是不可再分割的欄位沒有

出現像是「多個電話寫在同一欄」

這種反正規化設計 達到 1NF

```
use finally_0623

CREATE TABLE Customers (
  CustomerID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Phone VARCHAR(20),
  Address TEXT
) ENGINE=InnoDB;
```

Statistics 1 ×	Value
Query	CREATE TABLE Customers (CustomerID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Phone VARCHAR(20), Address TEXT) ENGINE=InnoDB
Updated Rows	0
Execute time	0.014s
Start time	Fri Jun 13 08:57:53 CST 2025
Finish time	Fri Jun 13 08:57:53 CST 2025

正規化與設計原則

第二正規形式 (2NF)

表格已經符合 1NF，並且每個非主鍵欄位都完全依賴主鍵。

OrderDetailID 是主鍵
Quantity 和 **UnitPrice** 都依賴於 **OrderDetailID**，而不是部分依賴（例如只依賴 **OrderID**）→ 達到 2NF

```
CREATE TABLE OrderDetails (
  OrderDetailID INT AUTO_INCREMENT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT NOT NULL,
  UnitPrice DECIMAL(10,2) NOT NULL,
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
) ENGINE=InnoDB;
```

Name	Value
Query	CREATE TABLE OrderDetails (OrderDetailID INT AUTO_INCREMENT PRIMARY KEY, OrderID INT, ProductID INT, Quantity INT NOT NULL, UnitPrice DECIMAL(10,2) NOT NULL, FOREIGN KEY (OrderID) REFERENCES Orders(OrderID), FOREIGN KEY (ProductID) REFERENCES Products(ProductID)) ENGINE=InnoDB
Updated Rows	0
Execute time	0.003s
Start time	Fri Jun 13 08:58:30 CST 2025
Finish time	Fri Jun 13 08:58:30 CST 2025

正規化與設計原則

第三正規形式 (3NF) 表格已經符合 2NF，且非主鍵欄位不得依賴其他非主鍵欄位（消除遞移依賴）。ProductID 是主鍵所有欄位如 Name、Price、Stock、SupplierID 都只依賴主鍵，沒有一個欄位依賴另一個非主鍵欄位（例如 Price 沒有依賴 Name）✅ → 達到 3NF

```
CREATE TABLE Products (  
  ProductID INT AUTO_INCREMENT PRIMARY KEY,  
  Name VARCHAR(100) NOT NULL,  
  Price DECIMAL(10,2) NOT NULL,  
  Unit VARCHAR(20),  
  Stock INT DEFAULT 0,  
  SupplierID INT,  
  FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)  
) ENGINE=InnoDB;
```

Name	Value
Query	CREATE TABLE Products (ProductID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Price DECIMAL(10,2) NOT NULL, Unit VARCHAR(20), Stock INT DEFAULT 0, SupplierID INT, FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)) ENGINE=InnoDB
Updated Rows	0
Execute time	0.004s
Start time	Fri Jun 13 08:58:13 CST 2025
Finish time	Fri Jun 13 08:58:13 CST 2025

資料表語法與結構

```
CREATE TABLE Customers (  
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(100) NOT NULL,  
    Phone VARCHAR(20),  
    Address TEXT  
) ENGINE=InnoDB;
```

建立客戶表

```
CREATE TABLE Suppliers (  
    SupplierID INT AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(100) NOT NULL,  
    Contact VARCHAR(100),  
    Phone VARCHAR(20)  
) ENGINE=InnoDB;
```

建立供應商表

資料表語法與結構

```
CREATE TABLE Products (  
  ProductID INT AUTO_INCREMENT PRIMARY KEY,  
  Name VARCHAR(100) NOT NULL,  
  Price DECIMAL(10,2) NOT NULL,  
  Unit VARCHAR(20),  
  Stock INT DEFAULT 0,  
  SupplierID INT,  
  FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)  
) ENGINE=InnoDB;  
  
CREATE TABLE Orders (  
  OrderID INT AUTO_INCREMENT PRIMARY KEY,  
  CustomerID INT,  
  OrderDate DATE DEFAULT CURRENT_DATE,  
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
) ENGINE=InnoDB;
```

建立產品表

建立訂單主表

資料表語法與結構

```
● CREATE TABLE OrderDetails (  
    OrderDetailID INT AUTO_INCREMENT PRIMARY KEY,  
    OrderID INT,  
    ProductID INT,  
    Quantity INT NOT NULL,  
    UnitPrice DECIMAL(10,2) NOT NULL,  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
) ENGINE=InnoDB;  
  
● CREATE TABLE Purchases (  
    PurchaseID INT AUTO_INCREMENT PRIMARY KEY,  
    SupplierID INT,  
    PurchaseDate DATE DEFAULT CURRENT_DATE,  
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)  
) ENGINE=InnoDB;
```

建立訂單明細

建立進貨主表

資料表語法與結構

```
CREATE TABLE PurchaseDetails (  
  PurchaseDetailID INT AUTO_INCREMENT PRIMARY KEY,  
  PurchaseID INT,  
  ProductID INT,  
  Quantity INT NOT NULL,  
  UnitCost DECIMAL(10,2),  
  FOREIGN KEY (PurchaseID) REFERENCES Purchases(PurchaseID),  
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
) ENGINE=InnoDB;  
  
CREATE TABLE InventoryRecords (  
  InventoryID INT AUTO_INCREMENT PRIMARY KEY,  
  ProductID INT,  
  ChangeType VARCHAR(10) CHECK (ChangeType IN ('IN', 'OUT')),  
  Quantity INT,  
  ChangeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
) ENGINE=InnoDB;
```

建立進貨明細

建立庫存異動
紀錄

資料表語法與結構

```
● INSERT INTO Customers (Name, Phone, Address) VALUES  
('王小明', '0912345678', '台北市中正區'),  
('李小美', '0922333444', '新北市板橋區');
```

插入客戶資料

```
● INSERT INTO Suppliers (Name, Contact, Phone) VALUES  
('全方位文具供應', '張經理', '0223456789'),  
('三商行', '陳小姐', '0222334455');
```

插入供應商資料

```
● INSERT INTO Products (Name, Price, Unit, Stock, SupplierID) VALUES  
('A4影印紙', 120, '包', 50, 1),  
('原子筆', 10, '支', 100, 1),  
('記事本', 45, '本', 80, 2);
```

插入商品資料

資料表語法與結構

```
INSERT INTO Orders (CustomerID, OrderDate) VALUES (1, '2025-06-10');
```

將一筆訂單資料新增到
Orders 訂單主表中

資料表語法與結構

```
INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice) VALUES  
(1, 1, 5, 120),  
(1, 2, 10, 10);
```

插入訂單明細

```
INSERT INTO Purchases (SupplierID, PurchaseDate) VALUES (1, '2025-06-01');
```

插入進貨主表資料

```
INSERT INTO PurchaseDetails (PurchaseID, ProductID, Quantity, UnitCost) VALUES  
(1, 1, 100, 100),  
(1, 2, 200, 8);
```

插入進貨明細

```
INSERT INTO InventoryRecords (ProductID, ChangeType, Quantity) VALUES  
(1, 'IN', 100),  
(2, 'IN', 200),  
(1, 'OUT', 5),  
(2, 'OUT', 10);
```

插入庫存異動紀錄

資料表語法與結構

```
SELECT
  p.ProductID,
  p.Name,
  p.Stock,
  SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
           WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
           ELSE 0 END) AS StockChange,
  p.Stock + SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
                    WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
                    ELSE 0 END) AS CurrentStock
FROM Products p
LEFT JOIN InventoryRecords ir ON p.ProductID = ir.ProductID
GROUP BY p.ProductID, p.Name, p.Stock;
```

查詢客戶訂單與總金額

資料表語法與結構

```
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    SUM(od.Quantity * od.UnitPrice) AS TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
GROUP BY o.OrderID, c.Name;
```

查詢商品目前庫存計算

資料表語法與結構

```
●SELECT
    s.Name AS Supplier,
    p.Name AS Product,
    p.Price,
    p.Unit
FROM Products p
JOIN Suppliers s ON p.SupplierID = s.SupplierID
ORDER BY s.Name;
```

查詢商品與供應商資料

資料表語法與結構

```
SELECT  
    p.Name,  
    SUM(od.Quantity) AS TotalSold  
FROM OrderDetails od  
JOIN Products p ON od.ProductID = p.ProductID  
GROUP BY p.Name  
ORDER BY TotalSold DESC;
```

查詢商品銷售總數排行

進階 SQL 功能

```
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    SUM(od.Quantity * od.UnitPrice) AS TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
GROUP BY o.OrderID, c.Name;
```

GROUP BY 搭配 SUM / 聚合查詢

使用了 SUM 聚合函數來計算訂單總金額。使用 JOIN + GROUP BY 的多表彙總查詢。

屬於進階查詢設計與報表需求。

進階 SQL 功能

```
SELECT
  p.ProductID,
  p.Name,
  p.Stock,
  SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
          WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
          ELSE 0 END) AS StockChange,
  p.Stock + SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
                    WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
                    ELSE 0 END) AS CurrentStock
FROM Products p
LEFT JOIN InventoryRecords ir ON p.ProductID = ir.ProductID
GROUP BY p.ProductID, p.Name, p.Stock;
```

複雜條件與庫存計算（含 CASE 條件語句）

使用了 CASE WHEN 條件式進行邏輯運算。即時計算「目前庫存」的邏輯。

屬於資料邏輯運算與分析。

進階 SQL 功能

```
● CREATE VIEW TopSellingProducts AS  
  SELECT p.Name, SUM(od.Quantity) AS TotalSold  
  FROM OrderDetails od  
  JOIN Products p ON od.ProductID = p.ProductID  
  GROUP BY p.Name  
  ORDER BY TotalSold DESC;
```

建立 VIEW 視圖

VIEW 是一種虛擬資料表，可以重複使用複雜查詢。

進階 SQL 功能

```
DELIMITER //
```

```
● CREATE PROCEDURE GetCustomerTotal(IN cid INT)  
BEGIN  
  SELECT o.OrderID, SUM(od.Quantity * od.UnitPrice) AS Total  
  FROM Orders o  
  JOIN OrderDetails od ON o.OrderID = od.OrderID  
  WHERE o.CustomerID = cid  
  GROUP BY o.OrderID;  
END //
```

```
DELIMITER ;
```

將查詢封裝為 Stored Procedure

儲存程序 (Stored Procedure) 將查詢邏輯封裝在資料庫中，可重複執行。

進階 SQL 功能

```
● CREATE TRIGGER after_order_insert  
  AFTER INSERT ON OrderDetails  
  FOR EACH ROW  
  BEGIN  
    INSERT INTO InventoryRecords (ProductID, ChangeType, Quantity)  
    VALUES (NEW.ProductID, 'OUT', NEW.Quantity);  
  END;
```

Trigger 可自動對其他資料表做異動，
維持資料一致性。

進階 SQL 功能

```
INSERT INTO Orders (CustomerID) VALUES (1);
```

TRANSACTION 控制資料一致性

使用交易機制避免資料只插入一半導致錯誤。

測試與執行結果

```
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    SUM(od.Quantity * od.UnitPrice) AS TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
GROUP BY o.OrderID, c.Name;
```

客戶訂單與總金額查詢：
顯示訂單的客戶名稱與總金額。

SQL Server Enterprise Manager - Query Results

1 (+) 1 x

Enter a SQL expression to filter results

123 OrderID	124 CustomerName	125 TotalAmount
1	王小明	700

測試與執行結果

```
●SELECT
  p.ProductID,
  p.Name,
  p.Stock,
  SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
        WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
        ELSE 0 END) AS StockChange,
  p.Stock + SUM(CASE WHEN ir.ChangeType = 'IN' THEN ir.Quantity
        WHEN ir.ChangeType = 'OUT' THEN -ir.Quantity
        ELSE 0 END) AS CurrentStock
FROM Products p
LEFT JOIN InventoryRecords ir ON p.ProductID = ir.ProductID
GROUP BY p.ProductID, p.Name, p.Stock;
```

商品目前庫存計算（含異動紀錄）：

查詢每項商品目前剩餘庫存量。

products 1 ×

SELECT p.ProductID, p.Name, ||
Enter a SQL expression to filter results (use Ctrl+Space)

	ProductID	Name	Stock	StockChange	CurrentStock
1	1	A4影印紙	50	95	145
2	2	原子筆	100	190	290
3	3	記事本	80	0	80

測試與執行結果

```
●SELECT
    s.Name AS Supplier,
    p.Name AS Product,
    p.Price,
    p.Unit
FROM Products p
JOIN Suppliers s ON p.SupplierID = s.SupplierID
ORDER BY s.Name;
```

商品與供應商對應查詢：
了解每項商品由哪個供應商提供、其單價與單位。

Suppliers(+) 1 ×

SELECT sName AS Supplier, p | Enter a SQL expression to filter results (

	A-Z Supplier	A-Z Product	123 Price	A-Z Unit	
1	三商行	記事本	45	本	
2	全方位文具供應	A4影印紙	120	包	
3	全方位文具供應	原子筆	10	支	

測試與執行結果

```
SELECT
    p.Name,
    SUM(od.Quantity) AS TotalSold
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.Name
ORDER BY TotalSold DESC;
```

products 1 ×

SELECT p.Name, SUM(od.Quantity) AS TotalSold FROM OrderDetails od JOIN Products p ON od.ProductID = p.ProductID GROUP BY p.Name ORDER BY TotalSold DESC;

	A-Z Name	123 TotalSold
1	原子筆	10
2	A4影印紙	5

商品銷售總數排行（熱門商品）：

顯示所有商品的總銷售數量與排行。

組員分工

- 楊竣安411631053 :
 - 製作ppt、資料庫設計、demo錄影
- 陳奕嘉411631384 :
 - 製作ppt、口頭報告ppt、製作系統文件

**報告結束
謝謝大家**