MYSQL 資料庫正規化完整指南

> 從 1NF 到 3NF 的完整演進過程 - 露營用品銷售系統實例

為什麼需要正規化？

以下是一個未正規化的專案管理資料表，存在許多問題：

專案號碼	專案名稱	員工編號	員工姓名	職稱	時薪	工時	給付
21	庫存系統	102	戴友鑫	工程師	500	20	10000
		112	劉燕玲	分析師	700	10	7000
		105	薛玉芳	副理	800	10	8000
		107	趙逸仙	工程師	500	20	10000
25	財務系統	107	趙逸仙	工程師	500	10	5000
		110	陳立民	分析師	700	20	14000
		105	薛玉芳	副理	800	20	16000
27	行銷系統	108	張為文	副理	800	10	8000
		112	劉燕玲	分析師	700	20	14000
		102	戴友鑫	工程師	500	10	5000

可能產生的問題

- **資料重複**：劉燕玲的資料重複出現多次

- **更新異常**：職稱錯誤需要多處修改（分析師 vs 系統分析師）
- **插入異常**：新進員工沒有專案無法建立記錄
- **刪除異常**：員工離職會導致專案資料遺失

正規化流程

```
未正規化資料 → 1NF → 2NF → 3NF
```

第一正規化 (1NF) - 原子值

要求： 所有欄位都具有原子值（不可再分割），且每一列必須唯一識別。

露營用品銷售 - 1NF 結構

```sql
-- 露營用品銷售 - 1NF 結構
CREATE TABLE 訂單明細 (
    訂單編號 VARCHAR(20) NOT NULL,
    客戶ID INT NOT NULL,
    客戶姓名 VARCHAR(50) NOT NULL,
    客戶電話 VARCHAR(15),
    產品ID INT NOT NULL,
    產品名稱 VARCHAR(100) NOT NULL,
    類別 VARCHAR(30),
    單價 DECIMAL(10,2) NOT NULL,
    購買數量 INT NOT NULL,
    訂單日期 DATE NOT NULL,
    PRIMARY KEY (訂單編號, 產品ID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

達成： 使用複合主鍵唯一識別每筆記錄，所有欄位都是原子值。

第二正規化 (2NF) - 完全函數相依

要求： 滿足1NF，且所有非主鍵屬性必須完全依賴於主鍵，不能只依賴於主鍵的一部分。

問題分析

在1NF資料表中，我們發現：

- **客戶資訊**（客戶ID, 客戶姓名, 客戶電話）只依賴於客戶ID
- **產品資訊**（產品ID, 產品名稱, 類別, 單價）只依賴於產品ID
- **訂單日期**只依賴於訂單編號

2NF 結構設計

```
-- 客戶資料表
CREATE TABLE 客戶 (
    客戶ID INT AUTO_INCREMENT PRIMARY KEY,
    客戶姓名 VARCHAR(50) NOT NULL,
    客戶電話 VARCHAR(15),
    註冊日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 產品資料表
CREATE TABLE 產品 (
    產品ID INT AUTO_INCREMENT PRIMARY KEY,
    產品名稱 VARCHAR(100) NOT NULL,
    類別 VARCHAR(30) NOT NULL,
    單價 DECIMAL(10,2) NOT NULL CHECK(單價 > 0),
    建立日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 訂單主表
CREATE TABLE 訂單 (
    訂單編號 VARCHAR(20) PRIMARY KEY,
    客戶ID INT NOT NULL,
    訂單日期 DATE NOT NULL,
    訂單總額 DECIMAL(12,2) DEFAULT 0,
    FOREIGN KEY (客戶ID) REFERENCES 客戶(客戶ID) ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 訂單明細表
CREATE TABLE 訂單明細 (
    訂單編號 VARCHAR(20),
    產品ID INT,
    購買數量 INT NOT NULL CHECK(購買數量 > 0),
    PRIMARY KEY (訂單編號, 產品ID),
    FOREIGN KEY (訂單編號) REFERENCES 訂單(訂單編號) ON DELETE
CASCADE,
    FOREIGN KEY (產品ID) REFERENCES 產品(產品ID) ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

達成： 消除了部分相依，每個屬性都完全依賴於所屬表的主鍵。

第三正規化 (3NF) - 消除傳遞相依

要求： 滿足2NF，且所有非主鍵屬性不可傳遞相依於主鍵（非主鍵屬性之間不應有依賴關係）。

問題分析

在2NF中，產品表的「類別」可能會依賴於其他產品屬性，形成傳遞相依，需要進一步分離。

3NF 完整結構設計

```
-- 建立資料庫
CREATE DATABASE 露營用品銷售系統
    DEFAULT CHARACTER SET = 'utf8mb4'
    COLLATE utf8mb4_0900_ai_ci;

USE 露營用品銷售系統;

-- 客戶資料表
CREATE TABLE 客戶 (
    客戶ID INT AUTO_INCREMENT PRIMARY KEY,
    客戶姓名 VARCHAR(50) NOT NULL,
    客戶電話 VARCHAR(15),
    客戶郵箱 VARCHAR(100) UNIQUE,
    客戶地址 TEXT,
    註冊日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_客戶姓名 (客戶姓名),
    INDEX idx_客戶電話 (客戶電話)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 產品類別資料表
CREATE TABLE 類別 (
    類別ID INT AUTO_INCREMENT PRIMARY KEY,
    類別名稱 VARCHAR(30) NOT NULL UNIQUE,
    類別描述 TEXT,
    建立日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 產品資料表
CREATE TABLE 產品 (
```

```
    產品ID INT AUTO_INCREMENT PRIMARY KEY,
    產品名稱 VARCHAR(100) NOT NULL,
    類別ID INT NOT NULL,
    單價 DECIMAL(10,2) NOT NULL CHECK(單價 > 0),
    庫存量 INT NOT NULL DEFAULT 0 CHECK(庫存量 >= 0),
    產品描述 TEXT,
    產品規格 JSON,
    建立日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    更新日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_產品名稱 (產品名稱),
    INDEX idx_類別 (類別ID),
    INDEX idx_單價 (單價),
    FOREIGN KEY (類別ID) REFERENCES 類別(類別ID) ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 訂單主表
CREATE TABLE 訂單 (
    訂單編號 VARCHAR(20) PRIMARY KEY,
    客戶ID INT NOT NULL,
    訂單日期 DATE NOT NULL DEFAULT (CURDATE()),
    訂單狀態 ENUM('處理中', '已付款', '已出貨', '已送達', '已完
成', '已取消')
                NOT NULL DEFAULT '處理中',
    付款方式 ENUM('信用卡', '轉帳', 'PayPal', '貨到付款') NOT
NULL,
    運送地址 TEXT,
    訂單總額 DECIMAL(12,2) DEFAULT 0 CHECK(訂單總額 >= 0),
    備註 TEXT,
    建立時間 TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    更新時間 TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_客戶 (客戶ID),
    INDEX idx_訂單日期 (訂單日期),
    INDEX idx_訂單狀態 (訂單狀態),
    FOREIGN KEY (客戶ID) REFERENCES 客戶(客戶ID) ON DELETE
RESTRICT
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 訂單明細表
CREATE TABLE 訂單明細 (
    明細ID INT AUTO_INCREMENT,
    訂單編號 VARCHAR(20),
    產品ID INT,
    購買數量 INT NOT NULL CHECK(購買數量 > 0),
    實際單價 DECIMAL(10,2) NOT NULL CHECK(實際單價 >= 0),
    小計 DECIMAL(12,2) GENERATED ALWAYS AS (購買數量 * 實際單
價) STORED,
    PRIMARY KEY (明細ID),
    UNIQUE KEY unique_訂單產品 (訂單編號, 產品ID),
    INDEX idx_訂單 (訂單編號),
    INDEX idx_產品 (產品ID),
    FOREIGN KEY (訂單編號) REFERENCES 訂單(訂單編號) ON DELETE
CASCADE,
    FOREIGN KEY (產品ID) REFERENCES 產品(產品ID) ON DELETE
RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 供應商資料表（擴展功能）
CREATE TABLE 供應商 (
    供應商ID INT AUTO_INCREMENT PRIMARY KEY,
    供應商名稱 VARCHAR(100) NOT NULL,
    聯絡人 VARCHAR(50),
    聯絡電話 VARCHAR(15),
    聯絡郵箱 VARCHAR(100),
    公司地址 TEXT,
    統一編號 VARCHAR(10) UNIQUE,
    建立日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

-- 產品供應關聯表（多對多關聯）
CREATE TABLE 產品供應 (
    產品ID INT,
```

```
    供應商ID INT,
    供應價格 DECIMAL(10,2) NOT NULL CHECK(供應價格 > 0),
    供應週期 INT DEFAULT 7 COMMENT '供應週期（天）',
    最小訂購量 INT DEFAULT 1,
    建立日期 TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (產品ID, 供應商ID),
    FOREIGN KEY (產品ID) REFERENCES 產品(產品ID) ON DELETE
CASCADE,
    FOREIGN KEY (供應商ID) REFERENCES 供應商(供應商ID) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

達成： 完全消除傳遞相依，類別獨立成表，結構最佳化。

正規化後的優點

消除資料重複

每個資訊只儲存在一個地方，大幅減少儲存空間需求，避免不一致問題。

更新容易

修改資料時只需更新一個地方，確保資料一致性，降低維護成本。

資料完整性

透過外鍵約束和檢查約束，確保資料的正確性和完整性。

擴展性佳

模組化設計讓系統更容易擴展新功能，適應業務變化。

範例資料插入

```sql
-- 插入類別資料
INSERT INTO 類別 (類別名稱, 類別描述) VALUES
('帳篷', '各種露營帳篷，包含單人、雙人、家庭型'),
('睡袋', '保暖睡袋，適合不同季節使用'),
('炊具', '露營用炊事器具，烹飪必備'),
('照明', '露營燈具、手電筒等照明設備');

-- 插入客戶資料
INSERT INTO 客戶 (客戶姓名, 客戶電話, 客戶郵箱, 客戶地址) VALUES
('王小明', '0912345678', 'wang@email.com', '台北市信義區信義路100號'),
('李美玲', '0923456789', 'lee@email.com', '新北市板橋區中山路200號'),
('張志豪', '0934567890', 'zhang@email.com', '桃園市中壢區中正路300號');

-- 插入產品資料
INSERT INTO 產品 (產品名稱, 類別ID, 單價, 庫存量, 產品描述)
VALUES
('Coleman 4人帳篷', 1, 3500.00, 10, '適合4人使用的家庭帳篷'),
('保暖睡袋', 2, 1200.00, 15, '適合5-15度使用的保暖睡袋'),
('瓦斯爐', 3, 800.00, 20, '輕便型單爐頭瓦斯爐'),
('LED露營燈', 4, 450.00, 25, '高亮度LED露營燈，可充電');

-- 插入訂單資料
INSERT INTO 訂單 (訂單編號, 客戶ID, 付款方式, 運送地址) VALUES
('ORD20240101', 1, '信用卡', '台北市信義區信義路100號'),
('ORD20240102', 2, 'PayPal', '新北市板橋區中山路200號');

-- 插入訂單明細
INSERT INTO 訂單明細 (訂單編號, 產品ID, 購買數量, 實際單價)
VALUES
('ORD20240101', 1, 1, 3500.00),
('ORD20240101', 2, 2, 1200.00),
('ORD20240102', 3, 1, 800.00),
```

```
('ORD20240102', 4, 3, 450.00);
```

常用查詢範例

查詢客戶訂單明細

```
SELECT
    o.訂單編號,
    c.客戶姓名,
    o.訂單日期,
    p.產品名稱,
    cat.類別名稱,
    od.購買數量,
    od.實際單價,
    od.小計
FROM 訂單 o
JOIN 客戶 c ON o.客戶ID = c.客戶ID
JOIN 訂單明細 od ON o.訂單編號 = od.訂單編號
JOIN 產品 p ON od.產品ID = p.產品ID
JOIN 類別 cat ON p.類別ID = cat.類別ID
ORDER BY o.訂單日期 DESC;
```

查詢各類別銷售統計

```
SELECT
    cat.類別名稱,
    COUNT(od.產品ID) as 銷售次數,
    SUM(od.購買數量) as 總銷售量,
    SUM(od.小計) as 總銷售額
FROM 類別 cat
LEFT JOIN 產品 p ON cat.類別ID = p.類別ID
LEFT JOIN 訂單明細 od ON p.產品ID = od.產品ID
GROUP BY cat.類別ID, cat.類別名稱
ORDER BY 總銷售額 DESC;
```

查詢庫存不足產品

```sql
SELECT
    p.產品名稱,
    cat.類別名稱,
    p.庫存量,
    p.單價
FROM 產品 p
JOIN 類別 cat ON p.類別ID = cat.類別ID
WHERE p.庫存量 < 5
ORDER BY p.庫存量 ASC;
```

正規化總結

正規化階段	主要目標	解決問題	達成效果
1NF	原子值	多值欄位	每欄位不可再分割
2NF	完全函數相依	部分相依	屬性完全依賴主鍵
3NF	消除傳遞相依	非主鍵間相依	最佳化資料結構

結論： 透過三個階段的正規化，我們成功建立了一個結構良好、效能優異、易於維護的 MySQL 資料庫系統，完全符合現代資料庫設計的最佳實務。