

# 📘 系統架構設計規格書 (System Design Spec) v2.1 - Grand Unification

版本：v2.1 (Final Executable) 日期：2026-01-07 狀態：準備開發 (Ready for Dev)

## 00. 版本編修歷程 (Version History)

版本	日期	修改人	變更重點說明
v2.1	2026-01-07	AI Architect	<b>新增實體資產層 (Physical Assets Layer)</b>  1. 新增 <code>customers</code> 表：對應 QBO 客戶資料。  2. 新增 <code>products</code> 表：定義產品目錄 (SKU/Model)。  3. 新增 <code>serial_numbers</code> 表：納入庫存與序號追蹤。  4. 定義 CSV 原始資料對應欄位。
v2.0	2026-01-06	AI Architect	<b>大一統架構 (Grand Unification)</b>  1. 確立訂單、出貨、財務三權分立模型。  2. 定義 QBO (財務真理) 與 PostgreSQL (業務真理) 的雙核心架構。  3. 制定 QBO TxnID 連結策略。

## 0. 設計初衷與核心原則 (Design Philosophy)

### 目標 (Goal)

建立一個「訂單、出貨、財務三權分立」的中台系統。短期實現自動化應收帳款 (AR) 監控，減少漏收款風險；長期實現全流程數位化與數據整合。

### 核心原則 (Core Principles)

#### 1. 三權分立模型 (Separation of Powers)

- 訂單 (Orders)：業務承諾 (來源：PandaDoc / Dealer PO)。
- 出貨 (Shipments)：物流操作 (來源：倉儲 / 物流系統)。

- **財務 (Finance)：** 金流結算 (來源：QuickBooks Online)。
2. 單一真實來源 (Single Source of Truth)
- **QuickBooks Online (QBO)** 是財務數據的唯一真理。
  - **PostgreSQL** 是業務邏輯與歷史紀錄的真理。
  - **Google Sheets** 僅作為暫時性的「操作介面 (UI)」，不儲存永久資料。
3. 穩健連結 (Robust Linking)
- 捨棄脆弱的文字比對。
  - 全面採用 QBO TxnID (內部系統碼) 與 LinkedTxn (交易繼承) 機制，確保從「訂單」到「發票」的數位鎖鏈永不斷裂。

1. 系統架構圖 (System Architecture)

採用 混合式架構 (Hybrid Architecture)，結合雲端運算、關聯式資料庫與試算表介面。

- **輸入層 (Input Layer)**
  - **Zapier / Webhook：** 監聽 PandaDoc 合約簽署或 Dealer PO 上傳事件，將原始資料寫入 DB orders 表。
- **核心層 (Core Layer)**
  - **Google Cloud Run：** 運算中心。負責 API 接收、每日排程邏輯、與資料轉換。
  - **PostgreSQL (Supabase)：** 記憶中心。儲存所有訂單、發票、出貨單及其關聯的完整歷史紀錄。
- **同步層 (Sync Layer)**
  - **Cloud Scheduler：** 每日定時 (例如 09:00 AM) 喚醒同步程序。
  - **QBO API：** 進行雙向財務數據同步 (Estimate / Invoice)。
- **表現層 (Presentation Layer)**
  - **Google Sheets：** 內勤人員的操作儀表板，僅顯示 Active/Overdue 的案件。

2. 統一狀態機 (Unified Status Workflow)

系統維護 兩套 平行運作的狀態，互不干擾但互相參照。

A. 訂單生命週期 (Order Lifecycle) - 業務視角

狀態代碼	說明	觸發條件
DRAFT	草稿	報價階段 (Quote)。
CONFIRMED	訂單確立	PandaDoc 簽署完成 或 Dealer PO 上傳成功。
ALLOCATING	配貨中	內勤開始規劃庫存與倉庫分配。

PARTIALLY_SHIPPED	部分出貨	產生第一張 BOL (提單)。
SHIPPED	全數出貨	所有訂單商品皆已出貨。
COMPLETED	結案	交付完成且財務結清。

B. 發票財務狀態 (Invoice Status) - 財務視角

狀態代碼	說明	計算邏輯
OPEN	未結 / 待付款	Balance > 0 且 Today <= DueDate 。
OVERDUE	逾期	Balance > 0 且 Today > DueDate 。
PAID	已結清	Balance = 0 。

3. 資料庫模型 (Database Schema)

A. 訂單主表 ( orders ) - 業務核心


- id (UUID, PK): 系統唯一碼。
- order\_number (String): PO# 或 Quote# (業務識別碼)。
- source (Enum): 'DEALER', 'QUOTE'。
- status (Enum): DRAFT, CONFIRMED... (參考狀態機 A)。
- customer\_info (JSON): 快照客戶當下的聯絡資訊 (可關聯至 customers 表)。
- created\_at (Timestamp): 建立時間。

B. 財務預算表 ( estimates ) - 連接橋樑

- id (UUID, PK): 系統唯一碼。
- qbo\_estimate\_id (String):


SyncKey

QBO 內部的 TxnID 。

- qbo\_doc\_number (String): QBO 顯示編號 (如 EST-1001)。
- order\_id (UUID, FK):  關鍵連結。透過 qbo\_doc\_number 比對 orders.order\_number 自動建立關聯。
- total\_amount (Decimal): 預算總額。

C. 發票表 ( invoices ) - AR 監控核心

- id (UUID, PK): 系統唯一碼。

- **estimate\_id** (UUID, FK):  關鍵連結。指向 estimates 表 (透過 QBO API 的 LinkedTxn 欄位取得)。
- **qbo\_invoice\_id** (String):

### *SyncKey*

QBO 內部的 TxnID (不可變)。

- **qbo\_doc\_number** (String): 發票號碼 (如 INV-1001)。
- **amount** (Decimal): 發票總額。
- **balance** (Decimal): 剩餘未付金額 (每日更新)。
- **due\_date** (Date): 到期日。
- **status** (String): OPEN / OVERDUE / PAID。
- **last\_synced\_at** (Timestamp): 上次同步時間。

### D. 出貨單表 ( shipments ) - 物流紀錄

- **id** (UUID, PK): 系統唯一碼。
- **order\_id** (UUID, FK): 指向 orders 表。
- **tracking\_number** (String): BOL 或貨運單號。
- **shipped\_at** (Date): 出貨日期。
- **items** (JSON): 紀錄本次出貨的商品明細與數量。

### E. 客戶表 ( customers ) - 客戶資料庫

來源: QBO / customers\_rows.csv

- **id** (UUID, PK): 系統唯一碼。
- **qbo\_id** (String, Unique): **[Sync Key]** 對應 CSV 的 qbo\_id，來自 QBO。
- **company\_name** (String): 公司名稱。
- **contact\_email** (String): 聯絡 Email。
- **payment\_terms** (String): 付款條件。
- **rsm\_name** (String): 負責業務 (RSM)。

### F. 產品表 ( products ) - 產品目錄

來源: products\_rows.csv

- **sku** (String, PK): 產品編號 (SKU)，系統內的產品唯一識別。
- **model\_name** (String): 型號名稱。
- **description** (String): 產品描述。
- **unit\_price** (Decimal): 單價。
- **cost** (Decimal): 成本。

- **category** (String): 類別 (如 Finished Goods, Spare Parts)。
- **item\_type** (String): 項目類型 (如 Inventory, Non-inventory)。

#### G. 序號與庫存表 ( `serial_numbers` ) - 資產追蹤

來源: `serial_numbers_rows.csv`

- **serial\_no** (String, PK): 序號 (S/N)。
- **sku** (String, FK): 指向 `products.sku`。
- **status** (String/Enum): 目前狀態 (如 'In Stock', 'Shipped', 'Assigned')。
- **warehouse\_location** (String): 倉庫位置。
- **inbound\_date** (Timestamp): 入庫時間。
- **bol\_number** (String): 關聯的出貨單號 (未來可透過 ETL 轉換關聯至 `shipments.tracking_number`)。
- **po\_sku\_key** (String): 原始資料的關聯鍵值 (用於資料追溯)。
- **created\_at** (Timestamp): 建立時間。

## 4. 自動化邏輯流程 (Automation Logic)

### 流程一：訂單輸入 (Ingestion)

1. **觸發**：Zapier 接收到來源 (PandaDoc/Form) 事件。
2. **動作**：呼叫 Cloud Run API，將資料寫入 PostgreSQL `orders` 表，初始狀態為 `CONFIRMED`。

### 流程二：每日財務同步 (The Daily Sync)

1. **觸發**：Cloud Scheduler 每日早上喚醒 Cloud Run。
2. **QBO 查詢**：透過 API 撈取所有近期變動的 Estimate 與 Invoice。
3. **關聯建構 (Linking)**：
  - 程式讀取 Invoice 的 `LinkedTxn` 屬性，找到來源 Estimate ID。
  - 程式讀取 Estimate 的 `DocNumber`，找到對應的 Order ID。
4. **DB 更新**：使用 Upsert (更新或插入) 邏輯，將最新狀態寫入 PostgreSQL。

### 流程三：Google Sheet 儀表板更新 (UI Update)

1. **讀取 DB**：程式從 PostgreSQL 查詢所有 `status = OVERDUE` 或 `status = OPEN` 的發票。
2. **比對 Sheet**：讀取 Google Sheet 目前內容進行比對。
  - **新單 (Insert)**：若 Sheet 無此單號，於底部新增一行。
  - **舊單 (Update)**：若 Sheet 有此單號，更新 `Days Past Due`，`Balance`，`Last Sync` 等欄位 (保留人工備註)。

- **已結清 (Close)：** 若 DB 狀態已變為 PAID ，將 Sheet 上對應列的 Balance 更新為 0 ，並在 Status 欄位標記 "Paid" (或依需求移至封存頁)。

5. Google Sheet 欄位地圖 (UI Schema)

此表單設計為「內勤人員作業用」，僅列出需關注項目。

欄號	欄位名稱 (Header)	來源 / 邏輯	寫入權限	備註
A	Invoice #	QBO DocNumber	🤖 程式	Unique Key
B	Customer	QBO CustomerRef.name	🤖 程式	顯示客戶名稱
C	Order Source	DB orders.source	🤖 程式	顯示來源 (PO/Quote)
D	Balance	QBO Balance	🤖 程式	未付金額
E	Due Date	QBO DueDate	🤖 程式	到期日
F	Days Past Due	Today - Due Date	🤖 程式	每日計算更新
G	Last Sync	System Timestamp	🤖 程式	上次更新時間
H	Status / Note	人工輸入 / 程式覆寫	👉 協作	人工備註；結清時程式填入