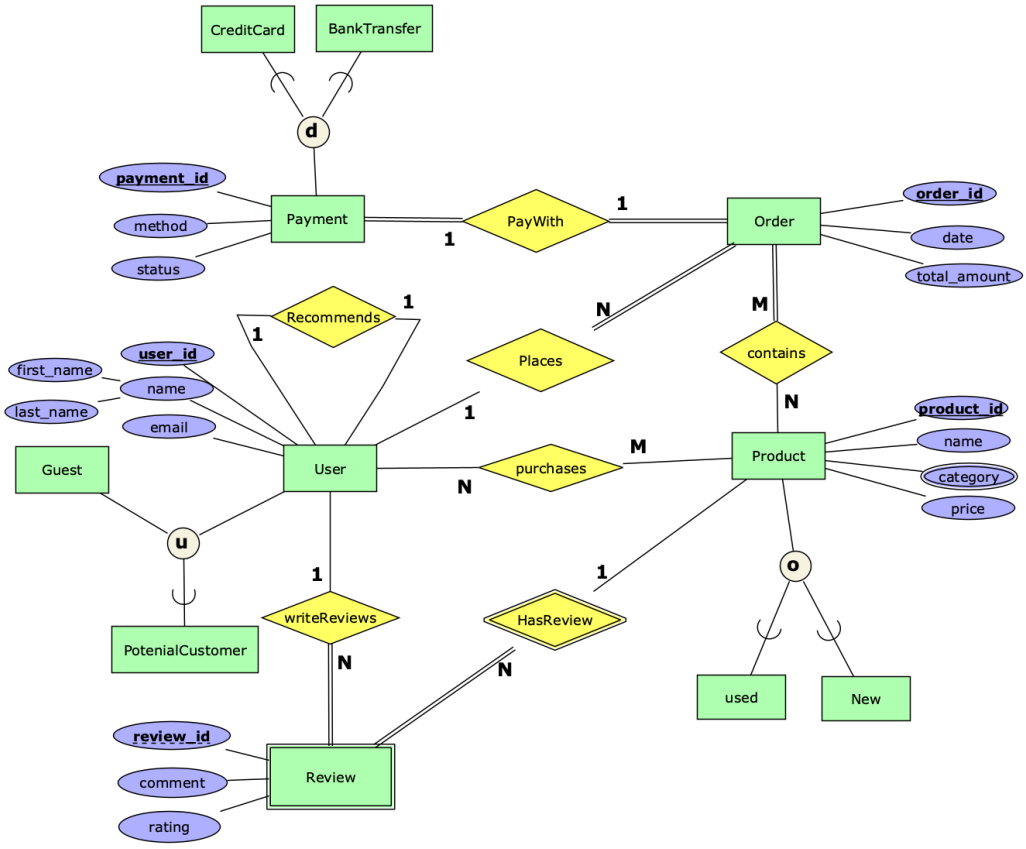
二手家具交易平台資料庫設計說明文件

1. 資料庫主題與用途說明

本資料庫的主題是一個**二手家具交易平台**，其主要用途在於管理平臺上的用戶、商品、訂單、付款和評價等相關資訊。透過此資料庫，平臺營運者可以方便地儲存與查詢每位用戶的基本資料、上架的家具商品資訊、買賣產生的訂單記錄，以及訂單的付款狀態與商品評價。這套資料庫設計確保了交易流程中各環節資料的完整性與一致性，讓平臺能有效地管理二手家具買賣的整體運作。



2. 各個表格的主鍵與欄位說明

本資料庫包含多個資料表（Table），每個資料表及其主要欄位說明如下：

- **User 資料表（使用者）**：儲存註冊用戶的基本資料。主鍵為 user_id。欄位包含用戶的 first_name（名）、last_name（姓）、email（電子郵件）等聯絡資訊。*(此表記錄平臺中註冊會員的身份資訊。)*
- **Guest 資料表（訪客）**：儲存未註冊用戶（訪客）的資料。主鍵為 guest_id。欄位包含 name（訪客姓名）與 email（電子郵件）等必要聯絡資訊。*(此表用於記錄以訪客身份進行購物的潛在顧客資料，例如下訂單時提供的聯絡方式。)*
- **Product 資料表（商品）**：儲存平臺上架的二手家具商品資訊。主鍵為 product_id。欄位包含 name（商品名稱）、price（價格）、category（類別，例如家具種類）等屬性來描述商品。*(商品的新舊狀態會透過特化設計來呈現，詳見後述的特化設計說明。)*
- **Order 資料表（訂單）**：儲存顧客下單的訂單資訊。主鍵為 order_id。欄位包含 date（訂單日期）、total_amount（訂單總金額）、user_id（下單的用戶編號）等資料，用以紀錄每筆訂單由哪位用戶在何時、以何種金額購買了哪些商品。*(若訂單是由未註冊訪客產生，則透過聯集關係來處理，詳見後續「潛在顧客」說明。)*
- **Payment 資料表（付款）**：儲存訂單的付款資訊。主鍵為 payment_id。欄位包含 method（付款方式，例如信用卡或銀行轉帳）、status（付款狀態，如完成、處理中等）、order_id（所對應訂單的編號，為外鍵）等。*(每筆訂單對應一筆付款記錄，此表紀錄了付款的方式與狀態。)*
- **Review 資料表（評價）**：儲存用戶對商品所給的評價記錄。主鍵為 review_id。欄位包含 rating（評分，例如 1~5 星）、comment（評價留言）、user_id（撰寫評價的用戶編號，為外鍵）、product_id（被評價的商品編號，為外鍵）等。*(一則評價對應一個用戶針對一項商品的評論與評分。)*
- **Contains 資料表（訂單明細）**：此表用於表示訂單與商品之間的多對多關係。複合主鍵為 (order_id, product_id) 組合。欄位包含 order_id（訂單編號，外鍵）、product_id（商品編號，外鍵）、quantity（數量，表示該商品在此訂單中購買的數量）等。*(每筆記錄代表某訂單包含某件商品及其數量，用來連結訂單與商品。)*

3. 特殊設計（特化、聯集）與資料結構選擇說明

本資料庫在 E-R 模型中運用了特化（Specialization）及聯集（Union）等進階設計概念。以下說明這些概念在資料表設計中的應用及對應的資料結構：

- **Payment 特化設計（CreditCard 與 BankTransfer）**：在概念模型中，Payment 實體可特化為兩種類型：信用卡付款（CreditCard）與銀行轉帳付款（BankTransfer）。為了在關聯資料庫中實現這個特化，我們在 Payment 資料表中加入了一個欄位 method，採用 **ENUM** 資料類型來儲存付款方式。method 欄位的值被限制為 'CreditCard' 或 'BankTransfer' 兩種，以表示該筆付款記錄的實際類型。我們選擇使用 ENUM 而非為兩種付款方式建立獨立的子表，是因為這樣可以將付款資訊集中在同一資料表中，設計更精簡。同時，如果某些付款方式有專屬欄位（例如信用卡號碼或銀行帳戶資訊），可以將這些欄位加至 Payment 表並允許其為 NULL，以便只對應特定類型時填入。透過 ENUM 控制付款類型有效值，不僅簡化了特化實現，亦確保資料的一致性。
- **Product 重疊特化設計（Used 與 New）**：Product 實體在 EER 模型中被特化為 **「二手 (Used)」** 和 **「全新 (New)」** 兩種子類別，而且這種特化屬於**重疊特化（Overlapping Specialization）**。表示一項商品可以被標記為二手或全新，甚至在模型概念上允許同時屬於兩者（雖然實務上同一商品不太可能同時是新與二手，但模型上不強制其互斥）。在關聯式實作上，我們沒有為 Used 和 New 各自建立獨立資料表，而是透過在 Product 表中使用一個屬性來表示商品的新舊狀態。例如，可新增一個布林欄位 is_used（或用一個類似 condition 的欄位，其值可為 'Used' 或 'New'）來區分商品狀態。如果使用布林欄位，當 is_used = TRUE 表示二手商品，FALSE 表示新品；亦可用 ENUM 或字串欄位直接標註 "Used" 或 "New"。由於採用**重疊特化**，模型上允許 is_used 同時為 TRUE 和商品標註為新品（假設存在另一欄位標示新品），但在業務規則上我們會避免這種情況。因此，此設計提供彈性之餘，也維持了資料結構的簡單性，只需透過欄位值即可區分商品性質。
- **PotentialCustomer 聯集設計（User 和 Guest 組合）**：在概念模型中，我們定義了一個 **PotentialCustomer（潛在顧客）** 的類別，它是一個聯集（Union），由 User 和 Guest 兩種實體組成。這表示無論是註冊用戶還是未註冊的訪客，都被視為潛在顧客，可以在平臺上下訂單。在 EER 圖中，User 和 Guest 透過聯集關係連結到 PotentialCustomer，意味著這兩者的實體集合共同構成了潛在顧客的集合。由於關聯式資料庫中沒有直接的「聯集類別」對應，我們在實作時並未建立獨立的 PotentialCustomer 資料表，而是**利用既有的 User 和 Guest 表來實現**這

一概念。具體而言，在需要參照潛在顧客的地方（例如 Order 訂單記錄的下單者），我們允許透過兩種途徑來連結：倘若下單者為註冊用戶，Order 表的 user_id 外鍵會指向 User 表中的對應紀錄；若下單者為訪客，則我們會有機制記錄其 guest_id（例如在 Order 表中額外紀錄一個 guest_id，當此值有資料時表示訂單由訪客建立）。換言之，一筆訂單可以關聯到 User 或 Guest 的其中一方，以此達成聯集關係的效果。在資料庫應用層級，我們可以透過邏輯控制確保同一筆訂單不會同時關聯兩者，從而正確地表達「潛在顧客」的概念。

4. 關係 (Relationships) 轉換成資料表或外鍵的方式

根據 ER 模型中的實體關係，我們將各種關聯轉換為資料庫中的外鍵設定或關聯表。以下列出主要的關係及其在資料庫實作中的對應方式：

- **User 與 Order 的 1:N 關係**：一位用戶可以對應多筆訂單，每筆訂單只屬於一位下單用戶。在資料表實作上，我們在 Order 資料表中加入一個外鍵欄位 user_id 來參照 User 資料表的主鍵。透過此外鍵聯結，每筆訂單紀錄都指明了是哪位用戶所建立，從而實現了用戶與訂單的一對多關係。
- **Order 與 Product 的 M:N 關係（contains）**：一筆訂單可以包含多項商品，而同一件商品也能出現在多筆不同訂單中，這形成了多對多的關係。為了在關聯式資料庫中表達 M:N 關係，我們建立了前述的 **Contains 資料表**作為訂單與商品的關聯表。Contains 表含有訂單編號 (order_id) 和商品編號 (product_id) 兩個外鍵，分別參照 Order 表和 Product 表的主鍵。同時這兩個外鍵組合作為 Contains 表的複合主鍵，以確保同一訂單與同一商品的組合在此表中不會重複出現。此外，Contains 表還可以包含例如 quantity 等額外欄位來表示訂單中某商品的購買數量。透過 Contains 資料表，我們成功將訂單和商品的多對多關係拆解為兩個一對多關係（Order 對 Contains 以及 Product 對 Contains），符合資料庫正規化設計。
- **User 與 Review 的 1:N 關係**：一位用戶可以留下多筆商品評價，每筆評價僅屬於一位發表的用戶。在資料表設計上，Review 資料表中設置了外鍵 user_id，參照 User 表的主鍵，以指明每則評價的作者是誰。如此實現用戶與評價的一對多關係聯繫，方便日後查詢某用戶發表過的所有評價。
- **Product 與 Review 的 1:N 關係**：一個商品可以收到多筆不同用戶撰寫的評價，而每筆評價只針對單一商品。為了表達這關係，我們在 Review 資料表中加入了另一個外鍵 product_id，參照 Product 表的主鍵。透過此

外鍵，每筆評價紀錄都關聯到被評價的商品，達成商品與評價的一對多對應。在實務應用中，這允許我們輕鬆地統計某商品所收到的所有評價，以及計算平均評分等。

- **User/Guest 與 PotentialCustomer 的關係（聯集對應）**：在 EER 模型中，User 和 Guest 透過聯集關係成為 PotentialCustomer 類別的一部分。這並非傳統意義上的「兩個表之間」關係，而是描述類別繼承/集合的關係。在實作上，如前節所述，我們沒有單獨的 PotentialCustomer 表，因而沒有直接的外鍵來參照它。取而代之的是：User 和 Guest 各自保持自己的主鍵，並分別對應到訂單的建立者。也就是說，一筆訂單的下單者可以是 User 或 Guest。為了支援這點，我們在 Order 表中可以同時具備 user_id 與 guest_id 欄位（或其他等價的設計），用來記錄下單者的身份來源。當訂單由註冊用戶建立時，我們填入其 user_id，而將 guest_id 留空；反之，若訂單由訪客建立，則填入對應的 guest_id，user_id 留空。這種設計透過允許兩種外鍵其一為空的方式，實現了 PotentialCustomer 聯集關係在訂單層面的映射。資料完整性方面，可以加上檢查條件確保同一筆訂單不會同時存在用戶與訪客兩個參照。同時，這也體現了在 ER 模型中 User 和 Guest 同屬 PotentialCustomer 集合的概念：無論來源是哪一種實體，皆可視為有效的下單者。

5. Views 的用途與查詢範例

為了簡化常用查詢並提供易用的數據介面，我們在此資料庫上定義了數個檢視（View）。以下說明兩個主要的檢視及其用途，並提供查詢範例：

- **UserOrders 檢視表**：此檢視將用戶與其訂單資訊預先結合，方便快速查詢用戶的所有訂單明細。UserOrders 是由 User 表與 Order 表透過用戶編號 (User.user_id = Order.user_id) 連接所構成的視圖，提供每筆訂單以及對應的用戶資訊（例如用戶姓名、訂單日期、總金額等）。使用這個檢視，查詢特定用戶的訂單時無需手動撰寫 JOIN 語句。例如，要查詢用戶編號為 5 的所有訂單，可以執行以下 SQL 查詢：
    - SELECT *
    - FROM UserOrders
    - WHERE user_id = 5;

  上述查詢將返回用戶編號為 5 的用戶所建立的所有訂單列表，包括每筆訂單的編號、日期、總金額等欄位（以及該用戶的姓名或其它需要的用戶資訊）。藉由 UserOrders 檢視，我們能快速獲取某用戶的訂單紀錄，提升查詢效率與可讀性。

- ProductReviews 檢視表：此檢視將商品與其相關的評價資訊結合起來，提供快速查詢每項商品評價細節的途徑。ProductReviews 是由 Product 表和 Review 表透過商品編號 (Product.product_id = Review.product_id) 連接而成的視圖，其中包含商品的基本資訊（如名稱、價格等）以及該商品收到的評價內容（如評分、評論文字，以及評價的作者等透過關聯可以取得的資訊）。利用此檢視，用戶可以輕鬆查詢某商品的所有評價，而不需手動撰寫多表 JOIN。舉例來說，若要查詢商品編號為 10 的所有評價，可以使用以下 SQL 語句：
    - SELECT *
    - FROM ProductReviews
    - WHERE product_id = 10;

    執行上述查詢將列出編號為 10 的商品的相關資訊，以及該商品所有評價的細節（例如每則評價的評分與評論內容，以及撰寫該評價的用戶編號）。透過 ProductReviews 檢視，管理者或賣家可以迅速地瀏覽商品的反饋紀錄，以了解商品表現或顧客滿意度。

上述兩個檢視提供了方便的查詢介面，使常見的資料檢索需求（如查詢用戶所有訂單、查詢商品所有評價）更為簡潔明瞭。同時，這些檢視也有助於提高查詢的可讀性，讓使用者或開發人員更容易理解查詢結果所代表的意涵。