資料庫管理（112-1）
期末專案完整報告

D10725003 孫子龍、D09725001 鐘智瑋、D07725004 林文堯
R11722045 李宗霖、B08702068 吳佳芸
第 7 組
2023 年 12 月 20 日
GitHub 專案連結、展示影片連結

1 系統分析

路況是瞬息萬變的，尤其交通事故不僅影響人們的安全，更會對社會交通網路與運輸業者造成深遠影響，在交通事故管理上，即時的資訊更新尤為重要。為降低交通事故發生，提供即時且完整資訊給各方使用者與決策者，成為本專案設計動機。設計理念在建構一個彙整交通事故資訊、天氣狀況、交通流量之視覺化查詢平台。透過此平台，希望實現對道路使用者、運輸業者及主管機關提供實用、即時且全面的交通資訊。為提升交通安全與效率，除提供基本事故資料外，亦結合氣象、道路設施、交通流量等多維度資訊，協助道路使用者、運輸業者及主管機關作為參考並藉以擬定相關實施措施。

資訊來源主要透過道路使用者即時回報，回傳資訊包含事故發生時間與地點、位於國道位置、附近雨量、風速及氣溫，及各類車流量與車速等。綜合這些資訊，形成一個所謂「事故熱點」。此外，其他道路使用者可以檢視所有已回報「事故熱點」及其詳細資訊，針對將經過該事故熱點路段之道路使用者，請其回覆熱點資訊，以確保該熱點確實存在並對交通造成影響，排除誤報或已解決不再影響交通的情況。對於即將上路之道路使用者，本平臺亦提供事故熱點作為行程規劃參考。主管機關可即刻收到事故回報，迅速調度相關車輛與資源，進行事故處理及疏運，抑或用於收集特定期間的交通事故資訊，作為研擬改善交通政策依據。運輸業者則可根據「事故熱點」頻率，提前計畫好應注意路段，規劃最佳與最安全的客運路線，並提供乘客最安全及最省時的運輸服務。

1.1 系統功能

1.1.1 系統相關設定

根據不同功能及掌控權限，本平臺設計使用者角色分為一般道路使用者 User、運輸業者與主管機關 Viewer 及管理者 Admin 三類。作為一般道路使用者 (User)，可回報與確認事故熱點，操作簡單，僅需按下對應按鈕便可完成資訊送出。系統自動匯入回報地點之經緯度及國道里程，並與車流與氣候資料庫對接，形成完整的事故熱點資料。此外，User 亦可查看 7 天內全體道路使用者回報與確認次數。運輸業者與主管機關 (Viewer) 則有權查詢任意時間段所有路段事故熱點，但不能新增、更動、刪除或確認

事故熱點資料。Admin 作為管理角色，負責維護資料庫資料有效性，並有權查詢所有使用者活動，包括 User 回報與確認的事故熱點歷史，以及 Viewer 曾查詢的路段與時間範疇，以此資料進行分析來優化平台運營。

1.1.2 供使用者操作功能

在本系統中，使用者可以執行以下功能：

1. **回報事故熱點：**觀察到事故發生後，按下專用按鈕並選擇是否屬於重大事故，即可完成資料提交。此新增事故熱點將儲存於資料庫並於地圖上呈現。
2. **查閱最近 7 天事故熱點資料：**使用者可查詢 7 天內所有發生過的事故熱點及其詳細資料。
3. **查閱當前事故熱點：**使用者得以於地圖上觸碰事故熱點，進一步獲取詳細資訊。

1.1.3 供管理者操作功能

在本系統中，管理者可以執行以下功能：

1. **事故熱點資料管理：**具備刪除重複或濫報事故熱點之權限。
2. **使用者管理：**管理者得以封鎖濫用平台之使用者及查閱者。
3. **交通與氣象資料庫管理：**找出並修正資料庫內的異常值。
4. **查詢使用者活動紀錄：**得以查詢所有使用者回報或確認的事故熱點歷史資料，以及查閱者的查詢紀錄。

1.1.4 供查閱者操作功能

在本系統中，查閱者可以執行以下功能：

1. **查詢事故熱點歷史資料：**查閱者得以查詢任意時間段內事故熱點之歷史資料，並可詳閱其內容。

2 系統設計

2.1 ER Diagram

圖 1 是「國道『事故熱點』回報平台」的 ER Diagram，本專案的 ER Diagram（Entity-Relationship Diagram）由五個主要實體（Entity）組成，分別為 `WeatherObs`、

GeoLocatingStaticInfo、eTagObs、Hotspot 及 User，User 和 Hotspot 之間有 Report、Confirm 與 Query 之 Relationship。這些實體透過精確的設計，以及彼此間的關聯，組成「國道『事故熱點』回報平台」的運作。

1. **WeatherObs：**此實體專門用於儲存氣象觀測數據，包括氣象站 ID、觀測時間、緯度、經度、地點名稱、風速、溫度和降雨量等。它的主要目的是為提供即時、準確的氣候數據，以針對事故熱點進行詳細分析。

2. **GeoLocatingStaticInfo：**此實體包含國道位置的靜態資訊，例如道路 ID、位置座標、里程、最近的 ETag 測量站 ID 及最近的氣象站 ID。設計目的是為提供準確地理位置信息，協助用戶與系統確定事故具體位置。

3. **eTagObs：**此實體專門用於記錄 ETag 測量站交通數據，例如: ETag 測量路段起點站 ID、日期、車輛類型、平均車速及交通量。此些數據有助於分析事故發生時的交通狀況。

4. **Hotspot：**此實體核心聚焦於事故熱點的具體資訊，包括熱點 ID、用戶 ID、是否為重大事故、時間戳記、位置座標、道路 ID、行車方向、最近的里程標示及事故確認次數。這個設計能夠讓用戶快速回報事故，並為其他用戶提供即時事故訊息。

5. **User：**此實體包含用戶個人資訊，如用戶 ID、用戶名、密碼、用戶類型及權限狀態。這個設計允許對用戶不同角色進行管理及區分，如一般用戶、查閱者及管理者。

Users 和 Hotspot 之間的 Relationship：

1. **Report：**用於記錄用戶對事故資料的確認回報行為，包括用戶 ID、用戶位置(經緯度)、回報時間。

2. **Confirm：**用於記錄用戶對事故熱點的確認行為，包括用戶 ID、熱點 ID 與確認時間。此設計有助於追蹤事故確認次數與狀態。

3. **Query：**用於記錄用戶對事故資料的查詢行為，包括用戶 ID、查詢執行時間、起始時間與結束時間。此設計方便用戶查閱事故歷史資料。

在設計此 ER Diagram 時，即考量這些實體之間的關聯性確實保持資料完整性與一致性。例如，Hotspot 實體透過外鍵與 GeoLocatingStaticInfo 及 User 實體相連接，從而關聯事故熱點地理位置與回報用戶資料。這種設計使得當一位用戶回報一個事故熱點時，系統能夠自動關聯及提取相應的地理位置與氣象數據，以及用戶的個人資訊。

我們可以舉例說明一個使用場景：一位駕駛員在國道上發現事故後，通過平台回報事故熱點。系統自動記錄其地理位置、時間及相關交通和氣象數據；其他用戶亦可通過平台查看此事故熱點詳細資訊，包括事故位置、發生時間和當時的交通狀況，並

根據這些資訊調整自己的行車路線；同時，管理者也可根據收集到的數據分析事故頻發的原因與趨勢，以指導未來交通安全政策制定。此 ER Diagram 的設計形成一個有效支援「國道『事故熱點』回報平台」運作的資料庫架構。

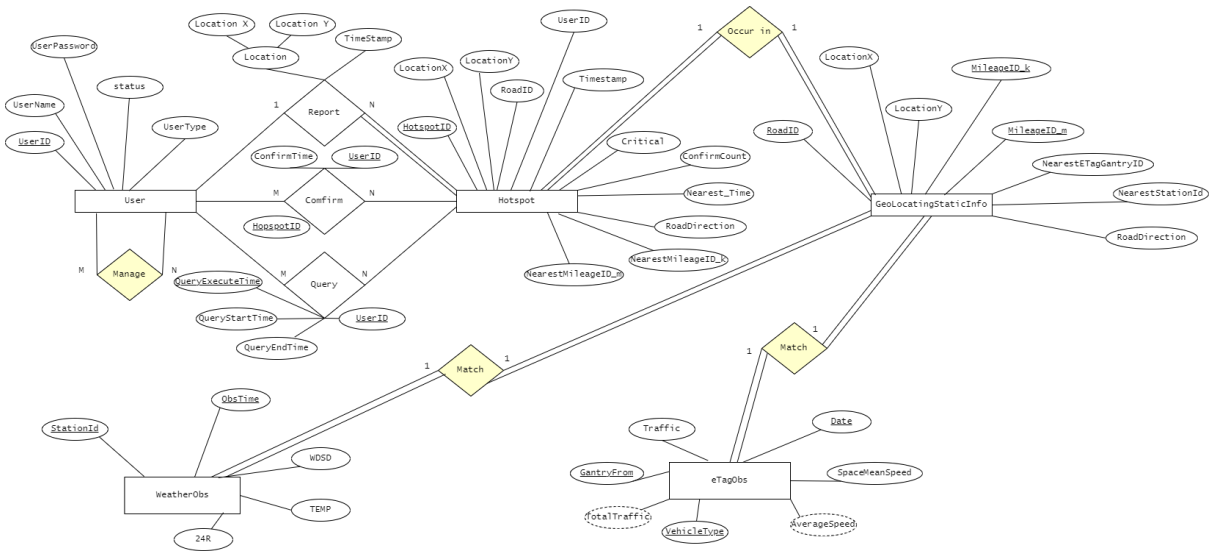

圖 1: 「國道『事故熱點』回報平台」的 ER Diagram

2.2 Relational Database Schema Diagram

我們可以將圖 1 的 ER Diagram 轉換成圖 2 的 Relational Database Schema Diagram，本專案一共有七個關聯（relation），分別為 `User`、`Confirm`、`Query`、`Hotspot`、`GeoLocatingStaticInfo`、`eTagObs` 及 `WeatherObs`。

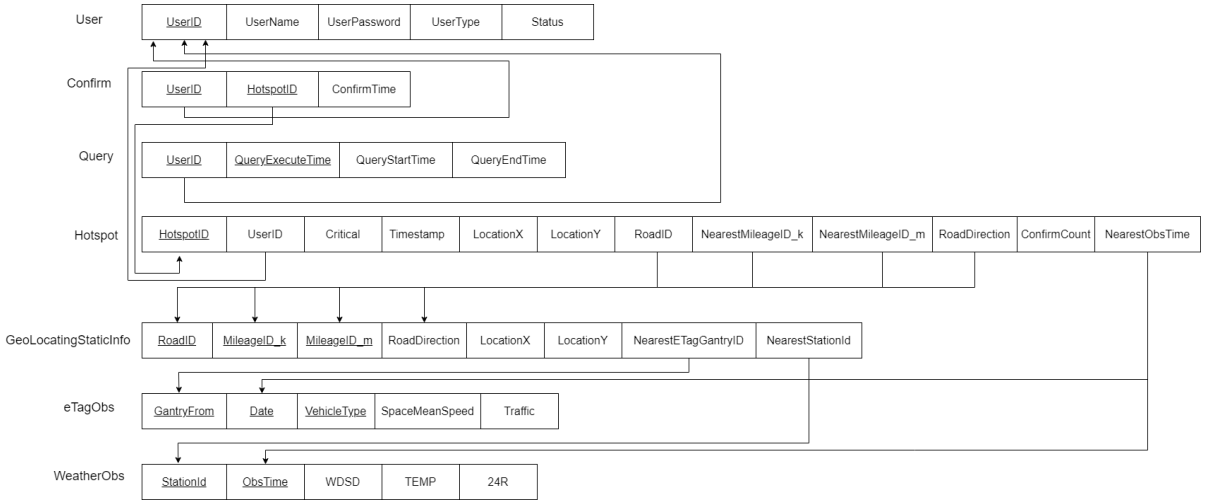

圖 2: 「國道『事故熱點』回報平台」的 Relational Database Schema Diagram

在本專案的關聯性模式（Relational Schema）中，透過主鍵（Primary Keys, PK）與外鍵（Foreign Keys, FK）的設計，確立各 table 間的關聯。此設計使得每個實體皆擁有獨特識別碼，並透過外鍵與其他 table 建立聯繫，從而保證資料完整性與一致性。

1. WeatherObs 實體，以 StationId 及 ObsTime 組成主鍵，確保每筆氣象觀測數據的唯一性及可追溯性。這些數據包括風速、溫度及 24 小時降雨量等，對於進行氣候相關分析或事故影響評估至關重要。

2. GeoLocatingStaticInfo 實體，其主鍵為 RoadID、MileageID_k、MileageID_m 和 RoadDirection，與 Hotspot 結合後，透過 NearestETagGantryID、Hotspot 的 NearestObsTime 與 eTagObs 連結，並透過NearestStationID、Hotspot 的 NearestObsTime 和 WeatherObs 實體相連結，提供精確國道位置資訊，將相關交通與氣象數據相互關聯，對分析特定路段交通狀況和天氣影響具重要價值。

3. eTagObs 實體，主鍵為 GantryFrom 和 Date，記錄每段起點為 ETag 測量站之路段的交通數據，包括日期、起點閘道、車輛類型、平均車速及交通流量。這些數據可用於分析特定時間與地點交通流量和車速，對交通規劃與事故風險評估具重要貢獻。

4. Hotspot 實體，主鍵為 HotspotID，透過外鍵 UserID 與 User 實體連接，並透過 RoadID、NearestMileageID_k、NearestMileageID_m 和 RoadDirection 與 GeoLocatingStaticInfo 實體相連結。這使得每個事故熱點都能夠與特定用戶及地理位置相互關聯，對事故原因進行深入分析，並為防範未來事故提供有利數據。

5. User 實體，主鍵為 UserID，確保每個用戶的資料獨立且完整，用於管理用戶權限並維護用戶個資安全。

6. Confirm 和 Query，分別與 User 和 Hotspot 實體關聯，記錄用戶對事故熱點的確認行為及事故資料的查詢行為。這些實體使得 SQL 查詢能夠追蹤與分析用戶行為，從而改進平台功能與用戶體驗。

此關聯性模式通過級聯刪除/更新操作（Cascade Delete/Update）維護數據參照完整性。例如，當 Hotspot 實體中某個事故熱點被刪除時，相關確認記錄（Confirm）也會自動被刪除，確保資料一致性。應用實例方面，交通管理部門可以利用這些關聯數據來分析特定時段和地點的事故發生頻率，進而指導交通安全政策的制定。運輸業者也可以通過查詢 Hotspot 實體中的數據獲取特定路段的事故信息，調整行車路線或時刻表，提高運營效率和乘客安全。為「國道『事故熱點』回報平台」提供了堅實的數據支持，使其能夠高效地收集、處理和分析事故數據。

2.3 Data Dictionary

國道「事故熱點」回報平台的資料表共有七個，如圖 2 所示，各個資料表的欄位相關資訊依序呈現，如下列表 1 到表 7。

Column Name	Meaning	Data Type	Key	Constraint	Domain
StationId	測站編號	INT	PK	Not Null	
ObsTime	觀測資料時間	TIMESTAMP	PK	Not Null	
WDSD	風速，單位公尺/秒	FLOAT		Not Null	
TEMP	溫度，單位攝氏	FLOAT		Not Null	
24R	日累積雨量，單位公釐	FLOAT		Not Null	

表 1: WeatherObs 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
RoadID	道路代碼	VARCHAR(6)	PK	Not Null	{000010, 000030, 000050}
LocationX	坐標 X (經度)	FLOAT		Not Null	
LocationY	坐標 Y (緯度)	FLOAT		Not Null	
MileageID_k	里程代碼（公里處）	INT	PK	Not Null	
MileageID_m	里程代碼（公尺處）	INT	PK	Not Null	
NearestETagGantryID	eTag 偵測站代碼	VARCHAR(8)	FK: eTagObs(GantryFrom)	Not Null	
NearestStationId	測站編號	VARCHAR(8)	FK: WeatherObs(StationId)	Not Null	
RoadDirection	道路方向	CHAR(1)	PK	Not Null	{N, S}
Referential triggers	On Delete	On Update			
NearestETagGantryID: eTagObs(GantryFrom)	Cascade	Cascade			
NearestStationId: WeatherObs(StationId)	Cascade	Cascade			

表 2: GeoLocatingStaticInfo 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
Date	日期及時間	TIMESTAMP	PK		
GantryFrom	偵測路段起點 ETag 測量站編號	VARCHAR(6)	PK		
VehicleType	車種代碼	INT		Not Null	{5, 31, 32, 41, 42}
SpaceMeanSpeed	平均車速 (指定車種下), 單位:KM/Hr, -99= 資料異常	INT		Not Null	
Traffic	運輸量	INT		Not Null	

表 3: eTagObs 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
HotspotID	事故熱點代碼	VARCHAR(10)	PK	Not Null	
UserID	用戶代碼	VARCHAR(5)	FK: User(UserID)	Not Null	
Critcial	是否嚴重事故	CHAR(1)		Not Null	{Y, N}
Timestamp	發生時間	TIMESTAMP		Not Null	
NearestObsTime	最近發生時間（每十分鐘為一單位）	TIMESTAMP	FK: WeatherObs(ObsTime), eTagObs(Date)	Not Null	
LocationX	坐標 X (經度)	FLOAT		Not Null	
LocationY	坐標 Y (緯度)	FLOAT		Not Null	
RoadID	道路代碼	VARCHAR(6)	FK: GeoLocatingStaticInfo(RoadID)	Not Null	{000010, 000030, 000050}
RoadDirection	行車方向	VARCHAR(1)	FK: GeoLocatingStaticInfo(RoadDirection)	Not Null	{N, S}
NearestMileageID_k	最近之里程代碼（公尺處）	INT	FK: GeoLocatingStaticInfo(MileageID_k)	Not Null	
NearestMileageID_m	最近之里程代碼（公里處）	INT	FK: GeoLocatingStaticInfo(MileageID_m)	Not Null	
ConfirmCount	確認次數	INT		Not Null	
Referential triggers	On Delete	On Update			
UserID: User(UserID)	Cascade	Cascade			
RoadID: GeoLocatingStaticInfo(RoadID)	Cascade	Cascade			
NearestObsTime: WeatherObs(ObsTime), eTagObs(Date)	Cascade	Cascade			
NearestMileageID_k: GeoLocatingStaticInfo(MileageID_k)	Cascade	Cascade			
NearestMileageID_m: GeoLocatingStaticInfo(MileageID_m)	Cascade	Cascade			
RoadDirection: GeoLocatingStaticInfo(RoadDirection)	Cascade	Cascade			

表 4: Hotspot 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserID	用戶代碼	VARCHAR(5)	PK	Not Null	
UserName	用戶姓名	VARCHAR(20)		Not Null	
UserPassword	用戶密碼	VARCHAR(10)		Not Null	
UserType	用戶種類	CHAR(1)		Not Null	{A, V, U}
status	狀態	CHAR(10)			

表 5: Users 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserID	用戶代碼	VARCHAR(5)	PK, FK: User(UserID)	Not Null	
HotspotID	事故熱點代碼	VARCHAR(10)	PK, FK: Hotspot(HotspotID)	Not Null	
ConfirmTime	事故確認時間	TIMESTAMP		Not Null	
Referential triggers	On Delete	On Update			
UserID: User(UserID)	Cascade	Cascade			
HotspotID: Hotspot(HotspotID)	Cascade	Cascade			

表 6: Confirm 表格資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserID	用戶代碼	VARCHAR(5)	PK, FK: User(UserID)	Not Null	
QueryExecuteTime	執行查詢時間	TIMESTAMP	PK	Not Null	
QueryStartTime	事故熱點時間查詢範圍起始時間	TIMESTAMP		Not Null	
QueryEndTime	事故熱點時間查詢範圍結束時間	TIMESTAMP		Not Null	
Referential triggers	On Delete	On Update			
UserID: User(UserID)	Cascade	Cascade			

表 7: Query 表格資訊

2.4 正規化分析

當設計關聯式資料庫時，我們可以檢視資料庫綱目（database schema）是否滿足正規化（normalization）條件，因此我們將依序從第一正規式（1NF）到第四正規式（4NF）來說明「國道『事故熱點』回報平台」的關聯是如何滿足這些規則。

1. **第一正規式 (1NF):** 每個關聯都滿足 1NF ，因為所有欄位都沒有任何複合或多值屬性。我們在任何可能發生多值欄位的情況都做了正規化，譬如 `Confirm` 中不會有同一事故熱點有兩個以上的 User 進行確認。

2. **第二正規式 (2NF):** 2NF 要求在滿足 1NF 的基礎上，非主鍵屬性必須完全依賴於主鍵，這代表我們的實體沒有部分相依性（一個屬性僅依賴於複合主鍵的一部分）。我們特別設計了 `GeoLocatingStaticInfo` 實體作爲串聯 `Hotspot` 和 `WeatherObs` 及 `eTagObs` 的手段，就是爲了避免 `Hotspot` 資料出現部分相依性。

3. **第三正規式 (3NF):** 3NF 要求沒有傳遞依賴，即一個非主鍵屬性不應依賴於另一個非主鍵屬性。經過檢查後，我們確認了所有非主鍵屬性都只依賴主鍵，所以所有實體都符合 3NF。但我們包含經緯度資料的實體都沒有符合 BCNF，意即衆多主鍵中的某個欄位都會相依於其他非主鍵的欄位，譬如：GeoLocatingStaticInfo 中的 MileageID_k 和 MileageID_m 會相依於經緯度（也就是 LocationX 和 LocationY），但考慮到空間資料的複雜性，爲了精簡資料表數量及關聯，因此我們只有做到 3NF 正規化。但其他單純記錄使用者資料的實體，譬如：Users、Confirm 及 Query 皆有符合 BCNF，每個主鍵會不會相依於其他非主鍵的欄位。

4. **第四正規式 (4NF):** 4NF 專門處理多值依賴，要求我們的資料庫中不存在非主鍵屬性的多值依賴。因我們包含經緯度資料的實體只有做到 3NF，因此沒有符合這個規則。但是，Users、Confirm 及 Query 皆有符合 4NF，每個主鍵會不會相依於其他非主鍵的欄位。

3 系統實作

3.1 資料庫建置方式及資料來源說明

本專案採用部分真實資料作為實作依據，力求接近實際運作情形，所使用資料主要來自政府公開資料平臺，來源如下：

1. **國道里程資料：**提供由道路里程之地理編碼 (定位) 服務。
2. **國道交通量資料：**來自交通部高速公路局之交通資料庫。
3. **氣候環境資料：**取自中央氣象局自動氣象站觀測資料彙整。

GeoLocatingStaticInfo 裡存有道路代碼、里程公里處及公尺處、經緯度、行車方向、最近觀測時間、最近氣象站及 ETag 測量站資訊。首先，我們篩選出三條國道作為 User 可回報位置範圍，分別為國道一號、國道三號及國道五號。為控制地理資料筆數，將國道里程最小單位設為 100 公尺，意即三條國道路長加總為 858.3 公里的情況下，將被劃分成 8,583 個座標點，並存入 GeoLocatingStaticInfo。當 User 在回報事故熱點時，除紀錄 User 所在位置準確經緯度，也透過「交通部運輸資料流通服務平臺」(Transport Data eXchange ,TDX) 找出此地點在國道上對應的里程座標點，隨著行車方向存入 Hotspot。

關於 EtagObs、WeatherObs 資料表內的資料建置方式，我們首先利用政府公開資料平臺收集 2023 年 9 月 1 日到 9 月 7 日全台交通量與氣候資料，分別為 2,066,400 筆交通資料及 15,234 筆氣象觀測資料。交通資料涵蓋三條國道上共 335 台 ETag 測量站的代碼和經緯度，以及其在 7 天內每 10 分鐘偵測到五種車型（小客車、大客車、計程車、大型車、聯結車）的平均車速和車輛數。氣候資料包含全台 31 個局屬氣象站的站名和經緯度，以及其在 7 天內每 10 分鐘偵測的降雨量、平均風速及氣溫。

我們事先將 GeoLocatingStaticInfo 上的每一里程計算出距離最近的 ETag 測量站和局屬氣象站，並分別存入 NearestETag 和 NearestStationID。另外，在 User 在回報事故熱點時，除紀錄回報準確時間，也往前追溯最小單位為 10 分鐘的最近發生時間（例如：下午 3 點 29 分追溯成下午 3 點 20 分、早上 9 點 3 分追溯成早上 9 點），以配合交通量和氣象站每 10 分鐘收集一次觀測資料的時間欄位。如此，系統將可透過配對的國道里程及標準化時間，針對每個事故熱點帶入空間與時間同時對應的交通量和氣候資料。

舉例說明：假設 User 於 2023 年 9 月 1 日中午 12 點 58 分在北上行車途中回報事故熱點，經緯度為 24°19'51.1"N，120°42'58.0"E。系統將透過上述一連串的標準化，最後轉譯成「User 於 2023 年 9 月 1 日中午 12 點 50 分在國道一號北上 158.3 公里處發生一個事故熱點」，並帶入最近 ETag 測量站（代號為「01F1572N」）的交通資料，以及最近局屬氣象站（站名為「臺中」）的氣象資料。

以下針對國道事故將使用程式模擬實際世界的情形，生成 Hotspot、User、Confirm 與 Query 四個模擬資料集，分別描述國道事故發生紀錄、所有 User 的 id 及個人資訊、User 針對國道事故所執行之確認行為與次數、所有 User 查詢每個時間內的國道事故等情形。Hotspot 資料集共有 100 筆資料，存有國道事故編號、使用者編號、是否為重大事故、事故發生時間、發生時間最近之時間 (每十分鐘計)、經度、緯度、道路代碼、行車方向、最近之里程公里數、最近之里程公尺數、事故確認次數等欄位資訊。重大事故參考「道安資訊查詢網」上之統計數據，隨機抽取 0.11 - 0.15 範圍內的數字作為依據。事故發生時間將於 2023 年 9 月 1 日至 9 月 7 日之間平均分配隨機抽取的時間點，並以國道一號、三號、五號上的經緯度座標點作為後續位置相關資料的模擬。

User 資料集共有 300 筆資料，存有使用者編號、名字、密碼、使用者身分、存取狀態等資訊，將本專案小組的五名成員列為最高權限管理者 (A)、高速公路局與 15 家客運公司作為檢視者身分 (V)、其餘之用戶皆為單純的使用者身分 (U)。而在使用狀態中，拒絕五位使用者的存取權限。Confirm 資料集共有 353 筆資料，存有使用者編號、國道事故編號、確認時間等資訊，隨機生成每個國道事故 0 至 8 筆的確認紀錄，並以一個右偏常態分布加以描述九個數值的機率分布，並在每個國道事故發生後的一段時間內進行使用者確認，以非重大事故在事故發生後兩小時之內，而重大事故則是在事故發生後六小時之內。Query 資料集共有 1,017 筆資料，存有使用者編號、執行查詢之時間、查詢起始時間、查詢終止時間等資訊，執行查詢之時間將於研究期間之起始日至終止日的後五日內隨機設定，查詢起始與終止時間則為使用者查詢國道事故的起迄時間點，以半小時為單位。

3.2 重要功能及對應的 SQL 指令

於 1.1 節中已介紹 User、Viewer 及 Admin 的功能。在第 3.2.1 和第 3.2.2 小節中將列出 User 及 Admin 附帶特定情境下，完成這些功能所使用的（一或數個）SQL 指令。此外，在第 3.2.3 小節中，我們也列出特定 Viewer 功能的 SQL 指令。

3.2.1 給 User 的功能

1. 「回報事故熱點」：假設情境為「使用者代號 UserID 「U0028」的 ReneeWu 想要回報尚未被其他使用者回報的事故熱點，他觀察到該事故為重大事故，因此在使用者介面勾選重大事故、車道編號、及該車道為南下或北上車道，另外，系統會攫取使用者的經緯度位置，形成一筆 Hotspot 資料。

```
SET @currentTime = now();
SET @nearestTime = (SELECT ROUND(MINUTE(@currentTime) /
   10) * 10);
Insert Into Hotspot (UserID, Critical, Timestamp,
   NearestObsTime, LocationX, LocationY, RoadID,
   RoadDirection, NearestMileageID_k, NearestMileageID_m,
    ConfirmCount)
Values ('U0028', 'Y', @currentTime, @nearestTime,
   121.7318, 25.1188, 10, 'S', 0, 100, 0);
```

Listing 1: 回報事故熱點 SQL 指令

2. 確認由其他 User 提出的「事故熱點」：假設情境為 UserID 為「U0028」的 ReneeWu 目前身處 HotspotID 為「HS000006」的事故熱點附近，ReneeWu 看到事故尚未被排除而按下「事故尚未排除」之按鈕，此時」Confirm 會新增一筆資料、Hotspot 的 confirmCount 加 1。

```
INSERT INTO Confirm
VALUES('U0028', 'HS_000006', now());
UPDATE Hotspot
SET ConfirmCount = ConfirmCount + 1
WHERE HotspotID = 'HS_000006';
```

Listing 2: 確認由其他 User 提出的「事故熱點」SQL 指令

3. 檢視 24 小時內的「事故熱點」資料：假設情境為 UserID 為「U0028」的 ReneeWu 想要檢視過去 8 小時內的事故熱點資料，包含：嚴重性、發生時間、發生地點經緯度、事故被確認次數、事故發生時的風速、溫度、雨量、交通量等資訊，當 ReneeWu 按下查詢按鈕時，系統會攫取查詢的時間，並將資料呈現在使用者頁面；此外，ReneeWu 的查詢也會在 Query 中新增一筆紀錄。

```
SELECT hg.Critical, hg.Timestamp, hg.LocationX, hg.
   LocationY, hg.ConfirmCount, w.WDSD, w.TEMP, w.24R, AVG(
   e.SpaceMeanSpeed) AS AverageSpeed, SUM(e.Traffic) AS
   TotalTraffic
FROM
(    SELECT *
```

```
    FROM Hotspot AS h
    LEFT JOIN GeoLocatingStaticInfo AS g
        ON  h.RoadID = g.RoadID
        AND h.NearestMileageID_k = g.MileageID_k
        AND h.NearestMileageID_m = g.MileageID_m
        AND h.RoadDirection = g.RoadDirection
) AS hg
    LEFT JOIN WeatherObs AS w ON
        hg.StationID = w.StationID
        AND hg.NearestObsTime = w.ObsTime
    LEFT JOIN ETagObs AS e ON
        hg.NearestETagGantryID = e.GantryFrom
        AND hg.NearestObsTime = e.Date
GROUP BY e.GantryFrom
WHERE Timestamp >= now() - INTERVAL '8 HOUR';

INSERT INTO Query
VALUES('U0028', now(), now() - INTERVAL '8 HOUR', now());
```

Listing 3: 檢視 24 小時內的「事故熱點」資料 SQL 指令

3.2.2 給 Admin 的功能

1. 查詢所有用戶及其確認的「事故熱點」數量：統計每個用戶確認的「事故熱點」數量。

```
SELECT User.UserID, UserName, COUNT(UserID) AS
   ConfirmedHotspots
  FROM User
  LEFT JOIN Confirm ON User.UserID = Confirm.UserID
  GROUP BY User.UserID;
```

Listing 4: 統計每個用戶確認「事故熱點」數量 SQL 指令

2. 更新特定「事故熱點」狀態：針對特定的「事故熱點」代碼，更新其是否為嚴重事故的狀態。

```
UPDATE Hotspot
  SET Critcial = 'Y'
  WHERE HotspotID = 'HS_000101';
```

Listing 5: 更新特定「事故熱點」狀態 SQL 指令

3. 刪除長時間未活動用戶：刪除最後查詢時間超過特定期限用戶。

```
DELETE FROM User
  WHERE UserID NOT IN (
    SELECT UserID
    FROM Query
    WHERE QueryExecuteTime > DATE_SUB(CURDATE(), INTERVAL
       1 YEAR)
  );
```

Listing 6: 刪除長時間未活動用戶 SQL 指令

4. 查詢特定時間範圍內「事故熱點」報告：查詢在特定時間範圍內報告的所有「事故熱點」。

```
SELECT *
  FROM Hotspot
  WHERE Timestamp BETWEEN '2023-09-01' AND '2023-09-07';
```

Listing 7: 查詢特定時間範圍內「事故熱點」報告 SQL 指令

5. 創建用戶活動匯總報告：匯總用戶在特定時間範圍內查詢次數與確認事故熱點數量。

```
SELECT
    User.UserID,
    UserName,
    COUNT(DISTINCT Query.QueryExecuteTime) AS TotalQueries
       ,
    COUNT(DISTINCT Confirm.HotspotID) AS
       TotalConfirmedHotspots
  FROM User
  LEFT JOIN Query ON User.UserID = Query.UserID
  LEFT JOIN Confirm ON User.UserID = Confirm.UserID
  WHERE Query.QueryExecuteTime BETWEEN '2023-09-01' AND '
     2023-09-07'
  GROUP BY User.UserID;
```

Listing 8: 創建用戶活動匯總報告 SQL 指令

3.2.3 給 Viewer 的功能

1. 找出事故熱點發生前 10 名的車輛種類及當時的降雨量

```sql
SELECT vehicletype, 24R
FROM (
    SELECT vehicletype, 24R, ROW_NUMBER() OVER(ORDER BY
       count DESC) as row_num
    FROM (
        SELECT vehicletype, 24R, COUNT(*) as count
        FROM (
            SELECT vehicletype, 24R FROM weatherobs
            UNION ALL
            SELECT vehicletype, 24R FROM hotspot
        ) combined_tables
        WHERE vehicletype IS NOT NULL
        GROUP BY vehicletype, 24R
    ) counts
) ranked_counts
WHERE row_num <= 10;
```

Listing 9: 找出事故熱點發生前 10 名的車輛種類及當時的降雨量

2. 在 GeoLocatingStaticInfo 及 eTagObs 資料表中找出發生嚴重事故的前三名路段及其平均速度

```sql
SELECT roaddirection, spacemeanspeed
FROM (
    SELECT roaddirection, spacemeanspeed, ROW_NUMBER()
       OVER(ORDER BY count DESC) as row_num
    FROM (
        SELECT roaddirection, spacemeanspeed, COUNT(*) as
           count
        FROM (
            SELECT roaddirection, spacemeanspeed FROM
               GeoLocatingStaticInfo
            WHERE critical = 'Y'
            UNION ALL
            SELECT roaddirection, spacemeanspeed FROM
               eTagObs
            WHERE critical = 'Y'
        ) combined_tables
        WHERE roaddirection IS NOT NULL
        GROUP BY roaddirection, spacemeanspeed
    ) counts
) ranked_counts
```

```
WHERE row_num <= 3;
```

Listing 10: 發生嚴重事故的前三名路段及其平均速度

3.3 SQL 指令效能優化與索引建立分析

eTagObs 是資料量最大的資料表，而系統最常做的操作應為查詢事故熱點，因此我們考慮對 eTagObs 與 Hotspot 和 GeoLocatingStaticInfo 進行 JOIN 的欄位 GantryFrom 和 Date 建立索引。

以下是查詢事故熱點與相關 eTag 資訊的 SQL 查詢指令：

```
SELECT *
FROM
(
        SELECT *
        FROM Hotspot AS h
        JOIN GeoLocatingStaticinfo AS g
          ON h.RoadID = g.RoadID
          AND h.NearestMileageid_K = g.MileageID_K
          AND h.NearestMileageid_M = g.MileageID_M
          AND h.RoadDirection = g.RoadDirection
) AS hg
        JOIN EtagObs AS e
          ON hg.NearestEtagGantryID = e.GantryFrom
          AND hg.NearestObsTime = e.Date
```

Listing 11: 查詢事故熱點與相關 eTag 資訊

在尚未建立索引前，4 次查詢的執行時間分別為 412、741、472、576 毫秒。

接下來，我們在 Hotspot 與 GeoLocatingStaticInfo 欄位上創建索引，並再次執行相同的查詢，此時 4 次查詢的執行時間縮短為 153、171、172、159 毫秒，相較尚未建立索引前省了一些查詢時間。

```
CREATE INDEX idx_GaFr ON eTagObs(GantryFrom);
CREATE INDEX idx_Date ON eTagObs(Date);
```

Listing 12: 建立 index

3.4 交易管理

在給 Admin 功能中交通氣象資料管理的部分，Admin 必須先勾選要更改的內容，再點選修改圖示才能進行資料修改，資料修改完後要再按下確認按鈕才算完成整個修

改流程。在寫入過程中，如果出現違反欄位的資料型別等情況，修改動作會立即停止，或當使用者離開修改頁面，系統將 ROLLBACK 回滾該次交易，取消之前所有資料庫異動。若 Admin 修改過程順利且未發生資料型別不一致之問題，系統將在最後 COMMIT 提交交易，確保資料被正確修改。

3.5 併行控制

在我們的系統中，Admin 可修改或刪除事故熱點及其相關資料，而 User 和 Viewer 則可以查詢事故熱點資訊，我們認為這段流程需要做併行控制。在 Admin 發現有資料需要修改至資料真正被修改完成，User 或 Viewer 有可能正好在查詢事故熱點資訊，這會造成 Dirty Read 問題，因此必須對此流程進行併行控制。

例如：Admin 發現 9 月 1 日下午 3 點的氣溫資料不正確，原紀錄為 15 度，但實際上應為 30 度，若 Admin 修改期間並未進行併行控制，則剛好某位 Viewer 查詢 9 月 1 日至 9 月 7 日的事故熱點歷史資料，就可能被尚未 COMMIT 的異常值誤導。

我們認為適合的做法為：在 Admin 為了修改資料而查詢某事故熱點時，按下「送出查詢」後開始加鎖，並在 Admin 完成修改並按下「確認」按鈕或取消後後釋放鎖。透過這種作法，可以防止由 User 和 Viewer 發起的其他交易讀到不正確的數值，Viewer 在後續對查詢的資料進行分析時也不會被錯誤資料誤導。

4 分工資訊

在本次專案中，我們的分工資訊如下：

1. 孫子龍：彙整團隊成員工作成果並撰寫綜合報告，負責前端界面程式開發，協助資料庫資料整合導入，並準備及錄製前端操作介面影片。
2. 鐘智瑋：負責國道事故、使用者互動、查詢與確認相關資料之規劃與模擬，建立資料庫架構，並參與專案相關影片的製作與錄製。
3. 林文堯：專注於資料庫資料清理與校正工作，進行查詢案例分析，協助系統架構建立，撰寫說明文件，並協助整合報告內容。
4. 李宗霖：擔任系統分析與設計工作，處理氣候和交通量資料前期處理工作，並負責資料庫建立，協助撰寫報告與影片製作。
5. 吳佳芸：專注於前端介面設計，負責測試索引功能，製作影片簡報，並參與 ER 模型的協作。

5 專案心得

在本次專案中，我們的專案心得如下：

1. 孫子龍：於此次專案中，我得以重新審視並復習先前所學之資料庫知識，包括基礎資料庫架構設計、資料匯入，以及前後端連接等。透過此次的實踐經驗，我迫使自己重新熟悉前端網頁撰寫及安全性的技巧。在此過程中，我對整體系統設計架構有了更深層理解。上一個學年，我曾嘗試使用 MySQL 及 AWS 的 RDS 架構進行設定，而本次我學習了 Postgres 與 PHP 語言，實在是頗為有趣。此外，本次專案中的成員皆展現出卓越團隊合作精神，各盡其職。我深信，這些同學日後投身職場，必將成為極佳團隊夥伴。

2. 鐘智瑋：在與組員們討論好專案題目後，首先困擾的是該蒐集哪些數據，以及該如何蒐集，待分工與確認完成後，也要反覆思考資料正規化的議題，若能早期規劃與確認好，就可以避免後續 Data dictionary 與 ER Diagram 的問題或反覆修改的情況，但仍被助教提醒可能潛在 PK 欄位不具唯一性的問題。雖然老師在學期初說過一般去業界應該都是已有建立好的資料庫，大部分應該都是不斷的使用 SQL 查詢資料，但有這個機會可以透過專案來學習整個資料庫的構想與建置，注意到系統設計與資料檢核在實作上的細節，對於這門課的目標和內容真的有更深的感觸。

3. 林文堯：這是我第一次接觸完整的資料庫設計與系統建置的流程，去實作課堂中學到的各種理論，與老師幫我們都準備好的 backup 檔不同，實際上在訂定主題並找到需要的資料後，要成功的將資料匯入資料庫中比想像中的困難許多，需不斷的重複檢視資料的格式或是在 Data Dictionary 中給定的資料型態。另外方面也學習到在既定的主題及可以蒐集到的資料下，如何正確的透過這些資料去交叉分析出有意義的資訊，這個過程也十分有趣，最後，要再次謝謝老師的指導，有路影較彈性的上課方式可以多次複習上課內容，也感謝內同學們的幫忙，在大家通力合作下才能完成這個專案。

4. 李宗霖：我在這個專案學習到最多的部分是資料欄位設計實務，尤其可以從政府公開資料平臺推出的各種資料庫參考到許多有意義的資料結構，例如經緯度的精準度會影響大地線距離的計算、氣象觀測站的異常值必須以「-99」呈現等等。這說明了不同領域的人士都會有自己的 domain knowledge 來決定最有效率或最符合需求的資料庫設計，而我本身也在提出一個期末專案的想法之後，也面臨著要如何將現實世界中不同來源的數據轉換成擁有具分析意義的資料庫這個問題。幸好在組員的合作與幫助下，很多想法得以跳脫草稿階段，真正被實踐在我們的期末專案裡面。

5. 吳佳芸：在修這堂課以前，我做過的小組專案是用 C++ 或 Python 寫出遊戲程式和介面，這是第一次接觸到要連接前後端的完整系統，因此幾乎每一個步驟都要上網查很久或問 ChatGPT，例如：將資料匯入資料庫比想像中有更多需要注意的細節。我在這次專案學到最多的是資料庫操作和前端介面設計：對於在資料庫匯入資料、下 SQL 指令、讀懂錯誤訊息熟悉了一些；設計前端介面時也有許多細節，比如說「刪除資料」這種較危險的操作按鈕會標為顯眼的紅色、每一介面都應該標上「回到前一頁」等，頓時發覺自己平常用的系統介面對使用者很貼心。這次專案多虧組員的協助和提醒，讓我對完整的系統有更多瞭解！