資料庫管理 (112-1)
期末專案完整報告

Group 15
B08901164 林霈瑀　B11705050 林稚翔　B03611040 蕭鈺龍　R11227120 彭瑋琳

GitHub 連結

展示影片連結

1　系統分析

在社區遇到受傷動物或動物攻擊事件，一時之間不知道該向誰請求協助 ?「 FurAlert! 」提供一個即時的動物救助系統，幫助使用者立即聯繫相關單位尋求協助。使用者可以透過系統通報事件，系統將依據事件的動物種類、事件類型和地點，即時發送通知給相關單位。在收到通知後，救助單位可依事件內容和當下的人力配置選擇是否接手處理。若事件被順利排除，處理事件的救助單位會發出官方報告描述事件的後續處置，若事件無法被順利排除，救助單位也會發送警告至可能遭受影響的使用者和其他單位，提醒他們多加注意。

根據不同的功能及權限，「 FurAlert! 」系統的用户可分為三種身分，分別是 User、Responder 及 Admin。若作為一般使用者，身份別定義為 User，可依自身需求，選擇通報動物相關突發事件，或是瀏覽特定類型事件的通知。若選擇通報事件，使用者可透過介面輸入動物種類、事發地點、事件類型，並用文字和照片敘述事件內容。如果使用者只是想瀏覽特定事件通知，可透過介面選擇感興趣的地點、動物類型、事件類型進行頻道訂閱，完成訂閱後，就能瀏覽相關事件內容，並在該頻道有新的事件通報時收到通知。Responder 是系統合作的救助單位，包含獸醫院、消防局、警察局、區公所和動物保護團體，救助單位透過介面選擇符合自身專業和所在地區的頻道進行訂閱，在頻道有新進的事件通報時，救助單位會收到通知，選擇是否接下處理事件的任務，並在處理後發布報告或警告。而 Admin 則是「 FurAlert! 」系統的管理角色，負責管理使用者帳號和事件通報內容，並且可以查詢所有事件的通報紀錄和處理紀錄，並據此做分析以優化平台營運。另外在第 1.1.5 小節中，我們也列出幾個不是針對特定使用者、救助單位或管理員的功能，而是系統本身運行及維護所需的功能。

1.1　系統功能

1.1.1　關於頻道訂閱的相關設定

系統會提供三個篩選器供使用者選擇欲訂閱的頻道，篩選器的預設值為「 全部 」。第一個篩選器「 地點 」，選項包含台北市和新北市共 40 個行政區；第二個篩選器「 動物 」，包含「 狗/貓/鳥/蛇/鹿/猴/魚/熊/其他 」共 9 個選項；第三個篩選器是「 事件類型 」，包含「 路殺/動物阻礙交通/流浪動物/動物攻擊/虐待動物/危險野生動物出/其他 」共 7 個選項。使用

者可以在三個篩選器中各選 0 至 1 個選項成為一個組合，並新增這個組合至訂閱頻道的清單中。例如若某一使用者想訂閱「大安區/動物阻礙交通」的頻道，可在「地點」選擇「大安區」、「動物」欄位維持預設的「全部」、「事件類型」選擇「動物阻礙交通」，新增這個頻道訂閱後，即可掌握大安區所有動物阻礙交通的事件通報。

救助單位同樣可以依據專業領域、所處地區及特定事件類型選擇訂閱的頻道，以精準掌握與單位最相關的通報事件。

1.1.2 給 User 的功能

在本系統中，User 可以執行以下功能：

1. 註冊帳號：使用者可以輸入姓名、電子信箱、電話的資訊註冊帳號，使用者需註冊帳號並登入才能使用平台功能。完成註冊後，系統會將使用者狀態設定為"Active"。
2. 通報事件：使用者可以描述動物種類、事發的縣市區和路名、事件類型大致的原因 (例如交通事故、攻擊事件、虐待動物、動物出現在馬路上)，並用文字簡單敘述事件內容。使用者也可以選擇上傳事件照片，輔助救助單位衡量事件的嚴重性。
3. 刪除通報事件：使用者可以刪除錯誤發布的通報事件。
4. 訂閱頻道：使用者可以設定想要訂閱的頻道，當該頻道有相關事件被通報時，使用者就會收到通知，可以選擇避開該地點。
5. 查看訂閱的所有頻道：使用者可以查看自己訂閱的所有頻道。
6. 查看收到的歷史通知：使用者可以查看自己收到的歷史通知，包括通知類型、通知時間和事件代號。

1.1.3 給 Responder 的功能

在本系統中，救助單位 Responder 可以執行以下功能：

1. 訂閱頻道：救助單位可以依據地域、救助專業，選擇他們想要訂閱的頻道，當該頻道有相關事件被通報時，救助單位就會收到通知。
2. 接受任務：救助單位收到通知後，可以主動接下救助任務。一個事件任務只能被一個救助單位接受處理。
3. 發布任務完成公告：處理完一個事件後，發出事件的報告，描述事件的動物種類、事件類型、事發地點和處理方式與安置場所。
4. 發布警告：如果事件沒辦法被處理，救助單位會發出警告，通知訂閱相關頻道的使用者和其他單位該事件的地點、動物種類、危急程度和簡短敘述，提醒他們注意這個通報事件。

5. 更新事件狀態：在接受任務後，救助單位可以更新事件的資訊，例如修正事件的確切地點、當事動物的種類和數量。完成任務後，救助單位可以依照事件的處理結果將事件狀態更新為「尚未解決」、「進行中」、「已解決」、「失敗」、「假警報」或「已刪除」。
6. 查看收到的歷史通知：救助單位可以查看自己收到的歷史通知，包括通知類型、通知時間和事件代號。

1.1.4 給 Admin 的功能

在本系統中，平台管理者 Admin 可以執行以下功能：

1. 查詢使用者資訊：管理員可查詢所有使用者的資訊及事件通報紀錄。
2. 查詢救助單位資訊：管理員可查詢所有救助單位的資訊，以及單位處置的事件內容。
3. 管理使用者帳號：在使用者違反平台規定 (例如通報假事件) 時，管理者可將使用者帳號狀態設為"Banned" 予以停權。
4. 管理事件通報：管理者可在事件違反平台規定 (例如通報與動物無關的事件) 時將該事件刪除。

1.1.5 系統級指令

由系統本身運行的功能如下：

1. 根據事件的分類，針對不同的頻道訂閱者發出通知。
2. 根據警告對應的事件的分類，針對不同的頻道訂閱者發出通知。

2 系統設計

2.1 ER Diagram

圖 1是「FurAlert!」的 ER Diagram，共包括六個實體 (entity)，分別是`User`、`Responder`、`Event`、`Channel`、`Animal`、`Placement`，以及三個弱實體 (weak entity)，分別是`Notification`、`Warning`、`Report`，以及 13 個關係 (relationship)。

`User`代表的是平台使用者，經過註冊後才可以開始使用平台功能。系統要求使用者在註冊時提供姓名、密碼、電子郵件信箱、手機號碼，註冊後，系統會為使用者產生一個專屬編號，並將該使用者的狀態設定為"Active"。使用者可以建立和刪除多個事件。

`Responder`代表的是接收事件的救助單位，會有機構名稱、電話、地址、電子郵件信箱、密碼、機構種類 (獸醫院/消防局/警察局/駐警隊/區公所/動物保護團體)。當機構接收一個事件後，可以按照處理進度更改事件狀態。

Event代表的是一個事件。使用者建立一個事件時，會需要提供事件類型、動物資訊、地點、簡短敘述、相關相片。系統會產生一個專屬事件編號以及事件狀態屬性。一個事件被建立後，會由於相關動物、地點、事件類型，對應到一個或多個頻道。

Animal代表的是一個動物，記錄使用者建立事件時的提供的動物資訊，包括動物種類、動物描述。每一個事件的動物都會有一個系統給定的編碼。由於每隻動物實際上沒有編碼，因此相同動物在不同事件當中，有可能被重複記錄，但是我們假設該情況發生的機率很低。

Placement代表的是動物的最終安置，記錄著事件被處理完成後動物的安置地點，包括安置地名稱、安置地地址、安置地電話，資訊由接收事件的機構提供。每個安置點都有獨特的編號，我們假設安置的可能性有野放 (PlacementId=0)、安寧 (PlacementId=1)、多間收容所 (PlacementId>1)。

Warning代表的是一個官方警告，包括警示的危險程度、簡短敘述。當Responder處理失敗一個事件的時候，會將Event的狀態設成「失敗」("Failed")，並可以發出一個警告來警示民衆保護自身或他人的安全。

Report代表的是一個官方報告，包括簡短敘述。當Responder處理完成一個事件的時候，會將Event的狀態設成「已解決」("Resolved")，並發出一個報告書來記錄處理結果。

Channel代表的是一個頻道，由動物種類、發生地點、事件類型，三種屬性排列組合唯一決定。系統會額外對於每一個排列組合產生一個專屬的頻道編號 (ChannelId) 當作主鍵。Users和Responder皆可以「訂閱」自己關注的頻道，當該頻道有相關事件或警告被發布的時候，使用者和機構會收到通知。

Notification代表的是一個通知，包括了通知時間、通知種類 (事件/警告)，由一個頻道發送給頻道訂閱者。

2.2 Relational Database Schema Diagram

圖 2 是「FurAlert!」從 ER Diagram 映射 (mapping) 的 relational database schema diagram。由圖所示，這個系統一共有 12 張 table，分別為 Users、UserInfo、ResponderInfo、Event、EventImages、Channel、Warning、Report、SubscriptionRecord、Notification、Animal、Placement。

考量到使用者的種類有 User 、Responder 、Admin，三種角色，我們設計了 Users 來存放登入需要的帳號、密碼以及角色，再把 User 的資訊存到 UserInfo ，把 Responder 的資訊存到 ResponderInfo 。

Users 的主鍵是 UserId，是系統在使用者在註册帳號時產生的。另外，還會儲存Password, Email, Role。

ResponderInfo 的主鍵是 ResponderId，也是指向 Users 的主鍵 UserId 的外部

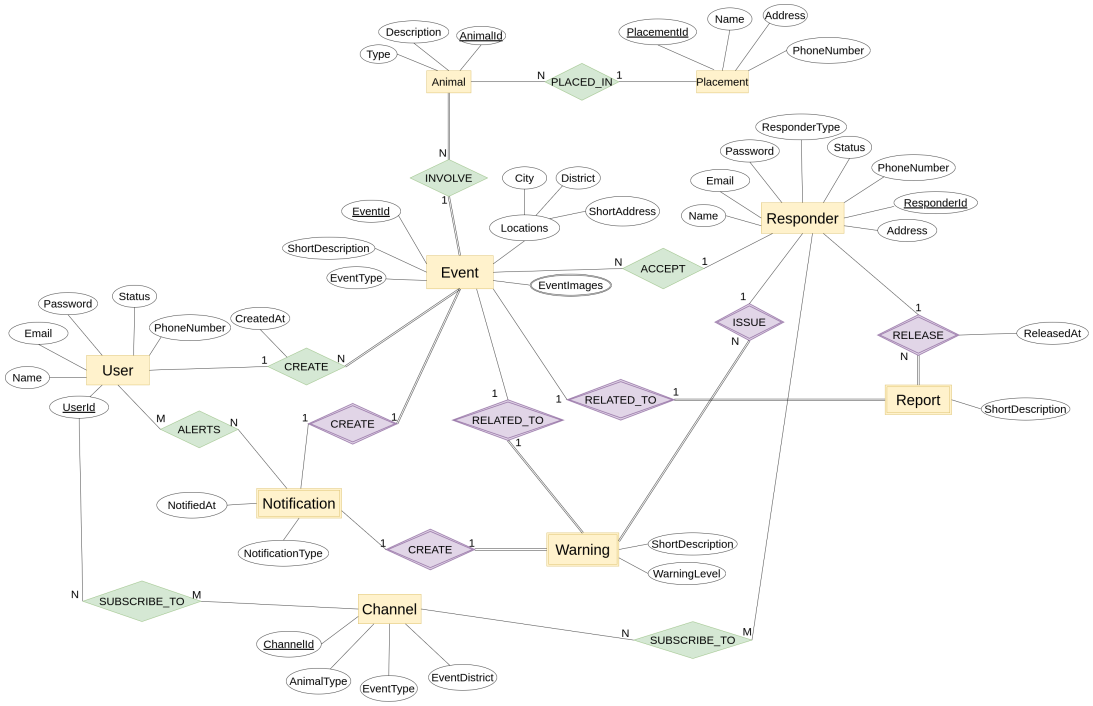

圖 1: Entity Relationship Diagram

鍵，來記錄對應到哪一個系統使用者。此表還會記錄 ResponderName, PhoneNumber, ResponderType, Address。

UserInfo 的主鍵是 UserId，也是指向 Users 的主鍵 UserId 的外部鍵，來記錄對應到哪一個系統使用者。此表還會記錄 Name, PhoneNumber, Status。

Event 的主鍵是 EventId，是使用者在建立事件時系統會產生的。此表還會記錄 EventType, Status, ShortDescription, City, District, ShortAddress, CreatedAt，另外還會記錄 UserId 和 ResponderId 指向 Users 的主鍵來對應是哪一個使用者建立的，和哪一個機構處理這個事件的。

EventImages 的主鍵是 ImageId，是使用者在上傳照片時系統會建立的，使用連結的方式來記錄 ImageLink，因為一個事件可能會對應超過一張的圖片，是多值屬性，所以我們獨立出 EventImages，透過一個 EventId 的外部鍵指向 Event 的主鍵。

Channel 的主鍵是 ChannelId，是由系統在建置的時候，排列組合其他紀錄的屬性：EventDistrict, EventType, EventAnimal的方式形成編號建立在 Channel 中。

Warning 的主鍵是 {EventId, ResponderId}，因為是依賴 EventId 和 ResponderId 形成的弱實體，同時也是指向 Event 的 EventId 和 Users 的 UserId 形成外部鍵同時也會記錄 WarningLevel, ShortDescription, CreatedAt。

Report 的主鍵也是 {EventId, ResponderId}，因為是依賴 EventId 和 ResponderId

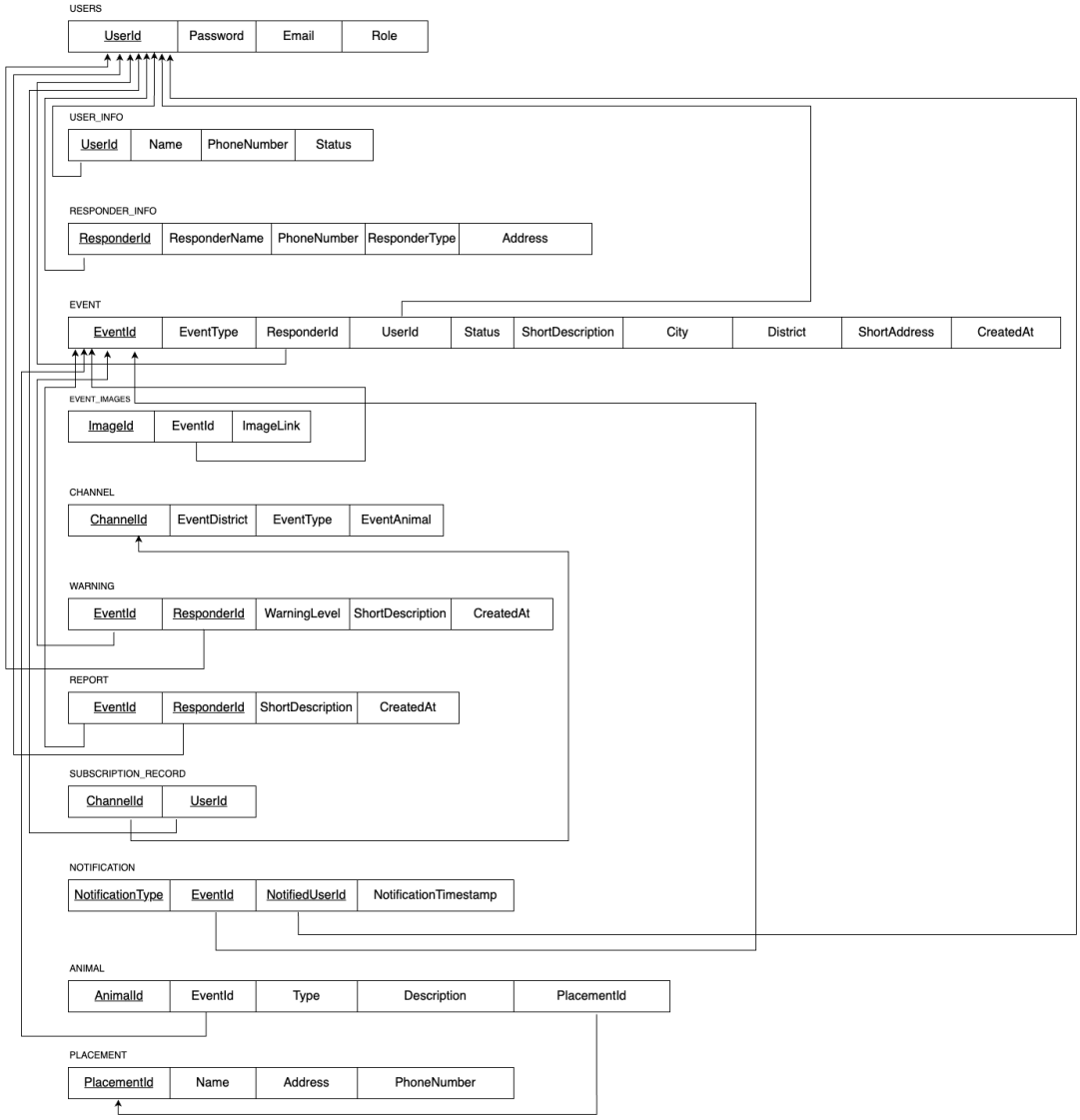

圖 2: Relational Database Schema Diagram

形成的弱實體，同時也是指向 Event 的 EventId 和 Users 的 UserId 形成外部鍵。同時也會記錄 ShortDescription, CreatedAt。

SubscriptionRecord 的主鍵是 {ChannelId, UserId}，因為這個表記錄 Users 訂閱的 Channel 的多對多關係，同時也是指向 User 的 UserId 和
Channel 的 ChannelId 的外部鍵。

Notification 的主鍵是 {EventId, NotifiedUserId}，因為這個表是記錄每一個 Event 發送給 User 的通知的多對多關係，同時也是指向 Event 的 EventId 和 Users 的 UserId 的外部鍵。

Animal 的主鍵是 AnimalId，是使用者在建立事件時輸入通報的動物系統會產生的。此表還會紀錄 Type, Description，另外還會記錄 EventId 和

PlacementId 指向 Event 的 EventId 和 Placement 的 PlacementId，來記錄動物是對應哪一個事件，和最後安置的地方。

Placement 的主鍵是 PlacementId，是系統在建立安置場所時會產生的。此表還會記錄 Name, Address, PhoneNumber。

2.3 Data Dictionary

「FurAlert!」的資料表共有圖 2 所示的 12 張，各個資料表的欄位相關資訊依序呈現在表 1 到表??。

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserId	使用者編號	int	PK	Not Null	
Password	使用者密碼	varchar (60)		Not Null	
Email	使用者信箱	varchar(30)		Not Null, Unique	
Role	使用者身份	char(20)		Not Null	{User, Responder, Admin}

表 1: 資料表 Users 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
UserId	使用者編號	int	PK, FK: Users(UserId)	Not Null	
Name	使用者名稱	varchar (20)		Not Null	
PhoneNumber	使用者電話	char(10)		Not Null	
Status	使用者狀態	varchar(15)		Not Null	{Active, Banned}
Referential triggers		On Delete		On Update	
UserId: Users(UserId)		Cascade		Cascade	

表 2: 資料表 UserInfo 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ResponderId	救助單位編號	int	PK, FK: Users(UserId)	Not Null	
ResponderName	救助單位名稱	varchar (60)		Not Null	
PhoneNumber	救助單位電話	char(10)		Not Null	
ResponderType	救助單位種類	varchar(60)		Not Null	{Vet, Police, FireAgency, AnimalProtGroup, DistOffice, Other}
Address	救助單位地址	varchar(60)		Not Null	
Referential triggers		On Delete		On Update	
ResponderId: Users(UserId)		Cascade		Cascade	

表 3: 資料表 ResponderInfo 的欄位資訊

2.4 正規化分析

在 1NF 方面，如果每個屬性都是單值的 (simple, single-valued)，則滿足 1NF。在第 2.2 節，我們將 EventImages 從 Event 當中移出來，成為 EventImages 這個關聯。因此設計符合 1NF。

在 2NF 方面，如果每個關聯中的所有次要屬性 (non-prime attribute) 都完全功能相依 (fully functional dependent) 於任一候選鍵，則關聯滿足 1NF 和 2NF。我們的設計沒有部分功能相依 (partially dependent) 的情況發生，因此滿足 2NF。

Column Name	Meaning	Data Type	Key	Constraint	Domain
EventId	事件編號	int	PK	Not Null	
EventType	事件類型	varchar (30)		Not Null	{RoadkillAccident, AnimalBlockingTraffic, StrayAnimal, AnimalAttack, AnimalAbuse, DangerousWildlifeSighting, Other}
ResponderId	救助單位編號	int	FK: Users(UserId)		
UserId	使用者編號	int	FK: Users(UserId)	Not Null	
Status	事件狀態	varchar(20)		Not Null	{Ongoing, Resolved, Unresolved, Failed, False alarm, Deleted}
ShortDescription	事件簡述	varchar(100)			
City	事件發生城市	varchar(20)		Not Null	{Taipei, NewTaipei }
District	事件發生區域	varchar(20)		Not Null	
ShortAddress	事件詳細地址	varchar(30)			
CreatedAt	事件發生時間	timestamp		Not Null	
Referential triggers		On Delete		On Update	
ResponderId: Users(UserId)		Cascade		Cascade	
UserId: Users(UserId)		Cascade		Cascade	

表 4: 資料表 Event 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
PlacementId	安置方式編號	int	PK	Not Null	
Name	安置方式名稱	varchar (40)		Not Null	
Address	收容所地址	varchar(30)			
PhoneNumber	收容所電話	char(10)			

表 5: 資料表 Placement 的欄位資訊

在 3NF 方面，因為「FurAlert!」的關聯中的非鍵屬性都沒有遞移相依於主鍵，所以滿足 3NF 的條件。

在 4NF 方面，由於「FurAlert!」的所有關聯都不存在多值相依，因此滿足 4NF 的條件。

3 系統實作

3.1 資料庫建置方式及資料來源說明

關於Userinfo資料表內的資料採取隨機生成，共生成 50000 個使用者。接著，我們利用這些使用者模擬「新增事件通報」的行為，隨機生成事件通報的記錄，儲存在Event資料表，共 220000 筆事件通報資料。從事件資料表的資料，再隨機生成圖片連結共 440007 筆資料存在EventImages資料表，並且根據事件資料表的狀態和救助單位，生成Warning資料表中 33009 筆資料和Report資料表中 87883 筆資料。

關於Responderinfo資料表內的資料，我們利用爬蟲從台北市政府和新北市政府網頁搜集 40 筆行政區公所資料、136 筆消防局資料、255 筆警察局 (含派出所) 資料，並從台北市與新北市獸醫師公會搜集共 496 間獸醫院資料和 11 筆動保協會資料，建置共 939 筆救助單位匯入資料庫對應的資料表。

Column Name	Meaning	Data Type	Key	Constraint	Domain
AnimalId	動物編號	int	PK	Not Null	
EventId	相應事件編號	int	FK: Event(EventId)	Not Null	
Type	動物種類	varchar(6)		Not Null	{Dog, Cat, Bird, Snake, Deer, Monkey, Fish, Bear, Other}
Description	動物相關簡述	varchar(150)			
PlacementId	動物安置方式	int	FK: Placement(PlacementId)		
Referential triggers		On Delete		On Update	
EventId: Event(EventId)		Cascade		Cascade	
PlacementId: Placement(PlacementId)		Cascade		Cascade	

表 6: 資料表 Animal 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ImageId	圖片編號	int	PK	Not Null	
EventId	事件編號	int	FK: Event(EventId)	Not Null	
ImageLink	事件圖片連結	varchar (100)		Not Null	
Referential triggers		On Delete		On Update	
EventId: Event(EventId)		Cascade		Cascade	

表 7: 資料表 EventImages 的欄位資訊

關於Users資料表內的資料，除了 50000 個一般使用者和 939 個救助單位之外，還存放兩位管理員的資料，共 50941 筆資料。

Placement資料表內的資料建置方式，除了PlacementId 為 0 是「野放」、1 是「安寧緩和」之外，其他資料是透過爬蟲從台北市和新北市動物之家網頁、台灣動物緊急救援小組網站和台北市動保處網頁，搜集共 8 個動物收容所、219 筆貓狗園資料，共 229 筆資料，匯入資料庫對應的安置方式資料表。

關於Channel資料表內的資料，我們透過先前設定的 40 個行政區、7 種事件類型和 9 種動物排列組合，生成共 3360 筆資料。根據頻道資料表內的資料，搭配訂閱頻道的使用者編號和救助單位編號，生成 SubscriptionRecord中共 15000 筆資料。接著，為訂閱頻道的使用者和救助單位產生事件通知的 337843 筆資料，存在 Notification 資料表中。

3.2 重要功能及對應的 SQL 指令

第 1.1 節中我們有介紹一些給 User、Responder 和 Admin 的功能，在第 3.2.1 至第 3.2.3 小節中，將列出特定情境下，完成這些功能所使用的 SQL 指令。此外，在第 3.2.4和 3.2.5 小節中，我們也列出系統分析及系統級指令對應的 SQL 指令。

3.2.1 給 User 的功能

1. 新增事件：若要實現此功能，假設情境為「通報者代號 UserId『1234』想通報一個目擊事件 Event ，事件類型 EventType 為『StrayAnimal』，發生地點在城市 City『台北市』、區域 District『大安』，而事件發生地址 Road 為『羅斯福路四段 1 號』，事件簡述 ShortDescription 為『路旁的草叢內看到兩隻幼犬，附近沒有其他

Column Name	Meaning	Data Type	Key	Constraint	Domain
ChannelId	頻道編號	int	PK	Not Null	
EventDistrict	頻道事件地區	varchar (20)			
EventType	頻道事件類型	varchar (30)			{RoadkillAccident, AnimalBlockingTraffic, StrayAnimal, AnimalAttack, AnimalAbuse, DangerousWildlifeSighting, Other}
EventAnimal	頻道動物類型	varchar (15)			{Dog, Cat, Bird, Snake, Deer, Monkey, Fish, Bear, Other}

表 8: 資料表 Channel 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
EventId	事件編號	int	PK, FK: Event(EventId)	Not Null	
ResponderId	救助單位編號	int	PK, FK: Users(UserId)	Not Null	
WarningLevel	警告等級	int		Not Null	[1, 10]
ShortDescription	警告簡述	varchar(150)			
CreatedAt	警告發出時間	timestamp		Not Null	
Referential triggers		On Delete		On Update	
EventId: Event(EventId)		Cascade		Cascade	
ResponderId: Users(UserId)		Cascade		Cascade	

表 9: 資料表 Warning 的欄位資訊

類似父母的犬隻。』。與該事件相關的動物 Animal 數量為二，動物種類 Type 皆為『Dog』，而其特徵簡述 Description 分別為『白色，大約七個月大』與『黑色，大約七個月大，身上有傷口』。該通報者也上傳了一張該事件的圖片 EventImages，圖片連結 ImageLink 為後端產生的圖片路徑，目前是以 UUID 字串為檔名，因此可能的值為 :『uploads/92a97190-b61a-4cca-be87-4a45a0f7edea.jpg』。」則對應的 SQL 指令如下。系統會在 Event 的資料表新增一筆事件的資料，並在 Animal 資料表中為該事件新增兩筆相關動物的紀錄，以及在 EventImages 資料表中新增一筆有關該事件的圖片的紀錄。

```
Do $$
Declare
        eId Event.EventId%type;
Begin
    Insert Into Event(EventType, UserId, Status,
       ShortDescription, City, District, ShortAddress,
       CreatedAt)
    Values ('StrayAnimal', 1234, 'Ongoing', '路旁的草叢內
       看到兩隻幼犬，附近沒有其他類似父母的犬隻。', '台北
       市', '大安區', '羅斯福路四段1號', now());
    eId = lastval();
    Insert Into Animal(EventId, Type, Description)
    Values (eId, 'Dog', '白色，大約七個月大');
    Insert Into Animal(EventId, Type, Description)
    Values (eId, 'Dog', '黑色，大約七個月大，身上有傷口');
    Insert Into EventImages(EventId, ImageLink)
    Values (eId, 'uploads/92a97190-b61a-4cca-be87-4
       a45a0f7edea.jpg');
End; $$;
```

Listing 1: 新增事件

Column Name	Meaning	Data Type	Key	Constraint	Domain
EventId	事件編號	int	PK, FK: Event(EventId)	Not Null	
ResponderId	救助單位編號	int	PK, FK: Users(UserId)	Not Null	
ShortDescription	報告簡述	varchar(150)			
CreatedAt	報告發出時間	timestamp		Not Null	
Referential triggers		On Delete		On Update	
EventId: Event(EventId)		Cascade		Cascade	
ResponderId: Users(UserId)		Cascade		Cascade	

表 10: 資料表 Report 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
ChannelId	頻道編號	int	PK, FK: Channel(ChannelId)	Not Null	
UserId	使用者或救助單位編號	int	PK, FK: Users(UserId)	Not Null	
Referential triggers		On Delete		On Update	
ChannelId: Channel(ChannelId)		Cascade		Cascade	
ResponderId: Users(UserId)		Cascade		Cascade	

表 11: 資料表 SubscriptionRecord 的欄位資訊

2. 刪除事件：若要實現此功能，假設情境為「通報者代號 UserId『1234』想要刪除由其通報的事件編號 EventId 為『10001』的事件。」則對應的 SQL 指令如下。系統會在 Event 資料表中將該事件的狀態轉成”Deleted”。

```
BEGIN;
SELECT status FROM Event WHERE eventid = 10001 FOR UPDATE;
UPDATE Event SET status = 'Deleted' WHERE eventid = 10001;
COMMIT;
```

Listing 2: 刪除事件

3. 訂閱頻道：若要實現此功能，假設情境為「通報者代號 UserId『1234』想訂閱一個頻道 Channel，該頻道對應發生地區 EventDistrict 位於『大安區』、類型 EventType 為『StrayAnimal』並且相關動物 EventAnimal 為『Dog』的事件。」則對應 SQL 指令如下。系統會在 Channel 資料表中找到符合條件的 ChannelId，並新增一筆包含該頻道和該使用者的資料到 UserSubscriptionRecord 資料表中。

```
Insert Into SubscriptionRecord(ChannelId, UserId)
Select c.ChannelId, 1234
From Channel As c
Where c.EventDistrict = '大安區' And c.EventType = '
   StrayAnimal' And c.EventAnimal = 'Dog'
```

Listing 3: 訂閱頻道

4. 查詢訂閱的所有頻道：若要實現此功能，假設情境為「通報者代號 UserId『1234』想要查看自己訂閱的所有頻道。」則對應 SQL 指令如下。系統會執行該查詢，並回傳 UserId 為 1234 的通報者訂閱的所有頻道的相關資訊。

```
Select c.*
From Channel As c
    Join SubscriptionRecord As subscription On
       subscription.ChannelId = c.ChannelId
```

Column Name	Meaning	Data Type	Key	Constraint	Domain
NotificationType	通知種類	varchar(7)	PK	Not Null	{Event, Warning}
EventId	事件編號	int	PK, FK: Event(EventId)	Not Null	
NotifiedUserId	通知使用者編號	int	PK, FK: Users(UserId)	Not Null	
NotificationTimestamp	通知時間	timestamp		Not Null	
Referential triggers		On Delete		On Update	
EventId: Event(EventId)		Cascade		Cascade	
UserId: Users(UserId)		Cascade		Cascade	

表 12: 資料表 Notification 的欄位資訊

```
Where subscription.UserId = 1234
```

Listing 4: 查詢訂閱的所有頻道

3.2.2 給 Responder 的功能

1. 接受任務：機構在看到尚未被接受的通報事件時，可以接下該事件進行調查與回報。假設情境為「機構的ResponderId為『368』，並且想要接下事件代號EventId『789』的事件時」則對應的 SQL 指令如下。

```
Update EVENT
Set ResponderId = 368, Status = 'Ongoing'
Where EventId = 789;
```

Listing 5: 回應事件 SQL 指令

2. 查詢收到的歷史通知：機構可以查詢自己收到的歷史通知，包括通知類型、通知時間和事件代號。假設情境為「機構的ResponderId為『368』，並且想要查詢過去的歷史通知資訊。」則對應的 SQL 指令如下。

```
Select n.*
From Notification As n
Where NotifiedUserId = 368;
```

Listing 6: 回應事件 SQL 指令

3.2.3 給 Admin 的功能

1. 查詢使用者資訊及其事件通報紀錄：假設情境為「查詢使用者代號UserId為『1234』的使用者帳號資訊和通報紀錄。」則對應的 SQL 指令如下。系統會執行該查詢，並回傳使用者代號為 1234 的使用者帳號資訊及其所有事件通報紀錄。

```
Select *
From USERS As u
  Join EVENT As e On e.userid = u.userid
Where u.userid = 1234
```

Listing 7: 查詢使用者資訊 SQL 指令

2. 管理使用者帳號：假設情境為「某管理員想要將使用者代號UserId『1357』的帳號狀

態Status改為Banned『停權』。」則對應的 SQL 指令如下。系統會更新Users資料表中，使用者代號及帳號狀態兩個欄位資訊。

```
Update USERS
Set Status = 'Banned'
Where UserId = 1357;
```

Listing 8: 停止使用者帳號權限 SQL 指令

3.2.4 系統級指令

1. 根據事件（給定一個 EventId = 93）對應的頻道，針對訂閱頻道的一般使用者（Users）發出通知。

```
-- insert notification
INSERT INTO Notification(notificationtype, eventid,
   notifieduserid, notificationtimestamp)
SELECT 'Event', 93, userinfo.userid, NOW()
FROM(
        SELECT DISTINCT user_sub.userid
        FROM SubscriptionRecord AS sub
        WHERE sub.channelid IN
        (
                SELECT c.channelid
                FROM(
                        SELECT e.eventtype, e.district, an
                           .type AS animaltype
                        FROM Event AS e
                                JOIN Animal AS an ON e.
                                   eventid = an.eventid
                        WHERE e.eventid = 93
                ) AS info
                JOIN Channel AS c ON c.eventanimal = info.
                   animaltype OR c.eventdistrict = info.
                   district OR c.eventtype = info.
                   eventtype
        )
) AS userinfo;
```

Listing 9: 根據事件對於訂閱者發出通知 SQL 指令

3.2.5 系統分析指令

1. 知道解決最多事件的前十間處理機構（Responder）的編號和名字

```
SELECT res.ResponderId, res.ResponderName
FROM Report AS rep
    JOIN Responder AS res ON res.ResponderId = rep.
       ResponderId
GROUP BY res.ResponderId
ORDER BY COUNT(*) DESC
```

```
LIMIT 10;
```

Listing 10: 解決最多事件的前十間處理機構 SQL 指令

2. 知道前十名容易發生路殺事件的地點 (城市和區域的組合)

```
SELECT e.City, e.District
FROM Event AS e
WHERE e.EventType = 'Roadkill'
GROUP BY e.City, e.District
ORDER BY COUNT(*) DESC
LIMIT 10;
```

Listing 11: 前十名容易發生路殺事件的城市和區域 SQL 指令

3.3 SQL 指令效能優化與索引建立分析

由於「Furalert!」中經常被系統執行的動作是頻道訂閱者的查尋，因此會需要對這個程序做加速。以下方「查找事件相關頻道訂閱者」的功能為例，這些需要以頻道為條件來搜尋頻道訂閱者的操作，執行時間會比較費時。

```
SELECT DISTINCT sub.userid
FROM SubscriptionRecord AS sub
WHERE EXISTS
(
        SELECT *
        FROM Event AS e
                JOIN Animal AS an ON e.EventId = an.EventId
                JOIN Channel AS c ON c.EventAnimal = an.Type
                   OR c.EventDistrict = e.District OR c.
                   EventType = e.EventType
        WHERE e.EventId = 93 AND c.ChannelId = sub.ChannelId
);
```

Listing 12: 查找事件相關頻道訂閱者 SQL 指令

為了能有效優化效能，因此為 Inner Loop(Animal) 建立索引，語法如下。

```
Create Index idx_Animal
On Animal(EventId)
```

Listing 13: 建立 index 語法

不幫 Channel 的屬性建立索引的原因是因為他的條件取聯集而非交集，經過實驗發現無法加速查詢的速度。在建立索引之前，查找跟某個事件相關訂閱的頻道的使用者，平均會需要耗費 155.4ms；建立索引後，則可以加速成平均耗時 131.7ms。建立索引後平均運行時間降低，表示索引可能對查詢性能的提升有幫助。

3.4 交易管理

每一個 flask endpoint 配上特定的 request method(GET, POST)，都可以看成是在實做某一個功能，因此，對於上述的情況我們有做交易管理。以新增一個事件為例：我們總共

會需要新增一個事件以及相關的動物和照片，因此在 addevent 這個 endpoint 可以看到，這個 session 之內我們會連續的加入事件、動物、照片，如果當中有任何的 error，則會進行 rollback，中間所加入的資訊也不會被儲存。簡化過得程式碼如下。

```
try:
    # CREATE EVENT
    new_event = Event(eventtype=eventtype, \
        userid=userid, \
        responderid=responderid, \
        status=status, \
        shortdescription=shortdescription, \
        city=city_str, district=district_str, \
        shortaddress=shortaddress, \
        createdat=createdat)

    db_session.add(new_event)
    # FLUSH TO GET THE NEW EVENTID
    db_session.flush()

    # CREATE EVENTIMAGES
    eventimages = []
    if 'eventimages' in request.files:
        files = request.files.getlist('eventimages')
        ...
        for file in files:
            ... # ERROR CHECKING, RENAMING AND RESIZING
            eventimages.append(EventImages(eventid=new_event.
               eventid, imagelink=image_link))

        db_session.bulk_save_objects(eventimages)

    # CREATE ANIMALS
    animal_types = set()
    for animal in eventanimals:
        ...
        new_animal = Animal(eventid=new_event.eventid, \
                            placementid=None, \
                            type=new_animaltype, \
                            description=new_animaldescription)
        db_session.add(new_animal)
    # COMMIT ALL CHANGES TO DATABASE
    db_session.commit()
except exc.SQLAlchemyError as e:
    db_session.rollback()
    db_session.close()
    return redirect(url_for('add_event'))

db_session.close()
```

Listing 14: 插入新事件的交易管理流程

3.5 併行控制

我們的專案內部對併行控制有要求的部份主要是對事件屬性的修改。當一個事件的狀態是 Unresolved，任何機構都可以去接收這個事件。我們的上鎖程序如下：(1) 接收到一個來自一個機構帳號的 POST request (2) 先 query 該事件，確定狀態其狀態 (3) 如果狀態是 Unresolved，代表這個 request 想做的事情是接收這個事件 (4) 對這個事件上鎖 (FOR UPDATE 的鎖) (5) 再次檢查上鎖的事件是不是 Unresolved，如果是才能做接收，否則應該將 session 關閉，因為該事件的狀態已經改變，所以不能做接收的動作 (6) 如果通過檢查，則接收事件：將狀態設置成為 Ongoing，ResponderId 設成這個機構帳號的 Id。

在步驟 (4) 當中，導致檢查不通過的情況有以下種：原先的通報者或是管理員把事件刪掉、有令一個機構對這個事件做接收。以上的程序對應到的程式碼如下方敘述。

當一個機構接收到一個事件以後，不會再有其他機構和他競爭接收，但是仍有可能在編輯事件內容的時候，該事件突然被原先的通報者或管理員刪掉。因此，在對於已經接收的事件的編輯時，仍然需要對事件上鎖。所以事件如果被刪除，一定是發生在對於該事件的改動發生之前或是之後，不會在改動到一半的時候被刪掉。

```
event = db_session.query(Event).filter(Event.eventid ==
    eventid).first()
# check the event status
if event.status == EVENT_STATUS[EventStatus.UNRESOLVED.value]:
    # responder accepts event
    try:
        # lock
        locked_event = db_session.query(Event).with_for_update
            ().filter(Event.eventid == eventid).first()

        # check status again after getting the lock
        if locked_event.status == EVENT_STATUS[EventStatus.
            UNRESOLVED.value]:
            # update
            locked_event.status = EVENT_STATUS[EventStatus.
                ONGOING.value]
            locked_event.responderid = current_user.userid
            db_session.commit()

        db_session.close()
        return redirect(url_for("event", eventid=eventid))
```

Listing 15: 編輯事件的併行控制流程

4 分工資訊

在本次專案中，我們的分工資訊如下：

林霈瑀：規劃系統功能、系統設計與介面線框圖；撰寫計畫書 Relational Database Schema

和正規化分析；撰寫設計書系統分析與功能、ERD、系統實作的 SQL 指令與效能優化分析；撰寫完整報告的交易管理和併行控制內容；系統整體架構的建立和後端開發、系統除錯。

林稚翔：規劃系統功能、系統設計與介面線框圖；撰寫計畫書系統功能；撰寫設計書 Data Dictionary、系統實作的 SQL 指令；系統前端開發。

蕭鈺龍：規劃系統功能、系統設計與介面線框圖；撰寫計畫書 Data Dictionary；撰寫設計書 Relational Database Schema、系統實作的 SQL 指令；系統資料建置與資料庫優化；開發系統後端 Admin 端點。

彭瑋琳：規劃系統功能、系統設計與介面線框圖；撰寫計畫書 ERD；撰寫設計書系統功能與分析、系統實作的 SQL 指令；系統 UI 設計；開發系統後端 Admin 端點、調整 css。

5 專案心得

在本次專案中，我們的專案心得如下：

林霈瑀：我覺得 ORM 和純粹寫 SQL 真的差滿多的，除了語法以外，有的功能 ORM 也沒有支援，或是 documentation 寫的不清楚，寫起來會卡卡的，像是 subquery 我最後都直接拆成兩個 query 來寫。但是或許這就是可讀性和效率之間的取捨吧！

林稚翔：上課說資訊系統最難處理的階段是設計，但自己親身做過一次後發現明明設計和實作都很困難！在設計時，對效率和可擴充性每個人都有自己的想法，而實作時把我們平時常見的網頁設計放上自己的前端之後，才知道裡面其實有很多眉角和令人困惑的 bug，因此終於完成可以用的系統時真的非常感動。

蕭鈺龍：在專案中獲得最大的學習就是和組員學習他們的長處，看到很多自己不會或是不曾思考過的想法和做法，刺激自己有更多的想法，遇到不會或是不熟悉的，大家也都會很熱心地分享解決辦法。最大的困難是設計和實作時才發現的問題，例如原本設計了覺得可以很方便存取資料的方式，實作時卻發現反而佔空間和拖累效能，還好在和大家討論之後都能夠獲得完善的解決方式。

彭瑋琳：做專案深刻體會到每個環節都很重要，前期設計的決定可能影響後期的資料串接等細節。對我來說最大的困難可能是將設計實作出來，這個從 0 到 1 的過程，很感謝組員帶著原先什麼都不會的我一路做到實作出一個系統，總是很有耐心和熱心的教我可以如何做，每週都覺得和大家學到很多，最後能自己做到前後端串接時真的很感動。