資料庫管理（113-1）
期末專案

第 27 組
B10704031 朱柏諺、B09705018 尤喆、B10705018 林至偉
2024 年 12 月 11 日
https://github.com/brianCHUCHU/iclinic.git
https://www.youtube.com/watch?v=LhqGdu6vF4I

1 系統分析

「iclinic」是一款診所與患者的多邊平臺，實質上可以視爲診所的資訊系統解決方案。現在除了大醫院有開發自己的系統、app 以外，小診所多難有餘力或必要性開發自己的應用程式，且由於缺乏整合的平台，患者下載單一一個診所的 app 效益也不大。不過站在平臺策略的思維，這當中存在潛在的商業利益，只要能夠招募足夠多的診所建立我們的資訊系統，以及吸引足夠多患者加入會員，就可以形成多邊平臺效應（即越多的診所對患者來說效益越大，越多的會員對診所效益越大），其潛在業務包含提供診所廣告服務、開放公開評價、診所顧問服務、交叉銷售等等。此企劃書先從一般醫療系統的基本建設做起，讓患者能夠查詢叫號進度、掛號、預約治療，讓診所方能管理這些系統，並將這些基本系統拓展到多邊平臺規模。

1.1 系統功能

1.1.1 系統設定

系統主要包含兩種使用者，診所與患者。患者。基本上所有跟診所功能相關的功能、資料範圍都能由診所自己定義，包含時段怎麼設定、時段長度、僱多少醫生、有幾個科別、使用分開還是合併叫號的模式等等，唯一的限制只有診所只能從兩種模式當中選擇一種叫號模式，不能採用混合模式。此外，爲了維持系統的一致性，不論患者是不是會員，都需要把最基本的資料 (一般狀況下診所都會有的例如出生年月日) 記錄在系統的就診記錄裡面，但詳細的資訊、病歷則因爲個資考量不需要記錄。

掛號與預約被視爲兩種不同的功能，其中掛號指的是在一個時段一個佇列，就診的精確時間純視看診速度而訂；預約則是一個時段只會有一個患者，就診的時間就是時段的起始時間。當然，兩者的時段長度都可以由診所自訂，兩個系統也可以在同一個時間區間內並行。

以下是叫號/掛號系統的詳細說明：

- 叫號系統：診所方可以選擇分開叫號或是合併叫號，分開叫號的每一個診間每一個時段有各自獨立的號碼，合併叫號則同一個科別同一個時段共用一組號碼。
- 掛號系統：患者可以選擇現在掛號或是預約未來時段的掛號，不論診所方的叫號系統屬於分開還是合併叫號，患者都可以在特定時段對科別掛號（不分診間）。如果院方屬於分開叫號系統，患者選擇對科別掛號時由系統分配醫生（診間）給予號碼，且患者也可以選擇對某一個醫生（診間）進行掛號。患者同一個時段內不能對同一科有一個以上的掛號。

1.1.2 給 Patient 的功能

在本系統中，Patient 可以執行以下功能：

1. 創建帳號：紀錄患者的姓名、生日、性別。
2. 查詢附近診所：患者可以查詢診所、科別。
3. 線上掛號：對使用者來說，線上掛號有兩種：
   (a) 對科掛號：不指定醫師，由診所方分配。
   (b) 對特定醫師（時段）掛號：指定醫師進行掛號。
4. 取消掛號：患者可以取消掛號。
5. 查詢叫號進度：患者可以看到自己的號碼以及診所現在的叫號進度。
6. 預約治療：患者可以透過系統或診所方，選擇診所預先訂定的治療內容，登記治療時段。(e.g. 牙醫：患者可以以時段預約牙醫看診)
7. 提供回饋：患者可以根據就診經驗對診所提供回饋。

1.1.3 給 Clinic 的功能

在本系統中，Clinic 可以執行以下功能：

1. 創建帳號：詳細登記診所資訊，包含診所位置、服務項目。
2. 建立醫生雇用紀錄：診所可以記錄醫生所屬的科別和雇用的開始和結束時間。
3. 查詢就診紀錄：診所可以查詢掛號時間、就診時間、患者資訊。
4. 管理診間、醫生排程：診所可以對診間、醫生的時間表安排進行調整，分配醫師至各診間的任意時段。

5. 叫號/掛號系統：診所可以選擇分開叫號或合併叫號系統，不得混合使用。此外診所方也可以手動調整號碼的順序，也可以線下協助掛號。診所也可以在超負荷的時候選擇關閉掛號系統，雖然理論上不能取消患者的掛號，但依然有設計這個功能以防緊急狀況。

6. 預約系統：與掛號系統類似，診所可以預先訂定治療方案，讓患者透過線上或線下預約治療。如果是線上預約，診所可以在患者預約以後視情況選擇預約成功與否。

7. 查看患者回饋：診所可以根據患者填寫的回饋表單了解業務問題進而改善業務。

1.1.4 給 Admin（平臺）的功能

在本系統中，除了未授權個資的患者以外，Admin 可以查看所有資料表當中的資料，以供商業分析使用，商業分析的主要功能包含但不限於：

1. 顧問服務：平臺方可以透過分析用户、診所等營業資訊，開展顧問服務，像是可以提供新診所的地點建議、診所科別的地區飽和程度分析。

2. 優化業務：邏輯與顧問服務類似，但服務對象變成系統方，可以思考如何透過這些資訊吸收更多用户、診所，甚至開拓新業務。

2 系統設計

2.1 ER Diagram

圖 1 是本專案的 ER Diagram，PATIENT 表示病患的基本資料，包含病患的姓名、生日、性別，以及會員狀態等資訊（如會員或非會員）。此實體與兩個弱實體有關，分別是 MEMBERSHIP 和 SEARCHING_RECORD。MEMBERSHIP 表記錄病患在系統中的帳户資料，包括密碼和電子郵件地址，用於會員管理；SEARCHING_RECORD 表記錄病患使用搜尋功能的行爲資料，例如搜尋關鍵字、地理位置、城市和目標科別等，爲診所推薦或數據分析提供支持。

CLINIC 是系統的核心關聯之一，包含診所的基本資訊，如診所名稱、地址、收費標準，以及診所的叫號模式（共享叫號或獨立叫號）。此外，CLINIC 表還記錄診所的帳户名稱和密碼，用於系統登入管理。CLINIC 與多個其他關聯相關聯，例如 DIVISION、DOCTOR 和 ROOM。DIVISION 表描述診所內的部門（如內科或外科），並確保診所可以對應多個部門。CLINIC_DIVISION 關聯進一步表示診所與部門的多對多關係，記錄診所內部的部門結構，包括部門是否仍在運營、隊列資訊和更新時間。

DOCTOR 關聯記錄醫生的基本資訊，包含其名稱和專業領域。每位醫生可以隸屬於多個診所和部門，這些關係通過 HIRE 關聯表表達，HIRE 表記錄了醫生的雇佣起始時間和結束時間，並確保醫生與診所、部門的雇佣關係是唯一的。此外，TREATMENT 表詳細記錄診所提供的療程，包含治療項目的名稱，並通過外部鍵關聯到醫生、診所和科別，以便於管理和操作。

PERIOD 關聯記錄診所的時段資訊，包括工作日、開始時間和結束時間，以及時段的可用性。這些時段通過 SCHEDULE 關聯進一步分配給醫生和科別，形成具體的排班計劃。SCHEDULE 表記錄排班的詳細資訊，如時段的狀態（可用或不可用）。此外，排班與房間（ROOM）之間存在多對多關係，這些關係通過 ROOM_SCHEDULE 表進行管理。ROOM 表記錄診所內的房間資訊，包括房間名稱、可用狀態，以及當前的隊列更新資訊。

病患的預約行爲通過 APPOINTMENT 和 RESERVATION 表表達。APPOINTMENT 表表示病患對具體時間段的診療預約，包括病患 ID、排班 ID 和日期，並記錄預約的狀態（如待確認、取消等）及是否到診的資訊。RESERVATION 表則記錄病患通過掛號獲得的就診號碼，包含與 TREATMENT 的關聯，用於確保病患的治療需求。APPOINTMENT_REMARK 和 RESERVATION_REMARK 表分別記錄病患對診療和掛號服務的回饋，包含反饋內容和提交時間，以便於診所改善服務質量。

最後，SEARCHING_RECORD 表通過記錄病患的搜尋行爲提供診所推薦和行爲分析的支持，記錄包括搜尋的關鍵字、病患位置（經緯度）、科別和地區資訊。

2.2 Relational Database Schema Diagram

PATIENT 關聯的主鍵是 pid，用於唯一標識每位病患。此表包含病患的核心資訊，包括 pname（病患名稱）、birthdate（出生日期）、gender（性別，值域爲 M 表示男性，F 表示女性）以及 status（會員狀態，值域爲 M 表示會員，G 表示非會員）。這些屬性是醫療管理系統的基礎資料，主鍵 pid 作爲外部鍵被多個其他關聯引用，用於支持病患資料在診療、預約和搜尋中的一致性。

MEMBERSHIP 關聯用於管理病患的帳户資訊，是 PATIENT 的延伸部分。此表以 pid 作爲主鍵，並作爲外部鍵參考到 PATIENT 表的主鍵，以保持病患與帳户的一致性。該表包含 acct_pw（帳户密碼）和 email（電子郵件地址）兩個屬性，提供病患的驗證和聯繫途徑。由於帳户資訊屬於病患的多值屬性，因此需要獨立設計爲一個關聯。

SEARCHING_RECORD 關聯記錄了病患的搜尋行爲，其主鍵是 recordid，確保每條搜尋記錄的唯一性。表內的屬性包括 keywords（搜尋關鍵字）、plat 和 plon（病患所在的經緯度位置）以及 divid（搜尋的科別）。此外，pid 作爲外部鍵參考到 PATIENT 的主鍵，表示每條記錄的發起者。該表支持診所與病患行爲數據的分析，爲個性化推薦提供數據基礎。

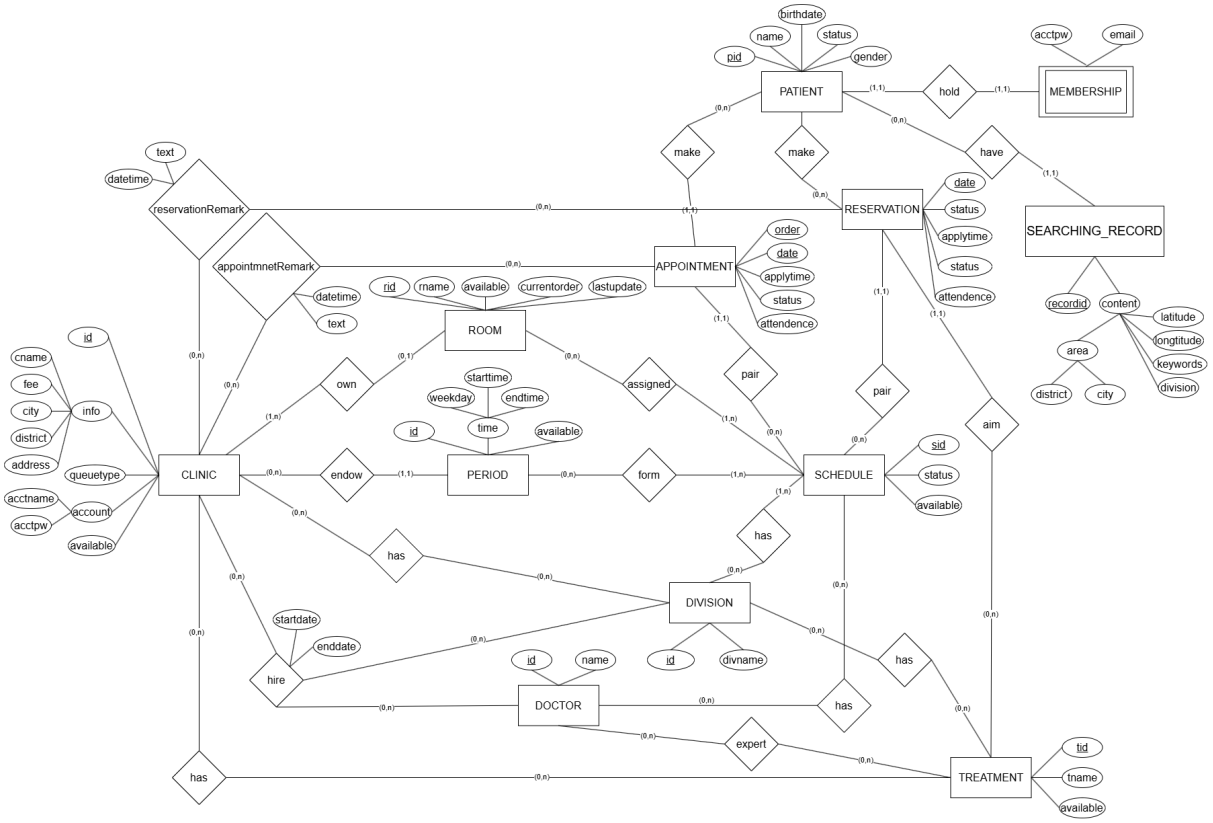

圖 1: 「iclinic」的 ER Diagram

CLINIC 關聯的主鍵是 cid，用於唯一標識每間診所。此表存儲診所的詳細資訊，包括 cname（診所名稱）、address（地址）、city（城市）、district（區域）以及診所的狀態 available（1 表示運營中，0 表示已關閉）。fee（診療費用）用於記錄診所的標準費用，而 queue_type（隊列類型，值域爲 S 表示共享隊列，I 表示獨立隊列）則描述診所採用的候診機制。

DIVISION 關聯的主鍵是 divid，表示診所內的部門分類，例如內科、外科等。此表包含 divname（部門名稱），每個值必須唯一，用於表示科別的專業方向。DIVISION 與診所的結構和醫生的專業技能直接相關，是診所管理的重要部分。

CLINIC_DIVISION 關聯管理診所與部門的多對多關係，其主鍵由 cid 和 divid 組成，用於描述診所內的部門結構。此表還記錄了診所內部的當前隊列順序 queuenumber 和隊列最後更新時間 lastupdate，用於支持診所的排隊系統和實時候診資訊。

DOCTOR 關聯的主鍵是 docid，用於唯一標識每位醫生。此表包含醫生的基本資料，例如 docname（醫生名稱），用於記錄醫生的身份資訊。DOCTOR 表是診所和排班系統的重要組成部分，支持醫療資源的分配和查詢。

HIRE 關聯記錄了醫生與診所之間的雇佣關係，主鍵由 docid 和 cid 組成，表示某位醫生與某診所的唯一雇佣記錄。此表還包含 startdate（雇佣起始日期）和 enddate

(雇佣結束日期)，用於描述醫生的服務時間範圍。HIRE 是管理醫生與診所合作記錄的核心。

PERIOD 關聯的主鍵是 perid，用於描述診所內的時段分配。此表包含 weekday (工作日，值域爲 1 至 7)、starttime 和 endtime (時段的起始與結束時間) 以及 available (時段是否可用)。該表通過 cid 作爲外部鍵參考到 CLINIC 的主鍵，表示每個診所的營業時間結構。

SCHEDULE 關聯的主鍵是 sid，記錄診所的排班資訊，包括 divid (科別)、perid (時段) 和 docid (醫生)。此外，此表還包含屬性 available，用於表示排班是否有效。SCHEDULE 是診所管理工作分配的核心。

ROOM 關聯的主鍵是 rid，用於描述診所的房間資訊，包括 rname (房間名稱) 和 available (房間是否可用)。ROOM_SCHEDULE 則表示房間與排班之間的多對多關係，其主鍵由 rid 和 sid 組成，並記錄每個房間的具體使用情況。

APPOINTMENT 關聯的主鍵由 pid、sid 和 date 組成，記錄病患的診療預約。此表包含 status (預約狀態，如 P 表示待確認，R 表示拒絕)、attendance (是否到診) 和 applytime (申請時間)。此表確保病患的每次診療都有唯一的預約記錄。

RESERVATION 關聯的結構類似於 APPOINTMENT，但額外包含 tid (治療項目)。此表描述病患在診所內的具體治療預約情況，支持診所的治療安排。

APPOINTMENT_REMARK 與 RESERVATION_REMARK 分別記錄病患對診療和治療預約的反饋。其主鍵包括 pid、sid 和 date，並記錄反饋內容 text 和時間 datetime，用於評估醫療服務質量。

TREATMENT 關聯的主鍵是 tid，描述診所內的治療項目。此表包含 tname (治療名稱)，並通過 docid、divid 和 cid 作爲外部鍵參考到醫生、科別和診所的主鍵，用於管理診所內的治療服務。

2.3 Data Dictionary

「iclinic」的資料表共有圖 2 所示的十七個，各個資料表的欄位相關資訊依序呈現在表 1 到表 17。

Column Name	Meaning	Data Type	Key	Constraint	Domain
pid	Patient Id	varchar(10)	PK	Not Null	
pname	Patient Name	varchar(10)		Not Null	
birthdate	Patient's Birthdate	date		Not Null	
gender	Biological Sex, 'M' for Male and 'F' for Female	char		Not Null	{M, F}
status	Account Status, 'M' for Member, 'G' for Non-member	char		Not Null	{M, G}

表 1: 資料表 PATIENT 的欄位資訊

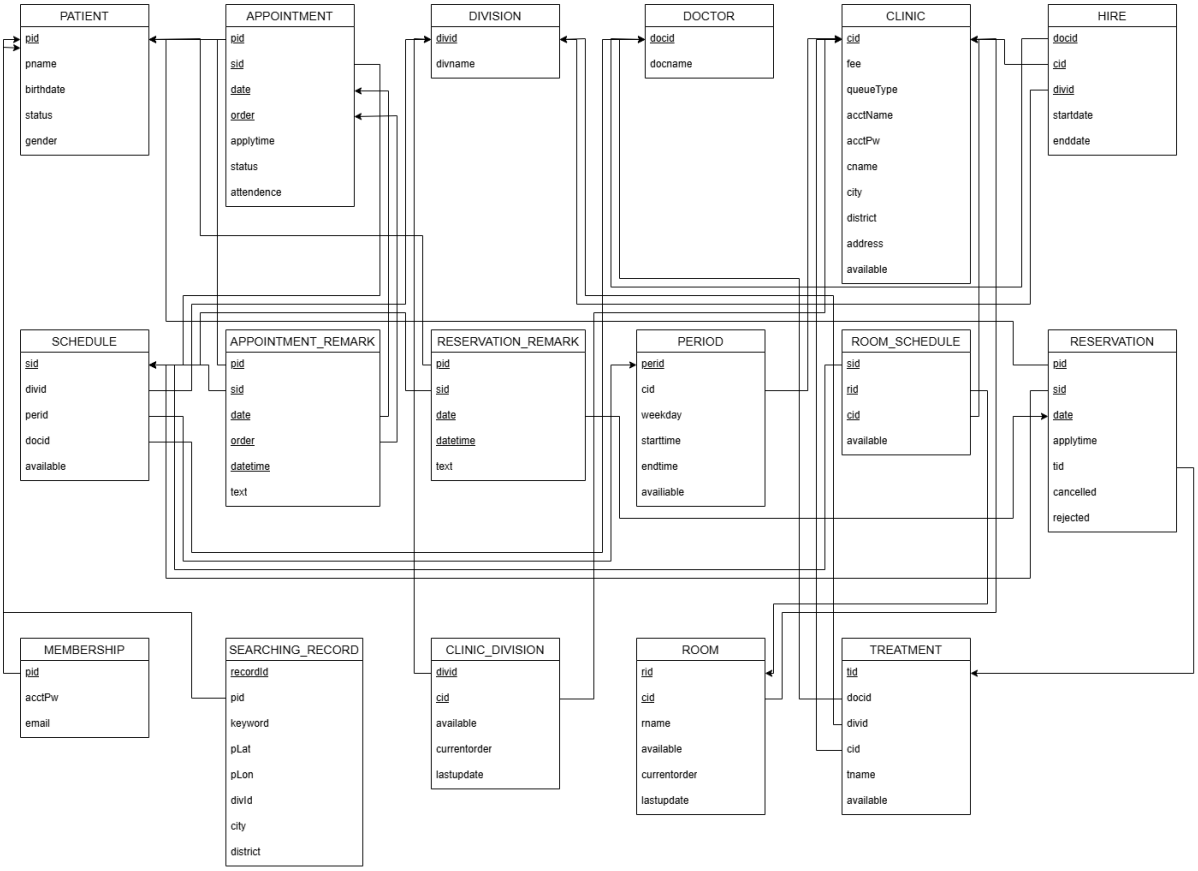

圖 2: 「iclinic」的 Relational Database Schema Diagram

Column Name	Meaning	Data Type	Key	Constraint
pid	Patient Id	varchar(10)	PK, FK: PATIENT(pid)	Not Null
sid	Schedule Id	varchar(20)	PK, FK: SCHEDULE(sid)	Not Null
date	Date of Attendance	date	PK, FK: APPOINTMENT(date)	Not Null
order	Order of the Appointment	int	PK, FK: APPOINTMENT(order)	Not Null
text	Content of the Remark	varchar(4000)		Not Null
datetime	Datetime of Remark Commented	timestamp	PK	Not Null
Referential triggers		On Delete	On Update	
pid: PATIENT(pid)			CASCADE	
sid: SCHEDULE(sid)			CASCADE	
date: APPOINTMENT(date)			CASCADE	
order: APPOINTMENT(order)			CASCADE	

表 2: 資料表 APPOINTMENT_REMARK 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
pid	Patient Id	varchar(10)	PK, FK: PATIENT(pid)	Not Null
sid	Schedule Id	varchar(20)	PK, FK: SCHEDULE(sid)	Not Null
date	Date of Attendance	date	PK, FK: RESERVATION(date)	Not Null
text	Content of the Remark	varchar(4000)		Not Null
datetime	Datetime of Remark Commented	timestamp	PK	Not Null
Referential triggers		On Delete	On Update	
pid: PATIENT(pid)			CASCADE	
sid: SCHEDULE(sid)			CASCADE	
date: RESERVATION(date)			CASCADE	

表 3: 資料表 RESERVATION_REMARK 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
cid	Clinic Id	varchar(10)	PK	Not Null	
fee	Clinic Appointment Fee	int		Not Null	
queue_type	Queue Type of Appointment, 'S' for Shared Queue System and 'I' for Individual one	char		Not Null	{S, I}
acct_name	Clinic Account Name	char		Not Null	
acct_pw	Clinic Password	varchar(30)		Not Null	
cname	Clinic Name	varchar(30)		Not Null	
city	Clinic City	varchar(100)		Not Null	
district	Clinic District	varchar(100)		Not Null	
address	Clinic Address (Exclude City and District)	varchar(1000)		Not Null	
available	If Closed Down	bool		NotNull	

表 4: 資料表 CLINIC 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
docid	Doctor Id	varchar(10)	PK, FK: DOCTOR(docid)	Not Null
cid	Clinic Id	varchar(10)	PK, FK: CLINIC(cid)	Not Null
divid	Division Id	varchar(10)	PK, FK: DIVISION(divid)	Not Null
startdate	Start Date of Hiring	date		
enddate	End Date of Hiring	date		
Referential triggers		On Delete	On Update	
docid: DOCTOR(docid)			CASCADE	
cid: CLINIC(cid)			CASCADE	
divid: DIVISION(divid)			CASCADE	

表 5: 資料表 HIRE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
docid	Doctor Id	varchar(10)	PK	Not Null
docname	Doctor Name	varchar(20)		Not Null

表 6: 資料表 DOCTOR 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
divid	Division Id	varchar(3)	PK, FK: DIVISION(divid)	Not Null
cid	Clinic Id	varchar(20)	PK, FK: CLINIC(cid)	Not Null
available	If Closed Down	bool		NotNull
queuenumber	The Current Order	int		
lastupdate	The Last Update	timestamp		
Referential triggers		On Delete	On Update	
divid: DIVISION(divid)			CASCADE	
cid: CLINIC(cid)			CASCADE	

表 7: 資料表 CLINICDIVISION 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
tid	Treatment Id	varchar(20)	PK	Not Null
docid	Doctor Id	varchar(10)	FK: DOCTOR(docid)	Not Null
divid	Division Id	varchar(3)	FK: DIVISION(divid)	Not Null
cid	Clinic Id	varchar(20)	FK: CLINIC(cid)	Not Null
tname	Treatment Name	varchar(100)		Not Null
available	If Closed Down	bool		NotNull
Referential Trigger		On Delete	On Update	
docid: DOCTOR(docid)			CASCADE	
divid: DIVISION(divid)			CASCADE	
cid: CLINIC(cid)			CASCADE	

表 8: 資料表 TREATMENT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
divid	Division Id	varchar(3)	PK	Not Null	
divname	Division Name	varchar(20)		Not Null, Unique	{ ' 家庭醫學科', ' 內科', ..., ' 中醫痔科' }

表 9: 資料表 DIVISION 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
rid	Room Id	varchar(10)	PK	Not Null
cid	Clinic Id	varchar(20)	PK, FK: CLINIC(cid)	Not Null
rname	Room Name	varchar(5)		Not Null
available	If Closed Down	bool		NotNull
queuenumber	The Current Order	int		
lastupdate	The Last Update	timestamp		
Referential triggers		On Delete	On Update	
cid: CLINIC(cid)			CASCADE	

表 10: 資料表 ROOM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
perid	Period Id	varchar(20)	PK	Not Null	
cid	Clinic Id	varchar(10)	FK: CLINIC(cid)	Not Null	
weekday	Weekday	int		Not Null	[1,7]
starttime	Start Time of one of the Period during the Business Hour	time		Not Null	
endtime	End Time of one of the Period during the Business Hour	time		Not Null	
available	Status of Period, 1 if Available and 0 Otherwise	bool		Not Null	
Referential triggers		On Delete	On Update		
cid: CLINIC(cid)			CASCADE		

表 11: 資料表 PERIOD 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
sid	Schedule Id	varchar(20)	PK	Not Null
divid	Division Id	varchar(3)	FK: DIVISION(divid)	Not Null
perid	Period Id	varchar(10)	FK: PERIOD(perid)	Not Null
docid	Doctor Id	varchar(10)	FK: DOCTOR(docid)	Not Null
available	Status of Schedule, 1 if Available and 0 Otherwise	bool		Not Null
Referential triggers		On Delete	On Update	
divid: DIVISION(divid)			CASCADE	
perid: PERIOD(perid)			CASCADE	
docid: DOCTOR(docid)			CASCADE	

表 12: 資料表 SCHEDULE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
sid	Schedule Id	varchar(20)	PK, FK	Not Null
rid	Room Id	varchar(10)	PK, FK	Not Null
cid	Clinic Id	varchar(10)	PK, FK	Not Null
available	Status of Schedule, 1 if Available and 0 Otherwise	bool		Not Null
Referential triggers		On Delete	On Update	
sid: SCHEDULE(sid)			CASCADE	
rid: ROOM(rid)			CASCADE	

表 13: 資料表 ROOM_SCHEDULE 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint
pid	Patient Id	varchar(10)	PK, FK: PATIENT(pid)	Not Null
$acct_pw$	Member Account Password	varchar(30)		Not Null
email	Member Email	varchar(30)		Not Null, Unique
Referential triggers		On Delete	On Update	
pid: PATIENT(pid)			CASCADE	

表 14: 資料表 MEMBERSHIP 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	
pid	Patient Id	varchar(10)	PK, FK: PATIENT(pid)	Not Null	
sid	Schedule Id	varchar(20)	PK, FK: SCHEDULE(sid)	Not Null	
date	Date of Attendance	date	PK	Not Null	
order	Order of the Appointment	int	PK	Not Null	
applytime	Datetime Applied	timestamp		Not Null	
status	'R' for Rejected, 'P' for Pending, 'C' for Cancelled, 'O' for On Site Application	bool		Not Null	{'P', 'R', 'C', 'O'}
attendance	1 for Attended and 0 otherwise	bool		Not Null	{0, 1}
Referential triggers		On Delete	On Update		
pid: PATIENT(pid)			CASCADE		
sid: SCHEDULE(sid)			CASCADE		

表 15: 資料表 APPOINTMENT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	
pid	Patient Id	varchar(10)	PK, FK: PATIENT(pid)	Not Null	
sid	Schedule Id	varchar(20)	PK, FK: SCHEDULE(sid)	Not Null	
date	Date of Attendance	date	PK	Not Null	
applytime	Datetime Applied	timestamp		Not Null	
tid	Treatment of the Reservation	varchar(20)	FK: TREATMENT(tid)	Not Null	
status	'R' for Rejected, 'P' for Pending, 'C' for Cancelled, 'O' for On Site Application	bool		Not Null	{'P', 'R', 'C', 'O'}
attendance	1 for Attended and 0 Otherwise	bool		Not Null	{0, 1}
Referential triggers		On Delete	On Update		
pid: PATIENT(pid)			CASCADE		
sid: SCHEDULE(sid)			CASCADE		

表 16: 資料表 RESERVATION 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
pid	Patient Id	varchar(10)	FK: PATIENT(pid)	Not Null	
recordid	Record Id	varchar(20)	PK	Not Null	
keyword	Keyword	varchar(30)			
plat	Patient's Latitude	Decimal(8,6)			
plon	Patient's Longitude	Decimal(9,6)			
divid	Division Id	varchar(3)			
city	City	varchar(50)			
district	District	varchar(50)			
Referential triggers		On Delete	On Update		
pid: PATIENT(pid)			CASCADE		

表 17: 資料表 SEARCHING_RECORD 的欄位資訊

2.4 正規化分析

在設計關聯式資料庫時，我們可以檢查資料庫綱目（database schema）是否符合正規化（normalization）的原則。以下將從第一正規式（1NF）到第四正規式（4NF）來逐步說明「iclinic」資料庫的關聯如何符合這些正規化的規範。

首先，1NF 要求每個關聯的屬性必須是單一且不可再分的，換言之，所有屬性應爲 simple 且 single-valued，不允許 composite 或 multi-valued 的屬性存在。以 `TREATMENT` 爲例，它原本在 `DOCTOR, CLINIC, DIVISION` 關聯中應作爲一個 multi-valued 屬性，但目前已透過多筆 tuple 拓展爲單一值，滿足 1NF 的要求。

接著，2NF 要求所有非鍵屬性（non-prime attribute）都必須完全依賴於任何候選鍵（candidate key），即關聯中不能有部分功能相依（partial functional dependency），且此關聯需符合 1NF 的條件。經過檢查，所有的關聯都符合 2NF 的規則。

在 3NF 中，要求關聯中的非鍵屬性不得 transitively 依賴於主鍵。經過分析，目前的設計確實符合 3NF 的規範。在比 3NF 更嚴謹的 BCNF（Boyce-Codd Normal Form）中，規範進一步要求每一個功能相依（functional dependency）中左側的屬性必須是超級鍵（superkey），即在所有 $X \to Y$ 的情況下，X 都必須爲超級鍵。經過檢查，我們的設計完全符合 BCNF 的要求，所有關聯均已避免非超級鍵的部分相依，確保資料庫結構的完整性和高階正規化的標準。

最後，4NF 要求資料庫中的關聯不存在多值相依（multi-valued dependency）。由於「iclinic」的所有關聯均未出現此類相依性，因而符合 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

3.1.1 CLINIC（診所）

CLINIC 資料表來自於健保特約醫事機構-診所公開資料集。我們使用 pandas 模組讀取該檔案，並利用自定義函式 parse_address_simple 將地址解析爲「縣市」、「鄉鎮市區」及「詳細地址」。此外，診所的其他欄位（如診所代碼、名稱、排隊模式等）使用隨機生成器補足。

3.1.2 DIVISIN（科別）

DIVISIN 爲靜態資料，根據醫療機構的標準分類定義，共包含 44 種科別（如內科、外科、骨科等）。資料生成時，爲每個科別分配唯一編號（如 D01 至 D44）及名稱，並將生成的資料插入資料庫。

3.1.3 DOCTOR(醫生) 和 HIRE(聘用)

DOCTOR 採用 Faker 模組生成隨機姓名，並基於現有診所與科別資料表建立關聯性。HIRE 包括醫生所屬診所、科別及起始日期，使用隨機生成器補足其他欄位。

3.1.4 PATIEN(病患)

PATIENT 完全由 Faker 模組生成，包括病患編號、姓名、性別、生日等。生成過程保證資料的唯一性，並通過資料庫會話（Session）插入資料庫。

3.1.5 PERIOD(診療時段) 和 ROOM(診間)

PERIOD 和 ROOM 基於診所資料動態生成。診療時段包含診所代碼、工作日及診療的開始與結束時間，診間則爲每間診所分配若干診療室，生成診間代碼與名稱後與診所表建立關聯。

3.2 功能與對應的 SQL 指令

給 Patient 的功能

3.2.1 創建會員（Create Membership）

此功能用於新增病患及其會員資訊。假設場景爲：「新病患希望成爲會員，系統需要將其基本資訊（如姓名、生日、性別）存儲於病患表中，並將其會員帳號、密碼和電子郵件存儲於會員表中。」系統會檢查並創建對應的資料記錄，確保病患與會員資訊的一致性。

```
-- 在 PATIENT 表中插入病患基本資料
INSERT INTO PATIENT (pid, pname, birthdate, status, gender)
VALUES ('A123456789', 'John Doe', '1990-01-01', 'Active', 'M');

-- 在 MEMBERSHIP 表中插入會員資訊
INSERT INTO MEMBERSHIP (pid, acct_pw, email)
VALUES ('A123456789', 'securepassword123', 'johndoe@example.com');
```

3.2.2 更新會員資料（Update Membership）

此功能用於更新會員的相關資訊（如電子郵件或密碼）。假設場景爲：「會員希望修改其電子郵件地址或密碼，系統需要查詢會員資料，進行更新並保存變更。」系統通過該功能確保會員資料可以動態更新，滿足用户需求。

```
-- 更新 MEMBERSHIP 表中會員的相關資訊
UPDATE MEMBERSHIP
SET email = 'newemail@example.com',
    acct_pw = 'newsecurepassword'
WHERE pid = 'A123456789';
```

3.2.3 建立預約（Appointment）

在建立預約的過程中，系統需要確保分配的 order（排隊序號）是唯一且連續的。爲實現此目標，必須先以 sid（時段 ID）和 date 鎖定相關排班的所有資料，並將這些資料進行分組（GROUP BY）後上鎖（FOR UPDATE）。確保在鎖住的資料範圍內進行更新，避免併發操作導致的競爭或資料不一致。

```
-- 查詢與鎖定現有的預約資料
SELECT sid, date, MAX("order") AS max_order
```

```
FROM APPOINTMENT
WHERE sid = '{sid}' AND date = '{date}'
GROUP BY sid, date
FOR UPDATE;

-- 插入新預約資料
INSERT INTO APPOINTMENT (pid, sid, date, "order", status, applytime,
    attendance)
VALUES ('{pid}', '{sid}', '{date}', {max_order + 1}, '{status}', NOW
   (), '{attendance}');
```

3.2.4 查詢鄉鎮市區診所資訊

此功能目的是根據指定的「鄉鎮市區」來查詢所有位於該地區的診所。

```
SELECT c.*
FROM CLINIC AS c
WHERE c.district = '信義區';
```

3.2.5 以自身經緯度查詢附近診所

此功能讓使用者可以透過提供自己的經緯度以及查詢半徑（例如：5 公里），快速找到距離自己最近的診所，並將結果按照距離由近到遠排序。

```
SELECT c.*,
       ( 6371 * ACOS(
           COS(RADIANS(25.0335)) * COS(RADIANS(c.latitude)) *
           COS(RADIANS(c.longitude) - RADIANS(121.5645)) +
           SIN(RADIANS(25.0335)) * SIN(RADIANS(c.latitude))
       ) ) AS distance
FROM CLINIC AS c
HAVING distance <= 5
ORDER BY distance;
```

給 Clinic 的功能

3.2.6 新增診所

此功能負責在系統中新增一筆診所資料。

```sql
-- 檢查診所帳戶名稱是否已存在
SELECT *
FROM CLINIC
WHERE acct_name = 'clinic_account';

-- 插入診所資料
INSERT INTO CLINIC (cid, cname, acct_name, acct_pw, city, district,
   address, fee, available)
VALUES ('C123456789', 'My Clinic', 'clinic_account', '
   hashed_password', 'Taipei', 'Xinyi', '123 Test Street', 500, TRUE
   );

-- 查詢新增診所代號
SELECT cid
FROM CLINIC
WHERE acct_name = 'clinic_account';
```

3.2.7 查詢預約記錄（Get Appointment）

此功能的目的是根據多個可選條件（如 pid、sid、date、order 等）進行 APPOINTMENT 資料表的查詢。

```sql
SELECT *
FROM APPOINTMENT
WHERE 1=1
    AND (pid = 'PATIENT_ID')
    AND (sid = 'SCHEDULE_ID')
    AND (date = '2024-12-20');
```

3.2.8 建立診所診間（Create Room）

此功能負責新增診所的房間記錄。

```sql
-- 檢查是否已存在相同房間
SELECT *
FROM ROOM
WHERE rid = 'R123456789'
  AND cid = 'C123456789';
```

```sql
-- 若不存在，插入新的房間記錄
INSERT INTO ROOM (rid, cid, rname, available)
VALUES ('R123456789', 'C123456789', 'Consultation Room', true);
```

3.2.9 移除診所內的指定診間（Room）

此功能允許診所的管理員根據診間的唯一識別碼從資料庫中刪除指定的診間記錄。

```sql
-- 查詢診間是否存在
SELECT *
FROM ROOM
WHERE rid = 'R123456789' AND cid = 'C987654321';

-- 移除診間，將指定診間的 available 欄位從 1 轉爲 0
UPDATE ROOM
SET available = 0
WHERE rid = 'R123456789' AND cid = 'C987654321';
```

3.2.10 查詢病患（Get Patient）

此功能旨在根據病患的 ID 或姓名來查詢病患資訊。

```sql
-- 根據 pid 查詢病患記錄
SELECT *
FROM PATIENT
WHERE pid = 'A123456789';

-- 根據 pname 查詢病患記錄
SELECT *
FROM PATIENT
WHERE pname = 'John Doe';
```

3.2.11 創建診間排程（Create Room Schedule）

此功能用於爲診所的診間創建排程資料。

```sql
-- 查詢是否已存在相同的診間排程
SELECT *
FROM ROOMSCHEDULE
WHERE cid = 'C123456789'
```

```
  AND rid = 'R123456789'
  AND sid = 'S123456789';

-- 插入新的診間排程
INSERT INTO ROOMSCHEDULE (cid, rid, sid, starttime, endtime, weekday
    )
VALUES ('C123456789', 'R123456789', 'S123456789', '09:00:00', '
    17:00:00', 'Monday');
```

3.3 SQL 指令效能優化與索引建立分析

3.3.1 在 Get Appointment 中建立索引設計及效能測試

在醫療診所的管理系統中，get_appointment 是非常重要的功能。它負責根據 pid、sid 以及其他條件來查詢預約記錄。這樣的功能在多種情境下被頻繁使用，例如患者想查看自己的預約，診所需要檢索某排班的所有預約記錄。

我們最初爲 pid 和 sid 建立了組合索引，期望提升查詢效率。然而，在效能測試中發現，爲 pid 和 sid 建立索引後，查詢時間並未顯著改善：

```
--爲pid 和 sid建立索引
CREATE INDEX idx_appointment_pid_sid
ON Appointment (pid, sid);
```

- 索引前：5.04s, 5.15s, 5.23s
- 索引後：5.14s, 5.20s, 5.08s

從結果可以看出，爲 pid 和 sid 建立索引並未顯著提升查詢效率。

經進一步分析，我們發現查詢的大部分條件集中於 date 欄位，且查詢範圍較廣。因此，我們嘗試改爲爲 date 建立索引，結果顯示查詢效率有顯著提升：

```
--爲date建立索引
CREATE INDEX idx_appointment_date
ON Appointment (date);
```

- 索引前：14.95s, 15.23s, 23.23s
- 索引後：14.39s, 14.42s, 14.58s

結果顯示，爲 date 建立索引有效縮短了查詢時間，顯著提升了查詢效率，證明在此場景中 date 索引的優化效果優於 pid 和 sid 的索引設計。

3.4 交易管理

在交易管理的部分，本系統主要透過 SQLAlchemy 的 Session 來進行資料庫操作，包括數據的新增、= 查詢與刷新，以及異常處理流程。這些操作可以在 services 資料夾中的多個服務層函數中找到，如 create_or_update_hire、create_clinic 等。

create_or_update_hire 函數是專案中一個關鍵的交易管理功能，用於確保醫生與雇佣關係的資料一致性與完整性。執行該函數時，首先會驗證傳入的醫生 ID 的有效性，這一步是通過 id_validator 函數實現的。如果驗證失敗，create_or_update_hire 函數會立即中止操作並回應錯誤訊息，防止後續操作基於無效的醫生 ID。

接下來，create_or_update_hire 函數會通過調用 get_doctor 函數查詢資料庫，檢查目標醫生的記錄是否已存在。如果記錄不存在，函數會進一步檢查是否提供了醫生的名稱。如果未提供名稱，函數會拋出 HTTPException，要求必須提供名稱才能創建新的醫生記錄。若名稱存在，函數會調用 create_doctor 函數，利用 DoctorCreate 模型生成新的醫生記錄，並將其保存到資料庫。

在確認醫生記錄存在後，create_or_update_hire 函數會嘗試更新雇佣記錄。這一步通過調用 update_hire 函數實現。若目標雇佣記錄已存在，函數會更新雇佣的起始與結束日期並保存變更。如果目標雇佣記錄不存在，函數捕獲 update_hire 拋出的 HTTPException，並繼續執行新增操作。新增操作通過調用 create_hire 函數完成，利用 HireCreate 模型生成一條新的雇佣記錄，記錄醫生與診所、科別的關係，並保存至資料庫。

3.5 併行控制

在我們的系統中，併行控制的設計核心在於避免多個使用者同時操作同一資源時可能引發的資料不一致問題。這類情況在涉及對資料表的新增、更新或刪除操作時尤爲重要。例如，在 services 模組中的各個函數內，當多個使用者同時對系統進行相同或衝突操作時，系統需確保資料的完整性和一致性。

在 membership_create 的場景中，當需要檢查 email 是否已存在，並根據結果進行更新或新增操作時，可能會涉及到資料競爭問題。例如，同時有多個請求嘗試使用相同的 email 進行創建或更新，這可能導致不一致的狀況。通過在檢查 email 存在性時加鎖，可以確保只有一個任務能對這一行進行變更操作。這避免了多個請求同時檢查結果並新增相同記錄的情況，確保資料的唯一性和一致性。

在處理 appointment（預約）功能時，系統可能需要從診所的 queue_num 中分配一個新的排隊號，並在完成預約後更新排隊號。這裡的核心問題是，如何確保多個請求同時發起預約時，分配的 queue_num 是唯一且連續的。通過在獲取 queue_num 時加鎖，可以確保多個請求無法同時讀取並修改該值，避免了分配重複號碼的情況。加鎖還能確保在更新 queue_num 的同時提交新的預約記錄，不會出現數據不一致的問題。

4 分工資訊

朱柏諺: 系統設計、功能設計、ER Diagram 設計、Data Dictionary 設計、正規化分析、前後端架構設計、後端開發、前端開發、簡報製作、demo 影片錄製。

尤喆: 系統設計、功能設計、ER Diagram 設計、Relational Database Schema Diagram 繪製、Data Dictionary 設計、測試資料產生、後端開發、前端開發、SQL indexing 設計。

林至偉: 系統設計、功能設計、ER Diagram 設計、Data Dictionary 設計、Relational Database Schema 說明、SQL 指令。

5 專案心得

朱柏諺: 這次的專案心得就是「事前繁瑣的規劃可以避免事後的雜亂無章」，由於缺乏開發前後端的經驗，所以很多該定義的都沒有預先定義也沒有討論清楚，包含 request 的命名、schema 甚至是 models 的建構一開始都不知道可以直接從資料庫輸出。不過這一次在這方面做得比較好的是花了我們大量時間的 tables 設計，直到最後都沒有出現致命的問題，都至少正確雖然不見得有效率。這次有很多進階功能還沒能實作出來，但建構好的資料庫結構完全可以負擔這些進階功能。相信有了這次經驗往後開發類似專案會走更少彎路。

尤喆: 這次的期末專案有很多挑戰，我認爲最大的是對於架構的不熟悉，雖然在初期討論時我們有很多想法，也增加了很多功能，但隨著開始實作，我們漸漸發現有些東西的設計不太方便，甚至是有許多功能在後端寫好了卻發現在撰寫前端實根本用不到，甚至必須寫新的檔案來彌補設計的不足。另外還有對於文法的標準也很重要，例如檔案該如何命名，因爲在多人協力的環境下，檔案有沒有加 s、加底線、大小寫都會造成很多對接的 bug，所以在一開始就定義好文法很重要。最後是關於 db 的設計，有些 id 是我們到了眞正使用時才發現不需要。

林至偉: 經過此次專案我對資料庫的設計和建構有了很深的了解，前期在設計系統、功能、tables 就耗費了我們極長的時間，設計出一個好的資料庫是需要經過鎮密的規劃，並根據要開發的功能做出調整，我們小組遇到最大的問題便是沒有完整的規劃，導致後期實作時遇到許多定義、命名的問題，前期設計了非常多的功能卻有不少沒有實現，爲它們設計的 tables 也沒用到。由於沒有寫前端的經驗，我們決定先完成後端再開始寫前端，然而卻碰上許多問題，我們認爲應該也要先設計好前端，才能使專案的進行更加順利，經過這次經驗我未來會更加注意對整個專案的規劃和安排。