資料庫管理（113-1）
期末專案完整報告範例

第 29 組
B11106001 李亮節、B11705016 黃子維、B11705029 劉芊儀
2024 年 12 月 10 日
GitHub 專案連結、展示影片連結

1 系統分析

當你聽到「補習班管理」這四個字，是否總會聯想到一堆凌亂的學生資料、複雜的家長聯絡方式、混亂的課程排程和老師的時間表難以協調？每次報名、查詢學習進度時，資料總是散落在不同的紙本文件和電子檔案中，讓人難以快速找到所需的資訊。老師往往需要花費大量時間翻閱堆積如山的資料，才能回應家長的問題，這樣的場景不僅讓管理人員頭痛，也使家長和老師感到不便。

這些管理上的瓶頸不僅影響了補習班的運營效率，更有可能牽涉到教學質量的下降，甚至影響到學生的學習體驗。爲了徹底改善這些問題，我們針對數學補習班 MathSchool 設計了一套全新的資料管理系統，旨在簡化繁瑣的手動管理流程，提高辦公效率，並優化學習體驗。

這套系統的核心功能是集中管理學生資料，讓所有個人資訊、報名記錄、課程安排等等內容在一個平台中輕鬆存取，避免繁瑣的資料查找與更新，並能即時查看資料。排課系統則簡化了老師的時間表管理，將課程、教室和學生資訊進行集中儲存和調整，顯著提升排課效率，減少手動調整所需時間。爲進一步提升學習體驗，專門設有諮詢「mentor」服務，提供即時解答與針對性建議。此外，系統界面簡單直觀，操作無需繁瑣培訓，無論管理人員、老師還是家長都能輕鬆上手，確保日常管理和臨時調整均能快速完成。總體而言，這套系統能全面提升補習班的運營效率、簡化管理流程，並改善學習體驗，讓您專注於提升教學質量，達成更好的學習成果。

1.1 系統功能

1.1.1 給 Workers 的功能

在本系統中，Workers 可以執行以下功能：

1. 新增學生資料：Workers 可以通過輸入學生的名稱、學校、電話、地址等資料來新增學生。新增後，系統會自動分配一個唯一的學生編號以便後續識別。
2. 查詢學生資料：Workers 可通過學生的姓名或編號來查詢學生資料，快速檢索並查看學生的詳細資訊。

3. 更新學生資料：Workers 可對已新增的學生資料進行更新，包括學生的聯絡方式、學校等基本信息，並實時反映在系統中。

4. 新增、查詢、更新教師資料：Workers 可新增、查詢或更新教師的基本資料，管理教師的姓名、授課科目等信息。

5. 新增、查詢、更新 mentor 資料：Workers 可新增、查詢或更新導師（mentor）的資料，並管理其負責的學生及服務信息。

6. 新增、查詢、更新試聽（audit）資料：Workers 可新增、查詢或更新試聽紀錄，監控系統內的操作行爲，保證管理過程中的資料安全與審核合規。

7. 管理教室使用：Workers 可安排並管理各班的上課教室，確保教室使用效率和課程安排的合理性。

8. 管理學生課表：Workers 可管理學生的課程安排，確定學生參加的各個班級，並確保班級和老師時間表的協調。

9. 管理輔導老師班表：Workers 可管理老師的授課班級及排課信息，確保老師的時間表與課程安排的精確對接。

1.1.2 給 Student 的功能

在本系統中，Student 可以執行以下功能：

1. 查詢 mentor shift 資料：Student 可查詢輔導老師（mentor）的排班資訊，包括導師的上班時間。此功能有助於學生了解導師的可用時間，並安排相應的學習計劃。

2. 查詢課程資料：Student 可查詢班級（Class）、教師（TEACH) 的資訊。此功能有助於學生了解自己的上課相關資訊，並安排相應的學習計劃。

1.1.3 給 Mentor 的功能

在本系統中，Mentor 可以執行以下功能：

1. 查詢 mentor shift 資料：Mentor 可查詢自身的排班資料，包括每日的工作時間、輔導時段，確保時間管理的有效性和準確性。

2 系統設計

2.1 ER Diagram

圖 1 是「MatchSchool」的 ER Diagram，在這個 ERD 中共有七個實體（entity），分別是 STUDENT、TEACHER、CLASS、CLASSROOM、MENTOR、WORKERS和AUDIT，以及兩弱實體（weak entity)，分別是分別是 PARENT和MENTOR_SHIFT，以及五個關係（relationship），包括 TAKE、USE、HAS、TEACH和TAKE[1]。其中 WORKERS 代表的是擁有所有系統權限的行政人員，任何查詢和修改的功能都可以使用，包含先前提及的，修改學生、老師、課程、試聽生、輔導老師和班級資料。

- **STUDENT（學生）**
  描述：包含學生的基本資訊，如學生編號（student_id）、姓名（student_name）、學校（school）、年級（grade）、電話（tel）、地址（address）和狀態（status）。
  關係：
  - 與家長的 **HAS** 關係（0,N）。
  - 與課程的 **TAKE** 關係（0,N），表示學生可以選修多門課程。
- **PARENT（家長）**
  描述：記錄學生家長的相關資訊，包括家長姓名（parent_name）和聯絡電話（parent_tel）。
  關係：家長和學生之間有監護關係，可對應到該學生的資料。
- **TEACHER（老師）**
  描述：記錄教師的相關資訊，包括教師編號（teacher_id）、姓名（teacher_name）、電話（tel）、地址（address）和狀態（status）。
  關係：
  - 與課程的 **TEACH** 關係（1,N），表示每位老師可以教授多門課程。
- **CLASSROOM（教室）**
  描述：記錄教室的相關資訊，包括教室編號（classroom_id）、名稱（classroom_name）和容納人數（capacity）。
  關係：
  - 與課程的 **USE** 關係（1,1），每門課程都會分配到一間教室。

[1]其中一個 TAKE 是學生上課，另外一個 TAKE 是輔導老師上班

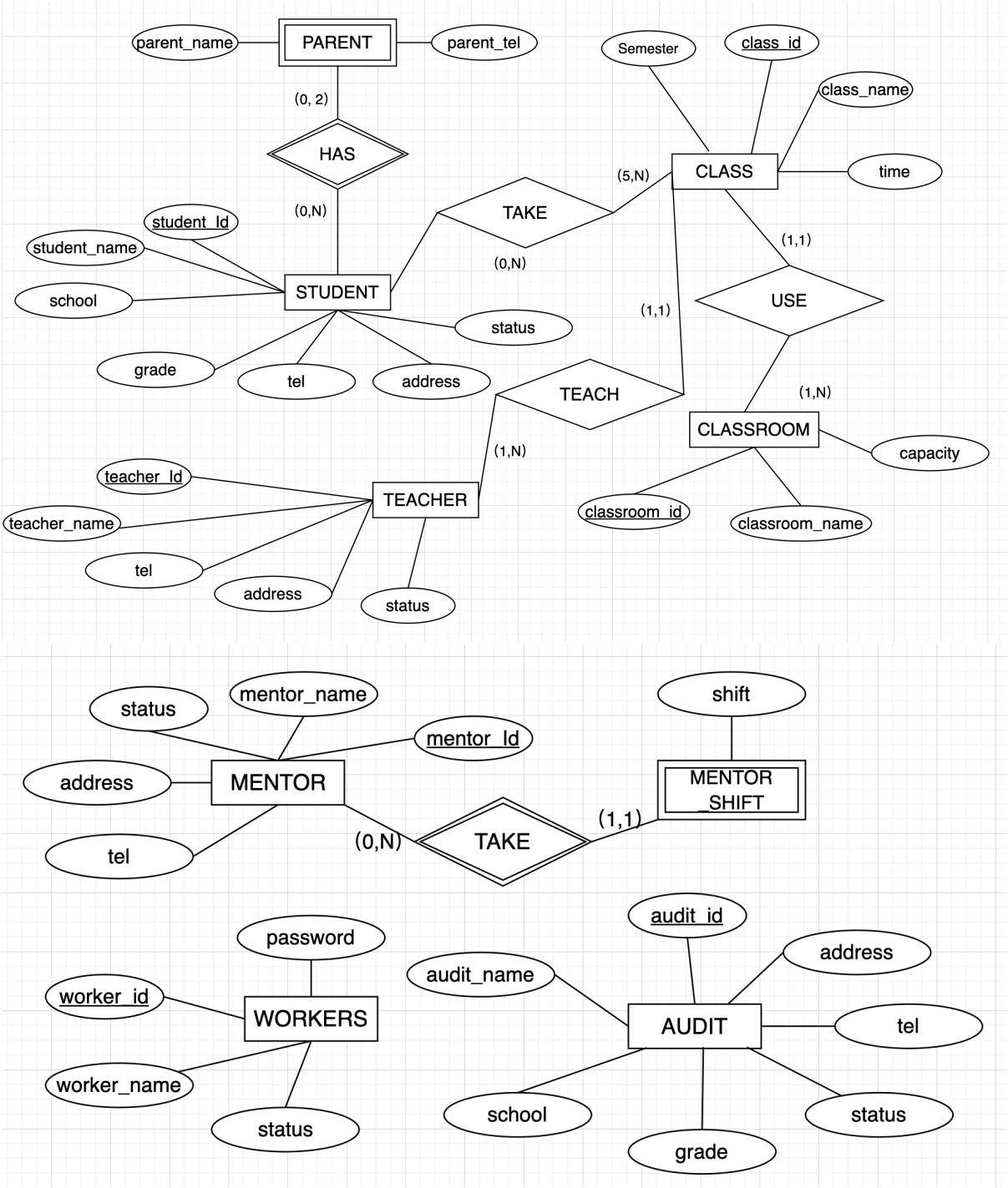

圖 1: 「I'm in」的 ER Diagram

- **CLASS（課程）**

  描述：記錄課程的相關資訊，包括課程編號（class_id）、名稱（class_name）、學期（semester）、上課時間（time）以及使用的教室編號（classroom_id）。

  關係：

– 與學生的 **TAKE** 關係（5,N），每門課程至少有 5 位學生選修。

– 與教師的 **TEACH** 關係（1,N），由多位教師教授。

- **MENTOR（輔導員）**
  描述：包含輔導員的基本資訊，如輔導員編號（mentor_id）、姓名（mentor_name）、電話（tel）、地址（address）和狀態（status）。
  關係：

  – 與工作班表的 **MENTOR_SHIFT** 關係（0,N），記錄輔導員的排班情況。

- **WORKERS（工作人員）**
  描述：記錄工作人員的基本資訊，包括工作人員編號（worker_id）、姓名（worker_name）、狀態（status）和密碼（password）。

- **AUDIT（試聽者）**
  描述：記錄試聽學生的基本資訊，包括試聽編號（audit_id）、姓名（audit_name）、學校（school）、年級（grade）、電話（tel）、地址（address）和狀態（status）。

- **MENTOR_SHIFT（輔導員班表）**
  描述：記錄輔導員的排班資訊，包括班表編號（shift）和對應的輔導員編號（mentor_id）。
  關係：

  – 與 **MENTOR** 關聯（1,1），每個班表記錄對應的輔導員。

- **TAKE（選修）**
  描述：表示學生與課程的選修關係，記錄學生編號（student_id）和課程編號（class_id）。
  關係：

  – 與 **STUDENT** 關聯（0,N）。

  – 與 **CLASS** 關聯（5,N）。

- **TEACH（授課）**
  描述：表示教師與課程的授課關係，記錄教師編號（teacher_id）和課程編號（class_id）。
  關係：

  – 與 **TEACHER** 關聯（1,N）。

  – 與 **CLASS** 關聯（1,N）。

2.2 Relational Database Schema Diagram

2.2.1 基本設計

我們可以將圖 1 的 ER Diagram 轉換成圖 2 的 Database Schema Diagram，一共有五個關聯（relation），分別是 `TAKE`、`USE`、`HAS`、`TEACH`和`TAKE`。

STUDENT 這個關聯的主鍵（Primary key, PK）是 student_id，該屬性被定義爲唯一識別碼，負責標示每位學生的身份。其他欄位如 student_name（學生姓名）爲非空屬性（NOT NULL），表示每位學生必須有名稱；school（學校）、grade（年級）、tel（電話）、address（地址）、status（狀態）則用來記錄學生的其他詳細資訊。這張表的外部鍵（Foreign key, FK）parent_id 原設計參考 PARENT 的主鍵 parent_id，並以 ON DELETE CASCADE 方式處理，但目前被註解掉。未來若啓用此設計，每位學生將能對應到一位家長，刪除家長時其相關聯的學生資料將自動刪除。

PARENT 這個關聯的主鍵由 student_id（學生編號）組成，同時作爲外部鍵參考 STUDENT 的主鍵 student_id，並定義爲 ON DELETE CASCADE，即當學生資料被刪除時，對應的家長資料也會自動刪除。其他屬性如 parent_name（家長姓名）和 parent_tel（家長電話）則是必要欄位（NOT NULL），用來記錄家長的基本資訊。

STUDENT 這個關聯的主鍵（Primary key, PK）是 student_id，該屬性被定義爲唯一識別碼，負責標示每位學生的身份。其他欄位如 student_name（學生姓名）爲非空屬性（NOT NULL），表示每位學生必須有名稱；school（學校）、grade（年級）、tel（電話）、address（地址）、status（狀態）則用來記錄學生的其他詳細資訊。這張表的外部鍵（Foreign key, FK）parent_id 原設計參考 PARENT 的主鍵 parent_id，並以 ON DELETE CASCADE 方式處理，但目前被註解掉。未來若啓用此設計，每位學生將能對應到一位家長，刪除家長時其相關聯的學生資料將自動刪除。

PARENT 這個關聯的主鍵由 student_id（學生編號）組成，同時作爲外部鍵參考 STUDENT 的主鍵 student_id，並定義爲 ON DELETE CASCADE，即當學生資料被刪除時，對應的家長資料也會自動刪除。其他屬性如 parent_name（家長姓名）和 parent_tel（家長電話）則是必要欄位（NOT NULL），用來記錄家長的基本資訊。

TEACHER 這個關聯的主鍵是 teacher_id，該屬性用於唯一標示教師身份。屬性包括 teacher_name（教師姓名）、tel（電話）、address（地址）和 status（狀態），用來記錄教師的詳細資訊。這張表目前沒有設置外部鍵。

CLASSROOM 這個關聯的主鍵是 classroom_id，負責標示教室的唯一身份，屬性如 classroom_name（教室名稱）和 capacity（容量）爲必填欄位（NOT NULL），用於記錄教室的基本資訊。

CLASS 這個關聯的主鍵是 class_id，屬性包括 class_name（課程名稱）、semester（學期）、time（上課時間）等。外部鍵 classroom_id 參考 CLASSROOM 的主鍵 classroom_id，並定義爲 ON DELETE SET NULL，即當教室被刪除時，課程的 classroom_id 將被設爲空值。

MENTOR 這個關聯的主鍵是 mentor_id，屬性包括 mentor_name（輔導員姓名）、tel（電話）、address（地址）和 status（狀態），用於記錄輔導員的詳細資訊。

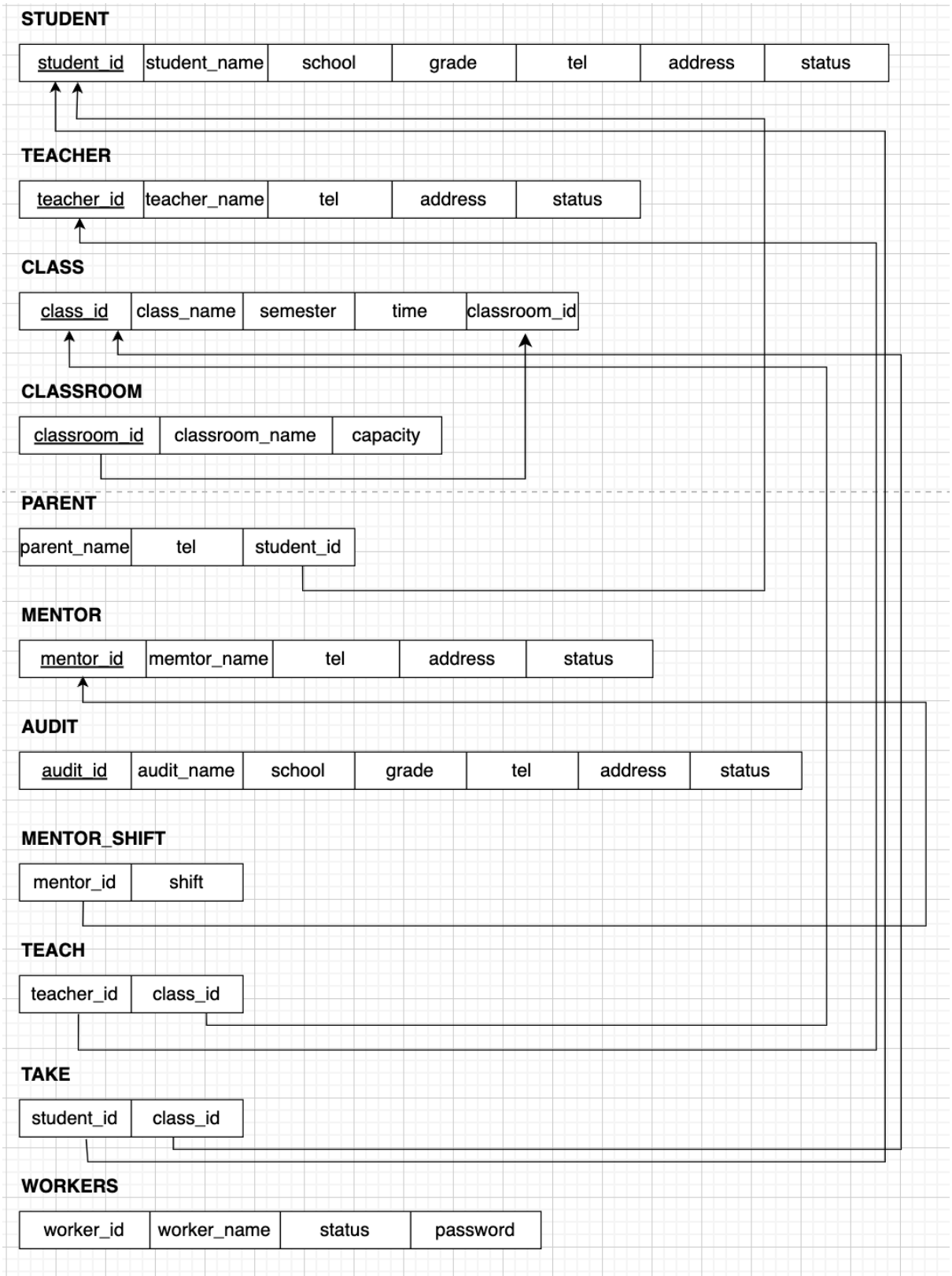

圖 2: 「I'm in」的 Relational Database Schema Diagram

WORKERS 這個關聯的主鍵是 worker_id，其屬性 worker_name（工作人員姓名）、status（狀態）和 password（密碼）用於記錄每位工作人員的基本資訊和系統認證資訊。

AUDIT 這個關聯的主鍵是 audit_id，屬性包括 audit_name（試聽者姓名）、school（學校）、grade（年級）、tel（電話）、address（地址）和 status（狀態），用於記錄試聽者的詳細資訊。

MENTOR_SHIFT 這個關聯的主鍵是 shift（班表），並且包含外部鍵 mentor_id，參考 MENTOR 的主鍵 mentor_id，並以 ON DELETE CASCADE 定義，當輔導員資料被刪除時，對應的班表也會一併刪除。

TAKE 這個關聯是由學生與課程的多對多關係型態產生而來。該關聯的主鍵由兩個外部鍵組成，分別是 student_id，參考 STUDENT 的主鍵，和 class_id，參考 CLASS 的主鍵。這兩個外部鍵的刪除行爲均設定爲 ON DELETE SET NULL，以防止聯動刪除。

TEACH 這個關聯是由教師與課程的多對多關係型態產生而來。該關聯的主鍵由 teacher_id 和 class_id 組成，分別參考 TEACHER 和 CLASS 的主鍵，並且都定義爲 ON DELETE CASCADE，當教師或課程資料刪除時，對應的關係記錄也會自動刪除。

COURSE 這個關聯的主鍵是 Course_id，一樣是被定義爲系統自動增加的唯一識別碼，剩餘的屬性則包含教室名稱、授課教師姓名、開設對象及課程時間這些 COURSE 實體下的屬性。最後是 CLASSROOM 關聯，該關聯的主鍵是 Classroom_id，被定義爲系統自動增加的唯一識別碼，其餘屬性則爲 CLASSROOM 實體下的屬性包括教室所在的建物名稱、教室所在樓層、教室名稱及教室容納人數上限。

2.3 Data Dictionary

「MathSchool」的資料表共有圖 2 所示的 11 個，各個資料表的欄位相關資訊依序呈現在下方。

Column Name	Meaning	Data Type	Key	Constraint	Domain
student_id	學生代號	varchar(100)	PK	Not Null	
student_name	學生姓名	varchar(100)		Not Null	
school	就讀學校	varchar(100)			
grade	年級	varchar(50)			
tel	電話	varchar(20)			
address	地址	varchar(255)			
status	狀態	varchar(50)			
Referential triggers		On Delete	On Update		
student_id: PARENT(student_id)		Cascade			

表 1: 資料表 STUDENT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
parent_name	家長姓名	varchar(100)		Not Null	
parent_tel	家長電話	varchar(20)		Not Null	
student_id	學生代號	varchar(100)	FK: STUDENT(student_id)		
Referential triggers		On Delete	On Update		
student_id: STUDENT(student_id)		Cascade			

表 2: 資料表 PARENT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
teacher_id	教師代號	varchar(20)	PK	Not Null	
teacher_name	教師姓名	varchar(100)		Not Null	
tel	電話	varchar(20)			
address	地址	varchar(255)			
status	狀態	varchar(20)			

表 3: 資料表 TEACHER 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
classroom_id	教室代號	varchar(20)	PK	Not Null	
classroom_name	教室名稱	int		Not Null	
capacity	容量	int		Not Null	

表 4: 資料表 CLASSROOM 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
class_id	課程代號	varchar(10)	PK	Not Null	
class_name	課程名稱	varchar(100)		Not Null	
semester	學期	varchar(20)			
time	時間	time			
classroom_id	教室代號	varchar(20)	FK: CLASSROOM(classroom_id)		
Referential triggers		On Delete	On Update		
classroom_id: CLASSROOM(classroom_id)		Set Null			

表 5: 資料表 CLASS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
teacher_id	教師代號	varchar(20)	FK: TEACHER(teacher_id)		
class_id	課程代號	varchar(10)	FK: CLASS(class_id)		
Referential triggers		On Delete	On Update		
teacher_id: TEACHER(teacher_id)		Cascade			
class_id: CLASS(class_id)		Cascade			

表 6: 資料表 TEACH 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
mentor_id	輔導老師代號	int	PK	Not Null	
mentor_name	輔導老師姓名	varchar(100)		Not Null	
tel	電話	varchar(20)			
address	地址	varchar(255)			
status	狀態	varchar(50)			

表 7: 資料表 MENTOR 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
worker_id	行政人員代號	varchar(20)	PK	Not Null	
worker_name	行政人員姓名	varchar(100)		Not Null	
status	狀態	varchar(50)			
password	密碼	varchar(100)			

表 8: 資料表 WORKERS 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
audit_id	試聽代號	varchar(100)	PK	Not Null	
audit_name	試聽者姓名	varchar(100)		Not Null	
school	就讀學校	varchar(100)			
grade	年級	varchar(50)			
tel	電話	varchar(20)			
address	地址	varchar(255)			
status	狀態	varchar(50)			

表 9: 資料表 AUDIT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
shift	輔導員班表	int			
mentor_id	輔導員代號	int	FK: MENTOR(mentor_id)		
Referential triggers		On Delete	On Update		
mentor_id: MENTOR(mentor_id)		Cascade			

表 10: 資料表 MENTOR_SHIFT 的欄位資訊

Column Name	Meaning	Data Type	Key	Constraint	Domain
student_id	學生代號	varchar(100)	FK: STUDENT(student_id)		
class_id	課程代號	varchar(10)	FK: CLASS(class_id)		
Referential triggers		On Delete	On Update		
student_id: STUDENT(student_id)		Set Null			
class_id: CLASS(class_id)		Set Null			

表 11: 資料表 TAKE 的欄位資訊

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「MathSchool」的關聯是如何滿足這些規則。

第一正規化 (1NF)

所有資料表的欄位均存儲單一值，例如：

- STUDENT 的 student_name 是一個單一值。

沒有重複組或多值屬性存在，因此符合第一正規化的要求。

第二正規化 (2NF)

- 對於單一主鍵的資料表（例如 STUDENT、TEACHER 等），每個非主鍵屬性都完全依賴於主鍵。
- 對於聯合主鍵的資料表（例如 TAKE 和 TEACH），所有非主鍵屬性（如果存在）完全依賴於聯合主鍵。
- 例如：TAKE 中，student_id 和 class_id 是聯合主鍵，且沒有其他非主鍵屬性，因此符合 2NF。

第三正規化 (3NF)

- STUDENT 表中的非主鍵屬性（如 school 和 grade）僅依賴於主鍵 student_id，不存在非主鍵到非主鍵的依賴。
- PARENT 表中的 parent_name 和 parent_tel 僅依賴於主鍵 student_id。

結論：資料庫滿足 3NF。

巴斯-柯德正規化 (BCNF)

- 所有資料表中的非主鍵屬性均依賴於主鍵，且主鍵是唯一候選鍵。
- 沒有非主鍵屬性作爲決定因素，因此符合 BCNF 的要求。

第四正規化 (4NF)

- 所有資料表中不存在多值屬性，因此不會出現多值依賴。例如：
  - CLASS 表中的 classroom_id 是一個外部鍵，不形成多值依賴。
  - TAKE 表中也沒有多值屬性。

本資料庫滿足這些正規化條件。

3 系統實作

3.1 資料庫建置方式及資料來源說明

關於 STUDENT、PARENT 、TEACHER、MENTOR、AUDIT 資料表內的資料建置方式，我們首先利用姓名隨機生成器一次各生成 30,000 筆的姓名資料，接著用相同的方式隨機生成符合現實情況的台灣地址、並隨機選取學生所就讀的高中。我們將這些.txt 檔案匯入爲.csv 檔，然后利用 Python 建立符合預期中資料表格式的 tables。接下來，我們假設數學補習班擁有橫跨三年的營業歷史，因此在進行學生編號時，透過報名的年度作爲編號前綴，接下來則依序紀錄這學期入學的人數。這樣的 Primary key 既具備唯一性質，又可以讓使用者清楚理解這樣編碼的邏輯。而假設學生都會填一位主要監護人（家長），所以家長資料所對應的學生 id 也具有同樣的唯一性。在學生紀錄 status 的部分，我們假設報名的人數中有一成的人已經離開，因此使用 random 生成 10% 的學生狀態列標註「已離開」，以增加資料的眞實性。

完成基本資料表格的建立後，我們進一步處理這些表格之間的關聯。我們用 Python 寫下這些實體表格間 relationship 生成表示其關聯的表格，如TEACH、TAKE等。這些表格因爲是來源於前面提及的實體之關係，所以我們仔細考量了上個段落生成的 foreign key，根據學生的年級，給沒有標注「已離開」的學生安排到該年級的班級上。

資料庫架構和表格名目來自北市某數學補習班的原設定系統，爲保護學生個資隱私我們不得獲得眞實數據內容，不過爲了眞實還原系統內部需求和針對不同使用者，我們對補習班行政人員進行訪問和交流，來增加我們系統設計的合理性和完整度。

3.2 重要功能及對應的 SQL 指令

接下來我們會介紹系統中的重要功能和其對應的 SQL 指令：

3.2.1 給 Workers 的功能介紹

1. 登入系統：由於只有 workers 才有編輯資料庫內資料的權限，我們需要讓 workers 有使用帳號密碼登入系統的機制：

```
SELECT worker_id, password, worker_name, status
        FROM WORKERS
        WHERE worker_id = ?
```

Listing 1: 登入系統 SQL 指令

新增、修改、查詢學生及家長資料：學生可能在任何時間點報名該補習班課程，或是針對該名學生的各種需求去進行資料的編輯，開放給行政人員在與學生或學生家長進行接洽時的編輯功能。以下以查詢學生資料的 SQL 程式碼爲範例，因爲涉及兩個

表STUDENT和PARENT的內容，所以需要用LEFT JOIN來實現這樣的需求，當然也不是每個學生家長都想留下資料，故我們不會在新增資料的時候要求不能空白：

```
SELECT
            s.student_id, s.student_name, s.school,
            s.grade, s.tel, s.address, s.status,
            p.parent_name, p.parent_tel
        FROM STUDENT s
        LEFT JOIN PARENT p ON s.student_id = p.student_id
        WHERE s.student_name LIKE ? OR s.student_id LIKE ?
```

Listing 2: 查詢學生及家長資料 SQL 指令

3. 新增、修改、查詢教師資料：會有教師新加入補習班，也會有教師離開補習班，故我們需要讓行政人員有這樣的功能權限來維持教師的資料，至於教師離開此補習班，我們則會用將 status 改爲"off" 的方法來邊標記該位教師目前已離開補習班。以下將以INSERT作爲此功能的 SQL 範例程式碼：

```
INSERT INTO TEACHER (teacher_id, teacher_name, tel, address,
   status)
VALUES (?, ?, ?, ?, ?)
```

Listing 3: 新增教師資料 SQL 指令

4. 新增、修改、查詢輔導老師資料：每一年都會有輔導老師新加入補習班，也會有輔導老師不再排班，故我們需要讓行政人員有這樣的功能權限來維持輔導老師的資料，至於輔導老師永遠不在此補習班排班，我們則會用將 status 改爲"off" 的方法來邊標記該位輔導老師目前不再於此補習班任職。另外，爲了資料實用性，我們只會選擇目前正在服務的輔導老師做顯示，來應應行政人員大概率只會編輯在職輔導老師資料的需求：

```
SELECT
            m.mentor_id, m.mentor_name, m.tel,
            m.address, m.status, ms.shift
        FROM MENTOR m
        LEFT JOIN MENTOR_SHIFT ms ON m.mentor_id = ms.mentor_id
        WHERE m.status = 'on'
```

Listing 4: 顯示輔導老師資料 SQL 指令

5. 修改、查詢輔導老師排班資料：輔導老師身份皆爲學生，會在各種時候需要進行調班、換班、代班的行爲，故給行政人員調整輔導老師班表的功能權限，以便在確認班表調整的情況後給予最即時的輔導老師班表狀況，因爲輔導老師的資訊在MENTOR表當中，所以會需要JOIN此表MENTOR_SHIFT來顯示我們需要進行編輯的資料：

```
SELECT
        MENTOR_SHIFT.shift, MENTOR_SHIFT.mentor_id,
        MENTOR.mentor_name
    FROM MENTOR_SHIFT
    LEFT JOIN MENTOR ON MENTOR_SHIFT.mentor_id = MENTOR.
       mentor_id
    ORDER BY MENTOR_SHIFT.shift ASC
```

Listing 5: 查詢輔導老師排班 SQL 指令

6. 新增、修改、查詢課程資料：因應每學年都會在課程的細節上進行更動，我們設計讓行政人員有修改課程相關事項的權限，以便維護整個補習班所有課程的規劃和運作。另外，因爲我們將課程存放於CLASS，將教師的資料存放於TEACHER以及教師授課的資料存放於TEACH，故需要顯示課程所有資料時會需要 JOIN 多張表格：

```
SELECT
        c.class_id, c.class_name, c.time, c.semester, c.
           classroom_id, t.teacher_id, tr.teacher_name
    FROM CLASS c
    LEFT JOIN TEACH t ON c.class_id = t.class_id
    LEFT JOIN TEACHER tr ON t.teacher_id = tr.teacher_id
    WHERE c.class_name LIKE ?
```

Listing 6: 查詢課程資料 SQL 指令

7. 查詢指定課程學生資料：在課程編輯的頁面，我們也提供額外的按鈕讓行政人員來查看該課程目前上課中的學生有哪些，以便做管理：

```
SELECT
        STUDENT.student_id, STUDENT.student_name,
        STUDENT.school, STUDENT.grade,
        STUDENT.tel, STUDENT.address,
        STUDENT.status, PARENT.parent_name,
        PARENT.parent_tel
    FROM TAKE
```

```
    INNER JOIN STUDENT ON TAKE.student_id = STUDENT.student_id
    LEFT JOIN PARENT ON STUDENT.student_id = PARENT.student_id
    WHERE TAKE.class_id = ?
```

Listing 7: 查詢課程學生資料 SQL 指令

8. 新增、修改、查詢試聽生資料：每天都有可能會有試聽生來補習班參觀和試聽課程，那麼我們爲了追蹤該試聽生後來的決定和其聯絡方式以持續追蹤來增加補習班效益，我們提供行政人員這樣的功能來提升學生數量的績效：

```
INSERT INTO AUDIT (audit_id, audit_name, school, grade, tel,
   address, status)
VALUES (?, ?, ?, ?, ?, ?, ?)
```

Listing 8: 新增試聽生 SQL 指令

3.2.2 給 Student 的功能

1. 查看課程資訊：學生可在此頁面查看其報名課程的相關資訊，包含上課教室、授課教師，來做確認或相關事宜，爲方便使用者閱覽，我們只會顯示該學期正在教授中的課程的相關資訊：

```
SELECT
        CLASS.class_id, CLASS.class_name,
        CLASS.semester, CLASSROOM.classroom_name,
        TEACHER.teacher_name
    FROM CLASS
    LEFT JOIN CLASSROOM ON CLASS.classroom_id = CLASSROOM.
       classroom_id
    LEFT JOIN TEACH ON CLASS.class_id = TEACH.class_id
    LEFT JOIN TEACHER ON TEACH.teacher_id = TEACHER.teacher_id
    WHERE CLASS.semester = ?
```

Listing 9: 查看課程資訊 SQL 指令

2. 查看輔導老師班表：學生可以依據自己的時間需求和對輔導老師的偏好去查閱此頁面來安排自己的輔導計劃：

```
SELECT
        MENTOR_SHIFT.shift, MENTOR.mentor_name
```

```
    FROM MENTOR_SHIFT
    LEFT JOIN MENTOR ON MENTOR_SHIFT.mentor_id = MENTOR.
       mentor_id
    ORDER BY MENTOR_SHIFT.shift ASC
```

Listing 10: 查看班表 SQL 指令

3.2.3 給 Mentor 的功能

1. 查看輔導老師班表：輔導老師可以確認自己的上班時間和其他同事的上班時間，來安排自己的時間或和行政人員申請班表更動：

```
SELECT
        MENTOR_SHIFT.shift, MENTOR.mentor_name
    FROM MENTOR_SHIFT
    LEFT JOIN MENTOR ON MENTOR_SHIFT.mentor_id = MENTOR.
       mentor_id
    ORDER BY MENTOR_SHIFT.shift ASC
```

Listing 11: 查看班表 SQL 指令

3.3 SQL 指令效能優化與索引建立分析

3.3.1 在 STUDENT 中加入學生代號的索引

我們認爲「MathSchool」系統中最常使用到的功能爲「查詢學生資訊」，因此我們決定針對該指令進行效能優化。

```
Create Index idx_student
On STUDENT(student_id)
```

Listing 12: 建立讀書會狀態索引

3.3.2 索引效果

對於 3.3.1 中我們設計的索引，我們利用同時顯示學生和其家長資訊的建立索引前運行五次的結果分別是 1.07 秒、1.48 秒、1.05 秒、1.15 秒、1.32 秒，平均運行時間約爲 1.213 秒，標準差約爲 0.1635359288 秒；建立索引後運行五次的結果分別是 0.95 秒、0.93 秒、0.91 秒、0.87 秒、0.93 秒，平均運行時間約爲 0.918 秒，標準差約爲 0.02712931993 秒。

在這兩個例子中，建立索引後平均運行時間的降低顯示了索引對查詢效率產生了顯著的影響。儘管如此，爲了評估索引的效果，可能需要更多的測試與分析，才能更全面地了解索引的實際效果。

3.4 交易管理

在多數 INSERT 和 UPDATE 操作中，我們使用了 PDO 的 prepare 和 execute 方法，這些操作通常是在隱式交易中進行的。如果執行成功，會自動提交；否則回滾。

```
$checkStmt = $pdo->prepare("SELECT version FROM
    STUDENT WHERE student_id = ?");
```

Listing 13: 交易管理

3.5 併行控制

以 student 爲範例，我們使用了 version 來進行並行控制，確保資料一直維持在 concurrency 的狀態：

```
try {
    $pdo->beginTransaction();

    $stmt = $pdo->prepare("
        INSERT INTO STUDENT (student_name, school,
            grade, tel, address, status, version)
        VALUES (?, ?, ?, ?, ?, ?, 0)
    ");
    $stmt->execute([...]);

    $stmt = $pdo->prepare("
        INSERT INTO PARENT (parent_name, parent_tel
            , student_id)
        VALUES (?, ?, ?)
    ");
    $stmt->execute([...]);

    $pdo->commit();
} catch (Exception $e) {
    $pdo->rollBack();
```

```
    echo "Transaction failed: " . $e->getMessage();
}
```

Listing 14: 交易管理

4 分工資訊

李亮節：Database 資料整理，黃子維：PHP 檔案，劉芊儀：HTML 和 CSS 檔案。

5 專案心得

李亮節：在建置工程中我的主要工作是掌握資料邏輯，生成符合資料庫格式的資料，並將資料導入資料庫中。我必須在掌握每個資料表的結構，並確保每筆資料都能夠正確地透過 Primary Key（主鍵）來唯一標識後，思考可行的大兩資料生成方法。我嘗試了連接 OpenAI API 來生成大量的資料，以加速資料建置的過程，但因爲額度沒辦法一次進行大量的資料處理，最後換成尋找現成方法。此外，我學會了在終端機快速進行資料導入、連結與操作，節省了很多時間。對於資料表之間的關聯和語法的除錯，例如設計 Foreign Key（外鍵）時可能遇到多種格式的衝突，設法解決的過程再次訓練了我對關聯式資料架構的邏輯與敏銳度。

黃子維：在這次資料庫期末專案的製作過程中，由於我負責的部分是 php 檔案的製作，我也學習到了蠻多相關的語法和架構。其中最有挑戰性的部分就是在設計功能，因爲需要了解使用者的需求，來決定再顯示資料時，是否需要 join 哪些表格來滿足使用者可能會需要查詢和編輯的資料欄位。另外，在整合不同頁面以及 debug 的過程中也會慢慢發現原本設計上的各種不足，所以在這個過程中除了程式能力以及對資料庫的相關了解有所提升之外，我更知道說設計一個能夠給使用者方便並且美觀的資料庫系統是具相當難度的，更何況我們現在處理的資料量不及眞實世界的巨大。總體而言這次專案非常疲累，卻也帶給我許多成長。

劉芊儀：這次的期末專案讓我對於整體的資料庫管理，以及如何建設一個可以實際操作的系統有了相當清楚的認識。在這次的專案中我主要負責前端的部分，因爲之前從未接觸過相關內容，在一開始的時候遭遇到不少困難，並且在實際開始設計前端之後，我才發現比我原本預想的來得費時及複雜許多，該如何做出一個要兼具美感跟實用功能的系統，需要考慮太多面向了，這讓我之後看到別人做的精美網站都深感佩服。雖然前端比我預想的還要不容易，但也因此我從過程中學習到不少新知識，擁有了一個實用的經驗。整個專案過程其實相當疲憊，但是看到我們一起眞的能夠成功做出一個系統時，眞的非常感動。儘管我們還有很多可以持續進步的空間，但我覺得這次專案眞的讓我們成長許多，是個非常難忘的體驗。