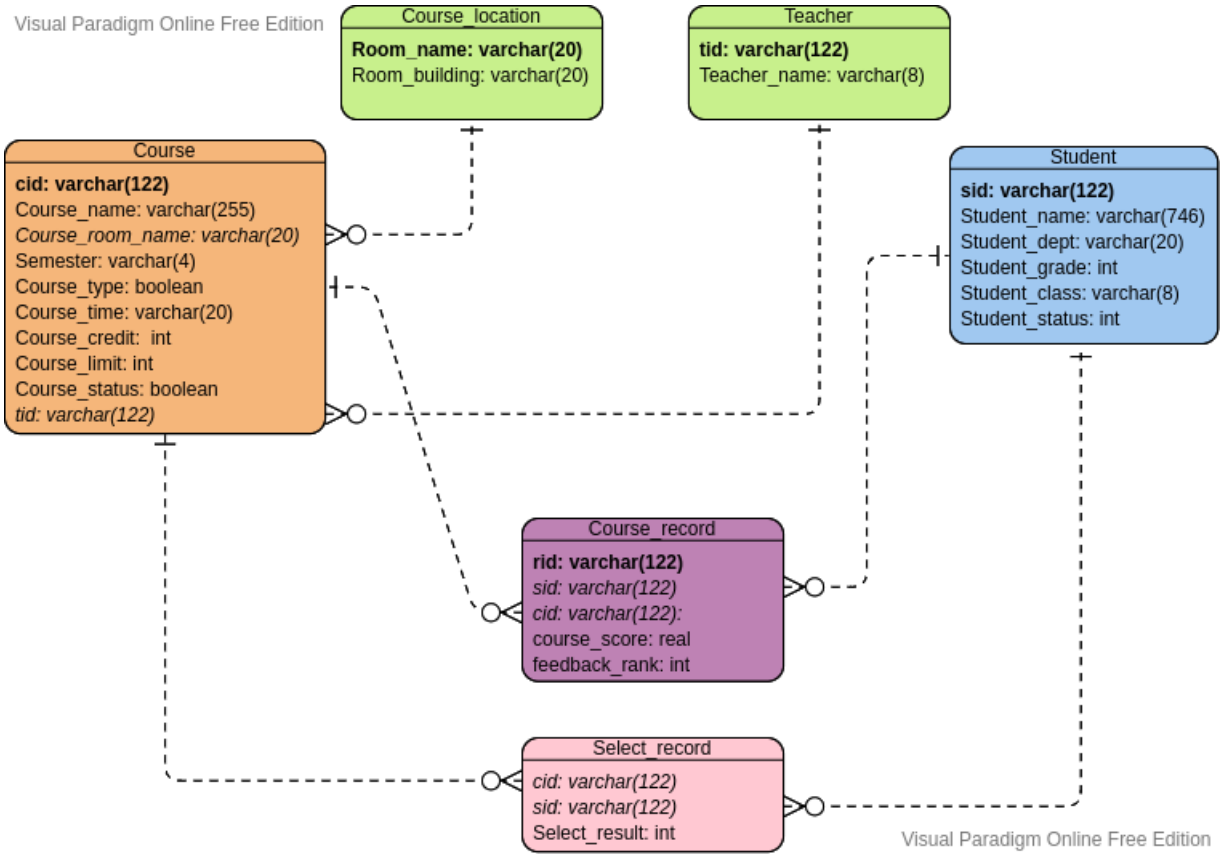
資料庫期末報告

題目要求

某校教務處記錄每位學生每學期的修課情況、該科成績、教學評量，設計了以下的資料表來儲存所需資料(未進行正規化)

#	欄位名稱	中文說明
1	semester	學期別（1091意為109上學期，1092則為109下學期）
2	course_no	課程編號
3	course_name	課程名稱
4	course_type	選修別（必修 / 或選修）
5	course_room	授課教室
6	course_building	授課地點
7	course_time	授課時間（一123：表示週一早上1-3節，一門課有多節次則以逗點隔開
8	course_credit	學分數
9	course_limit	課程人數限制
10	course_status	課程狀態(開課/停課)
11	course_is_online	是否為線上課程
12	teacher_name	授課教師姓名
13	student_name	修課者姓名
14	student_dept	修課者系所
15	student_grade	修課者年級
16	student_status	學生在學狀態 (在學 / 休學 / 退學)
17	student_class	修課者班別
18	select_result	選課結果
19	course_score	該科總成績
20	feedback_rank	教學評量結果 (1-5分)

正規化結果



Course

```sql
CREATE TABLE "course" (
    cid             VARCHAR(36)   NOT NULL UNIQUE,
    course_name     VARCHAR(255)  NOT NULL,
    course_room_name VARCHAR(20), -- If course_room_name is NULL, mean that course is on Internet.
    semester        VARCHAR(4)    NOT NULL,
    course_type     BOOLEAN       NOT NULL,
    course_time     VARCHAR(20)   NOT NULL,
    course_credit   INTEGER       NOT NULL,
    course_limit    INTEGER CHECK (course_limit > 0),
    course_status   BOOLEAN       NOT NULL,
    tid             VARCHAR(36)   NOT NULL,
    PRIMARY KEY (cid),
    FOREIGN KEY (course_room_name) REFERENCES course_location(room_name),
    FOREIGN KEY (tid) REFERENCES teacher(tid)
);
```

- 把課程抽出來，並且加上 `cid` 當作 Primary key
- 刪除 `course_is_online`，如果是線上課程， `course_room_name` 為 **NULL** 即可表達
- `course_type` 因為題目敘述只有表達必修/選修的用途，所以用 boolean 代替
- `course_status` 只有表達是否開課，所以同樣用 boolean 代替

Course_location

```sql
CREATE TABLE "course_location" (
    room_name     VARCHAR(20),
    room_building VARCHAR(20),
    PRIMARY KEY (room_name)
);
```

- `Room_name` 作為 Primary key, 讓 `Room_building` 不用重複記，其實更有效率的作法是 `Room_name` 的字串裡面加上 building 的資訊，但是這樣感覺跟原本題目改太多了，就沒有進一步的優化

Teacher

```
CREATE TABLE "teacher" (
    tid          VARCHAR(36) NOT NULL UNIQUE,
    teacher_name VARCHAR(8)  NOT NULL,
    PRIMARY KEY (tid)
);
```

- `tid` 作為 Primary key, 雖然題目沒有過多描述教師的欄位，但是一般情況會紀錄很多相關的基本資料，所以抽出來一個獨立的 table 並且配上 ID。

Student

```
 --- Reference https://reurl.cc/b29qoX
CREATE TABLE "student" (
    sid            VARCHAR(36)  NOT NULL,
    student_name   VARCHAR(746) NOT NULL,
    student_dept   VARCHAR(20)  NOT NULL,
    student_grade  INTEGER      NOT NULL CHECK (student_grade > 0),
    student_class  VARCHAR(8)   NOT NULL,
    student_status INTEGER      NOT NULL CHECK (student_status BETWEEN -1 AND 1),
    PRIMARY KEY (sid)
);
```

- `sid` 為 Primary key
- `student_name` 的長度設 746，是目前世界紀錄擁有最長名字的人，~~防止例外事件~~
- `student_status` 在規格書中只有三種狀態(在學 / 休學 / 退學)，所以我們直接用 int 存，並且加上只能介於 -1 ~ 1 的限制。

Select_record

```
CREATE TABLE "select_record" (
    cid VARCHAR(36) NOT NULL,
    sid VARCHAR(36)  NOT NULL,
    -- @select_result
    -- @中選: 0
    -- @落選: -1
    -- @備取順位: > 0
    select_result INTEGER NOT NULL CHECK (select_result > -2),
    PRIMARY KEY (cid, sid)
);
```

- `select_record` 存放選課紀錄
  - 由 `cid` 以及 `sid` 分別代表課程以及學生
- `select_result` 的部份我們選擇用 int 存放，原因是正常選課會有**備取**機制，雖然這次我們的專題注重在資料庫的設計，並不需要考慮的實際邏輯的處理，但是我們預先留了可以實作**備取**機制的欄位
  - 中選 -> 存 0
  - 落選 -> -1
  - 備取 -> 大於 0 的數字代表備取順位

Course_record

```
CREATE TABLE "course_record" (
    rid           VARCHAR(36) NOT NULL UNIQUE,
    sid           VARCHAR(36) NOT NULL,
    cid           VARCHAR(36) NOT NULL,
    course_score  REAL NOT NULL DEFAULT 0 CHECK (course_score BETWEEN 0.0 AND 100.0),
    feedback_rank INTEGER DEFAULT 1 CHECK (feedback_rank BETWEEN 1 AND 5),
    PRIMARY KEY (rid),
    FOREIGN KEY (sid) REFERENCES student(sid),
    FOREIGN KEY (cid) REFERENCES course(cid)
);
```

- `course_record` 存放的是正式選到課的紀錄
  - `rid` 為 Primary key
- `cid`, `sid` 分別為 Foreign key 指向 `course` 以及 `student`

實作 & 資料庫遷移

這次的規格中有附一些尚未正規化的資料，下面會介紹我們是如何一步一步的把資料庫遷移過去，並且簡單講一下跟原本設計的差別。

UUID (Universally Unique Identifier)

`sqlite` 作為一個輕量化的資料庫，基本的功能大致支援，但是後端常用的 uuid 卻一直沒有成為內建的型態，如果要內建的話甚至需要重新編譯 `sqlite` [1]，所以這次實作中為了要使用 uuid 的特性，我們使用 go 語言去操作 `sqlite`，而不是直接下 sql, 這樣我們就可以藉由應用層去生成 uuid，然後再寫入正規化完的資料庫中。

連接到資料庫

這邊使用 go 語言，先 `import` 資料庫還有 uuid 相關的 lib

```
import (
    "database/sql"
    "github.com/google/uuid"
    _ "github.com/mattn/go-sqlite3"
)
```

連線至資料庫

```
func main() {
    db, err := sql.Open("sqlite3", "test.db")
    if err != nil {
        log.Println(err)
    }
    defer db.Close()
}
```

遷移 course_location 資料

從原先文檔給的 `course_data` 裡面選取 `course_room`, `course_building` 搭配 **DISTINCT** 關鍵字，撈出所有的教室資料存到新建立的 `course_location` 中。

```
func createCourseLocationTable(db *sql.DB) {
    stmt := `
    INSERT INTO course_location (room_name, room_building)
        SELECT DISTINCT course_room, course_building
            FROM course_data;
`
    _, err := db.Exec(stmt)
    if err != nil {
        panic(err)
    }
}
```

遷移 student 資料並且分配 sid

```
type student struct {
    sid    string
    name   string
    dept   string
    grade  int
    class  string
    status int
}

func createStudentTable(db *sql.DB) {
    rows, err := db.Query("SELECT DISTINCT student_name, student_dept, student_grade, student_class, student_status FROM course_data
    if err != nil {
        panic(err)
    }

    students := make([]student, 0)

    defer rows.Close()
    for rows.Next() {
        var s studentTempTable

        err = rows.Scan(&s.name, &s.dept, &s.grade, &s.class, &s.status)
        if err != nil {
            panic(err)
        }
        students = append(students, student{sid: uuid.NewString(), name: s.name, dept: s.dept, grade: s.grade, class: s.class, statu
    }

    stmt := `
    INSERT INTO student (sid, student_name, student_dept, student_grade, student_class, student_status)
        VALUES (?, ?, ?, ?, ?, ?);`

    for _, s := range students {
        switch {
        case strings.HasSuffix(s.dept, "研究所"):
            s.grade += 4
        case strings.HasSuffix(s.dept, "博士班"):
            s.grade += 6
        default:
        }

        _, err := db.Exec(stmt, &s.sid, &s.name, &s.dept, &s.grade, &s.class, &s.status)
        if err != nil {
            panic(err)
        }
    }
}
```

這邊我們的 sql 分成兩段(應該有更好的寫法)，一開始利用 **DISTINCT** 選出所有不重複的學生資料，撈出來後利用 uuid 的 library 生成一個新的 `sid`，最後再一筆一筆 **Insert** 到 新建立的 `student` table。

因為及格成績會隨著大學/研究所變動，所以我們直接把 `grade` 欄位調整成類似美國的計算制度(研一 -> grade: 5, 研二 -> grade: 6)，這樣在下 query 的時候比較好處理。

遷移 teacher 資料並且分配 tid

```
func createTeacherTable(db *sql.DB) {
    rows, err := db.Query("SELECT DISTINCT teacher_name FROM course_data")
    if err != nil {
        panic(err)
    }
    defer rows.Close()

    tNames := make([]string, 0)
    for rows.Next() {
        var name string
        err := rows.Scan(&name)
        if err != nil {
            panic(err)
        }
        tNames = append(tNames, name)
    }

    stmt := `
    INSERT INTO teacher (tid, teacher_name)
        VALUES (?, ?)`

    for _, name := range tNames {
        _, err := db.Exec(stmt, uuid.NewString(), &name)
        if err != nil {
            panic(err)
        }
    }
}
```

因為在文檔中教師的資料相對單純，只有姓名，所以經過簡單的 query 後，為每個教師分配一個不同的 ID 再 **INSERT** 至 `teacher` table 中。

> 原本尚未正規化的 table 無法處理教師姓名相同的問題，所以可以看到測資裡面基本上都沒有考慮到這些例外事件。

遷移 course 資料

到這個步驟相較於原先尚未經過正規化的設計，我們已經把 `teacher`, `course_location` 拉出去了，所以在新建立 `course` table 時，也要把這些資料的 Foreign key 處理好。

```
type courseTempTable struct {
    semester    string
    name        string
    cType       string
    room        sql.NullString
    time        string
    credit      int
    limit       int
    status      string
    online      string
    teacherName string
}

func getCourseType(cType string) bool {
    return cType == "必修"
}

func getCourseStatus(status string) bool {
    return status == "開課"
}

func createCourseTable(db *sql.DB) {
    rows, err := db.Query("SELECT DISTINCT semester, course_name, course_type, course_room, course_time, course_credit, course_limit
    if err != nil {
        panic(err)
    }
    defer rows.Close()
```

```
    courses := make([]courseTempTable, 0)
    for rows.Next() {
        var c courseTempTable

        err := rows.Scan(&c.semester, &c.name, &c.cType, &c.room, &c.time, &c.credit, &c.limit, &c.status, &c.online, &c.teacherName
        if err != nil {
            panic(err)
        }
        courses = append(courses, c)
    }

    stmt := `
    INSERT INTO course (cid, course_name, course_room_name, semester, course_type, course_time, course_credit, course_limit, course_
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, (
            SELECT tid
                FROM teacher
                    WHERE teacher_name == ?
        ));
`


    for _, course := range courses {
        courseID := uuid.NewString()
        // 直接去除 online 欄位，如果 course.room_name 為 null，代表線上執行
        if course.online == "是" {
            course.room = sql.NullString{String: "", Valid: false}
        }

        // true 代表必修， false 代表選修
        courseType := getCourseType(course.cType)
        // true 代表開課， false 代表停開
        courseStatus := getCourseStatus(course.status)

        _, err := db.Exec(stmt, &courseID, &course.name, &course.room.String, &course.semester, &courseType, &course.time, &course.c
            &course.limit, &courseStatus, &course.teacherName)
        if err != nil {
            panic(err)
        }
    }
}
```

- 因為 `course_location` 我們沒有另外設 ID，而是直接用 `course_room_name` 當作 Primary key, 所以不需要額外的處理
- `teacher` 的部份我們每位教師都有一個 ID，所以先撈到 `teacher_name` 之後再去 `teacher` 的 table 中撈出對應的 `tid`
- 如果該課程是線上舉行， `course.room_name` 則會被存成 **NULL**
- `getCourseType` 把必修/選修的資料轉成 boolean
- `getCourseStatus` 把開課/停開的資料轉成 boolean
- 最後處理完就可以把精簡過的資料 **INSERT** 到 `course` table 中

遷移 select_record 資料

`select_record` 專門存放選課紀錄

```
// 之後可以加入備取機制
func createSelectRecordTable(db *sql.DB) {
    stme := `
    INSERT INTO select_record (cid, sid, select_result)
        SELECT course.cid, student.sid, select_result
            FROM course_data
                JOIN course
                    ON course_data.course_name == course.course_name
                JOIN student
                    ON course_data.student_name == student.student_name;
    `

    _, err := db.Exec(stme)
    if err != nil {
        panic(err)
    }
}
```

- `cid`, `sid` 需要利用 **JOIN** 的方式從 `course`, `student` 獲得
- 留了 `select_result` 欄位可以實作備取機制，這功能要求的假設太多(包含系所的優先順序，在網頁上的登記時間等等，所以沒有實作)

遷移 course_record 資料

`course_record` 存放的是正式選到課的紀錄(多出成績以及回饋欄位)

```
func createCourseRecordTable(db *sql.DB) {
    rows, err := db.Query("SELECT student_name, course_name, course_score, feedback_rank FROM course_data WHERE course_score IS NOT
    if err != nil {
        panic(err)
    }
    defer rows.Close()

    records := make([]courseRecordTempTable, 0)
    for rows.Next() {
        var record courseRecordTempTable
        err := rows.Scan(&record.studentName, &record.courseName, &record.courseScore, &record.feedback)
        if err != nil {
            panic(err)
        }

        records = append(records, record)
    }

    stmt := `
    INSERT INTO course_record (rid, sid, cid, course_score, feedback_rank)
        VALUES (?,
                (SELECT sid
                    FROM student
                        WHERE student_name == ?),
                (SELECT cid
                    FROM course
                        WHERE course_name == ?),
                ?, ?);

    `
    for _, record := range records {
        id := uuid.NewString()
        _, err := db.Exec(stmt, &id, &record.studentName, &record.courseName, &record.courseScore, &record.feedback)
        if err != nil {
            panic(err)
        }
    }
}
```

- 這裡的 query 其實是針對測資寫的，並不準確，因為我們發現只要沒有中選的人成績都會擺 **NULL**，就直接拿這個來判斷是否中選，但正統一點的寫法其實是去看 `select_result`。
- 撈到 `course_name` 以及 `student_name` 之後就去對應的 table 撈出他們的 ID

後續補充問題，請用 sql 解決以下事件

1.1102 學期的 A0001微積分，因故上課地點要由 K205 修改到 K210 大教室。

```
UPDATE course set course_room_name = "K210"
    WHERE course_name = "微積分" AND semester = "1102" AND course_room_name = "K205";
```

2.請列出 1102 學期的 A0002 計算機概論的修課名單

```
SELECT student.sid, student.student_name, student.student_dept
    FROM student, course, course_record
        WHERE course_record.sid == student.sid
            AND course_record.cid == course.cid
            AND course.course_name == "計算機概論"
            AND course.semester == "1102";
```

	sid	student_name	student_dept
1	d9de5003-2419-49d9-86db-7b86dab7279e	關羽	資訊工程系
2	b3c2e20a-8130-4bb3-ad4b-5ad53cbfc0af	周瑜	數學系
3	d3ded658-2774-4948-bb1b-3d56f623e3cd	黃蓋	數學系
4	09861700-5b6d-4231-a605-d9fb192f0db8	趙雲	數學系
5	7a8cf3a4-abe9-4f30-b6f8-6a82ab18fe59	夏侯惇	數學系
6	0b830741-0cbe-4ef3-8119-ab684e929bc5	華陀	資訊工程研究所

3.請列出 1102 學期，成績不及格的修課學生資料 (大學部低於 60 分，碩博 70 分)

```
SELECT course.course_name, student.sid, student.student_name, course_record.course_score
    FROM student, course, course_record
        WHERE course_record.sid == student.sid
            AND course_record.cid == course.cid
            AND course.semester == "1102"
            AND ((student.student_dept < 5 AND course_record.course_score < 60)
            OR (student.student_dept >= 5 AND course_record.course_score < 70))
    GROUP BY student.sid;
```

	course_name	sid	student_name	course_score
1	計算機概論	09861700-5b6d-4231-a605-d9fb192f0db8	趙雲	49.0
2	演算法	0b830741-0cbe-4ef3-8119-ab684e929bc5	華陀	68.0
3	經濟學	6074eb63-0518-4b73-9884-57da28ed5028	呂布	45.0
4	微積分	7a8cf3a4-abe9-4f30-b6f8-6a82ab18fe59	夏侯惇	67.0
5	虛擬實境	81e652c3-2cb0-4508-86f2-c20e2f02663c	呂蒙	65.0
6	統計學	81fce1b8-4b93-47d9-9493-47b11c7ee4f7	甘寧	46.0
7	統計學	a3f87706-4987-440f-9e76-9733b2ff5516	大喬	63.0
8	微積分	b3c2e20a-8130-4bb3-ad4b-5ad53cbfc0af	周瑜	56.0
9	音樂欣賞	ba4863e9-9672-4c7e-8705-39e99be5de48	諸葛亮	55.0
10	微積分	c5b132ed-b780-4131-a85c-aad9a8b7bfde	關興	55.0
11	微積分	d3ded658-2774-4948-bb1b-3d56f623e3cd	黃蓋	34.0
12	計算機概論	d9de5003-2419-49d9-86db-7b86dab7279e	關羽	66.0
13	音樂欣賞	e1055681-1b78-4a11-8346-31b880bccc3f	劉備	56.0
14	虛擬實境	e76b8b64-3029-4b17-8b54-119778c0c6bf	張飛	46.0

4.以中選比例 (中選人次/加選人次 * 100) 推測 1102 學期受學生歡迎的熱門加選課程

```
SELECT course.course_name AS 課名, teacher.teacher_name AS 授課教師, SUM(
    CASE
        WHEN select_record.select_result == "中選"
        THEN 1
        ELSE 0
    END) AS 中選人次, COUNT(select_record.select_result) AS 加選人次, ROUND(SUM(
    CASE
        WHEN select_record.select_result == "中選"
        THEN 1
        ELSE 0
    END)*100.0 / COUNT(select_record.select_result), 2) AS 中選比例
    FROM course, select_record, teacher
    WHERE course.cid == select_record.cid
        AND course.tid == teacher.tid
        AND course.semester == "1102"
    GROUP BY course.cid
    ORDER BY 中選比例 DESC;
```

	課名	授課教師	中選人次	加選人次	中選比例
1	計算機概論	陸羽	8	8	100.0
2	統計學	莊周	7	7	100.0
3	微積分	岳飛	7	7	100.0
4	虛擬實境	劉邦	9	10	90.0
5	經濟學	孔丘	6	7	85.71
6	演算法	達文西	4	5	80.0
7	音樂欣賞	巴哈	11	15	73.33

5.請列出 1102 學期線上課程教學評量平均分數及總分，找出大受好評的線上課程

```
SELECT course.course_name AS 課名, teacher.teacher_name AS 授課教師, SUM(course_record.feedback_rank) AS 教學評量總分, ROUND(AVG(cours
    FROM course, course_record, teacher
        WHERE course.cid == course_record.cid
            AND course.tid == teacher.tid
            AND course.semester == "1102"
    GROUP BY course.cid
    ORDER BY 教學評量平均分數 DESC;
```

	課名	授課教師	教學評量總分	教學評量平均分數
1	虛擬實境	劉邦	30	4.3
2	音樂欣賞	巴哈	38	4.2
3	演算法	達文西	12	4.0
4	計算機概論	陸羽	24	4.0
5	經濟學	孔丘	19	3.8
6	微積分	岳飛	17	2.8
7	統計學	莊周	17	2.8

Reference

- sqlite3-uuid