

資料庫管理 HW02

B12508026 戴偉璿

1. (a)
 - i. **TRUE**, because DEAN is a **relation** between COLLEGE and INSTRUCTOR; and CHAIR is a **relation** between DEPT and INSTRUCTOR.
 - ii. **FALSE**, there's no further restriction on DEAN and CHAIR, so one INSTRUCTOR can be a CHAIR and a DEAN at the same time.
 - iii. **TRUE**, the relation between STUDENT and HAS is a **(0, 1)** relation, so one STUDENT can HAS zero or one DEPT.
 - iv. **TRUE**, the cardinality between STUDENT and TAKES is **(0, N)**, so one student may take zero or more sections; while the cardinality between SECTION and TAKES is **(5, N)**, so one section must be taken by five or more students.
 - v. **TRUE**, the cardinality between COURSE and SECTION is **(1, 1)**, so one section must be related to exactly one course.
- (b) As the following diagram:

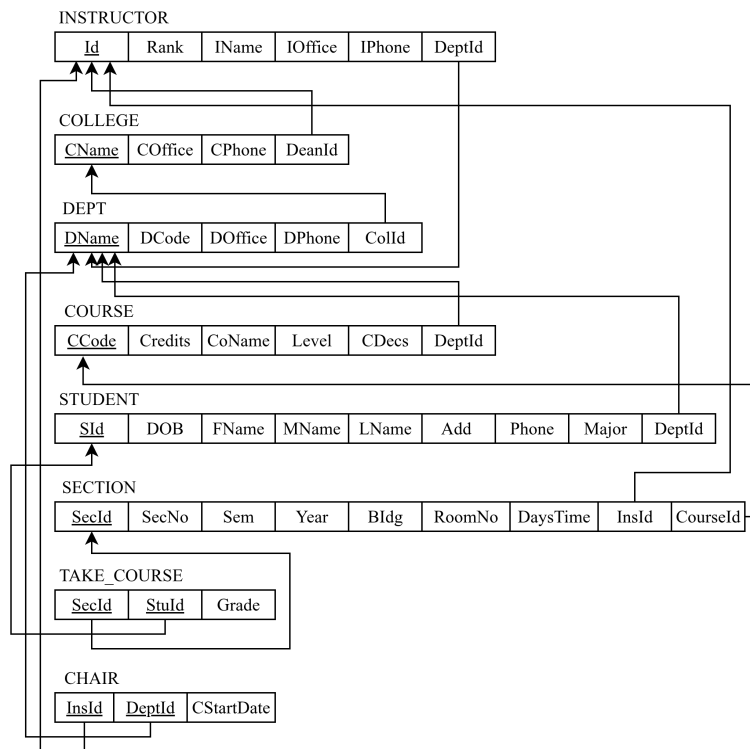


Figure 1: Relational Schema Diagram

2. (a) To record the full history of students' take or drop sections, we can just add more attributes to the **TAKES** relation. But it may cause some problems while querying the final result and grade (user must find the last record of the log to reach). So I decide to create a new weak entity to record the operation log.

If the operation is add, the **drop_data** would be record as NULL; vice versa.

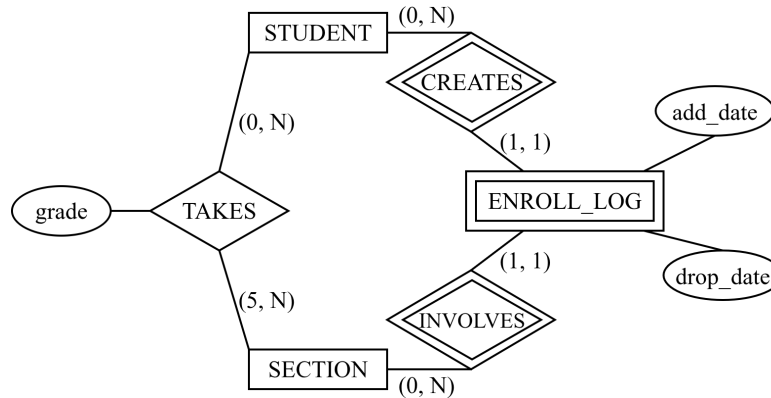


Figure 2: ER Diagram with Operation Log

- (b) As the following diagram:

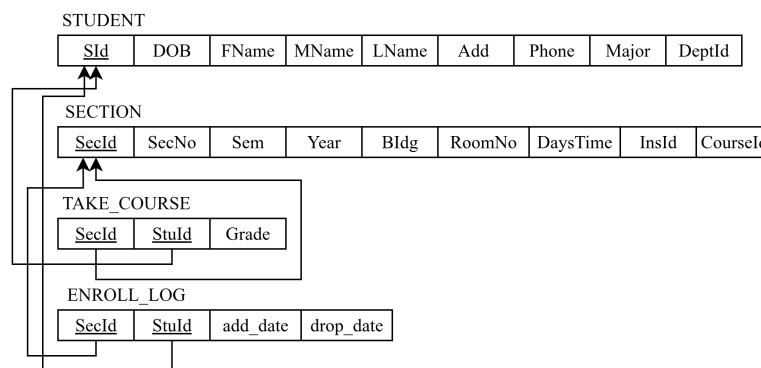


Figure 3: Relational Schema Diagram with Advisor

3. (a)
- ```

1 SELECT br.Card_no, br.Name, COUNT(*) AS LoanRec
2 FROM BOOK_LOANS b1
3 JOIN BORROWER br ON b1.Card_no = br.Card_no
4 WHERE b1.Branch_id = '[ASSIGNED_BRANCH_ID]'
5 GROUP BY br.Card_no, br.Name
6 ORDER BY LoanRec DESC;

```
- (b)
- ```

1  SELECT lib.Branch_id, lib.Branch_name, COUNT(*) AS LoanRec
2  FROM BOOK_LOANS b1
3  JOIN LIBRARY_BRANCH lib ON lib.Branch_id = b1.Branch_id
4  WHERE b1.Date_out BETWEEN '2024-01-01' AND '2024-12-31'
5  GROUP BY lib.Branch_id, lib.Branch_name
6  ORDER BY LoanRec DESC;

```

```

(c) 1      SELECT bk.Book_id, bk.Title, COUNT(DISTINCT ba.Author_id) AS AuthorNum, bc.No_of_copies,
      ↪      COUNT(DISTINCT bl.Loan_id) AS LoanRec
2      FROM BOOK_LOANS bl
3      JOIN BOOK bk ON bl.Book_id = bk.Book_id
4      JOIN BOOK_AUTHORS ba ON bl.Book_id = ba.Book_id
5      JOIN BOOK_COPIES bc ON bl.BOOK_id = bc.BOOK_id AND bl.Branch_id = bc.Branch_id
6      WHERE bl.Branch_id = (
7          SELECT b12.Branch_id
8          FROM BOOK_LOANS b12
9          JOIN LIBRARY_BRANCH lib ON lib.Branch_id = b12.Branch_id
10         WHERE b12.Date_out BETWEEN '2024-01-01' AND '2024-12-31'
11         GROUP BY b12.Branch_id
12         ORDER BY COUNT(*) DESC
13         LIMIT 1
14     )
15     AND bl.Date_out BETWEEN '2024-01-01' AND '2024-12-31'
16     GROUP BY bk.Book_id, bk.Title, bc.No_of_copies
17     ORDER BY LoanRec DESC;

```

```

(d) 1      SELECT bk.Book_id, bk.Title, lib.Branch_name, bc.No_of_copies
2      FROM BOOK bk
3      JOIN BOOK_COPIES bc ON bk.Book_id = bc.Book_id
4      JOIN LIBRARY_BRANCH lib ON bc.Branch_id = lib.Branch_id
5      WHERE bk.Book_id IN (
6          SELECT ba.Book_id
7          FROM BOOK_AUTHOR ba
8          GROUP BY ba.Book_id
9          HAVING COUNT(DISTINCT ba.Author_id) = 1
10     );

```

```

(e) 1      --Add new column to the DB
2      ALTER TABLE BOOK_LOANS
3      ADD COLUMN Date_return DATE;
4
5      --Execute Query
6      SELECT bk.Title, lib.Branch_name, bc.No_of_copies - COUNT(bl.Loan_id) AS AvaiCopies
7      FROM BOOK bk
8      JOIN BOOK_COPIES bc ON bk.Book_id = bc.Book_id
9      JOIN LIBRARY_BRANCH lib ON bc.Branch_id = lib.Branch_id
10     LEFT JOIN BOOK_LOANS bl ON bk.Book_id = bl.Book_id AND bc.Branch_id = bl.Branch_id AND
      ↪     bl.Date_return IS NULL
11     GROUP BY bk.Title, lib.Branch_name, bc.No_of_copies

```
