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 Figure 1 shows:

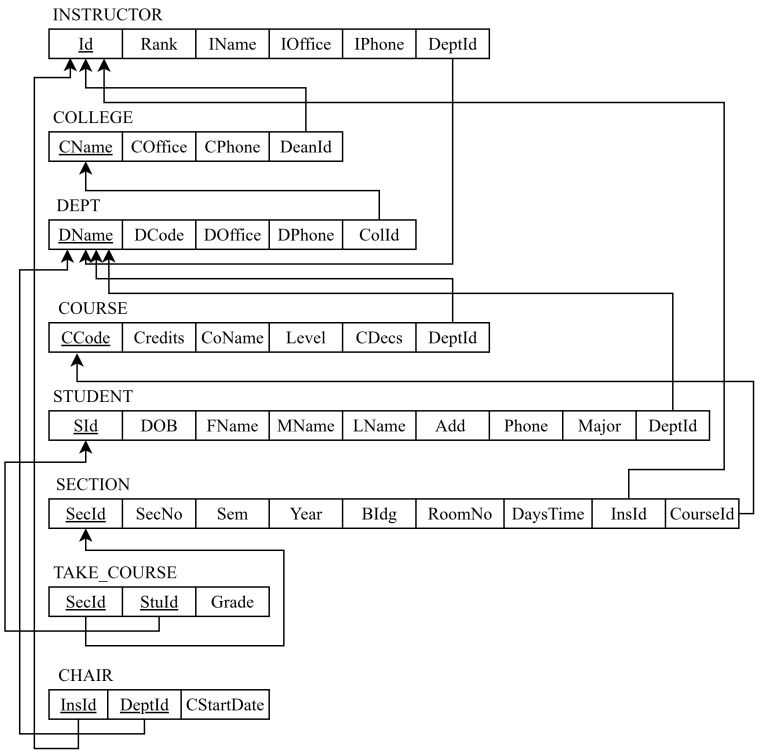

Figure 1: Relational Schema Diagram

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.

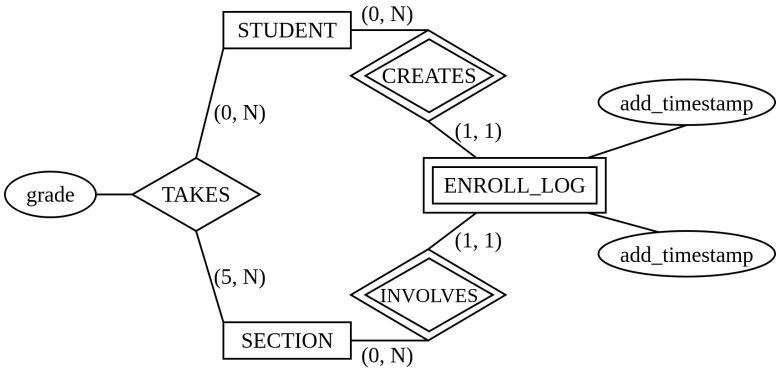

Figure 2: ER Diagram with Operation Log

(b) As the Figure 3 shows:

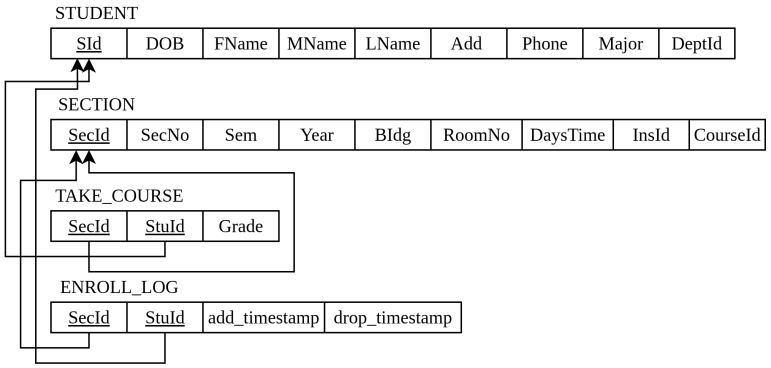

Figure 3: Relational Schema Diagram with Operation Log

3. (a)

```sql
SELECT br.Card_no, br.Name, COUNT(bl.Loan_id) AS LoanRec
FROM BORROWER br
LEFT JOIN BOOK_LOANS bl ON bl.Card_no = br.Card_no AND bl.Branch_id =
↪  '[ASSIGNED_BRANCH_ID]'
GROUP BY br.Card_no, br.Name
ORDER BY LoanRec DESC;
```

(b)

```sql
SELECT lib.Branch_id, lib.Branch_name, COUNT(bl.Loan_id) AS LoanRec
FROM LIBRARY_BRANCH lib
LEFT JOIN BOOK_LOANS bl ON lib.Branch_id = bl.Branch_id AND bl.Date_out BETWEEN
↪  '2024-01-01' AND '2024-12-31'
GROUP BY lib.Branch_id, lib.Branch_name
ORDER BY LoanRec DESC;
```

(c)

```sql
SELECT bk.Book_id, bk.Title, COUNT(DISTINCT ba.Author_name) AS AuthorNum,
↪  bc.No_of_copies, COUNT(DISTINCT bl.Loan_id) AS LoanRec
FROM BOOK_COPIES bc
JOIN BOOK bk ON bc.Book_id = bk.Book_id
JOIN BOOK_AUTHORS ba ON bc.Book_id = ba.Book_id
LEFT JOIN BOOK_LOANS bl ON bc.Book_id = bl.Book_id AND bc.Branch_id = bl.Branch_id AND
↪  bl.Date_out BETWEEN '2024-01-01' AND '2024-12-31'
WHERE bc.Branch_id = (
    SELECT bl2.Branch_id
    FROM BOOK_LOANS bl2
    JOIN LIBRARY_BRANCH lib ON lib.Branch_id = bl2.Branch_id
    WHERE bl2.Date_out BETWEEN '2024-01-01' AND '2024-12-31'
    GROUP BY bl2.Branch_id
    ORDER BY COUNT(*) DESC
    LIMIT 1
)
GROUP BY bk.Book_id, bk.Title, bc.No_of_copies
ORDER BY LoanRec DESC;
```

(d)

```sql
SELECT bk.Book_id, bk.Title, lib.Branch_name, bc.No_of_copies
FROM BOOK bk
JOIN BOOK_COPIES bc ON bk.Book_id = bc.Book_id
JOIN LIBRARY_BRANCH lib ON bc.Branch_id = lib.Branch_id
WHERE bk.Book_id IN (
    SELECT ba.Book_id
    FROM BOOK_AUTHORS ba
    GROUP BY ba.Book_id
    HAVING COUNT(DISTINCT ba.Author_name) = 1
);
```

(e)

```sql
--Add new column to the DB
ALTER TABLE BOOK_LOANS
ADD COLUMN Date_return DATE;

--Execute Query
SELECT bk.Title, lib.Branch_name, bc.No_of_copies - COUNT(bl.Loan_id) AS AvaiCopies
FROM BOOK bk
JOIN BOOK_COPIES bc ON bk.Book_id = bc.Book_id
JOIN LIBRARY_BRANCH lib ON bc.Branch_id = lib.Branch_id
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
GROUP BY bk.Title, lib.Branch_name, bc.No_of_copies;
```

4. (a) As the Table 1 shows:

Name	Data Type	Key	Constraint	Domain
BOOK				
Book_id	varchar(15)	PK	Not Null, Unique	
Title	varchar(100)		Not Null	
Publisher_name	varchar(50)	FK → PUBLISHER(Name)	Not Null	
BOOK_AUTHORS				
Book_id	varchar(15)	PK, FK → BOOK(Book_id)	Not Null	
Author_name	varchar(50)	PK	Not Null	
PUBLISHER				
Name	varchar(50)	PK	Not Null, Unique	
Address	varchar(100)			
Phone	varchar(15)			
BOOK_COPIES				
Book_id	varchar(15)	PK, FK → BOOK(Book_id)	Not Null	
Branch_id	varchar(10)	PK, FK → LIBRARY_BRANCH (Branch_id)	Not Null	
No_of_copies	int		Not Null, CHECK > 0	$\{1, 2, \dots\}$
BOOK_LOANS				
Book_id	varchar(15)	PK, FK → BOOK(Book_id)	Not Null	
Branch_id	varchar(10)	PK, FK → LIBRARY_BRANCH (Branch_id)	Not Null	
Card_no	varchar(10)	PK, FK → BORROWER(Card_no)	Not Null	
Date_out	date	PK	Not Null	
Due_date	date		Not Null	
Date_return	date			
LIBRARY_BRANCH				
Branch_id	varchar(10)	PK	Not Null, Unique	
Branch_name	varchar(50)		Not Null	
Address	varchar(100)		Not Null, Unique	
BORROWER				
Card_no	varchar(10)	PK	Not Null, Unique	
Name	varchar(50)		Not Null	
Address	varchar(100)			
Phone	varchar(15)			

Table 1: Data Dictionary for Library Management System

(b)

```sql
CREATE TABLE PUBLISHER (
    Name VARCHAR(50) PRIMARY KEY,
    Address VARCHAR(100),
    Phone VARCHAR(15)
);

CREATE TABLE LIBRARY_BRANCH (
    Branch_id VARCHAR(10) PRIMARY KEY,
    Branch_name VARCHAR(50) NOT NULL,
    Address VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE BORROWER (
    Card_no VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(100),
    Phone VARCHAR(15)
);

CREATE TABLE BOOK (
    Book_id VARCHAR(15) PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Publisher_name VARCHAR(50) NOT NULL,
    FOREIGN KEY (Publisher_name) REFERENCES PUBLISHER(Name)
);

CREATE TABLE BOOK_AUTHORS (
    Book_id VARCHAR(15) NOT NULL,
    Author_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (Book_id, Author_name),
    FOREIGN KEY (Book_id) REFERENCES BOOK(Book_id)
);

CREATE TABLE BOOK_COPIES (
    Book_id VARCHAR(15) NOT NULL,
    Branch_id VARCHAR(10) NOT NULL,
    No_of_copies INT NOT NULL CHECK (No_of_copies > 0),
    PRIMARY KEY (Book_id, Branch_id),
    FOREIGN KEY (Book_id) REFERENCES BOOK(Book_id),
    FOREIGN KEY (Branch_id) REFERENCES LIBRARY_BRANCH(Branch_id)
);

CREATE TABLE BOOK_LOANS (
    Book_id VARCHAR(15) NOT NULL,
    Branch_id VARCHAR(10) NOT NULL,
    Card_no VARCHAR(10) NOT NULL,
    Date_out DATE NOT NULL,
    Due_date DATE NOT NULL,
    Date_return DATE,   -- can be NULL
    PRIMARY KEY (Book_id, Branch_id, Card_no, Date_out),
    FOREIGN KEY (Book_id) REFERENCES BOOK(Book_id),
    FOREIGN KEY (Branch_id) REFERENCES LIBRARY_BRANCH(Branch_id),
    FOREIGN KEY (Card_no) REFERENCES BORROWER(Card_no)
);
```