SE 2141

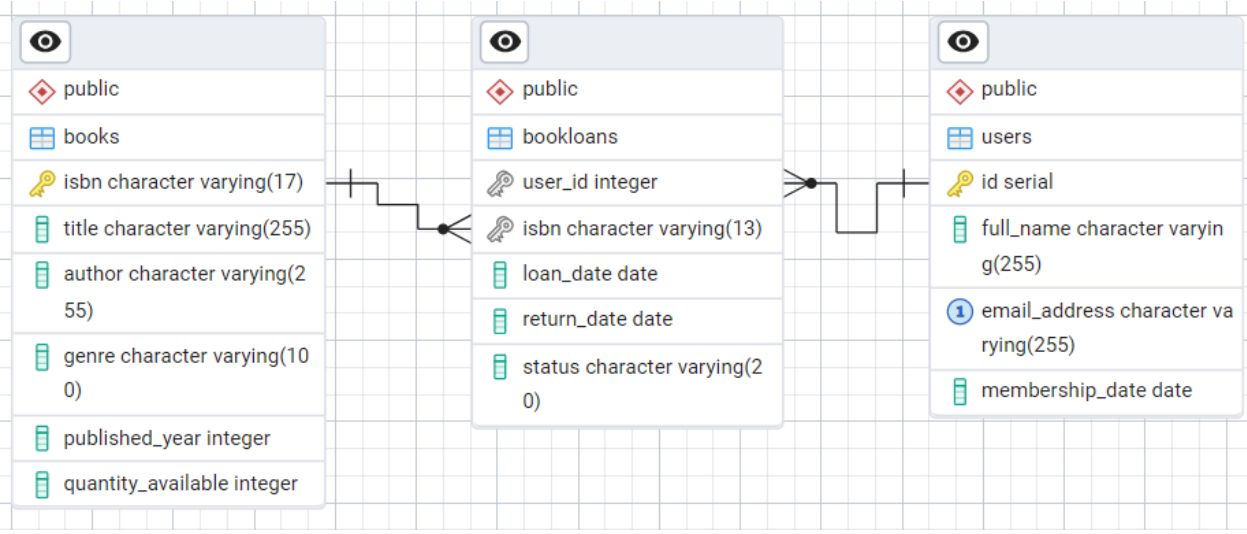
Laboratory 4

December 11, 2024

Kaizen Louie Somosera

BSSE - 2

Part 1: Conceptual Design - 25pts



Here is the conceptual model with the entities, its attributes, primary keys, and relationships with cardinalities. Each entity has an attribute of their own connecting to the bookloans which connects the two using their foreign key (user_id for users and isbn for books).

Part 2: Logical Design - 25pts

Query | Query History

```sql
CREATE TABLE IF NOT EXISTS Books (
    isbn VARCHAR(17) UNIQUE PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    genre VARCHAR(100) NOT NULL,
    published_year INT NOT NULL,
    quantity_available INT NOT NULL CHECK (quantity_available >= 0)
);

CREATE TABLE IF NOT EXISTS Users (
    id SERIAL UNIQUE PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email_address VARCHAR(255) UNIQUE NOT NULL,
    membership_date DATE NOT NULL
);

CREATE TABLE IF NOT EXISTS BookLoans (
    user_id INT NOT NULL,
    isbn VARCHAR(17) NOT NULL,
    loan_date DATE DEFAULT now() NOT NULL,
    return_date DATE DEFAULT now() + '30 days'::interval NOT NULL,
    status VARCHAR(20)DEFAULT 'borrowed' NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE CASCADE,
    FOREIGN KEY (isbn) REFERENCES Books(isbn) ON DELETE CASCADE
);
```

Data Output | Messages | Notifications

```
NOTICE:  relation "books" already exists, skipping
NOTICE:  relation "users" already exists, skipping
NOTICE:  relation "bookloans" already exists, skipping
CREATE TABLE

Query returned successfully in 260 msec.
```

I then translated the ER diagram into relational tables, you can see the attributes, data types, and constraints such as primary keys, foreign keys, and NOT NULL present here. I set the isbn as primary key on Books, id as primary key on Users, and none on BookLoans. It will check the book quantity if it's not less than 0 so that it won't have a negative number, and also automated the input on dated plus the 30 days which will set the deadline.

Part 3: SQL Queries

a. Insert a new book into the library with a quantity of 5.

Query | Query History

```
1  -- INSERT INTO Books (isbn, title, author, genre, published_year, quantity_available)
2  -- VALUES
3  --  ('978-1-60309-502-0', 'Animal Stories', 'Maria Hoey', 'Comics', 2022, 3),
4  --  ('978-1-60309-517-4', 'Ashes', 'Álvaro Ortiz', 'Fiction', 2019, 8),
5  --  ('978-1-60309-442-9', 'Belzebubs', 'J.P. Ahonen', 'Comics', 2019, 6),
6  --  ('978-1-60309-542-6', 'Belzebubs (Vol 2): No Rest for the Wicked', 'J.P. Ahonen', 'Comics', 2022, 9),
7  --  ('978-1-60309-527-3', 'But You Have Friends', 'Hélène Becquelin', 'Graphic Novel', 2021, 5);
8
9  SELECT * FROM Books;
```

Data Output | Messages | Notifications

	isbn [PK] character varying (17)	title character varying (255)	author character varying (255)	genre character varying (100)	published_year integer	quantity_available integer
1	978-1-60309-502-0	Animal Stories	Maria Hoey	Comics	2022	3
2	978-1-60309-517-4	Ashes	Álvaro Ortiz	Fiction	2019	8
3	978-1-60309-442-9	Belzebubs	J.P. Ahonen	Comics	2019	6
4	978-1-60309-542-6	Belzebubs (Vol 2): No Rest for the Wicked	J.P. Ahonen	Comics	2022	9
5	978-1-60309-527-3	But You Have Friends	Hélène Becquelin	Graphic Novel	2021	5

I added a list of books with a book that has a quantity of 5.

b. Add a new user to the system.

Query | Query History

```
1  -- INSERT INTO Users (full_name, email_address, membership_date)
2  -- VALUES
3  --  ('Jose Suoberon', 'jose@example.com', '2024-10-02'),
4  --  ('Rofer Casio', 'rofer@example.com', '2024-09-21'),
5  --  ('Nicholae Sara', 'nich@example.com', '2024-11-11'),
6  --  ('Paul Ardiente', 'paul@example.com', '2024-12-25'),
7  --  ('Dainz Trasadas', 'dainz@example.com', '2024-09-11');
8
9  SELECT * FROM Users;
```

Data Output | Messages | Notifications

SQL

	id [PK] integer	full_name character varying (255)	email_address character varying (255)	membership_date date
1	1	Jose Suoberon	jose@example.com	2024-10-02
2	2	Rofer Casio	rofer@example.com	2024-09-21
3	3	Nicholae Sara	nich@example.com	2024-11-11
4	4	Paul Ardiente	paul@example.com	2024-12-25
5	5	Dainz Trasadas	dainz@example.com	2024-09-11

I added users to fill the table with data.

c. Record a book loan for a user

Query | Query History

```
1  -- INSERT INTO BookLoans (user_id, isbn, loan_date, return_date, status)
2  -- VALUES
3  --     (3, '978-1-60309-442-9', '2024-11-12', '2024-11-19', 'returned'),
4  --     (4, '978-1-60309-542-6', '2024-12-09', '2025-01-02', 'borrowed'),
5  --     (5, '978-1-60309-527-3', '2024-09-12', '2024-09-19', 'returned');
6
7
8  -- INSERT INTO BookLoans (user_id, isbn)
9  -- VALUES
10 --     (1, '978-1-60309-502-0'),
11 --     (2, '978-1-60309-517-4');
12
13 SELECT * FROM BookLoans;
```

Data Output | Messages | Notifications

	user_id integer	isbn character varying (17)	loan_date date	return_date date	status character varying (20)
1	1	978-1-60309-502-0	2024-12-11	2025-01-10	borrowed
2	2	978-1-60309-517-4	2024-12-11	2025-01-10	borrowed
3	3	978-1-60309-442-9	2024-11-12	2024-11-19	returned
4	4	978-1-60309-542-6	2024-12-09	2025-01-02	borrowed
5	5	978-1-60309-527-3	2024-09-12	2024-09-19	returned

I added bookloans connecting the user_id and the book isbn, adding the loan_date the return_date, and the status. I separate the other two data to check if the code works where it will automatically set the dates based on the current and current + 30 days.

d. Find all books borrowed by a specific user.

```
SELECT U.full_name, B.title, B.author, BL.loan_date, BL.return_date, BL.status, B.quantity_available
FROM BookLoans BL
JOIN Books B ON BL.isbn = B.isbn
JOIN Users U ON BL.user_id = U.id
WHERE U.id = 5;
```

	full_name character varying (255)	title character varying (255)	author character varying (255)	loan_date date	return_date date	status character varying (20)	quantity_available integer
1	Dainz Trasadas	But You Have Friends	Hélène Becquelin	2024-09-12	2024-09-19	returned	4

I checked if what books were borrowed by User.id = 5 and it shows what book it is as well as the quantity were diminished by 1.

e. List all overdue loans.

```
SELECT * FROM BookLoans WHERE status = 'overdue';
```

user_id integer	isbn character varying (17)	loan_date date	return_date date	status character varying (20)

As you can see there is no returned books that has a status overdue.

Part 4: Data Integrity and Optimization

The prevention of borrowing books when no copies are available. (Check table creation for code)

Query Query History

```
1  -- '978-1-60309-502-0' quantity_available = 0
2  UPDATE Books
3  SET quantity_available = quantity_available - 1
4  WHERE ISBN = '978-1-60309-502-0';
5
```

Scratch Pad

Data Output Messages Notifications

```
ERROR:  Failing row contains (978-1-60309-502-0, Animal Stories, Maria Hoey, Comics, 2022, -1).new row for relation "books" violates check constraint
"books_quantity_available_check"
```

Fast retrieval of overdue loans. (20 pts - with CODE and actual screenshot of performance)

Query Query History

```
1  CREATE INDEX idx_return_date ON BookLoans(return_date);
2  CREATE INDEX idx_status ON BookLoans(status);
3
```

Data Output Messages Notifications

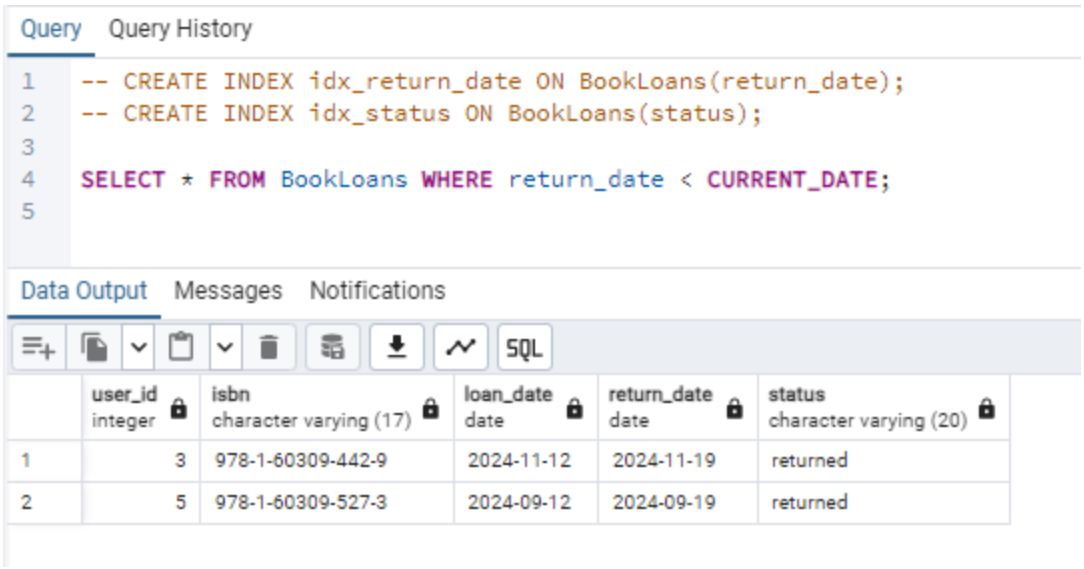
```
CREATE INDEX

Query returned successfully in 361 msec.
```

Query Query History

```
1  -- CREATE INDEX idx_return_date ON BookLoans(return_date);
2  -- CREATE INDEX idx_status ON BookLoans(status);
3
4  SELECT * FROM BookLoans WHERE return_date < CURRENT_DATE;
5
```

Data Output Messages Notifications

```
Successfully run. Total query runtime: 660 msec.
2 rows affected.
```



Part 5: Reflection (25 pts)

What challenges might arise when scaling this database to handle millions of users and books? Suggest one solution for each challenge.

Challenge #1: As the number of users, books, and book loans increases, query performance may reduce. For example, searching for books or finding overdue loans may become slower, especially if there are no proper indexing strategies.

Solutions: Use indexing and query optimization

Challenge #2: As the number of books and users grows, the database will require more storage space for both the data and the indexes.

Solutions: Split the database into smaller, more manageable pieces. Archiving old data that have been closed or inactive.

Challenge #3: With millions of data, the back up process could take a long time and recovering the system after a failure might be more difficult.

Solution: Implement incremental backups to only back up changes since the last backup and set up database replication across multiple nodes and regions.