Project Proposal

COMS 4111: Introduction to Databases

Omkar Vijay Pitale
op2281@columbia.edu

Akshay Ganesh Iyer
agi2108@columbia.edu

October 16, 2023

Contents

1 StudySpace

Imagine having access to all the high quality eBooks at your fingertips. StudySpace is a revolutionary eBook subscription service portal that aims to transform the way people read, learn and explore the world of knowledge. We plan to give access to books based on subscription plans instead of users having to individually select and purchase books, so that reading is easier than ever before.

2 Primary functionality

1. The user will first be asked to login, as a customer, author or employee. If the customer/author does not have an existing account, they will be asked to sign up.
2. There will be different views for each of the three categories of users.
3. Customers will have a search bar, which they can use to filter books and go through our available collection.
4. Author will have a portal where they upload new books, which will then be verified by employees.
5. Additional functionalities to be added as we go ahead with the project.

3 Entities

1. Users (user_id, given_name, last_name, mobile, email, password) with child entities inheriting Users -
   a Customers (address, payment_details, subscription_status)
   b Employees (address, ssn)
   c Authors (is_verified)

   Note: Only specialized attributes are mentioned in parenthesis of child entities here.
2. Subscriptions (subscription_id, subscription_name, subscription_cost, description)
3. PaymentTransactions (transaction_id, time_stamp)
4. Books (book_id, book_name, edition, pages, publication_year, description, language, google_link)
5. Publishers (publisher_id, publisher_name)
6. Sections (section_id, section_name)

Note: More detailed description of entity-sets is given in the Entity-Relationship Diagram i.e. domain of each attributes, constraints, etc.

4 Constraints

1. Data level constraints -
   a. Auto incrementing primary key on each entity-set table.
   PostgreSQL supports this with `SERIAL` data type.
   b. Users
      i. Has 3 child entities i.e. Customers, Employees and Authors.
      ii. email is a candidate key.
      iii. given_name, last_name, password should not be `NULL`.
      iv. mobile number should be unique.

v. Customers have their subscription_status set to false by default as everyone is enrolled in the free plan when they sign up.

vi. payment_details can hold only 4 string values i.e. "Apple Pay", "Google Pay", "Credit Card", "Debit Card" or can be `NULL`

vii. Authors have their verification status set to false i.e is_verified set to false by default.

viii. Employees have ssn as a candidate key and employee address cannot be `NULL`.

c. Books

i. book_name, google_link cannot be `NULL`.

d. Publisher

i. publisher_name cannot be `NULL`.

e. Subscriptions

i. subscription_name is a candidate key.

ii. subscription_cost cannot be `NULL`.

2. Application level constraints -

i A same user account cannot have more than 2 roles, i.e. for example cannot be a Customer and an Author.

ii A book is definitely published by at least one author but it is not necessary to have a publisher.

iii Multiple Customers can have the same subscription plan but a single customer cannot be in more than one subscription plan.

5 Data Plan

Since there is no official data available online which we can inject directly into our database. We plan to create synthetic data according to our entities using the publicly available free eBooks on google which we can download and use.

6 Challenges

1. Creating synthetic data is our first challenge as we could not find any official dataset we could use to create our database. We plan on using the publicly available eBooks to avoid piracy issues. There is a large chunk of books available for free online.

2. The challenge of storing pdfs. Since our platform is a digital library, we also need to store the eBooks in pdf format. It is not a good practice to store pdfs in a database. To overcome this situation we plan to exploit the storage given by Columbia University on google drive and upload it over there while adding the shareable view links in our PostgreSQL database.

7 Entities and Relationships

We have submitted the Entity-Relationship Diagram as a separate SVG file but you can access it from here as well - ER Diagram file.

Note: This is a view only google link only, which can only be opened using Columbia's LionMail.

Note: On Google drive SVG renders a bit blurr, we recommend downloading the SVG file and then viewing it on the browser for high quality rendering.

8 SQL Mapping

Note: Since we have decided to go with auto-incrementing primary keys, PostgreSQL supports it with the help of `SERIAL` data type. So foreign keys for this primary key in other tables should have `INT` data type.

Note: For Customers, Employees, Authors we cannot map the non-overlap constraint yet.

1. Mapping Customers -

```
CREATE TABLE Customers (
    customer_id CHAR(10),
    given_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    mobile CHAR(10),
    email VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL,
    address VARCHAR(60),
    payment_details VARCHAR(20),
    subscription_status BOOLEAN DEFAULT false,
    UNIQUE (email),
    UNIQUE (mobile),
    PRIMARY KEY (customer_id),
);
```

2. Mapping Employees -

```
CREATE TABLE Employees (
    employee_id CHAR(10),
    given_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    mobile CHAR(10),
    email VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL,
    address VARCHAR(60),
    ssn CHAR(13) NOT NULL,
    UNIQUE (email),
    UNIQUE (mobile),
    UNIQUE(ssn),
    PRIMARY KEY (employee_id),
);
```

3. Mapping Authors -

```
CREATE TABLE Authors (
    author_id CHAR(10),
    given_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    mobile CHAR(10),
    email VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL,
    is_verified BOOLEAN DEFAULT false,
    UNIQUE (email),
    UNIQUE (mobile),
    PRIMARY KEY (author_id),
);
```

4. Mapping Books -

```
CREATE TABLE Books (
    book_id SERIAL,
    book_name VARCHAR(80) NOT NULL,
    edition INT,
    pages INT,
    publication_year INT,
    description VARCHAR(200),
    language VARCHAR(100),
    google_link VARCHAR(60) NOT NULL,
    PRIMARY KEY (book_id)
);
```

5. Mapping PaymentTransactions -

```
CREATE TABLE PaymentTransactions (
    transaction_id SERIAL,
    time_stamp TIMESTAMP,
    PRIMARY KEY (transaction_id)
);
```

6. Mapping Subscriptions -

```
CREATE TABLE Subscriptions (
    subscription_id SERIAL,
    subscription_name VARCHAR(30) NOT NULL,
    subscription_cost REAL NOT NULL,
    description VARCHAR(100),
    UNIQUE(subscription_name),
    PRIMARY KEY (subscription_id)
);
```

7. Mapping Sections -

```
CREATE TABLE Sections (
    section_id SERIAL,
    section_name VARCHAR(40) NOT NULL,
    UNIQUE (section_name),
    PRIMARY KEY (section_id)
);
```

8. Mapping Publishers -

```
CREATE TABLE Publisher (
    publisher_id SERIAL,
    publisher_name VARCHAR(50) NOT NULL,
    UNIQUE (publisher_name),
    PRIMARY KEY (publisher_id)
);
```

9. Mapping Published_by -

```
CREATE TABLE Published_by (
    book_id INT,
    publisher_id INT NOT NULL,
    PRIMARY KEY (book_id),
    FOREIGN KEY (book_id) REFERENCES Books,
    FOREIGN KEY (publisher_id) REFERENCES Publisher
);
```

10. Mapping Classified_by -
Note: Cannot capture the total participation constraint by Books yet.

```
CREATE TABLE Classified_by (
    book_id INT,
    section_id INT,
    PRIMARY KEY (book_id, section_id),
    FOREIGN KEY (book_id) REFERENCES Books,
    FOREIGN KEY (section_id) REFERENCES Sections
);
```

11. Mapping Written_by -
Note: Cannot capture the total participation constraint by Author yet.

```
CREATE TABLE Written_by (
    author_id CHAR(10),
    book_id INT,
    PRIMARY KEY (author_id, book_id),
    FOREIGN KEY (author_id) REFERENCES Authors,
    FOREIGN KEY (book_id) REFERENCES Books
);
```

12. Mapping Subscribe -

```
CREATE TABLE Subscribe (
    customer_id CHAR(10),
    subscription_id INT NOT NULL,
    start_date TIMESTAMP NOT NULL,
    end_date TIMESTAMP NOT NULL,
    PRIMARY KEY (customer_id),
    FOREIGN KEY (customer_id) REFERENCES Customers,
    FOREIGN KEY (subscription_id) REFERENCES Subscriptions
);
```

13. Mapping Accessed_by -

```
CREATE TABLE Accessed_by (
    book_id INT,
    subscription_id INT,
    PRIMARY KEY (book_id, subscription_id),
    FOREIGN KEY (book_id) REFERENCES Books,
    FOREIGN KEY (subscription_id) REFERENCES Subscriptions
);
```

14. Mapping Payments -
    Note: Cannot capture the total participation constraint by PaymentTransactions yet. Also cannot capture the rightful subscription payment constraint as explained in ER diagram yet.

```
CREATE TABLE Payments (
    transaction_id INT,
    subscription_id INT NOT NULL,
    customer_id CHAR(10),
    PRIMARY KEY (transaction_id, customer_id),
    FOREIGN KEY transaction_id REFERENCES PaymentTransactions,
    FOREIGN KEY subscription_id REFERENCES Subscriptions,
    FOREIGN KEY customer_id REFERENCES Customers
);
```

9 Contingency Plan

Although we don't have any intentions to drop this course, but in worst case situations i.e. in case of emergencies, we would like to have a contingency plan.

In our contingency plan, we would drop the entire idea of subscription based approach. So we would remove the entities/functionalities regarding subscriptions.