COMP 3005 B

NERD BOOKS PROJECT

David Kenyi

100956691

Due on April 14th, 2020

Github link: https://github.com/johnnyalier/Nerd-Books-Database.git

**2.1 Conceptual Design ER-Diagram**

A close up of text on a white background

Description automatically generated

Based on the information given in the problem statement my design revolve around how a bookstore owner would keep information. A user relation is used to store information about bookstore customers. Attributes in this relation are user\_ID (primary key), first\_name, last\_name, username, address, city, province, and postal\_code. The owner would also want to keep information about the publishers of the books they sell. The publishers’ relation has pub\_ID (primary key), name, bank, address, email and phone as attributes. The most important relation in this database is the books relation which has ISBN (primary key), title, author, genre, pub\_ID, pages, price and quantity as attributes. A user places an order to purchase a couple of books and this information is stored in a relation called order. user\_ID, ISBN track\_No, price and quantity are kept per order.

**Assumptions**

* For the simplicity of implementation, users’ entries are assumed to be valid and therefore no error checking is done.
* Since each user has a unique id, it’s used as a login credential though the username and password are more accurate.
* Each book has one publisher for simplicity of payments.
* The email and phone attributes of the publishers’ relation are used to request more copied of the books if quantity is below and threshold.

**2.2 Reduction to Relation Schemas**

* books(ISBN, title, author, genre, pub\_ID, pages, price, quantity)
* publishers(pub\_ID, first\_name, last\_name, bank, address, city, province, postal\_code, email, phone)
* users(user\_ID, first\_name, last\_name, username, password, address, city, province, postal\_code)
* order(user\_ID, ISBN, track\_No, price, quantity)

**2.3 Normalization of Relation Schemas**

To check if the relations above are in good normal form, we need to check if all of the four relation schemas are in BCNF. By the textbook definition, a database design is in BCNF if each member of the set of relation schemas that constitutes the design is in BCNF. Starting with the book relation schema, we see the functional dependency below holds.

ISBN --> title, author, genre, pub\_ID, pages, price, quantity

Because there's no nontrivial functional dependency with any combination of title, author, genre, pub\_ID, pages, price, and quantity without ISBN on the left side of the arrow. therefore, book schema is in BCNF.

Similarly, all attributes in publisher and user relation schemas depend on pub\_ID and user\_ID respectively. These dependencies hold because ISBN, pub\_ID and user\_ID are primary keys of their respective schemas. Hence the publisher and user relation schemas are also in BCNF.

Finally, we need to determine if the order relation schema is also in BCNF. The set of keys formed by user\_ID and ISBN form the superkey for this relation since both keys are foreign keys, they form the primary key for this relation. There are two nontrivial functional dependencies in this schema, and both holds.

ISBN --> price

user\_ID, ISBN --> track\_No, price, quantity

Since ISBN is a foreign key and form a composite primary key, the order schema is in BCNF. We have therefore shown that all members of the relation schemas are in BCNF and thus our bookstore database design is in BCNF.

**2.4 Database Schema Diagram**

A screenshot of a cell phone

Description automatically generated

**2.5 Implementation**

PostgreSQL queries for creating the tables

create table books

(ISBN varchar(10),

title varchar(100),

author varchar(20),

genre varchar(100),

pub\_ID numeric(4, 0),

pages numeric(3, 0) check(pages > 0),

price numeric(3, 0) check(price > 0),

quantity numeric(3, 0),

primary key (ISBN),

foreign key (pub\_ID) references publisher

on delete cascade

);

create table publishers

(pub\_ID numeric(4, 0),

name varchar(100) not null,

bank numeric(9),

address varchar(20),

city varchar(20),

province varchar(2),

postal\_code varchar(6),

email varchar(40),

phone varchar(10) check(phone = 10),

primary key (pub\_ID)

);

create table users

(user\_ID numeric(4, 0),

first\_name varchar(20),

last\_name varchar(20),

username varchar(20) not null,

password varchar(8) check(password = 8),

address varchar(20),

city varchar(20),

province varchar(2) check(province = 2),

postal\_code varchar(6),

primary key (user\_ID)

);

create table orders

(user\_ID numeric(4, 0),

ISBN varchar(10),

track\_No varchar(40),

price numeric(4, 0),

quantity numeric(2, 0),

primary key (user\_ID, ISBN),

foreign key (user\_ID) references users

on delete cascade,

foreign key (ISBN) references books

on delete cascade

);

**Screenshots from the database**

Home page

A screenshot of a cell phone

Description automatically generated

Manager Sign In

A screenshot of a cell phone

Description automatically generated

Manager portal

A screenshot of a cell phone

Description automatically generated

User sign up portal

A screenshot of a cell phone

Description automatically generated

Signing Up User

A screenshot of a cell phone

Description automatically generated

Postgres User table showing Rachel Green added

A screenshot of a computer

Description automatically generated

Books display

A screenshot of a cell phone

Description automatically generated

Books added to the orders relation by user id 9.

Web browser

A screenshot of a cell phone

Description automatically generated

Postgresql orders table

A screenshot of a computer

Description automatically generated

NOTE:

Because of unavoidable circumstances, this implementation isn’t completed. I’ll however continue to work on it for my personal project portfolio.