COMP 3005 Winter 2020 Project

Sharjeel Ali

March 28th, 2020

# 1. Conceptual Design

# 2. Reduction to Relation Schemas

Relations:

1. *author(id, name)*
2. *genre(id, name, type)*
3. *publisher(id, phone, email, address\_id, dd\_id)*
4. *book(isbn, title, description, price, page\_count, published\_date, add\_date, rating, rating\_count, sale\_percent, author\_id, genre\_id, publisher\_id)*
5. *orders(order\_number, username, order\_date, status\_id, card\_number)*
6. *status(status\_id, name, description)*
7. *order\_book(order\_number, isbn, warehouse\_id, quantity)*
8. *cart(username, isbn, quantity)*
9. *client(username, email, first\_name, last\_name, password)*
10. *admin(email, first\_name, last\_name, password)*
11. *sales\_report(report\_number, admin\_email, start\_date, end\_date, file)*
12. *report\_transaction(report\_number, transaction\_id)*
13. *transaction(transaction\_id, transaction\_name, transaction\_type, amount, date)*
14. *warehouse(id, address\_id)*
15. *warehouse\_books(warehouse\_id, isbn, stock)*
16. *address(id, country, state, city, code, street, apartment\_number)*
17. *client\_address(username, address\_id)*
18. *direct\_deposit(dd\_id, transit\_number, account\_number, institution\_number)*
19. *payment\_info(card\_number, name, expiry\_date, security\_code)*
20. *client\_billing(username, card\_number)*
21. *restock(restock\_number, isbn, warehouse\_id, quantity, restock\_date)*
22. *request\_book(request\_number, username, status, request\_name, request\_isbn, date)*
23. *admin\_decides(email, request\_number, decision)*
24. *history(username, isbn, rank)*

test

# 3. Normalization of Relation Schemas

For the following section, all relations will be reduced to BCNF form. For this section, before I reduce, and normalize my relations, the following relations from section 2 above cannot be normalized any further due to them containing only two attributes, where the primary key can determine the other attribute.

Each relation that can be normalized will follow under the following three steps:

1. Check if R is in BCNF Using the simplified test (If *R* has not been decomposed yet)
   1. Find a non-trivial dependency *a 🡪 b* such that it causes a violation of BCNF
   2. Computer *a+*
   3. Verify that *a+* includes all attributes of *R*, proving it is a superkey of *R*.
2. Decompose *R* into BCNF form
3. *author(id, name)*

Functional dependencies:

F = {

­*id 🡪 name*

}

This relation is already normalized due to the following rule:

*For every set of attributes a ⊆ Ri , check that a+ (the attribute closure of a) either includes no attribute of Ri - a, or includes all attributes of R­i*

Because there is only two attributes, it cannot be normalized any further.

1. *genre(id, name, type)*

Functional dependencies:

F = {

Id 🡪 name, type

*genre\_name 🡪 id, type*

}

This relation is already normalized due to the following rule:

*For every set of attributes, a ⊆ Ri , check that a+ (the attribute closure of a) either includes no attribute of Ri - a, or includes all attributes of R­i*

Because there are only two attributes, it cannot be normalized any further.

1. *publisher(id, phone, email, address\_id, dd\_id)*

Functional dependencies:

F = {

*Id🡪 phone, email, address\_id, dd\_id*

*phone🡪 Id, email, address\_id, dd\_id*

*email 🡪 Id, phone, address\_id, dd\_id*

*address\_id 🡪 Id, phone, email, dd\_id*

*dd\_id 🡪 Id, phone, email, address\_id*

}

1. *book(isbn, title, description, price, page\_count, published\_date, add\_date, rating, rating\_count, sale\_percent, author\_id, genre\_id, publisher\_id)*

Functional dependencies:

F = {

*Book🡪 title, description, price, page\_count, published\_date, add\_date, rating, rating\_count, sale\_percent, author\_id, genre\_name, publisher\_id*

}

Note: The rating and ratings\_count. These two although one would assume are dependent upon each other, and therefore not a superkey, these are not functional dependencies of book, due to multiple books having the same count but different ratings and vise versa.

1. *orders(order\_number, username, order\_date, status\_id, card\_number)*
2. *status(status\_id, name, description)*
3. *order\_book(order\_number, isbn, warehouse\_id, quantity)*
4. *cart(username, isbn, quantity)*
5. *client(username, email, first\_name, last\_name, password)*
6. *admin(email, first\_name, last\_name, password)*
7. *sales\_report(report\_number, admin\_email, start\_date, end\_date, file)*
8. *report\_transaction(report\_number, transaction\_id)*
9. *transaction(transaction\_id, transaction\_name, transaction\_type, amount, date)*
10. *warehouse(id, address\_id)*
11. *warehouse\_books(warehouse\_id, isbn, stock)*
12. *address(id, country, state, city, code, street, apartment\_number)*
13. *client\_address(username, address\_id)*
14. *direct\_deposit(dd\_id, transit\_number, account\_number, institution\_number)*
15. *payment\_info(card\_number, name, expiry\_date, security\_code)*
16. *client\_billing(username, card\_number)*
17. *restock(restock\_number, isbn, warehouse\_id, quantity, restock\_date)*
18. *request\_book(request\_number, username, status, request\_name, request\_isbn, date)*
19. *admin\_decides(email, request\_number, decision)*
20. *history(username, isbn, rank)*

Mention discussion with email and username and how it is normalized for admin and client. Assumption that admin emails are constant and cannot be changed, but client ones can.

# 4. Database Schema Diagram

# 5. Implementation

# 6. Bonus Features

1. **Fuzzy search**
2. **Bestsellers**
3. **Recently viewed**
4. **Requesting books**
5. **GUI Webpage**
6. **Recently released**
7. **Download PDF sales reports**

# 7. Github Repository

<https://github.com/SharjeelAliBCS/comp3005W20-project>