University of Pittsburgh

Final Project Report

Model of a Library

Jennifer McIntyre, Ibrahim Qutyan, Hanzhang Bao

INFSCI 1500 SEC 1080

Dr. Robert Perkoski

27 April 2023

**Table of Contents**

Title Page………………………………………………………………………………………………….1

Table of Contents…………………………………………………………………………………………2

Introduction/ Abstract…………………………………………………………………………………..3

UML-compliant E-R Model…………………………………………………………………………….4

Business Rules……………………………………………………………………………………………5

Entity/ Attribute descriptions…………………………………………………………………...…….6 - 8

List of Questions……………………………………………………………………...……………..9 - 11

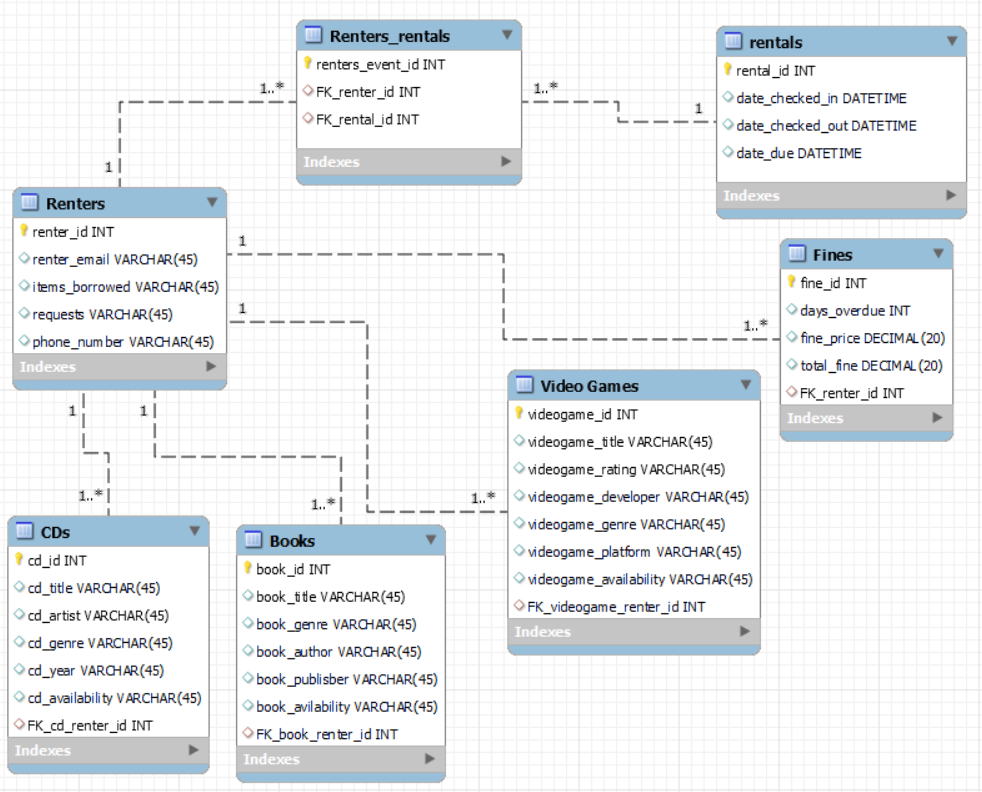
Closing Section……………………………………………………………………………………….…12

**Introduction/ Abstract**

This project focuses on modeling a typical public library dataset, including books, fines, librarians, and renters. A library usually contains items available to check out such as books, cds, and video games. This model allows the user to easily view all the items available in the library, which items are checked out, the price of current fines and who holds them, and various information on workers at the said library. Over the course of this project, group members worked together to determine realistic business practices of libraries to implement into the code and diagram.

The intended user (audience) of this dataset and model would be the manager or company owner of the building as well as the librarians who work there. They would be able to see various important information to keep the library running. For example, John Doe has a fine of $5 based on the rate of 25 cents per day not checked back in. This tells the librarian that they should likely contact John Doe, using the contact information provided in the dataset, in order to remind him of his book fine.

**UML Compliant E-R Model**

****

**Business Rules:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity 1** | **Entity 2** | **Cardinality on Entity 1 Side** | **Cardinality in Entity 2 Side** | **Business Rule(s)** |
| Renters | Cds | 1 | 1..\* | A renter can have multiple cd’s. A cd cannot be owned by multiple renters. |
| Renters | Books | 1 | 1..\* | A renter can have multiple books. A book cannot be owned by multiple renters. |
| Renters | Video\_Games | 1 | 1..\* | A renter can have multiple video games. A video game cannot be owned by multiple renters. |
| Renters | Fines | 1 | 1..\* | A renter could have multiple fines but all those fines belong to one renter. |
| Renters | Renters\_rentals | 1 | 1..\* | A renter has multiple events assigned to them, but events can’t be assigned multiple renters. |
| Renters\_rentals | rentals | 1..\* | 1 | Each event has multiple aspects; all those aspects belong to their respective *singular* event |

**Entity/Attribute Descriptions:**

|  |  |  |
| --- | --- | --- |
| **Entity Name: Renters\_rentals** |  |  |
| renters\_event\_id | INT | Unique identification number for each rental. |
| FK\_renter\_id | INT | Foreign key that refers to renter\_id in Renters entity, indicating which renter is associated with the rental event. |
| FK\_rental\_id | INT | Foreign key that refers to rental\_id in rentals entity, indicating which rental item is associated with the rental event. |
| **Entity Name: Renters** |  |  |
| renter\_id | INT | Unique identification number for each renter. |
| renter\_email | VARCHAR(45) | Renter’s own email address. |
| items\_borrowed | VARCHAR(45) | Items including CD, book, and video games the renter borrowed. |
| requests | VARCHAR(45) | Any special requests or notes made by the renter. |
| phone\_number | VARCHAR(45) | Renter’s own phone number. |
| **Entity Name: CDs** |  |  |
| cd\_id | INT | Unique identification number for each cd. |
| cd\_title | VARCHAR(45) | Title of the CD. |
| cd\_artist | VARCHAR(45) | Artist of the CD. |
| cd\_genre | VARCHAR(45) | Genre of the CD. |
| cd\_year | VARCHAR(45) | Year the CD was released. |
| cd\_availability | VARCHAR(45) | Whether the CD is available or not. |
| FK\_cd\_renter\_id | INT | Foreign key that refers to renter\_id in renter entity, indicating which renter has rented the CD. |
| **Entity Name: Books** |  |  |
| book\_id | INT | Unique identification number for each book. |
| book\_title | VARCHAR(45) | Title of the book. |
| book\_genre | VARCHAR(45) | Genre of the book. |
| book\_author | VARCHAR(45) | Author of the book. |
| book\_publisher | VARCHAR(45) | Publisher of the book. |
| book\_availibility | VARCHAR(45) | Whether the book is available or not. |
| FK\_book\_renter\_id | INT | Foreign key that refers to renter\_id in renter entity, indicating which renter has rented the book. |
| **Entity Name: rentals** |  |  |
| rental\_id | INT | Unique identification number for each rental event. |
| date\_checked\_in | DATETIME | Date and time when the rental item was returned. |
| date\_checked\_out | DATETIME | Date and time when the rental item was borrowed. |
| date\_due | DATETIME | Date and time when the rental item is due for return. |
| **Entity Name: Fines** |  |  |
| fine\_id | INT | Unique identification number for each fine. |
| days\_overdue | INT | Number of days that a rental is overdue when the fine is issued. |
| fine\_price | DECIMAL(20) | Amount of money charged for the fine. |
| total\_fine | DECIMAL(20) | Total amount of money that the renter has to pay for all the fines. |
| FK\_renterf\_id | INT | Foreign key that refers to renter\_id in renters entity, indicating which renter needs to pay the fine.. |
| **Entity Name: Video Games** |  |  |
| videogame\_id | INT | Unique identification number for each video game. |
| videogame\_title | VARCHAR(45) | Title of the video game. |
| videogame\_rating | VARCHAR(45) | Rating of the video game. |
| videogame\_developer | VARCHAR(45) | Developer of the video game. |
| videogame\_genre | VARCHAR(45) | Genre of the video game. |
| videogame\_platform | VARCHAR(45) | Platform of the video game. |
| videogame\_availability | VARCHAR(45) | Whether the video game is available or not. |
| FK\_videogame\_renter\_id | INT | Foreign key that refers to renter\_id in renter entity, indicating which renter has rented the video game. |

**List of Questions**

1. **Which books in the library are fantasy? (SELECT)**

SELECT book\_title, book\_genre, book\_availability FROM Books WHERE book\_genre = ‘Fantasy’;

* This information would be valuable for librarian use if a customer or renter were to ask which fantasy books they could check out.

1. **Which books, cds and videogames are available? (INNER JOIN)**

SELECT

book\_title,

cd\_title,

videogame\_title,

book\_availability,

cd\_availability,

Videogame\_availability

FROM

renters INNER JOIN books ON renters.renter\_id = books.FK\_book\_renter\_id

INNER JOIN cds ON renters.renter\_id = cds.FK\_cd\_renter\_id

INNER JOIN video\_games ON renters.renter\_id = video\_games.FK\_videogame\_renter\_id;

* This would be useful for a user of the library to look up which books, cds, and videogames are available to them

1. **Which books and cds available? (LEFT JOIN)**

SELECT

book\_title,

cd\_title,

book\_availability,

Cd\_availability

FROM

renters LEFT JOIN books ON renters.renter\_id = books.FK\_book\_renter\_id

LEFT JOIN cds ON renters.renter\_id = cds.FK\_cd\_renter\_id;

* This would be useful for a user of the library to look up which books and cds are available.This would be useful for a user of the library to look up which books and cds are available.

1. Two queries using aggregate functions (MAX, MIN ETC)

**4.a) What's the average fine price for all fines? (AVG)**

SELECT AVG(fine\_price) AS avg\_fine\_price

FROM Fines;

* By calculating the average fine price, the library can know the overall effectiveness and fairness of their fine system, and make adjustments as needed.

**4.b) What's the number of rentals for each renter? (COUNT)**

SELECT renter\_id, COUNT(\*) AS rental\_count

FROM Renters\_rentals

GROUP BY renter\_id;

* It helps the library to identify its most frequent renters and determine whether there are any renters who have not used the library's resources for a long time.

1. **How many renters have current requests? (GROUP BY)**

SELECT COUNT(renter\_id), requests

FROM Renters

GROUP BY requests;

* This allows the librarians to view which customers have requests, how many requests there are, and what each customer is requesting.

1. **How many renters have the request for a certain book (HAVING)?**

SELECT COUNT(renter\_id), requests FROM Renters GROUP BY requests HAVING requests = "More Harry Potter series book!";

* This allows librarians to see the popularity of a book by looking at which requests are high in volume

1. **What are the most recent items to be checked out? (ORDER BY)**

SELECT \* FROM rentals

ORDER BY date\_checked\_out DESC;

* This allows the staff to see what items have recently been checked out, perhaps to double check availability of an item or track down an item that is not found in the library.

1. **What are the top 3 highest rated video games, ranked from highest to lowest? (LIMIT)**

SELECT videogame\_title, videogame\_rating FROM video\_games ORDER BY videogame\_rating DESC LIMIT 3;

* This lets patrons of the library to narrow down their choices to the games with the highest ratings

1. **What are the titles of CDs that have been borrowed by renters? (IN)**

SELECT cd\_title FROM CDs WHERE cd\_id IN (SELECT cd\_id FROM CDs);

* The library can determine which items are popular, and which items may need to be replaced, so the library can meet the needs and preferences of its renters.

**Closing Section**

Overall, this project was found by our group to be informative on the basis of further increasing our knowledge about sql code, databases, and business rules. Not only did we learn technical skills in reaction to the code and diagrams we created, but we also learned real world knowledge about how businesses operate. The most difficult part was making sure certain business rules applied to the code. The code can work, but if the business rules do not line up with the concepts, the database has no true purpose. Our group found this project to be insightful and built our collaboration skills for future projects.