Library Management System

Files in project:

Categorized scripts and *main scripts* both folders have same sql script, only difference is categorical distribution of scripts into multiple files.

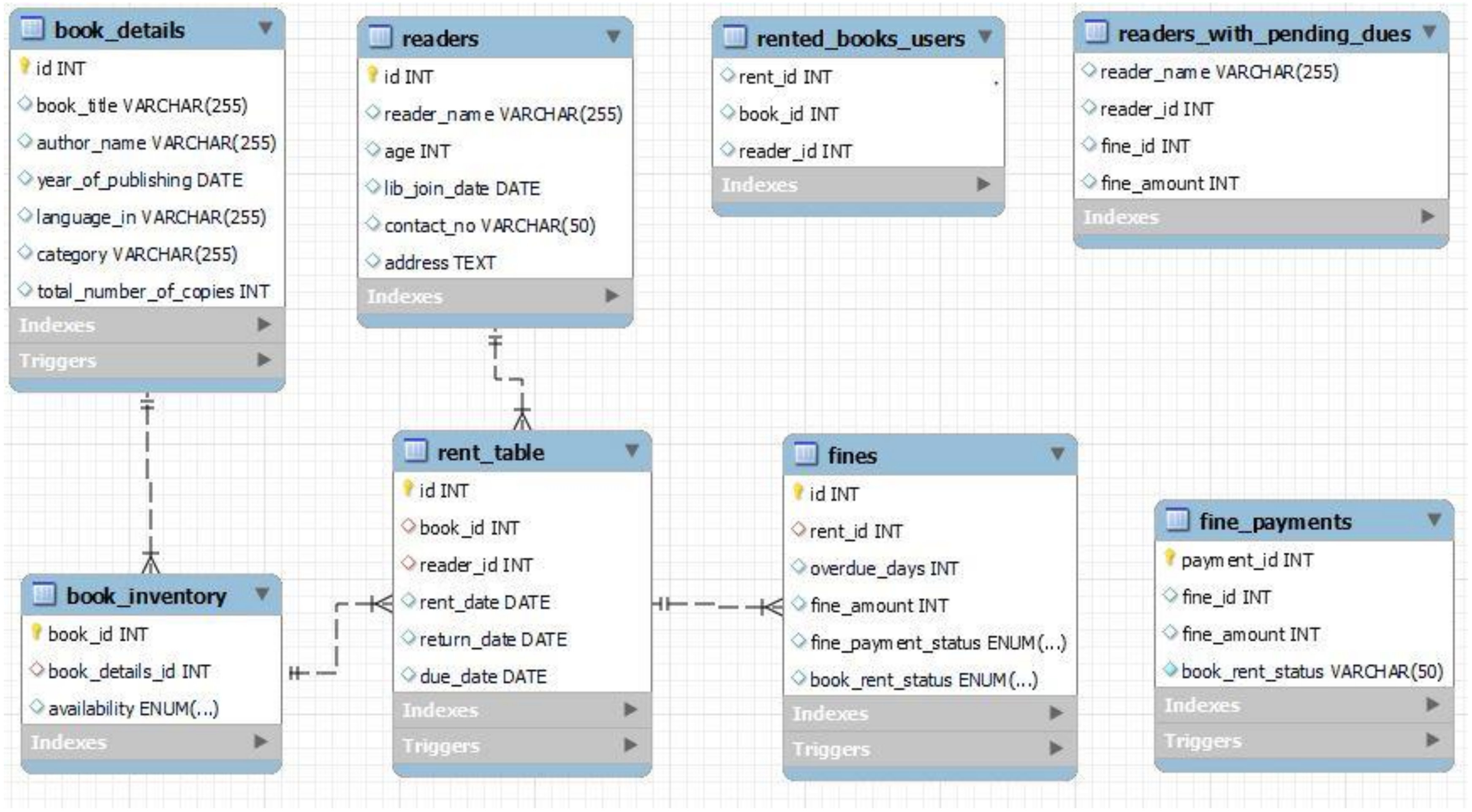
To use Main Scripts folder execute *‘master_script.sql’* file, then use *‘common_tasks.sql’* and *‘common_queries.sql’* file to execute individual queries.

To use categorized scripts folder execute script files in this order ‘*create schema >> triggers >> stored procedures >> events >> insert dummy data,’* then use *‘common_tasks.sql’* and *‘common_queries.sql’* file to execute individual queries.

Librarian Tasks:

- Rent a book_copy to a registered reader
- Return the book in database whenever reader returns it.
- Daily remind readers_with_pending_dues.
- Update payments in datebase for respective fine_ids
- Introduce a new book to library
- Manage inventory

ER Diagram for Library Management System



Schema definition

1. Book_details : Contains details of all the distinct books available in library.
    a. Contains only 1 row for each book.
    b. Number_of_copies column represent total no. of copies of respective book including the ones currently rented
2. Book_inventory : Contains details of all individual book copies in the library.
    a. Can contain multiple copies of same book but maintains availability status of each copy separately.
3. Readers: Contains details of individual readers/customers of library who have at least rented 1 book.
4. Rent_table: Contains detail of the all the book borrow and return transactions.
5. Fines: contain applicable monetary fine details of all the rent_table transactions that exceeded due dates, table is scheduled to update every night, for the purpose of project procedure can be called to update table at any desired time.
6. Rented_books_users: Contain detail of actively rented books at any time.
7. Readers_with_pending_dues: Contain detail of readers who haven't paid their last imposed fine irrespective of whether they have returned the book or not.
8. Fine_payments: Contain details of all the fines for which the payment has been done.

Assumptions:

- New book will be introduced into library/database manually by staff member.
- New copies in Book_inventory will also be updated by staff member, Although, practically each book should be added to inventory individually, along with a unique barcode id imprinted on the book however for the purpose of the project we have created a stored procedure to update inventory directly by total number of new copies to add and book_details_id or book_title.
- A reader can only rent one book at a time.
- Rent period is 7 days and fine will be charged on 9th day if the book is not returned by end of 8th Day.
- overdue fine is Rs 5/Day.
- Payment transactions will be manually updated by staff member.
- New Fine is being calculated in the midnight events with respect to CURDATE() and can also be updated at any given time using procedure ‘midnight_data_refresh()’.
- dummy data has been manually generated for all use cases of the database but for older random dates.

Triggers

- add_new_book_to_inventory_list
  - whenever a new book is introduced to library a row representing that book is inserted into book_inventory
- add_new_book_supplies
  - whenever new copies of already available books are added to library, previously available copies of respective book are added to new number of copies

- rent_a_book
  - sets due date to 7 days from rent date

- add_book_to_rented
  - add user row to rented_books_users
- available_books_copies
  - returns book_copy details for books available for rent, searched by respective book_title
- update_available_availability
  - called whenever a book is returned
  - sets respective book copy available in book_inventory
  - updates fines table if applicable, with respect to return_date
- update_rented_availability
  - whenever a new rent_transaction is added, availability of respective book copy in book_inventory table is updated from "Available" to "Rented"
- update_payment_status
  - called whenever new fine_payment is received,
  - payment status for respective fine in fines table is updated from "Unpaid" to "Paid".
  - Reader row is dropped from readers_with _pending_dues table
  - Sets fine amount in payment table equal to fine amount in fines table for respective fine id/rent id
- Transfer_book_to_rented
  - Whenever a transaction is added to rent_table, total number of copies of respective book in book_details table is reduced by 1
- Update_book_rent_status_in_fines & Update_book_rent_status_in_fines_update

    - Called when either new row is inserted into fines or fines table is updated
    - Fetches overdue_days, fine amount and other details from rent_table wherever fine is applicable

Stored Procedures

- Update_inventory_by_book_title
    - Arguments: Book_title, number of new copies
    - Returns: Adds number of new copies of book to already available number of copies of same book in book_details_table
    - Also triggers insertion of equal number of rows as number of new copies into book_inventory table, to track each newly added copy individually.
- Add_new_book
    - Arguments: book_title, author_name, year_of_publishing, language_in, category
    - Returns: Insert a new entry to book_details table with above arguments as values, i.e. first copy of a new book is added to library
- Add_new_reader
    - Adds new reader to database.
- Rent_a_book
    - Arguments: book_copy_id, reader_id
    - Returns: checks if book_copy is not rented and reader doesn't have any pending books to return, then on true inserts a new transaction to rent_table, with rent_date as current date and due date as 7 days from rent_date.
- Take_payment
    - Takes only fine_id from fines table where fine is due, fetches fine amount from fines table for respective fine_id for which the fine is being paid.
    - use readers_with_pending_dues to check fine_id/fine_amount by rent_id/reader_name
- midnight_data_refresh
    - Executes events scheduled for midnight, i.e. fine table is updated with respect to current date.
- Return_a_book
    - Takes rent_id of any of rented transactions where book is not yet returned, sets return_date to current date for respective rent_id

- Dummy_rent_transaction
    - Arguments: book_id, reader_id, rent_date, return_date, inp_id

- Returns: inserts a rent_table transaction with defined rent_date and return date, used in case of missed rent transaction.

Events

Fine_classifier : Scheduled for everyday 11:59 PM,

A row of is inserted into fines for each of those rent_table transactions which have exceeded their due date (i.e. 7 days from rent_date) to return book on that day and New Overdue days and fine amount is updated in Previous unpaid fines in the fines table.