Project Overview

The Goodreads Database is designed to support a website where book enthusiasts can read, organize, and share their experiences with books. Users can create personal bookshelves, write reviews, and connect with others who share similar interests. The platform aims to foster a sense of community by allowing users to make friends and engage in social interactions around their love of reading. Additionally, users are motivated through various challenges, gifts, and points to encourage continued engagement and participation.

Execution Order

In general we would start with

create_table
indexing
Functions
triggers_before_insertion
Insert
triggers_after_insertion
Views
Queries

However since we have test cases in our functions and triggers that are partially based on the inserted data, we kindly ask to run the code in the following order

1.create_table
2.Insert
3.indexing
4.Functions
5.triggers_before_insertion
6.triggers_after_insertion
7.Views
8.Queries

Entities and Their Key Attributes

1. Users

- Stores information about the platform's users.
- **Key Attributes**: `user_id`, `first_name`, `last_name`, `email`, `username`, `password`, `join_date`, `current_points`

2. Authors

- Contains details about book authors.
- **Key Attributes**: `author_id`, `first_name`, `last_name`, `bio`, `birth_date`, `death_date`, `nationality`

3. Genres

- Defines the genres associated with books.
- **Key Attributes**: `genre_id`, `name`

4. Series

- Represents series of books.
- **Key Attributes**: `series_id`, `name`, `description`

5. Publishers

- Contains publisher information.
- **Key Attributes**: `publisher_id`, `name`, `country`, `city`, `street_number`, `building_number`, `founded_year`

6. Books

- Represents individual books with references to authors, series, publishers, and genres.
- **Key Attributes**: `book_id`, `title`, `description`, `series_id`, `publisher_id`, `author_id`, `genre_id`

7. Awards

- Stores information about book-related awards.
- **Key Attributes**: `award_id`, `name`, `description`, `year_started`

8. Characters

- Details about characters in books or series.
- **Key Attributes**: `character_id`, `name`, `role`, `description`, `series_id`, `book_id`

9. Editions

- Contains information about different editions of a book.
- **Key Attributes**: `edition_id`, `book_id`, `publisher_id`, `publication_date`, `page_count`, `format`, `language`

10. Reviews

- Stores user reviews for books.
- **Key Attributes**: `review_id`, `user_id`, `book_id`, `review_text`, `review_date`

11. Comments

- User comments on reviews.

- **Key Attributes**: `comment_id`, `comment_text`, `comment_date`, `user_id`, `review_id`, `parent_comment_id`

12. Quotes

- User-generated book quotes.
- **Key Attributes**: `quote_id`, `quote_text`, `quote_date`, `book_id`, `user_id`

13. Bookshelves

- Represents a user's collection of books on the platform.
- **Key Attributes**: `shelf_id`, `user_id`, `shelf_name`

14. Challenges

- Represents reading challenges for users.
- **Key Attributes**: `challenge_id`, `name`, `description`, `goal`, `duration`, `points`

15. Group Discussion

- Groups of users discussing books.
- **Key Attributes**: `group_id`, `name`, `description`, `max_people`, `participant_1` to `participant_7`

16. Giveaways

- Stores giveaway events for books.
- **Key Attributes**: `giveaway_id`, `name`, `description`, `cost_in_points`

17. BookshelfBooks

- Tracks books in a user's bookshelf and their reading status.
- **Key Attributes**: `shelf_id`, `book_id`, `status`, `progress`, `last_updated`

18. BookGenres

- Links books to multiple genres.
- **Key Attributes**: `book_id`, `genre_id`

19. BookAwards

- Links books to awards.
- **Key Attributes**: `book_id`, `award_id`, `year_won`

20. Friendships

- Stores information about friendships between users.
- **Key Attributes**: `user_id1`, `user_id2`, `friendship_date`

21. Ratings

- User ratings for books.
- **Key Attributes**: `user_id`, `book_id`, `rating`, `rating_date`

22. Challenge Participation

- Tracks user participation in challenges.
- **Key Attributes**: `user_id`, `challenge_id`, `date_joined`, `state`

23. Giveaway Wins

- Stores information about users who won giveaways.

- **Key Attributes**: `user_id`, `giveaway_id`, `win_date`

24. Book Recommendations

- Represents user book recommendations.
- **Key Attributes**: `user_id`, `book_id`, `reason`

Design Decisions and Normalization

Key Design Decisions

- **Entities and Attributes**:
  Each table was designed to represent a single concept or entity (e.g., Users, Books, Reviews), and the attributes of each table were chosen to be directly related to that entity.
- **Functional Dependencies**:
  The schema follows the given functional dependencies for each table (see in Normalization file). No table contains attributes that are dependent on non-prime attributes that are not directly related to the primary key.
- **Candidate Keys and Superkeys**:
  Each table has a clearly defined candidate key (or multiple keys in some cases). The prime attributes were selected as the attributes that appear in the candidate keys.
  All functional dependencies in the schema are implied by the superkeys, ensuring no violations of the BCNF.
- **No BCNF Violations**:
  All tables were reviewed for potential violations of BCNF, and it was confirmed that there are no such violations in the schema.

Functions

The Goodreads Database includes several important functions that enhance the platform's interaction with users and books. These functions allow you to retrieve essential data such as user information, book ratings, challenge points, and much more.

Key functions implemented include:

1. **GetUserFullName**: Retrieves the full name of a user based on their user ID.
2. **GetBookAvgRating**: Retrieves the average rating of a book based on its book ID.
3. **HasFinishedBook**: Checks if a user has finished reading a particular book.
4. **GetBookCountByAuthor**: Counts the number of books published by a specific author.
5. **GetBooksInShelf**: Counts the number of books a user has in their bookshelf.
6. **GetChallengePoints**: Calculates the total points a user has earned from completed challenges.
7. **GetLastReviewDate**: Retrieves the most recent review date for a particular book.
8. **GetFriendCount**: Counts the number of friends a user has on the platform.
9. **GetGiveawayWins**: Tracks the number of giveaways a user has won.
10. **IsUserInGroup**: Checks if a user is part of a specific group discussion.
11. **GetShelfAdditionsForBook**: Counts how many times a book has been added to users' bookshelves.
12. **GetUserCommentCount**: Counts the total number of comments a user has written on the platform.

For detailed implementation and code, please refer to the functions file.

Triggers

The database includes several triggers that enforce business rules and ensure data integrity within the platform. These triggers are activated before or after specific events, such as inserting or updating data, and are used to enforce rules like preventing users from making invalid actions or ensuring the database reflects the most accurate data.

1. Set the current date as a join date when a user is added

- **Trigger**: `set_user_join_date`
- **Event**: Before INSERT on `Users`

- **Purpose**: Automatically sets the `join_date` to the current date if not provided.
- **Test**: Insert a new user without a join date and verify the current date is set as the join date.

2. Prevent giveaway participation if user lacks sufficient points

- **Trigger**: `prevent_giveaway_if_insufficient_points`
- **Event**: Before INSERT on `GiveawayWins`
- **Purpose**: Raises an error if a user tries to claim a giveaway but doesn't have enough points.
- **Test**: Insert a user with insufficient points for a giveaway and verify the error message.

3. Deduct points from a user after winning a giveaway

- **Trigger**: `deduct_points_on_giveaway`
- **Event**: After INSERT on `GiveawayWins`
- **Purpose**: Updates the user's `current_points` by deducting the points required for the giveaway.
- **Test**: Insert a giveaway win and verify that points are deducted from the user.

4. Update the last updated timestamp when progress is changed on bookshelves

- **Trigger**: `update_bookshelf_timestamp`
- **Event**: Before UPDATE on `BookshelfBooks`
- **Purpose**: Automatically updates the `last_updated` timestamp whenever the `progress` is changed.

- **Test**: Update the progress of a book and verify that the `last_updated` timestamp changes.

5. Prevent duplicate friendship entries

- **Trigger**: `prevent_duplicate_friendships`
- **Event**: Before INSERT on `Friendships`
- **Purpose**: Prevents the creation of a friendship if it already exists in reverse order.
- **Test**: Insert a friendship in reverse order and verify that the error message is raised.

6. Check that group participants don't exceed the maximum

- **Trigger**: `check_group_participants`
- **Event**: Before INSERT on `GroupDiscussion`
- **Purpose**: Raises an error if the number of participants exceeds the maximum allowed.
- **Test**: Try inserting more participants than the allowed limit and verify the error message.

7. Prevent having the same book in the same shelf

- **Trigger**: `prevent_duplicate_book_in_shelf`
- **Event**: Before INSERT on `BookshelfBooks`
- **Purpose**: Prevents a user from adding the same book to the same shelf multiple times.
- **Test**: Insert the same book in the same shelf twice and verify that the error message is raised.

8. Enforce password length constraints

- **Trigger**: `enforce_password_length`

- **Event**: Before INSERT on `Users`
- **Purpose**: Ensures that a user's password length is between 8 and 100 characters.
- **Test**: Insert a user with a password shorter than 8 characters and verify the error message.

9. Prevent deleting a book if it has been quoted or reviewed

- **Trigger**: `prevent_deleting_referenced_books`
- **Event**: Before DELETE on `Books`
- **Purpose**: Prevents the deletion of a book if it has associated quotes or reviews.
- **Test**: Attempt to delete a book that has quotes or reviews and verify that the error message is raised.

10. Automatically set book status to 'reading' if progress is between 1 and 99

- **Trigger**: `auto_set_reading_status`
- **Event**: Before INSERT on `BookshelfBooks`
- **Purpose**: Automatically sets the book status to 'reading' when progress is between 1 and 99.
- **Test**: Insert a book with progress between 1 and 99 and verify the status is set to 'reading'.

11. Ensure users can only quote books they've finished reading

- **Trigger**: `validate_quote_from_read_book`
- **Event**: Before INSERT on `Quotes`
- **Purpose**: Ensures that users can only quote books they have marked as finished.

- **Test**: Try inserting a quote for a book that is not marked as finished and verify the error message.

12. Prevent recommending a book if the user hasn't finished it

- **Trigger**: `prevent_unfinished_book_recommendation`
- **Event**: Before INSERT on `BookRecommendations`
- **Purpose**: Prevents users from recommending books they haven’t finished reading.
- **Test**: Attempt to insert a recommendation for a book that has not been finished and verify the error message.

Indexing Strategy

In a real-world application, the Goodreads Database is expected to handle millions of rows, especially considering that a single book can have more than 10,000 comments. To ensure optimal performance, especially for queries that involve frequent searches based on IDs, we have implemented an indexing strategy.

Key Considerations:

- **Primary keys** are automatically indexed.
- **Indexes** are added to foreign key columns and other frequently queried columns that are not already indexed as part of a primary or unique composite key.

You can have a look at our indexes at index file.

These indexes are designed to speed up search operations and optimize query performance, particularly for foreign key lookups and frequently accessed columns.

Views

The Goodreads Database includes several views that combine and summarize data from multiple tables to optimize common queries and improve data retrieval. Each view serves a specific purpose for the platform, from tracking user activity to providing recommendations based on friends’ ratings.

Key Views:

1. **UserInfo**: Displays each user's total number of reviews, ratings, and points.

2. **ReadingStatus**: Tracks how many books each user has in different reading statuses (currently reading, finished, want to read).

3. **QuotesandAuthors**: Shows user-added quotes along with book titles and author names.

4. **PublicUserInfo**: Displays public user information, hiding sensitive data such as password, email, and points.

5. **UserDiscussion**: Shows the most active users in group discussions based on the number of discussions they participate in.

6. **FriendRecommendations**: Displays books rated 5 stars by a user's friends for friend-based recommendations.

7. **PopularBooks**: Identifies and displays popular books based on review counts and average ratings.

8. **MostPopularBooks**: Filters and shows highly popular books with ratings above 4.5 and review counts exceeding 20.

9. **FriendshipSimiliarity**: Measures the similarity between two friends based on shared bookshelf books and similar ratings.

10. **MostAwardsBooks**: Displays books that have won the most awards.

For more detailed implementation and SQL code, please refer to the views file.

Queries

The Goodreads Database supports a wide range of queries that provide insightful data for both platform administrators and users. These queries cover various scenarios including user activity, book ratings, genre preferences, and user interactions.

Key Queries:

1. **Active Users**
   Users who joined in the last 6 months and have rated more than 3 books.
   Type: Aggregation, Subquery

2. **Books with Higher Ratings in Genre**
   Books with an average rating higher than the average rating of books in the same genre(s).
   Type: Join, Aggregation, Subquery

3. **Highly Rated Books**
   Books with at least 5 ratings and an average rating above 3.
   Type: Join, Aggregation, Filtering

4. **Most Active Reviewer's Top-Rated Book**
   Retrieves the most top-rated book from the most active reviewer.
   Type: Join, Aggregation, Sorting

5. **Books in Multiple Genres**
   Books that belong to both 'Fantasy' and 'Adventure' genres.
   Type: Join, Subquery

6. **Top-Rated Books by User**
   Books that are the highest-rated in a given user's rating history.
   Type: Join, Aggregation, Filtering

7. **Authors Without Reviews**
   Authors who don't have any books reviewed.
   Type: Join, Subquery

8. **Top 5 Users by Review Word Count**
   Shows the top 5 users with the highest total review word count.
   Type: Aggregation, Sorting, Subquery

9. **Books Shelved by Users**
   Shows how many books each user has added to their bookshelf.
   Type: Join, Aggregation

10. **Bonus Points for Challenge Winners**
    Grants 100 bonus points to users who have won a challenge and written more than 3 reviews.
    Type: Update, Join, Subquery

11. **Reviewer Levels**
    Assigns "New", "Intermediate", or "Advanced" labels to users based on how many books

they’ve reviewed.
Type: Aggregation, Conditional Logic (CASE WHEN)

12. **Authors with High Ratings**
Authors whose books have an average rating above 3, and none of their books have a rating below 3.
Type: Join, Aggregation, Filtering

13. **User Ranking by Review Metrics**
Ranks users based on total review length, number of reviews, and average comment count.
Type: Join, Aggregation, Sorting

14. **Books Not Finished**
Books that users have added to bookshelves but where the status is never 'finished'.
Type: Join, Subquery

15. **Award-Winning Books Without Reviews**
Books that have won awards but have never been reviewed.
Type: Join, Aggregation, Subquery

16. **Challenge Success Rate**
Calculates each user’s challenge success rate based on challenges marked as ‘Won’.
Type: Join, Aggregation, Filtering

17. **Series Published by Multiple Publishers**
Identifies series where different books in the same series were published by different publishers.
Type: Join, Aggregation, Grouping

18. **Most Quoted Books with Low Ratings**
Finds the top 5 most quoted books with an average rating below 3.5.
Type: Join, Aggregation, Filtering, Sorting

19. **Suggested Friends Based on Mutual Friends**
Suggests new friends to a user by showing people who share at least one mutual friend.
Type: Join, Aggregation

20. **User Retention Analysis**
Identifies users who joined more than a year ago and have not interacted with the platform in the last 6 months.
Type: Subquery, Aggregation

21. **Users Who Spent the Most on Giveaways**
    Finds users who have spent the most points on giveaways.
    Type: Join, Aggregation, Sorting

22. **Giveaway with Most Winners**
    Identifies the giveaway with the most winners.
    Type: Join, Aggregation, Sorting

23. **Books Quoted but Not Reviewed**
    Finds books that have been quoted but not reviewed.
    Type: Join, Subquery

24. **Books Never Recommended, Quoted, or Reviewed**
    Finds books that have never been recommended, quoted, or reviewed.
    Type: Join, Subquery

25. **Users Who Rated Books but Never Reviewed**
    Identifies users who have rated at least 3 books but never written a review.
    Type: Join, Subquery

26. **Books Finished Relative to Time Since Joining**
    Ranks users based on how many books they've read relative to the time since they joined.
    Type: Join, Aggregation, Sorting

27. **Most Diverse Readers**
    Displays the most diverse readers based on genres.
    Type: Join, Aggregation

28. **Books with High Rating Standard Deviation**
    Finds books with a high standard deviation of ratings (i.e., people either love them or hate them).
    Type: Join, Aggregation, Filtering

29. **Users with Lower Average Rating than Overall Average**
    Identifies users whose average rating is at least 1 point below the overall average rating.
    Type: Join, Aggregation, Subquery

30. **Genres that Attract New Users**
    Identifies genres that attract new users based on the first book they rated.
    Type: Join, Subquery, Aggregation

The database is built using **MySQL**, a powerful relational database management system, to ensure efficient data handling and retrieval for a platform with potentially millions of rows of data.

Team

This project is developed by a team of 3 members:

- **Armen Tamrazyan**
- **Hmayak Paravyan**
- **Meline Mamikonyan**