BookDB Stage 1 and Stage 2

Leib Blam and Avi Parshan

Disclaimer

This library book database system was created solely for educational and demonstration purposes. All data contained within is entirely fabricated and does not represent any real-world information. Any resemblance to actual book titles, authors, or other entities is purely coincidental and unintentional.

Stage 1

Project Proposal

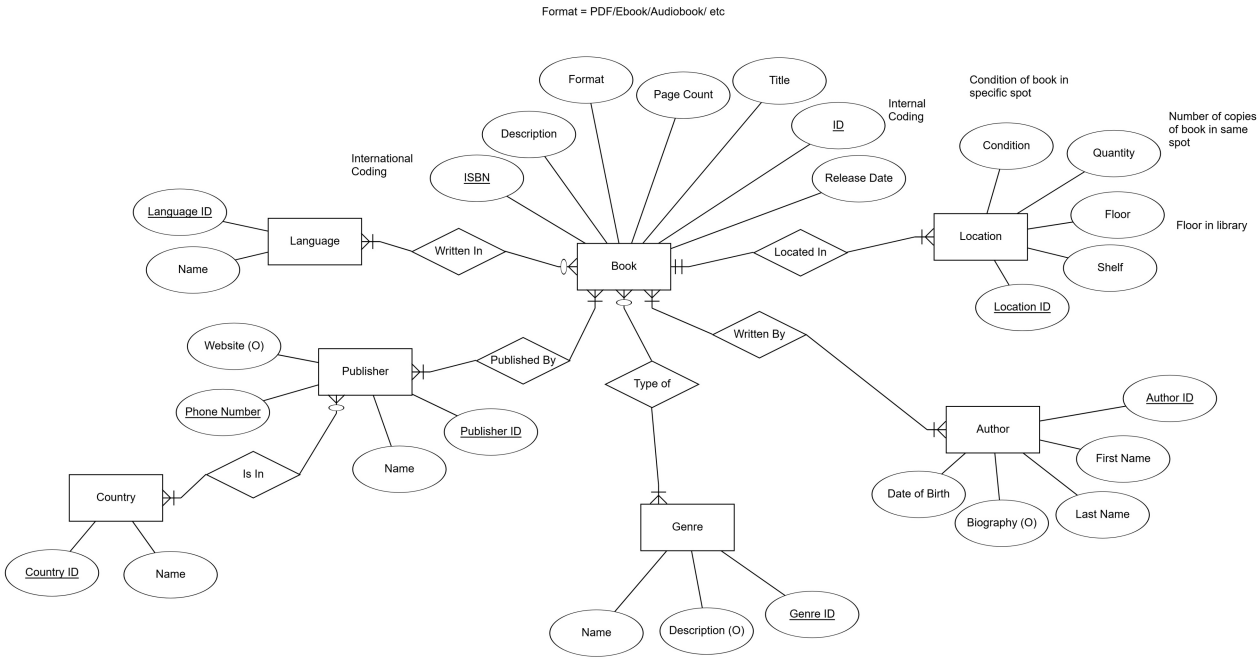
Build a database system to manage books in a library.

Features and Functionalities

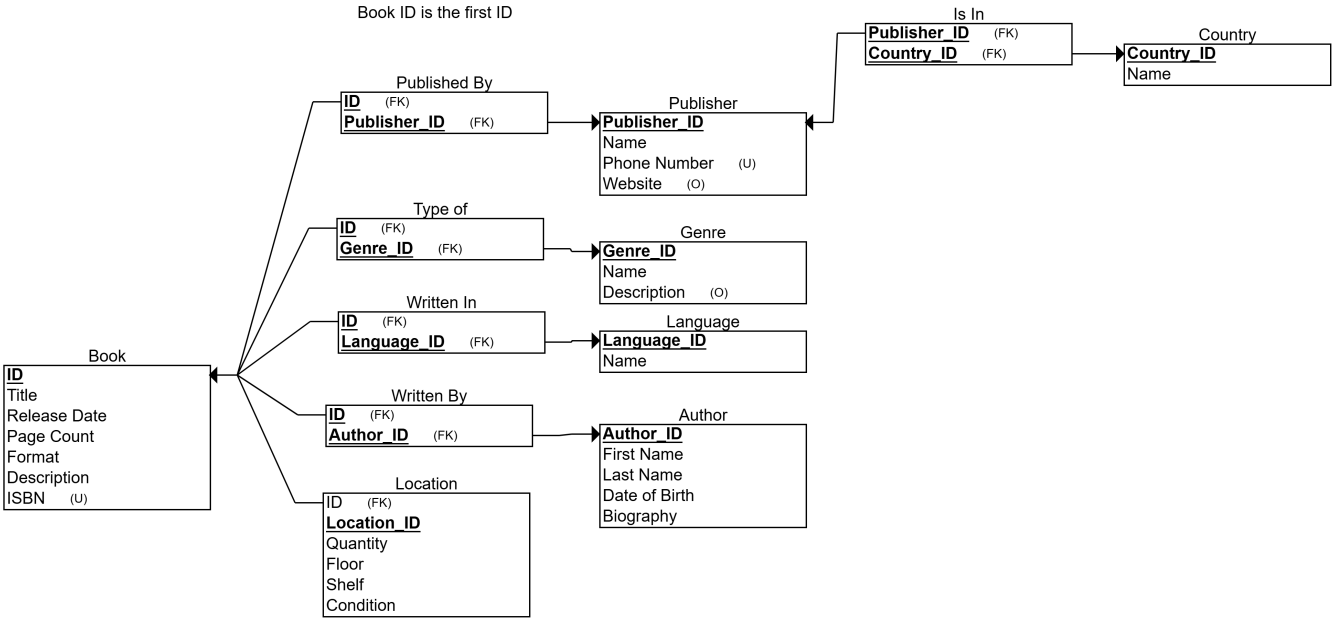
1. **Coding Systems**
    - Internal Code: Unique identifiers for internal processes.
    - International Code: Integration with global standards like ISBN.
2. **Location Tracking**
    - Make it easier for librarians and patrons to find books based on floor and shelf.
3. **Condition Monitoring**
    - Track each book's condition assists librarians with inventory management and overall quality.
4. **Copies Management**
    - Total copies available per title.

Design

- ERD



- DSD



Main Entities:

1. Book
2. Genre
3. Location
4. Language
5. Author
6. Publisher
7. Country

Data Generation

- Schema Definition CreateTables.sql is the script used to create the tables with the required schema.
- Utilizing sampleDataCreation.py we created SQL insert statements that deal with 100.000 Books, 5.000 Authors, 30.000 Publishers, and 70.000 Locations.

Each SQL file can be found in the Data Samples Directory

Run the SQL files in the following order:

1. Schema definition **CreateTables.sql**

2. Data for tables

    1. Country
    - **Independent table**.
    - Script: `random_countries.sql`
    - 24 rows

    2. Publisher
    - **Depends on Country** for the `Is_In` table.
    - Script: `random_publishers.sql`
    - 30,000 rows

    3. Author
    - **Independent table**.
    - Script: `random_authors.sql`
    - 5,000 rows

    4. Language
    - **Independent table**.
    - Script: `random_languages.sql`
    - 62 rows

    5. Genre
    - **Independent table**.
    - Script: `random_genres.sql`
    - 77 rows

    6. Book
    - **Independent table** but referenced by several others.
    - Script: `random_books.sql`
    - 100,000 rows

    7. Location
    - **Depends on Book.**
    - Script: `random_locations.sql`
    - 70,000 rows

    8. Written_By
    - **Depends on Book and Author.**
    - Script: `written_by.sql`
    - 132,406 rows

    9. Published_By
    - **Depends on Book and Publisher.**
    - Script: `published_by.sql`
    - 125,171 rows

    10. Written_In
    - **Depends on Book and Language.**
    - Script: `written_in.sql`
    - 124,727 rows

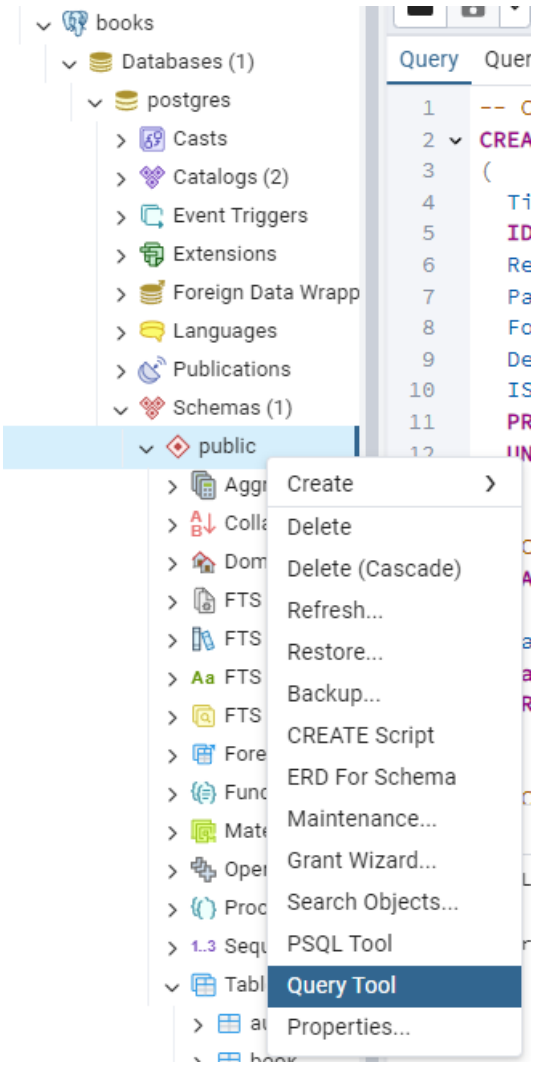
    11. Type_of
    - **Depends on Book and Genre.**
    - Script: `type_of.sql`
    - 124,756 rows

    12. Is_In
    - **Depends on Publisher and Country.**
    - Script: `is_in.sql`
    - 37,048 rows

Table name	Total Size	Tuples inserted	Tuples updated	Tuples deleted	Tuples HOT updated	Live tuples
author	904 KB	5000	0	0	0	5000
book	23.03 MB	100000	0	0	0	100000
country	32 KB	24	0	0	0	24
genre	32 KB	77	0	0	0	77
is_in	2.45 MB	37048	0	0	0	37048
language	32 KB	62	0	0	0	62
location	6.2 MB	70000	0	0	0	70000
published_by	7.92 MB	125171	0	0	0	125171
publisher	4.9 MB	30000	0	0	0	30000
type_of	8.05 MB	124756	0	0	0	124756
written_by	8.48 MB	132406	0	0	0	132406
written_in	7.91 MB	124727	0	0	0	124727

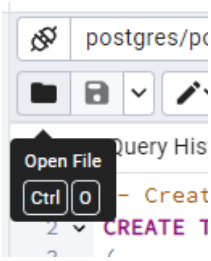
Stage 2

Create Tables in PostgreSQL

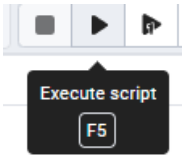


-

Click Query Tool

Now open the CreateTable.sql script via Open File 

And click execute script



Load data

In a similar fashion to the CreateData.sql script, we now bring in each sql file and execute them in the order listed above

Dump Data

Via command line, we can dump the data from the database into a file.

clean to first drop tables

if-exists to avoid errors if tables do not exist

- backupSQL (with DROP . . . CREATE . . . INSERT)

  settled on 1000 rows per insert to balance speed and file size

  create to include create table statements

  inserts to include insert statements

  ```
  pg_dump -U postgres -d postgres -v -f "backupSQL.sql" --create --clean --if-exists --verbose --inserts --rows-per-insert "1000
  ```

  Full dump output is here

- backupPSQL (binary format)

  ```
  pg_dump --file "backupPSQL.sql" --host "localhost" --port "5432" --username "postgres" --format=c --create --clean --if-exists
  ```

  Full dump output is here

Restore Data

- Restore backupPSQL (binary format)

  send logs to be appended to original log file, disable triggers helps us avoid future constraint issues with order of insertion into the table (preventative measure)

  no owner and no privileges to avoid potential issues with permissions

  ```
  pg_restore --host "localhost" --port "5432" --username "postgres" --dbname "postgres" --clean --if-exists --disable-triggers
  ```

Basic Queries

- found here

SELECT:

- (Query 1) Find the 5 oldest authors (Name, DOB) who published at least 1 book in the database
- (Query 2) Get average amount of books published by any publisher
- (Query 3) Get publisher (ID) with books published in the most languages
- (Query 4) Get book title and quantity of book with the lowest quantity in stock

UPDATE:

- (Query 5) Change all books released between 1999-12-28 and 1999-12-31 to 2000-01-01

- (Query 6) Move all returned Children's books from Returns that are in decent condition to the Kids Corner

DELETE:

- (Query 7) Delete books with 0 copies that are moldy or damaged
- (Query 8) Delete all books written in russian that have more than 90 copies in stock

Parameterized Queries

- found here
    - (Query 9) Top N prolific authors in a genre (who wrote the most books in that genre)
    - (Query 10) Get the top N books written in a specified language
    - (Query 11) Get all publishers associated with a specified book
    - (Query 12) Lists all books published by a specified publisher

Indexing

To optimize the performance of the previous parameterized queries, we should create indexes on the columns that are frequently used in JOIN operations, WHERE clauses, and ORDER BY clauses. Columns to index:

- Genre.Name (for filtering the genre)
- Written_By.Author_ID (for the join with the Author table)
- Written_By.ID (for joining with the Book table)
- Type_of.Genre_ID (for the join with the Genre table)
- Book.ID (for joining with the Written_By and Type_of tables)
- Language.Name (for filtering by language)
- Written_In.Language_ID (for the join with the Language table)
- Written_In.ID (for the join with the Book table)
- Book.Title (for filtering by book title)
- Published_By.Publisher_ID (for the join with the Publisher table)
- Published_By.ID (for the join with the Book table)
- Publisher.Publisher_ID (for joining with the Published_By table)
- Publisher.Name (for filtering by publisher name)
- File is found here

Timing

Query Number	Normal Runtime (ms)	Runtime With Indexes (ms)
1	280.25	257.85
2	179.344	178.423
3	520.412	514.267
4	15.532	10.922
5	26.422	12.63
6	35.571	31.103
7	79.197	78.133
8	250.913	152.8542844
9	541.529	31.633
10	348.897	25.609
11	97.931	9.133

Query Number	Normal Runtime (ms)	Runtime With Indexes (ms)
12	73,311	1,294

- Logs and queries are found here

Constraints

To make our database system more sturdy, we enforced the following rules:

- Book quantity is minimum of 0
- Book page count is minimum of 1
- ISBN is unique
- Release date for books is not in the future
- Author birth date is not in the future
- Every book has a publisher
- Shelf number is minimum of 0

To test the constraints, we attempted to break them as follows:

- Negative book quantity
- Negative page count
- Duplicate ISBN
- Future release date
- Future author birth date
- Book without a publisher
- Negative shelf number

Files and logs found here