Database Project

Paperstacks

https://paperstacks.herokuapp.com/

Project Step 3, Draft Version: HTML Interface

CS 340 - Introduction to Databases

Oregon State University, Winter 2020

Team #25: “The Considerate Hedgehogs”

Heather DiRuscio & Kayla Smith

@wrongenvelope @smithkaylar

February 10, 2020

Quick Contents:

URL for Review: https://paperstacks.herokuapp.com/

Site Map:

	app route:
/	(home.html)
|_ about/	(about.html)
|_ search/	(search.html)
|_ genres/	(genres.html)
|_ genre/	/sample_genre: (genre/<id>/)
|_ add genre/	(add_genre.html)
|_ remove genre/	(rem_genre.html)
|_ books & authors/	(books.html)
|_ book/	(sample_book.html)
|_ add book/	(add_book.html)
|_ remove book/	(rem_book.html)
|_ author/	(sample_author.html)
|_ add author/	(add_author.html)
|_ remove author/	(rem_author.html)
|_ rate_and_review	(add_review.html)
|_ reviews/	(sample_review.html)
|_ ratings/	(sample_rating.html)
|_ privacy/	(privacy.html)
|_ 404/	(404.html)

Database Entities Mapped to Subpages

Books ⇒	sample_book.html	(book/<id>)
Genres ⇒	sample_genre.html	(genre/<id>)
Authors ⇒	sample_author.html	(author/<id>)
Ratings ⇒	sample_review.html	(review/<id>)
Reviews ⇒	sample_rating.html	(rating/<id>)

Feedback by Peer Reviewers

Feedback #1: Fewer Books

"For the scope of this class, I think you can just take at most 10 books as testing samples in the database."

We decided on 50 as an initial starting point to make the database into a more functional product, but may choose to start with a smaller subset for testing purposes and expand later.

Feedback #2: Typo

"Just a small typo in Books -> genre_id: varchar, FK, I believe it should be type int, because in Genres -> genre_id: int, not NULL, unique, PK."

Yes, this was a typo. Good catch!

Feedback #3: Reconsider the Relationship Between Books and Genres

"A book can be in one or more genres, but a genre must be at least one of the books? Do you mean that a genre must be related to at least one book before they can be entered in the database just like the note on Authors? I think it would be a better idea to make a note on it."

Yes, we can update this so that genre is associated with 0 or more books. This had been our original intention, but it was not reflected correctly in the ER diagram.

Feedback #4: Rename Compound Entity Tables

- **Instance 1 of this Suggestion:** *"But I think we should treat the joint tables as an entity, so that "book_author" should be named like "Books_Authors"?"*
- **Instance 2 of this Suggestion:** *"The linker tables could be named with the plurals, like book_author needs to be changed to books_authors but that could be just me*

nitpicking."

It had been intentional to list our compound entities book_author and genre_book to delineate them from our base tables, but after further reflection we decided to update our Schema to reflect these suggestions for greater clarity and style cohesion.

Feedback #5: Split Attribute into Multiple Smaller Attributes

- **Instance 1 of this Suggestion:** *"Note: For the Authors entity, it would still be a better idea to separate their names into first, middle, and last name if you will implement a search function by name."*
- **Instance 2 of this Suggestion:** *"I agree with the statement above, since there are so many books, you could think about adding middle name to discern differences. I'll edit as I look more but nice job!"*

No. This can be implemented with a substring search so there is absolutely no reason to break these out into multiple attributes. As we stated in our initial plan, an author's name is essentially a brand in ways that a legal name for other uses is not. We have discussed this in-depth and feel confident that our initial design choice is correct for the use here.

Feedback #6: Ensure Cohesion Between ER Diagram and Document

"In the outline, for each entity its purpose is listed. Under each entity, attributes along with each attribute data type are listed. Each entity describes its relationship with other entities, except Books -> Reviews in Books entity and Books -> Ratings in Books entity—both are depicted in ERD but not in outline."

Excellent catch—updated.

Feedback #7: Add Constraints On Data Values

"Consider adding a constraint for star_rating (e.g. 1 to 5 stars). Also, for each primary key, except "Books", specify where each id will start—for example, author_id will auto increment, starting at 1 or 100."

We can implement a constraint for star_rating. We can start primary keys at 100.

Feedback #8: Remove Foreign Keys

"Design Question 1: Can the attribute, "review_id" be NULL under Ratings since "review_id" is used as FK? You have "review_id" listed as the primary key under Reviews entity, and that value cannot be NULL. Conceptually, I understand that someone can leave a rating without a review and vice versa. A solution I think (but to disclaim I didn't think too deeply about this) is to omit "review_id" since the relationship with Reviews is optional, and to do the same for "rating_id" under Reviews entity since the relationship with Ratings is optional."

We do need this relationship intact; we updated this to be optional both ways. However, we cannot omit Reviews from Ratings nor Ratings from Reviews, as that would sever the relationship entirely.

Feedback #9: Add Average Rating Attribute

"Design Question 2: You mentioned in the Overview section that users can find books based on average rating. I don't see an average rating attribute in the outline, ERD, or schema. From this, how do you plan to capture/list average rating for a book?"

This can be calculated on the fly based using an SQL query on the star Ratings themselves and the number of entries in the database. We will leave this as-is.

Initial Feedback by Grader

In our first draft, we did not receive any feedback regarding improvements, however we decided upon looking at our schema that we would need to add ISBN as a Foreign Key to both Author and Reviews in order to properly link them. This change is reflected in our project plan.

Actions Taken Based on Feedback

We made a minor update to our Schema, and fixed a couple of typos in our ER Diagram and Design Document. Additionally, we incorporated suggestions about the relationship between Books and Genres, so that a Genre may be associated with 0 or more Books. This is in the event that a Genre is added to the database but not associated with any Books.

There were also suggested changes that we did not implement:

- The suggestion to break out author_name into first, middle, and last name attributes was disregarded - we can implement a search function with a substring search, and authors will not always have full names to populate all three fields (or even two).

- The suggestion to store the data about average review as its own entity or attribute somewhere was also disregarded, as this can be calculated when the database is called based on the ratings and the number of entries in the table. For the scope of this class we are confident that these calculations will not take long, however in a production-level industry environment we might choose to implement this in a different way if the calculation proved lengthy to execute.

- We did not remove the relationship between Ratings and Reviews. We want the two entities to be linked together via foreign keys if the user inputs both a Rating and a Review for a Book, but we do not want to enforce a mandatory relationship. By severing the relationship we could not represent when ratings and reviews are tied together, and by design we want that relationship to be maintained if the user submits it that way.

Revisions to Draft Version (Project Step 2, Final Version)

Summary of revisions:

- **Updates to Schema:** The compound entities are now listed as plural and capitalized (e.g. Books_Genres), based on Feedback #3.
- **Updates to Document:** We fixed an attribute that was mislabeled as varchar instead of int, based on Feedback #6.
- **Updates to Project Design:** We reconsidered number of books to include in first iteration of database, potentially to make project more manageable, based on Feedback #1. Our team may also confer further with instructors to determine an appropriate project scope.
- **Updates to ER Diagram:** We fixed typos in our ER Diagram (Feedback #2), and determined it would be appropriate to add constraints to the star rating of Ratings (Feedback #7). Additionally, we also added in starting values for our integer attributes that auto-increment (also Feedback #7). Lastly, we correct the relationship between Books and Genres (Feedback #3).

Project Plan: Overview

Hundreds of thousands of books are published each year in the United States. This can easily leave a would-be bookworm overwhelmed in a sea of choice. Our database-driven website, Paperstacks, will enable users to find books based on attributes like title, genre, author, year published, and average rating. In addition, they will be able to leave their own ratings and reviews on books they have read as well as see reviews by others. This eliminates much of the guesswork for a bookworm looking for their new favorite book.

A production environment database for such a project would be absolutely massive - to document even a fraction of the approximately 130 million books in existence [source] would be a monumental undertaking. For the scope of this class we plan to focus on around 50 books of varied genres, authors, and publishing dates.

We will assign multiple (3-4) simulated reviews and ratings to each book for demonstration purposes. From there, the website and database would record input by users adding books, ratings, and reviews and include search functionality of books that already exist in the database. Our goal is to connect bookworms to books they want to read, period.

Project Plan: Database Outline

Books (Records details of books entered into the database)

- **isbn:** int, not NULL, unique, PK
- **book_title:** varchar, not NULL
- **author_id:** int, not NULL, unique, FK
- **year_published:** int, not NULL
- **genre_id:** varchar, FK

- **Relationship:** **M:M between Books and Authors**
  - implemented with author_id as FK inside Books.
- **Relationship:** **M:M between Books and Genres**
  - implemented with genre_id as FK inside Books.
- **Relationship:** **1:M between Books and Ratings**
  - implemented with rating_id as FK inside Books
- **Relationship:** **1:M between Books and Reviews**
  - implemented with review_id as FK inside Books

- **Note on ISBN:** While populating sample database information, our team discovered that ISBN only identifies one *edition* of one print version of a book title (e.g. 6th edition hardcover release). Additionally, there are two formats for ISBNs: ISBN-10 and ISBN-13. ISBN-10 can have a leading zero, which is not recorded correctly when stored as an integer. Currently, we plan to limit book titles to one ISBN-10 identifier for database simplicity, but we may also assign a unique integer ID and remove ISBNs altogether to ensure that website users do not erroneously believe a title to be absent from our database as a result of an ISBN search.

Reviews (Records details of each user review)

- **review_id:** int, not NULL, unique, PK, count starting at 100
- **rating_id:** int, can be null, FK
- **isbn:** int, not NULL, unique, FK

- **Relationship: 1:M between Reviews and Books**
  - implemented with isbn as FK in Reviews
  - mandatory participation
- **Relationship: 1:1 between Ratings and Reviews**
  - implemented with review_id as FK in Ratings
  - partial participation

★★★★☆ Ratings (Records details of each user star rating)

- **rating_id:** int, not NULL, unique, PK, count starting at 100
- **review_id:** int, can be null, FK
- **isbn:** int, not NULL, unique, FK
- **star_rating:** int, not NULL, constraint: must be in range from 1-5

- **Relationship: 1:M between Ratings and Books**
  - implemented with isbn as FK in Ratings
  - mandatory participation
- **Relationship: 1:1 between Ratings and Reviews**
  - implemented with review_id as FK in Ratings
  - partial participation

✍ Authors (Records details of authors entered into the database)

- **author_id:** int, not NULL, unique, PK, count starting at 100
- **author_name:** varchar, not NULL
  - Note: We intentionally decided to keep the author name as all one attribute rather than breaking out to first, middle, and last name. This is because a full author name is akin to a brand name in a way that an individual person's name is not.

- **Relationship:** M:M between Authors and Books
  - implemented with isbn as FK in Authors
  - **Note:** We determined that an instance of Authors is in a mandatory relationship with Books, meaning that an author must have written at least one book (or be authoring an upcoming book) before they can be entered in the database.

📖 Genres (Records book genres)

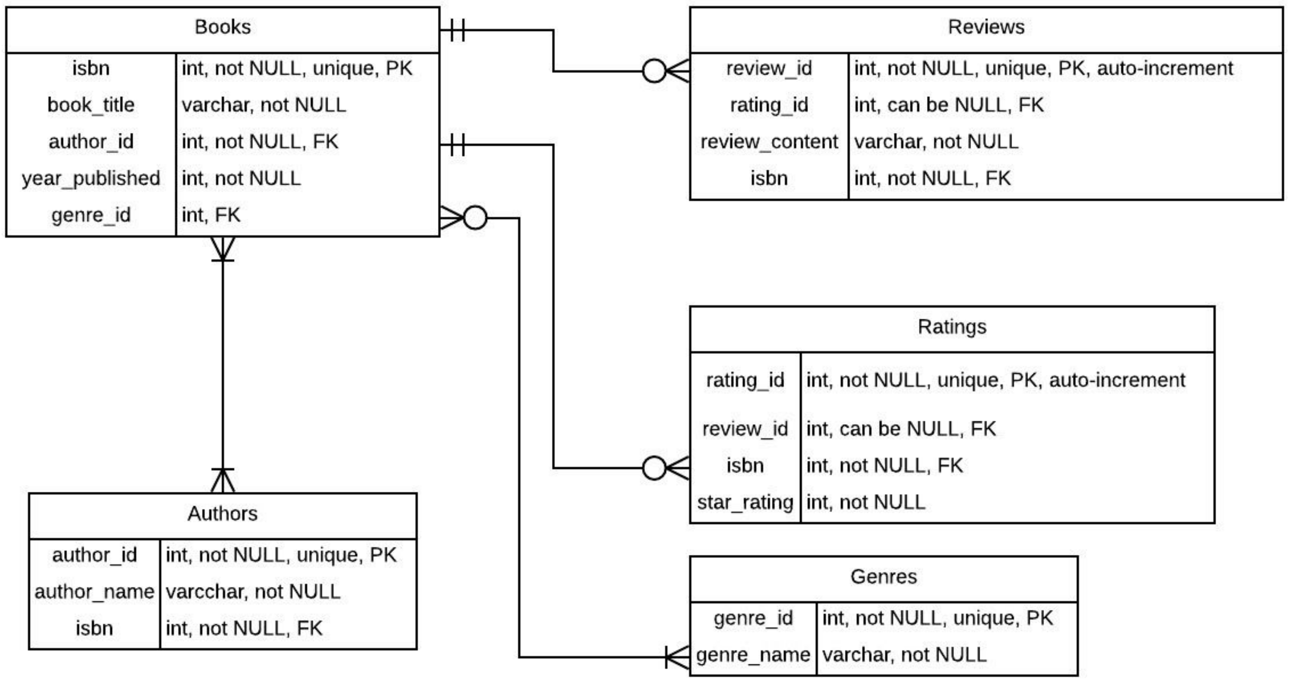
- **genre_id:** int, not NULL, unique, PK, count starting at 1
- **genre_name:** varchar, not NULL

- **Relationship:** M:M between Genres and Books
  - implemented with genre_id as FK in Books
- **Relationship:** M:M between Genres and Authors
  - implemented with genre_id as FK in Authors

Note: Rather than get bogged down in an exhaustive list of every possible genre, our team will select a subset of popular genres for the database. This may result in minor reclassifications (e.g. a book filed under "Wine" may be considered most closely related to "Cooking"), or genre aggregations (e.g. "Mystery" may become "Mystery & Thriller.")

ER Diagram

Crow’s Foot Notation of Entity Attributes and Relationships



Books	
isbn	int, not NULL, unique, PK
book_title	varchar, not NULL
author_id	int, not NULL, FK
year_published	int, not NULL
genre_id	int, FK

Reviews	
review_id	int, not NULL, unique, PK, auto-increment
rating_id	int, can be NULL, FK
review_content	varchar, not NULL
isbn	int, not NULL, FK

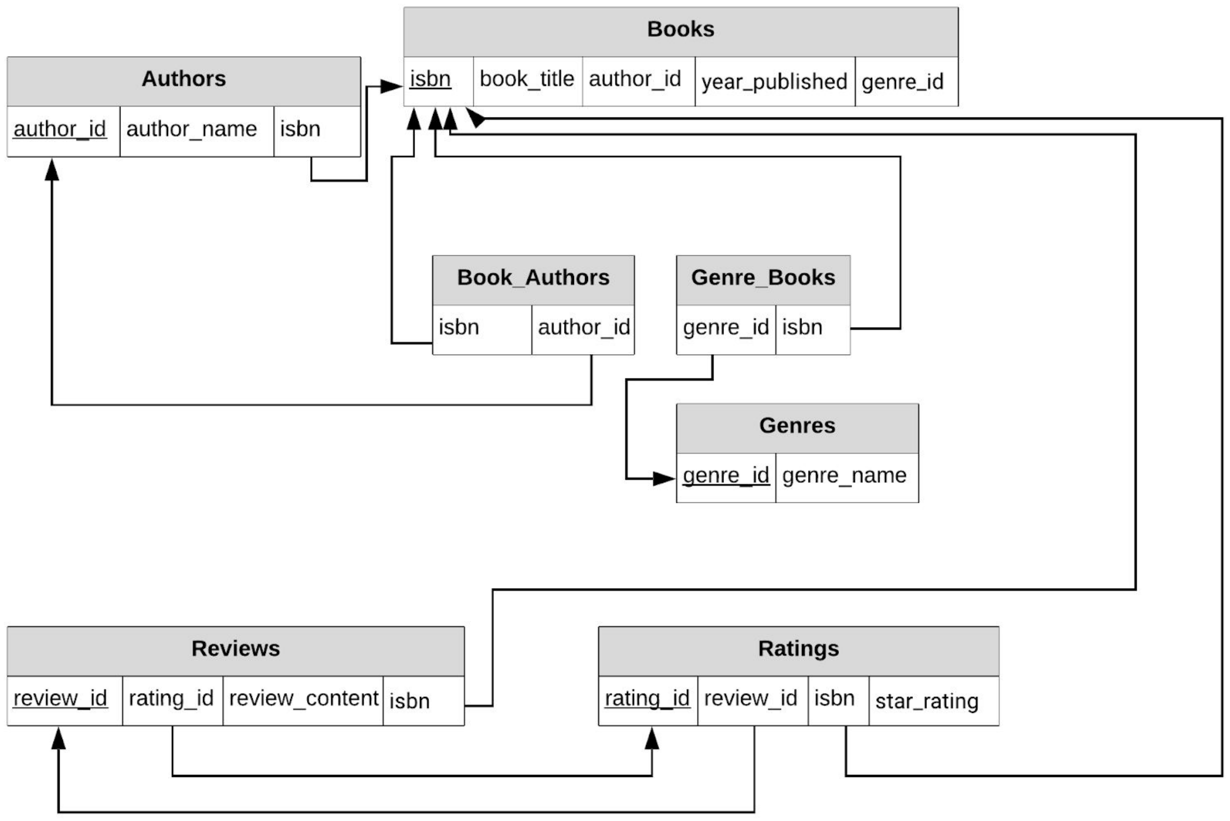
Ratings	
rating_id	int, not NULL, unique, PK, auto-increment
review_id	int, can be NULL, FK
isbn	int, not NULL, FK
star_rating	int, not NULL

Authors	
author_id	int, not NULL, unique, PK
author_name	varcchar, not NULL
isbn	int, not NULL, FK

Genres	
genre_id	int, not NULL, unique, PK
genre_name	varchar, not NULL

Database Schema

- Visualization of Relationships Between Primary and Foreign Keys
- Resolution of Many-to-Many Relationships with Compound Entities



Note: Underline indicates primary key. In tables with no underlines, the combination of both foreign keys is the primary key.