Smith's Library

Brian Jahnke

Executive Summary

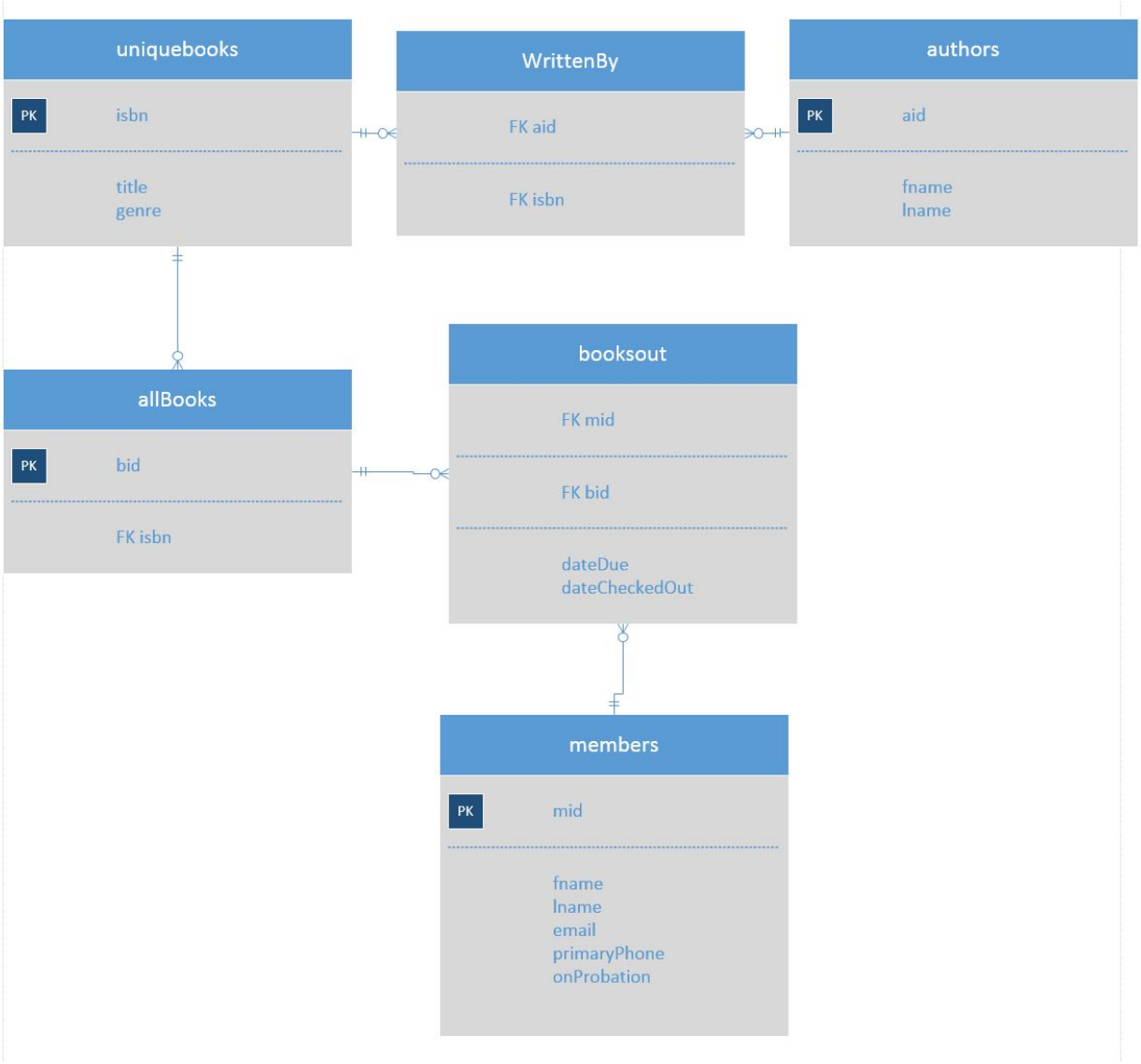
Overview:

Public libraries act as invaluable sources of information and literature in communities across the globe. Millions of people depend on libraries for access to literature they would otherwise have to pay for. While these establishments are committed to serving the information needs of the public, if they are unable to run in an efficient manner, their purpose would be lost and they would fall out of use.

Objectives:

The purpose of this document is to outline a database system which keeps records on every books within a given library as well as interactions between members and the library. On top of this, this document provides tools that can aid in the self gauging of the library's efficiency. The higher purpose is to provide a system for libraries to run in an efficient manner that allows users to find the books they need with ease and to maintain maximum book availability.

Entity Relationship Diagram



Tables

UniqueBooks

Purpose:

This table is used hold the basic information of every new book introduced to the library's database.

Create Statement:

```
CREATE TABLE uniquebooks
(
        isbn int NOT NULL,
        title text NOT NULL,
        genre text NOT NULL,

        PRIMARY KEY(isbn)
);
```

Functional Dependencies:

Isbn → title, genre

Sample Data:

	isbn integer	title text	genre text
1	17713	The Count of Monte Cristo	Adventure
2	29527	The Lord of the Rings	High Fantasy
3	11000	The Hobbit	High Fantasy
4	16017	The Alchemist	Fantasy
5	23608	House to House	Autobiography
6	12525	How I Got a Nuke on MW2	Autobiography
7	98821	American Sniper	Autobiography

AllBooks

Purpose:

This table shows every book in the library's system including multiple copies of the same book.

Create Statement:

```
CREATE TABLE allBooks
(
        bid SERIAL PRIMARY KEY,
        isbn int NOT NULL REFERENCES uniqueBooks(isbn)
);
```

Functional Dependencies:

bid → isbn

Sample Data:

	bid integer	isbn integer
1	1	12525
2	2	12525
3	3	17713
4	4	29527
5	5	29527
6	6	23608
7	7	98821
8	8	16017
9	9	11000
10	10	11000
11	11	11000

Members

Purpose:

Used to store the name and contact information of every member of this library.

Create Statement:

```
CREATE TABLE members
(
        mid SERIAL PRIMARY KEY,
        fname text NOT NULL,
        lname text NOT NULL,
        email text NOT NULL,
        primaryPhone text NOT NULL,
        onProbation boolean NOT NULL
);
```

Functional Dependencies:

Mid → fname, lname, email, primaryPhone, onProbation

Sample Data:

	mid integer	fname text	lname text	email text	primaryphone text	onprobation boolean
1	1	Jal	Gandhi	gandhi@email.com	631-555-1349	f
2	2	John	DiBlasi	diblasi@email.com	631-555-2326	f
3	3	Lindsey	Hans	hans@email.com	631-555-2968	f
4	4	James	Variano	variano@email.com	631-555-7325	f
5	5	Casey	Connolly	connolly@email.com	631-555-4958	f
6	6	Emily	Grupski	grupski@email.com	631-555-1095	f

Authors

Purpose:

This table stores the author of every book in the library's system.

Create Statement:

```
CREATE TABLE authors
(
        aid SERIAL PRIMARY KEY,
        fname text NOT NULL,
        lname text NOT NULL
);
```

Functional Dependencies:

aid → fname, lname

Sample Data:

	aid integer	fname text	lname text
1	1	Jordan	Garafolo
2	2	Alexandre	Dumas
3	3	J.R.R	Tolkien
4	4	David	Bellavia
5	5	John	Bruning
6	6	Chris	Kyle
7	7	Scott	McEwen
8	8	Jim	DeFelice
9	9	Paulo	Ceolho

BooksOut

Purpose:

This table shows which books are currently checked and displays the member who is currently borrowing it by their mid.

Create Statement:

```
CREATE TABLE booksOut
(
        mid int NOT NULL REFERENCES members(mid),
        bid int NOT NULL REFERENCES allBooks(bid),
        DateofCheckout Date,
        DateDue Date
);
```

Functional Dependencies:

Mid, bid → DateofCheckout, DateDue

Sample Data:

	mid integer	bid integer	dateofcheckout date	datedue date
1	1	11	2016-04-01	2016-04-15
2	1	8	2016-04-05	2016-04-19
3	2	7	2016-04-03	2016-04-17
4	3	6	2016-04-07	2016-04-21
5	4	5	2016-04-10	2016-04-24
6	5	4	2016-04-06	2016-04-20
7	6	3	2016-04-03	2016-04-17

Views

Books_members_have

Purpose:

It is important to present a list of all the books that are currently being borrowed in a way that is easily comprehensible by the librarians.

View:

```
CREATE OR REPLACE VIEW books_members_have AS
  SELECT ub.title, m.fname, m.lname, m.mid
  FROM booksout bo
  INNER JOIN members m ON bo.mid = m.mid
  INNER JOIN allbooks ab ON bo.bid = ab.bid
  INNER JOIN uniquebooks ub ON ab.isbn = ub.isbn;
```

Output:

	bid integer	title text	dateofcheckout date	datedue date	fname text	lname text	mid integer
1	11	The Hobbit	2016-04-01	2016-04-15	Jal	Gandhi	1
2	8	The Alchemist	2016-04-05	2016-04-19	Jal	Gandhi	1
3	7	American Sniper	2016-04-03	2016-04-17	John	DiBlasi	2
4	6	House to House	2016-04-07	2016-04-21	Lindsey	Hans	3
5	5	The Lord of the Rings	2016-04-10	2016-04-24	James	Variano	4
6	4	The Lord of the Rings	2016-04-06	2016-04-20	Casey	Connolly	5
7	3	The Count of Monte Cristo	2016-04-03	2016-04-17	Emily	Grupski	6

Books_past_due

Purpose:

In order to stay efficient, a library must be able to keep track of all overdue books in its system. This view allows the librarians to see every overdue book as well as the current holder of the book so that they may resolve the issue as easily as possible.

View:

```
DROP VIEW IF EXISTS books_past_due;
CREATE OR REPLACE VIEW books_past_due AS
SELECT bid, title, dateDue, CAST(CURRENT_TIMESTAMP AS DATE)-dateDue AS
daysPastDue, fname, lname, mid
FROM books_members_have
WHERE dateDue < CAST(CURRENT_TIMESTAMP AS DATE);
```

Output:

	bid integer	title text	datedue date	dayspastdue integer	fname text	lname text	mid integer
1	11	The Hobbit	2016-04-15	20	Jal	Gandhi	1
2	8	The Alchemist	2016-04-19	16	Jal	Gandhi	1
3	7	American Sniper	2016-04-17	18	John	DiBlasi	2
4	6	House to House	2016-04-21	14	Lindsey	Hans	3
5	5	The Lord of the Rings	2016-04-24	11	James	Variano	4
6	4	The Lord of the Rings	2016-04-20	15	Casey	Connolly	5
7	3	The Count of Monte Cristo	2016-04-17	18	Emily	Grupski	6

Searchable_genres

Purpose:

It is useful to know all the genres a library holds so that members may search the library's database by some personal preference.

View:

```
CREATE OR REPLACE VIEW searchable_genres AS
SELECT DISTINCT genre
FROM uniqueBooks
ORDER BY genre ASC;
```

Output:

	genre text
1	Adventure
2	Autobiography
3	Fantasy
4	High Fantasy
5	Poetry

Author_books

Purpose:

Since the only thing tying books to their authors is a weak entity table pairing the isbn and the author id, it would be useful for librarians to be able to see this list in a format that is comprehensible to someone who does not know the format of the database.

View:

```
CREATE OR REPLACE VIEW author_books AS
SELECT wb.isbn, ub.title, a.fname, a.lname
FROM writtenby wb
INNER JOIN uniquebooks ub ON wb.isbn = ub.isbn
INNER JOIN authors a ON wb.aid = a.aid;
```

Output:

	isbn integer	title text	fname text	lname text
1	12525	How I Got a Nuke on MW2	Jordan	Garafolo
2	17713	The Count of Monte Cristo	Alexandre	Dumas
3	29527	The Lord of the Rings	J.R.R	Tolkien
4	11000	The Hobbit	J.R.R	Tolkien
5	23608	House to House	David	Bellavia
6	23608	House to House	John	Bruning
7	98821	American Sniper	Chris	Kyle
8	98821	American Sniper	Scott	McEwen
9	98821	American Sniper	Jim	DeFelice
10	16017	The Alchemist	Paulo	Ceolho
11	99999	Leaves of Grass	Walt	Whitman

Stored Procedures

Get_books_by_member

Purpose:

Allows members and librarians to search what specific member has what books taken out.

Function:

```
CREATE OR REPLACE FUNCTION get_books_by_member(int, REFCURSOR)
RETURNS REFCURSOR AS
$$
DECLARE
  memberID  int      := $1;
  resultset REFCURSOR := $2;
BEGIN
  OPEN resultset FOR
  SELECT *
  FROM books_members_have
  WHERE memberID = mid;
RETURN resultset;
END;
$$
language plpgsql;
```

Output:

```
SELECT get_books_by_member(1, 'results');
FETCH ALL FROM results;
```

	bid integer	title text	dateofcheckout date	datedue date	fname text	lname text	mid integer
1	11	The Hobbit	2016-04-01	2016-04-15	Jal	Gandhi	1
2	8	The Alchemist	2016-04-05	2016-04-19	Jal	Gandhi	1

Get_books_by_genre

Purpose:

This function allows members and librarians to easily search the library based on a specific genre.

Function:

```
CREATE OR REPLACE FUNCTION get_books_by_genre(TEXT, REFCURSOR)
RETURNS REFCURSOR AS
$$
DECLARE
  genreQ    TEXT       := $1;
  genreQ1   TEXT       := $1;
  results   REFCURSOR   := $2;
BEGIN
  OPEN results FOR
    SELECT isbn, title
    FROM uniqueBooks
    WHERE genre LIKE genreQ;

  RETURN results;
END;
$$ language plpgsql;
```

Output:

```
SELECT get_books_by_genre('Autobiography', 'results');
FETCH ALL FROM results;
```

	isbn integer	title text
1	23608	House to House
2	12525	How I Got a Nuke on MW2
3	98821	American Sniper

Get_books_by_author_name

Purpose:

This function's purpose is simply to allow members to search for books written by a specific author.

Function:

```
CREATE OR REPLACE FUNCTION get_books_by_author_name(TEXT, TEXT,
REFCURSOR)
RETURNS REFCURSOR AS
$$
DECLARE
  authorFname   TEXT       := $1;
  authorLname   TEXT       := $2;
  results       REFCURSOR   := $3;
BEGIN
  OPEN results FOR
    SELECT isbn, title
    FROM author_books
    WHERE fname LIKE authorFname
    AND lname LIKE authorLname;

  RETURN results;
END;
$$ language plpgsql;
```

Output:

```
SELECT get_books_by_author_name('J.R.R', 'Tolkien', 'results');
FETCH ALL FROM results;
```

	isbn integer	title text
1	29527	The Lord of the Rings
2	11000	The Hobbit

Get_aid

Purpose:

This function's only purpose is to be used in conjunction with the 'add_book_info' function. This function is useful because it can be used to check whether an author is already in the database or not. Also, since we won't know the aid of the given author passed into 'add_book_info', this function makes it so all we need is the name of the author to retrieve the aid.

Function:

```
CREATE OR REPLACE FUNCTION get_aid(TEXT, TEXT)
RETURNS INTEGER AS
$$
DECLARE
  authorFname   TEXT   := $1;
  authorLname   TEXT   := $2;
BEGIN

RETURN (SELECT aid
      FROM authors
      WHERE fname LIKE authorFname
      AND lname LIKE authorLName);

END;
$$ language plpgsql;
```

Output:

```
SELECT get_aid('Chris', 'Kyle');
```

	get_aid integer
1	6

add_book_info

Purpose:

Due to the nature of the database and the properties of books, the relationship between books and authors is spread across several tables. It is unreasonable and time consuming for a database to expect data to be inputted manually into each table. The purpose of this function is to take in all attributes of a book at once and distribute the data into all relevant tables.

Function:

```
CREATE OR REPLACE FUNCTION add_book_info(INT, TEXT, TEXT, TEXT, TEXT) RETURNS void AS
$$
DECLARE
  bookisbn      INT   := $1;
  booktitle     TEXT  := $2;
  bookgenre     TEXT  := $3;
  authorFname   TEXT  := $4;
  authorLname   TEXT  := $5;
BEGIN
  IF get_aid(authorFname, authorLname) IS NULL THEN
    INSERT INTO authors(fname, lname)
      VALUES(authorFname, authorLname);
  END IF;
  IF (SELECT 1 FROM uniquebooks WHERE isbn = bookisbn LIMIT 1) IS NULL
  THEN
    INSERT INTO uniquebooks(isbn, title, genre)
      VALUES(bookisbn, booktitle, bookgenre);
  END IF;

  IF (SELECT 1
     FROM writtenby
     WHERE isbn = bookisbn
     AND aid = get_aid(authorFname, authorLname)
     LIMIT 1) IS NULL
     THEN

     INSERT INTO writtenby(aid, isbn)
       VALUES(get_aid(authorFname, authorLname), bookisbn);
   END IF;
   INSERT INTO allbooks(isbn)
      VALUES(bookisbn);

END;
$$ language plpgsql;
```

Output:

For the sake of brevity, I will not give outputs.

Sign_out_book

Purpose:

One of the main functions of a library is to allow members to sign out books to borrow for a short period of time. This function takes in the id of the member who wishes to take out a book and the id of the book the member wants to borrow. The transaction will fail if the user has too many books out, has an overdue book, the book is already being borrowed/doesn't exist, or if the member id does not exist. It then stores this information in the booksout table while giving it a 14 borrow period.

Function:

```
CREATE OR REPLACE FUNCTION sign_out_book(INT, INT)
RETURNS void AS
$$
DECLARE
  memberID      INT   := $1;
  bookID        INT   := $2;
BEGIN
IF (SELECT 1 FROM members WHERE mid = memberID) IS NOT NULL THEN
  IF (SELECT 1 FROM books_past_due WHERE mid = memberID) IS NULL THEN
    IF (SELECT COUNT(*)
       FROM booksout
       WHERE mid = memberID) < 3 THEN
      IF(SELECT 1
        FROM booksout
        WHERE bid = bookID) IS NULL THEN
          IF(SELECT 1
            FROM allbooks
            WHERE bid = bookID) IS NOT NULL THEN

              INSERT INTO booksout(mid, bid, dateofcheckout, dateDue)
                VALUES(memberID, bookID, CAST(CURRENT_TIMESTAMP AS DATE),
                              (CAST(CURRENT_TIMESTAMP AS DATE) + INTERVAL '14' day));
          ELSE
            RAISE EXCEPTION 'The book you requested does not exist in this library';
          END IF;
       ELSE
         RAISE EXCEPTION 'The book requested for checkout is not available';
       END IF;

      ELSE
      RAISE EXCEPTION 'This member has the limit of 3 books out and may not borrow another.';
    END IF;
  ELSE
    RAISE EXCEPTION 'This member may not sign out a book until their overdue book is returned.';
```

```
  END IF;
 ELSE
   RAISE EXCEPTION 'No such member with that Member ID exists.';
END IF;
END;
$$ language plpgsql;
```

return_book

Purpose:

Just as it is important to allow books to be borrowed, there must be a function that updates the database when the book is returned. In this case it is only necessary for the book id to be verified so it is the only variable this function passes in.

Function:

```
CREATE OR REPLACE FUNCTION return_book(INT)
RETURNS void AS
$$
DECLARE
  bookID   INT   := $1;
BEGIN
  IF(SELECT 1
    FROM booksout
    WHERE bid = bookID) IS NOT NULL THEN

      DELETE FROM booksout
      WHERE bid = bookID;

  ELSE
    RAISE EXCEPTION 'This book was not listed as borrowed.';
  END IF;
END;
$$ language plpgsql;
```

Implementation Notes

It should be noted that many of the stored procedures and views are meant to be seen only by librarians with some small exceptions where users should be allowed access to a function. Any stored procedure that allows data to be altered in the database should be reserved for librarian or database admin. Functions that aid in the searching of the database for certain books should be accessible to members of the library.

Known Problems

It is know that this database has several undesirable issues. First being that all views and result sets accessible to librarians and members do not disguise the column names of the database. Also, some aspects of the database where it would be beneficial to be customizeable are not currently changeable unless a database administrator edits code within the database. These features include book checkout period and number of books that a user may take out. On top of this, the database relies much on the fact that the librarians make sure data is being input into the database correctly as some values don't have proper constraints. As a result, the database may subject to constant maintenance. Also, much of the security of the database relies on librarians and database administrators not divulging how the database works or critical functions of the database.

Future Enhancements

Adding these features would enhance database's functionality:

- A table that keeps track of every book that has been returned late in the past 'X' number of months and who returned it late.
- A table that allows members to put in a request for a book that is not currently available or exists in the library.
- Add attributes to books such as books that can only be viewed and not borrowed.
- A Procedure that automatically contacts a member if one of the books they have is overdue or if a book they requested is available.
- A feature that would allow members to request rare books/articles/documents in the possession of another library and 'loan' them out for a period of time allow this library to do same and loan to other libraries.