CS4416

Library Database System

Jordan Marshall	18256716
Marcin Sek	18254187
Ademide Adenuga	18220258
Rioghan Lowry	18269699

Logged Work

Jordan Marshall - Report, SQL Schema, SQL Views, ERD

Marcin Sek - Report, SQL Data, SQL Triggers, SQL Views

Ademide Adenuga - Report, SQL Triggers, SQL Schema

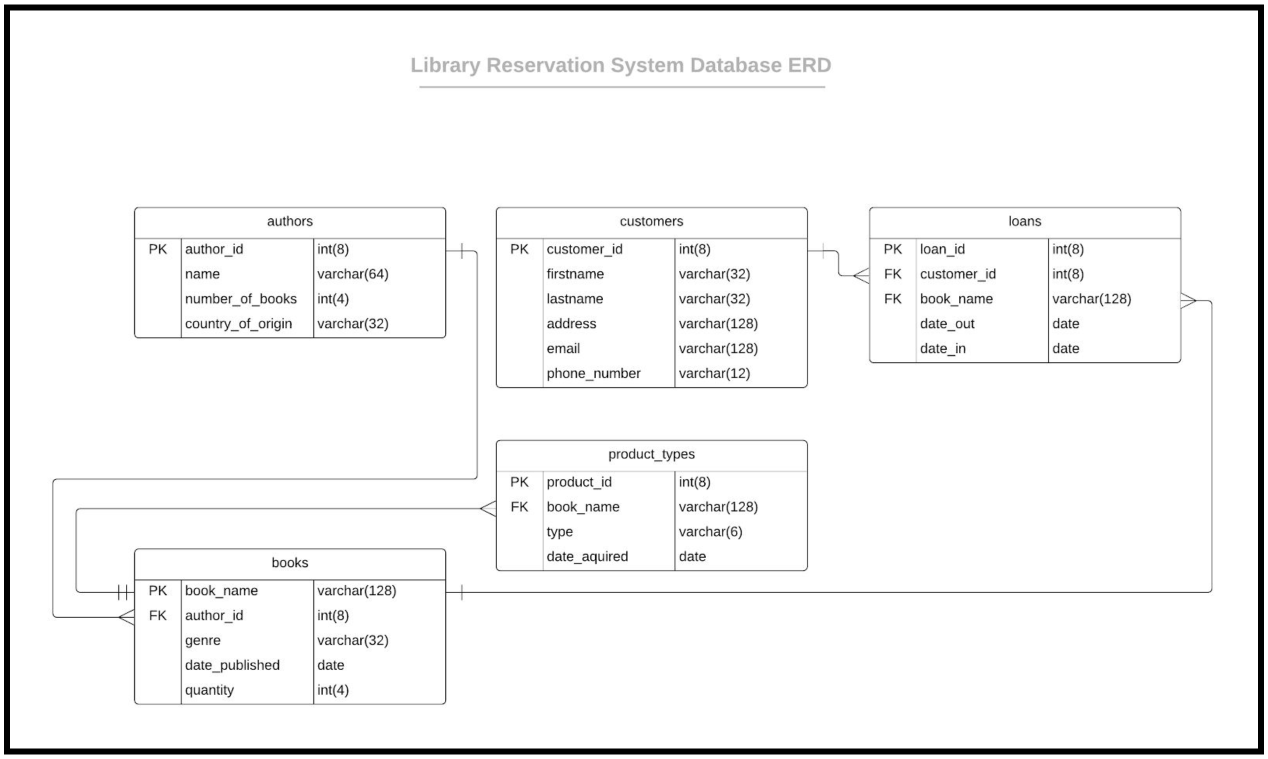
Rioghan Lowry - Report, SQL Schema, ERD

About Our Database

Our database was created for the intention of being used by a library system. This involves the operations of customers, books in the library, their authors, loans, and the products available. It is mainly based on the loan system that libraries have been known to have. When a customer visits the library, they can choose from a wide selection of books of their liking and borrow it from the library. Should they take the book for longer than the proposed return date is, the customer will be required to return the book along with a fee. The fee is 5 euro for each day over the due date and is limited to 100 euro. The database allows for expandability, allowing the owner to add books to the system as well as new product types. Ex: eBooks, CD’s, DVD’s, Internet or Tapes. The five tables included in the database are Books, Authors, Products, Customers and Loans.

A library software system can use this database to log records of all product transactions. The system should also register a customer with the system and log it into the database. Loans can be recorded including the exact date the product was loaned out from the database. In loans a due date can be proposed by the system, if no date is proposed the database automatically sets it to 1 week after the book has been loaned out. The customer has until the due date to return the product. If the customer wishes to loan it out for a longer duration then the product has to be returned and loaned out again.

Entity Relationship Diagram



authors

PK	author_id	int(8)
	name	varchar(64)
	number_of_books	int(4)
	country_of_origin	varchar(32)

customers

PK	customer_id	int(8)
	firstname	varchar(32)
	lastname	varchar(32)
	address	varchar(128)
	email	varchar(128)
	phone_number	varchar(12)

loans

PK	loan_id	int(8)
FK	customer_id	int(8)
FK	book_name	varchar(128)
	date_out	date
	date_in	date

product_types

PK	product_id	int(8)
FK	book_name	varchar(128)
	type	varchar(6)
	date_aquired	date

books

PK	book_name	varchar(128)
FK	author_id	int(8)
	genre	varchar(32)
	date_published	date
	quantity	int(4)

In this ERD, the table authors contains author_id, name, number_of_books and country_of_origin. author_id is a primary key and is also present in the books table as a foreign key. The books table also contains book_name, genre, date_published and quantity. book_name is the primary key. The customers table contains customer_id as its primary key, firstname, lastname, address, email and phone_numbe. Finally the loans table contains loan_id, customer_id, date_out and date_in. Loan_id is the primary key while customer_id and book_name are foreign key.

Tables in Our Database

Authors

Authors is a table that refers to the writers of the books in the library and the attributes that relate to them. Authors consists of an author_id, name, number_of_books and country_of_origin. author_id is the primary key of the table as it is a unique type of data that individually identifies each author of the books in the library. author_id is an 8 digit ID of an int type. Authors can have multiple books but only one author_id. Name refers to the name of the author themselves. This name can be any number of characters up to 64. number_of_books is the amount of copies of the book in question in the library. The max number of the same book can be up to 9999, this is due to the type being int with a limit of 4 digits. A library won't have more than 9999 copies of the same book. This reduces the amount of redundant memory needed on the system. country_of_origin refers to the country that the book in question was published from. This is given an allowance of up to 32 characters.

Products Types

The products_types table contains product_id, book_name, type, date acquired. The primary key is product_id. Each product is identified by its product_id. The foreign key is book_name. Different versions of the same products are stated in the name of the product. The date acquired is a date type and is the date the library acquired the book and was entered it into the database. The type of the product can currently be books. The type was set to type varchar with a limit of 6 characters. Though no other type of product is currently available, this table was created with the intention of ease of updates should there be another product type that the librarian wishes to add.

Books

Books are a product type. Books are made up of a book_name, author_id, genre, date_published and quantity. book_name is the name of the book and has a type varchar with a limit of 128 characters. book_name is the primary key of table books. The author_id is a unique ID given to an author. author_id is the foreign key. There can only be one author, but an author can have many books. The author is a unique 8 digit type int ID. The genre is the genre of the book. The genre can be up to 32 characters, as it is of a varchar type. The date_published attribute is a date type and consists of the date the book was published. The quantity of a product is specified as the quantity attribute and increments when a new product shares the same name.

Customers

The customer table consists of a customer_id, firstname, lastname, address, email and phone_number. Each customer is given an 8 digit ID so they can be identified. customer_id is the primary key for the table. customer_id is an int type with an allowance of 8 integers. Customers can't have the same customer_id. We selected varchar for firstname, lastname, address and email because the input can differ in character amount. Eg: John has a total of 4 characters while Smith has a total of 5 characters. The phone number was given an int type with an allowance of 12 integers. The reason we chose to allow only 12 integers is because a phone number with an area code is 12 integers. Eg: +353-123456789

Loans

Loans is a table that has the primary function of listing all products that have been loaned out to customers. Loans takes the product and puts it into the loaned table. The loan table consists of five attributes: loan_id, customer_id, book_name, date_out and date_due.

loan_id is the primary key. customer_id and book_name are both foreign keys. The loan_id is a unique 8 digit ID given to a customer when a loans out product(s). A customer can have many loan IDs but a loan_id cannot have multiple customer_ids. The customer_id is taken from the customer table and is linked to the products loaned out. The customer_id is an 8 digit unique ID of type int. The book name attribute is the name of the loaned book. The date_out attribute is when a product is taken out / loaned out of the system. Eg: A customer loans out a book for a certain duration. The date_due is when the book has to be returned by the customer. If the product is returned to the system then it is available to be loaned out again. Both date_out and date_due are date types.

Any ID in our database has a possible combination of 10^8 = 100,000,000 different users

EG: 18212356

Functional Dependencies

In terms of functional dependencies, Loans reference the customer loaning out a book as well as the book being loaned meaning you cannot delete either the customer or book while the book is being loaned by said customer, Product_types reference the book meaning you cannot delete the book before you remove the product_type, and books references the author while the book is in the database the Author cannot be deleted.

Table	FD
Loans	INSERT - Must have valid customer_id and valid book_name
Books	INSERT - Must have a valid author_id, DELETE / UPDATE - Must not be used by a child table (Loans, Product_types)
Product_types	INSERT - Must have valid book_name
Customers	DELETE / UPDATE - Must not be used by a child table (Loans)
Authors	DELETE / UPDATE - Must not be used by a child table (Books)

3NF

Third normal form (3NF) is a design in databases that reduces the duplication of data. Each table in the database has been created to be in 3NF format.

Authors

In the authors table, the attributes all relate to the primary key of the table, that is author_id. Name of the author, number of published books and country of origin all describe an author. Also there is no duplication of data in the name. All 3 of the attributes do not link to each other and are independent.

Customers

All the attributes in this table relate directly to the customer who holds the customer_id which is the primary key of the customers table, This information only exists in the customers table which can be accessed by loans as customer_id is a foreign key there. All of the attributes are unique.

Loans

The attributes in this table describe the loan made, it holds two foreign keys customer_id from Customers and book_name from books. This allows the user to access the customer details and book details. The attributes are independent apart from date_out and date_due in their case date_due must be after date_out and defaults to 1 week after the date_out.

Books

The book table inherits the foreign key author_id from authors. The genre, date_published and quantity solely relate to the primary key. Also they are all independent of each other.

Product_types

The product type table inherits the foriegn key book_name from books. The table contains all products and their respective types. Both type and date_acquired are also independent of each other.

Justification of Triggers

For our database, we have included six triggers. The first trigger is called **deletedBook**, which handles the updating of books in the **books** table. Though it deletes the entry in the table, it is meant to act as though a book has been borrowed from the library because when books have been borrowed, they would no longer be available to be borrowed by anyone else until it has been returned. The next trigger is **acquiredBook** which acts as an update for library once a book has been returned to them. This trigger would be needed to coexist with the deletedBook trigger so that returned books can be accounted for.

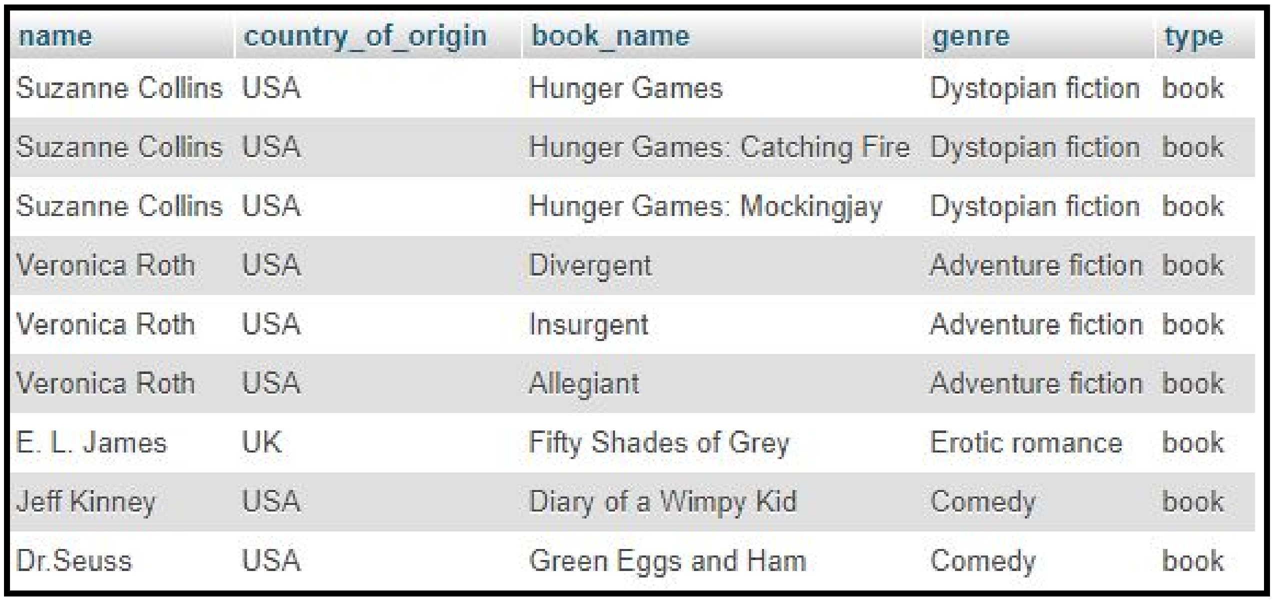
The third trigger is called **loanedBook** and it updates the **loans** table to add a book that has been borrowed by a customer. The trigger that follows it is called **returnedBook** which removes a loaned book from the loans table once it has been returned. This would help librarians keep track of how many books are going in and out of the library.

The fifth trigger is called **setAcquiredDate** and this helps replace the automatic NULL value of the **date_acquired** attribute in the **product_types** table with the date of when a new book has been added. This can be used by librarians for organisational purposes. Our final trigger, **setOutDate**, does a similar thing for the **loans** table but instead updates the value of **date-out** of when the book has been borrowed by a customer. The **due_date** will in turn be updated for an interval of a week. Such a trigger is needed to keep up with loans and their important dates.

Justification of Views

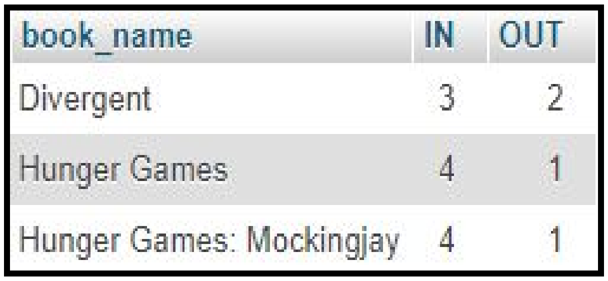
booksdetails

The first view we proposed in the database was for a function that would select all the books that have been borrowed in a certain genre. This can serve to be useful information for librarians as they can view what kind of books that customers are interested in at that period of time. Be it fantasy or horror, the librarians can tend to those who wish to find the books of their genre of interest and make it more available.



name	country_of_origin	book_name	genre	type
Suzanne Collins	USA	Hunger Games	Dystopian fiction	book
Suzanne Collins	USA	Hunger Games: Catching Fire	Dystopian fiction	book
Suzanne Collins	USA	Hunger Games: Mockingjay	Dystopian fiction	book
Veronica Roth	USA	Divergent	Adventure fiction	book
Veronica Roth	USA	Insurgent	Adventure fiction	book
Veronica Roth	USA	Allegiant	Adventure fiction	book
E. L. James	UK	Fifty Shades of Grey	Erotic romance	book
Jeff Kinney	USA	Diary of a Wimpy Kid	Comedy	book
Dr.Seuss	USA	Green Eggs and Ham	Comedy	book

bookavailability

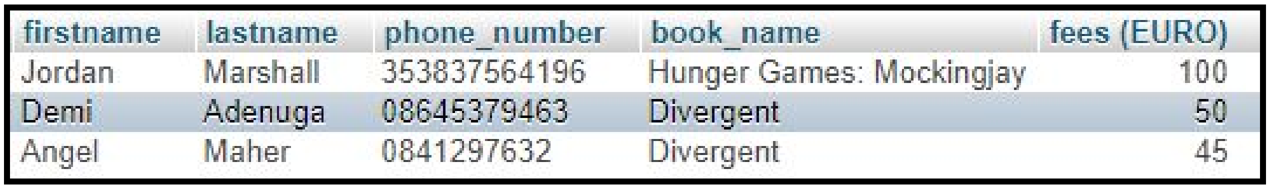


book_name	IN	OUT
Divergent	3	2
Hunger Games	4	1
Hunger Games: Mockingjay	4	1

The second view we proposed was a function that would select a book and provide data for the amount of copies of that book to the person who requested it. We considered this to be an important aspect of the library system, especially for librarians, as it helps with inventory knowledge and helps prevent confusion for missing books.

overduebooks

The third view we proposed was for a function that would display contact details for customers of the library who owe books to the library by date and the overdue books themselves as well as their fees. It would be important for librarians to know what books go in and out of the library but it would be a good idea to also know when these books have been lent out and when they are due so as fees can be clarified for their correct times and receipts can be kept for organisational purposes.



firstname	lastname	phone_number	book_name	fees (EURO)
Jordan	Marshall	353837564196	Hunger Games: Mockingjay	100
Demi	Adenuga	08645379463	Divergent	50
Angel	Maher	0841297632	Divergent	45

Analysis of the Speed - View Queries

```
-- overduebooks        0.0004 seconds

SELECT loans.date_due, customers.firstname, customers.lastname,
customers.phone_number, loans.book_name, IF(5 * DATEDIFF(now(), loans.date_due) >
100, 100, 5 * DATEDIFF(now(), loans.date_due))  AS 'fees (EURO)'
FROM loans
INNER JOIN customers ON customers.customer_id = loans.customer_id
GROUP BY loans.date_due
HAVING loans.date_due < now();
```

This query had our best time of 0.0004 seconds. This contains a SELECT statement, DATEDIFF() function, an IF statement, an INNER JOIN and a GROUP BY and HAVING clause. The DATEDIFF () function used in the query slows down the execution time but also allows for the query to be simplified. The INNER JOIN also uses sufficient memory by joining two tables together. Both tables have to be parsed first which slows down the execution time.

The GROUP BY and HAVING clause parses through the table and selects tuples that satisfy the HAVING clause.

```
--bookavailability       0.0005 seconds

SELECT books.book_name, books.quantity AS 'IN', tb2.OUT
FROM books
INNER JOIN (SELECT book_name, count(*) AS 'OUT' FROM loans GROUP BY(book_name))
tb2
ON books.book_name =  tb2.book_name;
```

This query had a time of 0.0005 seconds. It contains a SELECT statement, INNER JOIN and sub-query containing a SELECT statement and GROUP BY clause which takes 0.0003 seconds of the overall time. The query joins books to tb2 delivered by our sub-query. The GROUP BY clause makes the query take longer to execute but it also allows us to get an accurate count for each book loaned out.

```
-- booksdetails       0.0009 seconds

SELECT authors.name, authors.country_of_origin, tb2.book_name, tb2.genre, tb2.type
FROM authors
INNER JOIN (SELECT books.author_id, books.book_name, books.genre,
product_types.type
  FROM books
  INNER JOIN product_types
  ON product_types.book_name = books.book_name
) tb2
ON tb2.author_id = authors.author_id;
```

This query has our longest time of 0.0009 seconds. It contains a SELECT statement, INNER JOIN and a sub-query containing a SELECT statement and INNER JOIN which takes 0.0006 seconds from the overall time. The sub-query joins books and product_types while the main query joins the table delivered from the sub-query to authors joining them on their author.id’s.

To conclude, we could reduce execution time of the queries used in the views. This can be achieved by reducing the number of joins used and also selecting the better viable options for the joins. We also saw a recurring theme of GROUP BY and HAVING clauses decreasing speed as it needs to parse all tuples and see do they satisfy the condition stated in the HAVING clause. If we would like more specific data to be displayed in the views, we could use the WHERE clause at the cost of performance.

Analysis of the Speed - Justification for Indexes

- author_id
- book_name
- customer_id
- product_id
- loan_id

These are the indexes chosen for the library database system, We have chosen these indexes as they fit perfectly into the context of each table, with book_name, author_id and customer_id being foreign keys in other tables allowing for our SELECT statements to link in with other related tables in substantial time.