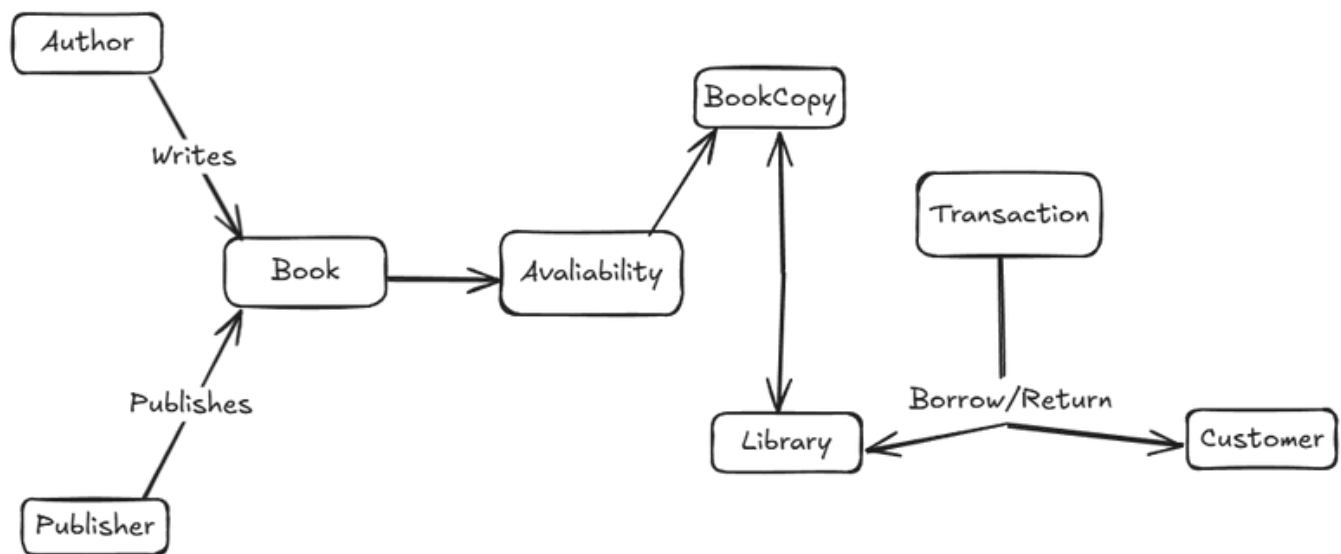


SQL Database Design and Querying for a Library Management System

Assignment

Schema Design

Conceptual Diagram



Explanation

The Following Diagram Shows a Basic Understanding of library Management System,

1. Relationship Author, Publisher and Book

First there is a relation from author and publisher to Book as one author can write many books while a publisher also can publish many books.

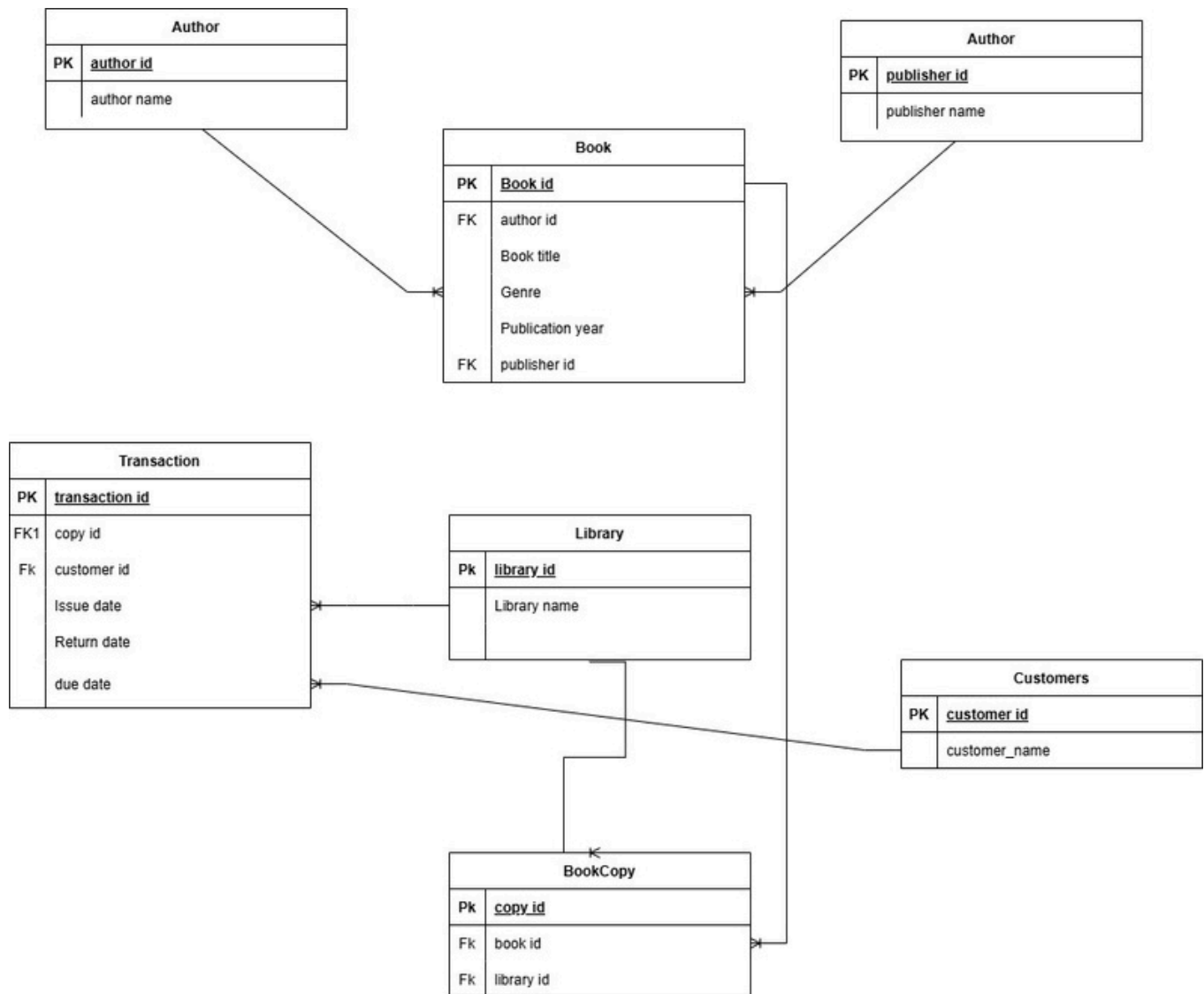
2. Library and Bookcopy

Secondly There's Library which will have the Bookcopy for a specific book, so there can be many Bookcopy's of a single Book where if the customer asks for a specific book to be Borrowed then the Library need to first confirm whether the Bookcopy of the specific book is available then only the transaction can be done.

3. Interaction between Customer, transaction and Library

Thirdly The connection between Library, Customer and Transaction, Here the transaction hold all the records of the customer along with all the dates like due date, return date issue date and the customer details

Entity Relationship Diagram



Explanation

This ER diagram represents the key entities and relationships in a Library Management System.

Main Entities

1. Author Table

- So author table stores author details(author id, author name).
- Primary Key : author id

2. Publisher Table

- This Table manages Publisher information(publisher id, publisher name).
- Primary Key : publisher id

3. Book Table

- Table Contains Book Details like Genre, book title, Publication year and links to Author and Publisher Table.
- Primary Key: Book id

4. Library Table

- Represents library details(library id, library name).
- Primary Key : Library id

5. BookCopy Table

- BookCopy is like a track of copies of books with unique copy id which is then linked to book and library Table.
- Primary Key: Copy id

6. Customer Table

- Stores Customer Details (customer id, customer name).
- Primary Key: customer id

7. Transaction Table

- Holds the record for borrowing books(issue date, return date and due date) and links to BookCopy and customer Table
- Primary Key : transaction id

Relationships

- **Author Publisher and Book** : so one author can write multiple books which which one publisher can publish multiple books which represent one to many relationship.
- **Library and Book Copy** : One library can hold multiple book copies so it represent one to many relationship.
- **Customer, BookCopy and Transaction** : Customers can borrow BookCopies, which can be tracked via Transactions with the help of issue and return dates.

Query 2 - List books along with their authors that belong to the "Fantasy" genre, sorted by publication year in descending order.

```
/*List books along with their authors that belong to the "Fantasy" genre,  
sorted by publication year in descending order.*/
```

```
--Create Index idx_authorid on author(authorid);  
EXPLAIN  
Select b.title as book_title,  
b.genre as genre ,b.publicationyear, a.authurname  
From book b  
join author a  
on b.authorid = a.authorid  
Where b.genre = 'Fantasy'  
Order by b.publicationyear DESC
```

Execution plan

	QUERY PLAN text
1	Sort (cost=12.73..12.74 rows=1 width=1154)
2	Sort Key: b.publicationyear DESC
3	-> Nested Loop (cost=0.00..12.72 rows=1 width=1154)
4	Join Filter: (a.authorid = b.authorid)
5	-> Seq Scan on book b (cost=0.00..11.50 rows=1 width=...
6	Filter: ((genre)::text = 'Fantasy'::text)
7	-> Seq Scan on author a (cost=0.00..1.10 rows=10 wid=...

Explanation

As per my understanding from the above execution plan

- Line 1 and 2 is for like sorting purpose where it's sorting publication year in descending order while the cost here is i think like the storage, rows = 1 indicates the number of rows its effecting.
- Line 3 and 4 is Nested Loop so here we have used the inner join for book and author table based on the attribute match of author id.
- line 5 and 6 is like a Scan of the book table where it tries to find a match as genre = 'fantasy'
- line 7 is also a scan for the author table generally i think this was due to author id as it was joined with book table

After adding a index for the genre in the query

```
--Create Index idx_authorid on author(authorid);
--Create Index idx_genre on book(genre);
EXPLAIN
Select b.title as book_title,
b.genre as genre ,b.publicationyear, a.authername
From book b
join author a
on b.authorid = a.authorid
Where b.genre = 'Fantasy'
Order by b.publicationyear DESC
```

Execution plan

	QUERY PLAN text
1	Sort (cost=2.42..2.42 rows=1 width=1154)
2	Sort Key: b.publicationyear DESC
3	-> Hash Join (cost=1.26..2.41 rows=1 width=1154)
4	Hash Cond: (a.authorid = b.authorid)
5	-> Seq Scan on author a (cost=0.00..1.10 rows=10 ...)
6	-> Hash (cost=1.25..1.25 rows=1 width=642)
7	-> Seq Scan on book b (cost=0.00..1.25 rows=1 ...)
8	Filter: ((genre)::text = 'Fantasy'::text)


Explanation

- Line 1 and 2 is same as above but here the cost is decreased
- Line 3 and 4 here instead of nested loop a Hash join is used to join the book and author table here the cost is
- line 5 is like a Scan of the author table (though i don't know why its not using the index)
- line 6 is hash like it make temporary table for book table
- line 7 and 8 is also a scan for the book table but the cost is lowered as compared to above execution plan and then it like filter based on genre = 'fantasy'

Query 4- List all customers who have overdue books (assume overdue if ReturnDate is null and IssueDate is older than 30 days).

```
--List all customers who have overdue books
--(assume overdue if ReturnDate is null and IssueDate is older than 30 days).
Explain
Select Distinct c.customername
From customer c
Join transaction t
on c.customerid = t.customerid
Where t.returndate Is NULL and t.issuedate < Current_date - Interval '30 days'
```

Execution plan

	QUERY PLAN	
	text	
1	HashAggregate (cost=14.55..14.97 rows=42 width=516)	
2	Group Key: c.customername	
3	-> Hash Join (cost=11.57..14.45 rows=42 width=516)	
4	Hash Cond: (t.customerid = c.customerid)	
5	-> Seq Scan on transaction t (cost=0.00..2.75 rows=42 width=4)	
6	Filter: ((returndate IS NULL) AND (issuedate < (CURRENT_DATE - '30 days'::interval)))	
7	-> Hash (cost=10.70..10.70 rows=70 width=520)	
8	-> Seq Scan on customer c (cost=0.00..10.70 rows=70 width=520)	

Explanation

- Firstly Hash Aggregate is like used to group rows and then perform aggregation functions like sum, count , Distinct etc while it cost like 14.55 and taking effect on 42 rows.
- Line 2 Here the rows are grouped for customername column to remove duplicates
- Line 3 and 4 in this step hash join is used to join customer and transaction table based on customerid
- line 5 and 6 is like a Scan of the transaction table to filter the rows where return date is null and issue date older than 30 days.
- line 7 is hash like it make temporary table for customer table for joining purpose
- line 8 is also a scan for the customer table to read all rows.