Project Part 5: Physical Database Design

Tej Gumaste, Jay Patel, Shayaan Mohammed, Kaleb Howard,
Andrew Reyes, James Hanselmann

Introduction

Project Overview:

The Library Database System is a comprehensive platform designed to manage library resources and support user services efficiently. It facilitates inventory tracking, technology lending, overdue fee management, and report generation for all library items including books, academic articles, and technological equipment. With multi-user support for students, faculty, administrators, and residents, the system also emphasizes secure transactions, data integrity, and user accountability.
This part of the project translates the previously designed logical model into a fully functional physical database, implemented with actual SQL scripts and real or representative data. This implementation supports the core functionalities identified in the earlier phases, ensuring robust performance and reliability.

Scope:

The database covers internal operations including inventory management, technology checkout, overdue tracking, and user-role-based access. It does not cover inter-library loan systems, multimedia databases, or features for real-time collaboration or content viewing.

Glossary (Updated):

- **DDL**: Data Definition Language, used to define schema (e.g., `CREATE TABLE`).
- **DML**: Data Manipulation Language, used to insert/update/delete data.(e.g., `INSERT`, `UPDATE`, `DELETE`).
- **Primary Key**: Uniquely identifies each record in a table.
- **Foreign Key**: Ensures referential integrity by linking tables.
- **Bulk Loading**: Method of inserting large volumes of data efficiently.
- **SQL Report**: A query designed to extract meaningful summarized data.
- **Relational Integrity:** Ensures accuracy and consistency of data across relationships.

Choose Your Platform

For our physical implementation, we chose to use PostgreSQL as our database platform, and connected to it using DBeaver, a universal database management client.

We selected PostgreSQL because it is a powerful, open-source relational database management system (RDBMS) that fully supports the SQL standards required for our project. The decision was influenced by the following factors:

- Robust Feature Set: PostgreSQL offers strong support for relational integrity, indexing, constraints (including foreign keys and ENUM types), and transaction control.
- SQL Standards Compliance: PostgreSQL adheres closely to ANSI SQL, which made it straightforward to translate our ER model into physical schema using SQL DDL and DML scripts.
- Stability and Performance: PostgreSQL is widely used in production environments and known for reliability and consistent performance, especially for relational workloads like ours.
- Free and Open-Source: PostgreSQL's open-source nature allowed us to set up and test our system without licensing restrictions.

We used DBeaver as the database management interface to interact with our PostgreSQL database. DBeaver simplified database development through:

- Visual SQL Editing: The built-in SQL editor with syntax highlighting and autocompletion made writing and testing queries much easier.
- Data Visualization: Tables, schemas, and query results could be viewed and edited through DBeaver's spreadsheet-like interface, reducing the risk of syntax errors and improving efficiency.
- Ease of Connection: DBeaver supports direct connections to PostgreSQL via a simple configuration screen, allowing us to quickly test, view, and manage our physical schema.

Although we were initially introduced to MariaDB on the EECS cycle servers, we chose PostgreSQL for its stronger support of modern SQL features, better error messaging, and the flexibility of using a local installation. DBeaver also provided a smoother and more responsive development environment than the EECS command-line interface.

Create Your Database

Print Your Physical Schema

```
  CREATE TABLE public.libraryuser (
OnlineID INT PRIMARY KEY,
   StudentFacultyID VARCHAR(20),
   Name VARCHAR(100),
   Email VARCHAR(100),
   UserType VARCHAR(20),
   Payment DECIMAL(10,2),
   BorrowingHistory TEXT
```

```
);

CREATE TABLE Books (
    ISBN VARCHAR(13) PRIMARY KEY,
    Title VARCHAR(255),
    Author VARCHAR(255),
    YearReleased INT,
    Type VARCHAR(20),
    Storage VARCHAR(20),
    Status VARCHAR(20),
    ReturnDate DATE
);

CREATE TABLE Articles (
    Title VARCHAR(255),
    Author VARCHAR(255),
    Status VARCHAR(20),
    FieldOfStudy VARCHAR(100),
    JournalName VARCHAR(255),
    YearPublished INT,
    DOI VARCHAR(50),
    ReturnDate DATE,
    Storage VARCHAR(20),
    PRIMARY KEY (Title, Author)
);

CREATE TABLE Technology (
    SerialNumber VARCHAR(50) PRIMARY KEY,
    Type VARCHAR(20),
    Status VARCHAR(20),
    ReturnDate DATE,
    LicenseDuration DATERANGE
);

CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    UserID INT,
    ItemID VARCHAR(50),
    ItemType VARCHAR(20),
```

```
  TransactionType VARCHAR(20),
  DueDate DATE,
  Timestamp DATE,
  FOREIGN KEY (UserID) REFERENCES libraryuser(OnlineID)
);

-- COPY TABLE
CREATE TABLE Copy (
  CopyID INT PRIMARY KEY,
  ISBN VARCHAR(13),
  Status VARCHAR(20),
  ReturnDate DATE,
  FOREIGN KEY (ISBN) REFERENCES Books(ISBN)
);
```

Data Population

- We manually populated test data for each table using INSERT INTO statements in insert_data.sql.
- We used realistic data including actual ISBNs, article DOIs, and tech serials. The data was generated by AI.
- Datasets were validated for uniqueness and correct relationships.

Example Data Insertion (Books):

```sql
-- Insert into technology
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1000', 'Tablet', 'available', '2025-05-06', '[2024-07-24,2024-11-26]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1001', 'Laptop', 'available', NULL, '[2024-11-14,2025-10-12]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1002', 'Laptop', 'available', NULL, '[2024-11-17,2024-11-24]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1003', 'Tablet', 'available', '2025-04-22', '[2024-05-27,2024-07-24]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1004', 'Camera', 'borrowed', '2025-05-02', '[2024-11-11,2024-12-08]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1005', 'Tablet', 'borrowed', NULL, '[2024-11-26,2026-01-29]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1006', 'Camera', 'available', '2025-04-21', '[2024-06-10,2025-05-09]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1007', 'Projector', 'borrowed', NULL, '[2024-04-30,2024-12-25]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1008', 'Laptop', 'available', NULL, '[2024-09-02,2024-10-02]');
INSERT INTO Technology (SerialNumber, Type, Status, ReturnDate, LicenseDuration) VALUES ('SN-1009', 'Laptop', 'available', '2025-04-25', '[2025-03-29,2025-08-14]');
```

Printing Table Contents

```
SELECT * FROM User;
```

onlineid	studentfacultyid	name	email	usertype	payment	borrowinghistory
1	S20231001	Allison Hill	jillrhodes@miller.com	student	1.25	Borrowed: Focused motivating portal, Returned: 2023-05-05
2	S20231002	Michelle Miles	shaneramirez@gmail.com	faculty	12.24	Borrowed: Virtual cohesive standardization, Returned: 2024-03-08
3	S20231003	Christopher Bernard	daviscolin@yahoo.com	student	36.82	Borrowed: Robust empowering workforce, Returned: 2023-05-18
4	S20231004	Kimberly Dudley	arnoldmaria@hotmail.com	student	29.52	Borrowed: Multi-lateral object-oriented core, Returned: 2024-10-13
5	S20231005	Matthew Mejia	michellejames@reid-diaz.com	student	1.49	Borrowed: Multi-channeled demand-driven secured line, Returned: 2023-11-08
6	S20231006	Melinda Jones	frankgray@watts.com	student	11.63	Borrowed: Focused 5thgeneration workforce, Returned: 2023-12-16
7	S20231007	Austin Gentry	dianafoster@hotmail.com	student	28.06	Borrowed: Persevering client-server firmware, Returned: 2023-08-01
8	S20231008	Justin Baker	icox@hotmail.com	faculty	11.02	Borrowed: Proactive background system engine, Returned: 2023-05-21
9	S20231009	Brittany Moore	cruzcaitlin@yahoo.com	faculty	40.47	Borrowed: Progressive eco-centric neural-net, Returned: 2025-02-11
10	S20231010	Melanie Wilson	daniel62@yahoo.com	student	37.94	Borrowed: Adaptive uniform success, Returned: 2023-11-05

(10 rows)

```
SELECT * FROM Books;
```

isbn	title	author	yearreleased	type	storage	status	returndate
2654235116155	Seven medical blood personal	Katie Gonzalez	2018	Novel	Shelf A4	available	2025-05-03
6184959310341	Much single morning	Fred Smith	1994	Textbook	Shelf D3	borrowed	2025-05-02
5255341928327	Dream drive note	Nicholas Cabrera	2004	Novel	Shelf E2	borrowed	2025-05-10
0305641395376	Campaign little near	Shane Henderson	2021	Novel	Shelf B3	available	
8849696532871	Up sense ready	Jennifer Powers	1983	Textbook	Shelf B2	borrowed	2025-05-03
6697848018451	Tonight later easy ask	Julian Chapman	1998	Novel	Shelf B4	available	2025-04-15
4828148932528	Remain arrive	Andre Rivera	2013	Textbook	Shelf E3	borrowed	
5430391171822	Draw protect Democrat car very	Kaitlyn Mueller	2022	Novel	Shelf E2	borrowed	2025-05-04
9638346578713	Behind probably great	Scott Brown	1994	Textbook	Shelf C1	available	2025-04-19
0103105183473	I fund	Danielle Lee	2014	Textbook	Shelf E5	borrowed	

(10 rows)

```
SELECT * FROM Articles;
```

```
LibrarySystem=# SELECT * from Articles;
```

title	author	status	fieldofstudy	journalname	yearpublished	doi	returndate	storage
Cell contain leg	Angela Morton	borrowed	Statistician	Williams PLC	2013	10.4119/think.97		Shelf I3
Better	Sarah Martin	borrowed	Radiographer, diagnostic	Lewis-Morales	2013	10.8651/artist.885	2025-04-22	Shelf F1
Able hospital unit	Dominique Horton	borrowed	Surveyor, commercial/residential	Higgins, Moore and Phillips	2023	10.6559/article.820	2025-05-13	Shelf G2
Article finish anyone	Thomas Schmidt	available	Barista	Miller, Lopez and Larson	2017	10.8350/Republican.144		Shelf H4
Identify walk	Lauren Williams	available	Ceramics designer	Sandoval-Cunningham	2002	10.8260/four.828	2025-04-27	Shelf J1
Prove fire	Carlos Brewer	available	Programmer, systems	Hickman-Walls	2020	10.9856/act.857		Shelf F2
Perform none beyond	Katie Anderson	available	Magazine journalist	Suarez LLC	2013	10.8956/feel.493		Shelf I1
Just myself	William Roman	available	Seismic interpreter	Contreras PLC	2012	10.1035/firm.400		Shelf I3
Voice care	Brian Tran	borrowed	Glass blower/designer	Jacobs-Robbins	2022	10.8973/Democrat.159		Shelf G1
Measure economy	Amber Walters	available	Health and safety inspector	Smith and Sons	2023	10.6138/public.59		Shelf I5

(10 rows)

```
SELECT * FROM Technology;
```

```
LibrarySystem=# SELECT * from Technology;
```

serialnumber	type	status	returndate	licenseduration
SN-1000	Tablet	available	2025-05-06	[2024-07-24,2024-11-27)
SN-1001	Laptop	available		[2024-11-14,2025-10-13)
SN-1002	Laptop	available		[2024-11-17,2024-11-25)
SN-1003	Tablet	available	2025-04-22	[2024-05-27,2024-07-25)
SN-1004	Camera	borrowed	2025-05-02	[2024-11-11,2024-12-09)
SN-1005	Tablet	borrowed		[2024-11-26,2026-01-30)
SN-1006	Camera	available	2025-04-21	[2024-06-10,2025-05-10)
SN-1007	Projector	borrowed		[2024-04-30,2024-12-26)
SN-1008	Laptop	available		[2024-09-02,2024-10-03)
SN-1009	Laptop	available	2025-04-25	[2025-03-29,2025-08-15)

(10 rows)

```
SELECT * FROM Transactions;
```

transactionid	userid	itemid	itemtype	transactiontype	duedate	timestamp
6001	3	Identify walk	Article	borrow	2025-04-26	2025-03-15
6002	6	8849696532871	Book	return	2025-04-22	2025-02-18
6003	5	6184959310341	Book	return	2025-04-23	2025-03-22
6004	5	2654235116155	Book	return	2025-05-05	2024-04-23
6005	3	SN-1004	Technology	borrow	2025-04-23	2025-01-25
6006	8	SN-1006	Technology	borrow	2025-05-01	2025-02-11
6007	2	5255341928327	Book	borrow	2025-05-04	2025-02-19
6008	6	SN-1008	Technology	borrow	2025-04-27	2024-07-26
6009	7	2654235116155	Book	return	2025-04-22	2025-02-13
6010	6	6697848018451	Book	borrow	2025-04-20	2024-09-09
6011	4	SN-1001	Technology	return	2025-04-16	2025-03-13
6012	9	Measure economy	Article	borrow	2025-05-07	2024-07-09
6013	4	5255341928327	Book	return	2025-04-24	2025-02-25
6014	1	6697848018451	Book	return	2025-05-07	2024-08-27
6015	4	Able hospital unit	Article	borrow	2025-04-29	2025-02-26
6016	7	5430391171822	Book	borrow	2025-04-26	2024-07-09
6017	4	Prove fire	Article	return	2025-04-16	2024-12-26
6018	4	2654235116155	Book	borrow	2025-04-24	2024-09-15
6019	7	Identify walk	Article	borrow	2025-04-30	2024-07-31
6020	5	Voice care	Article	return	2025-04-18	2025-02-08
6021	9	Cell contain leg	Article	borrow	2025-04-28	2024-08-28
6022	5	0103105183473	Book	return	2025-05-09	2024-09-16
6023	1	0103105183473	Book	return	2025-05-11	2024-08-20
6024	6	SN-1005	Technology	return	2025-04-16	2024-05-28
6025	10	SN-1001	Technology	return	2025-04-27	2024-06-22
6026	10	8849696532871	Book	borrow	2025-05-13	2024-07-26
6027	7	9638346578713	Book	borrow	2025-04-16	2024-05-26

;

```
SELECT * FROM Copy;
```

```
LibrarySystem=# SELECT * from Copy;
 copyid |      isbn       |  status   | returndate
--------+-----------------+-----------+------------
   2000 | 2654235116155 | borrowed  |
   2001 | 6184959310341 | borrowed  | 2025-04-27
   2002 | 5255341928327 | available |
   2003 | 0305641395376 | available |
   2004 | 8849696532871 | borrowed  | 2025-04-19
   2005 | 6697848018451 | available | 2025-05-02
   2006 | 4828148932528 | borrowed  | 2025-05-12
   2007 | 5430391171822 | available | 2025-05-04
   2008 | 9638346578713 | available |
   2009 | 0103105183473 | available | 2025-04-22
(10 rows)
```

Functionality Testing

We developed a suite of SQL queries and reports to verify functionality and validated the system's behavior with queries simulating real operations:

Example Queries:

- **Get all overdue books:**

```
SELECT b.ISBN, b.Title, b.Author, c.CopyID, c.ReturnDate,
    u.Name AS BorrowerName, u.Email
FROM Books b
JOIN Copy c ON b.ISBN = c.ISBN
JOIN Transactions t ON c.CopyID = t.ItemID AND t.ItemType = 'Book'
JOIN libraryuser u ON t.UserID = u.OnlineID
WHERE c.Status = 'Borrowed'
 AND c.ReturnDate < CURRENT_DATE;
```

- **Search articles by field:**

```
SELECT a.Title, a.Author, a.JournalName, a.YearPublished, a.DOI,
    a.Status, a.ReturnDate
FROM Articles a
WHERE a.FieldOfStudy LIKE :searchField
ORDER BY a.YearPublished DESC, a.Title ASC;
```

Edge Case Testing:

- Insertions violating foreign key constraints were correctly rejected.
- Return date logic for overdue items worked correctly with various dates.

Conclusion

Our Library Database System is now fully implemented at the physical level, supporting the major functionalities that we have highlighted throughout the previous project parts. We have ensured schema and data integrity and query accuracy through testing.