Workshop N° 2 - Data System Architecture and Information Retrieval

Universidad Distrital Francisco José de Caldas
School of Engineering
Computer Engineering Program

David Stiven Muñoz Amaya - 20202020071

Janeth Oliveros Ramírez - 20182020100
Daniel Felipe Paez Mosquera - 20202020070

September 2025

1 Introduction

This project proposes the development of an Academic Digital Library Platform designed to complement, rather than replace, the university's physical library. The platform provides exclusive access for students and faculty, enabling DRM-free consultation and downloads of institutional content such as theses, research papers, and authorized digital books. Licensed materials from external providers (e.g., IEEE, Elsevier, Springer) are not hosted locally, but rather indexed and redirected through secure links, clearly delimiting the project's implementation scope to institutional resources.

The first release of the system focuses on essential functionalities (user authentication, role-based access control, advanced search, and metadata-driven downloads) ensuring a feasible and scalable implementation within the academic context. Future extensions, such as real-time analytics or AI-based recommendations, are considered out of scope for the current phase but remain part of the long-term vision.

Users have distinct roles: students search, consult, download, and review materials; professors access resources and suggest acquisitions; administrators manage catalogs, users, and reports; and optional digital librarians validate content and oversee licenses. Core features emphasize security, performance, and scalability, guaranteeing reliable 24/7 access that fosters research and learning excellence.

2 Canvas Business Model

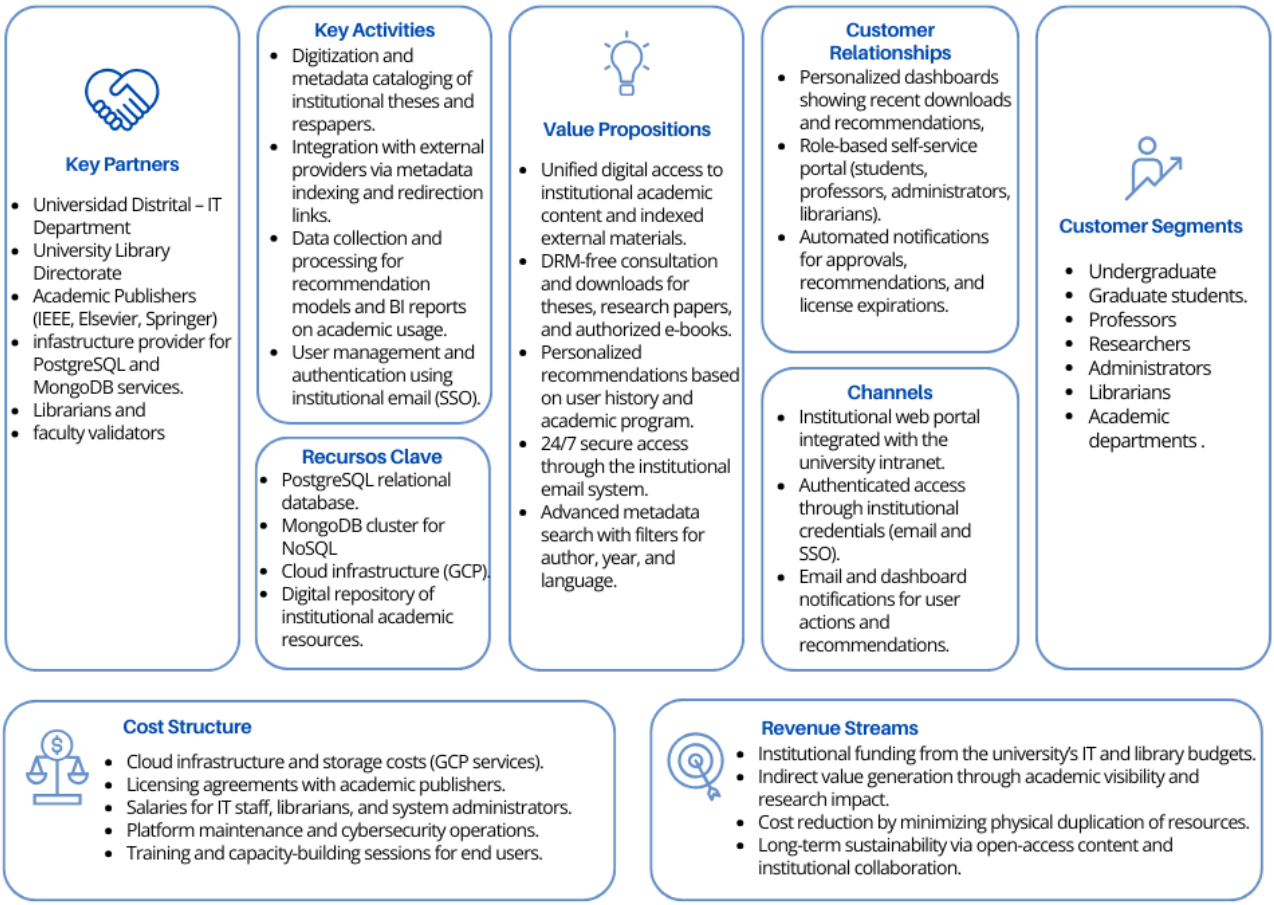

Figure 1: Canvas Model

3 Requirements Documentation

3.1 Functional Requirements

Scope reminder. Digital University Library focused on students and professors. Resources are books and articles only. Downloads are DRM-free. Hybrid access model: (i) local university content is stored and downloadable; (ii) licensed/commercial content is indexed and redirects to the provider.

1. **User Registration and Authentication.** Users (Student, Professor, Admin, Librarian) must authenticate with institutional email (SSO or local login).
2. **Role-Based Access Control (RBAC).** System enforces role permissions for search, upload requests, approvals, and administration.
3. **Resource Ingestion (Local).** Admins and Librarians can create a resource with mandatory metadata and upload files (PDF/ePub).

4. **Upload Request (Professor Draft).** Professors can submit a draft resource for review (metadata + file or external link).

5. **Curation/Approval Workflow.** Admin/Librarian can approve, reject, or request changes with reason.

6. **External (Licensed) Resource Registration.** System supports records that store metadata and an `external_url` (no local file).

7. **Metadata Management.** CRUD for metadata (authors, affiliations, keywords, ISBN/DOI, abstract, language, license).

8. **Advanced Search & Browse.** Search by title, author, keywords; filters by year, type (book/article), language.

9. **Resource Details & Download.** Resource page shows full metadata, availability (local/external), and download or access button.

10. **Recommendations (Near Real-Time).** Show "You may also like" based on recent views/downloads and shared keywords.

11. **Reviews & Ratings.** Students/Professors can rate (1–5) and review resources; one review per user-resource.

12. **Citation Export.** Export APA/MLA and BibTeX for books and articles.

13. **Notifications.** Email/onsite notifications for workflow events (draft submitted, approved/rejected).

14. **BI Reports (Batch).** Nightly jobs compute: most consulted by program/semester/year, top authors, trending topics.

15. **Concurrency Control for Licensed Limits (Optional).** If a licensed resource has max concurrent accesses, system enforces limit.

3.2 Non-Functional Requirements

1. **Performance.** Searches should return results in less than 1 second for indexed queries. Opening a resource page should be fast (under 2 seconds).

2. **Scalability.** The system must support growth in the number of users and resources each semester (e.g., from hundreds to a few thousands).

3. **Availability.** The system should be available during the semester with minimal downtime. 24/7 availability is expected, but maintenance windows are acceptable.

4. **Security.** Access must be restricted by user role (student, professor, admin). Only authorized users (with institutional email) can log in and download resources.

5. **Data Management.** Store resources in a relational database. Optionally complement with a NoSQL or search engine for fast queries.

6. **Privacy.** Only minimal personal data should be stored (name, institutional email, role). Protect intellectual property of resources (local vs. external access).

4 Data System Architecture

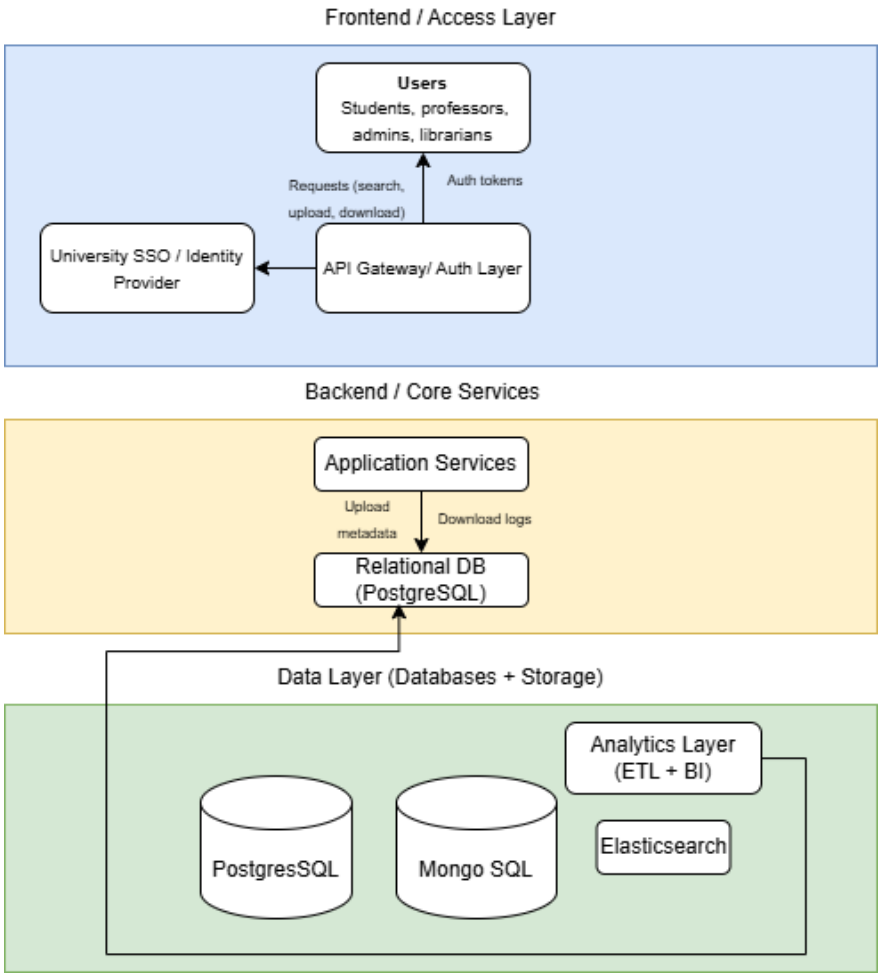

Figure 2: Initial Proposal for the Database System Architecture."

The initial proposal for the database system architecture of the Academic Digital Library Platform defines a hybrid, multi-layered data design that integrates relational, NoSQL, and search-index technologies to ensure scalability, consistency, and high-performance access to institutional resources.

At its core, the relational database (PostgreSQL) serves as the authoritative system of record, storing structured information such as user profiles, resource metadata, access rights, and license data. Complementing this, a NoSQL database (MongoDB) maintains denormalized documents that support fast read operations, personalized recommendations, and flexible data representation for research materials.

For advanced retrieval, an Elasticsearch index enables full-text search and filtering by keywords, author, or program, ensuring sub-second response times. Large binary files such as theses, research papers, and e-books are stored separately in Object Storage (e.g., Google Cloud Storage or MinIO) to optimize performance and reduce database load.

The data architecture also incorporates a streaming and ETL layer to process usage logs, download events, and metadata updates in near real-time. These transformed datasets feed a Data Warehouse used by the BI layer, enabling analytical dashboards that provide insights into content usage, research trends, and library performance.

This architecture proposal emphasizes data integrity, modularity, and scalability, ensuring that both transactional and analytical operations can coexist efficiently while maintaining the security and availability

required by the university's academic environment.

5 Information Requirements

The **University Digital Library** manages several types of information that support both academic and administrative objectives. Each information category is directly linked to a specific User Story and an element of the Business Model Canvas, ensuring full alignment between functional and strategic goals.

5.1 1. Information Types and Their Relationships

ID	Information Type	Related User Story	Related Business Model Canvas Element
IR1	Book and article search queries (by title, author, keywords)	US1 – Search for books and articles	Value Proposition: quick and efficient access to academic materials.
IR2	Digital resource metadata (title, author, file type, access type, upload date)	US1, US4, US9	Key Resources: management of digital content and academic assets.
IR3	Download logs (user, resource, timestamp, program)	US2, US4	Customer Relationship: monitoring of user activity and engagement.
IR4	User reviews and ratings (text, score, author, date)	US3	Customer Relationship: user interaction and feedback collection.
IR5	Acquisition requests from professors (title, author, ISBN/DOI, justification, status)	US5	Key Partners: collaboration between faculty and library for content curation.
IR6	Validation and approval data of uploaded resources (pending, approved, rejected)	US6	Key Activities: ensuring academic and copyright compliance.
IR7	External resource licenses (provider, expiration date, renewal status)	US7	Key Resources: maintenance of legal access to licensed materials.
IR8	User accounts and roles (student, professor, librarian, admin)	US8	Customer Segments: user classification and controlled access.
IR9	Catalog management records (resources added, updated, removed)	US9	Key Activities: continuous updating of the digital collection.

Table 1: Information Types and Their Relation to User Stories and Business Model Canvas

5.2 2. Functional Scope and Query Examples

Functional Area	Information Requirement	Example Query Type

User Behavior	Downloads by program and most active users	SQL – Aggregated downloads by academic program.
Resource Popularity	Most downloaded digital resources	SQL – Top 5 most downloaded books or articles.
Authors and Research	Authors with the largest number of resources	SQL – Most prolific authors.
Search	Find resources by title or author	NoSQL – Full-text search using regex or text indexes.
Suggestions Workflow	Status of acquisition requests	SQL – Grouped by status (pending, approved, rejected).
Recent Activity	Latest downloads or submissions	NoSQL – Recent activity logs (real-time collection).
Licenses and Providers	Track licenses per external provider	SQL – Resources per external provider.

Table 2: Functional Scope and Example Queries for the Digital Library

5.3 3. Information Flow Overview

The platform handles two major categories of information:

- **Operational Information:** Includes daily user interactions such as searches, downloads, and reviews. This data enables the library to measure engagement and usage trends.
- **Administrative Information:** Includes content validation, acquisition requests, and license management. This information supports decision-making, resource management, and ensures academic integrity.

All records are stored in a relational database for structured queries and consistency. Some real-time logs and analytical data (e.g., recent activity) are managed through NoSQL collections to ensure scalability and performance.

5.4 4. Reflection: Challenges and Design Decisions

Defining the information requirements required balancing functionality, feasibility, and data privacy. Initially, advanced analytics such as personalized recommendations and predictive BI reports were considered but later excluded to maintain the project's academic focus and technical simplicity.

Another significant decision was separating operational and analytical data. This design ensures that daily operations such as downloads and searches remain fast and reliable, while reports and summaries are processed in scheduled batches. Maintaining metadata consistency and compliance with digital rights was also identified as a key factor for ensuring the credibility and legal sustainability of the digital library.

6 User Stories

US1 - Search for resources

Priority: High **Estimate:** 5 points

User Story:
As a student, I want to search for books and articles by title, author, or keywords so that I can quickly find the material I need.

Acceptance Criteria:

- Given I am on the search page, When I enter a title, author, or keyword and click search, Then the results show matching resources with metadata and availability within 1 second.

US2 - Download resources

Priority: High **Estimate:** 3 points

User Story:
As a student, I want to download books and articles in PDF/ePub format so that I can study them offline.

Acceptance Criteria:

- Given I have selected a resource, When I click the download button, Then the file downloads in PDF/ePub and the action is logged with user, resource, and timestamp.

US3 - Leave reviews and ratings

Priority: Medium **Estimate:** 3 points

User Story:
As a student, I want to leave reviews and ratings for the resources I use so that I can share my opinion with other students.

Acceptance Criteria:

- Given I have accessed a resource, When I submit a review (max 500 characters) and a rating (1-5 stars), Then the review is displayed with reviewer name and date.

US4 - Professors access resources

Priority: High **Estimate:** 5 points

User Story:
As a professor, I want to access the digital books and articles available in the library so that I can use them in my classes and research.

Acceptance Criteria:

- Given I am logged in as a professor, When I search for and download resources, Then the system grants access only to authenticated users.

US5 - Suggest acquisition

Priority: Medium **Estimate:** 5 points

User Story:
As a professor, I want to suggest the acquisition of a book or article not available in the catalog so that the library can evaluate including it.

Acceptance Criteria:

- Given I searched and found no matching resource, When I click "Suggest acquisition" and fill the form (title, author, ISBN/DOI, justification), Then the suggestion is saved with status = Pending.

US6 - Validate and approve uploads

Priority: High **Estimate:** 8 points

User Story:
As a librarian, I want to validate and approve uploaded digital resources so that only authorized materials are published.

Acceptance Criteria:

- Given a resource is uploaded, When I review it as librarian, Then I can approve or reject it with a reason, and approved resources become visible.

US7 - Manage external resource licenses

Priority: Medium **Estimate:** 5 points

User Story:
As a librarian, I want to manage licenses of external digital resources so that users always have access to subscribed journals.

Acceptance Criteria:

- Given a resource has a license, When the expiration date is within 30 days, Then the system notifies the librarian and flags expired resources.

US8 - Manage user accounts and roles

Priority: High **Estimate:** 8 points

User Story:
As an administrator, I want to manage user accounts and assign roles (student, professor, librarian) so that access is secure and controlled.

Acceptance Criteria:

- Given I am an admin, When I create, update, or deactivate a user, Then only predefined roles can be assigned and all actions are logged.

US9 - Manage digital catalog

Priority: High **Estimate:** 5 points

User Story:
As an administrator, I want to manage the digital catalog so that the content available to users is always updated.

Acceptance Criteria:

- Given I am an admin, When I add, update, or remove a resource, Then the changes are reflected immediately in the catalog.

7 Entity–Relationship (ER) Model

The Entity–Relationship (ER) model of the *Digital University Library* system was reviewed and refined based on the instructor's feedback and the guidelines from *Workshop 2*. The main goal of this version is to enhance the diagram's readability, clarify conceptual hierarchies, and standardize table and attribute naming conventions according to PostgreSQL database design best practices.

7.1 General Description

The model is organized into five functional domains:

- **Users and Roles:** includes the entities `user_account`, `role`, and `program`, which represent the system's actors and their academic affiliation.
- **Digital Content:** composed of `digital_resource`, `author`, `digital_resource_author`, `external_provider`, and `resource_license`, responsible for managing bibliographic information, authorship, and licensing.
- **Suggestion Workflow:** modeled through the `resource_suggestion` table, where professors propose materials that are later reviewed by librarians or administrators.
- **Audit and Logging:** the `download_log` table stores user downloads and actions, allowing subsequent statistical or BI analysis.
- **Support and Referential Integrity:** indexes, foreign keys, and uniqueness constraints ensure relational integrity and data consistency across all entities.

7.2 Main Improvements Compared to Workshop 1

1. Entity and key names were standardized to maintain consistency and avoid reserved words: `User` → `user_account`, `Resource` → `digital_resource`, `Download` → `download_log`.
2. All `varchar(n)` types were replaced with `text` or `char(2)` as appropriate, removing arbitrary limits and improving text search flexibility.
3. Every foreign key was typed as `bigint` (instead of `bigserial`), ensuring referential integrity and clear relationship definitions.
4. The relationship between `external_provider` and `digital_resource` was reoriented, defining the provider as the parent entity (one provider can supply multiple resources).
5. The attribute `author_order` was added to the bridge table `digital_resource_author` to preserve the order of authorship in academic publications.

6. The direct link between `author` and `digital_resource` was removed and replaced by the intermediate N:M relationship table.

7. The semantics of secondary tables (`resource_license`, `download_log`, `resource_suggestion`) were improved to distinguish structural entities from transactional or event records.

7.3 Graphical Representation

Figure 3 presents the final ER diagram, generated in *dbdiagram.io* from the SQL script exported from *Enterprise Architect 16.1*. The diagram clearly shows the primary and foreign keys as well as the cardinalities that define the logical structure of the database for the Digital University Library.

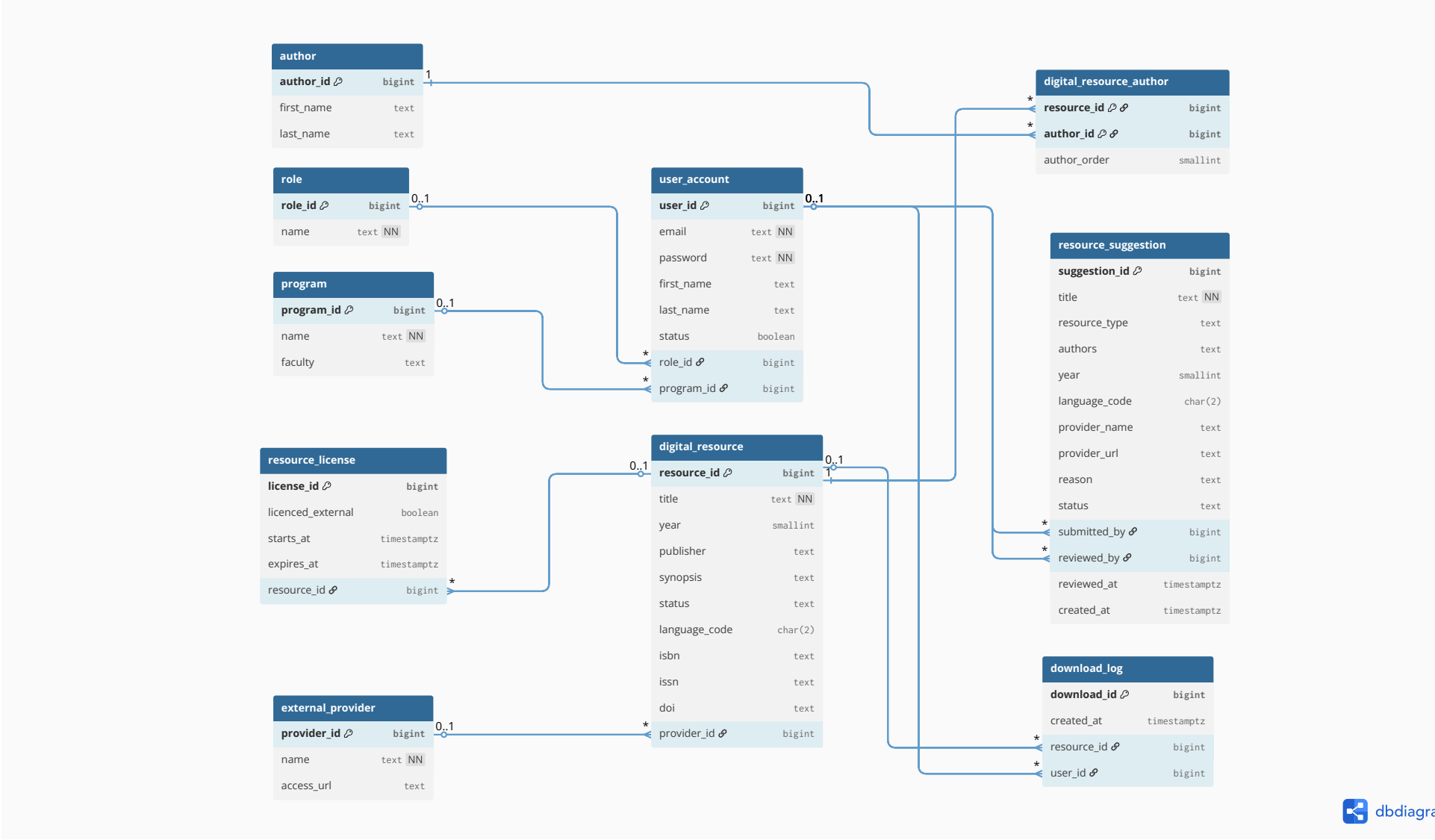

Figure 3: Refined Entity–Relationship (ER) Model of the Digital University Library system.

With this refinement, the ER model now exhibits a coherent, scalable, and well-structured design aligned with both the functional and non-functional requirements of the project. The adopted conventions improve maintainability and facilitate future implementation in PostgreSQL, ensuring integrity, semantic clarity, and traceability among the system's core entities.

8 Database Implementation (SQL)

The relational database schema for the *Digital University Library* system was implemented in PostgreSQL 16 based on the refined Entity–Relationship model. All entity and attribute names were standardized using `snake_case` convention to ensure readability and compatibility with SQL syntax. The implementation follows good database design practices, including primary keys, foreign keys, indexes, and referential integrity constraints.

Schema Overview

The system is organized into five functional domains:

- **Users and Roles:** manages user profiles, authentication roles, and academic programs.
- **Digital Content:** stores metadata for digital resources, authorship, and licensing.
- **Suggestions Workflow:** handles professors' acquisition requests and librarian reviews.
- **Audit and Logging:** records download activity and usage metrics.
- **Support and Integrity:** enforces constraints, data types, and indexing for performance.

DDL Implementation Example

The main script (`ModeloER_BD2_Sierral.sql`) defines the complete database structure. Below is a representative excerpt showing table creation and key constraints:

```
CREATE TABLE user_account (
    id_user         BIGSERIAL PRIMARY KEY,
    email           TEXT NOT NULL,
    password        TEXT NOT NULL,
    first_name      TEXT NOT NULL,
    last_name       TEXT NOT NULL,
    status          BOOLEAN NOT NULL,
    id_role         BIGINT REFERENCES role(id_role),
    id_program      BIGINT REFERENCES program(id_program)
);

CREATE TABLE digital_resource (
    id_resource     BIGSERIAL PRIMARY KEY,
    title           TEXT NOT NULL,
    year            SMALLINT NOT NULL,
    publisher       TEXT NOT NULL,
    synopsis        TEXT NOT NULL,
    status          TEXT NOT NULL,
    language_code   CHAR(2) NOT NULL,
    isbn            TEXT,
    doi             TEXT,
    id_provider     BIGINT REFERENCES external_provider(id_provider)
);
```

Data Population

A small dataset (10–20 records) was inserted to enable validation of the schema and testing of analytical queries. This sample data covers all major entities: `role`, `program`, `user_account`, `author`, `external_provider`, `digital_resource`, and `download_log`. Each record was designed to preserve referential integrity and allow meaningful joins across tables.

Validation and Integrity

The database was successfully deployed and verified in PostgreSQL 16 using `pgAdmin 4`. Referential constraints, data types, and indexes were checked to ensure:

- Each user belongs to a valid academic program and role.
- Each digital resource optionally references an external provider.
- Downloads and suggestions maintain valid foreign key links to users and resources.

Overall, the SQL implementation guarantees consistency, normalization (up to 3NF), and extensibility for future modules such as reviews, ratings, and recommendation tracking.

9 Query Proposals (SQL)

This section presents three representative SQL queries developed for the relational model of the *Digital University Library*. Each query addresses a main information requirement identified during the system analysis phase, focusing on user activity, resource popularity, and authorship statistics. The results were validated using the sample dataset implemented in PostgreSQL 16.

Query 1 – Most Downloaded Resources

Information Requirement: Identify which digital resources (books or articles) have been downloaded most frequently by users. This metric helps the administrator and librarian analyze reading trends and resource popularity across the university community.

```
SELECT dr.title, COUNT(*) AS total_downloads
FROM download_log dl
JOIN digital_resource dr ON dr.id_resource = dl.id_resource
GROUP BY dr.title
ORDER BY total_downloads DESC
LIMIT 5;
```

Explanation: This query aggregates download data by resource title. The use of the `COUNT()` aggregate function combined with a `GROUP BY` clause allows ranking of the most accessed resources. The `LIMIT 5` clause retrieves the top five most downloaded items, useful for generating dashboards or BI reports.

Query 2 – Most Prolific Authors

Information Requirement: Determine which authors have contributed the largest number of digital resources in the repository. This supports academic analysis and reporting of institutional publishing activity.

```
SELECT
    a.first_name_author || ' ' || a.last_name_author AS author_name,
    COUNT(*) AS total_publications
FROM author a
JOIN digital_resource_author ra ON ra.author_id = a.id_author
GROUP BY a.first_name_author, a.last_name_author
ORDER BY total_publications DESC;
```

Explanation: This query joins the `author` and `digital_resource_author` tables to count the total number of publications per author. By using string concatenation for names and grouping by author identity, it provides clear visibility of the most active contributors within the digital catalog.

Query 3 – Downloads by Academic Program

Information Requirement: Measure how many downloads were made by students from each academic program. This helps to understand which faculties or departments are most engaged with the library's digital resources.

```
SELECT
    p.name_program,
    COUNT(*) AS total_downloads
FROM program p
JOIN user_account u ON u.id_program = p.id_program
JOIN download_log dl ON dl.id_user = u.id_user
GROUP BY p.name_program
ORDER BY total_downloads DESC;
```

Explanation: This query performs a three-table join to relate academic programs with user accounts and their download activity. It enables administrators to evaluate library usage by program and make decisions about resource allocation, promotion, or acquisition priorities.

Summary: These three SQL queries collectively demonstrate the relational database's ability to support descriptive analytics, trend discovery, and institutional reporting. They confirm the consistency and usability of the schema implemented in PostgreSQL and provide a solid foundation for extending the system toward BI or NoSQL-based recommendation modules.

10 NoSQL Model

To complement the relational schema, a document-oriented NoSQL model was designed to support advanced search, recommendation, and real-time activity tracking. This model follows the principles of **MongoDB**-style document storage, where information is organized into collections of JSON-like documents instead of normalized relational tables.

Conceptual Structure

The NoSQL model groups data by functionality rather than normalization rules. Each collection represents a key entity of the *Digital University Library* system, embedding related information when appropriate to optimize read performance and simplify queries.

- **users** — stores user profiles and roles, with references to downloaded resources.
- **resources** — stores digital resources with embedded author and provider details.
- **suggestions** — records acquisition suggestions submitted by professors and reviewed by librarians.
- **logs** — registers recent download or access events.

Example Document Structure

The following JSON example illustrates the structure of a resource document in the NoSQL model. It includes embedded subdocuments for authors, provider, and license details.

```
{
  "_id": "res_101",
  "title": "Machine Learning in Education",
  "year": 2023,
  "publisher": "IEEE",
  "language_code": "en",
  "authors": [
    { "name": "John Doe", "order": 1 },
    { "name": "Jane Smith", "order": 2 }
  ],
  "provider": {
    "name": "IEEE Xplore",
    "url": "https://ieeexplore.ieee.org"
  },
  "license": {
    "licenced_external": true,
    "expires_at": "2025-12-31"
  },
  "downloads": 125,
  "topics": ["data science", "education", "AI"]
}
```

Design Rationale

This structure reduces the need for complex joins and allows for high-speed retrieval of frequently accessed data, such as resource metadata and author information. Embedding subdocuments (e.g., authors, provider) increases read efficiency, which is essential for full-text search and personalized recommendations. References are kept only when necessary (e.g., user identifiers in the `logs` collection).

Comparison with the Relational Model

- The relational model ensures **data integrity**, normalization, and referential consistency.
- The NoSQL model offers **flexibility**, faster queries, and scalability for large read workloads.
- Together, they form a **hybrid architecture**, combining the robustness of PostgreSQL for transactional data with the performance of MongoDB for analytical and user-facing operations.

Overall, the NoSQL model complements the relational implementation by enabling near real-time queries and supporting future integration of intelligent recommendation and search systems.

11 Query Proposals (NoSQL)

This section presents three representative queries designed for the document-oriented implementation of the *Digital University Library* using a MongoDB-like query language. Each query addresses a key information requirement related to text search, content recommendation, or user activity tracking. The results illustrate how NoSQL complements the relational database by enabling flexible and scalable operations.

Query 1 – Full-text Search by Title or Author

Information Requirement: Allow users to search for resources that contain a given keyword either in the title or in the author list. This supports the advanced search functionality of the digital library interface.

```
db.resources.find({
  $or: [
```

```
    { title: { $regex: "data science", $options: "i" } },
    { "authors.name": { $regex: "data science", $options: "i" } }
  ]
});
```

Example Result:

```
[
  {
    "title": "Data Science Fundamentals",
    "authors": [{ "name": "John Doe" }],
    "publisher": "Springer"
  }
]
```

Explanation: The query performs a case-insensitive search using the `$regex` operator on both the `title` field and the embedded `authors.name` array. This design supports dynamic and flexible keyword-based search capabilities without the need for predefined indexes or strict schema constraints.

Query 2 – Recommendations by Shared Authors

Information Requirement: Generate recommendations for users by retrieving other resources written by the same author(s) of a given book or article.

```
db.resources.aggregate([
  { $match: { "authors.name": "John Doe" } },
  { $lookup: {
      from: "resources",
      localField: "authors.name",
      foreignField: "authors.name",
      as: "related_resources"
  }},
  { $project: { title: 1, related_resources: { title: 1 } } }
]);
```

Explanation: This aggregation pipeline first selects resources by a specific author and then performs a self-lookup to find other materials by the same author. It demonstrates how MongoDB can implement recommendation logic directly within the database layer using `$match`, `$lookup`, and `$project` stages.

Query 3 – Recent Download Activity

Information Requirement: Display the most recent download actions to provide administrators with real-time insights on user engagement and system usage.

```
db.logs.find().sort({ created_at: -1 }).limit(5);
```

Example Result:

```
[
  { "user_id": "usr_03", "resource_id": "res_204", "created_at": "2025-10-17T14:25:00Z" },
  { "user_id": "usr_02", "resource_id": "res_101", "created_at": "2025-10-17T13:50:00Z" }
]
```

Explanation: The query sorts the `logs` collection by the `created_at` field in descending order to return the five most recent downloads. It enables administrators to monitor system activity and user behavior in near real time.

Summary: These NoSQL queries complement the relational SQL ones by enabling fast, schema-flexible operations. They illustrate how document databases can efficiently handle full-text search, relationship-based recommendations, and streaming activity data. Together, the SQL and NoSQL models provide a hybrid architecture combining transactional consistency with analytical flexibility for the digital library system.

12 Conclusions

The development of the *Digital University Library* database allowed the design and implementation of a hybrid architecture that integrates both relational and NoSQL models. The relational schema implemented in PostgreSQL 16 guarantees data integrity, normalization, and consistent relationships among users, resources, authors, and downloads. It provides a robust foundation for transactional operations, ensuring that every access, download, and suggestion is properly validated and traceable.

On the other hand, the NoSQL model—structured as MongoDB-style collections—adds flexibility and high performance for operations that require full-text search, recommendations, and real-time monitoring. By representing data in semi-structured JSON documents, the system can retrieve and process user-centric information more efficiently, especially when dealing with large datasets or frequent queries.

The SQL and NoSQL query proposals demonstrate the complementary nature of both approaches. While SQL excels in analytical and structured queries (e.g., aggregation by program, author productivity, or download statistics), NoSQL supports dynamic and scalable access patterns such as text search and activity feeds. Together, they achieve a balance between consistency and scalability, essential for a digital library serving a growing academic community.