BS DEGREE IN APPLIED MATHEMATICS & COMPUTER SCIENCE
Academic year: 2024/25 - 3rd year, 2nd term
Subject: File Structures and Databases
First Assignment's Report: Relational DB Design and Population

uc3m | Universidad Carlos III de Madrid

Lecturers:	EIDER ERNESTO ELBITAR LOPEZ		
Group:	R87-228	Lab User	
Student:	Ishaan Kalra	NIA:	100559793
Student:	Ben Purchase	NIA:	100557337
Student:	Lucas Pedemonte	NIA:	100558486

Introduction

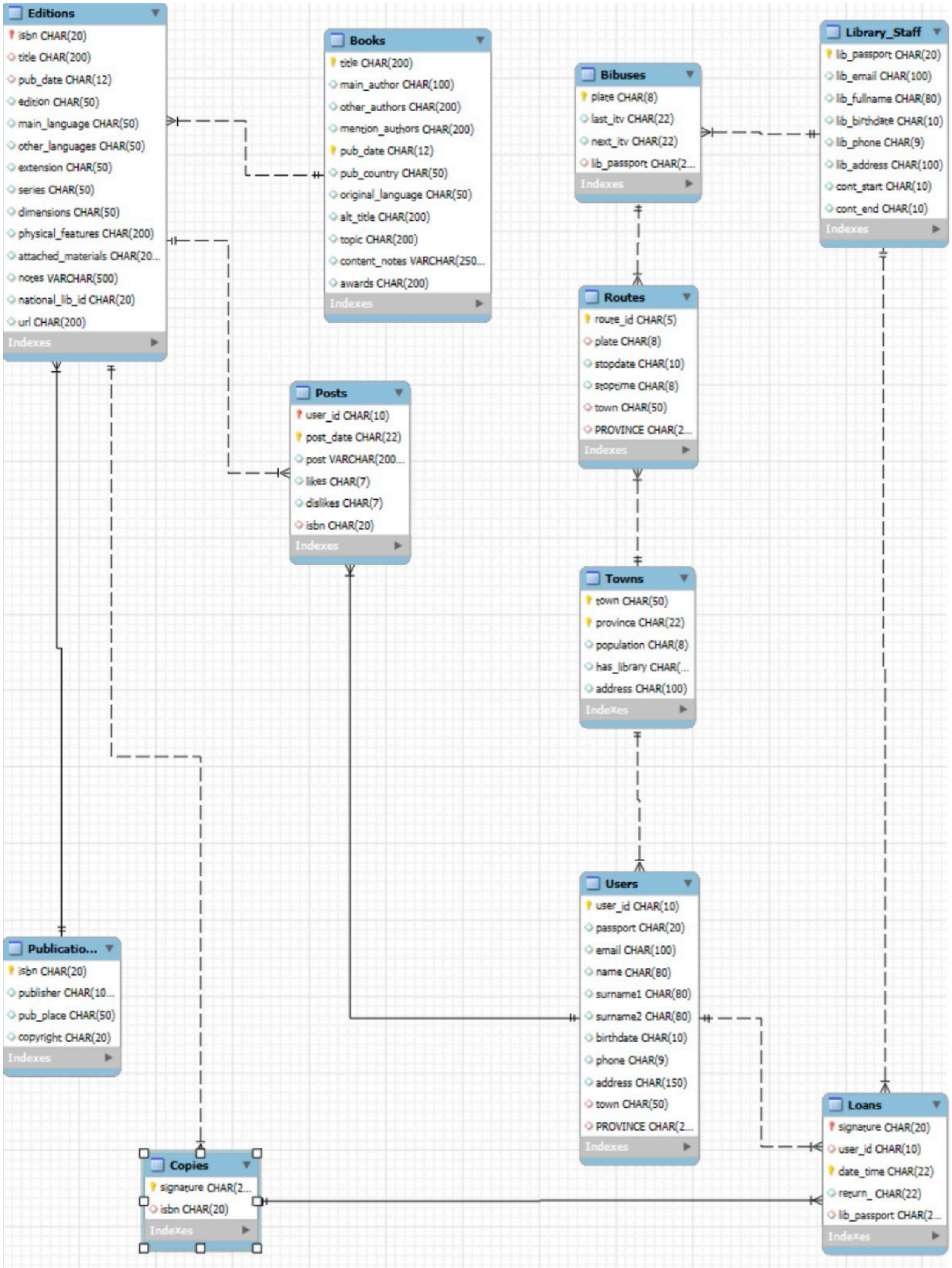
- The Foundation for the Diffusion of Culture (Foundicu Org.) requires an optimized relational database system to efficiently manage its bibliographic collections and mobile library operations. The current database is inadequate, with only three loosely connected tables—acervus (bibliographic collection), loans (book lending records), and busstops (mobile library routes). These tables lack the necessary structure and constraints for seamless data management. The objective of this lab work is to analyze these deficiencies, design a robust relational schema that adheres to the given specifications, and implement it using PL/SQL on an Oracle DBMS.
- This report is accompanied by two SQL scripts:
  - NEWcreation.sql – Defines the database schema.
  - NEWload.sql – Handles data migration from the old system to the new structure.
- The document is structured as follows:
  - Relational Design – Outlines the complete relational schema, implicit semantics, and non-observed explicit semantics using relational graph notation and tabular representations.
  - Relational Statics Implementation in SQL (DDL) – Provides the SQL implementation of the relational schema, detailing re-incorporated, newly incorporated, and excluded semantics.
  - Workload (DML) – Describes the data migration process, including the order of table population, challenges faced, and solutions applied to ensure data

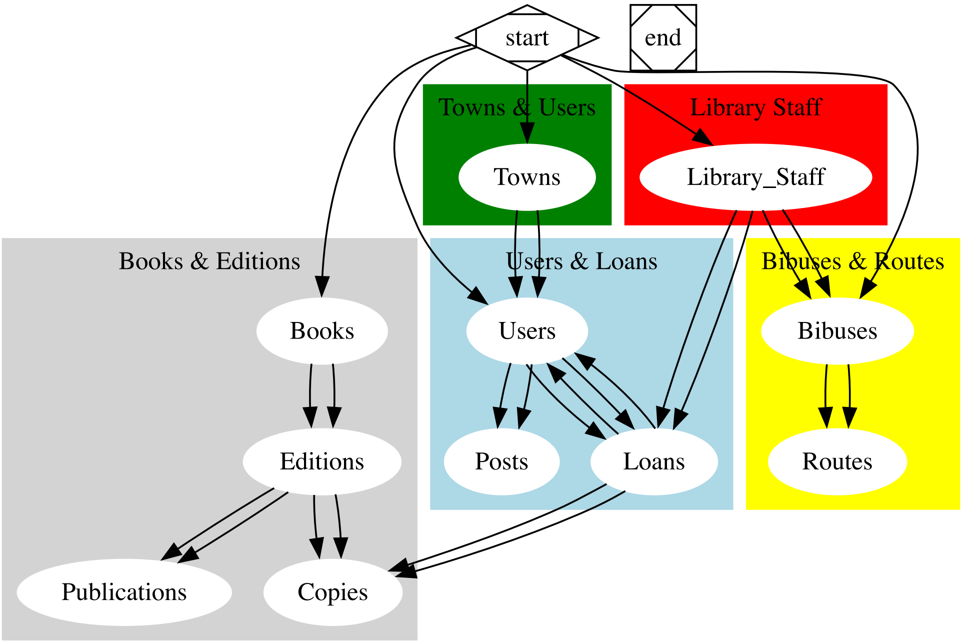
integrity. This structured approach ensures that the newly designed database meets the operational needs of Foundicu Org., improving efficiency, accuracy, and usability.

Relational Design

This section is subdivided into three subsections:

1. Relational Schema


Editions
isbn CHAR(20)
title CHAR(200)
pub_date CHAR(12)
edition CHAR(50)
main_language CHAR(50)
other_languages CHAR(50)
extension CHAR(50)
series CHAR(50)
dimensions CHAR(50)
physical_features CHAR(200)
attached_materials CHAR(20...
notes VARCHAR(500)
national_lib_id CHAR(20)
url CHAR(200)
Indexes
Books
title CHAR(200)
main_author CHAR(100)
other_authors CHAR(200)
mention_authors CHAR(200)
pub_date CHAR(12)
pub_country CHAR(50)
original_language CHAR(50)
alt_title CHAR(200)
topic CHAR(200)
content_notes VARCHAR(250...
awards CHAR(200)
Indexes
Bibuses
plate CHAR(8)
last_itv CHAR(22)
next_itv CHAR(22)
lib_passport CHAR(2...
Indexes
Library_Staff
lib_passport CHAR(20)
lib_email CHAR(100)
lib_fullname CHAR(80)
lib_birthdate CHAR(10)
lib_phone CHAR(9)
lib_address CHAR(100)
cont_start CHAR(10)
cont_end CHAR(10)
Indexes
Routes
route_id CHAR(5)
plate CHAR(8)
stopdate CHAR(10)
stoptime CHAR(8)
town CHAR(50)
PROVINCE CHAR(2...
Indexes
Posts
user_id CHAR(10)
post_date CHAR(22)
post VARCHAR(200...
likes CHAR(7)
dislikes CHAR(7)
isbn CHAR(20)
Indexes
Towns
town CHAR(50)
province CHAR(22)
population CHAR(8)
has_library CHAR(...
address CHAR(100)
Indexes
Publicatio...
isbn CHAR(20)
publisher CHAR(10...
pub_place CHAR(50)
copyright CHAR(20)
Indexes
Users
user_id CHAR(10)
passport CHAR(20)
email CHAR(100)
name CHAR(80)
surname1 CHAR(80)
surname2 CHAR(80)
birthdate CHAR(10)
phone CHAR(9)
address CHAR(150)
town CHAR(50)
PROVINCE CHAR(2...
Indexes
Loans
signature CHAR(20)
user_id CHAR(10)
date_time CHAR(22)
return_ CHAR(22)
lib_passport CHAR(2...
Indexes
Copies
signature CHAR(2...
isbn CHAR(20)
Indexes



2. Implicit Semantics

Semantic presuppositions that are not found in the explicit description but are required to complete the relational design.

Table 1: Implicit semantics incorporated into the relational graph

Presp_id	Stage	Mechanism	Description
I1	Design	Primary key	Each book is uniquely identified by its title.
I2	Design	Primary key	Each edition of a book is uniquely identified by its ISBN.
I3	Design	Primary key	Each user is uniquely identified by their user ID.
I4	Design	Primary key	Each loan transaction is uniquely identified by the signature of the borrowed copy.
I5	Implementation	Foreign key	The `isbn` field in Copies must reference a valid book edition.
I6	Implementation	Foreign key	The `user_id` in Loans must reference an existing user.
I7	Design	Check constraint	A book can have multiple editions, but each edition must have a valid ISBN.
I8	Implementation	Check constraint	A user can only borrow a book if their account is in good standing.
I9	Implementation	Not null constraint	The `route_id` in Routes must always have a valid reference.
I10	Implementation	Default value	`has_library` in Towns defaults to `false` if not specified.

3. Non-Observed Explicit Semantics

Each of the explicit presuppositions (stated in the problem description) that could not be included in the relational graph will be identified (with a label, such as S1, S2, ...) and described in this section.

Table 2: Non-observed explicit semantics

Presp_id	Description
S1	Phone numbers have 9 digits (at least, at most).
S2	Users cannot borrow more than 2 books at a time, except municipal libraries.
S3	Books are borrowed for a fixed period of two weeks.
S4	Municipal libraries can hold 2 copies for every ten registered inhabitants.
S5	Users must return books on the next visit of the mobile library.
S6	Overdue returns result in penalties based on the delay period.
S7	Users can leave comments on books, but only if they have borrowed them before.
S8	Likes and dislikes are stored, but only positive votes are counted.
S9	Deregistered books remain in the database but cannot be loaned out.
S10	Mobile libraries follow predefined routes and do not deviate.

Relational Statics Implementation in SQL (DDL)

This section must include the creation of each table. In addition to the code (`NEWcreation.sql` script) for creating tables (valid syntax in PL/SQL), the following subsections are also covered for an indepth analysis of the Relational Statistics.

- **Excluded semantics that are re-incorporated**
- **Newly incorporated implicit semantics**
- **Explicit semantics that were observed but are now excluded**

Table 3: Re-incorporated explicit semantics

Presp_id	Solution Description
S1	Field size is 9; a constraint (constraint_name) CHECK (phone >= 100000000) is added to the table `Library_Staff`.
S2	A foreign key constraint is added between `Books` and `Editions` on `isbn`, which was excluded in previous versions of the schema but is now re-incorporated.

Presp_id	Solution Description
S3	A primary key constraint is added for the `Books` table using (title, pub_date).
S4	A primary key constraint is added for the `Publications` table using (isbn).
S5	A foreign key constraint between `Editions` and `Publications` is added on (isbn).

Table 4: Implicit semantics incorporated in the definition of each table

Presp_id	Stage	Mechanism	Description
In+1	Implementation	Check	No phone number in `Library_Staff` should be less than 100000000.
In+2	Implementation	Foreign Key	Foreign key relationship is established between `Users` and `Towns` based on (town, province).
In+3	Implementation	Check	`Books` table ensures no duplicate entries using a primary key on (title, pub_date).
In+4	Implementation	Foreign Key	Foreign key relationship is established between `Editions` and `Publications`.
In+5	Implementation	Foreign Key	Foreign key relationship is established between `Books` and `Editions` based on (title, pub_date).

Excluded Semantics

Table 5: Explicit semantics excluded in the creation of each table

Presp_id	Description	Cause	Explicit/Implicit
E1	The foreign key on `Loans` referencing `Copies` should enforce cascading updates, but PL/SQL does not support cascading updates for foreign keys in this case.	PL/SQL does not support automatic cascading updates for foreign keys in this implementation.	Implicit
E2	The table should automatically check for the existence of 'books' before a loan is processed, but this check is excluded for simplicity.	Design decision: Flexibility is needed for the processing of loans.	Explicit
E3	There should be automatic checks on loan duration (e.g., no loan can exceed a certain period), but this is excluded for flexibility.	Design decision: No automatic enforcement of loan durations.	Explicit

Presp_id	Description	Cause	Explicit/Implicit
E4	The system should track the number of books borrowed per user automatically, but this is not implemented here.	Design decision: No enforcement of automatic book count during loans.	Explicit

---

Workload Uploading Process

The workload upload process involves transferring data from the old database structure into the newly designed relational schema. The migration is carried out using the `NEWload.sql` script, which ensures that the data is properly formatted, constraints are enforced, and inconsistencies are handled effectively.

Order of Table Population

To maintain referential integrity and avoid foreign key violations, data is uploaded in the following order:

```
MERGE INTO Books dest
USING (
    SELECT Title, Main_Author, Other_Authors, Mention_Authors,
NVL(PUB_DATE, 'N/A') AS PUB_DATE,
           Pub_Country, Original_Language, Alt_Title, Topic,
Content_Notes, Awards
    FROM (
        SELECT Title, Main_Author, Other_Authors, Mention_Authors,
PUB_DATE, Pub_Country,
               Original_Language, Alt_Title, Topic, Content_Notes,
Awards,
               ROW_NUMBER() OVER (PARTITION BY Title, PUB_DATE
ORDER BY Title) AS rn
        FROM fsdb.acervus
    )
    WHERE rn = 1
) src
ON (dest.title = src.Title AND dest.pub_date = src.PUB_DATE)
WHEN NOT MATCHED THEN
    INSERT (title, main_author, other_authors, mention_authors,
pub_date, pub_country,
            original_language, alt_title, topic, content_notes,
awards)
    VALUES (src.Title, src.Main_Author, src.Other_Authors,
src.Mention_Authors, src.PUB_DATE,
            src.Pub_Country, src.Original_Language, src.Alt_Title,
src.Topic, src.Content_Notes, src.Awards);
```

- This table transfers all the necessary data into the Books table from fsdb.acervus, using the ROW_NUMBERI() query to make sure that only one unique combination of TITLE and PUB_DATE (the primary key) are recorded.

```
MERGE INTO Publications dest
USING (
    SELECT ISBN, Publisher, Pub_Place, Copyright
    FROM (
        SELECT ISBN, Publisher, Pub_Place, Copyright,
               ROW_NUMBER() OVER (PARTITION BY ISBN ORDER BY ISBN)
AS rn
        FROM fsdb.acervus
    )
    WHERE rn = 1
) src
ON (dest.isbn = src.ISBN)
WHEN NOT MATCHED THEN
    INSERT (isbn, publisher, pub_place, copyright)
    VALUES (src.ISBN, src.Publisher, src.Pub_Place, src.Copyright);
```

- This table transfers all the necessary data into the Publications table, also using the ROW_NUMBER query to ensure that each publication has a unique ISBN.

```
MERGE INTO Posts dest
USING (
    SELECT USER_ID, POST_DATE, POST, LIKES, DISLIKES, ISBN
    FROM (
        SELECT l.USER_ID, l.POST_DATE, NVL(l.Post, 'N/A') AS Post,
l.LIKES, l.DISLIKES, e.ISBN,
               ROW_NUMBER() OVER (PARTITION BY l.USER_ID,
l.POST_DATE, NVL(l.Post, 'N/A') ORDER BY l.USER_ID) AS rn
        FROM fsdb.loans l
        JOIN fsdb.acervus a ON l.SIGNATURE = a.SIGNATURE
        JOIN Editions e ON a.ISBN = e.ISBN
        JOIN Users u ON l.USER_ID = u.user_id
    )
    WHERE rn = 1
) src
ON (dest.user_id = src.USER_ID AND dest.post_date = src.POST_DATE
AND dest.post = src.POST)
WHEN NOT MATCHED THEN
    INSERT (user_id, post_date, post, likes, dislikes, isbn)
    VALUES (src.USER_ID, src.POST_DATE, src.POST, src.LIKES,
src.DISLIKES, src.ISBN);
```

- The posts table also has the same features but uses JOINs to receive the data from multiple tables and effectively compile them together.Likewise with the loans table.

```
MERGE INTO Loans dest
USING (
    SELECT
```

```
        l.SIGNATURE,
        u.USER_ID,
        l.DATE_TIME,
        l.RETURN
    FROM (
        SELECT
            SIGNATURE,
            USER_ID,
            DATE_TIME,
            RETURN,
            ROW_NUMBER() OVER (PARTITION BY SIGNATURE ORDER BY SIGNATURE) AS rn
        FROM fsdb.loans
        WHERE SIGNATURE IS NOT NULL
    ) l
    JOIN Users u ON l.USER_ID = u.USER_ID
    WHERE l.rn = 1 -- Keep only one row per unique SIGNATURE
    AND NOT EXISTS (
        SELECT 1
        FROM Loans dest
        WHERE dest.SIGNATURE = l.SIGNATURE
    )
) src
ON (dest.SIGNATURE = src.SIGNATURE)
WHEN NOT MATCHED THEN
    INSERT (SIGNATURE, USER_ID, DATE_TIME, RETURN)
    VALUES (src.SIGNATURE, src.USER_ID, src.DATE_TIME, src.RETURN);
```

Challenges Faced & Solutions

Challenge	Solution
Ensuring Unique Identifiers – The old tables contained duplicate records for books and publications.	Used `ROW_NUMBER()` query to ensure uniqueness for `TITLE` and `PUB_DATE` in `Books` and `ISBN` in `Publications`.
Handling Missing or Null Values – Some entries had missing publication dates or ISBNs.	Applied default values or excluded incomplete records from migration.
Data Normalization – The old schema stored redundant data across multiple tables.	Consolidated and mapped related attributes into normalized tables in the new schema.
Merging Data from Multiple Tables – `Posts` and `Loans` required data from different sources.	Used `JOIN` queries to aggregate and structure the necessary information before insertion.
Foreign Key Constraints – Dependencies between tables led to insertion errors if not handled sequentially.	Followed a structured data insertion order, starting with independent tables before dependent ones.

This structured approach ensures that the newly designed database meets the operational needs of Foundicu Org., improving efficiency, accuracy, and usability.