Advanced Data Bases TC3041 – ITESM Puebla

Professor: Dr. Daniel Pérez Rojas

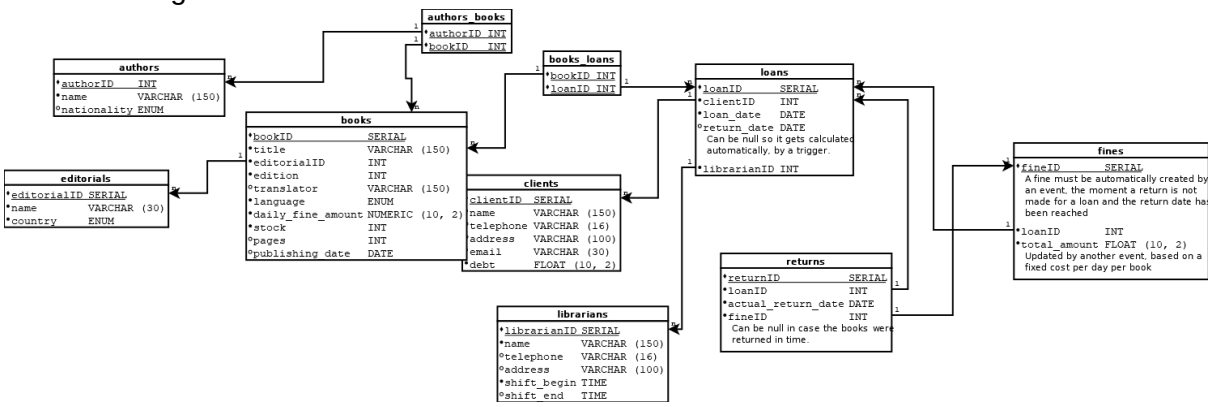
Homework 2.3 – Library DB

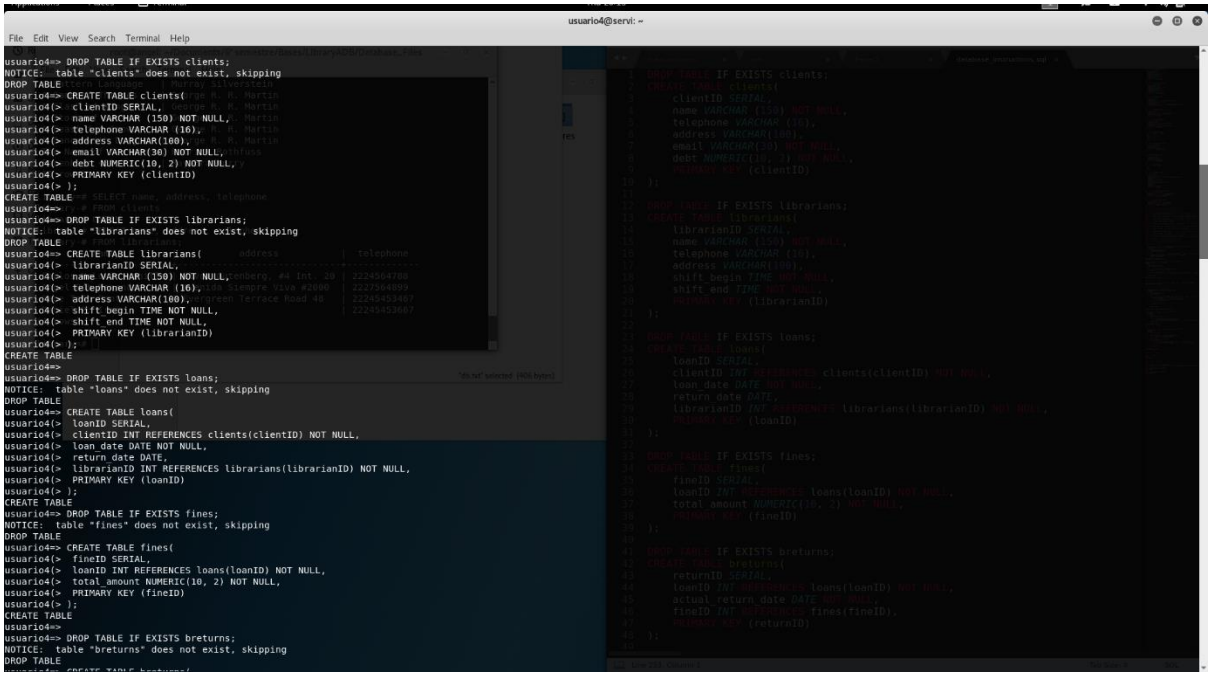
9/03/2018

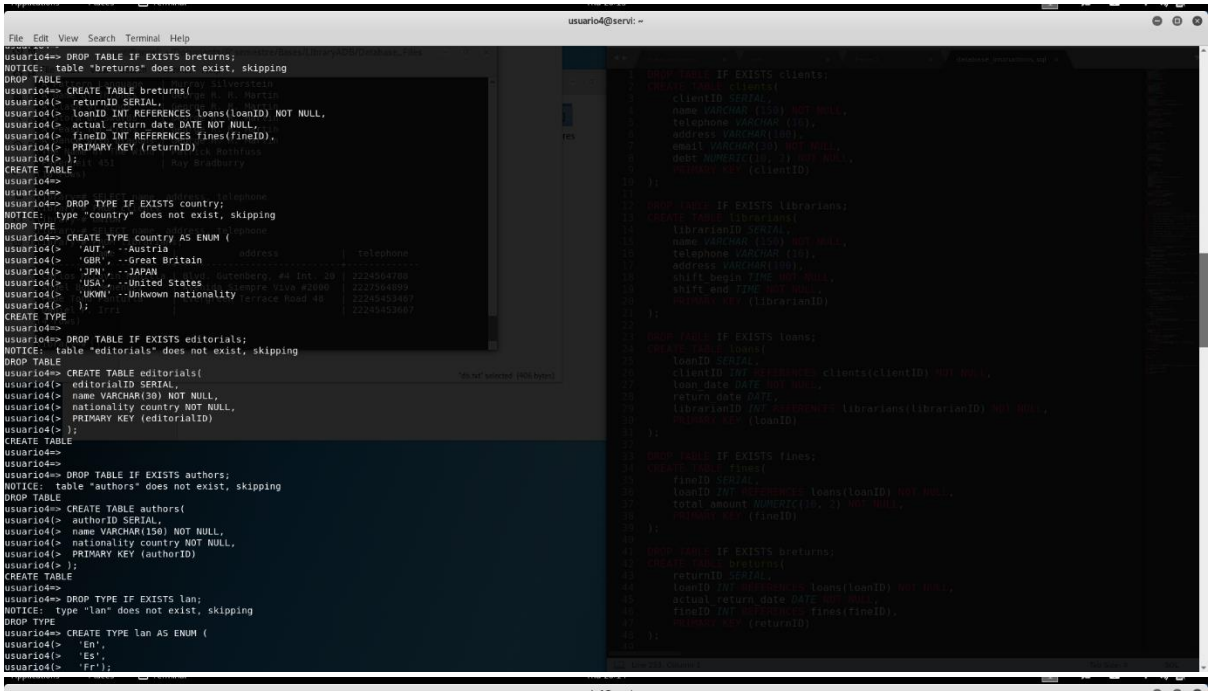
Rafael Antonio Comonfort Viveros A01324276
Angel Roberto Ruíz Mendoza A01324489

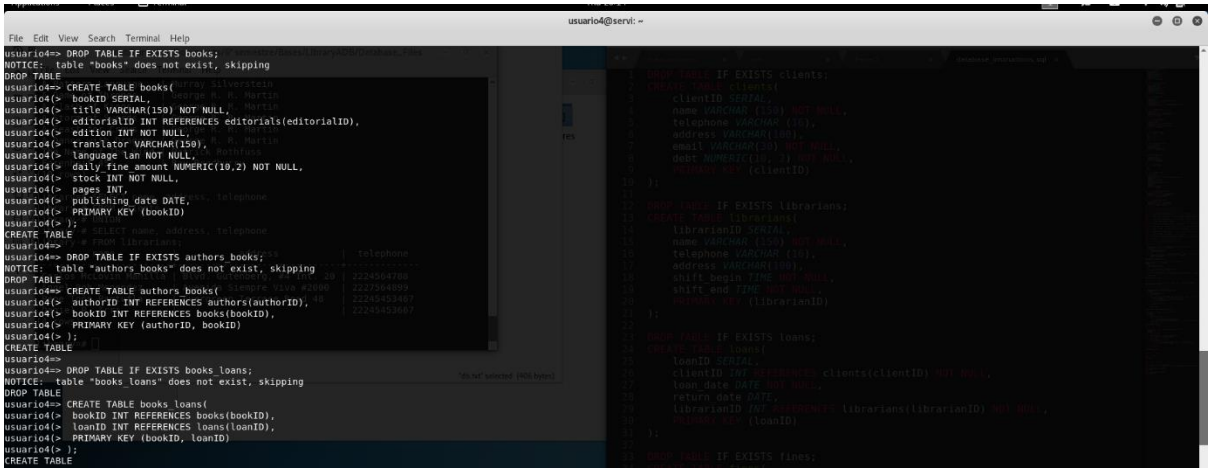
Database Creation

This are the database creation instructions, performed on the server, for all the tables in our relational diagram:



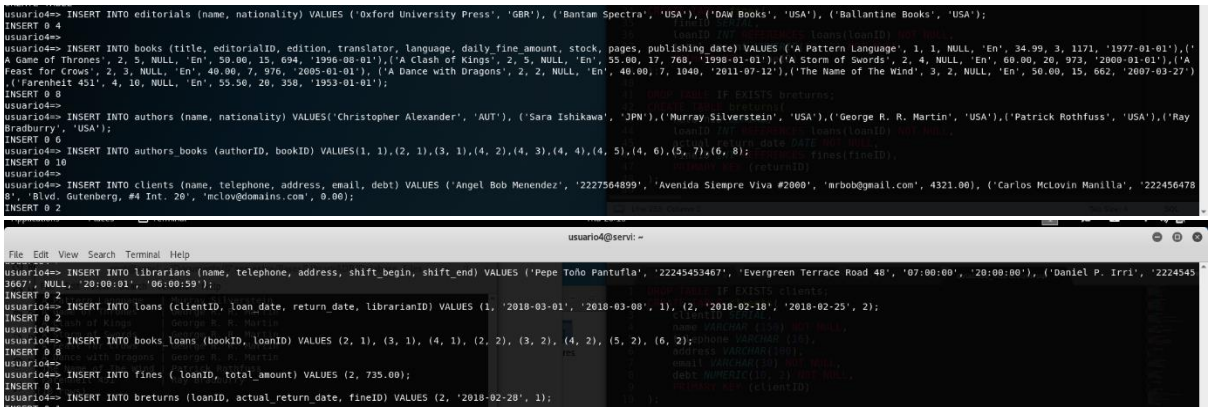

usuario4@servi: ~
File Edit View Search Terminal Help
usuario4=> DROP TABLE IF EXISTS clients;
NOTICE: table "clients" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE clients(
usuario4(> clientID SERIAL,
usuario4(> name VARCHAR (150) NOT NULL,
usuario4(> telephone VARCHAR (16),
usuario4(> address VARCHAR(100),
usuario4(> email VARCHAR(30) NOT NULL,
usuario4(> debt NUMERIC(10, 2) NOT NULL,
usuario4(> PRIMARY KEY (clientID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TABLE IF EXISTS librarians;
NOTICE: table "librarians" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE librarians(
usuario4(> librarianID SERIAL,
usuario4(> name VARCHAR (150) NOT NULL,
usuario4(> telephone VARCHAR (16),
usuario4(> address VARCHAR(100),
usuario4(> shift_begin TIME NOT NULL,
usuario4(> shift_end TIME NOT NULL,
usuario4(> PRIMARY KEY (librarianID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TABLE IF EXISTS loans;
NOTICE: table "loans" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE loans(
usuario4(> loanID SERIAL,
usuario4(> clientID INT REFERENCES clients(clientID) NOT NULL,
usuario4(> loan_date DATE NOT NULL,
usuario4(> return_date DATE,
usuario4(> librarianID INT REFERENCES librarians(librarianID) NOT NULL,
usuario4(> PRIMARY KEY (loanID)
usuario4(>);
CREATE TABLE
usuario4=> DROP TABLE IF EXISTS fines;
NOTICE: table "fines" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE fines(
usuario4(> fineID SERIAL,
usuario4(> loanID INT REFERENCES loans(loanID) NOT NULL,
usuario4(> total_amount NUMERIC(10, 2) NOT NULL,
usuario4(> PRIMARY KEY (fineID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TABLE IF EXISTS breturns;
NOTICE: table "breturns" does not exist, skipping
DROP TABLE


usuario4@servi: ~
File Edit View Search Terminal Help
usuario4=> DROP TABLE IF EXISTS breturns;
NOTICE: table "breturns" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE breturns(
usuario4(> returnID SERIAL,
usuario4(> loanID INT REFERENCES loans(loanID) NOT NULL,
usuario4(> actual_return_date DATE NOT NULL,
usuario4(> fineID INT REFERENCES fines(fineID),
usuario4(> PRIMARY KEY (returnID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=>
usuario4=> DROP TYPE IF EXISTS country;
NOTICE: type "country" does not exist, skipping
DROP TYPE
usuario4=> CREATE TYPE country AS ENUM (
usuario4(> 'AUT', --Austria
usuario4(> 'GBR', --Great Britain
usuario4(> 'JPN', --JAPAN
usuario4(> 'USA', --United States
usuario4(> 'UKWN' --Unkwown nationality
usuario4(>);
CREATE TYPE
usuario4=>
usuario4=> DROP TABLE IF EXISTS editorials;
NOTICE: table "editorials" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE editorials(
usuario4(> editorialID SERIAL,
usuario4(> name VARCHAR(30) NOT NULL,
usuario4(> nationality country NOT NULL,
usuario4(> PRIMARY KEY (editorialID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=>
usuario4=> DROP TABLE IF EXISTS authors;
NOTICE: table "authors" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE authors(
usuario4(> authorID SERIAL,
usuario4(> name VARCHAR(150) NOT NULL,
usuario4(> nationality country NOT NULL,
usuario4(> PRIMARY KEY (authorID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TYPE IF EXISTS lan;
NOTICE: type "lan" does not exist, skipping
DROP TYPE
usuario4=> CREATE TYPE lan AS ENUM (
usuario4(> 'En',
usuario4(> 'Es',
usuario4(> 'Ff');


usuario4@servi: ~
File Edit View Search Terminal Help
usuario4=> DROP TABLE IF EXISTS books;
NOTICE: table "books" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE books(
usuario4(> bookID SERIAL,
usuario4(> title VARCHAR(150) NOT NULL,
usuario4(> editorialID INT REFERENCES editorials(editorialID),
usuario4(> edition INT NOT NULL,
usuario4(> translator VARCHAR(150),
usuario4(> language lan NOT NULL,
usuario4(> daily_fine_amount NUMERIC(10,2) NOT NULL,
usuario4(> stock INT NOT NULL,
usuario4(> pages INT,
usuario4(> publishing_date DATE,
usuario4(> PRIMARY KEY (bookID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TABLE IF EXISTS authors_books;
NOTICE: table "authors_books" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE authors_books(
usuario4(> authorID INT REFERENCES authors(authorID),
usuario4(> bookID INT REFERENCES books(bookID),
usuario4(> PRIMARY KEY (authorID, bookID)
usuario4(>);
CREATE TABLE
usuario4=>
usuario4=> DROP TABLE IF EXISTS books_loans;
NOTICE: table "books_loans" does not exist, skipping
DROP TABLE
usuario4=> CREATE TABLE books_loans(
usuario4(> bookID INT REFERENCES books(bookID),
usuario4(> loanID INT REFERENCES loans(loanID),
usuario4(> PRIMARY KEY (bookID, loanID)
usuario4(>);
CREATE TABLE

Database Population

We made some sample tuples to verify everything gets inserted as it should, while also allowing us to test our functions and triggers.



Database Stored Programs

These are the created programs for this small data base.

Functions

- Calculates the owned fine of a book, given its ID and the supposed return date.

```
CREATE OR REPLACE FUNCTION FinePerBook(bookI INT, returnDate DATE)
   RETURNS NUMERIC(10,2) AS $finePerBook$
   DECLARE
      finePerBook NUMERIC(10,2);
      bookFine NUMERIC(10,2);
   BEGIN
      SELECT books.daily_fine_amount INTO bookFine FROM books WHERE
books.bookID = bookI;
      finePerBook:= bookFine * (CURRENT_DATE - returnDate);
      RETURN  finePerBook;
   END;
$finePerBook$ LANGUAGE plpgsql;
```

- Uses the above function, to calculate the total fine owned from a particular loan, it finds all books that were lent and sums the fine for all of them.

```
CREATE OR REPLACE FUNCTION CalculateTotalFine (lID INT)
RETURNS NUMERIC (10, 2) AS $total$
DECLARE
        total NUMERIC (10, 2);
        originalReturnDate DATE;
BEGIN
        SELECT return_date INTO originalReturnDate
        FROM loans l WHERE l.loanID = lID;

        SELECT SUM(FinePerBook(b.bookID, originalReturnDate)) INTO total
FROM books b INNER JOIN books_loans bl ON bl.loanID = lID;

        RETURN total;
END;
```

```
$total$ LANGUAGE plpgsql;
```

- This is the function called by our trigger, after a book return (breturns table) has been created, the system calculates automatically the fine due.

```
CREATE OR REPLACE FUNCTION DoFine()
   RETURNS TRIGGER AS $after_insert_breturns$
   DECLARE
      return_date_loan DATE;
      new_fineID INT;
   BEGIN
      SELECT loans.return_date INTO return_date_loan FROM loans
WHERE loans.loanID = NEW.loanID;
      IF return_date_loan < NEW.actual_return_date THEN
         SELECT CreateFine(NEW.loanID) INTO new_fineID;
         NEW.fineID:=new_fineID;
      ELSE
         NEW.fineID:=NULL;
      END IF;

      RETURN NEW;
   END;
$after_insert_breturns$ LANGUAGE plpgsql;
```

- Helper function to the one above, that actually creates the fine and returns its ID in case the return date was after the one specified by the loan.

```
CREATE OR REPLACE FUNCTION CreateFine(loID INT)
   RETURNS INT AS $fID$
   DECLARE
      fID INT;
      amount NUMERIC(10,2);
   BEGIN
      SELECT CalculateTotalFine(loID) INTO amount;
      INSERT INTO fines (loanID, total_amount) VALUES (loID, amount)
RETURNING fineID INTO fID;
      RETURN fID;
   END;
$fID$ LANGUAGE plpgsql;
```

Example: When we insert a return for a book that was due a day before, we get automatically its fine.

```
usuario4=> insert into breturns (loanID, actual_return_date) VALUES (3, '2018-03-09');
INSERT 0 1
usuario4=> select * from fines;
 fineid | loanid | total_amount
--------+--------+--------------
      1 |      2 |       735.00
      4 |      3 |        55.50
(2 rows)
```

Transaction

We developed this transaction inside PostgreSql, to perform all necessary modifications after a loan has been created. It updates the book stock and matches it to that loan. Notice this is implemented as a function, which deliberately locks the table and does a rollback in case of error. This function is expected to be called by an outside application, for each book selected in the loan (which in turn, is obtained from the front-end).

```
CREATE OR REPLACE FUNCTION LoanBook(loID INT, boID INT)
  RETURNS VOID AS $$
  DECLARE
    book_stock INT;
  BEGIN
      LOCK TABLE books IN ACCESS EXCLUSIVE MODE;
    SELECT books.stock INTO book_stock
    FROM books
    WHERE books.bookID = boID;
    IF book_stock > 0 THEN
      INSERT INTO books_loans(bookID, loanID) VALUES (boID, loID);
      UPDATE books SET stock = (stock - 1) WHERE books.bookID = boID;
    ELSE
      ROLLBACK;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

View

This is a simple view, that gets all books in the library.

```
CREATE VIEW allBooks AS
SELECT *
FROM books;
```

```
 bookid |        title         | editorialid | edition | translator | language | daily_fine_amount | stock | pages |
publishing_date
--------+----------------------+-------------+---------+------------+----------+-------------------+-------+-------+--
---------------
      1 | A Pattern Language   |           1 |       1 |            | En       |             34.99 |     3 |  1171 |
1977-01-01
      2 | A Game of Thrones    |           2 |       5 |            | En       |             50.00 |    15 |   694 |
1996-08-01
      3 | A Clash of Kings     |           2 |       5 |            | En       |             55.00 |    17 |   768 |
1998-01-01
      4 | A Storm of Swords    |           2 |       4 |            | En       |             60.00 |    20 |   973 |
2000-01-01
      5 | A Feast for Crows    |           2 |       3 |            | En       |             40.00 |     7 |   976 |
2005-01-01
      6 | A Dance with Dragons |           2 |       2 |            | En       |             40.00 |     7 |  1040 |
2011-07-12
      7 | The Name of The Wind |           3 |       2 |            | En       |             50.00 |    15 |   662 |
2007-03-27
      8 | Farenheit 451        |           4 |      10 |            | En       |             55.50 |    20 |   358 |
1953-01-01
(8 rows)
```

The result of calling *“SELECT * FROM allBooks;”*

Query

Finally, two simple queries were written to gather some information and showcase the use of INNER JOIN and UNION structures:

- Gathers books by author

```
SELECT b.title, a.name AS Author_Name
```

FROM authors a INNER JOIN (books b INNER JOIN authors_books ab ON b.bookID=ab.bookID) ON a.authorID=ab.authorID;

```
usuario4=> select b.title, a.name as author_name from authors a inner join (books b inner join authors_books ab on b.
bookid = ab.bookid) ON a.authorID = ab.authorID;
        title         |      author_name
----------------------+----------------------
 A Pattern Language   | Christopher Alexander
 A Pattern Language   | Sara Ishikawa
 A Pattern Language   | Murray Silverstein
 A Game of Thrones    | George R. R. Martin
 A Clash of Kings     | George R. R. Martin
 A Storm of Swords    | George R. R. Martin
 A Feast for Crows    | George R. R. Martin
 A Dance with Dragons | George R. R. Martin
 The Name of The Wind | Patrick Rothfuss
 Farenheit 451        | Ray Bradburry
(10 rows)
```

- Gets addresses from clients and librarians

  SELECT name, address, telephone
  FROM clients
  UNION
  SELECT name, address, telephone
  FROM librarians;

```
usuario4=> select name, address, telephone from clients union select name, address, telephone from librarians;
          name          |           address           |  telephone
------------------------+-----------------------------+-------------
 Daniel P. Irri         |                             | 22245453667
 Pepe Toño Pantufla     | Evergreen Terrace Road 48   | 22245453467
 Angel Bob Menendez     | Avenida Siempre Viva #2000  | 2227564899
 Carlos McLovin Manilla | Blvd. Gutenberg, #4 Int. 20 | 2224564788
(4 rows)
```