**ОТЧЁТ**

**По циклу лабораторных работ**

По дисциплине «Базы данных».

Выполнила студентка гр. 5130904/10104 Ролецкая С.Н.

Руководитель Александрова О.В.

«30» сентября 2023

Оглавление

[Введение. 3](#_Toc146981543)

[Постановка задачи. 3](#_Toc146981544)

[Структура базы данных. 5](#_Toc146981545)

[Основная часть. 6](#_Toc146981546)

[Работа №1. Создание базы данных. 6](#_Toc146981547)

[Работа №2. Язык SQL-DML. 8](#_Toc146981548)

[Работа №3. Представления, хр. процедуры, триггера и курсоры. 14](#_Toc146981549)

[Курсовая работа. 22](#_Toc146981550)

[Постановка задачи. 22](#_Toc146981551)

[Требования к клиентскому приложению. 22](#_Toc146981552)

[Описание работы. 23](#_Toc146981553)

[Вывод. 30](#_Toc146981554)

# Введение.

## Постановка задачи.

Вариант 2. Автоматизация работы библиотеки.

В рамках данного цикла лабораторных работ необходимо автоматизировать работу

библиотеки. Для этого в рамках базы данных PostgreSQL необходимо создать объекты в

схеме вашего пользователя (логин и пароль пользователя для доступа к базе данных

студент должен получить у преподавателя) и написать клиентское приложение на базе

компонентов ADO.NET или JDBC.

Библиотека является государственным предприятием и выдает литературу жителям

Выборгского района (далее клиентам) на безвозмездной основе. Клиенты характеризуются

фамилией, именем и номером паспорта. Клиент не может иметь на руках более 10 книг.

Книги библиотеки характеризуются полным наименованием и разбиты на 3 категории:

обычные, редкие и уникальные. Максимальные сроки удержания клиентом книги зависят

от категории книги и составляют соответственно 60, 21 и 7 дней. При невозвращении

клиентом книги в оговоренный срок, клиент обязан уплатить штраф из расчета 10, 50 и

300 рублей за день задержки так же в зависимости от категории книги. Экземпляров книг

в библиотеке ограниченное количество и при выдаче и возврате их необходимо вести учет

числа экземпляров каждой книги. Библиотекари при выдаче и приеме книг обязаны

учитывать все операции в журнале. При выдаче книги в журнал необходимо заносить дату

выдачи (автоматически: текущая дата), клиента (библиотекарь выбирает из справочника),

книгу (библиотекарь выбирает из справочника) и максимальную дату возврата

(автоматически: текущая дата + количество дней, в зависимости от типа книги). При

приеме книги устанавливается реальная дата возврата и, при необходимости, исчисляется

штраф.

База данных должна удовлетворять следующим требованиям:

1. Контроль целостности данных, используя механизм связей
2. Операции модификации групп данных и данных в связанных таблицах должны быть выполнены в рамках транзакций.
3. Логика работы приложения должна контролироваться триггерами. В частности:

* Триггер должен не позволять выдать книгу, не имеющуюся в наличии.
* Триггер должен контролировать, что выдаваемая книга выдается клиенту, имеющему на руках менее 10 книг

1. Все операции вычисления различных показателей (из требований к клиентскому приложению) должны реализовываться хранимыми процедурами.

Требования к клиентскому приложению:

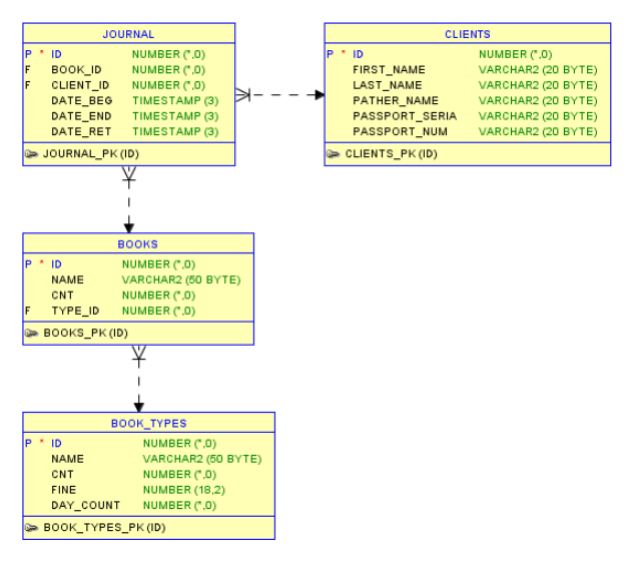
1. Необходимо реализовать интерфейсы для ввода, модификации и удаления

* Клиентов
* Книг
* Типов книг.

1. В главном окне приложения должен быть реализован журнал библиотекаря с возможностью выдачи и приема книг.
2. Необходимо реализовать возможность просмотра библиотекарем следующих показателей:

* Число книг на руках у клиента.
* Размер самого большого штрафа
* Размер штрафа данного клиента.
* Три самые популярные книги

## Структура базы данных.



Таблицы:

|  |  |
| --- | --- |
| **Имя таблицы** | **Расшифровка** |
| clients | Читатели (фамилия, имя, паспорт) |
| books | Книги (наименование, количество) |
| book\_types | Типы книг (название, штраф, число дней) |
| journal | Журнал библиотекаря |

# Основная часть.

## Работа №1. Создание базы данных.

1. Запустить pgAdmin
2. Создать соединение, используя логин и пароль.
3. Создать необходимые таблицы.
4. Создать связи между таблицами:

|  |  |  |
| --- | --- | --- |
| **Название** | **Primary Key** | **Foreign Key** |
| fk\_journal\_clients | clients.id | journal.client\_id |
| fk\_journal\_books | books.id | journal.book\_id |
| fk\_books\_book\_types | book\_types.id | books.type\_id |

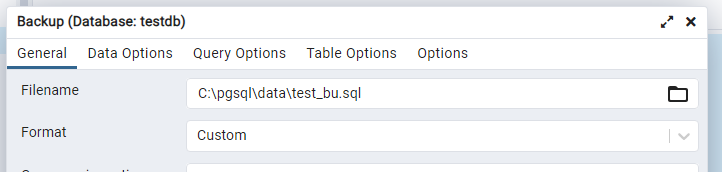
Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

1. Создать Backup базы и запомнить место его расположения.

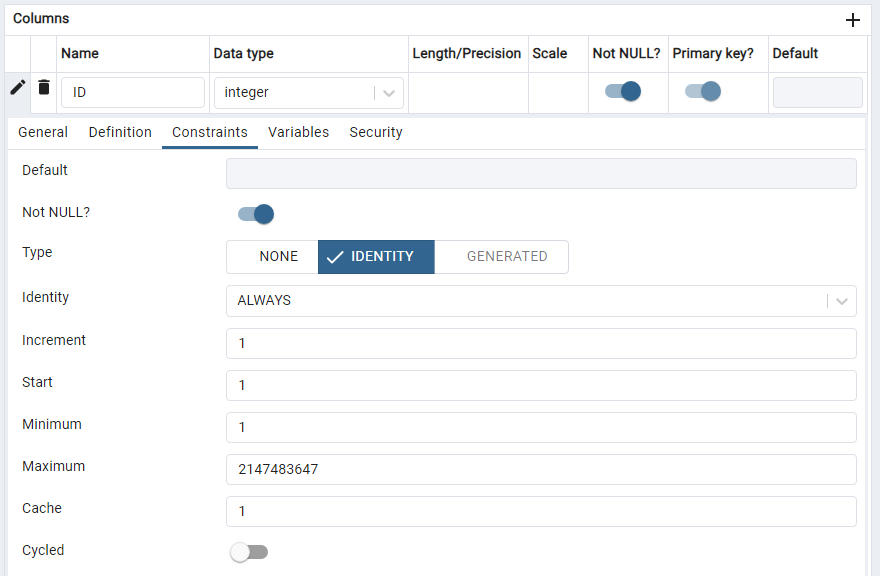


1. Удалить базу с сервера
2. Восстановить базу из Backup базы

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

1. Для всех таблиц реализовать автоматическое заполнение первичного ключа при вставке данных (автоинкремент).



## Работа №2. Язык SQL-DML.

**Выборка данных**

* однотабличная выборка

1. Вывести все строки из журнала библиотекаря дата возврата которых меньше некоторой даты

|  |
| --- |
| SELECT \* FROM public."JOURNAL"  WHERE "DATE\_RET" < '2023-09-01 00:00:00' |

1. Посчитать количество книг, которых нет в наличии

|  |
| --- |
| SELECT COUNT("NAME") FROM public."BOOKS"  WHERE "CNT" = 0 |

* выборка с подзапросами

1. Вывести все книги типа «уникальные», которые на руках у читателей

|  |
| --- |
| SELECT "NAME" FROM public."BOOKS"  WHERE "TYPE\_ID" = 3 AND "ID" IN  (SELECT "BOOK\_ID" FROM public."JOURNAL") |

* соединение таблиц (join)

1. Вывести журнал библиотекаря, читателей, включая читателей, которые не брали книг и книги, включая книги, которых не выдавали

|  |
| --- |
| SELECT \* FROM public."JOURNAL"  FULL OUTER JOIN public."CLIENTS" ON "JOURNAL"."CLIENT\_ID" = "CLIENTS"."ID"  FULL OUTER JOIN public."BOOKS" ON "JOURNAL"."BOOK\_ID" = "BOOKS"."ID" |

* для реализации проекта

1. Число книг на руках у заданного клиента.

|  |
| --- |
| SELECT COUNT("BOOK\_ID") FROM public."JOURNAL"  WHERE "CLIENT\_ID" = 5 |

1. Размер штрафа заданного клиента.

|  |
| --- |
| SELECT SUM(EXTRACT(DAY FROM(j."DATE\_RET" - j."DATE\_END")) \*  (SELECT bt."FINE" FROM public."BOOKS" books LEFT JOIN "BOOK\_TYPES" bt ON books."TYPE\_ID" = bt."ID" WHERE books."ID" = j."BOOK\_ID")) AS sum  FROM public."JOURNAL" j  LEFT JOIN "CLIENTS" c ON j."CLIENT\_ID" = c."ID"  WHERE EXTRACT(DAY FROM(j."DATE\_RET" - j."DATE\_END")) > 1 AND c."ID" = '3'; |

1. Размер самого большого штрафа

|  |
| --- |
| SELECT SUM(EXTRACT(DAY FROM(j."DATE\_RET" - j."DATE\_END")) \*  (SELECT bt."FINE" FROM public."BOOKS" books  LEFT JOIN "BOOK\_TYPES" bt ON books."TYPE\_ID" = bt."ID"  WHERE books."ID" = j."BOOK\_ID")) AS sum FROM public."JOURNAL" j  WHERE EXTRACT (DAY FROM (j."DATE\_RET" - j."DATE\_END")) > 1  GROUP BY j."CLIENT\_ID"  ORDER BY sum DESC  LIMIT 1; |

1. Три самые популярные книги

|  |
| --- |
| SELECT "BOOKS"."NAME", COUNT("BOOK\_ID") as count FROM public."JOURNAL"  INNER JOIN public."BOOKS" ON "BOOKS"."ID" = "JOURNAL"."BOOK\_ID"  GROUP BY "BOOKS"."NAME", "BOOK\_ID" ORDER BY count DESC LIMIT 3 |

**Вставка данных**

* однотабличная вставка

1. Добавить нового клиента

|  |
| --- |
| INSERT INTO public."CLIENTS" (  "LAST\_NAME", "FIRST\_NAME", "FATHER\_NAME", "PASSPORT\_SERIA", "PASSPORT\_NUM")  VALUES (  'Тесленко'::character varying, 'Ольга'::character varying, 'Руслановна'::character varying, '6587'::character varying, '345678'::character varying)  returning "ID"; |

* многотабличная вставка в рамках транзакции

1. Добавить в рамках транзакции клиента, книгу и запись в журнал библиотекаря о выдачи книги этому клиенту

|  |
| --- |
| START TRANSACTION;  INSERT INTO public."CLIENTS" (  "LAST\_NAME", "FIRST\_NAME", "FATHER\_NAME", "PASSPORT\_SERIA", "PASSPORT\_NUM") VALUES (  'Тесленко'::character varying, 'Людмила'::character varying,  'Григорьевна'::character varying, '6907'::character varying, '125678'::character varying)  returning "ID" ;    INSERT INTO public."BOOKS" (  "NAME", "CNT", "TYPE\_ID") VALUES (  'Над пропастью во ржи'::character varying, '2'::integer, '2'::integer)  returning "ID";    INSERT INTO public."JOURNAL" (  "DATE\_BEG" , "DATE\_END", "BOOK\_ID", "CLIENT\_ID") VALUES (  '2023-09-05 00:00:00'::timestamp, '2023-10-04 00:00:00'::timestamp,  (SELECT "ID" FROM public."BOOKS" WHERE "NAME" = 'Над пропастью во ржи'  AND "CNT" = '2' AND "TYPE\_ID" = '2'),  (SELECT "ID" FROM public."CLIENTS" WHERE "PASSPORT\_SERIA" = '6907'  AND "PASSPORT\_NUM" = '125678'))  returning "ID";    COMMIT; |

1. Добавить запись в журнал, в случае если книг у данного клиента больше 10, транзакцию откатить

|  |
| --- |
| DO $$  BEGIN  INSERT INTO public."JOURNAL" (  "DATE\_BEG", "DATE\_END", "BOOK\_ID", "CLIENT\_ID") VALUES (  '2023-09-05 00:00:00'::timestamp, '2023-10-04 00:00:00'::timestamp,  (SELECT "ID" FROM public."BOOKS" WHERE "NAME" = 'Над пропастью во ржи'),  (SELECT "ID" FROM public."CLIENTS" WHERE "PASSPORT\_SERIA" = '4352' AND "PASSPORT\_NUM" = '908534'));    IF (SELECT COUNT("BOOK\_ID") > '10' FROM public."JOURNAL" as BookCount  WHERE "CLIENT\_ID" = (  SELECT "ID" FROM public."CLIENTS"  WHERE "PASSPORT\_SERIA" = '4352' AND "PASSPORT\_NUM" = '908534'))  THEN ROLLBACK;  END IF;  END;  $$; |

**Удаление данных**

* удаление по фильтру и удаление из связанных таблиц

1. Удалить книги, не имеющие ссылок из записей в журнале

|  |
| --- |
| DELETE FROM public."BOOKS"  WHERE NOT EXISTS  (SELECT "BOOK\_ID" FROM public."JOURNAL"  WHERE public."BOOKS"."ID" = public."JOURNAL"."BOOK\_ID") |

* удаление в рамках транзакции

1. Удалить в рамках транзакции книгу и записи о ее выдаче

|  |
| --- |
| START TRANSACTION;  DELETE FROM public."JOURNAL"  WHERE "BOOK\_ID" = (SELECT "ID" FROM public."BOOKS" WHERE "NAME" = 'Мэри Поппинс');  DELETE FROM public."BOOKS" WHERE "NAME" = 'Мэри Поппинс';  COMMIT; |

1. то же, что и п1, но транзакцию откатить

|  |
| --- |
| START TRANSACTION;  DELETE FROM public."JOURNAL"  WHERE "BOOK\_ID" = (SELECT "ID" FROM public."BOOKS" WHERE "NAME" = 'Мэри Поппинс');  DELETE FROM public."BOOKS" WHERE "NAME" = 'Мэри Поппинс';  ROLLBACK; |

**Модификация данных**

* модификация по фильтру

1. Изменить количество экземпляров книг, которые выданы заданному клиенту

|  |
| --- |
| UPDATE public."BOOKS"  SET "CNT" = '4'  WHERE "ID" = (SELECT "BOOK\_ID" FROM public."JOURNAL" WHERE "CLIENT\_ID" = '8') |

* модификация в рамках транзакции

1. В рамках транзакции поменять заданную книгу во всех записях журнала на другую и удалить ее.

|  |
| --- |
| START TRANSACTION;  UPDATE public."JOURNAL"  SET "BOOK\_ID" = 3  WHERE "BOOK\_ID" = 1;  DELETE FROM public."BOOKS"  WHERE "ID" = 1;  COMMIT; |

1. то же, что и п1, но транзакцию откатить

|  |
| --- |
| START TRANSACTION;  UPDATE public."JOURNAL"  SET "BOOK\_ID" = 3  WHERE "BOOK\_ID" = 1;  DELETE FROM public."BOOKS"  WHERE "ID" = 1;  ROLLBACK; |

## Работа №3. Представления, хр. процедуры, триггера и курсоры.

**Представления**

1. Создать представление, отображающее все книги и читателей, о которых найдены записи в журнале с заданной даты по заданную дату

|  |
| --- |
| CREATE VIEW clients\_books\_per\_days AS  SELECT "CLIENTS"."LAST\_NAME", "CLIENTS"."FIRST\_NAME", "CLIENTS"."FATHER\_NAME",  "BOOKS"."NAME", "JOURNAL"."DATE\_BEG" FROM public."CLIENTS"  INNER JOIN public."JOURNAL" ON "CLIENTS"."ID" = "JOURNAL"."CLIENT\_ID"  INNER JOIN public."BOOKS" ON "BOOKS"."ID" = "JOURNAL"."BOOK\_ID"  WHERE "JOURNAL"."DATE\_BEG" > '2023-08-05 00:00:00'::timestamp  AND "JOURNAL"."DATE\_BEG" < '2023-09-05 00:00:00'::timestamp |

1. Создать представление, отображающее всех читателей и количество книг, находящихся у них на руках

|  |
| --- |
| CREATE VIEW clients\_per\_books AS  WITH counts AS (SELECT "CLIENT\_ID", COUNT("BOOK\_ID") as book\_count  FROM public."JOURNAL" GROUP BY "CLIENT\_ID")  SELECT "CLIENTS"."LAST\_NAME", "CLIENTS"."FIRST\_NAME", "CLIENTS"."FATHER\_NAME",  counts.book\_count FROM public."CLIENTS"  INNER JOIN counts ON "CLIENTS"."ID" = counts."CLIENT\_ID" |

**Хранимые процедуры**

* без параметров

1. Создать хранимую процедуру, выводящую все книги и среднее время, на которое их брали в днях

|  |
| --- |
| CREATE OR REPLACE FUNCTION BooksAveregeF()  RETURNS TABLE(name text, averege interval)  AS $$  WITH avereges AS (SELECT "BOOK\_ID", AVG("DATE\_RET" - "DATE\_BEG") as averege  FROM public."JOURNAL" GROUP BY "BOOK\_ID")  SELECT "BOOKS"."NAME", avereges.averege FROM public."BOOKS"  INNER JOIN avereges ON "BOOKS"."ID" = avereges."BOOK\_ID";  $$  LANGUAGE 'sql' STABLE; |
| SELECT \* FROM booksaveregef(); |

* с входными параметрами

1. Создать хранимую процедуру, имеющую два параметра «книга1» и «книга2». Она должна возвращать клиентов, которые вернули «книгу1» быстрее чем «книгу2». Если какой-либо клиент не брал одну из книг – он не рассматривается.

|  |
| --- |
| CREATE OR REPLACE FUNCTION ClientsReturnF(book1 varchar, book2 varchar)  RETURNS varchar  AS $$  SELECT CONCAT(c."LAST\_NAME", ' ', c."FIRST\_NAME", ' ', c."FATHER\_NAME") as clients  FROM (SELECT j1."CLIENT\_ID"  FROM public."JOURNAL" j1  JOIN public."JOURNAL" j2 ON j1."CLIENT\_ID" = j2."CLIENT\_ID"  AND j1."BOOK\_ID" =  (SELECT "ID" FROM public."BOOKS"  WHERE "NAME" = book1)  AND j2."BOOK\_ID" =  (SELECT "ID" FROM public."BOOKS"  WHERE "NAME" = book2)  WHERE j1."DATE\_RET" IS NOT NULL  AND (j2."DATE\_RET" IS NULL OR j2."DATE\_RET" > j1."DATE\_RET")) data  LEFT JOIN public."CLIENTS" c ON data."CLIENT\_ID" = c."ID"  $$  LANGUAGE 'sql' STABLE; |
| SELECT \* FROM ClientsReturnF('Физика 7 класс', 'Том Сойер'); |

* с выходными параметрами

1. Создать хранимую процедуру с входным параметром «клиент» и выходным параметром – количеством книг, находящихся у него

|  |
| --- |
| CREATE OR REPLACE FUNCTION CountBooksF(last\_name varchar, first\_name varchar, father\_name varchar)  RETURNS bigint  AS $$  WITH clients AS (SELECT "ID" FROM public."CLIENTS"  WHERE "LAST\_NAME" = last\_name AND "FIRST\_NAME" = first\_name AND "FATHER\_NAME" = father\_name)  SELECT COUNT("BOOK\_ID") as book\_count FROM public."JOURNAL"  INNER JOIN clients ON clients."ID" = "JOURNAL"."CLIENT\_ID"  GROUP BY "CLIENT\_ID";  $$  LANGUAGE 'sql' STABLE; |
| SELECT \* FROM countbooksf('Ролецкая', 'Софья', 'Николаевна'); |

1. Создать хранимую процедуру с входным параметром «книга» и двумя выходными параметрами, возвращающими самое большое время на который брали книгу и читателя, поставившего рекорд

|  |
| --- |
| CREATE OR REPLACE FUNCTION DayRecordF(book\_name varchar)  RETURNS TABLE(last\_name text, first\_name text, father\_name text, max\_days interval)  AS $$  SELECT "CLIENTS"."LAST\_NAME", "CLIENTS"."FIRST\_NAME", "CLIENTS"."FATHER\_NAME",  ("JOURNAL"."DATE\_RET" - "JOURNAL"."DATE\_BEG") FROM public."JOURNAL"  INNER JOIN public."CLIENTS" ON "CLIENTS"."ID" = "JOURNAL"."CLIENT\_ID"  WHERE "BOOK\_ID" = (SELECT "ID" FROM public."BOOKS" WHERE "NAME" = book\_name)  AND ("DATE\_RET" - "DATE\_BEG")  = (SELECT MAX("DATE\_RET" - "DATE\_BEG") FROM public."JOURNAL");  $$  LANGUAGE 'sql' STABLE; |

**Триггера**

* Триггера на вставку

1. Создать триггер, который не позволяет добавить читателя с номером паспорта, который уже есть у существующего читателя

|  |
| --- |
| CREATE OR REPLACE FUNCTION processInsertTrigger() RETURNS TRIGGER AS $$  BEGIN  IF EXISTS(SELECT "ID" FROM public."CLIENTS" WHERE  "PASSPORT\_SERIA" = NEW."PASSPORT\_SERIA" AND "PASSPORT\_NUM" = NEW."PASSPORT\_NUM") THEN  RAISE EXCEPTION 'ERROR';  ELSE NEW.VALUE = NEW.VALUE;  END IF;  RETURN NEW;  END;  $$  LANGUAGE plpgsql VOLATILE;  CREATE TRIGGER insertTrigger  BEFORE INSERT ON public."CLIENTS"  FOR EACH ROW EXECUTE PROCEDURE processInsertTrigger(); |

* Триггера на модификацию

1. Создать триггер, который не позволяет установить реальную дату возврата журнала библиотекаря меньше, чем дата выдачи

|  |
| --- |
| CREATE OR REPLACE FUNCTION processUpdateTrigger() RETURNS TRIGGER AS $$  BEGIN  IF NEW."DATE\_RET" < OLD."DATE\_BEG" THEN  RAISE EXCEPTION 'Дата возврата не может быть меньше даты выдачи!';  ELSE NEW."DATE\_RET" = NEW."DATE\_RET";  END IF;  RETURN NULL;  END;  $$  LANGUAGE plpgsql VOLATILE;  CREATE TRIGGER updateTrigger  AFTER UPDATE OF "DATE\_RET" ON public."JOURNAL"  FOR EACH ROW EXECUTE PROCEDURE processUpdateTrigger(); |

* Триггера на удаление

1. Создать триггер, который при удалении строки журнала в случае, если книга не возвращена - откатывает транзакцию

|  |
| --- |
| CREATE OR REPLACE FUNCTION processDeleteTrigger() RETURNS TRIGGER AS $$  BEGIN  DELETE FROM public."JOURNAL" WHERE "ID" = OLD."ID";  IF (OLD."DATE\_RET" IS NULL) THEN  BEGIN  RAISE EXCEPTION 'Нельзя удалить запись, пока книгу не вернули';  ROLLBACK;  END;  END IF;  RETURN OLD;  END;  $$  LANGUAGE plpgsql VOLATILE;  CREATE TRIGGER deleteTrigger  AFTER DELETE ON public."JOURNAL"  FOR EACH ROW EXECUTE PROCEDURE processDeleteTrigger(); |

**Курсоры**

* Хранимая процедура для расчета суммы штрафов библиотеки

Необходимо реализовать хранимую процедуру, рассчитывающую сумму штрафов,

полученную библиотекой за некоторый период времени. Хранимая процедура должна иметь два входных параметра задающие интервал времени и один выходной, в котором возвращать размер штрафа.

Предлагаемый алгоритм: создаем курсор, который пробегает по строкам журнала, реальная дата возврата которых попадает в заданный интервал. Для каждой строки рассчитываем размер штрафа и суммируем его в некоторой переменной, значение которой по окончании работы курсора будет выдано в качестве выходного параметр с общей суммой штрафов.

|  |
| --- |
| CREATE OR REPLACE FUNCTION clientFine(first\_date interval, last\_date interval)  RETURNS numeric  AS $$  DECLARE  sum numeric DEFAULT 0;  fine numeric;  crs CURSOR FOR SELECT EXTRACT(DAY FROM(j."DATE\_RET" - j."DATE\_END")) \*  (SELECT bt."FINE"  FROM public."BOOKS" books  LEFT JOIN "BOOK\_TYPES" bt ON books."TYPE\_ID" = bt."ID"  WHERE books."ID" = j."BOOK\_ID")  FROM public."JOURNAL" j  LEFT JOIN "CLIENTS" c ON j."CLIENT\_ID" = c."ID"  WHERE EXTRACT(DAY FROM(j."DATE\_RET" - j."DATE\_END")) > 1 AND j."DATE\_BEG" >= first\_date AND j."DATE\_RET" <= last\_date;  BEGIN  OPEN crs;  LOOP  FETCH crs INTO fine;  EXIT WHEN NOT FOUND;  sum := sum + fine;  END LOOP;    RETURN sum;    END;  $$  LANGUAGE 'plpgsql' VOLATILE; |
| SELECT clientFine('2023-08-18 00:00:00', '2023-10-25 00:00:00'); |

* Хранимая процедура для расчета трех самых популярных книг

Необходимо реализовать хранимую процедуру, выбирающую три самые популярные книги за некоторый интервал времени. Хранимая процедура должна иметь два входных параметра задающие интервал времени.

Предлагаемый алгоритм: создаем три переменные, хранящие идентификаторы самых популярных книг и 3 переменные, соответственно хранящие число их выдачей. Создаем курсор, который пробегает по всем книгам, реальная дата выдачи которых попадает в заданный интервал. Для каждой книги рассчитываем количество ее выдачей и, в случае если она была выдана большее число раз, нежели одна из сохраненных в наших переменных, то заменяем ее новой. По окончании работы курсора выбираем идентификаторы самых популярных книг.

|  |
| --- |
| CREATE OR REPLACE FUNCTION booksTop(first\_date interval, last\_date interval)  RETURNS text  AS $$  DECLARE  book1 text;  count1 bigint;  book2 text;  count2 bigint;  book3 text;  count3 bigint;  result text;  crs CURSOR FOR SELECT "BOOKS"."NAME", COUNT("BOOK\_ID") as count FROM public."JOURNAL"  INNER JOIN public."BOOKS" ON "BOOKS"."ID" = "JOURNAL"."BOOK\_ID"  WHERE "DATE\_BEG" >= first\_date AND "DATE\_RET" <= last\_date  GROUP BY "BOOKS"."NAME", "BOOK\_ID" ORDER BY count DESC LIMIT 3;  BEGIN  OPEN crs;  FETCH crs INTO book1, count1;  FETCH crs INTO book2, count2;  FETCH crs INTO book3, count3;  CLOSE crs;    result := book1 || ' - ' || CAST(count1 AS VARCHAR(15)) || '; '  || book2 || ' - ' || CAST(count2 AS VARCHAR(15)) || '; '  || book3 || ' - ' || CAST(count3 AS VARCHAR(15)) || ';';    RETURN result;    END;  $$  LANGUAGE 'plpgsql' VOLATILE; |
| SELECT bookstop('2023-08-18 00:00:00', '2023-10-25 00:00:00'); |

# Курсовая работа.

## Постановка задачи.

Приложение представляет собой клиентскую часть GUI для БД, спроектированной в семестре. Использованный язык GUI - Java.

Приложение включает следующие элементы:

1. отдельный файл параметров подключения к БД, который задаются файлом (формат xml file);
2. авторизацию с хешированием паролей для пользователей (2 пользователя);
3. меню приложения содержит группы элементов: СПРАВОЧНИКИ, ЖУРНАЛЫ, ОТЧЕТЫ;
4. приложение должно обеспечивать ввод, редактирование, просмотр информации из БД;
5. используется пользовательский элементы вида GridView для обзора таблиц (представлений) БД;
6. GridView, который содержит внешние ключи, должен содержать нужные/необходимые атрибуты внешней таблицы.
7. есть отдельная форма для работы с записью таблицы, содержащая эл-ты интерфейса ДОБАВИТЬ, ИЗМЕНИТЬ, УДАЛИТЬ, ЗАКРЫТЬ; открывается из GridView

### Требования к клиентскому приложению.

1. Необходимо реализовать интерфейсы для ввода, модификации и удаления
   * Клиентов
   * Книг
   * Типов книг.
2. В главном окне приложения должен быть реализован журнал библиотекаря

с возможностью выдачи и приема книг.

1. Необходимо реализовать возможность просмотра библиотекарем следующих показателей:
   * Число книг на руках у клиента.
   * Размер самого большого штрафа
   * Размер штрафа данного клиента.
   * Три самые популярные книги

## Описание работы.

Клиентское приложение представляет собой GUI, написанную на языке Java с использованием фреймворка JavaFX. Взаимодействие с базой данных происходит посредством фреймворка Hibernate.

* Экран блокировки

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

* + Неверный логин или пароль

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Автоматически созданное описание

* Главное окно приложения и таблица журнала

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

* + Вставка в журнал

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

* Справочники

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

* + Таблица читателей

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

* + - Вставка в таблицу читателей

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

* + Таблица книг

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

* + - Вставка в таблицу книг

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

* + Таблица типов книг

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

* + - Вставка в таблицу типов книг

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

* Отчеты

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

* + Просмотр числа книг на руках у клиента

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

* + - Неправильный формат данных

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

* + Просмотр размера самого большого штрафа

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

* + Просмотр размера штрафа данного клиента.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

* + Просмотр трех самых популярных книг

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

# Вывод.

В ходе работы ознакомилась с СУБД postgresql, были изучены основы языка sql и графический интерфейс PgAdmin для работы с ним.  
При выполнении курсовой работы были освоены навыки взаимодействия с базами данных с помощью программного языка Java и различные фреймворки для работы с базой.