**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе “Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL”

по дисциплине «**Базы данных**»

Автор: Жигалова Анастасия Евгеньевна

Факультет: ИКТ

Группа: K32392

Преподаватель: Говорова М. М.

Дата: 03.05.2023



Санкт-Петербург 2023

# Цель работы

Овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

# Практическое задание

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
   1. Создайте запросы:
      1. Вывести список читателей, имеющих на руках книги, переведенные с английского языка, изданные позднее 2000 года.
      2. Вывести список читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
      3. Найти количество читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
      4. Вывести список книг, которые находятся в библиотеке в единственном экземпляре.
      5. Подсчитать количество читателей, которые не обращались в библиотеку в течение года.
      6. Подсчитать количество читателей библиотеки по уровню образования.
      7. Вывести список книг по программированию на С#, экземпляры которых отсутствуют в библиотеке, и которые должны быть возвращены не позднее, чем через 3 дня.
   2. Создать представления для администрации библиотеки, содержащие:
      1. сведения о должниках;
      2. сведения о наиболее популярных книгах (все экземпляры находятся на руках у читателей).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

# Схема базы данных

Изображение выглядит как диаграмма

Автоматически созданное описание

# Выполнение

### База Данных: library\_bd

## Создание запросов к бд

### Вывести список читателей, имеющих на руках книги, переведенные с английского языка, изданные позднее 2000 года.

SELECT DISTINCT f.number\_ticket, f.reader  
FROM book b  
 JOIN instance i ON b.id\_book = i.id\_book  
 JOIN issuance\_books ib ON i.id\_instance = ib.id\_instance  
 JOIN form f ON f.number\_ticket = ib.number\_ticket  
WHERE b.language = 'английский' AND b.year\_of\_publication > 2000;

К исходной таблице book присоединяются 3 дополнительные таблицы с помощью оператора JOIN и отсеиваются все книги, которые не имеют английский язык и год публикации ниже 2001. В итоге у нас остаются только формуляры имеющие на руках книги переведенные с английского языка с годом публикации > 2000

#### Вывод таблицы:

Изображение выглядит как стол

Автоматически созданное описание

### Вывести список читателей, не вернувших в срок книги и имеющих на руках более десяти книг.

SELECT DISTINCT f.number\_ticket, f.reader  
FROM form f  
 JOIN issuance\_books ib on f.number\_ticket = ib.number\_ticket  
WHERE (ib.date\_of\_return is null and ib.date\_of\_issue < *current\_date* - interval '10 day')  
GROUP BY f.number\_ticket  
HAVING *count*(\*) > 10;

С помощью оператора JOIN присоединяется таблица выдачи книг. В условиях оператора WHERE отсеиваются все строки таблицы выдачи книг, в которых имеется дата возврата и с момента выдачи книги прошло менее 10 дней(некоторая константа, которая означает разрешенный срок хранения книг). Далее вывод группируется по номеру билета формуляра. И с помощью оператора Having убираются все строки таблицы, количество которых менее 10. Для того, чтобы убрать повторяющиеся элемента при выводе, добавляется оператор DISTINCT.

#### Вывод таблицы:

Изображение выглядит как стол

Автоматически созданное описание

### Найти количество читателей, не вернувших в срок книги и имеющих на руках более десяти книг.

SELECT count(\*)  
FROM(SELECT DISTINCT f.number\_ticket, f.reader  
FROM form f  
 JOIN issuance\_books ib on f.number\_ticket = ib.number\_ticket  
WHERE (ib.date\_of\_return is null and ib.date\_of\_issue < *current\_date* - interval '10 day')  
GROUP BY f.number\_ticket  
HAVING *count*(\*) > 10;

) as count;

Данный запрос анологичен предыдущему, только идет подсчет выведенных строк из предыдщего.

#### Вывод таблицы:

Изображение выглядит как стол

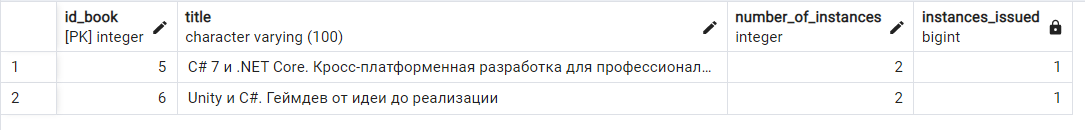
Автоматически созданное описание

### Вывести список книг, которые находятся в библиотеке в единственном экземпляре.

SELECT b.id\_book, b.title, b.number\_of\_instances, t.count as instances\_issued  
FROM book b  
 JOIN (SELECT i.id\_book, *COUNT*(\*) as count  
 FROM instance i  
 JOIN issuance\_books ib ON i.id\_instance = ib.id\_instance  
 WHERE ib.date\_of\_return IS NULL  
 GROUP BY i.id\_book  
) t  
 ON b.id\_book = t.id\_book  
WHERE b.number\_of\_instances - t.count = 1;

Присоединяется таблица, созданная подзапросом, которая хранит id книг и количество взятых экземпляров этих книг. Благодаря WHERE остаются только варинаты, в которых количество экземпляров в библиотеке – количество взятых = 1.

#### Вывод таблицы:



### Подсчитать количество читателей, которые не обращались в библиотеку в течение года.

SELECT *COUNT*(\*)  
FROM form  
WHERE number\_ticket NOT IN (  
 SELECT DISTINCT number\_ticket  
 FROM issuance\_books AS ib  
 WHERE (  
 ib.date\_of\_issue, ib.number\_ticket  
 ) IN (  
 SELECT *MAX*(date\_of\_issue), number\_ticket  
 FROM issuance\_books  
 GROUP BY number\_ticket  
 )  
 AND date\_of\_issue >= *DATE\_TRUNC*('year', *NOW*() - INTERVAL '1 year')  
);

В первом условии WHERE в подзапроссе выбираются только последние запросы формуляров, как максимальную дату обращения, во втором условии выбираются только те даты обращения, которые были совершены более года назад. С помощью функции DATE\_TRUNC() совершается округление до годов

#### Вывод таблцицы:

Изображение выглядит как стол

Автоматически созданное описание

### Подсчитать количество читателей библиотеки по уровню образования.

SELECT *coalesce*(f.degree, 'не указано'), *count*(\*) as count  
FROM form f  
GROUP BY f.degree;

Идет подсчет указанных в системе уровней образований для каждого владельца формуляра. В случае, если уровень образование не указан, то данная строка записывается как ‘не указано’ благодаря функции coalesce.

#### Вывод таблицы:

Изображение выглядит как стол

Автоматически созданное описание

### Вывести список книг по программированию на С#, экземпляры которых отсутствуют в библиотеке, и которые должны быть возвращены не позднее, чем через 3 дня.

SELECT b.id\_book, b.title, b.number\_of\_instances, t.max\_date  
FROM book b  
 JOIN (SELECT i.id\_book, *count*(\*) as count, *MAX*(date\_of\_issue) as max\_date  
 FROM instance i  
 JOIN issuance\_books ib ON i.id\_instance = ib.id\_instance  
 WHERE ib.date\_of\_return IS NULL  
 GROUP BY i.id\_book  
) t  
 ON b.id\_book = t.id\_book  
WHERE b.number\_of\_instances = t.count  
 AND b.area\_of\_knowledge = 'программирование'  
 AND b.title LIKE '%C#%'  
 AND t.max\_date + interval '7 day' < *current\_date*;

С помощью оператора JOIN мы добавляем таблицу, содержащюю количество выданных экземпляров для каждой книги и максимальную дату выдачи для книг, которые не были вовращены. Далее в запросе мы отсеиваем все строки таблицы, в которых количество выданных книг не равно количеству экземпляров в бибилотке и название которых не содержит C#. Дальше мы сравниваем максимальную дату выдачи, к которой прибавляем 7 дней (у нас есть некоторая константа, обозночающая допустимый срок хравнения книг 10 дней) с текущей датой. То есть проверяем на необходимость возврата не позднее чем через 3 дня. Если наиболее позднея книга не удовлетвоярет, то и наиболее ранняя также не будет удовлетворять.

#### Вывод таблицы:

Изображение выглядит как текст

Автоматически созданное описание

## Создание предствлений

### сведения о должниках;

CREATE VIEW debtors as  
SELECT b.id\_book, b.title, f.number\_ticket, f.reader, ib.date\_of\_issue  
FROM form f  
 JOIN issuance\_books ib ON f.number\_ticket = ib.number\_ticket  
 JOIN instance i on i.id\_instance = ib.id\_instance  
 JOIN book b on b.id\_book = i.id\_book  
WHERE ib.date\_of\_return IS NULL AND ib.date\_of\_issue + interval '10 days' < *current\_date*;

Выбираются все книги, экземпляры которых были не возвращены и с момента выдачи которых прошло более 10 дней(некоторая константа, которая хранит срок, на который выдается книга). Выводится информация о книге, экземпляр которой был не возвращен и номер билета с именем читающего, который задолжал.

После выполнения команды SELECT \* FROM debtors мы получим:

Изображение выглядит как текст

Автоматически созданное описание

## 1.2.2. сведения о наиболее популярных книгах (все экземпляры находятся на руках у читателей).

CREATE VIEW popular\_book as  
SELECT b.id\_book, b.title, b.number\_of\_instances, t.count AS collected\_instances  
FROM book b  
 JOIN (SELECT i.id\_book, *count*(\*) as count   
 FROM instance i  
 JOIN issuance\_books ib ON i.id\_instance = ib.id\_instance  
 WHERE ib.date\_of\_return IS NULL  
 GROUP BY i.id\_book  
) t  
 ON b.id\_book = t.id\_book  
WHERE b.number\_of\_instances <= t.count  
GROUP BY b.id\_book, b.title, b.number\_of\_instances, t.count;

Решение схоже с 1.1.7 в подзапросе выбираются id взятых книг и количество взятых экземпляров этих книг. Далее сравнивается количество экземпляров в бибилотеке и количество взятых книг. <= стоит на случай, если будет взята несуществующая книга

После выполнения команды SELECT \* FROM popular\_book мы получим:

Изображение выглядит как текст

Автоматически созданное описание

## Запросы на модификацию данных

INSERT INTO list\_of\_parish(id\_book, number\_document)  
VALUES((SELECT b.id\_book  
 FROM book b  
 WHERE b.author = 'Скит Джон'),  
 (SELECT number\_document  
 FROM parish\_document   
 LIMIT 1));

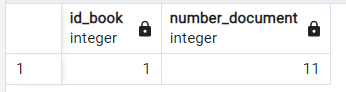
UPDATE list\_of\_parish  
SET id\_book = (SELECT b.id\_book  
 FROM book b  
 WHERE b.author = 'Михаил Булгаков'  
 )  
WHERE id\_book IN (SELECT b.id\_book  
 FROM book b  
 WHERE b.author = 'Скит Джон');  
  
DELETE FROM list\_of\_parish  
 WHERE number\_document = (SELECT number\_document  
 FROM parish\_document  
 WHERE date\_document = '2023-04-30');

Здесь у нас 3 запроса на модификацию данных. Сначала мы добавляем значение в таблицу списка документов на выдачу. В первое поле мы записываем id первой выданной нам книги, автором которой является ‘Скит Джон’, номером документа указываем первый в таблице parish\_document.

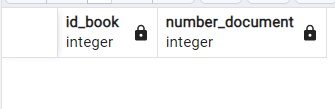
Изображение выглядит как стол

Автоматически созданное описание

Далее мы обновляем книги, автором которых является ‘Скит Джон’ и на их место ставим книгу Михаила Булгакова.



В конце мы удаляем из list\_parish запись о документе от 2023-04-30



## Создание индексов

### Создание простого индекса

Для проверки времени выполнения индексов таблица book была заполнина незначащими строками.

select \* from book  
where author = 'автор1'

Successfully run. Total query runtime: 118 msec. 6126 rows affected.

Изображение выглядит как диаграмма

Автоматически созданное описание

CREATE INDEX book\_author on book(author);

select \* from book  
where author = 'автор1'

Successfully run. Total query runtime: 78 msec. 6126 rows affected.

Изображение выглядит как Прямоугольник

Автоматически созданное описание

### Создание составного индекса

SELECT \* FROM book)  
WHERE author = 'автор1' and title = 'книга2';

Successfully run. Total query runtime: 158 msec. 1021 rows affected.

Изображение выглядит как Прямоугольник

Автоматически созданное описание

CREATE INDEX book\_author\_title on book(author, title);

SELECT \* FROM book)  
WHERE author = 'автор1' and title = 'книга2';

Successfully run. Total query runtime: 130 msec. 1021 rows affected.

Изображение выглядит как Прямоугольник

Автоматически созданное описание

## Вывод

Я успешно выполнила лабораторную работу по овладению практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использованию подзапросов при модификации данных и индексов.

Я создала необходимые запросы и представления на выборку данных, что позволило мне получить нужную информацию из базы данных PostgreSQL. Кроме того, я составила три запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов, что подтверждает моё понимание работы с СУБД PostgreSQL.

Я изучила графическое представление запросов и просмотрела историю запросов, что позволило мне более полно оценить производительность базы данных и улучшить её работу.

Также я создала простой и составной индексы для двух произвольных запросов и сравнила время выполнения запросов без индексов и с индексами, используя команду EXPLAIN. Это позволило мне улучшить производительность запросов и ускорить их выполнение. Было выяснено, что запросы с индексами ускоряют время обращения к базам данных с большим количеством строк.