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: К32392

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

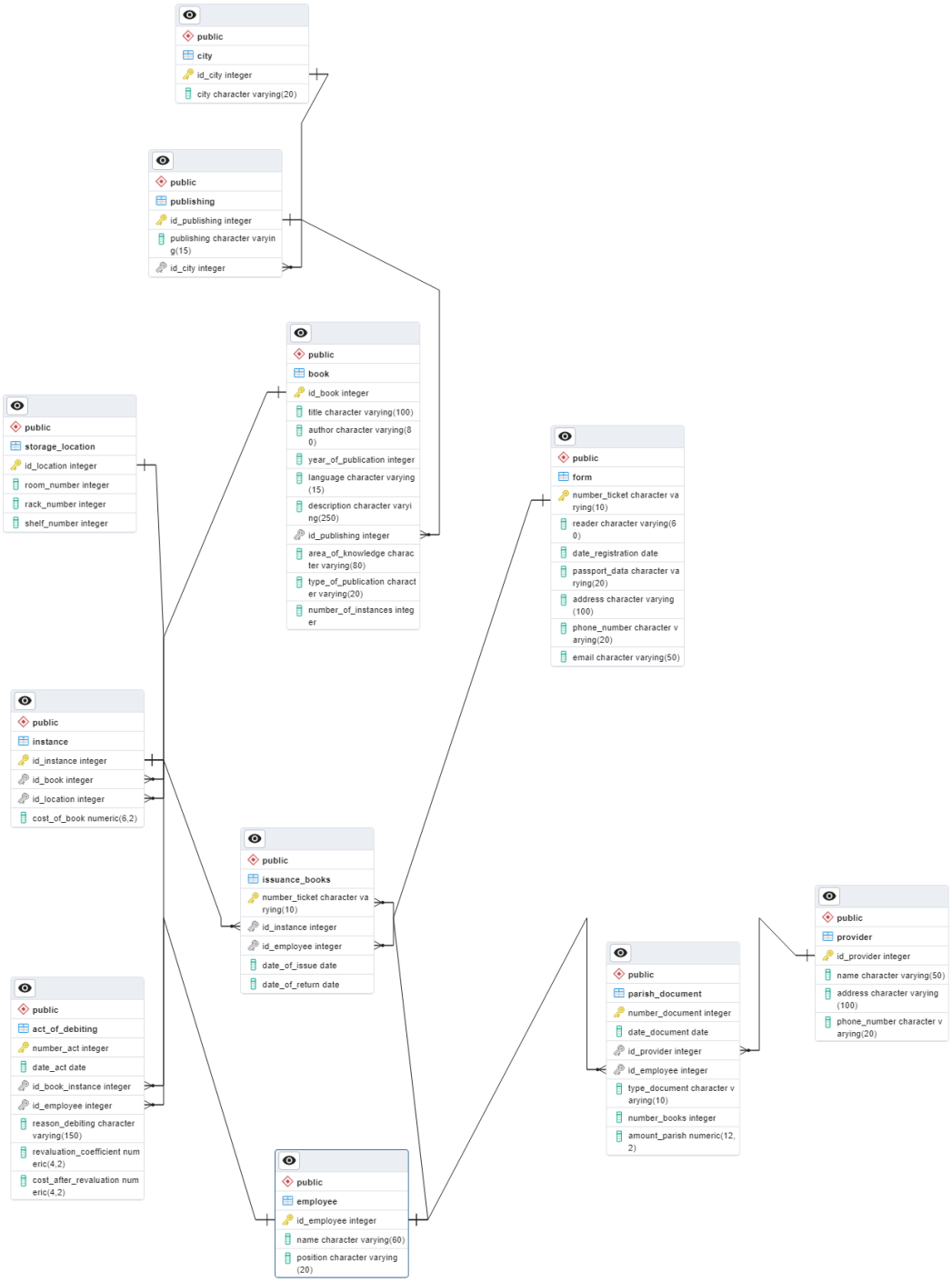
1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).

   1.1. Создайте запросы:
      1.1.1. Вывести список читателей, имеющих на руках книги, переведенные с английского языка, изданные позднее 2000 года.
      1.1.2. Вывести список читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
      1.1.3. Найти количество читателей, не вернувших в срок книги и имеющих на руках более десяти книг.
      1.1.4. Вывести список книг, которые находятся в библиотеке в единственном экземпляре.
      1.1.5. Подсчитать количество читателей, которые не обращались в библиотеку в течение года.
      1.1.6. Подсчитать количество читателей библиотеки по уровню образования.

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

   1.2. Создать представления для администрации библиотеки, содержащие:
      1.2.1. сведения о должниках;
      1.2.2. 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



Выполнение

База Данных: library_bd

Создание запросов к бд

1.1.1. 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:

	number_ticket [PK] character varying (10)	reader character varying (60)
1	1-10	Иванов Дмитрий Дмитриевич
2	1-11	Александра Лидер

1.1.2. 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:

	number_ticket [PK] character varying (10)	reader character varying (60)
1	1-10	Иванов Дмитрий Дмитриевич

1.1.3. 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:

	count bigint
1	1

1.1.4. 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:

	id_book [PK] integer	title character varying (100)	number_of_instances integer	instances_issued bigint
1	5	C# 7 и .NET Core. Кросс-платформенная разработка для профессионал...	2	1
2	6	Unity и C#. Геймдев от идеи до реализации	2	1

1.1.5. 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 таблицы:

	count bigint
1	2

1.1.6. 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:

	coalesce character varying	count bigint
1	высшее	2
2	среднее	1
3	не указано	1

1.1.7. Вывести список книг по программированию на C#, 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:

	id_book [PK] integer	title character varying (100)	max_date date
1	4	C# для чайников	2023-02-10
2	7	C# для профессионалов. Тонкости программирования	2023-02-10

Создание предствлений

1.2.1 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:

	id_book integer	title character varying (100)	id_instance integer	number_ticket character varying (10)	reader character varying (60)	date_of_issue date
1	1	Собачье Сердце	1	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
2	4	C# для чайников	3	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
3	4	C# для чайников	4	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
4	5	C# 7 и .NET Core. Кросс-платформенная разработка для профессионал...	5	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
5	6	Unity и C#. Геймдев от идеи до реализации	7	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
6	7	C# для профессионалов. Тонкости программирования	10	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
7	1	Собачье Сердце	11	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
8	1	Собачье Сердце	12	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
9	2	Севастопольские рассказы	13	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
10	2	Севастопольские рассказы	14	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
11	3	Цветы для Элджернона	15	1-10	Иванов Дмитрий Дмитриевич	2023-02-10
12	2	Севастопольские рассказы	2	1-11	Александра Лидер	2023-02-10
13	7	C# для профессионалов. Тонкости программирования	9	1-11	Александра Лидер	2023-02-10
14	3	Цветы для Элджернона	15	2-10	Иванов Сергей Дмитриевич	2021-02-10

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:

	id_book integer	title character varying (100)	number_of_instances integer	collected_instances bigint
1	1	Собачье Сердце	3	
2	2	Севастопольские рассказы	2	
3	3	Цветы для Элджернона	2	
4	4	C# для чайников	2	
5	7	C# для профессионалов. Тонкости программирования	2	

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.

	id_book integer	number_document integer
1	7	11

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

	id_book integer	number_document integer
1	1	11

В конце мы удаляем из list_parish запись о документе от 2023-04-30

	id_book integer	number_document integer

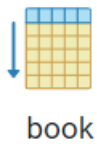
Создание индексов

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.



book

```
CREATE INDEX book_author on book(author);

select * from book
where author = 'автор1'
```

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

book_author book

Создание составного индекса

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

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


```
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.

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.