Министерство образования и науки Российской Федерации
Российский государственный университет нефти и газа
(национальный исследовательский университет)
имени И.М. Губкина

Факультет Автоматики и вычислительной техники
Кафедра Автоматизированных систем управления

Отчёт по домашнему заданию №1

«ПРОДВИНУТАЯ РАБОТА С SQL, РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С СУБД»

по дисциплине *Базы данных*

Выполнил:
студент группы АС-23-04
Ханевский Ярослав

Проверили:
доцент кафедры АСУ, к.т.н. Волков Д.А.
ст. преп. кафедры АСУ Мухина А.Г.

Москва, 2025 г.

Ход работы:

1. Для своих таблиц выполнить несколько запросов на группировку данных, используя GROUP_CONCAT(в Postgres STRING_AGG) с разными разделителями.

Таблицы, с которыми мы будем работать:

```
postgres=# select * from library_books order by book_id;
 book_id |          title           |      author       | pages | is_available | genre_id
---------+--------------------------+-------------------+-------+--------------+----------
       1 | Crime and punishment     | Fyodor Dostoevsky |   672 | t            |        1
       2 | War and Peace            | Leo Tolstoy       |  1225 | f            |        1
       3 | A dogs heart             | Mikhail Bulgakov  |   176 | f            |        2
       4 | The Master and Margarita | Mikhail Bulgakov  |   384 | t            |        3
       5 | Anna Karenina            | Leo Tolstoy       |   864 | t            |        1
       6 | 1984                     | George Orwell     |   328 | f            |        4
```

Рисунок 1. Таблица с книгами

```
postgres=# select * from library_books_genres;
 genre_id |    genre
----------+-------------
        1 | Classic
        2 | Satire
        3 | Fantasy
        4 | Anti-utopia
```

Рисунок 2. Таблица с жанрами книг

Используя агрегирующую функцию STRING_AGG, сгруппируем книги по авторам с объединением названий книг через точку с запятой:

```
postgres=# SELECT author, STRING_AGG(title, ';'),
postgres-# COUNT(*) FROM library_books GROUP BY author;
```

Рисунок 3. Использование STRING_AGG

В результате получим следующую таблицу:

```
      author       |              string_agg               | count
-------------------+---------------------------------------+-------
 George Orwell     | 1984                                  |     1
 Leo Tolstoy       | War and Peace;Anna Karenina           |     2
 Fyodor Dostoevsky | Crime and punishment                  |     1
 Mikhail Bulgakov  | A dogs heart;The Master and Margarita |     2
```

Рисунок 4. Результат вывода с STRING_AGG

Теперь сгруппируем книги по id жанров, применив INNER JOIN для того, чтобы взять книги из таблицы library_books и вывести их с жанрами из таблицы library_books_genres. Также изменим разделитель на " -- " и отсортируем вывод по возрастанию id жанров:

```
postgres=# select library_books_genres.genre_id,
postgres-# library_books_genres.genre,
postgres-# string_agg(library_books.title, ' -- ')
postgres-# from library_books_genres inner join library_books on library_books_genres.genre_id=library_books.genre_id
postgres-# group by library_books_genres.genre_id
postgres-# order by library_books_genres.genre_id;
```

Рисунок 5. Группировка и смена разделителя

В этом случае таблица будет выглядеть следующим образом:

```
 genre_id |    genre    |                        string_agg
----------+-------------+-----------------------------------------------------
        1 | Classic     | Anna Karenina -- War and Peace -- Crime and punishment
        2 | Satire      | A dogs heart
        3 | Fantasy     | The Master and Margarita
        4 | Anti-utopia | 1984
```

Рисунок 6. Отсортированная по жанрам таблица с разделителем '' -- ''

2. Для своих таблиц выполнить несколько запросов, содержащих подзапросы в разных операторах (WHERE, SELECT…).

Выполним запрос, содержащий подзапрос для поиска книг определенного жанра:

```
postgres=# SELECT title, author, pages
postgres-# FROM library_books
postgres-# WHERE genre_id IN (
postgres(# SELECT genre_id
postgres(# FROM library_books_genres
postgres(# WHERE genre = 'Classic');
```

Рисунок 7. Запрос с подзапросом для поиска книг определенного жанра

Получим результат:

```
        title          |      author       | pages
-----------------------+-------------------+-------
 Crime and punishment  | Fyodor Dostoevsky |   672
 War and Peace         | Leo Tolstoy       |  1225
 Anna Karenina         | Leo Tolstoy       |   864
```

Рисунок 8. Результат запроса с подзапросом (1)

Следующий запрос, выводящий статистику по жанрам:

```
postgres=# SELECT g.genre, book_stats.avg_pages, book_stats.book_count
postgres-# FROM library_books_genres g,
postgres-# (SELECT genre_id,
postgres(# AVG(pages) AS avg_pages,
postgres(# COUNT(*) AS book_count
postgres(# FROM library_books GROUP BY genre_id) book_stats
postgres-# WHERE g.genre_id = book_stats.genre_id
postgres-# ORDER BY book_stats.avg_pages DESC;
```

Рисунок 9. Запрос с подзапросом, выводящий нужные данные

Результат данного запроса:

```
    genre     |      avg_pages        | book_count
--------------+-----------------------+------------
 Classic      | 920.3333333333333333  |          3
 Fantasy      | 384.0000000000000000  |          1
 Anti-utopia  | 328.0000000000000000  |          1
 Satire       | 176.0000000000000000  |          1
```

Рисунок 10. Результат запроса с подзапросом (2)

Запрос с подзапросом, отображающий самый популярный жанр и книги этого жанра через разделитель:

```
postgres=# SELECT g.genre AS most_popular_genre,
postgres-# (SELECT STRING_AGG(title, '; ')
postgres(# FROM library_books WHERE genre_id = g.genre_id)
postgres-# AS books_list FROM library_books_genres g
postgres-# ORDER BY (SELECT COUNT(*) FROM library_books) DESC
postgres-# LIMIT 1;
```

Рисунок 11. Запрос, отображающий самый популярный жанр и книги этого жанра

Получаем следующее:

```
 most_popular_genre |                        books_list
--------------------+-----------------------------------------------------
 Classic            | Crime and punishment; War and Peace; Anna Karenina
```

Рисунок 12. Результат запроса с подзапросом (3)

3. Создать несколько транзакций, изменить данные, часть завершить успешно, часть откатить.

Создадим транзакцию с произвольными действиями и завершим её:

```
postgres=# BEGIN;
BEGIN
postgres=*# INSERT INTO library_books (title, author, pages, is_available, genre_id)
postgres-*# VALUES ('Idiot', 'Fyodor Dostoevsky', 640, TRUE, 1);
INSERT 0 1
postgres=*# UPDATE library_books
postgres-*# SET is_available = FALSE
postgres-*# WHERE title = 'The Master and Margarita';
UPDATE 1
postgres=*# SELECT * FROM library_books;
 book_id |          title           |      author       | pages | is_available | genre_id
---------+--------------------------+-------------------+-------+--------------+----------
       1 | Crime and punishment     | Fyodor Dostoevsky |   672 | t            |        1
       2 | War and Peace            | Leo Tolstoy       |  1225 | f            |        1
       5 | Anna Karenina            | Leo Tolstoy       |   864 | t            |        1
       3 | A dogs heart             | Mikhail Bulgakov  |   176 | f            |        2
       6 | 1984                     | George Orwell     |   328 | f            |        4
       7 | Idiot                    | Fyodor Dostoevsky |   640 | t            |        1
       4 | The Master and Margarita | Mikhail Bulgakov  |   384 | f            |        3
(7 rows)


postgres=*# SAVEPOINT p1;
SAVEPOINT
postgres=*# DELETE FROM library_books
postgres-*# WHERE title = '1984';
DELETE 1
postgres=*# COMMIT;
COMMIT
postgres=# SELECT * FROM library_books;
 book_id |          title           |      author       | pages | is_available | genre_id
---------+--------------------------+-------------------+-------+--------------+----------
       1 | Crime and punishment     | Fyodor Dostoevsky |   672 | t            |        1
       2 | War and Peace            | Leo Tolstoy       |  1225 | f            |        1
       5 | Anna Karenina            | Leo Tolstoy       |   864 | t            |        1
       3 | A dogs heart             | Mikhail Bulgakov  |   176 | f            |        2
       7 | Idiot                    | Fyodor Dostoevsky |   640 | t            |        1
       4 | The Master and Margarita | Mikhail Bulgakov  |   384 | f            |        3
```

Рисунок 13. Успешная транзакция

Транзакция завершена успешно, данные обновлены.

Осуществим следующую транзакцию с внедрением точек сохранения SAVEPOINT и отката к ним ROLLBACK TO и ROLLBACK:

```
postgres=# BEGIN;
BEGIN
postgres=*# ALTER TABLE library_books DROP COLUMN is_available;
ALTER TABLE
postgres=*# SAVEPOINT p1;
SAVEPOINT
postgres=*# UPDATE library_books SET genre_id=10;
UPDATE 6
postgres=*# ROLLBACK TO p1;
ROLLBACK
postgres=*# SELECT * FROM library_books;
 book_id |          title           |      author       | pages | genre_id
---------+--------------------------+-------------------+-------+----------
       1 | Crime and punishment     | Fyodor Dostoevsky |   672 |        1
       2 | War and Peace            | Leo Tolstoy       |  1225 |        1
       5 | Anna Karenina            | Leo Tolstoy       |   864 |        1
       3 | A dogs heart             | Mikhail Bulgakov  |   176 |        2
       7 | Idiot                    | Fyodor Dostoevsky |   640 |        1
       4 | The Master and Margarita | Mikhail Bulgakov  |   384 |        3
(6 rows)


postgres=*# ROLLBACK;
ROLLBACK
postgres=# SELECT * FROM library_books;
 book_id |          title           |      author       | pages | is_available | genre_id
---------+--------------------------+-------------------+-------+--------------+----------
       1 | Crime and punishment     | Fyodor Dostoevsky |   672 | t            |        1
       2 | War and Peace            | Leo Tolstoy       |  1225 | f            |        1
       5 | Anna Karenina            | Leo Tolstoy       |   864 | t            |        1
       3 | A dogs heart             | Mikhail Bulgakov  |   176 | f            |        2
       7 | Idiot                    | Fyodor Dostoevsky |   640 | t            |        1
       4 | The Master and Margarita | Mikhail Bulgakov  |   384 | f            |        3
```

Рисунок 14. Транзакция с точками сохранения и откатом к ним

В этой транзакции сначала происходит обновление данных по заданному требованию, а затем таблица приводится к первоначальному виду с помощью отката к точке сохранения.

4. Создать несколько функций, триггеров.

В PostgreSQL процедуры – это набор SQL-инструкций, которые объединены вместе для выполнения определенной задачи или операции. Они могут принимать параметры в качестве входных данных и возвращать значения в качестве выходных данных.

Создадим процедуру с помощью CREATE PROCEDURE, предназначенную для добавления новых книг в библиотечную систему с автоматической проверкой и созданием жанров, если они не существуют:

```
postgres=# CREATE PROCEDURE add_new_book(
postgres(# p_title VARCHAR(200),
postgres(# p_author VARCHAR(100),
postgres(# p_pages INT,
postgres(# p_is_available BOOLEAN,
postgres(# p_genre VARCHAR(100))
postgres-# LANGUAGE plpgsql
postgres-# AS $$
postgres$# DECLARE v_genre_id INT;
postgres$# BEGIN SELECT genre_id INTO v_genre_id
postgres$# FROM library_books_genres WHERE genre = p_genre;
postgres$# IF NOT FOUND THEN INSERT INTO library_books_genres (genre)
postgres$# VALUES (p_genre) RETURNING genre_id INTO v_genre_id;
postgres$# END IF;
postgres$# INSERT INTO library_books (title, author, pages, is_available, genre_id)
postgres$# VALUES (p_title, p_author, p_pages, p_is_available, v_genre_id);
postgres$# COMMIT;
postgres$# END;
postgres$# $$;
CREATE PROCEDURE
```

Рисунок 15. Создание процедуры

Вызовем процедуру через CALL:

```
postgres=# CALL add_new_book('New book', 'Autor', 300, TRUE, 'SOME NEW GENRE');
CALL
postgres=# SELECT * FROM library_books WHERE title='New book';
 book_id |  title   | author | pages | is_available | genre_id
---------+----------+--------+-------+--------------+----------
       8 | New book | Autor  |   300 | t            |        5
(1 row)


postgres=# SELECT * FROM library_books_genres WHERE genre_id=5;
 genre_id |     genre
----------+---------------
        5 | SOME NEW GENRE
```

Рисунок 16. Результат работы процедуры

Триггеры в PostgreSQL представляют собой специальные хранимые процедуры, которые выполняются автоматически при определенных событиях, таких как вставка, обновление или удаление данных в таблице. Они позволяют реагировать на изменения данных в реальном времени и выполнять определенные действия в ответ на эти изменения. Триггеры часто используются для обеспечения целостности данных, аудита изменений, автоматизации сложных бизнес-логик и соблюдения бизнес-правил.

Создадим триггер, обновляющий время последнего редактирования записи о книге:

```
postgres=# ALTER TABLE library_books
postgres-# ADD COLUMN last_updated TIMESTAMP;
ALTER TABLE
postgres=# CREATE FUNCTION update_book_timestamp()
postgres-# RETURNS TRIGGER
postgres-# LANGUAGE plpgsql
postgres-# AS $$
postgres$# BEGIN
postgres$# NEW.last_updated = NOW();
postgres$# RETURN NEW;
postgres$# END;
postgres$# $$;
CREATE FUNCTION
postgres=# CREATE TRIGGER trg_update_book_timestamp
postgres-# BEFORE UPDATE ON library_books
postgres-# FOR EACH ROW
postgres-# EXECUTE FUNCTION update_book_timestamp();
```

Рисунок 17. Создание триггера

Проведем проверку, обновим некоторые данные:

```
postgres=# SELECT book_id, title, last_updated FROM library_books WHERE book_id = 1;
 book_id |        title         | last_updated
---------+----------------------+--------------
       1 | Crime and punishment |
(1 row)


postgres=# UPDATE library_books SET pages = 680 WHERE book_id = 1;
UPDATE 1
```

Рисунок 18. Обновление данных в таблице

```
postgres=# SELECT book_id, title, last_updated FROM library_books WHERE book_id = 1;
 book_id |        title         |        last_updated
---------+----------------------+----------------------------
       1 | Crime and punishment | 2025-04-23 02:17:18.546017
```

Рисунок 19. Результат работы триггера

5. Выполнить несколько запросов с оконными функциями.

Оконные функции позволяют добавить в строку значения аггрегирующих функций для группы строк в контексте окна.

Запрос, осуществляющий вывод всех строк таблицы с суммарным числом страниц в книгах:

```
postgres=# ALTER TABLE library_books
postgres-# DROP COLUMN last_updated;
ALTER TABLE
postgres=# SELECT book_id, title,
postgres-# author, pages,
postgres-# SUM(pages) OVER ()
postgres-# FROM library_books
postgres-# ORDER BY pages DESC;
```

Рисунок 20. Запрос с оконной функцией SUM

Получим следующий результат:

book_id	title	author	pages	sum
2	War and Peace	Leo Tolstoy	1225	4269
5	Anna Karenina	Leo Tolstoy	864	4269
1	Crime and punishment	Fyodor Dostoevsky	680	4269
7	Idiot	Fyodor Dostoevsky	640	4269
4	The Master and Margarita	Mikhail Bulgakov	384	4269
8	New book	Autor	300	4269
3	A dogs heart	Mikhail Bulgakov	176	4269

Рисунок 21. Результат работы запроса с оконной функцией SUM

Функция ранжирования RANK() назначает одинаковый ранг всем строкам с одинаковыми значениями.

Реализуем запрос с оконной агрегатной функцией RANK(), ранжирующий книги по убыванию количества страниц:

```
postgres=# SELECT title, author,
postgres-# pages, RANK() OVER (ORDER BY pages DESC)
postgres-# FROM library_books ORDER BY rank DESC;
```

Рисунок 22. Запрос с оконной функцией RANK

title	author	pages	rank
A dogs heart	Mikhail Bulgakov	176	7
New book	Autor	300	6
The Master and Margarita	Mikhail Bulgakov	384	5
Idiot	Fyodor Dostoevsky	640	4
Crime and punishment	Fyodor Dostoevsky	680	3
Anna Karenina	Leo Tolstoy	864	2
War and Peace	Leo Tolstoy	1225	1

(7 rows)

Рисунок 23. Результат работы запроса с оконной функцией RANK

Напишем запрос с функциями LAG() и LEAD:

```
postgres=# SELECT title, author,
postgres-# pages, LAG(title) OVER () AS prev_title,
postgres-# LEAD(pages) OVER () AS prev_pages
postgres-# FROM library_books
postgres-# ;
```

Рисунок 24. Запрос с оконными функциями LAG и LEAD

title	author	pages	prev_title	prev_pages
War and Peace	Leo Tolstoy	1225		864
Anna Karenina	Leo Tolstoy	864	War and Peace	176
A dogs heart	Mikhail Bulgakov	176	Anna Karenina	640
Idiot	Fyodor Dostoevsky	640	A dogs heart	384
The Master and Margarita	Mikhail Bulgakov	384	Idiot	300
New book	Autor	300	The Master and Margarita	680
Crime and punishment	Fyodor Dostoevsky	680	New book	

Рисунок 25. Результат работы запроса

6. Реализовать приложение.

CRUD — это аббревиатура, обозначающая четыре основных операции управления данными: создание, чтение, обновление и удаление.

Для создания программы используем python, PyQT и psycopg2.

Скриншоты программы:

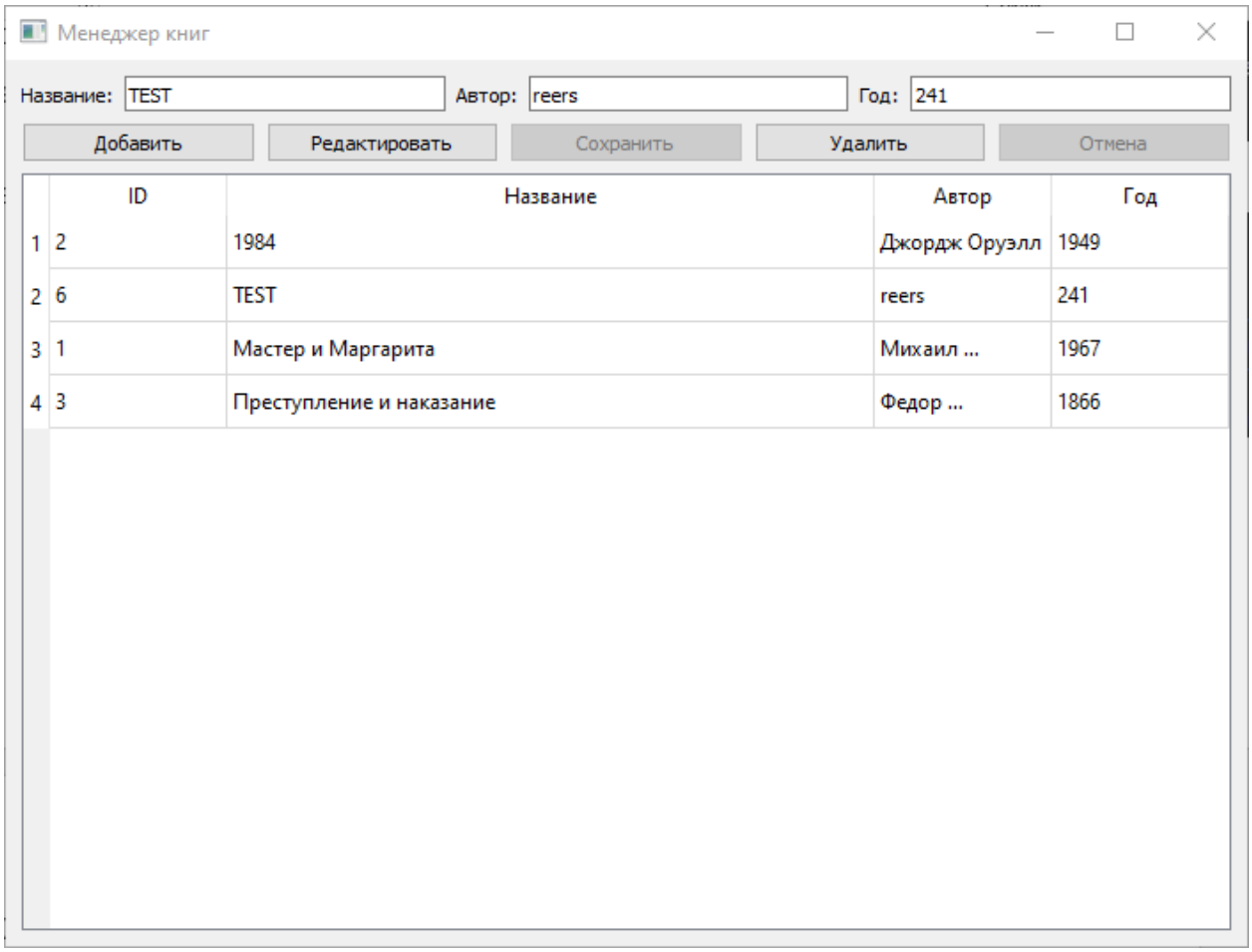

Рисунок 26. Окно программы

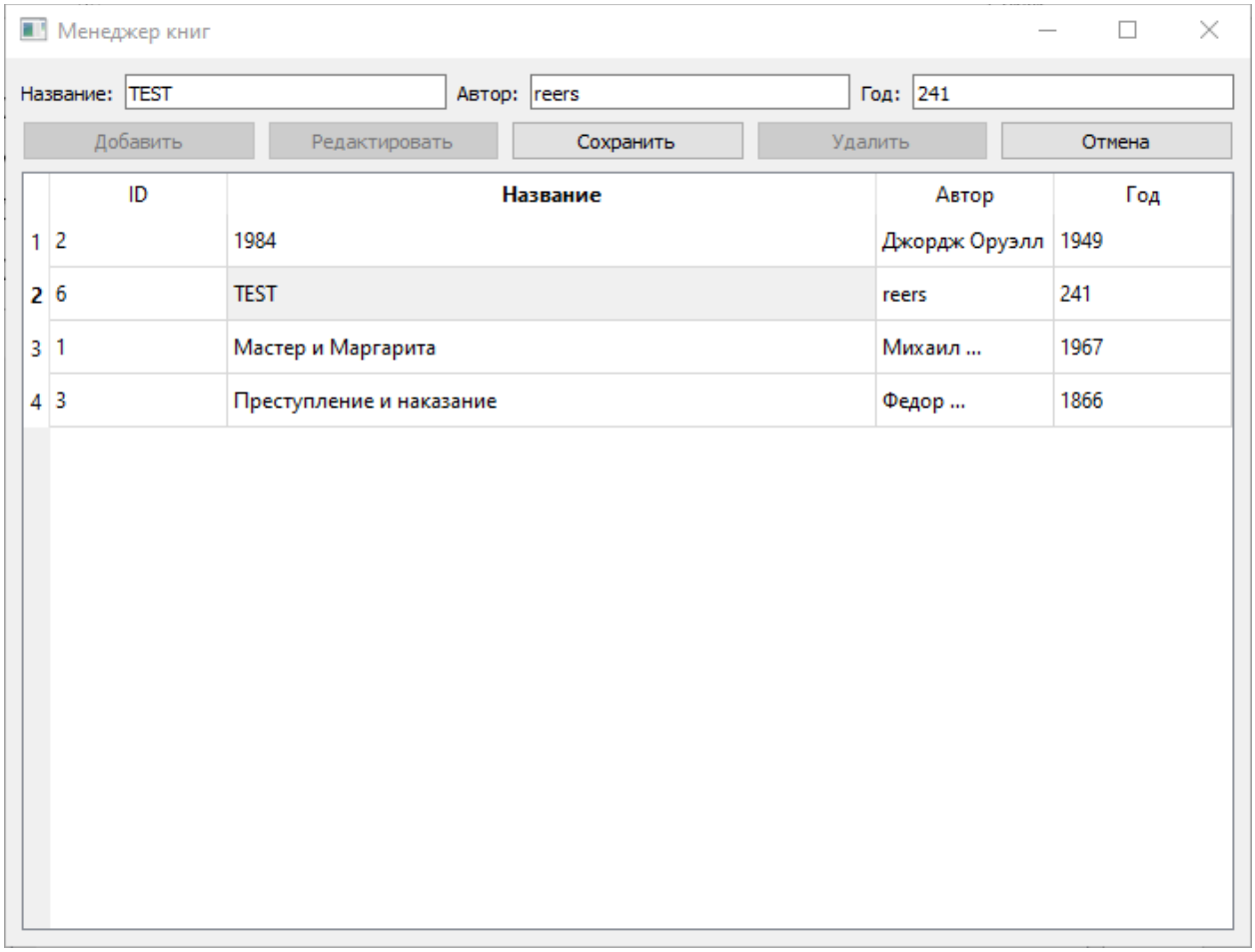

Рисунок 27. Редактирование строки

```
postgres=# SELECT * FROM simple_books;
 id |          title           |      author       | year
----+--------------------------+-------------------+------
  1 | Мастер и Маргарита       | Михаил Булгаков   | 1967
  2 | 1984                     | Джордж Оруэлл     | 1949
  3 | Преступление и наказание | Федор Достоевский | 1866
  6 | TEST                     | reers             |  241
```

Рисунок 28. Вывод БД в консоли