

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

**«Запросы на выборку и модификацию данных. Представления. Работа с  
индексами»**  
**по дисциплине «Проектирование и реализация баз данных»**

**Обучающийся Крамарь Кирилл Александрович**

**Факультет прикладной информатики**

**Группа К3239**

**Направление подготовки 09.03.03 Прикладная информатика**

**Образовательная программа Мобильные и сетевые технологии**

**2023 Преподаватель Говорова Марина Михайловна**

**Санкт-Петербург**

**2024/2025**

# СОДЕРЖАНИЕ

Стр.

<b>1 Цель работы .....</b>	<b>3</b>
<b>2 Задание практической работы .....</b>	<b>4</b>
<b>3 ERD Диаграмма базы данных.....</b>	<b>5</b>
<b>4 Выполнение ЛР №4.....</b>	<b>6</b>
<b>4.1 Запросы к базе данных .....</b>	<b>6</b>
<b>4.2 Представления .....</b>	<b>12</b>
<b>4.3 Запросы на модификацию данных .....</b>	<b>14</b>
<b>4.4 Индексы .....</b>	<b>17</b>
<b>Выводы .....</b>	<b>20</b>

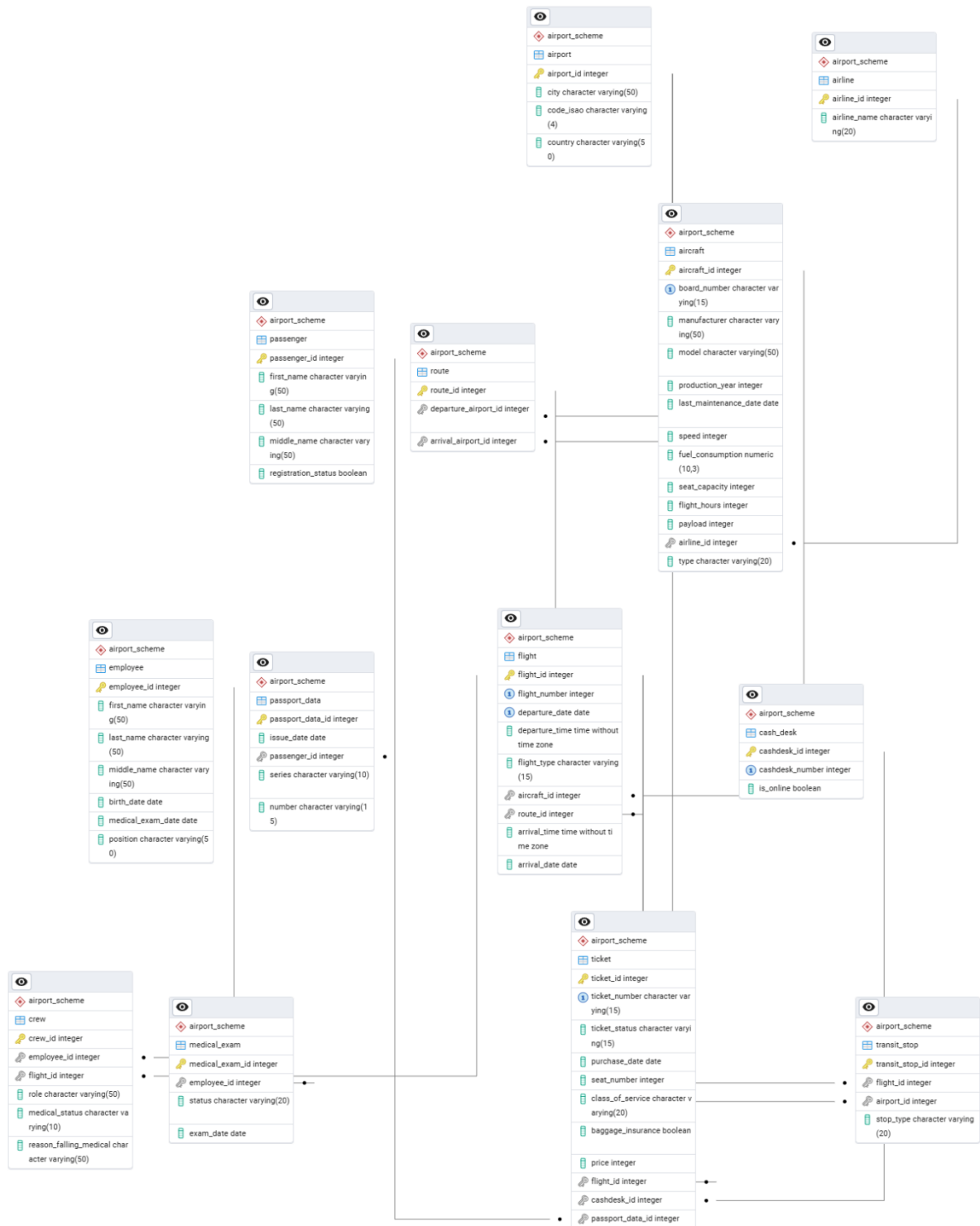
## **1. Цель работы**

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

## 2. Задание практической работы

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).
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.

### 3 ERD Диаграмма базы данных



## 4 Выполнение

### 4.1 Запросы к базе данных

Запрос №1: Определить расчетное время полета по всем маршрутам.

#### SQL код запроса:

---

```
SELECT route_id, EXTRACT(  
EPOCH FROM ((CAST(arrival_date AS timestamp) + arrival_time) -  
(CAST(departure_date AS timestamp) + departure_time))) / 3600 AS  
flight_duration_hours,  
(EXTRACT(EPOCH FROM ((CAST(arrival_date AS timestamp) + arrival_time) -  
(CAST(departure_date AS timestamp) + departure_time) )) % 3600)  
/ 60 AS flight_duration_minutes  
FROM airport_scheme.flight  
ORDER BY  
route_id, flight_number;  
---
```

Что происходит в данном запросе?

Данный SQL-запрос вычисляет продолжительность полета для каждого рейса в часах и минутах, используя время вылета и прибытия. Для этого сначала формируются полные временные метки вылета и прибытия: функция CAST преобразует дату (departure\_date и arrival\_date) в тип timestamp, чтобы добавить к ней время (departure\_time и arrival\_time). Например, дата «2024-06-01» и время «08:00:00» объединяются в «2024-06-01 08:00:00». Разница между меткой прибытия и вылета вычисляется через вычитание, что дает интервал времени. Функция EXTRACT с параметром EPOCH переводит этот интервал в секунды. Далее секунды преобразуются в часы делением на 3600 (flight\_duration\_hours), а оставшиеся секунды (через оператор %) делятся на 60 для получения минут (flight\_duration\_minutes). Например, 9000 секунд (2.5 часа) превратятся в 2.5 часа и 30 минут. Результаты сортируются по идентификатору маршрута (route\_id) и номеру рейса (flight\_number) для удобства анализа.

	route_id integer	flight_duration_hours numeric	flight_duration_minutes numeric
1	1	2.0000000000000000	0.0000000000000000
2	2	3.0000000000000000	0.0000000000000000
3	3	3.0000000000000000	0.0000000000000000
4	4	2.0000000000000000	0.0000000000000000
5	5	5.0000000000000000	0.0000000000000000
6	6	4.0000000000000000	0.0000000000000000
7	7	2.0000000000000000	0.0000000000000000
Total rows: 50		Query complete 00:00:00.037	

Запрос №2: Определить расход топлива по всем маршрутам.

---

```
SELECT route.route_id,  
SUM(aircraft.fuel_consumption * (EXTRACT(EPOCH FROM  
(CAST(flight.arrival_date AS timestamp) + flight.arrival_time) -  
CAST(flight.departure_date AS timestamp) - flight.departure_time) / 3600)) AS  
fuel_consumed_tonnes  
FROM airport_scheme.flight  
JOIN airport_scheme.route ON flight.route_id = route.route_id  
JOIN airport_scheme.aircraft ON flight.aircraft_id = aircraft.aircraft_id  
GROUP BY route.route_id  
---
```

Что происходит в данном запросе?

Данный SQL-запрос рассчитывает суммарный расход топлива (в тоннах) для каждого маршрута на основе данных о рейсах, самолетах и маршрутах. Запрос соединяет три таблицы: flight (рейсы), route (маршруты) и aircraft (самолеты) через JOIN. Для каждого рейса вычисляется длительность полета в часах: разница между временем прибытия и вылета переводится в секунды (EXTRACT(EPOCH)), затем в часы (/3600). Расход топлива определяется умножением часового расхода самолета (fuel\_consumption) на длительность рейса. Результаты группируются по route\_id (GROUP BY), чтобы получить общий расход для каждого маршрута. Ограничения: не учитываются рейсы с пересечением полуночи (например, вылет в 23:00, прибытие в 01:00) и NULL-значения в датах/времени. Пример вывода: для маршрута 1 общий расход — 150.25 тонн. Запрос используется для анализа затрат на топливо и оптимизации маршрутов.

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	route_id [PK] integer	fuel_consumed_tonnes numeric
1	22	7.000000000000000000
2	42	11.400000000000000000
3	40	11.400000000000000000
4	43	6.200000000000000000
5	19	5.400000000000000000
6	29	27.000000000000000000
7	4	7.000000000000000000
Total rows: 50		Query complete 00:00:00.042

Запрос №3: Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по заданному рейсу за вчерашний день.

---

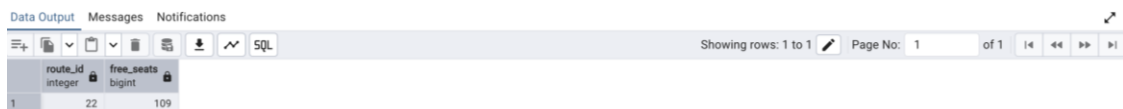
```
SELECT flight.route_id, aircraft.seat_capacity - COALESCE(COUNT(ticket.ticket_id),
```

```

0) AS free_seats
FROM airport_scheme.flight
JOIN airport_scheme.aircraft ON flight.aircraft_id = aircraft.aircraft_id
LEFT JOIN airport_scheme.ticket ON flight.flight_id = ticket.flight_id
WHERE flight.departure_date = CURRENT_DATE - 1
GROUP BY flight.route_id, aircraft.seat_capacity;
---
```

Что происходит в данном запросе?

Данный SQL-запрос вычисляет количество свободных мест на рейсах, вылетевших вчера. Для этого он соединяет таблицы рейсов (flight), самолетов (aircraft) и билетов (ticket). Вместимость самолета (seat\_capacity) уменьшается на количество проданных билетов (COUNT(ticket.ticket\_id)), а LEFT JOIN гарантирует учет рейсов без билетов. WHERE фильтрует рейсы по вчерашней дате вылета. GROUP BY группирует результат по маршруту и вместимости самолета. COALESCE заменяет NULL (если билетов нет) на 0. Ограничения: не учитывает отмененные билеты и групповые брони.



route_id	free_seats
1	22

Запрос №4: Рассчитать убытки компании за счет непроданных билетов за вчерашний день.

```

SELECT flight.route_id, ticket.price * (aircraft.seat_capacity -
COALESCE(COUNT(ticket.ticket_id), 0)) AS fail_sum
FROM airport_scheme.flight flight
JOIN airport_scheme.aircraft aircraft ON flight.aircraft_id = aircraft.aircraft_id
LEFT JOIN airport_scheme.ticket ticket ON flight.flight_id = ticket.flight_id
WHERE flight.departure_date = CURRENT_DATE - 1
GROUP BY flight.route_id, ticket.price, aircraft.seat_capacity;
---
```

Что происходит в данном запросе?

Данный SQL-запрос вычисляет потенциальные убытки авиакомпании за вчерашние рейсы из-за непроданных билетов. Основа: Соединяет таблицы рейсов (flight), самолетов (aircraft) и билетов (ticket). Формула: цена билета \* (вместимость самолета - проданные билеты). LEFT JOIN включает рейсы без билетов, COALESCE заменяет NULL на 0. Фильтр: departure\_date = вчера. Группировка по маршруту (route\_id), цене билета и вместимости самолета. Ограничения: предполагает, что цена билета на маршруте фиксированная.



Data Output Messages Notifications		
		SQL
route_id integer	fail_sum bigint	
1	22	3052000

Запрос №5: Определить, какой тип самолетов чаще всего летал в заданный аэропорт назначения.

```

SELECT airport.code_isao, aircraft.type
FROM airport_scheme.aircraft
JOIN airport_scheme.flight ON flight.aircraft_id = aircraft.aircraft_id
JOIN airport_scheme.route ON route.route_id = flight.route_id
JOIN airport_scheme.airport ON route.departure_airport_id = airport.airport_id
GROUP BY airport.code_isao, aircraft.type
HAVING COUNT(flight.flight_id) = (SELECT MAX(frequency) FROM (SELECT
COUNT(flight.flight_id) AS frequency
FROM airport_scheme.flight JOIN airport_scheme.route ON route.route_id =
flight.route_id JOIN airport_scheme.airport ON route.departure_airport_id
= airport.airport_id WHERE route.departure_airport_id = airport.airport_id GROUP BY
flight.aircraft_id))

```

Что происходит в данном запросе?

Данный SQL-запрос определяет типы самолетов, которые чаще всего использовались для вылетов из каждого аэропорта.

1. Соединяет таблицы: самолеты (aircraft), рейсы (flight), маршруты (route), аэропорты (airport).
2. Группирует данные по коду аэропорта (code\_isao) и типу самолета (type).
3. Фильтрует группы через HAVING, оставляя только те, где количество рейсов равно максимальному для аэропорта (подзапрос вычисляет это значение).
4. Подзапрос внутри HAVING:
  - Считает количество рейсов для каждого самолета (flight.aircraft\_id) в аэропорте отправления (route.departure\_airport\_id).
  - Находит максимальное значение этого количества (MAX(frequency)).

Data Output Messages Notifications		
<div> <div>☰+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>		
	code_isao character varying (4) 🔒	type character varying (20) 🔒
1	KJA	пассажирский
2	PEE	пассажирский
3	SIP	пассажирский
4	DYR	пассажирский
5	YKS	пассажирский
6	UUD	пассажирский
7	VOG	пассажирский

Запрос №6: Вывести список самолетов, “возраст” которых превышает средний “возраст” самолетов этого типа.

---

```
SELECT aircraft.aircraft_id From airport_scheme.aircraft
WHERE aircraft.production_year > (select AVG(aircraft.production_year) from
airport_scheme.aircraft)
```

---

Что происходит в данном запросе?

Данный SQL-запрос выбирает идентификаторы самолетов, год производства которых новее среднего года производства всех самолетов в базе. Коротко:

1. Подзапрос вычисляет средний год производства самолетов (AVG(production\_year)).
2. Основной запрос фильтрует самолеты, у которых production\_year больше этого среднего.

Пример вывода: самолеты с aircraft\_id 5, 10, 15, если их год выпуска новее среднего (например, средний год — 2015, а их — 2020).

Ограничения: не учитывает самолеты с NULL в production\_year.

	aircraft_id [PK] integer
1	1
2	2
3	3
4	4
5	5
6	9
7	10

Запрос №7: Определить тип самолетов, летающих во все аэропорты назначения.

---

```

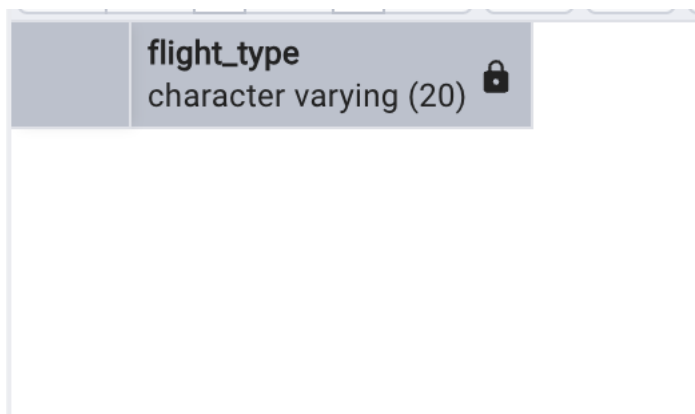
WITH airport_counts AS (SELECT aircraft.type AS flight_type, COUNT(DISTINCT
airport.code_isao) AS airports_count
FROM airport_scheme.aircraft JOIN airport_scheme.flight ON flight.aircraft_id =
aircraft.aircraft_id
JOIN airport_scheme.route ON route.route_id = flight.route_id JOIN
airport_scheme.airport ON route.arrival_airport_id = airport.airport_id
GROUP BY aircraft.type) SELECT flight_type FROM airport_counts WHERE
airports_count = (SELECT COUNT(DISTINCT code_isao) FROM
airport_scheme.airport)
---
```

Что происходит в данном запросе?

Данный SQL-запрос определяет типы самолетов, которые летали во все аэропорты назначения в базе. Коротко:

1. СТЕ (airport\_counts): для каждого типа самолета (aircraft.type) считает количество уникальных аэропортов назначения (COUNT(DISTINCT airport.code\_isao), куда он летал.
2. Основной запрос: выбирает типы самолетов, у которых это количество равно общему числу аэропортов в базе (подзапрос SELECT COUNT(DISTINCT code\_isao)).

Ограничения: не учитывает аэропорты без рейсов.



## 4.2 Представления

Представление №1: Для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю  
---

```

CREATE OR REPLACE VIEW view_moscow AS
SELECT flight.flight_number,
flight.departure_time,flight.arrival_time,flight.departure_date FROM
airport_scheme.flight
JOIN airport_scheme.route ON route.route_id = flight.flight_id
JOIN airport_scheme.airport ON airport.airport_id = route.arrival_airport_id
JOIN airport_scheme.aircraft ON flight.aircraft_id = aircraft.aircraft_id
WHERE airport.city = 'Москва' AND aircraft.type != 'грузовой' AND
```

```
(flight.departure_date BETWEEN CURRENT_DATE AND CURRENT_DATE +
INTERVAL '7 days')
---
```

Что происходит в данном запросе?

Данный SQL-запрос создает/обновляет представление view\_moscow, которое отображает пассажирские рейсы в Москву на ближайшие 7 дней. Коротко:

1. Выбирает номер рейса, время вылета/прибытия, дату вылета.
2. Соединяет таблицы рейсов (flight), маршрутов (route), аэропортов (airport) и самолетов (aircraft).
3. Фильтрует:
  - Аэропорт назначения — Москва (city = 'Москва').
  - Самолеты — не грузовые (type != 'грузовой').
  - Дата вылета — ближайшая неделя (BETWEEN CURRENT\_DATE AND CURRENT\_DATE + 7 дней).

CREATE VIEW

Query returned successfully in 81 msec.

	flight_number integer	departure_time time without time zone	arrival_time time without time zone	departure_date date
1	3700	14:30:00	16:30:00	2025-05-26

Представление №2: Количество самолетов каждого типа, летавшими за последний месяц.

```
CREATE OR REPLACE VIEW type_flight_month AS SELECT aircraft.type,
COUNT(*) as count_plane
FROM airport_scheme.aircraft JOIN airport_scheme.flight ON flight.aircraft_id =
aircraft.aircraft_id
WHERE flight.departure_date BETWEEN CURRENT_DATE - INTERVAL '1 month'
AND CURRENT_DATE GROUP BY aircraft.type
---
```

Что происходит в данном запросе?

Данный SQL-запрос создает/обновляет представление type\_flight\_month, которое показывает количество рейсов для каждого типа самолета за последний месяц.

Коротко:

1. Соединяет таблицы самолетов (aircraft) и рейсов (flight).
2. Фильтрует рейсы за последние 30 дней (BETWEEN CURRENT\_DATE - 1 month AND CURRENT\_DATE).
3. Группирует по типу самолета (aircraft.type) и считает количество рейсов (COUNT(\*)).

14

Data Output

Messages

Notifications

CREATE VIEW

Query returned successfully in 95 msec.

Data Output

Messages

Notifications

SQL

	type character varying (20)	count_plane bigint
1	грузовой	3
2	пассажирский	28

### 4.3 Запросы на модификацию данных

Запрос на модификацию данных №1: INSERT с подзапросом

Задача: Добавить билет для пассажира на рейс указанного тип самолета, который летит сегодня.

---

```
INSERT INTO airport_scheme.ticket (ticket_id, ticket_number, ticket_status,
purchase_date,
seat_number, class_of_service, baggage_insurance, price, flight_id, cashdesk_id,
passport_data_id)
SELECT (SELECT COALESCE(MAX(ticket_id)) + 1 FROM
airport_scheme.ticket), 'TICK1', 'активен', CURRENT_DATE, (SELECT MIN(seat)
FROM generate_series(1, aircraft.seat_capacity) AS seat
WHERE seat NOT IN ( SELECT seat_number
FROM airport_scheme.ticket WHERE ticket.flight_id = flight.flight_id )
AS seat_number, 'эконом', TRUE, 15000, flight.flight_id, 1,
passport_data.passport_data_id
FROM airport_scheme.passport_data
JOIN airport_scheme.flight ON TRUE
JOIN airport_scheme.aircraft ON flight.aircraft_id = aircraft.aircraft_id
WHERE aircraft.type = 'пассажирский' AND flight.departure_date =
CURRENT_DATE
ORDER BY flight.departure_date LIMIT 1
---
```

Что происходит в данном запросе?

Данный SQL-запрос добавляет новый билет в таблицу ticket для пассажирского рейса сегодняшней даты. Основное:

1. Генерирует ticket\_id как MAX(ticket\_id) + 1 (автоинкремент вручную).
2. Выбирает первое свободное место в самолете: минимальный номер места из диапазона 1–seat\_capacity, которого нет в уже проданных билетах на этот рейс.
3. Фиксирует параметры билет: статус "активен", класс "эконом", страховка багажа (TRUE), цена 15 000.
4. Соединяет таблицы passport\_data, flight и aircraft.
5. Фильтрует:
  - Только пассажирские самолеты (type = 'пассажирский').
  - Рейсы с вылетом сегодня (departure\_date = CURRENT\_DATE).
  - Выбирает первый подходящий рейс (LIMIT 1).

ДО

Data Output Messages Notifications											
Showing rows: 1 to 40 Page No: 1 of 1											
	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
30	30	T12374	использован	2025-04-15	38	бизнес	true	28000	30	30	30
31	31	T12375	активен	2025-04-16	39	эконом	false	15000	31	31	31
32	32	T12376	отменен	2025-04-14	40	первый	true	36000	32	32	32
33	33	T12377	активен	2025-04-17	41	эконом	true	15500	33	33	33
34	34	T12378	использован	2025-04-15	42	бизнес	false	27000	34	34	34
35	35	T12379	активен	2025-04-16	43	эконом	true	14500	35	35	35
36	36	T12380	отменен	2025-04-14	44	первый	false	33000	36	36	36
37	37	T12381	активен	2025-04-15	45	эконом	true	16000	37	37	37
38	38	T12382	использован	2025-04-13	46	бизнес	true	29000	38	38	38
39	39	T12383	активен	2025-04-16	47	эконом	false	15000	39	39	39
40	40	T12384	отменен	2025-04-14	48	первый	true	34000	40	40	40

ПОСЛЕ

Data Output Messages Notifications											
Showing rows: 1 to 41 Page No: 1 of 1											
	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
31	31	T12375	активен	2025-04-16	39	эконом	false	15000	31	31	31
32	32	T12376	отменен	2025-04-14	40	первый	true	36000	32	32	32
33	33	T12377	активен	2025-04-17	41	эконом	true	15500	33	33	33
34	34	T12378	использован	2025-04-15	42	бизнес	false	27000	34	34	34
35	35	T12379	активен	2025-04-16	43	эконом	true	14500	35	35	35
36	36	T12380	отменен	2025-04-14	44	первый	false	33000	36	36	36
37	37	T12381	активен	2025-04-15	45	эконом	true	16000	37	37	37
38	38	T12382	использован	2025-04-13	46	бизнес	true	29000	38	38	38
39	39	T12383	активен	2025-04-16	47	эконом	false	15000	39	39	39
40	40	T12384	отменен	2025-04-14	48	первый	true	34000	40	40	40
41	41	TICK1	активен	2025-05-21	1	эконом	true	15000	32	1	1

Запрос на модификацию данных №2: UPDATE с подзапросом

Задача: Обновление цены билетов на рейсы следующего месяца.

---

UPDATE airport\_scheme.ticket

SET price = price \* 1.1 WHERE flight\_id IN (SELECT flight\_id FROM airport\_scheme.flight WHERE departure\_date BETWEEN CURRENT\_DATE AND CURRENT\_DATE + INTERVAL '1 month');

---

Что происходит в данном запросе?

Данный SQL-запрос повышает цену на 10% для всех билетов, связанных с рейсами, которые вылетают в течение следующего месяца. Коротко:

1. Подзапрос выбирает flight\_id рейсов с датой вылета от сегодня (CURRENT\_DATE) до +1 месяц.
2. Основной запрос увеличивает цену (price \* 1.1) для билетов, привязанных к этим рейсам.

ДО

Data Output Messages Notifications											
Showing rows: 1 to 40 Page No: 1 of 1											
	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
1	1	T12345	активен	2025-04-15	12	эконом	true	15000	1	1	1
2	2	T12346	активен	2025-04-15	14	бизнес	false	30000	2	2	2
3	3	T12347	активен	2025-04-15	5	эконом	true	14000	3	3	3
4	4	T12348	использован	2025-04-14	8	первый	true	40000	4	4	4
5	5	T12349	активен	2025-04-16	3	эконом	false	12000	5	5	5
6	6	T12350	активен	2025-04-15	7	бизнес	true	25000	7	6	6
7	7	T12351	активен	2025-04-17	10	бизнес	false	25000	7	7	7
8	8	T12352	использован	2025-04-13	6	эконом	true	13000	8	8	8
9	9	T12353	активен	2025-04-15	15	первый	false	32000	9	9	9
10	10	T12354	отменен	2025-04-14	11	бизнес	true	27000	10	10	10
11	11	T12355	активен	2025-04-16	18	эконом	false	16000	11	11	11
12	12	T12356	активен	2025-04-17	20	первый	true	35000	12	12	12
13	13	T12357	использован	2025-04-14	21	эконом	false	14000	13	13	13
14	14	T12358	активен	2025-04-15	22	бизнес	true	29000	14	14	14
15	15	T12359	отменен	2025-04-16	23	эконом	true	14500	15	15	15
16	16	T12360	активен	2025-04-17	24	первый	false	34000	16	16	16
17	17	T12361	активен	2025-04-15	25	эконом	true	15500	17	17	17
18	18	T12362	активен	2025-04-13	26	бизнес	false	25000	7	18	18
19	19	T12363	активен	2025-04-16	27	эконом	true	16000	19	19	19
20	20	T12364	отменен	2025-04-14	28	первый	false	33000	20	20	20
21	21	T12365	активен	2025-04-17	29	эконом	true	15000	21	21	21
22	22	T12366	использован	2025-04-15	30	бизнес	true	28000	22	22	22
23	23	T12367	активен	2025-04-17	31	эконом	false	14000	23	23	23
24	24	T12368	отменен	2025-04-14	32	первый	true	34000	24	24	24

## ПОСЛЕ

	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
1	1	T12345	активен	2025-04-15	12	эконом	true	15000	1	1	1
2	2	T12346	активен	2025-04-15	14	бизнес	false	30000	2	2	2
3	3	T12347	активен	2025-04-15	5	эконом	true	14000	3	3	3
4	4	T12348	использован	2025-04-14	8	первый	true	40000	4	4	4
5	5	T12349	активен	2025-04-16	3	эконом	false	12000	5	5	5
6	6	T12350	активен	2025-04-15	7	бизнес	true	25000	7	6	6
7	7	T12351	активен	2025-04-17	10	бизнес	false	25000	7	7	7
8	8	T12352	использован	2025-04-13	6	эконом	true	13000	8	8	8
9	9	T12353	активен	2025-04-15	15	первый	false	32000	9	9	9
10	10	T12354	отменен	2025-04-14	11	бизнес	true	27000	10	10	10
11	11	T12355	активен	2025-04-16	18	эконом	false	16000	11	11	11
12	12	T12356	активен	2025-04-17	20	первый	true	35000	12	12	12
13	13	T12357	использован	2025-04-14	21	эконом	false	14000	13	13	13
14	14	T12358	активен	2025-04-15	22	бизнес	true	29000	14	14	14
15	15	T12359	отменен	2025-04-16	23	эконом	true	14500	15	15	15
16	16	T12360	активен	2025-04-17	24	первый	false	34000	16	16	16
17	17	T12361	активен	2025-04-15	25	эконом	true	15500	17	17	17
18	18	T12362	активен	2025-04-13	26	бизнес	false	25000	7	18	18
19	19	T12363	активен	2025-04-16	27	эконом	true	16000	19	19	19
20	20	T12364	отменен	2025-04-14	28	первый	false	33000	20	20	20
21	21	T12365	активен	2025-04-17	29	эконом	true	15000	21	21	21
22	22	T12366	использован	2025-04-15	30	бизнес	true	28000	22	22	22
23	23	T12367	активен	2025-04-17	31	эконом	false	14000	23	23	23
24	24	T12368	отменен	2025-04-14	32	первый	true	34000	24	24	24

Запрос на модификацию данных №3: DELETE с подзапросом

Задача: удаление билетов на рейсы с датой вылета раньше текущей даты  
 ...

```
DELETE FROM airport_scheme.ticket
WHERE flight_id IN (SELECT flight_id FROM airport_scheme.flight
WHERE departure_date < CURRENT_DATE);
...
```

Что происходит в данном запросе?

Данный SQL-запрос удаляет все билеты, связанные с рейсами, которые уже вылетели (дата вылета раньше текущей). Коротко:

1. Подзапрос выбирает flight\_id рейсов с departure\_date < сегодня.
2. Основной запрос удаляет билеты, привязанные к этим рейсам.

ДО



Data Output Messages Notifications											
Showing rows: 1 to 41 Page No: 1 of 1											
	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
1	1	T12345	активен	2025-04-15	12	эконом	true	15000	1	1	1
2	2	T12346	активен	2025-04-15	14	бизнес	false	30000	2	2	2
3	3	T12347	активен	2025-04-15	5	эконом	true	14000	3	3	3
4	4	T12348	использован	2025-04-14	8	первый	true	40000	4	4	4
5	5	T12349	активен	2025-04-16	3	эконом	false	12000	5	5	5
6	6	T12350	активен	2025-04-15	7	бизнес	true	25000	7	6	6
7	7	T12351	активен	2025-04-17	10	бизнес	false	25000	7	7	7
8	8	T12352	использован	2025-04-13	6	эконом	true	13000	8	8	8
9	9	T12353	активен	2025-04-15	15	первый	false	32000	9	9	9
10	10	T12354	отменен	2025-04-14	11	бизнес	true	27000	10	10	10
11	11	T12355	активен	2025-04-16	18	эконом	false	16000	11	11	11
12	12	T12356	активен	2025-04-17	20	первый	true	35000	12	12	12
13	13	T12357	использован	2025-04-14	21	эконом	false	14000	13	13	13
14	14	T12358	активен	2025-04-15	22	бизнес	true	29000	14	14	14
15	15	T12359	отменен	2025-04-16	23	эконом	true	14500	15	15	15
16	16	T12360	активен	2025-04-17	24	первый	false	34000	16	16	16
17	17	T12361	активен	2025-04-15	25	эконом	true	15500	17	17	17
18	18	T12362	активен	2025-04-13	26	бизнес	false	25000	7	18	18
19	19	T12363	активен	2025-04-16	27	эконом	true	16000	19	19	19
20	20	T12364	отменен	2025-04-14	28	первый	false	33000	20	20	20
21	21	T12365	активен	2025-04-17	29	эконом	true	15000	21	21	21
22	22	T12366	использован	2025-04-15	30	бизнес	true	28000	22	22	22
23	23	T12367	активен	2025-04-17	31	эконом	false	14000	23	23	23
24	24	T12368	отменен	2025-04-14	32	первый	true	34000	24	24	24
25	25	T12369	активен	2025-04-15	33	эконом	true	15000	25	25	25

## ПОСЛЕ

	ticket_id [PK] integer	ticket_number character varying (15)	ticket_status character varying (15)	purchase_date date	seat_number integer	class_of_service character varying (20)	baggage_insurance boolean	price integer	flight_id integer	cashdesk_id integer	passport_data_id integer
1	32	T12376	отменен	2025-04-14	40	первый	true	39600	32	32	32
2	33	T12377	активен	2025-04-17	41	эконом	true	17050	33	33	33
3	34	T12378	использован	2025-04-15	42	бизнес	false	29700	34	34	34
4	35	T12379	активен	2025-04-16	43	эконом	true	15950	35	35	35
5	36	T12380	отменен	2025-04-14	44	первый	false	36300	36	36	36
6	37	T12381	активен	2025-04-15	45	эконом	true	17600	37	37	37
7	38	T12382	использован	2025-04-13	46	бизнес	true	31900	38	38	38
8	39	T12383	активен	2025-04-16	47	эконом	false	16500	39	39	39
9	40	T12384	отменен	2025-04-14	48	первый	true	37400	40	40	40
10	41	TICK1	активен	2025-05-21	1	эконом	true	16500	32	1	1

## 4.4 Индексы

Индекс №1: Простой индекс  
ДО создания индекса

...

EXPLAIN ANALYZE

SELECT \* FROM airport\_scheme.passport\_data WHERE passport\_data\_id = 140

...

Что происходит?

Индекс анализирует производительность выборки данных паспорта с passport\_data\_id = 140.

1. EXPLAIN ANALYZE показывает план выполнения и реальное время работы запроса.
2. Если есть индекс на passport\_data\_id, используется Index Scan (быстрый поиск по индексу).
3. Без индекса — Seq Scan (перебор всех строк, медленнее).
4. Вывод включает: время поиска, количество строк, использование индекса.

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>		
	<b>QUERY PLAN</b> text <div>🔒</div>	
1	Index Scan using idx_passport_data_id on passport_data (cost=0.14..8.16 rows=1 width=22) (actual time=0.154..0.157 rows=1 loops=...	
2	Index Cond: (passport_data_id = 140)	
3	Planning Time: 0.227 ms	
4	Execution Time: 0.217 ms	

После создания индекса

...

```
CREATE INDEX idx_passport_data_id ON
airport_scheme.passport_data(passport_data_id)
```

...

Что сделали?

Создали `idx_passport_data_id` для столбца `passport_data_id` в таблице `passport_data` (схема `airport_scheme`). Индекс ускоряет поиск и фильтрацию по этому столбцу (например, `WHERE passport_data_id = 140`).

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>		
	<b>CREATE INDEX</b>	
	Query returned successfully in 222 msec.	

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>		
	<b>QUERY PLAN</b> text <div>🔒</div>	
1	Index Scan using passport_data_pkey on passport_data (cost=0.15..8.17 rows=1 width=98) (actual time=0.023..0.023 rows=0 loops=...	
2	Index Cond: (passport_data_id = 140)	
3	Planning Time: 0.091 ms	
4	Execution Time: 0.039 ms	

Query

Query History

1

DROP INDEX airport\_scheme.idx\_passport\_data\_id

Data Output

Messages

Notifications

DROP INDEX

Индекс №2: Составной индекс

До создания индекса

---

EXPLAIN ANALYZE

SELECT \* FROM airport\_scheme.passport\_data

WHERE (passenger\_id BETWEEN 41 AND 140) AND series = '678901'

---

6	Planning Time: 0.200 ms
7	Execution Time: 0.171 ms

После создания индекса

---

CREATE INDEX idx\_passport ON airport\_scheme.passport\_data(passenger\_id, series)

---

Query

Query History

1

CREATE INDEX idx\_passport ON airport\_scheme.passport\_data(passenger\_id, series)

Data Output		Messages	Notifications
<div> <div>≡+</div> <div></div> <div>▼</div> <div></div> <div>▼</div> <div></div> <div></div> <div></div> <div></div> <div>SQL</div> </div>			
	<b>QUERY PLAN</b> text <div>🔒</div>		
1	Seq Scan on passport_data (cost=0.00..21.20 rows=1 width=98) (actual time=0.010..0.011 rows=0 loops=...		
2	Filter: ((passenger_id >= 41) AND (passenger_id <= 140) AND ((series)::text = '678901'::text))		
3	Rows Removed by Filter: 50		
4	Planning Time: 0.044 ms		
5	Execution Time: 0.018 ms		

Query		Query History
1	<b>DROP INDEX</b> airport_scheme.idx_passport	

## Выводы

В данной лабораторной работе были получены основные навыки создания представлений и запросов на выборку данных к базе данных PostgreSQL, а также навыки по работе с индексами.