



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (ЦТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Разработка баз данных»

Практическое занятие № 5

Студенты группы

ИКБО-66-23 Смирнов А.Ю.

(подпись)

Ассистент

Копылова Я.А.

(подпись)

Отчет представлен

«___»_____2025 г.

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №5. ОБЪЕКТЫ БАЗЫ ДАННЫХ: ПРЕДСТАВЛЕНИЯ И ХРАНИМЫЕ ПРОЦЕДУРЫ

Цель работы: Работа направлена на формирование у студентов углубленных навыков работы с объектами баз данных в СУБД PostgreSQL, смещая акцент от прямого манипулирования данными к созданию переиспользуемых логических конструкций.

Таблица 1. Car_type

car_type 1 (3) × insurance 1 (4) client 1 (5) payment_method 1 (6) booking_status 1 (7) car 1 (8) tariff 1 (9)							
SELECT * FROM Payment Введите SQL выражение чтобы отфильтровать результаты							
Таблица	AZ id_car_type	AZ type_name	123 seats	123 luggage_capacity	AZ fuel_type	AZ transmission	
1	CT001	Эконом	5	2	Бензин	Механика	
2	CT002	Комфорт	5	3	Бензин	Автомат	
3	CT003	Бизнес	5	3	Дизель	Автомат	
4	CT004	Премиум	5	4	Бензин	Автомат	

Таблица 2. insurance

car_type 1 (3) × insurance 1 (4) × client 1 (5) payment_method 1 (6) booking_status 1 (7) car 1 (8)					
SELECT * FROM Payment Введите SQL выражение чтобы отфильтровать результаты					
Таблица	AZ id_insurance	AZ insurance_name	123 daily_cost	AZ coverage_description	is_active
1	INS003	Премиум	800	Покрытие без франшизы	[v]
2	INS001	Стандарт	300	Базовое покрытие	[v]
3	INS002	Расширенная	500	Полное покрытие	[v]

Таблица 3. client

car_type 1 (3) × insurance 1 (4) client 1 (5) × payment_method 1 (6) booking_status 1 (7) car 1 (8) tariff 1 (9) invoice 1 (10) booking 1 (11) payment 1 (12) payment 2										
Connection: dbstud Time: 2025-11-05 23:10:08.295 Query: -- 3. Вывод всех типов автомобилей SELECT * FROM Car_Type										
Таблица	AZ last_name	AZ middle_name	AZ driver_license	AZ phone	AZ email	AZ passport_number	registration_date	AZ client_type		
1	CL001	Лариса	77BB654321	+79180001122	larisa@gmail.com	4510987654	2025-09-25	VIP		
2	CL003	Сергей	77CC112233	+79182223344	volkov@yandex.ru	4510567890	2025-09-25	Corporate		
3	CL001	Олег	77AA123456	+79181112233	oleg@mail.ru	4510123456	2025-09-25	Regular		

Таблица 4. *Payment_method*

car_type 1 (3) X		insurance 1 (4)		client 1 (5)		payment_method 1 (6) X	
SELECT * FROM Car_Type		Connection: dbstud Time: 2025-11-05 23:10:08.295 Query: -- 3. Вывод всех типов автомобилей SELECT * FROM Car_Type		бы отфильтровать результаты		name <input type="text"/> is_active <input checked="" type="checkbox"/>	
1	PM001	Кредитная карта		[v]			
2	PM002	Дебетовая карта		[v]			
3	PM003	Наличные		[v]			
4	PM004	Банковский перевод		[v]			

Таблица 5. *Booking_status*

car_type 1 (3) X		insurance 1 (4)		client 1 (5)		payment_method 1 (6)		booking_status 1 (7) X	
SELECT * FROM Car_Type		Connection: dbstud Time: 2025-11-05 23:10:08.295 Query: -- 3. Вывод всех типов автомобилей SELECT * FROM Car_Type		бы отфильтровать результаты		description <input type="text"/>			
1	BS001	Ожидание		Бронь ожидает подтверждения					
2	BS002	Подтверждена		Бронь подтверждена					
3	BS003	Активна		Автомобиль у клиента					
4	BS004	Завершена		Автомобиль возвращен					
5	BS005	Отменена		Бронь отменена					

Таблица 6. *Car*

car_type 1 (3)		insurance 1 (4)		client 1 (5)		payment_method 1 (6)		booking_status 1 (7)		car 1 (8) X		tariff 1 (9)		invoice 1 (10)		booking 1 (11)		payment 1 (12)		Статистика 1	
LOC001		INS001		Kia		Rio		2 023		A123BC77		Белый		15 000		Available		[NUL]			
LOC002		INS002		Toyota		Camry		2 023		B456HE77		Черный		18 000		Available		[NUL]			
LOC003		INS003		BMW		5 Series		2 024		C789TX77		Синий		5 000		Available		[NUL]			
LOC004		INS004		Mercedes		E-Class		2 023		E111AA77		Черный		12 000		Available		[NUL]			

Таблица 7. *tariff*

car_type 1 (3)insurance 1 (4)client 1 (5)payment_method 1 (6)booking_status 1 (7)car 1 (8)tariff 1 (9)invoice 1 (10)booking 1 (11)payment 1 (12)Статистика

Connection: dbstud
Time: 2025-11-05 23:24:02.937
Query: -- 3. Вывод всех типов автомобилей
SELECT * FROM Car_Type

бы отфильтровать результаты

Таблица

	AZ id_invoice	tariff_name	123 daily_rate	123 weekly_rate	123 monthly_rate	123 km_included	123 extra_km_cost	valid_from	valid_to
1	TAR001	CT001	Эконом суточный	1 500	9 000	35 000	200	10	2024-01-01
2	TAR002	CT002	Комфорт суточный	2 800	16 800	65 000	200	12	2024-01-01
3	TAR003	CT003	Бизнес суточный	4 200	25 200	98 000	300	15	2024-01-01
4	TAR004	CT004	Премиум суточный	6 500	39 000	150 000	300	20	2024-01-01

Таблица 8. invoice

car_type 1 (3)insurance 1 (4)client 1 (5)payment_method 1 (6)booking_status 1 (7)car 1 (8)tariff 1 (9)invoice 1 (10)booking 1 (11)payment 1 (12)Статистика

Введите SQL выражение чтобы отфильтровать результаты

Таблица

	AZ id_invoice	issue_date	due_date	123 base_amount	123 insurance_cost	123 tax_amount	123 discount_amount	123 total_amount	AZ payment_status
1	INV001	2024-09-20	2024-09-27	10 500	2 100	1 260	0	13 860	Paid
2	INV002	2024-09-22	2024-09-29	8 400	1 680	1 008	0	11 088	Paid
3	INV003	2024-09-25	2024-10-02	12 600	2 520	1 512	0	16 632	Pending

Таблица 9. booking

car_type 1 (3)insurance 1 (4)client 1 (5)payment_method 1 (6)booking_status 1 (7)car 1 (8)tariff 1 (9)invoice 1 (10)booking 1 (11)payment 1 (12)Статистика

Введите SQL выражение чтобы отфильтровать результаты

Таблица

	AZ id_booking	AZ id_client	AZ id_car	AZ id_booking_status	AZ id_employee	AZ id_invoice	AZ id_tariff	pickup_date	return_date
1	BK001	CL001	CAR001	BS004	EMP001	INV001	TAR001	2024-09-15	2024-09-22
2	BK002	CL002	CAR002	BS003	EMP002	INV002	TAR002	2024-09-20	2024-09-27
3	BK003	CL003	CAR003	BS002	EMP003	INV003	TAR003	2024-09-25	2024-10-02

Таблица 10. Payment

car_type 1 (3) insurance 1 (4) client 1 (5) payment_method 1 (6) booking_status 1 (7) car 1 (8) tariff 1 (9) invoice 1 (10) booking 1 (11) payment 1 (12) X									
SELECT * FROM Payment Введите SQL выражение чтобы отфильтровать результаты									
Таблица		AZ id_payment	AZ id_invoice	AZ id_payment_method	123 amount	payment_date	AZ transaction_id	AZ receipt_number	
	1	PAY001	INV001	PM001	13 860	2024-09-20 14:30:00.000	TXN00123456	[NULL]	
	2	PAY002	INV002	PM002	11 088	2024-09-22 10:15:00.000	TXN00123457	[NULL]	

Таблица 11. Location

location 1 X employee 1 (2) car_type 1 (3) insurance 1 (4) client 1 (5) payment						
SELECT * FROM Location Введите SQL выражение чтобы отфильтровать результаты						
Таблица		AZ id_location	AZ address	AZ city	AZ phone	AZ manager_id
	1	LOC001	ул. Ленина, 123	Москва	+74951234567	[NULL]
	2	LOC002	пр. Мира, 45	Санкт-Петербур	+78127654321	[NULL]
	3	LOC003	ул. Садовая, 67	Казань	+78431234567	[NULL]

Таблица 11. Employee

employee 1 (2) X

car_type 1 (3)

insurance 1 (4)

client 1 (5)

payment_method 1 (6)

booking_status 1 (7)

car 1 (8)

tariff 1 (9)

invoice 1 (10)

booking 1 (11)

⌵

SELECT * FROM Employee

Введите SQL выражение чтобы отфильтровать результаты

⌵

⌵

Таблица

	AZ id_employee	AZ first_name	AZ last_name	AZ middle_name	AZ position	AZ phone	AZ email	hire_date	123 salary	AZ id_location
1	EMP001	Анна	Иванова	[NULL]	Менеджер	+79161234567	anna@rentcar.ru	2023-01-15	80 000	LOC001
2	EMP002	Игорь	Смирнов	[NULL]	Администратор	+79167654321	igor@rentcar.ru	2023-02-01	60 000	LOC001
3	EMP003	Мария	Петрова	[NULL]	Агент	+79165556677	maria@rentcar.ru	2023-03-10	50 000	LOC002
4	EMP004	Дмитрий	Козлов	[NULL]	Менеджер	+79164443322	dmitry@rentcar.ru	2023-01-20	75 000	LOC003

Задание №1: Создание модифицируемого представления

The screenshot displays a database IDE interface. The top pane shows a SQL script for creating a modifiable view named `vip_clients`. The script uses `CREATE OR REPLACE VIEW` and selects columns from the `Client` table where `client_type = 'VIP'`.

```
CREATE OR REPLACE VIEW vip_clients AS
SELECT
  id_client,
  first_name,
  last_name,
  driver_license,
  phone,
  email,
  passport_number,
  client_type
FROM
  Client
WHERE
  client_type = 'VIP';
```

The bottom pane shows the execution of the query `select * from vip_clients`. The results are displayed in a table with 8 columns and 2 rows.

	AZ id_client	AZ first_name	AZ last_name	AZ driver_license	AZ phone	AZ email	AZ passport_number	AZ client_type
1	CL002	Лариса	Кузнецова	778B654321	+79180001122	larisa@gmail.com	4510987654	VIP
2	CL004	Елена	Соколова	77DD445566	+79183334455	sokolova@mail.ru	4510112233	VIP

Представление выбирает клиентов с типом "VIP" с полной информацией об их данных. Оно является модифицируемым, так как удовлетворяет всем условиям PostgreSQL: одна базовая таблица (`Client`), нет агрегатных функций, `GROUP BY`, `DISTINCT`, оконных функций и других ограничивающих конструкций.

Задание №2: Модификация данных через представление

The screenshot shows a database IDE with a script editor and a results pane. The script editor contains the following SQL code:

```
INSERT INTO vip_clients (  
    id_client, first_name, last_name, driver_license,  
    phone, email, passport_number, client_type  
) VALUES (  
    'L004', 'Елена', 'Соколова', '77DD445566', |  
    '+79183334455', 'sokolova@mail.ru', '4510112233', 'VIP'  
);  
SELECT * FROM Client WHERE client_type = 'VIP';
```

The results pane shows the output of the SELECT statement, displaying two rows of data from the Client table where client_type is 'VIP'.

	id_client	first_name	last_name	middle_name	driver_license	phone	email	passport_number	registration_date	client_type
1	L002	Лариса	Кузнецова	[NULL]	77BB654321	+79180001122	larisa@gmail.com	4510987654	2025-09-25	VIP
2	L004	Елена	Соколова	[NULL]	77DD445566	+79183334455	sokolova@mail.ru	4510112233	2025-11-06	VIP

The screenshot shows a database IDE with a script editor. The script contains the following SQL code:

```
-- Удаление VIP клиента через представление  
DELETE FROM vip_clients  
WHERE first_name = 'Елена' AND last_name = 'Соколова';  
  
-- Проверка, что запись действительно удалена из основной таблицы  
SELECT * FROM Client WHERE first_name = 'Елена' AND last_name = 'Соколова';
```

The results pane shows the output of the SELECT statement, which is empty, indicating that the client has been successfully deleted from the Client table.

	id_client	first_name	last_name	middle_name	driver_license	phone	email	passport_number	registration_date	client_type
--	-----------	------------	-----------	-------------	----------------	-------	-------	-----------------	-------------------	-------------

Демонстрируется возможность выполнения DML-операций через представление. DELETE удаляет VIP клиента по имени и фамилии, а SELECT проверяет отсутствие записи в базовой таблице Client. Изменения автоматически применяются к базовой таблице Client через модифицируемое представление vip_clients.

Задание №3: Создание немодифицируемого аналитического представления

Script-1Script-2Script-3Script-6Script-5Script-4Script-7

CREATE OR REPLACE VIEW car_type_summary AS

SELECT

ct.type_name,

COUNT(c.id_car) AS total_cars,

COALESCE(ROUND(AVG(t.daily_rate), 2), 0) AS average_daily_rate,

COUNT(c.id_car) FILTER (

WHERE c.status = 'Available'

) AS available_cars_count

FROM

Car_Type ct

LEFT JOIN

Car c ON ct.id_car_type = c.id_car_type

LEFT JOIN

Tariff t ON ct.id_car_type = t.id_car_type

GROUP BY

ct.id_car_type, ct.type_name;

Статистика 1Статистика 2Статистика 3

NameValue

Updated Rows0

Execute time0.033s

Start timeThu Nov 06 00:55:39 MSK 2025

Finish timeThu Nov 06 00:56:00 MSK 2025

Query-- Создание немодифицируемого аналитического представления с минимальным использованием JOIN

CREATE OR REPLACE VIEW car_type_summary AS

SELECT

ct.type_name,

COUNT(c.id_car) AS total_cars,

COALESCE(ROUND(AVG(t.daily_rate), 2), 0) AS average_daily_rate,

COUNT(c.id_car) FILTER (

WHERE c.status = 'Available'

) AS available_cars_count

FROM

Car_Type ct

LEFT JOIN

Car c ON ct.id_car_type = c.id_car_type

LEFT JOIN

Tariff t ON ct.id_car_type = t.id_car_type

GROUP BY

ct.id_car_type, ct.type_name

Script-1Script-2Script-3Script-5Script-6

SELECT * FROM car_type_summary

vip_clients 1car_type_summary 2

SELECT * FROM car_type_summary

Введите SQL выражение чтобы отфильтровать результаты

Таблица

AZ type_name123 total_cars123 average_daily_rate123 available_cars_count

1Комфорт12 8001

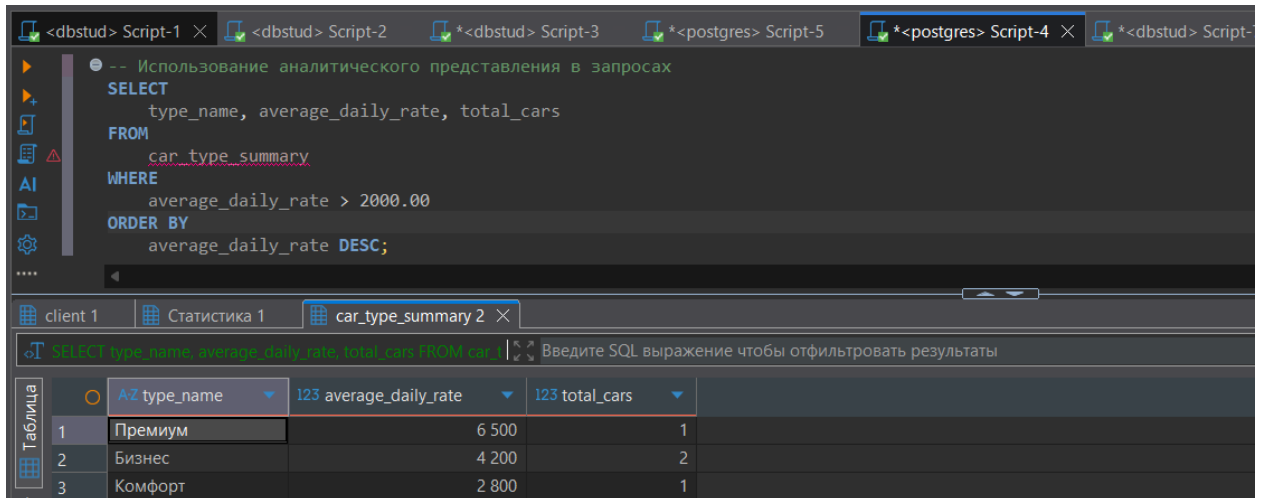
2Премиум6 5001

3Бизнес4 2002

4Эконом1 5001

Представление объединяет данные из трех таблиц (Car_Type, Car, Tariff) и использует агрегатные функции для анализа статистики по типам автомобилей. Оно является немодифицируемым из-за использования JOIN, GROUP BY и агрегатных функций.

Задание №4: Использование аналитического представления в запросах



The screenshot shows a database client window with multiple tabs. The active tab is titled "car_type_summary 2". The SQL editor contains the following query:

```
-- Использование аналитического представления в запросах
SELECT
    type_name, average_daily_rate, total_cars
FROM
    car_type_summary
WHERE
    average_daily_rate > 2000.00
ORDER BY
    average_daily_rate DESC;
```

Below the query editor, the results are displayed in a table. The table has three columns: "type_name", "average_daily_rate", and "total_cars". The results are sorted by "average_daily_rate" in descending order.

	A-Z type_name	123 average_daily_rate	123 total_cars
1	Премиум	6 500	1
2	Бизнес	4 200	2
3	Комфорт	2 800	1

Запрос использует созданное представление для отбора типов автомобилей с высокой средней суточной стоимостью аренды (больше 2000 рублей) и наличием автомобилей в парке. Это демонстрирует удобство использования представлений для сложной аналитики в сфере проката автомобилей.

Задание №5: Создание и обновление материализованного представления

totalrentalsbyclient 5					
Введите SQL выражение чтобы отфильтровать результаты					
	AZ id_client	AZ first_name	AZ last_name	123 total_rentals	123 total_spent
1	CL002	Лариса	Кузнецова	1	11 088
2	CL003	Сергей	Волков	1	16 632
3	CL004	Елена	Соколова	0	0
4	CL001	Олег	Морозов	1	13 860

```
-- Создание материализованного представления для отчета по аренде автомобиля
CREATE MATERIALIZED VIEW TotalRentalsByClient AS
SELECT
    c.id_client,
    c.first_name,
    c.last_name,
    COUNT(b.id_booking) AS total_rentals,
    COALESCE(SUM(i.total_amount), 0) AS total_spent
FROM
    Client c
LEFT JOIN
    Booking b ON c.id_client = b.id_client
LEFT JOIN
    Invoice i ON b.id_invoice = i.id_invoice
GROUP BY
    c.id_client, c.first_name, c.last_name;

-- Просмотр данных материализованного представления
SELECT * FROM TotalRentalsByClient;
```

```
-- 2. Демонстрация обновления данных
-- Добавим новую сессию для проверки

INSERT INTO Booking (id_booking, id_client, id_car, id_booking_status, id_employee,
id_invoice, id_tariff, pickup_date, return_date, total_days, pickup_location,
return_location)

VALUES ('BK004', 'CL001', 'CAR002', 'BS004', 'EMP001', 'INV004', 'TAR002', '2024-09-
28', '2024-09-30', 2, 'LOC001', 'LOC001');
```

The screenshot shows a database management tool interface. The top part displays a table named 'booking' with columns: id_booking, id_client, id_car, id_booking_status, id_employee, id_invoice, id_tariff, pickup_date, return_date, total_days, pickup_location, and return_location. The table contains 4 rows of data.

Below the table, there is a SQL editor with the following queries:

```
REFRESH MATERIALIZED VIEW TotalRentalsByClient;
SELECT * FROM TotalRentalsByClient;
```

The bottom part of the screenshot shows a table named 'totalrentalsbyclient' with columns: id_client, first_name, last_name, total_rentals, and total_spent. The table contains 4 rows of data.

id_client	first_name	last_name	total_rentals	total_spent
CL002	Лариса	Кузнецова	1	11 088
CL003	Сергей	Волков	1	16 632
CL004	Елена	Соколова	0	0
CL001	Олег	Морозов	2	21 252

Материализованное представление TotalRentalsByClient хранит физическую копию данных для быстрого доступа к аналитике по арендной активности клиентов. Команда REFRESH MATERIALIZED VIEW обновляет данные при изменениях в базовых таблицах Booking.

Задан пользовательской функции

```
-- создает механизм для расчета общей суммы, потраченной конкретным
-- клиентом на аренду автомобилей

CREATE OR REPLACE FUNCTION get_client_total_revenue(p_client_id
VARCHAR(10))
RETURNS DECIMAL(12, 2) AS $$
DECLARE
    total_revenue DECIMAL(12, 2);
    client_exists BOOLEAN;
BEGIN
    -- Проверяем существование клиента
    SELECT EXISTS(SELECT 1 FROM Client WHERE id_client = p_client_id)
    INTO client_exists;

    IF NOT client_exists THEN
        RAISE EXCEPTION 'Client with ID % does not exist.', p_client_id;
    END IF;

    -- Вычисляем общую сумму потраченную клиентом
    SELECT
        COALESCE(SUM(i.total_amount), 0.00)
    INTO total_revenue
    FROM Booking AS b
    JOIN Invoice AS i ON b.id_invoice = i.id_invoice
    WHERE b.id_client = p_client_id AND i.payment_status = 'Paid';

    RETURN total_revenue;
END;
$$ LANGUAGE plpgsql;
```

=

Name	Value
Execute time	0.034s
Start time	Thu Nov 06 01:37:05 MSK 2025
Finish time	Thu Nov 06 01:37:30 MSK 2025
Query	<pre>CREATE OR REPLACE FUNCTION get_client_total_revenue(p_client_id VARCHAR(10)) RETURNS DECIMAL(12, 2) AS \$\$ DECLARE total_revenue DECIMAL(12, 2); client_exists BOOLEAN; BEGIN -- Проверяем существование клиента SELECT EXISTS(SELECT 1 FROM Client WHERE id_client = p_client_id) INTO client_exists; IF NOT client_exists THEN RAISE EXCEPTION 'Client with ID % does not exist.', p_client_id; END IF; -- Вычисляем общую сумму потраченную клиентом SELECT COALESCE(SUM(i.total_amount), 0.00) INTO total_revenue FROM Booking AS b JOIN Invoice AS i ON b.id_invoice = i.id_invoice WHERE b.id_client = p_client_id AND i.payment_status = 'Paid'; RETURN total_revenue; END; \$\$ LANGUAGE plpgsql</pre>

```
-- Демонстрация вызова функции
SELECT
    c.id_client,
    c.first_name,
    c.last_name,
    c.client_type,
    get_client_total_revenue(c.id_client) AS total_revenue
FROM
    Client c
ORDER BY
    total_revenue DESC;
```

	AZ id_client	AZ first_name	AZ last_name	AZ client_type	123 total_revenue
1	CL001	Олег	Морозов	Regular	21 252
2	CL002	Лариса	Кузнецова	VIP	11 088
3	CL003	Сергей	Волков	Corporate	0
4	CL004	Елена	Соколова	VIP	

Задание №7: Разработка хранимой процедуры

```
-- Процедура для добавления нового клиента
-- Процедура для добавления нового клиента
CREATE OR REPLACE PROCEDURE add_new_client(
    p_first_name VARCHAR(20),
    p_last_name VARCHAR(20),
    p_driver_license VARCHAR(15),
    p_phone VARCHAR(15),
    p_passport_number VARCHAR(20),
    OUT p_success BOOLEAN,
    OUT p_message TEXT
)
LANGUAGE plpgsql AS $$
DECLARE
    v_new_client_id VARCHAR(10);
    v_client_count INTEGER;
BEGIN
    -- Проверка уникальности водительского удостоверения
    IF EXISTS(SELECT 1 FROM Client WHERE driver_license = p_driver_license) THEN
        p_success := FALSE;
        p_message := 'Ошибка: водительское удостоверение уже существует';
        RETURN;
    END IF;

    -- Проверка уникальности паспорта
    IF EXISTS(SELECT 1 FROM Client WHERE passport_number = p_passport_number) THEN
        p_success := FALSE;
        p_message := 'Ошибка: номер паспорта уже существует';
        RETURN;
    END IF;

    -- Генерация ID клиента
    SELECT COALESCE(MAX(CAST(SUBSTRING(id_client FROM 3) AS INTEGER)), 0) + 1
    INTO v_client_count FROM Client;
    v_new_client_id := 'CL' || LPAD(v_client_count::TEXT, 3, '0');

    -- Добавление клиента
    INSERT INTO Client (
        id_client, first_name, last_name, driver_license, phone, passport_number
    ) VALUES (
        v_new_client_id, p_first_name, p_last_name, p_driver_license, p_phone,
        p_passport_number
    );

    p_success := TRUE;
    p_message := 'Клиент ' || v_new_client_id || ' успешно добавлен';

EXCEPTION
    WHEN OTHERS THEN
        p_success := FALSE;
        p_message := 'Ошибка: ' || SQLERRM;
END;
$$;
```

Статистика 1		Статистика 2	×
Name	Value		
Updated Rows	0		
Execute time	0.014s		
Start time	Thu Nov 06 02:31:15 MSK 2025		
Finish time	Thu Nov 06 02:31:15 MSK 2025		
Query	<pre> -- Процедура для добавления нового клиента CREATE OR REPLACE PROCEDURE add_new_client(p_first_name VARCHAR(20), p_last_name VARCHAR(20), p_driver_license VARCHAR(15), p_phone VARCHAR(15), p_passport_number VARCHAR(20), OUT p_success BOOLEAN, OUT p_message TEXT) LANGUAGE plpgsql AS \$\$ DECLARE v_new_client_id VARCHAR(10); v_client_count INTEGER; BEGIN -- Проверка уникальности водительского удостоверения IF EXISTS(SELECT 1 FROM Client WHERE driver_license = p_driver_license) THEN p_success := FALSE; p_message := 'Ошибка: водительское удостоверение уже существует'; RETURN; END IF; -- Проверка уникальности паспорта IF EXISTS(SELECT 1 FROM Client WHERE passport_number = p_passport_number) THEN p_success := FALSE; p_message := 'Ошибка: номер паспорта уже существует'; RETURN; END IF; -- Генерация ID клиента SELECT COALESCE(MAX(CAST(SUBSTRING(id_client FROM 3) AS INTEGER)), 0) + 1 INTO v_client_count FROM Client; v_new_client_id := 'CL' LPAD(v_client_count::TEXT, 3, '0'); -- Добавление клиента INSERT INTO Client (id_client, first_name, last_name, driver_license, phone, passport_number) VALUES (v_new_client_id, p_first_name, p_last_name, p_driver_license, p_phone, p_passport_number); p_success := TRUE; p_message := 'Клиент ' v_new_client_id ' успешно добавлен'; EXCEPTION WHEN OTHERS THEN p_success := FALSE; p_message := 'Ошибка: ' SQLERRM; END; \$\$ </pre>		

Процедура process_new_client инкапсулирует бизнес-логику добавления нового клиента

с проверками уникальности водительского удостоверения и паспортных данных.

Использует выходные параметры для возврата статуса операции.

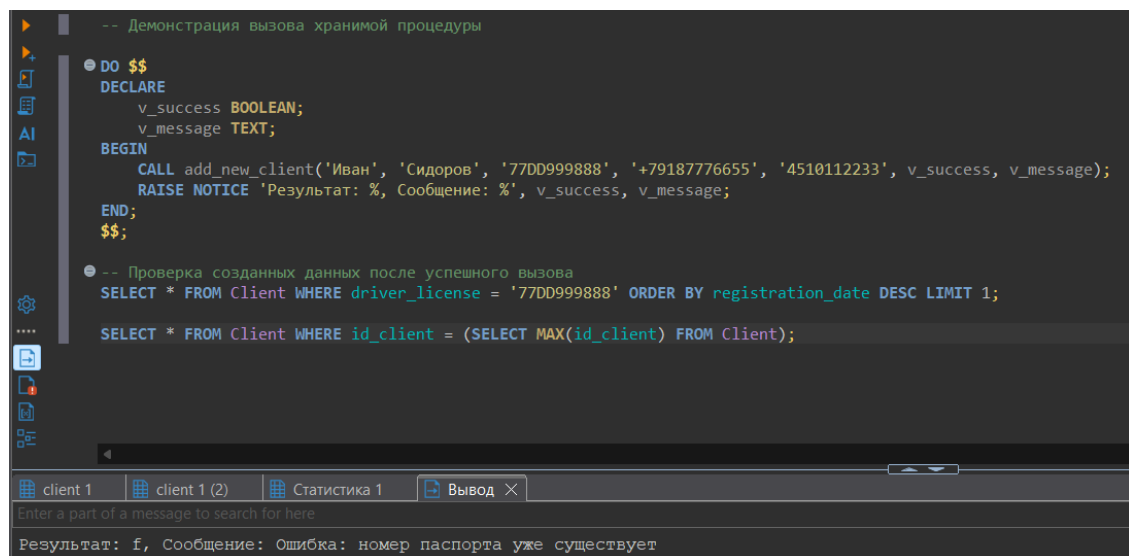
Задание №8: Демонстрация вызова хранимой процедуры

```
-- 1. Успешный вызов процедуры
DO $$
DECLARE
    v_success BOOLEAN;
    v_message TEXT;

BEGIN
    CALL add_new_client('Иван', 'Сидоров', '7700999288', '+79187776455',
'4510212299', v_success, v_message);

    RAISE NOTICE 'Результат: %, Сообщение: %', v_success, v_message;

END;
$;
```



The screenshot shows a PostgreSQL IDE with a SQL script in the editor and its execution results in the output pane. The script is as follows:

```
-- Демонстрация вызова хранимой процедуры
DO $$
DECLARE
    v_success BOOLEAN;
    v_message TEXT;
BEGIN
    CALL add_new_client('Иван', 'Сидоров', '77DD999888', '+79187776655', '4510112233', v_success, v_message);
    RAISE NOTICE 'Результат: %, Сообщение: %', v_success, v_message;
END;
$;
```

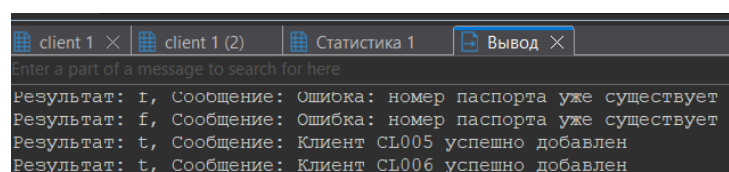
Below the script, there are two SQL queries for verification:

```
-- Проверка созданных данных после успешного вызова
SELECT * FROM Client WHERE driver_license = '77DD999888' ORDER BY registration_date DESC LIMIT 1;

SELECT * FROM Client WHERE id_client = (SELECT MAX(id_client) FROM Client);
```

The output pane shows the results of the script execution:

```
client 1 | client 1 (2) | Статистика 1 | Вывод X
Enter a part of a message to search for here
Результат: f, Сообщение: Ошибка: номер паспорта уже существует
```



This screenshot shows the output pane with multiple tabs and execution results:

```
client 1 X | client 1 (2) | Статистика 1 | Вывод X
Enter a part of a message to search for here
Результат: t, Сообщение: Ошибка: номер паспорта уже существует
Результат: f, Сообщение: Ошибка: номер паспорта уже существует
Результат: t, Сообщение: Клиент CL005 успешно добавлен
Результат: t, Сообщение: Клиент CL006 успешно добавлен
```

Первый вызов демонстрирует успешное добавление клиента, второй - срабатывание проверки на уникальность данных.

Процедура возвращает детальные сообщения о результате операции через выходные параметры.

Вывод: в ходе работы освоены представления и хранимые процедуры в PostgreSQL. Научены создавать модифицируемые, немодифицируемые и материализованные представления, различать функции и процедуры, использовать PL/pgSQL с условной логикой и обработкой ошибок. Освоен вызов процедур через CALL и работа с их результатами.