Практика по базам данных

ОТЧЁТ

Родионов Максим
Группа 23.Б15

Предметная область: «Продажа автомобилей»

Содержание

1 ОПИСАНИЕ СИСТЕМЫ

1.1 Требования

Задача заключается в автоматизации работы компании, которая занимается продажей новых и подержанных автомобилей. В системе должен вестись учет продаж автомобилей, причем должны фиксироваться не только модели и года выпуска, но и другие базовые характеристики новых и подержанных автомобилей, а для последних – еще и показатели их состояния. При учете продаж следует автоматически отслеживать рейтинг сотрудников компании (в первую очередь - количество продаж), поскольку эти данные впоследствии будут передаваться в отдельную расчетную систему. Кроме того, система должна хранить информацию для создания статистических отчетов по клиентам (физическим и юридическим лицам).

1.2 Модель данных

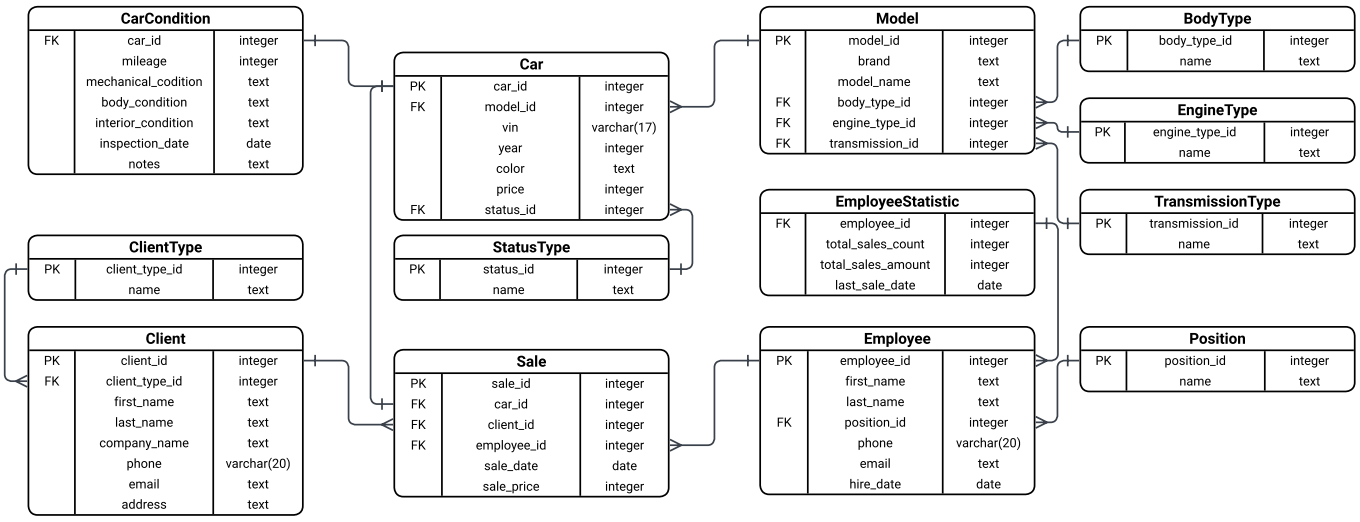

Рис. 1: ER-диаграмма

1.3 Функциональность

1.3.1 Серверная часть

Серверная логика реализована с использованием хранимых процедур, триггеров и представлений. Основные компоненты:

Хранимые процедуры / функции	**Реализация**	**Комментарии**
Регистрация нового клиента	—	
Оформление продажи	—	автоматически обновляет статус авто и статистику
Резервирование автомобиля	`reserve_car()`	меняет статус на «Reserved»
Возврат продажи	`refund_sale()`	удаляет продажу и восстанавливает статус
Обновление статистики сотрудника	`update_employee_stat()`	универсальная функция, которая вызывается в триггерах и процедурах
Отчёт по складу в JSON	`get_inventory_json()`	для интеграции с внешними системами
...		

Триггеры	**Реализация**	**Комментарии**
Автоматическое обновление статистики	`trg_update_stat_after_sale`	после каждой продажи
Смена статуса авто на «Sold»	`trg_set_car_sold_after_sale`	при оформлении продажи
Создание записи о состоянии авто	`trg_maintain_car_condition`	при добавлении авто
Проверка корректности VIN-кода	—	при добавлении авто
Запрет повторной продажи	`trg_prevent_sale_of_sold_car`	блокирует продажу уже проданного авто
Обновление даты последнего технического осмотра	—	
Проверка корректности пробега	—	не может уменьшаться при обновлении записи
...		

Представления	**Реализация**	**Комментарии**
Доступные автомобили	`v_available_cars`	для каталога

Представления	**Реализация**	**Комментарии**
История продаж	`v_sales_report`	с именами клиентов и сотрудников
Эффективность сотрудников	`v_employee_performance`	для рейтинга и отчётов
Статистика по моделям	—	для анализа прибыльности моделей
Автомобили, находящиеся в сервисе	—	
Статистика продаж по месяцам с динамикой и сравнением показателей	—	
История покупок клиентов	—	
...		

1.3.2 Клиентская часть

Клиентская часть использует представления, процедуры и прямые SQL-запросы для формирования экранов и отчётов:

Экранные формы основные	**Дополнительные**	**Реализация (запрос)**	**Серверная часть**
Реестр клиентов			
	Добавление клиента		
	Изменение данных клиента		
	Фильтр клиентов	(05) Клиенты, купившие автомобили марки Volkswagen (с именем/названием, email, адресом)	
Реестр автомобилей			

Экранные формы основные	**Дополнительные**	**Реализация (запрос)**	**Серверная часть**
	Добавление авто		`trg_maintain_car_condition` (для каскадного добавления CarCondition)
	Фильтры авто (по бренду, году, цене, статусу, . . .)	(01) Автомобили без записей в Sale и (06) Доступные автомобили с пробегом более 50 000 км	`v_available_cars` (доступные авто)
	Изменение данных авто		
	Резервирование авто		`reserve_car()`
	Удаление авто		`trg_maintain_car_condition` (для каскадного удаления CarCondition)
	Рейтинг авто	(12) Автомобили, цена которых выше средней по их бренду	
Реестр сотрудников			
	Добавление сотрудника		
	Изменение данных сотрудника		
Реестр продаж			
	Добавление продажи		`trg_set_car_sold_after_sale` (автоматическая установка статуса «Продано» для авто) и `trg_update_stat_after_sale` (автоматическое обновление статистики продавца)
	Удаление продажи		`refund_sale()`

Экранные формы основные	**Дополнительные**	**Реализация (запрос)**	**Серверная часть**
Статистика по авто			
	Средняя цена	(02) Средняя стоимость автомобилей по каждому бренду	
	Рейтинг брендов	(07) Бренды с наибольшим числом продаж за последние 3 месяца	
Статистика по сотрудникам			
	Обновление статистики сотрудника		`update_employee_stat()`
	Рейтинг сотрудников	(11) Сотрудники с наибольшим числом продаж в текущем месяце	`v_employee_performance`
	Фильтр сотрудников	(08) Сотрудники, не совершавшие продажи последние 2 месяца	
Склад			
	Аналитика/отчёт		`get_inventory_json()`
Отчёты по продажам			
	История продаж		`v_sales_report`
	Аналитика по прибыли	(13) Средняя разница между первоначальной и продажной ценой по брендам	

Экранные формы основные	Дополнительные	Реализация (запрос)	Серверная часть
	Отчёт по месяцам	(04) Количество и выручка по месяцам за последний год	
	Топ продаж	(10) Три самых дорогих продажи в истории	

2 СКРИПТЫ

2.1 Серверная часть

Хранимые функции представлены на листинге 1.

```
-- ====================================
-- 1) Универсальная функция обновления статистики по одному сотруднику
-- ====================================
CREATE OR REPLACE FUNCTION update_employee_stat(p_employee_id INT)
   RETURNS VOID LANGUAGE plpgsql AS $$ BEGIN
UPDATE EmployeeStatistic emp_stat
SET total_sales_count = sub.cnt,
    total_sales_amount = sub.amt,
    last_sale_date = sub.last_dt
FROM (
        SELECT COUNT(Sale.sale_id) AS cnt,
            COALESCE(SUM(Sale.sale_price), 0) AS amt,
            MAX(Sale.sale_date) AS last_dt
        FROM Sale
        WHERE Sale.employee_id = p_employee_id
    ) sub
WHERE emp_stat.employee_id = p_employee_id;

INSERT INTO EmployeeStatistic(
        employee_id,
        total_sales_count,
        total_sales_amount,
        last_sale_date
    )
```

```
SELECT p_employee_id AS employee_id,
    sub.cnt,
    sub.amt,
    sub.last_dt
FROM (
        SELECT COUNT(sale_id) AS cnt,
            COALESCE(SUM(sale_price), 0) AS amt,
            MAX(sale_date) AS last_dt
        FROM Sale
        WHERE employee_id = p_employee_id
    ) sub ON CONFLICT (employee_id) DO
UPDATE
SET total_sales_count = EXCLUDED.total_sales_count,
    total_sales_amount = EXCLUDED.total_sales_amount,
    last_sale_date = EXCLUDED.last_sale_date;

END;

$$;

-- CALL update_employee_stat(1);
--
-- ====================================
-- 2) Оценка стоимости склада автомобилей в формате JSON
-- ====================================
CREATE OR REPLACE FUNCTION get_inventory_json() RETURNS JSONB LANGUAGE
   plpgsql AS $$
DECLARE report_data JSONB;

tag_summary JSONB;

tag_financials JSONB;

tag_brands JSONB;

tag_status_breakdown JSONB;

tag_analytics JSONB;

tag_total_cars INT;
```

```
63
64 tag_sold_cars INT;
65
66 BEGIN
67 SELECT jsonb_build_object(
68         'total_cars',
69         COUNT(*),
70         'available_cars',
71         COUNT(*) FILTER (
72             WHERE st.name = 'Available'
73         ),
74         'reserved_cars',
75         COUNT(*) FILTER (
76             WHERE st.name = 'Reserved'
77         ),
78         'sold_cars',
79         COUNT(*) FILTER (
80             WHERE st.name = 'Sold'
81         )
82     ),
83     jsonb_build_object(
84         'total_inventory_value',
85         COALESCE(
86             SUM(Car.price) FILTER (
87                 WHERE st.name = 'Available'
88             ),
89             0
90         ),
91         'total_reserved_value',
92         COALESCE(
93             SUM(Car.price) FILTER (
94                 WHERE st.name = 'Reserved'
95             ),
96             0
97         ),
98         'total_sold_value',
99         COALESCE(
100             SUM(Car.price) FILTER (
101                 WHERE st.name = 'Sold'
102             ),
```

```
            0
        ),
        'avg_available_price',
        COALESCE(
            ROUND(
                AVG(Car.price) FILTER (
                    WHERE st.name = 'Available'
                ),
                2
            ),
            0
        )
    ),
    COUNT(*),
    COUNT(*) FILTER (
        WHERE st.name = 'Sold'
    ) INTO tag_summary,
    tag_financials,
    tag_total_cars,
    tag_sold_cars
FROM Car
    JOIN StatusType st ON Car.status_id = st.status_id;

SELECT jsonb_agg(
        jsonb_build_object(
            'brand',
            brand,
            'car_count',
            car_count,
            'total_value',
            total_value,
            'avg_price',
            avg_price
        )
    ) INTO tag_brands
FROM (
        SELECT Model.brand,
            COUNT(*) AS car_count,
            SUM(Car.price) AS total_value,
            ROUND(AVG(Car.price), 2) AS avg_price
```

```
        FROM Car
            JOIN Model ON Car.model_id = Model.model_id
            JOIN StatusType st ON Car.status_id = st.status_id
        WHERE st.name = 'Available'
        GROUP BY Model.brand
    ) AS brand_data;

SELECT jsonb_agg(
        jsonb_build_object(
            'status',
            status_name,
            'count',
            car_count,
            'total_value',
            total_value,
            'min_price',
            min_price,
            'max_price',
            max_price
        )
    ) INTO tag_status_breakdown
FROM (
        SELECT st.name AS status_name,
            COUNT(*) AS car_count,
            COALESCE(SUM(Car.price), 0) AS total_value,
            COALESCE(MIN(Car.price), 0) AS min_price,
            COALESCE(MAX(Car.price), 0) AS max_price
        FROM Car
            JOIN StatusType st ON Car.status_id = st.status_id
        GROUP BY st.name
        ORDER BY st.name
    ) AS status_data;

tag_analytics := jsonb_build_object(
    'turnover_rate_percent',
    CASE
        WHEN tag_total_cars > 0 THEN ROUND(
            (tag_sold_cars::NUMERIC / tag_total_cars) * 100,
            2
        )
```

```
        ELSE 0
    END
);

report_data := jsonb_build_object(
    'generated_at',
    CURRENT_TIMESTAMP,
    'summary',
    tag_summary,
    'financials',
    tag_financials,
    'brands',
    COALESCE(tag_brands, '[]'::JSONB),
    'status_breakdown',
    COALESCE(tag_status_breakdown, '[]'::JSONB),
    'analytics',
    tag_analytics
);

RETURN report_data;

END;

$$;

-- SELECT get_inventory_json();
```

Листинг 1: Хранимые функции

Хранимые процедуры представлены на листинге 2.

```
-- ====================================
-- 1) Бронирование автомобиля
-- ====================================
CREATE OR REPLACE PROCEDURE reserve_car(
        car_id_param INT,
        client_id_param INT,
        employee_id_param INT
    ) LANGUAGE plpgsql AS $$
DECLARE reserved_status_id INT;

current_status TEXT;
```

```
12
13 BEGIN
14 SELECT StatusType.name INTO current_status
15 FROM Car
16     JOIN StatusType ON Car.status_id = StatusType.status_id
17 WHERE Car.car_id = car_id_param;
18
19 IF current_status = 'Sold' THEN RAISE EXCEPTION 'Cannot reserve sold car'
       ;
20
21 END IF;
22
23 SELECT status_id INTO reserved_status_id
24 FROM StatusType
25 WHERE name = 'Reserved'
26 LIMIT 1;
27
28 UPDATE Car
29 SET status_id = reserved_status_id
30 WHERE car_id = car_id_param;
31
32 END;
33
34 $$;
35
36 -- CALL reserve_car(3, 1, 1);
37 --
38 -- ====================================
39 -- 2) Возврат/отмена последней продажи автомобиля
40 -- ====================================
41 CREATE OR REPLACE PROCEDURE refund_sale(
42         IN p_car_id INT,
43         OUT refunded_sale_id INT,
44         OUT refunded_amount INT,
45         OUT refunded_car_id INT
46     ) LANGUAGE plpgsql AS $$
47 DECLARE v_sale RECORD;
48
49 BEGIN
50 SELECT * INTO v_sale
```

```
FROM Sale
WHERE car_id = p_car_id
ORDER BY sale_date DESC,
    sale_id DESC
LIMIT 1;

IF NOT FOUND THEN RAISE EXCEPTION 'No sale found for car %',
p_car_id;

END IF;

refunded_sale_id := v_sale.sale_id;

refunded_amount := v_sale.sale_price;

refunded_car_id := v_sale.car_id;

DELETE FROM Sale
WHERE sale_id = v_sale.sale_id;

PERFORM update_employee_stat(v_sale.employee_id);

UPDATE Car
SET status_id = (
        SELECT status_id
        FROM StatusType
        WHERE name = 'Available'
        LIMIT 1
    )
WHERE car_id = p_car_id;

END;

$$;

-- CALL refund_sale(1, NULL, NULL, NULL);
```

Листинг 2: Хранимые процедуры

Триггеры представлены на листинге 3.

```
-- ===================================
```

```
-- 1) Автоматически обновлять EmployeeStatistics при продаже
-- ===================================
CREATE OR REPLACE FUNCTION trg_update_stat_after_sale() RETURNS TRIGGER
   LANGUAGE plpgsql AS $$ BEGIN PERFORM update_employee_stat(NEW.
   employee_id);

RETURN NEW;

END;

$$;

CREATE TRIGGER trg_update_stat_after_sale
AFTER
INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION trg_update_stat_after_sale()
   ;

-- ===================================
-- 2) Автоматически устанавливать статус "Sold" при продаже автомобиля
-- ===================================
CREATE OR REPLACE FUNCTION set_car_status_to_sold() RETURNS TRIGGER AS $$
    BEGIN
UPDATE Car
SET status_id = (
        SELECT status_id
        FROM StatusType
        WHERE name = 'Sold'
        LIMIT 1
    )
WHERE car_id = NEW.car_id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_set_car_sold_after_sale
AFTER
INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION set_car_status_to_sold();
```

```
38
39 -- ====================================
40 -- 3) Ведение данных CarCondition при добавлении/удалении автомобиля
41 -- ====================================
42 CREATE OR REPLACE FUNCTION maintain_car_condition() RETURNS TRIGGER AS $$
       BEGIN IF (TG_OP = 'INSERT') THEN
43 INSERT INTO CarCondition (
44         car_id,
45         mileage,
46         mechanical_condition,
47         body_condition,
48         interior_condition,
49         inspection_date,
50         notes
51     )
52 VALUES (
53         NEW.car_id,
54         NULL,
55         NULL,
56         NULL,
57         NULL,
58         NULL,
59         NULL
60     );
61
62 RETURN NEW;
63
64 ELSIF (TG_OP = 'DELETE') THEN
65 DELETE FROM CarCondition
66 WHERE car_id = OLD.car_id;
67
68 RETURN OLD;
69
70 END IF;
71
72 RETURN NULL;
73
74 END;
75
76 $$ LANGUAGE plpgsql;
```

```
77
78 CREATE TRIGGER trg_maintain_car_condition
79 AFTER
80 INSERT
81     OR DELETE ON Car FOR EACH ROW EXECUTE FUNCTION maintain_car_condition
   ();
82
83 -- ====================================
84 -- 4) Запрет на продажу автомобиля, если его статус - продан
85 -- ====================================
86 CREATE OR REPLACE FUNCTION prevent_sale_of_sold_car() RETURNS TRIGGER AS
   $$
87 DECLARE current_status_name TEXT;
88
89 BEGIN
90 SELECT st.name INTO current_status_name
91 FROM Car
92     JOIN StatusType st ON Car.status_id = st.status_id
93 WHERE Car.car_id = NEW.car_id;
94
95 IF current_status_name = 'Sold' THEN RAISE EXCEPTION 'Cannot sell car %:
   it is already sold',
96 NEW.car_id;
97
98 END IF;
99
100 RETURN NEW;
101
102 END;
103
104 $$ LANGUAGE plpgsql;
105
106 CREATE TRIGGER trg_prevent_sale_of_sold_car BEFORE
107 INSERT ON Sale FOR EACH ROW EXECUTE FUNCTION prevent_sale_of_sold_car();
```

Листинг 3: Триггеры

Представления представлены на листинге 4.

```
1 -- ====================================
2 -- 1) Доступные автомобили с подробной информацией о модели
3 -- ====================================
```

```sql
CREATE OR REPLACE VIEW v_available_cars AS
SELECT Car.car_id,
    Model.brand,
    Model.model_name,
    Car.year,
    CarCondition.mileage,
    Car.color,
    Car.price,
    BodyType.name AS body_type,
    EngineType.name AS engine_type_name
FROM Car
    JOIN Model USING (model_id)
    JOIN BodyType ON Model.body_type_id = BodyType.body_type_id
    JOIN EngineType ON Model.engine_type_id = EngineType.engine_type_id
    JOIN StatusType ON Car.status_id = StatusType.status_id
    JOIN CarCondition ON CarCondition.car_id = Car.car_id
WHERE StatusType.name = 'Available';

SELECT *
FROM v_available_cars;

-- ===================================
-- 2) Полная история продаж с именами
-- ===================================
CREATE OR REPLACE VIEW v_sales_report AS
SELECT Sale.sale_id AS sale_id,
    Sale.sale_date AS sale_date,
    Sale.sale_price AS sale_amount,
    Model.brand || ' ' || Model.model_name AS automobile,
    CASE
        WHEN Client.client_type_id = 1 THEN Client.first_name || ' ' ||
   Client.last_name
        ELSE Client.company_name
    END AS customer,
    Employee.first_name || ' ' || Employee.last_name AS salesperson
FROM Sale
    JOIN Employee ON Sale.employee_id = Employee.employee_id
    JOIN Client ON Sale.client_id = Client.client_id
    JOIN Car ON Sale.car_id = Car.car_id
    JOIN Model ON Car.model_id = Model.model_id;
```

```sql

SELECT *
FROM v_sales_report;

-- ====================================
-- 3) Статистика сотрудников с указанием имени
-- ====================================
CREATE OR REPLACE VIEW v_employee_performance AS
SELECT Employee.employee_id,
    Employee.first_name,
    Employee.last_name,
    Position.name AS position,
    emp_stat.total_sales_count,
    emp_stat.total_sales_amount,
    emp_stat.last_sale_date
FROM Employee
    JOIN Position ON Employee.position_id = Position.position_id
    JOIN EmployeeStatistic emp_stat ON Employee.employee_id = emp_stat.
   employee_id;

SELECT *
FROM v_employee_performance;
```

Листинг 4: Представления

2.2 Клиентская часть

Запросы для экранных форм и отчетов представлены на листинге 5.

```sql
-- ====================================
-- 1) Автомобили, которые никогда не были проданы
-- ====================================
SELECT Car.car_id,
    Model.brand || ' ' || Model.model_name AS model,
    Car.year,
    Car.price,
    CarCondition.inspection_date
FROM Car
    JOIN Model ON Car.model_id = Model.model_id
    LEFT JOIN Sale ON Car.car_id = Sale.car_id
    JOIN CarCondition ON CarCondition.car_id = Car.car_id
WHERE Sale.sale_id IS NULL
```

```
ORDER BY Car.year DESC;

-- ====================================
-- 2) Средняя цена по бренду
-- ====================================
SELECT Model.brand,
    ROUND(AVG(Car.price), 2) AS average_price
FROM Car
    JOIN Model ON Car.model_id = Model.model_id
GROUP BY Model.brand
ORDER BY average_price DESC;

-- ====================================
-- 3) Средний пробег по типу кузова для автомобилей
-- ====================================
SELECT bt.name AS body_type,
    COUNT(*) AS cars_count,
    ROUND(AVG(cc.mileage), 0) AS avg_mileage
FROM Car
    JOIN Model ON Car.model_id = Model.model_id
    JOIN BodyType bt ON Model.body_type_id = bt.body_type_id
    JOIN CarCondition cc ON Car.car_id = cc.car_id
WHERE cc.mileage > 0
GROUP BY bt.name
HAVING COUNT(*) >= 1
ORDER BY avg_mileage DESC;

-- ====================================
-- 4) Ежемесячный отчет о продажах за прошлый год
-- ====================================
SELECT EXTRACT(
        YEAR
        FROM sale_date
    ) AS year,
    EXTRACT(
        MONTH
        FROM sale_date
    ) AS MONTH,
    TO_CHAR(sale_date, 'Month') AS month_name,
    COUNT(*) AS sales_count,
```

```
    SUM(sale_amount) AS total_revenue
FROM v_sales_report
WHERE sale_date >= CURRENT_DATE - INTERVAL '1 year'
GROUP BY year,
    MONTH,
    month_name
ORDER BY year,
    MONTH;

-- ====================================
-- 5) Клиенты, купившие Volkswagen
-- ====================================
SELECT Client.client_id,
    CASE
        WHEN Client.client_type_id = 1 THEN Client.first_name || ' ' ||
   Client.last_name
        ELSE Client.company_name
    END AS customer,
    Client.email,
    Client.address,
    Model.brand || ' ' || Model.model_name AS automobile
FROM Client
    JOIN Sale ON Client.client_id = Sale.client_id
    JOIN Car ON Sale.car_id = Car.car_id
    JOIN Model ON Model.model_id = Sale.car_id
WHERE EXISTS (
        SELECT *
        FROM Sale
            JOIN Car ON Sale.car_id = Car.car_id
            JOIN Model ON Car.model_id = Model.model_id
        WHERE Sale.client_id = Client.client_id
            AND Model.brand = 'Volkswagen'
    );

-- ====================================
-- 6) В наличии автомобили с пробегом более 50 000 км
-- ====================================
SELECT car_id,
    brand,
    model_name,
```

```
    year,
    mileage,
    color,
    price
FROM v_available_cars
WHERE mileage > 50000
ORDER BY mileage DESC;

-- ===================================
-- 7) Бренды с самыми высокими продажами за последние 3 месяца
-- ===================================
SELECT SPLIT_PART(automobile, ' ', 1) AS brand,
    COUNT(*) AS sales_count,
    SUM(sale_amount) AS total_revenue
FROM v_sales_report
WHERE sale_date >= CURRENT_DATE - INTERVAL '3 months'
GROUP BY SPLIT_PART(automobile, ' ', 1)
HAVING COUNT(*) >= 1
ORDER BY sales_count DESC,
    total_revenue DESC;

-- ===================================
-- 8) Сотрудники, у которых не было продаж за последние 2 месяца
-- ===================================
SELECT e.employee_id,
    e.first_name || ' ' || e.last_name AS full_name,
    p.name AS position
FROM Employee e
    JOIN Position p ON e.position_id = p.position_id
WHERE NOT EXISTS (
        SELECT 1
        FROM Sale s
        WHERE s.employee_id = e.employee_id
            AND s.sale_date >= CURRENT_DATE - INTERVAL '2 months'
    )
ORDER BY e.hire_date DESC;

-- ===================================
-- 9) Все контактные адреса электронной почты (клиенты + сотрудники)
-- ===================================
```

```sql
SELECT email
FROM Client
WHERE email IS NOT NULL
UNION
SELECT email
FROM Employee
WHERE email IS NOT NULL;

-- ====================================
-- 10) Топ-3 самых дорогих продаж
-- ====================================
SELECT *
FROM Sale
WHERE sale_price IN (
        SELECT DISTINCT sale_price
        FROM Sale
        ORDER BY sale_price DESC
        LIMIT 3
    )
ORDER BY sale_price DESC;

-- ====================================
-- 11) Самые прибыльные сотрудники за текущий месяц
-- ====================================
SELECT es.employee_id,
    Employee.first_name || ' ' || Employee.last_name AS salesperson,
    es.total_sales_amount,
    (
        SELECT COUNT(*)
        FROM Sale
        WHERE employee_id = es.employee_id
            AND EXTRACT(
                MONTH
                FROM sale_date
            ) = EXTRACT(
                MONTH
                FROM CURRENT_DATE
            )
            AND EXTRACT(
                YEAR
```

```
                FROM sale_date
            ) = EXTRACT(
                YEAR
                FROM CURRENT_DATE
            )
    ) AS sales
FROM Employee
    JOIN EmployeeStatistic es ON Employee.employee_id = es.employee_id
WHERE es.total_sales_amount > 0
ORDER BY sales DESC;

-- ====================================
-- 12) Автомобили по цене выше средней для своей марки
-- ====================================
SELECT ac.car_id,
    ac.brand,
    ac.model_name,
    ac.price,
    (
        SELECT ROUND(AVG(price), 2)
        FROM Car c2
            JOIN Model m2 ON c2.model_id = m2.model_id
        WHERE m2.brand = ac.brand
            AND c2.status_id = (
                SELECT status_id
                FROM StatusType
                WHERE name = 'Available'
            )
    ) AS avg_brand_price
FROM v_available_cars ac
WHERE ac.price > (
        SELECT AVG(c2.price)
        FROM Car c2
            JOIN Model m2 ON c2.model_id = m2.model_id
        WHERE m2.brand = ac.brand
            AND c2.status_id = (
                SELECT status_id
                FROM StatusType
                WHERE name = 'Available'
            )
```

```
213     );
214
215 -- ====================================
216 -- 13) Средняя разница в цене между изначальными и проданными автомобилям
    и по маркам
217 -- ====================================
218 SELECT Model.brand,
219     ROUND(AVG(Sale.sale_price - Car.price), 2) AS avg_price_difference,
220     COUNT(*) AS sales_count,
221     ROUND(AVG(Car.price), 2) AS avg_original_price,
222     ROUND(AVG(Sale.sale_price), 2) AS avg_sold_price
223 FROM Sale
224     JOIN Car ON Sale.car_id = Car.car_id
225     JOIN Model ON Car.model_id = Model.model_id
226 GROUP BY Model.brand
227 ORDER BY avg_price_difference DESC;
```

Листинг 5: Запросы

3 ПРИЛОЖЕНИЕ

3.1 Создание базы данных

Создание таблиц и индексов представлено на листинге 6.

```
1  -- ====================================
2  -- Справочные таблицы (словари)
3  -- ====================================
4  CREATE TABLE ClientType (
5      client_type_id SERIAL PRIMARY KEY,
6      name TEXT NOT NULL UNIQUE
7  );
8
9  CREATE TABLE BodyType (
10     body_type_id SERIAL PRIMARY KEY,
11     name TEXT NOT NULL UNIQUE
12 );
13
14 CREATE TABLE EngineType (
15     engine_type_id SERIAL PRIMARY KEY,
16     name TEXT NOT NULL UNIQUE
```

```
);

CREATE TABLE TransmissionType (
    transmission_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);

CREATE TABLE Position (
    position_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);

CREATE TABLE StatusType (
    status_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);

-- ====================================
-- Основные таблицы
-- ====================================
CREATE TABLE Client (
    client_id SERIAL PRIMARY KEY,
    client_type_id INT REFERENCES ClientType(client_type_id),
    first_name TEXT,
    last_name TEXT,
    company_name TEXT,
    phone VARCHAR(20),
    email TEXT,
    address TEXT
);

CREATE TABLE Employee (
    employee_id SERIAL PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    position_id INT REFERENCES Position(position_id),
    phone VARCHAR(20),
    email TEXT,
    hire_date DATE NOT NULL
);
```

```sql

CREATE TABLE Model (
    model_id SERIAL PRIMARY KEY,
    brand TEXT NOT NULL,
    model_name TEXT NOT NULL,
    body_type_id INT REFERENCES BodyType(body_type_id),
    engine_type_id INT REFERENCES EngineType(engine_type_id),
    transmission_id INT REFERENCES TransmissionType(transmission_id)
);

CREATE TABLE Car (
    car_id SERIAL PRIMARY KEY,
    model_id INT NOT NULL REFERENCES Model(model_id),
    vin VARCHAR(17) NOT NULL UNIQUE,
    year INT CHECK (
        year >= 1900
        AND year <= EXTRACT(
            YEAR
            FROM CURRENT_DATE
        ) + 1
    ),
    color TEXT,
    price INT CHECK (price >= 0),
    status_id INT REFERENCES StatusType(status_id)
);

CREATE TABLE CarCondition (
    car_id INT PRIMARY KEY REFERENCES Car(car_id) ON DELETE CASCADE,
    mileage INT CHECK (mileage >= 0),
    mechanical_condition TEXT,
    body_condition TEXT,
    interior_condition TEXT,
    inspection_date DATE,
    notes TEXT
);

CREATE TABLE Sale (
    sale_id SERIAL PRIMARY KEY,
    car_id INT NOT NULL REFERENCES Car(car_id),
    client_id INT NOT NULL REFERENCES Client(client_id),
```

```
    employee_id INT NOT NULL REFERENCES Employee(employee_id),
    sale_date DATE NOT NULL,
    sale_price INT CHECK (sale_price >= 0)
);

CREATE TABLE EmployeeStatistic (
    employee_id INT PRIMARY KEY REFERENCES Employee(employee_id) ON
   DELETE CASCADE,
    total_sales_count INT DEFAULT 0 CHECK (total_sales_count >= 0),
    total_sales_amount INT DEFAULT 0 CHECK (total_sales_amount >= 0),
    last_sale_date DATE
);

-- ====================================
-- Индексы
-- ====================================
CREATE INDEX idx_car_model ON Car(model_id);

CREATE INDEX idx_sale_employee ON Sale(employee_id);

CREATE INDEX idx_sale_client ON Sale(client_id);

CREATE INDEX idx_sale_date ON Sale(sale_date);

CREATE INDEX idx_car_status ON Car(status_id);

CREATE INDEX idx_model_brand_model ON Model(brand, model_name);

CREATE INDEX idx_car_year ON Car(year);
```

Листинг 6: Создание таблиц и индексов

Заполнение таблиц тестовыми данными представлено на листинге 7.

```
-- ====================================
-- Значения для справочных словарей
-- ====================================
INSERT INTO ClientType (name)
VALUES ('Individual'),
    ('Legal') ON CONFLICT DO NOTHING;

INSERT INTO StatusType (name)
```

```
VALUES ('Available'),
    ('Reserved'),
    ('Sold'),
    ('In Service') ON CONFLICT DO NOTHING;

INSERT INTO Position (name)
VALUES ('Salesperson'),
    ('Manager'),
    ('Director') ON CONFLICT DO NOTHING;

INSERT INTO BodyType (name)
VALUES ('Sedan'),
    ('Hatchback'),
    ('SUV'),
    ('CUV'),
    ('Coupe'),
    ('Wagon') ON CONFLICT DO NOTHING;

INSERT INTO EngineType (name)
VALUES ('Petrol'),
    ('Diesel'),
    ('Hybrid'),
    ('Electric') ON CONFLICT DO NOTHING;

INSERT INTO TransmissionType (name)
VALUES ('Manual'),
    ('Automatic') ON CONFLICT DO NOTHING;

-- ====================================
-- Сотрудники
-- ====================================
INSERT INTO Employee (
        first_name,
        last_name,
        position_id,
        phone,
        email,
        hire_date
    )
VALUES (
```

```
        'Ivan',
        'Petrov',
        1,
        '+7-900-111-2222',
        'ivan.petrov@example.com',
        '2019-06-15'
    ),
    (
        'Olga',
        'Sidorova',
        1,
        '+7-900-333-4444',
        'olga.sidorova@example.com',
        '2020-03-01'
    ),
    (
        'Mikhail',
        'Ivanov',
        2,
        '+7-900-555-6666',
        'm.ivanov@example.com',
        '2018-01-10'
    ),
    (
        'Filipp',
        'Kovalev',
        3,
        '+7-900-777-8888',
        'f.kovalev@example.com',
        '2021-05-20'
    ) ON CONFLICT DO NOTHING;

-- ====================================
-- Клиенты (физические и корпоративные)
-- ====================================
INSERT INTO Client (
        client_type_id,
        first_name,
        last_name,
        company_name,
```

```
        phone,
        email,
        address
    )
VALUES (
        1,
        'Petr',
        'Smirnov',
        NULL,
        '+7-912-000-1111',
        'p.smirnov@example.com',
        'Moscow, Tverskaya 1'
    ),
    (
        2,
        NULL,
        NULL,
        'TechSolutions LLC',
        '+7-912-222-3333',
        'sales@techsolutions.ru',
        'Saint Petersburg, Nevsky 5'
    ),
    (
        1,
        'Anna',
        'Kuznetsova',
        NULL,
        '+7-912-444-5555',
        'anna.k@example.com',
        'Kazan, Baumana 10'
    ),
    (
        1,
        'Dmitriy',
        'Orlov',
        NULL,
        '+7-912-666-7777',
        'd.orlov@example.com',
        'Novosibirsk, Lenina 20'
    ) ON CONFLICT DO NOTHING;
```

```
129
130 -- ====================================
131 -- Модели
132 -- ====================================
133 INSERT INTO Model (
134         brand,
135         model_name,
136         body_type_id,
137         engine_type_id,
138         transmission_id
139     )
140 VALUES ('Toyota', 'Camry', 1, 1, 2),
141     ('Porsche', 'Cayenne', 3, 2, 2),
142     ('Porsche', '911', 5, 1, 1),
143     ('BMW', 'M4', 1, 1, 2),
144     ('Tesla', 'Model 3', 4, 4, 2),
145     ('Audi', 'RS6', 6, 1, 2),
146     ('Skoda', 'Rapid', 1, 1, 1),
147     ('Volkswagen', 'Golf', 2, 1, 2),
148     ('Volkswagen', 'Touareg', 3, 1, 2) ON CONFLICT DO NOTHING;
149
150 -- ====================================
151 -- Автомобили
152 -- ====================================
153 INSERT INTO Car (
154         model_id,
155         vin,
156         year,
157         color,
158         price,
159         status_id
160     )
161 VALUES (
162         (
163             SELECT model_id
164             FROM model
165             WHERE brand = 'Toyota'
166                 AND model_name = 'Camry'
167             LIMIT 1
168         ), 'JTNBB46KX20000001', 2021, 'Black', 3000000, 1
```

```
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Porsche'
                AND model_name = '911'
            LIMIT 1
        ), '2T3RFREV9FW000002', 2024, 'White', 28000000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Porsche'
                AND model_name = '911'
            LIMIT 1
        ), 'HP3RFRGEV0W004022', 2022, 'Silver', 25900000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'BMW'
                AND model_name = 'M4'
            LIMIT 1
        ), 'WBA8E1C50GK000003', 2021, 'Blue', 9000000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Audi'
                AND model_name = 'RS6'
            LIMIT 1
        ), '1HGBH41JXMN109186', 2023, 'Gray', 12000000, 2
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Skoda'
                AND model_name = 'Rapid'
            LIMIT 1
        ), 'TRGLH34JZMU164106', 2017, 'White', 1050000, 4
```

```
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Tesla'
                AND model_name = 'Model 3'
            LIMIT 1
        ), 'TURBH31DSGN103158', 2023, 'White', 6000000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Volkswagen'
                AND model_name = 'Golf'
            LIMIT 1
        ), '3YGMH65JZOPU64146', 2018, 'Gray', 1800000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Volkswagen'
                AND model_name = 'Touareg'
            LIMIT 1
        ), 'RUDGL55JFOOH54421', 2019, 'Black', 3950000, 1
    ), (
        (
            SELECT model_id
            FROM model
            WHERE brand = 'Volkswagen'
                AND model_name = 'Touareg'
            LIMIT 1
        ), 'UEGGH45JDOFU39046', 2020, 'Gray', 4900000, 1
    ) ON CONFLICT DO NOTHING;

-- ==================================
-- Состояние автомобиля
-- ==================================
INSERT INTO CarCondition (
        car_id,
        mileage,
```

```
        mechanical_condition,
        body_condition,
        interior_condition,
        inspection_date,
        notes
    )
VALUES (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'JTNBB46KX20000001'
        ),
        50000,
        'Ok',
        'Replaced bumper',
        'Good',
        '2025-11-01',
        'Service done at 40000 km'
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = '2T3RFREV9FW000002'
        ),
        112,
        'Perfect',
        'Perfect',
        'Perfect',
        '2025-10-15',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'HP3RFRGEV0W004022'
        ),
        9900,
        'Perfect',
```

```
        'Perfect',
        'Perfect',
        '2025-08-22',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'WBA8E1C50GK000003'
        ),
        32000,
        'Engine goes tuk-tuk',
        'Good',
        'Good',
        '2025-09-20',
        'Engine needs check'
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'TRGLH34JZMU164106'
        ),
        119000,
        'Need technical service',
        'Good',
        'Need detailing',
        '2025-10-23',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'TURBH31DSGN103158'
        ),
        27000,
        'Strange sounds',
        'Replaced bumper',
```

```
        'Good',
        '2025-05-25',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = '3YGMH65JZOPU64146'
        ),
        85000,
        'Normal',
        'Good',
        'Need detailing',
        '2025-09-02',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'RUDGL55JFOOH54421'
        ),
        198000,
        'Good',
        'Good',
        'Good',
        '2025-08-29',
        NULL
    ),
    (
        (
            SELECT car_id
            FROM car
            WHERE vin = 'UEGGH45JDOFU39046'
        ),
        84000,
        'Good',
        'Good',
        'Not ideal',
```

```
        '2025-07-12',
        NULL
    ) ON CONFLICT (car_id) DO
UPDATE
SET mileage = EXCLUDED.mileage,
    mechanical_condition = EXCLUDED.mechanical_condition,
    body_condition = EXCLUDED.body_condition,
    interior_condition = EXCLUDED.interior_condition,
    inspection_date = EXCLUDED.inspection_date,
    notes = EXCLUDED.notes;

-- ====================================
-- Продажи
-- ====================================
INSERT INTO Sale (
        sale_date,
        car_id,
        client_id,
        employee_id,
        sale_price
    )
VALUES (
        '2025-11-02',
        (
            SELECT car_id
            FROM Car
            WHERE vin = 'JTNBB46KX20000001'
        ),
        (
            SELECT client_id
            FROM Client
            WHERE last_name = 'Smirnov'
            LIMIT 1
        ),(
            SELECT employee_id
            FROM Employee
            WHERE last_name = 'Petrov'
            LIMIT 1
        ), 2930000
    ), (
```

```
        '2025-10-10',(
            SELECT car_id
            FROM Car
            WHERE vin = 'WBA8E1C50GK000003'
        ),
        (
            SELECT client_id
            FROM Client
            WHERE last_name = 'Kuznetsova'
            LIMIT 1
        ),(
            SELECT employee_id
            FROM Employee
            WHERE last_name = 'Sidorova'
            LIMIT 1
        ), 9005000
    ), (
        '2025-10-25',(
            SELECT car_id
            FROM Car
            WHERE vin = 'RUDGL55JFOOH54421'
        ),
        (
            SELECT client_id
            FROM Client
            WHERE company_name = 'TechSolutions LLC'
            LIMIT 1
        ),(
            SELECT employee_id
            FROM Employee
            WHERE last_name = 'Petrov'
            LIMIT 1
        ), 3900000
    ), (
        '2025-11-01',(
            SELECT car_id
            FROM Car
            WHERE vin = 'TRGLH34JZMU164106'
        ),
        (
```

```
            SELECT client_id
            FROM Client
            WHERE company_name = 'TechSolutions LLC'
            LIMIT 1
        ),(
            SELECT employee_id
            FROM Employee
            WHERE last_name = 'Ivanov'
            LIMIT 1
        ), 950000
    ) ON CONFLICT DO NOTHING;
```

Листинг 7: Заполнение тестовыми данными

Удаление всех созданных объектов представлено на листинге 8.

```
-- ====================================
-- Удалить процедуры
-- ====================================
DROP PROCEDURE IF EXISTS reserve_car(INT, INT, INT);

DROP PROCEDURE IF EXISTS refund_sale(INT, OUT INT, OUT INT, OUT INT);

-- ====================================
-- Удалить триггеры
-- ====================================
DROP TRIGGER IF EXISTS trg_update_stat_after_sale ON Sale;

DROP TRIGGER IF EXISTS trg_set_car_sold_after_sale ON Sale;

DROP TRIGGER IF EXISTS trg_maintain_car_condition ON Car;

DROP TRIGGER IF EXISTS trg_prevent_sale_of_sold_car ON Sale;

-- ====================================
-- Функции удаления
-- ====================================
DROP FUNCTION IF EXISTS update_employee_stat(INT);

DROP FUNCTION IF EXISTS get_inventory_json();

DROP FUNCTION IF EXISTS set_car_sold();
```

```
27
28 DROP FUNCTION IF EXISTS maintain_car_condition();
29
30 DROP FUNCTION IF EXISTS prevent_sale_of_sold_car();
31
32 -- ====================================
33 -- Удалить представления
34 -- ====================================
35 DROP VIEW IF EXISTS v_available_cars;
36
37 DROP VIEW IF EXISTS v_sales_report;
38
39 DROP VIEW IF EXISTS v_employee_performance;
40
41 -- ====================================
42 -- Удалить таблицы
43 -- ====================================
44 DROP TABLE IF EXISTS EmployeeStatistic CASCADE;
45
46 DROP TABLE IF EXISTS Sale CASCADE;
47
48 DROP TABLE IF EXISTS CarCondition CASCADE;
49
50 DROP TABLE IF EXISTS Car CASCADE;
51
52 DROP TABLE IF EXISTS Model CASCADE;
53
54 DROP TABLE IF EXISTS Client CASCADE;
55
56 DROP TABLE IF EXISTS Employee CASCADE;
57
58 DROP TABLE IF EXISTS StatusType CASCADE;
59
60 DROP TABLE IF EXISTS Position CASCADE;
61
62 DROP TABLE IF EXISTS TransmissionType CASCADE;
63
64 DROP TABLE IF EXISTS EngineType CASCADE;
65
66 DROP TABLE IF EXISTS BodyType CASCADE;
```

```
67
68 DROP TABLE IF EXISTS ClientType CASCADE;
```

Листинг 8: Удаление созданных объектов