Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего

образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Лабораторная работа №5 по дисциплине

“Базы данных”

Студенты: Терещенко Никита Юрьевич, Шокина Анна Сергеевна

Факультет: ПИиКТ

Группа: P33212

Преподаватель: Шешуков Дмитрий Михайлович

Санкт-Петербург, 2023

Задание:

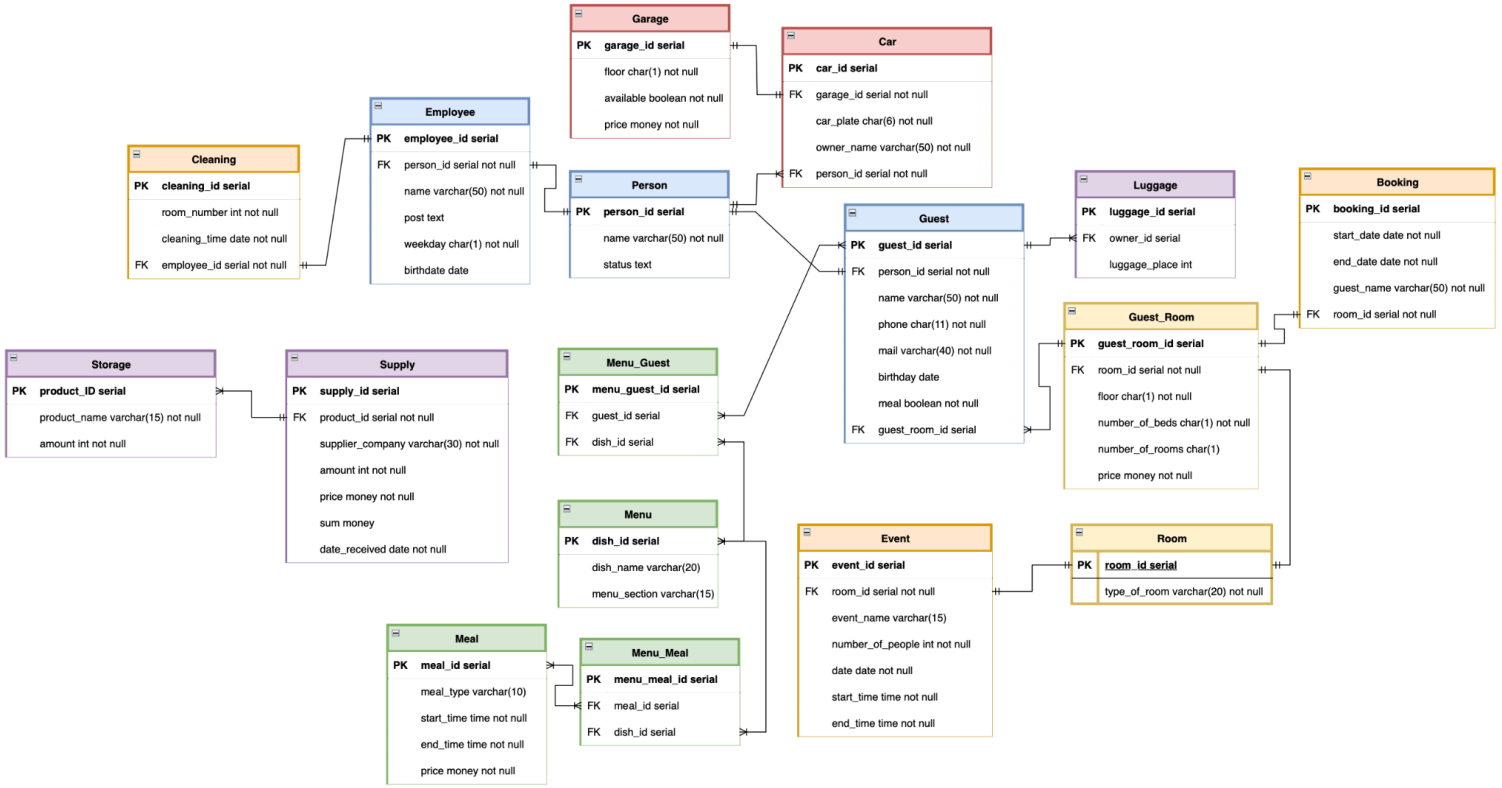
- Добавить в ранее созданную базу данных (лр №4) триггеры для обеспечения комплексных ограничений целостности.
- Реализовать функции и процедуры на основе описания бизнес-процессов, определенных при описании предметной области (лр №1). Должна быть обеспечена проверка корректности вводимых данных для созданных функций и процедур.
- Необходимо произвести анализ использования созданной базы данных, выявить наиболее часто используемые объекты базы данных, виды запросов к ним. Результаты должны быть представлены в виде текстового описания.
- На основании полученного описания требуется создать подходящие индексы и доказать, что они будут полезны для представленных в описании случаев использования базы данных.

Описание предметной области

В качестве предметной области, с которой мы в дальнейшем будем работать и которую будем развивать, мы выбрали сферу услуг, а именно — отели, оказывающиеся очень полезными во время путешествий. Обычно отель предлагает различные категории номеров: от стандартных одноместных до президентских люксов. Каждое такое “средство размещения” людей имеет собственную структуру и организацию и для его налаженной работы необходимо грамотное управление. Управление отелем включает в себя бронирование номеров, проверку гостей, налаживание графика уборки номеров, учёт финансовых операций и т.д.
Создание базы данных для такой обширной категории в сфере услуг значительно облегчит работу управляющему персоналу и положительно скажется на их эффективности. Например, можно хранить информацию о гостях, включая их личные данные, контактную информацию, историю пребывания(дата заселения/выселения), а также информацию о доступности номеров, тарифных планах, бронировании, платежах. Вдобавок ко всему можно также

хранить информацию о персонале: их рабочие графики, должности, выплаты, учёт поставщиков и т.д.
Таким образом, база данных отеля позволяет эффективно управлять ресурсами, автоматизировать некоторые процессы и улучшать качество обслуживания гостей.

Даталогическая модель:



Триггеры:

Триггер для автоматического обновления статуса доступности гаража при добавлении или удалении автомобиля:

```
CREATE OR REPLACE FUNCTION update_garage_availability()
RETURNS TRIGGER AS $$
BEGIN
        IF TG_OP = ‘INSERT’ THEN
                UPDATE garage SET available = false WHERE garage_id = NEW.garage_id;
        ELSE IF TG_OP = ‘DELETE’ THEN
                UPDATE garage set available = true WHERE garage_id = OLD.garage_id;
        END IF;
        RETURN NEW;
```

```
END;
$$ LANGUAGE plpsql;

CREATE TRIGGER after_car_insert_or_delete
AFTER INSERT OR DELETE on car
FOR EACH ROW
EXECUTE FUNCTION update_garage_availability()
```

Работа триггера:
Теперь при добавлении в новой записи в car триггер after_car_insert_or_delete будет срабатывать и обновлять доступность места в гараже. Например, при добавлении машины с garage_id = 2 в таблицу car значение available в таблице garage поменяется с ‘true’ на ‘false’

```
studs=> insert into car(garage_id, car_plate,owner_name, person_id) values (2, 'YUW192','Jon Doe', 1);
INSERT 0 1
studs=> select available from garage where garage_id = 2;
 available
-----------
 f
(1 строка)
```

Триггер для проверки даты начала и окончания бронирования номера:

```
CREATE OR REPLACE FUNCTION validate_booking_dates()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.start_date < CURRENT_DATE THEN
    RAISE EXCEPTION 'Impossible to book for past dates';
  END IF;

  IF NEW.start_date > NEW.end_date THEN
    RAISE EXCEPTION 'End date have to be after start date';
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_booking
BEFORE INSERT ON booking
FOR EACH ROW
EXECUTE FUNCTION validate_booking_dates();
```

Работа триггера:

В данном случае я пытаюсь забронировать номер, где дата начала бронирования больше, чем дата окончания бронирования, триггер сработает и выдаст исключение с соответствующим сообщением.

```
studs=> insert into booking(start_date, end_date, guest_name, room_id) values ('2023-12-29', '2023-12-18', 'Bob Johnson', 3);
ERROR:  End date have to be after start date
КОНТЕКСТ:  PL/pgSQL function validate_booking_dates() line 8 at RAISE
```

Триггер для автоматического обновления суммы в таблице supply при добавлении поставки:

```
CREATE OR REPLACE FUNCTION update_supply_sum()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE supply SET sum = NEW.amount * NEW.price WHERE supply_id = NEW.supply_id;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_supply_insert
AFTER INSERT ON supply
FOR EACH ROW
EXECUTE FUNCTION update_supply_sum();
```

Работа триггера:
В случае вставки значений в supply триггер сработает и значение sum для поставки будет обновлено до значения 25,00 * 100.

```
studs=> insert into supply(product_id, supplier_company, amount, price, date_received) values (3,'test Supplier', 100, 25, '2023-0
8-01');
INSERT 0 1
studs=> select * from supply;
 supply_id | product_id | supplier_company | amount |  price  |    sum     | date_received
-----------+------------+------------------+--------+---------+------------+---------------
         6 |          3 | test Supplier    |    100 | 25,00 ₽ | 2 500,00 ₽ | 2023-08-01
(1 строка)
```

Процедуры

Процедура для бронирования комнаты

```
CREATE OR REPLACE PROCEDURE Book_room(guest_name varchar(50), guest_room_id int,
start_date date, end_date date)
LANGUAGE plpgsql
AS $$
DECLARE
        room_is_occupied BOOLEAN;
BEGIN
        room_is_occupied := FALSE;

        SELECT EXISTS (
```

```
		SELECT 1
		FROM booking
		WHERE
			guest_room_id = guest_room_id AND
			start_date <= start_date AND
			end_date >= end_date
	) INTO room_is_occupied;
	IF room_is_occupied THEN
		RAISE EXCEPTION 'Комната уже занята в указанный период времени';
	ELSE
		INSERT INTO booking (start_date, end_date, guest_name, room_id) VALUES
	(start_date,end_date,guest_name,guest_room_id);
	RAISE NOTICE 'Комната забронирована успешно';
	END IF;
END;
$$;


call Book_room('John Cena',2,'2023-04-20','2023-04-30');
```

Процедура для планирования графика уборки:

```
CREATE OR REPLACE PROCEDURE Schedule_cleaning(room_number int, cleaning_time date,
employee_id int)
LANGUAGE plpgsql
AS $$
DECLARE
	room_is_cleaning BOOLEAN;
BEGIN
	room_is_cleaning := FALSE;

	SELECT EXISTS (
		SELECT 1
		FROM cleaning
		WHERE
			room_number = room_number AND
			cleaning_time = cleaning_time
	) INTO room_is_cleaning;
	IF room_is_cleaning THEN
		RAISE EXCEPTION 'В этой комнате в этот день убирается другой сотрудник';
	ELSE
```

```
            INSERT INTO cleaning (room_number, cleaning_time, employee_id) VALUES
        (room_number, cleaning_time, employee_id);
        RAISE NOTICE 'Уборка успешно добавлена в расписание';
        END IF;
END;
$$;


call Schedule_cleaning(202, '2023-05-01', 5);
```

Процедура для приема поставки:

```
CREATE OR REPLACE PROCEDURE add_supply(product_id int, supplier_company varchar, amount int,
price money, sum money, date_received date)
LANGUAGE plpgsql
AS $$
BEGIN

        INSERT INTO supply (product_id, supplier_company, amount, price, sum, date_received)
        VALUES (product_id, supplier_company, amount, price, sum, date_received);
        UPDATE storage
          SET amount = amount + p_amount
          WHERE product_id = p_product_id;
        RAISE NOTICE 'Поставка принята';
        COMMIT;
END;
$$;


call add_supply(1, 'ABC Supplier', 10, 1.50, 15.0, '2023-06-02');
```

Описание наиболее часто используемых сценариев при работе с базой данных:

Управление парковкой:
SQL-запрос: SELECT * FROM garage WHERE available = true;

Учет гостевых платежей:
SQL-запрос: SELECT SUM(price) FROM menu_guest WHERE guest_id = 1;

Индексы:

Индекс для поиска свободных мест в гараже по столбцу available в таблице garage:

```
create INDEX idx_garage_available ON garage(available);
```

```
studs=> create INDEX idx_garage_available ON garage(available);
CREATE INDEX
studs=> EXPLAIN ANALYZE SELECT * from garage where available=true;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on garage  (cost=0.00..1.16 rows=8 width=25) (actual time=0.011..0.013 rows=10 loops=1)
   Filter: available
   Rows Removed by Filter: 6
 Planning Time: 0.112 ms
 Execution Time: 0.027 ms
(5 строк)

studs=> drop index idx_garage_available;
DROP INDEX
studs=> EXPLAIN ANALYZE SELECT * from garage where available=true;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on garage  (cost=0.00..1.16 rows=8 width=25) (actual time=0.012..0.014 rows=10 loops=1)
   Filter: available
   Rows Removed by Filter: 6
 Planning Time: 0.197 ms
 Execution Time: 0.026 ms
```

Индекс для поиска названия блюд, относящихся к определенному типу блюд в таблице menu:

create INDEX idx_menu_section ON menu(menu_section);

```
studs=> create INDEX idx_menu_section ON menu(menu_section);
CREATE INDEX
studs=> explain analyze select * from menu where menu_section='Dessert';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on menu  (cost=0.00..1.44 rows=1 width=110) (actual time=0.012..0.016 rows=13 loops=1)
   Filter: ((menu_section)::text = 'Dessert'::text)
   Rows Removed by Filter: 22
 Planning Time: 0.207 ms
 Execution Time: 0.032 ms
(5 строк)

studs=> drop index idx_menu_section;
DROP INDEX
studs=> explain analyze select * from menu where menu_section='Dessert';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on menu  (cost=0.00..1.44 rows=1 width=110) (actual time=0.012..0.016 rows=13 loops=1)
   Filter: ((menu_section)::text = 'Dessert'::text)
   Rows Removed by Filter: 22
 Planning Time: 0.124 ms
 Execution Time: 0.033 ms
(5 строк)
```

Вывод:

Таким образом, были добавлены триггеры, функции и процедуры на основе описания бизнес-процессов, проведён анализ использования созданной БД и созданы индексы на основании данного анализа.