

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет прикладной информатики
Образовательная программа Мобильные и сетевые технологии
Направление подготовки 09.03.03 Мобильные и сетевые технологии

ОТЧЕТ по Лабораторной работе № 3
по дисциплине «Проектирование и реализация баз данных»

Обучающийся: Шукалов Андрей Денисович
Преподаватель: Говорова Марина Михайловна

Санкт-Петербург, 2025

Описание задания

Цель работы

Овладеть практическими навыками создания таблиц базы данных PostgreSQL 1X, заполнения их рабочими данными, резервного копирования и восстановления БД.

Практическое задание

1. Создать базу данных с использованием pgAdmin 4 (согласно индивидуальному заданию).
2. Создать схему в составе базы данных.
3. Создать таблицы базы данных.
4. Установить ограничения на данные: Primary Key, Unique, Check, Foreign Key.
5. Заполнить таблицы БД рабочими данными.
6. Создать резервную копию БД.
7. Восстановить БД.

Схема инфологической модели БД ЛР 2 (IDEF1X)

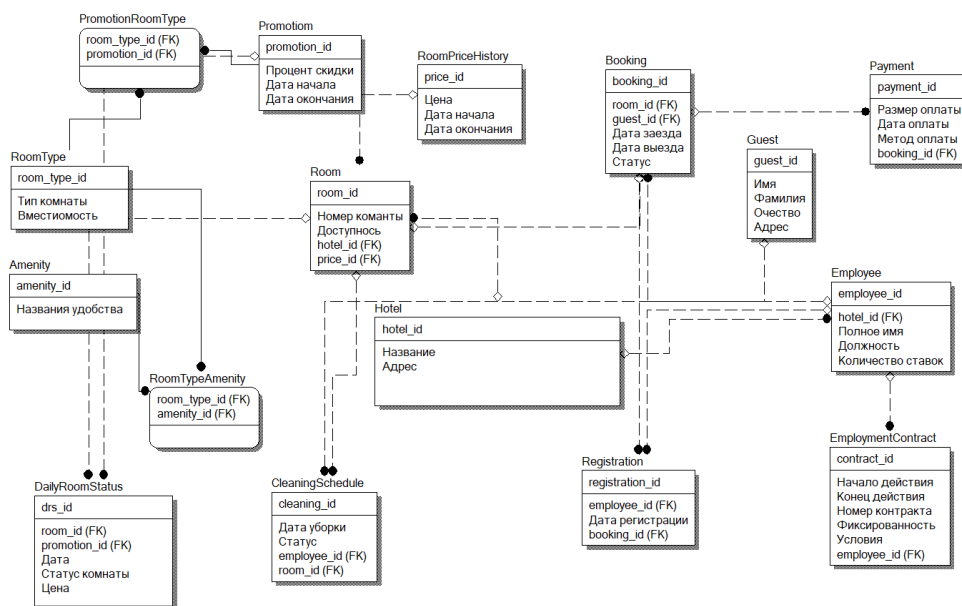


Рис. 1: Схема инфологической модели БД Отель

Ход работы

1 Создание базы данных

В pgAdmin4 создана база данных `hotel_db` согласно варианту 1 из лабораторной 2 БД "ОТЕЛЬ".

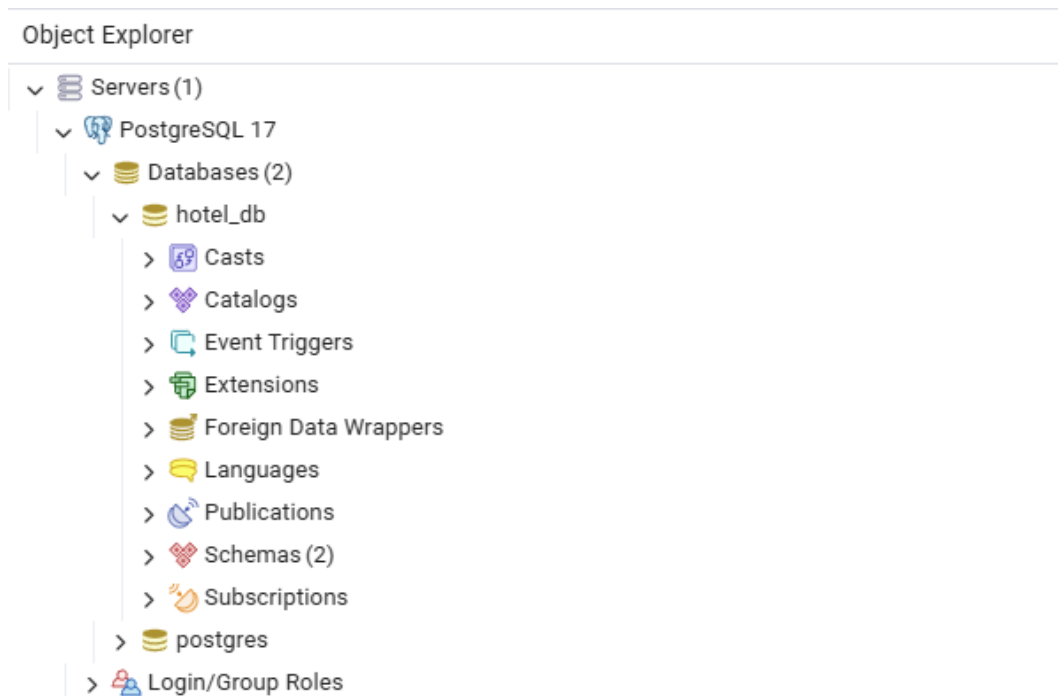


Рис. 2: База данных `hotel_db`

2 Создание схемы в составе базы данных

В составе этой базы данных была создана схема `hotel_schema`.

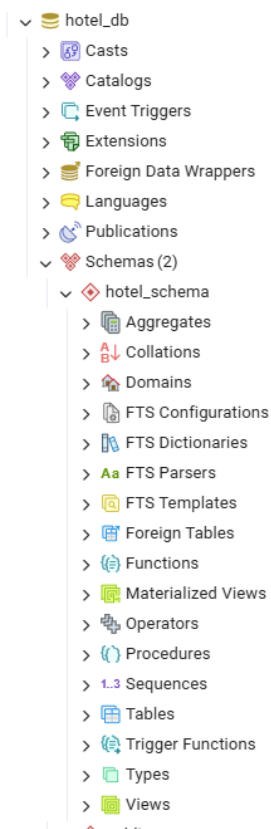


Рис. 3: Схема `hotel_schema`

3 Создание таблиц базы данных и установки ограничения

Создание таблиц производилось через SQL Editor.

3.1 Отель

Создание таблицы *Hotel*:

```
CREATE TABLE Hotel (  
    hotel_id SERIAL PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    address VARCHAR(200) NOT NULL  
);
```

hotel_id – первичный ключ, serial.

name – название отеля, строка, не пустая.

address – адрес отеля, строка, не пустая.

3.2 Тип комнаты

Создание таблицы *RoomType*:

```
CREATE TABLE RoomType (  
    room_type_id SERIAL PRIMARY KEY,  
    type_name VARCHAR(10) CHECK (type_name IN ('стандарт', 'люкс')),  
    capacity INT NOT NULL CHECK (capacity > 0)  
);
```

room_type_id – первичный ключ, serial.

type_name – название типа комнаты, строка, имеет значение 'стандарт' или 'люкс'.

capacity – вместимость комнаты, int, не пустое, больше 0.

3.3 Номер в отеле

Создание таблицы *Room*:

```
CREATE TABLE Room (  
    room_id SERIAL PRIMARY KEY,  
    hotel_id INT NOT NULL REFERENCES Hotel(hotel_id) ON DELETE CASCADE,  
    room_number INT NOT NULL,  
    room_type_id INT NOT NULL REFERENCES RoomType(room_type_id),  
    is_available BOOLEAN DEFAULT TRUE,  
    UNIQUE(hotel_id, room_number)  
);
```

room_id – первичный ключ, serial.

hotel_id – идентификатор отеля, где находится комната, int, не пустое, foreign key связанный с таблицей *Hotel*, удаляется при удалении соответствующего отеля.

room_number – номер комнаты в отеле, число, не пустое.

room_type_id – идентификатор типа комнаты, int, не пустое, foreign key связанный с соответствующей *RoomType*.

is_available – свободна ли комната, bool, изначально заполнена true.

UNIQUE – запрещаем наличие комнат с одинаковым номером в одном отеле.

3.4 Акции

Создание таблицы для акций:

```
CREATE TABLE Promotion (  
    promotion_id SERIAL PRIMARY KEY,  
    percent INT NOT NULL CHECK (percent > 0 AND percent <= 100),  
    start_date DATE NOT NULL,  
    end_date DATE NOT NULL,  
    CHECK (end_date > start_date)  
);
```

`promotion_id` – первичный ключ, serial.

`percent` – процент скидки, число, не пустое, больше 0 и не превышает 100.

`start_date` – начало действия скидки, не пустое.

`end_date` – окончания действия скидки, не пустое.

Ограничение: `start_date` идёт до `end_date`.

3.5 История цен на номер

Создание таблицы для истории цен на номер `RoomPriceHistory`:

```
CREATE TABLE RoomPriceHistory (  
    price_id SERIAL PRIMARY KEY,  
    room_type_id INT NOT NULL REFERENCES RoomType(room_type_id) ON DELETE CASCADE,  
    price NUMERIC(10,2) NOT NULL CHECK (price >= 0),  
    start_date DATE NOT NULL,  
    end_date DATE  
);
```

`price_id` – первичный ключ, serial.

`room_type_id` – foreign key, int, не пустое, ссылается на соответствующий тип комнаты.

`price` – стоимость номера в соответствующие даты, int, не пустое, больше или равно 0.

`start_date` – начало промежутка времени действия данной цены, date, не пустое.

`end_date` – конец промежутка времени действия данной цены, date, может быть пустым, если действие всё ещё идёт.

3.6 Удобства

Создание таблиц для удобств `Amenity`:

```
CREATE TABLE Amenity (  
    amenity_id SERIAL PRIMARY KEY,  
    name VARCHAR(100) NOT NULL UNIQUE  
);
```

`amenity_id` – первичный ключ, serial.

`name` – название удобства, строка, не пустое, не совпадает с другими удобствами.

3.7 Связь удобств и типов номеров

Создание таблицы:

```
CREATE TABLE RoomTypeAmenity (  
    room_type_id INT NOT NULL REFERENCES RoomType(room_type_id),  
    amenity_id INT NOT NULL REFERENCES Amenity(amenity_id) ON DELETE CASCADE,  
    PRIMARY KEY (room_type_id, amenity_id)  
);
```

Связываем `RoomType` и `Amenity` делая первичным ключом пару из их идентификаторов.

3.8 Постоялец

Создание таблицы для постояльцев **Guest**:

```
CREATE TABLE Guest (  
    guest_id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,  
    last_name VARCHAR(50) NOT NULL,  
    patronymic VARCHAR(50),  
    address VARCHAR(200)  
);
```

Имеет первичный ключ `guest_id`, имя и фамилию – непустые строки, отчество и адрес постоянного проживания – строки.

3.9 Бронирование

Создание таблицы для бронирования **Booking**:

```
CREATE TABLE Booking (  
    booking_id SERIAL PRIMARY KEY,  
    room_id INT NOT NULL REFERENCES Room(room_id),  
    guest_id INT NOT NULL REFERENCES Guest(guest_id),  
    checkin_date DATE NOT NULL,  
    checkout_date DATE NOT NULL,  
    status VARCHAR(20) CHECK (status IN ('booked', 'checked_in', 'checked_out', 'cancelled')),  
    promotion_id INT REFERENCES Promotion(promotion_id)  
);
```

`booking_id` – первичный ключ, serial.

`room_id` – foreign key, не пустой, ссылается на соответствующую комнату в отеле.

`guest_id` – foreign key, не пустой, ссылается на соответствующего гостя.

`checkin_date` – дата заселения, date, не пустое.

`checkout_date` – дата выселения, date, не пустое.

`status` – статус бронирования, строка, может принимать значения: 'booked' – забронировано, 'checked_in' – гость заселён, 'checked_out' – гость выехал, 'cancelled' – бронирование отменено.

`promotion_id` – foreign key, ссылается на соответствующую скидку или пустое, если скидки нет.

3.10 Сотрудник

Создание таблицы для сотрудников **Employee**:

```
CREATE TABLE Employee (  
    employee_id SERIAL PRIMARY KEY,  
    hotel_id INT NOT NULL REFERENCES Hotel(hotel_id) ON DELETE CASCADE,  
    full_name VARCHAR(150) NOT NULL,  
    position VARCHAR(50) NOT NULL,  
    rate_count INT NOT NULL CHECK (rate_count >= 1)  
);
```

`employee_id` – первичный ключ, serial.

`hotel_id` – foreign key, число, не пустое, ссылается на соответствующий инстанс отеля, где работает сотрудник, при удалении отеля также удаляется.

`full_name` – полное имя сотрудника, строка, не пустая.

`position` – занимаемая должность, строка, не пустая.

`rate_count` – количество ставок, число, не пустое, хотя бы 1.

3.11 Регистрация

Создание таблицы для регистрации при заезде Registration:

```
CREATE TABLE Registration (  
    registration_id SERIAL PRIMARY KEY,  
    booking_id INT NOT NULL REFERENCES Booking(booking_id),  
    employee_id INT NOT NULL REFERENCES Employee(employee_id),  
    registration_date DATE DEFAULT CURRENT_DATE  
);
```

registration_id – первичный ключ, serial.

booking_id – foreign key, число, не пустое, ссылается на соответствующий инстанс Booking.

employee_id – foreign key, число, не пустое, ссылается на соответствующего сотрудника.

registration_date – дата заселения, date, стандартно – текущая дата.

3.12 Договоры сотрудников

Создание таблицы для договоров сотрудников EmploymentContract:

```
CREATE TABLE EmploymentContract (  
    contract_id SERIAL PRIMARY KEY,  
    employee_id INT NOT NULL REFERENCES Employee(employee_id) ON DELETE CASCADE,  
    start_date DATE NOT NULL,  
    end_date DATE,  
    contract_number VARCHAR(100) UNIQUE,  
    is_fixed_term BOOLEAN DEFAULT FALSE,  
    terms TEXT  
);
```

contract_id – первичный ключ, serial.

employee_id – foreign key, число, не пустое, ссылается на соответствующего сотрудника, удаляется при его удалении.

start_date – начало работы сотрудника, date, не пустое.

end_date – конец работы сотрудника, date.

3.13 График уборки номеров

Создание таблицы для графика уборки номеров CleaningSchedule:

```
CREATE TABLE CleaningSchedule (  
    cleaning_id SERIAL PRIMARY KEY,  
    room_id INT NOT NULL REFERENCES Room(room_id) ON DELETE CASCADE,  
    cleaning_date DATE NOT NULL,  
    status VARCHAR(20) NOT NULL CHECK (status IN ('убран', 'не убран')),  
    maid_id INT NOT NULL REFERENCES Employee(employee_id),  
    UNIQUE (room_id, cleaning_date)  
);
```

cleaning_id – первичный ключ, serial.

room_id – foreign key, число, не пустое, ссылается на соответствующую комнату, удаляется при удалении комнаты.

cleaning_date – дата уборки, не пустая.

status – статус уборки, строка, не пустая, имеет значение 'убран' или 'не убран'.

main_id – foreign key сотрудник, который выполняет уборку.

3.14 Оплата проживания

Создание таблицы для оплаты проживания:

```
CREATE TABLE Payment (
    payment_id SERIAL PRIMARY KEY,
    booking_id INT NOT NULL REFERENCES Booking(booking_id),
    amount NUMERIC(10,2) NOT NULL CHECK (amount >= 0),
    payment_date DATE DEFAULT CURRENT_DATE,
    method VARCHAR(50) CHECK (method IN ('cash', 'card'))
);
```

payment_id – первичный ключ, serial.

booking_id – foreign key соответствующий номер бронирования.

amount – размер оплаты, число, десять знаков до запятой, два после, не пустое, должно быть неотрицательным.

payment_date – дата оплаты, date, стандартно – текущая дата.

method – метод оплаты, строка, должно иметь значение 'cash' или 'card' (наличными или по карте).

3.15 Связь типов номеров и акций

Создание связи для типов номеров и акций PromotionRoomType:

```
CREATE TABLE PromotionRoomType (
    promotion_id INT NOT NULL REFERENCES Promotion(promotion_id) ON DELETE CASCADE,
    room_type_id INT NOT NULL REFERENCES RoomType(room_type_id) ON DELETE CASCADE,
    PRIMARY KEY (promotion_id, room_type_id)
);
```

3.16 Статус комнаты в конкретный день

Создание таблицы для удобного подсчёта цен DailyRoomStatus:

```
CREATE TABLE DailyRoomStatus (
    drs_id SERIAL PRIMARY KEY,
    room_id INT NOT NULL REFERENCES Room(room_id) ON DELETE CASCADE,
    status_date DATE NOT NULL,
    status VARCHAR(20) NOT NULL CHECK (status IN ('free', 'booked', 'occupied', 'out_of_service')),
    price_for_date NUMERIC(10,2),
    promotion_id INT REFERENCES Promotion(promotion_id),
    UNIQUE(room_id, status_date)
);
```

drs_id – первичный ключ, serial.

room_id – foreign key, не пустой, ссылается на конкретную комнату, удаляется при удалении комнаты.

status_date – дата текущего состояния, не пустая.

status – строка, не пустая, имеет значение: 'free' – свободна, 'booked' – забронирована, 'occupied' – занята, 'out_of_service' – недоступна.

price_for_date – текущая цена на данную дату.

promotion_id – foreign key, ссылается на действующую скидку или пустое. Не может быть повторяющихся пар комната, текущее состояние.

4 Заполнение таблиц тестовыми данными

Заполним таблицы тестовыми данными.

Пример INSERT:

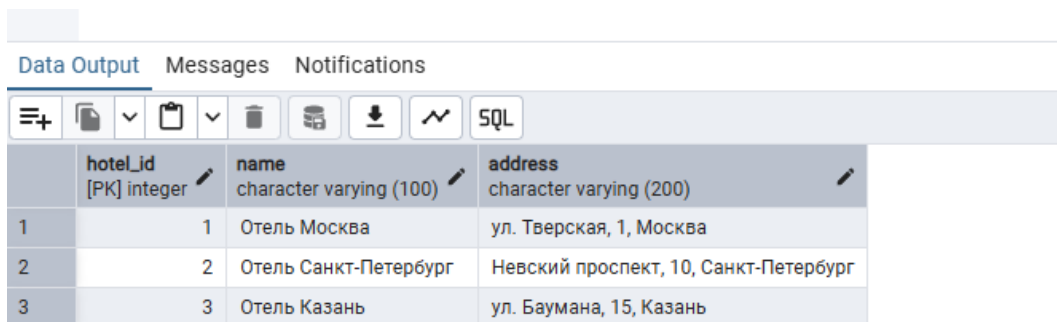
```
INSERT INTO Hotel (name, address) VALUES
('Отель Москва', 'ул. Тверская, 1, Москва'),
('Отель Санкт-Петербург', 'Невский проспект, 10, Санкт-Петербург'),
('Отель Казань', 'ул. Баумана, 15, Казань');
```

```
INSERT INTO Room (hotel_id, room_number, room_type_id) VALUES
(1, '101', 1),
(1, '102', 2),
(1, '201', 3),
(1, '202', 4);
```

```
INSERT INTO Room (hotel_id, room_number, room_type_id) VALUES
(2, '101', 2),
(2, '102', 2),
(2, '201', 3);
```

```
INSERT INTO Room (hotel_id, room_number, room_type_id) VALUES
(3, '101', 1),
(3, '102', 2),
(3, '201', 3);
```

Примеры выводов SELECT * FROM <имя таблицы>:



The screenshot shows a database management interface with three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with three columns: 'hotel_id' (integer, primary key), 'name', and 'address' (both character varying). The table contains three rows of data representing different hotels.

	hotel_id [PK] integer	name character varying (100)	address character varying (200)
1	1	Отель Москва	ул. Тверская, 1, Москва
2	2	Отель Санкт-Петербург	Невский проспект, 10, Санкт-Петербург
3	3	Отель Казань	ул. Баумана, 15, Казань

Рис. 4: SELECT * FROM Hotel

	room_type_id [PK] integer	amenity_id [PK] integer
1	1	1
2	1	2
3	2	1
4	2	2
5	3	1
6	3	3
7	3	4
8	4	1
9	4	2
10	4	3
11	4	4

Рис. 5: SELECT * FROM RoomTypeAmenity

5 ER Диаграмма

С помощью pgAdmin была построена ER диаграмма.

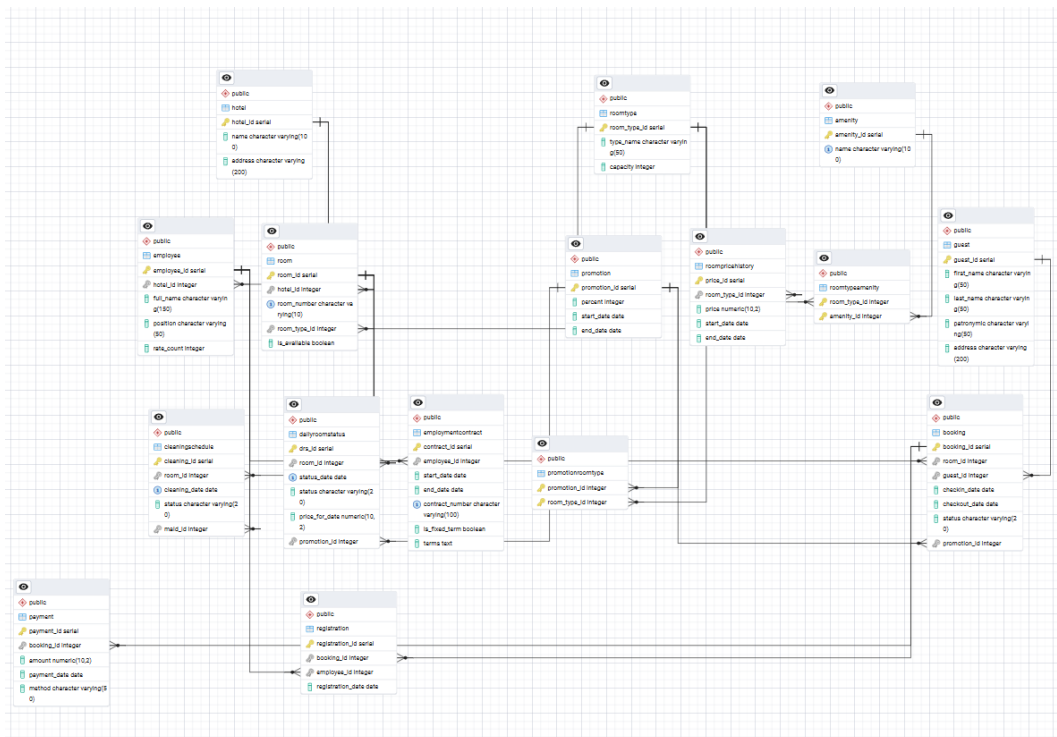


Рис. 6: ER диаграмма

6 Создание резервной копии базы данных

Через окно "Backup" были созданы две копии базы данных, с настройкой Custom для восстановления через pg_restore и с настройкой Plain для восстановления в командной строке psql.

Выводы

В ходе лабораторной работы была создана база данных "ОТЕЛЬ" созданы таблицы, ограничения на значения и связи между сущностями. База данных была наполнена тестовыми данными, создана ER диаграмма.

Также были сделаны бекапы базы данных и её восстановление через бекап.