Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №3.2 «Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Евдокимова У.В.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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.

   Указание:

   Создать две резервные копии:

   - с расширением CUSTOM для восстановления БД;

   - с расширением PLAIN для листинга (в отчете);

   - при создании резервных копий БД настроить параметры Dump options для Type of objects и Queries.
7. Восстановить БД.

Вариант 1. БД «Отель»

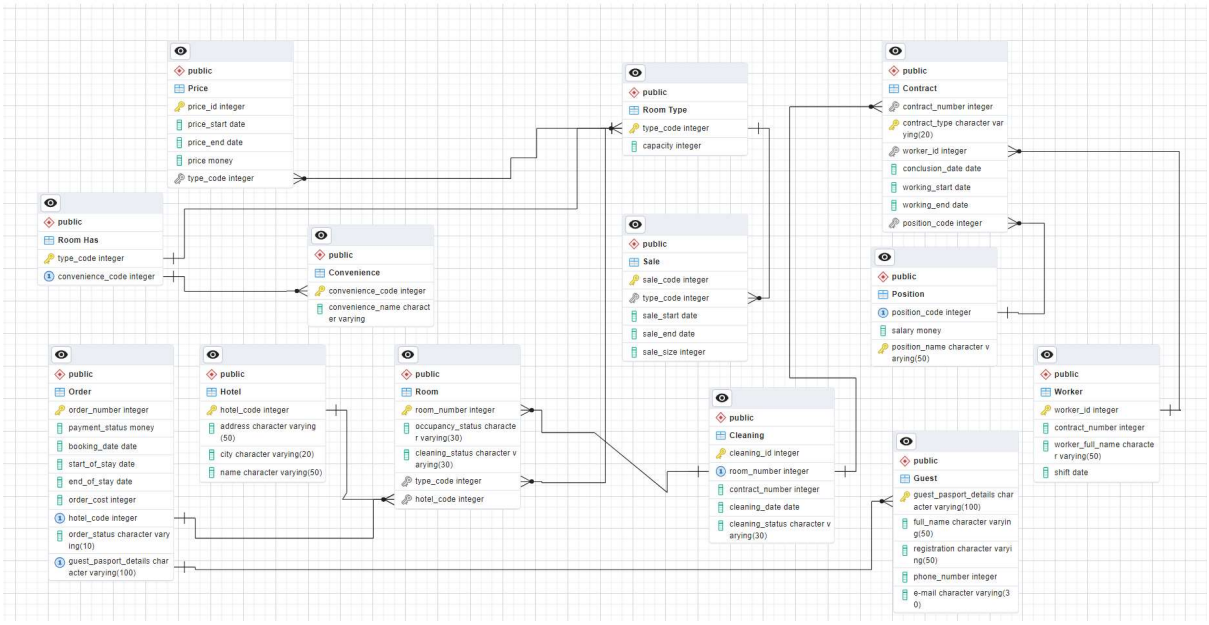

Схема логической модели базы данных, сгенерированная в Generate.

Так как по техническим причинам я не могу создать резервную копию бд, а, следовательно, получить дамп-файл, ниже будут представлены SQL-скрипты таблиц и их заполнения:

```
-- Database: Hotel

-- DROP DATABASE IF EXISTS "Hotel";

CREATE DATABASE "Hotel"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Russian_Russia.1251'
    LC_CTYPE = 'Russian_Russia.1251'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;

    -- SCHEMA: public

-- DROP SCHEMA IF EXISTS public ;

CREATE SCHEMA IF NOT EXISTS public
    AUTHORIZATION pg_database_owner;

COMMENT ON SCHEMA public
    IS 'standard public schema';

GRANT USAGE ON SCHEMA public TO PUBLIC;

GRANT ALL ON SCHEMA public TO pg_database_owner;

--создаём таблицу уборки
-- Table: public.Cleaning

-- DROP TABLE IF EXISTS public."Cleaning";

CREATE TABLE IF NOT EXISTS public."Cleaning"
(
    cleaning_id integer NOT NULL,
    room_number integer,
    contract_number integer,
    cleaning_date date,
    cleaning_status character varying(30) COLLATE pg_catalog."default",
    CONSTRAINT "Cleaning_pkey" PRIMARY KEY (cleaning_id),
    CONSTRAINT room_number_r UNIQUE (room_number)
        INCLUDE(room_number)
)

TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public."Cleaning"
    OWNER to postgres;
-- Index: fki_contract_number_c

-- DROP INDEX IF EXISTS public.fki_contract_number_c;

CREATE INDEX IF NOT EXISTS fki_contract_number_c
    ON public."Cleaning" USING btree
    (contract_number ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_contract_number_cleaning

-- DROP INDEX IF EXISTS public.fki_contract_number_cleaning;

CREATE INDEX IF NOT EXISTS fki_contract_number_cleaning
    ON public."Cleaning" USING btree
    (contract_number ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу с договорами
    -- Table: public.Contract

-- DROP TABLE IF EXISTS public."Contract";

CREATE TABLE IF NOT EXISTS public."Contract"
(
    contract_number integer NOT NULL,
    contract_type character varying(20) COLLATE pg_catalog."default" NOT NULL,
    worker_id integer NOT NULL,
    conclusion_date date,
    working_start date,
    working_end date,
    position_code integer,
    CONSTRAINT "Contract_pkey" PRIMARY KEY (contract_type),
    CONSTRAINT contract_number_c FOREIGN KEY (contract_number)
        REFERENCES public."Cleaning" (room_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT position_code_c FOREIGN KEY (position_code)
        REFERENCES public."Position" (position_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT worker_id_c FOREIGN KEY (worker_id)
        REFERENCES public."Worker" (worker_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT dt_chk CHECK (working_start < working_end)
)
```

```
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Contract"
    OWNER to postgres;
-- Index: fki_position_code_c

-- DROP INDEX IF EXISTS public.fki_position_code_c;

CREATE INDEX IF NOT EXISTS fki_position_code_c
    ON public."Contract" USING btree
    (position_code ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_worker_id_c

-- DROP INDEX IF EXISTS public.fki_worker_id_c;

CREATE INDEX IF NOT EXISTS fki_worker_id_c
    ON public."Contract" USING btree
    (worker_id ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу с удобствами
    -- Table: public.Convenience

-- DROP TABLE IF EXISTS public."Convenience";

CREATE TABLE IF NOT EXISTS public."Convenience"
(
    convenience_code integer NOT NULL,
    convenience_name character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Convenience_pkey" PRIMARY KEY (convenience_code),
    CONSTRAINT convenience_code_rh FOREIGN KEY (convenience_code)
        REFERENCES public."Room Has" (convenience_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Convenience"
    OWNER to postgres;
-- Index: fki_convenience_code_rh

-- DROP INDEX IF EXISTS public.fki_convenience_code_rh;

CREATE INDEX IF NOT EXISTS fki_convenience_code_rh
    ON public."Convenience" USING btree
    (convenience_code ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
    --создаём таблицу постояльцев
    -- Table: public.Guest

-- DROP TABLE IF EXISTS public."Guest";

CREATE TABLE IF NOT EXISTS public."Guest"
(
    guest_pasport_details character varying(100) COLLATE pg_catalog."default" NOT
NULL,
    full_name character varying(50) COLLATE pg_catalog."default",
    registration character varying(50) COLLATE pg_catalog."default",
    phone_number integer,
    "e-mail" character varying(30) COLLATE pg_catalog."default",
    CONSTRAINT "Guest_pkey" PRIMARY KEY (guest_pasport_details),
    CONSTRAINT guest_pasport_details_o FOREIGN KEY (guest_pasport_details)
        REFERENCES public."Order" (guest_pasport_details) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT phn_chk CHECK (phone_number > 11111111)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Guest"
    OWNER to postgres;
-- Index: fki_guest_pasport_details_o

-- DROP INDEX IF EXISTS public.fki_guest_pasport_details_o;

CREATE INDEX IF NOT EXISTS fki_guest_pasport_details_o
    ON public."Guest" USING btree
    (guest_pasport_details COLLATE pg_catalog."default" ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу с отелями
    -- Table: public.Hotel

-- DROP TABLE IF EXISTS public."Hotel";

CREATE TABLE IF NOT EXISTS public."Hotel"
(
    hotel_code integer NOT NULL,
    address character varying(50) COLLATE pg_catalog."default",
    city character varying(20) COLLATE pg_catalog."default",
    name character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT "Hotel_pkey" PRIMARY KEY (hotel_code),
    CONSTRAINT hk_chk CHECK (hotel_code > 0)
)

TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public."Hotel"
    OWNER to postgres;

    --создаём таблицу заказов
    -- Table: public.Order

-- DROP TABLE IF EXISTS public."Order";

CREATE TABLE IF NOT EXISTS public."Order"
(
    order_number integer NOT NULL,
    payment_status money,
    booking_date date,
    start_of_stay date,
    end_of_stay date,
    order_cost integer,
    hotel_code integer,
    order_status character varying(10) COLLATE pg_catalog."default",
    guest_pasport_details character varying(100) COLLATE pg_catalog."default",
    CONSTRAINT "Order_pkey" PRIMARY KEY (order_number),
    CONSTRAINT guest_pasport_details_o UNIQUE (guest_pasport_details)
        INCLUDE(guest_pasport_details),
    CONSTRAINT hotel_code_o UNIQUE (hotel_code)
        INCLUDE(hotel_code)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Order"
    OWNER to postgres;

    --создаём таблицу должностей
    -- Table: public.Position

-- DROP TABLE IF EXISTS public."Position";

CREATE TABLE IF NOT EXISTS public."Position"
(
    position_code integer,
    salary money,
    position_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Position_pkey" PRIMARY KEY (position_name),
    CONSTRAINT position_code_c UNIQUE (position_code)
        INCLUDE(position_code),
    CONSTRAINT slr_chk CHECK (salary > money(10))
)

TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public."Position"
    OWNER to postgres;

    -- Table: public.Position

-- DROP TABLE IF EXISTS public."Position";

CREATE TABLE IF NOT EXISTS public."Position"
(
    position_code integer,
    salary money,
    position_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Position_pkey" PRIMARY KEY (position_name),
    CONSTRAINT position_code_c UNIQUE (position_code)
        INCLUDE(position_code),
    CONSTRAINT slr_chk CHECK (salary > money(10))
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Position"
    OWNER to postgres;

    -- Table: public.Price

-- DROP TABLE IF EXISTS public."Price";

CREATE TABLE IF NOT EXISTS public."Price"
(
    price_id integer NOT NULL,
    price_start date,
    price_end date,
    price money,
    type_code integer,
    CONSTRAINT "Price_pkey" PRIMARY KEY (price_id),
    CONSTRAINT type_code_rt FOREIGN KEY (type_code)
        REFERENCES public."Room Type" (type_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT prc_check CHECK (price > money(70))
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Price"
    OWNER to postgres;
-- Index: fki_r

-- DROP INDEX IF EXISTS public.fki_r;
```

```
CREATE INDEX IF NOT EXISTS fki_r
    ON public."Price" USING btree
    (type_code ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу комнат
    -- Table: public.Room

-- DROP TABLE IF EXISTS public."Room";

CREATE TABLE IF NOT EXISTS public."Room"
(
    room_number integer NOT NULL,
    occupancy_status character varying(30) COLLATE pg_catalog."default",
    cleaning_status character varying(30) COLLATE pg_catalog."default",
    type_code integer,
    hotel_code integer,
    CONSTRAINT "Room_pkey" PRIMARY KEY (room_number),
    CONSTRAINT hotel_code_h FOREIGN KEY (hotel_code)
        REFERENCES public."Hotel" (hotel_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT hotel_code_o FOREIGN KEY (hotel_code)
        REFERENCES public."Order" (hotel_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT room_number_c FOREIGN KEY (room_number)
        REFERENCES public."Cleaning" (room_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT type_code_r FOREIGN KEY (type_code)
        REFERENCES public."Room Type" (type_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Room"
    OWNER to postgres;
-- Index: fki_hotel_code_h

-- DROP INDEX IF EXISTS public.fki_hotel_code_h;

CREATE INDEX IF NOT EXISTS fki_hotel_code_h
    ON public."Room" USING btree
    (hotel_code ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_hotel_code_o
```

```
-- DROP INDEX IF EXISTS public.fki_hotel_code_o;

CREATE INDEX IF NOT EXISTS fki_hotel_code_o
    ON public."Room" USING btree
    (hotel_code ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_room_number_c

-- DROP INDEX IF EXISTS public.fki_room_number_c;

CREATE INDEX IF NOT EXISTS fki_room_number_c
    ON public."Room" USING btree
    (room_number ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_type_code_r

-- DROP INDEX IF EXISTS public.fki_type_code_r;

CREATE INDEX IF NOT EXISTS fki_type_code_r
    ON public."Room" USING btree
    (type_code ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу характерстик комнаты
    -- Table: public.Room Has

-- DROP TABLE IF EXISTS public."Room Has";

CREATE TABLE IF NOT EXISTS public."Room Has"
(
    type_code integer NOT NULL,
    convenience_code integer,
    CONSTRAINT "Room Has_pkey" PRIMARY KEY (type_code),
    CONSTRAINT convenience_code_rh UNIQUE (convenience_code)
        INCLUDE(convenience_code),
    CONSTRAINT type_code_h UNIQUE (type_code)
        INCLUDE(type_code)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Room Has"
    OWNER to postgres;

    --создаём таблицу типов комнат
    -- Table: public.Room Type

-- DROP TABLE IF EXISTS public."Room Type";

CREATE TABLE IF NOT EXISTS public."Room Type"
```

```
(
    type_code integer NOT NULL,
    capacity integer,
    CONSTRAINT "Room Type_pkey" PRIMARY KEY (type_code),
    CONSTRAINT type_code_h FOREIGN KEY (type_code)
        REFERENCES public."Room Has" (type_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT cpc_chk CHECK (capacity > 0)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Room Type"
    OWNER to postgres;
-- Index: fki_type_code_h

-- DROP INDEX IF EXISTS public.fki_type_code_h;

CREATE INDEX IF NOT EXISTS fki_type_code_h
    ON public."Room Type" USING btree
    (type_code ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу акций
    -- Table: public.Sale

-- DROP TABLE IF EXISTS public."Sale";

CREATE TABLE IF NOT EXISTS public."Sale"
(
    sale_code integer NOT NULL,
    type_code integer,
    sale_start date,
    sale_end date,
    sale_size integer,
    CONSTRAINT "Sale_pkey" PRIMARY KEY (sale_code),
    CONSTRAINT type_code_rt FOREIGN KEY (type_code)
        REFERENCES public."Room Type" (type_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT sz_chk CHECK (sale_size < 55)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Sale"
    OWNER to postgres;
-- Index: fki_type_code_rt
```

```sql
-- DROP INDEX IF EXISTS public.fki_type_code_rt;

CREATE INDEX IF NOT EXISTS fki_type_code_rt
    ON public."Sale" USING btree
    (type_code ASC NULLS LAST)
    TABLESPACE pg_default;

    --создаём таблицу сотрудников

    -- Table: public.Worker

-- DROP TABLE IF EXISTS public."Worker";

CREATE TABLE IF NOT EXISTS public."Worker"
(
    worker_id integer NOT NULL,
    contract_number integer,
    worker_full_name character varying(50) COLLATE pg_catalog."default",
    shift date,
    CONSTRAINT "Worker_pkey" PRIMARY KEY (worker_id),
    CONSTRAINT worker_id_c UNIQUE (worker_id)
        INCLUDE(worker_id),
    CONSTRAINT cn_chk CHECK (contract_number > 0)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Worker"
    OWNER to postgres;

 --далее заполняем базу данных



INSERT INTO public."Price" (price_id, price_start, price_end, price, type_code)
VALUES (1, '2023-01-01', '2023-12-31', 100.00, 1),
(2, '2023-01-01', '2023-12-31', 150.00, 2),
(3, '2023-01-01', '2023-12-31', 200.00, 3),
(4, '2023-01-01', '2023-12-31', 250.00, 4);

INSERT INTO public."Room Type" (type_code, capacity)
VALUES (1, 2),
(2, 4),
(3, 6),
(4, 8);

INSERT INTO public."Room Has" (type_code, convenience_code)
VALUES (1, 1),
(2, 2),
(3, 3),
```

```
(4, 4);

INSERT INTO public."Sale" (sale_code, type_code, sale_start, sale_end, sale_size)
VALUES (1, 1, '2023-01-01', '2023-01-31', 10),
(2, 2, '2023-02-01', '2023-02-28', 15),
(3, 3, '2023-03-01', '2023-03-31', 20),
(4, 4, '2023-04-01', '2023-04-30', 25);

INSERT INTO public."Worker" (worker_id, contract_number, worker_full_name, shift)
VALUES (1, 1001, 'John Smith', '2023-01-01'),
(2, 1002, 'Jane Doe', '2023-01-02'),
(3, 1003, 'Michael Johnson', '2023-01-03'),
(4, 1004, 'Emily Williams', '2023-01-04');

INSERT INTO public."Convenience" (convenience_code, convenience_name)
VALUES (1, 'Wi-Fi'),
(2, 'TV'),
(3, 'Air conditioning'),
(4, 'Mini fridge');

INSERT INTO public."Hotel" (hotel_code, address, city, name)
VALUES (1, '123 Main Street', 'New York', 'Hotel A'),
(2, '456 Elm Street', 'Los Angeles', 'Hotel B'),
(3, '789 Oak Street', 'Chicago', 'Hotel C'),
(4, '321 Pine Street', 'San Francisco', 'Hotel D');

INSERT INTO public."Room" (room_number, occupancy_status, cleaning_status,
type_code, hotel_code)
VALUES (101, 'Occupied', 'Clean', 1, 1),
(102, 'Vacant', 'Dirty', 2, 1),
(103, 'Occupied', 'Clean', 3, 2),
(104, 'Vacant', 'Dirty', 4, 2);

INSERT INTO public."Order" (order_number, payment_status, booking_date,
start_of_stay, end_of_stay, order_cost, hotel_code, order_status,
guest_pasport_details)
VALUES (1, 100.00, '2023-10-01', '2023-10-10', '2023-10-15', 500, 1, 'Confirmed',
'AB123456'),
(2, 150.00, '2023-11-05', '2023-11-10', '2023-11-15', 750, 2, 'Confirmed',
'CD987654');

INSERT INTO public."Guest" (guest_pasport_details, full_name, registration,
phone_number, "e-mail")
VALUES ('AB123456', 'John Doe', 'New York', 1234567890, 'john@example.com'),
('CD987654', 'Jane Smith', 'Los Angeles', 9876543210, 'jane@example.com');

INSERT INTO public."Position" (position_code, salary, position_name)
VALUES (1, 5000.00, 'Manager'),
(2, 3000.00, 'Receptionist'),
(3, 2500.00, 'Housekeeper'),
```

```
(4, 4000.00, 'Chef'),
(5, 3500.00, 'Waiter');

INSERT INTO public."Cleaning" (cleaning_id, room_number, contract_number,
cleaning_date, cleaning_status)
VALUES (1, 1, 1, '2023-10-25', 'Clean'),
(2, 2, 2, '2023-10-25', 'Dirty'),
(3, 3, 3, '2023-10-25', 'Clean'),
(4, 4, 4, '2023-10-25', 'Dirty');

INSERT INTO public."Contract" (contract_number, contract_type, worker_id,
conclusion_date, working_start, working_end, position_code)
VALUES (101, 'Type A', 1, '2023-01-01', '2023-01-02', '2023-01-31', 1),
(102, 'Type B', 2, '2023-02-01', '2023-02-02', '2023-02-28', 2),
(103, 'Type C', 3, '2023-03-01', '2023-03-02', '2023-03-31', 3);
```

Вывод

В ходе лабораторной работы я освоила практические навыки по созданию, заполнению и восстановлению баз данных в PostgreSQL с использованием инструмента управления pgAdmin 4. Была создана структура базы данных, включая таблицы с различными ограничениями для обеспечения целостности данных. Далее, таблицы были заполнены рабочими данными. Для безопасности информации были созданы резервные копии с разными расширениями, что позволило как восстановить базу данных, так и просмотреть листинг данных. Конечным этапом стало успешное восстановление БД, подтверждающее корректность ранее выполненных действий.