МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Технология разработки ПО

Лабораторная работа №7

«Создание физической модели данных»

Авиаперевозки

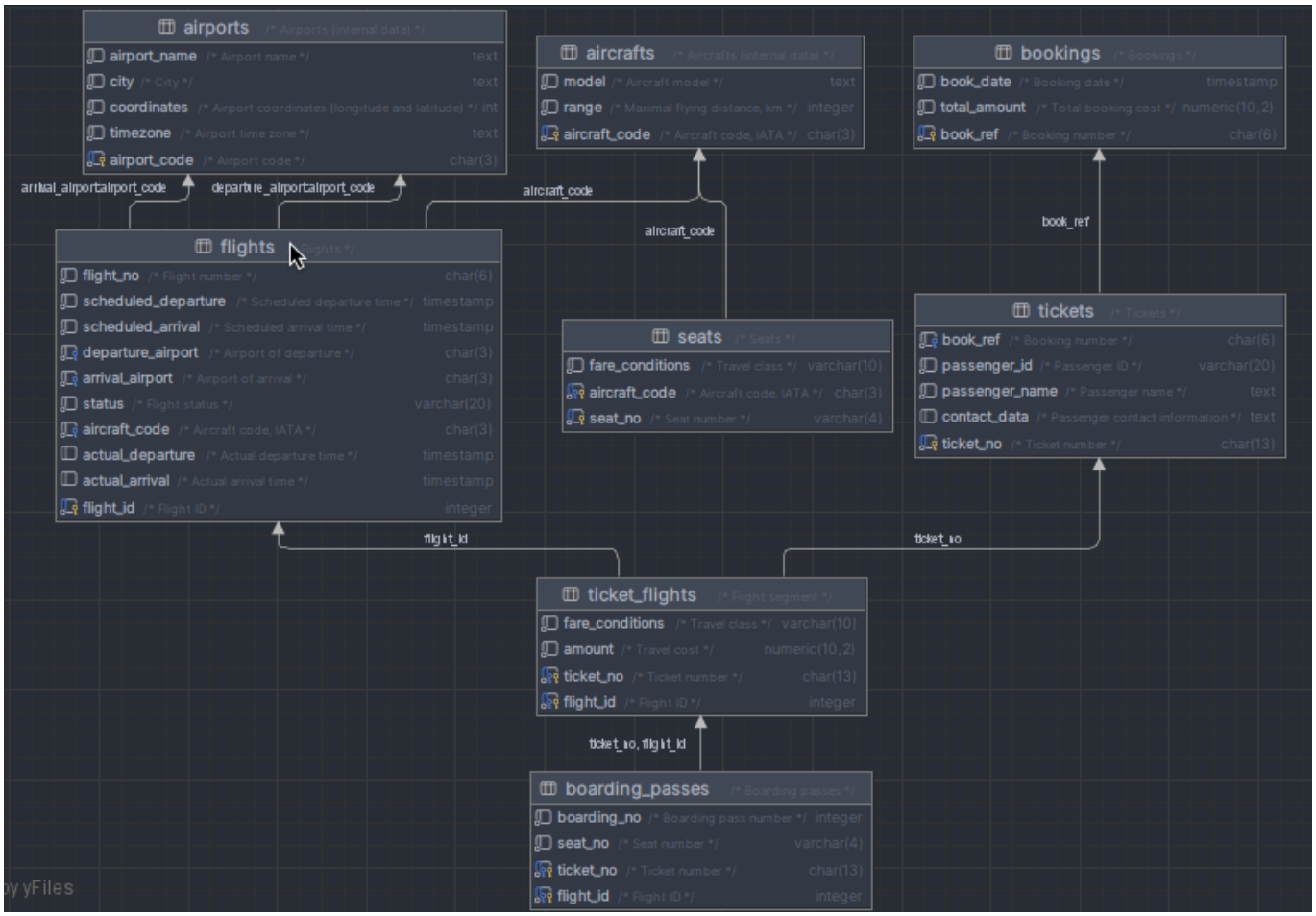
Выполнил:
студент группы ИВТ-41-20
Галкин Д.С.

Проверил:
Ржавин В.В.

Чебоксары 2023

Цель работы: разработать физическую модель данных

Диаграмма физической модели:

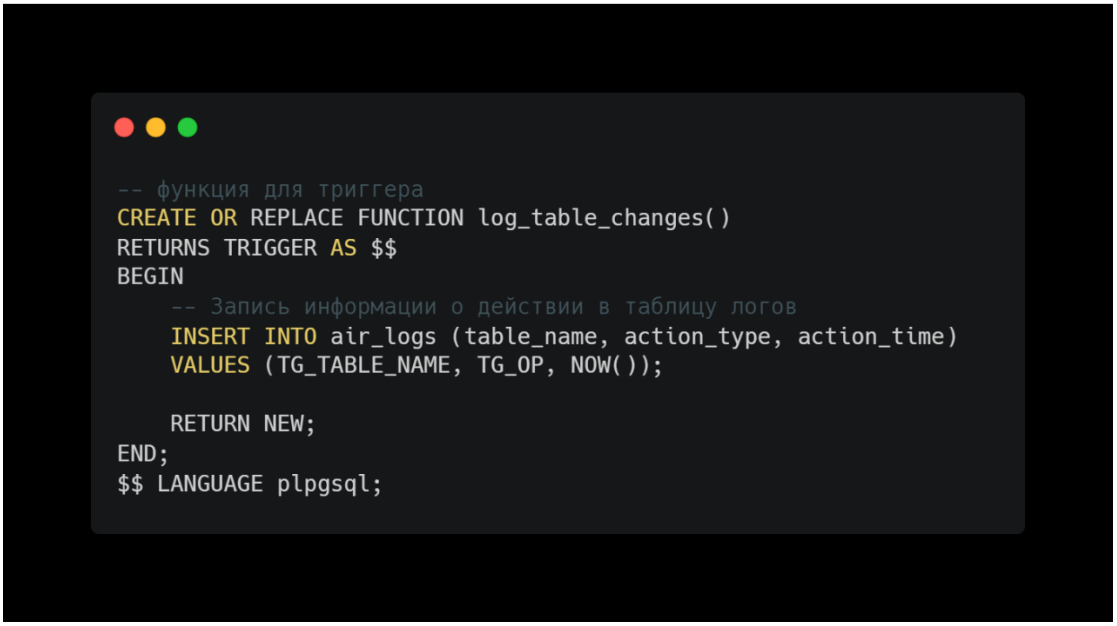


Тексты разработанных триггеров и хранимых процедур:

- Логирование
  - Создание дополнительной таблицы

```sql
-- Таблиц логов
CREATE TABLE IF NOT EXISTS air_logs (
    log_id serial PRIMARY KEY,
    table_name text,
    action_type text,
    action_time timestamp
);
```

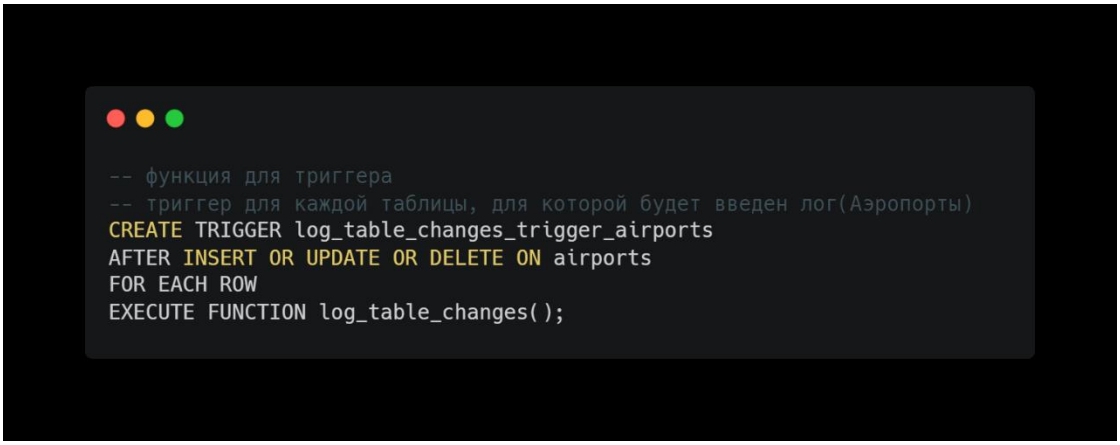
- Реализация хранимой процедуры



```
-- функция для триггера
CREATE OR REPLACE FUNCTION log_table_changes()
RETURNS TRIGGER AS $$
BEGIN
    -- Запись информации о действии в таблицу логов
    INSERT INTO air_logs (table_name, action_type, action_time)
    VALUES (TG_TABLE_NAME, TG_OP, NOW());

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

- Триггеры логов для всех таблиц



```
-- функция для триггера
-- триггер для каждой таблицы, для которой будет введен лог(Аэропорты)
CREATE TRIGGER log_table_changes_trigger_airports
AFTER INSERT OR UPDATE OR DELETE ON airports
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();
```

```sql
-- функция для триггера
-- триггер для каждой таблицы, для которой будет введен лог(Самолеты)
CREATE TRIGGER log_table_changes_trigger_aircrafts
AFTER INSERT OR UPDATE OR DELETE ON aircrafts
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Посадочные талоны)
CREATE TRIGGER log_table_changes_trigger_boarding_passes
AFTER INSERT OR UPDATE OR DELETE ON boarding_passes
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Бронирование)
CREATE TRIGGER log_table_changes_trigger_bookings
AFTER INSERT OR UPDATE OR DELETE ON bookings
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Рейсы)
CREATE TRIGGER log_table_changes_trigger_flights
AFTER INSERT OR UPDATE OR DELETE ON flights
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Посадочные места)
CREATE TRIGGER log_table_changes_trigger_seats
AFTER INSERT OR UPDATE OR DELETE ON seats
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Перелеты)
CREATE TRIGGER log_table_changes_trigger_ticket_flights
AFTER INSERT OR UPDATE OR DELETE ON ticket_flights
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();

-- триггер для каждой таблицы, для которой будет введен лог(Билеты)
CREATE TRIGGER log_table_changes_trigger_tickets
AFTER INSERT OR UPDATE OR DELETE ON seats
FOR EACH ROW
EXECUTE FUNCTION log_table_changes();
```

- Перевод типа Jsonb в текст
    - Настраиваем локализацию

```sql
-- Настройки локализации языка
ALTER DATABASE "Air" SET bookings.lang = ru;
```

- Создаем хранимую процедуру

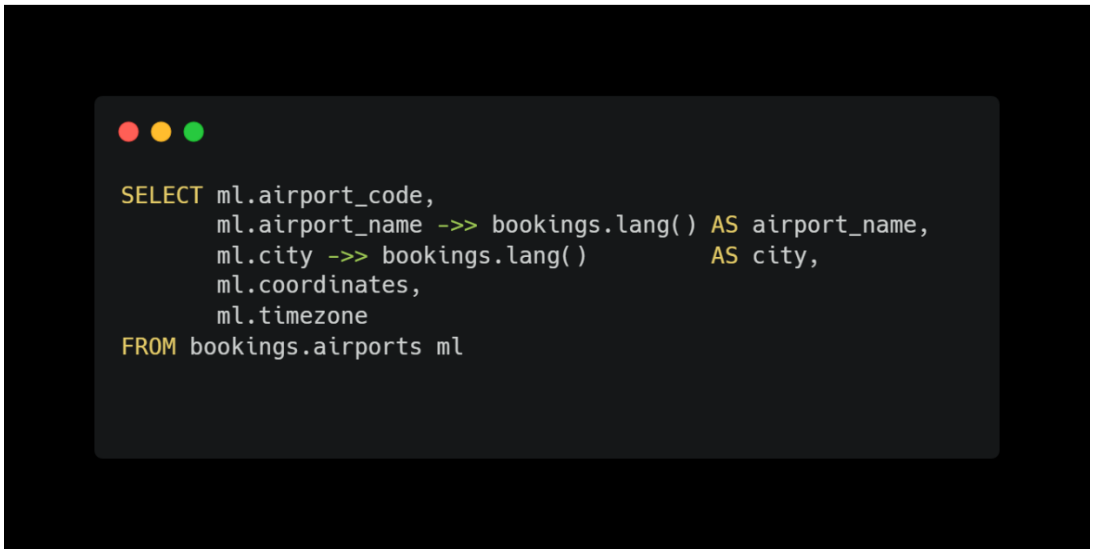
```
create function lang() returns text
    stable
    language plpgsql
as
$$
BEGIN
  RETURN current_setting('bookings.lang');
EXCEPTION
  WHEN undefined_object THEN
    RETURN NULL;
END;
$$;

alter function lang() owner to postgres;
```

- Создаем View(aircrafts)

```
SELECT ml.aircraft_code,
       ml.model ->> bookings.lang() AS model,
       ml.range
FROM bookings.aircrafts ml
```

- Создаем View(airports)



```
SELECT ml.airport_code,
       ml.airport_name ->> bookings.lang() AS airport_name,
       ml.city ->> bookings.lang()         AS city,
       ml.coordinates,
       ml.timezone
FROM bookings.airports ml
```

PostgreSQL-код для создания базы данных в DataGrip:

```
-- Database generated with pgModeler (PostgreSQL Database Modeler).
-- pgModeler version: 1.1.0-alpha
-- PostgreSQL version: 15.0
-- Project Site: pgmodeler.io
-- Model Author: ---
-- -- object: pg_database_owner | type: ROLE --
-- -- DROP ROLE IF EXISTS pg_database_owner;
-- CREATE ROLE pg_database_owner WITH
--  INHERIT
--   PASSWORD '********';
-- -- ddl-end --
--

-- Database creation must be performed outside a multi lined SQL file.
-- These commands were put in this file only as a convenience.
--
-- object: "Air_test" | type: DATABASE --
-- DROP DATABASE IF EXISTS "Air_test";
CREATE DATABASE "Air_test"
    ENCODING = 'UTF8'
    LC_COLLATE = 'English_United States.1251'
    LC_CTYPE = 'English_United States.1251'
    TABLESPACE = pg_default
    OWNER = postgres;
-- ddl-end --


SET check_function_bodies = false;
-- ddl-end --

-- object: bookings | type: SCHEMA --
-- DROP SCHEMA IF EXISTS bookings CASCADE;
CREATE SCHEMA bookings;
-- ddl-end --
ALTER SCHEMA bookings OWNER TO postgres;
-- ddl-end --
COMMENT ON SCHEMA bookings IS E'Airlines demo database schema';
-- ddl-end --

-- Appended SQL commands --
ALTER TABLE bookings
ADD COLUMN test_amount type_amount;
-- ddl-end --

SET search_path TO pg_catalog,public,bookings;
-- ddl-end --

-- object: bookings.lang | type: FUNCTION --
-- DROP FUNCTION IF EXISTS bookings.lang() CASCADE;
CREATE FUNCTION bookings.lang ()
    RETURNS text
```

```
    LANGUAGE plpgsql
    STABLE
    CALLED ON NULL INPUT
    SECURITY INVOKER
    PARALLEL UNSAFE
    COST 100
    AS $$
BEGIN
  RETURN current_setting('bookings.lang');
EXCEPTION
  WHEN undefined_object THEN
    RETURN NULL;
END;
$$;
-- ddl-end --
ALTER FUNCTION bookings.lang() OWNER TO postgres;
-- ddl-end --

-- object: bookings.now | type: FUNCTION --
-- DROP FUNCTION IF EXISTS bookings.now() CASCADE;
CREATE FUNCTION bookings.now ()
    RETURNS timestamp with time zone
    LANGUAGE sql
    IMMUTABLE
    CALLED ON NULL INPUT
    SECURITY INVOKER
    PARALLEL UNSAFE
    COST 100
    AS $$
SELECT '2017-08-15 18:00:00'::TIMESTAMP AT TIME ZONE 'Europe/Moscow';
$$;
-- ddl-end --
ALTER FUNCTION bookings.now() OWNER TO postgres;
-- ddl-end --
COMMENT ON FUNCTION bookings.now() IS E'Point in time according to
which the data are generated';
-- ddl-end --

-- object: bookings.aircrafts | type: TABLE --
-- DROP TABLE IF EXISTS bookings.aircrafts CASCADE;
CREATE TABLE bookings.aircrafts (
    aircraft_code character(3) NOT NULL,
    model text NOT NULL,
    range integer NOT NULL,
    CONSTRAINT aircrafts_range_check CHECK ((range > 0)),
    CONSTRAINT aircrafts_pkey PRIMARY KEY (aircraft_code)
);
-- ddl-end --
COMMENT ON TABLE bookings.aircrafts IS E'Aircrafts (internal data)';
-- ddl-end --
COMMENT ON COLUMN bookings.aircrafts.aircraft_code IS E'Aircraft code,
IATA';
-- ddl-end --
```

```
COMMENT ON COLUMN bookings.aircrafts.model IS E'Aircraft model';
-- ddl-end --
COMMENT ON COLUMN bookings.aircrafts.range IS E'Maximal flying dis-
tance, km';
-- ddl-end --
ALTER TABLE bookings.aircrafts OWNER TO postgres;
-- ddl-end --

-- object: bookings.airports | type: TABLE --
-- DROP TABLE IF EXISTS bookings.airports CASCADE;
CREATE TABLE bookings.airports (
    airport_code character(3) NOT NULL,
    airport_name text NOT NULL,
    city text NOT NULL,
    coordinates point NOT NULL,
    timezone text NOT NULL,
    CONSTRAINT airports_data_pkey PRIMARY KEY (airport_code)
);
-- ddl-end --
COMMENT ON TABLE bookings.airports IS E'Airports (internal data)';
-- ddl-end --
COMMENT ON COLUMN bookings.airports.airport_code IS E'Airport code';
-- ddl-end --
COMMENT ON COLUMN bookings.airports.airport_name IS E'Airport name';
-- ddl-end --
COMMENT ON COLUMN bookings.airports.city IS E'City';
-- ddl-end --
COMMENT ON COLUMN bookings.airports.coordinates IS E'Airport coordi-
nates (longitude and latitude)';
-- ddl-end --
COMMENT ON COLUMN bookings.airports.timezone IS E'Airport time zone';
-- ddl-end --
ALTER TABLE bookings.airports OWNER TO postgres;
-- ddl-end --

-- object: bookings.boarding_passes | type: TABLE --
-- DROP TABLE IF EXISTS bookings.boarding_passes CASCADE;
CREATE TABLE bookings.boarding_passes (
    ticket_no character(13) NOT NULL,
    flight_id integer NOT NULL,
    boarding_no integer NOT NULL,
    seat_no character varying(4) NOT NULL,
    CONSTRAINT boarding_passes_pkey PRIMARY KEY (ticket_no,flight_id)
);
-- ddl-end --
COMMENT ON TABLE bookings.boarding_passes IS E'Boarding passes';
-- ddl-end --
COMMENT ON COLUMN bookings.boarding_passes.ticket_no IS E'Ticket num-
ber';
-- ddl-end --
COMMENT ON COLUMN bookings.boarding_passes.flight_id IS E'Flight ID';
-- ddl-end --
COMMENT ON COLUMN bookings.boarding_passes.boarding_no IS E'Boarding
```

```
pass number';
-- ddl-end --
COMMENT ON COLUMN bookings.boarding_passes.seat_no IS E'Seat number';
-- ddl-end --
ALTER TABLE bookings.boarding_passes OWNER TO postgres;
-- ddl-end --

-- object: bookings.bookings | type: TABLE --
-- DROP TABLE IF EXISTS bookings.bookings CASCADE;
CREATE TABLE bookings.bookings (
    book_ref character(6) NOT NULL,
    book_date timestamp NOT NULL,
    total_amount numeric(10,2) NOT NULL,
    CONSTRAINT bookings_pkey PRIMARY KEY (book_ref)
);
-- ddl-end --
COMMENT ON TABLE bookings.bookings IS E'Bookings';
-- ddl-end --
COMMENT ON COLUMN bookings.bookings.book_ref IS E'Booking number';
-- ddl-end --
COMMENT ON COLUMN bookings.bookings.book_date IS E'Booking date';
-- ddl-end --
COMMENT ON COLUMN bookings.bookings.total_amount IS E'Total booking
cost';
-- ddl-end --
ALTER TABLE bookings.bookings OWNER TO postgres;
-- ddl-end --

-- object: bookings.flights_flight_id_seq | type: SEQUENCE --
-- DROP SEQUENCE IF EXISTS bookings.flights_flight_id_seq CASCADE;
CREATE SEQUENCE bookings.flights_flight_id_seq
    INCREMENT BY 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    START WITH 1
    CACHE 1
    NO CYCLE
    OWNED BY NONE;

-- ddl-end --
ALTER SEQUENCE bookings.flights_flight_id_seq OWNER TO postgres;
-- ddl-end --

-- object: bookings.flights | type: TABLE --
-- DROP TABLE IF EXISTS bookings.flights CASCADE;
CREATE TABLE bookings.flights (
    flight_id integer NOT NULL DEFAULT next-
val('flights_flight_id_seq'::regclass),
    flight_no character(6) NOT NULL,
    scheduled_departure timestamp NOT NULL,
    scheduled_arrival timestamp NOT NULL,
    departure_airport character(3) NOT NULL,
    arrival_airport character(3) NOT NULL,
```

```
    status character varying(20) NOT NULL,
    aircraft_code character(3) NOT NULL,
    actual_departure timestamp,
    actual_arrival timestamp,
    CONSTRAINT flights_pkey PRIMARY KEY (flight_id)
);
-- ddl-end --
COMMENT ON TABLE bookings.flights IS E'Flights';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.flight_id IS E'Flight ID';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.flight_no IS E'Flight number';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.scheduled_departure IS E'Scheduled
departure time';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.scheduled_arrival IS E'Scheduled
arrival time';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.departure_airport IS E'Airport of
departure';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.arrival_airport IS E'Airport of ar-
rival';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.status IS E'Flight status';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.aircraft_code IS E'Aircraft code,
IATA';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.actual_departure IS E'Actual depar-
ture time';
-- ddl-end --
COMMENT ON COLUMN bookings.flights.actual_arrival IS E'Actual arrival
time';
-- ddl-end --
ALTER TABLE bookings.flights OWNER TO postgres;
-- ddl-end --

-- object: bookings.seats | type: TABLE --
-- DROP TABLE IF EXISTS bookings.seats CASCADE;
CREATE TABLE bookings.seats (
    aircraft_code character(3) NOT NULL,
    seat_no character varying(4) NOT NULL,
    fare_conditions character varying(10) NOT NULL,
    CONSTRAINT seats_fare_conditions_check CHECK (((fare_condi-
tions)::text = ANY (ARRAY[('Economy'::character varying)::text, ('Com-
fort'::character varying)::text, ('Business'::character vary-
ing)::text]))),
    CONSTRAINT seats_pkey PRIMARY KEY (aircraft_code,seat_no)
);
-- ddl-end --
COMMENT ON TABLE bookings.seats IS E'Seats';
```

```
-- ddl-end --
COMMENT ON COLUMN bookings.seats.aircraft_code IS E'Aircraft code,
IATA';
-- ddl-end --
COMMENT ON COLUMN bookings.seats.seat_no IS E'Seat number';
-- ddl-end --
COMMENT ON COLUMN bookings.seats.fare_conditions IS E'Travel class';
-- ddl-end --
ALTER TABLE bookings.seats OWNER TO postgres;
-- ddl-end --

-- object: bookings.ticket_flights | type: TABLE --
-- DROP TABLE IF EXISTS bookings.ticket_flights CASCADE;
CREATE TABLE bookings.ticket_flights (
    ticket_no character(13) NOT NULL,
    flight_id integer NOT NULL,
    fare_conditions character varying(10) NOT NULL,
    amount numeric(10,2) NOT NULL,
    CONSTRAINT ticket_flights_pkey PRIMARY KEY (ticket_no,flight_id)
);
-- ddl-end --
COMMENT ON TABLE bookings.ticket_flights IS E'Flight segment';
-- ddl-end --
COMMENT ON COLUMN bookings.ticket_flights.ticket_no IS E'Ticket num-
ber';
-- ddl-end --
COMMENT ON COLUMN bookings.ticket_flights.flight_id IS E'Flight ID';
-- ddl-end --
COMMENT ON COLUMN bookings.ticket_flights.fare_conditions IS E'Travel
class';
-- ddl-end --
COMMENT ON COLUMN bookings.ticket_flights.amount IS E'Travel cost';
-- ddl-end --
ALTER TABLE bookings.ticket_flights OWNER TO postgres;
-- ddl-end --

-- object: bookings.tickets | type: TABLE --
-- DROP TABLE IF EXISTS bookings.tickets CASCADE;
CREATE TABLE bookings.tickets (
    ticket_no character(13) NOT NULL,
    book_ref character(6) NOT NULL,
    passenger_id character varying(20) NOT NULL,
    passenger_name text NOT NULL,
    contact_data text,
    CONSTRAINT tickets_pkey PRIMARY KEY (ticket_no)
);
-- ddl-end --
COMMENT ON TABLE bookings.tickets IS E'Tickets';
-- ddl-end --
COMMENT ON COLUMN bookings.tickets.ticket_no IS E'Ticket number';
-- ddl-end --
COMMENT ON COLUMN bookings.tickets.book_ref IS E'Booking number';
-- ddl-end --
```

```
COMMENT ON COLUMN bookings.tickets.passenger_id IS E'Passenger ID';
-- ddl-end --
COMMENT ON COLUMN bookings.tickets.passenger_name IS E'Passenger
name';
-- ddl-end --
COMMENT ON COLUMN bookings.tickets.contact_data IS E'Passenger contact
information';
-- ddl-end --
ALTER TABLE bookings.tickets OWNER TO postgres;
-- ddl-end --

-- object: bookings.type_amount | type: DOMAIN --
-- DROP DOMAIN IF EXISTS bookings.type_amount CASCADE;
CREATE DOMAIN bookings.type_amount AS numeric(10,2);
-- ddl-end --
ALTER DOMAIN bookings.type_amount OWNER TO postgres;
-- ddl-end --

-- object: boarding_passes_ticket_no_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.boarding_passes DROP CONSTRAINT IF EXISTS
boarding_passes_ticket_no_fkey CASCADE;
ALTER TABLE bookings.boarding_passes ADD CONSTRAINT board-
ing_passes_ticket_no_fkey FOREIGN KEY (ticket_no,flight_id)
REFERENCES bookings.ticket_flights (ticket_no,flight_id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: flights_aircraft_code_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.flights DROP CONSTRAINT IF EXISTS flights_air-
craft_code_fkey CASCADE;
ALTER TABLE bookings.flights ADD CONSTRAINT flights_aircraft_code_fkey
FOREIGN KEY (aircraft_code)
REFERENCES bookings.aircrafts (aircraft_code) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: flights_arrival_airport_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.flights DROP CONSTRAINT IF EXISTS flights_ar-
rival_airport_fkey CASCADE;
ALTER TABLE bookings.flights ADD CONSTRAINT flights_arrival_air-
port_fkey FOREIGN KEY (arrival_airport)
REFERENCES bookings.airports (airport_code) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: flights_departure_airport_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.flights DROP CONSTRAINT IF EXISTS flights_de-
parture_airport_fkey CASCADE;
ALTER TABLE bookings.flights ADD CONSTRAINT flights_departure_air-
port_fkey FOREIGN KEY (departure_airport)
REFERENCES bookings.airports (airport_code) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --
```

```
-- object: seats_aircraft_code_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.seats DROP CONSTRAINT IF EXISTS seats_air-
craft_code_fkey CASCADE;
ALTER TABLE bookings.seats ADD CONSTRAINT seats_aircraft_code_fkey
FOREIGN KEY (aircraft_code)
REFERENCES bookings.aircrafts (aircraft_code) MATCH SIMPLE
ON DELETE CASCADE ON UPDATE NO ACTION;
-- ddl-end --

-- object: ticket_flights_flight_id_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.ticket_flights DROP CONSTRAINT IF EXISTS
ticket_flights_flight_id_fkey CASCADE;
ALTER TABLE bookings.ticket_flights ADD CONSTRAINT
ticket_flights_flight_id_fkey FOREIGN KEY (flight_id)
REFERENCES bookings.flights (flight_id) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: ticket_flights_ticket_no_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.ticket_flights DROP CONSTRAINT IF EXISTS
ticket_flights_ticket_no_fkey CASCADE;
ALTER TABLE bookings.ticket_flights ADD CONSTRAINT
ticket_flights_ticket_no_fkey FOREIGN KEY (ticket_no)
REFERENCES bookings.tickets (ticket_no) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: tickets_book_ref_fkey | type: CONSTRAINT --
-- ALTER TABLE bookings.tickets DROP CONSTRAINT IF EXISTS tick-
ets_book_ref_fkey CASCADE;
ALTER TABLE bookings.tickets ADD CONSTRAINT tickets_book_ref_fkey FOR-
EIGN KEY (book_ref)
REFERENCES bookings.bookings (book_ref) MATCH SIMPLE
ON DELETE NO ACTION ON UPDATE NO ACTION;
-- ddl-end --

-- object: "grant_CU_26541e8cda" | type: PERMISSION --
GRANT CREATE,USAGE
   ON SCHEMA public
   TO pg_database_owner;
-- ddl-end --

-- object: "grant_U_cd8e46e7b6" | type: PERMISSION --
GRANT USAGE
   ON SCHEMA public
   TO PUBLIC;
-- ddl-end --
```

Вывод: в ходе выполнения данной лабораторной работы я разработал логическую модель данных «Авиоперевозки» в PGModeler.