# Домашнее задание №9

Андрей Козлов, гр. 4538

- Создадим таблицы, описывающие физическую модель.

```
CREATE TABLE customers (
        customer_id SERIAL PRIMARY KEY,
        name VARCHAR(50) NOT NULL
);

CREATE TABLE planes (
        plane_id SERIAL PRIMARY KEY,
        model VARCHAR(10) NOT NULL
);

CREATE TABLE seats (
        seat_id SERIAL PRIMARY KEY,
        plane_id INT NOT NULL REFERENCES planes (plane_id),
        seat_no INT NOT NULL CHECK (seat_no > 0)
);

CREATE TABLE flights (
        flight_id SERIAL PRIMARY KEY,
        departure_time TIMESTAMP NOT NULL,
        plane_id INT NOT NULL REFERENCES planes (plane_id),
        is_blocked BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE tickets (
        ticket_id SERIAL PRIMARY KEY,
        flight_id INT NOT NULL REFERENCES flights (flight_id),
        seat_id INT NOT NULL REFERENCES seats (seat_id),
        customer_id INT NOT NULL REFERENCES customers (customer_id),
        booking_last_update TIMESTAMP,
        UNIQUE (flight_id, seat_id)
);
```

- Также создадим несколько вспомогательных представлений.

```
CREATE OR REPLACE VIEW not_blocked_seats AS
          SELECT flight_id , seat_no , seat_id FROM flights
                    NATURAL JOIN seats
          WHERE ((departure_time - localtimestamp) >= INTERVAL '2 hour')
          AND NOT is_blocked
;

CREATE OR REPLACE VIEW not_blocked_tickets AS
          SELECT flight_id , seat_no , seat_id , customer_id ,
                    booking_last_update
          FROM flights NATURAL JOIN tickets NATURAL JOIN seats
          WHERE ((departure_time - localtimestamp) >= INTERVAL '2 hour')
          AND NOT is_blocked
;

CREATE OR REPLACE VIEW available_seats AS
          SELECT * FROM not_blocked_seats
          EXCEPT ALL
          SELECT flight_id , seat_no , seat_id FROM tickets
                    NATURAL JOIN seats
;

CREATE OR REPLACE VIEW ovedue_bookings AS
          SELECT flight_id , seat_no , seat_id FROM flights
                    NATURAL JOIN tickets NATURAL JOIN seats
          WHERE (localtimestamp - booking_last_update >= INTERVAL '1 day')
          AND (departure_time - booking_last_update >= INTERVAL '2 days')
          AND NOT is_blocked
;

CREATE OR REPLACE VIEW available_bookings AS
          SELECT flight_id , seat_no , seat_id FROM not_blocked_tickets
                    WHERE NOT booking_last_update IS NULL
          EXCEPT ALL
          SELECT * FROM ovedue_bookings
;
```

1. По номеру рейса — список мест, доступных для продажи и бронирования.

```
CREATE OR REPLACE FUNCTION free_seats(flight_id_ INT)
RETURNS TABLE (seat_no INT, seat_id INT) AS $$
          BEGIN
                    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY;

                    RETURN QUERY (SELECT avs.seat_no , avs.seat_id FROM (
                              SELECT * FROM available_seats
                              UNION
                              SELECT * FROM ovedue_bookings
                    ) AS avs WHERE avs.flight_id = flight_id_);
          END;
$$ LANGUAGE plpgsql;
```

2. Бронирование места.

```
CREATE OR REPLACE FUNCTION book_seat(
        customer_id_ INT, flight_id_ INT, seat_id_ INT
) RETURNS BOOLEAN AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;

                IF (SELECT count(*) FROM available_seats
                        WHERE flight_id = flight_id_
                        AND seat_id = seat_id_) > 0
                THEN
                        INSERT INTO tickets (
                                flight_id, seat_id, customer_id,
                                booking_last_update
                        ) VALUES (flight_id_, seat_id_, customer_id_,
                                localtimestamp);
                        RETURN TRUE;
                ELSIF (SELECT count(*) FROM ovedue_bookings
                        WHERE flight_id = flight_id_
                        AND seat_id = seat_id_) > 0
                THEN
                        UPDATE tickets
                        SET customer_id = customer_id_,
                        booking_last_update = localtimestamp
                                WHERE flight_id = flight_id_
                                AND seat_id = seat_id_;
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
                END IF;
        END;
$$ LANGUAGE plpgsql;
```

3. Продление брони.

```
CREATE OR REPLACE FUNCTION update_booking(
        customer_id_ INT, flight_id_ INT, seat_id_ INT
) RETURNS BOOLEAN AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;

                IF (SELECT count(*) FROM available_bookings
                        WHERE flight_id = flight_id_
                        AND seat_id = seat_id_) > 0
                THEN
                        UPDATE tickets
                        SET booking_last_update = localtimestamp
                                WHERE flight_id = flight_id_
                                AND seat_id = seat_id_;
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
                END IF;
        END;
$$ LANGUAGE plpgsql;
```

4. Покупка места.

```
CREATE OR REPLACE FUNCTION buy_seat(
        customer_id_ INT, flight_id_ INT, seat_id_ INT
) RETURNS BOOLEAN AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;

                IF (SELECT count(*) FROM available_seats
                        WHERE flight_id = flight_id_) > 0
                THEN
                        INSERT INTO tickets (
                                flight_id, seat_id, customer_id
                        ) VALUES (flight_id_, seat_id_, customer_id_);
                        RETURN TRUE;
                ELSIF (SELECT count(*) FROM ovedue_bookings
                        WHERE flight_id = flight_id_) > 0
                THEN
                        UPDATE tickets SET customer_id = customer_id_,
                        booking_last_update = NULL
                                WHERE flight_id = flight_id_
                                AND seat_id = seat_id_;
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
                END IF;
        END;
$$ LANGUAGE plpgsql;
```

5. Покупка места по брони.

```
CREATE OR REPLACE FUNCTION buy_seat_by_booking(
        customer_id_ INT, flight_id_ INT, seat_id_ INT
) RETURNS BOOLEAN AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;

                IF (SELECT count(*) FROM not_blocked_tickets
                        NATURAL JOIN available_bookings
                                WHERE customer_id = customer_id_
                                AND flight_id = flight_id_
                                AND seat_id = seat_id_) > 0
                THEN
                        UPDATE tickets SET booking_last_update = NULL
                                WHERE customer_id = customer_id_
                                AND flight_id = flight_id_
                                AND seat_id = seat_id_;
                        RETURN TRUE;
                ELSE
                        RETURN FALSE;
                END IF;
        END;
$$ LANGUAGE plpgsql;
```

6. Закрытие продаж на рейс по запросу администратора.

```
CREATE OR REPLACE FUNCTION block_flight(flight_id_ INT)
RETURNS VOID AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;

                UPDATE flights SET is_blocked = TRUE
                        WHERE flight_id = flight_id_;
        END;
$$ LANGUAGE plpgsql;
```

7. Статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.

```
CREATE OR REPLACE FUNCTION flight_statistics() RETURNS TABLE (
        flight_id INT, is_blocked BOOLEAN, available BIGINT,
        booked BIGINT, sold BIGINT
) AS $$
        BEGIN
                SET TRANSACTION ISOLATION LEVEL SERIALIZABLE READ ONLY;

                RETURN QUERY (SELECT f.flight_id, f.is_blocked,
                         (SELECT count(*) FROM available_seats AS avs
                                 WHERE avs.flight_id = f.flight_id)
                        AS available,
                         (SELECT count(*) FROM available_bookings AS avb
                                 WHERE avb.flight_id = f.flight_id)
                        AS booked,
                         (SELECT count(*) FROM tickets AS t
                                 WHERE booking_last_update IS NULL
                                 AND t.flight_id = f.flight_id)
                        AS sold
                        FROM flights AS f);
        END;
$$ LANGUAGE plpgsql;
```