Домашнее задание № 9

Кокорин Илья, M3439

25 ноября 2019 г.

Весь код писался и тестировался в PostgreSQL 11.1 on x86_64-apple-darwin17.7.0, compiled by Apple LLVM version 10.0.0 (clang-1000.11.45.5), 64-bit

1 Схема базы данных

```
CREATE TABLE Planes
(
    plane_id            INT        NOT NULL PRIMARY KEY,
    registration_number VARCHAR(7) NOT NULL
);

CREATE TABLE Seats
(
    plane_id INT NOT NULL,
    seat_no  INT NOT NULL,
    PRIMARY KEY (plane_id, seat_no),
    FOREIGN KEY (plane_id) REFERENCES Planes (plane_id)
);

CREATE TABLE Flights
(
    flight_id               INT       NOT NULL PRIMARY KEY,
    flight_time             TIMESTAMP NOT NULL,
    plane_id                INT       NOT NULL,
    closed_by_administrator BOOLEAN   NOT NULL DEFAULT FALSE,
    FOREIGN KEY (plane_id) REFERENCES Planes (plane_id)
);

CREATE TABLE Users
(
    user_id        INT         NOT NULL PRIMARY KEY,
    first_name     VARCHAR(50) NOT NULL,
    surname        VARCHAR(50) NOT NULL,
    encrypted_pass TEXT        NOT NULL -- пароль + соль, используемый в функции crypt
);

CREATE TABLE Reservation
(
    flight_id             INT       NOT NULL,
    seat_no               INT       NOT NULL,
    reservation_timestamp TIMESTAMP NOT NULL,
    user_id               INT       NOT NULL,
    is_bought             BOOLEAN   NOT NULL,
    -- FALSE, если место забронировано, но пока не выкуплено,
    -- иначе TRUE
```

```
    FOREIGN KEY (user_id) REFERENCES Users (user_id),
    FOREIGN KEY (flight_id) REFERENCES Flights (flight_id)
);

ALTER TABLE Reservation
    ADD PRIMARY KEY (flight_id, seat_no) DEFERRABLE INITIALLY DEFERRED;
```

2 FreeSeats(FlightId) — список мест, доступных для продажи и бронирования

```
-- Заметим, что не бывает такой ситуации, когда место можно забронировать, но нельзя купить.
-- Ситуация, когда место можно купить, но нельзя забронировать, возникает, например,
-- за 3 часа до вылета.
CREATE TYPE AvailableActions AS ENUM ('Buy', 'BuyAndReserve', 'Nothing');

CREATE OR REPLACE FUNCTION get_available_actions(request_timestamp TIMESTAMP,
                                                 flight_timestamp TIMESTAMP)
    RETURNS AvailableActions AS
$$
DECLARE
    booking_closed_interval INTERVAL := INTERVAL '1 Day';
    buy_closed_interval     INTERVAL := INTERVAL '2 Hours';
BEGIN
    IF request_timestamp + booking_closed_interval < flight_timestamp THEN
        RETURN 'BuyAndReserve';
    ELSEIF request_timestamp + buy_closed_interval < flight_timestamp THEN
        RETURN 'Buy';
    ELSE
        RETURN 'Nothing';
    END IF;
END;
$$
    LANGUAGE plpgsql;

-- Возвращает список мест, доступных для продажи и бронирования.
-- Для каждого места говорит, что с ним можно сделать (либо купить, либо купить и забронировать).
-- Заметим, что, так как пользователь неизвестен, не бывает такой ситации,
-- когда мы можем выкупить ранее забронированное нами место или продлить его бронь.
CREATE OR REPLACE FUNCTION free_seats(flight_id_arg INT)
    RETURNS TABLE
            (
                seat_no           INT,
                available_actions AvailableActions
            )
AS
$$
DECLARE
    request_timestamp           TIMESTAMP := now();
    booking_expiration_interval INTERVAL  := INTERVAL '1 Day';
BEGIN
    RETURN QUERY
        SELECT Seats.seat_no                                AS seat_no,
               get_available_actions(request_timestamp,
                                     Flights.flight_time) AS available_actions
```

```
        FROM Flights
                 NATURAL JOIN Seats
                 LEFT OUTER JOIN Reservation USING (flight_id, seat_no)
        WHERE Flights.flight_id = flight_id_arg
          AND (NOT Flights.closed_by_administrator)
          AND get_available_actions(request_timestamp,
                                    Flights.flight_time) <> 'Nothing'
          AND (Reservation.user_id IS NULL OR -- в таблице Reservation в принципе нет такой записи,
               Reservation.user_id IS NOT NULL -- либо в таблице Reservation есть такая запись
                   AND NOT Reservation.is_bought -- место не выкуплено
                   AND Reservation.reservation_timestamp + -- а бронь истекла
                       booking_expiration_interval < request_timestamp);
END;
$$
    LANGUAGE plpgsql;
```

3 Reserve(UserId, Pass, FlightId, SeatNo) — пытается забронировать место. Возвращает истину, если удалось и ложь — в противном случае

```
-- Возвращает TRUE, если получилось создать пользователя, и
-- FALSE, если нет (если пользователь с таким id уже существует).
-- Заметим, что функция не падает с ошибкой если user_id не уникален,
-- а возвращает FALSE.
CREATE OR REPLACE FUNCTION
    create_user(user_id_arg INT,
                first_name_arg VARCHAR(50),
                surname_arg VARCHAR(50),
                pass_arg TEXT) RETURNS BOOLEAN AS
$$
DECLARE
    pass_hash       TEXT;
    new_users_count INT;
BEGIN
    pass_hash := crypt(pass_arg, gen_salt('bf', 8));

    INSERT INTO Users (user_id, first_name, surname, encrypted_pass)
    VALUES (user_id_arg, first_name_arg, surname_arg, pass_hash)
    ON CONFLICT DO NOTHING;

    GET DIAGNOSTICS new_users_count = ROW_COUNT;
    RETURN new_users_count = 1;
END;
$$
    LANGUAGE plpgsql;

-- Возвращает TRUE, если пользователь с таким user_id существует и пароль корректный,
-- FALSE, если нет (если пользователя не существует или пароль неверен).
CREATE OR REPLACE FUNCTION
    check_credentials(user_id_arg INT,
                      password_arg TEXT) RETURNS BOOLEAN AS
$$
DECLARE
    user_pass_hash TEXT;
```

```
BEGIN
    SELECT Users.encrypted_pass
    INTO user_pass_hash
    FROM Users
    WHERE Users.user_id = user_id_arg;

    RETURN user_pass_hash IS NOT NULL AND crypt(password_arg, user_pass_hash) = user_pass_hash;
END;
$$
    LANGUAGE plpgsql;

-- Проверяет, что рейс flight_id_arg существует, в самолёте,
-- совершающем данный рейс, есть место seat_no_arg,
-- бронирование (или покупка) на рейс flight_id_arg может быть
-- совершено во время action_timestamp_arg. (то есть осталось
-- больше суток/двух часов до вылета и
-- продажа на рейс не запрещена администратором)
CREATE OR REPLACE FUNCTION
    seat_exists_and_can_be_processed(flight_id_arg INT,
                                     seat_no_arg INT,
                                     action_timestamp_arg TIMESTAMP,
                                     interval_arg INTERVAL)
    RETURNS BOOLEAN AS
$$
BEGIN
    RETURN EXISTS(
            SELECT *
            FROM Flights
                     NATURAL JOIN Seats
            WHERE Flights.flight_id = flight_id_arg
              AND Seats.seat_no = seat_no_arg
              AND action_timestamp_arg + interval_arg < Flights.flight_time
              AND NOT Flights.closed_by_administrator
        );
END;
$$
    LANGUAGE plpgsql;

-- Производит операцию со свободным (то есть не купленным и не забронированным) сидением.
-- Если is_bought_arg = TRUE, ты покупаем, иначе бронируем.
-- action_closed_interval обозначает промежуток времени до вылета,
-- за который закрывается эта операция (для покупки - два часа, для бронирования - сутки)
-- Функция фозвращает TRUE если операция прошла удачно, иначе FALSE.
CREATE OR REPLACE FUNCTION process_free_seat(user_id_arg INT,
                                             pass_arg TEXT,
                                             flight_id_arg INT,
                                             seat_no_arg INT,
                                             action_closed_interval INTERVAL,
                                             is_bought_arg BOOLEAN)
    RETURNS BOOLEAN
AS
$$
DECLARE
    booking_expiration_interval INTERVAL  := INTERVAL '1 Day';
    request_timestamp           TIMESTAMP := now();
    seat_is_bought              BOOLEAN;
```

```
    seat_reservation_timestamp  TIMESTAMP;
BEGIN
    IF check_credentials(user_id_arg, pass_arg) AND
       seat_exists_and_can_be_processed(flight_id_arg,
                                        seat_no_arg,
                                        request_timestamp,
                                        action_closed_interval) THEN
        SELECT Reservation.is_bought,
               Reservation.reservation_timestamp
        INTO seat_is_bought,
            seat_reservation_timestamp
        FROM Reservation
        WHERE Reservation.flight_id = flight_id_arg
          AND Reservation.seat_no = seat_no_arg;

        IF seat_is_bought IS NULL THEN
            INSERT INTO Reservation (flight_id,
                                     seat_no,
                                     reservation_timestamp,
                                     user_id,
                                     is_bought)
            VALUES (flight_id_arg,
                    seat_no_arg,
                    request_timestamp,
                    user_id_arg,
                    is_bought_arg);

            RETURN TRUE;
        ELSEIF NOT seat_is_bought AND
               seat_reservation_timestamp
                   + booking_expiration_interval < request_timestamp THEN
            UPDATE Reservation
            SET user_id               = user_id_arg,
                reservation_timestamp = request_timestamp,
                is_bought             = is_bought_arg
            WHERE flight_id = flight_id_arg
              AND seat_no = seat_no_arg;

            RETURN TRUE;
        ELSE
            RETURN FALSE;
        END IF;
    ELSE
        RETURN FALSE;
    END IF;
END;
$$
    LANGUAGE plpgsql;

-- Пытается забронировать место, то есть создать новую бронь, а не продлить старую.
-- Возвращает TRUE, если удалось и FALSE - в противном случае
CREATE OR REPLACE FUNCTION reserve(user_id_arg INT,
                                   pass_arg TEXT,
                                   flight_id_arg INT,
                                   seat_no_arg INT)
    RETURNS BOOLEAN
```

```
AS
$$
DECLARE
    reservation_closed_interval INTERVAL := INTERVAL '1 Day';
BEGIN
    RETURN process_free_seat(
            user_id_arg,
            pass_arg,
            flight_id_arg,
            seat_no_arg,
            reservation_closed_interval,
            FALSE
        );
END;
$$
    LANGUAGE plpgsql;
```

4 ExtendReservation(UserId, Pass, FlightId, SeatNo) — пытается продлить бронь места. Возвращает истину, если удалось и ложь — в противном случае

```
-- Пытается продлить бронь места.
-- Возвращает TRUE, если удалось и FALSE - в противном случае
CREATE OR REPLACE FUNCTION extend_reservation(user_id_arg INT,
                                              pass_arg TEXT,
                                              flight_id_arg INT,
                                              seat_no_arg INT)
    RETURNS BOOLEAN
AS
$$
DECLARE
    request_timestamp           TIMESTAMP := now();
    booking_expiration_interval INTERVAL  := INTERVAL '1 Day';
    booking_closed_interval     INTERVAL  := INTERVAL '1 Day';
BEGIN
    IF check_credentials(user_id_arg, pass_arg) AND EXISTS(
            SELECT *
            FROM Reservation
                     NATURAL JOIN Flights
            WHERE Reservation.flight_id = flight_id_arg
              AND Reservation.seat_no = seat_no_arg
              AND Reservation.user_id = user_id_arg
              AND NOT Flights.closed_by_administrator
              AND NOT Reservation.is_bought
              AND Reservation.reservation_timestamp +
                  booking_expiration_interval >= request_timestamp
              AND request_timestamp + booking_closed_interval <
                  Flights.flight_time
        ) THEN
        UPDATE Reservation
        SET reservation_timestamp = request_timestamp
        WHERE flight_id = flight_id_arg
          AND seat_no = seat_no_arg;
    ELSE
```

```
        RETURN FALSE;
    END IF;
END;
$$
    LANGUAGE plpgsql;
```

5 BuyFree(UserId, Pass, FlightId, SeatNo) — пытается купить свободное место. Возвращает истину, если удалось и ложь — в противном случае.

```
-- пытается купить свободное место.
-- Возвращает TRUE, если удалось и FALSE - в противном случае
CREATE OR REPLACE FUNCTION buy_free(user_id_arg INT,
                                    pass_arg TEXT,
                                    flight_id_arg INT,
                                    seat_no_arg INT)
    RETURNS BOOLEAN
AS
$$
DECLARE
    buy_closed_interval INTERVAL := INTERVAL '2 Hours';
BEGIN
    RETURN process_free_seat(
            user_id_arg,
            pass_arg,
            flight_id_arg,
            seat_no_arg,
            buy_closed_interval,
            TRUE
        );
END;
$$
    LANGUAGE plpgsql;
```

6 BuyReserved(UserId, Pass, FlightId, SeatNo) — пытается выкупить забронированное место (пользователи должны совпадать). Возвращает истину, если удалось и ложь — в противном случае.

```
-- Пытается выкупить забронированное место.
-- Возвращает TRUE, если удалось и FALSE - в противном случае
CREATE OR REPLACE FUNCTION buy_reserved(user_id_arg INT,
                                        pass_arg TEXT,
                                        flight_id_arg INT,
                                        seat_no_arg INT)
    RETURNS BOOLEAN
AS
$$
DECLARE
    request_timestamp           TIMESTAMP := now();
    booking_expiration_interval INTERVAL  := INTERVAL '1 Day';
    buy_closed_interval         INTERVAL  := INTERVAL '2 Hours';
```

```
BEGIN
    IF check_credentials(user_id_arg, pass_arg) AND EXISTS(
            SELECT *
            FROM Reservation
                     NATURAL JOIN Flights
            WHERE Reservation.flight_id = flight_id_arg
              AND Reservation.seat_no = seat_no_arg
              AND Reservation.user_id = user_id_arg
              AND NOT Flights.closed_by_administrator
              AND NOT Reservation.is_bought
              AND Reservation.reservation_timestamp +
                  booking_expiration_interval >= request_timestamp
              AND request_timestamp + buy_closed_interval <
                  Flights.flight_time
        ) THEN
        UPDATE Reservation
        SET reservation_timestamp = request_timestamp,
            is_bought             = TRUE
        WHERE flight_id = flight_id_arg
          AND seat_no = seat_no_arg;
    ELSE
        RETURN FALSE;
    END IF;
END;
$$
    LANGUAGE plpgsql;
```

7 FlightStatistics(UserId, Pass) — статистика по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест

```
-- В таблице Reservation может не быть записи, так что
-- is_bought_arg, user_that_reserved_arg и reservation_timestamp_arg могут быть NULL.
-- Считаем, что если пользователь, делающий запрос, уже бронировал это место и бронь не истекла,
-- то он может забронировать это место (продлить бронь).
-- Считаем, что человек не может забронировать или купить ничего,
-- если у него неправильный логин или пароль.
-- Возвращает 1, если место может быть обработано неким образом (куплено или зарезервировано)
-- данным пользователем, 0 иначе. Считаем, что если данный пользователь забронировал место,
-- то он может его как купить (выкупить бронь), так и забронировать (продлить бронь)
-- process_closed_interval - время до вылета, за которое закрывается возможность
-- совершить данное действие (2 часа для покупки, 1 день - для брони).
CREATE OR REPLACE FUNCTION count_available_to_process(closed_by_administrator_arg BOOLEAN,
                                                      flight_timestamp_arg TIMESTAMP,
                                                      user_id_arg INT,
                                                      pass_arg TEXT,
                                                      user_that_reserved_arg INT,
                                                      is_bought_arg BOOLEAN,
                                                      reservation_timestamp_arg TIMESTAMP,
                                                      request_timestamp_arg TIMESTAMP,
                                                      process_closed_interval INTERVAL)
    RETURNS INT AS
$$
DECLARE
```

```
    reservation_expires_interval INTERVAL := INTERVAL '1 Day';
BEGIN
    IF check_credentials(user_id_arg, pass_arg)
        AND NOT closed_by_administrator_arg
        AND request_timestamp_arg + process_closed_interval <= flight_timestamp_arg
        AND (is_bought_arg IS NULL OR NOT is_bought_arg) THEN
        IF user_that_reserved_arg IS NULL THEN
            RETURN 1;
        ELSEIF reservation_timestamp_arg +
               reservation_expires_interval < request_timestamp_arg THEN
            RETURN 1;
        ELSEIF user_that_reserved_arg = user_id_arg THEN
            RETURN 1;
        ELSE
            RETURN 0;
        END IF;
    ELSE
        RETURN 0;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION count_available_to_reserve(closed_by_administrator_arg BOOLEAN,
                                                      flight_timestamp_arg TIMESTAMP,
                                                      user_id_arg INT,
                                                      pass_arg TEXT,
                                                      user_that_reserved_arg INT,
                                                      is_bought_arg BOOLEAN,
                                                      reservation_timestamp_arg TIMESTAMP,
                                                      request_timestamp_arg TIMESTAMP)
    RETURNS INT AS
$$
DECLARE
    reservation_closed_interval INTERVAL := INTERVAL '1 Day';
BEGIN
    RETURN count_available_to_process(
            closed_by_administrator_arg,
            flight_timestamp_arg,
            user_id_arg,
            pass_arg,
            user_that_reserved_arg,
            is_bought_arg,
            reservation_timestamp_arg,
            request_timestamp_arg,
            reservation_closed_interval
        );
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION count_available_to_buy(closed_by_administrator_arg BOOLEAN,
                                                  flight_timestamp_arg TIMESTAMP,
                                                  user_id_arg INT,
                                                  pass_arg TEXT,
                                                  user_that_reserved_arg INT,
                                                  is_bought_arg BOOLEAN,
                                                  reservation_timestamp_arg TIMESTAMP,
```

```
                                                    request_timestamp_arg TIMESTAMP)
    RETURNS INT AS
$$
DECLARE
    buy_closed_interval INTERVAL := INTERVAL '2 Hours';
BEGIN
    RETURN count_available_to_process(
            closed_by_administrator_arg,
            flight_timestamp_arg,
            user_id_arg,
            pass_arg,
            user_that_reserved_arg,
            is_bought_arg,
            reservation_timestamp_arg,
            request_timestamp_arg,
            buy_closed_interval
        );
END;
$$ LANGUAGE plpgsql;

-- статистика по рейсам: число возможных для бронирования и покупки данным человеком,
-- число свободных, забронированных и проданных мест
CREATE OR REPLACE FUNCTION flight_statistic(user_id_arg INT, pass_arg TEXT)
    RETURNS TABLE
            (
                flight_id                   INT,
                available_to_reserve_seats INT,
                available_to_buy_seats     INT,
                free_seats                  INT,
                reserved_seats              INT,
                bought_seats                INT
            )
AS
$$
DECLARE
    request_timestamp TIMESTAMP := now();
BEGIN
    RETURN QUERY
        WITH temp_stats AS (
            SELECT Flights.flight_id,
                   sum(
                           count_available_to_reserve(
                                   Flights.closed_by_administrator,
                                   Flights.flight_time,
                                   user_id_arg,
                                   pass_arg,
                                   Reservation.user_id,
                                   Reservation.is_bought,
                                   Reservation.reservation_timestamp,
                                   request_timestamp
                               )
                       ):: INT     AS available_to_reserve_seats,
                   sum(
                           count_available_to_buy(
                                   Flights.closed_by_administrator,
                                   Flights.flight_time,
```

```
                                    user_id_arg,
                                    pass_arg,
                                    Reservation.user_id,
                                    Reservation.is_bought,
                                    Reservation.reservation_timestamp,
                                    request_timestamp
                                )
                            ) :: INT    AS available_to_buy_seats,
                        sum(
                                count_bought(
                                    Reservation.is_bought
                                )
                            ) :: INT    AS bought_seats,
                        sum(
                                count_reserved(
                                    Flights.closed_by_administrator,
                                    Flights.flight_time,
                                    Reservation.reservation_timestamp,
                                    Reservation.is_bought,
                                    request_timestamp
                                )
                            ) :: INT    AS reserved_seats,
                        count(*) :: INT AS total_seats
                FROM Flights
                         NATURAL JOIN Seats
                         LEFT OUTER JOIN Reservation USING (flight_id, seat_no)
                GROUP BY Flights.flight_id
        )
        SELECT temp_stats.flight_id,
               temp_stats.available_to_reserve_seats,
               temp_stats.available_to_buy_seats,
               temp_stats.total_seats - temp_stats.reserved_seats
                   - temp_stats.bought_seats AS free_seats,
               temp_stats.reserved_seats,
               temp_stats.bought_seats
        FROM temp_stats;
END;
$$ LANGUAGE plpgsql;
```

8 FlightStat(UserId, Pass, FlightId) — статистика по рейсу: возможность бронирования и покупки, число свободных, забронированных и проданных мест.

```
CREATE TYPE FlightStat AS (
    available_to_reserve_seats INT,
    available_to_buy_seats INT,
    free_seats INT,
    reserved_seats INT,
    bought_seats INT
    );

-- статистика по рейсу: число возможных для бронирования и покупки данным человеком,
-- число свободных, забронированных и проданных мест
CREATE OR REPLACE FUNCTION flight_stat(user_id_arg INT, pass_arg TEXT, flight_id_arg INT)
```

```
    RETURNS FlightStat
AS
$$
DECLARE
    result FlightStat;
BEGIN
    SELECT available_to_reserve_seats,
           available_to_buy_seats,
           free_seats,
           reserved_seats,
           bought_seats
    INTO result.available_to_reserve_seats,
        result.available_to_buy_seats,
        result.free_seats,
        result.reserved_seats,
        result.bought_seats
    FROM flight_statistic(user_id_arg, pass_arg)
    WHERE flight_id = flight_id_arg;

    IF result.available_to_reserve_seats IS NOT NULL THEN
        RETURN result;
    ELSE
        RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

9 CompressSeats(FlightId) — оптимизирует занятость мест в самолете. В результате оптимизации, в начале самолета должны быть купленные места, затем — забронированные, а в конце — свободные. Примечание: клиенты, которые уже выкупили билеты так же должны быть пересажены.

```
-- оптимизирует занятость мест в самолете.
-- В результате оптимизации, в начале самолета должны быть купленные места,
-- затем - забронированные, а в конце - свободные
CREATE OR REPLACE PROCEDURE
    compress_seats(flight_id_arg INT) AS
$$
DECLARE
    request_timestamp   TIMESTAMP := now();
    reservations_cursor CURSOR FOR
        SELECT Reservation.seat_no
        FROM Reservation
        WHERE Reservation.flight_id = flight_id_arg
          AND (Reservation.is_bought OR
               Reservation.reservation_timestamp + '1 Day' >= request_timestamp)
        ORDER BY is_bought DESC;
    seats_cursor CURSOR FOR
        SELECT Seats.seat_no
        FROM Flights
                 NATURAL JOIN Seats
        WHERE Flights.flight_id = flight_id_arg
```

```
        ORDER BY seat_no;
    reservation_seat_no INT;
    plane_seat_no       INT;
BEGIN
    SET CONSTRAINTS ALL DEFERRED;
    DELETE
    FROM Reservation
    WHERE Reservation.flight_id = flight_id_arg
      AND NOT Reservation.is_bought
      AND Reservation.reservation_timestamp
              + INTERVAL '1 Day' < request_timestamp;

    OPEN reservations_cursor;
    OPEN seats_cursor;
    LOOP
        FETCH reservations_cursor INTO reservation_seat_no;
        FETCH seats_cursor INTO plane_seat_no;

        IF NOT FOUND THEN
            EXIT;
        END IF;

        UPDATE Reservation
        SET seat_no = plane_seat_no
        WHERE flight_id = flight_id_arg
          AND seat_no = reservation_seat_no;

    END LOOP;
    CLOSE reservations_cursor;
    CLOSE seats_cursor;
END;
$$
    LANGUAGE plpgsql;
```