ДЗ 9

Содержание

Весь sql-код написан и протестирован на `PostgreSQL 9.5.4`.

1 Создание таблиц и представлений, заполнение данными

```
CREATE TABLE Seats(
    PlaneId int NOT NULL PRIMARY KEY,
    SeatNum int NOT NULL
);

CREATE TABLE Flights(
  FlightId int NOT NULL PRIMARY KEY,
  FlightTime timestamp NOT NULL,
  PlaneId int NOT NULL,
  SoldOut boolean NOT NULL,

  FOREIGN KEY (PlaneId) REFERENCES Seats(PlaneId) ON DELETE CASCADE
);

CREATE TABLE Tickets(
    UserId int NOT NULL,
    FlightId int NOT NULL,
    SeatId int NOT NULL,
    Expiration timestamp,
    IsPaid boolean NOT NULL DEFAULT False,

    PRIMARY KEY (UserId, FlightId, SeatId),
    FOREIGN KEY (FlightId) REFERENCES Flights(FlightId) ON DELETE CASCADE,
    UNIQUE(FlightId, SeatId) -- место не может быть продано/забронировано дважды
);

DELETE FROM Seats WHERE True;
INSERT INTO Seats VALUES
  (111, 10),
  (222, 15),
  (333, 20);

DELETE FROM Flights WHERE True;
INSERT INTO Flights VALUES
  (0, '2016-12-06 12:00:00', 111, False),
  (1, '2016-12-07 22:00:00', 222, False),
  (2, '2016-12-08 06:30:00', 333, False),
  (3, '2016-12-07 06:30:00', 222, True);

DELETE FROM Tickets WHERE True;
INSERT INTO Tickets VALUES
  (0,0,2,null,True),
  (1,0,3,'2016-12-06 04:00:00',False),
  (2,2,5,'2016-12-06 12:00:00',False);

-- Flights that we can manipulate tickets for.
CREATE VIEW ValidFlights AS
SELECT FlightId, PlaneId FROM Flights
WHERE NOT SoldOut AND
      FlightTime - now() > INTERVAL '2 hour';

-- (planeId, seatId) for every accessible seat in the plane
-- let's suppose there can't be more than 100 seats in plane
-- (we can use 1000 instead)
CREATE VIEW AllSeats AS
SELECT S.PlaneId, n AS SeatId
FROM generate_series(1,100) AS a(n), Seats AS S
WHERE n <= S.SeatNum
ORDER BY S.PlaneId;

-- Valid reservations + sold out seats.
CREATE VIEW NonFreeSeats AS
SELECT FlightId, SeatId
FROM ValidFlights NATURAL JOIN Flights NATURAL JOIN Tickets
WHERE IsPaid OR                          -- reedemed tickets
```

```
      (Expiration IS NOT NULL AND       -- valid reservations
       now() < Expiration AND
       Expiration < FlightTime AND
       FlightTime - now() > INTERVAL '1 day');

-- Seats that we can buy (only).
CREATE VIEW PurchaseAvailable AS
SELECT FlightId, SeatId
FROM ValidFlights NATURAL JOIN AllSeats
WHERE (FlightId,SeatId) NOT IN (SELECT * from NonFreeSeats);

-- Seats that can be bought _and_ reserved. PurchaseAvailable \ those
-- that can't be booked.
CREATE VIEW ReservationAvailable AS
SELECT FlightId, SeatId
FROM ValidFlights NATURAL JOIN Flights NATURAL JOIN AllSeats
WHERE FlightTime - now() > INTERVAL '1 day' AND
      (FlightId,SeatId) NOT IN (SELECT * FROM NonFreeSeats);
```

2 Запросы/функции

1. Список мест, доступных для продажи и бронирования. Есть два варианта интеретации формулировки:

    1. Доступны для (продажи и бронирования).
    2. Доступны для продажи и доступны для бронирования – доступны для продажи или бронирования.

   Вот реализация для (2). Для (1) надо заменить `PurchaseAvailable` на `ReservationAvailable`.

```
CREATE FUNCTION FreeSeatsAll(fId INT)
  RETURNS TABLE (chairId INT) AS
$func$
BEGIN
    RETURN QUERY SELECT SeatId FROM PurchaseAvailable WHERE FlightId = fId;
END
$func$ LANGUAGE plpgsql;
```

2. Попытка зарезервировать место.

```
CREATE FUNCTION Reserve(uId INT, fId INT, sId INT)
  RETURNS boolean AS
$func$
BEGIN
   INSERT INTO Tickets (UserId, FlightId, SeatId, Expiration, IsPaid)
     SELECT uId,fId,sId,now() + INTERVAL '1 day',FALSE
     ON CONFLICT (FlightId,SeatId) DO
       UPDATE SET Expiration = now() + INTERVAL '1 day',
                  UserId = uId
     WHERE (fId,sId) IN (SELECT * FROM ReservationAvailable);

   IF FOUND THEN RETURN TRUE;
   ELSE RETURN FALSE;
   END IF;
END
$func$ LANGUAGE plpgsql;
```

3. Продление брони.

```
CREATE FUNCTION ExtendReservation(fId INT, sId INT)
  RETURNS boolean AS
$func$
BEGIN
   UPDATE Tickets
   SET Expiration = now() + INTERVAL '1 day'
   WHERE
       FlightId = fId AND
       SeatId = sId AND
       IsPaid = FALSE AND
       (fId, sId) in (SELECT * FROM NonFreeSeats);

   IF FOUND THEN RETURN TRUE;
   ELSE RETURN FALSE;
   END IF;
END
$func$ LANGUAGE plpgsql;
```

4. Покупка свободного места.

```
CREATE FUNCTION BuyFree(uId INT, fId INT, sId INT)
  RETURNS boolean AS
$func$
BEGIN
   INSERT INTO Tickets (UserId, FlightId, SeatId, IsPaid)
     SELECT uId,fId,sId,TRUE
     ON CONFLICT (FlightId,SeatId) DO
       UPDATE SET Expiration = null
                , UserId = uId
                , IsPaid = TRUE
     WHERE (fId,sId) IN (SELECT * FROM FreeSeats);

   IF FOUND THEN RETURN TRUE;
   ELSE RETURN FALSE;
   END IF;
END
$func$ LANGUAGE plpgsql;
```

5. Покупка зарезервированного места.

```
CREATE FUNCTION BuyReserved(uId INT, fId INT, sId INT)
  RETURNS boolean AS
$func$
BEGIN
   UPDATE Tickets
   SET Expiration = NULL, IsPaid = TRUE, UserId = uId
   WHERE
       UserId = uId AND
       FlightId = fId AND
       SeatId = sId AND
       IsPaid = FALSE AND
       (fId, sId) in (SELECT * FROM NonFreeSeats);

   IF FOUND THEN RETURN TRUE;
   ELSE RETURN FALSE;
   END IF;
END
$func$ LANGUAGE plpgsql;
```

6. Статистика по полетам.

```
CREATE FUNCTION FlightStatistics()
  RETURNS TABLE ( Id INT
                , Active BOOLEAN
                , FreeN BIGINT
                , ReservedN BIGINT
                , SoldN BIGINT) AS
$func$
BEGIN
    RETURN QUERY (
        SELECT
          f.FlightId
          , NOT f.SoldOut
          , (SELECT COUNT(*)
             FROM PurchaseAvailable
             WHERE FlightId = f.FlightId
             ) AS FreeN
          , (SELECT COUNT(*)
             FROM ValidFlights NATURAL JOIN Flights NATURAL JOIN Tickets
             WHERE NOT IsPaid AND
                   (Expiration IS NOT NULL AND
                    now() < Expiration AND
                    Expiration < FlightTime) AND
                   FlightId = f.FlightId
             ) AS ReservedN
          , (SELECT COUNT(*)
             FROM ValidFlights NATURAL JOIN Flights NATURAL JOIN Tickets
             WHERE IsPaid AND FlightId = f.FlightId
             ) AS SoldN
         FROM Flights AS f);
END
$func$ LANGUAGE plpgsql;
```

7. Оптимизация занятости мест.

SQL

```sql
CREATE FUNCTION CompressSeats() RETURNS VOID AS
$func$
DECLARE
  flight Flights%rowtype;
  ticket Tickets%rowtype;
-- https://www.postgresql.org/docs/current/static/plpgsql-structure.html
-- Body will run in transaction so we don't need any extra locks
BEGIN
    FOR flight IN (SELECT FlightId FROM ValidFlights) LOOP

        CREATE TEMPORARY TABLE TicketsNew (
            UserId int NOT NULL,
            FlightId int NOT NULL,
            SeatId serial NOT NULL,
            FormerSeatId int NOT NULL,
            Expiration timestamp,
            IsPaid boolean NOT NULL DEFAULT False,

            PRIMARY KEY (UserId, FlightId, SeatId),
            UNIQUE(FlightId, SeatId)
        );
        -- start with seat 1 (not that important though)
        ALTER SEQUENCE TicketsNew_seatid_seq RESTART WITH 1;

        -- Aggrgegate all sold tickets
        FOR ticket
        IN (SELECT *
            FROM Tickets
            WHERE IsPaid AND FlightId = flight.FlightId)
        LOOP
            INSERT INTO TicketsNew (UserId,FlightId,FormerSeatId,Expiration,IsPaid)
                VALUES (ticket.UserId, ticket.FlightId, ticket.SeatId, ticket.Expiration, ticket.IsPaid);
        END LOOP;

        -- Aggrgegate all reserved tickets
        FOR ticket
        IN (SELECT *
            FROM Tickets
            WHERE NOT IsPaid AND
                  FlightId = flight.FlightId AND
                  (Expiration IS NOT NULL AND
                   now() < Expiration AND
                   Expiration < flight.FlightTime))
        LOOP
            INSERT INTO TicketsNew (UserId,FlightId,FormerSeatId,Expiration,IsPaid)
                VALUES (ticket.UserId, ticket.FlightId, ticket.SeatId, ticket.Expiration, ticket.IsPaid);
        END LOOP;

        -- Remove all tickets from real table that we've added to temp
        -- table.
        DELETE FROM Tickets
        WHERE (UserId, FlightId, SeatId)
        IN (SELECT UserId,FlightId,FormerSeatId FROM TicketsNew);

        -- Put updated data
        INSERT INTO Tickets
        SELECT UserId,FlightId,SeatId,Expiration,IsPaid FROM TicketsNew;

        DROP TABLE TicketsNew;
    END LOOP;
    RETURN;
END
$func$ LANGUAGE plpgsql;
```

Автор: Михаил Волхов M3438

Created: 2016-12-06 Tue 12:09

Validate