Филиппов Дмитрий, М3439

Домашнее задание 9.

Схема БД: Flights(FlightId, FligtTime, PlaneId, ClosedByRequest), Seats(PlaneId, SeatNo), Transaction(FlightId, PlaneId, SeatNo, PassportSeries, PassportNo, TransTime, TransType).
Использованная БД: PostgreSQL 9.4.5.

Реализуйте запросы к базе данных Airline с применением хранимых процедур и функций.

Во всех процедурах учитывались условия прошлого домашнего задания (возможность закрытия регистрации администратором, невозможность бронирования за 24 часа до вылета, невозможность покупки за 2 часа до вылета).

Описание БД из прошлого дз:
Схема изначально данной БД: Flights(FlightId, FligtTime, PlaneId), Seats(PlaneId, SeatNo).

- Добавим таблицу *Transaction*, содержающую информацию о покупке/бронировании места;
- А именно, в ней мы будем хранить:
  - информацию о полете — *FlightId*, *PlaneId*, которые также будут ссылаться на таблицу *Flights*;
  - информацию о месте — *PlaneId*, *SeatNo*, которые также будут ссылаться на таблицу *Seats*;
  - время и тип транзакции — *TransTime*, *TransType*, где *TransType* — либо бронирование, либо покупка.
- Также нам нужно поддерживать возможность закрытия продаж по запросу администратора, для этого в таблицу *Flights* добавим флаг *ClosedByRequest*.

Код создания БД:

```
CREATE TABLE Flights (
  FlightId INT NOT NULL,
  FlightTime TIMESTAMP DEFAULT '1970-01-01␣00:00:01',
  PlaneId INT NOT NULL,
  ClosedByRequest BOOLEAN DEFAULT FALSE,
  PRIMARY KEY (FlightId)
);

CREATE TABLE Seats (
  PlaneId INT NOT NULL,
  SeatNo INT NOT NULL,
  PRIMARY KEY (PlaneId, SeatNo)
);

CREATE TABLE Transaction (
  FlightId INT NOT NULL,
  PlaneId INT NOT NULL,
  SeatNo INT NOT NULL,
  TransTime TIMESTAMP DEFAULT now(),
  TransType INT DEFAULT 0,
```

```
      //0 - reservation, 1 - bying
      //Sorry for commenting using not double dashes
      //I'm␣not␣good␣in␣latex␣:(
␣␣␣␣␣␣FOREIGN␣KEY␣(FlightId)␣REFERENCES␣Flights,
␣␣␣␣␣␣FOREIGN␣KEY␣(PlaneId,␣SeatNo)␣REFERENCES␣Seats
␣␣␣␣␣␣ON␣DELETE␣CASCADE
␣␣␣␣␣␣ON␣UPDATE␣CASCADE,
␣␣␣␣␣␣PRIMARY␣KEY␣(FlightId,␣SeatNo)
␣␣␣␣);
```

Сначала реализуем функцию, проверяющую, что бронь истекла, которая не относится ни к одному заданию, но будет использоваться:

```
CREATE FUNCTION IsReservationExpired(FlightId INT, SeatNo INT)
RETURNS BOOLEAN AS $BODY$
  DECLARE FlightTime TIMESTAMP;
  DECLARE LastReservationUpdate TIMESTAMP;
  DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
BEGIN
  SELECT (F.FlightTime, F.ClosedByRequest) INTO
    FlightTime, IsClosedByRequest
  FROM
    Flights as F
  WHERE
    F.FlightId = FlightId;

  SELECT T.TransTime INTO
    LastReservationUpdate
  FROM
    Transaction as T
  WHERE
    T.FlightId = FlightId AND
    T.SeatNo = SeatNo AND
    T.TransType = 0;

  RETURN IsClosedByRequest OR
         (LastReservationUpdate IS NOT NULL AND
         LastReservationUpdate + INTERVAL '24␣hours' < NOW()) OR
         NOW() + INTERVAL '24␣hours' > FlightTime;
END;
$BODY$ LANGUAGE plpgsql;
```

1. **FreeSeats(FlightId) — список мест, доступных для продажи и бронирования.**

```
CREATE FUNCTION FreeSeats(FlightId INT)
RETURNS TABLE (SeatNo INT) AS $BODY$
  DECLARE PlaneId INT DEFAULT NULL;
BEGIN
  SELECT (PlaneId) INTO
    PlaneId
  FROM
    Flights as F
  WHERE
```

```
        F.FlightId = FlightId;

      RETURN QUERY
      SELECT
        S.SeatNo
      FROM
        Seats as S
      WHERE
        S.PlaneId = PlaneId
      EXCEPT
      SELECT
        T.SeatNo
      FROM
        Transaction as T
      WHERE
        T.FlightId = FlightId AND
        T.PlaneId = PlaneId AND
        (T.TransType = 1 OR
         (T.TransType = 0 AND NOT IsReservationExpired(T.FlightId, T.SeatNo)));
END;
$BODY$ LANGUAGE plpgsql;
```

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

Если старая бронь истекла, удаляет ее из *Transactions*. В случае, если бронирование успешно, добавляет в таблицу бронь.

```
    CREATE FUNCTION Reserve(FlightId INT, SeatNo INT)
    RETURNS BOOLEAN AS $BODY$
      DECLARE FlightTime TIMESTAMP;
      DECLARE PlaneId INT;
      DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
      DECLARE TransTime TIMESTAMP DEFAULT NULL;
      DECLARE TransType INT DEFAULT NULL;
    BEGIN
      SELECT (F.FlightTime, F.isClosedByRequest) INTO
        FlightTime, IsClosedByRequest
      FROM
        Flights as F
      WHERE
        F.FlightId = FlightId;

      IF NOT IsClosedByRequest AND NOW() <= FlightTime - INTERVAL '24␣hours' THEN
        SELECT (T.PlaneId, T.TransTime, T.TransType) INTO
          PlaneId, TransTime, TransType
        FROM
          Transaction as T
        WHERE
          T.FlightId = FlightId AND
          T.SeatNo = SeatNo;
        IF (TransTime IS NULL AND
```

```
            TransType IS NULL) OR
           (TransType = 0 AND
            IsReservationExpired(FlightId, SeatNo)) THEN
          DELETE FROM
            Transaction as T
          WHERE
            T.FlightId = FlightId AND
            T.SeatNo = SeatNo;

          INSERT INTO Transaction
            (FlightId, PlaneId, SeatNo, TransTime, TransType)
          VALUES
            (FlightId, PlaneId, SeatNo, now(), 0);
          RETURN TRUE;
        ELSE
          //Seat is already reserved (maybe by yourself, but it doesn't matter)
␣␣␣␣␣␣␣␣␣␣RETURN␣FALSE;
␣␣␣␣␣␣␣␣END␣IF;
␣␣␣␣␣␣␣ELSE
␣␣␣␣␣␣␣␣//Reserving␣is␣closed␣or␣it␣is␣too␣late
␣␣␣␣␣␣␣␣RETURN␣FALSE;
␣␣␣␣␣␣␣END␣IF;
␣␣␣␣END;
␣␣␣␣$BODY$␣LANGUAGE␣plpgsql;
```

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

Если бронь истекла, продлить ее нельзя, она удаляется из таблицы *Transactions*. Иначе, обновляется, устанавливается $TransTime = now()$.

```
    CREATE FUNCTION ExtendReservation(FlightId INT, SeatNo INT)
    RETURNS BOOLEAN AS $BODY$
      DECLARE PlaneId INT;
    BEGIN
      IF IsReservationExpired(FlightId, SeatNo) THEN
        //Reservation is expired, we can't␣update␣it.
␣␣␣␣␣␣␣␣DELETE␣FROM
␣␣␣␣␣␣␣␣␣␣Transaction␣as␣T
␣␣␣␣␣␣␣␣WHERE
␣␣␣␣␣␣␣␣␣␣T.FlightId␣=␣FlightId␣AND
␣␣␣␣␣␣␣␣␣␣T.SeatNo␣=␣SeatNo␣AND
␣␣␣␣␣␣␣␣␣␣T.TransType␣=␣0;
␣␣␣␣␣␣␣␣RETURN␣FALSE;
␣␣␣␣␣␣␣ELSE
␣␣␣␣␣␣␣␣UPDATE
␣␣␣␣␣␣␣␣␣␣Transaction␣as␣T
␣␣␣␣␣␣␣␣SET
␣␣␣␣␣␣␣␣␣␣T.TransTime␣=␣now()
␣␣␣␣␣␣␣␣WHERE
␣␣␣␣␣␣␣␣␣␣T.FlightId␣=␣FlightId␣AND
␣␣␣␣␣␣␣␣␣␣T.SeatNo␣=␣SeatNo␣AND
```

```
␣␣␣␣␣␣␣␣␣␣T.TransType␣=␣0;
␣␣␣␣␣␣␣␣RETURN␣TRUE;
␣␣␣␣␣␣END␣IF;
␣␣␣␣END;
␣␣␣␣$BODY$␣LANGUAGE␣plpgsql;
```

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

```
CREATE FUNCTION BuyFree(FlightId INT, SeatNo INT)
RETURNS BOOLEAN AS $BODY$
  DECLARE PlaneId INT;
  DECLARE FlightTime TIMESTAMP;
  DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
  DECLARE TransTime TIMESTAMP DEFAULT NULL;
  DECLARE TransType INT DEFAULT NULL;
BEGIN
  SELECT (F.FlightTime, F.isClosedByRequest) INTO
    FlightTime, IsClosedByRequest
  FROM
    Flights as F
  WHERE
    F.FlightId = FlightId;

  IF NOT IsClosedByRequest AND NOW() <= FlightTime - INTERVAL '2 hours' THEN
    SELECT (T.PlaneId, T.TransTime, T.TransType) INTO
      PlaneId, TransTime, TransType
    FROM
      Transaction as T
    WHERE
      T.FlightId = FlightId AND
      T.SeatNo = SeatNo;
    IF TransTime IS NULL AND TransType IS NULL THEN
      //Seat is free
      INSERT INTO Transaction
        (FlightId, PlaneId, SeatNo, TransTime, TransType)
      VALUES
        (FlightId, PlaneId, SeatNo, now(), 1);
      RETURN TRUE;
    ELSE
      //Seat it already bought or reserved, can't buy it
␣␣␣␣␣␣␣␣␣␣RETURN␣FALSE;
␣␣␣␣␣␣␣␣END␣IF;
␣␣␣␣␣␣␣ELSE
␣␣␣␣␣␣␣␣//Bying␣is␣closed␣or␣it␣is␣too␣late
␣␣␣␣␣␣␣␣RETURN␣FALSE;
␣␣␣␣␣␣␣END␣IF;
␣␣␣␣END;
␣␣␣␣$BODY$␣LANGUAGE␣plpgsql;
```

5. **BuyReserved(FlightId, SeatNo) — пытается выкупить забронированное место. Возвращает истину, если удалось и ложь — в противном случае.**

```
CREATE FUNCTION BuyReserved(FlightId INT, SeatNo INT)
RETURNS BOOLEAN AS $BODY$
  DECLARE PlaneId INT;
  DECLARE FlightTime TIMESTAMP;
  DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
  DECLARE TransTime TIMESTAMP DEFAULT NULL;
  DECLARE TransType INT DEFAULT NULL;
BEGIN
  SELECT (F.FlightTime, F.isClosedByRequest) INTO
    FlightTime, IsClosedByRequest
  FROM
    Flights as F
  WHERE
    F.FlightId = FlightId;

  IF NOT IsClosedByRequest AND NOW() <= FlightTime - INTERVAL '2 hours' THEN
    SELECT (T.PlaneId, T.TransTime, T.TransType) INTO
      PlaneId, TransTime, TransType
    FROM
      Transaction as T
    WHERE
      T.FlightId = FlightId AND
      T.SeatNo = SeatNo;
    IF TransType = 0 AND NOT IsReservationExpired(FlightId, SeatNo) THEN
      //Seat is reserved
      //Here we need to check that it was reversed by ourselves
      //But we have no information about reservation in our tables
      //In 8th hometask I used passport series/no to check info, but here
      //we haven't them in parameters, so we'll just leave this step,
      //assuming that reservation was done by ourselves.
      DELETE FROM
        Transaction as T
      WHERE
        T.FlightId = FlightId AND
        T.SeatNo = SeatNo;

      INSERT INTO Transaction
        (FlightId, PlaneId, SeatNo, TransTime, TransType)
      VALUES
        (FlightId, PlaneId, SeatNo, now(), 1);
      RETURN TRUE;
    ELSE
      //Seat is already bought or reserved, can't buy it
          RETURN FALSE;
      END IF;
          //Bying is closed or it's too late
      RETURN FALSE;
  END IF;
END;
```

```
$BODY$ LANGUAGE plpgsql;
```

6. **FlightStatistics() — возвращает статистику по рейсам: возможность бронирования и покупки, число свободных, забронированных и проданных мест.**

```
CREATE FUNCTION MayBuy(FId INT)
RETURNS BOOLEAN AS $BODY$
  DECLARE PId INT;
  DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
  DECLARE FTime TIMESTAMP DEFAULT NULL;
  DECLARE FreeSeatsCount INT DEFAULT NULL;
BEGIN
  SELECT (F.IsClosedByRequest, F.FTime, F.PlaneId)
  INTO IsClosedByRequest, FTime, PId
  FROM Flights F
  WHERE F.FlightId = FId;

  SELECT COUNT(*)
  INTO FreeSeatsCount
  FROM FreeSeats(FId);
  IF FreeSeatsCount == 0 OR
    IsClosedByRequest OR NOW() > FTime - INTERVAL '2␣hours' THEN
    RETURN FALSE;
  ELSE
    RETURN TRUE;
  END IF;
END;
$BODY$ LANGUAGE plpgsql;

CREATE FUNCTION MayReserve(FId INT)
RETURNS BOOLEAN AS $BODY$
  DECLARE PId INT;
  DECLARE IsClosedByRequest BOOLEAN DEFAULT FALSE;
  DECLARE FTime TIMESTAMP DEFAULT NULL;
  DECLARE FreeSeatsCount INT DEFAULT NULL;
BEGIN
  SELECT (F.IsClosedByRequest, F.FTime, F.PlaneId)
  INTO IsClosedByRequest, FTime, PId
  FROM Flights F
  WHERE F.FlightId = FId;

  SELECT COUNT(*)
  INTO FreeSeatsCount
  FROM FreeSeats(FId);
  IF FreeSeatsCount == 0 OR
    IsClosedByRequest OR NOW() > FTime - INTERVAL '24␣hours' THEN
    RETURN FALSE;
  ELSE
    RETURN TRUE;
  END IF;
END;
```

```
$BODY$ LANGUAGE plpgsql;

CREATE FUNCTION FreeSeatsCount(FId INT)
RETURNS INT AS $BODY$
  DECLARE FreeSeatsCount INT DEFAULT NULL;
BEGIN
  SELECT COUNT(*)
  INTO FreeSeatsCount
  FROM FreeSeats(FId);
  RETURN FreeSeatsCount;
END;
$BODY$ LANGUAGE plpgsql;

CREATE FUNCTION ReservedSeatsCount(FId INT)
RETURNS INT AS $BODY$
  DECLARE ReservedSeatsCount INT DEFAULT NULL;
BEGIN
  SELECT COUNT(*)
  INTO ReservedSeatsCount
  FROM (
    SELECT
      T.SeatNo
    FROM
      Transaction as T
    WHERE
      T.FlightId = FId AND
      T.TransType = 0
  ) as SUBQ;
  RETURN ReservedSeatsCount;
END;
$BODY$ LANGUAGE plpgsql;

CREATE FUNCTION BoughtSeatsCount(FId INT)
RETURNS INT AS $BODY$
  DECLARE BoughtSeatsCount INT DEFAULT NULL;
BEGIN
  SELECT COUNT(*)
  INTO BoughtSeatsCount
  FROM (
    SELECT
      T.SeatNo
    FROM
      Transaction as T
    WHERE
      T.FlightId = FId AND
      T.TransType = 1
  ) as SUBQ;
  RETURN BoughtSeatsCount;
END;
$BODY$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION FlightStatistics()
RETURNS TABLE(FlightId INT, MayBuy BOOLEAN, MayReserve BOOLEAN,
              FreeSeats INT, ReservedSeats INT, BoughtSeats INT) AS $BODY$
BEGIN
  RETURN QUERY
  SELECT F.FlightId,
        MayBuy(F.FlightId) as MayBuy,
        MayReserve(F.FlightId) as MayReserve,
        FreeSeatsCount(F.FlightId) as FreeSeats,
        ReservedSeatsCount(F.FlightId) as ReservedSeats,
        BoughtSeatsCount(F.FlightId) as BoughtSeats
  FROM Flights F;
END;
$BODY$ LANGUAGE plpgsql;
```