Google Forms

Благодарим за заполнение формы Введение в СУБД. ДЗ-9!

Полученные ответы

Изменить ответ

Введение в СУБД. ДЗ-9

Хранимые процедуры

Электронная почта *

mirzo.zaynidinov@gmail.com

Студент *

Зайнидинов Мирзофирдавс Шавкатович

0. Подготовка

0.1. Укажите название и версию СУБД, для которой вы писали хранимые процедуры *

PostgreSQL 17.0

0.2. Таблицы и представления *

Укажите определения таблиц (в том числе, указанных в условии) и представлений, созданных в рамках ДЗ. Описание таблиц и представлений укажите в комментариях (--) к ним.

CREATE TABLE Flights (FlightId INT NOT NULL PRIMARY KEY, FlightTime TIMESTAMP NOT NULL, PlaneId INT NOT NULL, -- атрибуты для отключения бронирования и продаж вручную BookingAvailable BOOLEAN DEFAULT TRUE, BuyingAvailable BOOLEAN DEFAULT TRUE); CREATE TABLE Seats (PlaneId INT NOT NULL, SeatNo VARCHAR(4) NOT NULL -- 123A); -- Таблица юзеров CREATE TABLE Users (UserId INT NOT NULL PRIMARY KEY, UserPass TEXT NOT NULL); -- Таблица заказов CREATE TABLE Orders (FlightId INT NOT NULL, SeatNo VARCHAR(4) NOT NULL, -- OrderStatus = TRUE - билет оплачен, иначе он забронирован OrderStatus BOOLEAN NOT NULL, UserId INT, -- Дата окончания срока бронирования билета ExpirationTime TIMESTAMP, CONSTRAINT orders_pk PRIMARY KEY (FlightId, SeatNo), CONSTRAINT orders_flights_fk FOREIGN KEY (FlightId) REFERENCES Flights (FlightId), CONSTRAINT orders_users_fk FOREIGN KEY (UserId) REFERENCES Users (UserId), CHECK (OrderStatus = TRUE AND ExpirationTime IS NULL OR OrderStatus = FALSE AND ExpirationTime IS NOT NULL));

0.3. Дополнительные процедуры

Укажите определения вспомогательных хранимых процедур, созданных в рамках ДЗ. Описание процедур укажите в комментариях к ним.

CREATE EXTENSION pgcrypto; -- Возможность бронирования должна автоматически отключаться за трое суток до начала рейса. Продажа мест должна автоматически отключаться за три часа до начала рейса. Также должна быть предусмотрена возможность отключения бронирования и продаж вручную. CREATE OR REPLACE FUNCTION CanBook(FId INT) RETURNS BOOLEAN AS $$ BEGIN RETURN EXISTS(SELECT 1 FROM Flights WHERE FlightId = FId AND (FlightTime < NOW() + INTERVAL '3 days' OR BookingAvailable = FALSE)); END; $$ LANGUAGE plpgsql; CREATE OR REPLACE FUNCTION CanBuy(FId INT) RETURNS BOOLEAN AS $$ BEGIN RETURN EXISTS(SELECT 1 FROM Flights WHERE FlightId = FId AND (FlightTime < NOW() + INTERVAL '3 hours' OR BuyingAvailable = FALSE)); END; $$ LANGUAGE plpgsql; -- Смотрим можем ли мы авторизоваться CREATE OR REPLACE FUNCTION CantAuth(UId INT, Pass TEXT) RETURNS BOOLEAN AS $$ BEGIN RETURN NOT EXISTS(SELECT UserId FROM Users WHERE UserId = UId AND UserPass = CRYPT(Pass, UserPass)); END; $$ LANGUAGE plpgsql;

1. Администрирование

1.1. RegisterUser *

-- Возвращаем TRUE если не нашли юзера в таблице и добавляем его туда, иначе возвращаем FALSE CREATE OR REPLACE FUNCTION RegisterUser(UId INT, Pass TEXT) RETURNS BOOLEAN AS $$ BEGIN IF NOT EXISTS(SELECT UserId FROM Users WHERE UserId = UId AND UserPass = CRYPT(Pass, UserPass)) THEN INSERT INTO Users (UserId, UserPass) VALUES (UId, CRYPT(Pass, gen_salt('md5'))); RETURN TRUE; ELSE RETURN FALSE; END IF; END; $$ LANGUAGE plpgsql;

1.2. ManageFlight *

-- Если нашли юзера в таблице обновляем его, иначе ничего не делаем. CREATE OR REPLACE FUNCTION ManageFlight(UId INT, Pass TEXT, FId INT, SellAllowed BOOLEAN, ReservationAllowed BOOLEAN) RETURNS VOID AS $$ BEGIN IF EXISTS(SELECT UserId FROM Users WHERE UserId = UId AND UserPass = CRYPT(Pass, UserPass)) THEN -- Обновляем настройки рейса, если пользователь существует UPDATE Flights SET BuyingAvailable = SellAllowed, BookingAvailable = ReservationAllowed WHERE FlightId = FId; END IF; RETURN; END; $$ LANGUAGE plpgsql;

2. Покупка и бронирование

2.1. FreeSeats *

-- Смотрим не отключены ли бронирования и покупка билетов, по таким билетам берем те которые не оплачены и срок бронирования у них истек или же соответствующего билета вообще нет. В других случаях ничего не возвращаем. CREATE OR REPLACE FUNCTION FreeSeats(FId INT) RETURNS TABLE (SeatNo VARCHAR(4)) AS $$ BEGIN IF NOT (CanBook(FId) OR CanBuy(FId)) THEN RETURN QUERY (SELECT o.SeatNo FROM Flights f JOIN Orders o ON f.FlightId = o.FlightId WHERE f.FlightId = FId AND o.OrderStatus = FALSE AND o.ExpirationTime <= NOW() UNION SELECT s.SeatNo FROM Seats s WHERE s.PlaneId = (SELECT PlaneId FROM Flights WHERE FlightId = FId) AND s.SeatNo NOT IN (SELECT o.SeatNo FROM Orders o WHERE o.FlightId = FId)); END IF; RETURN; -- Возврат пустого результата, если условия выполняются END; $$ LANGUAGE plpgsql;

2.2. Reserve *

-- Если место FreeSeat и пароль корректный добавляем бронирование и возвращаем TRUE, иначе FALSE CREATE OR REPLACE FUNCTION Reserve(UId INT, Pass TEXT, FId INT, Seat VARCHAR(4)) RETURNS BOOLEAN AS $$ BEGIN IF CantAuth(UId, Pass) OR NOT EXISTS(SELECT * FROM FreeSeats(FId) WHERE SeatNo = Seat) THEN RETURN FALSE; ELSE INSERT INTO

Orders(FlightId, UserId, SeatNo, OrderStatus, ExpirationTime) VALUES (FId, UId, Seat, FALSE, NOW() + INTERVAL '3 days') ON CONFLICT (FlightId, SeatNo) DO UPDATE SET UserId = UId, ExpirationTime = now() + INTERVAL '3 days'; RETURN TRUE; END IF; END; $$ LANGUAGE plpgsql;

2.3. ExtendReservation *

-- Возвращаем TRUE, если бронь была сделала тем же юзером можно авторизоваться и бронирование не закрыто, иначе FALSE CREATE OR REPLACE FUNCTION ExtendReservation(UId INT, Pass TEXT, FId INT, Seat VARCHAR(4)) RETURNS BOOLEAN AS $$ BEGIN IF NOT (NOT EXISTS (SELECT 1 FROM Tickets WHERE FlightId = FId AND UserId = UId AND SeatNo = Seat AND OrderStatus = FALSE AND ExpirationTime < (SELECT FlightTime FROM Flights WHERE FlightId = FId)) OR CantAuth(UId, Pass) OR CanBook(FId)) THEN UPDATE Orders SET ExpirationTime = ExpirationTime + INTERVAL '3 days' WHERE FlightId = FId AND SeatNo = Seat; RETURN TRUE; ELSE RETURN FALSE; END IF; END; $$ LANGUAGE plpgsql;

2.4. BuyFree *

-- Возвращаем FALSE если не можем купить билет или записи нету в свободных местах, иначе TRUE CREATE OR REPLACE FUNCTION BuyFree(FId INT, Seat VARCHAR(4)) RETURNS BOOLEAN AS $$ BEGIN IF NOT CanBuy(FId) OR NOT EXISTS(SELECT * FROM FreeSeats(FId) Where SeatNo = Seat) THEN RETURN FALSE; ELSE INSERT INTO Orders(FlightId, SeatNo, OrderStatus, UserId, ExpirationTime) VALUES (FId, Seat, TRUE, NULL, NULL); RETURN TRUE; END IF; END; $$ LANGUAGE plpgsql;

2.5. BuyReserved *

-- Возвращаем FALSE если не можем купить билет или не можем авторизоваться или билета нет, иначе TRUE CREATE OR REPLACE FUNCTION BuyReserved(UId int, Pass text, FId int, Seat varchar(4)) RETURNS BOOLEAN AS $$ BEGIN IF CantAuth(UId, Pass) OR CanBuy(FId) OR not exists(select * from Orders WHERE FlightId = FId AND UserId = UId AND SeatNo = Seat AND OrderStatus = false) THEN RETURN FALSE; ELSE UPDATE Orders SET ExpirationDate = NULL, OrderStatus = true where FlightId = FId AND SeatNo = Seat; RETURN TRUE; END IF; END; $$ LANGUAGE plpgsql;

3. Статистика

3.1. FlightsStatistics *

```
CREATE OR REPLACE FUNCTION FlightsStatistics(UId INT, Pass TEXT) RETURNS TABLE( FlightId INTEGER, CanBook BOOLEAN, CanBuy BOOLEAN, FreeSeats BIGINT, ReservedSeats BIGINT, SoldSeats BIGINT ) AS $$ BEGIN IF CantAuth(UId, Pass) THEN RETURN; END IF; RETURN QUERY SELECT f.FlightId AS FlightId, not CanBook(f.FlightId) AND (COUNT(SeatNo) FILTER (WHERE OrderStatus IS NULL OR OrderStatus = FALSE AND ExpirationTime <= NOW())) > 0 AS CanBook, CanBuy(f.FlightId) AND (COUNT(SeatNo) FILTER (WHERE UserId = UId AND OrderStatus = FALSE AND ExpirationTime > NOW())) > 0 AS CanBuy, COUNT(SeatNo) FILTER (WHERE OrderStatus IS NULL OR OrderStatus = FALSE AND ExpirationTime <= NOW()) AS FreeSeats, COUNT(SeatNo) FILTER (WHERE UserId = UId AND OrderStatus = FALSE AND ExpirationTime > NOW()) AS ReservedSeats, COUNT(SeatNo) FILTER (WHERE UserId = UId AND OrderStatus = TRUE) AS SoldSeats FROM Flights f NATURAL JOIN Seats NATURAL LEFT JOIN Orders GROUP BY f.FlightId; END; $$ LANGUAGE plpgsql;
```

3.2. FlightStat *

```
CREATE OR REPLACE FUNCTION FlightStat(UId INT, Pass TEXT, FId INT) RETURNS TABLE( FlightId INTEGER, CanBook BOOLEAN, CanBuy BOOLEAN, FreeSeats BIGINT, ReservedSeats BIGINT, SoldSeats BIGINT ) AS $$ BEGIN RETURN QUERY SELECT * FROM FlightsStatistics(UId, Pass) s WHERE s.FlightId = FId; END; $$ LANGUAGE plpgsql;
```

4. CompressSeats *

```
CREATE OR REPLACE PROCEDURE CompressSeats(FId INT) AS $$ DECLARE CurSNo VARCHAR(4); Orders RECORD; BEGIN CREATE TEMP TABLE TempOrders ( FlightId INT, SeatNo VARCHAR(4), OrderStatus BOOLEAN, UserId INT, ExpirationTime TIMESTAMP ); FOR CurSNo IN SELECT DISTINCT SeatNo FROM Seats WHERE PlaneId = (SELECT PlaneId FROM Flights WHERE FlightId = FId) ORDER BY SeatNo LOOP SELECT * INTO Orders FROM Orders t WHERE FlightId = FId AND (OrderStatus = TRUE OR (OrderStatus = FALSE AND ExpirationTime > NOW())) ORDER BY NOT OrderStatus, SeatNo LIMIT 1; IF FOUND THEN INSERT INTO TempOrders (FlightId, SeatNo, OrderStatus, UserId, ExpirationTime) VALUES (Orders.FlightId, CurSNo, Orders.OrderStatus, Orders.UserId, Orders.ExpirationTime); DELETE FROM Orders WHERE ctid = (SELECT ctid FROM Orders t2 WHERE t2.FlightId = FId AND t2.UserId = Orders.UserId AND t2.SeatNo = Orders.SeatNo LIMIT 1); END IF; END LOOP; INSERT INTO Orders (FlightId, SeatNo, OrderStatus, UserId, ExpirationTime) SELECT FlightId, SeatNo, OrderStatus, UserId, ExpirationTime FROM TempOrders; END; $$ LANGUAGE plpgsql;
```