De procedure waar er een potentieel concurrency probleem kan voorkomen is PROC_INSERT_NEW_PLAYER

```
CREATE OR ALTER PROCEDURE PROC_INSERT_NEW_PLAYER @Country_name VARCHAR(128),
                                                 @First_name VARCHAR(128),
                                                 @Last_name VARCHAR(128),
                                                 @Middle_name VARCHAR(128) = NULL,
                                                 @Birth_date DATE,
                                                 @Club_name VARCHAR(128),
                                                 @Jersey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @Inserted_personID TABLE
                               (
                                   person_id INT NOT NULL
                               );

    INSERT INTO PERSON (COUNTRY_NAME, FIRST_NAME, LAST_NAME, MIDDLE_NAME, BIRTH_DATE)
    OUTPUT (inserted.PERSON_ID) INTO @Inserted_personID
    VALUES (@Country_name, @First_name, @Last_name, @Middle_name, @Birth_date)

    INSERT INTO PLAYER (PERSON_ID, CLUB_NAME, JERSEY)
    SELECT person_id, @Club_name, @Jersey
    FROM @Inserted_personID;
END
```

Op dit moment kan er een probleem ontstaan in deze procedure.

Waarneer deze procedure wordt gestart zal na de eerste insert een person_id terug gegeven worden.

Deze person_id wordt in de volgende insert gebruikt om een player te koppelen, maar als tussendoor de person_id wordt veranderd dan komt er een error.

Dit probleem kan je oplossen door de procedure het isolatie level `REPEATABLE READ` te geven, nu kan de geïnserte waarde niet worden aangepast totdat de procedure is afgelopen.

Je had ook kunnen gaan voor de zwaardere isolatie level `SERIALIZABLE`. Maar dat is niet nodig, omdat andere inserts geen problemen opleveren en `SERIALIZABLE` zorgt voor een lagere performance dan `REPEATABLE READ`.