Филиппов Дмитрий, М3439

Домашнее задание 7.

Схема БД: Course(CId, GId, CName), Lecturer(LId, LName), Mark(SId, CId, GId, Mark), Groups(GId, GName), Student(SId, GId, SName), Plan(LId, CId, GId).

Запросы:

1. **Напишите запрос, удаляющий всех студентов, не имеющих долгов.**

```
DELETE FROM
    Student S
WHERE
    NOT EXISTS
    (SELECT * FROM
        Mark M
     WHERE
        M.SId = S.SId AND M.Mark < 60);
```

2. **Напишите запрос, удаляющий всех студентов, имеющих 3 и более долгов.**

```
DELETE FROM
    Student S
WHERE
    (SELECT COUNT(*) FROM
        (SELECT * FROM
            Mark M
         WHERE
            M.SId = S.SId AND M.Mark < 60
        ) as debts) >= 3;
```

3. **Напишите запрос, удаляющий все группы, в которых нет студентов.**

```
DELETE FROM
    Groups G
WHERE
    NOT EXISTS
    (SELECT * FROM
        Student S
     WHERE S.GId = G.GId);
```

4. **Создайте view Losers в котором для каждого студента, имеющего долги указано их количество.**

```
CREATE VIEW
    Losers
AS
    SELECT
        S.SId, S.SName,
        (SELECT COUNT(*) FROM
            (SELECT * FROM
                Mark M
             WHERE
                M.SId = S.SId AND M.Mark < 60
```

```
            ) as debts_info) as debts_count
    FROM
        Student S;
```

5. **Создайте таблицу LoserT, в которой содержится та же информация, что во view Losers. Эта таблица должна автоматически обновляться при изменении таблицы с баллами.**

```
SELECT * INTO LoserT FROM Losers;

CREATE FUNCTION updateLoserT()
    RETURNS TRIGGER AS $BODY$
BEGIN
    TRUNCATE LoserT;
    INSERT INTO LoserT (SELECT * FROM Losers);
    RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER UpdateLoserT
AFTER INSERT OR UPDATE ON Mark
FOR EACH ROW EXECUTE PROCEDURE updateLoserT();
```

6. **Отключите автоматическое обновление таблицы LoserT.**

```
DROP TRIGGER UpdateLoserT ON Mark;
```

или же

```
ALTER TABLE Mark DISABLE TRIGGER UpdateLoserT;
```

(с возможностью последующего включения с помощью `ALTER TABLE Mark ENABLE TRIGGER UpdateLoserT;`)

7. **Напишите запрос (один), которой обновляет таблицу LoserT, используя данные из таблицы NewPoints, в которой содержится информация о баллах, проставленных за последний день.**

Будем считать, что таблица NewPoints содержит поля `SId INT`, `SName VARCHAR(45)`, `CId INT`, `Mark INT`.

```
MERGE INTO LoserT LT
USING (SELECT * FROM NewPoints NP, Mark M) NPM
ON LT.SId = NPM.SId AND M.CId = NPM.CId
WHEN MATCHED THEN
    UPDATE SET LT.debts_count =
        CASE
            WHEN M.Mark < 60 AND NPM.Mark >= 60 THEN LT.debts_count - 1
            WHEN M.Mark >= 60 AND NPM.Mark < 60 THEN LT.debts_count + 1
            ELSE LT.debts_count
        END
WHEN NOT MATCHED THEN
    IF NPM.Mark < 60 THEN
        INSERT VALUES (NPM.SId, NPM.SName, 1);
    ELSE
        INSERT VALUES (NPM.SID, NPM.SName, 0);
    END IF;
```

8. **Добавьте проверку того, что все студенты одной группы изучают один и тот же набор курсов.**

В нашей схеме базы данных эта проверка не нужна, потому что нельзя добавить данные, не удовлетворяющие этому условию, в базу: таблица $Plan$ хранит LId, CId, GId, но не SId, поэтому все студенты группы GId (а каждый студент привязан к своей группе) обучаются по этому плану, и все предметы у них одинаковые. При добавлении же в $Mark$ добавить данные, не удовлетворяющие условию, тоже не получится, потому что там (CId, GId) — primary key, а (SId, GId) — foreign key.

9. **Создайте триггер, не позволяющий уменьшить баллы студента по предмету. При попытке такого изменения, баллы изменяться не должны.**

```
CREATE FUNCTION DontAllowToDecreaseMarks()
    RETURNS TRIGGER AS $BODY$
BEGIN
    IF NEW.Mark < OLD.Mark THEN
        RETURN OLD;
    END IF;
    RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER DontAllowToDecreaseMarks
BEFORE UPDATE ON Mark
FOR EACH ROW EXECUTE PROCEDURE DontAllowToDecreaseMarks();
```