Домашнее задание №7

Андрей Козлов, гр. 4538

Пусть используются таблицы со следующими именами:

- students
- groups
- lecturers
- courses
- marks
- grouplists
- schedule

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

```
DELETE FROM grouplists WHERE student_id NOT IN
        (SELECT student_id FROM marks WHERE points < 60);
```

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

```
DELETE FROM grouplists WHERE student_id IN
        (SELECT student_id FROM marks WHERE points < 60
        GROUP BY student_id HAVING count(*) >= 3);
```

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

```
DELETE FROM groups WHERE group_id NOT IN
        (SELECT group_id FROM grouplists);
```

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

```
CREATE VIEW Losers AS SELECT student_name, count(*) FROM
        marks NATURAL JOIN students NATURAL JOIN grouplists
                WHERE points < 60 GROUP BY student_name;
```

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

```
CREATE MATERIALIZED VIEW LosersT AS
        (SELECT student_name, count(*) FROM
        marks NATURAL JOIN students NATURAL JOIN grouplists
                WHERE points < 60 GROUP BY student_name);

CREATE FUNCTION refresh_loserst() RETURNS trigger AS $refresh_loserst$
BEGIN
        REFRESH MATERIALIZED VIEW LosersT;
        RETURN NEW;
END;
$refresh_loserst$ LANGUAGE plpgsql;

CREATE TRIGGER refresh_loserst AFTER UPDATE ON marks
        EXECUTE PROCEDURE refresh_loserst();
```

6. Отключите автоматическое обновление таблицы LoserT.

```
DROP FUNCTION IF EXISTS refresh_loserst() CASCADE;
```

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

```
UPDATE marks SET points = NewPoints.points
        WHERE marks.student_id = NewPoints.student_id
        AND marks.course_id = NewPoints.course_id;
```

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

```
CREATE FUNCTION check_course() RETURNS trigger AS $check_course$
BEGIN
        IF EXISTS (SELECT group_id FROM grouplists
                WHERE student_id = NEW.student_id
                AND group_id NOT IN
                        (SELECT group_id FROM schedule
                                WHERE course_id = NEW.course_id)) THEN
                RAISE EXCEPTION 'student % cannot have
                a mark for % course', NEW.student_id, NEW.course_id;
        END IF;
        RETURN NEW;
END;
$check_course$ LANGUAGE plpgsql;

CREATE TRIGGER check_courses BEFORE INSERT OR UPDATE ON marks
        FOR EACH ROW EXECUTE PROCEDURE check_course();
```

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

```
CREATE FUNCTION not_decrease_mark() RETURNS trigger AS $decrease_mark$
BEGIN
        IF (NEW.points < OLD.points) THEN
                RAISE EXCEPTION 'student % already has % points',
                NEW.student_id, OLD.points;
        END IF;
        RETURN NEW;
END;
$decrease_mark$ LANGUAGE plpgsql;

CREATE TRIGGER not_decrease_marks BEFORE UPDATE ON marks
        FOR EACH ROW EXECUTE PROCEDURE not_decrease_mark();
```