Домашнее задание 7

Дарья Яковлева, M3439

22.11.2016

Задание: Составить запросы.

P.S. Будем считать, что у студента долг по предмету, если он изучает этот предмет и имеет по нему менее 60 баллов.

1. **Запрос, удаляющий всех студентов, не имеющих долгов**

```
DELETE
FROM students
WHERE student_id NOT IN
(
        SELECT marks.student_id
        FROM marks
        WHERE marks.mark_value <= 60 and marks.course_id IN
        (
                SELECT academicplan.course_id
                FROM academicplan
                WHERE students.group_id = academicplan.group_id
    )
);
```

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

```
DELETE
FROM students
WHERE student_id IN
(
        SELECT marks.student_id
        FROM marks
        WHERE marks.mark_value <= 60 AND marks.course_id IN
        (
                SELECT academicplan.course_id
                FROM academicplan
                WHERE students.group_id = academicplan.group_id
    )
    GROUP BY marks.student_id
    HAVING COUNT(marks.student_id) >= 3
);
```

3. **Запрос, удаляющий все группы, в которых нет студентов**

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

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

```
CREATE VIEW Losers AS
SELECT students.student_name, COUNT(marks.student_id)
FROM students
NATURAL JOIN marks
WHERE marks.mark_value <= 60 AND marks.course_id IN
(
        SELECT academicplan.course_id
        FROM academicplan
        WHERE students.group_id = academicplan.group_id
)
GROUP BY students.student_id;
```

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

```
CREATE TABLE LoserT AS SELECT * FROM Losers;

CREATE OR REPLACE FUNCTION loser_procedure()
RETURNS TRIGGER AS $LoserTrig$
    BEGIN
                DELETE FROM LoserT *;
                INSERT INTO LoserT (SELECT * FROM Losers);
                RETURN NEW;
    END;
$LoserTrig$ LANGUAGE plpgsql;


CREATE TRIGGER LoserTrig
AFTER INSERT OR UPDATE OR DELETE ON marks
FOR EACH ROW EXECUTE PROCEDURE loser_procedure();
```

6. **Отключить автоматическое обновление таблицы LoserT**

```
DROP TRIGGER IF EXISTS LoserTrig ON marks;
```

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

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

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

```
CREATE OR REPLACE FUNCTION marks_procedure()
 RETURNS TRIGGER AS $LoserTrig$
    BEGIN
        IF (NEW.mark_value < OLD.mark_value) THEN
                NEW.mark_value = OLD.mark_value;
                RETURN NEW;
        END IF;
        RETURN NULL;
    END;
$LoserTrig$ LANGUAGE plpgsql;


CREATE TRIGGER marksUpdate
BEFORE UPDATE ON marks
FOR EACH ROW EXECUTE PROCEDURE marks_procedure();
```

Приложение. База данных.

```
CREATE TABLE IF NOT EXISTS groups (
        group_id int PRIMARY KEY,
        group_name varchar(100)
);

CREATE TABLE IF NOT EXISTS students (
        student_id int PRIMARY KEY,
        student_name varchar(100),
        group_id int REFERENCES groups (group_id)
                ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS lecturers (
        lecturer_id int PRIMARY KEY,
        lecturer_name varchar(50)
);
```

```
CREATE TABLE IF NOT EXISTS courses (
        course_id int PRIMARY KEY,
        course_name varchar(50)
);

CREATE TABLE IF NOT EXISTS marks (
        mark_value int,
        course_id int not null REFERENCES courses (course_id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        student_id int not null REFERENCES students (student_id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        CHECK (mark_value BETWEEN 0 AND 100),
        PRIMARY KEY (course_id, student_id)
);

CREATE TABLE IF NOT EXISTS academicplan (
        lecturer_id int not null REFERENCES lecturers (lecturer_id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        course_id int not null REFERENCES courses (course_id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        group_id int not null REFERENCES groups (group_id)
                 ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY (lecturer_id, course_id, group_id)
);
```