Волков Александр, М3437

Домашнее задание

Структура базы данных «Университет»:

- *Students(StudentId, StudentName, GroupId)*
- *Groups(GroupId, GroupName)*
- *Courses(CourseId, CourseName)*
- *Lecturers(LecturerId, LecturerName)*
- *Plan(GroupId, CourseId, LecturerId)*
- *Marks(StudentId, CourseId, Mark)*

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

```
DELETE
FROM Students
WHERE StudentId NOT IN
(
    SELECT StudentId
    FROM (Students NATURAL JOIN Plan) as P LEFT JOIN Marks
    ON P.StudentId = Marks.StudentId AND P.CourseId = Marks.CourseId
    WHERE Mark IS NULL OR Mark < 60

)
```

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

```
DELETE
FROM Students
WHERE StudentId IN
(
    SELECT StudentId
    FROM (Students NATURAL JOIN Plan) as P LEFT JOIN Marks
    ON P.StudentId = Marks.StudentId AND P.CourseId = Marks.CourseId
    WHERE Mark IS NULL OR Mark < 60
    GROUP BY StudentId HAVING COUNT (StudentId) >= 3
)
```

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

```
DELETE
FROM Groups
WHERE GroupId NOT IN
```

```
(
        SELECT Students.GroupId FROM Students
)
```

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

```
SELECT VIEW Losers AS
SELECT StudentId, COUNT(Mark)
FROM (Students NATURAL JOIN Plan) as P LEFT JOIN Marks
        ON P.StudentId = Marks.StudentId AND P.CourseId = Marks.CourseId
        WHERE Mark IS NULL OR Mark < 60
GROUP BY StudentId
```

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 TABLE LoserT(
        StudentId int,
        Debt int,
        PRIMARY KEY StudentId,
        FOREIGN KEY StudentId references Students(StudentId)
)
INSERT INTO LoserT
SELECT StudentId, COUNT(Mark) as Debt
FROM (Students NATURAL JOIN Plan) as P LEFT JOIN Marks
        ON P.StudentId = Marks.StudentId AND P.CourseId = Marks.CourseId
        WHERE Mark IS NULL OR Mark < 60
GROUP BY StudentId

CREATE TRIGGER UpdateLosser ON Marks AFTER UPDATE
REFERENCING OLD TABLE O, REFERENCING NEW TABLE N
MERGE LoserT USING
 (
        SELECT StudentId,
                sum(O.Mark IS NOT NULL AND O.Mark >= 60) AS OldPassed,
                sum(N.Mark IS NOT NULL AND N.Mark>= 60) AS NewPassed
          from O right join N
        on O.StudentId = N.StudentId and O.CourseId =     N.CourseId
) AS L
ON L.StudentId = LoserT.StudentId
WHEN MATCHED THEN UPDATE LoserT SET Debt= Debt + L.OldPassed — L.NewPassed
```

```
WHEN NOT MATCHED THEN INSERT (StudentId, Debt)
VALUES (L.StudentId, L.OldPassed — L.NewPassed)
WHERE L.OldPassed — L.NewPassed > 0;

CREATE TRIGGER InsertLosser ON Marks AFTER INSERT
REFERENCING NEW ROW N
UPDATE LoserT SET Debt = Debt - 1
WHERE StudentId= N.StudentId AND N.Mark >= 60;

CREATE TRIGGER RemoveSuccessfulStudents ON LosertT AFTER UPDATE, INSERT
DELETE FROM LoserT WHERE Debt = 0;
```

6. Отключите автоматическое обновление таблицы LoserT.

```
drop trigger if exists UpdateLosser on Marks
drop trigger if exists InsertLosser on Marks
drop trigger if exists RemoveSuccessfulStudents on  LosertT
```

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

```
MERGE LoserT USING
 (
        SELECT StudentId,
                sum(Mark.Mark IS NOT NULL AND NewPoints.Mark >= 60) AS OldPassed,
                sum(Mark.Mark IS NOT NULL AND NewPoints.Mark>= 60) AS NewPassed
          from Mark right join NewPoints
        on Mark.StudentId = NewPoints.StudentId and Mark.CourseId =  NewPoints.CourseId
) AS L
ON L.StudentId = LoserT.StudentId
WHEN MATCHED THEN UPDATE LoserT SET Debt= Debt + L.OldPassed — L.NewPassed
WHEN NOT MATCHED THEN INSERT (StudentId, Debt)
VALUES (L.StudentId, L.OldPassed — L.NewPassed)
WHERE L.OldPassed — L.NewPassed > 0;
```

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

```
CREATE TRIGGER OnlyIncrease on Marks BEFORE UPDATE
REFERENCING OLD ROW P, REFERENCING NEW ROW S
FOR EACH ROW SET S.Mark = max(P.Mark, S.Mark);
```