Домашнее задание № 7

Кокорин Илья, M3439

28 октября 2019 г.

1 Создание и заполнение таблиц

```
\c CTD;

CREATE TABLE Groups
(
  GroupId   DECIMAL(5) NOT NULL PRIMARY KEY,
  GroupName CHAR(5)    NOT NULL UNIQUE
);

CREATE TABLE Students
(
  StudentId   DECIMAL(6)  NOT NULL PRIMARY KEY,
  StudentName VARCHAR(50) NOT NULL,
  GroupId     DECIMAL(5)  NOT NULL,
  FOREIGN KEY (GroupId) REFERENCES Groups (GroupId)
);

CREATE TABLE Courses
(
  CourseId   DECIMAL(5)  NOT NULL PRIMARY KEY,
  CourseName VARCHAR(50) NOT NULL
);

CREATE TABLE Lecturers
(
  LecturerId   DECIMAL(6)  NOT NULL PRIMARY KEY,
  LecturerName VARCHAR(50) NOT NULL
);

CREATE TABLE Marks
(
  StudentId DECIMAL(6) NOT NULL,
  CourseId  DECIMAL(5) NOT NULL,
  Mark      DECIMAL(3) NOT NULL,
  PRIMARY KEY (StudentId, CourseId),
  FOREIGN KEY (CourseId) REFERENCES Courses (CourseId),
  FOREIGN KEY (StudentId) REFERENCES Students (StudentId)
    ON DELETE CASCADE ON UPDATE NO ACTION
);

CREATE TABLE Plan
(
  CourseId   DECIMAL(5) NOT NULL,
  GroupId    DECIMAL(5) NOT NULL,
```

```
  LecturerId DECIMAL(6) NOT NULL,
  PRIMARY KEY (CourseId, GroupId),
  FOREIGN KEY (CourseId) REFERENCES Courses (CourseId),
  FOREIGN KEY (GroupId) REFERENCES Groups (GroupId),
  FOREIGN KEY (LecturerId) REFERENCES Lecturers (LecturerId)
);

INSERT INTO Groups(GroupId, GroupName)
VALUES (1, 'M3435'),
       (2, 'M3437'),
       (3, 'M3439'),
       (4, 'M3438');

INSERT INTO Students (StudentId, StudentName, GroupId)
VALUES (1, 'Илья Кокорин', 3),
       (2, 'Лев Довжик', 3),
       (3, 'Артём Абрамов', 3),
       (4, 'Николай Рыкунов', 1),
       (5, 'Ярослав Балашов', 1),
       (6, 'Никита Дугинец', 2);

INSERT INTO Courses(CourseId, CourseName)
VALUES (1, 'Математический анализ'),
       (2, 'Технологии Java'),
       (3, 'Базы данных'),
       (4, 'Теория вероятностей'),
       (5, 'Матстат');

INSERT INTO Lecturers (LecturerId, LecturerName)
VALUES (1, 'Георгий Корнеев'),
       (2, 'Константин Кохась'),
       (3, 'Ольга Семёнова'),
       (4, 'Ирина Суслина');

INSERT INTO Plan (CourseId, GroupId, LecturerId)
VALUES (1, 1, 3),
       (1, 2, 3),
       (1, 3, 2),
       (2, 2, 1),
       (2, 3, 1),
       (3, 3, 1),
       (4, 3, 4),
       (5, 3, 4);


INSERT INTO Marks (StudentId, CourseId, Mark)
VALUES (1, 1, 100),
       (1, 2, 85),
       (1, 3, 75),
       (2, 1, 75),
       (2, 2, 85),
       (2, 3, 100),
       (4, 1, 100),
       (5, 1, 75),
       (6, 1, 100),
       (6, 2, 85),
```

```
       (1, 4, 100),
       (2, 5, 85),
       (1, 5, 50);
```

2 Напишите запрос, удаляющий всех студентов, не имеющих долгов

```
WITH StudentsWithStudiedCoursesCount AS (
  SELECT Students.StudentId,
         count(Plan.CourseId) AS StudiedCoursesCount
  FROM Students
         LEFT OUTER JOIN Plan ON
    Students.GroupId = Plan.GroupId
  GROUP BY (Students.StudentId)
),
     OnlyPassedCoursesMarks AS (
       SELECT Marks.StudentId,
              Marks.CourseId,
              Marks.Mark
       FROM Marks
       WHERE Marks.Mark >= 60
     ),
     StudentsWithPassedCoursesCount AS (
       SELECT Students.StudentId,
              count(OnlyPassedCoursesMarks.Mark) AS PassedCoursesCount
       FROM Students
              LEFT OUTER JOIN OnlyPassedCoursesMarks ON
         Students.StudentId = OnlyPassedCoursesMarks.StudentId
       GROUP BY (Students.StudentId)
     ),
     StudentsWitouthDepts AS (
       SELECT StudentId
       FROM StudentsWithStudiedCoursesCount
              NATURAL JOIN StudentsWithPassedCoursesCount
       WHERE PassedCoursesCount = StudiedCoursesCount
     )
DELETE
FROM Students
WHERE Students.StudentId IN (
  SELECT StudentsWitouthDepts.StudentId
  FROM StudentsWitouthDepts
);
```

2.1 Напишите запрос, удаляющий всех студентов, имеющих 3 и более долгов

```
WITH StudentsWithStudiedCoursesCount AS (
  SELECT Students.StudentId,
         count(Plan.CourseId) AS StudiedCoursesCount
  FROM Students
         LEFT OUTER JOIN Plan ON
    Students.GroupId = Plan.GroupId
  GROUP BY (Students.StudentId)
),
     OnlyPassedCoursesMarks AS (
       SELECT Marks.StudentId,
```

```
            Marks.CourseId,
            Marks.Mark
      FROM Marks
      WHERE Marks.Mark >= 60
    ),
    StudentsWithPassedCoursesCount AS (
      SELECT Students.StudentId,
            count(OnlyPassedCoursesMarks.Mark) AS PassedCoursesCount
      FROM Students
            LEFT OUTER JOIN OnlyPassedCoursesMarks ON
        Students.StudentId = OnlyPassedCoursesMarks.StudentId
      GROUP BY (Students.StudentId)
    ),
    Losers AS (
      SELECT StudentId
      FROM StudentsWithStudiedCoursesCount
            NATURAL JOIN StudentsWithPassedCoursesCount
      WHERE StudiedCoursesCount - PassedCoursesCount >= 3
    )
DELETE
FROM Students
WHERE Students.StudentId IN (
  SELECT Losers.StudentId
  FROM Losers
);
```

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

```
WITH GroupsWithStudents AS (
  SELECT DISTINCT Students.GroupId
  FROM Students
)
DELETE
FROM Groups
WHERE Groups.GroupId NOT IN (
  SELECT GroupsWithStudents.GroupId
  FROM GroupsWithStudents
);
```

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

```
CREATE VIEW Losers AS (
  WITH StudentsWithStudiedCoursesCount AS (
    SELECT Students.StudentId,
           Students.StudentName,
           count(Plan.CourseId) AS StudiedCoursesCount
    FROM Students
           LEFT OUTER JOIN Plan USING (GroupId)
    GROUP BY (Students.StudentId)
  ),
       OnlyPassedCoursesMarks AS (
```

```
        SELECT Marks.StudentId,
               Marks.CourseId,
               Marks.Mark
        FROM Marks
        WHERE Marks.Mark >= 60
      ),
      StudentsWithPassedCoursesCount AS (
        SELECT Students.StudentId,
               Students.StudentName,
               Students.GroupId,
               count(OnlyPassedCoursesMarks.Mark) AS PassedCoursesCount
        FROM Students
               LEFT OUTER JOIN OnlyPassedCoursesMarks USING (StudentId)
        GROUP BY (Students.StudentId)
      )
  SELECT StudentsWithStudiedCoursesCount.StudentId,
         StudentsWithStudiedCoursesCount.StudentName,
         StudiedCoursesCount - PassedCoursesCount AS DeptsCount
  FROM StudentsWithStudiedCoursesCount
         INNER JOIN StudentsWithPassedCoursesCount USING (StudentId)
  WHERE StudiedCoursesCount > PassedCoursesCount
);
```

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 LoserT AS (
  SELECT Losers.StudentId,
         Losers.StudentName,
         Losers.DeptsCount
  FROM Losers
);

CREATE OR REPLACE FUNCTION refreshLosers() RETURNS TRIGGER AS
$refreshLosers$
BEGIN
  REFRESH MATERIALIZED VIEW LoserT;
END;
$refreshLosers$ LANGUAGE plpgsql;

CREATE TRIGGER UpdateLoserTWhenMarksChanged
  AFTER INSERT OR UPDATE OR DELETE
  ON Marks
  FOR EACH STATEMENT
EXECUTE PROCEDURE refreshLosers();
```

6 Отключите автоматическое обновление таблицы LoserT

```
DROP TRIGGER IF EXISTS UpdateLoserTWhenMarksChanged ON Marks;
```

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

Набор предметов, изучаемых студентом, определяется его группой (в таблице Plan).
Поэтому, такая проверка выполняется автоматически

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

```
CREATE OR REPLACE FUNCTION checkMarksNotDecreased() RETURNS TRIGGER AS
$checkMarksNotDecreased$
BEGIN
  IF NEW.Mark < OLD.Mark THEN
    RETURN OLD;
  ELSE
    RETURN NEW;
  END IF;
END;
$checkMarksNotDecreased$ LANGUAGE plpgsql;

CREATE TRIGGER NotAllowDecreaseMarks
  BEFORE UPDATE
  ON Marks
  FOR EACH ROW
EXECUTE PROCEDURE checkMarksNotDecreased();
```