Домашнее задание № 6

Кокорин Илья, M3439

21 октября 2019 г.

1 Информацию о студентах, с заданной оценкой по предмету «Базы данных»

Считаем заданной оценкой диапазон оценок.

1.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$M :: Marks$;
$DatabaseCourse :: Courses$ **WHERE** $CourseName =$ 'Базы данных';
SELECT $S.StudentId,\ S.StudentsName,\ G.GroupName$
FROM $S,\ G$ **WHERE** $S.GroupId = G.GroupId \wedge \exists\ DatabaseCourse\ (\exists\ M\ (M.CourseId = DatabaseCourse.CourseId \wedge M.Mark \leq ?\ \wedge\ M.Mark \geq ?\ \wedge M.StudentId = S.StudentId))$

1.2 Datalog

$DatabaseCourses(CourseId) : -Courses(CourseId, CourseName), CourseName =$ 'Базы данных'

$R(StudentId, StudentName, GroupName) : -Students(StudentId, StudentName, GroupId),$
$Groups(GroupId, GroupName), Marks(StudentId, CourseId, Mark),$
$DatabaseCourses(CourseId), Mark <=?, Mark >=?$

1.3 SQL

```
WITH DatabaseCourse AS (
  SELECT DISTINCT *
  FROM Courses
  WHERE Courses.CourseName = 'Базы данных'
),
     StudentsWithCorrectMarks AS (
       SELECT DISTINCT Students.StudentId, Students.StudentName, Students.GroupId
       FROM Students
       WHERE EXISTS(SELECT DISTINCT *
                    FROM DatabaseCourse
                    WHERE EXISTS(SELECT DISTINCT *
                                 FROM Marks
                                 WHERE Marks.Mark <= ?
                                   AND Marks.Mark >= ?
                                   AND Marks.StudentId = Students.StudentId
                                   AND Marks.CourseId = DatabaseCourse.CourseId))
     )
SELECT DISTINCT StudentsWithCorrectMarks.StudentId,
                StudentsWithCorrectMarks.StudentName,
                Groups.GroupName
```

```
FROM StudentsWithCorrectMarks
       NATURAL JOIN Groups;
```

2 Информацию о студентах не имеющих оценки по предмету «Базы данных»

2.1 среди всех студентов

2.1.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$M :: Marks$;
$DatabaseCourse :: Courses$ **WHERE** $CourseName =$ 'Базы данных';
SELECT $S.StudentId,\ S.StudentsName,\ G.GroupName$
FROM $S,\ G$ **WHERE** $S.GroupId = G.GroupId\ \wedge$
$\exists\ DatabaseCourse\ (\neg\ \exists\ M\ (M.StudentId = S.StudentId\ \wedge\ M.CourseId = DatabaseCourse.CourseId\ \wedge$
$M.StudentId = S.StudentId))$

2.1.2 Datalog

$DatabaseCourses(CourseId) : -Courses(CourseId, CourseName), CourseName =$ 'Базы данных'

$StudentsWithMarks(StudentId) : -Students(StudentId, \_, \_), DatabaseCourses(CourseId),$
$Marks(StudentId, CourseId, \_)$

$R(StudentId, StudentName, GroupName) : -Groups(GroupId, GroupName),$
$Students(StudentId, StudentName, GroupId),\ not\ StudentsWithMarks(StudentId)$

2.1.3 SQL

```
WITH DatabaseCourse AS (
  SELECT DISTINCT *
  FROM Courses
  WHERE Courses.CourseName = 'Базы данных'
),
     StudentsWithNoMarks AS (
       SELECT DISTINCT Students.StudentId, Students.StudentName, Students.GroupId
       FROM Students
       WHERE EXISTS(SELECT DISTINCT *
                    FROM DatabaseCourse
                    WHERE NOT EXISTS(SELECT DISTINCT *
                                     FROM Marks
                                     WHERE Marks.StudentId = Students.StudentId
                                       AND Marks.CourseId = DatabaseCourse.CourseId))
     )
SELECT StudentsWithNoMarks.StudentId,
       StudentsWithNoMarks.StudentName,
       Groups.Groupname
FROM StudentsWithNoMarks
       NATURAL JOIN Groups;
```

2.2 среди студентов, у которых есть этот предмет

2.2.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$M :: Marks$;
$DatabaseCourse :: Courses$ **WHERE** $CourseName =$ 'Базы данных';
$P :: Plan$;
SELECT $S.StudentId,\ S.StudentsName,\ G.GroupName$
FROM $S,\ G$ **WHERE** $S.GroupId = G.GroupId\ \wedge$
$\exists\ DatabaseCourse\ (\exists\ P\ (P.GroupId = S.GroupId \wedge P.CourseId = DatabaseCourse.CourseId) \wedge \neg\ \exists M\ (M.StudentId = S.StudentId\ \wedge\ M.CourseId = DatabaseCourse.CourseId))$

2.2.2 Datalog

$DatabaseCourses(CourseId) : -Courses(CourseId, CourseName), CourseName =$ 'Базы данных'

$StudentsWithoutMark(StudentId) : -Students(StudentId, \_, GroupId), DatabaseCourses(CourseId),$
$P(GroupId, CourseId, \_), not\ Marks(StudentId, CourseId, \_)$

$R(StudentId, StudentName, GroupName) : -Groups(GroupId, GroupName),$
$Students(StudentId, StudentName, GroupId), StudentsWithoutMark(StudentId)$

2.2.3 SQL

```
WITH DatabaseCourse AS (
  SELECT DISTINCT *
  FROM Courses
  WHERE Courses.CourseName = 'Базы данных'
),
     StudentsWithNoMarks AS (
       SELECT DISTINCT Students.StudentId, Students.StudentName, Students.GroupId
       FROM Students
       WHERE EXISTS(SELECT DISTINCT *
                    FROM DatabaseCourse
                    WHERE EXISTS(
                        SELECT DISTINCT *
                        FROM Plan
                        WHERE Plan.CourseId = DatabaseCourse.CourseId
                          AND Plan.GroupId = Students.GroupId
                      )
                      AND NOT EXISTS(SELECT DISTINCT *
                                     FROM Marks
                                     WHERE Marks.StudentId = Students.StudentId
                                       AND Marks.CourseId = DatabaseCourse.CourseId))
     )
SELECT StudentsWithNoMarks.StudentId,
       StudentsWithNoMarks.StudentName,
       Groups.Groupname
FROM StudentsWithNoMarks
       NATURAL JOIN Groups;
```

3 Информацию о студентах, имеющих хотя бы одну оценку у заданного лектора

3.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$M :: Marks$;
$P :: Plan$;
$LecturerCourse :: Plan$ **WHERE** $LecturerId =?$;
SELECT $S.StudentId,\ S.StudentsName,\ G.GroupName$
FROM $S,\ G$ **WHERE** $S.GroupId = G.GroupId\ \wedge\ \exists\ M\ (M.StudentId = S.StudentId\ \wedge\ \exists\ LecturerCourse$
$(LecturerCourse.GroupId = S.GroupId\ \wedge\ LecturerCourse.CourseId = M.CourseId))$

3.2 Datalog

$LecturerCourses(GroupId, CourseId) : -Plan(GroupId, CourseId, LecturerId), LecturerId =?$

$StudentsWithMarks(StudentId) : -Students(StudentId, \_, GroupId),$
$LecturerCourses(GroupId, CourseId), Marks(StudentId, CourseId, \_)$

$R(StudentId, StudentName, GroupName) : -StudentsWithMarks(StudentId),$
$Students(StudentId, StudentName, GroupId), Groups(GroupId, GroupName)$

3.2.1 SQL

```sql
WITH LecturerCourses AS (
  SELECT DISTINCT Plan.GroupId,
                  Plan.CourseId
  FROM Plan
  WHERE Plan.LecturerId = ?
),
     StudentsWithMarks AS (
       SELECT DISTINCT Students.StudentId,
                       Students.StudentName,
                       Students.GroupId
       FROM Students
       WHERE EXISTS(
                 SELECT *
                 FROM Marks
                 WHERE Marks.StudentId = Students.StudentId
                   AND EXISTS(
                     SELECT *
                     FROM LecturerCourses
                     WHERE LecturerCourses.GroupId = Students.GroupId
                       AND LecturerCourses.CourseId = Marks.CourseId
                   )
               )
     )
SELECT StudentsWithMarks.StudentId,
       StudentsWithMarks.StudentName,
       Groups.Groupname
FROM StudentsWithMarks
       NATURAL JOIN Groups;
```

4 Идентификаторы студентов, не имеющих ни одной оценки у заданного лектора

4.1 Исчисление кортежей

$S :: Students$;
$M :: Marks$;
$P :: Plan$;
$LecturerCourse :: Plan$ **WHERE** $LecturerId =?$;
SELECT $S.StudentId$
FROM S **WHERE** $\neg(\exists\ M\ (M.StudentId = S.StudentId\ \wedge\ \exists\ LecturerCourse$
$(LecturerCourse.GroupId = S.GroupId\ \wedge\ LecturerCourse.CourseId = M.CourseId)))$

4.2 Datalog

$LecturerCourses(GroupId, CourseId) : -Plan(GroupId, CourseId, LecturerId), LecturerId =?$

$StudentsWithMarks(StudentId) : -Students(StudentId, \_, GroupId),$
$LecturerCourses(GroupId, CourseId), Marks(StudentId, CourseId, \_)$

$R(StudentId) : -Students(StudentId, \_, \_), not\ StudentsWithMarks(StudentId)$

4.2.1 SQL

```
WITH LecturerCourses AS (
  SELECT DISTINCT Plan.GroupId,
                  Plan.CourseId
  FROM Plan
  WHERE Plan.LecturerId = ?
),
     StudentsWithMarks AS (
       SELECT DISTINCT Students.StudentId,
                       Students.StudentName,
                       Students.GroupId
       FROM Students
       WHERE EXISTS(
                 SELECT *
                 FROM Marks
                 WHERE Marks.StudentId = Students.StudentId
                   AND EXISTS(
                     SELECT *
                     FROM LecturerCourses
                     WHERE LecturerCourses.GroupId = Students.GroupId
                       AND LecturerCourses.CourseId = Marks.CourseId
                   )
               )
     )
SELECT Students.StudentId
FROM Students
WHERE Students.StudentId NOT IN (SELECT StudentsWithMarks.StudentId FROM StudentsWithMarks);
```

5 Студентов, имеющих оценки по всем предметам заданного лектора

Считаем, что студент должен иметь оценку по всем предметам, которые данный лектор ведёт во всех группах, причём эта оценка должна быть пославлена самим лектором (т.е. если мы запрашиваем студентов, имеющих оценки по всем курсам Кохася, оценку по матану он должен получить именно от Кохася, а не от преподавателей матана в других группах)

5.1 Исчисление кортежей

$P :: Plan$;
$S :: Students$;
$M :: Marks$;
$P :: Plan$
$G :: Groups$;
$CorrectLecturerCourse :: Plan$ **WHERE** $LecturerId =?$;
$CorrectLecturerMark ::$ **SELECT** $S.StudentId, M.CourseId$ **FROM** S, M, P **WHERE** $S.StudentId = M.StudentId \wedge$
$M.CourseId = P.CourseId \wedge S.GroupId = P.GroupId \wedge \exists CorrectLecturerCourse(CorrectLecturerCourse.LecturerId =$
$P.LecturerId)$;
SELECT $S.StudentId, S.StudentName, G.GroupName$ **FROM** S, G **WHERE**
$S.GroupId = G.GroupId \ \wedge$
$\forall CorrectLecturerCourse(\exists CorrectLecturerMark$
$(S.StudentId = CorrectLecturerMark.StudentId \ \wedge$
$CorrectLecturerCourse.CourseId = CorrectLecturerMark.CourseId))$;

5.2 Datalog

$CorrectLecturerCourses(CourseId, LecturerId) : -Plan(\_, CourseId, LecturerId), LecturerId =?$

$CorrectLecturerMarks(StudentId, CourseId) : -Students(StudentId, \_, GroupId)$
$Marks(StudentId, CourseId, \_), CorrectLecturerCourses(CourseId, LecturerId), Plan(GroupId, CourseId, LecturerId)$

$WithoutAtLeastOneMark(StudentId) : -Students(StudentId, \_, \_),$
$CorrectLecturerCourses(CourseId, \_),\ not\ CorrectLecturerMarks(StudentId, CourseId)$

$R(StudentId, StudentName, GroupName) : -Students(StudentId, StudentName, GroupId),$
$Groups(GroupId, GroupName),\ not\ WithoutAtLeastOneMark(StudentId)$

5.3 SQL

```
WITH CorrectLecturerCourses AS (
  SELECT DISTINCT Plan.CourseId,
                  Plan.LecturerId
  FROM Plan
  WHERE Plan.LecturerId = ?
),
     CorrectLecturerMarks AS (
       SELECT DISTINCT Students.StudentId,
                       Marks.CourseId
       FROM Students
              NATURAL JOIN Plan
              NATURAL JOIN Marks
       WHERE Plan.LecturerId = (SELECT DISTINCT CorrectLecturerCourses.LecturerId
                                FROM CorrectLecturerCourses)
     ),
     WithoutAtLeastOneMark AS (
```

```
       SELECT DISTINCT Students.StudentId
       FROM Students,
            CorrectLecturerCourses
       WHERE NOT EXISTS(SELECT DISTINCT *
                        FROM CorrectLecturerMarks
                        WHERE CorrectLecturerMarks.StudentId = Students.StudentId
                          AND CorrectLecturerMarks.CourseId = CorrectLecturerCourses.CourseId
         )
     ),
     WithAllMarks AS (
         (SELECT DISTINCT Students.StudentId FROM Students)
         EXCEPT
         ALL
         (SELECT DISTINCT WithoutAtLeastOneMark.StudentId FROM WithoutAtLeastOneMark)
     )
SELECT DISTINCT WithAllMarks.StudentId,
                Students.StudentName,
                Groups.GroupName
FROM WithAllMarks
       NATURAL JOIN students
       NATURAL JOIN Groups;
```

6 Для каждого студента имя и предметы, которые он должен посещать

Считаем, что студент должен посещать те предметы, которые изучает его группа, и по которым у него ещё нет положительной оценки.

6.1 Исчисление кортежей

$S :: Students$;
$P :: Plan$;
$C :: Courses$;
SELECT $S.StudentId, S.StudentName, P.CourseId, C.CourseName$ **FROM** S, P, C **WHERE** $S.GroupId = P.GroupId \ \wedge \ P.CourseId = C.CourseId \ \wedge$
$\neg \exists \, M(M.CourseId = P.CourseId \ \wedge \ M.StudentId = S.StudentId \ \wedge \ M.Mark \geq 60)$

6.2 Datalog

$R(StudentId, StudentName, CourseId, CourseName) : -Students(StudentId, StudentName, CourseId)$,
$Plan(GroupId, CourseId, \_), Courses(CourseId, CourseName), \ not \ Marks(StudentId, CourseId, \_)$

$R(StudentId, StudentName, CourseId, CourseName) : -Students(StudentId, StudentName, CourseId)$,
$Plan(GroupId, CourseId, \_), Courses(CourseId, CourseName), Marks(StudentId, CourseId, Mark), Mark < 60$

6.3 SQL

```
SELECT DISTINCT Students.StudentId,
                Students.StudentName,
                Plan.CourseId,
                Courses.CourseName
FROM (Students NATURAL JOIN Plan)
       NATURAL JOIN Courses
WHERE NOT EXISTS(SELECT *
                 FROM Marks
```

```
                WHERE Marks.CourseId = Plan.CourseId
                  AND Marks.StudentId = Students.StudentId
                  AND Marks.Mark >= 60);
```

7 По лектору всех студентов, у которых он хоть что-нибудь преподавал

7.1 Для заданного лектора

7.1.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$P :: Plan$;
$LecturerPlan :: Plan$ **WHERE** $LecturerId =?$;
SELECT $S.StudentId, S.StudentName, G.GroupName$ **FROM** S, G **WHERE** $S.GroupId = G.GroupId \ \wedge$
$\exists\ LecturePlan(LecturePlan.GroupId = S.GroupId)$

7.1.2 Datalog

$R(StudentId, StudentName, GrouName) : -Students(StudentId, StudentName, Groupid)$,
$Groups(Groupid, GroupName), Plan(GroupId, \_, LecturerId), LecturerId =?$

7.1.3 SQL

```
WITH LectureGroups AS (
  SELECT DISTINCT Plan.GroupId
  FROM Plan
  WHERE Plan.LecturerId = ?
),
     LecturerStudents AS (
       SELECT DISTINCT Students.StudentId,
                       Students.StudentName,
                       Students.GroupId
       FROM Students
       WHERE EXISTS(SELECT *
                    FROM LectureGroups
                    WHERE LectureGroups.GroupId = Students.GroupId)
     )
SELECT DISTINCT LecturerStudents.StudentId,
                LecturerStudents.StudentName,
                Groups.GroupName
FROM LecturerStudents
       NATURAL JOIN Groups;
```

7.2 Для каждого лектора

7.2.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$P :: Plan$;
$L :: Lecturers$;
SELECT $L.LecturerId, L.LecturerName, S.StudentId, S.StudentName, G.GroupName$
FROM L, S, G
WHERE $S.GroupId = G.GroupId \ \wedge\ \exists\ P(P.LecturerId = L.LecturerId \ \wedge\ P.GroupId = S.GroupId)$

7.2.2 Datalog

$R(LecturerId, LecturerName, StudentId, StudentName, GroupName) :-$
$Students(StudentId, StudentName, GroupId), Groups(GroupId, GroupName),$
$Plan(GroupId, \_, LecturerId), Lecturers(LecturerId, LecturerName)$

7.2.3 SQL

```
SELECT DISTINCT Lecturers.LecturerId,
                Lecturers.LecturerName,
                Students.StudentId,
                Students.StudentName,
                Groups.GroupName
FROM Plan
       NATURAL JOIN Lecturers
       NATURAL JOIN Students
       NATURAL JOIN Groups;
```

8 Пары студентов, такие, что все сданные первым студентом предметы сдал и второй студент

Выкинем такие пары, которые составлены из одного и того же студента.

8.1 Исчисление кортежей

$Student1 :: Students;$
$Student2 :: Students;$
$Mark1 :: Marks;$
$Mark2 :: Marks;$
$Group1 :: Groups;$
$Group2 :: Groups;$
SELECT $Student1.StudentId, Student1.StudentName, Group1.GroupName,$
$Student2.StudentId, Student2.StudentName, Group2.GroupName$
FROM $Student1, Student2, Group1, Group2$
WHERE $Student1.StudentId \neq Student2.StudentId \wedge$
$Student1.GroupId = Group1.GroupId \wedge Student2.GroupId = Group2.GroupId$
$\wedge \forall Mark1(Mark1.StudentId \neq Student1.StudentId \vee Mark1.Mark < 60 \vee \exists Mark2(Mark2.StudentId =$
$Student2.StudentId \wedge Mark2.CourseId = Mark1.CourseId \wedge Mark2.Mark \geq 60))$

8.2 Datalog

$IncorrectPairs(StudentId1, StudentId2) :-$
$Students(StudentId1, \_, \_), Students(StudentId2, \_, \_),$
$Marks(StudentId1, CourseId, Mark1),\ not\ Marks(StudentId2, CourseId, Mark2),$
$Mark1 \geq 60, Mark2 \geq 60$

$R(StudentId1, StudentId2, StudentName1, StudentName2, GroupId1, GroupId2) :-$
$Students(StudentId1, StudentName1, GroupId1), Students(StudentId2, StudentName2, GroupId2),$
$Groups(GroupId1, GroupName1), Groups(GroupId2, GroupName2),$
$not\ IncorrectPairs(StudentId1, StudentId2), StudentId1 \neq StudentId2$

8.3 SQL

```
WITH IncorrectPairs AS (
  SELECT DISTINCT Students1.StudentId AS StudentId1,
```

```
                Students2.StudentId AS StudentId2
  FROM Students AS Students1,
       Students AS Students2
  WHERE EXISTS(
            SELECT DISTINCT *
            FROM Courses
            WHERE EXISTS(SELECT *
                         FROM Marks
                         WHERE Marks.StudentId = Students1.StudentId
                           AND Marks.CourseId = Courses.CourseId
                           AND Marks.Mark >= 60)
              AND NOT EXISTS(SELECT *
                             FROM Marks
                             WHERE Marks.StudentId = Students2.StudentId
                               AND Marks.CourseId = Courses.CourseId
                               AND Marks.Mark >= 60)
          )
),
     CorrectPairs AS (
       ((SELECT DISTINCT Students1.StudentId AS StudentId1,
                         Students2.StudentId AS StudentId2
         FROM Students AS Students1,
              Students AS Students2)
        EXCEPT
        ALL
        (SELECT DISTINCT IncorrectPairs.StudentId1,
                         IncorrectPairs.StudentId2
         FROM IncorrectPairs))
     )
SELECT DISTINCT CorrectPairs.StudentId1,
                CorrectPairs.StudentId2,
                Students1.StudentName AS StudentName1,
                Students2.StudentName AS StudentName2,
                Groups1.GroupName     AS GroupName1,
                Groups2.GroupName     AS GroupName2
FROM CorrectPairs
       INNER JOIN Students AS Students1 ON CorrectPairs.StudentId1 = Students1.StudentId
       INNER JOIN Students AS Students2 ON CorrectPairs.StudentId2 = Students2.StudentId
       INNER JOIN Groups AS Groups1 ON Students1.GroupId = Groups1.GroupId
       INNER JOIN Groups AS Groups2 ON Students2.GroupId = Groups2.GroupId
WHERE CorrectPairs.StudentId1 <> CorrectPairs.StudentId2;
```

9 Такие группы и предметы, что все студенты группы сдали предмет

Считаем, что если в группе не учится ни одного студента, то все студенты этой группы успешно сдели любой предмет (даже если этого предмета нет в расписании этой группы)

9.1 Исчисление кортежей

$S :: Students$;
$G :: Groups$;
$C :: Courses$;
$M :: Marks$;
$P :: Olan$;
SELECT $G.GroupId, G.GroupName, C.CourseId, C.CourseName$ **FROM** G, C **WHERE**

$\forall\ S(S.GroupId \neq G.GroupId\ \vee$
$\exists\ M(M.StudentId = S.StudentId\ \wedge\ M.CourseId = C.CourseId\ \wedge\ M.Marks \geq 60));$

9.2 Datalog

$IncorrectPairs(GroupId, CourseId) :- Courses(CourseId, \_), Students(StudentId, \_, GroupId),$
$not\ Marks(StudentId, CourseId, Mark), Mark \geq 60$

$R(GroupId, GroupName, CourseId, CourseName) :-$
$Groups(GroupId, GroupName), Courses(CourseId, CourseName),$
$not\ IncorrectPairs(GroupId, CourseId)$

9.3 SQL

```
WITH IncorrectPairs AS (
  SELECT DISTINCT Groups.GroupId,
                  Courses.CourseId
  FROM Courses,
       (Groups
         NATURAL JOIN Students)
  WHERE NOT EXISTS(
      SELECT *
      FROM Marks
      WHERE Marks.StudentId = Students.StudentId
        AND Marks.CourseId = Courses.CourseId
        AND Marks.Mark >= 60
    )
),
     CorrectPairs AS (
       (SELECT DISTINCT Groups.GroupId,
                        Courses.CourseId
        FROM Groups,
             Courses)
       EXCEPT
       ALL
       (SELECT DISTINCT IncorrectPairs.GroupId,
                        IncorrectPairs.CourseId
        FROM IncorrectPairs)
     )
SELECT DISTINCT CorrectPairs.GroupId,
                Groups.GroupName,
                CorrectPairs.CourseId,
                Courses.CourseName
FROM CorrectPairs
       NATURAL JOIN Groups
       NATURAL JOIN Courses;
```