Королева Алиса, M3437

Структура базы данных «Деканат»:

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

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

Запрос в реляционной алгебре:

$\pi_{StudentId,\ StudentName,\ GroupId}(Students \bowtie (\sigma_{Mark\ =\ 3}Marks \ltimes \sigma_{CourseName\ =\ «Базы\ данных»}Courses))$

SQL-запрос:

```
select StudentID, StudentName, GroupID from Students
   natural join Marks
   natural join Plan
   natural join Courses where (Courses.CourseName = 'Базы данных' and Marks.MarkValue = 3);
```

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

- среди всех студентов
- среди студентов, у которых есть этот предмет

Среди всех студентов
Запрос в реляционной алгебре:

$Students - \pi_{StudentId,\ StrudentName,\ GroupId}(Students \bowtie (Marks \ltimes \sigma_{CourseName\ =\ «Базы\ данных»}Courses))$

SQL-запрос:

```
select * from students except all
        (select StudentID, StudentName, GroupID from Students
                natural join Marks
                natural join Plan
                natural join Courses where (Courses.CourseName = 'Базы данных'));
```

Среди всех студентов, у которых есть этот предмет
Запрос в реляционной алгебре:

$\pi_{StudentId,\ StudentName,\ GroupId}(Students \bowtie Plan \bowtie \sigma_{CourseName\ =\ «Базы\ данных»}Courses) - \pi_{StudentId,\ StrudentName,\ GroupId}(Students \bowtie Plan \bowtie Marks \bowtie \sigma_{CourseName\ =\ «Базы\ данных»}Courses))$

SQL-запрос:

```
select distinct StudentId, StudentName, GroupId from students
   natural join plan
   natural join Lecturers where lecturers.LecturerName = 'Георгий Корнеев'
   except all (select distinct StudentId, StudentName, GroupId from students
   natural join marks
   natural join plan
   natural join Lecturers where lecturers.LecturerName = 'Георгий Корнеев');
```

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

Запрос в реляционной алгебре:

$\pi_{StudentId,\ StudentName,\ GroupId}$ (Students ⋈ Plan ⋈ Marks ⋈ $\sigma_{LecturerName\ =\ «Георгий\ Корнеев»}$ Lecturers))

SQL-запрос:

```
select StudentId, StudentName, GroupId from students
   natural join marks
   natural join plan
   natural join Lecturers where lecturers.LecturerName = 'Георгий Корнеев';
```

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

Запрос в реляционной алгебре:

$\pi_{StudentId}$ Students - $\pi_{StudentId}$ (Students ⋈ Plan ⋈ Marks ⋈ $\sigma_{LecturerName\ =\ «Георгий\ Корнеев»}$ Lecturers))

SQL-запрос:

```
select StudentId from students
   except all (select StudentId from students
   natural join marks
   natural join plan
   natural join Lecturers where lecturers.LecturerName = 'Георгий Корнеев');
```

5. Студентов, имеющих оценки по всем предметам заданного лектора.

Запрос в реляционной алгебре:

$\pi_{StudentId,\ StudentName,\ GroupId,\ CourseId}(Students \bowtie Plan \bowtie Marks) \div \pi_{CourseId}(Plan \bowtie \sigma_{LecturerName = «Георгий\ Корнеев»}\ Lecturers)$

SQL-запрос:

```
select distinct studentid, studentname, students.groupid from students
   natural join marks
   natural join plan,
   (select distinct courseid from plan natural join lecturers where lecturers.LecturerName = 'Георгий
         Корнеев') as planlec
   except all
   (select studentid, studentname, withoutpair.groupid from
   (select distinct studentid, studentname, students.groupid, planlec.courseid from students
   natural join marks
   natural join plan,
   (select distinct courseid from plan natural join lecturers where lecturers.LecturerName = 'Георгий
         Корнеев') as planlec
   except all
   (select distinct studentid, studentname, students.groupid, courseid from students
   natural join plan
   natural join marks)) as withoutpair);
```

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

Запрос в реляционной алгебре:

$\pi_{StudentName,\ CourseName}(Students \bowtie Plan \bowtie Courses)$

SQL-запрос:

```
select studentname, coursename from students
        natural join plan
        natural join courses;
```

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

Запрос в реляционной алгебре:

$\pi_{LecturerId,\ LecturerName,\ StudentName}(Lecturers \bowtie Plan \bowtie Students)$

SQL-запрос:

```
select LecturerID, LecturerName, StudentName from Lecturers
        natural join plan
        natural join students;
```

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

Запрос в реляционной алгебре:

$\pi_{SN2,\ SN1}(\pi_{SN1,\ CourseId}\ (\rho_{StudentName\ =\ SN1}\ (Students \bowtie Plan \bowtie Marks)) \div \pi_{CourseId,\ SN2}\ (\rho_{StudentName\ =\ SN2}\ (Students \bowtie Plan \bowtie Marks)))$

SQL-запрос:

```
select distinct sndElem.SN2, StudentName as SN1 from students
        natural join plan
        natural join marks,
        (select StudentName as SN2 from students
        natural join plan
        natural join marks) as sndElem
        except all (
                select distinct SN2, SN1 from
                (select distinct StudentName as SN1, sndElem.SN2, sndElem.CID2 from students
                        natural join plan
                        natural join marks,
                        (select StudentName as SN2, CourseID as CID2 from students
                                natural join plan
                                natural join marks) as sndElem
                        except all
                        (select distinct StudentName as SN1, sndElem.SN2, courseid from students
                                natural join plan
                                natural join marks
                                natural join
                                (select distinct StudentName as SN2, courseid from students
                                        natural join plan
                                        natural join marks) as sndElem)) as proj);
```

10. Средний балл студента.

- **по идентификатору**
- **для каждого студента**

По идентификатору

Запрос в реляционной алгебре:

$avg_{Mark,\ \varnothing}(\pi_{Mark}((\sigma_{StudentId\ =\ «1»}Students) \bowtie Plan \bowtie Marks))$

SQL-запрос:

```
select avg(Mark) as Mark from students
        natural join marks
        natural join plan
        where students.studentid = '1';
```

Для каждого студента
Запрос в реляционной алгебре:

$$avg_{Mark,\ \{StudentId\}}\ (\pi_{Mark,\ StudentId}(Students \bowtie Plan \bowtie Marks))$$

SQL-запрос:

```
select avg(Mark) as Mark, StudentId from students
        natural join marks
        natural join plan
        group by StudentId;
```

11. Средний балл средних баллов студентов каждой группы.

$$avg_{Mark,\ \{GroupId\}}(avg_{Mark,\ \{StudentId,\ GroupId\}}\ (\pi_{Mark,\ StudentId,\ GroupId}(Students \bowtie Plan \bowtie Marks))$$

SQL-запрос:

```
select avg(Mark) as Mark, GroupID from
   (select avg(Mark) as Mark, StudentId, GroupID from students
   natural join marks
   natural join plan
   group by StudentId) as studAvg group by GroupID;
```

12. Для каждого студента число предметов, которые у него были, число сданных предметов и число не сданных предметов.

Запрос в реляционной алгебре:

$=\!\bowtie$ здесь означает левое соединение

$$\varepsilon_{countFailed=countAll-countPassed}(count_{countAll,\ \{StudentID\}}(Students \bowtie Plan) =\!\bowtie count_{countPassed,\ \{StudentID\}}(Students \bowtie Plan \bowtie Marks))$$

SQL-запрос:

```
select *, allandpassed.countall - allandpassed.countPassed as countfailed from
        (select t1.studentid as studentid, t1.countall as countall, t2.countPassed as countPassed from
                (select count(plan.courseid) as countall, StudentID from students
                        natural join plan
                        group by studentid) as t1,
                (select allMarks.StudentID, count(allMarks.markvalue) as countPassed from
                        (select distinct students.studentid as studentid, plan.courseid as courseid,
markvalue from students
                                natural join plan
                                left join marks on (students.studentid = marks.studentid and
marks.courseid = plan.courseid)) as allMarks
                        group by studentid) as t2
        where t1.studentid = t2.studentid) as allandpassed;
```