ДЗ 6

Содержание

1 Условие

Составление запросов в терминах исчисления кортежей, Datalog и SQL над базой данных "деканат".

2 Схема базы данных

Таблицы:

1. GroupInfo (GroupId, GroupName)
2. CourseInfo (CourseId, CourseName)
3. LecturerInfo (LecturerId, LecturerName)
4. StudentInfo (StudentId, StudentName, GroupId)
5. Marks (StudentId, CourseId, Mark)
6. WorkingPlan (CourseId, GroupId, LecturerId)

```
hw4=# CREATE TABLE GroupInfo (GroupId int NOT NULL,
                              GroupName varchar(20) NOT NULL,
                              PRIMARY KEY (GroupId));
CREATE TABLE
hw4=# CREATE TABLE CourseInfo (CourseId int NOT NULL,
                               CourseName varchar(20) NOT NULL,
                               PRIMARY KEY (CourseId));
CREATE TABLE
hw4=# CREATE TABLE LecturerInfo (LecturerId int NOT NULL,
                                 LecturerName varchar(20) NOT NULL,
                                 PRIMARY KEY (LecturerId));
CREATE TABLE
hw4=# CREATE TABLE StudentInfo (StudentId int NOT NULL,
                                StudentName varchar(20) NOT NULL,
                                GroupId int NOT NULL,
                                PRIMARY KEY (StudentId),
                                FOREIGN KEY (GroupId) REFERENCES GroupInfo(GroupId));
CREATE TABLE
hw4=# CREATE TABLE Marks (StudentId int NOT NULL,
                          CourseId int NOT NULL,
                          Mark int NOT NULL,
                          PRIMARY KEY (StudentId,CourseId),
                          FOREIGN KEY (StudentId) REFERENCES StudentInfo(StudentId),
                          FOREIGN KEY (CourseId) REFERENCES CourseInfo(CourseId));
CREATE TABLE
hw4=# CREATE TABLE WorkingPlan (CourseId int NOT NULL,
                                GroupId int NOT NULL,
                                LecturerId int NOT NULL,
                                PRIMARY KEY (CourseId,GroupId,LecturerId),
                                FOREIGN KEY (CourseId) REFERENCES CourseInfo(CourseId),
                                FOREIGN KEY (GroupId) REFERENCES GroupInfo(GroupId),
                                FOREIGN KEY (LecturerId) REFERENCES LecturerInfo(LecturerId));
CREATE TABLE
```

3 Обозначения

- Оговоримся сокращать названия таблиц до первых букв (большие), что не приводит к двузначности.
- Переменные – X, Y, Z, R.
- $\exists A.\ \exists B.\ \exists C.\ (R)$ – то же, что и $\exists A, B, C(R)$.
- Нижнее подчеркивание – "игнорируем аргумент".
- Импликация $A \to B$ – это сахар для $\neg A \vee B$.

4 Решение

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

```
select S.StudentId S.StudentName
from S
where ∃M,C.(S.StudentId = M.StudentId ∧
            M.CourseId = C.CourseId ∧
            C.CourseName = "Базы Данных" ∧
            M.Mark = 92)
```

```
HasMarkDatabases(StudentId, StudentName) :-
  S(StudentId, StudentName),
  M(StudentId, CourseId, Mark),
  C(CourseId, CourseName),
  CourseName = "Базы Данных",
  Mark = 92
```

```
select S.StudentId, S.StudentName
from StudentInfo as S, Marks as M, CourseInfo as C
where M.Mark > 60
and C.CourseName = 'Базы данных'
and M.StudentId = S.StudentId
and C.CourseId = M.CourseId;
```

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

    1. Среди всех студентов.

```
select S.StudentId S.StudentName
from S
where ¬∃C.(∃M.(S.StudentId = M.StudentId ∧
               M.CourseId = C.CourseId ∧
               C.CourseName = "Базы Данных"))
```

```
HasMarkSubject(StudentId, StudentName) :-
  S(StudentId, StudentName),
  not M(StudentId, CourseId, Mark),
  C(CourseId, CourseName),
  CourseName = "Базы Данных",
```

```
select studentid, studentname
from StudentInfo as S
where StudentId not in
    (select StudentId
     from Marks as M, CourseInfo as C
     where M.CourseId = C.CourseId and
           C.CourseName = 'Базы данных');
```

    2. Среди студентов, у которых есть этот предмет.

```
select S.StudentId S.StudentName
from S
where ∀C,W.(¬∃M.(C.CourseId = W.CourseId ∧
                 C.CourseName = 'Базы Данных' ∧
                 S.GroupId = W.GroupId ∧
                 M.StudentId = S.StudentId ∧
                 M.CourseId = W.CourseId))
```

Следует трансформировать $\forall C, W.\ \neg\exists M.\ EXP$ в $\neg(\exists C, W.\ \neg\neg\exists M.\ EXP)$, что, очевидно, равносильно $\neg\exists C.\ \neg\exists W.\ \exists M.\ EXP$:

```
HasMarkSubjectAttended(StudentId, StudentName) :-
  S(StudentId, StudentName),
  not C(CourseId, CourseName),
  not W(CourseId,GroupId,_),
  CourseName = "Базы Данных",
  M(StudentId, CourseId, _),
```

```
select S.studentid, S.studentname
from StudentInfo as S, CourseInfo as C, WorkingPlan as W
where
  C.CourseId = W.CourseId and
  C.CourseName = 'Базы данных' and
  S.GroupId = W.GroupId and
  not exists
    (select mark
     from Marks as M
     where M.StudentId = S.StudentId and
           M.CourseId = W.CourseId);
```

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

Будем искать лектора по имени, хотя понятно, что в случае двух лекторов с одинаковыми именами правильно производить поиск по $LecturerId$. Замена тривиальна, поэтому я оставлю первый вариант для большей показательности.

```
select S.StudentId S.StudentName
from S
where ∃C,L,W,M.(C.CourseId = W.CourseId ∧
                 S.GroupId = W.GroupId ∧
                 L.LecturerId = W.LecturerId ∧
                 L.LecturerName = 'Георгий Корнеев' ∧
                 M.StudentId = S.StudentId ∧
                 M.CourseId = W.CourseId))
```

```
HasMarkFromLecturer(StudentId, StudentName) :-
  S(StudentId, StudentName),
  C(CourseId, CourseName),
  W(CourseId,GroupId,LecturerId),
  L(LecturerId,LecturerName),
  LecturerName = "Георгий Корнеев",
  M(StudentId, CourseId, _)
```

```
select distinct S.StudentId, S.StudentName
from StudentInfo as S,
     CourseInfo as C,
     LecturerInfo as L,
     WorkingPlan as W,
     Marks as M
where C.CourseId = W.CourseId and
```

```
      S.GroupId = W.GroupId and
      L.LecturerId = W.LecturerId and
      L.LecturerName = 'Георгий Корнеев' and
      M.StudentId = S.StudentId and
      M.CourseId = W.CourseId;
```

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

```
select S.StudentId S.StudentName
from S
where ∀C,L,W.(¬∃M.(C.CourseId = W.CourseId ∧
                   S.GroupId = W.GroupId ∧
                   L.LecturerId = W.LecturerId ∧
                   L.LecturerName = 'Георгий Корнеев' ∧
                   M.StudentId = S.StudentId ∧
                   M.CourseId = W.CourseId))
```

Понятно, что это суждение является отрицанием предыдущего (3), следовательно:

```
DoesNotHaveMarkFromLecturer(StudentId,StudentName) :-
   ¬HasMarkFromLecturer(StudentId,StudentName);
```

```
select StudentId, StudentName
from StudentInfo as S
where StudentId not in
 (select M.StudentId
  from CourseInfo as C,
       LecturerInfo as L,
       WorkingPlan as W,
       Marks as M
  where C.CourseId = W.CourseId and
        S.GroupId = W.GroupId and
        L.LecturerId = W.LecturerId and
        L.LecturerName = 'Георгий Корнеев' and
        M.StudentId = S.StudentId and
        M.CourseId = W.CourseId);
```

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

Для каждого курса: если найдется лектор и рабочий план, которые связываются в "курс который ведет лектор X", то при этом должна существовать оценка.

```
select S.StudentId S.StudentName
from S
where
  ∃L.(L.LecturerName = 'Георгий Корнеев' ∧
      ∀C,W.((S.GroupId = W.GroupId ∧
             L.LecturerId = W.LecturerId ∧
             C.CourseId = W.CourseId) →
            ∃M.(M.StudentId = S.StudentId,
                M.CourseId = W.CourseId)))
```

Трансформируем:

$$\exists L.\,(E1 \wedge \forall C, W.\,(E2 \to \exists M.\,E3)) \Rightarrow$$
$$\exists L.\,(E1 \wedge \neg\exists C, W.\,\neg(E2 \to \exists M.\,E3)) \Rightarrow$$
$$\exists L.\,(E1 \wedge \neg\exists C, W.\,(E2 \wedge \neg\exists M.\,E3))$$

Выразим внутреннюю часть $\exists C, W.\,(E2 \wedge \neg\exists M.\,E3)$ в отдельное отношение:

```
Exp0(LecturerId,StudentId,GroupId) :-
  W(LecturerId,CourseId,GroupId),
  C(CourseId,_),
  not M(StudentId,CourseId);

HasAllLecturerMarks(StudentId, StudentName) :-
  S(StudentId,StudentName,GroupId)
  L(LecturerId,LecturerName),
  LecturerName = "Георгий Корнеев",
  not Exp0(LecturerId,StudentId,GroupId);
```

```
select S.StudentId, S.StudentName
from StudentInfo as S
where exists (
  select *
  from LecturerInfo as L
  where LecturerName = 'Георгий Корнеев' and
        not exists (
    select *
    from WorkingPlan as W,
         CourseInfo as C
    where (S.GroupId = W.GroupId and
           L.LecturerId = W.LecturerId and
           C.CourseId = W.CourseId) and
          not exists (
      select *
      from Marks as M
      where
         M.StudentId = S.StudentId and
         M.CourseId = W.CourseId)));
```

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

```
select S.StudentName C.CourseName
from S, C
where ∃W.(S.GroupId = W.GroupId ∧ C.CourseId = W.CourseId))
```

```
ShouldAttend(StudentName, CourseName) :-
  S(StudentId,StudentName,GroupId),
  C(CourseId,CourseName),
  W(CourseId,GroupId,_);
```

```
select S.StudentName, C.CourseName
from StudentInfo as S, CourseInfo as C
where (StudentId, CourseId) in (
  select StudentId, CourseId
  from WorkingPlan as W
  where S.GroupId = W.GroupId and
        C.CourseId = W.CourseId);
```

7. По лектору все студенты, у которых он хоть что-нибудь преподавал.

   Будем выводить имена для наглядности. Правильнее, конечно, ID, но существенной разницы в запросе нет.

```
select L.LecturerName, S.StudentName
from L, S
where ∃W.(S.GroupId = W.GroupId ∧ L.LecturerId = W.LecturerId)
```

```
LecturersStudents(LecturerName,StudentName) :-
  S(StudentId,StudentName,GroupId),
  L(LecturerId,LecturerName),
  W(_,GroupId,LecturerId);
```

```
select L.LecturerName, S.StudentName
from LecturerInfo as L, StudentInfo as S
where exists (
  select *
  from WorkingPlan as W
  where S.GroupId = W.GroupId and
        L.LecturerId = W.LecturerId);
```

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

```
select S1.StudentName, S2.StudentName
from S1, S2
where ∀C,M1.(C.CourseId = M1.CourseId ∧
             M1.StudentId = S1.StudentId →
             ∃M2.(C.CourseId = M2.CourseId ∧
                  M2.StudentId = S2.StudentId));
```

Преобразуем квантор всеобщности:

$$\forall C, M1.(E1 \to \exists M2.E2) \Rightarrow$$
$$\neg\exists C, M1.\neg(E1 \to \exists M2.E2) \Rightarrow$$
$$\neg\exists C, M1.(E1 \wedge \neg\exists M2.E2)$$

```
Exp0(StudentId1,StudentId2) :-
  CourseInfo(CourseId,_),
  Marks(CourseId,StudentId1),
  not Marks(CourseId,StudentId2)

StudentPairs(StudentName1,StudentName2) :-
  S(StudentId1,StudentName1,_),
  S(StudentId2,StudentName2,_),
  not Exp0(StudentId1,StudentId2);
```

```
select S1.StudentName, S2.StudentName
from StudentInfo as S1, StudentInfo as S2
where not exists (
  select *
  from CourseInfo as C, Marks as M1
  where C.CourseId = M1.CourseId and
        M1.StudentId = S1.StudentId and
        not exists (
    select *
    from Marks as M2
    where C.CourseId = M2.CourseId and
          M2.StudentId = S2.StudentId and
          M2.Mark > 60 and M1.Mark > 60
          ));
```

Автор: Михаил Волхов M3438
Created: 2016-11-15 Tue 13:08
Validate