Домашнее задание 5

Дарья Яковлева, M3439

31.10.2016

Задание: Составить выражения реляционной алгебры и соответствующие SQL-запросы для базы данных «Деканат».

Отношения

- **Groups**

  $G(\underline{GID}, GName)$

- **Students**

  $S(\underline{SID}, SName, GID)$

- **Lecturers**

  $L(\underline{LID}, LName)$

- **Courses**

  $C(\underline{CID}, CName)$

- **Marks**

  $M(Mark, \underline{CID}, \underline{SID})$

- **Academin plan**

  $P(\underline{LID}, \underline{CID}, \underline{GID})$

Операции реляционной алгебры и SQL-запросы

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

   $\pi_{S.SName,M.Mark}(\sigma_{C.CName=X}(S \bowtie C \bowtie M))$

   ```
   SELECT students.student_name
   , marks.mark_value
   FROM students
   NATURAL JOIN courses
   NATURAL JOIN marks
   WHERE courses.course_name = 'Databases';
   ```

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

(a) **среди всех студентов**

$\pi_{S.SName}(S) - \pi_{S.SName}(\sigma_{C.CName=X}(S \bowtie C \bowtie M))$

```
SELECT students.student_name
FROM students
EXCEPT
SELECT students.student_name
FROM students
NATURAL JOIN courses
NATURAL JOIN marks
WHERE courses.course_name = 'Databases';
```

(b) **среди студентов, у которых есть этот предмет**

$\pi_{S.SName}(\sigma_{C.CName=X}(S \bowtie P \bowtie C)) - \pi_{S.SName}(\sigma_{C.CName=X}(S \bowtie C \bowtie M))$

```
SELECT students.student_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN courses
WHERE courses.course_name = 'Databases'
EXCEPT
SELECT students.student_name
FROM students
NATURAL JOIN courses
NATURAL JOIN marks
WHERE courses.course_name = 'Databases';
```

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

$\pi_{S.SName}(\sigma_{L.LName=X}(S \bowtie P \bowtie C \bowtie L \bowtie_{S.SID=M.SID \wedge C.CID=M.CID} M))$

```
SELECT DISTINCT students.student_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN courses
NATURAL JOIN lecturers
INNER JOIN marks
ON students.student_id = marks.student_id AND courses.course_id = marks.course_id
WHERE lecturers.lecturer_name = 'Georgiy Korneev';
```

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

$\pi_{S.SName}(S) - \pi_{S.SName}(\sigma_{L.LName=X}(S \bowtie P \bowtie C \bowtie L \bowtie_{S.SID=M.SID \wedge C.CID=M.CID} M))$

```
SELECT students.student_name
```

```
FROM students
EXCEPT
SELECT DISTINCT students.student_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN courses
NATURAL JOIN lecturers
INNER JOIN marks
ON students.student_id = marks.student_id AND courses.course_id = marks.course_id
WHERE lecturers.lecturer_name = 'Georgiy Korneev';
```

5. **Студентов, имеющих оценки по всем предметам заданного лектора**
   $\pi_{S.SName,C.CID}(S \bowtie P \bowtie M) \div (\pi_{C.CID}(\sigma_{L.LName=X}(P \bowtie L)))$

```
SELECT students.student_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN courses
NATURAL JOIN lecturers
INNER JOIN marks
ON students.student_id = marks.student_id AND courses.course_id = marks.course_id
WHERE lecturers.lecturer_name = 'Georgiy Korneev'
GROUP BY students.student_id
HAVING COUNT(lecturers.lecturer_name) = (
SELECT COUNT(DISTINCT academicplan.course_id)
FROM academicplan
INNER JOIN lecturers
ON academicplan.lecturer_id = lecturers.lecturer_id
WHERE lecturers.lecturer_name = 'Georgiy Korneev'
)
;
```

6. **Для каждого студента имя и предметы, которые он должен посещать**
   $\pi_{S.SName,C.CName}(S \bowtie P \bowtie C)$

```
SELECT students.student_name
, courses.course_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN courses
ORDER BY students.student_name;
```

7. **По лектору всех студентов, у которых он хоть что-нибудь преподавал**
   $\pi_{S.SName}(\sigma_{L.LName=X}(S \bowtie P \bowtie L))$

```
SELECT DISTINCT students.student_name
FROM students
NATURAL JOIN academicplan
NATURAL JOIN lecturers
WHERE lecturers.lecturer_name = 'Georgiy Korneev';
```

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

   $(\pi_{S.SName,C.CID}(\sigma_{S.SName=Y,M.Mark\geq 60}(S \bowtie P \bowtie M)) \divideontimes (\pi_{S.SName,C.CID}(\sigma_{S.SName=X,M.Mark\geq 60}(S \bowtie P \bowtie M)))$

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

   $(\pi_{S.SID,C.CName}(\sigma_{M.Mark\geq 60}(S \bowtie P \bowtie C \bowtie M))) \divideontimes (\pi_{S.SID,G.GName}(S))$

10. **Средний балл студента**

    (a) **по идентификатору**

    $avg_{M.Mark,\emptyset}(\sigma_{S.SID=X}(S \bowtie M))$

    ```
    SELECT AVG(marks.mark_value)
    FROM students
    NATURAL JOIN marks
    WHERE students.student_id = 3;
    ```

    (b) **для каждого студента**

    $avg_{M.Mark,\{S.SName\}}(S \rtimes M)$

    ```
    SELECT students.student_name
    , AVG(marks.mark_value)
    FROM students
    LEFT JOIN marks
    ON students.student_id = marks.student_id
    GROUP BY students.student_id
    ORDER BY students.student_name;
    ```

11. **Средний балл средних баллов студентов каждой группы**

    $avg_{M.Mark,\{G.GName\}}(S \bowtie M)$

    ```
    SELECT groups.group_name
    , AVG(marks.mark_value)
    FROM groups
    NATURAL JOIN students
    NATURAL JOIN marks
    GROUP BY groups.group_id
    ORDER BY groups.group_name;
    ```

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

$(\varepsilon_{Total=count_{C.CID,S.SName}(\pi_{C.CID,S.SName}(S \rhd\lhd P))} \circ$

$\varepsilon_{Passed=count_{C.CID,S.SName}(\sigma_{M.Mark \geq 60}(\pi_{C.CID,S.SName}(S \rhd\lhd P \rhd\lhd M))}$

$\circ \varepsilon_{Failed=Total-Passed})(\pi_{S.SName}(S))$

Приложение. База данных.

```
CREATE TABLE IF NOT EXISTS groups (
group_id int PRIMARY KEY,
group_name varchar(100)
);

CREATE TABLE IF NOT EXISTS students (
student_id int PRIMARY KEY,
student_name varchar(100),
group_id int REFERENCES groups (group_id)
);

CREATE TABLE IF NOT EXISTS lecturers (
lecturer_id int PRIMARY KEY,
lecturer_name varchar(50)
);


CREATE TABLE IF NOT EXISTS courses (
course_id int PRIMARY KEY,
course_name varchar(50)
);

CREATE TABLE IF NOT EXISTS marks (
mark_value int,
course_id int not null REFERENCES courses (course_id),
student_id int not null REFERENCES students (student_id),
CHECK (mark_value BETWEEN 0 AND 100),
PRIMARY KEY (course_id, student_id)
);

CREATE TABLE IF NOT EXISTS academicplan (
lecturer_id int not null REFERENCES lecturers (lecturer_id),
course_id int not null REFERENCES courses (course_id),
group_id int not null REFERENCES groups (group_id),
PRIMARY KEY (lecturer_id, course_id, group_id)
);
```