University of Toronto
CSC343, Summer 2019

Outer Joins etc.: Solutions

Schema

Student(sID, surName, firstName, campus, email, cgpa)
Course(dept, cNum, name, breadth)
Offering(oID, dept, cNum, term, instructor)
Took(sID, oID, grade)

Offering[dept, cNum] ⊆ Course[dept, cNum]
Took[sID] ⊆ Student[sID]
Took[oID] ⊆ Offering[oID]

Questions

1. Which of these queries is legal?

   (a)
   ```
   SELECT count(distinct dept), count(distinct instructor)
   FROM Offering
   WHERE term >= 20089;
   ```
   (b)
   ```
   SELECT distinct dept, distinct instructor
   FROM Offering
   WHERE term >= 20089;
   ```
   (c)
   ```
   SELECT distinct dept, instructor
   FROM Offering
   WHERE term >= 20089;
   ```

   Solution:

   (a) Legal, and here is the result:
   ```
    count | count
   -------+-------
        6 |    16
   (1 row)
   ```
   (b)
   ```
   ERROR:  syntax error at or near "distinct"
   LINE 1: SELECT distinct dept, distinct instructor
   ```
   (c) Legal, and here is the result:
   ```
    dept | instructor
   ------+------------
    CSC  | Gries
    CSC  | Chechik
    HIS  | Dow
    CSC  | Heap
    ENV  | Suzuki
    HIS  | Young
    CSC  | Craig
    ANT  | Davies
    EEB  | Johancsik
    ENG  | Percy
    CSC  | Horton
   ```

```
 ANT  | Zorich
 CSC  | Truta
 ENG  | Reisman
 ENG  | Atwood
 CSC  | Jepson
(16 rows)
```

2. Under what conditions could these two queries give different results? If that is not possible, explain why.

```
SELECT surName, campus                   SELECT distinct surName, campus
FROM Student;                            FROM Student;
```

   Solution: If there were two students on the same campus with the same surname, their surname and campus would be repeated in the result of the first query, but not in the result of the second.

3. For each student who has taken a course, report their sid and the number of different departments they have taken a course in.

   Solution:

```
SELECT sid, count(distinct dept)
FROM Took JOIN Offering ON Took.oid = Offering.oid
GROUP BY sid;
  sid  | count
-------+-------
   157 |     5
 11111 |     3
 98000 |     6
 99132 |     4
 99999 |     5
(5 rows)
```

   The ‘distinct‘ is necessary, otherwise every course the student has taken (unless it had a ‘NULL‘ value for ‘dept‘) would count, even if they were all in the same department!

```
SELECT sid, count(dept)
FROM Took JOIN Offering ON Took.oid = Offering.oid
GROUP BY sid;
  sid  | count
-------+-------
 98000 |    15
 99132 |     7
 99999 |    12
   157 |    15
 11111 |     5
(5 rows)
```

4. Suppose we have two tables with content as follows:

```
SELECT *
FROM One;

 a  |  b
----+-----
  1 |   2
  6 |  12
    | 100
 20 |
(4 rows)
```

```
SELECT *
FROM Two;

  b  |  c
-----+-----
   2 |   3
 100 | 101
  20 |  21
   2 |   4
   2 |   5
(5 rows)
```

(a) What query could produce this result?

```
 a |  b  |  c
---+-----+-----
 1 |   2 |   3
 1 |   2 |   4
 1 |   2 |   5
   |  20 |  21
   | 100 | 101
(5 rows)
```

Solution:

```
SELECT * FROM One NATURAL RIGHT JOIN Two;
```

But note that postgreSQL changes the column order on this query, actually producing:

```
  b  | a |  c
-----+---+-----
   2 | 1 |   3
   2 | 1 |   4
   2 | 1 |   5
  20 |   |  21
 100 |   | 101
(5 rows)
```

This would also provide the same rows, although in different column order:

```
SELECT * FROM Two NATURAL LEFT JOIN One;
```

(b) What query could produce this result?

```
 a  |  b  |  c
----+-----+-----
  1 |   2 |   3
  1 |   2 |   4
  1 |   2 |   5
  6 |  12 |
    | 100 | 101
 20 |     |
(6 rows)
```

Solution:

```
SELECT * FROM One NATURAL LEFT JOIN Two;
```

But note that postgreSQL changes the column order on this query, actually producing:

```
  b  | a  |  c
-----+----+-----
   2 |  1 |   3
   2 |  1 |   4
   2 |  1 |   5
  12 |  6 |
 100 |    | 101
     | 20 |
(6 rows)
```

This would also provide the same rows, although in different column order:

```
SELECT * FROM Two NATURAL RIGHT JOIN One;
```