Bases de données

Lecture 8

Nabil Mustafa

mustafa@lipn.univ-paris13.fr

Concepts and Keywords

from, where, select
group by, having
distinct, case, null, as, order by
with, union, intersect, except
exists, all, some, in, is, not

aggregations:
max, min, sum, count, avg

joins:
inner, left/right, outer,
natural, cross, using, on

PRACTICE EXAM

For each question, answer **TRUE** or **FALSE**.
If FALSE, explain what the query does.

take your time to read each question carefully

don't hurry: think **clearly** about each question

not evaluated, so no stress or pressure

use this opportunity to test yourself

if logic not 100% clear to you, **ask me to explain**

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.

```
SELECT S.name
FROM student AS S
WHERE ('Fall', 2009) IN
              (SELECT semester, year
                FROM takes
                WHERE takes.id = S.id);
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.

```
SELECT S.name
FROM student AS S
WHERE ('Fall', 2009) =
            SOME (SELECT semester, year
                  FROM takes
                  WHERE takes.id = S.id);
```

✓


```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.


```
SELECT DISTINCT student.name
FROM takes NATURAL INNER JOIN student
WHERE takes.semester = 'Fall' AND
      takes.year = 2009;
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.

```
SELECT name
FROM student
WHERE EXISTS(SELECT *
             FROM takes
             WHERE takes.id = student.id AND
                   semester = 'Fall' AND
                   year = 2009);
```

✓


```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.


```
SELECT name
FROM student
WHERE EXISTS (SELECT COUNT(*)
              FROM takes
              WHERE takes.id = student.id AND
                    semester = 'Fall' AND
                    year = 2009);
```

✗

COUNT(*) returns an integer, even if it is 0
so list contains one integer and it will print all student names

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Find the names of all students who have taken a class in Fall 2009.

```
SELECT student.name
FROM takes, student
WHERE takes.semester = 'Fall' AND
      takes.year = 2009 AND
      student.id = takes.id;
```

✗

it contains duplicates, must add **DISTINCT**


```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```


List all pairs of students who have taken at least one class together.

```
SELECT A.name, B.name
FROM (student NATURAL INNER JOIN takes) AS A
        CROSS JOIN
     (student NATURAL INNER JOIN takes) AS B
WHERE A.course_id = B.course_id AND
      A.sec_id = B.sec_id AND
      A.semester = B.semester AND
      A.year = B.year
GROUP BY A.name, A.id, B.name, B.id
HAVING COUNT(*) >= 1;
```

✗

prints pairs where the two names are the same

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

List all pairs of students who have taken at least one class together.

```
SELECT A.name, B.name
FROM (student NATURAL INNER JOIN takes) AS A
        CROSS JOIN
      (student NATURAL INNER JOIN takes) AS B
WHERE A.course_id = B.course_id AND
      A.sec_id = B.sec_id AND
      A.semester = B.semester AND
      A.year = B.year AND
      A.id <> B.id
GROUP BY A.name, A.id, B.name, B.id
HAVING COUNT(*) >= 1;
```

✗

prints each pair of students twice

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

List all pairs of students who have taken at least one class together.

```
SELECT A.name, B.name
FROM (student NATURAL INNER JOIN takes) AS A
        CROSS JOIN
      (student NATURAL INNER JOIN takes) AS B
WHERE A.course_id = B.course_id AND
      A.sec_id = B.sec_id AND
      A.semester = B.semester AND
      A.year = B.year AND
      A.id < B.id
GROUP BY A.name, A.id, B.name, B.id
HAVING COUNT(*) >= 1;
```

✓

List all pairs of students who have taken most classes together.

```
SELECT A.name, B.name
FROM (student NATURAL INNER JOIN takes) AS A,
     (student NATURAL INNER JOIN takes) AS B
WHERE A.course_id = B.course_id AND
      A.sec_id = B.sec_id AND
      A.semester = B.semester AND
      A.year = B.year AND
      A.id < B.id
GROUP BY A.name, A.id, B.name, B.id
HAVING COUNT(*) > ALL(
  SELECT COUNT(*)
  FROM (student NATURAL INNER JOIN takes) AS A,
       (student NATURAL INNER JOIN takes) AS B
  WHERE A.course_id = B.course_id AND
        A.sec_id = B.sec_id AND
        A.semester = B.semester AND
        A.year = B.year AND
        A.id < B.id
  GROUP BY A.id, B.id);
```

✗

$>$ ALL condition never satisfied

List all pairs of students who have taken most classes together.

```
SELECT A.name, B.name
FROM (student NATURAL INNER JOIN takes) AS A,
     (student NATURAL INNER JOIN takes) AS B
WHERE A.course_id = B.course_id AND
      A.sec_id = B.sec_id AND
      A.semester = B.semester AND
      A.year = B.year AND
      A.id < B.id
GROUP BY A.name, A.id, B.name, B.id
HAVING COUNT(*) >= ALL(
  SELECT COUNT(*)
  FROM (student NATURAL INNER JOIN takes) AS A,
       (student NATURAL INNER JOIN takes) AS B
  WHERE A.course_id = B.course_id AND
        A.sec_id = B.sec_id AND
        A.semester = B.semester AND
        A.year = B.year AND
        A.id < B.id
  GROUP BY A.id, B.id);
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give the total number of students taught by each teacher (the same student in two classes is counted twice), including teachers who have not taught any students.

```
SELECT A.name, count(A.course_id)
FROM (teacher LEFT OUTER JOIN teaches
                    USING (id)) as A
        CROSS JOIN
     takes as B
WHERE (A.course_id, A.sec_id, A.semester, A.
    year) = (B.course_id, B.sec_id, B.semester
    , B.year)
GROUP BY A.name, A.id;
```

✗

teachers who have not taught any class removed in **WHERE**

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give the total number of students taught by each teacher (the same student in two classes is counted twice), including teachers who have not taught any students.

```
SELECT teacher.name, count(course_id)
FROM (takes INNER JOIN teaches USING
         (course_id, sec_id, semester, year))
     RIGHT OUTER JOIN teacher ON
          teaches.id = teacher.id
GROUP BY teacher.name, teacher.id
ORDER BY count(course_id) DESC;
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give the number of 'A' grades given by **each** teacher.

```
SELECT teacher.name, count(course_id)
FROM (takes INNER JOIN teaches USING
        (course_id, sec_id, semester, year))
     INNER JOIN teacher ON
         teaches.id = teacher.id
WHERE grade LIKE 'A'
GROUP BY teacher.name, teacher.id
ORDER BY count(course_id) DESC;
```

✗

teachers who gave no 'A' grades not printed, as using **INNER JOIN**

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give the number of 'A' grades given by **each** teacher.

```
SELECT teacher.name, count(course_id)
FROM (takes INNER JOIN teaches USING
       (course_id, sec_id, semester, year))
     RIGHT OUTER JOIN teacher ON
        teaches.id = teacher.id
WHERE grade LIKE 'A'
GROUP BY teacher.name, teacher.id
ORDER BY count(course_id) DESC;
```

✗

teachers who gave no 'A' grades not printed, as **WHERE** condition applied *before* **GROUP BY** statement

Give the number of 'A' grades given by **each** teacher.

```
WITH mytakes (id, course_id, sec_id, semester,
     year, grade) AS
   (SELECT id, course_id, sec_id, semester,
           year, grade
    FROM takes
    WHERE grade LIKE 'A')

SELECT teacher.name, COUNT(course_id)
FROM (mytakes INNER JOIN teaches USING
         (course_id, sec_id, semester, year))
     RIGHT OUTER JOIN teacher ON
         teaches.id = teacher.id
GROUP BY teacher.name, teacher.id
ORDER BY count(course_id) DESC;
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Compute the total credits over all taken courses for each year.

```
SELECT year, SUM(credits)
FROM takes NATURAL JOIN course
GROUP BY year;
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give all pairs of teachers and students where the student has taken the course of the teacher, together with how many times that student has been in a course of that teacher.

```
SELECT teacher.name, student.name, COUNT(*)
FROM (teacher NATURAL JOIN teaches)
       NATURAL JOIN
     (takes NATURAL JOIN student)
GROUP BY teacher.name, student.name;
```

✗

the second natural join over not the same **id** which is wrong as it is teacher.id and student.id. The output is an empty table.

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give all pairs of teachers and students where the student has taken the course of the teacher, together with how many times that student has been in a course of that teacher.

```
SELECT teacher.name, student.name, COUNT(*)
FROM (teacher NATURAL JOIN teaches) INNER JOIN
     (takes NATURAL JOIN student) USING
        (course_id, sec_id, semester, year)
GROUP BY teacher.name, student.name;
```

✓

Is the result of these two queries always equal ?

```sql
SELECT course_id, sec_id, semester, year,
       AVG(tot_cred)
FROM takes NATURAL JOIN student
WHERE year = 2009
GROUP BY course_id, sec_id, semester, year
HAVING COUNT(id) >= 2;
```

✓

```sql
SELECT course_id, sec_id, semester, year,
      AVG(tot_cred)
FROM (takes NATURAL JOIN student)
     NATURAL JOIN section
WHERE year = 2009
GROUP BY course_id, sec_id, semester, year
HAVING COUNT(id) >= 2;
```

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

Give the pair of teachers and students where the student has taken at least two classes from that teacher.

```sql
SELECT mytable.tn, mytable.sn
FROM (SELECT teacher.name AS tn, student.name
             AS sn, COUNT(*) AS tc
      FROM (teacher NATURAL JOIN teaches)
             INNER JOIN
           (takes NATURAL JOIN student) USING
             (course_id, sec_id, semester, year)
    GROUP BY teacher.name, student.name) AS mytable
WHERE tc >= 2;
```

✓

```
takes (id, course_id, sec_id, semester, year, grade)
student (id, name, dept_name, tot_cred)
course (course_id, title, dept_name, credits)
teacher (id, name, dept_name, salary)
section (course_id, sec_id, semester, year, building, rn, time_id)
teaches (id, course_id, sec_id, semester, year)
department (dept_name, building, budget)
```

For **each** student s, list the other student who has taken the most classes with s.