CIS5516 HW2

Zhijia Chen

Basic SQL queries:

1. Find the names of all the instructors from Biology department.
   SQL query:
   SELECT DISTINCT name FROM instructor WHERE dept_name = 'Biology';
   Relational algebra query:
   $\Pi_{name}\left(\sigma_{dept\_name=\text{"Biology"}}(instructor)\right)$
   Tuple calculus query:
   $\{t \mid \exists s \in instructor(t[name] = s[name] \wedge s[dept\_name] = \text{"Biology"})\}$
   Domain calculus query:
   $\{< nam > \mid < id, nam, dep, sal > \in instructor \wedge dep = \text{"Biology"}\}$

```
CIS5516=# SELECT DISTINCT name FROM instructor WHERE dept_name = 'Biology';
 name
-------
 Crick
(1 row)
```

2. Find the names of courses in Computer Science department which have 3 credits.
   SQL query:
   SELECT title FROM course WHERE dept_name = 'Comp. Sci.' AND credits = 3;
   Relational algebra query:
   $\Pi_{title}\left(\sigma_{dept\_name=\text{"Comp. Sci."}\wedge \text{credits}=3}(course\ )\right)$
   Tuple calculus query:
   $\{\text{t} \mid \exists s \in course\ (t[title] = s[title] \wedge s[dept\_name] = \text{"Comp. Sci."}$
   $\wedge\, s[credits] = 3)\}$
   Domain calculus query:
   $\{< tit > \mid < cou, tit, dep, cre > \in course \wedge dep = \text{"Comp. Sci."} \wedge cre = 3\}$

```
CIS5516=# SELECT title FROM course WHERE dept_name = 'Comp. Sci.' AND credits = 3;
        title
--------------------------
 Robotics
 Image Processing
 Database System Concepts
(3 rows)
```

3. For the student with ID 12345 (or any other value), show all course_id and title of all courses registered for by the student.
   SQL query:
   SELECT course.course_id, title FROM (takes INNER JOIN course ON takes.course_id = course.course_id) WHERE id = '00128';
   Relational algebra query:
   $\Pi_{course\_id,title}\left(\sigma_{id=\text{"00128"}}(takes \bowtie course)\right)$
   Tuple calculus query:

$\{t | \exists s \in course(t[course\_id] = s[course\_id] \wedge t[title] = s[title] \wedge \exists u \in takes(s[course\_id] = u[course\_id] \wedge \mathrm{u}[id] = "00128"))\}$

Domain calculus query:

$\{< cou, tit > | \exists dep, cre(< cou, tit, dep, cre > \in course \wedge \exists sec, sem, yea, gra(< "00128", cou, sec, sem, yea, gra > \in takes))\}$

```
CIS5516=# SELECT course.course_id, title FROM (takes INNER JOIN course ON takes.course_id =
course.course_id) WHERE id = '00128';
 course_id |           title
-----------+----------------------------
 CS-101    | Intro. to Computer Science
 CS-347    | Database System Concepts
(2 rows)
```

4. For the student with ID 12345 (or any other value), show the total number of credits taken by that student. Use SQL aggregation on courses taken by the student.
   SQL query:
   SELECT sum(credits) FROM (takes INNER JOIN course ON takes.course_id = course.course_id) WHERE id = '00128';
   Relational algebra query: Not applicable.
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# SELECT sum(credits) FROM (takes INNER JOIN course ON takes.course_id = course.course_id)
WHERE id = '00128';
 sum
-----
   7
(1 row)
```

5. Display the total credits for each student, along with the ID of the student; don't worry about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted).
   SQL query:
   SELECT id, sum(credits) FROM (takes INNER JOIN course ON takes.course_id = course.course_id) GROUP BY id;
   Relational algebra query: Not applicable.
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# SELECT id, sum(credits) FROM (takes INNER JOIN course ON takes.course_id =
course.course_id) GROUP BY id;
  id   | sum
-------+-----
 76543 |   7
 19991 |   3
 00128 |   7
 98765 |   7
 54321 |   8
 12345 |  14
 45678 |  11
 98988 |   8
 55739 |   3
 76653 |   3
 44553 |   4
 23121 |   3
(12 rows)
```

6. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names).
   SQL query:
   SELECT DISTINCT name FROM (student INNER JOIN takes ON student.id = takes.id INNER JOIN course ON takes.course_id = course.course_id) WHERE course.dept_name = 'Comp. Sci.';
   Relational algebra query:
   $\Pi_{name}\left(\sigma_{dept\_name=\text{"Comp. Sci."}}(student \bowtie takes)\right)$
   Tuple calculus query:
   $\{t \mid \exists s \in student(t[name] = s[name] \wedge \exists u \in takes(s[course\_id] = u[course\_id]$
   $\wedge\, u[id] = s[id] \wedge \exists v \in course(v[course\_id] = s[course\_id]$
   $\wedge\, v[dept\_name] = \text{"Comp. Sci."})))\}$
   Domain calculus query:
   $\{< nam > \mid \exists id, dep, tot(< id, cou, dep, tot > \in student \wedge \exists sec, sem, yea, gra($
   $< id, cou, sec, sem, yea, gra > \in takes$
   $\wedge\, \exists tit, crd(< cou, tit, \text{"Comp. Sci."}, crd > \in course)))\}$

```
CIS5516=# SELECT DISTINCT name FROM (student INNER JOIN takes ON student.id = takes.id INNER JOIN
course ON takes.course_id = course.course_id) WHERE course.dept_name = 'Comp. Sci.';
   name
----------
 Bourikas
 Brown
 Levy
 Shankar
 Williams
 Zhang
(6 rows)
```

7. Display the IDs of all instructors who have never taught a course. (Note: Oracle uses the keyword minus in place of except).
   SQL query:
   SELECT id FROM instructor EXCEPT (SELECT id FROM teaches);

Relational algebra query:

$\Pi_{id}\big(\sigma(instructor)\big) - \Pi_{id}\big(\sigma(teaches)\big)$

Tuple calculus query:

$\{t \mid \exists s \in instructor(t[id] = s[id] \land \forall u \in teaches(s[id] \neq u[id]))\}$

Domain calculus query:

$\{< id > \mid \exists nam, dep, sal\big(< id, nam, dep, sal > \in instructor$

$\land \forall(cou, sec, sem, yea)(\neg < id, cou, sec, sem, yea > \in teaches)\big)\}$

```
CIS5516=# SELECT id FROM instructor EXCEPT (SELECT id FROM teaches);
  id
-------
 33456
 58583
 76543
(3 rows)
```

Intermediate SQL queries:

1. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment. (Don't worry about those that had no students taking that section.).
   SQL query:
   WITH tmp AS (SELECT count(id) AS enrol_num FROM (takes INNER JOIN section ON takes.course_id = section.course_id AND takes.sec_id = section.sec_id AND takes.year = section.year) GROUP BY takes.course_id, takes.sec_id, takes.year, takes.semester HAVING count(id) > 0) SELECT max(enrol_num) AS max_enrol, min(enrol_num) AS min_enrol FROM tmp;
   Relational algebra query: Not applicable.
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# WITH tmp AS (SELECT count(id) AS enrol_num FROM (takes INNER JOIN section ON
takes.course_id = section.course_id AND takes.sec_id = section.sec_id AND takes.year = section.year)
GROUP BY takes.course_id, takes.sec_id, takes.year, takes.semester HAVING count(id) > 0) SELECT
max(enrol_num) AS max_enrol, min(enrol_num) AS min_enrol FROM tmp;
 max_enrol | min_enrol
-----------+-----------
         6 |         1
(1 row)
```

2. Find all sections that had the maximum enrollment (along with the enrollment), using a subquery.
   SQL query:
   WITH tmp AS (SELECT count(id) AS enrol_num, takes.year, takes.semester, takes.course_id, takes.sec_id FROM (takes INNER JOIN section ON takes.course_id = section.course_id AND takes.sec_id = section.sec_id AND takes.year = section.year) GROUP BY takes.course_id, takes.sec_id, takes.year, takes.semester HAVING count(id) > 0) SELECT year, semester, course_id, sec_id, enrol_num FROM tmp WHERE enrol_num = (SELECT max(enrol_num) FROM tmp);

Relational algebra query: Not applicable.
Tuple calculus query: Not applicable.
Domain calculus query: Not applicable.

```
CIS5516=# WITH tmp AS (SELECT count(id) AS enrol_num, takes.year, takes.semester, takes.course_id,
takes.sec_id FROM (takes INNER JOIN section ON takes.course_id = section.course_id AND takes.sec_id =
section.sec_id AND takes.year = section.year) GROUP BY takes.course_id, takes.sec_id, takes.year,
takes.semester HAVING count(id) > 0) SELECT year, semester, course_id, sec_id, enrol_num FROM tmp
WHERE enrol_num = (SELECT max(enrol_num) FROM tmp);
 year | semester | course_id | sec_id | enrol_num
------+----------+-----------+--------+-----------
 2009 | Fall     | CS-101    | 1      |         6
(1 row)
```

3. Find all courses whose identifier starts with the string "CS-1".
   SQL query:
   SELECT title FROM course WHERE course_id LIKE 'CS-1%';
   Relational algebra query:
   $\Pi_{title}\left(\sigma_{course\_id=\text{"CS-1\%"}}(course)\right)$
   Tuple calculus query:
   $\{t \mid \exists s \in course(t[title] = s[title] \wedge s[course\_id] = \text{"CS-1\%"})\}$
   Domain calculus query:
   $\{< tit > \mid < cou, tit, dep, sal > \in course \wedge cou = \text{"CS-1\%"}\}$

```
CIS5516=# select title from course where course_id like 'CS-1%';
           title
----------------------------
 Intro. to Computer Science
 Game Design
(2 rows)
```

Advanced SQL queries:

1. Create a view faculty showing only the ID, name, and department of instructors.
   SQL query:
   CREATE VIEW faculty AS SELECT id, name, dept_name FROM instructor;
   Relational algebra query:
   $\rho_{faculty}\left(\Pi_{id,name,dept_{name}}(\sigma(instructor))\right)$
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# CREATE VIEW faculty AS SELECT id, name, dept_name FROM instructor;
CREATE VIEW
CIS5516=# select * from faculty limit 1;
  id   |    name    | dept_name
-------+------------+------------
 10101 | Srinivasan | Comp. Sci.
(1 row)
```

2. Create a view CSinstructors, showing all information about instructors from the Comp. Sci. department.
   SQL query:
   CREATE VIEW CSinstructor AS SELECT * FROM instructor WHERE dept_name = 'Comp. Sci.';
   Relational algebra query:
   $\rho_{CSinstructor}\left(\sigma_{dept\_name=\text{"Comp. Sci."}}(instructor)\ \right)$
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# CREATE VIEW CSinstructor AS SELECT * FROM instructor WHERE dept_name = 'Comp. Sci.';
CREATE VIEW
CIS5516=# SELECT * FROM CSinstructor LIMIT 1;
  id   |    name    | dept_name  |  salary
-------+------------+------------+----------
 10101 | Srinivasan | Comp. Sci. | 65000.00
(1 row)
```

3. Find all rooms that have been assigned to more than one section at the same time. Display the rooms along with the assigned sections. (Hint: use WITH or views.)
   SQL query:
   SELECT course_id, sec_id, room_number, building FROM section WHERE (building, room_number, time_slot_id) IN (SELECT building, room_number, time_slot_id FROM section GROUP BY building, room_number, time_slot_id HAVING count(course_id) > 1);
   Relational algebra query: Not applicable.
   Tuple calculus query: Not applicable.
   Domain calculus query: Not applicable.

```
CIS5516=# select course_id, sec_id, room_number, building from section where (building, room_number,
time_slot_id) in (select building, room_number, time_slot_id from section group by building,
room_number, time_slot_id having count(course_id) > 1);
 course_id | sec_id | room_number | building
-----------+--------+-------------+----------
 CS-190    | 2      | 3128        | Taylor
 CS-319    | 2      | 3128        | Taylor
 CS-347    | 1      | 3128        | Taylor
 EE-181    | 1      | 3128        | Taylor
(4 rows)
```