Database Systems — CSci 4380
Midterm Exam #2 Answers

In this database, we will store information about a university offering all of its courses online. The database stores detailed information about the courses including which online sites are used as a resource for different components of courses. Additionally, it stores information about exam dates, office hours etc. For each attribute, example values are provided.

Note: Each class in the database may have zero to many of each of the following: instructors, sites for resources, office hours, class meetings, exams and students. Example date formatting: 2020/11/05.

```
-- All courses offered in the university
CREATE TABLE courses (
     classcode      varchar(40) PRIMARY KEY -- ex: CSCI-4380
     , coursename    varchar(200) -- ex: Database Systems
     , credits      int          -- ex: 4
     , department   varchar(40)  -- ex: Computer Science
) ;

-- Classes are offerings of a specific course in a specific semester,
-- year and section. We will assume no cross listed courses for simplicity.
CREATE TABLE classes(
     , crn           int   PRIMARY KEY
     , classcode      varchar(40)
     , semester       varchar(10) -- ex: Fall, Spring, Summer I, Summer II
     , year           int         -- ex: 2020, 2021
     , sectionno      int         -- ex: 1,2
     , FOREIGN KEY (classcode) REFERENCES courses (classcode)
     , UNIQUE (classcode, semester, year, sectionno)
) ;

-- All instructors in the university
CREATE TABLE instructors(
     , id           int   PRIMARY KEY
     , name         varchar(40)    -- ex: Sibel Adali
     , email        varchar(40)  -- ex: adalis@rpi.edu
     , onlineroom   varchar(40)  -- ex: rensselaer.webex.com/meet/adalis
     , note         text
) ;

-- Who teaches which course(s)
CREATE TABLE teaches(
     crn              int
     , instructorid   int
     , PRIMARY KEY (crn, instructorid)
     , FOREIGN KEY (crn) REFERENCES classes(crn)
       ON DELETE CASCADE ON UPDATE CASCADE
     , FOREIGN KEY (instructorid) REFERENCES instructors(id)
       ON DELETE CASCADE ON UPDATE CASCADE
) ;

-- When classes meet
CREATE TABLE classmeetings(
     crn       int
     , dayofweek   varchar(10)  -- ex: Monday, Tuesday
     , starttime   time         -- ex: time '14:30'
     , duration    int          -- in minutes, ex: 150
     , note        text
     , PRIMARY KEY (crn, dayofweek, starttime)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE ON UPDATE CASCADE
) ;
```

```sql
-- When classes have office hours
CREATE TABLE officehours(
     crn         int
     , dayofweek     varchar(10)  -- ex: Monday, Tuesday
     , starttime     time         -- ex: time '18:00'
     , endtime       time         -- ex: time '19:30'
     , PRIMARY KEY (crn, dayofweek, starttime)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
) ;

-- When classes have exams
CREATE TABLE exams(
     crn       int
     , examname   varchar(40) -- ex: Exam 1, Exam 2, Final Exam
     , examdate   date        -- ex: date '2020/11/02'
     , pointvalue int         -- ex: 12, 20
     , starttime  time        -- ex: time '14:30'
     , duration   int         -- in minutes, ex: 130
     , note       text
     , PRIMARY KEY (crn, examname)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE ON UPDATE CASCADE
) ;

-- All sites that are used for different courses.
CREATE TABLE sites(
     , sitename     varchar(40) PRIMARY KEY --ex: slack, discord, teams, submitty
     , bestbrowser  varchar(40)  --ex: firefox, chrome
     , generalurl   varchar(100) --ex: webex.com
) ;

-- Which sites are used for which courses, example resourcetypes are
-- discussions, hw, videos, exams, coursenotes, meetings, officehours
CREATE TABLE resourcesites(
     rid              int  PRIMARY KEY
     , crn            int
     , resourcetype   varchar(100) -- ex: see above.
     , sitename       varchar(40)
     , resourceurl    varchar(100)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
     , FOREIGN KEY (sitename) REFERENCES sites(sitename) ON UPDATE CASCADE
) ;

-- All students in the database
CREATE TABLE students (
     studentid     int   PRIMARY KEY
     , email       varchar(100)
     , firstname   varchar(100)
     , lastname    varchar(100)
) ;

-- Who is enrolled in which class.
CREATE TABLE enrollment (
    crn            int
    , studentid    int
    , PRIMARY KEY (crn, studentid)
    , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
    , FOREIGN KEY (studentid) REFERENCES students(studentid) ON DELETE CASCADE

) ;
```

Here is a shorthand of the schema:

```
courses(classcode, coursename, credits, department)
classes(crn, classcode, semester, year, sectionno)
instructors(id, name, email, onlineroom, note)
teaches(crn, instructorid)
classmeetings(crn, dayofweek, starttime, duration, note)
officehours(crn, dayofweek, starttime, endtime)
exams(crn, examname, examdate, pointvalue, starttime, duration, note)
sites(sitename, bestbrowser, generalurl)
resourcesites(rid, crn, resourcetype, sitename, resourceurl)
students(studentid, email, firstname, lastname)
enrollment(crn, studentid)
```

Version 1 (v1).

Question 1 (10 points). Return the `crn` and `coursename` for all classes offered in `Fall 2020` (`semester`, `year`) that have at least one class meeting on `Tuesday` (`dayofweek`) and at least one class meeting on `Friday` (`dayofweek`), and at least one set of office hours on `Wednesday` (`dayofweek`).

Answer.

```sql
select DISTINCT
   cl.crn
   , c.coursename
from
   courses c
   , classes cl
   , classmeetings cm1
   , classmeetings cm2
   , officehours o
where
   c.classcode = cl.classcode
   and cl.crn = cm1.crn
   and cl.crn = cm2.crn
   and cl.crn = o.crn
   and cl.semester = 'Fall'
   and cl.year = 2020
   and cm1.dayofweek = 'Tuesday'
   and cm2.dayofweek = 'Friday'
   and o.dayofweek = 'Wednesday';
```

Version 2 (v2).

Question 1 (10 points). Return the `crn` and `coursename` for all classes offered in `Fall 2020` (`semester`, `year`) with at least one office hour on `Monday` (`dayofweek`) and at least one office hour on `Wednesday` (`dayofweek`) and uses `Submitty` (`sitename`) as a resource site.

Answer.

```
select DISTINCT
   cl.crn
   , c.coursename
from
   courses c
   , classes cl
   , officehours o1
   , officehours o2
   , resourcesites r
where
   c.classcode = cl.classcode
   and cl.crn = o1.crn
   and cl.crn = o2.crn
   and cl.crn = r.crn
   and cl.semester = 'Fall'
   and cl.year = 2020
   and o1.dayofweek = 'Monday'
   and o2.dayofweek = 'Wednesday'
   and r.sitename = 'Submitty';
```

Version 3 (v3).

Question 1 (10 points). Return the `crn` and `coursename` for all classes offered in `Fall 2020` (`semester`, `year`) that use both `Piazza` and `Submitty` (`sitename`) as resources and have at least one class meeting on `Tuesday` (`sitename`).

Answer.

```
select DISTINCT
   cl.crn
   , c.coursename
from
   courses c
   , classes cl
   , classmeetings cm
   , resourcesites r1
   , resourcesites r2
where
   c.classcode = cl.classcode
   and cl.crn = cm.crn
   and cl.crn = r1.crn
   and cl.crn = r2.crn
   and cl.semester = 'Fall'
   and cl.year = 2020
   and r1.sitename = 'Piazza'
   and r1.sitename = 'Submitty'
   and cm.dayofweek = 'Tuesday'; --typo in the first exam said sitename
```

Version 4 (v4).

Question 1 (10 points). Return the `crn` and `coursename` for all classes offered by the same professor both in `Fall 2020` and `Spring 2021` (`semester, year`).

Answer.

```
select DISTINCT
   cl.crn
   , c.coursename
from
   courses c
   , classes cl1
   , classes cl2
   , teaches t1
   , teaches t2
where
   c.classcode = cl1.classcode
   and c.classcode = cl2.classcode
   and cl1.crn = t1.crn
   and cl2.crn = t2.crn
   and cl1.semester = 'Fall'
   and cl1.year = 2020
   and cl2.semester = 'Spring'
   and cl2.year = 2021
   and t1.instructorid = t2.instructorid;
```

Version 1 (v1).

Question 2 (12 points). Return the `id` and `name` of all instructors in the database who taught a class that has at least one class meeting on `Tuesday` (`dayofweek`) and have no office hours on `Tuesday` (`dayofweek`).

Answer.

```
SELECT DISTINCT
    i.id
    , i.name
FROM
    instructors i
    , teaches t
    , classmeetings cm
WHERE
    i.id = t.instructorid
    and t.crn = cm.crn
    and cm.dayofweek = 'Tuesday'
    and cm.crn NOT IN (select crn from officehours where dayofweek = 'Tuesday');


SELECT DISTINCT
    i.id
    , i.name
FROM
    instructors i
    , teaches t
    , classmeetings cm
WHERE
    i.id = t.instructorid
    and t.crn = cm.crn
    and cm.dayofweek = 'Tuesday'
    and NOT EXISTS (SELECT * FROM officehours o
                   WHERE o.dayofweek = 'Tuesday' and cm.crn = o.crn);


SELECT DISTINCT
    i.id
    , i.name
FROM
    instructors i
      join teaches t on i.id = t.instructorid
      join classmeetings cm on t.crn = cm.crn and cm.dayofweek = 'Tuesday'
      left join officehours o on o.dayofweek = 'Tuesday' and cm.crn = o.crn
WHERE
    o.crn is null ;

-- Be careful that using EXCEPT here may not work because the condition
-- is about the courses, not the instructors
```

Version 2 (v2).

Question 2 (12 points). Return the `id` and `name` of all instructors that are using `Submitty` (`sitename`) as a resourcesite in a class that has an exam on `2020/11/5` (`examdate`) and are not using `Gradescope` (`sitename`) for `exams` (`resourcetype`) for the same class.

Answer.

```
SELECT DISTINCT
   i.id
   , i.name
FROM
   instructors i
   , teaches t
   , resourcesites s
   , exams e
WHERE
   i.id = t.instructorid
   and t.crn = s.crn
   and e.crn = t.crn
   and e.examdate = date '2020/11/5'
   and s.sitename = 'Submitty'
   and s.crn not in (SELECT crn
                     FROM resourcesites
                     WHERE sitename = 'Gradescope' and resourcetype='exams');
-- Be careful that using except here may not work because the condition
-- is about the courses, not the instructors

-- Solutions similar to v1 also exist for this version.
```

Version 1 (v1).

Question 3 (12 points). Return the `crn` and `coursename` for all classes offered in `Fall` or `Spring` (`semester`) of 2020 (`year`) that satisfy at least one of the following two conditions: either (1) the class is using at least three different sites as resourcesites, or (2) the class has exactly two instructors.

Answer.

```
SELECT
   c1.crn
   , c.coursename
FROM
   courses c
   , classes cl
   , resourcesites r
WHERE
   c.classcode = cl.classcode
   and r.crn = cl.crn
   and cl.year = 2020
   and cl.semester in ('Spring', 'Fall')
GROUP BY
   cl.crn, c.coursename
HAVING
   count(distinct r.sitename) >= 3
UNION
SELECT
   c1.crn
   , c.coursename
FROM
   courses c
   , classes cl
   , teaches t
WHERE
   c.classcode = cl.classcode
   and t.crn = cl.crn
   and cl.year = 2020
   and cl.semester in ('Spring', 'Fall')
GROUP BY
   cl.crn, c.coursename
HAVING
   count(distinct t.instructorid) = 2;


SELECT
   c1.crn
   , c.coursename
FROM
   courses c
    join classes cl on c.classcode = cl.classcode
                 and cl.year = 2020 and cl.semester in ('Spring', 'Fall')
    left join resourcesites r on r.crn = cl.crn
    left join teaches t on t.crn = cl.crn
GROUP BY
   cl.crn, c.coursename
HAVING
   count(distinct r.sitename) >= 3
   or  count(distinct t.instructorid) = 2;

-- Note: we need left join here because a course may not have any resource site
-- or instructors
```

Version 2 (v2).

Question 3 (12 points). Return the `id` of all students who satisfy at least one of the following two conditions: either (1) the student is taking at least 3 classes with prefix `CSCI-` (`coursename`) in `Fall 2020` (`semester, year`), or (2) the student's courses in `Fall 2020` use at least 5 different `sitenames` in resourcesites.

Answer.

```
SELECT
   e.studentid
FROM
   courses c
   , classes cl
   , enrollment e
WHERE
   c.classcode = cl.classcode
   and c.crn = cl.crn
   and c.crn = e.crn
   and cl.year = 2020
   and cl.semester = 'Fall'
   and c.classcode like 'CSCI-%' -- also accept c.classname
GROUP BY
   e.studentid
HAVING
   count(distinct c.crn) >= 3
UNION
SELECT
   e.studentid
FROM
   courses c
   , classes cl
   , enrollment e
   , resourcesites r
WHERE
   c.classcode = cl.classcode
   and c.crn = cl.crn
   and c.crn = e.crn
   and c.crn = r.crn
   and cl.year = 2020
   and cl.semester = 'Fall'
GROUP BY
   e.studentid
HAVING
   count(distinct r.sitename) >= 5 ;

-- Similar solution to v1 may exist here as well., but is likely
-- more complex than v1 solution. Two different versions of class may need
-- to be left joined, one for CSCI- prefix and one for sites.
```

Version 1 (v1).

Question 4 (10 points). For each professor in the database who is teaching at least one class, return the `id` and `name` of the professor, the number of courses they are teaching and the number of different sites they are using for `discussions` (`resources.resourcetype`).

Answer.

```sql
SELECT
   i.id
   , i.name
   , count(distinct t.crn)
   , count(distinct r.sitename)
FROM
   instructors i
     join teaches t on i.id = t.instructorid
     left join resourcesites r
         on t.crn = r.crn and r.resourcestype = 'discussions'
GROUP BY
   i.id
   , i.name --- i.name is optional if grouped by i.id,
           --- this is needed if grouping by t.instructorid
;

--- Note using regular join would not return zero values which this
--- query allows.

SELECT
   i.id
   , i.name
   , count(distinct t.crn)
   , (SELECT count(distinct r.sitename)
      FROM resourcesites r, teaches t
      WHERE r.crn = t.crn
            AND t.instructorid = i.id
            AND r.resourcestype = 'discussions')
FROM
   instructors i
     join teaches t on i.id = t.instructorid
GROUP BY
   i.id
   , i.name --- i.name is optional if grouped by i.id,
           --- this is needed if grouping by t.instructorid
;
```

Version 2 (v2).

Question 4 (10 points). For each student in the database who is enrolled in at least one class, return their `id`, `name` and the total number of exams they are taking in `November 2020` (`examdate`) and the total number of days on which they have exams (`examdate`).

Answer.

```
-- Two interpretations for number of days: all days they have exams, or
-- or days they have exams in November. We will solve both.

-- All days they have exam in November:

SELECT
   s.studentid
   , s.name
   , count(ex.crn)
   , count(distinct ex.examdate)
FROM
   students s
   join enrollment e on e.studentid = s.studentid
   left join exams ex on ex.crn = s.crn
                    and s.examdate >= date '2020/1/11'
                    and s.examdate >= date '2020/31/11'
GROUP BY
   s.studentid
;


-- All days they have exam in November:

SELECT
   s.studentid
   , s.name
   , count(ex.crn)
   , (select count(distinct ex2.examdate) from enrollment e2, exams ex2
     where ex2.crn = e2.crn and e2.studentid = s.studentid)
FROM
   students s
   join enrollment e on e.studentid = s.studentid
   left join exams ex on ex.crn = s.crn
                    and s.examdate >= date '2020/1/11'
                    and s.examdate >= date '2020/31/11'
GROUP BY
   s.studentid
;
```

Version 1 (v1).

Question 5 (10 points). Return the `crn`, `dayofweek` and `starttime` of all office hours for a class with only one exam.

Answer.

```
SELECT
    crn
    , dayofweek
    , startime
FROM
    officehours
WHERE
    1 = (SELECT count(*)
         FROM exams
         WHERE exams.crn = officehours.crn) ;

---- OR -----


SELECT
    o.crn
    , o.dayofweek
    , o.startime
FROM
    officehours o
    , exams e
WHERE
    e.crn = o.crn
GROUP BY
    o.crn
    , o.dayofweek
    , o.startime
HAVING
    count(*) = 1;

-- Ok to have/not have Limit 1 here.
```

Version 2 (v2).

Question 5 (10 points). Return the `crn`, `examdate` of all exams for a class with exactly one block of office hours (i.e. single tuple).

Answer.

```
SELECT
    crn
    , examdate
FROM
    exams
WHERE
    1 >= (SELECT count(*)
         FROM officehours
         WHERE exams.crn = officehours.crn) ;


---- OR -----

SELECT
    e.crn
    , e.examdate
FROM
    exams e
    , officehours o
WHERE
    e.crn = o.crn
GROUP BY
    e.crn
    , e.examdate
HAVING
    count(*)= 1;
-- Ok to have/not have LIMIT 1
```

Version 1 (v1).

Question 6 (14 points). Find the `dayofweek` and `starttime` in `Fall 2020` (`semester, year`) that has the highest number of students on `Webex` (`sitename`) at the same time in a class meeting (assuming all enrolled students join at the start time of their respective classes). Return the `dayofweek`, `starttime` and the total number of students.

It is possible to solve this question with a single SQL query (including subqueries). However, you can break this problem into a multi-step code, using multiple SQL queries, creating temporary tables and dropping them if you wish.

Answer.

```
WITH webexcounts AS(
SELECT
   cm.dayofweek
   , cm.time
   , count(*) as numstudents
FROM
    classmeetings cm
    , resourcesites rs
    , enrollment e
    , classes c
WHERE
    cm.crn = rs.crn
    and rs.crn = e.crn
    and c.crn = cm.crn
    and c.semester = 'Fall'
    and c.year = 2020
    and rs.resourcetype = 'meetings'
    and rs.sitename = 'webex'
GROUP BY
    cm.dayofweek
    , cm.time
)
SELECT
    *
FROM
    webexcounts w
WHERE
    w.numstudents = (SELECT max(numstudents) FROM webexcounts);

-- limit 1 solution does not work for this question because
-- there can be multiple dayofweek/time with the max students.
```

Version 2 (v2).

Question 6 (14 points). Find students who are fortunate enough to use the smallest number of distinct `sitenames` in the resourcesites table for their courses in `Fall 2020` (`semester, year`). Return the `studentid`, `name` of the students and the number of distinct `sitenames` they use.

It is possible to solve this question with a single SQL query (including subqueries). However, you can break this problem into a multi-step code, using multiple SQL queries, creating temporary tables and dropping them if you wish.

Answer.

```
WITH ssites AS (
SELECT
   s.id
   , s.name
   , count(DISTINCT r.sitename) as numsites
FROM
   students s
   , enrollment e
   , resourcesites r
   , classes c
WHERE
   s.studentid = e.studentid
   and e.crn = r.crn
   and e.crn = c.crn
   and e.semester = 'Fall'
   and e.year = 2020
GROUP BY
   s.id
)
SELECT
   *
FROM
   ssites
WHERE
   numsites = (select min(numsites) FROM ssites) ;

-- same in v1, limit 1 does not work.
```

Version 1 (v1).

Question 7 (10 points). Enroll `Baby Yoda` (`firstname='Baby'` and `lastname='Yoda'`) in section `1` (`sectionno`) of all classes in `Spring 2021` (`semester, year`) that are using `Signal` (`sitename`) as a resourcesite.

Answer.

```
insert into enrollment
select distinct
   c.crn
   , s.studentid
from
   classes c
   , resourcesites r
   , students s
where
   c.semester = 'Spring'
   and c.year = 2021
   and c.crn = r.crn
   and c.section = 1
   and r.sitename = 'Signal'
   and s.firstname = 'Baby'
   and s.lastname = 'Yoda';
```

Version 2 (v2).

Question 7 (10 points). For all courses `Baby Yoda` (`firstname='Baby'` and `lastname='Yoda'`) is taking in `Spring 2021` (`semester, year`), update the resourcesites table to change the `sitename` value for any `discussion` resourcetype to `Signal`.

Answer.

```
UPDATE resourcesites
SET sitename = 'Signal'
WHERE
   resourcetype = 'discussion'
   AND crn IN (SELECT e.crn
              FROM enrollment e, classes c, student s
              WHERE e.studentid = s.studentid
                   and c.crn = e.crn
               and c.semester='Spring'
                   and c.year = 2021
          and s.firstname = 'Baby'
                   and s.lastname = 'Yoda');

---

UPDATE resourcesites
SET sitename = 'Signal'
WHERE
   resourcetype = 'discussion'
   AND EXISTS (SELECT *
              FROM enrollment e, classes c, student s
              WHERE e.studentid = s.studentid
                   and c.crn = e.crn
               and c.semester='Spring'
                   and c.year = 2021
          and s.firstname = 'Baby'
                   and s.lastname = 'Yoda'
                   and e.crn = resourcesite.crn);
```

Version 1 (v1).

Question 8 (12 points). Write a single transaction block (`BEGIN;/COMMIT;`) to reprimand professors for using `TikTok` (`sitename`) as a resourcesite in their classes.

To accomplish this, update the `note` attribute for the instructor to string `'inappropriate resource: TikTok'` and cancel any classes using `TikTok` by removing all tuples for these classes from any table in the database.

You do not need to create any temporary tables to achieve this, but if you wish, you can create such a table to aid you in this query. Drop the table at the end of your transaction block.

Answer.

```
BEGIN ;
UPDATE instructor
SET note = 'inappropriate resource use: tiktok'
WHERE id IN (SELECT t.instructorid
            FROM teaches t, resourcesites r
       WHERE t.crn = r.crn and r.sitename = 'TikTok') ;

DELETE FROM classes
WHERE crn IN (SELECT crn FROM resourcesites
             WHERE sitename = 'TikTok') ;
COMMIT ;

----

UPDATE instructor
SET note = 'inappropriate resource use: tiktok'
WHERE EXISTS (SELECT *
            FROM teaches t, resourcesites r
       WHERE t.crn = r.crn and r.sitename = 'TikTok'
                 AND  t.instructorid = instructor.id ) ;

DELETE FROM classes
WHERE EXISTS (SELECT * FROM resourcesites r
        WHERE r.sitename = 'TikTok' and r.crn = classes.crn) ;
COMMIT ;
```

Version 2 (v2).

Question 8 (12 points). Write a single transaction block (`BEGIN;/COMMIT;`) for the following.

First, enroll all students taking a class from a professor `Boba Fett` (`name`) in `Fall 2020` (`semester, year`) in the class with `crn=10302020` (already in the database).

Then, cancel any classes taught by `Boba Fett` in `Fall 2020` by removing all tuples for these classes from any table in the database.

You do not need to create any temporary tables to achieve this, but if you wish, you can create such a table to aid you in this query. Drop the table at the end of your transaction block.

Answer.

```sql
BEGIN;
INSERT INTO enrollment
SELECT DISTINCT 10302020, e.studentid
FROM teaches t, instructors i, enrollment e
WHERE t.instructorid = i.id and t.crn = e.crn and i.name = 'Boba Fett';

DELETE FROM classes
WHERE crn IN (SELECT t.crn FROM teaches t, instructors i
             WHERE t.instructorid = i.id and i.name = 'Boba Fett');
END ;

-- alternate version for DELETE similar to v1.
```

Question 9 (10 points). You are given the following two tables with no tuples initially.

```
CREATE TABLE a(val INT) ;
CREATE TABLE b(val INT) ;

CREATE FUNCTION atrgf () RETURNS trigger AS $$
    BEGIN
        IF NEW.val > 3 THEN
            INSERT INTO b SELECT sum(val) FROM a;
   END IF ;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER atrg BEFORE INSERT ON a
    FOR EACH ROW EXECUTE FUNCTION atrgf();

CREATE FUNCTION btrgf () RETURNS trigger AS $$
    BEGIN
        IF NEW.val - OLD.val > 5 THEN
           INSERT INTO a VALUES(NEW.val);
   END IF ;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER btrg AFTER UPDATE ON b
    FOR EACH ROW EXECUTE FUNCTION btrgf();
```

Check all tuples in the database after the following transaction executes:

```
BEGIN ;
INSERT INTO a VALUES(4) ;
INSERT INTO a VALUES(2) ;
INSERT INTO a VALUES(5) ;
UPDATE b SET val = 14 WHERE val = 4 ;
INSERT INTO a VALUES(8) ;
UPDATE b SET val = val*10 WHERE val < 10 ;
COMMIT;
```

```
Table a (4)
Table a (2)
Table a (5)
Table a (8)
Table a (0)
Table a (40)
Table a (20)
Table a (60)
Table a (80)
Table a (110)

Table b (0)
Table b (NULL)
Table b (4)
Table b (2)
Table b (6)
Table b (11)
Table b (19)
Table b (20)
Table b (60)
Table b (79)
Table b (80)
Table b (110)
```

Answer.

```
Table a (4)
Table a (2)
Table a (5)
Table a (8)
Table a (60)
Table b (NULL)
Table b (11)
Table b (19)
Table b (60)
```