NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
MIDTERM ASSESSMENT FOR
Semester 2 AY2024/2025

IT5008 Database Design and Programming

March 2025 Time Allowed 60 minutes

INSTRUCTIONS TO CANDIDATES

1. This assessment paper contains **9** questions and comprises **6** printed pages, including this page. The last **2** pages is left blank for your working.
2. The total marks for this assessment is **50**.
3. This is a **CLOSED-BOOK** assessment. You are only allowed to refer to one double-sided A4-size paper.
4. Shade and write your student number in the answer sheet. Do **NOT** write your name.
5. Submit only the answer sheet at the end of the assessment.
6. Write all your answers in the answer sheet provided.
7. Your answer must fit into the given space (*i.e.*, number of lines, box, *etc*).
8. Please read the additional instruction for multiple choice (MCQ) and multiple response (MRQ) questions. More specific instructions *may* be given at the beginning of each section, please read them carefully.
    - Any answer not in the space provided will not be graded.
    - For MCQ and MRQ, **shade your answer in the corresponding bubble on the answer sheet**.
    - For MCQ, select the most appropriate answer.
    - For MCQ, if multiple answers are equally appropriate, pick one and shade **ONLY** the chosen answer on the answer sheet. Do **NOT** shade more than one answer.
    - For MRQ, shade **ALL** correct answers. Partial marks *may* be given, but not guaranteed.
    - For MRQ, choose "*None of the above*" only if there is no other appropriate choice. If chosen, you should not choose any other choice.
9. All SQL query in this assessment are run on PostgreSQL 16.
10. We will use the monospace `NULL` to represent NULL-values. We will represent string with single-quote `'string'`.

Question	Points
1 - 4	14
5 - 8	24
9	12
TOTAL	50

GOOD LUCK

In this paper, we will use the following schema.

```
CREATE TABLE employee (
  eid   INT PRIMARY KEY,
  name  VARCHAR(256) NOT NULL
);

CREATE TABLE room (
  name  VARCHAR(32) PRIMARY KEY,
  num   INT NOT NULL,
  floor INT NOT NULL,
  UNIQUE (num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);

CREATE TABLE booked_room (
  name    VARCHAR(32) REFERENCES room(name),
  date    DATE,
  purpose VARCHAR(256) NOT NULL,
  PRIMARY KEY (name, date)
);

CREATE TABLE booking (
  eid   INT NOT NULL REFERENCES employee(eid),
  name  VARCHAR(32),
  date  DATE,
  FOREIGN KEY (name, date) REFERENCES booked_room(name, date),
  PRIMARY KEY (name, date)
);
```

We further consider the following **preliminary data**.

`employee` **Table**

eid	name
101	'Amy'
102	'Dieter'
103	'Isidor'

`booked_room` **Table**

name	date	purpose
'Meeting Room 1'	2025-03-11	'Meeting
'Meeting Room 2'	2025-03-10	'Meeting
'Meeting Room 2'	2025-03-09	'Meeting
'Meeting Room 3'	2025-03-09	'Meeting

`room` **Table**

name	num	floor
'Meeting Room 1'	1	1
'Meeting Room 2'	1	2
'Meeting Room 3'	2	1
'Meeting Room 4'	2	3

`booking` **Table**

eid	name	date
101	'Meeting Room 2'	2025-03-10
103	'Meeting Room 1'	2025-03-11

You may tear this page as we will only collect the answer sheet.

Data Definition Language (14 points)

1. (3 points) **(MCQ)** Which of the following `INSERT` statement will cause an error. Treat each insertion separately starting from the given preliminary data.

   A. `INSERT INTO room VALUES ('Meeting Room 5A', 2, 2);`

   B. `INSERT INTO room VALUES ('Meeting Room 5B', 1, 3);`

   C. `INSERT INTO room VALUES ('Meeting Room 5C', 3, -1);`

   D. `INSERT INTO room VALUES ('Meeting Room 5D', 3, 1);`

   E. *None of the above.*

   Notes: C. `floor` must be greater than `0`.

2. (3 points) **(MCQ)** We focus on the table `room`. We want to reimplement the table `room` differently. Assume that by modifying `room`, we also modify the other tables referencing `room`.

   Which of the following alternative capture exactly the same constraint as the schema above?

   A.
   ```
   CREATE TABLE room (
     name  VARCHAR(32) UNIQUE NOT NULL,
     num   INT,
     floor INT,
     PRIMARY KEY (num, floor),
     CHECK (num > 0 AND floor > 0)
   );
   ```

   B.
   ```
   CREATE TABLE room (
     name  VARCHAR(32) UNIQUE,
     num   INT,
     floor INT,
     PRIMARY KEY (num, floor),
     CHECK (num > 0),
     CHECK (floor > 0)
   );
   ```

   C.
   ```
   CREATE TABLE room (
     name  VARCHAR(32) UNIQUE NOT NULL,
     num   INT,
     floor INT,
     PRIMARY KEY (num, floor),
     CHECK (num > 0 OR floor > 0)
   );
   ```

   D.
   ```
   CREATE TABLE room (
     name  VARCHAR(32),
     num   INT,
     floor INT,
     PRIMARY KEY (name, num, floor),
     CHECK (num > 0),
     CHECK (floor > 0)
   );
   ```

   E. *None of the above.*

   Notes: A. `UNIQUE` and `NOT NULL` is equivalent to `PRIMARY KEY`. Additionally, two different `CHECK` must both be satisfied. Hence, it is equivalent to `AND`.

3. (4 points) **(MRQ)** Which of the following constraints are not enforced by the given schema? Select all the answers that apply. Choose "*None of the above*" only if no other answers apply. If you choose "*None of the above*", you should not choose other answers.

   A. Rooms can be identified by the floor number (`floor`) and room number (`num`).

   B. Rooms can be identified by the room name (`name`).

   C. No two different employees can book the same room on the same day (`date`).

   D. No two different rooms can be booked by the same employee on the same day (`date`).

   E. *None of the above.*

Notes: D. The rest are enforced.

We can enumerate the choices and see how they are enforced.

A. Enforced by having `NOT NULL` on `room.num` and `room.floor` as well as `UNIQUE (num, floor)`.

B. Enforced by having `PRIMARY KEY` on `room.name`.

C. Enforced by having `PRIMARY KEY` on the pair `(booking.name, booking.date)`. Consider a triple `(eid1, name, date)`. We cannot have another triple `(eid2, name, date)`.

D. Not enforced. To be enforced, we need to prevent the following pair of triple from coexisting: `(eid, name1, date)` and `(eid, name2, date)`.

E. *None of the above*.

4. (4 points) **(MRQ)** Which of the following constraints are not enforced by the given schema? Select all the answers that apply. Choose "*None of the above*" only if no other answers apply. If you choose "*None of the above*", you should not choose other answers.

A. Employees are identified by their employee id (`eid`).

B. There can be a booked room (*i.e.,* according to `booked_room`) but not by any employee.

C. It is possible that an employee does not book any room at all.

D. It is possible that a room is not booked at all.

E. *None of the above*.

Notes: E. All are enforced.

We can enumerate the choices and see how they are enforced.

A. Enforced by having `PRIMARY KEY` on `employee.eid`.

B. Enforced by allowing insert into `booked_room` but not inserting to `booking`.

C. Enforced by allowing insert into `employee` but not inserting to `booking`.

D. Enforced by allowing insert into `room` but not inserting to `booking_room`.

E. *None of the above*.

SQL Query (24 points)

5. (4 points) **(MRQ)** Which of the following query **cannot be answered** given the schema above? Note that there are no other constraints except those specified by the schema. Select all answers that apply.

   A. Find the different employees (`employee.eid`) that have never booked any room (`room.name`).

   B. Find the different floors (`room.floor`) that have no room (`room.name`).

   C. Find the different room (`room.name`) that has never become a booked room (*i.e.,* according to `booked_room`).

   D. Find the different room (`room.name`) that has never been booked (*i.e.,* according to `booking`).

   E. *None of the above.*

Notes: B. We do not know what other floors are there. Consider the data above, we do not know if there are floor 4 or not.

6. (4 points) **(MRQ)** Which of the following query is guaranteed to have no duplicate? Note that you should consider any possible valid instance satisfying the schema and not just the current data. Select all answers that apply.

A.
```
SELECT b.date, r.num, r.floor
FROM room r, booked_room b
WHERE b.name = r.name;
```

B.
```
SELECT b.date, r.name
FROM room r, booked_room b
WHERE b.name = r.name;
```

C.
```
SELECT e.eid, b.name, b.date
FROM employee e, booking b
WHERE b.eid = e.eid;
```

D.
```
SELECT e.name, b.name, b.date
FROM employee e, booking b
WHERE b.eid = e.eid;
```

E. *None of the above.*

Notes: A, B, and C.

For A and B, note that `b.name = r.name`, so the primary key of both `room` an d`booked_room` are equal. If the primary key or candidate key appears in the `SELECT` clause, then the query has no duplicate values.

For C and D, we only have `b.eid = e.eid`. So the query has no duplicate only if the primary key or candidate key of BOTH `employee` and `booking` are in the `SELECT` clause. This is not satisfied by option D.

We can construct a counter example of two different employees having the same name `e.name` but different employee id `e.eid`.

7. (4 points) **(Fill in the Blank)** What is the result of the following query? Fill in the table in the answer sheet. You do not have to use all rows. The order of the rows does not matter.

```
SELECT br.name, br.date, e.name AS employee
FROM booked_room br, booking b, employee e
WHERE br.name = b.name AND b.eid = e.eid
  AND br.name IN (
        SELECT r.name
        FROM room r
        WHERE r.floor = 1
);
```

Notes: Here, the inner subquery is *uncorrelated*. So, we can evaluate it on its own separately from the outer query. The inner subquery finds all the room name located on floor number 1. There are 2 of them: (i) `'Meeting Room 1'` and (ii) `'Meeting Room 3'`.

We then look at `booked_room` table to find the rows corresponding to this. However, when looking at the outer query, we will only consider the rows for which the room has been booked by an employee. Hence, we ignore `'Meeting Room 3'` because it is a booked room but no employee is booking it.

Finally, we obtain the name of the meeting room, the date the room become a booked room, and the name of the employee booking the room. This gives us the following table.

name	**date**	**employee**
`'Meeting Room 1'`	`2025-03-11`	`'Isidor'`

8. (12 points) **(SQL)** Find the different room that is not booked (*i.e.,* according to `booked_room`) on 2025-03-09. Order the result in **descending order** of room name. Recap that you can write a date by enclosing it in single quote like a string (*e.g.,* `'2025-03-09'`).

Using the data above, we will have the following result.

name	**num**	**floor**
`'Meeting Room 1'`	`1`	`1`
`'Meeting Room 4'`	`2`	`3`

(a) (6 points) Use algebraic query in your solution.

Notes:

```
-- Using FULL JOIN (here, left)
SELECT r.name, r.num, r.floor
FROM room r FULL JOIN booked_room b
  ON r.name = b.name AND b.date = '2025-03-09'
WHERE b.name ISNULL
ORDER BY r.name DESC;

-- Using EXCEPT
SELECT * FROM room r
EXCEPT
SELECT r1.name, r1.num, r1.floor
FROM room r1, booked_room br
WHERE r1.name = br.name AND br.date = '2025-03-09'
ORDER BY r.name DESC;
```

(b) (6 points) Use nested query in your solution. Note that you cannot use algebraic query for this part.

Notes:

```
-- Using NOT IN
SELECT * FROM room r
WHERE r.name NOT IN (
  SELECT b.name FROM booked_room b
  WHERE b.date = '2025-03-09'
)
ORDER BY r.name DESC;
```

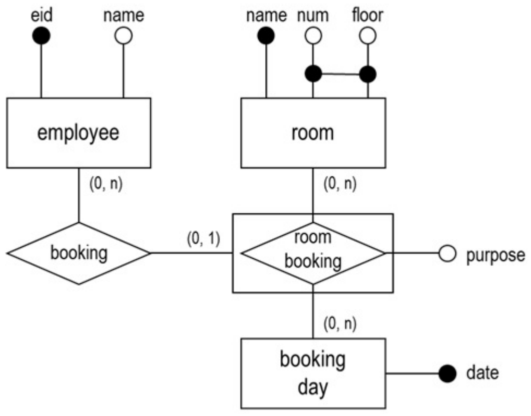
```
-- Using NOT EXISTS
SELECT * FROM room r
WHERE NOT EXISTS (
  SELECT b.name FROM booked_room b
  WHERE b.date = '2025-03-09' AND b.name = r.name  -- correlation
)
ORDER BY r.name DESC;
```

Entity-Relationship Diagram (12 points)

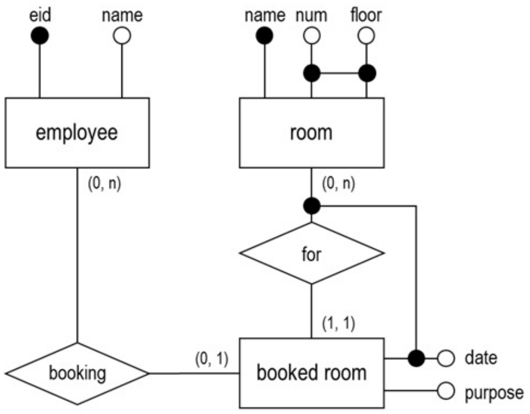
9. (12 points) **(ERD)** Given the schema, draw one possible entity-relationship diagram such that by using schema translation rule in the lecture (*i.e.,* 3 rules + 3 exceptions), we will get the schema.

Notes:

Alternative #1



Alternative #2



END OF PAPER

This space is intentionally left blank.