CS2102: Database Systems (AY2022-2023 – Sem 2)

Final Exam

Instructions

1. Please read **ALL** instructions carefully.
2. This assessment contains **SIXTEEN (16)** questions including subquestions:
   (a) There are 11 Multiple Response Questions (MRQ):
   1.2 - 1.4, 2.2, 3.1b, 3.2b, 4.1(a-d) - 4.2
   (b) There is 1 Fill in the Blank Questions (FITB): 1.1
   (c) There are 2 Multiple Choice Question (MCQ): 2.1, 4.3,
   (d) There are 2 True/False Question: 3.1a, 3.2a
3. There is a total of **EIGHTY (80)** points for this assessment.
4. Answer **ALL** questions.
5. All the assessment is be done using Examplify:
   (a) This is a secure assessment (no internet ; no other software)
   (b) This is a closed-book exam ; you are allowed 1 A4 cheat sheet double-sided
   (c) The choices on Examplify may be in a different order
6. No additional time will be given to submit.
7. Use the question number shown on Examplify when asking question.
   - If the answer is clear from the question pdf/Examplify, we will reply with "No Comment".
8. Failure to follow each of the instructions above may result in deduction of your marks.

Good Luck!

1 SQL

We want to create a database for room booking. Our database should contain:

- **Employees**
  - Identified by employee id (`INT`).
  - Must store the name of the employee (`TEXT`).
- **Rooms**
  - Identified by floor number (`INT`) and room number (`INT`).
  - Must store the name of the room (`TEXT`).

A room can be booked by employees following the constraints below.

- When a room is booked, it is booked for the whole day.
- A room can be booked by at most one employee on any single day. In other words, no two different employees can book the same room on the same day. However, it is possible that the room is not booked by any employee.
- An employee can book at most one room on any single day. In other words, no two different rooms can be booked by the same employee on the same day. However, it is possible that the employee does not book any room.

1.1 (14 points) **CREATE TABLE.** *This questions is split into two on Examplify*

Your answer for each blank should only be exactly **one** simple constraint (*e.g.,* ***`UNIQUE`****, etc*) which cannot be a `CHECK` constraint or an attribute. You should not add unnecessary constraints. Unnecessary constraints includes constraints not specified above as well as constraints already enforced in other parts (*e.g.,* ***`NOT NULL`*** *or* ***`UNIQUE`*** *on an attribute with* ***`PRIMARY KEY`*** *constraint*). If you feel that there should be nothing inside the box (*e.g., because it will only be unnecessary constraints*), simply put in "-" (*without the quote*). An empty box is treated as *unanswered*. Lastly, in some cases, the order may not matter (*e.g, some primary key or foreign key attributes ordering*).

```
CREATE TABLE Employees (
  eid    INT  [          ] ,
  ename  TEXT [          ]
);

CREATE TABLE Rooms (
  rfloor INT  [          ] ,
  rnum   INT  [          ] ,
  rname  TEXT [          ] ,
  PRIMARY KEY ( [      ], [      ] )
);
```

```
CREATE TABLE Bookings (
  bfloor INT [          ] ,
  bnum   INT [          ] ,
  beid   INT [          ] ,
  bday   DATE [         ] ,
  PRIMARY KEY ( [      ], [      ], [      ] ),
  UNIQUE ( [      ], [      ] ),
  FOREIGN KEY ( [      ], [      ] ) REFERENCES Rooms(rnum, rfloor),
  FOREIGN KEY ( [      ] ) REFERENCES Employees
);
```

1.2 (6 points) **INSERT INTO.** Assume that the following insert statements have been performed successfully.

```
INSERT INTO Employees VALUES (1, 'A');
INSERT INTO Employees VALUES (2, 'B');
INSERT INTO Employees VALUES (3, 'C');
INSERT INTO Rooms VALUES (1, 1, 'R1');
INSERT INTO Rooms VALUES (1, 2, 'R2');
INSERT INTO Rooms VALUES (2, 1, 'R3');
```

Select ALL insert statement that will **NOT** be rejected. For choices with two insert statements, select only if none are rejected. Consider each choices independently and assumes all the constraints are enforced correctly. Note that dates can be inserted as a string in the format `'YYYY-MM-DD'`, where `YYYY` is the year, `MM` is the month, and `DD` is the date.

(A) `INSERT INTO Bookings VALUES (1, 2, 3, '2023-04-29');`

(B) `INSERT INTO Bookings VALUES (2, 2, 2, '2023-04-29');`

(C) `INSERT INTO Bookings VALUES (3, 2, 1, '2023-04-29');`

(D) `INSERT INTO Bookings VALUES (1, 1, 1, '2023-04-29');`

(E)
```
INSERT INTO Bookings VALUES (1, 2, 1, '2023-04-29');
INSERT INTO Bookings VALUES (2, 1, 1, '2023-04-29');
```

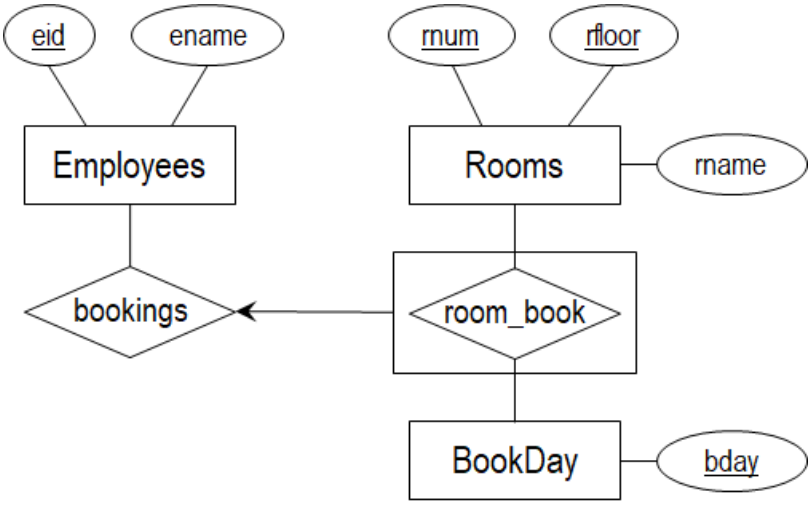
(F)
```
INSERT INTO Bookings VALUES (1, 2, 1, '2023-04-29');
INSERT INTO Bookings VALUES (2, 1, 2, '2023-04-29');
```

(G)
```
INSERT INTO Bookings VALUES (2, 2, 2, '2023-04-29');
INSERT INTO Bookings VALUES (2, 2, 1, '2023-04-29');
```

(H)
```
INSERT INTO Bookings VALUES (1, 2, 3, '2023-04-29');
INSERT INTO Bookings VALUES (3, 2, 1, '2023-04-29');
```

(I) *None of the above*

1.3 (5 points) **ER Diagram.** Now consider the following ER diagram representation.



Select ALL constraints that are **NOT** enforced by the ER diagram.

(A) Employees are identified by their employee id.
(B) Rooms are identified by floor number and room number.
(C) No two different employees can book the same room on the same day.
(D) It is possible that a room is not booked by any employee.
(E) No two different rooms can be booked by the same employee on the same day.
(F) It is possible that an employee does not book any room.
(G) *None of the above*

1.4 (5 points) **SQL Query.** Consider the following instance of table `Bookings`.

`Bookings`

bfloor	bnum	beid	bday
1	1	1	2023-04-29
1	3	3	2023-04-29
1	4	2	2023-04-29
1	2	1	2023-04-29
2	2	2	2023-04-29
1	1	3	2023-04-28
2	1	3	2023-04-28
2	2	3	2023-04-28
2	3	2	2023-04-28
1	2	1	2023-04-27
2	1	2	2023-04-27
1	1	3	2023-04-27

Select ALL the rows that will appear in the result if we run the following query.

```
SELECT bfloor, COUNT(DISTINCT beid)
FROM   Bookings
WHERE  bday > '2023-04-27'
GROUP BY bday, bfloor;
```

(A) $\boxed{1}\boxed{1}$ (C) $\boxed{1}\boxed{3}$ (E) $\boxed{2}\boxed{1}$ (G) $\boxed{2}\boxed{3}$

(B) $\boxed{1}\boxed{2}$ (D) $\boxed{1}\boxed{4}$ (F) $\boxed{2}\boxed{2}$ (H) $\boxed{2}\boxed{4}$

(I) *None of the above*

2 Stored Procedures

Consider the following `Scores` table. The data types of `sid` and `name` are `TEXT` while the data type of `score` is `INT`.

`Scores`

sid	name	score
s1	Alice	71
s2	Bob	78
s3	Cathy	84
s4	David	89
s5	Eric	93

2.1 (6 points) Consider the `test_func` function below.

```
CREATE OR REPLACE FUNCTION test_func()
RETURNS INT AS $func$
DECLARE
  curs CURSOR for (SELECT * FROM Scores ORDER BY score desc);
  r RECORD;
  score1 INT;
  score2 INT;
  gap1 INT;
  gap2 INT;
BEGIN
  score1 := -1;
  gap2 := -1;

  OPEN curs;
  LOOP
    FETCH curs INTO r;
    EXIT WHEN NOT FOUND;

    IF (score1 = -1) THEN
      score1 := r.score;
    ELSE
      gap1 := score1 - r.score;
      IF (gap1 > gap2) THEN
        gap2 := gap1;
        score2 := score1;
      END IF;
      score1 := r.score;
    END IF;
  END LOOP;
  CLOSE curs;
  // continue on the next page
```

```
  IF (gap2 > -1) THEN
    RETURN score2;
  ELSE
    RETURN -1;
  END IF;
END;
$func$ LANGUAGE plpgsql;
```

Suppose that we execute the following query. What will be the query result?

```
SELECT * FROM test_func();
```

(A) 4 (C) 6 (E) 8 (G) 78 (I) 89

(B) 5 (D) 7 (F) 71 (H) 84 (J) 93

(K) *None of the above*

2.2 (6 points) Consider the `scores_check_func` function below.

```
CREATE OR REPLACE FUNCTION scores_check_func()
RETURNS TRIGGER AS $func$
DECLARE
  fail_cnt INT;
BEGIN
  SELECT COUNT(*) INTO fail_cnt
  FROM Scores
  where score < 60;

  IF (fail_cnt > 3) THEN
    RAISE EXCEPTION 'You failed too many students!';
  END IF;

  RETURN NEW;
END;
$func$ LANGUAGE plpgsql;
```

Suppose that we create a trigger `scores_check_trigger` based on the above function, and the execute the following transaction.

```
BEGIN
  INSERT INTO Scores values ('s6', 'Fred', 59);
  INSERT INTO Scores values ('s7', 'Gigi', 58);
  INSERT INTO Scores values ('s8', 'Helen', 57);
  INSERT INTO Scores values ('s9', 'Ivan', 56);
COMMIT;
```

Assumes that `Scores` has **nine (9)** tuples after the transaction is executed. In that case, which of the following definitions of `scores_check_trigger` are possible?

(A)
```
CREATE TRIGGER scores_check_trigger
BEFORE INSERT ON Scores
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(B)
```
CREATE TRIGGER scores_check_trigger
AFTER INSERT ON Scores
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(C)
```
CREATE CONSTRAINT TRIGGER scores_check_trigger
BEFORE INSERT ON Scores
DEFERRABLE INITIALLY IMMEDIATE
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(D)
```
CREATE CONSTRAINT TRIGGER scores_check_trigger
AFTER INSERT ON Scores
DEFERRABLE INITIALLY IMMEDIATE
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(E)
```
CREATE CONSTRAINT TRIGGER scores_check_trigger
BEFORE INSERT ON Scores
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(F)
```
CREATE CONSTRAINT TRIGGER scores_check_trigger
AFTER INSERT ON Scores
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

(G) *None of the above*

3 Functional Dependencies

3.1 (6 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies

$$\Sigma = \{\{F\} \to \{B\}, \{D\} \to \{A\}, \{C\} \to \{F\}, \{E, F\} \to \{C\}, \\ \{A, F\} \to \{E\}, \{A, B\} \to \{F\}\}$$

Suppose that we decompose R into $R1(A, B, C, D)$ and $R2(C, D, E, F)$.

(a) (2 points) Is this a lossless-join decomposition?

(A) True (B) False

(b) (4 points) Is this a dependency-preserving decomposition? If not, please select all FDs that are **NOT** preserved.

(A) $\{F\} \to \{B\}$
(B) $\{D\} \to \{A\}$
(C) $\{E, F\} \to \{C\}$
(D) $\{A, F\} \to \{E\}$
(E) $\{A, F\} \to \{B\}$
(F) $\{A, B\} \to \{F\}$
(G) *None of the above*

3.2 (7 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies

$$\Sigma = \{\{D\} \to \{E\}, \{C\} \to \{A\}, \{C, F\} \to \{E\}, \{A, F\} \to \{C\}, \\ \{C, E\} \to \{D\}, \{B, E\} \to \{A\}, \{A, D\} \to \{B\}, \{B, C\} \to \{F\}\}$$

Suppose that we decompose R into $R1(A, B, C, D)$ and $R2(C, D, E, F)$.

(a) (2 points) Is this a lossless-join decomposition?

(A) True (B) False

(b) (5 points) Is this a dependency-preserving decomposition? If not, please select all FDs that are **NOT** preserved.

(A) $\{D\} \to \{E\}$
(B) $\{C\} \to \{A\}$
(C) $\{C, F\} \to \{E\}$
(D) $\{A, F\} \to \{C\}$
(E) $\{C, E\} \to \{D\}$
(F) $\{B, E\} \to \{A\}$
(G) $\{A, D\} \to \{B\}$
(H) $\{B, C\} \to \{F\}$
(I) *None of the above*

4 Normal Forms

4.1 (15 points) Consider the following relation $R(A, B, C, D, E)$ with the following set of functional dependencies

$$\Sigma = \{\{C\} \to \{E\}, \{B\} \to \{E\}, \{B\} \to \{A\}, \{C, E\} \to \{D\}, \\ \{B, E\} \to \{C\}, \{A, D\} \to \{B\}, \{A, D\} \to \{C\}\}$$

(a) (4 points) Write ALL the keys of R with respect to Σ. (*choices in Examplify*)

(b) (3 points) Is R in BCNF with respect to Σ? If not, write down all FDs in Σ that violates the BCNF requirements. (*choices in Examplify*)

(c) (5 points) Is R in BCNF with respect to Σ? If not, apply the BCNF decomposition algorithm (*introduced in CS2102 lectures*) on R, and select the relations in the final result of your BCNF decomposition. (*choices in Examplify*)

(d) (3 points) Is R in 3NF with respect to Σ? If not, write down all FDs in Σ that violates the 3NF requirements. (*choices in Examplify*)

4.2 (5 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies

$$\Sigma = \{\{F\} \to \{D\}, \{B, F\} \to \{C\}, \{B, C, D, F\} \to \{E\}, \{D\} \to \{B\}, \\ \{C, D\} \to \{A\}, \{B, D\} \to \{C\}, \{A, D\} \to \{C\}, \{B\} \to \{A\}\}$$

Derive a minimal basis of Σ. (*choices in Examplify*)

4.3 (5 points) Consider an arbitrary relation $R(A, B, C, D, E, F)$. Suppose that R is not in 3NF with respect to some unknown set of functional dependencies Σ. In that case, what is the maximum number of keys that R can have? (*choices in Examplify*)