CS2102: Database Systems (AY2020-2021 – Sem 1)

Final Exam

Instructions

1. Please read **ALL** instructions carefully.
2. This assessment contains **14** questions:
   (a) There are 10 Fill-in-the-Blank Question (FITB)
   (b) There is 2 Long Answer Question
   (c) There is 2 Hot Spot Questions
3. All the assessment is be done using Examplify:
   (a) This is a secure assessment:
      i. Your Internet connection will be blocked.
      ii. You will not be able to access any other software besides Examplify.
   (b) This is a closed-book exam *(with one A4 page cheatsheet)*
4. Use the question number shown on Examplify when asking.
   - If the answer is clear from the question pdf/Examplify, we will reply with "No Comment".
5. Failure to follow each of the instructions above may result in deduction of your marks.

Good Luck!

1 Relational Algebra and SQL

Consider the following schema:

- `Students(matric, sname)`
- `Workings(pid, matric, since)`
- `Projects(pid, pname)`
- `Categories(pid, cname)`

1.1 Complete the relational algebra expression below to find all pairs of different students' matric number `(m1, m2)` that are working on the same project, excluding students not working on any project.

$\pi_{(\_\_A\_\_)}(\sigma_{(\_\_B\_\_\wedge\_\_C\_\_)}(\rho_{(w_1(p_1,m_1,s_1)}(\texttt{Workings}) \times \rho_{w_2(p_2,m_2,s_2)}(\texttt{Workings})))$

__A__ = [ ]
__B__ = [ ]
__C__ = [ ]

1.2 Complete the relational algebra expression below to find the oldest project `pid` in the database.

$\pi_{(\_\_A\_\_)}(\pi_{\_\_B\_\_,\_\_C\_\_}(\texttt{Workings}) - \pi_{p_2,s_2}(\sigma_{\_\_D\_\_>\_\_E\_\_}(\rho_{w_1(p_1,m_1,s_1)}(\texttt{Workings}) \times \rho_{w_2(p_2,m_2,s_2)}(\texttt{Workings}))))$

__A__ = [ ]
__B__ = [ ]
__C__ = [ ]
__D__ = [ ]
__E__ = [ ]

1.3 Complete the SQL create table statement below to create the schema above. The first table (*i.e.,* ***`Students`*** *table)* has been filled in for you.

```
CREATE TABLE Students (
  matric  VARCHAR(9) PRIMARY KEY,
  sname   VARCHAR(50)
);
CREATE TABLE Projects (
  __A__  VARCHAR(9) __B__,
  __C__  VARCHAR(50)
);
CREATE TABLE Workings (
  __D__ VARCHAR(9) REFERENCES __E__,
  __F__ VARCHAR(9) REFERENCES __G__,
  __H__ DATE,
  PRIMARY KEY (pid, matric)
);
```

```
CREATE TABLE Categories (
  __I__ VARCHAR(9) REFERENCES __J__,
  __K__ VARCHAR(9)
  PRIMARY KEY (pid, cname)
);
```

__A__ = _____
__B__ = _____
__C__ = _____
__D__ = _____
__E__ = _____
__F__ = _____
__G__ = _____
__H__ = _____
__I__ = _____
__J__ = _____
__K__ = _____

1.4 Complete the SQL query below to find all pair of distinct projects' `pid (p1, p2)` such that the two projects have exactly the same set of categories.

```
SELECT P1.pid, P2.pid
FROM   Projects P1, Projects P2
WHERE  __A__
  AND  ( SELECT COUNT(*)
         FROM ( SELECT __B__ FROM Categories C0 WHERE __C__
                __D__
                SELECT __B__ FROM Categories c0 WHERE __E__ ) AS T1 )
       =
       ( SELECT COUNT(*)
         FROM ( SELECT __B__ FROM Categories C0 WHERE __C__
                __F__
                SELECT __B__ FROM Categories c0 WHERE __E__ ) AS T2 );
```

Each blanks must be *short* answers, they cannot be nested queries.

__A__ = _____
__B__ = _____
__C__ = _____
__D__ = _____
__E__ = _____
__F__ = _____

1.5 Consider the following SQL query:

```
SELECT *
FROM   Students NATURAL JOIN Workings
               NATURAL JOIN Projects
               NATURAL JOIN Categories;
```

Say it produces the following result:

matric	sname	pid	since	cname
A0001	AA	P01	2002	CA
A0001	AA	P01	2002	CB
A0001	AA	P02	2004	CB
A0002	BB	P01	2003	CA
A0002	BB	P01	2003	CB
A0003	CC	P03	2004	CA
A0003	CC	P03	2004	CC
A0003	CC	P03	2004	CD
A0004	AA	P03	2004	CA
A0004	AA	P03	2004	CC
A0004	AA	P03	2004	CD

Now consider another the query from the previous question. It produces the following result:

pid	pid
P01	P04
P04	P01
P03	P05
P05	P03

What will be the result of the following SQL query?

```
SELECT pid FROM Categories
EXCEPT ALL
SELECT DISTINCT pid FROM Categories WHERE cname != 'CA';
```

Note: The result may be fewer than 7 rows.

pid

----

2 Schema Refinement

Consider the schema $R(A, B, C, D, E)$ with $\Sigma = \{$

$\{B, D, E\} \rightarrow \{A, E\}$

$\{A, D\} \rightarrow \{C\}$

$\{C, D\} \rightarrow \{E\}$

$\{B, E\} \rightarrow \{A, D\}$

$\}$

2.1 Write **four (4)** *completely non-trivial* functional dependencies that are *implied* by Σ but not in Σ. If there are fewer than 4 answers, write the answer as "-" (*without the quote*) instead of leaving it blank.

`FD #1 =` ____
`FD #2 =` ____
`FD #3 =` ____
`FD #4 =` ____

2.2 Write **four (4)** *superkeys* of R with *at most* **three (3)** attributes. If there are fewer than 4 answers, write the answer as "-" (*without the quote*) instead of leaving it blank.

`Superkey #1 =` ____
`Superkey #2 =` ____
`Superkey #3 =` ____
`Superkey #4 =` ____

2.3 Write **four (4)** *keys* of R with *at most* **three (3)** attributes. If there are fewer than 4 answers, write the answer as "-" (*without the quote*) instead of leaving it blank.

`Key #1 =` ____
`Key #2 =` ____
`Key #3 =` ____
`Key #4 =` ____

2.4 Write one *possible* lossless BCNF decomposition of R into **exactly three (3)** fragments such that each fragment has exactly **three (3)** attributes.

R_1(____, ____, ____)
R_2(____, ____, ____)
R_3(____, ____, ____)

2.5 Is your decomposition in previous question a *dependency-preserving* decomposition? Simply write True/False, there is no need for justification.

2.6 Write one possible *minimal cover* of Σ.

2.7 Write one possible lossless decomposition of R that is dependency preserving and in either BCNF or 3NF. Additionally, it must have exactly **three (3)** fragments.

$R_1 =$
$R_2 =$
$R_3 =$

3 Armstrong Axioms

We know that Armstrong axioms are both *sound* and *complete*. Consider the **augmentation** rule in Armstrong axioms:

$$\alpha \to \beta \Rightarrow \alpha \cup \gamma \to \beta \cup \gamma$$

Our aim now is to show that the *"reverse"* of the augmentation rule is not sound. By the reverse, we meant the following rule:

$$\alpha \cup \gamma \to \beta \cup \gamma \Rightarrow \alpha \to \beta$$

To use the "reverse" augmentation rule:

1. $\{A, C\} \to \{B, C\}$ [Given]
2. $\{A\} \to \{B\}$ [Remove $\{C\}$ from (1)]

3.1 Show that the reverse of augmentation rule of Armstrong axioms is not sound by showing *without any given starting functional dependency* that we can always imply $\{A\} \to \{B\}$. Your proof should not exceed **three (3)** lines.

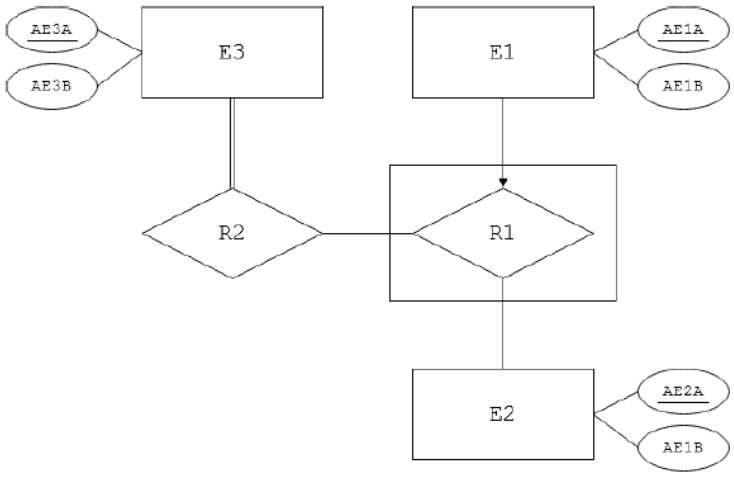
4 ER Diagram

For each ER diagram in this question, find the mistake by marking it on the ER diagram:

- If the mistake is on a line, make sure you mark on the line ***outside*** of any box/diamond/oval.
- If the mistake is on an arrow, make sure you mark on the arrow and not on the line.
- If the mistake is on the box/diamond/oval, make sure you mark ***inside*** the box/-diamond/oval *without* touching any other lines.

Note: We disallow primary key (*i.e.,* *underline*) on relation for this semester.

4.1 Find the mistake in the ER diagram below:



4.2 Find the mistake in the ER diagram below: