CS2102: Database Systems (AY2122 – Sem 1)

Midterm Exam

Date: Monday, 3 October 2022, Time: 18:30–19:30

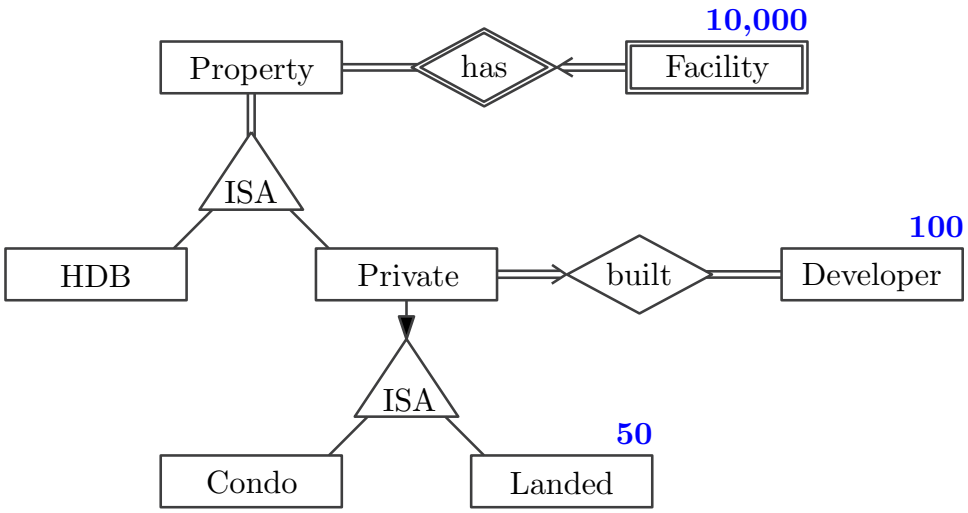
Submission Instructions

1. Please read **ALL** instructions carefully.
2. All the assessment is be done using Examplify; the assessment contains
   (a) Fill in Blank Questions: 1.1
   (b) ...
   (c) Essay Questions: 2.1–2.5 (5 SQL queries)
   (d) Your Internet connection will be blocked
   (e) This is an open-book exam
3. This assessment starts at 18:30 and ends at 19:30.
   - Submit your answers by 19:30
   - No additional time will be given to submit.
4. For the SQL questions, there are additional instructions below; in a nutshell:
   - Use an IDE or text editor to write your queries and test them using `psql`
   - Prepare your final answer before submitting it to Examplify according to the instructions in Section 2
   - Allocate sufficient time to prepare your final answers and to copy-&-paste them into Examplify.
5. Failure to follow each of the instructions above may result in deduction of your marks.

Good Luck!

1 ER model & Relational Algebra (10 Marks)

1.1 Cardinalities (2 Marks). Given is the Entity-Relationship Diagram below; it does not show any attributes as they are of no relevance for this task. The the numbers **BLUE** indicates the number of instances of the entity sets `Facility`, `Developer`, and `Landed`.



Complete the table below by identifying the **minimum** and **maximum** possible numbers of instances for the remaining entity sets.

	minimum	**maximum**
`Property`		
`HDB`		
`Private`		
`Condo`		

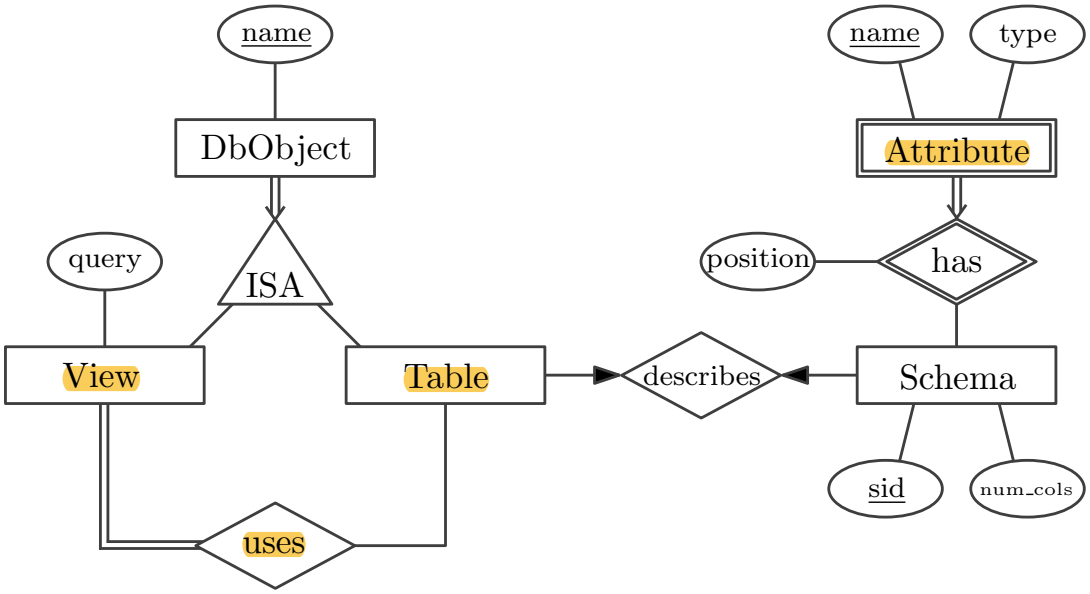
Solution:

	minimum	**maximum**
`Property`	100	10,000
`HDB`	0	10,000
`Private`	100	10,000
`Condo`	50	9,500 9,950

Explanation

- The 10,000 instance of **Facility** and the participation constraint from `Property` specify the upper bound for **Property**: 10,000
- The 100 instances of `Developer` and the 2 constraints around `built` specify the lower bound for `Private`: 100
- The upper bound of `Property` specifies the upper bound for `Private`: 10,000 (since each property can be both HDB and private: e.g., executive condos are hybrid)
- The lower bound of `Private` specifies the lower bound of `Property`: 100
- The upper bound of `Private` minus the 50 instances of `Landed` specify the upper bound of `Condo`: 9,950
- The lower bound of `Private` minus the 50 instances of `Landed` specify the upper bound of `Condo`: 50
- The upper bound of `Property` specifies the upper bound of `HDB`: 10,000 (again, a property can be both an HDB and a private property)
- The lower bound of `HDB` is 0

1.2 Constraints (4 Marks). A RDBMS must keep track of all database of users: tables (with their schema), the views, etc. DBMS stores is information in special table, the so called *system catalog*. The ERD below models an over-simplified system catalog. A database object in a RDBMS is any data structure used to either store or reference data. We only learned about 2 types of objects: tables and views.



Based on this ER diagram, the following database schema has been created:

- `DbObject(name)`
- `View(name, query)`
- `Table(name, sid)`
- `Schema(sid, num_cols)`
- `Attribute(sid, name, type, position)`
- `Uses(vname, tname)`

The foreign key constraints are as follows:

- `View.name→DbObject.name`
- `Table.name→DbObject.name`
- `Table.sid→Schema.sid`
- `Attribute.sid→Schema.sid`
- `Uses.vname→View.name`
- `Uses.tname→Table.name`

Solution:

Nr	Constraint	ERD	Schema
1	All instances of `View` and `Table` are uniquely identified.	**C**	**C**
2	Each `DbObject` must either be a `View` or a `Table` and cannot be both.	**C**	
3	Each `Table` is associated with exactly one `Schema` and vice versa.	**C**	**C**
4	The `name` of an `Attributes` must be unique for a given `Schema`.		**C**
5	The value of `num_cols` for a schema mast match the number of attributes for that schema.		
6	A `View` uses 1 or more `Table` instances.	**C**	
7	If a schema has N attributes, all `position` values have to describe a sequence $1, 2, ..., N$		
8	The `query` of a `View` must always be specified and cannot be an empty string.		C

Given these 8 constraints, answer the following 2 questions:

(a) Which of constraints above are **NOT CAPTURED** by the ER diagram? Select all that apply.

Solution:

- The value of `num_cols` for a schema mast match the number of attributes for that schema.
- If a schema has N attributes, all `position` values have to describe a sequence $1, 2, ..., N$
- The `query` of a `View` must always be specified and cannot be an empty string. (We cannot models something like NOT NULL with the ERD.)

(b) Which constraints **CANNOT BE CAPTURED** when converting the ER Diagram into the database schema without using `TRIGGER`? Select all that apply.

Solution:

- Each `DbObject` must either be a `View` or a `Table` and cannot be both. (We cannot check of there's a database object that is not a view or table; we also cannot check that a database object is both view and table)
- The value of `num_cols` for a schema mast match the number of attributes for that schema. (Just not possible without triggers.)
- A `View` uses 1 or more `Table` instances. (Table `Uses` can be empty)
- If a schema has N attributes, all `position` values have to describe a sequence $1, 2, ..., N$. (Just not possible without triggers.)
-

2 Formulating SQL Queries (10 Marks)

In this part of the exam your task is to formulate queries in SQL. We use the same database that we introduced for Assignment 1. This means you can directly run and test your queries using `psql`. If needed, we provide the ER diagram and the `CREATE TABLE` statements for the database schema in the Appendix. But you should be very familiar with the database by now.

Instructions. To ensure successful submission and grading of your SQL queries adhere to the following instructions:

- For each question, you are to write a **single CREATE OR REPLACE VIEW SQL statement** to answer the question. You **MUST** use the view schema provided for each question without changing any part of the view schema (i.e., view name, column names, or the order of the columns). For example, the provided view schema for the question is `"q2 (area)"`, and if your answer to this question is the query `"SELECT name FROM areas;"`, then you must enter your answer in the answer box as follows:

```
CREATE OR REPLACE VIEW q2 (area) AS
SELECT name
FROM areas
;
```

- Each `CREATE OR REPLACE VIEW` statement must terminate with a semicolon.
- Each answer must be a syntactically valid SQL query. Test with `psql`!
- Each question must be answered independently of other questions, i.e., the answer for a question must not refer to any other view that is created for another question.
- Your answers must not use SQL constructs that are not covered in Lectures 1-6.
- Your answers must be executable on PostgreSQL. Test with `psql`!
- You must not enter any extraneous text for your answer (e.g., "My answer is: ...), or enter multiple answers (i.e., submit multiple SQL statements). Your answer should not contain any comments.

Recommendations. We recommend the following steps to answer each question:

- Write the answer query using your IDE or text editor of choice, and test it on your machine with `psql`.
- Once you are happy with your query, add the `CREATE OR REPLACE VIEW` statement for the corresponding question in front of the query. You can test the complete statement again with `psql` and create the view.
- Copy the complete `CREATE OR REPLACE VIEW` statement into the answer field in Examplify (don't forget to the semicolon at the end!)

2.1 Query 1 (1 Mark): Find most common height of HDB blocks across all of Singapore! Here, the height of an HDB block is represented by the number of floors the block has (`'max_floor'`). Include the number of blocks of that height in the result.

```
CREATE OR REPLACE VIEW q1 (height, num_blocks) AS

;
```

Solution:

```
CREATE OR REPLACE VIEW q1 (height, num_blocks) AS
SELECT max_floor AS height, COUNT(*) AS num_blocks
FROM hdb_blocks
GROUP BY max_floor
ORDER BY num_blocks DESC
LIMIT 1
;
```

2.2 Query 2 (2 Marks): Find the names of all areas in the `'central'` region that do not contain any MRT station!

```
CREATE OR REPLACE VIEW q2 (area) AS

;
```

Solution:

```
CREATE OR REPLACE VIEW q2 (area) AS
SELECT a.name
FROM areas a
WHERE a.region = 'central'
AND NOT EXISTS (SELECT 1
                FROM mrt_stations m, subzones s1
                WHERE m.subzone = s1.name
                AND s1.area = a.name)
;
```

2.3 Query 3 (2 Marks): Find the name of the top-3 areas with the most MRT stops (not stations)! Include the number of stops in your result set!

```
CREATE OR REPLACE VIEW q3 (area, num_stops) AS

;
```

Solution:

```
CREATE OR REPLACE VIEW q3 (area, num_stops) AS
SELECT z.area, COUNT(*) AS num_stops
FROM subzones z, mrt_stations m, mrt_stops s
WHERE z.name = m.subzone
AND m.name = s.station
GROUP BY z.area
ORDER BY num_stops DESC
LIMIT 3
;
```

2.4 Query 4 (2 Marks): Find the names of MRT stations along the East-West Line (`'ew'`) from which you can reach any other line in just 1 stop. For example, from `'aljunied'` (`'ew9'`) you can reach `'paya lebar'` (`'cc9'`), so `'aljunied'` should be in your result set. In contrast, `'kallang'` (`'ew10'`) only connects to stations of the `'ew'` line, so it should not be in your result set.

```
CREATE OR REPLACE VIEW q4 (mrt_station) AS

;
```

Solution:

```
CREATE OR REPLACE VIEW q4 (mrt_station) AS
SELECT s1.station
FROM mrt_stops s1, mrt_connections c, mrt_stops s2
WHERE s1.code = c.from_code
AND s2.code = c.to_code
AND s1.line = 'ew'
AND s2.line <> 'ew'
GROUP BY s1.station
;
```

2.5 Query 5 (3 Marks): Find all areas that are served by 3 or more different MRT lines – that is, all areas that contain any number of MRT stations of at least 3 different lines. So if an area contains 5 MRT stations but from only 1 or 2 lines, this area should not be part of your result. Include the area names and the number of lines that serve each area in your result and sort by the number of lines in descending order.

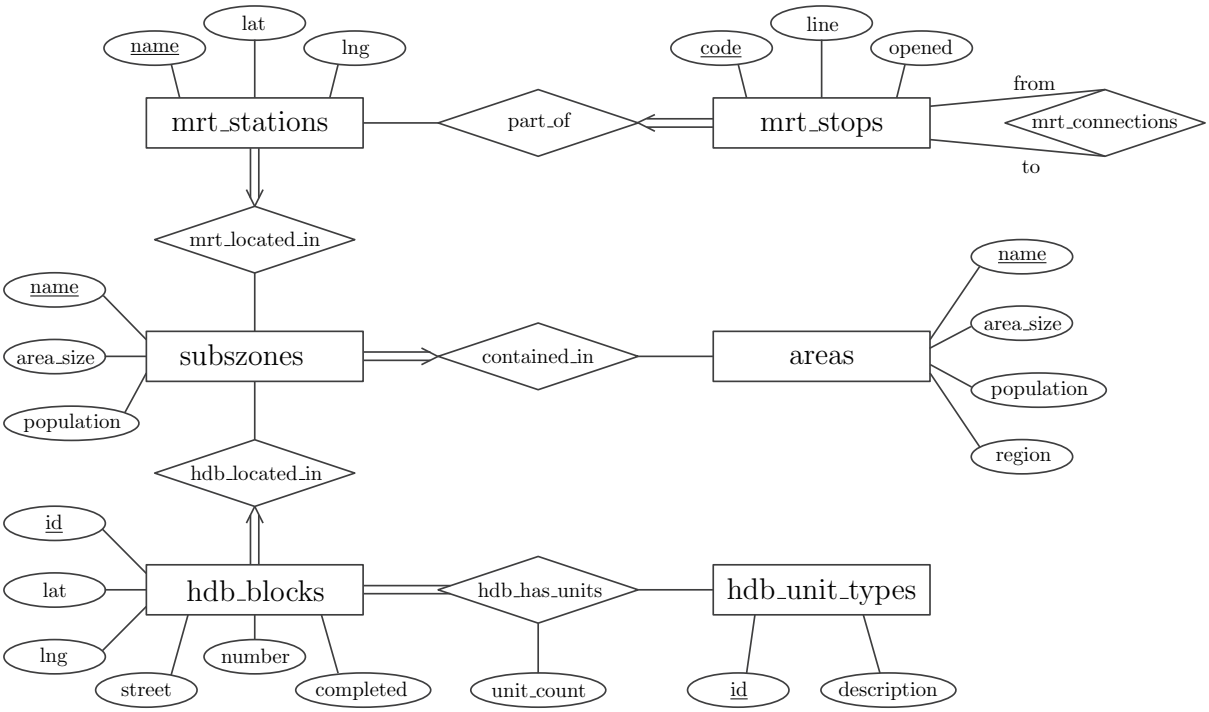
```
CREATE OR REPLACE VIEW q5 (area, num_lines) AS

;
```

Solution:

```
CREATE OR REPLACE VIEW q5 (area, num_lines) AS
SELECT tab.area, COUNT(*) AS num_lines
FROM (SELECT s.area, h.line
      FROM mrt_stops h, mrt_stations m, subzones s
      WHERE h.station = m.name
      AND m.subzone = s.name
      GROUP BY s.area, h.line) tab
GROUP BY tab.area
HAVING COUNT(*) >= 3
ORDER BY num_lines DESC
;
```

A Appendix

A.1 ER Diagram of Database for Task 2



A.2 CREATE TABLE statements of Database for Task 2

```
CREATE TABLE areas (
 name TEXT PRIMARY KEY,
 area_size NUMERIC CHECK (area_size >= 0),
 population INTEGER CHECK (population >= 0),
 region TEXT NOT NULL
);

CREATE TABLE subzones (
 name TEXT PRIMARY KEY,
 area_size NUMERIC CHECK (area_size >= 0),
 population INTEGER CHECK (population >= 0),
 area TEXT NOT NULL,
 FOREIGN KEY (area) REFERENCES areas (name)
);

CREATE TABLE hdb_blocks (
 id INTEGER PRIMARY KEY,
 number TEXT NOT NULL,
 street TEXT NOT NULL,
 postal_code INTEGER NOT NULL CHECK (postal_code > 0),
 completed INTEGER NOT NULL,
 max_floor INTEGER NOT NULL CHECK (max_floor >= 0),
 lat NUMERIC NOT NULL,
 lng NUMERIC NOT NULL,
 subzone TEXT NOT NULL,
```

```
 FOREIGN KEY (subzone) REFERENCES subzones (name)
);

CREATE TABLE hdb_has_units (
 block_id INTEGER NOT NULL,
 unit_type TEXT NOT NULL,
 unit_count INTEGER NOT NULL CHECK (unit_count >= 0),
 PRIMARY KEY (block_id, unit_type),
 FOREIGN KEY (block_id) REFERENCES hdb_blocks (id)
);

CREATE TABLE mrt_stations (
 name TEXT PRIMARY KEY,
 lat NUMERIC NOT NULL,
 lng NUMERIC NOT NULL,
 subzone TEXT NOT NULL,
 FOREIGN KEY (subzone) REFERENCES subzones (name)
);

CREATE TABLE mrt_stops (
 code CHAR(4) PRIMARY KEY,
 line CHAR(2) NOT NULL,
 opened INTEGER NOT NULL CHECK (opened >= 0),
 station TEXT NOT NULL,
 FOREIGN KEY (station) REFERENCES mrt_stations (name)
);

CREATE TABLE mrt_connections (
 from_code CHAR(4),
 to_code CHAR(4),
 PRIMARY KEY (from_code, to_code),
 FOREIGN KEY (from_code) REFERENCES mrt_stops (code),
 FOREIGN KEY (to_code) REFERENCES mrt_stops (code)
);

CREATE TABLE hdb_unit_types (
 id CHAR(10) PRIMARY KEY,
 description TEXT NOT NULL
);
```