UNIVERSITY OF VICTORIA

PRACTICE EXAMINATION #3

(SQL & Transactions)

CSC 370: Database Systems

(0 hours, 45 minutes)

This examination consists of ten equally-weighted multiple choice questions. You should record your solutions in the provided bubble sheet. Each question has a single best solution; if you record more than one solution for the same question, you will receive a score of zero on that question. If you answer x *questions correctly, then your grade on the exam will be* x*/10, i.e., you must answer at least five questions correctly to pass. This exam is closed-book: you are welcome to bring with you empty pages and a single-sided A4/US letter note sheet, but you cannot bring other notes nor electronic devices to your desk. Please confirm immediately after the exam starts that you have all ? pages and ten questions.*

Sections: A01, A02, A03

CRN's: 10874, 10875, 14303

Instructors: Mr. Yichun Zhao + Dr. Sean Chester

Data		
x	**y**	**z**
1	'foo'	'9 July'
2	'bar'	'13 Aug'
3	'foobar'	NULL

Question 1

You are provided with the table "Data", in which *x* is the primary key. Which of the three provided queries could plausibly add a new tuple to the table?

(a) INSERT INTO Data(y) VALUES(NULL);

- This is valid if x has an auto increment specifier or a default value that is not 1, 2, or 3.

(b) INSERT INTO Data VALUES(1, NULL, NULL);

- This will be rejected due to the duplicate value on the primary key attribute

(c) INSERT INTO Data(y,x,z) VALUES(NULL, NULL, NULL)

- The primary key attribute cannot be null

Question 2

You are provided with the table "Data", below in which *x* is the primary key.Which of the three provided queries would be the best choice if you wanted to remove all tuples from that table?

(a) DELETE * FROM Data;

- This is syntactically invalid

(b) DELETE FROM Data WHERE y LIKE 'foo%' OR x <3;

(c) DROP TABLE Data;

- This would delete all the metadata about the table, not just the tuples

Question 3

Which of the following SQL statements creates a database constraint?

(a) ALTER TABLE R MODIFY x INT CHECK(x > 0);

(b) DELETE FROM S WHERE S.x <= 0;

(c) SELECT * FROM T WHERE T.x > 0;

(d) All of the above

Question 4

Which of the scenarios below best describes an example of the *Consistency* property being violated?

(a) One transaction runs twice and produces a different result each time

(b) A transaction commits a duplicate primary key value

(c) A transaction causes non-deterministic (i.e., random) behaviour

Question 5

A table and a SQL query are specified below. Indicate which of the three specified tuples would be returned by this query if executed on this table using MySQL 11 with default settings.

R		
x	**y**	**z**
1	1	'Alice'
1	2	'Eve'
1	3	'Eve'
2	4	'Carol'
3	5	'Eve'
4	6	'Bob'

```
SELECT x, COUNT(z)
FROM R
GROUP BY x;
```

(a) (1,2)

- There is no DISTINCT keyword applied to z

(b) (1,3)

(c) (1,4)

- 'Bob' and 'Carol' are not among the tuples in the equivalence class defined by x=1.

Question 6

A pair of tables and a SQL query are specified below. Indicate which of the three specified tuples would ***not*** be returned by this query if executed on these tables using MySQL 11 with default settings.

R		
x	**y**	**z**
1	1	'Alice'
1	2	'Eve'
1	3	'Eve'
2	4	'Carol'
3	5	'Eve'
4	6	'Bob'

S		
t	**u**	**v**
1	NULL	'a'
2	4	'b'
3	2	'a'
4	5	'b'
5	3	'a'
6	2	'b'

```
SELECT *
FROM R
        JOIN S ON (R.x = S.u);
```

(a) (1,1,'Alice',1,NULL,'a')

- inner join always throws away tuples with NULL values

(b) (2,4,'Carol',2,'b')

- Only the NATURAL JOIN projects away the duplicate copy of the join attribute(s)

(c) Neither of these tuples would be returned by this query

(d) Both of these tuples would be returned by this query

Question 7

Which of the behaviours written below best describe the effect of this trigger?

```
DELIMITER //
CREATE TRIGGER t
AFTER INSERT ON R
FOR EACH ROW
BEGIN
        IF NEW.x > NEW.y THEN
                SET NEW.x = NEW.y;
        END IF;
END; //
DELIMITER ;
```

(a) This trigger has no effect / is invalid

- it is invalid to update NEW in an AFTER trigger

(b) If a tuple is added to R then *all* tuples that have a larger x value than y value will be updated to have equal x and y values

- An insert trigger only affects the tuples that are being inserted (unless there is a full query in the trigger)

(c) It ensures that no tuples can be added to R in which the value for the x attribute is larger than the value for the y attribute

- this is true if this is a BEFORE trigger

Question 8

You have tables corresponding to the relations below and you would like to retrieve all states that have fewer than 30 counties.

Which of the three proposed queries is the simplest one of the three that is correct?

County(county_id, state_id, name)

State(state_id, name)

(a) SELECT *
FROM County
NATURAL JOIN State
GROUP BY state_id
HAVING COUNT(*) <30;

(b) SELECT *
FROM State
WHERE state_id NOT IN (
SELECT state_id
FROM County
WHERE COUNT(*) <30
);

(c) SELECT *
FROM State
NATURAL JOIN (
SELECT state, COUNT(*) As num
FROM County
GROUP BY state
) AS Subquery
WHERE Subquery.num <30;

Question 9

You have a table Enrollments described by the relation below.

Enrollments(student_id, course_id, semester, grade)

What is the effect on the Enrollments table of executing the following two queries in sequence?

```
CREATE VIEW `SummerEnrollments` AS (
        SELECT *
        FROM Enrollments
        WHERE semester = '202205'
);
DELETE
FROM SummerEnrollments
WHERE student_id = 123456;
```

(a) This query will remove the student with id=123456 from all Summer 2022 courses in which they are enrolled

(b) This query will remove the student with id=123456 from all courses in which they are enrolled, irrespective of the semester

(c) This query will only remove the student with id=123456 from the view;
if we recreate the view from scratch, the student's summer enrollments will be restored

Question 10

You have a MySQL 11 database in which the following relations have been created as tables.

The tables currently have *no* indexes. Which of the three queries below would ***not*** create a new index?

Employee(employee_id, department_id, job_title)

Department(department_id, name, manager)

(a) ALTER TABLE Employee ADD INDEX (employee_id, department_id)

- This is syntactically invalid. We use a CREATE INDEX statement, not an ADD INDEX specifier.

(b) ALTER TABLE Employee ADD FOREIGN KEY (department_id) REFERENCES Department(id);

- The creation of a foreign key automatically constructs an index in MySQL 11

(c) ALTER TABLE Employee ADD UNIQUE KEY(employee_id, department_id);

- The creation of a unique key automatically constructs an index in MySQL 11