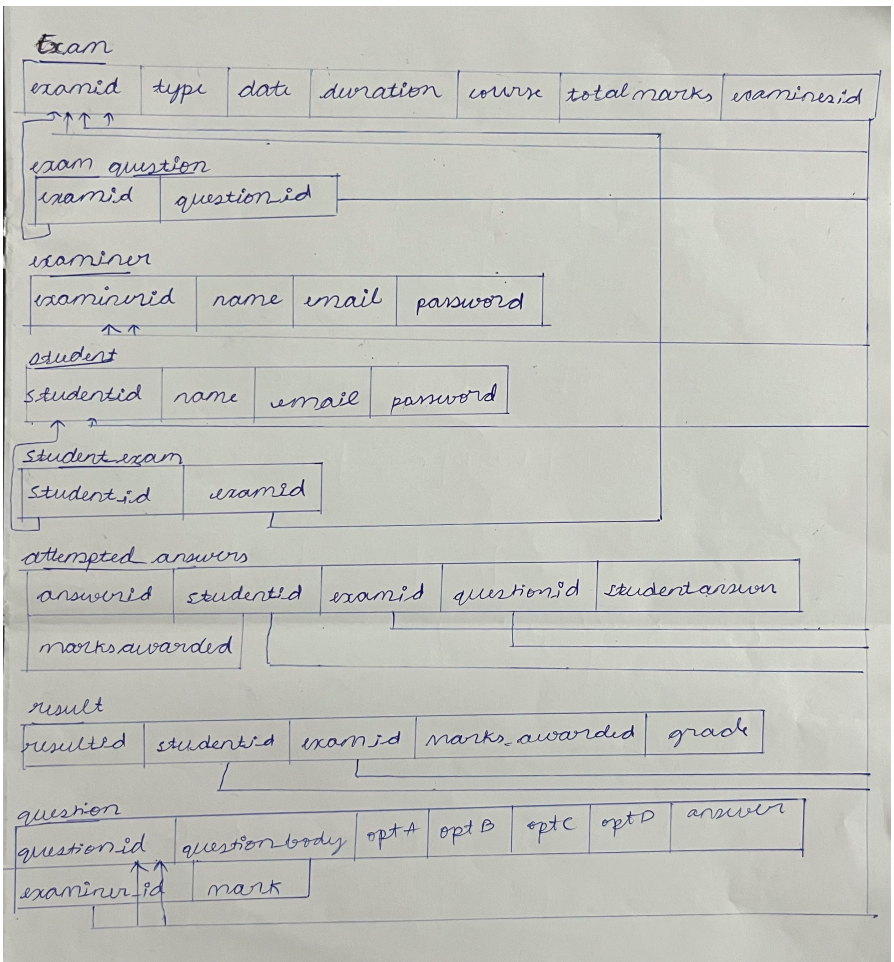
ONLINE EXAMINATION MANAGEMENT SYSTEM REPORT
DBMS MINI PROJECT

M ANIRUDDHA SESHU **: PES1UG22CS309** ,
MADHUSUDANA M **: PES1UG22CS320**

ER diagram :

Relationship schema :



queries:
Triggers :

```
CREATE TRIGGER set_marks_awarded
BEFORE INSERT ON attempted_answers
FOR EACH ROW
BEGIN
    DECLARE correct_answer CHAR(1);
    DECLARE question_mark INT;

    -- Retrieve the correct answer and mark for the question
    SELECT answer, mark INTO correct_answer, question_mark
    FROM question
    WHERE question_id = NEW.question_id;

    -- Check if the student's answer matches the correct answer
    IF NEW.student_answer = correct_answer THEN
        SET NEW.marks_awarded = question_mark;
    ELSE
        SET NEW.marks_awarded = 0;  -- Set to 0 if the answer is incorrect
    END IF;
END//
DELIMITER //
```

1.
before insert on attempted answers from student , it checks for the answer key and give marks accordingly

2.

```
DELIMITER //

CREATE TRIGGER update_total_marks_after_insert
AFTER INSERT ON exam_question
FOR EACH ROW
BEGIN
    DECLARE question_mark INT;

    -- Get the mark for the question being added
    SELECT mark INTO question_mark
    FROM question
    WHERE question_id = NEW.question_id;

    -- Update the total_marks in the exam table
    UPDATE exam
    SET total_marks = total_marks + question_mark
    WHERE exam_id = NEW.exam_id;
END;
//

DELIMITER ;
```

after
insert on exam_question , it updates the total marks on the exam after incrementing the inserted questions mark

3.

```
DELIMITER //

CREATE TRIGGER update_total_marks_after_delete
AFTER DELETE ON exam_question
FOR EACH ROW
BEGIN
    DECLARE question_mark INT;

    -- Get the mark for the question being deleted
    SELECT mark INTO question_mark
    FROM question
    WHERE question_id = OLD.question_id;

    -- Update the total_marks in the exam table
    UPDATE exam
    SET total_marks = total_marks - question_mark
    WHERE exam_id = OLD.exam_id;
END;
//

DELIMITER ;
```

after delete on exam_question , it updates the total marks on the exam after decrementing the inserted questions mark

4.

```
DELIMITER //

CREATE TRIGGER update_result_after_attempt
AFTER INSERT ON attempted_answers
FOR EACH ROW
BEGIN
    DECLARE total_marks INT;
    DECLARE current_marks INT;
    DECLARE percentage DECIMAL(5, 2);
    DECLARE calculated_grade CHAR(1);

    -- Get the total marks for the exam
    SELECT total_marks INTO total_marks
    FROM exam
    WHERE exam_id = NEW.exam_id;

    -- Check if there's already a result entry for this student and exam
    IF EXISTS (SELECT 1 FROM result WHERE student_id = NEW.student_id AND exam_id = NEW.exam_id) THEN
        -- Update existing result entry: add the new marks_awarded to current marks_attained
        UPDATE result
        SET marks_attained = marks_attained + NEW.marks_awarded
        WHERE student_id = NEW.student_id AND exam_id = NEW.exam_id;

        -- Get the updated marks_attained for grade calculation
        SELECT marks_attained INTO current_marks
        FROM result
        WHERE student_id = NEW.student_id AND exam_id = NEW.exam_id;

    ELSE
        -- Insert a new result entry if it doesn't exist
        INSERT INTO result (student_id, exam_id, marks_attained)
        VALUES (NEW.student_id, NEW.exam_id, NEW.marks_awarded);

        -- Set current_marks to marks_awarded for grade calculation
        SET current_marks = NEW.marks_awarded;
    END IF;

    -- Calculate percentage and determine grade
    SET percentage = current_marks / total_marks;

    IF percentage >= 0.9 THEN
        SET calculated_grade = 'S';
    ELSEIF percentage >= 0.8 THEN
        SET calculated_grade = 'A';
    ELSEIF percentage >= 0.7 THEN
        SET calculated_grade = 'B';
    ELSEIF percentage >= 0.6 THEN
        SET calculated_grade = 'C';
    ELSEIF percentage >= 0.5 THEN
        SET calculated_grade = 'D';
    ELSEIF percentage >= 0.4 THEN
        SET calculated_grade = 'E';
    ELSE
        SET calculated_grade = 'F';
    END IF;

    -- Update the grade in the result table
    UPDATE result
    SET grade = calculated_grade
    WHERE student_id = NEW.student_id AND exam_id = NEW.exam_id;

END //

DELIMITER ;
```

this trigger updates the result after each insert on attempted answers , and grades them accordingly at the end

procedures :

1.

```
CREATE PROCEDURE GetPendingExams(IN student_id INT)
BEGIN
    SELECT e.exam_id, e.course, e.type, e.duration, e.date
    FROM exam e
    WHERE EXISTS (
        SELECT 1
        FROM student_exam se
        WHERE se.student_id = student_id AND se.exam_id = e.exam_id
    )
    AND NOT EXISTS (
        SELECT 1
        FROM result r
        WHERE r.student_id = student_id AND r.exam_id = e.exam_id
    );
END //
```

procedure to get all the pending exams the student hasn't taken yet

2.

```
CREATE PROCEDURE GetUnassignedQuestions(IN exam_id INT)
BEGIN
    SELECT q.question_id, q.question_body
    FROM question q
    WHERE NOT EXISTS (
        SELECT 1
        FROM exam_question eq
        WHERE eq.exam_id = exam_id AND eq.question_id = q.question_id
    );
END //

DELIMITER ;
```

procedure to get all the unassigned questions for a particular exam while adding questions to the exam

3.

```
delimiter //
create procedure incorrect_answers_by_exam(in student_id INT , in exam_id INT)
    begin
    select a.question_id from attempted_answers a where a.student_id =student_id and a.exam_id = exam_id
    and marks_awarded = 0;
   end //
elimiter ;
```

procedure to get all the incorrect answers attempted by student ina particular exam

functions :

1.

```
CREATE FUNCTION GetTotalQuestions(p_exam_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE question_count INT;

    SELECT COUNT(*) INTO question_count
    FROM exam_question
    WHERE exam_id = p_exam_id;

    RETURN question_count;
END //
```

function to get the total questions in an exam

2.

```
CREATE FUNCTION GetTotalStudentsInExam(p_exam_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_students INT;

    SELECT COUNT(*) INTO total_students
    FROM student_exam
    WHERE exam_id = p_exam_id;

    RETURN total_students;
END //
```

function to get total students in an exam

3.

```
CREATE FUNCTION GetTotalExamsOfStudent(student_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_exams INT;

    SELECT COUNT(*) INTO total_exams
    FROM student_exam
    WHERE student_id = student_id;

    RETURN total_exams;
END //
```

function
to get total exams taken by a student

nested queries :

```
query = """
    SELECT e.exam_id, e.course, e.type, e.duration, e.date
    FROM exam e
    WHERE NOT EXISTS (
        SELECT 1
        FROM student_exam se
        WHERE se.student_id = %s AND se.exam_id = e.exam_id
    )
"""
```

```
if conn is None:
    print("Failed to get a connection in get_all_exams.")
    return []

try:
    cursor = conn.cursor()
    query = """
        SELECT e.exam_id, e.course, e.type, e.duration, e.date
        FROM exam e
        WHERE EXISTS (
            SELECT 1
            FROM student_exam se
            WHERE se.student_id = %s AND se.exam_id = e.exam_id
        )
    """
```

aggregate queries :

```
SELECT COUNT(*) INTO total_exams
FROM student_exam
WHERE student_id = student_id;
```

```
SELECT COUNT(*) INTO total_students
FROM student_exam
WHERE exam_id = p_exam_id;
```

```
SELECT COUNT(*) INTO question_count
FROM exam_question
WHERE exam_id = p_exam_id;
```

insert queries :

```
insert_query = f"INSERT INTO {table}  VALUES (%s, %s, %s, %s)"
```

```
insert_query = """
INSERT INTO question (question_body, opt_A, opt_B, opt_C, opt_D, answer, examiner_id, mark)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
"""
```

```
query = "insert into exam_question values(%d, %d)" % (exam_id, question_id)
```

```
insert_query = f"INSERT INTO {table} (exam_id,type,date,duration,course,examiner_id) VALUES (%s, %s, %s, %s, %s, %s)"
```

```
table="student_exam"
insert_query = f"INSERT INTO {table} VALUES (%s, %s)"
cursor.execute(insert_query, (student_id,exam_id,))
```

```
insert_query = f"INSERT INTO {table} (student_id,exam_id,question_id,student_answer) VALUES (%s, %s, %s, %s)"
```

update queries :

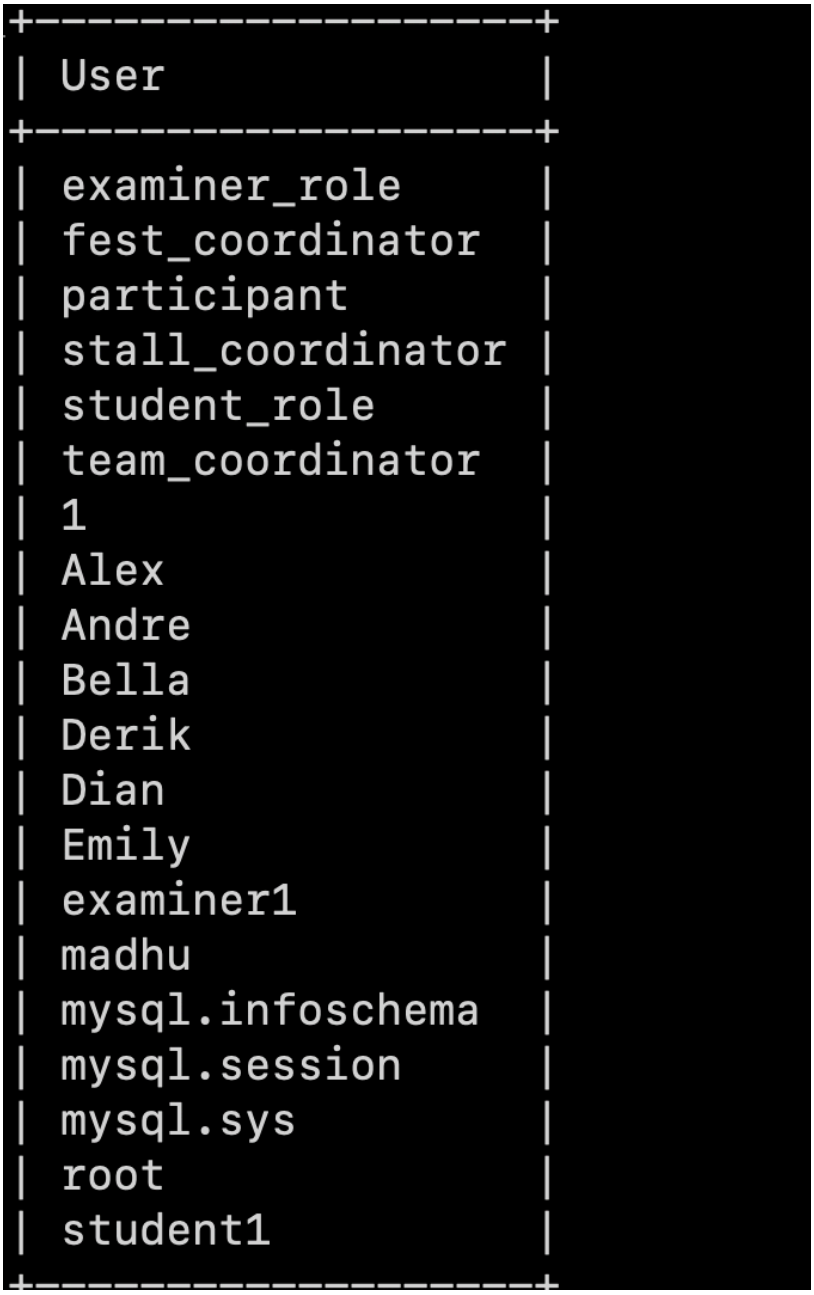
```
    update_query=f"update {table} set password=%s  where {id}=%s "
```

```
    cursor.execute("""
        UPDATE result
        SET grade = %s
        WHERE result_id = %s
    """, (grade, result_id))
```

delete queries :

```
table="student_exam"
delete_query = f"delete from {table} where student_id = %s and exam_id = %s"
cursor.execute(delete_query, (student_id,exam_id,))
```

users :



```
+--------------------+
| User               |
+--------------------+
| examiner_role      |
| fest_coordinator   |
| participant        |
| stall_coordinator  |
| student_role       |
| team_coordinator   |
| 1                  |
| Alex               |
| Andre              |
| Bella              |
| Derik              |
| Dian               |
| Emily              |
| examiner1          |
| madhu              |
| mysql.infoschema   |
| mysql.session      |
| mysql.sys          |
| root               |
| student1           |
+--------------------+
```

join queries :

```
try:
    # Step 1: Get the total marks and marks attained for the given result_id
    cursor.execute("""
        SELECT e.total_marks, r.marks_attained
        FROM result r
        JOIN exam e ON r.exam_id = e.exam_id
        WHERE r.result_id = %s
    """, (result_id,))

    result = cursor.fetchone()
```

```
query=f"select * from question q join exam_question eq on eq.question_id = q.question_id where eq.exam_id = {exam_id}"
```

```
query="select * from result r join exam e on r.exam_id = e.exam_id where r.student_id=%s"
```