Note : I have written the code in text format so that the whole Question can look subjective and more descriptive . Code is written in bold italic style.

To create a database for student entries in a college, we can use the following schema as an example:

CREATE TABLE students (

id INT PRIMARY KEY,

name VARCHAR(50),

age INT,

major VARCHAR(50)

);

example dataset

INSERT INTO students (id, name, age, major)

VALUES (1, 'John Doe', 20, 'Computer Science'),

(2, 'Jane Smith', 21, 'Mathematics'),

(3, 'Mike Johnson', 19, 'English');

Now, let's proceed with Task 1 and demonstrate how different Isolation Levels lead to different results. We'll use the following four isolation levels:

Read Uncommitted

Read Committed

Repeatable Read

Serializable

For each isolation level, we'll perform a transaction that updates a record while another transaction reads the same record. We'll observe the behavior and the results obtained.

Task 1: Different Isolation Levels and Their Effects

1. Read Uncommitted:

-- Transaction 1

START TRANSACTION;

UPDATE students SET major = 'Physics' WHERE id = 1;

COMMIT;

-- Transaction 2

START TRANSACTION;

SELECT * FROM students WHERE id = 1;

In this isolation level, Transaction 2 will read the uncommitted value 'Physics' for the major of student with ID 1. This level allows dirty reads.

2. Read Committed:

-- Transaction 1

START TRANSACTION;

UPDATE students SET major = 'Chemistry' WHERE id = 2;

COMMIT;

-- Transaction 2

START TRANSACTION;

SELECT * FROM students WHERE id = 2;

In Read Committed isolation level, Transaction 2 will only see the committed value 'Chemistry' for the major of student with ID 2. Dirty reads are prevented.

3. Repeatable Read:

-- Transaction 1

START TRANSACTION;

UPDATE students SET age = age + 1 WHERE major = 'Mathematics';

COMMIT;

-- Transaction 2

START TRANSACTION;

SELECT COUNT(*) FROM students WHERE major = 'Mathematics';

In Repeatable Read isolation level, Transaction 2 will see the same count of students majoring in 'Mathematics' throughout its execution. This level prevents non-repeatable reads.

4. Serializable:

-- Transaction 1

START TRANSACTION;

DELETE FROM students WHERE age > 20;

COMMIT;

-- Transaction 2

START TRANSACTION;

SELECT * FROM students WHERE age > 19;

In Serializable isolation level, Transaction 2 will be blocked by Transaction 1 until it completes. This level prevents phantom reads.

Now let's move on to Task 2 and vary the database size to measure the query execution time for the four isolation levels. We'll use MySQL and measure the execution time using the EXPLAIN ANALYZE command.

Task 2: Query Execution Time for Different Database Sizes

To measure the execution time in MySQL, you can use the EXPLAIN ANALYZE command. Here's an example:

EXPLAIN ANALYZE SELECT * FROM students WHERE major = 'Computer Science';