Laboratory Exercise
Concurrency Control

Objectives:

At the end of the exercise, the students should be able to:

- Identify the role of concurrency control in maintaining database integrity; and
- Perform a transaction isolation level.

Software Requirement:

- Microsoft SQL Server Management Studio 18.0 or higher
- Microsoft SQL Server Express 2017

Procedures:

1. Create a database named ConcurrencyControl and create a two (2) tables named Accounts and Transac_History and put the following data.

A_ID	Name	Balance
1	Thor	8,000.00
2	Hulk	4,000.00
3	Thanos	12,000.00
4	Loki	3,000.00
5	Stark	15,000.00

Table 1. *Accounts*

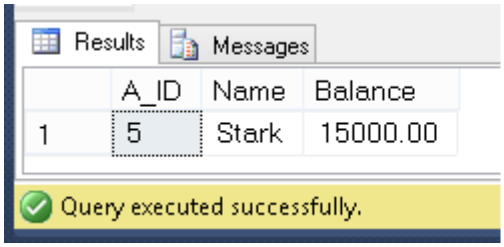
T_ID	Date	Amount	A_ID
1	2020-09-02	-1500.00	3
2	2020-09-03	3000.00	5
2	2020-09-03	-3000.00	1

Table 2. *Transac_History*

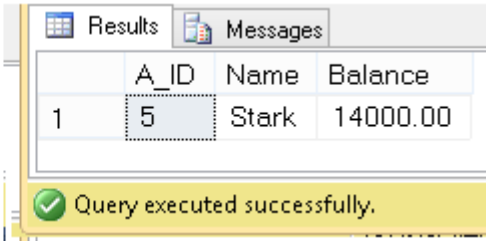
Note: *To understand the data from the Transac_History table clearly, a negative amount means the transaction for that entry deducts the specified amount in the corresponding Account ID or A_ID. Also, if the two (2) entries have the same T_ID, it means that they have a transaction between them where they transfer their funds.*

2. Mr. and Mrs. Stark share an access to an account named Stark. On August 21, 2020 Mr. Stark wants to withdraw an amount of ₱1,000 to his account. While Mr. Stark is working on his transaction, Mrs. Stark start a new transaction that will view the balance of their account. She sees the balance as ₱14,000. Stark on the other hand, did not confirm his transaction and the system automatically aborted his requested transaction. Using READ UNCOMMITTED, write queries that will demonstrate this concurrent transaction.

Output for Mr. Stark's transaction:

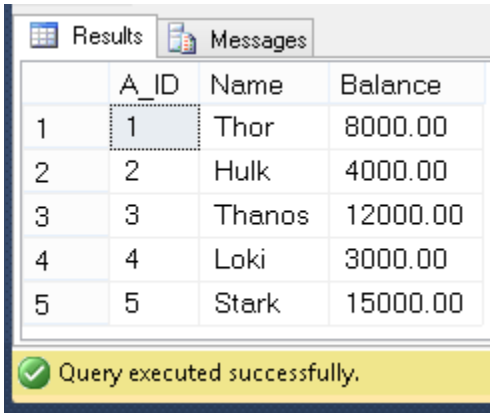


Output for Mrs. Stark's transaction:

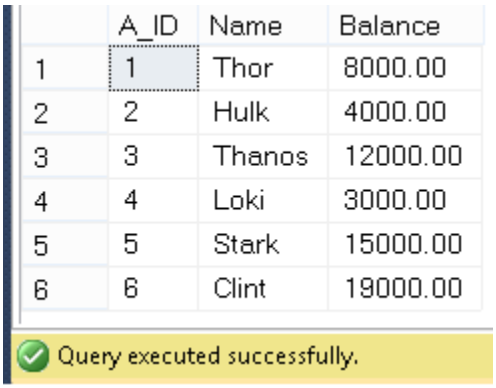


3. On August 28, 2020, the database administrator creates a new transaction on the Accounts table. Still, he wants to obtain a lock that will prevent any other transaction from modifying any rows from it while he was working with his transaction. But he wants other transactions to be able to insert new entries. Meanwhile, an application programmer inserts a new entry in Accounts table: T_ID: 6, Name: Clint, Balance: ₱19,000. At the same time, a sophisticated user wants to modify the balance of Thor by deducting a ₱500. Using the REPEATABLE READ command, write queries that will demonstrate these three (3) concurrent transaction.

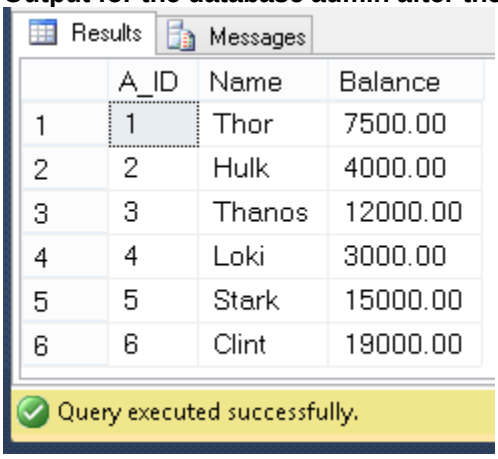
Output for database admin starting the transaction:



Output for the database admin during the execution of the transaction:



Output for the database admin after the execution of the transaction:

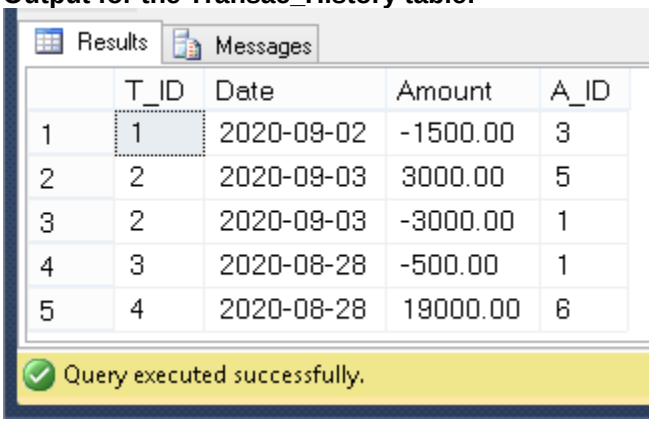


Results | Messages

	A_ID	Name	Balance
1	1	Thor	7500.00
2	2	Hulk	4000.00
3	3	Thanos	12000.00
4	4	Loki	3000.00
5	5	Stark	15000.00
6	6	Clint	19000.00

Query executed successfully.

Output for the Transac_History table:



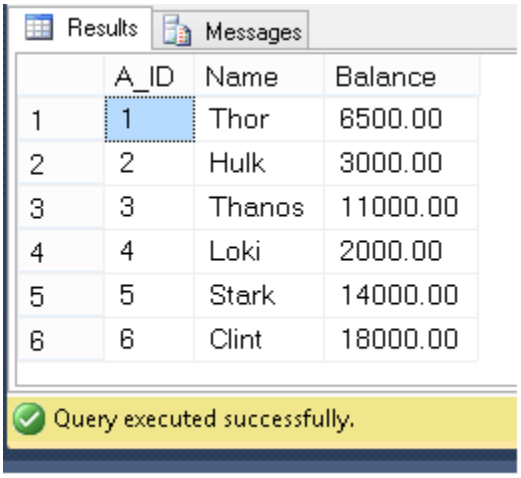
Results | Messages

	T_ID	Date	Amount	A_ID
1	1	2020-09-02	-1500.00	3
2	2	2020-09-03	3000.00	5
3	2	2020-09-03	-3000.00	1
4	3	2020-08-28	-500.00	1
5	4	2020-08-28	19000.00	6

Query executed successfully.

4. On the same day, the database administrator creates a new transaction again by adding up a ₱1000 balance to all rows in the Accounts table. This time, he wants his transaction to obtain an exclusive lock. While the database administrator is working with his transaction, a sophisticated user named Joe wants to add a new entry: T_ID: 7, Name: Natasha, Balance: ₱9,000. Using SERIALIZABLE command, write queries that will demonstrate these two (2) concurrent transactions.
*(**Note:** You can use while loop in order to update all the rows with a specific value.)*

Output for the database administrator:

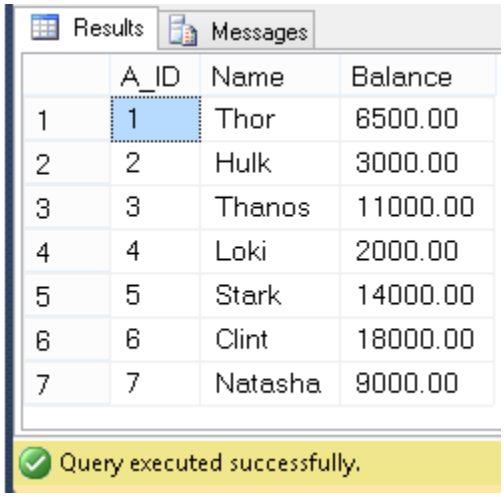


Results | Messages

	A_ID	Name	Balance
1	1	Thor	6500.00
2	2	Hulk	3000.00
3	3	Thanos	11000.00
4	4	Loki	2000.00
5	5	Stark	14000.00
6	6	Clint	18000.00

Query executed successfully.

Output for Joe:



Results | Messages

	A_ID	Name	Balance
1	1	Thor	6500.00
2	2	Hulk	3000.00
3	3	Thanos	11000.00
4	4	Loki	2000.00
5	5	Stark	14000.00
6	6	Clint	18000.00
7	7	Natasha	9000.00

Query executed successfully.

GRADING RUBRIC:

CRITERIA	PERFORMANCE INDICATORS	POINTS
Correctness	The code produces the expected result.	30
Logic	The code meets the specifications of the problem.	30
Efficiency	The code is concise without sacrificing correctness and logic.	20
Syntax	The code adheres to the rules of the database management system.	20
Total		**100**