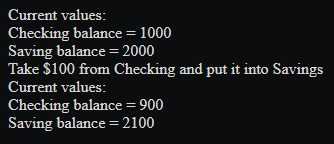
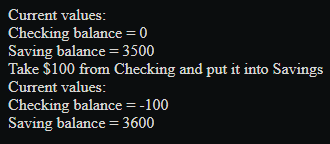
**Database ACID Principles**

**Screenshots**

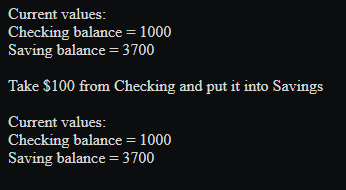
1. **Successful transfer between bank accounts without use of atomic transactions.**



The transaction was completed in two parts in a simple and successful manner.

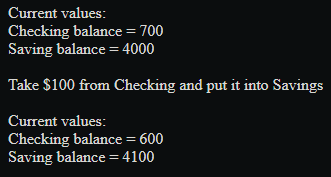
1. **Erroneous transfer when one of the two transfer SQL statements fails.**  
   

As a result of the failed SQL statement, money continues to be withdrawn from Checking and added to Savings despite the Checking account having no money within it.

1. **Successful rollback when one of the two transactions is written incorrectly.**  
   

This process was halted for both checking and saving because incorrect logic was fed into one of the SQL statements.

1. **Successful transfer protected by the transaction and rollback statements.**



Upon successful completion of both SQL statements, the change can be officially committed to the database, as it is an successful atomic transaction.

**ACID Questions**

**Which of the concepts of C.I.A. (Confidentiality, Integrity, and Availability) does the "A" in ACID relate to the most closely?**

With Atomicity, a transaction involving two or more discrete pieces of information must either commit all of them at the same time or none of them at all. This is most closely related to referential integrity or "I" in C.I.A. If we fail to maintain the "I" in the C.I.A., we may encounter data quality issues that are problematic and expensive, (Sluiter, 2018).

**Describe any situation where the social media application could introduce data errors or inconsistencies in the database records.**

Because there is no atomicity in place to check whether a user has liked a photo before increasing its value, a common data error could include a user simply incrementing it repeatedly unchecked.

**Write some pseudo code (or PHP code) that would demonstrate a proper way to utilize transactions and rollbacks so that data consistency errors are not introduced into the tables.**

$db = new Database();

$conn = $db->getConnection();

$conn->autocommit(FALSE);

$conn->begin\_transaction();

$getLikes = $this->getLikes();

$checkLikes = new CheckLikesOnPost($conn);

$okLikes = $checkLikes->updateLikes($getLikes++);

$getUsersWhoLiked = $this->getUsersWhoLiked();

$checkUsers = new CheckUsersWhoLikedPost($conn);

$okUser = $checkUsers->updateUser($user);

if($okLikes && $okUser)

{

$conn->commit();

}

else {

$conn->rollback();

}

# 

# **References**

Sluiter, P. W. P. (2018, July 24). *Activity 5 Part 1 - ACID database transactions.* YouTube. <https://www.youtube.com/watch?v=H59Y4sZ6eb4&feature=youtu.be>

# 

Sluiter, P. W. P. (2018, July 24). *Activity 5 Part 3 - Atomic SQL Transactions in PHP Part 2* YouTube. <https://www.youtube.com/watch?v=O25JsAQ-2O8>

# 

Sluiter, P. W. P. (2018, July 24). *Activity 5 Part 4 - Atomic SQL Transactions in PHP Part 3* YouTube. <https://www.youtube.com/watch?v=g3WZJrnUZzw>

# 