Consistency

by Sophia

WHAT'S COVERED

This tutorial explores the consistency property in a transaction and how it affects the database in three parts:

1. Introduction
2. Criteria for Consistency
3. Consistency Example

1. Introduction

Consistency within the ACID properties focuses on ensuring that the data in the database moves from one valid state to another valid state. This ensures that any data that has been modified in the database is uncorrupted and correct at the end of the transaction. With the consistency property, the database should not be in a partially completed state.

2. Criteria for Consistency

The consistency property follows the following criteria:

1. If the transaction has completed successfully, the changes will be applied to the database.
2. If there was an error in the transaction, all of the changes should be reverted/rolled back automatically. This means that the database should restore the pre-transaction state.
3. If there was a system failure or external issue while the transaction was executing, all of the changes that were made in the transaction up to that point should automatically be reverted/rolled back.

3. Consistency Example

Let's look at our banking example again. Jennifer would like to make a $100 payment to Randall through an account transfer. This transaction is a balance transfer between two accounts at two different branches of the same bank. Let us review what the transaction would look like:

1. Jennifer's (10) account would be deducted by $100.
2. The banking location where Jennifer has her account would have their location's account deducted by

$100.

3. The banking location where Randall (50) has his account would be increased by $100.
4. Randall's account would be increased by $100.

The consistency property ensures that the total value of each type of account is the same at the start and the end of the transaction. This means that the customer_accounts and branch_accounts would have a consistent total to account for each statement. Let us look back at the transaction in SQL:

```
BEGIN;

UPDATE customer_account
SET balance = balance – 100
WHERE account_id = 10;

UPDATE branch_account
SET balance = balance – 100
WHERE branch_id = (SELECT branch_id FROM customer_account where account_id = 10);

UPDATE branch_account
SET balance = balance + 100
WHERE branch_id = (SELECT branch_id FROM customer_account where account_id = 50);

UPDATE customer_account
SET balance = balance +100
WHERE account_id = 50;

COMMIT;
```

Imagine that during the second UPDATE statement, the system had a failure and when it recovered, the transaction had only partially executed. There would be an inconsistent state because the total balances would not match up. In this situation, the system would roll back those UPDATE statements to the consistent state prior to the transaction starting. Unlike with atomicity, the issue was not caused by an error in the database statement.

If both Jennifer's and Randall's account balances started at $1000, the end result should have the appropriate expected balances. Jennifer's account balance should be set at $900, and Randall's balance should be at $1100. If for any reason, the end values were not what was expected, the transaction would also be rolled back.

SUMMARY

The consistency property ensures that each transaction starts in a consistent state and ends in a consistent state.

Source: Authored by Vincent Tran