ACID: My Personal “C” Change

**HOW COULD
I MISS SUCH A
SIMPLE THING?**

PAT HELLAND

For decades, I thought the C in transactional ACID was the weakest property. Just shows how I was a dummy for 37 years.

Under its entry for Consistency (database systems), Wikipedia provides a pretty typical explanation of the C in ACID:

Consistency in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

This does not guarantee correctness of the transaction in all ways the application programmer might have wanted (that is the responsibility of application-level code) but merely that any programming errors cannot result in the violation of any defined database constraints.

This description doesn't seem very concrete, so I've always been somewhat dismissive of consistency as being equal to atomicity, isolation, and durability. In general, I figured the C characteristics as described by Wikipedia were really just a restatement of isolation.

I had a chance recently to chat with my old friend,

*It's Not Your Grandmother's
Database Anymore

[Andreas Reuter](#), the inventor of ACID. He and his Ph.D. advisor, Theo Härder, coined the term in their famous 1983 paper, Principles of Transaction-Oriented Database Recovery.¹

Since transactions, both theory and practice, had been one of my biggest passions since 1978, I vividly remembered the publication of this paper and the creation of the ACID concept. When I got the chance to meet Andreas in person in 1985, we became friends.

Lately, I've been thinking more about consistency and what it means in our many computer science communities. In advance of my recent chat with Andreas, I pulled up the paper that introduced the concept of C and read it again. Nothing new jumped out at me.

As Andreas and I caught up on our lives and on our thoughts on technology, I asked him about his intentions when he added consistency to the ACID test. Wasn't it just warmed-over isolation? He said that an application needed to control what was included in a transaction to ensure that the rules of the application were consistent. The C meant *the application decided the completion of the set of changes*. Hence, as the application wrote the set of changes, it could enforce constraints, cascades, triggers, and so forth. It could also enforce application-specific rules meaningful only within that instance [for example, an airline's special treatment of frequent flyers].

I was gobsmacked! For 37 years, I'd had it wrong! C is a powerful, simple, and important member of ACID. That simple additional rule—side by side with atomic, isolated, and durable—did allow for a more cohesive semantic enforced by an application as it changed the database.

One big
lesson for
me is to
work hard
to *always*
question your
assumptions

It was the definitions I'd seen for so many years, as typified by the Wikipedia entry cited earlier, that cast C in a narrower light by focusing on the *consequences* of the actual rule and *not on the rule itself*.

But what of the original paper that introduced ACID in the first place? I had read it just minutes before chatting with Andreas. Looking at it [yet again] after our conversation I clearly saw:

Consistency. A transaction reaching its normal end [EOT, end of transaction], thereby committing its results, preserves the consistency of the database. In other words, each successful transaction by definition commits only legal results. This condition is necessary for the fourth property - durability.

So, even as I looked at the definition before asking Andreas, I had blinders on after almost four decades of seeing C based on my assumptions. I just didn't get the simplicity until Andreas said it so bluntly.

One big lesson for me is to work hard to *always* question your assumptions. Try hard to surround yourself with curious and passionate people, both young and old, who will challenge you and try to dislodge your blinders. Foster a culture that makes them safe as they do so.

Thanks to my old friend Andreas too!

References

1. Härder, T., Reuter, A. 1983. Principles of transaction-oriented database recovery. *ACM Computing Surveys* 15(4); <https://dl.acm.org/doi/10.1145/289.291>.

Pat Helland *has been implementing transaction systems, databases, application platforms, distributed systems, fault-tolerant systems, and messaging systems since 1978. For recreation, he occasionally writes technical papers. He works at Salesforce. His blog is at pathelland.substack.com. Follow him on Twitter at [@pathelland](https://twitter.com/pathelland).*

Copyright © 2021 held by owner/author. Publication rights licensed to ACM.