TECHNICAL PERSPECTIVE: Ad Hoc Transactions: What They Are and Why We Should Care

Kenneth Salem
ken.salem@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Most database research papers are prescriptive. They identify a technical problem and show us how to solve it. They present new algorithms, theorems, and evaluations of prototypes. Other papers follow a different path: descriptive rather than prescriptive. They tell us how data systems behave in practice, and how they are actually used. They employ a different set of tools, such as surveys, software analyses or user studies. These papers are much rarer at database research conferences, and they're all the more valuable for that.

The paper I'm introducing here, *Ad Hoc Transactions: What They Are and Why We Should Care*, by Tang et al., is of this descriptive type. It presents an analysis of open-source database applications, focusing on how these applications synchronize concurrent operations. According to the authors, this analysis is the result of five person-years (!) of effort. (I suspect that this is one of the reasons that work like this is not more common.) This paper originally appeared at SIGMOD'22 [2], and it extends a (short) research thread that first came to my attention through earlier work by Warszawski, Bailis and others [1, 3].

"Wait", I hear you say. "Database systems provide transactions, and applications use transactions to synchronize their database accesses. What's to study here?" If this is your immediate reaction, I urge you to read this paper. Transactions are, of course, widely used, but database applications also have other tools at their disposal for coordinating concurrent actions. These include synchronization primitives provided by database systems, such as explicit locks, as well as synchronization mechanisms in the application itself. Tang et al refer to applications' use of these mechanisms as *"ad hoc transactions"*, to distinguish them from transactions implemented in the database system.

As it turns out, ad hoc transactions are quite common. Tang et al analyzed eight popular database applications, and found ad hoc transactions in all of them - more than 90 examples of ad hoc transactions in all. This ad hoc transaction corpus is the focus of the paper.

Let's consider one example from the corpus, as an illustration. It is an ad hoc transaction used in the Discourse forum application to update the content of a forum post. It spans two application-level HTTP requests. In the first, a client requests the current content of a forum post for editing. In response, the server-side application uses a read-only database transaction to retrieve the post content and version number, and returns them to the client. The client edits the content, and then issues a second HTTP request to install the new post content, providing the original version number. In response, the server-side application first uses a read-only database transaction to verify that the version number has not changed since the content was originally retrieved. If it has, the update fails. Otherwise, the application uses another database transaction to install the new content and update the version number. The application also uses an explicit lock on the post identifier to prevent concurrent updates to the post between the version check transaction and the update transaction.

Logically, this whole process is a single long-lived transaction spanning two HTTP requests. In practice, the application uses an ad hoc synchronization strategy involving three database transactions, an explicit lock, and an explicit optimistic concurrency control (via the version number check) to coordinate the activity.

The paper does a great job addressing the important questions about these kinds of ad hoc transactions. First, what kinds of ad hoc transactions are found in practice? Here the paper identifies a number of common patterns that were found in the corpus. Second, how are these ad hoc transactions being implemented, i.e., what synchronization primitives are the applications using? Third, *why* are applications using ad hoc synchronization, rather than simply relying on database transactions.

Synchronization is tricky, and so the paper also examines what can go wrong when applications rely on ad hoc transactions. You may be unsurprised to hear that authors found dozens of correctness issues related to ad hoc transactions in these applications. They nicely summarize and classify these issues.

This a super informative paper, representing a whole lot of work. It should be valuable for engineers who build database applications. For researchers interested in designing useful synchronization mechanisms and abstractions for data systems, it should be considered a must-read.

REFERENCES

[1] Peter Bailis, Alan Fekete, Michael J. Franklin, Ali Ghodsi, Joseph M. Hellerstein, and Ion Stoica. 2015. Feral Concurrency Control: An Empirical Investigation of Modern Application Integrity. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1327–1342. https://doi.org/10.1145/2723372.2737784

[2] Chuzhe Tang, Zhaoguo Wang, Xiaodong Zhang, Qianmian Yu, Binyu Zang, Haibing Guan, and Haibo Chen. 2022. Ad Hoc Transactions in Web Applications: The Good, the Bad, and the Ugly. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*. 4–18. https://doi.org/10.1145/3514221.3526120

[3] Todd Warszawski and Peter Bailis. 2017. ACIDRain: Concurrency-Related Attacks on Database-Backed Web Applications. In *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*. 5–20. https://doi.org/10.1145/3035918.3064037

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06.
https://doi.org/XXXXXXX.XXXXXXX