Transactions best practices [closed]

Asked 16 years, 3 months ago Modified 15 years, 2 months ago

Viewed 18k times

31

How much do you rely on database transactions?

Do you prefer small or large transaction scopes ?

Do you prefer client side transaction handling (e.g. TransactionScope in .NET) over server side transactions or vice-versa?

What about nested transactions?

Do you have some tips&tricks related to transactions ?

Any gotchas you encountered working with transaction ?

All sort of answers are welcome.

database architecture transactions

Share

Improve this question

Follow

edited Sep 5, 2008 at 14:07

asked Sep 2, 2008 at 13:59

aku

124k ● 33 ● 176 ● 203

7 Answers

Sorted by: Highest score (default)

21

I always wrap a transaction in a using statement.

```
using(IDbTransaction transaction )
{
// logic goes here.
   transaction.Commit();
}
```

Once the transaction moves out of scope, it is disposed. If the transaction is still active, it is rolled back. This behaviour fail-safes you from accidentally locking out the database. Even if an unhandled exception is thrown, the transaction will still rollback.

In my code I actually omit explicit rollbacks and rely on the using statement to do the work for me. I only explicitly perform commits.

I've found this pattern has drastically reduced record locking issues.

Share Improve this answer Follow

edited Sep 2, 2008 at 14:14

answered Sep 2, 2008 at 14:08

Simon Johnson

7,902 ● 5 ● 38 ● 50

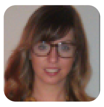
7

Personally, developing a website that is high traffic perfomance based, I stay away from database transactions whenever possible. Obviously they are neccessary, so I use an ORM, and page level object variables to minimize the number of server side calls I have to make.

Nested transactions are an awesome way to minimize your calls, I steer in that direction whenever I can as long as they are quick queries that wont cause locking. NHibernate has been a savior in these cases.

Share Improve this answer Follow

answered Sep 2, 2008 at 14:10



Sara Chipps

9,372 ● 11 ● 59 ● 104

please, help me to decode this "i'm avoiding transaction" "but they are necessary" "so I use an ORM". these three are kinda a puzzle - because the first statement contradicts the last, or I didn't get something? – maxkoryukov Aug 19, 2022 at 11:58

3

I use transactions on every write operation to the database.
So there are quite a few small "transactions" wrapped in a larger transaction and basically there is an outstanding transaction count in the nesting code. If there are any outstanding children when you end the parent, its all rolled back.

I prefer client-side transaction handling where available. If you are relegated to doing sps or other server side logical units of work, server side transactions are fine.

Share Improve this answer

Follow

answered Sep 2, 2008 at 14:04

DevelopingChris

40.7k ● 30 ● 89 ● 119

1 Client side transactions??! – Totty.js Aug 16, 2017 at 16:37

3

Wow! Lots of questions!

Until a year ago I relied 100% on transactions. Now its only 98%. In special cases of high traffic websites (like Sara mentioned) and also high partitioned data, enforcing the need of distributed transactions, a transactionless architecture can be adopted. Now you'll have to code referential integrity in the application.

Also, I like to manage transactions declaratively using annotations (I'm a Java guy) and aspects. That's a very clean way to determine transaction boundaries and it includes transaction propagation functionality.

Share Improve this answer

Follow

answered Sep 2, 2008 at 15:53

Marcio Aguiar

14.5k ● 6 ● 40 ● 42

it is worth mentioning, that in the case of a regular application in a relational DB it is better to go with transactions. because *Now you'll have to code referential integrity in the application.*

- it is just the same transactions (already implemented for the DB), but for the application with a complicated DB, optimized for a particular task (high speed for a particular table / or handling shards/replicas) – maxkoryukov Aug 19, 2022 at 12:03

3

Just as an FYI... Nested transactions can be dangerous. It simply increases the chances of getting deadlock. So, though it is good and necessary, the way it is implemented is important in higher volume situation.

Share Improve this answer Follow

answered Sep 5, 2008 at 14:45

oglester
6,655 ● 8 ● 44 ● 64

2

Server side transactions, 35,000 transactions per second, SQL Server: 10 lessons from 35K tps

We only use server side transactions:

- can start later and finish sooner
- not distributed
- can do work before and after
- SET XACT_ABORT ON means immediate rollback on error
- client/OS/driver agnostic

Other:

- we nest calls but use @@TRANCOUNT to detect already started TXNs
- each DB call is always atomic

We deal with millions of INSERT rows per day (some batched via staging tables), full OLTP, no problems. Not 35k tps though.

Share Improve this answer Follow

answered Oct 16, 2009 at 15:40

gbn
432k ● 83 ● 597 ● 684

0

As Sara Chipps said, transaction is overkill for high traffic applications. So we should avoid it as much as possible. In other words, we use a BASE architecture rather than ACID. Ebay is a typical case. Distributed transaction is not used at all in Ebay architecture. But for eventual consistency, you have to do some sort of trick on your own.

Share Improve this answer Follow

answered May 14, 2009 at 3:38

yanky
141 ● 1 ● 6