

## Re: pooled connection myth

**Source:** <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-03/1898.html>

---

**From:** Lee Fesperman ([firstsql\\_at\\_ix.netcom.com](mailto:firstsql_at_ix.netcom.com))

**Date:** 03/17/05

Date: Thu, 17 Mar 2005 22:31:32 GMT

David McDivitt wrote:

>  
> >From: Lee Fesperman <[firstsql@ix.netcom.com](mailto:firstsql@ix.netcom.com)>  
> >Subject: Re: pooled connection myth  
> >Date: Thu, 17 Mar 2005 19:58:04 GMT  
> >  
> >David McDivitt wrote:  
> >>  
> >> *There is much to be gained by reusing known open connection objects, rather  
> >> than issuing single use connection objects for each request. If connection  
> >> objects are not modified by using transaction blocks, there's no reason not  
> >> to share them between threads. The word "pooling" implies this, but pooling  
> >> itself doesn't do very much. For the most part, all that happens is caching  
> >> the database user name and password.*  
> >  
> >As John said, the usefulness of concurrent sharing for connection objects varies by the  
> >driver. In addition, the Connection object does contain state information. Take a look  
> >at the various `setXXX()` methods in `java.sql.Connection`.  
> >  
> >Your assumptions about pooling are not very valid. Other types of resources, including  
> >threads, are pooled in Java but rarely concurrently. Concurrent use of pooled resources  
> >would tend to imply that all state be read-only, thus no pooling as such is not needed.  
> >All you need to do is to keep a common reference (GC will take care of disposing of the  
> >object when no longer needed). While this isn't absolutely true in all cases, it covers  
> >most of the interesting ones.  
> >  
> >Also, caching of connection properties is just one aspect and will normally include more  
> >than just user/password. Then there is the very important feature of avoiding the  
> >overhead of making (and ending) a connection to the db and resultant pressure on  
> >resources on the server. In addition, some connection pooling software support  
> >(automatic, hidden) pooling of prepared statements.  
> >  
> >Finally, please do not top-post. It's just basic netiquette.  
>  
> If more functionality is present in the connection manager I want to see it.  
> I do not want to take it on faith. What you say here does not represent  
> enough value compared to increased complexity. Shared connections are good.  
> Pooled connections, are not good, since pooling itself does not accomplish

comp.lang.java.programmer: Re: pooled connection myth

- > *anything. The only way it can be of value is to intercept close and open*
- > *calls and reuse hot connections. This is not being done where I am.*

Thanks for responding (and for inline posting). I don't know the connection manager that you are using. Note: ConnectionManager is a specific facility in JCA; this is probably not the JCA one. The one you're using may be a poor facility, however you also made assertions about connection pooling in general which aren't valid.

I think I did list the major benefits of connection pooling. The one I missed was reduced resource requirements and better control of such by the containing software (for one thing, it gives the container the ability to say: you can't have another connection at this time.) If those are not of interest to you, fine, though tighter control of resources is quite important to many containers. For instance, JCA gives the container control over the use of threads by connectors.

As to shared connections, I see that to be of very limited use. I made comments on that in the other sub-thread and would be interested in your counter. If you also look at the thread on comp.lang.databases (I'll get the title if you wish), you will see arguments why a server would not want to increase that capability.

--

Lee Fesperman, FFE Software, Inc. (<http://www.firstsql.com>)

=====

\* The Ultimate DBMS is here!

\* FirstSQL/J Object/Relational DBMS (<http://www.firstsql.com>)