

Re: SQL Compact Edition – connection and multi–threading

Source:

<http://www.tech–archive.net/Archive/SQL–Server/microsoft.public.sqlserver.ce/2007–06/msg00002.html>

- *From:* "José Joye" <jose.joye_KILLTHESPAMS_SMAPSEHTLLIK_@xxxxxxxxxx>
 - *Date:* Fri, 1 Jun 2007 21:39:03 +0200
-

Thanks for your feedback,

I have implemented it and it works really great.

–José

"joker" <joker@xxxxxxxxxx> a écrit dans le message de news:
1180610174.075671.31500@xx
So you wrote your own "connection pool"

Some of the issues you will find are:

Keeping track of which connections are "in use" and which are "free"
What do you do when you have no more "free" connections and one is
requested?

I don't foresee any threading issues should you solve the simple
problems above.

I'm also not convinced that this design is "naive". I think mobile
applications are changing rapidly and given more and more threads,
should you really be giving each thread a connection which it holds
open indefinitely?

Your design also gives you the opportunity to actually close the
connections once in a while (quite easily) if you need to move/copy/
delete/compact the database file.

Good luck!

Doug

On May 31, 2:27 am, "José Joye" <M...@xxxxxx> wrote:

Re: SQL Compact Edition – connection and multi–trheading

This question relates to another one I posted. However it derives from the original post and found better to start a new one.

I'm writting a library application that will be used internally within our company (dB is used in RO mode). I already expect it to be used by multithreaded applications.

Having that in mind, I naively designed it with connection pooling in mind (--> get a connection, use it and release it as soon as possible [kind of single call pattern]).

However, the connection pool is not supported by the SQLServer CE 3.1 and this means every time I'm asking for a connection object I have to wait about 2ms on my desktop and I expect to be much more on the CF devices

The question I have is related to connection together with multithreading. In fact, to overcome the speed problem I have, I'm planning to write a kind

of "DataObjectProvider singleton Class" that will hold a set of already initialized "db onnection".

Whenever needed, clients of this class may ask for an available connection and when they are done, they can replace it in the pool.

1. Assuming, my code ensures the connections are really given back to the singleton class when not needed anymore. Is this design valid?
2. Does it cause problem if different threads will query for connections (which were created by a different thread)?
3. Assuming I keep the connection open, when I place them back to the pool, will it make problem when used in multithreaded environment.

Thanks,
– José