

Re: Portable stored procedures

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.databases/2004-02/0079.html>

From: Robert Klemme (*bob.news_at_gmx.net*)

Date: 02/09/04

Date: Mon, 9 Feb 2004 13:05:04 +0100

"Dennis" <df@tdc-broadband.dk> schrieb im Newsbeitrag
news:40276024\$0\$1599\$edfadb0f@dread14.news.tele.dk...

>

> > > *It also have a nice features that lets you write binary streams to
it,*

> > *so in*

> > > *addition to the database, you get yourself a virtual file system.*

> >

> > *Well, that's not really a surprising feature: nearly all db's you can
get*

> > *today support BLOB's.*

>

> *JDatastore lets you store binary streams outside the table scema. You
can*

> *store them in a directory structure. A BLOB (which is of course
supported by*

> *almost all databases today) is usually a binary field stored in a
record in*

> *a table.*

What do you gain by this? Other databases usually do not store BLOB's
physically at the same page as other fields of a record. So there
shouldn't be performance issue.

> > > *By the way – using java as the language for stored procedures is of*

> > *course*

> > > *extremely smart. They probably perform way better than other
languages,*

> > *and*

> > > *you only have to develop the code once.*

> >

> > *I haven't used Java stored procedures with databases but your
statement is*

> > *only true, if all databases involved had the same API's and
conventions*

> > *regarding stored procedure parameters, return values and exceptions.*

>

comp.lang.java.databases: Re: Portable stored procedures

- > *What i meant is that if you have for instance a function to parse or change*
- > *a string, that was needed in the client and in the server – in many cases,*
- > *you will have to write the function in the client in Java (or whatever language you are using) and in the server in some sql–script language – or C*
- > *for speed. With JDatastore, you can put the same class on client and server,*
- > *therefore ensuring that the client executes the same functions as the server*
- > *for the same tasks.*

Ok, that's true.

- > > *Apart from that, Java stored procedures are not necessarily faster than*
- > > *stored procedures written in some proprietary language.*
- >
- > *Not neccesarily, i agree. That would be up to a test.*
- >
- > > *I personally would not believe promises that claim portable stored*
- > > *procedures. From my experience stored procedures is one of the fields*
- > > *where databases differ most.*
- >
- > *I didn't mean portable *that* way. i meant that you have the opportunity to*
- > *re–use certian classes in the client, in the middle layer, and inside the*
- > *SQL server.*

So you meant "reuse" while writing "portability" – actually two very different concepts.

- > > > *Performance is comparable to other (even non–java) databases. Java keeps*
- > > *on*
- > > > *getting faster and faster.*
- > >
- > > *That may be true as the main performance factor of a database is the*
- > > *smartness of the algorithms and disk speed. Nevertheless I would not*
- > > *expect a db implemented using Java to be as performant as any other native*
- > > *compiled db. There are some obstacles to that (object creation*
- > > *overerhead, memory size limits for JVM's on certain platforms etc.).*
- >
- > *Actually, with java versions running on sdk 1.2 (1.4.1 and 1.4.2 i think)*
- > *from SUN, the creation of an object is comparable or faster than C and C++.*
- > *Reason being that the heap is guarenteed to be unfragmented, and thus,*

Re: Portable stored procedures

> *addition of an object is simply a matter of adding to a pointer.*

There's some additional management overhead involved so it's not that simple. I'm not going to dig deeper into this since then it would be drifting too far off-topic...

> *Object dallocation is very fast for shortlived objects. So fast that sun's*

> *tests suggests that object pooling of small objects is outperformed by*
> *creating and removing these objects.*

> *Reason is that the garbage collector copies any surviving objects, then*
> *wipes that part of the heap. So it takes the same amount of constant*
time to

> *GC 10000 objects, as it takes to GC 10 objects. What matter is the*
number of

> *objects that live longer than the shortest GC interval.*

>

> *there is a discussion on this subject at*

> <http://www-106.ibm.com/developerworks/library/j-jtp01274.html?ca=dnt-54>

> *(not sure if you have to be a member to read it. if so – become member,*
it

> *is free, and they post good stuff from time to time)*

You don't need to. Thanks for that link. I'll have a look.

> *a slideshow highlighing important changes to sun's JVM:*

>

<http://servlet.java.sun.com/javaone/resources/content/sf2003/conf/sessions/pdfs/1522.pdf>

>

> *You do mention one thing that is VERY important to those who are about*
to

> *consider database for a big project... platforms... Java is performing*
very

> *very fast on the windows/intel platform, but is not performing as well*
on

> *some other platforms (it will in due time, but i guess the newest JVM's*
on

> *these platforms are a little older, or not as optimized). You might end*
up

> *with running a 300% slower JVM on a 200% faster unix box, compared to a*

> *windows/intel. Therefore – if you are deploying onto some heavy duty*

> *equipment, check out that you have the newest JVM, and check out if it*

> *performs well. I'm sure there is no problem on the most popular*
platforms

> *(ibm, linux, whatever).*

Even more so benchmark the VM's and profile applications.

Regards

robert