

Re: Should I use LISP for this?

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-03/0344.html>

From: John Thingstad (*john.thingstad_at_chello.no*)

Date: 03/03/04

Date: Wed, 03 Mar 2004 12:57:23 +0100

On Mon, 1 Mar 2004 21:11:33 -0600, Alan Walker
<alan-walker-hater-of-spam@charter.net> wrote:

MySQL is fast and adequate for his problem.
It uses ISAM (IBM) tables directly.
In it's fastest implementation (it has two) it sacrefises recoverability
for speed.
Nevertheless it should be up to the job and many times faster than, say,
Oracle.

> *Seems more like a database problem. Check out MySQL. If you want data*
> *in*
> *memory, use MySQL Cluster.*
>
> *The query optimizer will look at your query and determine the best way to*
> *handle it. You can beat the database using pre-navigated, in-memory*
> *structures, but it will shine for ad-hoc queries. Also, you can add*
> *secondary indexes on the fly in a database as needs change. It will also*
> *handle concurrency, updating the structure while people query it.*
>
> *If you really insit on building in-memory structures, Lisp is a good*
> *choice.*
> *C++ will work for simple structures, but will drive you nuts with complex*
> *structures. GC is also a plus.*
>
> *Now, using Lisp as a front-end to your database, that's a good idea...*
>
> *Alan.*
>
>
> *"Star" <staryon@hotmail.com> wrote in message*
> *news:e0a0c75d.0403011854.7b8419c8@posting.google.com...*
>> *Hi*
>>
>> *I have a program that handle very big structures of graphs. I was*
>> *using C++ language a SQL Server to handle that, but the perfomance*
>> *that I have is very slow. I have been suggested to use LISP to handle*
>> *this, but I haven't used LISP in many years, and before starting*

comp.lang.lisp: Re: Should I use LISP for this?

>> *working on this, I would like to know your opinion. Here is the*
>> *problem:*
>>
>> *I have about 15 tables. Each table has several fields, however each*
>> *table*
>> *has a common field called 'Code'*
>>
>> *ex.*
>>
>> *Table Vehicles*
>> -----
>> *Code*
>> *Make*
>> *Color*
>>
>> *Table People*
>> -----
>> *Code*
>> *Name*
>> *Age*
>>
>> *Table Addresses*
>> -----
>> *Code*
>> *City*
>> *ZIP*
>> ...
>>
>> *Now we want to relate records from one table to another. For example,*
>> *we*
>> *want to say 'John has two cars'*
>> *In order to do that, I have an external table called Links with source*
>> *an*
>> *destination fields (I store there the code of each table). For the*
>> *example we would have*
>>
>> *Source Dest*
>> -----
>> *P132 V100*
>> *P132 V242*
>>
>> *(P132 is a record from the People table with code 132, V100 a vehicle*
>> *from*
>> *the Vehicle table with code 100, and V242 a vehicle*
>> *with code 242)*
>>
>> *So, with all that, we know that John (P132) is related with two*
>> *vehicles. If*
>> *I want to run the query 'People associated with vehicles'*
>> *it would be easy to query that table and get that information.*
>>

>> *Now, let's suppose we have this situation:*
>>
>> *Source Dest*
>> -----
>> *P132 V100*
>> *P132 V242*
>> *P140 V500*
>> *P132 P140*
>>
>> *In this example, John is related with Michael (P140) and Michael is*
>> *related*
>> *with a vehicle (V500).*
>> *If I want to run the same query than before 'People associated with*
>> *vehicles', because of P132 is related with P140 and P140 is related*
>> *with*
>> *a V500 this implies that P132 is related with V500.*
>>
>> *Now, the query is not so simple, because it's like a graph and I need*
>> *to*
>> *make a Depth First Search to get what I want. Doing that, I will get*
>> *that*
>> *P132 is*
>> *related with V500 because there is a 'path' to get there.*
>>
>> *I have done all this and it works fine. The problem that I have is*
>> *that I*
>> *need to make that Depth First Search for each people that I have in*
>> *the*
>> *database*
>> *in order to see if I have vehicles associated with it. If I don't have*
>> *many*
>> *records it's ok, but If I have 100,000 people on the database I would*
>> *have*
>> *to*
>> *make that search 100,000 times just to see if there is a path between*
>> *them,*
>> *I mean, if there is a way to get from a person to a vehicle.*
>> *In this case... this solution doesn't work, because it takes*
>> *forever...*
>> *I would need to make any kind of query on this structure.. ex.*
>> *vehicles associated with vehicles, persons with vehicles, addresses*
>> *with persons..*
>>
>> *What do you guys suggest I can try? I though that one solution would*
>> *be to*
>> *have a process running on the server that maintains a table with ALL*
>> *the*
>> *links that we have in the database and the depth of the link. So, in*
>> *our*
>> *example we'd have a entry "P132 V500 with depth 2"*
>> *Having that table it would be easy to query. Inconvenience: the table*

comp.lang.lisp: Re: Should I use LISP for this?

>> *is*
>> *going to be HUGE, and also the process has to be smart enough to keep*
>> *track*
>> *of*
>> *deleted links, new links...*
>>
>> *I'd appreciate if you guys have any idea or suggestions about this and*
>> *how I*
>> *can get a better performance.*
>>
>> *Thanks a lot for reading this whole message!!*
>
>

--

Using M2, Opera's revolutionary e-mail client: <http://www.opera.com/m2/>