John Meegan

From: Gavin Schultz
Sent: Monday, 27 April 2009 10:16 PM
To: QESTLab
Cc: QESTMix; Spectra QEST Help Desk
Subject: Security model updated again

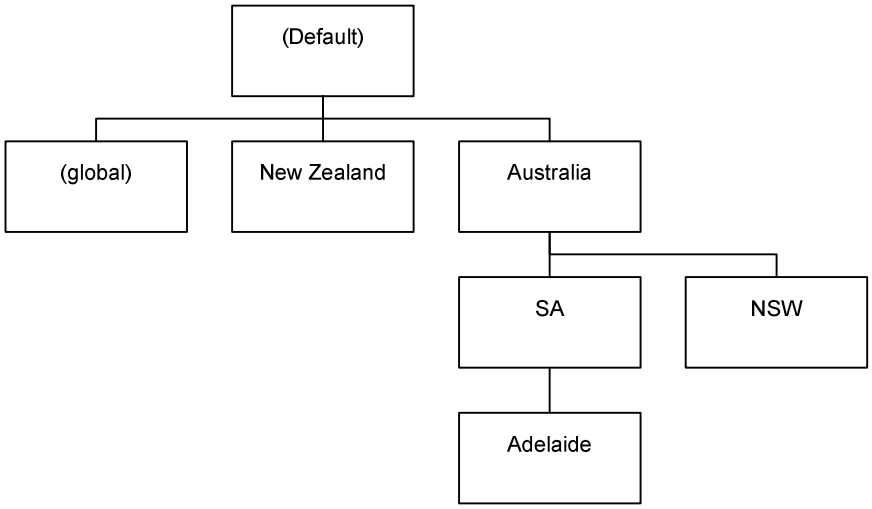
Hi team QESTLab (and for non-QESTLabbers who may be interested in database designs, see a bit further below…),

I've made some more changes to QESTLab dev which make security lookups about as fast as they can be; in fact to me it feels faster than before, but you be the judge. Logging in, traversing the tree and adding work orders, samples & documents should all be really fast. Some areas, such as opening users in the QLA, still do a bit of unnecessary looping, but such places should not be critical.

I'm on leave on Wednesday and Thursday, so either try it out soon or hold off until Friday if you want help when it breaks. There's a reasonable chunk of code that's gone in.

The changes
You may want to pay attention to this, as the key tables - Roles, Activities and Laboratory – are now, let's say, more *sensitive* than before. Also the changes are very interesting.

Consider this basic tree:

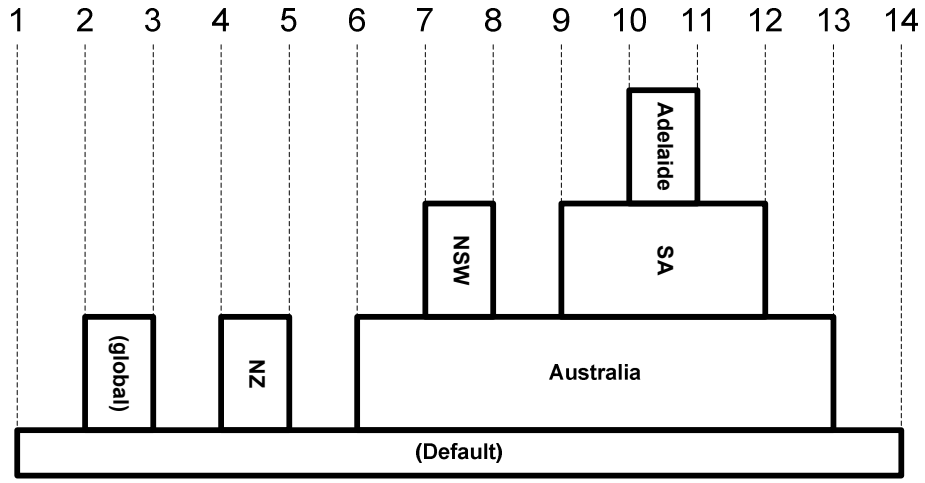


QESTLab tables always implement this hierarchy by having each row store both the ID and the parent ID. This is known as the **adjacency list model**:

Name	QestUniqueID	QestUniqueParentID
(Default)	1	NULL
(global)	2	1
New Zealand	3	1
Australia	4	1
SA	5	4
NSW	6	4
Adelaide	7	5

The problem with this model is that if you want to traverse the tree, you have to use recursion and/or looping, either up or down the tree as you require. Every security operation in the new model requires just that, traversing at least 3 different tables every time. This is a bit expensive and a bit complicated to code in both VB6 and SQL, and as the trees grow and get deeper, it gets slower and slower.

One alternative I've found is called the **nested sets model**. Basically, imagine all the data as sets which contain other sets, and then slice the entire domain space into unique values to define the boundaries:



Notice that we can then define every record by a unique left value and a unique right value, like this:

Name	Left	Right
(Default)	1	14
(global)	2	3
New Zealand	4	5
Australia	6	13
SA	9	12
NSW	7	8
Adelaide	10	11

The really beautiful thing about this model is that now you can query the entire hierarchy without using any recursion or looping, simply by doing a self-join and using the Left and Right values as parameters.

So to get Australia and all of the locations in Australia (Australia, NSW, SA, Adelaide):

```
SELECT l1.Name
FROM Laboratory l1
      INNER JOIN Laboratory l2 ON (l1.Lft BETWEEN l2.Lft AND l2.Rgt)
WHERE l2.Name = 'Australia'
```

And to go the other way, to get, say, Adelaide and all the ancestors (Adelaide, SA, Australia, (Default)):

```
SELECT l2.Name
FROM Laboratory l1
      INNER JOIN Laboratory l2 ON (l1.Lft BETWEEN l2.Lft AND l2.Rgt)
WHERE l1.Name = 'Adelaide'
```

See how similar the queries are, and how simple? Plus, the Left and Right values are all uniquely indexed, which means the lookups will always perform super-fast, and it doesn't become noticeably slower as the tables grow.

You can also work out the level in the hierarchy really easily as well:

```
SELECT l1.Name, COUNT(l2.Name) as [Level]
FROM Laboratory l1
      INNER JOIN Laboratory l2 ON (l1.Lft BETWEEN l2.Lft AND l2.Rgt)
GROUP BY l1.Name
```

Wonderful, isn't it?

So the three core tables (Activities, Laboratory, Roles) all now have three extra columns: Lft, Rgt and Lvl.

For fast tree lookups, this model is definitely the way to go. The only downside is that deleting, inserting and moving nodes becomes more complicated and CPU-expensive, because you have to change the Left and Right values of many, maybe most, other records in the table. So it's not so great if the tree are frequently updated, which luckily QESTLab security is not. I've written stored procs to manage these operations.

This model will operate in parallel with the old model; the QESTLab object model will keep using the old model, while most permissions lookups will use the new model via stored procedures. You can insert and move nodes using the stored procs I wrote:
qest_AddUpdateActivity – used by QESTLab internally, better to avoid it in everyday use
qest_UpdateActivity
qest_MoveActivity
qest_AddRole
qest_UpdateRole
qest_MoveRole
qest_AddLocation
qest_MoveLocation

No stored procs for deletion yet, since the tables all use an Inactive flag instead. Except regions… I will have to correct that shortly.

Other important stored procs:
qest_NestedSetActivities }
qest_NestedSetRoles } --- run these if you accidentally mess up the nested sets hierarchy and need to rebuild it!
qest_NestedSetLaboratory }
qest_GetPermission – get the raw permission integer bitmask for a given activity, location and person
qest_IsPermitted – determine if an activity for a given location and person is valid
qest_GetChildDocuments –used to build the document tree based on various parameters; used in Connection. Hidden_GetObjectTypeList. In the end I realised I didn't need to write this, but it's slightly faster than the existing version so I kept it.

If you want to look at the underlying SQL, it's all scripted in $/Product Development/QEST Lab/Database Template/DynamicScripts_install.sql.

Changes have all been checked in (hopefully). Good luck to whoever goes first!

More info: http://www.intelligententerprise.com/001020/celko.jhtml
http://www.sqlservercentral.com/articles/SQL+Server+2005+-+TSQL/recursivequeriesinsql1999andsqlserver2005/1846/, see section 6

Gavin Schultz
SOFTWARE ENGINEER
direct +61 8 8360 8916 | mobile +61 4 0579 9437 | gavin.schultz@spectraqest.com

spectraqest

ASIA PACIFIC / HEAD OFFICE
Suite 7-9, 132 O'Connell Street | North Adelaide SA 5006 | Australia
+61 8 8360 8900 | fax +61 8 8360 8988 | help desk +61 8 8360 8999
www.spectraqest.com.au

NORTH AMERICA
3540 W Sahara Avenue | Suite 202 | Las Vegas NV 89102 | United States of America
toll free +1 888 360 8901 | fax +1 888 8360 8988
www.spectraqest.com

between paper and electronic versions the paper version is to take precedence. No warranty is made that this email and its contents are free from computer viruses or other defects.