How to salvage referential integrity with mutiple databases

Asked 16 years, 2 months ago Modified 9 years, 2 months ago

Viewed 451 times

2

I am in the middle of designing a system that will be used to feed several production sites around the country (all information is in one site) with the potential to add more. Initially I thought that I could get away with only using one database. I am now re-thinking my original design and leaning toward a more scalable solution. Keeping down the size of each database / tables is also important.

There will be a "master" database that has information that spans the notion of a site and then a separate database for each site with site-specific information in it.

My struggle is where to separate the data. The data is all fairly related. No matter where I do it I will lose some referential integrity. Everything I've read says to avoid this at all costs for what I think are very good reasons, but I don't see a way around it.

I have looked into triggers, but I don't think that they work if the databases are on separate servers (not sure though - I think Oracle does this). I am limited to an open source solution so it'll be MySQL or postgre if that helps at all.

Does anybody have some suggestions to mitigate this problem or have another design suggestion?

database-design integrity

Share

Improve this question

Follow

edited Oct 23, 2015 at 18:19

Brian Tompsett - 汤莱恩

5,875 ● 72 ● 61 ● 133

asked Oct 21, 2008 at 22:39

Chris Kloberdanz

4,526 ● 4 ● 32 ● 31

5 Answers

Sorted by: Highest score (default)

1

Without knowing more about your specific situation, it's a little difficult to help - but here's my gut feeling...

I'm guessing that the information that you have suggested should go in your 'Master' database is perhaps more likely to be stable (a low number of changes to the data) than the databases for each site.

Perhaps you could look at a solution where the data in the 'Master' database is also stored in each site's database. Then you could look at some sort of replication system to propagate changes that are made to the master database down to the site databases.

That way, you can still maintain the referential integrity within each site's database.

Share Improve this answer

Follow

answered Oct 21, 2008 at 23:44

Chris Roberts

18.8k ● 12 ● 61 ● 68

> Yeah, explaining everything would be quite the novel. The master database data would change less often mainly in the form of additions. I guess I had not considered replication just because there would be a decent amount to replicate n times. – Chris Kloberdanz Oct 21, 2008 at 23:51

0

MySQL has federated tables, but it's unclear whether foreign key constraints will work across them. I kind of doubt it - but a trigger should.

Otherwise, you have to move your referential integrity up a layer - into the app.

Share Improve this answer

Follow

answered Oct 21, 2008 at 23:37

Mark Brackett

85.6k ● 17 ● 111 ● 155

0

How much data are you talking about? Do you really need this architecture? DBs can drive a lot of capacity.

The "don't do this" warnings come from hard, bitter experience. And distributed datasets are simply a real

pain to maintain and manage. So, think hard about doing it at all.

Perhaps consider breaking the data up into an operational store vs a reporting store or data warehouse that you can feed nightly or weekly (depending on how current you need your analysis reports). Many operational data stores don't need to be that big.

It's also a different issue about tables that are maintained solely on the back end (say, for data integrity purposes) vs those operational tables that are updated and added to often by users. The more "static" tables can be considered that -- simply static. With a solid procedure in place to update them across your nodes if necessary, and ideally, rarely.

Once your data broken in to your "dynamic" vs "static" tables, partitioning is a bit easier, since your static data can be single mastered and replicated as necessary (from a root instance), while the partioned stores are single sources for truth that are used to feed the back end data warehouse and reporting systems. Then there's little actuall replication necessary, but, rather, more of a "which machine is it on" issue which can be automated readily.

Share Improve this answer

Follow

answered Oct 21, 2008 at 23:54

Will Hartung

118k ● 20 ● 133 ● 207

possibly hundreds of GB's of data. It's the anticipation of growth that's driving me to this idea. The largest tables will

contain data that provide indexes to search for documents on the web and could half a billion records on a Dell. The n + 1 acquired company might break the camel's back.
– Chris Kloberdanz Oct 22, 2008 at 0:14

hundreds of GB isn't that big of a deal dude. I've worked on a dozen terabyte databases over the years.. When I worked at Microsoft, we had 300gb of new data per day. We collected proxy log information for Microsoft global.. We even captured proxy log information from nations where this sort of operation was / is illegal. Microsoft doesn't care about silly things like laws, lol. The punchline is that SQL databases can easily scale to hundreds of GB. – Aaron Kempf Jul 16, 2011 at 19:28

0

If understand you correctly, you want to (maybe) use triggers to check, for every insert/update/delete if the referential integrity is kept on remote databases?

If so, I reckon you should steer clear of this, I just see performance overheads being too much of an issue. Especially if you want the solution to be scaleable.

I would worry about how the data is inserted, and be very strict about it, your app logic should cover this is a high level of detail. You could run weekly reports to see what data is not-correct and see why it gets inserted incorrectly etc, but I think that if your app is done properly, multi-database referential integrity would be difficult to enforce.

But dont get me wrong, I am 100% for keep data in a solid, robust state, but sometimes this is not always enforceable.

But as it was stated earlier, without more info about the solution, its hard to give advice... :)

Share Improve this answer

Follow

answered Oct 22, 2008 at 0:11

Mark

14.9k ● 17 ● 102 ● 160

0

Let me see if I can give a better synopsis for the problem domain:

Looking to create an "enterprise" solution where are n production sites where n WILL increase.

We process data to create documents both web and print.

The system will manage process flow to take a data file from submission (via a centralized web site) to the printer or the web or both.

Each production site has their own customers, etc. All that information would be stored in a database. Most administration of that information would occur at a central site

We process the data all on one server due to licensing restrictions in the software we use.

So there would be a daemon that looks at a queue (in the database) and processes jobs. The flow would be controlled by a status column in the database so that other processes would know where it is in the process.

Where the massive amount of data comes in is for our web tool. We need to store search indexes for each document we produce for the web. This gets rather large rather quickly. These records are not retained forever, but it will be large (estimated 500 million rows) at least most of the time.

I thought that to get rid of the table size issue separate db's could be the answer as well as the ability to separate production sites on different servers.

The thing is I don't know when another site will be acquired or how big it will be.

I guess I want to nip the scalability thing in the bud rather than a year down the road acquire a site that pushes us over the edge and not have to buy a better server to house the monster. Money is, unfortunately, an object.

I would not even consider databases if the growth wasn't an unknown.

I have also considered creating separate databases completely for each site. This makes administration for our apps much harder as well as other issues.

I apologize for the scatterbrained response. It's been a 12 hour day. I really could go on forever, but hopefully that give a little more insight anyways.

An example relationship with one DB

site has many customers customers have many submitters submitters have many submissions submissions have many documents documents have many indexes

So I could count the number of documents for a customer easily with joins

Share Improve this answer Follow

edited Oct 22, 2008 at 0:49

answered Oct 22, 2008 at 0:40

Chris Kloberdanz

4,526 ● 4 ● 32 ● 31