How do you keep two related, but separate, systems in sync with each other?

Asked 16 years, 4 months ago Modified 16 years, 3 months ago

Viewed 12k times

24

My current development project has two aspects to it. First, there is a public website where external users can submit and update information for various purposes. This information is then saved to a local SQL Server at the colo facility.

The second aspect is an internal application which employees use to manage those same records (conceptually) and provide status updates, approvals, etc. This application is hosted within the corporate firewall with its own local SQL Server database.

The two networks are connected by a hardware VPN solution, which is decent, but obviously not the speediest thing in the world.

The two databases are similar, and share many of the same tables, but they are not 100% the same. Many of the tables on both sides are very specific to either the internal or external application.

So the question is: when a user updates their information or submits a record on the public website, how do you transfer that data to the internal application's database so it can be managed by the internal staff? And vice versa... how do you push updates made by the staff back out to the website?

It is worth mentioning that the more "real time" these updates occur, the better. Not that it has to be instant, just reasonably quick.

So far, I have thought about using the following types of approaches:

1. Bi-directional replication
2. Web service interfaces on both sides with code to sync the changes as they are made (in real time).
3. Web service interfaces on both sides with code to asynchronously sync the changes (using a queueing mechanism).

Any advice? Has anyone run into this problem before? Did you come up with a solution that worked well for you?

sql-server **database** **synchronization** **distributed**

Share

Improve this question

Follow

asked Aug 17, 2008 at 2:57

jeremcc

8,733 ● 12 ● 46 ● 55

Hi, I know its an old post, but I am just looking at a very similar scenario. Do you mind me asking What option did you go for in the end? thanks – Sean Nov 11, 2012 at 11:01

5 Answers

Sorted by: Highest score (default)

25

This is a pretty common integration scenario, I believe. Personally, I think an asynchronous messaging solution using a queue is ideal.

You should be able to achieve near real time synchronization without the overhead or complexity of something like replication.

Synchronous web services are not ideal because your code will have to be very sophisticated to handle failure scenarios. What happens when one system is restarted while the other continues to publish changes? Does the sending system get timeouts? What does it do with those? Unless you are prepared to lose data, you'll want some sort of transactional queue (like MSMQ) to receive the change notices and take care of making sure they get to the other system. If either system is down, the changes (passed as messages) will just accumulate and as soon as a connection can be established the re-starting server will process all the queued messages and catch up, making system integrity much, much easier to achieve.

There are some open source tools that can really make this easy for you if you are using .NET (especially if you

want to use MSMQ).

1. nServiceBus by Udi Dahan
2. Mass Transit by Dru Sellers and Chris Patterson

There are commercial products also, and if you are considering a commercial option see here for a list of of options on .NET. Of course, WCF can do async messaging using MSMQ bindings, but a tool like nServiceBus or MassTransit will give you a very simple Send/Receive or Pub/Sub API that will make your requirement a very straightforward job.

If you're using Java, there are any number of open source service bus implementations that will make this kind of bi-directional, asynchronous messaging a snap, like Mule or maybe just ActiveMQ.

You may also want to consider reading Udi Dahan's blog, listening to some of his podcasts. Here are some more good resources to get you started.

Share Improve this answer

Follow

edited May 23, 2017 at 12:00

Community Bot

1 ● 1

answered Aug 17, 2008 at 3:22

Nathan

12.3k ● 3 ● 30 ● 28

3

I'm mid-way through a similar project except I have multiple sites that need to keep in sync over slow connections (dial-up in some cases).

Firstly you need to track changes, if you can use SQL 2008 (even the Express version is enough if the 2Gb limit isn't a problem) this will ease the pain greatly, just turn on Change Tracking on the database and each table. We're using SQL Server 2008 at the head office with the extended schema and SQL Express 2008 at each site with a sub-set of data and limited schema.

Secondly you need to track your changes, Sync Services does the trick nicely and supports using a WCF gateway into the main database. In this example you will need to use the Sync using SQL Express Client sample as a starting point, note that it's based on SQL 2005 so you'll need to update it to take advantage of the Change Tracking features in 2008. By default the Sync Services uses SQL CE on the clients, which I'm sure isn't enough in your case. You'll need a service that runs on your Web Server that periodically (could be as often as every 10 seconds if you want) runs the Synchronize() method. This will tell your main database about changes made locally and then ask the server for all changes made there. You can set up the get and apply SQL code to call stored procedures and you can add event handlers to handle conflicts (e.g. Client Update vs Server Update) and resolve them accordingly at each end.

Share Improve this answer

Follow

answered Sep 17, 2008 at 1:53

Timothy Walters

16.9k ● 2 ● 43 ● 49

2

Recently I have had a lot of success with SQL Server Service Broker which offers reliable, persisted asynchronous messaging out of the box with very little implementation pain.

- It is quick to set up and as you learn more you can use some of the more advanced features.
- Unknown to most, it is also part of the desktop editions so it can be used as a workstation messaging system
- If you have existing T-SQL skills they can be leveraged as all the code to read and write messages is done in SQL
- It is blindingly fast

It is a vastly under-hyped part of SQL Server and well worth a look.

Share Improve this answer

Follow

answered Sep 2, 2008 at 8:20

Simon Munro

5,419 ● 7 ● 34 ● 40

1

We have a shop as a client, with three stores connected to the same VPN

Two of the shops have a computer running as a "server"

for that shop and the the third one has the "master database"
To synchronize all to the master we don't have the best solution, but it works: there is a dedicated PC running an application that checks the timestamp of every record in every table of the two stores and if it is different that the last time you synchronize, it copies the results
Note that this works both ways. I.e. if you update a product in the master database, this change will propagate to the other two shops. If you have a new order in one of the shops, it will be transmitted to the "master".
With some optimizations you can have all the shops synchronize in around 20minutes

Share Improve this answer

Follow

answered Aug 21, 2008 at 11:52

Romjin

0

I'd say just have a job that copies the data in the pub database input table into a private database pending table. Then once you update the data on the private side have it replicated to the public side. If you don't have any of the replicated data on the public side updated it should be a fairly easy transactional replication solution.

Share Improve this answer

Follow

answered Aug 17, 2008 at 4:07

Ryan

4,662 ● 8 ● 39 ● 43