What are the advantages of using a single database for EACH client?

Asked 16 years, 4 months ago Modified 15 years, 3 months ago

Viewed 22k times

69

In a database-centric application that is designed for multiple clients, I've always thought it was "better" to use a single database for ALL clients - associating records with proper indexes and keys. In listening to the Stack Overflow podcast, I heard Joel mention that FogBugz uses one database per client (so if there were 1000 clients, there would be 1000 databases). What are the advantages of using this architecture?

I understand that for some projects, clients need direct access to all of their data - in such an application, it's obvious that each client needs their own database. However, for projects where a client does not need to access the database directly, are there any advantages to using one database per client? It seems that in terms of flexibility, it's much simpler to use a single database with a single copy of the tables. It's easier to add new features, it's easier to create reports, and it's just easier to manage.

I was pretty confident in the "one database for all clients" method until I heard Joel (an experienced developer)

mention that his software uses a different approach -- and I'm a little confused with his decision...

I've heard people cite that databases slow down with a large number of records, but any relational database with some merit isn't going to have that problem - especially if proper indexes and keys are used.

Any input is greatly appreciated!

database database-design multi-tenant

Share

Improve this question

Follow

edited Sep 3, 2009 at 11:09

hlovdal

28k ● 11 ● 100 ● 175

asked Aug 16, 2008 at 20:39

Ryan

955 ● 1 ● 10 ● 11

3 See also One Database vs. Multiple Databases in serverfault. – Örjan Jämte Jul 6, 2011 at 6:32

8 Answers

Sorted by: Highest score (default)

55

Assume there's no scaling penalty for storing all the clients in one database; for most people, and well configured databases/queries, this will be fairly true these




days. If you're not one of these people, well, then the benefit of a single database is obvious.

In this situation, benefits come from the encapsulation of each client. From the code perspective, each client exists in isolation - there is no possible situation in which a database update might overwrite, corrupt, retrieve or alter data belonging to another client. This also simplifies the model, as you don't need to ever consider the fact that records might belong to another client.

You also get benefits of separability - it's trivial to pull out the data associated with a given client ,and move them to a different server. Or restore a backup of that client when the call up to say "We've deleted some key data!", using the builtin database mechanisms.

You get easy and free server mobility - if you outscale one database server, you can just host new clients on another server. If they were all in one database, you'd need to either get beefier hardware, or run the database over multiple machines.

You get easy versioning - if one client wants to stay on software version 1.0, and another wants 2.0, where 1.0 and 2.0 use different database schemas, there's no problem - you can migrate one without having to pull them out of one database.

I can think of a few dozen more, I guess. But all in all, the key concept is "simplicity". The product manages one client, and thus one database. There is never any

complexity from the "But the database also contains other clients" issue. It fits the mental model of the user, where they exist alone. Advantages like being able to doing easy reporting on all clients at once, are minimal - how often do you want a report on the whole world, rather than just one client?

Share Improve this answer

Follow

answered Aug 16, 2008 at 20:48

Adam Wright

49.3k ● 12 ● 133 ● 152

3 In terms of keeping the "wall" between clients, that's what stored procedures and triggers are for (warning: not recommended for MySQL)--it is also trivially easy to (re)move clients' data to different servers if one has built the schema properly. "Simplicity" works the other way, too. If I have a single database, I can easily pool my connections and simplify that code. If I run out of connections, I just increase the pool; no need to monitor each client separately. – BryanH Oct 19, 2009 at 17:48

16

Here's one approach that I've seen before:

- Each customer has a unique connection string stored in a master customer database.
- The database is designed so that everything is segmented by CustomerID, even if there is a single customer on a database.
- Scripts are created to migrate all customer data to a new database if needed, and then only that

customer's connection string needs to be updated to point to the new location.

This allows for using a single database at first, and then easily segmenting later on once you've got a large number of clients, or more commonly when you have a couple of customers that overuse the system.

I've found that restoring specific customer data is really tough when all the data is in the same database, but managing upgrades is much simpler.

When using a single database per customer, you run into a huge problem of keeping all customers running at the same schema version, and that doesn't even consider backup jobs on a whole bunch of customer-specific databases. Naturally restoring data is easier, but if you make sure not to permanently delete records (just mark with a deleted flag or move to an archive table), then you have less need for database restore in the first place.

Share Improve this answer

Follow

answered Aug 16, 2008 at 23:14

The How-To Geek

1,105 ● 9 ● 13

12

To keep it simple. You can be sure that your client is only seeing their data. The client with fewer records doesn't have to pay the penalty of having to compete with hundreds of thousands of records that may be in the database but not theirs. I don't care how well everything

is indexed and optimized there will be queries that determine that they have to scan every record.

Share Improve this answer

Follow

answered Aug 16, 2008 at 22:52

bruceatk

5,138 ● 2 ● 27 ● 36

1 This should not be the case. If you can't use indexes to isolate rows, either the database is poorly designed, or you're trying to query the entire cross-client data-set, which would be harder to do with separate databases to begin with. – Nick Feb 19, 2019 at 16:51

11

Well, what if one of your clients tells you to restore to an earlier version of their data due to some botched import job or similar? Imagine how your clients would feel if you told them "you can't do that, since your data is shared between all our clients" or "Sorry, but your changes were lost because client X demanded a restore of the database".

Share Improve this answer

Follow

answered Aug 16, 2008 at 20:45

Lasse V. Karlsen

391k ● 106 ● 646 ● 844

2 All tables should be partitioned by TenantID. This gives you possibility to backup/restore partitions for only one client :) – dariol May 20, 2010 at 23:03

@dario-g: Sounds like you're supporting a shared database approach. Please explain what you mean by partitioning by

TenantID, as this may not be obvious. – Gruber Sep 21, 2012 at 14:56

1 In Oracle you can partition a table and move the partitions to remote locations, however you still have a single table. – givanse Feb 6, 2014 at 18:18

10

As for the pain of upgrading 1000 database servers at once, some fairly simple automation should take care of that. As long as each database maintains an identical schema, then it won't really be an issue. We also use the database per client approach, and it works well for us.

Here is an article on this exact topic (yes, it is MSDN, but it is a technology independent article): http://msdn.microsoft.com/en-us/library/aa479086.aspx.

Another discussion of multi-tenancy as it relates to your data model here: http://www.ayende.com/Blog/archive/2008/08/07/Multi-Tenancy--The-Physical-Data-Model.aspx

Share Improve this answer

Follow

answered Aug 16, 2008 at 23:18

Nathan

12.3k ● 3 ● 30 ● 28

2 Anybody knows a tool which automates upgrading multiple databases for say, MySQL? If you decide to use the isolation approach you need to make sure that the 1000 databases are up to date, mirroring the schema of a designated master database. – Gruber Sep 21, 2012 at 14:50

1 @Nathan, are you still using one db per tenant? – jonathancardoso Oct 17, 2016 at 16:12

3 No, ended up going with one multi-tenant database and putting tenant-privacy into the app framework. The one database per tenant was too much overhead for us. – Nathan Oct 24, 2016 at 16:18

6

Scalability. Security. Our company uses 1 DB per customer approach as well. It also makes code a bit easier to maintain as well.

Share Improve this answer

Follow

answered Aug 16, 2008 at 20:48

Darren Kopp

77.6k ● 9 ● 80 ● 93

2 Hey, are you still using that approach? Would you mind sharing some numbers about how many clients/db you currently have? – jonathancardoso Jul 12, 2016 at 17:24

3

In regulated industries such as health care it may be a requirement of one database per customer, possibly even a separate database server.

The simple answer to updating multiple databases when you upgrade is to do the upgrade as a transaction, and take a snapshot before upgrading if necessary. If you are running your operations well then you should be able to apply the upgrade to any number of databases.

Clustering is not really a solution to the problem of indices and full table scans. If you move to a cluster, very little changes. If you have have many smaller databases to distribute over multiple machines you can do this more cheaply without a cluster. Reliability and availability are considerations but can be dealt with in other ways (some people will still need a cluster but majority probably don't).

I'd be interested in hearing a little more context from you on this because clustering is not a simple topic and is expensive to implement in the RDBMS world. There is a lot of talk/bravado about clustering in the non-relational world Google Bigtable etc. but they are solving a different set of problems, and lose some of the useful features from an RDBMS.

Share Improve this answer

Follow

answered Aug 16, 2008 at 23:31

Brian Lyttle

14.6k ● 15 ● 69 ● 106

0

There are a couple of meanings of "database"

- the hardware box
- the running software (e.g. "the oracle")
- the particular set of data files
- the particular login or schema

It's likely Joel means one of the lower layers. In this case, it's just a matter of software configuration management...

you don't have to patch 1000 software servers to fix a security bug, for example.

I think it's a good idea, so that a software bug doesn't leak information across clients. Imagine the case with an errant where clause that showed me your customer data as well as my own.

Share Improve this answer

Follow

answered Aug 16, 2008 at 22:46

Mark Harrison

304k ● 131 ● 350 ● 489

6 No. he means the database. The thing that gets created when you issue a "CREATE DATABASE" statement.
– BobbyShaftoe Dec 15, 2008 at 6:25