One DB per developer or not?

Asked 16 years, 2 months ago Modified 5 years, 7 months ago

Viewed 6k times

35

In a corporate development environment writing mostly administrative software, should every developer use their own database instance, or should they use a central database instance during development? What are the advantages and disadvantages of each approach? What about other environments and other products?

database development-environment

Share

Improve this question

Follow

edited Oct 25, 2008 at 3:04

Mitch Wheat

300k ● 44 ● 477 ● 550

asked Oct 24, 2008 at 17:46

Pablo Cabrera

5,839 ● 5 ● 25 ● 28

That entirely depends on what you're developing, honestly. Is it a DB schema for a product? Is it for holding "dummy" data while testing? More info will definitely be needed to answer this question well :) – warren Oct 24, 2008 at 17:47

The question could also be "One DB per new feature?" - if more than one developer wants to contribute to the same change set, it is more efficient to have a 'central' instance

dedicated to this feature request. This also has the advantage that a developer can contribute to two ore more features at the same time, but keeping them in separate databases. – mjn Jan 22, 2012 at 10:54

19 Answers

Sorted by: Highest score (default)

39

If you all share the same database, you might have some issues if someone make a structure change to the database and that the code is not "Synchronized" with it.

I highly recommend one DB per developer for the only reason that you don't want to do "write" test to see someone else override you right after. A simple exemple? You try to display product for a website. Everything works until all the products disappear. Problem? Another developer decided to play with the "Active" flag of the product to test something else. In cases like that, a transaction might not even work. End of the story, you spend time debugging for someone else action.

I highly recommend replicating the staging database to the developer database once in a while to synchronize the structure (or better, have a tool to rebuild a database from scratch).

Of course, we require scripts for changes to the database and EVERYTHING is in a Source Control.

Share Improve this answer Follow

answered Oct 24, 2008 at 17:50

Maxime Rouiller
13.7k ● 9 ● 60 ● 108

I usually have two DBs per application on my local system, one to develop and run the local web server against, and the other for automated integration tests. – tvanfosson Oct 25, 2008 at 2:47

If multiple developers share the same database, data conflicts may occur while running the developing versions, may interfere other developers. – linquize Feb 8, 2013 at 5:35

17

The days when database environments should be scarce are long gone. I'm writing this posting on a XW9300 with 5x15k SCSI disks in it. This machine will run a substantial ETL job in a fairly reasonable length of time and (in mid-2007) cost me about £1,700 on ebay including the disks. From a developer's perspective, especially on database centric projects like data warehousing, the line between a developer and a DBA is quite blurred. As I write this I am building a partition management framework for a SQL Server 2005 data warehouse.

Developers should have one or more development databases of their own for (IMO) these reasons:

- Requires people to keep stored procedures, patch scripts and schema definition files in source control. Applying the patches can be automated to a fairly large extent. There are even tools such as Redgate

SQL Compare Pro that do much of the grunt work for this.

- Encourages an application architecture that facilitates easy configuration management and deployment, as people have to deploy onto their own workstations. Many deployment wrinkles will get sorted out long before they hit production or people even realise they could have gone wrong.
- Avoids developers tripping up on each other's work. On something like a data warehouse where people are working with ETL code this is an even bigger win.
- It encourages a degree of responsibility as developers have to learn basic database administration. This also eliminates a lot of the requirements for operational support staff and some of the dev-vs. ops friction.
- If you have your own database, there are no gatekeepers obstructing experimentation or other work on it. The politics around managing 'servers' disappear as there are no 'servers'. This is a productivity win in an any environment with significant incumbent bureaucracy.
- For small data volumes an ordinary PC is fast enough for this. Developer editions or licencing are available for most if not all database management systems and will run on a desktop O/S. If you're working with Linux or Unix this is even less of an issue. For larger data volumes, up to and including most MIS applications, a workstation like an HP

XW9400 or Lenovo D10 can be outfitted with 5 15k disks for less than the cost of a lot of professional development tooling. (Yes, I know it's dual licence, but a commercial all-platform licence for QT is about £4000 a seat). A machine like this will run an ETL process with 10's to 100's of millions of rows faster than you might think.

- It facilitates setting up more than one environment for smoke testing or reconciliation purposes. As you have complete control over the machine, you have quite a lot of scope for mocking up conditions in a production environment. For example, I once made a simple emulator for Control-M by just bodging some of its runtime scripts. Where you have this level of control and transparency over the environment you can produce a fairly robustly tested deployment process which does quite a lot to eliminate opportunities for finger-pointing in production deployment.

I've seen small teams working with 14 environments, and had 7 active on a workstation at the same time. On database heavy work such as ETL, where you're with with whole tables, working in a single dev environment is a recipe for time wastage or spending your time walking on eggshells.

Also, you can use single user development licences for database platforms, which can save you the cost of the workstations just in database licencing. Most developer licences (Microsoft and OTN are a couple of examples

I'm familiar with) will let you use the system on a single workstation for a single developer free or for a nominal price.

Conversely, licencing terms on shared development servers are often somewhat murky and I've seen vendors try to shake customers down for licencing on dev servers on more than one occasion.

Share Improve this answer

Follow

edited May 15, 2019 at 2:01

Laurel

6,163 ● 14 ● 33 ● 59

answered Oct 24, 2008 at 18:13

ConcernedOfTunbridge Wells

66.5k ● 15 ● 148 ● 198

7

Each of our developers has a fully functional database. Changes are scripted and source controlled like any other code.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:48

dacracot

22.3k ● 28 ● 109 ● 159

5

My recommendation is to have 2 levels of development environment:

Each developer has their own personal development system, with its own dp, web servers, etc. This allows them to code against a known setup, write automated (system level) tests that initialize their database and systems to a known state, etc.

The development integration environment is shared by all developers and used to make sure everything is working together as expected before handing it off to QA. Code is checked out from source control and installed there, and there's only a single instance of any servers (db or otherwise).

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:55

RHSeeger

16.3k ● 7 ● 54 ● 41

5

Ideally, yes, each developer should have a "sandbox" development environment, so they can test their code even before deploying it to a shared testing/staging environment.

Each developer's environment should run scripted tests that reset the database to a known state. This is impossible to do in a shared environment.

The cost of giving each developer their own instance is less than the cost of the chaos resulting from multiple developers trying to test volatile changes together in a shared environment.

On the other hand, in many IT shops the system uses complex infrastructure, involving multiple application servers or multiple physical nodes. Then the economics change; it's less expensive for people to cooperate and avoid stepping on each other's work than it would be to replicate it for each developer. Especially true if you integrate expensive third-party systems that don't give you licenses for multiple development environments.

So the answer is yes and no. :-) Do give each developer their own environment if that environment can be reproduced inexpensively.

Share Improve this answer

Follow

answered Oct 25, 2008 at 2:20

Bill Karwin

561k ● 87 ● 698 ● 854

3

This question hints at what a developer needs to do his/her job. Certainly a private DB instance should be provided. Equally important, I would make sure that the DB is the same product/version as what you intend to deploy to. Don't develop on MySQL 6.x and deploy to MySQL 5.x. (This goes for app servers, and web servers as well!)

Having a developer DB doesn't necessarily ean you need it hosted on your local machine. You could have a central DBMS host machine with all dev dbs located on it. The pros are the garauntee that you develop against the target DB. Less overhead on dev boxes, more space/horsepower for beefy IDEs and app servers. The

cons are single point of failure for all devs. (The DBMS server goes down nobody can work.) Lack of dev exposure to setting up and administering the DBMS. Devs cannot experiment as easily with upcoming DB releases or alternate DB choices to solve tough problems.

Some of the pros can be cons and vice-versa depending on your organization and structure. Maybe you don't want devs administering the DBMS. Maybe you do plan to support varying DB platforms. The decision boils down to your organization as well as your target platform choices. If you plan to target a variety of DB/OS/app server combinations then each dev should not only have their own DB but should work in a unique combination. (MySQL/Tomcat/OSX for one DB2/Jetty/Linux for another Postegres/Geronimo/WinXP for a 3rd, etc.) If you setup an ASP (Application Service Provider) type shop on an iSeries on the other hand then of course you'll likely have a central host with all dev dbmses still each dev should have at least a separate db instance to allow structural changes to schema.

edited Nov 23, 2011 at 21:26

answered Oct 25, 2008 at 3:17

Cliff
11.2k ● 7 ● 69 ● 103

This answer is good, but it would be easier to read if it had line breaks. – mo. Nov 22, 2011 at 21:00

1

I have an instance of SQLServer Development Edition installed locally. We have a QA DB server, as well as multiple production servers. All development and integration testing is done using my local server (or other developers local servers). New releases are staged to the QA server. Each release, after acceptance by the customer, is put into production.

Since I mostly do web development, I use the web server bundled with VS2008 for development and local test, then publish the web app to a QA web server hosted on a VM. Once accepted by the customer, it is published to one of several different production web servers -- some virtual, some not, depending on the application.

Share Improve this answer Follow

edited Oct 25, 2008 at 3:05

answered Oct 25, 2008 at 2:46

tvanfosson

532k ● 102 ● 699 ● 798

1

My department at my company only has limited development environments, purely because of cost of support and hardware. We have a couple of

environments which are based on t-1 nightly refreshes from production, and some static ones.

Ideally, everyone should have their own, but in many cases, this is going to be impractical when the following are true:

- you have a large number of developers needing resources (our department has maybe 80)
- each developer needs multiple resources (typically i use 4-5 different dbs each day)
- up to date data is important (you just cant refresh them fast enough)

In these cases, shared instances and good communication are whats needed.

Share Improve this answer

Follow

answered Oct 25, 2008 at 6:41

Simon P

1,206 ● 1 ● 14 ● 26

0

One advantage to one database per developer, each developer has a snapshot of their own data in a "known" state.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:53

Mat Nadrofsky

8,274 ● 8 ● 51 ● 73

0

I like the idea of using a local version when a developer *must* be isolated - developing a schema change, performance testing, setting up specific scenarios, etc...

At other times use the shared version as to insure everything is in sync with each other.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:56

Aardvark

8,561 ● 7 ● 47 ● 64

0

I think there's a terminology problem here. It's been a while since I've worn my DBA hat (golly gee, almost 10 years) - so someone else can chime in and correct me.

I think everyone is in agreement that each developer should have his own sandbox schema set.

In MySQL and Sybase/MS SQLServer, each database engine can support multiple databases. Each database is (normally) fully independent of the other database. So you can have one database engine instance, and give each developer his database space to do as he wish. the only problem is if the developers are using tempdb -- there can be collisions there (I think -- this you will need to look up). Just be careful that cross-database queries with fixed database names are not used.

In Oracle, the database engine instance is tied to a particular schema set. If you have multiple developers on

the same engine, they are all pointing to the same tables. In this case, yes, you will need to run multiple instances.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:58

Toybuilder

11.5k ● 5 ● 30 ● 34

0

Each of our developers has a local database. We store the create script AND a dump of the "standard data" in our SVN repo. We have an extensive set of tests that must pass against this test data. We also have a "sandbox" database that is available for people to put data in that they want shared into the standard data. This works well for us and allows us to let developers modify their local copies of data to test things, but we control what gets passed to other developers. We also strictly control schema changes, so we don't encounter the problems that someone else mentioned.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:58

Joey Gibson

7,104 ● 1 ● 22 ● 14

0

It really depends on the nature of your application. If yours is a client-server architecture in a distributed environment, it is best to have a central database that everyone uses. If the product gives users an environment with local database instances, you can use that. It is best if your development mirrors the real world environment as closely as possible.

It is also dependent on what stage of development you are in. Probably in the early stages, you dont want to get bogged down by connectivity, network and distributed environment issues and just want to be up and running. In such a case, you can start with a database instance-per-user model before switching to the central model as the product reaches some level of maturity.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:55

Ather

1,600 11 17

0

In my company we tend to copy the entire DB when working on non-trivial new features. The reasoning there is disk space is cheap, whereas accidental data loss (even if it's test data) isn't.

Share Improve this answer

Follow

answered Dec 11, 2008 at 17:17

user42092

0

I've worked in both types of development environments. Personally, I prefer to have my own DB/app server. However, there may be some advantages to using a shared infrastructure for development.

The main one is that a shared environment more closely resembles a real-world scenario: you are more likely to uncover problems with locking or transactions when all

developers share a DB. Giving each developer their own DB may lead to "it works on my DB" syndrome.

However, if you need to apply and test schema changes or optimisations, then I can see problems in this sort of set-up.

Maybe a compromise solution would work best: all developers share infrastructure, and if someone needs to test schema changes, they create their own temporary DB instance (maybe there is one just sitting there for this purpose?) until they are happy to commit the new schema to source control.

You do have your entire schema (and test data) in source control, right? Right???

Share Improve this answer

Follow

answered Mar 3, 2009 at 20:09

KarstenF

5,631 ● 3 ● 22 ● 16

0

I like the compromise solution (all developers share infrastructure, and if someone needs to test schema changes, they create their own temporary DB instance (maybe there is one just sitting there for this purpose?) until they are happy to commit the new schema to source control.)

Share Improve this answer

Follow

edited Jun 16, 2009 at 8:09

Marc Gravell

1.1m ● 273 ● 2.6k ● 3k

answered Jun 15, 2009 at 7:09

Catherine Sea

0

One DB per developer. No question. But the issue really is how to script entire databases, "control data", and version them. My solution is here : http://dbsourcetools.codeplex.com/ Have fun. - Nathan.

Share Improve this answer Follow

answered Jul 7, 2009 at 13:11

Nathan Rozentals

0

The database schemas should be held in source control and developers should own the changesets checked in for code and db together. Prior to checkin the developer should be working on his own database. After checkin, an automated build (eg: on checkin, nightly, etc), should update a central integrated db, along with the apps themselves.

At developer instance level the data loaded should be appropriate for unit testing, at least. At integrated level, the shared db should hold data also appropriate for testing, but should not rely on production replication - this is just a slack substitute for managed test data.

In my experience the only reason that developers opt for a shared db is that they believe that developing and running on recent production data is somehow 'real' and

means that they can put less effort into testing. They prefer to tread on each others toes and put up with a shared db that slowly corrupts before the next production refresh than write and manage proper tests. It's this kind of management practice that gives the IT world the poor reputation to deliver that it currently has.

Share Improve this answer

Follow

answered Jan 25, 2012 at 6:27

Sentinel

3,677 ● 2 ● 34 ● 45

-1

I'd suggest to use one instance of the database. You don't want your database to be a moving target.

Share Improve this answer

Follow

answered Oct 24, 2008 at 17:51

community wiki

Aaron Palmer