Storing Images in DB - Yea or Nay?

Asked 15 years ago Modified 11 years, 7 months ago Viewed 817k times

415 votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

So I'm using an app that stores images heavily in the DB. What's your outlook on this? I'm more of a type to store the location in the filesystem, than store it directly in the DB.

What do you think are the pros/cons?

database image theory storage blob

Share edited Nov 28, 2008 at 5:41 community wiki
8 revs, 5 users 57%
James Hall

Comments disabled on deleted / locked posts / reviews | Show 1 more comment

56 Answers

Sorted by: Highest score (default)

1 2 Next

350 votes

I'm in charge of some applications that manage many TB of images. We've found that storing **file paths** in the database to be best.

There are a couple of issues:

- database storage is usually more expensive than file system storage
- you can super-accelerate file system access with standard off the shelf products
  - for example, many web servers use the operating system's **sendfile()** system call to asynchronously send a file directly from the file system to the network interface. Images stored in a database don't benefit from this optimization.
- things like web servers, etc, need no special coding or processing to access images in the file system
- databases win out where transactional integrity between the image and metadata are important.
  - it is more complex to manage integrity between db metadata and file system data
  - it is difficult (within the context of a web application) to guarantee data has been flushed to disk on the filesystem

Share edited Nov 20, 2010 at 17:25 community wiki

6 revs, 4 users 64%
Mark Harrison

33 what off the shelf products are available for "super-accelerating" the file system? – Andrei Rînea Oct 4, 2008 at 10:53

22 While I only manage 3TB of files, I definitely agree. Databases are for structured data, not blobs. – derobert Mar 22, 2009 at 19:31

7 @derobert: quite so, if you will never use a data element in a query, as a condition or for a join, it probably doesn't belong in the database. Then again, if you have a nice database function to query images for likeness... – Nils Weinander May 18, 2009 at 14:34

14 what off the shelf products are available for "super-accelerating" the file system? – faceclean Jul 31, 2009 at 15:16

5 Re: "super-accelerating" products: Most web servers can now take advantage of the sendfile() system call to deliver static files asynchronously to the client. It offloads to the operating system the task of moving the file from disk to the network interface. The OS can do this much more efficiently, operating in kernel space. This, to me, seems like a big win for file system vs. db for storing/serving images. – Alan Donnelly Nov 20, 2010 at 17:07

Show **9** more comments

140 votes

As with most issues, it's not as simple as it sounds. There are cases where it would make sense to store the images in the database.

- You are storing images that are changing dynamically, say invoices and you wanted to get an invoice as it was on 1 Jan 2007?
- The government wants you to maintain 6 years of history
- Images stored in the database do not require a different backup strategy. Images stored on filesystem do
- It is easier to control access to the images if they are in a database. Idle admins can access any folder on disk. It takes a really determined admin to go snooping in a database to extract the images

On the other hand there are problems associated

- Require additional code to extract and stream the images
- Latency may be slower than direct file access
- Heavier load on the database server

Share

edited Mar 31, 2011 at 22:43

community wiki
2 revs, 2 users 97%
Rad

2 Not having a seperate backup strategy can be a big deal when you are writing applications that are installed on premise (like SharePoint). When you create a SharePoint backup everything is in the DB which makes it very easy. – Eric Schoonover Oct 2, 2008 at 23:40

44 Security by obscurity is not really an access control strategy! – Jon Cage Oct 9, 2008 at 10:46

5 I don't think he's advocating security by obscurity - he's saying that putting images in the DB adds another

layer of security. (I think... @Conrad, don't want to put words in your mouth) – AJ. Oct 7, 2010 at 9:06

Show **3** more comments

99 votes

File store. Facebook engineers had a great talk about it. One take away was to know the practical limit of files in a directory.

Needle in a Haystack: Efficient Storage of Billions of Photos

Share

answered Aug 20, 2008 at 18:35

community wiki
jason saldo

Show **1** more comment

56 votes

This might be a bit of a long shot, but if you're using (or planning on using) SQL Server 2008 I'd recommend having a look at the new FileStream data type.

FileStream solves most of the problems around storing the files in the DB:

1. The Blobs are actually stored as files in a folder.
2. The Blobs can be accessed using *either* a database connection *or* over the filesystem.
3. Backups are integrated.
4. Migration "just works".

However SQL's "Transparent Data Encryption" does not encrypt FileStream objects, so if that is a consideration, you may be better off just storing them as varbinary.

From the MSDN Article:

> Transact-SQL statements can insert, update, query, search, and back up FILESTREAM data.
> Win32 file system interfaces provide streaming access to the data.
> FILESTREAM uses the NT system cache for caching file data. This helps reduce any effect that FILESTREAM data might have on Database Engine performance. The SQL Server buffer pool is not used; therefore, this memory is available for query processing.

Share

edited Jul 26, 2011 at 21:47

community wiki
3 revs, 2 users 70%
John Gietzen

Show **3** more comments

39 votes

File paths in the DB is **definitely** the way to go - I've heard story after story from customers with TB of images that it became a nightmare trying to store any significant amount of images in a DB - the performance hit alone is too much.

Share

edited Aug 6, 2008 at 18:17

community wiki
2 revs
Greg Hurlman

Add a comment

35 votes

In my experience, sometimes the simplest solution is to **name the images according to the primary key**. So it's easy to find the image that belongs to a particular record, and vice versa. But at the same time you're not storing *anything* about the image in the database.

Share

answered Aug 6, 2008 at 17:59

community wiki
Patrick McElhaney

6 @Marijn: That's only if you expose the images to the world. – Seun Osewa Nov 26, 2010 at 14:15

Show **3** more comments

31 votes

The trick here is to not become a zealot.

One thing to note here is that no one in the pro file system camp has listed a particular file system. Does this mean that everything from FAT16 to ZFS handily beats every database?

No.

The truth is that many databases beat many files systems, even when we're only talking about raw speed.

The correct course of action is to make the right decision for your precise scenario, and to do that, you'll need some numbers and some use case estimates.

Share

answered Aug 31, 2008 at 17:54

community wiki
dicroce

6 I don't see anyone claiming that a filesystem is faster than a DB 100% of the time (read Mark Harrison's answer). That's a bit of a strawman. There are probably situations in which it's preferable not to wear your seatbelt, but *generally speaking*, wearing a seatbelt is a good idea. – Calvin Apr 8, 2009 at 16:56

Add a comment

30 votes

In places where you MUST guarantee referential integrity and ACID compliance, storing images in the database is required.

You cannot transactionaly guarantee that the image and the meta-data about that image stored in the database refer to the same file. In other words, it is impossible to guarantee that the file on the filesystem is only ever altered at the same time and in the same transaction as the metadata.

Share

edited Sep 3, 2009 at 0:48

community wiki
2 revs
mluebke

7 Actually, no, you can. As long as image files are never deleted, changed or over-written once created, all image files are synced before attempting to commit transactions, there is no filesystem corruption, you can be sure

that image files and metadata are in sync. For some applications, those are too many ifs, I guess. – Seun Osewa Nov 5, 2010 at 1:36

Show **1** more comment

28 votes

As others have said SQL 2008 comes with a Filestream type that allows you to store a filename or identifier as a pointer in the db and automatically stores the image on your filesystem which is a great scenario.

If you're on an older database, then I'd say that if you're storing it as blob data, then you're really not going to get anything out of the database in the way of searching features, so it's probably best to store an address on a filesystem, and store the image that way.

That way you also save space on your filesystem, as you are only going to save the exact amount of space, or even compacted space on the filesystem.

Also, you could decide to save with some structure or elements that allow you to browse the raw images in your filesystem without any db hits, or transfer the files in bulk to another system, hard drive, S3 or another scenario - updating the location in your program, but keep the structure, again without much of a hit trying to bring the images out of your db when trying to increase storage.

Probably, it would also allow you to throw some caching element, based on commonly hit image urls into your web engine/program, so you're saving yourself there as well.

Share answered Aug 30, 2008 at 9:50 community wiki crucible

Add a comment

27 votes

Small static images (not more than a couple of megs) that are not frequently edited, should be stored in the database. This method has several benefits including easier portability (images are transferred with the database), easier backup/restore (images are backed up with the database) and better scalability (a file system folder with thousands of little thumbnail files sounds like a scalability nightmare to me).

Serving up images from a database is easy, just implement an http handler that serves the byte array returned from the DB server as a binary stream.

Share answered Aug 6, 2008 at 18:46 community wiki urini

Show **1** more comment

26 votes

Here's an interesting white paper on the topic.

To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem

The answer is "It depends." Certainly it would depend upon the database server and its approach to blob storage. It also depends on the type of data being stored in blobs, as well as how that data is to be accessed

be accessed.

Smaller sized files can be efficiently stored and delivered using the database as the storage mechanism. Larger files would probably be best stored using the file system, especially if they will be modified/updated often. (blob fragmentation becomes an issue in regards to performance.)

Here's an additional point to keep in mind. One of the reasons supporting the use of a database to store the blobs is ACID compliance. However, the approach that the testers used in the white paper, (Bulk Logged option of SQL Server,) which doubled SQL Server throughput, effectively changed the 'D' in ACID to a 'd,' as the blob data was not logged with the initial writes for the transaction. Therefore, if full ACID compliance is an important requirement for your system, halve the SQL Server throughput figures for database writes when comparing file I/O to database blob I/O.

Share edited Jul 26, 2011 at 21:27 community wiki 2 revs, 2 users 95% user13550

Add a comment

---

25 votes

One thing that I haven't seen anyone mention yet but is definitely worth noting is that there are issues associated with storing large amounts of images in most filesystems too. For example if you take the approach mentioned above and name each image file after the primary key, on most filesystems you will run into issues if you try to put all of the images in one big directory once you reach a very large number of images (e.g. in the hundreds of thousands or millions).

Once common solution to this is to hash them out into a balanced tree of subdirectories.

Share answered Aug 20, 2008 at 18:25 community wiki John

1 It's better to use a filesystem that has no problem with large directories – Seun Osewa Oct 29, 2008 at 3:46

8 I had an app with millions of files in one directory (server running RHEL 4) - to even list the directory contents (piping to a file) took days and created an output file 100's of MB in size. Now they are in a database I have a single file that I can move or backup quite easily. – Richard Jun 17, 2009 at 15:24

1 @Seun Osewa : every file system has limitations ... and if you know of one that has no problems storing millions of entries in the same directory, please let me know ! – Guillaume Nov 4, 2010 at 12:51

1 @Seun Osewa : the database is up to 28GB now, with 5.4 M records. I ended up having to partition the database table so I have several files to back up that are approx 5GB in size.Moving the individual images onto Amazon S3 now so I only have to store the filename in the DB (and Amazon can do the backups) – Richard Nov 12, 2010 at 7:41

Show **7** more comments

---

22 votes

Something nobody has mentioned is that the DB guarantees atomic actions, transactional integrity and deals with concurrency. Even referentially integrity is out of the window with a filesystem - so how do you know your file names are really still correct?

If you have your images in a file-system and someone is reading the file as you're writing a new version or even deleting the file - what happens?

We use blobs because they're easier to manage (backup, replication, transfer) too. They work well for us.

Share

answered Nov 28, 2008 at 6:33

community wiki
Draemon

1 you don't need simultaneous updates to have problems - it can be a read and a write. In our case this is almost guaranteed to happen. – Draemon Apr 12, 2009 at 22:12

Show **1** more comment

20 votes

The problem with storing only filepaths to images in a database is that the database's integrity can no longer be forced.

If the actual image pointed to by the filepath becomes unavailable, the database unwittingly has an integrity error.

Given that the images are the actual data being sought after, and that they can be managed easier (the images won't suddenly disappear) in one integrated database rather than having to interface with some kind of filesystem (if the filesystem is independently accessed, the images MIGHT suddenly "disappear"), I'd go for storing them directly as a BLOB or such.

Share

answered Apr 8, 2009 at 16:35

community wiki
wiseguy

Add a comment

17 votes

At a company where I used to work we stored 155 million images in an Oracle 8i (then 9i) database. 7.5TB worth.

Share

answered Aug 6, 2008 at 18:37

community wiki
graham.reeds

5 Absolutely. Apparently the database is a lot bigger now. Having the data in a database means that replicating the database at different sites is a lot easier too. – graham.reeds Mar 12, 2009 at 12:06

Show **2** more comments

14 votes

Normally, I'm storngly against taking the most expensive and hardest to scale part of your infrastructure (the database) and putting all load into it. On the other hand: It greatly simplifies backup strategy, especially when you have multiple web servers and need to somehow keep the data synchronized.

Like most other things, It depends on the expected size and Budget.

Share

answered Aug 6, 2008 at 17:42

community wiki
Michael Stum

Add a comment

13 votes

We have implemented a document imaging system that stores all it's images in SQL2005 blob fields. There are several hundred GB at the moment and we are seeing excellent response times and little or no performance degradation. In addition, fr regulatory compliance, we have a middleware layer that archives newly posted documents to an optical jukebox system which exposes them as a standard NTFS file system.

We've been very pleased with the results, particularly with respect to:

1. Ease of Replication and Backup
2. Ability to easily implement a document versioning system

Share

answered Oct 26, 2008 at 17:55

community wiki
dan90266

Add a comment

11 votes

If this is web-based application then there could be advantages to storing the images on a third-party storage delivery network, such as Amazon's S3 or the Nirvanix platform.

Share

answered Aug 6, 2008 at 17:52

community wiki
David

Add a comment

11 votes

Assumption: Application is web enabled/web based

I'm surprised no one has really mentioned this ... delegate it out to others who are specialists -> **use a 3rd party image/file hosting provider**.

Store your files on a paid online service like

- Amazon S3
- Moso Cloud Storage

Another StackOverflow threads talking about this here.

This thread explains why you should use a 3rd party hosting provider.

It's so worth it. They store it efficiently. No bandwith getting uploaded from your servers to client requests, etc.

Share

edited May 23, 2017 at 11:47

community wiki
2 revs
Pure.Krome

Add a comment