Storing Images in DB - Yea or Nay?

Asked 16 years, 4 months ago Modified 12 years, 11 months ago

Viewed 877k 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

Well, you can do both with a transactional disk cache.
– Lilith River Aug 15, 2011 at 21:16

Comments disabled on deleted / locked posts / reviews

56 Answers

Sorted by: Highest score (default)

2 votes

I'm the lead developer on an enterprise document management system in which some customers store hundreds of gigabytes of documents. Terabytes in the not too distant future. We use the **file system** approach for many of the reasons mentioned on this page plus another: archiving.

Many of our customers must conform to industry specific archival rules, such as storage to optical disk or storage in a non-proprietary format. Plus, you have the flexibility of simply adding more disks to a NAS device. If you have your files stored in your database, even with SQL Server 2008's file stream data type, your archival options just became a whole lot narrower.

Share

answered Aug 30, 2008 at 10:15

community wiki
flipdoubt

1

vote

I would personally store the large data outside of the database.

Pros: Stores everything in one please, easy access to data files, easy baskup Cons: Decreases database performance, many page splits, possible database coruption

Share

answered Aug 6, 2008 at 17:41

community wiki
GateKiller

do you mean inside the database? – nickf Nov 28, 2008 at 5:46

1

vote

Your web-server (I'm assuming you are using one) is designed to handle images while a database is not. Thus I would vote heavily on the nay side.

Store just the path (and maybe file info too) in the database.

Share

edited Aug 6, 2008 at 18:55

community wiki
2 revs
Teifion

1

vote

The **only** reason we store images in our tables is because each table (or set of tables per range of work) is temporary and dropped at the end of the workflow. If there was any sort of long term storage we'd definitely opt for storing file paths.

It should also be noted that we work with a client/server application internally so there's no web interface to worry about.

Share

answered Aug 20, 2008 at 18:49

community wiki
hometoast

1

vote

If you need to store lots of images on the file system a couple of things to think about include:

- Backup and restore. How do you keep the images in sync.
- Filesystem performance. Depends on what you are doing and the filesystem, but you may want to implement a hashing mechanism so that you don't have a single directory with billions of files.
- Replication. Do you need to keep the files in sync between multiple servers?

Share

answered Aug 22, 2008 at 15:49

community wiki
Brian Lyttle

1

vote

As someone mentioned already, "it depends". If storage in a database is supposed to be a 1-to-1 fancy replacement for filesystem, it may not be quite a best option.

However, if a database backend will provide additional values, not only a serialization and storage of a blob, then it may make a real sense.

You may take a look at WKT Raster which is a project aiming at developing raster support in PostGIS which in turn serves as a geospatial extension for PostgreSQL database system. Idea behind the WKT Raster is not only to define a format for raster serialization and storage (using PostgreSQL system), but, what's much more important than storage, is to specify database-side efficient image processing accessible from SQL. Long story short, the idea is to move the operational weight from client to database backend, so it take places as close to storage itself as possible. The WKT Raster, as PostGIS, is dedicate to applications of specific domain, GIS.

For more complete overview, check the website and presentation (PDF) of the system.

Share

answered Feb 3, 2010 at 20:39

community wiki
mloskot

0
votes

Attempting to mimic a file system using SQL is generally a bad plan. You ultimately write less code with equal or better results if you stick with the file system for external storage.

Share

answered Aug 20, 2008 at 18:15

community wiki
Gabriel Isenberg

0
votes

Pulling loads of binary data out of your DB over the wire is going to cause huge latency issues and won't scale well.

Store paths in the DB and let your webserver take the load - it's what it was designed for!

Share

answered Aug 22, 2008 at 16:08

community wiki
James Marshall

0
votes

File system, for sure. Then you get to use all of the OS functionality to deal with these images - back ups, webserver, even just scripting batch changes using tools

like imagemagic. If you store them in the DB then you'll need to write your own code to solve these problems.

Share

answered Aug 28, 2008 at 20:12

community wiki
Gordon

0

votes

One thing you need to keep in mind is the size of your data set. I believe that Dillie-O was the only one who even remotely hit the point.

If you have a small, single user, consumer app then I would say DB. I have a DVD management app that uses the file system (in Program Files at that) and it is a PIA to backup. I wish EVERY time that they would store them in a db, and let me choose where to save that file.

For a larger commercial application then I would start to change my thinking. I used to work for a company that developed the county clerks information management application. We would store the images on disk, in an encoded format [to deal with FS problems with large numbers of files] based on the county assigned instrument number. This was useful on another front as the image could exist before the DB record (due to their workflow).

As with most things: 'It depends on what you are doing'

Share

answered Aug 29, 2008 at 23:04

community wiki
Andrew Burns

0
votes

Another benefit of storing the images in the file system is that you don't have to do anything special to have the client cache them...

...unless of course the image isn't accessible via the document root (e.g. authentication barrier), in which case you'll need to check the cache-control headers your code is sending.

Share

answered Aug 30, 2008 at 4:27

community wiki
Rob

0
votes

I prefer to store image paths in the DB and images on the filesystem (with rsync between servers to keep everything reasonably current).

However, some of the content-management-system stuff I do needs the images in the CMS for several reasons- visibility control (so the asset is held back until the press release goes out), versioning, reformatting (some CMS's

will dynamically resize for thumbnails)and ease of use for linking the images into the WYSIWYG pages.

So the rule of thumb for me is to always stash application stuff on the filesystem, unless it's CMS driven.

Share

answered Sep 2, 2008 at 16:15

community wiki
Tim Howland

---

0 votes

I would go with the file system approach. No need to create or maintain a DB with images, it will save you some major headaches in the long run.

Share

answered Sep 2, 2008 at 19:33

community wiki
Tammen

---

0 votes

I would go with the file system approach, primarily due to its better flexibility. Consider that if the number of images gets huge, one database may not be able to handle it. With file system, you can simple add more file servers, assuming that you're using NFS or kind.

Another advantage the file system approach has is to be able to do some fancy stuffs, such as you can use Amazon S3 as the primary storage (save the url in the database instead of file path). In case of outages happen to S3, you fall back to your file server (may be another database entry containing the file path). Some voodoo to apply to Apache or whatever web server you're using.

Share

answered Dec 9, 2008 at 19:51

community wiki
Cygwin98

0

votes

Database for data

Filesystem for files

Share

answered Mar 2, 2009 at 7:14

community wiki
cherouvim

2 You can put that like this: don't put the data in a database column if you cannot use it for a where condition or a join. That is unlikely for binary data. – Nils Weinander Dec 16, 2009 at 14:38

0

votes

I'd almost never store them in the DB. The best approach is usually to store your images in a path controlled by a central configuration variable and name the images according to the DB table and primary key (if possible). This gives you the following advantages:

- Move your images to another partition or server just by updating the global config.
- Find the record matching the image by searching on its primary key.
- Your images are accessable to processing tools like imagemagick.
- In web-apps your images can be handled by your webserver directly (saving processing).
- CMS tools and web languages like Coldfusion can handle uploading natively.

Share

answered May 18, 2009 at 13:36

community wiki
SpliFF

0

votes

I have worked with many digital storage systems and they all store digital objects on the file system. They tend to use a branch approach, so there will be an archive tree on the file system, often starting with year of entry e.g. 2009, subdirectory will be month e.g. 8 for August, next directory

will be day e.g. 11 and sometimes they will use hour as well, the file will then be named with the records persistent ID. Using BLOBS has its advantages and I have heard of it being used often in the IT parts of the chemical industry for storing thousands or millions of photographs and diagrams. It can provide more granular security, a single method of backup, potentially better data integrity and improved inter media searching, Oracle has many features for this within the package they used to call Intermedia (I think it is called something else now). The file system can also have granular security provided through a system such as XACML or another XML type security object. See D Space of Fedora Object Store for examples.

Share

answered Aug 11, 2009 at 11:27

community wiki
shane

0

votes

For a **large number of small images**, the database might be better.

I had an application with many small thumbnails (2Kb each). When I put them on the filesystem, they each consumed 8kb, due to the filesystem's blocksize. A 400% increase in space!

See this post for more information on block size: What is the block size of the iphone filesystem?

Share

edited May 23, 2017 at 12:18

community wiki
2 revs
TJez

0
votes

If you are on Teradata, then Teradata Developer Exchange has a detailed article on loading and retrieving lobs and blobs..

http://developer.teradata.com/applications/articles/large-objects-part-1-loading

Share

answered Sep 27, 2011 at 11:34

community wiki
visakh

0
votes

I will go for both solution, I mean...I will develop a litle component (EJB) that store the images in a DB plus the path of this image into the server. This DB only will be updated if we have a new image or the original image it's updated. Then I will also store the path in the business DB.

From an application point of view, I will always user the file system (retrieving the path from th business DB) and by this way we will fix the backup issue, and also avoid possible performance issues.

The only weakness is that we will store the same image 2 times...the good point is that the memory is cheap, come on!.

Share

answered Jan 26, 2012 at 16:05

community wiki
Pablo

-1

votes

I would go with the file system approach. As noted by a few others, most web servers are built to send images from a file path. You'll have much higher performance if you don't have to write or stream out BLOB fields from the database. Having filesystem storage for the images makes it easier to setup static pages when the content isn't changing or you want limit the load on the database.

Share

answered Aug 6, 2008 at 21:45

community wiki
Chris Miller

-1

votes

No, due to page splits. You're essentially defining rows that can be 1KB - n MB so your database will have a lot of empty spaces in its pages which is bad for performance.

Share

answered Aug 22, 2008 at 16:48

community wiki
Quibblesome

-1

votes

In my current application, I'm doing both. When the user identifies an image to attach to a record, I use ImageMagick to resize it to an appropriate size for display on screen (about 300x300 for my application) and store that in the database for ease of access, but then also copy the user's original file to a network share so that it's available for applications that require higher resolution (like printing).

(There are a couple other factors involved as well: Navision will only display BMPs, so when I resize it I also convert to BMP for storage, and the database is replicated to remote sites where it's useful to be able to display the image. Printing is only done at the head office, so I don't need to replicate the original file.)

Share

answered Sep 8, 2008 at 19:51

community wiki
Randy Orrison

-1

votes

In my little application I have at least a million files weighing in at about 200GB at last count. All the files are sitting in an XFS file system mounted on a linux server over iscsi. The paths are stored in the database. use some kind of intelligent naming convention for your file paths and file names.

IMHO, use the file system for what it was meant to do - store files. Databases generally do not offer you any advantage over a standard file system in storing binary data.

Share

answered Sep 15, 2008 at 19:27

community wiki
siculars

-1

votes

Images on a file store are the best bet, and supplement this with storing the meta data in a database. From a web server perspective, the fast way to serve stuff up is to point to it directly. If it's in the database - ala Sharepoint - you have the overhead of ADO.Net to pull it out, stream it, etc.

Documentum - while bloated and complicated - has it right in that the files are out on the share and available for you to determine how to store them - disk on the server, SAN, NAS, whatever. The Documentum strategy is to store the files a tree structure by encoding the folders and file names according to their primary key in the DB. The DB becomes

the resource for knowing what files are what and for enforcing security. For high volume systems this type of approach is a good way to go.

Also consider this when dealing with metadata: should you ever need to update the attributes of your meta data corpus, the DB is your friend as you can quickly perform the updates with SQL. With other tagging systems you do not have the easy data manipulation tools at hand

Share

answered Oct 2, 2008 at 23:33

community wiki
David Robbins

-1

votes

If you are planning a public facing web site then you should not go with either option. Your should use a Content Delivery Network (CDN). There are price, scalability and speed advantages to a CDN when delivering a large amount of static content over the internet.

Share

answered Nov 4, 2008 at 20:30

community wiki
Craig
