Realtime publishers

Intelligently Reducing SharePoint Costs through Storage Optimization

Don Jones

Copyright Statement

Chapter 4: Optimizing SharePoint Storage for Dormant and Archived Content

Do we really need to keep *every* version of *every* file in the SharePoint database *forever?* Probably not—but where do you draw the line? How can you maximize your version history while minimizing your storage impact? Better yet, how can you take advantage of your existing tiered storage—including tape storage—to archive content without creating a SharePoint database that just grows and grows and grows?

When Content Goes "Dormant"

So what, exactly, is "dormant" SharePoint content? Where does it come from? What makes us consider it dormant instead of active—and why use the word *dormant* rather than something more final like *dead*? I'll offer up four examples of dormant content—and provide you with some abbreviated case study examples from customers I've worked with to help explain the differences between *dormant* and *dead.*

Entire Sites Are No Longer Needed

Sometimes you'll find that you have completely superfluous SharePoint sites. I think this probably happens most frequently through mergers and acquisitions, but I've also seen a lot of them come about from simple bad planning—or in some cases because things didn't pan out quite like anyone expected.

> I worked with one customer who had deployed separate SharePoint sites for their main Human Resources department *and* a separate one for employee perks—you know, discounted tickets to the local theme park, information on the upcoming company picnic, application forms for the local credit union, that kind of thing. The theory was that the perks weren't exactly work-related, so while they were managed by someone in HR, they would be kept separate from the HR site.
>
> The problem is that the *employees* kept going to the HR site to find that information because they knew someone in HR was responsible for organizing it. Prior to SharePoint, in fact, one HR employee—let's call her Kathy—was famous for maintaining the highly-decorated bulletin board of perks and other information in the employee break room.
>
> At first, the company just put some very visible pointers onto the HR SharePoint site. Then they started to realize that *most* of the HR site's traffic was just redirecting off to the perks site. Everyone started to wonder why two sites even made sense, and the perks content was quickly integrated into the HR site.

> The only problem with declaring the perks site "dead," and preventing the company from wanting to delete it outright, is that it still contained a lot of historical information that everyone wanted to preserve. Kathy, in particular, wanted to make sure she'd have access to all of that older information so that she could make sure she was negotiating reasonable new perks with outside companies.

In a merger or acquisition, it's common to want to merge duplicate SharePoint sites—like merging two HR sites into one. It's also common to want to maintain the original content someplace, for reference purposes. You might want to make sure that you migrated everything of importance, for example, or you might be required by law to retain some pieces of information for a designated period of time. In Kathy's case, she wanted to hold on to all that outdated content simply because it served as a good reference for her. Other employees also needed some of the older information—programs that they were still participating in, for example, but that were no longer enrolling new participants. The company could have just migrated all of the information to the main HR site, but that *dormant* content—the stuff a few people would need some of the time but that nobody needed very often—just felt like bloat.

Projects End

One of the biggest reasons to see SharePoint content become dormant is when projects end. Often a project will have a dedicated section of a SharePoint site, or might even have an entirely dedicated SharePoint site devoted to it. As the project winds down, that content becomes dormant—but not so "dead" that everyone's willing to just delete it and be done with it.

> I've done a good amount of work for technology companies, and one of the larger ones I work with spins up a new SharePoint site for every new network integration client they begin courting. All of the bids and proposals, requests for proposals, and other sales details are kept in that SharePoint site—an invaluable way for everyone on the sales team to access everything, collaborate on documents, and so forth.
>
> When the customer is finally acquired—or, sadly, when they lose the bid—that SharePoint site is no longer needed. However, nobody ever wants to just delete the thing. Many of the materials might be leveraged in future bids, for example, and in cases where the company won the bid, they need to maintain that information so that everyone can see exactly what was pitched to the customer—in case there's ever any disagreement.
>
> When I first started working with this company, standard policy was to simply leave every SharePoint site up and running. Forever. Needless to say, they used a *tremendous* amount of storage, all for things that someone *might* need to access *on occasion.*

The company had just about convinced themselves to adopt the "archive and delete" approach, which I'll look at later in this chapter, when I arrived. Before they did that, though, I made them sit down and really lay out some of the business and technology concerns around that dormant—but not *dead*—content so that we could come up with a solution that actually met all of those needs.

Older Versions No Longer Actively Needed

One of the great features of SharePoint is its ability to maintain old versions of documents, and to provide any authorized user with access to those older versions. But sometimes you don't actively *need* all of that old data, and it can take up a significant amount of space in a SharePoint database.

> I've worked with a couple of pharmaceutical firms over the years, and one of the smaller ones uses SharePoint to keep track of a number of documents. Even though those documents remain in active daily use, some of them have old versions going back *years*. Nobody ever needs those old versions—they're hopelessly outdated in most cases—but the company is required by law to maintain them all. There's never been a single instance in the company's history where regulators asked for older versions—but it *could* happen, and so they need to comply with the law.
>
> The results are some seriously bloated SharePoint databases. Even in some of their newer SharePoint sites, where they've implemented some kind of BLOB offloading, they're consuming an enormous amount of disk space storing these old versions.

Regulatory compliance—or in some cases compliance with internal policies—is one of the biggest reasons I see for dormant SharePoint content being maintained in live, active SharePoint sites. Some companies will actually set up separate "archive" sites where they move old stuff—but in some cases, depending on how they move data, they've found that they've lost the old version info that was stored in the original site.

Content Is Phased Out

Sometimes, you just don't *need* content anymore. But that doesn't mean you're ready to get rid of it permanently.

> One of my clients recently started changing their entire business model. They had been a software development consulting firm, and they were moving toward network management and out of the software development business. They still have *tons* of information stored in SharePoint—project specifications, customer files, and so on—that they're simply not going to need on a day-to-day basis anymore. The owner of the company is concerned about deleting it, though. He knows there's information in there that someone will need every other month or so—they've already seen that pattern, in fact, which is why they've been reluctant to finally decommission those old SharePoint sites.

> They've actually considered moving that content to *outsourced* SharePoint storage, which in some cases can be pretty inexpensive, just so that they don't have to maintain the servers, the additional Microsoft software licenses, and so on. He just can't bring himself to pay for services that are just there for dormant content.

The first company I ever worked for was an electronics repair company, and for a variety of reasons, they needed to keep *every* customer invoice they'd ever written. An entire room was full of those cardboard banker's boxes, stuffed with old invoices, payment records, and so on—and, every month or so, someone would actually need to go in there and dig up something out of the past. Just because content is old or has been phased out doesn't mean it's *dead,* but it doesn't mean you have to relish spending production-level resources and money to maintain it.

Business Concerns for Dormant Content

When I sit down with my customers to talk about why all their dormant content is still live, and what they'd like to do with it, the reasons tend to break down into three main categories. These are their business concerns, and in some cases they're a bit contradictory.

Storage Utilization

First and foremost, everyone is concerned about the cost of storing all that old information. Server storage—especially for data-intensive systems like SharePoint—isn't cheap. It has to be redundant, often incorporates expensive hot-swappable drives and powerful drive controllers, and frequently comes in the form of expensive Storage Area Network (SAN) storage. Tell a director that he needs to buy more storage when he *knows* that a lot of it is being occupied by dormant content, and he just gets frustrated. Here's what one IT executive told me in a letter one time:

> It's like file servers, all over again, and worse. When we had file servers, we knew we had a ton of duplicate and useless content, so we started implementing file type filtering, storage quotas, and so forth. That started to bring the problem under control, but only barely. We still knew we had tons of dormant content that should have been in a tape archive or something, but we made a regular effort to identify that stuff and get it moved on a schedule. In some cases we probably deleted stuff we shouldn't have, but it kept the storage bloat somewhat under control.
>
> Now, with SharePoint, we're right back where we started from—only it's worse, because we're not only storing dormant content, we're storing every version of every document. So for every dormant file we have, there's probably a dozen or more old versions of that file floating around. We have less control over filtering and quotas, and it's actually more difficult for us to go in and identify the old stuff. Plus, it's all tied into the SharePoint database—we can't just copy a bunch of files onto tape and then delete them. We have to wrangle them out of SharePoint somehow—and if we do that, how will anyone ever find it again?

It's definitely frustrating. One customer told me that something like 80% of their SharePoint database space was consumed by file attachments (which matches what I hear from other customers), and that something like 70% of *that* space was files and old versions that weren't being actively used by anyone—but that couldn't be entirely deleted, yet.

Imagine living in a nice, three-bedroom house—where two bedrooms and every closet, not to mention the garage, are filled with your kids' memorabilia. You don't want to throw that kind of stuff away—you'll want to pull it out when they finally bring the grandkids over someday—but at the same time, it's a bit frustrating to have all of your space consumed by this stuff. So why keep it?

Regulatory and Industry Compliance

One reason companies hang on to their dormant content is, as I've mentioned, regulatory. Either there's a law, or some industry rule, that's forcing them to maintain it all. HIPAA, Sarbanes-Oxley, GLBA—they all specify retention times for information. In many cases, though, they specify minimum *and maximum* times, meaning you *must* retain information for a certain period of time—no matter how much disk space it takes to do so—but then you *must* also remove the information once it's retention period has passed. Managing that can be quite complicated. In fact, after having a discussion with one IT executive on this very topic, he wrote me to say:

> It's not complicated for us. We're just not going to permit data covered by those regulations to be stored in SharePoint. We've built an "island"—a specific file server—where that information is stored. Information in databases, like our patient records, go into specific database servers. That way we only have to perform retention and cleaning in a few well-known places. We're considering adding a SharePoint site to that "island of compliance," but it's obviously a bit more overhead than a file server. We have to make sure we can get the right tools in place to manage the content within SharePoint, archive it, and so forth—just like we already do for file servers.

On top of regulations and requirements imposed by external regulations and industry rules, companies often have to deal with their own internal policies that require them to retain content for some period of time.

Retention and Archiving Policies

Those internal policies can be just as restrictive and cumbersome—especially when it comes to SharePoint-based data—as external requirements. One of my customers isn't subject to *any* regulatory or industry rules, and he still struggles with maintaining retention policies.

> We're complicated. We don't have to keep *any* data, legally, but we *choose* to keep quite a bit. Anything related to one of our products has to be retained for ten years past the time we discontinue that product family entirely. Personnel records we keep for three or four years after the person leaves, unless they're entitled to ongoing benefits of some kind, in which case it's pretty much forever. Any communications with customers are retained for five years. Tax and financial records are seven years, I think—that's a legal thing, I guess, and not an internal policy.
>
> Those of us in IT just can't deal with it. Nobody actually classifies the data, so we're supposed to figure out what to keep and for how long. Since nobody has rules on the *maximum* time we have to keep something, we pretty much just keep everything, forever. We tend to grab anything that's more than a couple of years old and write it out to tape, so it keeps our production servers from getting too bogged down—but I tell you, it's a major production when we have to go grab something from tape.

That "major production" is one of the significant disadvantages of the "archive and delete" approach, which I'll discuss in just a moment. But this excerpt does bring up a new and interesting point: IT overhead. Too often, IT is tasked with figuring out what content can be deleted, archived, or whatever—and they often lack the tools and background to make the right decision. In those cases, it's not unusual to see IT err on the side of caution and just "keep everything, forever." Which can certainly be expensive.

The "Archive & Delete" Approach

The traditional approach to dormant content is to write it to some kind of archival medium, and then delete it. Simple. Of course, when it comes to SharePoint things can get a bit more complicated.

Techniques

Think about how you might traditionally archive dormant content. I've worked with companies that used optical media archival—a fancy way of saying they bought DVD recorders for their users, along with stacks of blank DVD-Rs. Other companies write data off to tape, and then delete it from the file server or whatever. The idea has always been to move the data to some durable external storage—typically offline storage, such as a tape or DVD-R or CD-R—and then delete that data from main-line, production storage.

In SharePoint, that gets a bit trickier. First, you have to extract the actual content from the database. Ideally, you'll grab old versions and metadata, too. All that gets written to your archive media, and then you have to delete it from SharePoint. That's definitely a bit more complex than just burning a few files onto a DVD-R. I approached one customer who currently uses the "archive and delete" approach and asked them how they did it. I figured they had acquired some kind of SharePoint archival tool that found unused data, extracted it, stored it with some kind of XML file that contained the metadata, and then perhaps built an index of data that had been stored offline. Here's what they told me:

> We wish we had something that fancy. No, we just take a full backup of the SharePoint database and store it as a permanent archive, outside of our normal backup tape rotation. We do it once a month or so.
>
> It works great until someone actually needs it. We don't actually have any way of describing what's different between any of the tapes, so basically we take a guess as to which tape we need, restore the database to a spare SharePoint server that we have running in a virtual machine, and then see if it contains what we think it does. It's a huge pain in the neck.
>
> As for deleting the data, we did write some scripts to do that. We find anything that hasn't been accessed in about a year and just delete the item entirely. We're starting to produce a list of what gets deleted by the tool during each run, so we basically know that the most recent version of any deleted file is on whatever tape backup we made right before we ran the delete job. We've been using Windows' Desktop Search feature to search through those files and find filenames, and that does help us identify the right backup tape more quickly.

Ouch. Yet, as I touched base with more and more customers, *this* was their approach for dealing with dormant content: Make regular backups, and then delete stuff that's not being actively used. Maybe keep track of what you deleted, so that you'll at least have a clue about where to go find it in your archives. The "archive and delete" approach is definitely not a pretty picture.

Pros and Cons

There's really only one "pro" here: You get dormant content off of your production servers, and you do so in a way that makes it at least theoretically possible to get that content back if you need it. The "cons," however, are many:

- **Finding the content.** Actually locating content in the archive can be impossible, especially if you're not keeping track of the content as you delete it, or if you're not tracking it in some searchable fashion. The bottom line is that you've pulled the content out of SharePoint, whose *main job* was to make that content easier to find. You won't be able to search by content keyword, on metadata, on tags, or anything else. The content is gone from SharePoint.
- **IT overhead.** Find the content in your indexes—if you have them. Figure out the right backup tape to use. Restore the tape to a spare server. Spin up the spare server and access SharePoint to get the file you need. Need another file? Start over. To say that the "archive and delete" approach involves significant IT overhead is a major understatement.
- **No self-service.** Using the "archive and delete" approach certainly makes it difficult to offer users a self-service options for retrieving archived data. Typically, someone from IT has to retrieve tapes and process them, because the traditional approach doesn't involve using near-line storage.
- **Manual archival selection.** With the traditional approach—especially the "back up the whole server and then delete stuff" approach—there are no real automated means of locating data that's dormant and ready for archival. That means there's even *more* overhead—often on IT—to select content for clean-up.

With those cons and that pro in mind, as well as all of the business concerns and requirements already covered, let's establish some specific goals for managing dormant SharePoint content.

Goals for Dormant Content

The idea is to synthesize what we like about the "archive and delete" approach with our business goals and concerns, while seeing if we can find a way to mitigate the downsides of the more traditional offline storage-based archive approaches. We'll come up with a short list of major capabilities that we want to add to SharePoint, enabling us to look for solutions that provide those specific capabilities.

Reduce SharePoint Database Utilization

Our first goal is to reduce SharePoint storage utilization—something even a stack of DVD-Rs, a DVD burner, and the Delete key on your keyboard can accomplish. But we ideally want something that's not going to require some poor human being to go through every SharePoint site and figure out what's dormant and what's still active. Ideally, we a way to specify some business rules. Content with a "Retain" metadata tag, for example, might be kept online for a year and then considered "dormant" if it isn't being accessed. Other content might be considered "dormant" after a few months. Whatever our rules are, we need a way to define them so that the *computer* can do the hard work of figuring out what's considered dormant. After all, isn't performing repetitive, boring, manual tasks *exactly why we invented computers in the first place?*

We also need to accommodate those situations where we're ready to take an entire SharePoint site offline when, for whatever reason, we don't need that site anymore. Taking it offline is, of course, the easy part—we also want to make sure we can still *get to it.*

Keeping Dormant Content Accessible

That's the real trick, here. Once dormant content has been identified, we want it *out* of our expensive SharePoint storage—but we don't want to give up easy access to it. When I say *accessible,* I don't just mean, "we can get to it if someone goes and finds the right tape." I want my users to continue to be able to search for the content *right within SharePoint*—I just don't want the content to be actually *in* SharePoint right then.

Utilizing Existing Tiered Storage Infrastructure

I also don't want to have to buy a lot of expensive, task-specific storage. After all, this is *dormant* content—stuff I'd really rather just get rid of, but can't quite do so for some reason. Ideally, content removed from SharePoint could perhaps per compressed and even de-duplicated, so that it takes up a *lot* less space than it used to. Of course, I still want people to be able to get to it right through SharePoint, so I don't necessarily want the data written to offline storage that will require IT intervention to bring back online.

Techniques and Concerns for SharePoint Content Lifecycle Management

The basic way to accomplish this is as follows:

- Identify, preferably through some kind of automated business rules, data that is considered dormant.
- Remove that data from SharePoint—but leave a "stub" behind, so that the data can still be found. This won't completely free up all the space in the SharePoint database—but it will reduce the amount of database space by perhaps 95-98%, since I'm removing the actual *content* and just leaving the pointer records behind.

- Copy that data to some kind of online or near-line compressed, de-duplicated data store. If I happen to have something like an EMC Centera storage system hanging around, that'd be perfect, but I might also want to push it to cloud-based storage, a network file server, or some other kind of file system. I definitely don't want the dormant data going into SQL Server, though. Encryption might be nice, too, since some of that data is bound to be sensitive.
- Users can still "see" the content in SharePoint—but it's not entirely there. So if they try to access it, I might want to just prevent them from doing so unless they have a specific permission set. If they do, then I want the content to be invisibly "re-hydrated" from storage. I'm actually of two minds about how I might want that re-hydration to happen, and I'll discuss them both in a bit.

In some cases, I might even want a specific area in SharePoint—perhaps a Web part—that I can use to provide specific users with easier access to archived data. Figure 4.1 shows an example, where a user might want to explicitly search through archived data using keywords, metadata, and so forth.

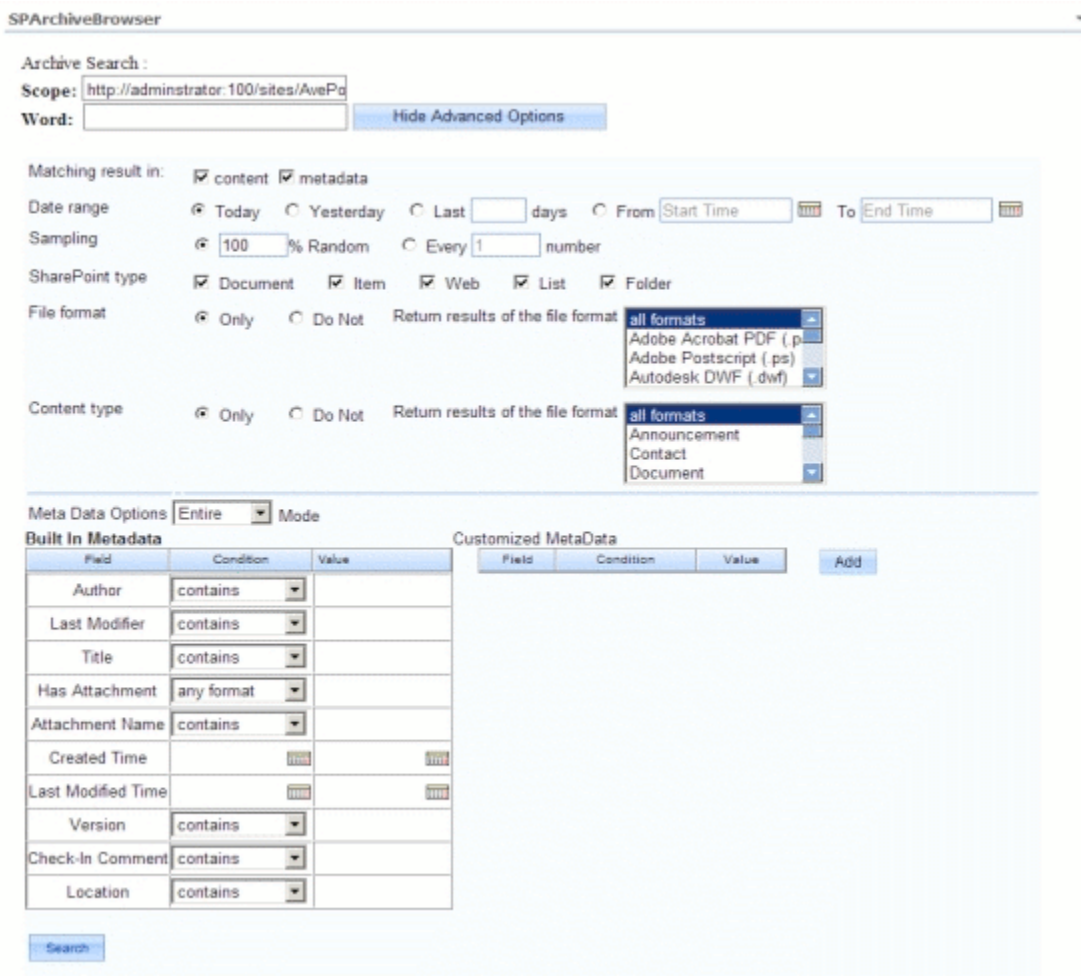

Figure 4.1: Searching archived data through a Web part.

There are, of course, a few tricks into making this do everything I want.

"Dehydrating" and "Re-Hydrating" Data

There's really two phases to archiving: What I'll call *dehydrating* the data, and then the later *re-hydrating* of that data. Here's how I think the dehydrating part should work: Having identified dormant content (based on business rules, which I'll get to in a moment), the archiving solution (shown as the "Archiver" in Figure 4.2) actually identifies that content in the SQL Server database, or by talking to SharePoint. It then moves the *file* portion of the content—the BLOB—out of the SQL Server database, leaving behind what I call the *header* record for that data. Vendors tend to call that a *stub*; it's the bit in the database that contains the file's name, type, metadata, permissions, and so forth. It's actually quite tiny compared to the file itself—perhaps a few kilobytes in most cases. So SharePoint still thinks the content exists in the database—and it does, just not the *content* part of the content, if you follow me.

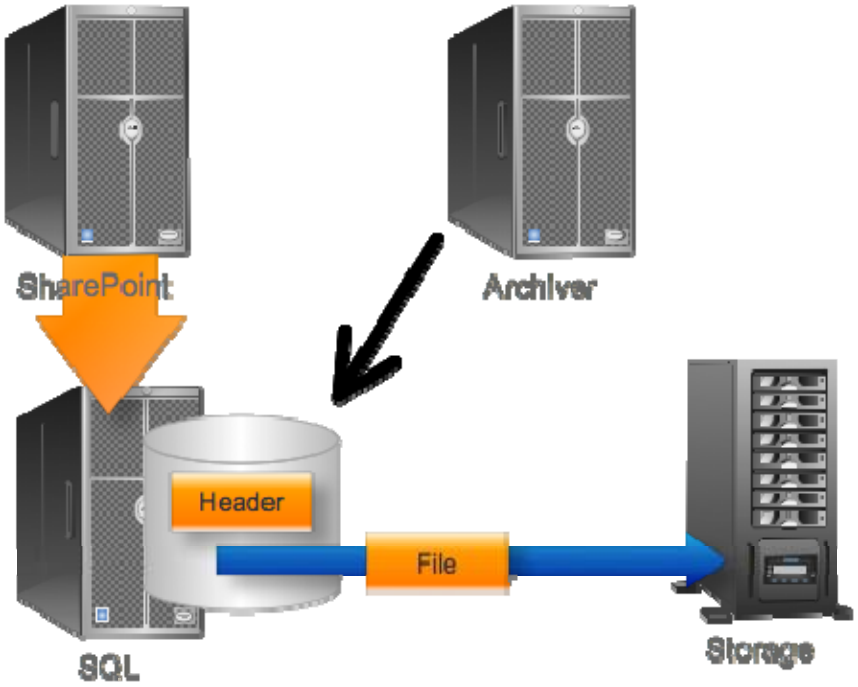

Figure 4.2: Dehydrating data.

When it comes to the re-hydration, I said I was of two minds, and I am. Figure 4.3 shows the first way I think this could happen: When someone accesses a piece of content in SharePoint, the archiver transparently copies the file data back into the SQL Server database.

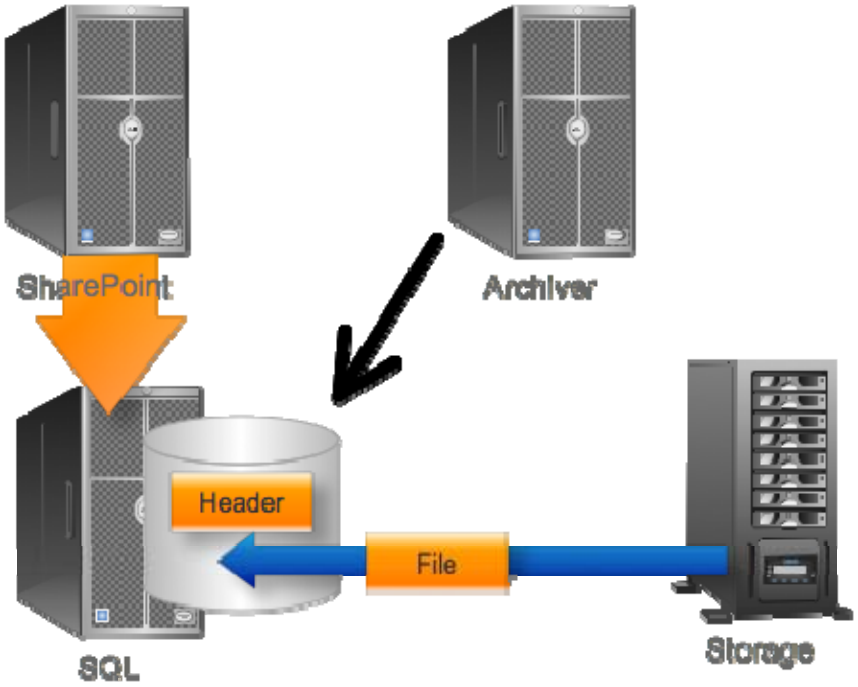

Figure 4.3: Rehydrating data.

That's a perfectly valid technique, but I suspect some vendors will approach the problem a bit differently. Having already implemented BLOB offloading, why not just let the BLOBs *stay* offloaded? In other words, why load the content back into SQL Server—where it might not be used again for years and would just need to be archived again—when you could invisibly provide access to it without moving it? N Figure 4.4, I illustrate how access to the file might be granted *through* SQL Server, using BLOB offloading as I've discussed in previous chapters, without actually putting the file data back into SQL Server.

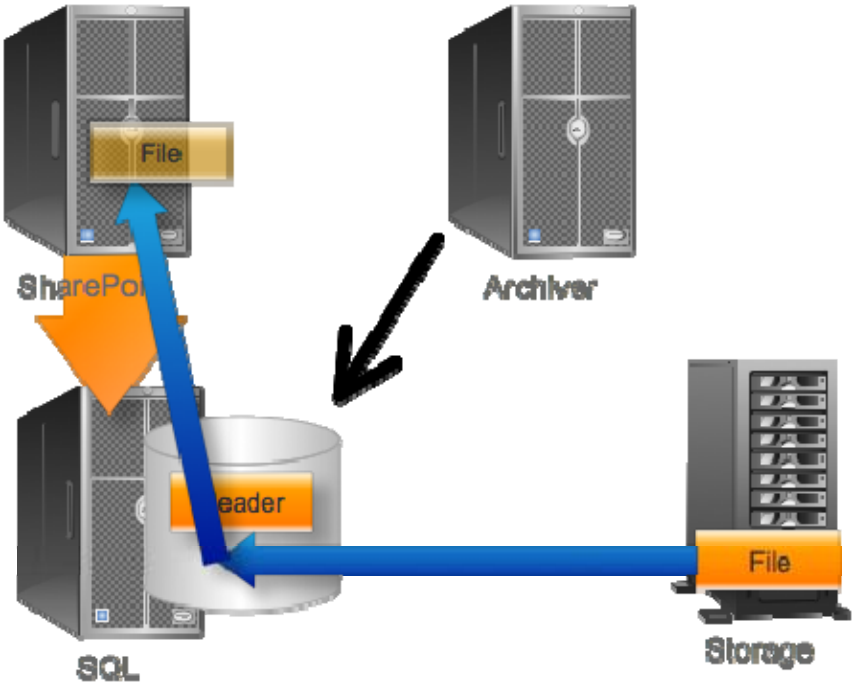

Figure 4.4: Accessing file contents without restoring to SQL Server.

Both approaches have their pros and cons, and it's possible that some solution vendors might even offer both approaches as options. Whatever route you choose, I think the most important thing is that the process be as invisible as possible to your end users. I don't mean *transparent,* because that implies your users are looking "through" something. This process needs to happen magically on the back-end, without your users being any the wiser. *That* means the recovery process has to be *fast*—on the order a gigabyte a minute or so, which is a speed that will seem pretty swift to an average network user.

Business Rules

As I've mentioned, we don't want to have to manually specify what data gets archived. I might want to set rules based on last access time or last modification time, file size, or even metadata. In fact, if I could start convincing my users to apply metadata to help categorize data, that would be an excellent way of identifying items that have unique recovery requirements, like financial data, tax information, and so on. Since I'm not sure if I'll be able to teach them that, I'd like the flexibility to specify less user-reliant rules based on dates and sizes, file types, and so on. Figure 4.5 shows how you might specify some of these rules.

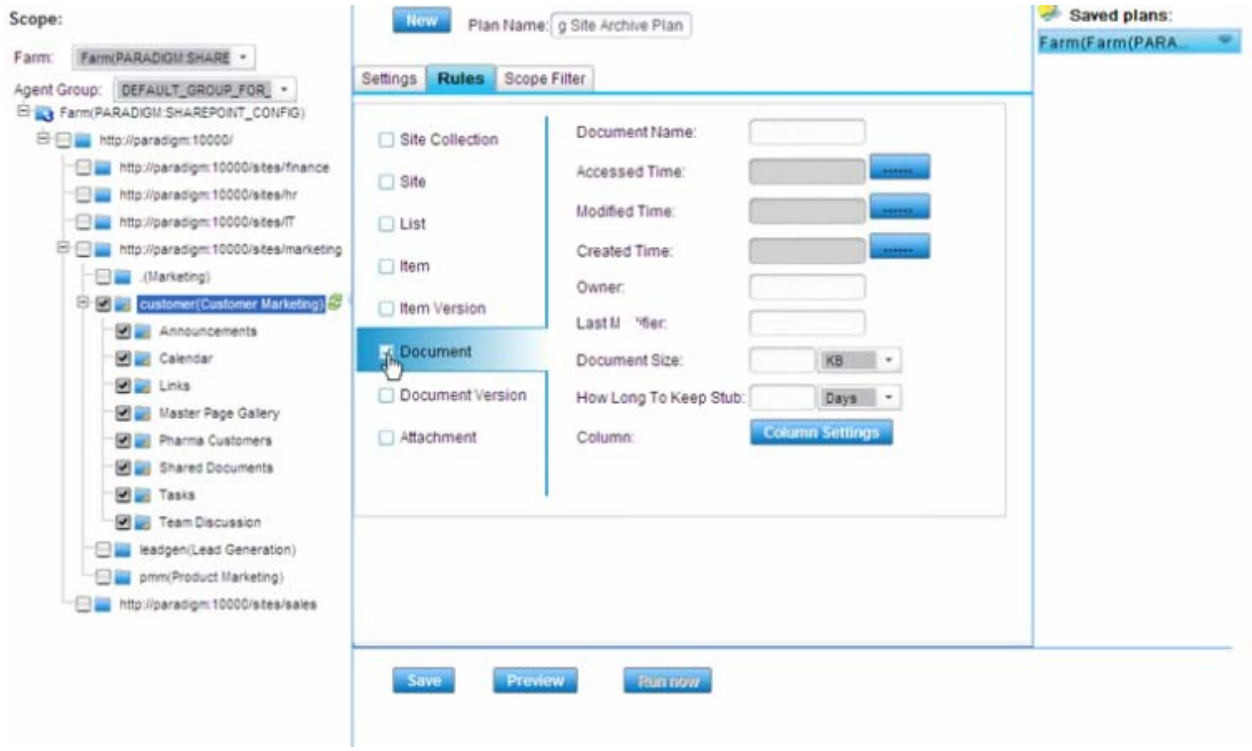

Figure 4.5: Setting business rules for archiving.

I also want to be able to preview what my business rules are about to archive, so that I'm not accidentally archiving *everything* out of SharePoint just because I mistyped a rule criterion or something.

Caution: Sticking with Official APIs

I want to offer a brief word of caution, here: Stick with solutions that use official, published Microsoft application programming interfaces (APIs) for SharePoint. Here's a story from my own experience about the dangers in *not* doing so:

> I worked for a company that had implemented a third-party SharePoint archival solution. This was some time ago, before a lot of public APIs were available to help with this, and so the solution vendor was basically jumping into the database directly, wedging their own code into SharePoint, and so on. This caused a few problems in the long run.
>
> First, when a new version of SharePoint came out, we couldn't upgrade until our solution vendor had upgraded their bits, so they were effectively hacking SharePoint to accomplish their tasks.
>
> Second, when Microsoft finally did release public APIs, they also shut down the back-door access that our solution had been relying on. Our particular vendor took a long time rewriting to use the new APIs, which meant we waited a long time for our SharePoint upgrade.

When Microsoft does decide to implement official APIs for something, they sometimes leave "back doors" in place for a version or two, *deprecating* them and hoping vendors get the message and rewrite their code. Other times, Microsoft just shuts off the unofficial access, especially if it's something they've seen or experienced other problems with. The moral here is that archiving *can* be accomplished through published, official, supported APIs—and you should only consider solutions that use *those* APIs.

Permissions on Archived Data

I want my archived data to maintain *exactly* the same permissions that it had in SharePoint. If I have to manage an additional set of permissions for archived items, I'll go nuts—there are already too many places my IT team has to manage permissions!

This means that when people search for content, they should see archived content *that they have permission to*, and they shouldn't suddenly gain access to new content just because it's been archived. This should actually be pretty easy for a solution to pull off: Remember that a stub of the content is staying in the SharePoint database. That stub is where permissions get applied; all we're doing is archiving the large content associated with that stub. Figure 4.6 shows what I'm talking about: The permissions live *in SharePoint,* and the archival solution should only permit access to content *when that access comes through SharePoint,* helping to ensure that SharePoint remains "in charge" of security of that content.

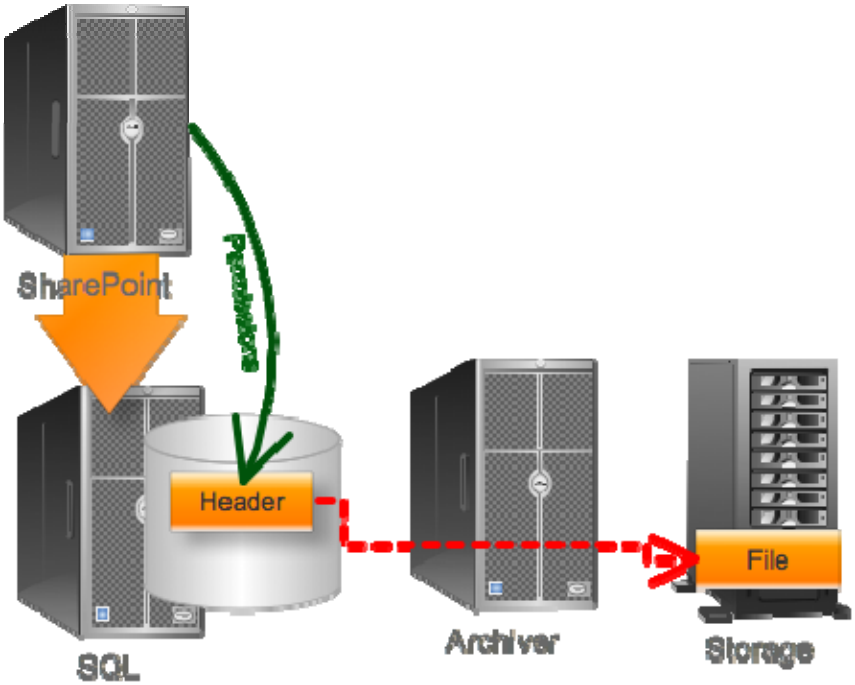

Figure 4.6: Maintaining permissions on dehydrated content.

I'll say this, though: When your archive storage is a file server, there are *going* to be other permissions involved. A Windows file server doesn't do "no permissions," and they don't do "delegated permissions" either. So in that case, the best you can hope for—and this is certainly sufficient—is one of these two approaches:

- The files on the storage system are permissioned in such a way that only the archive solution's service account can access them. This effectively locks everyone else out, and the archive solution only provides access to the file via SharePoint—which respects its own permissions.
- The archive solution actually synchronizes, in some fashion, SharePoint's permissions onto the archived files. That's certainly a valid approach, too, since it keeps SharePoint "in charge."

Many archive solutions store files in such a compressed and encrypted format that accessing them directly from the archival storage isn't feasible; you don't need to worry as much about permissions when the file itself is unusable. You should, however, spend some time understanding how a potential solution handles this security and permissions situation, so that you're comfortable that the approach you're buying will work for your organization.

More Shopping List Items

There are a few other capabilities that are more easily explained, that you should definitely keep in mind when you're shopping for SharePoint archival solutions. You might not need every single one of these, but the ones you do need, you don't want to overlook:

- **Flexibility**. You want to be able to archive anything, from a single item, to a list, to a library, to all items of a specific type, all the way to an entire site.
- **End-User Access.** Hey, let's let our users—those we trust, of course—get in on the act. If they can identify something as dormant, let them do so, and our solution should archive it immediately.
- **Any archive endpoint**. You really want a system that isn't going to lock you into a specific kind of storage. Yes, if you're an EMC customer, you want to use your EMC solution—but if you're not, maybe a simple file server is your archival choice. You should get compression and encryption either way, but be prepared for things like document de-duplication to only come with higher-end storage systems.
- **Archive-specific permissions**. Despite what I said earlier, I do want some specific permissions to the archiving system. I want to be able to decide who can view and restore archived data, and perhaps who is allowed to archive data and mark it as dormant.

- **Direct access to archived data**. When you archive an entire list, library, or site, there's nothing left in SharePoint for people to search for—no *stubs,* in other words. So an archive solution still has to provide some dedicated retrieval interface, and that will often involve actually re-hydrating content back into the production database so that SharePoint can "see" it again. Look for:
    - A solution that provides a SharePoint-integrated interface for retrieving content, rather than forcing users to turn to some proprietary tool.
    - A solution that provides *full fidelity* for restored content—including content, metadata, permissions, version history, and so on. You don't want to lose a thing if you choose to bring something back from dormancy.
- **Full SharePoint integration.** An archival solution should work whether users are accessing content through the SharePoint Web Front-End, or via Office, or via anything else that connects through SharePoint.
- **Reporting.** Solutions need to provide some kind of management reporting, including information on who is using the archive, what data is being accessed, and so on.

You'll doubtless have some requirements of your own, and it's a good idea to document those. Make up an "evaluation checklist" as you start to look at solutions, and keep track not only of which potential solutions meet your requirements, but *how* they meet those requirements, so that you can select the solution that will do the best job for your business in the long run.

Conclusion

There you have it: Some modern ideas for improving and optimizing SharePoint storage, for getting *more* content into SharePoint without bloating the database, and making SharePoint an overall more-useful part of your enterprise. I hope you've found these ideas useful, and that you're ready to start researching solutions that can give you these capabilities in your own SharePoint environment.