Directory Creation setting

The “directory creation” database setting is one of the first things I change for a new database. The default is “automatic”, but I switch it to “manual”, as I was taught by more experienced developers when I joined MarkLogic. The reason given was performance — use manual so that MarkLogic doesn’t bother creating directories. I’ve been doing this for a couple years without ever exploring what the different settings really mean. I’ve never even used the third option: “manual-enforced”.

The MarkLogic [Administrator’s Guide](http://community.marklogic.com/pubs/5.0/books/admin.pdf) has definitions for the directory creation settings (more details in section 12.1.4.4, Document and Directory Settings):

* automatic: “directories are automatically created based on the URI of a document”
* manual-enforced: “requires that the directory hierarchy corresponding to the URI exists before creating a document”
* manual: “directories are not automatically created, but documents can still be created without corresponding directories”

Directories in MarkLogic are pretty analogous to directories on a file system: they are a hierarchical system for organizing documents, which exist inside the directories. When a document is inserted with a URI like “/content/binary/admin.pdf”, there are three directories mentioned in that URI: “/”, “content/”, and “binary/”.

**Experimenting**

Let’s see what happens when we insert a document with a multilevel URI, using each of the different settings. Each of these tests was run on a clear database with default settings.

manual:  
xdmp:document-insert(“/a/b/c.xml”, <test/>)  
2 fragments: c.xml, c.xml properties

manual-enforced:  
xdmp:document-insert(“/a/b/c.xml”, <test/>)  
error

automatic:  
xdmp:document-insert(“/a/b/c.xml”, <test/>)  
5 fragments: /, a/, b/, c.xml, c.xml properties

We see that the automatic setting creates a total of 5 fragments: three for the directories, one for the document itself, and one for the properties fragment on the document. Under manual, the directories are skipped, and only the document and its properties fragment are created. Manual-enforced throws an error. Under that setting, we would need to call[xdmp:directory-create()](http://community.marklogic.com/pubs/5.0/apidocs/UpdateBuiltins.html#xdmp:directory-create) for each of the directories we need. All three perform as advertised in the Admin Guide.

Interesting…

Here’s an interesting thing I noticed: the docs tell us that with “manual”, directories are not automatically created. That tells me that we don’t have the directories “/”, “a/”, and “b/”. The fragment counts in the database correspond to this. How else can we test for the existence of a directory? One way is to ask MarkLogic for the contents of the directory:

xdmp:directory("/", "infinity")

Running that query retrieves my test document, as does running the same query with “/a/” or “/a/b/”. (I get the same results running a search based on cts:directory-query.) That tells me that even with directory creation set to manual and skipping the actual manual construction of directories, the directories still have some kind of existence in the sense that they can be used for search. But we know that directories aren’t being created, so we must be missing out on something by not having them.

**Tradeoffs**

I mentioned that the reason for shifting to manual is performance. Automatically creating unnecessary directories does a couple things. First, there is the directory creation itself, slightly slowing document ingest. Second, that directory fragment increases the number of items the database needs to maintain, including copying those fragments while merging stands.

So automatic directory creation has an impact. Does it have a benefit? Yes — if you are using some specific features.

WebDAV

As the Admin Guide tells us, WebDAV requires directories to exist in order for a client to see database content — just having slashes in your URIs doesn’t cut it here. In fact, when I specified “/” as the root directory while trying to create a WebDAV app server, I got an error:

The root directory hierarchy for this server is missing the following directory: /

Manual creation didn’t make the “/” directory, so WebDAV doesn’t see it. Next, I wondered what I’d see by creating the directories manually:

xdmp:directory-create(“/”)

By creating just the root directory, I still couldn’t see anythings. I needed to create “/a/” and “/a/b/” before I could see anything with WebDAV. So if you intend to create a WebDAV server, you’ll either need to use “automatic” directory creation or manually create every directory you wish to expose.

Modules Databases

The Admin Guide also tells us to use “automatic” for modules databases. I looked into this more, running some experiments and asking around on a MarkLogic-internal mailing list. The consensus is that the only reason you would need automatic directory creation for modules databases comes back to the WebDAV use case, which is more common for a modules database. Some people will set up WebDAV and use that to directory edit the source code in the modules db.

I ran my own tests using a [Roxy](https://github.com/marklogic/roxy)-based application. I ran bootstrap, then changed the modules database’s directory creation setting to manual, then deployed the code. It worked with no problems. I also set up a non-admin user just to make sure there wasn’t something special about admin that made things work. Still no problems. My conclusion, between my experiments and following up with others, is that strictly speaking there is no requirement that modules databases use automatic directory creation. If you want to allow WebDAV source code editing, you’ll want automatic, but that’s really the same use case as the above.

## [Directory Assistance](http://blakeley.com/blogofile/2012/03/19/directory-assistance)

March 19, 2012 at 12:34 PM | categories: [MarkLogic](http://blakeley.com/blogofile/category/marklogic) | [1 Comment](http://blakeley.com/blogofile/2012/03/19/directory-assistance#disqus_thread)

For a long time now, MarkLogic Server has implemented two distinct features that are both called "directories". This causes confusion, especially since one of these features scales well and the other often causes scalability problems. Let's try to distinguish between these two features, and talk about why they both exist.

Directories were first introduced to accommodate WebDAV. Since WebDAV clients treat the database as if it were a filesystem, they expect document URIs with the solidus, or /, to imply directory structure. That's one feature called "directories": if you insert a document with the URI/a/b/c.xml, you can call xdmp:directory('/a/b/', '1') to select that document - and any other document with the same URI prefix. These URI prefixes are indexed in much the same way that document URIs and collection URIs are indexed, so queries are "searchable" and scale well.

This "implied directory structure" works with any database configuration. You do not needdirectory-creation=automatic to use the cts:directory-query and xdmp:directoryfunctions.

|  |  |
| --- | --- |
| 1 | **xdmp:plan**(**xdmp:directory**('/a/b/', '1')) |

[**view raw**](https://gist.github.com/mblakele/2127471/raw/gistfile1.xq)[**gistfile1.xq**](https://gist.github.com/mblakele/2127471#file-gistfile1-xq) hosted with ❤ by **[GitHub](https://github.com/)**

This returns a query plan in XML:

|  |  |
| --- | --- |
| 12345678910111213141516171819202122232425262728293031 | <qry:query-plan xmlns:qry="http://marklogic.com/cts/query">  <qry:info-trace>xdmp:eval("xdmp:plan(xdmp:directory('/a/b/', '1'))", (), &lt;options xmlns="xdmp:eval"&gt;&lt;database&gt;18400529833056734238&lt;/database&gt;&lt;root&gt;/Users/mblakele/S...&lt;/options&gt;)</qry:info-trace>  <qry:info-trace>Analyzing path: xdmp:directory("/a/b/")</qry:info-trace>  <qry:info-trace>Step 1 is searchable: xdmp:directory("/a/b/")</qry:info-trace>  <qry:info-trace>Path is fully searchable.</qry:info-trace>  <qry:info-trace>Gathering constraints.</qry:info-trace>  <qry:info-trace>Step 1 contributed 2 constraints: xdmp:directory("/a/b/")</qry:info-trace>  <qry:partial-plan>  <qry:term-query weight="0">  <qry:key>14693750357517395314</qry:key>  </qry:term-query>  </qry:partial-plan>  <qry:partial-plan>  <qry:term-query weight="0">  <qry:key>15041569596143136458</qry:key>  </qry:term-query>  </qry:partial-plan>  <qry:info-trace>Executing search.</qry:info-trace>  <qry:final-plan>  <qry:and-query>  <qry:term-query weight="0">  <qry:key>14693750357517395314</qry:key>  </qry:term-query>  <qry:term-query weight="0">  <qry:key>15041569596143136458</qry:key>  </qry:term-query>  </qry:and-query>  </qry:final-plan>  <qry:info-trace>Selected 0 fragments</qry:info-trace>  <qry:result estimate="0"/>  </qry:query-plan> |

[**view raw**](https://gist.github.com/mblakele/2127484/raw/gistfile1.xml)[**gistfile1.xml**](https://gist.github.com/mblakele/2127484#file-gistfile1-xml) hosted with ❤ by **[GitHub](https://github.com/)**

But WebDAV clients expect more than just directory listings. They also want to lock documents and directories. It is easy to understand document locking: the idea here is that a WebDAV-aware editor might lock a document, copy it to the local filesystem for editing, and copy it back to the server when the editing session ends. It may be less clear that a WebDAV client sometimes needs to lock directories, but it does.

Directory locking is implemented using special directory fragments. There are no documents associated with these properties, so they are sometimes called "naked properties." Here is an example.

|  |  |
| --- | --- |
| 1 | **xdmp:directory-create**('/a/b/') |

[**view raw**](https://gist.github.com/mblakele/2127498/raw/gistfile1.xq)[**gistfile1.xq**](https://gist.github.com/mblakele/2127498#file-gistfile1-xq) hosted with ❤ by **[GitHub](https://github.com/)**

Once this update has committed to the database, we can query the directory fragment.

|  |  |
| --- | --- |
| 1 | **xdmp:document-properties**('/a/b/') |

[**view raw**](https://gist.github.com/mblakele/2127503/raw/gistfile1.xq)[**gistfile1.xq**](https://gist.github.com/mblakele/2127503#file-gistfile1-xq) hosted with ❤ by **[GitHub](https://github.com/)**

|  |  |
| --- | --- |
| 1234 | **<?xml version="1.0" encoding="UTF-8"?>**  <prop:properties xmlns:prop="http://marklogic.com/xdmp/property">  <prop:directory/>  </prop:properties> |

[**view raw**](https://gist.github.com/mblakele/2127509/raw/gistfile1.xml)[**gistfile1.xml**](https://gist.github.com/mblakele/2127509#file-gistfile1-xml) hosted with ❤ by **[GitHub](https://github.com/)**

Once you have a directory fragment, you have something that the database can lock for WebDAV clients. It's rare for anything else to use this behavior, but xdmp:lock-acquire is available for custom content management applications.

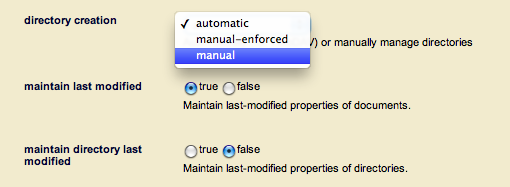
Earlier I mentioned that there are two kinds of "directories", one that scales well and one that sometimes causes problems. I wrote that queries based on directory URIs scale well, so you might guess that directory fragments sometimes cause problems. That's correct, and it results from a database feature called "automatic directory creation".

When automatic directory creation is enabled - as it is by default - the database will ensure that directory fragments exist for every implied directory in the URI for every new or updated document. The document URI /a/b/c.xml implies a directory fragment for /, /a/, and /a/b/. So the database will ensure that these exist whenever a request updates /a/b/c.xml.

So what happens when one request updates /a/b/c.xml and another request updates/a/b/d.xml?

Both requests try to ensure that there are directory fragments for /, /a/, and /a/b/. This causes lock contention. The same problem shows up if another request is updating /fubar.xml, because both queries look for the / directory fragment. The situation gets worse as concurrency increases. It gets even worse if "maintain directory last-modified" is enabled, because the directory fragments have to be updated too. But happily that feature is not enabled by default.

The solution to this problem is simple. In my experience at least 80% of MarkLogic Server customers do not use WebDAV, so they do not need automatic directory creation. Instead, they can set directory creation to "manual". Do this whenever you create a new database, or script it using admin:database-set-directory-creation.



If you do use WebDAV, try to limit its scope. Perhaps you can get by with a limited number of predefined WebDAV directories, which you create manually using xdmp:directory-create as part of your application deployment. Or perhaps you only use WebDAV for your XQuery modules, which only contains a few hundred or at most a few thousand documents. In that case you can use automatic directory creation without a problem.

Generally speaking, really large databases don't use WebDAV anyway. "Big content" databases, with hundreds of millions or billions of documents, tend to be much to large for WebDAV to be useful. For smaller databases where WebDAV is useful, automatic directory creation is fine.

Sometimes it is useful to set "directory-creation" to "manual-enforced". With this configuration you will see an XDMP-PARENTDIR error whenever your code tries to insert a document with an implied directory structure that does not have corresponding directory fragments. But this feature is rarely used.

To sum up, directory URIs are highly scalable and very useful, and are always indexed. Your code can call xdmp:directory with any database settings. The default "automatic directory creation" feature creates directory fragments, which can be a bottleneck for large databases. Most applications are better off with "directory-creation" set to "manual".