

Confluence attachment filesystem paths

Sometimes we'd like to know where on the filesystem a particular Confluence attachment is. For example, after doing upgrades or maintenance, one might like to check that all attachments are present.

The following SQL query (Postgres syntax) does just that: for every attachment found in the database, it prints the filesystem path, expected file size, page URL and attachment download URL:

```
WITH baseurl as (
    SELECT regexp_replace(bandanavalue, '.*<baseUrl>([^\<]+)</baseUrl>.*', '\1') as baseurl FROM bandana
WHERE bandanakey = 'atlassian.confluence.settings'
)
,attachments AS (
    SELECT content.*
        , mt.stringval AS mediatype
        , fs.longval AS filesize
    FROM content JOIN contentproperties mt USING (contentid)
        JOIN contentproperties fs USING (contentid)
    WHERE content.contenttype='ATTACHMENT' and mt.propertyname='MEDIA_TYPE' and fs.propertyname='FILESIZE'
    -- and content.pageid = 131170321
)
,attachments_data AS (
    SELECT
        pageid
        , title
        , extract(epoch from lastmoddate)*1000 AS moddate
        , version
        , contentid
        , creationdate
        , concat_ws('/',
            'ver003',
            coalesce(
                right(spaceid::text, 3)::integer % 250
                || '/' ||
                substring(spaceid::text from '(...)...$')::integer % 250
                || '/' ||
                spaceid
            , 'nonspaced'
        )
        , right(pageid::text, 3)::integer % 250
        , substring(pageid::text from '(...)...$')::integer % 250
        , pageid
        , first_value(contentid) OVER (PARTITION BY pageid, title order by version desc)
        , version
    ) as attpath
    , filesize
    FROM attachments
    WHERE content_status='current' and contenttype='ATTACHMENT'
)
SELECT
    concat_ws('/', 'attachments', attpath) as fullpath
    ,filesize
    ,concat_ws('/', baseurl, 'pages/viewpage.action?pageId=||pageid) AS pagelink
    ,concat_ws('/', baseurl, 'download', 'attachments', pageid, title) -- Note: title is unencoded. If
this bugs you, install urlencode() function from https://stackoverflow.com/questions/10318014/javascript-
encodeuri-like-function-in-postgresql
    || '?' || concat_ws('&', 'version=||version, 'modificationDate=||moddate, 'api=v2') AS
attachmentlink
FROM
    attachments_data CROSS JOIN baseurl;
```

To check that all attachments are present, use a bash pipeline like this:

```
cat /tmp/confattachments.sql | psql confluence -tAq | while IFS='|' read path filesize pageurl atturl ; do test
-f /var/atlassian/application-data/confluence/current/"$path" || echo "Missing attachment $path on page
$pageurl"; done
```

How does it work?

The official documentation on Confluence's attachment filesystem structure is [Hierarchical File System Attachment Storage](#). At a high level, an attachment's path consists of:

- A fixed part, `ver003`
- 3 directories derived from the attachment's **space**
- 3 directories derived from the attachment's **page**
- a directory for each distinct attachment, as identified by attachment **filename** or title
- a file for the particular **version** of an attachment

Here is an example: two attachments on a page, one having two versions:

Name	Size	Creator	Creation Date	Labels	Comment	
test.sh	0.1 kB	Jeff Turner	Aug 30, 2017 13:15	No labels	Test script	Properties Delete
Version 1 (current)	0.1 kB	Jeff Turner	Aug 30, 2017 13:15		Test script	Delete
My File.txt	0.1 kB	Jeff Turner	Aug 30, 2017 13:15	No labels	second version	Properties Delete
Version 2 (current)	0.1 kB	Jeff Turner	Aug 30, 2017 13:15		second version	Delete
Version 1	0.1 kB	Jeff Turner	Aug 30, 2017 13:15		First!	Delete

The URL of this particular page is <https://example.com/pages/viewpageattachments.action?pageId=131629058>

How does this look in the database?

Attachments, like most things in Confluence, are stored in the content table:

```
redradish_confluence=> select spaceid, pageid, contentid, contenttype, title, creationdate, version, prevver
from content where pageid = 131629058;
```

spaceid	pageid	contentid	contenttype	title	creationdate	version	prevver
3014657	131629058	131629061	ATTACHMENT	test.sh	2017-08-30 13:15:33.575	1	
3014657	131629058	131629059	ATTACHMENT	My File.txt	2017-08-30 13:15:20.239	2	
3014657	131629058	131629060	ATTACHMENT	My File.txt	2017-08-30 13:15:07.179	1	131629059

(3 rows)

Reading the [Hierarchical File System Attachment Storage](#) page, you'll see details of how spaceid and pageid have to be split into bits, mod 250'd and so forth:

level	Derived From
1 (top)	Always 'ver003' indicating the Confluence version 3 storage format
2	The least significant 3 digits of the <i>space id</i> , modulo 250
3	The next 3 least significant digits of the <i>space id</i> , modulo 250
4	The full <i>space id</i>
5	The least significant 3 digits of the <i>content id</i> of the page the file is attached to, modulo 250
6	The next 3 least significant digits of the <i>content id</i> of the page the file is attached to, modulo 250
7	The full <i>content id</i> of the page the file is attached to
8	The full <i>content id</i> of the attached file

Here is a first, incorrect attempt in Postgres:

```

select spaceid, pageid, contentid, title, version
      , concat_ws('/', -- Join each bit together with '/'
        'ver003'
        , right(spaceid::text, 3)::integer % 250 -- Rightmost 3 chars of spaceid, mod 250
        , substring(spaceid::text from '(...)...$')::integer % 250 -- chars 3-6 from right of
spaceid, mod 250
      , spaceid
      , right(pageid::text, 3)::integer % 250 -- Rightmost 3 chars of pageid, mod 250
      , substring(pageid::text from '(...)...$')::integer % 250 -- chars 3-6 from right of
pageid, mod 250
      , pageid
      , contentid
      , version) AS path
from content where pageid =131629058;

```

spaceid	pageid	contentid	title	version	path
3014657	131629058	131629061	test.sh	1	ver003/157/14/3014657/58/129/131629058/131629061/1
3014657	131629058	131629059	My File.txt	2	ver003/157/14/3014657/58/129/131629058/131629059/2
3014657	131629058	131629060	My File.txt	1	ver003/157/14/3014657/58/129/131629058/131629060/1

(3 rows)

When I check in the filesystem, the first two paths are correct, but the last path does not exist.

What is the right path for version 1 of 'My File.txt'? This can be discovered on Linux with the handy `iosnoop` utility from <https://github.com/iovisor/bcc>, which shows what files are being accessed. Clicking on version 1 of 'My File.txt' yields:

```

root@jturner-desktop ~ # opensnoop-perf /var/atlassian/application-data/redradish_confluence/6.3.1/attachments
Tracing open()s for filenames containing "/var/atlassian/application-data/redradish_confluence/6.3.1
/attachments". Ctrl-C to end.
COMM          PID      FD FILE
java          27335   0x67 /var/atlassian/application-data/redradish_confluence/6.3.1/attachments/ver003/157
/14/3014657/58/129/131629058/131629059/1

```

So the correct path is `ver003/157/14/3014657/58/129/131629058/131629059/1` not `ver003/157/14/3014657/58/129/131629058/131629060/1`. This is part 8 of the path:

8	The full <i>contentid</i> of the attached file
---	--

We used `contentid`. So what's going wrong?

It appears that level 8 is not simply the `contentid`, but rather the `contentid` of the *most recent version of the attached file*. This can be seen in the source at `HierarchicalMultiStreamAttachmentDataFileSystem.java`. The `getContainerDirectoryForAttachmentVersions()` function takes, as arguments, the latest attachment's `contentid`, then the `pageid`, then `spaceid`:

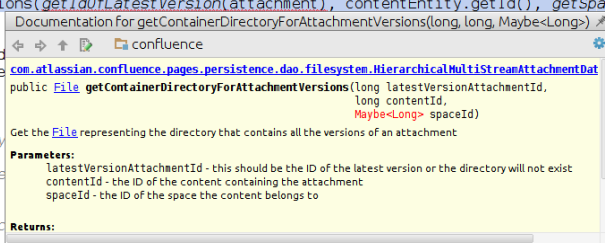
```

private File getContainerDirectoryForAttachmentVersions(final ContentEntityObject contentEntity, final Attachment attachment) {
    return getContainerDirectoryForAttachmentVersions(getIdOfLatestVersion(attachment), contentEntity.getId(), getSpaceIdForContent(contentEntity));
}

private static <T extends EntityObject & Versioned>
return ((EntityObject) versionedEntity).getLatestVersion().getId();
}

/**
 * Get the {@link File} representing the directory
 *
 * @param latestVersionAttachmentId this should be
 * @param contentId the ID of the
 * @param spaceId the ID of the
 * @return the {@link File} representing the directory
 * @since 5.7
 */

```



In our case, we have two versions of 'My File.txt'. Version 1 (the 1st) has `contentid` 131629060, version 2 has `contentid` 131629059. We need to use the version 2 (or whatever is latest) `contentid`, which is 131629059.

Back to our SQL query. Instead of just `contentid`, we need `contentid` of the attachment with the same filename, on the same page, with the highest version. That can be expressed with a Postgres [Window function](#):

```
select
    spaceid, pageid, contentid, contenttype, title, creationdate, version, prevver,
    FIRST_VALUE(contentid) OVER (PARTITION BY pageid, title order by version desc) AS newest_contentid
from content where pageid = 131629058;
```

spaceid	pageid	contentid	contenttype	title	creationdate	version	prevver	newest_contentid
3014657	131629058	131629059	ATTACHMENT	My File.txt	2017-08-30 13:15:20.239	2		131629059
3014657	131629058	131629060	ATTACHMENT	My File.txt	2017-08-30 13:15:07.179	1	131629059	131629059
3014657	131629058	131629061	ATTACHMENT	test.sh	2017-08-30 13:15:33.575	1		131629061

(3 rows)

So our more-correct path-calculating SQL is:

```
select spaceid, pageid, contentid, title, version
    , concat_ws('/',
        'ver003'
        , right(spaceid::text, 3)::integer % 250
        , substring(spaceid::text from '(...)...$')::integer % 250
        , spaceid,
        , right(pageid::text, 3)::integer % 250
        , substring(pageid::text from '(...)...$')::integer % 250
        , pageid
        , first_value(contentid) OVER (PARTITION BY pageid, title order by version desc)
        , version) AS path
from content where pageid =131629058;
```

spaceid	pageid	contentid	title	version	path
3014657	131629058	131629059	My File.txt	2	ver003/157/14/3014657/58/129/131629058/131629059/2
3014657	131629058	131629060	My File.txt	1	ver003/157/14/3014657/58/129/131629058/131629059/1
3014657	131629058	131629061	test.sh	1	ver003/157/14/3014657/58/129/131629058/131629061/1

(3 rows)

The paths for our test attachments are now correct.

There is one more wrinkle to consider: **attachments on new draft pages**. To test:

- click 'Create' to create a page
- Use the '!' notation to add an attachment (here called `setenv.diff`)
- Without saving, click on your profile icon, then 'Drafts'
- Find your draft page and 'resume editing'.
- Observe the page ID in the URL, which will be something like `https://wiki.redradishtech.com/pages/createpage.action?useDraft=true&spaceKey=-jturner&draftId=131629078&`

In my example, the draft ID is 131629078:

```

select spaceid, pageid, contentid, title, version
, concat_ws('/',
, right(spaceid::text, 3)::integer % 250
, substring(spaceid::text from '(...)...$')::integer % 250
, spaceid, right(pageid::text, 3)::integer % 250
, substring(pageid::text from '(...)...$')::integer % 250
, pageid
, first_value(contentid) OVER (PARTITION BY pageid, title order by version desc)
, version) AS path
from content where pageid = 131629078;

```

spaceid	pageid	contentid	title	version	path
	131629078	131629079	setenv.diff	1	ver003/78/129/131629078/131629079/1

(1 row)

You will see the calculated path is not correct. It is entirely missing the 3 segments derived from spaceid, because spaceid is null.

In this situation Confluence uses a directory 'nonspaced' . We can do this with the coalesce() function, to use 'nonspaced' if the spaceid-using expression evaluates to null:

```

select spaceid, pageid, contentid, title, version
, concat_ws('/',
, coalesce(
, right(spaceid::text, 3)::integer % 250
|| '/' ||
, substring(spaceid::text from '(...)...$')::integer % 250
|| '/' ||
, spaceid,
, 'nonspaced'
)
, right(pageid::text, 3)::integer % 250
, substring(pageid::text from '(...)...$')::integer % 250
, pageid
, first_value(contentid) OVER (PARTITION BY pageid, title order by version desc)
, version) AS path
from content where pageid = 131629078;

```

spaceid	pageid	contentid	title	version	path
	131629078	131629079	setenv.diff	1	ver003/nonspaced/78/129/131629078/131629079/1

(1 row)

..and that forms the core of the final SQL query presented at the beginning.



JIRA has its own elaborate attachment structure nowadays too - see <http://blog.valiantys.com/en/jira-en/million-jira-issues> for details.