Posted in: General (/spaces/127/sandpiper/forums/4997/general)

Grain Type "full-file"

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/3/2020

Hello, all, though most specifically **Doug Winsby (https://autocare.communifire.com/people/dougwinsby)** ! I'm building up the "master" slice arrangement and realized it might be better to have a slightly changed grain type.

Originally we had "pies-file", "aces-file", etc. -- when we were going to have multiple files as grains in a slice.

Since Sandpiper doesn't distinguish content, especially not at Level 1, I feel like the "ACES" or "PIES" of it is covered in the format and version (e.g. "Auto Care PIES" "7.1") anyways. And it would be beneficial to set up a single grain type that means "This is a full file, a single grain, master slice" essentially. So that any granulator coming in to do parsing can easily build logic around identifying full file slices that are candidates for granulation.

To that end I'd like to propose we just have a single grain type "full-file" or "file" for those, discarding pies-file, aces-file, partspro-file, etc.. When we get digital assets set we can do something that indicates these are a different category, perhaps "multi-file".

I'm writing this into the next rev of the documentation, but can drop it out however we decide. Just easier to get the thoughts in that way..

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/3/2020

So you're recommending the removal of "slice_type" from the `slices` table, right?

How would a receiver know what kind of data they're working with? Are you thinking it would be indicated as a key in `slice_metadata`?

There are currently no standards around slice_metadata key/value pairs (although we could add them). It would be a bit more difficult (and error-prone) to query by those values rather than something enforced by the database (in the form of an enum data type) and stored directly in the slices table.

I do like that we would not need to change the database to add a new slice_type, however.

slice_type (/spaces/127/sandpiper/searchresults?keyword=%22slice_type%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5887&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/3/2020

I was only thinking of the grain type -- perhaps for a full file it would be "none"? Or null? Since the grain type is on the slice it would more truthfully expose what the contents are.

However, maybe it does make sense to use a slice type for the granulator's work -- one way or another I felt that the grain types "aces-file" etc. are redundant or out of spec, since they describe the same state (a file) and the format / type of that file is specified elsewhere anyways (as part of the format and version).

Like

Reply (/forums/post?tid=5884&ReplyPostID=5889&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/3/2020

We no longer have a "grain_type". We moved that information to the slice (per your suggestion) and renamed it to "slice_type"

```
CREATE TABLE IF NOT EXISTS "slices" (
  "id"            uuid PRIMARY KEY,
  "name"          text UNIQUE NOT NULL,
  "slice_type"    slice_type_enum NOT NULL,
  "content_hash"  text,
  "content_count" integer,
  "content_date"  timestamp,
  "created_at"    timestamp,
  "updated_at"    timestamp
);
```

with these possible values (notice plurals are purposeful):

```
CREATE TYPE slice_type_enum AS ENUM (
  'aces-file',
  'aces-items',
  'asset-files',
  'pies-file',
  'pies-items',
  'pies-marketcopy',
  'pies-pricesheet',
  'partspro-file',
  'partspro-items'
);
```

Like

Reply (/forums/post?tid=5884&ReplyPostID=5890&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/3/2020

Ah, I see, I didn't pick up that detail. The point still stands then -- what I was referring to as the grain type is the slice type..

I think the partspro-item, pies-pricesheet, etc. still are necessary. It's just the partspro-file, pies-file, etc. that I'd like to do away with and replace with just "full-file".

Like

Reply (/forums/post?tid=5884&ReplyPostID=5891&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/3/2020

Sorry I haven't looked at your latest docs yet. Do you define the valid Slice Metadata Keys?

slice-metadata-keys (/spaces/127/sandpiper/searchresults?keyword=%22slice-metadata-keys%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5892&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/3/2020

No, I didn't -- just mentioned the "full-file" one (as a placeholder for this discussion, I can remove or modify -- probably bad development practice but we can keep it a little looser since it's only internal release). And I mentioned that the other types are granular but full-file is singular.

I will need to update the "Grain Type" mentions though! I will do that.

Like

Reply (/forums/post?tid=5884&ReplyPostID=5893&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/4/2020

Instead of using "full-file" for an L1 slice_type value, maybe slice_type should store the Sandpiper Level. For example, "L1" (which means full files).

The only problem I see with this is that we are again removing data integrity checks from the database. It becomes a convention to store the information that defines the slice in the slice meta-data.

Still, it's probably a move in the right direction (removing any content knowledge from the database structure itself).

content_knowledge (/spaces/127/sandpiper/searchresults?keyword=%22content_knowledge%22&searchtags=1)
slice_type (/spaces/127/sandpiper/searchresults?keyword=%22slice_type%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5899&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/5/2020

It might make sense to move `slice_type` into the metadata as well. You could then have "sandpiper-level": "1" as a key/value. It makes the check to see if we can add a grain to a slice slightly easier (just query the slice-metadata before every grain is added. If level 1, we can allow only one).

That's probably the biggest problem I'm running into by the decision to move content information out of the tables. It forces us to perform more validation queries that the database was handling before with unique keys.

data-integrity (/spaces/127/sandpiper/searchresults?keyword=%22data-integrity%22&searchtags=1)
validation (/spaces/127/sandpiper/searchresults?keyword=%22validation%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5903&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/5/2020

There's no way to enforce the "one grain per L1 slice" rule without extra queries. One way we can eliminate that overhead is by saying that the `grain_key` for a level 1 grain must always be "L1". (The sandpiper CLI can easily enforce this rule).

This rule would greatly simplify the add because we already have a unique key on (slice_id, grain_key).

If no one has an objection to this, that is how we will code it.

level-1-convention (/spaces/127/sandpiper/searchresults?keyword=%22level-1-convention%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5904&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/6/2020

> **On 4/5/2020 7:17 PM, Doug Winsby said:**
>
> There's no way to enforce the "one grain per L1 slice" rule without extra queries. One way we can eliminate that overhead is by saying that the `grain_key` for a level 1 grain must always be "L1". (The sandpiper CLI can easily enforce this rule).

I think I still would like the slice type to be on the slice directly -- like you said, this reduces the risk of data divergence due to implementation coding differences.

Although I do see what you're saying -- there is an advantage to disassociating any content awareness from the slice storage. I think, conceptually, having a proper type for the slice makes sense, as long as it doesn't actually try to interpret that data. For the same reason I still prefer "full-file" over “L1" because we're not mixing metaphors; it accurately describes the expected contents in the same scope as any other value, i.e. the type of grain it holds.

Like

Reply (/forums/post?tid=5884&ReplyPostID=5905&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/6/2020

> **On 4/6/2020 8:36 AM, Krister Kittelson said:**
>
> I think I still would like the slice type to be on the slice directly -- like you said, this reduces the risk of data divergence due to implementation coding differences.

I disagree. Since we have a structure to hold "information about the slice" (metadata), I think it makes more sense to have it there. I don't see how having some of that information directly on the slice helps reduce "data divergence".

> **On 4/6/2020 8:36 AM, Krister Kittelson said:**
>
> For the same reason I still prefer "full-file" over “L1" because we're not mixing metaphors

I was actually thinking we'd have both (as metadata keys): "sandpiper-level" and "content-type".

Like

Reply (/forums/post?tid=5884&ReplyPostID=5906&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/6/2020

> **On 4/6/2020 9:26 AM, Doug Winsby said:**
>
> I was actually thinking we'd have both (as metadata keys): "sandpiper-level" and "content-type".

Okay, if that seems secure enough then I'm all right with it. Let's just make sure we get the content type name and value list down so I can document it. So you're thinking the content-type would be like this?

```
CREATE TYPE content_type_enum AS ENUM (
  'full-file',
  'aces-items',
  'asset-files',
  'pies-items',
  'pies-marketcopy',
  'pies-pricesheet',
  'partspro-items'
);
```

Like

Reply (/forums/post?tid=5884&ReplyPostID=5909&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
4/6/2020

Yes, except they would not be an enum in the database. They would just exist in the API documentation.

I'm not sure about "partspro-items". We should probably only document the ones that are really available now. PartsPRO still requires full-files (although net-change files are supported).

content-type (/spaces/127/sandpiper/searchresults?keyword=%22content-type%22&searchtags=1)

Like

Reply (/forums/post?tid=5884&ReplyPostID=5910&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
4/6/2020

I agree about the partspro-items -- perhaps it can be adapted, but at the moment I think it'd have to be full rows, and no idea the best way to handle partspro delete records (meaning, not sandpiper generated but actual D records) except as a blind, full file. So I'll take that one off my list too, and get this new list into the documentation.

Like

Reply (/forums/post?tid=5884&ReplyPostID=5911&SpaceID=127) Answer

Page 1 of 1 (15 items)