Acrescenta informação sobre filtragem por id_situacao e troca pipe simples por dupla.
mateuswetah authored 1 week ago
b9bfbe71

script-preparacao-csv-importacao.md 15.3 KB

Script de preparação do CSV de Migração do SPUNet Acervo para o MemóriaSPU

Este documento busca detalhar o fluxo e etapas de transformações necessárias para se converter os dados disponíveis no Datalake que guarda os dumps do SPUNet Acervo em uma planilha CSV pronta para importação de dados no Tainacan do MemóriaSPU.

As análises a seguir foram feitas levando em conta o Modelo Entidade Relacionamento do SPUNet Acervo e consultas feitas diretamente ao Datalake.

Há também uma planilha de orientação que acompanha esta documentação. Na página "Campos Script Migração SPUNET", estão listados todos os campos que devem estar presentes no CSV (exceto os anexos) e alguns sinalizadores que indicam se as transformações pedidas abaixo se aplicam ou não.

Filtragem

A base total de itens do SPUNet Acervo possui mais de 69 mil itens. Somente deverão ser enviados para o MemóriaSPU aqueles que foram revisados para publicação na Web, ou seja, aqueles cujo campo `ic_publicado_internet` for explicitamente `true`. Também existem alguns poucos itens que estão "excluídos" (`id_situacao 2`) no sistema que não devem ir.

```
tb_arquivo_acervo.ic_publicado_internet = true AND tb_arquivo_acervo.id_situacao != 2
```

Estes dois campos entretanto não precisam estar presentes no CSV final.

Consulta aos dados

O objetivo do script é gerar um CSV onde cada linha corresponda à um item da tabela `tb_arquivo_acervo`.

Vários campos da tabela são índices para outras tabelas onde de fato está a descrição, o valor desejado. No CSV, queremos quase sempre os valores derivados ao invés dos índices. Por exemplo, ao invés do campo `id_categoria`, queremos o campo `ds_categoria`, presente na tabela `tb_categoria`. Representaremos esta situação na consulta na seguinte forma:

Origem	Campo
`tb_arquivo_acervo > id_categoria > tb_categoria`	`ds_categoria`

Há também campos que serão derivados com um ou mais saltos em tabelas complementares. É o caso de tabelas como a `tb_descritor_catalografico` `tb_descritor_catalografico_iconografico`, `tb_descritoes_informacionais` e `tb_descritoes_informacionais_iconografico` (ficar atento aos erros de escrita como `descritoes`, estão assim mesmo no banco). Por exemplo o Campo `dt_original` da tabela `tb_descritoes_informacionais_iconografico` deve estar presente na linha do item e para acessá-lo é preciso consultar o `id_descritoes_informacionais_iconografico`, que leva a `tb_arquivo_acervo` até a `tb_descritoes_informacionais_iconografico`. Neste caso representaremos esta consulta da seguinte forma:

Origem	Campo
`tb_arquivo_acervo > id_descritoes_informacionais_iconografico > tb_descritoes_informacionais_iconografico`	`dt_original`

A planilha final não precisa conter o campo `id_descritoes_informacionais_iconografico` neste caso, só precisamos dele para este tipo de inferência.

Há casos onde o próprio dado de tabela complementar é derivado de índices, formando saltos mais longos. Por exemplo, queremos o campo `ds_especie_documental`. No caso dele será:

Origem	Campo
`tb_arquivo_acervo > id_descritor_informacional > tb_descritoes_informacionais > id_especie_documental > tb_especie_documental`	`ds_especie_documental`

Por fim, a tabela também possui chaves estrangeiras, em geral usadas para mostrar relacionamentos multivalorados. Neste caso, a outra tabela é que possuirá um ID vinculando a tabela que temos acesso. Por exemplo, queremos o campo `ds_processo_vinculado`. Este campo é referenciado por um `id_descritoes_informacionais_iconografico`, que está presente tanto na tabela `tb_descritoes_informacionais_iconografico` quanto na `tb_arquivo_acervo`:

Origem	Campo
`tb_arquivo_acervo > id_descritoes_informacionais_iconografico > tb_descritoes_informacionais_iconografico < id_descritoes_informacionais_iconografico < tb_processo_vinculado`	`ds_processo_vinculado`

A lista completa de quais sampos são desejados na consulta final está na planilha de orientação. Ao total são cerca de uma centena de campos.

Padronização de dados em branco

O banco possui uma série de inconsistências de dados, a mais clara sendo uma mistura de representação de campos vazios. Para resolver isso precisamos que **todos os campos***, independentemente do tipo, sejam padronizados para células vazias no CSV. Exemplos de conteúdos que devem ser considerados vazio:

- `NULL`
- `null`
- `NaN`
- `NAN`
- `NA`
- `N/A`
- `BLANK`
- `Blank`
- (espaço em branco puro)

Note que valores como `0`, `false`, `FALSE` devem continuar presentes (numéricos são válidos e booleanos serão tradados adiante).

Padronização de datas

Os campos que forem do tipo data (indicados na planilha de orientação pela coluna DATE e em geral prefixados por `dt_`) devem ter seus dados padronizados para o formato ISO 8601 `YYYY-MM-DD`. Entretando há alguns campos de data que possuem valores inválidos, como por exemplo o `dt_copia` que possui valores como `0/0/0` ou `0/0`. Estes casos devem ser convertidos para células em branco.

Padronização de dados multivalorados

Todos os valores multivalorados (como os gerados por chaves estrangeiras mencionados antes) devem ser concatenados em uma única célula do campo específico. Nesta célula eles devem estar separados por `||` (pipe dupla) sem a necessidade de espaços ao redor. Por exemplo:

co_protocolo_acervo	ds_palavra_chave
`2019110017484`	`PCERTT||SESMARIA||FNSC`

Estes metadados estão referenciados na planilha de orientação pela coluna MULTIVALORADO

Padronização de dados "flags"

Há alguns campos no banco que se comportam como flags. Eles estão sinalizados na planilha de orientação pela coluna FLAGS e em geral começam com o prefixo `ic_`. A maioria destes é booleanos, embora hajam exceções (`ic_iconografia`, por exemplo). Nestes casos a orientação é converter para Strings simples:

- `false`, `FALSE`, `0`, `N` -> `Não`
- `true`, `TRUE`, `1`, `S` -> `Sim`

Note que a regra anterior de verificação de vazios já deve ter retirado valores como `NULL` neste caso, deixando-os em branco.

Unificação de campos "flags"

Em alguns casos raros queremos substituir a representação "Sim/Não" de flags complementares por um único campo. Hoje um exemplo disto são os campos `ic_tiff`, `ic_pdf` e `ic_jpeg` da tabela `tb_descritores_tecnicos`. Ao invés de três campos "flags", queremos unificá-los em um único campo `ds_formato_extensao` que poderá ter os valores `TIFF`, `PDF` e `JPEG`. Caso nenhum dos três booleanos seja true, este campo deverá vir vazio.

Mesclagem de dados redundantes

Em vários casos no banco de dados os itens possuem seus dados expandidos pelas tabelas de descritores informacionais e descritores catalográficos. Porém há sempre duas "versões" destas tabelas, uma para os itens cujo campo `id_categoria` é `1` (`ds_categoria` = `Textual`) e outra para os itens cujo campo `id_categoria` é `2` (`ds_categoria` = `Iconográfico`), estas últimas geralmente com o nome terminando em `_iconografico`. Sabemos que esta é uma relação excludente e obrigatória, ou seja, ou é um ou é outro.

De início isto não gera problemas, podemos ter os campos vazios naquelas onde a relação não ocorre. Porém notamos que vários campos que estão presentes em uma tabela possuem versões equivalentes na outra, levando a campos distintos que na verdade poderiam ser o mesmo campo (e mais idealmente pertencer diretamente à tabela `tb_arquivo_acervo`). Nestes casos queremos unificar os campos em um só. Por exemplo, o campo `id_municipio` existe tanto na `tb_descritoes_informacionais` quanto na `tb_descritoes_informacionais_iconografico`, mas queremos apenas um campo `ds_municipio` com o nome do Município derivado daquele `id_municipio`.

Estes campos que demandam mesclagem estão idenficados na planilha de orientação pela coluna MESCLAGEM DE DADOS REDUNDANTES. Na maioria dos casos, o nome do campo nas duas tabelas será o mesmo.

Construção dos dados de anexos

Os anexos também são dados multivalorados e vêm da tabela `tb_arquivo_anexo` através da chave estrangeira `id_arquivo_acervo`. Existem vários campos descritivos dos anexos nesta tabela mas não temos interesse em levá-los ao MemóriaSPU. O dado relevante aqui é o `id_arquivo_anexo`, porque através dele deve ser obtida a URL para o link público do anexo durante a importação.

No Tainacan, há o conceito de "Documento principal". Convencionaremos aqui que o primeiro anexo vinculado à um item será o Documento principal. Neste caso, sua URL deverá estar na coluna `special_document`. Para os demais anexos (caso haja), usaremos a coluna `special_attachments|REPLACE`. Por exemplo, caso um item da tabela `tb_arquivo_acervo` tenha vinculado ao seu `id_arquivo_acervo`, os anexos de `id_arquivo_anexo` `123`, `124`, `125` e `126`, teremos as seguintes colunas no CSV:

special_document	special_attachments|REPLACE
`file:https://link-para-anexos/123.pdf`	`https://link-para-anexos/124.pdf||https://link-para-anexos/125.pdf||https://link-para-anexos/126.pdf`

Isto é claro supondo que as URLs públicas fiquem dispostas desta forma.

> O prefixo `file:` só é necessário nos valores coluna `special_document` (isto porque no Tainacan, documentos podem ser de diferentes tipos). O sufixo `|REPLACE` no cabeçalho da coluna de anexos é para garantir que sejam substituídos ao invés de criados novos em futuras importações. Note também o uso do `||` novamente para separar os campos multivalorados.

Construção dos auto-relacionamentos

Há dois casos de "auto-relacionamentos" no acervo, onde itens do acervo se relacionam com outro itens. Um é através do campo `id_arquivo_acervo_pai`. Se um item for "filho" de outro, ele terá ali o `id_arquivo_acervo` do pai. Outra é através da tabela `tb_arquivo_acervo_associado`, que estabelece uma relação N:N, podendo ser consultada de forma bidirecional.

No Tainacan é possível mostrar estes relacionamentos através de dois metadados de Relacionamento configurados para a própria coleção. Nestes casos, o dado guardado é o ID do item no banco do WordPress. O problema é que no cenário inicial de importação, não teremos estes IDs (apenas o `id_arquivo_acervo_pai`). De modo que o preenchimento dos relacionamentos precisá ser feito em uma planilha CSV diferente, montada após a primeira importação.

Esta planilha será inicialmente gerada a partir da exportação de dados do próprio Tainacan, usando o Exportador CSV. O Exportador trás vários campos, mas precisamos apenas de algumas colunas. Por exemplo:

special_item_id	ID do SPUNET	Arquivo Pai	Arquivos Associados
`123`	`93845`		
`124`	`63542`		
`125`	`63225`		
`126`	`85431`		

Neste caso, *ID do SPUNET*, *Arquivo Pai* e *Arquivos Associados* seriam os nomes dos metadados criados no Memória SPU. Uma vez populado pela primeira importação, teremos tanto o ID no banco de dados do Tainacan quanto o ID no banco de dados do SPUNet Acervo.

Será que necessário então um script que mapeie:

Se há `id_arquivo_acervo_pai` no item de `id_arquivo_acervo` X, pesquisar o `special_item_id` cuja coluna *ID do SPUNET* seja igual ao valor de `id_arquivo_acervo_pai` e caso encontrar colocar na coluna *Arquivo Pai*. Se há item de `id_arquivo_acervo` X no campo `id_arquivo_acervo_1` da tabela `tb_arquivo_acervo_associado`, pesquisar o `special_item_id` cuja coluna *ID do SPUNET* seja igual ao valor de `id_arquivo_acervo` e caso encontrar, pesquisar pelo `special_item_id` associado ao valor da `id_arquivo_acervo_2`, para colocá-lo na coluna *Arquivos Associados*. Caso haja mais de um, concatenar os ids separados por `||`.

Exemplo de uma planilha atualizada:

special_item_id	ID do SPUNET	Arquivo Pai	Arquivos Associados
`123`	`93845`	`124`	
`124`	`63542`		`125`
`125`	`63225`		`124||126`
`126`	`85431`		`125`

Como os dados são oriundos de uma adaptação da planilha exportada, este script pode ser incorporado na lógica de atualização que descreveremos a seguir.

Atualização de dados

Uma vez feita a primeira importação, os dados do SPUNet acervo serão atualizados de tempos em tempos e novos itens passarão a ter o campo `ic_publicado_internet` definido como `true`. Porém é esperado também que correções sejam feitas e nestes casos **os dados já importados para o MemóriaSPU precisarão ser sobrescritos**.

Para o importador CSV do Tainacan, a presença da coluna `special_item_id` dentro do CSV informa que se aquela célula não está vazia, aquele item já existe no Banco de dados do Tainacan e neste caso deve ser atualizado ao invés de criado. Se algum valor estiver em alguma coluna da nova planilha, ele será sobrescrito.

Mesmo assim, seria importante ser elaborada uma estratégia para enviar nestas novas importações somente itens que precisem de fato ser importados, visto que o processo pesa (em especial se mantivermos a substituição completa de anexos). O ideal para isso seria a existência de um campo no banco de dados do SPUNet Acervo que guardasse a data da última modificação. Este campo poderia então ser comporado com o campo `modification_date` que já é gerado em exportações do Tainacan. Porém hoje este campo não existe. Se não for possível demandar ele uma possível solução seria implementar e guardar uma hash com os dados de cada item.