Opzet gebruik SOLR

# SOLR conventies

Een input document voor indexering door SOLR ziet er bijvoorbeeld als volgt uit:

<doc>

<field name="id">3245r</field>

<field name="ocn\_s">3245r</field>

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

<field name="year\_txt">ca. 1930</field>

<field name="year\_dr">1930</field>

<field name="listEntry">{"title":"De titel: subtitel","author": "Jan","year": "ca. 1930"}</field>

</doc>

Wat er met een veld moet worden gedaan staat in schema.xml, onder dezelfde ‘name=’. Voorbeeld uit schema.xml:

<field name="id" type="string" indexed="true" stored="true" multiValued="false"/>

<field name="text" type="text\_general" indexed="true" stored="false"/>

<field name="listEntry" type="text\_general" indexed="false" stored="true" multiValued="false"/>

<field name="detailEntry" type="text\_general" indexed="false" stored="true" multiValued="false"/>

<dynamicField name="\*\_i" type="pint" indexed="true" stored="true"/>

<dynamicField name="\*\_is" type="pints" indexed="true" stored="true"/>

<dynamicField name="\*\_s" type="string" indexed="true" stored="true" />

<dynamicField name="\*\_ss" type="strings" indexed="true" stored="true"/>

<dynamicField name="\*\_t" type="text\_general" indexed="true" stored="true" multiValued="false"/>

<dynamicField name="\*\_txt" type="text\_general" indexed="true" stored="true"/>

<dynamicField name="\*\_dr" type="rdate" indexed="true" stored="true"/>

<dynamicField name="\*\_drs" type="rdate" indexed="true" stored="true" multiValued="true"/>

Let op: moet SOLR ingesteld worden op UTF-8? Lijkt me gewenst.

## Field name convention

Om het schema simpel te houden worden field name conventies gebruikt. Het komt erop neer dat een groot aantal velden niet expliciet in het schema wordt opgenomen, maar onder een veldnaam met een achtervoegsel. In het voorbeeld hierboven zijn maar 4 velden expliciet opgenomen, de rest is “dynamisch”. Zie de voorbeelden onder de tabel.

Hieronder staan de conventies die ik heb gehanteerd:

|  |  |
| --- | --- |
| \*\_i | Integer veld, niet herhaald |
| \*\_is | Herhaald integer veld |
| \*\_s | String veld, in z’n geheel geïndexeerd |
| \*\_ss | Herhaald string veld, elke occurence in z’n geheel geïndexeerd |
| \*\_t | Tekst veld, wordt woord voor woord geïndexeerd |
| \*\_txt | Herhaald tekst veld, elke occurence woord voor woord geïndexeerd |
| \*\_dr | Datum veld |
| \*\_drs | Herhaald datumveld |

Voorbeelden:

* Als je de veldnamen isbn\_ss en issn\_ss in de input gebruikt, dan leg je daarmee al vast dat het een herhaald veld kan zijn en dat deze in z’n geheel worden geïndexeerd
* Hetzelfde voor title\_txt, table\_of\_contents\_txt, enz.

Zie ook de paragraaf: Het SOLR schema

En zie: <https://github.com/FritsvanLatum/OCLC_Services/tree/master/SOLR>

# Werkwijze 1: aparte tabellen voor BIB, LBD en LHR

~~Er zijn meerdere input bestanden: BIB, LBD en LHR, THES~~

~~Voorstel: maak voor elk inputbestand 1 tabel in de database D. De tabelstructuur is dan:~~

Het inputbestand bestaat uit losse marc records. Per ocn zijn er meerdere records: een BIB record, gevolgd door een LBD record, gevolgd door 1 of meer LHR records. Deze werkwijze leest steeds 1 record in en kijkt in welke tabel het record thuis hoort. Consequentie: het bij elkaar zoeken van de records voor indexering in SOLR moet achteraf gebeuren.

Zie ook:

## Tabelstructuur

database D:

table: D.bib

column: D.bib.ocnPlus //main key

column: D.bib.ocn //indexed

column: D.bib.ppn //indexed

column: D.bib.data //een titelbeschrijving in SOLR XML

table: D.LBD

column: D.LBD.lbdnumPlus //main key

column: D.LBD.ocn //indexed

column: D.LBD.data //LBD fields in SOLR XML

table: D.LHR

column: D.LHR.lhrnumPlus //main key

column: D.LHR.ocn //indexed

column: D.LHR.data //LHR fields in SOLR XML

table: D.THES

column: D.THES.ppnPlus //main key

column: D.THES.ppn //indexed

column: D.THES.data //THES fields in SOLR XML

table: D.links

column: D.links.fromOcnPlus //indexed

column: D.links.toOcnPLUS //indexed

column: D.links.linktype //indexed

main key (if required): fromOcnPlus + linktype + toOcnPlus

## Werkwijze stap voor stap:

Belangrijk: de data zijn meestal ok, maar kunnen inconsistent, incompleet en fout zijn….

1. ~~Lees elk inputbestand in in z’n tabel.~~ Lees het inputbestand in en sla – na bewerking, zie hieronder – elk record op in de tabel waar dat record thuis hoort. Doe dat eventueel batchgewijs (splits het inputbestand in brokken).
2. Converteer per record MARC XML naar SOLR XML. Daarvoor is een ‘traverse’ nodig van de MARC XML en een XPath-achtige functie om bij een veld een ander veld te zoeken. Gebruik bijvoorbeeld simpleXML of DOM.
3. Voor BIB
   1. Behandel het OCN. Kijk of het OCN eerder is ingelezen.

Zo nee: voer het OCN in in D.bib.ocnPlus en in D.bib.ocn

Zo ja: voer het OCN + volgnummer in in D.bib.ocnPlus en het OCN in D.bib.ocn. Gebruik SELECT ocn FROM D.bib WHERE ocn=OCN om het aantal eerder ingevoerde records met hetzelfde OCN te weten.

* 1. Start het SOLR XML record, bijvoorbeeld:

<field name="id">ocnPlus</field>

<field name="ocn\_s">ocn</field>

Merk op dat het start element <doc> ontbreekt, dat komt later.

* 1. Loop door de MARC XML velden. Er zijn meerdere mogelijkheden:
     1. Neem “as is” over in SOLR XML, bijvoorbeeld:

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

* + 1. Verbeter en neem over in SOLR XML, bijvoorbeeld:

<field name="language\_txt">Taal voluit</field>

* + 1. Neem over en maak een apart veld met een verbeterde versie:

<field name="year\_txt">ca. 1930</field>

<field name="year\_dr">1930</field>

* + 1. Combineer met andere velden, plak velden aan elkaar, leid velden af uit andere velden, etc.
    2. Neem links op in de tabel links
  1. Sla het SOLR XML record in z’n geheel op in bib.data

1. Voor LBD: doe analoog hetzelfde, maar:
   1. Zorg dat eventuele PPN’s apart bewaard worden:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

1. Voor LHR: doe analoog hetzelfde
2. Voor THES: doe analoog hetzelfde, hopelijk is de export ook in een XML format.
3. Loop door D.bib, stel per record het definitieve SOLR XML record samen t.b.v. invoer:
   1. Start het definitieve SOLR XML record:

<doc>

* 1. Neem het XML record uit D.bib.data over:

<doc>

<field name="id">ocnPlus</field>

<field name="ocn\_s">ocn</field>

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

<field name="year\_txt">ca. 1930</field>

<field name="year\_dr">1930</field>

…

* 1. Zoek het bijbehorende LBD record (eventueel meerdere?) m.b.v. het ocn (SELECT data FROM D.lbd WHERE ocn=OCN) en plak de data uit D.lbd.data aan het definitieve SOLR XML record
  2. Doe hetzelfde met de LHR’s
  3. Verrijk de thesaurus termen, ga voor elk veld van de vorm:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

Het ppn opzoeken in D.thes en plak de inhoud van D.thes.data in het definitieve SOLR XML record.

* 1. Maak de inhoud van het SOLR veld listEntry:

Verzamel alle relevante veldinhouden, voeg ze samen in een vorm die handig is voor later gebruik bij het opmaken van de lijst van gevonden titels op de website. Bijvoorbeeld in JSON: title:de titel,author:de author,… (quotes voor het gemak weggelaten) en voeg toe:

<field name="listEntry">…</field>

Het mag ook yaml zijn of iets anders, maar geen XML omdat dan SOLR in de war raakt.

* 1. Maak de inhoud van het SOLR veld detailEntry:

Analoog aan listEntry. Maar met meer velden.

* 1. Behandel de eventuele links:

SELECT \* FROM D.links WHERE fromOcnPlus=OCN

Voeg voor elk resultaat van deze query toe:

<field name="D.links.linktype".’\_ss’”>D.links.toOcnPLUS</field>

Voorbeeld:

<field name="chapInBook\_ss">123456</field>

SELECT \* FROM D.links WHERE toOcnPLUS=OCN

Voeg voor elk resultaat van deze query toe:

<field name="inverse(D.links.linktype).’\_ss’">D.links.fromOcnPlus</field>

Voorbeeld:

<field name="bookHasChap\_ss">654321</field>

* 1. GEAVANCEERD: Zorg ervoor dat je alle VIAF nummers bewaard (analoog aan 4.1). Ga voor elke VIAF link de VIAF database bevragen (VIAF\_Service.php) en voeg relevante content toe aan het definitieve SOLR XML record.
  2. Sluit het definitieve SOLR XML record af:

</doc>

* 1. Stuur het record naar SOLR

# Werkwijze 2: één tabel voor gecombineerde BIB, LBD en LHR

Het inputbestand bestaat uit losse marc records. Per ocn zijn er meerdere records: een BIB record, gevolgd door een LBD record, gevolgd door 1 of meer LHR records. Deze werkwijze zoekt eerst de bijbehorende records bij elkaar (dubbele while loop), bewerkt het zo tot stand gekomen record en slaat het dan op in de tabel. Consequentie: het inlezen is complexer, maar het bij elkaar zoeken van de records voor indexering in SOLR, zoals in werkwijze 1 stap 7, is hier overbodig.

## Tabelstructuur

database D:

table: D.all

column: D.all.ocnPlus //main key

column: D.all.ocn //indexed

column: D.all.ppn //indexed

column: D.all.data //een titelbeschrijving in SOLR XML

table: D.THES

column: D.THES.ppnPlus //main key

column: D.THES.ppn //indexed

column: D.THES.data //THES fields in SOLR XML

table: D.links

column: D.links.fromOcnPlus //indexed

column: D.links.toOcnPLUS //indexed

column: D.links.linktype //indexed

main key (if required): fromOcnPlus + linktype + toOcnPlus

## Werkwijze stap voor stap:

Belangrijk: de data zijn meestal ok, maar kunnen inconsistent, incompleet en fout zijn….

1. Lees het inputbestand in en combineer het BIB records met de bijbehorende LBD en LHR records in een tijdelijk XML record..
2. Converteer per zo gecombineerd record MARC XML naar SOLR XML. Daarvoor is een ‘traverse’ nodig van het gecombineerde record en een XPath-achtige functie om bij een veld een ander veld te zoeken. Gebruik bijvoorbeeld simpleXML of DOM.
   1. Behandel het OCN. Kijk of het OCN eerder is ingelezen.

Zo nee: voer het OCN in in D.bib.ocnPlus en in D.bib.ocn

Zo ja: voer het OCN + volgnummer in in D.bib.ocnPlus en het OCN in D.bib.ocn. Gebruik SELECT ocn FROM D.bib WHERE ocn=OCN om het aantal eerder ingevoerde records met hetzelfde OCN te weten.

* 1. Start het SOLR XML record, bijvoorbeeld:

<doc>

<field name="id">ocnPlus</field>

<field name="ocn\_s">ocn</field>

* 1. Loop door de MARC XML velden. Er zijn meerdere mogelijkheden:
     1. Neem “as is” over in SOLR XML, bijvoorbeeld:

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

* + 1. Verbeter en neem over in SOLR XML, bijvoorbeeld:

<field name="language\_txt">Taal voluit</field>

* + 1. Neem over en maak een apart veld met een verbeterde versie:

<field name="year\_txt">ca. 1930</field>

<field name="year\_dr">1930</field>

* + 1. Combineer met andere velden, plak velden aan elkaar, leid velden af uit andere velden, etc.
    2. Neem altijd PPN over

<field name="ppn\_s">PPN</field>

* + 1. Neem links op in de tabel links
    2. Zorg dat eventuele thesaurus PPN’s apart bewaard worden:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

* 1. Verrijk de thesaurus termen, ga voor elk veld van de vorm:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

Het ppn opzoeken in D.thes en plak de inhoud van D.thes.data in het definitieve SOLR XML record.

* 1. Maak de inhoud van het SOLR veld listEntry:

Verzamel alle relevante veldinhouden, voeg ze samen in een vorm die handig is voor later gebruik bij het opmaken van de lijst van gevonden titels op de website. Bijvoorbeeld in JSON: title:de titel,author:de author,… (quotes voor het gemak weggelaten) en voeg toe:

<field name="listEntry">…</field>

Het mag ook yaml zijn of iets anders, maar geen XML omdat dan SOLR in de war raakt.

* 1. Maak de inhoud van het SOLR veld detailEntry:

Analoog aan listEntry. Maar met meer velden.

* 1. Behandel de eventuele links:

SELECT \* FROM D.links WHERE fromOcnPlus=OCN

Voeg voor elk resultaat van deze query toe:

<field name="D.links.linktype".’\_ss’”>D.links.toOcnPLUS</field>

Voorbeeld:

<field name="chapInBook\_ss">123456</field>

SELECT \* FROM D.links WHERE toOcnPLUS=OCN

Voeg voor elk resultaat van deze query toe:

<field name="inverse(D.links.linktype).’\_ss’">D.links.fromOcnPlus</field>

Voorbeeld:

<field name="bookHasChap\_ss">654321</field>

* 1. GEAVANCEERD: Zorg ervoor dat je alle VIAF nummers bewaard (analoog aan 4.1). Ga voor elke VIAF link de VIAF database bevragen (VIAF\_Service.php) en voeg relevante content toe aan het definitieve SOLR XML record.
  2. Sluit het definitieve SOLR XML record af:

</doc>

* 1. Stuur het record naar SOLR

1. Voor THES: doe analoog hetzelfde, hopelijk is de export ook in een XML format.

# Het SOLR schema

Het SOLR schema ziet er globaal als volgt uit:

**<schema>**

**<types>**

**<fields>**

**<uniqueKey>**

**<copyField>**

**</schema>**

De file ../server/solr/configsets/\_default/conf/managed-schema is een uitgewerkt én goed gedocumenteerd schema. Het idee is dat deze file gekopieerd wordt naar schema.xml en wordt aangepast.

Zie: <https://github.com/FritsvanLatum/OCLC_Services/tree/master/SOLR> voor een door mij alvast aangepaste schema.xml

# Zoeken

## Het zoekscherm

Het zoekscherm bestaat uit de volgende onderdelen:

1. Query input met de query zoals ingevoerd door de gebruiker
2. Header, met aantal gevonden, volgende/vorige pagina, etc.
3. Facets
4. Details

Bij elke zoekactie genereert SOLR een zoekresultatenlijst met alle gegevens. Dus ook **listEntry’s** en **detailEntry’s** worden aangeleverd.

Voorbeeld:

{

"responseHeader":{

"status":0,

"QTime":0,

"params":{

"q":"\*:\*"

}

},

"response":{

"numFound":4,

"start":0,

"docs":

[

{

"id":"3234",

"ocn\_s":"3234",

"title\_txt":["De titel",

"Subtitel",

"Nog een subtitel"],

"year\_txt":["ca. 1930"],

"listEntry":["\"title\":\"De titel: subtitel\",\"author\": \"Jan\""]

},

{

"id":"3245",

"ocn\_s":"3245",

"title\_txt":[

"De titel",

"Subtitel",

"Nog een subtitel"

],

"year\_txt":["ca. 1930"],

"listEntry":["\"title\":\"De titel: subtitel\",\"author\": \"Jan\",\"year\": \"ca. 1930\""]

},

{

"id":"3234r",

"ocn\_s":"3234r",

"title\_txt":["De titel",

"Subtitel",

"Nog een subtitel"],

"year\_txt":["ca. 1930"],

"year\_rt":[1930],

"listEntry":["\"title\":\"De titel: subtitel\",\"author\": \"Jan\""]

},

{

"id":"3245r",

"ocn\_s":"3245r",

"title\_txt":["De titel",

"Subtitel",

"Nog een subtitel"],

"year\_txt":["ca. 1930"],

"year\_rt":[1931],

"listEntry":["\"title\":\"De titel: subtitel\",\"author\": \"Jan\",\"year\": \"ca. 1930\""

]

}

]

}

}

De listEntry’s zijn handig voor het snel opmaken van de lijst.

Wanneer op een element in de lijst wordt geklikt kan het detailscherm worden opgemaakt door de bijbehorende detailEntry te vertonen.

## Dynamische verrijking lijst en details

Als het SOLR zoekresultaat binnen is, dan moeten er via de PHP libraries *Availability\_Service* en *WorldCat\_KB\_Service* gegevens uit WMS worden opgehaald. Dit moet voor elk item in de lijst gebeuren. Om dit redelijk snel te kunnen doen lijken mij Ajax calls vanuit de pagina handig. Het scherm kan dan worden opgebouwd en elke keer als een antwoord van WMS binnenkomt kan dat worden toegevoegd.

Werkwijze:

require\_once './Availability\_Service.php';

$avail = new Availability\_Service('keys\_availability.php');

$holding = $avail->get\_circulation\_info($ocn)

Analyseer de $holding array om gegevens te vinden over de fysieke beschikbaarheid.

require\_once './WorldCat\_KB\_Service.php';

$KB = new WorldCat\_KB\_Service('keys\_worldcat\_kb.php');

$found = $KB->search\_kb\_record($ocn);

//delete the previous line if only the link is needed

$href = $KB->getlink($ocn);

// complete response in $KB->kb\_record

Zie *test\_dynamic\_fields.php* in <https://github.com/FritsvanLatum/OCLC_Services/tree/master/TEST> en met name het commentaar over AJAX calls.

Als de $href een lege string is dan is er geen digitale versie geregistreerd in de WorldCat Knowledge Base.

## Zoeken

Het lijkt mij handig om een class te maken voor het zoekscherm: zie in <https://github.com/FritsvanLatum/OCLC_Services/tree/master/SOLR>: searchPage.php en test\_SOLR.php.

Voor het sturen van zoekopdrachten naar SOLR is inzicht in de URL parameters noodzakelijk. Deze worden gebruikt voor sorteren, pagineren van de zoekresultaten lijst, facetten, enzovoort.

Zie <https://lucene.apache.org/solr/guide/7_3/common-query-parameters.html> voor alle parameters die mogelijk zijn in de zoekopdracht van SOLR.

Zie bijvoorbeeld <https://gist.github.com/mankyKitty/5906859> voor facets in de query.