ETL Processes for Web of Science data

Dhiraj Eadara 857136

MS Academic Research

Which data is extracted?

Microsoft Academic Search is a public search engine for academic/scientific literature developed by Microsoft. This search engine contains the following information about publications.

- Title and ID of the publication
- Names and IDs of the authors
- Conference that are related to the publication
- Journals which published the publication
- The year of the beginning of publication
- DOI code
- Type of publication
- List of keywords and their IDs, etc.

The search engine also contains the following information about the author.

- ID and Name
- Number of Publications
- Number of Citations
- Names of interest domains

The information related to keywords

- Name and ID of the keyword
- Number of publications in the database containing the keyword
- Number of citations citing publications containing the keyword in the database

The information related to research interest domains

- Name and ID of the domains
- Number of publications in the database containing the domains
- Number of citations citing publications containing the domains in the database

The search engine also contains other information

Interface Description

The MS API provides two interfaces to access the data. The method of authentication is the ID provided by the administrators of the search engine. The two interfaces are GET and SOAP. I chose the GET method, since this method was the easiest and most convenient way to gain access to the information.

An example of a URL is as follows:

http://academic.research.microsoft.com/json.svc/search?AppId=Your_AppID&FullTextQuery=data+mining&ResultObjects=publication&PublicationContent=title,author&StartIdx=1&EndIdx=1

The parameters are passed in the fields of the URL. Here the AppId, FullTextQuery, ResultObjects, PublicationContent, StartIdx, EndIdx.

Request parameters are used to retrieve specific information. The more the fields are filled, the more accurate the information retrieved. The Request parameters can be as follows:

- AppID
- PublicationIDAuthorID
- ConferenceID
- JournalID
- OrganizationID
- DomainID
- SubDomainID
- KeywordID
- AuthorQuery
- ConferenceQuery
- JournalQuery
- FulltextQuery
- ResultObjects
- ReferenceType
- PublicationContent
- OrderBy
- YearStart
- YearEnd
- StartIdx
- EndIdx

The arrangement and the type of information retrieved can also be controlled by using parameters called ResultObjects, ObjectType, ReferenceRelationship, Response and PublicationType.

ResultObjects

- Publication
- Author
- Conference
- Journal
- Organization
- Domain

- Keyword

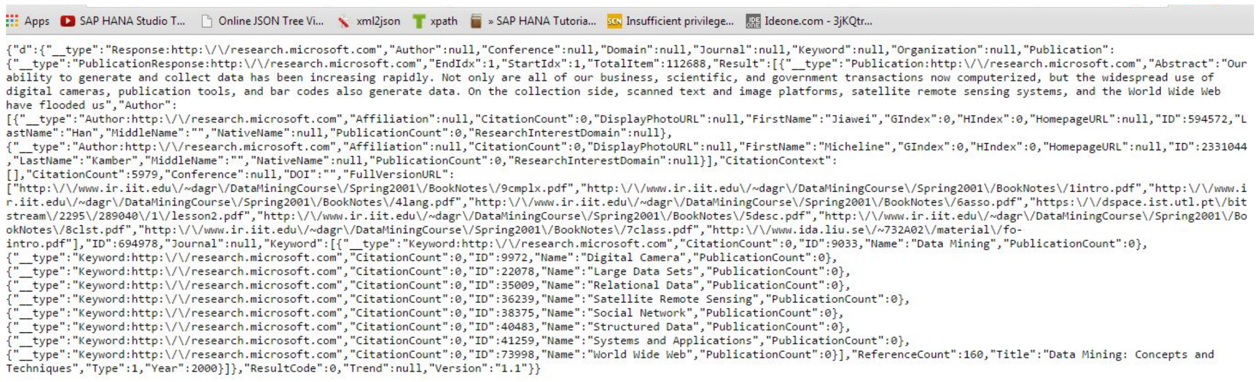
<u>ReferenceRelationship</u>
- None --- To get the object itself
- Reference --- To get the object cited by this object
- Citation --- To get the objects which cited this object

<u>PublicationContentType</u>

- MetaOnly --- Only to get the id, citation number, ids of authors and year of the publication
- Title --- To get all the MetaOnly properties and title of the publication
- Author --- To get all MetaOnly properties and the author names of the publication
- Abstract --- To get all MetaOnly properties and the abstract of the publication
- ConferenceAndJournalInfo --- To get all MetaOnly properties and conference/journal info of the
- publication
- FullVersionURL --- To get all MetaOnly properties and the download URL list
- AllInfo --- To get all the properties of the publication
- Keyword --- To get all the keywords of the publication

<u>OrderType</u>
- Rank --- The default ranking order, which considers the relevance between object and query as
- well as the importance of the object.
- Year --- Sort by the year of publication
- CitationCount --- Sort by the total number of citations
- HIndex --- Sort by H-Index
- GIndex --- Sort by G-Index

The retrieval is in the form of a JSON file. An example is provided below

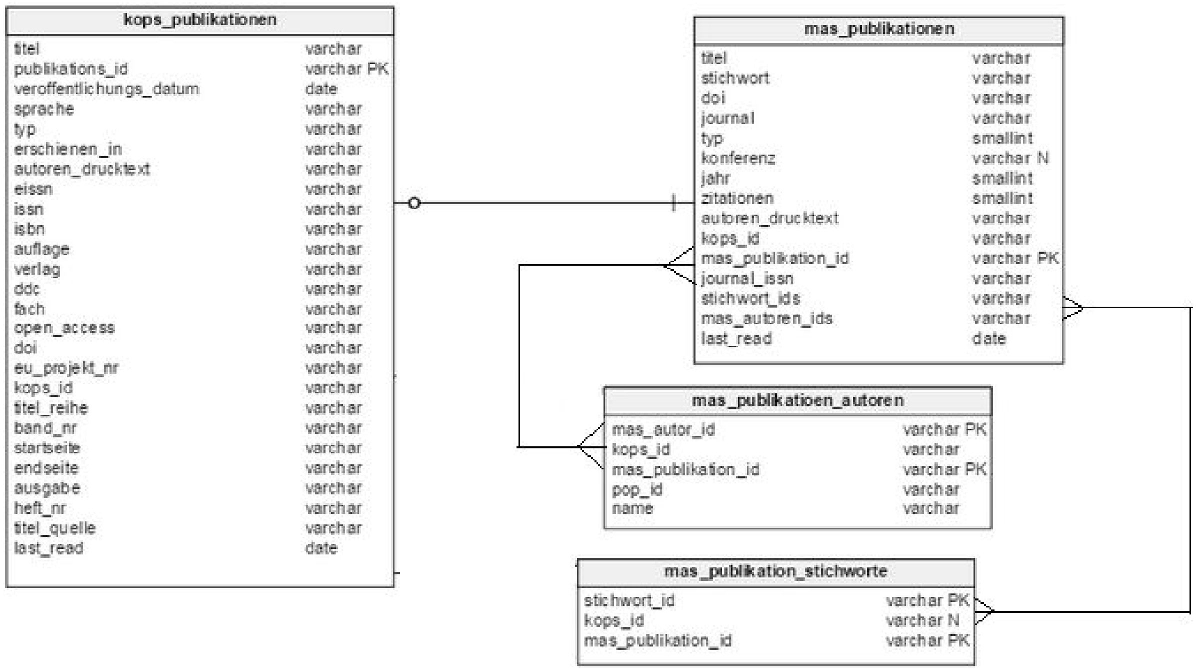


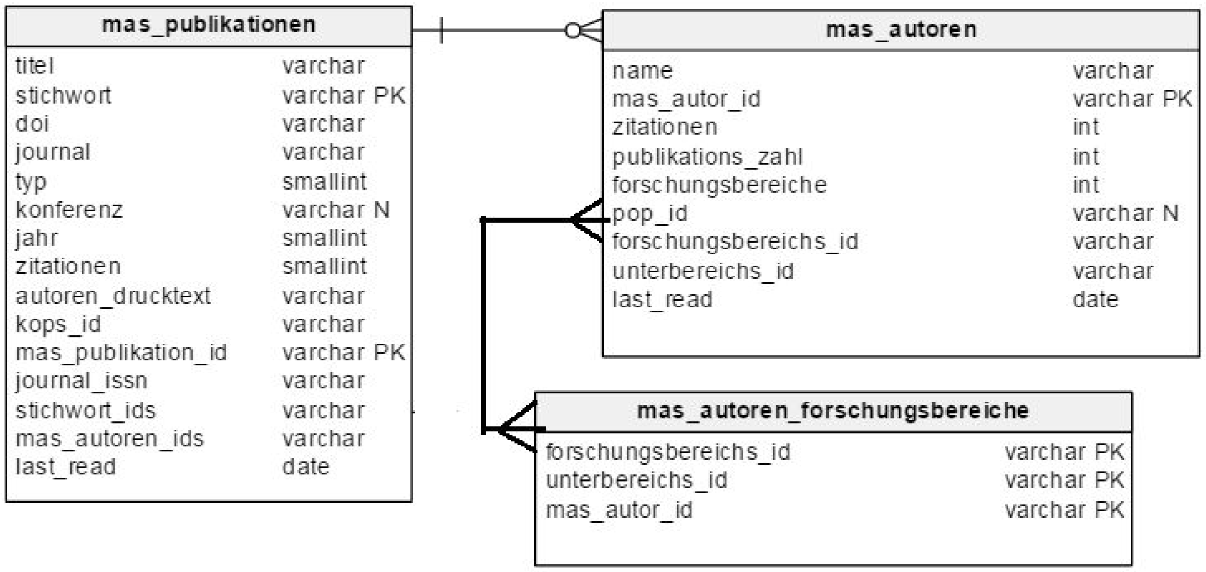
```
{"d":{"__type":"Response:http:\/\/research.microsoft.com","Author":null,"Conference":null,"Domain":null,"Journal":null,"Keyword":null,"Organization":null,"Publication":
{"__type":"PublicationResponse:http:\/\/research.microsoft.com","EndIdx":1,"StartIdx":1,"TotalItem":112688,"Result":[{"__type":"Publication:http:\/\/research.microsoft.com","Abstract":"Our
ability to generate and collect data has been increasing rapidly. Not only are all of our business, scientific, and government transactions now computerized, but the widespread use of
digital cameras, publication tools, and bar codes also generate data. On the collection side, scanned text and image platforms, satellite remote sensing systems, and the World Wide Web
have flooded us","Author":
[{"__type":"Author:http:\/\/research.microsoft.com","Affiliation":null,"CitationCount":0,"DisplayPhotoURL":null,"FirstName":"Jiawei","GIndex":0,"HIndex":0,"HomepageURL":null,"ID":594572,"L
astName":"Han","MiddleName":"","NativeName":null,"PublicationCount":0,"ResearchInterestDomain":null},
{"__type":"Author:http:\/\/research.microsoft.com","Affiliation":null,"CitationCount":0,"DisplayPhotoURL":null,"FirstName":"Micheline","GIndex":0,"HIndex":0,"HomepageURL":null,"ID":2331044
,"LastName":"Kamber","MiddleName":"","NativeName":null,"PublicationCount":0,"ResearchInterestDomain":null}],"CitationContext":
[],"CitationCount":5979,"Conference":null,"DOI":"","FullVersionURL":
["http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/9cmplx.pdf","http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/1intro.pdf","http:\/\/www.i
r.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/4lang.pdf","http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/6asso.pdf","https:\/\/dspace.ist.utl.pt\/bit
stream\/2295\/289040\/1\/lesson2.pdf","http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/5desc.pdf","http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/Bo
okNotes\/8clst.pdf","http:\/\/www.ir.iit.edu\/~dagr\/DataMiningCourse\/Spring2001\/BookNotes\/7class.pdf","http:\/\/www.ida.liu.se\/~732A02\/material\/fo-
intro.pdf"],"ID":694978,"Journal":null,"Keyword":[{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":9033,"Name":"Data Mining","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":9972,"Name":"Digital Camera","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":22078,"Name":"Large Data Sets","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":35009,"Name":"Relational Data","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":36239,"Name":"Satellite Remote Sensing","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":38375,"Name":"Social Network","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":40483,"Name":"Structured Data","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":41259,"Name":"Systems and Applications","PublicationCount":0},
{"__type":"Keyword:http:\/\/research.microsoft.com","CitationCount":0,"ID":73998,"Name":"World Wide Web","PublicationCount":0}],"ReferenceCount":160,"Title":"Data Mining: Concepts and
Techniques","Type":1,"Year":2000}]},"ResultCode":0,"Trend":null,"Version":"1.1"}}
```

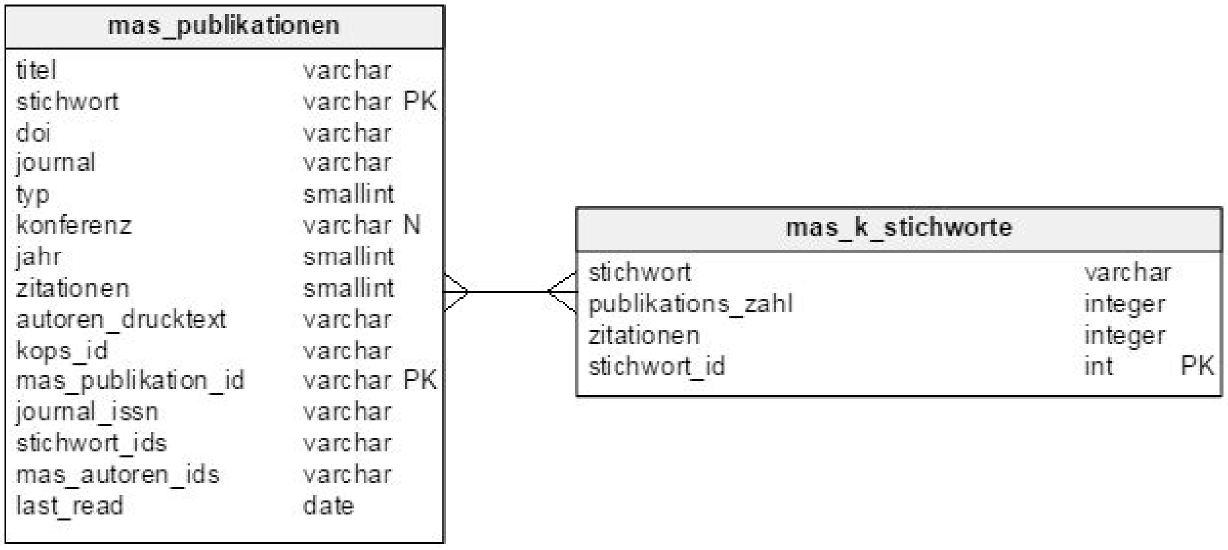
Talend provides a component called tExtractJSONFields. Information can be retrieved by knowing the locations of the data. The tool uses xml paths to retrieve the data and store into variables which can be then transferred to the database after processing if required.

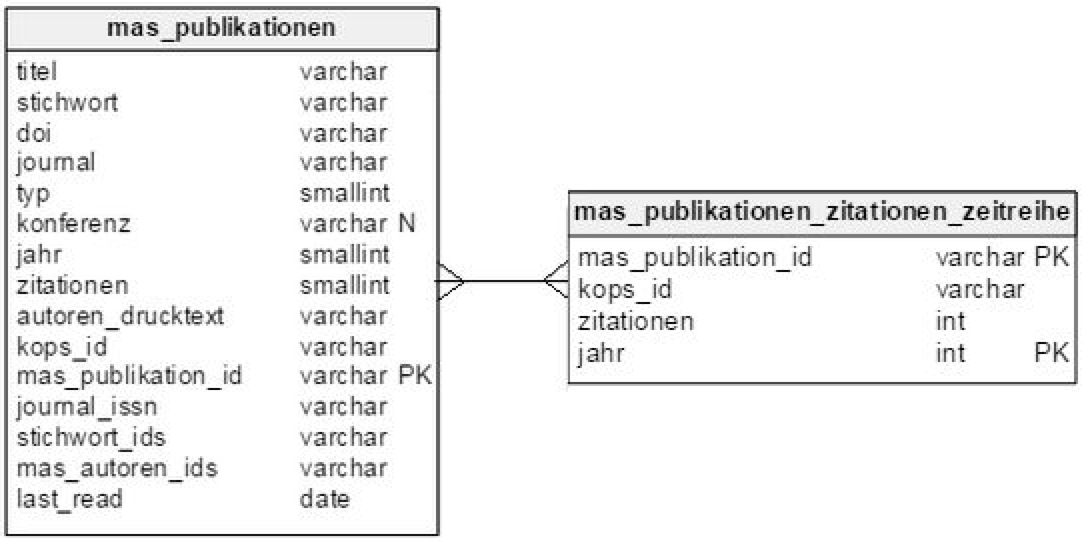
The json can contain a maximum of 100 entries for any query. Also the performance is smooth when there are less than 2 queries placed per second. For the latter, Thread.sleep() function is used with the value of 333 in tJavaRow components directly preceding tHttpRequest components.

Description of the target schema



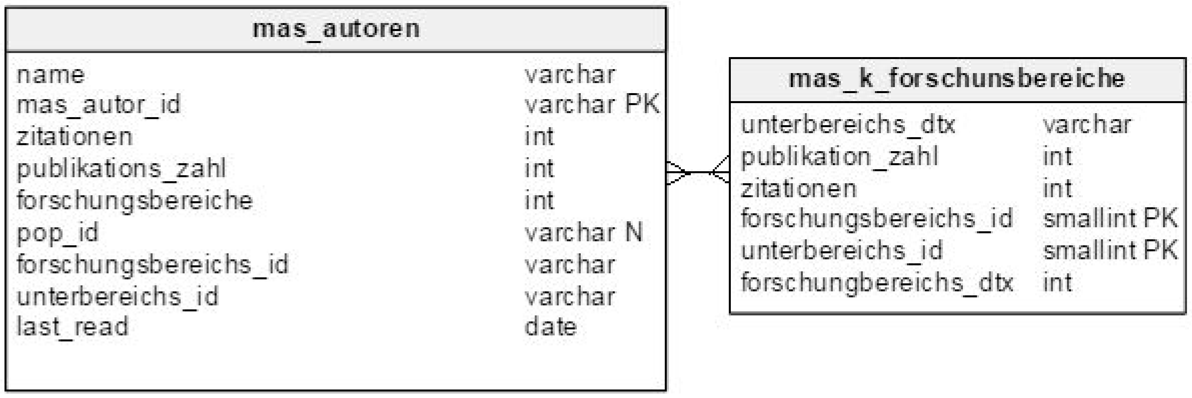

mas_publikationen
titel varchar
stichwort varchar PK
doi varchar
journal varchar
typ smallint
konferenz varchar N
jahr smallint
zitationen smallint
autoren_drucktext varchar
kops_id varchar
mas_publikation_id varchar PK
journal_issn varchar
stichwort_ids varchar
mas_autoren_ids varchar
last_read date
mas_autoren
name varchar
mas_autor_id varchar PK
zitationen int
publikations_zahl int
forschungsbereiche int
pop_id varchar N
forschungsbereichs_id varchar
unterbereichs_id varchar
last_read date
mas_autoren_forschungsbereiche
forschungsbereichs_id varchar PK
unterbereichs_id varchar PK
mas_autor_id varchar PK


mas_publikationen
titel varchar
stichwort varchar PK
doi varchar
journal varchar
typ smallint
konferenz varchar N
jahr smallint
zitationen smallint
autoren_drucktext varchar
kops_id varchar
mas_publikation_id varchar PK
journal_issn varchar
stichwort_ids varchar
mas_autoren_ids varchar
last_read date
mas_k_stichworte
stichwort varchar
publikations_zahl integer
zitationen integer
stichwort_id int PK



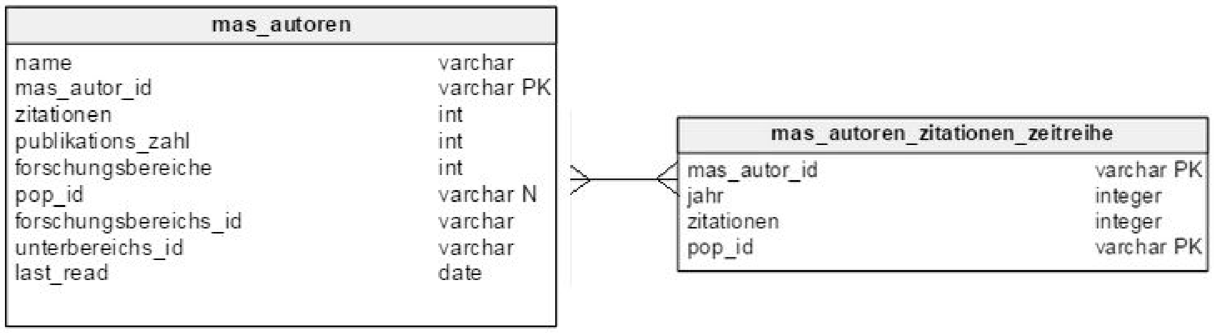
mas_publikationen		
titel	varchar	
stichwort	varchar	
doi	varchar	
journal	varchar	
typ	smallint	
konferenz	varchar	N
jahr	smallint	
zitationen	smallint	
autoren_drucktext	varchar	
kops_id	varchar	
mas_publikation_id	varchar	PK
journal_issn	varchar	
stichwort_ids	varchar	
mas_autoren_ids	varchar	
last_read	date	

mas_publikationen_zitationen_zeitreihe		
mas_publikation_id	varchar	PK
kops_id	varchar	
zitationen	int	
jahr	int	PK



mas_autoren		
name	varchar	
mas_autor_id	varchar	PK
zitationen	int	
publikations_zahl	int	
forschungsbereiche	int	
pop_id	varchar	N
forschungsbereichs_id	varchar	
unterbereichs_id	varchar	
last_read	date	

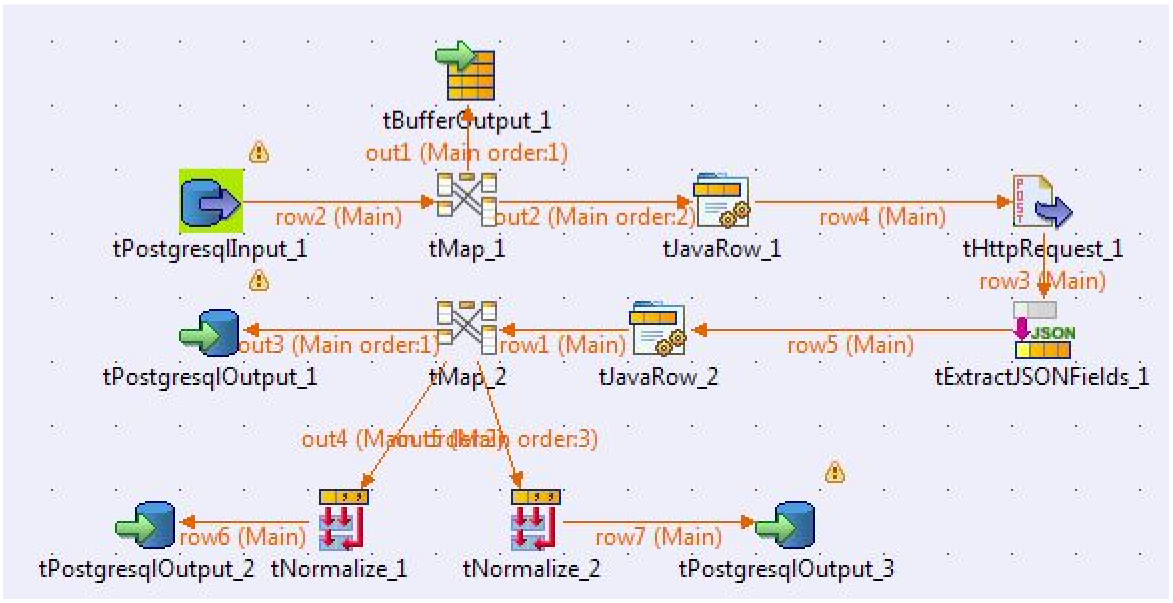
mas_k_forschunsbereiche		
unterbereichs_dtx	varchar	
publikation_zahl	int	
zitationen	int	
forschungsbereichs_id	smallint	PK
unterbereichs_id	smallint	PK
forschungbereichs_dtx	int	



mas_autoren		
name	varchar	
mas_autor_id	varchar	PK
zitationen	int	
publikations_zahl	int	
forschungsbereiche	int	
pop_id	varchar	N
forschungsbereichs_id	varchar	
unterbereichs_id	varchar	
last_read	date	

mas_autoren_zitationen_zeitreihe		
mas_autor_id	varchar	PK
jahr	integer	
zitationen	integer	
pop_id	varchar	PK

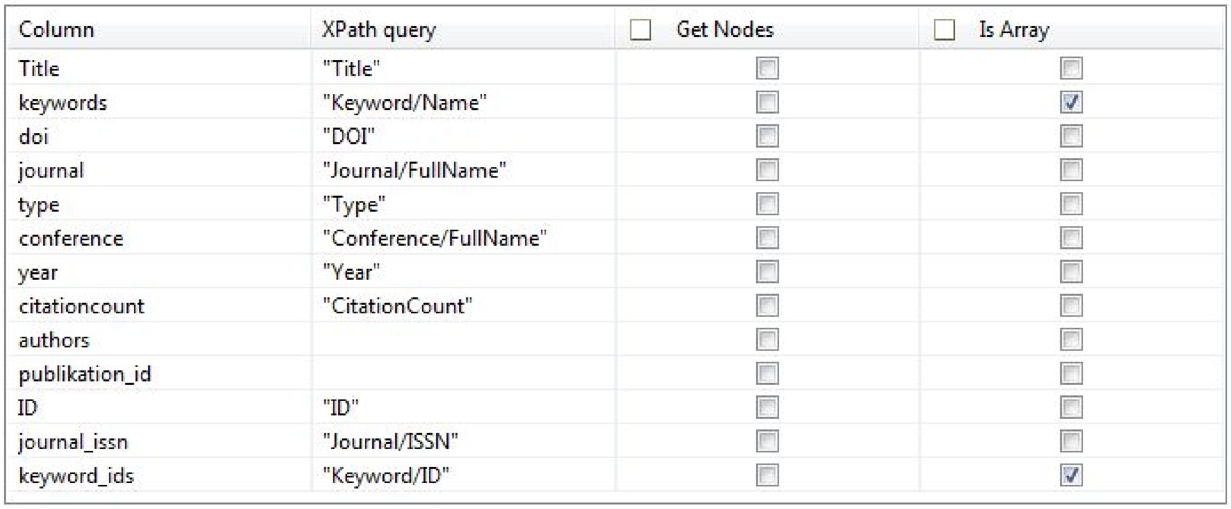
kops_maspi



This job extracts title text, author and year of publication from the kops_publikationen table and passes it as parameters to the url to retrieve publication information. An example URL is as follows:

"http://academic.research.microsoft.com/json.svc/search?AppId=****&FullTextQuery="+row4.title+"&YearStart="+row4.year+"&AuthorQuery="+row4.authors_mod+"&ResultObjects=Publication&StartIdx=1&EndIdx=1"

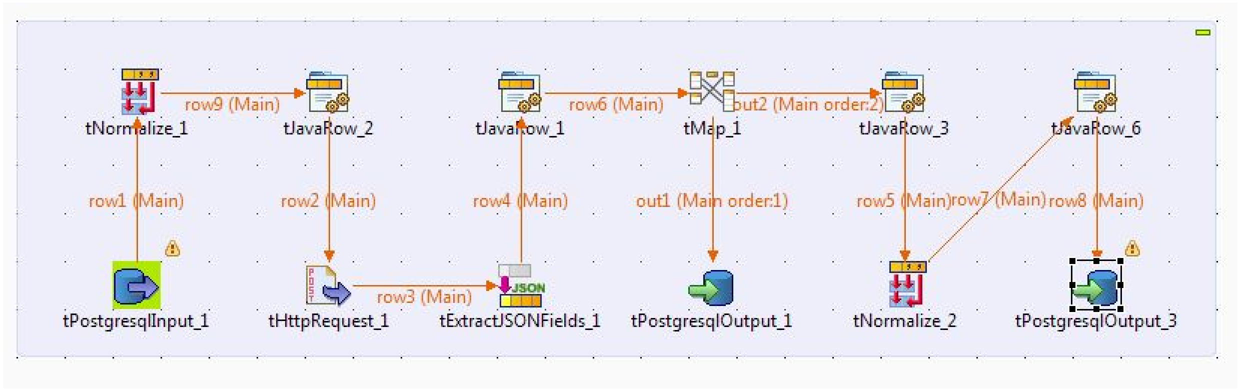
The retrieved json content is passed to tExtractJSONFields which then retrieves specific fields of information. All relevant publication information is to be stored along with keyword_ids. The mapping is follows:



Column	XPath query	☐ Get Nodes	☐ Is Array
Title	"Title"	☐	☐
keywords	"Keyword/Name"	☐	☑
doi	"DOI"	☐	☐
journal	"Journal/FullName"	☐	☐
type	"Type"	☐	☐
conference	"Conference/FullName"	☐	☐
year	"Year"	☐	☐
citationcount	"CitationCount"	☐	☐
authors		☐	☐
publikation_id		☐	☐
ID	"ID"	☐	☐
journal_issn	"Journal/ISSN"	☐	☐
keyword_ids	"Keyword/ID"	☐	☑

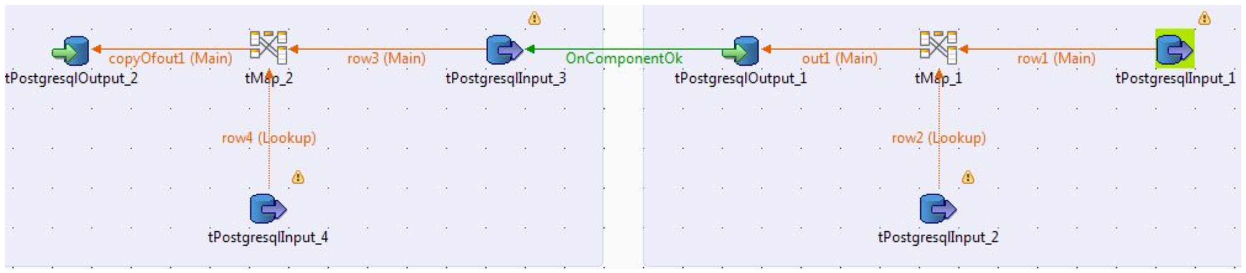
The components called tJavaRow does some processing to ensure data is removed of special characters and to restrict processing of more than 2 rows/second to ensure that the requests to the website are kept within the limit to prevent disruptions in access caused due to excess volume of requests. The table created is mas_publikationen. Additionally, two extra tables are created. These are mas_publikationen_stichworte and mas_publikationen_autoren. These tables are the fact tables related to keywords and authors respectively. The component tMap redirects values to relevant tables. This avoids repeated retrieval of the same values and the resulting wastage of time and loss of processing speed.

autoren



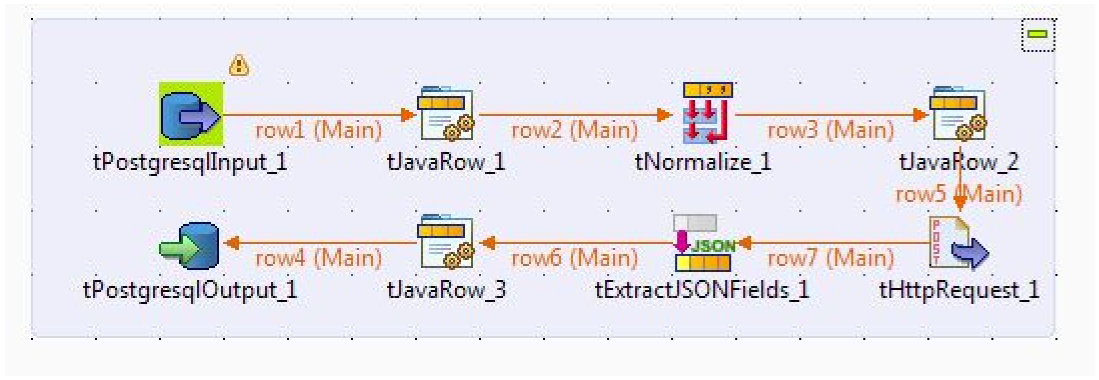
The next is retrieving the author information. Since, all author ids are already retrieved in the previous process, the task now is to normalize the author ids separated by a comma (', '). Then each author id is passed to the URL and the publication count, citation count and research domain information are retrieved and stored in relevant fields. The tables created are mas_autoren. Additionally, the table mas_autoren_forschungsbereiche is created. This is a fact table related to the research domains of the authors.

mas_autoren_2_pop



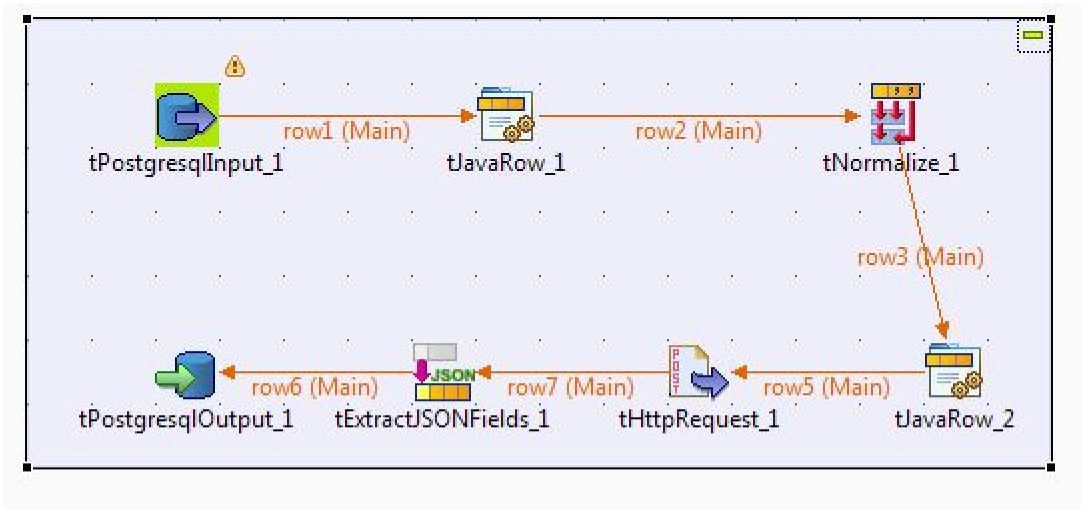
The previous job doesn't fill the pop_ids of the relevant fields. The mas_autoren_2_pop job matches name with relevant pop_id in the tables mas_autoren table and mas_publikationen_autoren from the kops_autor table.

domain_details



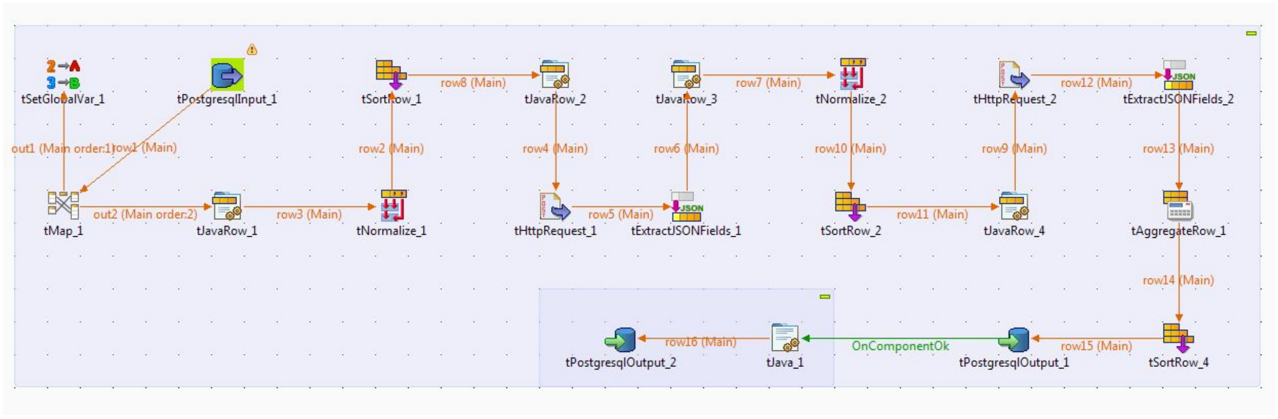
This job sends domain ids and sub domain ids after normalizing and filtering and gets details of publication and citation counts. The table created is mas_k_forschungsbereiche.

keyword_details



This job sends keyword ids after normalizing and filtering and gets details of publication and citation counts. The table created is mas_k_stichworte

year_wise_citation_author

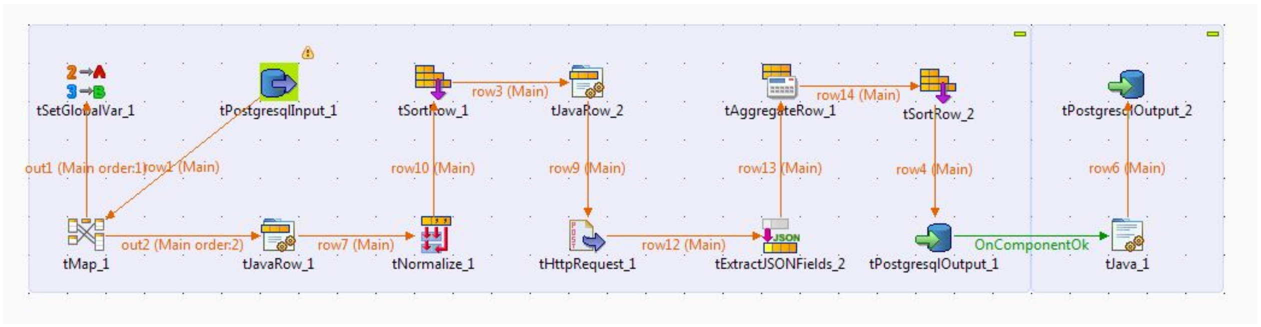


This job outputs a table with author id, year and citation of any of the authors' publications for that year. The id and year may appear multiple times. The inputs are the mas_autor_id and publication count from the newly formed mas_autor table. Then the publication count is divided into a series of hundreds. Since the API returns 100 queries at a time, it is important to store a series of values starting from 1 to the publication count and iterated for every 100. Therefore, for every 100 the query returns a list of publications. The table created is mas_autoren_zitationen_zeitreihe.

For every publication, one has to search all related publications where the latter cites the former. Again the list of latter such publications sometimes run into the hundreds and need to denormalized into a series of hundreds again.

In each list, there are publications which cite the former. And the year of each publication is considered. This shows that the former was cited in the year of publication of the latter. All years are counted by appearance. This is then aggregated over all the lists at the end.
This information is then stored in the table "mas_autoren_zitationen_zeitreihe".

year_wise_citation_publication



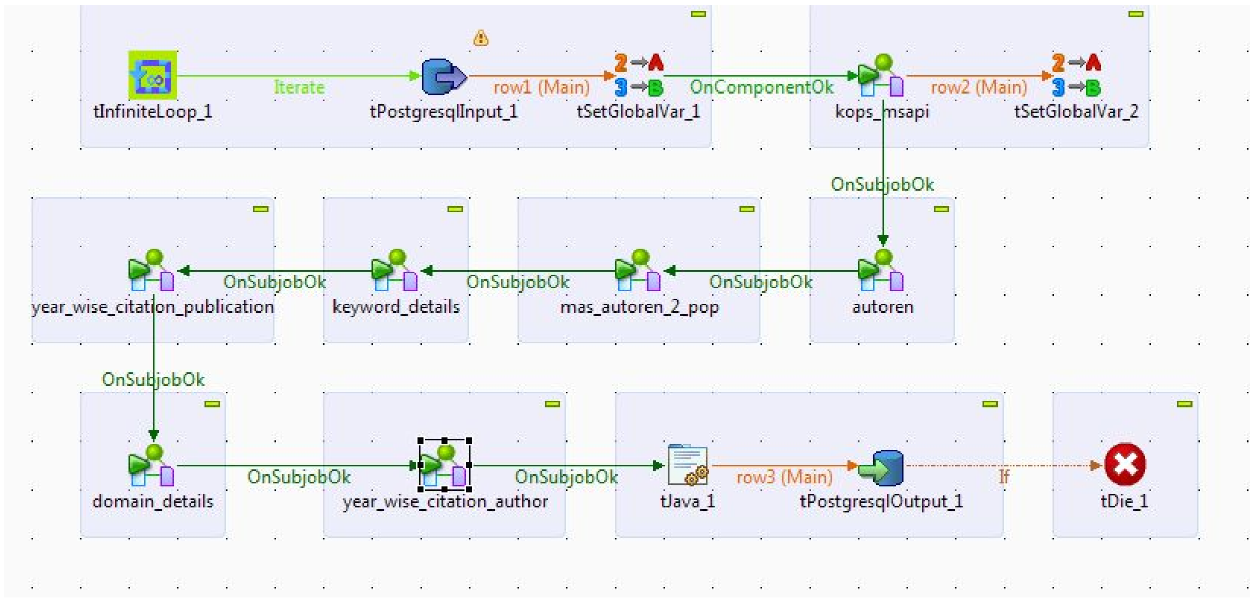
This job outputs a table with publication id, year and citation of any of the University's publications for that year. The id and year may appear multiple times. The inputs are the mas_publikation_id and publication count from the newly formed mas_publikationen table.

For every publication, one has to search all related publications where the latter cites the former. Again the list of latter such publications sometimes run into the hundreds and need to denormalized into a series of hundreds again.

In each list, there are publications which cite the former. And the year of each publication is considered. This shows that the former was cited in the year of publication of the latter. All years are counted by appearance. This is then aggregated over all the lists at the end.

This information is then stored in the table "mas_publikationen_zitationen_zeitreihe".

Combined jobs



The combined process is as above. Since the process takes too long to complete together and it is not feasible to complete each process individually, it was decided to enable the whole process to start and restart at any point of time. For this, one extra column is created in kops_publikationen called last_read. This column updates the column with the date of reading at the end. Similarly, mas_publikationen and mas_autoren also contain this column, which will be updated at the end of creation of citations tables respectively. One row is to be read for each iteration. After each iteration, the three tables need to be updated based on the date of reading. Finally, if the kops_publikationen table contains no more rows with last_read missing or having a date 6 months old, then the infinite loop is broken using tDie, displaying a message to the user, “All entries read. Come back after 6 months”. This completes the whole task.

Possible errors

The most likely errors are those which occur are timeout problems. This could be the result of breakage in the internet or due to server problems.

ETL Processes for and Web of Science data

Dhiraj Eadara 857136

Web of Science ETL

Which data is extracted?

Links Article Match Retrieval Service (LAMR) is the api of web of science. This service accepts xml files uploaded to it containing api for requests. An example of an xml request is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request xmlns="http://www.isinet.com/xrpc42"
src="app.id=PartnerApp,env.id=PartnerAppEnv,partner.email=EmailAddress"      >
  <fn name="LinksAMR.retrieve">
    <list>
<!-- WHO'S REQUESTING -->
      <map>
        <val name="username">username</val>
        <val name="password">test</val>
      </map>
<!-- WHAT'S REQUESTED -->
      <map>
        <list name="JCR">
          <val>impactGraphURL</val>
        </list>
      </map> <!--end "return_data" -->
<!-- LOOKUP DATA -->
      <map>
<!-- QUERY "cite_id" -->
        <map name="cite_id">
          <val name="title">full journal title</val>
          <val name="issn">1234-5678</val>
        </map> <!-- end of cite_id-->
<-- QUERY "cite_id2" -->
        <map name="cite_id2">
          ...
        </map>
-->
      </map> <!-- end of citations -->
    </list>
  </fn>
</request>
```

The above file can be uploaded to the URL https://ws.isiknowledge.com/cps/xrpc. The method is POST, i.e, the file needs to uploaded.

An example of response is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<response xmlns="http://www.isinet.com/xrpc42"
src="app.id=PartnerApp,env.id=PartnerAppEnv,partner.email=EmailAddress">
<fn name="LinksAMR.retrieve" rc="OK">
    <map>
<!-- RESPONSE for QUERY "cite_1" -->
      <map name="cite_1">
        <map name="WOS">
          <val name="timesCited">ts_val</val>
          <val name="ut">123456789</val>
          <val name="doi">10.224/xxxxx.xx.xx.xxx</val>
          <val name="sourceURL">URL_to_record</val>
          <val name="citingArticlesURL>URL_to_citing_articles</val>
          <val name="relatedRecordsURL">URL_to_related_records</val>
        </map>
      </map>
<!-- RESPONSE for QUERY "cite_2"
      <map name="cite_2">
        <map name="WOS">
          ...
        </map>
      </map>
-->
    </map>
  </fn>
</response>
```

Query limit will be 30,000 publications. There can be a maximum of 50 entries per file. For convenience, however, only one entry has been inserted for every file.

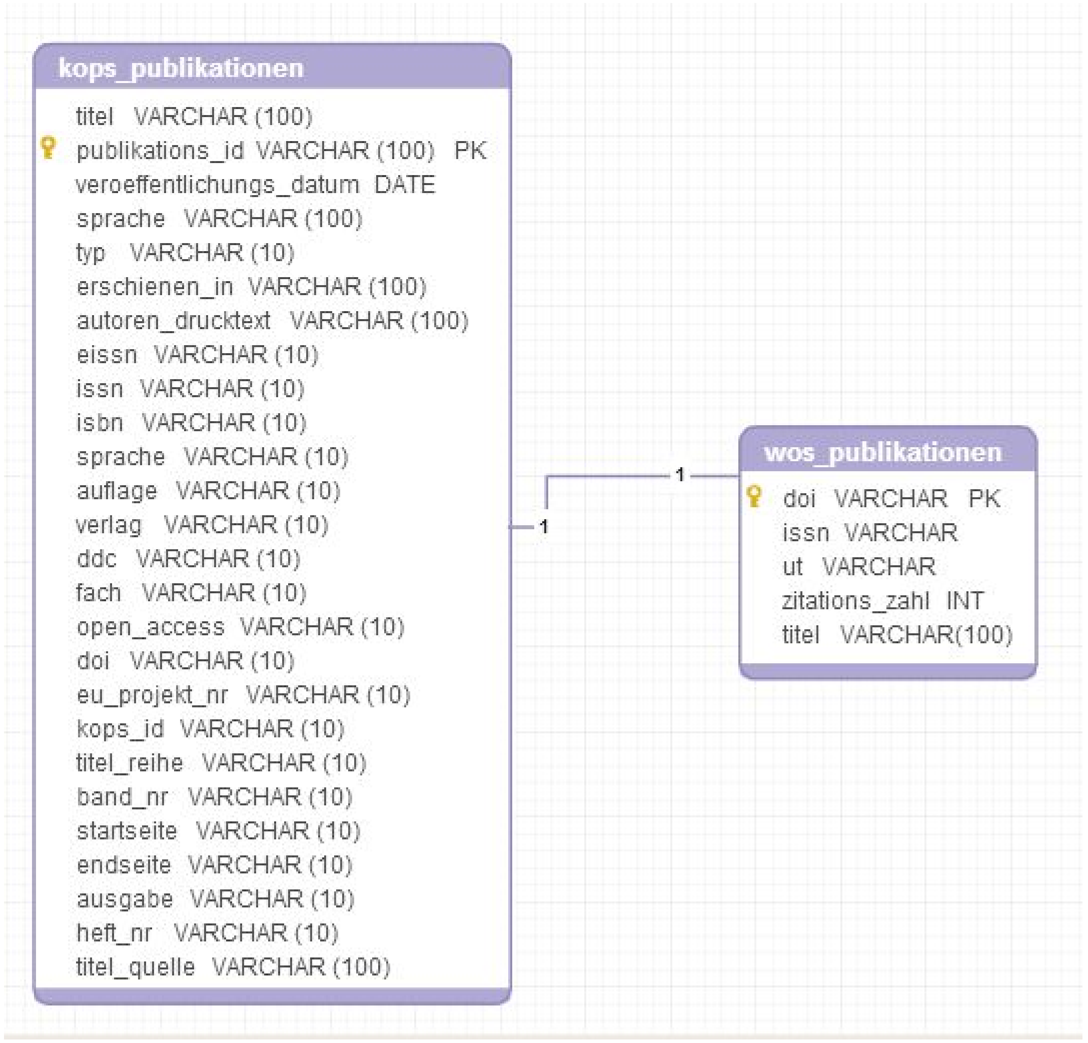
One may use doi as an identifier to specify an article. If an identifier is not available, you must supply all of the following bibliographic elements:
1. Journal title
2. Volume
3. Issue
4. Start page or article number

If the journal title is missing, the combination of author and ISSN in place of journal title should return results.

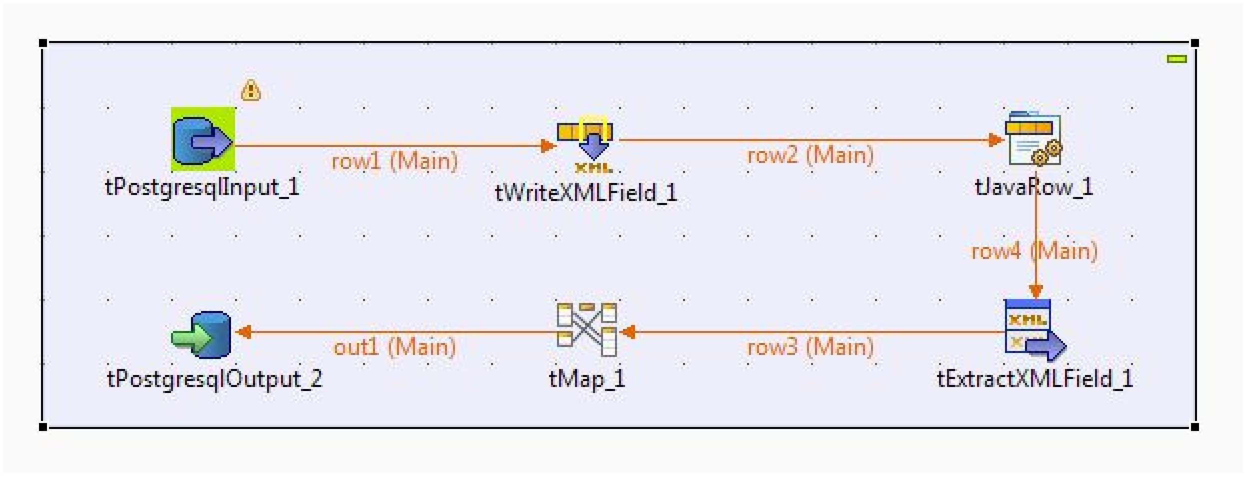
For example, each of the following combinations should uniquely identify an article:
Journal title + volume + issue + start page
Journal title + volume + issue + article number
Author + ISSN + volume + issue + start page
Author + ISSN + volume + issue + article number

In case of kops, article number is not specified.

data schema



wos_4_doi



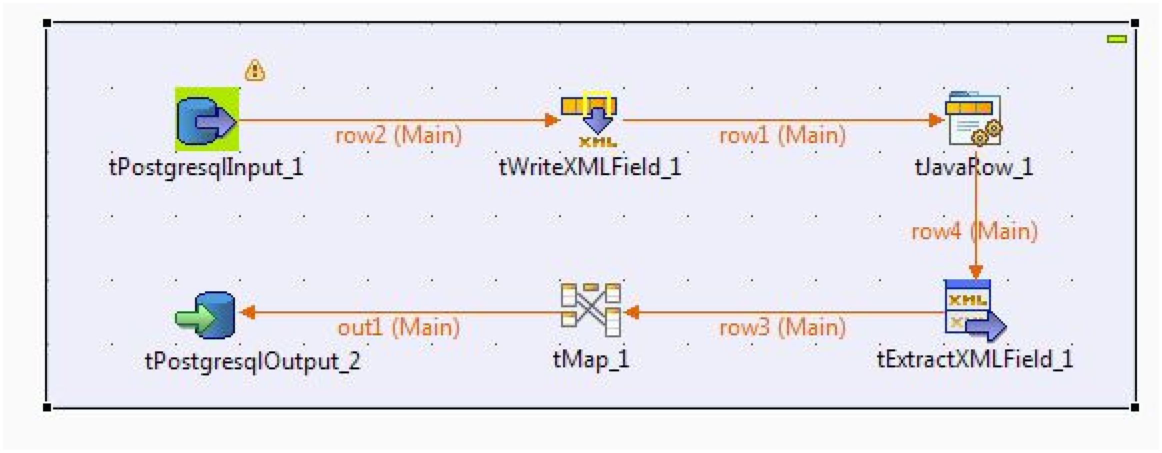
The entries with dois are taken from two tables. The sql is as follows:
"select kops_id,doi from kops_publikationen where doi is not null and doi<>''
UNION ALL
select kops_id,doi from mas_publikationen where doi is not null and doi<>''
"
The kops_id and doi are then extracted. The xml is to be contain these two elements.
The xml containing the api which is to be posted to the URL is then formed.

The xml strings are posted to the URL one after the other to the web of science URL. This outputs the response in xml which then is used to extract the following details:

- ut
- timescited
- doi
- title
- issn

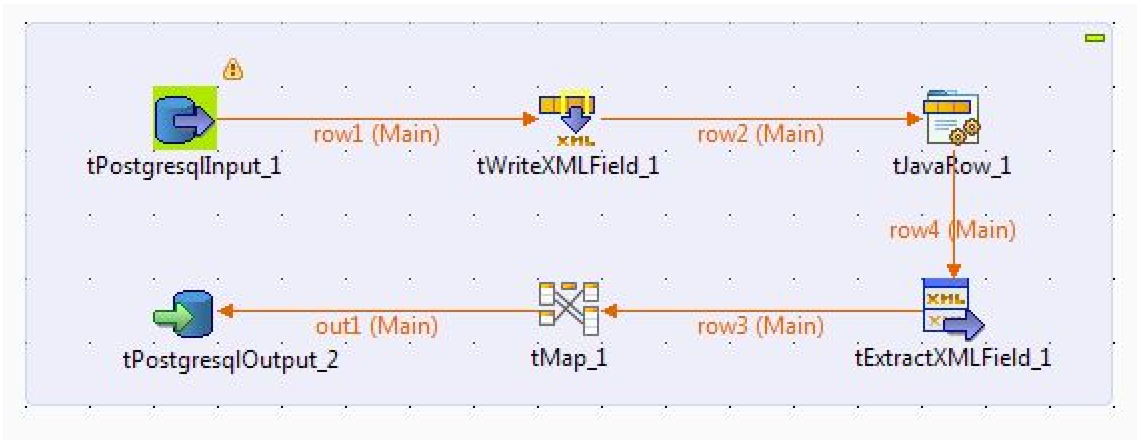
Once the job completed successfully, the files are deleted. This is to ensure that subsequent requests will not read the file which has been already read.

wos_xml_creator_for_journals



This job is to read those entries which don't have dois to identify the titles with. In such cases, whenever the Journal title, volume, issue and start page are filled, all these details can be inserted into the xml file.

wos_xml_creator_for_authors



This job is to read those entries which don't have dois to identify the titles with. In such cases, whenever the author, issn, volume, issue and start page entries are filled, all these details can be inserted into the xml file.

Possible errors

The most likely errors are those which occur are timeout problems. This could be the result of breakage in the internet or due to server problems.