The Patent Application Priority Table (TLS204_APPLN_PRIOR)

Welcome to the fourth table of the PATSTAT database, the application priority table. This table contains the Paris Convention priorities of an application.

We initialize the environment and import the table.

In [1]:
```
from epo.tipdata.patstat import PatstatClient

# Initialize the PATSTAT client
patstat = PatstatClient(env='PROD')

# Access ORM
db = patstat.orm()

# Importing the as models
from epo.tipdata.patstat.database.models import TLS204_APPLN_PRIO
R
```

APPLN_ID (Primary Key)

Again we have the application ID linking this table to table TLS201. In the following section, we will see how to properly join these two tables.

PRIOR_APPLN_ID

Surrogate key of an application of which the priority is claimed under the Paris convention. It is the `appln_id` of the priority application in PATSTAT.

Suppose that we want to retrive earlier applications from table TLS201 that are claimed by applications in table TLS204. Obviously, we need a `JOIN` between the two tables. However, notice that even if the common attribute is `appln_id`, table TLS204 refers to table TLS201 via the `prior_appln_id` attribute. Therefore, we join the two table imposing the match between `prior_appln_id` in table TLS201 and `appln_id` in table TLS204. To limit the number of rows of the resulting table, we filter by selecting applications whose earliest filing is on 2016-05-09.

In [2]:
```
# Import table TLS201_APPLN and date class from datetime
from epo.tipdata.patstat.database.models import TLS201_APPLN
from datetime import date

# Create the date object
d = date(2016,5,9)

prior_id = db.query(
    TLS201_APPLN.inpadoc_family_id,
    TLS204_APPLN_PRIOR.prior_appln_id,
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_filing_date,
    TLS201_APPLN.earliest_filing_date
).join(
    TLS201_APPLN, TLS204_APPLN_PRIOR.prior_appln_id == TLS201_APP
LN.appln_id  # Join table TLS201 and TLS204
).filter(
    TLS201_APPLN.appln_filing_date == d  # Filter out the applica
tions that were not filed in the specified date
)

prior_id_df = patstat.df(prior_id)
prior_id_df
```

Out[2]:

	inpadoc_family_id	prior_appln_id	appln_id	appln_filing_date	earliest_filing_date
0	486002014	905502174	905502174	2016-05-09	2016-05-09
1	478614773	905522917	905522917	2016-05-09	2016-05-09
2	478884329	486153321	486153321	2016-05-09	2016-05-09
3	478125932	903774318	903774318	2016-05-09	2016-05-09
4	486110473	486354541	486354541	2016-05-09	2016-05-09
...	...	...	...	...	...
4879	447022017	900160286	900160286	2016-05-09	2016-05-09
4880	473298426	905501493	905501493	2016-05-09	2016-05-09
4881	482995448	482995448	482995448	2016-05-09	2016-04-19
4882	478614218	905504228	905504228	2016-05-09	2016-05-09
4883	470294575	905480366	905480366	2016-05-09	2016-05-09

4884 rows × 5 columns

In this table we have all the applications whose priority is claimed by later applications. We can see that often `appln_filing_date` and `earliest_filing_date` coincides, meaning that many times the priority application corresponds to the earliest one to be filed.

One application can be claimed by more than one application. This means that if we filter one specific application from this table the result can consists of more rows referring to the same application (same `appln_id`) but different claims. This happens because in table TLS204 these are obviously different applications, so with different `appln_id`, but referring to the same priority application, so with the same `prior_appln_id`. Since when we join the two table we require these two attributes to match, the result contains "repetitions" of the same application. Let's give an example with `appln_id` equal to 452054224.

In [3]:
```
inpa = prior_id.subquery()

match = db.query(
    inpa.c.inpadoc_family_id,
    inpa.c.appln_id,
    inpa.c.appln_filing_date,
    inpa.c.earliest_filing_date
).filter(
    inpa.c.appln_id == 452054224
)

match_df = patstat.df(match)
match_df
```

Out[3]:

	inpadoc_family_id	appln_id	appln_filing_date	earliest_filing_date
0	452054224	452054224	2016-05-09	2016-05-09
1	452054224	452054224	2016-05-09	2016-05-09
2	452054224	452054224	2016-05-09	2016-05-09
3	452054224	452054224	2016-05-09	2016-05-09
4	452054224	452054224	2016-05-09	2016-05-09

We have a particular case. Indeed, the `appln_id` is equal to the `inpadoc_family_id` . This is because, for technical reasons, the `inpadoc_family_id` will be identical to the smallest `appln_id` of all members of that INPADOC family. Of course, when we search for applications with `prior_appln_id` equal to 452054224 in table TLS204 and show the corresponding `appln_id` we find different values of this last attribute.

In [4]:
```
prpr = db.query(
    TLS204_APPLN_PRIOR.prior_appln_id,
    TLS204_APPLN_PRIOR.appln_id
).filter(
    TLS204_APPLN_PRIOR.prior_appln_id == 452054224
)

prpr_df = patstat.df(prpr)
prpr_df
```

Out[4]:

	prior_appln_id	appln_id
0	452054224	505419139
1	452054224	514422128
2	452054224	477799206
3	452054224	513258422
4	452054224	477543588

Let's consider another INPADOC family, i.e. 482365125, to show that there are artifical priorities in PATSTAT. We can spot them because they have an ID higher than 900 millions.

In [5]:
```
prior_id = db.query(
    TLS204_APPLN_PRIOR.prior_appln_id,
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_filing_date,
    TLS201_APPLN.earliest_filing_date
).join(
    TLS201_APPLN, TLS204_APPLN_PRIOR.prior_appln_id == TLS201_APP
LN.appln_id
).filter(
    TLS201_APPLN.inpadoc_family_id == 482365125
)

prior_id_df = patstat.df(prior_id)
prior_id_df
```

Out[5]:

	prior_appln_id	appln_id	appln_filing_date	earliest_filing_date
0	486109434	486109434	2017-05-09	2016-05-09
1	905483142	905483142	2016-05-09	2016-05-09
2	905509469	905509469	2016-06-15	2016-06-15
3	905483142	905483142	2016-05-09	2016-05-09
4	905547584	905547584	2016-10-26	2016-10-26
...	...	...	...	...
2423	905784251	905784251	2019-05-06	2019-05-06
2424	905763490	905763490	2019-01-31	2019-01-31
2425	905743633	905743633	2018-11-08	2018-11-08
2426	905547584	905547584	2016-10-26	2016-10-26
2427	905483142	905483142	2016-05-09	2016-05-09

2428 rows × 4 columns

There are cases where an application is claimed as priority, but this application is not known to DOCDB. Then we nevertheless assume that this prior application does really exist, although for some reason it is not in DOCDB. Therefore, we will create an artificial prior application in PATSTAT. Typically, these artificial applications are applications which have been withdrawn or abandoned before publication, but which the applicant has used as a priority, or in America, for continuation. In particular, for artifical priorities the application ID ranges from 900 000 001 to 930 000 000.

By “priority” we here mean not only “Paris Convention priority”, but also other types of priorities which link one application to a “prior” application. By the way remember that this table refers only to Paris Convention priorities.

Link between INPADOC family and priority

Now we see how priorities and INPADOC families are actually linked. This time we join together the two tables TLS201 and TLS204 via the common attribute `appln_id`. The reason is that now we want to retrive the application IDs of all the applications that are claiming priority, i.e. the ones present in table TLS204, and find them in find them in table TLS201 to check other corresponding attributes. In particular, we are interested in `inpadoc_family_id` and `docdb_family_id`.

In [12]:
```
doc_inpa = db.query(
    TLS201_APPLN.appln_id,
    TLS204_APPLN_PRIOR.prior_appln_id,
    TLS201_APPLN.inpadoc_family_id,
    TLS201_APPLN.docdb_family_id
).join(
    TLS201_APPLN, TLS204_APPLN_PRIOR.appln_id == TLS201_APPLN.app
ln_id
).filter(
    TLS201_APPLN.inpadoc_family_id == 98645
).order_by(
    TLS204_APPLN_PRIOR.prior_appln_id
)

doc_inpa_df = patstat.df(doc_inpa)
doc_inpa_df
```

Out[12]:

	appln_id	prior_appln_id	inpadoc_family_id	docdb_family_id
0	55040615	21519332	98645	37908869
1	410457093	21519332	98645	37908869
2	267630635	21519332	98645	37908869
3	470852042	21519332	98645	37908869
4	317665131	21519332	98645	37908869
...	...	...	...	...
573	569638187	905456186	98645	80821536
574	477638220	905456186	98645	58523204
575	603334747	905456186	98645	80821536
576	476367222	905456186	98645	58237362
577	488675750	905456186	98645	60934953

578 rows × 4 columns

As we can see, all the applications referring to the same priority belongs to the same INPADOC family (same `inpadoc_family_id`) but they can belong to different DOCDB families (different `docdb_family_id`).

PRIOR_APPLN_SEQ_NR

Number indicating the place in the list of priorities claimed in the application.

If an application is claiming itself as a priority, then this priority is not stored in PATSTAT. So if a priority-claim element is the same as the application-reference, the application is claiming itself as a priority. These are normally the last priority in the priority-claims list of DOCDB. This means that the sequence numbers of any subsequent priorities claimed by this application must be reduced by 1.

In [8]:
```
prior_seq = db.query(
    TLS204_APPLN_PRIOR.prior_appln_id,
    TLS204_APPLN_PRIOR.appln_id,
    TLS204_APPLN_PRIOR.prior_appln_seq_nr
).order_by(
    TLS204_APPLN_PRIOR.prior_appln_seq_nr
).limit(50000)

prior_seq_df = patstat.df(prior_seq)
prior_seq_df
```

Out[8]:

	prior_appln_id	appln_id	prior_appln_seq_nr
0	904909545	50599725	1
1	423469203	443094909	1
2	13389463	2054278	1
3	23199852	46726044	1
4	11630075	18196677	1
...	...	...	...
49995	905614372	606392878	1
49996	906671189	273223781	1
49997	513245824	25673931	1
49998	469054827	470871968	1
49999	405669699	477151778	1

50000 rows × 3 columns

In [ ]: