The REG114_DATES Table

Welcome to table **REG114_DATES** of PATSTAT Register. This table contains some dates and, depending on the type of date, additional information:

- date when the request for examination was validly made
- date of dispatch of the first communication from the examining division
- date of publication of the B2 document (patent maintained as amended)
- date as of when the procedure was interrupted, with the cause of the interruption
- date of resumption of the procedure resumed
- date on which the procedure was suspended, with the reason for the suspension
- date on which the application was converted into a national application, with the country/countries to which it was converted
- date when the applicant withdrew his application
- legal effect date when the application was deemed to be withdrawn
- data concerning the handling of the request for revocation of the patent filed by the proprietor
- date when the time limit for filing opposition expired

In [1]:
```
from epo.tipdata.patstat import PatstatClient
from epo.tipdata.patstat.database.models import REG114_DATES
from sqlalchemy import func
import pandas as pd

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

# Access ORM
db = patstat.orm()
```

ID (Primary Key)

Technical identifier for an application, without business meaning. Its values will not change from one PATSTAT edition to the next.

In [5]:
```
i = db.query(
    REG114_DATES.id
).limit(1000)

df = patstat.df(i)
df
```

Out[5]:

	id
0	11712554
1	6814915
2	5701729
3	16723281
4	18924039
...	...
995	12762640
996	11700930
997	81300375
998	16752355
999	17789518

1000 rows × 1 columns

BULLETIN_YEAR

For actions that have been published in the EPO Bulletin, it is the year of the publication in the bulletin. The default value is 0, used for applications that are not published or for which the year is not known. The format is YYYY otherwise.

In [3]:
```
years = db.query(
    REG114_DATES.bulletin_year,
    REG114_DATES.id
).limit(1000)

years_df = patstat.df(years)
years_df
```

Out[3]:

	bulletin_year	id
0	2013	11712554
1	0	6814915
2	2012	5701729
3	0	16723281
4	2021	18924039
...	...	...
995	2017	12762640
996	2012	11700930
997	0	81300375
998	2017	16752355
999	2019	17789518

1000 rows × 2 columns

BULLETIN_NR

This is the issue number of the EPO Bulletin for actions that have been published in it. The Bulletin number indicates the calendar week the Bulletin has been published. The default value 0 is used when the attribute `bulletin_year` is 0.

In [4]:
```
bulletin_nr = db.query(
    REG114_DATES.id,
    REG114_DATES.bulletin_nr,
    REG114_DATES.bulletin_year
).limit(100)

bulletin_nr_df = patstat.df(bulletin_nr)
bulletin_nr_df
```

Out[4]:

	id	bulletin_nr	bulletin_year
0	17784711	21	2022
1	6846395	2	2009
2	20887221	38	2022
3	16907343	19	2019
4	15822765	6	2018
...	...	...	...
95	6768143	35	2008
96	16166707	18	2021
97	19786804	0	0
98	13824578	45	2015
99	14764556	0	0

100 rows × 3 columns

DATE_TYPE

This attribute indicates which type of date is stored in the corresponding row. The type can be one of the options presented at the beginning of this notebook. For major detail about the codes, please refer to the Data Catalog documentation.

Let's how many applications we find for each data type.

In [2]:
```
types_freq = db.query(
    REG114_DATES.date_type,
    func.count(REG114_DATES.id).label('Total applications')
).group_by(
    REG114_DATES.date_type
).order_by(
    func.count(REG114_DATES.id).desc()
)

types_freq_df = patstat.df(types_freq)
types_freq_df
```

Out[2]:

	date_type	Total applications
0	RQEXAM	4127147
1	FEXRRQ	3400980
2	DWDRWN	3303200
3	OPPNFI	2137004
4	WDRWNA	1793065
5	PMNTAM	30722
6	PRCITR	8432
7	PRCRES	7813
8	RQCONV	654
9	RQREVO	148

We can also take a look at all the date types linked to one specific application, let's say ID number 17784711.

In [4]:
```
types = db.query(
    REG114_DATES.date_type,
    REG114_DATES.id
).filter(
    REG114_DATES.id == 17784711
)

types_df = patstat.df(types)
types_df
```

Out[4]:

	date_type	id
0	RQEXAM	17784711
1	OPPNFI	17784711
2	FEXRRQ	17784711

EVENT_DATE

This attribute contains the date corresponding to the `date_type` attribute. The meaning of this attribute depends on the `date_type` attribute in table REG114_DATES.

For date with `date_type` equal to 'WDRWNA' and 'DWDRWN', in case of a deletion of the previous withdrawal date, a new date ‘9999-12-31’ will be issued.

As for `date_type` , we can retrieve all the date types and the corresponding event dates relative to one specific application. Let's use the same ID that we used in the previous section.

In [5]:
```
dates = db.query(
    REG114_DATES.event_date,
    REG114_DATES.date_type,
    REG114_DATES.id
).filter(
    REG114_DATES.id == 17784711
)

dates_df = patstat.df(dates)
dates_df
```

Out[5]:

	event_date	date_type	id
0	2019-03-11	RQEXAM	17784711
1	2022-03-17	OPPNFI	17784711
2	2020-05-04	FEXRRQ	17784711

This means that on 2019-03-11 the request for examination for application 17784711 was validly made. Then, on 2020-05-04 the request for the first examination report was requested. Lastly, on 2022-03-17 the time available to file an opposition to the application expired.

CAUSE_INTERRUPTION

Reason for interruption of proceedings. This attribute is only set to a non-default value (NA) if attribute `date_type` is equal to 'PRCITR', meaning “Proceedings interrupted”. The reason is one of the following:

- NA, not applicable or unknown
- DA, death of applicant
- DR, death of representative
- IA, legal incapacity of applicant
- IR, legal incapacity of representative
- PA, legal prevention of applicant
- PR, legal prevention of representative

In the following, we verify that there are no application with `cause_interruption` equal to a non-default value *and* for which the `date_type` is not 'PRCITR'. Thi swill result in an empty dataframe, so we would get an error when trying to access its value. To fix this, we use the `try-except` clause: the algorithm tries to retrieve the value and if this would generate an error then the command in the except will be 'executed' intead of launching the error message.

In [13]:
```
nd = db.query(
    REG114_DATES.date_type,
    REG114_DATES.cause_interruption,
    func.count(REG114_DATES.id).label('Total')
).filter(
    REG114_DATES.date_type != 'PRCITR',
    REG114_DATES.cause_interruption != 'NA'
).group_by(
    REG114_DATES.date_type,
    REG114_DATES.cause_interruption
)

nd_df = patstat.df(nd)

# try-except clause
try:
    nd_df['Total'].item()
except:
    print(f"No instances matching the filter found.")
```

```
No instances matching the filter found.
```

Let's take one application for which the proceeding was interrupted and retrieve the cause of it.

In [15]:
```
interrupted = db.query(
    REG114_DATES.cause_interruption
).filter(
    REG114_DATES.id == 16804857
)

interrupted_df = patstat.df(interrupted)
interrupted_df
```

Out[15]:

	cause_interruption
0	PA
1	NA
2	NA
3	NA

We can see that there three interruptions marked as NA, meaning that the reason is not known. It is worth to stress that the statement "the attribute `cause_interruption` is only set to a non-default value if attribute `date_type` is equal to 'PRCITR'" means that the non-default value is *possibly* ecountered if the date type is 'PRCITR'. More specifically, this is a necessary condition yet not sufficient.

CONVERTED_TO_COUNTRY

The office (country) to which the application was converted into a national application. This attribute is only set to a non-default (the default value is an empty string) value if attribute `date_type` is equal to 'RQCONV', meaning “Request for conversion to national application”. This sentence is to be intended the same way of the first sentence of the previous section.

Let's take one application for which the conversion to national application was requested and retrieve the country to which the application was converted.

In [22]:
```
country_conversion = db.query(
    REG114_DATES.id,
    REG114_DATES.converted_to_country
).filter(
    REG114_DATES.id == 79301195
)

country_conversion_df = patstat.df(country_conversion)
country_conversion_df
```

Out[22]:

	id	converted_to_country
0	79301195	GB
1	79301195	

In [ ]:
```

```