The REG102_PAT_PUBLN Table

Welcome to the second table of PATSTAT Register, namely table **REG102_PAT_PUBLN**. This table contains the EP and WO publications of the EP applications and international applications of table REG101_APPLN.

Here we can easily keep track of data concerning patent publication, such as publication date, publication kind, and publication in the EPO bulletin.

In [2]:
```
from epo.tipdata.patstat import PatstatClient
from epo.tipdata.patstat.database.models import REG102_PAT_PUBLN
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 [3]:
```
i = db.query(
    REG102_PAT_PUBLN.id
).limit(1000)

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

Out[3]:

	id
0	8867896
1	15715977
2	9798429
3	10176809
4	17751121
...	...
995	15150483
996	18805095
997	5748092
998	11741559
999	16762908

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 [5]:
```
years = db.query(
    REG102_PAT_PUBLN.bulletin_year,
    REG102_PAT_PUBLN.id
).limit(1000)

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

Out[5]:

	bulletin_year	id
0	2012	1118800
1	2008	7120366
2	2021	19854903
3	2020	20767366
4	1994	93112422
...	...	...
995	2008	7106325
996	1997	97919707
997	2023	23157527
998	2015	14814833
999	2005	4805470

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 [10]:
```
bulletin_nr = db.query(
    REG102_PAT_PUBLN.id,
    REG102_PAT_PUBLN.bulletin_nr,
    REG102_PAT_PUBLN.bulletin_year
).limit(100)

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

Out[10]:

	id	bulletin_nr	bulletin_year
0	1118800	25	2012
1	7120366	7	2008
2	19854903	27	2021
3	20767366	37	2020
4	93112422	6	1994
...	...	...	...
95	12753716	28	2014
96	15801722	26	2016
97	13849570	18	2014
98	20196282	4	2021
99	13785429	18	2015

100 rows × 3 columns

We can see that there are more than 6 thousand applications with `bulletin_nr` equal to 0, i.e. applications with `bulletin_year` equal to 0.

In [11]:
```
default_value = db.query(
    REG102_PAT_PUBLN.id,
    REG102_PAT_PUBLN.bulletin_nr,
    REG102_PAT_PUBLN.bulletin_year
).filter(
    REG102_PAT_PUBLN.bulletin_year == 0
)

default_value_df = patstat.df(default_value)
default_value_df
```

Out[11]:

	id	bulletin_nr	bulletin_year
0	79100875	0	0
1	78300670	0	0
2	78200107	0	0
3	79300266	0	0
4	78300417	0	0
...	...	...	...
6396	78200061	0	0
6397	79100028	0	0
6398	79100251	0	0
6399	78100542	0	0
6400	78300200	0	0

6401 rows × 3 columns

PUBLN_AUTH

Publication authority, which is either EPO or WIPO.

In [12]:
```
auth = db.query(
    REG102_PAT_PUBLN.publn_auth,
    func.count(REG102_PAT_PUBLN.id).label('number_of_application
s')
).group_by(
    REG102_PAT_PUBLN.publn_auth
).order_by(
    func.count(REG102_PAT_PUBLN.id).label('number_of_application
s').desc()
)

auth_df = patstat.df(auth)
auth_df
```

Out[12]:

	publn_auth	number_of_applications
0	EP	7494479
1	WO	4870724

PUBLN_NR

Document number for publication. This number consists of up to 10 digits. Leading zeros are significant, so this attribute must be used as a text string not as a numerical value. EP publications number always consists of 7 digits.

In [15]:
```
publn_nr = db.query(
    REG102_PAT_PUBLN.publn_nr,
    REG102_PAT_PUBLN.publn_auth
).limit(1000)

publn_nr_df = patstat.df(publn_nr)
publn_nr_df
```

Out[15]:

	publn_nr	publn_auth
0	0125588	EP
1	2310627	EP
2	0501610	EP
3	3267684	EP
4	1516492	EP
...	...	...
995	0520957	EP
996	1008418	EP
997	2018132247	WO
998	2013070611	WO
999	3950327	EP

1000 rows × 2 columns

PUBLN_KIND

Publication kind code: up to 2 characters consisting of A or B followed by a digit.

Let's count the number of occurrencies of each combination `publn_kind` - `publn_auth` .

In [16]:
```
kind = db.query(
    REG102_PAT_PUBLN.publn_kind,
    REG102_PAT_PUBLN.publn_auth,
    func.count(REG102_PAT_PUBLN.id).label('number_of_application
s')
).group_by(
    REG102_PAT_PUBLN.publn_kind,
    REG102_PAT_PUBLN.publn_auth
).order_by(
    func.count(REG102_PAT_PUBLN.id)
)

kind_df = patstat.df(kind)
kind_df
```

Out[16]:

	publn_kind	publn_auth	number_of_applications
0	B3	EP	578
1	A9	EP	2778
2	A8	EP	4511
3	B9	EP	7221
4	B8	EP	20285
5	B2	EP	30782
6	A2	WO	645569
7	A3	EP	674805
8	A2	EP	1043367
9	B1	EP	2344884
10	A1	EP	3365268
11	A1	WO	4225155

PUBLN_DATE

Date of publication. Also for this attribute there may be missing dates.

In [17]:
```
publn_date = db.query(
    REG102_PAT_PUBLN.publn_nr,
    REG102_PAT_PUBLN.publn_date
).limit(1000)

publn_date_df = patstat.df(publn_date)
publn_date_df
```

Out[17]:

	publn_nr	publn_date
0	1193552	2012-06-20
1	1887449	2008-02-13
2	3843576	2021-07-07
3	2020179757	2020-09-10
4	0582273	1994-02-09
...	...	...
995	1914813	2008-04-23
996	9739898	1997-10-30
997	4218567	2023-10-04
998	2015101477	2015-07-09
999	2005054049	2005-06-16

1000 rows × 2 columns

It is possible to retrieve the total number of publications occured in a specific window of time, for example after 2022.

In [3]:
```
recent_publn = db.query(
    REG102_PAT_PUBLN.publn_nr,
    REG102_PAT_PUBLN.publn_date
).filter(
    REG102_PAT_PUBLN.publn_date > '2022-12-31'
)

recent_publn_df = patstat.df(recent_publn)
recent_publn_df
```

Out[3]:

	publn_nr	**publn_date**
0	2024008244	2024-01-11
1	2024150356	2024-07-18
2	2023238996	2023-12-14
3	2024016631	2024-01-25
4	2023236208	2023-12-14
...	...	...
920679	3880168	2024-08-21
920680	4415967	2024-08-21
920681	4416387	2024-08-21
920682	4417006	2024-08-21
920683	4415725	2024-08-21

920684 rows × 2 columns

PUBLN_LG

Language of publication. The domain consists of up to 2 characters, according to ISO 639-1 language codes.

We can rank the most frequent languages.

In [18]:
```
lang = db.query(
    REG102_PAT_PUBLN.publn_lg,
    func.count(REG102_PAT_PUBLN.id).label('number_of_application
s')
).group_by(
    REG102_PAT_PUBLN.publn_lg
).order_by(
    func.count(REG102_PAT_PUBLN.id).desc()
)

lang_df = patstat.df(lang)
lang_df
```

Out[18]:

	publn_lg	number_of_applications
0	en	8035495
1	de	1825194
2	ja	739744
3	fr	580756
4	zh	535478
5		394839
6	ko	189792
7	es	26233
8	ru	20297
9	pt	6872
10	sv	3954
11	fi	2048
12	nl	1727
13	no	1225
14	da	971
15	it	291
16	ar	245
17	hu	11
18	cs	10
19	hr	7
20	sl	6
21	sk	3
22	tr	2
23	sh	2
24	ee	1

We can combine the language search with the date. Let's say that we are interested in the number of publications in English in 2023.

In [4]:
```
year_lang = db.query(
    REG102_PAT_PUBLN.publn_nr,
    REG102_PAT_PUBLN.publn_date
).filter(
    REG102_PAT_PUBLN.publn_date > '2022-12-31',
    REG102_PAT_PUBLN.publn_date < '2024-01-01',
    REG102_PAT_PUBLN.publn_lg == 'en'
)

year_lang_df = patstat.df(year_lang)
year_lang_df
```

Out[4]:
```
          publn_nr  publn_date
      0  2023148585  2023-08-10
      1  2023136821  2023-07-20
      2  2023141532  2023-07-27
      3  2023133607  2023-07-20
      4  2023168184  2023-09-07
    ...         ...         ...
 367070  2023245550  2023-12-28
 367071  2023250053  2023-12-28
 367072  2023247794  2023-12-28
 367073  2023248251  2023-12-28
 367074  2023247364  2023-12-28

367075 rows × 2 columns
```

In [ ]: