The REG109_DESIGNATED_STATES Table

Welcome to table **REG109_DESIGNATED_STATES** in PATSTAT Register. This table contains the information about states designated for an application. There are these types of designated states:

- Member states of the European Patent Convention (EPC)
- Extension states
- Validation states

In [1]:
```
from epo.tipdata.patstat import PatstatClient
from epo.tipdata.patstat.database.models import REG109_DESIGNATED
_STATES
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(
    REG109_DESIGNATED_STATES.id
).limit(1000)

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

Out[3]:

	id
0	2254003
1	19921300
2	18801043
3	19884183
4	6838856
...	...
995	19709812
996	13713989
997	19886929
998	98117794
999	2701955

1000 rows × 1 columns

STATE_TYPE

This attribute indicates whether the designated state is a member state, an extension state or a validation state.

Let's count how many applications we have for each type.

In [8]:
```
kind = db.query(
    REG109_DESIGNATED_STATES.state_type,
    func.count(REG109_DESIGNATED_STATES.id).label('number_of_appl
ications')
).group_by(
    REG109_DESIGNATED_STATES.state_type
).order_by(
    func.count(REG109_DESIGNATED_STATES.id)
)

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

Out[8]:

	state_type	number_of_applications
0	VAL	1823974
1	EXT	3240555
2	MEM	7959462

CHANGE_DATE

It is the date of when the record was saved in the database.

In [4]:
```
change_date = db.query(
    REG109_DESIGNATED_STATES.change_date,
    REG109_DESIGNATED_STATES.id
).limit(100)

change_date_df = patstat.df(change_date)
change_date_df
```

Out[4]:

	change_date	id
0	2004-09-29	4748576
1	2015-06-09	14197917
2	9999-12-31	19863610
3	9999-12-31	1955921
4	2011-07-15	9768344
...	...	...
95	2023-12-08	23733486
96	1998-01-09	97305551
97	2007-01-20	6823362
98	9999-12-31	19770280
99	2023-09-01	22929152

100 rows × 2 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(
    REG109_DESIGNATED_STATES.bulletin_year,
    REG109_DESIGNATED_STATES.id
).limit(1000)

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

Out[5]:

	bulletin_year	id
0	1991	91101631
1	1987	87101577
2	2013	12199742
3	2004	3078560
4	2003	2015790
...	...	...
995	2019	19169097
996	1992	91306378
997	0	21868506
998	1997	94904182
999	2020	18203634

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 [6]:
```
bulletin_nr = db.query(
    REG109_DESIGNATED_STATES.id,
    REG109_DESIGNATED_STATES.bulletin_nr,
    REG109_DESIGNATED_STATES.bulletin_year
).limit(100)

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

Out[6]:

	id	bulletin_nr	bulletin_year
0	91101631	33	1991
1	87101577	35	1987
2	12199742	40	2013
3	3078560	27	2004
4	2015790	4	2003
...	...	...	...
95	19951698	0	0
96	5731411	0	0
97	23912372	0	0
98	16188013	44	2017
99	21831122	47	2023

100 rows × 3 columns

DESIGNATED_STATES

List of designated states. Let's display the applications with corresponding list of designated state and the state type.

In [2]:
```
des_states = db.query(
    REG109_DESIGNATED_STATES.id,
    REG109_DESIGNATED_STATES.state_type,
    REG109_DESIGNATED_STATES.designated_states
).limit(1000)

des_states_df = patstat.df(des_states)
des_states_df
```

Out[2]:

	id	state_type	designated_states
0	21730192	EXT	BA,ME
1	14863086	MEM	AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,H...
2	16736609	MEM	AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,H...
3	3733441	EXT	AL,LT,LV,MK
4	18755708	EXT	BA,ME
...	...	...	...
995	22938489	MEM	AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,H...
996	17747768	VAL	MA,MD
997	15808341	VAL	MA
998	106314	MEM	AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LI,LU,M...
999	12187737	MEM	AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,H...

1000 rows × 3 columns

We can also check which states appear more frequently as designated states.

In [13]:
```
# Create an empty list to add the number of symbols for each appl
ication id
num_labels = []

# Iterate over the list of symbols (over the rows under "ipc_tex
t")
for label in des_states_df['designated_states']:
    # Convert label in a list: a new element is defined every tim
e a comma is encountered
    sequence = label.split(",")
    # Append the length of list just created
    num_labels.append(len(sequence))

# Create a new column of the dataframes containing the length of
the symbols sequence
des_states_df['designated_states'] = num_labels
# Sort the dataframe by the number_of_labels attribute in descend
ing order
des_states_df = des_states_df.sort_values(by='designated_states',
ascending=False)
# Save the top 10 applications
most_labels = des_states_df.head(100)
most_labels
```

Out[13]:

	id	designated_states
528	23183229	39
562	23700406	39
167	23158409	39
168	22962739	39
270	22812839	39
...	...	...
851	18206751	38
437	21160455	38
433	17862884	38
480	17382344	38
424	21199205	38

100 rows × 2 columns

In [ ]: