The Cooperative Patent Classification by DOCDB Family (TLS225_DOCDB_FAM_CPC)

Welcome to the Cooperative Patent Classification by DOCDB Family Table in PATSTAT, also TLS224_APPLN_CPC, from which table TLS224 is derived. All applications of the same DOCDB simple family have the same cooperative patent classifications (CPC symbols) assigned. The same CPC symbol can be assigned to the same DOCDB family by one or more patent offices.

A small number of invalid or obsolete CPC classification codes can possibly occur.

In [17]:
```
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 TLS225_DOCDB_FAM_
CPC
```

DOCDB_FAMILY_ID

We already encountered this attribute in table TLS201, which is the identifier of a DOCDB simple family. We can join tables TLS225 and TLS201 via this field.

In [5]:
```
# Import table TLS201
from epo.tipdata.patstat.database.models import TLS201_APPLN

show_join = db.query(
    TLS201_APPLN.docdb_family_id,
    TLS201_APPLN.appln_auth,
    TLS225_DOCDB_FAM_CPC.cpc_class_symbol
).join(
    TLS201_APPLN, TLS225_DOCDB_FAM_CPC.docdb_family_id == TLS201_
APPLN.docdb_family_id
).limit(1000)

show_join_df = patstat.df(show_join)
show_join_df
```

Out[5]:

	docdb_family_id	appln_auth	cpc_class_symbol
0	50050528	TW	H01L2924/13091
1	50050528	TW	H01L2924/13091
2	50050528	CN	H01L2924/13091
3	50050528	CN	H01L2924/13091
4	50050528	US	H01L2924/13091
...	...	...	...
995	27397859	US	C01P2006/60
996	24703631	DE	F02B 3/06
997	24703631	US	F02B 3/06
998	24703631	AT	F02B 3/06
999	24703631	EP	F02B 3/06

1000 rows × 3 columns

CPC_CLASS_SYMBOL

Classification symbol according to the Cooperative Patent Classification. It consists of up to 19 characters (A-Z, 0-9, /, space).

As aforementioned, all the applications of the same DOCDB simple family have the same CPC symbols assigned. For computational reasons, we show it for one DOCDB family only. The idea is to query one particular `docdb_family_id` and convert the result in a dataframe. Then, we iterate on all the applications belonging to that family with a `for loop`. At each iteration we limit the dataset to the rows having the `appln_id` attribute equal to the current application. We define the list of CPC symbols corresponding to both the previous and current application and check if they have the same content. If not, we increment a pre-defined counter.

In [34]:
```
from sqlalchemy import func
from sqlalchemy import select
import pandas as pd

mismatch = 0  # Define the counter for different lists of CPC sym
bols

docdb_fam = 23307812  # Choose a specific DOCDB family

# Query the docdb_family_id, the appln_id and the cpc_class_symbo
l filtering only the selected DOCDB family
query = db.query(
    TLS201_APPLN.docdb_family_id,
    TLS201_APPLN.appln_id,
    TLS225_DOCDB_FAM_CPC.cpc_class_symbol
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).filter(
    TLS201_APPLN.docdb_family_id == docdb_fam  # Filter the desir
ed DOCDB family
)

# Convert the resulting table into a Pandas dataframe object
dataframe = patstat.df(query)
previous = None  # Previous list initially set to None
for application in dataframe['appln_id'].unique():  # Iterate on
all the applications in the family
    data = dataframe[dataframe['appln_id'] == application]  # Red
uce the dataset to the rows with appln_id equal to the current ap
plication
    curr_list = data['cpc_class_symbol']  # Define the list of CP
C symbols corresponding to the current application
    if previous is None:  # At the first round 'previous' is None
so simply update it to 'curr_list'
        previous = curr_list
    else:
        if (set(previous) == set(curr_list)) == False:  # Check i
f the current list and the previous one contain the same elements
(the 'set' function ignores duplicates): if it is the case update
'previous' to 'curr_list' otherwise add 1 to the counter and upda
te
```

```
            mismatch += 1
            previous = curr_list
        else:
            previous = curr_list

if mismatch == 0:
    print("All applications of the same DOCDB simple family have
the same CPC symbols assigned.")
else:
    print("There are "+str(mismatch)+" DOCDB families for which t
he claim does not hold true.")
```

```
All applications of the same DOCDB simple family have the same CP
C symbols assigned.
```

CPC_GENER_AUTH

This attribute indicates the patent office that classified the application with a CPC symbol.

Let's check for how many applications the attributes `appln_auth` and `cpc_gener_auth` differ.

In [6]:
```
count_clashes = db.query(
    func.count(TLS201_APPLN.appln_id).label('clashes_counting')
).select_from(
    TLS201_APPLN  # Use select_from to specify how to join the tw
o tables an avoid an InvalidRequestError
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).filter(
    TLS201_APPLN.appln_auth != TLS225_DOCDB_FAM_CPC.cpc_gener_aut
h
)

count_clashes_df = patstat.df(count_clashes)
count_clashes_df = count_clashes_df['clashes_counting'].item()
print("There are "+str(count_clashes_df)+" applications for which
application authority and IPC generating authority differ.")
```

```
There are 465084927 applications for which application authority
and IPC generating authority differ.
```

For the applications having different values for the two attributes, we rank the generative authorities.

In [7]:
```
most_gen_auth = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_gener_auth,
    func.count(TLS201_APPLN.appln_id).label('Number of occurrenci
es')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).filter(
    TLS201_APPLN.appln_auth != TLS225_DOCDB_FAM_CPC.cpc_gener_aut
h
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_gener_auth
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

most_gen_auth_df = patstat.df(most_gen_auth)
most_gen_auth_df
```

Out[7]:

	cpc_gener_auth	Number of occurrencies
0	EP	252940058
1	US	151498037
2	KR	34348114
3	CN	12897475
4	IL	5115228
5	RU	2499480
6	GB	2265870
7	EA	898131
8	BR	793910
9	NO	421694
10	SE	389354
11	AU	298213
12	FI	198411
13	ES	157336
14	AT	143068
15	MX	76621
16	CH	39808
17	DK	39250
18	CZ	28111
19	GR	12156
20	HU	11198
21	PT	8267
22	PL	3852
23	RO	1285

Notice the difference with the IPC ranking. EPO is still the first generative authority but the Japanese authority, second in the IPC ranking, is not even in the CPC ranking. Indeed, Japan uses a different classification system, as we have seen in table TLS222.

CPC_VERSION

This field simply indicates which is the version of the CPC. It is a date between '2013-01-01' and current date.

Let's find out how many CPC versions there are in PATSTAT.

In [8]:
```
num_versions = db.query(
    func.count(TLS225_DOCDB_FAM_CPC.cpc_version.distinct()).labe
l('Distinct versions')
)

num_versions = patstat.df(num_versions)
num_versions = num_versions['Distinct versions'].item()
print("There are "+str(num_versions)+" distinct versions in PATST
AT.")
```

```
There are 60 distinct versions in PATSTAT.
```

We can also check which is the most common CPC version.

In [9]:
```
version = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_version,
    func.count(TLS201_APPLN.appln_id).label('Number of applicatio
ns')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_version
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

version_df = patstat.df(version)
version_df
```

Out[9]:

	cpc_version	Number of applications
0	2013-01-01	516664358
1	2017-12-29	22878477
2	2015-01-15	9953071
3	2021-01-01	5560944
4	2015-04-01	5338675
5	2023-02-01	4964535
6	2020-01-01	4659990

7	2022-01-01	4485449
8	2023-01-01	4033449
9	2018-12-31	3254043
10	2014-11-03	3239521
11	2019-01-31	3189194
12	2015-11-01	2857426
13	2016-02-14	2653929
14	2016-07-31	2503994
15	2017-07-31	2438608
16	2024-01-01	2388517
17	2022-05-01	2082312
18	2018-01-31	2067027
19	2020-02-01	1392357
20	2020-05-01	1234122
21	2021-05-01	1218718
22	2020-08-01	1206037
23	2018-04-30	1172217
24	2016-12-31	1132296
25	2015-05-01	1083292
26	2016-05-01	1029185
27	2018-07-31	939125
28	2014-12-01	878843
29	2014-02-04	798448
30	2017-04-30	727551
31	2015-10-01	654192
32	2014-09-02	584155
33	2019-04-30	479445
34	2014-10-01	302255
35	2023-08-01	301514
36	2016-01-14	284967
37	2022-08-01	259498
38	2023-05-01	242602
39	2021-08-01	202166
40	2019-07-31	191645

41	2016-10-31	169995
42	2017-01-31	147127
43	2022-02-01	83916
44	2014-06-03	68443
45	2013-12-03	42477
46	2015-07-01	17930
47	2013-11-05	14308
48	2013-09-05	10190
49	2020-07-31	5098
50	2013-06-08	2026
51	2014-07-01	1306
52	2015-08-31	1275
53	2013-03-28	631
54	2019-01-01	266
55	2013-07-17	131
56	2018-01-01	72
57	2019-05-01	15
58	2018-05-01	9
59	2018-08-01	4

CPC_POSITION

Indicates the position of the class symbol in the sequence of classes that form the classification.

The domain is represented by 1 character:

- F = fist
- L = later
- space = unidentified

In [10]:
```
position = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_position,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_position
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

position_df = patstat.df(position)
position_df
```

Out[10]:

	cpc_position	Occurrencies
0	L	498647193
1	F	123446175

CPC_VALUE

Indication of the value of the classification, i.e. is the class symbol relating to the invention or to aspects not related to the invention (but in the application). The value can be:

- I = Invention
- N = Additional (Non-Invention)

In [11]:
```
value = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_value,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_value
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

value_df = patstat.df(value)
value_df
```

Out[11]:

	cpc_value	Occurrencies
0	I	434576250
1	A	187517118

CPC_ACTION_DATE

The date of assigning the classification symbol.

In [16]:
```
act_date = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.docdb_family_id,
    TLS225_DOCDB_FAM_CPC.cpc_action_date,
    TLS225_DOCDB_FAM_CPC.cpc_class_symbol,
    TLS225_DOCDB_FAM_CPC.cpc_status,
    TLS225_DOCDB_FAM_CPC.cpc_data_source
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).filter(TLS201_APPLN.appln_id == 45505059)#.limit(1000)

# 45505059
act_date_df = patstat.df(act_date)
act_date_df
```

Out[16]:

	appln_id	docdb_family_id	cpc_action_date	cpc_class_symbol	cpc_status	cpc_data_s
0	45505059	2134330	2016-12-23	D01H 1/00	B	
1	45505059	2134330	2016-12-23	D01H 1/00	B	

CPC_STATUS

Indication of whether the CPC is as originally assigned or whether and how it has been reclassified. The domain consists of 1 character:

- B = basic or original data, that is the first data assigned to the document
- R = reclassified data, i.e. data changed due to a change in the classification schemes

In [13]:
```
status = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_status,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_status
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

status_df = patstat.df(status)
status_df
```

Out[13]:

	cpc_status	Occurrencies
0	B	582334647
1	R	39758721

CPC_DATA_SOURCE

Source of CPC classification data. The domain consists of 1 character:

- H = Human generated data (intellectual classification by persons)
- C = Classification by concordance, e.g. by copying symbols allocated by other patent offices, or by copying IPC symbols into CPC allocations
- G = Classification symbols generated by software using automatic analysis of the content of the patent document

In [14]:
```
source = db.query(
    TLS225_DOCDB_FAM_CPC.cpc_data_source,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS225_DOCDB_FAM_CPC, TLS201_APPLN.docdb_family_id == TLS225_
DOCDB_FAM_CPC.docdb_family_id
).group_by(
    TLS225_DOCDB_FAM_CPC.cpc_data_source
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

source_df = patstat.df(source)
source_df
```

Out[14]:

	cpc_data_source	Occurrencies
0	H	584504643
1	C	30212846
2	G	7375879

In [ ]:
```
```