The International Patent Classification Table (TLS209_APPLN_IPC)

Welcome to the **International Patent Classification Table** in PATSTAT, namely table TLS209_APPLN_IPC. The table contains all international patent classifications linked to the applications. The set of classifications linked to a single application is a de-duplicated merge of all classifications of the various publication instances linked to the specific application. Additionally, only the latest version of the IPC classifications is used. This means that the user does not have to worry about reclassifications because older applications will always be classified according to the latest IPC version.

Information on classification according to the International Patent Classification (IPC) can be found on the WIPO website (https://www.wipo.int/classifications/ipc/en/).

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 TLS209_APPLN_IPC
```

APPLN_ID ¶

This is the unique identifier for each application that allows to link this table to table TLS201.

In [2]:

```
# Import table TLS201
from epo.tipdata.patstat.database.models import TLS201_APPLN

show_join = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_auth,
    TLS209_APPLN_IPC.ipc_class_symbol
).join(
    TLS201_APPLN, TLS209_APPLN_IPC.appln_id == TLS201_APPLN.appln
_id
).limit(1000)

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

Out[2]:

	appln_id	appln_auth	ipc_class_symbol
0	605758632	AR	A23L 33/15
1	457199360	AR	A61K 31/437
2	423580474	AT	F23N 1/02
3	56760638	AT	E04B 1/343
4	375544956	AU	A01K 15/04
...	...	...	...
995	495374036	CN	F21W 107/10
996	510193008	CN	F21W 107/10
997	423968323	CN	F21W 111/04
998	418124148	CN	F21Y 115/10
999	602638908	CN	F22B 37/22

1000 rows × 3 columns

IPC_CLASS_SYMBOL

Classification symbol according to the International Patent Classification, eights edition (entered into force January 1, 2006). It consists of up to 15 characters, including the Latin letters, Arabic numbers, space and '/'. Some spaces may be required since the slash '/' is always on the 9th position.

Each application can be associated with more than one class symbol. Let's verify this by counting the `ipc_class_symbol` and grouping by `appln_id`. We show only those applications that have a count greater than 1.

In [3]:

```
from sqlalchemy import func

symb_appln = db.query(
    TLS209_APPLN_IPC.appln_id,
    func.count(TLS209_APPLN_IPC.ipc_class_symbol).label('Number of symbols')
).group_by(
    TLS209_APPLN_IPC.appln_id
).having(
    func.count(TLS209_APPLN_IPC.ipc_class_symbol) > 1  # Consider only applications with more than 1 class symbol
).order_by(
    func.count(TLS209_APPLN_IPC.ipc_class_symbol).desc()
).limit(1000)

symb_appln_df = patstat.df(symb_appln)
symb_appln_df
```

Out[3]:

	appln_id	Number of symbols
0	8161391	246
1	8687976	244
2	8418577	243
3	8154006	242
4	7921091	242
...	...	...
995	38290131	128
996	315635470	128
997	328045065	128
998	590000518	128
999	24371026	128

1000 rows × 2 columns

In [13]:

```
check = db.query(
    TLS209_APPLN_IPC.appln_id,
    TLS209_APPLN_IPC.ipc_class_symbol
).filter(
    TLS209_APPLN_IPC.appln_id == 8161391
)

check_df = patstat.df(check)
check_df
```

Out[13]:

	appln_id	ipc_class_symbol
0	8161391	H04W 48/16
1	8161391	H04J 13/16
2	8161391	G06F 9/50
3	8161391	H04W 28/02
4	8161391	G09G 1/28
...	...	...
241	8161391	H04W 88/18
242	8161391	H04N 5/60
243	8161391	G09G 1/16
244	8161391	H04W 24/00
245	8161391	H04N 101/00

246 rows × 2 columns

In [14]:
```
from sqlalchemy import select

sub = check.subquery()

duplicates = db.query(
    func.count(sub.c.appln_id).label('Number of duplicates'),
    sub.c.ipc_class_symbol
).filter(
    sub.c.appln_id == 8161391
).group_by(
    sub.c.ipc_class_symbol
).order_by(
    func.count(sub.c.appln_id)
)

duplicates_df = patstat.df(duplicates)
duplicates_df
```

Out[14]:

	Number of duplicates	ipc_class_symbol
0	1	G09G 3/02
1	1	G06K 17/00
2	1	H04W 28/18
3	1	G01S 5/02
4	1	H04M 1/73
...	...	...
241	1	H04N 9/79
242	1	G06F 1/16
243	1	H04W 8/26
244	1	H03D 7/00
245	1	G06F 13/362

246 rows × 2 columns

The contrary is also true, i.e. the same symbol can be assigned to more than one application ID.

In [4]:

```
appln_symb = db.query(
    TLS209_APPLN_IPC.ipc_class_symbol,
    func.count(TLS209_APPLN_IPC.appln_id).label('Number of symbols')
).group_by(
    TLS209_APPLN_IPC.ipc_class_symbol
).having(
    func.count(TLS209_APPLN_IPC.appln_id) > 1  # Consider only applications with more than 1 class symbol
).order_by(
    func.count(TLS209_APPLN_IPC.appln_id).desc()
)

appln_symb_df = patstat.df(appln_symb)
appln_symb_df
```

Out[4]:

	ipc_class_symbol	Number of symbols
0	A61P 35/00	703764
1	G06F 17/30	502457
2	H04L 29/06	481798
3	A61P 43/00	479164
4	H04L 29/08	393321
...	...	...
80298	B41L 47/28	2
80299	B64U 50/16	2
80300	H04L 12/953	2
80301	F27D 23/02	2
80302	B41L 27/20	2

80303 rows × 2 columns

IPC_CLASS_LEVEL

This attribute denotes whether an authority classified either in the full IPC, in main groups or in sub classes only. The domain of this attribute is 1 character:

- A = classification in the full IPC
- C = classification in main groups only
- S = classification in subclasses only

Let's find out which is the most frequent type of classifications.

In [5]:

```
# Count the number of applications (appln_id) grouped by ipc_class_level
levels = db.query(
    TLS209_APPLN_IPC.ipc_class_level,
    func.count(TLS201_APPLN.appln_id).label('Total number')
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.appln_id
).group_by(
    TLS209_APPLN_IPC.ipc_class_level
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()  # Rank according to the most frequent class level
)

levels_df = patstat.df(levels)
levels_df
```

Out[5]:

	ipc_class_level	Total number
0	A	342062277
1	S	1673043
2	C	354752

IPC_VERSION

This is the version of the IPC. First of all, we can check how many applications there are in PATSTAT.

In [6]:

```
num_versions = db.query(
    func.count(TLS209_APPLN_IPC.ipc_version.distinct()).label('Distinct versions')
)

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

```
There are 21 distinct versions in PATSTAT.
```

Now let's find out which is the most frequent version.

In [7]:

```
version = db.query(
    TLS209_APPLN_IPC.ipc_version,
    func.count(TLS201_APPLN.appln_id).label('Number of applications')
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.appln_id
).group_by(
    TLS209_APPLN_IPC.ipc_version
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

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

Out[7]:

	ipc_version	Number of applications
0	2006-01-01	300559752
1	2012-01-01	4160438
2	2016-01-01	4139771
3	2018-01-01	3675056
4	2009-01-01	3348840
5	2013-01-01	3258030
6	2014-01-01	3149906
7	2022-01-01	3078994
8	2011-01-01	3041818
9	2010-01-01	2970072
10	2019-01-01	2790816
11	2021-01-01	2412958
12	2020-01-01	2160673
13	2017-01-01	1802723
14	2015-01-01	1513848
15	2023-01-01	1280732
16	2007-01-01	287669
17	2008-01-01	203995
18	2007-10-01	183171
19	2024-01-01	41818
20	2008-04-01	28992

IPC_VALUE

This attributes tells the value of the classification, i.e. the class symbol relating to the invention or to aspects not related to the invention but present in the application. The domain is 1 character:

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

In [8]:

```
value = db.query(
    TLS209_APPLN_IPC.ipc_value,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.a
ppln_id
).group_by(
    TLS209_APPLN_IPC.ipc_value
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

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

Out[8]:

	ipc_value	Occurrencies
0	I	333273873
1	N	10816193
2		6

IPC_POSITION

Indicates the position of the class symbol in the sequence of classes that form the classification. For patent authorities (e. g. USPTO) where the law entails the concept of "first" class, the first class symbol in a list of class symbols is the main class. For other authorities, like the EPO, there is no meaning in the position - classes may be quoted in alphabetical order for instance.

The domain is represented by 1 character:

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

In [9]:

```
position = db.query(
    TLS209_APPLN_IPC.ipc_position,
    func.count(TLS201_APPLN.appln_id).label('Occurrencies')
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.a
ppln_id
).group_by(
    TLS209_APPLN_IPC.ipc_position
).order_by(
    func.count(TLS201_APPLN.appln_id).desc()
)

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

Out[9]:

	ipc_position	Occurrencies
0	L	176021243
1	F	87013497
2		81055332

IPC_GENER_AUTH

This attribute indicates which authority generated the IPC. It can differ from the application authority.

Let's find the applications that do not have 'EP' as `appln_auth` but having 'EP' as `ipc_gener_auth` . We limit the result to the first 1000 results for sake of computation time.

In [10]:

```
clashes = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_auth,
    TLS209_APPLN_IPC.ipc_gener_auth
).join(
    TLS201_APPLN, TLS209_APPLN_IPC.appln_id == TLS201_APPLN.appln_id
).filter(
    TLS201_APPLN.appln_auth != 'EP',
    TLS209_APPLN_IPC.ipc_gener_auth == 'EP'
).limit(1000)

clashes_df = patstat.df(clashes)
clashes_df
```

Out[10]:

	appln_id	appln_auth	ipc_gener_auth
0	24265666	IL	EP
1	43605349	SE	EP
2	24166976	IL	EP
3	18502156	FI	EP
4	17785210	ES	EP
...	...	...	...
995	36426858	JP	EP
996	1351874	AT	EP
997	36819909	JP	EP
998	48345149	US	EP
999	18695322	FR	EP

1000 rows × 3 columns

Suppose that we want to know how many times the attributes `appln_auth` and `ipc_gener_auth` differ in the entire database.

In [11]:

```
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 two tables an avoid an InvalidRequestError
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.appln_id
).filter(
    TLS201_APPLN.appln_auth != TLS209_APPLN_IPC.ipc_gener_auth
)

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 121434052 applications for which application authority
and IPC generating authority differ.
```

Top generative authorities

Of these applications, which is the most frequent IPC generative authority?

In [12]:

```
most_gen_auth = db.query(
    TLS209_APPLN_IPC.ipc_gener_auth,
    func.count(TLS201_APPLN.appln_id).label('Number of occurrencies')
).join(
    TLS209_APPLN_IPC, TLS201_APPLN.appln_id == TLS209_APPLN_IPC.appln_id
).filter(
    TLS201_APPLN.appln_auth != TLS209_APPLN_IPC.ipc_gener_auth
).group_by(
    TLS209_APPLN_IPC.ipc_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[12]:

	ipc_gener_auth	Number of occurrencies
0	EP	79136639
1	JP	31715222
2	US	2562692
3	KR	1959619
4	RU	1680677
...	...	...
85	MK	2
86	GH	2
87	CG	2
88	VN	1
89	SM	1

90 rows × 2 columns

The top 2 generative authorities are EPO and the Japanese authority. Notice that they classified one order of magnitude of applications more than the following authorities.

In [ ]: