Paststat - Exercise 4

In this exercise we will take a look at the classification tables in PATSTAT global, and learn how to perform left joins.

In [3]:
```
# Importing the patstat client
from epo.tipdata.patstat import PatstatClient

# Initialize the PATSTAT client
patstat = PatstatClient()

# Access ORM
db = patstat.orm()
# Importing tables as models
from epo.tipdata.patstat.database.models import TLS201_APPLN, TLS
224_APPLN_CPC, TLS209_APPLN_IPC
```

IPC Classifications

IPC classifications are stored in the `tls209_appln_ipc` table. This table holds information about the IPC classification symbols assigned to patent applications. Important fields in this table include:

- `appln_id` : The unique identifier for the patent application. This field is typically used to link the 209 table with the `TLS201_APPLN` table.
- `ipc_class_symbol` : The IPC classification symbol assigned to the application.
- `ipc_version` : The version of the IPC classification system used.
- `ipc_position` : The position of the IPC classification symbol in the classification hierarchy.
- `ipc_gener_auth` : The authority that generated the IPC classification.

A query about IPC classification with an inner join

Let's build a query aims to retrieve a dataset of South African patent applications filed after 2010, along with their respective IPC classifications and the authority that generated these classifications.

We need to perform a join between `TLS209_APPLN_IPC` and `TLS201_APPLN` tables to link the IPC classification data to the corresponding application data. By joining these tables on the appln_id, we can combine the classification symbols and the authority that generated them with specific details about the patent applications, such as the application authority and number.

In [9]:
```
q = db.query(
    TLS201_APPLN.appln_auth,
    TLS201_APPLN.appln_nr,
    TLS209_APPLN_IPC.ipc_class_symbol,
    TLS209_APPLN_IPC.ipc_gener_auth
).join(
    TLS209_APPLN_IPC, TLS209_APPLN_IPC.appln_id == TLS201_APPLN.a
ppln_id,
).filter(
    TLS201_APPLN.appln_auth == 'ZA',
    TLS201_APPLN.appln_filing_year > 2010
).order_by(
    TLS201_APPLN.appln_nr
)

ipc_inner = patstat.df(q)

ipc_inner
```

Out[9]:

	appln_auth	appln_nr	ipc_class_symbol	ipc_gener_auth
0	ZA	10401D	C07D 307/60	EP
1	ZA	10401D	C07D 239/32	EP
2	ZA	10401D	C07D 241/18	EP
3	ZA	10401D	C07D 277/22	DE
4	ZA	10401D	C07D 213/64	EP
...	...	...	...	...
34257	ZA	928156D	C08L 25/06	EP
34258	ZA	928156D	C08L 39/08	EP
34259	ZA	928156D	C08L 23/02	EP
34260	ZA	928156D	D06P 5/22	EP
34261	ZA	928156D	C08L 39/06	EP

34262 rows × 4 columns

Understanding the results

We can see that typically there are multiple entries in the joined table for a given application. This happens because table `TLS209_APPLN_IPC` has a many-to-many relationship between IPC symbols and applications.

We also need to consider that the default join in ORM is an inner join. Using an inner join in this query means that only records with matching `appln_id` values in both the `TLS209_APPLN_IPC` and `TLS201_APPLN` tables will be included. This ensures that each resulting record contains both application details and corresponding IPC classification data.

The effect of this inner join is:

- Only applications with at least one IPC classification are included.
- Applications without an IPC classification are excluded.
- IPC classification entries without a corresponding application are also excluded.

Left queries in ORM

The implementation of ORM in TIP allows us to perform left queries. If we change the join in the last example to a left join with `isouter=True` , the behavior of the query will change as follows:

Using a left join (`isouter=True`) in this query means that all records from the `TLS201_APPLN` table (the left table) will be included in the result set, regardless of whether there is a matching record in the `TLS209_APPLN_IPC` table (the right table). This ensures that each application detail from the `TLS201_APPLN` table is included, along with the corresponding IPC classification data if it exists.

The effect of this left join is:

- All applications from the `TLS201_APPLN` table will be included, even if they do not have an IPC classification.
- Applications without an IPC classification will have `NULL` values for the columns from the `TLS209_APPLN_IPC` table.
- Applications with at least one IPC classification will have the corresponding classification data included.
- IPC classification entries without a corresponding application will still be excluded.

In [19]:
```
q = db.query(
    TLS201_APPLN.appln_nr,
    TLS209_APPLN_IPC.ipc_class_symbol,
).join(
    TLS209_APPLN_IPC, TLS209_APPLN_IPC.appln_id == TLS201_APPLN.a
ppln_id, isouter=True # making it a left join
).filter(
    TLS201_APPLN.appln_auth == 'ZA',
    TLS201_APPLN.appln_filing_year > 2010
).order_by(
    TLS209_APPLN_IPC.ipc_class_symbol
)

ipc_left = patstat.df(q)
print (f'Length of the inner join {len(ipc_inner):,}')
print (f'Length of the left join {len(ipc_left):,}')
ipc_left
```

```
Length of the inner join 34,262
Length of the left join 89,240
```

Out[19]:

	appln_nr	ipc_class_symbol
0	201806601	None
1	201806017	None
2	201506968	None
3	A201500548	None
4	201502036	None
...	...	...
89235	201106915	H05K
89236	201301890	H05K
89237	201209651	H05K
89238	201404709	H05K
89239	201101036	H05K

89240 rows × 2 columns

Filtering the left join using pandas

We can see that the left join indeed gives results where an application has no IPC classification. The size of the `ipc_left` dataframe is also bigger than the `ipc_inner` dataframe. Let's get only the applications with no IPC classification.

We could modify the query with a filter to retrieve only null values, but we can also filter the existing `ipc_left` dataframe with pandas. We will use the `isnull()` method of pandas to generate a boolean series where each entry is True if the corresponding ipc_class_symbol is NULL, and False otherwise. With that boolean series we can filter the `ipc_outer` dataframe, so we get only the applications without a classification.

In [20]:
```
filter_series = ipc_outer['ipc_class_symbol'].isnull() # a boolean series with True for null values
filtered_outer = ipc_outer[filter_series] # the filtered dataframe using the boolean series

print (f'Length of the inner join {len(ipc_inner):,}')
print (f'Length of the left join {len(ipc_left):,}')
print (f'Length of the filtered dataframe {len(filtered_outer):,}')
filtered_outer
```

```
Length of the inner join 34,262
Length of the left join 89,240
Length of the filtered dataframe 54,978
```

Out[20]:

	appln_nr	ipc_class_symbol
0	201806601	None
1	201806017	None
2	201506968	None
3	A201500548	None
4	201502036	None
...	...	...
54973	F201800954	None
54974	202302944	None
54975	202109684	None
54976	201802347	None
54977	202302998	None

54978 rows × 2 columns