

The Patstat library - Lesson 3

This notebook expands on the second lesson about Patstat. We take a look at the two tables that contain information about inventors and applicants, and learn to join tables using ORM.

The `tls207_pers_appln` persons table

PATSTAT links two types of persons to a given application: inventors and applicants. Please note that applicants can be physical persons or legal entities such as corporations. There can be a many-to-many relation between persons and applications. The `tls207_pers_appln` table serves as an intermediary table that links applications from the `tls201_appln` table to persons (both inventors and applicants).

Fields in `tls207_pers_appln`

- `appln_id` : This field is a foreign key that references the `appln_id` in the `tls201_appln` table, establishing a link between an application and its related persons.
- `person_id` : This field is a foreign key that references the `person_id` in the `tls206_person` table, identifying the person associated with the application.
- `applt_seq_nr` : This field denotes the sequence number for applicants, allowing the identification of multiple applicants for a single application.
- `invt_seq_nr` : This field denotes the sequence number for inventors, distinguishing inventors from other types of persons associated with the application.

The most influential inventor of the decade

In lesson 2 we used the `nb_citing_docdb_fam` field of the applications table to determine the granted European patents that have been cited by other patents the most, as a proxy for finding the most influential invention of the decade.

In this lesson we will find out who is the most influential inventor by the same metric. First we need to know who are the inventors listed in each application from our query.

Getting the persons per application

To find the persons associated with an application, we query PATSTAT with a `JOIN` between `tls207_pers_appln` and `tls201_appln`. We use the `appln_id` for the join, as this is the field shared between the two tables.

```
In [2]: # 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
207_PERS_APPLN
```

```
In [12]: # Define the query in ORM
q = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_nr,
    TLS201_APPLN.nb_citing_docdb_fam,
    TLS207_PERS_APPLN.person_id,      # person ID from table 207
    TLS207_PERS_APPLN.invt_seq_nr,   # inventor sequence number f
from table 207, different than 0 if the person is an inventor
    TLS207_PERS_APPLN.applt_seq_nr  # applicant sequence number
from table 207, different than 0 if the person is an applicant
).join(
    TLS207_PERS_APPLN, TLS201_APPLN.appln_id == TLS207_PERS_APPL
N.appln_id # tables 201 and 207 are joined with the common field
appln_id
).filter(
    TLS201_APPLN.appln_filing_year >= 2020,
    TLS201_APPLN.appln_auth == 'EP',
    TLS201_APPLN.granted == 'Y'
).order_by(
    TLS201_APPLN.appln_id.desc()
)

# Creating a dataframe with the results
res = patstat.df(q)

res
```

Out[12]:

	appln_id	appln_nr	nb_citing_docdb_fam	person_id	invt_seq_nr	applt_seq_nr
0	583185955	22208726		83401926	5	0
1	583185955	22208726		88844123	3	0
2	583185955	22208726		74415260	0	1
3	583185955	22208726		83169538	4	0
4	583185955	22208726		83168542	1	0
...
98011	524353208	20150109		81639814	1	0
98012	524353208	20150109		74444155	2	0
98013	524353208	20150109		46993391	3	0
98014	524353208	20150109		74608749	4	0
98015	521666076	19796005		79309198	1	1

98016 rows × 6 columns

Understanding the results

Since we joined two tables, we can see that we can have multiple entries for a given application ID, while when we query the applications table the application ID is unique. This happens because an application will be linked with at least two persons, one applicant and one inventor. We can also see that some persons have a 0 in the `invt_seq_nr` field, and a value greater than 0 in the `applt_seq_nr`. This means that those persons are just applicants. It is important to note that in some applications the inventor and the applicant may be the same person.

Filtering out applicants

We need to filter out the persons who have a value 0 in the inventor sequence number, since those persons are **just** applicants.

```
In [4]: # Importing necessary modules
from epo.tipdata.patstat.database.models import TLS201_APPLN, TLS
207_PERS_APPLN

# Define the query in ORM
q = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_nr,
    TLS201_APPLN.nb_citing_docdb_fam, # number of families citin
g the application
    TLS207_PERS_APPLN.person_id
).join(
    TLS207_PERS_APPLN, TLS201_APPLN.appln_id == TLS207_PERS_APPL
N.appln_id
).filter(
    TLS201_APPLN.appln_filing_year >= 2020,
    TLS201_APPLN.appln_auth == 'EP',
    TLS201_APPLN.granted == 'Y',
    TLS207_PERS_APPLN.invt_seq_nr > 0 # filter to exclude applic
ants that are not inventors
).order_by(
    TLS207_PERS_APPLN.person_id.desc()
)

# Creating a dataframe with the results
res = patstat.df(q)

res
```

Out[4]:

	appln_id	appln_nr	nb_citing_docdb_fam	person_id
0	530745512	20722741	4	89023590
1	530646374	20721357	0	89023590
2	529952419	20719307	3	89023545
3	529498216	20717027	2	89023516
4	530646358	20721341	4	89023397
...
71183	543216254	20215597	0	263
71184	539175428	20201433	0	263
71185	542802321	20213057	1	263
71186	544408074	21151856	0	263
71187	535625565	20187827	1	223

71188 rows × 4 columns

The tls206_person Table

We have the `person_id` in the table above as the unique identifier of the inventors with granted patents that have been filed this decade, but we do not know their names. We need another table for that.

The `tls206_person` table contains details about inventors and applicants. This table is typically joined with the `tls207_pers_appln` table to link persons to specific patent applications.

Fields of `tls206_person`:

- `person_id`: Unique identifier for each person.
- `person_name`: Name of the person or organization.
- `doc_std_name`: Standardized name for the person or organization.
- `person_address`: Address of the person.
- `person_ctry_code`: Country code associated with the person.
- `psn_sector`: Sector classification of the person (e.g., academia, industry).

Adding the person name to the query

We need to add a further `JOIN` to our query, where we join the `TLS207_PERS_APPLN` and the `TLS206_PERSON` tables, with the common field `person_id`. We will now have three tables joined as a result

```
In [5]: # Importing table 206
from epo.tipdata.patstat.database.models import TLS201_APPLN, TLS
207_PERS_APPLN, TLS206_PERSON

# Define the query in ORM
q = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_nr,
    TLS201_APPLN.appln_filing_date,
    TLS201_APPLN.nb_citing_docdb_fam,
    TLS207_PERS_APPLN.person_id,
    TLS206_PERSON.person_name # inventor's name as found in tabl
e 206
).join(
    TLS207_PERS_APPLN, TLS201_APPLN.appln_id == TLS207_PERS_APPL
N.appln_id # tables 201 and 207 are joined with the common field
appln_id
).join(
    TLS206_PERSON, TLS207_PERS_APPLN.person_id == TLS206_PERSON.p
erson_id # tables 206 and 207 are joined with the common field
person_id
).filter(
    TLS201_APPLN.appln_filing_year >= 2020,
    TLS201_APPLN.appln_auth == 'EP',
    TLS201_APPLN.granted == 'Y',
    TLS207_PERS_APPLN.invt_seq_nr > 0
).order_by(
    TLS206_PERSON.person_name.desc()
)

# Creating a dataframe with the results
res = patstat.df(q)

res
```

Out[5]:

	appln_id	appln_nr	appln_filing_date	nb_citing_docdb_fam	person_id	person_na
0	524815902	20151377	2020-01-13	0	54453897	Ünker, Ya
1	528859167	20714139	2020-03-11	0	83233746	Überm
2	524947529	20152100	2020-01-16	1	4610366	Übelac Röhl
3	524947511	20152090	2020-01-16	1	4610366	Übelac Röhl
4	527050575	20159886	2020-02-27	0	81754560	ÜYÜ Mete
...
71183	529839878	20170374	2020-04-20	0	263	
71184	536873482	20192527	2020-08-25	0	263	
71185	526146908	20156852	2020-02-12	0	263	
71186	527201101	20160388	2020-03-02	0	263	
71187	566486262	22154163	2022-01-31	0	263	

71188 rows × 6 columns

Aggregating citations

Since we have a many-to-many relationship between persons and applications, we can see that some inventors have more than one granted patent resulting from our query. In order to see the ranking of influential inventors, we need to aggregate the `nb_citing_docdb_fam` field per unique name. For that we need to import a module from SQLAlchemy called `func`. This module contains several useful methods for querying data, such as the `sum()` method that we will use now.

```
In [7]: # Importing the func model
from sqlalchemy import func

# Define the query in ORM
q = db.query(
    TLS206_PERSON.person_id,
    TLS206_PERSON.person_name, # inventor's name
    func.sum(TLS201_APPLN.nb_citing_docdb_fam).label('total_citations') # sum of families citing patents by a given inventor
).join(
    TLS207_PERS_APPLN, TLS201_APPLN.appln_id == TLS207_PERS_APPLN.appln_id
).join(
    TLS206_PERSON, TLS207_PERS_APPLN.person_id == TLS206_PERSON.person_id
).filter(
    TLS201_APPLN.appln_filing_year >= 2020,
    TLS201_APPLN.appln_auth == 'EP',
    TLS201_APPLN.granted == 'Y',
    TLS207_PERS_APPLN.invt_seq_nr > 0 # filter to include only inventors
).group_by(
    TLS206_PERSON.person_id,
    TLS206_PERSON.person_name
).order_by(
    func.sum(TLS201_APPLN.nb_citing_docdb_fam).desc() # order by total citations in descending order
)

# Creating a dataframe with the results
res = patstat.df(q)

res
```

Out[7]:

	person_id	person_name	total_citations
0	53448894	HARRIS, Jason L.	793
1	77725875	TIMM, Richard W.	604
2	74508759	BOUDREAUX, Chad P.	604
3	55712232	BAKOS, Gregory J.	604
4	54539129	SHELTON IV, Frederick E.	491
...
62694	83194481	STAMER, Martina	0
62695	74496483	SLAMA, Tahar	0
62696	79339667	Flint, Paul	0
62697	85867820	SCHMITT, Rainer	0
62698	85876809	NEGREL, Florence	0

62699 rows × 3 columns

In []: