The Patstat library - Lesson 1

This notebook is an introduction to the Patstat library in TIP. The European Patent Office's (EPO) PATSTAT (Patent Statistical Database) is a comprehensive resource designed for advanced statistical analysis of patent data. Launched in 2006, PATSTAT consolidates data from patent offices worldwide, facilitating in-depth research and trend analysis in the field of intellectual property. It includes detailed information on patent applications, applicants, inventors, and legal events, enabling users to perform complex queries and extract valuable insights into patenting activities and innovation trends. PATSTAT is widely used by policymakers, researchers, and businesses to inform decision-making and strategy development in technology and innovation sectors.

The PATSTAT Data Structure and Key Tables

PATSTAT is organized into a relational database structure. This design facilitates complex statistical analysis and research, structured across multiple interrelated tables, each containing specific categories of data. The data structure of patstat is rather complex, with two databases containing a multitude of interrelated tables. Below you have a list of some of the main tables that we will be using in this course.

The two PATSTAT databases

PATSTAT in TIP has access to two primary databases.

1. **PATSTAT Global**: This database is a comprehensive collection of worldwide patent data, ideal for broad statistical analysis and research. It includes detailed information on patent applications, filings, classifications, and the entities involved. See the full documentation here (https://link.epo.org/web/searching-for-patents/business/patstat/data-catalog-patsat-global-en.pdf).
2. **PATSTAT EP Register**: This database specifically focuses on the European Patent Register, providing detailed statistical analysis of EP applications and their status. It includes data on the legal status of patents, procedural details, and administrative events. See the full documentation here (https://link.epo.org/web/searching-for-patents/business/patstat/data-catalog-patsat-ep-register-en.pdf).

These databases support various joins and relationships between tables through primary and foreign keys, enabling users to derive detailed insights and conduct in-depth patent analysis efficiently.

Key Tables in PATSTAT

- **tls201_appln**: Contains basic information about patent applications, including application numbers, filing dates, and types of patents.
- **tls206_person**: Stores details about individuals and entities involved in patent applications, such as inventors and applicants, including their names and country codes.
- **tls207_pers_appln**: Acts as a link between applications and individuals, indicating which persons are associated with which applications.
- **tls209_appln_ipc**: Lists the International Patent Classification (IPC) codes assigned to applications, crucial for identifying the technical field of the inventions.
- **tls224_appln_cpc**: Contains Cooperative Patent Classification (CPC) codes, another classification system used to categorize patents.

This relational structure allows users to perform detailed queries and analyses. For instance, an application in the **tls201_appln** table can be linked to its inventors in the **tls207_pers_appln** and **tls206_person** tables, and to its technical classifications in the **tls209_appln_ipc** and **tls224_appln_cpc** tables.

Our first query with Object Relational mapping

The Patstat library in TIP allows you to access the Patstat databases with Object-Relational Mapping (ORM). In simple terms, ORM allows you to interact with a database using Python objects instead of writing raw SQL queries.

By using ORM, we abstract the underlying SQL queries, making the code cleaner, more maintainable, and easier to understand. This is particularly advantageous for complex queries involving multiple joins and filters, which are common when working with patent databases.

SQLAlchemy

PATSTAT client contains an implementation of SQLAlchemy,a well known Object-Relational Mapping (ORM) library for Python.

It is recommended that users familiarize themselves with SQLAlchemy. For more information, refer to the official SQLAlchemy documentation (https://www.sqlalchemy.org).

Step 1: Setup PATSTAT Client

To begin, you need to initialize the PATSTAT client. This client will be used to interact with the PATSTAT database.

In [1]:
```
from epo.tipdata.patstat import PatstatClient

# Initialize the PATSTAT client
patstat = PatstatClient()
```

Step 2: Instantiate ORM access

The Patstat library comes with an `orm()` method that allows us to access the databases, as described above.

In [3]:
```
# Access ORM
db = patstat.orm()

print (db)
```

```
<sqlalchemy.orm.session.Session object at 0x79f50dde5f50>
```

Step 3: Import Necessary Models

You need to import the necessary tables (models) that you want to query. For this example, we'll use just one table: `TLS201_APPLN` .

In [4]:
```
# Importing tables as models
from epo.tipdata.patstat.database.models import TLS201_APPLN
```

Step 4: launch the query

For our first query we want to get all the EP applications that were filed in 2010 and have been granted.

We are going to select specific fields `appln_id` , `appln_auth` , `appln_nr` , `appln_kind` , and `appln_filing_date` from the `TLS201_APPLN table` This is equivalent to a `SELECT` satement in SQL, and we do this with the `query()` method.

The filter method adds conditions to the query, retrieving only records where the application filing year is 2010, the application authority is 'EP' (European Patent Office), and the application is granted (granted is 'Y').

In [5]:
```
# Define the query
q = db.query(
    TLS201_APPLN.appln_id,
    TLS201_APPLN.appln_auth,
    TLS201_APPLN.appln_nr,
    TLS201_APPLN.appln_kind,
    TLS201_APPLN.appln_filing_date
).filter(
    TLS201_APPLN.appln_filing_year == 2010,
    TLS201_APPLN.appln_auth == 'EP',
    TLS201_APPLN.granted == 'Y'
)

# Execute the query and convert the result to a DataFrame
result_df = patstat.df(q)

# Display the results
result_df.head()
```

Out[5]:

	appln_id	appln_auth	appln_nr	appln_kind	appln_filing_date
0	317738099	EP	10007331	A	2010-07-15
1	273998248	EP	10150106	A	2010-01-05
2	274087970	EP	10150427	A	2010-01-11
3	275089977	EP	10156489	A	2010-03-15
4	315848659	EP	10165315	A	2010-06-09

Querying Patstat with SQL

You can also pass SQL queries to the Patstat library, with the same method `sql_query` that you can use with the EPAB library. In the example below we query Patstat using SQL, passing the same query we just did above: granted EP patents filed in 2010

In [15]:
```
res = patstat.sql_query("""
SELECT
    appln_id,
    appln_auth,
    appln_nr,
    appln_kind,
    appln_filing_date
FROM
    tls201_appln
WHERE
    appln_filing_year = 2010
    AND appln_auth = 'EP'
    AND granted = 'Y';

""")
## printing the first 5 results to compare with the dataframe
res[0:5]
```

Out[15]:
```
[{'appln_id': 274875721,
  'appln_auth': 'EP',
  'appln_nr': '10002082',
  'appln_kind': 'A ',
  'appln_filing_date': datetime.date(2010, 3, 1)},
 {'appln_id': 275066717,
  'appln_auth': 'EP',
  'appln_nr': '10002369',
  'appln_kind': 'A ',
  'appln_filing_date': datetime.date(2010, 3, 8)},
 {'appln_id': 275089809,
  'appln_auth': 'EP',
  'appln_nr': '10002521',
  'appln_kind': 'A ',
  'appln_filing_date': datetime.date(2010, 3, 10)},
 {'appln_id': 328601175,
  'appln_auth': 'EP',
  'appln_nr': '10014042',
  'appln_kind': 'A ',
  'appln_filing_date': datetime.date(2010, 10, 27)},
 {'appln_id': 274197226,
  'appln_auth': 'EP',
  'appln_nr': '10150900',
  'appln_kind': 'A ',
  'appln_filing_date': datetime.date(2010, 1, 15)}]
```

In [ ]: