Politechnika Poznańska
Wydział Informatyki i Telekomunikacji

Generowanie struktury bazy danych na podstawie taksonomii oraz rozładunek danych do tej bazy.

Autorzy:
Adam Korbik
Krystian Bryński
Marta Jóźwiak

05.2025r.

Spis treści

1 Cel projektu

Celem projektu jest zaprojektowanie i implementacja kompletnego systemu do automatycznego przetwarzania danych finansowych opartych na zewnetrznej strukturze taksonomii. System umożliwia:

- ekstrakcje informacji strukturalnych z plików XML opisujacych taksonomie,
- wygenerowanie oraz utrzymywanie relacyjnej struktury bazy danych odzwierciedlajacej istotne zależności,
- ekstrakcje i standaryzacje danych z raportów Excel zgodnie z definicja taksonomii,
- automatyczne ładowanie struktury oraz danych raportowych do bazy danych Microsoft SQL Server,
- integracje z systemem API, pozwalajacym na dostep do danych oraz ich dalsze przetwarzanie.

Projekt zakłada pełna automatyzacje przetwarzania — od momentu dostarczenia plików wejściowych (taksonomia i raporty) po zasilenie hurtowni danych zgodnej ze zdefiniowana struktura, co umożliwia analize danych zgodnie z jednolitym i kontrolowanym modelem informacyjnym.

2 Użyte technologie

- Biblioteki Python
  - **xml.etree.ElementTree** – Używana do przetwarzania plików XML. Umożliwia skuteczne i wydajne parsowanie oraz nawigacje po strukturze dokumentów XML.
  - **json** – W projekcie używamy tego formatu jako standardowego formatu transportowego do przechowywania danych wyodrebnionych z raportów oraz struktury opartej na taksonomii. Biblioteka JSON jest niezbedna do zapisu i odczytu tych danych.
  - **os** – Moduł wykorzystywany do operacji na systemie plików i zarzadzania ścieżkami plików. W projekcie służy do uzyskiwania docelowych ścieżek, łaczenia ich w sposób niezależny od systemu operacyjnego, a także do obsługi operacji takich jak tworzenie katalogów, sprawdzanie istnienia plików czy pobieranie nazw plików.
  - **dataclasses** – Używamy tego modułu, aby kod był bardziej czytelny i przejrzysty. To ułatwia zarzadzanie struktura danych. Ponadto, dataclasses wspieraja typowanie, co pomaga pisać bardziej bezbłedny kod.
  - **pandas** – W naszym projekcie służy przede wszystkim do łatwego parsowania i przetwarzania plików raportów w formacie Excel (.xls, .xlsx). Dzieki `pandas` możliwe jest szybkie odczytanie danych z arkuszy kalkulacyjnych, ich filtrowanie oraz przygotowanie do dalszego przetwarzania i eksportu do plików transportowych.

  - **pydobc** - Służy jako sterownik ODBC umożliwiajacy nawiazanie połaczenia z instancja serwera MSSQL. Biblioteka ta zapewnia niskopoziomowy interfejs do wykonywania zapytań SQL oraz komunikacji z baza danych poprzez protokół ODBC.
  - **SQLAlchemy** – Pełni funkcje warstwy abstrakcji nad silnikiem bazy danych. Umożliwia wykonywanie zapytań SQL, mapowanie danych do struktur Pythona z użyciem mechanizmu `bind parameters`, zarzadzanie sesja oraz transakcjami, a także obsługe błedów aplikacyjnych.

- **Microsoft SQL Server** - Pełni w projekcie funkcje centralnej relacyjnej bazy danych, w której gromadzone sa zarówno dane strukturalne wynikajace z definicji taksonomii, jak i dane raportowe pochodzace z plików wejściowych. Zapewnia integralności danych dzieki zastosowaniu kluczy głównych i obcych, które łacza tabele miedzy soba.
- **FastAPI** - Pełni role interfejsu dostepowego (API), umożliwiajacego komunikacje miedzy aplikacja a baza danych Microsoft SQL Server.
- **Git** - Systemu kontroli wersji.

3 Generowanie struktury na podstawie taksonomii

3.1 Ekstrakcja ścieżek kluczowych plików

Opracowaliśmy metode automatycznego identyfikowania i zapisywania ścieżek do kluczowych plików taksonomicznych, aby uogólnić i usprawnić nasze rozwiazanie. Naszym celem jest wyodrebnienie plików o określonych prefiksach: rend, lab-pl oraz lab-codes, ponieważ zawieraja one istotne informacje dotyczace struktury danych formularzy. W ramach taksonomii każdy arkusz posiada unikalny folder, np. n.ro.bk.00, w którym znajduja sie odpowiednie pliki źródłowe. Aby zwiekszyć przejrzystość i efektywność pracy, zdecydowaliśmy sie na automatyczne wyodrebnienie nazwy arkusza (czyli nazwy folderu) oraz trzech kluczowych ścieżek do plikow zawierajacych: rend, lab-pl i lab-codes dla każdego arkusza. Dzieki temu podejściu możemy w sposób systematyczny i zautomatyzowany analizować strukture taksonomii, co znaczaco ułatwia dalsze przetwarzanie danych. W kolejnych podpunktach szczegółowo opiszemy zawartość plików rend, lab-pl oraz lab-codes, a także dokładnie określimy, jakie informacje sa z nich wyciagane i w jaki sposób sa wykorzystywane.

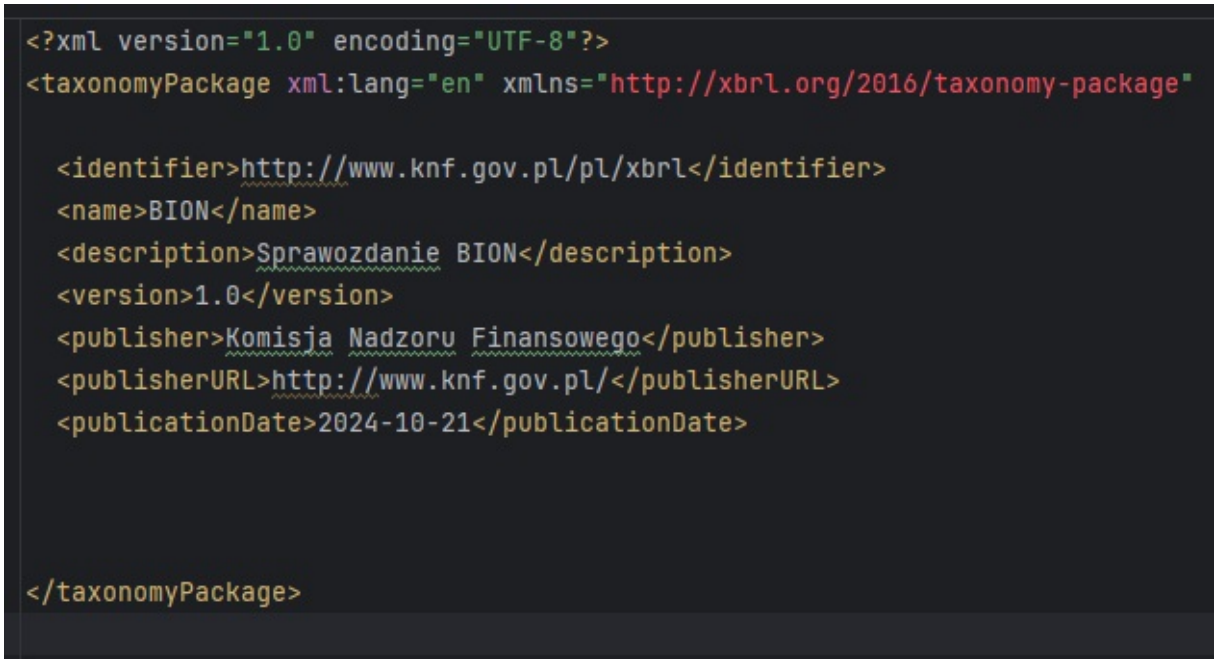
Funkcja o nazwie `collect_target_paths_and_sheet_name` została zaprojektowana tak, aby automatycznie przeszukiwać katalog z taksonomia, identyfikować odpowiednie pliki oraz zwracać liste zawierajaca krotki z kluczowymi informacjami. Każda zwrócona krotka składa sie z dwóch elementów: Listy ścieżek do trzech plików (rend, lab-pl, lab-codes), które zawieraja istotne informacje o strukturze danych i nazwy arkusza, odpowiadajacej nazwie folderu, w którym pliki sie znajduja.

3.2 Ekstrakcja nazwy oraz wersji taksonomii

Aby zachować pełna strukture danych w bazie, niezbedne jest uwzglednienie informacji o nazwie oraz wersji taksonomii. Jest to kluczowa informacja, ponieważ przed dodaniem nowej struktury konieczne jest sprawdzenie, czy dana taksonomia już istnieje w bazie. Dzieki

temu unikamy duplikacji danych i zapewniamy spójność przechowywanych informacji. Weryfikacja wersji taksonomii pozwala na kontrole aktualizacji oraz integracje nowych danych bez ryzyka nadpisania istniejacych wpisów.

Te infroamcje możemy znaleźć w katalogu META-IFNO w pliku taxonomyPackage.xml z którego można pobrac nazwe oraz wersje taksonomi.



Rysunek 1: Przykładowa zawartość pliku taxonomyPackage.xml

W celu zapewnienia poprawnej identyfikacji struktury danych, został opracowany skrypt, który automatycznie wyodrebnia nazwe oraz wersje taksonomii. Funkcja realizujaca to zadanie nosi nazwe `collect_name_and_version_taxonomy` i została zaprojektowana tak, aby skutecznie przeszukiwać plik w celu odnalezienia tych kluczowych informacji.

Po zakończeniu analizy funkcja zapisuje pobrane dane jako plik transportowy JSON w formacie obiektu:

```
{
    "name": "name_tax",
    "version": "version_tax"
}
```

Dzieki temu podejściu możliwe jest przechowywanie informacji o nazwie i wersji taksonomii w czytelny, ustandaryzowany sposób. Jeśli w tabeli naszej bazy danych brakuje wpisu dotyczacego konkretnej nazwy i wersji taksonomii, dane z pliku transportowego JSON zostana załadowane, obejmujac zarówno nazwe i wersje taksonomii, jak i cała jej strukture – pod warunkiem, że nie istnieje już wpis dla tej taksonomii. Szczegóły dotyczace procesu ładowania oraz integracji struktury danych zostana opisane w kolejnych punktach.

3.3 Ekstrakcja etykiet wraz z przypisanym tekstem

W plikach XML zawierajacych tzw. „lab-pl" znajduja sie etykiety wraz z przypisanymi do nich tekstami. Na podstawie tych plików jesteśmy w stanie określić, jaki tekst odpowiada poszczególnym etykietom. Operacja ta jest niezbedna do prawidłowego odwzorowania struktury danych.

```
<link:loc xlink:type="locator" xlink:href="n.zb.bk.02.01-rend.xml#uknf_c19" xlink:label="loc_uknf_c19" />
<label:label xlink:type="resource" xlink:label="label_uknf_c19" xml:lang="pl" xlink:role="http://www.xbrl.org/2008/role/label">inne</label:label>
<gen:arc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/element-label" xlink:from="loc_uknf_c19" xlink:to="label_uknf_c19" />
```

Rysunek 2: Fragment pliku zawierajacego „lab-pl"

Na Rysunku 2 przedstawiono fragment pliku XML. Jak widać, z tego pliku należy wyodrebnić etykiete wraz z przypisanym jej tekstem. Docelowo interesuje nas wydobycie pary, np. `uknf_c19: Inne`.

W tym celu została zaimplementowana funkcja `extract_lab_pl_labels_and_values`, która zwraca liste wszystkich znalezionych par etykieta–tekst, zgodnie z opisanym powyżej formatem.

3.4 Ekstrakcja etykiet wraz z przyporzadkowanymi punktami danych

W plikach XML zawierajacych tzw. „lab-codes" znajduja sie etykiety wraz z przypisanymi do nich punktami danych. Korzystajac z tych plików, dla każdego arkusza jesteśmy w stanie wyodrebnić etykiety wraz z odpowiadajacym jej punktami danych. Proces ten jest niezbedny, aby sklasyfikować etykiete wraz z przypisanym punktem danych i utworzyć poprawna strukture danych.

```
<link:loc xlink:type="locator" xlink:href="n.pif.bk.00-rend.xml#uknf_c5" xlink:label="loc_uknf_c5" />
<label:label xlink:type="resource" xlink:label="label_uknf_c5" xml:lang="en" xlink:role="http://www.eurofiling.info/xbrl/role/rc-code">0040</label:label>
<gen:arc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/element-label" xlink:from="loc_uknf_c5" xlink:to="label_uknf_c5" />
```

Rysunek 3: Fragment pliku zawierajacego „lab-codes"

Na Rysunku 3 przedstawiono fragment pliku XML, który można sparsować w celu ekstrakcji etykiet wraz z przypisanymi do nich wartościami, czyli punktami danych, na przykład: `uknf_c5` odpowiada wartości `0040`. Taka ekstrakcja jest kluczowa do dalszego przetwarzania danych. W naszym skrypcie została zaimplementowana funkcja `extract_lab_codes_label` która przyjmuje na wejściu sparsowany plik XML. Funkcja ta służy do wyodrebnienia powiazań miedzy etykietami a odpowiadajacymi im punktami danych. Dzieki temu możliwe jest dokładne określenie, jaki konkretny punkt danych jest przypisany do której etykiety. W praktyce funkcja przeszukuje elementy XML o nazwie `label`, pobiera ich atrybuty identyfikujace oraz tekst zawierajacy wartość, a nastepnie zwraca liste par (etykieta, punkt danych). Do dalszego przetwarzania potrzebujemy identyfikatorów etykiet w formacie np. `uknf_c50`, a nie z prefiksem `label_uknf_c50`. Dlatego w funkcji zastosowano mechanizm filtrowania, który usuwa prefiks `label_` z nazwy etykiety. Takie podejście zapewnia spójność identyfikatorów etykiet w całym procesie przetwarzania oraz umożliwia poprawne i jednolite mapowanie etykiet na odpowiadajace im punkty danych. Dzieki temu unikamy problemów wynikajacych z różnic w nazwach miedzy różnymi plikami XML i upraszczamy dalsza analize danych.

3.5 Ekstrakcja etykiet wraz z przypisaniem do osi

Ważnym etapem przetwarzania danych jest przypisanie poszczególnych etykiet do odpowiednich osi `x` lub `y`. W tym celu wykorzystywany jest plik XML zawierajacy w nazwie człon „rend", z którego możliwe jest odczytanie kolejności wystepowania etykiet oraz znaczników osi.

Dla arkusza dwuwymiarowego przykładowa sekwencja może wygladać nastepujaco:

```
[uknf_c1, uknf_c2, uknf_c3, x, uknf_c4, y]
```

Na podstawie takiej listy etykiety pojawiajace sie przed pierwszym wystapieniem osi `x` sa przypisywane do osi `x`, natomiast wszystkie etykiety po tym znaczniku klasyfikowane sa jako należace do osi `y`.

W przypadku arkusza jednowymiarowego sekwencja może przyjać forme:

```
[uknf_c1, uknf_c2, uknf_c3, x]
```

Wówczas wszystkie etykiety poprzedzajace wystapienie znacznika osi `x` przypisuje sie do tej właśnie osi.

Taka klasyfikacja umożliwia prawidłowe odwzorowanie struktury arkuszy w dalszych etapach przetwarzania danych.

Funkcja `extract_rend_ordered_labels_and_axes` odpowiada za odczyt kolejności wystepowania etykiet oraz osi na podstawie struktury pliku XML zawierajacego człon „rend". Jej zadaniem jest wygenerowanie listy identyfikatorów, w której zachowana jest kolejność pojawiania sie zarówno etykiet, jak i znaczników osi.

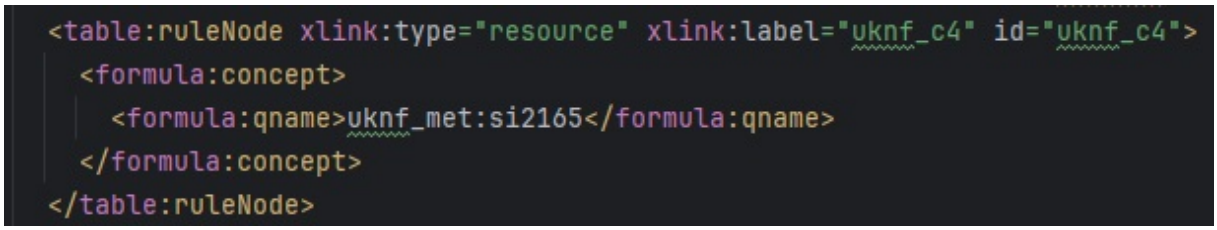
Funkcja działa w nastepujacy sposób:

- przeszukuje elementy XML w poszukiwaniu tagów `ruleNode`, z których odczytuje atrybut `id` — identyfikator etykiety,
- identyfikuje elementy `tableBreakdownArc`, z których pobierany jest atrybut `axis` — reprezentujacy oś (np. `x`, `y`),
- na końcu stosowane jest filtrowanie, które usuwa elementy pomocnicze, tj. identyfikatory rozpoczynajace sie od prefiksu `uknf_a`, ponieważ nie sa one istotne dla przypisania etykiet do osi.

Tak przygotowana lista pozwala na jednoznaczne przypisanie każdej etykiety do odpowiedniej osi, zgodnie z jej kolejnościa wystepowania w strukturze dokumentu XML. To podejście jest kluczowe dla dalszego przetwarzania danych i poprawnego odwzorowania struktury arkusza.

3.6 Ekstrakcja etykiet wraz z nazwa metryki

W pliku XML zawierajacym człon „rend" można znaleźć informacje o powiazaniach pomiedzy etykietami a odpowiadajacymi im metrykami. Informacja ta jest kluczowa, ponieważ umożliwia zachowanie spójności typów danych w strukturze danych — każda etykieta jest powiazana z metryka, która definiuje jej typ danych.



Rysunek 4: Fragment pliku zawierajacego „rend”

Jak przedstawiono na Rysunku 4, możliwe jest odczytanie z pliku nazwy metryki (atrybut `qname`), która została przypisana do konkretnej etykiety. W zaprezentowanym przykładzie etykieta `uknf_c4` jest powiazana z metryka `uknf_met:si2165`. Tego rodzaju odwzorowanie stanowi istotny krok w procesie budowania kompletnej struktury danych, ponieważ pozwala na późniejsze przypisanie odpowiednich typów danych na podstawie informacji z pliku `met.xsd`.

W naszym skrypcie funkcja `extract_rend_labels_and_qnames` odpowiada za realizacje tej operacji. Przetwarza odpowiedni plik XML i zwraca liste krotek w formacie: `(etykieta, nazwa metryki)`. Dzieki temu możliwe jest jednoznaczne powiazanie każdej etykiety z odpowiadajaca jej metryka, co stanowi podstawe do przypisania właściwego typu danych w dalszych etapach przetwarzania.

3.7 Ekstrakcja typów danych

W katalogu `dict/met` znajduje sie plik `met.xsd`, który odpowiada za klasyfikacje nazw metryk według typów danych. Plik ten jest niezbedny do zapewnienia poprawnego typowania danych w dalszym etapie przetwarzania. Na podstawie tego pliku ekstraktowane sa wszystkie dostepne metryki wraz z przypisanymi do nich typami danych. Każda etykieta w strukturze danych jest powiazana z konkretna metryka, dlatego wyodrebnienie tych informacji stanowi kluczowy element budowy spójnej struktury danych. Dzieki pozyskanym z pliku `met.xsd` informacjom możliwe jest przypisanie odpowiedniego typu danych do każdej etykiety, poprzez odniesienie sie do metryki, z która dana etykieta jest powiazana. Pozwala to zachować integralność typów w całym zbiorze danych oraz umożliwia poprawna interpretacje wartości wystepujacych w poszczególnych polach.

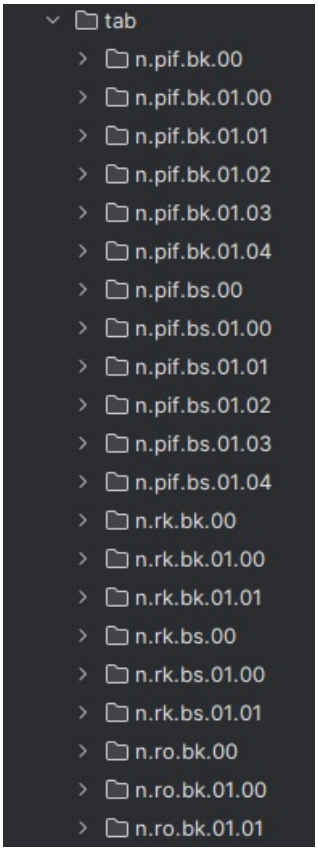
Rysunek 5: Fragment pliku met.xsd

W celu pozyskania metryk wraz z odpowiadajacymi im typami danych została zaimplementowana funkcja `extract_types_and_names`. Funkcja ta przyjmuje jako argument

sparsowany plik `met.xsd`, a nastepnie przeszukuje wszystkie elementy typu `element`. Dla każdego z nich wyodrebnia atrybuty `name` oraz `type`, które reprezentuja odpowiednio nazwe metryki oraz przypisany jej typ danych. Wynikiem działania funkcji jest lista par `(nazwa, typ)`, która stanowi podstawe do dalszego przypisywania typów danych poszczególnym etykietom w strukturze danych. Takie podejście pozwala zachować spójność i poprawność typowania w całym procesie przetwarzania raportów.

3.8 Ekstrakcja unikalnych nazw formularzy

W naszej strukturze danych niezbedne jest pozyskanie unikalnych nazw formularzy wystepujacych w taksonomii. W tym celu wykorzystujemy zawartość katalogu o nazwie `'tab'`, który zawiera podkatalogi odpowiadajace poszczególnym arkuszom formularzy, np. `n.pif.bk.00`, `n.pif.bs.01.00` itp. Każdy z tych folderów zawiera pliki opisujace strukture pojedynczego arkusza danego formularza. Aby uzyskać zestaw unikalnych nazw formularzy, analizujemy wszystkie nazwy folderów w katalogu `tab` i redukujemy je do postaci składajacej sie wyłacznie z pierwszych trzech segmentów oddzielonych kropkami. Przykładowo, z nazwy `n.pif.bk.00` uzyskujemy `n.pif.bk`, a z `n.pif.bs.01.00` — `n.pif.bs`. W ten sposób otrzymujemy zestaw jednoznacznych identyfikatorów formularzy wykorzystywanych w taksonomii, które moga być dalej wykorzystywane w procesach analizy i odwzorowywania danych.



Rysunek 6: Przykład zawartości katalogu tab

W tym celu została zaimplementowana funkcja `collect_unique_form_names`, która przeszukuje katalog `tab`, analizuje nazwy podkatalogów i zwraca liste unikalnych nazw

formularzy. Funkcja ta usuwa z nazw folderów wszelkie segmenty wystepujace po trzecim znaku kropki, co pozwala na uzyskanie jednoznacznych identyfikatorów formularzy. Dzieki temu mechanizmowi możliwe jest efektywne zarzadzanie oraz dalsze przetwarzanie danych zwiazanych z poszczególnymi formularzami, zapewniajac spójność i unikalność ich identyfikatorów w całej strukturze taksonomii.

3.9 Integracja wyekstrahowanych danych

W celu wstepnego połaczenia danych została opracowana funkcja `combine_data_from_files`, której zadaniem jest określenie, czy dany formularz jest jednoosiowy czy dwuosiowy. Na podstawie tej klasyfikacji proces przetwarzania danych odbywa sie poprzez wywołanie odpowiedniej funkcji: `combine_one_dimensional_data` dla formularzy jednoosiowych oraz `combine_two_dimensional_data` dla formularzy dwuosiowych.

Funkcja `combine_one_dimensional_data` generuje strukture w formie słownika, w którym kluczami sa etykiety odnoszace sie do wierszy lub kolumn (w zależności od arkusza), a wartościami tekst powiazany z etykieta, przypisane punkty danych oraz informacja o osi, w której znajduje sie dany klucz. Informacja o osi bedzie wykorzystywana w dalszych etapach analizy do określenia orientacji arkusza (pionowej lub poziomej).

Funkcja `combine_two_dimensional_data` tworzy strukture słownikowa, w której kluczami sa etykiety wierszy, a wartościami tekst przypisany do wiersza, zintegrowane punkty danych oraz uporzadkowane teksty kolumnowe odpowiadajace kolejności punktów danych. Proces integracji punktów danych obejmuje krzyżowanie wartości, polegajace na połaczeniu wszystkich punktów przypisanych do danej etykiety, np. `0010X0010, 0010X0020`. Dzieki temu możliwa jest pełna synchronizacja kolumn i wierszy, co usprawnia dalsza analize struktury arkusza.

Do tej struktury danych dodawany jest również typ danych oraz nazwa metryki. Funkcja `match_labels_with_data_types` tworzy mape etykiet, przypisujac im odpowiednie typy danych oraz nazwy metryk. Dzieki temu możliwe jest późniejsze wykorzystanie funkcji `match_datatypes_and_qnames`, która przyjmuje wcześniej zdefiniowana strukture słownika. W jej działaniu nastepuje dopasowanie typu danych oraz nazwy metryki do każdego klucza w słowniku na podstawie przypisanych etykiet.

Możliwa jest sytuacja, w której funkcja nie przypisze żadnej nazwy metryki ani typu danych. Może to wynikać z braku odpowiedniego wpisu dla danej etykiety w pliku `met.xsd`, który zawiera wszystkie dostepne typy danych wraz z nazwami metryk.

W przypadku braku dopasowania obowiazuje domyślna konfiguracja, w której: `typ danych` przyjmuje wartość `"xbrli:stringItemType"`, `nazwa metryki` zostaje ustawione na `"None"`.

Do finalnej struktury bedzie jezcze dodana nazwa arkusza i nazwa formularza

3.10 Proces tworzenia finalnej struktury dla jednego arkusza

Jako plik transportowy wybrano format JSON, który zapewnia spójność i czytelność danych oraz umożliwia ich bezproblemowe załadowanie do bazy danych. W takim formacie finalna struktura bedzie przechowywana, co pozwoli na jej łatwa integracje i dalsze przetwarzanie.

Funkcja `transform_data` przyjmuje na wejściu słownik zawierajacy wiekszość informacji opisanych w punkcie 3.9, a nastepnie generuje finalna strukture JSON, dostosowana do wymagań systemu.

Finalny obiekt JSON przyjmuje nastepujaca postać:

```
{
    "form_name": "example_form",
    "data": {
        "label_1": {
            "value_row": "Example row value",
            "data_points": ["0010X0010", "0010X0020"],
            "value_columns": ["Example column 1", "Example column 2"],
            "datatype": "xbrli:stringItemType",
            "qname": "None",
            "sheet_name": "Sheet1"
        },
        "label_2": {
            "value_row": "Another row value",
            "data_points": ["0020X0030"],
            "value_columns": "Single column value",
            "datatype": "xbrli:monetaryItemType",
            "qname": "SomeQName",
            "sheet_name": "Sheet1"
        }
    }
}
```

W ten sposób dla pojedynczego arkusza został utworzony plik transportowy, w którym głównym kluczem jest nazwa formularza. Wszystkie zwiazane z nim dane sa przechowywane wewnatrz tej struktury, co pozwala na jednoznaczna identyfikacje formularza oraz jego zawartości. Taki sposób organizacji pliku zapewnia spójność i czytelność.

3.11 Proces automatyzacji

W celu zautomatyzowania procesu została opracowana funkcja `create_structure`, której głównym zadaniem jest iteracja przez wszystkie unikalne nazwy formularzy, identyfikacja powiazanych arkuszy wraz z plikami docelowymi (proces ich ekstrakcji i opis znajduje sie w punkcie 3.1). Nastepnie każda nazwa formularza oraz arkusza, wraz z odpowiadajacymi im plikami docelowymi, jest przekazywana do funkcji `build_json_for_sheet`. Funkcja ta odpowiada za integracje wszystkich modułów, utworzenie kompletnej struktury w formacie JSON dla danego arkusza oraz jej zapis do dedykowanego folderu.

Po zakończeniu głównej iteracji w funkcji `create_structure` generowana jest pełna struktura obejmujaca wszystkie arkusze, zapisana w formatach JSON. Liczba utworzonych plików transportowych jest równa liczbie arkuszy wystepujacych w obrebie formularzy, co zapewnia przejrzystość danych oraz ich jednoznaczna identyfikacje. Struktura ta pozwala na łatwe przechowywanie i dalsze przetwarzanie informacji w kontekście analizy taksonomicznej.

4 Struktura tabel i relacji w bazie danych

4.1 Opis struktury bazy danych

Diagram obrazuje docelowa strukture relacyjnej bazy danych, złożona z sześciu kluczowych tabel: `Reports`, `Data`, `Data_points`, `Labels`, `Taxonomy` oraz `Forms`. Każda tabela posiada własny klucz główny (PK), zaś relacje pomiedzy nimi realizowane sa poprzez odpowiadajace im klucze obce (FK). W praktyce:

Tabela `Taxonomy` definiuje kategorie i ich wersje, a tabela `Forms` odwołuje sie do niej za pomoca klucza obcego `id_taxonomy`, wskazujac, do której taksonomii należy dany formularz.

Formularze w tabeli `Forms` składaja sie z wielu etykiet przechowywanych w tabeli `Labels`, gdzie kolumna `id_form` stanowi klucz obcy łaczacy etykiety z odpowiednim formularzem.

Każda etykieta w tabeli `Labels` zawiera zestaw punktów danych zdefiniowanych w tabeli `Data_points`, które sa powiazane poprzez kolumne `id_label` (FK).

Tabela `Data` gromadzi dane ze raportów, przy czym każdy rekord wskazuje, do którego punktu danych (`id_data_point`) oraz do którego raportu (`id_report`) należy (obie kolumny sa kluczami obcymi).

Tabela `Reports` pełni role nadrzednej kolekcji zestawów danych z raportó, agregujac wszystkie rekordy z tabeli `Data`.

5 Ekstrakcja danych z raportu

Celem procesu ekstrakcji jest automatyczne wydobywanie danych z tabel zawartych w raportach finansowych zgodnych z Taksonomia BION. W trakcie działania skryptu pozyskiwane sa nazwy arkuszy, kody pozycji (datapointy), a nastepnie z tych pozycji pobierane sa odpowiadajace im dane. Proces obsługuje trzy różne układy tabel i zapisuje wyodrebnione dane w formacie JSON, aby umożliwić dalszy ładunek do bazy danych.

5.1 Struktura wejściowa

Skrypt przyjmuje jako dane wejściowe pliki Excel (*.xlsx, *.xls) zawierajace raporty. Pliki te maja ustandaryzowana strukture wynikajaca z wytycznych Taksonomii BION:

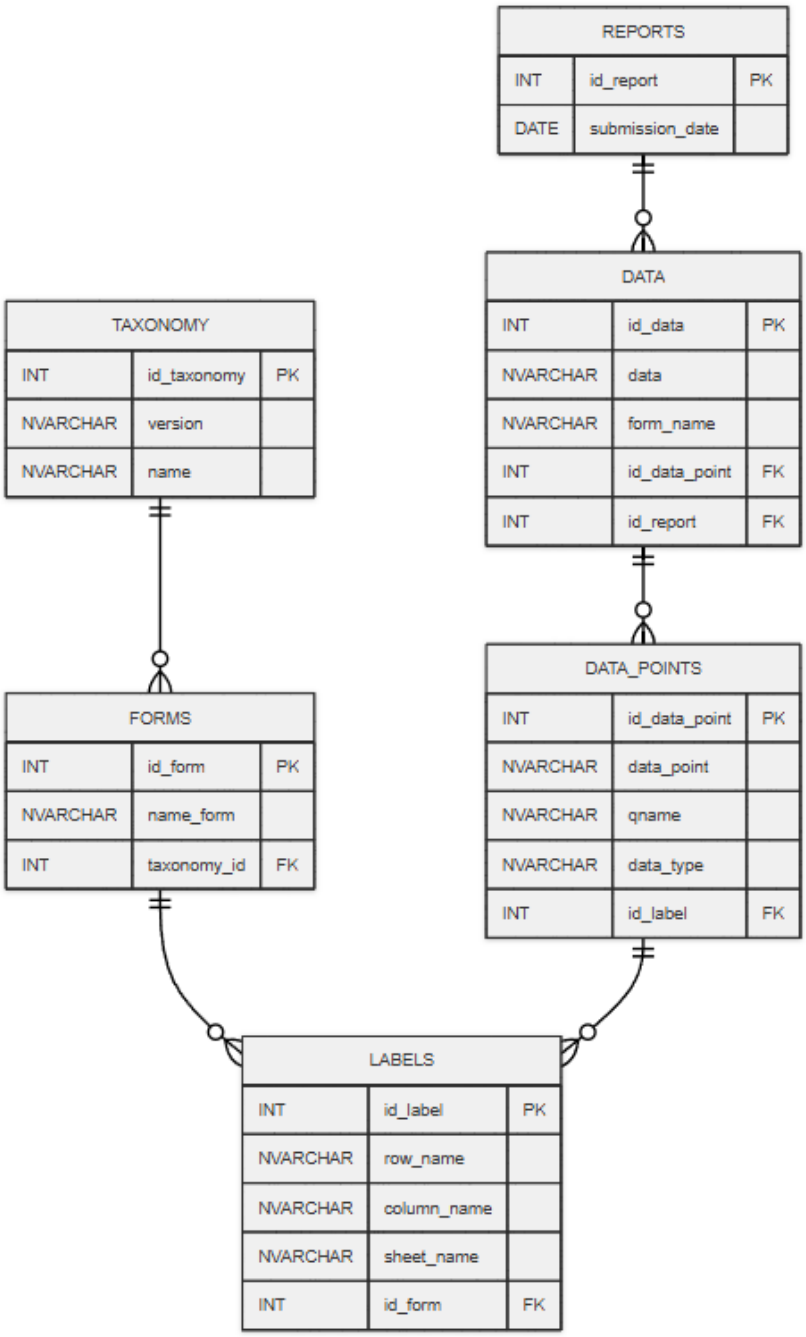
- Każdy raport zawiera wiele arkuszy
- Każda wartość finansowa reprezentowana jest przez datapoint/y
- Dane tabelaryczne moga być zapisane w układach: horyzontalny, poziomy, mieszany.

Poniżej przedstawione sa przykładowe obrazy trzech układów danych tabelarycznych z zaznaczonymi na czerwono wartościami do ekstrakcji. [zdj]

5.2 Opis procesu ekstrakcji danych z raportu

Z założeń wynikajacych w pudpunkcie 4.1. procesem ekstrakcji zostały objete poniższe dane:

- `nazwy arkuszy`



Rysunek 7: Struktura bazy danych

- `kody pozycji (datapoint'y)`
- `dane finansowe`

Dane finansowe zostały wyekstraktowane w odniesieniu do współrzednych kodów pozycji. Poniżej znajduja sie szczegółowe opisy etapów algorytmu ekstrakcji.

5.2.1 Wyodrebnienie nazwy arkusza

Pierwszym krokiem jest pozyskanie wszystkich nazw arkuszy z danego pliku Excel.

5.2.2 Usuniecie kolumn pomocniczych

W nastepnym kroku nastepuje usuniecie z arkusza z kolumn pomocniczych, które wystepuja w arkuszach nieregularnie i utrudniaja proces ekstrakcji.

5.2.3 Ekstrakcja kodów pozycji

Trzy różne układy danych tabelarycznych oznaczaja trzy różne ułożenia kodów pomocniczych (tzw. datapoint'ów). Poczatek procesu ich ekstrakcji rozpoczyna sie wiec od zidentyfikowania, który z trzech układ jest ekstraktowany.

Za identyfikacje odpowiada funkcja `find_sequence_positions`, a jej działanie polega najpierw na odnalezieniu datapoint'ow "0010"oraz sprawdzenia ich liczby. Jedna komórka "0010"oznacza układ horyzontalny lub wertykalny, a dwie komórki "0010"oznaczaja układ mieszany. Nastepnie algorytm odnajduje reszte sekwencji datapoint'ów, w poniższy sposób:

- W przypadku jednej komórki "0010"algorytm porusza sie w dół i w prawo szukajac kontynuacji datapoint'ów oraz dodatkowo pomija wartości puste. Jeżeli kontynuacja odnaleziona zostanie w prawym kierunku oznacza to, że układ jest horyzontalny, a jeżeli w dolnym kierunku układ jest wertykalny. Współrzedne odnalezionych datapoint'ów zostaja wiec zwracane przez funkcje w odpowiednich listach: `positions_horizontal` lub `positions_vertical`
- W przypadku dwóch komórek "0010"algorytm porównuje numer wierszy we współrzednych komórkach. Z rozmieszczenia datapointów we wszystkich arkuszach można założyć, że komórka z niższym numerem wiersza rozpoczyna sekwencje horyzntalna, a komórka z wyższym numerem sewkencje wertykalna, Dalej algorytm wyszukuje kontynuacje datapoint'ów z pominieciem wartości pustych (odpowiednio w dół lub w prawo) i zwraca ich współrzedne do obu list `positions_horizontal` i `positions_vertical` .

5.2.4 Ekstrakcja danych finansowych

W zależności jakie funkcja `find_sequence_positions` zwraca współrzedne, algorytm wybiera jedna z trzech strategi ekstrakcji:

- Układ mieszany: oba zbiory datapoint'ów sa obecne → `extract_intersections` Funkcja przeszukuje dane znajdujace sie na skrzyżowaniu datapointów zidentyfikowanych zarówno w kolumnach, jak i w wierszach. Dla każdej niepustej komórki w tej siatce tworzony jest wpis w formacie:

`{datapoint_poziomy}x{datapoint_pionowy}: wartość`

- Układ horyzntalny: tylko datapoint'y w poziomie → `extract_from_horizontal` Funkcja stosowana w przypadku, gdy datapointy wystepuja w górnym wierszu (w poziomie). Każda kolumna traktowana jest jako osobny datapoint, a dane znajduja sie bezpośrednio pod nim. Dla każdej niepustej komórki, tworzony jest wpis w formacie:

  `{datapoint_poziomy}: wartość`

- Układ wertykalny: tylko Datapointy w pionie → `extract_from_vertical` Funkcja stosowana w przypadku, gdy Datapointy wystepuja w kolumnie po lewej stronie (w pionie). Każdy wiersz traktowany jest jako osobny Datapoint, a dane znajduja sie po jego prawej stronie. Analogicznie dla każdej niepustej komórki, tworzony jest wpis w formacie:

  `{datapoint_pionowy}: wartość`

Każdy wyodrebniony element jest reprezentowany jako para klucz-wartość.

5.2.5 Zapis danych z raportu do formatu JSON

Dla każdego arkusza generowany jest jeden plik transportowy w formacie JSON, zawierajacy dane wyodrebnione z tabel. Struktura pliku przyjmuje postać:

```
{
    "sheet_name": [
        {
            "datapointA x datapointB": wartość
        },
        {
            "datapointC x datapointD": wartość
        },
        ...
    ]
}
```

- `sheet_name` – nazwa arkusza kalkulacyjnego, z którego pochodzi zestaw danych.
- `datapointA x datapointB` - klucz reprezentuje kontekst danej wartości.
- `data` – wartość odpowiadajaca danemu kluczowi.

f

6 Automatyczny proces ładowania danych do bazy MSSQL

Proces ładowania danych do relacyjnej bazy danych MSSQL został podzielony na dwa główne etapy:

- załadunek struktury taksonomii,
- załadunek danych z wcześniej wyekstrahowanych plików transportowych raportów.

Całość realizowana jest przez aplikacje FastAPI udostepniajaca interfejsy dostepowe (API) do inicjalizacji każdego z tych etapów.

6.1 Połaczenie z MSSQL za pomoca SQLAlchemy

Do komunikacji z baza danych wykorzystywana jest biblioteka SQLAlchemy. Połaczenie realizowane jest za pomoca sterownika ODBC oraz mechanizmu uwierzytelniania Windows. Każda operacja wykonywana jest wewnatrz transakcji, przy użyciu konstrukcji `engine.begin()`.

6.2 Utworzenie tabel w MSSQL

Po połaczeniu z MSSQL funkcja `create_tables` tworzy puste tabele (4.1) jeśli nie istnieja.

6.3 Załadunek struktury taksonomii

Struktura taksonomii jest ładowana z dwóch rodzaji plików transportowych. Jeden rodzajów plików zawiera nazwe i wersje taksonomi (3.2). Jeden plik transportowy odnosi sie do jednego arkusza (formularza).

6.3.1 Logika załadunku struktury

Proces załadunku struktury taksonomii do wcześniej utworzonych tabel `Taxonomy`, `Datapoints`, `Labels`, `Forms` realizuje nastepujaca logike:

1. Odczytywany jest plik transportowy z nazwa i wersja taksonomii. W bazie danych w tabeli `Taxonomy` nastepuje sprawdzenie czy ładowana wersja taksonomii znajduje sie już w bazie danych. Jeżeli tak - program przerywa działanie procesu. Jeżeli nie - do tabeli wstawiany jest nowy wpis odpowiadajacy tej wersji i kontyowany jest proces ładowania.
2. Dla każdej wartości `form_name` zdefiniowanej w strukturze:
   (a) Tworzony jest rekord w tabeli `Forms`, zawierajacy nazwe formularza (`name_form`) oraz powiazany identyfikator taksonomii (`id_taxonomy`),
   (b) Nastepnie, dla każdego elementu opisanego w polu `data`:
   - Do tabeli `Labels` dodawany jest rekord zawierajacy:
     - nazwa wiersza (`row_name`),
     - nazwa kolumny (`column_name`),

- nazwa arkusza (`sheet_name`),
- powiazanie z identyfikatorem formularza (`id_form`).

[zdj]

- Nastepnie dla każdego elementu listy `data_points` tworzony jest rekord w tabeli `Data_points`, zawierajacy:
  - identyfikator datapoint'u (`id_data_pint`),
  - datapoint (`data_point`),
  - qname (`qname`),
  - typ danych (`data_type`),
  - klucz obcy wskazujacy na odpowiadajacy rekord z tabeli `Labels` (id_label)

[zdj]

6.4 Załadunek danych z raportów

Proces ładowania danych z raportów nastepuje z wcześniej wyekstrahowanych plików transportowych (5.2.5). Jeden plik JSON zawiera dane w kontekście konkretnego formularza (arkusza Excel) (`sheet_name`).

6.4.1 Logika załadunku danych z raportów

Proces załadunku do tabel wcześniej utworzonych tabel `Data` i `Reports` obejmuje:

1. Wstawienie nowego rekordu do tabeli `Reports`, zawierajacego znacznik czasu rozpoczecia załadunku (domyślnie `GETDATE()`).
2. Dla każdego formularza (`form_name`) wystepujacego w pliku danych raportowych:
   (a) Na podstawie nazwy formularza pobierane sa wszystkie punkty danych (`data_point`) z tabeli `Data_points`, dla których odpowiadajacy arkusz (`sheet_name`) odpowiada danemu formularzowi.
   (b) Dla każdej pary `data_point`: `wartość` znajdujacej sie w pliku JSON:
   - Wykonywana jest walidacja – sprawdzane jest, czy `data_point` znajduje sie w strukturze taksonomii załadowanej do bazy danych. Jeżeli nie – proces zostaje przerwany i zgłaszany jest bład spójności.
   - Jeżeli `data_point` jest poprawny, tworzony jest rekord w tabeli `Data`, zawierajacy:
     - identyfikator raportu (`id_report`),
     - identyfikator punktu danych (`id_data_point`),
     - nazwe formularza (`form_name`),
     - wartość (`data`).

6.5 Integracja z interfejsem dostepowym (API)

...

7 Obsługa programu

7.1 Instalacja bibliotek

Zainstaluj wymagane zależności i biblioteki za pomoca poniższego polecenia:

```
pip install -r .\requirements.txt
```

7.2 Tworzenie katalogów

Aby stworzyć potrzebne katalogi należy uruchomic skrypt `src/main.py`. Skrypt ten automatycznie utworzy wszystkie niezbedne katalogi. W terminalu skorzystaj z polecenia:

```
python src/main.pya
```

7.3 Użycie programu

Po uruchomieniu programu `src/main.py` w terminalu wyświetli sie menu dla użytkownika, a zaraz po nim prośba o wybranie numeru akcjii.

```
=== MENU PROGRAMU ===
       Wybierz jedna z opcji:

       1 - Tworzenie struktury bazodanowej na podstawie taksonomii
       2 - Ekstrakcja danych z raportu
       3 - Ladowanie struktury bazodanowej do bazy danych
       4 - Ladowanie danych z raportu do bazy danych
       0 - Wyjście z programu

       ====================
```

7.3.1 Tworzenie struktury bazodanowej na podstawie taksonomii.

Do powstałego katalogu `data/taxonomy` należy wrzucić model taksonomii, przed wybraniem akcjii.

Input

```
KNF/
data/
   taxonomy/
       Taxonomy
```

Po wrzuceniu modelu taksonomii można w terminalu wpisać wartość 1, aby uruchomić skrypt realizujacy to zadanie.

Output

```
KNF/
structure/
   full_structure/
      form_name_1.json
      form_name_2.json
```

```
      form_name_3.json
   taxonomy_info/
      taxonomy_info.json
```

Skrypt, na podstawie Twojej taksonomii, wygeneruje pliki transportowe. Plików transportowych (zawierajacych strukture dla danego arkusza) bedzie tyle ile arkuszy znajduje sie we wszystkich formularzach. Każdy plik zostanie zapisany w formacie JSON i nazwany zgodnie z nazwa odpowiadajacego mu arkusza. Skrypt również zapisze plik `taxonomy info.json` do katalogu `taxonomy info` wraz z nazwa oraz wersja taksonomii.

7.3.2 Ekstrakcja danych z raportu.

Do katalogu `data/reports` należy wrzucić jeden raport w formacie `.xls` lub `.xlsx` z uzupełnionymi danymi, przed wykonaniem akcjii.

Input

```
KNF/
data/
   reports/
       report.xlsx
```

Po wrzuceniu raportu można w terminalu wpisać wartość 2, aby uruchomić skrypt.

Output

```
KNF/
report_data/
   reports/
      sheet_name1.json
      sheet_name2.json
      sheet_name3.json
```

Skrypt na podstawie wrzuconego raportu wyekstraktuje dane z raportu oraz utworzy tyle plików transportowych ile jest arkuszy w danym formularzu.

7.3.3 Ładowanie struktury bazodanowej do bazy danych.

Przed załadunkiem struktury bazodanowej trzeba najpierw wykonać akcje nr 1, aby utworzyć pliki transportowe (pełna strukture dla taksonomii zapisana w formatach JSON). Należy uruchomić api w terminalu, jeśli tego jeszcze nie zrobiłeś.

```
uvicorn src.api_interface.api:app --reload
```

po poprawnym uruchomieniu dostaniemu informacje: `INFO: Application startup complete`. Teraz możemy uruchomić akcje i wpisać w terminal wartość 3. Skrypt załaduje Twoja strukture do bazy danych i wyświetli w terminalu komunikat: `"Struktura zosta␣la za␣ladowana do bazy danych"`, jeśli operacja zakończy sie sukcesem.

7.3.4 Ładowanie danych z raportu do bazy danych.

Przed załadunkiem danych do bazy danych, upewnij sie że wykonałeś akcje 2. Należy uruchomić api w terminalu, jeśli tego jeszcze nie zrobiłeś.

```
uvicorn src.api_interface.api:app --reload
```

po poprawnym uruchomieniu dostaniemu informacje: `INFO: Application startup complete`. Teraz możemy uruchomić akcje i wpisać w terminal wartość. Skrypt załaduje twoje wyekstrahowane dane do bazy danych i wyświetli w terminalu komentarz: `"Raport zosta␣l pomyślnie rozadowany do bazy danych"`, jeśli operacja zakończy sie sukcesem.

7.3.5 0 - Wyjście z programu.

Wybierz akcje 0 jeśli chcesz zakończyć program.