Uniwersytet im. A. Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

Praca magisterska

Automatyczna konwersja „Słownika bibliograficznego języka polskiego" do bazy danych

Automatic Conversion of „Bibliographical dictionary of the Polish language" into a Database

Karol Kaczmarek
Numer albumu: 402227
Kierunek: Informatyka

Promotor:

prof. UAM, dr hab. Krzysztof Jassem

Poznań, 2018

Poznań .....

Oświadczenie

Ja, niżej podpisany Karol Kaczmarek student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt: *Automatyczna konwersja „Słownika bibliograficznego języka polskiego" do bazy danych* napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej. Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowipodstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

[TAK]* - wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

[TAK]* - wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

*Należy wpisać TAK w przypadku wyrażenia zgody na udostępnianie pracy w czytelni Archiwum UAM, NIE w przypadku braku zgody. Niewypełnienie pola oznacza brak zgody na udostępnianie pracy.

Spis treści

Streszczenie

Celem niniejszej pracy jest przedstawienie procesu automatycznej konwersji *„Słownika bibliograficznego języka polskiego”* do postaci bazy danych. Temat rozpocznie się wstępem teoretycznym wprowadzającym w tematykę wykonywanego projektu. Automatyczna konwersja zostanie przedstawiona etapami, rozpoczynając od zamiany nieustrukturyzowane danych do postaci ustrukturyzowanej. Następnie ukazany zostanie proces zamiany ustrukturyzowanych dany do postaci XML z wykorzystaniem wyrażeń regularnych. Kolejnym etapem jest utworzenie schematu bazy danych, z wykorzystaniem biblioteki *Yesod* języka programowania *Haskell*. Ostatnim etapem pracy jest automatyczna zamiana niepopranych informacji na poprawną wartość.

Abstract

The purpose of this project is the automatic conversion of *„Bibliographical dictionary of the Polish language"* into a database. Thes is begins with a theoretical introduction to the project. Automatic conversion will be presented step by step. The first stage of the project is conversion of the unstructured data into the structured data. Next, the lines are converted into the XML nodes by means of pre-defined regular expression. The scheme has been designed by *Haskell Persistent* library. The last stage is to replace typos in the structured data.

Wstęp

„Słownik bibliograficzny języka polskiego" Jana Wawrzyńczyka to obszerny, wielotomowy słownik zawierający informacje o charakterze chronologizacyjnym i metasłownikowym. Na podstawie danych w słowiku możliwe jest odpowiedzenie na pytania takie jak: „W jakich słownikach słowo pojawia się jako hasło?", „Jakie są znane wczesne poświadczenia słowa?". Słownik ma bogatą strukturę, obecnie jest jednak dostępny jedynie jako dokument w formacie DOC/PDF. Celem niniejszej pracy jest opracowanie schematu bazy danych reprezentującego słownik oraz automatyczna konwersja słownika w bieżącej postaci do bazy danych. W efekcie możliwe będzie uzyskiwanie w sposób automatyczny odpowiedzi na zapytania typu: „Podaj przymiotniki znane w latach 1901-1914", „Podaj słowa poświadczone cytatami z dzieł Juliusza Słowackiego" w ogólno-dostępnym serwisie internetowym.

Pierwszy dwa rozdziały rozdział przybliżą tematykę w postaci informacji teoretycznych niezbędnych do dalszej pracy. W kolejnym rozdziale przedstawiony zostanie proces zamiany danych nieustrukturyzowanych do ustrukturyzowanych. Następnie, opisany zostanie proces zamiany przetworzonych danych do jednolitego formatu w postaci plików XML. Następnie przedstawiony zostanie schemat bazy danych wygenerowana przy pomocy biblioteki *Yesod* języka programowania *Haskell*. Ostatni etap pracy dotyczy wykrycia i zamiany błędnych informacji na poprawne odpowiedniki.

Rozdział 1

Podstawowe pojęcia

1.1. Wstęp

W niniejszym rozdziale zostanie przestawione zagadnienia związane z przetwarzaniem dokumentów nieustrukturyzowanym, ustrukturyzowanym oraz różnice występujące pomiędzy nimi. Proces automatycznej konwersji ma na celu przetworzenie informacji z nieustrukturyzowanego formatu do jednoznacznie określonego formatu w postaci bazy danych.

1.2. Dokument nieustrukturyzowany

Dokumentem nieustrukturyzowanym (informacją nieustrukturyzowaną) nazywamy dowolny zbiór danych, który nie jest zapisany w jednolitym formacie danych. Przykładem nieustrukturyzowanych danych jest dowolny zbiór danych zapisanych w różnych formatach, np. zestaw plików tekstowych, graficznych czy dźwiękowych. Zbiór takich informacji może ponadto charakteryzować się różnicami w typie zapisu danych, np. dane tekstowe mogą być zapisane w postaci małych, czy wielkich liter. Innym przykładem nieustrukturyzowanej informacji dla danych tekstowych jest zapis tekstu ze znakami diakrytycznymi, jak i z ich pominięciem. W przypadku „Słownika bibliograficznego języka polskiego" dane zostały zapisane w postaci danych tekstowych oraz danych w postaci plików graficznych, które zostały dokładnie opisane w rozdziale 3.

1.3. Dokument ustrukturyzowany

Dokumentem ustrukturyzowanym (informacją ustrukturyzowaną) nazywamy dowolny zbiór danych zapisany zgodnie ze zdefiniowanym schematem.

Schemat zapisu danych określa, w jaki sposób zapisane zostały informacje, zawierający takie informację jak:

- kolejność zapisu danych,
- typ danych (dane liczbowe, tekstowe),
- opcjonalność wystąpienia.

Przykładem ustrukturyzowanej danych jest zapis w postaci bazy danych, czy zapis danych w formacie XML ze schematem opisującym sposób zapisu informacji.

Rozdział 2

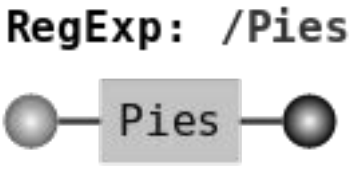
Opis narzędzi

2.1. Wstęp

W tym rozdziale zostanie przedstawione zagadnienie wyrażeń regularnych, które zostały wykorzystywane do pozyskania niezbędnych informacji z dokumentów nieustrukturyzowanych. Następnie zostaną przedstawione narzędzia, które zostały wykorzystane do automatycznej konwersji „Słownika bibliograficznego języka polskiego".

2.2. Wyrażenia regularne

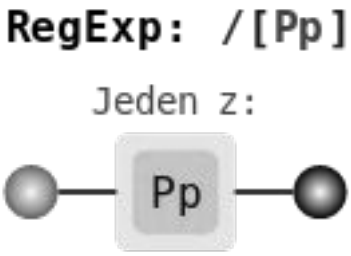
Wyrażenie regularne (ang. *regular expressions*, *regex* lub *regexp*) to wzorzec (ang. *pattern*) opisany za pomocą definiowanego alfabetu określający język wyrażenia regularnego. Alfabetem nazywamy dostępny zbiór symboli (liter, cyfr, znaków diakrytycznych, czy innych znaków dostępnych ze standardu Unikod[1]), gdzie dowolny ciąg symboli nazywamy słowem. Językiem wyrażenia regularnego nazywamy dowolny zbiór akceptowanych słów (pasujących do zdefiniowanego wzorca). Przykładem wyrażenia regularnego jest język przedstawiony przez wzorzec *Pies* akceptujący słowo *Pies* (rys. 2.1).



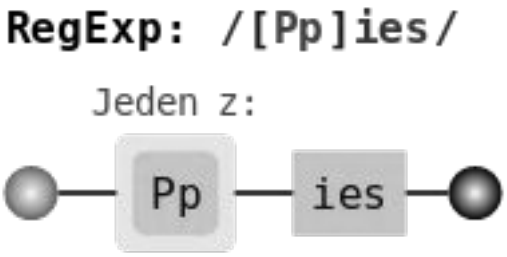
Rys. 2.1. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „Pies".

[1]Unikod (ang. Unicode) jest zbiorem znaków mającym na celu obejmować wszystkie znaki pisma z dowolnego języka na świecie.

Wzorzec wyrażenia regularnego posiada zdefiniowaną listę znaków specjalnych. Jednym z takich znaków specjalnych jest klasa znaków. Klasą znaków nazywamy dowolny zbiór znaków akceptujący znak z wcześniej zdefiniowanego zbioru, zapisany w postaci []. Język akceptujący słowo p lub P wyrażony jest za pomocą wzorca: *[Pp]* (rys. 2.2). Połączenie wzorców z rys. 2.1 oraz 2.2 daje język akceptujący słowa *Pies* oraz *pies* przedstawiony jako wzorzec: *[Pp]ies* (rys. 2.3).



Rys. 2.2. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „P" lub „p".

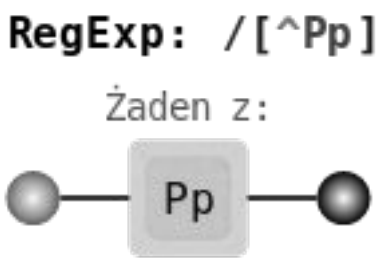


Rys. 2.3. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „Pies" lub „pies".

Zakres znaków to zbiór znaków, który zdefiniowany jest jako zakres początku i końca zakresu oddzielone znakiem myślnika, oznaczający dowolny znak pomiędzy początkiem i końcem zakresu. Język akceptujący cyfrę zdefiniowany jest jako wzorzec: *[0-9]* (rys. 2.4).

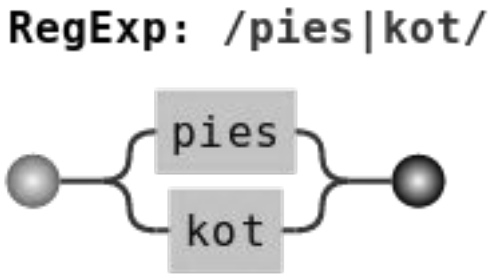
Rys. 2.4. Uproszczony schemat automatu wyrażenia regularnego akceptujący cyfrę.

Innym znakiem specjalnym wyrażenia regularnego jest negacja zbioru. Negacją zbioru nazywamy dowolny znaków różny od zdefiniowanego, zapisana w postaci [ˆ]. Językiem akceptującym wszystkie słowa różne od P oraz p przedstawia wzorzec: *[ˆPp]* (rys. 2.5)



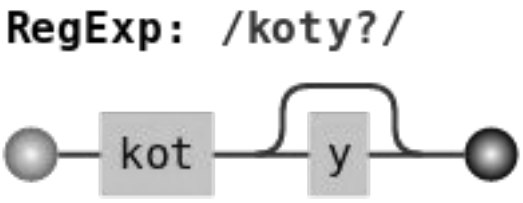
Rys. 2.5. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa różne od „P” oraz „p”.

Kolejnym znakiem specjalnym jest alternatywa zapisana jako znak |. Alternatywą nazywamy wybór jednej z dwóch możliwości wzorca w postaci *wzorzec1*|*wzorzec2*. Język akceptujący słowa *pies* lub *kot* zdefiniowany za pomocą wzorca *pies*|*kot* (rys. 2.6).



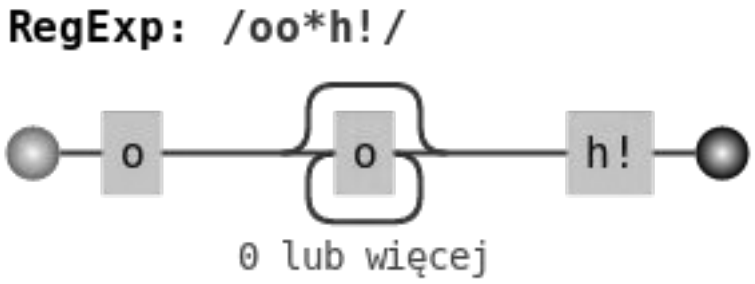
Rys. 2.6. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „pies" lub „kot".

Następnym znakiem specjalnym jest znak ? oznaczający opcjonalność. Opcjonalnością nazywamy możliwość wystąpienia poprzedzającego wzorca, np. język akceptujący słowo *kot* lub *koty* zapisany jest jako wzorzec: *koty?* (rys. 2.7).



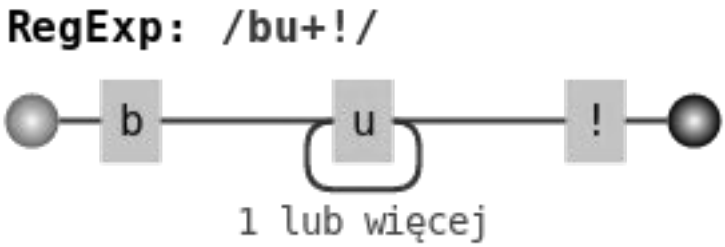
Rys. 2.7. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „kot" lub „koty".

Innym dostępnym znakiem specjalnym jest znak *, który oznacza możliwość wielokrotnego wystąpienia dowolnej ilości razy wzorca (w tym możliwość niewystąpienia wzorca). Język akceptujący słowo *oh!* z uwzględnieniem wielokrotności litery *o*, akceptujący słowa *ooh!*, *oooh!*, *ooooh!*, itd. zdefiniowany jest wzorcem: *oo*h!* (rys. 2.8)



Rys. 2.8. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa „oh!", „ooh!", „oooh!", „ooooh!", itd.

Innym znakiem specjalnym jest znak +, który oznacza możliwość wystąpienia raz lub wielokrotnie zdefiniowanego wzorca. Język akceptujący słowo *bu!* z uwzględnieniem wielokrotności litery *u*, akceptujący słowa *buu!*, *buuu!*, *buuuu!*, itd. Zdefiniowany jest wzorcem: *bu+!* (rys. 2.9)

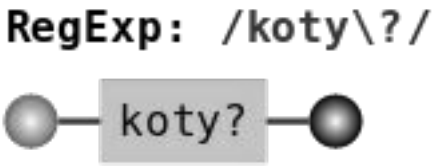


Rys. 2.9. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa „bu!", „buu!", „buuu!", „buuuu!", itd.

Ostatnim omawianym znakiem specjalnym jest znak **.**, który oznacza jeden dowolny znak. Przykładowy wzorzec dowolnego znaku przedstawia rys. 2.10.

Rys. 2.10. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa: „cześć", „część", „cz3ść", itd.

Do zdefiniowania jednego z dostępnych znaków specjalnych we wzorcu jako akceptowalny znak w słowie, należy go poprzedzić znakiem \ (*baskslash*). Przykładem takiego wzorca jest rys. 2.11.

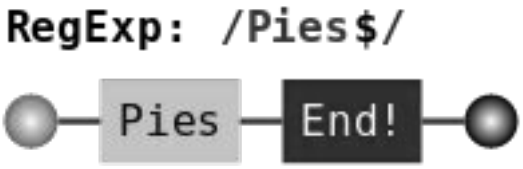


Rys. 2.11. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „koty?".

Kotwice określają zasięg dopasowania wzorca. Pierwszym omawianym znakiem jest znak ^ oznaczający dopasowanie słów rozpoczynających się zdefiniowanym wzorcem. Język akceptujący słowa rozpoczynające się zdefiniowanym wzorcem przedstawia rys. 2.12.

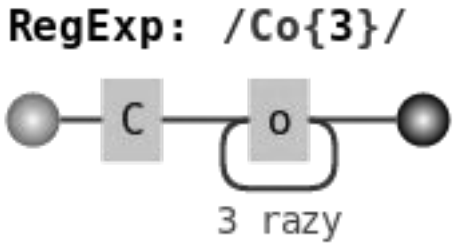
Rys. 2.12. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo rozpoczynające się wzorcem „Kot".

Drugim omawianym znakiem jest znak **$** oznaczający dopasowanie słów kończących się zdefiniowanym wzorcem. Język akceptujący słowa kończące się zdefiniowanym wzorcem przedstawia rys. 2.13.



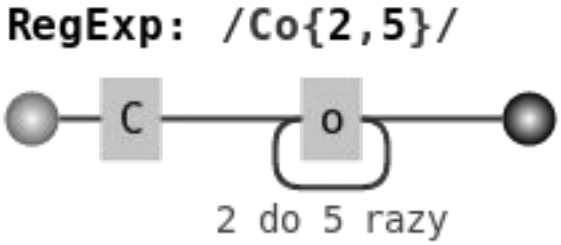
Rys. 2.13. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo kończące się wzorcem „Pies".

Wzorzec określający liczbę wystąpień mówi, ile razy podany wzorzec może wystąpić, oznaczony jest przez **{n}** – gdzie **n** oznacza dokładną liczbę wystąpień wzorca. Przykład z wykorzystaniem liczby wystąpień przedstawia rys. 2.14.



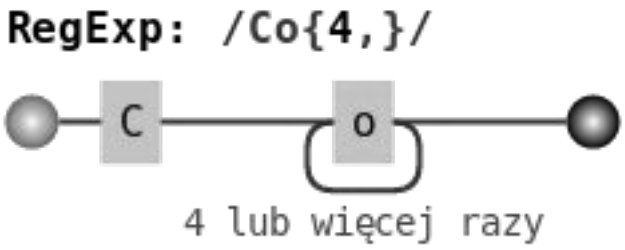
Rys. 2.14. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo „Cooo".

Liczba wystąpień wzorca z zakresu od **m** do **n** opisana jest jako **{m,n}**. Język akceptujący słowa: *Coo*, *Cooo*, *Coooo*, *Cooooo* przedstawiony jest w rys. 2.15.



Rys. 2.15. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa „Coo", „Cooo", „Coooo", „Cooooo".

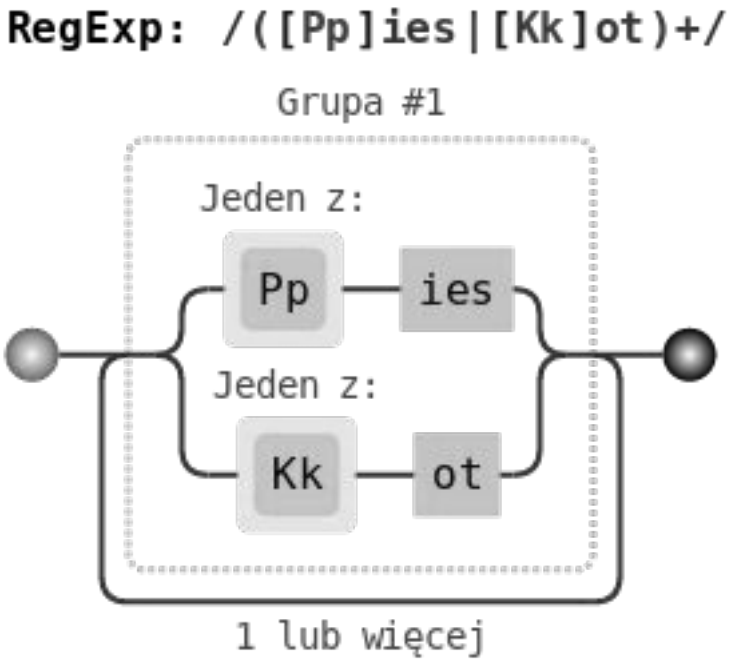
Wystąpienie wzorca co najmniej **n** razy opisane jest jako **{n,}**. Język akceptujący słowa *Coooo* z co najmniej 4 literami **o** przedstawia rys. 2.16.



Rys. 2.16. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowa „Coooo”, „Cooooo”, „Coooooo”, itd.

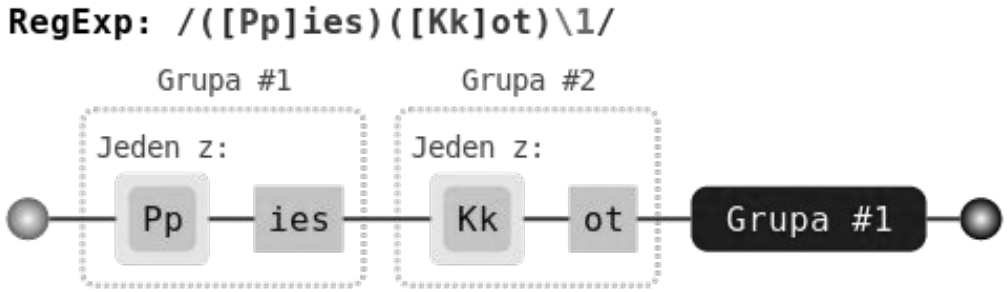
2.2.1. Grupowanie

Grupowanie umożliwia wydzielenie fragmentu wzorca w celu opcjonalności wystąpienia, czy wielokrotnym powtórzeniu. Zapisane jest w postaci (). Przykładowe grupowanie przedstawia rys. 2.17.



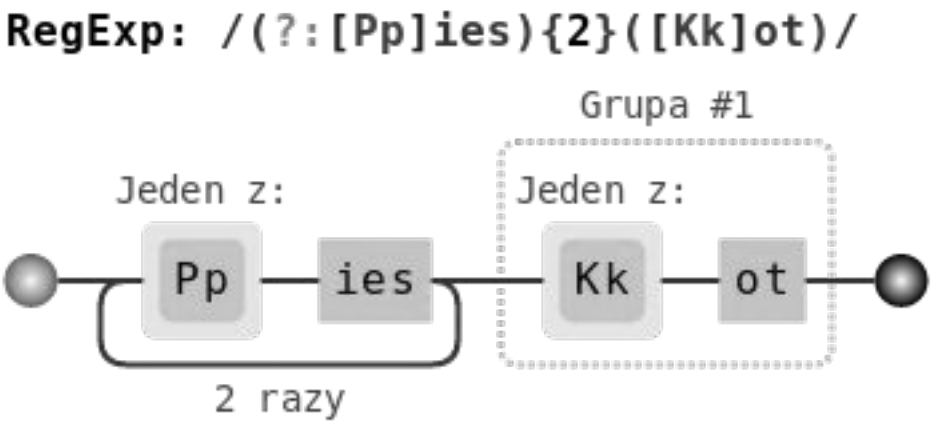
Rys. 2.17. Uproszczony schemat automatu wyrażenia regularnego akceptujący wielokrotność słów: „Kot", „kot", „Pies", „pies".

Referencją nazywamy powtórzeniem zdefiniowanej grupy poprzez wywołanie identyfikatora grupy poprzedzone znakiem \ (*baskslash*). Referencje grupy identyfikowane są od liczby 1, a każda kolejna grupa otrzymuje następny identyfikator. Przykład grupowania przedstawia rys. 2.18.



Rys. 2.18. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo: „PiesKotPies", „PieskotPies", „piesKotpies" lub „pieskotpies"

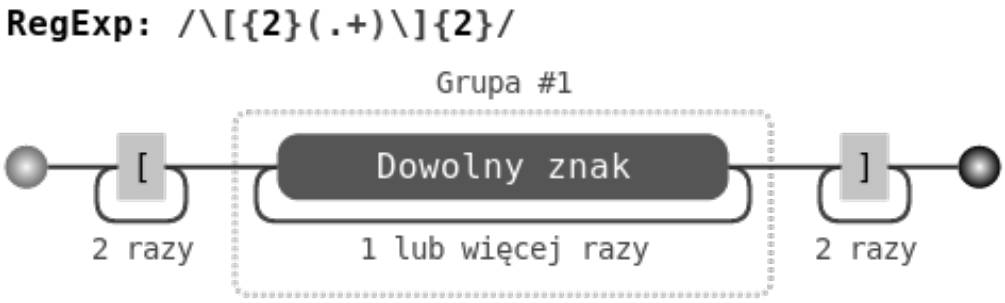
Nieprzechwytywaną grupą nazywamy grupę, dla której nieprzypisywany jest identyfikator. Zastosowanie grupy nieprzechwytywanej daje możliwość pominięcia jednej z operacji: zwielokrotnienia, czy opcjonalnego wystąpienia. Nieprzechwytywanie zapisane jest w postaci **(?:)**. Przykład zwielokrotnienia nieprzechwytywanej grupy przedstawia rys. 2.19.



Rys. 2.19. Uproszczony schemat automatu wyrażenia regularnego akceptujący słowo: „PiesKot", „Pieskot", „piesKot", „pieskot" z pominięciem grupowania słowa „Pies" lub „pies".

2.2.2. Zachłanność dopasowywania

Zachłanność (greediness) dopasowania wzorca odpowiada frazie: „dopasuj tyle ile możesz" – efektem takiego działania jest nadmiarowość dopasowania słowa. Przykładem zachłanności jest dopasowanie słów znajdujących się pomiędzy podwójnymi nawiasami kwadratowymi (rys. 2.20).



Rys. 2.20. Przykład zachłannego dopasowania.

Załóżmy, że ma nastąpić wydobycie informacji zawartych w kwadratowych nawiasach z tekstu:

> Wyrażenia regularne ([[język angielski—ang.]] regular expressions, w skrócie regex lub regexp) – wzorce, które opisują łańcuchy [[symbol]]i. [2]

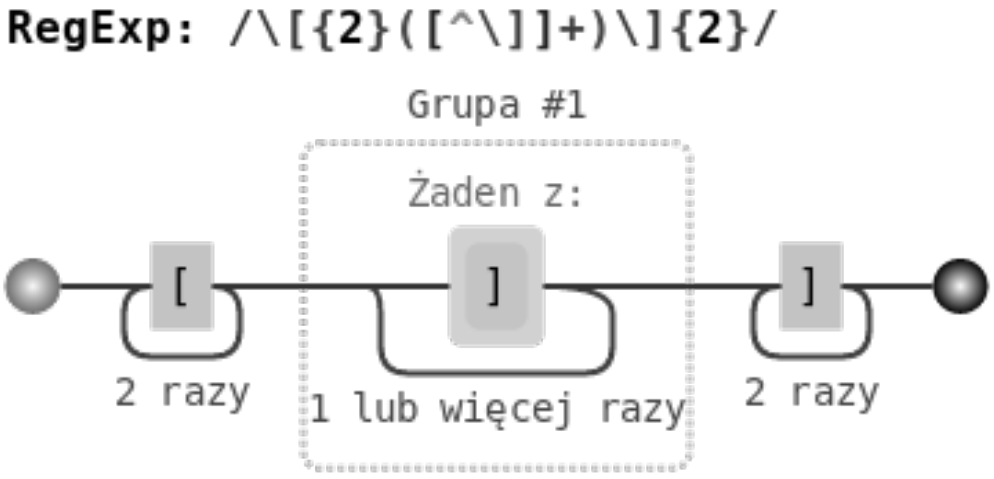
Dopasowanie wyrażenia regularnego z rys. 2.20 jest następujące:

> Wyrażenia regularne ([[**język angielski—ang.**]] **regular expressions, w skrócie regex lub regexp) – wzorce, które opisują łańcuchy** [[**symbol**]]i.

Efektem dopasowania jest nadmiarowo długi napis, wydobyty tekst zawiera więcej danych, niż pierwszy znaleziony w podwójnych kwadratowych nawia-

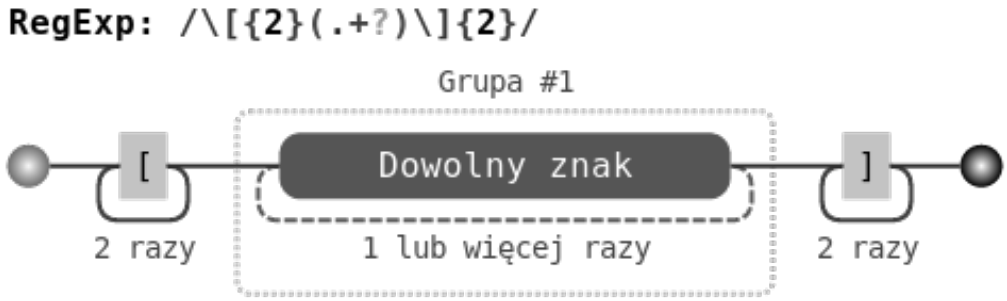
[2]Źródło: Wikipedia (2018) - Wyrażenie regularne

sach. Wykorzystanie znaku kropki jako fragmentu wzorca zapewnia dopasowanie możliwie najdłuższe, jakie istnieje. Jednym ze sposobów, aby uniknąć nadmiaru dopasowania jest zastosowanie negacji dla klasy znaków, co przedstawia rys. 2.21.



Rys. 2.21. Przykład zapobiegnięcia zachłannego dopasowania.

Innym dostępnym sposobem na uniknięcie zachłannego dopasowania jest dopasowanie leniwe, które odpowiada frazie: „dopasuj jak najmniej". Oznaczone znakiem **?**. Przykładem wykorzystania leniwego dopasowania jest wzorzec z rys. 2.22.



Rys. 2.22. Przykład leniwego dopasowania wyrażenia regularnego.

Stosując jedno z wymienionych rozwiązań uzyskuje się dopasowanie:

> Wyrażenia regularne ([[**język angielski**—**ang.**]] regular expressions, w skrócie regex lub regexp) – wzorce, które opisują łańcuchy [[symbol]]i..

Efektem jest uzyskanie pierwszego napisu, który jest otoczony podwójnymi nawiasami kwadratowymi.

2.2.3. Standardy wyrażeń regularnych

Istnieje wiele standardów opisujących wyrażenia regularne. Najbardziej popularne to Basic Regular Expressions (BRE) oraz Extended Regular Expressions (ERE).

Basic Regular Expressions

Basic Regular Expressions (BRE) to jeden z pierwszych standardów opisujących wyrażenia regularne. Charakteryzuje się znakami specjalnymi poprzedzonymi znakiem \– *backslash*:

- [ ] – klasy znaków,
- [^] – negacji klasy znaków,
- **.** – dowolnego znaku,
- *, **{n}**, **{m,n}**, **{n,}** – powtórzeń wzorca,
- ( ) – grupowania,
- ^, **$** – kotwic.

Extended Regular Expressions

Kolejnym standardem opisujący wyrażenia regularne jest *Extended Regular Expressions*. Ze względu na brak konieczności poprzedzania znaków specjalnych znakiem \(*backslash*) stał się najczęściej stosowanym standardem dla wyrażeń regularnych. W przeciwieństwie do *Basic Regular Expressions*, *Extended Regular Expressions* rozszerza o dostęp do znaków specjalnych dotyczących:

- | – alternatywy,
- **?** – opcjonalności,
- + – powtórzeń wzorca.

2.2.4. Zdefiniowane klasy znaków

Wyrażenia regularne posiadają zdefiniowane klasy znaków, które służą skróceniu zapis wyrażenia regularnego. Przykłady zdefiniowanych klas przedstawia tabela 2.1.

Opis	**Zbiór znaków**	**POSIX**[3]	**Inny zapis skrótowy**
Cyfra	[0-9]	[:digit:]	\d
Duże znaki	[A-Z]	[:upper:]	\l
Małe znaki	[a-z]	[:lower:]	\u
Znaki interpunkcyjne	[!"#$%&'()*+,\-./ <=>?@[]^_‘{|}~]	[:punct:]	
Słowo	[A-Za-z0-9_]	[:word:]	\w
Znaki białe	[\t\r\n\v\f]	[:space:]	\s

Tab. 2.1. Lista zdefiniowanych klas znaków dla wyrażeń regularnych.

[3]POSIX jest standardem zapewniający kompatybilność funkcjonalności pomiędzy przeróżnymi systemami operacyjnymi. Definiuje jednorodny standard do przeróżnych dziedzin informatyki, np. dla wyrażeń regularnych. Zapewnia determinizm wykonywanych operacji, niezależnie od systemu operacyjnego.

2.3. Narzędzia

Mnogość dostępnego oprogramowania pozwoliła przyspieszyć proces automatycznej konwersji „Słownika bibliograficznego języka polskiego" poprzez dobór odpowiedniego oprogramowania do zamierzanego celu. W tym przypadku wykorzystany został programy *sed* oraz *tidy-html5* w celu oczyszczenia danych w postaci HTML ze zbędnej, redundantnej informacji. Do zapisu danych w postaci danych ustrukturyzowanych posłużyła biblioteka *lxml* dla języka programowania *Python* oraz relacyjna baza danych zarządzana przez system *PostgreSQL*.

2.3.1. sed

Program **sed** jest narzędziem, które umożliwia wykorzystanie zdefiniowanych wyrażenia regularne do zamiany łańcuchów znaków. Schemat dla zamiany danych jest następujący:

s/wyrażenie_regularne/tekst_zamiany/flaga,

gdzie:

- **wyrażenie_regularne** – wzorzec wyrażenia regularnego
- **tekst_zamiany** – tekst, na który ma zostać zamienione dopasowanie z wyrażenia regularnego,
- **flaga** – sposób działania zamiany dopasowań wyrażeń regularnych:
  - flaga **g** oznacza zmianę wszystkich dopasować dla wyrażenia regularnego (domyślnie zmieniane jest tylko pierwsze dopasowanie do wzorca),

– flaga **i** oznacza zignorowanie wielkości znaków (wielkość znaków we wzorcu nie ma znaczenia).

2.3.2. tidy-html5

Program **tidy-html5** (tidy) jest narzędziem służącym korekcie struktury oraz oczyszczeniu plików HTML. Korekta struktury ma na celu przystosowanie struktury HTML do aktualnych standardów, np. zamianę przestarzałych tagów, czy zamknięcie otwartych znaczników. Oczyszczanie służy niwelowaniu redundancji oraz utworzeniu przejrzystego pliku HTML. Użycie programu przebiega według schematu:
tidy -config tidy.conf -o output.html input.html, gdzie:

- **-config tidy.conf** – oznacza plik konfiguracyjny o nazwie *tidy.conf*
- **-o output.html** – oznacza plik wyjściowy o nazwie *output.html*
- **input.html** – oznacza plik wejściowy o nazwie *input.html*

Plik konfiguracyjny umożliwia zmianę sposobu działania programu *tidy-html5*. Fragment wykorzystanych opcji podczas automatycznej konwersji „Słownika bibliograficznego języka polskiego" przedstawia się następująco:

- **bare** – zamień niełamiący znak spacji na znak spacji,
- **ascii-chars** – zapisz encję HTML jako znak ASCII, np. zapisz encję ” jako znak **"**,
- **clean** – oczyść strukturę HTML ze zbędnych danych, np. pomięcie pustych znaczników HTML,
- **hide-comments** – pomiń komentarze,

- **indent-space** – ustaw stałą szerokość wcięcia struktury HTML, np. na wartość 2 (każde wcięcie zostanie poprzedzone 2 znakami spacji),
- **word-2000** – oczyść strukturę HTML z informacji generowanych przez program Microsoft Word,
- **char-encoding** – zamień kodowanie znaków na podane przez użytkownika, np. na kodowanie UTF-8,

Pozostała lista wykorzystanych opcji konfiguracyjnych przedstawia załącznik 3.12.

2.3.3. lxml

Biblioteka **lxml** dla języka programowania *Python* zapewnia odczyt, zapis oraz walidację danych w formacie HTML oraz XML. Łatwość operacji na plikach HTML oraz XML jest kluczową cechą podczas procesu automatycznej konwersji *„Słownika bibliograficznego języka polskiego”*. Podstawowymi operacjami z wykorzystaniem biblioteki *lxml* są:

1. Wczytanie danych w postaci XML oraz HTML (listing 2.1).

```
# Wczytanie danych w postaci XML
file_xml = 'sbjp.xml'
parser_xml = etree.XMLParser()
tree = etree.parse(file_xml, parser_xml)
# <word>
# <name>dachować</name>
# <text>
# <p>Jadący tuż przede mną samochód przelatuje przez
    barierę, dachuje i zatrzymuje się dopiero w lesie,
```

```
    wywrócony do góry kołami. &lt;„Odrodzenie" (Warszawa)
    1987, 25: 5&gt;</p>
# ...
# <text>
# </word>

# Wczytanie danych w postaci HTML
file_html = 'sbjp.html'
parser_html = etree.HTMLParser()
tree = etree.parse(file_hml, parser_html)
# <html>
# ...
# <body>
# <b>dachować</b>
# <p>Jadący tuż przede mną samochód przelatuje przez
    barierę, dachuje i zatrzymuje się dopiero w lesie,
    wywrócony do góry kołami. &lt;„Odrodzenie" (Warszawa)
    1987, 25: 5&gt;</p>
# ...
# </body>
# </html>
```

Listing 2.1. Wczytanie plik w formacie XML lub HTML przy pomocy biblioteki lxml.

2. Pobranie wszystkich paragrafów tekstu z formatu HTML (listing 2.2).

```
# Pobranie wszystkich paragrafów
result = tree.xpath('//body/div/p')
```

```
print(result)
# [ 'Jadący tuż przede mną samochód przelatuje przez
    barierę, dachuje i zatrzymuje się dopiero w lesie,
    wywrócony do góry kołami. <„Odrodzenie" (Warszawa)
    1987, 25: 5>', ... ]
```

Listing 2.2. Pobranie wszystkich paragrafów z formatu HTML przy pomocy biblioteki lxml.

3. Pobranie danych tekstowych z paragrafu z formatu HTML (listing 2.3).

```
# Pobranie danych tekstowych z paragrafu
result_text = result[0].text
print(result_text)
# Jadący tuż przede mną samochód przelatuje przez barierę,
    dachuje i zatrzymuje się dopiero w lesie, wywrócony do
    góry kołami. <„Odrodzenie" (Warszawa) 1987, 25: 5>
```

Listing 2.3. Pobranie danych tekstowych z paragrafu z formatu HTML przy pomocy biblioteki lxml.

4. Pobranie nazwy encji z formatu HTML (listing 2.4).

```
# Pobranie nazwy encji
result_tag = result[0].tag
print(result_tag)
# p
```

Listing 2.4. Pobranie nazwy encji z formatu HTML przy pomocy biblioteki lxml.

5. Zapisanie nowo utworzonego elementu do formatu XML. (listing 2.5).

```
# Utworzenie nowego elementu o nazwie: 'word'
page = Element('word')
# Dodanie elementu o nazwie 'name' i treści 'dachować'
SubElement(page, 'name').text = 'dachowaę'
# <name>dachować</name>
...
# Zapisanie do formatu XML
ElementTree(page).write('dachowaę.xml', encoding='utf-8',
    xml_declaration=True)
# <word>
# <name>dachować</name>
# ...
# </word>
```

Listing 2.5. Zapisanie nowo utworzonego elementu do formatu XML.

Łatwość i prostota biblioteki pozwalają skupić się na procesie automatycznej konwersji, a nie na procesie wczytywania danych.

2.3.4. PostgreSQL

Miejscem zapisu ustrukturyzowanych danych jest relacyjna baza danych zarządzana przez system PostgreSQL. Wybór struktury bazy danych wiąże się z podziałem danych na table. Każda z tabel posiada zdefiniowaną listę pól z określonym typem danych. Listę typów danych wykorzystanych w projekcie prezentuje tabla 2.2. Wydzielenie danych do osobnych tabel wiąże ze sobą

Type	**Opis**	**Przykład typu danych słownika**
bigint	Liczbowy typ danych (liczby całkowite)	Unikalny identyfikator table, numer czasopisma, strony czasopisma
date	Czasowy typ danych	Data wydania utworu
text	Tekstowy typ danych	Nazwa czasopisma, tytuł utworu, treść cytatu, imię, nazwisko autora

Tab. 2.2. Lista typów danych wykorzystanych w bazie danych PostgreSQL.

możliwość połączenia pojedynczego rekordu tabeli do wielu rekordów z innej tabeli. W takim przypadku wymagana jest pośrednia tabela łącząca poszczególne tabele.

Rozdział 3

Proces automatycznej konwersji

3.1. Wstęp

W tym rozdziale zostanie przestawiony proces konwersji „Słownika bibliograficznego języka polskiego" do postaci bazy danych. Proces przetwarzania został rozpoczęty od analizy i normalizacji danych wejściowych. Następnie przeprowadzono proces przetwarzana danych do jednolitego formatu. Efektem końcowym procesu są dane w postaci ustrukturyzowanych dokumentów, które łatwo zapisać do bazy danych.

3.2. Analiza danych wejściowych

Przed rozpoczęciem przetwarzania wykonana została analiza danych wejściowych, której efektem jest szablon procesu przetwarzania, zawierający informację o:

1. formacie zapisu danych wejściowych,
2. etapach przetwarzania i wykorzystaniu gotowych narzędzi,
3. formacie zapisu danych wyjściowych.

W przypadku „Słownika bibliograficznego języka polskiego" danymi wejściowymi są pliki programu *Microsoft Word*, które zostały podzielone na dwie kategorie:

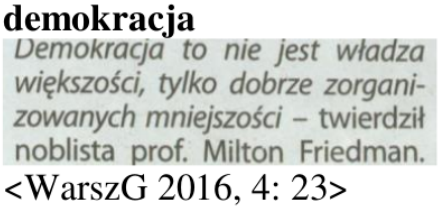
- dane tekstowe – zawierające hasła w postaci tekstowej (rys. 3.1),
- foto-aneks – zawierający hasła w postaci graficznej (rys. 3.2).

Dodatkowo, każda z kategorii została podzielona względem zakresu obejmowanych słów, które zostały zapisane w nazwie pliku. Przykładowo, nazwa

pliku **SBJP (D-G).docx** mówi, iż powyższy plik obejmuje hasła rozpoczynające się od litery **D** i kończy się na hasłach rozpoczynających się literą **G**; są to np. słowa: dachówka, dalece, demokracja, ewangelicki, Fukushima, fotooperator, Gutenberg, gżynek. Przedrostek **SBJP** jest skrótem pierwszych liter od nazwy „**S**łownika **B**ibliograficznego **J**ęzyka **P**olskiego" i ma na celu identyfikację plików.

dalba D+
dalbóg D+
JPol 54 (1974): 214
dalece D+
Bogusławski A., Wawrzyńczyk J. 1993: 97
dalecyna
Sobolewska K. 2004

Rys. 3.1. Dane tekstowe.



Rys. 3.2. Foto-aneks.

Hasła zapisane w postaci tekstowej (rys. 3.1) charakteryzują się strukturą, która opisana jest schematem: pogrubione słowo kluczowe (główka hasła) oraz informacje o miejscu wystąpienia (cytat, nazwa czasopisma, publikacja lub inne informacje identyfikujące). Dane zapisane w foto-aneksie (rys. 3.2) występują w formacie: pogrubione słowo kluczowe (główka hasła), wystąpienie słowa w postaci pliku graficznego oraz informacja o miejscu wystąpienia.

Plan przetworzenia plików programu *Microsoft Word* składa się z następujących etapów: zapisanie danych wejściowych do postaci plików HTML, konwersja do postaci pojedynczych haseł w formacie XML oraz zapis do bazy danych. Cały proces został zaimplementowany z wykorzystaniem bibliotek języka programowania *Python* oraz poleceń systemu operacyjnego *Linux*. Wykorzystane zostały takie narzędzia jak: *sed* (2.3.1), *lxml* (2.3.3), *tidy-html5* (2.3.2), *PostgreSQL* (2.3.4).

3.3. Przetworzenie plików wejściowych

Zapis danych w postaci plików HTML ułatwia dalszą pracę w celu przekształcenia do zunifikowanego formatu. Dane generowane za pomocą programu Microsoft Word zawierają redundantną informację, która została wyróżniona kolorem czerwonym (rys. 3.3).

```
<p class=MsoNormal style='text-align:justify'>
 <b style='mso-bidi-font-weight:normal'>
  <span style='font-size:8.5pt;mso-bidi-font-size:10.0pt'>dalba</span>
 </b>
 <span style='font-size:8.5pt;mso-bidi-font-size:10.0pt'>
  <span style='mso-spacerun:yes'> </span>D+<o:p></o:p>
 </span>
</p>
```

Rys. 3.3. Dane wygenerowane z formatu DOCX do HTML za pomocą programu *Microsoft Word*

Dzięki programowi *tidy-html5* możliwe jest wyczyszczenie plików HTML ze zbędnych tagów, czy atrybutów (rys. 3.4). Wykorzystane parametry dla programu *tidy-html5* zostały umieszczone w załączniku numer 3.12.

```
<p>
 <b>dalba</b> D+
</p>
```

Rys. 3.4. Wyczyszczone dane w formacie HTML dzięki programowi *tidy-html5*

Dane przetworzone programem *tidy-html5* należy oczyścić ze znaków specjalnych lub specyficznych znaków za pomocą programu *sed*. Przykładowe polecenia programu *sed*:

- usuwające znaki miękkiego łącznika przedstawia listing 3.1,

```
sed 's/\xC2\xAD//g'
```

Listing 3.1. Przykładowe usunięcie znaków miękkiego łącznika dzięki programowi *sed*

- normalizujące znaki spacji – zamiana znaków twardej spacji (0xA0) na spację (0x20) przedstawia listing 3.2,

```
sed 's/[&]nbsp;/ /g'
```

Listing 3.2. Przykładowa zamiana znaków twardej spacji na znak spacji z wykorzystaniem programu *sed*

- scalające identyczne tagi HTML – usunięcie powtórzeń przedstawia listing 3.3,

```
sed 's/<\/b><b>//g'
```

Listing 3.3. Przykładowe scalenie tagów <b> z wykorzystaniem programu *sed*

3.4. Wydobycie haseł do pojedynczych próbek

Oczyszczony plik HTML (rys 3.5) został podzielony na pojedyncze pliki (próbki) w obrębie główki hasła. Wydobycie każdej główki hasła zapisane jest w przejściowym formacie danych, który pozwala znormalizować niejednolite dane wejściowe. Domyślnym formatem zapisu danych jest format XML, który reprezentuje dane w strukturalny sposób.

```
<p><b>dąbrowszczak</b> Ds+</p>
<p>JPol 51 (1971): 194-204; Pluta F. 1975: 237</p>
<p>& Turlejska M. 1989: 19</p>
<p><b>dąga</b></p>
<p>JPol 47 (1967): 61-63, 65, 71</p>
```

Rys. 3.5. Oczyszczony plik HTML przed podziałem na pojedyncze pliki.

Nazwa każdej próbki jest równoważna główce hasła, np. dla główki hasła **dąbrowszczak** utworzony został plik o nazwie **dąbrowszczak.xml**. Przykładowe wydzielenie główek hasła do pojedynczych plików ukazują rysunki 3.6 - 3.7.

```
<word>
 <name>dąbrowszczak</name>
 <text>
  <p>Ds+</p>
  <p>JPol 51 (1971): 194-204</p>
  <p>Pluta F. 1975: 237</p>
  <p>&amp;amp; Turlejska M. 1989: 19</p>
 </text>
</word>
```

Rys. 3.6. Przetworzenie główki hasła **dąbrowszczak** do pliku o nazwie **dąbrowszczak.xml**.

```
<word>
  <name>dąga</name>
  <text>
    <p>JPol 47 (1967): 61-63, 65, 71</p>
  </text>
</word>
```

Rys. 3.7. Przetworzenie główki hasła **dąga** do pliku o nazwie **dąga.xml**.

Model pojedynczego pliku dla główki hasła charakteryzuje się strukturą zawierającą:

- element **name** – odpowiadający za nazwę główki hasła,
- element **text** – zawierający wiersze z danymi.

Wiersze danych wydobywane są w obszarze aktualnie przetwarzanej główki hasła, a kończą się na wystąpieniu kolejnej główki hasła. Każdy z wierszy sprawdzany pod kątem zawierania znaku separacji. Znakami separacji są:

- znaki średnika (**;**),
- znaki nowej linii (**\n**, równoważny znacznikowi <**br**>),
- znaki kuli (•).

Znak separacji może wystąpić kilka razy w obrębie jednego wiersza, co prezentuje rysunek 3.8 - 3.9.

```
<p>Kurkowska H., Skorupka S. 1966: 25 • Smółkowa T.
1976 • JPol 57 (1977): 129 • Wawrzyńczyk J. 1990<br>
& Manifest Dada wydał po raz pierwszy Tristan Tzara,
Żyd rumuński. &lt;NPLiS 1921, 1-3: 147&gt;</p>
```

Rys. 3.8. Jednolity tekst przed podziałem według znaków separacji.

```
<p>Kurkowska H., Skorupka S. 1966: 25</p>
<p>Smółkowa T. 1976</p>
<p>JPol 57 (1977): 129</p>
<p>Wawrzyńczyk J. 1990</p>
<p>&amp;amp; Manifest Dada wydał po raz pierwszy Tristan
Tzara, Żyd rumuński. &lt;NPLiS 1921, 1-3: 147&gt;</p>
```

Rys. 3.9. Tekst podzielony według znaków separacji.

Efektem końcowym są dane zapisane w pojedynczych próbach, które zawierają podzielone informacje według wierszy. Każdy z wierszy odpowiada pojedynczemu źródłu, ułatwiając dalszy proces przetwarzania.

3.5. Przygotowanie danych z zapisu do bazy danych

Dane zapisane w jednolitym formacie umożliwiają wydzielenie procesu ekstrakcji danych do formatu wyjściowego (modelu bazy danych) jako niezależnego etapu. Etap zamiany dany do postaci bazy danych przebiega według schematu:

1. Pobierz wiersz z pliku pojedynczej próbki.
2. Sprawdź czy wiersz pasuje do jednego z wzorców (3.5.1).
   (a) Jeżeli znaleziono dopasowanie, zamień wiersz do postaci bazy danych.
   (b) W przeciwnym wypadku oznacz wiersz jako nieprzetworzony.
3. Ponów cały proces, gdy istnieje kolejny wiersz tekstu.

3.5.1. Wzorce dopasowania wierszy

Wzorcem dopasowania pojedynczego wiersza jest zbiór wyrażeń regularnych (rozdział 2.2), który opisuje proces pozyskiwania elementów z aktualnie przetwarzanego wiersza. Elementami pozyskanymi za pomocą wzorców są takie części jak:

- autor:
  - imię autora,
  - nazwisko autora,
  - inicjały imion autora,

  - pseudonim artystyczny,

- treść cytatu,
- nazwa czasopisma,
- tytuł utworu,
- miejsce publikacji,
- nazwa wydawcy,
- data wydania,
- numer utworu,
- numer strony w utworze,
- nazwa słownika,
- odnośnik do innej główki hasła,
- adnotacje, komentarz.

Każdemu elementowi odpowiada wyrażenie regularne, które zostało zdefiniowane tak, aby umożliwić wykorzystanie w wielu wzorcach dopasowania wiersza. Dodatkowo, każdy element został połączony w grupę (2.2.1) o określonej nazwie w celu identyfikacji zwracanej wartości. Przykładowe wyrażenie regularne dopasowującego nazwisko autora przedstawione jest na rysunku 3.10.

```
REGEX_TEXT_AUTHOR_SURNAME =
    r'(?P<surname>\p{Lu}\p{Ll}+(?:[-]\p{Lu}\p{Ll}+)?)'
```

Rys. 3.10. Wyrażenie regularne dopasowujące nazwisko autora.

Wyrażenie regularne (rys. 3.10) zostało przypisane do zmiennej (zaznaczonej kolorem zielonym na rysunku 3.10) w celu późniejszego wykorzystania we wzorcu dopasowania wierszy. Wartość nazwiska autora została przypisana do grupy o nazwie *surname*, która odpowiada za dopasowanie nazwiska (w tym nazwiska dwuczłonowego), np. *Mickiewicz*, *Nowak-Frankowska*, *Słowacki*, *Śnieżyńska-Stolot*, *Rytel-Kuc*.

Innym wyrażeniem regularnym, odpowiadającym za dopasowanie informacji o inicjałach imienia, przedstawione jest na rysunku 3.11.

```
REGEX_TEXT_AUTHOR_INITIAL =
    r'(?P<initial>\p{Lu}\p{Ll}?[.](?: \p{Lu}\p{Ll}?[.])?)'
```

Rys. 3.11. Wyrażenie regularne dopasowujące inicjały imion autora.

Wyrażenie regularne (rys. 3.11) zostało również przypisane do zmiennej (zaznaczonej kolorem zielonym na rysunku 3.11), w której inicjały imienia zostały zapisane do grupie o nazwie *initial*. Przykładem dopasowania inicjałów dla imienia są wartości: *A.*, *K.*, *T. A.*, *Ju. L.*.

Połączenie dwóch wcześniej zdefiniowanych wyrażeń regularnych (3.10 oraz 3.11) daje wzorzec dopasowujący sekwencję w postaci:

- inicjały imienia, nazwisko autora (rys. 3.12)

```
REGEX_TEXT_AUTHOR_INITIAL_SURNAME =
   REGEX_TEXT_AUTHOR_INITIAL + r' ' + REGEX_TEXT_AUTHOR_SURNAME
```

Rys. 3.12. Wzorzec dopasowujący dane autora w postaci inicjałów imienia i nazwiska.

Przykładem pozyskanych danych przez wzorzec z rys. 3.12 są wartości: *A. Mickiewicz*, *M. Nowak-Frankowska*, *D. Rytel-Kuc*, *J. Słowacki*, *E. Śnieżyńska-Stolot*.

- nazwisko, inicjały imienia autora (rys. 3.13)

```
REGEX_TEXT_AUTHOR_SURNAME_INITIAL =
    REGEX_TEXT_AUTHOR_SURNAME + r' ' + REGEX_TEXT_AUTHOR_INITIAL
```

Rys. 3.13. Wzorzec dopasowujący dane autora w postaci nazwiska i inicjału imienia.

Przykładem pozyskanych przez wzorzec z rys. 3.13 są wartości: *Celma-Panek J.*, *Czepieliński F.*, *Kaproń-Charzyńska I.*, *Mosiołek-Kłosińska K.*, *Stadtmüller K.*, *Wierzchoń P.*

Tworzenie wzorców składających się z innym wzorców, ułatwia manipulację wyrażeniami regularnymi w celu uszczegółowienia pozyskiwanych elementów. Zmiana w jednym wyrażeniu regularnym przynosi zmianę we wszystkich wzorcach, które wykorzystują zmodyfikowane wyrażenie regularne. Zdefiniowanie nazwy grup mają na celu automatyczne przeniesienie danych ze wzorców do plików wyjściowych (modelu bazy danych) w postaci XML[1]. Schemat pliku w formacie XML przedstawiony został w załączniku 3.45.

Dane zapisane w słowniku bibliograficznym języka polskiego zostały podzielone na trzy kategorie:

- dane słownikowe,
- dane dotyczące utworu,
- niepoprawne dane.

3.5.2. Dane słownikowe

Dane słownikowe opisują, w jakim słowniku języka polskiego znaleziona została główka hasła. Wiersze tekstu należącej do danych słownika zapisane

[1]Sposób mapowania pozyskanych grup ze wzorców do pliku wyjściowego (modelu bazy danych) przedstawia załącznik 3.7.

są w postaci dwuczłonowego skrótu:

- nazwy skróconej – składającej się z jednego lub dwóch znaków,
- statusu dostępności:
  - znakiem + – informującym, że główka hasła znajduje się w słowniku,
  - znakiem * – informującym, że główka hasła znajduje się w słowniku jako wariant pisowni główki hasła, np. **delektować się** wyprowadzone z **delektować**, czy **faksowy** wyprowadzony z główki hasła **faks**,
  - znakiem - – informującym, że główka hasła nie znajduje się w słowniku.

Wyrażenie regularne opisujące dane słownikowe przedstawione jest na rysunku 3.14.

```
REGEX_TEXT_DICT =
    r'(?P<abbr>[A-Z][1a-z]?)(?P<status>[*+-])'
```

Rys. 3.14. Wyrażenie regularne dopasowujące dane słownikowe.

Pierwszym elementem pozyskanym z wyrażenia regularnego (rys. 3.14) jest nazwa słownika (grupa o nazwie *abbr*) oraz status dostępności główki hasła w słowniku (grupa o nazwie *status*). Przykładem dopasowania jest wartość: *Ds+* (skrót *Ds* oznacza słownik W. Doroszewskiego z roku 1969).

Opcjonalnym elementem dla danych słownikowych są dopiski (adnotacja) autora lub informacja o nazwie główki hasła, z jakiego zostało wyprowadzone. Dopiskiem autora jest krótka notatka dotycząca główki hasła, np. informacja o braku cytatu dla główki hasła (skrót *bez cyt.*).

```
REGEX_TEXT_DICT_ANNOTATION =
    r'[(](?P<dict_annotation>[^)]+)[)]'
```

Rys. 3.15. Wyrażenie regularne dopasowujące dopisek autora dla danych słownikowych.

```
REGEX_TEXT_DICT_BASE =
   REGEX_TEXT_DICT + r'(?: ' + REGEX_TEXT_DICT_ANNOTATION + r')?'
```

Rys. 3.16. Wzorzec dopasowujący dane słownikowe z dopiskami autora.

Pierwszy wzorzec (rys. 3.15) odpowiada za wydobycie adnotacji (dopisek) autora, (grupa *dict_annotation*), a kolejny wzorzec (rys. 3.16) służy do pozyskania kompletnej informacji o danych słownikowych, np. *W+ (bez cyt.)* (skrót *W* oznacza słownik J. Karłowicza, A. A. Kryńskiego, W. Niedźwiedzkiego z roku 1900-1927, a dopiskiem autora jest wartość *bez cyt.*).

Wydobycie informacji o nazwie główki hasła, z jakiego została wyprowadzona (grupa o nazwie *dict_reference*), następuje z dopisku autora (grupa *dict_annotation* - 3.15). Wzorzec przedstawiony został na rysunku 3.17. Przykładem pozyskania informacji o główce hasła jest wartość *delektować* dla wiesza: *D* (s.v. delektować).*

```
REGEX_TEXT_DICT_REFERENCE_FROM_ANNOTATION =
    r's[.]v[.](?P<dict_reference>.+)'
```

Rys. 3.17. Wyrażenie regularne dopasowujące informację o nazwie główki hasła, z jakiego została wyprowadzone.

Dane dotyczące utworu

Dane dotyczące utworu składają się z takich elementów jak: nazwa czasopisma, tytuł utworu, cytat, dane o autorze, data wydania, nazwa wydawcy z adresem oraz lokalizacja główki hasła: numer czasopisma, strony.

Elementem najczęściej występującym dla każdego wiersza jest data wydania czasopisma lub utworu, która reprezentowana jest wyrażeniem regularnym na rysunku 3.18.

```
REGEX_TEXT_SOURCE_YEAR =
    r'(?P<year>[12][0-9]{3}
    (?:[-](?:[12][0-9]{3}|[0-9]{2}))?)(?P<subinfo>[abcd])?'
```

Rys. 3.18. Wyrażenie regularne dopasowujące datę wydania.

Data wydania występuje w 3 możliwych formatach:

- pojedynczego roku, np. *2017*,
- zakresu lat:
  - w postaci pełnych lat, np. *2017-2018*,
  - w postaci skróconej, np. *2017-18*.

Opcjonalnym elementem dotyczącym daty jest kwartał roku, przedstawiony wartościami:

- a – I kwartał roku,
- b – II kwartał roku,
- c – III kwartał roku,
- d – IV kwartał roku.

Przykładem dopasowania dla daty wydania są wartości: *1986*, *1990*, *1980c*, *2017a*, *1905-1907*, *1939-1949*, *1935-36*.

Pierwszym przykładem dopasowania wiersza do postaci przetworzonych danych jest połączenie wzorca dla autora (3.12) ze wzorcem daty (3.18), który przedstawiony jest na rysunku 3.19.

```
REGEX_SOURCE_1 =
  r'^(?:[&-] )?' + REGEX_TEXT_AUTHOR_SURNAME_INITIAL
  + r' ' + REGEX_TEXT_SOURCE_YEAR + r'$'
```

Rys. 3.19. Wzorzec dopasowujący źródło danych (numer 1).

W części przypadków wiersze dotyczące utworów rozpoczynają się znakiem:

- łącznika (-) – łącznik pomiędzy główką hasła, a treścią,
- ampersand (& – w plikach programu Microsoft Word występuje jako znak książki: 🕮) – cytat dokumentacyjny lub adres bibliograficzny tekstu.

Efektem dopasowania dla wzorca na rysunku 3.19 są dane w formacie XML.

Wiersz o treści *„Smółkowa T. 1976”* został przekształcony do postaci XML przestawionej na rysunku 3.20.

```
<quote>
 <quotetext/>
 <source>
  <authors>
   <author>
    <initial>T.</initial>
    <surname>Smółkowa</surname>
   </author>
  </authors>
  <year_from>1976</year_from>
  <year_to>1976</year_to>
 </source>
</quote>
```

Rys. 3.20. Przetworzony wiersz według wzorca 3.19 (przykład numer 1).

Wiersz o treści *„Umińska-Tytoń E. 2001”* został przekształcony do postaci XML przestawionej na rysunku 3.21.

```
<quote>
 <quotetext/>
 <source>
  <authors>
   <author>
    <initial>E.</initial>
    <surname>Umińska-Tytoń</surname>
   </author>
  </authors>
  <year_from>2001</year_from>
  <year_to>2001</year_to>
 </source>
</quote>
```

Rys. 3.21. Przetworzony wiersz według wzorca 3.19 (przykład numer 2).

Wiersz o treści *„Javorskaja Ju. L. 1983”* został przekształcony do postaci XML przestawionej na rysunku 3.22.

```
<quote>
 <quotetext/>
 <source>
  <authors>
   <author>
    <initial>Ju. L.</initial>
    <surname>Javorskaja</surname>
   </author>
  </authors>
  <year_from>1983</year_from>
  <year_to>1983</year_to>
 </source>
</quote>
```

Rys. 3.22. Przetworzony wiersz według wzorca 3.19 (przykład numer 3).

Wiersz o treści *„- Lyons J. 1989”* został przekształcony do postaci XML przestawionej na rysunku 3.23.

```
<quote>
 <quotetext/>
 <source>
  <authors>
   <author>
    <initial>J.</initial>
    <surname>Lyons</surname>
   </author>
  </authors>
  <year_from>1989</year_from>
  <year_to>1989</year_to>
 </source>
</quote>
```

Rys. 3.23. Przetworzony wiersz według wzorca 3.19 (przykład numer 4).

Wiersz o treści *„& Szulborski T. 1914”* został przekształcony do postaci XML przestawionej na rysunku 3.24.

```
<quote>
 <quotetext/>
 <source>
  <authors>
   <author>
    <initial>T.</initial>
    <surname>Szulborski</surname>
   </author>
  </authors>
  <year_from>1914</year_from>
  <year_to>1914</year_to>
 </source>
</quote>
```

Rys. 3.24. Przetworzony wiersz według wzorca 3.19 (przykład numer 5).

Dane dotyczące utworu (3.20-3.24) zapisane są w elemencie o nazwie *quote*. Każdy przetworzony wiersz zapisany jest w pojedynczym elemencie, co pozwoli odróżnić wiersze w pliku. Pierwszym elementem podrzędnym jest element o nazwie *quotetext*, który zawiera informację o treści cytatu, w tym przypadku (3.20-3.24) jest to element pusty (brak cytatu). Element o nazwie *source* zawiera pełną informację o utworze (dla przykładów 3.20-3.24):

- element *authors* – zawiera informację o wszystkich autorach (może być pusty):
  - element *author* – zawiera informacje o poszczególnym autorze:
    - element *initial* – zawiera informacje o inicjałach imienia autora,
    - element *surname* – zawiera informacje o nazwisku autora,
- element *year_from* oraz *year_to* – zawiera informację o dacie wydania (odpowiednia *data od* oraz *data do*).

Ponadto, zdefiniowane zostały wyrażenia regularne, które odpowiadają za pozyskanie:

- nazwy czasopisma (rys. 3.25):

```
REGEX_TEXT_PERIODICAL =
  r'(?P<periodical>\p{L}+)'
```

Rys. 3.25. Wyrażenie regularne dopasowujące nazwę czasopisma.

  Grupa odpowiedzialna za pozyskanie nazwy czasopisma ma nazwę *periodical*. Przykładem nazw czasopism są dane:

  - *GPolC* (skrót od: Gazeta Polska Codziennie),
  - *NDzien* (skrót od: Nasz Dziennik),
  - *Rzeczpospolita*,
  - *WarszG* (skrót od: Warszawska Gazeta),

- tytuł utworu (rys. 3.26):

```
REGEX_TEXT_TITLE_COMMA =
  r'(?P<title>[^,]+)[,]'
```

Rys. 3.26. Wyrażenie regularne dopasowujące tytuł utworu.

  Grupa o nazwie *title* odpowiada za pozyskanie tytułu utworu. Przykładem są wartości:

  - *Czteroletnia filozofka*,
  - *Ksiądz Helena. Wybór utworów dramatycznych*,
  - *Pani Eliza. Opowieść o Elizie Orzeszkowej*,
  - *Świeradów Zdrój i okolice*,

- treści cytatu (rys. 3.27):

```
REGEX_TEXT_QUOTE_LT =
    r'(?P<quote>[^<]+)[<]'
```

Rys. 3.27. Wyrażenie regularne dopasowujące treść cytatu.

Grupa odpowiedzialna za pozyskanie treści cytatu nazywa się *quote*. Przykładem cytatów są wartości:

  - *Aż dziw, że wilkowi jeszcze smakuje... ta lisem podszyta owieczkowatość.*,
  - *budynki przysłupowo-zrębowe*,
  - *Pewnego poobiedzia, z nieodłączną wyżlicą Azą u nogi, zajrzał do żony.*,
  - *przyrządowe media*,
  - *Rodzina jest wielkim bogactwem społecznym, którego nie mogą zastąpić żadne inne instytucje, podkreślił Ojciec Święty w Guayaquil.*,
  - *Te moje zapiski – co to jest? Dziennik, niedziennik, codziennik, poniedzielnik i wtornik, a nawet niekiedy nocnik (od pisania nocą) esej, eseidło, powieść, powieścidełko?*,

- nazwa wydawcy (rys. 3.28):

```
REGEX_TEXT_PUBLISHER_COMMA =
  r'(?P<publisher>[^,]+)[,]'
```

Rys. 3.28. Wyrażenie regularne dopasowujące nazwę wydawcy.

Nazwa wydawcy pozyskana jest z grupy o nazwie *publisher*, np.

  - *Archidiecezjalne Wydawnictwo Łódzkie*,
  - *Nasza Księgarnia*,
  - *Sport i Turystyka*,
  - *Wyd. Literackie*,

- adres wydawcy (rys. 3.29):

```
REGEX_TEXT_ADDRESS =
  r'(?P<address>\p{Lu}\p{Ll}+)'
```

Rys. 3.29. Wyrażenie regularne dopasowujące adres wydawcy.

Grupa o nazwie *address* odpowiada za pozyskanie adresu wydawcy. Przykładem adresów wydawcy są wartości:

  - *Amsterdam*,
  - *Bydgoszcz*,
  - *Kraków*,
  - *Oxford*,
  - *Poznań*,
  - *Warszawa*,

- numer strony (rys. 3.30):

```
REGEX_TEXT_SOURCE_PAGE =
  r'(?P<page>[0-9]{1,4}(?:[-][0-9]{1,4})?)'
```

Rys. 3.30. Wyrażenie regularne dopasowujące numer strony.

Grupa odpowiedzialna za pozyskanie numeru strony nazywa się *page*. Przykładem dopasowania numeru strony są:

  - *22*,
  - *101-103*,
  - *157*,
  - *208*,
  - *360-361*,

- numer wydania (rys. 3.31):

```
REGEX_TEXT_SOURCE_ISSUE =
  r'(?P<issue>[0-9]{1,4}(?:[-][0-9]{1,4})?)'
```

Rys. 3.31. Wyrażenie regularne dopasowujące numer wydania.

Grupa odpowiedzialna za pozyskanie numeru wydania nazywa się *issue*. Przykładem dopasowania numeru wydania są:

  - *9*,
  - *56*,
  - *11-12*,
  - *195*,
  - *256*.

Dane dotyczące czasopism Przetworzenie wierszy zawierających nazwę czasopisma następuje według zdefiniowanego wzorca przedstawionego na rysunku 3.32.

```
REGEX_QUOTE_1 =
  r'^[&] ' + REGEX_TEXT_QUOTE_LT
  + REGEX_TEXT_PERIODICAL + r' '
  + REGEX_TEXT_SOURCE_YEAR + r'[,] '
  + REGEX_TEXT_SOURCE_ISSUE + r'[:] '
  + REGEX_TEXT_SOURCE_PAGE + r'[>]$'
```

Rys. 3.32. Wzorzec dopasowujący wszystkie dane zawierające nazwę czasopisma.

Przykłady przetworzenia danych przedstawione są na rysunkach 3.33 - 3.36.

Wiersz o treści *„& Rodzina jest wielkim bogactwem społecznym, którego nie mogą zastąpić żadne inne instytucje, podkreślił Ojciec Święty w Guayaquil. <NDzien 2015, 157: 9>”* został przekształcony do postaci XML przestawionej na rysunku 3.33.

```
<quote>
 <quotetext>Rodzina jest wielkim bogactwem społecznym, którego
nie mogą zastąpić żadne inne instytucje, podkreślił Ojciec Święty
w Guayaquil.</quotetext>
 <source>
  <authors/>
  <periodical>NDzien</periodical>
  <year_from>2015</year_from>
  <year_to>2015</year_to>
  <issue>157</issue>
  <page>9</page>
 </source>
</quote>
```

Rys. 3.33. Przetworzony wiersz według wzorca 3.32 (przykład numer 1).

Wiersz o treści *„& przyrządowe media <WarszG 2014, 47: 20>”* został przekształcony do postaci XML przestawionej na rysunku 3.34.

```
<quote>
 <quotetext>przyrządowe media</quotetext>
 <source>
  <authors/>
  <periodical>WarszG</periodical>
  <year_from>2014</year_from>
  <year_to>2014</year_to>
  <issue>47</issue>
  <page>20</page>
 </source>
</quote>
```

Rys. 3.34. Przetworzony wiersz według wzorca 3.32 (przykład numer 2).

Wiersz o treści *„& W Polsce udział płac w PKB jest aż dwukrotnie niższy niż w USA. Inne sukcesy transformacji to 40 proc. młodych ludzi pracujących na umowach śmieciowych /.../. <GPolC 2015, 152: 2>”* został przekształcony do postaci XML przestawionej na rysunku 3.35.

```
  <quote>
   <quotetext>W Polsce udział płac w PKB jest aż dwukrotnie niższy
niż w USA. Inne sukcesy transformacji to 40 proc. młodych ludzi
pracujących na umowach śmieciowych /.../.</quotetext>
   <source>
    <authors/>
    <periodical>GPolC</periodical>
    <year_from>2015</year_from>
    <year_to>2015</year_to>
    <issue>152</issue>
    <page>2</page>
   </source>
  </quote>
```

Rys. 3.35. Przetworzony wiersz według wzorca 3.32 (przykład numer 3).

Wiersz o treści *„& Od początku 1988 roku nic nie zakłóca odbioru zachodnich radiostacji polskojęzycznych. <Rzeczpospolita 1988, 69: 5>”* został przekształcony do postaci XML przestawionej na rysunku 3.36.

```
<quote>
 <quotetext>Od początku 1988 roku nic nie zakłóca odbioru
zachodnich radiostacji polskojęzycznych.</quotetext>
 <source>
  <authors/>
  <periodical>Rzeczpospolita</periodical>
  <year_from>1988</year_from>
  <year_to>1988</year_to>
  <issue>69</issue>
  <page>5</page>
 </source>
</quote>
```

Rys. 3.36. Przetworzony wiersz według wzorca 3.32 (przykład numer 4).

Dane dotyczące utworów Pozyskanie danych o określonym tytule utworu następuje według zdefiniowanego wzorca przedstawionego na rysunku 3.37.

```
REGEX_QUOTE_2 =
    r'^[&] ' + REGEX_TEXT_QUOTE_LT
    + REGEX_TEXT_AUTHOR_INITIAL_SURNAME
    + r'[,] ' + REGEX_TEXT_TITLE_COMMA + r' '
    + REGEX_TEXT_ADDRESS + r' (?:etc[.] )?[:] '
    + REGEX_TEXT_PUBLISHER_COMMA + r' '
    + REGEX_TEXT_SOURCE_YEAR + r'[:] '
    + REGEX_TEXT_SOURCE_PAGE + r'[>]$'
```

Rys. 3.37. Wzorzec dopasowujący wszystkie dane zawierające z tytułem utworu.

Przykłady przetworzenia danych przedstawione są na rysunkach od 3.38 do 3.41.

Wiersz o treści *„& Pewnego poobiedzia, z nieodłączną wyżlicą Azą u nogi, zajrzał do żony. <G. Pauszer-Klonowska, Pani Eliza. Opowieść o Elizie Orzeszkowej, Warszawa : Nasza Księgarnia, 1958: 93>”* został przekształcony do postaci XML przestawionej na rysunku 3.38.

```
<quote>
 <quotetext>Pewnego poobiedzia, z nieodłączną wyżlicą Azą u nogi,
zajrzał do żony.</quotetext>
 <source>
  <authors>
   <author>
    <initial>G.</initial>
    <surname>Pauszer-Klonowska</surname>
   </author>
  </authors>
  <title>Pani Eliza. Opowieść o Elizie Orzeszkowej</title>
  <address>Warszawa</address>
  <publisher>Nasza Księgarnia</publisher>
  <year_from>1958</year_from>
  <year_to>1958</year_to>
  <page>93</page>
 </source>
</quote>
```

Rys. 3.38. Przetworzony wiersz według wzorca 3.37 (przykład numer 1).

Wiersz o treści *„& Te moje zapiski – co to jest? Dziennik, niedziennik, codziennik, poniedzielnik i wtornik, a nawet niekiedy nocnik (od pisania nocą) esej, eseidło, powieść, powieścidełko? <A. Nasiłowska, Czteroletnia filozofka, Kraków : Wyd. Literackie, 2004: 133>”* został przekształcony do postaci XML przestawionej na rysunku 3.39.

```
<quote>
 <quotetext>Te moje zapiski - co to jest? Dziennik, niedziennik,
codziennik, poniedzielnik i wtornik, a nawet niekiedy nocnik (od
pisania nocą) esej, eseidło, powieść, powieścidełko?</quotetext>
 <source>
  <authors>
   <author>
    <initial>A.</initial>
    <surname>Nasiłowska</surname>
   </author>
  </authors>
  <title>Czteroletnia filozofka</title>
  <address>Kraków</address>
  <publisher>Wyd. Literackie</publisher>
  <year_from>2004</year_from>
  <year_to>2004</year_to>
  <page>133</page>
 </source>
</quote>
```

Rys. 3.39. Przetworzony wiersz według wzorca 3.37 (przykład numer 2).

Wiersz o treści *„& Aż dziw, że wilkowi jeszcze smakuje... ta lisem podszyta owieczkowatość. <M. Pankowski, Ksiądz Helena. Wybór utworów dramatycznych, Kraków : Wyd. Literackie, 1996: 208>”* został przekształcony do postaci XML przestawionej na rysunku 3.40.

```
<quote>
 <quotetext>Aż dziw, że wilkowi jeszcze smakuje... ta lisem
podszyta owieczkowatość.</quotetext>
 <source>
  <authors>
   <author>
    <initial>M.</initial>
    <surname>Pankowski</surname>
   </author>
  </authors>
  <title>Ksiądz Helena. Wybór utworów dramatycznych</title>
  <address>Kraków</address>
  <publisher>Wyd. Literackie</publisher>
  <year_from>1996</year_from>
  <year_to>1996</year_to>
  <page>208</page>
 </source>
</quote>
```

Rys. 3.40. Przetworzony wiersz według wzorca 3.37 (przykład numer 3).

Wiersz o treści *„& budynki przysłupowo-zrębowe <K. R. Mazurski, Świeradów Zdrój i okolice, Warszawa : Sport i Turystyka, 1974: 64>”* został przekształcony do postaci XML przestawionej na rysunku 3.41.

```
<quote>
 <quotetext>budynki przysłupowo-zrębowe</quotetext>
 <source>
  <authors>
   <author>
    <initial>K. R.</initial>
    <surname>Mazurski</surname>
   </author>
  </authors>
  <title>Świeradów Zdrój i okolice</title>
  <address>Warszawa</address>
  <publisher>Sport i Turystyka</publisher>
  <year_from>1974</year_from>
  <year_to>1974</year_to>
  <page>64</page>
 </source>
</quote>
```

Rys. 3.41. Przetworzony wiersz według wzorca 3.37 (przykład numer 4).

Elementy pozyskane z przykładów (3.33-3.41) przedstawiają się następująco:

- element *quotetext* – zawierający treść cytatu,
- element *source*:
  - element *periodical* – zawierający informację o nazwie czasopisma,
  - element *title* – zawierający informację o tytule utworu,
  - element *publisher* – zawierający informację o nawie wydawcy,
  - element *address* – zawierający informację o adresie wydawcy,
  - element *issue* – zawierający informację o numerze czasopisma,
  - element *page* – zawierający informację o numerze strony, w którym wystąpiła główka hasła.

3.5.3. Niepoprawne dane

Ostatnią kategorią danych jest informacja o nieprzetworzeniu, spowodowana błędem danych. Błędy danych zostały uzupełnione o brakuję znaki, które pozwoliły poprawnie przetworzyć wiersze. Najczęstszymi błędami niedopasowania są:

- niepoprawne znaki końca wiersza (poprawnym znakiem końca wiersza jest znak > – większości), np.
  - *& Winni temu oczywiście centraliści i prusofile /.../. <Feldman W. ?*,
  - *& /.../ Rzymem miało być nowe państwo polsko-rosyjskie. <J. Krzyżanowski, Historia literatury polskiej. Alegoryzm – preromantyzm, wyd. 2*,
- brakujące znaki > (większości) na końcu wiersza, np.
  - *& /.../ normą jest platoniczność pierwszych młodzieńczych miłości /.../. <Kozakiewicz M. 1968: 13*,
  - *& /.../ pójdzie pan do kina. Albo na dziwki. <Szczepański J. J. 1982: 210.*

3.6. Odczyt danych z wykorzystaniem zapytań SQL

Przetworzone wiersze z rozdziału 3.5 w postaci XML zostały zapisane do postaci bazy danych. Przed zapisem danych został wygenerowany model bazy danych za pomocą biblioteki *Persistent*[2] w języku *Haskell*. Baza danych została utworzona w relacyjnym modelu z wykorzystaniem technologi *PostgreSQL* (2.3.4). Tabela zostaje wygenerowana z przygotowanego szablonu, który zawiera kolejno informację o nazwie tabeli[3] oraz nazwie pól[4] z typem danych wypisanych w następnych wierszach.

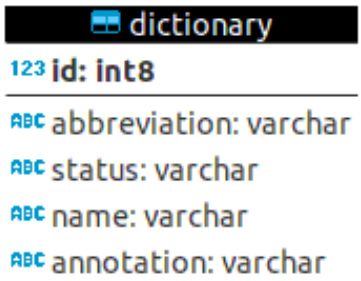
Do utworzenia tabeli o nazwie *Dictionary* z polem typu tekstowego o nazwie *abbreviation* oraz opcjonalnymi polami typu tekstowego o nazwach *status*, *name* oraz *annotation*. Niezbędne jest utworzenie szablonu przedstawionego na listingu 3.4.

```
Dictionary
    abbreviation Text
    status Text Maybe
    name Text Maybe
    annotation Text Maybe
```

Listing 3.4. Szablon tabeli *Dictionary* w bibliotece *Haskell Persistent*.

[2]https://www.yesodweb.com/book/persistent
[3]Nazwa tabeli musi rozpoczynać się od wielkiej litery.
[4]Nazwa pola w tabeli musi rozpoczynać się od małej litery.

Oczekiwanym wynikiem jest tabela o graficznym schemacie przedstawionym na rysunku 3.42.



Rys. 3.42. Graficzny schemat tabeli *Dictionary*.

Dostępne typy danych dla tabel w bibliotece *Haskell Persistent* przedstawia tabela 3.1.

Typ	**Opis**	**Przykład**
Maybe	Opcjonalność wystąpienia pola	*status Text Maybe*
Text	Tekstowy typ danych	"Przykładowy tekst.", "Tekst 123 abc"
Int	Liczbowy typ danych (liczby całkowite)	-980, -25, 0, 4, 123456
Double	Liczbowy typ danych (liczby rzeczywiste)	-123.456, 0.0, 12.0, 463346.12424
Bool	Logiczny typ danych	*True*, *False*
UTCTime	Czasowy typ danych	2018-03-01 12:18:15.198 UTC

Tab. 3.1. Dostępne opcje dla szablonu tabeli w bibliotece *Haskell Persistent*.

Klucze podstawowe tworzone są automatycznie jako identyfikator, który generowany jest podczas dodawania rekordu do tabeli. Nazwa klucza podstawowego zapisywana jest jako kolumna o nazwie *Id*, np. dla tabeli *Dictionary* (Listing 3.4) utworzona została kolumna o nazwie *Id*.

Klucze obce tworzy się poprzez podanie nazwy kolumny[5] oraz referencję

[5]Nazwa pola w tabeli musi rozpoczynać się od małej litery.

do tabeli[6] połączoną z nazwą pola[7]. Utworzony klucz obcy o nazwie *dictionaryId* o referencji do tabeli *Dictionary* i polu *Id* przedstawia listing 3.5.

```
dictionaryId DictionaryId
```

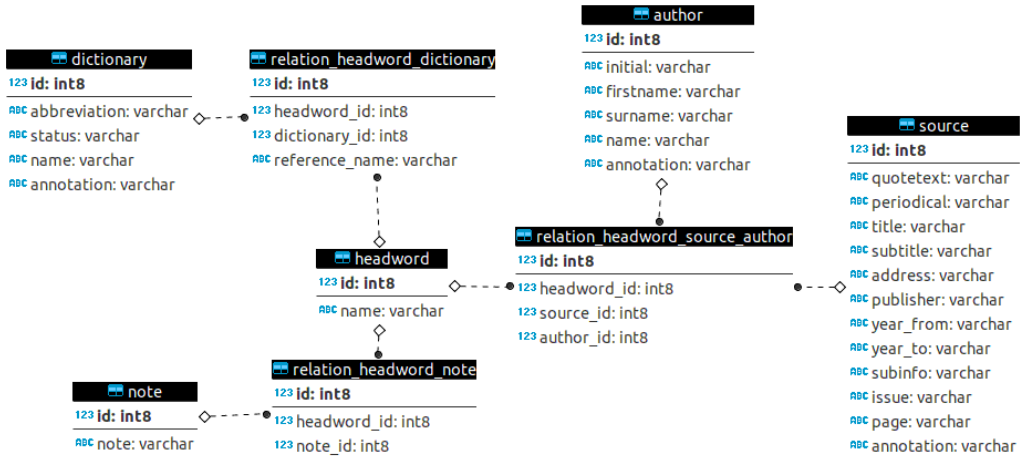
Listing 3.5. Szablon klucza obcego w bibliotece *Haskell Persistent*.

Tworzenie ograniczeń dla wartości (w celu uniknięcia duplikatów) w tabeli następuje przez wybór nazwy ograniczenia[8] oraz nazwę pól[9] oddzielonych od siebie spacją[10]. Utworzone ograniczenie o nazwie *UniqueAbbrevationStatus* dla pola *abbrevation* oraz pola *status* przedstawia listing 3.6.

```
UniqueAbbrevationStatus abbrevation status
```

Listing 3.6. Szablon ograniczenia wartości w bibliotece *Haskell Persistent*.

Graficzny schemat bazy danych dla słownika bibliograficznego języka polskiego przedstawia rysunek 3.43.



Rys. 3.43. Schemat bazy danych słownika bibliograficznego języka polskiego.

[6]Nazwa tabeli musi rozpoczynać się od wielkiej litery.
[7]Nazwa pola z tabeli musi rozpoczynać się od wielkiej litery.
[8]Nazwa ograniczenia musi rozpoczynać się od wielkiej litery.
[9]Nazwa pola powinna być taka sama jak nazwa pola w szablonie tabeli.
[10]Możliwe jest podanie jednej lub więcej nazw pól.

Szablon bazy danych w bibliotece *Haskell Persistent* znajduje się w załączniku 3.14. W słowniku bibliograficznym języka polskiego wyróżnia się dwa rodzajami tabel:

1. tabele źródłowe (tabele zawierające przetworzone dane):
   (a) *headword* – tabela zawierająca wszystkie główki hasła,
   (b) *dictionary* – tabela zawierająca wszystkie dane słownikowe,
   (c) *source* – tabela zawierająca wszystkie dane o utworach,
   (d) *author* – tabela zawierająca wszystkie dane o autorach,
   (e) *note* – tabela zawierająca wszystkie nieprzetworzone wiersze,
2. tabele łączące (rozpoczynające się od nazwy *relation*):
   (a) *relation_headword_dictionary* – tabela łącząca tabele *headword* i *dictionary* – połączenie danych słownikowych z główką hasła,
   (b) *relation_headword_source_author* – tabela łącząca tabele *headword*, *source* i *author* – połączenie informacji o utworze i autorze z główką hasła,
   (c) *relation_headword_note* – tabela łącząca tabele *headword* i *note* – połączenie nieprzetworzonych wierszy z główką hasła.

3.6.1. Przykładowe zapytania SQL

Odczytanie wszystkich przetworzonych główek hasła posortowanych malejąco z bazy danych następuje przez wykonanie zapytanie SQL przedstawia listing 3.7.

```
SELECT name FROM headword ORDER BY name ASC;
```

Listing 3.7. Zapytanie SQL pobierające wszystkie główki hasła posortowane malejąco.

Wynikiem zapytania (listing 3.7) są główki hasła przedstawione w tabeli 3.2.

name
1-funtowy
2-rzędny
DACA
dach
dacha
dacharz
Dachau
...

Tab. 3.2. Efekt zapytania SQL dla wszystkich główek hasła posortowanych malejąco.

Odczytanie główek hasła rozpoczynających się od frazy *pies*, które wystąpiły w słowniku (status dostępności równy *+*) o nazwie skrótowej *Ds* (słownik W. Doroszewskiego z roku 1969) przedstawia listing 3.8.

```
SELECT h.name FROM headword h
  INNER JOIN relation_headword_dictionary rhd
    ON rhd.headword_id = h.id
  INNER JOIN dictionary d
```

```
    ON d.id = rhd.dictionary_id
WHERE h.name ~ '^pies'
  AND d.status = '+'
  AND d.abbreviation = 'Ds';
```

Listing 3.8. Zapytanie SQL pobierające główki rozpoczynające się od frazy *pies*, które wystąpiły w słowniku (status dostępności równy *+*) o nazwie skrótowej *Ds* (słownik W. Doroszewskiego z roku 1969).

Wynikiem zapytania (listing 3.8) jest główka hasła przedstawiona w tabeli 3.3.

name
piesiec[11]

Tab. 3.3. Efekt zapytania SQL dla główek hasła rozpoczynających się od frazy *pies*, które wystąpiły w słowniku (status dostępności równy *+*) o nazwie skrótowej *Ds* (słownik W. Doroszewskiego z roku 1969).

Odczytanie wszystkich cytatów, dla których główki hasła rozpoczynają się od litery *r* oraz autorem cytatu jest nazwisko *Mickiewicz* lub *Słowacki* przedstawia listing 3.9.

```
SELECT h.name, s.quotetext, a.surname FROM headword h
  INNER JOIN relation_headword_source_author rhsa
    ON rhsa.headword_id = h.id
  INNER JOIN source s
    ON s.id = rhsa.source_id
  INNER JOIN author a
    ON a.id = rhsa.author_id
WHERE h.name ~ '^r'
```

[11] piesiec – lis polarny

```
  AND s.quotetext IS NOT NULL
  AND ( a.surname = 'Mickiewicz'
      OR a.surname = 'Słowacki' );
```

Listing 3.9. Zapytanie SQL pobierające cytaty, których autorem jest *Mickiewicz* lub *Słowacki*, dla główek hasła rozpoczynających się od litery *r*.

Wynikiem zapytania (listing 3.9) są dane przedstawione w tabeli 3.4.

name	**quotetext**	**surname**
rozbić	Trzy razy księżyc odmienił się złoty, Jak na tym piasku rozbiłem namioty.	Słowacki
rzymsko-grecki	państwo rzymsko-greckie	Mickiewicz

Tab. 3.4. Efekt zapytania SQL dla cytatów, dla których główki hasła rozpoczynają się od litery *r* oraz autorem cytatu jest nazwisko *Mickiewicz* lub *Słowacki*.

Odczytanie główek hasła, dla których cytat wystąpił w czasopiśmie o nazwie *Płomyczek*, posortowanych po dacie malejąco, przedstawia listing 3.10.

```
SELECT h.name, s.quotetext, s.year_from FROM headword h
  INNER JOIN relation_headword_source_author rhsa
    ON rhsa.headword_id = h.id
  INNER JOIN source s
    ON s.id = rhsa.source_id
WHERE s.periodical = 'Płomyczek'
  AND s.quotetext IS NOT NULL
ORDER BY s.year_from ASC;
```

Listing 3.10. Zapytanie SQL pobierające główki hasła, dla których cytat wystąpił w czasopiśmie o nazwie *Płomyczek*, posortowanych po dacie malejąco.

Wynikiem zapytania (listing 3.10) są dane przedstawione w tabeli 3.5.

name	quotetext	year
prymus	Janek był prymusem, ja ledwie wyciągałem na trójki.	1982
dziczyć się	Pod brzozą stało brunatnoczarne wielkie dziczysko i z wielkim apetytem wyżerało grzyby z mojego koszyka...	1982
dwójołap	/.../ a to wynaleziono szczotkę do... zapamiętywania, a to znowu przyrząd zwany dwójołapem, a to urządzenie, które było talentołapem.	1983

Tab. 3.5. Efekt zapytania SQL dla główek hasła, dla których cytat wystąpił w czasopiśmie o nazwie *Płomyczek*, posortowane po dacie malejąco.

Odczytanie pięciu najstarszych główek hasła, które wystąpiły w dowolnym utworze (nie czasopiśmie) przedstawia listing 3.11.

```
SELECT h.name, s.title, s.year_from FROM headword h
  INNER JOIN relation_headword_source_author rhsa
    ON rhsa.headword_id = h.id
  INNER JOIN source s
    ON s.id = rhsa.source_id
WHERE s.title IS NOT NULL
ORDER BY s.year_from ASC, h.name ASC
LIMIT 5;
```

Listing 3.11. Zapytanie SQL pobierające pięć najstarszych główek hasła, które wystąpiły w dowolnym utworze (nie czasopiśmie).

Wynikiem zapytania (listing 3.11) są dane przedstawione w tabeli 3.6.

name	title	year
polerowność	Dzieje narodu litewskiego. T. 8	1840
przekopski	Dzieje narodu litewskiego. T. 8	1840
przed-splinowy	Pisma przed-ślubne i przed-splinowe. T. 1	1841
poezyjka	Piśmiennictwo polskie od czasów najdawniejszych aż do roku 1830. T. 1	1851
rozumkowanie	Piśmiennictwo polskie od czasów najdawniejszych aż do roku 1830. T. 1	1851

Tab. 3.6. Efekt zapytania SQL dla pięciu najstarszych główek hasła, które wystąpiły w dowolnym utworze (nie czasopiśmie).

3.7. Korekta niepoprawnych danych

Zapisane w dane w postaci bazy danych zostały sprawdzone pod względem podobnych informacji. Podobnymi informacji nazywamy dwie informacje, które różnią się od siebie kilkoma znakami. Dzięki wykryciu podobnych informacją możliwe jest wychwycenie błędów danych, takich jak:

1. literówki,
2. różnice wielkości liter,
3. skróty zapisu.

3.7.1. Odległość Hamminga

Pierwszą metodą wykorzystaną w wyszukaniu błędów jest odległość Hamminga, która odpowiada na pytanie, na ilu miejscach różnią się dwa ciągi o takiej samej długości. Przykładem wykrycia różnic są:

1. Dane różniące się na jednym miejscu (znakiem), np.

   (a) Metafizyczno**œ**ć języka

   (b) Metafizyczno**ś**ć języka

   Różnica dotyczy znaków **œ** oraz **ś** (literówka).

2. Dane różniące się na jednym miejscu (znakiem), np.

   (a) „Po **p**rostu”

   (b) „Po **P**rostu”

   Różnica dotyczy znaków **p** oraz **P** (różnica wielkości liter).

3. Dane różniące się na trzech miejscach (znakach), np.

   (a) Historyczna i wsp**ol**czesna fonologia j**ê**zyka polskiego

   (b) Historyczna i wsp**ół**czesna fonologia j**ę**zyka polskiego

   Różnica dotyczy par znaków: **o** - **ó**, **l** - **ł**, **ê** - **ę**.

Efektem podobnych danych są poprawne dane, które różnią się kilkoma znakami. Takimi przykładami są dane dotyczące:

1. nazwisk:

   (a) Potock**a** oraz Potock**i** (różnica dla znaków **a** - **i**),

   (b) Mi**c**kiewicz oraz Mi**n**kiewicz (różnica dla znaków **c** - **n**),

2. skrótów:

   (a) „Wyd. Nauk. **P**WN” oraz „Wyd. Nauk. **D**WN” (różnica dla znaków **P** - **D**),

(b) **G**Pol oraz **N**Pol (różnica dla znaków **G** - **N**).

Odległość Hamminga pozwala wykryć podobne dane, które w części przypadków są błędami danych. Błędy danych zostały zamienione na poprawne odpowiedniki. Pominięcie poprawnych danych podobnych względem siebie nastąpiło przez porównanie informacji z listą dostępnych nazwisk, czy listą dostępnych skrótów dla „Słownika bibliograficznego języka polskiego".

3.7.2. Odległość Levenshteina

Drugą metodą wykorzystaną w wyszukiwaniu błędów jest odległość Levenshteina, która odpowiada na pytanie, ile operacji na danej źródłowej należy wykonać, aby otrzymać daną docelową. Operacjami takimi są:

1. zamiana znaku,
2. dodanie nowego znaku,
3. usunięcie istniejącego znaku,
4. transpozycja znaków.

Przykładem wykrycia różnic są:

1. Dane różniące się znakiem cudzysłowu:
   (a) **„**Płomyczek**"**
   (b) Płomyczek

   W danej źródłowej należy usunąć 2 znaki: **„** oraz **"**, aby otrzymać daną docelową.
2. Dane różniące się znakiem łącznika:

(a) „Workshop. Gazeta Autorów”

(b) „Workshop – Gazeta Autorów”

Z danej źródłową należy zamienić znak kropki na znak odstępu (spacji) oraz dodać znak –, aby otrzymać daną docelową.

3. Dane z pominiętym fragmentem tekstu:

(a) „Polska myśl glottodydaktyczna. Wybór artykułów z zakresu glottodydaktyki ogólnej”

(b) „Polska myśl glottodydaktyczna **1945-1975**. Wybór artykułów z zakresu glottodydaktyki ogólnej”

Do danej źródłową należy dodać fragment tekstu: **1945-1975**, aby otrzymać daną docelową.

Odległość Levenshteina pozwala wykryć więcej błędnych danych. Tak jak miało to miejsce w odległości Hamminga dane zostały zastąpione poprawnymi odpowiednikami. Wyselekcjonowanie tylko tych danych, które zawierają błędną informację nastąpiło dzięki porównaniu z poprawnymi danymi „Słownika bibliograficznego języka polskiego”.

Podsumowanie

Proces automatycznej konwersji przeprowadzony został w kilku etapach. Pierwszym kluczowym elementem projektu był wybór formatu danych wejściowych. Zapisanie danych w postaci formatu HTML umożliwiło zamianę plików programu *Microsoft Word* do ustrukturalnego schematu danych. Dzięki programowi *tidy-html5* możliwe było oczyszczenie plików w formacie HTML ze zbędnej informacji. Kolejnym etapem automatycznej konwersji była zamiana wierszy danych z formatu HTML do jednolitego formatu XML dzięki zdefiniowanym wyrażeniom regularnym. Utworzenie bazy danych możliwe było dzięki wykorzystaniu biblioteki *Yesod* w języku programowania *Haskell*. Ostatnim elementem całego procesu było wyszukanie niepoprawnych informacji i zastąpienie poprawnymi odpowiednikami.

Efektem pracy jest ustrukturyzowana informacja zapisana w formacie bazy danych. Dzięki przetworzonym danych możliwe jest odpowiedzenie na zapytania typu: „Podaj przymiotniki znane w latach 1901-1914", „Podaj słowa poświadczone cytatami z dzieł Juliusza Słowackiego", czy „Podaj wszystkie hasła, które wystąpiły w czasopiśmie Płomyczek w latach 1980-1985". Utworzenie publicznej strona WWW umożliwia zaprezentowanie danych w czytelny sposób, niż surowych danych zwróconych przez system bazy danych.

Automatyczna konwersja nieustrukturyzowanych danych do postaci ustrukturyzowanie zapewnia jednolitą postać danych. Dzięki jasno zdefiniowane-

mu schematowi zapisu danych możliwe jest wykonanie analizy oraz korekty danych. Ogólny dostęp do danych zapewnia powszechniejszy dostęp do informacji. Przetworzone dane mogą zostać przestawione w czytelny sposób dla użytkownika, niż przedstawienie zestawu danych zapisanych w osobnych plikach.

Spis rysunków

Spis tabel

Listingi kodu

Załączniki

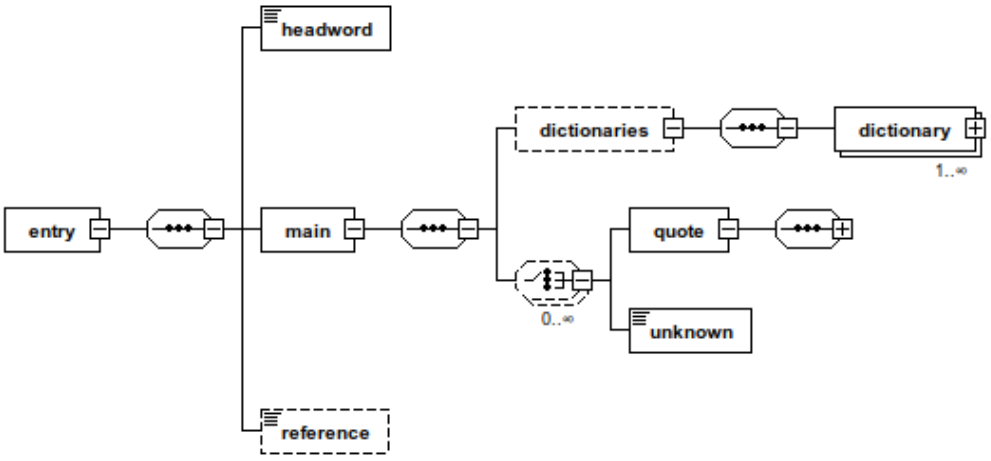
Konfiguracja programu tidy-html5:

```
bare: false
ascii-chars: false
clean: true
coerce-endtags: true
doctype: html5
drop-empty-elements: true
drop-empty-paras: true
output-html: true
hide-comments: true
literal-attributes: false
word-2000: true
show-errors: 0
show-info: false
show-warnings: false
char-encoding: utf8
input-encoding: utf8
output-encoding: utf8
force-output: true
enclose-block-text: true
```

```
quiet: true
wrap: 0
indent: false
indent-spaces: 1
```

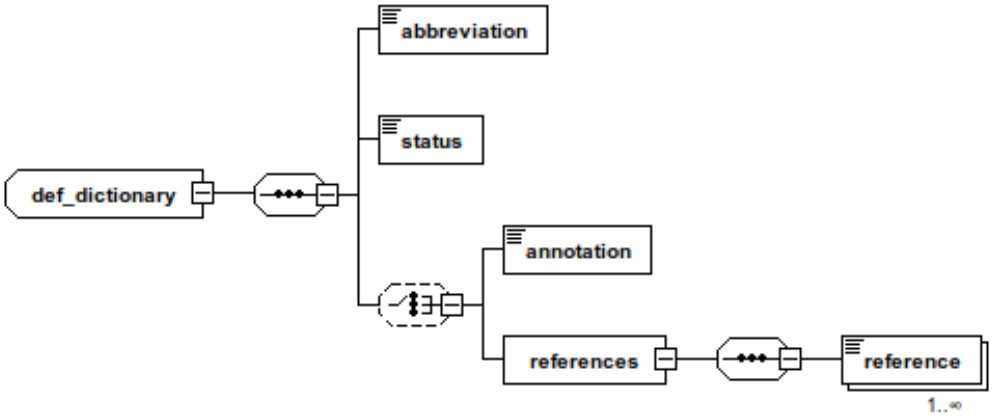
Listing 3.12. Konfiguracja programu tidy-html5.

Diagram pliku wyjściowego (modelu bazy danych) w formacie XML (diagram został podzielony na osobne części dla elementu *dictionary* – 3.45 oraz elementu *quote* – 3.46):



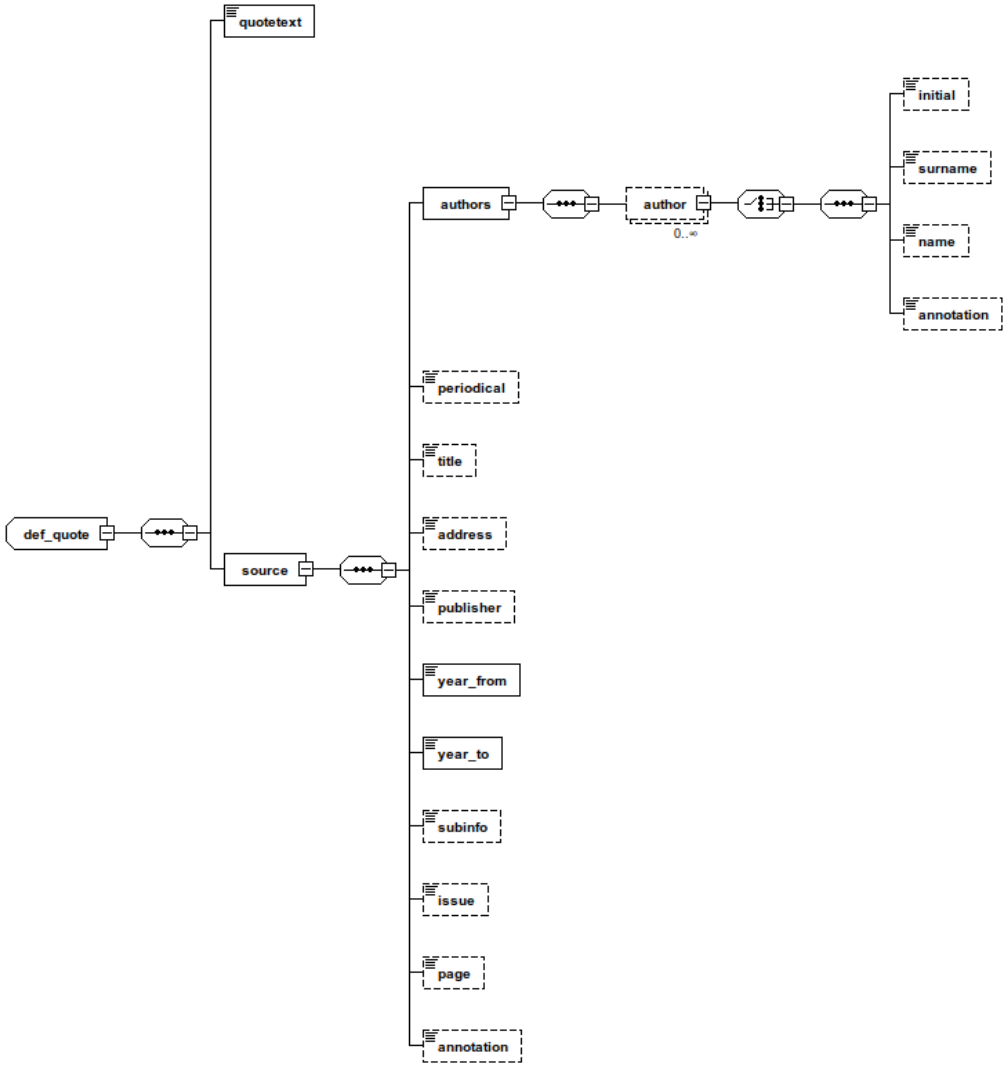
Rys. 3.44. Diagram pliku wyjściowego (modelu bazy danych) w formacie XML.

Definicja elementu *dicionary* dla diagramu pliku wyjściowego (modelu bazy danych) w formacie XML:



Rys. 3.45. Definicja elementu *dicionary* dla pliku wyjściowego (modelu bazy danych) w formacie XML.

Definicja elementu *quote* dla pliku wyjściowego (modelu bazy danych) w formacie XML:



Rys. 3.46. Definicja elementu *quote* dla pliku wyjściowego (modelu bazy danych) w formacie XML.

Schemat XSD pliku wyjściowego w formacie XML:

```
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definicaiotns -->
  <!-- definition: authors -->
  <xs:complexType name="def_authors">
    <xs:sequence>
      <xs:element type="def_author" name="author"
          minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- definition: author -->
  <xs:complexType name="def_author">
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:sequence>
        <xs:element type="xs:string" name="initial"
            minOccurs="1" maxOccurs="1"/>
        <xs:element type="xs:string" name="surname"
            minOccurs="1" maxOccurs="1"/>
        <xs:element type="xs:string" name="annotation"
            minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element type="xs:string" name="firstname"
            minOccurs="1" maxOccurs="1"/>
        <xs:element type="xs:string" name="surname"
```

```
            minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element type="xs:string" name="name" minOccurs="1"
            maxOccurs="1"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element type="xs:string" name="surname"
            minOccurs="1" maxOccurs="1"/>
        <xs:element type="xs:string" name="annotation"
            minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:choice>
  </xs:complexType>
  <!-- definition: reference -->
  <xs:complexType name="def_reference">
    <xs:sequence>
      <xs:element type="xs:string" name="reference"
          minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- definition: source -->
  <xs:complexType name="def_source">
    <xs:sequence>
      <xs:element type="def_authors" name="authors"
          minOccurs="1" maxOccurs="1"/>
      <xs:element type="xs:string" name="periodical"
```

```
        minOccurs="0" maxOccurs="1"/>
    <xs:element type="xs:string" name="title" minOccurs="0"
        maxOccurs="1"/>
    <xs:element type="xs:string" name="address"
        minOccurs="0" maxOccurs="1"/>
    <xs:element type="xs:string" name="publisher"
        minOccurs="0" maxOccurs="1"/>
    <xs:element type="xs:string" name="year_from"
        minOccurs="1" maxOccurs="1"/>
    <xs:element type="xs:string" name="year_to"
        minOccurs="1" maxOccurs="1"/>
    <xs:element type="xs:string" name="subinfo"
        minOccurs="0" maxOccurs="1"/>
    <xs:element type="xs:string" name="issue" minOccurs="0"
        maxOccurs="1"/>
    <xs:element type="xs:string" name="page" minOccurs="0"
        maxOccurs="1"/>
    <xs:element type="xs:string" name="annotation"
        minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<!-- definition quote -->
<xs:complexType name="def_quote">
  <xs:sequence>
    <xs:element type="xs:string" name="quotetext"
        minOccurs="1" maxOccurs="1"/>
    <xs:element type="def_source" name="source"
```

```
        minOccurs="1" maxOccurs="1">
    </xs:element>
  </xs:sequence>
</xs:complexType>
<!-- definition dictionary -->
<xs:complexType name="def_dictionary">
  <xs:sequence>
    <xs:element type="xs:string" name="abbreviation"
        minOccurs="1" maxOccurs="1"/>
    <xs:element type="xs:string" name="status" minOccurs="1"
        maxOccurs="1"/>
    <!-- optional annotation or reference -->
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element type="xs:string" name="annotation"
          minOccurs="1" maxOccurs="1"/>
      <xs:element type="def_reference" name="references"
          minOccurs="1" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!-- definition dictionaries -->
<xs:complexType name="def_dictionaries">
  <xs:sequence>
    <xs:element type="def_dictionary" name="dictionary"
        minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

```
<!-- end definicaiotns -->
<!-- main -->
<xs:element name="entry">
  <xs:complexType>
    <xs:sequence>
      <xs:element type="xs:string" name="headword"
         minOccurs="1" maxOccurs="1"/>
      <xs:element name="main" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="def_dictionaries"
               name="dictionaries" minOccurs="0"
               maxOccurs="1"/>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element type="def_quote" name="quote"
                 minOccurs="1" maxOccurs="1"/>
              <xs:element type="xs:string" name="unknown"
                 minOccurs="1" maxOccurs="1"/>
            </xs:choice>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element type="xs:string" name="reference"
         minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
</xs:schema>
```

Listing 3.13. Schemat XSD pliku wyjściowego w formacie XML.

Schemat bazy danych w bibliotece *Haskell Persistent*:

```
--Headword
Headword
    name Text
    UniqueName name
--Author
Author
    initial Text Maybe
    firstname Text Maybe
    surname Text Maybe
    name Text Maybe
    annotation Text Maybe
--Source
BibSource sql=source
    quotetext Text Maybe
    periodical Text Maybe
    title Text Maybe
    subtitle Text Maybe
    address Text Maybe
    publisher Text Maybe
    yearFrom Text Maybe
    yearTo Text Maybe
    subinfo Text Maybe
    issue Text Maybe
```

```
    page Text Maybe
    annotation Text Maybe
--Dictionary
Dictionary
    abbreviation Text
    status Text Maybe
    name Text Maybe
    annotation Text Maybe
--Note
Note
    note Text
    UniqueNote note

--Relation
RelationHeadwordDictionary
    headwordId HeadwordId
    dictionaryId DictionaryId
    referenceName String Maybe
RelationHeadwordSourceAuthor
    headwordId HeadwordId
    sourceId BibSourceId
    authorId AuthorId Maybe
RelationHeadwordNote
    headwordId HeadwordId
    noteId NoteId
    UniqueRelationHeadwordNote headwordId noteId
```

Listing 3.14. Schemat bazy danych w bibliotece *Haskell Persistent*.

Hierarchia elementów dla przetworzonego modelu XML:

Nazwa pozyskanej grupy	**Level***	**Nazwa elementu w pliku XML**	**Opis elementu**
-	0	entry	główny element XML
-	1	headword	nazwa główki hasła
-	1	main	lista wszystkich przetworzonych wierszy**
-	2	dictionaries	dane dotyczące danych słownikowych
-	3	dictionary	pojedyncza dana słownikowa
abbr	4	abbreviation	nazwa skrótowa słownika
status	4	status	status słownika
dict_annotation	4	annotation	adnotacja słownika
-	4	references	lista powiązanych główek hasła
dict_reference	5	reference	powiązana główka hasła
-	2	quote	dane dotyczące utworu
quote	3	quotetext	treść cytatu
-	3	source	informacje o utworze
-	4	authors	lista autorów
-	5	author	informacje o autorze
initial	6	initial	inicjały imienia
surname	6	surname	nazwisko
name	6	name	pseudonim artystyczny
author_annotation	6	annotation	adnotacje o autorze
periodical	4	periodical	nazwa czasopisma
title	4	title	tytuł utworu
address	4	address	miejsce wydania
publisher	4	publisher	nazwa wydawcy
year	4	year_from	data wydania
year	4	year_to	data wydania
subinfo	4	subinfo	kwartał roku
issue	4	issue	numer czasopisma
page	4	page	numer strony
source_annotation	4	annotation	adnotacje do utworu
-	2	unknown	nieprzetworzony wiersz
-	1	reference	nazwa powiązanej główki hasła

Tab. 3.7. Hierarchia elementów dla przetworzonego modelu XML.

∗ Informacja jak głęboko znajduje się element.
∗∗ W tym wiersze, które nie udało się przetworzyć.

Sposób mapowania przetworzonego pliku XML do bazy danych:

Nazwa elementu z pliku XML	**Nazwa tabeli**	**Nazwa pola w tabeli**
dictionary.abbreviation	Dictionary	abbreviation
dictionary.status	Dictionary	status
dictionary.annotation	Dictionary	annotation
references.reference	RelationHeadwordDictionary	referenceName
author.initial	Author	initial
author.firstname	Author	firstname
author.surname	Author	surname
author.name	Author	name
author.annotation	Author	annotation
quote.quotetext	Source	quotetext
source.periodical	Source	periodical
source.title	Source	title
source.address	Source	address
source.publisher	Source	publisher
source.year	Source	yearFrom
source.year	Source	yearTo
source.subinfo	Source	subinfo
source.issue	Source	issue
source.page	Source	page
source.annotation	Source	annotation

Tab. 3.8. Sposób mapowania przetworzonego pliku XML do bazy danych.