Grzegorz Myrda, Izabela Karsznia, Piotr Kulicki

Metodyka gromadzenia i harmonizacji danych

Uwagi wstępne

Opisana w poprzednim rozdziale ontologicznie ufundowana struktura bazy danych służy do gromadzenia informacji krytycznych na temat jednostek osadniczych oraz administracyjnych w ich rozwoju historycznym. Można wyróżnić dwa główne scenariusze gromadzenia danych historycznych w takiej strukturze:

1. zasilenie masowe z już istniejących zasobów geoinformacyjno-historycznych, np. przygotowanych w ramach projektu badawczego
2. wprowadzanie danych jednostkowych ze źródeł historycznych w celu uzupełnienia lub aktualizacji istniejącego zasobu, np. crowdsourcing, citizen science

Oba scenariusze różnią się nie tylko ilością materiału i liczbą rekordów, ale także ich jakością oraz sposobem przygotowania. W obu przypadkach inna charakterystyka pracy dotyczy jednostek osadniczych, które mają geometrię punktową (point), a odmienna jednostek administracyjnych, których geometria ma charakter obszarowy (polygon).

W pierwszym przypadku należy brać pod uwagę fakt, że dane historyczne nie są gromadzone w jednolity i spójny sposób. Każdy zbiór, posiada odmienną strukturę czy format. Należy brać pod uwagę zarówno różne modele danych jak też zróżnicowane formy ich zapisu: od prostych arkuszy xls, przez dane w plikach tekstowych (np. csv), przez formaty bazodanowe (np. mdb) aż po formaty GIS (np. shp, gml). Przyjęte zostało założenie, że są to informacje krytyczne, które będą służyły bezpośrednio tworzeniu manifestacji cząstkowych nazwy, położenia i typu dla miejscowości oraz jednostek administracyjnych. Cały proces przygotowania danych, ich weryfikacji oraz ujednolicenia odbywa się przed ich wykorzystaniem do zasilenia bazy danych ‘ontohgis’.

W drugim scenariuszu użytkownik systemu może wprowadzać dane bezpośrednio do bazy danych za pomocą aplikacji desktopowych lub dedykowanych aplikacji internetowych. W ramach projektu został przygotowany prototyp takiej aplikacji dla wprowadzania informacji dotyczących miejscowości (robocza nazwa: ontoform). Wymusza on gromadzenie danych w sposób dostosowany do architektury systemu ‘ontohgis’, który następnie ułatwia przetwarzanie i walidację wprowadzonych informacji oraz automatyzuje proces przekształcania informacji

historycznych o nazwach, typach czy położeniu miejscowości lub jednostki administracyjnej w odpowiednie manifestacje cząstkowe.

1. Dane masowe i zewnętrzne zbiory danych

Główną zaletą systemów *historical GIS* jest możliwość łączenia, przetwarzania i wizualizacji danych. Z tego względu jednym z najważniejszych aspektów jego budowania jest włączenie już dostępnych danych pochodzących z różnych źródeł oraz zbudowanie procedur dołączania nowych danych. Wyróżnić tu można dwa ich rodzaje. Po pierwsze uwzględniono informacje dotyczące współczesnej nam organizacji przestrzeni: współczesnych miejscowości, ich aktualnych nazw, charakteru i położenia oraz współczesnych podziałów terytorium Polski pozwalających wiązać z nimi informacje historyczne. W kwestii jednostek administracyjnych szczególnie ważne są jednostki najmniejsze, na podstawie których można odtworzyć z wystarczającą, na potrzeby tworzenia systemów geoinformacyjno-historycznych, dokładnością wszelkie historyczne podziały administracyjne.

Po drugie wzięto pod uwagę zasoby dotyczące historii Polski, w szczególności te, które są dobrze opracowane od strony historycznej i gotowe do umieszczenia w systemie komputerowym. W szczególności wykorzystano tu wcześniejsze prace zrealizowane w ramach zasłużonego dla geografii historycznej Polski Pracowni Atlasu Historycznego, aktualnie działającego jako Zakład Atlasu Historycznego Instytutu Historii PAN.

Najpełniejszym i najbardziej wiarygodnym źródłem współczesnych danych z terenu Polski są zasoby udostępniane przez Główny Urząd Geodezji i Kartografii (GUGiK). Urząd ten zobowiązany jest przepisami prawa do prowadzenia kilku ogólnodostępnych rejestrów związanych z różnego rodzaju obiektami geograficznymi. Rejestry te są udostępniane w postaci przetwarzalnej komputerowo, więc nadają się dobrze do wykorzystania w naszym projekcie.

W odniesieniu do miejscowości właściwym, choć nie jedynym zasobem1, jest Państwowy Rejestr Nazw Geograficznych (PRNG). Dane zostały pobrane ze strony http://codgik.gov.pl dział „Dane bez opłat": http://www.gugik.gov.pl/pzgik/dane-bez-oplat/dane-z-panstwowego-rejestru-nazw-geograficznych-prng (12 września 2016). Spośród rożnych dostępnych formatów wybrano format Shapefile: ftp://91.223.135.109/prng/PRNG_MIEJSCOWOSCI_SHP.zip. PRNG nie zapewnia w pełni informatywnej dokumentacji bazy danych, w której przechowywany jest rejestr. Specyfikację modelu pojęciowego PRNG zawiera za to Rozporządzenie Ministra Administracji i Cyfryzacji z dnia 14 lutego 2012 r. w sprawie państwowego rejestru nazw geograficznych (Dz.U. 2012 poz. 309) jako Załącznik nr 3.

---

1 Na temat zbiorów danych dotyczących jednostek osadniczych i administracyjnych w Polsce oraz ich jakości zob. tekst. J. Zielińskiego w niniejszym tomie.

Jeśli chodzi o jednostki administracyjne właściwym rejestrem jest tu Państwowy Rejestr Granic (PRG) i powierzchni jednostek podziałów terytorialnych kraju. W tym przypadku właściwą podstawą prawną zwierającą model pojęciowy rejestru jest Rozporządzenie Rady Ministrów z dnia 10 stycznia 2012 r. w sprawie państwowego rejestru granic i powierzchni jednostek podziałów terytorialnych kraju (Dz.U. z 2012 r. poz. 199). Model pojęciowy zawarty jest w Załączniku nr 1. do rozporządzenia. Minimalnymi jednostkami, które zostały wykorzystane w projekcie do oznaczania granic dawnych jednostek administracyjnych są obręby ewidencyjne. Obręby te ujęte są w PRG, a ich status prawny i podstawowe właściwości określa Rozporządzenie Ministra Rozwoju Regionalnego i Budownictwa z dnia 29 marca 2001 r. w sprawie ewidencji gruntów i budynków (Dz.U. z 2019 r. poz. 393). Dane z PRG zostały pobrane z lokalizacji http://www.gugik.gov.pl/pzgik/dane-bez-oplat/dane-z-panstwowego-rejestru-granic-i-powierzchni-jednostek-podzialow-terytorialnych-kraju-prg w formacie *Shapefile*. (9 maja 2017, v.15).

Jeśli chodzi o dane historyczno-geograficzne z obszaru współczesnej Polski rezultaty prac Pracowni Atlasu Historycznego wspomnianego w poprzedniej sekcji upubliczniane są w postaci Atlasu Historycznego Polski (AHP). Dane w formacie *Shapefile* zostały przekazane w ramach Zakładu Atlasu Historycznego Instytutu Historii PAN. Dane te są ogólnodostępne i można je uzyskać kontaktując się z pracownikami Zakładu lub pobierając ze strony atlasfontium.pl w postaci geobazy.

1.1. Jednostki osadnicze

1.1.1. Pierwotne zasilenie bazy danych

Do podstawowych informacji o miejscowościach zawartych w rejestrze PRNG prowadzonym przez GUGiK należą: nazwy miejscowości, przynależność do typu oraz lokalizacja – główna i pomocnicze. Rozumienie atrybutów dotyczących nazwy i typu miejscowości jest klarowne i naturalne, w związku z tym nie wymaga komentarza. Ograniczymy się więc do omówienia kwestii związanych z lokalizacjami. Lokalizacje miejscowości oznaczone są punktowo. Większości miejscowości przypisana jest tylko jedna, główna lokalizacja. Odpowiada ona umiejscowieniu umownego centralnego punku tej miejscowości. Znaczna część miejscowości posiada lokalizację dodatkową lub kilka lokalizacji dodatkowych. Taka dodatkowa lokalizacja przypisywana jest miejscowościom o charakterze rozproszonym przestrzennie, wskazując na punkty centralny części miejscowości oddalonych od głównego centrum miejscowości.

Wciągnięcie podstawowych danych PRNG do bazy danych projektu polegało na tym, że:

1. każdy rekord w PRNG odpowiadający głównej lokalizacji miejscowości (rodzaj reprezentacji: punkt centralny) miejscowości stworzył meta-obiekt w tabeli *VariableSettlements*,

2. każda nazwa główna zawarta w PRNG stworzyła manifestację cząstkową nazwy w tabeli *SettlementNames* datowaną na 2016 rok ('StartsAt', 'EndsAt'),
3. każda lokalizacja wyrażona zarówno przez punkt centralny jak i punkt dodatkowy (rodzaj reprezentacji) stworzyła manifestację cząstkową położenia geograficznego w tabeli *SettlementLocations* datowaną na 2016 rok ('StartsAt', 'EndsAt'),
4. każda nazwa typu miejscowości określona w PRNG (rodzaj obiektu), została zamieniona na identyfikator typu zgodny z ontologią 'ontohgis' i stworzyła manifestację cząstkową typu miejscowości w tabeli *SettlementTypes* datowaną na 2016 rok ('StartsAt', 'EndsAt'),
5. każda relacja miejscowości nadrzędnej i podrzędnej została przekształcona poprzez przypisanie miejscowościom nadrzędnym identyfikatorów i stworzyła manifestację mereologiczną w tabeli *SettlementMereologyLinks* datowaną na 2016 rok ('StartsAt', 'EndsAt').

Punkty 4 i 5, ze względu na przekształcenie źródłowego zbioru PRNG wymagają szerszego omówienia.

Ad. 4. Przed procedurą zasilenia bazy danych, typy miejscowości oraz ich charakterystyka opracowane w ramach projektu zostały zapisane w formacie OWL jako klasy obiektów (dostępne na stronie projektu ontohgis.pl). Jednym z elementów opisu każdej takiej klasy obiektu jest adnotacja 'ma obcy identyfikator', która odnosi się do klucza głównego typu miejscowości w bazie danych w tabeli *SettlementTypesDictionary*. Każda nazwa typu określona w PRNG po przeprowadzeniu weryfikacji czy już istnieje w systemie, została zamieniona na identyfikator typu, który następnie był wstawiany do manifestacji typu (w tabeli *SettlementTypes*: 'SettlementTypeIdentifiers'). Jeśli typ nie istniał lub nie odpowiadał żadnemu z już istniejących, to był dodawany do ontologii. W końcowym etapie procedury wszystkie typy występujące w PRNG znalazły swoją reprezentację w ontologii.

Ad. 5. W PRNG miejscowość może mieć określony obiekt nadrzędny, który też jest miejscowością. Problem polega na tym że reprezentacja relacji podrzędności/nadrzędności w PRNG jest niestandardowa: w polu reprezentującym obiekt nadrzędny wpisana jest bowiem nazwa miejscowości, a nie jak należałoby oczekiwać jednoznaczny identyfikator tej miejscowości. Powoduje to znaczny problem szczególnie w sytuacji, w której jest wiele miejscowości o tej samej nazwie. Wtedy bowiem nie jest jednoznacznie określone, o którą z miejscowości o wymienionej nazwie chodzi.

Żeby ten problem rozwiązać zdecydowano się wybrać właściwą miejscowość na podstawie jej lokalizacji, szukając miejscowości najbliższej w stosunku do miejscowości wyjściowej (podrzędnej). Reguły, warunki oraz przebieg procedury zostały opisane poniżej. Wyszukiwano wszystkie obiekty o odpowiedniej nazwie, a następnie odfiltrowano po takiej samej przynależności do jednostki podziału terytorialnego (czyli najczęściej gminy). Jeśli to nie dawało

rezultatów, wyszukiwana była miejscowość o tej nazwie w promieniu 50 km od analizowanej miejscowości (było kilka przypadków, w których obiektem nadrzędnym jest miejscowość z innej gminy/powiatu). Tym sposobem udało się odtworzyć hierarchię dla większości przypadków. Było trochę przypadków błędnie wpisanej w PRNG miejscowości nadrzędnej (nie istniejącej jako niezależna miejscowość), np. Podzagnańcze, które prawdopodobnie miało być Podzagnańszcze. Takie przypadki zostały pominięte.

Generalne reguły kwalifikacji jednostek osadniczych jako nadrzędne. Sytuacja nie pozostawiająca wątpliwości, czyli powodująca zaakceptowanie znalezionego obiektu jako obiektu nadrzędnego, to spełnianie następujących warunków:

1. Oprócz zgodnej nazwy obiekt nadrzędny musi być punktem centralnym i znajdować się w odległości nie większej niż zadana wartość graniczna, dla której przyjęto nadmiarową wartość 50 km. Warunki dodatkowe i tak nie pozwalają na akceptację zbyt odległych miejscowości. Zbyt mała odległość, np.10 km powodowała odrzucanie obiektów na obrzeżach dużych miast. Dla technicznych celów optymalizacji czasu działania skryptu do analizy pobieranych jest nie więcej niż 50 pobliskich miejscowości spełniających ten warunek. Obiekty analizowane są w kolejności od najbliższego.
2. Rodzajem obiektu nadrzędnego może być tylko *miasto* albo *wieś*.
3. Obiekt nadrzędny leży w ramach tej samej JPT (Jednostki Podziału Terytorialnego) co obiekt podrzędny (z pewnymi wyjątkami opisanymi dalej).
4. Spełniony jest warunek zależności typów (patrz pkt. *Warunek zależności typów*).

Pozostałe reguły.

1. Jeśli nazwa wykazana jako obiekt nadrzędny nie istnieje w najbliższej okolicy, to obiekt nie jest akceptowany, a w Rejestrze pojawia się wpis: *NOTFOUND;nie znaleziono obiektu nadrzędnego.*
2. Jeśli nazwa obiektu nadrzędnego jest taka sama jak podrzędnego i w zadanym promieniu nie ma innej miejscowości o takiej samej nazwie to obiekt nie jest akceptowany, a w Rejestrze pojawia się wpis: *ERROR;obiekt nie może należeć do siebie samego*.
3. Wyjątek od reguły położenia w ramach tej samej JPT: w dużych miastach istnieją przypadki osiedli (będących osobnymi jednostkami) należących do miasta, ale leżących w dzielnicy będącej osobną gminą. Np. Osiedle Medyków (dzielnica-gmina Podgórze) ->Kraków. Stąd obiektem nadrzędnym może zostać jednostka z innej JPT pod warunkiem że typ obiektu podrzędnego to: „*część miasta*“, a nadrzędnego *„miasto“*.

Warunek zależności typów. Warunek zależności typów sprawdza relację pomiędzy typem obiektu nadrzędnego i podrzędnego. Jeżeli żaden z warunków wyszczególnionych w

ramach tej reguły nie będzie spełniony, wtedy warunek zależności typów w całości nie będzie spełniony. Spełnienie jednego z warunków szczegółowych powoduje, że warunek zależności typów w całości jest spełniony (retvalue:=TRUE), czyli obiekt może zostać obiektem nadrzędnym przy jednoczesnym spełnianiu innych niezbędnych warunków lub nie (retvalue:=FALSE).

```
IF wh_t = 'miasto' AND part_t='miasto' THEN
retvalue := FALSE;
ELSIF wh_t = 'miasto' AND part_t='wieś' THEN
retvalue := TRUE;
ELSIF wh_t = 'wieś' AND part_t='miasto' THEN
retvalue := FALSE;
ELSIF wh_t = 'wieś' AND part_t='wieś' THEN
retvalue := FALSE;
ELSIF wh_t = 'wieś' THEN
retvalue := TRUE;
ELSIF wh_t = 'miasto' THEN
retvalue := TRUE;
END IF;
```

Log z procesu zasilenia. Możliwe wpisy w tabeli niezgodności obrazującej przebieg procesu zasilenia inicjalnego:

- NOTFOUND: oznacza że nazwa obiektu nadrzędnego nie występuje w rejestrze PRNG jako punkt centralny w zadanej odległości od obiektu podrzędnego (typ nie ma znaczenia). Występuje 208 takich przypadków. W tej sytuacji, nie jest dokonywany wpis dla tego obiektu do tabeli *ManifestationMereologicalLinks*.
- ERROR: oznacza że jedyną możliwością jest przypisanie miejscowości do samej siebie, co jest traktowane jako sytuacja błędna. Brak wpisu dla tego obiektu do tabeli SettlementMereologyLinks. Występuje 1484 takich przypadków.
- WARNING: oznacza że wystąpiły niejednoznaczności w czasie procesu zamiany nazwy w identyfikator. W tej sytuacji, nie jest dokonywany wpis dla tego obiektu do tabeli *SettlementMereologyLinks*. Występuje 4993 takich przypadków.
- OK;niezgodne JPT: oznacza warunkową zamianę nazwy w identyfikator. Następuje wpis dla tego obiektu do tabeli *SettlementMereologyLinks*. Występuje 907 takich przypadków.
- OK;wstawiono: oznacza zamianę nazwy w identyfikator. Następuje wpis dla tego obiektu do tabeli *SettlementMereologyLinks*. Występuje 83575 takich przypadków.

Wnioski. Rejestr PRNG prowadzony przez GUGIK zawiera dwie kategorie błędów jednoznacznych: tj. wpisy w polu nazwa jednostki nadrzędnej, nazw miejscowości, które nie istnieją w rejestrze oraz wpisy w postaci nazw identycznych jak nazwa obiektu podrzędnego przy jednoczesnym braku innego obiektu o takiej samej nazwie w okolicy. Czasami błędy te wynikają z błędów literowych, a czasami prawdopodobnie z tego, że nazwy te były wprowadzane na podstawie różnych materiałów źródłowych, które nie zawsze były równocześnie podstawą do wprowadzania danych do pola z nazwami obiektów podrzędnych.

Istnieje również duża część wpisów, dla których trudno jest jednoznacznie zamienić nazwę w identyfikator konkretnego istniejącego obiektu. Na drodze automatycznego algorytmu, w sytuacjach niejednoznacznych, nie jest możliwe ustalenie, którą z możliwych sytuacji operator wprowadzający dane miał na myśli. Przykładem takiej sytuacji jest występowanie w bliskiej odległości od siebie kilku jednostek o takiej samej nazwie, ale różnych typach. GUGIK nie udostępnia szczegółów informacji na temat zasad tworzenia relacji pomiędzy poszczególnymi typami jednostek. Dodatkowo zmiany w rejestrze PRNG dokonywane są bez zachowywania wcześniejszych wersji rejestru. W danym momencie dostępna jest jedynie wersja aktualna rejestru. Występujące gwałtowne zmniejszanie się zawartości rejestru z roku na rok, przy jednoczesnym braku informacji o przyczynach zmian, powoduje, że do kolejnych publikowanych przez GUGIK wersji rejestru należy podchodzić bardzo ostrożnie. Być może rejestr jest, dzięki postępującym w jego ramach zmianom, bardziej zgodny z aktualnym stanem rzeczywistym, ale niestety bezpowrotnie tracona jest przy tym informacja o procesie dochodzenia do takiego stanu czyli historii.

Pomimo (lub w wyniku) minimalizacji możliwości występowania anomalii w relacjach pomiędzy różnymi typami jednostek poprzez wprowadzenie ograniczenia na typ jednostki nadrzędnej (tylko wieś albo miasto), oraz warunku zależności typów, występuje wiele relacji dla których prawdopodobnie skrócona została hierarchia wielostopniowej relacji podrzędny-nadrzędny:

osada leśna -> miasto
kolonia kolonii -> wieś
część kolonii -> wieś
przysiółek osady -> wieś
część osady -> wieś
osada osady -> wieś
część miasta -> wieś
kolonia osady -> wieś
część wsi -> miasto

Z kolei najczęściej odrzucanym przypadkiem, który jednak występuje w rejestrze PRNG jest relacja:

wieś -> wieś

podrzędne	nadrzędne	liczba
część wsi	wieś	45945
przysiółek wsi	wieś	12444
część miasta	miasto	11350
kolonia wsi	wieś	4417
osada	wieś	3867

podrzędne	nadrzędne	liczba
osada leśna	wieś	2205
kolonia	wieś	1774
leśniczówka	wieś	1220
osada wsi	wieś	809
osada leśna wsi	wieś	234
przysiółek	wieś	104
schronisko turystyczne	wieś	27
leśniczówka	miasto	16
schronisko turystyczne	miasto	14
przysiółek kolonii	wieś	10
osiedle wsi	wieś	7
osada leśna	miasto	6
osiedle	wieś	6
część przysiółka	wieś	5
kolonia kolonii	wieś	4
część kolonii	wieś	3
przysiółek osady	wieś	2
część osady	wieś	2
osada osady	wieś	2
część miasta	wieś	2
dzielnica	miasto	2
dwór	miasto	1
osada kolejowa	wieś	1
zamek	miasto	1
ruiny zamku	dzielnica	1
osada	miasto	1
przysiółek wsi	miasto	1
ruiny zamku	wieś	1
osada młyńska	wieś	1
kolonia osady	wieś	1
część wsi	miasto	1

Tab. 1. Podział na grupy relacji pomiędzy jednostkami które zostały zakwalifikowane jako nadrzędne.

Typ relacji	Liczba
wieś (rodz nadrz)-wieś (rodz podrz)	1625
osada (rodz nadrz)-osada (rodz podrz)	411
kolonia (rodz nadrz)-część kolonii (rodz podrz)	290
kolonia (rodz nadrz)-kolonia (rodz podrz)	236
leśniczówka (rodz nadrz)-część wsi (rodz podrz)	152
osada leśna (rodz nadrz)-część wsi (rodz podrz)	148
część wsi (rodz nadrz)-wieś (rodz podrz)	117
osada leśna (rodz nadrz)-przysiółek wsi (rodz podrz)	89
osada (rodz nadrz)-osada leśna (rodz podrz)	84
osada (rodz nadrz)-przysiółek osady (rodz podrz)	81
osada (rodz nadrz)-część osady (rodz podrz)	73
kolonia (rodz nadrz)-przysiółek kolonii (rodz podrz)	72
osada leśna (rodz nadrz)-osada leśna (rodz podrz)	71
leśniczówka (rodz nadrz)-osada (rodz podrz)	65
leśniczówka (rodz nadrz)-przysiółek wsi (rodz podrz)	60
część miasta (rodz nadrz)-część wsi (rodz podrz)	59
wieś (rodz nadrz)-część wsi (rodz podrz)	57
część wsi (rodz nadrz)-część wsi (rodz podrz)	51
osada leśna (rodz nadrz)-osada (rodz podrz)	47
osada (rodz nadrz)-kolonia (rodz podrz)	46
część miasta (rodz nadrz)-wieś (rodz podrz)	41
leśniczówka (rodz nadrz)-osada leśna (rodz podrz)	41
osada (rodz nadrz)-część wsi (rodz podrz)	37
osada (rodz nadrz)-wieś (rodz podrz)	37
osada (rodz nadrz)-leśniczówka (rodz podrz)	33
kolonia (rodz nadrz)-osada leśna (rodz podrz)	32
kolonia (rodz nadrz)-część wsi (rodz podrz)	30
wieś (rodz nadrz)-przysiółek wsi (rodz podrz)	29
osada (rodz nadrz)-przysiółek wsi (rodz podrz)	28
kolonia (rodz nadrz)-kolonia kolonii (rodz podrz)	27
leśniczówka (rodz nadrz)-kolonia wsi (rodz podrz)	26
kolonia (rodz nadrz)-wieś (rodz podrz)	25
osada leśna (rodz nadrz)-kolonia wsi (rodz podrz)	25
osada leśna (rodz nadrz)-wieś (rodz podrz)	24
wieś (rodz nadrz)-część kolonii (rodz podrz)	23
przysiółek wsi (rodz nadrz)-część wsi (rodz podrz)	22
leśniczówka (rodz nadrz)-część miasta (rodz podrz)	21

Typ relacji	Liczba
leśniczówka (rodz nadrz)-kolonia (rodz podrz)	20
leśniczówka (rodz nadrz)-wieś (rodz podrz)	20
przysiółek wsi (rodz nadrz)-wieś (rodz podrz)	20
część wsi (rodz nadrz)-przysiółek wsi (rodz podrz)	18
kolonia wsi (rodz nadrz)-część wsi (rodz podrz)	17
część miasta (rodz nadrz)-część miasta (rodz podrz)	16
osada leśna (rodz nadrz)-kolonia (rodz podrz)	16
część wsi (rodz nadrz)-część kolonii (rodz podrz)	15
kolonia wsi (rodz nadrz)-część kolonii (rodz podrz)	15
kolonia wsi (rodz nadrz)-kolonia wsi (rodz podrz)	15
osada (rodz nadrz)-kolonia osady (rodz podrz)	15
osada (rodz nadrz)-osada osady (rodz podrz)	15
osada leśna (rodz nadrz)-leśniczówka (rodz podrz)	15
kolonia wsi (rodz nadrz)-przysiółek wsi (rodz podrz)	14
kolonia wsi (rodz nadrz)-wieś (rodz podrz)	14
leśniczówka (rodz nadrz)-leśniczówka (rodz podrz)	14
osada wsi (rodz nadrz)-część wsi (rodz podrz)	14
wieś (rodz nadrz)-kolonia (rodz podrz)	14
osada (rodz nadrz)-kolonia wsi (rodz podrz)	12
wieś (rodz nadrz)-osada (rodz podrz)	12
część miasta (rodz nadrz)-część kolonii (rodz podrz)	11
część miasta (rodz nadrz)-osada (rodz podrz)	10
kolonia (rodz nadrz)-leśniczówka (rodz podrz)	10
osada leśna wsi (rodz nadrz)-część wsi (rodz podrz)	10
osada wsi (rodz nadrz)-kolonia wsi (rodz podrz)	10
część miasta (rodz nadrz)-przysiółek wsi (rodz podrz)	9
kolonia (rodz nadrz)-przysiółek wsi (rodz podrz)	9
przysiółek wsi (rodz nadrz)-część kolonii (rodz podrz)	9
wieś (rodz nadrz)-osada leśna (rodz podrz)	9
część wsi (rodz nadrz)-osada osady (rodz podrz)	8
przysiółek (rodz nadrz)-przysiółek (rodz podrz)	8
część wsi (rodz nadrz)-osada (rodz podrz)	7
część wsi (rodz nadrz)-osada leśna (rodz podrz)	7
osada (rodz nadrz)-część miasta (rodz podrz)	7
osada wsi (rodz nadrz)-przysiółek wsi (rodz podrz)	7
przysiółek wsi (rodz nadrz)-kolonia (rodz podrz)	7
kolonia (rodz nadrz)-osada (rodz podrz)	6

Typ relacji	Liczba
osada wsi (rodz nadrz)-osada (rodz podrz)	6
osada wsi (rodz nadrz)-wieś (rodz podrz)	6
miasto (rodz nadrz)-część wsi (rodz podrz)	5
osada (rodz nadrz)-część kolonii (rodz podrz)	5
przysiółek wsi (rodz nadrz)-przysiółek wsi (rodz podrz)	5
część miasta (rodz nadrz)-osada leśna (rodz podrz)	4
część wsi (rodz nadrz)-osada wsi (rodz podrz)	4
kolonia (rodz nadrz)-osada kolonii (rodz podrz)	4
miasto (rodz nadrz)-przysiółek wsi (rodz podrz)	4
osada leśna (rodz nadrz)-osada wsi (rodz podrz)	4
osada leśna wsi (rodz nadrz)-osada (rodz podrz)	4
osada leśna wsi (rodz nadrz)-osada leśna wsi (rodz podrz)	4
wieś (rodz nadrz)-kolonia wsi (rodz podrz)	4
wieś (rodz nadrz)-leśniczówka (rodz podrz)	4
część wsi (rodz nadrz)-kolonia (rodz podrz)	3
część wsi (rodz nadrz)-kolonia wsi (rodz podrz)	3
część wsi (rodz nadrz)-leśniczówka (rodz podrz)	3
kolonia kolonii (rodz nadrz)-część kolonii (rodz podrz)	3
kolonia wsi (rodz nadrz)-osada wsi (rodz podrz)	3
leśniczówka (rodz nadrz)-część kolonii (rodz podrz)	3
leśniczówka (rodz nadrz)-część osady (rodz podrz)	3
osada (rodz nadrz)-przysiółek (rodz podrz)	3
osada leśna (rodz nadrz)-część miasta (rodz podrz)	3
osada leśna (rodz nadrz)-przysiółek kolonii (rodz podrz)	3
osada leśna wsi (rodz nadrz)-kolonia wsi (rodz podrz)	3
osada leśna wsi (rodz nadrz)-osada wsi (rodz podrz)	3
osada leśna wsi (rodz nadrz)-przysiółek wsi (rodz podrz)	3
osada leśna wsi (rodz nadrz)-wieś (rodz podrz)	3
osada wsi (rodz nadrz)-osada leśna (rodz podrz)	3
osada wsi (rodz nadrz)-osada leśna wsi (rodz podrz)	3
przysiółek wsi (rodz nadrz)-osada (rodz podrz)	3
wieś (rodz nadrz)-kolonia kolonii (rodz podrz)	3
wieś (rodz nadrz)-osada osady (rodz podrz)	3
wieś (rodz nadrz)-osada wsi (rodz podrz)	3
wieś (rodz nadrz)-przysiółek osady (rodz podrz)	3
część kolonii (rodz nadrz)-wieś (rodz podrz)	2
część miasta (rodz nadrz)-część osady (rodz podrz)	2

Typ relacji	Liczba
część miasta (rodz nadrz)-kolonia wsi (rodz podrz)	2
część miasta (rodz nadrz)-schronisko turystyczne (rodz podrz)	2
część wsi (rodz nadrz)-część osady (rodz podrz)	2
część wsi (rodz nadrz)-osada kolonii (rodz podrz)	2
część wsi (rodz nadrz)-przysiółek (rodz podrz)	2
kolonia (rodz nadrz)-część osady (rodz podrz)	2
kolonia (rodz nadrz)-osada wsi (rodz podrz)	2
kolonia wsi (rodz nadrz)-kolonia (rodz podrz)	2
kolonia wsi (rodz nadrz)-kolonia kolonii (rodz podrz)	2
kolonia wsi (rodz nadrz)-osada (rodz podrz)	2
leśniczówka (rodz nadrz)-osada wsi (rodz podrz)	2
osada (rodz nadrz)-osada wsi (rodz podrz)	2
osada leśna (rodz nadrz)-część kolonii (rodz podrz)	2
osada leśna (rodz nadrz)-część osady (rodz podrz)	2
osada leśna (rodz nadrz)-osada leśna wsi (rodz podrz)	2
osada leśna (rodz nadrz)-przysiółek osady (rodz podrz)	2
osada leśna wsi (rodz nadrz)-osada leśna (rodz podrz)	2
przysiółek (rodz nadrz)-kolonia (rodz podrz)	2
przysiółek wsi (rodz nadrz)-przysiółek (rodz podrz)	2
wieś (rodz nadrz)-osada kolonii (rodz podrz)	2
wieś (rodz nadrz)-przysiółek kolonii (rodz podrz)	2
część kolonii (rodz nadrz)-część kolonii (rodz podrz)	1
część miasta (rodz nadrz)-kolonia (rodz podrz)	1
część miasta (rodz nadrz)-leśniczówka (rodz podrz)	1
część miasta (rodz nadrz)-osada osady (rodz podrz)	1
część wsi (rodz nadrz)-część miasta (rodz podrz)	1
część wsi (rodz nadrz)-przysiółek kolonii (rodz podrz)	1
kolonia (rodz nadrz)-przysiółek osady (rodz podrz)	1
kolonia wsi (rodz nadrz)-leśniczówka (rodz podrz)	1
kolonia wsi (rodz nadrz)-osada leśna (rodz podrz)	1
kolonia wsi (rodz nadrz)-osada osady (rodz podrz)	1
kolonia wsi (rodz nadrz)-przysiółek kolonii (rodz podrz)	1
leśniczówka (rodz nadrz)-osada leśna wsi (rodz podrz)	1
leśniczówka (rodz nadrz)-przysiółek osady (rodz podrz)	1
miasto (rodz nadrz)-osada (rodz podrz)	1
miasto (rodz nadrz)-osada leśna (rodz podrz)	1
osada (rodz nadrz)-schronisko turystyczne (rodz podrz)	1

Typ relacji	Liczba
osada leśna (rodz nadrz)-przysiółek (rodz podrz)	1
osada osady (rodz nadrz)-część osady (rodz podrz)	1
osada osady (rodz nadrz)-osada leśna (rodz podrz)	1
osada wsi (rodz nadrz)-część kolonii (rodz podrz)	1
osada wsi (rodz nadrz)-osada wsi (rodz podrz)	1
osada wsi (rodz nadrz)-przysiółek osady (rodz podrz)	1
przysiółek (rodz nadrz)-część przysiółka (rodz podrz)	1
przysiółek (rodz nadrz)-kolonia wsi (rodz podrz)	1
przysiółek (rodz nadrz)-osada (rodz podrz)	1
przysiółek (rodz nadrz)-osada leśna (rodz podrz)	1
przysiółek wsi (rodz nadrz)-część osady (rodz podrz)	1
przysiółek wsi (rodz nadrz)-kolonia wsi (rodz podrz)	1
wieś (rodz nadrz)-część miasta (rodz podrz)	1
wieś (rodz nadrz)-osada leśna wsi (rodz podrz)	1
wieś (rodz nadrz)-przysiółek (rodz podrz)	1

Tab. 2. Podział na grupy relacji pomiędzy jednostkami, które nie zostały zakwalifikowane jako nadrzędne.

Opisaną wyżej procedurę stworzenia unikatowych identyfikatorów dla miejscowości nadrzędnych w rejestrze PRNG należy zweryfikować w przyszłości w oparciu o inne rejestry państwowe zawierające zestawienia miejscowości oraz określające relacje mereologiczne między nimi (np. BDOO, BDOT czy też GUS-SIMC). W sumie, w wyniku opisanego wyżej przekształcenia zasobu źródłowego PRNG, w wyniku importu do bazy danych powstało 124729 miejscowości (metaobiektów) i ich 155635 manifestacji (różnego typu). Każda manifestacja jest datowana na 18 lipca 2016 (data publikacji tych danych przez GUGIK).

1.1.2. Harmonizacja i aktualizacja

Dodatkowo, w bardzo zbliżonej procedurze, na podstawie danych Atlasu Historycznego Polski (atlasfontium.pl), do bazy danych ‘ontohgis’ załadowane zostały nazwy, typy oraz położenie miejscowości datowane na rok 1600. Zostały przy tym uwzględnione tylko miejscowości posiadające numer PRNG, a więc tożsame z miejscowościami współczesnymi. Powstało w ten sposób 17 857 manifestacji dla nazwy oraz tyle samo dla położenia oraz typu miejscowości. Biorąc pod uwagę, że zbiory danych zewnętrznych mogą zawierać miejscowości nie istniejące w zasobie, został przygotowany skrypt aktualizujący bazę danych ‘ontohgis’ w ten sposób, że dla miejscowości już istniejących w tabeli *VariableSettlements* tworzone są tylko nowe manifestacje położenia, nazwy, typu (wszystkie lub wybrane), natomiast w przypadku miejscowości nie istniejącej w tabeli *VariableSettlements* tworzony jest rekord w tej tabeli i nowe manifestacje położenia, nazwy, typu (raport techniczny oraz skrypt znajdują się w załączniku).

1.2. Jednostki administracyjne

1.2.1. Pierwotne zasilenie bazy danych

Jak zostało powiedziane wyżej jako podstawowa warstwa geometryczna (Least Common Geometry, LCG) przyjęte zostały obręby ewidencyjne z 2017 r. System dopuszcza podział w przyszłości takich jednostek (obrębów) na mniejsze, np. w celu wydzielenia historycznej odrębnej jednostki administracyjnej mniejszej niż dzisiejszy obręb ewidencyjny lub go przecinającej. Uniemożliwia jednak zmianę przebiegu istniejących kształtów. Taka reguła ułatwia diachroniczną harmonizację danych oraz porównywanie przebiegu granic, w tym efektywne stosowanie metody retrogresywnej przy wytyczaniu granic historycznych.

Pierwotne zasilenie bazy danych współczesnymi obrębami ewidencyjnymi oraz ich atrybutami (nazwą gminy, powiatu, województwa) wymagało wprowadzenia do danych źródłowych pewnych modyfikacji. Chodziło głównie o wprowadzenie numerów typów jednostek administracyjnych, korespondujących z opracowaną ontologią, analogicznie jak w przypadku miejscowości. Zadanie to jest o tyle proste, że współczesny podział administracyjny wymaga uwzględnienia tylko 3 poziomów jednostek administracyjnych w ich zgeneralizowanej hierarchii: gmina, powiat, województwo. Następnie każdy wiersz z tabeli wejściowej został przetworzony na zestaw rekordów we wszystkich tabelach manifestacji jednostek podziału terytorialnego (JPT), z uwzględnieniem tego, że tworzone JPT są unikalne, czyli jeśli kolejny rekord (obręb), zawiera te same JPT to nie są one duplikowane. Na podstawie wykonanego mapowania typów, w czasie przetwarzania następowało uwzględnienie łączenia nazw z odpowiednimi identyfikatorami typów.

Na etapie przygotowania danych, oprócz podania właściwych numerów typów jednostek administracyjnych (45: gmina, 46: powiat, 47: województwo, 48: państwo), dokonano pobrania danych o nazwach gmin, powiatów, województw z bazy TERYT (GUS), która zawiera informacje na ten temat. Warto dodać, że dane o powierzchni jednostek podziału terytorialnego w corocznym raporcie GUS „Powierzchnia i ludność w przekroju terytorialnym" podawana jest wg wykazów państwowego rejestru granic i powierzchni jednostek podziałów terytorialnych kraju (PRG). Zbiory te powinny być więc spójne. Informacje o nazwach pobrane z TERYT posłużyły do przygotowania manifestacji nazw JPT, zaś obręby ewidencyjne posłużyły do zasilenia manifestacji geometrycznej (obszaru)[2]. Typ jednostki administracyjnej nie wchodzi w skład jej manifestacji, gdyż zgodnie z przyjętym założeniem – zmiana typu jednostki administracyjnej powoduje jej zanik.

---

2 Więcej na temat architektury powstawania manifestacji geometrycznych dla JPT, por. rozdział poświęcony strukturze bazy danych 'ontohgis'.

W ten sposób w tabeli *VariableAdministrativeUnits* powstało 2906 jednostek administracyjnych (państwo, 16 województw, 380 powiatów oraz 2509 gmin, a także 2906 manifestacji nazw oraz 2906 manifestacji obszarów dla 2017 roku. Liczba gmin podawana wyżej różni się nieznacznie od oficjalnie podawanej (2478). W procedurze uzgadniania jednostek administracyjnych między rejestrem PRG oraz TERYT (GUS) uwidoczniło się szereg niespójności między tymi dwoma zbiorami danych. Okazało się m.in., że jest problem z 3 obszarami miejskimi (Warszawa, Kraków, Łódź), dla których identyfikatory gmin w obrębach ewidencyjnych PRG odnoszą się do nieistniejących identyfikatorów gmin w tabeli gmin TERYT (GUS). Prawdopodobnie GUGIK „zreformował" system identyfikacji gmin w tych 3 obszarach. Dla przykładu: istnieje obręb o kodzie identyfikacyjnym: 106103_9.0035, gdzie kolejne pary cyfr to: 10 to województwo, 61 to powiat, 03 to gmina, _9 to typ gminy, .0035 to nr obrębu, czyli 106103 to identyfikator gminy (bez określania jej typu). Tymczasem w zbiorze gmin nie istnieje gmina 106103. Gmina na terenie której leży ten obręb ma w tabeli gmin kod identyfikacyjny 106101. W przeciwieństwie do całej reszty kraju dla powyższych 3 miast nie istnieją identyfikatory gmin znajdujące się w zbiorze obrębów (około 2 tys. obrębów). Dotyczy to zarówno wersji PRG sprzed kilku lat (v.15) jak i bieżącej (v.22). Niespójności tego typu, o których więcej pisze J. Zieliński w odrębnym rozdziale tego opracowania, nie wpływają w zasadniczy sposób na architekturę i strukturę systemu, jednak dla utrzymania spójności danych historycznych i współczesnych powinny być w przyszłości rozwiązane.

1.2.2. Harmonizacja i aktualizacja

Bardziej złożona procedura, głównie ze względu na charakter geometrii, dotyczy harmonizacji i aktualizacji jednostek administracyjnych. Zbiory zewnętrzne dotyczące jednostek administracyjnych mogą być przygotowane w dwojaki sposób:

1. w oparciu o istniejące obręby ewidencyjne (warstwę z bazy danych 'ontohgis');
2. w oparciu o dowolną, inna geometrię niezgodną, z przebiegiem obrębów ewidencyjnych.

Harmonizacja i przygotowanie zbiorów danych w każdym z opisanych scenariuszy wygląda inaczej, przy czym jest bardziej złożona w drugim przypadku.

Ad. 1. Scenariusz, gdy zewnętrzny zbiór danych dla historycznych podziałów JPT zostanie przygotowane w oparciu o topologię warstwy obszarów (pierwotnie obrębów ewidencyjnych) bazy danych 'ontohgis' jest mało prawdopodobna, jednak należy brać ją pod uwagę. Aby opracować procedurę włączenia takiego zbioru do bazy danych, dla potrzeb testowych, współczesne obręby ewidencyjne otrzymały atrybuty historycznych jednostek administracji terytorialnej dla pięciu przekrojów czasowych: 1368, 1580, 1790, 1900 i 1933, których źródłem były mapy zawarte w Atlasie Rzeczypospolitej Polskiej (1993-1997). Dla każdego z tych przekrojów czasowych przygotowano dane dotyczące podziałów świeckich wyższego rzędu – najczęściej były to województwa, gubernie, prowincje, departamenty, rejencje, prowincje. PRG

w wersji pierwotnej zawierał 53969 obrębów ewidencyjnych. 5 obrębów zostało podzielonych, aby dostosować współczesne granice do granic historycznych (dodatkowym obrębom zostały dodane postfixy do identyfikatorów, w postaci: _[cyfra] (dopisywane na końcu kodu w kolumnie 'jpt_kod_je'). Następnie, analogicznie jak w przypadku współczesnych jednostek podziału terytorialnego zostały wprowadzone ontologiczne identyfikatory typów jednostek administracyjnych. Tak przygotowane dane stały się podstawą stworzenia po 49 manifestacji nazw i obszarów JPT dla 1368 r., 48 dla 1580 r., 47 dla 1790 r., 39 dla 1900 r. oraz 33 dla 1933 r.

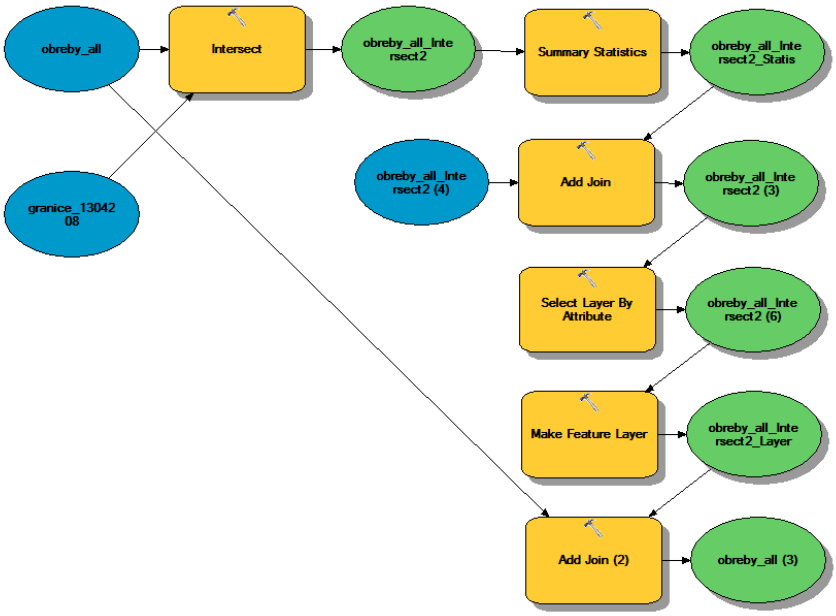
Ad. 2. W opisanej wcześniej procedurze zgodność struktury danych źródłowych (wejściowych) ze strukturą i topologią danych w bazie 'ontohgis' bardzo ułatwiały przeprowadzenie całego procesu. Dodatkowo, wprowadzane do systemu jednostki administracji terytorialnej nie miały swojej reprezentacji, więc nie było konieczności weryfikacji przez skrypt czy dana jednostka administracyjna w bazie już istnieje czy dopiero należy ją stworzyć. Duża część istniejących, czy będących w przygotowaniu zbiorów zewnętrznych, powstaje poza systemem 'ontohgis'. Jako testowy zbiór zewnętrzny przygotowany na własnej, całkowicie odrębnej od obrębów ewidencyjnych osnowie geometrycznej zostały wykorzystane granice Atlasu historycznego Polski dla II połowy XVI wieku (jako datę docelową zasobu przyjmuje się rok 1600). Oprócz odrębności topologicznej zasób ten zawiera jednostki administracyjne, które istnieją już w bazie danych i odpowiadają województwom z 1580 r., które zostały wcześniej wprowadzone do systemu (zob. Ad. 1). Każde z województw Korony w XVI wieku otrzymało już wcześniej swój unikatowy identyfikator w tabeli *VariableAdministrativeUnits*. Całą procedurę należało przeprowadzić w dwóch krokach: 2a) najpierw dokonać harmonizacji i przygotowania danych dostosowując osnowę gemetryczną do warstwy w bazie danych 'ontohgis' (AdministrativeUnitAreas), a następnie 2b) zasilić bazę danych w taki sposób, aby dla JPT już istniejących w tabeli *VariableAdministrativeUnits* skrypt tworzył tylko nowe manifestacje obszaru oraz nazwy, w rozszerzonej wersji także zależności mereologicznych między jednostkami, natomiast w przypadku JPT nie istniejących jeszcze w bazie tworzył nowy rekord w tabeli *VariableAdministrativeUnits* i nowe manifestacje obszaru, nazwy oraz relacji mereologicznych.

Ad. 2a. Harmonizację geometryczną jednostek administracyjnych i ich granic wykonano w trzech etapach. W ich realizacji wykorzystano oprogramowanie geoinformacyjne Esri ArcGIS oraz standardowe funkcje geoprzestrzenne. Opisaną niżej procedurę można wykonać także przy wykorzystaniu innego oprogramowania desktop (np. QGIS) lub przy użyciu zapytań SQL w przestrzennej bazie danych (np. PostgreSQL/Postgis).

Pierwszy etap obejmował przeniesienie atrybutów jednostek administracyjnych pochodzących z danych AHP do warstwy współczesnych obrębów ewidencyjnych znajdujących się w bazie projektu. Etap ten został wykonany w pełni automatycznie. Założono, że atrybuty jednostek administracyjnych z AHP zostaną automatycznie przypisane do obrębów przy założeniu, że w przypadku kiedy obręb ewidencyjny znajduje się w dwóch lub więcej jednostkach AHP

do obrębu zostaną przypisane atrybuty tej jednostki administracyjnej AHP, w granicach której położona jest większa część obrębu. Czynność tę wykonano za pomocą narzędzia *Intersect*, które pozwoliło na przecięcie granic obrębów ewidencyjnych oraz granic jednostek administracyjnych z danych AHP. Następnie obliczono statystyki po intersekcji za pomocą narzędzia *Summary statistics*. W kolejnym kroku za pomocą narzędzia *Join* do obrębów dodano informację o maksymalnej powierzchni - umożliwiło to wybór obrębów o maksymalnych powierzchniach z wykorzystaniem narzędzia *Select By Attruibutes*. Następnie do widoku mapy dodano wynik selekcji jako nową warstwę za pomocą narzędzia *Create layer from selected feature*. W efekcie dane o maksymalnych powierzchniach przypisano do źródłowych obrębów za pomocą narzędzia *Join Data*.

Wymienione czynności zaimplementowano w postaci modelu opracowanego z wykorzystaniem funkcjonalności *Model Builder* w programie ArcGIS 10.5. Model zaprezentowano na rycinie 1.



Ryc. 1. Model opracowany na potrzeby przeniesienia atrybutów z jednostek administracyjnych AHP do warstwy obrębów ewidencyjnych

Etap drugi obejmował opracowanie procedury pozwalającej na automatyczną aktualizację błędów po przeniesieniu atrybutów z warstwy AHP do warstwy obrębów powstałych na skutek przypisania z zastosowaniem „warunku większościowego" w pierwszym etapie. Na wstępie należało wskazać w warstwie obrębów ewidencyjnych takie poligony, które zawierają błędny atrybut przypisany po pierwszym etapie związanym z przeniesieniem atrybutów na podstawie warunku większościowego (warunku maksymalnej powierzchni). W tym celu wykorzystano warstwę obrębów ewidencyjnych, warstwę granic AHP oraz pomocniczo warstwę zawierającą punkty AHP, które reprezentują miejscowości w II połowie XVI wieku. Następnie

za pomocą narzędzia *Spatial Join* połączono warstwę punkty AHP z warstwą granic AHP oraz warstwę punkty AHP z warstwą obrębów ewidencyjnych. Po połączeniu jedna warstwa punktów AHP zawierała atrybuty z warstwy obrębów ewidencyjnych (w tym atrybuty pochodzące z warstwy granic AHP przyłączone w etapie pierwszym), natomiast druga warstwa punktów AHP atrybuty z warstwy granice AHP.

Celem dołączenia do warstwy punktów AHP: (a) atrybutów z warstwy obrębów ewidencyjnych przyłączonych w etapie pierwszym z granic AHP oraz (b) atrybutów z warstwy granic AHP było zidentyfikowanie tych punktów AHP, które zawierają różne wartości dla tych samych jednostek podziału terytorialnego (powiatów, województw, parafii, dekanatów, archidiakonatów i diecezji). W tym celu przygotowano selekcję tych punktów AHP, w których istnieje niezgodność między atrybutami dołączonymi z warstwy granic AHP i warstwy obrębów ewidencyjnych. Weryfikowano następujące wartości atrybutów: 'g_parafia', 'g_dekanat', 'g_archidiakonat', 'g_diecezja', 'g_powiat' oraz 'g_woj'. Weryfikację zgodności atrybutów przeprowadzono analogicznie dla wszystkich rozważanych atrybutów. W wyniku zastosowania tej procedury otrzymano 860 punktów AHP z przypisanymi błędnie wartościami atrybutu 'g_parafia' – największa liczba błędów dotyczyła najmniejszych terytorialnie jednostek administracyjnych. Punkty te były położone w 801 obrębach ewidencyjnych.

W kolejnym kroku ze źródłowej warstwy zawierającej obręby ewidencyjne wybrano wyłącznie obręby zawierające tylko jeden punkt AHP. W tym celu do warstwy obrębów ewidencyjnych za pomocą narzędzia *Spatial Join* dołączono warstwę punktów AHP. W warstwie obrębów po połączeniu, w atrybucie 'join count' program przypisał liczbę punktów znajdujących się w danym obrębie. Wyniku przeprowadzonej analizy zlokalizowano 17 331 obrębów zawierających po jednym punkcie AHP. Wybrane w wyniku analizy obręby ewidencyjne wyeksportowano do odrębnej warstwy.

W ostatnim kroku do warstwy obrębów zawierających tylko jeden punkt AHP za pomocą narzędzia *Join*, wykorzystującego unikatowy numer obrębu ewidencyjnego, dołączono uprzednio wygenerowaną tabelę zawierającą punkty AHP z przypisanymi błędnie wartościami atrybutu 'g_parafia' (860 punktów). W efekcie otrzymano 386 obrębów ewidencyjnych zawierających błędy w atrybucie 'g_parafia', którym następnie automatycznie zaktualizowano wartości atrybutu 'g_parafia'. Aktualizację wykonano w bazie danych Postgresql/Postgis, ale można ją przeprowadzić w dowolnym programie bazodanowym.

Ostatni, trzeci etap polegał na manualnej edycji warstwy obrębów ewidencyjnych w celu aktualizacji i korekty pozostałych 415 obrębów ewidencyjnych zawierających błędne wartości w kolumnie 'g_parafia' i jednocześnie więcej niż jeden punkt AHP. Sytuacja taka oznaczała, że przez obręb ewidencyjny przebiegała granica dwóch historycznych parafii i należało podzielić obręb tak, aby poszczególne jego części i punkty w nim leżące znalazły się w parafiach zgodnych z granicami AHP. Teoretycznie można wykorzystać w tym celu funkcję automatycznego

podziału obszaru między punkty np. diagramy Woronoja, jednak ze względu na konieczność uwzględnienia przy wytyczaniu granic przeszkód naturalnych, np. rzek, za bardziej właściwą uznano metodę weryfikacji ręcznej na podkładzie map historycznych. Opisana wyżej procedura przeprowadzona względem parafii jako najniższej jednostki administracji kościelnej umożliwiła łatwą aktualizację atrybutów dla jednostek wyższego rzędu (dekanatów, archidiakonatów i diecezji). W podobnym modelu przeprowadzono harmonizację dla powiatów i województw w II połowie XVI wieku.

Ad.2b. W wyniku harmonizacji danych przestrzennych i atrybutowych powstała warstwa danych w formacie .shp, która może być podstawą zasilenia bazy danych 'ontohgis'. Utrzymała ona topologiczną zgodność z istniejącą już w bazie danych warstwą przestrzenną bazującą na obrębach ewidencyjnych. Zawiera także kolumny z atrybutami opisującymi przynależność każdego obszaru do właściwej jednostki administracyjnej w II połowie XVI w. – powiatu, województwa, parafii, dekanatu, archidiakonatu i diecezji. Analiza porównawcza przeprowadzona metodą wizualną wskazała na większe lub mniejsze różnice w przebiegu między źródłowymi granicami AHP a granicami wygenerowanymi na podstawie agregacji obrębów. Całe zagadnienie jest bardziej złożone i dotyczy metodyki wyznaczania granic historycznych oraz metody retrogresywnej w kartografii historycznej. Krytycy prezentowanej w tym opracowaniu metody zwracają uwagę na sztuczność współczesnych granic obrębów ewidencyjnych, zwłaszcza na obszarach, które uległy silnym przekształceniom industrialnym oraz urbanizacji. Twierdzą tym samym, że nie mog*ą one* być podstawą wyznaczania granic historycznych. Zwolennicy z kolei wskazują na wady tradycyjnej metody wyznaczania linearnych granic historycznych, zwłaszcza dla epoki przedprzemysłowej, gdzie prowadzenie linii granicznej w wielu sytuacjach nie opiera się na żadnych przesłankach i jest wynikiem dość arbitralnej decyzji kartografa. Wskazują przy tym na zalety wynikające z harmonizacji oraz zgodności topologicznej, która ułatwia stosowanie badań porównawczych. Dostosowywanie i adaptacja współczesnych obrębów ewidencyjnych dla rekonstrukcji granic historycznych przy wykorzystaniu metody retrogresywnej wydaje się w tym momencie najlepszą drogą do uzyskania spójnego obrazu przekształceń jednostek administracji terytorialnej.

W celu zasilenia bazy danych 'ontohgis' przygotowaną w wyniku harmonizacji warstwą granic zawierającą informacje o podziałach dla II połowy XVI wieku został przygotowany odpowiedni skrypt (raport techniczny oraz skrypt znajdują się w załączniku). Należy przy tym pamiętać o określeniu typu importowanej jednostki administracyjnej wg klucza odwołującego się do ontologii JPT - w tym przypadku, dla 1600 roku oznaczało to zastosowanie dwóch systemów administracyjnych:

- dla jednostek podziału świeckiego: „Rzeczpospolita Obojga Narodów (1569-1795)" (id: 14), w jego ramach typy: „powiat" (id: 80) oraz „województwo" (id:82),

- dla jednostek administracji kościelnej: „Kościół katolicki ob. łacińskiego” (id: 26), w jego ramach typy: „parafia” (id: 144), „dekanat” (id:145), „archidiakonat” (id:146), „diecezja” (id: 147).

W przypadku już wcześniej istniejących jednostek administracyjnych (w tym przypadku województw wprowadzonych dla 1580 r.) identyfikowane są dane odpowiedniego meta-obiektu w tabeli *VariableAdministrativeUnits* i tworzone są powiązane z nim manifestacje cząstkowe reprezentujące odpowiednio nazwę tego obiektu, jego obszar w określonym momencie czasowym oraz relację mereologiczną z jednostką wyższego rzędu jeżeli istnieje. Z kolei w przypadku nowych jednostek proces wprowadzania danych rozpoczyna się od utworzenia nowego meta-obiektu reprezentującego nową jednostkę administracyjną. Po jej utworzeniu przeprowadzane są te same czynności co w przypadku jednostek występujących już wcześniej związane z tworzeniem odpowiednich manifestacji.

2. Dane jednostkowe

2.1. Identyfikacja miejscowości oraz jednostek administracyjnych

Zgodnie z założeniami przedstawionymi we wstępnej części rozdziału, oprócz masowego i „wydarzeniowego” zasilenia bazy danych, należy uwzględnić także wprowadzanie danych jednostkowych ze źródeł historycznych w celu uzupełnienia lub aktualizacji istniejącego zasobu dotyczącego miejscowości oraz podziałów administracyjnych. Stworzone rozwiązania informatyczne umożliwiają taką edycję danych, zarówno w zakresie tworzenia nowych obiektów (miejscowości, JPT) jak też tworzenia oraz edycji manifestacji dotyczących nazwy, położenia geograficznego (zasięgu) oraz typu. Pominięty zostanie przy tym opis bezpośredniej pracy z bazą danych czy to przy pomocy języków baz danych (SQL) czy to za pośrednictwem dedykowanych aplikacji (PgAdmin, EMS SQL Manager, Quantum GIS) obsługujących tabele lub perspektywy (widoki) bazy danych PostgreSQL. Przedstawiona zostanie natomiast architektura testowej aplikacji internetowej do aktualizacji informacji na temat miejscowości w bazie danych ‘ontohgis’.

Niezależnie od trybu pracy, bezpośrednio w bazie danych czy też poprzez dedykowane aplikacje został przygotowany trójstopniowy, hierarchiczny model dostępu i edycji danych (Anonymous User, LoggedIn User, Administrator), przy czym najniższy poziom oznacza dostęp do danych na poziomie odczytu, wyszukiwania i pobierania, zaś najwyższy uprawnia do administracji całego systemu. Kluczowy jest poziom środkowy (LoggedIn User), który obejmuje zarówno edycję danych jak też ich walidację przez ekspertów dziedzinowych. Właśnie na tym poziomie użytkownik systemu napotyka największe trudności związane z identyfikacją oraz utożsamieniem miejscowości oraz jednostek administracyjnych. Obszerne rozważania na ten temat w wymiarze krytyki materiału źródłowego zawiera pierwsza część niniejszego opra-

cowania (case studies), zaś od strony teoretycznej i ontologicznej część druga (Ontologia dziedzinowa – ONTOHGIS). Niżej zostaną podane tylko ogólne uwagi, odwołujące się niekiedy do poprzednich części pracy, które mają związek z prowadzeniem kwerendy i pracą badawczą w oparciu o przygotowane narzędzia informatyczne. Dotyczą one głównie miejscowości, ale mogą być także rozciągnięte na jednostki administracyjne. Powinny być brane pod uwagę zarówno przez użytkownika wprowadzającego dane do systemu (np. testowej aplikacji OntoForm), jak też przez eksperta dziedzinowego dokonującego walidacji danych.

Jedną z najważniejszych decyzji, którą należy podjąć wprowadzając informacje do systemu dotyczy powiązania informacji źródłowej z już istniejącym w bazie danych obiektem (miejscowością lub jednostką administracyjną). Problemy z identyfikacją geograficzną, zarówno miejscowości jak i jednostek administracyjnych w przeszłości, mają dwie płaszczyzny.

1. Pierwsza ma charakter bardziej epistemologiczny (źródłoznawczy) i dotyczy kwestii powiązania informacji ze źródła pisanego z konkretnym obiektem przestrzennym. W niektórych sytuacjach historyk napotyka duże trudności przy utożsamianiu przestrzennym informacji geograficznej ze źródeł pisanych.
2. Druga płaszczyzna, którą można określić jako ontologiczną, dotyczy zmiany nazwy, położenia oraz typu miejscowości a także relacji między poszczególnymi miejscowościami w przestrzeni i w czasie. Należy przyjąć bowiem jednolite kryteria odróżniające zmianę (nazwy, położenia, typu) od zerwania ciągłości ontologicznej miejscowości, która oznaczała zanik jednej miejscowości i powstanie innej.

Identyfikacja i utożsamienie miejscowości ze sobą odbywa się zarówno w oparciu o źródła pisane jak i kartograficzne. Źródła pisane informują przede wszystkim o nazwie miejscowości oraz jej cechach takich jak typ, przynależność administracyjna, liczba ludności, własność. Rzadziej – choć są tutaj oczywiście wyjątki np. *Opisanie parafii litewskich z II połowy XVIII w.* – dostarczają informacji o topografii terenu, cechach fizjograficznych czy przebiegu granic. Elementy te są rekonstruowane najczęściej na podstawie źródeł kartograficznych, które podają także nazwy własne, a przez różne metody reprezentacji kartograficznej informują o typie miejscowości, niekiedy własności czy wielkości osiedli oraz przedstawiają przebieg granic.

Kluczową rolę w identyfikacji miejscowości odgrywa charakter źródła oraz sposób podawania informacji o jednostkach osadniczych. Urzędowe rejestry miejscowości są zjawiskiem późnym na ziemiach polskich i pojawiają się powszechnie od II połowy XVIII w. Wiążą się z modernizacją państwa oraz kształtowaniem nowoczesnej administracji. W okresie wcześniejszym informacje o miejscowościach czerpane są ze źródeł, których zadaniem nie była rejestracja miejscowości, ale inne cele: podatkowe (rejestry podatkowe), gospodarcze (lustracje, spisy beneficjów) czy sądowe (zapisy transakcji). Wpływ charakteru źródła na sposób rejestracji miejscowości widać jeszcze w XIX wieku - za przykład niech posłuży sytuacja dwóch miejscowości **Turowszczyzna** oraz **Zubacze** na terenie obecnego powiatu hajnowskiego. W wy-

kazie własnościowym z 1890 r. (Spis 1890, s. 146-147), oprócz wsi (деревня) Turowszczyzna oraz wsi (село) Zubacze pojawiają się majątki (имение) Zubacze, Turowszczyzna-Zubacze, Zubacze-Turowszczyzna. W świetle dostępnych źródeł kartograficznych majątki te są trudne do wyodrębnienia - istniała z pewnością odrębność własnościowa (Konstantyn Kunachowicz – majątek Turowszczyzna-Zubacze, Józef Kunachowicz – majątek Zubacze-Turowszczyzna, Tytus Pusłowski, Aleksandra Bajkowska – majątek Zubacze). Widać wyraźnie, że spis z 1890 r. ma charakter bardziej własnościowy niż topograficzno-geograficzny. W wykazie z 1905 r. (s. 78, 87) znajdują się dwie odrębne miejscowości: wsi (деревня) Zubacze i Turowszczyzna oraz dwa majątki (имение) Zubacze i Turowszczyzna. W związku z tym zapis o połączonej miejscowości Zubacze-Turowszczyzna w rubryce „nazwa osady" (Названіе поселеній) w spisie z 1890 r. należy traktować pomocniczo w stosunku do kolumny podającej właścicieli ziemskich. Spis z 1890 r. dotyczy – jak sama nazwa wskazuje – właścicieli ziemskich, zaś spis z 1905 r. – dotyczy miejscowości. To samo należy wziąć pod uwagę przy analizie map topograficznych – nie jest ich celem wyszczególnienie miejscowości, lecz oddanie topografii terenu (głównie dla celów wojskowych); zupełnie inny charakter będzie miał Słownik Geograficzny Królestwa Polskiego, gdzie głównym przedmiotem spisu była rejestracja miejscowości oraz obiektów topograficznych.

Z wyżej wymienionych powodów w przygotowaniu map osadnictwa historycznego, tam gdzie jest to możliwe, powinny być w pierwszy mrzędzie wykorzystane spisy miejscowości, a nie inne źródła, gdzie nazwy miejscowości stanowią tylko atrybut opisowy dla rejestracji innych zagadnień. Większość wartość będą miały np. wizytacje kanoniczne czy opisy topograficzne (szkice czy opisy Karola Perthéesa z II połowy XVIII w.) niż rejestry podatkowe. Celem podawania nazw miejscowych w wizytacjach, przy różnych zastrzeżeniach, była rejestracja miejscowości należących do parafii (villae parochiales), a nie spis właścicieli odprowadzających podatek. Dokładniejsza analiza rejestrów poborowych wskazuje, że miejscowości i parafie podawane w nagłówkach pozycji spisowych miały tylko znaczenie pomocnicze i porządkujące. Najważniejszymi informacjami był płatnik podatku oraz podstawa opodatkowania – niekiedy podawana zbiorczo z kilku miejscowości położonych w różnych parafiach, a nawet powiatach, a należących do tej samej osoby[3].

Fundamentalnym elementem procesu analitycznego w badaniach dotyczących dziejów osadnictwa oraz podziałów terytorialnych jest powiązanie informacji źródłowych z konkretnymi, odniesionymi przestrzennie, punktami osadniczymi (np. miasto, wieś, folwark) lub jednostkami administracyjnymi (np. gmina, parafia). Takiej identyfikacji służy głównie nazewnictwo i położenie geograficzne, a w wątpliwych sytuacjach także dodatkowe cechy wspólne analizo-

3 Na temat wartości rejestrów poborowych zob. K. Boroda, *O przydatności rejestrów poborowych*, s. 21–38; M. Butkiewicz, *Wartość rejestrów poborowych*, s. 127–145.

wanych obiektów – typ jednostki, przynależność do jednostki administracyjnej, własność, wielkość (liczba ludności) itd. Nie zmienia to faktu, że w wielu przypadkach jednoznaczna identyfikacja budzi wątpliwości albo jest całkowicie niemożliwa. Nawet dokładne badania prowadzone w mikroskali nie dają pewności rozstrzygnięcia. Ciekawym przykładem, pokazującym pełny wachlarz środków i metod służących identyfikacji miejscowości w czasie i przestrzeni, może być niedawne studium R. Sikorskiego poświęcone wsi Newlende alias Herczigiswalde[4]. Napotkane trudności można usystematyzować wg trzech kryteriów: cecha identyfikująca, typ źródła oraz typ identyfikacji (synchroniczna, diachroniczna).

Podstawowa czynność badawcza w procesie identyfikacji polega na powiązaniu informacji o jednej miejscowości z dwóch źródeł pisanych, z dwóch źródeł kartograficznych lub ze źródła pisanego i ze źródła kartograficznego. Bardzo rzadko, a im dalej w przeszłość wręcz sporadycznie, istnieje możliwość porównywania informacji pochodzących z tego samego okresu, np. z konkretnego roku. Zazwyczaj badania geograficzno-historyczne prowadzone są dla określonego przedziału czasowego, w oparciu o szerszą podstawę źródłową - pisaną i kartograficzną – wytworzoną w różnych latach. Wraz ze wzrostem odstępów czasowych między analizowanymi źródłami stopień prawdopodobieństwa wnioskowania maleje. Na szczęście procesy osadnicze i zmiany nazw, jeżeli nie ma okoliczności nadzwyczajnych (zmiany granic politycznych, zmiana ustroju politycznego) przebiegają ewolucyjnie. Zawsze jednak należy pamiętać o relacji zachodzącej między czasem wykorzystywanych źródeł a okresem historycznym, dla którego wykonuje się rekonstrukcję krajobrazu kulturowego lub przyrodniczego.

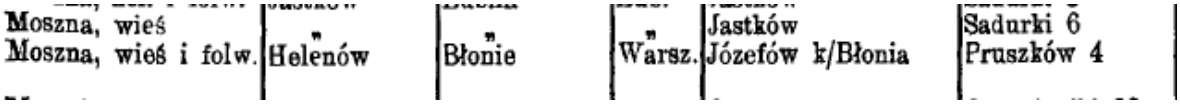
Jednym z najważniejszych problemów jest sposób reprezentacji oraz rejestracji miejscowości, które, przy jednej nazwie własnej, otrzymują w źródłach pisanych kategorie łączone lub złożone, np. wieś i folwark, folwark i leśniczówka. Opisane sytuacje niekiedy komplikuje dodatkowo charakter zabudowy rozproszonej (np. osadnictwo olęderskie) – nawet na dokładnych dawnych mapach topograficznych dość trudno ustalić przynależność poszczególnych domostw do konkretnej miejscowości. Należy przywołać w tym miejscu metodę krytyki wynikającą z doświadczenia *Atlasu historycznego Polski* w odniesieniu do folwarków czy młynów lub innych obiektów gospodarczych. Henryk Rutkowski wskazał trzy elementy determinujące odrębność osad: wyodrębnienie przestrzenne zabudowy, odgraniczenie terenu danej miejscowości od innych miejscowości oraz nazwa własna[5]. Takie podejście koresponduje częściowo z podejściem współczesnym oraz definicją miejscowości przyjętą w projekcie. Elementem decydującym okazuje się w tym przypadku odrębność przestrzenna, najlepiej wyrażona w źródłach kartograficznych przez odrębną etykietę nazwy własnej miejscowości (nie mylić ze skrótem dla typu obiektu topograficznego). Jeżeli folwark ma tę samą nazwę co wieś i leży w jej bezpo-

4 R. Sikorski, *Sprawa wsi Newlende*, s. 207-215.

5 H. Rutkowski, *Lokalizacja miejscowości*, s. 64.

średnim sąsiedztwie, przez co trudno wskazać terytorialną odrębność wsi i folwarku, wówczas mamy do czynienia z jedną miejscowością o typie złożonym (wieś i folwark). Folwark może być oznaczony właściwym skrótem (folw., Vorw.), ale nie posiada nazwy własnej i powinien być traktowany jak obiekt gospodarczy – syt. 1: Moszna (Ryc. 1, 2). Jeżeli w wyniku analizy

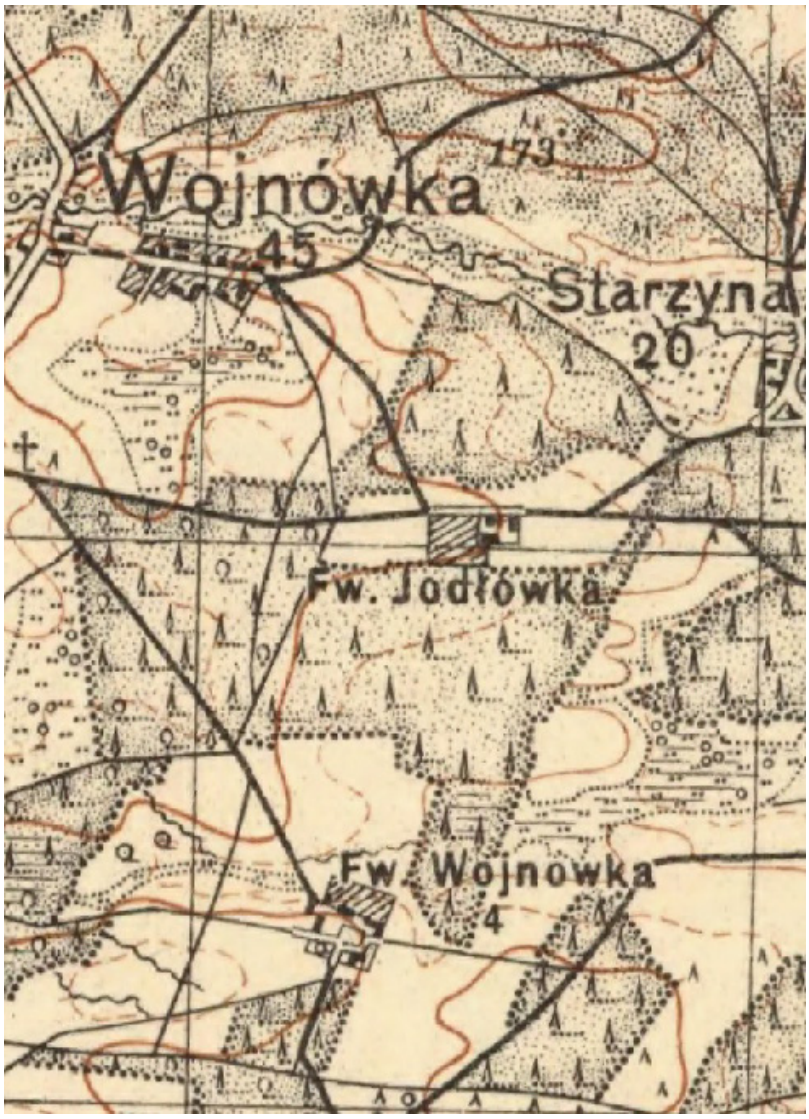
Ryc. 1. Moszna na mapie WIG.



Ryc. 2. Moszna w skorowidzu Bystrzyckiego (s. 1085).

źródeł kartograficznych można wyodrębnić oraz wydzielić przestrzennie folwark od wsi wówczas można mówić o dwóch miejscowościach (wieś, folwark). Folwark wówczas jest traktowany jako odrębna miejscowość i ma swoją nazwę własną, która może być identyczna z nazwą wsi – syt. 2: Wojnówka (Ryc. 3, 4)[6]. W każdej sytuacji taki szczególny przypadek należy poddać dokładniejszej analizie. Trzeba dodać, że pojęcie „granica miejscowości" nie funkcjonuje nawet we współczesnym polskim systemie prawnym, gdzie ciągle istnieją rozbieżności między

6 Wojnówka, tamże dwór ¼ od wsi, K. Perthées, *Geograficzno-statystyczne opisanie parafiów*, t. 12 (par. Kleszczele). Wieś (деревня) i majątek (имение) (dwie miejscowości), П.М. Диков, *Список землевладений*, s. 148; *Указатель населенным местностям*, s. 74. Wieś i dobra (jedna miejscowość), SGKP, t. 13, s. 757. Wieś i folwark (dwie miejscowość), *Skorowidz miejscowości. Województwo poleskie*, s. 11. Wieś i folwark (jedna miejscowość), T. Bystrzycki, *Skorowidz miejscowości*, s. 1871.



Ryc. 3. Wojnówka wieś i Wojnówka folwark na mapie WIG.

Wojnów, wieś i kol.	Stok Ruski	Siedlce	Lub.	Mordy
Wojnówka, wieś i folw.	Wierzchowice	Brześć n/Bugiem	Pol.	Wierzchowice koło Brześcia n/Bug.

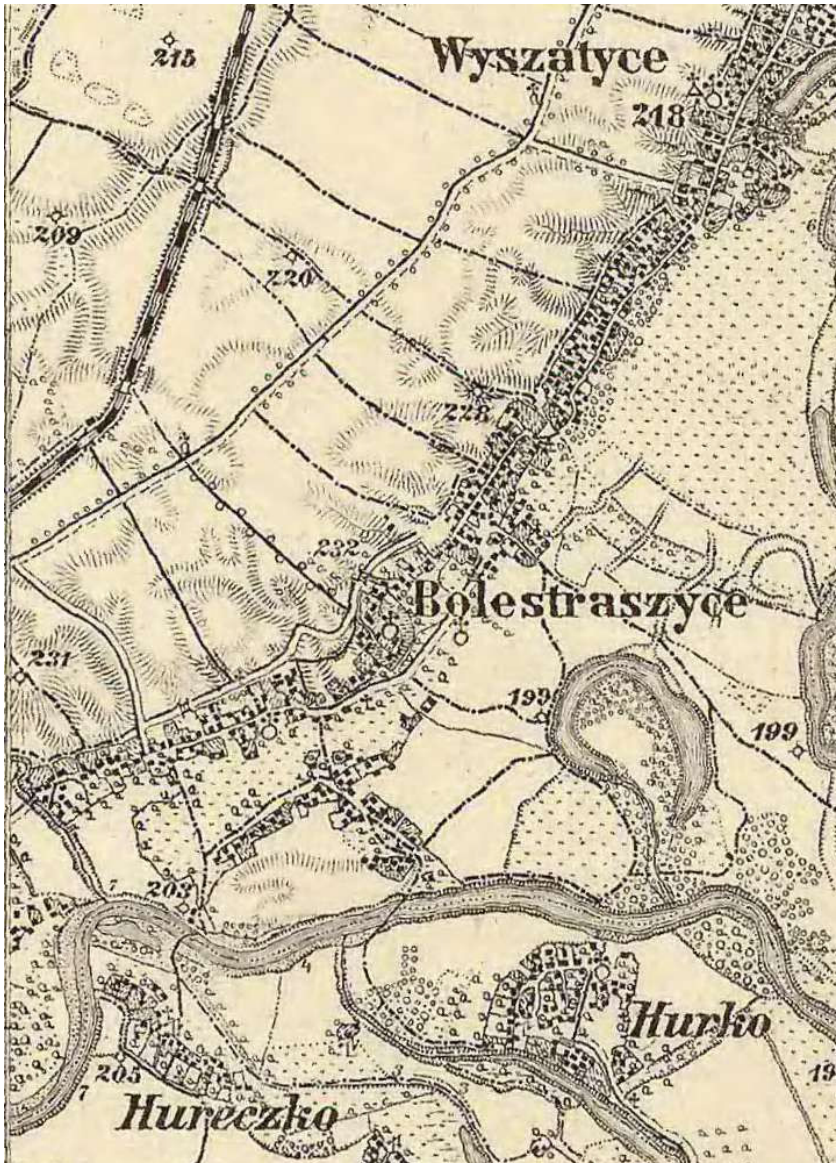
Ryc. 4. Wojnówka w skorowidzu Bystrzyckiego (s. 1871).

porządkiem administracyjnym (podziały administracyjne) oraz geodezyjnym (działki, obręby, jednostki ewidencyjne)[7].

Trudności identyfikacyjne powoduje także podawanie w źródłach pisanych nazw miejscowych, które posiadają wspólny główny człon nazwy i odmienne człony dodatkowe: Stare i Nowe, Górne i Dolne, Wielkie i Małe. Dotyczy to także działów własnościowych w przypadku drobnego osadnictwa mazowieckiego czy podlaskiego. Niekiedy trudno jednoznacznie stwierdzić czy chodzi o jedną miejscowość (nazwa wskazuje tylko na to, że miejscowość składa się z

7 Zob. rozdział autorstwa J. Zielińskiego w niniejszym opracowaniu.

Ryc. 5. Bolestraszyce na III zdjęciu wojskowym Galicji.



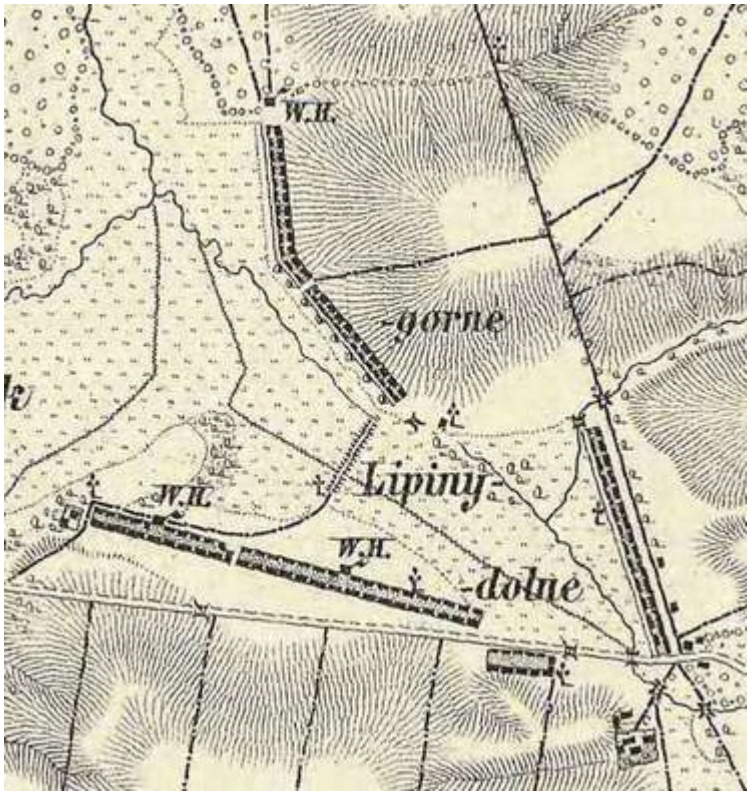
Bolesławska Kępa, wieś	Pawłów	Stopnica	Kiel.	Nowy Korczyn
Bolestraszyce, wieś	Bolestraszyce	Przemyśl	Lwow.	Żurawica
Bolesty, wieś	Łubin	Bielsk	Białst.	Bielsk Podl.

Ryc. 6. Bolestraszyce w skorowidzu Bystrzyckiego (s. 129).

dwóch części) czy też chodzi o dwie odrębne miejscowości. Jako ilustrację można podać dwa przykłady ze SGKP: syt. 3: „Bolestraszyce, dolne i górne, wieś, pow. przemyski" (jedna miejscowość)[8] (Ryc. 5, 6), syt. 4: Lipiny (Górne i Dolne), wieś, pow. biłgorajski (dwie wsi)[9] (Ryc. 7, 8). Podobnie jak w przypadku rejestracji miejscowości o typach złożonych decydująca będzie tutaj reprezentacja kartograficzna oraz zestawienie z dodatkowymi źródłami pisanymi.

8 SGKP, t. 1, s. 300.

9 SGKP, t. 5, s. 265.



Ryc. 7. Lipiny Górne i Dolne na III zdjęciu wojskowym Galicji.

Lipiny Dolne, wieś	Potok	Biłgoraj	Lub.	Potok Górny
Lipiny Główne, folw.	Potok	Biłgoraj	„	Potok Górny
Lipiny Górne, wieś	Potok	Biłgoraj	„	Potok Górny

Ryc. 8. Lipiny Górne i Dolne w skorowidzu Bystrzyckiego (s. 895).

Jako pewnego rodzaju podsumowanie tych krótkich rozważań nad epistemologiczną i ontologiczną płaszczyzną identyfikacji i utożsamienia miejscowości może posłużyć przykład miejscowości **Golakowa Szyja** z obecnego powiatu Hajnówka. W spisie z 1890 r. występują dwie miejscowości o nazwie **Golakowa Szyja** leżące tuż obok siebie, w odrębnych gminach i powiatach. Jedna z nich była osadą (поселок) w gminie Masiewo (Spis 1890, s. 213), zaś druga uroczyskiem (урочище) w gminie Łosinka (Spis 1890, s. 89). Sytuację potwierdza SGKP (t. 15/2, s. 565), gdzie wzmiankowane jest uroczysko w powiecie bielskim i gm. Łosinka oraz osada włościańska w powiecie prużańskim i gm. Masiewo. W wykazie osad z 1905 r. występuje uroczysko w gm. Łosinka - 11 mieszkańców (Spis 1905, s. 57), zaś drugie uroczysko w gm. Masiewo - 34 mieszkańców (Spis 1905, s. 114). Obie osady występują też na trójwiorstówce (Ryc. 9). Na innych mapach występuje tylko jedna miejscowość – na dwuwiorstówce jako kolonia (выселок). Późniejsze mapy rejestrują pojedyncze zabudowanie w lokalizacji dawnego uroczyska w gminie Łosinka, ale bez odrębnej nazwy, zaś wykaz z 1933 r. (Bystrzycki, s. 455) tylko osadę Golikowa Szyja w gm. Masiewo. Obecnie przysiółek Golakowa Szyja [PRNG: 34357] wsi Wasilkowo, jest zlokalizowany w miejscu dawnej osady Golakowa Szyja należącej

Ryc. 9. Golaszkowa Szyja na trójwiorstówce.

do gminy Masiewo. Opisana podstawa źródłowa pozwala na przyjęcie dwóch odmiennych i sprzecznych interpretacji.

1. istniała przez cały okres tylko jedna miejscowość Golakowa Szyja, rejestrowana w różny sposób z powodu przynależności do dwóch jednostek administracyjnych,
2. istniały dwie miejscowości, jedna należąca do gminy Masiewo, zaś druga do gminy Łosinka, przy czym Golakowa Szyja należąca do gminy Łosinka zanikła w I połowie XX wieku.

Pamiętając o cytowanym w innym miejscu monografii przykładzie wsi Gruzka (Hruzka) i możliwości podziału miejscowości między dwie jednostki administracyjne należy raczej odrzucić zmianę o charakterze ontologicznym (zanik miejscowości, absorpcja) i przyjąć jako podstawę rozstrzygnięcia założenie, że podwójna rejestracja miejscowości w źródłach była spowodowana jej przynależnością do dwóch różnych jednostek administracyjnych. Tym samym należy traktować o jednej miejscowości Golakowa Szyja[10]. Warto dodać, że zaszłości historyczne powodują nawet dzisiaj trudności w przypisaniu miejscowości do jednej tylko jednostki podziału terytorialnego. Z ostatnich opisywanych w prasie przypadków można wymienić: Budzyń (gm. Księżpol i Łukowa), Paary (gm. Susiec i Narol), Gulzów (gm. Pilica i Ogrodzieniec) czy Chwałki (gm. Sandomierz i Obrazów).

2.2. Ontoform (prototyp dla gromadzenia danych dotyczących miejscowości i jednostek osadniczych)

Procedura gromadzenia danych jednostkowych zakłada wykorzystanie oddzielenego schematu w bazie danych ‘ontohgis’ o nazwie ‘sources’. Umożliwia on zapis danych pochodzących ze źródeł historycznych a następnie przeprowadzenie weryfikacji oraz walidacji wprowadzonych informacji przez ekspertów dziedzinowych (dalej: opieków danych), aby uniknąć przenosze-

10 Tak też uznał Kondratiuk, M. Kondratiuk, *Nazwy miejscowe południowo-wschodniej Białostoczyzny*, s. 61.

nia do manifestacji infoormacji niespójnych, niepewnych czy sprzecznych. Źródła historyczne, zwłaszcza z epoki średniowiecza i czasów nowożytnych, cechuje słaba precyzja i często niepełność, niepewność oraz niejednoznaczność. W odniesieniu do tego samego obiektu (miejscowość, jednostka) mogą pojawiać się w źródłach historycznych dwie odmienne charakterystyki nazwy, typu lub położenia. Dopiero po walidacji danych następuje utworzenie manifestacji cząstkowych miejscowości i jednostek administracyjnych w schemacie 'ontology'. Ładowanie informacji jednostkowych (na podstawie danych wprowadzanych za pomocą OntoForm), przebiega podobnie do ładowania informacji w trybie masowym z tym że: (1) informacje wejściowe mają inną stałą strukturę, która jest podstawą działania OntoForm, (2) zakres danych do załadowania wybiera się poprzez oznaczenie rekordów (fisz), które mają podlegać załadowaniu.

Opracowana w ramach projektu struktura bazy danych została zaimplementowana z wykorzystaniem serwera bazy danych PostgreSQL wraz z rozszerzeniem przestrzennym PostGIS. Wszelkie operacje na danych możliwe są do wykonania za pomocą języka SQL. Aby jednak ułatwić pracę z gromadzonymi danymi, dodatkowo podjęto prace nad stworzeniem aplikacji obsługującej wprowadzanie i edycję danych. Model aplikacji przedstawiono na rycinie 10. W czasie tworzenia modelu aplikacji przygotowany został jej prototyp zawierający podstawowe funkcjonalności dotyczące edycji informacji dotyczących miejscowości i jednostek osadniczych (topologia punktowa).

Poszczególne przypadki użycia zostały opisane poniżej:

Przypadek użycia: Information Searching

Opis: Wyszukiwanie informacji poprzez podanie słów kluczowych. Wyszukiwanie może przebiegać po jednym lub wielu atrybutach, w tym również po atrybucie czasowym. Wyszukiwanie może również dotyczyć metadanych.

Aktor: użytkownik zewnętrzny

Przypadek użycia: Data Downloading

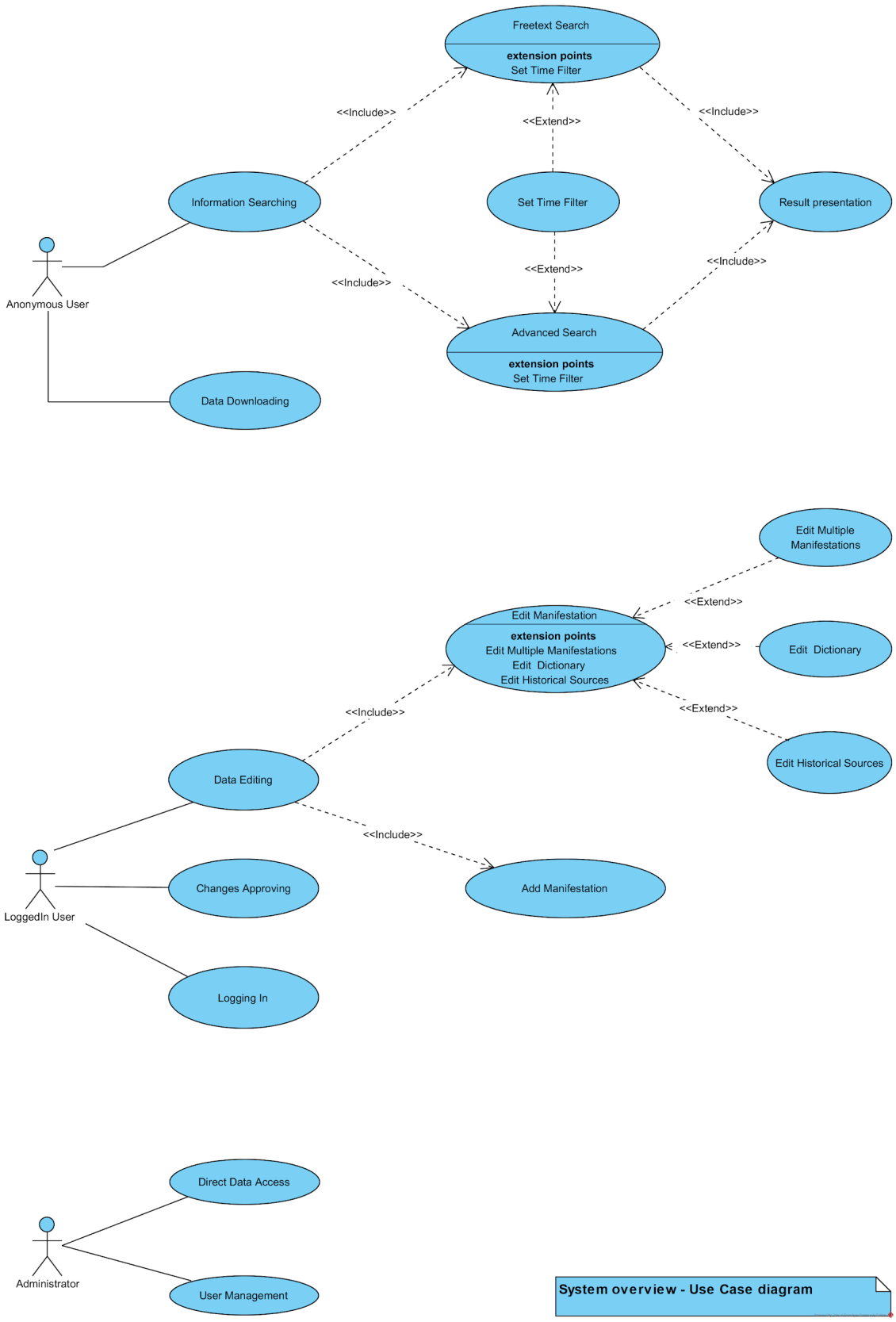
Opis: Pobranie zawartości bazy danych. Wybrane tabele z wybranymi kolumnami mogą być zapisane w postaci plikowej. Plik nie zawiera danych służących do administrowania zawartością bazy danych. Pobierana jest zawsze cała zawartość bazy danych.

Aktor: użytkownik zewnętrzny

Przypadek użycia: Freetext Search

Opis: Uproszczone wyszukiwanie do używania którego nie jest konieczna znajomość struktury bazy danych. Wartości do wyszukania podaje się w jednym polu. Wyszukiwanie przebiega w całej zawartości bazy danych.

Aktor: użytkownik zewnętrzny



Ryc. 10. Diagram przypadków użycia aplikacji służącej do obsługi danych źródłowych i manifestacji

Przypadek użycia: Result presentation

Opis: Prezentacja wyników wyszukiwania. Wyniki prezentowane są w ten sam sposób niezależnie od wybranej wcześniej metody wyszukiwania. Prezentacja wykorzystuje stronicowanie w celu obsługi długiej listy wynikowej. Wyniki prezentowane są jednocześnie w postaci tabelarycznej oraz w postaci mapy.

Aktor: użytkownik zewnętrzny

Przypadek użycia: Set Time Filter

Opis: Ustalanie parametru czasowego w celu wyszukania danych. Czas można określić za pomocą podania daty w postaci numerycznej lub za pomocą suwaka osi czasu. Możliwość określania daty posiada ograniczenia w postaci limitu dolnego daty, limitu górnego daty oraz wybranych wartości, które mogą być ustawiane pomiędzy limitem dolnym a górnym.

Aktor: użytkownik zewnętrzny

Przypadek użycia: Data Editing

Opis: Edycja istniejącej zawartości bazy danych. Edycji mogą podlegać: manifestacje, listy wartości do wyboru (słowniki), źródła historyczne, dotyczące

- miejscowości
- jednostek administracyjnych
- obiektów (jako potencjalne miejscowości)

Dane edytowane za pomocą aplikacji OntoForm (gromadzenie danych źródłowych) są następnie automatycznie przetwarzane do postaci manifestacji. Przetworzenie następuje po analizie materiału źródłowego, przez opiekuna danych. W wyniku powstają dane krytyczne.

Dodawanie oraz edycja danych obejmuje następujące atrybuty:

- Miejscowości
  - ◊ geometria punktowa (w przyszłości także poligonowa)
  - ◊ nazwy
  - ◊ typy
  - ◊ zależność mereologiczna (jest powiązana, jest częścią etc.)
- Jednostki administracyjne:
  - ◊ geometria poligonowa
  - ◊ nazwy
  - ◊ typy
  - ◊ zależność mereologiczna

W przypadku wprowadzania do bazy danych nazw których typ to tablica łańcuchów znakowych, należy stosować następujące reguły. Poszczególne listy nazw ograniczone są nawiasami klamrowymi. Na przykład zapis: {{},{},{}} oznacza 3 listy nieposiadające żadnych

wartości (puste). Nazwy podaje się w cudzysłowach oddzielając je przecinkami. Kolejne listy to kolejne języki. Ustalonych zostało 10 języków (czyli używamy 10 list (wierszy):

1. polski,
2. rosyjski,
3. niemiecki,
4. ukraiński,
5. białoruski,
6. litewski,
7. łacina
8. czeski
9. węgierski
10. angielski

Języki rozróżniane są po kolejności w jakiej nazwa wystąpi na głównej liście.

Przykład nazwy określonej tylko w języku rosyjskim: {{„”},{Постолово}}

Aktor: zalogowany użytkownik posiadający odpowiednie uprawnienia

Przypadek użycia: Changes Approving

Opis: Weryfikacja wprowadzonych nowych danych lub zmian w danych.

Aktor: zalogowany użytkownik posiadający uprawnienie do dokonywania weryfikacji i uzgadniania danych

Przypadek użycia: Logging In

Opis: Logowanie do systemu. W wyniku logowania, dostępne w systemie operacje mogą być dostępne lub nie w zależności od uprawnień związanych z danym użytkownikiem.

Aktor: użytkownik posiadający konto w systemie

Przypadek użycia: Edit Manifestation

Opis: Edycja manifestacji zawartych w bazie danych. Składa się na nią: edycja danych obejmujących wiele manifestacji jednocześnie, edycja słowników, edycja źródeł historycznych. Zmienione informacje oczekują następnie na weryfikację przez opiekuna danych.

Aktor: zalogowany użytkownik posiadający uprawnienie do edycji danych

Przypadek użycia: Add Manifestation

Opis: Utworzenie nowej manifestacji, nieistniejącej wcześniej w bazie danych. Nowa manifestacja dołączana jest do istniejącej jednostki lub tworzona jest wraz z nową jednostką. Wprowadzone za pomocą aplikacji OntoForm informacje oczekują na weryfikację przez opiekuna danych, a następnie są automatycznie przetwarzane do postaci manifestacji.

Dodawanie nowych miejscowości odbywa się za pomocą okna mapy. Dodawana jest nazwa i lokalizacja. Przed dodaniem miejscowości należy się upewnić że nie istnieje już taka miejscowość w bazie danych.

Nazwy miejscowości podawane są w polach nazwa_zrodlo i nazwa_standard.

Oprócz tego można podać je w polu fisza. Wówczas należy zachować brzmienie oryginalne (zapis i alfabet).

W polu fisza można (ale nie jest to konieczne) podać modernizację zapisu nazwy źródłowej. Modernizacja może dotyczyć zarówno modernizacji zapisu jak też zmiany formy gramatycznej (np. z dopełniacza na mianownik). Modernizacja polega też na rozwinięciu skrótów, występujących w źródłach. W przypadku stwierdzenia oczywistych błędów literowych należy podać też formę poprawną czy też regularnie występującą w danym źródle. Ze względu na ograniczenia techniczne (baza danych) określenie kroju pisma (kursywa) należy podawać tylko w wyjątkowych i niezbędnych sytuacjach.

W kolumnie ‘nazwa_zrodlo’ podaje się nazwy w mianowniku oraz rozwiniętej formie [po ewentualnej modernizacji polegającej na sprowadzeniu do mianownika lub rozwinięciu skrótu, czy też korekcie oczywistego błędu literowego czy redakcyjnego].

W kolumnie ‘nazwa_standard’ podaje się nazwy w mianowniku, po transliteracji oraz modernizacji zapisu, jeżeli w zapisie źródłowym zastosowana została dawna gramatyka lub pisownia. Wypełnienie pola nazwa_standard nie jest obowiązkowe.

Nazwy miejscowości w polach nazwa_zrodlo i nazwa_standard powinny być podawane w mianowniku, także jeżeli w źródle podana jest w innej formie np. dopełniacza. Wszelkie wątpliwości odnośnie przekształcenia nazwy do formy mianownikowej lub metod modernizacji lub transliteracji powinny być odnotowywane w polu fisza.

Transliteracja cyrylicy odbywa się wg standardu ISO 9:1995 – oficjalnie przyjętego także przez Polskę, jako PN-ISO 9:2000. Jeżeli chodzi o modernizację zapisu nazw w alfabetach łacińskich to powinna brać ona pod uwagę instrukcje wydawnicze dla źródeł historycznych.

Aktor: zalogowany użytkownik posiadający uprawnienie do edycji danych

Przypadek użycia: Edit Multiple Manifestations

Opis: Edycja wielu manifestacji (związanych zwykle z jednym, tym samym źródłem) jednocześnie. Wszystkie edytowane manifestacje związane są z tą samą jednostką i tym samym źródłem historycznym.

Aktor: zalogowany użytkownik posiadający uprawnienie do edycji danych

Przypadek użycia: Edit Dictionary

Opis: Edycja słownika zawierającego listy dopuszczalnych wartości. Edycja słownika może dodawać nowe pozycje na liście lub modyfikować istniejące. Edycja nie może usuwać

istniejących pozycji. Nie powinno się również zmieniać znaczenia istniejących pozycji o określonych identyfikatorach.

Aktor: zalogowany użytkownik posiadający uprawnienie do edycji danych

Przypadek użycia: Edit Historical Sources

Opis: Edycja informacji o wykorzystywanych źródłach historycznych

Aktor: użytkownik wewnętrzny (administrator danych)

Przypadek użycia: Direct Data Access

Opis: Bezpośredni dostęp do danych, bez użycia aplikacji użytkowej. Przeglądanie i modyfikacja danych za pomocą języka SQL.

Aktor: użytkownik wewnętrzny (administrator danych)

Przypadek użycia: User Management

Opis: Zarządzanie kontami użytkowników z wykorzystaniem mechanizmu SSO (LDAP). Konta zakładane ręcznie. Zmiana hasła samodzielnie przez użytkowników.

Aktor: użytkownik wewnętrzny (administrator danych)

ZAŁĄCZNIK

Ładowanie danych zawierających jednostki podziału terytorialnego

- Obejmuje wstawianie nowych VAU i aktualizację istniejących VAU.
- Dotyczy nazwy, typu, geometrii, mereologii.

Dane wejściowe mogą zawierać wiele poziomów podziału administracyjnego, ale jednorazowo ładowany jest jeden poziom.

Dane pobierane są poprzez widok **vau_import_view** do odpowiednich tabel z manifestacjami w schemacie „ontology". Przed uruchomieniem skryptu należy dostosować zawartość widoku **vau_import_view** do odpowiedniego poziomu który ma zostać załadowany. Skrypt załaduje dane tak jak je widać w widoku bazodanowym **vau_import_view**. W razie braku pewnych danych które są niezbędne do uruchomienia procesu ładowania, można je uzupełnić w tabeli źródłowej (dodając wcześniej odpowiednią kolumnę), bądź „wirtualnie" dodając je w widoku.

Parametry wejściowe:

- src_table: tabela z danymi wejściowymi. Przykładowa wartość: public.jpt_kosciel-ne_1600
- jpt_name_field pole z nazwą jednostki. Przykładowa wartość: g_diecezja
- jpt_vau_field = pole z identyfikatorem VAU. Przykładowa wartość: g_diecezja_vau. Służy do zapisu informacji o nowo utworzonej jednostce, jeśli takie są tworzone.
- clear_tables: przełącznik usuwania danych z poprzedniego uruchomienia (opcjonalny). W przypadku ustawienia na 1, wszystkie wcześniej załadowane dane z tego źródła zostaną usunięte z tabel ontologicznych.

Tabela z danymi wejściowymi powinna zawierać następujące dane, dla każdego poziomu podziału terytorialnego lub umożliwiać automatyczne przetworzenie do struktury zawierającej dane zorganizowane w postaci kolumn:

nazwa: nazwa jednostki podziału terytorialnego, wyrażona w języku polskim.

data_p: data początkowa obowiązywania danych.

data_k: data końcowa obowiązywania danych.

typ_id: typ jednostki podziału terytorialnego zgodny z podziałem na typy jednostek stosowanym w systemie ontologicznym IHPAN.

vau_id: identyfikator jednostki podziału terytorialnego (VAU) jednoznacznie identyfikujący konkretną jednostkę w systemie ontologicznym IHPAN. W przypadku podania tej wartości dane będą przyłączane do istniejącej w bazie danych jednostki. Jeśli zawiera wartość 0 to tworzony jest nowy VAU. Jeśli wartość jest inna niż >=0, rekord jest pomijany.

gid: identyfikator AUA zgodny z identyfikatorami stosowanymi w systemie ontologicznym IHPAN, dla najmniejszych jednostek obszarowych z których składają się poszczególne jednostki podziału terytorialnego.

UWAGA: W sytuacji powtarzających się nazw jednostek w ramach tej samej jednostki nadrzędnej, albo po prostu istnienia różnych jednostek o takiej samej nazwie, bez podawania jednostek nadrzędnych, należy do widoku bazodanowego **vau_import_view** dodać kolumnę zawierającą identyfikatory jednoznacznie identyfikujące obszary należące do różnych JPT. Jednym z rodzajów weryfikacji tego czy załadowane zostaną wszystkie dane jest sprawdzenie liczby rekordów w widoku **vau_import_view.** Powinna ona być zgodna z liczbą jednostek podziału terytorialnego które chcemy załadować.

Mereologia

Powszechnie stosowany zapis zależności mereologicznych w postaci: komórka tabeli po prawej stronie danej komórki w tym samym wierszu zawiera jednostkę nadrzędną, w wejściowym widoku bazodanowym **vau_import_view** przyjmuje podobną postać, ale należy jawnie wskazać nazwę kolumny zawierającej identyfikatory jednostek nadrzędnych. Identyfikatory VAU powstają w czasie ładowania danego (poprzedniego, wyższego) poziomu i można je wykorzystać jako dane wejściowe w czasie ładowania kolejnego poziomu. W tym celu należy wskazać kolumnę z identyfikatorami jednostek nadrzędnych w widoku **vau_import_view** za pomoca kolumny **nadrzedna**.

UWAGA: Należy odróżniać sytuacje szczególne kiedy JPT należy do JPT na innym poziomie niż bezpośrednio wyższy z powodu posiadania takiej wiedzy, od sytuacji braku wiedzy o JPT bezpośrednio nadrzędnej. W tej drugiej sytuacji należy w danych wejściowych umieszczać wartość "-" (myślnik), oznaczający istnienie JPT o nieznanych cechach.

UWAGA: Niedopuszczalna jest sytuacja istnienia kilku nazw różnych jednostek w jednym polu (Ryc. 1). W celu przechowania informacji o tym że dana jednostka (albo obręb) należy być może do jednej, albo do drugiej jednostki, należy utworzyć dwa osobne rekordy.

Oprócz nazwy i przypisania do zbioru geometrii, należy również podać w postaci osobnej kolumny bądź zbiorczo w postaci metadanych: datę początkową, końcową oraz typ dla każdego z poziomów.

gid	woj_p	powiat_p
7 861	rawskie	sochaczewski
7 862	rawskie	rawski
7 863	rawskie	sochaczewski, rawski
7 864	rawskie	rawski
7 865	łęczyckie	brzeziński

Ryc. 1. Błędna sytuacja opisania obrębu (AUA) dwoma jednostkami w tym samym rekordzie i polu.

Uruchamianie skryptu

Skrypt można uruchomić poleceniem SQL:

```
SELECT * FROM ontology.import_table_au(?src_table, ?jpt_name_field,
?jpt_vau_field, ?clear_tables);
```

podając odpowiednie parametry.

Przed uruchomieniem skryptu do tabeli wejściowej należy również dodać kolumnę **onto_import_log** do przechowania informacji z procesu ładowania.

Przykładowa struktura danych wejściowych:

```
CREATE TABLE public.jpt_koscielne_1600 (
  gid INTEGER NOT NULL,
  g_parafia VARCHAR(254),
  g_dekanat VARCHAR(50),
  g_archidia VARCHAR(50),
  g_diecezja VARCHAR(50),
  g_diecezja_vau INTEGER,
  g_archidia_vau INTEGER,
  g_dekanat_vau INTEGER,
  g_parafia_vau INTEGER,
  CONSTRAINT jpt_koscielne_1600_pkey PRIMARY KEY(gid)
)
```

które po przekształceniu do struktury zgodnej z widokiem bazodanowym **vau_import_view**:

```
SELECT
  gid as id,
  g_diecezja as nazwa,
  to_date(,1600-01-01',’YYYY-MM-DD’) as data_p,
  to_date(,1600-12-31',’YYYY-MM-DD’) as data_k,
  147 as typ_id,
  0 as vau_id
FROM
  public.jpt_koscielne_1600
WHERE
  g_diecezja IS NOT NULL
```

przyjmują postać podlegającą bezpośredniemu załadowaniu do bazy dla pojedynczego poziomu. W powyższym przypadku jest to diecezja.

Po załadowaniu pojedynczego poziomu jest on zwrotnie oznaczany w tabeli z danymi wejściowymi identyfikatorem VAU nadanym przez bazę danych, dla danych tworzących nowe VAU.

WAŻNE: ładowanie zaczynamy od JPT najwyższego poziomu.

Przed uruchomieniem skryptu i utworzeniem odpowienich manifestacji danego poziomu, można przejrzeć dane zawarte w widoku bazodanowym **au_import_view** które będą przetwarzane do manifestacji. Przykładowe dane wejściowe widoczne są na rycinie 2. Są one przed zasileniem przekształcane za pomocą widoku bazodanowego do postaci widocznej na rycinie 3. Przekształcenie odbywa się przed załadowaniem każdego poziomu oddzielnie.

gid	g_parafia	g_dekanat	g_archidia	g_diecezja	geom
12 418	Abramowice	Chodel	lubelski	krakowska	010600002
12 175	Abramowice	Chodel	lubelski	krakowska	010600002
22 951	Andrzejów	Nur	pułtuski	płocka	010600002
23 230	Andrzejów	Nur	pułtuski	płocka	010600002
54 012	Andrzejów	Andrzejów	krakowski	krakowska	010600002
42 837	Andrzejów	Andrzejów	krakowski	krakowska	010600002
22 867	Andrzejów	Nur	pułtuski	płocka	010600002
42 351	Andrzejów	Andrzejów	krakowski	krakowska	010600002
23 262	Andrzejów	Nur	pułtuski	płocka	010600002
35 895	Andrzejów	Nur	pułtuski	płocka	010600002
42 106	Andrzejów	Andrzejów	krakowski	krakowska	010600002
36 646	Andrzejów	Nur	pułtuski	płocka	010600002
42 410	Andrzejów	Andrzejów	krakowski	krakowska	010600002
35 998	Andrzejów	Nur	pułtuski	płocka	010600002
54 974	Andrzejów	Nur	pułtuski	płocka	010600002
24 246	Andrzejów	Nur	pułtuski	płocka	010600002
23 231	Andrzejów	Nur	pułtuski	płocka	010600002
42 835	Andrzejów	Andrzejów	krakowski	krakowska	010600002
23 121	Andrzejów	Nur	pułtuski	płocka	010600002
23 189	Andrzejów	Nur	pułtuski	płocka	010600002
23 124	Andrzejów	Nur	pułtuski	płocka	010600002

Ryc. 2. Tabela z danymi wejściowymi

nazwa	data_p	data_k	typ_id	vau_id
płocka	1600-01-01	1600-12-31	147	0
wrocławska	1600-01-01	1600-12-31	147	0
krakowska	1600-01-01	1600-12-31	147	0
poznańska	1600-01-01	1600-12-31	147	0
przemyska	1600-01-01	1600-12-31	147	0
gnieźnieńska	1600-01-01	1600-12-31	147	0
łucka	1600-01-01	1600-12-31	147	0
włocławska	1600-01-01	1600-12-31	147	0

Ryc. 3. Dane wejściowe z Ryc. 2 przekształcone do postaci pojedynczego poziomu diecezji w widoku bazodanowym **au_import_view**.

Przykład sekwencji czynności w celu załadowania danych z kilkoma poziomami.

1. Dostosowanie widoku bazodanowego **vau_import_view** do poziomu diecezji**.**

```
CREATE OR REPLACE VIEW public.vau_import_view (
    nazwa,
    data_p,
    data_k,
    typ_id,
    vau_id,
    nadrzedna)
```

```
AS
SELECT DISTINCT jpt_koscielne_1600.g_diecezja AS nazwa,
    to_date(,1600-01-01'::text, ,YYYY-MM-DD'::text) AS data_p,
    to_date(,1600-12-31'::text, ,YYYY-MM-DD'::text) AS data_k,
    147 AS typ_id,
    0 AS vau_id,
    0 AS nadrzedna
FROM jpt_koscielne_1600
WHERE jpt_koscielne_1600.g_diecezja IS NOT NULL;
```

Ponieważ w danych wejściowych nie została podana data początku i końca, podajemy ją arbitralnie**.** Podobnie, ponieważ nie został podany typ, ustalamy i podajemy typ dla tego poziomu w tym widoku bazodanowym. Jednostki nie istnieją w bazie ontologicznej więc vau_id wypełniamy zerami. Diecezje nie posiadają jednostek nadrzędnych, dlatego nadrzedna również wypełniamy zerami.

Po dostosowaniu widoku dobrze jest przeprowadzić wizualną weryfikację danych które widać w widoku do załadowania.

2. Wykonanie polecenia:

```
SELECT * FROM ontology.import_table_au(,public.jpt_koscielne_1600',
,g_diecezja',  ,g_diecezja_vau');
```

3. Dostosowanie widoku bazodanowego **vau_import_view** do poziomu archidiakonatów

```
CREATE OR REPLACE VIEW public.vau_import_view (
    nazwa,
    data_p,
    data_k,
    typ_id,
    vau_id,
    nadrzedna)
AS
SELECT DISTINCT jpt_koscielne_1600.g_archidia AS nazwa,
    to_date(,1600-01-01'::text, ,YYYY-MM-DD'::text) AS data_p,
    to_date(,1600-12-31'::text, ,YYYY-MM-DD'::text) AS data_k,
    146 AS typ_id,
    0 AS vau_id,
    g_diecezja_vau AS nadrzedna
FROM jpt_koscielne_1600
WHERE jpt_koscielne_1600.g_archidia IS NOT NULL;
```

Dla tego poziomu zmieniamy kolumnę z której będą pobierane dane z g_diecezja na g_archidia. Zmieniamy typ na 146. Jako nadrzedna podajemy kolumnę wypełnioną poprzednim cyklem: **g_diecezja_vau**.

4. Wykonanie polecenia:

```
    SELECT  *  FROM  ontology.import_table_au(,public.jpt_koscielne_1600',
,g_archidia',  ,g_archidia_vau');
```

5. Dostosowanie widoku bazodanowego **vau_import_view** do poziomu dekanatów

```
CREATE OR REPLACE VIEW public.vau_import_view (
    nazwa,
    data_p,
    data_k,
    typ_id,
    vau_id,
    nadrzedna)
AS
SELECT DISTINCT jpt_koscielne_1600.g_dekanat AS nazwa,
    to_date(,1600-01-01'::text, ,YYYY-MM-DD'::text) AS data_p,
    to_date(,1600-12-31'::text, ,YYYY-MM-DD'::text) AS data_k,
    145 AS typ_id,
    0 AS vau_id,
    COALESCE(g_archidia_vau,g_diecezja_vau) AS nadrzedna
FROM jpt_koscielne_1600
WHERE jpt_koscielne_1600.g_dekanat IS NOT NULL;
```

Dla tego poziomu zmieniamy kolumnę z której będą pobierane dane z g_archidia na g_dekanat. Zmieniamy typ na 145. Jako nadrzedna podajemy kolumnę wypełnioną poprzednim cyklem: **g_archidia_vau**.

6. Wykonanie polecenia: SELECT * FROM ontology.import_table_au(,public.jpt_koscielne_1600', ,g_dekanat', ,g_dekanat_vau');

7. Dostosowanie widoku bazodanowego **vau_import_view** do poziomu parafii

```
CREATE OR REPLACE VIEW public.vau_import_view (
    nazwa,
    data_p,
    data_k,
    typ_id,
    vau_id,
    nadrzedna)
AS
SELECT DISTINCT jpt_koscielne_1600.g_parafia AS nazwa,
    to_date(,1600-01-01'::text, ,YYYY-MM-DD'::text) AS data_p,
    to_date(,1600-12-31'::text, ,YYYY-MM-DD'::text) AS data_k,
    144 AS typ_id,
    0 AS vau_id,
     COALESCE(g_dekanat_vau,g_archidia_vau,g_diecezja_vau) AS
nadrzedna
FROM jpt_koscielne_1600
WHERE jpt_koscielne_1600.g_parafia IS NOT NULL;
```

Dla tego poziomu zmieniamy kolumnę z której będą pobierane dane na g_parafia. Zmieniamy typ na 144. Jako nadrzedna podajemy kolumnę wypełnioną poprzednim cyklem: **g_dekanat_vau**.

8. Wykonanie polecenia:

```
SELECT * FROM ontology.import_table_au(,public.jpt_koscielne_1600',
,g_parafia',  ,g_parafia_vau');
```

Rezultatem powyższej sekwencji czynności powinny być komplet manifestacji dla wszystkich poziomów podziału kościelnego obejmujący: manifestacje nazwy, manifestacje lokalizacji, mereologię.

Ładowanie danych zawierających miejscowości

W analogiczny sposób obsługiwane są miejscowości, dla których nazwa skryptu to: import_table_settlements a nazwa widoku: vs_import_view.

Skrypt ładujący dane tabelaryczne i przekształcający je w manifestacje, wersja dla jednostek podziału terytorialnego:

```
CREATE OR REPLACE FUNCTION ontology.import_table_au (
  src_table text,
  jpt_name_field text,
  jpt_vau_field text,
  clear_tables integer = 0
)
RETURNS void AS
$body$
DECLARE
  t timestamptz := clock_timestamp();
  r RECORD;
  info TEXT;
  typ TEXT;
  log_txt TEXT;
  newVAU INTEGER;
  newAUL INTEGER;
  query_txt TEXT;
BEGIN

RAISE INFO ,Start:%',clock_timestamp();

IF clear_tables THEN
  RAISE INFO ,Clearing tables...';
  EXECUTE ,DELETE FROM ontology.”AdministrativeUnitLocationsSets” WHE-
RE source=' ||quote_literal(src_table);
   EXECUTE ,DELETE FROM ontology.”AdministrativeUnitLocations” WHERE
source=' ||quote_literal(src_table);
  EXECUTE ,DELETE FROM ontology.”AdministrativeUnitNames” WHERE sour-
ce=' ||quote_literal(src_table);
   EXECUTE ,DELETE FROM ontology.”AdministrativeUnitMereologyLinks”
WHERE source=' ||quote_literal(src_table);
  EXECUTE ,DELETE FROM ontology.”VariableAdministrativeUnits” WHERE
source = ,||quote_literal(src_table);
END IF;
```

```
      log_txt = ,';
      log_txt = log_txt||clock_timestamp()||' ,;
      FOR r IN
        SELECT
            nazwa,
            data_p,
            data_k,
            typ_id,
            vau_id,
            nadrzedna
        FROM
            public.vau_import_view v
      LOOP
        RAISE INFO ,nazwa: %',r.nazwa;

        --
        -- Dodanie lub wyznaczenie VAU
        --

       IF r.vau_id=0 THEN
           --wstawienie nowego VAU. Powstaje nowa tożsamość (jpt nie istniała
wczesniej w bazie pod żadną postacią).
            query_txt = ,INSERT INTO ontology.'
            || quote_ident(,VariableAdministrativeUnits')
            || ,(, || quote_ident(,Names')
            || ,,' || quote_ident(,AdministrativeUnitTypeIdentifiers')
            || ,,' || quote_ident(,source')
             || ,)' || , VALUES (,|| quote_literal(r.nazwa) || ,,' ||r.typ_
id|| ,,' ||quote_literal(src_table)|| , ) RETURNING , || quote_ident(,Iden-
tifiers');
             EXECUTE query_txt INTO newVAU; --VAU do którego należy dokleić
manifestacje
            RAISE INFO ,VAU inserted: %',newVAU;
            log_txt = log_txt||' VAU inserted.';

       --aktualizacja tabeli źródłowej o nowy VAU

         EXECUTE ,UPDATE ,||src_table||' SET ,||jpt_vau_field||' = ,||newVAU||'
WHERE ,||jpt_name_field||'='||quote_literal(r.nazwa);

        ELSE
            IF r.vau_id>0 THEN
               newVAU = r.vau_id; --VAU do którego należy dokleić manifesta-
cje
            ELSE
                RAISE INFO ,VAU unknown: %',r.vau_id;
                log_txt = log_txt||' Warning: VAU unknown! ,;
               CONTINUE;
            END IF;
        END IF;
```

```
      -- W tym miejscu mamy w newVAU identyfikator istniejącego wcześniej lub
nowego VS do którego należy dodać manifestacje cząstkowe
        --
        -- Dodanie manifestacji nazwy
        --

        RAISE INFO ,Name datap:%', r.data_p;
        RAISE INFO ,Name datak:%', r.data_k;
        IF r.nazwa is NOT NULL THEN
             EXECUTE format(,INSERT INTO ontology."AdministrativeUnitNames"
(„Names", „VariableAdministrativeUnitIdentifiers", „StartsAt", „EndsAt", „so-
urce")
               VALUES (%s,%L,%s,%s,%s)',quote_literal(string_to_array(r.na-
zwa,'@')), newVAU, quote_literal(r.data_p), quote_literal(r.data_k), quote_
literal(src_table));
            RAISE INFO ,Name manifestation inserted: %',r.nazwa;
            log_txt = log_txt||' Name manifestation inserted.';
        END IF;

        --
        -- Dodanie manifestacji lokalizacji
        --

        RAISE INFO ,Location datap:%', r.data_p;
        RAISE INFO ,Location datak:%', r.data_k;
        IF r.nazwa is NOT NULL THEN
             EXECUTE format(,INSERT INTO ontology."AdministrativeUnitLoca-
tions" ( „VariableAdministrativeUnitIdentifiers", „StartsAt", „EndsAt", „so-
urce")
           VALUES (%L,%s,%s,%s)', newVAU, quote_literal(r.data_p), quote_li-
teral(r.data_k), quote_literal(src_table))|| ,  RETURNING , || quote_iden-
t(,Identifiers') INTO newAUL;

           -- w newAUL jest id manifestacji cząstkowej, a tabela źródłowa ma
juz wpisany VAU dla wszystkich obrębów

            EXECUTE ,INSERT INTO ontology."AdministrativeUnitLocationsSets"
(geom_id,obreb,source)
             SELECT ,||newAUL||', gid,'|| quote_literal(src_table)||' FROM
,||src_table||'
            WHERE ,||jpt_vau_field||' = ,||newVAU;

            RAISE INFO ,Location manifestation inserted: %',r.nazwa;
            log_txt = log_txt||' Location manifestation inserted.';
        END IF;
```

```
        --
        -- Dodanie manifestacji przynależności mereologicznej
        --

        IF (r.nadrzedna is not NULL) AND (r.nadrzedna>0) THEN
            --wstawienie jpt nadrzednej
              RAISE INFO ,Mereology manifestation inserting...WholeId:%',r.
nadrzedna;
            RAISE INFO ,Mereology manifestation inserting...PartId%',newVAU;
            EXECUTE format(,INSERT INTO ontology.”AdministrativeUnitMereolo-
gyLinks” („PartIdentifiers”, „WholeIdentifiers”, „StartsAt”, „EndsAt”, „sour-
ce”) VALUES (%L,%L,%s,%s,%s)',newVAU, r.nadrzedna, quote_literal(r.data_p),
quote_literal(r.data_k), quote_literal(src_table));
            RAISE INFO ,Mereology manifestation inserted: %',r.nadrzedna;
            log_txt = log_txt||' Mereology manifestation inserted.';
        END IF;

         EXECUTE ,UPDATE ,||src_table||' SET onto_import_log='||quote_lite-
ral(log_txt)||' WHERE ,||jpt_name_field||'='||quote_literal(r.nazwa);
        log_txt = ,';
        log_txt = log_txt||clock_timestamp()||' ,;
        RAISE INFO ,Next';

      END LOOP;
      RAISE INFO ,Completed sucessfully.';
      RAISE INFO ,Time spent=%', clock_timestamp() - t;

      EXCEPTION
      WHEN OTHERS THEN
        RAISE NOTICE ,NOT completed sucessfully.';
        RAISE NOTICE ,% %', SQLERRM, SQLSTATE;
        RAISE NOTICE ,%', query_txt;
        RAISE NOTICE ,Time spent=%', clock_timestamp() - t;
      END;
      $body$
      LANGUAGE ,plpgsql'
      VOLATILE
      CALLED ON NULL INPUT
      SECURITY INVOKER
      PARALLEL UNSAFE
      COST 100;
```

Skrypt ładujący dane tabelaryczne i przekształcający je w manifestacje, wersja dla miejscowości:

```
CREATE OR REPLACE FUNCTION ontology.import_table_settlements (
  src_table text,
  jpt_name_field text,
  jpt_vau_field text,
  clear_tables integer = 0
)
RETURNS void AS
$body$
DECLARE
  t timestamptz := clock_timestamp();
  r RECORD;
  info TEXT;
  typ TEXT;
  log_txt TEXT;
  newVAU INTEGER;
  newAUL INTEGER;
  query_txt TEXT;
BEGIN

RAISE INFO ,Start:%',clock_timestamp();

IF clear_tables THEN
  RAISE INFO ,Clearing tables...';
  EXECUTE ,DELETE FROM ontology."AdministrativeUnitLocationsSets" WHE-
RE source=' ||quote_literal(src_table);
   EXECUTE ,DELETE FROM ontology."AdministrativeUnitLocations" WHERE
source=' ||quote_literal(src_table);
  EXECUTE ,DELETE FROM ontology."AdministrativeUnitNames" WHERE sour-
ce=' ||quote_literal(src_table);
   EXECUTE ,DELETE FROM ontology."AdministrativeUnitMereologyLinks"
WHERE source=' ||quote_literal(src_table);
  EXECUTE ,DELETE FROM ontology."VariableAdministrativeUnits" WHERE
source = ,||quote_literal(src_table);
END IF;

log_txt = ,';
log_txt = log_txt||clock_timestamp()||' ,;
FOR r IN
  SELECT
      nazwa,
      data_p,
      data_k,
      typ_id,
      vau_id,
      jpt_mereology
  FROM
      public.vau_import_view v
LOOP
  RAISE INFO ,nazwa: %',r.nazwa;
```

```
        --
        -- Dodanie lub wyznaczenie VAU
        --
        IF r.vau_id=0 THEN
           --wstawienie nowego VAU. Powstaje nowa tożsamość (jpt nie istniała
wczesniej w bazie pod żadną postacią).
            query_txt = ,INSERT INTO ontology.'
            || quote_ident(,VariableAdministrativeUnits')
            || ,(, || quote_ident(,Names')
            || ,,' || quote_ident(,AdministrativeUnitTypeIdentifiers')
            || ,,' || quote_ident(,source')
             || ,)' || , VALUES (,|| quote_literal(r.nazwa) || ,,' ||r.typ_
id|| ,,' ||quote_literal(src_table)|| , ) RETURNING , || quote_ident(,Iden-
tifiers');
             EXECUTE query_txt INTO newVAU; --VAU do którego należy dokleić
manifestacje
            RAISE INFO ,VAU inserted: %',newVAU;
            log_txt = log_txt||' VAU inserted.';

            --aktualizacja tabeli źródłowej o nowy VAU

         EXECUTE ,UPDATE ,||src_table||' SET ,||jpt_vau_field||' = ,||newVAU||'
WHERE ,||jpt_name_field||'='||quote_literal(r.nazwa);

        ELSE
            IF r.vau_id>0 THEN
                 newVAU = r.vs; --VAU do którego należy dokleić manifesta-
cje
            ELSE
                RAISE INFO ,VAU unknown: %',r.vau_id;
                log_txt = log_txt||' Warning: VAU unknown! ,;
               CONTINUE;
            END IF;
        END IF;

        -- W tym miejscu mamy w newVAU identyfikator istniejącego wcześniej
lub nowego VS do którego należy dodać manifestacje cząstkowe
        --
        -- Dodanie manifestacji nazwy
        --

        RAISE INFO ,Name datap:%', r.data_p;
        RAISE INFO ,Name datak:%', r.data_k;
        IF r.nazwa is NOT NULL THEN
             EXECUTE format(,INSERT INTO ontology."AdministrativeUnitNames"
(„Names", „VariableAdministrativeUnitIdentifiers", „StartsAt", „EndsAt", „so-
urce")
               VALUES  (%s,%L,%s,%s,%s)',quote_literal(string_to_array(r.na-
zwa,'@')), newVAU, quote_literal(r.data_p), quote_literal(r.data_k), quote_
literal(src_table));
            RAISE INFO ,Name manifestation inserted: %',r.nazwa;
            log_txt = log_txt||' Name manifestation inserted.';
        END IF;
```

```
        --
        -- Dodanie manifestacji lokalizacji
        --

        RAISE INFO ,Location datap:%', r.data_p;
        RAISE INFO ,Location datak:%', r.data_k;
        IF r.nazwa is NOT NULL THEN
             EXECUTE format(,INSERT INTO ontology."AdministrativeUnitLoca-
tions" ( „VariableAdministrativeUnitIdentifiers", „StartsAt", „EndsAt", „so-
urce")
           VALUES (%L,%s,%s,%s)', newVAU, quote_literal(r.data_p), quote_li-
teral(r.data_k), quote_literal(src_table))|| ,  RETURNING , || quote_iden-
t(,Identifiers') INTO newAUL;

           -- w newAUL jest id manifestacji cząstkowej, a tabela źródłowa ma
juz wpisany VAU dla wszystkich obrębów

            EXECUTE ,INSERT INTO ontology."AdministrativeUnitLocationsSets"
(geom_id,obreb,source)
             SELECT ,||newAUL||', gid,'|| quote_literal(src_table)||' FROM
,||src_table||'
            WHERE ,||jpt_vau_field||' = ,||newVAU;

            RAISE INFO ,Location manifestation inserted: %',r.nazwa;
            log_txt = log_txt||' Location manifestation inserted.';
        END IF;

        --
        -- Dodanie manifestacji przynależności mereologicznej
        --

        IF (r.jpt_mereology is not NULL) AND (r.jpt_mereology>0) THEN
            --wstawienie jpt nadrzednej
             RAISE INFO ,Mereology manifestation inserting...WholeId:%',r.
jpt_mereology;
            RAISE INFO ,Mereology manifestation inserting...PartId%',newVAU;
            EXECUTE format(,INSERT INTO ontology."AdministrativeUnitMereolo-
gyLinks" („PartIdentifiers", „WholeIdentifiers", „StartsAt", „EndsAt", „sour-
ce") VALUES (%L,%L,%s,%s,%s)',newVAU, r.jpt_mereology, quote_literal(r.da-
ta_p), quote_literal(r.data_k), quote_literal(src_table));
           RAISE INFO ,Mereology manifestation inserted: %',r.jpt_mereology;
            log_txt = log_txt||' Mereology manifestation inserted.';
        END IF;

        EXECUTE ,UPDATE ,||src_table||' SET onto_import_log='||quote_lite-
ral(log_txt)||' WHERE ,||jpt_name_field||'='||quote_literal(r.nazwa);
        log_txt = ,';
        log_txt = log_txt||clock_timestamp()||' ,;
        RAISE INFO ,Next';

      END LOOP;
      RAISE INFO ,Completed sucessfully.';
      RAISE INFO ,Time spent=%', clock_timestamp() - t;
```

```
EXCEPTION
WHEN OTHERS THEN
  RAISE NOTICE ,NOT completed sucessfully.';
  RAISE NOTICE ,% %', SQLERRM, SQLSTATE;
  RAISE NOTICE ,%', query_txt;
  RAISE NOTICE ,Time spent=%', clock_timestamp() - t;
END;
$body$
LANGUAGE ,plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
PARALLEL UNSAFE
COST 100;
```