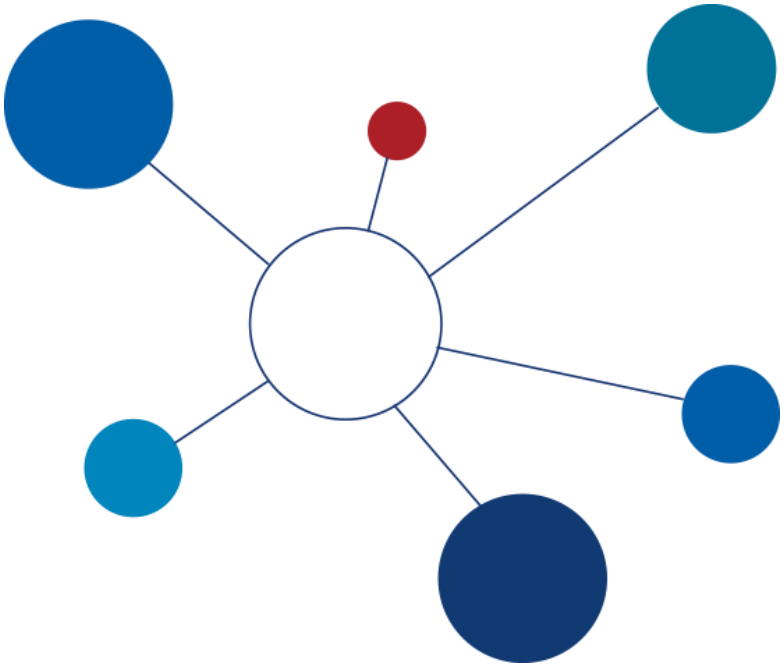


Technologie Zasilania i Odświeżania Hurtowni Danych

laboratorium

część V

v20170324

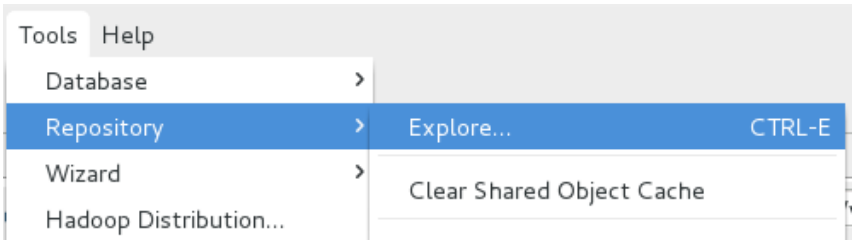
I. WYKORZYSTANIE ZADAŃ JAKO ZESTAWU TRANSFORMACJI.

W ramach poprzednich zestawów ćwiczeń stworzyliśmy zestaw transformacji przeznaczonych do zasilania hurtowni danych na podstawie dwóch źródeł danych. Hurtownia danych wymaga oczywiście regularnego zasilania. Uruchamianie kolejno wszystkich zdefiniowanych przez nas transformacji, w każdym przypadku gdy hurtownia będzie wymagała odświeżenia swojej zawartości byłoby uciążliwe. Aby to ułatwić utworzymy teraz tzw. zadanie (*job*), którego zadaniem będzie uruchomienie wszystkich transformacji we właściwej kolejności.

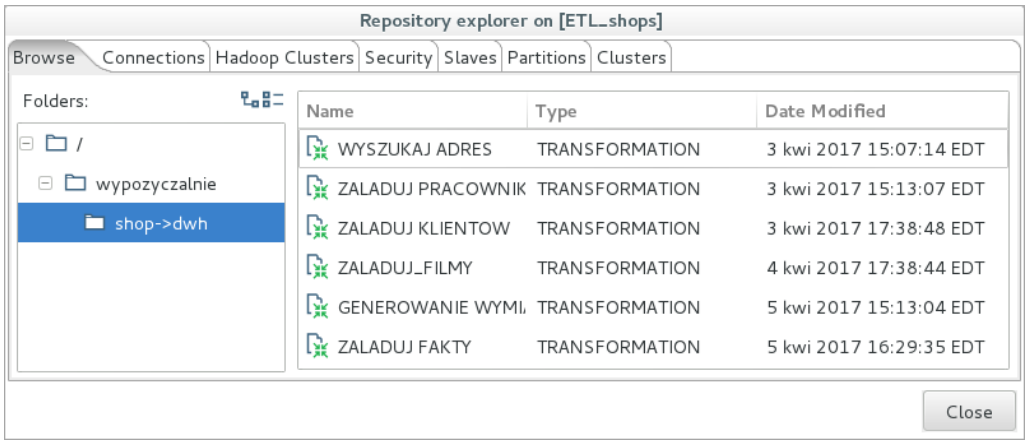
1. Na początku zobaczmy jakie transformacje udało się nam do tej pory wykonać.

   a) W tym celu uruchommy *Pentaho DI* i zalogujmy się do jego repozytorium.

   b) Korzystając z menu *Tools->Repository->Explore* otwórzmy narzędzie *Repository explorer*.



   c) Przejdźmy do katalogu `wypozyczalnie/shop->dwh` i zobaczmy listę stworzonych przez nas transformacji.

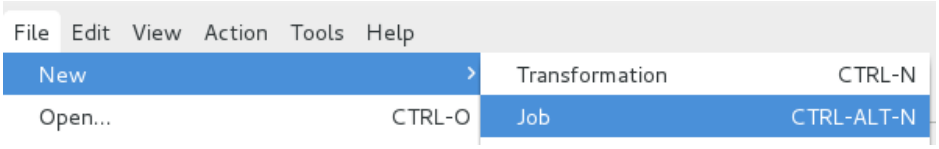


   d) Dwie z naszych transformacji mają charakter techniczny. `GENEROWANIE WYMIARU DATY` uruchomiliśmy raz i nie ma potrzeby uruchamiać go wielokrotnie. `WYSZUKAJ ADRES` jest transformacją podrzędną dla innych transformacji. Pozostałe transformacje powinny być natomiast składowymi procedury odświeżania hurtowni danych.

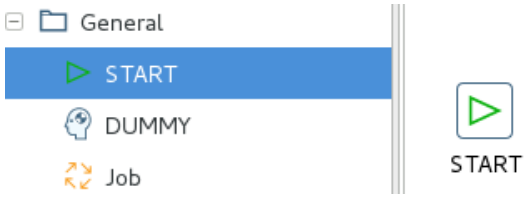
   e) Przy pomocy przycisku *Close* zamknijmy narzędzie *Repository explorer*.

2. Rozpocznijmy zatem utworzenie odpowiedniego zadania, które będzie wywoływało sekwencję stosownych transformacji.

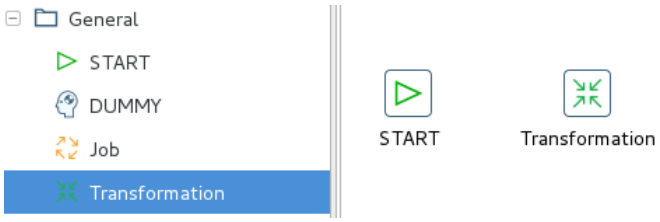
   a) Przy pomocy menu *File->New->Job* utwórzmy nowe zadanie.



   b) Rozpoczniemy nasze zadanie od komponentu *START* z katalogu *General*.

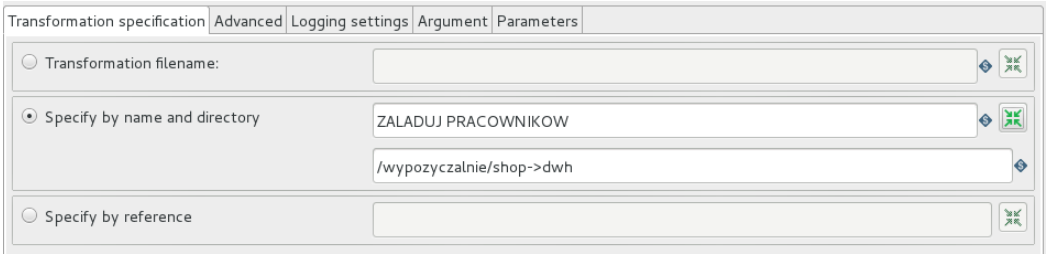


c) Kolejnym krokiem będzie komponent *Transformation* z tego samego katalogu.



d) Połączmy oba komponenty tak jak poprzednio łączyliśmy kroki w transformacjach.

e) Wejdźmy do edycji własności komponentu *Transformation*. Pierwszą transformacją jaką będziemy wywoływać w naszym zadaniu, będzie pierwsza transformacja, którą stworzyliśmy – `ZALADUJ PRACOWNIKOW`. Wskaż ją korzystając z przycisku pozwalającego na wybór odpowiedniej transformacji.

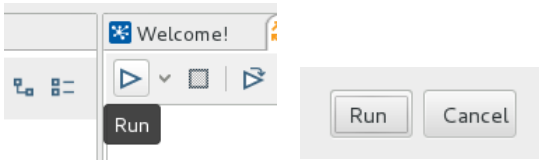


f) Zamknijmy własności nowego komponentu. Zwróćmy uwagę, że jego nazwa automatycznie uległa modyfikacji.

g) W analogiczny sposób dołóżmy do naszego zadania trzy kolejne transformacje łącząc je w sekwencję przedstawioną poniżej.

h) Na zakończenie dołóżmy do naszej sekwencji komponent *Success* z katalogu *General*. Połączmy wyjście ostatniej transformacji z tym nowo dodanym komponentem.

i) Zapiszmy utworzone zadanie w katalogu `/wypozyczalnie/shop->dwh` jako `ODSWIEZ DWH`. Uruchommy je w celu sprawdzenia poprawnego działania.



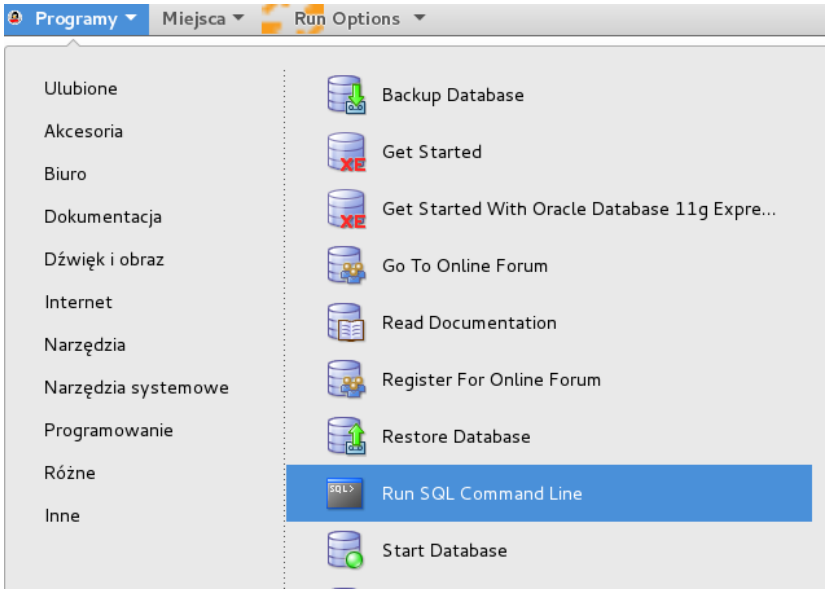
Execution Results

History | Logging | Job metrics | Metrics

Job / Job Entry	Comment	Result	Reason
ODSWIEZ DWH			
Job: ODSWIEZ DWH	Start of job execution		start
START	Start of job execution		start
START	Job execution finished	Success	
ZALADUJ PRACOWNIKOW	Start of job execution		Followed unconditional link
ZALADUJ PRACOWNIKOW	Job execution finished	Success	
ZALADUJ KLIENTOW	Start of job execution		Followed link after success
ZALADUJ KLIENTOW	Job execution finished	Success	
ZALADUJ_FILMY	Start of job execution		Followed link after success
ZALADUJ_FILMY	Job execution finished	Success	
ZALADUJ FAKTY	Start of job execution		Followed link after success
ZALADUJ FAKTY	Job execution finished	Success	
Job: ODSWIEZ DWH	Job execution finished	Success	finished

3. Na podstawie powyższego wyniku wiemy, że nasze zadanie działa poprawnie, ale spróbujmy jeszcze zaobserwować wyniki jego działania.

   a) W tym celu zalogujemy się do naszej hurtowni danych. Z pomocą menu start uruchommy *SQL*Plus*.



   b) Zalogujmy się do naszej hurtowni danych korzystając z polecenia `connect dwh/dwh@xe`.

```
SQL> connect dwh/dwh@xe
Connected.
```

   c) Wykonamy następujące zapytanie w celu sprawdzenia liczby krotek w tabeli `wypozyczenia`:

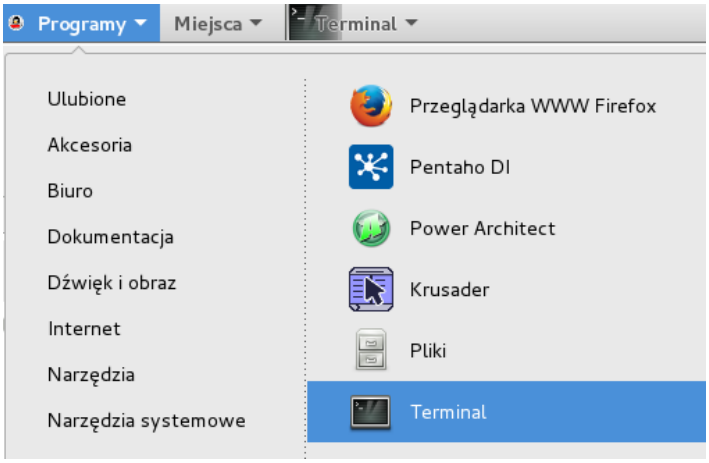
```
select count(*) from wypozyczenia;
```

```
SQL> select count(*) from wypozyczenia;

  COUNT(*)
----------
      3467

SQL>
```

d) Nie wychodząc z programu *SQL*Plus* wykonamy teraz dwie rzeczy. Pierwszą z nich będzie zasymulowanie upływu czasu w naszych źródłach danych. Drugą z nich będzie ponowne uruchomienie zadania `ODSWIEZ DWH`. W efekcie powinniśmy zaobserwować zmiany w zawartości hurtowni danych. Na początku zasymulujmy upływ czasu. W tym celu korzystając z menu start uruchommy terminal poleceń



e) Przejdźmy do katalogu `labs` wykonując polecenie:

```
cd labs
```

Następnie uruchommy skrypt `zasilanie2.sh` za pomocą polecenia:

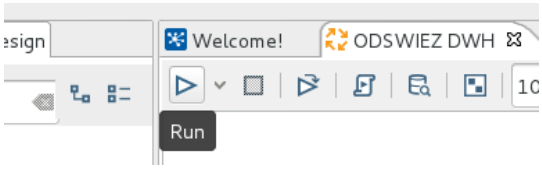
```
./zasilanie2.sh
```

```
[etl@localhost ~]$ cd labs
[etl@localhost labs]$ ./zasilanie2.sh
```

f) Zasilanie źródeł uzupełnia dane w źródłach o kolejną partię danych obejmującą określony okres czasu i dane które z tym okresem czasu są związane – nowych klientów, nowe wypożyczenia itp. Czas przetwarzania może być różny, ale zazwyczaj w ciągu 2-3 minut procedura się kończy.

```
2017/04/06 14:23:02 - zasil_zrodlo - Finished job entry [adresy i filmy] (result=[true])
2017/04/06 14:23:02 - zasil_zrodlo - Job execution finished
2017/04/06 14:23:02 - Kitchen - Finished!
2017/04/06 14:23:02 - Kitchen - Start=2017/04/06 14:21:32.967, Stop=2017/04/06 14:23:02.741
2017/04/06 14:23:02 - Kitchen - Processing ended after 1 minutes and 29 seconds (89 seconds total).
[etl@localhost labs]$
```

g) Po zakończonym przetwarzaniu przełączmy się do narzędzia *Pentaho DI* i uruchommy zdefiniowane przez nas zadanie `ODSWIEZ DWH`.



h) Po zakończonym odświeżaniu hurtowni danych, ponownie wykonajmy nasze zapytanie w programie *SQL*Plus*. Możemy w tym celu skorzystać z instrukcji `r`, która uruchamia poprzednie polecenie SQL.

```
SQL> select count(*) from wypozyczenia;

  COUNT(*)
----------
      3467

SQL> r
  1* select count(*) from wypozyczenia

  COUNT(*)
----------
     10176
```

i) Zmiana odczytanej wartości może nas upewnić w przeświadczeniu, że tak zadanie jak i wszystkie zdefiniowane przez nas transformację spełniają swoją rolę.

II. WYKORZYSTANIE ARKUSZA KALKULACYJNEGO JAKO ŹRÓDŁA DANYCH.

Do tej pory jako źródła danych wykorzystywaliśmy systemy baz danych. W przypadku zasilania hurtowni danych wykorzystywane są, z oczywistych względów, także inne typy źródeł. W szczególności mogą to być tekstowe pliki *CSV* lub arkusze kalkulacyjne.

Pliki takie często pochodzą z zewnątrz i stanowią bardzo ważne źródło dodatkowych wymiarów czy ich atrybutów.

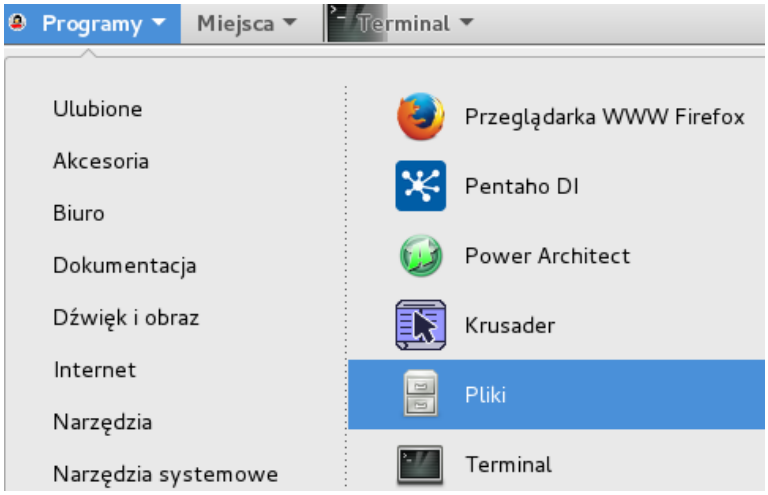
W ramach ćwiczenia wykorzystamy plik *CSV* przechowujący informacje pozwalającą na rozszerzenie wymiaru klient o dodatkowe atrybuty adresowe dotyczące stanu oraz okręgu zamieszkania klienta.

1. Na początku przyglądniemy się naszemu nowemu źródłu danych, ale zanim to zrobimy przypomnijmy sobie jak wygląda nasza hurtownia danych w zakresie wymiaru klient.

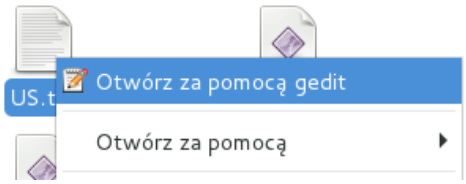
   a) Nasz wymiar klient jest dość ubogi w zakresie danych adresowych (geograficznych). Kod pocztowy i miasto oraz nazwa ulicy nie pozwalają na zaawansowaną analizę. Dlatego chcielibyśmy to zmienić.

```
SQL> desc klienci
 Name
 ---------------------------
 KL_KLIENT_ID
 KL_KOD_POCZTOWY
 KL_MIASTO
 KL_IMIE
 KL_NAZWISKO
 KL_EMAIL
 KL_ULICA
 KL_OSTATNIA_MODYFIKACJA
```

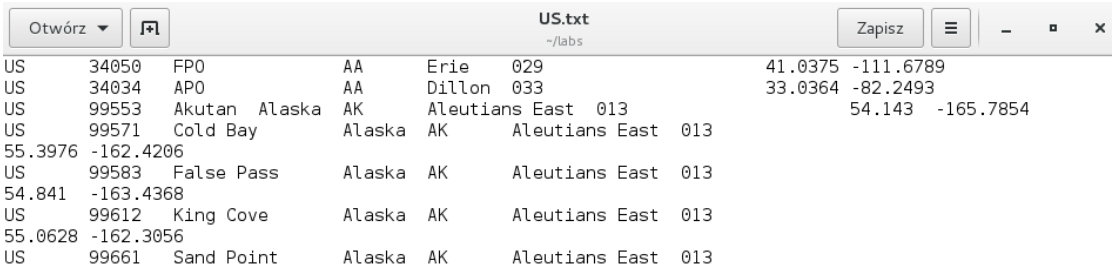
   b) Wykorzystamy do tego celu dane teleadresowe pochodzące z portalu `www.geonames.org`. Dostarcza on wielu informacji, które mają postać zarówno plików *CSV* jak i różnorodnych usług sieciowych. W naszym ćwiczeniu wykorzystamy pliki *CSV*. Zobaczmy co zawierają. Otwórz za pomocą menu *Programy* przeglądarkę plików



c) Przejdźmy do katalogu `/home/etl/labs`, a następnie za pomocą menu kontekstowego otwórzmy plik `US.txt`.



d) Jak widać plik ten jest plikiem strukturalnym.



e) Wg informacji na stronie `http://download.geonames.org/export/zip/`, będącej źródłem tego pliku, każdy wiersz tego pliku zawiera następujące kolumny:

- *country code* – kod kraju iso
- *postal code* – kod pocztowy
- *place name* – nazwa miejscowości
- *admin name1* – stan
- *admin code1* – kod stanu
- *admin name2* – okręg (prowincja)
- *admin code2* – kod okręgu (prowincji)
- *admin name3* – gmina (jednostka terytorialna)
- *admin code3* – kod gminy (jednostki terytorialnej)
- *latitude* – szerokość geograficzna (wgs84)
- *longitude* – długość geograficzna (wgs84)
- *accuracy* – dokładność określenia długości I szerokości

f) Dane zawarte w pliku `US.txt` dotyczą terytorium Stanów Zjednoczonych. W naszym przypadku to właściwe źródło danych, gdyż kody i miasta wykorzystywane w naszych źródłowych danych dotyczą właśnie tego kraju.

g) Wykorzystamy te dane do uzupełnienia naszej hurtowni o dodatkowe atrybuty wymiaru klienci.

2. Opracowanie i implementacja modyfikacji schematu hurtowni danych

   a) Przypomnieliśmy sobie przed chwilą jak wygląda schemat naszej hurtowni danych w zakresie wymiaru klienci. Nowy zestaw danych pochodzący z zewnętrznego pliku *CSV* będzie przydatny do jego rozszerzenia. Nowe atrybuty tego wymiaru, które chcemy utworzyć to stan i okręg. Dokonajmy zatem odpowiednich modyfikacji. Najprostszy sposób to wykorzystanie oprogramowania *SQL*Plus*. Jeżeli zamknąłeś konsolę z połączeniem do bazy danych, z menu *Programy/Inne* wybierz *Run SQL Command Line*.

   b) Zaloguj się jako właściciel schematu hurtowni danych za pomocą następującego polecenia `connect dwh@xe/dwh`.

   a. Sprawdź strukturę tabeli klienci korzystając z polecenia:
   `desc klienci`

```
SQL> desc klienci
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 KL_KLIENT_ID                              NOT NULL NUMBER
 KL_KOD_POCZTOWY                                    VARCHAR2(10)
 KL_MIASTO                                 NOT NULL VARCHAR2(50)
 KL_IMIE                                   NOT NULL VARCHAR2(45)
 KL_NAZWISKO                               NOT NULL VARCHAR2(45)
 KL_EMAIL                                  NOT NULL VARCHAR2(50)
 KL_ULICA                                  NOT NULL VARCHAR2(50)
 KL_OSTATNIA_MODYFIKACJA                   NOT NULL TIMESTAMP(6)
```

   c) Wprowadźmy zmiany za pomocą następujących poleceń:

```
alter table klienci add KL_STAN VARCHAR2(100);
alter table klienci add KL_OKREG VARCHAR2(100);
```

```
SQL> alter table klienci add KL_STAN VARCHAR2(100);

Table altered.

SQL> alter table klienci add KL_OKREG VARCHAR2(100);

Table altered.
```

   d) Z oczywistych powodów zmiany w schemacie hurtowni danych powinny

   - być udokumentowane
   - uwzględniać istniejące transformacje
   - być przetestowane

   e) Zmiany, które wprowadziliśmy nie powinny wpływać negatywnie na istniejące już transformacje.

3. Zanim rozpoczniemy implementację transformacji , która wykorzysta nowy zestaw danych, musimy zwrócić uwagę na kilka faktów:

   - hurtownia danych już istnieje, dlatego nasza nowa transformacja powinna ten fakt uwzględniać i uzupełnić jej zawartość
   - zawartość arkusza kalkulacyjnego może się zmieniać w czasie, dlatego nie wystarczy aktualizować tylko danych, które nie posiadają określonego stanu i okręgu, ale także należy uwzględnić konieczność modyfikacji tych danych, które uległy zmianie (np. w wyniku zmian administracyjnych)
   - plik *CSV* nie posiada funkcjonalności, która pozwoliłaby wskazać zmienione składowe – musimy opracować metodę, która pozwoli je wykryć

   Uwzględniając powyższe fakty opracujemy i zaimplementujemy w rzeczywistości dwie transformacje.

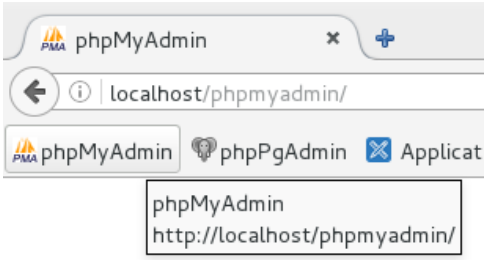
   Pierwsza z nich będzie przeznaczona do wykrywania zmian w źródle. Jej rezultatem będą odpowiednie

znaczniki tych danych, które zostały zmienione pomiędzy kolejnymi aktualizacjami hurtowni danych.

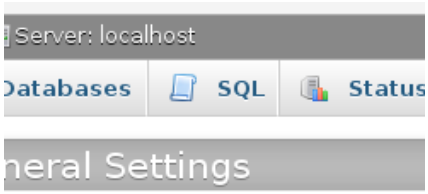
Druga z transformacji będzie aktualizowała dane wymiaru zarówno tych klientów, którzy nie posiadają określonego stanu i okręgu (np. pojawiły się w ramach ostatniego odświeżenia hurtowni danych) jak i tych klientów, dla których dane dotyczące stanu i okręgu uległy zmianie.

4. Aby wykryć zmiany w źródle danych jakim jest *CSV* wykorzystamy technikę *CDC* opartą na porównywaniu aktualnej postaci źródła z poprzednio zarejestrowanym (*Snapshot-Based CDC*). Naszą poprzednio zarejestrowaną postać źródła będziemy przechowywali w tabeli `geonames_org` w schemacie `ods`, który będzie służył jako operacyjna składnica danych (*operational data store*).

   a) Na początku utworzymy tabelę `geonames_org`. W tym celu za pomocą przeglądarki zaloguj się do bazy danych *MySQL* jako użytkownik `ods` z hasłem `ods`.



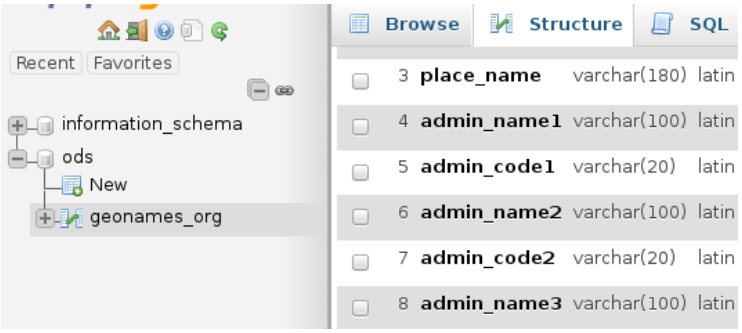
   b) Następnie wybierzmy bazę danych (schemat) `ods`, po czym przejdźmy do wprowadzania poleceń SQL.



   c) Jako treść polecenia SQL wprowadź poniższe polecenie. Struktura utworzonej tabeli odpowiada strukturze zawartości arkusza kalkulacyjnego (pliku *CSV*). Dodatkowe dwie kolumny będą pełniły rolę znaczników zmian.

```
create table geonames_org (
  country_code varchar(2),
  postal_code  varchar(20),
  place_name   varchar(180),
  admin_name1  varchar(100),
  admin_code1  varchar(20),
  admin_name2  varchar(100),
  admin_code2  varchar(20),
  admin_name3  varchar(100),
  admin_code3  varchar(20),
  latitude     decimal(7,4),
  longitude    decimal(7,4),
  accuracy     decimal(7,4),
  inserted     char(1) default 'T',
  updated      char(1) );
```

   d) Wykonajmy polecenie, a następnie sprawdź strukturę nowo utworzonej tabeli.

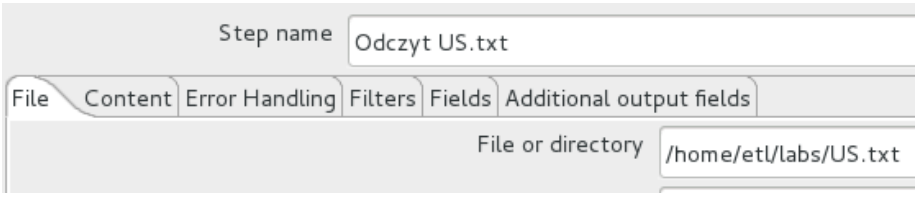


5. Czas na rozpoczęcie definiowania transformacji.

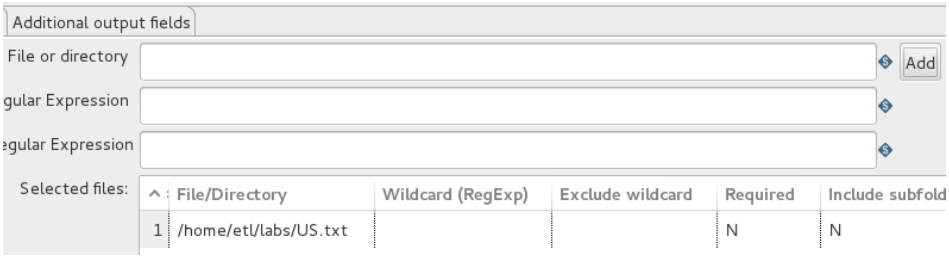
   a) Utwórzmy nową transformację.

   b) Aby móc porównać poprzednią postać danych z nową wersją należy dokonać odczytu obu z nich. W tym celu pierwszym krokiem transformacji będzie *Text file input* z katalogu *Input* odczytujący dane z pliku *CSV*.

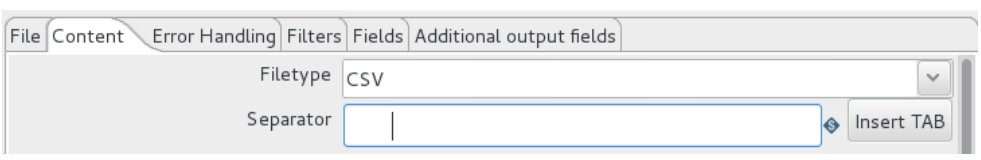
   c) Otwórzmy własności tego kroku. Zmieńmy jego nazwę na `Odczyt US.txt`. Na zakładce *File* wybierzmy przycisk *Browse* w celu wskazania położenia pliku `/home/etl/labs/US.txt`.



   d) Następnie przyciskiem *Add* dodajmy wskazany plik do listy wybranych plików.

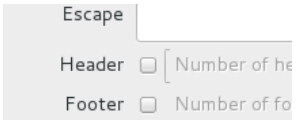


   e) Aby zdefiniować strukturę zawartości pliku przejdźmy na zakładkę *Content*. Skasujmy w polu Separator domyślny znak średnika, a następnie za pomocą przycisku *Insert TAB* wstawmy w to miejsce znak tabulatora.

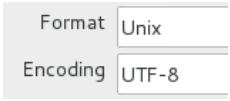


   f) Na tej samej zakładce odznaczmy pole wyboru *Header* – nasze źródło danych nie posiada wiersza

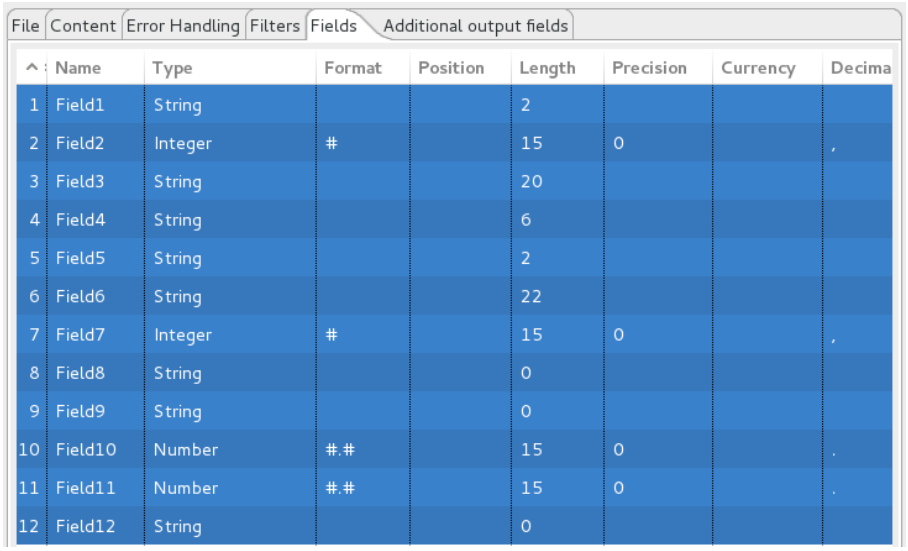
nagłówka.



g) W dalszym ciągu nie zmieniając zakładki zmieńmy format pliku na *Unix* oraz stronę kodową na *UTF-8*.

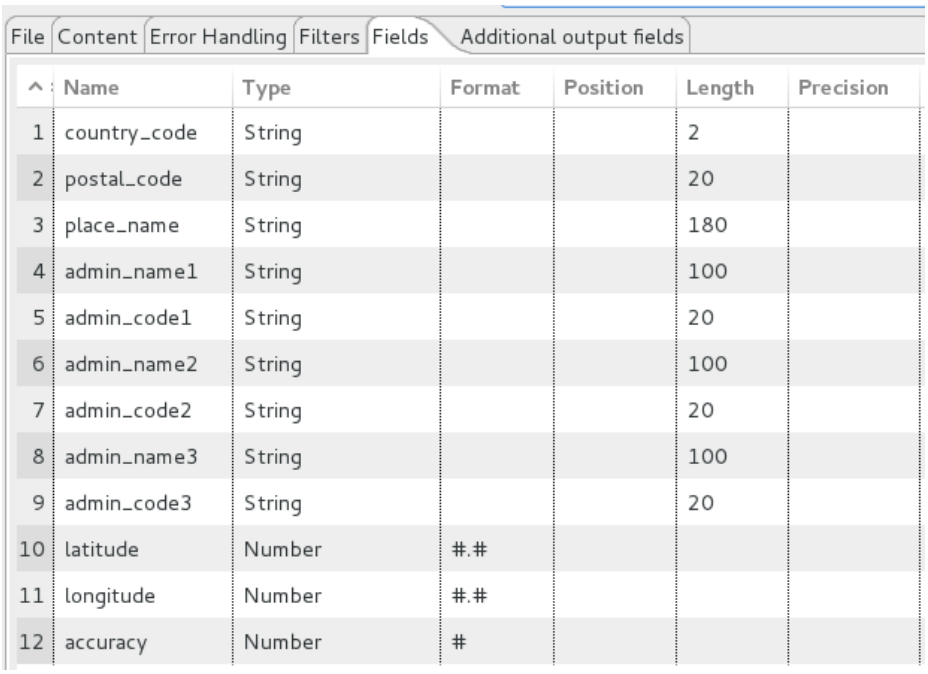


h) Przejdźmy na zakładkę *Fields*. Przy pomocy przycisku *Get Fields* dokonajmy analizy 100 pierwszych wierszy w naszym pliku celem określenia jego budowy.


File | Content | Error Handling | Filters | Fields | Additional output fields

	Name	Type	Format	Position	Length	Precision	Currency	Decima
1	Field1	String			2			
2	Field2	Integer	#		15	0		,
3	Field3	String			20			
4	Field4	String			6			
5	Field5	String			2			
6	Field6	String			22			
7	Field7	Integer	#		15	0		,
8	Field8	String			0			
9	Field9	String			0			
10	Field10	Number	#.#		15	0		.
11	Field11	Number	#.#		15	0		.
12	Field12	String			0			

i) Jak widać ta pobieżna analiza nie była najlepsza, dlatego dokonajmy kilku poprawek. Po pierwsze, zmieńmy nazwy pól na: `country_code`, `postal_code`, `place_name`, `admin_name1`, `admin_code1`, `admin_name2`, `admin_code2`, `admin_name3`, `admin_code3`, `latitude`, `longitude`, `accuracy`. **Ponadto zmieńmy typ oraz długość poszczególnych kolumn zgodnie z poniższą tabelką**.


File | Content | Error Handling | Filters | Fields | Additional output fields

	Name	Type	Format	Position	Length	Precision
1	country_code	String			2	
2	postal_code	String			20	
3	place_name	String			180	
4	admin_name1	String			100	
5	admin_code1	String			20	
6	admin_name2	String			100	
7	admin_code2	String			20	
8	admin_name3	String			100	
9	admin_code3	String			20	
10	latitude	Number	#.#			
11	longitude	Number	#.#			
12	accuracy	Number	#			

j) Aby sprawdzić poprawność działania komponentu wybierzmy przycisk *Preview rows*, a następnie ograniczmy liczbę przeglądanych wierszy do pierwszych `5`.

Examine preview data

Rows of step: Odczyt US.txt (5 rows)

	country_code	postal_code	place_name	admin_name1	admin_code1	admin_name2	admin_code2
1	US	34050	FPO	<null>	AA	Erie	00000000000000000029
2	US	34034	APO	<null>	AA	Dillon	00000000000000000033
3	US	99553	Akutan	Alaska	AK	Aleutians East	00000000000000000013
4	US	99571	Cold Bay	Alaska	AK	Aleutians East	00000000000000000013
5	US	99583	False Pass	Alaska	AK	Aleutians East	00000000000000000013

k) Zamknijmy definicję własności tego komponentu.

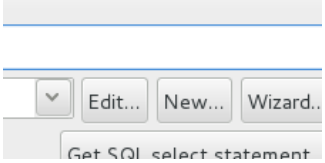
l) Ze względu na fakt, iż porównywanie starych i nowych postaci danych wykonamy w oparciu o komponent *Merge Rows (diff)*, który wymaga posortowanych wierszy z obu źródeł. Dołóżmy do naszej transformacji znany już komponent *Sort Rows* z katalogu *Transform*, a następnie połączmy go z poprzednim.

m) Wejdźmy do własności nowego komponentu, zmieńmy nazwę tego kroku na `Posortuj wg kodu i miasta`, a następnie zdefiniujmy porządek zgodnie z nazwą komponentu.

Fields :

	Fieldname	Ascending	Case sensitive compare?
1	postal_code	Y	N
2	place_name	Y	N

n) Drugim źródłem naszej transformacji będzie odczyt z tabeli zawierającej poprzedni obraz danych. A zatem będzie to element *Table input* odczytujący dane z tabeli `geonames_org`. W związku z tym, że w naszym repozytorium nie istnieje jeszcze odpowiednie połączenie wejdźmy do własności nowego komponentu i za pomocą przycisku New zdefiniujmy nowe połączenie.



o) Podczas definiowania nowego połączenia określmy jego:

- nazwę: `ods`
- typ: `MySQL`
- nazwę hosta: `localhost`
- nazwę bazy danych: `ods`
- użytkownika: `ods`
- hasło: `ods`

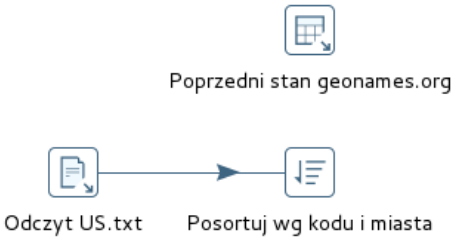
p) Po zakończonej edycji połączenia powinno stać się ono połączeniem naszego nowego komponentu.

Możemy przystąpić teraz do zdefiniowania treści polecenia SQL (wykorzystaj przycisk *Get SQL select statement…*). **Koniecznie pamiętajmy o klauzuli `ORDER BY`. Na liście kolumn pomińmy kolumny będące znacznikami zmian (`inserted` i `updated`).**

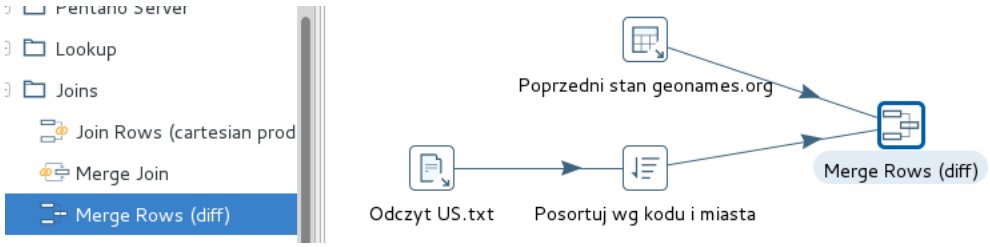
```
SQL

SELECT
  country_code
, postal_code
, place_name
, admin_name1
, admin_code1
, admin_name2
, admin_code2
, admin_name3
, admin_code3
, latitude
, longitude
, accuracy
FROM geonames_org
ORDER BY postal_code, place_name
```

q) Przed zamknięciem własności zmieńmy także nazwę kroku na `Poprzedni stan geonames.org`.



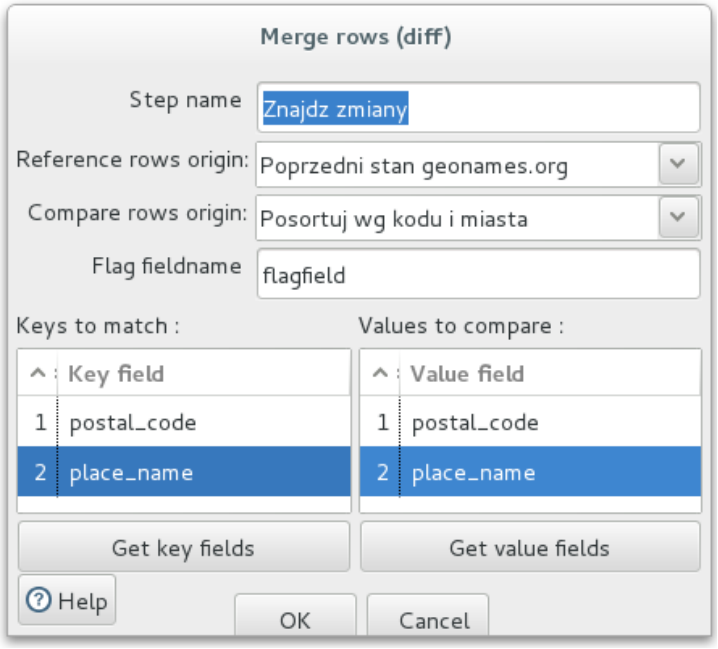
r) Komponentem transformacji, który idealnie nadaje się w przypadku techniki *Snapshot-Based CDC* jest *Merge Rows (diff)*. Dlatego to on właśnie, z katalogu *Joins*, będzie kolejną składową naszej transformacji. Dane wejściowe do tego kroku będą pochodziły z dwóch poprzednio utworzonych kroków.



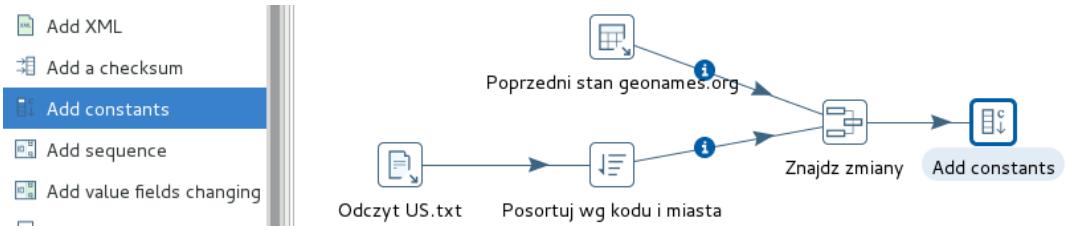
s) Otwórzmy właściwości komponentu *Merge Rows (diff)*.

- Zmieńmy jego nazwę na `Znajdz zmiany`.
- Jako własność *Reference rows origin:* wskażmy źródło, która dostarcza poprzedni obraz danych (`Poprzedni stan geonames.org`).
- Natomiast jako własność *Compare rows origin:* wskażmy źródło, która dostarcza najnowszy obraz danych (`Posortuj wg kodu i miasta`).
- Pole, które będzie dostarczało efekt porównania (informację czy doszło do zmiany czy też nie oraz jakiego typu była to zmiana) niech będzie posiadało zaproponowaną nazwę `flagfield`.
- Jako pola klucze wskażmy: `postal_code` i `place_name` (uważaj aby nazwy pól nie zawierały spacji, np. na końcu)
- Pola, których wartości będziemy porównywać w celu wykrycia zmian będą miały identyczne nazwy: `postal_code` i `place_name` (uważaj aby nazwy pól nie zawierały spacji, np. na

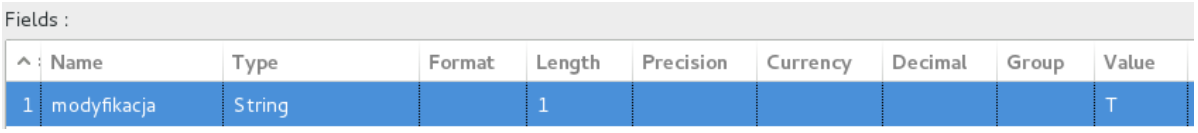
końcu).



t) W związku z tym, że do aktualizacji atrybutu (flagi) `updated` w tabeli `geonames_org` będziemy potrzebowali stałej '`T`', kolejny krok będzie ją tworzył. Z katalogu *Transform* dodajmy do naszej transformacji składnik *Add constants*, a następnie połączmy go z krokiem poprzednim.



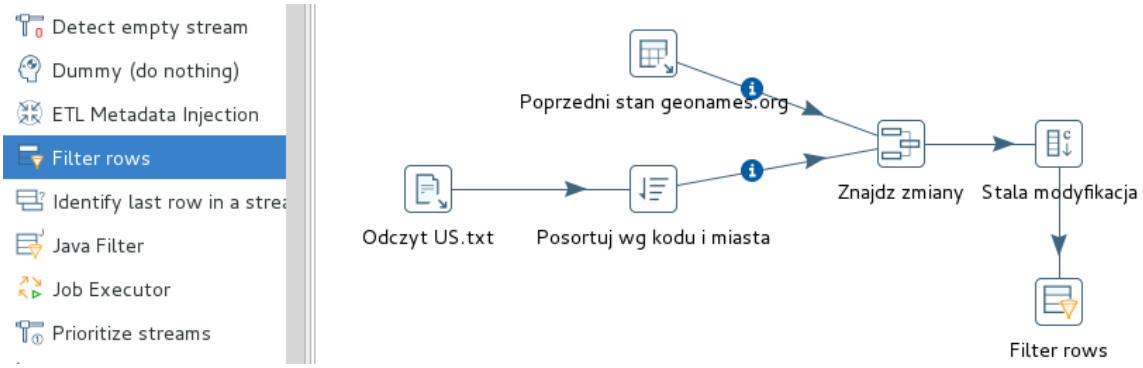
u) Otwórzmy własności tego kroku i zdefiniujmy stałą `modyfikacja` o oczekiwanym typie, długości i samej wartości `T`. Jako nazwę kroku wstawmy `Stala modyfikacja`.



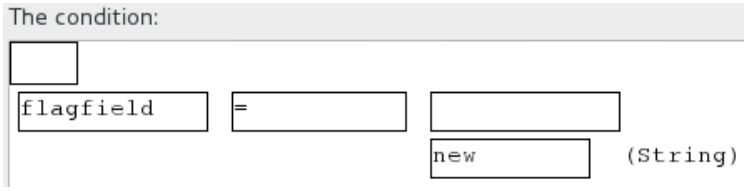
v) Krok *Merge Rows (diff)* tworzy strumień krotek obejmujący krotki, które nie uległy żadnej zmianie a także te które zostały wg niego usunięte, zmodyfikowane lub wstawione. Ze względu na to, że nie jesteśmy zainteresowani przetwarzaniem następujących typów krotek:

- które nie uległy zmianie – gdyż nie będą wpływały na zmianę naszych danych w hurtowni danych
- które zostały usunięte – gdyż nie chcemy niczego z naszej hurtowni usuwać – chcemy tylko dodawać nowe informacje oraz aktualizować te, które się zmieniły

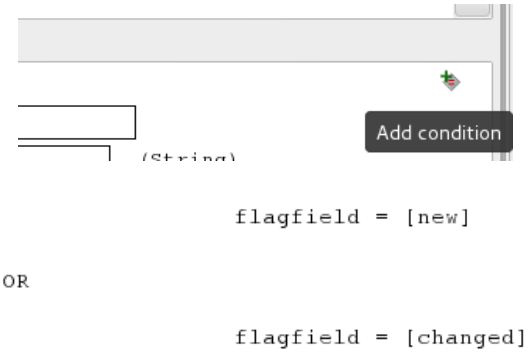
dlatego dokonamy filtrowania strumienia krotek, tak aby przetwarzać dalej tylko nowe krotki oraz te, które uległy zmianie. W tym celu umieścimy w naszej transformacji komponent *Filter rows* z katalogu *Flow*. Połączmy go z poprzednim krokiem.



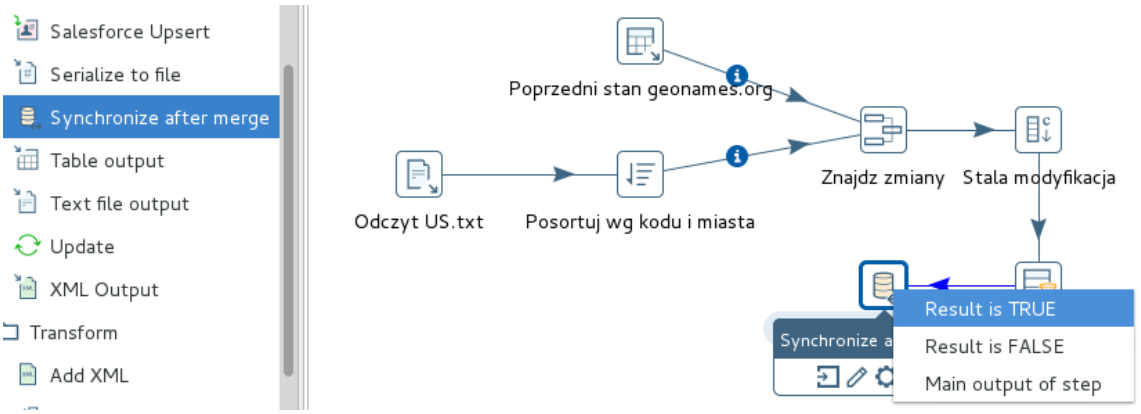
w) Otwórzmy własności tego kroku. W pierwszej kolejności zmieńmy nazwę kroku na `Nowe i zmienione`. Następnie zdefiniujmy nasz warunek. Pierwszym członem warunku będzie `flagfield = 'new'`



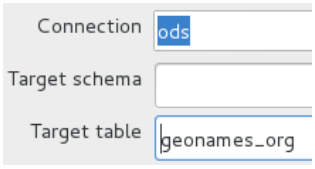
x) Drugi człon warunku będziemy mogli zdefiniować dzięki opcji *Add condition*. Będzie on miał postać `flagfield = 'changed'`. Z poprzednim członem warunku będzie połączony spójnikiem `OR`.



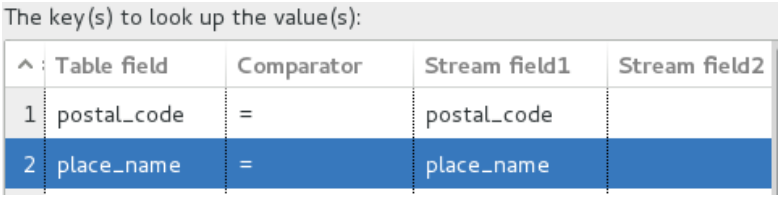
y) Czas na krok najważniejszy, integrujący dane nowe z poprzednią ich wersją. Idealnie do tego będzie się nadawał komponent *Synchronize after merge* z katalogu *Output*. Dodajmy go do naszej transformacji. Źródłem danych dla niego będzie krok `Nowe i zmienione`, przy czym dostarczone zostaną do niego krotki, które spełniają zdefiniowany przez nas uprzednio warunek.



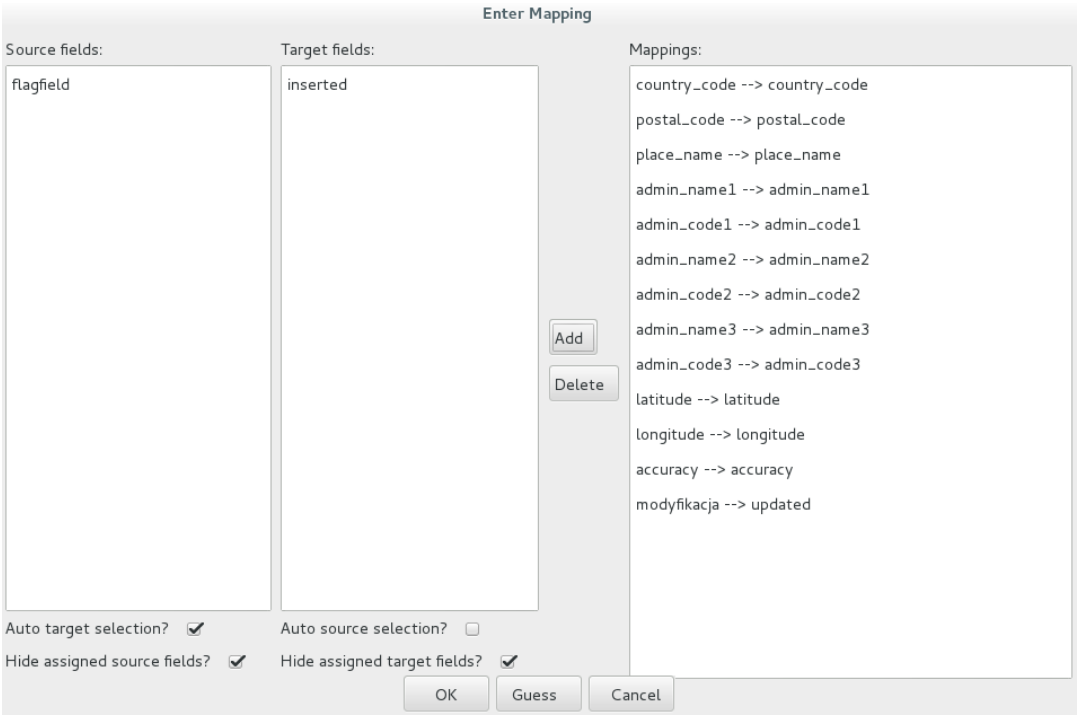
z) Przejdźmy do właściwości nowo dodanego komponentu. Jako własności *Connection* oraz *Target table* wprowadźmy odpowiednio `ods` oraz `geonames_org`.



aa) Jako wartości klucza służące do porównania istniejących danych z nowymi wskażmy: `postal_code` oraz `place_name`.



bb) Aby zdefiniować zakres wykonywanych modyfikacji musimy dokonać mapowania poszczególnych pól. W tym celu wykorzystaj przycisk *Edit mapping*, a następnie przycisk *Guess* aby zautomatyzować to działanie. Dodaj ręcznie mapowanie dla pary atrybutów i `updated`. Ostateczne mapowanie powinno wyglądać jak poniżej:

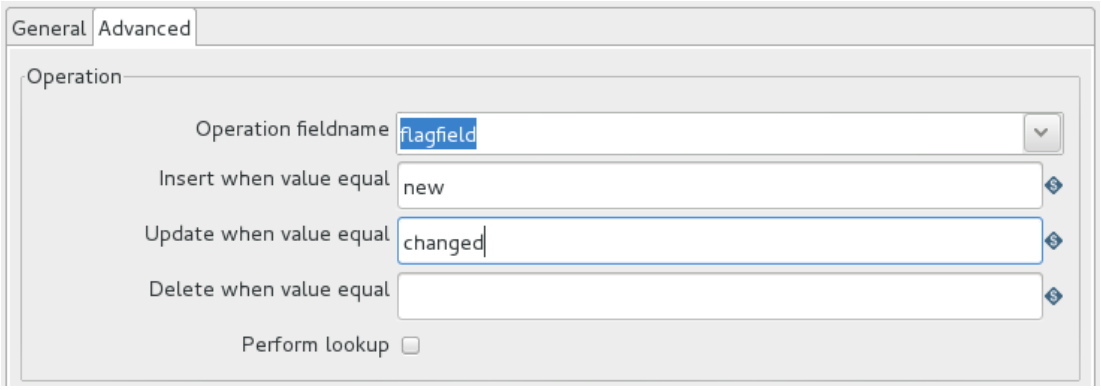


cc) Zamknijmy definicję mapowania i uzupełnijmy sposób wykonania modyfikacji w sekcji *Update fields*. Definicja modyfikacji powinna uwzględniać konieczność modyfikacji **WSZYSTKICH pól oprócz** `postal_code` i `place_name`, których wartości będą określane jedynie podczas wstawiania wierszy.

	Table field	Stream field	Update
3	place_name	place_name	N
4	admin_name1	admin_name1	Y
5	admin_code1	admin_code1	Y
6	admin_name2	admin_name2	Y
7	admin_code2	admin_code2	Y
8	admin_name3	admin_name3	Y
9	admin_code3	admin_code3	Y
10	latitude	latitude	Y
11	longitude	longitude	Y
12	accuracy	accuracy	Y
13	updated	modyfikacja	Y

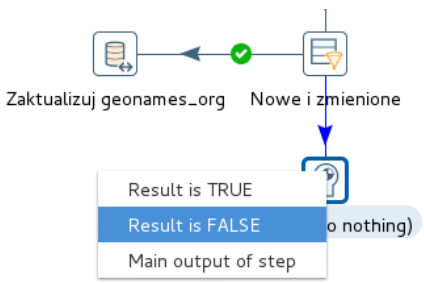
dd) Po określeniu powyższych własności przejdźmy na zakładkę *Advanced* aby określić kiedy jakie operacje będą podejmowane.

- Na początku wskażmy pole, które określa typ modyfikacji (`flagfield`).
- Następnie określmy wartość tego pola wskazującą konieczność wstawienia nowych danych (`new`).
- A także wartość wskazującą konieczność modyfikacji danych (`changed`).



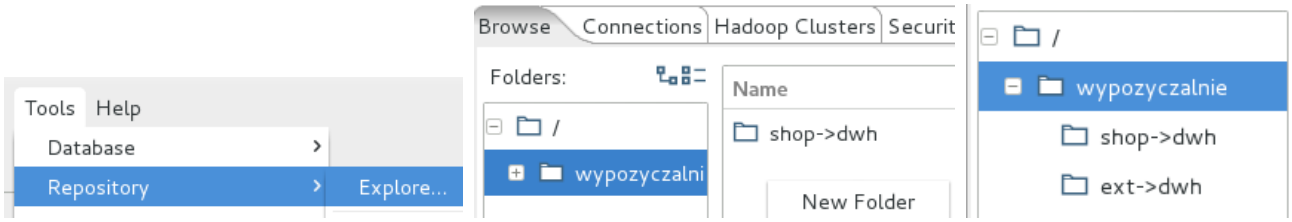
ee) Zakończmy edycję własności kroku *Synchronize after merge* zmieniając jego nazwę na `Zaktualizuj geonames_org`.

ff) Filtrowanie danych (komponent `Nowe i zmienione`) wymaga określenia zarówno docelowego miejsca dla krotek spełniających warunek filtra, jak i miejsca docelowego dla krotek, które warunku nie spełniają. W naszym przypadku nie jesteśmy zainteresowani jakimkolwiek przetwarzaniem krotek usuniętych lub niezmienionych, w związku z czym do naszej transformacji dołożymy dość specyficzny komponent *Dummy (do nothing)* z katalogu *Flow*. Źródłem danych dla tego komponentu będzie krok `Nowe i zmienione` w zakresie krotek, które nie spełniły warunku filtra.

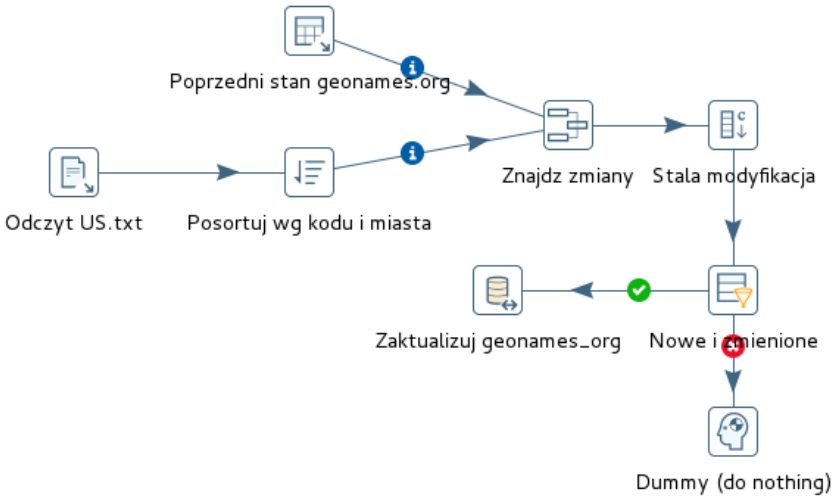


gg) Naszą transformację będziemy chcieli zapisać jako `POROWNANIE STANY KRAJE` w katalogu `/wypozyczalnie/ext->dwh`. Jednak w związku z tym, że katalog `ext->dwh` jeszcze nie istnieje

utwórzmy go najpierw korzystając z narzędzia *Repository explorer*.



hh) Teraz możemy naszą transformację zapisać jako `POROWNANIE STANY KRAJE` w katalogu `/wypozyczalnie/ext->dwh`, a następnie możemy spróbować ją uruchomić.



ii) Rezultat uruchomienia transformacji powinien wyglądać następująco:

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

	Stepname	Copynr	Read	Written	Input	Output	Updated
1	Odczyt US.txt	0	0	43630	43630	0	1
2	Poprzedni stan geonames.org	0	0	0	0	0	0
3	Posortuj wg kodu i miasta	0	43630	43630	0	0	0
4	Znajdz zmiany	0	43630	43630	0	0	0
5	Stala modyfikacja	0	43630	43630	0	0	0
6	Nowe i zmienione	0	43630	43630	0	0	0
7	Zaktualizuj geonames_org	0	43630	43630	0	43630	0
8	Dummy (do nothing)	0	0	0	0	0	0

jj) Jeśli ponownie uruchomimy naszą transformację wynik powinien ulec zmianie (żadne wiersze nie powinny być propagowane do kroku `Zaktualizuj geonames_org`).

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

	Stepname	Copynr	Read	Written	Input	Output	Updated
1	Odczyt US.txt	0	0	43630	43630	0	1
2	Poprzedni stan geonames.org	0	0	43630	43630	0	0
3	Posortuj wg kodu i miasta	0	43630	43630	0	0	0
4	Znajdz zmiany	0	87260	43630	0	0	0
5	Stala modyfikacja	0	43630	43630	0	0	0
6	Nowe i zmienione	0	43630	43630	0	0	0
7	Zaktualizuj geonames_org	0	0	0	0	0	0
8	Dummy (do nothing)	0	43630	43630	0	0	0

6. Po uruchomieniu stworzonej przed chwilą transformacji w tabeli `geonames_org` będziemy posiadali wiersze z flagami informującymi o dokonanych zmianach. Wykorzystamy teraz te dane w dwóch transformacjach modyfikujących wymiar klienci w hurtowni danych.

   a) Utwórzmy nową transformację. Pierwsza z transformacji ma obejmować modyfikację uwzględniającą dane zmienione w `geonames_org`. Dlatego pierwszym krokiem tej transformacji będzie komponent *Table input* i zawarte w nim zapytanie (wykorzystaj przycisk *Get SQL select statement* do wstępnego wygenerowania zapytania):

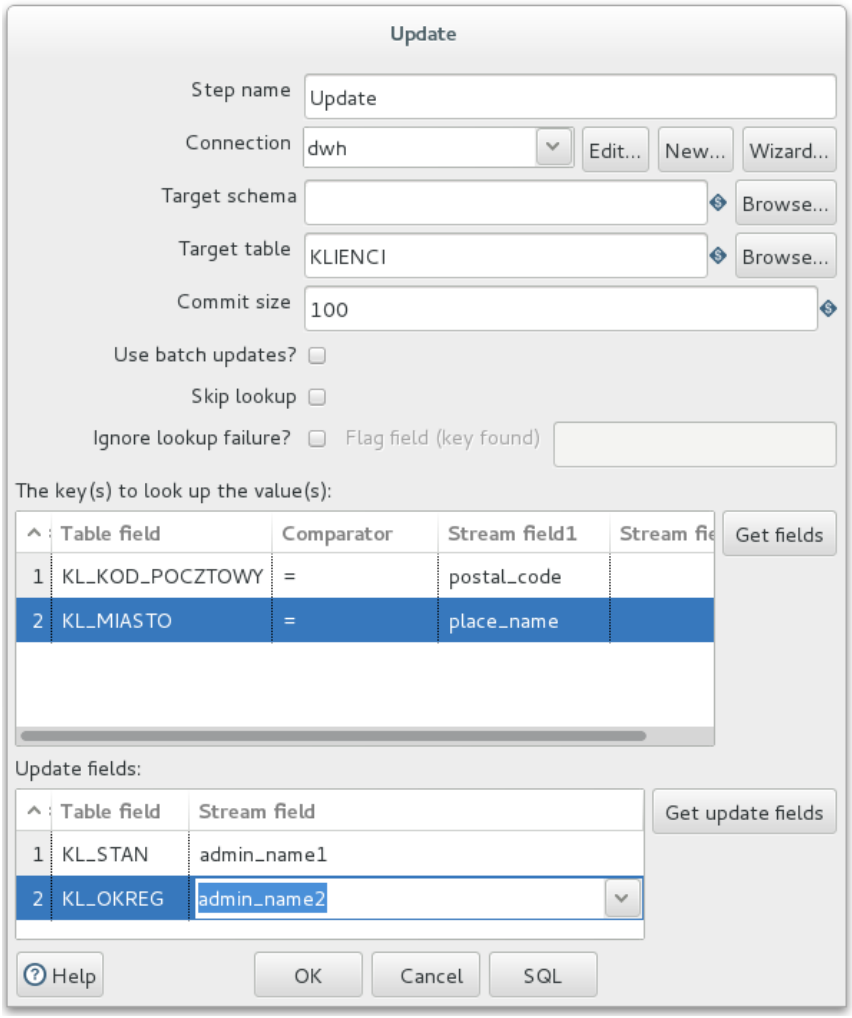
Connection: ods | Edit... | New... | Wizard...

SQL | Get SQL select statement...

```
SELECT
  country_code
, postal_code
, place_name
, admin_name1
, admin_code1
, admin_name2
, admin_code2
FROM geonames_org
WHERE inserted = 'N' AND updated = 'T'
```

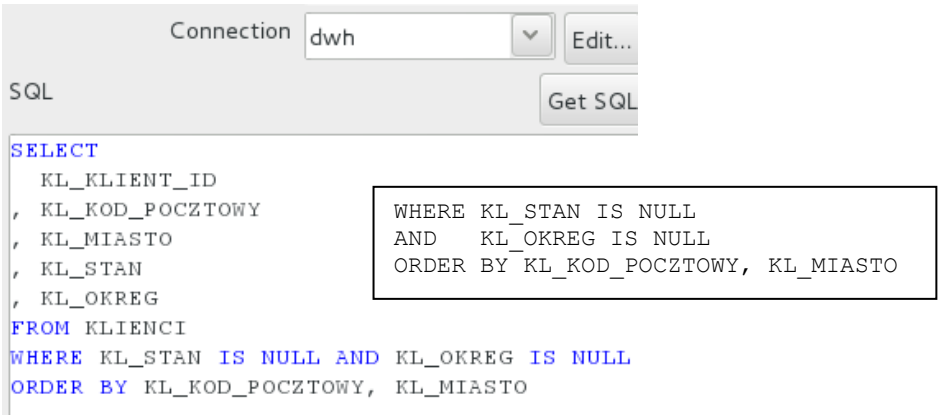
   b) Drugim krokiem tej transformacji będzie komponent *Update* z katalogu *Output*, który dokona odpowiednich modyfikacji w wymiarze klienci (tabeli `KLIENCI`). Umieśćmy stosowny komponent w ramach transformacji i połączmy go z poprzednim.

   c) Przejdźmy do własności komponentu *Update* i zdefiniujmy go tak aby aktualizował zmienione stany i okręgi zgodnie z przypisanym im miastem i numerem kodu pocztowego.



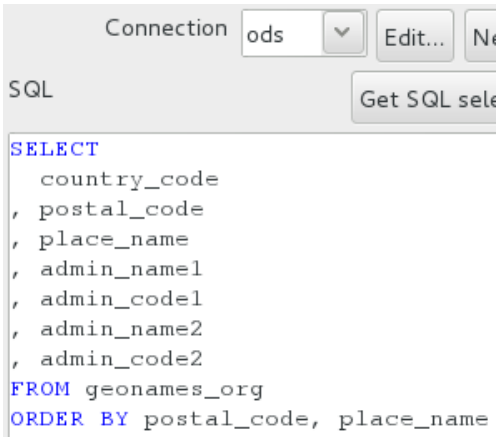
d) Zapiszmy transformację jako `POPRAW ZMIENIONE STANY KRAJE` w katalogu `/wypozyczalnie/ext->dwh`. Nie ma sensu uruchamiać naszej transformacji gdyż wszystkie wiersze w tabeli `geonames_org` są nowe.

e) Utwórzmy kolejną nową transformację. Zadaniem drugiej transformacji będzie określenie wartości atrybutów stan i okręg dla wszystkich krotek wymiaru klienci, które wartości w tych atrybutach nie posiadają. Dlatego pierwszym krokiem tej transformacji będzie zapytanie (komponent *Table input*) wydobywające wszystkie wiersze z wymiaru klienci, które tych danych nie posiadają. Dołóżmy do zapytania odpowiedni warunek a także klauzulę `ORDER BY`, która posortuje klientów wg kodu pocztowego i nazwy miasta. Dla wygody postać warunku i klauzuli `ORDER BY` znajduje się poniżej. Jako nazwę tego kroku wpiszmy `Klienci bez stanu i okregu`.

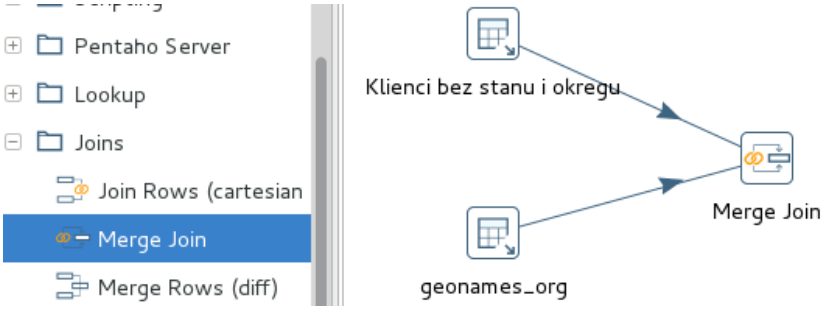


f) Drugim krokiem transformacji będzie odczyt (komponent *Table input*) wszystkich danych z tabeli

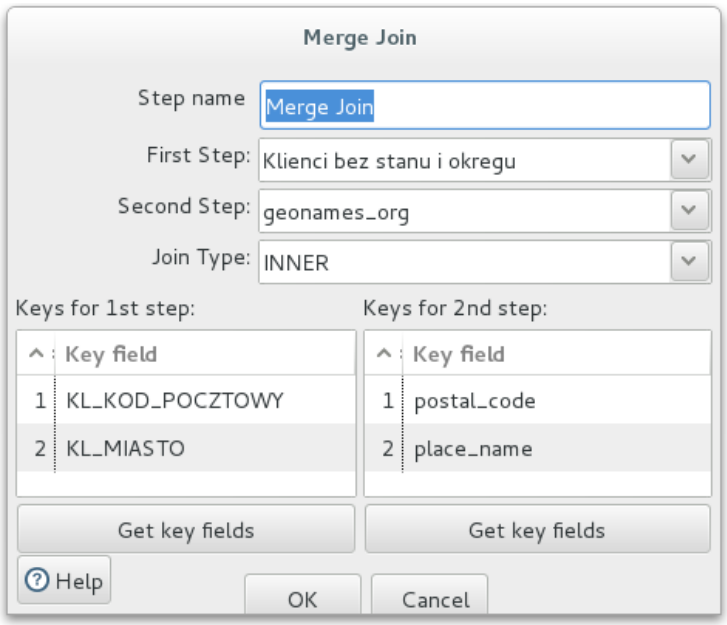
`geonames_org` – każda krotka może być przydatna np. do aktualizacji nowych krotek w tabeli `KLIENCI`. W tym przypadku również pamiętajmy o posortowaniu wierszy. Nazwą tego kroku niech będzie `geonames_org`.



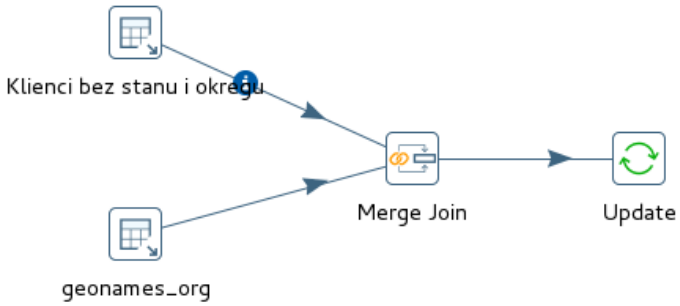
g) Dzięki temu, że oba źródła danych zostały posortowane w ten sam sposób, będziemy mogli je ze sobą połączyć korzystając z komponentu *Merge Join* z katalogu *Join*. Dołóżmy go do transformacji, a następnie połączmy go z każdym źródłem danych.



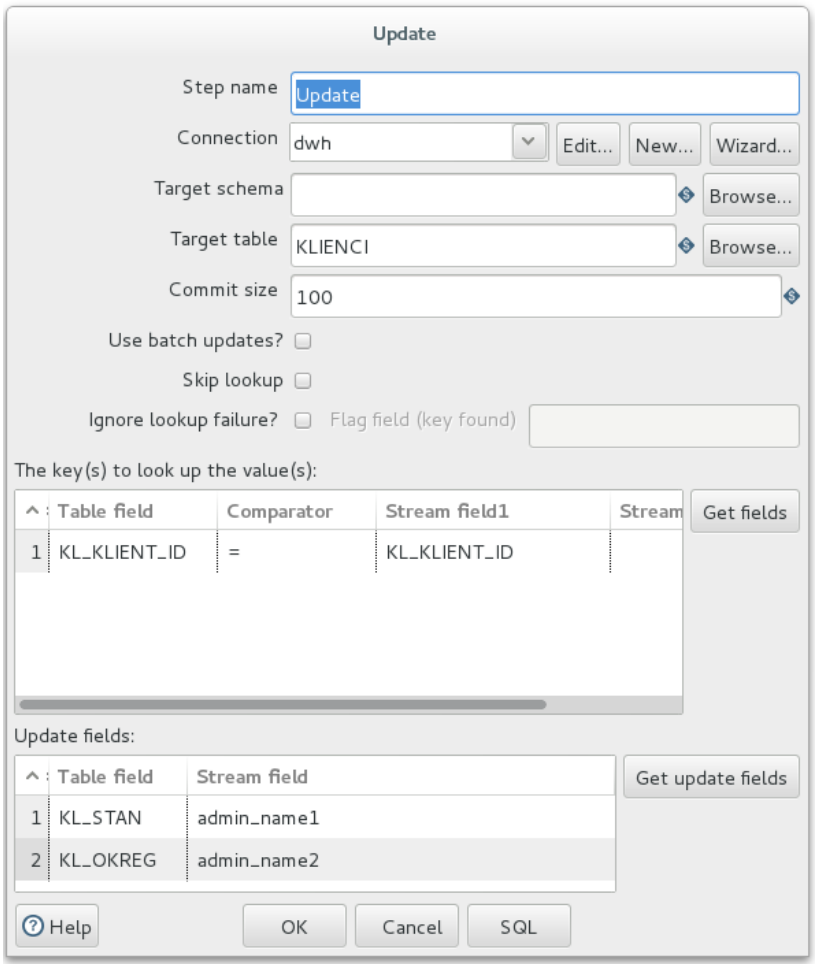
h) Określmy definicja połączenia. Nazwę kroku możemy pozostawić oryginalną.



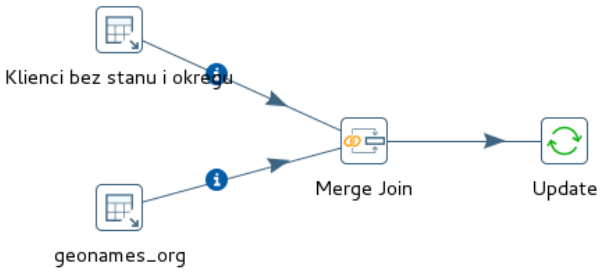
i) Wynikiem połączenia będą krotki, które należy uwzględnić podczas modyfikacji wierszy w tabeli klienci. Dlatego w tej transformacji również użyjemy komponentu *Update*.



j) Modyfikacja będzie aktualizowała kolumny `KL_STAN` i `KL_OKREG` w oparciu o pola `admin_name1` i `admin_name2` na podstawie identyfikatora klienta.



k) Zapiszmy w katalogu `/wypozyczalnie/ext->dwh` naszą transformację jako `UZUPELNIJ STANY KRAJE`. Jej ostateczna postać powinna być następująca:



l) Uruchommy transformację. Wynik powinien być analogiczny do tego poniżej:

Execution Results

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

	Stepname	Copynr	Read	Written	Input	Output	Updated
1	Klienci bez stanu i okregu	0	0	599	599	0	0
2	geonames_org	0	0	43630	43630	0	0
3	Merge Join	0	44229	588	0	0	0
4	Update	0	588	588	588	0	588

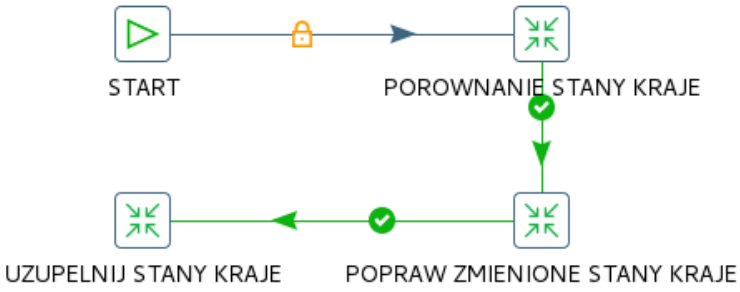
m) Zwróćmy uwagę, że zmodyfikowano mniej wierszy niż odczytano z tabeli `KLIENCI`. Fakt ten może świadczyć albo o niedokładnych danych referencyjnych (pochodzących z `geonames.org`) , co możemy raczej wykluczyć, albo o błędach w danych źródłowych (`shop1` i `shop2`). Zajmiemy się tym później.

7. Pozostało nam już tylko złożyć wszystkie transformacje w jedną całość.

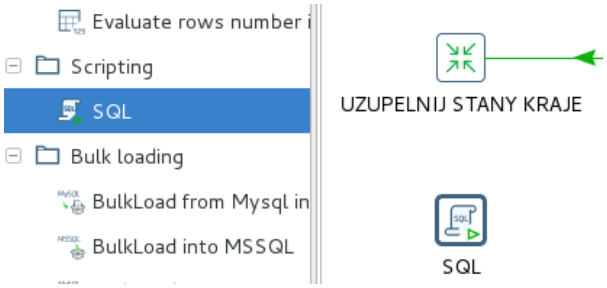
a) Utwórzmy nowe zadanie a następnie umieść na nim kolejno:

- komponent *START*
- transformacje: `PorowanieStanyKraje`, `PoprawZmienioneStanyKraje`, `UzupelnijStanyKraje`

b) Połączmy wstawione komponenty w analogicznej do poniższej kolejności



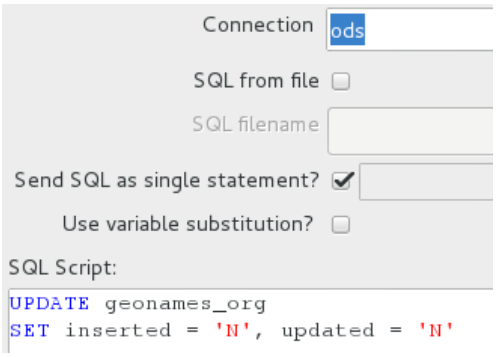
c) W związku z tym, że znaczniki określające typ zmian nie zostały skasowane w żadnej transformacji (a to spowodowałoby, że transformacje każdorazowo traktowałyby je jak nowe dane), zrobimy to teraz za pomocą komponentu *SQL* z katalogu *Scripting*.



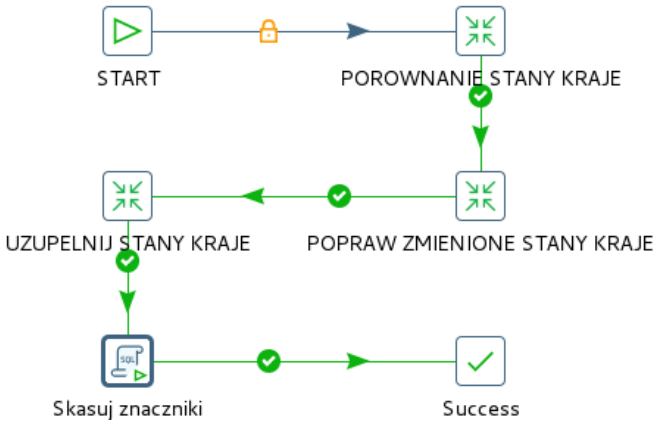
d) Skrypt będzie uruchamiany w ramach połączenia `ods`, a jego treść skryptu będzie następująca

```
UPDATE geonames_org
SET    inserted = 'N',
       updated = 'N'
```

i zostanie ona wysłana jako **pojedyncze** polecenie SQL

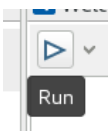


e) Nazwijmy ten krok jako `Skasuj znaczniki`. Zamknijmy jego własności i połączmy dodany komponent *SQL* tak, aby był wykonywany po wszystkich transformacjach. Na końcu łańcucha operacji umieśćmy komponent *Success*.



f) Zapiszmy w katalogu `/wypozyczalnie/ext->dwh` nasze zadanie jako `LADOWANIE STANOW OKREGOW`.

g) Na zakończenie uruchommy `LADOWANIE STANOW OKREGOW` w celu jego przetestowania.



Execution Results

History | Logging | Job metrics | Metrics

Job / Job Entry	Comment	Result
LADOWANIE STANOW OKREGOW		
Job: LADOWANIE STANOW OKREGOW	Start of job execution	
START	Start of job execution	
START	Job execution finished	Success
POROWNANIE STANY KRAJE	Start of job execution	
POROWNANIE STANY KRAJE	Job execution finished	Success
POPRAW ZMIENIONE STANY KRAJE	Start of job execution	
POPRAW ZMIENIONE STANY KRAJE	Job execution finished	Success
UZUPELNIJ STANY KRAJE	Start of job execution	
UZUPELNIJ STANY KRAJE	Job execution finished	Success
Skasuj znaczniki	Start of job execution	
Skasuj znaczniki	Job execution finished	Success
Success	Start of job execution	
Success	Job execution finished	Success
Job: LADOWANIE STANOW OKREGOW	Job execution finished	Success