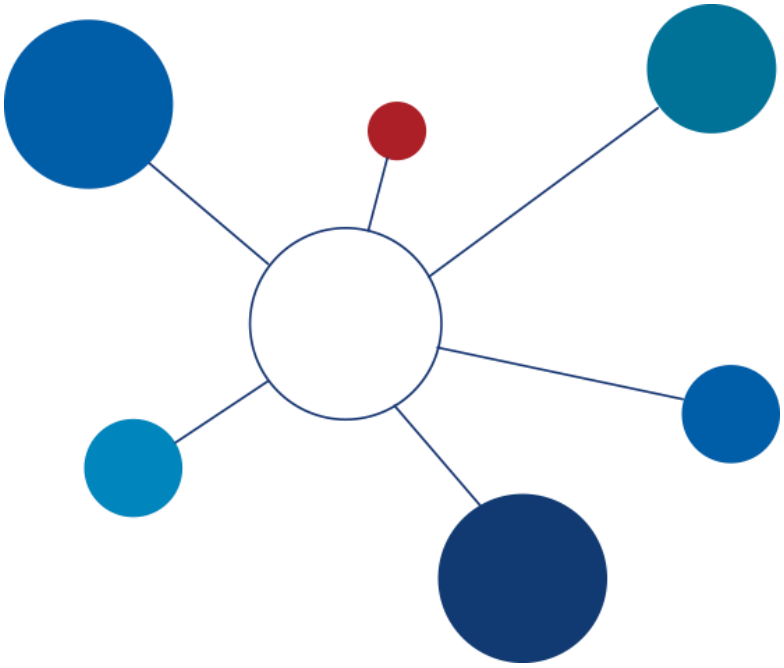


Technologie Zasilania i Odświeżania Hurtowni Danych

laboratorium

część VI

v20170324

I. DOKUMENT XML JAKO ŹRÓDŁO DANYCH.

W ramach poprzednich ćwiczeń wykorzystaliśmy plik *CSV* w celu uzupełnienia wymiaru klienci o dodatkowe atrybuty. Tym razem wykorzystamy dokument *XML* w celu uzupełnienia wymiaru filmy.

W chwili obecnej nasz wymiar dotyczący filmów jest stosunkowo ubogi – zawiera jedynie tytuł filmu oraz rok wydania. Z filmami wiążą się jednakże inne informacje, które mogą być bardzo ciekawe z punktu widzenia analiz opracowujących różnorodne akcje promocyjne. Przykładami takich informacji mogą być: gatunek filmu, kategoria wiekowa, dodatki dołączone do płyty, długość trwania filmu itp.

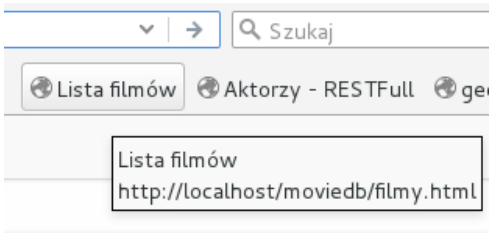
W wielu przypadkach podobnego typu informacje są do uzyskania z zewnętrznych źródeł danych. My skorzystamy ze strony internetowej serwisu o filmach.

1. Na początku przyglądniemy się naszemu nowemu źródłu danych.

   a) Uruchommy przeglądarkę stron WWW i wprowadźmy do niej adres

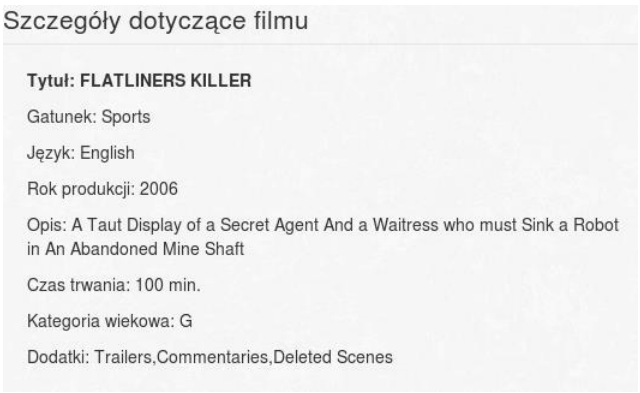
   `http://localhost/moviedb/filmy.html`

   Alternatywnie możemy skorzystać z linku dostępnego na pasku zakładek w naszej przeglądarce



   b) W polu *Fragment tytułu* wprowadźmy przykładowo ciąg znaków `KIL`, a następnie za pomocą przycisku *Wyszukaj* znajdźmy filmy posiadające taki ciąg znaków w tytule.

   c) Kliknijmy w jeden z wyświetlonych tytułów aby przejść do szczegółów dotyczących określonego filmu.



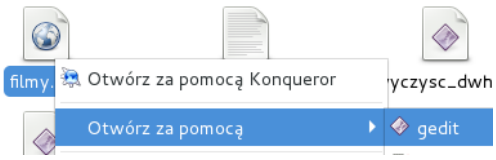
   d) Bezpośrednie przetwarzanie stron WWW byłoby stosunkowo trudne, na szczęście ta strona webowa oparta jest o jedno źródło danych mające postać pliku *XML*. Podglądnijmy jego zawartość

wprowadzając w przeglądarce adres: `http://localhost/moviedb/filmy.xml`

```
−<FILMY>
 −<FILM id="1">
    <TYTUL>ACADEMY DINOSAUR</TYTUL>
   −<OPIS>
      A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher
      in The Canadian Rockies
    </OPIS>
    <DATA_PRODUKCJI>2006</DATA_PRODUKCJI>
    <JEZYK>English</JEZYK>
    <DLUGOSC>86</DLUGOSC>
    <KATEGORIA_WIEKOWA>PG</KATEGORIA_WIEKOWA>
    <DODATKI>Deleted Scenes,Behind the Scenes</DODATKI>
    <GATUNEK>Documentary</GATUNEK>
  </FILM>
 −<FILM id="2">
    <TYTUL>ADAPTATION HOLES</TYTUL>
   −<OPIS>
```

e) Elementem głównym dokumentu jest element `FILMY`. Zawiera on sekwencję elementów `FILM`, który z kolei posiada atrybut `id`, oraz sekwencję elementów o następujących nazwach: `TYTUL`, `OPIS`, `DATA_PRODUKCJI`, `JEZYK`, `DLUGOSC`, `KATEGORIA_WIEKOWA`, `DODATKI`, `GATUNEK`. Większość z tych elementów posiada prostą zawartość tekstową, jedynie element `DODATKI` jest elementem „wielowartościowym”. Dodatki w filmach mogą obejmować następujące przypadki: trailery, dodatkowe sceny, komentarze oraz sceny usunięte. W związku z tym, że wygląd serwisu może się zmieniać bardzo często, a jego źródło danych niekoniecznie, tym bardziej oprzemy naszą transformację właśnie na nim.

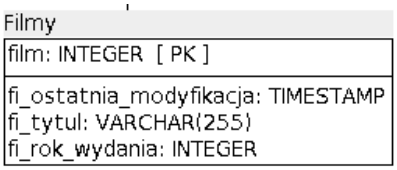
f) Nie możemy jednak ufać bezgranicznie stałości formatu źródła, dlatego opracujemy dokument, który pozwoli nam na weryfikację jego struktury. W naszym przypadku będzie to dokument *XSD* (*XML Schema Definition*). Aby przyspieszyć realizację naszej transformacji został on już wcześniej utworzony. Korzystając z przeglądarki plików wejdź do katalogu `/home/etl/labs` i otwórz za pomocą edytora plików testowych plik `filmy.xsd`




```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="FILMY">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="FILM"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="FILM">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="TYTUL"/>
        <xs:element ref="OPIS"/>
        <xs:element ref="DATA_PRODUKCJI"/>
        <xs:element ref="JEZYK"/>
        <xs:element ref="DLUGOSC"/>
```

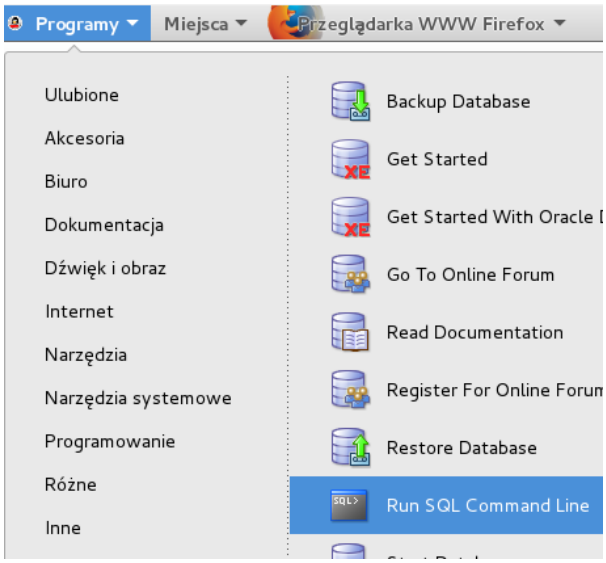
2. Opracowanie i implementacja modyfikacji schematu hurtowni danych.

a) Tak jak wspomnieliśmy, nasz wymiar filmy jest wyjątkowo ubogi, dlatego dokonamy teraz jego rozbudowy. Nowy zestaw danych pochodzący z serwisu WWW będzie do tego celu bardzo przydatny. Przypomnijmy sobie jak wygląda postać wymiaru filmy w naszej hurtowni danych. W chwili obecnej reprezentująca go tabela ma następującą postać:



b) Nowe atrybuty tego wymiaru, które chcemy utworzyć to: opis, język, długosc, kategoria_wiekowa, czy_trailery, czy_komentarze, czy_usuniete_sceny, czy_dodatkowe_sceny, gatunek. Dokonajmy zatem odpowiednich modyfikacji. Najprostszy sposób to wykorzystanie oprogramowania *SQL*Plus*. Z

menu *Programy/Inne* wybierz *Run SQL Command Line*.



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

d) Sprawdź strukturę tabeli klienci korzystając z polecenia `desc filmy`.

```
SQL> desc filmy
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 FI_FILM_ID                                NOT NULL NUMBER
 FI_ROK_WYDANIA                            NOT NULL NUMBER
 FI_TYTUL                                  NOT NULL VARCHAR2(255)
 FI_OSTATNIA_MODYFIKACJA                   NOT NULL TIMESTAMP(6)
```

e) Wprowadźmy zmiany za pomocą następujących poleceń:

```
alter table filmy
add ( FI_OPIS                 VARCHAR2(150),
      FI_JEZYK                VARCHAR2(10),
      FI_DLUGOSC              NUMBER,
      FI_KATEGORIA_WIEKOWA    VARCHAR2(5),
      FI_CZY_TRAILERY         CHAR(1),
      FI_CZY_KOMENTARZE       CHAR(1),
      FI_CZY_USUNIETE_SCENY   CHAR(1),
      FI_CZY_DODATKOWE_SCENY  CHAR(1),
      FI_GATUNEK              VARCHAR2(25) );
```

f) Z oczywistych powodów zmiany w schemacie hurtowni danych powinny

- być udokumentowane,
- uwzględniać istniejące transformacje,
- być przetestowane.

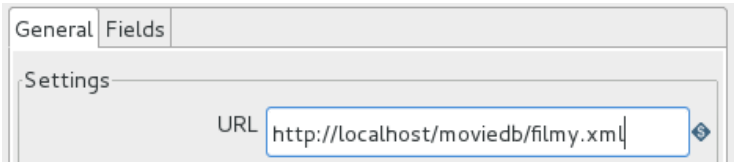
g) Zmiany, które wprowadziliśmy nie powinny wpływać negatywnie na istniejące już transformacje.

3. Czas na naszą transformację.

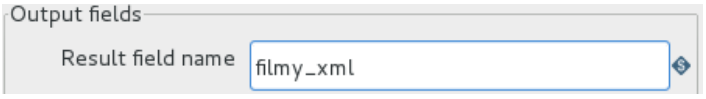
a) Utwórzmy nową transformację. Pierwszym krokiem tej transformacji będzie *HTTP client* z katalogu *Lookup*.

b) Otwórzmy własności tego komponentu. Na pierwszej zakładce *General* do własności *URL* wstawmy adres pliku będącego źródłem danych dla naszego serwisu WWW:

```
http://localhost/moviedb/filmy.xml
```

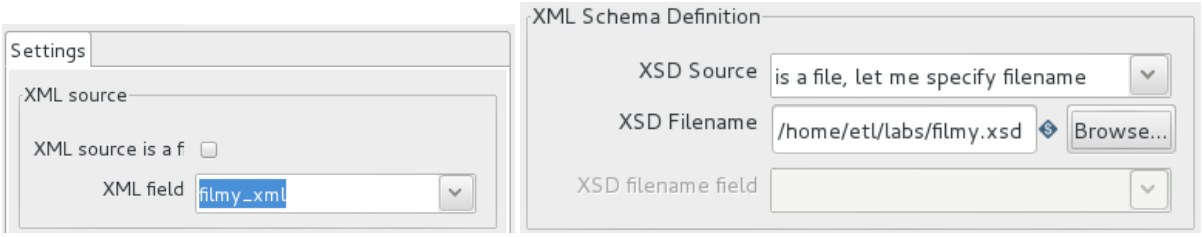


c) Jako nazwę pola wynikowego wprowadźmy `filmy_xml`.

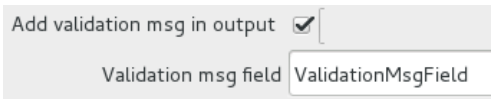


d) Zamknijmy własności tego kroku i wprowadźmy kolejny do naszej transformacji – *XSD Validator* z katalogu *Validation*, który będzie odpowiadał za weryfikację naszego dokumentu źródłowego.

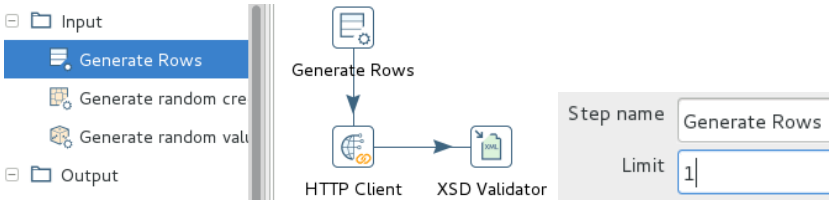
e) Połączmy pierwszy krok z drugim, a następnie otwórzmy własności kroku *XSD Validator*. Wskażmy pole, które będzie zawierało źródłowy dokument *XML* podlegający weryfikacji, oraz dokument *XSD* który będzie podczas weryfikacji wykorzystywany.



f) Ponadto zaznaczmy, że wyniki weryfikacji mają być umieszczone w dodatkowym polu wynikowym.



g) W związku z tym, że transformacje przetwarzają strumień danych, a *HTTP Client* nie jest przeznaczony do ich utworzenia, dodajmy z katalogu *Input* komponent *Generate Rows* i umieścimy go na samym początku transformacji. Zmień liczbę krotek generowanych przez ten komponent na 1. Połącz nowy komponent z *Http Client*.



h) Teraz możemy podglądnąć czy nasze źródło jest dostępne oraz czy jego struktura jest zgodna z

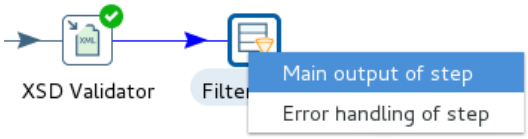
oczekiwaną. W tym celu wywołaj menu kontekstowe na elemencie *XSD Validator* i wybierz opcję *Preview*. Przyciskiem *Quick Launch* uruchom transformację. Wynik powinien być jak poniżej:

Rows of step: XSD Validator (1 rows)

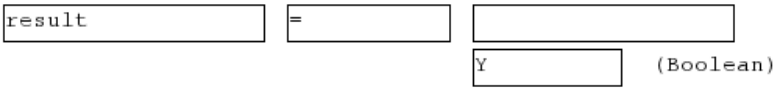
	filmy_xml	result	ValidationMsgField
1	<?xml version="1.0" encoding="UTF-8"?> <FILMY><FILM id="1"><TYTUL>ACADEMY DI	Y	<null>

i) Tym razem postać dokumentu wejściowego była poprawna, ale nie zawsze tak może być, dlatego umieścimy w naszej transformacji krok, który zapobiegnie przetwarzaniu błędnego dokumentu *XML*.

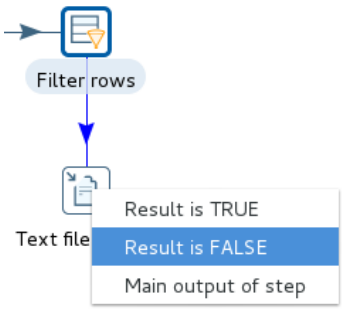
j) Umieśćmy w naszej transformacji *Filter rows* z katalogu *Flow*. A następnie podłączmy go jako wyjście dla kroku *XSD Validator* wskazując, że ma on zostać wykorzystany jako krok, do którego będzie kierowany główny wynik kroku poprzedniego.



k) Otwórzmy własności tego kroku. A następnie zdefiniujmy warunek weryfikujący poprawność walidacji.



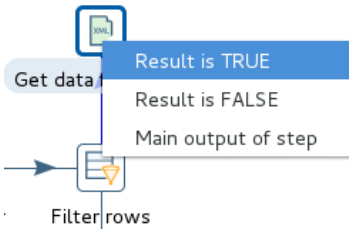
l) Utworzymy teraz krok rejestrujący pojawiające się ewentualne błędy. Dodajmy do transformacji krok *Text file output* z katalogu *Output*. Połączmy go na wyjście poprzedniego kroku (filtra krotek) i określmy, że ma być wywoływany wtedy, gdy sprawdzany w poprzednim kroku warunek nie będzie spełniony.



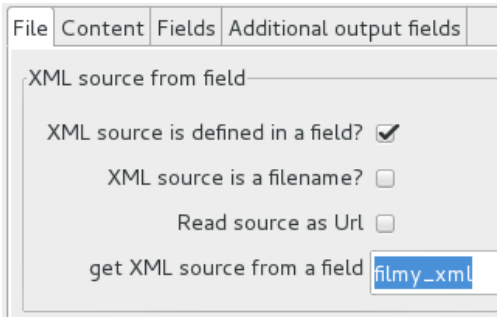
m) Otwórzmy własności `Text file ouput`. W nazwie pliku podajmy `/home/etl/results/filmy_xml_vs_xsd`. Jako rozszerzenie pozostawmy `txt`. Na zakładce *Fields* wskażmy, że w wynikowym pliku ma być umieszczana zawartość pola `ValidationMsgField`.

n) Jeśli walidacja zostanie wykonana w sposób poprawny, możemy rozpocząć przetwarzanie naszego źródła danych. Dołóżmy zatem komponent *Get data from XML* z katalogu *Input* i podłączmy go jako wyjście kroku *Filter rows*, używane w przypadku, kiery warunek będzie spełniony (walidacja będzie

pomyślna).



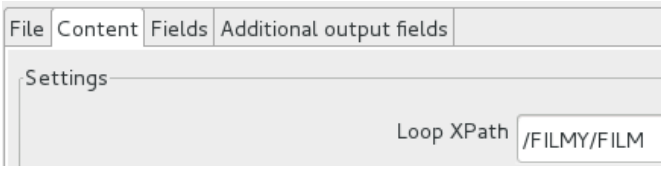
o) Otwórzmy własności tego kroku i wskażmy, że źródłem danych *XML* będzie pole `filmy_xml`



p) Na zakładce *Content* wskażemy poziom iteracji. Aby odbyło się to sprawnie wykorzystajmy przycisk *Get XPath nodes*. W związku z tym, że zawartość *XML* pochodzi z pola, którego postać nie jest znana, aplikacja prosi o podanie fragmentu dokumentu XML (jako wzorca). Podajmy poniższy fragment (pochodzący z pierwszej „krotki" dokumentu *XML*):

```
<?xml version="1.0" encoding="UTF-8"?>
<FILMY>
  <FILM id="1">
    <TYTUL>ACADEMY DINOSAUR</TYTUL>
    <OPIS>A Epic Drama of a Feminist And a Mad Scientist who must Battle a
Teacher in The Canadian Rockies</OPIS>
    <DATA_PRODUKCJI>2006</DATA_PRODUKCJI>
    <JEZYK>English</JEZYK>
    <DLUGOSC>86</DLUGOSC>
    <KATEGORIA_WIEKOWA>PG</KATEGORIA_WIEKOWA>
    <DODATKI>Deleted Scenes,Behind the Scenes</DODATKI>
    <GATUNEK>Documentary</GATUNEK>
  </FILM>
</FILMY>
```

q) Pojawi się okno z listą dostępnych ścieżek. Wybierzmy ścieżkę `/FILMY/FILM`, tak aby każdy element `FILM` występujący we wnętrzu elementu głównego `FILMY` generował kolejną krotkę w strumieniu przetwarzania.

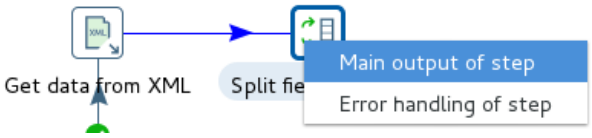


r) Przejdźmy na zakładkę *Fields*, a następnie za pomocą przycisku *Get fields* oraz podanego wcześniej fragmentu *XML* określmy te pola, które będziemy przetwarzali w naszym strumieniu.

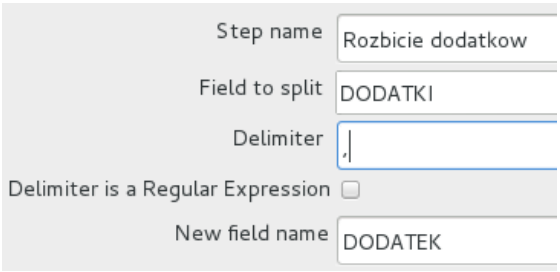
File | Content | Fields | Additional output fields

	Name	XPath	Element	Result type	Type
1	TYTUL	TYTUL	Node	Value of	String
2	OPIS	OPIS	Node	Value of	String
3	DATA_PRODUKCJI	DATA_PRODUKCJI	Node	Value of	Integer
4	JEZYK	JEZYK	Node	Value of	String
5	DLUGOSC	DLUGOSC	Node	Value of	Integer
6	KATEGORIA_WIEKOWA	KATEGORIA_WIEKOWA	Node	Value of	String
7	DODATKI	DODATKI	Node	Value of	String
8	GATUNEK	GATUNEK	Node	Value of	String

s) Zamknijmy edycję tego komponentu. Na obecnym etapie tworzenia transformacji, nasz dokument *XML*, generuje strumień, który prawie w stu procentach odpowiada oczekiwanej postaci wynikowej (atrybutom wymiaru filmy w hurtowni danych). Jedynym problemem jest wielowartościowe pole `DODATKI`. Do jego rozbicia na cztery oddzielne pola wykorzystamy trzy kolejne kroki transformacji. Pierwszym z nich będzie komponent *Split field to rows* z katalogu *Transform*. Umieśćmy go jako docelowy krok głównego wyniku poprzednio dodanego komponentu *Get data from XML*.



t) Wejdźmy do własności nowego kroku transformacji. Zmieńmy jego nazwę na `Rozbicie dodatkow`. Jako pole podlegające rozbiciu wskażmy `DODATKI`, jako ciąg znaków rozdzielający poszczególne wartości wstawmy przecinek, a jako pole, w którym w oddzielnych krotkach zostaną umieszczone indywidualne wartości dodatków podajmy `DODATEK`.



u) W wynikowych polach transformacji odpowiadających atrybutom wymiaru klienci `FI_CZY_TRAILERY`, `FI_CZY_KOMENTARZE`, `FI_CZY_USUNIETE_SCENY` i `FI_CZY_DODATKOWE_SCENY` chcemy wstawiać stałe ciągi znaków `T` lub `N` w zależności od tego, który z dodatków występuje w danym filmie. Dlatego kolejnym krokiem będzie *Add constants*, również z katalogu *Transform*.

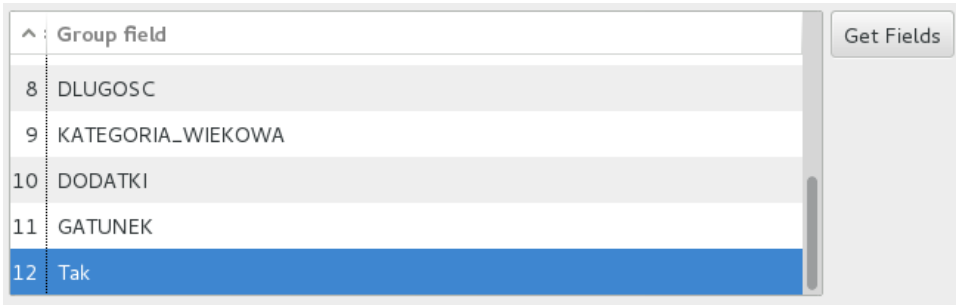
v) Jako pole tworzone przez ten krok zdefiniujmy pole `Tak`. Pokazano to poniżej:

Fields :

	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set empty string?
1	Tak	String		1					T	

w) Każde pole DODATKI zawierające kilka wartości, będzie tworzyło tyle krotek w przetwarzanym strumieniu ile wartości było (każda krotka będzie posiadała różną wartość w polu DODATEK). Kolejny krok będzie łączył ponownie krotki dotyczące tego samego filmu, przy czym z każdej z krotek zawierającej w polu DODATEK określony typ tego dodatku, będzie wynikała wartość w oddzielnym, nowym polu naszego wynikowego strumienia. Dodajmy do transformacji komponent *Row denormaliser* również z katalogu *Transform*.

x) Przejdźmy do własności tego kroku. Jako pole klucza wskażmy DODATEK. To jego zawartość będziemy testowali umieszczając w określonych polach wynikowych naszej transformacji przed chwilą utworzone stałe. Jako pola grupowania podajmy wszystkie pola poza: DODATEK. Użyj przycisku *Get Fields*, a następnie usuń z listy pole DODATEK.



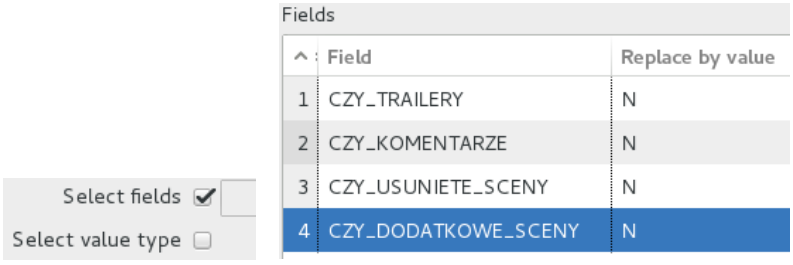
y) Jako pola wynikowe (docelowe) w sekcji *Target fields* wprowadźmy następujące pola:

Nazwa pola docelowego	Pole wartości	Wartość klucza	Typ	Długość
CZY_TRAILERY	Tak	Trailers	String	1
CZY_KOMENTARZE	Tak	Commentaries	String	1
CZY_USUNIETE_SCENY	Tak	Deleted Scenes	String	1
CZY_DODATKOWE_SCENY	Tak	Behind the Scenes	String	1

Target fields:

	Target fieldname	Value fieldname	Key value	Type	Format	Length
1	CZY_TRAILERY	Tak	Trailers	String		1
2	CZY_KOMENTARZE	Tak	Commentaries	String		1
3	CZY_USUNIETE_SCENY	Tak	Deleted Scenes	String		1
4	CZY_DODATKOWE_SCENY	Tak	Behind the Scenes	String		1

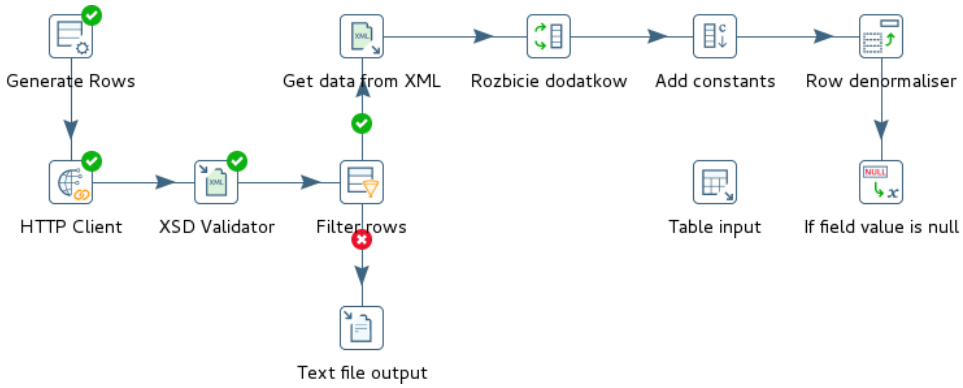
z) W związku z tym, że nie chcemy pozostawiać pustych wartości w dodanych przed chwilą polach, kolejnym krokiem będzie *If field value is null* z katalogu *Utility*. Połącz go z poprzednio dodanym krokiem. Edytując własności komponentu *If field value is null* wskażmy, że puste wartości zostaną zastąpione tylko we wskazanych polach.



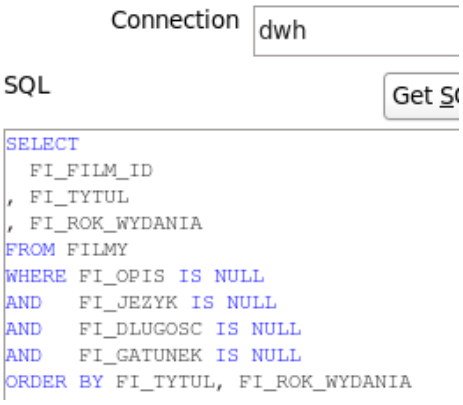
aa) W naszej transformacji pozostało jeszcze do umieszczenia kilka komponentów. Zrobimy to w ramach kolejnego zadania.

4. Strumień danych został już przygotowany. Teraz musimy go wykorzystać do aktualizacji danych w hurtowni danych. W praktyce powinniśmy zastosować odpowiednią technikę wykrywania zmian w źródle (np. opartą na jego poprzednim obrazie), a następnie aktualizować dane w wymiarze przy uwzględnieniu wykrytych zmian. My jednak trochę uprościmy naszą transformację. Wykorzystamy utworzony w poprzednim kroku strumień danych do aktualizacji tylko tych filmów, które nie mają wypełnionych atrybutów pochodzących z naszego zewnętrznego źródła w formacie *XML* ignorując zmiany jakie mogły nastąpić w źródle od czasu poprzednich transformacji.

   a) Dodajmy zatem nowy komponent do naszej transformacji, który będzie wydobywał dane dotyczące filmów, które wymagają aktualizacji. W tym celu dodajmy komponent *Table input* z katalogu *Input*.

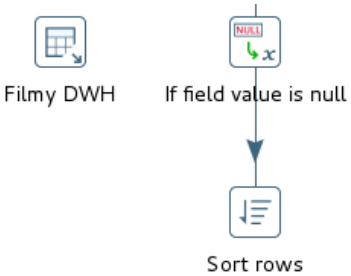


   b) We własnościach tego komponentu zmień jego nazwę na `Filmy DWH`, określ połączenie jako `dwh`. Zapytanie niech wydobywa dane dotyczące filmów wymagających aktualizacji danych, które pozwolą nam połączyć informację z tych dwóch różnych źródeł. W naszym rozwiązaniu zakładamy, że filmy są identyfikowalne za pomocą tytułu i roku produkcji. Koniecznie pamiętajmy zatem o wprowadzeniu odpowiedniego porządku, który potem zostanie wykorzystany podczas połączenia.




```
SELECT
  FI_FILM_ID
, FI_TYTUL
, FI_ROK_WYDANIA
FROM FILMY
WHERE FI_OPIS IS NULL
AND   FI_JEZYK IS NULL
AND   FI_DLUGOSC IS NULL
AND   FI_GATUNEK IS NULL
ORDER BY FI_TYTUL, FI_ROK_WYDANIA
```

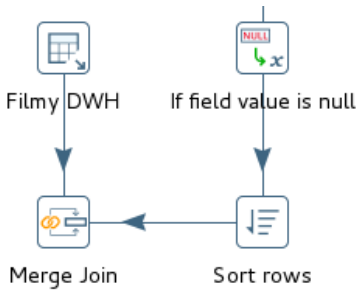
   c) Do połączenia danych wymagane jest posortowanie również źródła pochodzącego z naszej strony WWW. Dlatego dołóżmy komponent *Sort rows*, połączmy go z ostatnim z kroków przetwarzającym dane ze strony WWW, a następnie zdefiniujmy za pomocą jego własności odpowiedni porządek wierszy.



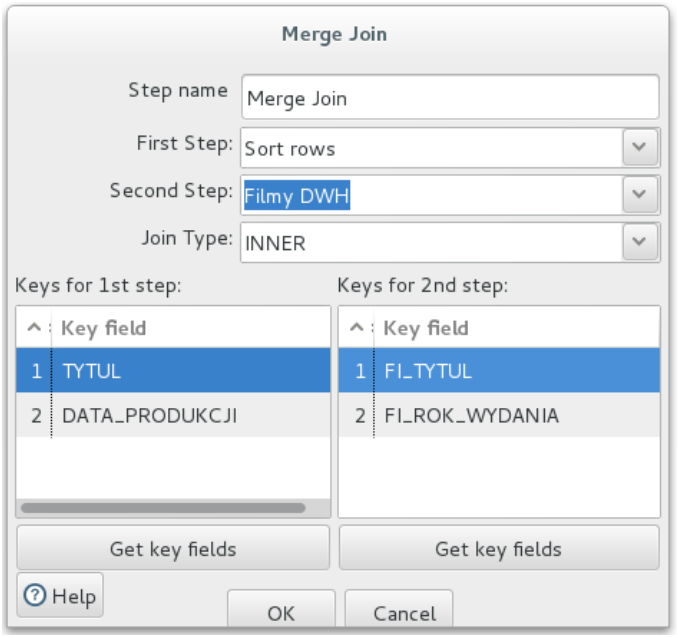
Fields :

	Fieldname	Ascending	Case sensitive compare?
1	TYTUL	Y	N
2	DATA_PRODUKCJI	Y	N

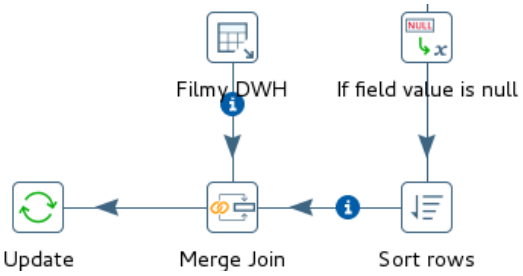
d) Teraz możemy połączyć oba strumienie danych. Dodajmy komponent *Merge Join* z katalogu *Join* i połączmy go z naszymi strumieniami danych.



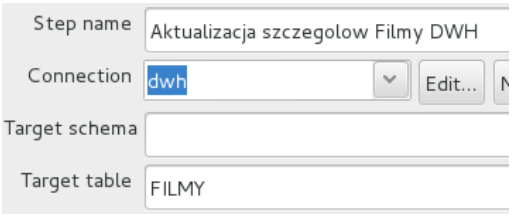
e) Przejdź do własności kroku *Merge Join* i zdefiniuj właściwy sposób połączenia naszych danych.



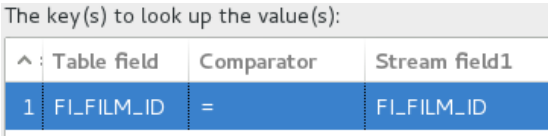
f) Wynikowy strumień połączenia zawiera teraz wszystkie potrzebne dane do wykonania odpowiednich modyfikacji w wymiarze filmy. Dlatego na zakończenie dołożymy do naszej transformacji komponent *Update* z katalogu *Output*, do którego podłączymy wyjście kroku *Merge Join*.



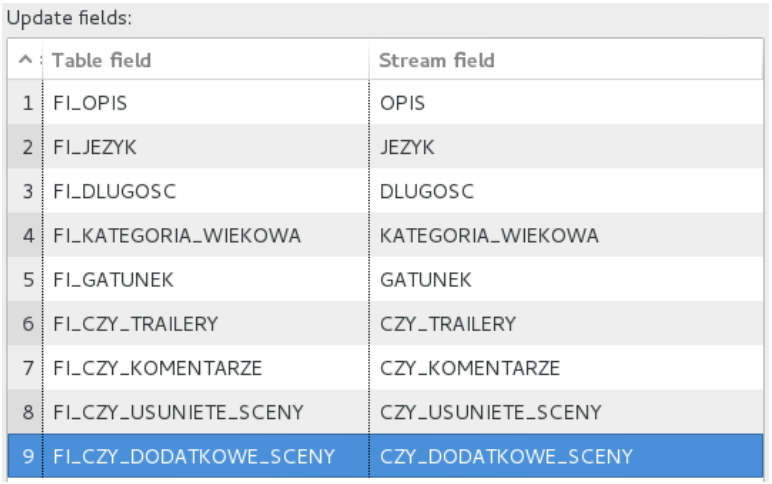
g) Otwórzmy własności kroku *Update*. Zmień nazwę kroku na `Aktualizacja szczegolow Filmy DWH`. Określ nazwę połączenia na `dwh` i tabeli modyfikowanej `FILMY`.



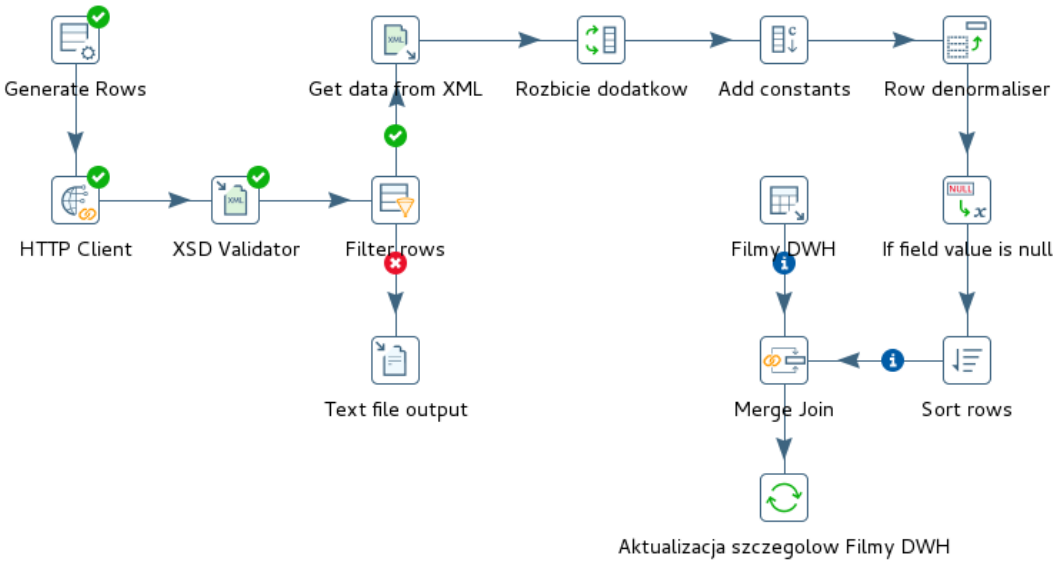
h) Określ warunek modyfikacji.



i) A następnie listę aktualizacji. Uwaga: jeżeli na liście pól tabeli nie ma wszystkich wymaganych pozycji, wywołaj polecenie *Tools->Database-Clear Cache* (z menu *Pentaho DI*).



j) Ostatecznie nasza transformacja powinna wyglądać w sposób następujący



k) Zapisz gotową transformację w katalogu `/wypozyczalnie/ext->dwh` jako `DODAJ FILMY`, a następnie uruchom ją w celu zweryfikowania poprawności jej działania. Upewnij się, że wszystkie filmy zostały zaktualizowane.

Execution Results

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

	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected
1	Generate Rows	0	0	1	0	0	0	0
2	HTTP Client	0	1	1	0	0	0	0
3	XSD Validator	0	1	1	0	0	0	0
4	Filmy DWH	0	0	958	958	0	0	0
5	Filter rows	0	1	1	0	0	0	0
6	Get data from XML	0	1	958	958	0	0	0
7	Rozbicie dodatkow	0	958	2016	0	0	0	0
8	Add constants	0	2016	2016	0	0	0	0
9	Row denormaliser	0	2016	958	0	0	0	0
10	If field value is null	0	958	958	0	0	0	0
11	Text file output	0	0	0	0	0	0	0
12	Sort rows	0	958	958	0	0	0	0
13	Merge Join	0	1916	958	0	0	0	0
14	Aktualizacja szczegolow Filmy DWH	0	958	958	958	0	958	0

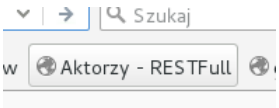
II. USŁUGA SIECIOWA RESTFUL JAKO ŹRÓDŁO DANYCH.

1. Podczas przyszłej analizy preferencji klientów równie interesujące jak kategorie wiekowe czy gatunki poszczególnych filmów mogą być informacje dotyczące tego jacy aktorzy grali w poszczególnych filmach. Tym razem naszym źródłem danych będzie usługa sieciowa *RESTful*. Na początku sprawdźmy jak wygląda jej interfejs oraz na czym polega jej funkcjonalność.

   a) Dla uproszczenia, nasza usługa sieciowa dotycząca aktorów przyjmuje tylko dwa parametry. Pierwszym jest nazwa raportu (`aktorzy`), który generuje listę aktorów. Drugim parametrem jest nazwa filmu. Przykładowy adres URL pozwalający na wywołanie naszej usługi *RESTful* to:

   `http://localhost:8080/ords/actors/data/aktorzy/YOUTH%20KICK`

   Jedna z zakładek na pasku zakładek przeglądarki odwołuje się do powyższego adresu.



   b) Wprowadź powyższy adres do przeglądarki (lub skorzystaj z zakładki) i zaobserwuj postać wynikową.

{"items":[{"actor_id":23,"first_name":"SANDRA","last_name":"KILMER"},{"actor_id":37,"first_name":"VAL","last_name":"BOLGER"},
{"actor_id":124,"first_name":"SCARLETT","last_name":"BENING"},{"actor_id":155,"first_name":"IAN","last_name":"TANDY"},
{"actor_id":198,"first_name":"MARY","last_name":"KEITEL"}]}

   c) Wynik ma format *JSON* i zawiera dane o aktorach, którzy zagrali w filmie `YOUTH KICK` (parametr raportu ogranicza listę aktorów do tych, którzy zagrali w filmie o podanym tytule). Sam wynik jest tablicą zawierającą rekordy o polach `actor_id`, `first_name` i `last_name`. Każde z tych pól posiada prostą zawartość tekstową.

   d) W naszym ćwiczeniu specjalnie wykorzystamy format *JSON*, gdyż ze względu na swoją prostotę zdobywa on coraz większą popularność.

   e) Sprawdźmy jeszcze jak zachowuje się nasza usługa w sytuacji, gdy tytuł filmu nie zostanie przez nią rozpoznany. W tym celu w jako adres URL podajmy następującą wartość:

   `http://localhost:8080/ords/actors/data/aktorzy/ABCD`

{"items":[]}

2. Podobnie jak w poprzednich przypadkach będziemy musieli przygotować naszą hurtownie danych na wprowadzaną modyfikację.

   a) Na początku zalogujmy się do bazy danych za pomocą *SQL*Plus*. 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`

   c) Wprowadźmy zmiany za pomocą następujących poleceń:

```
create table aktorzy (
  AK_AKTOR_ID NUMBER NOT NULL PRIMARY KEY,
  AK_NAZWISKO VARCHAR2(45) NOT NULL,
  AK_IMIE     VARCHAR2(45) NOT NULL);

create table aktorzy_filmy (
  AF_AKTOR_ID NUMBER NOT NULL
              REFERENCES aktorzy(AK_AKTOR_ID),
  AF_FILM_ID  NUMBER NOT NULL
              REFERENCES filmy(FI_FILM_ID),
  UNIQUE (AF_AKTOR_ID, AF_FILM_ID) );
```

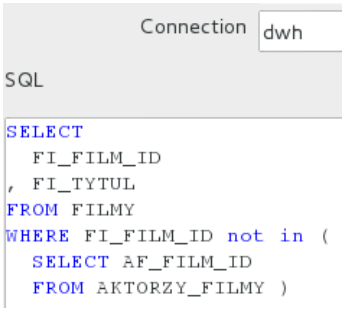
   d) Z oczywistych powodów zmiany w schemacie hurtowni danych powinny

   - być udokumentowane,
   - uwzględniać istniejące transformacje,
   - być przetestowane.

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

3. Czas na naszą transformację. Idea transformacji jest następująca: dla każdego filmu, dla którego nie posiadamy w hurtowni informacji o aktorach będziemy wywoływali naszą usługę *RESTful*. Jej wynik posłuży nam do aktualizacji zarówno informacji o aktorach, jak i do wstawienia powiązań pomiędzy aktorami i filmami, w których grali.

   a) Utwórz nową transformację. Pierwszym krokiem naszej transformacji będzie zapytanie, które wskaże nam filmy z brakującymi informacjami o aktorach. Wstawmy do naszej transformacji krok *Table input*. We własnościach tego kroku określmy zarówno połączenie jak i zapytanie.

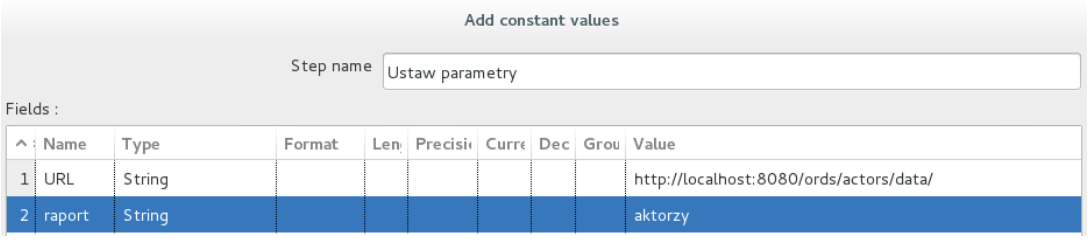



```
SELECT
  FI_FILM_ID
, FI_TYTUL
FROM FILMY
WHERE FI_FILM_ID not in (
  SELECT AF_FILMY_ID
  FROM AKTORZY_FILMY)
```

b) Następny krok wykorzystamy do zdefiniowania adresu i parametrów naszej usługi *RESTful*. Dodajmy do naszej transformacji krok *Add constants* z katalogu *Transform*.

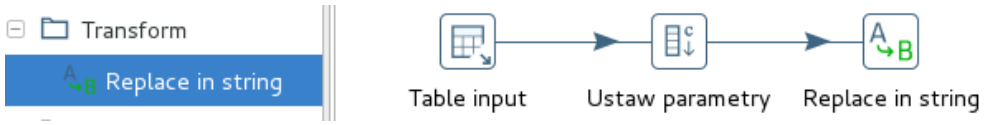
c) We własnościach tego kroku zdefiniujmy dwa pola - adres URL usługi i nazwę wykorzystywanego raportu:

Nazwa (stałej)	Typ	Wartość
URL	String	http://localhost:8080/ords/actors/data/
raport	String	aktorzy

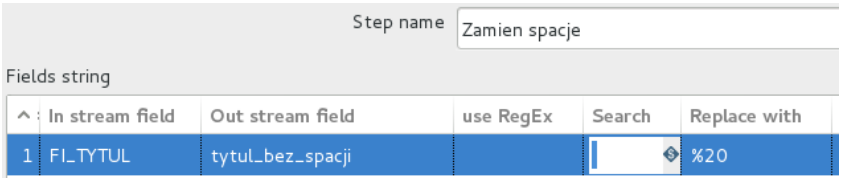


d) Przed wyjściem z własności tego kroku zmieńmy jeszcze jego nazwę na `Ustaw parametry`.

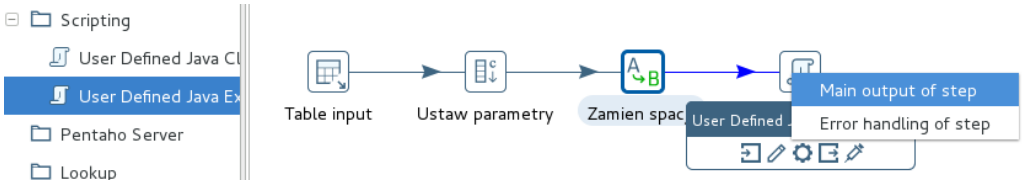
e) Na powyższej liście parametrów brakuje tytułu filmu. Tytuł ten będzie pochodził z zapytania, które napisaliśmy uprzednio. Bezpośrednie wykorzystanie tytułu nie jest jednak możliwe. Spacje, które w tytułach się pojawiają, musimy zamienić na ich odpowiedniki mające następującą postać: `%20`. Dlatego kolejnym krokiem będzie *Replace in string z katalogu Transform*.



f) Wejdźmy do jego właściwości. Zmieńmy nazwę tego kroku na `Zamien spacje`, a następnie zdefiniujmy zastępowanie w tytule filmu spacji na ciąg znaków `%20`. W kolumnie *Search* koniecznie wprowadźmy jedną spację. Wynikową kolumną zawierającą tytuł po zamianie spacji ma być `tytul_bez_spacji`.



g) W tak zdefiniowanym strumieniu możemy przystąpić do utworzenia pełnego adresu URL, który wykorzystamy podczas wywoływania usługi *RESTful*. W tym celu do naszej transformacji wstawmy z katalogu *Scripting* komponent *User Defined Java Expression*. Podczas definiowania połączenia z poprzednim krokiem wskażmy, że główny wynik poprzedniego kroku ma być przekazywany do nowego komponentu.

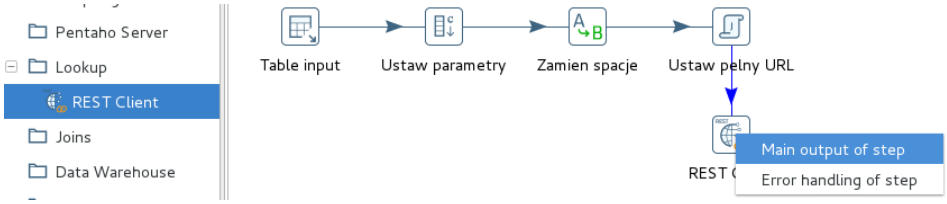


h) We właściwościach dodanego kroku zmieńmy jego nazwę na `Ustaw pelny URL`. Następnie zdefiniujmy w kolumnie *Java expression* sposób tworzenia nowego pola `PELNY_URL` w strumieniu za pomocą następującego wyrażenia:

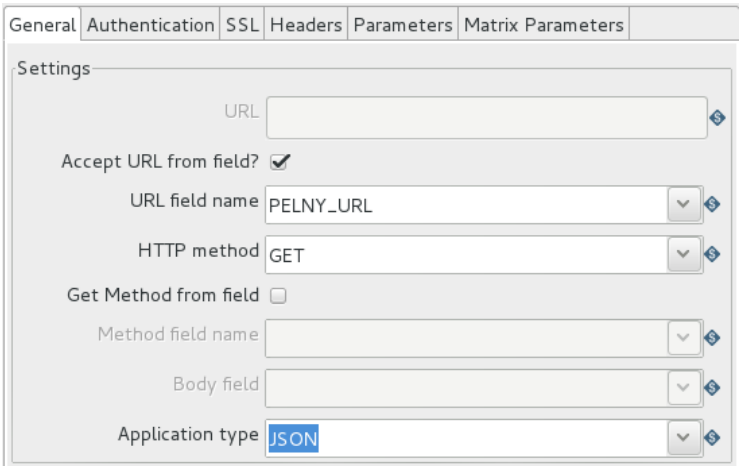
```
URL+raport"/"+Tytul_bez_spacji
```

i) Nie zapomnijmy wskazać jego typu: `String`

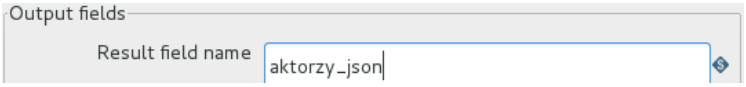
j) Mając przygotowany adres URL możemy dołączyć nasz ostatni krok do kolejnego komponentu, którym będzie *REST Client* z katalogu *Lookup*.



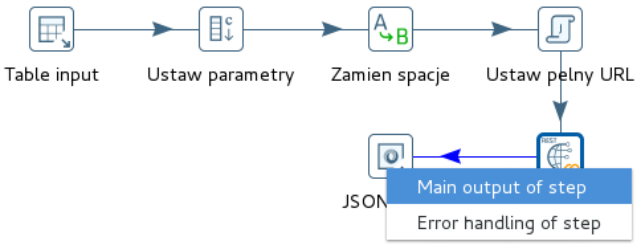
k) Otwórzmy właściwości tego komponentu i zdefiniujmy źródło adresu URL, metodę HTTP, która będzie wykorzystana podczas wywoływania usługi oraz format wyniku (w polu *Application type*)



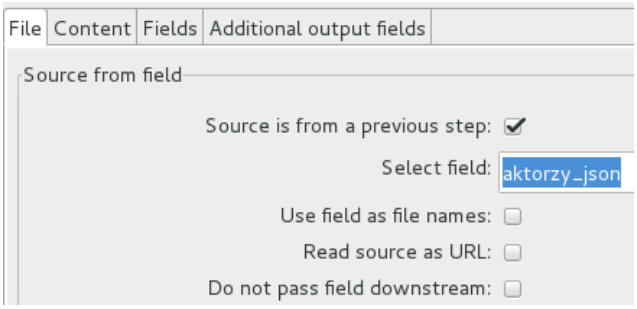
l) Następnie podajmy pole `aktorzy_json`, które będzie zawierało wynik wywołania naszej usługi.



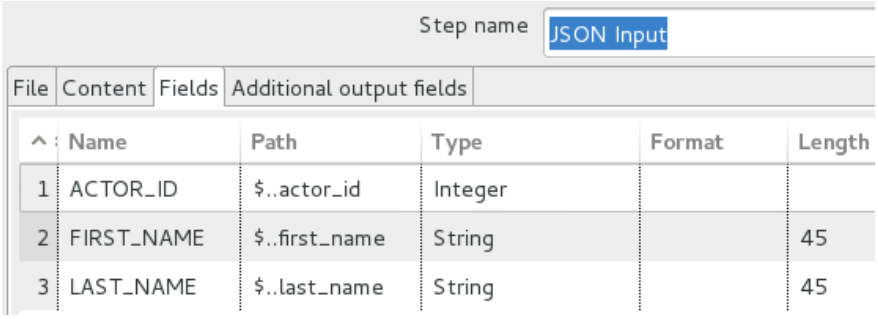
m) Czas na obsługę wyniku naszej usługi *RESTful*. Jeszcze do niedawna *Pentaho DI* nie posiadało żadnego specjalizowanego komponentu do obsługi danych w formacie *JSON*, co skazywało programistów na programistyczną obsługę takich danych. W chwili obecnej możemy skorzystać z odpowiednika użytego wcześniej komponentu do przetwarzania danych w formacie *XML*. Tym odpowiednikiem dla formatu *JSON* jest komponent *Json Input* z katalogu *Input*. Przekierujmy do niego główny strumień danych z poprzedniego kroku.



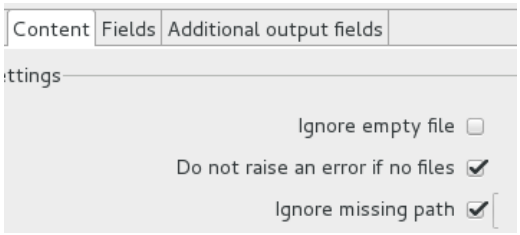
n) Na zakładce *File* we właściwościach *Json Input* wskażmy, że źródłem danych w formacie *JSON* będzie pole `aktorzy_json`.



o) Na zakładce *Fields* zdefiniujmy trzy pola, które będą wydobywały dane z wyniku naszej usługi. Utworzymy trzy pola: `ACTOR_ID`, `FIRST_NAME` i `LAST_NAME` oparte odpowiednio na ścieżkach: `$..actor_id`, `$..first_name` i `$..last_name`. Typy tych pól niech będą odpowiednio `Integer`, `String` i `String`. W przypadku pól będących ciągami znaków określmy maksymalną długość na `45`.



p) Przypominając sobie, że odpowiedź usługi *RESTFul* niekoniecznie musi posiadać powyższe ścieżki, przejdźmy na zakładkę *Content*, a następnie zaznaczmy pole wyboru *Ignore missing path* w sekcji *Settings*.

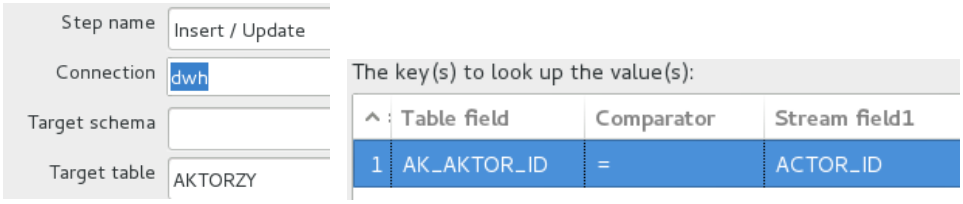


q) W przypadku, gdy dla danego filmu nie zostanie odnaleziony żaden aktor, żadna krotka nie zostanie wygenerowana. Nie ma więc konieczności testowania przed aktualizacją hurtowni danych, czy dane aktora istnieją.

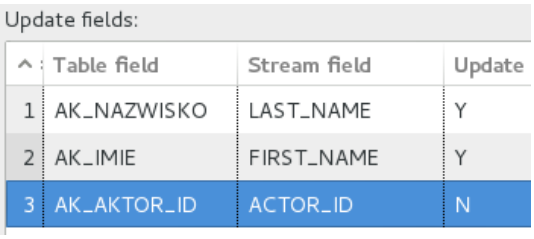
r) Utworzyliśmy zatem strumień, który może być wykorzystany do aktualizacji zawartości hurtowni danych. Mamy zamiar aktualizować dwie tabele: `AKTORZY` oraz `AKTORZY_FILMY`. Aktualizacja drugiej z nich musi następować po aktualizacji pierwszej, dlatego obie transformacje umieścimy w jednym strumieniu przetwarzania, zachowując właściwą kolejność. Wstawmy do naszej transformacji

pierwszy z kroków *Insert/Update* z katalogu *Output*.

s) We własnościach określmy tabelę, która będzie podlegała modyfikacji oraz warunek modyfikacji.

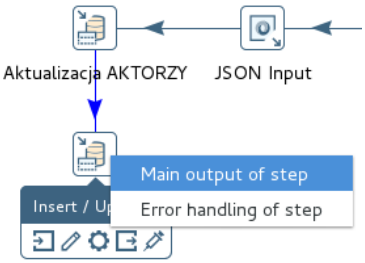


t) Podajmy także listę par atrybut w tabeli – pole w strumieniu, które będą odpowiadały za właściwe przeprowadzenie modyfikacji (i wstawienia nowych wierszy)

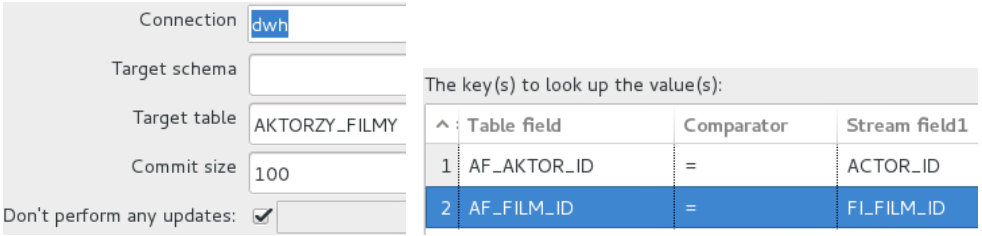


u) Na zakończenie edycji własności tego kroku zmieńmy jego nazwę na `Aktualizacja AKTORZY`.

v) Zgodnie z wcześniejszą analizą, następny komponent będzie aktualizował zawartość tabeli `AKTORZY_FILMY`. Dołóżmy do transformacji kolejny komponent *Insert/Update* i połączmy go z poprzednim przekierowując do niego główny wynik poprzedniego kroku.



w) We własnościach określmy odbiorcę naszych aktualizacji, a także to, że nie będą wykonywane żadne aktualizacje (jedynie wstawienia nowych wierszy). Podajmy także definicję powiązania pomiędzy krotkami w tabeli i ich odpowiednikami w strumieniu.



x) Ponadto określmy odpowiadające sobie atrybuty tabela-strumień w celu właściwego wykonania operacji wstawiania nowych wierszy.

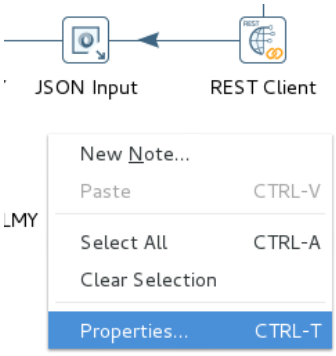
Update fields:

	Table field	Stream field	Update
1	AF_FILM_ID	FI_FILM_ID	N
2	AF_AKTOR_ID	ACTOR_ID	N

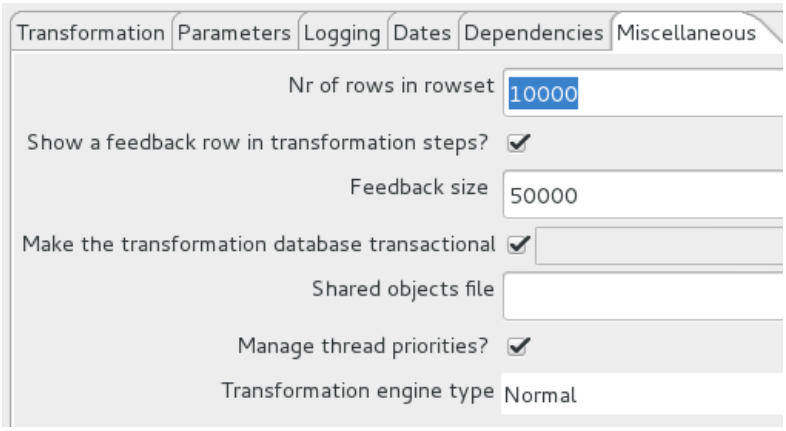
y) Przed zakończeniem edycji własności zmieńmy nazwę tego kroku na: `Aktualizacja AKTORZY_FILMY`.

z) W chwili obecnej komponenty `Aktualizacja AKTORZY` i `Aktualizacja AKTORZY_FILM` mogą być uruchomione w ramach różnych sesji (transakcji bazy danych) korzystających z połączenia `dwh`. W rezultacie może się zdarzyć, że drugi komponent nie będzie widział zmian dokonanych w pierwszym (np. w sytuacji gdy zatwierdzanie będzie trwało zbyt długo). Dlatego dokonamy teraz modyfikacji opcji transformacji.

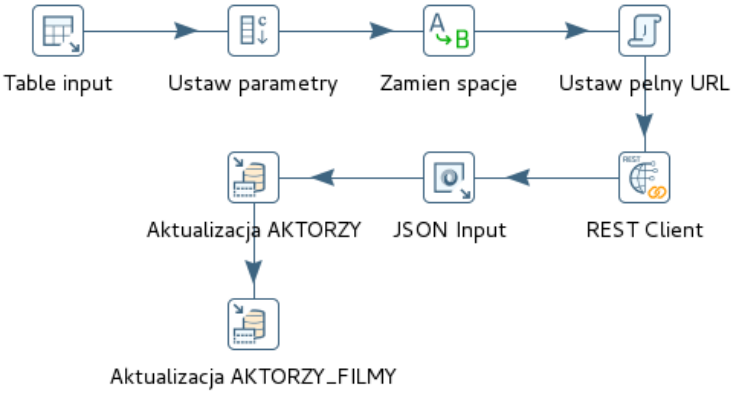
aa) Za pomocą menu kontekstowego na pustym polu diagramu wywołaj opcję *Properties…*.



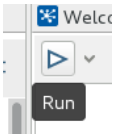
bb) Na zakładce *Miscellaneous* zaznacz opcję *Make the transformation database transactional*. Efekt tej zmiany będzie taki, ze transformacja będzie korzystała z jednej sesji dla każdego połączenia z bazą danych, a to oznacza, że każdy komponent będzie widział wszystkie modyfikacje wprowadzone przez poprzednie komponenty wykorzystujące to samo połączenie.



cc) Zapiszmy transformację w katalogu `/wypozyczalnie/ext->dwh` jako `DODAJ AKTOROW`. Ostatecznie postać tej transformacji powinna przypominać poniższą.



dd) Teraz możemy uruchomić naszą transformację.



ee) Wywoływanie naszych usług *RESTful* dla każdego filmu może trochę potrwać. Ostateczny wynik powinien być analogiczny do poniższego.

Execution Results

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

	Stepname	Copynr	Read	Written	Input	Output	Updated
1	Table input	0	0	958	958	0	0
2	Ustaw parametry	0	958	958	0	0	0
3	Zamien spacje	0	958	958	0	0	0
4	Ustaw pelny URL	0	958	958	0	0	0
5	REST Client	0	958	958	0	0	0
6	JSON Input	0	958	5246	5246	0	0
7	Aktualizacja AKTORZY	0	5246	5246	5246	200	0
8	Aktualizacja AKTORZY_FILMY	0	5246	5246	5246	5246	0

ff) Na zakończenie utwórzmy zadanie o nazwie `LADOWANIE FILMOW`, który połączy obie stworzone przez nas transformacje w jedną całość.

gg) Zapiszmy je w katalogu `/wypozyczalnie/ext->dwh` i sprawdźmy czy jego uruchomienie kończy się sukcesem.

Execution Results

History | Logging | Job metrics | Metrics

Job / Job Entry	Comment	Result
LADOWANIE FILMOW		
Job: LADOWANIE FILMO'	Start of job execution	
START	Start of job execution	
START	Job execution finished	Success
DODAJ FILMY	Start of job execution	
DODAJ FILMY	Job execution finished	Success
DODAJ AKTOROW	Start of job execution	
DODAJ AKTOROW	Job execution finished	Success
Success	Start of job execution	
Success	Job execution finished	Success
Job: LADOWANIE FILMO'	Job execution finished	Success