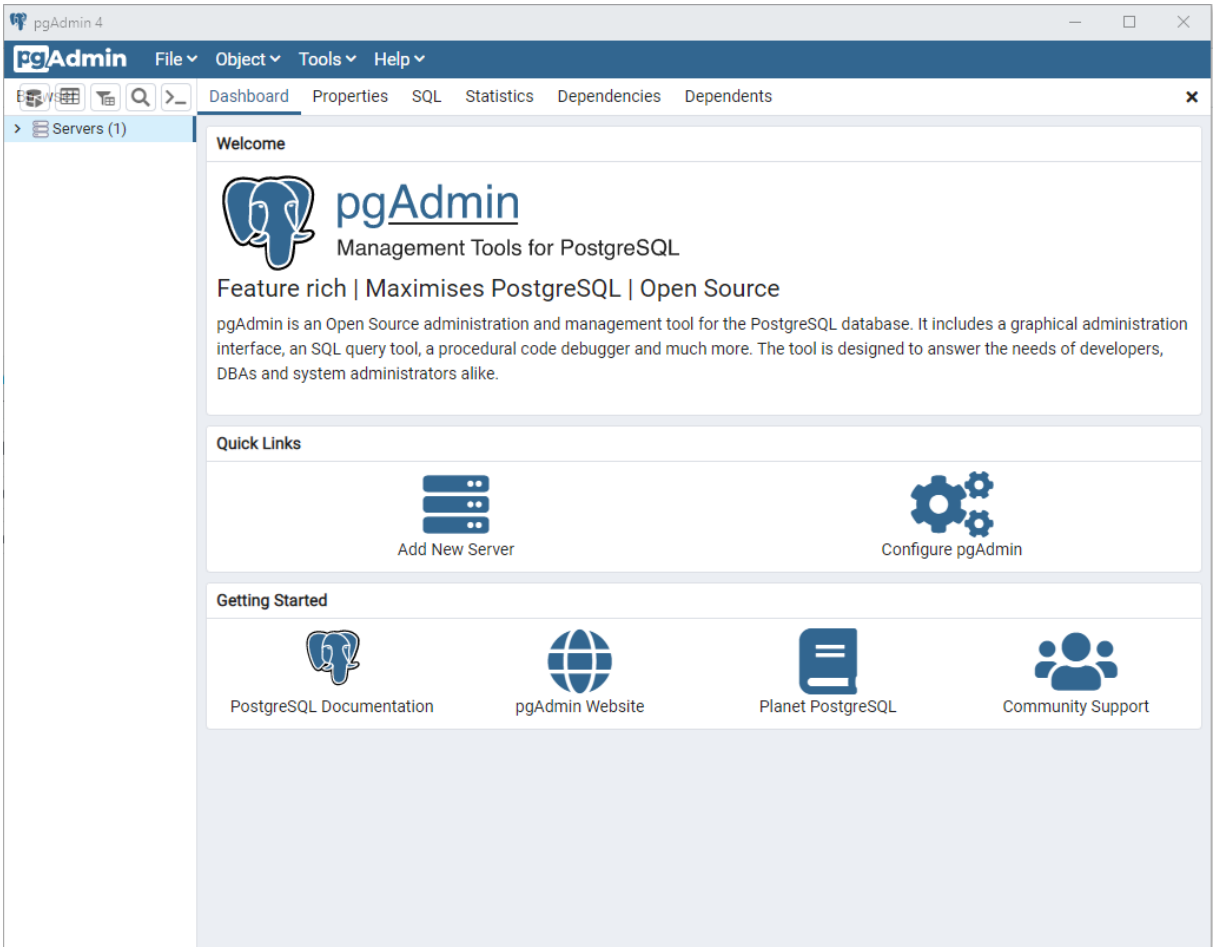
Zaawansowane systemy baz danych

Projekt – etap 3

Krzysztof Wyszyński

1. System bazodanowy

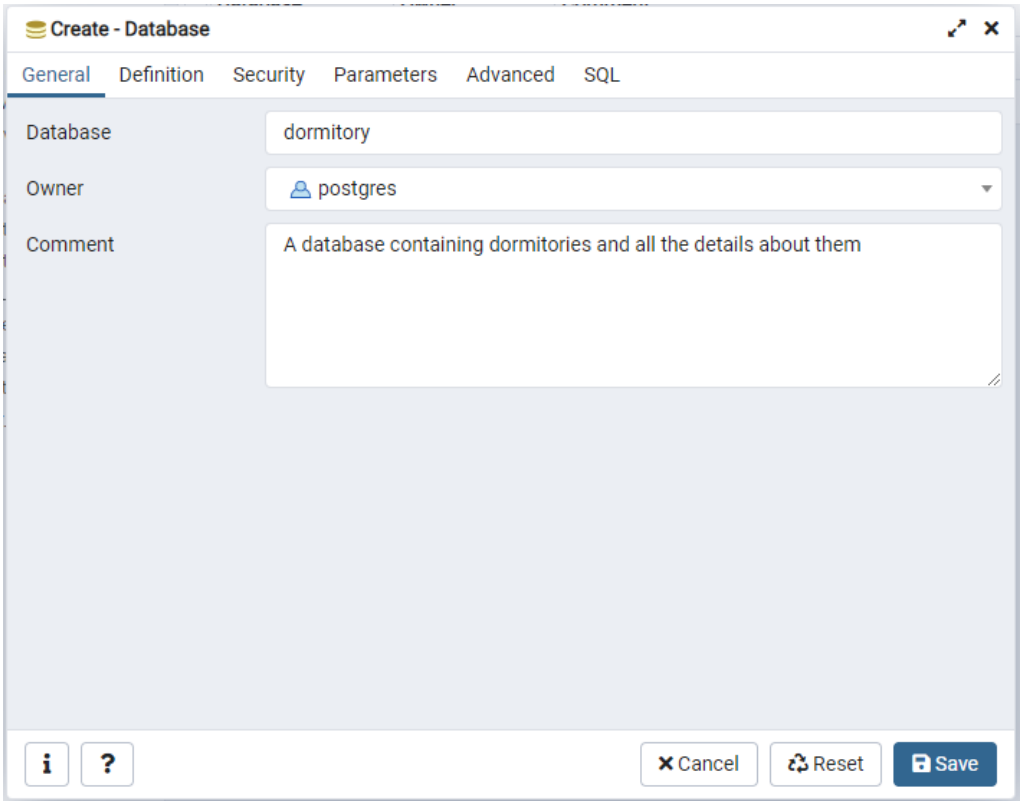
W celu migracji dotychczas stworzonej bazy danych postanowiłem wykorzystać obiektowo-relacyjny system bazodanowy PostgreSQL. Poza dedykowanym narzędziem *pgAdmin* pozwalającym na zarządzanie serwerem bazodanowym za pomocą interfejsu graficznego, PostgreSQL oferuje również dostęp do zasobów z poziomu konsoli.

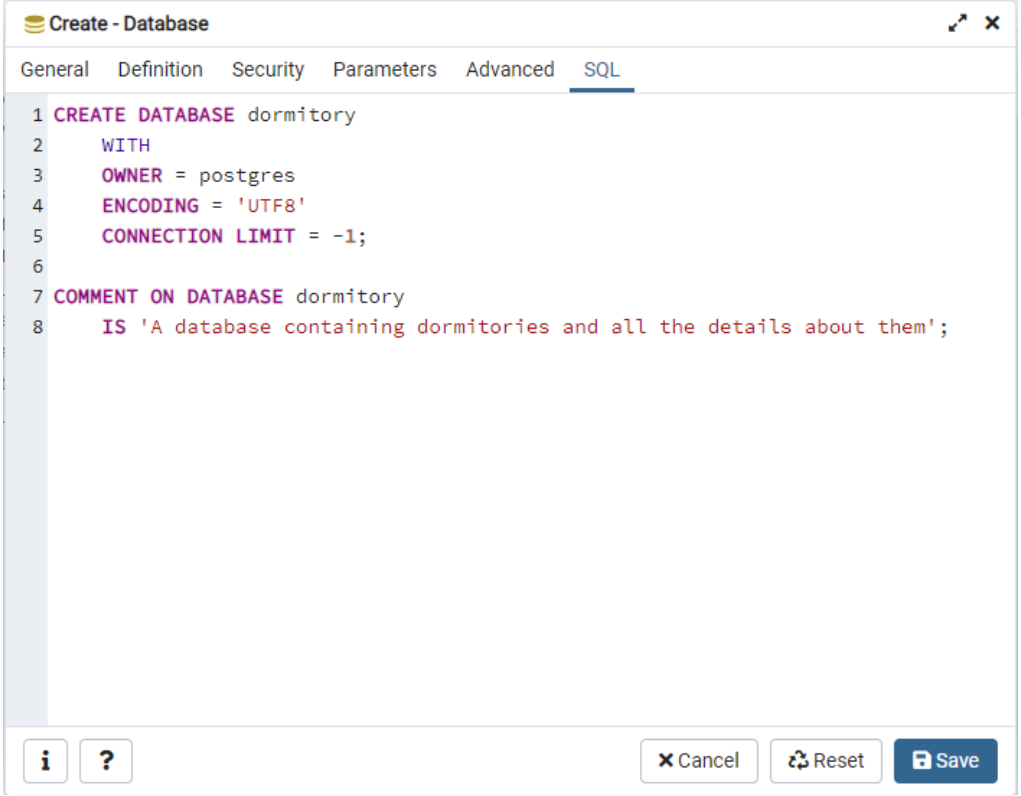


Ekran główny narzędzia pgAdmin

PostgreSQL to projekt *open source*, co stanowi dużą zaletę tego rozwiązania. Przykładowo, platforma chmurowa Heroku oferuje możliwość darmowego stworzenia bazy danych PostgreSQL o ograniczonych zasobach (maksymalnie 20 połączeń, 10000 rekordów), co pozwala developerom na wdrażanie drobnych projektów na środowisko produkcyjne, na przykład w celu zaprezentowania swoich umiejętności przyszłym pracodawcom.

Za pomocą narzędzia *pgAdmin* stworzyłem nową bazę danych:





2. Analiza różnic

Poniżej przedstawiono analizę różnic między systemami PostgreSQL oraz SQL Server.

2.1. Składnia

W celu stworzenia bazy danych w nowym systemie koniecznym okazało się przepisanie wcześniej wygenerowanego skryptu w taki sposób, aby był on zgodny ze składnią PostgreSQL.

2.1.1. Wstępne usuwanie tabel

```
DROP TABLE IF EXISTS announcement;
DROP TABLE IF EXISTS issue;
```

Usuwanie tabel - PostgreSQL

```
IF  EXISTS (SELECT *
FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[announcement]') AND type in (N'U'))
DROP TABLE [dbo].[announcement]
GO

IF  EXISTS (SELECT *
FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[issue]') AND type in (N'U'))
DROP TABLE [dbo].[issue]
GO
```

Usuwanie tabel – SQL Server

Należy podkreślić, iż PostgreSQL narzuca obowiązek umieszczania średnika na końcu wykonywanej komendy, co nie było konieczne w składni MSSQL.

2.1.2. Deklaracja klucza głównego

```
id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
```

Deklaracja klucza głównego - PostgreSQL

```
id INT PRIMARY KEY IDENTITY(1, 1),
```

Deklaracja klucza głównego – SQL Server

2.1.3. Deklaracja klucza obcego

```
address_id INT REFERENCES address(id) UNIQUE  NOT NULL
```

Deklaracja klucza obcego – PostgreSQL

```
address_id INT FOREIGN KEY REFERENCES address(id) UNIQUE  NOT NULL
```

Deklaracja klucza obcego – SQL Server

2.1.4. Typy danych

a) Najmniejszym typem numerycznym oferowanym przez PostgreSQL jest *SMALLINT*. System MSSQL pozwala na wykorzystanie jednobajtowego typu *TINYINT* o zakresie (0,255). W związku z tym kolumny wcześniej zadeklarowane jako TINYINT zostały zastąpione przez *SMALLINT*.

b) W poprzednich etapach do deklaracji kolumn przechowujących datę wykorzystano typ *SMALLDATETIME*, zajmujący 4 bajty pamięci. Alternatywą oferowaną przez PostgreSQL jest 8-bajtowy typ *TIMESTAMP*.

2.1.5. Umieszczanie danych w tabeli

Istotną różnicą dotyczącą tworzenia rekordów w PostgreSQL względem SQL Server jest konieczność uwzględnienia kolumn, do których będą umieszczane dane. Przykładowo, poniżej przedstawiona próba nie zostanie wykonana:

```
insert into address values
('Plac Politechniki 1', null, '00-661', 'Warszawa', 'Poland')
```

```
ERROR:  invalid input syntax for type integer: "Plac Politechniki 1"
LINE 2: ('Plac Politechniki 1', null, '00-661', 'Warszawa', 'Poland'...
         ^
```

Wynika to z faktu, iż pierwszą wartością w tabeli *address* jest id – klucz główny. W składni T-SQL kolumna ta była automatycznie ignorowana. Poniżej przedstawiono poprawną implementację powyższej operacji:

```
1  insert into address(street, flat_number, postcode, city, country) values
2  ('Plac Politechniki 1', null, '00-661', 'Warszawa', 'Poland')
```

```
INSERT 0 1

Query returned successfully in 100 msec.
```

2.1.6. Wyszukiwanie rekordów

PostgreSQL nie pozwala na używanie pojedynczych apostrofów (‘ ’) podczas wykonywania zapytania SELECT <column> AS <name>:

```
1  SELECT street AS 'street' FROM address
```

```
ERROR:  syntax error at or near "'street'"
LINE 1: SELECT street AS 'street' FROM address
                         ^
```

Konieczne jest wykorzystanie apostrofów podwójnych:

```
1  SELECT street AS "street" FROM address
```

	street character varying (70)
1	Plac Politechniki 1
2	Krakowskie Przedmieście 26/28
3	Nowoursynowska 166
4	Al.Niepodległości 162
5	Żwirki i Wigury 61
6	Grójecka 39
7	Księcia Janusza 39
8	Ludwika Waryńskiego 10
9	Akademicka 5
10	Karolkowa 84
11	Batalionu AK Pięść 9
12	Radomska 11
13	Żwirki i Wigury 97/99
14	Al.Niepodległości 147

2.1.7. Perspektywy

W przypadku perspektyw nie wystąpiły żadne różnice między dialektami inne niż wymienione we wcześniejszych punktach.

2.1.8. Funkcje

Poniżej przedstawiono składnię tworzenia funkcji za pomocą PostgreSQL:

Synopsis

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
    [ WITH ( attribute [, ...] ) ]
```

Ze względu na różnice składniowe konieczne było dostosowanie wcześniej napisanych funkcji do nowego środowiska:

```
CREATE OR REPLACE FUNCTION StudentDataByPesel (pesel CHAR(11))
RETURNS SETOF students
AS
$$
BEGIN
RETURN QUERY(SELECT * FROM students s WHERE s."PESEL"=pesel);
END;
$$
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION NonFullRooms(dormitoryName VARCHAR(100))
RETURNS TABLE (
    id INT,
    number VARCHAR(10),
    capacity SMALLINT,
    dormitory VARCHAR(100),
    locators BIGINT,
    "available spots" BIGINT
)
AS
$$
BEGIN
RETURN QUERY(SELECT *, (r."capacity" - r."locators")
FROM rooms r WHERE r."dormitory" LIKE ('%' || dormitoryName || '%') AND r."capacity" > r."locators")
;
END;
$$
LANGUAGE plpgsql;
```

Wywoływanie funkcji odbywa się w taki sam sposób jak w przypadku MSSQL:

```
SELECT * FROM StudentDataByPesel('95050859964');
SELECT * FROM NonFullRooms('Bratniak')
```

2.1.9. Procedury

Poniżej przedstawiono składnię tworzenia procedur za pomocą PostgreSQL:

Synopsis

```
CREATE [ OR REPLACE ] PROCEDURE
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
```

Jak można zauważyć, różni się ona od składni SQL Server, w związku z tym konieczna była modyfikacja kodu odpowiedzialnego za procedury. Poniżej przedstawiono implementację procedury *addAnnouncement* w systemie PostgreSQL:

```
DROP PROCEDURE IF EXISTS addAnnouncement(text,smallint);
CREATE PROCEDURE addAnnouncement(
content text = NULL,
administrator_id SMALLINT = NULL
)
AS
$$
BEGIN
IF NOT EXISTS (SELECT aws.id FROM administration_workers aws WHERE aws.id=administrator_id) THEN
RAISE 'Please enter a valid administration_worker_id';
END IF;
INSERT INTO announcement(date, content, administration_worker_id) VALUES(
    CURRENT_TIMESTAMP, content, administrator_id
);
END;
$$
LANGUAGE plpgsql;
```

Wywołanie procedury możliwe jest z wykorzystaniem jednej z dwóch metod – pierwsza z nich to umieszczenie argumentów w odpowiedniej kolejności. Dodatkowo, w przypadku niektórych danych konieczne jest podanie konkretnych typów.

```
CALL addAnnouncement('Ważne ogłoszenie', 1::smallint)
```

Drugi sposób to wywołanie procedury przy użyciu argumentów nazwanych:

```
CALL addAnnouncement(
    content => 'Ważne ogłoszenie',
    administrator_id => 1 :: SMALLINT
);
```

2.1.10. Wyzwalacze

PostgreSQL wymaga podania nazwy tabeli, której dotyczy wyzwalacz podczas jego usuwania:

Synopsis

```
DROP TRIGGER [ IF EXISTS ] name ON table [ CASCADE | RESTRICT ]
```

```
DROP TRIGGER IF EXISTS trg_student
```

Usuwanie wyzwalacza – SQL Server

```
DROP TRIGGER IF EXISTS trg_student ON student;
```

Usuwanie wyzwalacza - PostgreSQL

Składnia tworzenia wyzwalacza również różni się od składni używanej w SQL Server:

Synopsis

```
CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }
    ON table
    [ FROM referenced_table_name ]
    [ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } ]
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( condition ) ]
    EXECUTE PROCEDURE function_name ( arguments )

where event can be one of:

    INSERT
    UPDATE [ OF column_name [, ... ] ]
    DELETE
    TRUNCATE
```

Aby utworzyć wyzwalacz, należy wcześniej zaimplementować funkcję, która zostanie przez niego wywołana oraz wywołać ją w ciele wyzwalacza:

```
DROP TRIGGER IF EXISTS trg_student ON student;

CREATE OR REPLACE FUNCTION validateIfPersonIsAStudent() RETURNS TRIGGER
AS
$$
BEGIN
IF EXISTS (SELECT aw.person_id FROM administration_worker aw WHERE aw.person_id = NEW.person_id) THEN
RAISE 'This person is already an administration worker';
END IF;
IF EXISTS (SELECT d.person_id FROM doorkeeper d WHERE d.person_id = NEW.person_id) THEN
RAISE 'This person is already a doorkeeper';
END IF;
RETURN NULL;
END;
$$
LANGUAGE PLPGSQL;

CREATE TRIGGER trg_student
BEFORE INSERT
ON student
FOR EACH ROW EXECUTE PROCEDURE validateIfPersonIsAStudent();
```

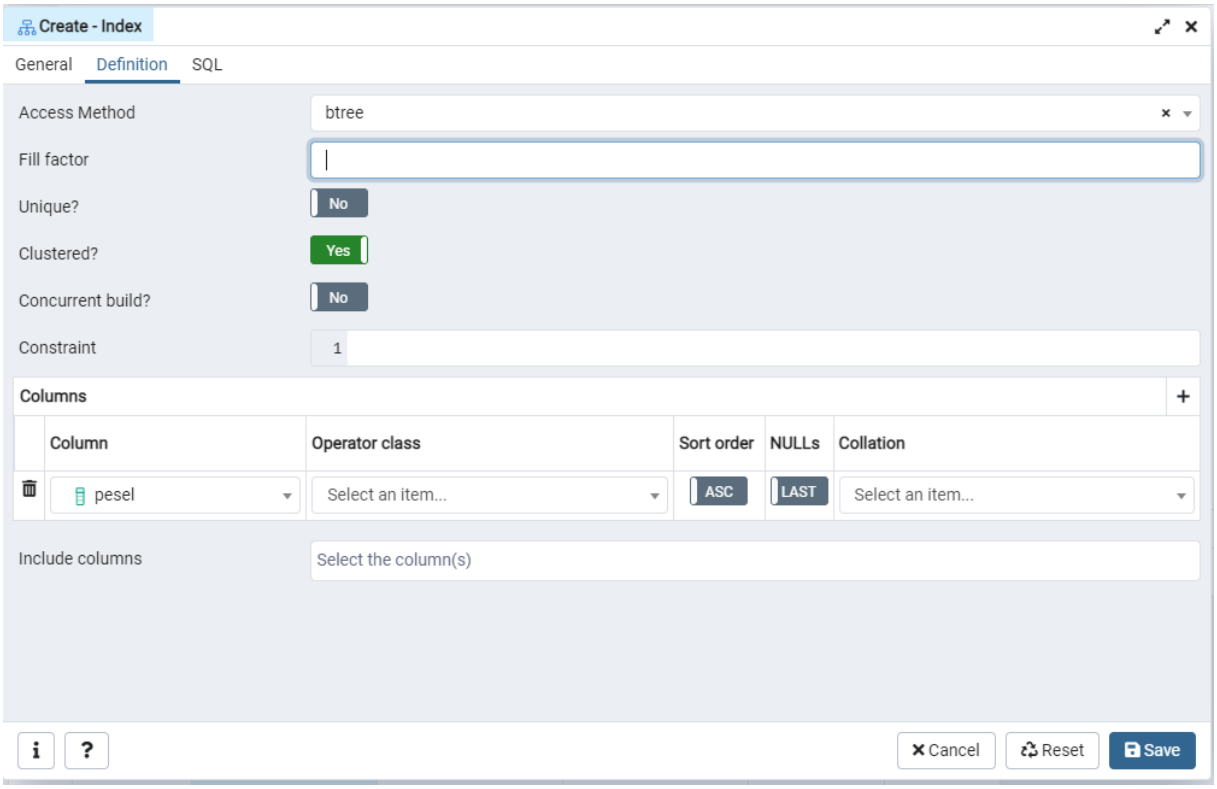
W przeciwieństwie do SQL Server, PostgreSQL pozwala na użycie dyrektywy **BEFORE INSERT** (w poprzednim etapie projektu wykorzystano **INSTEAD OF INSERT**, co wiązało się z koniecznością ponownego wstawienia danych po ich walidacji). Za pomocą słowa kluczowego NEW można odwoływać się do nowo wstawianych pól.

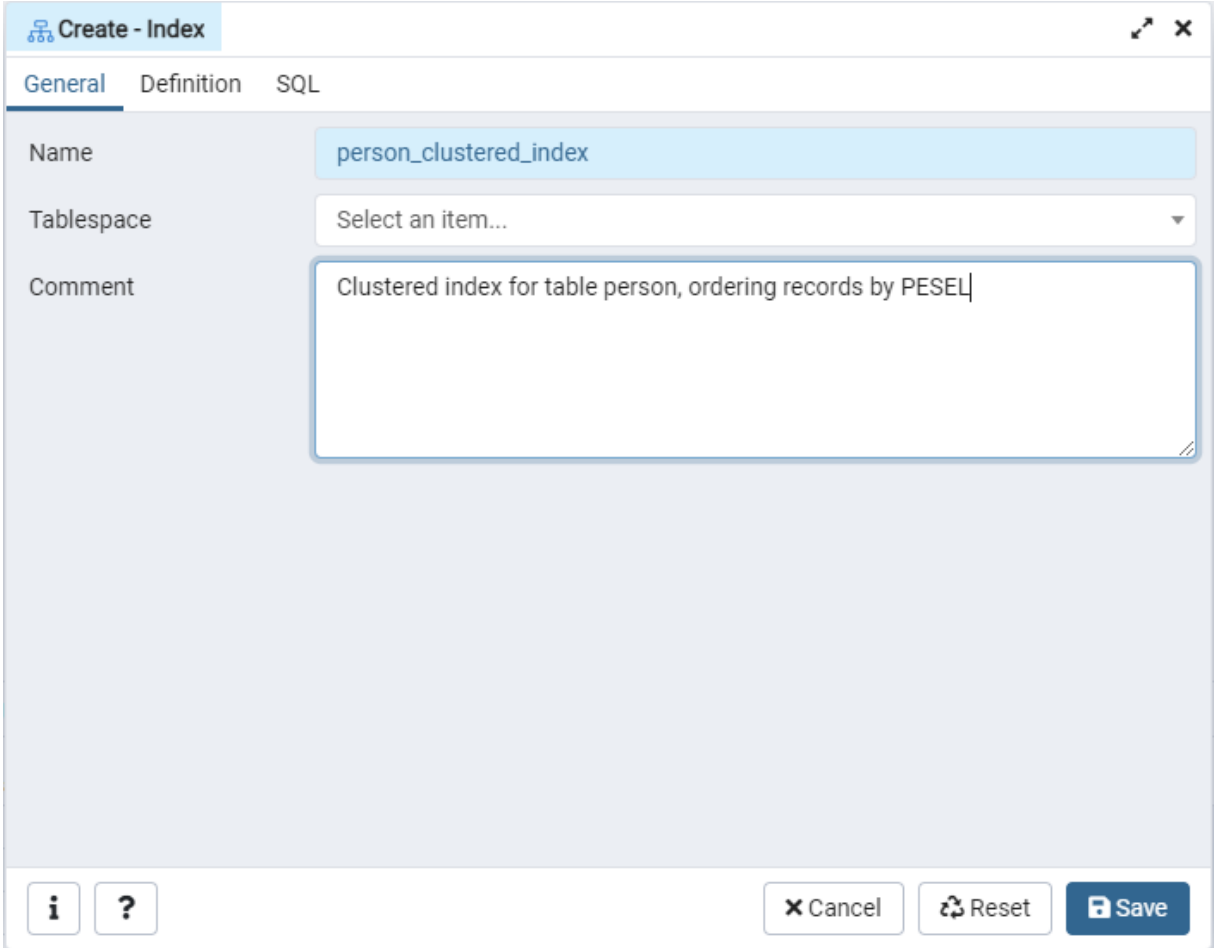
2.2. Indeksy

Podczas tworzenia indeksów w poprzednim etapie projektu okazało się, że SQL Server domyślnie nakłada indeks zgrupowany na klucz główny tabeli, co niekoniecznie było rozwiązaniem pożądanym. PostgreSQL nie narzuca domyślnego indeksu klastrowanego, lecz nakłada na klucz główny indeks niezgrupowany:

```
dormitory=# \d person
                                  Table "public.person"
   Column    |         Type          | Collation | Nullable |           Default
-------------+-----------------------+-----------+----------+-------------------------------
 id          | integer               |           | not null | generated always as identity
 first_name  | character varying(25) |           | not null |
 second_name | character varying(25) |           |          |
 last_name   | character varying(50) |           | not null |
 pesel       | character(11)         |           | not null |
 address_id  | integer               |           | not null |
Indexes:
    "person_pkey" PRIMARY KEY, btree (id)
    "person_pesel_key" UNIQUE CONSTRAINT, btree (pesel)
Foreign-key constraints:
    "person_address_id_fkey" FOREIGN KEY (address_id) REFERENCES address(id)
Referenced by:
    TABLE "accomodation" CONSTRAINT "accomodation_person_id_fkey" FOREIGN KEY (person_id) REFERENCES person(id)
    TABLE "administration_worker" CONSTRAINT "administration_worker_person_id_fkey" FOREIGN KEY (person_id) REFERENCES p
erson(id)
    TABLE "doorkeeper" CONSTRAINT "doorkeeper_person_id_fkey" FOREIGN KEY (person_id) REFERENCES person(id)
    TABLE "student" CONSTRAINT "student_person_id_fkey" FOREIGN KEY (person_id) REFERENCES person(id)
```

Jak można zauważyć, domyślną strukturą danych dla generowanych indeksów jest drzewo binarne (*btree)*, lecz dostępne są też inne struktury, takie jak *heap, hash, gist, gin, spgist, brin*. Za pomocą narzędzia *pgAdmin* można w wygodny sposób tworzyć indeksy, podobnie jak w przypadku Microsoft SQL Server Management Studio.





Po utworzeniu indeksu zgrupowanego należy „przeklastrować” tabelę, za pomocą komendy **CLUSTER** *<table_name>.*

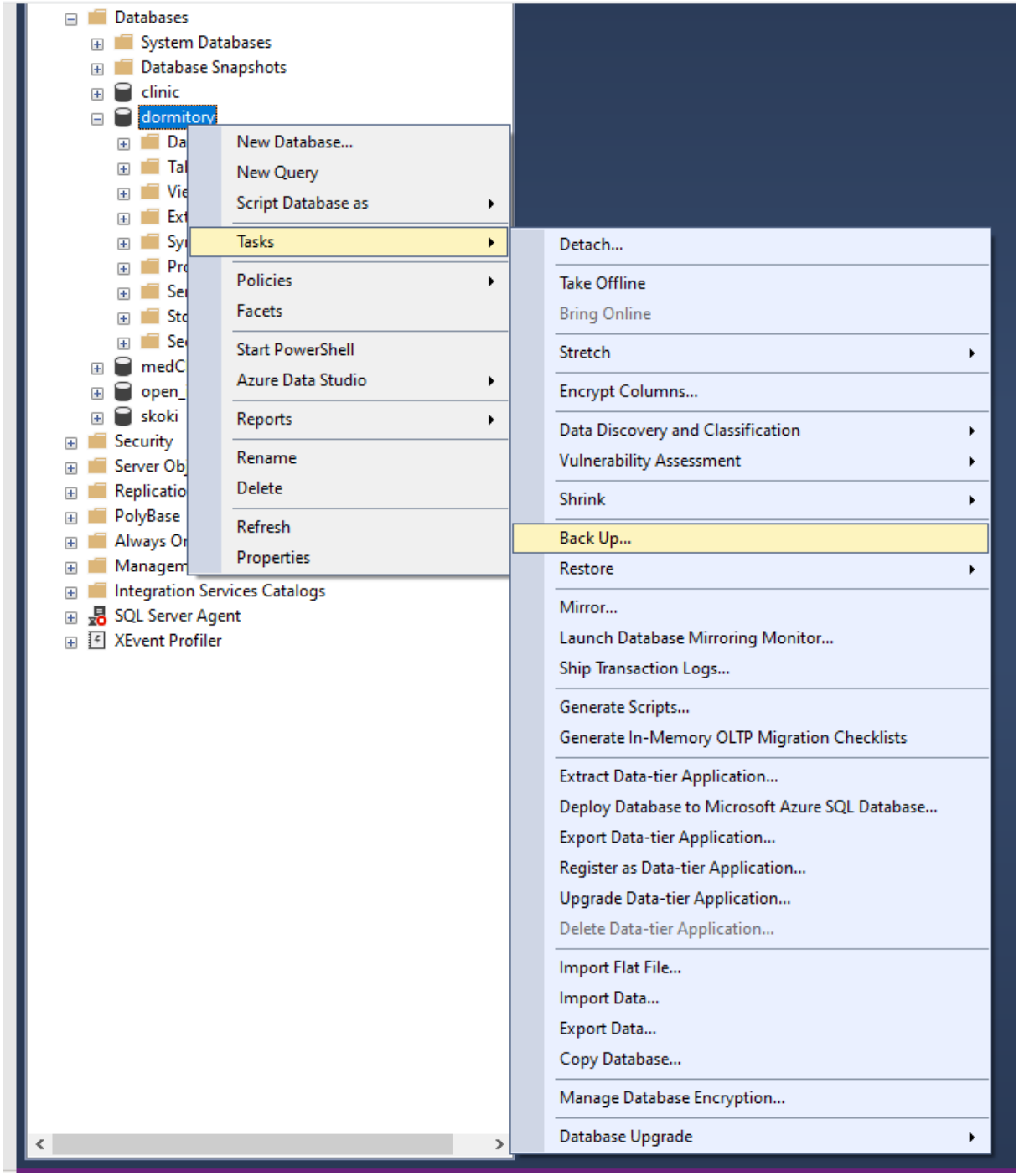
```
dormitory=# \d person
                                    Table "public.person"
   Column    |          Type          | Collation | Nullable |            Default
-------------+------------------------+-----------+----------+-----------------------------
 id          | integer                |           | not null | generated always as identity
 first_name  | character varying(25)  |           | not null |
 second_name | character varying(25)  |           |          |
 last_name   | character varying(50)  |           | not null |
 pesel       | character(11)          |           | not null |
 address_id  | integer                |           | not null |
Indexes:
    "person_pkey" PRIMARY KEY, btree (id)
    "person_clustered_index" btree (pesel) CLUSTER
    "person_pesel_key" UNIQUE CONSTRAINT, btree (pesel)
```

W rezultacie tabela zostanie fizycznie i logicznie uporządkowana zgodnie z indeksem:

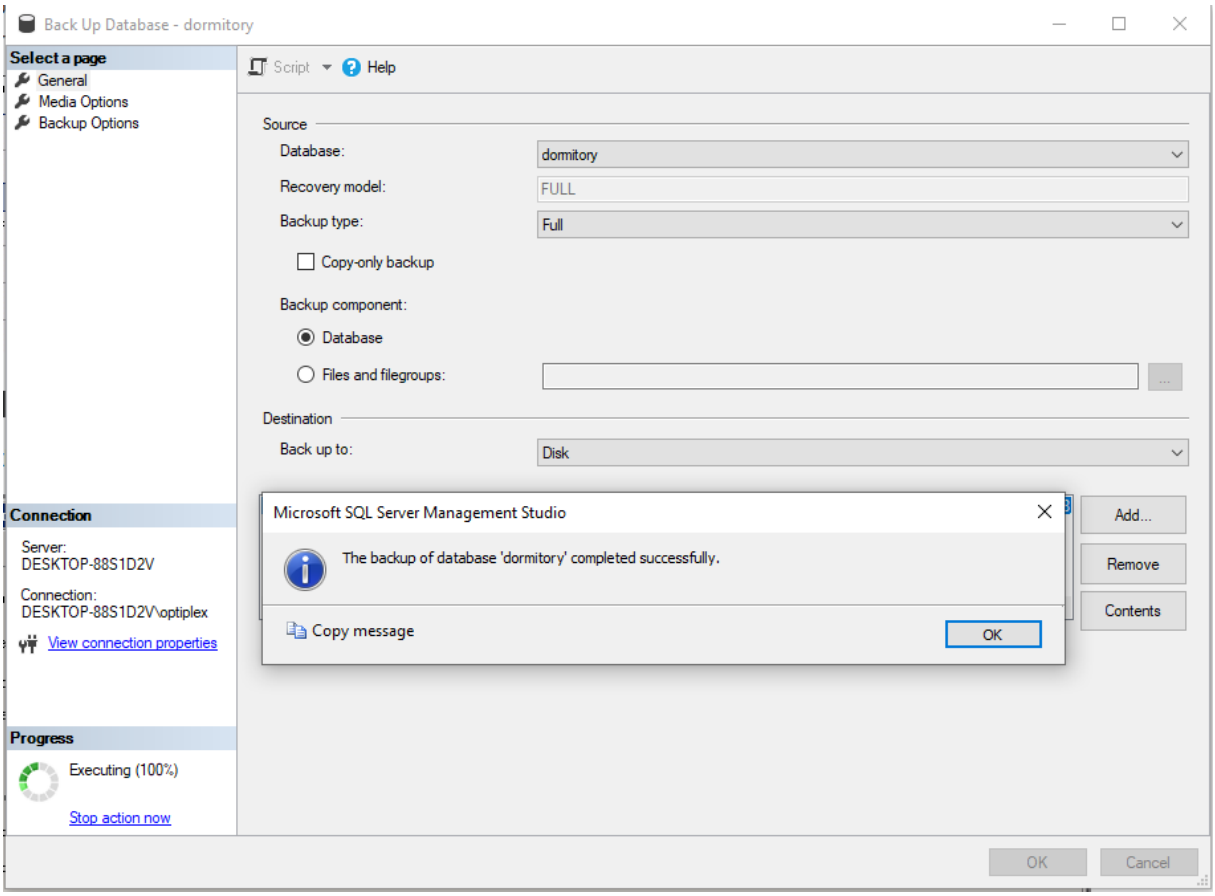
	id [PK] integer	first_name character varying (25)	second_name character varying (25)	last_name character varying (50)	pesel character (11)	address_id integer
1	1465	Arnold	[null]	Eglise	00210171464	1522
2	1531	Johna	[null]	Mohan	00210232514	1588
3	1613	Maxie	[null]	Bauser	00210249361	1670
4	1124	Cinderella	[null]	Coie	00210284269	1181
5	1356	Alanna	[null]	D'eye	00210359934	1413
6	1248	Deni	[null]	Delacourt	00210426359	1305
7	1164	Dix	[null]	Le Provest	00210522239	1221
8	1104	Aleda	[null]	Fihelly	00210721591	1161
9	1334	Estrellita	[null]	Lackemann	00210727733	1391
10	1181	Carly	[null]	Brosius	00210739318	1238
11	1289	Felita	[null]	Scrivens	00210832837	1346
12	1230	Filmore	[null]	Gley	00210967999	1287
13	1165	Netta	[null]	Clubley	00211053844	1222
14	1633	Ode	[null]	Davidof	00211148294	1690
15	1496	Beverley	[null]	De Carolis	00211166243	1553

3. Kopia zapasowa bazy danych

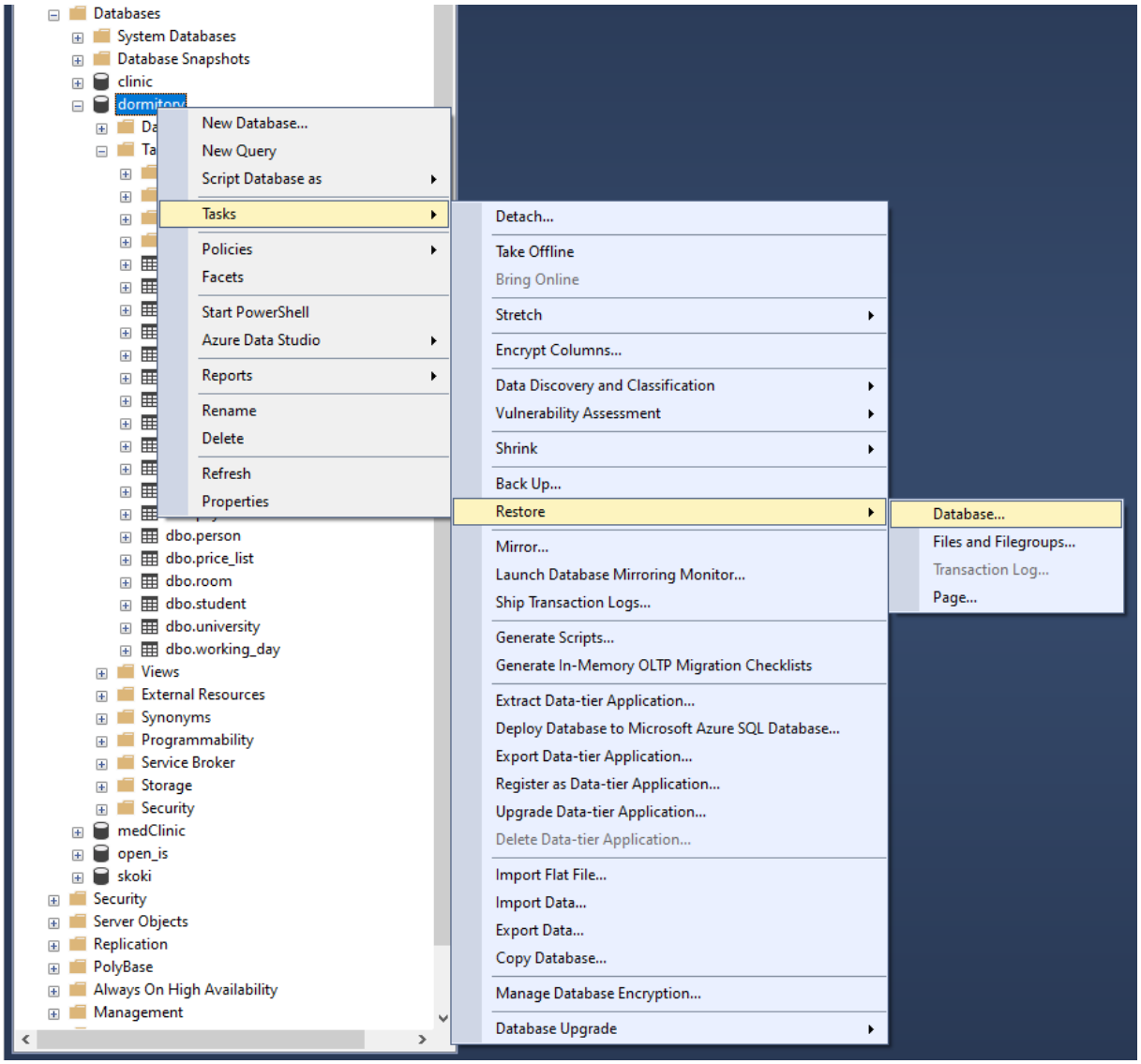
Tworzenie kopii zapasowej bazy danych w narzędziu Microsoft SQL Server Management Studio jest bardzo proste. Należy wybrać opcję *Back Up*, jak wskazano poniżej:



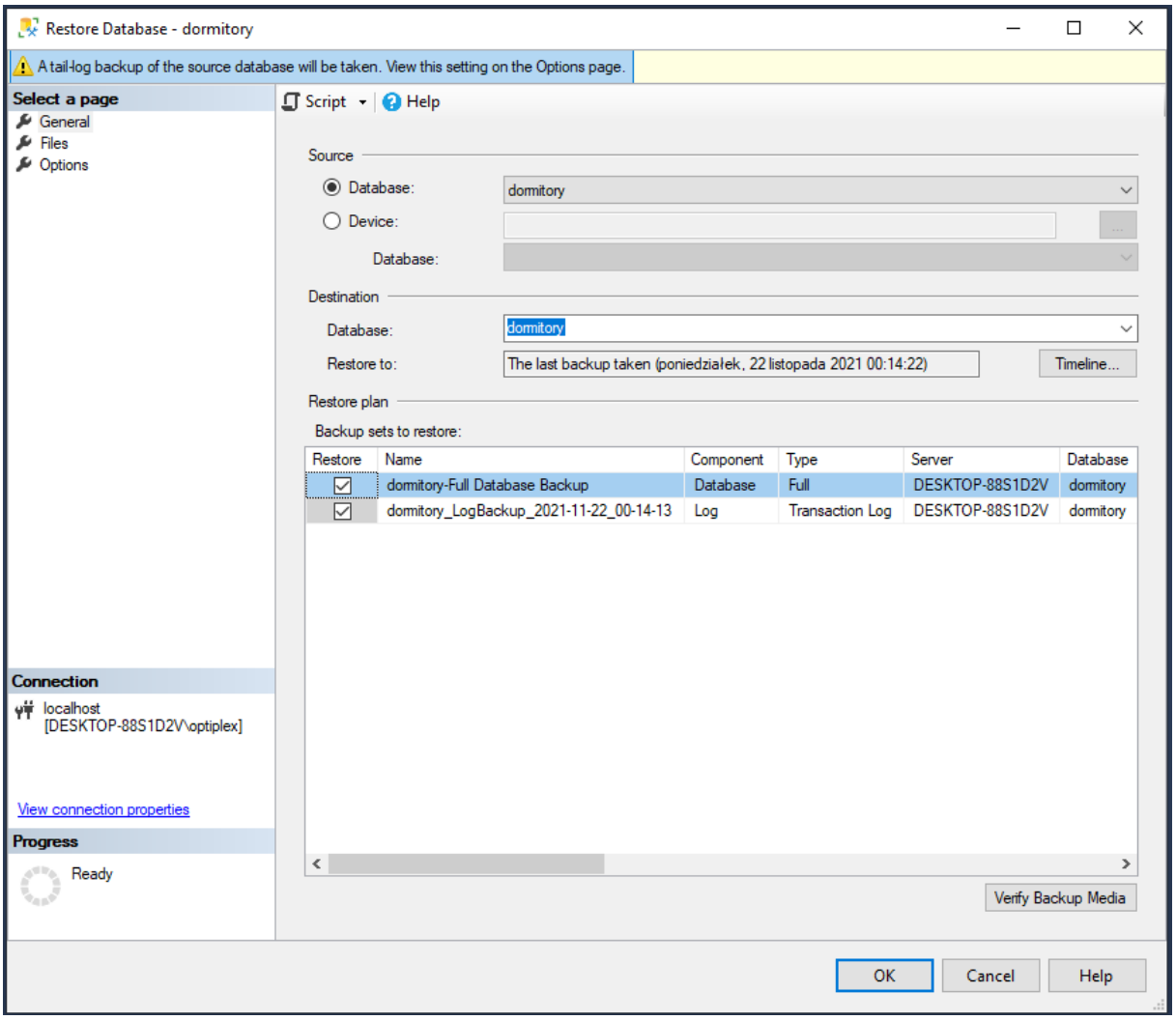
Następnie istnieje możliwość wyboru typu kopii zapasowej oraz lokalizacji tworzonego pliku. W przypadku pomyślnego procesu, zostanie wyświetlony następujący komunikat:



Aby odtworzyć kopię zapasową, należy skorzystać z opcji *Restore*, jak ukazano poniżej:



W otwartym oknie należy wybrać odpowiedni plik przechowujący kopię zapasową oraz nacisnąć przycisk *OK*.



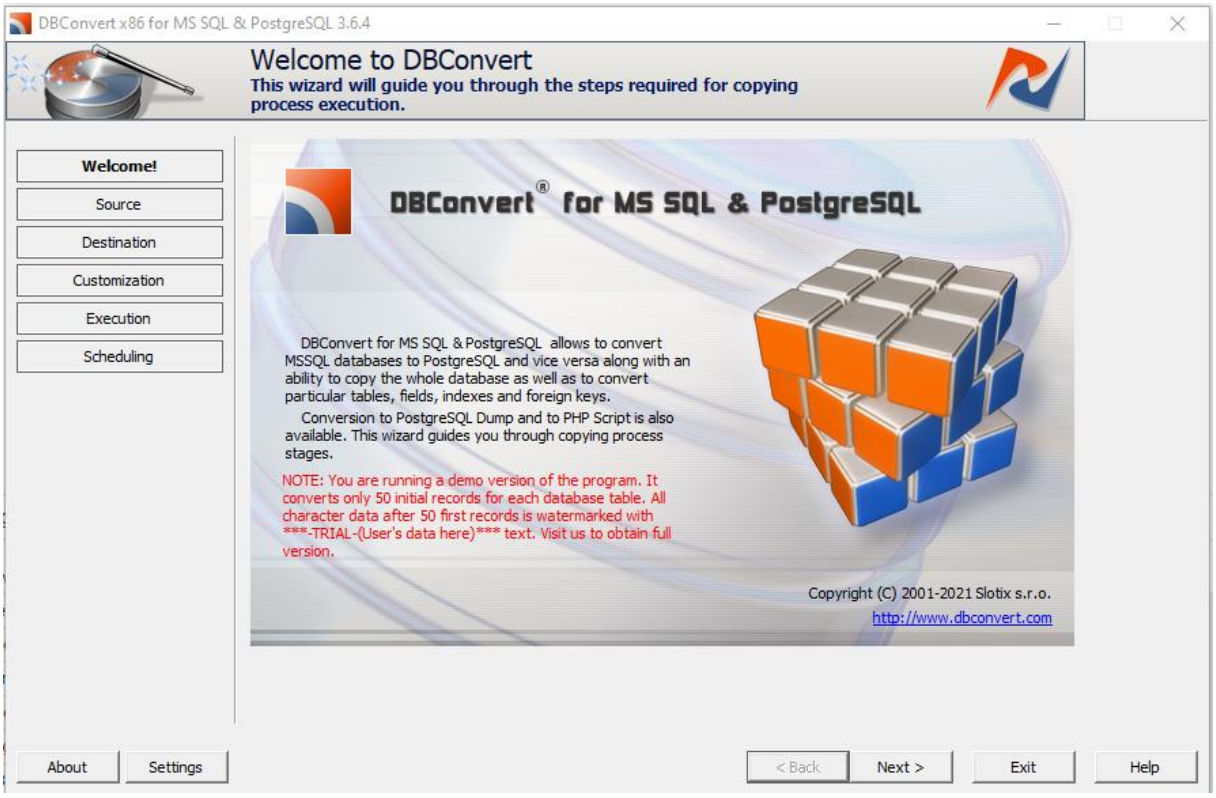
W rezultacie w bardzo prosty sposób odtworzono kopię zapasową bazy danych.

4. Migracja bazy danych

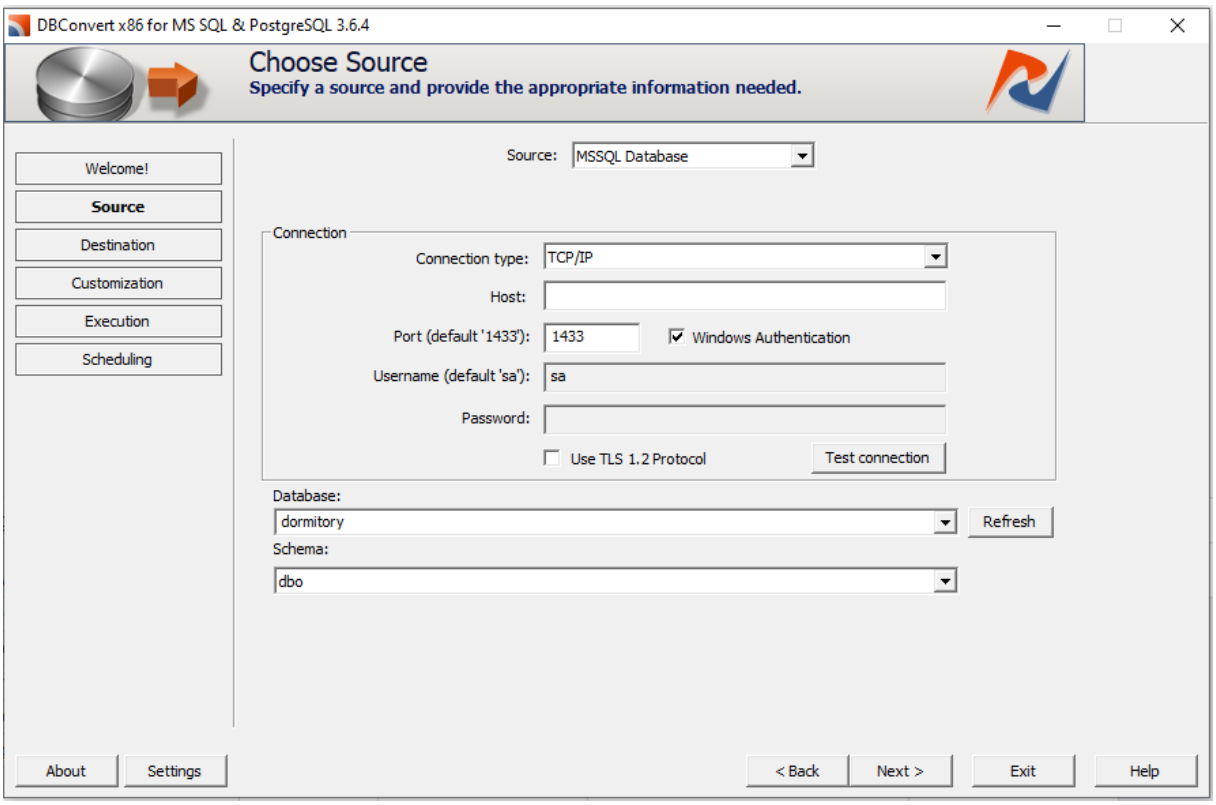
Ze względu na wymienione w poprzednim punkcie różnice składniowe między systemami SQL Server oraz PostgreSQL, migracja bazy danych może okazać się nietrywialnym zadaniem. Dysponując gotowymi skryptami oraz znając składnię innego dialektu, można przepisać gotowe rozwiązania, tak jak przedstawiono to w poprzednim punkcie. Jest to jednak zadanie pracochłonne, dlatego warto wypróbować gotowe narzędzia do migracji baz danych, które mogą częściowo uprościć proces przenoszenia zasobów.

4.1. DBConvert for MS SQL & PostgreSQL

Pierwszym narzędziem, które zostanie przetestowane do migracji bazy danych będzie *DBConvert for MS SQL & PostgreSQL*. Pozwala ono na przeniesienie bazy za pomocą interfejsu graficznego. Niestety, jest to narzędzie płatne, które pozwala jedynie na demonstracyjną migrację w wersji próbnej. Poniżej przedstawiono okno główne aplikacji:

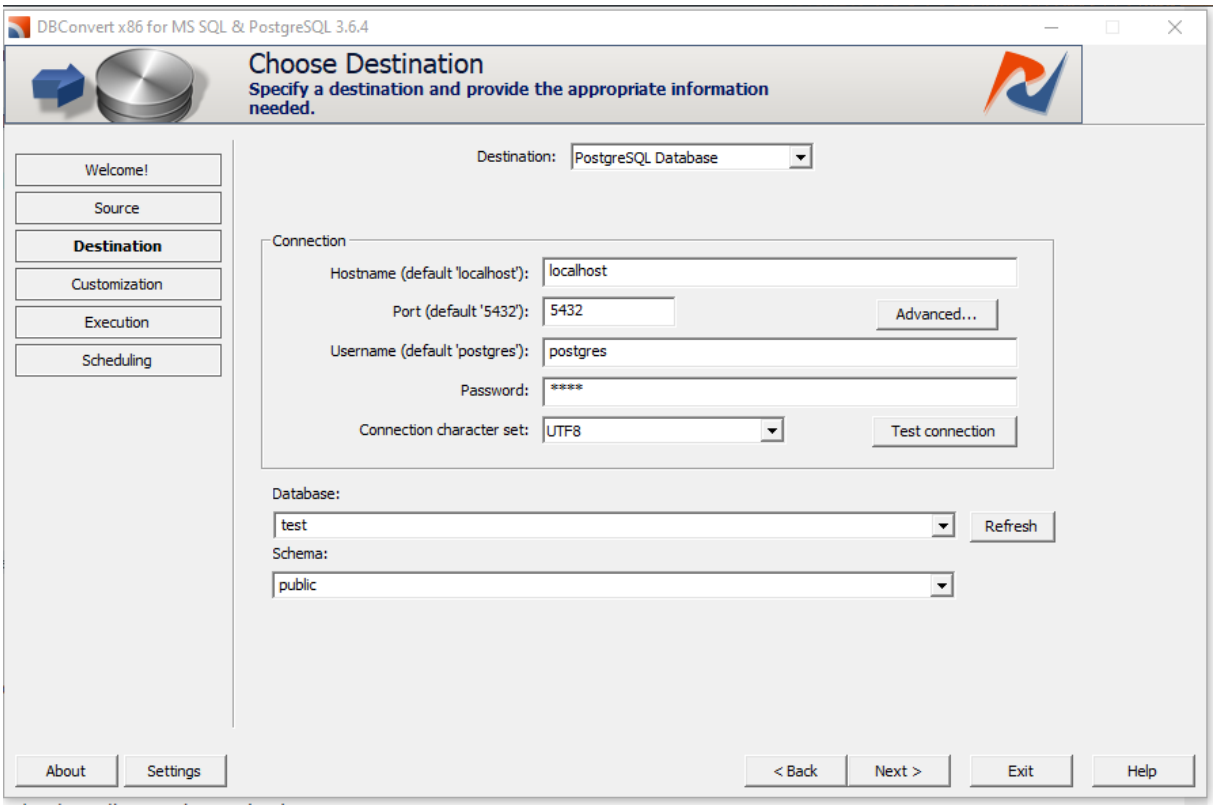


Krokiem pierwszym jest połączenie z bazą źródłową – w tym przypadku będzie to baza MSSQL.

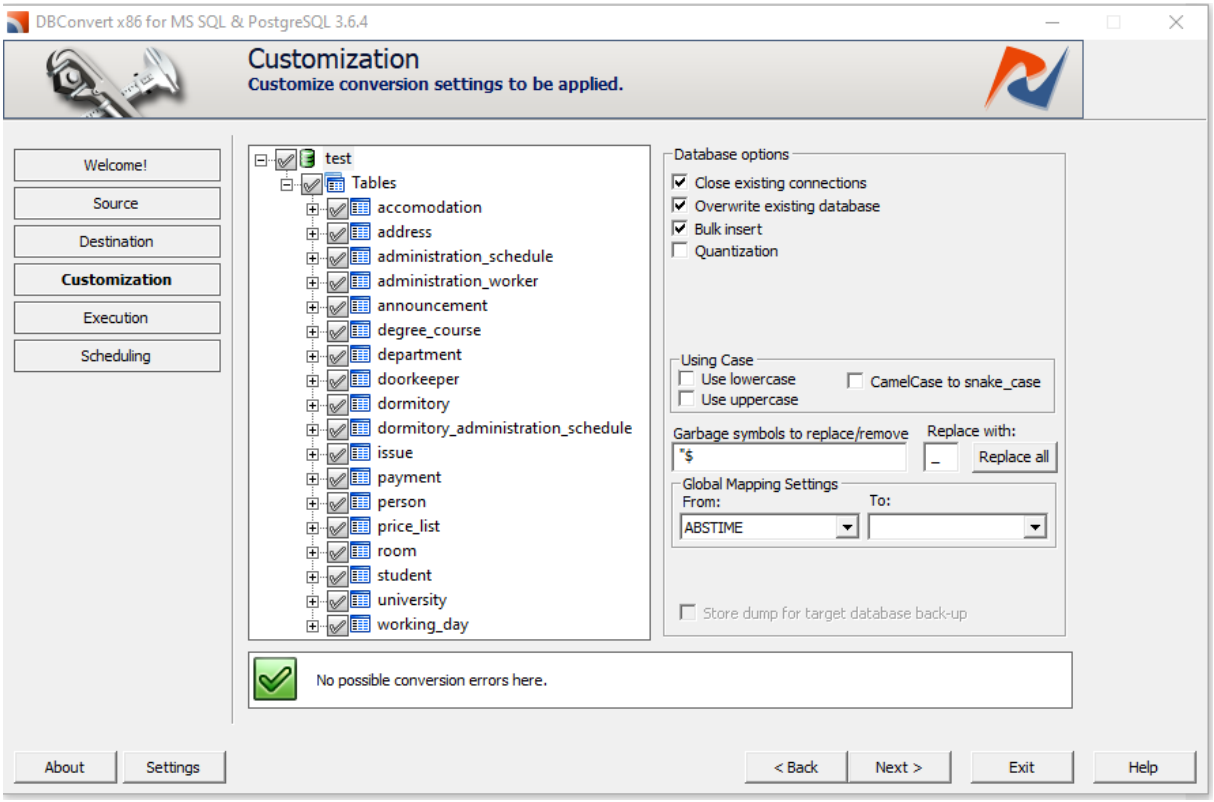


Za pomocą przycisku *Test connection* można zweryfikować, czy udało się uzyskać prawidłowe połączenie.

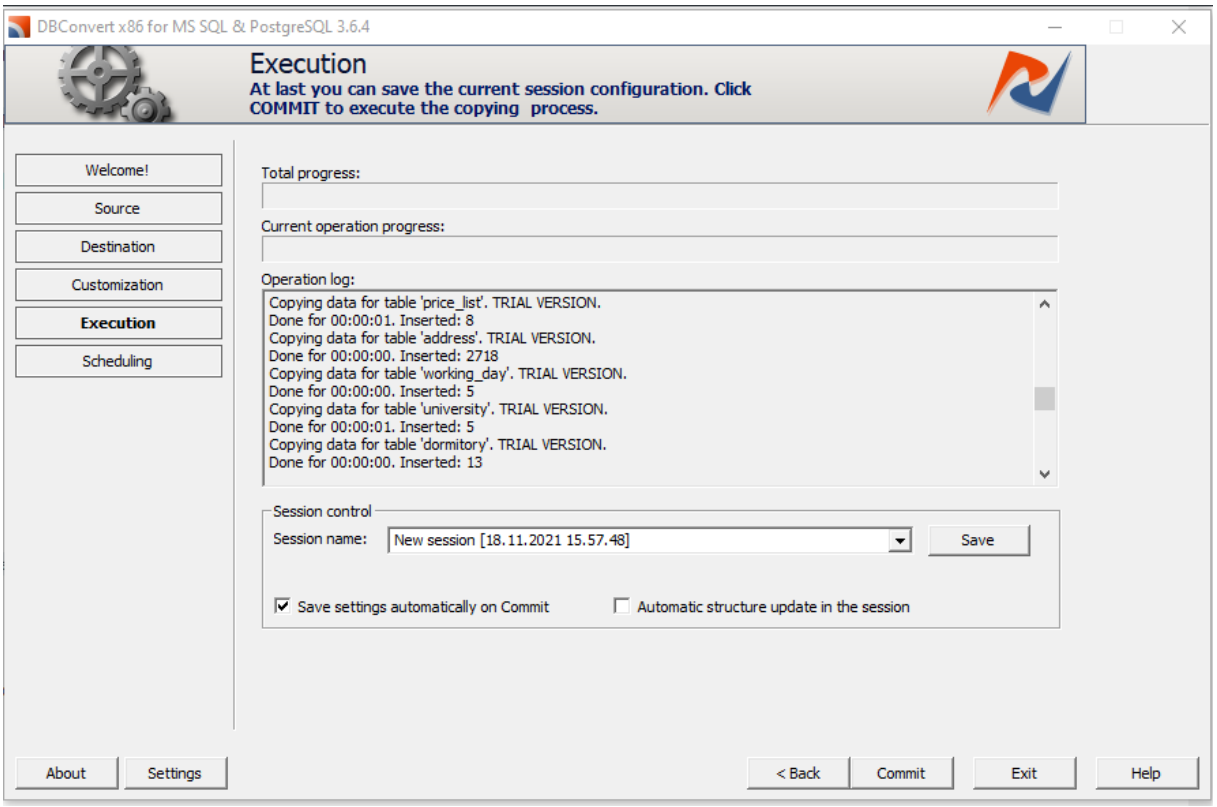
W kroku drugim należy powtórzyć tę samą czynność, tym razem dla bazy PostgreSQL.



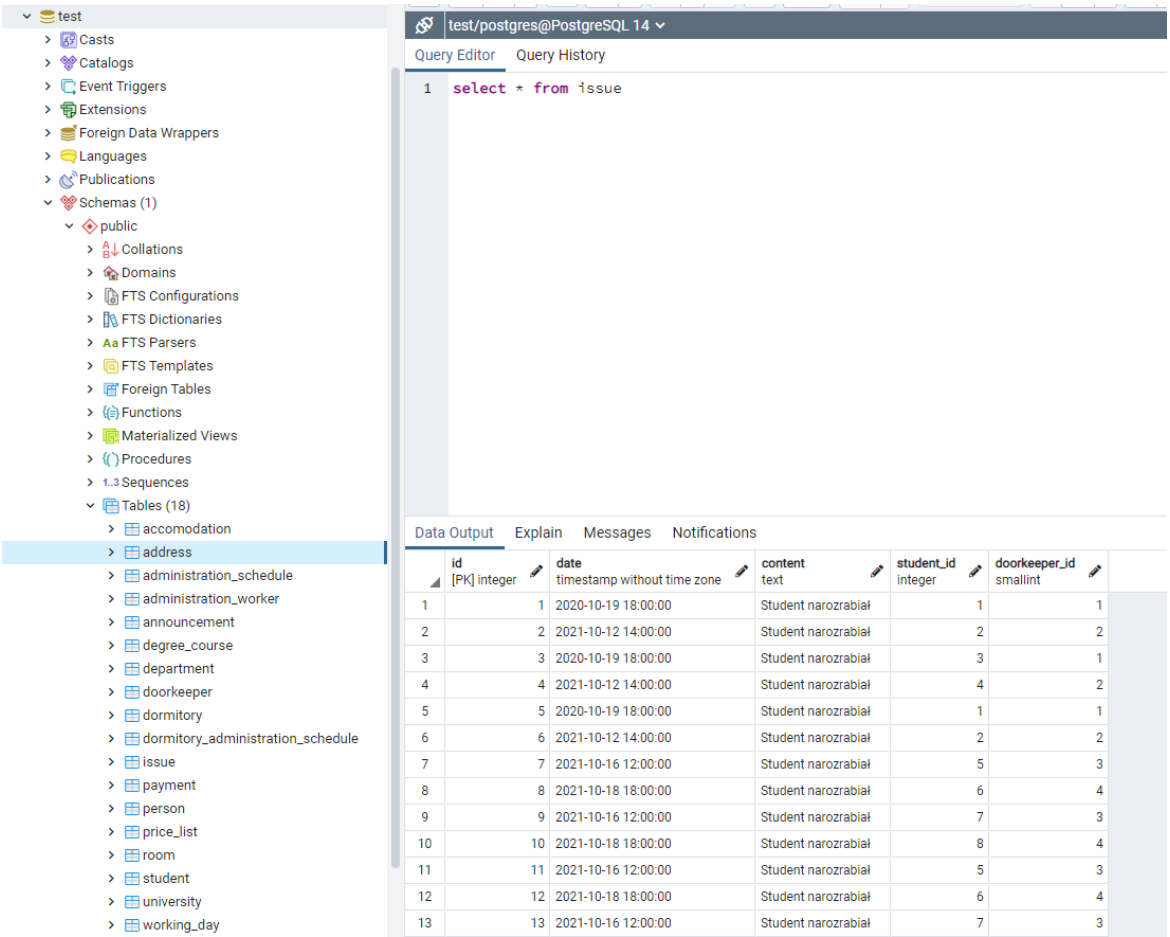
Krok trzeci pozwala na wybór tabel, które zostaną przeniesione oraz zawiera kilka dodatkowych opcji pozwalających na nadpisanie istniejącej bazy, zmianę sposobu nazewnictwa (np. camelCase to snake_case) itp.



W kolejnym kroku wystarczy nacisnąć przycisk *Commit*, aby rozpocząć migrację bazy. Podczas przenoszenia widoczny jest log, który wyświetli ewentualne błędy. Istnieje możliwość zapisu logu do pliku.



W przypadku pomyślnej migracji, baza danych zostanie utworzona i wypełniona danymi.



Jak można zauważyć, narzędzie automatycznie dopasowało typy danych do nowej składni oraz prawidłowo przeniosło dane. Niestety, w darmowej wersji nie ma możliwości przenoszenia innych elementów niż tabel. Dodatkowo, wiele danych zostało zmodyfikowanych i otrzymało dopisek *TRIAL.*

	id [PK] integer	street character varying (70)	flat_number smallint	postcode character varying (10)	city character varying (50)	country character varying (50)
49	49	Ap #591-7779 Vestibulum Rd.	[null]	28-237	Wałbrzych	Poland
50	50	2314 Dolor Rd.	[null]	65-340	Tarnów	Poland
51	51	3316 Leo, Av.	[null]	79-388	Bydgoszcz	Poland
52	52	8198 Malesuada Street	[null]	41-TRIAL-	34-TRIAL-Ostrowiec Świętokrzyski 100	269-TRIAL-Poland 124
53	53	756-9989 Dui, Ave	[null]	78-TRIAL-	262-TRIAL-Szczecin 164	5-TRIAL-Poland 245
54	54	629-5817 At Street	[null]	181-TRIAL-	61-TRIAL-Elbląg 191	295-TRIAL-Poland 242
55	55	Ap #138-2597 Ullamcorper Ave	[null]	27-TRIAL-	291-TRIAL-Lublin 204	2-TRIAL-Poland 153
56	56	218-6979 Quisque Avenue	[null]	292-TRIAL-	21-TRIAL-Stargard Szczeciński 116	218-TRIAL-Poland 95
57	57	454-2107 Phasellus Avenue	[null]	47-TRIAL-	71-TRIAL-Szczecin 138	69-TRIAL-Poland 112
58	58	Ap #647-237 Montes, Rd.	[null]	167-TRIAL-	235-TRIAL-Kraków 294	203-TRIAL-Poland 111
59	59	Ap #924-1558 Condimentum. Avenue	[null]	122-TRIAL-	273-TRIAL-Gliwice 164	141-TRIAL-Poland 211
60	60	Ap #719-734 Vel Rd.	[null]	53-TRIAL-	47-TRIAL-Rzeszów 44	262-TRIAL-Poland 57
61	61	1887 Velit Rd.	[null]	237-TRIAL-	23-TRIAL-Białystok 141	229-TRIAL-Poland 178
62	62	179-6880 Luctus Av.	[null]	16-TRIAL-	290-TRIAL-Gliwice 42	288-TRIAL-Poland 106
63	63	Ap #604-2674 Porttitor Avenue	[null]	40-TRIAL-	64-TRIAL-Kalisz 148	146-TRIAL-Poland 105

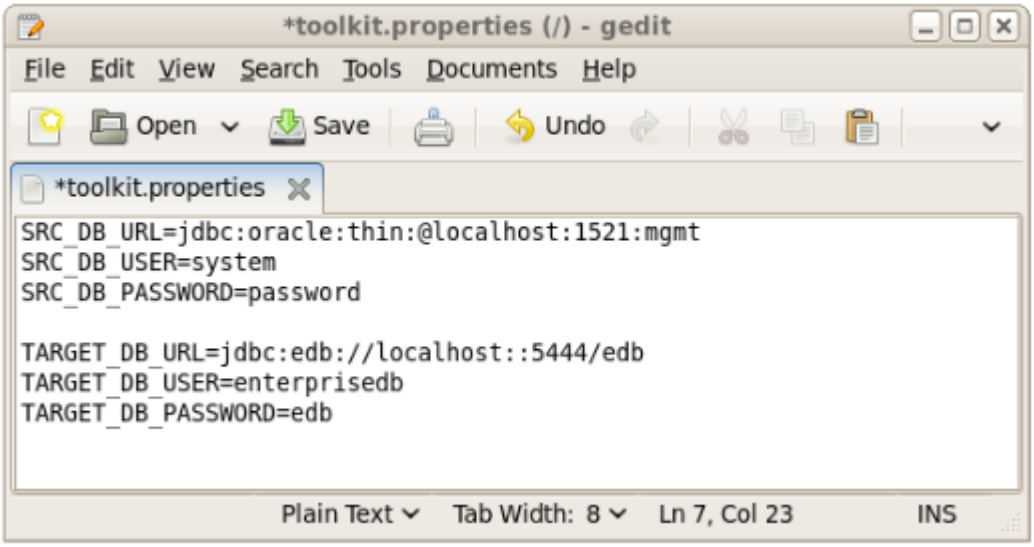
Subiektywna opinia:

Narzędzie *DBConvert* pozwala na przeniesienie bazy danych za pomocą kilku kliknięć – jest bardzo proste i wygodne w obsłudze. Niestety, konsekwencją wygody są koszty związane z korzystaniem z oprogramowania.

4.2. EDB Migration Toolkit

Aktualne wersje oprogramowania EDB Migration Toolkit są prawdopodobnie płatne, jednak istnieje możliwość znalezienia w internecie starych, darmowych wersji tegoż oprogramowania. Za pomocą wspomnianego narzędzia istnieje możliwość migracji bazy danych z wykorzystaniem wiersza poleceń.

Pierwszym krokiem po instalacji narzędzia jest modyfikacja pliku *toolkit.properties*. W tym momencie należy uzupełnić dane dotyczące źródłowej oraz docelowej bazy danych:



Kolejnym krokiem jest stworzenie pustej bazy danych w docelowym systemie bazodanowym – w tym przypadku stworzona została pusta baza o nazwie *test*. Następnie należy przenieść się do odpowiedniego folderu oraz w wierszu poleceń wprowadzić odpowiednią komendę:

C:\Program Files\edb\mtk\bin>.\runMTK.bat -sourcedbtype sqlserver -targetdbtype postgresql -targetSchema public -allTables dbo -allViews dbo

Powyższe polecenie zleca przeniesienie wszystkich tabel oraz widoków z systemu SQL Server do bazy danych w systemie PostgreSQL. Poniżej przedstawiono rezultat:

```
C:\Program Files\edb\mtk\bin>.\runMTK.bat -sourcedbtype sqlserver -targetdbtype postgresql -targetSchema public -allTables dbo -allViews dbo
Running EnterpriseDB Migration Toolkit (Build 55.1.0) ...
Source database connectivity info...
conn =jdbc:jtds:sqlserver://localhost:1433/dormitory
user =root
password=******
Target database connectivity info...
conn =jdbc:postgresql://localhost:5432/test
user =postgres
password=******
Connecting with source SQL Server database server...
Connected to Microsoft SQL Server, version '15.00.2000'
Connecting with target Postgres database server...
Connected to PostgreSQL, version '14.0'
Importing sql server schema dbo...
Creating Tables...
Creating Table: address
Creating Table: university
Creating Table: working_day
Creating Table: administration_schedule
Creating Table: price_list
Creating Table: dormitory
Creating Table: dormitory_administration_schedule
Creating Table: department
Creating Table: degree_course
Creating Table: person
Creating Table: room
Creating Table: student
Creating Table: payment
Creating Table: doorkeeper
Creating Table: accomodation
Creating Table: issue
Creating Table: administration_worker
Creating Table: announcement
Created 18 tables.
Initializing Data Loader 1...
Loading Table Data in 8 MB batches...
Loading Table: public.address ...
[address] Migrated 2718 rows.
[address] Table Data Load Summary: Total Time(s): 0.192 Total Rows: 2718 Total Size(KB): 146,289
Loading Table: public.university ...
[university] Migrated 5 rows.
[university] Table Data Load Summary: Total Time(s): 0.019 Total Rows: 5 Total Size(KB): 0,175
Loading Table: public.working_day ...
[working_day] Migrated 5 rows.
[working_day] Table Data Load Summary: Total Time(s): 0.017 Total Rows: 5 Total Size(KB): 0,055
Loading Table: public.administration_schedule ...
[administration_schedule] Migrated 15 rows.
[administration_schedule] Table Data Load Summary: Total Time(s): 0.023 Total Rows: 15 Total Size(KB): 0,563
Loading Table: public.price_list ...
[price_list] Migrated 8 rows.
[price_list] Table Data Load Summary: Total Time(s): 0.011 Total Rows: 8 Total Size(KB): 0,29
Loading Table: public.dormitory ...
[dormitory] Migrated 13 rows.
[dormitory] Table Data Load Summary: Total Time(s): 0.016 Total Rows: 13 Total Size(KB): 0,52
```

```
Loading Table: public.person ...
[person] Migrated 2700 rows.
[person] Table Data Load Summary: Total Time(s): 0.111 Total Rows: 2700 Total Size(KB): 103,129
Loading Table: public.room ...
[room] Migrated 2600 rows.
[room] Table Data Load Summary: Total Time(s): 0.057 Total Rows: 2600 Total Size(KB): 32,709
Loading Table: public.student ...
[student] Migrated 2600 rows.
[student] Table Data Load Summary: Total Time(s): 0.048 Total Rows: 2600 Total Size(KB): 46,036
Loading Table: public.payment ...
[payment] Migrated 2601 rows.
[payment] Table Data Load Summary: Total Time(s): 0.122 Total Rows: 2601 Total Size(KB): 96,899
Loading Table: public.doorkeeper ...
[doorkeeper] Migrated 26 rows.
[doorkeeper] Table Data Load Summary: Total Time(s): 0.016 Total Rows: 26 Total Size(KB): 0,254
Loading Table: public.accomodation ...
[accomodation] Migrated 8 rows.
[accomodation] Table Data Load Summary: Total Time(s): 0.008 Total Rows: 8 Total Size(KB): 0,415
Loading Table: public.issue ...
[issue] Migrated 46 rows.
[issue] Table Data Load Summary: Total Time(s): 0.041 Total Rows: 46 Total Size(KB): 2,289
Loading Table: public.administration_worker ...
[administration_worker] Migrated 13 rows.
[administration_worker] Table Data Load Summary: Total Time(s): 0.01 Total Rows: 13 Total Size(KB): 0,562
Loading Table: public.announcement ...
[announcement] Migrated 26 rows.
[announcement] Table Data Load Summary: Total Time(s): 0.013 Total Rows: 26 Total Size(KB): 1,752
Data Load Summary: Total Time (sec): 0,743 Total Rows: 13477 Total Size(KB): 433,113
Creating View: dormitories
MTK-15008: Error Creating View: dormitories
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'dormitory_name'"
  Pozycja: 49
Creating View: rooms
MTK-15008: Error Creating View: rooms
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'dormitory'"
  Pozycja: 74
Creating View: students
MTK-15008: Error Creating View: students
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'first_name'"
  Pozycja: 65
Creating View: doorkeepers
MTK-15008: Error Creating View: doorkeepers
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'first_name'"
  Pozycja: 69
Creating View: administration_workers
MTK-15008: Error Creating View: administration_workers
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'dormitory'"
  Pozycja: 118
Creating View: issues
MTK-15008: Error Creating View: issues
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "'student_first_name'"
  Pozycja: 70
Creating View: announcements
MTK-15008: Error Creating View: announcements
DB-42601: org.postgresql.util.PSQLException: ERROR: syntax error at or near "["
  Pozycja: 41
Creating View: payments
MTK-15008: Error Creating View: payments
DB-42P01: org.postgresql.util.PSQLException: ERROR: relation "students" does not exist
  Pozycja: 83

Schema dbo imported with errors.
```

Jak można zauważyć, tabele zostały stworzone pomyślnie oraz wypełniono je danymi. Niestety, z powodu różnic składniowych, nie udało się utworzyć widoków. W tym przypadku konieczna jest modyfikacja skryptu generującego widoki w źródłowej bazie danych.

```sql
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [id]
      ,[street]
      ,[flat_number]
      ,[postcode]
      ,[city]
      ,[country]
  FROM [dormitory].[dbo].[address]
```

100 %

Results | Messages

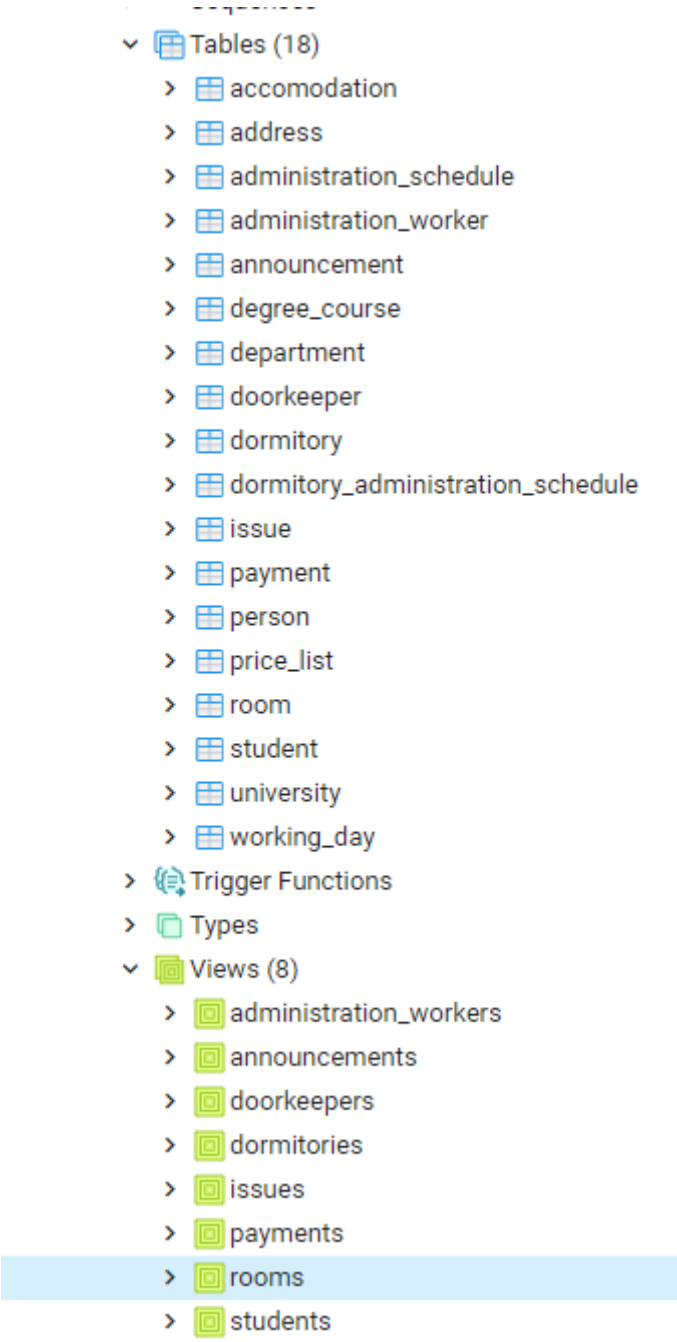
	id	street	flat_number	postcode	city	country
1	1	Plac Politechniki 1	NULL	00-661	Warszawa	Poland
2	2	Krakowskie Przedmieście 26/28	NULL	00-927	Warszawa	Poland
3	3	Nowoursynowska 166	NULL	02-787	Warszawa	Poland
4	4	Al.Niepodległości 162	NULL	02-554	Warszawa	Poland
5	5	Żwirki i Wigury 61	NULL	02-091	Warszawa	Poland
6	6	Grójecka 39	NULL	02-031	Warszawa	Poland
7	7	Księcia Janusza 39	NULL	01-452	Warszawa	Poland
8	8	Ludwika Waryńskiego 10	NULL	00-631	Warszawa	Poland
9	9	Akademicka 5	NULL	02-038	Warszawa	Poland
10	10	Karolkowa 84	NULL	01-193	Warszawa	Poland
11	11	Batalionu AK Pięść 9	NULL	01-406	Warszawa	Poland
12	12	Radomska 11	NULL	05-077	Warszawa	Poland
13	13	Żwirki i Wigury 97/99	NULL	02-089	Warszawa	Poland
14	14	Al.Niepodległości 147	NULL	02-555	Warszawa	Poland
15	15	Antoniego Józefa Madalińskiego 31/33	NULL	02-544	Warszawa	Poland
16	16	Nowoursynowska 161E	NULL	02-787	Warszawa	Poland
17	17	Zygmunta Vogla 44	NULL	02-890	Warszawa	Poland

Ponieważ w PostgreSQL nie można nazywać kolumn z wykorzystaniem pojedynczych apostrofów (‘), wszystkie pojedyncze apostrofy zostały zastąpione znakiem (”). Następnie ponownie wygenerowano skrypt odpowiadający za migrację danych.

```
Creating View: dormitories
Creating View: rooms
Creating View: students
Creating View: doorkeepers
Creating View: administration_workers
Creating View: issues
Creating View: announcements
Creating View: payments

Schema dbo imported successfully.


Migration process completed successfully.
```



```
1  select * from payments
```

Data Output Explain Messages Notifications

	date timestamp without time zone	amount numeric (6,2)	id integer	First name character varying (25)	Last name character varying (50)	PESEL character (11)	Street character varying (70)	Flat number smallint	ZIP character varying (10)	Country character varying (50)
1	2021-10-12 00:00:00	380.00	27	Rosette	Bentley	95091961976	461-3731 Leo, Road	[null]	73-278	Poland
2	2021-10-10 00:00:00	380.00	28	Fabien	Pigden	95030429965	Ap #759-2681 Sapien, Road	[null]	15-370	Poland
3	2021-10-08 00:00:00	380.00	29	Jarad	Baudet	95083016338	P.O. Box 635, 4311 Mauris, Avenue	[null]	81-981	Poland
4	2021-10-04 00:00:00	380.00	30	Stefano	Hofner	95060848756	245-4289 Tellus St.	[null]	65-837	Poland
5	2021-10-08 00:00:00	380.00	31	Gusti	MacCaughan	95032923214	Ap #293-3765 Est, St.	[null]	54-581	Poland
6	2021-10-01 00:00:00	380.00	32	Kaycee	Thyng	95071668949	830-6215 Sem Road	[null]	98-233	Poland
7	2021-10-13 00:00:00	380.00	33	Sergeant	Handrek	95033035246	927-1812 Sapien. Av.	[null]	52-088	Poland
8	2021-10-11 00:00:00	380.00	34	Earvin	Alfwy	95121461281	227-3037 Enim Av.	[null]	12-804	Poland
9	2021-10-09 00:00:00	380.00	35	Lauretta	Pipworth	95110752525	Ap #537-5491 Purus. Avenue	[null]	41-302	Poland
10	2021-10-10 00:00:00	380.00	36	Sheffield	Smewing	95112237664	910 Quis Rd.	[null]	19-516	Poland
11	2021-10-12 00:00:00	380.00	37	Austina	Millington	95121758316	8336 Consectetuer St.	[null]	62-932	Poland
12	2021-10-07 00:00:00	380.00	38	Brander	Hunnam	95072534122	6286 Natoque Street	[null]	24-986	Poland
13	2021-10-13 00:00:00	380.00	39	Rayshell	Roobottom	95121153777	P.O. Box 355, 720 Turpis St.	[null]	83-366	Poland
14	2021-10-01 00:00:00	380.00	40	Yolanthe	Gannaway	95081743328	992-1726 Donec Rd.	[null]	03-839	Poland

W bazie danych *test* utworzone zostały wszystkie tabele, które wypełniono danymi oraz perspektywy. Jak wcześniej wspomniano, wyzwalacze, procedury oraz perspektywy należy utworzyć ręcznie.

Subiektywna opinia:

Narzędzie EDB Migration Toolkit nie jest najprostsze w obsłudze – wynika to głównie z braku interfejsu graficznego oraz konieczności posługiwania się wierszem poleceń. Wymaga zainstalowanej wersji JDK 8 oraz ustawionych odpowiednich ścieżek systemowych. W moim przypadku konieczna okazała się również modyfikacja pliku *runMTK.bat*, ponieważ pewne biblioteki nie były wykrywane przez program. Ostatecznie udało się jednak przenieść wszystkie dane bez poniesienia żadnych kosztów, co uważam za sukces.

5. Demonstracja działania funkcji, procedur oraz wyzwalaczy

```
SELECT * FROM StudentDataByPesel('95050859964');
```

Data Output Explain Messages Notifications

	id integer	First name character varying (25)	Last name character varying (50)	PESEL character (11)	Street character varying (70)	Flat number smallint	ZIP character varying (10)	City character varying (50)	Country character varying (50)	Term smallint	Degree course character vary
1	25	Arlin	Ricciardiello	95050859964	791-4770 Ultrices. Rd.	[null]	72-111	Biała Podlaska	Poland	3	Elektronika

```
SELECT * FROM NonFullRooms('Bratniak')
```

Data Output Explain Messages Notifications

	id integer	number character varying	capacity smallint	dormitory character varying	locators bigint	available spots bigint
1	2	101	2	Dom Studencki Bratniak-Muszelka	1	1
2	3	102	3	Dom Studencki Bratniak-Muszelka	1	2
3	4	103	4	Dom Studencki Bratniak-Muszelka	1	3
4	5	104	5	Dom Studencki Bratniak-Muszelka	1	4
5	7	106	2	Dom Studencki Bratniak-Muszelka	1	1
6	8	107	3	Dom Studencki Bratniak-Muszelka	1	2
7	9	108	4	Dom Studencki Bratniak-Muszelka	1	3
8	10	109	5	Dom Studencki Bratniak-Muszelka	1	4
9	12	111	2	Dom Studencki Bratniak-Muszelka	1	1
10	13	112	3	Dom Studencki Bratniak-Muszelka	1	2
11	14	113	4	Dom Studencki Bratniak-Muszelka	1	3
12	15	114	5	Dom Studencki Bratniak-Muszelka	1	4
13	17	116	2	Dom Studencki Bratniak-Muszelka	1	1
14	18	117	3	Dom Studencki Bratniak-Muszelka	1	2
15	19	118	4	Dom Studencki Bratniak-Muszelka	1	3

```
CALL addNewStudent(
p_firstName => 'Adam',
p_lastName =>'Adamiak',
p_pesel => '99634502199',
p_street => 'Kwiatowa',
p_flat => 13::smallint,
p_term => 5::smallint,
p_zip => '02-031',
p_city => 'Radom',
p_country => 'Polska',
p_university => 'Politechnika Warszawska',
p_department => 'Wydzialik Elektryczny',
p_degreeCourse => 'Informatyka',
p_room => '264',
p_dormitory =>'Dom Studencki Bratniak-Muszelka'
);
```

Data Output Explain Messages Notifications

```
ERROR:  Invalid department name
```

```
CALL addNewStudent(
p_firstName => 'Adam',
p_lastName =>'Adamiak',
p_pesel => '98634502199',
p_street => 'Kwiatowa',
p_flat => 13::smallint,
p_term => 5::smallint,
p_zip => '02-031',
p_city => 'Radom',
p_country => 'Polska',
p_university => 'Politechnika Warszawska',
p_department => 'Wydział Elektryczny',
p_degreeCourse => 'Informatyka',
p_room => '190',
p_dormitory =>'Dom Studencki Bratniak-Muszelka'
);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  This room is already full
```

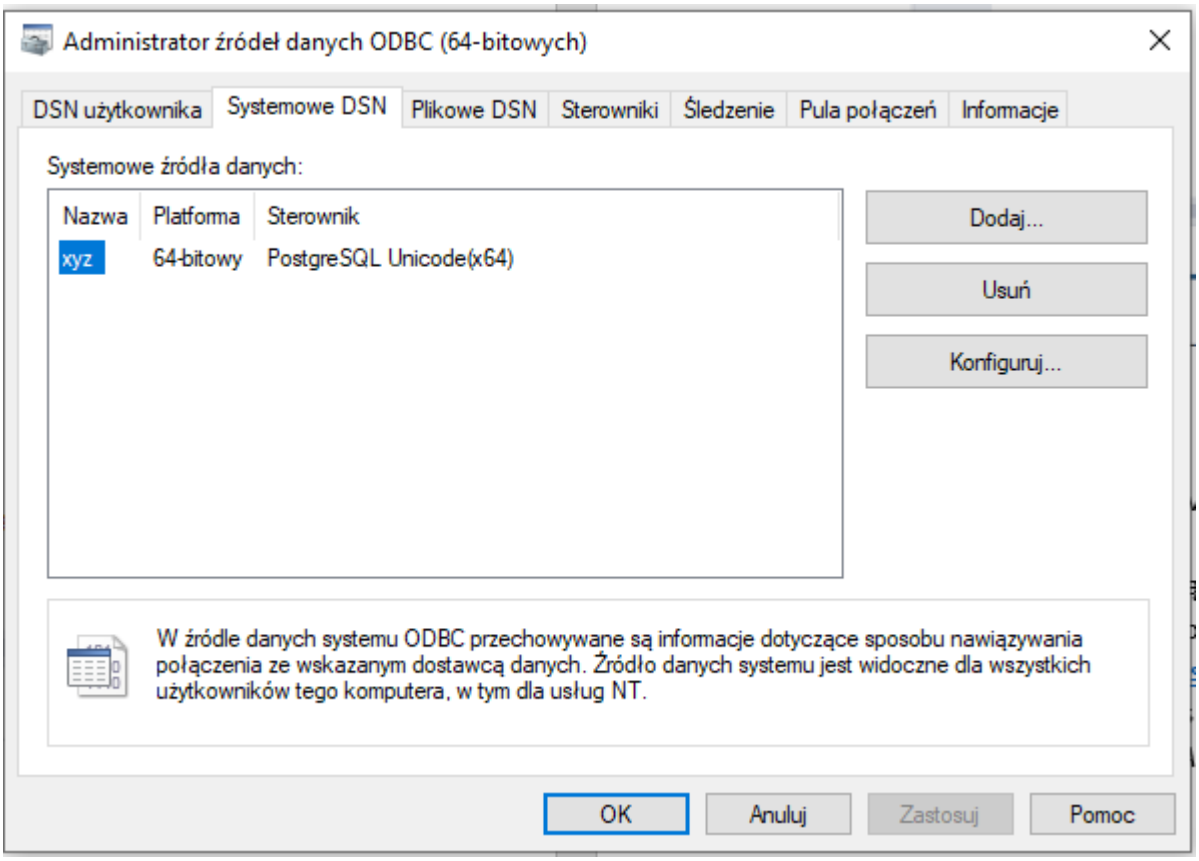
```
INSERT INTO student(term, degree_course_id, person_id, room_id) VALUES (7, 1, 2688, 1)
```

Data Output | Explain | Messages | Notifications

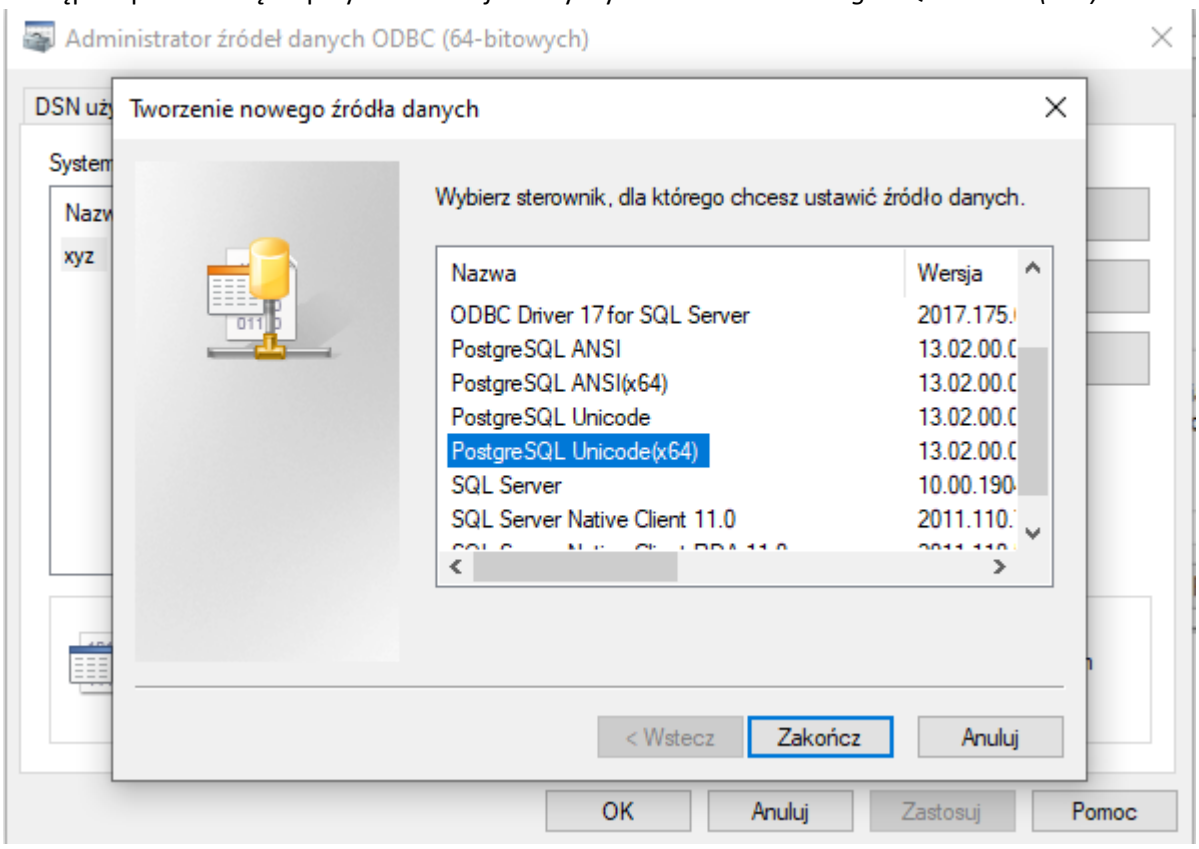
```
ERROR:  This person is already an administration worker
CONTEXT:  funkcja PL/pgSQL validateifpersoncanbestudent(), wiersz 4 w RAISE
SQL state: P0001
```

6. Połączenie bazodanowe (database link)

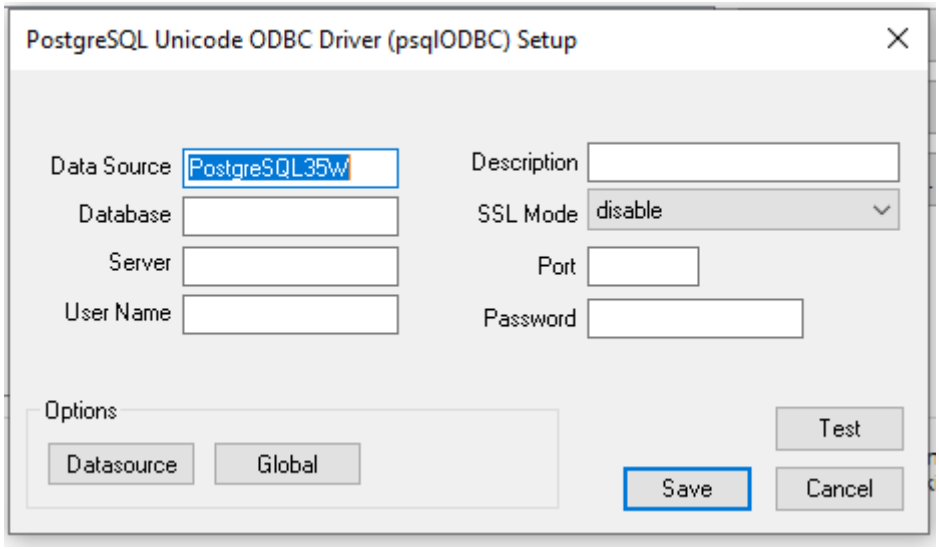
W celu uzyskania połączenia bazodanowego między systemami wykorzystano PostgreSQL ODBC Driver, który pobrać można ze strony https://www.postgresql.org/ftp/odbc/versions/msi/. Po pobraniu i instalacji sterownika należy odpowiednio skonfigurować źródła danych ODBC. W systemie Windows należy uruchomić program *Administrator źródeł danych ODBC*, a następnie wybrać zakładkę *Systemowe DSN*.



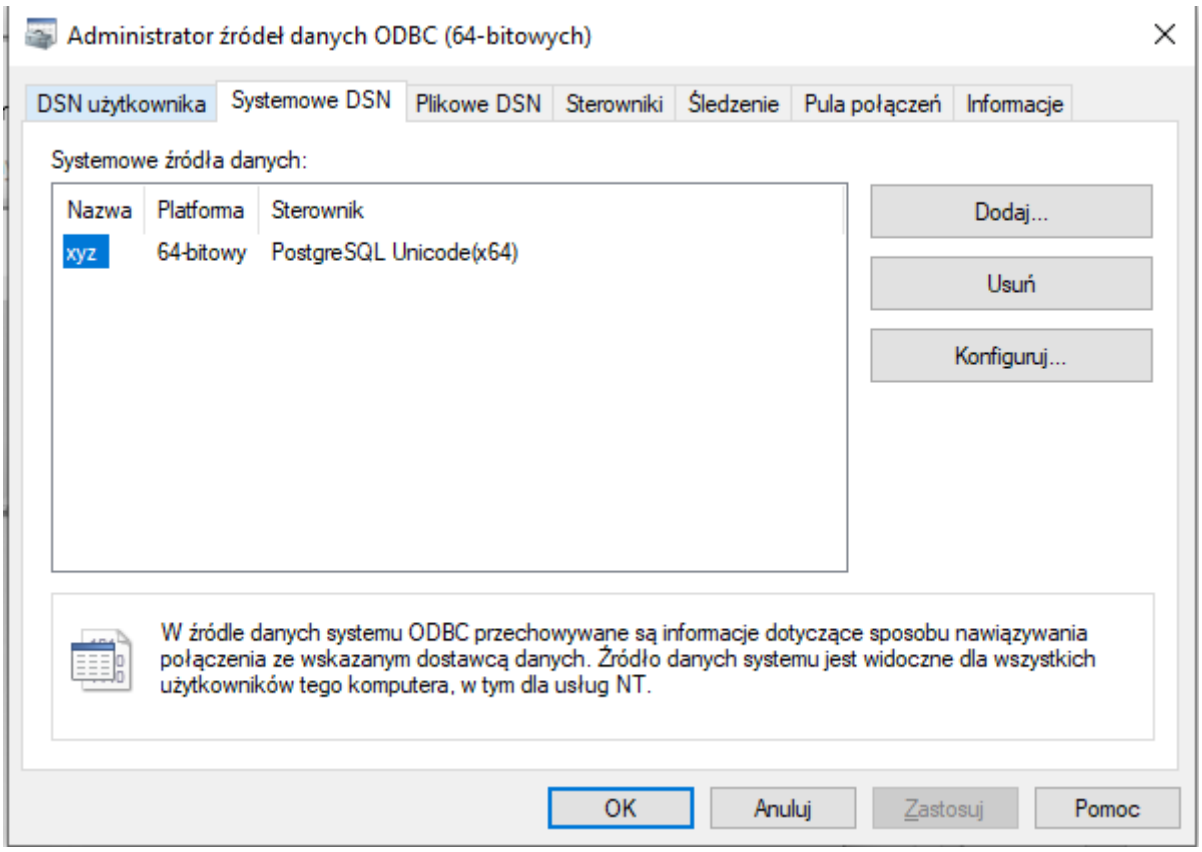
Następnie po naciśnięciu przycisku *Dodaj* należy wybrać sterownik *PostgreSQL Unicode(x64).*



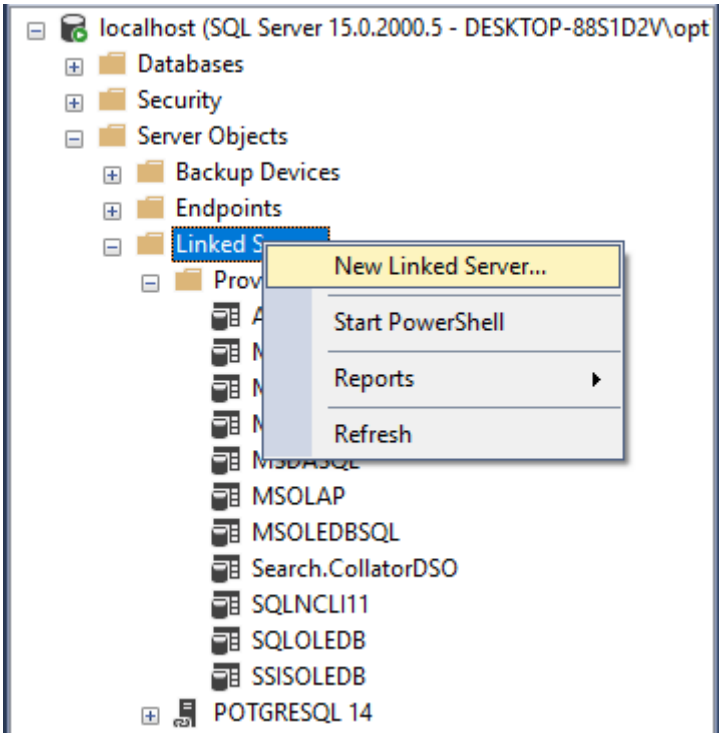
W nowo otwartym oknie należy podać dowolną nazwę źródła oraz wprowadzić dane serwera PostgreSQL.



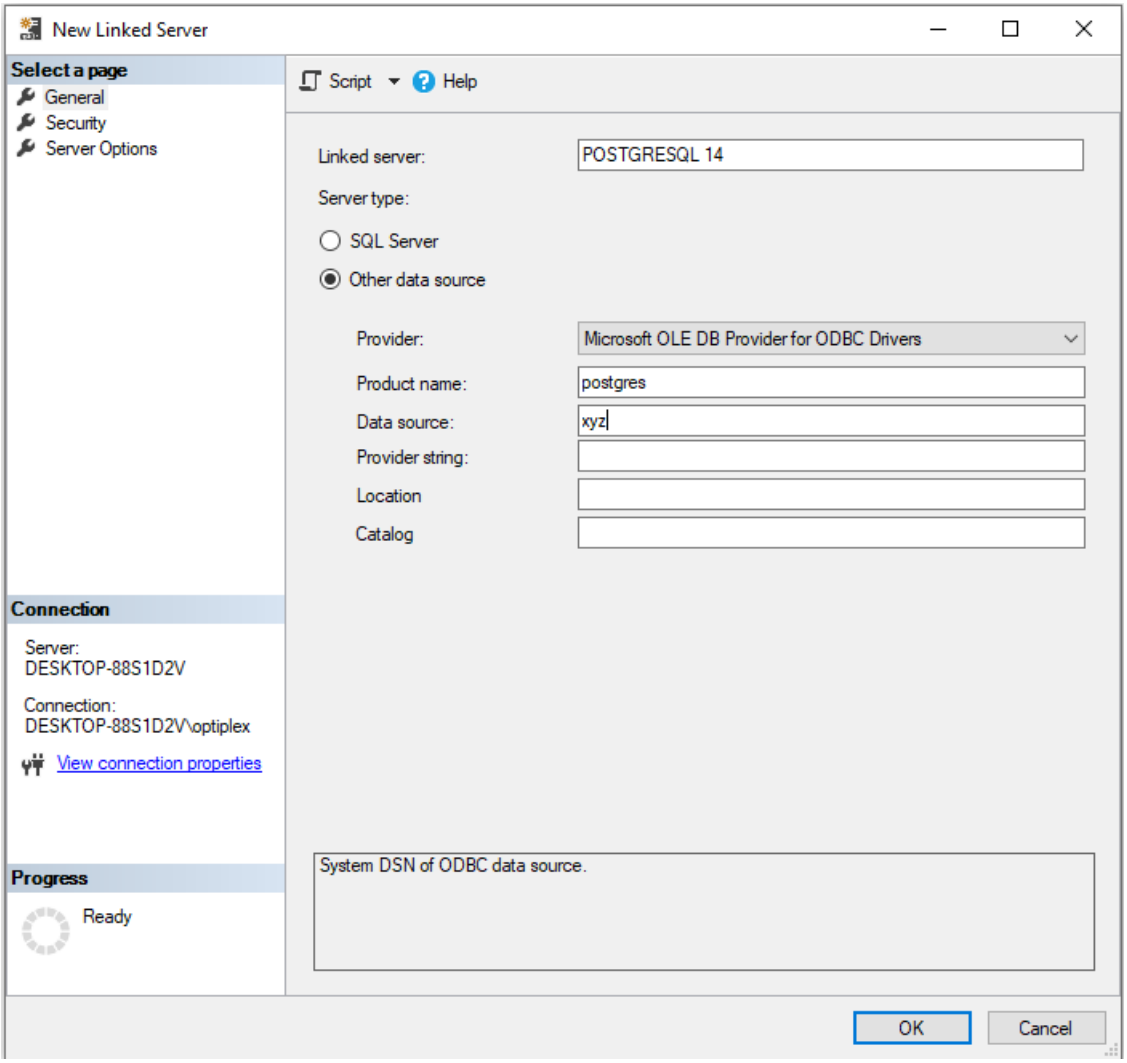
Naciśnięcie przycisku *Test* pozwala zweryfikować, czy udało się pomyślnie połączyć z bazą danych. W przypadku pomyślnego połączenia należy nacisnąć przycisk *Save*. W ten sposób poprawnie skonfigurowano sterownik, który pozwoli na połączenie systemów.



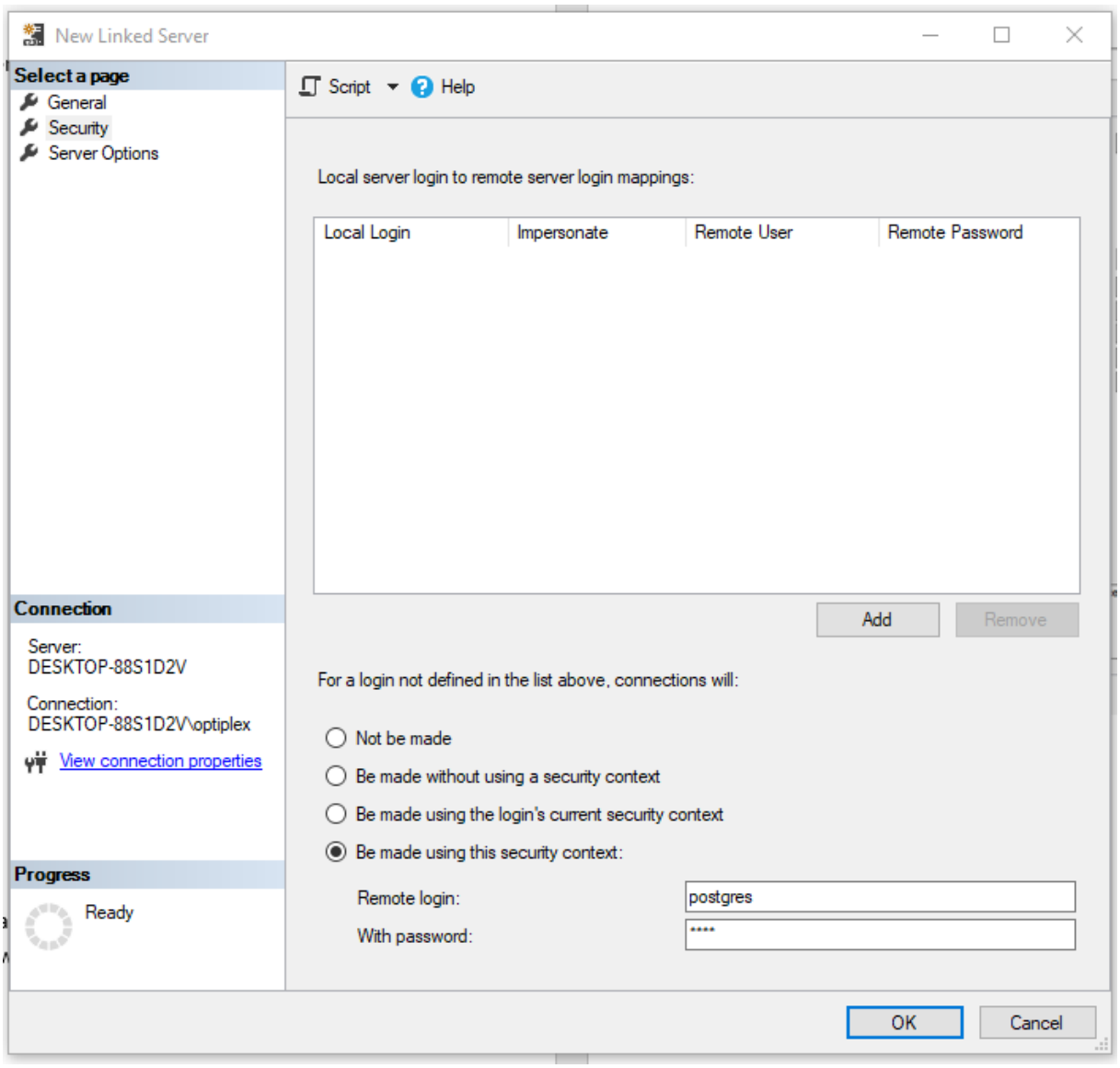
Kolejnym krokiem jest uruchomienie narzędzia Microsoft SQL Server Management Studio oraz wybór opcji *New Linked Server*:



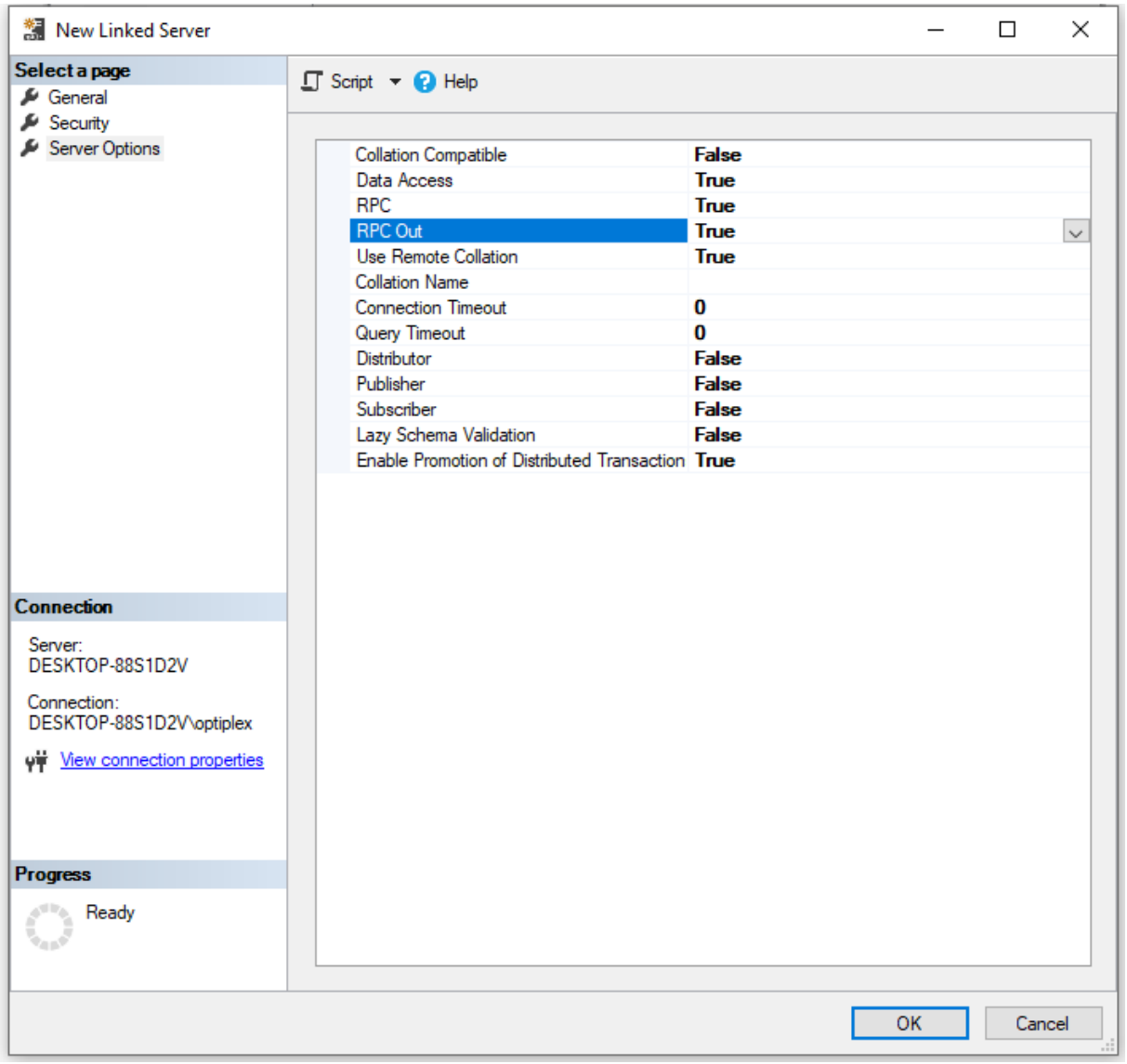
W oknie otwartym po naciśnięciu elementu listy należy podać nazwę linkowanego serwera, nazwę wcześniej zdefiniowanego źródła danych oraz nazwę produktu (w tym przypadku będzie to *postgres*). Należy również wybrać odpowiedni Provider – *Microsoft OLE DB Provider for ODBC Drivers*.



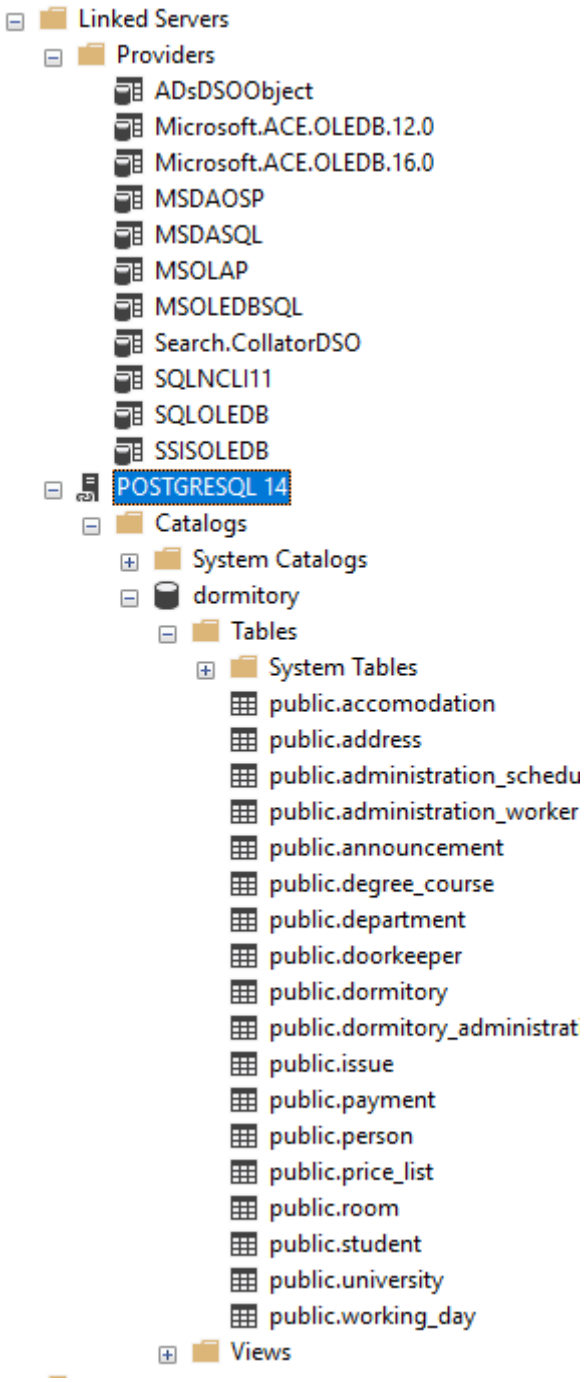
W zakładce *Security* podać należy dane użytkownika, który uzyska możliwość połączenia z serwerem:



W zakładce *Server Options* wartości *RPC oraz RPC OUT* powinny zostać ustawione na **True.**



W przypadku pomyślnej realizacji kroków połączenie powinno zostać ustanowione.



Poniżej przedstawiono rezultat pobrania danych z widoku *students*:

```sql
select * from [POSTGRESQL_14].dormitory.[public].students
```

Results | Messages

	id	First name	Last name	PESEL	Street	Flat number	ZIP	City	Country	Term	Degree course	Department	University
1	1439	Arnold	Eglise	00210171464	1390 Nisl. Av.	NULL	94-613	Przemyśl	Poland	6	Chemia	Wydział Chemii	Uniwersytet Warszawski
2	1505	Johna	Mohan	00210232514	9903 Id Street	NULL	19-294	Wałbrzych	Poland	5	Archeologia	Wydział Archeologii	Uniwersytet Warszawski
3	1587	Maxie	Bauser	00210249361	1263 Odio. Ave	NULL	09-357	Jelenia Góra	Poland	4	Biologia	Wydział Biologii	Uniwersytet Warszawski
4	1098	Cinderella	Coie	00210284269	P.O. Box 990, 5307 Proin Rd.	NULL	71-367	Wrocław	Poland	1	Archeologia	Wydział Archeologii	Uniwersytet Warszawski
5	1330	Alanna	D'eye	00210359934	9025 Fringilla Rd.	NULL	02-784	Lublin	Poland	4	Chemia	Wydział Chemii	Uniwersytet Warszawski
6	1222	Deni	Delacourt	00210426359	827-8201 Mauris Rd.	NULL	51-234	Toruń	Poland	3	Filozofia	Wydział Filozofii	Uniwersytet Warszawski
7	1138	Dix	Le Provest	00210522239	Ap #986-3887 Fames St.	NULL	19-813	Ełk	Poland	6	Filozofia	Wydział Filozofii	Uniwersytet Warszawski
8	1078	Aleda	Fihelly	00210721591	Ap #760-8993 Natoque Road	NULL	82-193	Stalowa Wola	Poland	6	Chemia	Wydział Chemii	Uniwersytet Warszawski
9	1308	Estrellita	Lackemann	00210727733	Ap #848-4082 Ipsum St.	NULL	73-588	Wrocław	Poland	1	Filozofia	Wydział Filozofii	Uniwersytet Warszawski
10	1155	Carly	Brosius	00210739318	139-1213 Nascetur Rd.	NULL	58-852	Toruń	Poland	1	Biologia	Wydział Biologii	Uniwersytet Warszawski
11	1263	Felita	Scrivens	00210832837	171-2241 Integer Rd.	NULL	45-158	Opole	Poland	5	Chemia	Wydział Chemii	Uniwersytet Warszawski
12	1204	Filmore	Gley	00210967999	6365 Lectus Rd.	NULL	95-029	Łomża	Poland	7	Biologia	Wydział Biologii	Uniwersytet Warszawski
13	1139	Netta	Clubley	00211053844	146-9157 Ad Av.	NULL	28-222	Zielona Góra	Poland	4	Archeologia	Wydział Archeologii	Uniwersytet Warszawski
14	1607	Ode	Davidof	00211148294	Ap #835-8275 Sit St.	NULL	86-084	Piotrków Trybunalski	Poland	4	Informatyka i ekonometria	Wydział Rolnictwa i Ekologii	Szkoła Główna Gospodarstwa Wiejskiego
15	1470	Beverley	De Carolis	00211166243	Ap #302-3999 Dolor, Avenue	NULL	06-169	Ostrowiec Świętokrzyski	Poland	4	Biologia	Wydział Biologii	Uniwersytet Warszawski
16	1117	Hendrika	Galbraith	00211273592	6036 Proin Avenue	NULL	40-577	Kraków	Poland	6	Biologia	Wydział Biologii	Uniwersytet Warszawski
17	1197	Tanny	Winnett	00211363589	177-3995 In Av.	NULL	84-542	Kielce	Poland	1	Filozofia	Wydział Filozofii	Uniwersytet Warszawski
18	1082	Charmane	MacTerlagh	00211446323	905-6001 At Av.	NULL	15-316	Łomża	Poland	2	Chemia	Wydział Chemii	Uniwersytet Warszawski

Wprowadzenie zmian po stronie systemu SQL Server:

```sql
update [POSTGRESQL_14].dormitory.[public].person set first_name = 'Stefan' where pesel = '00210171464'

select * from [POSTGRESQL_14].dormitory.[public].person where pesel = '00210171464'
```

Results | Messages

	id	first_name	second_name	last_name	pesel	address_id
1	1465	Stefan	NULL	Eglise	00210171464	1522

Odczyt danych po stronie PostgreSQL:

```sql
1  select * from person where pesel = '00210171464'
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	first_name character varying (25)	second_name character varying (25)	last_name character varying (50)	pesel character (11)	address_id integer
1	1465	Stefan	[null]	Eglise	00210171464	1522

Wprowadzenie zmian po stronie PostgreSQL:

```
1  update person set first_name='Roman' where pesel='00210171464'
2  select * from person where pesel='00210171464'
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	first_name character varying (25)	second_name character varying (25)	last_name character varying (50)	pesel character (11)	address_id integer
1	1465	Roman	[null]	Eglise	00210171464	1522

Odczyt danych po stronie SQL Server:

```
select * from [POSTGRESQL_14].dormitory.[public].person where pesel = '00210171464'
```

Results | Messages

	id	first_name	second_name	last_name	pesel	address_id
1	1465	Roman	NULL	Eglise	00210171464	1522

Jak można zauważyć, uzyskano możliwość modyfikacji danych z poziomu dwóch systemów bazodanowych.

7. Podsumowanie

a) Przeniesienie bazy danych z systemu SQL Server do PostgreSQL nie jest zadaniem trywialnym. Podstawowym krokiem przed przystąpieniem do migracji bazy danych jest zapoznanie się z dialektem docelowego systemu bazodanowego.

b) Istnieją narzędzia pozwalające na automatyczną migrację bazy danych. W przypadku przejścia z SQL Server do PostgreSQL, ze względu na różnice dialektowe, większość gotowych rozwiązań umożliwia jedynie przeniesienie tabel oraz widoków. Funkcje, procedury oraz wyzwalacze należy wygenerować za pomocą skryptów dopasowanych do nowej składni.

c) Za pomocą połączenia bazodanowego można zarządzać bazą danych z wykorzystaniem różnych systemów bazodanowych.