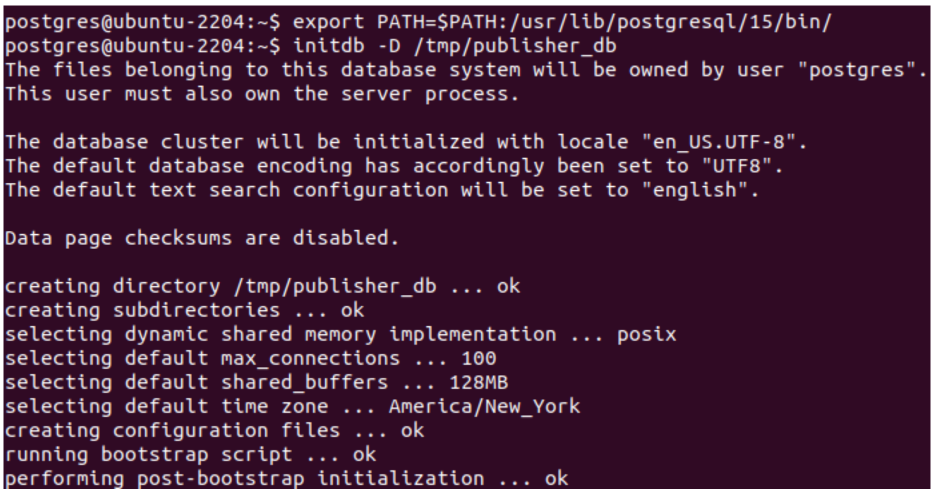
Przedmiot: Organizacja Systemów Zarządzania Baz Danych
Laboratorium 2: PgSQL – Replikacja logiczna
Autor: Bartłomiej Jamiołkowski, Adrianna Bodziony

Faza VI – Przygotowanie bazy danych

a. Na potrzeby ćwiczenia utworz z poziomu użytkownika postgres nowy klaster postgresowy w lokalizacji /tmp/publisher_db
b. .
c. .

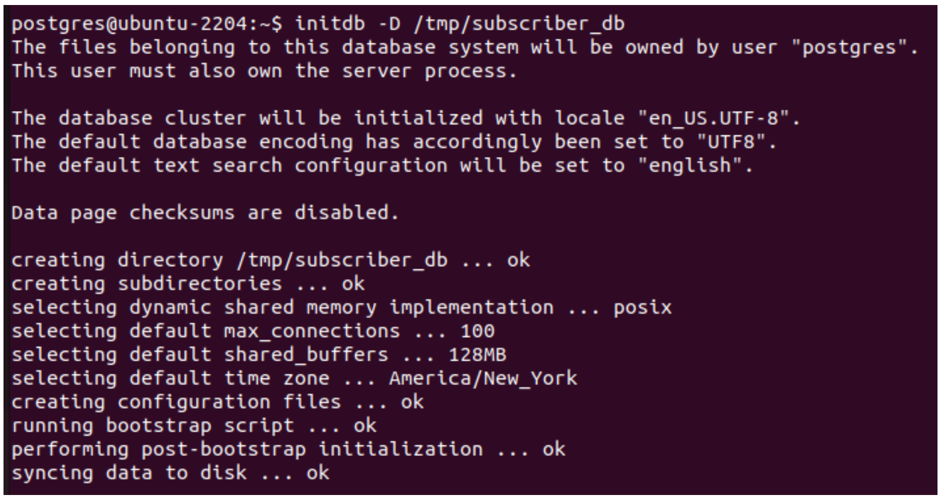
Polecenie:



```
postgres@ubuntu-2204:~$ export PATH=$PATH:/usr/lib/postgresql/15/bin/
postgres@ubuntu-2204:~$ initdb -D /tmp/publisher_db
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /tmp/publisher_db ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... America/New_York
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
```

d. Utworz z poziomu użytkownika postgres nowy klaster postgresowy w lokalizacji /tmp/subscriber_db

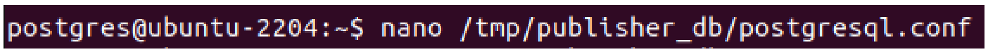
Polecenie:



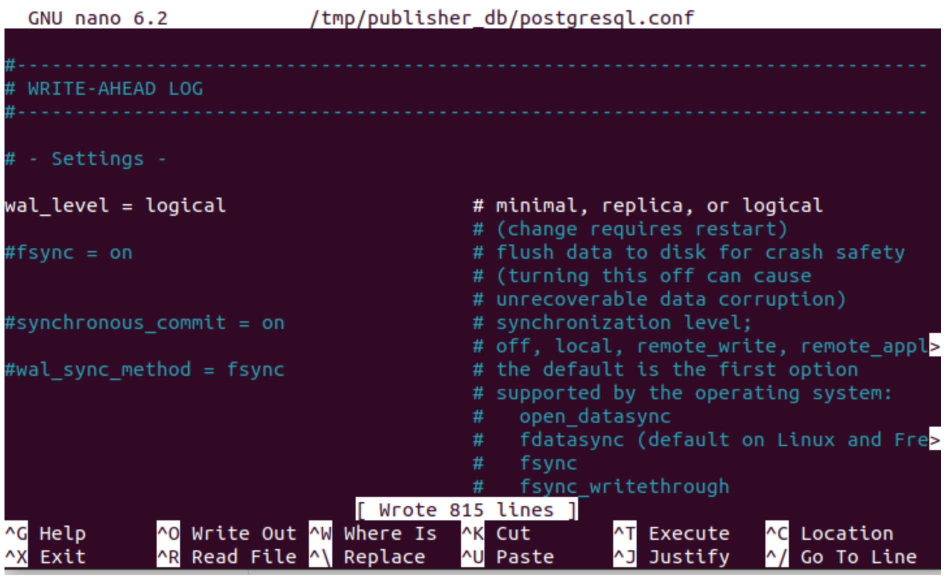
```
postgres@ubuntu-2204:~$ initdb -D /tmp/subscriber_db
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /tmp/subscriber_db ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... America/New_York
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
```

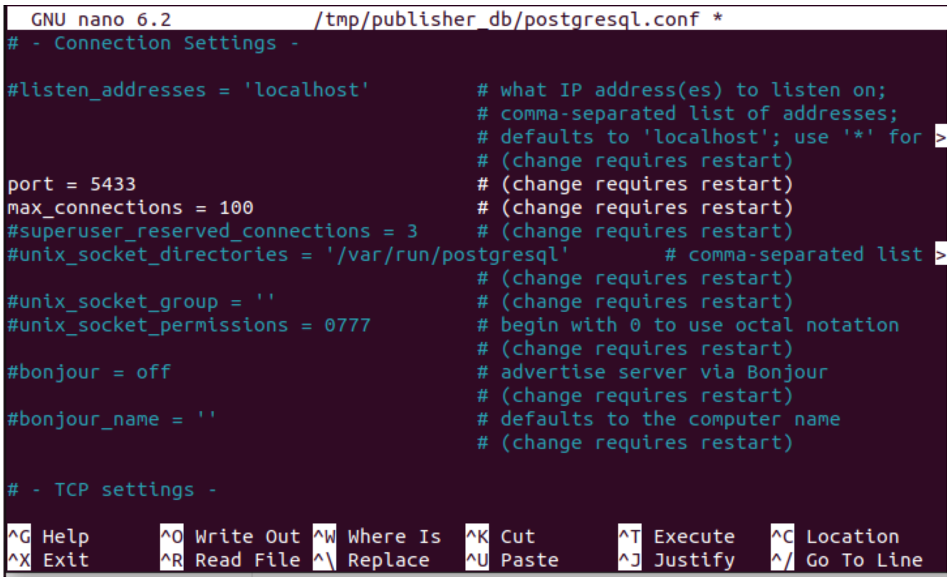
e. Dokonujemy zmian w konfiguracji publishera ustawiając „wal_level" na logical a port na którym instancja ta będzie uruchomiona na 5433

Polecenie:



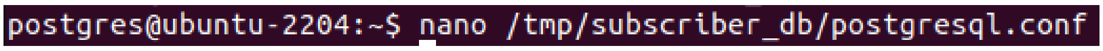
```
postgres@ubuntu-2204:~$ nano /tmp/publisher_db/postgresql.conf
```

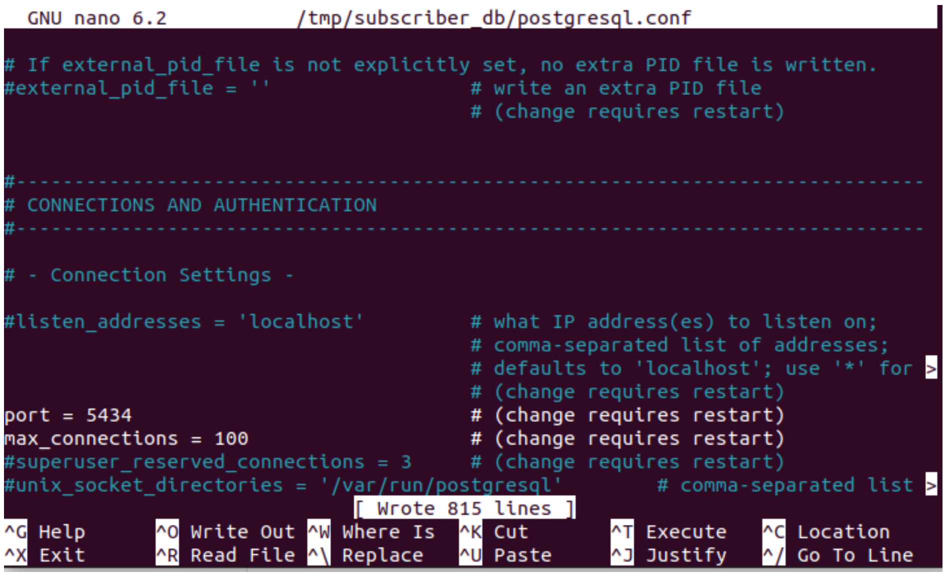

```
  GNU nano 6.2             /tmp/publisher_db/postgresql.conf
#------------------------------------------------------------------------------
# WRITE-AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = logical                     # minimal, replica, or logical
                                        # (change requires restart)
#fsync = on                             # flush data to disk for crash safety
                                        # (turning this off can cause
                                        # unrecoverable data corruption)
#synchronous_commit = on                # synchronization level;
                                        # off, local, remote_write, remote_appl>
#wal_sync_method = fsync                # the default is the first option
                                        # supported by the operating system:
                                        #   open_datasync
                                        #   fdatasync (default on Linux and Fre>
                                        #   fsync
                                        #   fsync_writethrough
                              [ Wrote 815 lines ]
^G Help        ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```


```
  GNU nano 6.2             /tmp/publisher_db/postgresql.conf *
# - Connection Settings -

#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for >
                                        # (change requires restart)
port = 5433                             # (change requires restart)
max_connections = 100                   # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/var/run/postgresql'        # comma-separated list >
                                        # (change requires restart)
#unix_socket_group = ''                 # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
                                        # (change requires restart)
#bonjour = off                          # advertise server via Bonjour
                                        # (change requires restart)
#bonjour_name = ''                      # defaults to the computer name
                                        # (change requires restart)

# - TCP settings -

^G Help        ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```

f. Zmieniamy port subscribera na 5434

Polecenie:

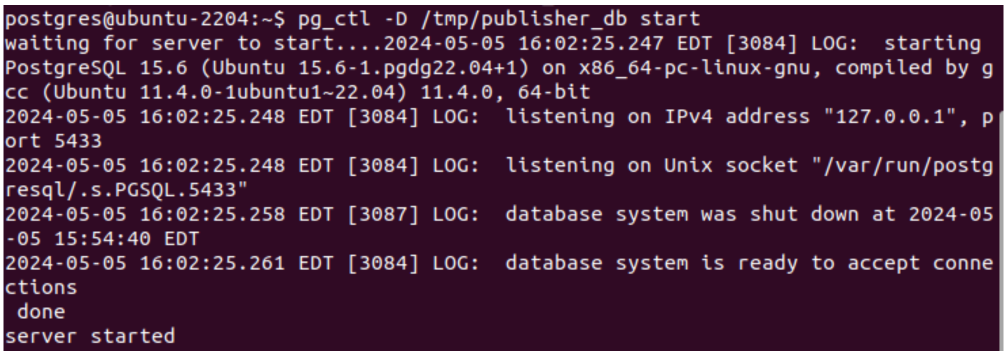

```
postgres@ubuntu-2204:~$ nano /tmp/subscriber_db/postgresql.conf
```



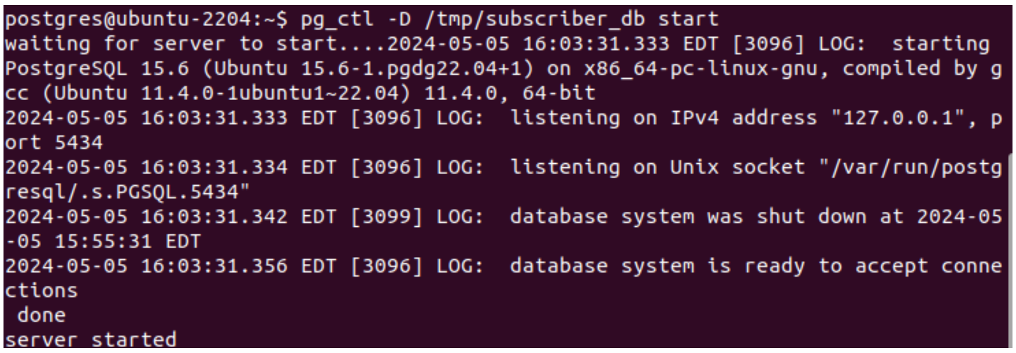
```
  GNU nano 6.2           /tmp/subscriber_db/postgresql.conf

# If external_pid_file is not explicitly set, no extra PID file is written.
#external_pid_file = ''                 # write an extra PID file
                                        # (change requires restart)


#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for >
                                        # (change requires restart)
port = 5434                             # (change requires restart)
max_connections = 100                   # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/var/run/postgresql'        # comma-separated list >
                                [ Wrote 815 lines ]
^G Help         ^O Write Out ^W Where Is  ^K Cut          ^T Execute    ^C Location
^X Exit         ^R Read File ^\ Replace   ^U Paste        ^J Justify    ^/ Go To Line
```

g. Uruchom obie instancje (polecenie pg_ctl)

Polecenie:



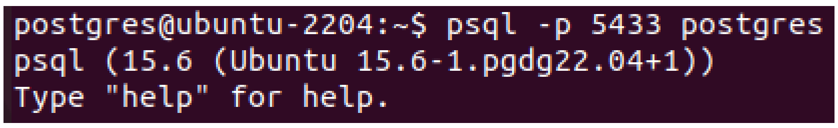
```
postgres@ubuntu-2204:~$ pg_ctl -D /tmp/publisher_db start
waiting for server to start....2024-05-05 16:02:25.247 EDT [3084] LOG:  starting
PostgreSQL 15.6 (Ubuntu 15.6-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compiled by g
cc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
2024-05-05 16:02:25.248 EDT [3084] LOG:  listening on IPv4 address "127.0.0.1", p
ort 5433
2024-05-05 16:02:25.248 EDT [3084] LOG:  listening on Unix socket "/var/run/postg
resql/.s.PGSQL.5433"
2024-05-05 16:02:25.258 EDT [3087] LOG:  database system was shut down at 2024-05
-05 15:54:40 EDT
2024-05-05 16:02:25.261 EDT [3084] LOG:  database system is ready to accept conne
ctions
 done
server started
```



```
postgres@ubuntu-2204:~$ pg_ctl -D /tmp/subscriber_db start
waiting for server to start....2024-05-05 16:03:31.333 EDT [3096] LOG:  starting
PostgreSQL 15.6 (Ubuntu 15.6-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compiled by g
cc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit
2024-05-05 16:03:31.333 EDT [3096] LOG:  listening on IPv4 address "127.0.0.1", p
ort 5434
2024-05-05 16:03:31.334 EDT [3096] LOG:  listening on Unix socket "/var/run/postg
resql/.s.PGSQL.5434"
2024-05-05 16:03:31.342 EDT [3099] LOG:  database system was shut down at 2024-05
-05 15:55:31 EDT
2024-05-05 16:03:31.356 EDT [3096] LOG:  database system is ready to accept conne
ctions
 done
server started
```

h. Łączymy się z instancją podstawową (polecenie psql, baza danych postgres)

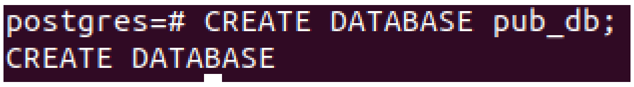
Polecenie:



```
postgres@ubuntu-2204:~$ psql -p 5433 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.
```

i. Tworzymy nową bazę danych pub_db

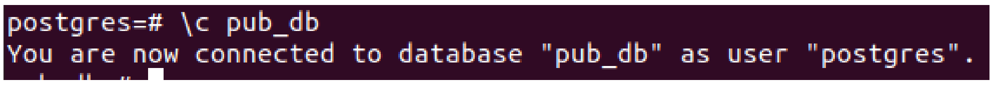
Polecenie:



```
postgres=# CREATE DATABASE pub_db;
CREATE DATABASE
```

j. Przełączamy się na nowo stworzoną baze danych

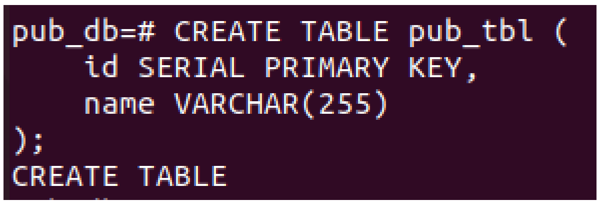
Polecenie:



```
postgres=# \c pub_db
You are now connected to database "pub_db" as user "postgres".
```

k. Tworzymy nowa tabele pub_tbl z polami int id, oraz varchar name

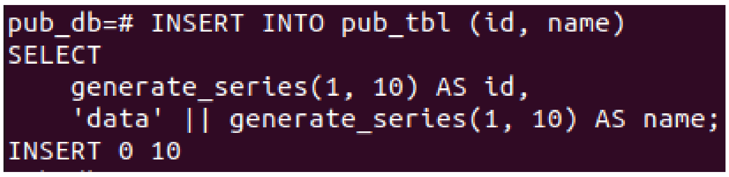
Polecenie:



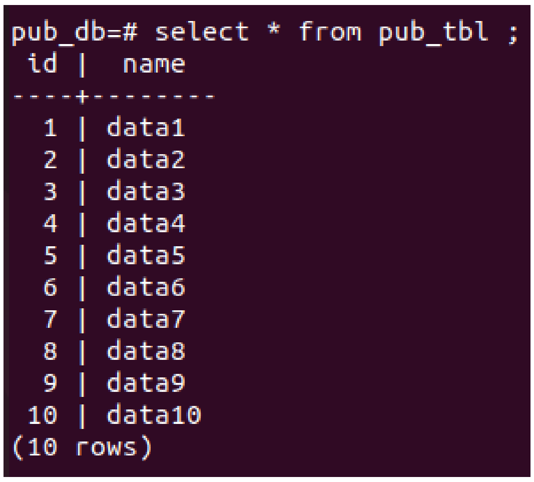
```
pub_db=# CREATE TABLE pub_tbl (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255)
);
CREATE TABLE
```

l. Wstawiamy do nowo stworzonej tabeli 10 wierszy (żeby przyspieszyc uzyjmy funkcji generate_series dla wygenerowania kolejnych wartości id oraz name) . Po wykonaniu tego kroku zawartość tabeli powinna być następująca:

Polecenie:



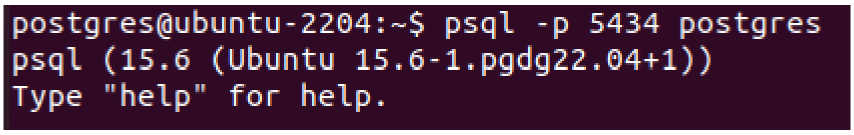
```
pub_db=# INSERT INTO pub_tbl (id, name)
SELECT
    generate_series(1, 10) AS id,
    'data' || generate_series(1, 10) AS name;
INSERT 0 10
```



```
pub_db=# select * from pub_tbl ;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
(10 rows)
```

m. .
n. Ponieważ sam mechanizm replikacyjny nie odtworzy / zreplikuje nam struktury bazy danych / tabeli – musimy o to zadbać samodzielnie.
o. Laczymy się zatem z instancja subscriber (port 5434, baza danych postgres)

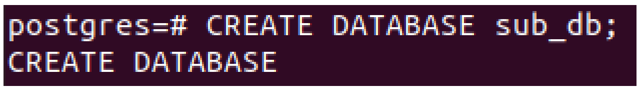
Polecenie:



```
postgres@ubuntu-2204:~$ psql -p 5434 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.
```

p. Tworzymy baze danych sub_db;

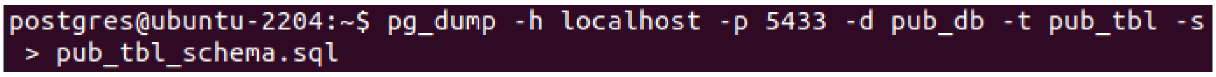
Polecenie:



```
postgres=# CREATE DATABASE sub_db;
CREATE DATABASE
```

q. Żeby przenieść na subscribera sam schemat tabeli pub_tbl możemy wspomóc się tutaj poleceniem pg_dump

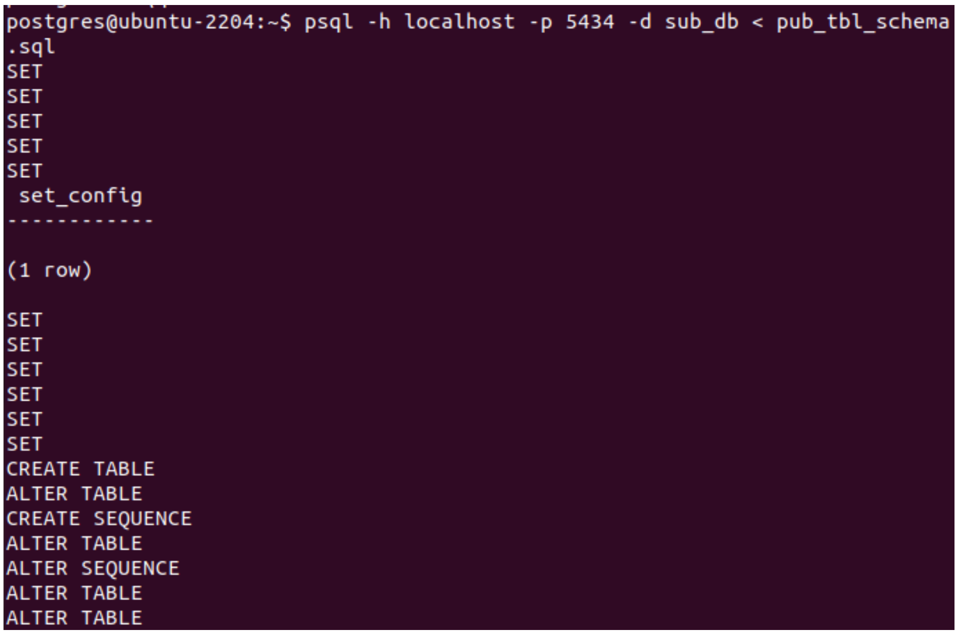
Polecenie:



```
postgres@ubuntu-2204:~$ pg_dump -h localhost -p 5433 -d pub_db -t pub_tbl -s
 > pub_tbl_schema.sql
```

r. Zatem, rozłączamy się z bazą danych poublisera i z poziomu linii poleceń wykonujemy zrzut tabeli pub_tbl z bazy danych pub_db instancji podstawowej i wynik przekierowujemy na wejście polecenia psql laczac się do instancji subscribera do wlasnie stworzonej bazy danych sub_db

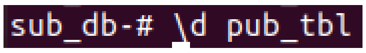
Polecenie:



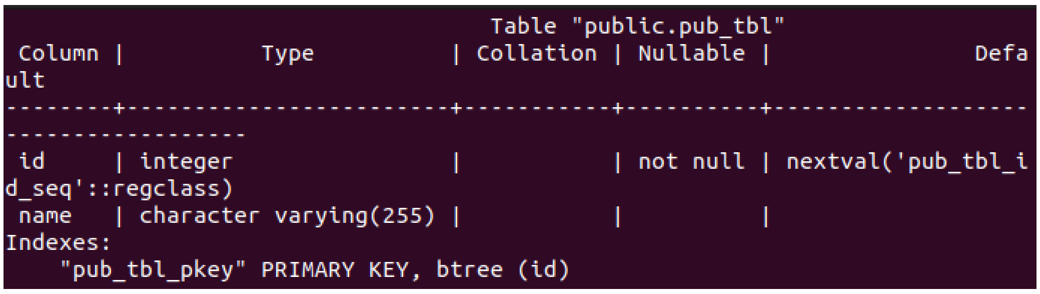
```
postgres@ubuntu-2204:~$ psql -h localhost -p 5434 -d sub_db < pub_tbl_schema
.sql
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER TABLE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
```

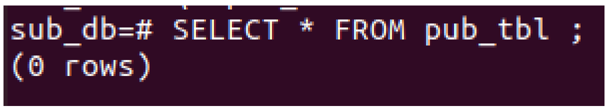
s. Następnie, łączymy się z instancja subscribera z baza danych sub_db wyświetlamy schemat tabeli pub_tbl oraz pobieramy z niej dane. Efekt na tym etapie powinien być następujący:

Polecenie:



```
sub_db-# \d pub_tbl
```


```
                                    Table "public.pub_tbl"
 Column |          Type          | Collation | Nullable |              Defa
ult
--------+------------------------+-----------+----------+------------------
------------------
 id     | integer                |           | not null | nextval('pub_tbl_i
d_seq'::regclass)
 name   | character varying(255) |           |          |
Indexes:
    "pub_tbl_pkey" PRIMARY KEY, btree (id)
```

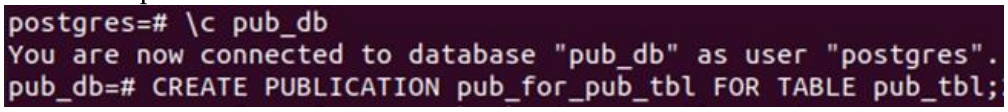

```
sub_db=# SELECT * FROM pub_tbl ;
(0 rows)
```

t. .

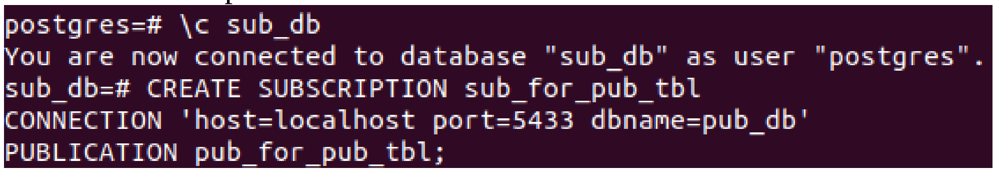
u. A zatem, mamy na subscribera przeniesioną strukturę tabeli pub_tbl, natomiast nie przenieśliśmy tam żadnych danych.

v. W kolejnym kroku musimy na obu instancjach stworzy odpowiednio „publikacje" i subskrypcje (polecenia odpowienio create publication oraz create subscription)

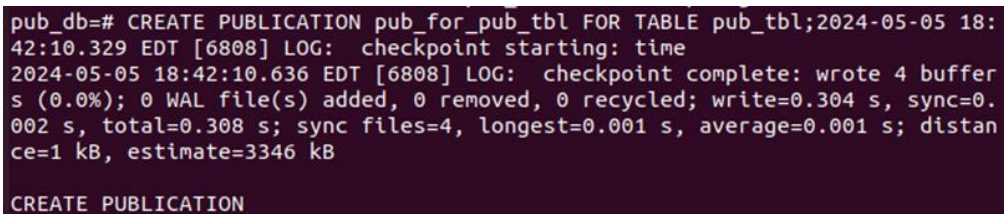
Polecenie create publication:

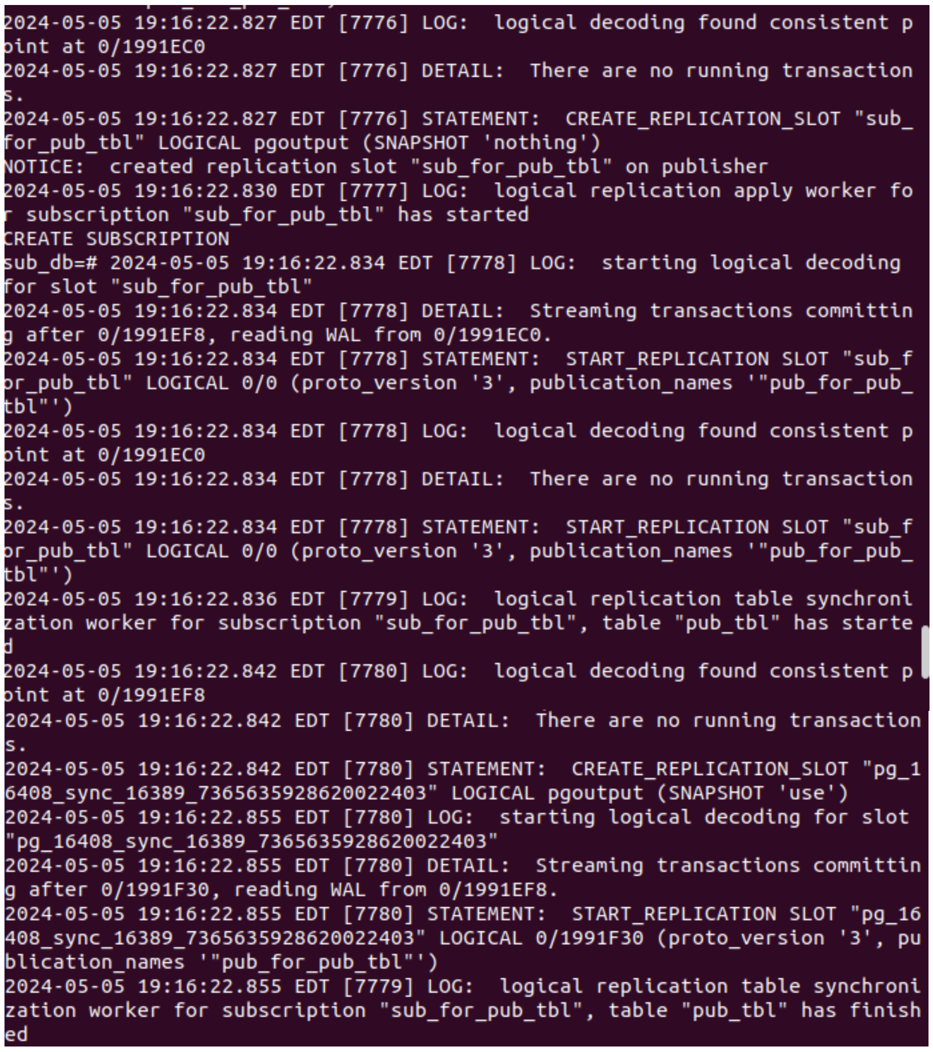

```
postgres=# \c pub_db
You are now connected to database "pub_db" as user "postgres".
pub_db=# CREATE PUBLICATION pub_for_pub_tbl FOR TABLE pub_tbl;
```

Polecenie create subscription:


```
postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# CREATE SUBSCRIPTION sub_for_pub_tbl
CONNECTION 'host=localhost port=5433 dbname=pub_db'
PUBLICATION pub_for_pub_tbl;
```

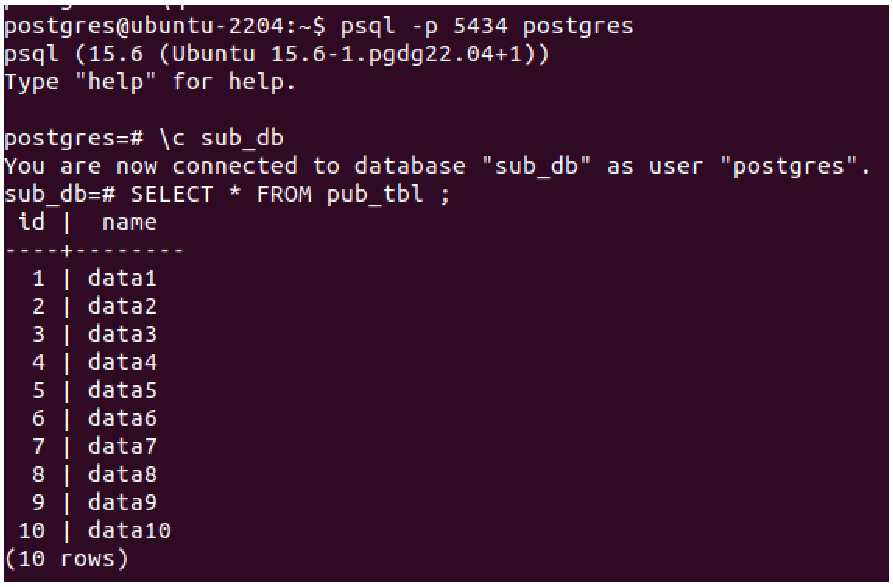
w. Po wykonaniu tego kroku w logach obu instancji powinniśmy zobaczyć informacje o utworzeniu publikacji, subskrypcji i o uruchomieniu replikacji logicznej pomiędzy oboma instancjami


```
pub_db=# CREATE PUBLICATION pub_for_pub_tbl FOR TABLE pub_tbl;2024-05-05 18:
42:10.329 EDT [6808] LOG:  checkpoint starting: time
2024-05-05 18:42:10.636 EDT [6808] LOG:  checkpoint complete: wrote 4 buffer
s (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.304 s, sync=0.
002 s, total=0.308 s; sync files=4, longest=0.001 s, average=0.001 s; distan
ce=1 kB, estimate=3346 kB

CREATE PUBLICATION
```



```
2024-05-05 19:16:22.827 EDT [7776] LOG:  logical decoding found consistent p
oint at 0/1991EC0
2024-05-05 19:16:22.827 EDT [7776] DETAIL:  There are no running transaction
s.
2024-05-05 19:16:22.827 EDT [7776] STATEMENT:  CREATE_REPLICATION_SLOT "sub_
for_pub_tbl" LOGICAL pgoutput (SNAPSHOT 'nothing')
NOTICE:  created replication slot "sub_for_pub_tbl" on publisher
2024-05-05 19:16:22.830 EDT [7777] LOG:  logical replication apply worker fo
r subscription "sub_for_pub_tbl" has started
CREATE SUBSCRIPTION
sub_db=# 2024-05-05 19:16:22.834 EDT [7778] LOG:  starting logical decoding
for slot "sub_for_pub_tbl"
2024-05-05 19:16:22.834 EDT [7778] DETAIL:  Streaming transactions committin
g after 0/1991EF8, reading WAL from 0/1991EC0.
2024-05-05 19:16:22.834 EDT [7778] STATEMENT:  START_REPLICATION SLOT "sub_f
or_pub_tbl" LOGICAL 0/0 (proto_version '3', publication_names '"pub_for_pub_
tbl"')
2024-05-05 19:16:22.834 EDT [7778] LOG:  logical decoding found consistent p
oint at 0/1991EC0
2024-05-05 19:16:22.834 EDT [7778] DETAIL:  There are no running transaction
s.
2024-05-05 19:16:22.834 EDT [7778] STATEMENT:  START_REPLICATION SLOT "sub_f
or_pub_tbl" LOGICAL 0/0 (proto_version '3', publication_names '"pub_for_pub_
tbl"')
2024-05-05 19:16:22.836 EDT [7779] LOG:  logical replication table synchroni
zation worker for subscription "sub_for_pub_tbl", table "pub_tbl" has starte
d
2024-05-05 19:16:22.842 EDT [7780] LOG:  logical decoding found consistent p
oint at 0/1991EF8
2024-05-05 19:16:22.842 EDT [7780] DETAIL:  There are no running transaction
s.
2024-05-05 19:16:22.842 EDT [7780] STATEMENT:  CREATE_REPLICATION_SLOT "pg_1
6408_sync_16389_7365635928620022403" LOGICAL pgoutput (SNAPSHOT 'use')
2024-05-05 19:16:22.855 EDT [7780] LOG:  starting logical decoding for slot
"pg_16408_sync_16389_7365635928620022403"
2024-05-05 19:16:22.855 EDT [7780] DETAIL:  Streaming transactions committin
g after 0/1991F30, reading WAL from 0/1991EF8.
2024-05-05 19:16:22.855 EDT [7780] STATEMENT:  START_REPLICATION SLOT "pg_16
408_sync_16389_7365635928620022403" LOGICAL 0/1991F30 (proto_version '3', pu
blication_names '"pub_for_pub_tbl"')
2024-05-05 19:16:22.855 EDT [7779] LOG:  logical replication table synchroni
zation worker for subscription "sub_for_pub_tbl", table "pub_tbl" has finish
ed
```

x. .
y. W efekcie czego dane na instancji „subscriber” powinny zostać zreplikowane z instancji publisher



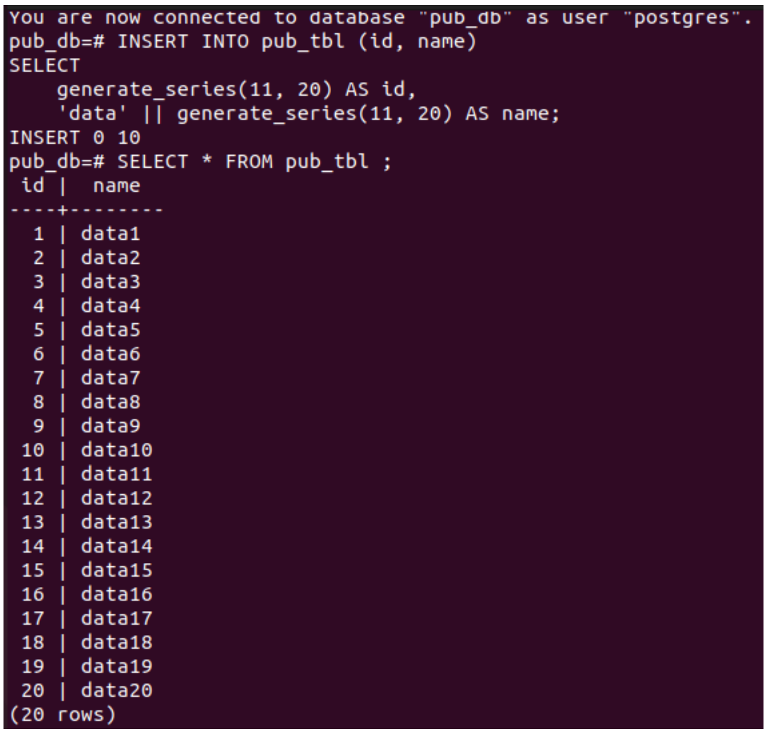
```
postgres@ubuntu-2204:~$ psql -p 5434 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.

postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
(10 rows)
```

z. .

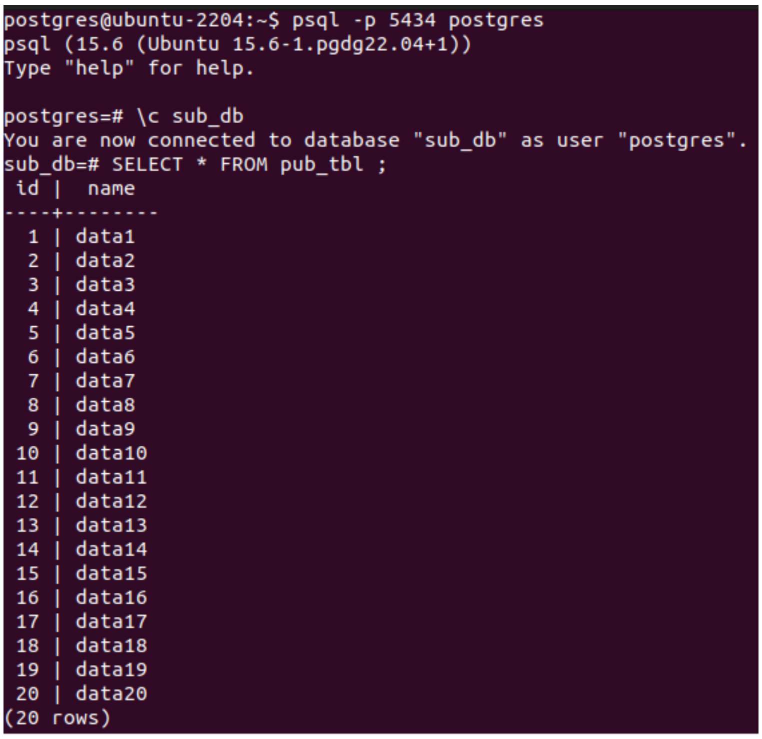
aa. Na potrzeby weryfikacji działania:

i. Wstaw na pierwszej instancji kolejnych 10 wierszy, sprawdź ich replikacje na drugą instancje

Polecenie:



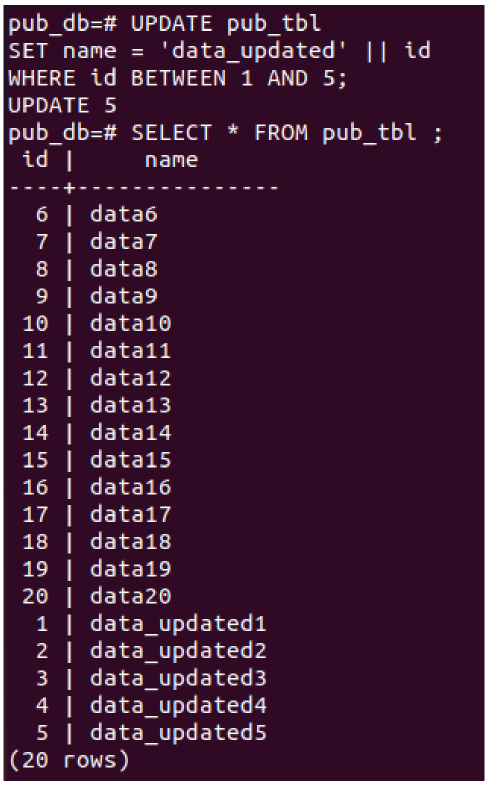
```
You are now connected to database "pub_db" as user "postgres".
pub_db=# INSERT INTO pub_tbl (id, name)
SELECT
    generate_series(11, 20) AS id,
    'data' || generate_series(11, 20) AS name;
INSERT 0 10
pub_db=# SELECT * FROM pub_tbl ;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
 16 | data16
 17 | data17
 18 | data18
 19 | data19
 20 | data20
(20 rows)
```



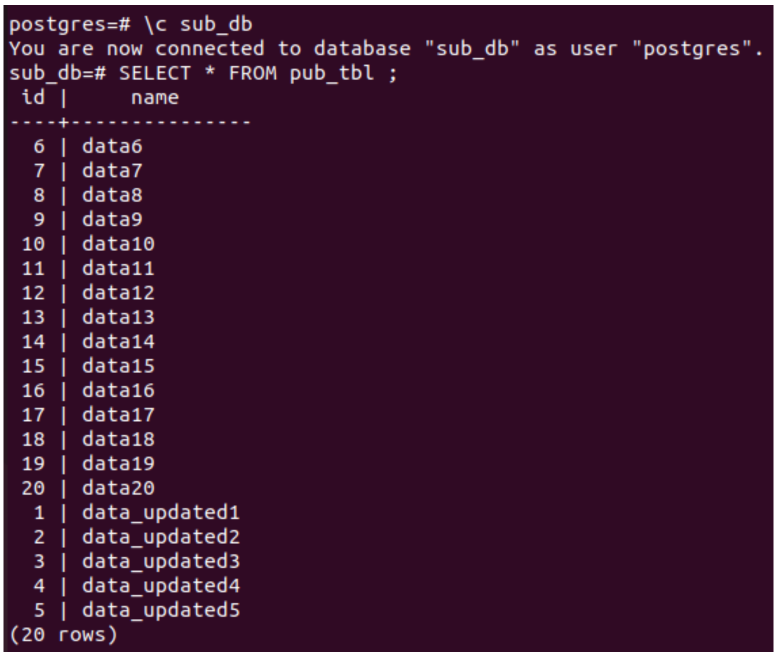
```
postgres@ubuntu-2204:~$ psql -p 5434 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.

postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
 16 | data16
 17 | data17
 18 | data18
 19 | data19
 20 | data20
(20 rows)
```

ii. Wykonaj update na części rekordow na pierwszej instancji, zweryfikuj zreplikowanie danych na druga instancje

Polecenie:



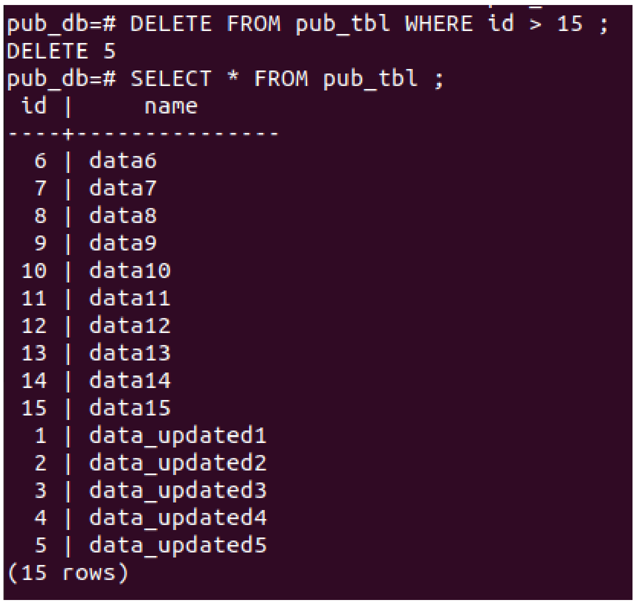
```
pub_db=# UPDATE pub_tbl
SET name = 'data_updated' || id
WHERE id BETWEEN 1 AND 5;
UPDATE 5
pub_db=# SELECT * FROM pub_tbl ;
 id |      name
----+---------------
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
 16 | data16
 17 | data17
 18 | data18
 19 | data19
 20 | data20
  1 | data_updated1
  2 | data_updated2
  3 | data_updated3
  4 | data_updated4
  5 | data_updated5
(20 rows)
```



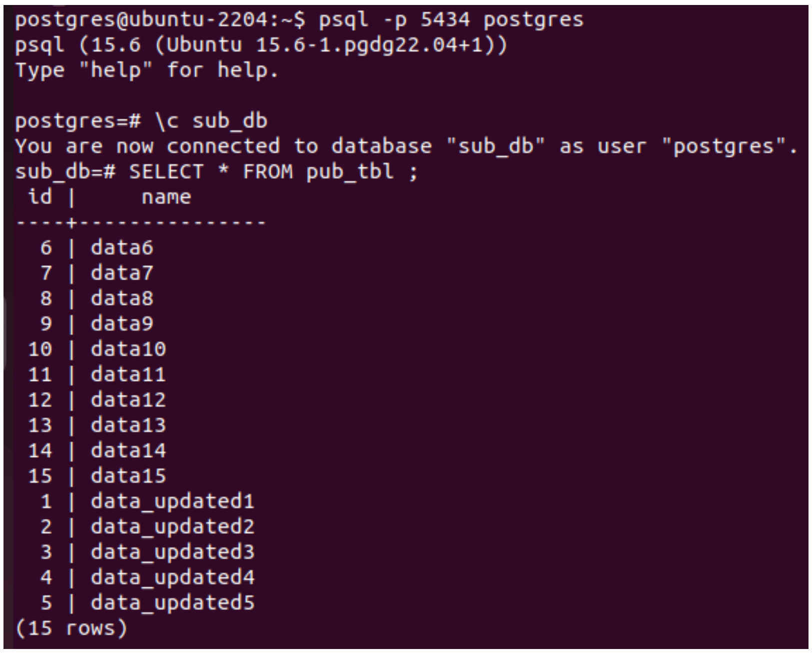
```
postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id |      name
----+---------------
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
 16 | data16
 17 | data17
 18 | data18
 19 | data19
 20 | data20
  1 | data_updated1
  2 | data_updated2
  3 | data_updated3
  4 | data_updated4
  5 | data_updated5
(20 rows)
```

iii. Usuń część wierszy na pierwszej instancji, zweryfikuj zreplikowanie danych na druga instancje

Polecenie:



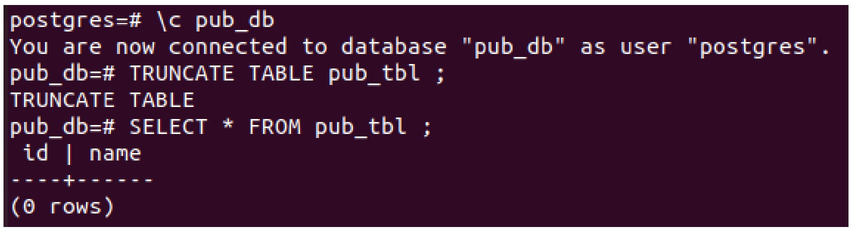
```
pub_db=# DELETE FROM pub_tbl WHERE id > 15 ;
DELETE 5
pub_db=# SELECT * FROM pub_tbl ;
 id |      name
----+---------------
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
  1 | data_updated1
  2 | data_updated2
  3 | data_updated3
  4 | data_updated4
  5 | data_updated5
(15 rows)
```



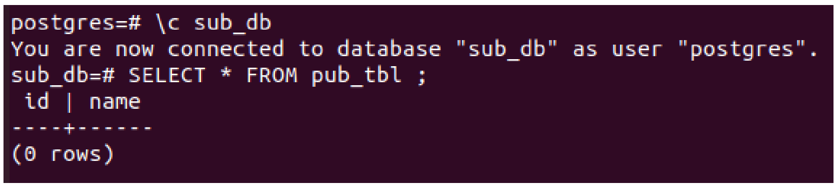
```
postgres@ubuntu-2204:~$ psql -p 5434 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.

postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id |      name
----+---------------
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
 11 | data11
 12 | data12
 13 | data13
 14 | data14
 15 | data15
  1 | data_updated1
  2 | data_updated2
  3 | data_updated3
  4 | data_updated4
  5 | data_updated5
(15 rows)
```

iv. Wykonaj na pierwszej instancji operację truncate, zweryfikuj zreplikowanie danych na druga instancje

Polecenie:



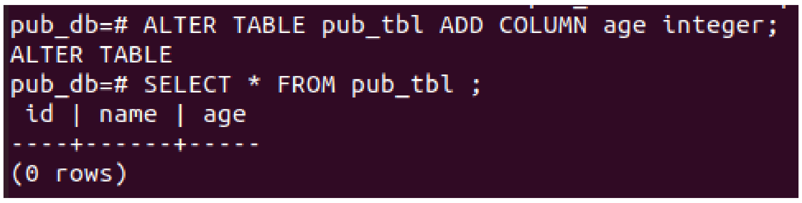
```
postgres=# \c pub_db
You are now connected to database "pub_db" as user "postgres".
pub_db=# TRUNCATE TABLE pub_tbl ;
TRUNCATE TABLE
pub_db=# SELECT * FROM pub_tbl ;
 id | name
----+------
(0 rows)
```



```
postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id | name
----+------
(0 rows)
```

v. Zmodyfikuj na instancji pierwszej schemat tabeli pub_tbl dodając nową kolumnę dowolnego typu.

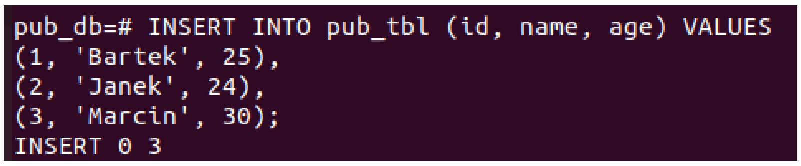
Polecenie:



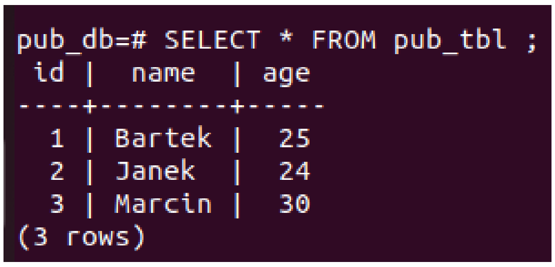
```
pub_db=# ALTER TABLE pub_tbl ADD COLUMN age integer;
ALTER TABLE
pub_db=# SELECT * FROM pub_tbl ;
 id | name | age
----+------+-----
(0 rows)
```

vi. Dodaj do zmodyfikowanej tabeli kilka wierszy, sprawdz działanie replikacji. Zweryfikuj i rozwiąż zaistniały problem

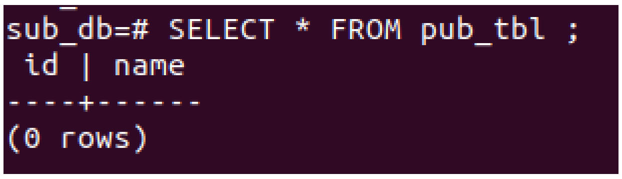
Polecenie:



```
pub_db=# INSERT INTO pub_tbl (id, name, age) VALUES
(1, 'Bartek', 25),
(2, 'Janek', 24),
(3, 'Marcin', 30);
INSERT 0 3
```



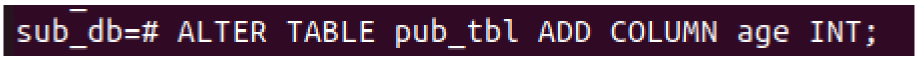
```
pub_db=# SELECT * FROM pub_tbl ;
 id |  name   | age
----+--------+-----
  1 | Bartek |  25
  2 | Janek  |  24
  3 | Marcin |  30
(3 rows)
```



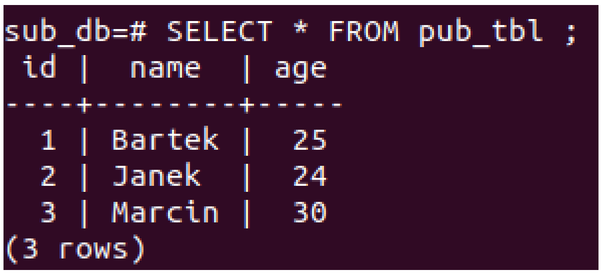
```
sub_db=# SELECT * FROM pub_tbl ;
 id | name
----+------
(0 rows)
```

Zaistniał problem - replikacja nie uwzględniła nowej kolumny.

Replikacja logiczna wymaga, aby wszystkie kolumny obecne na instancji wydającej publisher były również obecne na instancji subskrybenta.



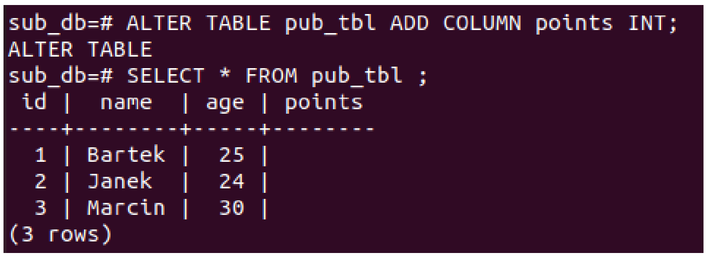
```
sub_db=# ALTER TABLE pub_tbl ADD COLUMN age INT;
```



```
sub_db=# SELECT * FROM pub_tbl ;
 id |  name   | age
----+---------+-----
  1 | Bartek  |  25
  2 | Janek   |  24
  3 | Marcin  |  30
(3 rows)
```

vii. Zmodyfikuj schemat tabeli pub_tbl na drugiej instancji dodając nową kolumnę dowolnego typu.

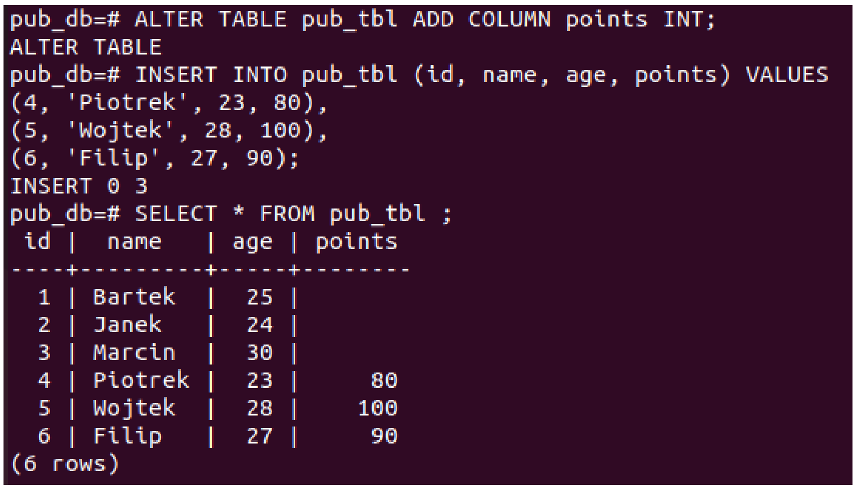
Polecenie:



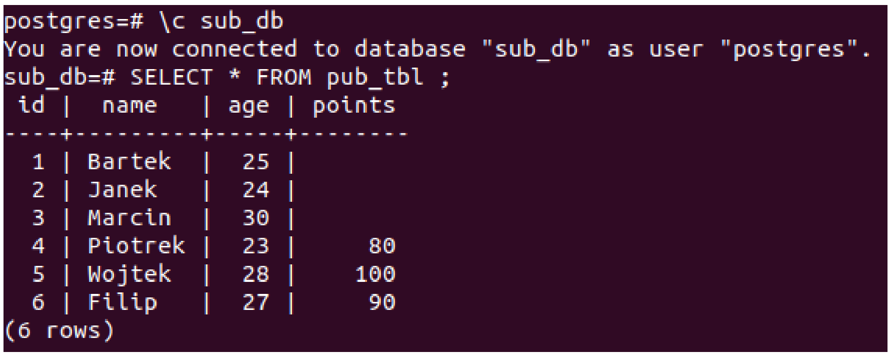
```
sub_db=# ALTER TABLE pub_tbl ADD COLUMN points INT;
ALTER TABLE
sub_db=# SELECT * FROM pub_tbl ;
 id |  name   | age | points
----+---------+-----+--------
  1 | Bartek  |  25 |
  2 | Janek   |  24 |
  3 | Marcin  |  30 |
(3 rows)
```

viii. Dodaj do tabeli pub_tbl kilka wierszy na pierwszej instancji. Zweryfikuj działanie replikacji.

Polecenie:



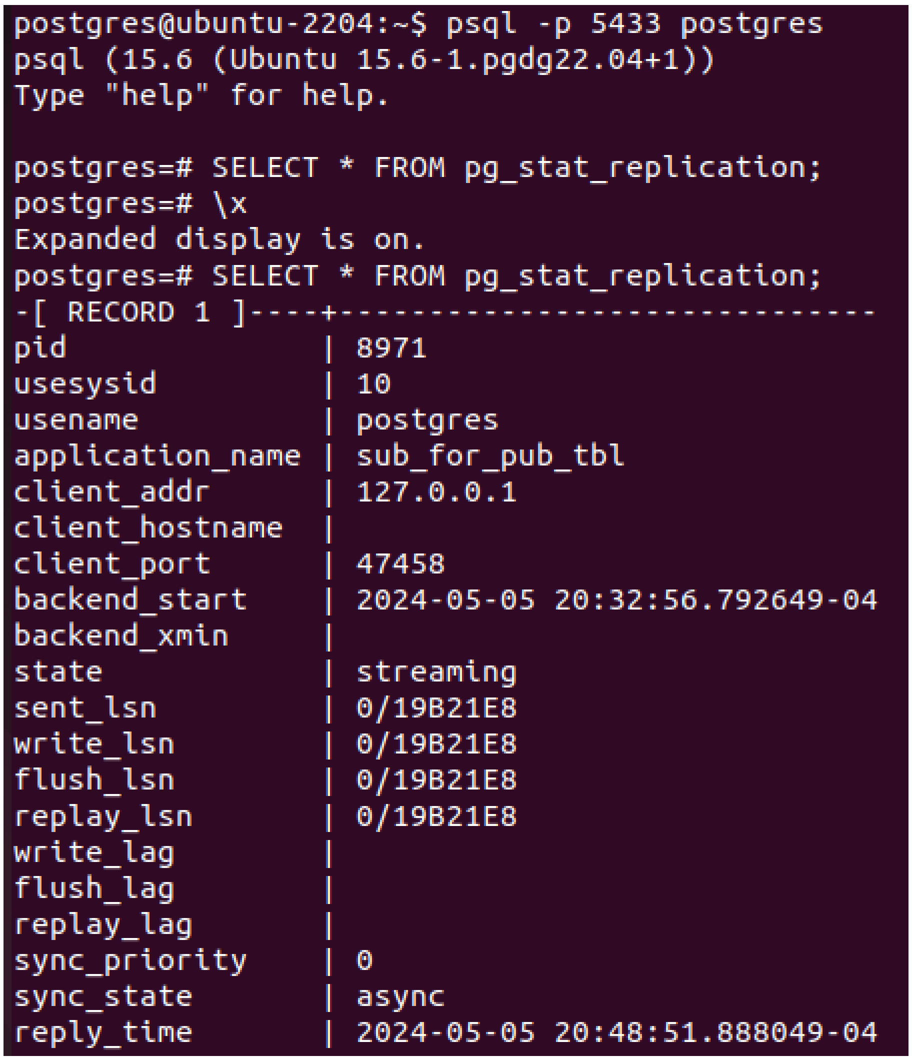
```
pub_db=# ALTER TABLE pub_tbl ADD COLUMN points INT;
ALTER TABLE
pub_db=# INSERT INTO pub_tbl (id, name, age, points) VALUES
(4, 'Piotrek', 23, 80),
(5, 'Wojtek', 28, 100),
(6, 'Filip', 27, 90);
INSERT 0 3
pub_db=# SELECT * FROM pub_tbl ;
 id |  name    | age | points
----+---------+-----+--------
  1 | Bartek   |  25 |
  2 | Janek    |  24 |
  3 | Marcin   |  30 |
  4 | Piotrek  |  23 |     80
  5 | Wojtek   |  28 |    100
  6 | Filip    |  27 |     90
(6 rows)
```



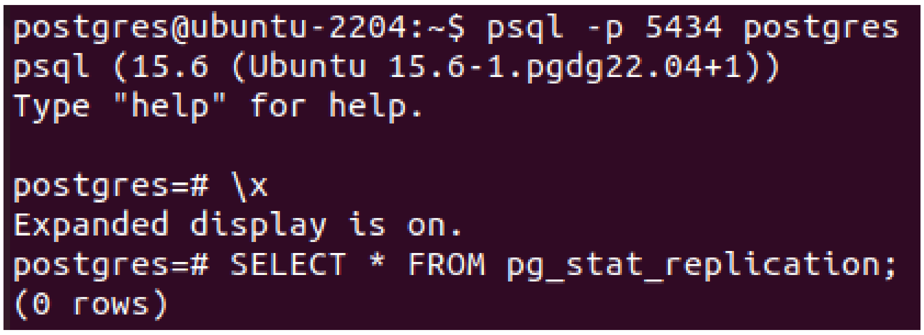
```
postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pub_tbl ;
 id |  name    | age | points
----+---------+-----+--------
  1 | Bartek   |  25 |
  2 | Janek    |  24 |
  3 | Marcin   |  30 |
  4 | Piotrek  |  23 |     80
  5 | Wojtek   |  28 |    100
  6 | Filip    |  27 |     90
(6 rows)
```

ix. Wyciagnij korzystając z widoku pg_stat_replication podstawowe parametry opisujące zkonfigurowaną replikacje. Porównaj dane odczytywane na instancji pierwszej oraz drugiej

Polecenie:



```
postgres@ubuntu-2204:~$ psql -p 5433 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.

postgres=# SELECT * FROM pg_stat_replication;
postgres=# \x
Expanded display is on.
postgres=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+-----------------------------
pid              | 8971
usesysid         | 10
usename          | postgres
application_name | sub_for_pub_tbl
client_addr      | 127.0.0.1
client_hostname  |
client_port      | 47458
backend_start    | 2024-05-05 20:32:56.792649-04
backend_xmin     |
state            | streaming
sent_lsn         | 0/19B21E8
write_lsn        | 0/19B21E8
flush_lsn        | 0/19B21E8
replay_lsn       | 0/19B21E8
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2024-05-05 20:48:51.888049-04
```

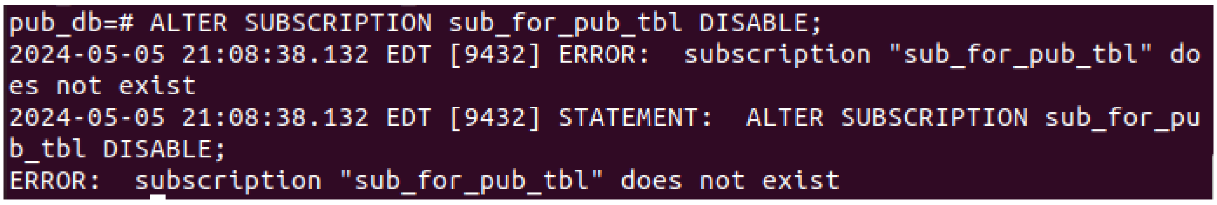


```
postgres@ubuntu-2204:~$ psql -p 5434 postgres
psql (15.6 (Ubuntu 15.6-1.pgdg22.04+1))
Type "help" for help.

postgres=# \x
Expanded display is on.
postgres=# SELECT * FROM pg_stat_replication;
(0 rows)
```

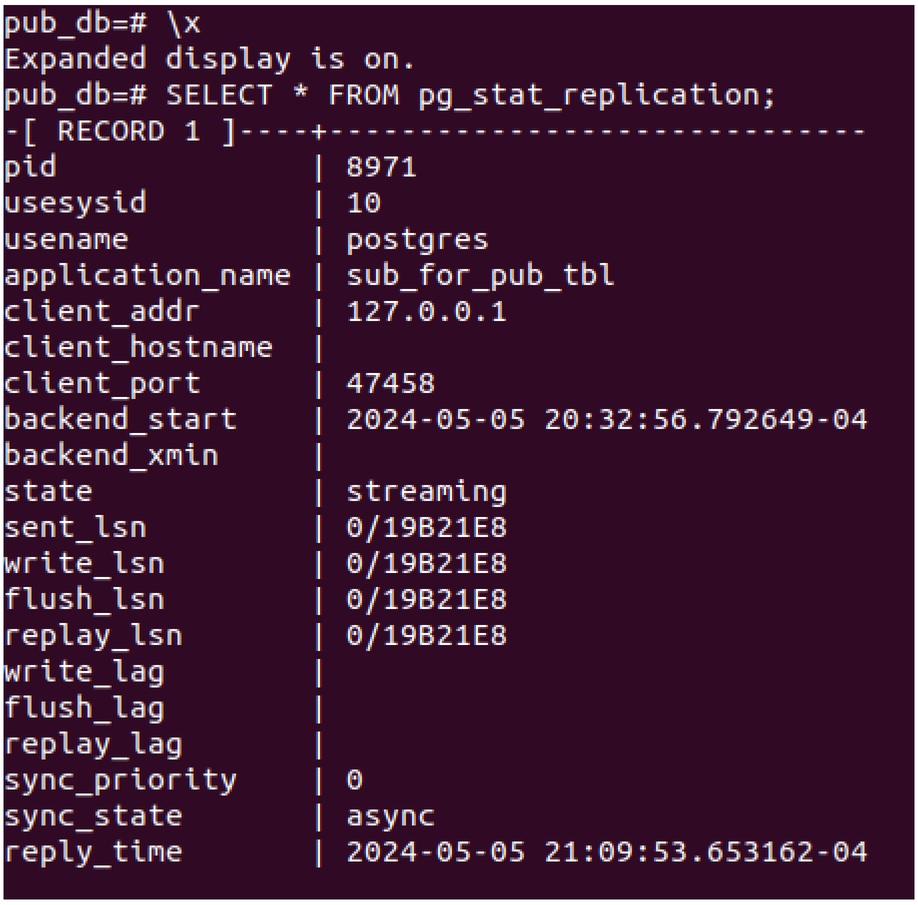
Obserwując wyniki można dojść do wniosku, że dane (parametry) skonfigurowanej replikacji są przechowywane tylko w pierwszej instancji.

x. Zatrzymaj subskrypcje (alter subscription) – pobierz ponownie dane opisujące proces replikacji na instancji pierwszej

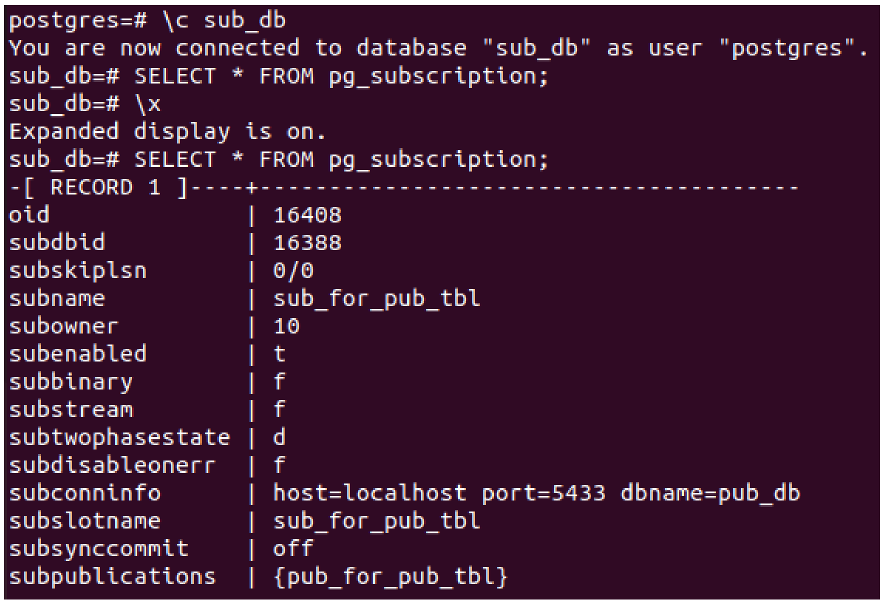
Polecenie:



```
pub_db=# ALTER SUBSCRIPTION sub_for_pub_tbl DISABLE;
2024-05-05 21:08:38.132 EDT [9432] ERROR:  subscription "sub_for_pub_tbl" do
es not exist
2024-05-05 21:08:38.132 EDT [9432] STATEMENT:  ALTER SUBSCRIPTION sub_for_pu
b_tbl DISABLE;
ERROR:  subscription "sub_for_pub_tbl" does not exist
```



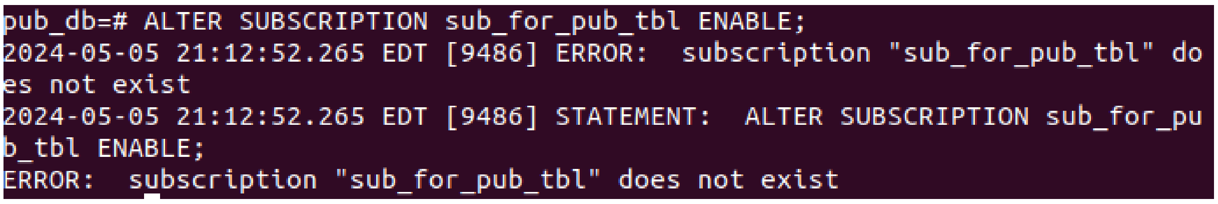
```
pub_db=# \x
Expanded display is on.
pub_db=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 8971
usesysid         | 10
usename          | postgres
application_name | sub_for_pub_tbl
client_addr      | 127.0.0.1
client_hostname  |
client_port      | 47458
backend_start    | 2024-05-05 20:32:56.792649-04
backend_xmin     |
state            | streaming
sent_lsn         | 0/19B21E8
write_lsn        | 0/19B21E8
flush_lsn        | 0/19B21E8
replay_lsn       | 0/19B21E8
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2024-05-05 21:09:53.653162-04
```



```
postgres=# \c sub_db
You are now connected to database "sub_db" as user "postgres".
sub_db=# SELECT * FROM pg_subscription;
sub_db=# \x
Expanded display is on.
sub_db=# SELECT * FROM pg_subscription;
-[ RECORD 1 ]----+---------------------------------------
oid              | 16408
subdbid          | 16388
subskiplsn       | 0/0
subname          | sub_for_pub_tbl
subowner         | 10
subenabled       | t
subbinary        | f
substream        | f
subtwophasestate | d
subdisableonerr  | f
subconninfo      | host=localhost port=5433 dbname=pub_db
subslotname      | sub_for_pub_tbl
subsynccommit    | off
subpublications  | {pub_for_pub_tbl}
```

xi. Uruchom ponownie subskrypcje

Polecenie:



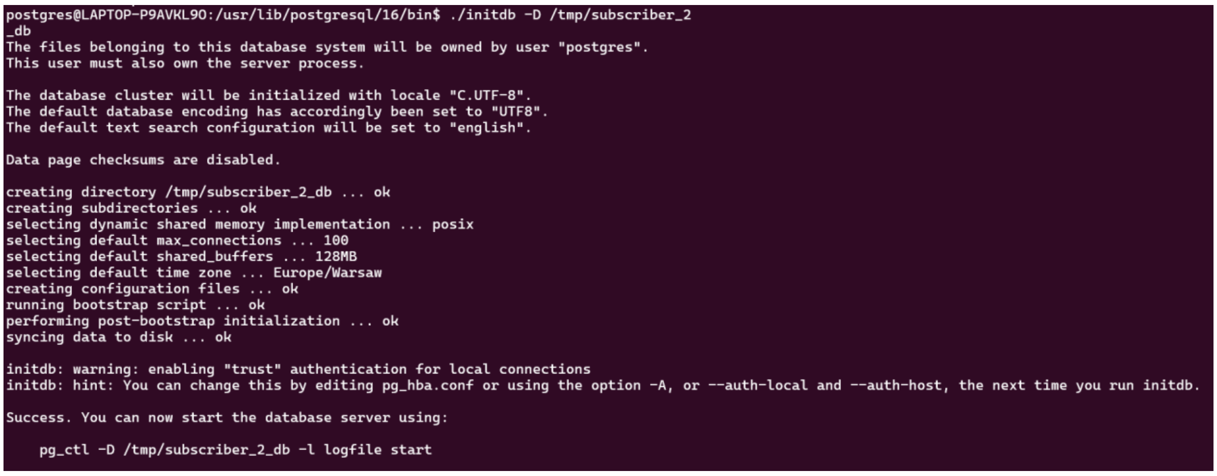
```
pub_db=# ALTER SUBSCRIPTION sub_for_pub_tbl ENABLE;
2024-05-05 21:12:52.265 EDT [9486] ERROR:  subscription "sub_for_pub_tbl" do
es not exist
2024-05-05 21:12:52.265 EDT [9486] STATEMENT:  ALTER SUBSCRIPTION sub_for_pu
b_tbl ENABLE;
ERROR:  subscription "sub_for_pub_tbl" does not exist
```

Faza VII Rozszerzenie konfiguracji

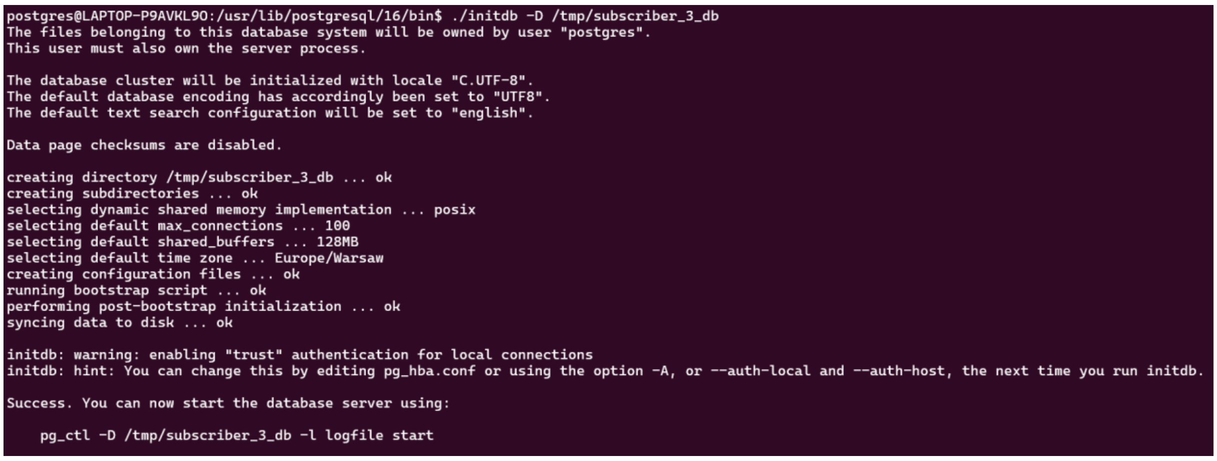
a. Rozszerz konfigurację o dwa dodatkowe serwery replikacji (subscribery) i skonfiguruj całość tak aby replikacja odbywała się:

i. Bezpośrednio z instancji publisher do wszystkich serwerów zapasowych

Utworzenie dwóch dodatkowych serwerów replikacji:

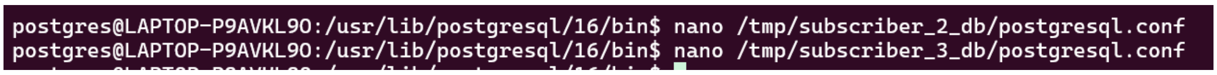


```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./initdb -D /tmp/subscriber_2
_db
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /tmp/subscriber_2_db ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Warsaw
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /tmp/subscriber_2_db -l logfile start
```



```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./initdb -D /tmp/subscriber_3_db
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /tmp/subscriber_3_db ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Warsaw
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /tmp/subscriber_3_db -l logfile start
```

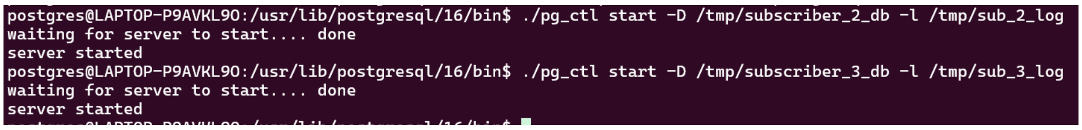
W poleceniach:



```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ nano /tmp/subscriber_2_db/postgresql.conf
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ nano /tmp/subscriber_3_db/postgresql.conf
```

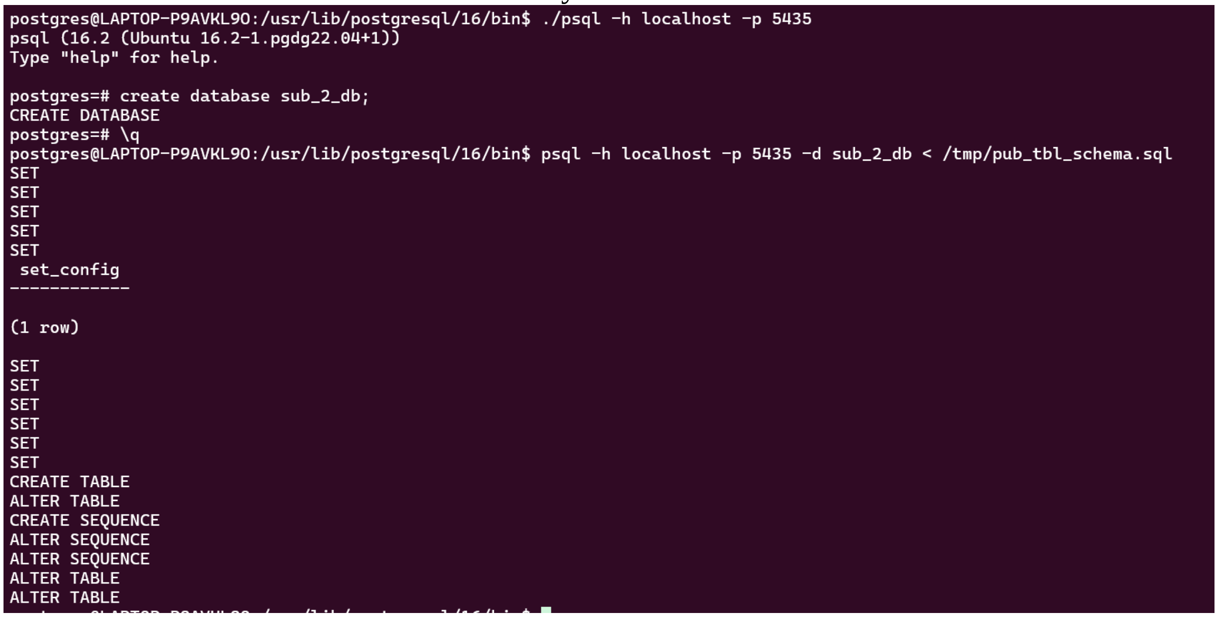
Porty zmienione odpowiednio na 5435 i 5436.

Start serwerów:



```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./pg_ctl start -D /tmp/subscriber_2_db -l /tmp/sub_2_log
waiting for server to start.... done
server started
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./pg_ctl start -D /tmp/subscriber_3_db -l /tmp/sub_3_log
waiting for server to start.... done
server started
```

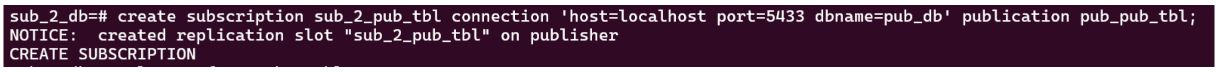
Przeniesienie schematu tabeli na subscribery:



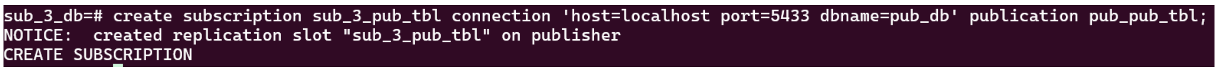
```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./psql -h localhost -p 5435
psql (16.2 (Ubuntu 16.2-1.pgdg22.04+1))
Type "help" for help.

postgres=# create database sub_2_db;
CREATE DATABASE
postgres=# \q
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ psql -h localhost -p 5435 -d sub_2_db < /tmp/pub_tbl_schema.sql
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
```

Analogicznie dla sub_3_db (port 5436)

Utworzenie subskrypcji od tych serwerów do publishera:

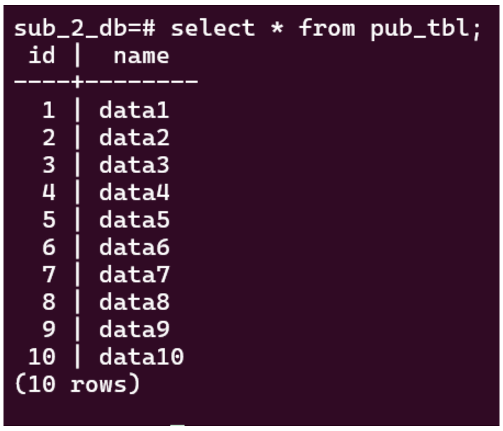


```
sub_2_db=# create subscription sub_2_pub_tbl connection 'host=localhost port=5433 dbname=pub_db' publication pub_pub_tbl;
NOTICE:  created replication slot "sub_2_pub_tbl" on publisher
CREATE SUBSCRIPTION
```



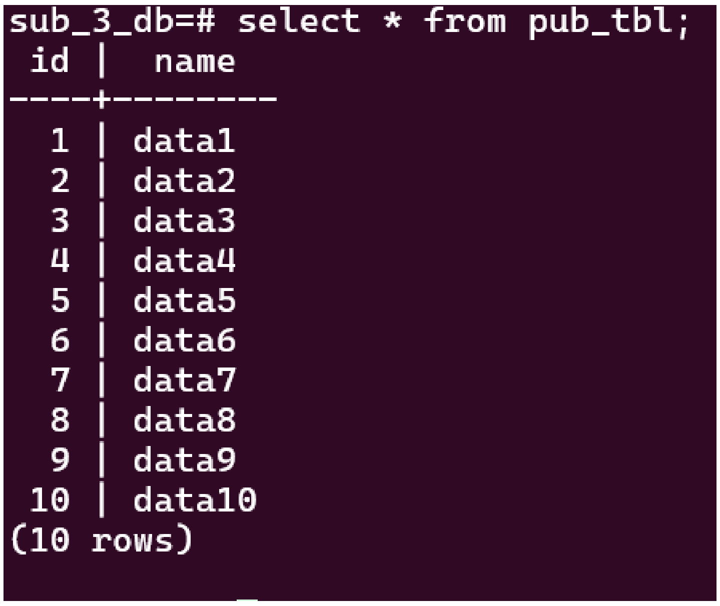
```
sub_3_db=# create subscription sub_3_pub_tbl connection 'host=localhost port=5433 dbname=pub_db' publication pub_pub_tbl;
NOTICE:  created replication slot "sub_3_pub_tbl" on publisher
CREATE SUBSCRIPTION
```

Potwierdzenie działania



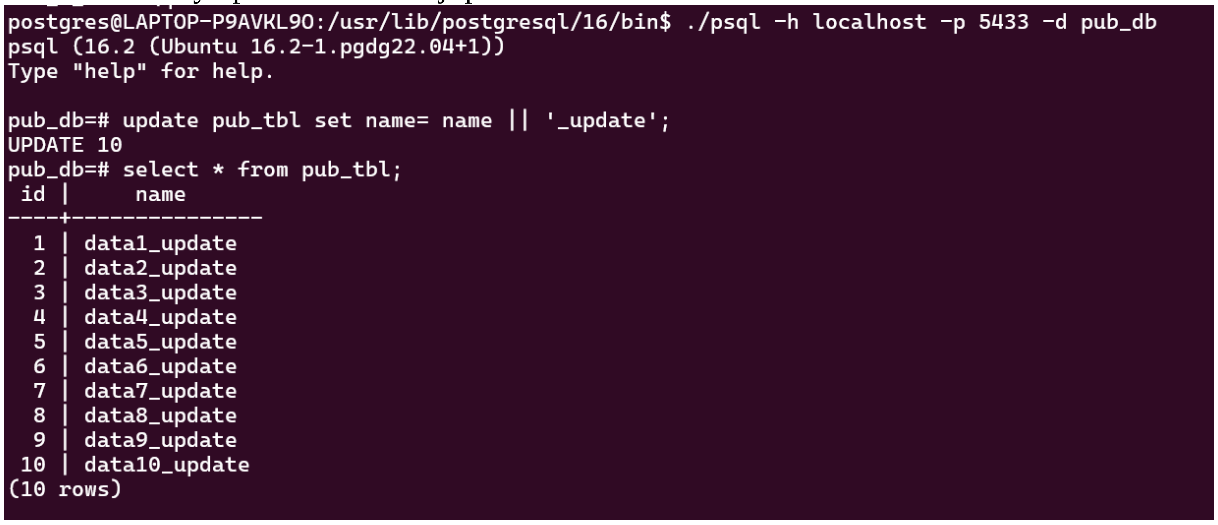
```
sub_2_db=# select * from pub_tbl;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
(10 rows)
```

I dla sub_3_db:



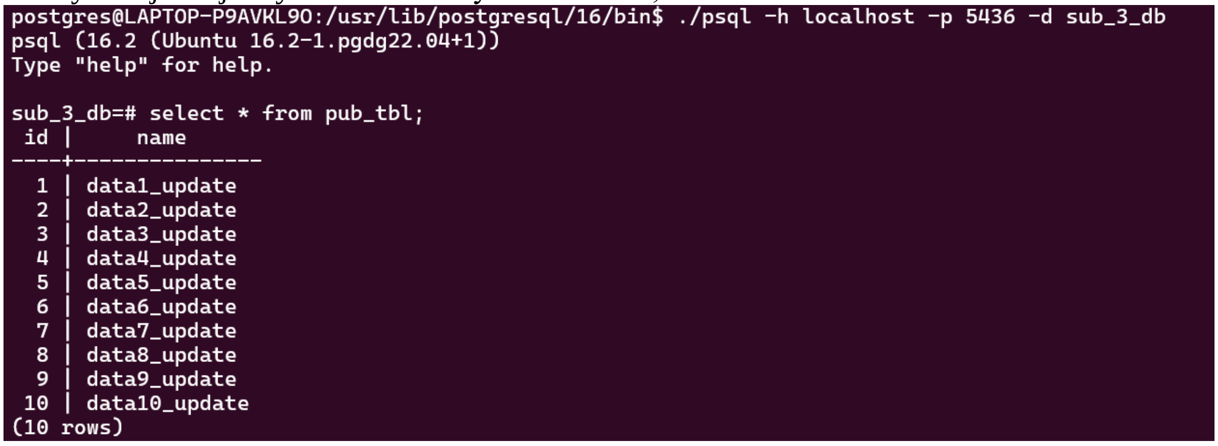
```
sub_3_db=# select * from pub_tbl;
 id |  name
----+--------
  1 | data1
  2 | data2
  3 | data3
  4 | data4
  5 | data5
  6 | data6
  7 | data7
  8 | data8
  9 | data9
 10 | data10
(10 rows)
```

Testowy update na instancji publishera:



```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./psql -h localhost -p 5433 -d pub_db
psql (16.2 (Ubuntu 16.2-1.pgdg22.04+1))
Type "help" for help.

pub_db=# update pub_tbl set name= name || '_update';
UPDATE 10
pub_db=# select * from pub_tbl;
 id |     name
----+---------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
(10 rows)
```

I weryfikacja na jednym z dodatkowych serwerów;

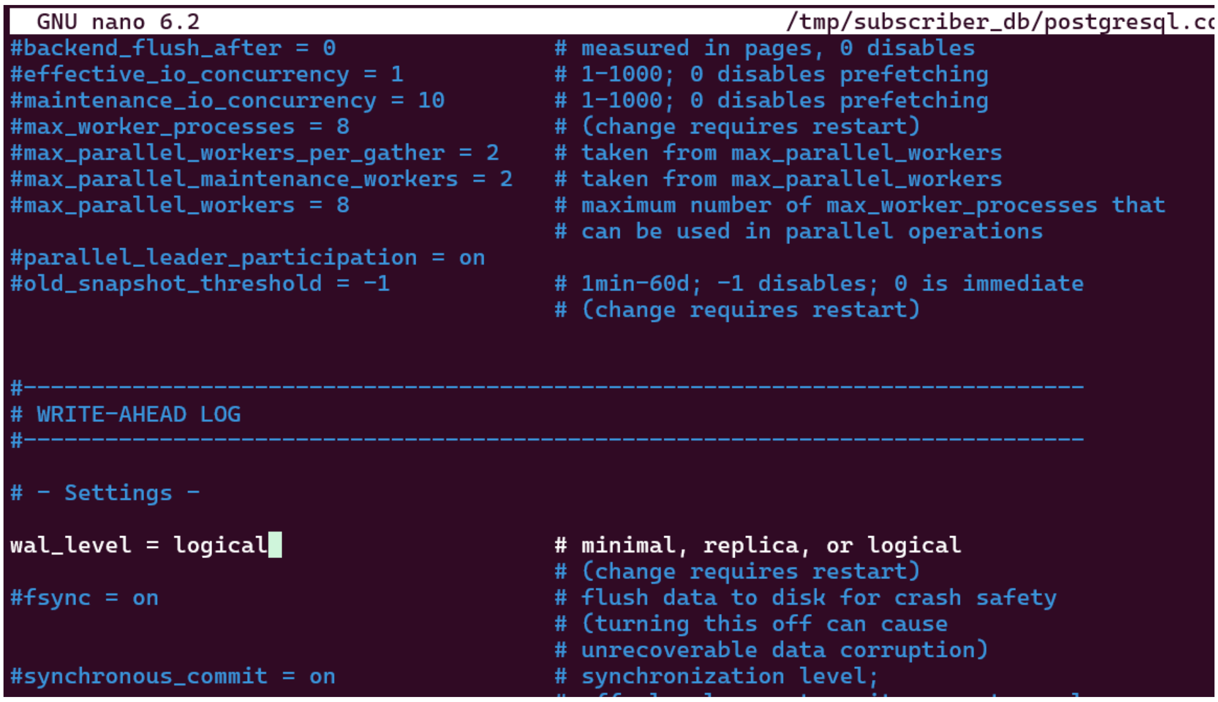


```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./psql -h localhost -p 5436 -d sub_3_db
psql (16.2 (Ubuntu 16.2-1.pgdg22.04+1))
Type "help" for help.

sub_3_db=# select * from pub_tbl;
 id |     name
----+---------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
(10 rows)
```

ii. Kaskadowo z serwera primary do jednego z serwerów zapasowych, stamtąd do kolejnego serwera zapasowego etc (tryb cascade setup)

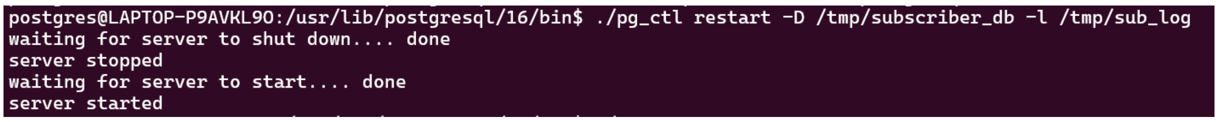
Stworzymy na pierwszej utworzone instancji subskrybującej: subscriber_db (port 5434), publication dla tabeli, subscrybowanej z instancji publisher_db.

Do tego celu zmieniamy w pliku postgresql.conf wal_level = logical :



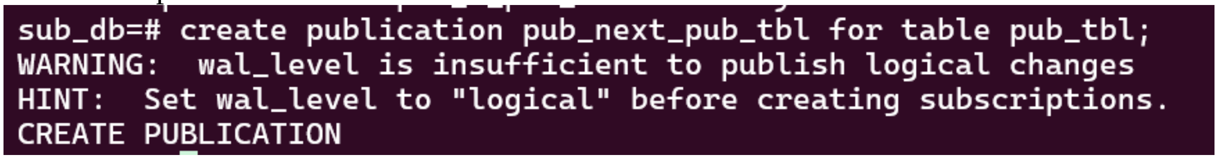
```
  GNU nano 6.2                                         /tmp/subscriber_db/postgresql.co
#backend_flush_after = 0                # measured in pages, 0 disables
#effective_io_concurrency = 1           # 1-1000; 0 disables prefetching
#maintenance_io_concurrency = 10        # 1-1000; 0 disables prefetching
#max_worker_processes = 8               # (change requires restart)
#max_parallel_workers_per_gather = 2    # taken from max_parallel_workers
#max_parallel_maintenance_workers = 2   # taken from max_parallel_workers
#max_parallel_workers = 8               # maximum number of max_worker_processes that
                                        # can be used in parallel operations
#parallel_leader_participation = on
#old_snapshot_threshold = -1            # 1min-60d; -1 disables; 0 is immediate
                                        # (change requires restart)


#------------------------------------------------------------------------------
# WRITE-AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = logical                     # minimal, replica, or logical
                                        # (change requires restart)
#fsync = on                             # flush data to disk for crash safety
                                        # (turning this off can cause
                                        # unrecoverable data corruption)
#synchronous_commit = on                # synchronization level;
```

I następnie restart serwera by załadować zmiany:



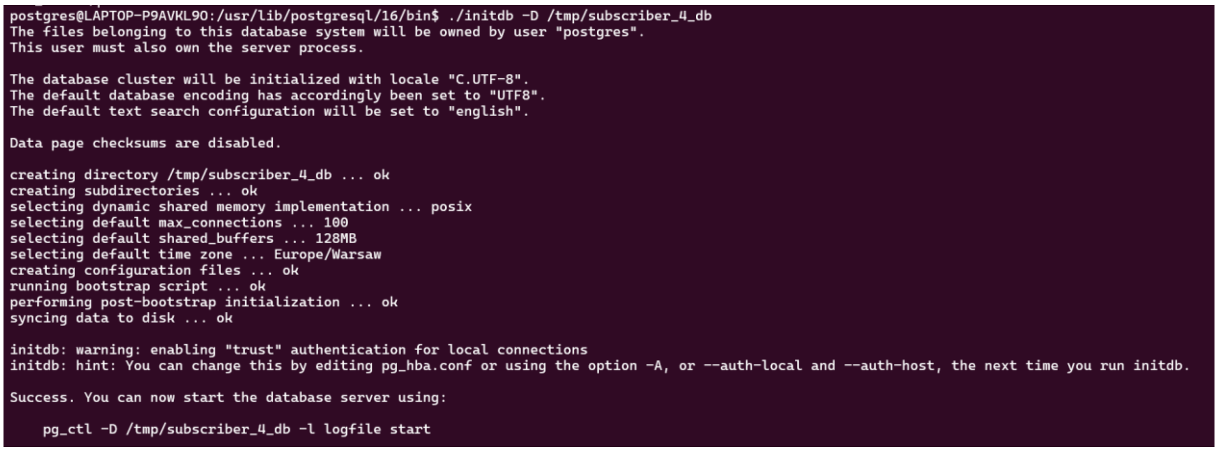
```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./pg_ctl restart -D /tmp/subscriber_db -l /tmp/sub_log
waiting for server to shut down.... done
server stopped
waiting for server to start.... done
server started
```

Stworzenie publication:



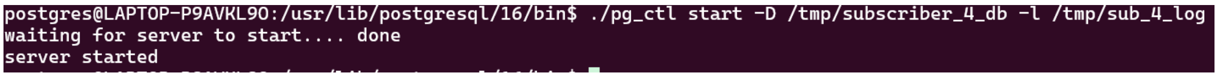
```
sub_db=# create publication pub_next_pub_tbl for table pub_tbl;
WARNING:  wal_level is insufficient to publish logical changes
HINT:  Set wal_level to "logical" before creating subscriptions.
CREATE PUBLICATION
```

Następnie tworzymy kolejną instancję, która będzie subskrybować kaskadowo z tej powyższej:



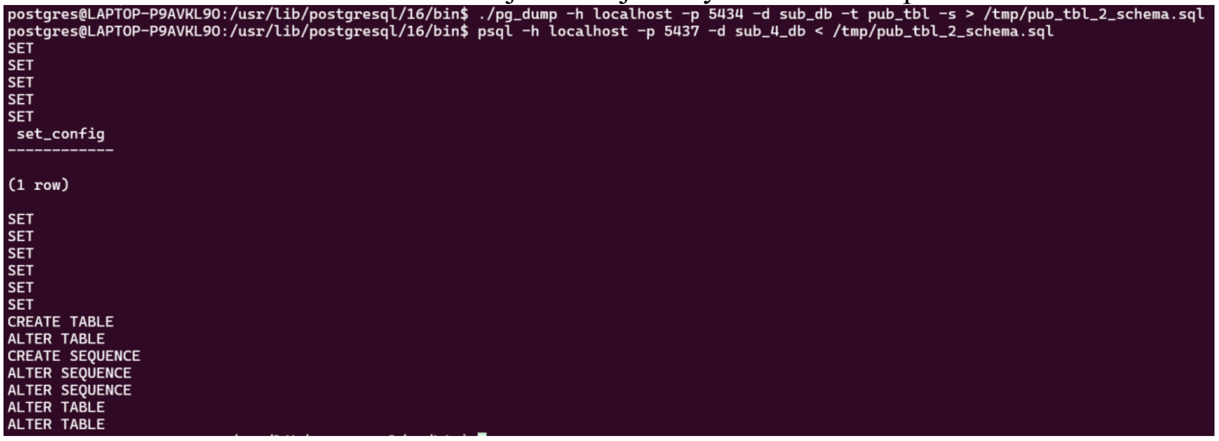
```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./initdb -D /tmp/subscriber_4_db
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /tmp/subscriber_4_db ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Warsaw
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /tmp/subscriber_4_db -l logfile start
```

Port 5347 – Start serwera:



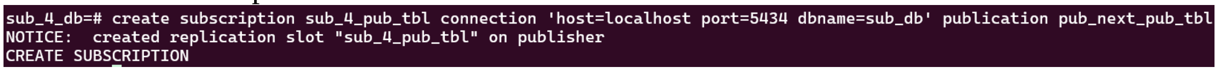
```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./pg_ctl start -D /tmp/subscriber_4_db -l /tmp/sub_4_log
waiting for server to start.... done
server started
```

Zrzut i załadowanie schematu z instancji z której mamy kaskadowo replikować:



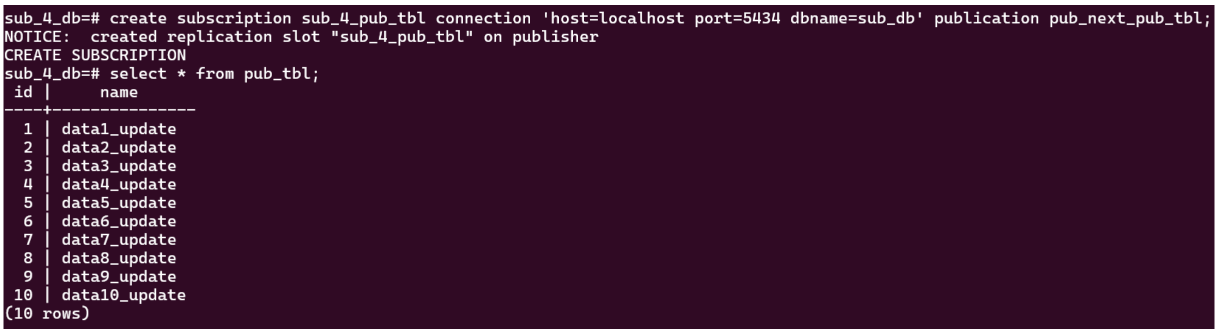
```
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ ./pg_dump -h localhost -p 5434 -d sub_db -t pub_tbl -s > /tmp/pub_tbl_2_schema.sql
postgres@LAPTOP-P9AVKL9O:/usr/lib/postgresql/16/bin$ psql -h localhost -p 5437 -d sub_4_db < /tmp/pub_tbl_2_schema.sql
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER TABLE
ALTER TABLE
```

Utworzenie subscription:



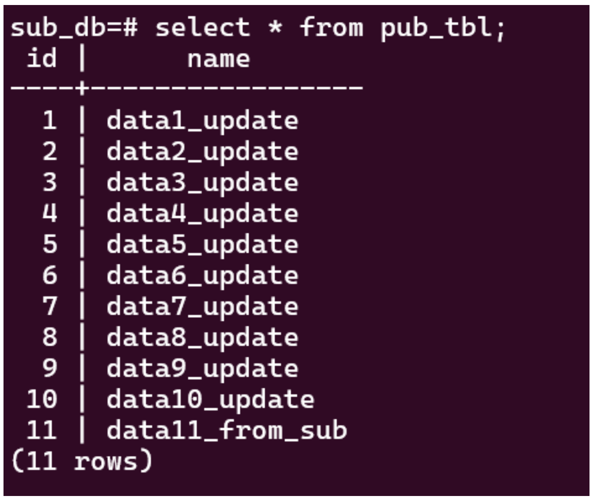
```
sub_4_db=# create subscription sub_4_pub_tbl connection 'host=localhost port=5434 dbname=sub_db' publication pub_next_pub_tbl
NOTICE:  created replication slot "sub_4_pub_tbl" on publisher
CREATE SUBSCRIPTION
```

Potwierdzenie działania:



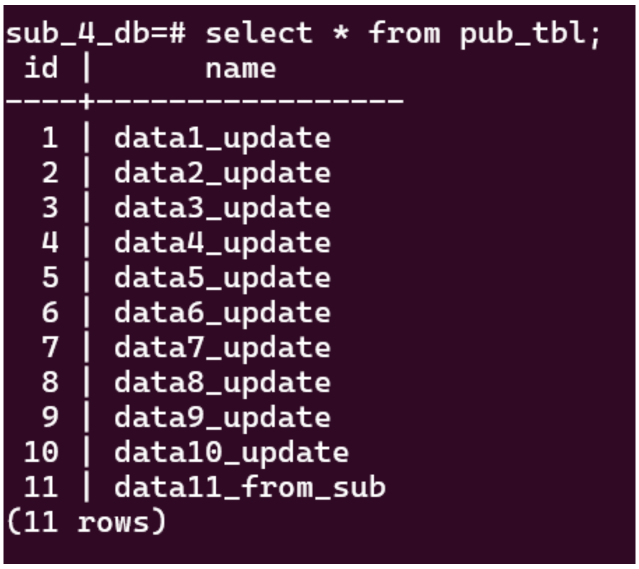
```
sub_4_db=# create subscription sub_4_pub_tbl connection 'host=localhost port=5434 dbname=sub_db' publication pub_next_pub_tbl;
NOTICE:  created replication slot "sub_4_pub_tbl" on publisher
CREATE SUBSCRIPTION
sub_4_db=# select * from pub_tbl;
 id |     name
----+---------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
(10 rows)
```

 Testowo dodano również kolumnę na subscriberze, który jest zarazem publisherem dla naszej nowej instancji.



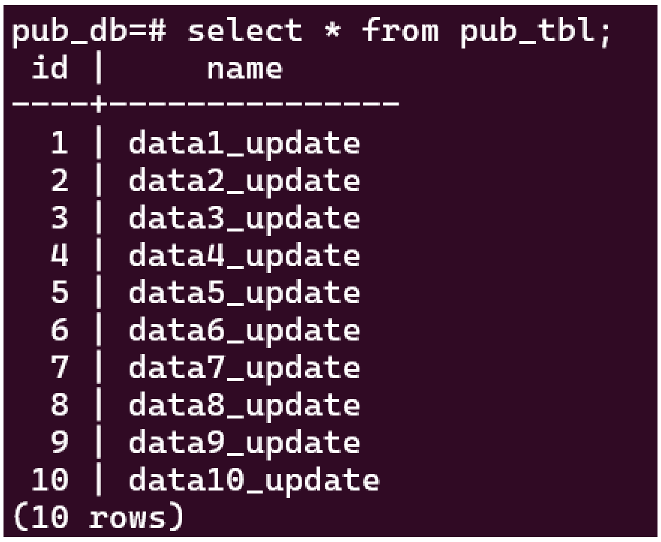
```
sub_db=# select * from pub_tbl;
 id |      name
----+------------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
 11 | data11_from_sub
(11 rows)
```

Rekord widoczny dla nowej instancji:



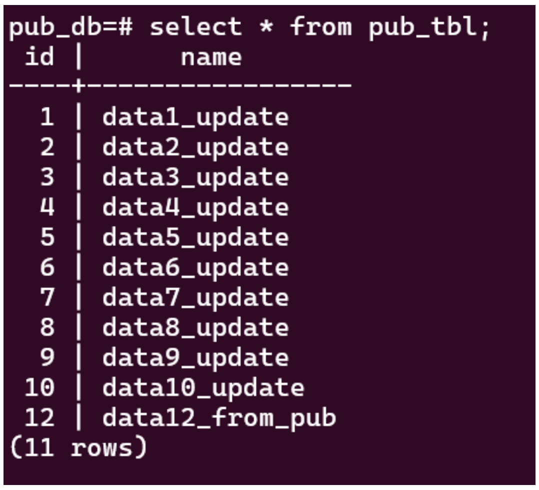
```
sub_4_db=# select * from pub_tbl;
 id |      name
----+-----------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
 11 | data11_from_sub
(11 rows)
```

Ale nie widoczny dla publishera będącego poziom wyżej:



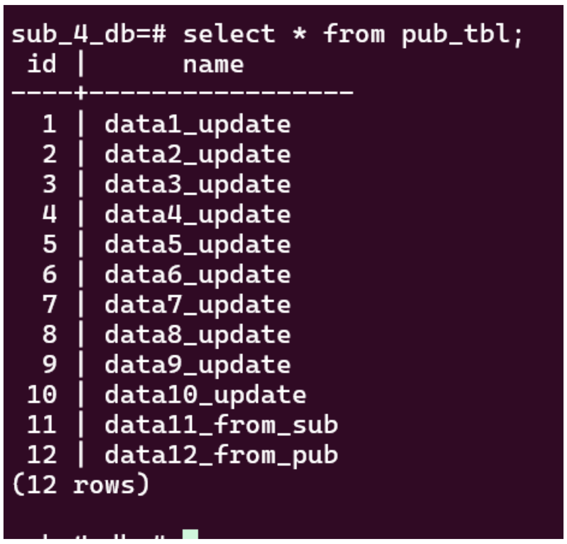
```
pub_db=# select * from pub_tbl;
 id |     name
----+---------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
(10 rows)
```

Ale po dodaniu rekordu na publisherze (najwyższy poziom kaskadowej subskrypcji):



```
pub_db=# select * from pub_tbl;
 id |      name
----+-----------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
 12 | data12_from_pub
(11 rows)
```

kaskadowo przechodzi on do instancji subskrybującej na drugim poziomie, najniższym:



```
sub_4_db=# select * from pub_tbl;
 id |      name
----+-----------------
  1 | data1_update
  2 | data2_update
  3 | data3_update
  4 | data4_update
  5 | data5_update
  6 | data6_update
  7 | data7_update
  8 | data8_update
  9 | data9_update
 10 | data10_update
 11 | data11_from_sub
 12 | data12_from_pub
(12 rows)
```