

1. Cel i zakres zadania projektowego:

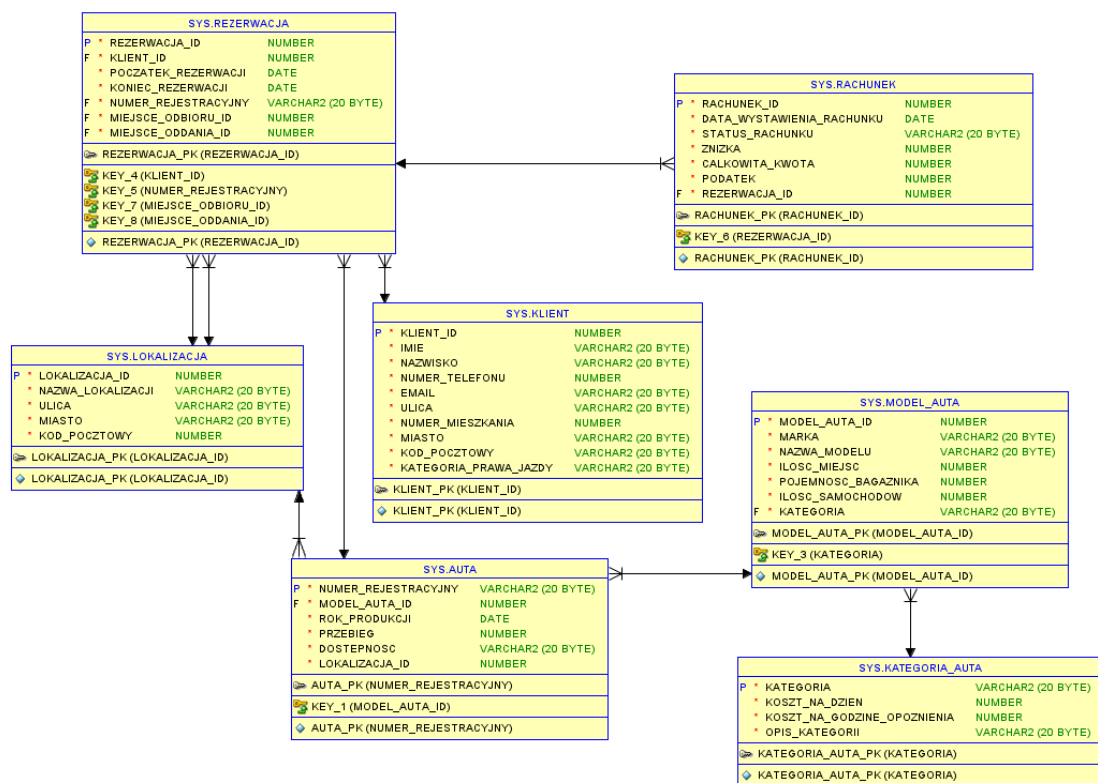
Celem naszego projektu było stworzenie pierwotnie scentralizowanej bazy danych wypożyczalni samochodów, a następnie jej rozproszenie na dwie bazy danych działające pomiędzy bazą na systemie operacyjnym Windows oraz bazą na systemie operacyjnym Linux z wykorzystaniem maszyny wirtualnej.

2. Przyjęte założenia podczas realizacji:

3. Przyjęte założenia podczas realizacji:

4. Opis implementacji baz danych:

- Diagram UML przygotowany przez nas za pomocą programu SQL Developer prezentuje się następująco:



Rysunek 1. Diagram UML naszej scentralizowanej bazy danych

Nasz diagram reprezentuje siedem tabel składających się łącznie na nasz schemat scentralizowanej bazy danych przed jej rozproszeniem. Na diagramie zaznaczone są primery keys, czyli klucze główne danej tabeli oraz „foreign keys”, czyli klucze obce, będące kluczami głównymi innych tabel.

- Implementację naszej bazy danych zaczynamy od pobrania środowiska SQL Developer, a także od pobrania Oracle DB Developer VM działającego na systemie Linux do naszej maszyny wirtualnej, na której planujemy stworzyć naszą drugą bazę danych.

Ponieważ istniał problem związany z instalacją Oracle na Windowsie politechnicznym udało nam się zainstalować program dopiero na innym urządzeniu.

Po szczęśliwym pobraniu i uruchomieniu SQL Developer w pierwszej kolejności musieliśmy stworzyć nowe połączenie w naszej bazie. Czynimy to w następujący sposób:

Rysunek 2. Połączenie z bazą Oracle na Windowsie

Na rysunku 1. zaprezentowano sposób połączenia się z bazą Oracle za pomocą SQL Developer. W pierwszej kolejności dodajemy nazwę naszego połączenia, w naszym przypadku jest to RENTALCARDATABASE. Następnie używamy nazwy użytkownika „sys” oraz hasła „admin” oraz wybieramy Role „SYSDBA”. W details jako hostname sieci wybieramy nasz localhost i port 1521 (na niektórych komputerach domyślnie występuje port 1522). Następnie jako SID wybieramy domyślne XE. Po tych czynnościach pomyślnie udało nam się połączyć z bazą.

Po pomyślnym połączeniu się z bazą przystępujemy do tworzenia pierwszej tabeli. Pierwszą utworzoną przez nas tabelą zostanie tabela Auta, która zawiera ważne dla wypożyczalni aut informacje o numerach rejestracyjnych wypożyczanych samochodów, czy też o numerze ID tego auta. W pierwszej kolejności tworzymy kolumny naszej tabeli auta oraz typ danych jaki chcemy w nich przechowywać, a następnie dodajemy dane do tabeli. Wybieramy też primary key tabeli Auta.

	↕ COLUMN_NAME	↕ DATA_TYPE	↕ NULLABLE	DATA_DEFAULT	↕ COLUMN_ID	↕ COMMENTS
1	NUMER_REJESTRACYJNY	VARCHAR2 (20 BYTE)	No	(null)	1 (null)	
2	MODEL_AUTA_ID	NUMBER	No	(null)	2 (null)	
3	ROK_PRODUKCJI	DATE	No	(null)	3 (null)	
4	PRZEBIEG	NUMBER	No	(null)	4 (null)	
5	DOSTEPNOSC	VARCHAR2 (20 BYTE)	No	(null)	5 (null)	
6	LOKALIZACJA_ID	NUMBER	No	(null)	6 (null)	

Rysunek 3. Kolumny tabeli Auta

	↕ NUMER_REJESTRACYJNY	↕ MODEL_AUTA_ID	↕ ROK_PRODUKCJI	↕ PRZEBIEG	↕ DOSTEPNOSC	↕ LOKALIZACJA_ID
1	FSU7531		1 05/12/11	200000	DOSTEPNY	2
2	SR62592		2 05/12/17	180000	DOSTEPNY	2
3	WRO2893		41 15/12/17	130000	DOSTEPNY	2
4	WRO9803		3 17/12/09	100000	NIEDOSTEPNY	2
5	WRO1234		41 15/12/18	140000	DOSTEPNY	2
6	WRO4387		2 05/12/03	239000	DOSTEPNY	2

Rysunek 4. Dane przypisane do kolumn tabeli Auta

Kod programu tworzącego tabelę wygląda następująco:

```

1 CREATE TABLE AUTA
2 (
3     NUMER_REJESTRACYJNY VARCHAR2(20 BYTE) NOT NULL
4     , MODEL_AUTA_ID NUMBER NOT NULL
5     , ROK_PRODUKCJI DATE NOT NULL
6     , PRZEBIEG NUMBER NOT NULL
7     , DOSTEPNOSC VARCHAR2(20 BYTE) NOT NULL
8     , LOKALIZACJA_ID NUMBER NOT NULL
9 )
10 LOGGING
11 TABLESPACE SYSTEM
12 PCTFREE 10
13 PCTUSED 40
14 INITRANS 1
15 STORAGE
16 (
17     INITIAL 65536
18     NEXT 1048576
19     MINEXTENTS 1
20     MAXEXTENTS UNLIMITED
21     FREELISTS 1
22     FREELIST GROUPS 1
23     BUFFER_POOL DEFAULT
24 )
25 NOCOMPRESS
26 NO INMEMORY
27 NOPARALLEL;
28
29 CREATE UNIQUE INDEX AUTA_PK ON AUTA (NUMER_REJESTRACYJNY ASC)
30 LOGGING
31 TABLESPACE SYSTEM
32 PCTFREE 10
33 INITRANS 2
34 STORAGE
35 (
36     INITIAL 65536
37     NEXT 1048576
38     MINEXTENTS 1
39     MAXEXTENTS UNLIMITED
40     FREELISTS 1
41     FREELIST GROUPS 1
42     BUFFER_POOL DEFAULT
43 )
44 NOPARALLEL;
45
46 ALTER TABLE AUTA
47 ADD CONSTRAINT AUTA_PK PRIMARY KEY
48 (
49     NUMER_REJESTRACYJNY
50 )
51 USING INDEX SYS.AUTA_PK
52 ENABLE;
53
54 ALTER TABLE AUTA
55 ADD CONSTRAINT KEY_1 FOREIGN KEY
56 (
57     MODEL_AUTA_ID
58 )
59 REFERENCES MODEL_AUTA1
60 (
61     MODEL_AUTA_ID
62 )
63 ENABLE;

```

Rysunek 5. Kod programu tworzącego tabelę Auta

Analogicznie w kolejnych krokach tworzymy tabele dla naszej bazy lokalnej na systemie Windows: Klient, Lokalizacja, Kategoria_Auta, Model_Auta, Rachunek, Rezerwacja.

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	KLIENT_ID	NUMBER	No	(null)	1 (null)	
2	IMIE	VARCHAR2 (20 BYTE)	No	(null)	2 (null)	
3	NAZWISKO	VARCHAR2 (20 BYTE)	No	(null)	3 (null)	
4	NUMER_TELEFONU	NUMBER	No	(null)	4 (null)	
5	EMAIL	VARCHAR2 (20 BYTE)	No	(null)	5 (null)	
6	ULICA	VARCHAR2 (20 BYTE)	No	(null)	6 (null)	
7	NUMER_MIESZKANIA	NUMBER	No	(null)	7 (null)	
8	MIASTO	VARCHAR2 (20 BYTE)	No	(null)	8 (null)	
9	KOD_POCZTOWY	VARCHAR2 (20 BYTE)	No	(null)	9 (null)	
10	KATEGORIA_PRAWA_JAZDY	VARCHAR2 (20 BYTE)	No	(null)	10 (null)	

Rysunek 6. Kolumny tabeli Klient

	KLIENT_ID	IMIE	NAZWISKO	NUMER_TELEFONU	EMAIL	ULICA	NUMER_MIESZKANIA	MIASTO	KOD_POCZTOWY	KATEGORIA_PRAWA_JAZDY
1	1111	Ma...	Musiał	989723896	Mariusz692@gmail.com	Śląska		20 Wrocław	50-001	B
2	2121	KA...	WIECKOWSKI	909876512	KAZ97@gmail.com	SKŁ...		18 OLAWA	55-200	B
3	3131	Emil	Haim	849087263	EMI909@gmail.com	Rozana		9 Wrocław	50-001	B
4	4141	Pa...	Michalski	907394758	patryk.mi@gmail.com	Zło...		4 Wrocław	50-001	B

Rysunek 7. Dane w tabeli Klient

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LOKALIZACJA_ID	NUMBER	No	(null)	1 (null)	
2	NAZWA_LOKALIZACJI	VARCHAR2 (20 BYTE)	No	(null)	2 (null)	
3	ULICA	VARCHAR2 (20 BYTE)	No	(null)	3 (null)	
4	MIASTO	VARCHAR2 (20 BYTE)	No	(null)	4 (null)	
5	KOD_POCZTOWY	VARCHAR2 (20 BYTE)	Yes	(null)	5 (null)	

Rysunek 8. Kolumny tabeli Lokalizacja

	LOKALIZACJA_ID	NAZWA_LOKALIZACJI	ULICA	MIASTO	KOD_POCZTOWY
1	2	PUNKT_2	CHROBREGO	KATOWICE	40-000

Rysunek 9. Dane w tabeli Lokalizacja (Możemy zaobserwować, że lokalizacja jest tylko jedna i ma ID = 2, to wynika z faktu, że nasza druga baza jest naszą drugą lokalizacją)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	KATEGORIA	VARCHAR2 (20 BYTE)	No	(null)	1 (null)	
2	KOSZT_NA_DZIEŃ	NUMBER	No	(null)	2 (null)	
3	KOSZT_NA_GODZINĘ_OPOZNIENIA	NUMBER	No	(null)	3 (null)	
4	OPIS_KATEGORII	VARCHAR2 (20 BYTE)	No	(null)	4 (null)	

Rysunek 10. Kolumny w tabeli Kategoria_Auta

	KATEGORIA	KOSZT_NA_DZIEŃ	KOSZT_NA_GODZINĘ_OPOZNIENIA	OPIS_KATEGORII
1	B	200		20 MIEJSKIE
2	A	150		15 MINI
3	C	250		25 KOMPAKTOWE
4	D	350		35 RODZINNE

Rysunek 11. Dane tabeli *Kategoria_Auta*

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MODEL_AUTA_ID	NUMBER	No	(null)	1	(null)
2	MARKA	VARCHAR2 (20 BYTE)	No	(null)	2	(null)
3	NAZWA_MODELU	VARCHAR2 (20 BYTE)	No	(null)	3	(null)
4	ILOSC_MIEJSC	NUMBER	No	(null)	4	(null)
5	POJEMNOSC_BAGAZNIKA	NUMBER	No	(null)	5	(null)
6	ILOSC_SAMOCODOW	NUMBER	No	(null)	6	(null)
7	KATEGORIA	VARCHAR2 (20 BYTE)	No	(null)	7	(null)

Rysunek 12. Kolumny tabeli *Model_Auta*

	MODEL_AUTA_ID	MARKA	NAZWA_MODELU	ILOSC_MIEJSC	POJEMNOSC_BAGAZNIKA	ILOSC_SAMOCODOW	KATEGORIA
1	41	HYUNDAI	I30	5	600	2	D
2	42	OPEL	ASTRA	5	550	1	D
3	1	PEUGEOT	107	4	200	1	A
4	2	FIAT	PUNTO	4	200	2	B
5	3	MAZDA	3	5	400	1	C

Rysunek 13. Dane tabeli *Model_Auta*

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	RACHUNEK_ID	NUMBER	No	(null)	1	(null)
2	DATA_WYSTAWIENIA_RACHUNKU	DATE	No	(null)	2	(null)
3	STATUS_RACHUNKU	VARCHAR2 (20 BYTE)	No	(null)	3	(null)
4	ZNIZKA	NUMBER	No	(null)	4	(null)
5	CALKOWITA_KWOTA	NUMBER	No	(null)	5	(null)
6	PODATEK	NUMBER	No	(null)	6	(null)
7	REZERWACJA_ID	NUMBER	No	(null)	7	(null)

Rysunek 14. Kolumny tabeli *Rachunek*

	RACHUNEK_ID	DATA_WYSTAWIENIA_RACHUNKU	STATUS_RACHUNKU	ZNIZKA	CALKOWITA_KWOTA	PODATEK	REZERWACJA_ID
1	1234	22/12/09	ROZLICZONY	0	600	50	1111
2	1235	22/12/01	ROZLICZONY	0	1000	200	1112
3	1236	22/12/11	NIEROZLICZONY	0	1400	400	1113
4	1237	22/12/13	ROZLICZONY	0	1200	300	1114

Rysunek 15. Dane tabeli *Rachunek*

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 REZERWACJA_ID	NUMBER	No	(null)	1 (null)	
2 KLIENT_ID	NUMBER	No	(null)	2 (null)	
3 POCZATEK_REZERWACJI	DATE	No	(null)	3 (null)	
4 KONIEC_REZERWACJI	DATE	No	(null)	4 (null)	
5 NUMER_REJESTRACYJNY	VARCHAR2(20 BYTE)	No	(null)	5 (null)	
6 MIEJSCE_ODBIORU_ID	NUMBER	No	(null)	6 (null)	
7 MIEJSCE_ODDANIA_ID	NUMBER	No	(null)	7 (null)	

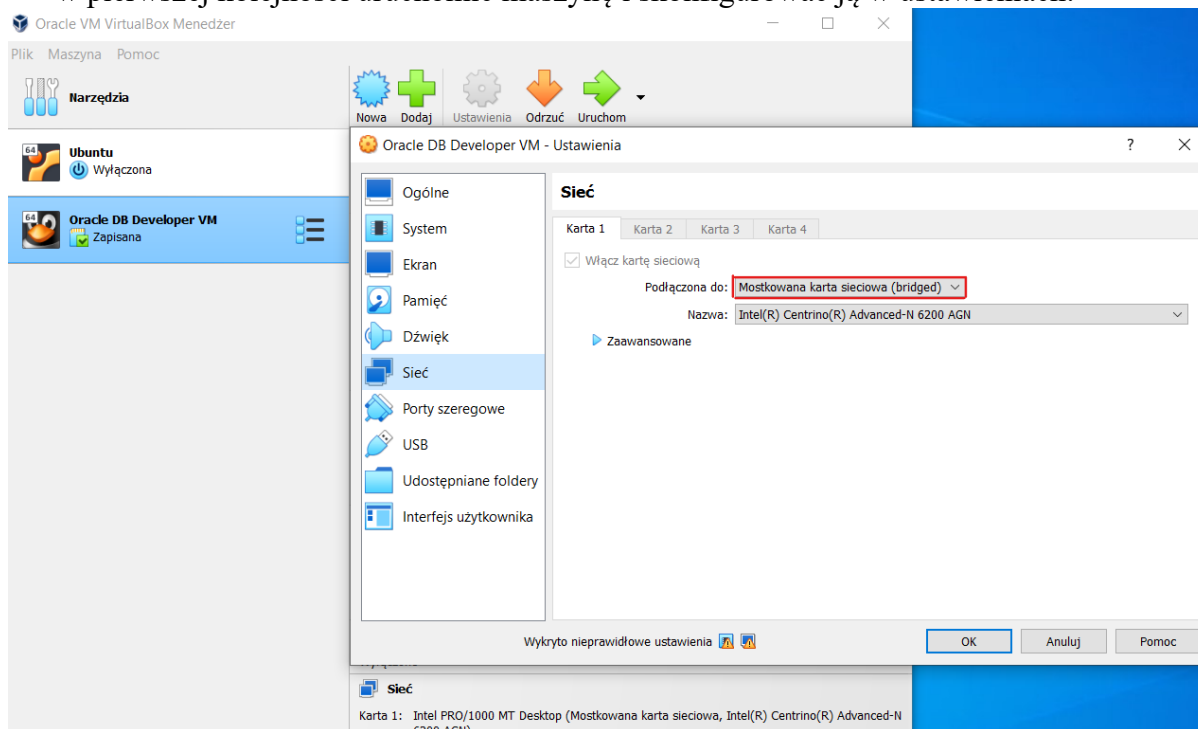
Rysunek 16. Kolumny tabeli Rezerwacja

	REZERWA...	KLIENT_ID	POCZATE...	KONIEC_...	NUMER_R...	MIEJSCE_...	MIEJSCE_...
1	1111	1111	22/12/08	22/12/14	SR62592	1	2
2	1112	2121	22/12/03	22/12/07	FSU7531	2	2
3	1113	3131	22/12/08	22/12/12	WRO2893	2	2
4	1114	4141	22/12/12	22/12/16	WRO9803	2	2

Rysunek 17. Dane tabeli Rezerwacja

Po pomyślnym utworzeniu wszystkich tabel, możemy przystąpić do uruchamiania maszyny wirtualnej i następnie uruchomić na niej Oracle'ową bazę na Linuxie, a następnie stworzyć na tej bazie analogiczne tabele, jak na bazie lokalnej.

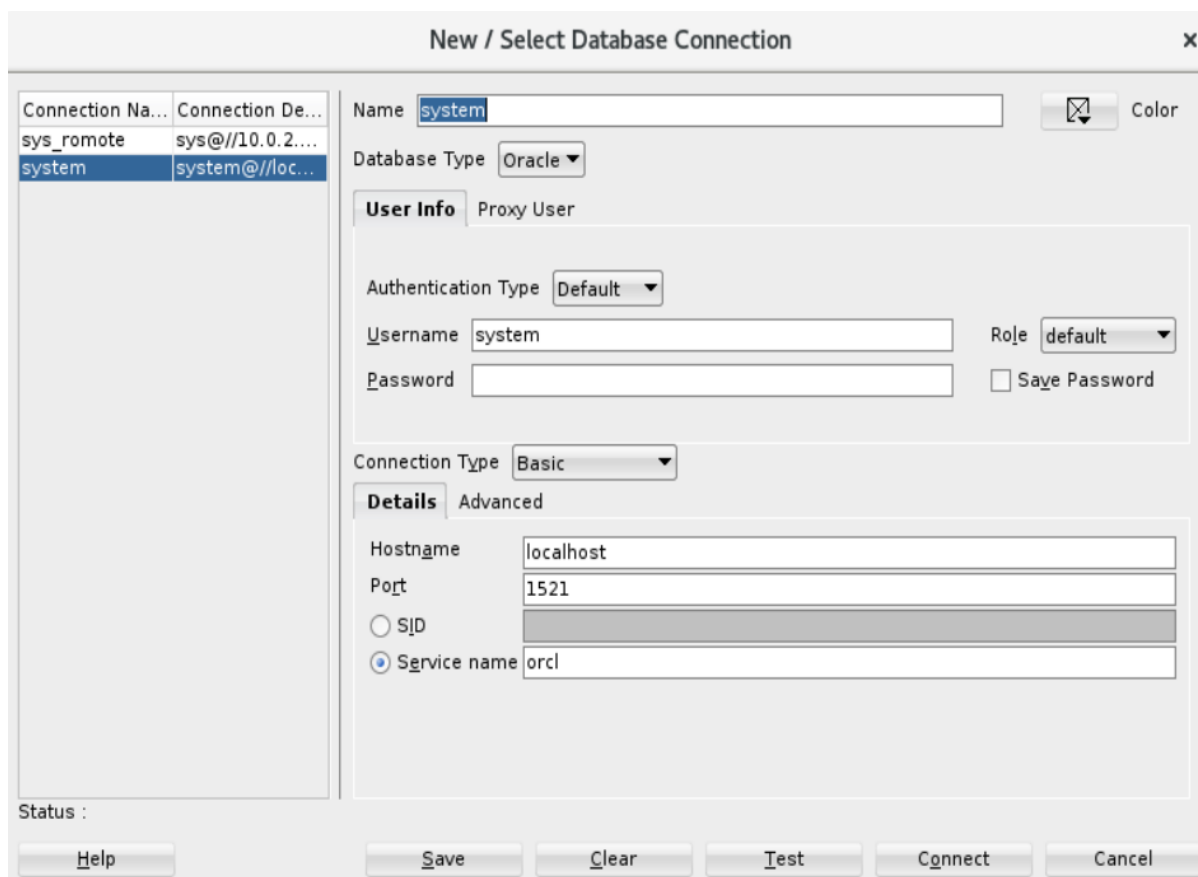
- Aby połączyć się z bazą Oracle DB Developer VM na maszynie wirtualnej, musimy w pierwszej kolejności uruchomić maszynę i skonfigurować ją w ustawieniach:



Rysunek 18. Konfiguracja bazy Oracle na Linuxie

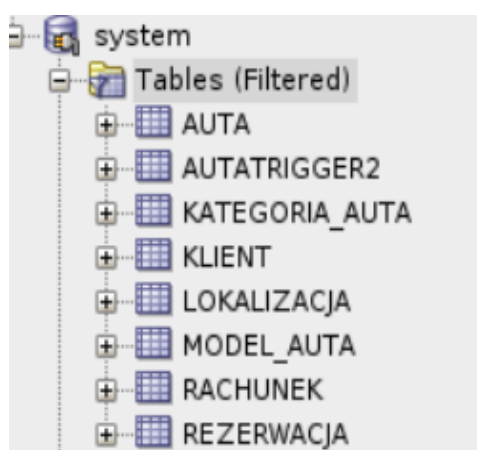
Konfiguracja bazy polega na wejściu w Ustawienia, następnie zakładkę Sieć i ustawieniu podłączenia do Mostkowanej karty sieciowej (bridged). Po wykonaniu tej czynności możemy uruchomić bazę.

Kolejnym krokiem, który wykonujemy jest utworzenie połączenia na nowej bazie.



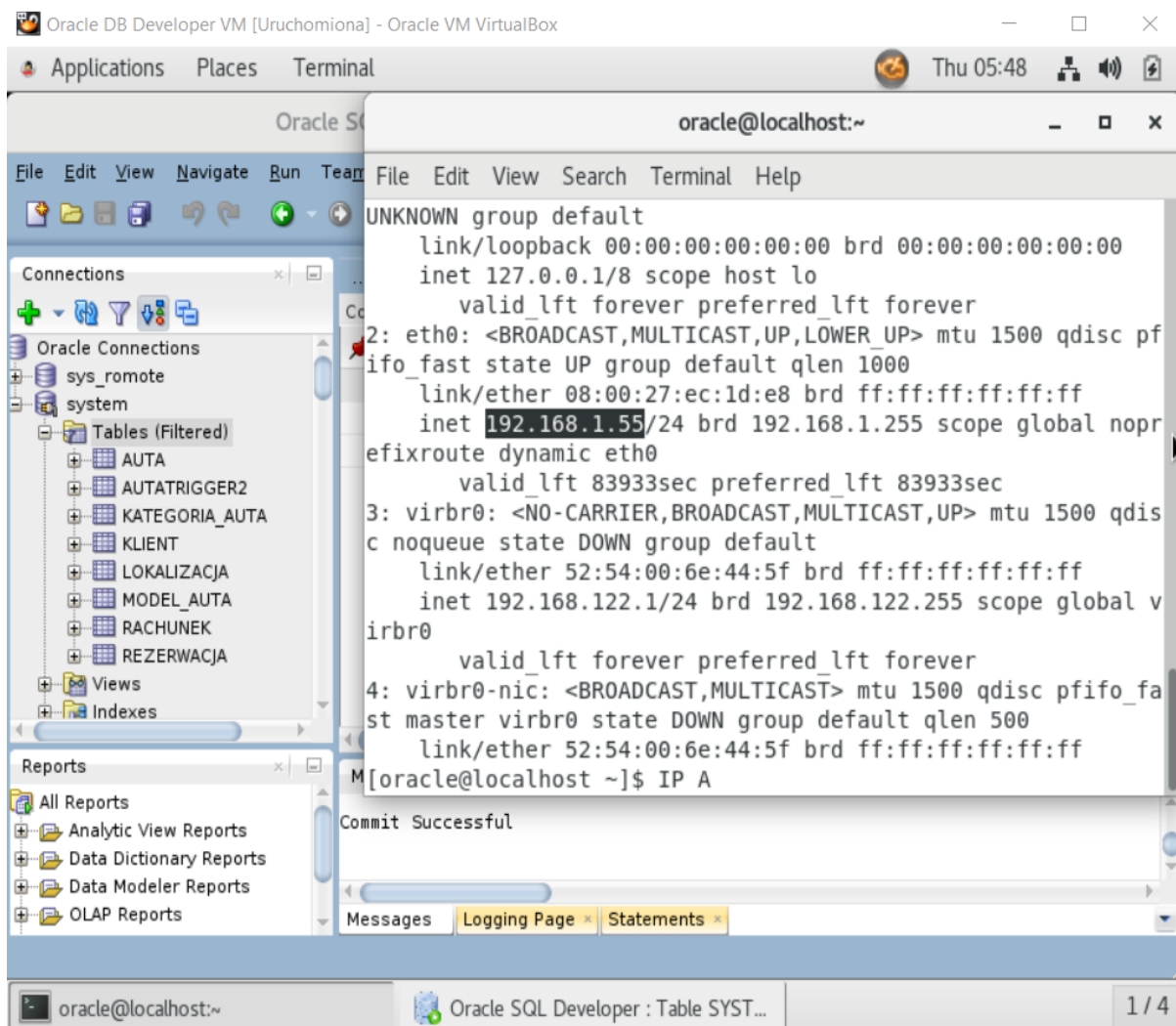
Rysunek 19. Połączenie z bazą Oracle na maszynie wirtualnej

Aby stworzyć połączenie z bazą Oracle tworzymy połączenie o nazwie „system” i o nazwie użytkownika „system” oraz hasło domyślnym „Oracle”. Następnie podajemy jako hostname nasz localhost oraz port analogicznie jak to było w przypadku łączenia się z bazą na Windowsie. Różnica jest jednak, że zamiast podawać jako SID XE, wybieramy Service name o nazwie specjalnej „orcl”. Po udanym połączeniu możemy utworzyć tabele na nowej bazie, analogiczne to tabel na bazie Windows, choć z odmiennymi danymi.



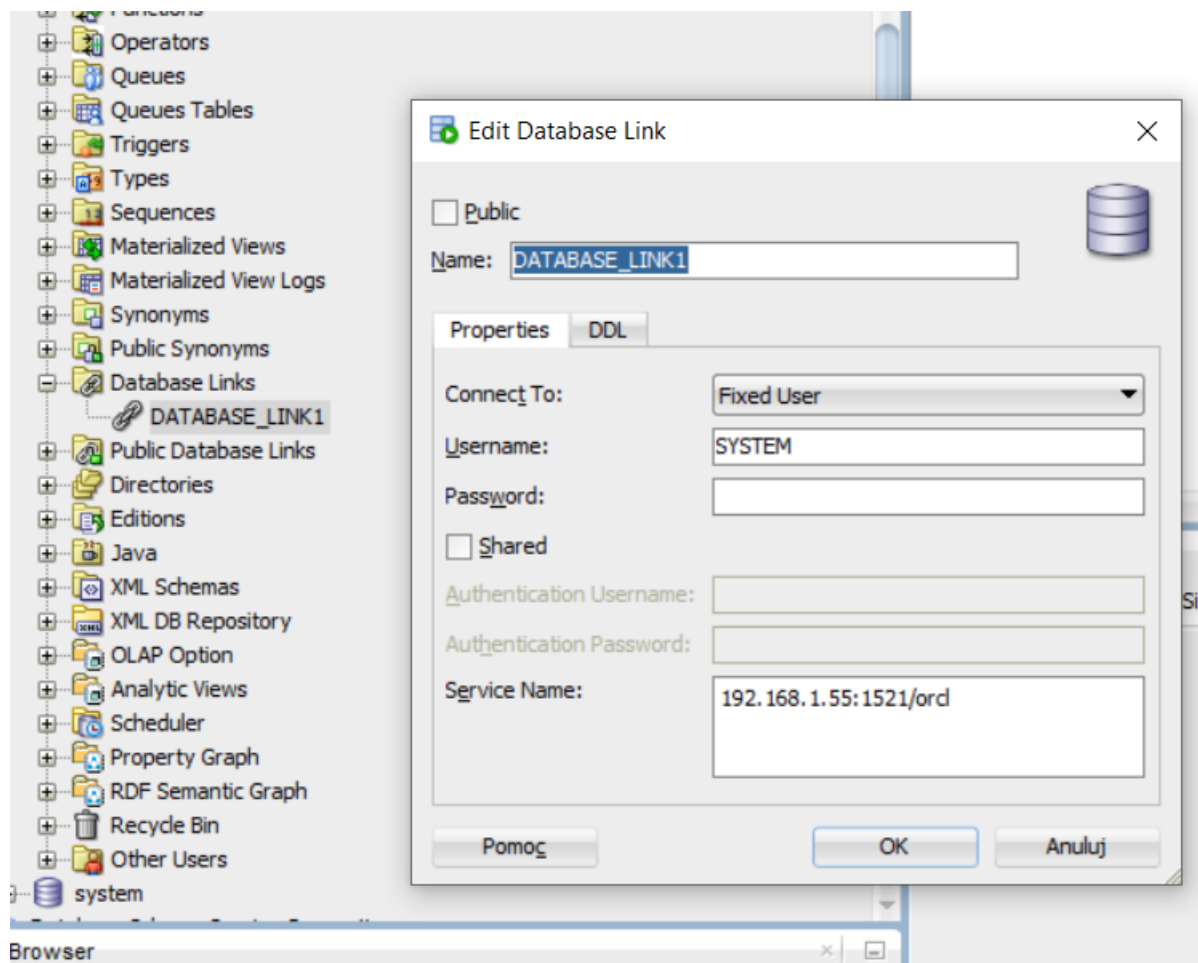
Rysunek 20. Tabele stworzone na bazie Oracle Linux

Po utworzeniu tabel następnym krokiem jest utworzenia połączenia między bazami. W tym celu potrzebny nam jest adres IP naszej sieci lokalnej. Pobieramy go otwierając `oracle@localhost` i używając komendy „`ip a`”.



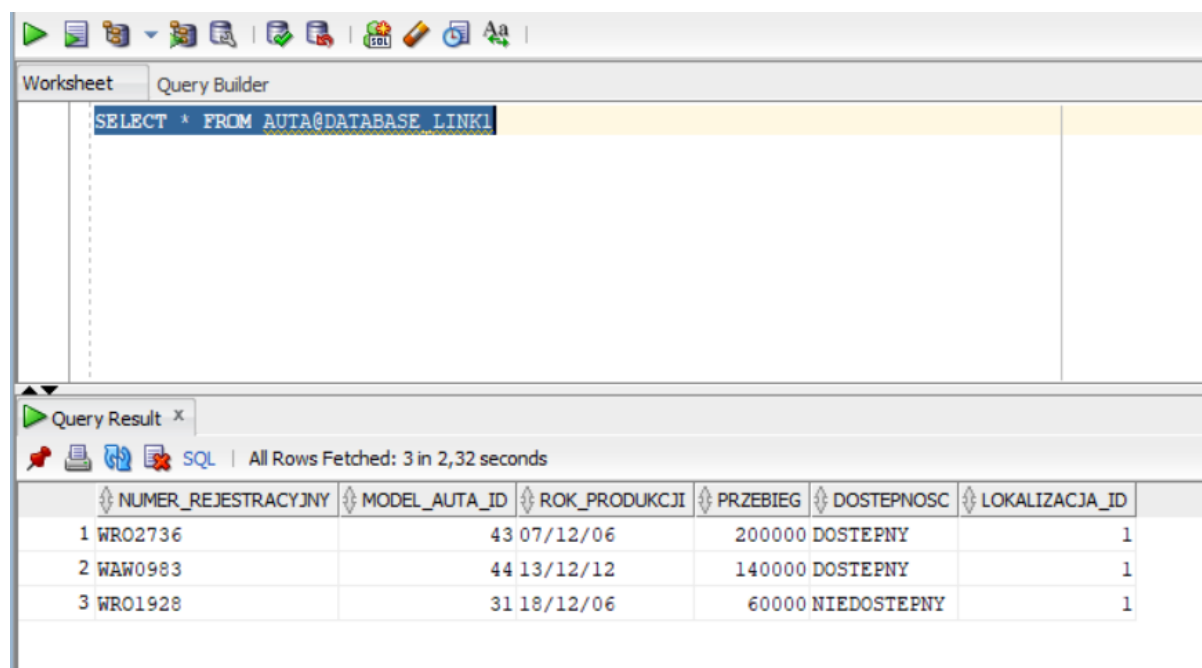
Rysunek 21. Pobranie adresu IP naszej sieci lokalnej

Po pobraniu adresu IP sieci jesteśmy już w stanie stworzyć nasze połączenie między bazami danych w postaci „Linka”. Tworzymy go na naszej bazie na Windowsie poprzez dodanie „Database_link1”. Następnie w parametrach wybieramy Username „system” i hasło „oracle”. W servis name wpisujemy adres IP naszej sieci, następnie po dwukropek dodajemy port oraz po ukośniku dodajemy „orcl”, czyli servis name naszej bazy Oracle na Linuksie i klikamy „ok”.

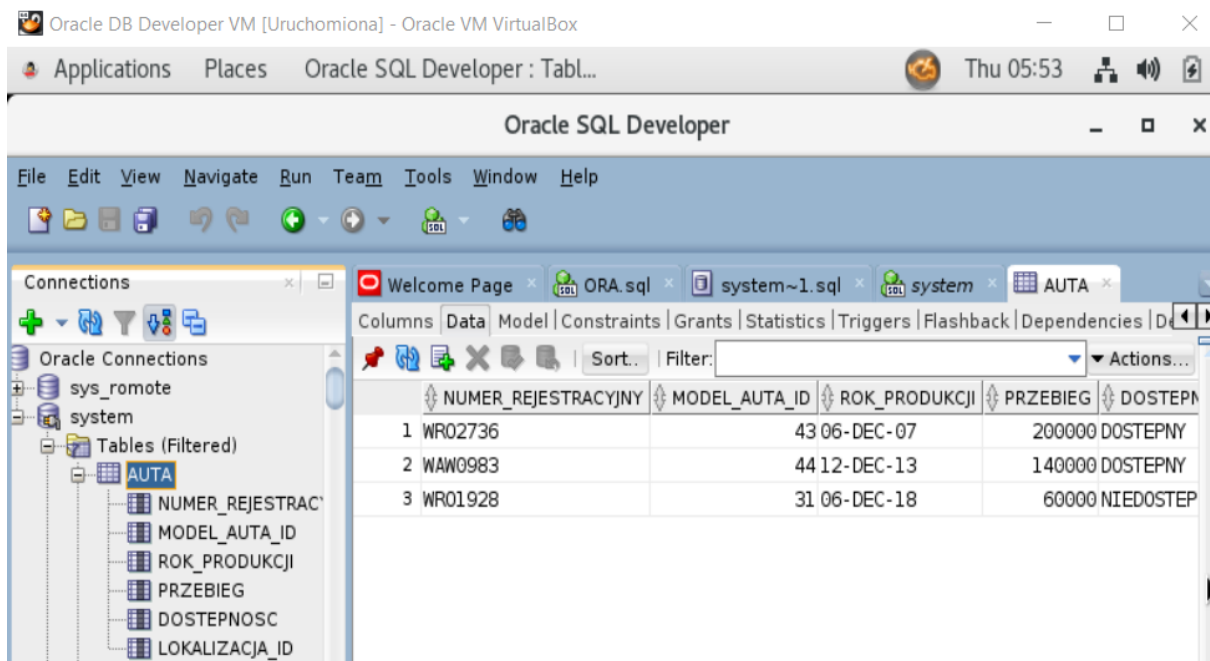


Rysunek 22. Tworzenie połączenia między bazami za pomocą "Link'a"

Teraz po poprawnym utworzeniu Database_link1 możemy przetestować odczytywanie danych z tabeli w bazie na Linuxie w worksheet bazy na Windowsie. Czynimy to za pomocą poniższej linijki kodu:



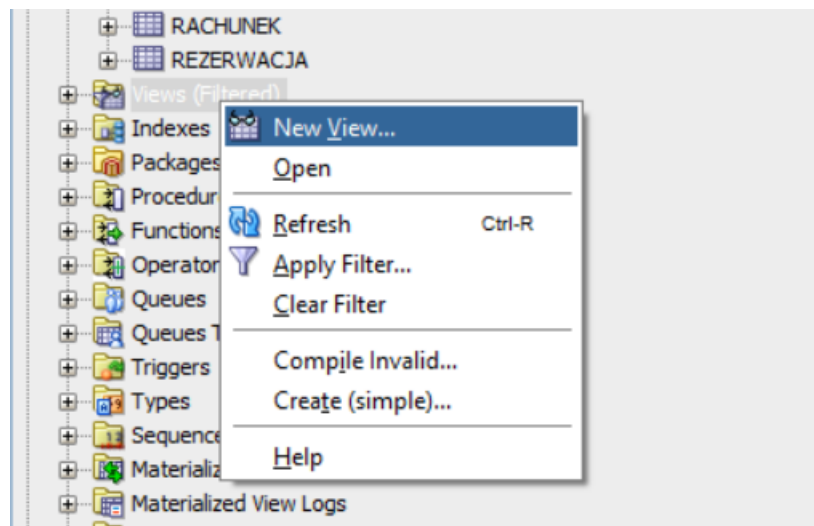
Rysunek 23. Wywołanie tabeli Auta z bazy na Linuxie



Rysunek 24. Tabela Auta na bazie Oracle Linux

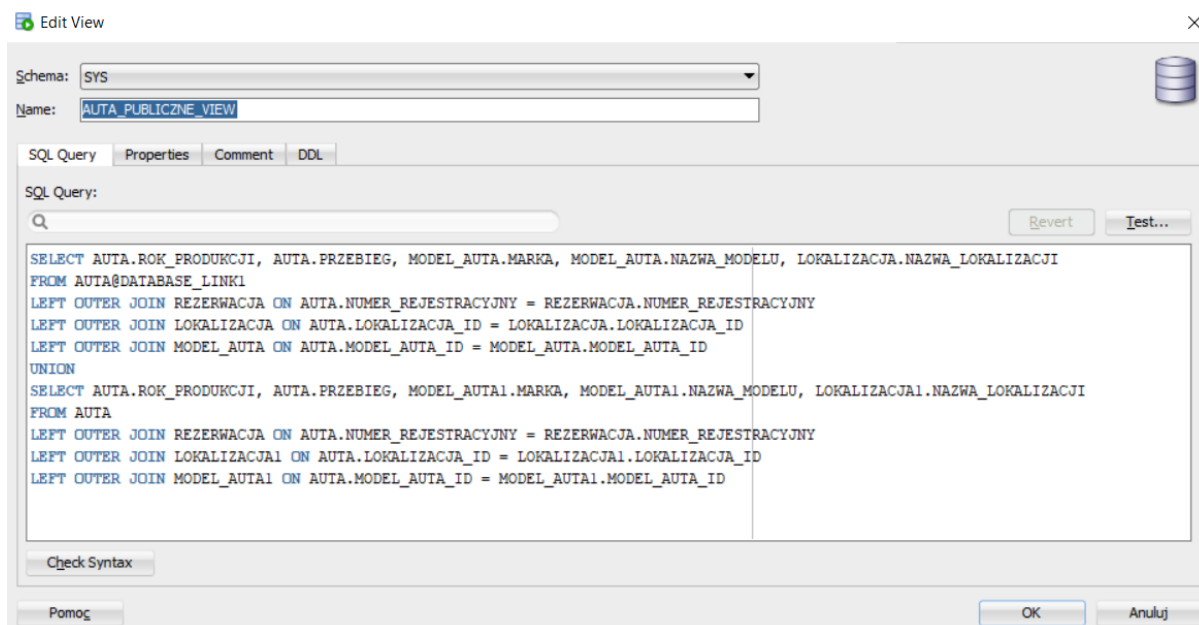
Jak widzimy na Rysunku 23 oraz 24. program odczytał poprawnie informacje z tabeli Auta na Linuxie.

- Po tych operacjach możemy przejść do tworzenia fragmentacji poziomej i pionowej oraz replikacji wybranych tabel. W tym celu tworzymy nowe widok w zakładce „View”.



Rysunek 25. Tworzenie nowego widoku

Aby stworzyć fragmentację poziomą dla tabeli Auta ze względu na Lokalizacje_ID aut, tworzymy we View następujący kod:



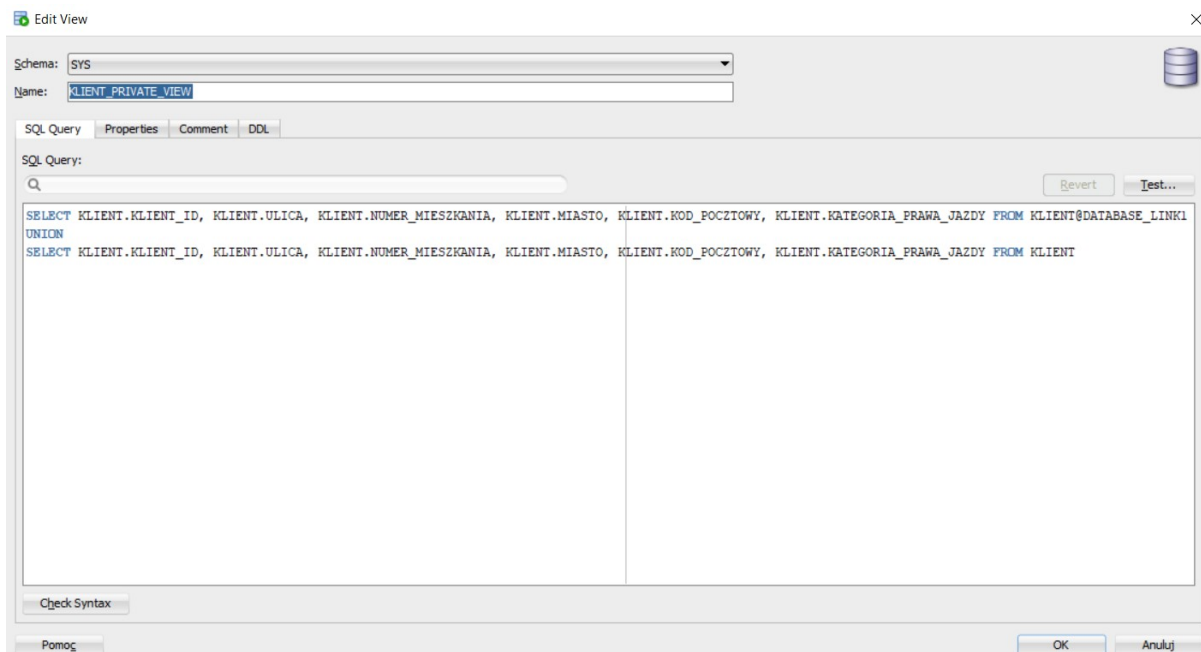
Rysunek 26. Stworzenie widoku dla połączonych tabel Auta z obu baz danych

W powyższym kodzie najpierw pobraliśmy tworzącym „AUTA_PUBLICZNE_VIEW” kolumny, które chcemy wyświetlić znajdują się po komendzie SELECT. Oprócz kolumn z tabeli AUTA, takich jak ROK_PRODUKCJI, czy PRZEBIEG zostały również wyświetlone kolumny z tabel MODEL_AUTA, jak MARKA, NAZWA_MODELU, czy też z tabeli LOKALIZACJA, jak NAZWA_LOKALIZACJI. Wszystkie te kolumny zostały pobrane, aby użytkownik miał dostęp do najważniejszych informacji na temat wypożyczanego samochodu, natomiast informacje niepotrzebne dla użytkownika zewnętrznego, jak LOKALIZACJA_ID, czy MODEL_AUTA_ID, join’ujemy z odniesieniami do innych tablic i ich nie wyświetlamy. Analogiczną operację przeprowadzamy dla obu baz, po czym je ze sobą łączymy. Efektem tego jest tabela zaprezentowana na Rysunku 27.

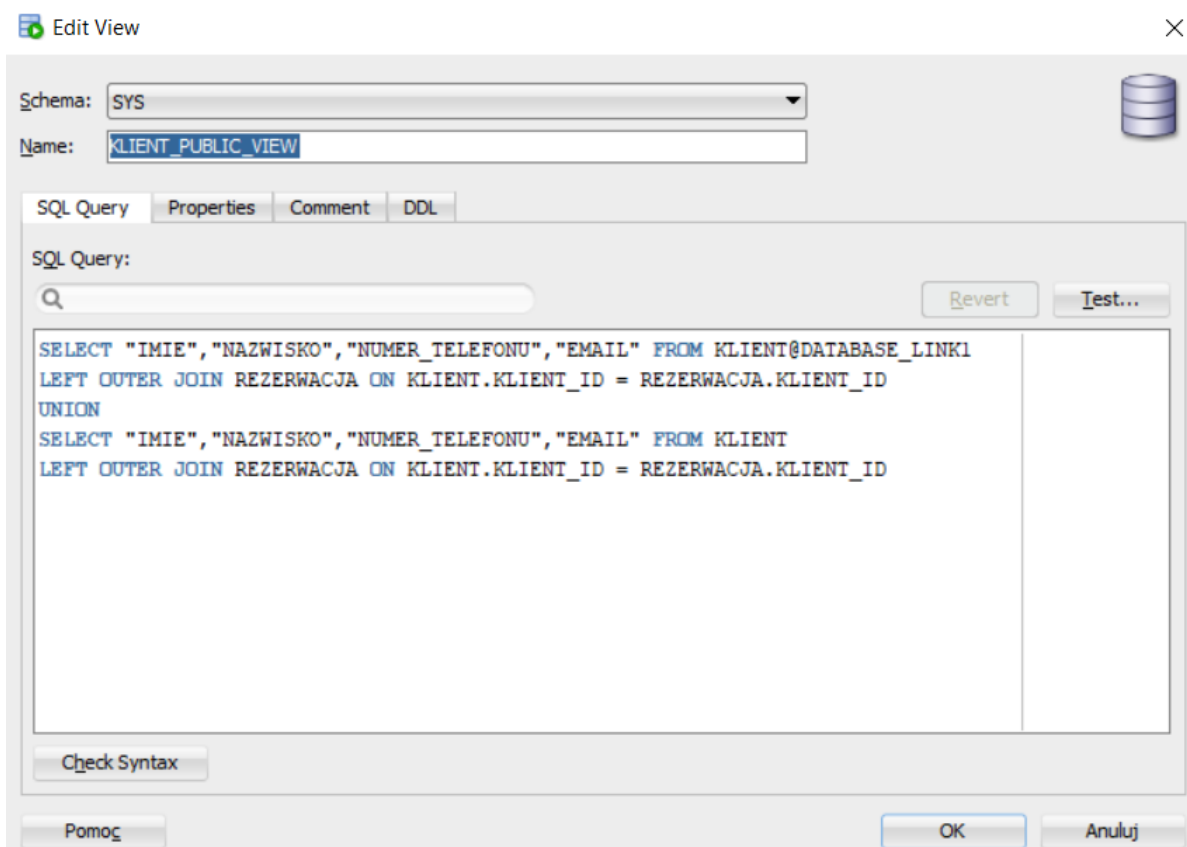
	ROK_PRODUKCJI	PRZEBIEG	MARKA	NAZWA_MODELU	NAZWA_LOKALIZACJI
1	18/12/06	60000	SKODA	OCTAVIA	PUNKT_1
2	13/12/12	140000	HONDA	ACCORD	PUNKT_1
3	07/12/06	200000	DAEWOO	NUBIRA	PUNKT_1
4	15/12/17	130000	HYUNDAI	I30	PUNKT_2
5	15/12/18	140000	HYUNDAI	I30	PUNKT_2
6	05/12/11	200000	PEUGEOT	107	PUNKT_2
7	05/12/17	180000	FIAT	PUNTO	PUNKT_2
8	05/12/03	239000	FIAT	PUNTO	PUNKT_2
9	17/12/09	100000	MAZDA	3	PUNKT_2

Rysunek 27. Tabela utworzona na podstawie "AUTA_PUBLICZNE_VIEW"

Następnie przechodzimy do pozycjonowania pionowego. W jego ramach rozdzielamy tabelę „Klient” ze względu na dane wrażliwe. W widoku tworzymy „KLIENT_PRIVATE_VIEW”, w którym znajdują się dane prywatne klienta, jak miejsce zamieszkania, adres ID, a następnie tworzymy drugi widok „KLIENT_PUBLIC_VIEW” dla danych publicznych, które będzie można zobaczyć.



Rysunek 28. Widok na prywatne dane klienta "KLIENT_PRIVATE_VIEW"



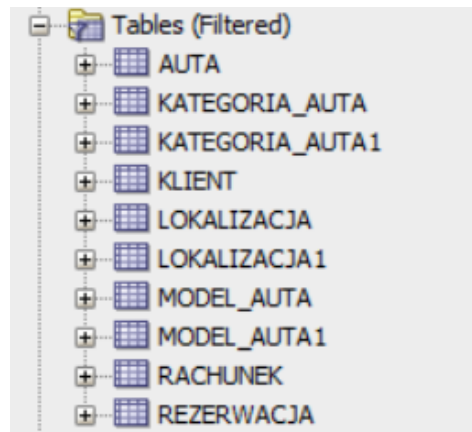
Rysunek 29. Widok na publiczne dane klienta Widok na prywatne dane klienta "KLIENT_PUBLIC_VIEW"

W następnej kolejności tworzymy replikację trzech tablic, które w każdej bazie różnią się one danymi: LOKALIZACJA, KATEGORIA_AUTA, MODEL_AUTA. Replikację przeprowadzamy za pomocą komendy SNAPSHOT. Tworzymy go za pomocą CREATE SNAPSHOT i od razu odświeżamy z wykorzystaniem procedury DBMS_SNAPSHOT.REFRESH.

```
CREATE SNAPSHOT MODEL_AUTA AS SELECT * FROM MODEL_AUTA@DATABASE_LINK1
SELECT * FROM MODEL_AUTA
BEGIN
DBMS_SNAPSHOT.REFRESH('MODEL_AUTA');
END;
```

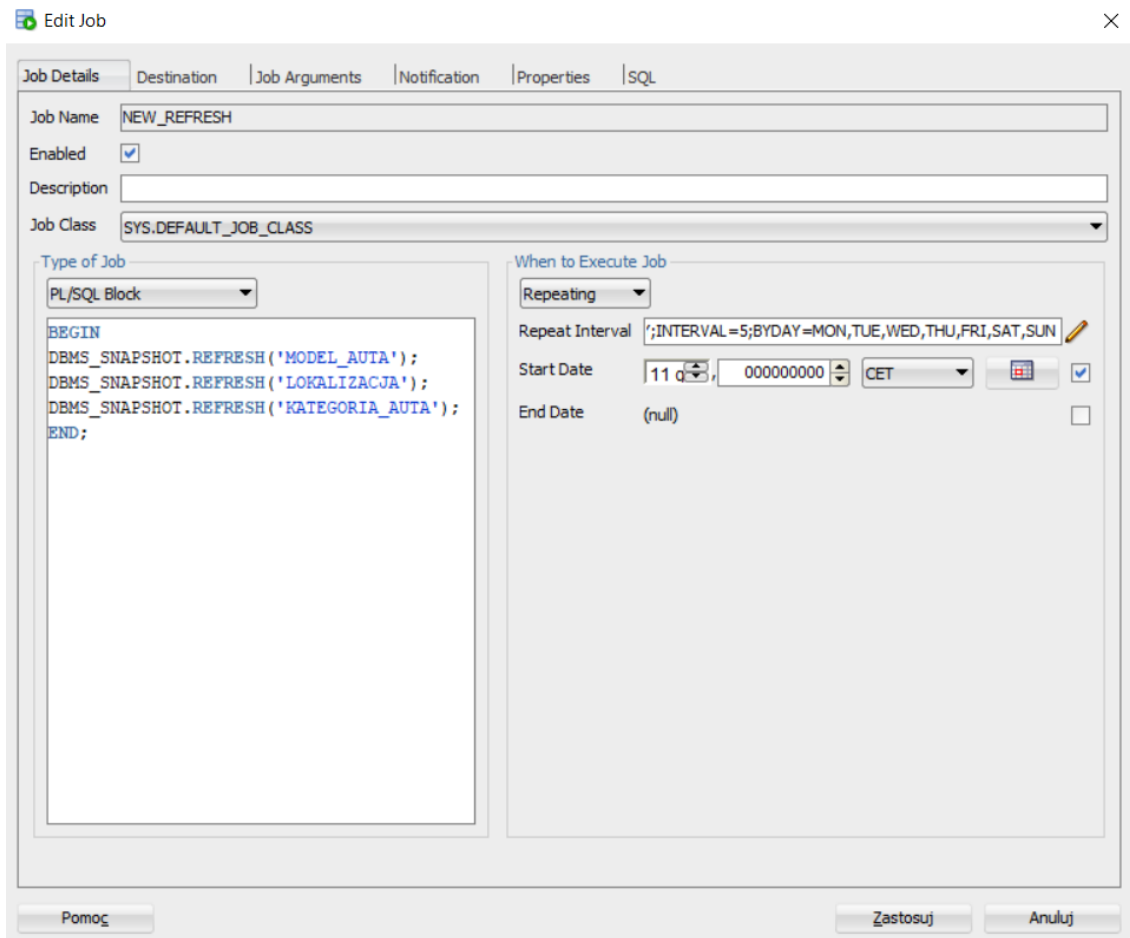
Rysunek 30. Utworzenie SNAPSHOT'a dla MODEL_AUTA z bazy na Linuxie

W ten sam sposób tworzymy SNAPSHOT'y dla kolejnych tablic LOKALIZACJA oraz KATEGORIA_AUTA. Tabele te pojawiają się nam następnie jako replikacje tabel z bazy stworzonej na Linuxie w bazie stworzonej na Windowsie.



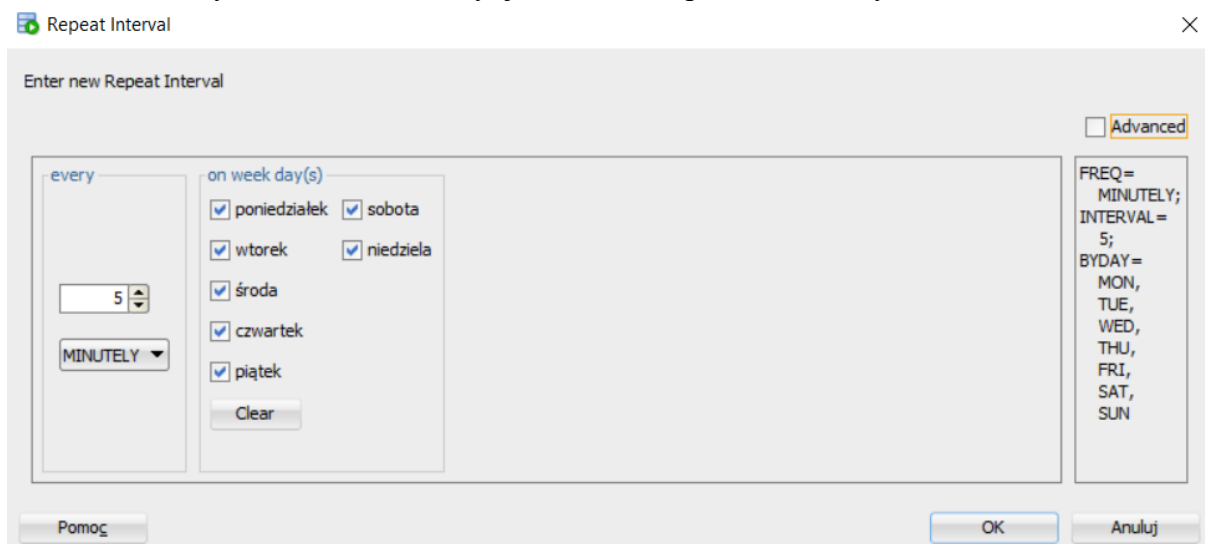
Rysunek 31. Tabele na bazie Windows po replikacji

Snapshot'y możemy następnie wykorzystać, aby ustalić interwał czasowy odświeżania danych w tabeli replikowanej w stworzonym nowo „Job'ie”, który odpowiada za regularne odświeżanie wartości tabel, gdy ulegają one zmianie.



Rysunek 32. Utworzenie "Job'a", który odpowiada za odświeżanie wartości tabeli replikowanej

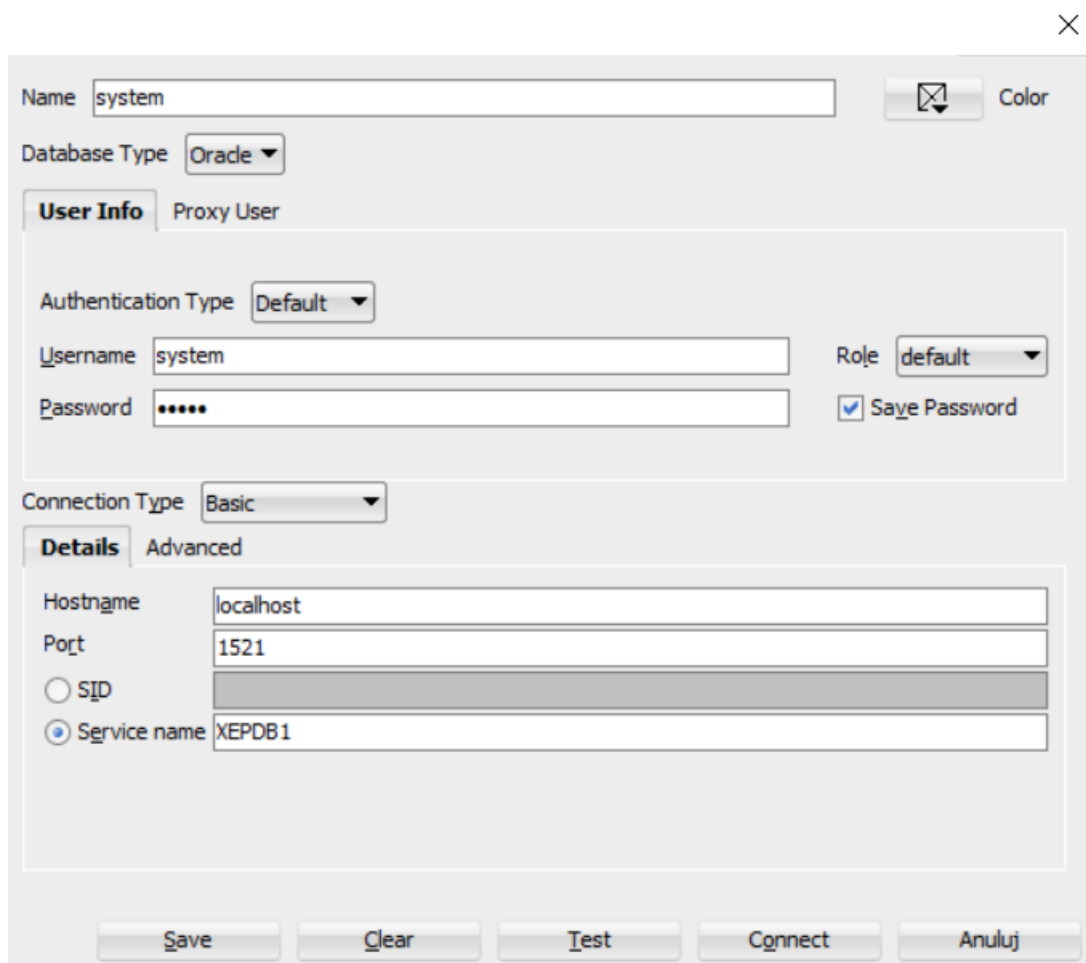
Po wybraniu Repeating w zakładce Repeat Interval możemy ustalić dokładny interwał czasowy odświeżania tablicy, jak zostało to pokazane na Rysunku 33.



Rysunek 33. Ustalenie czasu odświeżania wartości tablic na 5 minut

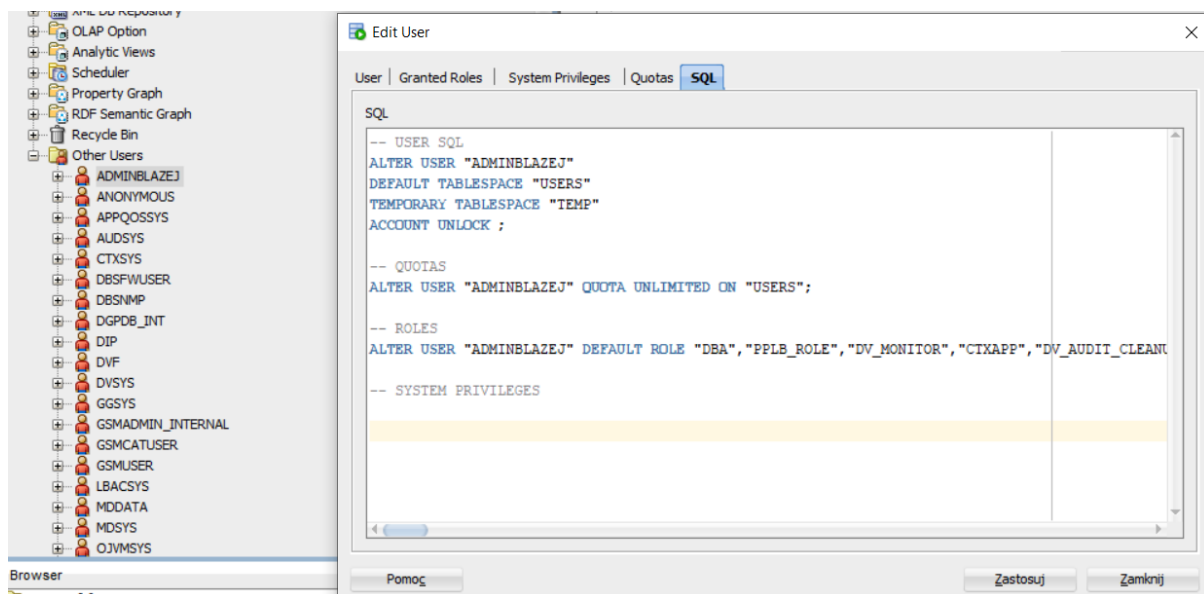
- Jako następny punkt naszego projektu postawiliśmy sobie za cel stworzenia triggera, który wykonywałby nam operację przypisania dodatkowych wartości do tabeli Auta w odpowiedniej bazie podczas, gdy program jednocześnie nie wiedziałby, że nasze bazy są rozproszone.

W celu utworzenia triggera do celów wizualizacji jego działania musieliśmy stworzyć osobne połączenia z bazami, gdyż przy połączeniu SYS trigger nie działał ze względu na brak uprawnień. Utworzyliśmy zatem na naszej bazie na Windowsie nowe połączenie o nazwie „system” jak pokazano na Rysunku 34.



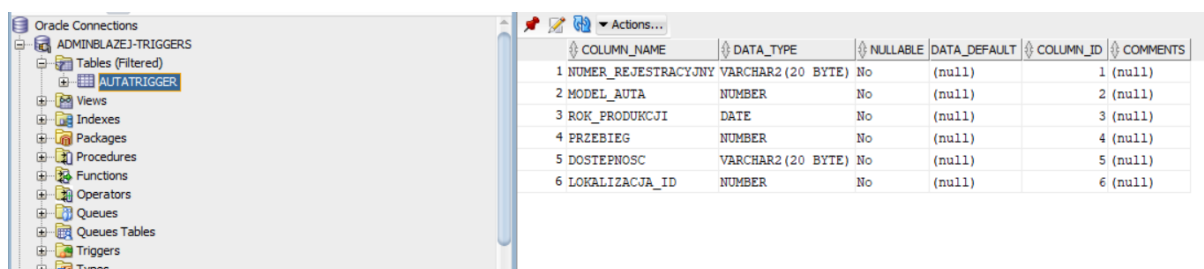
Rysunek 34. Utworzenie połączenia system do bazy na Windows

Różnica w porównaniu do poprzedniego połączenia RENTALCARDATABASE polega na zmianie Service name na XEPDB1. Dzięki temu możemy w następnym kroku utworzyć na tym połączeniu nowego użytkownika o nazwie ADMINBLAZEJ, któremu przyznajemy wszystkie uprawnienia, jak pokazano na Rysunku 35.



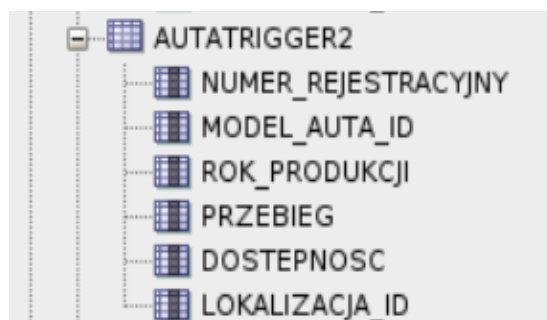
Rysunek 35. Utworzenie nowego użytkownika

Po tym kroku tworzymy nowe połączenie o nazwie ADMINBLAZEJ-TRIGGERS, wykorzystując nazwę użytkownika ADMINBLAZEJ, a w nim tworzymy tabelę AUTATRIGGER, która posiada te same kolumny co nasze wcześniejsze tabele AUTA. Dzięki utworzeniu nowego połączenia posiadamy już uprawnienia, aby tworzyć TRIGGER'y, a nowa tabela pozwoli nam zademonstrować ich działanie.

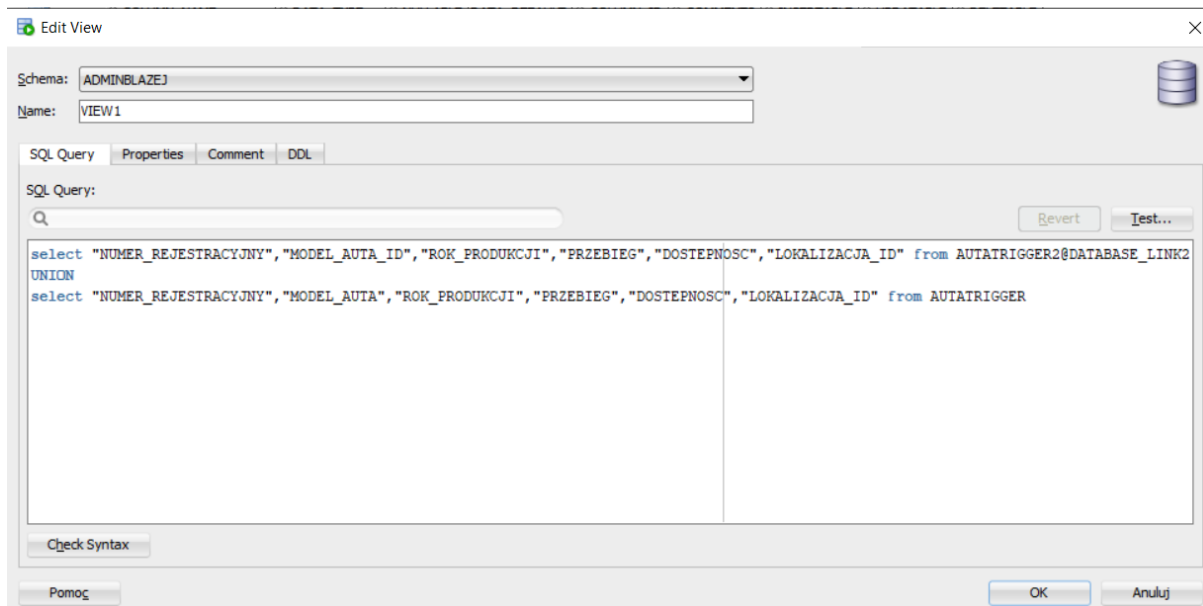


Rysunek 36. Utworzenie nowej tabeli AUTATRIGGER

Po wykonaniu analogicznych działań na bazie Oracle na Linuxie posiadamy już dwie bazy, na których możemy wykorzystać TRIGGER. Są to kolejno tabela AUTATRIGGER na bazie Windows oraz tabela AUTATRIGGER2 na bazie Linux. Najpierw robimy „View1” dla tych tabel, a następnie tworzymy TRIGGER.



Rysunek 37. Tabela AUTATRIGGER2 utworzona w bazie Oracle Linux

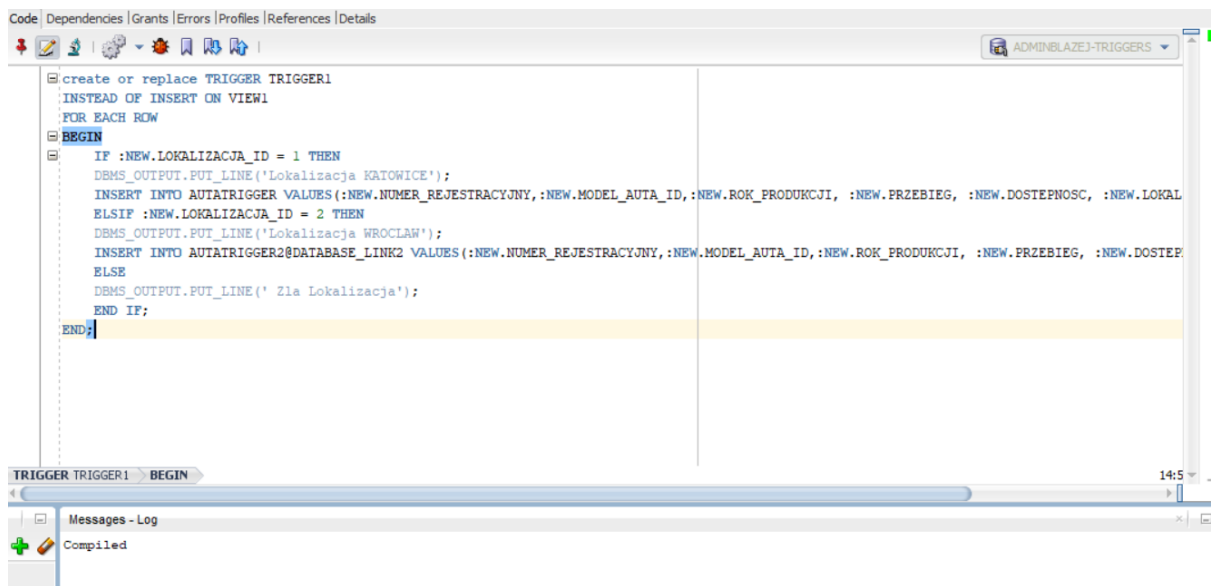


Rysunek 38. Stworzenie View1 dla tabel AUTATRIGGER i AUTATRIGGER2

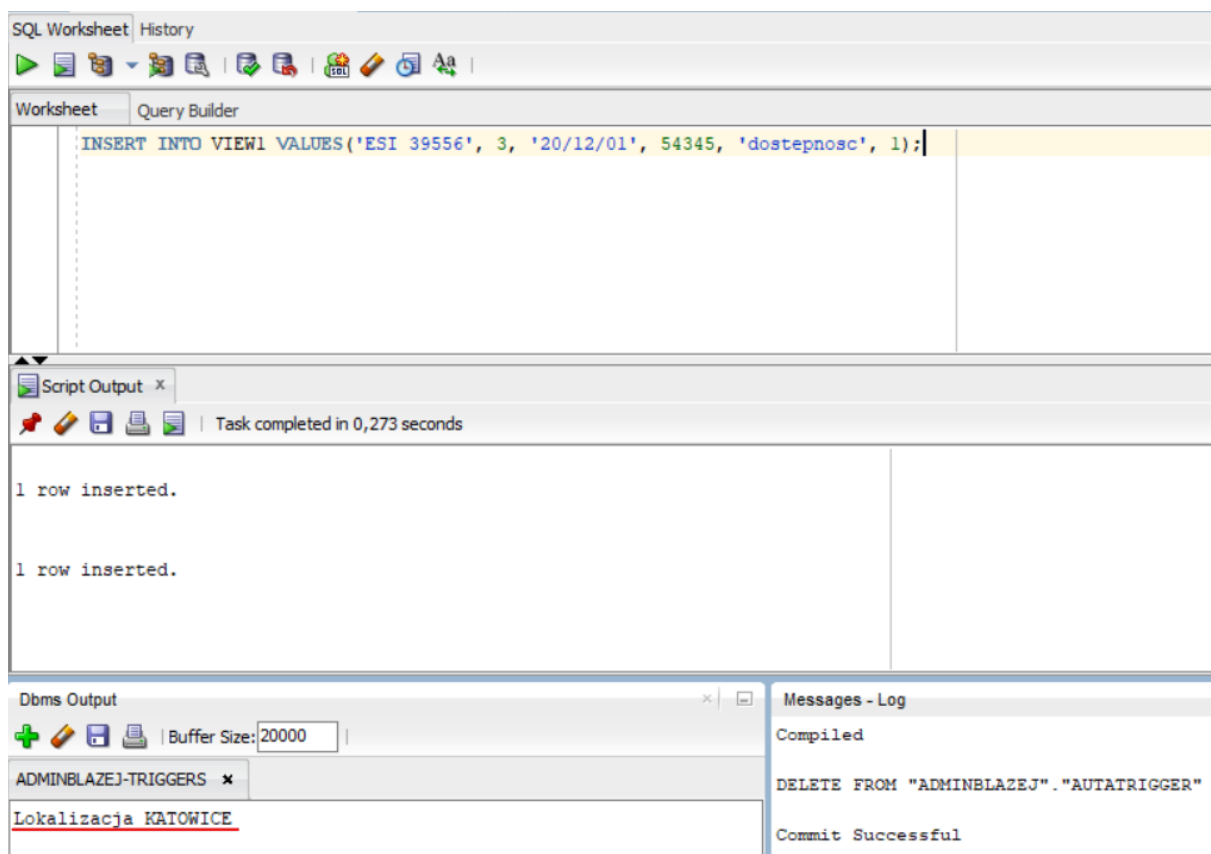
```
create or replace TRIGGER TRIGGER1
INSTEAD OF INSERT ON VIEW1
FOR EACH ROW
BEGIN
    IF :NEW.LOKALIZACJA_ID = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Lokalizacja KATOWICE');
        INSERT INTO AUTATRIGGER VALUES(:NEW.NUMER_REJESTRACYJNY,:NEW.MODEL_AUTA_ID,:NEW.ROK_PRODUKCJI, :NEW.PRZEBIEG, :NEW.DOSTEPNOSC,
        :NEW.LOKALIZACJA_ID );
    ELSIF :NEW.LOKALIZACJA_ID = 2 THEN
        DBMS_OUTPUT.PUT_LINE('Lokalizacja WROCLAW');
        INSERT INTO AUTATRIGGER2@DATABASE_LINK2 VALUES(:NEW.NUMER_REJESTRACYJNY,:NEW.MODEL_AUTA_ID,:NEW.ROK_PRODUKCJI, :NEW.PRZEBIEG,
        :NEW.DOSTEPNOSC, :NEW.LOKALIZACJA_ID );
    ELSE
        DBMS_OUTPUT.PUT_LINE(' Zła Lokalizacja');
    END IF;
END;
```

Rysunek 39. Kod programu tworzącego TRIGGER

Jak widać na powyższym kodzie, nasz TRIGGER sprawdza, czy lokalizacja naszej wypożyczalni znajduje się w dwóch z możliwych lokalizacji, czyli Katowicach, bądź Wrocławiu i następnie przypisuje nowe dane do odpowiedniej lokalizacji, czyli jednej z naszych dwóch baz danych. Na Rysunku 38. przedstawiono kompilujący się program triggera.



Rysunek 40. Kompilacja programu TRIGGERA



Rysunek 41. Przypisanie nowych danych do tablicy

Jak widzimy na Rysunku 39. program działa poprawnie i po przypisaniu lokalizacji „1” w values zwraca lokalizację numer „1”, czyli Katowice.

Na Rysunkach 40 i 41 zaprezentowano jak dodano nowe dane do tablicy za pomocą TRIGGERA.

