

Projekt nr 1 - Modelowanie i implementacja bazy - Dom seniora

Rafał Dadura, Juliusz Kuzyka

Listopad 2023

Spis treści

1 Zakres i cel projektu	2
2 Definicja systemu	2
2.1 Funkcjonalność systemu	2
2.2 Perspektywy użytkowników	2
3 Model konceptualny	3
3.1 Definicja zbiorów encji określonych w projekcie (decyzje projektowe)	3
3.2 Ustalenie związków między encjami i ich typów	3
3.3 Określenie atrybutów i ich dziedzin	4
3.4 Dodatkowe reguły integralnościowe (reguły biznesowe)	5
3.5 Klucze kandydujące i główne (decyzje projektowe)	5
3.6 Schemat ER na poziomie konceptualnym	6
3.7 Problem pułapek szczelinowych i wachlarzowych – analiza i przykłady	7
3.7.1 Pułapka wachlarzowa	7
3.7.2 Pułapka szczelinowa	7
4 Model logiczny	8
4.1 Charakterystyka modelu relacyjnego	8
4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady	8
4.3 Proces normalizacji – analiza i przykłady	10
4.3.1 Pierwsza postać normalna	10
4.3.2 Druga postać normalna	10
4.3.3 Trzecia postać normalna	10
4.4 Schemat ER na poziomie modelu logicznego	11
4.5 Więzy integralności	12
4.6 Proces denormalizacji – analiza i przykłady	12
5 Faza fizyczna	14
5.1 Projekt transakcji i weryfikacja ich wykonalności	14
5.2 Strojenie bazy danych – dobór indeksów	14
5.3 Skrypt SQL zakładający bazę danych	15
5.4 Przykładowe dane w naszej bazie danych	23
5.5 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych	25

1 Zakres i cel projektu

Projekt obejmuje stworzenie relacyjnej bazy danych, której głównym celem jest skoncentrowanie się na zaprojektowaniu i zaimplementowaniu struktury umożliwiającej efektywne przechowywanie oraz zarządzanie danymi. Zakres projektu obejmuje określenie relacji między różnymi elementami danych, ustalenie kluczy głównych i obcych, a także opracowanie struktury tabel, aby spełnić określone założenia funkcjonalne. W tej pierwszej fazie projektu priorytetem jest stworzenie solidnej podstawy relacyjnej bazy danych, która będzie efektywnie obsługiwać przewidywane potrzeby systemu.

2 Definicja systemu

System bazy danych dla domu seniora to kompleksowy zestaw powiązanych informacji i funkcji, skoncentrowany na gromadzeniu, zarządzaniu oraz dostępie do danych związanych z mieszkańcami i operacjami instytucji. Obejmuje to przechowywanie danych osobowych mieszkańców, ich kart medycznych, usług im oferowanych, a także informacji dotyczących pracowników - opiekunów i lekarzy. Dodatkowo, system ten może zawierać moduły umożliwiające śledzenie pokoi, w których znajdują się mieszkańcy. Celem takiego systemu jest zautomatyzowanie procesów administracyjnych oraz zapewnienie sprawnego zarządzania danymi, co przyczynia się do podniesienia standardu obsługi w domu seniora.

2.1 Funkcjonalność systemu

System naszej bazy danych powinien umożliwiać:

- Dodawanie nowych mieszkańców.
- Dodawanie pracowników.
- Dodawanie, modyfikowanie, usuwanie kart medycznych.
- Przechowywanie informacji, który mieszkaniec jest przypisany do usługi.
- Dostęp do informacji, który pracownik zajmuje się danym pokojem.
- Przeglądanie informacji o lekarzach i opiekunach.
- Przechowywanie informacji, kto zajmuje się mieszkańcem.
- Przechowywanie informacji, który pracownik wykonuje daną usługę.
- Sprawdzanie informacji na temat pokoi.

2.2 Perspektywy użytkowników

- **Perspektywa mieszkańców:** Mieszkańcy domu seniora skorzystają z systemu jako narzędzia poprawiającego ich codzienne życie. Dostęp do własnych danych medycznych, harmonogramu dziennej opieki, czy aktywnościach rekreacyjnych pozwala na lepszą organizację życia.
- **Perspektywa pracowników medycznych (lekarzy):** Dla lekarzy pracujących w domu seniora, system bazy danych ułatwia dostęp do aktualnych informacji medycznych mieszkańców. Może to obejmować historię chorób, historię badań, czy wyniki badań.
- **Perspektywa pracowników opieki:** Dla opiekunów system może stanowić narzędzie do efektywnego zarządzania codziennymi obowiązkami. Informacje o harmonogramie leków czy potrzebach opiekuńczych są łatwo dostępne, co ułatwia dostosowanie opieki do indywidualnych potrzeb mieszkańców.

3 Model konceptualny

3.1 Definicja zbiorów encji określonych w projekcie (decyzje projektowe)

Nasz encje w modelu konceptualnym zdefiniowaliśmy w następujący sposób:

- **Dom seniora** - instytucja lub miejsce zapewniające kompleksową opiekę i wsparcie dla osób starszych.
- **Mieszkaniec** - osoba zamieszkująca dom seniora, korzystająca z usług świadczonych przez tę instytucję.
- **Pokój** - przestrzeń mieszkalna przypisana konkretnym mieszkańcom w domu seniora.
- **Karta medyczna** - dokument zawierający informacje medyczne dotyczące danego mieszkańca.
- **Pracownik** - osoba zatrudniona w domu seniora, pełniąca różne funkcje związane z obsługą i opieką nad mieszkańcami.
- **Lekarz** - specjalista medyczny odpowiedzialny za udzielanie świadczeń zdrowotnych mieszkańcom domu seniora.
- **Opiekun** - osoba odpowiedzialna za opiekę nad mieszkańcami, zapewniająca wsparcie w różnych aspektach życia codziennego.
- **Usługa** - świadczenie dodatkowych usług dostarczanych mieszkańcom domu seniora w celu poprawy ich jakości życia.

3.2 Ustalenie związków między encjami i ich typów

- **Zajmuje_sie** - Dom Seniora - Mieszkaniec - jeden do wielu - binarny. Jeden dom seniora może opiekować się wieloma mieszkańcami. Dom seniora może nie mieć mieszkańców, aby istnieć.
- **Posiada** - Mieszkaniec - Karta medyczna - jeden do jednego - binarny. Każdy jeden mieszkaniec domu seniora musi posiadać jedną kartę medyczną.
- **Oferuje** - Dom seniora - Pokój - jeden do wielu - binarny. Dom seniora musi posiadać wiele pokoi.
- **Proponuje** - Dom seniora - Usługa - jeden do wielu - binarny. Dom seniora może proponować mieszkańcom oferty, ale nie musi.
- **Wykonuje** - Pracownik - Usługa - wielu do wielu - binarny. Wielu pracowników może wykonywać wiele usług, ale może być pracownik, który nie wykona żadnej usługi.
- **Dbą** - Pracownik - Mieszkaniec - wielu do wielu - binarny. Wielu pracowników musi zajmować się mieszkańcami.
- **Korzysta** - Mieszkaniec - Usługa - wielu do wielu - binarny. Wielu mieszkańców może korzystać z usług, ale może być usługa, z której nikt nie korzysta.
- **Mieszka** - Mieszkaniec - Pokój - wielu do jednego - binarny. Wielu mieszkańców może mieszkać w jednym pokoju, ale jeden mieszkaniec musi mieszkać w pokoju.
- **Zatrudnia** - Dom seniora - Pracownik - jeden do wielu - binarny. Dom seniora nie musi zatrudnić pracowników, aby istniał.
- **Zawód** - Pracownik - Lekarz i Opiekun - specjalizacja. Pracownik specjalizuje się jako lekarz albo opiekun.

3.3 Określenie atrybutów i ich dziedzin

Poniżej w tabelach znajdują się encje z atrybutami i ich dziedzinami.

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_domu_seniora	Integer	T	wartość unikatowa
Nazwa_domu_seniora	VarChar(30)	T	wartość unikatowa
Nr_telefonu	VarChar(12)	T	-
Adres	VarChar(150)	T	pole segmentowe
Email	VarChar(40)	T	-

Tabela 1: Encja Dom seniora

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_mieszkanca	Integer	T	wartość unikatowa
Nr_dokumentu_mieszkanca	VarChar(10)	T	wartość unikatowa
Imie	VarChar(20)	T	-
Nazwisko	VarChar(20)	T	-
Pesel	Char(11)	T	-
Plec	{'K','M'}	T	zdefiniowana domena
Adres	VarChar(150)	T	pole segmentowe
Email	VarChar(40)	T	-
Nr_telefonu	VarChar(12)	T	-

Tabela 2: Encja Mieszkaniec

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_pokoju	Integer	T	wartość unikatowa
Nr_pietra	VarChar(3)	T	-
Metraz	Float(120)	T	-
Wyposazenie	VarChar(1200)	T	pole wielowartościowe

Tabela 3: Encja Pokoj

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_karty_medycznej	Integer	T	wartość unikatowa
Historia_chorob	VarChar(1200)	T	-
Historia_leczenia	VarChar(1200)	T	-
Szczepienia	VarChar(500)	T	-
Dane_rodzinne	VarChar(1200)	N	-
Wzrost	Float(200)	T	-
Waga	Float(120)	T	-
Grupa_krwi	{'A','B','AB','0'}	T	zdefiniowana dziedzina

Tabela 4: Encja Karta medyczna

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_pracownika	Integer	T	wartość unikatowa
Nr_dokumentu_pracownika	VarChar(10)	T	wartość unikatowa
Imie	VarChar(20)	T	-
Nazwisko	VarChar(20)	T	-
Nr_telefonu	VarChar(12)	T	-
Email	VarChar(40)	T	-
Godziny_pracy	Time	T	-

Tabela 5: Encja Pracownik

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_licencji_lekarskiej	Integer	T	wartość unikatowa
Specjalizacja	VarChar(100)	T	-
Tytul_zawodowy	VarChar(20)	T	-
Ukonczone_uczelnie	VarChar(1200)	T	pole wielowartościowe

Tabela 6: Encja Lekarz

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Certyfikaty_z_zakresu_opieki	VarChar(500)	T	-
Doswiadczenie_zawodowe	VarChar(1200)	T	-
Obszar_odpowiedzialnosci	VarChar(1200)	T	pole wielowartościowe

Tabela 7: Encja Opiekun

Nazwa atrybutu	Dziedzina	Obowiązkowość	Dodatkowe informacje
Nr_uslugi	Integer	T	wartość unikatowa
Nazwa_uslugi	VarChar(120)	T	wartość unikatowa
Cena_uslugi	Money	T	-
Czas	Time	T	-
Opis	VarChar(1200)	N	-

Tabela 8: Encja Usługa

3.4 Dodatkowe reguły integralnościowe (reguły biznesowe)

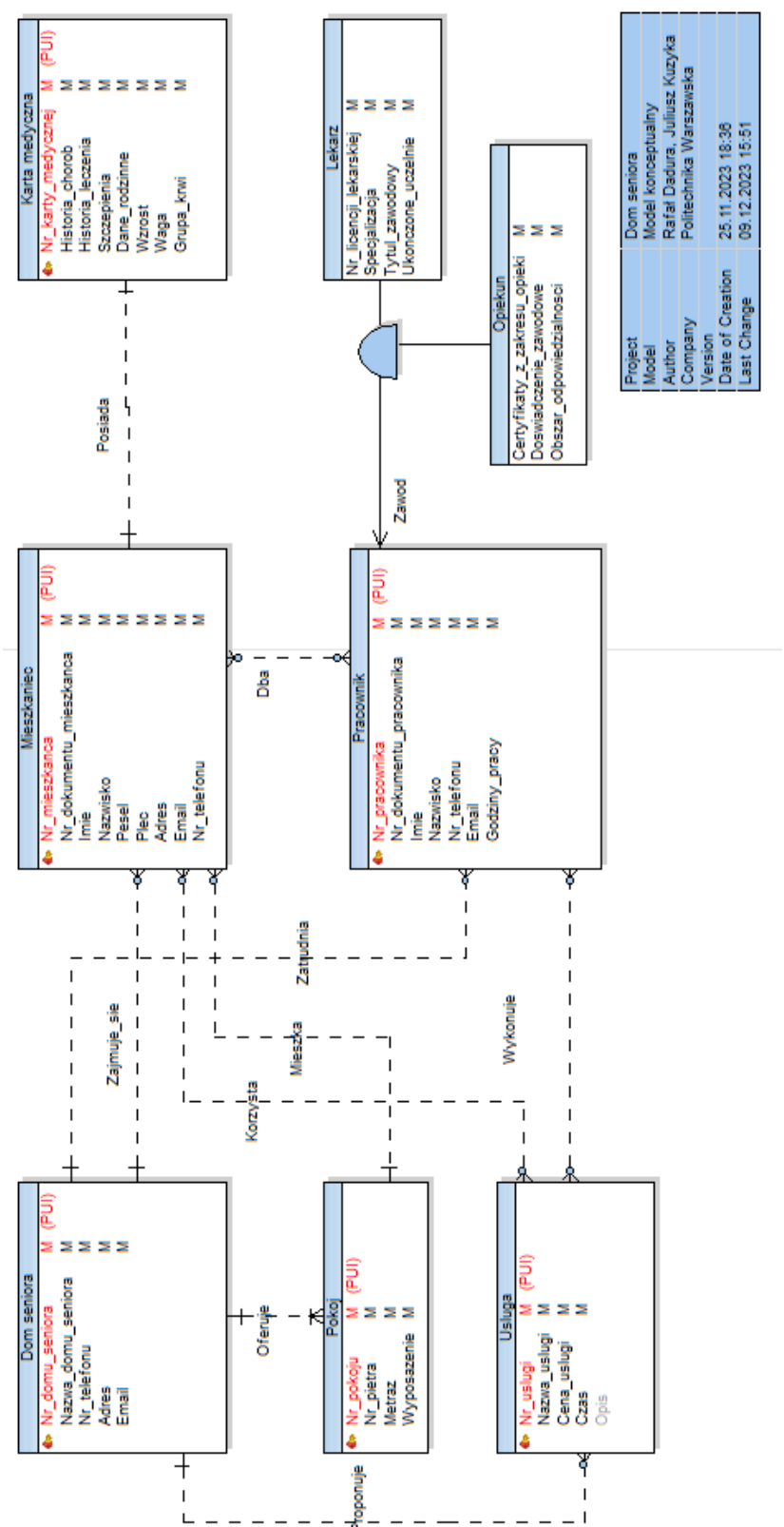
W projektowanej bazie danych zdecydowaliśmy się zastosować jedną regułę integralności biznesowej, która wymaga, aby data zameldowania klienta były zawsze równa lub późniejsza niż aktualna data. Ta reguła ma na celu zapewnienie spójności danych oraz zgodność z wymaganiami dotyczącymi obsługi rezerwacji w naszym systemie.

3.5 Klucze kandydujące i główne (decyzje projektowe)

Encja	Klucz główny	Klucz kandydujący
Dom seniora	Nr_domu_seniora	Nazwa_domu_seniora
Mieszkaniec	Nr_mieszkanca	Nr_dokumentu_mieszkanca
Pokoj	Nr_pokoju	-
Karta medyczna	Nr_karty_medycznej	-
Pracownik	Nr_pracownika	Nr_dokumentu_pracownika
Lekarz	Nr_lekarza	Nr_licencji_lekarskiej
Opiekun	Nr_opiekuna	-
Usługa	Nr_uslugi	Nazwa_uslugi

Tabela 9: Klucze kandydujące i główne

3.6 Schemat ER na poziomie konceptualnym



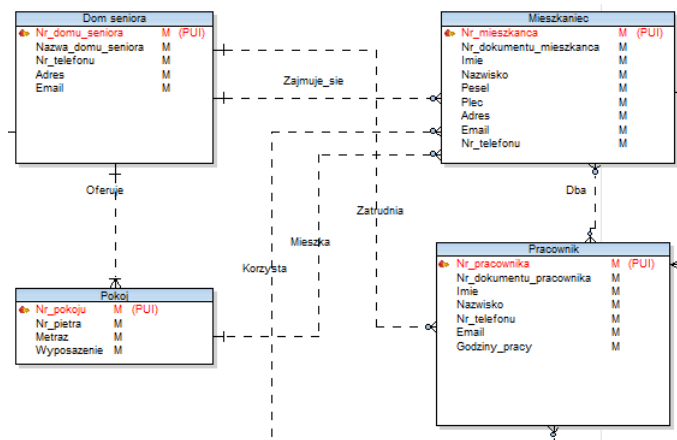
Rysunek 1: Schemat ER na poziomie konceptualnym

3.7 Problem pułapek szczelinowych i wachlarzowych – analiza i przykłady

W przeprowadzonej przez nas analizie bazy danych zauważyliśmy problem związany z pułapką wachlarzową. Poniżej zaprezentujemy rozwiązanie naszego problemu oraz pokażemy przykład pułapki szczelinowej.

3.7.1 Pułapka wachlarzowa

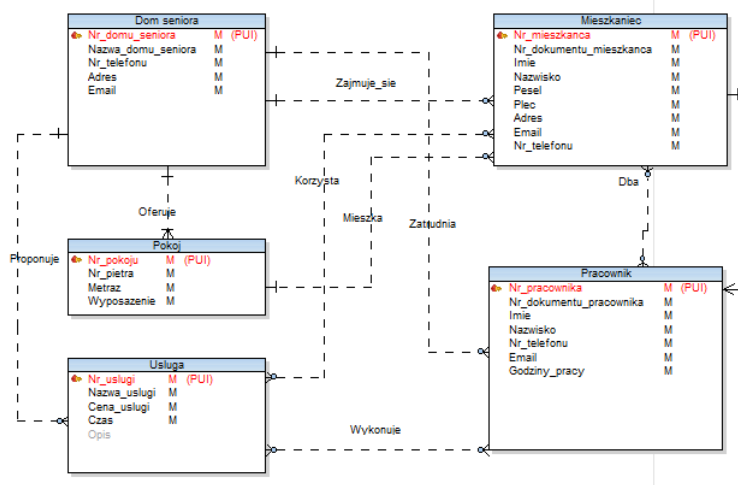
W naszym przypadku, dodaliśmy relację "Mieszka" pomiędzy encją Mieszkaniec a encją Pokój, ponieważ przed dodaniem tego wpadliśmy w pułapkę wachlarzową. Według naszych założeń, mieszkaniec przynależy tylko do jednego pokoju. Wielu mieszkańców należy do jednego domu seniora, a dom seniora oferuje wiele pokoi. Gdybyśmy nie dodali relacji "Mieszka", powiązanie między mieszkańcem a pokojem nie byłoby jednoznaczne, stąd powstałby wachlarz możliwości.



Rysunek 2: Poprawiona pułapka wachlarzowa

3.7.2 Pułapka szczelinowa

Gdybyśmy usunęli relację "Proponuje" między domem seniora i usługą wpadlibyśmy w pułapkę szczelinową, ponieważ dom seniora zatrudniałby pracowników, którzy mogliby wykonywać wiele usług lub żadnej. Wtedy mogłaby wydarzyć sytuacja, w której usługa nie byłaby ani oferowana przez dom seniora, ani wykonywana przez pracownika, ani mieszkaniec nie musiałby korzystać z niej, więc wpadłaby w "szczelinę".



Rysunek 3: Przykład pułapki szczelinowej

4 Model logiczny

4.1 Charakterystyka modelu relacyjnego

W następnej części mieliśmy za zadanie stworzyć model logiczny po przekształceniu naszego modelu konceptualnego. Model relacyjny baz danych jest powszechnie używanym modelem, w którym dane są zorganizowane w tabelach. Każda tabela składa się z wierszy, nazywanych krotkami, oraz kolumn, reprezentujących atrybuty danych.

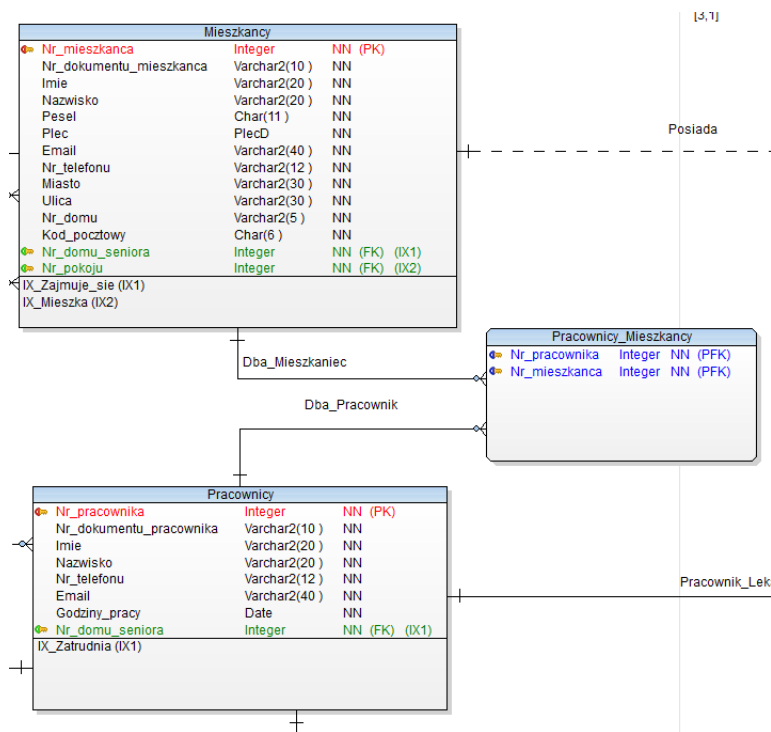
Krotki w tabelach zawierają pełne informacje o poszczególnych rekordach, na przykład o konkretnych obiektach czy jednostkach danych, dlatego pierwszym etapem była zamiana nazw encji na liczbę mnogą.

W modelu relacyjnym każda tabela posiada klucz główny, który jednoznacznie identyfikuje każdą krotkę w tej tabeli. Klucz główny musi być unikalny i nie może przyjmować wartości null. Ponadto, w modelu relacyjnym istnieje możliwość definiowania relacji między różnymi tabelami. Wprowadza się klucze obce, które umożliwiają określenie powiązań między danymi przechowywanymi w różnych tabelach.

Dodatkowo, model relacyjny podlega zasadom integralności referencyjnej, co oznacza, że dane muszą być spójne i zgodne z określonymi regułami integralności. W celu optymalnego organizowania danych, często stosuje się proces normalizacji, eliminujący redundancję i zależności funkcyjne.

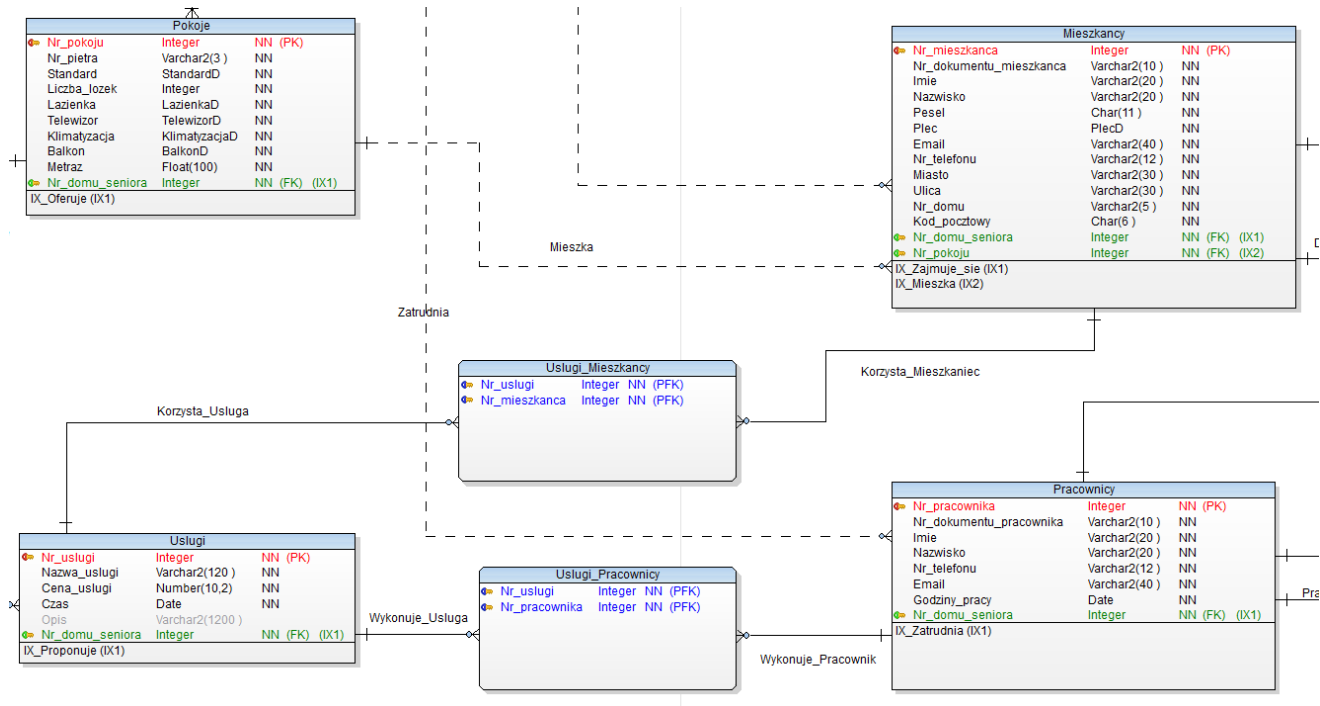
4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady

Pierwszym przykładem usunięcia właściwości niekompatybilnych było stworzenie dodatkowych tablic w relacjach wielu do wielu. W związku wielu do wielu jedna instancja jednej encji może być powiązana z wieloma instancjami drugiej encji, i odwrotnie. Bez użycia tabeli pośredniczącej, ciężko byłoby skutecznie reprezentować takie relacje w modelu relacyjnym. Poniżej zamieszczamy przykład takiej sytuacji, gdy w relacji mieszkańców i pracowników powstała nowa tablica Pracownicy_Mieszkanicy.



Rysunek 4: Przykład tablicy łączącej

Drugim przykładem będzie również stworzenie tablic łączących między usługami a mieszkańcami i usługami a pracownikami. Były to ponownie związki wielu do wielu, a z racji, że przy konwersji na model relacyjny, tablica w związkach wiele do wiele jest często wprowadzana, ponieważ model relacyjny bazuje na relacjach pomiędzy tabelami, a związki wiele do wiele są trudne do bezpośredniego przedstawienia w pojedynczej tabeli. Tabele w relacyjnej bazie danych reprezentują encje, a relacje pomiędzy nimi są przedstawiane za pomocą kluczy obcych.



Rysunek 5: Przykład tablicy łączącej

Ostatnim przykładem będzie eliminacja pól wielowartościowych. W modelu relacyjnym nie zaleca się przechowywania wielowartościowych atrybutów bezpośrednio w jednej kolumnie. Na przykład, jeśli istnieje atrybut, który może przyjmować wiele wartości (u nas to ukończone uczelnie lekarzy), należy rozdzielić te wartości na różne wiersze lub stworzyć oddzielną tabelę relacyjną dla tych wartości.




Rysunek 6: Przykład usunięcia pola wielowartościowego

4.3 Proces normalizacji – analiza i przykłady

4.3.1 Pierwsza postać normalna

Każda krotka w tabeli powinna zawierać atomowe wartości (brak powtarzających się grup lub wielowartościowych atrybutów), dlatego zmieniliśmy atrybut Adres, który był atrybutem segmentowym na atomowe wartości, takie jak ulica, nr domu, kod pocztowy.

Domy seniora		
 Nr_domu_seniora	Integer	NN (PK)
Nazwa_domu_seniora	Varchar2(30)	NN
Nr_telefonu	Varchar2(12)	NN
Email	Varchar2(40)	NN
Miasto	Varchar2(30)	NN
Ulica	Varchar2(30)	NN
Nr_domu	Varchar2(5)	NN
Kod_pocztowy	Char(6)	NN

Rysunek 7: Przykład poprawionej tablicy według pierwszej postaci normalnej

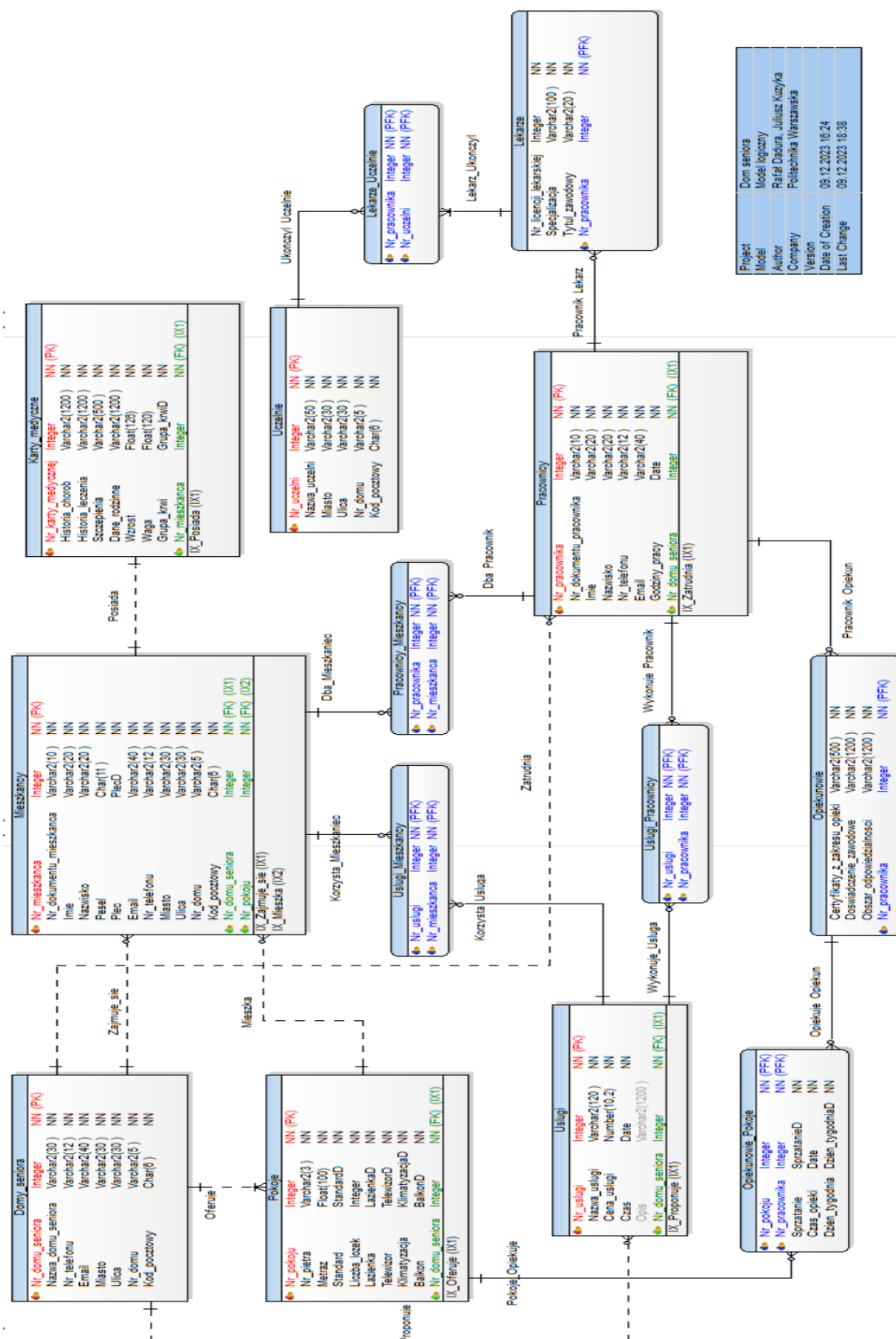
4.3.2 Druga postać normalna

Każda kolumna, która nie należy do klucza głównego, jest w pełni zależna od klucza głównego. Innymi słowy, żadna kolumna nie może zależeć jedynie od pewnego podzbioru klucza głównego, ale powinna zależeć od całego klucza głównego. Jeżeli mamy tabelę, w której klucz główny składa się z kilku kolumn, każda inna kolumna w tej tabeli powinna zależeć od wszystkich tych kolumn klucza głównego, a nie tylko od części z nich i naszym przypadku jest to spełnione.

4.3.3 Trzecia postać normalna

Każda kolumna, która nie jest kluczem kandydującym, jest wzajemnie niezależna od innych kolumn, a jedynie zależy od klucza głównego. Innymi słowy, każda kolumna, która nie jest kluczem głównym ani kluczem kandydującym, nie może zależeć od innych kolumn, które również nie są kluczami kandydującymi i w naszym modelu nie znaleźliśmy problemu niezgodnego z trzecią postacią normalną.

4.4 Schemat ER na poziomie modelu logicznego



Rysunek 8: Schemat ER na poziomie modelu logicznego

4.5 Więzy integralności

Więzy integralności w bazach danych to specjalne reguły, które zostały wprowadzone w celu utrzymania spójności, unikalności i poprawności danych między różnymi tabelami. Istnieje kilka rodzajów więzów integralności, z których każdy spełnia określone zadanie:

- **Więzy klucza głównego (Primary Key Constraint):** Zapewnia, że dane w wybranej kolumnie (lub grupie kolumn) są unikalne i nie zawierają wartości null. Klucz główny jednoznacznie identyfikuje każdy rekord w tabeli, co umożliwia łatwe odwoływanie się do konkretnych danych.
- **Więzy klucza obcego (Foreign Key Constraint):** Ustanawia relacje między tabelami, określając, że wartości w kolumnie klucza obcego muszą odpowiadać wartościom w kolumnie klucza głównego w innej tabeli. To umożliwia powiązywanie danych między różnymi tabelami w bazie.
- **Więzy unikalności (Unique Constraint):** Gwarantuje, że wartości w określonej kolumnie (lub grupie kolumn) są unikalne, co oznacza, że nie można mieć dwóch rekordów z identycznymi danymi w tej kolumnie. Niektóre systemy pozwalają na jeden null w kolumnie związanej z więzami unikalności.
- **Więzy sprawdzania (Check Constraint):** Określa warunki, które dane muszą spełniać, aby zostać dodane do tabeli. Można używać więzów sprawdzających do kontrolowania zakresu wartości w kolumnach, formatów dat, czy innych warunków logicznych.

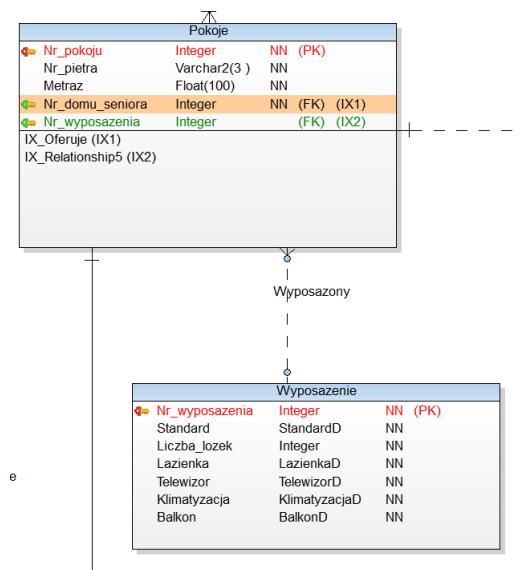
Poprzez zdefiniowanie i utrzymanie tych więzów integralności, baza danych pozostaje w spójnym stanie, co chroni ją przed wprowadzaniem nieprawidłowych, sprzecznych lub niekompletnych danych. System zarządzania bazą danych (DBMS) aktywnie monitoruje te więzy, uniemożliwiając operacje, które naruszają określone reguły integralności, co w rezultacie przyczynia się do poprawy jakości i niezawodności przechowywanych danych.

4.6 Proces denormalizacji – analiza i przykłady

Denormalizacja w kontekście baz danych to proces, w którym projektant bazy danych celowo odbiega od normalizacji, aby zwiększyć wydajność systemu lub uprościć zapytania. Normalizacja to proces organizowania struktury bazy danych w taki sposób, aby minimalizować zduplikowane dane, ale czasami, z różnych powodów, decyduje się zrezygnować z pewnych reguł normalizacji. Przyczyny, dla których można zastosować denormalizację, obejmują:

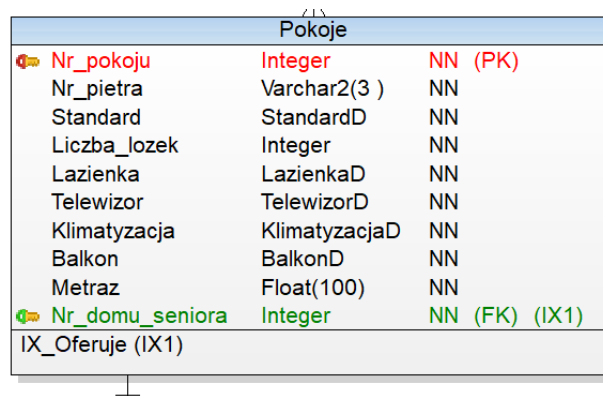
- **Szybkość zapytań:** Normalizacja może prowadzić do konieczności łączenia wielu tabel podczas zapytań, co może wpływać na wydajność. Denormalizacja może zredukować potrzebę łączenia poprzez przechowywanie pewnych danych redundancko.
- **Uproszczenie zapytań:** Czasami zapytania mogą być bardziej złożone w normalizowanej bazie danych, a denormalizacja może uprościć ich strukturę.
- **Optymalizacja dla konkretnej operacji:** Jeśli w systemie dominuje jedna operacja, można zoptymalizować strukturę bazy danych, aby ta operacja była bardziej wydajna.

Pierwszym przykładem denormalizacji jaki zastosowaliśmy było usunięcie słownika, tak aby tablica Pokoje zawierała wszystkie niezbędne informacje odnośnie pokoi.



Rysunek 9: Pokoje przed procesem denormalizacji

Poniżej przedstawiamy już zaaktualizowaną tablicę, w której jak widać dodaliśmy atrybuty z encji Wyposazenia, aby uprościć naszą bazę.



Rysunek 10: Pokoje po procesie denormalizacji

5 Faza fizyczna

5.1 Projekt transakcji i weryfikacja ich wykonalności

Nr	Transakcja	Zasób	Czy możliwe?
1	Dodawanie, modyfikowanie pensjonariuszy	Mieszkancy	tak
2	Dodawanie, modyfikowanie kart medycznych	Karty medyczne	tak
3	Wyświetlanie usług	Usługi	tak
4	Sprawdzanie przypisanych mieszkańców do usługi	Mieszkancy, Usługi _{Mieszkancy, Usługi}	tak
5	Dodawanie, modyfikowanie pracowników	Pracownicy	tak
6	Sprawdzanie, który pracownik zajmuje się danym pokojem	Pokoje, Opiekunowie _{Pokoje, Opiekunowie}	tak
7	Przegląd ukończonych uczelni przez danego lekarza	Lekarze, Lekarze _{Uczelnie, Uczelnie}	tak
8	Sprawdzanie, który pracownik zajmuje się danym mieszkańcem	Mieszkancy, Pracownicy _{Mieszkancy, Pracownicy}	tak
9	Przeglądanie certyfikatów danego opiekuna	Opiekunowie	tak
10	Sprawdzenie tytułu zawodowego lekarza	Lekarze	tak
11	Sprawdzanie, który pracownik wykonuje daną usługę	Usługi, Usługi _{Pracownicy, Pracownicy}	tak
12	Sprawdzenie standardu pokoju	Pokoje	tak

Tabela 10: Transakcje oraz zapytania i weryfikacja ich wykonalności

5.2 Strojanie bazy danych – dobór indeksów

```
— Create indexes for table Mieszkancy

CREATE INDEX IX_Zajmuje_sie ON Mieszkancy (Nr_domu_seniora)
/

CREATE INDEX IX_Mieszka ON Mieszkancy (Nr_pokoju)
/

— Create indexes for table Pokoje

CREATE INDEX IX_Oferuje ON Pokoje (Nr_domu_seniora)
/

— Create indexes for table Karty_medyczne

CREATE INDEX IX_Posiada ON Karty_medyczne (Nr_mieszkanca)
/

— Create indexes for table Pracownicy

CREATE INDEX IX_Zatrudnia ON Pracownicy (Nr_domu_seniora)
/

— Create indexes for table Uslugi

CREATE INDEX IX_Proponuje ON Uslugi (Nr_domu_seniora)
/
```

5.3 Skrypt SQL zakładający bazę danych

```
/*  
Created: 09.12.2023  
Modified: 09.12.2023  
Project: Dom seniora  
Model: Model logiczny  
Company: Politechnika Warszawska  
Author: Rafal Dadura, Juliusz Kuzyka  
Database: Oracle 18c  
*/
```

— Create sequences section —

```
CREATE SEQUENCE Nr_uczelni  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_domu_seniora  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_karty_medycznej  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_pokoju  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_mieszkanca  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_uslugi  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

```
CREATE SEQUENCE Nr_pracownika  
NOMAXVALUE  
NOMINVALUE  
CACHE 20  
/  

```

— *Create tables section* —

— *Table Uczelnie*

```
CREATE TABLE Uczelnie(  
    Nr_uczelni Integer NOT NULL,  
    Nazwa_uczelni Varchar2(50 ) NOT NULL,  
    Miasto Varchar2(30 ) NOT NULL,  
    Ulica Varchar2(30 ) NOT NULL,  
    Nr_domu Varchar2(5 ) NOT NULL,  
    Kod_pocztowy Char(6 ) NOT NULL  
)  
/
```

— *Add keys for table Uczelnie*

```
ALTER TABLE Uczelnie ADD CONSTRAINT PK_Uczelnie PRIMARY KEY (Nr_uczelni)  
/
```

— *Table Domy_seniora*

```
CREATE TABLE Domy_seniora(  
    Nr_domu_seniora Integer NOT NULL,  
    Nazwa_domu_seniora Varchar2(30 ) NOT NULL,  
    Nr_telefonu Varchar2(12 ) NOT NULL,  
    Email Varchar2(40 ) NOT NULL,  
    Miasto Varchar2(30 ) NOT NULL,  
    Ulica Varchar2(30 ) NOT NULL,  
    Nr_domu Varchar2(5 ) NOT NULL,  
    Kod_pocztowy Char(6 ) NOT NULL  
)  
/
```

— *Add keys for table Domy_seniora*

```
ALTER TABLE Domy_seniora ADD CONSTRAINT Unique_Identifier1 PRIMARY KEY  
(Nr_domu_seniora)  
/
```

— *Table Mieszkancy*

```
CREATE TABLE Mieszkancy(  
    Nr_mieszkanca Integer NOT NULL,  
    Nr_dokumentu_mieszkanca Varchar2(10 ) NOT NULL,  
    Imie Varchar2(20 ) NOT NULL,  
    Nazwisko Varchar2(20 ) NOT NULL,  
    Pesel Char(11 ) NOT NULL,  
    Plec Char(1 ) NOT NULL  
        CHECK (Plec IN ( 'K', 'M' ) ),  
    Email Varchar2(40 ) NOT NULL,  
    Nr_telefonu Varchar2(12 ) NOT NULL,  
    Miasto Varchar2(30 ) NOT NULL,  
    Ulica Varchar2(30 ) NOT NULL,  
    Nr_domu Varchar2(5 ) NOT NULL,  
    Kod_pocztowy Char(6 ) NOT NULL,
```



```

    Nr_domu_seniora Integer NOT NULL,
    Nr_pokoju Integer NOT NULL
)
/

— Add keys for table Mieszkancy

ALTER TABLE Mieszkancy ADD CONSTRAINT Unique_Identifier2 PRIMARY KEY
(Nr_mieszkanca)
/

— Table Pokoje

CREATE TABLE Pokoje(
    Nr_pokoju Integer NOT NULL,
    Nr_pietra Varchar2(3 ) NOT NULL,
    Metraz Float(100) NOT NULL,
    Standard Varchar2(7 ) NOT NULL
        CHECK (Standard IN ('zwykly', 'premium')),
    Liczba_lozek Integer NOT NULL,
    Lazienka Char(3 ) NOT NULL
        CHECK (Lazienka IN ('tak', 'nie')),
    Telewizor Char(3 ) NOT NULL
        CHECK (Telewizor IN ('tak', 'nie')),
    Klimatyzacja Char(3 ) NOT NULL
        CHECK (Klimatyzacja IN ('tak', 'nie')),
    Balkon Char(3 ) NOT NULL
        CHECK (Balkon IN ('tak', 'nie')),
    Nr_domu_seniora Integer NOT NULL
)
/

— Add keys for table Pokoje

ALTER TABLE Pokoje ADD CONSTRAINT Unique_Identifier3 PRIMARY KEY (Nr_pokoju)
/

— Table Karty_medyczne

CREATE TABLE Karty_medyczne(
    Nr_karty_medycznej Integer NOT NULL,
    Historia_chorob Varchar2(1200 ) NOT NULL,
    Historia_leczenia Varchar2(1200 ) NOT NULL,
    Szczepienia Varchar2(500 ) NOT NULL,
    Dane_rodzinne Varchar2(1200 ) NOT NULL,
    Wzrost Float(126) NOT NULL,
    Waga Float(120) NOT NULL,
    Grupa_krwi Char(2 ) NOT NULL
        CHECK (Grupa_krwi IN ('A', 'B', 'AB', '0')),
    Nr_mieszkanca Integer NOT NULL
)
/

— Add keys for table Karty_medyczne

```

```

ALTER TABLE Karty_medyczne ADD CONSTRAINT Unique_Identifier4 PRIMARY KEY
(Nr_karty_medycznej)
/

— Table Pracownicy

CREATE TABLE Pracownicy(
  Nr_pracownika Integer NOT NULL,
  Nr_dokumentu_pracownika Varchar2(10 ) NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(20 ) NOT NULL,
  Nr_telefonu Varchar2(12 ) NOT NULL,
  Email Varchar2(40 ) NOT NULL,
  Godziny_pracy Date NOT NULL,
  Nr_domu_seniora Integer NOT NULL
)
/

— Add keys for table Pracownicy

ALTER TABLE Pracownicy ADD CONSTRAINT Unique_Identifier5 PRIMARY KEY
(Nr_pracownika)
/

— Table Lekarze

CREATE TABLE Lekarze(
  Nr_licencji_lekarskiej Integer NOT NULL,
  Specjalizacja Varchar2(100 ) NOT NULL,
  Tytul_zawodowy Varchar2(20 ) NOT NULL,
  Nr_pracownika Integer NOT NULL
)
/

— Add keys for table Lekarze

ALTER TABLE Lekarze ADD CONSTRAINT Unique_Identifier6 PRIMARY KEY
(Nr_pracownika)
/

— Table Opiekunowie

CREATE TABLE Opiekunowie(
  Certyfikaty_z_zakresu_opieki Varchar2(500 ) NOT NULL,
  Doswiadczenie_zawodowe Varchar2(1200 ) NOT NULL,
  Obszar_odpowiedzialnosci Varchar2(1200 ) NOT NULL,
  Nr_pracownika Integer NOT NULL
)
/

— Add keys for table Opiekunowie

ALTER TABLE Opiekunowie ADD CONSTRAINT Unique_Identifier7 PRIMARY KEY
(Nr_pracownika)
/

```

```

— Table Usługi

CREATE TABLE Usługi(
    Nr_uslugi Integer NOT NULL,
    Nazwa_uslugi Varchar2(120 ) NOT NULL,
    Cena_uslugi Number(10,2) NOT NULL,
    Czas Date NOT NULL,
    Opis Varchar2(1200 ),
    Nr_domu_seniora Integer NOT NULL
)
/

— Add keys for table Usługi

ALTER TABLE Usługi ADD CONSTRAINT Unique_Identifier8 PRIMARY KEY (Nr_uslugi)
/

— Table Usługi_Pracownicy

CREATE TABLE Usługi_Pracownicy(
    Nr_uslugi Integer NOT NULL,
    Nr_pracownika Integer NOT NULL
)
/

— Table Usługi_Mieszkancy

CREATE TABLE Usługi_Mieszkancy(
    Nr_uslugi Integer NOT NULL,
    Nr_mieszkanca Integer NOT NULL
)
/

— Table Pracownicy_Mieszkancy

CREATE TABLE Pracownicy_Mieszkancy(
    Nr_pracownika Integer NOT NULL,
    Nr_mieszkanca Integer NOT NULL
)
/

— Table Lekarze_Uczelnie

CREATE TABLE Lekarze_Uczelnie(
    Nr_pracownika Integer NOT NULL,
    Nr_uczelni Integer NOT NULL
)
/

— Add keys for table Lekarze_Uczelnie

ALTER TABLE Lekarze_Uczelnie ADD CONSTRAINT PK_Lekarze_Uczelnie PRIMARY KEY
(Nr_pracownika ,Nr_uczelni)
/

— Table Opiekunowie_Pokoje

```

```

CREATE TABLE Opiekunowie_Pokoje(
  Nr_pokoju Integer NOT NULL,
  Nr_pracownika Integer NOT NULL,
  Sprzatanie Char(3 ) NOT NULL
    CHECK (Sprzatanie IN ('tak', 'nie')),
  Czas_opieki Date NOT NULL,
  Dzień_tygodnia Char(12 ) NOT NULL
    CHECK (Dzień_tygodnia IN ('Poniedziałek','Wtorek','Środa','Czwartek',
    'Piątek','Sobota','Niedziela'))
)
/

— Add keys for table Opiekunowie_Pokoje

ALTER TABLE Opiekunowie_Pokoje ADD CONSTRAINT PK_Opiekunowie_Pokoje PRIMARY KEY
(Nr_pokoju,Nr_pracownika)
/

— Trigger for sequence Nr_domu_seniora for column Nr_domu_seniora
in table Domy_seniora —————
CREATE OR REPLACE TRIGGER ts_Domy_seniora_Nr_domu_seniora BEFORE INSERT
ON Domy_seniora FOR EACH ROW
BEGIN
  :new.Nr_domu_seniora := Nr_domu_seniora.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Domy_seniora_Nr_domu_seniora AFTER UPDATE OF
Nr_domu_seniora
ON Domy_seniora FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(−20010,'Cannot_update_column_Nr_domu_seniora
  in_table_Domy_seniora_as_it_uses_sequence. ');
END;
/

— Trigger for sequence Nr_mieszkanca for column Nr_mieszkanca
in table Mieszkancy —————
CREATE OR REPLACE TRIGGER ts_Mieszkancy_Nr_mieszkanca BEFORE INSERT
ON Mieszkancy FOR EACH ROW
BEGIN
  :new.Nr_mieszkanca := Nr_mieszkanca.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Mieszkancy_Nr_mieszkanca
AFTER UPDATE OF Nr_mieszkanca
ON Mieszkancy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(−20010,'Cannot_update_column_Nr_mieszkanca
  in_table_Mieszkancy_as_it_uses_sequence. ');
END;
/

— Trigger for sequence Nr_pokoju for column Nr_pokoju in table Pokoje —————
CREATE OR REPLACE TRIGGER ts_Pokoje_Nr_pokoju BEFORE INSERT
ON Pokoje FOR EACH ROW

```

```

BEGIN
    :new.Nr_pokoju := Nr_pokoju.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pokoje_Nr_pokoju AFTER UPDATE OF Nr_pokoju
ON Pokoje FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot_update_column_Nr_pokoju
    in_table_Pokoje_as_it_uses_sequence. ');
END;
/

-- Trigger for sequence Nr_karty_medycznej for column Nr_karty_medycznej
in table Karty_medyczne
CREATE OR REPLACE TRIGGER ts_Karty_medyczne_Nr_karty_medycznej BEFORE INSERT
ON Karty_medyczne FOR EACH ROW
BEGIN
    :new.Nr_karty_medycznej := Nr_karty_medycznej.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Karty_medyczne_Nr_karty_medycznej
AFTER UPDATE OF Nr_karty_medycznej
ON Karty_medyczne FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot_update_column_Nr_karty_medycznej
    in_table_Karty_medyczne_as_it_uses_sequence. ');
END;
/

-- Trigger for sequence Nr_pracownika for column Nr_pracownika
in table Pracownicy
CREATE OR REPLACE TRIGGER ts_Pracownicy_Nr_pracownika BEFORE INSERT
ON Pracownicy FOR EACH ROW
BEGIN
    :new.Nr_pracownika := Nr_pracownika.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_Nr_pracownika
AFTER UPDATE OF Nr_pracownika
ON Pracownicy FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot_update_column_Nr_pracownika
    in_table_Pracownicy_as_it_uses_sequence. ');
END;
/

-- Trigger for sequence Nr_uslugi for column Nr_uslugi in table Uslugi
CREATE OR REPLACE TRIGGER ts_Uslugi_Nr_uslugi BEFORE INSERT
ON Uslugi FOR EACH ROW
BEGIN
    :new.Nr_uslugi := Nr_uslugi.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Uslugi_Nr_uslugi AFTER UPDATE OF Nr_uslugi
ON Uslugi FOR EACH ROW
BEGIN

```

```

    RAISE_APPLICATION_ERROR(−20010, 'Cannot_update_column_Nr_uslugi
    in_table_Uslugi_as_it_uses_sequence. ');
END;
/

— Trigger for sequence Nr_uczelni for column Nr_uczelni
in table Uczelnie
CREATE OR REPLACE TRIGGER ts_Uczelnie_Nr_uczelni BEFORE INSERT
ON Uczelnie FOR EACH ROW
BEGIN
    :new.Nr_uczelni := Nr_uczelni.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Uczelnie_Nr_uczelni AFTER UPDATE OF Nr_uczelni
ON Uczelnie FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(−20010, 'Cannot_update_column_Nr_uczelni
    in_table_Uczelnie_as_it_uses_sequence. ');
END;
/

— Create foreign keys (relationships) section —

ALTER TABLE Mieszkancy ADD CONSTRAINT Zajmuje_sie FOREIGN KEY (Nr_domu_seniora)
REFERENCES Domy_seniora (Nr_domu_seniora)
/

ALTER TABLE Karty_medyczne ADD CONSTRAINT Posiada FOREIGN KEY (Nr_mieszkanca)
REFERENCES Mieszkancy (Nr_mieszkanca)
/

ALTER TABLE Pokoje ADD CONSTRAINT Oferuje FOREIGN KEY (Nr_domu_seniora)
REFERENCES Domy_seniora (Nr_domu_seniora)
/

ALTER TABLE Uslugi ADD CONSTRAINT Proponuje FOREIGN KEY (Nr_domu_seniora)
REFERENCES Domy_seniora (Nr_domu_seniora)
/

ALTER TABLE Pracownicy ADD CONSTRAINT Zatrudnia FOREIGN KEY (Nr_domu_seniora)
REFERENCES Domy_seniora (Nr_domu_seniora)
/

ALTER TABLE Mieszkancy ADD CONSTRAINT Mieszka FOREIGN KEY (Nr_pokoju)
REFERENCES Pokoje (Nr_pokoju)

```

```

/
ALTER TABLE Lekarze_Uczelnie ADD CONSTRAINT Lekarz_Ukonczyl FOREIGN KEY
(Nr_pracownika) REFERENCES Lekarze (Nr_pracownika)
/

ALTER TABLE Lekarze_Uczelnie ADD CONSTRAINT Ukonczyl_Uczelnie FOREIGN KEY
(Nr_uczelni) REFERENCES Uczelnie (Nr_uczelni)
/

ALTER TABLE Opiekunowie_Pokoje ADD CONSTRAINT Pokoje_Opiekuje FOREIGN KEY
(Nr_pokoju) REFERENCES Pokoje (Nr_pokoju)
/

ALTER TABLE Opiekunowie_Pokoje ADD CONSTRAINT Opiekuje_Opiekun FOREIGN KEY
(Nr_pracownika) REFERENCES Opiekunowie (Nr_pracownika)
/

```

5.4 Przykładowe dane w naszej bazie danych

```

— Domy_seniora
INSERT INTO Domy_seniora (Nazwa_domu_seniora, Nr_telefonu, Email, Miasto,
Ulica, Nr_domu, Kod_pocztowy) VALUES ('Dom_Seniora_Wesole_Lata', '123456789',
'wesolelata@example.com', 'Warszawa', 'S?oneczna', '123', '00-001');

— Uczelnie
INSERT INTO Uczelnie (Nazwa_uczelni, Miasto, Ulica, Nr_domu, kod_pocztowy)
VALUES ('Politechnika_Warszawska', 'Warszawa', 'Nowowiejska', '123', '00-001');
INSERT INTO Uczelnie (Nazwa_uczelni, Miasto, Ulica, Nr_domu, kod_pocztowy)
VALUES ('Wyzsza_Szkola_Lansu_i_Bansu', 'Krakow', 'Ulicy', '56', '20-401');
INSERT INTO Uczelnie (Nazwa_uczelni, Miasto, Ulica, Nr_domu, kod_pocztowy)
VALUES ('Massachuchet_University', 'Massachuchet', 'Clay', '1', '03-586');

— Uslugi
INSERT INTO Uslugi (Nazwa_uslugi, Cena_uslugi, Czas, Opis, Nr_domu_seniora)
VALUES ('Opieka_pielegniarska', 150.00, TO_DATE('08:00', 'HH24:MI'),
'Opieka_nad_mieszkancami', 1);
INSERT INTO Uslugi (Nazwa_uslugi, Cena_uslugi, Czas, Opis, Nr_domu_seniora)
VALUES ('Kino', 300.00, TO_DATE('04:30', 'HH24:MI'),
'Wspolne_oglądanie_maratonu_Star_Wars', 1);

— Pokoje
INSERT INTO Pokoje (Nr_pietra, Metraz, Standard, Liczba_lozek, Lazienka,
Telewizor, Klimatyzacja, Balkon, Nr_domu_seniora)
VALUES ('1', 20.5, 'zwykly', 1, 'tak', 'nie', 'nie', 'nie', 1);
INSERT INTO Pokoje (Nr_pietra, Metraz, Standard, Liczba_lozek, Lazienka,
Telewizor, Klimatyzacja, Balkon, Nr_domu_seniora)
VALUES ('1', 40.6, 'premium', 4, 'tak', 'tak', 'tak', 'tak', 1);

— Mieszkancy
INSERT INTO Mieszkancy (Nr_dokumentu_mieszkanca, Imie, Nazwisko, Pesel, Plec,

```

```
Email, Nr_telefonu, Miasto, Ulica, Nr_domu, Kod_pocztowy, Nr_domu_seniora,
Nr_pokoju) VALUES ('ABC1234567', 'Anna', 'Kowalska', '91010112345', 'K',
'anna@example.com', '987344321', 'Poznan', 'Nocna', '132', '00-355', 1, 1);
INSERT INTO Mieszkancy (Nr_dokumentu_mieszkanca, Imie, Nazwisko, Pesel, Plec,
Email, Nr_telefonu, Miasto, Ulica, Nr_domu, Kod_pocztowy, Nr_domu_seniora,
Nr_pokoju) VALUES ('XYZ7890123', 'Jan', 'Nowak', '85020254321', 'M',
'jan@example.com', '123456789', 'Warszawa', 'Dzielna', '45', '02-100', 1, 2);
INSERT INTO Mieszkancy (Nr_dokumentu_mieszkanca, Imie, Nazwisko, Pesel, Plec,
Email, Nr_telefonu, Miasto, Ulica, Nr_domu, Kod_pocztowy, Nr_domu_seniora,
Nr_pokoju) VALUES ('DEF4567890', 'Marta', 'Zalewska', '88030367890', 'K',
'marta@example.com', '654987321', 'Krakow', 'S?owackiego', '78', '30-300', 1,
1);
```

— *Karty_medyczne*

```
INSERT INTO Karty_medyczne (Historia_chorob, Historia_leczenia, Szczepienia,
Dane_rodzinne, Wzrost, Waga, Grupa_krwi, Nr_mieszkanca) VALUES ('przykladowa
historia_chorob', 'przykladowa_historia_leczenia', 'przykladowe_szczepienia',
'przykladowe_dane_rodzinne', 160, 60, 'A', 1);
INSERT INTO Karty_medyczne (Historia_chorob, Historia_leczenia, Szczepienia,
Dane_rodzinne, Wzrost, Waga, Grupa_krwi, Nr_mieszkanca) VALUES ('inne
schorzenie', 'inne_leczenie', 'inne_szczepienia', 'inne_dane_rodzinne', 175,
70, 'B', 2);
INSERT INTO Karty_medyczne (Historia_chorob, Historia_leczenia, Szczepienia,
Dane_rodzinne, Wzrost, Waga, Grupa_krwi, Nr_mieszkanca) VALUES ('nowa_historia
chorob', 'nowe_leczenie', 'nowe_szczepienia', 'nowe_dane_rodzinne', 180, 75,
'AB', 3);
```

— *Pracownicy*

```
INSERT INTO Pracownicy (Nr_dokumentu_pracownika, Imie, Nazwisko, Nr_telefonu,
Email, Godziny_pracy, Nr_domu_seniora) VALUES ('XYZ9876543', 'Jan', 'Nowak',
'557124556', 'jan.nowak@example.com', TO_DATE('08:00', 'HH24:MI'), 1);
INSERT INTO Pracownicy (Nr_dokumentu_pracownika, Imie, Nazwisko, Nr_telefonu,
Email, Godziny_pracy, Nr_domu_seniora) VALUES ('UF569I84Y', 'Tomasz',
'Problem', '557912456', 'tomasz.problem@example.com', TO_DATE('08:00',
'HH24:MI'), 1);
INSERT INTO Pracownicy (Nr_dokumentu_pracownika, Imie, Nazwisko, Nr_telefonu,
Email, Godziny_pracy, Nr_domu_seniora) VALUES ('JK789L32P', 'Alicja',
'Rozwiazanie', '689234567', 'alicja.rozwiazanie@example.com', TO_DATE('09:00',
'HH24:MI'), 1);
INSERT INTO Pracownicy (Nr_dokumentu_pracownika, Imie, Nazwisko, Nr_telefonu,
Email, Godziny_pracy, Nr_domu_seniora) VALUES ('LM456N78O', 'Marcin',
'Pytanie', '987654321', 'marcin.pytanie@example.com', TO_DATE('10:30',
'HH24:MI'), 1);
```

— *Lekarze*

```
INSERT INTO Lekarze (Nr_licencji_lekarskiej, Specjalizacja, Tytul_zawodowy,
Nr_pracownika) VALUES (123456, 'Chirurgia', 'Dr.', 1);
INSERT INTO Lekarze (Nr_licencji_lekarskiej, Specjalizacja, Tytul_zawodowy,
Nr_pracownika) VALUES (98765f, 'Lekarz_pediatra', 'Dr.', 3);
```

— *Opiekunowie*

```
INSERT INTO Opiekunowie (Certyfikaty_z_zakresu_opieki, Doswiadczenie_zawodowe,
Obszar_odpowiedzialnosci, Nr_pracownika) VALUES ('Certyfikaty...',
```



```

'Doswiadczenie... ', 'Opieka_nad_seniorami... ', 2);
INSERT INTO Opiekunowie (Certyfikaty_z_zakresu_opieki, Doswiadczenie_zawodowe,
Obszar_odpowiedzialnosci, Nr_pracownika) VALUES ('Ciekawe_certyfikaty ',
'Doswiadczenie_albo_jego_brak ', 'Opieka_nad_seniorami_i_sprz?tanie ', 4);

— Usługi_Pracownicy
INSERT INTO Usługi_Pracownicy VALUES (1, 1);
INSERT INTO Usługi_Pracownicy VALUES (2, 2);

— Pracownicy_Mieszkancy
INSERT INTO Pracownicy_Mieszkancy VALUES (1, 1);
INSERT INTO Pracownicy_Mieszkancy VALUES (3, 2);
INSERT INTO Pracownicy_Mieszkancy VALUES (3, 3);
INSERT INTO Pracownicy_Mieszkancy VALUES (2, 3);

— Usługi_Mieszkancy
INSERT INTO Usługi_Mieszkancy VALUES (1, 1);
INSERT INTO Usługi_Mieszkancy VALUES (2, 2);

— Lekarz_Uczelnie
INSERT INTO Lekarze_Uczelnie VALUES (1, 1);
INSERT INTO Lekarze_Uczelnie VALUES (1, 2);
INSERT INTO Lekarze_Uczelnie VALUES (3, 3);

— Opiekunowie_Pokoje
INSERT INTO Opiekunowie_Pokoje VALUES (1, 4, 'tak ', TO_DATE('04:00 ',
'HH24:MI '), 'Poniedzialek ');
INSERT INTO Opiekunowie_Pokoje VALUES (2, 2, 'nie ', TO_DATE('06:15 ',
'HH24:MI '), 'Piatek ');

```

5.5 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych

```

— szukanie pracownika, który jest lekarzem
SELECT * FROM Pracownicy WHERE Nr_pracownika IN(SELECT Nr_pracownika FROM Lekarze);

— szukanie mieszkancow, którzy korzystaja z uslugi nr 2
SELECT * FROM Mieszkancy WHERE Nr_mieszkanca IN(SELECT Nr_mieszkanca FROM
Usługi_Mieszkancy WHERE Usługi_Mieszkancy.Nr_uslugi = 2);
— wyswietlenie wszystkich uslug
SELECT * FROM Usługi;
— szukanie lekarzy, którzy sa doktorami
SELECT * FROM Lekarze WHERE Tytul_zawodowy = 'Dr.';
— szukanie pracownika, ktorego imie zaczyna sie na J i jest lekarzem z tytulem
— doktora
SELECT * FROM Pracownicy WHERE Imie LIKE 'J%' AND Nr_pracownika
IN(SELECT Nr_pracownika FROM Lekarze WHERE Tytul_zawodowy = 'Dr. ');
— szukanie pracownika, który jest opiekunem i wykonuje usluge, ktora
— kosztuje wiecej niz 200
SELECT * FROM Pracownicy WHERE Nr_pracownika IN(SELECT Nr_pracownika
FROM Opiekunowie) AND Nr_pracownika IN(SELECT Nr_pracownika FROM Usługi_Pracownicy
WHERE Nr_uslugi IN (SELECT Nr_uslugi FROM Usługi WHERE Cena_uslugi > 200));
— pokazanie wszystkich uslug tanszych od 300
SELECT * FROM Usługi WHERE Cena_uslugi < 300;

```

Ponizej znajdują się wyniki podanych zapytań oraz poleceń.

Dane Pacjenta i Rodzina									
Imię i Nazwisko	Adres	Telefon	PESEL	Ulica	Nr. domu	Seniors	Godziny	Nr. domu	Seniors
Jan Nowak	ul. Główna 15	123 456 789	1234567890	ul. Główna	15	02-100	10:00	15	02-100
Dane Rodziny									
Imię i Nazwisko	Adres	Telefon	PESEL	Ulica	Nr. domu	Seniors	Godziny	Nr. domu	Seniors
Alicja Nowak	ul. Główna 15	123 456 789	1234567890	ul. Główna	15	02-100	10:00	15	02-100
Dane Usług									
Nazwa usługi	Cena	Opis	Nr. usługi	Nr. domu	Seniors	Godziny	Nr. domu	Seniors	Godziny
Opieka pielęgniarska	150	Opieka nad mieszkańcami	1	15	02-100	10:00	15	02-100	10:00
Kino	300	Wspólne oglądanie maratonu Star Wars	1	15	02-100	10:00	15	02-100	10:00
Dane Pracownika									
Nazwa usługi	Cena	Opis	Nr. usługi	Nr. domu	Seniors	Godziny	Nr. domu	Seniors	Godziny
Opieka pielęgniarska	150	Opieka nad mieszkańcami	1	15	02-100	10:00	15	02-100	10:00
Kino	300	Wspólne oglądanie maratonu Star Wars	1	15	02-100	10:00	15	02-100	10:00
Dane Pracownika									
Nazwa usługi	Cena	Opis	Nr. usługi	Nr. domu	Seniors	Godziny	Nr. domu	Seniors	Godziny
Opieka pielęgniarska	150	Opieka nad mieszkańcami	1	15	02-100	10:00	15	02-100	10:00
Kino	300	Wspólne oglądanie maratonu Star Wars	1	15	02-100	10:00	15	02-100	10:00

Rysunek 11: Wyniki poleceń oraz zapytań