

Bazy danych i Big Data - Projekt

Projekt i implementacja bazy danych

Temat nr 6: Spółdzielnia mieszkaniowa

1. Zakres i cel projektu (opis założeń funkcjonalnych projektowanej bazy danych)	3
2. Definicja systemu	4
2.1 Perspektywy użytkowników	5
3. Model koncepcyjny	6
3.1 Definicja zbiorów encji określonych w projekcie (decyzje projektowe)	6
3.2 Ustalenie związków między encjami i ich typów	6
3.3 Określenie atrybutów i ich dziedzin	8
3.4 Dodatkowe reguły integralnościowe (reguły biznesowe)	Błąd! Nie zdefiniowano zakładki.
3.5 Klucze kandydujące i główne (decyzje projektowe)	12
3.6 Schemat ER na poziomie koncepcyjnym	Błąd! Nie zdefiniowano zakładki.
3.7 Problem pułapek szczelinowych i wachlarzowych - analiza i przykłady	13
4. Model logiczny	16
4.1 Charakterystyka modelu relacyjnego	17
4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady	17
4.3 Proces normalizacji - analiza i przykłady	19
4.4 Schemat ER na poziomie modelu logicznego	20
4.5 Więzy integralności	22
4.6 Proces denormalizacji - analiza i przykłady	22
5. Faza fizyczna	24
5.1 Projekt transakcji i weryfikacja ich wykonywalności	24
5.2 Strojenie bazy danych - dobór indeksów	26
5.3 Skrypt SQL zakładający bazę danych	28
5.4 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych	43
Bibliografia	45

1. Zakres i cel projektu (opis założeń funkcjonalnych projektowanej bazy danych)

Celem projektu było zaprojektowanie oraz zaimplementowanie bazy danych typu relacyjnego. Składał się on z 3 faz:

1. Fazy projektowania conceptualnego - podczas której zdefiniowano podstawowe encje schematu bazy danych oraz związki między nimi bez uwzględniania docelowego relacyjnego typu bazy danych.
2. Fazy projektowania logicznego - podczas której model conceptualny uzyskany w fazie 1 przekształcono do relacyjnego modelu danych poprzez utworzenie odpowiadających encjom tabel, jak również związków między nimi przy wykorzystaniu par klucz główny - klucz obcy.
3. Fazy projektowania fizycznego - obejmującej dostosowanie modelu uzyskanego w fazie 2 do specyfiki wybranego silnika zarządzania bazą danych (Oracle 19c). Dzięki zastosowaniu narzędzia Toad Data Modeler 7.2 faza ta przebiegła w dużej mierze automatycznie, należało jedynie dostosować typy danych atrybutów tabel utworzonych w fazie logicznej do typów danych oferowanych przez silnik.

Zrealizowana baza danych ma na celu usprawnić zarządzanie spółdzielnią mieszkaniową. Spółdzielnia zarządza blokami, w których znajdują się mieszkania którymi dysponuje, Mieszkania te posiadają właściciela(-li) lub nie (wówczas mieszkanie jest własnością spółdzielni). W mieszkaniach mogą (ale nie muszą) mieszkać jeden lub więcej mieszkańców, przy czym mieszkańcy mogą być jednocześnie właścicielami mieszkania. Za użytkowanie mieszkania naliczane są opłaty w postaci ustalonego z góry czynszu oraz obliczanych na podstawie zużycia (konkretnego medium) opłat za media. Spółdzielnia zatrudnia pracowników, których podzielono na pracowników pracujących w siedzibie (biurze) spółdzielni (np. sekretarka, dyrektor, prezes) oraz pracowników pracujących "w terenie", czyli na stałe w konkretnym bloku (dozorca, sprzątaczką) lub tylko wtedy, gdy zajdzie taka potrzeba (hydraulik, elektryk). Pracownikom wypłacane jest comiesięczne wynagrodzenie składające się ze stałej pensji oraz ewentualnego dodatku. Spółdzielnia mieszkaniowa korzysta również z usług firm zewnętrznych takich jak wywóz śmieci, czy dostawy prądu, gazu. Spółdzielnia posiada również zarząd, w skład którego wchodzi niektórzy pracownicy pracujący w biurze (np. prezes).

2. Definicja systemu

Definicja systemu zakłada następujące funkcjonalności:

- podgląd informacji o spółdzielni mieszkaniowej
- dodawanie/modyfikowanie/usuwanie informacji o spółdzielni mieszkaniowej
- podgląd informacji o pracownikach
- dodawanie/modyfikowanie/usuwanie informacji o pracownikach
- podgląd informacji o pracownikach jako pracownikach biurowych
- dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach biurowych
- podgląd informacji o pracownikach jako pracownikach terenowych
- dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach terenowych
- podgląd informacji o blokach mieszkalnych
- dodawanie/modyfikowanie/usuwanie informacji o blokach mieszkalnych
- podgląd informacji o mieszkaniach
- dodawanie/modyfikowanie/usuwanie informacji o mieszkaniach
- podgląd informacji o adresach
- dodawanie/modyfikowanie/usuwanie informacji o adresach
- podgląd informacji o zarządach spółdzielni
- dodawanie/modyfikowanie/usuwanie informacji o zarządach spółdzielni
- podgląd informacji o pracownikach jako pracownikach biurowych wchodzących w skład zarządu spółdzielni
- dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach biurowych wchodzących w skład zarządu spółdzielni
- podgląd informacji o firmach usługowych
- dodawanie/modyfikowanie/usuwanie informacji o firmach usługowych
- podgląd informacji o firmach usługowych, których usługi wykonywane są w konkretnym bloku mieszkalnym
- dodawanie/modyfikowanie/usuwanie informacji o firmach usługowych, których usługi wykonywane są w konkretnym bloku mieszkalnym
- podgląd informacji o blokach mieszkalnych w których pełnią dozór pracownicy terenowi
- dodawanie/modyfikowanie/usuwanie informacji o blokach mieszkalnych w których pełnią dozór pracownicy terenowi
- podgląd informacji o wynagrodzeniach pracowników
- dodawanie/modyfikowanie/usuwanie informacji o wynagrodzeniach pracowników
- podgląd informacji o osobach
- dodawanie/modyfikowanie/usuwanie informacji o osobach
- podgląd informacji o osobach zamieszkujących mieszkanie
- dodawanie/modyfikowanie/usuwanie informacji o osobach zamieszkujących mieszkanie
- podgląd informacji o osobach posiadających mieszkanie
- dodawanie/modyfikowanie/usuwanie informacji o osobach posiadających mieszkanie
- podgląd informacji o mediach zużywanych w mieszkaniu
- dodawanie/modyfikowanie/usuwanie o mediach zużywanych w mieszkaniu

- podgląd informacji o opłatach związanych z mieszkaniem
- dodawanie/modyfikowanie/usuwanie o opłatach związanych z mieszkaniem
- podgląd informacji o czynszach mieszkań
- dodawanie/modyfikowanie/usuwanie o czynszach mieszkań
- podgląd informacji o opłatach związanych z mieszkaniem jako opłatach za media
- dodawanie/modyfikowanie/usuwanie informacji o opłatach związanych z mieszkaniem jako opłatach za media
- podgląd informacji o opłatach związanych z mieszkaniem jako opłatach za czynsz
- dodawanie/modyfikowanie/usuwanie informacji o opłatach związanych z mieszkaniem jako opłatach za czynsz

2.1 Perspektywy użytkowników

Wśród potencjalnych użytkowników mogących w przyszłości korzystać z opisanego powyżej systemu, wyróżnia się:

- **Dyrektora generalnego (zarządcę) spółdzielni mieszkaniowej** - posiada dostęp do wszystkich informacji zawartych w bazie oraz ma możliwość ich modyfikacji
- **Członka zarządu** - przywileje takie same jak w przypadku zarządcy spółdzielni
- **Pracownika biurowego** - dostęp do danych administracyjnych spółdzielni; ograniczony dostęp do informacji dotyczących innych pracowników oraz ograniczona możliwość ich modyfikacji (np. księgowy ma dostęp do wynagrodzeń zarówno każdego pracownika, jak i dyrektora głównego i ma możliwość ich modyfikacji, natomiast sekretarka nie ma takiego przywileju)
- **Pracownika terenowego** - ograniczony dostęp do informacji dotyczących innych pracowników oraz do danych na temat bloków i mieszkań
- **Użytkownika mieszkania (mieszkaniec, właściciel)** - dostęp do informacji na temat mieszkania oraz innych mieszkańców; ograniczony dostęp do danych osobowych pracowników

3. Model konceptualny

3.1 Definicja zbiorów encji określonych w projekcie (decyzje projektowe)

Encje na poziomie konceptualnym:

- **Spoldzielnia_mieszkaniowa** - główna encja przechowująca ogólne informacje o spółdzielni mieszkaniowej znajdującej się w bazie
- **Zarząd_spoldzielni** - encja zawierająca informacje o zarządzie spółdzielni mieszkaniowej
- **Pracownik** - encja zawierająca dane o pracowniku pracującym w spółdzielni mieszkaniowej
- **Pracownik_biurowy** - encja wyodrębniająca pracownika spółdzielni mieszkaniowej jako pracownika biurowego
- **Pracownik_terenowy** - encja wyodrębniająca pracownika spółdzielni mieszkaniowej jako pracownika terenowego
- **Blok** - encja zawierająca informacje na temat bloku administrowanego przez spółdzielnię mieszkaniową
- **Mieszkanie** - encja przechowująca informacje na temat mieszkania znajdującego się w bloku administrowanym przez spółdzielnię mieszkaniową
- **Firma_uslugowa** - encja przechowująca dane na temat zewnętrznej firmy usługowej, która świadczy usługi na rzecz spółdzielni mieszkaniowej

3.2 Ustalenie związków między encjami i ich typów

Nazwa relacji	Krotność	Typ uczestnictwa	Opis
Spoldzielnia_mieszkaniowa - Zarząd_spoldzielni	jeden do wielu	spółdzielnia mieszkaniowa jest obowiązkowa; zarząd spółdzielni jest nieobowiązkowy	Spółdzielnia mieszkaniowa może nie mieć żadnego zarządu, ale może też mieć wiele w okresie swojego istnienia; zarząd spółdzielni musi zarządzać dokładnie jedną spółdzielnią mieszkaniową
Spoldzielnia_mieszkaniowa - Blok	jeden do wielu	spółdzielnia mieszkaniowa jest obowiązkowa; blok jest nieobowiązkowy	Spółdzielnia mieszkaniowa może administrować wiele bloków, ale może też nie administrować żadnego; blok może być administrowany przez dokładnie jedną spółdzielnię mieszkaniową

Spółdzielnia_mieszkaniowa - Pracownik	jeden do wielu	spółdzielnia mieszkaniowa jest obowiązkowa; pracownik jest nieobowiązkowy	Spółdzielnia mieszkaniowa może zatrudniać wielu pracowników, ale też może nie mieć żadnego; pracownik musi być zatrudniony w dokładnie jednej spółdzielni mieszkaniowej
Spółdzielnia_mieszkaniowa - Firma_usługowa	jeden do wielu	spółdzielnia mieszkaniowa jest obowiązkowa; firma usługowa jest nieobowiązkowa	Spółdzielnia mieszkaniowa może zlecać wykonanie usług wielu firmom usługowym, ale może też nie zlecać wcale; firma usługowa może dostać zlecenie od dokładnie jednej spółdzielni mieszkaniowej
Blok - Mieszkanie	jeden do wielu	blok jest obowiązkowy; mieszkanie jest nieobowiązkowe	Blok może posiadać wiele mieszkań, ale może też nie posiadać wcale (encja zawiera informacje o bloku, ale nie żadne konkretne mieszkanie nie jest administrowane przez spółdzielnię); Mieszkanie może znajdować się tylko w jednym bloku
Firma_usługowa - Blok	wielu do wielu	firma usługowa jest nieobowiązkowa; blok jest nieobowiązkowy	Firma usługowa może świadczyć usługi w wielu blokach, lub też nie świadczyć wcale; blok może mieć wykonywanych wiele usług, ale też może nie mieć ich wcale
Pracownik_biurowy - Zarząd_spółdzielni	wielu do wielu	pracownik biurowy jest obowiązkowy; zarząd spółdzielni jest nieobowiązkowy	Pracownik biurowy może być członkiem wielu zarządów spółdzielni, ale też może nie być nim wcale; Zarząd spółdzielni może mieć wielu członków będących pracownikami biurowymi, ale musi mieć przynajmniej jednego
Pracownik_terenowy - Blok	wielu do wielu	pracownik terenowy jest nieobowiązkowy; blok jest nieobowiązkowy	Pracownik terenowy może nadzorować wiele bloków, jednak może też nie nadzorować żadnego; Blok może być nadzorowany przez wielu pracowników terenowych, ale może też nie być nadzorowany przez żadnego

3.3 Określenie atrybutów i ich dziedzin

Spoldzielnia_mieszkaniowa

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Spoldzielnia_mieszkaniowa_ID	Integer	obowiązkowy	Unikatowy identyfikator spółdzielni mieszkaniowej.
Nazwa	VarChar(50)	obowiązkowy	Nazwa spółdzielni mieszkaniowej.
Adres_siedziby	VarChar(300)	obowiązkowy	Adres siedziby spółdzielni mieszkaniowej, pole złożone (ulica, numer domu, numer lokalu, kod pocztowy, miasto).
Numer_telefonu	VarChar(12)	obowiązkowy	Numer telefonu do spółdzielni mieszkaniowej.
Email	VarChar(40)	obowiązkowy	Adres e-mail spółdzielni mieszkaniowej.
Strona_internetowa	VarChar(40)	nieobowiązkowy	Adres strony internetowej spółdzielni mieszkaniowej.
Data_zalozenia	Date	obowiązkowy	Data założenia spółdzielni mieszkaniowej.

Zarząd_spoldzielni

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Zarząd_spoldzielni_ID	Integer	obowiązkowy	Unikatowy identyfikator zarządu spółdzielni mieszkaniowej.
Członek_zarządu	VarChar(400)	nieobowiązkowy	Członkowie zarządu. Pole wielowartościowe, złożone (imię, nazwisko).
Data_zawiazania	Date	obowiązkowy	Data zawiązania zarządu spółdzielni.
Data_rozwiazania	Date	nieobowiązkowy	Data rozwiązania zarządu spółdzielni.

Pracownik

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Pracownik_ID	Integer	obowiązkowy	Unikatowy identyfikator pracownika.
Imie	VarChar(20)	obowiązkowy	Pierwsze imię pracownika.
Drugie_imie	VarChar(20)	nieobowiązkowy	Drugie imię pracownika.
Nazwisko	VarChar(30)	obowiązkowy	Nazwisko pracownika.
PESEL	VarChar(11)	nieobowiązkowy	Numer PESEL pracownika.
Data_urodzenia	Date	obowiązkowy	Data urodzenia pracownika.
Adres_zamieszkania	VarChar(300)	obowiązkowy	Adres zamieszkania pracownika.
Numer_telefonu	VarChar(12)	obowiązkowy	Numer telefonu pracownika.
Plec	Plec_D(K,M)	obowiązkowy	Płeć pracownika.
Numer_konta	Character(24)	obowiązkowy	Numer konta pracownika.
Wynagrodzenie	Money	obowiązkowy	Wynagrodzenie miesięczne pracownika, pole wielowartościowe, pole segmentowe (data wynagrodzenia, kwota).
Data_zatrudnienia	Date	obowiązkowy	Data zatrudnienia pracownika w spółdzielni mieszkaniowej.
Data_zwolnienia	Date	nieobowiązkowy	Data rozwiązywania umowy z pracownikiem spółdzielni mieszkaniowej.

Pracownik_biurowy

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Rodzaj_pracownika_biurowego	Rodzaj_pracownika_biurowego_D (sekretarka, księgowy, dyrektor, prezes)	obowiązkowy	Rodzaj pracownika biurowego. Dozwolone wartości (sekretarka, księgowy, dyrektor, prezes).

Email_sluzbowy	VarChar(40)	obowiązkowy	Adres e-mail pracownika biurowego.
Numer_burka	VarChar(4)	obowiązkowy	Numer biurka, przy którym pracuje pracownik biurowy.

Pracownik_terenowy

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Rodzaj_pracownika_terenowego	Rodzaj_pracownika_terenowego_D(dozorca, sprzątaczką, hydraulik, elektryk)	obowiązkowy	Rodzaj pracownika terenowego. Dozwolone wartości: (dozorca, sprzątaczką, hydraulik, elektryk).
Czy_praca_zmianowa	Boolean	obowiązkowy	Czy pracownik pracuje w wymiarze zmianowym.

Blok

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Blok_ID	Integer	obowiązkowy	Unikatowy identyfikator bloku.
Adres	VarChar(300)	obowiązkowy	Adres bloku, pole złożone (ulica, numer domu, kod pocztowy, miasto).
Liczba_pieter	SmallInt	obowiązkowy	Liczba pięter w bloku.
Liczba_lokali	Integer	obowiązkowy	Liczba lokali w bloku.
Liczba_wind	SmallInt	obowiązkowy	Liczba wind w bloku.
Liczba_klatek	SmallInt	nieobowiązkowy	Liczba klatek w bloku.
Rok_powstania	SmallInt	obowiązkowy	Rok powstania bloku.

Mieszkanie

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Mieszkanie_ID	Integer	obowiązkowy	Unikatowy identyfikator mieszkania.
Numer_lokalu	VarChar(4)	obowiązkowy	Numer mieszkania w bloku.

Pietra	SmallInt	obowiązkowy	Piętro, na którym znajduje się mieszkanie.
Powierzchnia	Decimal(6,2)	obowiązkowy	Powierzchnia mieszkania (m2).
Wlasciciel	VarChar(400)	obowiązkowy	Właściciel(e) mieszkania, pole wielowartościowe, złożone (imię, nazwisko).
Mieszkaniec	VarChar(400)	nieobowiązkowy	Mieszkańcy lokalu. Pole wielowartościowe, złożone.
Oplaty	Money	nieobowiązkowy	Wysokość i rodzaj opłat uiszczanych za użytkowanie lokalu. Pole złożone (data, rodzaj opłaty, wysokość), wielowartościowe.
Liczba_pomieszczen	SmallInt	obowiązkowy	Liczba pomieszczeń w lokalu.
Liczba_lazienek	SmallInt	obowiązkowy	Liczba łazienek w mieszkaniu.
Liczba_balkonow	SmallInt	obowiązkowy	Liczba balkonów w mieszkaniu.
Opis	VarChar(400)	nieobowiązkowy	Opis mieszkania.

Firma_uslugowa

Atrybut	Typ i dziedzina	Obowiązkowość	Opis
Firma_uslugowa_ID	Integer	obowiązkowy	Unikatowy identyfikator firmy usługowej.
Nazwa	VarChar(30)	obowiązkowy	Nazwa firmy usługowej.
NIP	Character(10)	obowiązkowy	Numer NIP firmy usługowej.
Adres	VarChar(300)	obowiązkowy	Adres siedziby firmy usługowej, pole złożone (ulica, numer domu, numer lokalu, kod pocztowy, miasto).
Numer_telefonu	VarChar(12)	obowiązkowy	Numer telefonu firmy usługowej.
Email	VarChar(40)	obowiązkowy	Adres email firmy usługowej.
Strona_internetowa	VarChar(40)	nieobowiązkowy	Strona internetowa firmy usługowej.
Rodzaj_uslugi	VarChar(40)	obowiązkowy	Rodzaj pobieranej usługi.
Cena_uslugi	Money	obowiązkowy	Kwota, jaką spółdzielnia mieszkaniowa płaci za wykonanie usługi.
Data_zawarcia_umowy	Date	obowiązkowy	Data zawarcia umowy z podmiotem wykonującym daną usługę.
Data_rozwiazania_umowy	Date	nieobowiązkowy	Data rozwiązania umowy z podmiotem wykonującym daną usługę.

3.4 Dodatkowe reguły integralnościowe (reguły biznesowe)

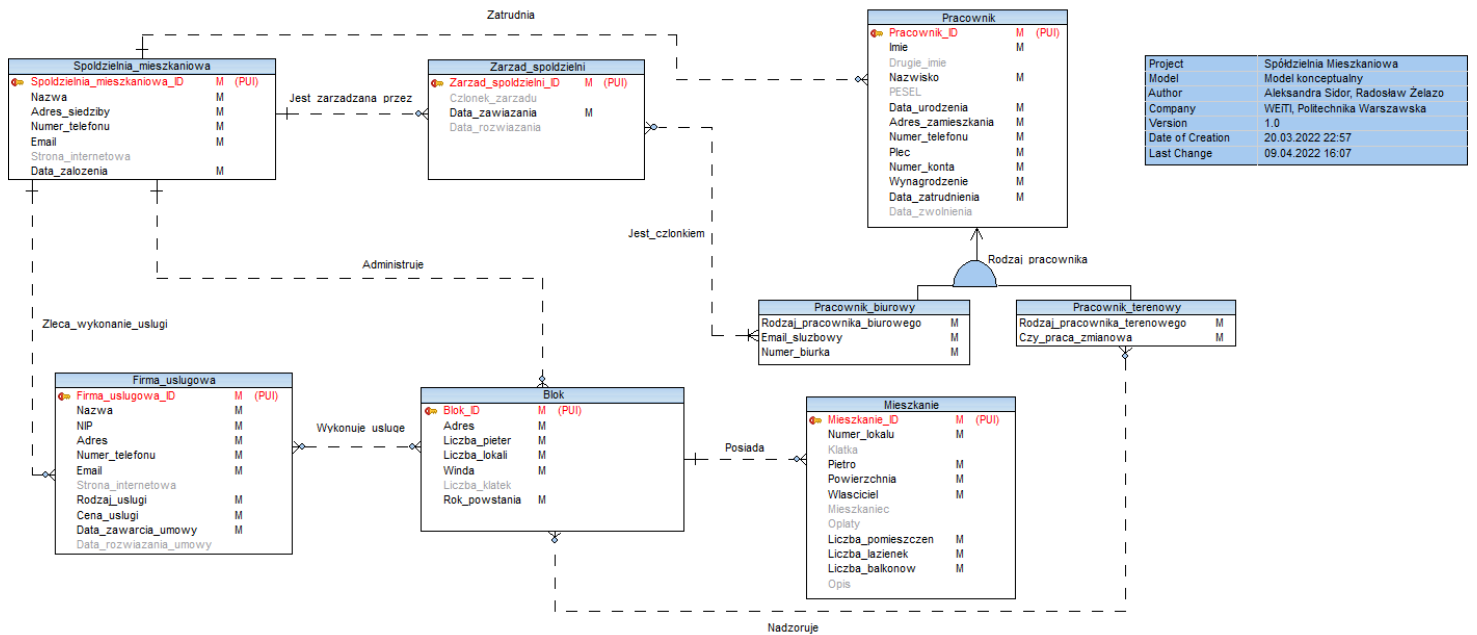
- W trakcie całego okresu pracy w spółdzielni mieszkaniowej, pracownik biurowy może kilkakrotnie zasiadać w zarządzie spółdzielni, jednak funkcja członka zarządu nie jest obligatoryjna dla pracownika biurowego.
- W przeciągu danego okresu, spółdzielnia mieszkaniowa może być zarządzana przez wiele zarządów (ale nie jednocześnie), tzn. w momencie, gdy zmieni się przynajmniej jeden członek, zarząd traktowany jest jako nowy i przypisywany jest mu nowy unikatowy klucz ID.

3.5 Klucze kandydujące i główne (decyzje projektowe)

<i>Nazwa encji</i>	<i>Klucz główny</i>	<i>Klucz kandydujący</i>
Spoldzielnia_mieszkaniowa	Spoldzielnia_mieszkaniowa_ID	Nazwa, Numer_telefonu, Email
Zarząd_spoldzielni	Zarząd_spoldzielni_ID	(Data_zawiazania, Data_rozwiazania)
Pracownik	Pracownik_ID	(Imie, Nazwisko, Data_urodzenia), Numer_telefonu
Pracownik_biurowy	Pracownik_ID	Email_sluzbowy
Pracownik_terenowy	Pracownik_ID	-
Blok	Blok_ID	Adres
Mieszkanie	Mieszkanie_ID	-
Firma_uslugowa	Firma_uslugowa_ID	Nazwa, NIP, Numer_telefonu, Email

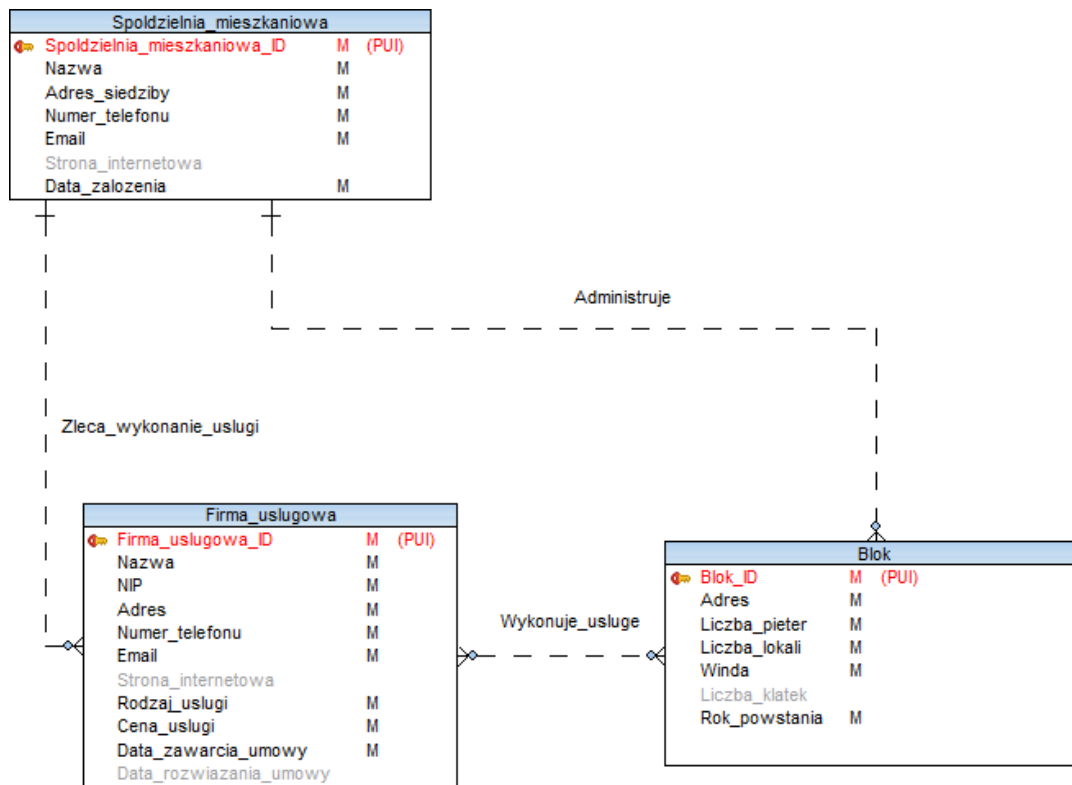
Atrybuty umieszczone w () tworzą klucz złożony.

3.6 Schemat ER na poziomie konceptualnym



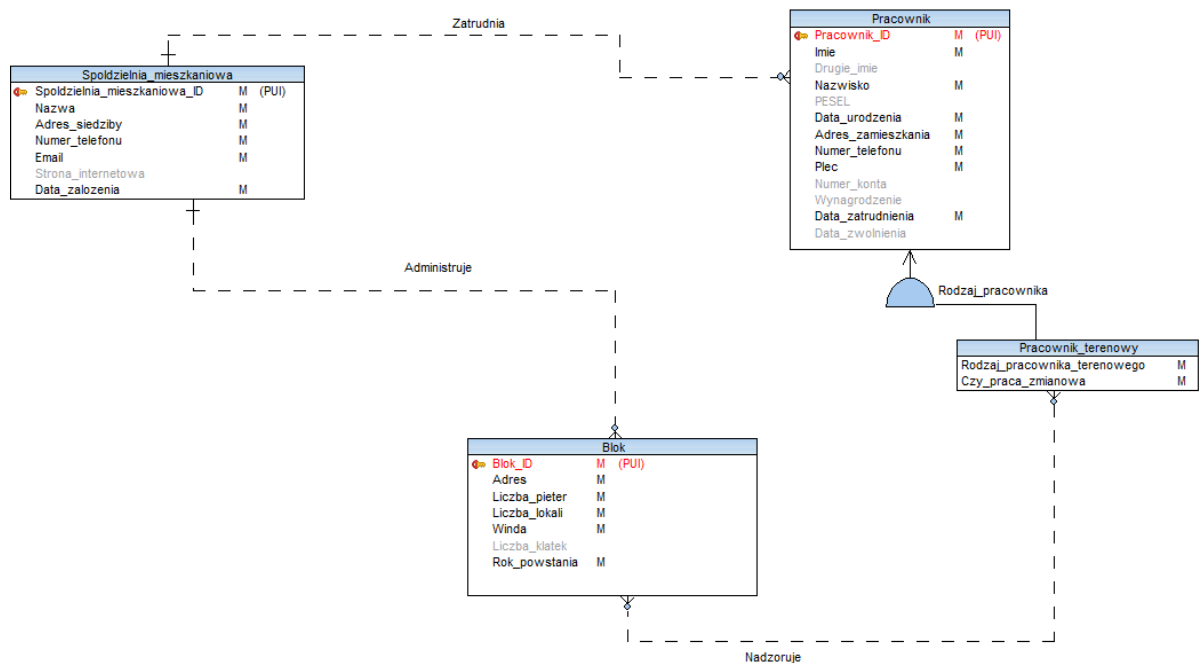
3.7 Problem pułapek szczelinowych i wachlarzowych - analiza i przykłady

Pułapki wachlarzowe



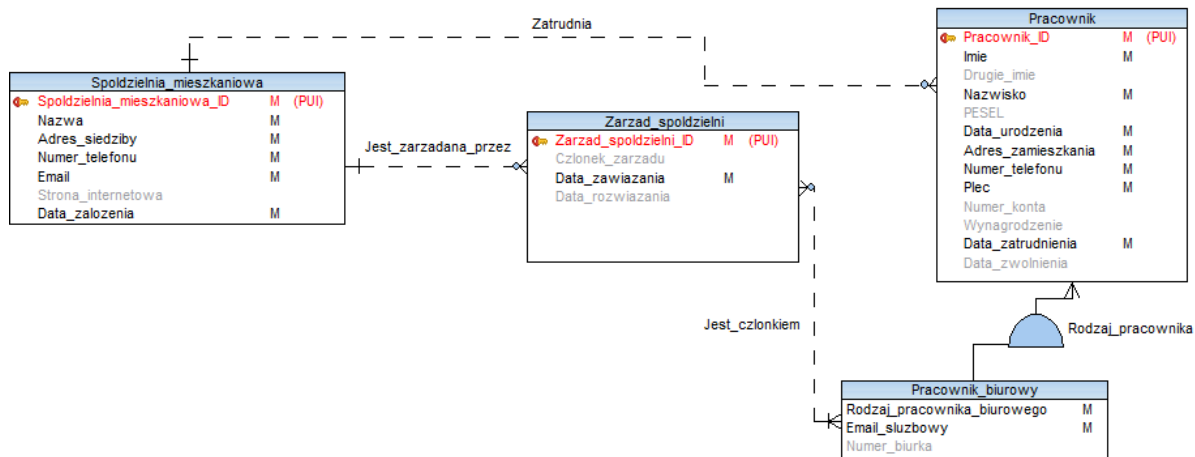
Spółdzielnia mieszkaniowa zatrudnia zewnętrzne firmy usługowe, którym zleca wykonywanie różnych usług w blokach. Gdyby nie było połączenia *Wykonuje_usluge* między encjami *Firma_uslugowa* a *Blok*, trudno byłoby zidentyfikować która firma wykonała/wykonuje daną usługę w którym bloku. Zatem wystąpiłby tu przypadek pułapki wachlarzowej.

Kolejny potencjalny przykład pułapki wachlarzowej w projekcie przedstawia poniższy schemat:



Pracownik terenowy jest rodzajem pracownika, którego praca polega na nadzorowaniu i opiece nad blokami. Gdyby nie było połączenia *Nadzoruje* między encjami *Pracownik_terenowy* a *Blok*, nie byłyby jednoznaczne, konkretnie którzy pracownicy pracują w konkretnych blokach.

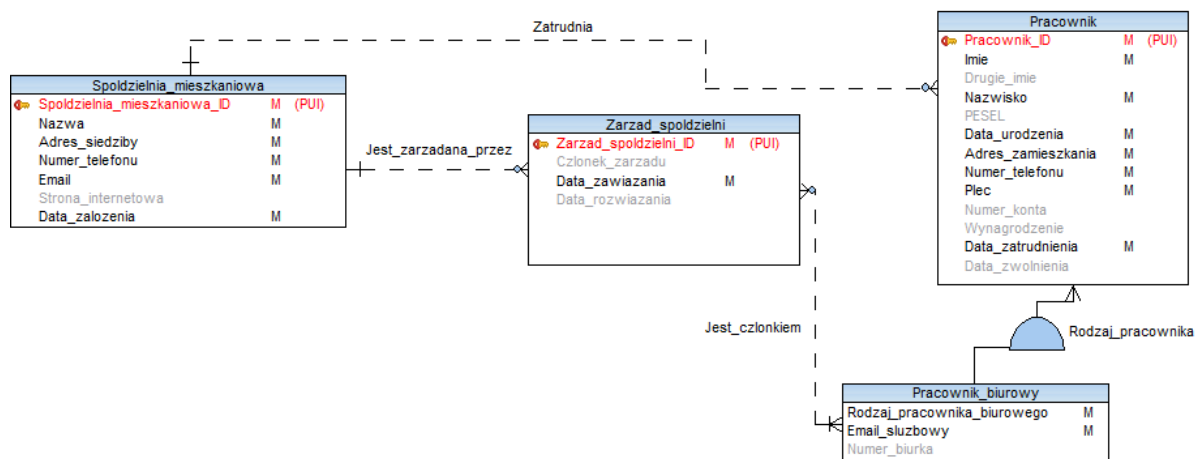
Dokładnie tak samo jest w przypadku pracowników biurowych zasiadających w zarządzie spółdzielni mieszkaniowej.



Istotne było stworzenie związku *Jest_członkiem* pomiędzy encjami *Zarzad_spoldzielni* a *Pracownik_biurowy*, ponieważ nie można by było określić, którzy pracownicy są/byli członkami zarządu spółdzielni, a wymagane jest, aby w zarządzie zasiadali pracownicy zatrudnieni w spółdzielni mieszkaniowej.

Pułapki szczelinowe

Na poniższych schematach można również wyszczególnić potencjalne pułapki szczelinowe.



Gdyby nie było związku *Zatrudnia* między encjami *Pracownik* a *Spoldzielnia_mieszkaniowa*, oznaczałoby to, że jedynymi pracownikami biurowymi w spółdzielni mieszkaniowej byłiby członkowie zarządu. Nie byłoby to prawdą, ponieważ, np. sekretarka lub administrator nie mają obowiązku być członkami zarządu.

4. Model logiczny

4.1 Charakterystyka modelu relacyjnego

Kolejnym etapem w projektowaniu bazy danych, jest etap tworzenia modelu logicznego. Przed przystąpieniem do jego tworzenia, istotne jest zweryfikowanie poprawności modelu konceptualnego oraz usunięcie wszelkich pułapek.

Konwersja projektu do poziomu relacyjnego została wygenerowana przy pomocy programu Toad Data Modeler 7.2 a jej skutkiem było:

- zastąpienie każdej relacji N:M (wielu do wielu) tabelą łączącą dwie encje, przy czym relacje obu encji z tabelą łączącą są typu 1:N (jeden do wielu), a tabela łącząca jest encją słabą
- zmiana nazw encji z liczby pojedynczej na liczbę mnogą (np. *Blok* → *Bloki*)
- utworzenie kluczy głównych tabel na podstawie identyfikującego atrybutu konkretnej encji oraz ustanowienie pozostałych atrybutów w tabelach jako niegłówne
- zamiana specjalizacji (encji uszczegóławiające) na odrębne tabele połączone z encją nadrzędną związkiem identyfikującym z obowiązkowym istnieniem encji nadrzędnej oraz przypisanie encjom podrzędnym klucza głównego encji nadrzędnej
- uzupełnienie encji o klucz obcy przy związkach 1:N (jeden do wielu)
- konwersja typu danych do kompatybilnych z silnikiem bazy danych
- usunięcie z encji wszystkich pól wielowartościowych oraz złożonych i przekształcenie ich w osobne encje związane z encją, której były atrybutami w modelu konceptualnym

4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady

Usunięcie związków N:M (wielu do wielu)

Związki wielu do wielu zostały usunięte, a w ich miejscach powstały tablice łączące:

Tabela *Czlonkowie_zarządu* łączy encje *Pracownicy_biurowi* oraz *Zarzady_spoldzielni* i przechowuje informacje o tym, który pracownik biurowy zasiadał w danym zarządzie.



Czlonkowie_zarządu		
	Zarząd_spoldzielni_ID	Integer
	Pracownik_ID	Integer

Tabela *Nadzory_blokow* pomiędzy encjami *Pracownicy_terenowi* a *Bloki* ma na celu przechowywać dane na temat pracowników terenowych oraz bloków przez nich nadzorowanych.

Nadzory_bloków			
🔑	Blok_ID	Integer NN	(PFK)
🔑	Pracownik_ID	Integer NN	(PFK)

Tabela *Usługi* łączy encje *Firmy_usługowe* oraz *Bloki*. Jej celem jest przechowywanie dane na temat usług pełnionych przez firmy usługowe w blokach. Zawiera również szczegółowe informacje dotyczące danej usługi, tzn. cenę, datę zawarcia umowy na jej wykonywanie itp.

Usługi			
	Cena_uslugi	Number(7,2)	NN
	Data_zawarcia_umowy	Integer	NN
	Data_rozwiazania_umowy	Integer	NN
🔑	Firma_uslugowa_ID	Integer	NN (PFK)
🔑	Blok_ID	Integer	NN (PFK)

Usunięcie pól złożonych oraz wielowartościowych

Tabela *Adresy* powstała z atrybutu o tej samej nazwie, który występował w encjach *Spoldzielnia_mieszkaniowa*, *Pracownik* oraz *Blok*.

Adresy			
🔑	Adres_ID	Integer	NN (PK)
	Ulica	Varchar2(56)	NN
	Numer_domu	Varchar2(5)	NN
	Numer_lokalu	Varchar2(5)	NN
	Miasto	Varchar2(30)	NN
	Kod_pocztowy	Varchar2(6)	NN



Tabela *Wynagrodzenia*, będąca w modelu konceptualnym atrybutem encji *Pracownik*, zawiera informacje na temat wysokości wynagrodzenia, możliwych dodatków do wynagrodzenia oraz dokładne daty związane z jego wypłatą.

Wynagrodzenia			
🔑	Wynagrodzenie_ID	Integer	NN (PK)
	Data_wypłaty	Date	NN
	Za_miesiac	Varchar2(2)	NN
	Wysokosc_wynagrodzenia	Number(7,2)	NN
	Dodatek_do_wynagrodzenia	Number(7,2)	NN
🔑	Pracownik ID	Integer	NN (FK)
	Ma_wynagrodzenie (IX1)		

Tabela *Oplaty*
atrybutu encji

powstała z
Mieszkanie.

Przechowuje dokładne informacje dotyczące opłat związanych z użytkowaniem danego mieszkania.

Oplaty			
 Oplaty_ID	Integer	NN (PK)	
Data_wystawienia_rachunku	Date	NN	
Nr_rachunku	Integer	NN	(AK1)
Za_okres_od	Date	NN	
Za_okres_do	Date	NN	
Czy_oplacony	Czy_oplacony_D	NN	
 Mieszkanie_ID	Integer	NN (FK)	(IX1)
IX_Ma_oplaty (IX1)			

4.3 Proces normalizacji - analiza i przykłady

Proces normalizacji ma na celu eliminację powtarzających się danych poprzez bezstratną modyfikację struktury bazy. Zmniejsza to ryzyko powstawania anomalii oraz zapewnia większe bezpieczeństwo danych. Ponadto, pozwala to na trzymanie danych w jednym miejscu.

Normalizacji dokonuje się poprzez dostosowanie bazy danych do kryteriów zwanych postaciami normalnymi. Dobrze zaprojektowana baza powinna spełniać co najmniej trzecią postać normalną.

Pierwsza postać normalna (1PN) zakłada, że każda wartość atrybutu w każdej krotce jest wartością atomową (elementarną) oraz nie występują powtarzające się grupy - nie występują dwa jednakowe wiersze.

- W celu osiągnięcia pierwszej postaci normalnej w niniejszym projekcie, należało rozważyć przypadek atrybutu *Adres*, który w modelu koncepcyjnym występował w wielu encjach (*Spółdzielnia_mieszkaniowa*, *Pracownik*, *Blok*, *Firma_uslugowa*). Atrybut był również polem segmentowym, ponieważ zawierał więcej niż jeden typ wartości: ulica, nr domu, nr mieszkania, kod pocztowy, miasto. Z tego powodu, w modelu koncepcyjnym powstała nowa relacja *Adresy*, związana z podanymi wyżej encjami, z których usunięty został atrybut *Adres*.
- Podobna sytuacja występowała w przypadku wyodrębnienia rodzaju zużywanych mediów do encji *Media*, która związana jest z relacją *Zuzycia*. Spowodowane jest to faktem, że mieszkanie może zużywać wiele różnych mediów, które mogą być rozliczane za różne okresy. Z tego powodu, uznaliśmy, że najlepiej będzie stworzenie nowej tabeli przechowującej informacje o rodzaju medium oraz o cenie za jednostkę podstawową.

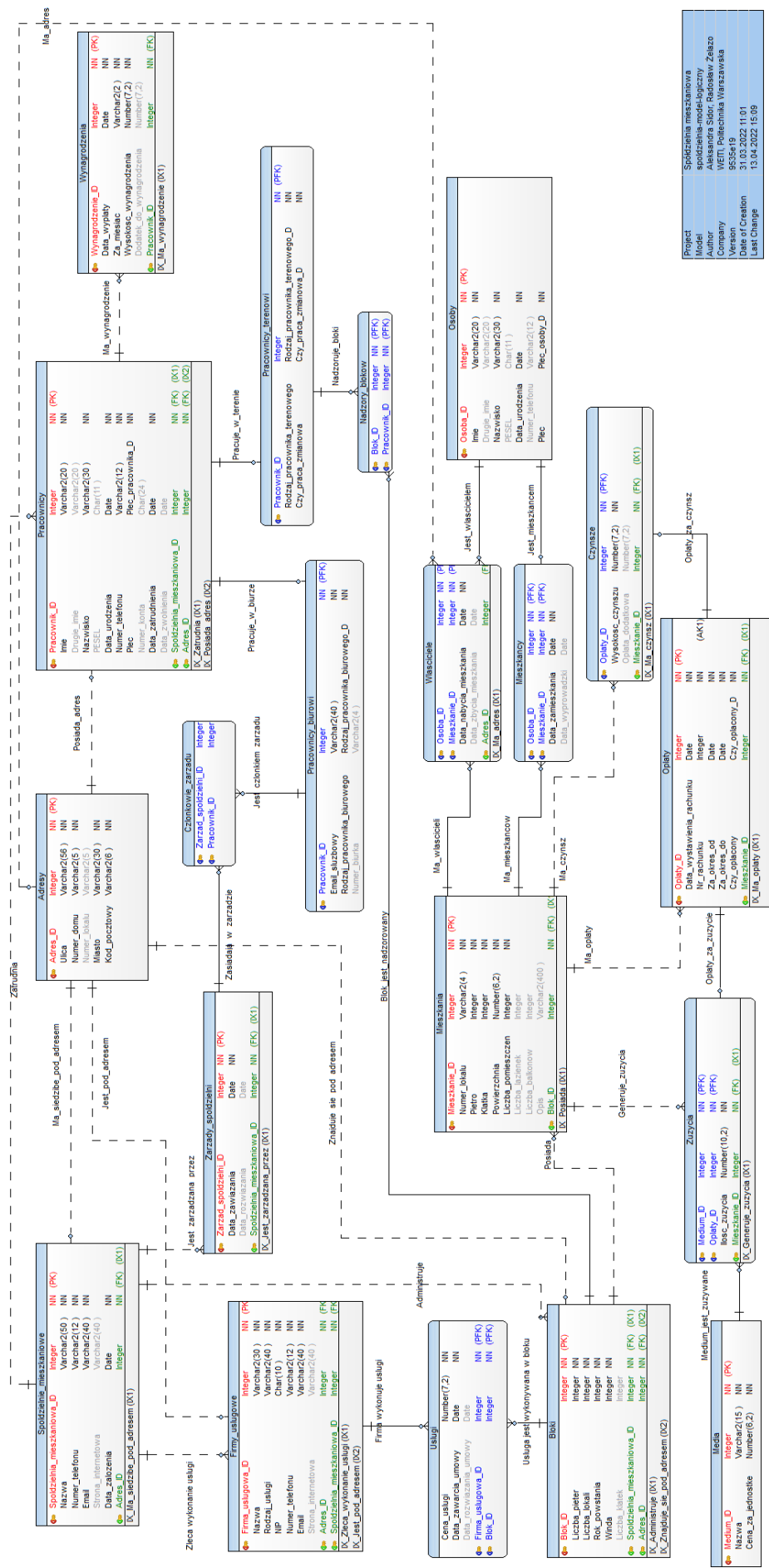
Druga postać normalna (2PN) występuje wtedy, gdy spełnione są kryteria 1PN oraz gdy wszystkie atrybuty niekluczowe są w pełni zależne od kluczy kandydujących, nie tylko od ich części.

- Aby w projekcie spełnione była druga postać normalna, należało zastanowić się nad przypadkiem encji *Zuzycia*. Ilosc_zuzycia zależna jest zarówno od rodzaju mediów, jak i od mieszkania, w którym dane medium jest zużywane. W tym przypadku, wspomniany atrybut encji jest całkowicie zależny od klucza złożonego, zatem spełnione jest kryterium 2PN.

Trzecia postać normalna (3PN) występuje, gdy tabela jest w 2PN oraz zakłada, że żaden atrybut nie będący kluczem nie może zależeć przechodnio od żadnego z kluczy kandydujących - wszystkie niekluczowe kolumny są określone kluczem, całym kluczem i tylko kluczem.

W projekcie, trzecią postać normalną udało się spełnić automatycznie, ponieważ wszystkie elementy niekluczowe zależą bezpośrednio od wszystkich kluczy kandydujących. Prawdopodobnie wynika to z dokładnej analizy struktury bazy danych oraz rozłożeniu modelu na części pierwsze, najbardziej jak to było możliwe.

4.4 Schemat ER na poziomie modelu logicznego



Project	Spółdzielnia mieszkaniowa
Model	spoldzielnia-model-logiczny
Author	Aleksandra Sidor, Radosław Żelazo
Company	WEIT, Politechnika Warszawska
Version	9535e19
Date of Creation	31.03.2022 11:01
Last Change	13.04.2022 15:09

4.5 Więzy integralności

W celu zapewnienia spójności i prawdziwości danych fizycznych wprowadzanych do bazy danych, istotne jest dokładne zdefiniowanie więzów integralności, już na poziomie projektowania bazy. Więzy integralności są to reguły określające sposób osiągnięcia wspomnianej wcześniej spójności danych oraz poprawności związków między rekordami. Uniemożliwiają również usunięcie lub zmianę powiązanych ze sobą danych, tzn. nie można usunąć wiersza, do którego odwołuje się klucz obcy.

W tym celu, do każdej encji przypisany jest unikatowy klucz (główny lub obcy), który jednoznacznie identyfikuje każdy rekord w tabeli. Ponadto, każdy atrybut w encji jest polem atomowym oraz w większości przypadków na atrybuty zostało nałożone ograniczenie NOT NULL.

Oprócz powyższych działań, zadaliśmy również o to, by struktura każdego pola była poprawna oraz typ danych był zgodny z zawartością przechowywanego atrybutu, np. w atrybucie *Data_nabycia_mieszkania* mogą być przechowywane tylko i wyłącznie dane typu Date, a atrybut *PESEL* może przechowywać wartość typu Char o konkretnie zdefiniowanej długości - 11 znaków oraz składać się musi wyłącznie z cyfr. Co więcej, w obrębie całego modelu, pola tego samego typu są identycznie zdefiniowane.

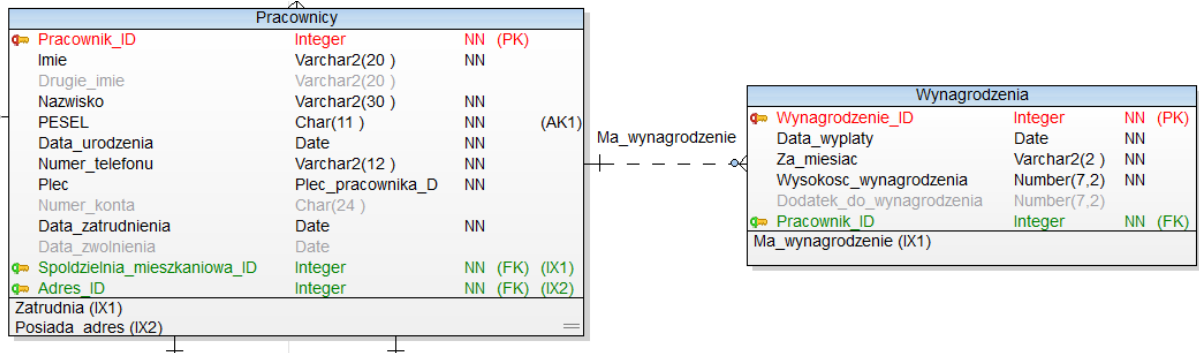
W celu zachowania prawdziwości informacji, na dane przechowujące daty zostało nałożone ograniczenie związane ze spójnością z czasem i chronologią. Oznacza to, że data zwolnienia pracownika, nie może być starsza niż data jego zatrudnienia. Dotyczy to również każdego takiego przypadku w pozostałych encjach.

4.6 Proces denormalizacji - analiza i przykłady

Doprowadzenie bazy danych do postaci wysoko znormalizowanej może spowodować spowolnienie odczytu w dużych bazach danych. Z tego powodu, często należy rozważyć denormalizację struktury bazy, czyli proces odwrotny do normalizacji. Ma to na celu przyspieszenie wykonywania, uniknięcie kosztownych operacji połączeń tabel oraz zwiększenie wydajności systemu.

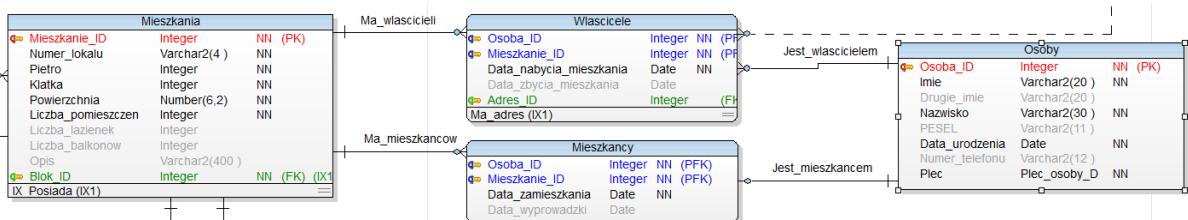
Niniejszy projekt bazy danych spółdzielni mieszkaniowej nie należy do skomplikowanych modeli, z tego względu nie zdecydowaliśmy się na proces denormalizacji. Niemniej jednak, występują w nim takie powiązania, które mogłyby ulec denormalizacji.

Przykładem takiego przypadku jest poniższy związek:



Wynagrodzenie zostało wyodrębnione jako osobna encja, ponieważ w naszym projekcie zawiera ona różne informacje dotyczące wynagrodzenia - składa się z różnych typów danych. Z tego powodu, zdecydowaliśmy się na wyszczególnienie tej encji i zawarcie w niej istotnych atrybutów. Jednak błędem nie byłoby usunięcie encji *Wynagrodzenia* i utworzenie w relacji *Pracownicy* atrybutu *Wynagrodzenie*, zawierającego jedynie kwotę.

Kolejnym przykładem, możliwego procesu denormalizacji jest:



W procesie denormalizacji, encja *Mieszkania* mogłaby zostać połączona bezpośrednio z encją *Osoby*, która reprezentowałaby wszystkie osoby związane z mieszkaniem, tj. mieszkańców i właścicieli. Identyfikacja rodzaju użytkownika mieszkania przez daną osobę, mogłaby zostać przeprowadzona poprzez stworzenie dwóch atrybutów: *Czy_wlasciciel*, *Czy_mieszkaniec*.

Jednak istotne dla nas było zaznaczenie, że jedna osoba może posiadać wiele mieszkań, co przechowywane będzie w relacji *Wlasciele*. Działa to również w drugą stronę - mieszkanie może mieć wielu właścicieli.

5. Faza fizyczna

5.1 Projekt transakcji i weryfikacja ich wykonywalności

Transakcja	Wykonalna?	Potrzebne dane
<ul style="list-style-type: none"> podgląd informacji o spółdzielni mieszkaniowej 	TAK	Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o spółdzielni mieszkaniowej 	TAK	Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> podgląd informacji o pracownikach 	TAK	Pracownicy, Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o pracownikach 	TAK	Pracownicy, Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> podgląd informacji o pracownikach jako pracownikach biurowych 	TAK	Pracownicy_biurowi, Pracownicy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach biurowych 	TAK	Pracownicy_biurowi, Pracownicy
<ul style="list-style-type: none"> podgląd informacji o pracownikach jako pracownikach terenowych 	TAK	Pracownicy_terenowi, Pracownicy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach terenowych 	TAK	Pracownicy_terenowi, Pracownicy
<ul style="list-style-type: none"> podgląd informacji o blokach mieszkalnych 	TAK	Bloki, Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o blokach mieszkalnych 	TAK	Bloki, Spoldzielnie_mieszkaniowe, Adresy
<ul style="list-style-type: none"> podgląd informacji o mieszkaniach 	TAK	Mieszkania, Bloki
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o mieszkaniach 	TAK	Mieszkania, Bloki
<ul style="list-style-type: none"> podgląd informacji o adresach 	TAK	Adresy
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o adresach 	TAK	Adresy
<ul style="list-style-type: none"> podgląd informacji o zarządach spółdzielni 	TAK	Zarzady_spoldzielni, Spoldzielnie_mieszkaniowe
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o zarządach spółdzielni 	TAK	Zarzady_spoldzielni, Spoldzielnie_mieszkaniowe
<ul style="list-style-type: none"> podgląd informacji o pracownikach jako pracownikach biurowych wchodzących w skład zarządu spółdzielni 	TAK	Pracownicy, Czlonkowie_zarzadu, Zarzady_spoldzielni
<ul style="list-style-type: none"> dodawanie/modyfikowanie/usuwanie informacji o pracownikach jako pracownikach biurowych wchodzących w skład zarządu spółdzielni 	TAK	Pracownicy, Czlonkowie_zarzadu, Zarzady_spoldzielni

• podgląd informacji o firmach usługowych	TAK	Firmy_uslugowe, Spoldzielnie_mieszkaniowe
• dodawanie/modyfikowanie/usuwanie informacji o firmach usługowych	TAK	Firmy_uslugowe, Spoldzielnie_mieszkaniowe
• podgląd informacji o firmach usługowych, których usługi wykonywane są w konkretnym bloku mieszkalnym	TAK	Firmy_uslugowe, Uslugi, Bloki
• dodawanie/modyfikowanie/usuwanie informacji o firmach usługowych, których usługi wykonywane są w konkretnym bloku mieszkalnym	TAK	Firmy_uslugowe, Uslugi, Bloki
• podgląd informacji o blokach mieszkalnych, w których pełnią dozór pracownicy terenowi	TAK	Pracownicy, Nadzory_blokow Bloki
• dodawanie/modyfikowanie/usuwanie informacji o blokach mieszkalnych, w których pełnią dozór pracownicy terenowi	TAK	Pracownicy, Nadzory_blokow, Bloki
• podgląd informacji o wynagrodzeniach pracowników	TAK	Wynagrodzenia, Pracownicy
• dodawanie/modyfikowanie/usuwanie informacji o wynagrodzeniach pracowników	TAK	Wynagrodzenia, Pracownicy
• podgląd informacji o osobach	TAK	Osoby
• dodawanie/modyfikowanie/usuwanie informacji o osobach	TAK	Osoby
• podgląd informacji o osobach zamieszkujących mieszkanie	TAK	Mieszkania, Mieszkancy, Osoby
• dodawanie/modyfikowanie/usuwanie informacji o osobach zamieszkujących mieszkanie	TAK	Mieszkania, Mieszkancy, Osoby
• podgląd informacji o osobach posiadających mieszkanie	TAK	Mieszkania, Wlasciele, Osoby, Adresy
• dodawanie/modyfikowanie/usuwanie informacji o osobach posiadających mieszkanie	TAK	Mieszkania, Wlasciele, Osoby, Adresy
• podgląd informacji o mediach zużywanych w mieszkaniu	TAK	Media, Zuzycia, Mieszkania
• dodawanie/modyfikowanie/usuwanie o mediach zużywanych w mieszkaniu	TAK	Media, Zuzycia, Mieszkania
• podgląd informacji o opłatach związanych z mieszkaniem	TAK	Oplaty, Mieszkania

• dodawanie/modyfikowanie/usuwanie o opłatach związanych z mieszkaniem	TAK	Oplaty, Mieszkania
• podgląd informacji o czynszach mieszkań	TAK	Czynsze, Mieszkania
• dodawanie/modyfikowanie/usuwanie o czynszach mieszkań	TAK	Czynsze, Mieszkania
• podgląd informacji o opłatach związanych z mieszkaniem jako opłatach za media	TAK	Media, Zuzycia, Oplaty, Mieszkania
• dodawanie/modyfikowanie/usuwanie informacji o opłatach związanych z mieszkaniem jako opłatach za media	TAK	Media, Zuzycia, Oplaty, Mieszkania
• podgląd informacji o opłatach związanych z mieszkaniem jako opłatach za czynsz	TAK	Czynsze, Oplaty, Mieszkania
• dodawanie/modyfikowanie/usuwanie informacji o opłatach związanych z mieszkaniem jako opłatach za czynsz	TAK	Czynsze, Oplaty, Mieszkania

5.2 Strojenie bazy danych - dobór indeksów

Zaprojektowana baza danych strojona jest poprzez wykorzystanie indeksów wygenerowanych automatycznie na podstawie utworzonych związków między tabelami.

- *IX_Ma_siedzibe_pod_adresem* - wyszukiwanie adresu siedziby spółdzielni

```
CREATE INDEX IX_Ma_siedzibe_pod_adresem ON Spoldzielnie_mieszkaniowe
(Adres_ID)
```

- *IX_Jest_zarządzana_przez* - wyszukiwanie zarządu spółdzielni

```
CREATE INDEX IX_Jest_zarządzana_przez ON Zarzady_spoldzielni
(Spoldzielnia_mieszkaniowa_ID)
```

- *IX_Administruje* - wyszukiwanie bloków administrowanych przez spółdzielnię

```
CREATE INDEX IX_Administruje ON Bloki (Spoldzielnia_mieszkaniowa_ID)
```

- *IX_Znajduje_sie_pod_adresem* - wyszukiwanie adresów bloków administrowanych przez spółdzielnię

```
CREATE INDEX IX_Znajduje_sie_pod_adresem ON Bloki (Adres_ID)
```

- *IX_Zatrudnia* - wyszukiwanie pracowników zatrudnianych przez spółdzielnię

```
CREATE INDEX IX_Zatrudnia ON Pracownicy (Spoldzielnia_mieszkaniowa_ID)
```

- *IX_Posiada_adres* - wyszukiwanie adresów pracowników

```
CREATE INDEX IX_Posiada_adres ON Pracownicy (Adres_ID)
```

- *IX_Zleca_wykonanie_uslugi* - wyszukiwanie firm usługowych świadczących usługi na rzecz spółdzielni

```
CREATE INDEX IX_Zleca_wykonanie_uslugi ON Firmy_uslugowe  
(Spoldzielnia_mieszkaniowa_ID)
```

- *IX_Jest_pod_adresem* - wyszukiwanie adresu firmy usługowej

```
CREATE INDEX IX_Jest_pod_adresem ON Firmy_uslugowe (Adres_ID)
```

- *IX_Ma_czynsz* - wyszukiwanie czynszów za mieszkanie

```
CREATE INDEX IX_Ma_czynsz ON Czynsze (Mieszkanie_ID)
```

- *IX_Ma_adres* - wyszukiwanie adresu właściciela mieszkania (jeśli nie jest on mieszkańcem)

```
CREATE INDEX IX_Ma_adres ON Wlasciciele (Adres_ID)
```

- *IX_Generuje_zuzycia* - wyszukiwanie zużyć mediów w mieszkaniu

```
CREATE INDEX IX_Generuje_zuzycia ON Zuzycia (Mieszkanie_ID)
```

- *IX_Ma_wynagrodzenie* - wyszukiwanie wynagrodzeń pracownika

```
CREATE INDEX IX_Ma_wynagrodzenie ON Wynagrodzenia (Pracownik_ID)
```

5.3 Skrypt SQL zakładający bazę danych

```
-- Create sequences section -----  
  
CREATE SEQUENCE Firmy_uslugowe_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Adresy_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Zarzady_spoldzielni_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Pracownicy_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Wynagrodzenia_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Bloki_Seq  
  INCREMENT BY 1  
  START WITH 1  
  NOMAXVALUE  
  NOMINVALUE  
  CACHE 20  
/  
  
CREATE SEQUENCE Mieszkania_Seq  
  INCREMENT BY 1
```

```

START WITH 1
NOMAXVALUE
NOMINVALUE
CACHE 20
/

CREATE SEQUENCE Osoby_Seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
CACHE 20
/

CREATE SEQUENCE Oplaty_Seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
CACHE 20
/

CREATE SEQUENCE Media_Seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
CACHE 20
/

CREATE SEQUENCE Spoldzielnie_mieszkaniowe_Seq
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOMINVALUE
CACHE 20
/

-- Create tables section -----

-- Table Spoldzielnie_mieszkaniowe

CREATE TABLE Spoldzielnie_mieszkaniowe(
    Spoldzielnia_mieszkaniowa_ID Integer NOT NULL,
    Nazwa Varchar2(50 ) NOT NULL,
    Numer_telefonu Varchar2(12 ) NOT NULL,
    Email Varchar2(40 ) NOT NULL,
    Strona_internetowa Varchar2(40 ),
    Data_zalozenia Date NOT NULL,
    Adres_ID Integer NOT NULL
)
/

-- Create indexes for table Spoldzielnie_mieszkaniowe

CREATE INDEX IX_Ma_siedziba_pod_adresem ON Spoldzielnie_mieszkaniowe (Adres_ID)
/

```

```

-- Add keys for table Spoldzielnie_mieszkaniowe

ALTER TABLE Spoldzielnie_mieszkaniowe ADD CONSTRAINT Spoldzielnia_mieszkaniowa_PK PRIMARY
KEY (Spoldzielnia_mieszkaniowa_ID)
/

-- Table Zarzady_spoldzielni

CREATE TABLE Zarzady_spoldzielni(
    Zarzad_spoldzielni_ID Integer NOT NULL,
    Data_zawiazania Date NOT NULL,
    Data_rozwiazania Date,
    Spoldzielnia_mieszkaniowa_ID Integer NOT NULL,
    CONSTRAINT Data_rozwiazania_C CHECK (Data_rozwiazania > Data_zawiazania)
)
/

-- Create indexes for table Zarzady_spoldzielni

CREATE INDEX IX_Jest_zarządzana_przez ON Zarzady_spoldzielni (Spoldzielnia_mieszkaniowa_ID)
/

-- Add keys for table Zarzady_spoldzielni

ALTER TABLE Zarzady_spoldzielni ADD CONSTRAINT Zarzad_spoldzielni_PK PRIMARY KEY
(Zarzad_spoldzielni_ID)
/

-- Table Bloki

CREATE TABLE Bloki(
    Blok_ID Integer NOT NULL,
    Liczba_pieter Integer NOT NULL
        CONSTRAINT ValidValuesLiczba_pieter CHECK ((Liczba_pieter >= 1)),
    Liczba_lokali Integer NOT NULL,
    Rok_powstania Integer NOT NULL,
    Winda Integer NOT NULL
        CONSTRAINT Winda_C CHECK ((Winda >= 0)),
    Liczba_klatek Integer,
    Spoldzielnia_mieszkaniowa_ID Integer NOT NULL,
    Adres_ID Integer NOT NULL
)
/

-- Create indexes for table Bloki

CREATE INDEX IX_Administruje ON Bloki (Spoldzielnia_mieszkaniowa_ID)
/

CREATE INDEX IX_Znajduje_sie_pod_adresem ON Bloki (Adres_ID)
/

-- Add keys for table Bloki

ALTER TABLE Bloki ADD CONSTRAINT Blok_PK PRIMARY KEY (Blok_ID)
/

```

-- Table Mieszkania

```
CREATE TABLE Mieszkania(  
    Mieszkanie_ID Integer NOT NULL,  
    Numer_lokalu Varchar2(4 ) NOT NULL,  
    Pietro Integer NOT NULL,  
    Klatka Integer NOT NULL  
        CONSTRAINT Klatka_C CHECK ((Klatka >= 1)),  
    Powierzchnia Number(6,2) NOT NULL  
        CONSTRAINT Powierzchnia_C CHECK ((Powierzchnia > 0)),  
    Liczba_pomieszczen Integer NOT NULL  
        CONSTRAINT Liczba_pomieszczen_C CHECK ((Liczba_pomieszczen > 0)),  
    Liczba_lazienek Integer  
        CONSTRAINT Liczba_lazienek_C CHECK ((Liczba_lazienek >= 0)),  
    Liczba_balkonow Integer  
        CONSTRAINT Liczba_balkonow_C CHECK ((Liczba_balkonow >= 0)),  
    Opis Varchar2(400 ),  
    Blok_ID Integer NOT NULL  
)  
/
```

-- Create indexes for table Mieszkania

```
CREATE INDEX IX_Posiada ON Mieszkania (Blok_ID)  
/
```

-- Add keys for table Mieszkania

```
ALTER TABLE Mieszkania ADD CONSTRAINT Lokal_PK PRIMARY KEY (Mieszkanie_ID)  
/
```

-- Table Pracownicy

```
CREATE TABLE Pracownicy(  
    Pracownik_ID Integer NOT NULL,  
    Imie Varchar2(20 ) NOT NULL,  
    Drugie_imie Varchar2(20 ),  
    Nazwisko Varchar2(30 ) NOT NULL,  
    PESEL Char(11 )  
        CONSTRAINT PESEL_Pracownicy_C CHECK (PESEL NOT LIKE '%[^0-9]%'),  
    Data_urodzenia Date NOT NULL,  
    Numer_telefonu Varchar2(12 ) NOT NULL,  
    Plec Char(1 ) NOT NULL  
        CONSTRAINT Plec_pracownika_C CHECK (Plec IN ('K', 'M')),  
    Numer_konta Char(24 ),  
    Data_zatrudnienia Date NOT NULL,  
    Data_zwolnienia Date,  
    Spoldzielnia_mieszkaniowa_ID Integer NOT NULL,  
    Adres_ID Integer NOT NULL,  
    CONSTRAINT Data_zwolnienia_C CHECK ((Data_zwolnienia > Data_zatrudnienia))  
)  
/
```

-- Create indexes for table Pracownicy

```
CREATE INDEX IX_Zatrudnia ON Pracownicy (Spoldzielnia_mieszkaniowa_ID)
```

```

/

CREATE INDEX IX_Poslada_adres ON Pracownicy (Adres_ID)
/

-- Add keys for table Pracownicy

ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_PK PRIMARY KEY (Pracownik_ID)
/

-- Table Pracownicy_biurowi

CREATE TABLE Pracownicy_biurowi(
    Pracownik_ID Integer NOT NULL,
    Email_sluzbowy Varchar2(40 ) NOT NULL,
    Rodzaj_pracownika_biurowego Varchar2(20 ) NOT NULL
        CONSTRAINT Rodzaj_pracownika_biurowego_C CHECK (Rodzaj_pracownika_biurowego IN
('sekretarka', 'ksiegowy', 'dyrektor', 'prezes')),
    Numer_burka Varchar2(4 )
)
/

-- Add keys for table Pracownicy_biurowi

ALTER TABLE Pracownicy_biurowi ADD CONSTRAINT Pracownik_biurowy_PK PRIMARY KEY
(Pracownik_ID)
/

-- Table Pracownicy_terenowi

CREATE TABLE Pracownicy_terenowi(
    Pracownik_ID Integer NOT NULL,
    Rodzaj_pracownika_terenowego Varchar2(20 ) NOT NULL
        CONSTRAINT Rodzaj_pracownika_terenowego_C CHECK (Rodzaj_pracownika_terenowego IN
('dozorca', 'sprzataczka', 'hydraulik', 'elektryk')),
    Czy_praca_zmianowa Char(1 ) NOT NULL
        CONSTRAINT Czy_praca_zmianowa_C CHECK (Czy_praca_zmianowa IN ('T', 'N'))
)
/

-- Add keys for table Pracownicy_terenowi

ALTER TABLE Pracownicy_terenowi ADD CONSTRAINT Pracownik_terenowy_PK PRIMARY KEY
(Pracownik_ID)
/

-- Table Firmy_uslugowe

CREATE TABLE Firmy_uslugowe(
    Firma_uslugowa_ID Integer NOT NULL,
    Nazwa Varchar2(30 ) NOT NULL,
    Rodzaj_uslugi Varchar2(40 ) NOT NULL,
    NIP Char(10 ) NOT NULL
        CONSTRAINT NIP_C CHECK (NIP NOT LIKE '%[^0-9]%'),
    Numer_telefonu Varchar2(12 ) NOT NULL,
    Email Varchar2(40 ) NOT NULL,
    Strona_internetowa Varchar2(40 ),

```



```

    Adres_ID Integer NOT NULL,
    Spoldzielnia_mieszkaniowa_ID Integer NOT NULL
)
/

-- Create indexes for table Firmy_uslugowe

CREATE INDEX IX_Zleca_wykonanie_uslugi ON Firmy_uslugowe (Spoldzielnia_mieszkaniowa_ID)
/

CREATE INDEX IX_Jest_pod_adresem ON Firmy_uslugowe (Adres_ID)
/

-- Add keys for table Firmy_uslugowe

ALTER TABLE Firmy_uslugowe ADD CONSTRAINT Firma_uslugowa_PK PRIMARY KEY (Firma_uslugowa_ID)
/

ALTER TABLE Firmy_uslugowe ADD CONSTRAINT NIP UNIQUE (NIP)
/

-- Table Czlonkowie_zarzadu

CREATE TABLE Czlonkowie_zarzadu(
    Zarzad_spoldzielni_ID Integer NOT NULL,
    Pracownik_ID Integer NOT NULL
)
/

-- Table Nadzory_blokow

CREATE TABLE Nadzory_blokow(
    Blok_ID Integer NOT NULL,
    Pracownik_ID Integer NOT NULL
)
/

-- Table Adresy

CREATE TABLE Adresy(
    Adres_ID Integer NOT NULL,
    Ulica Varchar2(56 ) NOT NULL,
    Numer_domu Varchar2(5 ) NOT NULL,
    Numer_lokalu Varchar2(5 ),
    Miasto Varchar2(30 ) NOT NULL,
    Kod_pocztowy Varchar2(6 ) NOT NULL
)
/

-- Add keys for table Adresy

ALTER TABLE Adresy ADD CONSTRAINT PK_Adresy PRIMARY KEY (Adres_ID)
/

-- Table and Columns comments section

COMMENT ON COLUMN Adresy.Adres_ID IS 'Unikatowy identyfikator adresu.'
```

```

/
COMMENT ON COLUMN Adresy.Ulica IS 'Nazwa ulicy.'
/
COMMENT ON COLUMN Adresy.Numer_domu IS 'Numer domu.'
/
COMMENT ON COLUMN Adresy.Numer_lokalu IS 'Numer lokalu.'
/
COMMENT ON COLUMN Adresy.Miasto IS 'Nazwa miejscowości.'
/
COMMENT ON COLUMN Adresy.Kod_pocztowy IS 'Kod pocztowy.'
/

-- Table Osoby

CREATE TABLE Osoby(
    Osoba_ID Integer NOT NULL,
    Imie Varchar2(20 ) NOT NULL,
    Drugie_imie Varchar2(20 ),
    Nazwisko Varchar2(30 ) NOT NULL,
    PESEL Char(11 )
        CONSTRAINT PESEL_Osoby_C CHECK (PESEL NOT LIKE '%[^0-9]%'),
    Data_urodzenia Date NOT NULL,
    Numer_telefonu Varchar2(12 ),
    Plec Char(1 ) NOT NULL
        CONSTRAINT Osoba_plec_C CHECK (Plec IN ('K', 'M'))
        CONSTRAINT Plec_C CHECK (Plec IN ('K', 'M'))
)
/

-- Add keys for table Osoby

ALTER TABLE Osoby ADD CONSTRAINT PK_Osoby PRIMARY KEY (Osoba_ID)
/

-- Table and Columns comments section

COMMENT ON TABLE Osoby IS 'Encja reprezentująca osobę. Zawiera podstawowe dane personalne konkretnej osoby.'
/

-- Table Usługi

CREATE TABLE Usługi(
    Cena_uslugi Number(7,2) NOT NULL,
    Data_zawarcia_umowy Date NOT NULL,
    Data_rozwiazania_umowy Date,
    Firma_uslugowa_ID Integer NOT NULL,
    Blok_ID Integer NOT NULL,
    CONSTRAINT Data_rozwiazania_umowy_C CHECK ((Data_rozwiazania_umowy >
Data_zawarcia_umowy))
)
/

-- Add keys for table Usługi

ALTER TABLE Usługi ADD CONSTRAINT PK_Usługi PRIMARY KEY (Firma_uslugowa_ID,Blok_ID)
/

```

```

-- Table Media

CREATE TABLE Media(
    Medium_ID Integer NOT NULL,
    Nazwa Varchar2(15 ) NOT NULL,
    Cena_za_jednostke Number(6,2) NOT NULL
)
/

-- Add keys for table Media

ALTER TABLE Media ADD CONSTRAINT PK_Media PRIMARY KEY (Medium_ID)
/

-- Table and Columns comments section

COMMENT ON COLUMN Media.Medium_ID IS 'Unikatowy klucz identyfikujący medium.'
/
COMMENT ON COLUMN Media.Nazwa IS 'Nazwa danego medium (np. woda, gaz, prąd, ogrzewanie).'
/
COMMENT ON COLUMN Media.Cena_za_jednostke IS 'Cena za jednostkę danego medium (np. za litr,
za kilowatogodzinę itp).'
/

-- Table Czysze

CREATE TABLE Czysze(
    Oplaty_ID Integer NOT NULL,
    Wysokosc_czynszu Number(7,2) NOT NULL,
    Oplata_dodatkowa Number(7,2),
    Mieszkanie_ID Integer NOT NULL
)
/

-- Create indexes for table Czysze

CREATE INDEX IX_Ma_czynsz ON Czysze (Mieszkanie_ID)
/

-- Add keys for table Czysze

ALTER TABLE Czysze ADD CONSTRAINT PK_Czynsze PRIMARY KEY (Oplaty_ID)
/

-- Table and Columns comments section

COMMENT ON COLUMN Czysze.Wysokosc_czynszu IS 'Wysokość czynszu.'
/

-- Table Wlasciciele

CREATE TABLE Wlasciciele(
    Osoba_id Integer NOT NULL,
    Mieszkanie_ID Integer NOT NULL,
    Data_nabycia_mieszkania Date NOT NULL,
    Data_zbycia_mieszkania Date,

```

```

    Adres_ID Integer,
    CONSTRAINT Data_zbycia_mieszkania_C CHECK (Data_zbycia_mieszkania >
Data_nabycia_mieszkania)
)
/

-- Create indexes for table Wlasciciele

CREATE INDEX IX_Ma_adres ON Wlasciciele (Adres_ID)
/

-- Add keys for table Wlasciciele

ALTER TABLE Wlasciciele ADD CONSTRAINT PK_Wlasciciele PRIMARY KEY (Osoba_id,Mieszkanie_ID)
/

-- Table Oplaty

CREATE TABLE Oplaty(
    Oplaty_ID Integer NOT NULL,
    Data_wystawienia_rachunku Date NOT NULL,
    Nr_rachunku Integer NOT NULL,
    Za_okres_od Date NOT NULL,
    Za_okres_do Date NOT NULL,
    Czy_oplacony Char(1 ) NOT NULL
        CONSTRAINT Czy_oplacony_C CHECK (Czy_oplacony IN ('T', 'N')),
    Mieszkanie_ID Integer NOT NULL,
    CONSTRAINT Za_okres_do_C CHECK ((Za_okres_do > Za_okres_od))
)
/

-- Create indexes for table Oplaty

CREATE INDEX IX_Ma_oplaty ON Oplaty (Mieszkanie_ID)
/

-- Add keys for table Oplaty

ALTER TABLE Oplaty ADD CONSTRAINT PK_Oplaty PRIMARY KEY (Oplaty_ID)
/

ALTER TABLE Oplaty ADD CONSTRAINT Nr_rachunku UNIQUE (Nr_rachunku)
/

-- Table Mieszkancy

CREATE TABLE Mieszkancy(
    Osoba_id Integer NOT NULL,
    Mieszkanie_ID Integer NOT NULL,
    Data_zamieszkania Date NOT NULL,
    Data_wyprowadzki Date,
    CONSTRAINT Data_wyprowadzki_C CHECK (Data_wyprowadzki > Data_zamieszkania)
)
/

-- Add keys for table Mieszkancy

```

```

ALTER TABLE Mieszkancy ADD CONSTRAINT PK_Mieszkancy PRIMARY KEY (Osoba_id,Mieszkanie_ID)
/

-- Table Zuzycia

CREATE TABLE Zuzycia(
    Medium_ID Integer NOT NULL,
    Oplaty_ID Integer NOT NULL,
    Ilosc_zuzycia Number(10,2) NOT NULL,
    Mieszkanie_ID Integer NOT NULL
)
/

-- Create indexes for table Zuzycia

CREATE INDEX IX_Generuje_zuzycia ON Zuzycia (Mieszkanie_ID)
/

-- Add keys for table Zuzycia

ALTER TABLE Zuzycia ADD CONSTRAINT PK_Zuzycia PRIMARY KEY (Medium_ID,Oplaty_ID)
/

-- Table Wynagrodzenia

CREATE TABLE Wynagrodzenia(
    Wynagrodzenie_ID Integer NOT NULL,
    Data_wyplaty Date NOT NULL,
    Za_miesiac Varchar2(2 ) NOT NULL,
    Wysokosc_wynagrodzenia Number(7,2) NOT NULL,
    Dodatek_do_wynagrodzenia Number(7,2),
    Pracownik_ID Integer NOT NULL
)
/

-- Create indexes for table Wynagrodzenia

CREATE INDEX IX_Ma_wynagrodzenie ON Wynagrodzenia (Pracownik_ID)
/

-- Add keys for table Wynagrodzenia

ALTER TABLE Wynagrodzenia ADD CONSTRAINT PK_Wynagrodzenia PRIMARY KEY (Wynagrodzenie_ID)
/

-- Trigger for sequence Spoldzielnie_mieszkaniowe_Seq for column
Spoldzielnia_mieszkaniowa_ID in table Spoldzielnie_mieszkaniowe -----
CREATE OR REPLACE TRIGGER ts_Spoldzielnie_mieszkaniowe_Spoldzielnie_mieszkaniowe_Seq BEFORE
INSERT
ON Spoldzielnie_mieszkaniowe FOR EACH ROW
BEGIN
    :new.Spoldzielnia_mieszkaniowa_ID := Spoldzielnie_mieszkaniowe_Seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER tsu_Spoldzielnie_mieszkaniowe_Spoldzielnie_mieszkaniowe_Seq AFTER
UPDATE OF Spoldzielnia_mieszkaniowa_ID
ON Spoldzielnie_mieszkaniowe FOR EACH ROW

```

```

BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Spoldzielnia_mieszkaniowa_ID in
table Spoldzielnie_mieszkaniowe as it uses sequence.');
```

END;

/

```

-- Trigger for sequence Zarzady_spoldzielni_Seq for column Zarzad_spoldzielni_ID in table
Zarzady_spoldzielni -----
CREATE OR REPLACE TRIGGER ts_Zarzady_spoldzielni_Zarzady_spoldzielni_Seq BEFORE INSERT
ON Zarzady_spoldzielni FOR EACH ROW
BEGIN
    :new.Zarzad_spoldzielni_ID := Zarzady_spoldzielni_Seq.nextval;
END;
```

/

```

CREATE OR REPLACE TRIGGER tsu_Zarzady_spoldzielni_Zarzady_spoldzielni_Seq AFTER UPDATE OF
Zarzad_spoldzielni_ID
ON Zarzady_spoldzielni FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Zarzad_spoldzielni_ID in table
Zarzady_spoldzielni as it uses sequence.');
```

END;

/

```

-- Trigger for sequence Bloki_Seq for column Blok_ID in table Bloki -----
CREATE OR REPLACE TRIGGER ts_Bloki_Bloki_Seq BEFORE INSERT
ON Bloki FOR EACH ROW
BEGIN
    :new.Blok_ID := Bloki_Seq.nextval;
END;
```

/

```

CREATE OR REPLACE TRIGGER tsu_Bloki_Bloki_Seq AFTER UPDATE OF Blok_ID
ON Bloki FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Blok_ID in table Bloki as it uses
sequence.');
```

END;

/

```

-- Trigger for sequence Mieszkania_Seq for column Mieszkanie_ID in table Mieszkania -----
--
CREATE OR REPLACE TRIGGER ts_Mieszkania_Mieszkania_Seq BEFORE INSERT
ON Mieszkania FOR EACH ROW
BEGIN
    :new.Mieszkanie_ID := Mieszkania_Seq.nextval;
END;
```

/

```

CREATE OR REPLACE TRIGGER tsu_Mieszkania_Mieszkania_Seq AFTER UPDATE OF Mieszkanie_ID
ON Mieszkania FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Mieszkanie_ID in table Mieszkania as
it uses sequence.');
```

END;

/

```

-- Trigger for sequence Pracownicy_Seq for column Pracownik_ID in table Pracownicy -----
-
CREATE OR REPLACE TRIGGER ts_Pracownicy_Pracownicy_Seq BEFORE INSERT
```

```

ON Pracownicy FOR EACH ROW
BEGIN
    :new.Pracownik_ID := Pracownicy_Seq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_Pracownicy_Seq AFTER UPDATE OF Pracownik_ID
ON Pracownicy FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Pracownik_ID in table Pracownicy as
it uses sequence.');
```

```

END;
/

-- Trigger for sequence Firmy_uslugowe_Seq for column Firma_uslugowa_ID in table
Firmy_uslugowe -----
CREATE OR REPLACE TRIGGER ts_Firmy_uslugowe_Firmy_uslugowe_Seq BEFORE INSERT
ON Firmy_uslugowe FOR EACH ROW
BEGIN
    :new.Firma_uslugowa_ID := Firmy_uslugowe_Seq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Firmy_uslugowe_Firmy_uslugowe_Seq AFTER UPDATE OF
Firma_uslugowa_ID
ON Firmy_uslugowe FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Firma_uslugowa_ID in table
Firmy_uslugowe as it uses sequence.');
```

```

END;
/

-- Trigger for sequence Adresy_Seq for column Adres_ID in table Adresy -----
CREATE OR REPLACE TRIGGER ts_Adresy_Adresy_Seq BEFORE INSERT
ON Adresy FOR EACH ROW
BEGIN
    :new.Adres_ID := Adresy_Seq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Adresy_Adresy_Seq AFTER UPDATE OF Adres_ID
ON Adresy FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Adres_ID in table Adresy as it uses
sequence.');
```

```

END;
/

-- Trigger for sequence Osoby_Seq for column Osoba_ID in table Osoby -----
CREATE OR REPLACE TRIGGER ts_Osoby_Osoby_Seq BEFORE INSERT
ON Osoby FOR EACH ROW
BEGIN
    :new.Osoba_ID := Osoby_Seq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Osoby_Osoby_Seq AFTER UPDATE OF Osoba_ID
ON Osoby FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010,'Cannot update column Osoba_ID in table Osoby as it uses
sequence.');
```

```

END;
/

-- Trigger for sequence Media_Seq for column Medium_ID in table Media -----
CREATE OR REPLACE TRIGGER ts_Media_Media_Seq BEFORE INSERT
ON Media FOR EACH ROW
BEGIN
    :new.Medium_ID := Media_Seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER tsu_Media_Media_Seq AFTER UPDATE OF Medium_ID
ON Media FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot update column Medium_ID in table Media as it uses
sequence. ');
END;
/

-- Trigger for sequence Oplaty_Seq for column Oplaty_ID in table Oplaty -----
CREATE OR REPLACE TRIGGER ts_Oplaty_Oplaty_Seq BEFORE INSERT
ON Oplaty FOR EACH ROW
BEGIN
    :new.Oplaty_ID := Oplaty_Seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER tsu_Oplaty_Oplaty_Seq AFTER UPDATE OF Oplaty_ID
ON Oplaty FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot update column Oplaty_ID in table Oplaty as it uses
sequence. ');
END;
/

-- Trigger for sequence Wynagrodzenia_Seq for column Wynagrodzenie_ID in table
Wynagrodzenia -----
CREATE OR REPLACE TRIGGER ts_Wynagrodzenia_Wynagrodzenia_Seq BEFORE INSERT
ON Wynagrodzenia FOR EACH ROW
BEGIN
    :new.Wynagrodzenie_ID := Wynagrodzenia_Seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER tsu_Wynagrodzenia_Wynagrodzenia_Seq AFTER UPDATE OF
Wynagrodzenie_ID
ON Wynagrodzenia FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Cannot update column Wynagrodzenie_ID in table
Wynagrodzenia as it uses sequence. ');
END;
/

-- Create foreign keys (relationships) section -----
-----

ALTER TABLE Zarzady_spoldzielni ADD CONSTRAINT Jest_zarządzana_przez FOREIGN KEY
(Spoldzielnia_mieszkaniowa_ID) REFERENCES Spoldzielnie_mieszkaniowe
(Spoldzielnia_mieszkaniowa_ID)

```


/

```
ALTER TABLE Bloki ADD CONSTRAINT Administruje FOREIGN KEY (Spoldzielnia_mieszkaniowa_ID)
REFERENCES Spoldzielnie_mieszkaniowe (Spoldzielnia_mieszkaniowa_ID)
```

/

```
ALTER TABLE Mieszkania ADD CONSTRAINT Posiada FOREIGN KEY (Blok_ID) REFERENCES Bloki
(Blok_ID)
```

/

```
ALTER TABLE Bloki ADD CONSTRAINT Znajduje_sie_pod_adresem FOREIGN KEY (Adres_ID) REFERENCES
Adresy (Adres_ID)
```

/

```
ALTER TABLE Firmy_uslugowe ADD CONSTRAINT Zleca_wykonanie_uslugi FOREIGN KEY
(Spoldzielnia_mieszkaniowa_ID) REFERENCES Spoldzielnie_mieszkaniowe
(Spoldzielnia_mieszkaniowa_ID)
```

/

```
ALTER TABLE Uslugi ADD CONSTRAINT Firma_wykonuje_uslugi FOREIGN KEY (Firma_uslugowa_ID)
REFERENCES Firmy_uslugowe (Firma_uslugowa_ID)
```

/

```
ALTER TABLE Uslugi ADD CONSTRAINT Usługa_jest_wykonywana_w_bloku FOREIGN KEY (Blok_ID)
REFERENCES Bloki (Blok_ID)
```

/

```
ALTER TABLE Zuzycia ADD CONSTRAINT Medium_jest_zuzywane FOREIGN KEY (Medium_ID) REFERENCES
Media (Medium_ID)
```

/

```
ALTER TABLE Mieszkancy ADD CONSTRAINT Jest_mieszkancem FOREIGN KEY (Osoba_id) REFERENCES
Osoby (Osoba_ID)
```

/

```
ALTER TABLE Wlasciciele ADD CONSTRAINT Jest_wlascicielem FOREIGN KEY (Osoba_id) REFERENCES
Osoby (Osoba_ID)
```

/

```
ALTER TABLE Oplaty ADD CONSTRAINT Ma_oplaty FOREIGN KEY (Mieszkanie_ID) REFERENCES
Mieszkania (Mieszkanie_ID)
/
```

```
ALTER TABLE Czysze ADD CONSTRAINT Oplaty_za_czynsz FOREIGN KEY (Oplaty_ID) REFERENCES
Oplaty (Oplaty_ID)
/
```

```
ALTER TABLE Zuzycia ADD CONSTRAINT Generuje_zuzycia FOREIGN KEY (Mieszkanie_ID) REFERENCES
Mieszkania (Mieszkanie_ID)
/
```

```
ALTER TABLE Zuzycia ADD CONSTRAINT Oplaty_za_zuzycie FOREIGN KEY (Oplaty_ID) REFERENCES
Oplaty (Oplaty_ID)
/
```

```
ALTER TABLE Pracownicy ADD CONSTRAINT Zatrudnia FOREIGN KEY (Spoldzielnia_mieszkaniowa_ID)
REFERENCES Spoldzielnie_mieszkaniowe (Spoldzielnia_mieszkaniowa_ID)
/
```

```
ALTER TABLE Wlasciciele ADD CONSTRAINT Ma_adres FOREIGN KEY (Adres_ID) REFERENCES Adresy
(Adres_ID)
/
```

```
ALTER TABLE Spoldzielnie_mieszkaniowe ADD CONSTRAINT Ma_siedzibe_pod_adresem FOREIGN KEY
(Adres_ID) REFERENCES Adresy (Adres_ID)
/
```

```
ALTER TABLE Firmy_uslugowe ADD CONSTRAINT Jest_pod_adresem FOREIGN KEY (Adres_ID)
REFERENCES Adresy (Adres_ID)
/
```

```
ALTER TABLE Pracownicy ADD CONSTRAINT Posiada_adres FOREIGN KEY (Adres_ID) REFERENCES
Adresy (Adres_ID)
/
```

```
ALTER TABLE Wynagrodzenia ADD CONSTRAINT Ma_wynagrodzenie FOREIGN KEY (Pracownik_ID)
REFERENCES Pracownicy (Pracownik_ID)
/
```

```
ALTER TABLE Wlasciciele ADD CONSTRAINT Ma_wlascicieli FOREIGN KEY (Mieszkanie_ID)
REFERENCES Mieszkania (Mieszkanie_ID)
/
```

```
ALTER TABLE Mieszkancy ADD CONSTRAINT Ma_mieszkancow FOREIGN KEY (Mieszkanie_ID) REFERENCES
Mieszkania (Mieszkanie_ID)
/
```

```
ALTER TABLE Czynsze ADD CONSTRAINT Ma_czynsz FOREIGN KEY (Mieszkanie_ID) REFERENCES
Mieszkania (Mieszkanie_ID)
/
```

5.4 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych

- Pobranie danych adresowych wszystkich tych lokali zarządzanych przez spółdzielnię, które mają powierzchnię co najmniej 80 m²:

```
SELECT a.ulica, a.numer_domu, m.numer_lokalu, a.miasto, a.kod_pocztowy
FROM mieszkania m, bloki b, adresy a
WHERE m.blok_id=b.blok_id
AND b.adres_id=a.adres_id
AND m.powierzchnia >= 80;
```

	ULICA	NUMER_DOMU	NUMER_LOKALU	MIASTO	KOD_POCZTOWY
1	Krakowska 1A		44	Wrocław	01-112
2	Krakowska 2		69	Wrocław	01-112

- Pobranie danych o zaległych opłatach za media w bloku znajdującym się przy ulicy *Krakowskiej 1A*:

```

SELECT m.numer_lokalu, o.data_wystawienia_rachunku,
(z.ilosc_zuzycia * md.cena_za_jednostke) as kwota_do_zaplaty,
o.za_okres_od, o.za_okres_do, md.nazwa as medium
FROM oplaty o, mieszkania m, bloki b, adresy a, zuzycia z, media md
WHERE o.mieszkanie_id=m.mieszkanie_id
AND m.blok_id=b.blok_id AND b.adres_id=a.adres_id
AND a.ulica='Krakowska' AND a.numer_domu='1A'
AND z.oplaty_id=o.oplaty_id
AND z.medium_id=md.medium_id
AND o.czy_oplacony='N';

```

	NUMER_LOKALU	DATA_WYSTAWIENIA_RACHUNKU	KWOTA_DO_ZAPLATY	ZA_OKRES_OD	ZA_OKRES_DO	MEDIUM
1	44	22/03/04	302	22/02/01	22/02/28	Prąd
2	44	22/03/04	107,42	22/02/01	22/02/28	Gaz
3	44	22/02/04	116,44	22/01/01	22/01/31	Gaz

- Sprawdzenie ile lat przepracowali na stanowisku obecni pracownicy biura spółdzielni:

```

SELECT p.imie, NVL(p.drugie_imie, '-') as drugie_imie,
p.nazwisko, pb.rodzaj_pracownika_biurowego,
(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM p.data_zatrudnienia)) as lat_na_stanowisku
FROM pracownicy_biurowi pb, pracownicy p
WHERE pb.pracownik_id=p.pracownik_id
AND p.data_zwolnienia IS NULL
ORDER BY lat_na_stanowisku DESC;

```

	IMIE	DRUGIE_IMIE	NAZWISKO	RODZAJ_PRACOWNIKA_BIUROWEGO	LAT_NA_STANOWISKU
1	Paweł	-	Krawczyk	prezes	17
2	Marzena	-	Komar	dyrektor	4

- Sprawdzenie daty zakupu mieszkania tych obecnych mieszkańców, którzy są również właścicielami swoich mieszkań:

```

SELECT o.imie, o.nazwisko, o.data_urodzenia, w.data_nabycia_mieszkania
FROM mieszkancy m, wlascciele w, osoby o
WHERE m.osoba_id=o.osoba_id
AND w.osoba_id=m.osoba_id
AND w.data_zbycia_mieszkania IS NULL
AND m.data_wyprowadzki IS NULL;

```

	IMIE	NAZWISKO	DATA_URODZENIA	DATA_NABYCIA_MIESZKANIA
1	Janusz	Mars	91/12/31	14/01/18
2	Beata	Kwaśna	71/11/13	16/06/01
3	Michał	Przezięty	70/10/11	16/06/01
4	Britney	Spears	81/12/02	16/06/01

Bibliografia

[1] Wykłady z przedmiotu Bazy Danych i Big Data (BDBT) - *dr hab. inż Marcin Kowalczyk*