Salon samochodowy

Natan Orzechowski

October 2020

Spis treści

1 Założenia oraz opis działalności biznesowej

Realizowany projekt dotyczy przedsiębiorstwa z branży samochodowej. Przedsiębiorstwo to zajmuje się sprzedażą oraz najmem aut, a także świadczeniem usług ubezpieczeniowych. Częścią administracyjną przedsiebiorstwa są biura, które prowadzą salony samochodowe na terenie całego kraju.

1.1 Sprzedaż aut

W wybranym salonie samochodowym, auta można nabyć za gotówkę (bez rat, kredytów). Na finalną cenę auta składają się:

- Auto
- Koszty transportu
- Marża salonu samochodowego
- Przeciętne wynagrodzenie osób zaangażowanych w obsługę procesu sprzedaży

1.2 Najem długoterminowy

Klient, zgłosiwszy chęć wynajęcia auta na przynajmniej rok, podpisuje umowę obligującą go do comiesięcznych płatności. Na finalną cenę najmu składają się:

- Amortyzacja samochodu (obliczana przez specjalistę na podstawie wartości auta)
- Koszty transportu
- Marża salonu samochodowego
- Przeciętne wynagrodzenie osób zaangazowanych w obsługę procesu sprzedaży

1.3 Ubezpieczenia

Każdy salon prowadzi usługę ubezpieczeń aut w trzech wariantach: pakiet brązowy, pakiet srebrny, pakiet złoty. Rodzaj pakietu determinuje cenę ubezpieczenia oraz wysokość ewentualnych odszkodowań z tytułu poniesionych strat, które jednak nie są obliczane przez przedsiębiorstwo, a zewnętrznych rzeczoznawców.

1.4 Podsumowanie opisu działalności

W przypadku każdej z powyższych usług, wymaga się od klienta podania danych personalnych, dzięki którym firma tworzy i utrzymuje bazę klientów. W bazie przechowywane są również informacje dotyczące pracowników, w szczególności są to dane personalne, wynagrodzenie, okres zawarcia umowy, stanowisko.

2 Definicja systemu

2.1 Zidentyfikowane perspektywy użytkowników

- **Kierownik przedsiębiorstwa** - ma dostep do wszystkich danych
- **Księgowa** - ma dostęp do wszystkich danych związanych z kosztami
- **Pracownik** - ma dostęp do swoich danych osobowych, historii wynagrodzenia, a także do zadań wynikających z aktualnej pracy

2.2 Zidentyfikowane transakcje

2.2.1 W obszarze obsługa pracowników

- Dodanie i usunięcie pracownika z bazy danych
- Modyfikacja danych osobowych
- Wprowadzenie danych o wynagrodzeniu, ich modyfikacji oraz usunięcia
- Zdefiniowanie pracownika jako opiekuna pojazdu

2.2.2 W obszarze biura

- Zdefiniowanie biura oraz salonów samochodowych
- Zapewnienie możliwości ich obsługi (modyfikacja, usuwanie danych)

2.2.3 W obszarze usług przedsiębiorstwa

- Dodawanie nowych aut: konkretna marka, model oraz specyfikacja
- Definiowanie jego wartości, dostępności
- Definiowanie kosztów transportu
- Definiowanie ubezpieczeń wykupionych przez klientów wraz z inforamcją o długości umowy, cenie, pakiecie
- Definiowanie transakcji kupna/najmu

2.2.4 W obszarze klienta

- Obsługa klienta; w szczególności:
  1. Dodawanie danych klienta do bazy
  2. Możliwość ich modyfikacji
  3. Dane klienta nie mogą być usunięte z bazy, jeśli skorzystał on z jakiejkolwiek usługi przedsiębiorstwa
- Przypisywanie klienta do usługi

3 Model konceptualny

3.1 Definicja zbiorów encji określonych w projekcie

Biorąc pod uwagę założenia przedstawionej działalności biznesowej, wyróżniono następujące encje:

1. Biuro
2. Klient
3. Pojazd
4. Pracownik
5. Salon
6. Usługa ze specjalizacjami:
    - Sprzedaż
    - Wynajem
    - Ubezpieczenie

Definiując encje, miano na celu podkreślenie konkretnych obiektów, składających się na istnienie przedsiębiorstwa, ze szczególnym uwzględnieniem usług świadczonych przez firmę. Encja Usługa powstała w celu nie powtarzania zdefiniowanych atrybutów, które w większości dla encji Sprzedaż, Wynajem oraz Ubezpieczenie, byłyby wspólne.

3.2 Ustalenie związków między encjami i ich typów

Szczegółowe zestawienie istniejących związków oraz ich typów, zamieszczono w dołączonym do niniejszej dokumentacji raporcie w zakładce **Relationships**. Poniżej zamieszczono proces dobierania związków z innymi encjami dla encji Biuro w celach demonstracyjnych.
Funkcją biura w przedsiębiorstwie, jako organu administracyjnego, będzie zatrudnianie pracowników, sprawowanie pieczy nad salonami, a także nabywanie aut. Oznacza to, iż encje reprezentujące wyżej wymienione obiekty, muszą być połączone:

Połączenie z	Rodzaj związku	Uzasadnienie
Pracownik	1 do N	Biuro **zatrudnia** pracowników (na początku może nie istnieć żaden pracownik, tylko właściciel), a więc N. Z drugiej strony każdy pracownik będzie zatrudniony przez jedno i tylko jedno biuro.
Salon	1 do N	Jedno biuro może **posiadać** wiele salonów z regionu lub też żaden, jeśli żaden salon jeszcze nie powstał. Z drugiej strony zaś, każdy salon podlegać może pod jedno i tylko jedno biuro, gdyż żaden salon nie istnieje niezależnie od jednostki administracyjnej.
Pojazd	1 do N	Biuro **posiada pojazdy**, lecz może też nie mieć żadnego, np. na początku działalności czy w wyniku wyprzedania wszystkich aut. Każdy pojazd zaś należy tylko do tego biura, które je nabyło.

3.3 Ustalenie atrybutów i ich dziedzin

Każda encja zyskała stosowne do opisywanego obiektu atrybuty. W doborze dziedzin atrybutów, kierowano się zdrowym rozsądkiem i minimalizacją miejsca wymaganego poprzez bazę danych, a więc nie stosowanie typów BigInt, stosowanie typu varchar. Szczegółowa lista zdefiniowanych atrybutów znajduje się w załączonym dokumencie w zakładce **Attributes**.

3.4 Dodatkowe reguły integralnościowe

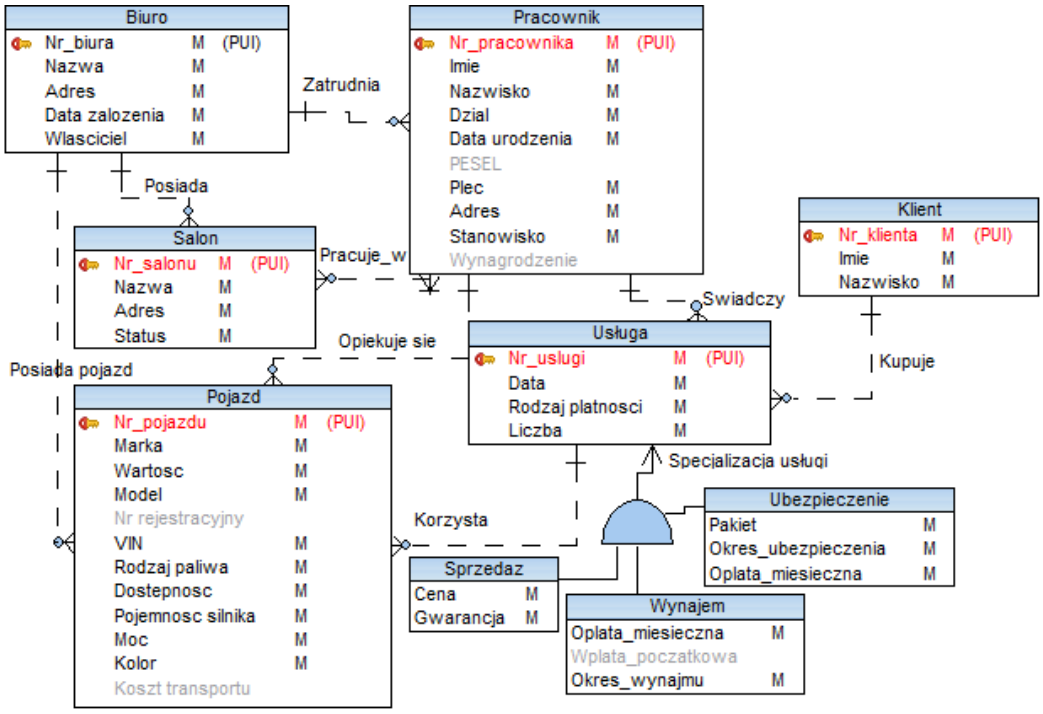
Wiedząc z góry, iż przedsiębiorstwo będzie oferować 3 pakiety ubezpieczeniowe lub też zatrudniać pracowników do działu administracji lub sprzedaży, zdefiniowano reguły zapewniające integralność poszczególnych atrybutów zdefiniowanych w krotkach należących do jednej tabeli. Pełen wykaz zdefiniowanych reguł znajduje się w zakładce **Rules**.

3.5 Klucze kandydujące i główne.

Po zdefiniowaniu atrybutów, w części encji można by wyróżnić przynajmniej jeden klucz kandydujący, złożony z wielu atrybutów. Dla przykładu, w encji Biuro, minimalnym zestawem atrybutów jednoznacznie identyfikującym krotkę, może być Nazwa oraz Adres biura, gdyż zakłada się, iż pod jednym adresem nie będą istnieć dwie oddzielne jednostki administracyjne (w najgorszym razie będą rózniły się numerem pokoju). Jednakże nie w każdej encji możliwe było wyłonienie takiego zestawu atrybutów, toteż każda encja zyskała atrybut, będący unikalnym identyfikatorem poszczególnych krotek. Przynosi to korzyści zarówno pod względem identyfikacji krotek, jak i biorąc pod uwagę późniejszą

konieczność normalizacji bazy danych. Każdy klucz główny jest kluczem prostym typu Integer. Szczegółowy wykaz kluczy znajduje się w zakładce **Unique Identifiers**.

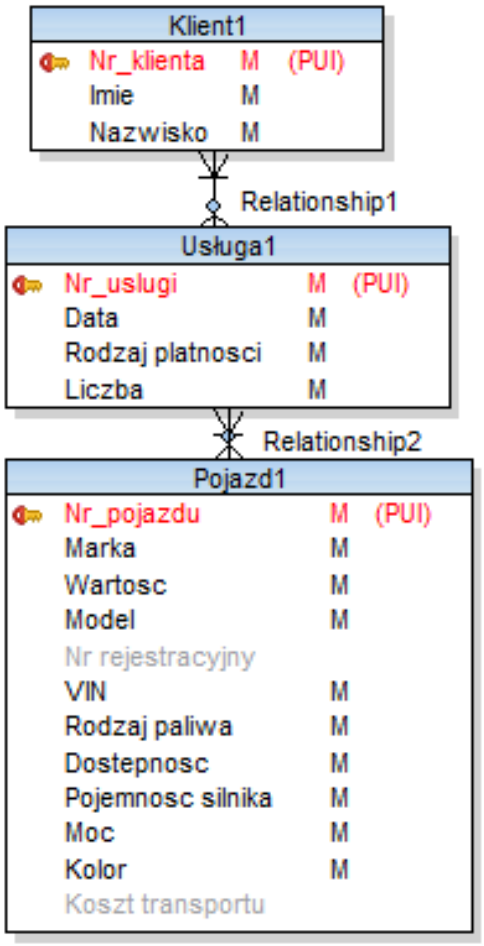
3.6 Schemat ER na poziomie konceptualnym



3.7 Problem pułapek szczelinowych i wachlarzowych

3.7.1 Pułapki wachlarzowe

W trakcie projektowania na poziomie konceptualnym, napotkano problem, polegający na niemożności ustalenia, do którego klienta należy które auto.

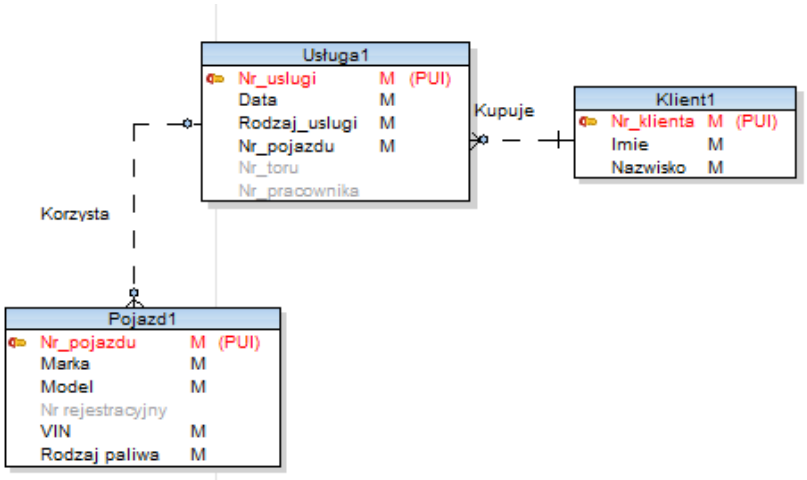


Jeżeli dopuścimy przypadek, w którym dwóch klientów zakupiło dwa auta w ramach jednej usługi, to nie jesteśmy w stanie powiedzieć, który klient nabył które auto. Jest to ewidentny błąd, który skorygowano poprzez zaostrzenie relacji Usługa - Klient do maksymalnie jednego klienta przypisanego do pojedynczej transakcji.
Konkludując: pułapka wachlarzowa występuje wówczas, gdy jedna encja jest związana z innymi encjami co najmniej dwoma związkami 1 do N.

3.7.2 Pułapki szczelinowe

Modyfikując powyższy przypadek, można by doprowadzić do sytuacji, w której związek encji Usługa i Pojazd nie wymaga krotki żadnej z encji. Może to doprowadzić do sytuacji, w której auta sprzedane w ramach usługi nie będą z nią stowarzyszone, a w rezultacie zapytanie do bazy danych o sprzedane auta w

ramach usługi x nie dostarczy nam żadnych informacji na ten temat.



Niepożądana jest więc sytuacja, w której minimalna krotność w związku wynosi 0.

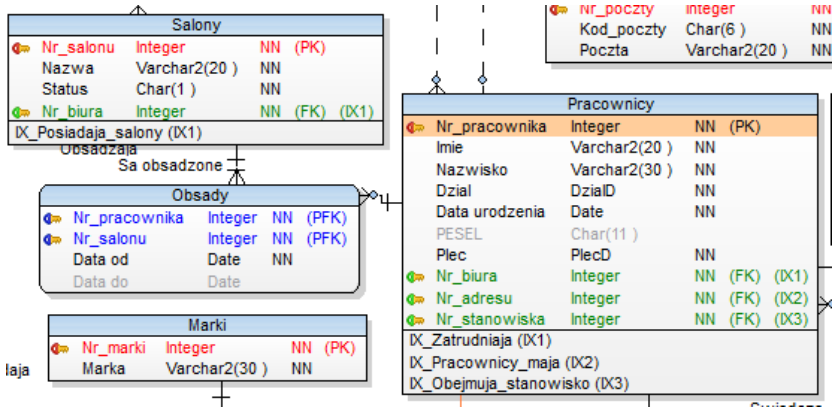
4 Model logiczny

4.1 Charakterystyka modelu relacyjnego

Model relacyjny to model organizacji danych bazujący na matematycznej teorii mnogości oraz rachunku predykatów pierwszego rzędu. W ujęciu modelu relacyjnego, dane grupowane są w relacje (tabele), które posiadają kolumny (atrybuty) i wiersze (krotki). W wyniku związków encji, w krotkach obu encji występują klucze obce, a więc unikalne identyfikatory pochodzące od "sąsiada". Dzięki takiemu podejściu, zyskujemy niezależność danych, uporządkowaną strukturę, w której łatwiej jest rozwiązywać problemy semantyki, spójności i redundancji danych, co z punktu widzenia projektanta jest szczególnie ważne.

4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym

Ten etap polega na zidentyfikowaniu oraz usunięciu związków wiele do wielu. W przedstawionym planie konceptualnym takowy zachodzi pomiędzy encjami Pracownik oraz Salon. Można taki związek zastąpić tabelą o dodatkowych kolumnach, odpowiadających kluczom głównym zbiorów encji biorących udział w związku. Wówczas dla jednego z kluczy możemy utworzyć dowolnie wiele "stowarzyszeń" z wybranymi, kluczami drugiego zbioru encji. Związek wiele do wielu zastąpiły dwa związki 1 do wielu. Tabela, która łączy obie encje, zyskała dwa dodatkowe atrybuty, aby móc definiować okres obsady danego pracownika w



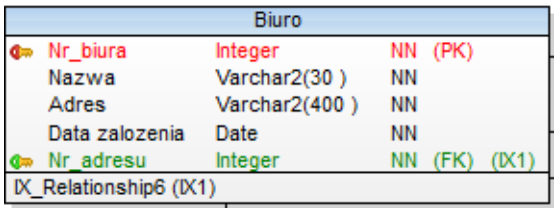
danym salonie. Dzięki temu nie zachodzi konieczność bieżącej aktualizacji bazy danych w przypadku zmian w obsadzie, a także zyskujemy możliwość prowadzenia historii obsad poszczególnych salonów.

4.3 Proces normalizacji - analiza

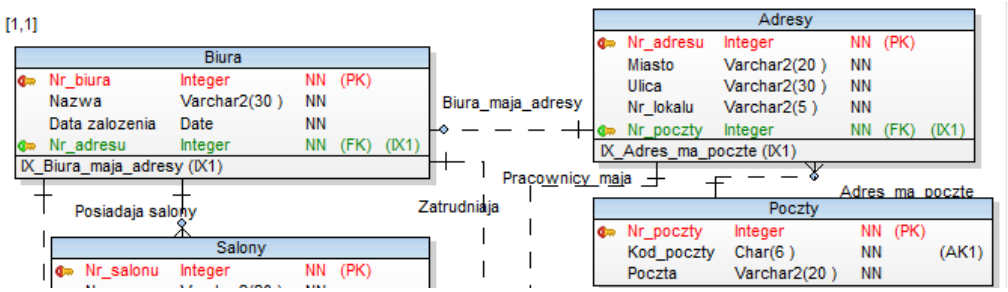
Proces normalizacji, a właściwie dekompozycji zbiorów rekordów w bazie danych, skutkuje ułatwioną pracą z danymi. Nie zachodzi konieczność edytowania wielu rekordów na potrzeby pojedynczych zmian. Ponadto, zabezpiecza to przed przypadkowym kasowaniem istotnych danych, związanych z kasowaniem innych danych, należących do tej samej krotki.
Prezentowana baza danych, po przekonwertowaniu na model logiczny, została poddana analizie pod kątem spełnienia wymogów trzeciej postaci normalnej, a więc przeanalizowano uprzednio zarówno pierwszą (w wersji restrykcyjnej) jak i drugą. Baza danych spełnia wymogi danej postaci normalnej wtedy i tylko wtedy, gdy wszystkie jej relacje je spełniają. Poniżej podano przykłady normalizacji.

4.3.1 Pierwsza postać normalna

Relacja jest w pierwszej postaci normalnej, jeśli każda wartość atrybutu w każdej krotce tej relacji jest wartością elementarną (atomową), czyli nierozkładalną. Ponadto, nie mogą występować powtarzające się grupy.



Adres biura ma zawierać informacje o mieście, ulicy, numerze lokalu oraz kodzie pocztowym, jest to więc pole segmentowe, a więc nie jest to wartość elementarna. Należy więc dokonać dekompozycji, czyli wydzielenia adresu biura do oddzielnej encji.



Pole segmentowe zostało rozbite na pojedyncze atrybuty. Ponadto, powtarzające się wartości atrybutów (siedziba poczty, kod pocztowy) dla adresów położonych geograficzne obok siebie, zostały wydzielone do kolejnej encji, aby spełnić wymóg rygorystycznej pierwszej postaci normalnej.

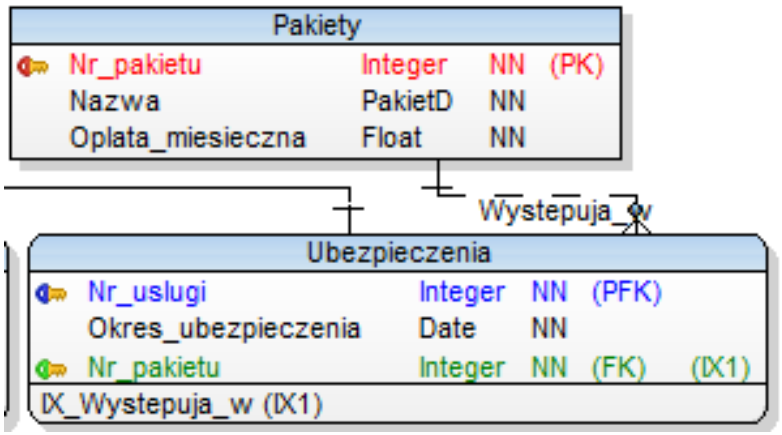
4.3.2 Druga postać normalna

Relacja będąca w pierwszej postaci normalnej, jest równocześnie w drugiej postaci normalnej, jeśli wszystkie jej klucze potencjalne są kluczami prostymi.
Na tym etapie należy upewnić się, iż każda relacja ma zdefiniowany swój Klucz główny i jest on typem prostym - w przypadku omawianej bazy danych są one typem Integer.

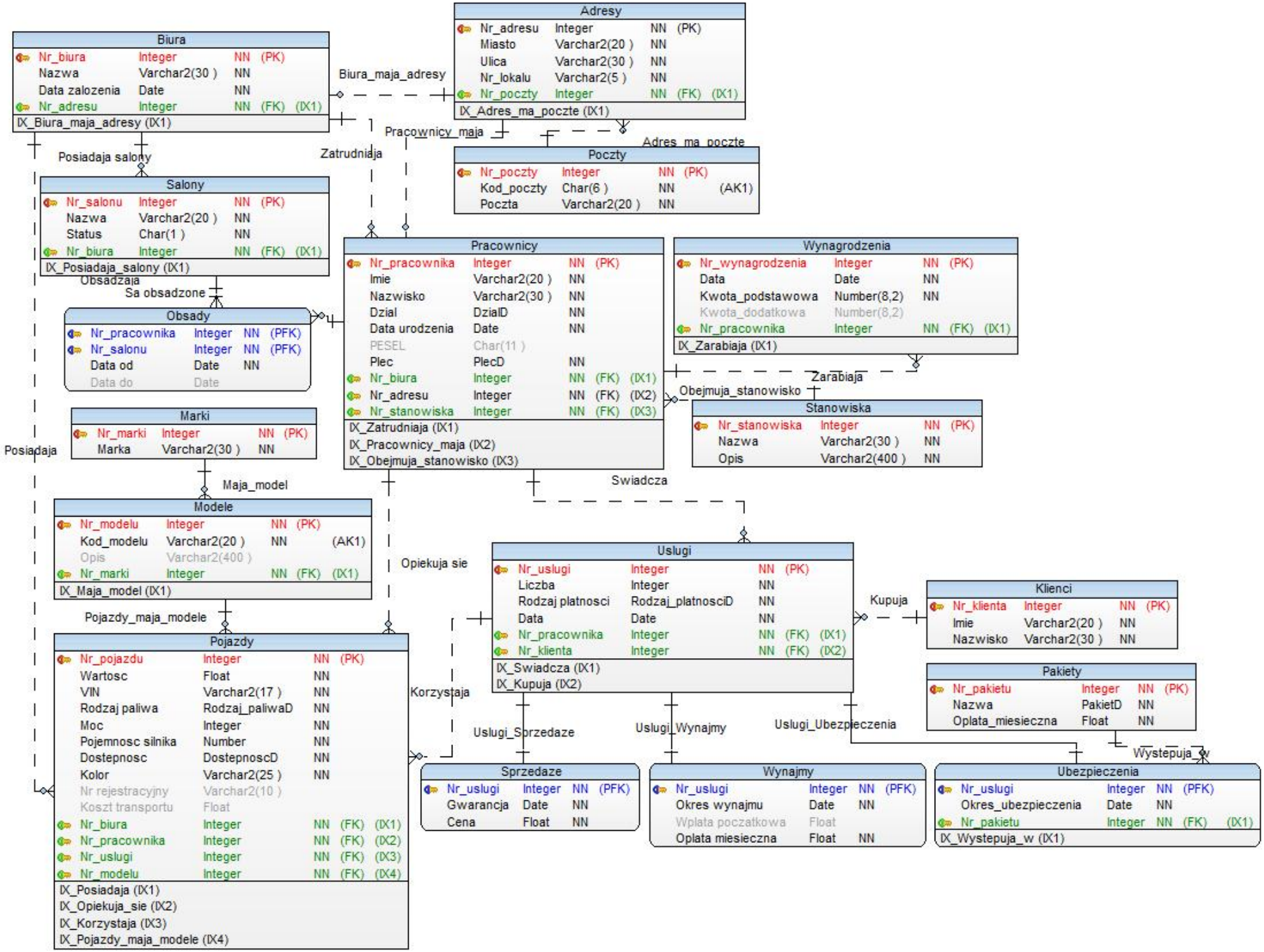
4.3.3 Trzecia postać normalna

Relacja będąca w drugiej postaci normalnej, jest równocześnie w trzeciej postaci normalnej, jeśli wszystkie niekluczowe kolumny są określone kluczem i tylko kluczem. W omawianej bazie danych zaszedł przypadek zależności ceny pakietu od nazwy pakietu.

Rozwiązano ten problem poprzez wydzielenie oddzielnej encji pakiety, w której zawarto nazwę oraz cenę. Nazwa należy do domeny definiującej konkretne 3 nazwy pakietów: Brązowy, Srebrny oraz Złoty.



4.4 Schemat ER na poziomie modelu logicznego


Adresy
Nr_adresu Integer NN (PK)
Miasto Varchar2(20) NN
Ulica Varchar2(30) NN
Nr_lokalu Varchar2(5) NN
Nr_poczty Integer NN (FK) (IX1)
IX_Adres_ma_poczte (IX1)
Biura
Nr_biura Integer NN (PK)
Nazwa Varchar2(30) NN
Data zalozenia Date NN
Nr_adresu Integer NN (FK) (IX1)
IX_Biura_maja_adresy (IX1)
Biura_maja_adresy
Pracownicy_maja
Adres_ma_poczte
Poczty
Nr_poczty Integer NN (PK)
Kod_poczty Char(6) NN (AK1)
Poczta Varchar2(20) NN
Posiadaja salony
Zatrudniaja
Salony
Nr_salonu Integer NN (PK)
Nazwa Varchar2(20) NN
Status Char(1) NN
Nr_biura Integer NN (FK) (IX1)
IX_Posiadaja_salony (IX1)
Obsadzaja
Sa obsadzone
Obsady
Nr_pracownika Integer NN (PFK)
Nr_salonu Integer NN (PFK)
Data od Date NN
Data do Date
Pracownicy
Nr_pracownika Integer NN (PK)
Imie Varchar2(20) NN
Nazwisko Varchar2(30) NN
Dzial DzialD NN
Data urodzenia Date NN
PESEL Char(11)
Plec PlecD NN
Nr_biura Integer NN (FK) (IX1)
Nr_adresu Integer NN (FK) (IX2)
Nr_stanowiska Integer NN (FK) (IX3)
IX_Zatrudniaja (IX1)
IX_Pracownicy_maja (IX2)
IX_Obejmuja_stanowisko (IX3)
Wynagrodzenia
Nr_wynagrodzenia Integer NN (PK)
Data Date NN
Kwota_podstawowa Number(8,2) NN
Kwota_dodatkowa Number(8,2)
Nr_pracownika Integer NN (FK) (IX1)
IX_Zarabiaja (IX1)
Zarabiaja
Obejmuja_stanowisko
Stanowiska
Nr_stanowiska Integer NN (PK)
Nazwa Varchar2(30) NN
Opis Varchar2(400) NN
Marki
Nr_marki Integer NN (PK)
Marka Varchar2(30) NN
Posiadaja
Maja_model
Modele
Nr_modelu Integer NN (PK)
Kod_modelu Varchar2(20) NN (AK1)
Opis Varchar2(400)
Nr_marki Integer NN (FK) (IX1)
IX_Maja_model (IX1)
Swiadcza
Opiekuja sie
Uslugi
Nr_uslugi Integer NN (PK)
Liczba Integer NN
Rodzaj platnosci Rodzaj_platnosciD NN
Data Date NN
Nr_pracownika Integer NN (FK) (IX1)
Nr_klienta Integer NN (FK) (IX2)
IX_Swiadcza (IX1)
IX_Kupuja (IX2)
Kupuja
Klienci
Nr_klienta Integer NN (PK)
Imie Varchar2(20) NN
Nazwisko Varchar2(30) NN
Pakiety
Nr_pakietu Integer NN (PK)
Nazwa PakietD NN
Oplata_miesieczna Float NN
Pojazdy_maja_modele
Pojazdy
Nr_pojazdu Integer NN (PK)
Wartosc Float NN
VIN Varchar2(17) NN
Rodzaj paliwa Rodzaj_paliwaD NN
Moc Integer NN
Pojemnosc silnika Number NN
Dostepnosc DostepnoscD NN
Kolor Varchar2(25) NN
Nr rejestracyjny Varchar2(10)
Koszt transportu Float
Nr_biura Integer NN (FK) (IX1)
Nr_pracownika Integer NN (FK) (IX2)
Nr_uslugi Integer NN (FK) (IX3)
Nr_modelu Integer NN (FK) (IX4)
IX_Posiadaja (IX1)
IX_Opiekuja_sie (IX2)
IX_Korzystaja (IX3)
IX_Pojazdy_maja_modele (IX4)
Korzystaja
Uslugi_Sprzedaze
Uslugi_Wynajmy
Uslugi_Ubezpieczenia
Wystepuja_w
Sprzedaze
Nr_uslugi Integer NN (PFK)
Gwarancja Date NN
Cena Float NN
Wynajmy
Nr_uslugi Integer NN (PFK)
Okres wynajmu Date NN
Wplata poczatkowa Float
Oplata miesieczna Float NN
Ubezpieczenia
Nr_uslugi Integer NN (PFK)
Okres_ubezpieczenia Date NN
Nr_pakietu Integer NN (FK) (IX1)
IX_Wystepuja_w (IX1)

4.5 Więzy integralności

Więzy integralności służą ochronie przed utraceniem spójności danych w trakcie ich modyfikacji. Ochronę taką osiągnięto poprzez zastosowanie:

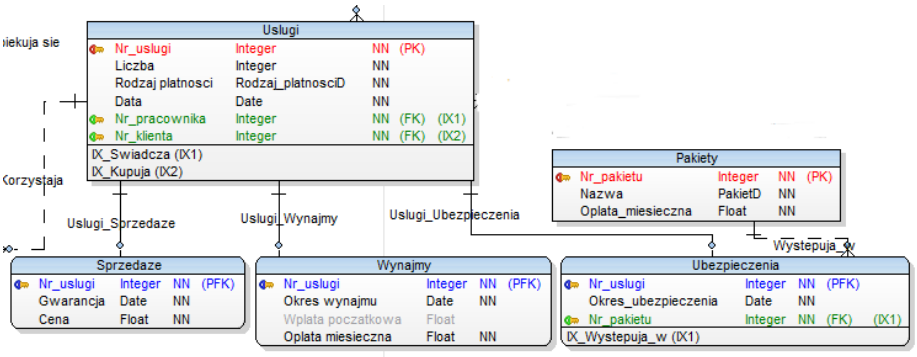
- Kluczy głównych - gwarantują unikalność wartości w kolumnie
- Kluczy obcych - gwarantują, iż rekord w tabeli podrzędnej zawsze będzie miał swojego odpowiednika w tabeli nadrzędnej
- Wartości NOT NULL

4.6 Proces denormalizacji - analiza

Omawianą strukturę poddano analizie pod kątem następujących metod denormalizacji:

4.6.1 Łączenie specjalizacji

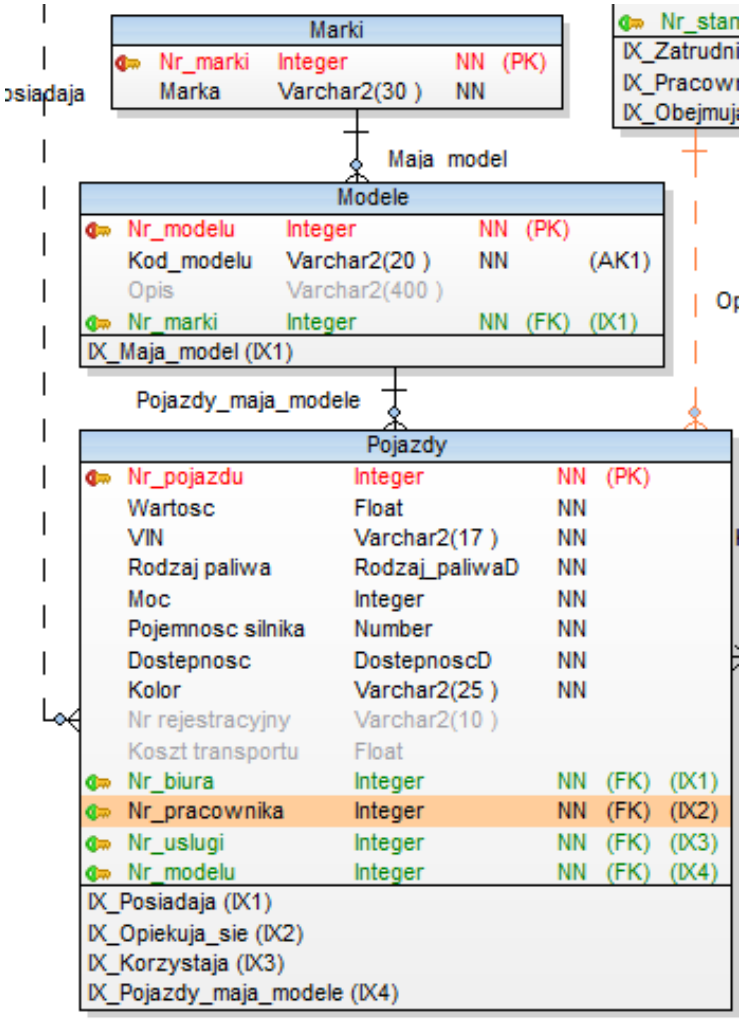
Łączenie specjalizacji może być korzystne w przypadku, w którym wyszczególniamy mało (a zwłaszcza jedną) specjalizację encji. W przypadku, w którym istnieje wiele tabel, które nie stanowią specjalizacji, będą w nich istniały luki (poszczególne pola pozostaną puste). W niniejszej bazie danych występuje jedna specjalizacja:



W tym przypadku, przyłączenie atrybutów należących do specjalizacji do encji głównej nie jest uzasadnione. Usługa nie będzie istnieć bez zdefiniowania jej jednej specjalizacji, zaś stworzenie niespecjalizowanej struktury stworzyłoby problemy z poprawnością wprowadzonych danych. Atrybuty należące do usługi sprzedaży, wynajmu i ubezpieczenia nie mogłyby być obowiązkowe, a w takiej sytuacji, wprowadzając dane jako usługę sprzedaży, można by nie wpisać choćby ceny, co jest sytuacją niedopuszczalną.
Opisanej metody denormalizacji nie zastosowano.

4.6.2 Redukcja liczby złączeń poprzez powielanie atrybutów, które nie należą do kluczy w związkach 1:N

Ten zabieg może okazać się korzystny w przypadku, w którym popularne zapytanie do bazy danych musi przywołać dwie struktury na potrzeby niewielkiej ilości danych, np. szukamy imienia i nazwiska pracownika, a także nazwy biura, w którym pracuje. Zamieszczenie w encji Pracownik atrybutu Nazwa_biura pozwoliłoby na uproszczenie procedury realizacji zapytania.
W przypadku analizowanej bazy danych, najczęstszym zapytaniem prawdopodobnie będzie pytanie o dostępne pojazdy danej marki/modelu.

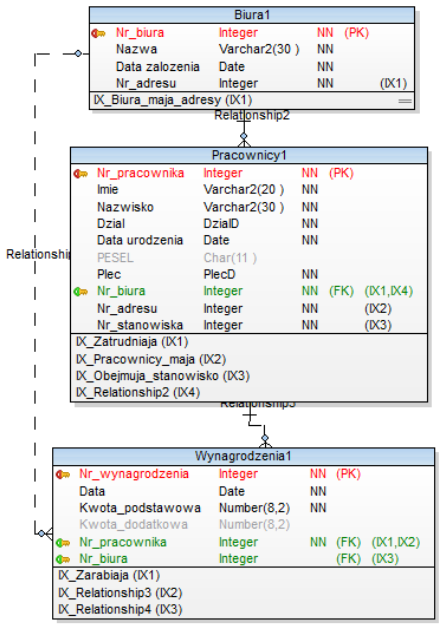


Optymalizacja zapytania zwłaszcza dot. konkretnej marki wydaje się być możliwa do osiągnięcia poprzez przyłączenie encji Marki oraz Modele do encji Po-

jazdy. Należy jednak rozważyć kwestię użyteczności bazy danych. W opisanym scenariuszu powstaną trudności związane z jednolitością opisu poszczególnych modeli. Dla przykładu: w jednym dostępnym aucie marki Audi modelu A4 będzie mógł figurować inny opis niż w drugim egzemplarzu tego samego modelu. Ponadto, wskutek błędu ludzkiego, poszczególne egzemplarze będą mogły zawierać np. literówki w kodach modelu, co w przypadku ujednoliconego zapisu (osiągniętego poprzez wydzielenie encji Modele od Pojazdy) będzie proste do wychwycenia przez pracowników. Wady wskazanego rozwiązania wydają się być bardziej znaczne od potencjalnych korzyści.
Wskazana metoda denormalizacji nie została zastosowana.

4.6.3 Redukcja liczby złączeń poprzez powielanie atrybutów kluczy obcych w związkach 1:N

W przypadku zapytania o listę pracowników z uwzględnieniem biur, w których pracują oraz pogrupowanych wg zarobków, wywołać należy 3 struktury: Biura, Pracownicy, Wynagrodzenia. Rozważyć można optymalizację takiego zabiegu, którą można osiągnąć poprzez połączenie encji Biura oraz Wynagrodzenia relacją 1 do N (z brakiem wymogu istnienia krotki tabeli Wynagrodzenia).

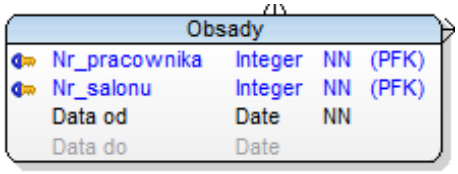


W ten sposób encja Wynagrodzenia otrzymuje nowy klucz obcy: Nr_biura. Powstaje jednak cykl. Wymóg integralności wymusza konieczność zmiany pola Nr_biura w dwóch miejscach, lecz wydaje się, iż pole to nie będzie ulegało zmianom często. Należy jednak wziąć pod uwagę aspekt biznesowy: o ile zmiany tych pól nie są częste, o tyle ewentualne błędy mogące mieć skutek na nieporządek

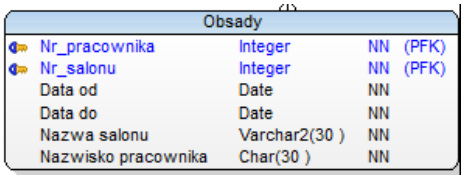
w księgowości firmy (rozważania odbywają się wokół tabeli wynagrodzeń), toteż zdecydowano o wyższości niezawodności bazy danych nad jej szybkością działania.
Wskazana metoda denormalizacji nie została zastosowana.

4.6.4 Redukcja liczby złączeń poprzez powielanie atrybutów w związkach N:M

Metoda ta polega na wprowadzeniu dodatkowego atrybutu do tabeli łączącej dwie encje o relacji N:M (powstałej w wyniku działań podjętych w punkcie 4.2) w ten sposób, by zawrzeć w niej informacje wystarczające do realizacji popularnego zapytania dot. obu encji. Zapytaniem popularnym niewątpliwie będzie pytanie o to, który pracownik, w jakim okresie, pracuje w którym salonie (lub inaczej: jaka jest obsada wskazanego salonu). Zapytanie to będzie się tyczyło encji Obsady.



Choć informacje wymagane do identyfikacji zarówno salonu jak i pracownika znajdują się już we wskazanej tabeli, użytkownikowi niewątpliwie byłoby wygodniej operować na nazwie salonu oraz nazwisku pracownika (uznaje się, iż nazwisko jest wystarczającą informacją do identyfikacji, lub w najgorszym razie znacznego zawężenia pola poszukiwań, pracownika). W warunkach normalnej pracy, zapytanie do bazy danych najpewniej zawierałoby również prośbę o te informacje. W celu umożliwienia realizacji takiego zapytania bez wywoływania trzech tabel, można dodać do encji Obsady redundantne pola: Nazwisko oraz Nazwa_salonu.



Oczywiście, w tej sytuacji znów powstaje problem integralności danych, lecz uznaje się, iż zmiany zachodzące w tabeli obsada będą okresowe, a także nie dość częste, aby powodować problemy. Ponadto, ewentualne skutki błędów wynikających z niepoprawnienia danych w tym miejscu wydają się nie mieć strategicznego znaczenia dla przedsiębiorstwa.
Wskazana metoda denormalizacji została zastosowana.

4.6.5 Wprowadzenie powtarzających się grup

Opisana metoda oznacza odwrotność bardziej restrykcyjnej wersji Pierwszej Postaci Normalnej. Chcąc utrzymać strukturę możliwie znormalizowaną w celu jej długiego, niezawodnego działania, nie podjęto działań.
Wskazana metoda denormalizacji nie została zastosowana.

4.6.6 Scalenie tabel podglądu z relacjami bazowymi

Opisana metoda oznacza usunięcie relacji słownikowych ze struktury. Zastosować można by ją było włączając encję Poczty do encji Adresy tak, aby zachodziła konieczność każdorazowego wypełniania pól Kod poczty oraz Poczta dla każdego nowego adresu biura. Metoda ta mogłaby również być zastosowana w połączeniu encji Pakiety i Ubezpieczenia z analogicznym skutkiem. Nie zauważono korzyści płynących z zastosowania tego rozwiązania.
Wskazana metoda denormalizacji nie została zastosowana.

4.6.7 Tworzenie tabel skrótowych (agregatów)

Żadna operacja nie wydaje się wprowadzać konieczności większej agregacji danych, toteż ze względu na integralność danych, nie podjęto kroków.
Wskazana metoda denormalizacji nie została zastosowana.

5 Faza fizyczna

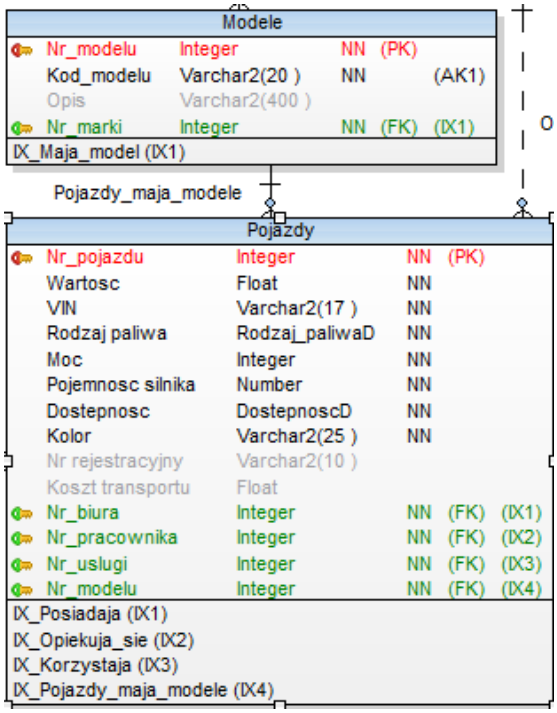
5.1 Projekt transakcji i weryfikacja ich wykonalności

Poniżej zamieszczono przewidziane transakcje, które zostały pogrupowane wg obszarów wyróżnionych w punkcie 2.2 z uwzględnieniem warunków ich realizacji.

Obszar	Transakcja	Zasób	Zawsze możliwe?	Uwagi
Obsługa pracowników	Dodanie i usunięcie pracownika	Pracownicy	Nie	Musi istnieć Biuro, Adres, Stanowisko
	Modyfikacja danych osobowych	Pracownicy, Adresy	Tak	Brak
	Wprowadzenie danych o wynagrodzeniu, ich modyfikacji oraz usunięcia	Wynagrodzenie	Nie	Musi istnieć Pracownik
	Zdefiniowanie pracownika jako opiekuna pojazdu	Pojazdy	Nie	Musi istnieć Pracownik
Biura	Zdefiniowanie biura	Biura	Tak	Brak
	Zdefiniowanie salonu samochodowego	Salony	Nie	Musi istnieć Biuro
	Modyfikacja, usuwanie danych biura	Biura, Adresy	Tak	Brak
	Modyfikacja, usuwanie danych Salony	Salony	Tak	Brak
Usługi przedsiębiorstwa	Dodawanie marki	Marki	Tak	Brak
	Dodawanie modelu	Modele, Marki	Nie	Musi istnieć Marka
	Dodawanie pojazdu	Pojazdy, Modele, Marki	Nie	Musi istnieć Marka, Model
	Dodawanie/modyfikowanie specyfikacji auta	Pojazdy	Nie	Musi istnieć Pojazd
	Definiowanie wartości, dostępności auta	Pojazdy	Nie	Musi istnieć Pojazd
	Definiowanie kosztów transportu	Pojazdy	Nie	Musi istnieć Pojazd
	Definiowanie ubezpieczeń	Pakiety	Tak	Brak
	Definiowanie/modyfikacja specyfikacji Ubezpieczenia	Pakiety, Ubezpieczenia, Usługi, Pracownicy	Nie	Musi istnieć Pakiet, Pracownik
	Definiowanie transakcji kupna/sprzedaży	Usługi, Pracownicy, Pojazdy, Klienci	Nie	Musi istnieć Klient, Pracownik, Pojazd
Obszar klienta	Dodawanie danych klienta	Klienci	Tak	Brak
	Modyfikacja danych klienta	Klienci	Nie	Musi istnieć Klient
	Usuwanie danych klienta	Klienci	Nie	Niemożliwe, jeśli skorzystał z jakiejkolwiek usługi
	Przypisanie klienta do usługi	Klienci, Usługi	Nie	Musi istnieć usługa

5.2 Strojenie bazy danych - dobór indeksów

Indeksy stanowią tabelę przypisaną do innej tabeli, zawierającej informacje. Indeksy zawierają informacje o ROWID (identyfikatorze wiersza), w którym daną informację mozna znaleźć. Indeksy są posortowane rosnąco. Korzystając z ich właściwości, w celu przyspieszenia realizacji zapytań do prezentowanej bazy danych, dla każdej tabeli, zawierającej związek z inną tabelą charakteru dziecko - rodzic, wygenerowano tabelę unikalnych indeksów (odnoszących się do kluczy obcych danej tabeli).



Przykładem wartości dodanej, którą wprowadzają indeksy, może być scenariusz, w którym istnieje konkretna krotka tabeli Pojazdy:

Nr_pojazdu (PK)	Wartosc	(...)	Nr_modelu (FK)
1	100000		104

Chcąc znaleźć informacje dot. modelu auta, a w szczególności Kod_modelu, silnik bazy danych przeszukuje indeksy *IX_Pojazdy_maja_modele* pod kątem wystąpienia Nr_modelu = 104. Znalazłszy odpowiedni, unikalny indeks, wie, do którego wiersza tabeli Modele powinien się odwołać. Choć w tym przypadku tabela Model nie będzie zawierać więcej niż jednego wiersza z numerem modelu, gdyż jest to PK, na korzyść szybkości wyszukiwania działać będzie posortowana zawartość tabeli z indeksami.
Nie definiowano dodatkowych indeksów, mając na uwadze fakt, iż indeksy spowalniają operacje DML.

5.3 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych

Dla zademonstrowania działania bazy danych, dodano do niej nową pocztę oraz adres:

```
INSERT ALL
    INTO ADRESY (MIASTO, ULICA, NR_LOKALU, NR_POCZTY) VALUES ('Warszawa', 'Niebieska', '13/2', 8)
    INTO ADRESY (MIASTO, ULICA, NR_LOKALU, NR_POCZTY) VALUES ('Warszawa', 'Zolta', '1', 8)
    INTO ADRESY (MIASTO, ULICA, NR_LOKALU, NR_POCZTY) VALUES ('Warszawa', 'Zielona', '10/22', 8)
SELECT * FROM dual;
```

W efekcie powstały nowe rekordy w tabelach.

NR_ADRESU	MIASTO	ULICA	NR_LOKALU	NR_POCZTY
1	Warszawa	Sosnowa	92	5
2	Warszawa	Kwiatowa	6	4
3	Warszawa	Chabrowa	10	4
4	Warszawa	Smakosza	126	5
5	Warszawa	Pileckiego	1	6
6	Bydgoszcz	Podgorna	26	3
7	Warszawa	Przyjaciol	110	7
8	Warszawa	Wesola	61	6
9	Warszawa	Artylerii Konnej	102	5
10	Warszawa	Niebieska	13/2	8
12	Warszawa	Zolta	1	8
13	Warszawa	Zielona	10/22	8

Jako, że tabela Adresy zawiera klucz obcy Nr_poczty, można sformułować zapytanie o to, jakie są znane ulice, które są przypisane do wskazanego kodu pocztowego:

```
SELECT ADRESY.ULICA, POCZTY.KOD_POCZTY FROM ADRESY NATURAL JOIN POCZTY
WHERE KOD_POCZTY = '01-010';
```

Wynik to wcześniej zdefiniowane adresy:

```
ULICA                          KOD_POCZTY
------------------------------ ----------
Niebieska                      01-010
Zolta                          01-010
Zielona                        01-010
```

Zwrócony wynik jest poprawny.

5.4 Skrypt SQL zakładający bazę danych

```
        /*
Created: 04.11.2020
Modified: 15.11.2020
Model: Salon samochodowy PL
Database: Oracle 19c
*/


-- Create sequences section -------------------------------------------------

CREATE SEQUENCE BiuraSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE AdresySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE StanowiskaSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE PracownicySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE WynagrodzeniaSeq1
 NOMAXVALUE
```

```
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE PojazdySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE PocztySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE SalonySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE MarkiSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE ModeleSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE UslugiSeq1
 INCREMENT BY 1
```

```
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE KlienciSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE PakietySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

-- Create tables section -------------------------------------------------

-- Table Biura

CREATE TABLE Biura(
  Nr_biura Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL,
  Data_zalozenia Date NOT NULL,
  Nr_adresu Integer NOT NULL
)
/

-- Create indexes for table Biura

CREATE INDEX IX_Biura_maja_adresy ON Biura (Nr_adresu)
/

-- Add keys for table Biura

ALTER TABLE Biura ADD CONSTRAINT Biuro_PK PRIMARY KEY (Nr_biura)
/

-- Table Salony
```

```
CREATE TABLE Salony(
  Nr_salonu Integer NOT NULL,
  Nazwa Varchar2(20 ) NOT NULL,
  Status Char(1 ) NOT NULL,
  Nr_biura Integer NOT NULL
)
/

-- Create indexes for table Salony

CREATE INDEX IX_Posiadaja_salony ON Salony (Nr_biura)
/

-- Add keys for table Salony

ALTER TABLE Salony ADD CONSTRAINT Unique_Identifier2 PRIMARY KEY (Nr_salonu)
/

-- Table Pracownicy

CREATE TABLE Pracownicy(
  Nr_pracownika Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  Dzial Varchar2(13 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  PESEL Char(11 ),
  Plec Char(1 ) NOT NULL
        CHECK (Plec IN ('K','M')),
  Data_umowy Date,
  Nr_biura Integer NOT NULL,
  Nr_adresu Integer NOT NULL,
  Nr_stanowiska Integer NOT NULL
)
/

-- Create indexes for table Pracownicy

CREATE INDEX IX_Zatrudniaja ON Pracownicy (Nr_biura)
/

CREATE INDEX IX_Pracownicy_maja ON Pracownicy (Nr_adresu)
/

CREATE INDEX IX_Obejmuja_stanowisko ON Pracownicy (Nr_stanowiska)
/
```

```
-- Add keys for table Pracownicy

ALTER TABLE Pracownicy ADD CONSTRAINT Unique_Identifier3 PRIMARY KEY (Nr_pracownika)
/

-- Table and Columns comments section

COMMENT ON COLUMN Pracownicy.Dzial IS 'Dzial'
/
COMMENT ON COLUMN Pracownicy.Data_umowy IS 'Data wygaśnięcia umowy z pracownikiem '
/

-- Table Pojazdy

CREATE TABLE Pojazdy(
  Nr_pojazdu Integer NOT NULL,
  Wartosc Float NOT NULL,
  VIN Varchar2(17 ) NOT NULL,
  Rodzaj_paliwa Varchar2(20 ) NOT NULL
        CHECK (Rodzaj_paliwa In ('OLEJ','BENZYNA','LPG','ELEKTRYCZNY','INNY')),
  Moc Integer NOT NULL,
  Pojemnosc_silnika Number NOT NULL,
  Dostepnosc Char(3 ) NOT NULL
        CHECK (Dostepnosc In ('TAK','NIE')),
  Kolor Varchar2(25 ) NOT NULL,
  Nr_rejestracyjny Varchar2(10 ),
  Koszt_transportu Float,
  Nr_biura Integer NOT NULL,
  Nr_pracownika Integer NOT NULL,
  Nr_uslugi Integer NOT NULL,
  Nr_modelu Integer NOT NULL
)
/

-- Create indexes for table Pojazdy

CREATE INDEX IX_Posiadaja ON Pojazdy (Nr_biura)
/

CREATE INDEX IX_Opiekuja_sie ON Pojazdy (Nr_pracownika)
/

CREATE INDEX IX_Korzystaja ON Pojazdy (Nr_uslugi)
/
```

```
CREATE INDEX IX_Pojazdy_maja_modele ON Pojazdy (Nr_modelu)
/

-- Add keys for table Pojazdy

ALTER TABLE Pojazdy ADD CONSTRAINT Unique_Identifier8 PRIMARY KEY (Nr_pojazdu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Pojazdy.Wartosc IS 'Wartosc auta'
/
COMMENT ON COLUMN Pojazdy.Moc IS 'Moc silnika'
/
COMMENT ON COLUMN Pojazdy.Dostepnosc IS 'Dostepnosc auta (tak lub nie)'
/
COMMENT ON COLUMN Pojazdy.Kolor IS 'Kolor auta'
/
COMMENT ON COLUMN Pojazdy.Koszt_transportu IS 'Koszt transportu do klienta (salonu)'
/

-- Table Klienci

CREATE TABLE Klienci(
  Nr_klienta Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL
)
/

-- Add keys for table Klienci

ALTER TABLE Klienci ADD CONSTRAINT Unique_Identifier13 PRIMARY KEY (Nr_klienta)
/

-- Table Obsady

CREATE TABLE Obsady(
  Nr_pracownika Integer NOT NULL,
  Nr_salonu Integer NOT NULL,
  Data_od Date NOT NULL,
  Data_do Date NOT NULL,
  Nazwa_salonu Varchar2(30 ) NOT NULL,
  Nazwisko_pracownika Char(30 ) NOT NULL
)
/
```

```
-- Table and Columns comments section

COMMENT ON COLUMN Obsady.Data_od IS 'Data, od której pracownik będzie realizował obsadę'
/
COMMENT ON COLUMN Obsady.Data_do IS 'Data, do której pracownik będzie realizował obsadę'
/
COMMENT ON COLUMN Obsady.Nazwa_salonu IS 'Nazwa salonu'
/
COMMENT ON COLUMN Obsady.Nazwisko_pracownika IS 'Nazwisko pracownika'
/

-- Table Adresy

CREATE TABLE Adresy(
  Nr_adresu Integer NOT NULL,
  Miasto Varchar2(20 ) NOT NULL,
  Ulica Varchar2(30 ) NOT NULL,
  Nr_lokalu Varchar2(5 ) NOT NULL,
  Nr_poczty Integer NOT NULL
)
/

-- Create indexes for table Adresy

CREATE INDEX IX_Adres_ma_poczte ON Adresy (Nr_poczty)
/

-- Add keys for table Adresy

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

-- Table and Columns comments section

COMMENT ON COLUMN Adresy.Nr_adresu IS 'Unikatowy numer adresu'
/
COMMENT ON COLUMN Adresy.Miasto IS 'Miasto w adresie'
/
COMMENT ON COLUMN Adresy.Ulica IS 'Ulica w adresie'
/
COMMENT ON COLUMN Adresy.Nr_lokalu IS 'Numer lokalu w adresie (np. 6A/10)'
/

-- Table Poczty
```

```
CREATE TABLE Poczty(
  Nr_poczty Integer NOT NULL,
  Kod_poczty Char(6 ) NOT NULL,
  Poczta Varchar2(20 ) NOT NULL
)
/

-- Add keys for table Poczty

ALTER TABLE Poczty ADD CONSTRAINT PK_Poczty PRIMARY KEY (Nr_poczty)
/

ALTER TABLE Poczty ADD CONSTRAINT Kod_poczty UNIQUE (Kod_poczty)
/

-- Table and Columns comments section

COMMENT ON COLUMN Poczty.Nr_poczty IS 'Unikatowy identyfikator poczty'
/
COMMENT ON COLUMN Poczty.Kod_poczty IS 'Kod pocztowy'
/
COMMENT ON COLUMN Poczty.Poczta IS 'Siedziba poczty'
/

-- Table Stanowiska

CREATE TABLE Stanowiska(
  Nr_stanowiska Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL,
  Opis Varchar2(400 ) NOT NULL
)
/

-- Add keys for table Stanowiska

ALTER TABLE Stanowiska ADD CONSTRAINT PK_Stanowiska PRIMARY KEY (Nr_stanowiska)
/

-- Table and Columns comments section

COMMENT ON COLUMN Stanowiska.Nr_stanowiska IS 'Unikatowy identyfikator stanowiska'
/
COMMENT ON COLUMN Stanowiska.Nazwa IS 'Nazwa stanowiska'
/
COMMENT ON COLUMN Stanowiska.Opis IS 'Opis stanowiska'
/
```

```
-- Table Wynagrodzenia

CREATE TABLE Wynagrodzenia(
  Nr_wynagrodzenia Integer NOT NULL,
  Data Date NOT NULL,
  Kwota_podstawowa Number(8,2) NOT NULL,
  Kwota_dodatkowa Number(8,2),
  Nr_pracownika Integer NOT NULL
)
/

-- Create indexes for table Wynagrodzenia

CREATE INDEX IX_Zarabiaja ON Wynagrodzenia (Nr_pracownika)
/

-- Add keys for table Wynagrodzenia

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

-- Table and Columns comments section

COMMENT ON COLUMN Wynagrodzenia.Nr_wynagrodzenia IS 'Unikatowy numer wynagrodzenia'
/
COMMENT ON COLUMN Wynagrodzenia.Data IS 'Data wypłaty'
/
COMMENT ON COLUMN Wynagrodzenia.Kwota_podstawowa IS 'Kwota podstawowa wynagrodzenia'
/
COMMENT ON COLUMN Wynagrodzenia.Kwota_dodatkowa IS 'Kwota dodatkowa (premia)'
/

-- Table Uslugi

CREATE TABLE Uslugi(
  Nr_uslugi Integer NOT NULL,
  Liczba Integer NOT NULL,
  Rodzaj platnosci Char(7 ) NOT NULL
        CHECK (Rodzaj_platnosci In ('PARAGON','FAKTURA')),
  Data Date NOT NULL,
  Nr_pracownika Integer NOT NULL,
  Nr_klienta Integer NOT NULL
)
/
```

```
-- Create indexes for table Uslugi

CREATE INDEX IX_Swiadcza ON Uslugi (Nr_pracownika)
/

CREATE INDEX IX_Kupuja ON Uslugi (Nr_klienta)
/

-- Add keys for table Uslugi

ALTER TABLE Uslugi ADD CONSTRAINT PK_Uslugi PRIMARY KEY (Nr_uslugi)
/

-- Table and Columns comments section

COMMENT ON COLUMN Uslugi.Nr_uslugi IS 'Unikatowy identyfiaktor uslugi'
/
COMMENT ON COLUMN Uslugi.Liczba IS 'Liczba zakupionych obiektow (aut, ubezpieczen)'
/
COMMENT ON COLUMN Uslugi.Rodzaj platnosci IS 'Rodzaj płatności (paragon lub faktura)'
/
COMMENT ON COLUMN Uslugi.Data IS 'Data'
/

-- Table Sprzedaze

CREATE TABLE Sprzedaze(
  Nr_uslugi Integer NOT NULL,
  Gwarancja Date NOT NULL,
  Cena Float NOT NULL
)
/

-- Add keys for table Sprzedaze

ALTER TABLE Sprzedaze ADD CONSTRAINT PK_Sprzedaze PRIMARY KEY (Nr_uslugi)
/

-- Table and Columns comments section

COMMENT ON COLUMN Sprzedaze.Gwarancja IS 'Gwarancja (podana data stanowi okres upłynięcia um
/
COMMENT ON COLUMN Sprzedaze.Cena IS 'Cena'
/

-- Table Wynajmy
```

```
CREATE TABLE Wynajmy(
  Nr_uslugi Integer NOT NULL,
  Okres_wynajmu Date NOT NULL,
  Wplata_poczatkowa Float,
  Oplata_miesieczna Float NOT NULL
)
/

-- Add keys for table Wynajmy

ALTER TABLE Wynajmy ADD CONSTRAINT PK_Wynajmy PRIMARY KEY (Nr_uslugi)
/

-- Table and Columns comments section

COMMENT ON COLUMN Wynajmy.Okres_wynajmu IS 'Okres wynajmu'
/
COMMENT ON COLUMN Wynajmy.Wplata_poczatkowa IS 'Wpłata początkowa'
/
COMMENT ON COLUMN Wynajmy.Oplata_miesieczna IS 'Opłata za miesiąc użytkowania auta'
/

-- Table Ubezpieczenia

CREATE TABLE Ubezpieczenia(
  Nr_uslugi Integer NOT NULL,
  Okres_ubezpieczenia Date NOT NULL,
  Nr_pakietu Integer NOT NULL
)
/

-- Create indexes for table Ubezpieczenia

CREATE INDEX IX_Wystepuja_w ON Ubezpieczenia (Nr_pakietu)
/

-- Add keys for table Ubezpieczenia

ALTER TABLE Ubezpieczenia ADD CONSTRAINT PK_Ubezpieczenia PRIMARY KEY (Nr_uslugi)
/

-- Table and Columns comments section

COMMENT ON COLUMN Ubezpieczenia.Okres_ubezpieczenia IS 'Okres ubezpieczenia'
/
```

```
-- Table Marki

CREATE TABLE Marki(
  Nr_marki Integer NOT NULL,
  Marka Varchar2(30 ) NOT NULL
)
/

-- Add keys for table Marki

ALTER TABLE Marki ADD CONSTRAINT PK_Marki PRIMARY KEY (Nr_marki)
/

-- Table and Columns comments section

COMMENT ON TABLE Marki IS 'Encja Marka'
/
COMMENT ON COLUMN Marki.Nr_marki IS 'Unikatowy identyfikator marki'
/
COMMENT ON COLUMN Marki.Marka IS 'Marka'
/

-- Table Modele

CREATE TABLE Modele(
  Nr_modelu Integer NOT NULL,
  Kod_modelu Varchar2(20 ) NOT NULL,
  Opis Varchar2(400 ),
  Nr_marki Integer NOT NULL
)
/

-- Create indexes for table Modele

CREATE INDEX IX_Maja_model ON Modele (Nr_marki)
/

-- Add keys for table Modele

ALTER TABLE Modele ADD CONSTRAINT PK_Modele PRIMARY KEY (Nr_modelu)
/

ALTER TABLE Modele ADD CONSTRAINT Kod_modelu UNIQUE (Kod_modelu)
/
```

```
-- Table and Columns comments section

COMMENT ON TABLE Modele IS 'Encja Model'
/
COMMENT ON COLUMN Modele.Nr_modelu IS 'Unikatowy identyfikator modelu'
/
COMMENT ON COLUMN Modele.Kod_modelu IS 'Kod modelu'
/
COMMENT ON COLUMN Modele.Opis IS 'Opis modelu'
/

-- Table Pakiety

CREATE TABLE Pakiety(
  Nr_pakietu Integer NOT NULL,
  Nazwa Varchar2(7 ) NOT NULL
        CHECK (Pakiet In ('ZLOTY','SREBRNY','BRAZOWY')),
  Oplata_miesieczna Float NOT NULL
)
/

-- Add keys for table Pakiety

ALTER TABLE Pakiety ADD CONSTRAINT PK_Pakiety PRIMARY KEY (Nr_pakietu)
/

-- Table and Columns comments section

COMMENT ON TABLE Pakiety IS 'Encja Pakiety'
/
COMMENT ON COLUMN Pakiety.Nr_pakietu IS 'Unikalny identyfikator pakietu'
/
COMMENT ON COLUMN Pakiety.Nazwa IS 'Nazwa pakietu (ZLOTY, SREBRNY lub BRAZOWY)'
/
COMMENT ON COLUMN Pakiety.Oplata_miesieczna IS 'Opłata miesięczna'
/

-- Trigger for sequence BiuraSeq1 for column Nr_biura in table Biura ---------
CREATE OR REPLACE TRIGGER ts_Biura_BiuraSeq1 BEFORE INSERT
ON Biura FOR EACH ROW
BEGIN
  :new.Nr_biura := BiuraSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Biura_BiuraSeq1 AFTER UPDATE OF Nr_biura
ON Biura FOR EACH ROW
```

```
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_biura in table Biura as it uses se
END;
/

-- Trigger for sequence SalonySeq1 for column Nr_salonu in table Salony ---------
CREATE OR REPLACE TRIGGER ts_Salony_SalonySeq1 BEFORE INSERT
ON Salony FOR EACH ROW
BEGIN
  :new.Nr_salonu := SalonySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Salony_SalonySeq1 AFTER UPDATE OF Nr_salonu
ON Salony FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_salonu in table Salony as it uses
END;
/

-- Trigger for sequence PracownicySeq1 for column Nr_pracownika in table Pracownicy --------
CREATE OR REPLACE TRIGGER ts_Pracownicy_PracownicySeq1 BEFORE INSERT
ON Pracownicy FOR EACH ROW
BEGIN
  :new.Nr_pracownika := PracownicySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_PracownicySeq1 AFTER UPDATE OF Nr_pracownika
ON Pracownicy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_pracownika in table Pracownicy as
END;
/

-- Trigger for sequence PojazdySeq1 for column Nr_pojazdu in table Pojazdy ---------
CREATE OR REPLACE TRIGGER ts_Pojazdy_PojazdySeq1 BEFORE INSERT
ON Pojazdy FOR EACH ROW
BEGIN
  :new.Nr_pojazdu := PojazdySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pojazdy_PojazdySeq1 AFTER UPDATE OF Nr_pojazdu
ON Pojazdy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_pojazdu in table Pojazdy as it use
END;
/
```

```
-- Trigger for sequence KlienciSeq1 for column Nr_klienta in table Klienci ---------
CREATE OR REPLACE TRIGGER ts_Klienci_KlienciSeq1 BEFORE INSERT
ON Klienci FOR EACH ROW
BEGIN
  :new.Nr_klienta := KlienciSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Klienci_KlienciSeq1 AFTER UPDATE OF Nr_klienta
ON Klienci FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_klienta in table Klienci as it use
END;
/

-- Trigger for sequence AdresySeq1 for column Nr_adresu in table Adresy ---------
CREATE OR REPLACE TRIGGER ts_Adresy_AdresySeq1 BEFORE INSERT
ON Adresy FOR EACH ROW
BEGIN
  :new.Nr_adresu := AdresySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Adresy_AdresySeq1 AFTER UPDATE OF Nr_adresu
ON Adresy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_adresu in table Adresy as it uses
END;
/

-- Trigger for sequence PocztySeq1 for column Nr_poczty in table Poczty ---------
CREATE OR REPLACE TRIGGER ts_Poczty_PocztySeq1 BEFORE INSERT
ON Poczty FOR EACH ROW
BEGIN
  :new.Nr_poczty := PocztySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Poczty_PocztySeq1 AFTER UPDATE OF Nr_poczty
ON Poczty FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_poczty in table Poczty as it uses
END;
/

-- Trigger for sequence StanowiskaSeq1 for column Nr_stanowiska in table Stanowiska --------
CREATE OR REPLACE TRIGGER ts_Stanowiska_StanowiskaSeq1 BEFORE INSERT
ON Stanowiska FOR EACH ROW
```

```
BEGIN
  :new.Nr_stanowiska := StanowiskaSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Stanowiska_StanowiskaSeq1 AFTER UPDATE OF Nr_stanowiska
ON Stanowiska FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_stanowiska in table Stanowiska as
END;
/

-- Trigger for sequence WynagrodzeniaSeq1 for column Nr_wynagrodzenia in table Wynagrodzenia
CREATE OR REPLACE TRIGGER ts_Wynagrodzenia_WynagrodzeniaSeq1 BEFORE INSERT
ON Wynagrodzenia FOR EACH ROW
BEGIN
  :new.Nr_wynagrodzenia := WynagrodzeniaSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Wynagrodzenia_WynagrodzeniaSeq1 AFTER UPDATE OF Nr_wynagrodzen
ON Wynagrodzenia FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_wynagrodzenia in table Wynagrodzen
END;
/

-- Trigger for sequence UslugiSeq1 for column Nr_uslugi in table Uslugi ---------
CREATE OR REPLACE TRIGGER ts_Uslugi_UslugiSeq1 BEFORE INSERT
ON Uslugi FOR EACH ROW
BEGIN
  :new.Nr_uslugi := UslugiSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Uslugi_UslugiSeq1 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
END;
/

-- Trigger for sequence MarkiSeq1 for column Nr_marki in table Marki ---------
CREATE OR REPLACE TRIGGER ts_Marki_MarkiSeq1 BEFORE INSERT
ON Marki FOR EACH ROW
BEGIN
  :new.Nr_marki := MarkiSeq1.nextval;
END;
/
```

```
CREATE OR REPLACE TRIGGER tsu_Marki_MarkiSeq1 AFTER UPDATE OF Nr_marki
ON Marki FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_marki in table Marki as it uses se
END;
/

-- Trigger for sequence ModeleSeq1 for column Nr_modelu in table Modele ---------
CREATE OR REPLACE TRIGGER ts_Modele_ModeleSeq1 BEFORE INSERT
ON Modele FOR EACH ROW
BEGIN
  :new.Nr_modelu := ModeleSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Modele_ModeleSeq1 AFTER UPDATE OF Nr_modelu
ON Modele FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_modelu in table Modele as it uses
END;
/

-- Trigger for sequence PakietySeq1 for column Nr_pakietu in table Pakiety ---------
CREATE OR REPLACE TRIGGER ts_Pakiety_PakietySeq1 BEFORE INSERT
ON Pakiety FOR EACH ROW
BEGIN
  :new.Nr_pakietu := PakietySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pakiety_PakietySeq1 AFTER UPDATE OF Nr_pakietu
ON Pakiety FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column Nr_pakietu in table Pakiety as it use
END;
/


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

ALTER TABLE Pracownicy ADD CONSTRAINT Zatrudniaja FOREIGN KEY (Nr_biura) REFERENCES Biura (N
/



ALTER TABLE Salony ADD CONSTRAINT Posiadaja_salony FOREIGN KEY (Nr_biura) REFERENCES Biura (
/
```

```
ALTER TABLE Pojazdy ADD CONSTRAINT Posiadaja FOREIGN KEY (Nr_biura) REFERENCES Biura (Nr_biu
/


ALTER TABLE Pojazdy ADD CONSTRAINT Opiekuja_sie FOREIGN KEY (Nr_pracownika) REFERENCES Praco
/


ALTER TABLE Adresy ADD CONSTRAINT Adres_ma_poczte FOREIGN KEY (Nr_poczty) REFERENCES Poczty
/


ALTER TABLE Biura ADD CONSTRAINT Biura_maja_adresy FOREIGN KEY (Nr_adresu) REFERENCES Adresy
/


ALTER TABLE Pracownicy ADD CONSTRAINT Pracownicy_maja FOREIGN KEY (Nr_adresu) REFERENCES Adr
/


ALTER TABLE Pracownicy ADD CONSTRAINT Obejmuja_stanowisko FOREIGN KEY (Nr_stanowiska) REFERE
/


ALTER TABLE Wynagrodzenia ADD CONSTRAINT Zarabiaja FOREIGN KEY (Nr_pracownika) REFERENCES Pr
/


ALTER TABLE Ubezpieczenia ADD CONSTRAINT Uslugi_Ubezpieczenia FOREIGN KEY (Nr_uslugi) REFERE
/


ALTER TABLE Wynajmy ADD CONSTRAINT Uslugi_Wynajmy FOREIGN KEY (Nr_uslugi) REFERENCES Uslugi
/
```

```
ALTER TABLE Sprzedaze ADD CONSTRAINT Uslugi_Sprzedaze FOREIGN KEY (Nr_uslugi) REFERENCES Usl
/


ALTER TABLE Uslugi ADD CONSTRAINT Swiadcza FOREIGN KEY (Nr_pracownika) REFERENCES Pracownicy
/


ALTER TABLE Uslugi ADD CONSTRAINT Kupuja FOREIGN KEY (Nr_klienta) REFERENCES Klienci (Nr_kli
/


ALTER TABLE Pojazdy ADD CONSTRAINT Korzystaja FOREIGN KEY (Nr_uslugi) REFERENCES Uslugi (Nr_
/


ALTER TABLE Modele ADD CONSTRAINT Maja_model FOREIGN KEY (Nr_marki) REFERENCES Marki (Nr_mar
/


ALTER TABLE Pojazdy ADD CONSTRAINT Pojazdy_maja_modele FOREIGN KEY (Nr_modelu) REFERENCES Mo
/


ALTER TABLE Ubezpieczenia ADD CONSTRAINT Wystepuja_w FOREIGN KEY (Nr_pakietu) REFERENCES Pak
/
```