Tomasz Boguski
Michał Krzemiński

27.01.2017r.

Wprowadzenie do Baz Danych

Projekt Stacji serwisowania pojazdów

Prowadzący mgr inż. Piotr Parewicz

Spis treści:

1. Zakres i cel projektu

Celem projektu jest zaprojektowanie (na poziomie konceptualnym oraz logicznym), a także fizyczna implementacja relacyjnej bazy danych obsługującej stacje serwisowania pojazdów. Założona baza danych będzie oparta na systemie zarządzania relacyjnymi bazami danych firmy Oracle. Zastosowanym językiem zapytań będzie ANSI SQL.

Oprogramowanie użyte podczas realizacji projektu:
● Oracle Database 12c
● Toad Data Modeler
● SQL Developer

Założenia funkcjonalne:

● W bazie są przechowywane dane dotyczące pojazdów i ich właścicieli którzy są klientami warsztatu
● W bazie są przechowywane dane o pojazdach takie jak marka i model samochodu, przebieg oraz rok produkcji pojazdu
● W bazie są przechowywane dane o dostawach wraz z przypisanymi do nich dostawcami i częściami które dostarczają.
● W bazie są przechowywane dane dotyczące wszystkich części jakich pracownicy warsztatu używają do naprawy pojazdów
● W bazie są przechowywane dane dotyczące wszystkich operacji wykonanych na pojeździe.

2. Definicja systemu

W ramach systemu bazy danych zdefiniowane zostały 4 perspektywy, zdefiniowane poprzez przypisanie użytkownikom systemu odpowiednich uprawnień.

2.1. Perspektywy użytkowników

Wyróżnione funkcjonalności systemu:

1. Podgląd danych personalnych pracowników.
2. Modyfikacja/dodawanie/usuwanie danych personalnych pracowników.
3. Podgląd danych personalnych klientów.
4. Modyfikacja/wprowadzanie/usuwanie danych personalnych klientów
5. Podgląd wykonywanych usług na pojazdach.
6. Wprowadzanie/usuwanie/modyfikacja danych dotyczących wykonywania usług na pojazdach.
7. Podgląd harmonogramu pracy każdego pracownika.
8. Podgląd historii serwisowania każdego z pojazdów.
9. Modyfikacja/wprowadzanie harmonogramy pracy dla warsztatu.
10. Uprawnienia do modyfikacji struktury bazy danych.
11. Podgląd w stan części na magazynie
12. Wgląd do danych wszystkich dostawców.
13. Wgląd w historię serwisowania każdego z pojazdów.
14. Możliwość przydziału stanowiska i urządzeń pod daną operację.
15. Możliwość otrzymania przez klienta karty stałego klienta.

Opis perspektyw występujących w systemie oraz dostępnych im funkcjonalności:

2.1.1. Administrator

Administrator ma dostęp do wszystkich funkcjonalności systemu oraz modyfikacji struktury bazy danych. Ma on uprawnienia administratora bazy danych Oracle (rola wytworzona automatycznie po stworzeniu bazy).

2.1.2. Właściciel warsztatu

Właściciel może zobaczyć wszystkie dane przechowywane w bazie danych. Zajmuje się opracowaniem grafiku dla swoich pracowników. Ma możliwość zatwierdzenia wszystkich dostaw a także ma uprawnienia do modyfikacji wyposażenia hali serwisowej.

2.13. Pracownik warsztatu

Pracownik ma podgląd do magazynu części, do harmonogramu pracy, wszystkie informacje potrzebne do wykonywania operacji.

2.1.4. Klient

Klient ma możliwość wprowadzenia do bazy danych swoich danych osobowych a także dane dotyczące swojego pojazdu. Klientowi zostaje udostępniona lista operacji jaką warsztat wykonuje na pojeździe i także może on wybrać jaką operację chcę zrealizować w przypadku naprawy. Dostaje informacje o planowanym czasie realizacji.

3. Model konceptualny

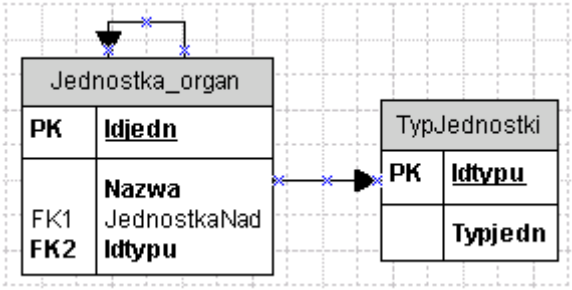
Model konceptualny jest to jeden z pierwszych etapów projektowania bazy danych. Jest to opis wymagać w postaci sformalizowanej, abstrahujący od problemów implementacyjnych. Przedstawia wszystkie wymagania funkcjonalne, definiuje encje i relacje między nimi, bez wnikania w użycie w implementacji. Jest przedstawiany za pomocą schematu ER.

3.1. Definicja zbiorów encji określonych w projekcie

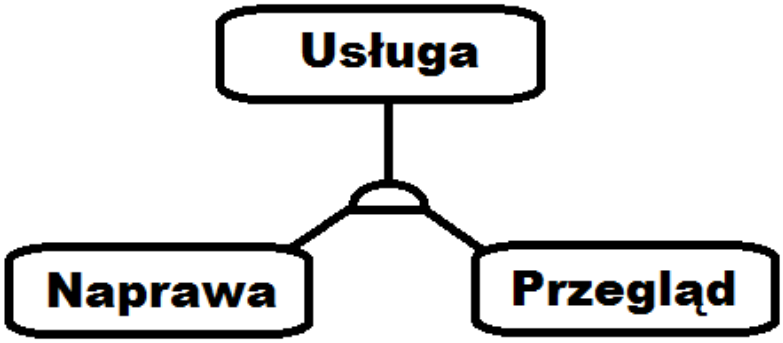
Encja – wyróżniony obiekt który powinien być odzwierciedlony w bazie danych. Własności encji zapisuje się za pomocą atrybutów.

Jest to decyzją projektową czy dany obiekt fizyczny powinien być reprezentowany jako oddzielna encja. Jeżeli ten obiekt nie będzie wykorzystany w wielu relacjach z innymi encjami to może zostać zapisany jako grupa atrybutów. W przeciwnym wypadku tworzone są dwie encje które są w relacji ze sobą.
Jako przykład możemy podać encje Model gdzie można by było umieścić informacje o modelu w encji Pojazd, jednakże tworzyło by to problem z przydzieleniem planu serwisowania dla każdego pojazdu.

Encja słownikowa – encja której celem jest ułatwić kontrolę poprawności wprowadzanych przez użytkownika wartości atrybutu. Jako przykład możemy podać encję jednostki organizacyjnej gdzie występuje atrybut *Typ*, którego wartością jest typ jednostki organizacyjnej np. "Departament", "Wydział". Zbiór takich wartości jest mało-liczny i rzadko ulega modyfikacji.



W projekcie mieliśmy zaprojektować przykład dziedziczenia podany na rysunku poniżej



Mieliśmy do wyboru 3 możliwości wyboru najlepszego rozwiązania do naszej bazy
1. Utworzenie jednej encji Usługa, gdzie atrybutami byłyby Naprawa i Przegląd. Została przez nas odrzucona ze względu na to iż chcielibyśmy aby móc wykonać naprawę bez przeglądu przez co nie było by to możliwe stosując to rozwiązanie.
2. Stworzenie jednej encji która by była podtypem jednocześnie naprawy i przeglądu i była by w z obiema encjami w relacji jeden do jednego. Tę możliwość także odrzuciliśmy ze względu na brak takich atrybutów.
3. Utworzenie dwóch encji Naprawa i Przegląd które są w relacji z encją Pojazd. Wybraliśmy tę opcję gdyż jest to rozsądne rozwiązanie dla naszej bazy, gdzie klient ma do wyboru czy chce aby zrobić przegląd czy też wykonać operacje naprawy na swoim pojeździe.

Przykłady opisu kliku encji:

Encja WLASCICIEL_POJAZDU:
Encja pozwalająca na przechowywanie podstawowych informacji o właścicielach pojazdów, a także możemy się dowiedzieć czy klient jest stałym klientem czy też nie.

Encja POJAZD:
Encja zawierająca dane wszystkich pojazdów których właściciele są klientami warsztatu.

Encja PRACOWNIK:
Encja przechowująca dane o pracownikach danego warsztatu. Możemy się dowiedzieć jaką operacje może wykonać każdy z pracowników.

3.2. Określenie atrybutów i ich dziedzin

Wszystkie atrybuty posiadające typ VARCHAR2 mają dziedzinę zawsze określoną, jako ilość znaków (CHAR) a nie, jako ilość bitów. Wszystkie klucze główne są obowiązkowe.

Atrybuty wartości typu INTEGER są liczbami całkowitymi z przedziału -32768 do 32767.
Atrybuty wartości typu NUMBER są liczbami zmiennoprzecinkowymi z precyzją do 38 znaków.
Atrybuty wartości typu DATE są zapisywane w formacie DD-MON-YY

Typy wyliczeniowe
Enum – typ danych wyliczeniowy może przechowywać wartości wybrane z danej listy, maksymalna ilość składowych listy wynosi 65535 elementów, np. Płeć w encji Właściciel_Pojazdu (M,K)

- Encja WLASCICIEL_POJAZDU

Nazwa atrybutu	Typ i dziedzina	Obowiązkowy	Unikatowy	Opis
Id_wlasciciela	INTEGER	TAK	TAK	Klucz główny
Imie	VARCHAR2(30)	TAK	NIE	Imię klienta
Nazwisko	VARCHAR2(30)	TAK	NIE	Nazwisko klienta
Miasto	VARCHAR2(30)	TAK	NIE	
Ulica	VARCHAR2(30)	TAK	NIE	
Nr_budynku	INTEGER	TAK	NIE	
Nr_lokalu	INTEGER	NIE	NIE	
Nr_telefonu	VARCHAR2(30)	NIE	NIE	
Adres_e_mail	VARCHAR2(30)	NIE	NIE	
Staly_klient	VARCHAR2(30)	TAK	NIE	Informacja czy klient jest stałym klientem

- Encja POJAZD

Nazwa atrybutu	Typ i dziedzina	Obowiązkowy	Unikatowy	Opis
Id_pojazdu	INTEGER	TAK	TAK	Klucz główny
Nr_rejestracyjny	VARCHAR2(30)	TAK	NIE	
Rok_produkcji	INTEGER	TAK	NIE	
Przebieg	INTEGER	TAK	NIE	Wartość w km

- Encja PRACOWNIK

Nazwa atrybutu	Typ i dziedzina	Obowiązkowy	Unikatowy	Opis
Id_pracownika	INTEGER	TAK	TAK	Klucz główny
Imie	VARCHAR2(30)			Imię pracownika
Nazwisko	VARCHAR2(30)			Nazwisko pracownika

Nr_telefonu	INTEGER			
Zawód	VARCHAR2(30)			Rodzaj wykonywanej pracy

3.3. Ustalenie związków i ich typów między encjami

Związek jeden do jednego - każdemu rekordowi (krotce) z tabeli nadrzędnej odpowiada dokładnie jeden rekord (jedna krotka) w powiązanej z nią tabeli podrzędnej i odwrotnie
przykład: jednemu modelowi pojazdu odpowiada jedna pozycja planu serwisowego.

Związek jeden do wielu - każdemu rekordowi (krotce) z tabeli nadrzędnej może odpowiadać wiele rekordów (krotek) w powiązanej z nią tabeli podrzędnej, ale każdemu rekordowi (krotce) z tabeli podrzędnej musi odpowiadać dokładnie j eden rekord w powiązanej z nią tabelą nadrzędną
przykład: jeden właściciel może mieć wiele pojazdów, ale pojazd ma jednego właściciela

Związek wiele do wielu - każdemu rekordowi (krotce) z tabeli nadrzędnej może odpowiadać wiele rekordów w powiązanej z nią tabeli podrzędnej i odwrotnie
przykład: jedna część do wielu operacji, wiele części do jednej operacji

Obowiązkowość atrybutów polega na konieczności zamieszczenia informacji danych np. pojazd nie może być zarejestrowany w bazie bez podania numeru rejestracyjnego.
Opcjonalność polega na tym że wartość atrybutu nie musi wystąpić np. numer lokalu Właściciela Pojazdu.

Wszystkie klucze główne są obowiązkowe.
Związki są obowiązkowe wtedy gdy dany obiekt nie może istnieć bez drugiego np. Książka nie może istnieć bez autora, natomiast autor może istnieć bez książki.

Związki nieobowiązkowe oznaczają iż dana encja może istnieć z pustymi kluczami obcymi.

3.4. Dodatkowe reguły integralnościowe

Warunki łaczące atrybuty różnych encji – atrybut w jednej encji wpływa na dziedzinę atrybutu w drugiej encji jeśli występuje między nimi relacja.

Przykład 1:
Mamy encje Zamówienie w której jest warunek mówiący że można złożyć zamówienie tylko o danej wadze mniejszej niż X. Atrybut waga jest wyliczany z atrybutów ilość (z encji Pozycja zamówienia) i masa(z encji Towar).

Przykład 2:
Dziedzina atrybutu zawód w encji pracownik jest określona przez możliwy zbiór wartości atrybutu NAZWA w encji Operacja. Pracownik może wykonywać tylko taki zawód który pozwala na wykonanie operacji dostępnej w naszym serwisie.

3.5. Klucze kandydujące i główne

Klucz główny – jest atrybutem który jednoznacznie identyfikuje dany rekord w tabeli. Jeśli klucz główny składa się z kilku pól określa się go mianem złożonego klucz głównego. Jest najistotniejszym kluczem w bazie danych.

Klucz obcy - kombinacja jednego lub wielu atrybutów tabeli, które wyrażają się w dwóch lub większej liczbie relacji. Wykorzystuje się go do tworzenia relacji pomiędzy parą tabel, gdzie w jednej tabeli ten zbiór atrybutów jest kluczem obcym, a w drugiej kluczem głównym. Jako przykład możemy podać encje Pojazd gdzie kluczem głównym jest ID_Pojazdu, a kluczem obcym jest ID_Wlasciciela gdzie ta wartość jest kluczem głównym w Encji Wlasciciel_Pojazdu.

Kluczem kandydującym możemy nazwać atrybuty które nie są klucze głównym ale także mogą jednoznacznie identyfikować dany atrybut w bazie danych.

Z przyczyn bezpieczeństwa zdecydowaliśmy się na stworzenie sztucznych kluczy ID. Wprowadzenie sztucznego klucza daje nam gwarancję jednoznaczności i spójności danych.
Poniżej znajdują się atrybuty które pełnią rolę potencjalnych kluczy kandydujących poszczególnych encji znajdujących się w naszej bazie.

Nazwa Encji	Potencjalny klucz kandydujący
CZESC	NR_SERYJNY
DOSTAWA	NUMER_DOSTAWY
POJAZD	NR_REJESTRACYJNY
PRACOWNIK	NR_TELEFONU
STANOWISKO	NR_STANOWISKA
URZADZENIE	NR_SERYJNY_URZADZENIA

3.6. Schemat ER na poziomie konceptualnym

Schemat ER na poziomie konceptualnym znajduje się w załączniku

3.7. Problem pułapek szczelinowych i wachlarzowych

Pułapka wachlarzowa – występuje w sytuacji, gdy model przedstawia związek pomiędzy pewnymi zbiorami encji, ale wynikające z tego ścieżki pomiędzy wystąpieniami encji nie są jednoznaczne. Pułapka taka może wystąpić gdy co najmniej dwa związki tylu jeden do wielu wychodzą z tej samej encji

Przykład :

Problem:

Rozwiązanie:

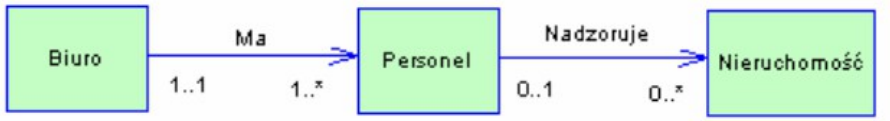
Zakładamy, że personel musi mieć przyporządkowane biuro, w pierwszym przypadku widzimy że każdy oddział przyporządkowuje pracowników ale nie mamy informacji w jakim biurze jest dany personel. W rozwiązaniu problemu widzimy że każdy pracownik jest przyporządkowany pod biuro a każde buro jest przyporządkowane pod dany oddział.

Pułapka szczelinowa – występuje, gdy model sugeruje istnienie związku pomiędzy zbiorami encji, ale nie istnieją ścieżki łączące pewne wystąpienia tych encji.

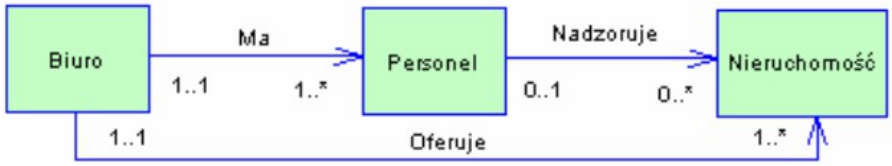
Pułapka taka może wystąpić gdy w modelu znajduje się co najmniej jeden związek o minimalnej krotności zero, który jest elementem ścieżki pomiędzy powiązanymi encjami

Przykład:

Problem:



Rozwiązanie:



Jeżeli nieruchomość nie jest powiązana z biurem, mogą wystąpić problemy dlatego też nieruchomość oprócz nadzoru przez personel musi być także powiązana z biurem(oferowana przez nią).

4. Model logiczny

4.1. Charakterystyka modelu relacyjnego oraz Usunięcie właściwości niekompatybilnych z modelem relacyjnym

Model relacyjny - w modelu tym dane grupowane są w relacje, które reprezentowane są przez tablice. Relacje są pewnym zbiorem rekordów o identycznej strukturze wewnętrznie powiązanych za pomocą związków zachodzących pomiędzy danymi.

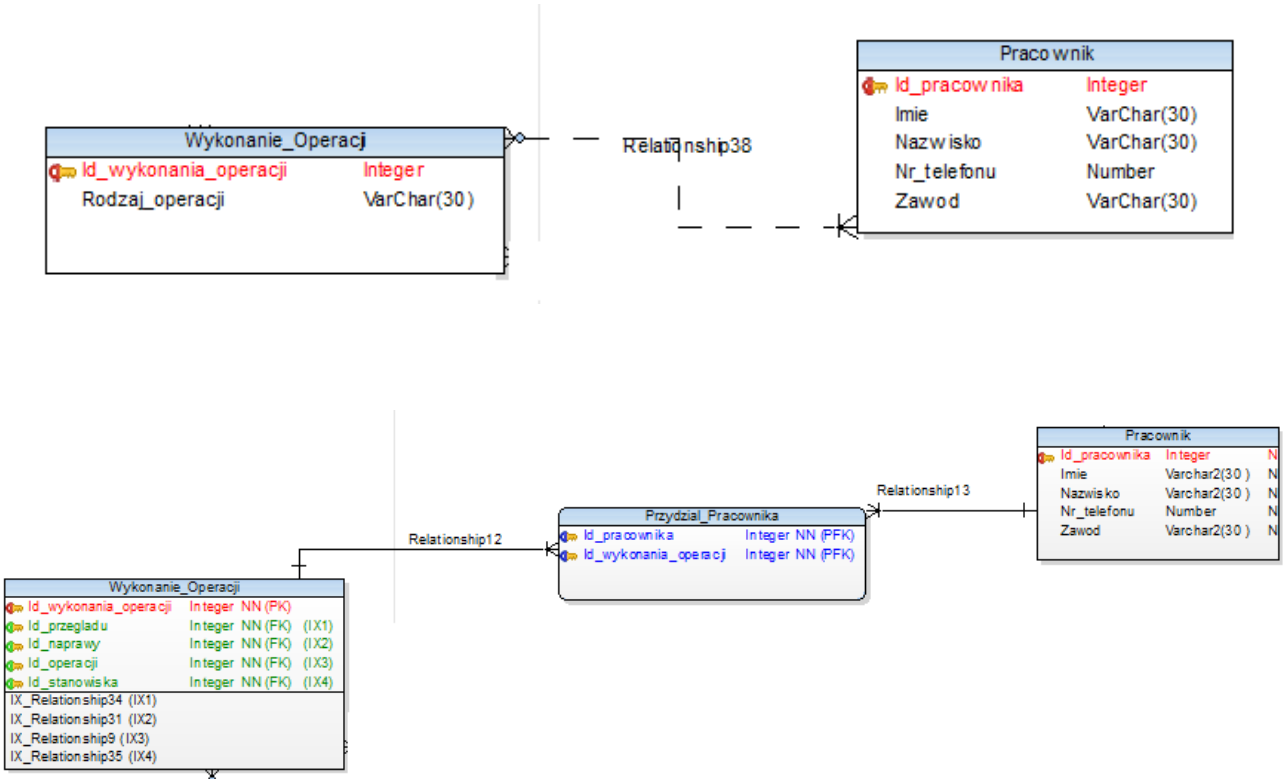
Model logiczny stworzono w programie Toad Data Modeler. Aby otrzymać model logiczny należało usunąć niekompatybilności z modelem relacyjnym. Usunięto związki wiele do wielu. Dodano w

tym celu niezbędne tabele bridge'ujące. Atrybutami tej encji są klucze obce relacji połączonych z nią w związek. Nie dodaje się do nich oddzielnego klucza głównego. Nazwy encji zostały zamienione na liczbę mnogą w celu odróżnienia relacji od encji. Mają one także ustalone klucze główne. Nazwy atrybutów nie potrzebowały zmian, ponieważ były unikalne w ramach encji już na poziomie konceptualnym. Schemat ER na poziomie konceptualnym nie zawierał pół wyliczalnych. Wszystkie atrybuty miały określone dziedziny.

W modelu relacyjnym nie występują atrybuty wielowartościowe. Jako przykład encji zastępującej atrybut wielowartościowy stworzyliśmy encje Grafik, dla każdego pracownika.

Związek 1:1 może być postrzegany jako podzielenie tabeli na dwie. Stosowany np. wtedy gdy zbiór dodatkowych atrybutów jest określony tylko dla wąskiego podzbioru wierszy w tabeli podstawowej albo wydzielenie pewnej grupy atrybutów które są rzadko odpytywane lub wydzielając je do osobnej tabeli możemy zapewnić dodatkowy poziom zabezpieczeń.

Poniżej przykład zastąpienia związku wiele do wielu z modelu konceptualnego poprzez wstawienie dodatkowej tabeli bridge'ującej:



4.2. Proces normalizacji i denormalizacji

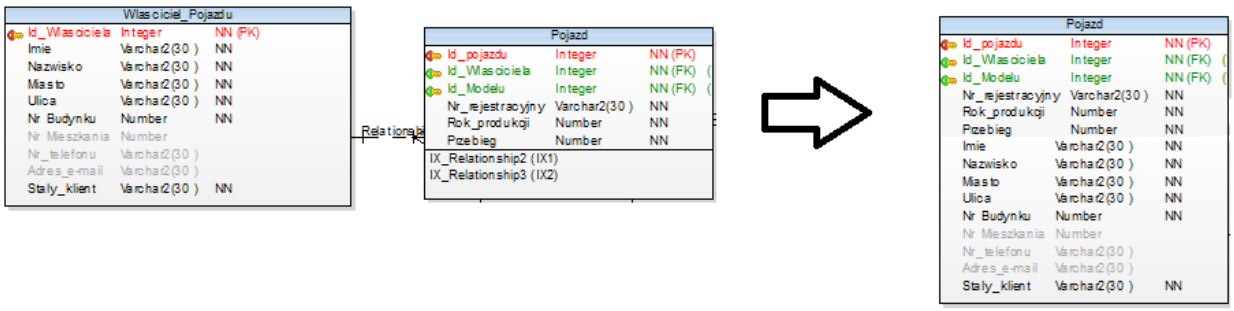
Normalizacja bazy danych jest to proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych. Główna idea polega na trzymaniu danych w jednym miejscu, a w razie potrzeby linkowania do danych. Taki sposób tworzenia bazy danych zwiększa bezpieczeństwo danych i zmniejsza ryzyko powstania niespójności (w szczególności problemów anomalii takich jak anomalia wstawiania, usuwania, modyfikacji). Normalizacja **nie usuwa danych**, tylko zmienia schemat bazy danych.

Cele normalizacji:
- Normalizacja ma na celu takie przekształcenie relacji, by uniknąć redundancji i anomalii
- Przekształcenie relacji do kolejnych postaci normalnych wiąże się często ze zmniejszeniem ilości pamięci potrzebnej do przechowywania informacji
- Unikanie powtórzeń pozwala na łatwiejszą i szybszą aktualizację danych
- Doprowadzenie bazy do wysokiej postaci normalnej może spowolnić odczyt w dużych bazach ze względu na skomplikowany schemat danych
- W większości przypadków po znormalizowaniu bazy danych przychodzi kolej na rozważanie możliwości wykonania odwrotnej operacji(a więc denormalizacji), polegającej na połączeniu niektórych znormalizowanych tabel, z myślą o przyspieszeniu dostępu do pewnych danych

Decyzją projektową jest wypracowanie kompromisu pomiędzy niepożądanymi anomaliami a wydajnością systemu. Wykonuje się to poprzez częściowa denormalizację bazy danych.

Denormalizacja oznacza zmianę (uszczegółowienia)schematu relacji, w efekcie której stopień normalizacji zmodyfikowanej relacji jest mniejszy od stopnia normalizacji co najmniej jednej z relacji oryginalnych.

Przykład decyzji projektowej:



W naszej bazie mamy dwie encje Właściciel_Pojazdu i Pojazd, przy denormalizacji a więc zrobieniu jednej encji gdzie będą zawarte atrybuty obu tych encji mamy sytuację gdzie przy wpisywaniu pojazdu do naprawy za każdym razem musimy wpisywać dane właściciela które są udostępnione wszystkim pracownikom warsztatu. Wprowadza to możliwą niespójność i nie zachowuje niezależności przechowywania danych. W postaci znormalizowanej (dwie encje połączone związkiem jeden do wielu) pracownicy znają dane tylko konkretnego pojazdu a nie właściciela.

Rodzaje anomalii jakie mogą wystąpić :

Anomalia usuwania – usuwanie części infomacji powoduje utratę informacji których nie chcielibyśmy stracić

Przykład: Gdybyśmy połączyli encje pojazd i właściciel to przy usuwaniu pojazdu zostałyby usunięte dane właściciela których nie chcemy usuwać

Anomalia dodawania – wprowadzenie pewnej informacji jest możliwe tylko wtedy, gdy jednocześnie wprowadzamy jakąś inną informację, która może być obecnie niedostępna

Przykład: Dla tego samego przykładu wpisanie do bazy danych właściciela bez pojazdu byłoby niemożliwe.

Anomalia aktualizacji - wartość występująca w wielu miejscach może ulec zmianie w jednym miejscu a w drugim nie co powoduje niespójność danych.

Przykład: Jeśli klient zmieni adres zamieszkzania który jest przechowywany w wielu relacjach może pojawić się niespójność. Po stronie aplikacji trzeba pamiętać aby zrobić zapytania modyfikujące we wszystkich miejscach.

Przykład denormalizacji:
Jako decyzję projektową przyjęliśmy że każdy pracownik wykonuje jeden typ operacji więc nie jest potrzebne osobnej encji Zawód. Można obie te encje połączyć w jedną, atrybut zawód w encji Pracownik. Nie wprowadza to anomalii, jest to skuteczna denormalizacja co skutkuje przyspieszeniu dostępu do danych.

4.3. Schemat ER na poziomie modelu logicznego

Schemet ER na poziomie modelu logicznego w załączniku

4.4. Więzy integrlaności

Więzami integralności są klucze główne.

PRIMARY KEY (= połączenie NOT NULL oraz UNIQUE) - Ograniczenie to zapewnia unikalność wartości w kolumnie, dlatego jest nałożone na klucze główne. W danej tabeli tylko jeden z atrybutów może być PRIMARY KEY. Wszystkie klucze główne muszą być unikalne.
FOREIGN KEY - Ograniczenie zapewniające integralność danych z obydwu tabel znajdujących się w relacji.
UNIQUE - Zapewnia unikalną wartość danego atrybutu dla każdej z krotek.
NOT NULL - Atrybuty z danym ograniczeniem nie mogą mieć wartości null
DEFAULT - Wprowadzanie do kolumn domyślnych wartości przy tworzeniu nowych krotek.
CHECK – ograniczenie wartości dopuszczalnych do zapisania jako wartość atrybutu

Przykład:

```
CREATE TABLE NAPRAWA
  (     ID_NAPRAWY NUMBER(*,0) NOT NULL ENABLE,
        ID_POJAZDU NUMBER(*,0) NOT NULL ENABLE,
        KOSZT NUMBER,
        CHECK(NUMBER >0)
)
```

5. Faza fizyczna

5.1. Projekt transakcji oraz przykłady zapytań i poleceń SQL odnoszących się do bazy danych

Transakcja – logiczna jednostka działania wykonywanego na bazie danych. Składa się z jednej lub wielu operacji dostępu do bazy danych np. operacje wstawiania, usuwania, pobierania i modyfikacji danych. Transakcja odpowiada spójnej operacji jak np. transakcja przelewu z konta na konto.

Scenariusze do implementacji:

1. Przyjęcie samochodu do serwisu (naprawa/przegląd), pokazanie wszystkich operacji do wykonania:

- użytkownik wpisuje w aplikacji swoje dane: Imię, Nazwisko,Adres itp.
- użytkownik wpisuje w aplikacji dane pojazdu który chce oddać do serwisu.
- użytkownik wybiera rodzaj usługi (przegląd/naprawa)
- użytkownik klika przycisk “zatwierdź”
- aplikacja zapisuje do bazy danych dane pojazdu oraz jego właściciela (o ile już takie dane nie są w bazie)

  INSERT INTO WLASCICIEL_POJAZDU (IMIE, NAZWISKO, ...) VALUES('Jan', 'Kowalski', …) INSERT INTO MODEL (NAZWA_MODELU, MARKA) VALUES ('Auris', 'Toyota') INSERT INTO POJAZD (NR_REJESTRACYJNY, ROK_PRODUKCJI, PRZEBIEG, ID_WLASCICIELA, ID_MODELU) VALUES(“XYZ345”, 1994, 35000, (SELECT ID_WLASCICIELA FROM WLASCICIEL_POJAZDU WHERE IMIE = “Jan” AND NAZWISKO = 'Kowalski'), (SELECT ID_MODELU FROM MODEL WHERE NAZWA_MODELU = 'Auris' AND MARKA = 'Toyota'))

- użytkownik klika przycisk “realizuj”
  1. aplikacja jeśli zostanie wybrana opcja Przegląd odwołuje się to tabeli z planem serwisowym dla danego pojazdu i wypisuje spis operacji które należy wykonać

  SELECT ID_OPERACJI, NAZWA FROM OPERACJA WHERE ID_OPERACJI IN(SELECT ID_OPERACJI FROM OPERACJA_SERWISOWA WHERE ID_POZYCJI_PLANU_SERWISOWEGO IN (SELECT ID_POZYCJI_PLANU_SERWISOWEGO FROM POZYCJA_PLANU_SERWISOWEGO WHERE ID_MODELU IN (SELECT ID_MODELU FROM "MODEL" WHERE NAZWA_MODELU='Toyota'

  - następnie wszystkie te operacje są przekazane do realizacji przez wpisanie do tabeli wykonanie operacji

  INSERT INTO WYKONANIE_OPERACJI (ID_WYKONANIA_OPERACJI, ID_PRZEGLADU, ID_OPERACJI, ID_STANOWISKA, RODZAJ_OPERACJI) VALUES (1,1,4,2,”Naprawa/Przegląd”);

2. aplikacja jeśli zostanie wybrana opcja Naprawa odwołuje się do tablicy operacje i wypisuje wszystkie operacje wykonywane w serwisie

SELECT ID_OPERACJI, NAZWA FROM OPERACJA

- użytkownik wybiera operację którą chce wykonać, klika przycisk “zrobione” i zostaje ona przekazana do realizacji przez wpisanie do tabeli wykonanie operacji

INSERT INTO WYKONANIE_OPERACJI (ID_WYKONANIA_OPERACJI, ID_PRZEGLADU, ID_OPERACJI, ID_STANOWISKA, RODZAJ_OPERACJI) VALUES (1,1,4,2,”Naprawa/Przegląd”);

Zaplanowanie wykonania operacji, przydział stanowiska, pracowników, części:

- użytkownik klika przycisk “Wyświetl”
- aplikacja zbiera operacje do wykonania z tablicy wykonanie operacji
  SELECT ID_WYKONANIA_OPERACJI FROM WYKONANIE_OPERACJI
- Aplikacja przydziela zasoby:
- Przydział pracownika
  INSERT INTO PRZYDZIAL_PRACOWNIKA VALUES ((SELECT ID_PRACOWNIKA FROM PRACOWNIK WHERE ZAWOD = (SELECT NAZWA FROM OPERACJA WHERE ID_OPERACJI = (SELECT ID_OPERACJI FROM WYKONANIE_OPERACJI WHERE ID_WYKONANIA_OPERACJI = 1))),1);

- Przydział stanowiska
  UPDATE WYKONANIE_OPERACJI SET ID_STANOWISKA = (SELECT ID_STANOWISKA FROM STANOWISKO WHERE ID_TYPU_STANOWISKA IN (SELECT ID_TYPU_STANOWISKA FROM OPERACJA WHERE ID_TYPU_STANOWISKA =WYKONANIE_OPERACJI.ID_OPERACJI)) WHERE ID_WYKONANIA_OPERACJI = 1;

- Przydział części
  INSERT INTO PRZYDZIAL_CZESCI VALUES (1,(SELECT ID_CZESCI FROM CZESC WHERE ID_OPERACJI = (SELECT ID_OPERACJI FROM WYKONANIE_OPERACJI WHERE ID_WYKONANIA_OPERACJI = 1)))

- Aplikacja wyświetla poniżej spisu operacji opis wykonywanych operacji, dane operacji, przydział pracowników, przydział części i stanowisko

Sposób 1

```
SELECT CZESC.NAZWA, STANOWISKO.NAZWA_STANOWISKA,
STANOWISKO.NR_STANOWISKA, PRACOWNIK.NAZWISKO FROM CZESC JOIN
PRACOWNIK ON ID_PRACOWNIKA IN (SELECT ID_PRACOWNIKA FROM
PRZYDZIAL_PRACOWNIKA WHERE ID_WYKONANIA_OPERACJI IN (SELECT
ID_WYKONANIA_OPERACJI FROM PRZYDZIAL_CZESCI WHERE ID_CZESCI IN
CZESC.ID_CZESCI)) JOIN STANOWISKO ON ID_STANOWISKA IN (SELECT
ID_STANOWISKA FROM WYKONANIE_OPERACJI WHERE
ID_WYKONANIA_OPERACJI IN (SELECT ID_WYKONANIA_OPERACJI FROM
PRZYDZIAL_CZESCI WHERE ID_CZESCI IN CZESC.ID_CZESCI));
```

Sposób 2

```
SELECT ID_WYKONANIA_OPERACJI, ID_OPERACJI, NAZWA_STANOWISKA,
NAZWISKO FROM WYKONANIE_OPERACJI JOIN STANOWISKO ON
NAZWA_STANOWISKA IN (SELECT NAZWA_STANOWISKA FROM STANOWISKO
WHERE ID_STANOWISKA = WYKONANIE_OPERACJI.ID_STANOWISKA) JOIN
PRACOWNIK ON NAZWISKO IN (SELECT NAZWISKO FROM PRACOWNIK
WHERE ID_PRACOWNIKA IN (SELECT ID_PRACOWNIKA FROM
PRZYDZIAL_PRACOWNIKA WHERE ID_WYKONANIA_OPERACJI =
WYKONANIE_OPERACJI.ID_WYKONANIA_OPERACJI));
```

Pozostałe scenariusze:

Zaplanowanie zamówienia części:

- użytkownik wpisuje nazwę części żeby sprawdzić jej stan
- aplikacja sprawdza stan wpisanej części

```
SELECT ILOSC_CZESCI FROM CZESC  WHERE NAZWA = 'śrubka' AND
NR_SERYJNY = 'ER235'
```

- aplikacja jeśli stan ilościowy części jest mniejszy niż 5 wypisuje komunikat “Nalezy zrobic zamowienie danej części”
- użytkownik naciskając przycisk “DOSTAWA” zleca zamówienie danej części
- aplikacja realizuje zamówienie sprawdzając u kogo zamówić dostawę

```
INSERT INTO DOSTAWA VALUES (1, (SELECT ID_DOSTAWCY FROM
DOSTEPNOSC_CZESCI WHERE ID_CZESCI = (SELECT ID_CZESCI FROM CZESC
WHERE NAZWA = 'srubka' AND    NR_SERYJNY = 'ER235')),1,Kurier,2016-12-30)
```

- aplikacja wyświetla na ekranie wszystkie dostawy

  SELECT * FROM DOSTAWA WHERE TERMIN_REALIZACJI> CURDATE()

Wyświetlenie zajętości pracowników:

- użytkownik naciska przycisk “lista pracowników”

  aplikacja wyświetla listę pracowników wykonujących operacje

  SELECT PRACOWNIK.IMIE, PRACOWNIK.NAZWISKO, OPERACJA.NAZWA FROM PRACOWNIK JOIN OPERACJA ON OPERACJA.ID_OPERACJI = (SELECT ID_OPERACJI FROM WYKONANIE_OPERACJI WHERE ID_WYKONANIA_OPERACJI = (SELECT ID_WYKONANIA_OPERACJI FROM PRZYDZIAL_PRACOWNIKA WHERE ID_PRACOWNIKA= PRACOWNIK.ID_PRACOWNIKA));

5.2. Strojenie bazy danych – dobór indeksów

Indeks – struktura bazy danych która służy do tego aby pobrać na wejściu pewną cechę rekordu, a następnie szybko znaleźć rekordy o tej samej własności. Składają się z rekordów a każdy rekord złożony jest z 2 pól - klucza i wskaźnika.

Indeksy warto stosować w przypadku, gdy:
- liczba bloków indeksu jest zbyt mała w porównaniu z liczbą bloków danych
- do wyszukiwania klucza można stosować przeszukiwanie binarne gdyż plik jest posortowany
- indeks może być na tyle mały że mieści się w pamięci operacyjnej. Jeżeli tak, nie trzeba wtedy wykonywać kosztownych operacji dostępu do dysku, tylko wymagany jest dostęp w obrębie tej pamięci.
- Zapytania są mało selektywne, zwracające małą ilość bloków

Wskazówki tworzenia indeksów:
- jeśli często występuje odwołanie do klucza obcego, warto założyć na niego indeks
- należy unikać indeksowania często modyfikowalnego atrybutu lub relacji
- należy unikać indeksowania atrybutów składających się z długich łańcuchów

Sprawdzenie użyteczności indeksów polega na sprawdzeniu selektywności dla konkretnych zapytań. Bez używania indeksu trzeba przejrzeć całą tabelę, SZBD musi pobrać wszystkie bloki logiczne do pamięci operacyjnej. Indeks daje nam to że będą pobrane tylko te bloki gdzie są szukane wiersze. Przy dużej selektywności zapytania jest prawdopodobieństwo że przy użyciu indeksu zostaną zwrócone wszystkie bloki i indeks nie przyspieszy dostęp do danych, więc nie ma sensu tworzenie indeksu. Trzeba przeanalizować każde zapytanie i wybrać optymalne rozwiązanie.

5.3. Skrypt SQL tworzący bazę danych

```
CREATE TABLE WLASCICIEL_POJAZDU(
  Id_Wlasciciela Integer NOT NULL,
  Imie Varchar2(30 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  Miasto Varchar2(30 ) NOT NULL,
```

```
  Ulica Varchar2(30 ) NOT NULL,
  Nr_Budynku Number NOT NULL,
  Nr_Mieszkania Number,
  Nr_telefonu Varchar2(30 ),
  Adres_e_mail Varchar2(30 ),
  Staly_klient Varchar2(30 ) NOT NULL
)
ALTER TABLE WLASCICIEL_POJAZDU ADD CONSTRAINT Key1 PRIMARY KEY
(Id_Wlasciciela)
CREATE TABLE POJAZD(
  Id_pojazdu Integer NOT NULL,
  Id_Wlasciciela Integer NOT NULL,
  Id_Modelu Integer NOT NULL,
  Nr_rejestracyjny Varchar2(30 ) NOT NULL,
  Rok_produkcji Number NOT NULL,
  Przebieg Number NOT NULL
)
CREATE INDEX IX_Relationship2 ON POJAZD (Id_Wlasciciela)
CREATE INDEX IX_Relationship3 ON POJAZD (Id_Modelu)
ALTER TABLE POJAZD ADD CONSTRAINT Key2 PRIMARY KEY (Id_pojazdu)
CREATE TABLE MODEL(
  Id_Modelu Integer NOT NULL,
  Nazwa_modelu Varchar2(30 ) NOT NULL,
  Marka Varchar2(30 ) NOT NULL
)
ALTER TABLE MODEL ADD CONSTRAINT Key4 PRIMARY KEY (Id_Modelu)
CREATE TABLE POZYCJA_PLANU_SERWISOWEGO(
  Id_pozycji_planu_serwisowego Integer NOT NULL,
  Id_Modelu Integer NOT NULL,
  Rodzaj_serwisowania Varchar2(30 ) NOT NULL
)
CREATE INDEX IX_Relationship11 ON POZYCJA_PLANU_SERWISOWEGO (Id_Modelu)
ALTER TABLE POZYCJA_PLANU_SERWISOWEGO ADD CONSTRAINT Key5 PRIMARY KEY
(Id_pozycji_planu_serwisowego)
CREATE TABLE NAPRAWA(
  Id_naprawy Integer NOT NULL,
  Id_pojazdu Integer NOT NULL,
  Koszt Integer NOT NULL
)
CREATE INDEX IX_Relationship30 ON NAPRAWA (Id_pojazdu)
ALTER TABLE NAPRAWA ADD CONSTRAINT Key6 PRIMARY KEY (Id_naprawy)
CREATE TABLE PRZEGLAD(
  Id_przegladu Integer NOT NULL,
  Id_pojazdu Integer NOT NULL,
  Ilosc_przegladow Integer NOT NULL
)
CREATE INDEX IX_Relationship33 ON PRZEGLAD (Id_pojazdu)
ALTER TABLE PRZEGLAD ADD CONSTRAINT Key7 PRIMARY KEY (Id_przegladu)
CREATE TABLE WYKONANIE_OPERACJI(
  Id_wykonania_operacji Integer NOT NULL,
  Id_przegladu Integer NOT NULL,
  Id_naprawy Integer NOT NULL,
  Id_operacji Integer NOT NULL,
  Id_stanowiska Integer NOT NULL
```

```
)
CREATE INDEX IX_Relationship34 ON WYKONANIE_OPERACJI (Id_przegladu)
CREATE INDEX IX_Relationship31 ON WYKONANIE_OPERACJI (Id_naprawy)
CREATE INDEX IX_Relationship9 ON WYKONANIE_OPERACJI (Id_operacji)
CREATE INDEX IX_Relationship35 ON WYKONANIE_OPERACJI (Id_stanowiska)
ALTER TABLE WYKONANIE_OPERACJI ADD CONSTRAINT Key8 PRIMARY KEY
(Id_wykonania_operacji)
CREATE TABLE PRZYDZIAL_PRACOWNIKA(
  Id_pracownika Integer NOT NULL,
  Id_wykonania_operacji Integer NOT NULL
)
ALTER TABLE PRZYDZIAL_PRACOWNIKA ADD CONSTRAINT Key9 PRIMARY KEY
(Id_pracownika,Id_wykonania_operacji)
CREATE TABLE PRACOWNIK(
  Id_pracownika Integer NOT NULL,
  Imie Varchar2(30 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  Nr_telefonu Number NOT NULL,
  Zawod Varchar2(30 ) NOT NULL
)
ALTER TABLE PRACOWNIK ADD CONSTRAINT Id_Pracownika PRIMARY KEY
(Id_pracownika)
CREATE TABLE DOSTEPNOSC_CZESCI(
  Id_dostawcy Integer NOT NULL,
  Id_czesci Integer NOT NULL
)
ALTER TABLE DOSTEPNOSC_CZESCI ADD CONSTRAINT Key11 PRIMARY KEY
(Id_dostawcy,Id_czesci)
CREATE TABLE ZAMOWIONA_CZESC(
  Id_dostawy Integer NOT NULL,
  Id_czesci Integer NOT NULL
)
ALTER TABLE ZAMOWIONA_CZESC ADD CONSTRAINT Key12 PRIMARY KEY
(Id_dostawy,Id_czesci)
CREATE TABLE STANOWISKO(
  Id_stanowiska Integer NOT NULL,
  Id_typu_stanowiska Integer NOT NULL,
  Nr_stanowiska Number NOT NULL,
  Nazwa_stanowiska Varchar2(30 ) NOT NULL
)
CREATE INDEX IX_Relationship16 ON STANOWISKO (Id_typu_stanowiska)
ALTER TABLE STANOWISKO ADD CONSTRAINT Key13 PRIMARY KEY (Id_stanowiska)
CREATE TABLE TYP_STANOWISKA(
  Id_typu_stanowiska Integer NOT NULL,
  Nazwa_typu Number NOT NULL
)
ALTER TABLE TYP_STANOWISKA ADD CONSTRAINT Key14 PRIMARY KEY
(Id_typu_stanowiska)
CREATE TABLE PRZYDZIAL_CZESCI(
  Id_wykonania_operacji Integer NOT NULL,
  Id_czesci Integer NOT NULL
)
ALTER TABLE PRZYDZIAL_CZESCI ADD CONSTRAINT Key15 PRIMARY KEY
(Id_wykonania_operacji,Id_czesci)
```

```
CREATE TABLE OPERACJA(
  Id_operacji Integer NOT NULL,
  Id_typu_stanowiska Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL,
  Czas_trwania Number NOT NULL
)
CREATE INDEX IX_Relationship18 ON OPERACJA (Id_typu_stanowiska)
ALTER TABLE OPERACJA ADD CONSTRAINT Key16 PRIMARY KEY (Id_operacji)
CREATE TABLE CZESC(
  Id_czesci Integer NOT NULL,
  Id_operacji Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL,
  Nr_seryjny Varchar2(30 ) NOT NULL,
  Ilosc_czesci Number NOT NULL
)
CREATE INDEX IX_Relationship23 ON CZESC (Id_operacji)
ALTER TABLE CZESC ADD CONSTRAINT Key17 PRIMARY KEY (Id_czesci)
CREATE TABLE TYP_URZADZENIA(
  Id_typu_urzadzenia Integer NOT NULL,
  Id_operacji Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL
)
CREATE INDEX IX_Relationship21 ON TYP_URZADZENIA (Id_operacji)
ALTER TABLE TYP_URZADZENIA ADD CONSTRAINT Key18 PRIMARY KEY
(Id_typu_urzadzenia)
CREATE TABLE OPERACJA_SERWISOWA(
  Id_pozycji_planu_serwisowego Integer NOT NULL,
  Id_operacji Integer NOT NULL
)
ALTER TABLE OPERACJA_SERWISOWA ADD CONSTRAINT Key19 PRIMARY KEY
(Id_pozycji_planu_serwisowego,Id_operacji)
CREATE TABLE POZYCJA_GRAFIKU(
  Id_pozycji_grafiku Integer NOT NULL,
  Id_pracownika Integer NOT NULL,
  Poniedzialek Number,
  Wtorek Number,
  Sroda Number,
  Czwartek Number,
  Piatek Number,
  Sobota Number
)
CREATE INDEX IX_Relationship14 ON POZYCJA_GRAFIKU (Id_pracownika)
ALTER TABLE POZYCJA_GRAFIKU ADD CONSTRAINT Key20 PRIMARY KEY
(Id_pozycji_grafiku)
CREATE TABLE POZYCJA_WYPOSAZENIA(
  Id_pozycji_wyposazenia Integer NOT NULL,
  Id_typu_stanowiska Integer NOT NULL,
  Stan Varchar2(30 ) NOT NULL
)
CREATE INDEX IX_Relationship17 ON POZYCJA_WYPOSAZENIA (Id_typu_stanowiska)
ALTER TABLE POZYCJA_WYPOSAZENIA ADD CONSTRAINT Key21 PRIMARY KEY
(Id_pozycji_wyposazenia)
CREATE TABLE DOSTAWCA(
  Id_dostawcy Integer NOT NULL,
```

```
  Nazwa_dostawcy Varchar2(30 ) NOT NULL,
  Nr_telefonu Varchar2(30 ),
  E_mail Varchar2(30 )
)
ALTER TABLE DOSTAWCA ADD CONSTRAINT Key22 PRIMARY KEY (Id_dostawcy)
CREATE TABLE URZADZENIE(
  Id_urzadzenia Integer NOT NULL,
  Id_typu_urzadzenia Integer NOT NULL,
  Nazwa_urzadzenia Varchar2(30 ) NOT NULL,
  Nr_seryjny_urzadzenia Varchar2(30 ) NOT NULL
)
CREATE INDEX IX_Relationship22 ON URZADZENIE (Id_typu_urzadzenia)
ALTER TABLE URZADZENIE ADD CONSTRAINT Key23 PRIMARY KEY (Id_urzadzenia)
CREATE TABLE DOSTAWA(
  Id_dostawy Integer NOT NULL,
  Id_dostawcy Integer NOT NULL,
  Numer_dostawy Number NOT NULL,
  Sposob_dostawy Varchar2(30 ) NOT NULL,
  Termin_realizacji Date NOT NULL
)
CREATE INDEX IX_Relationship26 ON DOSTAWA (Id_dostawcy)
ALTER TABLE DOSTAWA ADD CONSTRAINT Key24 PRIMARY KEY (Id_dostawy)
ALTER TABLE POJAZD ADD CONSTRAINT Relationship2 FOREIGN KEY (Id_Wlasciciela)
REFERENCES WLASCICIEL_POJAZDU (Id_Wlasciciela)
ALTER TABLE POJAZD ADD CONSTRAINT Relationship3 FOREIGN KEY (Id_Modelu)
REFERENCES MODEL (Id_Modelu)
ALTER TABLE WYKONANIE_OPERACJI ADD CONSTRAINT Relationship9 FOREIGN KEY
(Id_operacji) REFERENCES OPERACJA (Id_operacji)
ALTER TABLE PRZYDZIAL_CZESCI ADD CONSTRAINT Relationship10 FOREIGN KEY
(Id_wykonania_operacji) REFERENCES WYKONANIE_OPERACJI (Id_wykonania_operacji)
ALTER TABLE POZYCJA_PLANU_SERWISOWEGO ADD CONSTRAINT Relationship11
FOREIGN KEY (Id_Modelu) REFERENCES MODEL (Id_Modelu)
ALTER TABLE PRZYDZIAL_PRACOWNIKA ADD CONSTRAINT Relationship12 FOREIGN KEY
(Id_wykonania_operacji) REFERENCES WYKONANIE_OPERACJI (Id_wykonania_operacji)
ALTER TABLE PRZYDZIAL_PRACOWNIKA ADD CONSTRAINT Relationship13 FOREIGN KEY
(Id_pracownika) REFERENCES PRACOWNIK (Id_pracownika)
ALTER TABLE POZYCJA_GRAFIKU ADD CONSTRAINT Relationship14 FOREIGN KEY
(Id_pracownika) REFERENCES PRACOWNIK (Id_pracownika)
ALTER TABLE STANOWISKO ADD CONSTRAINT Relationship16 FOREIGN KEY
(Id_typu_stanowiska) REFERENCES TYP_STANOWISKA (Id_typu_stanowiska)
ALTER TABLE POZYCJA_WYPOSAZENIA ADD CONSTRAINT Relationship17 FOREIGN KEY
(Id_typu_stanowiska) REFERENCES TYP_STANOWISKA (Id_typu_stanowiska)
ALTER TABLE OPERACJA ADD CONSTRAINT Relationship18 FOREIGN KEY
(Id_typu_stanowiska) REFERENCES TYP_STANOWISKA (Id_typu_stanowiska)
ALTER TABLE OPERACJA_SERWISOWA ADD CONSTRAINT Relationship19 FOREIGN KEY
(Id_operacji) REFERENCES OPERACJA (Id_operacji)
ALTER TABLE OPERACJA_SERWISOWA ADD CONSTRAINT Relationship20 FOREIGN KEY
(Id_pozycji_planu_serwisowego) REFERENCES POZYCJA_PLANU_SERWISOWEGO
(Id_pozycji_planu_serwisowego)
ALTER TABLE TYP_URZADZENIA ADD CONSTRAINT Relationship21 FOREIGN KEY
(Id_operacji) REFERENCES OPERACJA (Id_operacji)
ALTER TABLE URZADZENIE ADD CONSTRAINT Relationship22 FOREIGN KEY
(Id_typu_urzadzenia) REFERENCES TYP_URZADZENIA (Id_typu_urzadzenia)
ALTER TABLE CZESC ADD CONSTRAINT Relationship23 FOREIGN KEY (Id_operacji)
```

REFERENCES OPERACJA (Id_operacji)
ALTER TABLE DOSTEPNOSC_CZESCI ADD CONSTRAINT Relationship24 FOREIGN KEY (Id_czesci) REFERENCES CZESC (Id_czesci)
ALTER TABLE DOSTEPNOSC_CZESCI ADD CONSTRAINT Relationship25 FOREIGN KEY (Id_dostawcy) REFERENCES DOSTAWCA (Id_dostawcy)
ALTER TABLE DOSTAWA ADD CONSTRAINT Relationship26 FOREIGN KEY (Id_dostawcy) REFERENCES DOSTAWCA (Id_dostawcy)
ALTER TABLE ZAMOWIONA_CZESC ADD CONSTRAINT Relationship27 FOREIGN KEY (Id_czesci) REFERENCES CZESC (Id_czesci)
ALTER TABLE ZAMOWIONA_CZESC ADD CONSTRAINT Relationship28 FOREIGN KEY (Id_dostawy) REFERENCES DOSTAWA (Id_dostawy)
ALTER TABLE PRZYDZIAL_CZESCI ADD CONSTRAINT Relationship29 FOREIGN KEY (Id_czesci) REFERENCES CZESC (Id_czesci)
ALTER TABLE NAPRAWA ADD CONSTRAINT Relationship30 FOREIGN KEY (Id_pojazdu) REFERENCES POJAZD (Id_pojazdu)
ALTER TABLE WYKONANIE_OPERACJI ADD CONSTRAINT Relationship31 FOREIGN KEY (Id_naprawy) REFERENCES NAPRAWA (Id_naprawy)
ALTER TABLE PRZEGLAD ADD CONSTRAINT Relationship33 FOREIGN KEY (Id_pojazdu) REFERENCES POJAZD (Id_pojazdu)
ALTER TABLE WYKONANIE_OPERACJI ADD CONSTRAINT Relationship34 FOREIGN KEY (Id_przegladu) REFERENCES PRZEGLAD (Id_przegladu)
ALTER TABLE WYKONANIE_OPERACJI ADD CONSTRAINT Relationship35 FOREIGN KEY (Id_stanowiska) REFERENCES STANOWISKO (Id_stanowiska)