**Kolegium Nauk Przyrodniczych  
Uniwersytet Rzeszowski**

**Przedmiot:**

**Bazy Danych 2**

**Projekt Komis Samochodowy**

**Wykonał:**

**Kacper Dusza, 131427**

**Radosław Cebula,**

**Prowadzący: dr. Piotr Grochowalski**

**Rzeszów 2025**

1. **Wstęp**

Projekt zakłada stworzenie bazy danych dla przedsiębiorstwa zajmującego się sprzedażą pojazdów oraz zarządzaniem placówkami i podwykonawcami. Celem jest efektywna organizacja danych związanych z klientami, zamówieniami, pracownikami, placówkami oraz pojazdami. Baza danych została stworzona w oparciu o PostgreSQL. Aby ułatwić dostęp do danych, wdrożono aplikację internetową w technologii PHP, z intuicyjnym interfejsem graficznym. Aplikacja umożliwia przeglądanie, edytowanie, dodawanie i usuwanie danych w tabelach.

Obraz zawierający tekst, diagram, Równolegle, linia

Opis wygenerowany automatycznie

1. **Specyfikacja tematu projektu**

Projektowana rzeczywistość obejmuje działalność firmy motoryzacyjnej, która:

* Zarządza placówkami, w których przechowywane są pojazdy i pracownicy.
* Współpracuje z podwykonawcami odpowiedzialnymi za naprawy i pojazdów.
* Obsługuje klientów indywidualnych i biznesowych.
* Realizuje zamówienia na zakup i ewentualną naprawę pojazdu.
* Zarządza pracownikami oraz ich stanowiskami
* Przechowuje informacje o klientach i ich zamówieniach.

1. **Aspekt projektowy bazy danych**

Baza danych została zaprojektowana z podziałem na następujące schematy:

* **Klienci:** dane osobowe i kontaktowe klientów.
* **Zamówienia:** szczegóły dotyczące zamówień realizowanych przez firmę.
* **Placówki:** informacje o placówkach firmy, w tym liczba miejsc i liczba pojazdów.
* **Podwykonawcy:** informacje o firmach współpracujących z przedsiębiorstwem.
* **Pojazdy:** szczegółowe dane o pojazdach w magazynie oraz w serwisie.
* **Firmy:** informacje o przedsiębiorstwach klientów.
* **Stanowiska:** szczegóły o stanowiskach pracowników firmy.
* **Pracownicy:** informacje o pracownikach przedsiębiorstwa.

W każdym schemacie zdefiniowano odpowiednie tabele, relacje, funkcje, procedury i triggery. Takie podejście zapewnia przejrzystość i optymalizację działania bazy danych.

1. **Aspekt projektowy funkcjonalności bazy danych**
2. *Schematy*

Baza danych została podzielona na schematy. W każdym z nich znajdują się funkcje, triggery oraz procedury, które to są związane z daną tabelą. Dzięki takiemu rozwiązaniu baza jest o wiele bardziej przejrzysta i łatwiej się z niej korzysta.

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

1. *Procedury*

Każda z tabel posiada odrębne procedury które pozwalają na dodawanie, usuwanie i edytowanie danych w tabeli.

*Przykład dla tabeli „firmy”*

*Obraz zawierający zrzut ekranu, Czcionka, tekst, czarne

Opis wygenerowany automatycznie*

*Dodaj:*

BEGIN

INSERT INTO firmy (nazwaFirmy, nip)

VALUES (p\_nazwaFirmy, p\_nip);

END;

*Edytuj*:

BEGIN

UPDATE firmy

SET nazwaFirmy = p\_nazwaFirmy, nip = p\_nip

WHERE nazwaFirmy = p\_nazwaFirmyDoZmiany;

END;

*Usuń:*

BEGIN

DELETE FROM firmy

WHERE nazwaFirmy = p\_nazwaFirmy;

END;

1. *Triggery*

Dla tabeli „placówki” zaimplementowano 5 triggerów. Przy ich tworzeniu zostały użyte sekwencje.

* *inkrementacjaLiczbyAut*() – w momencie gdy do danej placówki zostaje dodany samochód, wartość w kolumnie „liczbaSamochodow” zwiększana jest o jeden.

BEGIN

UPDATE placowki

SET iloscsamochodow = iloscsamochodow+1

WHERE id\_placowki = NEW.placowkaid;

RETURN NEW;

END;

* *dekrementacjaLiczbyAut*() – w momencie gdy z danej placówki usuwany jest samochód, wartość w kolumnie „liczbaSamochodow” zmniejszana jest o jeden.

BEGIN

UPDATE placowki

SET iloscsamochodow = iloscsamochodow -1

WHERE id\_placowki = OLD.placowkaid;

RETURN OLD;

END;

* *inkrementacjaMiejscaPozostałego*() – w momencie gdy z danej placówki usuwany jest samochód, wartość w kolumnie „iloscmiejscapozostalego” zostaje zwiększona o jeden.

BEGIN

UPDATE placowki

SET iloscmiejscapozostalego = iloscmiejscapozostalego - 1

WHERE id\_placowki = NEW.placowkaid;

RETURN NEW;

END;

* *dekrementacjaMiejscaPozostałego*() – w momencie gdy do danej placówki dodawany jest samochów, wartość w kolumnie „iloscmiejscapozostalego” jest zmniejszana o jeden.

*BEGIN*

*UPDATE placowki*

*SET iloscmiejscapozostalego = iloscmiejscapozostalego + 1*

*WHERE id\_placowki = OLD.placowkaid;*

*RETURN OLD;*

*END;*

* check\_*iloscMiejscaPozostalego*() – w momencie gdy będziemy chcieli dodać samochód do placówki w której wartość kolumny *iloscMiejscaPozostałego* będzie wynosić 0, trigger ten na to nie pozwoli.

BEGIN

IF (SELECT iloscmiejscapozostalego FROM placowki WHERE id\_placowki = NEW.placowkaid) = 0 THEN

RAISE EXCEPTION 'Nie można dodać samochodu: brak wolnych miejsc w placówce';

END IF;

RETURN NEW;

END;

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

# *Funkcje*

* Funkcje pobierające – każda z tabel posiada oddzielną funkcje, która służy do pobierania danych. Funkcja wykorzystywana jest to pobrania danych na interfejs graficzny.

*Przykład dla tabeli „klienci”*

BEGIN

RETURN QUERY

SELECT k.pesel,

k.imie,

k.nazwisko,

k.nip,

k.telefon

FROM klienci k;

END;

* historia\_podwykonawcy(p\_nip VARCHAR) – funkcja która zwraca zapytanie które wyświetla informacje na temat wszystkich zamówień danego podwykonawcy.

BEGIN

RETURN QUERY

SELECT

sp.vin,

sp.nippodwykonawcy

FROM samochody\_podwykonawcy as sp

WHERE sp.nippodwykonawcy = p\_nip;

END;

* oblicz\_statystyki\_podwykonawcy(nippodwykonawcy\_param CHAR) – funkcja ta zwraca zapytanie które wyświetla sumę wartości wszystkich samochodów nad którymi dany podwykonawca pracował oraz w nowej kolumnie wyświetla uśrednioną wartość samochodu. Takie rozwiązanie może dostarczyć przedsiębiorstwu niezwykle cennych informacji.

BEGIN

RETURN QUERY

SELECT

sp.nippodwykonawcy,

SUM(s.cena) AS suma\_wartosci\_aut,

SUM(s.cena) / COUNT(sp.vin) AS srednia\_wartosc\_auta

FROM samochody\_podwykonawcy sp

JOIN samochody s ON sp.vin = s.vin

WHERE sp.nippodwykonawcy = nippodwykonawcy\_param

GROUP BY sp.nippodwykonawcy;

END;

* ilość\_pracownikow(p\_placowkaid INT) – funkcja zwraca zapytanie, które wyświetla ilu pracowników jest przypisanych do danej placówki

DECLARE

ilosc NUMERIC := 0;

BEGIN

IF p\_placowkaid IS NOT NULL THEN

SELECT COUNT(stanowiskoid) INTO ilosc

FROM pracownicy

WHERE placowkaid = p\_placowkaid;

ELSE

SELECT COUNT(stanowiskoid) INTO ilosc

FROM pracownicy;

END IF;

RETURN COALESCE(ilosc, 0);

END;

* suma\_kwoty\_zamowien\_usluga\_naprawy() – funkcja zwraca wartość wszystkich zamówień, dla których zlecono usługę naprawy pojazdu

DECLARE

r RECORD;

suma double precision := 0;

cur CURSOR FOR

SELECT Kwota

FROM zamowienia

WHERE usluganaprawy = true;

BEGIN

OPEN cur;

LOOP

FETCH cur INTO r;

EXIT WHEN NOT FOUND;

suma := suma + r.Kwota;

END LOOP;

CLOSE cur;

RETURN suma;

END;

* zamowieina\_danego\_klienta(pesel CHAR) – funkcja zwraca zapytanie z informacjami na temat każdego zamówienia danego klienta

BEGIN

RETURN QUERY

SELECT

z.id\_zamowienia,

z.kwota,

z.pesel\_klienta,

z.vin,

z.usluganaprawy,

z.pesel\_pracownika

FROM zamowienia AS z

WHERE z.pesel\_klienta = pesel;

END;

* sumazarobkow(p\_placowkaid INT, DEFAULT NULL) – funkcja która zwraca sumę zarobków wszystkich pracowników w danej placówce. Jeżeli funkcja zostanie wywołana bez parametru, wówczas zwrócona zostanie informacja na temat zarobków wszystkich pracowników w całej firmie

DECLARE

suma NUMERIC := 0;

BEGIN

IF p\_placowkaid IS NOT NULL THEN

SELECT SUM(zarobki) INTO suma

FROM pracownicy

WHERE placowkaid = p\_placowkaid;

ELSE

SELECT SUM(zarobki) INTO suma

FROM pracownicy;

END IF;

RETURN COALESCE(suma, 0);

END;