Spis treści

[**1. Wprowadzenie i Opis Problemu** 3](#_Toc199869984)

[**2. Koncepcja Bazy Danych** 3](#_Toc199869985)

[**2.1. Opis Encji** 4](#_Toc199869986)

[**2.2. Diagram ERD** 6](#_Toc199869987)

[**2.3. Relacje Między Encjami** 6](#_Toc199869988)

[**3. Koncepcja Funkcjonalności w PL/SQL** 7](#_Toc199869989)

[**4. Opis Operacji CRUD (na przykładzie tabeli KLIENT)** 8](#_Toc199869990)

[**4.1. Tworzenie (Create) - PROCEDURE KLIENTCREATE** 8](#_Toc199869991)

[**4.2. Odczyt (Read)** 8](#_Toc199869992)

[**4.3. Aktualizacja (Update) - PROCEDURE KLIENTUPDATE** 9](#_Toc199869993)

[**4.4. Usuwanie (Delete/Dezaktywacja) - PROCEDURE KLIENTDELETE** 9](#_Toc199869994)

[**5. Szczegółowy Opis Wytworzonych Funkcjonalności PL/SQL** 9](#_Toc199869995)

[**5.1 FUNCTION STATISTICPOPULARCITY** 9](#_Toc199869996)

[**5.2 FUNCTION STATISTICBESTWORKER** 10](#_Toc199869997)

[**5.3 FUNCTION STATISTICBESTCLIENTS** 11](#_Toc199869998)

[**5.4 FUNCTION STATISTICAVERAGETRANSACTION** 12](#_Toc199869999)

[**5.5 FUNCTION STATISTICPRODUCT** 13](#_Toc199870000)

[**5.6 TRIGGER SOFTDELETE** 14](#_Toc199870001)

[**5.7 TRIGGER INSERT** 14](#_Toc199870002)

[**5.8 TRIGGER UPDATE** 15](#_Toc199870003)

[**6. Uporządkowanie projektu** 16](#_Toc199870004)

[**7. Prezentacja Działania Aplikacji (Interfejsu) w Kontekście Wykorzystania PL/SQL** 17](#_Toc199870005)

[**7.1. Zarządzanie Klientami** 17](#_Toc199870006)

[**7.2. Zarządzanie Asortymentem (Produkty, Amunicja, Kategorie)** 21](#_Toc199870007)

[**7.3. Realizacja Transakcji Sprzedaży** 22](#_Toc199870008)

[**7.4. Zarządzanie Pracownikami** 25](#_Toc199870009)

[**7.5. Dostęp do Statystyk i Raportów** 26](#_Toc199870010)

# **1. Wprowadzenie i Opis Problemu**

Projekt dotyczy stworzenia systemu bazodanowego dla sklepu specjalizującego się w sprzedaży broni i amunicji. Głównym celem systemu jest efektywne zarządzanie kluczowymi aspektami działalności takiego sklepu, z uwzględnieniem specyficznych wymagań prawnych i operacyjnych, takich jak weryfikacja pozwoleń na broń, śledzenie stanów magazynowych oraz kontrola dostępu pracowników.

Zdefiniowane główne problemy i potrzeby, które system ma adresować, to:

* **Zarządzanie asortymentem:** Sklep oferuje różnorodne produkty (broń, amunicja), które muszą być skatalogowane, przypisane do kategorii i odpowiednio powiązane (np. broń z kompatybilną amunicją). System umożliwia katalogowanie produktów według kategorii oraz definiowanie powiązań między bronią a dedykowaną dla niej amunicją.
* **Weryfikacja uprawnień klientów:** Zakup broni i niektórych rodzajów amunicji jest ściśle regulowany. System musi umożliwiać weryfikację posiadanych przez klientów pozwoleń, ich typów oraz dat ważności, aby zapewnić zgodność z obowiązującymi przepisami przy zakupie konkretnych typów broni.
* **Obsługa transakcji:** Proces sprzedaży musi być dokładnie rejestrowany, włączając w to identyfikację klienta, sprzedającego pracownika, zakupione produkty (broń oraz amunicję) oraz ich ilości. System powinien również kalkulować całkowitą wartość zamówienia.
* **Kontrola dostępu pracowników:** System musi zapewniać różne poziomy dostępu dla pracowników, w zależności od ich roli i uprawnień (np. do sprzedaży określonych kategorii broni), a także monitorować aktywność ich kont.
* **Kategoryzacja produktów:** Organizacja asortymentu w logiczne kategorie (np. broń sportowa, myśliwska, kolekcjonerska) jest kluczowa dla łatwego zarządzania i wyszukiwania. Każda kategoria może mieć zdefiniowane minimalne wymagane uprawnienia do zakupu.
* **Zarządzanie stanami magazynowymi:** Kluczowe jest śledzenie dostępności poszczególnych egzemplarzy broni (które są unikalne i sprzedawane pojedynczo) oraz ilości dostępnej amunicji, aby unikać sprzedaży towaru, którego nie ma na stanie, oraz planować zamówienia.
* **Analiza i raportowanie:** Możliwość generowania statystyk dotyczących sprzedaży, popularności produktów, aktywności klientów czy efektywności pracowników jest istotna dla podejmowania decyzji biznesowych.

Rozwiązaniem tych problemów jest zaprojektowanie i implementacja relacyjnej bazy danych oraz zestawu procedur i funkcji składowanych (w PL/SQL), które zautomatyzują i usprawnią wymienione procesy, zapewniając jednocześnie integralność danych i bezpieczeństwo.

# **2. Koncepcja Bazy Danych**

Koncepcja bazy danych opiera się na zidentyfikowaniu kluczowych encji (obiektów) biorących udział w procesach biznesowych sklepu oraz relacji zachodzących między nimi. Celem jest stworzenie struktury, która będzie logicznie odwzorowywać rzeczywistość operacyjną sklepu, umożliwiając efektywne przechowywanie i zarządzanie danymi.

## **2.1. Opis Encji**

* **ADRES:** Przechowuje szczegółowe dane adresowe.
  + ADRESID (PK): NUMBER(\*) - Unikalny identyfikator adresu.
  + ULICA: VARCHAR2(100) - Nazwa ulicy.
  + NUMERDOMU: VARCHAR2(10) - Numer domu.
  + NUMERMIESZKANIA: VARCHAR2(10) - Numer mieszkania (opcjonalny).
  + KODPOCZTOWY: VARCHAR2(10) - Kod pocztowy.
  + MIEJSCOWOSC: VARCHAR2(100) - Nazwa miejscowości.
* **KLIENT:** Reprezentuje osobę dokonującą zakupów.
  + KLIENTID (PK): NUMBER(\*) - Unikalny identyfikator klienta.
  + IMIE: VARCHAR2(50) - Imię klienta.
  + NAZWISKO: VARCHAR2(50) - Nazwisko klienta.
  + NUMERPOZWOLENIA: VARCHAR2(50) - Numer pozwolenia na broń.
  + TYPPOZWOLENIA: VARCHAR2(50) - Typ posiadanego pozwolenia (np. 'A', 'B', 'C').
  + DATAWAZNOSCIPOZWOLENIA: DATE - Data ważności pozwolenia.
  + ADRESID (FK): NUMBER(\*) - Klucz obcy do ADRES.
  + EMAIL: VARCHAR2(100) - Adres e-mail klienta.
  + TELEFON: VARCHAR2(20) - Numer telefonu klienta.
  + STATUSAKTYWNOSCI: CHAR(1) - Status konta ('1' - aktywny, '0' - nieaktywny).
* **PRACOWNIK:** Reprezentuje osobę zatrudnioną w sklepie.
  + PRACOWNIKID (PK): NUMBER(\*) - Unikalny identyfikator pracownika.
  + IMIE: VARCHAR2(50) - Imię pracownika.
  + NAZWISKO: VARCHAR2(50) - Nazwisko pracownika.
  + STANOWISKO: VARCHAR2(50) - Stanowisko pracownika.
  + ADRESID (FK): NUMBER(\*) - Klucz obcy do ADRES.
  + POZIOMUPRAWNIEN: VARCHAR2(20) - Poziom uprawnień pracownika w systemie (np. 'A', 'B', 'C', powiązany z WYMAGANEUPRAWNIENIA w KATEGORIAPRODUKTOW).
  + LOGIN: VARCHAR2(100) - Login pracownika.
  + HASLO: VARCHAR2(255) - Hasło pracownika (hash).
  + STATUSAKTYWNOSCI: CHAR(1) - Status konta ('1' - aktywny, '0' - nieaktywny).
* **KATEGORIAPRODUKTOW:** Grupuje produkty i określa wymagane uprawnienia.
  + KATEGORIAID (PK): NUMBER(\*) - Unikalny identyfikator kategorii.
  + NAZWAKATEGORII: VARCHAR2(100) - Nazwa kategorii.
  + OPIS: CLOB - Dodatkowy opis kategorii.
  + WYMAGANEUPRAWNIENIA: VARCHAR2(50) - Typ pozwolenia wymagany dla tej kategorii (np. 'A', 'B', 'C').
* **AMUNICJA:** Informacje o dostępnych rodzajach amunicji.
  + AMUNICJAID (PK): NUMBER(\*) - Unikalny identyfikator typu amunicji.
  + NAZWA: VARCHAR2(100) - Nazwa amunicji.
  + CENA: NUMBER(10,2) - Cena jednostkowa amunicji.
  + ILOSC: NUMBER(\*) - Dostępna ilość amunicji na stanie.
* **PRODUKT:** Główna tabela asortymentu (głównie broń).
  + PRODUKTID (PK): NUMBER(\*) - Unikalny identyfikator produktu.
  + KATEGORIAID (FK): NUMBER(\*) - Klucz obcy do KATEGORIAPRODUKTOW.
  + AMUNICJAID (FK): NUMBER(\*) - Klucz obcy (opcjonalny) do AMUNICJA, wskazujący domyślny typ amunicji dla broni.
  + NAZWA: VARCHAR2(100) - Nazwa produktu (model broni).
  + NUMERSERYJNY: VARCHAR2(50) - Unikalny numer seryjny broni.
  + CENA: NUMBER(10,2) - Cena produktu (broni).
  + DOSTEPNOSC: CHAR(1) - Status dostępności ('1' - dostępny, '0' - niedostępny/sprzedany).
* **TRANSAKCJA:** Rejestruje operacje sprzedaży.
  + TRANSAKCJAID (PK): NUMBER(\*) - Unikalny identyfikator transakcji.
  + KLIENTID (FK): NUMBER(\*) - Klucz obcy do KLIENT.
  + PRACOWNIKID (FK): NUMBER(\*) - Klucz obcy do PRACOWNIK.
  + DATATRANSAKCJI: DATE - Data i czas dokonania transakcji.
  + WARTOSCTRANSAKCJI: NUMBER(10,2) - Całkowita wartość transakcji.
* **TRANSAKCJAPRODUKT:** Tabela asocjacyjna (szczegóły transakcji). Nazwa w diagramie to ZAMOWIENIEPRODUKT, ale kod PL/SQL używa TRANSAKCJAPRODUKT.
  + TRANSAKCJAID (FK): NUMBER(\*) - Klucz obcy do TRANSAKCJA.
  + PRODUKTID (FK): NUMBER(\*) - Klucz obcy do PRODUKT.
  + ILOSC: NUMBER(\*) - W kontekście procedury TRANSAKCJAPRODUKTCREATE, to pole przechowuje ilość zakupionej *amunicji* powiązanej z danym produktem-bronią w tej transakcji. Dla samego produktu-broni ilość jest domyślnie 1 (pojedyncza sztuka).
  + CENA: NUMBER(10,2) - W procedurze TRANSAKCJAPRODUKTCREATE zapisywana jest tu cena produktu-broni (v\_ProduktCena).

## **2.2. Diagram ERD**

A screenshot of a computer

AI-generated content may be incorrect.

Legenda:

 Primary key

 Foreign key

## **2.3. Relacje Między Encjami**

* **KLIENT - ADRES (jeden-do-jednego):** Każdy klient ma jeden adres. KLIENT.ADRESID -> ADRES.ADRESID.
* **PRACOWNIK - ADRES (jeden-do-jednego):** Każdy pracownik ma jeden adres. PRACOWNIK.ADRESID -> ADRES.ADRESID.
* **TRANSAKCJA - KLIENT (wiele-do-jednego):** Wiele transakcji może być przypisanych do jednego klienta. TRANSAKCJA.KLIENTID -> KLIENT.KLIENTID.
* **TRANSAKCJA - PRACOWNIK (wiele-do-jednego):** Wiele transakcji może być obsłużonych przez jednego pracownika. TRANSAKCJA.PRACOWNIKID -> PRACOWNIK.PRACOWNIKID.
* **PRODUKT - KATEGORIAPRODUKTOW (wiele-do-jednego):** Wiele produktów należy do jednej kategorii. PRODUKT.KATEGORIAID -> KATEGORIAPRODUKTOW.KATEGORIAID.
* **PRODUKT - AMUNICJA (wiele-do-jednego, opcjonalna):** Produkt (broń) może mieć powiązany jeden typ amunicji (lub żaden). PRODUKT.AMUNICJAID -> AMUNICJA.AMUNICJAID.
* **TRANSAKCJA - PRODUKT (wiele-do-wielu poprzez TRANSAKCJAPRODUKT):** Jedna transakcja może zawierać wiele produktów, a jeden produkt może być w wielu transakcjach.
  + TRANSAKCJAPRODUKT.TRANSAKCJAID -> TRANSAKCJA.TRANSAKCJAID.
  + TRANSAKCJAPRODUKT.PRODUKTID -> PRODUKT.PRODUKTID.

# **3. Koncepcja Funkcjonalności w PL/SQL**

Warstwa PL/SQL w systemie pełni rolę silnika logiki biznesowej, zapewniając hermetyzację operacji na danych, walidację oraz spójność. Główne cele i zadania realizowane przez procedury i funkcje PL/SQL obejmują:

* **Zarządzanie Danymi (CRUD):** Implementacja operacji tworzenia, odczytu, aktualizacji i usuwania (lub dezaktywacji) dla kluczowych encji (KLIENT, PRACOWNIK, PRODUKT, AMUNICJA, ADRES, KATEGORIAPRODUKTOW). Procedury te zawierają wbudowaną walidację danych wejściowych (np. sprawdzanie pól wymaganych, formatów, zakresów wartości) oraz obsługę standardowych błędów.
* **Obsługa Procesu Sprzedaży (Transakcji):**
  + Tworzenie nagłówków transakcji.
  + Dodawanie pozycji (produktów-broni wraz z dedykowaną amunicją) do transakcji.
  + Automatyczna aktualizacja stanów magazynowych (dostępność broni, ilość amunicji).
  + Obliczanie i aktualizacja wartości transakcji.
  + Usuwanie pozycji z transakcji z odpowiednią korektą stanów i wartości.
* **Weryfikacja Uprawnień i Zgodności z Przepisami:**
  + Sprawdzanie ważności pozwolenia klienta.
  + Weryfikacja, czy typ pozwolenia klienta uprawnia go do zakupu produktu z danej kategorii.
  + Weryfikacja, czy poziom uprawnień pracownika pozwala mu na sprzedaż produktu z danej kategorii.
* **Zarządzanie Integralnością Danych:** Wykorzystanie mechanizmów transakcyjnych (COMMIT, ROLLBACK) do zapewnienia atomowości operacji modyfikujących wiele tabel (np. podczas dodawania produktu do transakcji).
* **Generowanie Statystyk i Raportów:** Funkcje agregujące dane sprzedażowe, dane o klientach, pracownikach i produktach w celu wsparcia analizy biznesowej.
* **Centralizacja Logiki Biznesowej:** Umieszczenie logiki biznesowej w bazie danych (a nie w aplikacji klienckiej) zwiększa spójność, bezpieczeństwo i ułatwia zarządzanie zmianami.

Każda procedura i funkcja jest zaprojektowana tak, aby realizować konkretne zadanie biznesowe, zwracając odpowiednie statusy lub dane, oraz informując o ewentualnych błędach poprzez mechanizm RAISE\_APPLICATION\_ERROR.

# **4. Opis Operacji CRUD (na przykładzie tabeli KLIENT)**

Operacje CRUD (Create, Read, Update, Delete) są fundamentalnymi działaniami na danych. Poniżej opisano ich implementację dla encji KLIENT, zaznaczając, że analogiczne operacje (z odpowiednimi modyfikacjami) stosuje się do innych tabel w systemie.

## **4.1. Tworzenie (Create) - PROCEDURE KLIENTCREATE**

**Parametry wejściowe:**  
p\_Imie, p\_Nazwisko, p\_NumerPozwolenia, p\_TypPozwolenia, p\_DataWaznosci, p\_Email, p\_Telefon,  
p\_Ulica, p\_NumerDomu, p\_NumerMieszkania, p\_KodPocztowy, p\_Miejscowosc

**Logika:**

* + **Walidacja danych wejściowych:** sprawdzenie obecności wymaganych danych i poprawności (np. format email, data ważności, typ pozwolenia).
  + **Tworzenie adresu:** wywołanie procedury ADRES\_PKG.AdresCreate, która tworzy adres i zwraca v\_AdresID.
  + **Generowanie ID:** pobranie nowego identyfikatora klienta za pomocą sekwencji SQ\_KL\_ID.NEXTVAL.
  + **Operacja SQL:** wstawienie danych do tabeli KLIENT (INSERT INTO).
  + **Obsługa błędów:** w przypadku błędu (np. duplikatu), adres jest usuwany; inne wyjątki są przechwytywane i przekazywane.

**Transakcja:** automatyczne zatwierdzenie (brak jawnego COMMIT – zakładana obsługa na poziomie aplikacji/PLSQL bloków).

**Cel:** Dodanie nowego klienta wraz z jego adresem.

## **4.2. Odczyt (Read)**

**Funkcja:** KLIENTREADBYID

**Parametry wejściowe:** p\_KlientID

**Zwracany typ:** SYS\_REFCURSOR

**Logika:**

* **Walidacja:** sprawdzenie obecności klienta w bazie.
* **Zwrócenie danych:** otwarcie i zwrócenie kursora z wynikiem zapytania SELECT \* FROM KLIENT WHERE KLIENTID = :id.

**Cel:** Pobranie danych pojedynczego klienta na podstawie ID.

## **4.3. Aktualizacja (Update) - PROCEDURE KLIENTUPDATE**

**Parametry wejściowe:**  
p\_KlientID, p\_Imie, p\_Nazwisko, p\_NumerPozwolenia, p\_TypPozwolenia, p\_DataWaznosciPozwolenia,  
p\_Email, p\_Telefon, p\_StatusAktywnosci, p\_Ulica, p\_NumerDomu, p\_NumerMieszkania, p\_KodPocztowy, p\_Miejscowosc

**Logika:**

* **Walidacja:** sprawdzenie obecności p\_KlientID oraz poprawności danych (np. format email, data ważności, typ i status pozwolenia).
* **Pobranie danych bieżących:** dane obecne w bazie są pobierane i przypisywane do zmiennych, aby umożliwić częściową aktualizację (NVL).
* **Aktualizacja klienta:** UPDATE tabeli KLIENT z nowymi danymi.
* **Aktualizacja adresu:** jeżeli podano nowe dane adresowe, wywołanie procedury ADRES\_PKG.ADRESUPDATE.

**Cel:** Zmodyfikowanie danych klienta i ewentualnie jego adresu.

## **4.4. Usuwanie (Delete/Dezaktywacja) - PROCEDURE KLIENTDELETE**

**Parametry wejściowe:** p\_KlientID

**Logika:**

* + **Walidacja:** sprawdzenie obecności klienta w bazie.
  + **Dezaktywacja:** aktualizacja pola STATUSAKTYWNOSCI na '0'.

**Cel:** Dezaktywacja klienta bez usuwania rekordu z bazy (miękkie usunięcie).

# **5. Szczegółowy Opis Wytworzonych Funkcjonalności PL/SQL**

## **5.1 FUNCTION STATISTICPOPULARCITY**

**Cel:** Zwraca statystyki dotyczące **najpopularniejszych miejscowości** (wg liczby klientów).

**Opis działania:**

* Łączy tabelę ADRES z KLIENT i TRANSAKCJA.
* Grupuje dane według MIEJSCOWOSC.
* Oblicza:
  + liczbę unikalnych klientów z danej miejscowości,
  + liczbę transakcji,
  + łączną wartość zakupów.
* Zwraca tylko **5 najpopularniejszych miejscowości** (z największą liczbą klientów).
* Dane są zwracane w formacie CITY\_STATS\_OBJ.

FUNCTION *STATISTICPOPULARCITY* RETURN CITY\_STATS\_TAB PIPELINED  
AS  
BEGIN  
 FOR r IN (  
 SELECT  
 a.MIEJSCOWOSC,  
 *COUNT*(DISTINCT k.KLIENTID) as LICZBA\_KLIENTOW,  
 *COUNT*(t.TRANSAKCJAID) as LICZBA\_TRANSAKCJI,  
 *NVL*(*SUM*(t.WARTOSCTRANSAKCJI), 0) as LACZNA\_WARTOSC\_ZAKUPOW  
 FROM ADRES a  
 JOIN KLIENT k ON a.ADRESID = k.ADRESID  
 LEFT JOIN TRANSAKCJA t ON k.KLIENTID = t.KLIENTID  
 GROUP BY a.MIEJSCOWOSC  
 ORDER BY *COUNT*(DISTINCT k.KLIENTID) DESC  
 FETCH FIRST 5 ROWS ONLY  
 ) LOOP  
 PIPE ROW(CITY\_STATS\_OBJ(  
 r.MIEJSCOWOSC,  
 r.LICZBA\_KLIENTOW,  
 r.LICZBA\_TRANSAKCJI,  
 r.LACZNA\_WARTOSC\_ZAKUPOW  
 ));  
 END LOOP;  
 RETURN;  
END *STATISTICPOPULARCITY*;

## **5.2 FUNCTION STATISTICBESTWORKER**

**Cel:** Zwraca statystyki **najlepszych pracowników** (aktywnych) pod względem liczby i wartości sprzedaży.

**Opis działania:**

* Łączy tabelę PRACOWNIK z TRANSAKCJA.
* Filtrowanie: tylko pracownicy, których STATUSAKTYWNOSCI = '1'.
* Grupowanie po PRACOWNIKID, IMIE, NAZWISKO.
* Oblicza:
  + liczbę transakcji obsłużonych przez pracownika,
  + sumę wartości tych transakcji.
* Wyniki zwracane są jako WORKER\_STATS\_OBJ.

FUNCTION *STATISTICBESTWORKER* RETURN WORKER\_STATS\_TAB PIPELINED  
AS  
BEGIN  
 FOR r IN (  
 SELECT  
 p.PRACOWNIKID,  
 p.IMIE,  
 p.NAZWISKO,  
 *COUNT*(t.TRANSAKCJAID) as LICZBA\_TRANSAKCJI,  
 *NVL*(*SUM*(t.WARTOSCTRANSAKCJI), 0) as WARTOSC\_SPRZEDAZY  
 FROM PRACOWNIK p  
 LEFT JOIN TRANSAKCJA t ON p.PRACOWNIKID = t.PRACOWNIKID  
 WHERE p.STATUSAKTYWNOSCI = '1'  
 GROUP BY p.PRACOWNIKID, p.IMIE, p.NAZWISKO  
 ORDER BY LICZBA\_TRANSAKCJI DESC  
 ) LOOP  
 PIPE ROW(WORKER\_STATS\_OBJ(  
 r.PRACOWNIKID,  
 r.IMIE,  
 r.NAZWISKO,  
 r.LICZBA\_TRANSAKCJI,  
 r.WARTOSC\_SPRZEDAZY  
 ));  
 END LOOP;  
 RETURN;  
END *STATISTICBESTWORKER*;

## **5.3 FUNCTION STATISTICBESTCLIENTS**

**Cel:** Zwraca statystyki **najlepszych klientów** pod względem wydatków.

**Opis działania:**

* Łączy KLIENT z TRANSAKCJA.
* Grupowanie po KLIENTID, IMIE, NAZWISKO.
* Oblicza:
  + liczbę transakcji klienta,
  + sumę jego wydatków.
* Zwraca dane jako CLIENT\_STATS\_OBJ.
* FUNCTION *STATISTICBESTCLIENTS* RETURN CLIENT\_STATS\_TAB PIPELINED  
  AS  
  BEGIN  
   FOR r IN (  
   SELECT  
   k.KLIENTID,  
   k.IMIE,  
   k.NAZWISKO,  
   *COUNT*(t.TRANSAKCJAID) as LICZBA\_TRANSAKCJI,  
   *NVL*(*SUM*(t.WARTOSCTRANSAKCJI), 0) as SUMA\_WYDATKOW  
   FROM KLIENT k  
   LEFT JOIN TRANSAKCJA t ON k.KLIENTID = t.KLIENTID  
   GROUP BY k.KLIENTID, k.IMIE, k.NAZWISKO  
   ORDER BY SUMA\_WYDATKOW DESC  
   ) LOOP  
   PIPE ROW(CLIENT\_STATS\_OBJ(  
   r.KLIENTID,  
   r.IMIE,  
   r.NAZWISKO,  
   r.LICZBA\_TRANSAKCJI,  
   r.SUMA\_WYDATKOW  
   ));  
   END LOOP;  
   RETURN;  
  END *STATISTICBESTCLIENTS*;

## **5.4 FUNCTION STATISTICAVERAGETRANSACTION**

**Cel:** Zwraca ogólne statystyki transakcji.

**Opis działania:**

* Zlicza dane w tabeli TRANSAKCJA, gdzie WARTOSCTRANSAKCJI > 0.
* Oblicza:
  + liczbę transakcji,
  + sumę wartości transakcji,
  + średnią wartość,
  + najmniejszą i największą wartość pojedynczej transakcji.
* Dane są opakowane w TRANSACTION\_STATS\_OBJ.
* FUNCTION *STATISTICAVERAGETRANSACTION* RETURN TRANSACTION\_STATS\_TAB PIPELINED  
  AS  
  BEGIN  
   FOR r IN (  
   SELECT  
   *COUNT*(TRANSAKCJAID) as LICZBA\_TRANSAKCJI,  
   *NVL*(*SUM*(WARTOSCTRANSAKCJI), 0) as SUMA\_TRANSAKCJI,  
   *NVL*(*ROUND*(*AVG*(WARTOSCTRANSAKCJI), 2), 0) as SREDNIA\_WARTOSC,  
   *NVL*(*MIN*(WARTOSCTRANSAKCJI), 0) as NAJMNIEJSZA\_TRANSAKCJA,  
   *NVL*(*MAX*(WARTOSCTRANSAKCJI), 0) as NAJWIEKSZA\_TRANSAKCJA  
   FROM TRANSAKCJA  
   WHERE WARTOSCTRANSAKCJI > 0  
   ) LOOP  
   PIPE ROW(TRANSACTION\_STATS\_OBJ(  
   r.LICZBA\_TRANSAKCJI,  
   r.SUMA\_TRANSAKCJI,  
   r.SREDNIA\_WARTOSC,  
   r.NAJMNIEJSZA\_TRANSAKCJA,  
   r.NAJWIEKSZA\_TRANSAKCJA  
   ));  
   END LOOP;  
   RETURN;  
  END *STATISTICAVERAGETRANSACTION*;

## **5.5 FUNCTION STATISTICPRODUCT**

**Cel:** Zwraca statystyki dotyczące **produktów z kategorii związanych z bronią i akcesoriami**.

**Opis działania:**

* Łączy tabele PRODUKT, KATEGORIAPRODUKTOW, TRANSAKCJAPRODUKT.
* Filtrowanie: NAZWAKATEGORII zawiera „broń”, „Broń” lub „Akcesoria”.
* Grupuje po nazwie produktu i kategorii.
* Oblicza:
  + liczbę zamówień na dany produkt,
  + łączną liczbę sztuk,
  + wartość sprzedaży (ILOŚĆ \* CENA).
* Zwraca dane jako PRODUCT\_STATS\_OBJ.

FUNCTION *STATISTICPRODUCT* RETURN PRODUCT\_STATS\_TAB PIPELINED  
AS  
BEGIN  
 FOR r IN (  
 SELECT  
 p.NAZWA as NAZWA\_BRONI,  
 *MIN*(p.PRODUKTID) as PRZYKLADOWE\_PRODUKTID,  
 kp.NAZWAKATEGORII as KATEGORIA,  
 *COUNT*(tp.PRODUKTID) as LICZBA\_ZAMOWIEN,  
 *SUM*(tp.ILOSC) as LACZNA\_ILOSC,  
 *SUM*(tp.ILOSC \* p.CENA) as LACZNA\_WARTOSC\_SPRZEDAZY  
 FROM PRODUKT p  
 JOIN KATEGORIAPRODUKTOW kp ON p.KATEGORIAID = kp.KATEGORIAID  
 LEFT JOIN TRANSAKCJAPRODUKT tp ON p.PRODUKTID = tp.PRODUKTID  
 WHERE kp.NAZWAKATEGORII LIKE '%broń%'  
 OR kp.NAZWAKATEGORII LIKE '%Broń%'  
 OR kp.NAZWAKATEGORII LIKE '%Akcesoria%'  
 GROUP BY p.NAZWA, kp.NAZWAKATEGORII  
 ORDER BY LICZBA\_ZAMOWIEN DESC  
 ) LOOP  
 PIPE ROW(PRODUCT\_STATS\_OBJ(  
 r.NAZWA\_BRONI,  
 r.PRZYKLADOWE\_PRODUKTID,  
 r.KATEGORIA,  
 r.LICZBA\_ZAMOWIEN,  
 r.LACZNA\_ILOSC,  
 r.LACZNA\_WARTOSC\_SPRZEDAZY  
 ));  
 END LOOP;  
 RETURN;  
END *STATISTICPRODUCT*;

## **5.6 TRIGGER SOFTDELETE**

**Obiekt:** TR\_PRACOWNIK\_SOFTDELETE (trigger)

**Ty :** BEFORE DELETE ON PRACOWNIK FOR EACH ROW

**Logika:**

* **Autonomiczna transakcja:** dzięki PRAGMA AUTONOMOUS\_TRANSACTION, operacja aktualizacji i zatwierdzenia COMMIT działa niezależnie od głównej transakcji.
* **Zmiana statusu:** zamiast fizycznego usunięcia rekordu z tabeli PRACOWNIK, trigger ustawia wartość pola STATUSAKTYWNOSCI na '0', oznaczając pracownika jako nieaktywnego.
* **Zatwierdzenie zmian:** COMMIT trwałych zmian w kontekście autonomicznej transakcji.
* **Zgłoszenie błędu:** użycie RAISE\_APPLICATION\_ERROR zatrzymuje próbę usunięcia rekordu i informuje, że pracownik został tylko oznaczony jako nieaktywny.

**Cel:** Zabezpieczenie danych przed fizycznym usunięciem przez automatyczne zastosowanie **soft delete** (dezaktywacja rekordu) oraz poinformowanie użytkownika o tym działaniu.

create trigger TR\_PRACOWNIK\_SOFTDELETE  
 before delete  
 on PRACOWNIK  
 for each row  
DECLARE  
 PRAGMA AUTONOMOUS\_TRANSACTION;  
BEGIN  
 UPDATE PRACOWNIK  
 SET STATUSAKTYWNOSCI = '0'  
 WHERE PRACOWNIKID = :OLD.PRACOWNIKID;  
  
 COMMIT;  
   
 *RAISE\_APPLICATION\_ERROR*(-20004, 'Pracownik został oznaczony jako nieaktywny.');  
END;  
/

## **5.7 TRIGGER INSERT**

**Obiekt:** TR\_TRANSAKCJAPRODUKT\_INSERT (trigger)

**Typ:** AFTER INSERT ON TRANSAKCJAPRODUKT FOR EACH ROW

**Logika:**

* + **Pobranie powiązanego ID amunicji:** Po wstawieniu nowego rekordu do TRANSAKCJAPRODUKT, trigger pobiera AMUNICJAID z tabeli PRODUKT dla przekazanego PRODUKTID.
  + **Zmniejszenie stanu magazynowego amunicji:** Jeśli produkt jest powiązany z amunicją (AMUNICJAID IS NOT NULL) i ilość zakupionego produktu (:NEW.ILOSC) jest większa od zera, zmniejsza się ilość dostępnej amunicji (ILOSC) w tabeli AMUNICJA o wartość z transakcji.
  + **Oznaczenie produktu jako niedostępnego:** Niezależnie od rodzaju produktu, jego dostępność (DOSTEPNOSC) jest ustawiana na '0' w tabeli PRODUKT, co oznacza, że produkt nie jest już dostępny (np. został sprzedany).
  + **Obsługa wyjątków:** Każdy wyjątek występujący podczas działania triggera jest propagowany dalej za pomocą RAISE, co może przerwać operację nadrzędną.

**Cel:** Automatyczna aktualizacja stanów magazynowych po dokonaniu zakupu oraz wyłączenie dostępności produktu po jego sprzedaży.

create trigger TR\_TRANSAKCJAPRODUKT\_INSERT  
 after insert  
 on TRANSAKCJAPRODUKT  
 for each row  
DECLARE  
 v\_AmunicjaID PRODUKT.AMUNICJAID%TYPE;  
BEGIN  
 SELECT AMUNICJAID INTO v\_AmunicjaID  
 FROM PRODUKT  
 WHERE PRODUKTID = :NEW.PRODUKTID;  
  
 IF v\_AmunicjaID IS NOT NULL AND :NEW.ILOSC > 0 THEN  
 UPDATE AMUNICJA  
 SET ILOSC = ILOSC - :NEW.ILOSC  
 WHERE AMUNICJAID = v\_AmunicjaID;  
 END IF;  
  
 UPDATE PRODUKT  
 SET DOSTEPNOSC = '0'  
 WHERE PRODUKTID = :NEW.PRODUKTID;  
  
EXCEPTION  
 WHEN OTHERS THEN  
 RAISE;  
END;  
/

## **5.8 TRIGGER UPDATE**

**Obiekt:** TR\_TRANSAKCJAPRODUKT\_UPDATE (trigger)

**Typ:** AFTER UPDATE ON TRANSAKCJAPRODUKT FOR EACH ROW

**Logika:**

* **Identyfikacja powiązania z amunicją:** Po każdej aktualizacji rekordu w tabeli TRANSAKCJAPRODUKT, trigger pobiera AMUNICJAID z tabeli PRODUKT na podstawie zaktualizowanego PRODUKTID.
* **Reakcja na zmianę ilości:**
  + Jeśli produkt jest powiązany z amunicją (v\_AmunicjaID IS NOT NULL) i ilość została **zmniejszona**, ilość amunicji w magazynie zostaje **zwiększona** o różnicę (:OLD.ILOSC - :NEW.ILOSC).
  + Jeśli ilość została **zwiększona**, ilość amunicji w magazynie zostaje **zmniejszona** o różnicę (:NEW.ILOSC - :OLD.ILOSC).
* **Obsługa wyjątków:** Wszystkie błędy są przekazywane dalej przy użyciu RAISE, co pozwala zidentyfikować źródło problemu przy modyfikacji danych.

**Cel:** Dynamiczna synchronizacja stanu magazynowego amunicji w przypadku edycji ilości zakupionego produktu w transakcji.

create trigger TR\_TRANSAKCJAPRODUKT\_UPDATE  
 after update  
 on TRANSAKCJAPRODUKT  
 for each row  
DECLARE  
 v\_AmunicjaID PRODUKT.AMUNICJAID%TYPE;  
BEGIN  
 SELECT AMUNICJAID INTO v\_AmunicjaID  
 FROM PRODUKT  
 WHERE PRODUKTID = :NEW.PRODUKTID;  
  
 IF v\_AmunicjaID IS NOT NULL THEN  
 IF :NEW.ILOSC < :OLD.ILOSC THEN  
 UPDATE AMUNICJA  
 SET ILOSC = ILOSC + (:OLD.ILOSC - :NEW.ILOSC)  
 WHERE AMUNICJAID = v\_AmunicjaID;  
 ELSIF :NEW.ILOSC > :OLD.ILOSC THEN  
 UPDATE AMUNICJA  
 SET ILOSC = ILOSC - (:NEW.ILOSC - :OLD.ILOSC)  
 WHERE AMUNICJAID = v\_AmunicjaID;  
 END IF;  
 END IF;  
  
EXCEPTION  
 WHEN OTHERS THEN  
 RAISE;  
END;  
/

# **6. Uporządkowanie projektu**

System zawiera **8 pakietów PL/SQL**, logicznie podzielonych według głównych obszarów funkcjonalnych aplikacji:

1. **ADRES\_PKG**
   * Obsługuje operacje CRUD na danych adresowych.
   * Współdziała z innymi modułami (np. KLIENT\_PKG, PRACOWNIK\_PKG).
2. **AMMO\_PKG**
   * Zajmuje się gospodarką magazynową amunicji.
   * Uwzględnia kontrolę stanów, dodawanie nowych typów..
3. **KATEGORIA\_PKG**
   * Zarządza kategoriami produktów.
   * Ułatwia kategoryzację i filtrowanie produktów w systemie.
4. **KLIENT\_PKG**
   * Odpowiada za pełen cykl życia klienta: rejestrację, edycję, dezaktywację.
   * Zapewnia integralność danych klienta i powiązanego adresu.
5. **PRACOWNIK\_PKG**
   * Obsługuje dane personalne i zarządzanie aktywnością pracowników.
   * Zapewnia integralność danych pracownika i powiązanego adresu.
6. **PRODUKT\_PKG**
   * Operuje na produktach – dodawanie, modyfikacja, powiązania z amunicją.
   * Istotna część logiki sprzedażowej systemu.
7. **STATYSTYKA\_PKG**
   * Generuje zestawienia i analizę danych.
   * Służy do celów analitycznych i zarządczych.
8. **TRANSAKCJA\_PKG**
   * Zajmuje się obsługą transakcji sprzedażowych.
   * Powiązany z produktami, klientami i pracownikami – kluczowy element działania sklepu lub systemu sprzedażowego.

# **7. Prezentacja Działania Aplikacji (Interfejsu) w Kontekście Wykorzystania PL/SQL**

Poniżej przedstawiono sposób, w jaki interfejs użytkownika wykorzystuje zdefiniowane procedury i funkcje PL/SQL do realizacji głównych funkcjonalności systemu.

## **7.1. Zarządzanie Klientami**

* **Wyświetlanie listy klientów:**
  + Aplikacja wywołuje FUNCTION KLIENTREAD() aby pobrać dane wszystkich klientów.
  + Wynikowy kursor jest przetwarzany i dane są wyświetlane w tabeli/liście w interfejsie.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* **Dodawanie nowego klienta:**
  + Użytkownik wypełnia formularz z danymi klienta i adresu.
  + Po zatwierdzeniu, aplikacja wywołuje PROCEDURE KLIENTCREATE(...) przekazując dane z formularza.
  + Aplikacja obsługuje komunikaty o sukcesie lub błędzie (np. na podstawie kodów RAISE\_APPLICATION\_ERROR).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

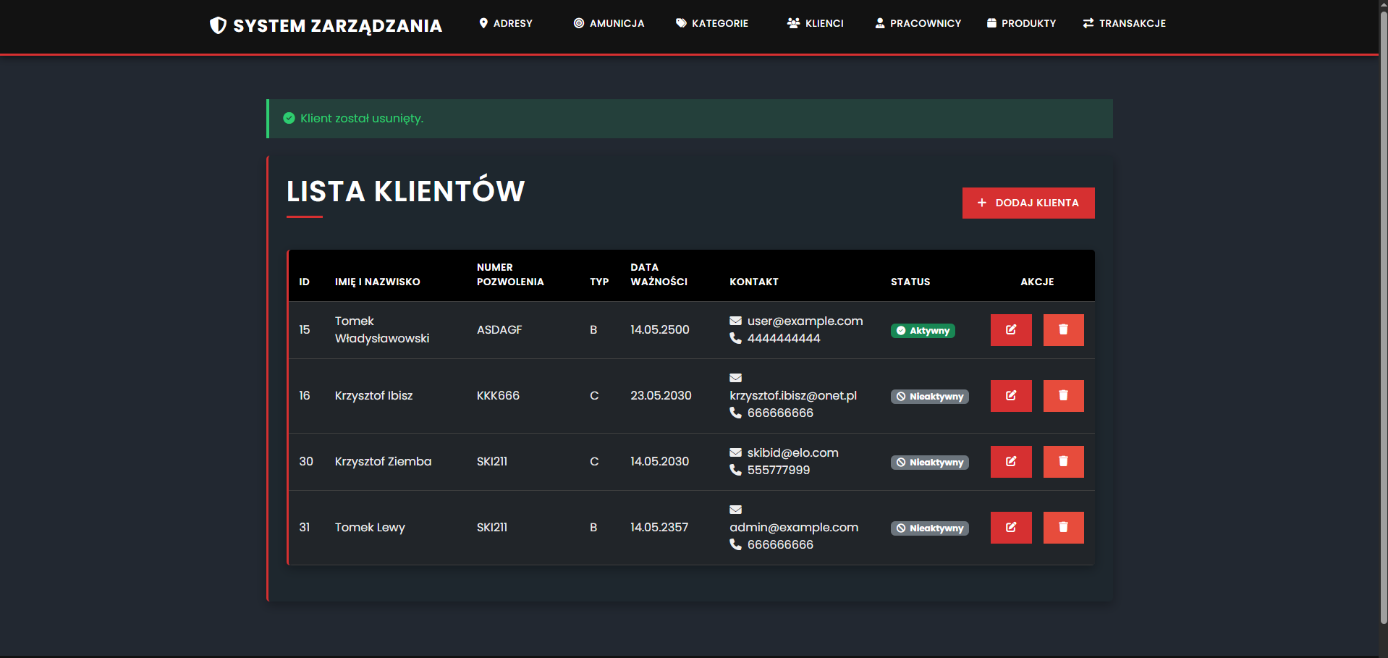
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* **Przeglądanie/Edycja danych klienta:**
  + Użytkownik wybiera klienta z listy. Aplikacja wywołuje FUNCTION KLIENTREADBYID(id\_klienta) aby pobrać szczegółowe dane.
  + Dane są wyświetlane w formularzu edycyjnym.
  + Po modyfikacji i zapisaniu, aplikacja wywołuje PROCEDURE KLIENTUPDATE(id\_klienta, ...) z nowymi danymi.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* **Dezaktywacja klienta:**
  + Użytkownik wybiera opcję "Usuń" dla klienta.
  + Aplikacja wywołuje PROCEDURE KLIENTDELETE(id\_klienta).
  + Interfejs odświeża listę lub status klienta.



Obraz zawierający tekst, zrzut ekranu, oprogramowanie, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## **7.2. Zarządzanie Asortymentem (Produkty, Amunicja, Kategorie)**

* **Przeglądanie katalogu produktów:** Aplikacja używa ProduktRead()  do wyświetlenia listy broni i akcesoriów.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* **Zarządzanie stanami amunicji:** Interfejs do edycji ilości dostępnej amunicji wywołuje AmunicjaUpdateIlosc(id\_amunicji, nowa\_ilosc).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* **Dodawanie nowych produktów/kategorii:** Formularze w interfejsie wywołują odpowiednie procedury ProduktCreate() lub KategoriaCreate().

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## **7.3. Realizacja Transakcji Sprzedaży**

1. **Inicjacja Transakcji:**
   * Użytkownik wybiera opcję “Dodaj transakcję”.
   * Aplikacja wywołuje PROCEDURE TRANSAKCJACREATE(p\_KlientID, p\_PracownikID, p\_NewTransakcjaID => id\_nowej\_transakcji).
   * Interfejs przechodzi do widoku dodawania nowej transakcji (z id\_nowej\_transakcji).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Dodawanie Produktu do Transakcji:**
   * Pracownik wyszukuje produkt (broń) w katalogu.
   * Wprowadza ilość amunicji (jeśli dotyczy).
   * Aplikacja wywołuje PROCEDURE TRANSAKCJAPRODUKTCREATE(id\_transakcji, id\_produktu, 1, ilosc\_amunicji).
   * Interfejs odświeża listę pozycji w transakcji, sumę częściową i całkowitą (która jest aktualizowana w bazie). W przypadku błędu (np. brak uprawnień klienta, brak produktu), wyświetlany jest komunikat.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Usuwanie Produktu z Transakcji:**
   * Pracownik wybiera pozycję do usunięcia z bieżącej transakcji.
   * Aplikacja wywołuje PROCEDURE TRANSAKCJAPRODUKTDELETE(id\_transakcji, id\_produktu\_do\_usuniecia).
   * Interfejs aktualizuje widok transakcji.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Finalizacja Transakcji:**
   * Po dodaniu wszystkich produktów, pracownik zatwierdza transakcję. W tym momencie dane są już zapisane w bazie (COMMIT w procedurach PL/SQL). Możliwe jest sprawdzenie szczegółów transakcji.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## **7.4. Zarządzanie Pracownikami**

* **Moduł administracyjny:** Umożliwia dodawanie, edycję (np. zmiana poziomu uprawnień) i dezaktywację kont pracowników poprzez wywoływanie odpowiednich procedur PL/SQL. Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

  Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## **7.5. Dostęp do Statystyk i Raportów**

* **Panel raportowy:** W dedykowanej sekcji aplikacji użytkownik ma wgląd w statystyki.
  + Aplikacja wywołuje odpowiednią funkcję statystyczną (np. STATISTICBESTCLIENTS(), STATISTICAVERAGETRANSACTION(), STATISTICPRODUCT()).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

* + Dla STATISTICPOPULARCITY(), aplikacja wyświetli listę 5 miast z największą liczbą klientów.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Wykorzystanie warstwy PL/SQL dla logiki biznesowej zapewnia, że reguły walidacji, obliczenia i modyfikacje danych są wykonywane spójnie i bezpiecznie, niezależnie od interfejsu aplikacji, który się z nimi komunikuje. Aplikacja pełni rolę interfejsu dla użytkownika, delegując przetwarzanie danych do bazy.