

Spis treści

1. Wprowadzenie i Opis Problemu	2
2. Koncepcja Bazy Danych.....	3
2.1. Opis Encji.....	3
2.2. Diagram ERD.....	5
2.3. Relacje Między Encjami	5
3. Koncepcja Funkcjonalności w PL/SQL.....	6
4. Opis Operacji CRUD (na przykładzie tabeli KLIENT)	7
4.1. Tworzenie (Create) - PROCEDURE KLIENTCREATE	7
4.2. Odczyt (Read)	8
4.3. Aktualizacja (Update) - PROCEDURE KLIENTUPDATE	8
4.4. Usuwanie (Delete/Dezaktywacja) - PROCEDURE KLIENTDELETE.....	8
5. Szczegółowy Opis Wytworzonych Funkcjonalności PL/SQL.....	9
5.1 FUNCTION STATISTICPOPULARCITY	9
5.2 FUNCTION STATISTICBESTWORKER	10
5.3 FUNCTION STATISTICBESTCLIENTS	10
5.4 FUNCTION STATISTICAVERAGETRANSACTION	11
5.5 FUNCTION STATISTICPRODUCT	12
5.6 TRIGGER SOFTDELETE.....	13
5.7 TRIGGER INSERT.....	13
5.8 TRIGGER UPDATE.....	14
6. Uporządkowanie projektu	16
7. Prezentacja Działania Aplikacji (Interfejsu) w Kontekście Wykorzystania PL/SQL.....	17
7.1. Zarządzanie Klientami.....	17
7.2. Zarządzanie Asortymentem (Produkty, Amunicja, Kategorie)	20
7.3. Realizacja Transakcji Sprzedaży	21
7.4. Zarządzanie Pracownikami.....	24
7.5. Dostęp do Statystyk i Raportów	25

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 różny poziom 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.

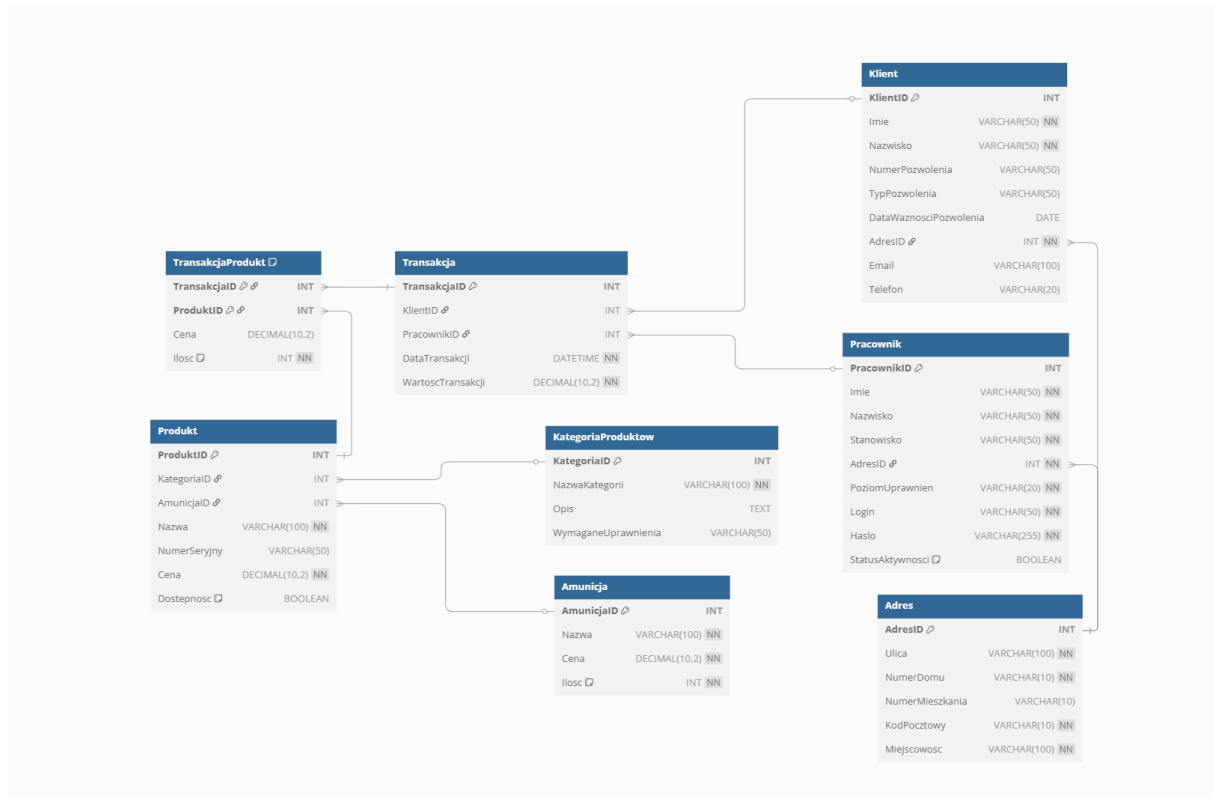
2.1. Opis Encji

- **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.

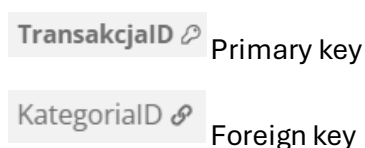
- HASLO: VARCHAR2(255) - Hasło pracownika (hash).
- STATUSAKTYWNOSCI: CHAR(1) - Status konta ('1' - aktywny, '0' - nieaktywny).
- **KATEGORIAPRODUKTOW:** Grupuje produkty i określa wymagane uprawnienia.
 - KATEGORIAID (PK): NUMBER(*) - Unikalny identyfikator kategorii.
 - NAZWAKATEGORII: VARCHAR2(100) - 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



Legenda:



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 obsługiwanych 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:
 - o liczbę zamówień na dany produkt,
 - o łączną liczbę sztuk,
 - o 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.');
```

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.
- Współdziela 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

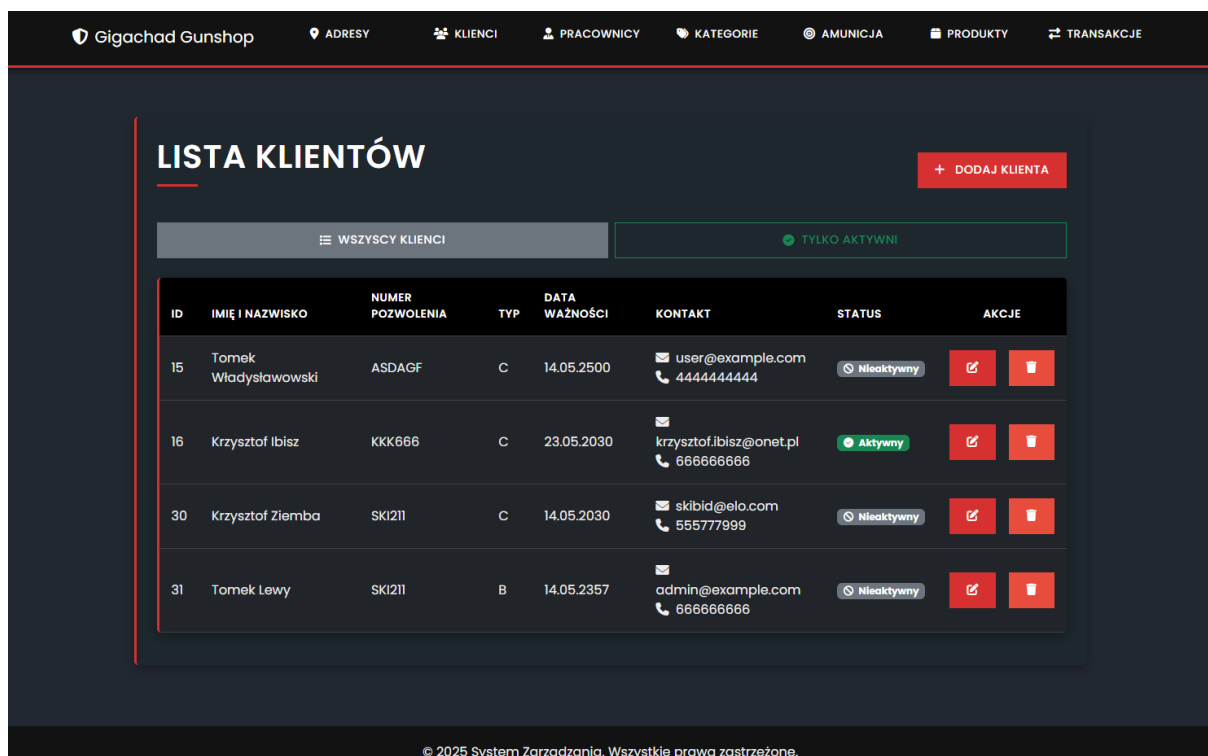
- Zajmuje się obsługą transakcji sprzedażowych.
- Powiązany z produktami, klientami i pracownikami – kluczowy element działania sklepu lub systemu sprzedażowego.

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.



- **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).

DODAJ NOWEGO KLIENTA

IMIĘ

Tomek

NAZWISKO

Zygmuntowski

NUMER POZWOLENIA

SKI219

TYP POZWOLENIA

B

DATA WAŻNOŚCI *

14.05.2060

EMAIL

tomek.zygmuntowski@onet.pl

TELEFON

+48 119 552 556

DANE ADRESOWE

ULICA

ul. Miejska

NUMER DOMU

14c

NUMER MIESZKANIA

KOD POCZTOWY

35-213

MIEJSCOWOŚĆ

Lipinki tużyckie

← ANULUJ

+ DODAJ KLIENTA

Klient został utworzony.

LISTA KLIENTÓW

+ DODAJ KLIENTA


WSZYSTCY KLIENTY


TYLKO AKTYWNI


ID	IMIĘ I NAZWISKO	NUMER POZWOLENIA	TYP	DATA WAŻNOŚCI	KONTAKT	STATUS	AKCJE
15	Tomek Władysławowski	ASDAGF	C	14.05.2500	<div>user@example.com</div> <div>4444444444</div>	Nieaktywny	<div></div> <div></div>
16	Krzysztof Ibisz	KKK666	C	23.05.2030	<div>krzysztof.ibisz@onet.pl</div> <div>6666666666</div>	Aktywny	<div></div> <div></div>
30	Krzysztof Ziomba	SKI211	C	14.05.2030	<div>skibid@elo.com</div> <div>555777999</div>	Nieaktywny	<div></div> <div></div>
31	Tomek Lewy	SKI211	B	14.05.2357	<div>admin@example.com</div> <div>6666666666</div>	Nieaktywny	<div></div> <div></div>
70	Tomek Zygmuntowski	SKI219	B	14.05.2060	<div>tomek.zygmuntowski@onet.pl</div> <div>+48 119 552 556</div>	Aktywny	<div></div> <div></div>


- 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.


 **EDYTUJ KLIENTA**


 IMIĘ


 NAZWISKO


 NUMER POZWOLENIA


 TYP POZWOLENIA


 DATA WAŻNOŚCI *


 EMAIL


 TELEFON


 STATUS AKTYWNOŚCI


 **DANE ADRESOWE**


 ULICA


 NUMER DOMU

 NUMER MIESZKANIA

 KOD POCZTOWY

 MIEJSCOWOŚĆ

 ANULUJ

 ZAPISZ ZMIANY

- **Dezaktywacja klienta:**

- Użytkownik wybiera opcję "Usuń" dla klienta.
- Aplikacja wywołuje PROCEDURE KLIENTDELETE(id_klienta).
- Interfejs odświeża listę lub status klienta.





















SYSTEM ZARZĄDZANIA

ADRESYAMUNICJA
KATEGORIE
KLIENTY
PRACOWNICY
PRODUKTY
TRANSAKCJE

Klient został usunięty.

LISTA KLIENTÓW

+ DODAJ KLIENTA

ID	IMIĘ I NAZWISKO	NUMER POZWOLENIA	TYP	DATA WAŻNOŚCI	KONTAKT	STATUS	AKCJE
15	Tomek Włodysławowski	ASDAGF	B	14.05.2500	 user@example.com  4444444444	 Aktywny	 
16	Krzysztof Ibisz	KKK666	C	23.05.2030	 krzysztof.ibisz@onet.pl  6666666666	 Nieaktywny	 
30	Krzysztof Ziomba	SKI211	C	14.05.2030	 skibid@elo.com  555777999	 Nieaktywny	 
31	Tomek Lewy	SKI211	B	14.05.2357	 admin@example.com  6666666666	 Nieaktywny	 

LISTA KLIENTÓW

+ DODAJ KIENTA

WSZYSCY KLIENTI

TYLKO AKTYWNI

ID	IMIĘ I NAZWISKO	NUMER POZWOLENIA	TYP	DATA WAŻNOŚCI	KONTAKT	STATUS	AKCJE
15	Tomek Władysławowski	ASDAGF	C	14.05.2500	<div>user@example.com</div> <div>4444444444</div>	Nieaktywny	<div></div> <div></div>
16	Krzysztof Ibisz	KKK666	C	23.05.2030	<div>krzysztof.ibisz@onet.pl</div> <div>666666666</div>	Aktywny	<div></div> <div></div>
30	Krzysztof Ziemia	SKI211	C	14.05.2030	<div>skibid@elo.com</div> <div>555777999</div>	Nieaktywny	<div></div> <div></div>
31	Tomek Lewy	SKI211	B	14.05.2357	<div>admin@example.com</div> <div>666666666</div>	Nieaktywny	<div></div> <div></div>
70	Krzystian Zygmuntowski	SKI219	B	14.05.2060	<div>tomek.zygmuntowski@onet.pl</div> <div>+48 119 552 556</div>	Aktywny	<div></div> <div></div>

7.2. Zarządzanie Asortymentem (Produkty, Amunicja, Kategorie)

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

LISTA PRODUKTÓW

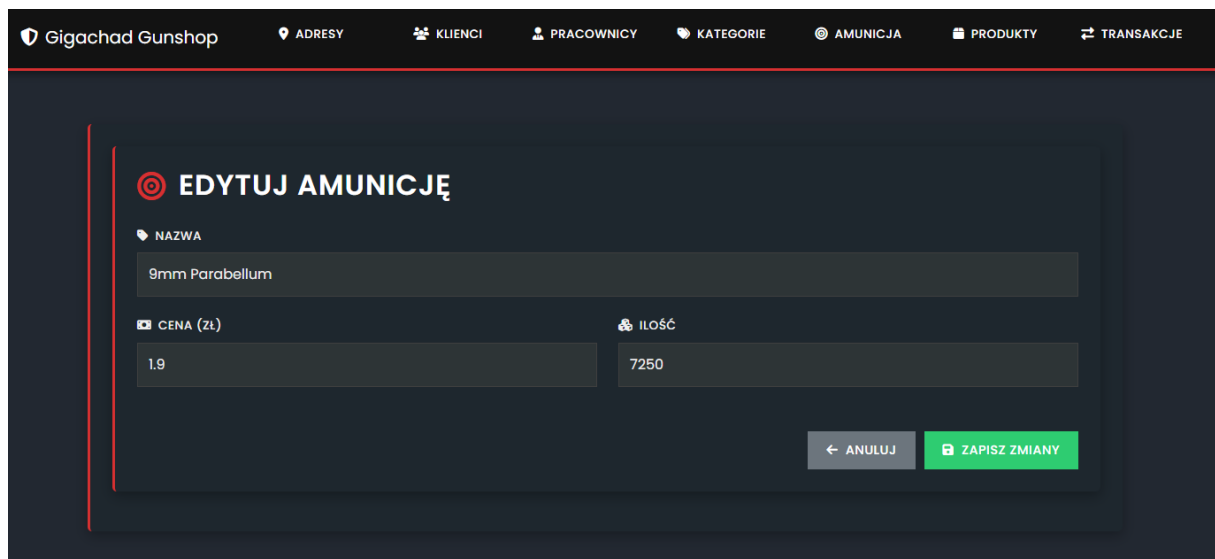
+ DODAJ PRODUKT

WSZYSTKIE PRODUKTY

TYLKO DOSTĘPNE

ID	NAZWA	KATEGORIA	TYP	NUMER SERYJNY	CENA	STATUS	AKCJE
3	Mi7	Broń Długa	C	SN54321	5,000.00 zł	Dostępny	<div>EDYTUJ</div> <div>USUŃ</div>
10	Desert Eagle	Broń Krótka	B	SNZ325	1,500.00 zł	Dostępny	<div>EDYTUJ</div> <div>USUŃ</div>
12	Sentinel	Broń Długa	C	SNZ326	4,000.00 zł	Niedostępny	<div>EDYTUJ</div> <div>USUŃ</div>
14	Desert Eagle	Broń Krótka	B	SN67890	4,000.00 zł	Niedostępny	<div>EDYTUJ</div> <div>USUŃ</div>
15	Sentinel	Broń Długa	C	SNZ327	5,000.00 zł	Dostępny	<div>EDYTUJ</div> <div>USUŃ</div>
16	30-30 Repeater	Broń Długa	C	APEX27	4,650.00 zł	Niedostępny	<div>EDYTUJ</div> <div>USUŃ</div>
17	Sentinel	Broń Krótka	B	SKS141	1,515.00 zł	Dostępny	<div>EDYTUJ</div> <div>USUŃ</div>
18	AK-47	Broń Długa	C	53525	2,512.00 zł	Dostępny	<div>EDYTUJ</div> <div>USUŃ</div>

- Zarządzanie stanami amunicji:** Interfejs do edycji ilości dostępnej amunicji wywołuje AmunicjaUpdateIlosc(id_amunicji, nowa_ilosc).



EDYTUJ AMUNICJĘ

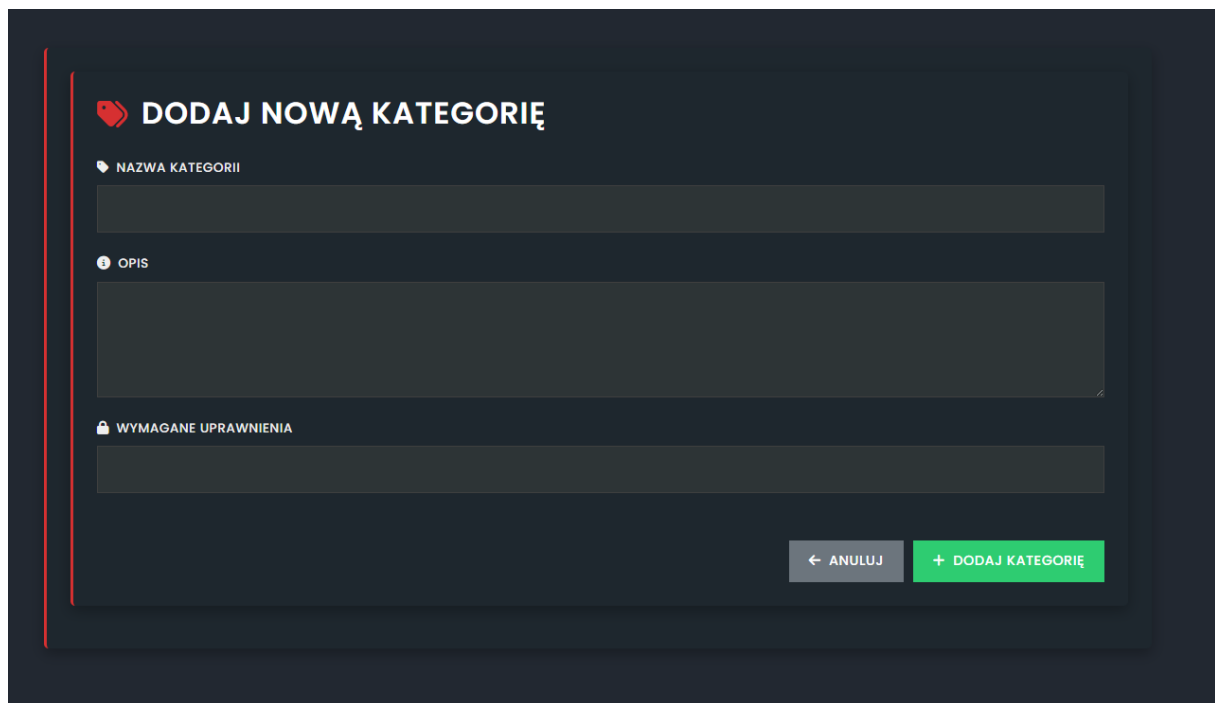
NAZWA
9mm Parabellum

CENA (zł)
1.9

ILOŚĆ
7250

← ANULUJ ZAPISZ ZMIANY

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



DODAJ NOWĄ KATEGORIĘ

NAZWA KATEGORII

OPIS

WYMAGANE UPRAWNIENIA

← ANULUJ + DODAJ KATEGORIĘ

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).

Gigachad Gunshop

ADRESY

KLIENCI

PRACOWNICY

KATEGORIE

AMUNICJA

PRODUKTY

TRANSAKCJE

DODAJ NOWĄ TRANSAKCJĘ

KLIENT

Wybierz klienta

PRACOWNIK

Wybierz pracownika

WYBRANE PRODUKTY

+ DODAJ PRODUKT

PRODUKT

Wybierz produkt

Wartość transakcji zostanie obliczona automatycznie.

Suma: 0.00 zł

← ANULUJ

ZAPISZ TRANSAKCJĘ

2. 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.

DODAJ NOWĄ TRANSAKCJĘ

KLIENT

Krzysztof Ibisz (C)

PRACOWNIK

Tomek Pędziwiatr (C)

WYBRANE PRODUKTY

+ DODAJ PRODUKT

PRODUKT

AK-47 (2,512.00 zł)

PRODUKT

Sentinel (5,000.00 zł)

ILOŚĆ AMUNICJI

0

PRODUKT

Wybierz produkt

Wybierz produkt

Desert Eagle (1,500.00 zł)

M17 (5,000.00 zł)

Sentinel (1,515.00 zł)

Wartość transakcji zostanie obliczona automatycznie.

Suma: 7512.00 zł

← ANULUJ

ZAPISZ TRANSAKCJĘ

3. 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.

DODAJ NOWĄ TRANSAKCJĘ

KLIENT

Krzysztof Ibisz (C)

PRACOWNIK

Tomek Pędziwiatr (C)

WYBRANE PRODUKTY

+ DODAJ PRODUKT

PRODUKT	
AK-47 (2,512.00 zł)	
PRODUKT	ILOŚĆ AMUNICJI
Sentinel (5,000.00 zł)	0
PRODUKT	
Wybierz produkt	

Wartość transakcji zostanie obliczona automatycznie.

Suma: 7512.00 zł

← ANULUJ

ZAPISZ TRANSAKCJĘ

DODAJ NOWĄ TRANSAKCJĘ

KLIENT

Krzysztof Ibisz (C)

PRACOWNIK

Tomek Pędziwiatr (C)

WYBRANE PRODUKTY

+ DODAJ PRODUKT

Sentinel (5,000.00 zł)	0
PRODUKT	
Wybierz produkt	

Wartość transakcji zostanie obliczona automatycznie.

Suma: 5000.00 zł

← ANULUJ

ZAPISZ TRANSAKCJĘ

4. 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.

Transakcja została utworzona.

LISTA TRANSAKCJI

+ DODAJ TRANSAKCJĘ

Od dd.mm.rrrr

Do dd.mm.rrrr

FILTRUJ

ID	KLIENT	PRACOWNIK	DATA	WARTOŚĆ	AKCJE		
172	Krzysztof Ibisz	Tomek Pędziwiatr	31.05.2025	9,695.00 zł	SZCZEGÓŁY	EDYTUJ	USUŃ
176	Krzysztof Ibisz	Tomek Pędziwiatr	31.05.2025	4,950.00 zł	SZCZEGÓŁY	EDYTUJ	USUŃ
178	Krzysztof Ibisz	Tomek Pędziwiatr	31.05.2025	5,019.00 zł	SZCZEGÓŁY	EDYTUJ	USUŃ

7.4. Zarządzanie Pracownikami

- Moduł administracyjny:** Umożliwia dodawanie, edycję (np. zmiana poziomu uprawnień) i dezaktywację kont pracowników poprzez wywoływanie odpowiednich procedur PL/SQL.

Pracownik został zaktualizowany.


LISTA PRACOWNIKÓW

+ DODAJ PRACOWNIKA

WSZYSCY PRACOWNICY

TYLKO AKTYWNI

ID	IMIĘ I NAZWISKO	STANOWISKO	LOGIN	TYP POZWOLENIA	STATUS	AKCJE	
2	Michał Lewandowski	Kierownik	mlewandowski	B	Nieaktywny	EDYTUJ	USUŃ
3	Tomek Pędziwiatr	Kierownik	ekubiak	C	Aktywny	EDYTUJ	USUŃ
11	Tomek Kubiak	Sprzedawca	bazada	B	Nieaktywny	EDYTUJ	USUŃ
50	Tomek Czółkowski	Kierownik	zadada	C	Nieaktywny	EDYTUJ	USUŃ

 EDYTUJ PRACOWNIKA

IMIĘ

Tomek

NAZWISKO

Pędziwiatr

STANOWISKO

Kierownik

TYP POZWOLENIA

C

LOGIN

ekubiak

HASŁO

STATUS AKTYWNOŚCI

Aktywny

DANE ADRESOWE

ULICA

ul. Miejska

NUMER DOMU

12A

NUMER MIESZKANIA

1144

KOD POCZTOWY

35-317

MIEJSCOWOŚĆ

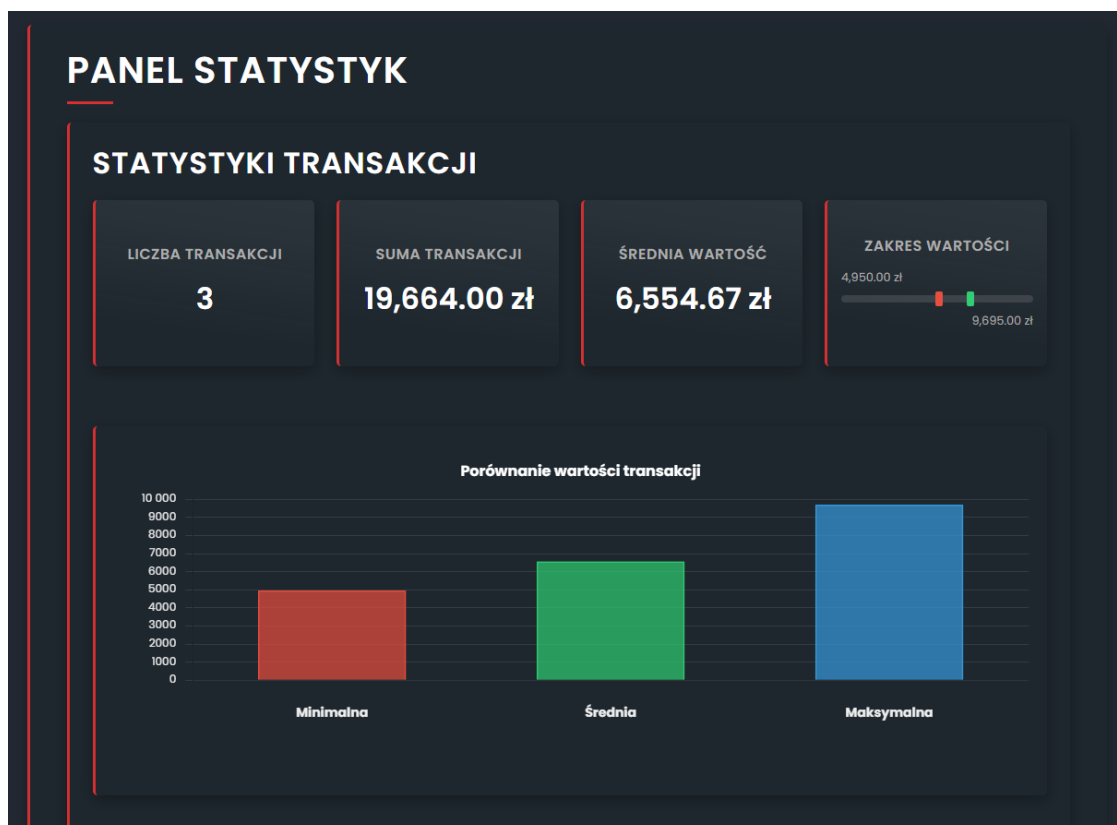
Rzeszów

← ANULUJ

ZAPISZ ZMIANY

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()`).



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



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.