

# Projekt bazy danych sklepu internetowego

Bazy danych

Prowadzący: Antoni Ligęza  
e-mail: [aligeza@pwsz-ns.edu.pl](mailto:aligeza@pwsz-ns.edu.pl)

# Spis treści

<b>1. Projektowanie Bazy Danych</b>	<b>3</b>
1.1. Sformułowanie celu i zadań . . . . .	3
1.1.1. Analiza istniejącej bazy — analiza stanu wyjściowego . . . . .	3
1.1.2. Lista tabel w projektowanej bazie . . . . .	4
1.1.3. Związki między elementami: . . . . .	7
1.1.4. Diagram przypadków użycia . . . . .	7
1.1.5. Diagram klas . . . . .	9
1.1.6. Wyjściowa funkcjonalność . . . . .	9
1.2. Definiowanie tabel, wierzchołków integralności, perspektyw . . . . .	10
1.2.1. Procedury . . . . .	10
1.2.2. Funkcje . . . . .	11
<b>Bibliografia</b>	<b>12</b>
<b>Spis rysunków</b>	<b>13</b>

# Projektowanie Bazy Danych

Projekt: Przygotowanie Systemu Zarządzania Bazą Danych (dalej SZBD) zawierającego interfejs i bazę danych dla sklepu internetowego zajmującego się sprzedażą akcesorii kuchennych

Obecnie zamawiający SZBD sprzedaje towar na popularnym serwisie aukcyjnym.

## 1.1. Sformułowanie celu i zadań

### 1. Minimalizacja czasu

- pomiędzy złożeniem zamówienia przez klienta a przekazaniem towaru do wysyłki w przypadku wysyłki za pobraniem
- pomiędzy zaksięgowaniem wpłaty klienta a przekazaniem towaru do wysyłki w przypadku płatności z góry

### 2. Minimalizacja ilości osób zaangażowanych w zarządzanie sprzedażą

### 3. Ujednolicenie istniejącej bazy produktów i ich kategorii

### 4. Uniezależnienie się od wykorzystywanego portalu aukcyjnego

### 5. Automatyzacja wystawiania faktur/rachunków

### 6. Zapewnienie łatwej kontroli nad aktualnymi stanami magazynowymi produktów

### 7. Zapewnienie kontroli dostępu do danych

### 8. Ograniczenie modyfikacji danych dla poszczególnych pracowników

#### 1.1.1. Analiza istniejącej bazy — analiza stanu wyjściowego

Zamawiający SZBD sprzedaje towar na popularnym serwisie aukcyjnym. Ze względu na powiększenie oferowanego asortymentu istniejący model funkcjonowania przedsiębiorstwa przestaje być wystarczająco wydajny ze względu na źle przemyślany schemat przechowywania danych

1. Opisy produktów jako pliki tekstowe przechowywane w strukturze katalogowej odpowiadającej kategoriom
2. Stany magazynowe w arkuszu kalkulacyjnym (bez multidostępu do zapisu)
3. Ceny w arkuszu kalkulacyjnym
4. Rozliczenia roczne w arkuszu kalkulacyjnym
5. Dokumenty tj. faktury, raporty z przekazania do wysyłki) w formie papierowej
6. Baza kupujących przechowywana jest przez system aukcyjny

### W jaki sposób baza jest wykorzystywana?

Aby dodać produkt do sprzedaży, wyszukiwany jest odpowiedni plik tekstowy z jego opisem oraz sprawdzane cena i stan magazynowy w arkuszu kalkulacyjnym. Kategoria ustalana jest ręcznie w arkuszu kalkulacyjnym zawierającym stany magazynowe. Produkty są wystawiane ręcznie na popularnym serwisie aukcyjnym. Informacje o kupnie: jaki towar, w jakiej ilości, w jakiej cenie oraz o wpłacie dostarczane są przez serwis aukcyjny w formie tekstowej (elektronicznie). Do osoby odpowiedzialnej za gromadzenie towaru w paczkę do wysyłki i przekazanie paczki kurierowi przekazywana jest w formie wydruku. Informacje potrzebne do generowania raportów przepisywane są do arkusza kalkulacyjnego ręcznie.

Problemy:

1. Dane wyszukiwane są ręcznie.
2. Duże ryzyko błędu ludzkiego podczas ręcznego przenoszenia danych.

### 1.1.2. Lista tabel w projektowanej bazie

#### 1. Kategoria:

Nazwa	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
# kategoria_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
nazwa	varchar(30)	-	nie	nieznana	-
FK id_nadrzędnej	int(10)	unsigned	tak	null	-

- # kategoria\_id - pole jednoznacznie identyfikujące kategorię
- nazwa - nazwa kategorii może się powtarzać ( np. podkategoria "pozostałe" w różnych kategoriach)
- FK id\_nadrzędnej - id kategorii nadrzędnej, kategoriami poziomu najwyższego są te, gdzie id\_nadrzędnej ma wartość NULL

#### 2. Klient:

Nazwa	Typ	Atrybuty	Null	Ustawienia domyslnie	Dodatkowo
klient_id	int(10)	unsigned	nie	brak	A_INC
typ	enum('p' 'f')	-	nie	p	-
nazwisko	varchar(50)	-	tak	null	-
imie	varchar(50)	-	tak	null	-
NIP	bigint(20)	-	tak	null	-
nazwa_firmy	varchar(100)	-	tak	null	-
dom_adr_wys	varchar(200)	-	tak	null	-
login	varchar(50)	-	nie	brak	-
haslo	varchar(32)	-	nie	brak	-

- # klient\_id, - pole jednoznacznie identyfikujące klienta
- typ - 'p'- klient indywidualny 'f' -firma
- nazwisko -
- imię
- NIP
- nazwa\_firmy
- domyslny\_adres\_wysylki - wartość wstawiana domyslnie do pola 'adres\_wysylki'
- login
- haslo

### 3. Pracownik

Nazwa	Typ	Atrybuty	Null	Ustawienia domyslnie	Dodatkowo
pracownik_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
login	varchar(50)	-	nie	brak	-
haslo	varchar(32)	-	nie	brak	-
uprawnienia	varchar(100)	-	nie	brak	-

- # pracownik\_id - pole jednoznacznie identyfikujące pracownika
- login
- haslo
- uprawnienia

### 4. Produkt

Nazwa	Typ	Atrybuty	Null	Ustawienia domyslnie	Dodatkowo
produkt_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
nazwa	varchar(50)	-	nie	brak	-
kategoria_id	int(10)	-	tak	null	-
opis	text	-	nie	brak	-
stan_magazyn	int(10)	unsigned	nie	0	-
cena	decimal(10,2)	unsigned	nie	brak	-
blokada	tinyint(1)	-	nie	0	-

- # produkt\_id pole jednoznacznie identyfikujące produkt
- nazwa - nazwa produktu wyświetlana klientowi na stronie internetowej, pakującemu produkt w raportach itd
- FK kategoria\_id id kategorii, do której jest przydzielany produkt, jeśli ma wartość NULL, produkt nie będzie nigdzie wyświetlany
- opis - opis produktu (HTML)
- stan\_magazyn - aktualny stan magazynowy wartość wprowadzana ręcznie i zmniejszana w momencie zatwierdzenia transakcji do realizacji
- cena
- blokada - wyświetlanie i sprzedaż produktu może być wstrzymana

### 5. produkt\_edycja

Nazwa	Typ	Atrybuty	Null	Ustawienia domyslne	Dodatkowo
produkt_edycja_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
produkt_id	int(10)	unsigned	nie	brak	-
kolumna	varchar(15)	-	nie	brak	-
poprz_wartosc	text	-	nie	brak	-
czas_edycji	timestamp	-	nie	CURRENT_TIMESTAMP	-
pracownik_id	int(10)	unsigned	nie	brak	-

- # produkt\_edycja\_id - pole jednoznacznie identyfikujące zmianę (edycję atrybutu lub dodanie) produktu, a więc konkretną wersję konkretnego produktu
- FK produkt\_id - id edytowanego produktu
- kolumna - nazwa edytowanego atrybutu lub informacja "dodano produkt"
- poprz\_wartosc - wartość atrybutu sprzed edycji lub wartość NULL gdy dodano nowy produkt
- czas\_edycji - czas wprowadzenia zmian
- FK pracownik\_id - id pracownika edytującego lub dodającego produkt

### 6. Transakcja

Nazwa	Typ	Atrybuty	Null	Ustawienia domyslne	Dodatkowo
transakcja_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
klient_id	int(10)	unsigned	tak	null	-
adres_wysylki	varchar(200)	-	nie	brak	-
kwota_wpłacona	decimal(12,2)	-	nie	0.00	-
status	enum ( 'u', 'a', 'p', 'z', 'w', 'd' )	-	nie	u	-

- # transakcja\_id pole jednoznacznie identyfikujące transakcję

- FK klient\_id - id zalogowanego klienta który utworzył transakcję lub NULL dla kupującego bez logowania
- adres\_wysylki - adres podany w formularzu dostawy transakcji
- kwota\_wplacona
- status - utworzono/potwierdzono(po wyświetleniu podsumowania transakcji)/zapłacono/wysłano/dostarczono

## 7. Transakcja\_produkty\_v

Nazwa	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
transakcja_id	int(10)	unsigned	nie	brak	-
sztuk	tinyint(3)	unsigned	nie	brak	-
produkt_edycja_id	int(10)	unsigned	nie	brak	-

- FK transakcja\_id
- sztuk
- FK produkt\_edycja\_id

### 1.1.3. Związki między elementami:

- Kategorie są wielopoziomowe (z podkategoriami)
- Produkt należy tylko do jednej (pod)kategorii
- Dane produktów mogą zmieniać się w każdej chwili
- Dla każdej transakcji należy przetrzymywać informacje o produkcie z momentu wyświetlenia podsumowania zamówienia klientowi
- Raport podsumowujący transakcję przed "zatwierdzam i płacę" = świętość, dane tam nie mogą ulec zmianie przez modyfikację parametrów produktu (w szczególności: cena) Bezwzględnie musi istnieć możliwość wyświetlenia stanu produktu:
  - nazwa
  - opis
  - cena

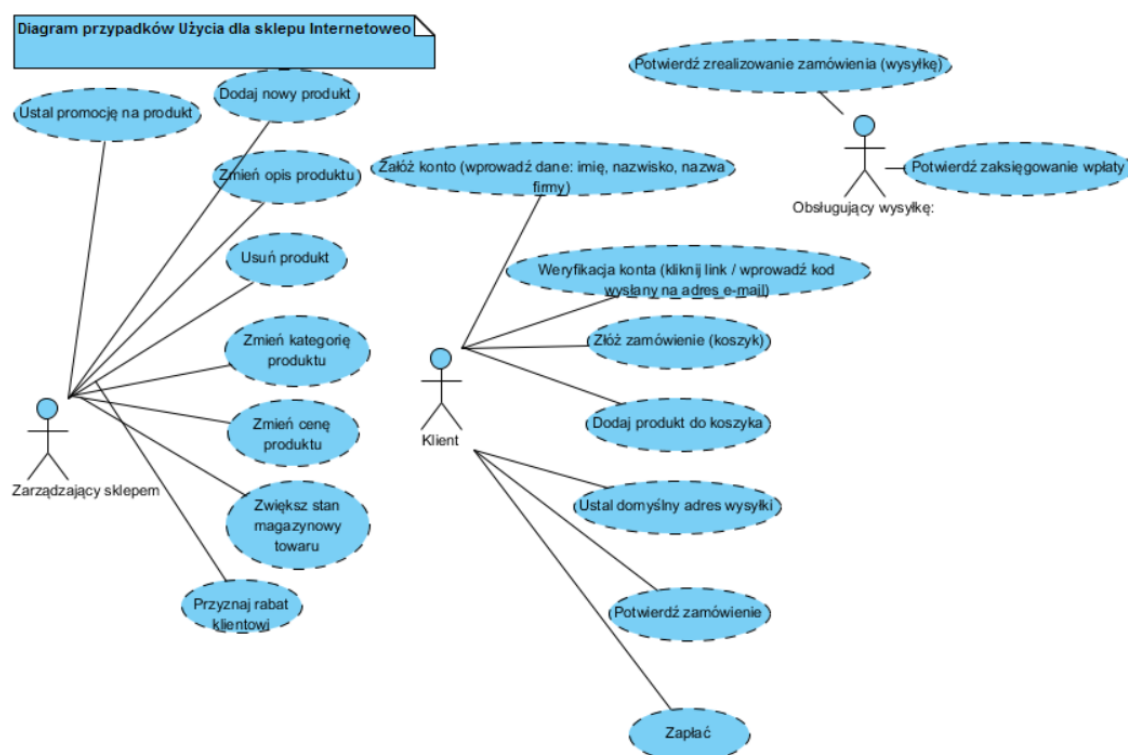
z momentu w którym klient dokonał zakupu

- Użytkownik niezalogowany ma możliwość zrobienia zakupów w sklepie internetowym

### 1.1.4. Diagram przypadków użycia

Diagram przypadków użycia (ang. use case) – graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi.

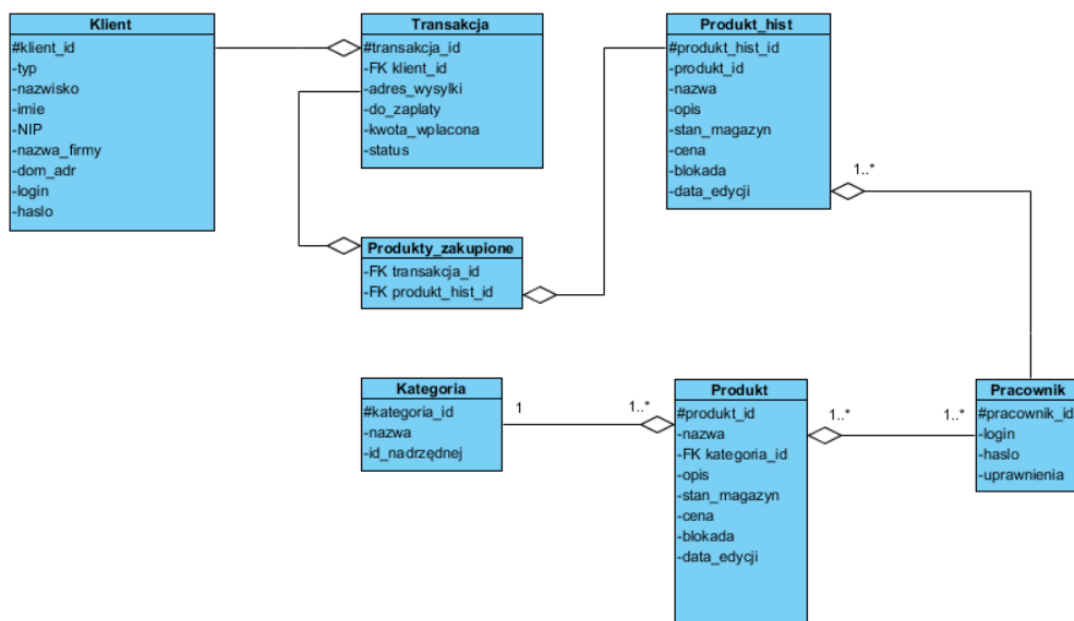
Diagram przypadków użycia tworzony jest w początkowych fazach modelowania. Diagram ten stanowi tylko przegląd możliwych działań w systemie, szczegóły ich przebiegu są modelowane za pomocą innych technik. Diagram przypadków użycia przedstawia usługi, które system świadczy aktorom, lecz bez wskazywania konkretnych rozwiązań technicznych.



Rys. 1.1. Diagram przypadków użycia



### 1.1.5. Diagram klas



Rys. 1.2. Diagram klas

### 1.1.6. Wyjściowa funkcjonalność

Sklep/serwis internetowy z możliwością założenia konta użytkownika.

Rodzaje kont w serwisie:

Co najmniej jedno konto z uprawnieniami administratora z zastrzeżeniem że nie może usunąć własnego konta (zabezpieczenia aby w serwisie istniało konto administratora)

Konta pracowników którym można nadać odpowiednie uprawnienia dodawanie/usuwanie kategorii/produktów generowanie raportów sprzedaży. Konto pracownika może posiadać wszystkie uprawnienia lub tylko wybrane. Konto pracownika może być utworzone/skasowane tylko przez innego pracownika z uprawnieniami do zarządzania kontami pracowników.

Konta użytkowników. Standardowe konta użytkowników. Użytkownik ma prawo edytowania/zmiany swoich danych osobowych. Do rozróżniania użytkowników wykorzystywane są ich unikalne loginy podawane przy rejestracji w serwisie. Zasady tworzenia loginów:

1. Login musi być unikalny w bazie
2. Nie jest rozróżniana wielkość znaków w loginie np. ADMIN, admin, AdMin - traktowany jest jak jeden login

Loginy są zapisywane w bazie, raz wykorzystany login nie może być użyty ponownie.

Użytkownik ma możliwość usunięcia konta z serwisu. w takim przypadku muszą zostać usunięte wszystkie jego dane osobowe. Zachowane zostają dane o przeprowadzonych z tym klientem transakcjach bez możliwości jego identyfikacji.

## 1.2. Definiowanie tabel, więzów integralności, perspektyw

### 1.2.1. Procedury

#### 1. kategoria\_lista\_podrzecznych

```
BEGIN
DECLARE n INT;
SET n=1;
CREATE TEMPORARY TABLE podrzedne ('kategoria\_id' int(10) unsigned NOT NULL, 'nazwa'
'id\_nadrzednej' int(10) unsigned, 'przetworzona' tinyint(1) DEFAULT 0);
CREATE TEMPORARY TABLE podrzedne2 ('kategoria\_id' int(10) unsigned NOT NULL, 'nazwa'
'id\_nadrzednej' int(10) unsigned, 'przetworzona' tinyint(1) DEFAULT 0);

INSERT INTO podrzedne SELECT kategoria\_id, nazwa, id\_nadrzednej, 0 FROM kategoria;

WHILE n>0 DO
INSERT INTO podrzedne2 SELECT kategoria\_id, nazwa, id\_nadrzednej, 0 FROM kategoria;
UPDATE podrzedne SET przetworzona=1;
INSERT INTO podrzedne SELECT * FROM podrzedne2;
TRUNCATE TABLE podrzedne2;
SELECT COUNT(*) FROM podrzedne WHERE przetworzona=0 INTO n;
END WHILE;

SELECT * FROM podrzedne;

DROP TEMPORARY TABLE podrzedne;
DROP TEMPORARY TABLE podrzedne2;
END
```

#### 2. menu\_drzewo\_kat

```
BEGIN
DECLARE kat\_id INT;
IF pkat\_id IS NOT NULL THEN
SET kat\_id = pkat\_id;
WHILE kat\_id IS NOT NULL DO
SELECT * FROM kategoria WHERE id\_nadrzednej=kat\_id;
SELECT id\_nadrzednej FROM kategoria WHERE kategoria\_id=kat\_id INTO kat\_id;
END WHILE;
END IF;
SELECT * FROM kategoria WHERE id\_nadrzednej IS NULL;
END
```

## 3. produkt\_by\_kategoria

```
BEGIN
DECLARE n INT;
SET n=1;
CREATE TEMPORARY TABLE podrzedne ('kategoria\_id' int(10) unsigned NOT NULL, 'prze
CREATE TEMPORARY TABLE podrzedne2 ('kategoria\_id' int(10) unsigned NOT NULL, 'prze

INSERT INTO podrzedne VALUES(id\_kat,0);

WHILE n>0 DO
INSERT INTO podrzedne2 SELECT kategoria\_id, 0 FROM kategoria WHERE id\_nadrzedne
UPDATE podrzedne SET przetworzona=1;
INSERT INTO podrzedne SELECT * FROM podrzedne2;
TRUNCATE TABLE podrzedne2;
SELECT COUNT(*) FROM podrzedne WHERE przetworzona=0 INTO n;
END WHILE;

SELECT p.produkt\_id, p.nazwa, p.kategoria\_id, k.nazwa AS nazwa\_kat FROM produ

DROP TEMPORARY TABLE podrzedne;
DROP TEMPORARY TABLE podrzedne2;
END
```

## 4. produkt\_hist

```
BEGIN
DECLARE vprodukt\_id INT;
DECLARE poprz\_nazwa VARCHAR(50);
DECLARE poprz\_cena DECIMAL(10,2);
DECLARE poprz\_opis TEXT;
DECLARE poprz\_blokada TINYINT(1);
SELECT produkt\_id INTO vprodukt\_id FROM produkt\_edycja WHERE produkt\_edycja\_
SELECT poprz\_wartosc INTO poprz\_nazwa FROM produkt\_edycja WHERE produkt\_edycja
SELECT poprz\_wartosc INTO poprz\_cena FROM produkt\_edycja WHERE produkt\_edycja
SELECT poprz\_wartosc INTO poprz\_opis FROM produkt\_edycja WHERE produkt\_edycja
SELECT poprz\_wartosc INTO poprz\_blokada FROM produkt\_edycja WHERE produkt\_edy

SELECT COALESCE(poprz\_nazwa,nazwa) AS nazwa, COALESCE(poprz\_cena,cena) AS cena,
END
```

**1.2.2. Funkcje**

Funkcje wykorzystane w projektowanej bazie danych

## 1. ostatnia\_produkt\_edycja\_id

```
RETURN (SELECT MAX('produkt\_edycja\_id') FROM 'produkt\_edycja' WHERE 'produkt\_
```

## **Interfejs - strona internetowa**

# Bibliografia

- [1] Balcerzak J., Pansiuk J.: *Wprowadzenie do kartografii matematycznej*, Warszawa, OWPW 2005.

# Spis rysunków

1.1. Diagram przypadków uycia . . . . .	8
1.2. Diagram klas . . . . .	9