Bazy danych

Dokumentacja Projektu

pt.: „Sklep muzyczny”

|  |  |
| --- | --- |
| **Data wykonania**: 21.01.2020 | **Grupa**: L03  Miłosz Stączek |

Spis treści

[Wstęp teoretyczny 3](#_Toc30443270)

[1. Cel i założenia projektu 4](#_Toc30443271)

[2. Wstępny opis projektu 4](#_Toc30443272)

[3. Wykorzystane technologie 5](#_Toc30443273)

[4. Diagram związków encji bazy danych 6](#_Toc30443274)

[5. Opis tabel bazy danych i ich funkcji 7](#_Toc30443275)

[6. Funkcje bazy danych realizowane przez SQL 8](#_Toc30443276)

# Wstęp teoretyczny

Baza danych jest to zbiór uporządkowanych informacji przechowywanych w systemie komputerowym. Bazą danych steruje się za pomocą Systemu zarządzania bazami danych (DBMS). DBMS stanowi interfejs między bazą danych a jego użytkownikami. Dane zawarte   
w bazie danych, system DBMS oraz powiązane z nimi aplikacje tworzą system bazodanowy. Dane umieszczane są w wierszach i kolumnach tabel, co umożliwia łatwiejsze zarządzanie danymi, ich wyszukiwanie oraz modyfikowanie. Większość baz danych wykorzystuje język strukturalny SQL (Strukturę Query Language) do tworzenia zapytań do bazy. Rozróżnia się kilka typów baz danych m.in. relacyjne, obiektowe, rozproszone.

# Cel i założenia projektu

Celem projektu było przedstawienie zakresu i tematyki projektu, zagadnień związanych   
z tematem, określenie funkcji bazy danych oraz ich priorytetu, wybór narzędzi i technologii do realizacji projektu. Prezentacja opisu tabel bazy danych oraz prezentacja Diagramu związków encji. Implementacja bazy danych, prezentacja jej funkcjonalności oraz funkcji realizowanych przez SQL, dostarczenie repozytorium wraz z dokumentacją projektu.

# Wstępny opis projektu

Tematem projektu będzie system obsługujący stacjonarny sklep muzyczny sprzedający sprzęt oraz albumy muzyczne. System będzie się zajmował przechowywaniem danych personalnych klientów, pracowników jak również ich adresy. System będzie pozwalał na dodawanie, usuwanie i modyfikację danych osobistych klientów i pracowników. Możliwe będzie przypisywanie klientom zakupionych przez nich towarów, klienci mogą podać swoje dane personalne jednak nie muszą tego robić, aby zakupić dany produkt. System będzie przede wszystkim przypisywał odpowiednie informacje dotyczące tego jaki produkt został kupiony i w jakiej ilości, ile danego towaru pozostało dostępne na magazynie oraz jaki pracownik sprzedał dany towar. Tabele, które planuję umieścić w projekcie: klienci, pracownicy, produkty, adresy, sprzedaz. Będzie to relacyjna baza danych. Będę korzystał   
z narzędzia phpMyAdmin do stworzenia i obsługi systemu zarządzania bazami danych MySQL.

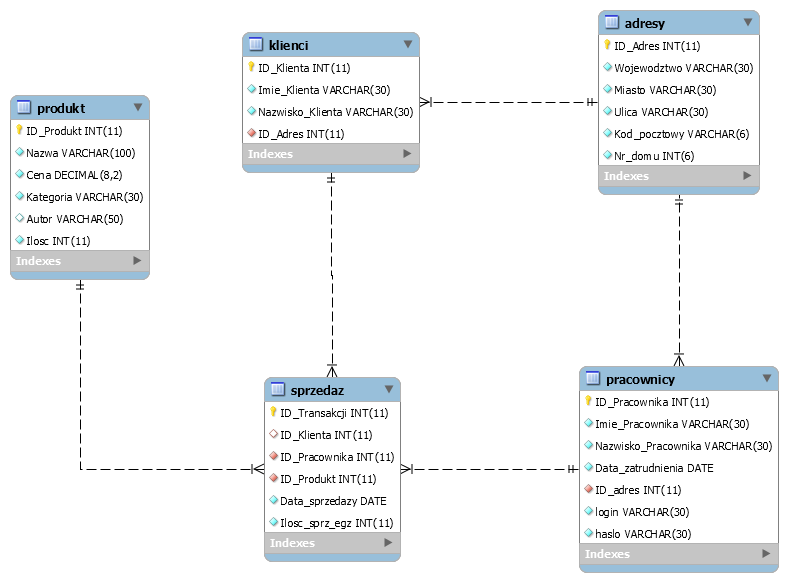
# Wykorzystane technologie

W projekcie wykorzystano darmowy pakiet XAMPP do postawienia serwera oraz obsługi phpMyAdmin. Jest to wieloplatformowy, zintegrowany pakiet, składający się głównie   
z serwera Apache, bazy danych MySQL i interpreterów dla skryptów napisanych w PHP   
i Perlu. Pakiet jest wydawany na licencji GNU jako darmowy serwer WWW do obsługi dynamicznych stron. Obecnie XAMPP jest dostępny na cztery platformy: Microsoft Windows, Linux, Sun Solaris oraz OS X.

phpMyAdmin jest to narzędzie pozwalające w łatwy sposób zarządzać bazą danych, napisane w języku PHP wydawane na licencji GNU. Prace nad phpMyAdmin rozpoczął Tobias Ratschiller w 1998 roku, autor jednak zawiesił prace nad programem i projekt został przejęty przez trzech innych programistów i we wrześniu 2000 roku doczekał się wydania kolejnej wersji. Od tego czasu program jest nieprzerwanie rozwijany.

MySQL jest to system zarządzania relacyjnymi bazami danych, rozwijany obecnie przez firmę Oracle. MySQL sprawdza się zarówno dla małych jak i dużych projektów, jest dobrze zoptymalizowany pod kątem szybkości oraz jest łatwy w użytkowaniu, używa standardowego języka SQL. Dane przechowywane są w kolumnach i wierszach tabel połączonych relacjami.

# Diagram związków encji bazy danych



Rysunek Diagram Związków encji

# Opis tabel bazy danych i ich funkcji

1. Klienci - Tabela "Klienci" odpowiedzialna jest za przechowywanie informacji o klientach, którzy byli chętni do podania swoich danych osobowych. Imię, Nazwisko i Adres jako klucz obcy. Każdy klient, który podał dane musi mieć przypisany adres.
2. Pracownicy - Tabela "Pracownicy" przechowuje informacje o pracownikach pracujących w danym sklepie. Imię, Nazwisko, Adres jako klucz obcy oraz datę zatrudnienia. Każdy pracownik musi mieć przypisany adres.
3. Adresy - Tabela "Adresy" przechowuje adresy klientów i pracowników. Adres przypisywany jest do Klienta lub Pracownika. Zawiera pola województwo, miasto, ulica, kod\_pocztowy, nr\_domu.
4. Sprzedaz - Tabela "Sprzedaz" zawiera dane łączące Pracownika, Produkt, który został sprzedany, ewentualnego klienta (jeśli ten zgodził się podać swoje dane), datę sprzedaży oraz ilość sprzedanych egzemplarzy danego produktu. Tabela ma za zadanie gromadzić informacje o tym, ile jakich produktów zostało sprzedanych o jakim czasie.
5. Produkt - Tabela "Produkt" zawiera informacje o produkcie, jego nazwę, kategorię, do której należy produkt, ewentualnego autora danego produktu (jeśli ten istnieje np. autor książki), cenę produktu oraz ilość dostępnych egzemplarzy danego produktu.

# Funkcje bazy danych realizowane przez SQL

1. Zapytanie wyszukująca gitary

SELECT \* FROM produkt

WHERE produkt.Nazwa LIKE '%gitara%';

Zapytanie wyświetli informacje o wszystkich rekordach z tabeli produkt które w nazwie zawierają słowo gitara.



Rysunek Wszystkie pola z gitarą w nazwie

1. Zapytanie modyfikujące nazwę konkretnego produktu

UPDATE produkt SET nazwa ='Whitestar'

WHERE produkt.ID\_Produkt=101;

Zapytanie, które modyfikuje podaną wartość pola (w tym przypadku pole „nazwa”), dla rekordu, w którym spełniony jest podany warunek (w tym przypadku rekord z ID =1).

1. Zapytanie usuwające konkretny rekord

DELETE FROM produkt WHERE ID\_Produkt='103';

Zapytanie usuwa podany rekord z podanej tabeli. W tym przypadku usuwany jest rekord z tabeli „produkt” z ID równym 3.

1. Zapytania dodające klienta i pracownika

INSERT INTO adresy(ID\_Adres,Wojewodztwo,Miasto,Ulica,Kod\_pocztowy,Nr\_domu)

VALUES (13,'Podkarpackie','Jaslo','Robotnicza','32-144','18');

INSERT INTO klienci (ID\_Klienta,Imie\_Klienta,Nazwisko\_Klienta,ID\_Adres)

VALUES (13,'Przemyslaw','Pioro', 13);

Pierwsze zapytanie dodaje nowy rekord do tabeli adresy uzupełniając podane pola (ID\_Adres, Wojewodztwo, Miasto, Ulica, Kod\_pocztowy, Nr\_domu) podanymi wartościami (13, Podkarpackie, Jaslo, Robotnicza, 32-144, 18). Następne zapytanie dodaje nowy rekord do tabeli klienci uzupełniając pola w taki sam sposób jak poprzednie.

INSERT INTO adresy (ID\_Adres,Wojewodztwo,Miasto,Ulica,Kod\_pocztowy,Nr\_domu)

VALUES (85,'Podkarpackie','Przemysl','Parkowa','36-654','54');

INSERT INTO pracownicy (ID\_Pracownika,Imie\_Pracownika,Nazwisko\_pracownika, Data\_zatrudnienia, id\_adres, login, haslo)

VALUES (85,'Kazimierz', 'Zajac', '2020/01/01',85,'login5', 'haslo5');

Zapytania dodają informacje dotyczące nowego pracownika, takie jak jego adres oraz jego dane personalne.

1. Zapytanie wypisujące adresy klientów i pracowników

SELECT klienci.Imie\_Klienta, klienci.Nazwisko\_Klienta, adresy.Miasto, adresy.Ulica, adresy.Nr\_domu

FROM klienci

INNER JOIN adresy ON klienci.ID\_Adres = adresy.ID\_Adres

ORDER BY klienci.Nazwisko\_Klienta asc;

Zapytanie, które wypisuje z bazy danych imię i nazwisko klienta z tabeli „klienci” a także informacje na temat adresu zamieszkania (miasto, ulica, nr domu) z tabeli „adresy”. Zapytanie sortuje alfabetycznie po nazwisku klienta.



Rysunek Spis klientów wraz z ich miejscem zamieszkania

SELECT pracownicy.Imie\_Pracownika, pracownicy.Nazwisko\_Pracownika, adresy.Miasto, adresy.Ulica, adresy.Nr\_domu

FROM pracownicy

INNER JOIN adresy ON pracownicy.ID\_adres = adresy.ID\_Adres

ORDER BY pracownicy.Nazwisko\_Pracownika asc;

Zapytanie wypisująca informacje o pracownikach oraz ich adresy.



Rysunek Spis pracowników wraz z ich adresem zamieszkania

1. Zapytania wyświetlające raport ze sprzedaży

SELECT

sprzedaz.ID\_Transakcji AS ID, produkt.Nazwa,

CONCAT (klienci.Imie\_Klienta ,' ', klienci.Nazwisko\_Klienta) AS 'Klient',

CONCAT (pracownicy.Imie\_Pracownika ,' ', pracownicy.Nazwisko\_Pracownika) AS 'Pracownik',

sprzedaz.Data\_sprzedazy,

sprzedaz.Ilosc\_sprz\_egz AS Ilosc,

CASE WHEN klienci.ID\_Klienta IS NOT NULL

THEN ((produkt.Cena \* sprzedaz.Ilosc\_sprz\_egz) \* 0.9)

ELSE (produkt.Cena \* sprzedaz.Ilosc\_sprz\_egz)

END AS Wartosc

FROM sprzedaz

INNER JOIN produkt ON sprzedaz.ID\_Produkt = produkt.ID\_Produkt

LEFT JOIN klienci ON sprzedaz.ID\_Klienta = klienci.ID\_Klienta

INNER JOIN pracownicy ON sprzedaz.ID\_Pracownika = pracownicy.ID\_Pracownika

ORDER BY ID\_Transakcji ASC;

Zapytania wyświetlają wszystkie przeprowadzone transakcje. Wyświetlana jest nazwa zakupionego produktu, klient, który zakupił dany towar (o ile ten zgodził się podać swoje dane osobowe i figuruje w systemie). Pracownik, który przeprowadził daną transakcję, datę sprzedaży, ilość sztuk sprzedanego towaru oraz wartość końcową danej transakcji. Zapytania liczą końcową wartość mnożąc ilość zakupionych sztuk przez cenę jednostkową danego towaru. Jeśli transakcji dokonał klient, który figuruje w systemie zostanie mu naliczony rabat w wysokości 10% od ceny końcowej. Zapytanie sortuje po ID transakcji.



Rysunek Raport ze sprzedaży

1. Zapytanie zliczające wartości zamówień dla poszczególnych klientów

SELECT

COUNT(sprzedaz.ID\_Transakcji) AS Ilosc\_Transakcji,

CONCAT (klienci.Imie\_Klienta, ' ' ,klienci.Nazwisko\_Klienta) AS 'Klient',

SUM(produkt.Cena \* sprzedaz.Ilosc\_sprz\_egz \* 0.9) AS Wartosc

FROM sprzedaz

INNER JOIN produkt ON sprzedaz.ID\_Produkt = produkt.ID\_Produkt

INNER JOIN klienci ON sprzedaz.ID\_Klienta = klienci.ID\_Klienta

GROUP BY klienci.ID\_Klienta

ORDER BY klienci.ID\_Klienta;

Zapytania zliczają, ile transakcji wykonał dany klient, wyświetla jego imię i nazwisko, oraz wartość całkowitą jego dotychczasowych transakcji. Transakcje anonimowe nie zostają uwzględnione. Zapytania sortują spis po ID klienta.



Rysunek Ilość oraz wartość całkowita transakcji klientów

1. Zapytania pozwalające sprzedać dany produkt.

SELECT ID\_Produkt AS ID, Nazwa, Cena, Ilosc FROM produkt WHERE Nazwa LIKE'%ziggy%';

Powyższe zapytanie pokazuje produkt, który ma zostać sprzedany. Wyświetlana jest jego ID, nazwa, cena jednostkowa oraz ilość dostępnych sztuk danego produktu. Służy ono do sprawdzenia, ile towary danego typu jest dostępnego w danym momencie w sklepie.



Rysunek 7 Informacje na temat konkretnego produktu

INSERT INTO

sprzedaz(ID\_Klienta,ID\_Pracownika,ID\_Produkt,Data\_sprzedazy,Ilosc\_sprz\_egz)

VALUES

(NULL, 82, 102, CURRENT\_DATE, 1);

UPDATE produkt, sprzedaz

SET produkt.Ilosc = (produkt.Ilosc - sprzedaz.Ilosc\_sprz\_egz)

WHERE sprzedaz.ID\_Transakcji=LAST\_INSERT\_ID()

AND produkt.ID\_Produkt = sprzedaz.ID\_Produkt;

Zapytanie INSERT INTO pozwala dodać nowy rekord do tabeli „sprzedaz”. Dodany rekord zawiera id transakcji (dodawany automatycznie poprzez funkcję auto\_increment),  
id klienta (o ile ten zgodził się podać swoje dane osobowe – w tym przypadku NULL – sprzedaż anonimowa) id pracownika, id produktu, obecną datę oraz ilość kupowanych podczas tej transakcji egzemplarzy.

Zapytanie UPDATE modyfikuje wartość pola „ilosc” w tabeli „produkt” odejmując ilość sprzedanych w danej transakcji od ilości całkowitej dostępnej w danym momencie w sklepie.



Rysunek Nowo dodana sprzedaż



Rysunek Zmodyfikowane pole "ilosc" po wykonaniu zapytań