Sklep płytowy

Ewa Lachowicz, 6SID, 225607

Piotr Jastrzębski, 6ISI, 225594

# Opis słowny projektu

System będzie umożliwiał składanie zamówień na płyty CD i mp3 zamieszczone w bazie sklepu. Każda piosenka jest przypisana do wykonawcy i kategorii, opcjonalnie do albumu. Użytkownik będzie miał możliwość składania zamówień i obsługi koszyka zakupów, a także komentowania piosenek i tworzenia własnej listy ulubionych utworów.

# Wymagania funkcjonalne

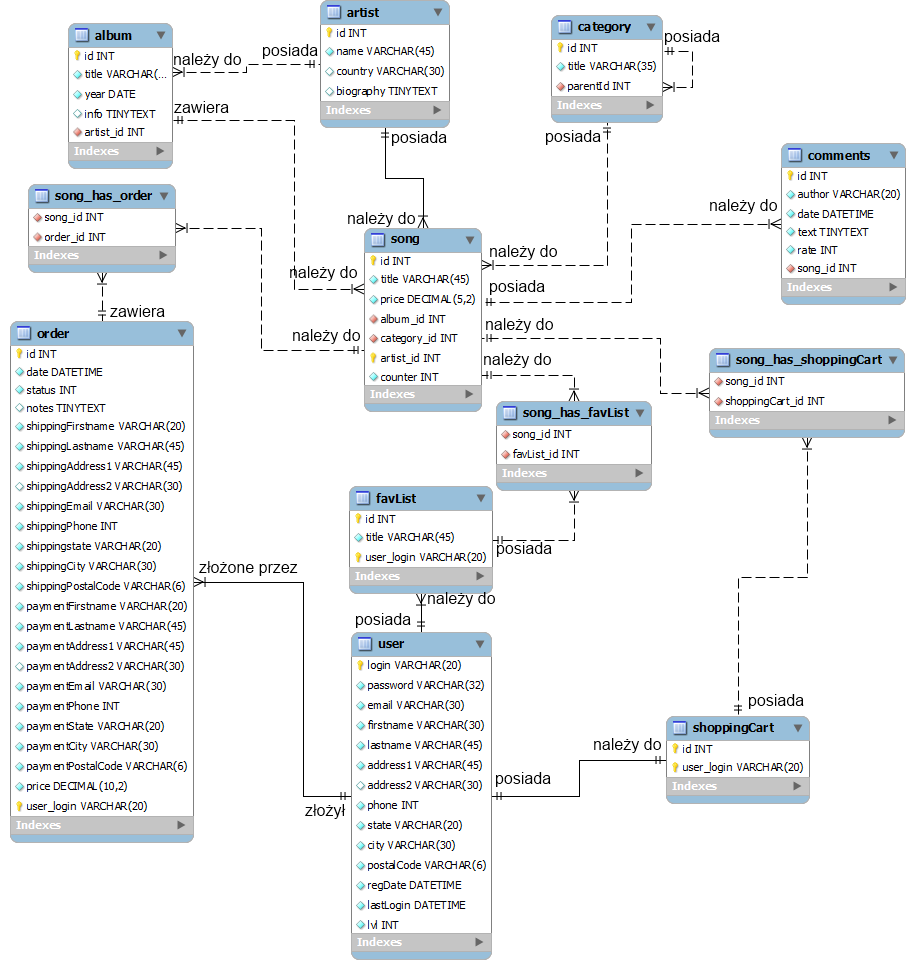
## User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Codename | Nazwa Wymagania | Opis Wymagania | Priorytet |
| 1 | ADD\_TO\_SC | Dodanie CD do koszyka zamówienia | Użytkownik wybiera utwory z katalogu i dodaje je do koszyka zamówienia | 1 |
| 2 | DELETE\_FROM\_SC | Usunięcie CD z koszyka zamówienia | Użytkownik usuwa wybrany utwór z koszyka zamówienia | 2 |
| 3 | VIEW\_SC | Pokazanie koszyka zamówienia | Użytkownik ogląda zawartość koszyka zamówienia | 1 |
| 4 | ORDER\_SONG | Zamówienie utworów | Użytkownik składa zamówienie w sklepie na wybrane wcześniej do koszyka utwory | 1 |
| 5 | MAKE\_COMMENT | Dodanie komentarza | Użytkownik dodaje komentarz do dowolnego CD | 3 |
| 6 | CREATE\_FL | Utworzenie listy ulubionych | Użytkownik tworzy listę ulubionych | 3 |
| 7 | ADD\_TO\_FL | Dodanie CD do listy ulubionych | Użytkownik dodaje utwór do listy ulubionych | 3 |
| 8 | DELETE\_FROM\_FL | Usunięcie CD z listy ulubionych | Użytkownik usuwa utwór z listy ulubionych | 3 |
| 9 | ADD\_USER | Założenie konta użytkownika | Użytkownik zakłada sobie nowe konto w systemie | 2 |
| 10 | EDIT\_USER | Edycja konta użytkownika | Użytkownik edytuje swoje dane w systemie | 3 |
| 11 | DEL\_USER | Usunięcie konta użytkownika | Użytkownik usuwa swoje konto z systemu | 3 |
| 12 | SEARCH | Wyszukanie albumu | Użytkownik wyszukuje album, artystę, utwór podając frazę w wyszukiwarce | 3 |
| Admin | | | | |
| 1 | ADD\_ARTIST | Dodanie artysty | Administrator dodaje nowego artystę do bazy | 1 |
| 2 | ADD\_SONG | Dodanie CD | Administrator dodaje nowy utwór do bazy | 1 |
| 3 | ADD\_ALBUM | Dodanie albumu | Administrator dodaje do bazy nowy album artysty |  |
| 4 | DELETE\_SONG | Usunięcie CD | Administrator usuwa utwór z bazy | 2 |
| 5 | DELETE\_ARTIST | Usunięcie artysty | Administrator usuwa artystę z bazy. Automatycznie usuwane są wszystkie utwory z nim powiązane. | 2 |
| 6 | DELETE\_ALBUM | Usunięcie albumu | Administrator usuwa album z bazy.  Automatycznie usuwane są wszystkie utwory powiązane z albumem. |  |
| 7 | EDIT\_USER | Edycja konta użytkownika | Użytkownik edytuje swoje dane w systemie | 3 |
| 6 | DEL\_USER | Usunięcie konta użytkownika | Użytkownik usuwa swoje konto z systemu | 3 |
| 8 | VIEW\_ORDERS | Wyświetlenie zamówień | Administrator wyświetla wszystkie zamówienia w systemie | 2 |
| 9 | DEL\_COMM | Usunięcie komentarzy | Administrator usuwa komentarze do piosenek | 3 |
| 10 | EDIT\_COMM | Edycja komentarzy | Administrator edytuje komentarze do piosenek | 3 |

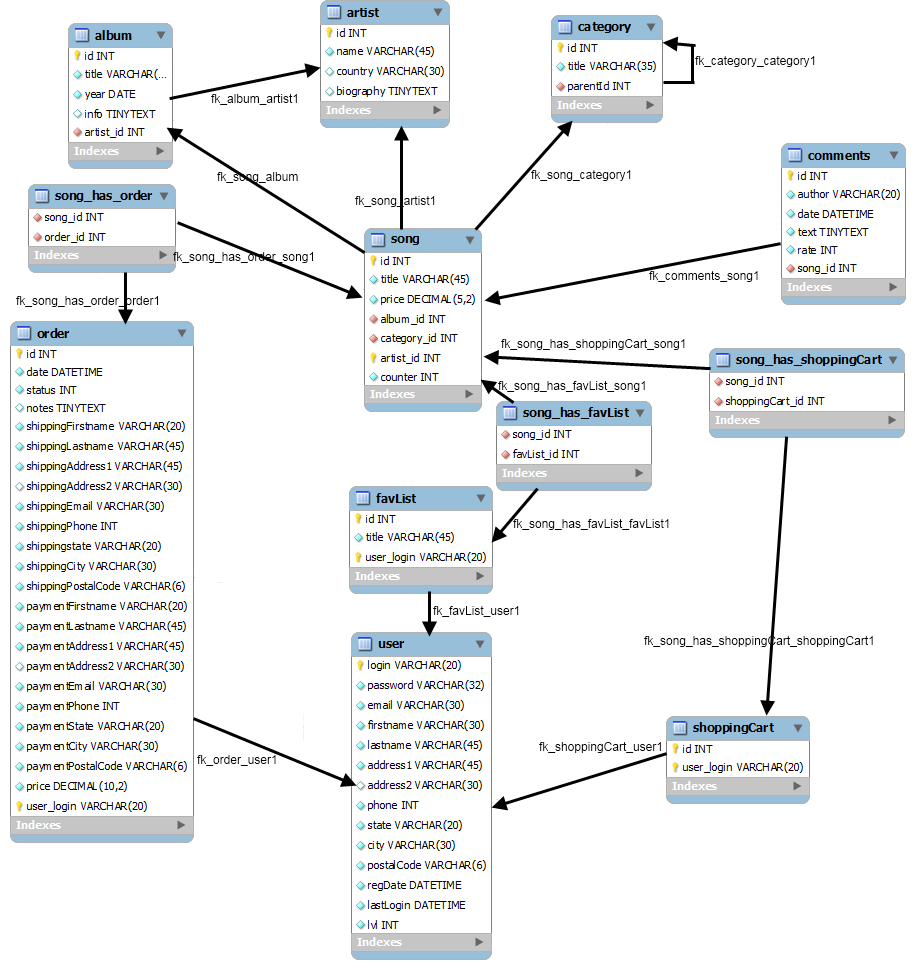
# Wymagania niefunkcjonalne

* brak możliwości dostępu użytkowników do danych/funkcji dostępnych dla użytkowników innego typu
* autoryzacja użytkowników i administratora
* czas reakcji systemu na zapytanie nie powinien przekraczać pięć sekund
* system powinien być dostępny dla użytkowników przez całą dobę. System powinien być jak najmniej podatny na awarie. Dopuszczalne są krótkotrwałe przerwy techniczne. W przypadku awarii, sprawność systemu powinna zostać przywrócona w czasie do ośmiu godzin
* system powinien kontrolować wprowadzane dane na każdym kroku
* system powinien być dobrze zabezpieczony przed wyciekiem danych użytkowników. Tylko administrator ma dostęp do danych użytkowników. Użytkownik nie może mieć możliwości uzyskania w niekontrolowany sposób praw administratora
* docelowym SZBD jest MySQL

# Diagram ER

****

# Diagram fizyczny

****

# Opis encji

## Album:

CREATE TABLE IF NOT EXISTS `mydb`.`album` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`title` VARCHAR(60) NOT NULL ,

`year` DATE NOT NULL ,

`info` TINYTEXT NULL ,

`artist\_id` INT NOT NULL ,

PRIMARY KEY (`id`) ,

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) ,

INDEX `title\_INDEX` (`title` ASC) ,

INDEX `fk\_album\_artist1` (`artist\_id` ASC) ,

CONSTRAINT `fk\_album\_artist1`

FOREIGN KEY (`artist\_id` )

REFERENCES `mydb`.`artist` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny albumu, klucz główny

Title – tytuł albumu, nałożony index dla szybszego wyszukiwania

Year – rok wydania albumu

Info – dodatkowe informacje o albumie

**Klucze i indexy:**

* Id – klucz główny, unikalny
* Title – index ze względu na możliwość wyszukiwania albumu po tytule

## Category:

CREATE TABLE IF NOT EXISTS `mydb`.`category` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`title` VARCHAR(35) NOT NULL ,

`parentId` INT ,

PRIMARY KEY (`id`) ,

INDEX `fk\_category\_category1` (`parentId` ASC) ,

CONSTRAINT `fk\_category\_category1`

FOREIGN KEY (`parentId` )

REFERENCES `mydb`.`category` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny kategorii, klucz główny

Title – tytuł kategorii

parentId – numer identyfikacyjny kategorii rodzica, dla stworzenia struktury drzewiastej w bazie

**Klucze i indexy:**

* Id – klucz główny, unikalny
* parentId – klucz obcy w tej samej tabeli – do stworzenia struktury drzewiastej kategorii

## Artist:

CREATE TABLE IF NOT EXISTS `mydb`.`artist` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`name` VARCHAR(45) NOT NULL ,

`country` VARCHAR(30) NULL ,

`biography` TINYTEXT NULL ,

PRIMARY KEY (`id`) ,

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) ,

INDEX `name\_INDEX` (`name` ASC)

)

Id – numer identyfikacyjny artysty, klucz główny

Name – nazwa artysty

Country – kraj pochodzenia artysty

Biography – krótka biografia artysty

**Klucze i indexy:**

* Id – klucz główny, unikalny
* Name – index ze względu na możliwość wyszukiwania artystów

## Song:

CREATE TABLE IF NOT EXISTS `mydb`.`song` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`title` VARCHAR(45) NOT NULL ,

`price` DECIMAL(5,2) NOT NULL DEFAULT 0.00 ,

`album\_id` INT NOT NULL ,

`category\_id` INT NOT NULL ,

`artist\_id` INT NOT NULL ,

`counter` INT NOT NULL DEFAULT 0 ,

PRIMARY KEY (`id`, `artist\_id`) ,

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) ,

INDEX `fk\_song\_album` (`album\_id` ASC) ,

INDEX `fk\_song\_category1` (`category\_id` ASC) ,

INDEX `fk\_song\_artist1` (`artist\_id` ASC) ,

INDEX `title\_INDEX` (`title` ASC) ,

CONSTRAINT `fk\_song\_album`

FOREIGN KEY (`album\_id` )

REFERENCES `mydb`.`album` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_song\_category1`

FOREIGN KEY (`category\_id` )

REFERENCES `mydb`.`category` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_song\_artist1`

FOREIGN KEY (`artist\_id` )

REFERENCES `mydb`.`artist` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id - numer identyfikacyjny piosenki, klucz główny

Title – tytuł CD

Price – cena CD

Album\_Id – wskazanie na przynależność do albumu,

Category\_Id – wskazanie na przynależność do kategorii

Artist\_Id – wskazanie na przynależność do artysty

Counter – licznik zakupień CD

**Klucze i indexy:**

* Id – klucz główny, unikalny
* Title – index ze względu na możliwość wyszukiwania piosenek
* Album\_id – klucz obcy w tabeli Album
* Category\_id – klucz obcy w tabeli Category
* Artist\_id – klucz obcy w tabeli Artist

## Comments:

CREATE TABLE IF NOT EXISTS `mydb`.`comments` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`author` VARCHAR(20) NOT NULL DEFAULT 'Anonymous' ,

`date` DATETIME NOT NULL ,

`text` TINYTEXT NOT NULL ,

`rate` INT NOT NULL DEFAULT 3 ,

`song\_id` INT NOT NULL ,

PRIMARY KEY (`id`) ,

INDEX `fk\_comments\_song1` (`song\_id` ASC) ,

CONSTRAINT `fk\_comments\_song1`

FOREIGN KEY (`song\_id` )

REFERENCES `mydb`.`song` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny komentarza, klucz główny

Author – autor komentarza

Date – data dodania komentarza

Text – treść komentarza

Rate – ocena dla piosenki

Song\_id – wskazanie na przynależność do piosenki

**Klucze i indexy:**

* Id – klucz główny, unikalny
* Song\_id – klucz obcy w tabeli Song

## User:

CREATE TABLE IF NOT EXISTS `mydb`.`user` (

`login` VARCHAR(20) NOT NULL ,

`password` VARCHAR(32) NOT NULL ,

`email` VARCHAR(30) NOT NULL ,

`firstname` VARCHAR(30) NOT NULL ,

`lastname` VARCHAR(45) NOT NULL ,

`address1` VARCHAR(45) NOT NULL ,

`address2` VARCHAR(30) NULL ,

`phone` INT NOT NULL ,

`state` VARCHAR(20) NOT NULL ,

`city` VARCHAR(30) NOT NULL ,

`postalCode` VARCHAR(6) NOT NULL ,

`regDate` DATETIME NOT NULL ,

`lastLogin` DATETIME NOT NULL ,

`lvl` INT NOT NULL DEFAULT 1 ,

PRIMARY KEY (`login`) ,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) ,

INDEX `password\_INDEX` (`password` ASC) ,

INDEX `lvl\_INDEX` (`lvl` ASC)

)

Login – nazwa użytkownika, klucz główny

Password – hasło użytkownika, w bazie trzymany hash md5

Email – adres email użytkownika

Firstname – imię użytkownika

Lastname – nazwisko użytkownika

Address1, address2, phone, state, city, postalCode – dane teleadresowe

RegDate – data rejestracji użytkownika

LastLogin – data ostatniego logowania użytkownika

Lvl – poziom uprawnień użytkownika

**Klucze i indexy:**

* Login – klucz główny, unikalny
* Email – pole o unikalnej wartości – nie może być dwóch użytkowników w bazie z identycznym adresem email
* Password – index ze względu na dopasowywanie hasła przy autoryzacji użytkownika
* Lvl – index ze względu na częste sprawdzanie poziomu uprawnień użytkownika

## FavList:

CREATE TABLE IF NOT EXISTS `mydb`.`favList` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`title` VARCHAR(45) NOT NULL ,

`user\_login` VARCHAR(20) NOT NULL ,

PRIMARY KEY (`id`, `user\_login`) ,

INDEX `fk\_favList\_user1` (`user\_login` ASC) ,

CONSTRAINT `fk\_favList\_user1`

FOREIGN KEY (`user\_login` )

REFERENCES `mydb`.`user` (`login` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny listy, klucz główny

Title – nazwa listy

User\_login – wskazanie na przynależność do użytkownika

**Klucze i indexy:**

* Id, user\_login – klucz główny, unikalny
* User\_login – klucz obcy w tabeli User

## ShoppingCart:

CREATE TABLE IF NOT EXISTS `mydb`.`shoppingCart` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`user\_login` VARCHAR(20) NOT NULL ,

PRIMARY KEY (`id`, `user\_login`) ,

INDEX `fk\_shoppingCart\_user1` (`user\_login` ASC) ,

CONSTRAINT `fk\_shoppingCart\_user1`

FOREIGN KEY (`user\_login` )

REFERENCES `mydb`.`user` (`login` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny koszyka zamówienia, klucz główny

User\_login – wskazanie na użytkownika

**Klucze i indexy:**

* Id, user\_login – klucz główny, unikalny
* User\_login – klucz obcy w tabeli User

## Order:

CREATE TABLE IF NOT EXISTS `mydb`.`order` (

`id` INT NOT NULL AUTO\_INCREMENT ,

`date` DATETIME NOT NULL ,

`status` INT NOT NULL DEFAULT 0 ,

`notes` TINYTEXT NULL ,

`shippingFirstname` VARCHAR(20) NOT NULL ,

`shippingLastname` VARCHAR(45) NOT NULL ,

`shippingAddress1` VARCHAR(45) NOT NULL ,

`shippingAddress2` VARCHAR(30) NULL ,

`shippingEmail` VARCHAR(30) NOT NULL ,

`shippingPhone` INT NOT NULL ,

`shippingstate` VARCHAR(20) NOT NULL ,

`shippingCity` VARCHAR(30) NOT NULL ,

`shippingPostalCode` VARCHAR(6) NOT NULL ,

`paymentFirstname` VARCHAR(20) NOT NULL ,

`paymentLastname` VARCHAR(45) NOT NULL ,

`paymentAddress1` VARCHAR(45) NOT NULL ,

`paymentAddress2` VARCHAR(30) NULL ,

`paymentEmail` VARCHAR(30) NOT NULL ,

`paymentPhone` INT NOT NULL ,

`paymentState` VARCHAR(20) NOT NULL ,

`paymentCity` VARCHAR(30) NOT NULL ,

`paymentPostalCode` VARCHAR(6) NOT NULL ,

`price` DECIMAL(10,2) NOT NULL DEFAULT 0.00 ,

`user\_login` VARCHAR(20) NOT NULL ,

PRIMARY KEY (`id`, `user\_login`) ,

INDEX `fk\_order\_user1` (`user\_login` ASC) ,

CONSTRAINT `fk\_order\_user1`

FOREIGN KEY (`user\_login` )

REFERENCES `mydb`.`user` (`login` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Id – numer identyfikacyjny zamówienia, klucz główny

Date – data złożenia zamówienia

Status – status zamówienia, przyjmowane wartości: 0 – zamówienie przyjęte, 1 – zamówienie zrealizowane

Notes – krótka notatka od użytkownika do zamówienia dla operatora systemu

Dalej są pola określające dane teleadresowe odbiorcy zamówienia i dane teleadresowe płatnika zamówienia.

User\_login – wskazanie na przynależność zamówienia do użytkownika

Price – cena zamówienia

**Klucze i indexy:**

* Id, user\_login – klucz główny
* User\_login – klucz obcy w tabeli User

## Song\_has\_order:

CREATE TABLE IF NOT EXISTS `mydb`.`song\_has\_order` (

`song\_id` INT NOT NULL ,

`order\_id` INT NOT NULL ,

INDEX `fk\_song\_has\_order\_order1` (`order\_id` ASC) ,

INDEX `fk\_song\_has\_order\_song1` (`song\_id` ASC) ,

CONSTRAINT `fk\_song\_has\_order\_song1`

FOREIGN KEY (`song\_id` )

REFERENCES `mydb`.`song` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_song\_has\_order\_order1`

FOREIGN KEY (`order\_id` )

REFERENCES `mydb`.`order` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Song\_id – wskazanie na piosenkę

Order\_id – wskazanie na zamówienie

**Klucze i indexy:**

Song\_id – klucz obcy w tabeli Song

Order\_id – klucz obcy w tabeli Order

## Song\_has\_favlist:

CREATE TABLE IF NOT EXISTS `mydb`.`song\_has\_favList` (

`song\_id` INT NOT NULL ,

`favList\_id` INT NOT NULL ,

INDEX `fk\_song\_has\_favList\_favList1` (`favList\_id` ASC) ,

INDEX `fk\_song\_has\_favList\_song1` (`song\_id` ASC) ,

CONSTRAINT `fk\_song\_has\_favList\_song1`

FOREIGN KEY (`song\_id` )

REFERENCES `mydb`.`song` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_song\_has\_favList\_favList1`

FOREIGN KEY (`favList\_id` )

REFERENCES `mydb`.`favList` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Song\_id – wskazanie na piosenkę

favList\_id – wskazanie na listę

**Klucze i indexy:**

* Song\_id – klucz obcy w tabeli Song
* favList\_id – klucz obcy w tabeli favList

## Song\_has\_shoppingCart:

CREATE TABLE IF NOT EXISTS `mydb`.`song\_has\_shoppingCart` (

`song\_id` INT NOT NULL ,

`shoppingCart\_id` INT NOT NULL ,

INDEX `fk\_song\_has\_shoppingCart\_shoppingCart1` (`shoppingCart\_id` ASC) ,

INDEX `fk\_song\_has\_shoppingCart\_song1` (`song\_id` ASC) ,

CONSTRAINT `fk\_song\_has\_shoppingCart\_song1`

FOREIGN KEY (`song\_id` )

REFERENCES `mydb`.`song` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_song\_has\_shoppingCart\_shoppingCart1`

FOREIGN KEY (`shoppingCart\_id` )

REFERENCES `mydb`.`shoppingCart` (`id` )

ON DELETE CASCADE

ON UPDATE CASCADE

)

Song\_id – wskazanie na piosenkę

ShoppingCart\_id – wskazanie na koszyk

**Klucze i indexy:**

* Song\_id – klucz obcy w tabeli Song
* ShoppingCart\_id – klucz obcy w tabeli ShoppingCart

## Procedury

* + Usuwanie komentarzy do zadanej piosenki

Create Procedure delComm(IN songId INT)

DELETE FROM `mydb`.`comments` WHERE song\_id = songId;

* Zwracanie piosenki z największą oceną

Create Procedure retTop()

Select song\_id from `mydb`.`comments` Group By song\_id Order By rate Limit 1;

* + Zwrócenie 10 najczęściej pobieranych piosenek:

Create Procedure retHot()

Select \* From `mydb`.`song` Order By counter, id Desc Limit 10;

* + Zwrócenie kategorii z ich dziećmi (2 procedury)

CREATE PROCEDURE get\_childs\_sub (IN nid INT)

BEGIN

DECLARE n INT;

DECLARE done INT DEFAULT 0;

DECLARE cur CURSOR FOR SELECT id FROM category WHERE parentId = nid;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

get\_childs\_fetch\_loop: LOOP

FETCH cur INTO n;

IF done = 1 THEN

LEAVE get\_childs\_fetch\_loop;

END IF;

INSERT INTO \_\_childs VALUES ( n );

CALL get\_childs\_sub(n);

END LOOP get\_childs\_fetch\_loop;

CLOSE cur;

END;

CREATE PROCEDURE get\_childs (IN nid INT)

BEGIN

DECLARE n INT;

DROP TEMPORARY TABLE IF EXISTS \_\_childs;

CREATE TEMPORARY TABLE \_\_childs (

node\_id INT NOT NULL PRIMARY KEY

);

SELECT COUNT(\*)

FROM category

WHERE parentId = nid INTO n;

IF n <> 0 THEN

CALL get\_childs\_sub(nid);

END IF;

SELECT t1.\*

FROM category AS t1

JOIN \_\_childs on node\_id = id;

END;

## Trigger

* + Licznik pobrań dla każdej piosenki

Create Trigger raiseCount Before Insert On `mydb`.`song\_has\_order`

For Each Row

UPDATE `mydb`.`song` SET counter = counter+1 WHERE id = NEW.song\_id;

* + Rabat przy zakupie powyżej 10 piosenek

CREATE TRIGGER addDiscount

BEFORE AFTER ON `mydb`.`order`

FOR EACH ROW

BEGIN

DECLARE i INTEGER;

SELECT COUNT(\*) INTO i FROM `mydb`.`song\_has\_order` WHERE order\_id = NEW.id;

IF i > 9 THEN

UPDATE `mydb`.`order` SET price = CEIL(price\*9/10) WHERE id = NEW.id;

END IF;

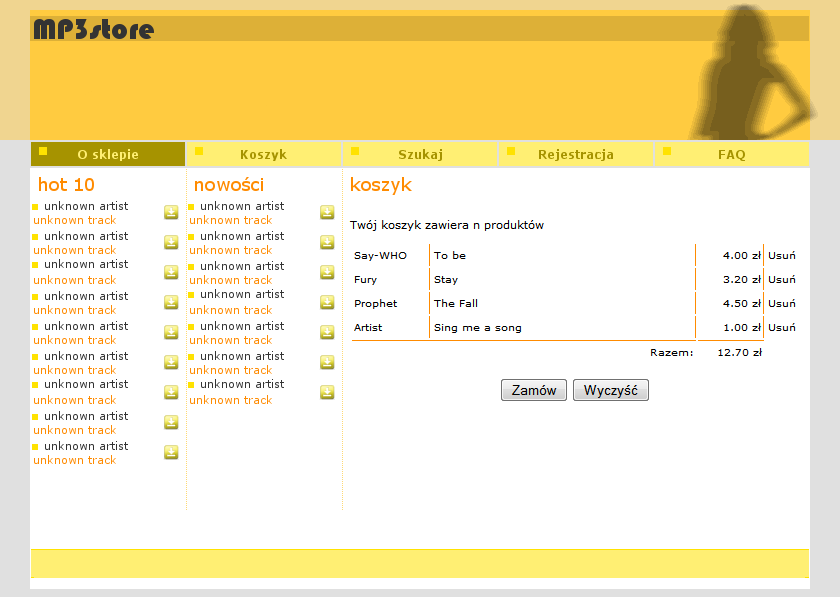
END;

# Przykładowy wygląd strony dla użytkownika

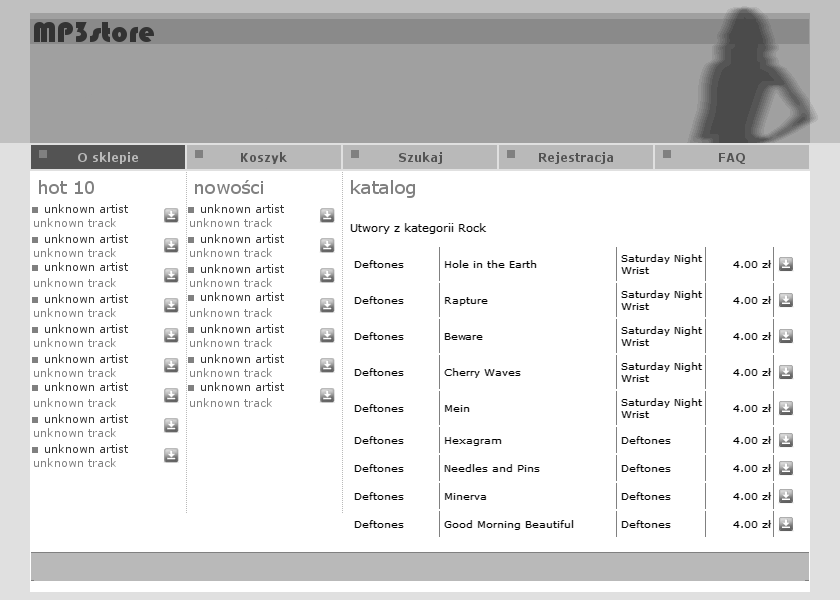
## Strona główna

****

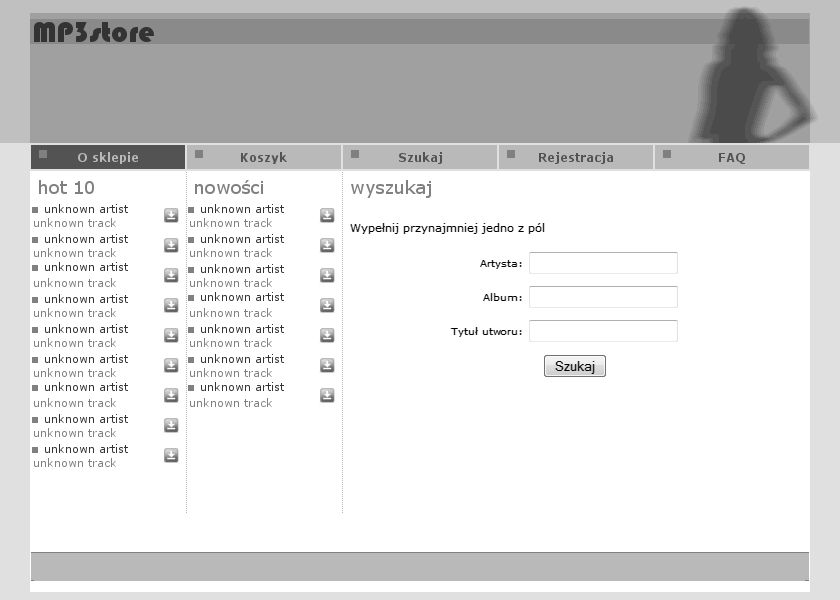
## Widok koszyka

****

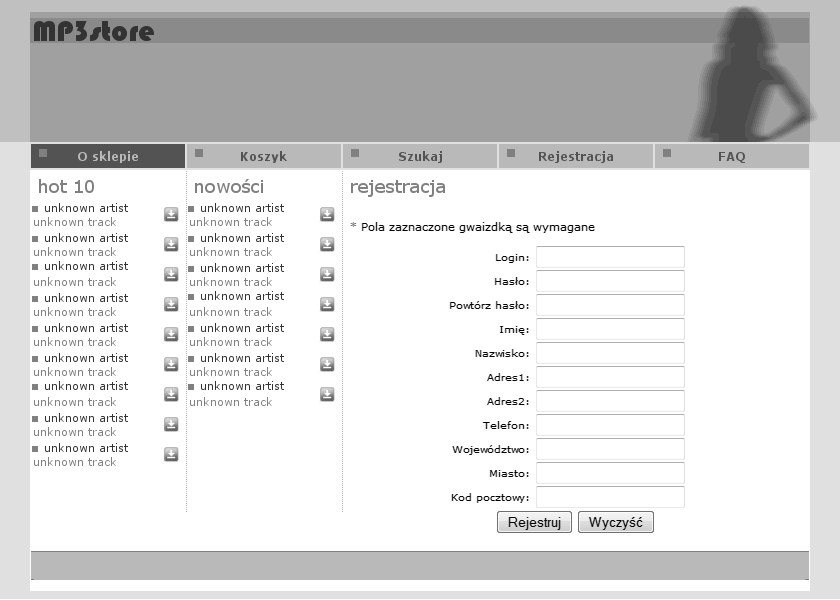
## Widok katalogu

****

## Wyszukiwarka

****

## Rejestracja

****

# Hierarchia menu

## User

* + Katalog
  + Wyszukiwarka
  + Koszyk
  + Rejestracja / Twoje konto
  + FAQ

## Admin

* + Katalog:
    - * Dodaj nowe
      * Edytuj
      * Usuń
  + Użytkownicy
    - * Edytuj
      * Usuń

# Szkielet aplikacji

* + User – moduł obsługuje rejestrację, logowanie i zarządzanie kontem użytkownika.
  + Admin – moduł obsługuje wyświetlenie panelu administracyjnego i jego główne funkcje (dodawanie, usuwanie edycję nowych utworów, artystów albumów oraz zarządzanie użytkownikami – edycja, usuwanie)
  + Cart – moduł obsługuje koszyk zamówień
  + Order – moduł obsługuje składanie zamówień przez użytkowników
  + View – moduł obsługi widoku sklepu

# Podział realizacji funkcjonalności pomiędzy stronę serwera i klienta

## Klient

* + Wyświetlanie formularzy rejestracji, logowania, zamówienia
  + Wyświetlanie katalogów produktów w przyjaznej dla użytkownika formie
  + Walidacja wprowadzanych przez użytkownika danych
  + Wyświetlanie GUI

## Serwer

* + Pobieranie i dodawanie rekordów do bazy, zarządzanie bazą danych
  + Obsługa sesji użytkownika
  + Przesyłanie do klienta danych z bazy danych
  + Trigger: przydzielanie rabatu do zamówienia jeśli spełnione zostaną warunki
  + Trigger: inkrementacja licznika pobrań dla każdej zakupionej płyty
  + Procedura zwracająca CD z największą oceną
  + Procedura usuwająca wszystkie komentarze do podanej płyty
  + Dwie procedury do zwracania drzewiastej struktury kategorii

# Określenie sposobu realizacji wymuszanie integralności danych

Tabele mają zdefiniowane wymuszenia integralności na zasadzie powiązań kluczami obcymi. Gdy usuwamy rekord z tabeli do której należy klucz obcy, dane w innych tabelach wykorzystujących ten klucz usuwane są kaskadowo. Jest to wymagane do zachowania integralności danych – np. piosenka nie może istnieć w systemie jeśli skasujemy artystę który ją wykonał; komentarz musi zostać usunięty, jeśli piosenka do której został przypisany zostanie usunięta – nie może zalegać w bazie.

# Opis transakcji dla wybranego modułu.

## Moduł order – złożenie zamówienia przez użytkownika.

Klient wybiera produkty do koszyka. Tymczasowo każdy produkt jest trzymany w zmiennej sesyjnej $\_SESSION[] dostarczoną przez mechanizm sesji w PHP. Zmienna ta – przyjmijmy jej nazwę ‘cart’ będzie tablicą do której dodawane będą nowe wpisy zawierające id piosenki dodanej do koszyka. Gdy użytkownik zakończy tworzenie koszyka i postanowi dokonać zakupu, dane zaczynamy przesyłać do bazy danych. Tworzy się nowa transakcja. Do bazy dodawany jest rekord do tabeli ‘order’ z informacjami o kupującym. Teraz dodawane dopiero są rekordy do tabeli ‘song\_has\_order’ (taka kolejność wymagana jest przez wymuszenie integralności – musiał zostać utworzony klucz w tabeli ‘order’, by wykorzystać go jako klucz obcy w tabeli ‘song\_has\_order’). Po dodaniu wszystkich produktów z koszyka wykonujemy Commit. Jeśli gdzieś po drodze baza danych zwróciła błąd wykonujemy Rollback i wysyłam do użytkownika komunikat o chwilowych problemach technicznych. Jeśli wszystko poszło dobrze czyścimy zawartość koszyka i wszelkich zmiennych trzymających dane z koszyka.

Na potrzeby prezentacji działania został użyty mechanizm sesji do trzymania produktów w koszyku. W finalnej wersji za przechowywanie produktów w koszyku odpowiedzialna byłaby tabela shoppingCart oraz song\_has\_shoppingCart. Produkty byłyby tymczasowo w niej składowane a przy zatwierdzeniu zamówienia dane przenoszone byłyby do tabeli order i song\_has\_order.