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

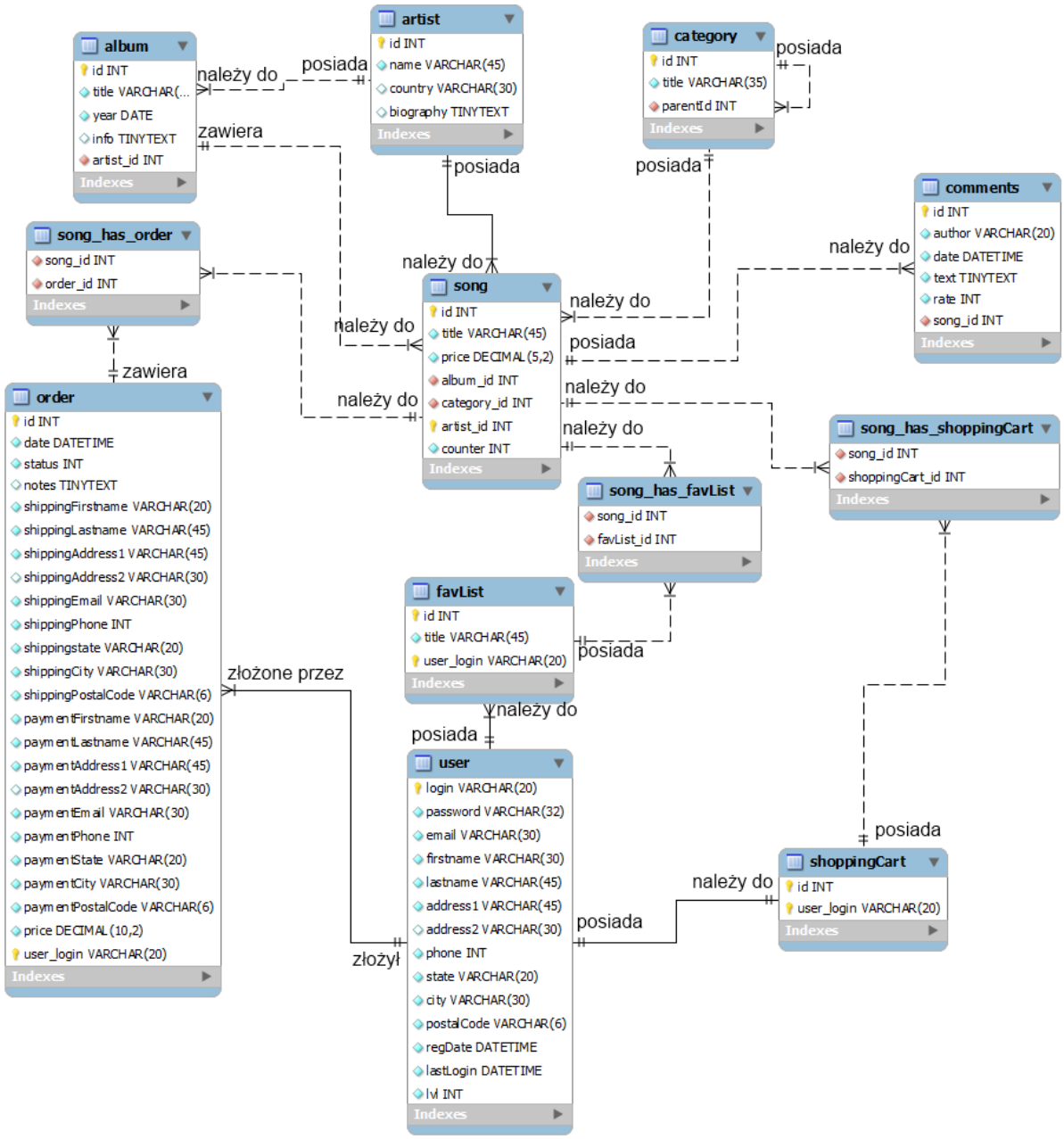
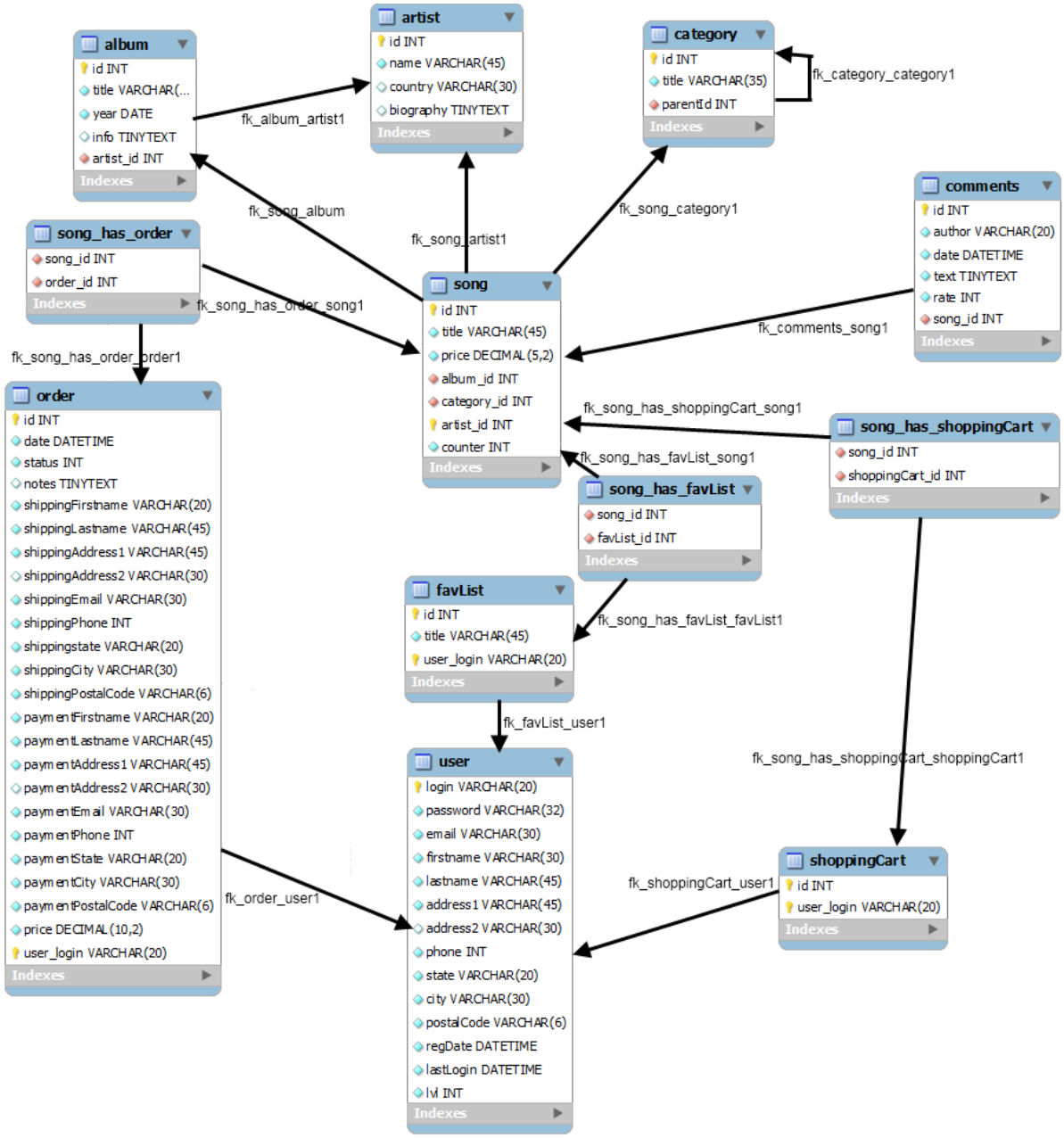

album
id INT
title VARCHAR(...
year DATE
info TINYTEXT
artist_id INT
Indexes
należy do
posiada
artist
id INT
name VARCHAR(45)
country VARCHAR(30)
biography TINYTEXT
Indexes
zawiera
posiada
category
id INT
title VARCHAR(35)
parentId INT
Indexes
posiada
comments
id INT
author VARCHAR(20)
date DATETIME
text TINYTEXT
rate INT
song_id INT
Indexes
należy do
song_has_order
song_id INT
order_id INT
Indexes
zawiera
należy do
song
id INT
title VARCHAR(45)
price DECIMAL(5,2)
album_id INT
category_id INT
artist_id INT
counter INT
Indexes
należy do
należy do
należy do
posiada
należy do
należy do
order
id INT
date DATETIME
status INT
notes TINYTEXT
shippingFirstname VARCHAR(20)
shippingLastname VARCHAR(45)
shippingAddress1 VARCHAR(45)
shippingAddress2 VARCHAR(30)
shippingEmail VARCHAR(30)
shippingPhone INT
shippingstate VARCHAR(20)
shippingCity VARCHAR(30)
shippingPostalCode VARCHAR(6)
paymentFirstname VARCHAR(20)
paymentLastname VARCHAR(45)
paymentAddress1 VARCHAR(45)
paymentAddress2 VARCHAR(30)
paymentEmail VARCHAR(30)
paymentPhone INT
paymentState VARCHAR(20)
paymentCity VARCHAR(30)
paymentPostalCode VARCHAR(6)
price DECIMAL(10,2)
user_login VARCHAR(20)
Indexes
song_has_shoppingCart
song_id INT
shoppingCart_id INT
Indexes
song_has_favList
song_id INT
favList_id INT
Indexes
favList
id INT
title VARCHAR(45)
user_login VARCHAR(20)
Indexes
posiada
należy do
złożone przez
złożył
posiada
user
login VARCHAR(20)
password VARCHAR(32)
email VARCHAR(30)
firstname VARCHAR(30)
lastname VARCHAR(45)
address1 VARCHAR(45)
address2 VARCHAR(30)
phone INT
state VARCHAR(20)
city VARCHAR(30)
postalCode VARCHAR(6)
regDate DATETIME
lastLogin DATETIME
lvl INT
Indexes
posiada
należy do
shoppingCart
id INT
user_login VARCHAR(20)
Indexes

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;
```

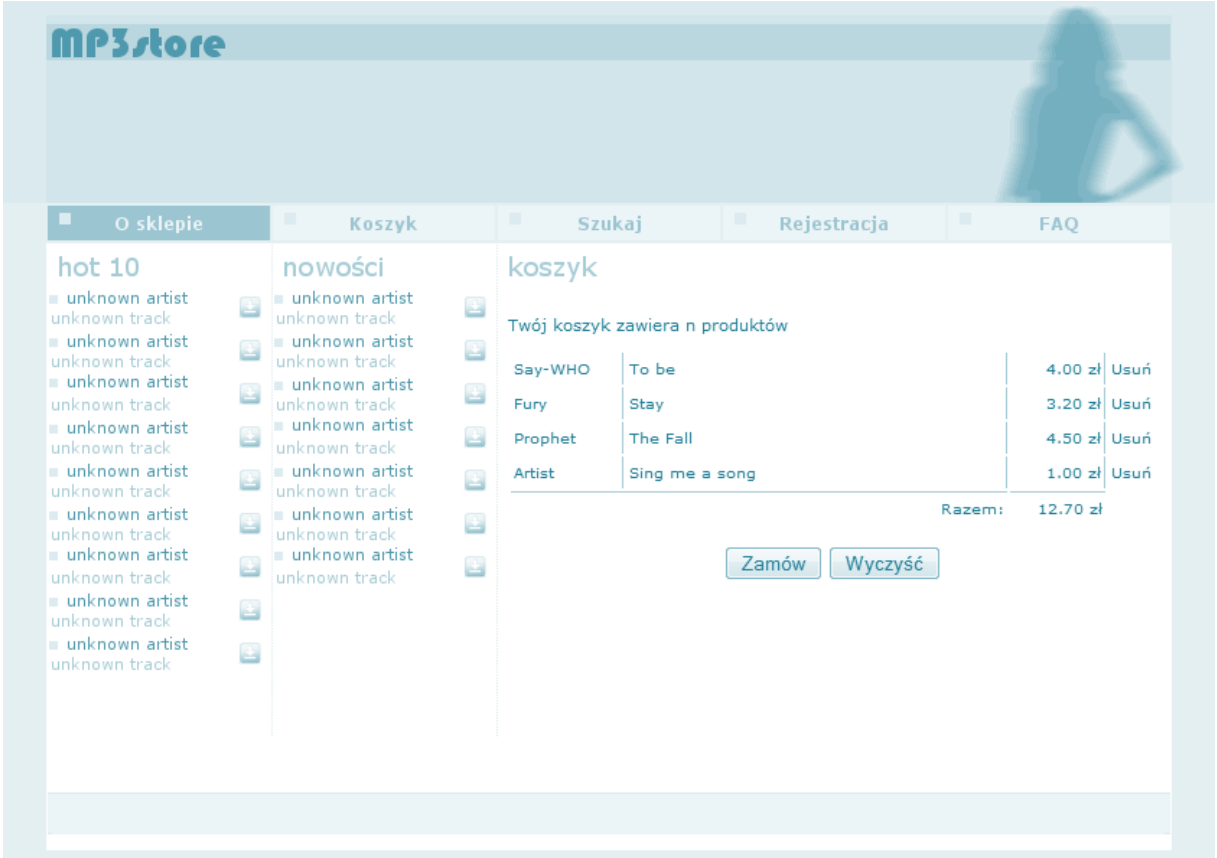
- Rabat przy zakupie powyżej 10 piosenek

```
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;
```

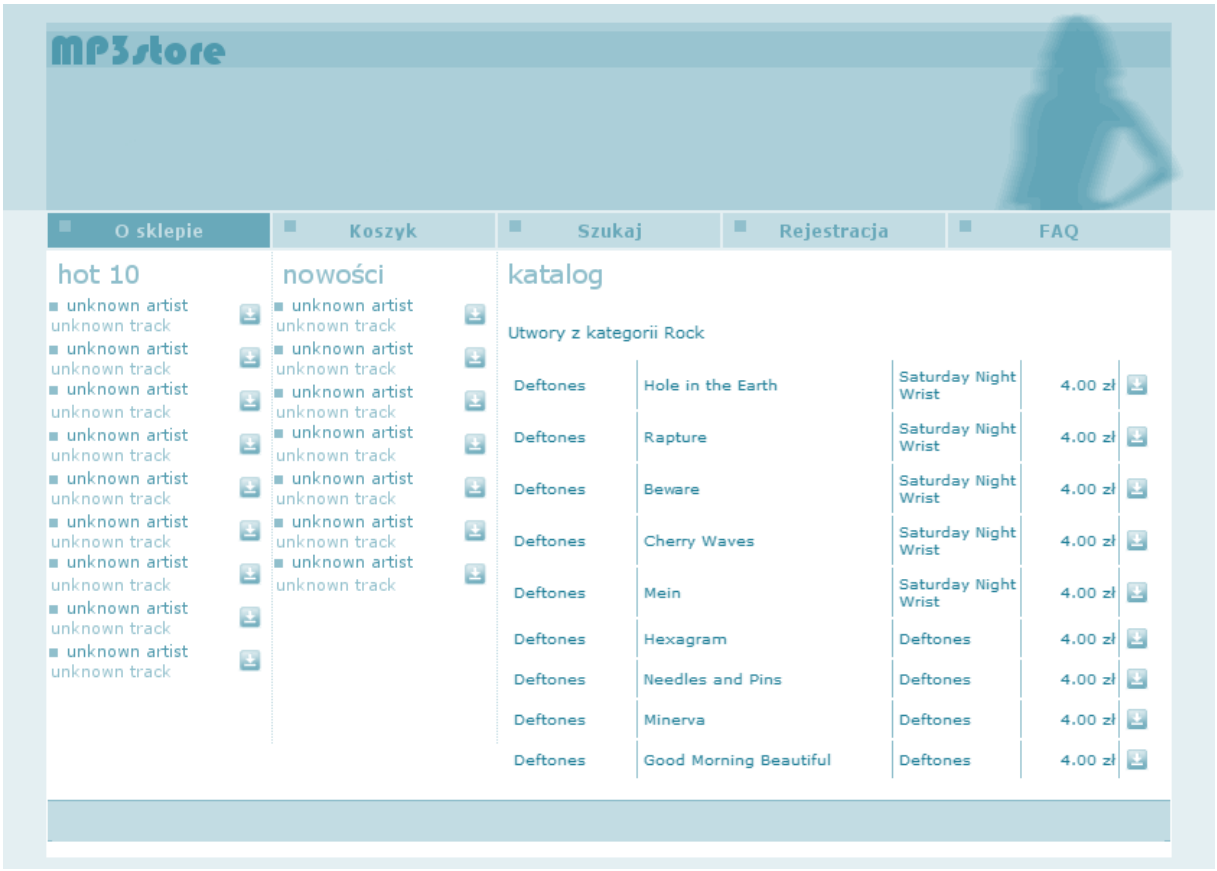
Przykładowy wygląd strony dla użytkownika

Strona główna

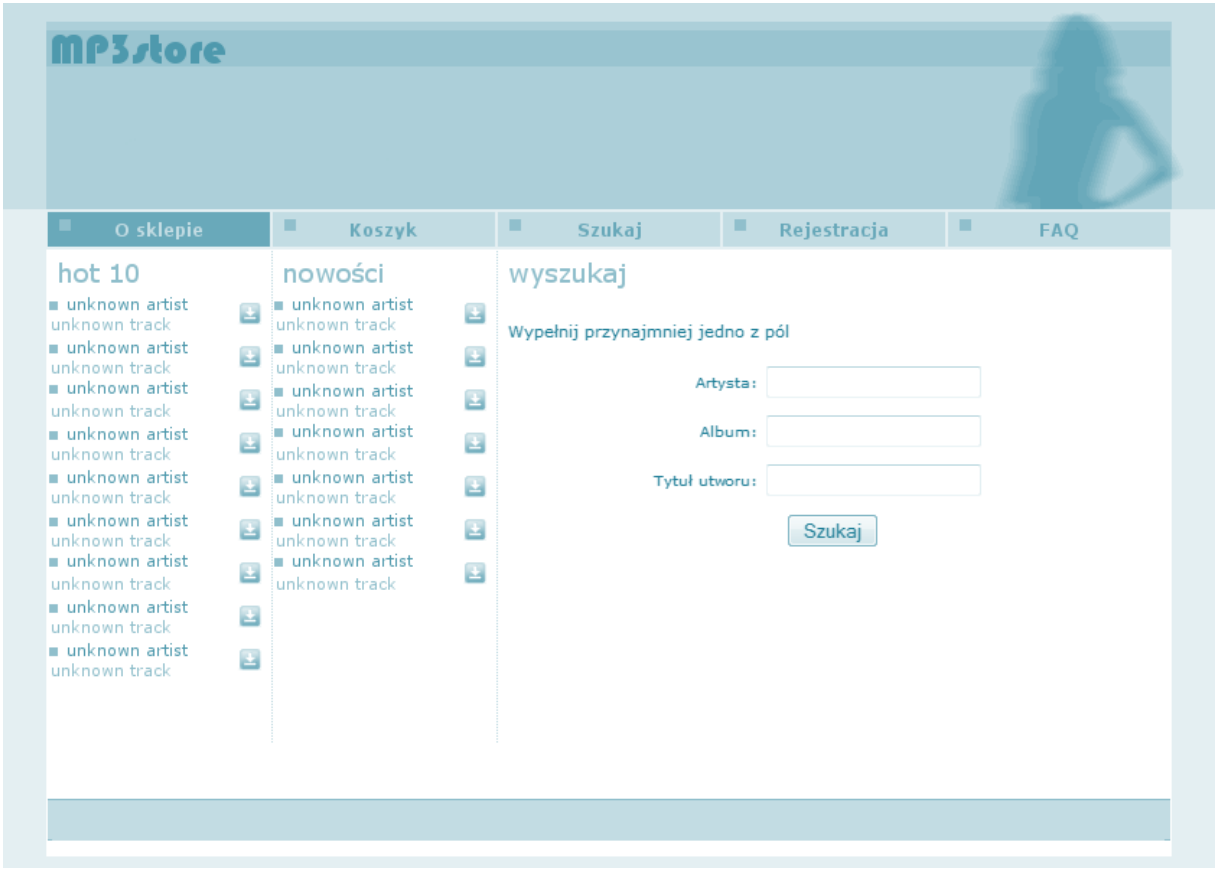
Widok koszyka



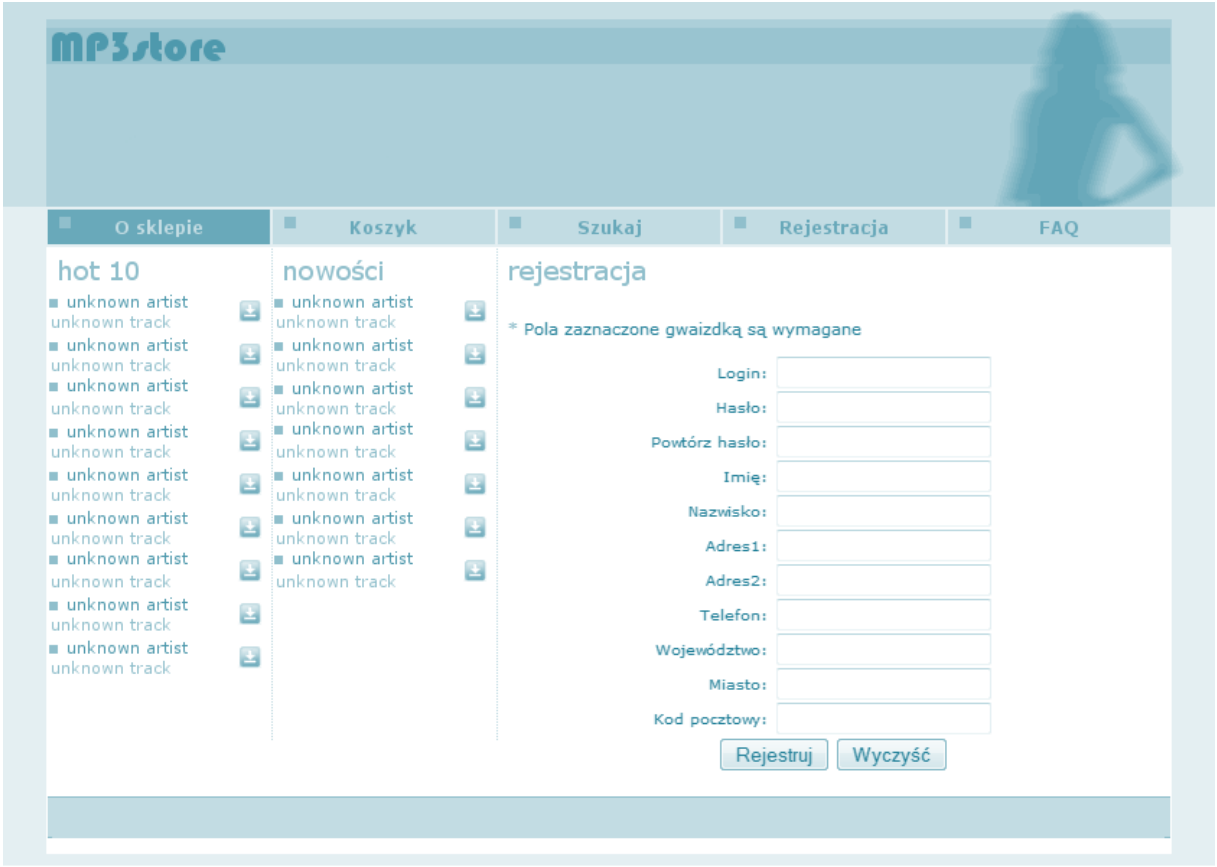
Widok katalogu



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.