**КУРСОВОЙ ПРОЕКТ**

**По дисциплине «Базы данных»**

**Вариант №9**

Выполнил студент гр.33504/1

Сапелкин А.В.

Вишневская Т.А.

Руководитель

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2015 г.

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА (КУРСОВОЙ РАБОТЫ)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студенту группы | | *33504/1* | |  | *Сапелкину Александру Викторовичу* | | | | | | | |
|  | | *(номер группы)* | |  | *(фамилия, имя, отчество)* | | | | | | | |
| ***1. Тема проекта (работы)*** | | | | *Разработка системы управления* | | | | | | | | |
| *Музыкальным магазином* | | | | | | | | | | | | |
| ***2. Срок сдачи студентом законченного проекта (работы)*** | | | | | | | | | | | |  |
| ***3. Исходные данные к проекту (работе)*** | | | | | | | | *Разработка* | | | | |
| *многопользовательской автоматизированной системы управления* | | | | | | | | | | | | |
| *организацией. Объект автоматизации – музыкальный магазин.* | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| ***4.Содержание пояснительной записки****.* | | | | | | | | | | | | |
| *Введение,* *исследование предметной области и построение схемы данных,* | | | | | | | | | | | | |
| *реализация, клиентское приложение, список использованных источников* | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| Примерный объем пояснительной записки | | | | | | ***12*** | | | страниц машинописного | | | |
| Текста | | | | | | | | | | | | |
| 5. Перечень графического материала (с указанием обязательных чертежей и | | | | | | | | | | | | |
| плакатов) |  | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| 6. Консультанты | | |  | | | | | | | | | |
|  | | | | | | | | | | | | |
| 7. Дата получения задания: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г. | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| Руководитель | | | | |  | | |  | | Т.А.Вишневская | | |
|  | | | | | *(подпись)* | | |  | | *(инициалы, фамилия)* | | |
| Задание принял к исполнению | | | | |  | |  | | | *А.В.Сапелкин* | | |
|  | | | | | *(подпись)* | |  | | | *(инициалы, фамилия)* | | |
|  | | | | | | | | | | |  | |
|  | | | | | | | | | | | *(дата)* | |

## СОДЕРЖАНИЕ

[Постановка задачи 4](#_Toc439230195)

[План работы 4](#_Toc439230196)

[Описание предметной области 4](#_Toc439230197)

[Необходимые запросы 5](#_Toc439230198)

[Схема данных 6](#_Toc439230199)

[Код программы 7](#_Toc439230200)

[Выводы 20](#_Toc439230201)

[Список литературы 20](#_Toc439230202)

[Приложение 1. Сценарий создания базы 21](#_Toc439230203)

## Постановка задачи

Разработка многопользовательской автоматизированной системы управления организацией. Объект автоматизации – музыкальный магазин

## План работы

1)Нарисовать ER-диаграмму нашей задачи. Изучить все связи между таблицами.

2)Создать базу данных, соответствующую поставленной задаче.

3)Создать диаграмму, соответствующую поставленной задаче, используя таблицы, созданные в пункте 2. Образцом будет служить ER-диаграмма, созданная в пункте 1.

4)Используя язык программирования Java написать программу, которая будет выполнять с базой данных, все действия, указанные в постановке задачи.

## Описание предметной области

Пользователи системы

|  |  |
| --- | --- |
| № пп | Наименование пользователя |
| 1 | Музыкант |
| 2 | Клиент |

Функции пользователей

|  |  |  |
| --- | --- | --- |
| № пп | Выполняемая функция | Результат |
| ***Первый пользователь*** | | |
| 1 | Прсмотр списка продукции | Отображение списка музыкального товара |
| 2 | Покупка | Покупка музыкального товара |
| 3 | Интеллектуальный поиск музыки |  |
|  |  |  |
| ***Второй пользователь*** | | |
| 1 | Добавление музыкальных альбомов | Товар для продажи |
| 2 | Продажа продукции | Пополнение баланса |

## Необходимые запросы

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Формулировка запроса | Пользователи | Название процедуры |
| 1 | Покупка Альбома | 1 | Buy\_album |
| 2 | Поиск Альбома | 1 | Search\_albums |
| 3 | Поиск клипа | 1 | Search\_clips |
| 4 | Поиск музыки похожей с той, что уже куплена пользователем | 1 | searchfavoritegenre |
| 5 | Поиск треков | 1 | Search\_tracks |
| 6 | Выбрать музыку артистов, которые пишут для твоего пола | 1 | Select\_artist\_auditory |
| 7 | Умная процедура поиска | 1 | findByCreationTemproraryTable |
| 8 | Регистрация пользователя | - | Прописано в коде клиента |
| 9 | Логирование пользователя | - | Прописано в коде клиента |
| 10 | Вспомогательные функции для музыканта | 2 | Items\_needed\_for\_production |

## Схема данных

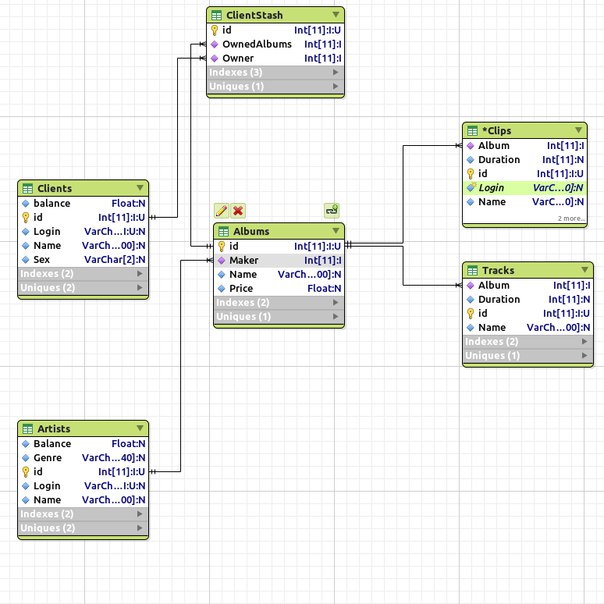


Рисунок 1. Схема данных

## Код программы

**Покупка альбома**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `buy\_album`(in albumtitle varchar(40), in clientlogin varchar(100))

BEGIN

declare user\_balance double;

declare user\_id int;

declare album\_id int;

declare artist\_login varchar(100);

set album\_id = (select id from Albums where Name = albumtitle);

set user\_balance = (select balance from Clients where login = clientlogin);

set user\_id = (select id from Clients where login = clientlogin);

if user\_balance >= (select al.Price from Albums al where al.Name = albumtitle) and ((select count(\*) from ClientStash cs join Clients cl on cs.owner =cl.id where OwnedAlbums=album\_id and cl.login=clientlogin )=0) then

insert into ClientStash(OwnedAlbums,Owner) values((select id

from Albums

where Name = albumtitle),user\_id);

UPdate Clients SET balance = balance - (select al.Price from Albums al where Name = albumtitle ) where login = clientlogin;

set artist\_login = (select ar.login from Artists ar join Albums al on al.Maker = ar.id where al.id = album\_id);

update Artists ar Set balance = balance +(select al.Price from Albums al where Name = albumtitle ) where ar.Name = artist\_login;

end if;

END

**Поиск альбома**

DELIMITER $$

CREATE DEFINER=`%`@`%` PROCEDURE `search\_albums`(in title varchar(40))

BEGIN

select al.Name as ALbumName, ar.Name as ArtistName ,al.Price as AlbumPrice

from Albums al join Artists ar on(Maker)

where al.name like concat('%',title,'%')

group by al.id;

END

**Поиск клипа**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search\_clips`(in title varchar(40))

BEGIN

select cl.Name, al.Name,ar.Name

from Clips cl join Albums al on cl.Album = al.id join Artists ar on al.Maker = ar.id

where tr.name like concat('%',title,'%');

END

**Поиск музыки похожей с той, что уже куплена пользователем**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `searchfavoritegenre`(in clientlogin varchar(40))

BEGIN

declare userLoginId int;

declare favorite\_genre varchar(100);

set userLoginId=(select id from Clients where login = clientlogin);

select distinct alb.Name as "Album Name",art.Genre as "Genre" from

ClientStash as cst join Albums as alb join Artists as art

where cst.id=userLoginId and art.genre =(select ar.Genre from ClientStash as cs

join Albums as al on cs.OwnedAlbums=al.id

join Artists ar on ar.id=al.Maker

where cs.Owner=userLoginId

group by ar.Genre

order by count(\*) desc

limit 1);

END

**Поиск треков**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search\_tracks`(in title varchar(40))

BEGIN

select tr.name, al.Name,ar.Name,al.Price

from Tracks tr join Albums al on tr.Album = al.id join Artists ar on al.Maker = ar.id

where tr.name like concat('%',title,'%');

END

**Выбрать музыку артистов, которые пишут для твоего пола**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sexStatisticsAlbum`(in userLogin varchar(40))

BEGIN

declare userSex varchar(1) default 'M';

declare anotherSex varchar(1) default 'M';

set userSex=(select sex from Clients where login=userLogin);

if userSex='M' then

set anotherSex='F';

else

set anotherSex='M';

end if;

select distinct tbl.alname as "Album Title" from

(select al.name as alname,cl.sex as csex,ar.name as arname

from ClientStash as cs

join Albums as al

join Artists as ar

join Clients as cl

where cl.sex=userSex and (select count(\*) from ClientStash join Albums join Atists join Clients as cl where cl.sex=userSex)>(select count(\*) from ClientStash join Albums join Artists as art join Clients as cl where cl.sex=anotherSex) group by cl.id

)tbl;

END

**Умная процедура поиска**

DELIMITER $$

CREATE DEFINER=`root`@`127.0.0.1` PROCEDURE `findByCreationTemproraryTable`(in userlogin varchar(40))

BEGIN

declare userSex varchar(1) default 'M';

declare userLoginId int;

declare favorite\_genre varchar(100);

declare anotherSex varchar(1) default 'M';

set userSex=(select sex from Clients where login=userLogin);

if userSex='M' then

set anotherSex='F';

else

set anotherSex='M';

end if;

set userLoginId=(select id from Clients where login = userlogin);

drop table if exists temp;

create table if not exists temp(albums varchar(100));

insert into temp(albums) select distinct tbl.alname from

(select al.name as alname,cl.sex as csex,ar.name as arname

from ClientStash as cs

join Albums as al

join Artists as ar

join Clients as cl

where cl.sex=userSex and (select count(\*) from ClientStash join Albums join Artists join Clients as cl where cl.sex=userSex)> (select count(\*) from ClientStash join Albums

join Artists join Clients as cl where cl.sex=anotherSex)

group by cl.id

)tbl;

insert into temp(albums) select distinct alb.Name from

ClientStash as cst join Albums as alb join Artists as art

where cst.id=userLoginId and art.genre =(select ar.Genre from ClientStash as cs

join Albums as al on cs.OwnedAlbums=al.id

join Artists ar on ar.id=al.Maker

where cs.Owner=userLoginId

group by ar.Genre

order by count(\*) desc

limit 1);

select \* from temp;

END

**Регистрация пользователя**

if (clientSexMaleRadioButton.isSelected()) {

sex = "M";

} else {

sex = "F";

}

try {

statement.executeUpdate(createUserTemplate+" '" + login+"'@'localhost'" + identifiedByTemplate +" '"+ password+"';");

statement.executeUpdate(privileguesTemplate + " '" + login + "'@'localhost' ");

String insertQuery = "INSERT INTO Clients(Name,Login, Sex, Balance) VALUES('" + name + "', '"+login+"', " + "'" + sex + "', " + "1000.0)";

System.out.println(insertQuery);

statement.executeUpdate(insertQuery);

} catch (SQLException e) {

e.printStackTrace();

}

}else if (artistRadioButton.isSelected()) {

String genre = artistGenreTextField.getText().toString();

try {

statement.executeUpdate(createUserTemplate+" '" + login+"'@'localhost'" + identifiedByTemplate +" '"+ password+"';");

String sqlquery="INSERT INTO Artists(Name,Login, Genre, Balance) VALUES('" + name + "', '" +login+"', "+ "'" + genre + "', " + "100.0)";

statement.executeUpdate(sqlquery);

System.out.println(sqlquery);

sqlquery = privileguesTemplate + " '" + login + "'@'localhost' ";

statement.executeUpdate(sqlquery);

System.out.println(sqlquery);

**Логирование пользователя**

final String clientExistanceQuery = "select count(\*) as logintest from Clients where login='"+userField.getText().toString()+"'";

final String artistExistanceQuery = "select count(\*) as logintest from Artists where login='"+userField.getText().toString()+"'";

Class.forName(Sqlwork.JDBC\_DRIVER);

String loginName = userField.getText().toString();

String password = String.valueOf(passwordField.getPassword());

connection = DriverManager.getConnection(DB\_URL+DATABASE\_NAME, loginName, password);

statement=connection.createStatement();

ResultSet queryResult = statement.executeQuery(clientExistanceQuery);

if(queryResult.next()) {

if (queryResult.getInt("logintest") > 0) {

clientFrame = new AnotherClientFrame(loginName, password);

} else {

queryResult = statement.executeQuery(artistExistanceQuery);

if(queryResult.next()) {

if (queryResult.getInt("logintest") > 0) {

(new ArtistMenuFrame(loginName, password)).setVisible(true);

}

}

}

closeFrame();

}

**Вспомогательные функции для музыканта**

**Добавление альбома**

int artistId=0;

Class.forName(JDBC\_DRIVER);

Connection connection = DriverManager.getConnection(DB\_URL + DATABASE\_NAME, login, password);

Statement statement = connection.createStatement();

String idQuery = "SELECT id from Artists where login = " + "'"+login+"'";

ResultSet resultSet = statement.executeQuery(idQuery);

if (resultSet.next()) {

artistId = resultSet.getInt("id");

System.out.println(artistId);

}

String insertQuery = "INSERT INTO Albums (Name, Price,Maker ) VALUES (" + "'" + nameTextField.getText().toString() + "', '" + priceTextField.getText().toString() + "', '" + artistId + "')";

System.out.println(insertQuery);

statement.executeUpdate(insertQuery);

**Добавление клипа**

String artistIdQuery = "select id from Artists where login =?";

String albumIdQuery = "select id from Albums where Maker = ? and Name =?";

String addClipQuery = "insert into Clips(Name, Duration, Album) values(?,?,?)";

PreparedStatement artistIdStatement = connection.prepareStatement(artistIdQuery);

PreparedStatement albumIdStatement = connection.prepareStatement(albumIdQuery);

PreparedStatement addClipStatement = connection.prepareStatement(addClipQuery);

String albumName = albumNameField.getText().toString();

String clipName = clipNameField.getText().toString();

String clipDuration = durationTextField.getText().toString();

String artistId = "";

String albumId = "";

artistIdStatement.setString(1, login);

ResultSet artistIdResultSet =artistIdStatement.executeQuery();

if (artistIdResultSet.next()) {

artistId = artistIdResultSet.getString("id");

}

albumIdStatement.setString(1, artistId);

albumIdStatement.setString(2,albumName);

ResultSet albumIdResultSet = albumIdStatement.executeQuery();

if (albumIdResultSet.next()) {

albumId = albumIdResultSet.getString("id");

}

addClipStatement.setString(1, clipName);

addClipStatement.setString(2, clipDuration);

addClipStatement.setString(3, albumId);

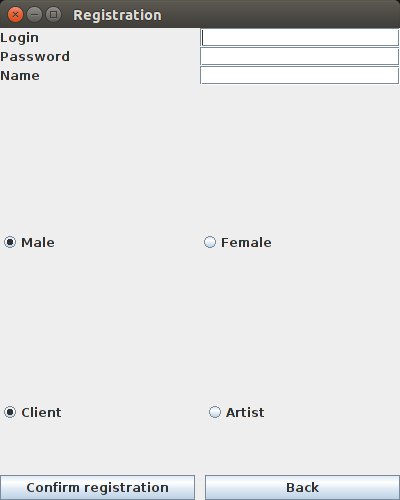
System.out.println("Izmeneno"+ addClipStatement.executeUpdate());

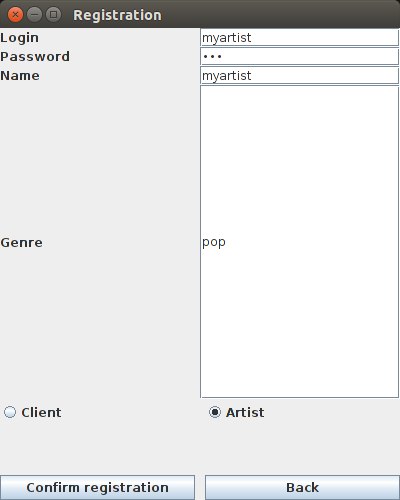
**etc...**

#### Запросы

Мои запросы экстремально связаны с клиентом. Поэтому для объяснения их работы привожу алгоритм работы клиента.

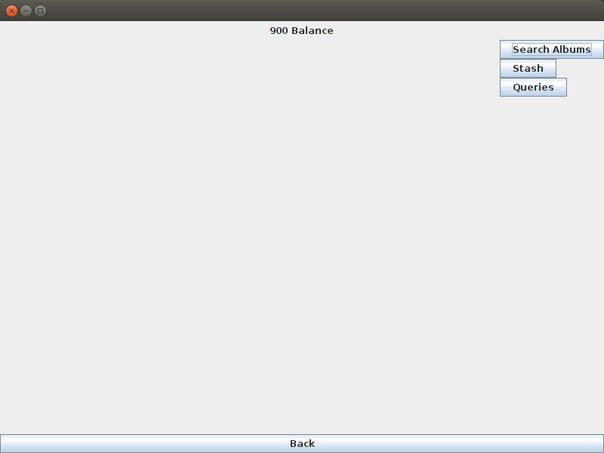
При первом запуске необходимо зарегистрироваться:



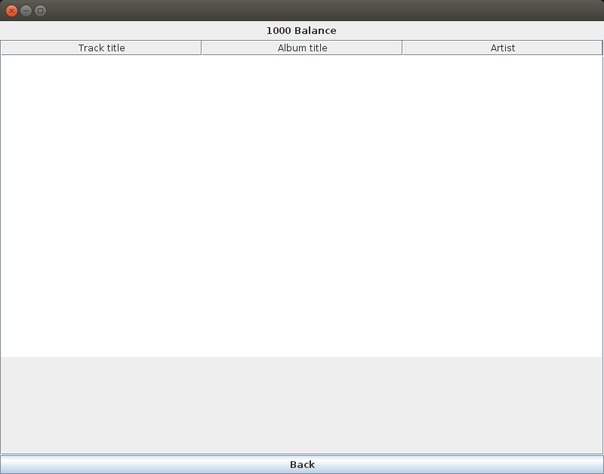


При регистрации соответственно выполняются запросы пункта 8.

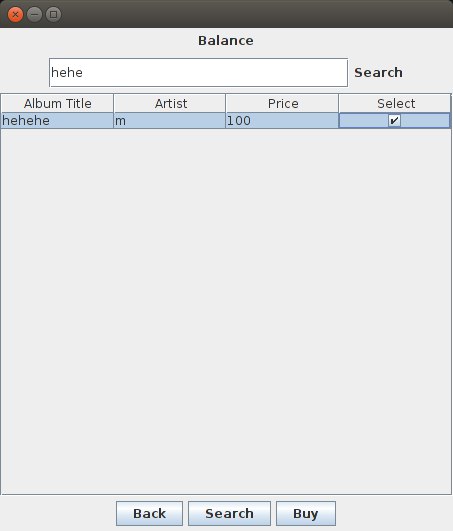
Затем возможен вход в двух режимах работы. Рассмотрим сначала работу покупателя(клиента магазина):



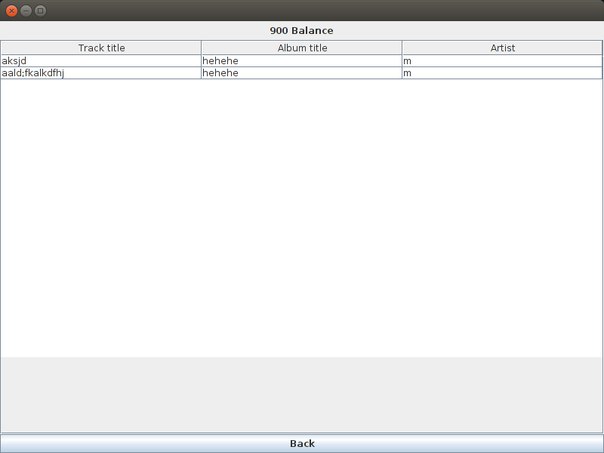
Его хранилище покупок изначально пусто:



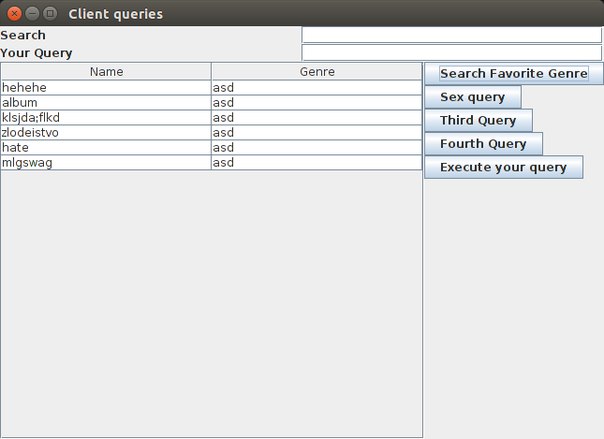
#### Затем покупатель может воспользоваться поиском и купить найденный альбом:



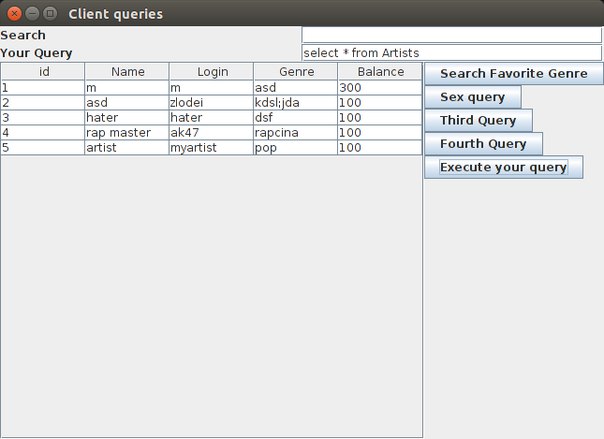
Содержимое его хранилища измениться:



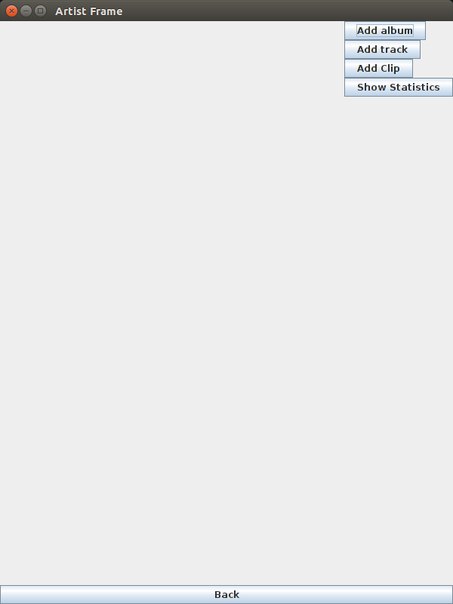
Также доступен “Умный поиск”:



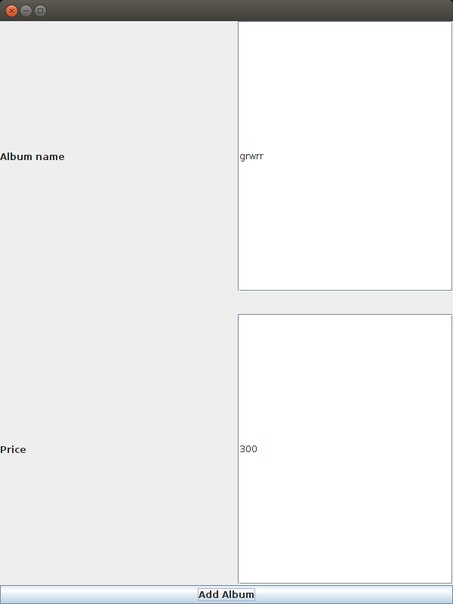
И произвольный поиск по базе данных (Подразумевается, наши клиенты исключительно образованные люди):



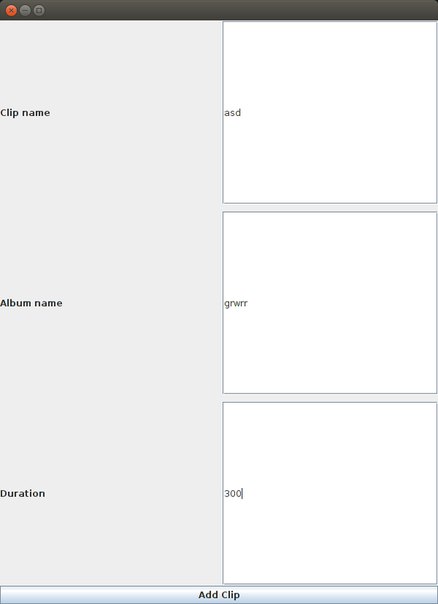
Теперь перейдем к музыканту:



Для начала необходимо добавить альбом:



Затем добавить или клипы или треки:



Моя клиентская программа содержит множество форм и окон. Особое внимание представляет окно запросов клиента магазина, которое подстраивается под запрос любой сложности.

## Выводы

В ходе работы было изучено написание запросов различной сложности для обработки информации из базы данных. Так же было написано многопользовательское приложение, позволяющее управлять работой музыкального магазина. Есть регистрация пользователей, в связи со статусом есть разграничение области действий. Доступные режимы – музыкант, клиент.

## Список литературы

* https://msdn.microsoft.com/
* Microsoft Developer Network Library // URL: *http://msdn.microsoft.com/library*
* SQL Tutorials // W3Schools Online Web Tutorials. – URL: *http://www.w3schools.com/sql/default.asp*
* StackOverflow // URL: *http://stackoverflow.com/*
* Б. Я. Советов, В. В. Цехановский, В. Д.Чертовской «Базы данных» — М. Высшая школа, 2007
* С. М. Диго «Базы данных» — М. Финансы и статистика,2005
* Владимир Кириллов, Геннадий Громов «Введение в реляционные базы данных» — СПб. БХВ-Петербург,2009

## Приложение 1. Сценарий создания базы

package com.company;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

\* Created by alexander on 28.10.15.

\*/

public class Sqlwork {

static final String JDBC\_DRIVER = "com.mysql.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost/";

static final String DATABASE\_NAME = "musicshop";

String USER = "root";

String PASS = "admin";

Connection connection = null;

Statement statement = null;

void createDatabase() {

try {

Class.forName(JDBC\_DRIVER);

System.out.println("Connecting database");

try {

connection = DriverManager.getConnection(DB\_URL, USER, PASS);

statement=connection.createStatement();

statement.executeUpdate("CREATE DATABASE IF NOT EXISTS "+DATABASE\_NAME + ";");

connection = DriverManager.getConnection(DB\_URL + DATABASE\_NAME, USER, PASS);

statement=connection.createStatement();

System.out.println("Database created succesfully");

System.out.println("trying to create table");

createTables();

} catch (SQLException e) {

e.printStackTrace();

}finally {

try {

if (statement != null) {

statement.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

private void createTable(String name,String... fields) throws SQLException {

String sqlCreateQuery = "CREATE TABLE IF NOT EXISTS " + name + "(";

for (int i = 0; i < fields.length-1; i++) {

sqlCreateQuery+=fields[i]+",";

}

sqlCreateQuery+=fields[fields.length-1] + ")ENGINE=InnoDB";

statement.executeUpdate(sqlCreateQuery);

}

void createTables(){

String loginTemplate = "Login Varchar(40) UNIQUE";

String id\_template="id int not null primary key auto\_increment";

String nameTemplate ="Name varchar(100) ";

try {

createTable("Artists",

id\_template,

nameTemplate,

loginTemplate,

"Genre varchar(40)",

"Balance float"

);

createTable("Albums",

id\_template,

nameTemplate,

"Maker int not null",

"Price float ",

"foreign key(Maker) references Artists(id)");

createTable("Tracks",

id\_template,

nameTemplate,

"Duration int",

"Album int not null",

"foreign key(Album) references Albums(id)"

);

createTable("Clips",

id\_template,

nameTemplate,

"Duration int",

"Album int not null",

"foreign key(Album) references Albums(id)"

);

createTable("Clients",

id\_template,

loginTemplate,

nameTemplate,

"Sex Varchar(2)",

"balance float");

createTable("ClientStash",

id\_template,

"OwnedAlbums int not null",

"Owner int not null",

"foreign key(OwnedAlbums) references Albums(id)",

"foreign key(Owner) references Clients(id)");

createTable("ShopAdministrator",

id\_template,

nameTemplate,

"balance float");

} catch (SQLException e) {

e.printStackTrace();

}

}

void deleteDatabase(){

try {

Class.forName(JDBC\_DRIVER);

connection = DriverManager.getConnection(DB\_URL, USER, PASS);

statement=connection.createStatement();

String sqlQuery="DROP DATABASE IF EXISTS "+DATABASE\_NAME;

statement.executeUpdate(sqlQuery);

} catch (SQLException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

Sqlwork(String username,String password){

this.USER=username;

this.PASS=password;

}

}