МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)   
  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Логическое программирование  
  
  
  
ОТЧЕТ**по лабораторной работе №4

Тема: Получение информации из реляционной базы данных.  
Вариант №8

Студент: Колбанов Д.О.   
Группа: 6301-020302D   
  
Преподаватель: Лобанков А.А.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2024

**Теоретические сведения**

В SWI-Prolog, ODBC (Open Database Connectivity) – это стандартный интерфейс доступа к базам данных, который позволяет Prolog-программам взаимодействовать с различными СУБД (системами управления базами данных) через ODBC-драйверы.

Для установления соединения с базой данных с использованием ODBC в SWI-Prolog необходимо выполнить несколько шагов. Прежде всего, убедитесь, что установлен ODBC-драйвер для используемой базы данных.

Затем понадобится создать конфиг ODBC (ODBC Data Source) для базы данных. Это позволит установить соединение с базой данных через ODBC. Далее нужно загрузить библиотеку odbc и использовать предикат odbc\_connect, чтобы установить соединение с базой данных. Пример подключения представлен на рисунке 1.

Изображение выглядит как текст, электроника, снимок экрана, Шрифт

Автоматически созданное описание  
Рисунок 1 – Пример подключения к базе данных PostgreSQL

В приведенном примере swi представляет собой имя ODBC-источника данных, user и password – учетные данные для подключения к базе данных, alias – идентификатор соединения, модификатор open(once) означает, что второй вызов для открытия того же DSN просто возвращает существующее соединение.

odbc\_query – это встроенный предикат в SWI-Prolog, который используется для выполнения SQL-запросов к базе данных через ODBC-соединение. Предикат odbc\_query принимает три аргумента: соединение с базой данных, SQL-запрос в виде строки и переменную для хранения результатов запроса.

odbc\_disconnect – это встроенный предикат в SWI-Prolog, который используется для отключения от базы данных после завершения работы с ней через ODBC-соединение. Предикат odbc\_disconnect принимает один аргумент – соединение с базой данных, которое необходимо закрыть.

В качестве базы данных будет использоваться PostgreSQL.

**Задание**

Задание 1

На основе 3 лабораторной работы создать базу данных о заданной предметной области в виде таблиц. Таблицы должны создаваться, используя соответствующие предикаты.

Задание 2

Разработать набор sql запросов, при помощи которых можно реализовать все запросы, приведенные в варианте задания к лабораторной 3

Задание 3

Используя HTTP server libraries написать (Create/Read/Update/Delete) для всех таблиц базы данных. Предоставить пользователю возможность сбросить базу данных к исходному (тестовому) варианту заполнения.

Задание 4

Реализовать вывод результатов всех поисковых запросов к базе данных на html страницах.

Задание 5

Оформить отчет. Ответить на контрольные вопросы. Предметная область и запросы следующие:

Предметная область – видеотека. Каждая видеокассета может быть описана структурой: название фильма, год создания, киностудия, атрибуты фильма. Атрибуты фильма могут быть описаны структурой: автор сценария, режиссер, список фамилий исполнителей главных ролей, премии. Премии могут быть описаны списком из следующих структур: название фестиваля, год проведения.

Реализовать следующие типы запросов:

1. Найти сценариста, в фильме которого снялось максимальное число актеров;
2. Найти режиссеров и сценаристов, у которых есть фильмы, получившие премии;
3. Найти все премии указанного фильма;
4. Найти все фильмы, определенного актера за указанный период времени;
5. Посчитать число киностудий, которые работали с указанным режиссером.

**Описание работы программы**

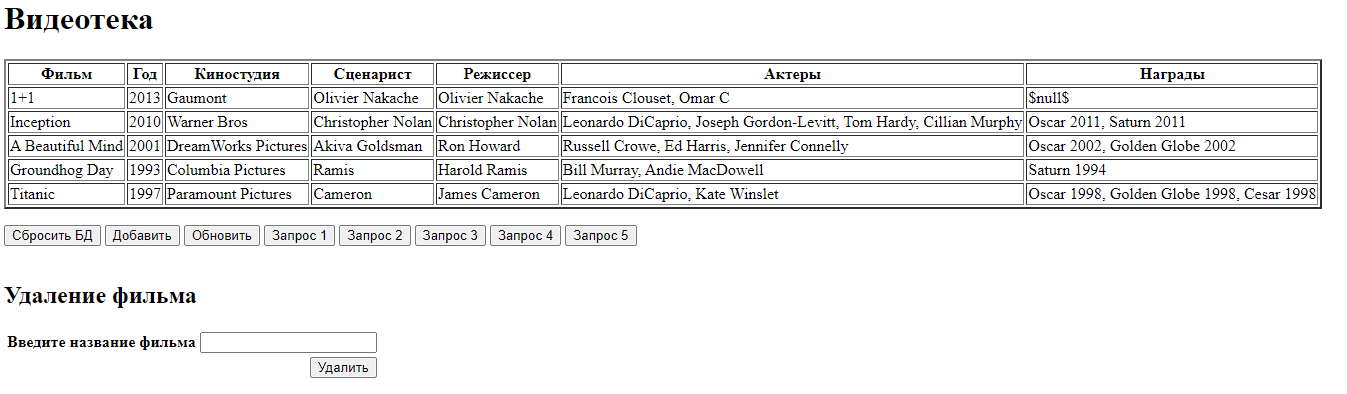
Предметная область реализована в виде таблиц videotape, film.

Создание таблиц, добавление, удаление, обновление и поиск записей реализовано с помощью средств PostgreSQL.

* Создание таблиц: CREATE TABLE название\_таблицы (название\_столбца1 тип\_данных атрибуты\_столбца1, название\_столбца2 тип\_данных атрибуты\_столбца2, ... название\_столбцаN тип\_данных атрибуты\_столбцаN, атрибуты\_таблицы);
* добавление записей: INSERT INTO имя\_таблицы (столбец1, столбец2, ... столбецN) VALUES (значение1, значение2, ... значениеN);
* удаление записей: DELETE FROM имя\_таблицы [WHERE условие\_удаления];
* обновление записей: UPDATE имя\_таблицы SET столбец1 = значение1, столбец2 = значение2, ... столбецN = значениеN [WHERE условие\_обновления];
* получение данных: SELECT список\_столбцов FROM имя\_таблицы.

**Примеры работы программы**

На рисунках 2-5 приведены скриншоты работы лабораторной работы.

  
Рисунок 2 – Пример главной html-страницы 1

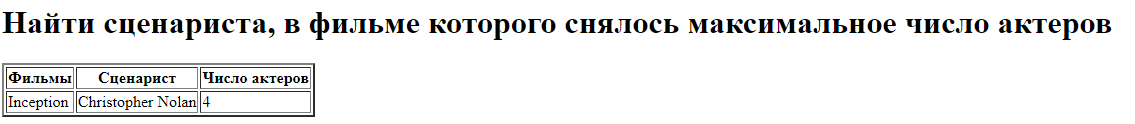
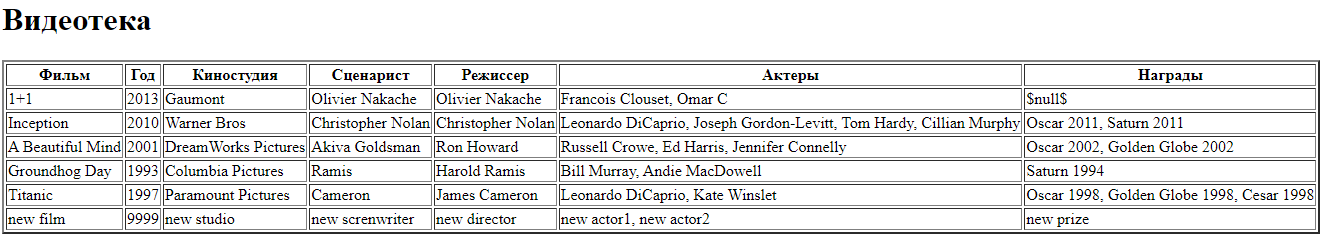
  
Рисунок 3 – Пример выполненного 1-ого запроса

  
Рисунок 4 – Пример страницы добавления записи

  
Рисунок 5 – Пример страницы с добавленной записью

**Листинг программы**

server(Port):- http\_server(http\_dispatch, [port(Port)]).

server:-server(8080).

%остановка сервера

stop(Port):- http\_stop\_server(Port, http\_dispatch).

stop:- stop(8080).

%Создание таблиц

create\_table :-

connect\_db(\_),

odbc\_query(

dsn,

'

DROP TABLE IF EXISTS public.videotape;

DROP TABLE IF EXISTS public.film;

CREATE TABLE public.film (

id integer NOT NULL PRIMARY KEY,

screenwriter varchar NULL,

director varchar NULL,

actors varchar NULL,

prize varchar NULL

);

CREATE TABLE public.videotape (

id integer NOT NULL PRIMARY KEY,

film\_name varchar NULL,

year integer NULL,

studio varchar NULL,

filmId integer REFERENCES public.film (id)

);

'

),

odbc\_disconnect(dsn).

%Сброс данных

reset\_DB(\_):-

connect\_db(Connection),

odbc\_query(

Connection,

'

DELETE FROM public.videotape;

DELETE FROM public.film;

INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (1, ''Olivier Nakache'', ''Olivier Nakache'', ''Francois Clouset, Omar C'' , NULL);

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (1, ''1+1'', 2013 , ''Gaumont'', 1);

INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (2, ''Christopher Nolan'', ''Christopher Nolan'', ''Leonardo DiCaprio, Joseph Gordon-Levitt, Tom Hardy, Cillian Murphy'', ''Oscar 2011, Saturn 2011'');

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (2, ''Inception'', 2010, ''Warner Bros'', 2);

INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (3, ''Akiva Goldsman'', ''Ron Howard'', ''Russell Crowe, Ed Harris, Jennifer Connelly'', ''Oscar 2002, Golden Globe 2002'');

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (3, ''A Beautiful Mind'', 2001, ''DreamWorks Pictures'', 3);

INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (4, ''Ramis'',''Harold Ramis'', ''Bill Murray, Andie MacDowell'', ''Saturn 1994'');

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (4, ''Groundhog Day'', 1993, ''Columbia Pictures'', 4);

INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (5, ''Cameron'', ''James Cameron'', ''Leonardo DiCaprio, Kate Winslet'', ''Oscar 1998, Golden Globe 1998, Cesar 1998'' );

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (5, ''Titanic'', 1997, ''Paramount Pictures'', 5)

'

),

odbc\_disconnect(Connection),

http\_redirect(moved, '/', \_Request).

%Обработка главной страницы

home\_page(\_Request):-

connect\_db(\_),

%Собираем название фильмов в список

findall(Name, odbc\_query(dsn, 'SELECT film\_name FROM public.videotape ORDER BY public.videotape.id ', Name), Names),

%Собираем года в список

findall(Year, odbc\_query(dsn, 'SELECT year FROM public.videotape ORDER BY public.videotape.id ', Year), Years),

%Собираем жанры в список

findall(Studio, odbc\_query(dsn, 'SELECT studio FROM public.videotape ORDER BY public.videotape.id ', Studio) , Studios),

findall(Screenwriter, odbc\_query(dsn, 'SELECT screenwriter FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId ORDER BY public.videotape.id ', Screenwriter), Screenwriters),

findall(Director, odbc\_query(dsn, 'SELECT director FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId ORDER BY public.videotape.id ', Director), Directors),

findall(Actor, odbc\_query(dsn, 'SELECT actors FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId ORDER BY public.videotape.id ', Actor), Actors),

findall(Prize, odbc\_query(dsn, 'SELECT prize FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId ORDER BY public.videotape.id ', Prize), Prizes),

odbc\_disconnect(dsn),

%Добавляем заголовки таблицы

generate\_rows(Names, Years, Studios, Screenwriters, Directors, Actors, Prizes , Rows),

ins(Rows, tr(

[

th('Фильм'),

th('Год'),

th('Киностудия'),

th('Сценарист'),

th('Режиссер'),

th('Актеры'),

th('Награды')

]

), Rows\_with\_Headers),

reply\_html\_page(

title('Видеотека'),

[

h1('Видеотека'),

table(

[border(2)],

Rows\_with\_Headers

),

form(

[style('display: inline-block'), method(post)],

p(button([type(submit), formaction(location\_by\_id('reset\_DB'))], 'Сбросить БД'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/add')], 'Добавить'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/update')], 'Обновить'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/request\_1')], 'Запрос 1'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/request\_2')], 'Запрос 2'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/request\_3')], 'Запрос 3'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/request\_4')], 'Запрос 4'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/request\_5')], 'Запрос 5'))

),

form(

[action=location\_by\_id(delete\_film), method(post)],

[

h2('Удаление фильма'),

table([

tr([th('Введите название фильма'), td(input([name(name)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Удалить'])))

])

]

)

]

).

generate\_rows([], [], [], [], [], [], [],[]).

generate\_rows([Name|Names], [Year|Years], [Gener|Geners], [ Screenwriter| Screenwriters], [Director|Directors], [Actor|Actors], [Prize|Prizes],

[tr([td(Name), td(Year), td(Gener), td(Screenwriter), td(Director), td( Actor), td(Prize) ]) |Rows]):-

generate\_rows(Names, Years, Geners, Screenwriters, Directors, Actors, Prizes, Rows).

%страница с добавлением нового фильма

add\_page(\_Request):-

reply\_html\_page(

title('Добавление студента'),

[form(

[action=location\_by\_id(add\_film), method(post)],

[

table([

tr([th('Id'), td(input([name(id), type(number)]))]),

tr([th('Фильм'), td(input([name(name)]))]),

tr([th('Год'), td(input([name(year), type(number)]))]),

tr([th('Киностудия'), td(input([name(studio)]))]),

tr([th('Сценарист'), td(input([name(screenwriter)]))]),

tr([th('Режиссер'), td(input([name(director)]))]),

tr([th('Актеры'), td(input([name(actors)]))]),

tr([th('Награды'), td(input([name(prize)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Добавить'])))

] )

]

)]

).

%Добавление нового фильма

add\_film(Request):-

http\_parameters(

Request,

[

id(Id, []),

name(Film\_name, []),

year(Year, []),

studio(Studio, []),

screenwriter(Screenwriter, []),

director(Director, []),

actors(Actors, []),

prize(Prize, [])

]

),

atom\_number(Id, Id\_num),

atom\_number(Year, Year\_num),

connect\_db(Connection),

odbc\_prepare(

Connection,

'INSERT INTO public.film (id, screenwriter, director, actors, prize) VALUES (?, ?, ?, ?, ?);

INSERT INTO public.videotape (id, film\_name, year, studio, filmId) VALUES (?, ?, ?, ?, ?)

',

[integer, varchar, varchar, varchar, varchar, integer, varchar, integer, varchar, integer ],

Statement

),

odbc\_execute(Statement, [ Id\_num, Screenwriter, Director, Actors, Prize, Id\_num, Film\_name, Year\_num, Studio, Id\_num ]),

%Закрыть prepared statement и connection

close\_conn(Connection, Statement),

http\_redirect(moved, '/', \_Request).

%Удаление фильма

delete\_film(Request):-

http\_parameters(

Request,

[

name(Film\_name, [])

]

),

connect\_db(Connection),

odbc\_prepare(

Connection,

'DELETE FROM public.videotape WHERE film\_name = (?);

',

[varchar],

Statement

),

odbc\_execute(Statement, [Film\_name]),

close\_conn(Connection, Statement),

http\_redirect(moved, '/', Request).

%добавление страницы обновления

update\_page(\_Request):-

reply\_html\_page(

title('Добавление фильма'),

[form(

[action=location\_by\_id(update\_film), method(post)],

[

table([

tr([th('Id'), td(input([name(id), type(number)]))]),

tr([th('new\_id'), td(input([name(new\_id), type(number)]))]),

tr([th('Фильм'), td(input([name(name)]))]),

tr([th('Год'), td(input([name(year), type(number)]))]),

tr([th('Киностудия'), td(input([name(studio)]))]),

%tr([th('Сценарист'), td(input([name(screenwriter)]))]),

% tr([th('Режиссер'), td(input([name(director)]))]),

% tr([th('Актеры'), td(input([name(actors), type(list)]))]),

% tr([th('Награды'), td(input([name(prize)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Обновить'])))

] )

]

)]

).

%обновление таблицы

update\_data(FilmId, Film\_name, Year, Studio, Id):-

connect\_db(Connection),

odbc\_prepare(

Connection,

'UPDATE public.videotape set id = (?), film\_name = (?), year = (?), studio = (?) WHERE id = (?)',

[integer, varchar, integer, varchar, integer],

Statement

),

odbc\_execute(Statement, [FilmId, Film\_name, Year, Studio, Id]),

close\_conn(Connection, Statement).

update\_film(Request):-

http\_parameters(

Request,

[

id(Id, []),

new\_id(New\_Id, []),

name(Film\_name, []),

year(Year, []),

studio(Studio, [])

]

),

atom\_number(Id, Id\_num),

atom\_number(New\_Id, New\_Id\_num),

atom\_number(Year, Year\_num),

connect\_db(Connection),

odbc\_prepare(

Connection,

'UPDATE public.videotape set id = (?), film\_name = (?), year = (?), studio = (?) WHERE id = (?)',

[integer, varchar, integer, varchar, integer],

Statement

),

odbc\_execute(Statement, [New\_Id\_num, Film\_name, Year\_num, Studio, Id\_num]),

close\_conn(Connection, Statement),

http\_redirect(moved, '/', \_Request).

ins(L, El, [El|L]).

close\_conn(Connection, Statement):-

odbc\_free\_statement(Statement),

odbc\_disconnect(Connection).

%ЗАПРОСЫ

%1. Найти сценариста, в фильме которого снялось максимальное число актеров;

extract\_max(row(Films, Director, Count), Films, Director, Count).

request1\_page(\_Request):-

connect\_db(\_),

odbc\_query(

dsn,

'SELECT public.videotape.film\_name, public.film.screenwriter, cardinality(string\_to\_array(public.film.actors, '', ''))

FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId

WHERE cardinality(string\_to\_array(public.film.actors, '', '')) =

(SELECT MAX(cardinality(string\_to\_array(public.film.actors, '', '')))

FROM public.videotape JOIN public.film ON public.film.id = public.videotape.filmId)',

Result

),

extract\_max(Result, Films, Screenwriter, Count),

generate\_rows1([Films],[Screenwriter], [Count], Rows ),

ins(Rows, tr(

[

th('Фильмы'),

th('Сценарист'),

th('Число актеров')

]

), Rows\_with\_Headers),

reply\_html\_page(

title('Запрос 1'),

[

h1('Найти сценариста, в фильме которого снялось максимальное число актеров'),

table(

[border(2)],

Rows\_with\_Headers

)

]).

generate\_rows1([],[],[], []).

generate\_rows1([Name|Names],[Year|Years], [Screenwriter| Screenwriters] ,[tr([td(Name), td(Year), td(Screenwriter)])|Rows]) :-

generate\_rows1(Names, Years, Screenwriters, Rows).

%2.Найти все фильмы, получившие премии, в которых снимался указанный актер

request2\_page(\_Request):-

reply\_html\_page(

title('Запрос 2'),

[

form([action=location\_by\_id(request2), method(post)],

[

h1('Найти все фильмы, получившие премии, в которых снимался указанный актер

'),

table([

tr([th('Актер'), td(input([name(actor)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Найти'])))

] )

]

)

]

).

extract\_row1(row(FilmId), FilmId).

processed\_fetch1(Statement, Rows ) :-

odbc\_fetch(Statement, Row, next),

( Row == end\_of\_file ->( true, reply\_html\_page(

title('Запрос 2'),

[

table(

[border(2)],

Rows)

])

);

(

extract\_row1(Row, Film\_name),

ins(Rows, tr(

[

td(Film\_name)

]

), New\_Rows)

),

processed\_fetch1(Statement, New\_Rows)

).

request2(Request):-

http\_parameters(

Request,

[

actor(Actor, [])

]

),

connect\_db(Connection),

odbc\_prepare(

Connection,

'SELECT public.videotape.film\_name

FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId

WHERE public.film.actors like ''%'' || (?) || ''%'' AND public.film.prize IS NOT NULL

ORDER BY public.videotape.id',

[varchar],

Statement,

[fetch(fetch)]

),

odbc\_execute(Statement, [Actor]),

reply\_html\_page(

title(' все фильмы, получившие премии, в которых снимался указанный актер ~w.' -Actor),

[

h1('Фильмы с наградами, где была роль у ~w.' -Actor)

]),

processed\_fetch1(Statement, []),

close\_conn(Connection, Statement).

%3. Найти все фильмы определенного сценариста, снятые в указанном году.

request3\_page(\_Request):-

reply\_html\_page(

title('Запрос 3'),

[

form([action=location\_by\_id(request3), method(post)],

[

h1('Все фильмы определенного сценариста, снятые в указанном году'),

table([

tr([th('Год'), td(input([name(year)]))]),

tr([th('Сценарист'), td(input([name(screenwriter)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Найти'])))

] )

]

)

]

).

extract\_row3(row(Film\_name, Year,Screenwriter), Film\_name, Year,Screenwriter).

processed\_fetch3(Statement, Rows ) :-

odbc\_fetch(Statement, Row, next),

( Row == end\_of\_file ->( true, ins(Rows, tr(

[

th('Фильмы'),

th('Год'),

th('Сценарист')

]

), Rows\_with\_Headers),

reply\_html\_page(

title('Запрос 3'),

[

table(

[border(2)],

Rows\_with\_Headers)

]));

(

extract\_row3(Row, Film\_name, Year,Screenwriter ),

ins(Rows, tr(

[

td(Film\_name),

td(Year),

td(Screenwriter)

]

), New\_Rows)

),

processed\_fetch3(Statement, New\_Rows)

).

request3(Request):-

http\_parameters(

Request,

[

year(Year, []),

screenwriter(Screenwriter, [])

]

),

atom\_number(Year, Year\_num),

connect\_db(Connection),

odbc\_prepare(

Connection,

'SELECT public.videotape.film\_name, public.videotape.year, public.film.screenwriter

FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId

WHERE public.film.screenwriter = (?) AND public.videotape.year = (?)

ORDER BY public.videotape.id',

[varchar, integer],

Statement,

[fetch(fetch)]

),

odbc\_execute(Statement, [Screenwriter, Year\_num ]),

reply\_html\_page(

title('Запрос 3'),

[

h1('Все фильмы определенного сценариста, снятые в указанном году')

]),

processed\_fetch3(Statement, []),

%Закрыть prepared statement и connection

close\_conn(Connection, Statement).

processed\_fetch\_actorsnum(Statement, Rows) :-

odbc\_fetch(Statement, Row, next),

( Row == end\_of\_file -> (true, ins(Rows, tr(

[

th('Количество актеров на студии')

]

), Rows\_with\_Headers),

reply\_html\_page(

title(' '),

[

table(

[border(2)],

Rows\_with\_Headers)

])

);

(

extract\_row2(Row, NumActors ),

ins(Rows, tr(

[

td(NumActors) ]

), New\_Rows)

),

processed\_fetch\_actorsnum(Statement, New\_Rows )

).

extract\_row2(row(Actors), Actors ).

%4. Посчитать количество актеров, снимавшихся на определенной киностудии;

request4\_page(\_Request):-

reply\_html\_page(

title('Запрос 4'),

[

form([action=location\_by\_id(request4), method(post)],

[

h1(' Kоличество актеров, снимавшихся на определенной киностудии'),

table([

tr([th('Киностудия'), td(input([name(studio)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Найти'])))

] )

]

)

]

).

request4(Request):-

http\_parameters(

Request,

[

studio(Studio, [])

]

),

connect\_db(Connection),

odbc\_prepare(

Connection,

'SELECT SUM(array\_size)

FROM

(SELECT cardinality(string\_to\_array(public.film.actors, '', '')) AS array\_size

FROM public.videotape JOIN public.film ON public.film.id = public.videotape.filmId

WHERE public.videotape.studio = (?)) AS array\_size',

[varchar],

Statement,

[fetch(fetch)]

),

odbc\_execute(Statement, [Studio]),

reply\_html\_page(

title('Запрос 4'),

[

h1(' Kоличество актеров, снимавшихся на определенной киностудии ~w' - Studio)

]),

processed\_fetch\_actorsnum(Statement, []),

%Закрыть prepared statement и connection

close\_conn(Connection, Statement) .

processed\_fetch\_actors(Statement, Rows ) :-

odbc\_fetch(Statement, Row, next),

( Row == end\_of\_file -> (true,

ins(Rows, tr( [ th('Актеры') ] ), Rows\_with\_Headers),

reply\_html\_page(

title(' '),

[

table(

[border(2)],

Rows\_with\_Headers)

]));

(

extract\_row2(Row, Actors),

ins(Rows, tr(

[

td(Actors) ]

), New\_Rows)

),

processed\_fetch\_actors(Statement,New\_Rows )

).

%5. Найти всех актеров, снимавшихся в фильмах определенного сценариста.

request5\_page(\_Request):-

reply\_html\_page(

title('Запрос 5'),

[

form([action=location\_by\_id(request5), method(post)],

[

h1( 'Все актеры, снимавшиеся в фильмах сценариста'),

table([

tr([th('Сценарист'), td(input([name(screenwriter)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Найти'])))

] )

]

)

]

).

request5(Request):-

http\_parameters(

Request,

[

screenwriter(Screenwriter, [])

]

),

connect\_db(Connection),

odbc\_prepare(

Connection,

'SELECT public.film.actors

FROM public.videotape JOIN public.film ON public.film.id= public.videotape.filmId

WHERE public.film.screenwriter = (?)',

[varchar],

Statement,

[fetch(fetch)]

),

odbc\_execute(Statement, [Screenwriter]),

reply\_html\_page(

title('Запрос 5'),

[

h1( 'Все актеры, снимавшиеся в фильмах сценариста ~w '- Screenwriter)

]),

processed\_fetch\_actors(Statement, []),

%Закрыть prepared statement и connection

close\_conn(Connection, Statement).