Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Тверской государственный университет»

Математический факультет

Направление «Компьютерная безопасность»

Кафедра компьютерной безопасности и математических методов управления

Индивидуальный проект

по дисциплине “Основы построения защищенных баз данных”

Построение защищенной базы данных и применение модели разграничения доступа к базе данных

Кинопоиск

Работу выполнили:

Студентка V курса, группы М-55

Мамонова Мария Александровна

Студент V курса, группы М-54

Коханчик Антон Дмитриевич

Проверила:

Доцент Кафедры КБ и ММУ Цирулева Валентина Михайловна

Тверь, 2017 г.

Содержание

[E-R диаграмма 3](#_Toc501617014)

[Запросы к базе данных 4](#_Toc501617015)

[Представления 4](#_Toc501617016)

[Процедуры и функции 6](#_Toc501617017)

[Триггеры 8](#_Toc501617018)

[Модели доступа 12](#_Toc501617019)

[Ролевая модель доступа 12](#_Toc501617020)

[1. Словесное описание и определение ролей и наборов их привилегий к объектам базы данных 12](#_Toc501617021)

[2. Схема ролей и их привилегий 13](#_Toc501617022)

[3. Иерархия ролей 13](#_Toc501617023)

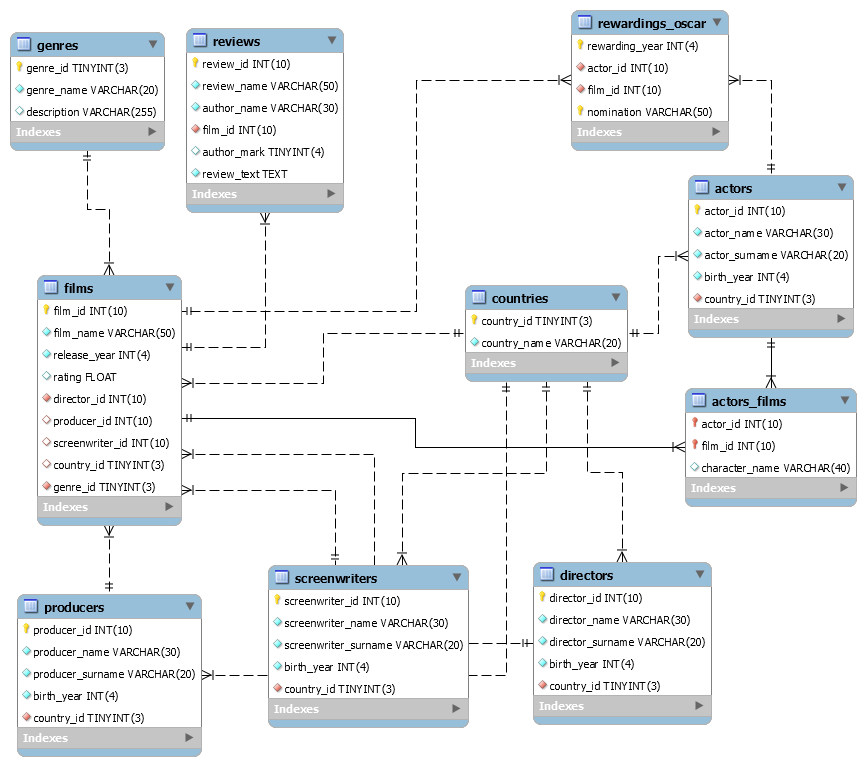
[4. Имитация ролевой модели с помощью дискреционной модели разграничения доступа в MySQL 14](#_Toc501617024)

[Мандатная модель разграничения доступа 17](#_Toc501617025)

[1. Словесное описание 17](#_Toc501617026)

[2. Реализация. 17](#_Toc501617027)

E-R диаграмма



Запросы к базе данных

# Представления

*1. Отобрать все фильмы, выпущенные позже 2007 года. Представление с условием проверки.*

create view newFilms as select \* from films where release\_year > 2007 order by release\_year with local check option;  
select \* from newFilms;

Результат:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| film\_id | film\_name | release\_year | rating | director\_id | producer\_id | screenwriter\_id | country\_id | genre\_id |
| 111543 | Темный рыцарь | 2008 | 8.502 | 41477 | 41477 | 41477 | 1 | 2 |
| 325439 | Чтец | 2008 | 7.935 | 10346 | 65747 | 36297 | 1 | 8 |
| 463647 | Железная леди | 2011 | 7.013 | 1190481 | 15755 | 1190793 | 3 | 22 |
| 470179 | Красная Шапочка | 2011 | 6.126 | 30189 | 37859 | 1047677 | 1 | 1 |
| 477764 | Дети шпионов 4D | 2011 | 3.695 | 30966 | 30966 | 30966 | 1 | 2 |
| 437410 | Темный рыцарь: Возрождение легенды | 2012 | 8.144 | 41477 | 41477 | 41477 | 1 | 2 |
| 461981 | Одинокий рейнджер | 2013 | 6.791 | 30870 | 10207 | 226493 | 1 | 13 |
| 522941 | Выживший | 2015 | 7.814 | 563585 | 6784 | 734897 | 1 | 4 |

*2. Отобрать количество фильмов, снятых на территории США в период 1980-2017 год*

create view americanFilms as select count(film\_id) from films where (release\_year between 1995 and 2000) and country\_id in (select country\_id from countries where country\_name = "США");  
select \* from americanFilms;

Результат:

|  |
| --- |
| count(film\_id) |
| 2 |

*3. Вывести Топ-7 фильмов*

create view TopFilms as select film\_name, rating from films order by rating desc limit 7;  
select \* from TopFilms;

Результат:

|  |  |
| --- | --- |
| film\_name | rating |
| Властелин колец: Возвращение Короля | 8.611 |
| Темный рыцарь | 8.502 |
| Матрица | 8.492 |
| Десятое королевство (сериал) | 8.457 |
| Титаник | 8.371 |
| Пираты Карибского моря:Проклятие Черной жемчужины | 8.342 |
| Гарри Поттер и философский камень | 8.193 |

*4. Вывести авторов статей, которые всегда ставят оценки ниже рейтинга фильма*

create view badUsers as select distinct author\_name from reviews as t1 where not exists(select author\_mark from reviews as t3 where author\_mark > (select rating from films as t2 where t3.film\_id = t2.film\_id) and t1.author\_name = t3.author\_name);  
select \* from badUsers;

Результат:

|  |
| --- |
| author\_name |
| 1mira1 |
| Cas\_000 |

# Процедуры и функции

*1. Отобрать всех актеров, играющих в заданном фильме N*

create procedure actorFromFilm (film varchar(50))   
select actor\_name, actor\_surname from actors natural join actors\_films where film\_id = (select film\_id from films where film\_name = film);  
call actorFromFilm("Индиана Джонс: В поисках утраченного ковчега");

Результат:

|  |  |
| --- | --- |
| actor\_name | actor\_surname |
| Карен | Аллен |
| Харрисон | Форд |
| Рональд | Лейси |

*2. Отобрать количество фильмов, выпущенные с Т1 по Т2 годы*

delimiter //  
create function numberFilmsInTimeInterval (t1 int(4), t2 int(4))  
returns int  
begin

declare num int;  
select count(\*) into num from films where release\_year between t1 and t2;  
 return num;

end//  
delimiter ;  
select numberFilmsInTimeInterval(1990, 2000);

Результат:

|  |
| --- |
| numberFilmsInTimeInterval(1990, 2000) |
| 5 |

*3. Вывести количество наград, полученные заданным актером*

delimiter //  
create function rewardingsOfActor (actor varchar(55))  
returns int  
begin

declare num int;  
select count(\*) as количество into num from rewardings\_Oscar as t1 natural join films where exists(select \* from actors\_films where actors\_films.film\_id = t1.film\_id and exists(select \* from actors where concat(actor\_name, " ", actor\_surname) = actor and actors.actor\_id = actors\_films.actor\_id));  
return num;

end//  
delimiter ;  
select rewardingsOfActor("Мэрил Стрип");

Результат:

|  |
| --- |
| rewardingsOfActor("Мэрил Стрип") |
| 2 |

*4. Вывести для заданного актера из списка его фильмов тот, у которого наибольший рейтинг*

create procedure bestFilmOfActor (actor varchar(55), out film varchar(50))  
select film\_name into film from actors inner join actors\_films as t1 on(actors.actor\_id = t1.actor\_id) inner join films on(t1.film\_id = films.film\_id) where concat(actor\_name, " ", actor\_surname) = actor group by t1.actor\_id;  
call bestFilmOfActor("Хит Леджер", @film);  
select @film;

Результат:

|  |
| --- |
| @film |
| Темный рыцарь |

# Триггеры

*1. Создать триггер, который при вставке актера в таблицу Актёры проверяет наличие указанной страны, и если та не указана, ставит страну 9 (Неизвестно)*

delimiter //  
create trigger insert\_check before insert on actors  
for each row

begin  
if new.country\_id is null then  
set new.country\_id = (select country\_id from countries where country\_name = 'Неизвестно');  
end if;  
end //

delimiter ;  
insert into actors value (1, "Джонни", "Пробный", 2017, null);

Результат:  


*2. Создать триггер для вставки в таблицу Страны, который бы переводил первую букву названия страны в верхний регистр*

delimiter //  
create trigger upper before insert on countries  
for each row

begin  
set new.country\_name = CONCAT(upper(left(new.country\_name,1)), substr(new.country\_name,2));  
end //

delimiter ;  
insert into countries value (14, "исландия");

Результат:  

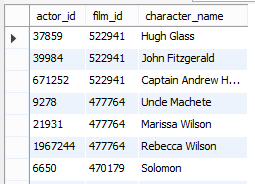

*2. Создать триггер для каскадного удаления фильма*

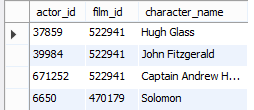
delimiter //  
create trigger before\_delete\_film before delete on films  
for each row

begin  
delete from reviews where film\_id = old.film\_id;  
delete from actors\_films where film\_id = old.film\_id;  
end //

delimiter ;  
delete from films where film\_name = "Дети шпионов 4D";

Результат:

До  


После  


*3. Создать триггер, который проверяет рецензии на наличие нецензурных слов (для примера используется слово "редиска") и заменяет их (его) на "нехороший человек"*

delimiter //  
create trigger before\_insert\_reviews before insert on reviews  
for each row

begin  
if new.review\_text like "%редиска%" then  
set new.review\_text = replace(new.review\_text, "редиска", "нехороший человек");  
end if;  
end//

delimiter ;  
insert into reviews value (9, "la-la", "anonimus", 2213, 9, "Ок, только режисер - редиска, убил Джека. Он точно редиска");

Результат:  


*4. Создать таблицу Люди, содержащую поле "деятельность". Теперь данные о всех актерах, режиссерах, продюсерах, сценаристах вставляются в эту таблицу. Создать триггер, который будет при вставке в таблицу Люди переносить данные о новом человеке в нужную таблицу.*

create table People(  
 p\_id int(10),  
 p\_name varchar(30),  
 p\_surname varchar(20),  
 p\_year int(4),  
 p\_country tinyint(3),  
 p\_special varchar(15) )CHARACTER SET = UTF8;

delimiter //  
create trigger after\_insert\_people after insert on people  
for each row

begin  
if new.p\_special = "актер" then  
insert into actors value (new.p\_id, new.p\_name, new.p\_surname, new.p\_year, new.p\_country);  
end if;  
if new.p\_special = "режиссер" then  
insert into directors value (new.p\_id, new.p\_name, new.p\_surname, new.p\_year, new.p\_country);  
end if;  
if new.p\_special = "сценарист" then  
insert into screenwriters value (new.p\_id, new.p\_name, new.p\_surname, new.p\_year, new.p\_country);  
end if;  
if new.p\_special = "продюсер" then  
insert into producers value (new.p\_id, new.p\_name, new.p\_surname, new.p\_year, new.p\_country);  
end if;  
end //

delimiter ;  
insert into people values  
(3, "иван", "реж", 2017, 9, "режиссер"),  
(3, "Петр", "Прод", 2017, 9, "продюсер");

Результат:  
  


*11. Создать триггер, который при удалении из таблицы Актеры заносит данные об удаляемом в таблицу Архив*

create table archive(  
 date\_operation datetime,  
 name\_operation varchar(20),  
 d\_id int(10),  
 d\_name varchar(30),  
 d\_surname varchar(20),  
 d\_year int(4),  
 d\_country tinyint(3))CHARACTER SET = UTF8;

delimiter //  
create trigger before\_delete\_actors before delete on actors  
for each row

begin  
insert into archive value(now(), "delete", old.actor\_id, old.actor\_name, old.actor\_surname, old.birth\_year, old.country\_id);  
end//

delimiter ;  
delete from actors where actor\_id = 1;

Результат:  


Модели доступа

# Ролевая модель доступа

## 1. Словесное описание и определение ролей и наборов их привилегий к объектам базы данных

Введем следующие роли: Администратор (admin), Модератор (moderator), Автор (author), Редактор статей (editor), Пользователь (customer).

Определим набор их привилегий по отношению к базе данных Кинопоиск.

Администратор имеет все возможные права: может создавать новые записи по всем таблицам, просматривать все таблицы, изменять их, удалять записи из них и сами таблицы, менять привилегии других ролей.

Автор может: создавать новые записи по таблице reviews, просматривать эту таблицу.

Редактор статей может: просматривать, изменять, добавлять и удалять записи из таблицы reviews.

Модератор может: создавать новые записи по таблицам actors, actors\_films, films, rewarding\_Oscar, directors, screenwriters, producers, изменять эти таблицы, просматривать их.

Пользователь может только просматривать все таблицы.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | admin | moderator | customer | editor | author |
| actors | i,s,u,d,g | s,u,i | s | - | - |
| films | i,s,u,d,g | s,u,i | s | - | - |
| actors\_films | i,s,u,d,g | s,u,i | s | - | - |
| countries | i,s,u,d,g | - | s | - | - |
| rewardings\_oscar | i,s,u,d,g | s,u,i | s | - | - |
| directors | i,s,u,d,g | s,u,i | s | - | - |
| screenwriters | i,s,u,d,g | s,u,i | s | - | - |
| producers | i,s,u,d,g | s,u,i | s | - | - |
| reviews | i,s,u,d,g | - | s | i,s,u,g | i,s |

## 2. Схема ролей и их привилегий

## 3. Иерархия ролей

Иерархия ролей данной модели следующая:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | admin | moderator | customer | editor | author |
| actors | i,s,u,d,g | s,u,i | s | - | - |
| films | i,s,u,d,g | s,u,i | s | - | - |
| actors\_films | i,s,u,d,g | s,u,i | s | - | - |
| countries | i,s,u,d,g | - | s | - | - |
| rewardings\_oscar | i,s,u,d,g | s,u,i | s | - | - |
| directors | i,s,u,d,g | s,u,i | s | - | - |
| screenwriters | i,s,u,d,g | s,u,i | s | - | - |
| producers | i,s,u,d,g | s,u,i | s | - | - |
| reviews | i,s,u,d,g | - | s | i,s,u,g | i,s |

## 4. Имитация ролевой модели с помощью дискреционной модели разграничения доступа в MySQL

*Создание «ролей»:*

create user admin identified by 'admin';  
create user moderator identified by 'moderator';  
create user editor identified by 'editor';  
create user author identified by 'author';  
create user customer identified by 'customer';

*Просмотр таблицы пользователей. Проверяем, что все «рол»и создались.*

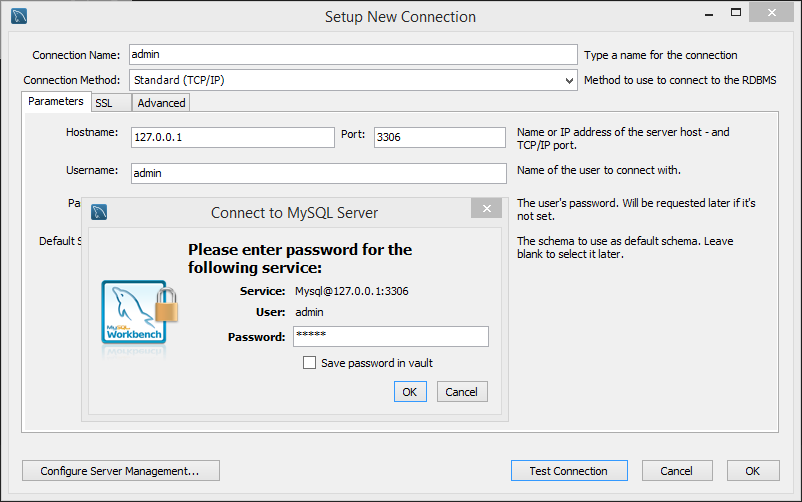
select user from mysql.user;

|  |
| --- |
| user |
| admin |
| author |
| customer |
| editor |
| moderator |
| mysql.sys |
| root |

*Наделение «роли» admin заявленными привилегиями*

grant all on films.\* to admin;  
grant grant option on films.\* to admin;

*Авторизация под «ролью» admin.*



*От имени admin даем заявленные привилегии «роли» moderator*

use films;

grant select, update, insert on films.actors to moderator;  
grant select, update, insert on films.actors\_films to moderator;  
grant select, update, insert on films.films to moderator;  
grant select, update, insert on films.rewardings\_Oscar to moderator;  
grant select, update, insert on films.directors to moderator;  
grant select, update, insert on films.screenwriters to moderator;  
grant select, update, insert on films.producers to moderator;

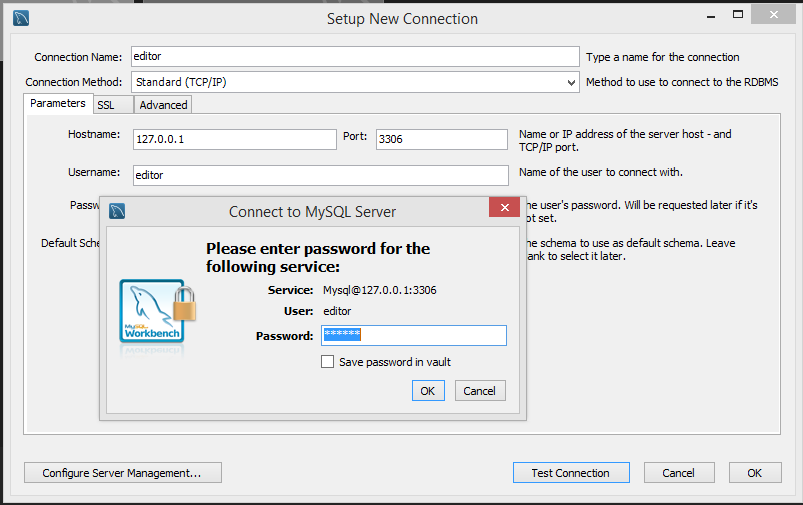
*От имени admin даем заявленные привилегии «роли» editor*

grant select, update, insert, grant option on films.reviews to editor;

*От имени admin даем заявленные привилегии «роли» customer*

grant select on films.\* to customer;

*Авторизация под «ролью» editor*



*От имени editor даем заявленные привилегии «роли» author*

use films;  
grant select, insert on films.reviews to author;

*Просмотр всех «ролей» с их привилегиями*

select Db, User, Grantor, Table\_name, Table\_priv from mysql.tables\_priv where Db != 'sys';

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Db | User | Grantor | Table\_name | Table\_priv |
| films | moderator | admin@localhost | directors | Select,Insert,Update |
| films | moderator | admin@localhost | actors | Select,Insert,Update |
| films | moderator | admin@localhost | rewardings\_oscar | Select,Insert,Update |
| films | moderator | admin@localhost | films | Select,Insert,Update |
| films | moderator | admin@localhost | actors\_films | Select,Insert,Update |
| films | moderator | admin@localhost | screenwriters | Select,Insert,Update |
| films | moderator | admin@localhost | producers | Select,Insert,Update |
| films | editor | admin@localhost | reviews | Select,Insert,Update,Grant |
| films | author | editor@localhost | reviews | Select,Insert |

# Мандатная модель разграничения доступа

## 1. Словесное описание

Рассмотрим применение данной модели на примере таблицы films.

Введем следующие роли с уровнями секретности:

|  |  |
| --- | --- |
| Роль | Уровень секретности (доступа) |
| Администратор (admin) | 2 |
| Пользователь (customer) | 1 |

## 2. Реализация.

Таким образом, Администратор имеет доступ к информации уровня 2 и 1, а Пользователь – к информации уровня 1.

Допустим (в рамках примера), что Пользователь не должен получать доступ к фильмам, снятым не в США. Тогда таблица films после выполнения кода

alter table films add access tinyint(2);  
update films set access = 1 where country\_id = 1;  
update films set access = 2 where country\_id != 1;

будет выглядеть следующим образом:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| film\_id | film\_name | release\_year | rating | director\_id | producer\_id | screenwriter\_id | country\_id | genre\_id | access |
| 301 | Матрица | 1999 | 8.492 | 23330 | 1790 | 23329 | 1 | 2 | 1 |
| 339 | Индиана Джонс: В поисках утраченного ковчега | 1981 | 8.007 | 22260 | 20397 | 26537 | 1 | 10 | 1 |
| 688 | Гарри Поттер и Тайная комната | 2002 | 8.03 | 24817 | 23449 | 10093 | 3 | 5 | 2 |
| 689 | Гарри Поттер и философский камень | 2001 | 8.193 | 24817 | 23449 | 10093 | 3 | 5 | 2 |
| 1902 | Крамер против Крамера | 1979 | 8.011 | 31799 | 14197 | 31799 | 1 | 8 | 1 |
| 2213 | Титаник | 1997 | 8.371 | 27977 | 27977 | 27977 | 1 | 8 | 1 |
| 3498 | Властелин колец: Возвращение Короля | 2003 | 8.611 | 32383 | 32383 | 32385 | 1 | 5 | 1 |
| 4374 | Пираты Карибского моря:Проклятие Черной жемчужины | 2003 | 8.342 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 5198 | Кошмар на улице Вязов | 1984 | 7.703 | 8442 | 3023 | 8442 | 1 | 1 | 1 |
| 5908 | Кинг Конг | 2005 | 7.573 | 32383 | 539928 | 32385 | 2 | 8 | 2 |
| 6226 | Кукловоды | 1994 | 6.438 | 104876 | 8288 | 30871 | 1 | 1 | 1 |
| 16686 | Блеск | 1996 | 7.87 | 29361 | 90143 | 90096 | 4 | 8 | 2 |
| 63991 | Пираты Карибского моря: Сундук мертвеца | 2006 | 8.098 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 77217 | Десятое королевство (сериал) | 1999 | 8.457 | 64418 | 78017 | 55143 | 3 | 5 | 2 |
| 81924 | 300 спартанцев | 2007 | 7.629 | 462603 | 8523 | 462603 | 1 | 5 | 1 |
| 111543 | Темный рыцарь | 2008 | 8.502 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 325439 | Чтец | 2008 | 7.935 | 10346 | 65747 | 36297 | 1 | 8 | 1 |
| 437410 | Темный рыцарь: Возрождение легенды | 2012 | 8.144 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 461981 | Одинокий рейнджер | 2013 | 6.791 | 30870 | 10207 | 226493 | 1 | 13 | 1 |
| 463647 | Железная леди | 2011 | 7.013 | 1190481 | 15755 | 1190793 | 3 | 22 | 2 |
| 470179 | Красная Шапочка | 2011 | 6.126 | 30189 | 37859 | 1047677 | 1 | 1 | 1 |
| 477764 | Дети шпионов 4D | 2011 | 3.695 | 30966 | 30966 | 30966 | 1 | 2 | 1 |
| 522941 | Выживший | 2015 | 7.814 | 563585 | 6784 | 734897 | 1 | 4 | 1 |

Следовательно, дальнейший доступ к данным таблицы может осуществляться следующим образом:

create user admin, customer;  
grant select on films to customer;  
grant all on films to admin;

create view for\_admin as  
 select \* from films where access <= 2 with local check option;  
create view for\_user as  
 select \* from films where access <= 1 with local check option;

delimiter //  
create procedure see\_films (user\_name varchar(50))  
begin  
 if user\_name = 'admin' then  
 select \* from for\_admin;  
 end if;  
 if user\_name = 'customer' then  
 select \* from for\_user;  
 end if;  
end //  
delimiter ;

call see\_films('customer');

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| film\_id | film\_name | release\_year | rating | director\_id | producer\_id | screenwriter\_id | country\_id | genre\_id | access |
| 301 | Матрица | 1999 | 8.492 | 23330 | 1790 | 23329 | 1 | 2 | 1 |
| 339 | Индиана Джонс: В поисках утраченного ковчега | 1981 | 8.007 | 22260 | 20397 | 26537 | 1 | 10 | 1 |
| 1902 | Крамер против Крамера | 1979 | 8.011 | 31799 | 14197 | 31799 | 1 | 8 | 1 |
| 2213 | Титаник | 1997 | 8.371 | 27977 | 27977 | 27977 | 1 | 8 | 1 |
| 3498 | Властелин колец: Возвращение Короля | 2003 | 8.611 | 32383 | 32383 | 32385 | 1 | 5 | 1 |
| 4374 | Пираты Карибского моря:Проклятие Черной жемчужины | 2003 | 8.342 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 5198 | Кошмар на улице Вязов | 1984 | 7.703 | 8442 | 3023 | 8442 | 1 | 1 | 1 |
| 6226 | Кукловоды | 1994 | 6.438 | 104876 | 8288 | 30871 | 1 | 1 | 1 |
| 63991 | Пираты Карибского моря: Сундук мертвеца | 2006 | 8.098 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 81924 | 300 спартанцев | 2007 | 7.629 | 462603 | 8523 | 462603 | 1 | 5 | 1 |
| 111543 | Темный рыцарь | 2008 | 8.502 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 325439 | Чтец | 2008 | 7.935 | 10346 | 65747 | 36297 | 1 | 8 | 1 |
| 437410 | Темный рыцарь: Возрождение легенды | 2012 | 8.144 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 461981 | Одинокий рейнджер | 2013 | 6.791 | 30870 | 10207 | 226493 | 1 | 13 | 1 |
| 470179 | Красная Шапочка | 2011 | 6.126 | 30189 | 37859 | 1047677 | 1 | 1 | 1 |
| 477764 | Дети шпионов 4D | 2011 | 3.695 | 30966 | 30966 | 30966 | 1 | 2 | 1 |
| 522941 | Выживший | 2015 | 7.814 | 563585 | 6784 | 734897 | 1 | 4 | 1 |

call see\_films('admin');

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| film\_id | film\_name | release\_year | rating | director\_id | producer\_id | screenwriter\_id | country\_id | genre\_id | access |
| 301 | Матрица | 1999 | 8.492 | 23330 | 1790 | 23329 | 1 | 2 | 1 |
| 339 | Индиана Джонс: В поисках утраченного ковчега | 1981 | 8.007 | 22260 | 20397 | 26537 | 1 | 10 | 1 |
| 688 | Гарри Поттер и Тайная комната | 2002 | 8.03 | 24817 | 23449 | 10093 | 3 | 5 | 2 |
| 689 | Гарри Поттер и философский камень | 2001 | 8.193 | 24817 | 23449 | 10093 | 3 | 5 | 2 |
| 1902 | Крамер против Крамера | 1979 | 8.011 | 31799 | 14197 | 31799 | 1 | 8 | 1 |
| 2213 | Титаник | 1997 | 8.371 | 27977 | 27977 | 27977 | 1 | 8 | 1 |
| 3498 | Властелин колец: Возвращение Короля | 2003 | 8.611 | 32383 | 32383 | 32385 | 1 | 5 | 1 |
| 4374 | Пираты Карибского моря:Проклятие Черной жемчужины | 2003 | 8.342 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 5198 | Кошмар на улице Вязов | 1984 | 7.703 | 8442 | 3023 | 8442 | 1 | 1 | 1 |
| 5908 | Кинг Конг | 2005 | 7.573 | 32383 | 539928 | 32385 | 2 | 8 | 2 |
| 6226 | Кукловоды | 1994 | 6.438 | 104876 | 8288 | 30871 | 1 | 1 | 1 |
| 16686 | Блеск | 1996 | 7.87 | 29361 | 90143 | 90096 | 4 | 8 | 2 |
| 63991 | Пираты Карибского моря: Сундук мертвеца | 2006 | 8.098 | 30870 | 10207 | 30871 | 1 | 5 | 1 |
| 77217 | Десятое королевство (сериал) | 1999 | 8.457 | 64418 | 78017 | 55143 | 3 | 5 | 2 |
| 81924 | 300 спартанцев | 2007 | 7.629 | 462603 | 8523 | 462603 | 1 | 5 | 1 |
| 111543 | Темный рыцарь | 2008 | 8.502 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 325439 | Чтец | 2008 | 7.935 | 10346 | 65747 | 36297 | 1 | 8 | 1 |
| 437410 | Темный рыцарь: Возрождение легенды | 2012 | 8.144 | 41477 | 41477 | 41477 | 1 | 2 | 1 |
| 461981 | Одинокий рейнджер | 2013 | 6.791 | 30870 | 10207 | 226493 | 1 | 13 | 1 |
| 463647 | Железная леди | 2011 | 7.013 | 1190481 | 15755 | 1190793 | 3 | 22 | 2 |
| 470179 | Красная Шапочка | 2011 | 6.126 | 30189 | 37859 | 1047677 | 1 | 1 | 1 |
| 477764 | Дети шпионов 4D | 2011 | 3.695 | 30966 | 30966 | 30966 | 1 | 2 | 1 |
| 522941 | Выживший | 2015 | 7.814 | 563585 | 6784 | 734897 | 1 | 4 | 1 |

Второй вариант написания процедуры:

delimiter //  
create procedure see\_films ()  
begin  
 if (SELECT SUBSTRING\_INDEX(USER(),'@',1)) = 'admin' then  
 select \* from for\_admin;  
 end if;  
 if (SELECT SUBSTRING\_INDEX(USER(),'@',1)) = 'customer' then  
 select \* from for\_user;  
 end if;  
end //  
delimiter ;