**Курсовая работа**

«Описание модели хранения данных на примере

популярного веб-сайта: Кинопоиск»

Created by:

Иванов Р.Ю

**Общее текстовое описание БД и решаемых ею задач.**

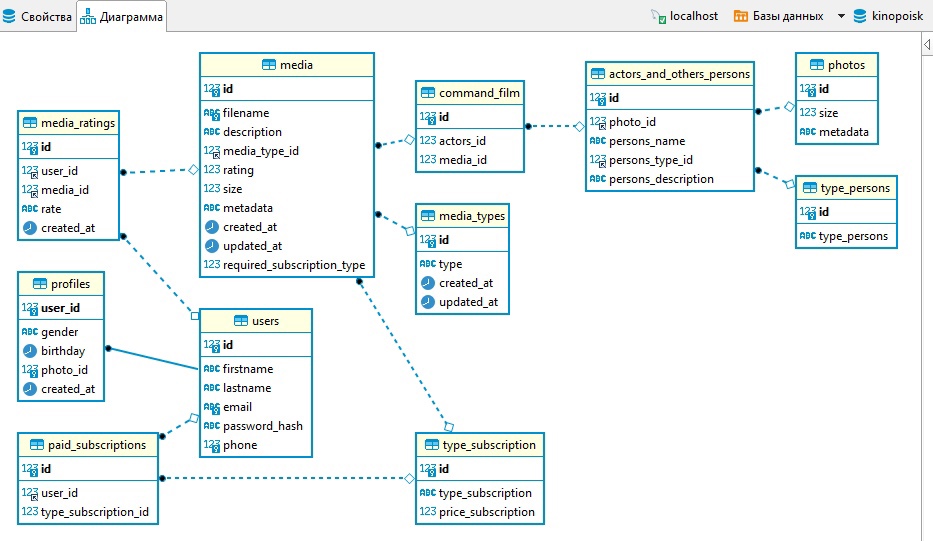
На основе популярно веб-сайта «Кинопоиск» была спроектирована реляционная база данных с использованием СУБД «MySQL».

Данная БД хранит в себе сведения о пользователях, медиа-файлах (фильмы, сериалы и т.п.), платных подписках и командном составе фильма.

С помощью, спроектированной БД можно получить информацию об актерском составе фильма, о платных подписках и какой пользователь имеет тип подписки.

Также есть возможность делать проверку на соответствие возрастного рейтинга и самого возраста пользователя.

**ER-Diagram:**



С помощью этой базы данных решим задачу: «Вывести имя пользователя, его возраст, какая у него подписка и ее стоимость» используя JOIN составим запрос SQL:

**SELECT** **CONCAT**(u.firstname, ' ', u.lastname) **as** Name,

**YEAR**(**CURRENT\_DATE**) - **YEAR**(p.birthday) **AS** `Year`,

tp.type\_subscription **AS** `Type`,

tp.price\_subscription **AS** Price

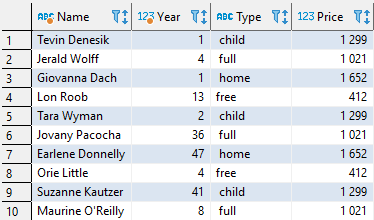
**FROM** users **AS** u

**JOIN** profiles **AS** p **on** u.id = p.user\_id

**JOIN** paid\_subscriptions **AS** pd **on** pd.user\_id = u.id

**JOIN** type\_subscription **AS** tp **on** tp.id = pd.type\_subscription\_id;

После выполнения которого, получим результат:



Так же можешь отсортировать пользователей от самого старшего до самого младшего, добавив в конец запроса **ORDER** **BY** `Year` **DESC**;

Благодаря агрегации данных можно найти, сколько и какие именно фильмы относятся к жанру «Action», для этого создадим запрос:

**SELECT** **COUNT**(media.filename) Count\_films,

**GROUP\_CONCAT**(media.filename **SEPARATOR** ', ')

Films, mt.`type`

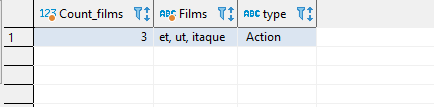
**FROM** media

**JOIN** media\_types mt **ON** mt.id = media.media\_type\_id

**WHERE** mt.id = 1

**GROUP** **BY** media.media\_type\_id;

После его выполнения получаем такой результат:



**Представления**

Основными структурными единицами в реляционных базах данных являются таблицы. Однако язык запросов SQL предоставляет еще один способ организации данных — представления.

Представление — это запрос на выборку (**SELECT**), которому присваивается уникальное имя и который можно сохранять или удалять из базы данных как обычную хранимую процедуру.

Поместим наш первый запрос в представление:

**CREATE** **OR** **REPLACE** **VIEW** name\_info **AS**

**SELECT** **CONCAT**(u.firstname, ' ', u.lastname) **as** Name,

**YEAR**(**CURRENT\_DATE**) - **YEAR**(p.birthday) **AS** `Year`,

tp.type\_subscription **AS** `Type`,

tp.price\_subscription **AS** Price

**FROM** users **AS** u

**JOIN** profiles **AS** p **on** u.id = p.user\_id

**JOIN** paid\_subscriptions **AS** pd **on** pd.user\_id = u.id

**JOIN** type\_subscription **AS** tp **on** tp.id = pd.type\_subscription\_id

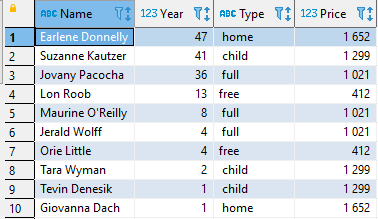
**ORDER** **BY** `Year` **DESC**;

После выполнения этого скрипта мы создаем представление в БД и имеем возможность обращаться к **name\_info** как к обычной таблице.

Например, запрос

**SELECT** \* **FROM** name\_info;

Даст такой результат:



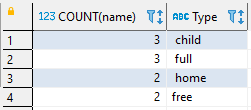
Или с помощью этого представления узнаем количество пользователей, имеющих одну подписку запросом:

**SELECT** **COUNT**(name), `type`

**FROM** name\_info

**GROUP** **BY** `type`;

Получая необходимые нам параметры



Также для примера можем объединить несколько таблиц в одну запросом:

**CREATE** **OR** **REPLACE** **VIEW** all\_values **AS**

**SELECT** u.\*,

p.\*,

pd.type\_subscription\_id,

tp.type\_subscription,

tp.price\_subscription,

mr.rate,

md.required\_subscription\_type,

cf.actors\_id

**FROM** users **AS** u

**JOIN** profiles **AS** p **on** u.id = p.user\_id

**JOIN** paid\_subscriptions **AS** pd **on** pd.user\_id = u.id

**JOIN** type\_subscription **AS** tp **on** tp.id = pd.type\_subscription\_id

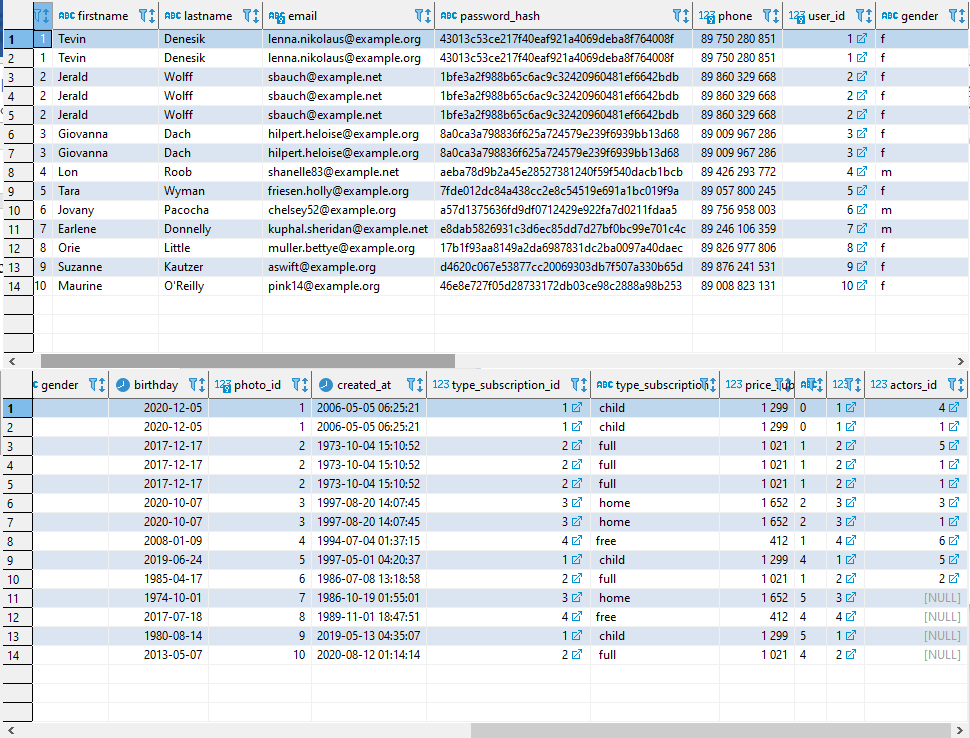
**JOIN** media\_ratings **AS** mr **on** mr.user\_id = u.id

**JOIN** media **AS** md **on** md.id = mr.media\_id

**LEFT** **JOIN** command\_film **AS** cf **on** cf.media\_id = md.id;

Где запросом : **SELECT** \* **FROM** all\_values;

Можем получить сразу всю информацию



**Процедуры**

Хранимые процедуры позволяют сохранить последовательность SQL-операторов и вызывать их по имени функции или процедуры

Процедура решает наиболее часто возникающие задачи, например изменим столбец only\_for\_adults (только для взрослых) где у нас стоит тип BIT (по умолчанию 0) на 1 в зависимости от жанра. Создадим процедуру:

**DROP** **PROCEDURE** **IF** **EXISTS** kinopoisk.change\_mode;

**DELIMITER //**

**CREATE** **PROCEDURE** kinopoisk.change\_mode()

**BEGIN**

**UPDATE** media **SET** only\_for\_adults = 1 **WHERE** media\_type\_id **IN** (2,4);

**END**//

**DELIMITER ;**

Теперь при вызове данной процедуры **CALL** change\_mode(); будет меняться значение в столбце only\_for\_adults с 0 на 1.

**Триггеры**

**Триггер** — специальная хранимая процедура, привязанная к событию изменения содержимого таблицы.

Чтобы избавиться от необходимости постоянного вызова процедуры, можно воспользоваться триггером, изменяющим значение only\_for\_adults автоматически после добавления строки в таблицу media. Воспользуемся скриптом создания триггера:

**DROP** **TRIGGER** **IF** **EXISTS** changes;

**DELIMITER //**

**CREATE** **TRIGGER** changes **BEFORE** **INSERT** **ON** media

**FOR** **EACH** **ROW**

**BEGIN**

**IF**(**NEW**.media\_type\_id **IN** (2, 4)) **THEN**

**SET** **NEW**.only\_for\_adults = 1;

**END** **IF**;

**END** //

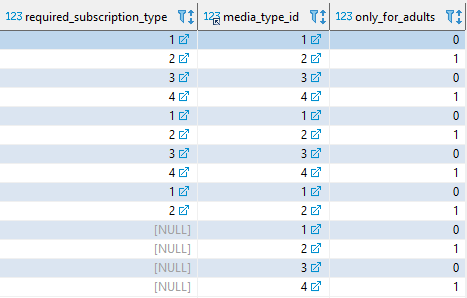
**DELIMITER ;**

После добавления триггера, значение строки в столбце only\_for\_adults, где media\_type\_id равен 2 или 4 будут автоматически изменяться на 1

Проверим триггер

**INSERT** **INTO** media (media\_type\_id) **VALUES** (1), (2), (3), (4);

Получаем:



**Используемые источники**

1. <https://dev.mysql.com/doc/refman/5.7/en/union.html>
2. <https://dev.mysql.com/doc/refman/5.7/en/subqueries.html>
3. <https://dev.mysql.com/doc/refman/5.7/en/join.html>
4. Линн Бейли. Head First. Изучаем SQL. — СПб.: Питер, 2012. — 592 с.
5. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппель, Эндрю Дж. SQL: полное руководство, 3-е изд. : Пер. с англ. — М.: ООО "И.Д. Вильямс", 2015. — 960 с.
6. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. — Пер. с англ. — СПб.: Символ-Плюс, 2010. — 480 с.
7. Кузнецов М.В., Симдянов И.В. MySQL на примерах. — СПб.: БХВ-Петербург, 2007. — 592с.
8. Кузнецов М.В., Симдянов И.В. MySQL 5. — СПб.: БХВ-Петербург, 2006. — 1024с.
9. Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.
10. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение. — Рид Групп, 2011. — 336 с.