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.

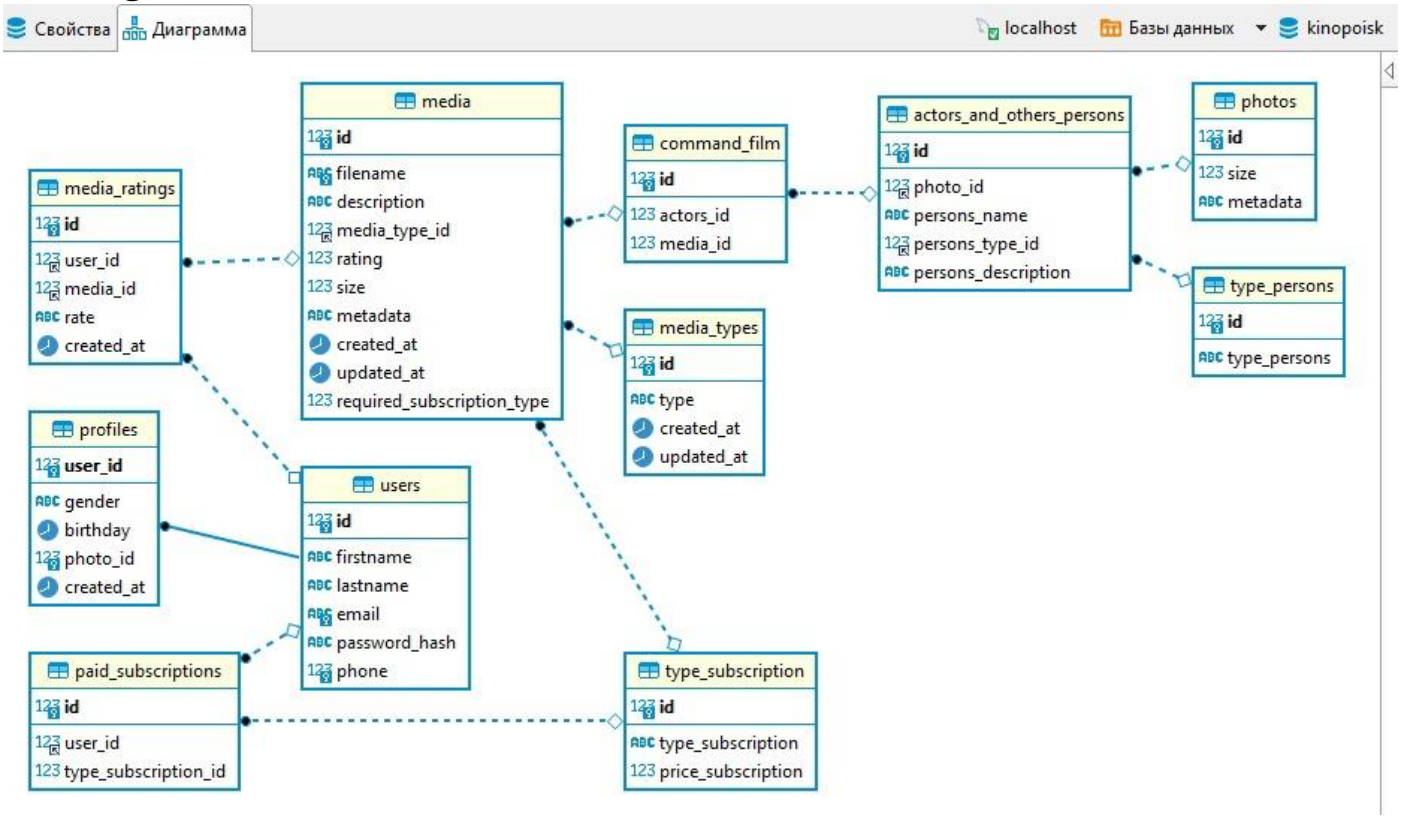
На основе популярно веб-сайта «Кинопоиск» была спроектирована реляционная база данных с использованием СУБД «MySQL».

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:

	Name	Year	Type	Price
1	Tevin Denesik	1	child	1 299
2	Jerald Wolff	4	full	1 021
3	Giovanna Dach	1	home	1 652
4	Lon Roob	13	free	412
5	Tara Wyman	2	child	1 299
6	Jovany Pacocha	36	full	1 021
7	Earlene Donnelly	47	home	1 652
8	Orie Little	4	free	412
9	Suzanne Kautzer	41	child	1 299
10	Maurine O'Reilly	8	full	1 021

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:

	Count_films	Films	type
1	3	et, ut, itaque	Action

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:

	Name	Year	Type	Price
1	Earlene Donnelly	47	home	1 652
2	Suzanne Kautzer	41	child	1 299
3	Jovany Pacocha	36	full	1 021
4	Lon Roob	13	free	412
5	Maurine O'Reilly	8	full	1 021
6	Jerald Wolff	4	full	1 021
7	Orie Little	4	free	412
8	Tara Wyman	2	child	1 299
9	Tevin Denesik	1	child	1 299
10	Giovanna Dach	1	home	1 652

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

	COUNT(name)	Type
1	3	child
2	3	full
3	2	home
4	2	free

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

	id	firstname	lastname	email	password_hash	phone	user_id	gender
1	1	Tevin	Denesik	lenna.nikolaus@example.org	43013c53ce217f40eaf921a4069deba8f764008f	89 750 280 851	1	f
2	1	Tevin	Denesik	lenna.nikolaus@example.org	43013c53ce217f40eaf921a4069deba8f764008f	89 750 280 851	1	f
3	2	Jerald	Wolff	sbauch@example.net	1bfe3a2f988b65c6ac9c32420960481ef6642bdb	89 860 329 668	2	f
4	2	Jerald	Wolff	sbauch@example.net	1bfe3a2f988b65c6ac9c32420960481ef6642bdb	89 860 329 668	2	f
5	2	Jerald	Wolff	sbauch@example.net	1bfe3a2f988b65c6ac9c32420960481ef6642bdb	89 860 329 668	2	f
6	3	Giovanna	Dach	hilpert.heloise@example.org	8a0ca3a798836f625a724579e239f6939bb13d68	89 009 967 286	3	f
7	3	Giovanna	Dach	hilpert.heloise@example.org	8a0ca3a798836f625a724579e239f6939bb13d68	89 009 967 286	3	f
8	4	Lon	Roob	shanelle83@example.net	aeba78d9b2a45e28527381240f59f540dacb1bcb	89 426 293 772	4	m
9	5	Tara	Wyman	friesen.holly@example.org	7fde012dc84a438cc2e8c54519e691a1bc019f9a	89 057 800 245	5	f
10	6	Jovany	Pacocha	chelsey52@example.org	a57d1375636fd9df0712429e922fa7d0211fdaa5	89 756 958 003	6	m
11	7	Earlene	Donnelly	kuphal.sheridan@example.net	e8dab5826931c3d6ec85dd7d27bf0bc99e701c4c	89 246 106 359	7	m
12	8	Orie	Little	muller.bettye@example.org	17b1f93aa8149a2da6987831dc2ba0097a40daec	89 826 977 806	8	f
13	9	Suzanne	Kautzer	aswift@example.org	d4620c067e53877cc20069303db7f507a330b65d	89 876 241 531	9	f
14	10	Maurine	O'Reilly	pink14@example.org	46e8e727f05d28733172db03ce98c2888a98b253	89 008 823 131	10	f

	gender	birthday	photo_id	created_at	type_subscription_id	type_subscription	price_sub		actors_id
1		2020-12-05	1	2006-05-05 06:25:21	1	child	1 299	0	1	4
2		2020-12-05	1	2006-05-05 06:25:21	1	child	1 299	0	1	1
3		2017-12-17	2	1973-10-04 15:10:52	2	full	1 021	1	2	5
4		2017-12-17	2	1973-10-04 15:10:52	2	full	1 021	1	2	1
5		2017-12-17	2	1973-10-04 15:10:52	2	full	1 021	1	2	1
6		2020-10-07	3	1997-08-20 14:07:45	3	home	1 652	2	3	3
7		2020-10-07	3	1997-08-20 14:07:45	3	home	1 652	2	3	1
8		2008-01-09	4	1994-07-04 01:37:15	4	free	412	1	4	6
9		2019-06-24	5	1997-05-01 04:20:37	1	child	1 299	4	1	5
10		1985-04-17	6	1986-07-08 13:18:58	2	full	1 021	1	2	2
11		1974-10-01	7	1986-10-19 01:55:01	3	home	1 652	5	3	[NULL]
12		2017-07-18	8	1989-11-01 18:47:51	4	free	412	4	4	[NULL]
13		1980-08-14	9	2019-05-13 04:35:07	1	child	1 299	5	1	[NULL]
14		2013-05-07	10	2020-08-12 01:14:14	2	full	1 021	4	2	[NULL]

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:

123 required_subscription_type	123 media_type_id	123 only_for_adults
1	1	0
2	2	1
3	3	0
4	4	1
1	1	0
2	2	1
3	3	0
4	4	1
1	1	0
2	2	1
[NULL]	1	0
[NULL]	2	1
[NULL]	3	0
[NULL]	4	1

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.