

Курсовая работа по курсу

Основы реляционных баз данных. MySQL

Выполнил Мальчевский Владимир

26 сентября 2022г.

Оглавление

1	Требования к курсовой работе.....	2
2	Текстовое описание БД и решаемых ею задач	2
2.1	Сценарий использования on-line кинотеатра Cinema.....	2
3	Требования к объему базы данных	3
4	Скрипты создания структуры БД.....	4
5	ER Diagram для БД.....	5
6	Скрипты наполнения БД данными.....	6
7	Скрипты характерных выборок	8
7.1	Выборка 7.1	8
7.2	Выборка 7.2	9
7.3	Выборка 7.3	9
7.4	Выборка 7.4	10
7.5	Выборка 7.5	11
7.6	Выборка 7.6	12
8	Представления	13
8.1	Представление 1.....	13
8.2	Представление 2.....	14
9	Хранимые процедуры / триггеры;	15
9.1	Процедура 1	15
9.2	Процедура 2	16
9.3	Триггер 1	18
9.4	Триггер 2	19
10	Список использованных материалов.....	20
11	Список скриптов базы данных	21

1 Требования к курсовой работе

Требования к практической работе:

1. Составить общее текстовое описание БД и решаемых ею задач;
2. минимальное количество таблиц - 10;
3. скрипты создания структуры БД (с первичными ключами, индексами, внешними ключами);
4. создать ERDiagram для БД;
5. скрипты наполнения БД данными;
6. скрипты характерных выборки (включающие группировки, JOIN'ы, вложенные таблицы);
7. представления (минимум 2);
8. хранимые процедуры / триггеры;

2 Текстовое описание БД и решаемых ею задач

База данных cinema предназначена для предоставления услуг on-line кинотеатра.

База данных содержит информацию о фильмах, актерах, пользователях, их кредитных картах и аккаунтах.

также хранится профиль пользователя, предоставляющий возможность получить статистику просмотра фильмов по различным критериям.

База данных включает таблицы взаимодействия между вышеперечисленными таблицами.

2.1 Сценарий использования on-line кинотеатра Cinema

Предполагается следующий сценарий использования on-line кинотеатра Cinema:

Пользователь услуги при входе на своем телевизоре в программу – кинотеатр увидит список новых фильмов, разделенный по нескольким жанрам.

Пользователь может запросить полный список фильмов в данном жанре, список новинок в данном жанре, список актеров данного фильма, список фильмов с участием какого-либо актера.

Администрация кинотеатра может проводить акции, начисляя дополнительные баллы пользователям, выбранным случайным образом. Также для

администрации доступно представление пользователей с наименьшим балансом на счете.

В базу данных On-line кинотеатра “cinema” входят следующие таблицы (с указанием полей):

- `actors` (`actor_id` , `actor_name` , `actor_surname`)
- `file_type` (`id` , `file_type`)
- `film_actors` (`id` , `actor_id` , `film_id`)
- `film_genre` (`genre_id` , `genre_name`)
- `film_user` (`id` , `user_id` , `film_id` , `date_of_view` , `film_user_rating`)
- `films` (`film_id` , `film_title` , `film_file_name` , `film_title_pict_filename` , `film_file_size` , `film_file_type` , `film_genre` , `film_duration` , `film_prime_date` , `film_rating`)
- `logs_users` (`id` , `record_created_at` , `tab_name` , `user_id`)
- `u_statuses` (`status_id` , `status_name`)
- `user_credit_card` (`user_id` , `cc_number` , `cc_vaild_date` , `cc_confirmed`)
- `user_payments` (`trans_id` , `user_id` , `pay_amount` , `pay_date` , `pay_balance` , 'Account Balance')
- `user_profile` (`user_id` , `u_login` , `u_pass_word` , `u_gender` , `u_email_address` , `u_phone_number` , `u_city` , `u_address` , `updated_at`)
- `users` (`u_id` , `u_name` , `u_surname` , `u_status_id` , `created_at`)

3 Требования к объему базы данных

По требованию к курсовой работе минимальное количество таблиц - 10;

База данных включает 11 таблиц, соответственно соответствует требованию;

4 Скрипты создания структуры БД

Скрипты создания структуры БД приложены в файле `cinema7_tables.sql`

Скрипт создания внешних ключей приложен в файле `cinema_frqn_key.sql`

5 ER Diagram для БД

ER диаграмма показана на *Рис. 1*.

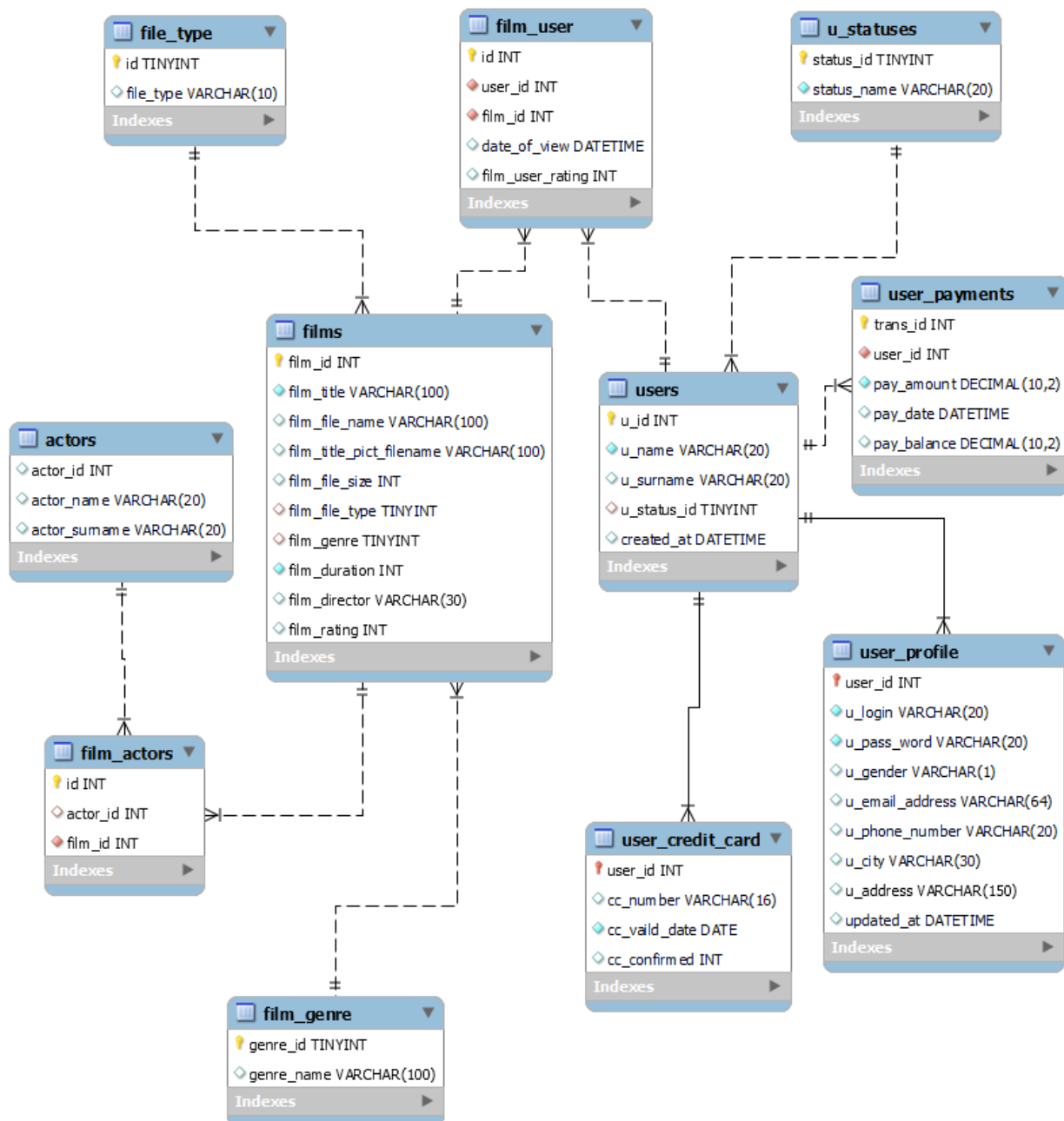


Рис. 1 ER диаграмма БД "cinema"

6 Скрипты наполнения БД данными

Следующие скрипты наполнения БД данными были заполнены вручную:

actors, file_type, film_genre, u_statuses;

Остальные скрипты наполнения БД данными были заполнены с помощью web-приложения <http://filldb.info/> и дополнительно отредактированы.

Скрипты которые применялись для редактирования:

-- Проверка условия что обновление профиля раньше создания аккаунта

```
USE cinema;
Select updated_at AS updated, u_id AS USER, created_at AS
created from user_profile
join users
ON users.created_at >= user_profile.updated_at and users.u_id =
user_profile.user_id;
```

-- исправление записей, где обновление профиля раньше создания аккаунта

```
update user_profile SET updated_at =
(select created_at FROM users where u_id = user_id)
where
user_profile.updated_at < (select created_at FROM users where
u_id = user_id);
```

-- Проверка условия что оплата раньше создания аккаунта

```
Select pay_date , u_id AS USER, created_at AS created from
user_payments
join users
ON users.created_at >= user_payments.pay_date and users.u_id =
user_payments.user_id;
```

-- исправление записей, где оплата раньше создания аккаунта

```
update user_payments SET pay_date =
(select created_at FROM users where u_id = user_id)
where
user_payments.pay_date < (select created_at FROM users where
u_id = user_id);
```

-- Проверка условия что просмотр фильма не ранее даты регистрации

```
Select date_of_view , user_id AS USER, created_at AS created
from film_user
join users
ON users.created_at >= film_user.date_of_view and users.u_id =
film_user.user_id;
```

-- исправление записей где просмотр фильма не ранее даты регистрации

```
update film_user SET date_of_view =  
(select created_at FROM users where u_id = user_id)  
where  
film_user.date_of_view < (select created_at FROM users where  
u_id = user_id);
```

-- Изменение колонки film_director на film_prime_date, наполнение данными

```
ALTER TABLE films RENAME COLUMN film_director TO  
film_prime_date;  
ALTER TABLE films MODIFY COLUMN film_prime_date date;  
UPDATE films SET film_prime_date = CURRENT_DATE - INTERVAL (  
3650 + FLOOR(RAND() * 15000)) DAY;  
SELECT * FROM films;
```

Скрипты наполнения БД данными приложены в файле cinema7_data.sql

7 Скрипты характерных выборок

Скрипты характерных выборок, включающие группировки, JOIN , вложенные таблицы представлены ниже:

7.1 Выборка 7.1

Список новых фильмов, разделенный по нескольким категориям.

```
(SELECT film_title AS NAME,
film_rating AS RATING,
film_genre.genre_name as GENRE ,
film_prime_date as Premier FROM films
join film_genre
on film_genre.genre_id=films.film_genre and
film_genre = 1
order by film_prime_date DESC
limit 4)
union
(SELECT film_title AS NAME,
film_rating AS RATING,
film_genre.genre_name as GENRE ,
film_prime_date as Premier FROM films
join film_genre
on film_genre.genre_id=films.film_genre and
film_genre = 2
order by film_prime_date DESC
limit 4)
union
(SELECT film_title AS NAME,
film_rating AS RATING, film_genre.genre_name as GENRE ,
film_prime_date as Premier FROM films
join film_genre
on film_genre.genre_id=films.film_genre and
film_genre = 3
order by film_prime_date DESC
limit 4);
```

	NAME	RATING	GENRE	Premier
►	Get Out	4	Comedy	1998-06-23
	A Space Odyssey	8	Comedy	1987-02-06
	Léon	4	Comedy	1975-11-01
	RPortrait Of A Lady On Fire	10	Musical	2012-05-06
	His Girl Friday	7	Musical	1992-09-08
	Lady Bird	8	Musical	1975-01-05
	The Dark Knight	7	Western	2008-08-29
	Groundhog Day	3	Western	1992-10-02
	La Dolce Vita	7	Western	1987-02-15
	One Flew Over The Cuckoo_s Nest	5	Western	1984-07-02

7.2 Выборка 7.2

Полный список фильмов в данном жанре, отсортированный по новизне

```
SELECT
film_title AS NAME,
film_rating AS RATING,
film_genre.genre_name as GENRE ,
film_prime_date as Premier FROM films
join film_genre
on film_genre.genre_id=films.film_genre and film_genre = 3
order by film_prime_date DESC
limit 20;
```

	NAME	RATING	GENRE	Premier
►	The Dark Knight	7	Western	2008-08-29
	Groundhog Day	3	Western	1992-10-02
	La Dolce Vita	7	Western	1987-02-15
	One Flew Over The Cuckoo_s Nest	5	Western	1984-07-02
	Moonlight	4	Western	1972-04-19

Рис. 3

7.3 Выборка 7.3

Список актеров данного фильма,

```
SELECT films.film_title AS TITLE,
CONCAT( actors.actor_name,' ', actors.actor_surname) AS ACTOR
FROM film_actors
JOIN films
JOIN actors
on films.film_id=film_actors.film_id
and actors.actor_id=film_actors.actor_id
and film_title LIKE "Bicycle%"
order by ACTOR DESC
limit 20;
```

	TITLE	ACTOR
►	Bicycle Thieves	Morgan Freeman
	Bicycle Thieves	Martin Freeman
	Bicycle Thieves	Liam Neeson
	Bicycle Thieves	Christian Bale

Рис. 4

7.4 Выборка 7.4

Список фильмов с участием какого-либо актера.

```
Select films.film_title as TITLE, CONCAT(actors.actor_name, ' ',  
actors.actor_surname) AS ACTOR, film_genre.genre_name as GENRE  
from films  
join actors  
join film_actors  
join film_genre  
on actors.actor_id=film_actors.actor_id  
and film_actors.film_id = films.film_id  
and films.film_genre = film_genre.genre_id  
and actors.actor_name = "Angelina";
```

	TITLE	ACTOR	GENRE
►	Get Out	Angelina Jolie	Comedy
	The Dark Knight	Angelina Jolie	Western
	Star Wars: Return Of The Jedi	Angelina Jolie	Fiction
	Guardians Of The Galaxy	Angelina Jolie	Adventures
	Citizen Kane	Angelina Jolie	Historical
	Arrival	Angelina Jolie	Thriller
	Point Break	Angelina Jolie	Adventures
	Psycho	Angelina Jolie	Fiction
	Paddington 2	Angelina Jolie	Historical
	His Girl Friday	Angelina Jolie	Musical
	La La Land	Angelina Jolie	Melodrama
	Mulholland Drive	Angelina Jolie	Melodrama
	Arrival	Angelina Jolie	Thriller
	Black Panther	Angelina Jolie	Fantasy
	Spider-Man: Into The Spider-...	Angelina Jolie	Thriller
	Forrest Gump	Angelina Jolie	Thriller
	Singin' in the Rain	Angelina Jolie	Criminal

Рис. 5

7.5 Выборка 7.5

Список фильмов отсортированный по рейтингу.

```
Select
  films.film_title as TITLE,
  film_genre.genre_name AS GENRE ,
  films.film_rating as RATING from films
join film_genre
  order by films.film_rating DESC
  limit 20 ;
```

	TITLE	GENRE	RATING
	Paddington 2	Historical	10
	Titanic	Criminal	10
	RPortrait Of A Lady On Fire	Musical	10
	Ghostbusters	Documental	10
	The Social Network	Melodrama	10
	Inglourious Basterds	Adventures	9
	A Space Odyssey	Comedy	8
	The Usual Suspects	Fantasy	8
	Raiders of the Lost Ark	Thriller	8
	Lady Bird	Musical	8
	Seven Samurai	Fantasy	8
	Lawrence Of Arabia	Adventures	8
	12 Angry Men	Adventures	8
	Black Panther	Fantasy	8
	The Exorcist	Melodrama	8
	Bicycle Thieves	Melodrama	8
	La Dolce Vita	Western	7

Рис. 6

7.6 Выборка 7.6

Список фильмов которые посмотрело больше всего пользователей (с использованием JOIN, ON)

```
SELECT films.film_title AS FILM,  
       COUNT(*) AS VIEWS  
FROM film_user  
JOIN films  
    ON films.film_id = film_user.film_id  
group by film_user.film_id  
order by VIEWS DESC  
LIMIT 10;
```

	FILM	VIEWS
►	The Godfather	5
	Moonlight	5
	Singin' in the Rain	5
	Ghostbusters	4
	Seven Samurai	4
	Black Panther	4
	The Usual Suspects	4
	His Girl Friday	4
	Star Wars: Return Of The Jedi	4
	Get Out	4

Рис. 7

-- (с использованием WHERE)

```
SELECT films.film_title AS FILM,  
       COUNT(*) AS VIEWS  
FROM film_user, films  
WHERE films.film_id = film_user.film_id  
group by film_user.film_id  
order by VIEWS DESC  
LIMIT 10;
```

	FILM	VIEWS
►	The Godfather	5
	Moonlight	5
	Singin' in the Rain	5
	Ghostbusters	4
	Seven Samurai	4
	Black Panther	4
	The Usual Suspects	4
	His Girl Friday	4
	Star Wars: Return Of The Jedi	4
	Get Out	4

Рис. 8

8 Представления

8.1 Представление 1

Список новых фильмов, разделенный по нескольким категориям.

```
DROP VIEW IF EXISTS first_screen;

CREATE OR REPLACE VIEW first_screen AS
(SELECT film_title AS NAME,
  film_rating AS RATING,
  film_genre.genre_name as GENRE ,
  film_prime_date as Premier FROM films
  join film_genre
    on film_genre.genre_id=films.film_genre and
    film_genre = 1
    order by film_prime_date DESC
  limit 4)
union
(SELECT  film_title AS NAME,
  film_rating AS RATING,
  film_genre.genre_name as GENRE ,
  film_prime_date as Premier FROM films
  join film_genre
    on film_genre.genre_id=films.film_genre and
    film_genre = 2
    order by film_prime_date DESC
  limit 4)
union
(SELECT film_title AS NAME,
  film_rating AS RATING, film_genre.genre_name as GENRE ,
  film_prime_date as Premier FROM films
  join film_genre
    on film_genre.genre_id=films.film_genre and
  film_genre = 3
    order by film_prime_date DESC
  limit 4);
SELECT * FROM first_screen;
```

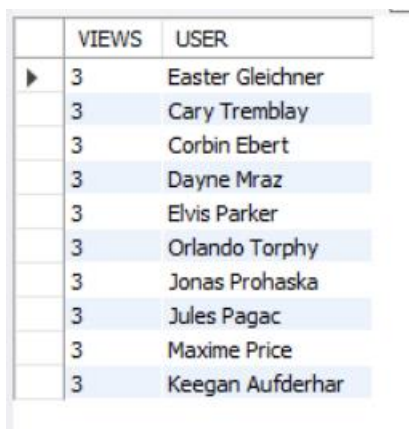
	NAME	RATING	GENRE	Premier
►	Get Out	4	Comedy	1998-06-23
	A Space Odyssey	8	Comedy	1987-02-06
	Léon	4	Comedy	1975-11-01
	RPortrait Of A Lady On Fire	10	Musical	2012-05-06
	His Girl Friday	7	Musical	1992-09-08
	Lady Bird	8	Musical	1975-01-05
	The Dark Knight	7	Western	2008-08-29
	Groundhog Day	3	Western	1992-10-02
	La Dolce Vita	7	Western	1987-02-15
	One Flew Over The Cuckoo_s Nest	5	Western	1984-07-02

8.2 Представление 2

Список наиболее активных пользователей

```
CREATE OR REPLACE VIEW active_users AS
SELECT COUNT(film_user.user_id) AS VIEWS ,
       CONCAT (users.u_name, ' ', u_surname) AS USER
from film_user
      join users
ON users.u_id = film_user.user_id
   GROUP BY user_id
   ORDER BY VIEWS DESC
   LIMIT 10;

SELECT * FROM active_users;
```



	VIEWS	USER
▶	3	Easter Gleichner
	3	Cary Tremblay
	3	Corbin Ebert
	3	Dayne Mraz
	3	Elvis Parker
	3	Orlando Torphy
	3	Jonas Prohaska
	3	Jules Pagac
	3	Maxime Price
	3	Keegan Aufderhar

Рис. 10

9 Хранимые процедуры / триггеры;

9.1 Процедура 1

Процедура, создающая временную таблицу и вносящая пользователей с балансом менее 100

```
drop procedure if exists users_low_balance ;
DELIMITER //
create procedure users_low_balance()
BEGIN
    drop table if exists users_low_balance ;
    create temporary table users_low_balance (
        u_name varchar(20),
        u_surname varchar(20),
        u_id INT,
        balance DECIMAL(10.2)
    );

    INSERT INTO users_low_balance ( u_name , u_surname, u_id,
balance)
    (SELECT users.u_name , users.u_surname , users.u_id, pay_balance
AS BALANCE
    FROM user_payments
    join users
    ON (user_payments.user_id = users.u_id AND pay_balance < 100 )
    ORDER BY BALANCE) ;

end //

CALL users_low_balance();
SELECT * FROM users_low_balance ;
```

	u_name	u_surname	u_id	balance
►	Frank	Pollich	84	6
	Norval	Spencer	58	14
	Rafael	Lowe	10	18
	Luigi	Kovacek	80	23
	Easton	Bergnaum	88	30
	Maxime	Price	66	31
	Fermin	Bahringer	94	35
	Augustus	Leffler	65	41
	Florian	Jast	56	49
	Emmanuel	Bergstrom	100	59
	Mackenzie	O'Kon	81	61
	Johnny	Schulist	3	62
	Tremaine	Abshire	76	64
	Dee	Miller	32	69

Рис. 11

9.2 Процедура 2

-- Процедура выбора пользователей - победителей акции и начисления им бонусных баллов

```
drop procedure if exists users_bonus ;
DELIMITER //
create procedure users_bonus()

BEGIN
    drop table if exists users_bonus;
```

-- Создание таблицы для записи победителей

```
create temporary table users_bonus (
    trans_id INT,
    u_name varchar(20),
    u_surname varchar(20),
    u_id INT,
    old_balance DECIMAL(10.2),
    bonus DECIMAL(10.2),
    new_balance DECIMAL(10.2)
);
```

Выбор 10 победителей случайным образом, текущий баланс берется из таблицы pay_balance,(по max (transaction_id) , бонус составляет 10% от баланса; запись в таблицу users_bonus;

```
INSERT INTO users_bonus ( trans_id, u_name , u_surname, u_id,
old_balance, bonus , new_balance)
(SELECT max(trans_id) , users.u_name , users.u_surname ,
users.u_id, pay_balance AS OLD_BAL , pay_balance * 0.1 AS BONUS,
pay_balance * 1.1 AS NEW_BAL
FROM user_payments
join users
ON (user_payments.user_id = users.u_id )
GROUP BY users.u_id
ORDER BY rand()
LIMIT 10 ) ;
```

Вставка новых записей в таблицу user_payments с обновленным балансом

```
-- Update table user_payments with new balance
insert user_payments ( trans_id, user_id, pay_amount, pay_date,
pay_balance)
SELECT null , u_id, bonus, date(now()), new_balance
FROM users_bonus
join user_payments
where u_id= user_id ;
```



```
end //
```

Запуск процедуры

```
CALL users_bonus();
```

Результат работы процедуры

```
SELECT * FROM users_bonus;
```

	trans_id	u_name	u_surname	u_id	old_balance	bonus	new_balance
▶	68	Manuel	Williamson	24	192	19	211
	36	Gonzalo	Strosin	36	248	25	273
	32	Maxime	Price	66	198	20	218
	99	Angel	Stark	62	222	22	244
	83	Keaton	Kulas	63	425	43	468
	6	Chris	Morar	46	413	41	454
	48	Chris	Crooks	42	257	26	283
	11	Travis	Dach	1	421	42	463
	60	Josue	Connelly	51	370	37	407
	87	Elvis	Parker	39	333	33	366

Рис. 12

```
SELECT * FROM user_payments  
order BY trans_id DESC  
LIMIT 15;
```

	trans_id	user_id	pay_amount	pay_date	pay_balance
▶	113	39	33.00	2022-09-26 00:00:00	366.00
	112	51	37.00	2022-09-26 00:00:00	407.00
	111	51	37.00	2022-09-26 00:00:00	407.00
	110	1	42.00	2022-09-26 00:00:00	463.00
	109	42	26.00	2022-09-26 00:00:00	283.00
	108	46	41.00	2022-09-26 00:00:00	454.00
	107	63	43.00	2022-09-26 00:00:00	468.00
	106	62	22.00	2022-09-26 00:00:00	244.00
	105	66	20.00	2022-09-26 00:00:00	218.00
	104	66	20.00	2022-09-26 00:00:00	218.00
	103	36	25.00	2022-09-26 00:00:00	273.00
	102	24	19.00	2022-09-26 00:00:00	211.00
	101	24	19.00	2022-09-26 00:00:00	211.00
	100	47	249.00	2018-12-23 01:43:47	120.00
	99	62	300.00	2009-07-06 03:58:46	222.00
*	NULL	NULL	NULL	NULL	NULL

Рис. 13

Появились новые записи пополнения баланса пользователя с новым балансом с учетом бонуса

9.3 Триггер 1

Автоматическое создание записи в таблице user_profile при создании записи в таблице users

```
USE cinema;
SHOW TRIGGERS;

DROP TRIGGER IF EXISTS insert_user_to_user_profile_after_insert;

DELIMITER //
CREATE TRIGGER insert_user_to_user_profile_after_insert AFTER
INSERT ON users
FOR EACH ROW
BEGIN
INSERT INTO user_profile(user_id, u_login, u_pass_word, u_gender,
u_email_address, u_phone_number, u_city, u_address )
VALUES
(new.u_id, 'login', 'password', NULL ,NULL
, NULL , NULL , NULL ) ;
END//
DELIMITER ;

SELECT * FROM users;
```

	u_id	u_name	u_surname	u_status_id	created_at
▶	109	Vladimir	Mal	2	2022-09-26 11:08:10
*	NULL	NULL	NULL	NULL	NULL

Рис. 14

```
SELECT * FROM user_profile where user_id > 100;
```

	user_id	u_login	u_pass_word	u_gender	u_email_address	u_phone_number	u_city	u_address	updated_at
▶	109	login	password	NULL	NULL	NULL	NULL	NULL	2022-09-26 1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 15

9.4 Триггер 2

Создание таблицы logs_users типа Archive.

При каждом создании записи в таблицах 'users' или 'user_profile' в таблицу 'logs_users' помещается время и дата создания записи, название таблицы и содержимое поля 'user_id'.

```
DROP TABLE IF EXISTS logs_users;
```

```
CREATE TABLE logs_users (  
  id SERIAL,  
  record_created_at DATETIME DEFAULT CURRENT_TIMESTAMP,  
  tab_name varchar(255) NOT NULL,  
  user_id INT UNSIGNED NOT NULL  
) COMMENT = 'Log Record' ENGINE=Archive;
```

```
SHOW TRIGGERS;
```

```
SELECT * FROM logs_users;
```

```
DROP TRIGGER IF EXISTS insert_log_users_after_insert;  
DELIMITER //  
CREATE TRIGGER insert_logs_users_users_after_insert AFTER INSERT  
ON users  
FOR EACH ROW  
BEGIN  
  INSERT INTO logs_users ( id, record_created_at, tab_name, user_id  
  )  
  VALUES (null, NOW(), "users", new.u_id) ;  
END//  
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS  
insert_logs_users_user_profile_after_insert;  
DELIMITER //  
CREATE TRIGGER insert_log_user_profile_after_insert AFTER INSERT  
ON user_profile  
FOR EACH ROW  
BEGIN  
  INSERT INTO logs_users ( id, record_created_at, tab_name, user_id  
  )  
  VALUES (null, NOW(), 'user_profile', new.user_id) ;  
END//  
DELIMITER ;
```

Тест триггера insert_log_users_after_insert

```
TRUNCATE logs_users;

INSERT INTO users (u_name, u_surname, u_status_id) VALUES
('Vladimir', 'Mal', 2);
SELECT * FROM users WHERE u_name='Vladimir';

SELECT * FROM logs_users;
-- DELETE FROM users WHERE u_id = 108 ;
SELECT * FROM user_profile where user_id > 100;
-- DELETE FROM user_profile WHERE user_id = 108 ;
SHOW TRIGGERS;

SELECT * FROM users;
```

	u_id	u_name	u_surname	u_status_id	created_at
▶	109	Vladimir	Mal	2	2022-09-26 11:08:10
*	NULL	NULL	NULL	NULL	NULL

Рис. 16

Запись в таблице user_profile автоматически создалась

```
SELECT * FROM user_profile where user_id > 100;
```

	user_id	u_login	u_pass_word	u_gender	u_email_address	u_phone_number	u_city	u_address	updated_at
▶	109	login	password	NULL	NULL	NULL	NULL	NULL	2022-09-26 11:08:10
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 17

2 записи в таблице logs_users автоматически создались

```
SELECT * FROM logs_users;
```

	id	record_created_at	tab_name	user_id
	6	2022-09-26 11:05:56	users	108
	7	2022-09-26 11:05:56	user_profile	108
	8	2022-09-26 11:08:10	users	109
	9	2022-09-26 11:08:10	user_profile	109
*	NULL	NULL	NULL	NULL

Рис. 18

10 Список использованных материалов

[1] Лекции курсов GeekBrains

11 Список скриптов базы данных

К настоящей работе приложены следующие скрипты:

	Название скрипта	Содержание
1.	cinema7_data.sql	Скрипты Данные таблиц
2.	cinema7_tables.sql	Скрипты Структура таблиц
3.	cinema_frqn_key.sql	Скрипты Назначение FOREIGN_KEY
4.	cinema_triggers_procedures.sql	Скрипты Триггеры и процедуры
5.	sinema_select_view.sql	Скрипты Выборки SELECT и создание VIEW