Каскадные операции

название дочерней таблицы	название родительской таблицы	удаление	обновление
content_series_url	content	каскадное	каскадное
content_series_url	resolutions	каскадное	каскадное
content	copyrigth_holders	каскадное	каскадное
content	content_types	каскадное	каскадное
content_m2m_content_genres	content_genres	запрет	каскадное
content_m2m_content_genres	content	каскадное	каскадное
content_voice_acting	content	каскадное	каскадное
content_voice_acting	languages	запрет	каскадное
content_subtitles	content	каскадное	каскадное
content_subtitles	languages	запрет	каскадное
copyrigth_holders	content_pocent_types	запрет	каскадное
histories	content	не нужно	каскадное
histories	user	не нужно	каскадное
content_movies_urls	content	каскадное	каскадное
content_movies_urls	resolutions	каскадное	каскадное
profiles	profile_types	Запрет	каскадное
profiles	user	каскадное	каскадное
profile_types_m2m_contents	profile_types	каскадное	каскадное
profile_types_m2m_contents	content	каскадное	каскадное
profile_types_m2m_contents _user_subscription_types	profile_types_m2m_contents	каскадное	каскадное
profile_types_m2m_contents _user_subscription_types	user_subscription_types	каскадное	каскадное
ratings	content	каскадное	каскадное
user_cards	user_card_payment_systems	запрет	каскадное
user_m2m_ user_cards	user_cards	не нужно	каскадное
user_m2m_ user_cards	user	каскадное	каскадное
user_comments	user	не нужно	каскадное
user_comments	content	каскадное	каскадное
user_subscription_types	user_subscription_prices	запрет	каскадное
user	user_subscription_types	Значение NULL	каскадное
user_subscription_types _m2m_resolutions	user_subscription_types	каскадное	каскадное
user_subscription_types _m2m_resolutions	resolutions	каскадное	каскадное

- Для сущностей, являющихся частью контента, определены каскадные операции удаления и обновления.

- Для сущностей, со связью многие ко многим каскадные операции проставлены в соответствии с требованиями заказчика.

Представления

```sql
create view kcinema.getTopContent as
SELECT content.c_name             as name,
       ratings.r_value            as raiting,
       content.c_release_date     as release_date,
       content.c_text_description as description,
       content.c_duration         as duration
FROM content
         JOIN ratings on content.ct_id = ratings.ct_id and content.ct_id = ratings.ct_id
WHERE r_value <= 5 OR r_value >= 4
```

Получение контента с лучшим рейтингом для главной страницы приложения

```sql
SELECT ust.usp_id as id,
       usp.usp_price as price,
       ust.ust_descriprion as description,
       usp.usp_description as valute_code,
       r.r_resolution as resolution,
       r.r_fps as fps
       FROM user_subscription_types ust
LEFT JOIN user_subscription_prices usp on ust.usp_id = usp.usp_id
LEFT JOIN user_subscription_types_m2m_resolutions ustm2mr on ust.ust_id = ustm2mr.ust_id
LEFT JOIN resolutions r on r.r_id = ustm2mr.r_id
```

Получить информации о типах подписок приложения в читаемом виде.

```sql
create view kcinema.getActiveCards as
SELECT uc.uc_number as card_number,
       uc.uc_cvv as cvv,
       ucps.ucps_description as card_type
 FROM user_cards uc
 LEFT JOIN user_card_payment_systems ucps on uc.ucps_id = ucps.ucps_id
LEFT JOIN users_m2m_user_cards um2muc on uc.uc_number = um2muc.uc_number WHERE um2muc.umuc_card_is_active = TRUE;
```

Получить список активных карт сервиса, для работы с операциями оплаты сервиса

```sql
create view kcinema.getHolderContent as
SELECT ch.ch_id as holder_id,
       ch.ch_name as name,
       c.ch_id as content_id,
       c.c_name as content_name,
       ct.ct_description as content_type,
       cpt.cpt_precent as precent,
       cpt.cpt_description as valute_code
FROM copyrigth_holders ch
LEFT JOIN content c on ch.ch_id = c.ch_id
LEFT JOIN content_types ct on c.ct_id = ct.ct_id
LEFT JOIN content_procent_types cpt on ch.cpt_id = cpt.cpt_id;
```

Получить информации о том, кто владелец какого контента в читаемом виде

```sql
create view kcinema.getFullContentInfo as
SELECT content.c_name             as name,
       ratings.r_value            as raiting,
       content.c_release_date     as release_date,
       content.c_text_description as description,
       content.c_duration         as duration,
       ch.ch_name                 as copyrigth_holder,
       pt.pt_description          as profile,
       ust.ust_descriprion        as sub_type
FROM content
         JOIN ratings on content.ct_id = ratings.ct_id and content.ct_id = ratings.ct_id
         JOIN profile_types_m2m_contents ptm2mc on content.c_id = ptm2mc.c_id
         JOIN profile_types pt on ptm2mc.pt_id = pt.pt_id
         JOIN profile_types_m2m_contents_user_subscription_types ptm2mcust on content.c_id = ptm2mcust.c_id
         JOIN user_subscription_types ust on ptm2mcust.ust_id = ust.ust_id
         JOIN copyrigth_holders ch on content.ch_id = ch.ch_id
```

Получить полную информацию о контенте

Проверки

1. Для всех полей были логически установлены проверки NULL/NOT NULL.
2. Для полей, определяющих количество объектов реального мира были установлены проверки > 0. Это такие поля как:
    a. длительность контента
    b. размер выплат
    c. количество эпизодов
    d. точка остановки просмотра
    e. количество людей, поставивших рейтинг
    f. необходимое пороговое кол-во людей, поставивших рейтинг.

3. Для пароля для аккаунта была поставлена проверка: длина > 6 символов.
4. Для сущности типы контентом было установлено пороговое количество возможных жанров 10.
5. Для количества людей, поставивших рейтинг N была установлена проверка по шаблону:
    a. CHECK (r_N_amount >= 0 AND r_N_amount <= ratings.r_common_amount).
6. Значение рейтинга должно быть >= 0 и <= 5
7. В сущности, разрешения контента поле количество кадров в секунду должно быть >12.
8. В сущности аккаунт, последняя дата оплаты должны быть не позже 30 дней от сегодняшней даты.
9. Для полей были логически установлены значений по умолчанию.

Код можно посмотреть в Приложении 1.

Триггеры

Банковская карточка может быть привязана к нескольким аккаунтам. Триггер позволяет автоматически удалить данные о карте из таблицы `users_cards` если она не привязана ни к 1 аккаунту.

```
CREATE TRIGGER incCardUseAmount
    AFTER INSERT
    on kcinema.users_m2m_user_cards
    FOR EACH ROW
BEGIN
    UPDATE user_cards uc SET uc_user_amount = uc_user_amount + 1 WHERE kcinema.users_m2m_user_cards.uc_number = uc.uc_number;
END;

CREATE TRIGGER decCardUseAmount
    AFTER DELETE
    on kcinema.users_m2m_user_cards
    FOR EACH ROW
BEGIN
    UPDATE user_cards uc SET uc_user_amount = uc_user_amount - 1 WHERE kcinema.users_m2m_user_cards.uc_number = uc.uc_number;
    IF kcinema.user_cards.uc_user_amount = 0 THEN
        DELETE FROM user_cards WHERE user_cards.uc_number = kcinema.users_m2m_user_cards.uc_number;
    END IF;
END;
```

Автоматический расчет рейтинга при вставке новой оценки. Если количество оценок не достигло порогового значения, рейтинг должен обнулится и не быть активным, т.е. доступным для отображения

```
create trigger calculateRatingValue
    BEFORE UPDATE
    on kcinema.ratings
    FOR EACH ROW
BEGIN
    IF NEW.r_1_amount != OLD.r_1_amount THEN
        SET NEW.r_common_amount = OLD.r_common_amount + (NEW.r_1_amount - OLD.r_1_amount);
    END IF;

    IF NEW.r_2_amount != OLD.r_2_amount THEN
        SET NEW.r_common_amount = OLD.r_common_amount + (NEW.r_2_amount - OLD.r_2_amount);
    END IF;
    IF NEW.r_3_amount != OLD.r_3_amount THEN
        SET NEW.r_common_amount = OLD.r_common_amount + (NEW.r_3_amount - OLD.r_3_amount);
    END IF;
    IF NEW.r_4_amount != OLD.r_4_amount THEN
        SET NEW.r_common_amount = OLD.r_common_amount + (NEW.r_4_amount - OLD.r_4_amount);
    END IF;
    IF NEW.r_5_amount != OLD.r_5_amount THEN
        SET NEW.r_common_amount = OLD.r_common_amount + (NEW.r_5_amount - OLD.r_5_amount);
    END IF;

    IF NEW.r_common_amount >= OLD.r_border_amount THEN
        SET NEW.r_is_active = true;
        SET NEW.r_value = (NEW.r_1_amount + 2 * NEW.r_2_amount + 3 * NEW.r_3_amount + 4 * NEW.r_4_amount +
                           5 * NEW.r_5_amount) / NEW.r_common_amount;
    ELSE
        SET NEW.r_is_active = false;
        SET NEW.r_value = 0;
    END IF;

END
```

Процедуры

```
CREATE PROCEDURE addGenresToContent(IN genres_id INT, IN c_id BIGINT, IN ct_id BIGINT)
BEGIN
    DECLARE max_amount TINYINT;
    DECLARE amount TINYINT;

    SET max_amount = (SELECT ct.ct_max_genres_amont FROM content_types ct WHERE ct.ct_id = ct_id);
    SET amount = (SELECT content.c_genres_amount FROM content WHERE ct_id = content.ct_id AND c_id = content.c_id);

    IF amount <= max_amount THEN
        INSERT content_m2m_content_genres (c_id, cg_id, ct_id) VALUES (c_id, genres_id, ct_id);
    ELSE
        BEGIN
            SELECT 'Error. Content have max amount of genres.' AS message;
        END;
    END IF;
END;
```

У каждого типа контента есть ограничение на количество жанров, которым он может принадлежать.

```
CREATE PROCEDURE addEpisodeToSeason(IN c_id BIGINT, IN ct_id BIGINT, IN season_id SMALLINT, resolution_id SMALLINT, IN url VARCHAR(300))
BEGIN
    DECLARE ep_amount TINYINT;
    DECLARE amount TINYINT;

    SET ep_amount = (SELECT css.css_episodes_amount
                     FROM content_series_seasons css
                     WHERE css.c_id = c_id
                       AND css.ct_id = ct_id
                       AND css.css_id = season_id);


    SET amount = (SELECT COUNT(csu.cu_episode_id)
                  FROM content_series_url csu
                  WHERE csu.c_id = c_id
                    AND csu.ct_id = ct_id
                    AND csu.css_id = season_id);

    IF amount <= ep_amount THEN
        INSERT content_series_url (ct_id, c_id, css_id, r_id, csu_url) VALUES (ct_id,c_id, season_id, resolution_id,url);
    ELSE
        BEGIN
            SELECT 'Error. Content have max amount of genres.' AS message;
        END;
    END IF;
END
```

Количество эпизодов каждого сезона устанавливается заранее, и количество ссылок на эпизод должно быть ограничено данным значением.

```
CREATE PROCEDURE addContentToProfile(IN profile_name VARCHAR(310), IN content_id BIGINT, IN content_type VARCHAR(50),
                                     IN subscribe_type VARCHAR(50))
BEGIN
    DECLARE profile_id TINYINT;
    DECLARE content_type_id TINYINT;
    DECLARE subscribe_type_id TINYINT;

    SET profile_id = (SELECT p_id FROM profiles WHERE p_name = profile_name);
    SET content_type_id = (SELECT ct_id FROM content_types WHERE ct_description = content_type);
    SET subscribe_type_id = (SELECT ust_id FROM user_subscription_types WHERE ust_descriprion = subscribe_type);

    INSERT profile_types_m2m_contents_user_subscription_types (c_id, pt_id, ct_id, ust_id)
    VALUES (content_id, profile_id, content_type_id, subscribe_type_id);
END;
```

Создана процедура, для более удобного добавления контента в список доступного контента для данного профиля с данным типом подписки.

```sql
CREATE PROCEDURE AndAddResolutionToAllSubscribe(IN resolution_id SMALLINT)
BEGIN

    DECLARE lastRows INT DEFAULT 0;
    DECLARE startRows INT DEFAULT 0;
    SELECT COUNT(*) FROM user_subscription_types INTO lastRows;
    SET startRows = 0;

    WHILE startRows < lastRows
        DO
            INSERT user_subscription_types_m2m_resolutions (ust_id, r_id)
            VALUES ((SELECT ust_id
                     FROM user_subscription_types
                     LIMIT startRows ,1), resolution_id);

            SET startRows = startRows + 1;
        END WHILE;
END;
```

Ссоздана процедура для упрощения добавления разрешения в типы подписок