Тема 11

Проект по база от данни за ТВ сериали

Изготвил: Марио Симеонов, ф.н. 7MI0700043

1. Обхват на модела. Дефиниране на задачата.

Информационна система съхранява информация за ТВ сериали. Съхранява се име, структура(episodic, serialized, anthology), дата на начало на сериала, дата за край на сериала, създател(showrunner),брой сезони и начин на предаване(streaming,cable). Съхранява се информация за сезони с данни за номер на сезон, брой епизоди, име на продукция, начална дата на сезона и крайна дата на сезона. Съхранява се също и информация за епизоди с данни за номер на епизод, дата на излъчване на епизода и продължителност на епизода ,режисьор и сценарист. Номерът на сезона и името на продукцията заедно определят уникално всеки сезон. Номерът на епизода е уникален в рамките на сезона. В информационната система се съхранява и информация за герои с данни за номер на герой — цяло положително число, уникално за всеки герой, име на герой, роля на герой и име на актьора, който играе или озвучава образа на героя. Един герой може да участва в много епизоди и в един епизод могат да участват много герои.

2. Множества от същности и техните атрибути

Сериал: име, структура, дата за начало на сериала, дата за край на сериала, създател, брой сезони, начин на предаване Сезон: номер на сезон, име на продукция, брой епизоди, начална дата на сезона и крайна дата на сезона Епизод: номер на епизод, дата на излъчване на епизода ,продължителност на епизода, режисьор и сценарист Герой: номер на герой, име на герой, роля на герой и име на актьор

3. Домейн на атрибутите

Сериал: име: низ, структура: низ, дата за начало на сериала: низ, дата за край на сериала: низ, създател: низ, брой сезони: цяло положително число , начин на предаване: низ

Сезон: номер на сезон: цяло положително число, име на продукция: низ, брой епизоди: цяло положително число, начална дата на сезона: низ и крайна дата на сезона: низ

Епизод: номер на епизод: цяло положително число, дата на излъчване на епизода: низ, продължителност на епизода: цяло положително число, режисьор: низ и сценарист: низ

Герой: номер на герой: цяло положително число, име на герой: низ, роля на герой: низ и име на актьор: низ

4. Връзки

Един showrunner може да е създал много сериали и един сериал може да е създаден от много създатели. Един сезон може да има много продукции и една продукция може да има много сезони. Един епизод може да има повече от един режисьор и един режисьор може да режисира повече от един епизод. Един епизод може да има повече от един сценарист и един сценарист може да напише повече от един епизод. Един герой може да участва в много епизоди и в един епизод могат да участват много герои. Един герой се играе от един актьор и един актьор може да играе повече от един герой. (пример: близнаци) Сериал - име: еднозначно определя сериала Сезон - номер на сезон, име на продукция: еднозначно определя сезона Епизод - номер на епизод: еднозначно определя епизода в рамките на сезона Герой - номер на герой: еднозначно определя героя

5. Ключове

Сериал - име: еднозначно определя сериала

Сезон - номер на сезон, име на продукция: еднозначно определя сезона

Епизод - номер на епизод: еднозначно определя епизода в рамките на сезона

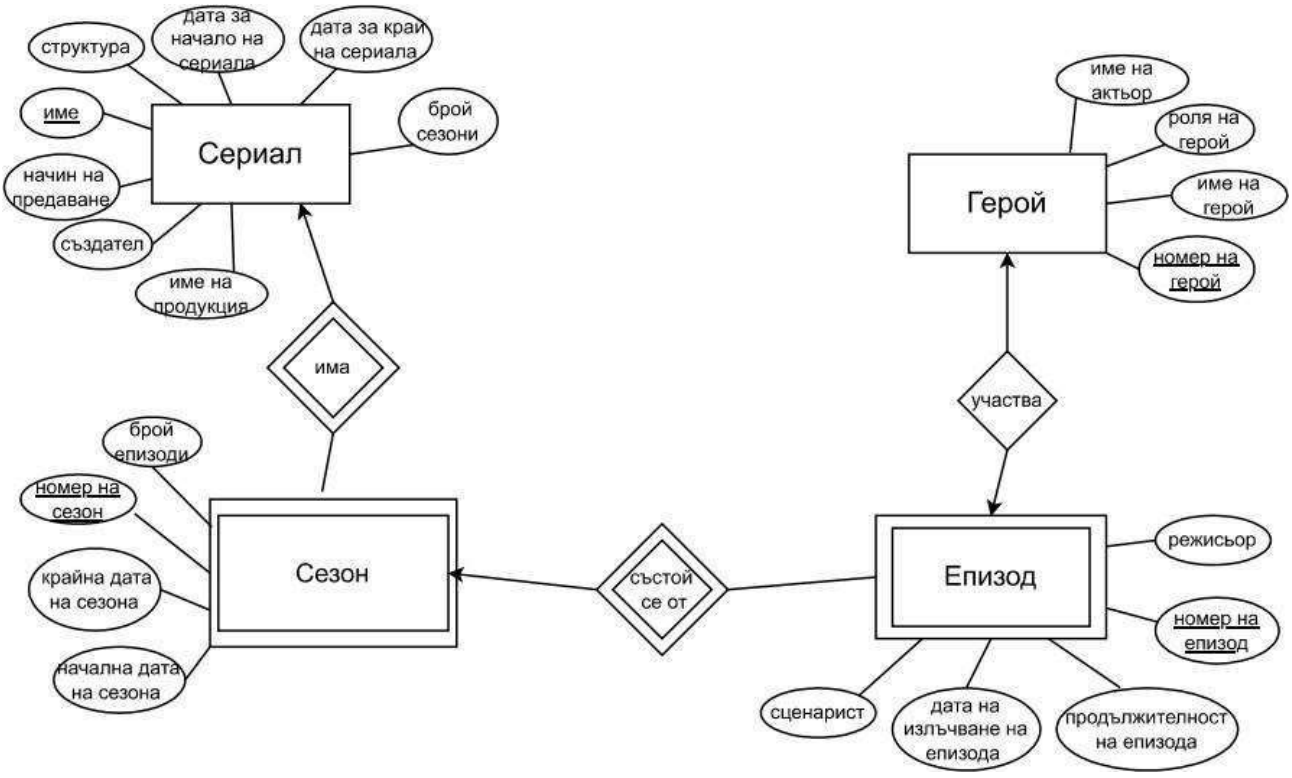
Герой - номер на герой: еднозначно определя героя

6. Правила и проверки

Проверка: Свършил ли е сериала?

За всички положителни числа, проверка за >0

7. E/R модел на данни



8. Релационен модел на данни

Сериал- (име, структура, дата за начало на сериала, дата за край на сериала, брой сезони, начин на предаване, създател, име на продукция)
Сезон-(номер на сезон, име на Сериал, брой епизоди, начална дата на сезона, крайна дата на сезона)
Епизод-(номер на епизод, номер на сезон, име на Сериал, режисьор, сценарист, продължителност, дата на излъчване) Герой (номер на герой, име на актьор, име на герой, роля на герой)
Участва-(номер на герой, номер на епизод, номер на сезон, име на Сериал)

Ключове:
- Сериал-име
- Сезон-номер на сезон.
- Епизод номер на епизод
- Герой- номер на герой

Чужди ключове:
Сезон(име на Сериал) -> Сериал (име на сериал)
-Епизод(име на Сериал) -> Сериал (име на сериал)
-Епизод(номер на сезон)-> Сезон (номер на епизод) -Участва(име на Сериал) -> Сериал (име на сериал) -Участва (номер на сезон) -> Сезон (номер на епизод)
-Участва (номер на герой) -> Герой (номер на герой)
-Участва (номер на епизод) -> Епизод (номер на епизод)

Check:

Сериал (брой епизоди-1,2,3,...)

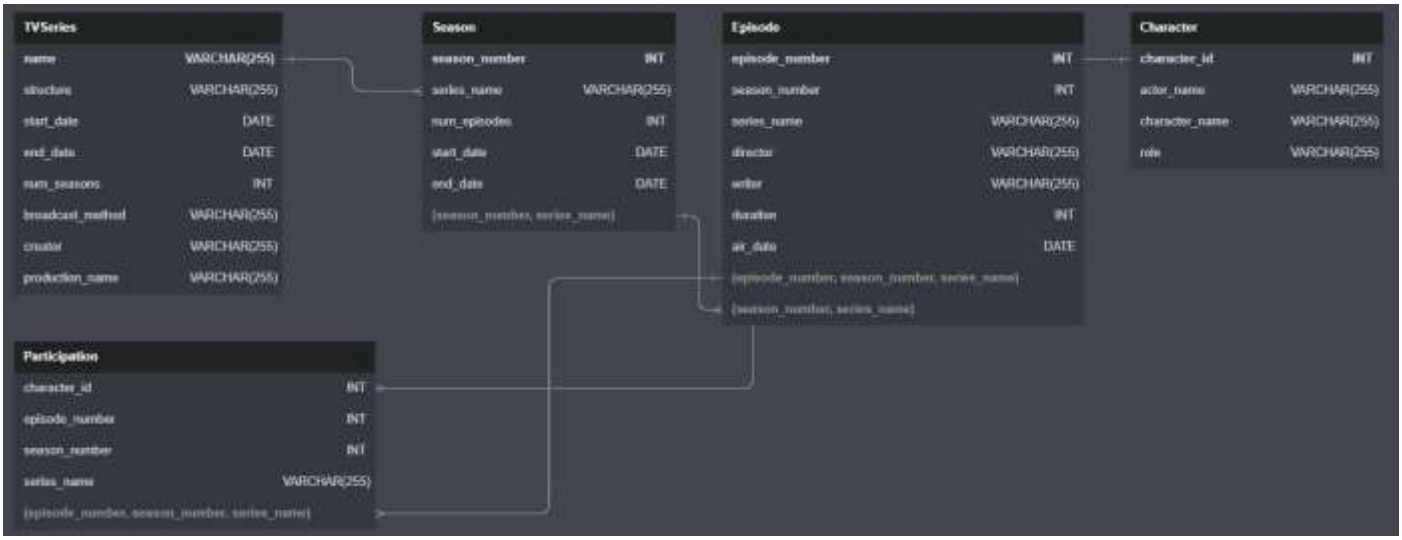
Сезон-(номер на сезон-1,2,3,...)

Сезон-(брой епизоди-1,2,3,...)

Епизод (номер на епизод-1,2,3,...)

Епизод (продължителност-_h_min)

9. Схема на базата от данни



10. Функции

```sql
-- Function 1: Get the total number of episodes for a series
CREATE FUNCTION GetTotalEpisodes(series_name VARCHAR(255)) RETURNS INT
BEGIN
    DECLARE total_episodes INT;
    SELECT SUM(num_episodes) INTO total_episodes
    FROM Season
    WHERE series_name = series_name;
    RETURN total_episodes;
END;

-- Function 2: Get the average duration of episodes for a series
CREATE FUNCTION GetAverageDuration(series_name VARCHAR(255)) RETURNS DECIMAL(5,2)
BEGIN
    DECLARE avg_duration DECIMAL(5,2);
    SELECT AVG(duration) INTO avg_duration
    FROM Episode
    WHERE series_name = series_name;
    RETURN avg_duration;
END;
```

Функция 1: GetTotalEpisodes Тази функция приема име на сериал като вход и връща общия брой на епизодите за този сериал. Използва се SELECT заявка, за да се сумират стойностите от колоната 'num_episodes' в таблицата 'Season', където подаденото 'series_name' съвпада. Резултатът се съхранява в променливата 'total_episodes', която после се връща.

Функция 2: GetAverageDuration Тази функция приема име на сериал като вход и връща средната продължителност на епизодите за този сериал. Използва SELECT заявка, за да изчисли средната стойност от колоната 'duration' в таблицата 'Episode', където подаденото 'series_name' съвпада. Резултатът се съхранява в променливата 'avg_duration' и се връща.

11. Изгледи

```sql
-- View 1: List all characters and their roles
CREATE VIEW CharacterRoles AS
SELECT character_name, role
FROM Character;

SELECT * FROM CharacterRoles;

-- View 2: List episodes with their directors and writers
CREATE VIEW EpisodeDetails AS
SELECT series_name, season_number, episode_number, director, writer
FROM Episode;

SELECT * FROM EpisodeDetails;
```

Изглед 1: CharacterRoles Този изглед изброява всички герои заедно с техните роли. Избира колоните 'character_name' и 'role' от таблицата 'Character' и след това може да бъде извикан с 'SELECT * FROM CharacterRoles.‘

Изглед 2: EpisodeDetails Този изглед предоставя детайли за епизодите, включително име на сериал, номер на сезон, номер на епизод, режисьор и сценарист. Избира тези колони от таблицата 'Episode' и може да бъде извикан с 'SELECT * FROM EpisodeDetails.'

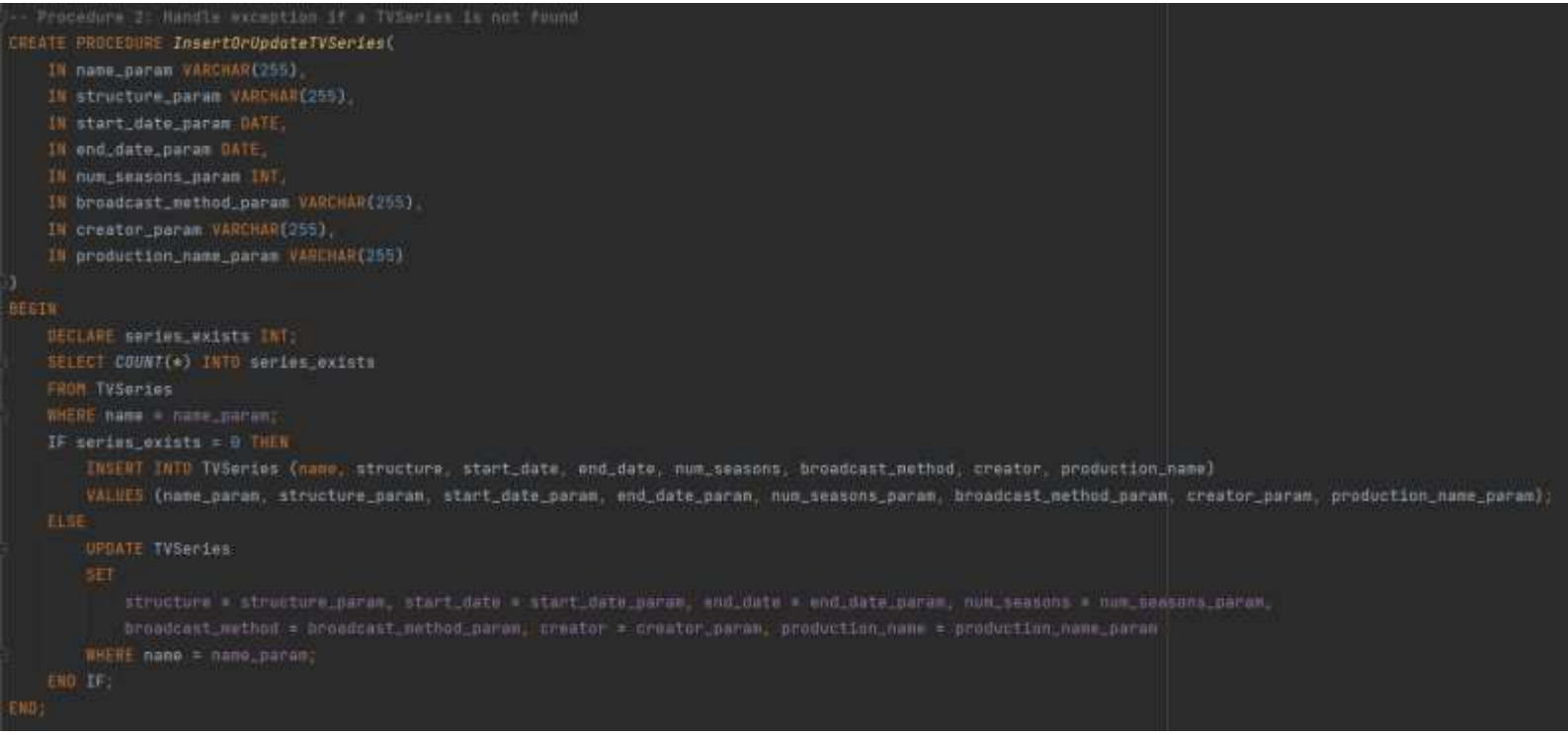
12. Процедури

```sql
-- Procedure 1: Handle exception if a character is not found
CREATE PROCEDURE InsertCharacter(
    IN actor_name_param VARCHAR(255),
    IN character_name_param VARCHAR(255),
    IN role_param VARCHAR(255)
)
BEGIN
    DECLARE character_id_param INT;

    SELECT character_id INTO character_id_param
    FROM Character
    WHERE character_name = character_name_param;

    IF character_id_param IS NULL THEN
        INSERT INTO Character (actor_name, character_name, role)
        VALUES (actor_name_param, character_name_param, role_param);
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Character already exists';
    END IF;
END;
```

Процедура 1: InsertCharacter Тази процедура е предназначена за вмъкване на нов герой в таблицата 'Character'. Приема имена на актьор, герой и роля като входни параметри. Първо проверява дали героят вече съществува, като използва SELECT заявка в таблицата 'Character'. Ако героят не съществува, вмъква новия герой; в противен случай генерира изключение, указващо, че героят вече съществува.



```sql
-- Procedure 2: Handle exception if a TVSeries is not found
CREATE PROCEDURE InsertOrUpdateTVSeries(
    IN name_param VARCHAR(255),
    IN structure_param VARCHAR(255),
    IN start_date_param DATE,
    IN end_date_param DATE,
    IN num_seasons_param INT,
    IN broadcast_method_param VARCHAR(255),
    IN creator_param VARCHAR(255),
    IN production_name_param VARCHAR(255)
)
BEGIN
    DECLARE series_exists INT;
    SELECT COUNT(*) INTO series_exists
    FROM TVSeries
    WHERE name = name_param;
    IF series_exists = 0 THEN
        INSERT INTO TVSeries (name, structure, start_date, end_date, num_seasons, broadcast_method, creator, production_name)
        VALUES (name_param, structure_param, start_date_param, end_date_param, num_seasons_param, broadcast_method_param, creator_param, production_name_param);
    ELSE
        UPDATE TVSeries
        SET
            structure = structure_param, start_date = start_date_param, end_date = end_date_param, num_seasons = num_seasons_param,
            broadcast_method = broadcast_method_param, creator = creator_param, production_name = production_name_param
        WHERE name = name_param;
    END IF;
END;
```

Процедура 2: InsertOrUpdateTVSeries Тази процедура се грижи за вмъкването или актуализирането на телевизионен сериал в таблицата 'TVSeries'. Приема различни параметри относно телевизионния сериал и проверява дали сериалът вече съществува, като брои редовете в таблицата 'TVSeries' с подаденото име на сериала. Ако сериалът не съществува, вмъква нов запис; в противен случай актуализира съществуващия запис с новата информация.

```sql
CREATE PROCEDURE CalculateTotalDuration(IN series_name VARCHAR(255), IN season_number INT, OUT total_duration INT)
BEGIN
    DECLARE at_end INT DEFAULT 0;
    DECLARE episode_duration INT;

    DECLARE episode_cursor CURSOR FOR
        SELECT duration
        FROM Episode
        WHERE series_name = series_name AND season_number = season_number;

    OPEN episode_cursor;

    SET total_duration = 0;

    FETCH episode_cursor INTO episode_duration;

    WHILE(at_end = 0) DO
        SET total_duration = total_duration + episode_duration;
        FETCH episode_cursor INTO episode_duration;
    END WHILE;

    CLOSE episode_cursor;
END;
```

Процедура 3: CalculateTotalDuration е създадена с цел изчисляване на общата продължителност на епизодите за даден телевизионен сериал и сезон. Тя приема два входни параметъра - име на сериала (VARCHAR) и номер на сезона (INT), както и OUT параметър, в който се записва изчислената обща продължителност (INT). Процедурата използва курсор, за да извлече продължителността на всеки епизод от съответния сериал и сезон. След това тя сумира тези продължителности и ги записва в OUT параметъра total_duration. Въпреки това, в кода се появява потенциална грешка, тъй като условието в WHILE цикъла (WHILE(at_end = 0)) може да предизвика безкраен цикъл. Препоръчително е това условие да бъде коригирано, за да осигури правилното прекъсване на цикъла в зависимост от наличието на още епизоди за извличане.

```sql
-- Trigger 1
CREATE TRIGGER After_Episode_Insert
AFTER INSERT ON Episode
REFERENCING NEW AS N
FOR EACH ROW
BEGIN
    UPDATE Season
    SET end_date = (
        SELECT MAX(air_date)
        FROM Episode
        WHERE season_number = Season.season_number
          AND series_name = Season.series_name
    )
    WHERE season_number = N.season_number
      AND series_name = N.series_name;
END;
```

Trigger: After_Episode_Insert се активира след вмъкване на нов епизод в таблицата 'Episode'. Целта му е да обнови крайната дата на сезона в таблицата 'Season', използвайки максималната дата на излъчване от всички епизоди в този сезон.