Projektarbeit: Datenbank zur Filmverwaltung

1. Fachlicher Anwendungsbezug

Grundidee und Szenario

Grundidee der Datenbank ist die Entwicklung eines zentralen, digitalen Verzeichnisses zur Verwaltung der gemeinsamen Filmsammlung eines privaten Haushalts (z.B. einer Familie oder WG).

Als Ausgangsproblem besitzt ein filminteressierter Haushalt eine über Jahre gewachsene, gemeinsame Sammlung an Filmen und Serien. Diese sind auf diversen Medien (z.B. Blu-rays, DVDs, digital) verteilt, was zu einem zunehmenden Verlust der Übersicht führt.

Nutzen und Sinn

- **Primäres Ziel:** Die Datenbank soll Ordnung, Struktur und eine klare Übersicht in die gemeinsame Mediensammlung des Haushalts bringen und die Planung zukünftiger Filmabende oder -käufe erleichtern.
- **Zentrale Katalogisierung:** Alle Filme und Serien werden an einem einzigen Ort erfasst. Dies entkoppelt die Verwaltung vom physischen oder digitalen Speicherort der Medien.
- **Leistungsfähige Such- und Filterfunktionen:** Die Sammlung kann gezielt nach einer Vielzahl von Kriterien durchsucht werden, darunter Genre, Erscheinungsjahr oder Regisseur. Zusätzlich kann jeder Nutzer nach seiner eigenen, persönlichen Bewertung filtern.
- **Verwaltung einer "Watchlist":** Jeder Nutzer kann eine eigene, persönliche Wunschliste führen, um Filmempfehlungen oder Kaufwünsche systematisch zu erfassen.
- **Verwaltung bereits gesehener Filme:** Jeder Nutzer kann für sich markieren, welche Filme er bereits gesehen hat. Dabei können individuelle Informationen wie das Datum des Sehens und eine persönliche Bewertung hinterlegt werden.
- **Potenzial für Statistiken:** Die erfassten Daten bilden die Grundlage für interessante Auswertungen. Dies umfasst sowohl Statistiken über die gesamte Sammlung (z. B. "Welches Genre ist am stärksten vertreten?") als auch benutzerspezifische Auswertungen.

Einschränkungen/Abgrenzung

- **Kein Streaming-Dienst oder Medien-Player:** Die Datenbank dient ausschließlich der Verwaltung von Metadaten. Sie beinhaltet nicht die eigentlichen Filmdateien und bietet keine integrierte Funktion zum Abspielen der Medien.
- **Fokus auf private Nutzung im kleinen Kreis:** Die Benutzer- und Rollenverwaltung ist für einen privaten, überschaubaren Personenkreis ausgelegt und nicht auf Skalierbarkeit für tausende Nutzer ausgelegt.
- **Manuelle Dateneinpflege:** Es wird keine Schnittstelle zu externen, öffentlichen Filmdatenbanken implementiert. Alle Filminformationen müssen manuell eingegeben werden.
- **Keine kommerziellen Funktionen:** Das System ist eine reine Verwaltungsanwendung. Es werden keine kommerziellen Aspekte (Shopsystem, Lizenzverwaltung etc.) abgebildet.
- **Kein Verleih- oder Bestandsmanagement:** Die Datenbank erfasst, welche Filme vorhanden sind, beinhaltet aber keine Funktion zur Verwaltung eines Verleihs an andere Personen.

2. Anforderungsanalyse

Funktionale Anforderungen

1. Verwaltung der Filmsammlung:

- Jedes `Mitglied` (und der `Administrator`) kann neue Filme, Serien und Personen in die Datenbank eintragen.
- Jedes `Mitglied` (und der `Administrator`) kann die Daten bereits existierender Einträge bearbeiten.
- Ausschließlich ein `Administrator` kann Einträge endgültig aus den Stammdaten (Filme, Personen etc.) löschen.
- Filme können einer übergeordneten `Filmreihe` zugeordnet werden.
- Die Beziehung zwischen Filmen und Personen wird über eine (n:m) Verknüpfungstabelle `Film_Beteiligungen` realisiert. Diese Tabelle speichert `filmID`, `personID` und die Rollen der Person (`istRegisseur`, `istSchauspieler`).

2. Personalisierte Nutzerfunktionen:

- Jeder registrierte Nutzer (`Administrator` oder `Mitglied`) kann Filme zu seiner persönlichen `Watchlist` hinzufügen oder davon entfernen.
- Jeder registrierte Nutzer (`Administrator` oder `Mitglied`) kann Filme als "gesehen" markieren und eine persönliche Bewertung (1-10) sowie ein Datum hinterlegen.
- **Gewährleistung der Privatsphäre:** Das System (via `VIEW`s) stellt sicher, dass ein Nutzer ausschließlich auf seine eigenen personalisierten Einträge zugreifen kann.

3. Datenauswertung und Suche:

- Alle Nutzer (`Gast` eingeschlossen) können die Sammlung nach Kriterien wie Titel, Genre, Regisseur oder Filmreihe durchsuchen.
- Ein `Gast` hat ausschließlich Lesezugriff auf öffentliche Filmdaten und kann keine personalisierten Listen einsehen oder bearbeiten.
- Das System kann Detailinformationen zu einem Film anzeigen, inklusive aller beteiligten Personen.
- Ein `Mitglied` oder `Admin` kann seine persönliche Watchlist und seine Liste der gesehenen Filme einsehen.

4. Benutzer- und Rollenverwaltung:

- Ein `Administrator` kann neue Benutzer anlegen und ihnen eine Rolle (`Administrator`, `Mitglied`, `Gast`) zuweisen.

Nicht-funktionale Anforderungen

- **Datenkonsistenz und -integrität:** Die Datenbank sichert durch `PRIMARY KEY`, `FOREIGN KEY` und `CHECK`-Constraints (z.B. `chk_bewertung`) die Stimmigkeit der Daten.
- **Bedienbarkeit und Zuverlässigkeit:** Die Datenbank ist durch die bereitgestellten SQL-Skripte (`main.sql`, `data.sql`) und die `README.md` auf einem anderen System lauffähig und reproduzierbar.
- **Sicherheit:** Der Zugriff ist über ein Berechtigungskonzept geregelt. Die Zuweisung von Rechten (`GRANT`) erfolgt auf Basis von Rollen (`rolle_admin`, `rolle_mitglied`, `rolle_gast`).

---

3. Technische Umsetzung

Inhalt der `main.sql` Datei

Das Skript `main.sql` ist in drei Hauptabschnitte unterteilt, die die gesamte Struktur und Sicherheit der Datenbank definieren:

1. **Abschnitt 1: Grundlegendes Datenbankschema**
    - Erstellt die Datenbank `filmverwaltung` (nachdem eine eventuell vorhandene Version gelöscht wurde).
    - Erstellt alle 9 Kerntabellen (`Filme`, `Personen`, `Benutzer`, `Rollen`, `Watchlist` etc.) mit den notwendigen Primärschlüsseln, Fremdschlüsseln, `UNIQUE`-Constraints und `CHECK`-Constraints.
2. **Abschnitt 2: Kernsystem und Berechtigungen**
    - Befüllt die Anwendungstabellen `Rollen` und `Benutzer` mit den Stammdaten für die Logik.
    - Erstellt die MariaDB-Systemrollen (`rolle_admin`, `rolle_mitglied`, `rolle_gast`).
    - Erstellt die MariaDB-Systembenutzer (z.B. 'julian', 'max', 'sophie') mit Passwörtern.
    - Erstellt die beiden Sicherheits-`VIEW`s (`MeineWatchlist`, `MeineGesehenenFilme`), die als "Brücke" zwischen den Systembenutzern und der Anwendungslogik dienen.
    - Vergibt detaillierte `GRANT`-Berechtigungen an die Rollen.
    - Weist den Benutzern ihre jeweiligen Rollen zu und setzt diese als `DEFAULT ROLE`, damit sie beim Login automatisch aktiv sind.
3. **Abschnitt 3: Datenbefüllung**
    - Befüllt die Tabellen `Genres` und `Filmreihen` mit den grundlegenden Kategorien
    - Fügt Beispieldaten für `Personen` (Regisseure und Schauspieler) hinzu, die in den Filmen vorkommen.
    - Befüllt die Tabelle `Filme` mit einer umfangreichen Sammlung von Beispielfilmen, inklusive Metadaten.
    - Verknüpft Filme mit Personen über die Tabelle `Film_Beteiligungen` und legt dabei fest, ob die Person als Regisseur und/oder Schauspieler beteiligt war.
    - Befüllt die personalisierten Listen (`Watchlist`, `GeseheneFilme`) mit Beispieldaten für jeden Benutzer, sodass jeder Nutzer 3–5 Filme auf seiner Watchlist und seiner Liste gesehener Filme hat. Dies ermöglicht das direkte Testen der personalisierten Funktionen und Abfragen.

---

4. SQL-Abfragen

Frage 1: "Welche Filme (Titel und Erscheinungsjahr) hat der Benutzer 'max' auf seiner persönlichen Watchlist?"

```
-- Frage 1:
-- Diese Abfrage kann nur als Benutzer mit Admin-Rechten ausgeführt werden!

SELECT
    F.titel,
    F.erscheinungsjahr
FROM
```

```
    Watchlist W

-- Verknüpfe die Watchlist-Einträge mit Filmen
JOIN
    Filme F ON W.filmID = F.filmID

-- Verknüpfe Watchlist-Einträge mit den Benutzern
JOIN
    Benutzer B ON W.benutzerID = B.benutzerID
WHERE
    B.benutzerName = 'max' -- Filtert auf Benutzer
ORDER BY
    F.titel;
```

Frage 2: "Welche 5 Personen sind in der gesamten Sammlung am häufigsten als Schauspieler vertreten? Zeige den Namen der Person und die Anzahl der Filme, in denen sie mitspielt."

```
-- Frage 2:
-- Abfrage nutzt Aggregation (COUNT) und Filter (WHERE istSchauspieler).

SELECT
    -- Kombiniere Vor- und Nachname für Ausgabe
    CONCAT(P.vorname, ' ', P.name) AS personName,

    -- Zähle Anzahl der Filmeinträge für die Person
    COUNT(FB.filmID) AS anzahlFilme
FROM
    Personen P
JOIN
    Film_Beteiligungen FB ON P.personID = FB.personID
WHERE
    -- Stelle sicher, dass die Person auch Schauspieler ist
    FB.istSchauspieler = TRUE
GROUP BY
    P.personID, personName -- Gruppiere die Zählung pro Person
ORDER BY
    anzahlFilme DESC -- Sortiere von der höchsten zur niedrigsten Anzahl
LIMIT 5; -- Zeige nur die Top 5 an
```

Frage 3: "Liste für jeden Benutzer (ausgenommen 'Gast') seine Top 3 am besten bewerteten Filme auf. Die Abfrage soll den Benutzernamen, den Filmtitel und die persönliche Bewertung anzeigen."

```
-- Frage 3:
-- Abfrage nutzt CTE (WITH...) und Window Function (ROW_NUMBER()).

-- 1. CTE definieren 'RankedFilme'
WITH RankedFilme AS (
    SELECT
        B.benutzerName,
```

```
        F.titel,
        GF.persoenlicheBewertung,

        -- Window Function: Erstellt eine separate Rangliste (rang) für jeden
Benutzer (PARTITION BY)
        -- sortiert nach der Bewertung von hoch nach niedrig (ORDER BY ... DESC)
        ROW_NUMBER() OVER(
            PARTITION BY B.benutzerName
            ORDER BY GF.persoenlicheBewertung DESC, F.titel ASC
        ) AS rang
    FROM
        GeseheneFilme GF
    JOIN
        Benutzer B ON GF.benutzerID = B.benutzerID
    JOIN
        Filme F ON GF.filmID = F.filmID
    JOIN
        Rollen R ON B.rollenID = R.rollenID
    WHERE
        R.rollenName != 'Gast' -- Schließt "Gast"-Benutzer aus
)

-- 2. Finale Abfrage:
-- Wähle nur Top 3 (rang <= 3) aus der CTE aus.
SELECT
    benutzerName,
    titel,
    persoenlicheBewertung,
    rang
FROM
    RankedFilme
WHERE
    rang <= 3
ORDER BY
    benutzerName, rang;
```