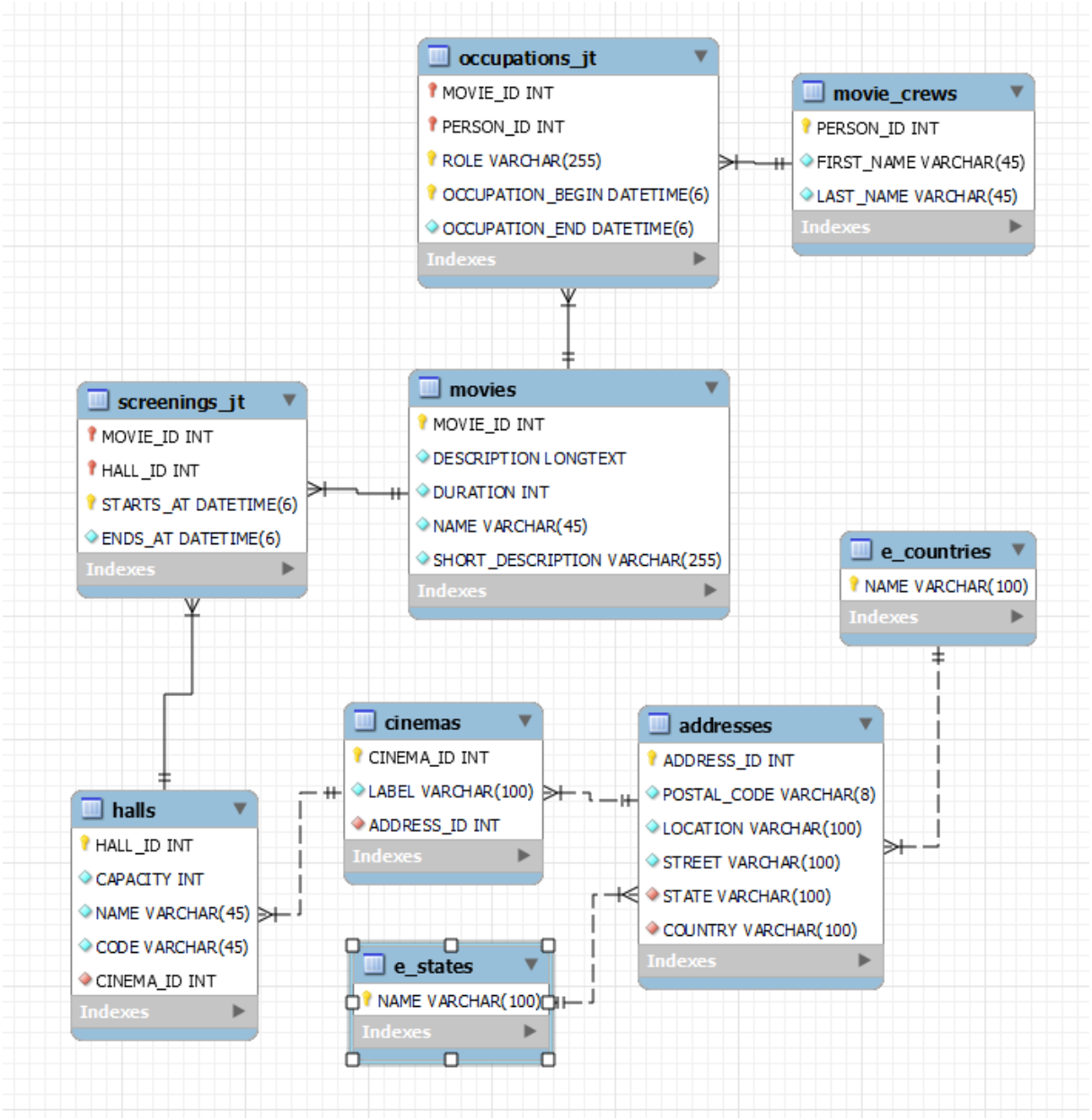
Übung 3

1. Erstellung des Cinema-Models

Für die Datenzugriffsschicht benötigen wir mehrere Entities die in Form von C# Klassen in unserem Projekt Model im Ordner Entities abgebildet werden. Das fertige Model wird durch das folgende ER-Diagramm beschrieben.



Ihr findet neben der Übungsangabe auch noch das ER-Diagramm als .mwb File, das in der MySql Workbench importiert werden kann für eine interaktive Ansicht des Models.

Erstelle nun alle benötigten Model-Klassen so, dass durch Migration mittels Entity Framework Core, das oben beschriebene Model in deiner Datenbank entsteht. Ich empfehle dir die Entwicklung Step-by-Step durchzuführen und nach jeder neu hinzugefügten Klasse eine Migration zu erstellen und diese in die Datenbank einzuspielen, um zu überprüfen, ob die Definition der Model-Klassen im C#-Code korrekt ist. Die folgenden Hinweise sollen dir außerdem bei der Erstellung behilflich sein.

- Du kannst das Mapping auf die Datenbank mit einer Mischung aus Attributen und der Verwendung der Fluent API im CinemaDbContext definieren. Verwende beide Varianten, je nachdem wo welche Variante sinnvoller ist.
- Alle IDs sind in der Datenbank entsprechend ihrer Entität benannt, zB MOVIE_ID. Alle IDs in C# sollen aber einheitlich den Identifier Id verwenden.
- In der Datenbank werden Identifier in CAPS mit Unterstrichen als Trennzeichen verwendet. In C# verwenden wir aber einheitlich Pascal-Case. ZB: SHORT_DESCRIPTION / ShortDescription
- In der Datenbank werden die Tabellen in der Plural-Form bezeichnet, im C#-Code hingegen in der Singular-Form. ZB: MOVIES / Movie
- Alle Zuordnungstabellen für m:n Relationen werden in der Datenbank mit dem Postfix _jt für Join Table gekennzeichnet. Im C# Code entfällt dieser Zusatz.
- Speziell für die Join Tables wird es nötig sein zusammengesetzte Primärschlüssel zu definieren. Die benötigten Schlüssel sind aus dem ER-Diagramm ersichtlich.
- Der Name der C#-Klasse die auf die Tabelle MOVIE_CREWS mappt, soll Person sein.
- Bei der Angabe von Fremdschlüsseln ist es möglich in C# zwei Properties für eine referenzierte Entity zu definieren. Einmal für die ID selbst und einmal für die Referenz auf die verknüpfte Entität. zB:
  public int MovieId { get; set; }
  public Movie Movie { get; set; }
  Versuche dies überall wo es möglich ist zu verwenden, da es später den Zugriff erleichtert.
- Die ROLE in der Tabelle OCCUPATIONS_JT soll durch ein Enum EOccupationType definiert werden, das folgende Werte annehmen kann: ACTOR, DIRECTOR, PRODUCER. Achte darauf, dass die Abbildung auf die Datenbank als String, also durch Verwendung der Enum-Bezeichner selbst erfolgt und nicht zB Integer-Werte in die Datenbank geschrieben werden, damit die Lesbarkeit in der Datenbank erhöht wird.
- Die DURATION in der Tabelle MOVIES kann einen Wert zwischen 0 und 500 annehmen. Das soll durch das Datenbankmodell sichergestellt werden.
- Alle IDs (ausgenommen zusammengesetzte Primärschlüssel in Join Tables) sollen von der Datenbank als AUTO INCREMENT Werte generiert werden.
- Beachte welche der Spalten in der Datenbank NOT NULL gekennzeichnet sind (rote Raute) und definiere die Properties in C# entsprechend.
- Die Spalte COUNTRY in der Tabelle ADRESSES ist ein Fremdschlüssel auf die Tabelle E_COUNTRIES. In diesem Fall wird kein Integer für die ID verwendet, sondern direkt der Name des Landes. Das Property in der Klasse Country soll CountryName heißen um zu verdeutlichen, dass hier gleich direkt der Name des Landes verfügbar ist. Der Vorteil dabei ist, dass der Name des Landes gleich direkt in der Adress-Tabelle ersichtlich ist, jedoch nur Werte annehmen kann, die auch in der E_COUNTRIES Tabelle angelegt wurden.
- Die C# Klasse die auf E_COUNTRIES mappt soll Country heißen.
- Dasselbe Konzept wie bei E_COUNTRIES wird auch für E_STATE verwendet. Dort heißt die entsprechende C#-Klasse State und das Property in Adress nennen wir StateCode.

Achte bei der Ausarbeitung auf alle Details und überprüfe zum Schluss noch einmal:
Entsteht beim Einspielen in die Datenbank dasselbe Model?
Ist alles richtig benannt? Unterscheide dabei zwischen Benennungen in C# und in der Datenbank.