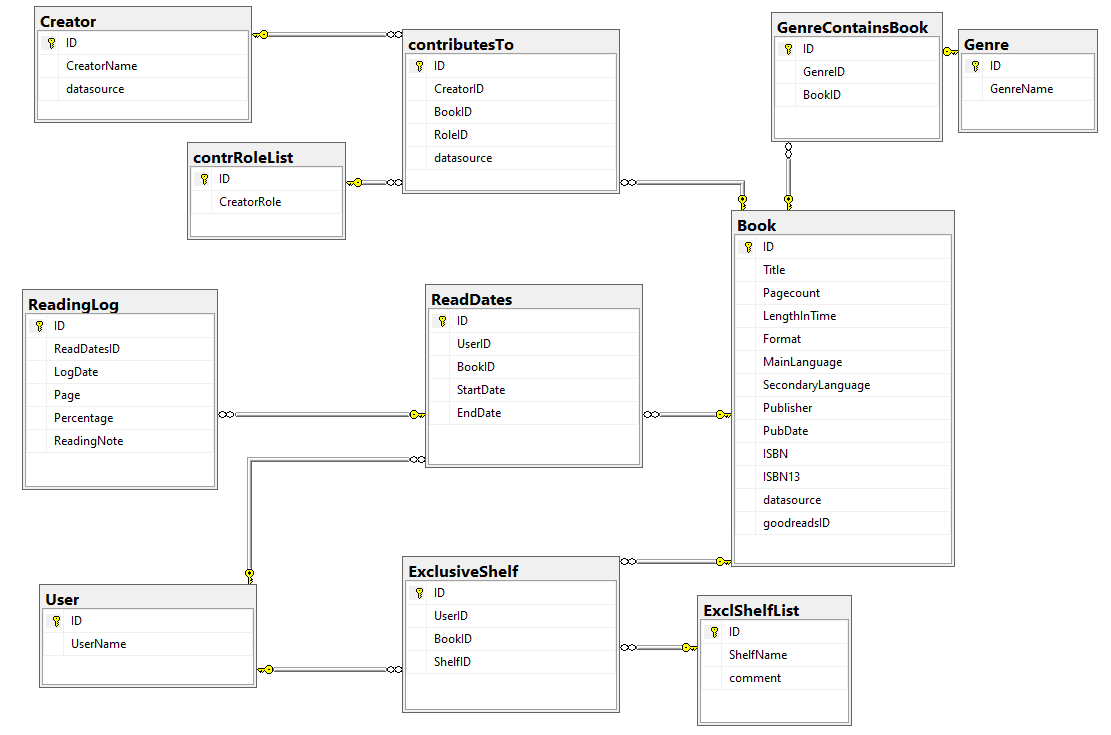
16. Juni 2023

Projektbeschreibung Maria Tammik: BookLog

Die Datenbank speichert einen Katalog an Büchern. Das passiert in den Tabellen **Book** und **Creator** die durch **contributesTo** verbunden werden. Der selbe Creator kann in mehreren verschiedenen Rollen zu einem Buch beitragen. Die möglichen Rollen stehen in der Tabelle **contrRoleList**. Außerdem kann in **GenreContainsBook** angegeben werden zu welchen Genres ein Buch gehört, wobei die möglichen Genres aus der Tabelle **Genre** kommen.

Nutzer aus der Tabelle **User** sich zu einem Buch eintragen, ob sie vorhaben sie zu lesen, schon gelesen haben oder ähnliches. Dazu werden in der Tabelle **ExclusiveShelf** UserID, BookID und ShelfID gespeichert. Der Name des Regals steht in der Tabelle **ExclShelfList**, das die Regale Read, CurrentlyReading, TBR (To Be Read), DNF (Did not finish), Considering, ProbablyNot und Paused bereithält. Der User kann ein bestimmtes Buch zu jedem Zeitpunkt nur auf einem der exklusiven Regale speichern.

Außerdem kann der User in der Tabelle **ReadDates** speichern, von wann bis wann er ein Buch gelesen hat. Während er ein Buch liest kann er in der Tabelle **ReadingLog** speichern, wie weit er ist sowie Notizen.



|  |  |  |  |
| --- | --- | --- | --- |
| Verzeichnis | Dateinamen | Beschreibung | verwendet |
| 001-DB-Create-Scripts | CREATE\_TABLE\_ReadingLog.sql |  |  |
|  | CREATE\_IX\_contributesTo-CreatorRoleBook.sql |  |  |
|  | ADD\_FK\_ReadingLog\_ReadDates.sql |  |  |
|  | ADD\_CK\_ReadingLog\_NotInFuture.sql |  |  |
| 002-Abfragen und Views | CREATE\_View\_UsersWithTheirExclShelves.sql |  |  |
|  | CREATE\_View\_Authors-with-several-books.sql |  |  |
|  | CREATE\_View\_Authors-alphabetically.sql |  |  |
|  | CREATE\_View\_AllBooksByCreatorsWithSeveralBooks.sql |  |  |
|  | CREATE\_View\_ Book-Author.sql |  |  |
|  | CREATE\_View\_ Book-Author |  |  |
| 003- StoredFunctions | CREATE\_sf\_one\_Author\_of\_Book.sql |  |  |
|  | CREATE\_sf\_UserIDfromName.sql | Gibt für einen Nutzernamen die ID zurück. (Nutzernamen sind eindeutig.) |  |
|  | CREATE\_tf\_AuthorsWithSeveralBooks.sql |  |  |
|  | CREATE\_tf\_BooksAndIDByAuthor.sql |  |  |
|  | CREATE\_tf\_BooksByAuthor.sql |  |  |
|  | CREATE\_tf\_BooksByCreator.sql |  |  |
|  | CREATE\_tf\_BooksWithCreatorsAndIDs.sql |  |  |
|  | TEST\_tf\_and\_sf\_BooksAndCreators.sql | Skript um die Funktionen zu testen |  |
| 004- StoredProcedures | CREATE\_sp\_deleteUserByID.sql | Löscht erst alle Einträge eines Nutzers in ReadingLog, ReadDates und ExclusiveShelf, um dann den Nuter aus User zu löschen. |  |
|  | TEST\_sp\_deleteUserByID.sql | Skript um die Prozedur zu testen, legt, wie der Triggertest einen neuen Nutzer mit Bucheintägen an, löscht ihn dann aber wieder.  Enthält unten einen auskommentierten Abschnitt mit dem Prozeduraufruf, um bei Bedarf einen Nutzer einfach löschen zu können. |  |
| 005-DML-Trigger | CREATE\_tr\_ExclusiveShelf\_INSERT\_intoReadingLog\_StartDate | Bei INSERT oder UPDATE in ExclusiveShelf: Wenn das Buch auf CurrentlyReading gestellt wird, wird ein Eintrag für ein neues Lesedatum mit heutigem Starttag in ReadDates angelegt.  Wenn das Buch auf Read gestellt wird und der Eintrag mit dem letzten Startdatum in ReadDates noch kein Enddatum hat, wird das heutige Datum dort eingetragen. |  |
|  | CREATE\_tr\_ExclusiveShelf\_UPDATE\_intoReadDates |  |
|  | TEST\_ExclShelfTrigger | Legt einen Nutzer an, fügt ein Buch auf CurrentlyReading hinzu und verschiebt es dann auf Read um die Trigger im Fall des üblichen Ablaufs zu testen. |  |
| 006-BackUp | Masterskript\_BookLog.sql |  |  |
|  | BackupBookLog |  |  |
| 007-Datenimport | Ich habe vergessen beim ursprünglichen Import in die Datenbank ein Skript zu erstellen |  |  |
|  | INSERT\_INTO\_ SELECT\_from\_gr.sql | Befüllt die Tabellen Book, Author und contributesTo mit Daten aus der gr Tabelle.  Fügt die Spalte Author generisch mit der Rolle Author hinzu, ignoriert zusätzliche Autoren, die in gr in einer weiteren Spalte aufgeführt sind. |  |