Baza danych biblioteki

Projekt zaliczeniowy z przedmiotu Bazy Danych

Mateusz Winiarski

1 lutego 2023

1 Założenia

1.1 Zadanie bazy danych

Celem projektu jest stworzenie bazy danych dla biblioteki ułatwiającej zarządzanie pozycjami, użytkownikami biblioteki oraz wypożyczeniami.

Baza zawiera podstawowe tabele zawierające najbardziej podstawowe dane o pozycjach, książkach, użytkownikach i wypożyczeniach. System może być łatwo rozbudowany o nowe funkcjonalności. W praktyce można umieścić w bazie dużo więcej informacji o książkach, użytkownikach, czy autorach książek.

1.2 Lista tabel

- `Items`
- `Books`
- `Other`
- `Authors`
- `Redactors`
- `Categories`
- `CategoryTree`
- `Users`
- `Borrowings`
- `Fines`

1.3 Diagram ER

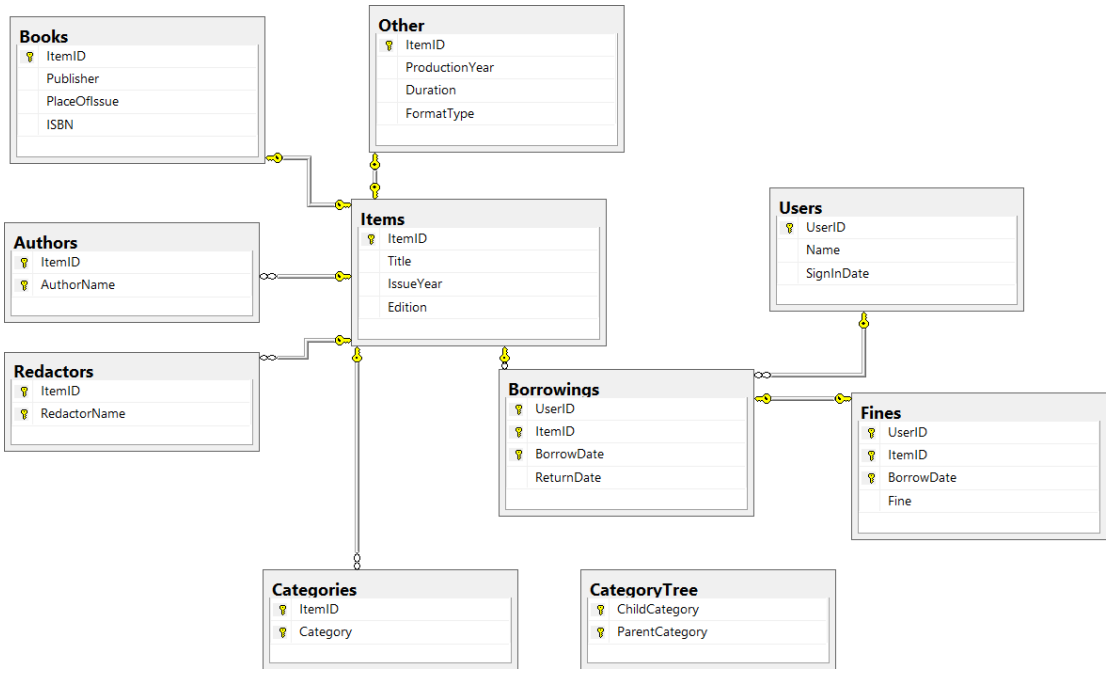

Figure 1: Diagram ER

1.4 Schemat bazy danych

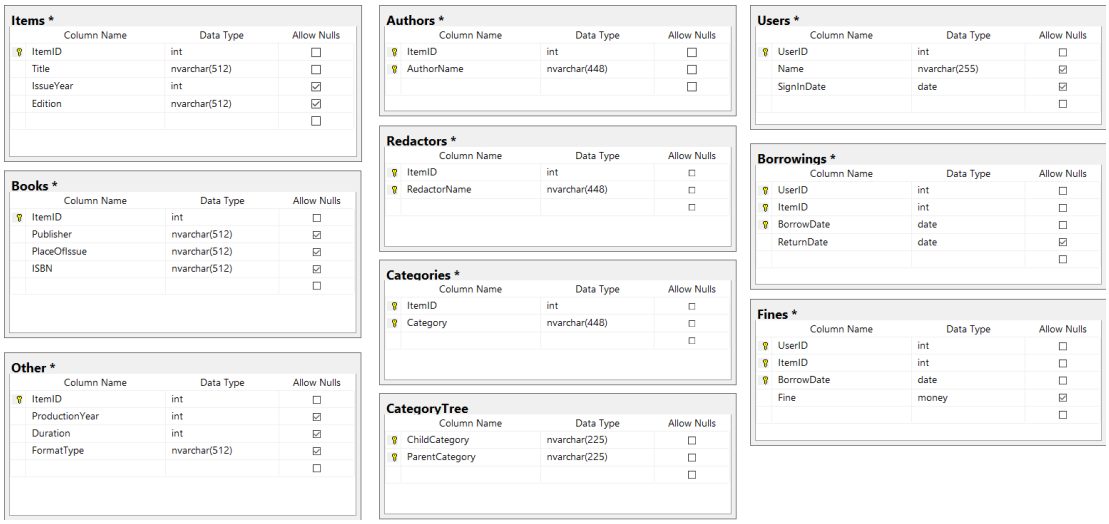

Figure 2: Schemat bazy danych

1.5 Konserwacja bazy danych

Baza danych powinna być często backupowana, ze względu na częste zmiany informacji i ilość pracy, jaka została włożona w spisanie danych wszystkich pozycji. Zapasowa kopia przyrostowa powinna być robiona codziennie (najlepiej w godzinach zamknięcia biblioteki), natomiast kopia całościowa co tydzień.

2 Zapytania

Do bazy danych napisane jest 10 widoków, 5 procedur oraz 5 wyzwalaczy.

2.1 Widoki

Widoki pozwalają na zobaczenie najczęściej wyświetlanych i najbardziej ogólnych zestawień, takich jak ogólnodostępne lista wszystkich książek `AllBooks`, lista wszystkich innych pozycji `AllOther`, lista wszystkich autorów `CountByAuthor`, lista wszystkich kategorii `CountByCategory`, lista książek według dekady `CountByDecade` oraz statystyki książek `ItemBorrowingStats`, a także dostępne dla administratorów lista użytkowników `AllUsers`, lista naliczonych kar `AllFines`, lista wypożyczonych książek `CurrentlyBorrowedBooks` oraz licznik książek z brakami w danych `CountUncompleteBooks`.

```
01 |  CREATE VIEW AllBooks AS
```

```
02 |          SELECT I.Title, B.Publisher, I.IssueYear, B.PlaceOfIssue, I.
       Edition, B.ISBN FROM
03 |          Items I JOIN Books B
04 |          ON I.ItemID = B.ItemID
05 |          ORDER BY I.Title OFFSET 0 ROWS
06 |  GO
```

```
01 |  CREATE VIEW AllOther AS
02 |          SELECT I.Title, I.IssueYear, O.ProductionYear, O.Duration, O.
       FormatType FROM
03 |          Items I JOIN Other O
04 |          ON I.ItemID = O.ItemID
05 |          ORDER BY I.Title OFFSET 0 ROWS
06 |  GO
```

```
01 |  CREATE VIEW AllUsers AS
02 |          SELECT U.*, T.BorrowCount, T.ReturnCount FROM
03 |          (SELECT U.UserID, COUNT(B.BorrowDate) BorrowCount, COUNT(B.
       ReturnDate) ReturnCount  FROM
04 |          Users U LEFT JOIN Borrowings B
05 |          ON U.UserID = B.UserID
06 |          GROUP BY U.UserID) T
07 |          JOIN Users U
08 |          ON U.UserID = T.UserID
09 |  GO
```

```
01 |  CREATE VIEW AllFines AS
02 |          SELECT F.Fine, B.*, U.Name, I.Title FROM
03 |          Fines F JOIN Borrowings B
04 |          ON F.UserID = B.UserID AND F.ItemID = B.ItemID AND F.BorrowDate =
        B.BorrowDate
05 |          JOIN Users U
06 |          ON U.UserID = F.UserID
07 |          JOIN Items I
08 |          ON I.ItemID = F.ItemID
09 |  GO
```

```
01 |  CREATE VIEW CountByAuthor AS
02 |          SELECT AuthorName, COUNT(ItemID) AuthorCount FROM Authors
03 |          GROUP BY AuthorName
04 |          ORDER BY AuthorCount DESC OFFSET 0 ROWS
05 |  GO
```

```
01 |  CREATE VIEW CountByCategory AS
02 |          WITH CategoryCount AS(
03 |                  SELECT DISTINCT I.Title, CS.Category1 Category FROM
```

```
04 |                   Items I  LEFT JOIN (
05 |                           SELECT C.ItemID, C.Category Category1, CT.[
       ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
06 |                           Categories C LEFT JOIN CategoryTree CT
07 |                           ON C.Category = CT.[ChildCategory]
08 |                           LEFT JOIN CategoryTree CT2
09 |                           ON CT.[ParentCategory] = CT2.[ChildCategory]
10 |                   ) CS ON I.ItemID = CS.ItemID
11 |                   WHERE Category1 IS NOT NULL
12 |                   UNION
13 |                   SELECT DISTINCT I.Title, CS.Category2 Category FROM
14 |                   Items I  LEFT JOIN (
15 |                           SELECT C.ItemID, C.Category Category1, CT.[
       ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
16 |                           Categories C LEFT JOIN CategoryTree CT
17 |                           ON C.Category = CT.[ChildCategory]
18 |                           LEFT JOIN CategoryTree CT2
19 |                           ON CT.[ParentCategory] = CT2.[ChildCategory]
20 |                   ) CS ON I.ItemID = CS.ItemID
21 |                   WHERE Category2 IS NOT NULL
22 |                   UNION
23 |                   SELECT DISTINCT I.Title, CS.Category3 Category FROM
24 |                   Items I  LEFT JOIN (
25 |                           SELECT C.ItemID, C.Category Category1, CT.[
       ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
26 |                           Categories C LEFT JOIN CategoryTree CT
27 |                           ON C.Category = CT.[ChildCategory]
28 |                           LEFT JOIN CategoryTree CT2
29 |                           ON CT.[ParentCategory] = CT2.[ChildCategory]
30 |                   ) CS ON I.ItemID = CS.ItemID
31 |                   WHERE Category3 IS NOT NULL
32 |           ) SELECT Category, COUNT(Title) AS Count FROM CategoryCount
33 |           GROUP BY Category
34 |   GO
```

```
01 |   CREATE VIEW CountByDecade AS
02 |           SELECT  ROUND(IssueYear,-1,1) IssueDecade, COUNT(ItemID)
       ItemCount FROM Items
03 |           GROUP BY ROUND(IssueYear,-1,1)
04 |           ORDER BY IssueDecade DESC OFFSET 0 ROWS
05 |   GO
```

```
01 |   CREATE VIEW CurrentlyBorrowedBooks AS
02 |       SELECT U.Name,I.Title,B.BorrowDate, DATEADD(day,60,B.BorrowDate)
       ExpectedReturnDate FROM
03 |       Users U JOIN Borrowings B
04 |       ON U.UserID = B.UserID
05 |       JOIN Items I
06 |       ON B.ItemID = I.ItemID
07 |       WHERE B.ReturnDate IS NULL
08 |   GO
```

```
01 |  CREATE VIEW CountUncompleteBooks AS
02 |           SELECT COUNT(*)-COUNT(I.ItemID) NullItemID, COUNT(*)-COUNT(I.
       Title) NullTitle, COUNT(*)-COUNT(I.IssueYear) NullIssueYear,
03 |            COUNT(*)-COUNT(I. Edition) NullEdition, COUNT(*)-COUNT(B.
       Publisher) NullPublisher,
04 |            COUNT(*)-COUNT(B.PlaceOfIssue) NullPlaceOfIssue, COUNT(*)-COUNT(
       B.ISBN) NullISBN FROM
05 |           Items I JOIN Books B
06 |           ON I.ItemID = B.ItemID
07 |  GO
```

```
01 |  CREATE VIEW ItemsBorrowingStats AS
02 |           SELECT I.Title, T.* FROM (
03 |           SELECT I.ItemID, COUNT(B.BorrowDate) BorrowCount, COUNT(B.
       ReturnDate) ReturnCount,
04 |            IIF(COUNT(B.BorrowDate)-COUNT(B.ReturnDate)>0,'YES','NO')
       IsBorrowed FROM
05 |           Items I LEFT JOIN Borrowings B
06 |           ON I.ItemID = B.ItemID
07 |           GROUP BY I.ItemID
08 |           ) T JOIN Items I
09 |           ON T.ItemID = I.ItemID
10 |  GO
```

2.2 Procedury

Procedury pozwalają na wyświetlenie bardziej szczegółowych informacji dotyczących pojedynczego elementu listy, wybranego przez użytkownika lub administratora. Są to dostępne z poziomu wyszukiwania opcje szukania po autorze `SearchByAuthor` i szukania po kategorii `SearchByCategory`, procedura zwracająca wszystkie informacje o książce `ItemAllInformations` oraz wszystkie jej kategorie `BookCategories`, a także dostępna dla administratora procedura zwracająca historię zamówień użytkownika `UserBorrowingHistory`.

```
01 |  CREATE PROCEDURE ItemAllInformations (@SearchTitle NVARCHAR(255)) AS
02 |
03 |           SELECT I.Title, B.Publisher, I.IssueYear, B.PlaceOfIssue, I.
       Edition, B.ISBN FROM
04 |           Items I LEFT JOIN Books B
05 |           ON I.ItemID = B.ItemID
06 |           LEFT JOIN Other O
07 |           ON I.ItemID = O.ItemID
08 |           WHERE I.Title LIKE @SearchTitle
09 |  GO
```

```
01 |  CREATE PROCEDURE UserBorrowingHistory (@SearchUser NVARCHAR(255)) AS
02 |           SELECT U.Name,I.Title,I.ItemID,B.BorrowDate,B.ReturnDate, DATEADD
       (day,60,B.BorrowDate) ExpectedReturnDate, F.Fine FROM
```

```
        Users U JOIN Borrowings B
        ON U.UserID = B.UserID
        JOIN Items I
        ON B.ItemID = I.ItemID
        LEFT JOIN Fines F
        ON B.ItemID = F.ItemID AND B.BorrowDate = F.BorrowDate AND B.
    UserID = F.UserID
        WHERE U.Name LIKE @SearchUser
GO
```

```
CREATE PROCEDURE SearchByAuthor (@SearchAuthor NVARCHAR(255)) AS
        SELECT A.AuthorName, I.Title FROM
        Authors A INNER JOIN Books B
        ON A.ItemID = B.ItemID
        INNER JOIN Items I
        ON B.ItemID = I.ItemID
        WHERE A.AuthorName LIKE @SearchAuthor
GO
```

```
CREATE PROCEDURE BookCategories (@SearchTitle NVARCHAR(255)) AS
                SELECT DISTINCT I.Title, CS.Category1 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                        ) CS ON I.ItemID = CS.ItemID
                WHERE Title LIKE @SearchTitle AND Category1 IS NOT NULL
        UNION
                SELECT DISTINCT I.Title, CS.Category2 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                        ) CS ON I.ItemID = CS.ItemID
                WHERE Title LIKE @SearchTitle AND Category2 IS NOT NULL
        UNION
                SELECT DISTINCT I.Title, CS.Category3 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                        ) CS ON I.ItemID = CS.ItemID
                WHERE Title LIKE @SearchTitle AND Category3 IS NOT NULL
```

```
GO
```

```
CREATE PROCEDURE SearchByCategory (@SearchCategory NVARCHAR(255)) AS
                SELECT DISTINCT I.Title, CS.Category1 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                ) CS ON I.ItemID = CS.ItemID
                WHERE Category1 LIKE @SearchCategory
        UNION
                SELECT DISTINCT I.Title, CS.Category2 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                ) CS ON I.ItemID = CS.ItemID
                WHERE Category2 LIKE @SearchCategory
        UNION
                SELECT DISTINCT I.Title, CS.Category3 Category FROM
                Items I  LEFT JOIN (
                        SELECT C.ItemID, C.Category Category1, CT.[
    ParentCategory] Category2, CT2.[ParentCategory] Category3 FROM
                        Categories C LEFT JOIN CategoryTree CT
                        ON C.Category = CT.[ChildCategory]
                        LEFT JOIN CategoryTree CT2
                        ON CT.[ParentCategory] = CT2.[ChildCategory]
                ) CS ON I.ItemID = CS.ItemID
                WHERE Category3 LIKE @SearchCategory
        ORDER BY Title;
GO
```

2.3 Wyzwalacze

W bazie danych znajduje się dużo zależności klucza obcego, które uniemożliwiają proste usuwanie elementów. W tym celu powstały wyzwalacze, które przed usunięciem elementu usuwają wszystkie rekordy, które z niego dziedziczą. Są to: `DeleteBorrowing` do usuwania wypożyczeń, `DeleteUser` do usuwania użytkowników, `DeleteBook` do usuwania książek, `DeleteOther` do usuwania innych pozycji, i `DeleteItem` do usuwania pozycji z bazy.

```
CREATE TRIGGER DeleteBorrowing
ON Borrowings
INSTEAD OF DELETE
AS
```

```
05 |          DELETE FROM Fines
06 |          WHERE BorrowDate IN (SELECT BorrowDate FROM DELETED)
07 |          AND UserID IN (SELECT UserID FROM DELETED)
08 |          AND ItemID IN (SELECT ItemID FROM DELETED)
09 |
10 |          DELETE FROM Borrowings
11 |          WHERE BorrowDate IN (SELECT BorrowDate FROM DELETED)
12 |          AND UserID IN (SELECT UserID FROM DELETED)
13 |          AND ItemID IN (SELECT ItemID FROM DELETED)
14 |  GO
```

```
01 |  CREATE TRIGGER DeleteUser
02 |  ON Users
03 |  INSTEAD OF DELETE
04 |  AS
05 |          DELETE FROM Borrowings
06 |          WHERE UserID IN (SELECT UserID FROM DELETED)
07 |
08 |          DELETE FROM Users
09 |          WHERE UserID IN (SELECT UserID FROM DELETED)
10 |  GO
```

```
01 |  CREATE TRIGGER DeleteBook
02 |  ON Books
03 |  INSTEAD OF DELETE
04 |  AS
05 |          DELETE FROM Authors
06 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
07 |
08 |          DELETE FROM Redactors
09 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
10 |
11 |          DELETE FROM Categories
12 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
13 |
14 |          DELETE FROM Borrowings
15 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
16 |
17 |          DELETE FROM Books
18 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
19 |
20 |  GO
```

```
01 |  CREATE TRIGGER DeleteOther
02 |  ON Other
03 |  INSTEAD OF DELETE
04 |  AS
05 |          DELETE FROM Categories
06 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
07 |
```

```
08 |          DELETE FROM Borrowings
09 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
10 |
11 |          DELETE FROM Books
12 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
13 |
14 |  GO
```

```
01 |  CREATE TRIGGER DeleteItem
02 |  ON Items
03 |  INSTEAD OF DELETE
04 |  AS
05 |          DELETE FROM Books
06 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
07 |
08 |          DELETE FROM Other
09 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
10 |
11 |          DELETE FROM Items
12 |          WHERE ItemID IN (SELECT ItemID FROM DELETED)
13 |
14 |  GO
```

2.4 Skrypt tworzący bazę danych

Poniższy skrypt tworzy bazę danych (usuwając ją wcześniej jeżeli istnieje) oraz tworzy tabele.

```
01 |  IF EXISTS(select * from sys.databases where name='bib')
02 |          DROP DATABASE bib
03 |  CREATE DATABASE bib
04 |  USE bib;
05 |
06 |  DROP TABLE Fines, Borrowings, Categories, CategoryTree, Authors,
       Redactors, Books, Other, Users, Items
07 |
08 |
09 |  CREATE TABLE [Items]
10 |  (
11 |      ItemID       INT PRIMARY KEY,
12 |      Title        NVARCHAR(512) NOT NULL,
13 |      IssueYear    INT,
14 |      Edition      NVARCHAR(512)
15 |  );
16 |
17 |  CREATE TABLE Books
18 |  (
19 |      ItemID       INT PRIMARY KEY,
20 |      Publisher NVARCHAR(512),
21 |      PlaceOfIssue NVARCHAR(512),
22 |      ISBN NVARCHAR(512),
23 |          FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
```

```
24 |  );
25 |
26 |  CREATE TABLE Other
27 |  (
28 |      ItemID      INT PRIMARY KEY,
29 |      ProductionYear      INT,
30 |      Duration    INT,
31 |      FormatType  NVARCHAR(512),
32 |          FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
33 |  );
34 |
35 |  CREATE TABLE Authors
36 |  (
37 |      ItemID      INT,
38 |      AuthorName NVARCHAR(448),
39 |          PRIMARY KEY (ItemID, AuthorName),
40 |          FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
41 |  );
42 |
43 |  CREATE TABLE Redactors
44 |  (
45 |      ItemID      INT,
46 |      RedactorName NVARCHAR(448),
47 |          PRIMARY KEY (ItemID, RedactorName),
48 |          FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
49 |  );
50 |
51 |  CREATE TABLE Categories
52 |  (
53 |      ItemID      INT,
54 |      Category NVARCHAR(448),
55 |          PRIMARY KEY (ItemID, Category),
56 |          FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
57 |  );
58 |
59 |  CREATE TABLE CategoryTree
60 |  (
61 |      ChildCategory       NVARCHAR(225),
62 |      ParentCategory      NVARCHAR(225),
63 |          PRIMARY KEY (ChildCategory,ParentCategory)
64 |
65 |  );
66 |
67 |  CREATE TABLE Users(
68 |  UserID int PRIMARY KEY,
69 |  Name nvarchar(255) NOT NULL,
70 |  SignInDate date NOT NULL
71 |  )
72 |
73 |  CREATE TABLE Borrowings(
74 |  UserID int,
75 |  ItemID int,
76 |  BorrowDate date,
77 |  ReturnDate date,
78 |  PRIMARY KEY (UserID, ItemID, BorrowDate),
```

```
79 |  FOREIGN KEY (UserID) REFERENCES Users(UserID),
80 |  FOREIGN KEY (ItemID) REFERENCES Items(ItemID)
81 |  )
82 |
83 |  CREATE TABLE Fines(
84 |  UserID int,
85 |  ItemID int,
86 |  BorrowDate date,
87 |  Fine money,
88 |  PRIMARY KEY (UserID, ItemID, BorrowDate),
89 |  FOREIGN KEY (UserID, ItemID, BorrowDate) REFERENCES Borrowings(UserID,
         ItemID, BorrowDate)
90 |  )
```