Aflevering 1
- Forlaget Fanzy -

Jeg har oprettet min database og alle mine tabeller og rækker ved hjælp af sql-snippets i Azure Data Studio (ADS). Henholdsvis sqlCreateDatabase, sqlCreateTable, sqlInsertRows. Alle scripts og mit sqltabel-diagram ligger på :
https://github.com/jamtuba/Databaser_For_Udviklere

Databasen:

Jeg oprettede databasen (DB) 'FanzyPublish' med følgende script :

```
-- Create a new database called 'FanzyPublish'
-- Connect to the 'master' database to run this snippet
USE master
GO
-- Create the new database if it does not exist already
IF NOT EXISTS (
    SELECT [name]
        FROM sys.databases
        WHERE [name] = N'FanzyPublish'
)
CREATE DATABASE FanzyPublish
GO
```

Hvor jeg først sørger for at bruge 'master' DB og så tjekker om min nye DB findes og hvis ikke, opretter den.

Forfattertabel:

Denne tabel er ligeledes lavet ved hjælp af en snippet:

```
-- Create a new table called '[Author]' in schema '[dbo]'
-- Drop the table if it already exists
IF OBJECT_ID('[dbo].[Author]', 'U') IS NOT NULL
DROP TABLE [dbo].[Author]
GO
-- Create the table in the specified schema
CREATE TABLE [dbo].[Author]
(
    [Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY, -- Primary Key column
    [FirstName] NVARCHAR(50) NOT NULL,
    [MiddleName] NVARCHAR(50),
    [LastName] NVARCHAR(50) NOT NULL,
    [EmailAddress] NVARCHAR(100) UNIQUE NOT NULL
);
GO
```

Der tjekkes først om tabellen findes og hvis den findes, slettes (dropTable) den. Jeg har valgt at lade DB oprette Id som int (heltal), der også er Primary Key, ved at bruge IDENTITY (1,1). Så starter den ved Id = 1 og forhøjer til næste Id med en. Jeg har lavet fornavn og efternavn samt emailadresse obligatoriske (altså NOT NULL) og som NvarChar type (string). Mellemnavn er valgfrit, mens emailadressen skal være unik.

Bogtabel:

```
-- Create a new table called '[Book]' in schema '[dbo]'
-- Drop the table if it already exists
IF OBJECT_ID('[dbo].[Book]', 'U') IS NOT NULL
DROP TABLE [dbo].[Book]
GO
-- Create the table in the specified schema
CREATE TABLE [dbo].[Book]
(
    [Id] INT IDENTITY (1,1) NOT NULL PRIMARY KEY, -- Primary Key column
    [Title] NVARCHAR(100) NOT NULL,
    [Summary] NVARCHAR(300) NOT NULL,
    [Price] DECIMAL (6,2),
    [ISBN] NVARCHAR(17) UNIQUE NOT NULL -- Including the hyphens for
ISBN-13
);
GO
```

Der tjekkes om tabellen findes og hvis den gør slettes den. Min bogtabel bruger igen en int og Identity som PK. Der er en titel og et sammendrag som nvarchar. En pris som decimal med 6 tals nøjagtighed og 2 tal efter kommaet (højre side af .). ISBN fik jeg researchet frem til højst kan blive på 17 karakterer inklusiv bindestregerne (ISBN13), så nvarchar med 17 karakterer og den skal være unik.

Kategoritabel:

```
-- Create a new table called '[Category]' in schema '[dbo]'
-- Drop the table if it already exists
IF OBJECT_ID('[dbo].[Category]', 'U') IS NOT NULL
DROP TABLE [dbo].[Category]
GO
-- Create the table in the specified schema
CREATE TABLE [dbo].[Category]
(
    [Id] INT NOT NULL IDENTITY (1,1) PRIMARY KEY, -- Primary Key column
    [CategoryName] NVARCHAR(50) NOT NULL
);
GO
```

Tabel findes tjek... Igen med en int PK. Her har jeg kun kategorinavne i min tabel (Og naturligvis et Id!).

Junctiontabeller:

For at få styr på udfordringen med at en bog kan have flere forfattere og flere forfattere kan have skrevet flere bøger, samt at en bog kan have flere kategorier og hver kategori kan have flere bøger tilknyttet har jeg lavet 2 junction-tabeller. Her er der krydsreferencer de andre tabeller imellem og det er nødvendigt når man har Many-To-Many relationer. Og det gør at vi undgår redundant data, altså gentagne data, i de andre tabeller.

AuthorBookRelation

```
-- Create a new table called '[AuthorBookRelation]' in schema '[dbo]'
-- Drop the table if it already exists
IF OBJECT_ID('[dbo].[AuthorBookRelation]', 'U') IS NOT NULL
DROP TABLE [dbo].[AuthorBookRelation]
GO
-- Create the table in the specified schema
CREATE TABLE [dbo].[AuthorBookRelation]
(
    [AuthorId] INT NOT NULL,
    [BookId]INT NOT NULL,
    PRIMARY KEY(AuthorId, BookId),
    FOREIGN KEY (AuthorId) REFERENCES Author (Id),
    FOREIGN KEY (BookId) REFERENCES Book (Id)
);
GO
```

BookCategoryRelation

```
-- Create a new table called '[BookCategoryRelation]' in schema '[dbo]'
-- Drop the table if it already exists
IF OBJECT_ID('[dbo].[BookCategoryRelation]', 'U') IS NOT NULL
DROP TABLE [dbo].[BookCategoryRelation]
GO
-- Create the table in the specified schema
CREATE TABLE [dbo].[BookCategoryRelation]
(
    [BookId]INT NOT NULL,
    [CategoryId] INT NOT NULL,
    PRIMARY KEY(BookId, CategoryId),
    FOREIGN KEY (BookId) REFERENCES Book (Id),
    FOREIGN KEY (CategoryId) REFERENCES Category (Id)
);
GO
```

Fælles for de 2 tabeller er at de opbevarer Id for de 2 entiteter der skal relatere til hinanden. Man angiver en Foreign Key, altså en henvisning til at Id er hentet i en anden tabel. Så opretter man en sammensat (Composite) Primary Key af de 2 Id´er.

Indsæt forfattere:

Fælles for de næste scripts er at de indsætter data i rækker i forskellige tabeller. Jeg har i alle tilfælde brugt sql-snippets i ADS. Man angiver tabellen der skal indsættes i, kolonnerne der skal indsættes i og værdierne. Her i forfatter tabellens MiddleName har jeg eksperimenteret lidt med hvad forskellen på NULL og en tom streng ville give for resultat.

```
-- Insert rows into table 'Author' in schema '[dbo]'
INSERT INTO [dbo].[Author]
( -- Columns to insert data into
 [FirstName], MiddleName, LastName, EmailAddress
)
VALUES
(
    'Jussi', NULL, 'Adler-Olsen', 'Adler@Olsen.dk'
),
(
    'Stephen', NULL, 'King', 'TheKing@Horror.com'
),
(
    'Svend', 'Åge', 'Madsen', 'svendaamadsen@mail.dk'
),
(
    'Vladimir', '', 'Khorikov', 'unit@testing.org'
),
(
    'Andrew', NULL, 'Troelsen', 'a@troelsen.com'
),
(
    'Philip', NULL, 'Japikse', 'japikse@gmail.com'
),
(
    'Joseph', '', 'Albahari', 'bigbrother@oreilly.com'
),
(
    'Ben', '', 'Albahari', 'littlebrother@oreilly.com'
),
(
    'Keith', '', 'Kellenberger','keller@hotmail.com'
),
(
    'Lee', 'mr T-SQL', 'Everest', 'sql@apress.com'
)

GO
```

Indsæt bøger:

Her kunne jeg ikke finde priser på alle bøger så der står NULL ved nogen af dem. Det kan man så få nogle spændende queries ud af....

```
-- Insert rows into table 'Book' in schema '[dbo]'
INSERT INTO [dbo].[Book]
( -- Columns to insert data into
    Title, Summary, Price, ISBN
)
VALUES
(
    'Beginning T-SQL', 'A Step-By-Step Approach', NULL, '978-1-4842-
6605-2'
),
(
    'C# 7.0', 'In a Nutshell', 79.99, '978-1-491-98765-0'
),
(
    'Pro C# 7', 'With .NET and .NET Core', 59.99, '978-1-4842-3017-6'
),
(
    'Unit Testing', 'Principles, Practices, and Patterns', 49.99, '978-
1-61729-627-1'
),
(
    'Fremtidsspejl', 'Spændende bog om en kronofysiker der forelsker sig
i 2 piger på samme tid', NULL, '978-87-02-30420-6'
),
(
    'Mange sære ting for', 'Forrygende filosofisk spændingsroman om
religion, sandhed og kampen for at få fat i den rette fortolkning',
NULL, '978-87-02-08254-8'
),
(
    'The Institute', 'Spændings roman', NULL, '978-1-529-35541-3'
),
(
    'Blodtørst', 'Og andre fortællinger', NULL, '978-87-400-6521-3'
),
(
    'Natrium Chlorid', 'Spændingsroman af en af Danmarks bedste
krimiforfattere', NULL, '978-87-400-451-30'
)
GO
```

Indsæt kategorier:

```
-- Insert rows into table 'Category' in schema '[dbo]'
INSERT INTO [dbo].[Category]
( -- Columns to insert data into CategoryName
    CategoryName
)
VALUES
(
    'Computer book'
),
```

```
(
    'Krimi'
),
(
    'C#'
),
(
    'T-SQL'
),
(
    'Test'
),
(
    'Skønlitteratur'
),
(
    'Teknologi'
)
GO
```

Indsæt Forfatter/Bog relationer:

Lavet med Id fra Author og Id fra Book tabellerne.

```
-- Insert rows into table 'AuthorBookRelation' in schema '[dbo]'
INSERT INTO [dbo].[AuthorBookRelation]
( -- Columns to insert data into
    AuthorId, BookId
)
VALUES
(
    1, 9
),
(
    2, 7
),
(
    2, 8
),
(
    3, 5
),
(
    3, 6
),
(
    4, 4
),
(
    5, 3
),
(
    6, 3
```

```
),
(
    7, 2
),
(
    8, 2
),
(
    9, 1
),
(
    10, 1
)
GO
```

Indsæt Bog/Kategori relationer:

Lavet med Id fra Book og Id fra Category tabellerne.

```
-- Insert rows into table 'BookCategoryRelation' in schema '[dbo]'
INSERT INTO [dbo].[BookCategoryRelation]
( -- Columns to insert data into
    [BookId], CategoryId
)
VALUES
(
    1, 1
),
(
    1, 4
),
(
    1, 7
),
(
    2, 1
),
(
    2, 3
),
(
    2, 7
),
(
    3, 1
),
(
    3, 3
),
(
    3, 7
),
(
    4, 1
```

```
),
(
    4, 5
),
(
    4, 7
),
(
    5, 6
),
(
    5, 7
),
(
    6, 6
),
(
    6, 7
),
(
    7, 2
),
(
    7, 6
),
(
    8, 6
),
(
    9, 2
),
(
    9, 6
)
GO
```

Queries: titel, kategori, forfatter(-e), ISBN

Henter alle bøger, med kategorier, forfatternavne og ISBN. Jeg har tryllet lidt med streng-manipulation for at mellemnavne kunne udelades hvis de er NULL eller en tom streng. Jeg joiner alle mine tabeller, som jeg har givet aliaser, for at få alle detaljer tilgængelige i min forespørgsel. Hvis man ville have haft færre informationer med kunne man have udeladt nogle mine joins, som alle er INNER JOINS der kun medtager søgninger med der matcher på id.

```
SELECT Title, c.CategoryName, a.FirstName + ISNULL(NULLIF(' ' +
MiddleName,' '), '') + ' ' + LastName AS AuthorName, ISBN
FROM Book AS b
JOIN AuthorBookRelation AS ab ON b.Id = ab.BookId
JOIN Author AS a ON ab.AuthorId = a.Id
JOIN BookCategoryRelation AS bc ON bc.BookId = b.Id
JOIN Category AS c ON bc.CategoryId = c.Id
ORDER BY Title;
```

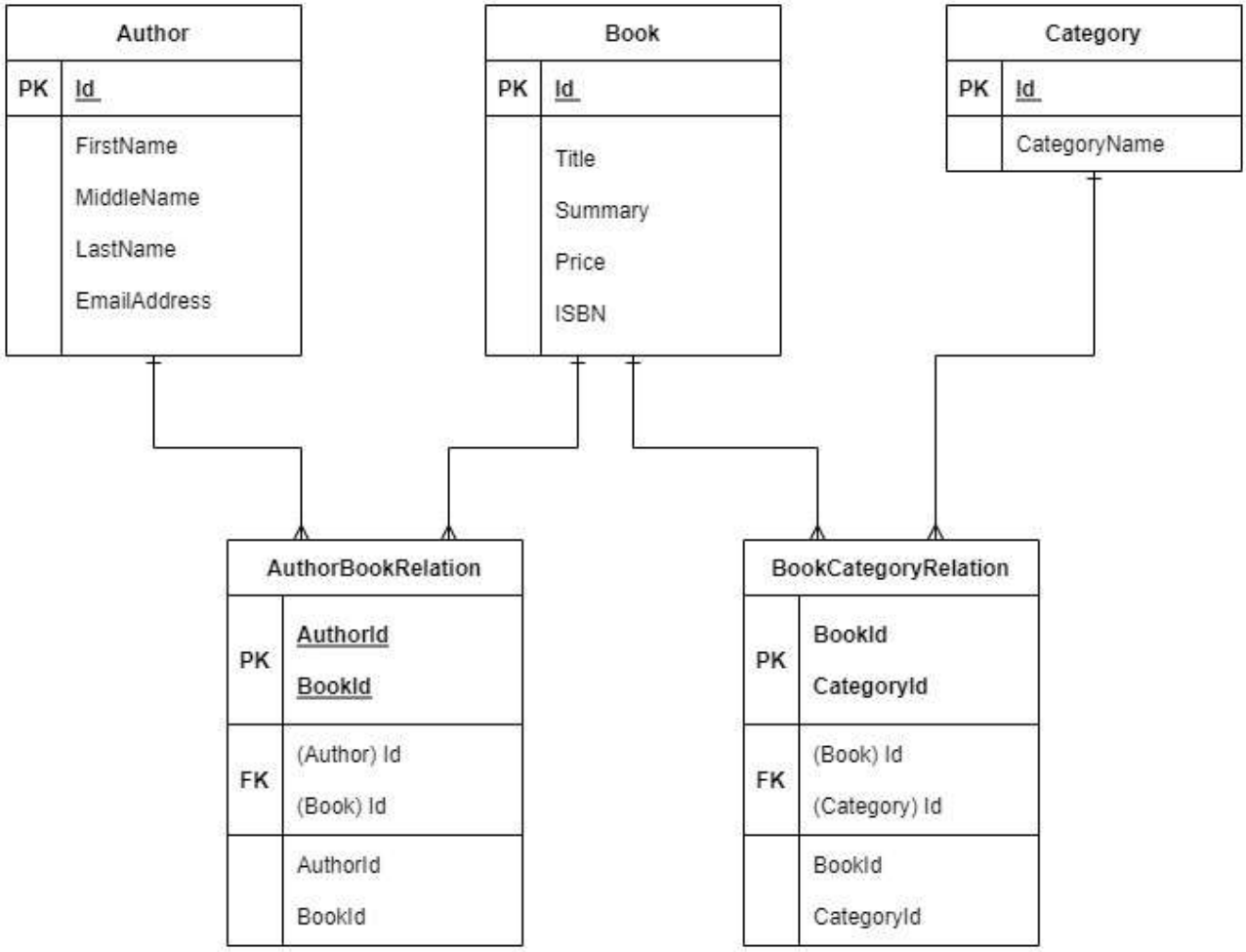
Queries: specifik bog med alle oplysninger.

Igen joiner jeg alle 5 tabeller og til sidst bruger jeg en where-clause der specificerer at titlen på bogen skal være 'Pro C# 7'.

```
SELECT Title, CategoryName, a.FirstName + ISNULL(NULLIF(' ' +
MiddleName,' '), '') + ' ' + LastName AS AuthorName, ISBN, Summary,
Price
FROM Book AS b
JOIN BookCategoryRelation AS bc ON b.Id = bc.BookId
JOIN Category AS c ON bc.CategoryId = c.Id
JOIN AuthorBookRelation AS ab ON b.Id = ab.BookId
JOIN Author AS a ON a.Id = ab.AuthorId
WHERE b.Title = 'Pro C# 7';
```

Diagram over tabeller:

Til sidst er der her et diagram over mine tabeller. En forfatter kan have mange forfatterbog relationer, men omvendt er der kun en forfatter i hver forfatterbog relation. Det samme princip er gældende for bog/forfatterbog, bog/bogkategori og kategori/bogkategori relationerne.



Konklusion:

Jeg er muligvis gået lidt ud over hvad vi har lært indtil nu på kurset, men kunne ikke se hvordan tabellerne ellers skulle være nogenlunde normaliserede. Der er givetvis andre smartere måder at udføre queries på, så det arbejder jeg på, på resten af kurset. Jeg har kun lavet 2 queries, som jeg har forstået opgavestillingen. Man kunne fint have lavet flere på forskellige leder og kanter.

Fremstillet af

Janus Mogensen