**Dokumentation for Normalisering af min Database - Opgave 18**

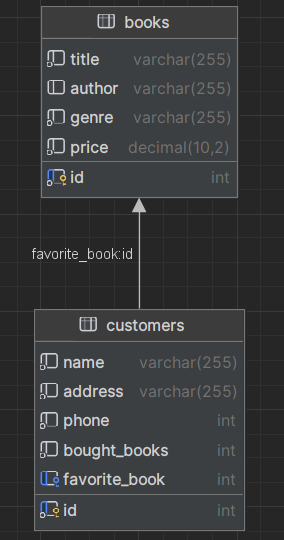
**Formål:**

At normalisere en tabel med bogdata for at reducere redundans, forbedre dataintegritet og skabe et mere effektivt databasesystem.

**Før Normalisering:**

Vi starter med en tabel, der indeholder følgende kolonner: ID, Titel, Forfatter, Genre, og Pris. Tabellen er ikke normaliseret, og følgende problemer er til stede:

* **Redundant data**: Flere bogtitler, forfatternavne og genre gentager sig, hvilket øger risikoen for datainkonsistens.
* **Dårlig dataintegritet**: Hvis en forfatters navn eller en genres navn ændres ét sted, skal det ændres på tværs af alle forekomster i tabellen, hvilket gør vedligeholdelsen sværere.
* **Manglende relationsstruktur**: Alle data er opbevaret i én tabel, hvilket gør det sværere at vedligeholde og udvide databasen.

****

**Mål med normalisering:**

Normalisering af data i flere mindre tabeller reducerer datadublikation og sikrer dataintegritet ved at etablere klare relationer mellem enhederne (bøger, forfattere og genrer).

**Trin for Trin Normalisering**

**1. Første Normalform (1NF)**

**Regel**: Ingen gentagne grupper eller flere værdier i én kolonne. Alle kolonner skal indeholde atomiske værdier.

**Handling**: Tabellen er allerede i 1NF, da der ikke er nogen gentagne grupper i felterne, og værdierne er atomiske (én værdi per felt).

**2. Anden Normalform (2NF)**

**Regel**: Alle ikke-nøgle attributter skal være fuldstændigt afhængige af primærnøglen.

**Problem**: I den oprindelige tabel gentager forfatternavne og genre sig for flere bøger. Dette betyder, at kolonnerne Forfatter og Genre ikke afhænger direkte af en bogs ID, men snarere af de enkelte forfattere og genrer. Vi har derfor partielle afhængigheder, som vi skal eliminere.

**Løsning**:

* Opret en separat tabel til **forfattere** (persons), hvor vi kun gemmer hver forfatter én gang.
* Opret en separat tabel til **genrer** (genres), hvor vi kun gemmer hver genre én gang.
* Opret en relation mellem books-tabellen og de nye tabeller persons og genres ved hjælp af fremmednøgler.

**3. Tredje Normalform (3NF)**

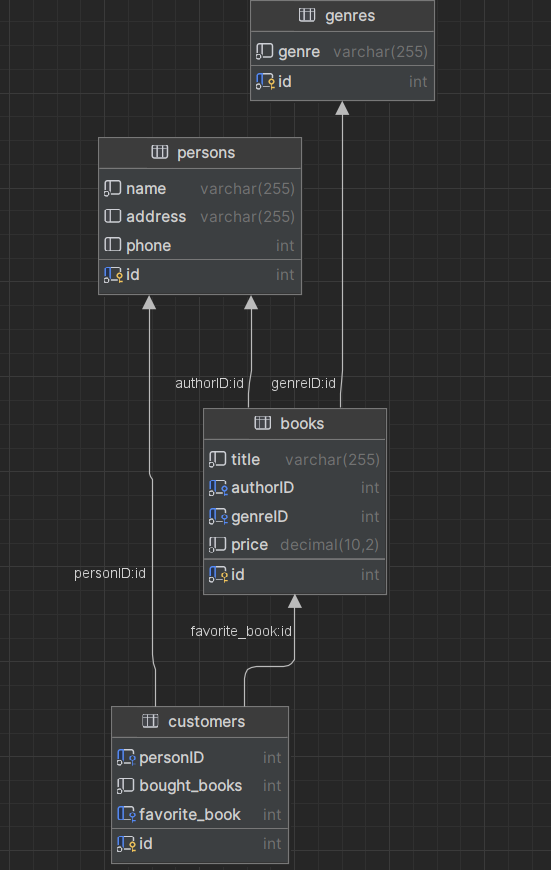
**Regel**: Ingen transitiv afhængighed (attributter må ikke være afhængige af andre ikke-nøgle attributter).

**Problem**: Den oprindelige tabel har transitive afhængigheder, da kolonner som Forfatter og Genre ikke er direkte afhængige af bogen selv, men snarere af forfatteren og genren.

**Løsning**: Ved at opdele data i separate tabeller og oprette relationer via fremmednøgler fjerner vi transitive afhængigheder. Forfatterens og genreens oplysninger er nu kun gemt i deres respektive tabeller og ikke i books-tabellen.

**Den Normaliserede Database Struktur**

**1. persons (Forfattere)**

* **Kolonner**: personID (Primær nøgle), name, address, phone
* **Beskrivelse**: Indeholder unik information om forfattere. personID bruges som fremmednøgle i books-tabellen.

**2. genres (Genrer)**

* **Kolonner**: genreID (Primær nøgle), genre
* **Beskrivelse**: Indeholder unikke genrer, der refereres til fra books-tabellen.

**3. books (Bøger)**

* **Kolonner**: bookID (Primær nøgle), title, authorID (Fremmednøgle til persons), genreID (Fremmednøgle til genres), price
* **Beskrivelse**: Indeholder bogspecifikke oplysninger, mens forfatteren og genren er knyttet via fremmednøgler.

**4. customers (Kunder)**

* **Kolonner**: customerID (Primær nøgle), personID (Fremmednøgle til persons), bought\_books, favorite\_book (Fremmednøgle til books)
* **Beskrivelse**: Indeholder oplysninger om kunder, hvor personID refererer til en forfatter i persons, hvis en forfatter også er en kunde.

**Fordele ved Normalisering:**

1. **Reduceret Data Redundans**: Oplysninger om forfattere og genrer opbevares kun én gang i separate tabeller, hvilket reducerer datadublikation.
2. **Forbedret Dataintegritet**: Ændringer i forfatter- eller genreoplysninger kan foretages ét sted og afspejles i hele databasen, hvilket reducerer risikoen for fejl og inkonsistens.
3. **Effektivitet i Spørgsmål og Vedligeholdelse**: Ved at opdele data i relaterede tabeller kan forespørgsler udføres mere effektivt, og vedligeholdelsen af databasen bliver nemmere og mere organiseret.
4. **Skalerbarhed**: Ved at opdele data kan vi nemt tilføje nye forfattere, bøger eller genrer uden at skulle foretage større ændringer i strukturen.
5. **Tydelige Relationer**: Fremmednøgler mellem tabellerne sikrer klare relationer og gør det nemmere at skabe avancerede forespørgsler (f.eks. for at finde alle bøger af en forfatter eller genre).

**Konklusion:**

Ved at normalisere tabellen opnår vi en mere struktureret database, hvor dataintegritet opretholdes, og redundans reduceres betydeligt. Normaliseringen gør det nemmere at vedligeholde og udvide databasen, samtidig med at den forbedrer ydeevnen for komplekse forespørgsler.