Skriven av Khaled Zraiqi

Rapport för inlämningsuppgift 2

Introduktion

Arbetet är utfört individuelt och handlar egentligen om att lära sig sig hur man bygger en normaliserad databas som går och utveckla vidare utan problem. Databasen kopplas vidare med C# med hjälp utav verktyget Dapper. Databasen används av ett enkelt prototyp program som hjälper testa databasen.

Bakgrund/Planeringen

Projektet handlar om att bygga ett smart och effektiv databas som hjälper ett helt biblioteks kedja lagra data i en databas. Projektet löser Biblioteket krav som:

- kunna lagra data om böcker, kunder, boklån, bibliotek inom kedjan och bibliotekarie.
- Kunder ska kunna ha ett enkelt sätt och logga in samt spara generell information om kunden
- Böcker ska lagras med komplett information samt om boken är tillgänglig och vilket sorts media och biblotek boken tillhör. Bokens författare ska oxå lagras på ett normaliserad sätt.
- Bibloiotekarier har ett admin inlogg och vart dom jobbar ska lagras samt generell information om personen.
- Kunder ska kunna fråga efter ett kvitto så det är viktigt att alla lån lagras med komplett information om boklånet.

Verktyg

Entity relation database för att plannera och bygga ett databas model.

C#/.Net för att prototypa databasen.

Dapper för att koppla ihop C# med databasen.

phpmyadmin/mariaDb för att skapa tabeller och en aktuell databas som sparar data och följer ERD modellen.

SOLID/OOP för att hjälpa strukturera C# koden och följa vissa standard som hjälper kodens läsbarhet/vidareutveckling.

Resultat

Efter en del plannering och olika versioner utav databasen så blev resultatet (Se bilaga A). Modellen representerar hela biblioteks kedjan och är fullt normaliserat. Modellen följer en strict snakecase vilket betyder att alla kolumner samt tabellnamn är små bokstäver. Alla mellanslag skrivs istället med en “_”.

För att beskriva kopplingarna kortfattad nedan.
customer och loan ett koppling där en kund kan ha flera lån samtidigt vilket skapar en till många relation.

book och loan har en till många relation. flera böcker ska kunna lånas ut samt som en bok inte kan lånas ut många gånger samtidigt.

book och media har en och endast en till många relation. bok har endast en mediatyp och mediatyper finns i flera böcker.

book och library har en till många relation. boken ägs utav ett bibliotek fast inte utav många bibliotek samtidigt

book och author har många till många relation därför använder vi kopplingstabellen author_to_book för att koppla ihop båda tabeller i ett normaliserat sätt. author kan skriva många books och en book kan vara skriven utav flera authors.

library och librarian har en till många relation. en librarian jobbar i en library och ett library har många librarians.

Interessanta problem

Dom mest interessanta problem uppstog mest under planneringsfasen. Frågan var hur löser man ett sätt för kunder och kunna enkelt låna en bok? Biblioteket ska också i spara gamla lån för statistik eller ge kund ett historik på vad dom har lånat innan. Det enklaste lösningen blev att om vi lagrar böcker med om dom är tillgängliga direkt så vet vi fort om boken är ledig eller inte. Andra steget är och lagra information i lånet om när började lånet, när ska den tillbaka och sedan när lämnaden boken tillbaka. Detta tre datum ger oss allt information vi behöver om lånet. För att kontra oberäknad problem som om boken blev aldrig returnerad kan vi enkelt lagra ett “ja/nej” i lån tabellen som visar om t.ex boken returnerades eller aldrig.

Ett annat kanske lite mindre problem var hur kan vi enkelt skapa ett system för customers inlogg. Det smidigaste blev att om customers får deras id som användarnamn och sedan får dom välja ett 4 siffrors pin code. Detta gjorde det enkelt för kunden att inte behöva hålla på mycket på den tekniska sidan.

Vidareutveckling

Hela projektet gick bra men antagligen finns det alltid några eller något del som går att utveckla. Det viktigaste och något jag hade vidareutvecklat är hur kunder lånar böcker. Just nu så är det ett “först till kvarn” system där kunder behöver snabbt låna en bok så fort den är tillgänglig. Jag hade villat implementera ett sätt för kunder att “köa” eller helt enkelt lägga en sorts request/order på en bok som inte är tillgänglig.

För att plannera ett sånt lösning gäller det att vi börjar med att planera hur implementationen utav detta hade sett ut.

Om vi plannerar hur vill vi koppla detta med book. Direkt kan vi känna av att en book kan ha många customers som väntar. sedan så ska kunder inte låsa sig till en och vänta på endast en bok samtidigt.

Nu har vi bestämt att vi har en många till många relation mellan customer och book. Vi kommer behöva skapa ett customer_to_book tabel som egentligen är våran waiting list och fungerar i tur och ordning.
Som vi ser implementation utav extra funktionalitet i databasen är inte svår och vissa problem kan enkelt lösas i ett normaliserat databas.

Bilaga

Referensbild (A) Kopplingsdiagram