# Mapping

Af: Lasse

Vi har været fælles i vores gruppe omkring, at lave mapping til vores database. Der er mange faktorer vi skulle tage højde for, og målet var at ramme det rigtige resultat første gang, så vi ikke senere skal ind og foretage ændringer i databasen. Grunden til vi vælger at bruge mapping er, at den giver os et overblik over designet, alternativet var at lave vores queries direkte. Med mapping får vi overblik over alle vores tabeller, primarykey og foreignkeys, og hvordan vores queries skal oprettes. Vi kan fx ikke oprette Finanskonti tabellen, før vi har oprette Finansgruppe tabellen, da der er en foreign key i Finanskonti som peger på Finansgrupper.

I vores mapping har vi 5 tabeller, hvoraf 1 er en samlingstabel mellem finanskonti og perioder.   
Vi har valgt at bruge en surrogatnøgle i Budget tabellen, da vi som tidligere nævnt, ikke har nogle entiteter vi kan bruge som primary key.   
Finanskonti har en foreign key fra Navn på finansgrupper, dette er nødvendigt for at få koblet de to tabeller sammen, så vi har mulighed for at se hvilken gruppe en konto hører under. Det samme gør sig gældende for id fra Budget, hvor vi også har en foreign key. Derudover har vi en composite key, der er sat sammen af KontoId og FK\_BudgetId. Vi har valgt dette, da KontoId i sig selv ikke er unik, da det er et tal vi henter ud fra Xena, og flere forskellige regnskaber kan indeholde det samme KontoId. På denne måde får vi skabt en unik nøgle der samtidig er koblet på et budget.  
Vores Perioder tabel indeholder som vores budget tabel også en surrogatnøgle, da vi ikke har en værdi vi ellers kan bruge som primary key.  
Den sidste tabel er vores FinansKontiPeriode, som er en samlingstabel mellem Finanskonti og Perioder. Denne tabel bruger vi til at opbevare vores estimater til budgetterne. Det estimat bliver så koblet sammen med en periode og en finanskonti via 2 foreign keys. Vi har været nødt til at lave en samlingstabel, da det skal være muligt, at give estimater på en Finanskonti på flere forskellige perioder, dette kun være for måneder, halv årligt, hel årligt osv.

## Normaliseringsregler

Da vi lavede vores mapping, har vi samtidig haft fokus på de tre første normaliseringsregler, dette har vi gjort for at gøre databasen mere fleksibel og fjerne overflødige data. Det kunne fx være at have den samme data gemt i forskellige kolonner, ved at følge reglerne har vi kun data gemt et sted, og skal ikke rette data i flere forskellige tabeller.

### Første normalform

* Fjern gentagende grupper i individuelle tabeller.
* Opret en særskilt tabel for hver sæt af relateret data.
* Identificer hver sæt af relateret data med en primærnøgle

Vi har opfyldt de tre punkter i den første normalform, vi mener at den data der er relevant for en tabel, er placeret i den pågældende tabel. Vi har ingen gentagende grupper i individuelle tabeller, alt vores data skal kun rettes et sted. Vi har også en primær nøgle til hver sæt af relateret data i de tabeller, hvor det er relevant.

### Anden normalform

* Opret særskilte tabeller til værdisæt, som kan anvendes til flere poster.
* Relater disse med en fremmednøgle.

Vi har oprettet vores mapping efter, at der skal være særskilte værdisæt i alle tabeller, som kan anvendes til flere poster. Vi har igen ingen data der er duplikeret og som skal ændres i flere tabeller ved hver rettelse.

### Tredje normalform

* Fjern de felter der ikke afhænger af nøglen

Vi har også fjernet de felter, som ikke er nødvendige for definition af en tabel, et eksempel på dette er Finansgruppe navnet hvor vi har oprettet en særskilt tabel, da det skal være muligt at trække en komplet liste over alle finansgrupper, selvom de ikke nødvendigvis er blevet brugt af en finanskonto. Man kunne også argumentere for, at år kolonnen i Budget tabellen, kunne flyttes ud i sin egen tabel. I vores tilfælde mener vi dog, at år er relevant for at definere et budget, og på den baggrund hører til Budget tabellen.