**DanielsDagbok**

Börjat med att utgå utifrån dom entiteterna vi fått.  
Lagt till dom attributen som jag tycker är relevanta utifrån en bankkund databas.

Har tänkt utifrån min erfarenhet jag har haft med banken och tänkt hur den databasen skulle kunnat se ut och försökt efterlikna den på en simpel nivå.

Börjat med plocka ut substantiv som används som entiteter.

Jag tänker att en kund kan ha fler kontaktpersoner och kontaktpersoner kan ha många kunder. Ex om sin kontaktperson har semester och man får hjälp av en annan.  
En kontaktperson måste även ha en unik mailadress så man inte kan blanda ihop med någon annan. Däremot kan flera kontaktpersoner ha samma telefonnummer pga ifall någon slutar eller blir föräldraledig eller semester så kan man ta över den andras telefonnummer.

Jag tänker att varje kund har ett bankID kopplat till sig och att man får välja ett lösenord som man kan logga in med. Ett BankID kan endast vara kopplat till en kund.   
Därav Unique i BankID tabellen på CustomerID. Har gjort så för att att flera kunder inte ska kunna ha samma bankID.

Tabellen Adress är kopplat till både kund och anställd för minskad redundans.  
Man kan endast lägga in en adress en gång så de inte blir dubbeldata i databasen.  
Därav UNIQUE på alla attributer tillsammans i den tabellen.

En kund blir tilldelad ett kontor på engång när man blir kund på banken. Detta är för att en kund MÅSTE vara kopplat till ett kontor.  
Kontoren är kopplad till en Region för att se vart kontoret ligger. Har flyttat ut region till en egen entitet för att minska data i Office tabellen. En liten INT tar mindre plats en NVARCHAR(50). Dock kräver detta en extra JOIN ifall man vill veta namnet på regionen.

Telefonnumret till ett kontor kan vara samma som till andra, tänker ifall man väljer att ha en växel tillexempel.

En kund kan skapas utan att ha varken konto eller kort.   
Det läggs till i efterhand när man bestämt vad för konto man vill ha etc.  
Detta gör det mer flexibelt och banken. En bank kan registrera potentiella kunder innan kunden bestämt vad för konto den vill ha. Det skapar en möjlighet för banken att bygga en relation med kunden och ge personlig service och rådgivning om vilka produkter som är bäst för kunden.

Banken kan även marknadsföra till dessa kunder för att uppmuntra kunden att använda fler av bankens tjänster.

En kund kan ha flera konton och även gemensamma konton med andra kunder,  
ett konto kan skapas utan att kopplas till ett kort. Därav är CardID nullable i Dispositions tabellen. Detta ger möjligheten att ha ett konto som inte har något kort kopplat till sig, men vill man ha ett kort kopplat till det konto så går det.

Ett kort är alltid kopplat till ett konto därav Not Null i CustomerID och AccountID i Dispositions tabellen. Detta är så att man kan se sitt saldo genom Account tabellen.   
Minskad redundans. Slipper ha Saldo i både Card och Account.

Jag har valt att göra 2 olika entiteter för kort och konto transaktioner.   
Detta är för att lättare och effektivare kunna söka upp tillexempel endast kort transaktioner under en viss period, så slipper databasen gå igenom alla kontotransaktioner också. Om en typ av transaktioner är vanligare än den andra kan det minska söktider och indexstorlek eftersom databasen kan optimera lagring och åtkomst.

Jag har valt att koppla lån till konto för att man måste ha ett konto för att få ta ett lån.  
Kunder kan ha flera konton och olika typer av lån kopplade till varje konto.

Genom att associera lån med specifika konton, kan man hantera denna komplexitet på ett strukturerat sätt.  
Lånebetalningar och amorteringar kan enkelt dras från det kopplade kontot.

Jag har valt att använda NVARCHAR på alla min datatyper för att inkludera unikode så att man slipper det problemet i framtiden.

Jag har valt NVARCHAR(MAX) på lösenord så att man kan slänga in en lång hash och man kanske inte alltid vet hur lång den hashen kommer vara.