Normalisering

Formålet med normalisering, er at rettelser i databasen skal udføres med med mindst mulig indflydelse på systemet. Man vil gerne undgå at gemme den samme data flere steder, så en evt. ændring i databasen ikke kræver ændringer mange steder.

Den redundante data har en effekt på performance og maintainability, da man bruger unødvendige ressourcer og opdatering skal ske flere steder.

Der er mange normalformer, dog normaliseres der kun til og med 3. normalform. Det skyldes at 4. og 5. normalform, for at det meste er opfyldte efter de første 3, og det kan være specifikt for domænet om det er nødvendigt om de efterfølgende bør være opfyldt.

NF	Beskrivelse	Huskeregel
1NF	Hvis der er repeterende felter, tages de over i deres egen tabel med foreginkey til den oprindelige tabel. Her er det påkrævet, at der eksisterer en primærnøgle. Hvis denne nøgle har ens attributter(pris1 og pris2), så skal de kolonner fjernes fra tabellen, og lave deres egen tabel med primærnøglens id	Ingen repeterende felter
2NF	1NF skal være overholdt. Hvis tabellen indeholder en sammensat primærnøgle, skal de alle resterende felter være funktionelt afhængige af primærnøglen. Det vil sige, at de felter der ikke indgår i nøglen, laver en ny tabel sammen med en del fra den sammenhængende nøgle. Denne delnøgle blvier derfor primærnøglen for den nye tabel.	Alle felter skal være funktionelt afhængig af primærnøgle
3NF	Skal opfylde 2. normalform. Der må ikke findes felter uden for primærnøglen, som er indbyrdes afhængige. Et klassisk eksempel medlemstabellen, hvor by og postnummer indgår. Da by er afhængige af postnr, men postnr ikke indgår i primærnøglen, så fjernes by feltet og laver en ny tabel med postnr som primærnøglen.	Postnummer
4NF	Skal opfylde 3. normalform. Et klassisk eksempel på 4. normalform er pizzaeksemplet. Et pizzaria leverer flere typer pizzaer, fx tyk og tynd bund. Pizzariaet lever til flere byer. Hvis pizzeriaet leverer alle typer pizza til alle områder hvor de leverer, er tabellen til venstre en fin løsning. Hvis de leverer bestemte typer pizzaer til bestemte områder skal tabellen deles op i de to mindre til højre.	Pizza

Restaurant	Pizzabund	Levering
Pizzahut	Tynd bund	Århus
Pizzahut	Tyk bund	Århus
Pizzahut	Tynd bund	Skive
Pizzahut	Tyk bund	Skive

Restaurant	Pizzabund
Pizzahut	Tynd bund
Pizzahut	Tyk bund

Restaurant	Levering
Pizzahut	Århus
Pizzahut	Skive

Funktionelle afhængigheder

Funktionelle afhængigheder beskriver afhængighederne to sæt attributter i en relationel database. Dette kan bruges som værktøj til at primærnøgler og til at normalisere sin database. Et eksempel på en FD kunne være:

FD_1: Medarbejdernummer -> medarbejdernavn
(medarbejdernummer udpeger et bestemt medarbejdernavn)

Objektorientering VS Relationelle databaser

Data i relationelle databaser ligner meget den opbygning der findes i Objektorienteret og det betyder at hvis vi sammenligner dem step for step, vil vi se at det nærmest blot er navne der ændre sig:

ObjektOrientering	**Relationel Database**
Klasser	Entiteter
Data i attribut i objekter	Data i attribut i tabeller
Ændre data via metoder: get(), set()	Ændre data via DML(data Manipulation Language) CRUD (Create, Read, Update, Delete)
Relationer dannes mellem klasser med pointere og referencer	Relationer mellem entiteter dannes på baggrund af foreignkeys
Beskrives med et Klassediagram	Beskrives med et ER Diagram

I en relationel model er data gemt i tabeller, hvor hver tabel har kolonner og rækker. Fremmednøgler anvendes til at til at lave relationer mellem andre tabeller. Heri findes der forskellige typer relationer: Many-to-many, one-many, one-to one og n-ary. Der regler hvorledes man overholder integriteten af data – bl.a. vha. normaliseringen.

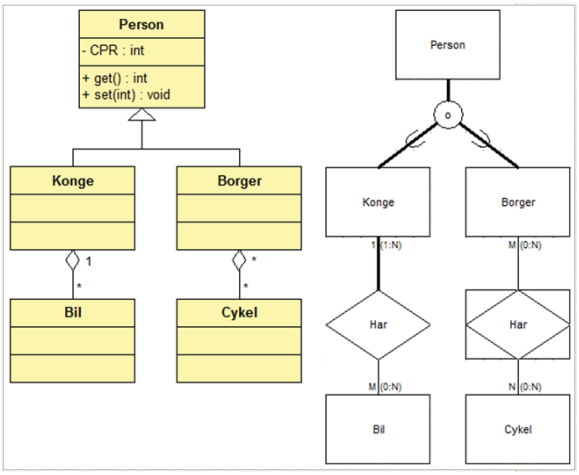
I en objekt-orienteret model anvender man objekter til at gemme og manipulere data på. De tre faser til modellering er analyse, design og implementering. Relationer mellem objekter kan ske via komposition (”har en”-relation), association (”anvender”-relation) og arv (”er en”-relation).

Klasser i den objekt-orienteret model kan mappes til en tabel, hvor hver attribut i klassen er kolonnen i tabellen.

Det er muligt at mappe et klasse objekt til relationel database. Heri bestemmes der hvilke attributter der skal være i OO-model og hvilke relationer et objekt har til andre objekter. Disse attributter og relationer kan oftest direkte konverteres til entiter med relationer imellem bestemt af fremmednøgler. Figur 1 giver et godt overblik over dette.

Entity Framework

Lader os lave en objektorienteret model, som automatisk bliver lavet til en relationel database uden SQL-statements



Figur 1: forskelle på diagrammerne