# Dette bilag er en del af inspirationsmaterialet til faget [Erhvervsinformatik](https://emu.dk/eud/erhvervsinformatik/digital-myndiggorelse) på emu.dk og er udarbejdet som supplement til vejledningen. ”[Teknologisk handleevne og computational tankegang](https://emu.dk/eud/erhvervsinformatik/teknologisk-handleevne-og-computationel-tankegang)” er et af de tre kompetenceområder i faget. Data er grundlæggende for it-systemer, og derfor bør eleverne få en forståelse for hvilke typer af data it-systemer anvender, og hvordan data repræsenteres og er modelleret. Nedenfor finder du forslag til hvordan der kan arbejdes med databaser i undervisningen.

Databaser

En **database** er enorganiseret samling af data. Databaser udnyttes til utallige formål på alle størrelser computere fra store centrale maskiner til pc'er og lagres normalt på harddiske. De kan findes lokalt på brugerens maskine eller nås ved kommunikation over netværk. En database kan være distribueret og dubleret på geografisk spredte maskiner af hensyn til sikkerhed og tilgængelighed.

En stor del af internettets sider viser data, som er trukket ud af databaser ved søgning eller anden form for databehandling.

Det er forholdsvist simpelt at arbejde med databaser i fx SQL og lade eleverne indsætte, ændre, slette og udtrække data i en simpel database.

Tabellen er databasens grundelement, det er her data gemmes. Man skelner grundlæggende mellem to typer af databaser, den flade database og relationsdatabasen. I den flade database gemmes al data i én stor tabel, men den i relationsdatabasen gemmes i flere, mindre tabeller som så kobles sammen ved hjælp af relationer, deraf navnet.

Moderne databaser er af typen relationsdatabaser og eksempler på disse er Oracle, DB2, MsSQL og MySQL.

For at nå frem til hvor mange, og hvilke tabeller der skal være i databasen benytter man E/R diagrammer og Normalisering.

### E/R diagrammer

Et E/R diagram består af to elementtyper, “Entiteten”, der illustreres med et kvadrat og “Relationen” mellem disse, der illustreres med en rombe.



N.B: E/R diagrammer benytter nogle af de samme geometriske former som flowcharts, der også gennemgås i værktøjssamlingen, men skal ikke forveksles med disse.

Et eksempel på en “Entitet” kan være “Medlem”, som repræsenterer oplysninger om de enkelte medlemmer. Hvis denne Entitet skulle bruges som udgangspunkt for en tabel, ville den komme til at indeholde felter som “kundenummer”, “kundenavn”, “kundeadresse”, “kundepostnummer”,  “kundeby”, “kundetelefon”.

### Normalisering

Materialet nedenfor er udformet af Thomas Jensen og hentet på [***http://www.vidas.dk/mysql/MySQL\_database\_normalisering.html***](http://www.vidas.dk/mysql/MySQL_database_normalisering.html) under følgende licens [***CC BY-NC-SA 4.0***](http://creativecommons.org/licenses/by-nc-sa/4.0/).  Du er velkommen til at anvende materialet og til at videreudvikle på de guides der er tilgængelige her, så længe du krediterer forfatteren, indikerer de ændringer, du har lavet, at det ikke er kommercielt og at materialet bliver delt på samme vilkår.

En database er grundlæggende en samling data der er organiseret i rækker og kolonner. I rækkerne (kaldes også en *post*) har vi de enkelte opslag og i kolonnerne de kategorier opslaget skal indeholde. Herunder er et skema over mentorfordelingen på en mindre skole.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mentor** | **Klasse 1** | **Klasse 2** | **Klasse 3** |
| Thomas Jensen (tj) | 1a (17 elever) | 2a (22 elever) | 3a (21 elever) |
| Bent Andersen (ba) | 1b (16 elever) | 2b (24 elever) | 3b (22 elever) |
| Christina Carlsen (cc) | 1x (28 elever) | 2x (31 elever) |  -  |

Skemaet ovenfor er let at læse for os, men det for at gøre det til effektiv til en database skal vi anvende *normalisering*. Normaliseringsprocessen sikrer at data ikke bliver gentaget flere steder og data er entydig. Det man vil undgå er *redundans* (gentagelser) og *inkonsistens* (mangel på sammenhæng).

For at undgå redundans og inkonsistens gennemløber man i databasens designfase disse tre trin:

**Normalform 1 (NF1)**

* Tabellen har et nøglefelt (behøver ikke at være unikt)
* Der må kun være en værdi af samme type i hver post
* Alle poster skal være lige lange dvs. have samme antal felter

**Normalform 2 (NF2)**

* Tabellen opfylder NF1
* Der skal være en nøgle i hver tabel, der entydigt bestemmer indholdet. Kaldes en unik nøgle.

**Normalform 3 (NF3)**

* Tabellen opfylder NF2
* Hvis der er mere end et felt der kan sættes som nøgle for andre felter skal tabellen opdeles i flere.

### Normalform 1 (NF1)

* Tabellen har et nøglefelt (behøver ikke at være unikt)
* Der må kun være en værdi af samme type i hver post
* Alle poster skal være lige lange dvs. have samme antal felter

Lige nu er skemaet menneskelæseligt, men for at få tabellen til at opfylde NF1 skal vi have ændret på nogle ting.

* Vi gør lærerens initialer til nøglefelt
* Vi inddeler tabellen i 4 kolonner (Initialer, Mentor, Klasse, Antal elever)

Resultatet herunder opfylder de 3 krav til NF1:

* Der er et nøglefelt
* Der er kun en værdi af samme type i hver post
* Alle poster er på 4 felter

|  |
| --- |
| Tabel 1 - Mentorer, klasser og elever |
| Initialer | Mentor | Klasse | Antal elever |
| tj | Thomas Jensen | 1a | 17 |
| tj | Thomas Jensen | 2a | 22 |
| tj | Thomas Jensen | 3a | 21 |
| ba | Bent Andersen | 1b | 16 |
| ba | Bent Andersen | 2b | 24 |
| ba | Bent Andersen | 3b | 22 |
| cc | Christina Carlsen | 1x | 28 |
| cc | Christina Carlsen | 2x | 31 |

### Normalform 2 (NF2)

* Tabellen opfylder NF1
* Der skal være en nøgle i hver tabel, der entydigt bestemmer indholdet. Kaldes en unik nøgle.

For at opfylde NF2 skal tabellen først opfylde NF1, men derudover skal tabellen have en unik nøgle. Lærerens initialer og navn er ikke unik da samme lærer kan have flere klasser. Herunder vælger vi at klassenavnet skal være den unikke nøgle

|  |
| --- |
| Tabel 1 - Klasser, mentorer og elever |
| Klasse (Nøgle) | Mentor | Navn | Antal elever |
| 1a | tj | Thomas Jensen | 17 |
| 2a | tj | Thomas Jensen | 22 |
| 3a | tj | Thomas Jensen | 21 |
| 1b | ba | Bent Andersen | 16 |
| 2b | ba | Bent Andersen | 24 |
| 3b | ba | Bent Andersen | 22 |
| 1x | cc | Christina Carlsen | 28 |
| 2x | cc | Christina Carlsen | 31 |

### Normalform 3 (NF3)

* Tabellen opfylder NF2
* Hvis der er mere end et felt der kan sættes som nøgle for andre felter skal tabellen opdeles i flere.

NF 2 sikrer, at vi kan tilgå tabellen igennem en unik nøgle og at den er sammenhængende. Men den bliver hurtig uoverskuelig. Derfor vælger vi at analysere tabellen for om den kan splittes op i mindre tabeller, hvor felter der afhænger af andre får deres egne tabeller.

Her vælger vi at opdele tabellen i 3:

* En tabel der angiver den mentor der er tilknyttet klassen
* En tabel med informationer om mentoren
* En tabel med informationer om klassen

|  |
| --- |
| Tabel 1 - Klasser og mentorer |
| Klasse (Primær nøgle) | Mentor |
| 1a | tj |
| 2a | tj |
| 3a | tj |
| 1b | ba |
| 2b | ba |
| 3b | ba |
| 1x | cc |
| 2x | cc |

og

|  |
| --- |
| Tabel 2 - Mentorer |
| Mentor (Primær nøgle) | Navn |
| tj | Thomas Jensen |
| ba | Bent Andersen |
| cc | Christina Carlsen |

og

|  |
| --- |
| Tabel 3 - Klasser og elever |
| Klasse (Primær nøgle) | Antal elever |
| 1a | 17 |
| 2a | 22 |
| 3a | 21 |
| 1b | 16 |
| 2b | 24 |
| 3b | 22 |
| 1x | 28 |
| 2x | 31 |

Fordelen ved at gøre dette er, at vi nu kan tilføje data til tabellerne uden at bryde sammenhængen mellem dem. Vi kan f.eks. udvide tabel 2 og 3 med flere informationer om mentorerne.

|  |
| --- |
| Tabel 2 (udvidet) - Mentorer |
| Mentor (Primær nøgle) | Navn | Lokale | Mobil | Mail |
| tj | Thomas Jensen | 17 | 23344517 | tj@fakemail.dk |
| ba | Bent Andersen | 18 | 23344518 | ba@fakemail.dk |
| cc | Christina Carlsen | 19 | 23344519 | cc@fakemail.dk |

|  |
| --- |
| Tabel 3 (udvidet)- Klasser og elever |
| Klasse (Primær nøgle) | Antal elever | Piger | Drenge | Startår |
| 1a | 17 | 9 | 8 | 2016 |
| 2a | 22 | 10 | 12 | 2015 |
| 3a | 21 | 8 | 13 | 2014 |
| 1b | 16 | 8 | 8 | 2016 |
| 2b | 24 | 13 | 11 | 2015 |
| 3b | 22 | 9 | 13 | 2014 |
| 1x | 28 | 13 | 15 | 2016 |
| 2x | 31 | 17 | 14 | 2015 |

Ved at anvende normalisering på vores database får vi en opbygning der sikrer at vi ikke har redundans (gentagelser) og inkonsistens (mangel på sammenhæng) i de enkelte tabeller. Det sikrer også, at tabellen bliver nemmere at vedligeholde og udvide med flere informationer.
Hver eneste gang man udvider sin database, bør man sikre sig, at databasen overholder de tre normaliseringskrav.